

**Handbuch für den BSB-LPB-LAN-Adapter und  
die BSB-LAN-Software - druckerfreundliche  
Version der Seite  
<https://1coderookie.github.io/BSB-LPB-LAN>**

# Inhaltsverzeichnis

[Zurück zur Einleitung](#)

## Inhaltsverzeichnis

*User des veralteten Setups Adapter v2 + Mega2560: Bitte Anhang D beachten!*

### Einleitung

#### Schnellstartanleitung für den Arduino Due

#### Schnellstartanleitung für ESP32-Boards

### 1. BSB-LAN: Die Hardware

#### 1.1 Adapter

##### 1.1.1 Due-Version

##### 1.1.2 ESP32-Version

#### 1.2 Arduino Due

##### 1.2.1 Due + LAN: Das LAN-Shield

##### 1.2.2 Due + WLAN: Die ESP8266-WiFi-Lösung

#### 1.3 ESP32

##### 1.3.1 ESP32 mit spezifischem "BSB-LAN ESP32"-Adapter

###### 1.3.1.1 ESP32: NodeMCU "Joy-It"

###### 1.3.1.2 ESP32: Olimex ESP32-EVB & ESP32-PoE

##### 1.3.2 ESP32 mit Due-kompatiblen BSB-LAN-Adapter ab V3

##### 1.3.3 ESP32 mit veraltetem BSB-LAN-Adapter V2

#### 1.4 Raspberry Pi

#### 1.5 Gehäuse

### 2. BSB-LAN: Die Software

#### 2.1 Installation

##### 2.1.1 Installation auf dem Due

##### 2.1.2 Installation auf dem ESP32

##### 2.1.3 Updates

#### 2.2 Konfiguration

##### 2.2.1 Konfiguration mittels Webinterface

##### 2.2.2 Konfiguration durch Anpassen der Datei `BSB_LAN_config.h`

#### 2.3 Manuelles Hinzufügen von Parametern aus v2.2

### 3. BSB-LAN Setup: Anschluss und Inbetriebnahme

#### 3.1 Anschluss des Adapters

#### 3.2 Funktionsüberprüfung und erste Nutzung

#### 3.3 Reglerspezifische Parameterliste erstellen

#### 3.4 Debugging und Fehlersuche

### 4. BSB-LAN: Das Webinterface

### 5. BSB-LAN: Abfragen und Steuern

#### 5.1 URL-Befehle

## 5.2 MQTT

## 5.3 JSON

## 5.4 Spezialparameter & Nummernbereiche

# 6. BSB-LAN: Spezialfunktionen

## 6.1 Loggen von Daten

## 6.2 IPWE-Erweiterung

## 6.3 Raumtemperatur übermitteln

## 6.4 Präsenztaste simulieren

## 6.5 Manuellen TWW-Push ausführen

## 6.6 Datum, Uhrzeit und Zeitprogramme verändern

## 6.7 Übermitteln einer alternativen Außentemperatur

## 6.8 Eigenen Code in BSB-LAN einbinden

## 6.9 Verwenden der Webserver-Funktion

## 6.10 Benutzen des alternativen AJAX Webinterface

## 6.11 Raumgerät-Emulation

## 6.12 EEPROM-Löschung mittels Pinkontakten

# 7. BSB-LAN Setup: Optionale Hardware

## 7.1 Verwendung optionaler Sensoren: DHT22, DS18B20, BME280

### 7.1.1 Hinweise zu DHT22-Temperatur-/Feuchtigkeitssensoren

### 7.1.2 Hinweise zu DS18B20-Temperatursensoren

### 7.1.3 Hinweise zu BME280-Sensoren

## 7.2 Relais und Relaisboards

## 7.3 MAX!-Komponenten

## 7.4 Eigene Hardwarelösungen

### 7.4.1 Raumgerätersatz (Arduino Uno, LAN-Shield, DHT22, Display, Taster)

### 7.4.2 Raumtemperaturfühler (Wemos D1 mini, DHT22, Display)

### 7.4.3 Raumgerätersatz mit UDP-Kommunikation

#### 7.4.3.1 UDP mit Arduino Due + LAN-Shield

#### 7.4.3.2 UDP mit ESP32

### 7.4.4 Xiaomi Mijia BLE Sensoren

## 7.5 LAN-Optionen für das BSB-LAN-Setup

### 7.5.1 Nutzung eines PowerLANs / dLANs

### 7.5.2 WLAN: Nutzung eines extra Routers

# 8. BSB-LAN: Einbindung mittels zusätzlicher Software

## 8.1 FHEM

## 8.2 openHAB

### 8.2.1 openHAB-Binding

### 8.2.2 openHAB mit Javascript Transformation

### 8.2.3 openHAB mit Javascript Transformation, MQTT, Network und Expire

## 8.3 HomeMatic (EQ3)

## 8.4 ioBroker

## 8.5 Loxone

## 8.6 IP-Symcon

## 8.7 MQTT, InfluxDB, Telegraf und Grafana

## 8.8 MQTT und FHEM

## 8.9 MQTT2 und FHEM

- 8.10 EDOMI
- 8.11 Home Assistant
- 8.12 SmartHomeNG
- 8.13 Node-RED
- 8.14 Datenverarbeitung mittels Bash-Skript
- 8.15 Volkszaehler
- 8.16 Homebridge
- 8.17 Jeedom

## 9. Exkurs: Auslesen neuer Parameter-Telegramme

## 10. Exkurs: Heizungsregler und Zubehör

- 10.1 Bussysteme der Heizungsregler: BSB, LPB, PPS
  - 10.1.1 BSB
  - 10.1.2 LPB
  - 10.1.3 PPS-Schnittstelle
- 10.2 Detaillierte Beschreibung der kompatiblen Regler
  - 10.2.1 LMx-Regler
    - 10.2.1.1 LMU-Regler
    - 10.2.1.2 LMS-Regler
  - 10.2.2 RVx-Regler
    - 10.2.2.1 RVA- und RVP-Regler
    - 10.2.2.2 RVS-Regler
  - 10.2.3 Hinweis: Inkompatible Systeme von Brötje und Elco
  - 10.2.4 Hinweis: Spezialfall LMU54/LMU64-Regler
  - 10.2.5 Hinweis: Spezialfall Weishaupt-Geräte
  - 10.2.6 Hinweis: LPB nachrüsten mittels OCI420 ClipIn-Modul
- 10.3 Erweiterungs- und ClipIn-Module
- 10.4 Bedieneinheiten
- 10.5 Konventionelle Raumgeräte für die aufgeführten Reglertypen
  - 10.5.1 QAA55 / QAA58
  - 10.5.2 QAA75 / QAA78
  - 10.5.3 QAA74
  - 10.5.4 Brötje IDA
  - 10.5.5 QAA53 / QAA73
  - 10.5.6 QAA50 / QAA70
- 10.6 Sonderzubehör: Webserver OZW672 und Servicetool OCI700
- 10.7 Fühlertypen

## 11. Exkurs: Erfolgreich getestete Heizungssysteme

- 11.1 Brötje
- 11.2 Elco
- 11.3 Weitere Hersteller

## 12. Exkurs: Arduino IDE

- 12.1 Installation
  - 12.1.1 Arduino Due
  - 12.1.2 ESP32
- 12.2 Serieller Monitor

## 13. Exkurs: BSB-LAN sicher aus dem Internet heraus erreichen



## 14. Etwaige Fehlermeldungen und deren mögliche Ursachen

- 14.1 Fehlermeldung "unknown type xxxxxxxx"
- 14.2 Fehlermeldung "error 7 (parameter not supported)"
- 14.3 Fehlermeldung "query failed"
- 14.4 Fehlermeldung "FEHLER: Setzen fehlgeschlagen! - Parameter ist nur lesbar"
- 14.5 Fehlermeldung "decoding error"

## 15. Etwaige Probleme und deren mögliche Ursachen

- 15.1 Arduino IDE stoppt beim Kompilieren
- 15.2 Die rote LED des Adapters leuchtet nicht
- 15.3 Die rote LED leuchtet, aber es ist keine Abfrage möglich
- 15.4 Zugriff auf das Webinterface nicht möglich
- 15.5 Keine Verbindung zum WLAN möglich
- 15.6 Keine Parameterabfrage möglich
- 15.7 Regler wird nicht korrekt erkannt
- 15.8 HK1 kann nicht bedient werden
- 15.9 Es kann keine Raumtemperatur an einen HK1 gesendet werden
- 15.10 HK2 kann nicht bedient werden
- 15.11 Es kann keine Raumtemperatur an einen HK2 gesendet werden
- 15.12 Einstellungen des Reglers können nicht via Adapter verändert werden
- 15.13 Der Adapter reagiert manchmal nicht auf Abfragen oder SET-Befehle
- 15.14 Bei der Abfrage der Logdatei passiert ‚nichts‘
- 15.15 Es werden keine 24h-Durchschnittswerte angezeigt
- 15.16 Bei der Abfrage der Daten von DS18B20-/DHT22-Sensoren passiert ‚nichts‘
- 15.17 Die DS18B20-Sensoren zeigen falsche Werte an
- 15.18 Der ‚Serielle Monitor‘ der Arduino IDE liefert keine Daten

## 16. FAQ

- 16.1 Kann ich Adapter & Software mit einem Raspberry Pi nutzen?
- 16.2 Kann ich einen Adapter gleichzeitig an zwei Regler anschließen?
- 16.3 Kann ich einen Adapter via LPB anschließen und mehrere Regler abfragen?
- 16.4 Ist ein multifunktionaler Eingang des Reglers direkt via Adapter schaltbar?
- 16.5 Ist zusätzlich ein Relaisboard am Arduino anschließ- und steuerbar?
- 16.6 Kann ich bspw. den Zustand eines angeschlossenen Koppelrelais abfragen?
- 16.7 Kann ich behilflich sein, um bisher nicht unterstützte Parameter hinzuzufügen?
- 16.8 Warum erscheinen bei einer Komplettabfrage einige Parameter doppelt?
- 16.9 Warum werden manchmal bestimmte Parameter nicht angezeigt?
- 16.10 Warum ist kein Zugriff auf angeschlossene Sensoren möglich?
- 16.11 Ich nutze ein W5500-LAN-Shield, was muss ich tun?
- 16.12 Können Stati oder Werte als Push-Mitteilungen abgesetzt werden?
- 16.13 Kann bspw. FHEM auf bestimmte Broadcasts ‚lauschen‘?
- 16.14 Warum kommt es manchmal zu timeout-Problemen bei FHEM?
- 16.15 Gibt es ein Modul für FHEM?
- 16.16 Warum werden unter /B bei Stufe 2 keine Werte angezeigt?
- 16.17 Ich habe den Eindruck, die angezeigten Werte bei /B sind nicht korrekt.
- 16.18 Was ist der genaue Unterschied zwischen /M1 und /V1?
- 16.19 Kann ich eigenen Code in BSB-LAN einbinden?

- 16.20 Kann ich MAX!-Thermostate einbinden?
- 16.21 Warum ist der Adapter nach einem Stromausfall nicht mehr erreichbar?
- 16.22 Warum ist der Adapter (ohne Stromausfall) manchmal nicht mehr erreichbar?
- 16.23 Warum kommen beim Senden manchmal ‚query failed‘-Meldungen?
- 16.24 Ich finde keinen LPB- oder BSB-Anschluss, nur L-BUS und R-BUS?!
- 16.25 Gibt es eine (W)LAN-Option für den Adapter?
- 16.26 Ich nutze das veraltete Setup Adapter v2 + Arduino Mega 2560 - muss ich irgendetwas beachten?
- 16.27 Ich bekomme Fehlermeldungen von der Arduino IDE, was kann ich tun?
- 16.28 Es kann keine Verbindung zum WLAN-Netzwerk hergestellt werden
- 16.29 BSB-LAN stürzt häufig ab oder die WLAN-Verbindung ist instabil.
- 16.30 Ich finde die Einstellung zum Schreiben der Parameter nicht / Es fehlt der „set“ Button
- 16.31 Warum finde ich nirgendwo den Parameter XYZ?
- 16.32 Die LED flackert, aber es kommt keine Verbindung zur Heizung zustande.
- 16.33 Ich habe weitere Fragen, an wen kann ich mich wenden?

## 17. Weiterführende Informationen und Quellen

### Anhang A1: Schaltplan BSB-LPB-LAN-Adapter v4 (Due-Version)

### Anhang A2: Anmerkungen zum Schaltplan

- A2.1 Kurze Legende zum Schaltplan
- A2.2 Teileliste
- A2.3 Generelle Hinweise

### Anhang B: Pinouts

- B1 Arduino DUE
- B2 Joy-It ESP32 NodeMCU
- B3 Olimex ESP32-EVB

### Anhang C: Changelog BSB-LAN-Software

### Anhang D: Hinweise für Nutzer des veralteten Setups Adapter v2 + Arduino Mega 2560



# Zurück zur Einleitung

[English language version of this manual available!](#)

---

## Einleitung

---

Dieses Handbuch wurde geschrieben, um den Einstieg in die Benutzung der BSB-LAN Hard- & Software zu vereinfachen und um als Nachschlagewerk zu dienen.

*Es wird empfohlen, dieses Handbuch vor einer initialen Verwendung des BSB-LPB-LAN-Adapters komplett zu lesen.*

Das Copyright des Handbuchs liegt bei dem Autor Ulf Dieckmann.

Hier geht es direkt zum [Inhaltsverzeichnis](#).

Zum Ausdrucken besser geeignet: [Die PDF-Version des Handbuchs](#).

Schnellstartanleitungen für die Installation und Inbetriebnahme des BSB-LAN-Setups sind hier verfügbar:

[Schnellstartanleitung für den Arduino Due](#)

[Schnellstartanleitung für ESP32-Boards](#)

### ACHTUNG:

**Es gibt KEINE GARANTIE oder Gewährleistung jeglicher Art, dass dieser Adapter dein Heizungssystem NICHT beschädigt!**  
**Jegliche Umsetzung der hier beschriebenen Schritte, jeder Nachbau des Adapters sowie jede Verwendung der beschriebenen Hard- und Software erfolgt auf eigene Verantwortung und eigenes Risiko!**  
**Keiner der Mitwirkenden oder Autoren kann für etwaige Schäden jeglicher Art haftbar gemacht werden!**

## BSB-LPB-LAN - ein kurzer Überblick

"BSB-LPB-LAN" ist ein gemeinschaftliches Hard- und Softwareprojekt, das den Zugriff auf die Steuerungen verschiedener Wärmeerzeuger bestimmter Hersteller über PC / Laptop / Tablet / Smartphone ermöglicht.

Das Projekt besteht aus zwei spezifischen Komponenten:

- der [Hardware](#), bei der es sich im Wesentlichen um einen Pegelwandler handelt und die im Folgenden "BSB-LAN-Adapter" genannt wird und
- der [BSB-LAN Software](#), die auf einen kompatiblen Mikrocontroller geflasht werden muss.

Der [BSB-LAN-Adapter](#) wandelt die 12V-Bussignale der Heizung in ein geeignetes 3,3V-Signal für den benötigten Mikrocontroller um. Der Adapter wird an den [kompatiblen Regler](#) des Wärmeerzeugers angeschlossen und muss in Verbindung mit einem kompatiblen Mikrocontroller ([Arduino Due](#) oder [ESP32](#)) verwendet werden.

Der Mikrocontroller selbst wird dann in das Heimnetzwerk eingebunden (je nach gewähltem Mikrocontroller entweder über LAN oder WLAN). Die Steuerung der Heizungsanlage muss mit einem "[Boiler-System-Bus](#)" (BSB), einem "[Local-Process-Bus](#) (LPB) oder einer "[Punkt-zu-Punkt-Schnittstelle](#)" (PPS) ausgestattet sein. Dies sind in i.d.R. Systeme, bei denen ein (gebrandeter) SIEMENS-Regler zum Einsatz kommt.

Die [BSB-LAN-Software](#) setzt dann die Logikpegel in spezifische 'Bustelegramme' um. Sie ermöglicht im Wesentlichen den Zugriff auf den Regler der Heizungsanlage. Sie bietet verschiedene Funktionen wie bspw. das Monitoring oder Loggen von Werten (bspw. Temperaturen) und Zuständen (bspw. Pumpenstatus, TWW-Ladestatus) via spezifischer Parameter des Heizungsreglers und (falls gewünscht) die Steuerung und Änderung von Einstellungen über ein [Webinterface](#).

Eine optionale Einbindung in ein bestehendes SmartHome-System ist ebenfalls möglich. Eine Integration in Systeme wie [FHEM](#), [openHAB](#), [HomeMatic](#), [ioBroker](#), [Loxone](#), [IP-Symcon](#), [EDOMI](#), [Home Assistant](#), [SmartHomeNG](#) oder [Node-RED](#) kann mittels [HTTPMOD](#), [MQTT](#) oder [JSON](#) erfolgen. Darüber hinaus ist der Einsatz des Adapters als [Standalone-Logger](#) ohne LAN-/WLAN- oder Internetanbindung bei Verwendung einer microSD-Karte ebenfalls möglich.

Zusätzlich können [Temperatur- und Feuchtigkeitssensoren](#) angeschlossen und deren Daten ebenso geloggt und ausgewertet werden. Außerdem besteht die Möglichkeit, [eigenen Code in die BSB-LAN-Software zu integrieren](#), was darüber hinaus ein weites Spektrum an Erweiterungsmöglichkeiten bietet.

Als erste grobe Orientierung, ob das eigene Heizungssystem kompatibel ist oder nicht, kann in der Bedienungsanleitung der Heizung nach einer Anschlussmöglichkeit für optionale Raumgeräte gesucht werden.

Sind dort Raumgeräte des Typs QAA55/QAA75 als kompatibel aufgeführt (bei Brötje werden diese u.a. auch als "RGB Basic" und "RGT B Top" bezeichnet), so ist erfahrungsgemäß der Anschluss des Adapters via BSB möglich und der volle Funktionsumfang von BSB-LAN gegeben. Dies ist bei den meisten Öl-, Gas- und Wärmepumpensystemen der letzten Jahre der Fall.

Sollten andere Raumgeräte aufgeführt sein, so kann im Kapitel "[Raumgeräte](#)" nachgesehen werden.

Genauen Aufschluss bietet letztlich aber immer nur die eigentliche Reglerbezeichnung.

Die folgende Auflistung gibt eine grobe Übersicht über die Reglertypen, die je nach Typ des Wärmeerzeugers (Öl, Gas, WP etc.) normalerweise verbaut sind (bzw. waren) und die mittels BSB-LAN bedient werden können. Gewisse Einzel- und Spezialfälle (wie bspw. ein RVS-Regler bei einem Gasgerät) sind hier nicht berücksichtigt. Für genauere Informationen bzgl der [Reglertypen](#) und der zu verwendenden [Anschlüsse](#) lies bitte die entsprechenden Kapitel.

**Um eine detailliertere Übersicht der gemeldeten Systeme einzusehen, die bisher erfolgreich mit BSB-LAN genutzt werden, folge bitte dem entsprechenden Link:**

- [Brötje](#)
- [Elco](#)
- [weitere Hersteller \(z.B. Fujitsu, Atlantic, Weishaupt\)](#)

#### Gasregler:

- [LMU74/LMU75](#) und (aktuelle Generation) [LMS14/LMS15](#), Anschluss via BSB
- [LMU54/LMU64](#), Anschluss via PPS

#### Öl-/Solar-/Zonenregler:

- [RVS43/RVS63/RVS46](#), Anschluss via BSB
- [RVA/RVP](#), Anschluss via PPS (modellspezifisch vereinzelt auch LPB)

#### Wärmepumpenregler:

- [RVS21/RVS61](#), Anschluss via BSB

#### Weishaupt (Modell WTU):

- [RVS23](#), Anschluss via LPB

**Die Software ist [hier](#) verfügbar.**

#### Autoren:

- Software, Schaltplan v1, urspr. Dokumentation EN, Ideenfindung, Support  
bis v0.16:  
*Gero Schumacher*
- Software, Platinenlayout v1 & v2, urspr. Dokumentation EN, Ideenfindung, Support  
ab v0.17:  
*Frederik Holst (bsb [ät] code-it.de)*
- Debugging, Handbuch, Übersetzungen, Ideenfindung, Support  
ab v0.17:  
*Ulf Dieckmann (adapter [ät] quantentunnel.de)*

*Basierend auf dem Code und der Mitarbeit von vielen anderen Entwicklern! Vielen Dank!*





# User des veralteten Setups Adapter v2 + Mega2560: Bitte Anhang D beachten!

[Zurück zum Inhaltsverzeichnis](#)

[Zurück zu Anhang C](#)

## Anhang D: Hinweise für Nutzer des veralteten Setups Adapter v2 + Arduino Mega 2560

Für Nutzer des veralteten Setups sind im Folgenden einige Fragen und Punkte aufgeführt, die evtl. der Klärung bedürfen oder die es zu beachten gilt. Etwaige weitere Fragen diesbzgl. stelle bitte im entspr. [Thread des FHEM Forums](#).

Bitte habe jedoch Verständnis, dass wir nicht auf Fragen eingehen werden, die sich bspw. darauf beziehen, sich nach der erfolgten Umstellung auf den Adapter v3/v4 jetzt noch einen Adapter v2 zu bauen.

**PCBs v2 sind nicht mehr verfügbar, Stand der Technik ist die Kombination Adapter v4.x + Due/ESP32.**

### Hinweis:

*Es ist möglich, den Adapter v2 durch eine Vollbestückung und kleinere Anpassungen mit einem ESP32 zu verwenden. Auf diese Weise könnte die aktuelle BSB-LAN-Version genutzt werden, ohne auf den aktuellen Adapter wechseln zu müssen. Für weitere Informationen lies bitte das [Kap. 1.3.3](#).*

- **Muss ich zwingend auf das neue Setup Adapter v3/v4 + Due wechseln?**

Nein, wenn du zufrieden mit dem veralteten Setup bist und der Funktionsumfang von BSB-LAN deinen Ansprüchen bisher genügt, dann kannst du das alte Setup natürlich weiterhin verwenden.

- **Gibt es bzgl. der BSB-LAN-Versionen etwas zu beachten?**

Ja. Die letzte 'offiziell' getestete und empfohlene Version für dein Setup ist die [Version v0.44](#). Im zip-file befindet sich auch die letzte 'Mega2560-spezifische' Version des Handbuchs (als PDF).

Aber: Es hat sich bei mehreren Usern gezeigt, dass auch die [v1.1](#) noch ohne große Einschränkungen läuft, aufgrund des Speichermangels des Mega 2560 vermutlich aber schon nicht mehr mit allen verfügbaren Optionen, die BSB-LAN bietet.

Ab **v2.x** ist es dann definitiv nötig, einzelne Module zu deaktivieren und somit auf spezifische Funktionen zu verzichten, die BSB-LAN bietet. Hinweise diesbzgl. findest du in [Kap. 2.2.2](#) bzw in den Kommentaren der Datei `BSB_LAN_config.h`. Besonderes Augenmerk ist auf die letzten Punkte zu richten, die u.a. ein komfortables Deaktivieren einzelner Module (bspw. Webconfig, MQTT, IPWE etc.) an zentraler Stelle ermöglicht.

- **Was gilt es zu beachten, wenn ich die aktuelle v2.x nutzen möchte?**

Wie bereits erwähnt muss die v2.x in der Konfiguration prinzipiell so angepasst werden, dass sie mit dem geringeren Speicher des Mega problemlos kompiliert und lauffähig ist. Neben dem bereits erwähnten Deaktivieren einzelner Module gibt es weitere Möglichkeiten:

1.) Die Größe der Variablen von bspw. `PASSKEY[]`, `avg_parameters[]`, `log_parameters[]`, `ipwe_parameters[]`, `max_device_list[]` kann verkleinert werden (wenn bspw. weniger Parameter als maximal möglich verwendet werden), um ein wenig Speicher einzusparen.

2.) In der Datei `BSB_LAN_config.h` finden sich im unteren Abschnitt verschiedene Definements für Mega-User, die bei bestimmten Einsatzszenarien aktiviert werden können, um nochmals Speicher zu sparen. Hinweise diesbzgl. findest du in der Datei selbst.

3.) Erstellung einer reglerspezifischen `BSB_LAN_defs.h`:

Im Repo liegt ein Perlscript namens `selected_defs.pl` sowie ein Windows-Executable namens `selected_defs.exe`, das die Datei `BSB_LAN_defs.h` nach ausgewählten Gerätefamilien filtert und eine spezifische Datei für den eigenen Reglertyp erstellt. Die Ersparnis beträgt im Schnitt etwa 20 bis 25 kB Flash-Speicher, den man dann für die (Re-)Aktivierung von anderen Funktionen nutzen kann. Im Falle eines Reglerwechsels (= andere Gerätefamilie) muss die Datei natürlich entsprechend neu generiert werden.

Das Script läuft unter Perl, was auf Mac- und Linux-Rechnern standardmäßig installiert ist, lässt sich aber auch auf Windows nachinstallieren.

Vorgehensweise zur Erstellung einer reglerspezifischen defs-Datei:

- Parameter 6225 "Gerätefamilie" via BSB-LAN abrufen und den Wert notieren.
- Datei `selected_defs.pl` bzw. `selected_defs.exe` vor dem Ausführen in den gleichen Ordner kopieren, in dem auch die Datei

*BSB\_LAN\_defs.h* liegt.

- Öffne ein Terminal, wechsele in den entspr. Ordner und erstelle die reduzierte Datei namens *BSB\_LAN\_defs\_filtered.h* mit Hilfe des Perlscripts bzw. des Windows-Executables, die nur die für die spezifische Gerätefamilie(n) relevanten Parameter enthält. Bei nur einem angeschlossenen Regler, bspw. mit der Gerätefamilie 162, lautet der Befehl

```
./selected_defs.pl 162 > BSB_LAN_defs_filtered.h bzw.
```

```
selected_defs.exe 162 > BSB_LAN_defs_filtered.h .
```

Wenn man bspw. zwei Geräte am Bus mit den Gerätefamilien 162 und 90 hat, kann man den Befehl um den zweiten Wert erweitern:

```
./selected_defs.pl 162 90 > BSB_LAN_defs_filtered.h bzw.
```

```
selected_defs.exe 162 90 > BSB_LAN_defs_filtered.h .
```

- Verschiebe die originale Datei *BSB\_LAN\_defs.h* aus dem "BSB\_LAN"-Verzeichnis an einen beliebigen Ort. Verschiebe dann die neu erzeugte Datei *BSB\_LAN\_defs\_filtered.h* in das Verzeichnis "BSB\_LAN" (falls du die Datei nicht bereits im Ordner "BSB\_LAN" erstellt hast).
- Wichtig: Die neu erzeugte Datei nun in "*BSB\_LAN\_defs.h*" umbenennen!

- **Gibt es bzgl. der zu verwendenden Pineinstellungen etwas zu beachten?**

Ja! Solltest du eine neuere Version als v0.44 auf dem Mega testen wollen, so achte darauf, dass du die zur jeweiligen Version zugehörige Datei *BSB\_LAN\_config.h.default* verwendest und entsprechend anpasst:

- Bei BSB-LAN-Versionen **vor v2.x** ist die Anpassung der Zeile `BSB_bus(19,18);` zwingend notwendig: Der DUE verwendet (im Gegensatz zum Mega) die HardwareSerial-Schnittstelle und andere RX-/TX-Pins als der Mega, was hier bereits voreingestellt ist. Bei Verwendung mit dem Mega muss die Zeile daher in `BSB_bus(68,69);` geändert werden!
- Bei BSB-LAN-Versionen **ab v2.x** ist in der Datei *BSB\_LAN\_config.h* eine automatische Erkennung der verwendeten Pins voreingestellt. Somit wird automatisch erkannt, ob ein Mega (= software serial) oder ein Due (= hardware serial) zum Einsatz kommt.

- **Warum gibt es überhaupt einen Umstieg auf den Due?**

Der Mega 2560 bot einfach nicht mehr genügend Speicher, um auch in Zukunft das stetig wachsende BSB-LAN zu beherbergen! ;)

- **Warum gibt es überhaupt einen neuen Adapter v3/v4?**

Das war nötig, da der bisherige Adapter v2 aus verschiedenen Gründen nicht kompatibel mit dem Due ist.

- **Kann ich den Adapter v2 an einem Due weiterverwenden?**

Nein! Der Grund dafür liegt primär darin, dass sowohl der Adapter v2 als auch der Due kein EEPROM aufweist, was für BSB-LAN jedoch notwendig ist.

Möchtest du also auch in Zukunft von den neuen Funktionen von BSB-LAN profitieren, musst du dir einen Adapter v3 besorgen oder selbst herstellen und ihn mit einem Arduino Due verwenden.

- **Kann ich den Adapter v2 zu einem Adapter v3/v4 'umbauen'?**

Nein! Der primäre Grund hierfür liegt (neben weiteren Gründen) auch wieder im fehlenden EEPROM des Due.

- **Kann ich den Adapter v3/v4 mit meinem bisherigen Mega 2560 weiterverwenden?**

Nein! Auch wenn es vielleicht nach gewissen Änderungen am Adapter v3/v4 möglich wäre, so würde es keinerlei Mehrwert gegenüber dem Adapter v2 bieten. Neue Funktionen von BSB-LAN würden aufgrund des mangelnden Speicher des Mega 2560 trotzdem nicht genutzt werden können. Wenn du also den neuen Adapter v3/v4 einsetzen möchtest, dann nur in Verbindung mit einem Arduino Due.

- **Warum ist auf der Platine v3/v4 ein EEPROM?**

Der Arduino Due weist kein EEPROM auf, was jedoch für BSB-LAN notwendig ist.

- **Kann ich das LAN-Shield bei einem Umstieg auf den Due weiterverwenden?**

Ja, das ist normalerweise problemlos möglich.

- **Kann ich mein bestehendes Gehäuse weiterverwenden?**

Jein. Der Due weist prinzipiell den gleichen Formfaktor auf wie der Mega 2560, insofern sollte das Gehäuse von den Abmessungen her passen. Allerdings musst du vermutlich dein Gehäuse etwas anpassen und einen Ausschnitt oder eine große Bohrung für den mittleren USB-Port des Due ('Programming Port') hinzufügen, damit du auch weiterhin bequem das entspr. USB-Kabel anschließen kannst.





# Einleitung

## English language version of this manual available!

---

## Einleitung

---

Dieses Handbuch wurde geschrieben, um den Einstieg in die Benutzung der BSB-LAN Hard- & Software zu vereinfachen und um als Nachschlagewerk zu dienen.

*Es wird empfohlen, dieses Handbuch vor einer initialen Verwendung des BSB-LPB-LAN-Adapters komplett zu lesen.*

Das Copyright des Handbuchs liegt bei dem Autor Ulf Dieckmann.

Hier geht es direkt zum [Inhaltsverzeichnis](#).

Zum Ausdrucken besser geeignet: [Die PDF-Version des Handbuchs](#).

Schnellstartanleitungen für die Installation und Inbetriebnahme des BSB-LAN-Setups sind hier verfügbar:

[Schnellstartanleitung für den Arduino Due](#)

[Schnellstartanleitung für ESP32-Boards](#)

### ACHTUNG:

*Es gibt KEINE GARANTIE oder Gewährleistung jeglicher Art, dass dieser Adapter dein Heizungssystem NICHT beschädigt! Jegliche Umsetzung der hier beschriebenen Schritte, jeder Nachbau des Adapters sowie jede Verwendung der beschriebenen Hard- und Software erfolgt auf eigene Verantwortung und eigenes Risiko! Keiner der Mitwirkenden oder Autoren kann für etwaige Schäden jeglicher Art haftbar gemacht werden!*

## BSB-LPB-LAN - ein kurzer Überblick

"BSB-LPB-LAN" ist ein gemeinschaftliches Hard- und Softwareprojekt, das den Zugriff auf die Steuerungen verschiedener Wärmeerzeuger bestimmter Hersteller über PC / Laptop / Tablet / Smartphone ermöglicht.

Das Projekt besteht aus zwei spezifischen Komponenten:

- der [Hardware](#), bei der es sich im Wesentlichen um einen Pegelwandler handelt und die im Folgenden "BSB-LAN-Adapter" genannt wird und
- der [BSB-LAN Software](#), die auf einen kompatiblen Mikrocontroller geflasht werden muss.

Der [BSB-LAN-Adapter](#) wandelt die 12V-Bussignale der Heizung in ein geeignetes 3,3V-Signal für den benötigten Mikrocontroller um. Der Adapter wird an den [kompatiblen Regler](#) des Wärmeerzeugers angeschlossen und muss in Verbindung mit einem kompatiblen Mikrocontroller ([Arduino Due](#) oder [ESP32](#)) verwendet werden.

Der Mikrocontroller selbst wird dann in das Heimnetzwerk eingebunden (je nach gewähltem Mikrocontroller entweder über LAN oder WLAN). Die Steuerung der Heizungsanlage muss mit einem "[Boiler-System-Bus](#)" (BSB), einem "[Local-Process-Bus](#) (LPB) oder einer "[Punkt-zu-Punkt-Schnittstelle](#)" (PPS) ausgestattet sein. Dies sind in i.d.R. Systeme, bei denen ein (gebrandeter) SIEMENS-Regler zum Einsatz kommt.

Die [BSB-LAN-Software](#) setzt dann die Logikpegel in spezifische 'Bustelegramme' um. Sie ermöglicht im Wesentlichen den Zugriff auf den Regler der Heizungsanlage. Sie bietet verschiedene Funktionen wie bspw. das Monitoring oder Loggen von Werten (bspw. Temperaturen) und Zuständen (bspw. Pumpenstatus, TWW-Ladestatus) via spezifischer Parameter des Heizungsreglers und (falls gewünscht) die Steuerung und Änderung von Einstellungen über ein [Webinterface](#).

Eine optionale Einbindung in ein bestehendes SmartHome-System ist ebenfalls möglich. Eine Integration in Systeme wie [FHEM](#), [openHAB](#), [HomeMatic](#), [ioBroker](#), [Loxone](#), [IP-Symcon](#), [EDOMI](#), [Home Assistant](#), [SmartHomeNG](#) oder [Node-RED](#) kann mittels [HTTPMOD](#), [MQTT](#) oder [JSON](#) erfolgen. Darüber hinaus ist der Einsatz des Adapters als [Standalone-Logger](#) ohne LAN-/WLAN- oder Internetanbindung bei Verwendung einer microSD-Karte ebenfalls möglich.

Zusätzlich können [Temperatur- und Feuchtigkeitssensoren](#) angeschlossen und deren Daten ebenso geloggt und ausgewertet werden. Außerdem besteht die Möglichkeit, [eigenen Code in die BSB-LAN-Software zu integrieren](#), was darüber hinaus ein weites Spektrum an Erweiterungsmöglichkeiten bietet.

Als erste grobe Orientierung, ob das eigene Heizungssystem kompatibel ist oder nicht, kann in der Bedienungsanleitung der Heizung nach einer Anschlussmöglichkeit für optionale Raumgeräte gesucht werden.

Sind dort Raumgeräte des Typs QAA55/QAA75 als kompatibel aufgeführt (bei Brötje werden diese u.a. auch als "RGB Basic" und "RGT B Top" bezeichnet), so ist erfahrungsgemäß der Anschluss des Adapters via BSB möglich und der volle Funktionsumfang von BSB-LAN gegeben. Dies ist bei den meisten Öl-, Gas- und Wärmepumpensystemen der letzten Jahre der Fall.

Sollten andere Raumgeräte aufgeführt sein, so kann im Kapitel "[Raumgeräte](#)" nachgesehen werden.

Genauen Aufschluss bietet letztlich aber immer nur die eigentliche Reglerbezeichnung.

Die folgende Auflistung gibt eine grobe Übersicht über die Reglertypen, die je nach Typ des Wärmeerzeugers (Öl, Gas, WP etc.) normalerweise verbaut sind (bzw. waren) und die mittels BSB-LAN bedient werden können. Gewisse Einzel- und Spezialfälle (wie bspw. ein RVS-Regler bei einem Gasgerät) sind hier nicht berücksichtigt. Für genauere Informationen bzgl der [Reglertypen](#) und der zu verwendenden [Anschlüsse](#) lies bitte die entsprechenden Kapitel.

**Um eine detailliertere Übersicht der gemeldeten Systeme einzusehen, die bisher erfolgreich mit BSB-LAN genutzt werden, folge bitte dem entsprechenden Link:**

- [Brötje](#)
- [Elco](#)
- [weitere Hersteller \(z.B. Fujitsu, Atlantic, Weishaupt\)](#)

#### Gasregler:

- [LMU74/LMU75](#) und (aktuelle Generation) [LMS14/LMS15](#), Anschluss via BSB
- [LMU54/LMU64](#), Anschluss via PPS

#### Öl-/Solar-/Zonenregler:

- [RVS43/RVS63/RVS46](#), Anschluss via BSB
- [RVA/RVP](#), Anschluss via PPS (modellspezifisch vereinzelt auch LPB)

#### Wärmepumpenregler:

- [RVS21/RVS61](#), Anschluss via BSB

#### Weishaupt (Modell WTU):

- [RVS23](#), Anschluss via LPB

**Die Software ist [hier](#) verfügbar.**

#### Autoren:

- Software, Schaltplan v1, urspr. Dokumentation EN, Ideenfindung, Support  
bis v0.16:  
*Gero Schumacher*
- Software, Platinenlayout v1 & v2, urspr. Dokumentation EN, Ideenfindung, Support  
ab v0.17:  
*Frederik Holst (bsb [ät] code-it.de)*
- Debugging, Handbuch, Übersetzungen, Ideenfindung, Support  
ab v0.17:  
*Ulf Dieckmann (adapter [ät] quantentunnel.de)*

*Basierend auf dem Code und der Mitarbeit von vielen anderen Entwicklern! Vielen Dank!*





# Schnellstartanleitung für den Arduino Due

[Zurück zur Einleitung](#)

## Schnellstartanleitung für den **Arduino Due**

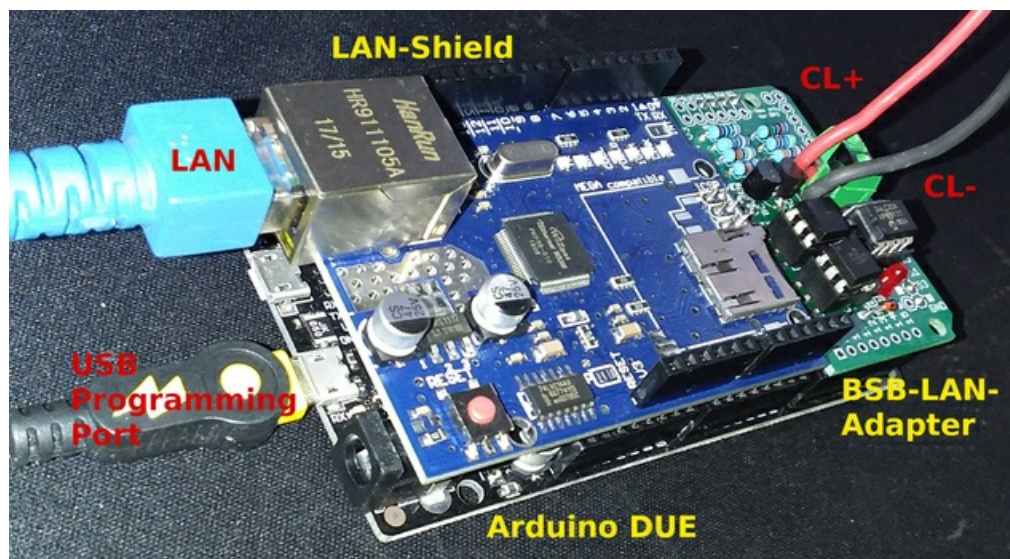
*Die folgende Kurzanleitung ersetzt nicht das Lesen des ausführlichen Handbuchs!*

*Bitte lies ebenso die jeweiligen detaillierteren Ausführungen in den entsprechenden Kapiteln.*

**Warnung:** *Elektrostatische Aufladungen können irreparable Schäden verursachen - erde dich vor Beginn der Arbeiten!*

Bereite das Setup vor, indem du das LAN-Shield und den BSB-LAN-Adapter auf den Arduino Due steckst. Schließe ein LAN-Kabel an und verbinde das Arduino-Setup mit einem USB-Kabel mit deinem Computer. Nutze dabei den 'Programming Port' des Due, das ist der 'mittlere' USB-Port, der neben der Netzteilbuchse platziert ist.

Sollte dein Rechner den Due nicht automatisch erkennen, ist der entspr. Treiber für dein Betriebssystem zu installieren.



Das komplette Setup (Arduino Due + LAN-Shield + BSB-LAN-Adapter v3) inklusive der entsprechenden Kabel.

**Führe nun die folgenden Schritte aus:**

1. Downloade und installiere die aktuelle Version der [Arduino IDE](#).
2. Downloade die [aktuelle Version von BSB-LAN](#).
3. Entpacke die heruntergeladene Datei "BSB\_LAN-master.zip" und wechsele in den Ordner.
4. Wechsle in den Ordner "BSB-LAN-master"/"BSB\_LAN" und benenne die Dateien `BSB_LAN_custom_defs.h.default` in **`BSB_LAN_custom_defs.h`** sowie `BSB_LAN_config.h.default` in **`BSB_LAN_config.h`** um!
5. Starte die Arduino IDE mit einem Doppelklick auf die Datei "BSB\_LAN.ino" im BSB\_LAN-Ordner.
  - Überprüfe den korrekten seriellen Port, an dem der Arduino Due am Rechner angeschlossen ist, unter "Werkzeuge/Port".
  - Stelle die Übertragungsgeschwindigkeit/Baudrate auf 115200 ein.

| Hinweis | |-----| | Sollten bis hier Probleme auftreten (bspw., dass das Board nicht erkannt wird), lies bitte die ausführliche Beschreibung in [Kapitel 2.1.1!](#) |

6. Passe die Einstellungen in der Datei "BSB\_LAN\_config.h" deinen Wünschen und Gegebenheiten entsprechend an. Dies gilt insbesondere für Einstellungen hinsichtlich der Nutzung von DHCP, einer ggf. abweichenden IP-Adresse sowie der optionalen Sicherheitsfunktionen.

Wenn alle Einstellungen angepasst wurden, starte den Flashvorgang mittels Klick auf "Sketch/Upload" bzw. "Sketch/Hochladen".

| Hinweis | |:-----| | Zusätzlich zur Anpassung der Datei "BSB\_LAN\_config.h" kann die Anpassung der Konfiguration von BSB-LAN auch später per Webinterface erfolgen. | | Weitere Hinweise sowie eine Beschreibung sämtlicher Konfigurationsmöglichkeiten findest du in [Kapitel 2.2 Konfiguration!](#) |

7. Nach Beenden des Flashvorgangs starte den [Seriellen Monitor der Arduino IDE](#) und beobachte die Ausgaben, die beim Start des Arduino erfolgen.

Dort wird u.a. auch die IP ausgegeben, die dem Setup bei Verwendung von DHCP zugeteilt wird.

**Nach erfolgreichem Beenden des Startvorgangs ist es empfehlenswert, die Stromversorgung des Arduino zu unterbrechen**, also das Board vom USB-Port deines Rechners entfernen. Dies ist nicht zwingend nötig, aus Sicherheitsgründen jedoch zu empfehlen.

8. **Schalte deine Heizung aus, damit der Heizungsregler stromlos ist.**

Schließe nun den Adapter des Arduino-Setups an den Regler an.

Verbinde dazu die reglerseitigen Anschlüsse "CL+" und "CL-" (bei BSB-Verwendung) bzw. "DB" und "MB" (bei LPB-Verwendung) mit den gleichnamigen Anschlüssen des Adapters.

Achte auf die korrekte Verbindung: Die verbundenen Anschlüsse müssen *namensgleich* sein, also bspw. "CL+" an "CL+" und "CL-" an "CL-".

| Hinweis | |:-----| | Eine ausführliche Anweisung diesbzgl. sowie Hinweise zum Anschluss eines Reglers mit PPS-Anschlüssen und Abbildungen diverser Regler und den entspr. zu nutzenden Anschlüssen findest du in [Kapitel 3.1!](#) |

9. Schalte die Heizung bzw. den Heizungsregler wieder ein.
10. Stelle die Stromversorgung des Arduino-Setups her, idealerweise mit einem spezifischen Netzteil mit Anschluss an der Hohlsteckerbuchse. Solltest du (noch) kein geeignetes Netzteil zur Hand haben, kannst du das Arduino-Setup auch über die USB-Buchse mit Strom versorgen. Dies kann entweder über ein entspr. leistungsstarkes USB-Netzteil oder über deinen USB-Port am Rechner erfolgen. Letztere Variante ist insofern von Vorteil, als dass du den [Seriellen Monitor der Arduino IDE](#) parallel zur Kontrolle des Startverhaltens des Setups nutzen kannst.
11. Starte einen Internetbrowser und rufe die Seite des BSB-LAN-Webinterfaces auf. Diese findest du unter der IP-Adresse, die du zuvor bei Schritt 6 eingestellt hast (voreingestellt ist "192.168.178.88"). Solltest du DHCP verwenden, so kann die vergebene IP während der Startsequenz des Arduino mittels des [Seriellen Monitors der Arduino IDE](#) ausgelesen werden.

*Wenn alles fehlerfrei und korrekt installiert ist, hast du nun (eingeschränkten) Zugriff auf deinen Heizungsregler. Um Zugriff auf alle verfügbaren Parameter deines Reglers zu erhalten, beachte bitte Schritt 12!*

| Hinweis | |:-----| | Sollten wider Erwarten Fehler oder Probleme auftauchen, so lies bitte *zusätzlich zu den bereits genannten Kapiteln* auch die Kapitel [13](#), [14](#) und [15](#)! |

12. *Um nun Zugriff auf sämtliche Parameter deines Reglers zu erhalten, muss eine reglerspezifische Datei `BSB_LAN_custom_defs.h` erstellt werden. Bitte lies hierfür das [Kap. 3.3](#)!*

 Support me on Ko-fi

[Weiter zur Schnellstartanleitung für ESP32-Boards](#)

# Schnellstartanleitung für ESP32-Boards

[Zurück zur Schnellstartanleitung für den Arduino Due](#)

## Schnellstartanleitung für ESP32-Boards

*Die folgende Kurzanleitung ersetzt nicht das Lesen des ausführlichen Handbuchs!*

*Bitte lies ebenso die jeweiligen detaillierteren Ausführungen in den entsprechenden Kapiteln.*

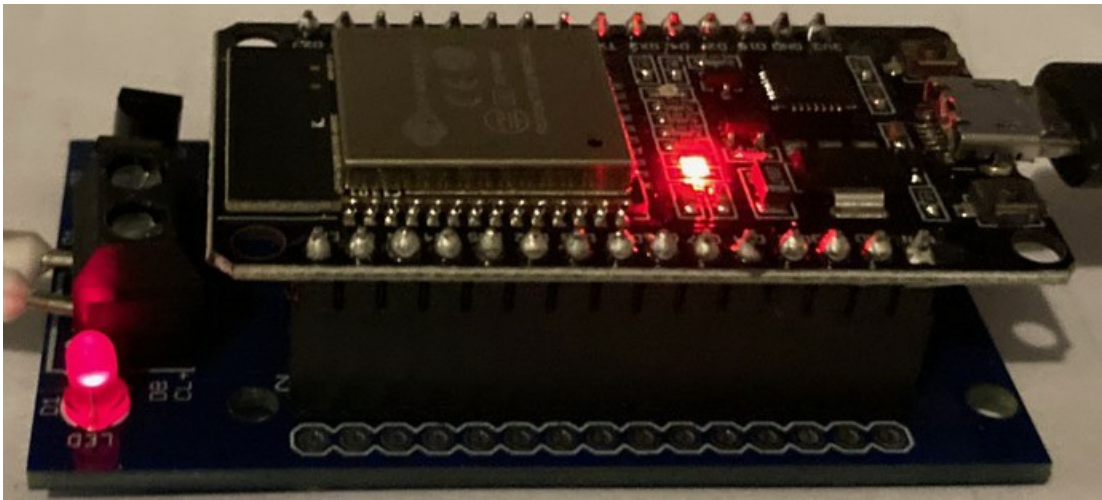
**Warnung:** *Elektrostatische Aufladungen können irreparable Schäden verursachen - erde dich vor Beginn der Arbeiten!*

*Achtung:* Berücksichtige bei der folgenden Anleitung deinen ESP32-Boardtyp!

- **Joy-It ESP32 NodeMCU**

Stecke den NodeMCU auf den BSB-LAN-Adapter und verbinde den NodeMCU mit einem USB-Kabel mit deinem Computer.

Sollte dein Rechner den NodeMCU nicht automatisch erkennen, ist der entspr. Treiber für dein Betriebssystem zu installieren.



*Das komplette Setup: Joy-It ESP32 NodeMCU samt aufgestecktem "BSB-LAN ESP32"-Adapter v4.2.*

- **Olimex ESP32-EVB & ESP32-PoE**

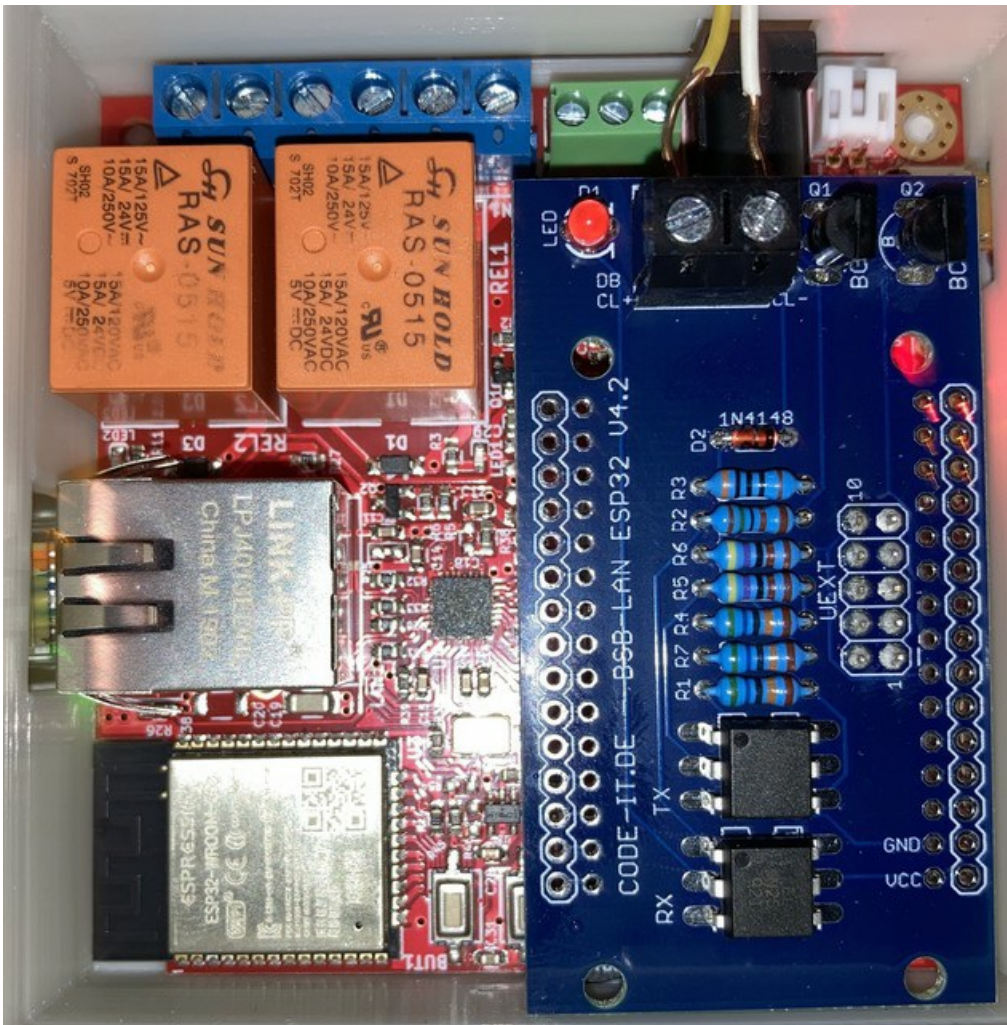
Stecke den BSB-LAN-Adapter auf den Olimex und verbinde den Olimex mit einem USB-Kabel mit deinem Computer.

**Achte beim Aufstecken des Adapterboards penibel darauf, dass die UEXT1-Buchse der Platine *exakt in der Mitte* der Olimex-Buchse aufgesteckt wird und alle Pins des Olimex Kontakt haben!** Ansonsten leuchtet beim korrekten Anschluss des Adapters an den Heizungsregler zwar die LED des Adapters, es ist aber kein Zugriff auf den Regler möglich.

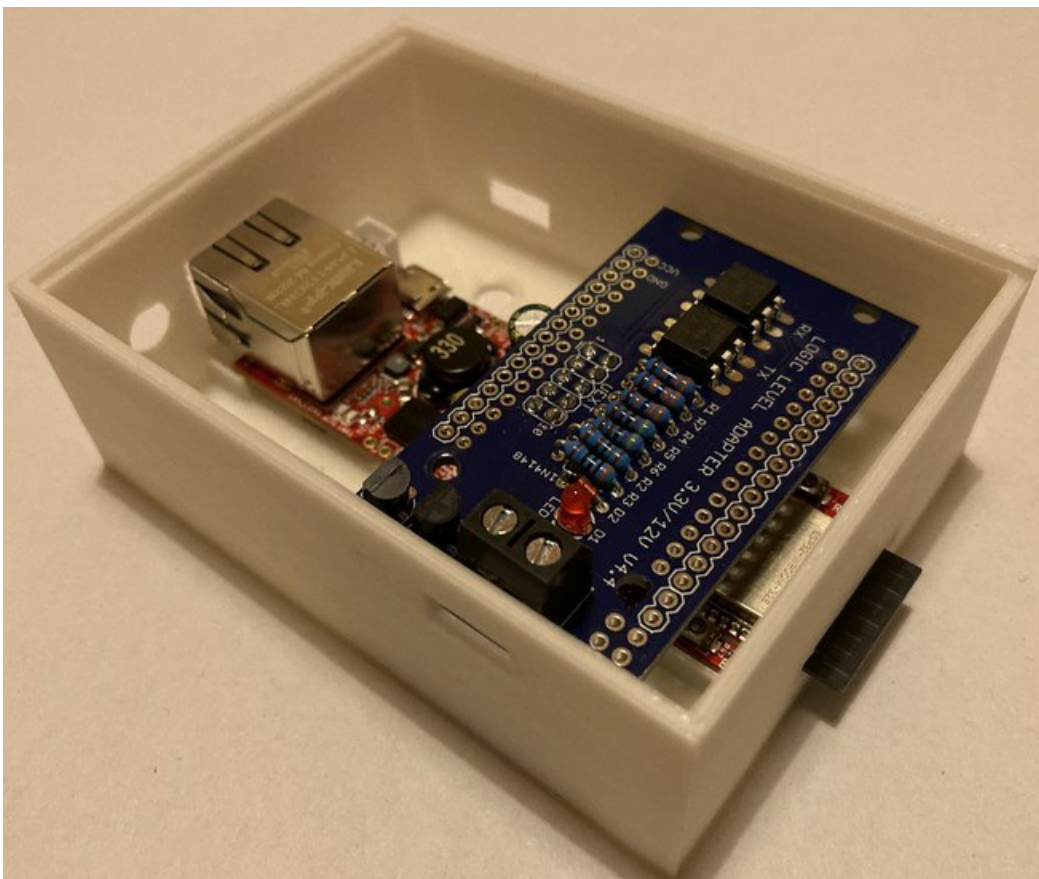
**Ebenso ist auf die korrekte Ausrichtung der Platine zu achten (s. Foto)!**

Sollte dein Rechner den Olimex nicht automatisch erkennen, ist der entspr. Treiber für dein Betriebssystem zu installieren.





Das komplette Setup: Olimex ESP32-EVB samt aufgestecktem "BSB-LAN ESP32"-Adapter v4.2.



Das komplette Setup: Olimex ESP32-PoE samt aufgestecktem "BSB-LAN ESP32"-Adapter v4.4.

Führe nun die folgenden Schritte aus:

1. Downloade und installiere die aktuelle Version der [Arduino IDE](#).

Füge dann das ESP32-SDK per Boardverwalter-URL hinzu (*Achtung: SDK 2.0.2 oder höher verwenden!*) und installiere dann die ESP32-Boardbibliotheken im Boardverwalter. [Hier](#) findest du eine ausführliche Schritt-für-Schritt-Anleitung dazu.

2. Downloade die [aktuelle Version von BSB-LAN](#).

3. Entpacke die heruntergeladene Datei "BSB\_LAN-master.zip" und wechsle in den Ordner.

4. Wechsle in den Ordner "BSB-LAN-master"/"BSB\_LAN" und benenne die Dateien *BSB\_LAN\_custom\_defs.h.default* in ***BSB\_LAN\_custom\_defs.h*** sowie *BSB\_LAN\_config.h.default* in ***BSB\_LAN\_config.h*** um!

- Öffne die Datei "BSB\_LAN\_config.h" und aktiviere das Definement `#define WIFI`, wenn du WLAN verwenden willst. Solltest du ein Olimex-Board verwenden und den LAN-Anschluss nutzen wollen, lasse das Definement bitte auskommentiert: `//#define WIFI`.

- Trage bei der Verwendung von WLAN die Zugangsdaten für dein WLAN-Netzwerk bei den Einträgen

```
char wifi_ssid[32] = "YourWiFiNetwork"; sowie
```

```
char wifi_pass[64] = "YourWiFiPassword"; ein.
```

5. Starte die Arduino IDE mit einem Doppelklick auf die Datei "BSB\_LAN.ino" im BSB\_LAN-Ordner.

- Überprüfe den korrekten seriellen Port, an dem das ESP32-Board am Rechner angeschlossen ist, unter "Werkzeuge/Port".

- Stelle die Übertragungsgeschwindigkeit/Baudrate auf 115200 ein.

| Achtung | |:-----| | Wähle nun den entspr. ESP32-Boardtyp unter Tools/Board bzw. Werkzeuge/Board aus! |

- Für den in diesem Handbuch empfohlenen [Joy-It ESP32-NodeMCU](#) (oder identische Clones mit einem "ESP32-WROOM"-Chip) lautet der passende Boardtyp "ESP32 Dev Module".

Wähle dann bei "Partition Scheme" die Variante "Default 4MB with spiiffs (1.2BM APP/1.5MB SPIFFS)" aus.

- Für das empfohlene [Olimex ESP32-EVB](#) wähle bitte den gleichnamigen Eintrag aus der Liste aus.

Wähle dann bei "Partition Scheme" die Variante "Minimal SPIFFS (Large APPS with OTA)" aus.

| Hinweis | |:-----| | Sollten bis hier Probleme auftreten (bspw., dass das Board nicht erkannt wird), lies bitte die ausführliche Beschreibung in [Kapitel 2.1.2!](#) |

6. Passe die weiteren Einstellungen in der Datei "BSB\_LAN\_config.h" deinen Wünschen und Gegebenheiten entsprechend an.

Dies gilt insbesondere für Einstellungen hinsichtlich der Nutzung von DHCP, einer ggf. abweichenden IP-Adresse sowie der optionalen Sicherheitsfunktionen.

Wenn alle Einstellungen angepasst wurden, starte den Flashvorgang mittels Klick auf "Sketch/Upload" bzw. "Sketch/Hochladen".

| Hinweis | |:-----| | Zusätzlich zur Anpassung der Datei "BSB\_LAN\_config.h" kann die Anpassung der Konfiguration von BSB-LAN auch später per Webinterface erfolgen. | | Weitere Hinweise sowie eine Beschreibung sämtlicher Konfigurationsmöglichkeiten findest du in [Kapitel 2.2 Konfiguration!](#) |

7. Nach Beenden des Flashvorgangs starte den [Seriellen Monitor der Arduino IDE](#) und beobachte die Ausgaben, die beim Start des ESP32 erfolgen. Dort wird u.a. auch die IP ausgegeben, die dem Setup bei Verwendung von DHCP zugeteilt wird.

**Nach erfolgreichem Beenden des Startvorgangs ist es empfehlenswert, die Stromversorgung des Arduino zu unterbrechen**, also das Board vom USB-Port deines Rechners entfernen. Dies ist nicht zwingend nötig, aus Sicherheitsgründen jedoch zu empfehlen.

8. **Schalte deine Heizung aus, damit der Heizungsregler stromlos ist.**

Schließe nun den Adapter des Arduino-Setups an den Regler an. Verbinde dazu die reglerseitigen Anschlüsse "CL+" und "CL-" (bei BSB-Verwendung) bzw. "DB" und "MB" (bei LPB-Verwendung) mit den gleichnamigen Anschlüssen des Adapters.

Achte auf die korrekte Verbindung: Die verbundenen Anschlüsse müssen *namensgleich* sein, also bspw. "CL+" an "CL+" und "CL-" an "CL-".

| Hinweis | |:-----| | Eine ausführliche Anweisung diesbzgl. sowie Hinweise zum Anschluss eines Reglers mit PPS-Anschlüssen und Abbildungen diverser Regler und den entspr. zu nutzenden Anschlüssen findest du in [Kapitel 3.1!](#) |

9. Schalte die Heizung bzw. den Heizungsregler wieder ein.

10. Starte das Setup durch Druck auf die Reset-Taste neu bzw. stelle die Stromversorgung des ESP32-Board-Setups wieder her, idealerweise mit einem spezifischen Netzteil mit Anschluss an der microUSB-Buchse (NodeMCU) bzw. Hohlsteckerbuchse (Olimex).



Solltest du (noch) kein geeignetes Netzteil zur Hand haben, kann die Stromversorgung auch über deinen USB-Port am Rechner erfolgen. Letztere Variante ist insofern von Vorteil, als dass du den [Seriellen Monitor der Arduino IDE](#) parallel zur Kontrolle des Startverhaltens des Setups nutzen kannst.

11. Starte einen Internetbrowser und rufe die Seite des BSB-LAN-Webinterfaces auf.

Diese findest du unter der IP-Adresse, die du zuvor bei Schritt 6 eingestellt hast (voreingestellt ist "192.168.178.88").

Solltest du DHCP verwenden, so kann die vergebene IP während der Startsequenz des Arduino mittels des [Seriellen Monitors der Arduino IDE](#) ausgelesen werden.

*Wenn alles fehlerfrei und korrekt installiert ist, hast du nun (eingeschränkten) Zugriff auf deinen Heizungsregler. Um Zugriff auf alle verfügbaren Parameter deines Reglers zu erhalten, beachte bitte Schritt 12!*

| Hinweis | |:-----| | Sollten wider Erwarten Fehler oder Probleme auftauchen, so lies bitte *zusätzlich zu den bereits genannten Kapiteln* auch die Kapitel [13](#), [14](#) und [15](#)! |

12. Um nun Zugriff auf sämtliche Parameter deines Reglers zu erhalten, muss eine reglerspezifische Datei `BSB_LAN_custom_defs.h` erstellt werden. Bitte lies hierfür das [Kap. 3.3](#)!

 Support me on Ko-fi

[Zurück zum Inhaltsverzeichnis](#)

# 1. BSB-LAN: Die Hardware

[Zurück zum Inhaltsverzeichnis](#)

## 1. BSB-LAN: Die Hardware

In den folgenden Kapiteln wird die Hardware des BSB-LAN Setups vorgestellt. Dabei handelt es sich zum einen um den jeweiligen plattformspezifischen BSB-LAN [Adapter](#) und zum anderen um den jeweiligen Mikrocontroller, auf den die BSB-LAN Software geflasht wird.

BSB-LAN kann sowohl mit einem [Arduino Due](#) samt spezifischem Adapter als auch auf einem [ESP32](#) samt spezifischem Adapter betrieben werden. Der jeweilige plattformspezifische Adapter erlaubt dabei ein einfaches und passgenaues Aufstecken auf den entspr. Mikrocontroller. In einigen Fällen kann ein plattformspezifischer Adapter auch mit den anderen Mikrocontrollern verwendet werden, in dem Fall ist jedoch kein passgenaues Aufstecken möglich. Weitere Hinweise diesbzgl. sind in den jeweiligen Hinweiskästen im entspr. Kapitel zu finden. Da es bei den kompatiblen Mikrocontrollern (Arduino Due / ESP32 NodeMCU / Olimex ESP32-EVB) jedoch plattform- und designspezifische Unterschiede gibt, die es bei bestimmten Einsatzzwecken zu beachten gilt (bspw. wenn weitere Hardware angeschlossen werden soll), werden die relevantesten Unterschiede im Folgen kurz als tabellarische Übersicht dargestellt.

*Eine etwaige mögliche Nachrüstung einzelner Komponenten (wie bspw. eines microSD-Kartenlesers bei einem NodeMCU) wird hierbei nicht berücksichtigt!*

Funktion	Arduino Due + LAN-Shield	ESP32 NodeMCU "JoyIt"	Olimex ESP32-EVB	Olimex ESP32-PoE
LAN onboard	Ja	Nein	Ja	Ja
WLAN onboard	Nein	Ja	Ja	Ja
OTA-Update	Nein	Ja	Ja	Ja
microSD-Kartenleser onboard	Ja	Nein	Ja	Ja
Freie Pins	sehr viele	viele	2 (GPIO 13&16)	viele
Relais onboard	Nein	Nein	Ja (2)	Nein
Bluetooth onboard	Nein	Ja	Ja	Ja
Power-over-Ethernet	Nein	Nein	Nein	Ja
Due-spezifischer Adapter verwendbar	Ja	Ja	Ja	Ja
ESP-spezifischer Adapter verwendbar	Nein	Ja	Ja	Ja

<b>Anmerkungen</b>
Insbesondere die nur <i>zwei</i> freien Pins bei einem Olimex ESP32-EVB (GPIO 13/I2C-SDA & GPIO 16/I2C-SCL), die problemlos für den Anschluss weiterer Hardware (wie bspw. Sensoren, Relais, Taster) genutzt werden können, sind u.U. ein Ausschlusskriterium, das es zu beachten gilt!
Soll die interne Loggingfunktion von BSB-LAN auf microSD-Karte genutzt werden, so ist wiederum vom NodeMCU abzuraten, da die u.U. häufigen Schreibzyklen beim Speichern der Daten auf dem EEPROM-Chip des ESP32 zu einem frühzeitigen Ausfall ("wear out") führen können.
<i>Vor der Entscheidung für einen der genannten Mikrocontroller ist es daher ratsam, den späteren Einsatzzweck und etwaige Erweiterungen auf Hardwarebasis gründlich zu überdenken. Hierfür empfiehlt es sich, das Handbuch im Vorfeld aufmerksam zu lesen.</i>

<b>Hinweis bzgl. Verwendung eines Raspberry Pi</b>
----------------------------------------------------

## Hinweis bzgl. Verwendung eines Raspberry Pi

Wie in [Kap. 1.4](#) beschrieben, ist der Anschluss des Adapters an einen Raspberry Pi zwar grundsätzlich möglich, der Einsatz der BSB-LAN-Software hingegen nicht. Davon abgesehen können wir den Einsatz eines RPi als Mikrocontroller für diesen Einsatzzweck nicht empfehlen. Als Hauptargumente sprechen unserer Ansicht nach insbesondere die Faktoren Preis, Stromverbrauch sowie das zu wartende Betriebssystem dagegen.

## 1.1 Adapter

Der BSB-LAN-Adapter ist im Prinzip ein Pegelwandler, der die 12V Bussignale in 3,3V Signale für den benötigten Mikrocontroller umsetzt. Die Busanschlüsse des Adapters sind von der eigentlichen elektronischen Schaltungstechnik galvanisch getrennt.

Der BSB-LAN Adapter ist grundsätzlich in zwei verschiedenen Versionen verfügbar. Zum einen als *Arduino Due-spezifische Version mit einem EEPROM*, zum anderen als eine *ESP32-spezifische Version ohne EEPROM*.

### Achtung

Es sei bereits an dieser Stelle angemerkt, dass die ESP32-spezifische Adapterversion aufgrund des fehlenden EEPROMs *nur* mit einem ESP32 genutzt werden kann - die Due-spezifische Version hingegen kann (wenn auch nicht komfortabel aufsteckbar) auch mit einem ESP32 genutzt werden.

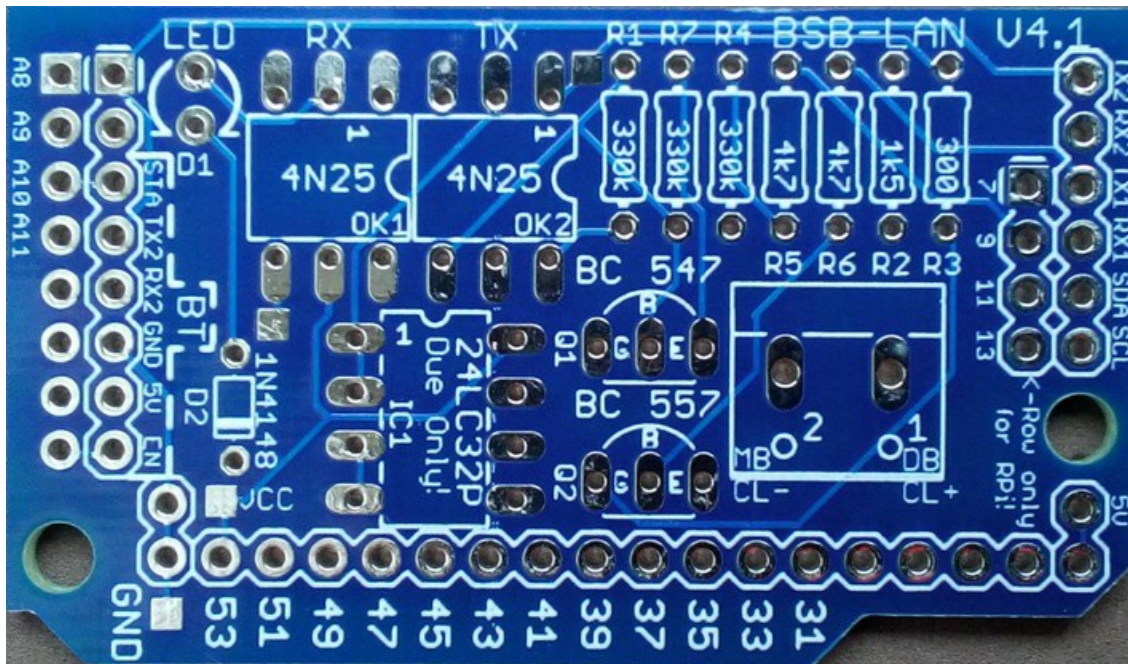
Der Adapter kann von versierten Usern selbst gebaut werden, einen entspr. Schaltplan für die Due-kompatible Version samt EEPROM findet sich in [Anhang A1](#) - bei der ESP32-kompatiblen Version entfällt lediglich das EEPROM, der Rest der Schaltung ist identisch.

**Neben dem kompletten Eigenbau besteht die Möglichkeit, fertige Adapterplatinen bei Frederik Holst ([bsb \[ät\] code-it.de](mailto:bsb[at]code-it.de)) zu erwerben.**

Die bei Frederik erhältlichen PCBs können auf die im Folgenden vorgestellten kompatiblen Mikrocontroller (Arduino Due / Joy-It ESP32-NodeMCU / Olimex ESP32-EVB) passgenau aufgesteckt werden, so dass man bereits im Vorfeld gründlich überlegen sollte, welchen Mikrocontroller man für das Setup im weiteren Verlauf einsetzen möchte.

### 1.1.1 Due-Version

Die Due-spezifische Version des BSB-LAN-Adapters weist ein EEPROM auf, in dem die Einstellungen der BSB-LAN-Software (ab v2.0) gespeichert werden. Der Adapter lässt sich komfortabel und sicher auf den Due aufstecken.



Die BSB-LAN-Adapterplatine, Due-Version, v4.1, Oberseite, unbestückt.

### Hinweis

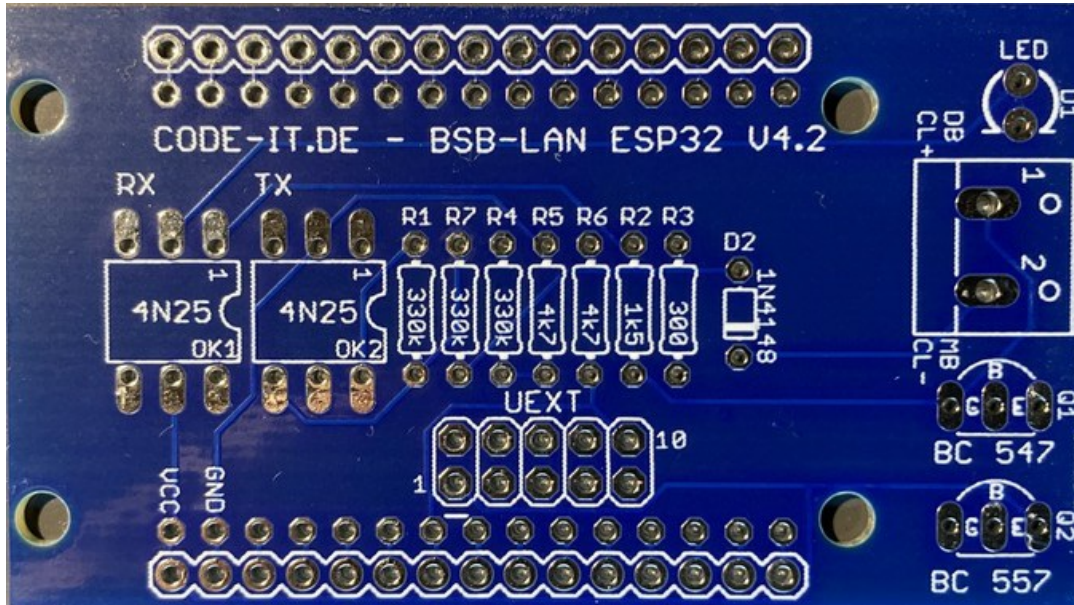


## Hinweis

Die Verwendung des Due-spezifischen Adapters an einem ESP32 ist trotz des EEPROMs prinzipiell möglich, der Adapter kann jedoch nicht wie bei einem Due problemlos auf ein ESP32-Board aufgesteckt werden. Sollte der Adapter trotzdem mit einem ESP32-Board genutzt werden, so ist darauf zu achten, dass die Verbindungen zwischen Adapter und ESP32 korrekt und zuverlässig hergestellt (also bestenfalls gelötet) werden.

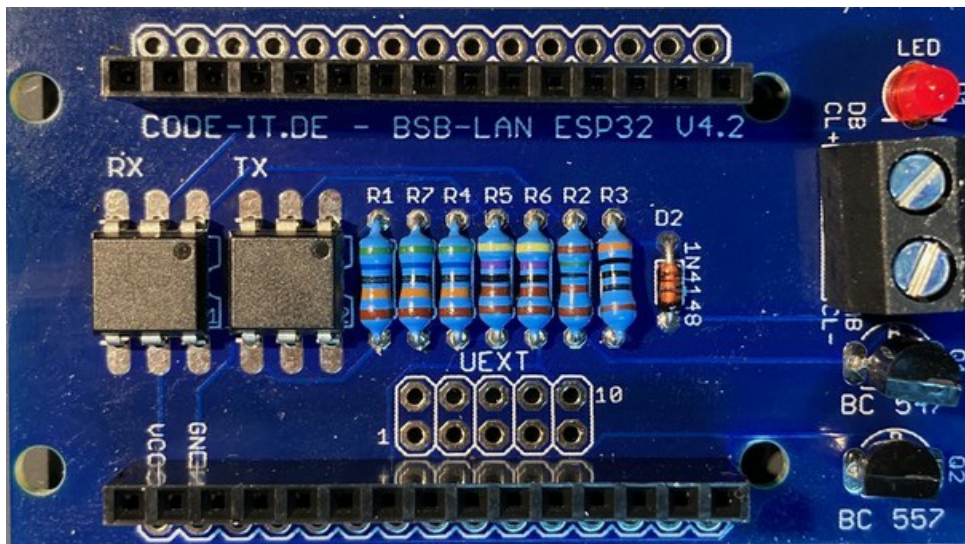
### 1.1.2 ESP32-Version

Für bestimmte ESP32-Boardvarianten gibt es eine eigene BSB-LAN-Adapterplatine: "BSB-LAN ESP32".



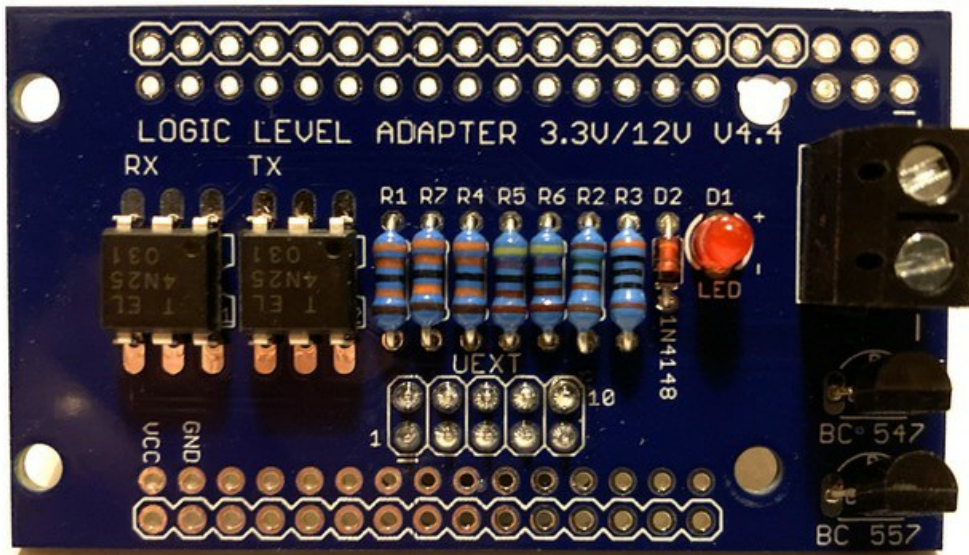
Die "BSB-LAN ESP32"-Adapterplatine, v4.2, unbestückt.

Diese BSB-LAN-Adapterplatine ist auf das 30 polige [ESP32-NodeMCU-Board von Joy-It](#) (WROOM32-Chip) ausgelegt.



Die "BSB-LAN ESP32"-Adapterplatine, v4.2, bestückt für den empfohlenen NodeMCU.

Darüber hinaus kann der Adapter außerdem mit einem [Olimex ESP32-EVB](#) sowie einem [Olimex ESP32-PoE](#) genutzt und durch Hinzufügen einer doppelreihigen fünfpoligen Pinbuchse (2x5 polig, RM 2,54mm) auf der Platinenunterseite direkt auf den zehnpoligen UEXT-Stecker von Olimex-Boards aufgesteckt werden.



Die "BSB-LAN ESP32"-Adapterplatine, v4.4, bestückt für die empfohlenen Olimex-Boards.

Die ESP32-spezifische Version des BSB-LAN-Adapters weist kein EEPROM auf, Einstellungen werden im Flashspeicher des ESP32 gespeichert.

#### Hinweis

Die Verwendung des ESP32-spezifischen Adapters an einem Due ist aufgrund des fehlenden EEPROMs *nicht* möglich!

## 1.2 Arduino Due

Grundsätzlich ist die Verwendung eines [originalen Arduino Due](#) zu empfehlen.

Erfahrungsgemäß können jedoch auch günstige Nachbauten des Arduino Due verwendet werden, der Einsatz dieser Clones ist normalerweise problemlos möglich. Bei diesen sollte beim Kauf allerdings darauf geachtet werden, ob in den Produktbeschreibungen auf ein verändertes Platinenlayout, geänderte Pinbelegungen o.ä. hingewiesen wird. Sollte dies der Fall sein, so sind ggf. in der Datei `BSB_LAN_config.h` diesbezügliche Anpassungen vorzunehmen.

**ACHTUNG: Die GPIOs des Arduino Due sind nur 3.3V kompatibel!**

Ein Pinout-Schema des Arduino Due ist im [Anhang B](#) abgebildet.



Ein kompatibler Clone des Arduino Due.

#### Hinweise



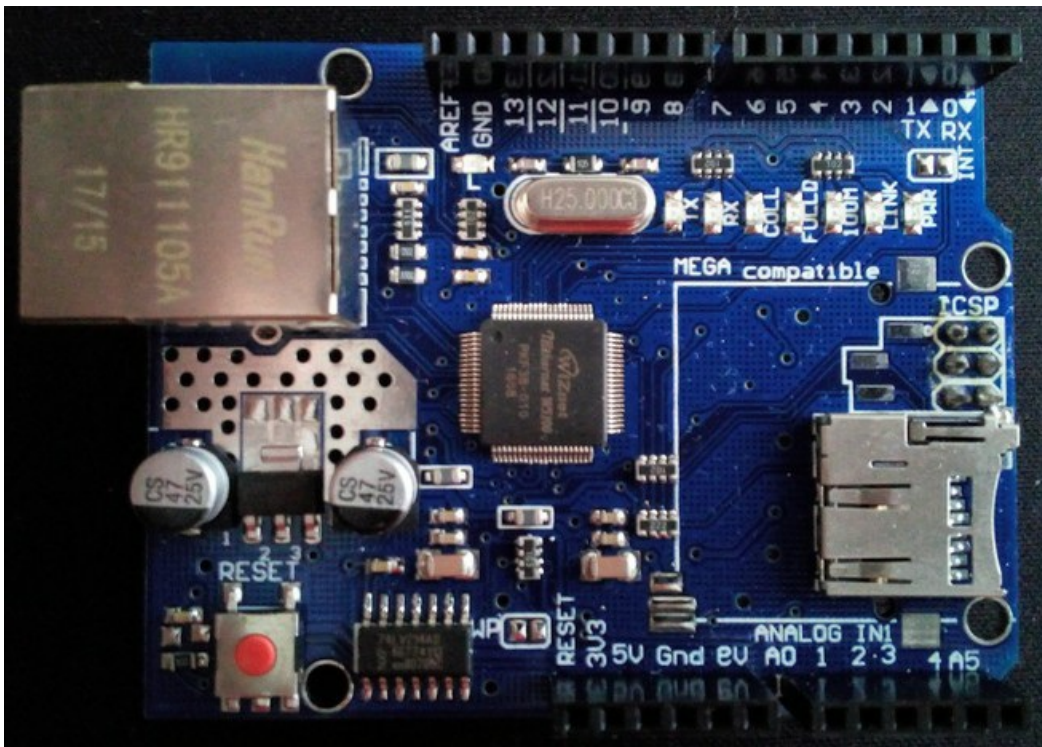
<b>Hinweise</b>
Es wird empfohlen, den Arduino mit einem externen Netzteil an der Hohlsteckerbuchse zu betreiben. Laut den technischen Daten von Arduino liegt dabei die empfohlene Versorgungsspannung in einem Bereich von 7-12V (Limit: 6-16V). Die Versorgung mit einem 9V-Steckernetzteil (mind. 1000mA) stellte sich bisher als zuverlässige Lösung dar.
Soll die Stromversorgung trotzdem über die USB Buchse des Due erfolgen, so ist möglichst der 'Programming Port', also der mittlere USB Port (neben der Hohlsteckerbuchse gelegen) zu nutzen.
Der Due kann via Netzteil an der Hohlsteckerbuchse mit Strom versorgt und gleichzeitig via USB am Programming Port mit dem Computer verbunden werden.
Der Adapter kann am Bus des Heizungsreglers beim Flashen des Due angeschlossen bleiben.
Achte darauf, dass du möglichst ein qualitativ hochwertiges USB-Kabel verwendest. Dies gilt sowohl für den Fall, dass du den Due via USB mit Strom versorgen willst, als auch für den Fall, dass du den Due zum Flashen an deinen PC anschließen möchtest. Insbesondere lange und <i>dünne</i> Kabel (bspw. Zubehör von Smartphones) können Probleme bei der Stromversorgung und somit der Stabilität des Due verursachen und/oder sind nicht immer voll beschaltet, so dass eine Nutzung für die Datenübertragung nicht möglich ist.
Bei einigen Due-Modellen/-Clones kann es vorkommen, dass sie nach einem initialen Start (bspw. nach einem Stromausfall) nicht richtig zu funktionieren scheinen und erst nach einem Betätigen des Reset-Buttons korrekt arbeiten. Hier kann anscheinend das <a href="#">Hinzufügen eines Kondensators</a> Abhilfe schaffen.

### 1.2.1 Due + LAN: Das LAN-Shield

Grundsätzlich ist die Verwendung eines [originalen Arduino Ethernet-Shields](#) zu empfehlen, das direkt auf den Arduino Due aufgesteckt werden kann.

Die LAN-Shields gibt (bzw. gab) es in zwei verschiedenen Ausführungen. Zum einen mit einem WIZnet W5100-Chip (v1), zum anderen mit einem W5500-Chip (v2).

Die Verwendung des aktuellen v2-Shields (W5500) wird empfohlen, es ist u.a. im offiziellen [Arduino-Store](#) und bei [Reichelt](#) erhältlich. Erfahrungsgemäß können jedoch auch günstige Nachbauten dieser Shields verwendet werden, der Einsatz dieser Clones ist normalerweise problemlos möglich. Allerdings sollte beim Kauf darauf geachtet werden, ob in den Produktbeschreibungen auf ein verändertes Platinenlayout, geänderte Pinbelegungen o.ä. hingewiesen wird. Sollte dies der Fall sein, so sind ggf. in der Datei `BSB_LAN_config.h` diesbezügliche Anpassungen vorzunehmen.



Ein kompatibler Clone eines LAN-Shields mit einem W5100-Chip.

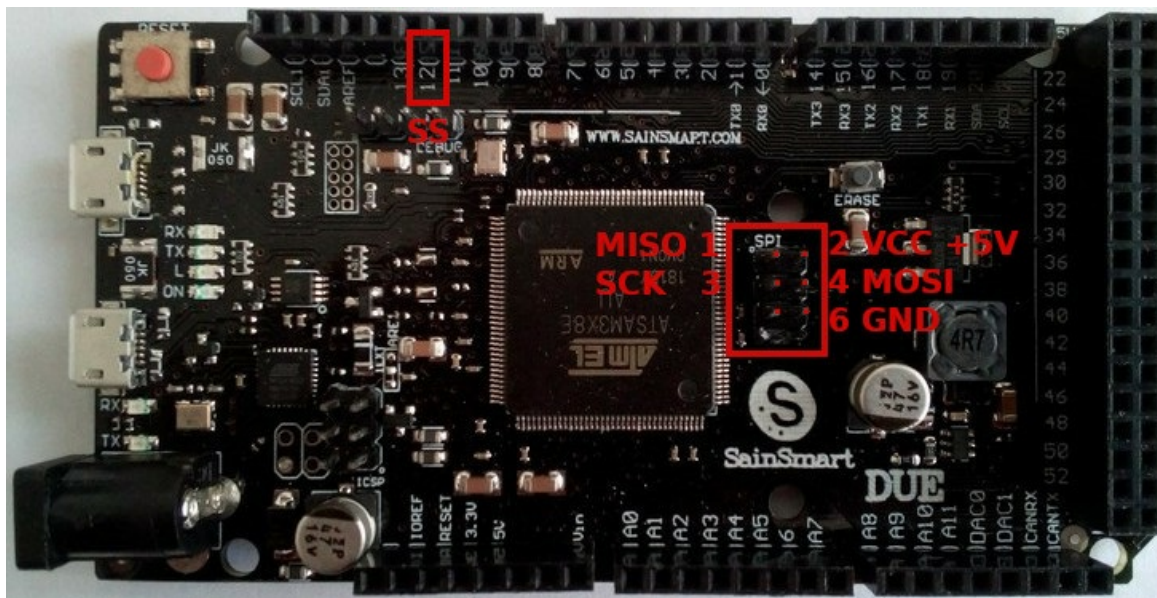
<b>Tipp</b>
-------------

## Tipp

Als LAN-Kabel sollte möglichst eine geschirmte Ausführung mit einer Mindestlänge von 1m verwendet werden.

### 1.2.2 Due + WLAN: Die ESP8266-WiFi-Lösung

Das Due-Setup lässt sich mittels eines ESP8266 (NodeMCU oder Wemos D1) anstelle des LAN-Shields auch in WLAN-Netzwerke integrieren. Hierfür ist der ESP8266 mit dem sechspoligen SPI-Anschluss des Arduino Due zu verbinden und wird dabei vom Due mit Strom versorgt (+5V). Der ESP8266 muss dafür mit einer speziellen Firmware geflasht werden, Näheres dazu erfährst du weiter unten in diesem Kapitel. Die BSB-LAN-Software wird weiterhin auf dem Due installiert.



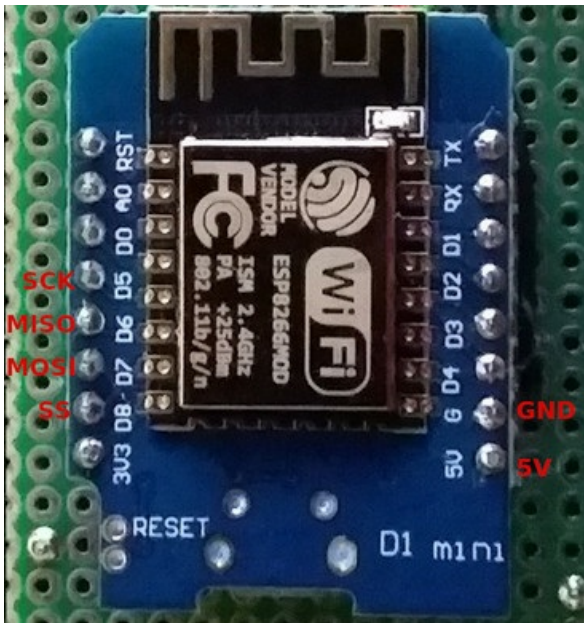
Der zu verwendende sechspolige SPI-Anschluss des Arduino Due.

Die Anschlüsse sind wie folgt zu verbinden:

Pin DUE	Funktion	Pin ESP8266
SPI 1	MISO (Master Input Slave Output)	D06
SPI 2	VCC (Stromversorgung ESP)	+5V / Vin
SPI 3	SCK (Serial Clock)	D05
SPI 4	MOSI (Master Output Slave Input)	D07
SPI 6	GND (Stromversorgung ESP)	GND
Pin 12	SS (Slave Select)	D08

Kommt keine weitere per SPI angeschlossene Komponente (bspw. LAN-Shield, Kartenleser) zum Einsatz, so kann auf den Anschluss von "SS" (SlaveSelect, DUE Pin 12 = D08 beim ESP8266) verzichtet werden.

Im Falle der Verwendung von SS kann der Anschluss auch an einem anderen Pin als Pin 12 erfolgen, der entspr. Pin muss in der Datei *BSB\_lan\_config.h* entspr. definiert werden. In diesem Fall ist jedoch darauf zu achten, dass der zu verwendende Pin nicht zu den geschützten Pins zählt und nicht anderweitig verwendet wird. Es wird daher empfohlen, es bei der Voreinstellung (Pin 12) zu belassen.



Die korrespondierenden Anschlüsse beim Wemos D1.

Es bietet sich an, das LAN-Shield zu entfernen, eine unbestückte Lochrasterplatine passend auf dem Due zu platzieren und mit den entspr. Anschlüssen zu versehen. So kann der Wemos D1 / NodeMCU stabil auf dem Due platziert werden. Je nach Gehäuse ist hier u.U. auf die Bauhöhe zu achten.



Wemos D1 auf einer Lochrasterplatine auf dem Arduino Due.

#### Achtung

Bei dieser Lösung entfällt jedoch die Möglichkeit, Daten auf eine microSD-Karte zu loggen. Soll dies trotz WiFi-Anbindung weiterhin möglich sein, so muss entweder ein entspr. Kartenmodul zusätzlich oder der ESP parallel zum bestehenden LAN-Shield angeschlossen werden. In beiden Fällen muss der SS-Pin *zwingend* angeschlossen werden (s. Pinbelegung/Anschluss).

*Ob ein paralleler Betrieb von LAN-Shield und ESP8266 problemlos möglich ist, wurde bisher jedoch noch nicht getestet.*

#### Flashen des ESP8266:

Der ESP8266 muss mit einer speziellen Firmware geflasht werden. Für die Verwendung der Arduino (o.ä.) muss darauf geachtet werden, dass zuvor die *Version 2.7.4* der ESP8266-Bibliotheken mittels des Boardverwalters installiert und ausgewählt wurde.

Die benötigte Firmware [WiFiSpiESP](#) für den ESP8266 liegt bereits als zip-file im BSB-LAN-Repository. Das zip-file *muss in einem anderen Ordner als BSB\_lan* entpackt werden! Der ESP8266 ist dann mit der Datei *WiFiSPIESP.ino* zu flashen.

#### Konfiguration von BSB-LAN:

Zur Verwendung muss das Definement `#define WIFI` in der Datei *BSB\_lan\_config.h* aktiviert werden. Des Weiteren müssen die beiden Variablen `wifi_ssid` und `wifi_pass` entsprechend angepasst und die SSID des WLAN sowie das Passwort eingetragen werden. Diese Angaben können auch im Nachhinein via Webinterface geändert werden.



<b>Hinweise</b>
Bei Verwendung von DHCP kann die vom Router vergebene IP-Adresse im Seriellen Monitor der Arduino IDE beim Start des DUE ausgelesen werden.
Bei der Verwendung der ESP-WiFi-Lösung lautet der Hostname <i>nicht</i> WIZnetXYZXYZ, sondern i.d.R. ESP-XYZXYZ, wobei sich die Ziffern-Buchstabenkombination "XYZXYZ" nach "ESP-" aus den letzten drei Bytes (also den letzten sechs Zeichen) der MAC-Adresse des ESP zusammensetzt.
Bei Verwendung der ESP-WiFi-Lösung lässt sich die MAC-Adresse des ESP <i>nicht</i> selbst festlegen.

## 1.3 ESP32

Die BSB-LAN-Software ist auch auf einem ESP32 lauffähig. Es sind allerdings zwingend bestimmte Anpassungen vorzunehmen, die im [Kap. 2.1.2](#) beschrieben sind.

<b>Achtung, wichtiger Hinweis</b>
Falls das ESP32-Framework bereits in der Arduino IDE installiert ist und dir die verschiedenen ESP32-Boardvarianten angezeigt werden, überprüfe bitte im "Boardverwalter" unter "Werkzeuge/Boards", dass die <b>Version 2.0.2</b> (oder höher, falls verfügbar) installiert ist. Sollte das Board <i>nicht</i> aufgeführt sein, so muss die ESP32-Plattform in der Arduino IDE hinzugefügt werden. Informationen hierzu findest du in <a href="#">Kap. 12.1.2</a> .

Im Grunde kann jeder ESP32 verwendet werden, aufgrund des spezifischen Platinendesigns wird jedoch die Verwendung folgender ESP32-basierter Modelle empfohlen:

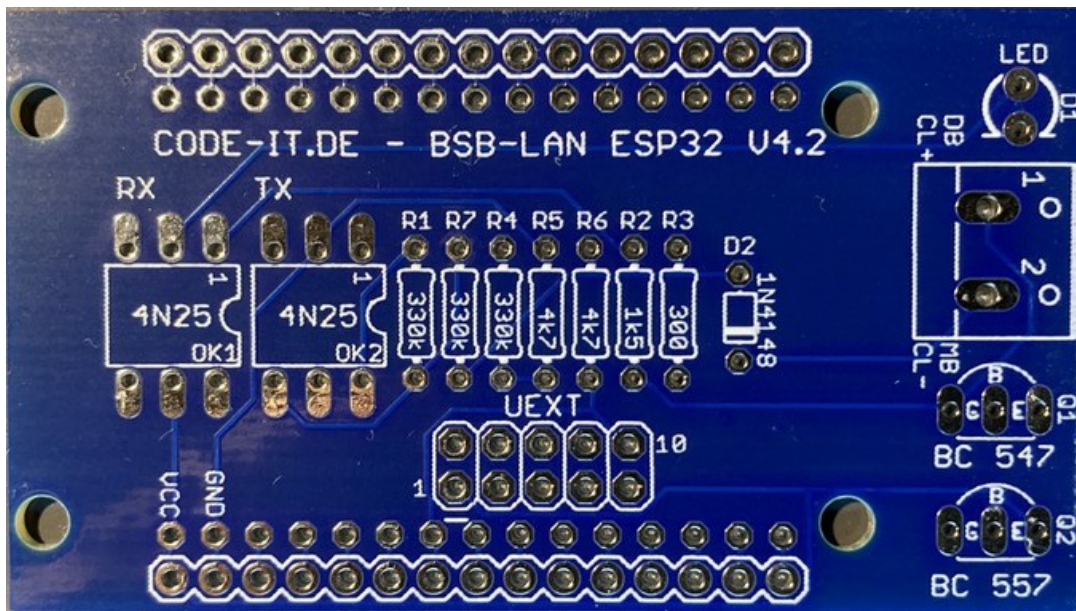
- [ESP32-NodeMCU-Board von Joy-It](#)
- [Olimex ESP32-EVB](#)
- [Olimex ESP32-PoE](#)

Die genannten ESP32-Boards nutzen das ESP32-**WROOM32**-Modul.

<b>Achtung, wichtige Hinweise</b>
Falls du ein anderes als die empfohlenen Boards verwenden möchtest, prüfe immer das spezifische Datenblatt und versichere dich, dass du Pins für RX/TX auswählst, die nicht anderweitig verwendet werden. Diese müssen dann in der Datei <i>BSB_LAN_config.h</i> bei dem entspr. Definement eingetragen werden!
Beachte bitte außerdem, dass die Autodetect-Funktion von BSB-LAN für den angeschlossenen Regler nur mit den von uns empfohlenen Boardtypen funktioniert!
Es ist ausdrücklich empfohlen, immer einen ESP32- <b>WROOM32</b> -Modultyp zu wählen, falls ein anderes ESP32-Board als die von uns empfohlenen Boards eingesetzt werden soll! Solltest du dennoch ein <i>WROVER</i> -Modul einsetzen wollen oder müssen, so müssen entweder andere Pins anstelle von 16/17 für RX/TX genutzt werden, da WROVER-Module diese beiden Pins intern für das SPI-PSRAM-Modul verwenden, oder es muss im BSB-LAN-Code die Verwendung des PSRAM deaktiviert werden.

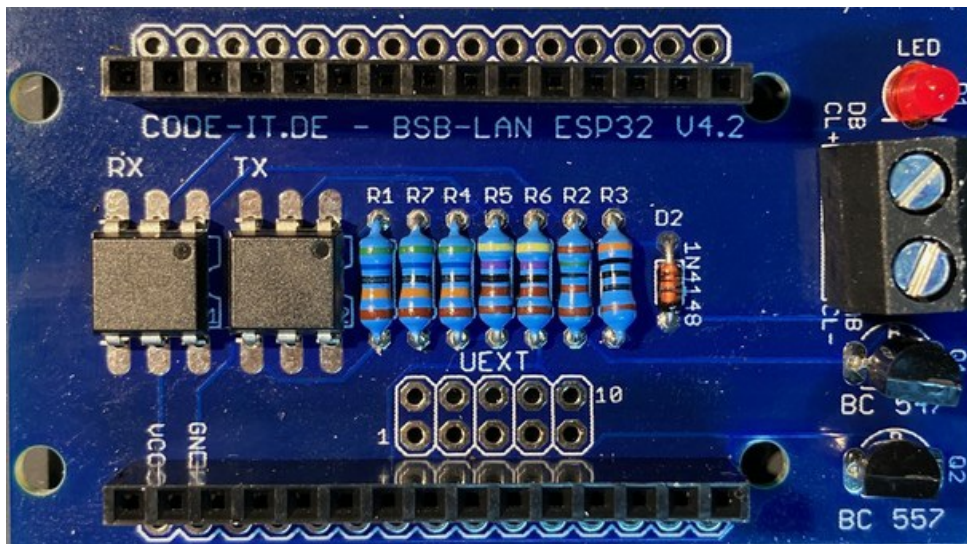
### 1.3.1 ESP32 mit spezifischem "BSB-LAN ESP32"-Adapter

Für eine bestimmte ESP32-Boardvariante gibt es eine eigene BSB-LAN-Adapterplatine: "BSB-LAN ESP32".



Die "BSB-LAN ESP32"-Adapterplatine, v4.2, unbestückt.

Diese BSB-LAN-Adapterplatine ist auf das 30 polige [ESP32-NodeMCU-Board von Joy-It](#) (WROOM32-Chip) ausgelegt.



Die "BSB-LAN ESP32"-Adapterplatine, v4.2, bestückt für den empfohlenen NodeMCU.

Die ESP32-Adapterversion kann außerdem mit einem [Olimex ESP32-EVB](#) und einem [Olimex ESP32-PoE](#) genutzt werden. In diesem Fall wird anstelle der beiden 15poligen Pinheader eine doppelreihige fünfpolige Pinbuchse (2x5 polig, RM 2,54mm) auf der Platinenunterseite verbaut, so dass der Adapter direkt auf den zehnpoligen UEXT-Stecker von Olimex-Boards aufgesteckt werden kann.

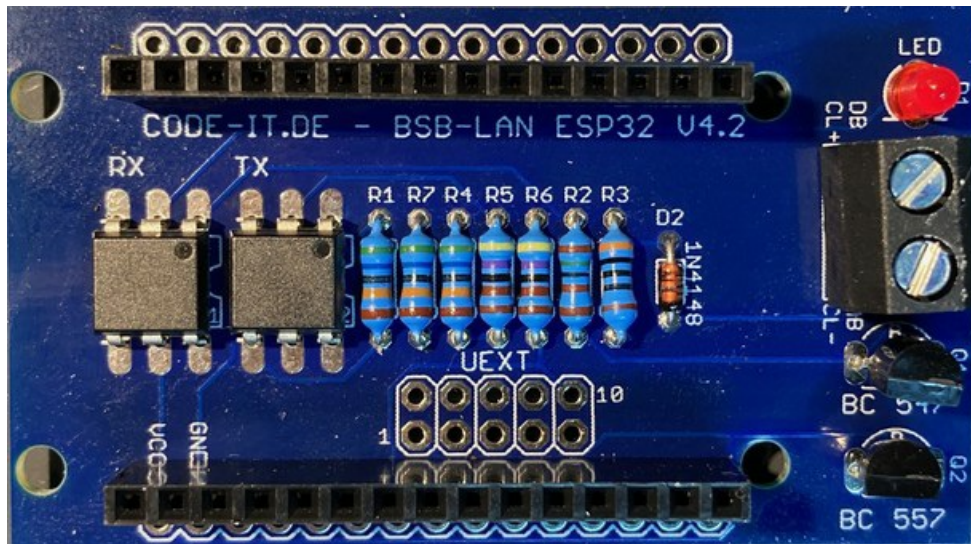




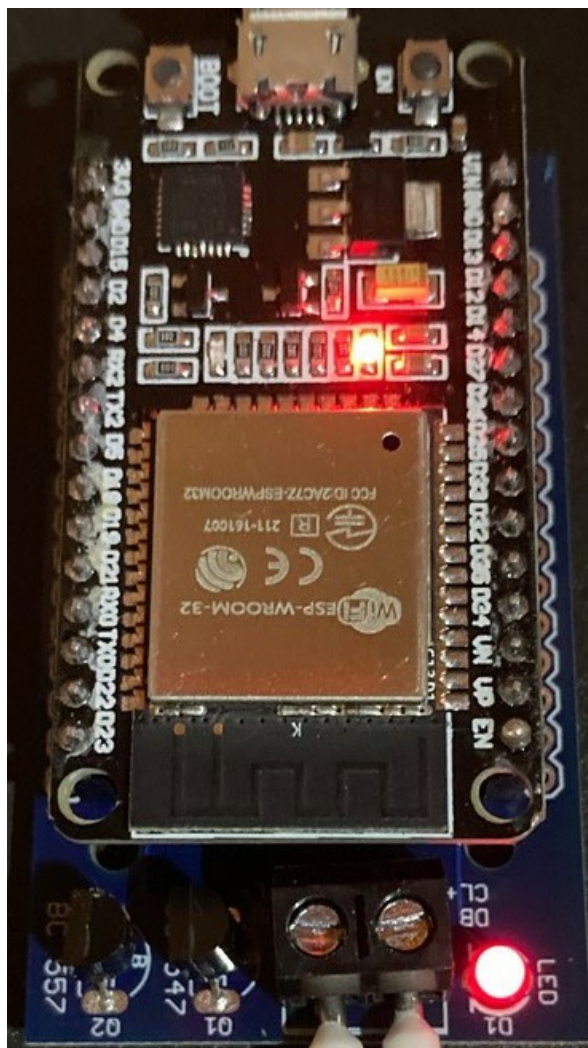
### 1.3.1.1 ESP32: NodeMCU "Joy-It"

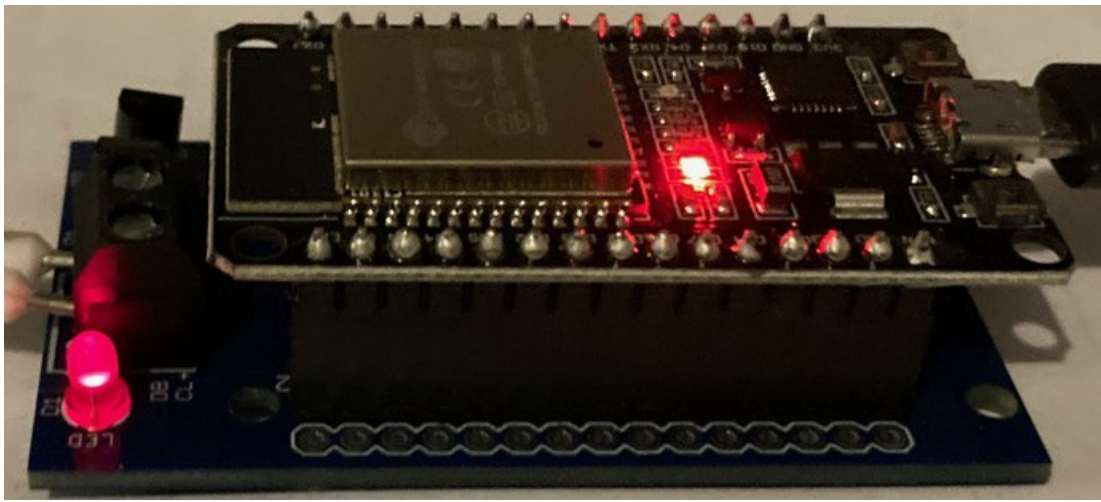
Die ESP32-Variante der BSB-LAN-Adapterplatine ist auf das 30 polige [ESP32-NodeMCU-Board von Joy-It](#) (WROOM32-Chip) ausgelegt. Es ist in Deutschland u.a. bei [Reichelt](#) erhältlich.

Für das Board ist beim Hersteller eine [Bedienungsanleitung](#) verfügbar. Dort sind sowohl das boardspezifische Pinoutschema als auch eine generelle Anleitung zur Verwendung von ESP32-Boards mit der Arduino IDE enthalten!



Die "BSB-LAN ESP32"-Adapterplatine, v4.2, bestückt für den empfohlenen NodeMCU.





Der Joy-It ESP32-NodeMCU auf dem "BSB-LAN ESP32"-Adapter.

Sollte das Joy-It-Board nicht erhältlich sein und ein anderes NodeMCU-ESP32-Board zum Einsatz kommen, so muss in jedem Fall auf zwei Dinge geachtet werden, damit der ESP32-spezifische BSB-LAN-Adapter passt:

1. Das Board *muss* ein **30 poliger** ESP32-NodeMCU sein! Es gibt auch 38 polige NodeMCUs - diese passen *nicht*!
2. Das Pinout-Schema *muss identisch* mit dem des Joy-It-Boards sein.

#### Achtung, wichtiger Hinweis

Falls das ESP32-Framework bereits in der Arduino IDE installiert ist und dir die verschiedenen ESP32-Boardvarianten angezeigt werden, überprüfe bitte im "Boardverwalter" unter "Werkzeuge/Boards", dass die **Version 2.0.2** (oder höher, falls verfügbar) installiert ist.

Sollte das Board *nicht* aufgeführt sein, so muss die ESP32-Plattform in der Arduino IDE hinzugefügt werden. Informationen hierzu findest du in [Kap. 12.1.2](#).

#### Stromversorgung:

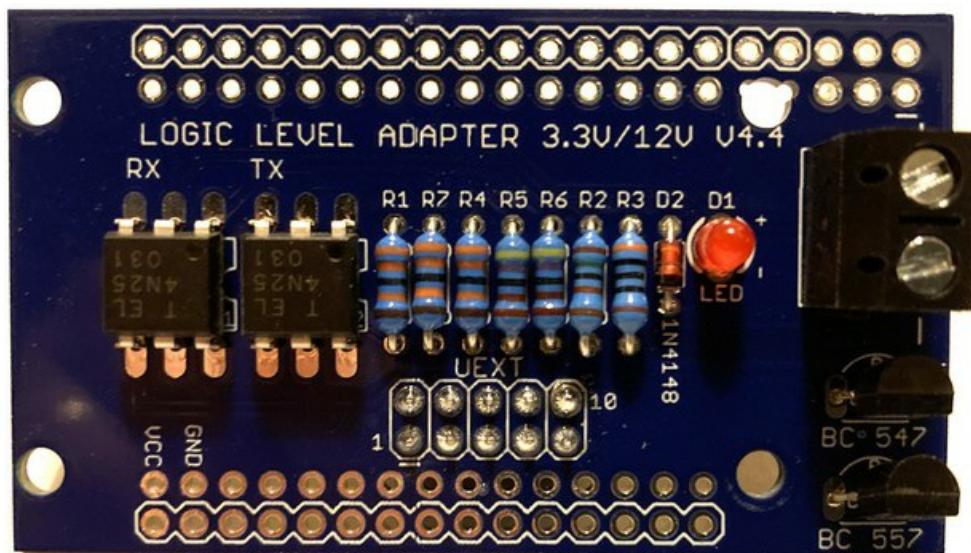
Die Stromversorgung des NodeMCU erfolgt über die microUSB-Buchse. Das Netzteil sollte mindestens 5V(DC)/1A liefern.

Sollten Probleme hinsichtlich des Datentransfers oder auch später beim Betrieb auftauchen, probiere zunächst ein anderes USB-Kabel. Es gibt Kabel, die reine Ladekabel sind und keine Datenleitung aufweisen und es gibt außerdem Kabel, die nur sehr dünne Litzen verbaut haben und aufgrund dessen im Betrieb Probleme hinsichtlich der Strom- & Spannungsversorgung bereiten können.

#### 1.3.1.2 ESP32: Olimex ESP32-EVB & ESP32-PoE

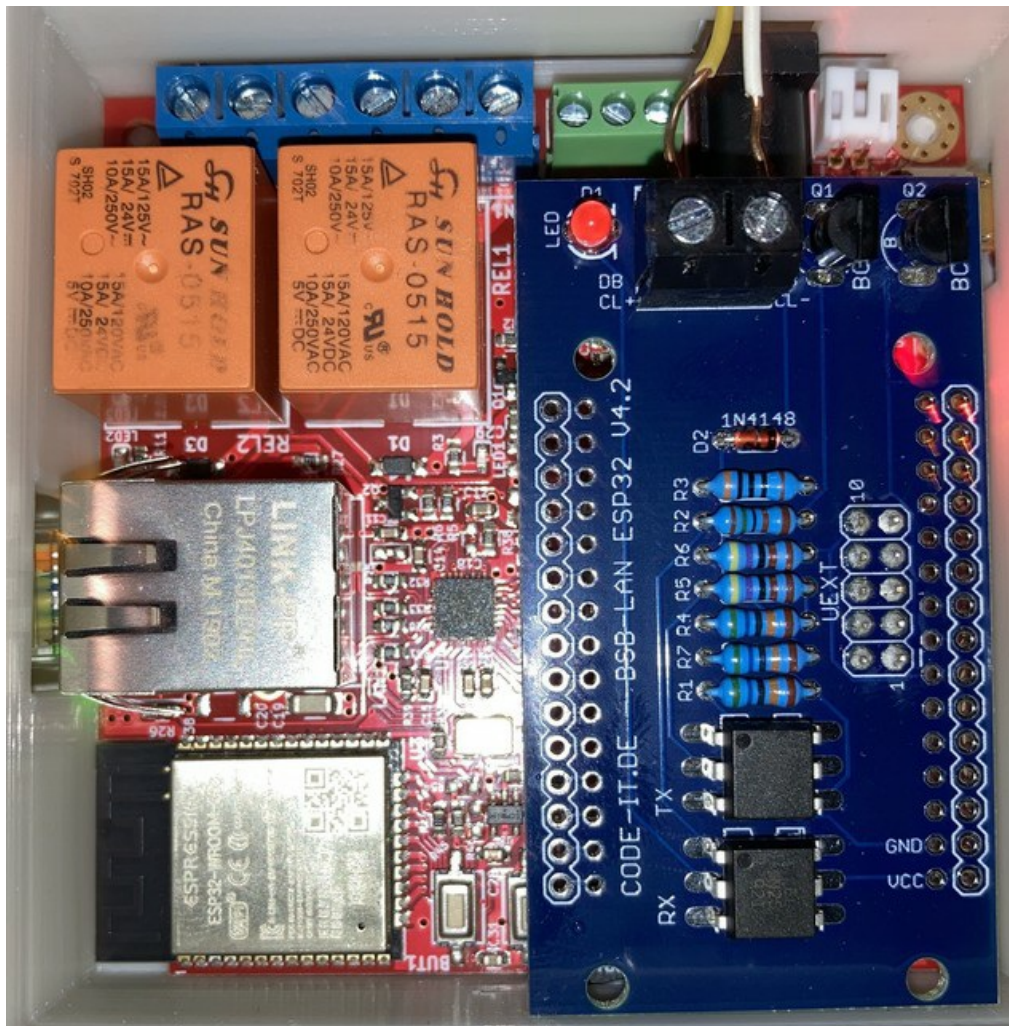
Die ESP32-Adapterversion kann außerdem mit einem [Olimex ESP32-EVB](#) und einem [Olimex ESP32-PoE](#) genutzt werden. In diesem Fall wird anstelle der beiden 15poligen Pinheader eine doppelreihige fünfpolige Pinbuchse (2x5 polig, RM 2,54mm) auf der Platineunterseite verbaut, so dass der Adapter direkt auf den zehnpoligen UEXT-Stecker von Olimex-Boards aufgesteckt werden kann.

Diese Olimex-Boardvarianten bieten neben der ESP32-basierten WLAN-Funktionalität u.a. einen LAN-Anschluss und einen microSD-Kartenleser und sind daher sehr empfehlenswert.

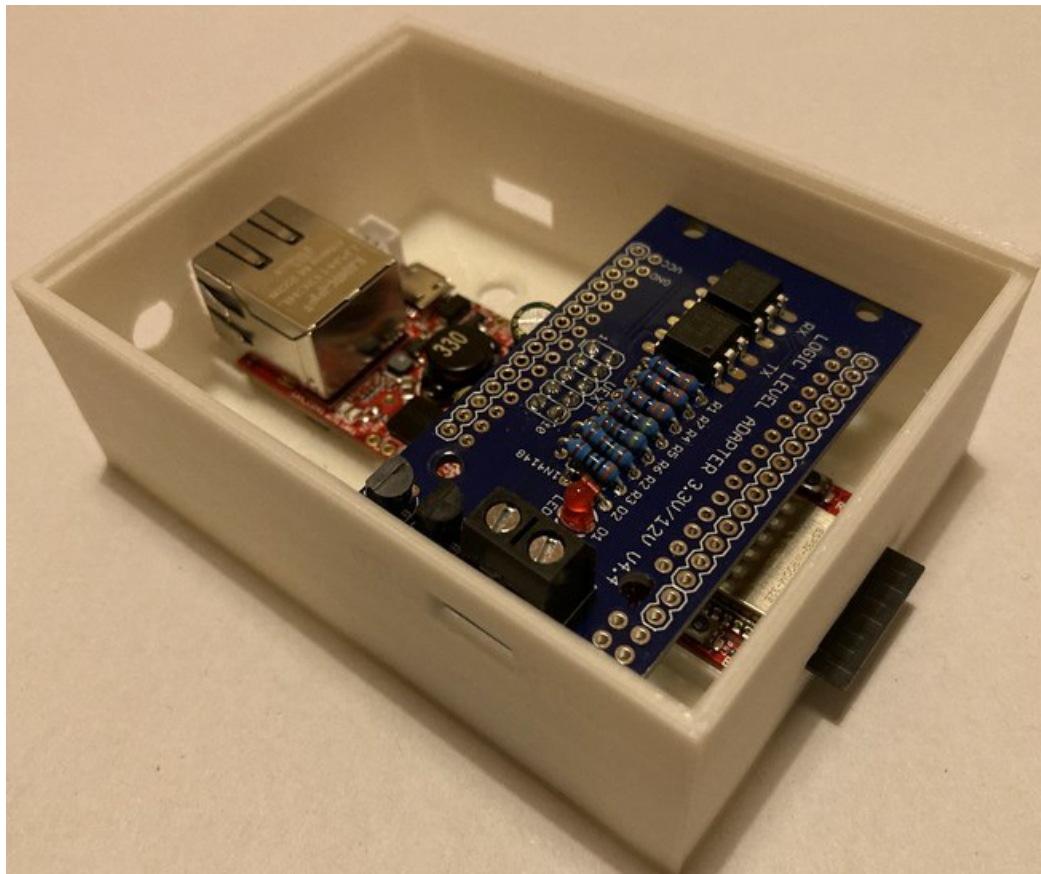




Die "BSB-LAN ESP32"-Adapterplatine, v4.4, bestückt für die empfohlenen Olimex-Boards.



Der Olimex ESP32-EVB samt aufgestecktem "BSB-LAN ESP32"-Adapter v4.2.



#### Achtung, wichtige Hinweise

**Achte beim Aufstecken des Adapterboards penibel darauf, dass die UEXT1-Buchse der Platine \* exakt in der Mitte\* der Olimex-Buchse aufgesteckt wird und alle Pins des Olimex Kontakt haben!** Ansonsten leuchtet beim korrekten Anschluss des Adapters an den Heizungsregler zwar die LED des Adapters, es ist aber kein Zugriff auf den Regler möglich.

**Ebenso ist auf die korrekte Ausrichtung der Platine zu achten (s. Foto)!**

Beim Olimex ESP32-EVB ist der Anschluss weiterer Hardware lediglich an den beiden GPIO-Pins 13 (I2C-SDA) und 16 (I2C-SCL) möglich.

Falls das ESP32-Framework bereits in der Arduino IDE installiert ist und dir die verschiedenen ESP32-Boardvarianten angezeigt werden, überprüfe bitte im "Boardverwalter" unter "Werkzeuge/Boards", dass die **Version 2.0.2** (oder höher, falls verfügbar) installiert ist. Sollte das Board *nicht* aufgeführt sein, so muss die ESP32-Plattform in der Arduino IDE hinzugefügt werden. Informationen hierzu findest du in [Kap. 12.1.2](#).

#### Stromversorgung:

Die Stromversorgung der Olimex-Boards kann auf zwei Arten erfolgen: Via Hohlsteckerbuchse oder via microUSB-Buchse. In beiden Fällen sollte das Netzteil *mindestens* 5V(DC)/1A liefern. Das Netzteil für den Betrieb via Hohlsteckerbuchse sollte einen 5.5/2.1mm (Pluspol innen) Hohlstecker aufweisen (die entspr. Produktbezeichnungen auf der Olimex-Webseite lauten "SY0605E" bzw. "SY0605E-CHINA").

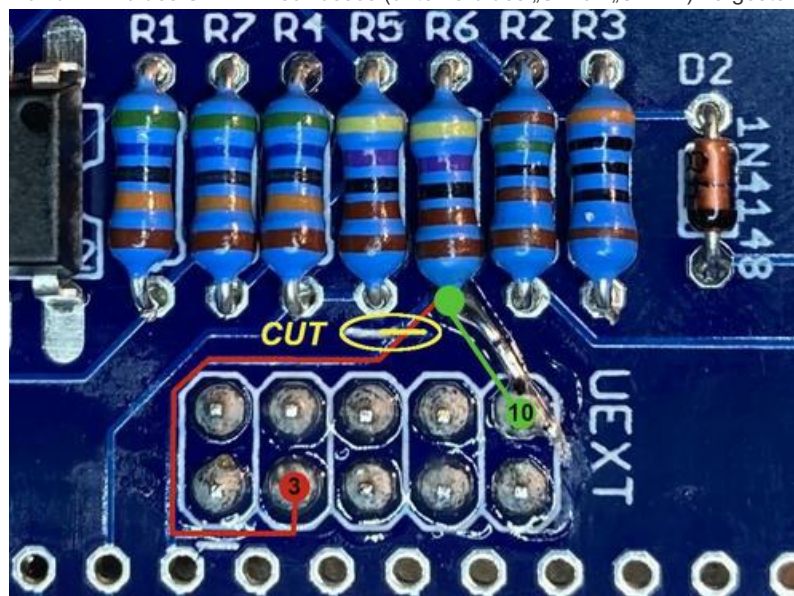
*Achtung: In Ausnahmefällen hat sich herausgestellt, dass die 1A des Netzteils nicht ausreichend waren, einen stabilen Betrieb des Boards zu gewährleisten!*

Sollten Probleme hinsichtlich des Datentransfers oder auch später beim Betrieb via microUSB-Anschluss auftauchen, probiere zunächst ein anderes USB-Kabel. Es gibt Kabel, die reine Ladekabel sind und keine Datenleitung aufweisen und es gibt außerdem Kabel, die nur sehr dünne Litzen verbaut haben und aufgrund dessen im Betrieb Probleme hinsichtlich der Strom- & Spannungsversorgung bereiten können.

#### ESP32-Adaptervariante v4.1 am Olimex nutzen:

Bei Adapterboards, die an Olimex-Boards am UEXT Anschluss verwendet werden *und* eine BSB-LAN-Board-Revision bis einschließlich 4.1 haben (und *nur* in dieser Kombination) starten diese nicht korrekt, wenn bei aufgestecktem BSB-LAN-Adapter die Stromzufuhr unterbrochen wurde. Es muss dann zusätzlich nach dem Einschalten einmal der Reset-Button gedrückt werden.

Um dieses Problem zu beheben, muss mit einem spitzen und scharfen Gegenstand (bspw. Rasierklinge/Teppichmesser/Skalpell) die vom Widerstand R6 in Richtung des UEXT-Steckers abgehende Leiterbahn (rot markiert) auf der Platine durchtrennt werden (gelb markiert). Am besten ist dies mit einem Multimeter vorher und nachher zu überprüfen, ob zwischen dem Ende von R6 und Pin 3 des UEXT-Anschlusses (k)eine Verbindung (mehr) besteht. Stattdessen muss dann eine leitende Verbindung mittels eines dünnen Drahtes zwischen diesem Ende des R6 zu Pin 10 des UEXT-Anschlusses (unterhalb des „U“ von „UEXT“) hergestellt werden (grün markiert).



BSB-LAN Boards ab der Board-Revision 4.2 sind von diesem Problem nicht mehr betroffen.

### 1.3.2 ESP32 mit Due-kompatiblen BSB-LAN-Adapter ab V3

Der bisherige Due-kompatible Adapter (ab v3) lässt sich ebenfalls mit einem ESP32 verwenden. Das EEPROM des Adapters wird hierbei nicht benötigt/verwendet und ist dementsprechend auch bei der Verkabelung nicht zu berücksichtigen.

Bei der Wahl eines ESP32 ist hier keine zwingende Einschränkung auf die zuvor genannte Joy-It-boardkompatible NodeMCU-Variante gegeben, da ohnehin eine 'lose' Verkabelung oder der Eigenbau einer kleinen Adapterplatine zur stabileren Aufnahme des BSB-LAN-Adapters und des

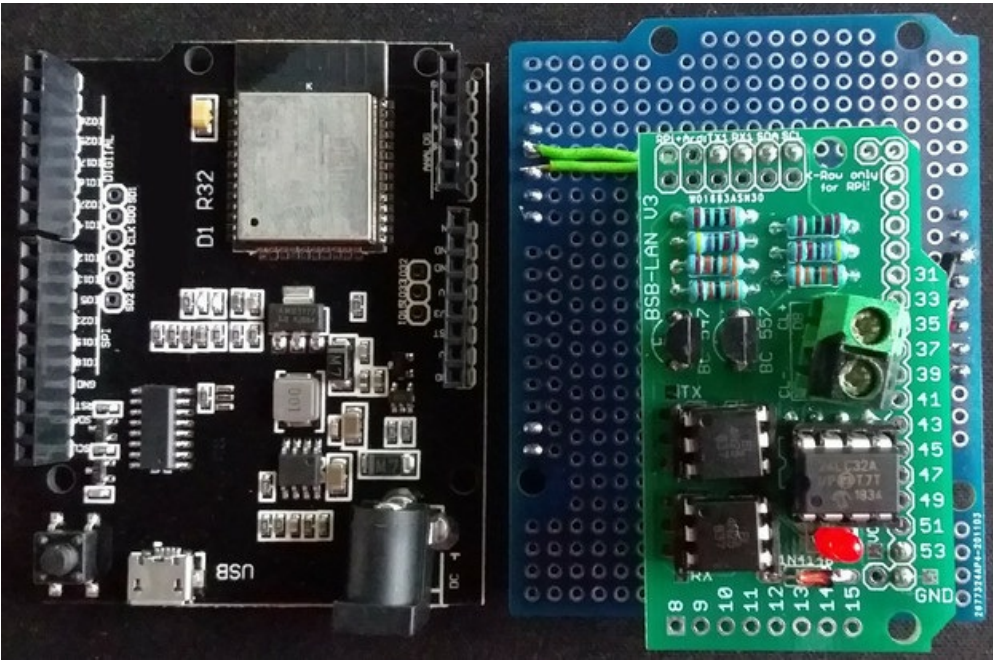


ESP32 nötig ist. Es sollte jedoch darauf geachtet werden, dass die unten angegebenen Pinnummern/-belegungen mit denen des gewählten ESP32 übereinstimmen.

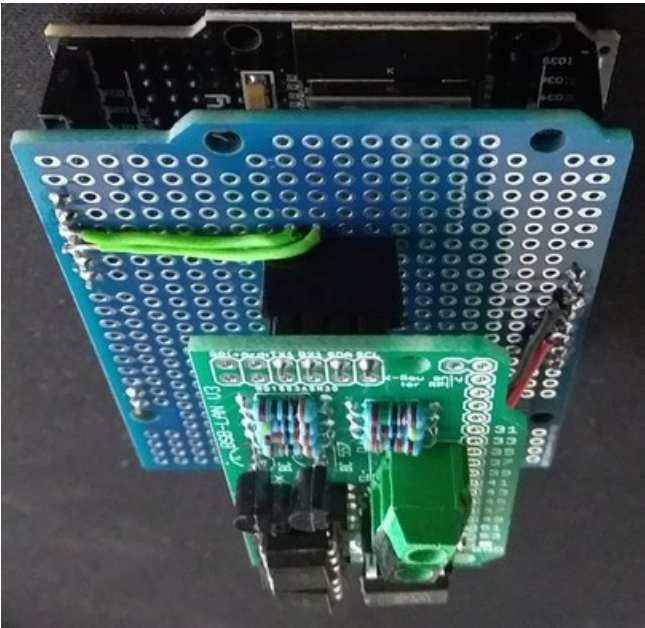
Die Verbindungen sind wie folgt vorzunehmen:

BSB-LAN-Adapter ab v3	Funktion	ESP32-Board
Pin 53	VCC (Stromversorgung Adapter)	3,3V
GND	GND (Stromversorgung Adapter)	GND
TX1	TX (Senden)	Pin 17 (TX2)
RX1	RX (Empfangen)	Pin 16 (RX2)

Beispielhaft wird im Folgenden ein "ESP32 D1 R32 Entwicklerboard" (WROOM32-Chip) in der Größe eines Arduino Uno mit einer selbstgebastelten Adapterplatine (Uno-kompatible Prototyping-Platine) für die Aufnahme des BSB-LAN-Adapters v3 (Due-Version) gezeigt. Selbstverständlich sind auch andere Varianten, wie bspw. mit einem ESP32-NodeMCU und einer entsprechend angepassten Lochrasterplatine möglich!



Links das "ESP32 D1 R32"-Board, rechts die entsprechende aufsteckbare Platine zur Aufnahme des BSB-LAN-Adapters v3 (Due-Version).



Der komplette Aufbau.

#### Hinweis

Das abgebildete ESP32 "D1 R32 Entwicklerboard" kann ich persönlich ausdrücklich NICHT empfehlen, da es offenbar deutlich schlechtere Empfangseigenschaften aufweist als andere ESP32-Boards. Obwohl der Router nur wenige Meter entfernt stand, war es mir nicht möglich, eine stabile WLAN-Verbindung aufzubauen. Auf Nachfrage beim Anbieter wurde mir dieser Eindruck bestätigt, die "Ursache dafür ist in der Bauform begründet".

#### Achtung, wichtiger Hinweis

Falls das ESP32-Framework bereits in der Arduino IDE installiert ist und dir die verschiedenen ESP32-Boardvarianten angezeigt werden, überprüfe bitte im "Boardverwalter" unter "Werkzeuge/Boards", dass die **Version 2.0.2** (oder höher, falls verfügbar) installiert ist. Sollte das Board *nicht* aufgeführt sein, so muss die ESP32-Plattform in der Arduino IDE hinzugefügt werden. Informationen hierzu findest du in [Kap. 12.1.2](#).

### 1.3.3 ESP32 mit veraltetem BSB-LAN-Adapter V2

Der (inzwischen veraltete) BSB-LAN-Adapter v2, der mit einem Arduino Mega2560 verwendet wurde, kann ebenfalls an einem ESP32 betrieben werden. So kann von der Weiterentwicklung und den neuen Funktionen der BSB-LAN-Software ab v2.x profitiert werden, ohne dass ein neuer Adapter angeschafft werden muss. Dazu müssen am Adapter selbst einige Änderungen vorgenommen werden, die im Folgenden beschrieben werden.

#### Achtung

Die nachfolgend beschriebenen Schritte zur 'Umrüstung' des Adapters auf 3,3V gelten nur für den Einsatz an einem ESP32 - an einem Due kann der Adapter v2 aufgrund des fehlenden EEPROMs nicht genutzt werden!

Um den Adapter v2 erfolgreich an einem ESP32 betreiben zu können, muss der Adapter auf den Betrieb mit 3,3V 'umgerüstet' werden. Dies ist für die Nutzung mit einem Raspberry Pi bereits vorgesehen. Nachfolgende Schritte müssen vorgenommen werden:

- Der Adapter ist *komplett* zu bestücken. Wenn der Adapter bisher nur für die Nutzung mit dem Arduino Mega 2560 bestückt ist, so müssen folgende Komponenten nachgerüstet werden:
  - 1x Widerstand 47kΩ (→ R11)
  - 2x Widerstand 10kΩ (→ R12, R13)
  - 1x Transistor BC557A (→ Q11)
  - 1x Transistor BC547A (→ Q12)
- Die Lötbrücken SJ2 und SJ3 sind durch einen Lötspunkt zu *schließen*.
- Die Lötbrücke SJ1 ist zu *entfernen*.

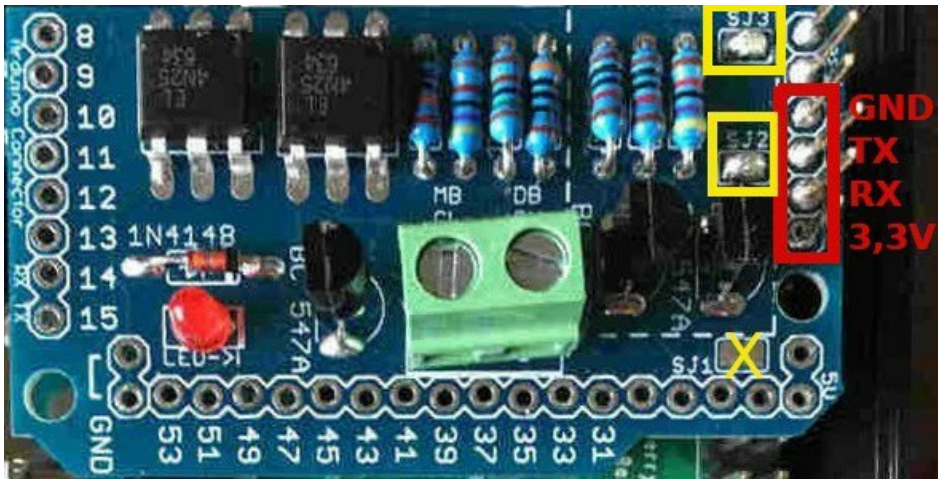
Nun ist der Adapter für den Betrieb an einem 3,3V-System vorbereitet.

Zum Anschluss an den ESP muss nun die "RasPi"-Kontaktreihe genutzt und wie folgt mit dem ESP32 verbunden werden:

BSB-LAN-Adapter v2	Funktion	ESP32-Board
Pin 06	GND (Stromversorgung Adapter)	GND
Pin 08	TX (Senden)	Pin 17 (TX2)
Pin 10	RX (Empfangen)	Pin 16 (RX2)
Pin 12	3,3V (Stromversorgung Adapter)	3,3V

Die folgende Abbildung zeigt einen entspr. bestückten Adapter v2. Das gelbe "X" bei SJ1 markiert die *entfernte* Lötbrücke (den nicht-geschlossenen Kontakt), die beiden gelben Umrandungen bei SJ2 und SJ3 markieren die *zu schließenden* Lötbrücken.





Der umgerüstete Adapter v2 für die Nutzung mit einem ESP32.

Es ist empfehlenswert, die vier Kontakte auf dem Adapter mit einer Pinleiste zu bestücken und sich eine kleine Adapterplatine aus einer Lochrasterplatine und Pinheadern aufzubauen, auf der der Adapter und das ESP32-Board aufgesteckt werden können, um einen stabilen Aufbau und eine sichere Verbindung zu gewährleisten.

#### Achtung, wichtiger Hinweis

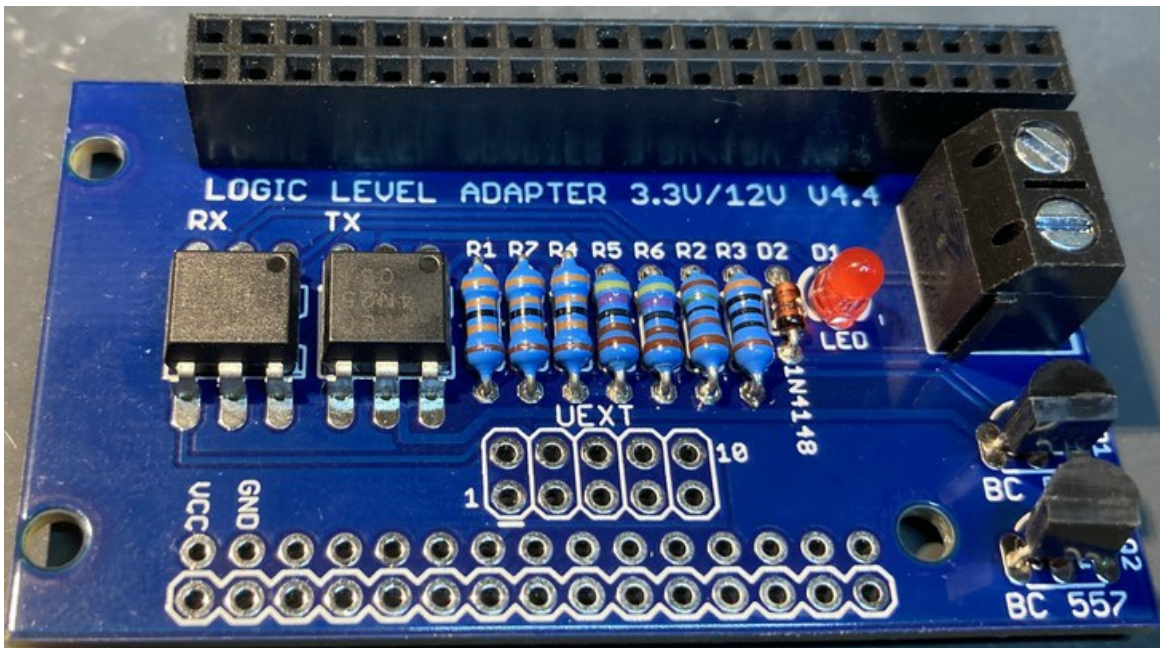
Falls das ESP32-Framework bereits in der Arduino IDE installiert ist und dir die verschiedenen ESP32-Boardvarianten angezeigt werden, überprüfe bitte im "Boardverwalter" unter "Werkzeuge/Boards", dass die **Version 2.0.2** (oder höher, falls verfügbar) installiert ist. Sollte das Board *nicht* aufgeführt sein, so muss die ESP32-Plattform in der Arduino IDE hinzugefügt werden. Informationen hierzu findest du in [Kap. 12.1.2](#).

## 1.4 Raspberry Pi

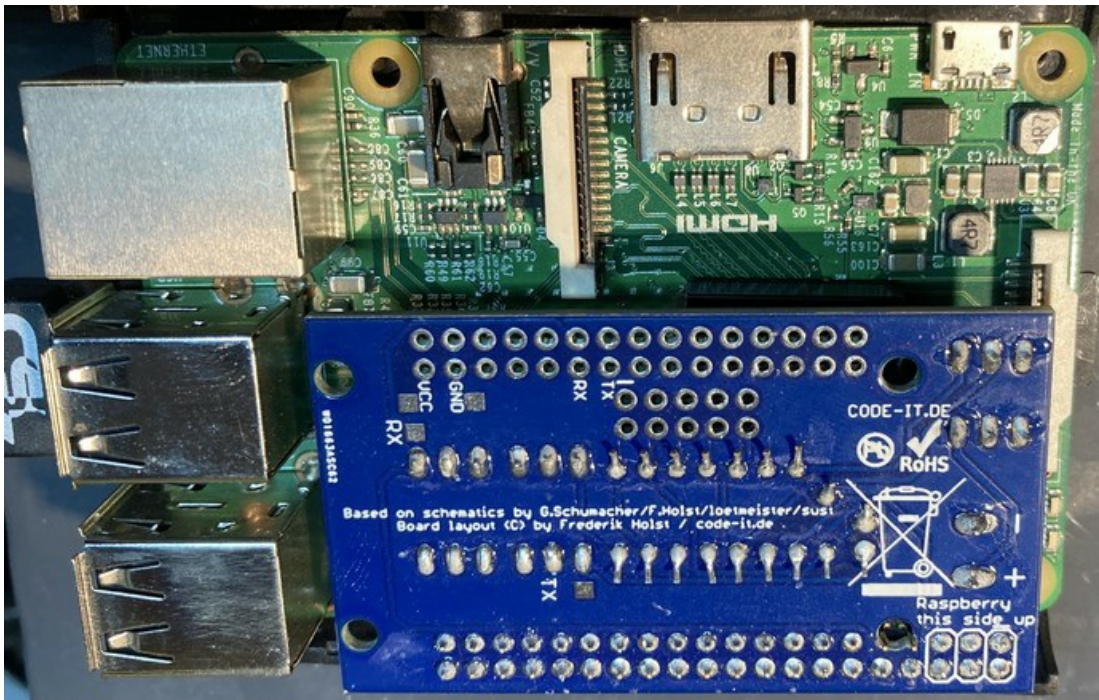
Der aktuelle ESP32-spezifische Adapter (v4.4) kann auch an einem Raspberry Pi genutzt werden.

Es sind dabei jedoch folgende Punkte zu beachten:

- Eine Verwendung der BSB-LAN-Software ist NICHT möglich (s. unten stehende Hinweise)!
- Es sollte eine entspr. lange doppelreihige Buchsenleiste verwendet werden.



Der ESP32-Adapter v4.4 mit der entspr. Buchsenleiste für einen RPi.



Der aufgesteckte ESP32-Adapter v4.4 auf einem RPi3.

<b>Achtung</b>
<b>Für die Verwendung des Adapters an einem RPi muss eine gänzlich andere Software genutzt werden: <a href="#">"bsb_gateway"</a> von J. Loehnert!</b> Für jeglichen Support in Zusammenhang mit der bsb_gateway-Software kontaktiere bitte direkt den Autor von bsb_gateway!
<b>Dieses Handbuch bezieht sich nur auf die BSB-LAN Hard- &amp; Software!</b> Es kann und wird von uns <i>keinerlei</i> Support bzgl. einer RPi-Nutzung erfolgen!
Von unserer Seite her wurde die Verwendung des Adapters mit der zuvor genannten Software lediglich auf einem RPi 3 getestet. Ob eine einwandfreie Funktion mit aktuelleren RPi-Versionen gegeben ist, können wir nicht beurteilen.
Für die Nutzung des Adapters mit einem RPi an der PPS-Schnittstelle kann das Python-Script <a href="#">PPS-monitor</a> von D. Spinellis genutzt werden.

## 1.5 Gehäuse

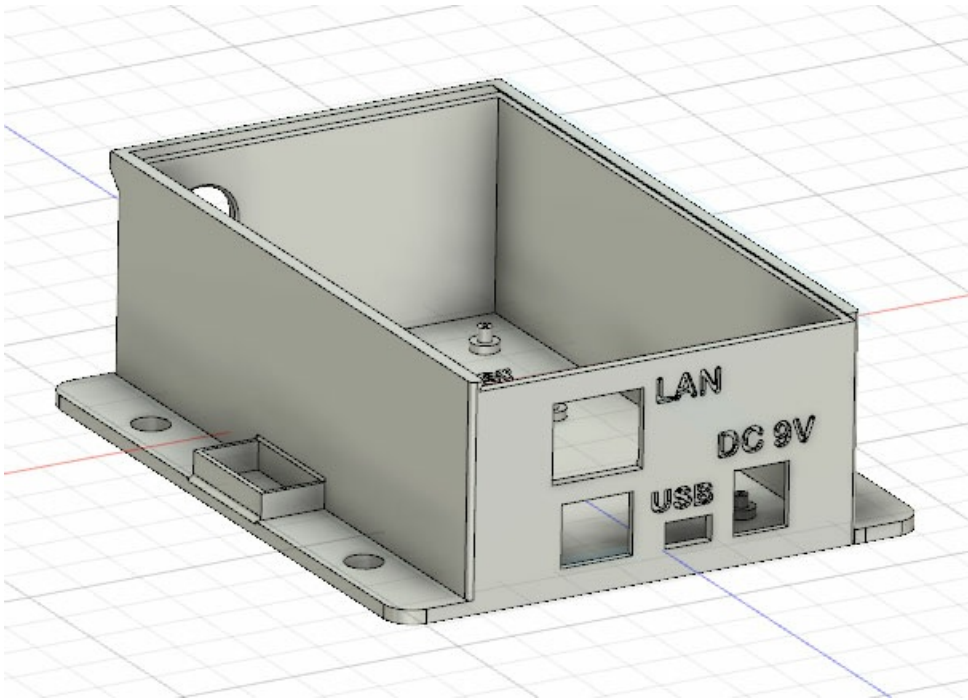
Das Angebot an verfügbaren Gehäusen für einen Arduino Due samt LAN-Shield oder auch einen NodeMCU samt Aufsteckplatine ist leider recht begrenzt, nur bei einzelnen Anbietern finden sich Kunststoff-, Plexiglas- oder Metallgehäuse. Noch knapper wird die Auswahl, wenn bspw. bei einem Due-Setup ein zusätzlich aufgestecktes Relaisboard mit untergebracht werden soll. Solltest du gezielt nach Gehäusen für den Due suchen, so wirst du u.U. nicht fündig. In dem Fall suche nach Gehäusen, die für den Arduino Mega 2560 konzipiert sind, denn er weist den gleichen Formfaktor wie der Due auf. Achte jedoch darauf, dass du möglichst ein Gehäuse wählst, das den Due samt aufgesteckten LAN-Shield aufnehmen kann. Gehäuse, die nur den Arduino aufnehmen und im Deckel Schlitz haben, so dass Shields aufgesteckt werden können, sind nicht zu empfehlen, da in dem Fall sowohl das LAN-Shield als auch der Adapter ungeschützt sind.

Neben kommerziellen Produkten und kreativen Selbstbau- und Bastellösungen bietet sich für Besitzer eines 3D-Druckers noch die Möglichkeit, ein entsprechendes Gehäuse selbst herzustellen.

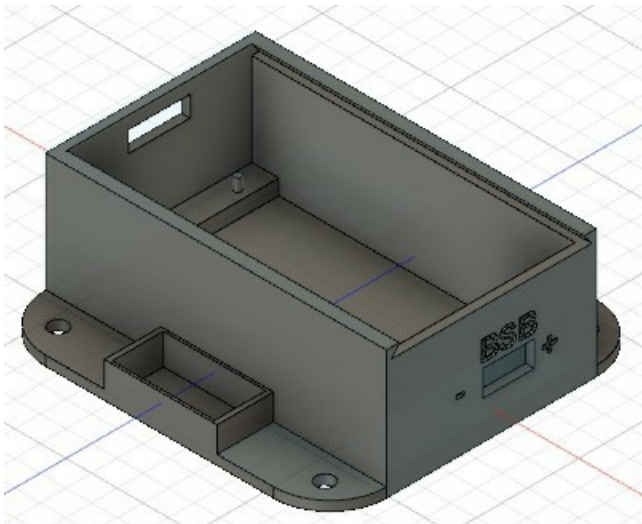
**FHEM-Forumsmitglied "EPo" war so freundlich, entsprechende STL-Dateien zu erstellen und zur Verfügung zu stellen. Vielen Dank!**

**Die STL-Dateien für den Arduino Due, den ESP32-NodeMCU sowie den Olimex ESP32-EVB samt BSB-LAN-Platine sind bereits im [GitHub-Repo von BSB-LAN](#) enthalten (Unterordner "[schematics](#)").**

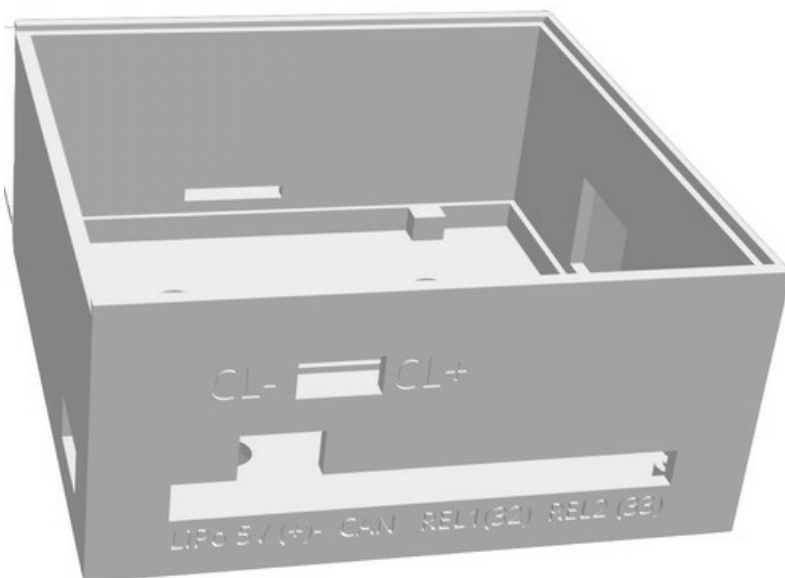




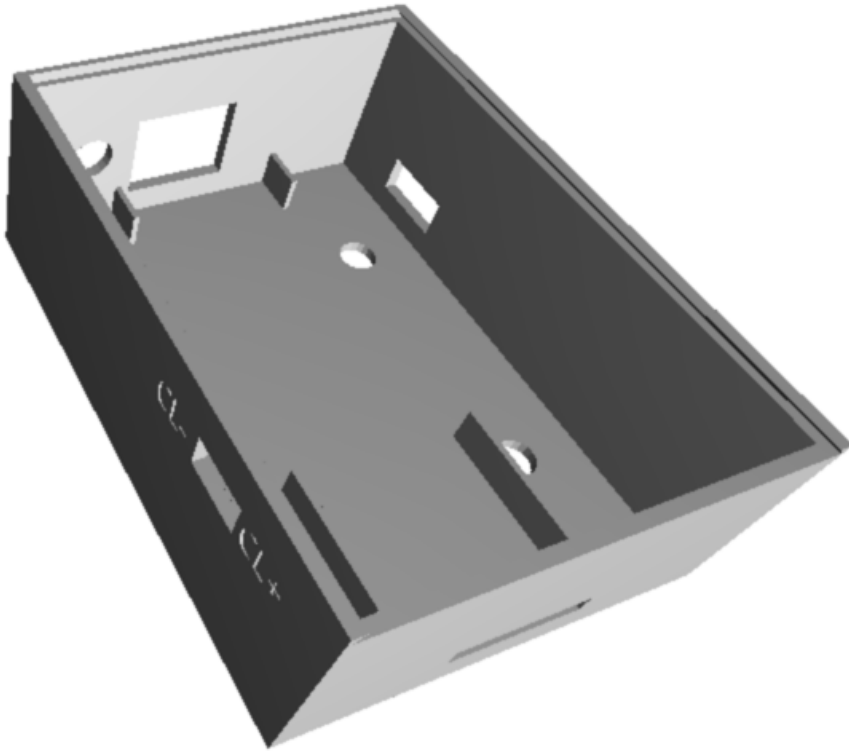
3D-Druckmodell des Gehäuses für den Arduino Due inkl. LAN-Shield und Adapter.



3D-Druckmodell des Gehäuses für den empfohlenen ESP32 NodeMCU inkl. Adapter.



3D-Druckmodell des Gehäuses für den empfohlenen Olimex ESP32-EVB inkl. Adapter.



*3D-Druckmodell des Gehäuses für den empfohlenen Olimex ESP32-PoE inkl. Adapter.*



Support me on Ko-fi

[Weiter zu Kapitel 2](#)

[Zurück zum Inhaltsverzeichnis](#)

## 2. BSB-LAN: Die Software

[Zurück zum Inhaltsverzeichnis](#)

[Zurück zu Kapitel 1](#)

## 2. BSB-LAN: Die Software

In den folgenden Kapiteln wird auf die Installation und die Konfiguration der BSB-LAN-Software eingegangen. Eine Beschreibung des Webinterface findest du hingegen in [Kap. 4](#), eine Beschreibung der Abfrage- und Steuermöglichkeiten in [Kap. 5](#) und eine Beschreibung der Spezialfunktionen in [Kap. 6](#).

### 2.1 Installation

Die BSB-LAN-Software muss zur Installation auf den jeweils verwendeten Mikrocontroller (Arduino Due oder ESP32) geflasht werden. Dies kann bspw. mittels der "Arduino IDE" erfolgen, selbstverständlich können aber auch andere Programme wie bspw. "PlatformIO" oder "Visual Studio Code" genutzt werden.

#### Hinweis

In diesem Handbuch wird davon ausgegangen, dass die Arduino IDE genutzt wird. Sämtliche Beschreibungen und Bezeichnungen beziehen sich daher auf die Arduino IDE.

Solltest du Anfänger und mit der Arduino IDE noch nicht vertraut sein, so findest du eine Beschreibung zur Installation und Konfiguration der Arduino IDE in [Kap. 12](#).

Je nach verwendeter Plattform (Arduino Due oder ESP32) unterscheiden sich die notwendigen Einstellungen der Arduino IDE. So müssen die entsprechenden Boardtypen installiert und ausgewählt sein, die Einstellungen plattformspezifisch angepasst werden etc. Auf diese Einstellungen wird im Folgenden eingegangen. Dabei wird davon ausgegangen, dass die nötigen Bibliotheken für die jeweilige Plattform bereits installiert sind. Sollte dies nicht der Fall sein, so findest du Informationen hierzu in [Kap. 12](#).

Darüber hinaus gibt es bei der Installation auf dem ESP32 noch weitere Dinge zu beachten, die im entspr. Kapitel ebenfalls behandelt werden.

#### 2.1.1 Installation auf dem Due

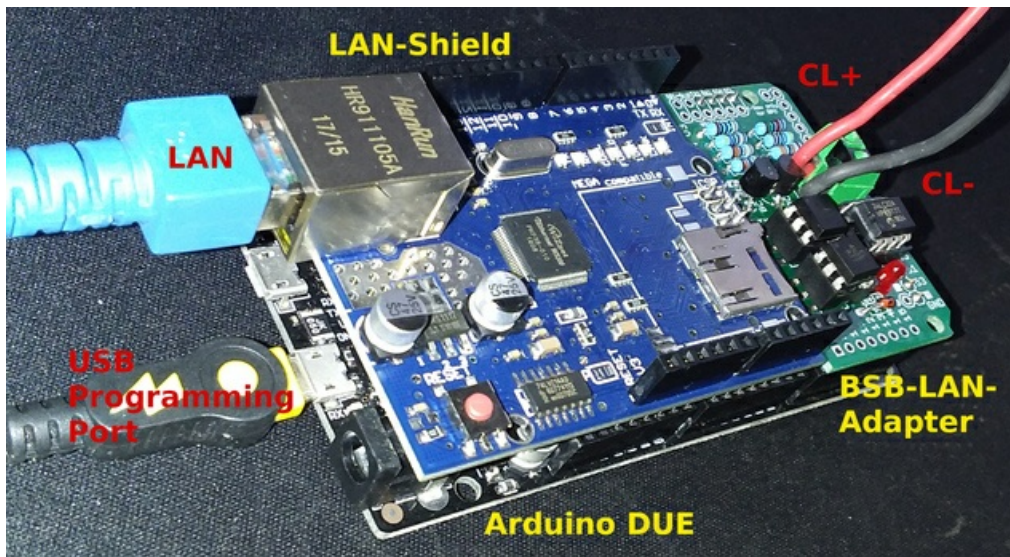
Im Folgenden wird die Installation der BSB-LAN-Software auf einem Arduino Due beschrieben. Die Beschreibung bezieht sich dabei auf die Verwendung der Arduino IDE. Mit den Voreinstellungen der BSB-LAN-Software wird für die IP-Adressvergabe DHCP genutzt. Solltest du dies nicht wünschen und eine feste IP vergeben wollen, so lies bitte [Kap. 2.2.2](#) und passe die Datei *BSB\_LAN\_config.h* vor dem Flashen an!

#### Hinweis

Solltest du Windows benutzen, so ist evtl. noch eine zusätzliche Treiberinstallation nötig. Auf der Seite <https://www.arduino.cc/en/Guide/ArduinoDue> findest du weitere Informationen.

Führe die folgenden Schritte aus:

1. Verbinde das Arduino-Setup mit einem USB-Kabel mit deinem Computer. Nutze dabei den 'Programming Port' des Due, das ist der 'mittlere' USB-Port, der neben der Netzteilbuchse platziert ist. Sowohl das LAN-Shield als auch der BSB-LAN-Adapter sollte zuvor bereits auf den Due gesteckt sein, dies ist jedoch nicht zwingend nötig.

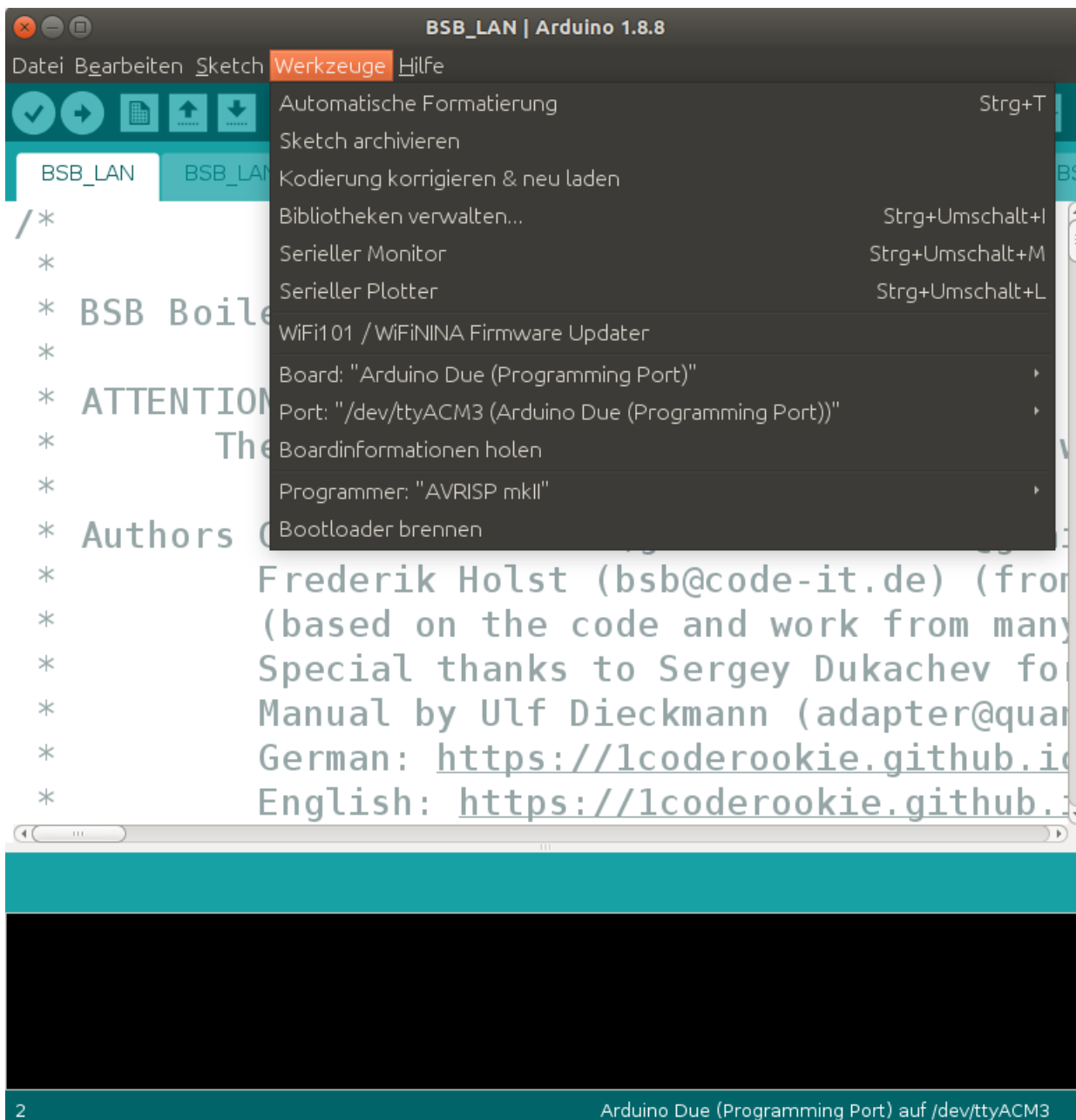


Das komplette Setup (Arduino Due + LAN-Shield + BSB-LPB-LAN-Adapter v3) inklusive der entsprechenden Kabel.

2. Downloade die [aktuelle BSB-LAN-Version](#) und entpacke die heruntergeladene Datei *BSB-LAN-master.zip*.
3. Wechsle in den Ordner "BSB-LAN-master"/"BSB\_LAN" und benenne die Dateien *BSB\_LAN\_custom\_defs.h.default* in ***BSB\_LAN\_custom\_defs.h*** sowie *BSB\_LAN\_config.h.default* in ***BSB\_LAN\_config.h*** um!
4. Öffne den BSB\_LAN-Sketch mittels eines Doppelklicks auf die Datei *BSB\_LAN.ino* im BSB\_LAN-Ordner. Die dazugehörigen Dateien *BSB\_LAN\_config.h*, *BSB\_LAN\_defs.h* sowie *BSB\_LAN\_custom\_defs.h* werden automatisch mit geladen.
5. Wähle "Arduino Due (Programming Port)" unter Tools/Board bzw. Werkzeuge/Board.

| Hinweis | |-----| | Sollte das Board nicht aufgeführt sein, so muss der Atmel SAM Core hinzugefügt werden. Informationen hierzu findest du in [Kap. 12.1.1](#). |

6. Wähle den korrekten Seriellen Port, an dem der Due am Rechner angeschlossen ist, unter Werkzeuge/Port aus.



Die Einstellungen für den Due in der Arduino IDE.

7. Solltest du BSB-LAN mittels Anpassung der Datei `BSB_LAN_config.h` konfigurieren wollen (s. [Kap. 2.2.2](#)), so tue dies bitte jetzt.
8. Starte den Flashvorgang und lade den Sketch mittels Klick auf "Sketch/Upload" bzw. "Sketch/Hochladen" auf den Arduino Due.
9. Nach Beenden des Flashvorgangs starte den Seriellen Monitor der Arduino IDE und beobachte die Ausgaben, die beim Start des Arduino Due erfolgen. Dort wird u.a. auch die IP ausgegeben, die dem Setup bei Verwendung von DHCP zugeteilt wird.

**Herzlichen Glückwunsch - du hast BSB-LAN installiert!**

Fahre nun mit der [Konfiguration der BSB-LAN-Software](#) und danach mit [dem Anschluss und der Inbetriebnahme des Setups](#) fort.

#### Wichtiger Hinweis

Um kompletten Zugriff auf alle Parameter deines Reglers zu erhalten, muss nach Abschluss aller hier genannten Schritte und dem erfolgreichen Flashen eine reglerspezifische Datei `BSB_LAN_custom_defs.h` erstellt und im Anschluss BSB-LAN mit dieser Datei neu installiert werden! Bitte beachte daher **zwingend** das [Kap. 3.3](#) und führe die dort genannten Schritte aus!

## 2.1.2 Installation auf dem ESP32

Im Folgenden wird die Installation der BSB-LAN-Software auf einem ESP32 beschrieben. Die Beschreibung bezieht sich dabei auf die Verwendung der Arduino IDE. Mit den Voreinstellungen der BSB-LAN-Software wird für die IP-Adressvergabe DHCP genutzt. Solltest du dies nicht wünschen und eine feste IP vergeben wollen, so lies bitte [Kap. 2.2.2](#) und passe die Datei `BSB_LAN_config.h` vor dem Flashen an!

## Hinweis

Sollte das ESP32-Board nicht von deinem Betriebssystem erkannt werden, so ist evtl. noch eine zusätzliche Treiberinstallation für den vom Board verwendeten USB-Chip nötig.

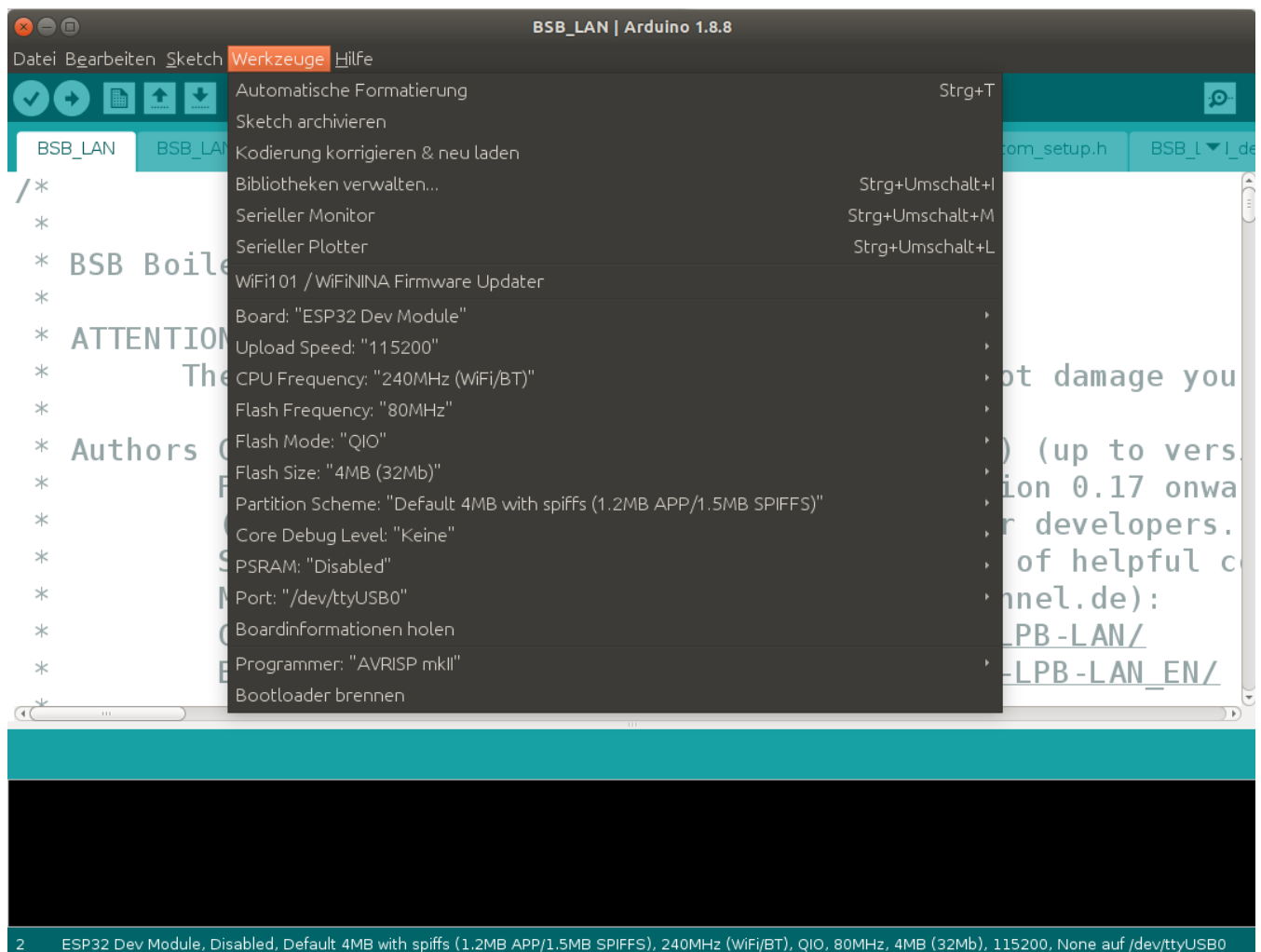
Führe die folgenden Schritte aus:

1. Verbinde dein ESP32-Board mit einem USB-Kabel mit deinem Computer. Den BSB-LAN-Adapter kannst du vorher bereits auf bzw. unter dein ESP32-Board gesteckt haben, dies ist jedoch nicht zwingend nötig.
2. Downloade die [aktuelle BSB-LAN-Version](#) und entpacke die heruntergeladene Datei *BSB-LAN-master.zip*.
3. Wechsle in den Ordner "BSB-LAN-master"/"BSB\_LAN" und benenne die Dateien *BSB\_LAN\_custom\_defs.h.default* in ***BSB\_LAN\_custom\_defs.h*** sowie *BSB\_LAN\_config.h.default* in ***BSB\_LAN\_config.h*** um!
4. Öffne den BSB\_LAN-Sketch mittels eines Doppelklicks auf die Datei *BSB\_LAN.ino* im BSB\_LAN-Ordner. Die dazugehörigen Dateien *BSB\_LAN\_config.h*, *BSB\_LAN\_defs.h* sowie *BSB\_LAN\_custom\_defs.h* werden automatisch mit geladen.
5. Wähle den entspr. ESP32-Boardtyp unter Tools/Board bzw. Werkzeuge/Board aus:
  - Für den in diesem Handbuch empfohlenen **Joy-It ESP32-NodeMCU** (oder identische Clones mit einem "ESP32-WROOM"-Chip) lautet der passende Boardtyp "ESP32 Dev Module".
  - Für das empfohlene **Olimex ESP32-EVB & ESP32-PoE** wähle bitte den gleichnamigen Eintrag aus der Liste aus.

| Achtung, wichtiger Hinweis | |:-----| | Falls das ESP32-Framework bereits installiert ist und dir die verschiedenen ESP32-Boardvarianten angezeigt werden, überprüfe bitte im "Boardverwalter" unter "Werkzeuge/Boards", dass die **Version 2.0.2** (oder höher, falls verfügbar) installiert ist.

Sollte das Board *nicht* aufgeführt sein, so muss die ESP32-Plattform in der Arduino IDE hinzugefügt werden. Informationen hierzu findest du in [Kap. 12.1.2](#). |
6. Wähle den korrekten Seriellen Port, an dem das ESP32-Board am Rechner angeschlossen ist, unter Werkzeuge/Port aus.
7. Stelle die Übertragungsgeschwindigkeit/Baudrate auf 115200 ein (Achtung: In der Arduino IDE ist bei ESP32-Boards i.d.R. 921600 voreingestellt!).
8. "Partition Scheme": Je nach gewähltem ESP32-Board sind hier entspr. Einstellungen vorzunehmen.
  - Für den empfohlenen **ESP32-NodeMCU** wähle bitte die Variante "Default 4MB with spiiffs (1.2BM APP/1.5MB SPIFFS)" aus. Der folgende Screenshot zeigt die gesamten Einstellungen für den **ESP32-NodeMCU**.





Die Einstellungen für den **ESP32-NodeMCU** in der Arduino IDE.

- Für die empfohlenen **Olimex-Boards** wähle bitte die Variante "Minimal SPIFFS (Large APPS with OTA)" aus.

9. Klicke nun auf den Reiter für die Datei `BSB_LAN_config.h` und passe **zwingend** die folgenden Einstellungen an:

- Aktiviere das Definement `#define WIFI` in der Datei `BSB_LAN_config.h`, wenn du WiFi verwenden willst. Solltest du ein Olimex-Board verwenden und den LAN-Anschluss nutzen wollen, lasse das Definement bitte auskommentiert: `//#define WIFI`.
- Trage bei der Verwendung von WLAN die Zugangsdaten für dein WLAN-Netzwerk bei den Einträgen  

```
char wifi_ssid[32] = "YourWiFiNetwork";
```

 sowie  

```
char wifi_pass[64] = "YourWiFiPassword";
```

 ein.

10. Solltest du BSB-LAN mittels Anpassung der Datei `BSB_LAN_config.h` konfigurieren wollen (s. [Kap. 2.2.2](#)), so tue dies bitte jetzt.

11. Starte den Flashvorgang mittels Klick auf "Sketch/Upload" bzw. "Sketch/Hochladen".

12. Nach Beenden des Flashvorgangs starte den Seriellen Monitor der Arduino IDE und beobachte die Ausgaben, die beim Start des ESP32 erfolgen. Dort wird u.a. auch die IP ausgegeben, die dem Setup bei Verwendung von DHCP zugeteilt wird.

#### Hinweise

Wenn der ESP32 sich nicht mit dem konfigurierten WLAN verbinden kann, richtet er seinen eigenen Accesspoint "BSB-LAN" mit dem Passwort "BSB-LPB-PPS-LAN" für 30 Minuten ein. Danach wird er neu starten und erneut versuchen, eine Verbindung zum eingerichteten WLAN-Netzwerk herzustellen.

Obwohl die Logging-Funktion auch mit dem ESP32 funktioniert, ist es nicht empfehlenswert, diese Funktion aufgrund des Verschleißes des Flash-Speichers übermäßig zu nutzen. Sollte ein **Olimex-Board** zum Einsatz kommen, so kann anstelle des SPIFF-Flashspeichers eine microSD-Karte genutzt werden. Die Verwendung ist in der Datei `BSB_LAN_config.h` zu aktivieren (Definement `#define ESP32_USE_SD`).

**Herzlichen Glückwunsch - du hast BSB-LAN installiert!**

Fahre nun mit der [Konfiguration der BSB-LAN-Software](#) und danach mit [dem Anschluss und der Inbetriebnahme des Setups](#) fort.

## Wichtiger Hinweis

Um kompletten Zugriff auf alle Parameter deines Reglers zu erhalten, muss nach Abschluss aller hier genannten Schritte und dem erfolgreichen Flashen eine reglerspezifische Datei `BSB_LAN_custom_defs.h` erstellt und im Anschluss BSB-LAN mit dieser Datei neu installiert werden! Bitte beachte daher **zwingend** das [Kap. 3.3](#) und führe die dort genannten Schritte aus!

## 2.1.3 Updates

Ein Updaten der BSB-LAN-Software erfolgt durch das gewohnte Flashen der neuen Version ( [Download als ZIP-File](#), per git o.ä.), wie es in den vorherigen Installationskapiteln beschrieben ist. Bitte beachte die folgenden Hinweise!

Für ESP32-basierte Boards (Olimex, NodeMCU) kann alternativ auch ein **OTA-Update** ("OverTheAir"-Update) erfolgen (diese Funktion ist NICHT mit dem Arduino DUE nutzbar!). Hierzu muss in der Webkonfig oder der Datei `BSB_LAN_config.h` die entspr. OTA-Funktion aktiviert werden. Die entspr. Firmware-Datei `BSB_LAN_ino.bin` kann man in der Arduino IDE unter "Sketch / Kompilierte Binärdatei exportieren..." erstellen lassen. Der Upload der Datei erfolgt dann per Browser auf Port 8080 der BSB-LAN-IP ( `http://<BSB-LAN-IP>:8080` bzw. `http://bsb-lan:8080` bei aktiviertem MDNS).

## Hinweise

**Bei einem Update auf v3.x nutze bitte keinerlei bereits bestehenden Dateien - installiere BSB-LAN bitte *komplett* neu! Beachte außerdem die notwendige Erstellung einer reglerspezifischen `BSB_LAN_custom_defs.h` - das Vorgehen ist in [Kap. 3.3](#) beschrieben.**

Solltest du in der Datei `BSB_LAN_config.h` bestimmte Änderungen bei der neu zu flashenden Version vorgenommen haben, wie bspw. die Zugangsdaten für dein WLAN oder eine fest vergebene IP, die dann nach dem Flashen offenbar nicht übernommen wurden, so liegt dies i.d.R. daran, dass die alten Einstellungen aus dem EEPROM gelesen wurden.

Um die neuen Einstellungen wirksam werden zu lassen, stelle in der [Webkonfiguration](#) die Einstellung "Konfiguration aus EEPROM lesen" einmal auf "Aus", speichere die Änderung und flashe nochmal neu.

Danach sollten die neuen Einstellungen wirksam geworden sein, weil BSB-LAN diese nun aus der Datei `BSB_LAN_config.h` und nicht aus dem EEPROM eingelesen hat.

Nach erfolgreicher Überprüfung stelle "Konfiguration aus EEPROM lesen" wieder auf "Ein".

Die bestehende und ggf. angepasste Datei `BSB_LAN_config.h` kann in der Regel bei einem Update auf eine neuere Version zwar übernommen werden, es jedoch ratsam, auch hier die jeweils aktuelle Datei `BSB_LAN_config.h.default` anstelle der bestehenden Datei `BSB_LAN_config.h` zu verwenden. Dazu muss die .default-Datei wie gehabt umbenannt und ggf. den vorherigen Einstellungen entspr. angepasst werden. So kann man sicher gehen, dass man ein komplettes Update der BSB-LAN-Software vorgenommen hat.

Wenn der Adapter an den Bus des Heizungsreglers angeschlossen ist, so kann er angeschlossen bleiben, wenn der Due/ESP32 erneut geflasht werden soll. Es besteht keine Notwendigkeit den Adapter vom Regler abzuklemmen, wenn man BSB-LAN updaten möchte.

Wenn in der Konfig die Funktion "überprüfe auf Updates" aktiviert ist, so wird bei der Startseite von BSB-LAN eine etwaige neuere Version angezeigt. Dies beinhaltet jedoch auch Entwicklerversionen (also nicht nur 'stable releases') - möchtest du hingegen nur 'stable releases' einsetzen, so musst du manuell auf der Projektseite nachsehen, ob eine neuere Version verfügbar ist.

## 2.2 Konfiguration

Die BSB-LAN-Software kann den individuellen Ansprüchen entsprechend konfiguriert werden.

Die Konfiguration kann dabei auf zwei Arten erfolgen: Mittels Anpassen der Datei `BSB_LAN_config.h` sowie mittels Webinterface (aufrufbar mittels Klick auf "Einstellungen" auf der BSB-LAN-Seite oder per direktem URL-Befehl `/c`).

Im Folgenden werden die Konfigurationsmöglichkeiten eingehender erklärt. Die Beschreibungen in Kap. 2.2.2 sind i.d.R. ausführlicher, daher ist es sinnvoll, beide Kapitel eingehend zu studieren.

Eine Übersicht der BSB-LAN-Einstellungen kann mit dem URL-Befehl `/co` aufgerufen werden.

### 2.2.1 Konfiguration mittels Webinterface

Die Seite der Webkonfiguration ist mittels Klick auf "Einstellungen" auf der BSB-LAN-Seite oder per direktem URL-Befehl `/c` aufrufbar. Die einzelnen Einstellungsmöglichkeiten sind zwar im Prinzip selbsterklärend, trotzdem seien die einzelnen Punkte hier nochmals mit einer kurzen Erklärung aufgeführt.

Für eine u.U. ausführlichere Erklärung zu den einzelnen Funktionen sieh bitte im [Kap. 2.2.2](#) nach.

Eine reine Übersicht der Einstellungen kann mittels des URL-Befehls `/co` angezeigt werden.

Die Übersicht der Webkonfiguration gliedert sich in drei Spalten:

Einstellungen		
Generell	Erweiterte Einstellungen anzeigen	Aus
Generell	Konfiguration aus EEPROM lesen	Ein
Generell	Schreibzugriff (Ebene)	Aus
Generell	Auf Updates überprüfen	Ein
Generell	OTA Update	Aus
Bus	RX Pin Nummer	0
Bus	TX Pin Nummer	0
Bus	Typ	BSB
Bus	Eigene Adresse	CC

- In der linken Spalte wird der Übersichtlichkeit halber eine grobe Kategorie angezeigt (bspw. "Generell", "Bus" etc.), so dass bereits auf den ersten Blick die Zuordnung des jeweiligen Eintrages ersichtlich ist.
- In der mittleren Spalte wird die Funktion genannt.
- In der rechten Spalte befindet sich das zugehörige Feld, das den derzeitigen Eintrag bzw. die Einstellung zeigt. Dabei werden die Einträge aus der Datei `BSB_LAN_config.h` übernommen, d.h. auch bei deaktivierten Funktionen sind die Voreinstellungen sichtbar, so dass deutlich wird, wie bspw. Parameter einzutragen sind. Je nach Art der Einstellung wird entweder ein PullDown-Menü mit den verfügbaren Einstellungen oder lediglich ein Feld angezeigt.

**Wichtig**

Zum Übernehmen geänderter Einstellungen muss schließlich unten auf den Button "Parameter speichern" geklickt werden!

Im Folgenden nun die tabellarische Übersicht der Funktionen mit den (Vor-)Einstellungen und den entspr. Erklärungen (auf die Nennung der linken Spalte "Kategorie" muss an dieser Stelle aus Platz- und Darstellungsgründen leider verzichtet werden):

Funktion	(Vor-)Einstellung	Erklärung
Erweiterte Einstellungen anzeigen	Aus	Anzeige der erweiterten Einstellungen von BSB-LAN (Aus/Ein). Für Zugriff auf sämtliche Einstellungsmöglichkeiten von BSB-LAN muss "Ein" ausgewählt (und anschließend unten auf "Parameter speichern" geklickt) werden.
Konfiguration aus EEPROM lesen	Ein	Liest die gespeicherte Konfiguration aus dem EEPROM beim Start des Due aus (Aus/Ein). Diese Einstellungen können von den Voreinstellungen abweichen, die in der Datei <code>BSB_LAN_config.h</code> hinterlegt wurden. <i>Sollen die im EEPROM gespeicherten Einstellungen bspw. bei einem Update überschrieben werden, so ist vor dem Flashen auf "Aus" zu stellen und die Einstellung zu speichern!</i> Wenn die Einstellung auf „Aus“ ist, werden Änderungen nur bis zum Neustart des Due aktiv bleiben.
Schreibzugriff (Ebene)	Aus	Schreibzugriff des Adapters auf den Heizungsregler (Aus/Standard/Komplett). <b>Soll Schreibzugriff auf den Heizungsregler gewährt werden, so ist es empfehlenswert, die Einstellung 'Standard' zu wählen, hierbei sind nahezu alle verfügbaren Parameter schreibbar.</b> Im Unterschied zu 'Komplett' sind jedoch einige funktionskritische Parameter nicht veränderbar, die reglerintern nochmals geschützt vorliegen. <i>Die Einstellung 'Komplett' sollte daher nur in Ausnahmefällen und mit Bedacht sowie einem sehr guten Kenntnisstand über die Reglerfunktionalität gewählt werden!</i>
Auf Updates überprüfen	Aus	Automatisches Überprüfen auf Updates von BSB-LAN (Aus/Ein)

Funktion	(Vor-)Einstellung	Erklärung
OTA Update	Aus	OTA-Updatefunktion für ESP32-basierte Boards (OTA = Over The Air) deaktiviert (Aus) / aktiviert (Ein). Zur weiteren Vorgehensweise für OTA-Updates lies bitte <a href="#">Kap. 2.1.3 Updates</a> .
RX Pin Nummer	0	0 = Autoselect. Falls ein anderer Pin als der voreingestellte RX-Pin (s. Datei <i>BSB_LAN_config.h</i> ) genutzt wird, ist dieser hier einzutragen.
TX Pin Nummer	0	0 = Autoselect. Falls ein anderer Pin als der voreingestellte TX-Pin (s. Datei <i>BSB_LAN_config.h</i> ) genutzt wird, ist dieser hier einzutragen.
Typ	BSB	Verwendeter Bustyp (BSB/LPB/PPS)
Eigene Adresse	66	Eigene Adresse des Adapters
Zieladresse	0	Zieladresse für die Abfragen
PPS: PPS-Modus	Passiv	Nur PPS: Benutzer, die den Adapter an der PPS-Schnittstelle verwenden, müssen zwei Einstellungen vornehmen: Zum einen muss der Modus ausgewählt werden, in dem auf den Bus zugegriffen werden soll (Passiv/Als Raumgerät). Bei Verwendung eines QAA-Raumgerätes muss hier „passiv“ ausgewählt werden. Dann werden nur die Werte, die über den Bus gehen, in der Weboberfläche angezeigt, ein Schreiben von Werten ist dann nicht möglich.  Wenn hier „als Raumgerät“ ausgewählt wird, können über die Weboberfläche auch Werte an die Heizung gesendet werden. Zum anderen ist dann noch der Typ des zu emulierenden Raumgerätes auszuwählen (s.u.). <i>Es sollte dann kein weiteres Raumgerät am Bus hängen, da sonst beide Sender ihre jeweils eigenen Werte an den Heizungsregler schicken, so dass kein konsistenter Betrieb möglich ist.</i>
PPS: QAA Modell	QAA70	Nur PPS: Modell des zu imitierenden Raumgerätes (QAA50/QAA70).
Gerätefamilie	0	0 = automatische Reglererkennung aktiv (empfohlene Einstellung). Bei einer fehlerhaften Erkennung kann die Gerätefamilie (Ausgabe von <span style="color: red;">/6225</span> ) des angeschlossenen Reglers fest eingetragen werden.
Gerätevariante	0	0 = automatische Reglererkennung aktiv (empfohlene Einstellung). Bei einer fehlerhaften Erkennung kann die Gerätevariante (Ausgabe von <span style="color: red;">/6226</span> ) des angeschlossenen Reglers fest eingetragen werden.
URL Passkey	-keine Voreinstellung-	Optionale Sicherheitsfunktion: "URL Passkey"
HTTP-Authentifizierung	-keine Voreinstellung-	Optionale Sicherheitsfunktion: "User-Pass" (Basic HTTP Auth)
DHCP verwenden	Ein	DHCP verwenden (= automatische IP-Adressvergabe durch Router) (Aus/Ein)
Statische IP-Adresse	192.168.178.88	Manuelle Netzwerkkonfiguration: Feste IP-Adresse
Subnetzmaske	255.255.255.0	Manuelle Netzwerkkonfiguration: Subnetz
Gateway	192.168.178.1	Manuelle Netzwerkkonfiguration: IP-Adresse des Gateways
DNS Server	192.168.178.1	Manuelle Netzwerkkonfiguration: IP-Adresse des DNS-Servers
TCP Port	80	TCP-Port des Setups
MAC-Adresse	00:80:41:19:69:90	(Voreingestellte) MAC-Adresse des LAN-Shields oder MAC-Adresse des ESP
Vertrauenswürdige IP-Adresse	0.0.0.0	Optionale Sicherheitsfunktion: "Trusted IP", Zugriff nur von dieser IP möglich
Vertrauenswürdige IP-Adresse	0.0.0.0	Optionale Sicherheitsfunktion: "Trusted IP", Zugriff nur von dieser IP möglich
WLAN SSID	-keine Voreinstellung-	SSID des WLAN

Funktion	(Vor-)Einstellung	Erklärung
WLAN Passwort	-keine Voreinstellung-	Passwort des WLAN
mDNS Hostname	BSB-LAN	Einstellbarer Hostname
Log-Modus	-keine Voreinstellung-	Auswählbare Optionen für die Loggingfunktion (Mehrfachauswahl möglich): Auf SD-Karte schreiben / 24-Stunden-Durchschnittswerte berechnen / An MQTT-Broker senden / Als UDP-Nachrichten senden
Logintervall (Sekunden)	3600	Logintervall in Sekunden
Parameter	8700,8743,8314	Zu loggende Parameter
Busteleggramme	Aus	Loggen von Bustelegrammen aktivieren (Aus/-diverse Optionen-), die gewünschte Einstellung ist der jeweiligen Optionsbeschreibung entspr. vorzunehmen.
Parameter	8700,8326	Parameter für die 24h-Durchschnittswertberechnung
Verwenden	Plain Text	MQTT-Übertragung in Plain Text / JSON / Rich JSON
IP-Adresse Broker	192.168.178.20	IP-Adresse des MQTT-Brokers
Username	User	MQTT: Username bei Verwendung von Username/Passwort
Passwort	Pass	MQTT: Passwort bei Verwendung von Username/Passwort
Geräte ID	BSB-LAN	Gerätename (Header in JSON-Payload)
Topic prefix	BSB-LAN	Topic prefix der MQTT-Nachrichten
Berechnung	Aus	Berechnung von 24h-Durchschnittswerten ausgewählter Parameter (Aus/Ein)
Pins	0	Verwendete(r) Pin(s) für OneWire-Sensoren (DS18B20) (0 = deaktiviert)
Pins	0	Verwendete(r) Pin(s) für DHT22-Sensoren (0 = deaktiviert)
Sensoren	0	Anzahl der angeschlossenen BME280-Sensoren (0 = deaktiviert)
TWW-Push Taste: Pin	0	Raumgerät-Emulation: Verwendeter Pin für den TWW-Push Taster.
RGT1 Temperatursensor Parameter	-keine Voreinstellung-	Raumgerät 1 Emulation: Trage hier die spezifische(n) Parameternummer(n) für den (die) Raumtemperatur-Sensor(en) ein. Bis zu fünf Sensoren können verwendet werden, die Aufzählung der Parameternummern ist lediglich durch ein Komma zu separieren. Wenn mehr als ein Sensor verwendet werden, wird automatisch der Mittelwert gebildet.
RGT1 Präsenztaste: Pin	0	Raumgerät 1 Emulation: Verwendeter Pin für die HK1-Präsenztaste.
RGT2 Temperatursensor Parameter	-keine Voreinstellung-	Raumgerät 2 Emulation: Trage hier die spezifische(n) Parameternummer(n) für den (die) Raumtemperatur-Sensor(en) ein. Bis zu fünf Sensoren können verwendet werden, die Aufzählung der Parameternummern ist lediglich durch ein Komma zu separieren. Wenn mehr als ein Sensor verwendet werden, wird automatisch der Mittelwert gebildet.
RGT2 Präsenztaste: Pin	0	Raumgerät 2 Emulation: Verwendeter Pin für die HK2-Präsenztaste.
RGT3 Temperatursensor Parameter	-keine Voreinstellung-	Raumgerät 3 Emulation: Trage hier die spezifische(n) Parameternummer(n) für den (die) Raumtemperatur-Sensor(en) ein. Bis zu fünf Sensoren können verwendet werden, die Aufzählung der Parameternummern ist lediglich durch ein Komma zu separieren. Wenn mehr als ein Sensor verwendet werden, wird automatisch der Mittelwert gebildet.



Funktion	(Vor-)Einstellung	Erklärung
RGT3 Präsenztaete: Pin	0	Raumgerät 3 Emulation: Verwendeter Pin für die HK3-Präsenztaete.
Verwenden	Aus	MAX!-Geräte verwenden (Aus/Ein)
IP-Adresse Cube	192.168.178.5	IP-Adresse des CUNO/CUNX/modifizierten MAX!Cube
Geräte	KEQ0502326,KEQ0505080	Seriennummern der zu verwendenden MAX!-Geräte
Verwenden	Aus	IPWE-Erweiterung verwenden (URL/ipwe.cgi) (Aus/Ein)
Parameter	8700,8743,8314	Darzustellende Parameter in der IPWE-Erweiterung
Verwenden	Serial	Debugging-Funktion verwenden (Aus/Serial/Telnet)
Verbositätsmodus	Ein	Verbositätsmodus aktiviert (Aus/Ein)
Monitor Modus	Aus	Monitor Modus aktiviert (Aus/Ein)
Unbekannte Parameter anzeigen	Ein	Unbekannte bzw. nicht verfügbare Parameter ("error 7 - parameter not supportet") anzeigen (Ein/Aus).

## 2.2.2 Konfiguration durch Anpassen der Datei *BSB\_LAN\_config.h*

Die Konfiguration der BSB-LAN-Software kann außerdem erfolgen, indem die Einstellungen in der Datei *BSB\_LAN\_config.h* angepasst werden. Hierzu werden nachfolgend sämtliche Einstellmöglichkeiten analog zu der Reihenfolge in der Datei *BSB\_LAN\_config.h* aufgeführt. Es ist daher ratsam, die Einstellungen Punkt für Punkt abzuarbeiten.

Hinweis
Wenn ein Definement deaktiviert ist oder werden soll, dann sind vor dem Hashtag zwei Slashes hinzuzufügen ("auskommentieren"): <pre>//#define XYZ = Definement XYZ ist deaktiviert.</pre>
Wenn ein Definement aktiviert werden soll, dann sind die beiden Slashes vor dem Hashtag zu entfernen: <pre>#define XYZ = Definement XYZ ist aktiv.</pre>

- Die **Sprache der Benutzeroberfläche** des Webinterface des Adapters sowie der Kategorie- und Parameterbezeichnungen muss *zwingend* ausgewählt bzw. definiert werden. Für "Deutsch" ist dabei das folgende Definement zu wählen:

```
#define LANG DE
```

Ab BSB-LAN v.042 ist es möglich, BSB-LAN auch in anderen Sprachen zu nutzen, wobei prinzipiell jede Sprache unterstützt werden kann (es müssen dann 'nur' die entspr. Übersetzungen erstellt werden).

Vorhanden sind momentan: Tschechisch (CZ), Deutsch (DE), Dänisch (DK), Englisch (EN), Spanisch (ES), Finnisch (FI), Französisch (FR), Griechisch (GR), Ungarisch (HU), Italienisch (IT), Niederländisch (NL), Polnisch (PL), Russisch (RU), Schwedisch (SE), Slovenisch (SI) und Türkisch (TR). Wenn gewisse Ausdrücke nicht in der spezifischen Sprache vorliegen, wird automatisch der englische Ausdruck angezeigt. Sollte auch dieser nicht vorhanden sein, wird schließlich der deutsche Ausdruck dargestellt.

- Konfigurationseinstellungen aus EEPROM oder der Datei *BSB\_LAN\_config.h* laden:**

```
byte UseEEPROM = 1;
```

Gemäß Voreinstellung werden die Konfigurationseinstellungen beim Start von BSB-LAN aus dem EEPROM gelesen. Als Fallback kann die Variable auf '0' gesetzt werden, dann werden die Einstellungen aus der Datei *BSB\_LAN\_config.h* gelesen.

### Netzwerkeinstellungen:

- MAC-Adresse des Ethernet-Shields:**

```
byte mac[] = { 0x00, 0x80, 0x41, 0x19, 0x69, 0x90 };
```

Die voreingestellte MAC-Adresse kann beibehalten werden. Eine Änderung ist i.d.R. nur nötig, wenn mehr als ein Adapter verwendet wird (es sollte in jedem Fall darauf geachtet werden, dass jede MAC-Adresse im Netzwerk nur *einmal* vorkommt!). Änderungen sollten in dem Fall möglichst nur bei dem letzten Byte erfolgen (also bspw. 0x91, wenn ein zweiter Adapter zum Einsatz kommt).

| Hinweis | |:-----| | Die hier einstellbare MAC-Adresse bezieht sich nur auf das LAN-Shield. Sie beeinflusst nicht die MAC-Adresse des ESP bei der WiFi-ESP-Lösung, dort ist die MAC-Adresse nicht einstellbar. | | Die hier vergebene MAC-Adresse beeinflusst auch den Hostnamen (bzw. ist ein Bestandteil davon), der bei der Verwendung von DHCP (s.u.) vom Router vergeben wird: Der Hostname setzt sich aus der Kennung "WIZnet" und den drei letzten Bytes der MAC-Adresse zusammen.

Für die o.g. voreingestellte MAC-Adresse lautet der Hostname somit "WIZnet196990". Dieser wird i.d.R. auch als solcher im Router angezeigt. Das Webinterface von BSB-LAN ist in dem Fall im Browser unter `http://wiznet196990` erreichbar.

Wird die MAC-Adresse bei einem zweiten Adapter nun also bspw. in

```
byte mac[] = { 0x00, 0x80, 0x41, 0x19, 0x69, 0x91 };
```

geändert, so lautet der Hostname entsprechend "WIZnet196991" bzw. `http://wiznet196991`. |

- **Ethernet-Port:**

```
uint16_t HTTPPort = 80;
```

Port 80 für HTTP voreingestellt.

- **DHCP:**

```
bool useDHCP = true;
```

Per default wird DHCP verwendet. Sollte dies jedoch nicht gewünscht sein, sondern soll selber eine feste IP vergeben werden, so ist *false* einzustellen.

| Hinweis | |:-----| | Bei der Nutzung von DHCP setzt sich der automatisch vergebene Hostname aus der Kennung "WIZnet" und den drei letzten Bytes der MAC-Adresse zusammen.

Für die o.g. voreingestellte MAC-Adresse lautet der Hostname somit "WIZnet196990". Dieser wird i.d.R. auch als solcher im Router angezeigt. Das Webinterface von BSB-LAN ist in dem Fall im Browser unter `http://wiznet196990` erreichbar.

Wird die MAC-Adresse bei einem zweiten Adapter nun also bspw. in

```
byte mac[] = { 0x00, 0x80, 0x41, 0x19, 0x69, 0x91 };
```

geändert, so lautet der Hostname entsprechend "WIZnet196991" bzw. `http://wiznet196991`.

Die IP, die bei der DHCP-Nutzung vom Router automatisch vergeben wird, wird beim Start des Due/ESP32 im Seriellen Monitor der Arduino IDE angezeigt. |

- **IP-Adresse:**

```
byte ip_addr[4] = {192,168,178,88};
```

IP-Adresse des Adapters, wenn DHCP nicht verwendet wird - *bitte beachte die Kommata anstelle von Punkten!*

| Achtung | |:-----| | Falls du die IP selbst fest vergeben willst, so vergewissere dich, dass die IP-Adresse nur einmal im Netzwerk vorkommt! |

- **Gateway-Adresse:**

```
byte gateway_addr[4] = {192,168,178,1};
```

IP-Adresse des Gateways (i.d.R. die des Routers) - *bitte beachte die Kommata anstelle von Punkten!*

- **DNS-Server:**

```
byte dns_addr[4] = {192,168,178,1};
```

IP-Adresse des DNS - *bitte beachte die Kommata anstelle von Punkten!*

- **Subnet:**

```
byte subnet_addr[4] = {255,255,255,0};
```

Subnetz-Adresse - *bitte beachte die Kommata anstelle von Punkten!*

- **WiFi:**

```
//#define WIFI
```

Dieses Definiment ist zu aktivieren, wenn die WiFi-Funktion mittels der [ESP8266-WiFi-Lösung](#) oder mittels eines [ESP32](#) genutzt werden soll.

```
char wifi_ssid[32] = "YourWiFiNetwork";
```

Bei Verwendung von WiFi, *YourWiFiNetwork* durch die SSID des WLAN-Netzwerkes ersetzen.

```
char wifi_pass[64] = "YourWiFiPassword";
```

Bei Verwendung von WiFi, *YourWiFiPassword* durch das Passwort des WLAN-Netzwerkes ersetzen.

```
#define WIFI_SPI_SS_PIN 12
```

Hier wird der beim DUE zu verwendende SS-Pin für die [ESP8266-WiFi-Lösung](#) definiert. Es ist ratsam, die Voreinstellung zu belassen. Soll dennoch ein anderer Pin genutzt werden, so ist zwingend darauf zu achten, dass der gewünschte Pin weder anderweitig genutzt wird, noch in der Liste der geschützten Pins aufgeführt ist.

| Hinweis | |:-----| | Die MAC-Adresse des ESP8266 lässt sich nicht einstellen! |

- **Nutzung von Multicast DNS:**

```
#define MDNS_SUPPORT char mDNS_hostname[32] = "BSB-LAN";
```

Per default ist die Nutzung von Multicast DNS mit dem Hostnamen "BSB-LAN" aktiviert, so dass das Adaptersetup im Netzwerk unter diesem Namen zu finden ist.

| Hinweis | |:-----| | mDNS ist nur bei einer LAN-Anbindung verfügbar, bei der [WiFi-Lösung mittels ESP8266](#) hingegen nicht! |

- **Zeit- und Datumsabfrage via NTP-Server:**

```
#define USE_NTP // Disable this in case you don't want to use NTP
const char ntp_server[20] = "pool.ntp.org";
const char local_timezone[30] = "CET-1CEST,M3.5.0,M10.5.0/3";
```

Per default ist die Zeitabfrage via NTP aktiviert.

| Hinweis | |:-----| | Die Zeitabfrage via NTP ist nur mit ESP32-Boards möglich! |

```
* NTP settings to acquire exact date and time via network.
* Attention: This only works with ESP32 microcontrollers so far!
* Use pool.ntp.org if your BSB-LAN installation can access the internet.
* Otherwise you may also use your router's address if it can act as a NTP server.
* The default timezone "CET-1CEST,M3.5.0,M10.5.0/3" covers most Central European countries (GMT+1) and takes care of daylight saving.
* Use "EET-2EEST,M3.5.0/3,M10.5.0/4" for Eastern European countries (GMT+2), or
* use "WETOWEST,M3.5.0/1,M10.5.0" for Western European countries (GMT+0).
* See here for a full list of timezones for places all over the world:
* https://github.com/nayarsystems/posix_tz_db/blob/master/zones.csv
```

- **Debugging und entspr. Einstellungen:**

- `#define DEBUG` → Debug-Modus aktivieren (s. nachfolgende Optionen)
- `byte debug_mode = 1;`

Folgende Debug-Optionen sind verfügbar:

0 - Debugging deaktiviert

1 - Debug-Nachrichten an das serielle Interface senden (einzustellen bei der Verwendung von bspw. dem Seriellen Monitor der Arduino IDE)

2 - Debug-Nachrichten an einen TelNet-Client anstelle des seriellen Interface senden

- `byte verbose = 1;`

Per default ist der Verbose Modus aktiviert (= 1), so dass neben den Rohdaten auch der jeweilige Klartext (falls vorhanden) von Parametern und Werten dargestellt wird. Es ist ratsam, diese Einstellung so zu belassen, da es eine etwaige Fehlersuche erleichtert. Darüber hinaus ist diese Einstellung nötig, falls Telegramme und CommandIDs neuer Parameter dekodiert werden sollen. zum Deaktivieren ist '0' anstelle der '1' einzutragen.

- `byte monitor = 0;`

Bus-Monitor-Modus, per default deaktiviert (= 0); zum Aktivieren auf '1' stellen.

- `bool show_unknown = true;`

Alle Parameter mitsamt der *unbekannten Parameter* (Fehlermeldung „error 7 (parameter not supportet)“) werden bei einer Abfrage via Webinterface (bspw. bei einer Abfrage einer kompletten Kategorie) angezeigt (Voreinstellung). Sollen der Übersichtlichkeit halber die vom Heizungsregler nicht unterstützten (also 'unbekannten') Parameter bei einer Abfrage ausgeblendet werden (bspw. bei der Abfrage einer kompletten Kategorie), so ist 'false' einzustellen ( `bool show_unknown = false;` ). Die Parameter werden jedoch bei einer solchen Abfrage (bspw. einer komplette Kategorie) trotzdem mit abgefragt.

## Sicherheitsfunktionen:

### • Passkey:

Um das System vor einem ungewollten Zugriff von außen zu schützen, kann die **Funktion des Sicherheitsschlüssels (PASSKEY)** genutzt werden (sehr einfach und nicht wirklich sicher!):

```
char PASSKEY[64] = "";
```

Für die Verwendung ist eine Zahlenfolge einzugeben, bspw. `char PASSKEY[64] = "1234";` → in diesem Beispiel lautet der Passkey 1234. Wird keine Zahlenfolge eingegeben (also die Voreinstellung nicht geändert), so ist die Funktion deaktiviert.

| Achtung | |-----| | Falls die PASSKEY-Funktion genutzt wird, muss die URL bei einem Aufruf des Webinterfaces den definierten Schlüssel als erstes Element enthalten, bspw. `http://<IP-Adresse>/<passkey>/` um die Startseite zu sehen.

Nur bei der URL der optionalen **IPWE-Erweiterung** darf der Passkey *nicht* zusätzlich eingegeben werden! | | Bitte nicht den Slash hinter dem Passkey vergessen! |

### • Trusted IP:

```
byte trusted_ip_addr[4] = {0,0,0,0};
```

```
byte trusted_ip_addr2[4] = {0,0,0,0};
```

Bei den Variablen `trusted_ip_addr` (und `trusted_ip_addr2` für eine weitere IP) kann man eine vertrauenswürdige IP eintragen (z.B. des FHEM-Servers), dann ist der Zugriff nur über diese IP. Lautet die vertrauenswürdige IP des Clients bspw. `192.168.178.20`, so ist `byte trusted_ip_addr[4] = {192,168,178,20};` einzustellen.

Wird die Voreinstellung `{0,0,0,0}` nicht geändert und/oder die erste Zahl ist eine 0, ist diese Funktion deaktiviert.

### • User-Pass:

```
char USER_PASS[64] = "";
```

Mit `USER_PASS[64]` kann eine Zugangssperre nach dem Muster *Username:Passwort* gesetzt werden:

```
//char USER_PASS[64] = "User:Password";
```

Ist kein String eingegeben (Voreinstellung), so ist die Funktion deaktiviert.

## Einstellungen für optionale Sensoren:

### • OneWire-Temperatursensoren (DS18B20):

```
#define ONE_WIRE_BUS
```

```
byte One_Wire_Pin = 0;
```

Sollen OneWire-Temperatursensoren (DS18B20) verwendet werden, muss das Definement aktiviert sein sowie die entsprechende GPIO-Pinbelegung definiert werden.

Voreingestellt ist das Modul aktiviert und Pin 0 eingestellt (0 = OneWire-Verwendung deaktiviert).

- **DHT22-Sensoren:**

```
#define DHT_BUS
uint8_t DHT_Pins[10] = {0};
```

Sollen DHT22-Sensoren (Temperatur & Feuchtigkeit; max. Anzahl: 10) verwendet werden, muss das entsprechende Definement aktiviert sein und die entsprechende GPIO-Pinbelegung definiert werden.

Voreingestellt ist das Modul aktiviert und Pin 0 eingestellt (0 = DHT-Verwendung deaktiviert).

- **BME280 Sensoren:**

```
#define BME280 byte BME_Sensors = 0;
```

Wenn BME280 Sensoren zur Anwendung kommen sollen, so muss das Definement aktiviert und die Anzahl der angeschlossenen Sensoren angegeben werden (Voreinstellung 0 = deaktiviert, maximal 2). Die Sensoren müssen am I2C-Bus angeschlossen werden. Die Adresse des ersten Sensors muss 0x76 lauten, die des zweiten Sensors 0x77.

- **24h-Durchschnittswerte:**

```
#define AVERAGES
```

Sollen 24h-Durchschnittswerte von bestimmten Parametern berechnet werden, so ist das Definement zu aktivieren (Voreinstellung: aktiviert).

Des Weiteren müssen die gewünschten Parameter (max. 40) bei der entsprechenden Variable eingetragen werden, bspw.:

```
parameter avg_parameters[40] = {
    {8700, -1},           // Außentemperatur
    {8326, -1}           // Brenner-Modulation
};
```

Beim Vorhandensein einer SD-Karte werden die jeweils aktuellen Werte dort regelmäßig gesichert, um nach einem Neustart die Berechnung lückenlos fortsetzen zu können.

Sollen die Durchschnittswerte der oben eingestellten Parameter *zusätzlich* bspw. in ein Logfile geschrieben und via URL-Befehl `/DG` angezeigt oder per MQTT verschickt werden, so sind sie als *Spezialparameter* mit den Nummern 20050-20099 bei den *zu loggenden Parametern* (s.u.) aufzuführen! Für diese gelten dann wiederum die entsprechenden Loggingeinstellungen (s.u.), wie bspw. das Logintervall.

- **Logging (auch auf microSD-Karte) und/oder Verwendung von MQTT/UDP:**

```
#define LOGGER → Das Logging-Modul wird kompiliert.
```

| Achtung | |-----| | Das genannte *aktivierte* Definement ist sowohl Voraussetzung für das Loggen auf eine microSD-Karte als auch für die Verwendung von MQTT und UDP (s.u.)! |

Nachfolgend können/sollten verschiedene Einstellungen vorgenommen werden:

- Logdaten via UDP broadcast senden:

```
#define UDP_LOG_PORT 6502 → Logdaten werden zusätzlich per UDP broadcast an den Port 6502 (default) gesendet. Der gewünschte Port kann hier eingestellt werden.
```

- Wenn ein microSD-Kartenadapter an einem ESP32-basierten Board verwendet wird und das Loggen auf Karte (empfohlen!) anstatt des SPIFF-Flashspeichers erfolgen soll, so ist das folgende Definement zu aktivieren:

```
//#define ESP32_USE_SD
```

- Sollen 'rohe' *Bus-Datentelegramme* geloggt werden, kann die Auswahl spezifiziert werden. Die Speicherung der Telegramme erfolgt in der Datei *journal.txt* auf der microSD-Karte. In der Voreinstellung ist das Loggen von Bustelegrammen deaktiviert:

```
int logTelegram = LOGTELEGRAM_OFF;
```

Folgende Einstelloptionen sind hier verfügbar:



`LOGTELEGRAM_OFF` → Bus-Telegramme werden nicht geloggt (Voreinstellung)

`LOGTELEGRAM_ON` → alle Bus-Telegramme werden geloggt

`LOGTELEGRAM_ON + LOGTELEGRAM_UNKNOWN_ONLY` → nur unbekannte Bus-Telegramme werden geloggt

`LOGTELEGRAM_ON + LOGTELEGRAM_BROADCAST_ONLY` → nur Broadcast-Telegramme werden geloggt

`LOGTELEGRAM_ON + LOGTELEGRAM_UNKNOWNBROADCAST_ONLY` → nur unbekannte Broadcast-Telegramme werden geloggt

- `bool logCurrentValue = false;`

Die Daten der zu loggenden Parameter werden bei Bedarf in der Datei 'datalog.txt' auf der microSD-Karte gespeichert. Dazu ist die Variable auf `true` zu setzen.

- `unsigned long log_interval = 3600;`

Das gewünschte Logintervall in *Sekunden*.

| Achtung | |-----| | Dieses Intervall ist auch für die Nutzung von MQTT (s.u.) einzustellen, selbst wenn kein Loggen stattfinden soll! |

- Die zu loggenden Parameter (max. 40) müssen dann zusammen mit der Ziel-Bus-Adresse (s.u.; -1 ist die default Zieladresse) bei der entsprechenden Variable eingetragen werden, bspw.:

```
parameter log_parameters[40] = {
  {8700, -1}, // Außentemperatur
  {8743, -1}, // Vorlauftemperatur
  {8314, -1}, // Rücklauftemperatur
};
```

Wenn bspw. die Messwerte mehrerer DS18B20- oder DHT22-Sensoren geloggt werden sollen, müssen die spezifischen Spezialparameternummern bei den Log-Parametern ebenfalls entsprechend aufgeführt werden, bspw.:

```
{20301, -1}, // Spezialparameter 20300-20499: DS18B20-Sensoren 1-100
{20303, -1},
{20305, -1},
```

loggt die Messwerte der DS18B20-Sensoren 1-3.

Zum Loggen der Brennerstarts und -laufzeiten müssen die Spezialparameter 20000 und 20001 aufgeführt werden (siehe auch die Beschreibung in der Datei `BSB_LAN_config.h`). Bei einem zweistufigen Ölbrenner, dessen Regler die entsprechenden Broadcasts schickt und bei dem eine Differenzierung der Brennerstufen möglich ist (derzeit nur RVS43.325), müssen hier zusätzlich 20002 und 20003 mit aufgeführt werden.

Weitere gängige Spezialparameter lauten:

- TWW-Laufzeit und TWW-Takte: 20004 und 20005,
- 24h-Durchschnittswerte: 20050-20099,
- DHT22-Sensoren: 20100-20299,
- BME280 Sensoren: 20200-20299,
- DS18B20-Sensoren: 20300-20499,
- MAX!-Sensoren: 20500-20699.

Für eine genauere Aufschlüsselung der Nummernbereiche einzelner optionaler Sensoren sieh bitte im entspr. Kapitel nach.

## • MQTT:

Soll MQTT zum Einsatz kommen, so sind *neben den obigen Logging-Parametern* die entspr. Variablen und Einstellungen anzupassen:

- `#define MQTT` → Das MQTT-Modul wird kompiliert (Voreinstellung)
- `byte mqtt_mode = 0;` → MQTT ist deaktiviert (Voreinstellung); folgende Optionen sind verfügbar:

1 = die Nachrichten werden im einfachen Textformat gesendet

2 = die Nachrichten werden im JSON-Format gesendet

```
Struktur der JSON-Payload:  
`{"MQTTDeviceID": {"status":{"log_param1":"value1","log_param2":"value2"}, ...}}`
```

3 = die Nachrichten werden im rich JSON-Format gesendet

```
Struktur der rich JSON-Payload:  
`{"MQTTDeviceID": {"id": one_of_logvalues, "name": "program_name_from_logvalues", "value": "query_result", "desc": "enum value description", "unit": "unit of measurement", "error": error_code}}`
```

- `byte mqtt_broker_ip_addr[4] = {192,168,1,20};` → IP des MQTT-Brokers.  
*Bitte beachte die Kommata anstelle von Punkten!*  
Der Standardport ist 1883 und muss nicht extra definiert werden.
- `char MQTTUsername[32] = "User";` → Username; wird Username/Passwort beim MQTT-Broker nicht verwendet, ist das *User* zu entfernen.
- `char MQTTPassword[32] = "Pass";` → Passwort; wird Username/Passwort beim MQTT-Broker nicht verwendet, ist das *Pass* zu entfernen.
- `char MQTTTopicPrefix[32] = "BSB-LAN";` → Optional: Die MQTT-Nachrichten haben das Topic-Format ('Thema') `BSB-LAN/<Parametername>` und den entsprechenden Wert dann in der Payload. Wenn nichts angegeben wird ( `char MQTTTopicPrefix[32] = "";` ), wird der Standard-Themenname verwendet.
- `char MQTTDeviceID[32] = "MyHeater";` → Optional: Device-Name, der als Header in der JSON-Payload genutzt wird. Wenn nichts angegeben wird ( `char MQTTDeviceID[32] = "";` ), wird "BSB-LAN" verwendet.

| Hinweis | |:-----| | Die zu übertragenden Parameter sowie das Übertragungsintervall für MQTT werden oben bei den zu loggenden Parametern und dem Logintervall für das Loggen auf microSD-Karte eingegeben! |

#### • IPWE:

`#define IPWE` → Das IPWE-Modul wird kompiliert.

`bool enable_ipwe = false;`

Soll die IPWE-Erweiterung (URL/ipwe.cgi) verwendet werden, ist die Variable auf 'true' zu setzen.

Die gewünschten Parameter (max. 40) sind ebenfalls einzutragen:

```
parameter ipwe_parameters[40] = {  
    {8700, -1}, // Außentemperatur  
    {8743, -1}, // Vorlauftemperatur  
    {8314, -1}, // Rücklauftemperatur  
};
```

#### • MAX! (CUNO/CUNX/modifizierter MAX!Cube):

Sollen optionale MAX!-Thermostate zum Einsatz kommen, müssen folgende Einstellungen angepasst werden:

- `//#define MAX_CUL` → Definement aktivieren (= `#define MAX_CUL`) (deaktiviert by default)
- `bool enable_max_cul = false;` → Variable auf 'true' setzen
- `byte max_cul_ip_addr[4] = {192,168,178,5};` → IP-Adresse des CUNO/CUNX/modifizierten MAX!Cubes - *bitte beachte die Kommata anstelle von Punkten!*
- Liste der abzufragenden MAX!-Thermostate (max. 20):

```
char max_device_list[20][11] = {
  "KEQ0502326",
  "KEQ0505080"
};
```

Hier bitte die entspr. 10-stellige Seriennummer / MAX!-ID eintragen.

Für weitere Informationen bzgl. der Einbindung von MAX!-Komponenten s. [Kap. 7.3](#).

- **Anzahl der maximalen Wiederholungsversuche bei einer Abfrage:**

```
#define QUERY_RETRIES 3
```

Hier kann bei Bedarf eingestellt werden, wieviele maximale Wiederholungsversuche ausgeführt werden, wenn bei einer Abfrage keine entsprechende Antwort vom Heizungsregler kommt. In der Regel kann die Voreinstellung (max. 3 Versuche) beibehalten werden.

### **Buseinstellungen (Pins und Typ):**

- **RX-/TX-Pin-Konfiguration:**

`byte bus_pins[2] = {0,0};` → automatische Erkennung und Einstellung der RX-/TX-Pinbelegung (Voreinstellung); ansonsten gilt:

- Hardware-Serial (ab Adapter v3) Arduino Due: RX-Pin = 19, TX-Pin = 18 ( `{19,18}` ); NodeMCU: 16,17; Olimex EVB 36,17.
- Software-Serial (bis einschließlich Adapter v2 & Arduino Mega 2560): RX-Pin = 68, TX-Pin = 69 ( `{68,69}` )

- **Bus-Typ/-Protokoll:**

```
uint8_t bus_type = 0;
```

Je nach Anschluss des Adapters an einen BSB/LPB/PPS-Anschluss muss der entspr. Bus-Typ definiert werden (bereits nach Booten des Arduino wirksam).

Voreingestellt ist 0 für BSB, für LPB ist 1 einzustellen, für PPS hingegen 2:

- 0 = BSB
- 1 = LPB
- 2 = PPS

- **Buseinstellungen:**

Abhängig vom Bus-Typ müssen unterschiedliche Einstellungen vorgenommen werden.

→ **BSB:**

- `byte own_address = 0x42;` → entspricht der eigenen Geräteadresse 66 des BSB-LAN-Adapters
- `byte dest_address = 0x00;` → entspricht der Zieladresse 0 (i.d.R. der Regler des Wärmeerzeugers)

→ **LPB:**

- `byte own_address = 0x42;` → eigene Adresse (BSB-LAN-Adapter), entspricht der Segmentadresse 4 mit Geräteadresse 3
- `byte dest_address = 0x00;` → Zieladresse (Heizungsregler), entspricht der Segmentadresse 0 mit Geräteadresse 1

→ **PPS:**

- `bool pps_write = 0;` → in der Standardeinstellung ist nur ein lesender Zugriff auf den via PPS angeschlossenen Heizungsregler möglich.

Soll Schreibzugriff ermöglicht werden, so ist eine `1` einzutragen ( `bool pps_write = 1;` ).

| Achtung | |-----| | Schreibzugriff NUR einstellen, wenn KEIN originales QAA50/QAA70-Raumgerät vorhanden ist! |

- `byte QAA_TYPE = 0x53;` → Typ des zu imitierenden Raumgerätes:  
0x53 = QAA70 (Voreinstellung) 0x52 = QAA50

0x37 = QAA95  
0x66 = BMU  
0xEA = MCBA/DC225

- **Erkennung bzw. Festlegung des Heizungsreglertyps:**

```
static const int fixed_device_family = 0;  
static const int fixed_device_variant = 0;
```

Wenn die Werte auf 0 gesetzt sind, ist die automatische Erkennung des angeschlossenen Reglers beim Starten des Arduino aktiviert (Voreinstellung). Dies kann i.d.R. so belassen werden.

Alternativ kann hier die Ausgabe von `http://<IP-Adresse>/6225/6226` eingetragen werden (6225 = Gerätefamilie / device family & 6226 = Gerätevariante / device variant).

Ein fest eingestellter Wert (laut Ausgabe von 6225&6226) stellt sicher, dass BSB-LAN auch dann noch korrekt arbeitet, wenn die Heizung bzw. der Regler erst nach dem Starten des Arduino/ESP eingeschaltet wird (da in dem Fall die automatische Erkennung des angeschlossenen Reglers nicht funktionieren kann, da ja keine Rückmeldung vom Regler kommt).

- **Schreib-/Lesezugriff auf den Heizungsregler:**

```
#define DEFAULT_FLAG FL_SW_CTL_RDONLY
```

In der Voreinstellung ist der Zugriff des Adapters auf den Heizungsregler auf Lesen beschränkt, d.h. ein Setzen bzw. Verändern von Parametern der Heizungssteuerung per Adapter ist in der Voreinstellung nicht möglich. Eine Änderung des Status auf *generellen* Schreibzugriff kann via Webinterface (Menüpunkt "Einstellungen") erfolgen.

| Hinweis für Mega-Nutzer | |-----| | Die Möglichkeit der Konfiguration via Webinterface bietet sich für Nutzer des Mega 2560 nicht, da das Modul WEBCONFIG mangels Speicher nicht kompiliert und nicht genutzt werden kann.

In diesem Fall muss der Schreibzugriff nach wie vor durch das Flag '0' gewährt werden:

```
#define DEFAULT_FLAG 0 |
```

Ist der Schreibzugriff aus Sicherheitsgründen hingegen nur bei *ausgewählten* Parametern (z.B. 10000 oder 710) gewünscht, muss bei dem genannten Definement nach wie vor das genannte Flag auf `FL_SW_CTL_RDONLY` (*Hinweis für Mega-Nutzer mit deaktiviertem WEBCONFIG-Modul: Hier bitte `FL_RDONLY` setzen!*) gesetzt sein und dann in der Datei `BSB_LAN_defs.h` das `DEFAULT_FLAG` des gewünschten Parameters durch 0 (Null) ersetzt werden.

*Beachte hierbei jedoch bitte, dass es im Falle eines Updates von BSB-LAN nötig sein kann/wird, diese Änderungen erneut vorzunehmen!*

Im folgenden Beispiel wird Parameter 700 auf diese Weise schreibbar gemacht:

```
{0x2D3D0574, CAT_HK1, VT_ENUM, 700, STR700, sizeof(ENUM700), ENUM700, DEFAULT_FLAG, DEV_ALL}, // [-] - Heizkreis 1 - Betriebsart  
***(virtuelle Zeile)***
```

→ aufgrund des „DEFAULT\_FLAG“ ist der Parameter momentan nur lesbar

```
{0x2D3D0574, CAT_HK1, VT_ENUM, 700, STR700, sizeof(ENUM700), ENUM700, 0, DEV_ALL}, // [-] - Heizkreis 1 - Betriebsart ***(virtue  
lle Zeile)***
```

→ das „DEFAULT\_FLAG“ wurde durch „0“ (Null, ohne Anführungszeichen) ersetzt

- **Eigenen Code** aus der Datei `BSB_LAN_custom.h` einfügen:

```
//#define CUSTOM_COMMANDS
```

Fügt die Befehle aus der Datei `BSB_LAN_custom.h` hinzu, die am Ende jedes 'main loops' ausgeführt werden (per default deaktiviert).

- **Überprüfen der BSB-LAN-Version:**

```
#define VERSION_CHECK  
bool enable_version_check = false;
```



Diese Funktion überprüft bei jedem Aufruf der Startseite des Webinterface, ob eine neuere Version von BSB-LAN verfügbar ist; Internetzugriff nötig (deaktiviert by default). Zum Aktivieren ist die Variable auf 'true' zu setzen.

| Hinweis | |:-----| | Hierbei ist es unvermeidlich, dass die IP-Adresse an den Server übertragen wird. Wir erwähnen dies hier dennoch, da es sich hierbei um "persönliche Daten" handelt und diese Funktion daher standardmäßig deaktiviert ist. Mit der Aktivierung dieser Funktion erklärst Du Dich damit einverstanden, dass Deine IP-Adresse an den BSB-LAN-Server übermittelt wird, wo sie bis zu zwei Wochen in den Log-Dateien des Servers gespeichert wird, um sowohl technische als auch Missbrauchsanalysen zu ermöglichen. Wie Du dem Quellcode entnehmen kannst, werden bei diesem Vorgang keine weiteren Daten (z.B. alles, was mit Deiner Heizungsanlage zu tun hat) übertragen. |

- **OTA-Updatefunktion (nur ESP32):**

```
#define ENABLE_ESP32_OTA
boolean enable_ota_update = false;
```

OTA-Updatefunktion (OTA = OverTheAir) für ESP32-basierte Boards (Voreinstellung: deaktiviert). Zum Aktivieren der Funktion muss `boolean enable_ota_update = true;` eingestellt werden. Eine fertige Firmware-Datei kann man in der Arduino IDE unter "Sketch / Kompilierte Binärdatei exportieren..." erstellen lassen. Der Upload der Datei erfolgt dann auf Port 8080 der BSB-LAN-IP.

- **"Externer" Webserver:**

```
//#define WEBSERVER
```

Wenn dieses Definement aktiviert ist, kann BSB-LAN als Webserver für statische Inhalte fungieren. Für weitere Informationen siehe bitte [Kapitel 6.9](#).

- **Speichern der Konfiguration im EEPROM (nur Arduino Due und ESP32):**

```
#define CONFIG_IN_EEPROM
```

Soll die Konfiguration nicht im EEPROM des Adapters (Due-Version) bzw. im Flashspeicher des ESP32 gespeichert werden, so ist das Definement zu deaktivieren.

- **Konfiguration via Webinterface:**

```
#define WEBCONFIG
```

Ermöglicht die Konfiguration via Webinterface (bei Speicherung der Einstellungen im EEPROM des Adapters (Due-Version) bzw. im Flashspeicher des ESP32). Falls nicht gewünscht, dann ist dieses Definement zu deaktivieren.

- **Compile JSON-based configuration and EEPROM config store module extension.**

```
#define JSONCONFIG
```

- **Variablen für eine zukünftige Verwendung, derzeit noch ohne Funktion:**

```
#define ROOM_UNIT → Raumgeräteersatz
byte udpIP[4] = {0,0,0,0}; → Ziel-IP-Adresse für UDP
uint16_t udpDelay = 15; → Sendeintervall in Sekunden für UDP

#define OFF_SITE_LOGGER → Off-Site-Logger Erweiterung
byte destinationServer[128] = ""; → IP des Off-Site-Loggers
uint16_t destinationPort = 80; → Port des Off-Site-Loggers
uint32_t destinationDelay = 84600; → Sendeintervall in Sekunden
```

## 2.3 Manuelles Hinzufügen von Parametern aus v2.2

Im Vergleich zu früheren Versionen von BSB-LAN mag auffallen, dass einige Parameter nun in der gerätespezifischen Parameterliste nicht mehr auftauchen, auch wenn diese grundsätzlich funktioniert haben.

Es ist aber weiterhin möglich, ausgewählte Parameter aus der Parameterliste der Version 2.2 in die aktuelle Version aufzunehmen.

Hinweis
Wir empfehlen ausdrücklich, diese nicht offiziell vom Hersteller des Reglers unterstützten Parameter nur nach eingehender Prüfung hinzuzufügen, insbesondere, wenn diese Werte auch geschrieben werden sollen!
Als sicheren Nummernbereich, in dem Parameter wie im Folgenden beschrieben hinzugefügt werden können, empfehlen wir 10600 und aufwärts.

Im ersten Schritt lädt man sich dazu die [Release-Version 2.2](https://github.com/fredlcore/BSB-LAN/releases) unter <https://github.com/fredlcore/BSB-LAN/releases> herunter.

Nachdem man die Datei entpackt hat, findet man in dem Unterverzeichnis *BSB\_LAN* die Datei *BSB\_LAN\_custom\_defs.h.default*. Diese öffnet man mit einem Texteditor wie z.B. Notepad unter Windows oder TextEdit unter MacOS.

Parallel dazu öffnet man außerdem die Datei *BSB\_LAN\_custom\_defs.h* aus der aktuellen BSB-LAN Version, die man benutzen möchte, in der Arduino IDE.

Wenn beide Dateien geöffnet sind, sucht man in der *BSB\_LAN\_custom\_defs.h.default* der Version 2.2 nach der Parameternummer des Parameters, den man hinzufügen möchte.

**Die weiteren Schritte werden anhand des Beispiels des früheren Parameters 701 – „Präsenztaste (temporäre Abwesenheit)“ erläutert.**

Dieser Parameter ist bei der gerätespezifisch erstellten Datei *BSB\_LAN\_custom\_defs.h* mittlerweile per default enthalten, lediglich bei den ersten erstellten Dateien (in der damaligen Umstellphase auf die BSB-LAN-Version 3.x) fehlt dieser noch.

Die Suche nach „701“ ergibt zuerst diesen Eintrag:

```
const char STR701[] PROGMEM = STR701_TEXT;
```

Diese Zeile kopiert man in die Zwischenablage.

In der Datei *BSB\_LAN\_custom\_defs.h* der aktuellen BSB-LAN Version sucht man nun nach dem Text `const char S` und findet dann eine Reihe solcher Einträge. Dort fügt man die oben ausgewählte Zeile an eine beliebige Stelle ein.

Findet sich für einen Parameter kein Eintrag, der mit `const char` beginnt, sondern ein Eintrag, der dem Muster `#define STR<gesuchte Parameternummer> STR<referenzierte Parameternummer>` entspricht, sind zwei Schritte nötig:

Man kopiert zuerst die Zeile `#define STR...` in die aktuelle *BSB\_LAN\_custom\_defs.h* Datei und sucht dann in der *BSB\_LAN\_custom\_defs.h.default* nach der referenzierten Parameternummer, bis man für diese Nummer die Zeile findet, die mit `const char S` beginnt.

Am Beispiel der Parameternummer 702 würde man daher diese Zeile zuerst finden: `#define STR702 STR701`.

Daraufhin sucht man dann erneut nach der referenzierten Parameternummer (in diesem Falle 701), so dass man bei einer erneuten Suche nach dieser Parameternummer dann die Zeile `const char STR701[] PROGMEM = STR701_TEXT;` finden würde, die man dann ebenfalls kopieren würde.

*Wichtig ist, dass die `#define`-Zeile unter der `const char S...`-Zeile stehen muss!*

Das Ergebnis sähe also für Parameter 702 so aus:

```
const char STR701[] PROGMEM = STR701_TEXT;
#define STR702 STR701
```

Alle weiteren Einträge, wo die Parameternummer ggf. in der Position des referenzierten Parameters in `#define`-Zeilen steht (wie z.B. `#define STR1301 STR701`) können ignoriert werden – es sei denn, man will in diesem Fall die Parameternummer 1301 ebenfalls hinzufügen.

Da es sich bei dem Parameter 701 um einen Parameter mit Auswahloptionen handelt, finden sich auch noch Zeilen, die mit `#define ENUM701_...` beginnen. Diese Zeilen sind ebenfalls in die aktuelle *BSB\_LAN\_custom\_defs.h* zu kopieren.

In diesem Zusammenhang taucht dann noch ein Eintrag auf, der mit `const char ENUM701[]` beginnt. Diese und die nachfolgenden Zeilen sind bis zur schließenden geschweiften Klammer ebenfalls in die aktuelle *BSB\_LAN\_custom\_defs.h* zu kopieren:

```
const char ENUM701[] PROGMEM_LATEST = {
  "\x00 " ENUM701_00_TEXT "\0"
  "\x01 " ENUM701_01_TEXT "\0"
  "\x02 " ENUM701_02_TEXT
};
```

Bei rein numerischen Parametern, die kein Auswahlmenü haben, sondern z.B. nur einen Temperaturwert anzeigen, entfällt dieser Schritt, weil es dazu dann keinen `const char ENUM...`-Eintrag gibt.

Schlussendlich findet man den eigentlichen Tabelleneintrag für Parameter 701, der folgendermaßen aussieht:

```
{0x2D3D0572, VT_ENUM, 701, STR701, sizeof(ENUM701), ENUM701, DEFAULT_FLAG+FL_WONLY, DEV_ALL},
```

Die entsprechende Tabelle findet sich in der aktuellen *BSB\_LAN\_custom\_defs.h* Datei am Ende der Datei. In der dritten Spalte sieht man dabei immer die Parameternummer. Nun geht man in dieser Datei soweit nach oben, dass der Parameter an der richtigen Stelle eingefügt wird. In unserem Beispiel wäre das nach der Zeile für Parameter 700.

#### Achtung

Es ist unbedingt wichtig, darauf zu achten, dass der Parameter in dieser Tabelle / cmdtbl-Struktur an der richtigen Stelle eingefügt wird (und nicht z.B. vor der Zeile für Parameter 700 oder irgendwo danach), weil sonst die Parameter in der Kategorienübersicht nicht mehr vollständig aufgelistet werden!

Bei einigen Reglern wird jedoch der Parameter 701 schon von einer anderen Funktion belegt sein. Neuere LMS-Regler haben dort z.B. die Funktion für „temporär wärmer/kälter“ abgelegt. Das Verlegen des neu hinzuzufügenden Parameters ist jedoch einfach: Man wählt eine freie Parameternummer (abgesehen von der hier erwähnten Funktion "Präsenztaste" empfehlen wir dafür die Parameternummern 10600 und aufwärts) und fügt die Zeile

```
{0x2D3D0572, VT_ENUM, 701, STR701, sizeof(ENUM701), ENUM701, DEFAULT_FLAG+FL_WONLY, DEV_ALL},
```

an der entsprechenden Stelle in der Datei ein. Dann braucht man lediglich die Parameternummer in der dritten Spalte zu ändern, in diesem Fall auf 10110. Die Parameternummern, die bei `STR...` oder `ENUM...` eingetragen sind, können jedoch so bleiben, da sie so gewählt wurden, dass sie nicht mit den neuen Parametern kollidieren.

Die neue, finale Zeile sähe dann so aus:

```
{0x2D3D0572, VT_ENUM, 10110, STR701, sizeof(ENUM701), ENUM701, DEFAULT_FLAG+FL_WONLY, DEV_ALL},
```

*Zusammengefasst noch einmal die Zeilen, die für die Funktion „Präsenztaste“ kopiert werden müssten, um sie in die Datei *BSB\_LAN\_custom\_defs.h* als Parameternummer 10110 einzufügen:*

```
const char STR701[] PROGMEM = STR701_TEXT;
const char ENUM701[] PROGMEM_LATEST = {
"\x00 " ENUM701_00_TEXT "\0"
"\x01 " ENUM701_01_TEXT "\0"
"\x02 " ENUM701_02_TEXT
};
```

Außerdem an der entspr. korrekten Stelle in der cmdtbl-Struktur:

```
{0x2D3D0572, VT_ENUM, 10110, STR701, sizeof(ENUM701), ENUM701, DEFAULT_FLAG+FL_WONLY, DEV_ALL},
```

Danach kann BSB-LAN erneut auf den Microcontroller geflasht werden und der neue Befehl ist einsatzbereit.

Möchte man in dem Zuge gleich die in diesem Fall evtl. uneindeutige Parameterbezeichnung „Präsenztaste“ auf z.B. die zutreffendere Bezeichnung „Zeitprogramm (temporär)“ ändern, kann man dies in dem Schritt auch gleich machen. Dazu würde man einfach nur die Zeile

```
const char STR701[] PROGMEM = STR701_TEXT;
```

in

```
const char STR701[] PROGMEM = "Zeitprogramm (temporär)";
```

ändern müssen und dann erneut flashen. Da alle diese Änderungen in der *BSB\_LAN\_custom\_defs.h* erfolgen, bleiben Sie auch bei einem Update der BSB-LAN-Software erhalten.

Möchte man bei der Gelegenheit auch gleich noch die Präsenztastenfunktion für HK2 (Parameter 1001 in v2.2) als Parameter 10111 hinzufügen, würden die entspr. Zeilen so aussehen:

```
#define STR1001 STR701
```

sowie diese Zeile an der entspr. korrekten Stelle in der cmdtbl-Struktur:

```
{0x2E3E0572, VT_ENUM, 10111, STR1001, sizeof(ENUM701), ENUM701, DEFAULT_FLAG+FL_WONLY, DEV_ALL}, // [-] - Heizkreis 2 - Präsenztaste
***(virtuelle Zeile)***
```

**Parameter, die von Interesse sein könnten und die dafür zu kopierenden Zeilen**

## Parameter, die von Interesse sein könnten und die dafür zu kopierenden Zeilen

### 1602 – Status Trinkwasserbereitung

```
const char STR1602[] PROGMEM = STR1602_TEXT;

const char ENUM1602[] PROGMEM_LATEST = {
"\x00\x02 " ENUM1602_00_02_TEXT "\0"
"\x02\x02 " ENUM1602_02_02_TEXT "\0"
"\x00\x04 " ENUM1602_00_04_TEXT "\0"
"\x04\x04 " ENUM1602_04_04_TEXT "\0"
"\x00\x08 " ENUM1602_00_08_TEXT "\0"
"\x08\x08 " ENUM1602_08_08_TEXT <br> };
```

Außerdem an der entspr. Stelle in der cmdtbl-Struktur:

```
{0x31000212, VT_BIT, 1602, STR1602, sizeof(ENUM1602), ENUM1602, DEFAULT_FLAG, DEV_ALL}, // Status Trinkwasserbereitung
```

### 10100 – Status Brenner

```
#define ENUM10100_01_TEXT ENUM_CAT_34_TEXT

const char ENUM10100[] PROGMEM_LATEST = {
"\x00" // index for payload byte
"\x01\x01 " ENUM10100_01_TEXT "\0"
"\x02\x02 " ENUM10100_02_TEXT "\0"
"\x04\x04 " ENUM10100_04_TEXT "\0"
"\x08\x08 " ENUM10100_08_TEXT "\0"
"\x10\x10 " ENUM10100_10_TEXT "\0"
"\x20\x20 " ENUM10100_20_TEXT "\0"
"\x40\x40 " ENUM10100_40_TEXT "\0"
"\x80\x80 " ENUM10100_80_TEXT
};
```

Außerdem an der entspr. Stelle in der cmdtbl-Struktur:

```
{0x053D00213, VT_CUSTOM_BIT, 10100, STR10100, sizeof(ENUM10100), ENUM10100, FL_RDONLY, DEV_ALL}, // INFO Brenner
```

### 10102 – Info HK1

```
{0x2D000211, VT_UNKNOWN, 10102, STR10102, 0, NULL, DEFAULT_FLAG, DEV_ALL}, // INFO HK1
```

### 10103 – Info HK2

```
{0x2E000211, VT_UNKNOWN, 10103, STR10103, 0, NULL, DEFAULT_FLAG, DEV_ALL}, // INFO HK2
```

### 10104 – Info HK3/P

```
{0x2F000211, VT_UNKNOWN, 10104, STR10104, 0, NULL, DEFAULT_FLAG, DEV_ALL}, // INFO HK3/P
```



[Weiter zu Kapitel 3](#)

[Zurück zum Inhaltsverzeichnis](#)



# 3. BSB-LAN Setup: Anschluss und Inbetriebnahme

[Zurück zum Inhaltsverzeichnis](#)  
[Zurück zu Kapitel 2](#)

## 3. BSB-LAN Setup: Anschluss und Inbetriebnahme

### 3.1 Anschluss des Adapters

**Warnung:** *Elektrostatische Aufladungen können irreparable Schäden verursachen - erde dich vor Beginn der Arbeiten!*

Prinzipiell erfolgt der Anschluss des Adapters analog zu dem Anschluss optionaler Raumgeräte.  
*Die jeweiligen Kontakte sind den herstelllerspezifischen Unterlagen zum Heizungssystem zu entnehmen.*

Ist nur ein BSB-Anschluss verfügbar (bspw. bei Wärmepumpen mit einem RVS21-Regler) und/oder bereits ein Raumgerät vorhanden, so kann der Adapter parallel zu einem bereits installierten Raumgerät an die gleichen Anschlüsse angeschlossen werden.

Hinweise
Da es sich bei BSB um ein Bussystem handelt, kann der Adapter auch bei einem bereits im Wohnraum installierten kabelgebundenen Raumgerät angeschlossen werden! Sollte kein Raumgerät vorhanden sein, so sollte man überprüfen, ob es nicht einfacher ist, ein langes dünnes zweiadriges Buskabel in die Wohnung zu verlegen als ein langes LAN-Kabel. Es ist also nicht zwingend nötig, den Adapter unmittelbar am Aufstellort der Heizung anzuschließen!
Beim Anschließen des Adapters sollte der betreffende Regler sicherheitshalber stets ausgeschaltet sein, ebenso bei einem Entfernen des Adapters.
Es ist unbedingt darauf zu achten, dass der Regler polrichtig angeschlossen wird! Ein verkehrter Anschluss kann eine Beschädigung des Reglers und/oder Adapters zur Folge haben!

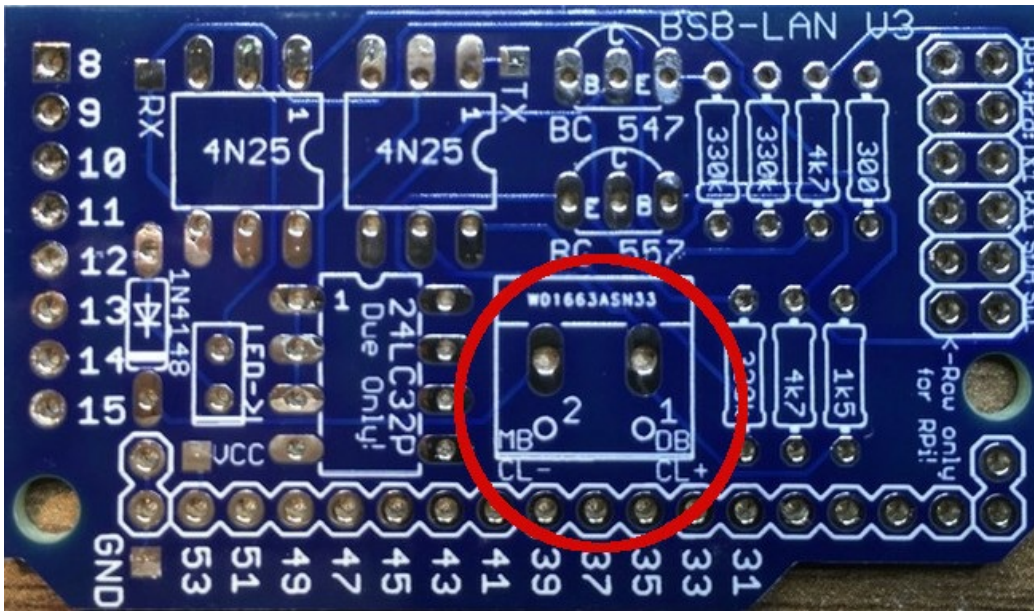
Im Folgenden wird der Anschluss des BSB-LAN-Adapters an die möglichen Anschlüsse der verschiedenen Heizungsregler ausführlich beschrieben.

Sollte man bereits die entspr. Anschlüsse ausfindig gemacht haben, so lässt sich das Procedere auf drei Schritte verkürzt beschreiben:

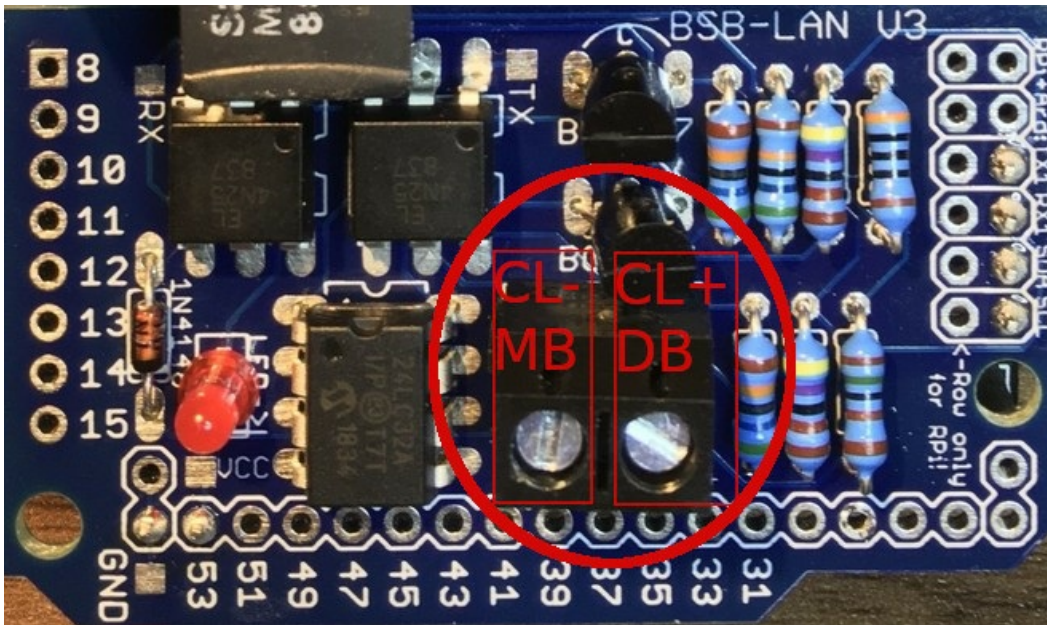
1. Schalte deine Heizung aus, damit der Heizungsregler stromlos ist.
2. Schließe nun den BSB-LAN-Adapter an den Regler an. Verbinde dazu die reglerseitigen Anschlüsse "CL+" und "CL-" (bei BSB-Verwendung) bzw. "DB" und "MB" (bei LPB-Verwendung) mit den gleichnamigen Anschlüssen des Adapters. Achte auf die korrekte Verbindung: Die verbundenen Anschlüsse müssen *namensgleich* sein, also bspw. "CL+" an "CL+" und "CL-" an "CL-"!
3. Schalte die Heizung bzw. den Heizungsregler wieder ein.

**Adapterplatine:**

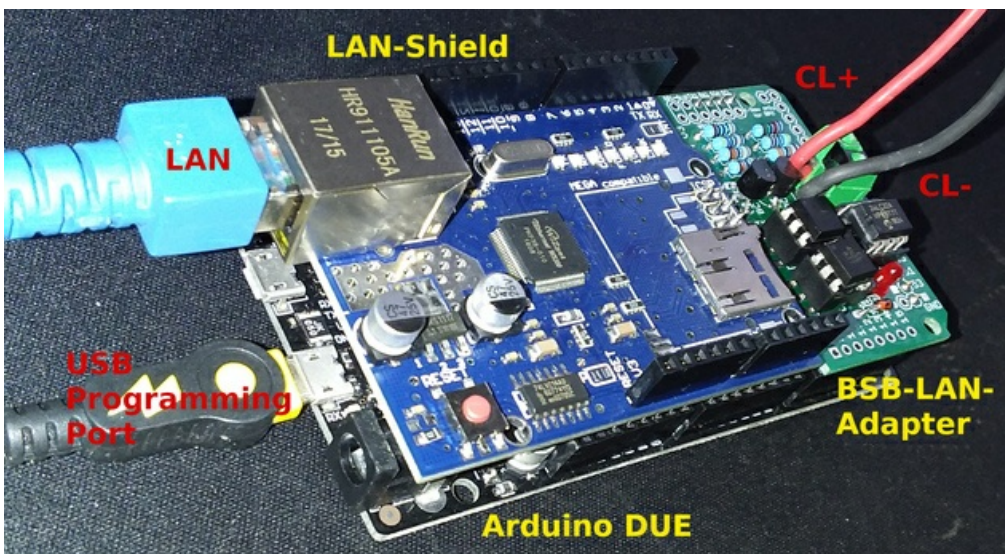
Bei der Adapterplatine sind die Anschlüsse mit CL+/DB und CL-/MB gekennzeichnet. Bei einem Nachbau ist der Schaltplan zu beachten.



Die unbestückte Platine.



Die bestückte Platine.



Das komplette Setup (Arduino Due + LAN-Shield + BSB-LPB-LAN-Adapter v3) inklusive der entsprechenden Kabel.

**BSB:**

Der Anschluss des Adapters erfolgt an den beschriebenen Pins des BSB mit 'Plus an Plus' und 'Minus an Minus':

Adapter-"CL+" an Regler-"CL+" sowie

Adapter-"CL-" an Regler-"CL-".

Der zusätzliche Anschluss „G+“ beim BSB führt konstante 12V und ist für die Hintergrundbeleuchtung der entsprechenden Raumgeräte vorgesehen. Dieser ist für den Anschluss des Adapters NICHT zu verwenden!

(Sollte der Adapter irrtümlicherweise an G+ statt an CL+ angeschlossen werden, so leuchtet zwar die LED, allerdings ist keinerlei Funktion gegeben.)

**LPB:**

Der Anschluss des Adapters erfolgt an den beschriebenen Pins des LPB, meist mit DB und MB gekennzeichnet:

Adapter-"DB" an Regler-"DB" sowie

Adapter-"MB" an Regler-"MB".

**PPS:**

Hier sind es häufig die Anschlüsse A6 und M oder MD, wobei dann

"A6" an "CL+" und

"M"/"MD" an "CL-"

des Adapters anzuschließen ist.

***Kennzeichnung der Anschlüsse:***

- Der **BSB** ist hersteller- und reglerübergreifend leider nicht einheitlich gekennzeichnet. Mögliche Bezeichnungen sind u.a.:
  - „CL+ & CL-“
  - „FB“ (Fernbedienung = Raumgerät)
  - „BSB“ (bei FB und BSB manchmal zusätzlich mit Nennung der Pole „CL+ & CL-“)
  - „BSB & M“ (bei der Kennzeichnung „BSB & M“ entspricht BSB → CL+ und M → CL-)
  - "X86" bei einem RVS21-Regler (s. Abb. weiter unten)

Ist bei den jeweiligen Anschlüssen lediglich eine Nummerierung zu erkennen (häufig bei "FB": 1,2,3), so ist in der spezifischen Bedienungsanleitung nachzusehen, welche dieser Pins mit CL+ und CL- belegt sind.

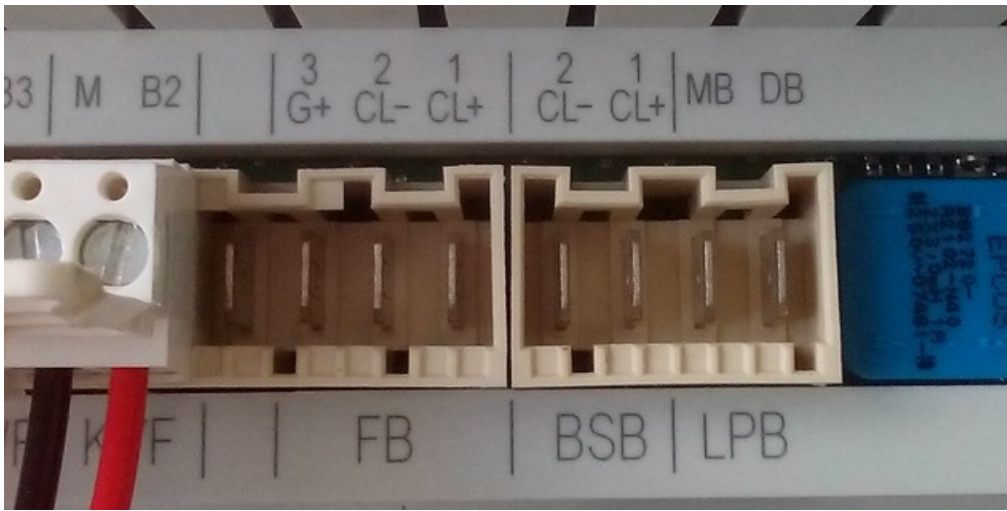
Bitte den Anschluss G+ *nicht* benutzen, dies ist kein Buspin!

Der Anschluss des Adapters an den BSB erfolgt wie erwähnt analog zu dem Anschluss von Raumgeräten. In der gerätespezifischen Bedienungsanleitung finden sich diesbzgl. die reglerspezifischen Angaben. Am Regler selbst sind manchmal auch kleine Abbildungen angebracht, die ein Raumgerät darstellen sollen - auch dies kann zum Auffinden des benötigten Anschlusses hilfreich sein.

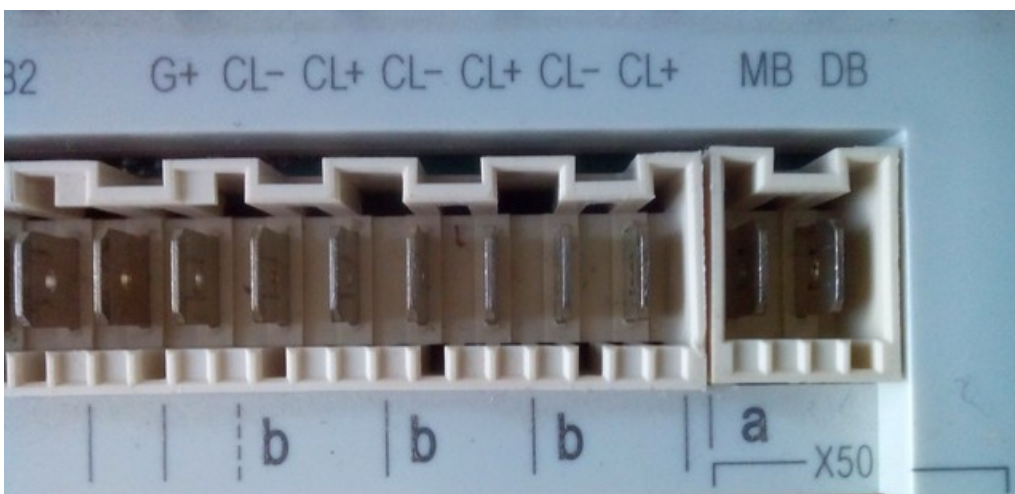
- Der **LPB** ist bei einigen Reglern als solcher gekennzeichnet, manchmal aber auch nur mit den Bezeichnungen „DB“(+) und „MB“(-) versehen.

**Die folgenden Abbildungen zeigen exemplarisch die Anschlüsse bei verschiedenen Reglermodellen:**

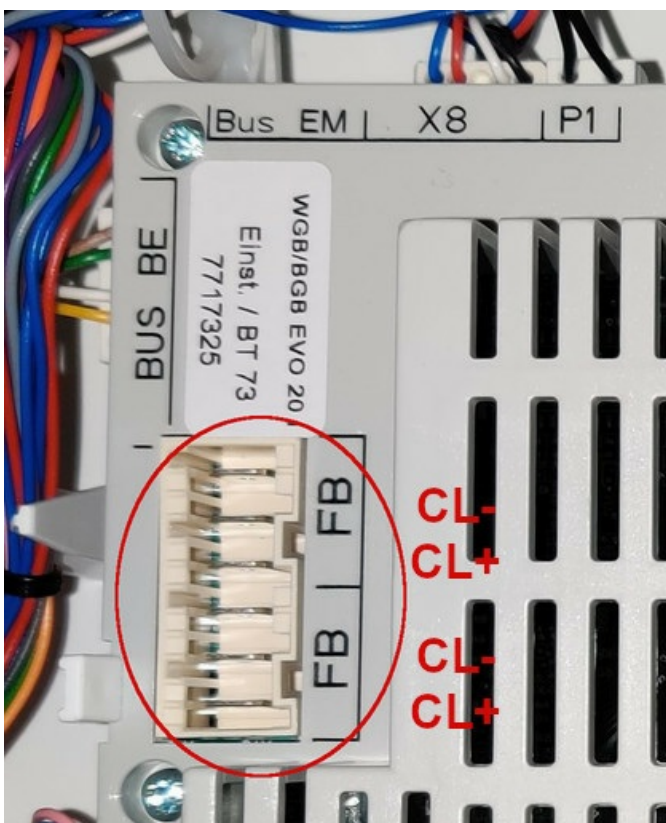




BSB (FB mit CL+ & CL-) und LPB (DB & MB) bei einem Brötje ISR-RVS43.222-Regler.



Anschlüsse b = BSB (CL+ & CL-) und a = LPB (DB & MB) bei einem Siemens RVS63.283-Regler.

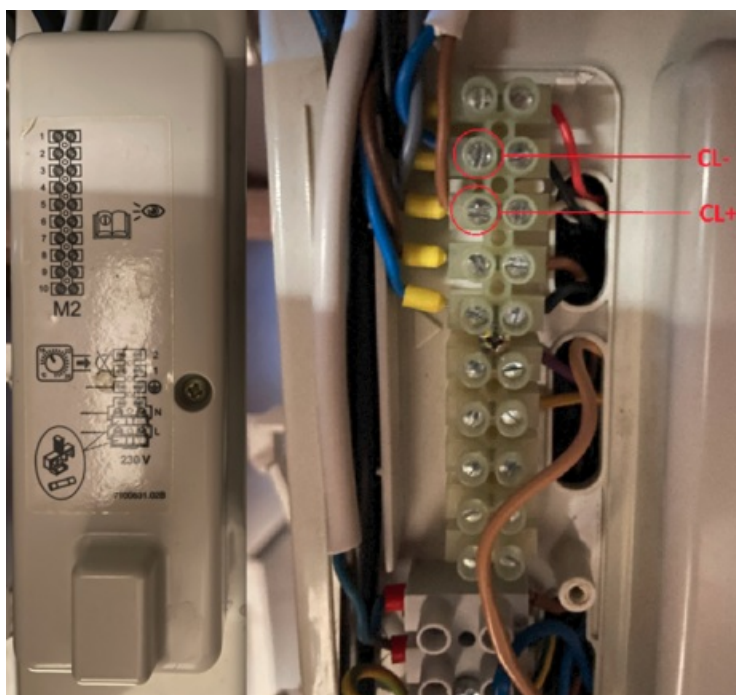




BSB am "FB"-Anschluss bei einem LMS1x-Regler.



BSB am "X86"-Anschluss eines RVS21-Reglers.



BSB am "M2"-Anschlussblock (hinter der Kunststoffabdeckung links im Bild) einer Baxi Luna Platinum.

User "olympia" hat freundlicherweise eine Anleitung für das Anschließen für die Baxi Luna Platinum geschrieben und auf [seinem GitHub-Account](#) zur Verfügung gestellt. Vielen Dank dafür!



BSB (CL+ & CL-) an der vierpoligen Servicebuchse vorne in der Bedieneinheit eines ISR Plus. Die (dauerhafte) Verwendung dieses Anschlusses ist aufgrund einer mangelnden Zugentlastung jedoch nicht zu empfehlen.

Hinweise zum Anschlussstecker
<p>Der Anschluss der Leitungen an die jeweiligen Kontakte sollte prinzipiell immer mit den spezifischen Steckern erfolgen, sofern diese vorhanden sind. Eine umfassende Nennung der entsprechenden Stecker kann hier leider nicht erfolgen, da die Stecker kodiert und teilweise unterschiedlich belegt sind. Meist findet man aber in den Bedienungsanleitungen Teilenummern der passenden Stecker, um ein Raumgerät an den Regler anzuschließen.</p> <p>Beispielhaft sei hier der Stecker für den dreipoligen FB-Anschluss genannt, der bei den meisten Reglern zu passen scheint: <a href="#">Brötje Stecker Raumgerät ISR, Rast 5- 3pol. = 627528</a></p>
<b>BSB / LPB / PPS:</b> Sollten die originalen Stecker nicht unmittelbar erhältlich oder verfügbar sein, können auch (möglichst isolierte) 6,3mm-Kabelschuhe verwendet werden.
<b>Vierpoliger Servicestecker:</b> Für den (vorübergehenden) Anschluss am vierpoligen Servicestecker vorne am Bedienteil können 2,54mm DuPont-Stecker (weiblich) genutzt werden. Diese finden sich bspw. bei den typischen Breadboard-Verbindungskabeln und bei diversen Kabeln im Desktop-PC-Bereich (bspw. interner Lüfteranschluss, interner Lautsprecher).

Hinweise zum Kabel
<b>LPB:</b> Um vor Störeinflüssen möglichst geschützt zu sein, sollten die Anschlusskabel für den LPB-Anschluss gemäß des Siemens Dokuments "CE1N2032D Local Process Bus LPB Projektierungsgrundlagen" einen Querschnitt von 1,5mm <sup>2</sup> aufweisen, zweiadrig verdreht und geschirmt sein (Leitungslänge max. 250m pro Busteilnehmer, max. Gesamtlänge 1000m). <i>Wo bzw. wie</i> die Schirmung anzuschließen ist, wird jedoch nicht erwähnt.
<b>BSB:</b> Für den BSB-Anschluss sind Cu-Leitungen mit mindestens 0,8mm <sup>2</sup> (bis 20m) Querschnitt zu wählen, bspw. LIYY oder LiYCY 2 x 0,8. Bei Leitungslängen bis 80m sollte 1mm <sup>2</sup> , bis 120m sollten 1,5mm <sup>2</sup> Querschnitt gewählt werden.
Generell ist eine parallele Verlegung mit Netzleitungen zu vermeiden (Störsignale).
Entgegen der offiziellen Empfehlungen berichteten verschiedene Nutzer von einem problemlosen Betrieb mit Telefon-Verlegekabeln, 0.5-0.75mm Lautsprecherkabeln, Netzerkabeln (bei denen dann zwei oder drei Adern pro Busleitung zusammen angeschlossen wurden) etc. Bevor also ein Kauf neuer Kabel getätigt wird, kann auch bereits vorhandenes Kabel getestet werden.

## 3.2 Funktionsüberprüfung und erste Nutzung

Um zu überprüfen, ob der angeschlossene Regler korrekt erkannt wird, sollte bei der Ersteinrichtung eine Funktionsüberprüfung vorgenommen werden.

Dazu bietet sich folgende Vorgehensweise an:

1. Den Regler ausschalten und *polrichtig* mit dem Adapter via BSB verbinden.

| Hinweis | |-----| | Wenn (später) der LPB genutzt werden soll, muss sowohl der Bus-Typ in der Datei BSB\_LAN\_config.h als auch der Anschluss am Regler geändert werden! |

2. Den Regler einschalten und überprüfen, ob die rote LED auf dem Adapter leuchtet.  
Sollte die LED in unregelmäßigen Abständen flackern, ist dies keine Fehlfunktion, sondern zeigt Aktivität auf dem Bus an.
3. Den Arduino Due nun via USB (nutze den "Programming Port" in der Boardmitte) bzw. das ESP32-Board via USB mit dem PC und via LAN/WLAN mit dem Netzwerk verbinden.
4. Nun die Arduino IDE starten, den korrekten Anschluss des Arduino Due bzw. ESP32-Boards auswählen (COM-Port) und dann unter ‚Werkzeuge‘ den ‚Seriellen Monitor‘ starten. Achte darauf, dass unten rechts die Übertragungsgeschwindigkeit auf 115200 Baud eingestellt ist.
5. Wird der angeschlossene Regler automatisch korrekt erkannt, steht am Anfang der Ausgabe des seriellen Monitors bei „Device family“ und „Device variant“ jeweils ein Wert, der nicht „0“ ist.

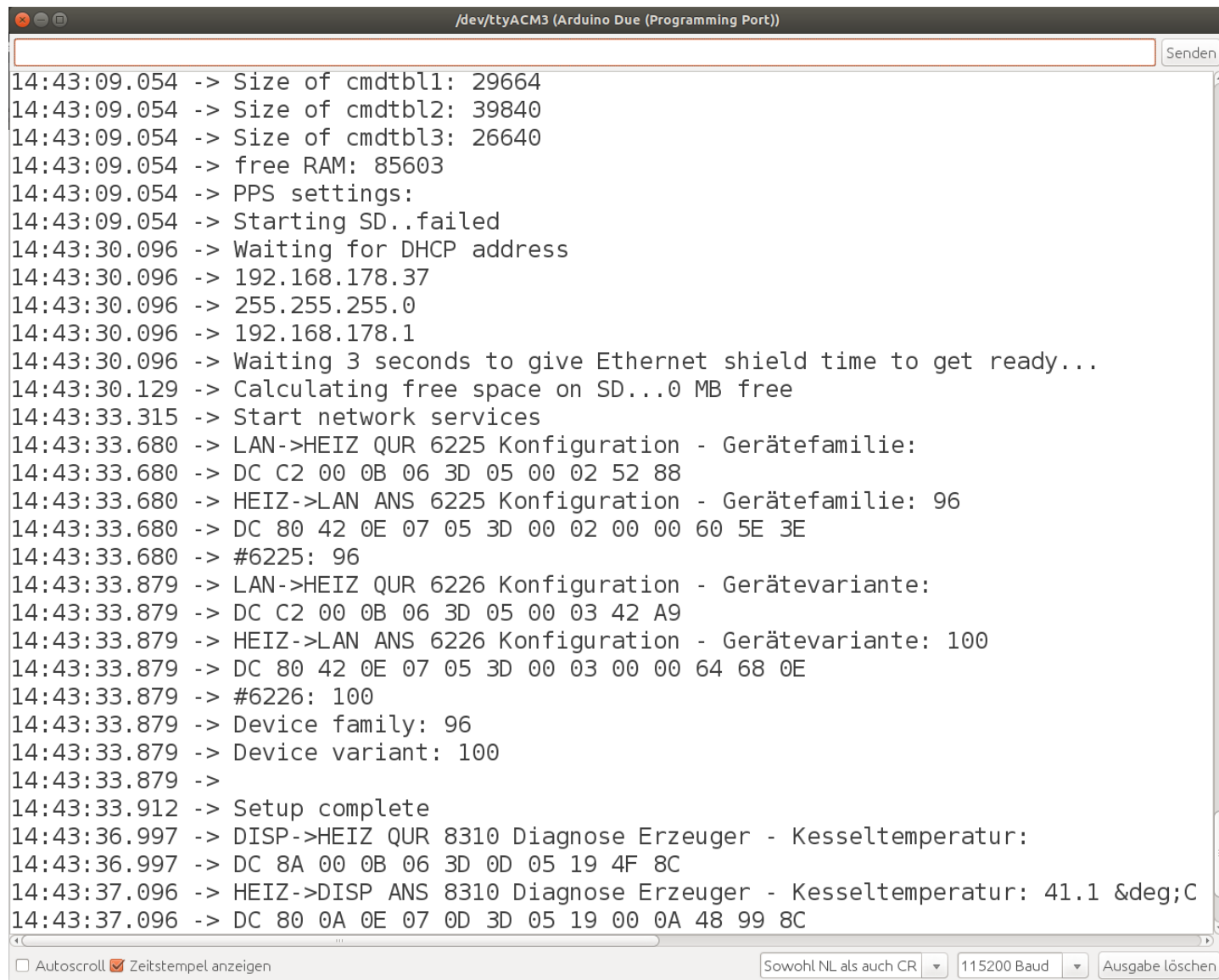
Die Ausgabe eines korrekt erkannten Reglers sieht bspw. so aus:

```
\[...\]  
Device family: 96  
Device variant: 100  
\[...\]
```

Die folgende Abbildung zeigt exemplarisch den Auszug einer solchen Ausgabe des 'Seriellen Monitors' der Arduino IDE nach erfolgreichem Start. Der Adapter ist im folgenden Beispiel mit der Standardeinstellung als "LAN" konfiguriert und fragt zur automatischen Erkennung beim Startvorgang einmalig die Parameter 6225 und 6226 des Heizungsreglers ab.

Die darauf folgenden Zeilen sind bereits empfangene Telegramme.

Die Anzeige des kesselseitigen Steuerungsdisplays (hier: Kesseltemperatur) erscheint regelmäßig als sog. Broadcast (BC) vom Heizungsregler (Kennung "HEIZ").



```
14:43:09.054 -> Size of cmdtbl1: 29664  
14:43:09.054 -> Size of cmdtbl2: 39840  
14:43:09.054 -> Size of cmdtbl3: 26640  
14:43:09.054 -> free RAM: 85603  
14:43:09.054 -> PPS settings:  
14:43:09.054 -> Starting SD..failed  
14:43:30.096 -> Waiting for DHCP address  
14:43:30.096 -> 192.168.178.37  
14:43:30.096 -> 255.255.255.0  
14:43:30.096 -> 192.168.178.1  
14:43:30.096 -> Waiting 3 seconds to give Ethernet shield time to get ready...  
14:43:30.129 -> Calculating free space on SD...0 MB free  
14:43:33.315 -> Start network services  
14:43:33.680 -> LAN->HEIZ QUR 6225 Konfiguration - Gerätefamilie:  
14:43:33.680 -> DC C2 00 0B 06 3D 05 00 02 52 88  
14:43:33.680 -> HEIZ->LAN ANS 6225 Konfiguration - Gerätefamilie: 96  
14:43:33.680 -> DC 80 42 0E 07 05 3D 00 02 00 00 60 5E 3E  
14:43:33.680 -> #6225: 96  
14:43:33.879 -> LAN->HEIZ QUR 6226 Konfiguration - Gerätevariante:  
14:43:33.879 -> DC C2 00 0B 06 3D 05 00 03 42 A9  
14:43:33.879 -> HEIZ->LAN ANS 6226 Konfiguration - Gerätevariante: 100  
14:43:33.879 -> DC 80 42 0E 07 05 3D 00 03 00 00 64 68 0E  
14:43:33.879 -> #6226: 100  
14:43:33.879 -> Device family: 96  
14:43:33.879 -> Device variant: 100  
14:43:33.879 ->  
14:43:33.912 -> Setup complete  
14:43:36.997 -> DISP->HEIZ QUR 8310 Diagnose Erzeuger - Kesseltemperatur:  
14:43:36.997 -> DC 8A 00 0B 06 3D 0D 05 19 4F 8C  
14:43:37.096 -> HEIZ->DISP ANS 8310 Diagnose Erzeuger - Kesseltemperatur: 41.1 &deg;C  
14:43:37.096 -> DC 80 0A 0E 07 0D 3D 05 19 00 0A 48 99 8C
```

#### Hinweis

Sollten in der Ausgabe des SerMo nur kryptische Zeichen erscheinen, so stelle bitte die Übertragungsrate rechts unten im Fenster auf 115200 Baud.

#### Überprüfen, ob der BSB-LAN-Server erreichbar ist

Als ersten Funktionstest, ob der BSB-LAN-Server erreichbar ist, gib nun die spezifische URL deines Setups ein (bei der Verwendung von DHCP erscheint die IP beim Startvorgang im SerMo). Du solltest nun auf die Startseite des BSB-LAN-Servers gelangen:



Heizungsfunktionen	Sensoren	Ausgabe Logdatei	Reglerspezifische Parameterliste
Einstellungen	URL-Befehle	Handbuch	FAQ

BSB-LAN, Version 2.1.8-20220731102301

**Heizungsfunktionen:** Hiermit können Sie die Funktionen Ihres Heizungssystems abfragen bzw. steuern. Die einzelnen Parameter sind in entsprechende Kategorien aufgeteilt, die Sie anklicken können.

**Einstellungen:** Hier sehen Sie, mit welchen Werten BSB-LAN konfiguriert ist. Eine Änderung dieser Parameter ist über die erweiterten URL-Befehle möglich.

**URL-Befehle:** Zeigt Ihnen eine Übersicht der Befehle an, die Sie über die direkte Eingabe in der Adresszeile des Browsers absenden können. Diese Befehle sind auch für die Anbindung an Hausautomations-Systeme wie FHEM nötig.

Zur weiteren Funktionsüberprüfung fahre nun mit dem Schritt im nächsten Kapitel fort. Sollte wider Erwarten bereits jetzt klar sein, dass eine Fehlfunktion vorliegt (was bspw. dadurch zu erkennen ist, dass bei der o.g. Ausgabe von "Device family" und "Device variant" jeweils eine "0" steht), fahre mit [Kap. 3.4](#) fort.

### 3.3 Reglerspezifische Parameterliste erstellen

#### Hinweis

Das nachfolgend beschriebene Procedere betrifft Regler, die per BSB oder LPB an das BSB-LAN-Setup angeschlossen sind. Solltest du einen Regler per PPS angeschlossen haben, so erübrigt sich das nachfolgend Geschriebene, da die Funktion `/Q` bei PPS-Reglern nicht verfügbar und das Erstellen einer spezifischen Datei `BSB_LAN_custom_defs.h` nicht notwendig ist!

Einschränkungen gibt es ebenfalls bei Reglern, die über einen LPB angeschlossen sind, der durch die Nachrüstung mittels OCI420 Busmodul ClipIn verfügbar geworden ist (also LMU54/64 und LMU74/75 Regler). Hier sollten anfangs zwar die entspr. Gerätedaten aufgeführt werden, der "complete dump" ist jedoch ebenfalls nicht verfügbar. Ob dies auch bei LMS14/15 Reglern mit dem neueren OCI345 Busmodul der Fall ist, ist noch nicht bekannt.

In der Grundversion werden nur sehr wenige ausgewählte Parameter unterstützt, die von allen Reglern unterstützt werden (z.B. Uhrzeit, Geräteidentifikation, Komforttemperatur Heizkreis 1, Außentemperatur). Um jedoch kompletten Zugriff auf deinen spezifischen Regler zu erhalten, muss hierfür erst eine passende Datei `BSB_LAN_custom_defs.h` erstellt werden, die genau die Parameter enthält, die dein Regler aufweist!

Für die Generierung der Textdatei, die zur Erstellung der Datei `BSB_LAN_custom_defs.h` erforderlich ist, klicke oben im Webinterface auf den Button "Reglerspezifische Parameterliste" und dann unten auf "Download".

*Achtung: Diese Abfrage dauert eine Weile - bitte warte, bis der ganze 'complete dump' bzw. der Download der Textdatei abgeschlossen ist!*

Diese Funktion fragt nun alle verfügbaren Parameter des angeschlossenen Reglers ab und speichert das Ergebnis in einer Textdatei.

Diese Textdatei muss im Anschluss an Frederik ([bsb@code-it.de](mailto:bsb@code-it.de)) geschickt werden, woraus dann die gerätespezifische Datei `BSB_LAN_custom_defs.h` für den angeschlossenen Regler erzeugt wird. Gib hierbei bitte außerdem die gewünschte Sprachversion an, die du später für BSB-LAN benutzen möchtest (also Deutsch, Englisch o.ä.).

Nachdem du diese Datei von Frederik erhalten hast, musst du die bisherige `BSB_LAN_custom_defs.h` mit dieser ersetzen und BSB-LAN einmal neu flashen. Erst danach hast du kompletten Zugriff auf alle Funktionen deines Reglers!

#### Hinweis



Hinweis
<p><b>Die gerätespezifische Parameterliste sollte nur über das Bussystem abgerufen werden, das auch später verwendet wird!</b></p> <p>Wenn es also einen LPB-Verbund gibt, muss die gerätespezifische Liste auch zwingend über LPB abgerufen werden, da sonst nicht alle Parameter aller Regler übermittelt werden.</p>
<p>Eine einmal erstellte Liste behält auch bei Updates von BSB-LAN ihren Funktionsumfang. Ein erneutes Erstellen ist daher nicht nötig, solange keine Regler ausgetauscht oder hinzugefügt wurden.</p>
<p>Bei einem LPB-Verbund wird die Kategorienliste vom Hauptgerät übernommen. Parameter von weiteren Reglern im Verbund können aber über die Adressierung über das Ausrufezeichen (siehe URL-Kommandos) direkt abgerufen werden, auch wenn sie in der Kategorienauflistung nicht auftauchen.</p>
<p>Es werden hierbei nur die Parameterdefinitionen des Reglers abgefragt, in keinem Fall werden dabei Konfigurationseinstellungen ausgelesen, gesetzt oder verändert!</p>
<p>Alternativ kann die Datei <code>BSB_LAN_custom_defs.h</code>, die in den vorherigen Versionen verwendet wurde, als Teil der Release-Version 2.2 heruntergeladen werden. Den früheren allgemeinen Parameterlisten fehlen jedoch Hunderte von Parametern - insbesondere von neueren Reglern. Darüber hinaus beinhalten sie eine Vielzahl von Ungenauigkeiten und teilweise auch Fehlern, weswegen wir den Einsatz dieser früheren Parameterliste ausdrücklich <i>nicht mehr empfehlen!</i></p>

## 3.4 Debugging und Fehlersuche

Sollten wider Erwarten Probleme auftauchen und das BSB-LAN-Setup nicht verwendungsfähig sein, so kann dies mehrere Ursachen haben.

Als erste Maßnahme bietet sich immer an, die Verkabelung zu überprüfen und nachzusehen, ob die rote LED auf dem BSB-LAN-Adapter leuchtet.

Als weiterer Schritt ist es immer sinnvoll, den Mikrocontroller zusätzlich an den PC anzuschließen und den Seriellen Monitor (SerMo) der Arduino IDE zu starten. Dort kann der Startvorgang überprüft werden. Sollten in der Ausgabe nur kryptische Zeichenfolgen erscheinen, so ist die eingestellte Baudrate zu überprüfen (unten rechts). Diese sollte auf 115200 Baud eingestellt sein.

Wird der angeschlossene Regler *nicht* automatisch korrekt erkannt, steht bei „Device family“ und „Device variant“ jeweils eine „0“, zusätzlich stehen vor „Device family“ sechs Zeilen „query failed“.

*Beispiel:*

```
[...]
query failed
query failed
query failed
query failed
query failed
query failed
Device family: 0
Device variant: 0
[...]
```

Meist liegt der Grund hierfür dann in einem Problem des Hardware-Setups oder der Verkabelung, da die Parameter 6225 und 6226 nicht erfolgreich abgerufen werden konnten ([Fehlermeldung "query failed"](#)).

Weitere Gründe für Fehlfunktionen sind in den Kapiteln [13](#), [14](#) und [15](#) aufgeführt.



[Weiter zu Kapitel 4](#)

[Zurück zum Inhaltsverzeichnis](#)

## 4. BSB-LAN: Das Webinterface

[Zurück zum Inhaltsverzeichnis](#)

[Zurück zu Kapitel 3](#)

## 4. BSB-LAN: Das Webinterface

Die Startseite des Webinterface wird angezeigt, wenn ohne weitere Parameter auf die URL des Servers zugegriffen wird:

```
http://<IP-Adresse>
```

Bei Verwendung eines Passkeys oder weiterer optionaler Sicherheitsfunktionen muss die URL entsprechend erweitert werden, bei Passkey-Verwendung bspw.:

```
http://<IP-Adresse>/<passkey>/
```

*Bitte den Slash hinter dem Passkey nicht vergessen!*



<a href="#">Heizungsfunktionen</a>	<a href="#">Sensoren</a>	<a href="#">Ausgabe Logdatei</a>	<a href="#">Reglerspezifische Parameterliste</a>
<a href="#">Einstellungen</a>	<a href="#">URL-Befehle</a>	<a href="#">Handbuch</a>	<a href="#">FAQ</a>

BSB-LAN, Version 2.1.8-20220731102301

**Heizungsfunktionen:** Hiermit können Sie die Funktionen Ihres Heizungssystems abfragen bzw. steuern. Die einzelnen Parameter sind in entsprechende Kategorien aufgeteilt, die Sie anklicken können.

**Einstellungen:** Hier sehen Sie, mit welchen Werten BSB-LAN konfiguriert ist. Eine Änderung dieser Parameter ist über die erweiterten URL-Befehle möglich.

**URL-Befehle:** Zeigt Ihnen eine Übersicht der Befehle an, die Sie über die direkte Eingabe in der Adresszeile des Browsers absenden können. Diese Befehle sind auch für die Anbindung an Hausautomations-Systeme wie FHEM nötig.

Im oberen Bereich des Webinterface sind einige Buttons angeordnet, die einen einfachen und schnellen Zugriff auf bestimmte Funktionen bieten:

- [Heizungsfunktionen](#)
- [Sensoren](#)
- [Ausgabe Logdatei](#)
- [Reglerspezifische Parameterliste](#)
- [Einstellungen](#)
- [URL-Befehle](#)
- [Handbuch](#)
- [FAQ](#)

Der Button "Ausgabe Logdatei" wird in schwarzer Schrift dargestellt, wenn die Loggingfunktion nicht aktiviert ist (wie im obigen Screenshot zu sehen). Ist die Logging-Funktion aktiviert, so heißt die Bezeichnung des Buttons "Zeichne Logdatei".

Unter dem Headerbereich wird die BSB-LAN-Version angezeigt, die derzeit verwendet wird.

BSB-LAN kann prüfen, ob eine neuere Version verfügbar ist und zeigt dieses im unteren Bereich der Seite an. Im Falle eines verfügbaren Updates führt der Link zum ZIP-File des Repos, so dass man direkt vom Webinterface heraus die Datei speichern kann.

Hinweis
Diese Funktion muss aktiviert werden, siehe dazu bitte <a href="#">Kap. 2.2.</a>

**Heizungsfunktionen (URL-Befehl: /K):**  
Prinzipiell sind alle Parameter in Kategorien zusammengefasst, die den im Display der dargestellten Untermenükategorien entsprechen, wenn auf den Regler des Heizungssystems vom integrierten Bedienteil aus zugegriffen wird.  
Ein Klick auf den Menüpunkt „Heizungsfunktionen“ zeigt eine vollständige Übersicht der Kategorien, die wiederum ebenfalls anwählbar sind:

<a href="#">Uhrzeit und Datum</a>	0 - 6
<a href="#">Bedieneinheit</a>	20 - 70
<a href="#">Funk</a>	120 - 140
<a href="#">Zeitprogramm Heizkreis 1</a>	500 - 516
<a href="#">Zeitprogramm Heizkreis 2</a>	520 - 536
<a href="#">Zeitprogramm 3/HKP</a>	540 - 556
<a href="#">Zeitprogramm 4</a>	560 - 576
<a href="#">Zeitprogramm 5</a>	600 - 616
<a href="#">Ferien Heizkreis 1</a>	632 - 648
<a href="#">Ferien Heizkreis 2</a>	649 - 665
<a href="#">Ferien Heizkreis P</a>	666 - 682
<a href="#">Heizkreis 1</a>	700 - 900
<a href="#">Kühlkreis 1</a>	901 - 969
<a href="#">Heizkreis 2</a>	1000 - 1200
<a href="#">Kühlkreis 2</a>	1201 - 1299
<a href="#">Heizkreis 3/P</a>	1300 - 1500
<a href="#">Trinkwasser</a>	1600 - 1680
<a href="#">Verbraucherkreis 1</a>	1850 - 1880
<a href="#">Verbraucherkreis 2</a>	1900 - 1930
<a href="#">Schwimmbadkreis</a>	1950 - 1980
<a href="#">Hx-Pumpe</a>	2008 - 2051
<a href="#">Schwimmbad</a>	2055 - 2080
<a href="#">Vorregler/Zubringerpumpe</a>	2110 - 2150
<a href="#">Vorregler/Zubringerpumpe 2</a>	2160 - 2160
<a href="#">Kessel</a>	2200 - 2682
<a href="#">Sitherm Pro</a>	2700 - 2753
<a href="#">Wärmepumpe</a>	2776 - 3029
<a href="#">Energiezähler</a>	3090 - 3267
<a href="#">Kaskade</a>	3510 - 3590
<a href="#">Zusatzерzeuger</a>	3690 - 3755
<a href="#">Solar</a>	3810 - 3887
<a href="#">Feststoffkessel</a>	4102 - 4204
<a href="#">Pufferspeicher</a>	4708 - 4813
<a href="#">Trinkwasserspeicher</a>	5007 - 5151
<a href="#">Trinkwasser Durchlauferhitzer</a>	5400 - 5544

Allgemeine Funktionen	5570 - 5608
Konfiguration	5700 - 6498
LPB-System	6600 - 6699
Fehler	6704 - 6996
Wartung/Sonderbetrieb	7001 - 7254
Konfiguration Erweiterungsmodule	7300 - 7500
Ein-/Ausgangstest	7700 - 7999
Status	8000 - 8099
Diagnose Kaskade	8100 - 8199
Diagnose Erzeuger	8300 - 8586
Diagnose Verbraucher	8700 - 9075
Feuerungsautomat	9500 - 9652
Benutzerdefiniert	10000 - 14999
PPS-Bus	15000 - 15099
One Wire, DHT & MAX! Sensors	20000 - 20899

Ein Klick auf eine der gezeigten Kategorien (bspw. Heizkreis 1) startet eine Komplettabfrage der jeweiligen Kategorie, also aller Parameter, die in dieser Kategorie verfügbar sind. Nicht verfügbare Parameter (also Parameter, die vom spezifischen Reglermodell nicht unterstützt werden), werden in grauer Schrift mit dem Hinweis "(parameter not supported)" angezeigt:

700 Heizkreis 1 - Betriebsart: 1 - Automatik

Automatik

Set

701 Heizkreis 1 - Präsenztaste (temporäre Abwesenheit): 141 - not found

---

Set

702 Heizkreis 1 - Präsenztaste (temporäre Abwesenheit): (parameter not supported)

703 Heizkreis 1 - Präsenztaste (temporäre Abwesenheit): (parameter not supported)

710 Heizkreis 1 - Komfortsollwert: 20.0 °C

20.0

Set

711 Heizkreis 1 - Komfortsollwert Maximum: 35.0 °C

35.0

Set

712 Heizkreis 1 - Reduziertsollwert: 6.0 °C

6.0

Set

714 Heizkreis 1 - Frostschuttsollwert: 4.0 °C

4.0

Set

720 Heizkreis 1 - Kennlinie Steilheit: 0.60

0.60

Set

721 Heizkreis 1 - Kennlinie Verschiebung: 0.0 °C

0.0

Set

726 Heizkreis 1 - Kennlinie Adaption: 0 - Aus

Aus

Set

730 Heizkreis 1 - Sommer-/ Winterheizgrenze: 30.0 °C

30.0

Set

732 Heizkreis 1 - Tagesheizgrenze: 0.0 °C

0.0

Set

733 Heizkreis 1 - Verlängerung Tagesheizgrenze: (parameter not supported)

734 Heizkreis 1 - Raumsollabsenkung mit Schaltuhr: °C (parameter not supported)

740 Heizkreis 1 - Vorlaufsollwert Minimum: 8.0 °C

8.0

Set

741 Heizkreis 1 - Vorlaufsollwert Maximum: 75.0 °C

75.0

Set

742 Heizkreis 1 - Vorlaufsollwert Raumthermostat: °C (parameter not supported)

744 Heizkreis 1 - Soll Einschaltverhalten Raumthermostat: % (parameter not supported)

746 Heizkreis 1 - Verzögerung Wärmeanforderung: s (parameter not supported)

750 Heizkreis 1 - Raumeinfluss: 10 %

10

Set

760 Heizkreis 1 - Raumtemperaturbegrenzung: --- °C

---

Set

761 Heizkreis 1 - Heizgrenze Raumregler: % (parameter not supported)

## Hinweis

Man kann die nicht verfügbaren Parameter ausblenden lassen, bei einer Kategorie- oder Komplettabfrage werden sie dennoch mit abgefragt. Siehe hierzu bitte [Kap. 2.2](#).

## Sensoren:

Wenn [optionale Sensoren](#) angeschlossen und korrekt konfiguriert sind, dann werden diese hier angezeigt.

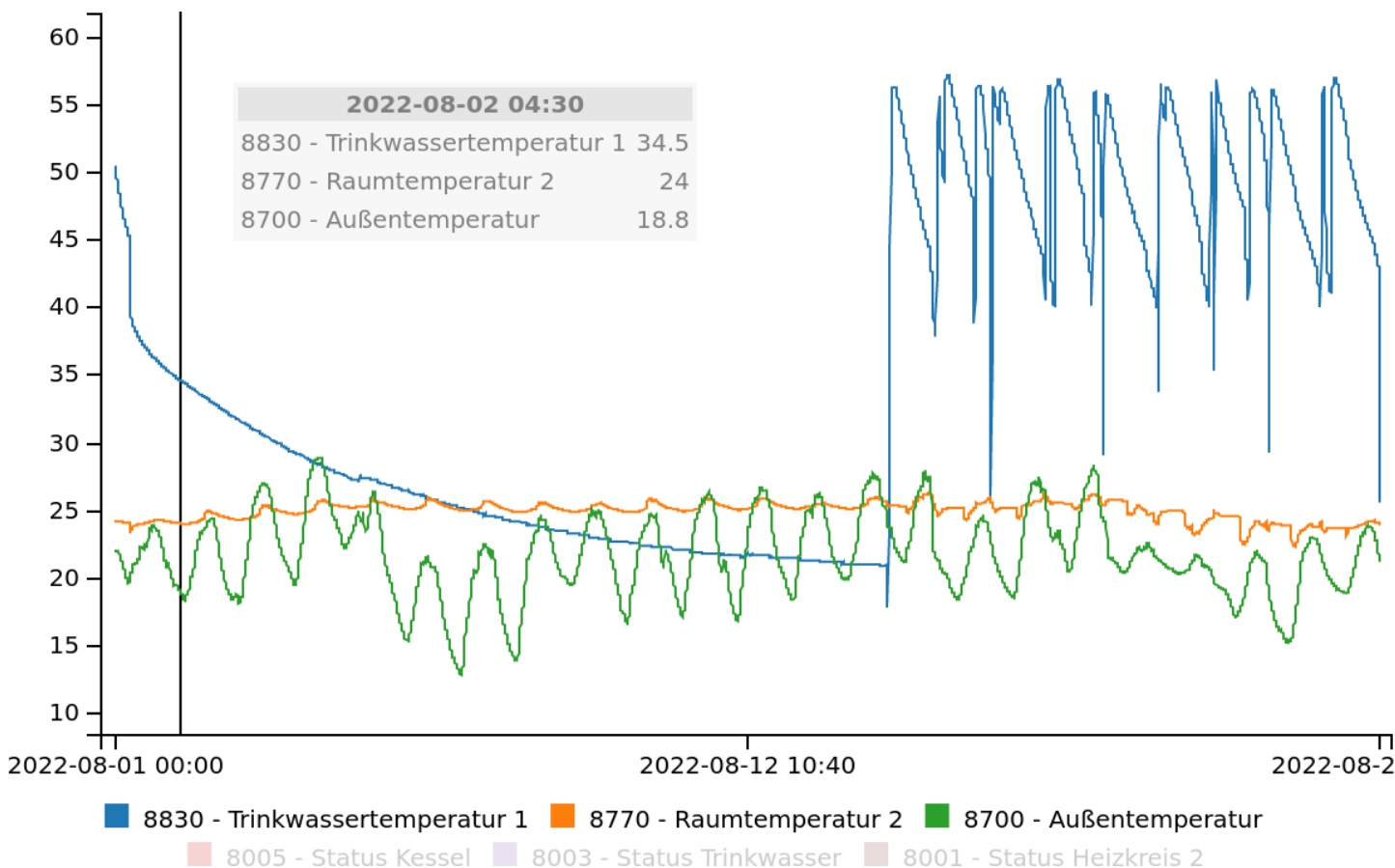
Die folgende Abbildung zeigt beispielhaft die Ausgabe der Messwerte eines angeschlossenen BME280 sowie fünf DS18B20-Sensoren.

20200.0 One Wire, DHT & MAX! Sensors - BME280 Sensor ID #1: 77	77
20200.1 One Wire, DHT & MAX! Sensors - BME280 Sensor Temperatur #1: 22.82 °C	22.82
20200.2 One Wire, DHT & MAX! Sensors - BME280 Sensor Luftfeuchtigkeit #1: 52.31 %	52.31
20200.3 One Wire, DHT & MAX! Sensors - BME280 Sensor Pressure #1: 1004.47 hPa	1004.47
20200.4 One Wire, DHT & MAX! Sensors - BME280 Sensor Altitude #1: 71.27 m	71.27
20300.0 One Wire, DHT & MAX! Sensors - DS18B20 Sensor ID #1: 28701A3711220166	28701A3711220166
20300.1 One Wire, DHT & MAX! Sensors - DS18B20 Sensor Temperatur #1: 20.00 °C	20.00
20301.0 One Wire, DHT & MAX! Sensors - DS18B20 Sensor ID #2: 28EAB2311122011C	28EAB2311122011C
20301.1 One Wire, DHT & MAX! Sensors - DS18B20 Sensor Temperatur #2: 19.81 °C	19.81
20302.0 One Wire, DHT & MAX! Sensors - DS18B20 Sensor ID #3: 28CD3233112201E3	28CD3233112201E3
20302.1 One Wire, DHT & MAX! Sensors - DS18B20 Sensor Temperatur #3: 19.81 °C	19.81
20303.0 One Wire, DHT & MAX! Sensors - DS18B20 Sensor ID #4: 2853B935112201F3	2853B935112201F3
20303.1 One Wire, DHT & MAX! Sensors - DS18B20 Sensor Temperatur #4: 19.87 °C	19.87
20304.0 One Wire, DHT & MAX! Sensors - DS18B20 Sensor ID #5: 288B8D2A112201AF	288B8D2A112201AF
20304.1 One Wire, DHT & MAX! Sensors - DS18B20 Sensor Temperatur #5: 19.87 °C	19.87

## Ausgabe/Zeichne Logdatei (URL-Befehle /D und /DG):

Ist die Funktion des [Loggens auf microSD-Karte](#) aktiviert, erfolgt eine grafische Darstellung des Logfiles (Datei *data.log.txt*) bei Klick auf den Button "Zeichne Logdatei".

Ist die Funktion deaktiviert, so wird der deaktivierte Button in schwarzer Schrift angezeigt und die Bezeichnung lautet "Ausgabe Logdatei".



Mouseover-, Klick- und Mausradaktionen innerhalb der grafischen Darstellung bieten diverse Steuerungsmöglichkeiten:



- verbesserte Lesbarkeit von Wertzahlen bei nahe beieinander liegenden Plotlinien (mouseover auf Plot)
- interaktives Hervorheben von Plotlinien zur besseren Übersicht (Mouseover auf Legendeneinträge)
- interaktives Ausschalten von Plotlinien zur besseren Übersicht und vertikalen Skalierung (Klick auf Legendeneinträge)
- zusätzliche Zoom- (Mausrad/Pinch auf Plot) und Pan-Funktionen (Ziehen des gezoomten Plots)

Hinweise

Für die grafische Anzeige der Logdatei wie im obigen Beispiel dürfen keine JavaScript-Blocker aktiv sein und es muss eine aktive Internetverbindung bestehen, da das JavaScript-Framework zur Darstellung von cdn.jsdelivr.net und d3js.org geladen wird.

Bitte beachte, dass der Mikrocontroller nicht multitaskingfähig ist. Eine neue Abfrage kann erst erfolgen, nachdem die vorhergehende Abfrage komplett beendet ist. Speziell die Abfrage mehrerer Parameter, ganzer Kategorien oder auch des Logfiles der microSD-Karte kann u.U. eine längere Zeit in Anspruch nehmen, während dieser der Adapter nicht ,ansprechbar' ist.

Reglerspezifische Parameterliste (URL-Befehl /Q):

Nach einem Klick auf "Download" am Ende des angezeigten Textes fragt diese Funktion alle Parameter des angeschlossenen Reglers ab und erzeugt die dazugehörige Textdatei mit der spezifischen Ausgabe. Diese Datei muss an Frederik (bsb(ät)code-it.de) geschickt werden, um die entsprechende Datei `BSB_LAN_custom_defs.h` für dieses Reglermodell erstellen zu lassen. Nachdem du diese Datei von Frederik erhalten hast, muss du die bestehende Datei durch diese Version ersetzen und BSB-LAN neu flashen, um vollen Zugriff auf deinen Regler zu erhalten. Für weitere Informationen siehe [Kap. 3.3](#).

Zur Erstellung der reglerspezifischen Parameterliste bitte unten auf 'Download' klicken. Diese Ermittlung kann mehrere Minuten dauern und gerade am Anfang auch vermeintlich pausieren - **bitte warten!** Danach die ausgegebene Datei an bsb(ät)code-it.de schicken. Wir können daraus dann die passende Datei 'BSB\_LAN\_custom\_defs.h' erstellen und bemühen uns, diese dann zeitnah zurück zu schicken.

Alternativ kann die Datei 'BSB\_LAN\_custom\_defs.h', die in den vorherigen Versionen verwendet wurde, als Teil der Release-Version 2.1 heruntergeladen werden. Den früheren allgemeinen Parameterlisten fehlen jedoch Hunderte von Parametern - insbesondere von neueren Reglern. Darüber hinaus beinhalten sie eine Vielzahl von Ungenauigkeiten und teilweise auch Fehlern, weswegen wir den Einsatz dieser früheren Parameterliste ausdrücklich **nicht mehr empfehlen!**

Download

Einstellungen (URL-Befehl: /C):

Hier wird eine Übersicht der Konfiguration dargestellt. Im oberen Bereich ist das [Webinterface zur Konfiguration](#) verfügbar, im unteren Bereich werden nochmals bestimmte Einstellungen (u.a. die genutzte Version von BSB-LAN, die Uptime, der Bustyp, möglicher Schreib- oder Lesezugriff, die definierten Pins für optional angeschlossene Sensoren, die zu loggenden Parameter etc.) aufgelistet.

Einstellungen

Generell	Erweiterte Einstellungen anzeigen	Aus
Generell	Konfiguration aus EEPROM lesen	Ein
Generell	Schreibzugriff (Ebene)	Aus
Generell	Auf Updates überprüfen	Ein
Generell	OTA Update	Aus
Bus	RX Pin Nummer	0
Bus	TX Pin Nummer	0
Bus	Typ	BSB
Bus	Eigene Adresse	CC

**URL-Befehle:**

Der Button ist mit diesem Handbuch verknüpft und führt zum Kapitel [URL-Befehle](#), in dem die möglichen Befehle übersichtlich und kurz aufgeführt sind. Internetzugriff wird benötigt.

**Handbuch:**

Der Button ist mit dem [Inhaltsverzeichnis](#) dieses Handbuchs verlinkt. Internetzugriff wird benötigt.

**FAQ:**

Der Button ist mit dem Kapitel [FAQ](#) dieses Handbuchs verlinkt. Internetzugriff wird benötigt.



[Weiter zu Kapitel 5](#)

[Zurück zum Inhaltsverzeichnis](#)

# 5. BSB-LAN: Abfragen und Steuern

[Zurück zum Inhaltsverzeichnis](#)  
[Zurück zu Kapitel 4](#)

# 5. BSB-LAN: Abfragen und Steuern

Da das Webinterface prinzipiell nur ‚aufgesetzt‘ ist, um eine Bedienung ohne weitere Programme wie bspw. FHEM zu ermöglichen, ist ein direkter Zugriff auf die einzelnen Funktionen und Parameter mittels anderer Programme grundsätzlich möglich.

## 5.1 URL-Befehle

Hinweis	
Bei der folgenden Aufzählung der URL-Befehle muss der jeweilige Wert oder Parameter ohne Klammern geschrieben werden. Beispiel: URL-Befehl <code>/&lt;x&gt;</code> für die einfache Abfrage von Parameter 8700 = <code>/8700</code> .	
URL-Befehl	Auswirkung
<code>/&lt;x&gt;</code>	Alle Werte von Parameter <code>&lt;x&gt;</code> abfragen
<code>/&lt;x&gt;!&lt;adr&gt;</code>	Alle Werte von Parameter <code>&lt;x&gt;</code> für Zieladresse <code>&lt;adr&gt;</code> abfragen
<code>/K&lt;x&gt;!&lt;adr&gt;</code>	Gesamte Kategorie <code>&lt;x&gt;</code> für Zieladresse <code>&lt;adr&gt;</code> abfragen.
<code>/&lt;x&gt;/&lt;y&gt;/&lt;z&gt;</code>	Alle Werte der Parameter <code>&lt;x&gt;</code> , <code>&lt;y&gt;</code> und <code>&lt;z&gt;</code> abfragen Hinweis: Mehrere Abfragen können miteinander verkettet werden, z.B.: <code>http://&lt;IP-Adresse&gt;/K11/8000/8003/8005/8300/8301/8730-8732/8820</code>
<code>/&lt;x&gt;-&lt;y&gt;</code>	Alle Werte eines Parameterbereichs von <code>&lt;x&gt;</code> bis <code>&lt;y&gt;</code> abfragen
<code>/&lt;x&gt;!&lt;adr&gt;-&lt;y&gt;</code>	Alle Werte eines Parameterbereichs von <code>&lt;x&gt;</code> bis <code>&lt;y&gt;</code> für Zieladresse <code>&lt;adr&gt;</code> abfragen
<code>/A=&lt;x&gt;,&lt;y&gt;,&lt;z&gt;</code>	24h-Durchschnittswertberechnung für Parameter <code>&lt;x&gt;</code> , <code>&lt;y&gt;</code> , <code>&lt;z&gt;</code> einstellen Während der Laufzeit kann <code>/A=[parameter1],...,[parameter20]</code> verwendet werden, um (bis zu 20) neue Parameter zu definieren.
<code>/A=0</code>	24h-Durchschnittswertberechnung temporär deaktivieren. Deaktiviert die 24-h Durchschnittswertberechnung vorübergehend bis zum nächsten Reset/Neustart des Mikrocontrollers. Für eine komplette und dauerhafte Deaktivierung müssen alle als zu berechnend aufgeführten Parameter in der Datei <code>BSB_LAN_config.h</code> auskommentiert werden.
<code>/B0</code>	Zurücksetzen des Zählers der akkumulierten Brennerlaufzeiten und -zyklen
<code>/C</code>	Konfigurationsseite (aka Webkonfig) von BSB-LAN
<code>/CO</code>	Anzeige der Konfiguration von BSB-LAN
<code>/D</code> oder <code>/DD</code>	Anzeige der Logdatei Zeigt den Inhalt der Datei <code>datalog.txt</code> an, die sich auf der microSD-Karte im Slot des Ethernet-Shields befindet.
<code>/D0</code>	Logfiles löschen und neuen CSV-header erstellen Dieser Befehl löscht die Dateien <code>datalog.txt</code> und <code>journal.txt</code> und erstellt einen neuen CSV-header für <code>datalog.txt</code> . Dieser Befehl sollte vor dem ersten Logging ausgeführt werden.
<code>/DD0</code>	Logfile <code>datalog.txt</code> löschen
<code>/D&lt;a&gt;,&lt;b&gt;</code>	Anzeige der Daten aus der Logdatei, die im Zeitbereich <code>&lt;a&gt;</code> , <code>&lt;b&gt;</code> liegen <code>&lt;a&gt;</code> und <code>&lt;b&gt;</code> sind dabei im Format <code>jjjj-mm-tt</code> anzugeben, z.B. <code>/D2023-04-01,2023-04-30</code>
<code>/D&lt;n&gt;</code>	Anzeige der neuesten <code>&lt;n&gt;</code> Kalendertage aus der Logdatei

URL-Befehl	Auswirkung
/DG	<p>Grafische Anzeige der Logdatei</p> <p>Wer Parameter auf SD-Karte loggt, hat neben der reinen Textform auch die Möglichkeit, einen Graphen angezeigt zu bekommen.</p> <p>Hinweis: Für /DG müssen bei Javascript-Blockern die Domains <code>cdn.jsdelivr.net</code> und <code>d3js.org</code> freigegeben werden, da der Arduino weiterhin nur die CSV-Datei in den Browser lädt und diese dann mit dem C3/D3-Framework grafisch aufbereitet wird. Alternativ können diese Frameworks auch lokal gespeichert und verwendet werden - Hinweise dazu finden sich in der Datei <code>*BSB_LAN_config.h.default*</code>.</p> <p>Wird die Log-Datei via Webinterface mittels Klick auf „Anzeige Logdatei“ aufgerufen, erfolgt standardmäßig zuerst die grafische Darstellung der Logdaten der jüngsten n Kalendertage (<code>n=DEFAULT_DAYS_TO_PLOT</code>, konfigurierbar in <code>BSB_LAN_config.h</code>). Anschließend lässt sich über Steuerelemente auf der Webseite ein davon abweichender Bereich auswählen, abhängig von den in der Logdatei enthaltenen Daten.</p> <p>Mouseover-, Klick- und Mausradaktionen innerhalb der grafischen Darstellung bieten diverse Steuerungsmöglichkeiten:</p> <ul style="list-style-type: none"> <li>- verbesserte Lesbarkeit von Wertzahlen bei nahe beieinander liegenden Plotlinien (mouseover auf Plot)</li> <li>- interaktives Hervorheben von Plotlinien zur besseren Übersicht (Mouseover auf Legendeneinträge)</li> <li>- interaktives Ausschalten von Plotlinien zur besseren Übersicht und vertikalen Skalierung (Klick auf Legendeneinträge bzw. Alt-Klick um ausschließlich die jeweilige Plotlinie anzuzeigen)</li> <li>- zusätzliche Zoom- (Mausrad/Pinch auf Plot) und Pan-Funktionen (Ziehen des gezoomten Plots)</li> </ul>
/DJ	<p>Anzeige des Logfiles von Telegrammen</p> <p>Dieser Befehl zeigt das Logfile <code>*journal.txt*</code> an, das den Inhalt der empfangenen und gesendeten Telegramme anzeigt. Dieses Logging ist nützlich bei der Fehlersuche und bei der Suche nach unbekannten Parametern. Um die Funktion nutzen zu können, muss das Modul <code>LOGGER</code> in der Datei <code>BSB_LAN_config.h</code> aktiviert und das erste Element des Arrays <code>log_parameters</code> auf 30000 gesetzt werden.</p>
/DJ0	Logfile <code>journal.txt</code> löschen
/DK<n>	Löschen der Daten in der Logdatei, mit Ausnahme von Daten der letzten <n> Kalendertage
/E<x>	<p>Alle ENUM-Werte für Parameter &lt;x&gt; auflisten</p> <p>Bei diesem Befehl kommuniziert der Adapter nicht mit dem Heizungssystem. Es ist eine softwareseitige, interne Funktion. Dieser Befehl ist nur für Parameter des Typs <code>VT_ENUM</code>, <code>VT_CUSTOM_ENUM</code>, <code>VT_BITS</code> und <code>VT_CUSTOM_BITS</code> verfügbar.</p>
/G<x>	<p>GPIO: Abfragen des GPIO-Pins &lt;x&gt; (GPIO wird als OUTPUT genutzt)</p> <p>Gibt den momentanen Status von GPIO Pin &lt;x&gt; an, wobei &lt;y&gt;=0 LOW und &lt;y&gt;=1 HIGH ist.</p>
/G<x>=<y>	<p>GPIO: Abfragen des GPIO-Pins &lt;x&gt; und Setzen auf &lt;y&gt; (GPIO wird als OUTPUT genutzt)</p> <p>Setzt GPIO Pin &lt;x&gt; auf LOW (&lt;y&gt;=0) oder HIGH (&lt;y&gt;=1).</p> <p>Reservierte Pins, die nicht gesetzt werden dürfen, können in der <code>BSB_LAN_config.h</code> unter dem Parameter <code>GPIO_exclude</code> gesperrt werden.</p>
/G<x>,I	<p>GPIO: Abfragen des GPIO-Pins &lt;x&gt; (GPIO wird als INPUT genutzt)</p> <p>Für die reine Abfrage eines externen Gerätes, das an einen GPIO angeschlossen ist (z.B. ein einfaches Koppelrelais), da die Pins per default auf ‚output‘ gesetzt sind. Der Pin bleibt nach diesem Befehl so lange auf ‚input‘, bis das nächste Mal mit <code>/G&lt;xx&gt;=&lt;y&gt;</code> ein Wert geschrieben wird - ab da ist er dann bis zum nächsten „I“ wieder auf ‚output‘.</p>
/I<x>=<y>	<p>Sende eine INF-Nachricht für den Parameter &lt;x&gt; mit dem Wert &lt;y&gt;</p> <p>Einige Werte können nicht direkt gesetzt werden. Das Heizungssystem wird mit einer <code>TYPE_INF</code>-Nachricht informiert, bspw. bei der Raumtemperatur: <code>http://&lt;ip-address&gt;/I10000=19.5</code> → Raumtemperatur beträgt 19.5°C</p>
/JB	JSON: Backup aller schreibbaren Parameter des Heizungsreglers (Wiederherstellen mit /JS)
/JC=<x>,<y>,<z>	<p>JSON: Abfrage der möglichen Werte der Parameter &lt;x&gt;,&lt;y&gt;,&lt;z&gt; für Parameter des Typs <code>ENUM</code></p> <p>Das Format der zurückgegeben Daten ist das gleiche wie bei dem Befehl <code>/JK=&lt;x&gt;</code>. Im Gegensatz zum Befehl <code>/JQ</code> werden die aktuellen Parameterwerte nicht zurückgemeldet.</p>
/JI	JSON: Konfiguration von BSB-LAN anzeigen lassen
/JK=<x>	JSON: Abfrage der verfügbaren Parameter der Kategorie <x>

URL-Befehl	Auswirkung
/JK=ALL	JSON: Auflistung aller Kategorien samt zugehöriger Parameternummern
/JL	JSON: Erstellt eine Liste der Konfiguration im JSON-Format
/JQ=<x>,<y>,<z>	JSON: Abfrage von Parameter <x>, <y> und <z>
/JQ	JSON: Abfrage von Parametern
/JR<x>	<p>JSON: Fragt den Reset-Wert für Parameter &lt;x&gt; ab</p> <p>Im Display der integrierten Heizungssteuerung gibt es für einige Parameter eine Reset-Option. Ein Reset wird vorgenommen, indem das System nach dem Reset-Wert gefragt wird und dieser anschließend gesetzt wird.</p>
/JS	JSON: Setzen von Parametern
/JV	JSON: Abfrage der JSON-API-Version. Payload: {"api_version": "major.minor"}
/JW	JSON: Liest die per /JL erstellte Konfigurationsliste aus und passt die Einstellungen entsprechend an.
/K	<p>Alle Kategorien auflisten</p> <p>Bei diesem Befehl kommuniziert der Adapter nicht mit dem Heizungssystem. Es ist eine softwareseitige, interne Funktion.</p>
/K<x>	<p>Alle Parameter von Kategorie &lt;x&gt; abfragen</p> <p>Bei diesem Befehl kommuniziert der Adapter nicht mit dem Heizungssystem. Es ist eine softwareseitige, interne Funktion.</p>
/L=0,0	<p>Vorübergehendes Deaktivieren des Loggens auf die microSD-Karte</p> <p>Prinzipiell erfolgt das Aktivieren/Deaktivieren der Log-Funktion durch das entsprechende Definiment in der Datei BSB_lan_config.h vor dem Flashen. Während des Betriebes kann das Loggen jedoch mit diesem Befehl deaktiviert werden. Zum Aktivieren werden dann wieder das Intervall und die gewünschten Parameter eingetragen. Bei einem Reset/Neustart des Mikrocontrollers werden die Einstellungen aus der Datei BSB_lan_config.h verwendet - eine dauerhafte Umstellung der Logging-Parameter sollte also dort erfolgen.</p>
/L=<x>,<y1>,<y2>,<y3>	<p>Setzen des Logging-Intervals auf &lt;x&gt; Sekunden mit (optional) zu loggenden Parametern &lt;y1&gt;,&lt;y2&gt;,&lt;y3&gt;</p> <p>Setzt während der Laufzeit das Logging-Intervall auf &lt;x&gt; Sekunden und (optional) die Logging-Parameter auf &lt;y1&gt;,&lt;y2&gt;, &lt;y3&gt; etc.</p> <p>Dabei sind stets alle zu loggenden Parameter anzugeben - also auch (falls gewünscht) diejenigen, die evtl. bereits in der Datei BSB_lan_config.h definiert wurden. Nach einem Neustart werden dann wieder nur die Parameter geloggt, die in der Datei BSB_lan_config.h definiert wurden.</p> <p>Hinweis: Das Logging muss durch das Definiment #define LOGGING in der Datei BSB_lan_config.h aktiviert werden und kann initial anhand der Variablen log_parameters und log_interval konfiguriert werden.</p>
/LB=<x>	<p>Konfiguration des Loggens von Bus-Telegrammen: Nur Broadcasts (&lt;x&gt;=1) oder alle Telegramme (&lt;x&gt;=0)</p> <p>Wenn Bus-Telegramme geloggt werden (Parameter 30000 als einzigen Parameter loggen), logge nur die Broadcasts (&lt;x&gt;=1) oder alle (&lt;x&gt;=0) Telegramme.</p>
/LD	Deaktivieren des Loggens von Bus-Telegrammen
/LE	Aktivieren des Loggens von Bus-Telegrammen
/LN	Löst ein sofortiges Logging aus und startet das konfigurierte Logintervall zu diesem Zeitpunkt neu.
/LU=<x>	<p>Konfiguration des Loggens von Bus-Telegrammen: Nur unbekannte (&lt;x&gt;=1) oder alle (&lt;x&gt;=0) Telegramme loggen</p> <p>Wenn Bus-Telegramme geloggt werden (Parameter 30000 als einzigen Parameter loggen), logge nur die unbekannten Command IDs (&lt;x&gt;=1) oder alle (&lt;x&gt;=0) Telegramme.</p>
/M<x>	<p>Aktivieren (&lt;x&gt; = 1) oder Deaktivieren (&lt;x&gt; = 0) des Bus-Monitormodus</p> <p>Standardmäßig ist der Monitor-Modus deaktiviert (&lt;x&gt;=0).</p> <p>Wenn &lt;x&gt; auf 1 gesetzt wird, werden alle Bytes auf dem Bus überwacht. Telegramme werden durch Umbruchzeichen als solche erkannt. Jedes Telegramm wird im Hex-Format auf der seriellen Konsole mit einem Zeitstempel in Millisekunden dargestellt. Die Ausgabe der Überwachung betrifft nur die serielle Konsole des Mikrocontrollers, die html-Ausgabe bleibt unverändert.</p> <p>Zum Deaktivieren des Monitor-Modus ist &lt;x&gt; wieder auf 0 zu setzen (URL-Befehl: /M0).</p>



URL-Befehl	Auswirkung
<code>/N</code>	<p>Reset &amp; Neustart des Mikrocontrollers (Dauer: ca. 15 Sekunden)</p> <p>Hinweis: Die Funktion muss zuvor in der Datei <code>BSB_lan_config.h</code> aktiviert werden: <code>#define RESET</code></p>
<code>/NE</code>	<p>Reset &amp; Neustart des Mikrocontrollers (Dauer: ca. 15 Sekunden) und löschen des EEPROMs</p> <p>Hinweis: Die Funktion muss zuvor in der Datei <code>BSB_lan_config.h</code> aktiviert werden: <code>#define RESET</code></p>
<code>/P&lt;x&gt;</code>	<p>Bus-Typ (BSB, LPB oder PPS) vorübergehend ändern: <code>&lt;x&gt; = 0 → BSB / 1 → LPB / 2 → PPS</code></p> <p>Wechselt zwischen BSB (<code>&lt;x&gt;=0</code>), LPB (<code>&lt;x&gt;=1</code>) und PPS (<code>&lt;x&gt;=2</code>). Nach einem Reset/Neustart des Mikrocontrollers wird die Einstellung aus der Datei <code>BSB_lan_config.h</code> verwendet. Um den Bus-Typ dauerhaft festzulegen, sollte die Option <code>setBusType</code> config in der Datei <code>BSB_lan_config.h</code> entsprechend angepasst werden.</p>
<code>/P&lt;x&gt;,&lt;y&gt;,&lt;z&gt;</code>	<p>Bus-Typ und zusätzlich die eigene oder die Zieladresse mittels URL-Befehl wechseln</p> <p><code>&lt;x&gt; = Bus (0 = BSB, 1 = LPB, 2 = PPS),</code></p> <p><code>&lt;y&gt; = eigene Adresse und</code></p> <p><code>&lt;z&gt; = Zieladresse</code></p> <p>Leerwerte bei den Adressen belassen den bisherigen Wert (= voreingestellte Adresse).</p>
<code>/Q</code>	Überprüfen auf nicht-freigegebene reglerspezifische Parameter
<code>/R&lt;x&gt;</code>	<p>Frage den Reset-Wert für Parameter <code>&lt;x&gt;</code> ab</p> <p>Im Display der integrierten Heizungssteuerung gibt es für einige Parameter eine Reset-Option. Ein Reset wird vorgenommen, indem das System nach dem Reset-Wert gefragt wird und dieser anschließend gesetzt wird.</p>
<code>/S&lt;x&gt;!&lt;z&gt;=&lt;y&gt;</code>	<p>Setze Wert <code>&lt;y&gt;</code> für den Parameter <code>&lt;x&gt;</code> mit optionaler Zieladresse <code>&lt;z&gt;</code></p> <p>Die gewünschte Gerätezieladresse ist als <code>&lt;z&gt;</code> einzufügen, wenn <code>!&lt;z&gt;</code> nicht eingegeben wird, wird die Standardzieladresse verwendet. Um einen Parameter auf 'abgeschaltet/deaktiviert' zu setzen, muss lediglich ein leerer Wert eingefügt werden: <code>http://&lt;ip-address&gt;/S&lt;x&gt;=</code></p>
<code>/U</code>	<p>Anzeige der in der Datei <code>BSB_lan_custom.h</code> benutzerdefinierten Variablen (falls verwendet)</p> <p>Für die Erstellung eigener Unterprogramme in der <code>BSB_lan_custom.h</code> stehen zwei globale Arrays, <code>custom_floats[]</code> und <code>custom longs[]</code>, zur Verfügung, die jeweils 20 Byte groß sind. Diese können nach Bedarf verwendet werden und über das URL-Kommando <code>/U</code> angezeigt werden. Dies kann nützlich sein, um z.B. eigene Sensoren anzubinden, die man dann in der <code>BSB_lan_custom.h</code> abfragen bzw. berechnen kann und dann über <code>/U</code> über die Weboberfläche abfragen kann.</p>
<code>/V&lt;x&gt;</code>	<p>Aktivieren (<code>&lt;x&gt; = 1</code>) oder Deaktivieren (<code>&lt;x&gt; = 0</code>) des Verboseitätsmodus</p> <p>Der voreingestellte Verboseitäts-Level ist 1. Somit wird standardmäßig der Bus überwacht und alle Daten werden zusätzlich im Raw-Hex-Format dargestellt.</p> <p>Soll der Modus deaktiviert werden, so ist <code>&lt;x&gt;</code> auf 0 zu setzen (URL-Befehl: <code>/V0</code>).</p> <p>Der Verboseitäts-Level betrifft sowohl die serielle Konsole des Mikrocontrollers als auch (optional) das Loggen der Bus-Daten auf die microSD-Karte, so dass die Speicherkarte u.U. sehr schnell voll wird! Im Fall des Loggens auf die interne microSD-Karte ist es daher empfehlenswert, den Verboseitätsmodus bereits in der Datei <code>BSB_lan_config.h</code> zu deaktivieren (<code>byte verbose = 0</code>).</p> <p>Die html-Ausgabe bleibt mit <code>/V1</code> unverändert.</p>
<code>/W</code>	Mit vorangehendem <code>/W</code> liefern die URL-Befehle C, S und Q Daten ohne HTML-header und -footer zurück (bspw. <code>/WC</code> oder <code>/WS&lt;x&gt;=&lt;y!z&gt;</code> ); Modul WEBSERVER muss kompiliert sein!
<code>/X</code>	<p>Abfragen optionaler MAX!-Thermostate</p> <p>Abfragen und Anzeigen der Temperaturen von optionalen MAX!-Thermostaten.</p> <p>Hinweis: MAX!-Komponenten müssen zuvor in der Datei <code>BSB_lan_config.h</code> definiert werden!</p>

## 5.2 MQTT

BSB-LAN unterstützt das MQTT-Protokoll, d.h. die Werte und Einstellungen des Heizungsreglers sind per MQTT empfangbar.

Um MQTT bei BSB-LAN zu nutzen, muss zwingend das Definiment `"#define LOGGER"` in der Datei `BSB_LAN_config.h` aktiviert sein. Dies ist in der Voreinstellung bereits der Fall.

Die zu sendenden (von BSB-LAN abgefragten) Parameter, das Sendeintervall (nur eines für alle Parameter möglich!) sowie die weiteren MQTT-spezifischen Einstellungen (Broker, Topic etc.) sind entweder via Webkonfiguration oder direkt in der Datei `BSB_LAN_config.h` einzustellen.

Beachte hierzu bitte die Erklärungen in den entspr. Unterkapiteln von [Kap. 5](#).

Beispiele für eine Einbindung von BSB-LAN findest du in den entspr. Unterkapiteln von [Kap. 11](#).

#### Hinweis

Wenn du die MQTT-Funktion per fest eingestellter Loggingparameter und -intervall nutzt, achte darauf, dass du das Logintervall (= MQTT-Sendeintervall) anpasst!  
Per default ist hier 3600 eingestellt, was bedeutet, dass die Parameter alle 3600 *Sekunden*, also alle 60 *Minuten* und somit *stündlich* gesendet werden! Wenn du also deinen MQTT-Broker konfiguriert hast und dich wundern solltest, warum keine Werte ankommen, überprüfe zuerst das Logintervall!

BSB-LAN sendet über den Subtopic "status" unter dem definierten "MQTTTopicPrefix" jederzeit seinen Online-Status. Dies ist in der Voreinstellung also "BSB-LAN/status". Über diesen Topic kann so jederzeit festgestellt werden, ob der BSB-LAN derzeit Werte sendet und Kommandos empfangen kann.

Ist BSB-LAN verfügbar, ist im Topic die Nachricht "online" zu sehen, ansonsten wird "offline" gesetzt. Die Nachricht ist per Retain-Flag dauerhaft verfügbar, der Subscriber muss also nicht zum Zeitpunkt des BSB-LAN Starts bereits den Topic abonniert haben.

Ein Neustart über die Software (z.B. per URL-Befehl /N) setzt den Topic sofort auf "offline". Fällt BSB-LAN unkontrolliert aus (z.B. durch Stromausfall oder Flashen der Firmware) verschickt der Broker nach einem Timeout (dieser ist m.W. abhängig vom Broker) ebenfalls die Offline-Nachricht.

Neben dem (brokerseitigen) reinen Empfangen ist es auch möglich, via MQTT vom Broker aus sowohl Abfragen als auch Steuerbefehle (URL-Befehle /S und /I) an BSB-LAN zu senden. Selbstverständlich muss BSB-LAN für das Umsetzen von Steuerbefehlen Schreibzugriff auf den Regler gewährt werden.

Die Befehlssyntax lautet:

```
set <MQTT-Server> publish <Topic> <Befehl>
```

- **<MQTT-Server>** = Name des MQTT-Servers
- **<Topic>** = In der Voreinstellung "BSB-LAN", ansonsten das in der Datei *BSB\_LAN\_config.h* entspr. definierte "MQTTTopicPrefix". Sollte kein Topic definiert sein (nicht ratsam), so muss als Topic "FromBroker" genommen werden.
- **<Befehl>** = Die abzufragende Parameternummer oder der entspr. parameterspezifische URL-Befehl **S** oder **I**.

| Achtung | |:------| | Es ist jeweils nur eine Abfrage bzw. nur ein Setzen möglich, es können also keine Parameterbereiche o.ä. abgefragt werden! |

Nachfolgend schickt BSB-LAN eine Empfangsbestätigung zurück ("ACK\_ \<Befehl>").

#### Beispiel

Der Befehl `set mqtt2Server publish BSB-LAN S700=1` sendet vom MQTT-Broker namens "mqtt2Server" den Befehl "S700=1" mit dem Topic "BSB-LAN" und bewirkt eine Betriebsartumschaltung in den Automatikmodus.

Der Befehl `set mqtt2Server publish BSB-LAN 700` sendet vom MQTT-Broker namens "mqtt2Server" den Befehl "700" mit dem Topic "BSB-LAN" und bewirkt eine Abfrage von Parameter 700.

#### Beispiel für Mosquitto

Befehl zum Abrufen von Parameter 1010 (inkl. Username & Passwort): `mosquitto_pub -h 192.168.178.35 -u USER -P PASSWORD -m "1010" -t BSB-LAN -d`

Befehl zum Setzen von Parmeter 1610 auf 41° (inkl. Username & Passwort): `mosquitto_pub -h 192.168.178.35 -u USER -P PASSWORD -m "S1610=41" -t BSB-LAN -d`

## 5.3 JSON

- **Abfrage von Kategorien:**

```
http://<IP-Adresse>/JK=<xx>
```

Abfrage einer spezifischen Kategorie (\<xx> = Kategorienummer)

```
http://<IP-Adresse>/JK=ALL
```

Abfrage aller Kategorien (samt Min. und Max.)

- **Abfragen und Setzen von Parametern per HTTP POST:**

Hierbei ist der Aufruf der URL

`http://<IP-Adresse>/JQ` für eine Abfrage und  
`http://<IP-Adresse>/JS` für das Setzen von Parametern zu verwenden.

Folgende Parameter sind dabei möglich:

```
http://<IP-Adresse>/JQ
Senden: "Parameter"
Empfangen: "Parameter", "Value", "Unit", "DataType" (0 = Zahl, 1 = ENUM, 2 = Bit-Wert (Dezimalwert gefolgt von Bitmaske gefolgt von ausgewählter Option), 3 = Wochentag, 4 = Stunde/Minute, 5 = Datum/Uhrzeit, 6 = Tag/Monat, 7 = String, 8 = PPS-Uhrzeit (Wochentag, Stunde:Minute)), "readonly" (0 = read/write, 1 = read only parameter), "error" (0 - ok, 7 - parameter not supported, 1-255 - LPB/BSB bus errors, 256 - decoding error, 257 - unknown command, 258 - not found, 259 - no enum str, 260 - unknown type, 261 - query failed), "isswitch" (1 = it VT_ONOFF or VT_YESNO data type (subtype of ENUM), 0 = all other cases)

http://<IP-Adresse>/JS
Senden: "Parameter", "Value", "Type" (0 = INF, 1 = SET)
Empfangen: "Parameter", "Status" (0 = Fehler, 1 = OK, 2 = Parameter read-only)
```

- Die Abfrage mehrerer Parameter mit einem Befehl ist ebenfalls möglich:

Der Befehl `http://<IP-Adresse>/JQ=<x>,<y>,<z>` fragt die Parameter \<x>, \<y> und \<z> ab.

- Beispiel zum Setzen von Parametern per *Linux-Kommandozeile* oder „*Curl for Windows*“, exemplarisch am Parameter 700 (Betriebsart HK1) → Setzen auf 1 (automatisch):

Linux-Kommandozeile:

```
curl -v -H "Content-Type: application/json" -X POST -d '{"Parameter": "700", "Value": "1", "Type": "1"}' http://<IP-Adresse>/JS
```

Curl for Windows:

```
curl -v -H "Content-Type: application/json" -X POST -d "{\"Parameter\": \"700\", \"Value\": \"1\", \"Type\": \"1\"}" http://<IP-Adresse>/JS
```

**User "hacki11" hat eine ausführliche und interaktive [API-Dokumentation zum Abfragen und Steuern via JSON](#) erstellt.**  
**Vielen Dank!**

#### Hinweis für Entwickler

Die API lässt sich mittels [Postman](#) am eigenen System testen. Dazu muss man die URL [https://raw.githubusercontent.com/fredcore/bsb\\_lan/master/openapi.yaml](https://raw.githubusercontent.com/fredcore/bsb_lan/master/openapi.yaml) in File/Import/Link hinzufügen und ggf. die spezifischen Angaben wie bspw. Adresse und Zugangsdaten anpassen.

Servers

http://bsb-lan

Authorize

General

GET

/JI Query configuration of BSB-LAN

GET

/JV Queries the JSON-API version

Parameter

POST

/JC List parameter definitions

GET

/JC={parameterIds} List parameter definitions

POST

/JQ List parameters

GET

/JQ={parameterIds} List parameters

Parameters

Try it out

Name	Description
<b>parameterIds</b> <span>★ required</span>	One or more comma separated parameter ids
string	Example : 700,710
(path)	<div>700,710</div>

Neben den Beschreibungen samt Beispielen zu den einzelnen Befehlen sind ebenfalls sämtliche Informationen zu den Typen, Formaten, möglichen Werten etc. aufgeführt.



## Schemas

### ApiVersion >

### Average >

### AverageList >

### Category v {

description: Category definition

name **string**  
Name of category

min **integer(\$int32)**  
Min parameter ID for category

max **integer(\$int32)**  
Max parameter ID for category

}  
example: `OrderedMap { "name": "Uhrzeit und Datum", "min": 0, "max": 6 }`

### Categories v {

description: HashMap of multiple categories with category ID as key

< \* >: **Category > {...}**  
example: `OrderedMap { "name": "Uhrzeit und Datum", "min": 0, "max": 6 }`

}  
example: `OrderedMap { "0": OrderedMap { "name": "Uhrzeit und Datum", "min": 0, "max": 6 }, "1": OrderedMap { "name": "Bedieneinheit", "min": 20, "max": 70 } }`

### Configuration v {

description: Details of a BSB-LAN configuration object

parameter **string**  
Configuration ID

type **integer(\$int32)**

## Hinweise

JSON-Befehle lassen sich auch per Linux-Kommandozeile oder „[Curl for Windows](#)“ nutzen. Bei der o.g. interaktiven API-Dokumentation können die entspr. Curl-Befehle generiert und danach zur weiteren Nutzung kopiert werden (die IP ist bei der weiteren Verwendung stets anzupassen). Dazu ist wie folgt vorzugehen:

1. Klicke auf die gewünschte Operation, bspw. `/JQ={parameterIds}`.
2. Bei dem aufklappenden Fenster klicke rechts auf "Try it out".
3. Trage den/die gewünschten Parameter ein (im unten gezeigten Beispiel: 700,8300).
4. Klicke auf "Execute".

Im Feld "Responses" werden dann die URL- und Curl-Befehle angezeigt, die man kopieren kann.

Achtung: Die Zeichenkombination `%2C` bei der Auflistung mehrerer Parameter wird von Swagger anstelle des Kommas eingefügt. Solltest du die URL-/Curl-Befehle kopieren und nutzen wollen, so ersetze bitte jedes `%2C` durch ein `,` (Komma)!

GET

/JQ={parameterIds} List parameters

🔒

Parameters

Cancel

Name	Description
<b>parameterIds</b> * required string (path)	One or more comma separated parameter ids <input type="text" value="700,8300"/>

Execute

Clear

Responses

Curl

```
curl -X 'GET' \
'http://bsb-lan/JQ=700%2C8300' \
-H 'accept: application/json'
```

📋

Request URL

```
http://bsb-lan/JQ=700%2C8300
```

Die Ausgabe des URL-/Curl-Befehls.

## 5.4 Spezialparameter & Nummernbereiche

### Spezialparameter

Ab BSB-LAN-Version 3.x mussten bestimmte Funktionen als Spezialparameter mit den Nummern 10000 aufwärts implementiert werden. Die folgende Auflistung zeigt diese Spezialparameter samt Parameternummer und Funktion. Abhängig vom Typ muss das Schreiben entweder per SET- oder der INF-Befehl erfolgen (Schreibzugriff muss gewährt sein). Für weitere Informationen zu den einzelnen Befehlen lies bitte die entspr. Kapitel.

Parameternummer	Funktion	Befehlstyp
10000	Raumtemperatur HK1	INF - <a href="#">s. Kap. 6.3</a>
10001	Raumtemperatur HK2	INF - <a href="#">s. Kap. 6.3</a>
10002	Raumtemperatur HK3	INF - <a href="#">s. Kap. 6.3</a>
10019	Manueller TWW-Push	SET - <a href="#">s. Kap. 6.5</a>
10110	Präsenztaste HK1 (temporärer Heizbetriebswechsel)	SET - <a href="#">s. Kap. 6.4</a>
10111	Präsenztaste HK2 (temporärer Heizbetriebswechsel)	SET - <a href="#">s. Kap. 6.4</a>
10112	Präsenztaste HK3 (temporärer Heizbetriebswechsel)	SET - <a href="#">s. Kap. 6.4</a>

### Nummernbereiche

Die folgende Übersicht zeigt, wie die Nummernbereiche aufgeteilt bzw. vergeben sind.

Nummernbereich	Verwendung
----------------	------------

Nummernbereich	Verwendung
0-9999	Parameter des Reglers
10000-10019	Raumgerätefunktionen (Raumtemperatur & TWW-Push)
10020-10099	Ursprünglich nummernlose Parameter des Reglers
10100-10109	Broadcast-Parameter
10110-10129	Präsenztaste (temporärer Heizbetriebswechsel)
10200-10999	Bereich für manuell hinzugefügte Parameter
20100-20199	Sensoren: DHT22
20200-20299	Sensoren: BME280
20300-20499	Sensoren: DS18B20
20500-20699	MAX!-Sensoren/Komponenten



[Weiter zu Kapitel 6](#)  
[Zurück zum Inhaltsverzeichnis](#)

## 6. BSB-LAN: Spezialfunktionen

[Zurück zum Inhaltsverzeichnis](#)

[Zurück zu Kapitel 5](#)

## 6. BSB-LAN: Spezialfunktionen

### 6.1 Loggen von Daten

#### Verwendung des Adapters als Standalone-Logger mittels BSB-LAN

Stecke eine (möglichst aktuelle) FAT32-formatierte microSD-Karte in den Speicherkartenplatz des Ethernet-Shields, bevor du den Mikrocontroller einschaltest.

Aktiviere vor dem Flashen das Definiment `#define LOGGER` in der Datei `BSB_LAN_config.h`, füge die zu loggenden Parameter zur Variable `log_parameters` hinzu und bestimme das Logintervall mit der Variable `log_interval`. Bitte beachte auch die entsprechenden Punkte in Kap. 2.2.

Kommt ein Olimex ESP32-EVB zum Einsatz (oder wird ein microSD-Kartenadapter an einem ESP32-basierten Board verwendet) und sollen die geloggten Werte auf die microSD-Karte anstatt in den Flash-Speicher geschrieben werden (was sehr zu empfehlen ist!), dann muss das folgende Definiment in der Datei `BSB_LAN_config.h` aktiviert werden: `#define ESP32_USE_SD`.

Später können während der Laufzeit sowohl das Intervall als auch die Logging-Parameter mittels des Befehls `"/L=[Intervall],[Parameter1],...,[Parameter20]"` geändert werden.

Sämtliche Daten werden auf der Karte in der Datei `datalog.txt` im CSV-Format gespeichert und können somit leicht in Excel oder OpenOffice Calc importiert werden.

Der Dateiinhalt kann mit dem URL-Befehl `/D` eingesehen werden, eine graphische Darstellung der Logdateien erfolgt mittels `/DG`.

Um die Datei `datalog.txt` zu löschen und neu zu erstellen, benutze den Befehl `/D0`.

Die Ausführung des URL-Befehls `/D0` sollte außerdem bei der ersten Benutzung erfolgen, da hierdurch die Datei mit dem passenden CSV-Header initiiert wird.

#### Hinweise

Vereinzelte kann es vorkommen, dass bestimmte microSD-Karten nicht problemlos vom LAN-Shield erkannt werden. In diesem Fall wird eine entspr. Fehlermeldung von BSB-LAN ausgegeben. Sollte bei dir dieser Fall eintreten, so probiere es mit einer anderen, möglichst aktuellen microSD-Karte.

Bitte beachte, dass der Mikrocontroller keine exakte Uhr ist. Auch wenn du bspw. das Intervall auf 60 Sekunden eingestellt hast, weicht die in der Datei dargestellte Zeit (welche von der Heizungssteuerung empfangen wird) möglicherweise davon ab - dies kann bis zu einer Sekunde pro Minute betragen. Sollte eine exakte Logzeit unbedingt erforderlich sein, kannst du die durchschnittliche Zeitabweichung zwischen der Mikrocontroller-Zeit und der wirklichen Zeit ermitteln, das Log-Intervall entsprechend anpassen und bspw. 59 Sekunden anstatt 60 Sekunden einstellen.

#### Verwendung des Adapters als Remote-Logger

Neben dem Einsatz komplexer Systeme wie bspw. FHEM und den spezifischen Log-Lösungen kann bspw. folgender Befehl periodisch ausgeführt werden (z.B. per cron job):

```
DATE=`date +%Y%m%d%H%M%S`; wget -qO- http://192.168.178.88/8310/720/710 | egrep "(8310|720|710)" | sed "s/^/$DATE /" >> log.txt
```

Das aus diesem Beispiel resultierende Logfile `\log.txt` enthält die aufgezeichneten Werte der Parameter 8310, 720 und 710. Später kannst du das Logfile basierend auf den Parameternummern sortieren, nutze hierfür den Befehl `\sort\`:

```
sort -k2 log.txt
```

#### Hinweis

Die IP, ggf. aktivierte optionale Sicherheitsfunktionen, die gewünschten Parameter etc. sind im obigen Beispiel anzupassen.

## 6.2 IPWE-Erweiterung

Die IPWE-Erweiterung (IPWE = IP-Wetterdaten-Empfänger) stellt eine Möglichkeit dar, zuvor festgelegte Parameter durch den Aufruf einer kurzen URL darzustellen. Um diese tabellarische Übersicht aufzurufen, muss die folgende URL genutzt werden:

`<IP-Adresse>/ipwe.cgi`

### Hinweis

Sollte die optionale Sicherheitsfunktion des Passkeys verwendet werden, so ist der Passkey in diesem Fall ausnahmsweise NICHT der URL hinzuzufügen!

Sensortyp	Adresse	Beschreibung	Temperatur	Luftfeuchtigkeit	Windgeschwindigkeit	Regenmenge
T	1	Außentemperatur	6.00	0	0	0
T	2	Trinkwassertemperatur 1	59.40	0	0	0
T	3	Kesseltemperatur	52.30	0	0	0
T	4	Status Heizkreis 1	114.00	0	0	0
T	5	1. Brennerstufe T2	255.00	0	0	0
T	6	2. Brennerstufe T8	0.00	0	0	0
T	7	Historie 1 Datum/Zeit	1.01	0	0	0
T	8	Historie 1 Fehlercode	98.00	0	0	0
T	9	284c453d07000082	21.25	0	0	0

Beispiel einer IPWE-Ausgabe.

Um die Funktion der IPWE-Erweiterung zu nutzen, müssen vor dem Flashen des Mikrocontroller zwei Einstellungen in der Datei

`BSB_lan_config.h` vorgenommen werden:

- Das Definement `#define IPWE` muss aktiviert werden.
- Die gewünschten Parameter die dargestellt werden sollen, müssen aufgelistet werden.

Zusätzlich zu den aufgeführten Parametern werden automatisch die Werte optional angeschlossener Sensoren (DHT22 / DS18B20) dargestellt. Wenn DS18B20-Sensoren zum Einsatz kommen, werden außerdem die spezifischen Sensor IDs dargestellt. Dies kann man in der letzten Zeile im obigen Beispiel erkennen, dort lautet die spezifische Sensor ID des einzigen angeschlossenen DS18B20-Sensors "284c453d07000082". Auf diese Weise lassen sich mehrere Sensoren eindeutig identifizieren.

### Hinweise

Sollten aus Versehen Parameter zur Anzeige definiert werden, die das spezifische Heizungssystem nicht aufweist, so wird als jeweiliger Wert "0.00" dargestellt. Das heißt aber nicht, dass der Wert des nicht-unterstützten Parameters "0.00" ist! Es ist daher sinnvoll, vor der Definition der anzuzeigenden Parameter zu überprüfen, ob diese auch wirklich vom Heizungssystem bereit gestellt werden.

Da die IPWE-Erweiterung ursprünglich entworfen wurde, um die Messwerte einer spezifischen Funk-Wetterstation darzustellen, machen nicht alle Spalten der Tabelle Sinn, wie bspw. "Windgeschwindigkeit" oder "Regenmenge". Diese können einfach ignoriert werden. Im Grunde sind bei den normalen Parametern lediglich die beiden Spalten "Beschreibung" und "Temperatur" relevant, da hier die Parameterbezeichnung und der jeweilige Wert dargestellt werden.

Die Darstellungen der jeweiligen Parameterwerte/-einstellungen erfolgen nicht im Klartext, sondern numerisch. Bei dem oben abgebildeten Beispiel ist dies u.a. beim Parameter "1. Brennerstufe T1" gut zu erkennen: Dort steht nicht "Ein", sondern "255", was wiederum "Ein" bedeutet.

## 6.3 Raumtemperatur übermitteln

Mittels einer INF-Nachricht kann eine Raumtemperatur an den Regler gesendet werden, um einen Raumeinfluss bei der Berechnung der VL-Temperatur geltend zu machen. Um diese Funktion zu nutzen, muss die Funktion ‚Raumeinfluss‘ vorher im Regler aktiviert und der Einflussfaktor prozentual festgelegt werden (bspw. Parameter 750 für HK1, Parameter 1050 für HK2).

BSB-LAN muss Schreibzugriff gewährt werden (s. [Kap. 2.2](#)).

Die Raumtemperatur muss regelmäßig in relativ kurzen Intervallen übermittelt werden, bspw. alle ein oder zwei Minuten.



<b>Hinweis</b>
Dieser Parameter ist nicht abrufbar.

Die folgenden Parameter müssen dafür genutzt werden:

- 10000 = Heizkreis 1
- 10001 = Heizkreis 2
- 10002 = Heizkreis 3/P

Beispiel:

Der URL-Befehl für den HK1, um eine Raumtemperatur von 19.5°C zu übermitteln, lautet: `http://<IP-Adresse>/I10000=19.5`

<b>Tipp</b>
Wird nur ein Temperaturwert als Einflussfaktor gemessen und übermittelt, ist die Temperaturmessung in einem Führungs- / Referenzraum zu empfehlen, in dem sich keinerlei weitere Wärmequelle (bspw. Kaminofen, große Fenster in Südlage etc.) befindet.
Werden die Raumtemperaturen von mehreren Räumen erfasst (bspw. mittels entspr. HK-Thermostate oder Raumtemperaturfühler und einer SmartHome-Software), kann es vorteilhaft sein, daraus eine gemittelte Ist- und Soll-Temperatur zu errechnen und an den Wärmerezeuger zu übermitteln. Ein Beispiel hierfür hat FHEM-Forumsuser "freetz" in <a href="#">diesem Thread</a> zur Verfügung gestellt - vielen Dank dafür!

#### Exkurs: Erklärung zum „Raumeinfluss“ bei Berücksichtigung der Raumtemperatur

FHEM-Forumsuser „freetz“ hat die Funktionsweise bzw. das Modell hinter dem „Raumeinfluss“ (Parameter 750) entschlüsselt, so dass die Auswirkungen auf die Vorlauftemperatur verständlicher geworden sind. Vielen Dank dafür!

Sein Beitrag sowie eine Excel-Tabelle zur Berechnung findet sich [hier](#).

Im Folgenden ein Auszug aus seinem Beitrag:

$dTV = dTRw * (1 + s)$  wobei: dTV = resultierende Vorlauftemperaturabweichung dTRw = Raumsollwertkorrektur s = Heizkurvensteilheit (Parameter 720)

Die Raumsollwertkorrektur dTRw berechnet sich wie folgt:  $dTRw = dTR * Raumeinfluss (Parameter 750) / 10$  wobei: dTR = Differenz Raumtemperatur-Ist - Raumtemperatur-Soll

Zusammengeführt lautet die Formel dann:  $dTV = dTR * Raumeinfluss / 10 * (1 + s)$

Bei einer Heizkurve von 1,5 und einer Raumtemperaturabweichung Ist/Soll von 2 Grad und einem Raumtemperatureinfluss von 25% bedeutet das:

$dTV = 2 * 25 / 10 * (1 + 1,5) = 12,5 \text{ °C}$  Vorlauftemperaturveränderung

Bei 4 °C Abweichung (z.B. nach Ende der Nachtabenkung) wäre man dann schon bei 25 Grad höherer VL-Temperatur, was vermutlich mehr ist, als das, was man bei Schnellaufheizung (Parameter 770) hinterlegen würde. Die Therme schaltet darüber hinaus bei Erreichen der Raumtemperaturbegrenzung (Parameter 760) auch bei einem RT-Einfluss von nur 1% ab. Für mich hat das die Konsequenz, dass ich den Einfluss auf max. 20% ansetzen werde. Vielleicht reicht sogar 1%, wenn die Heizkurve als solches gut eingestellt ist und der Einfluss dann nur dafür verwendet wird, bei Erreichen der RT-Begrenzung abzuschalten.

## 6.4 Präsenztaste simulieren

Die Funktion der Präsenztaste, die bei den Raumgeräten vorhanden ist, ist als "Heizbetrieb (temporär)" mit den Spezialparametern

- 10110 = Heizkreis 1
- 10111 = Heizkreis 2
- 10112 = Heizkreis 3 in der reglerspezifischen Datei `BSB_LAN_custom_defs.h` implementiert, der Befehl ist als SET-Befehl auszuführen.

Mit dieser Funktion kann im *Automatikbetrieb* zwischen den Modi Komfort- und Reduziertheizen im Zeitprogramm umgeschaltet werden. Die jeweilige Umschaltung ist dabei so lange gültig, bis gemäß Zeitprogramm (oder durch eine erneute Benutzung der Präsenztaste) die nächste Umschaltung erfolgt.

Bei *aktivem Automatikbetrieb* ist dabei

`http://<IP-Adresse>/S<Parameter>=1` für den Wechsel auf ‚Betriebsart Reduziert‘ und  
`http://<IP-Adresse>/S<Parameter>=2` für den Wechsel auf ‚Betriebsart Komfort‘ zu setzen.

Beispiel:

Der Befehl `<URL>/S10110=2` schaltet innerhalb des Automatikbetriebs den HK1 in den Modus Komfortheizen, mit `<URL>/S10110=1` schaltet man in den Modus Reduziertheizen.

Hinweise
Die genannten Parameter müssen schreibbar sein (s. Kap. 2.2).
Diese Spezialparameter (10110, 10111, 10112) sind NICHT abrufbar.
Die Präsenztaste ist nur im Automatikbetrieb wirksam!
Der jeweilige Wechsel ist bis zur nächsten Umschaltung des Heizbetriebmodus laut Zeitprogramm gültig.
Bei Usern der BSB-LAN-Version v3.x, in deren reglerspezifischen Datei <i>BSB_LAN_custom_defs.h</i> dieser Befehl noch nicht implementiert ist, kann er selbst nachträglich manuell hinzugefügt werden. Eine entspr. Beschreibung der notwendigen Schritte findet sich in Kap. 2.3.

## 6.5 Manuellen TWW-Push ausführen

Bei einigen Reglern ist die (nahezu undokumentierte) Funktion eines manuellen Trinkwasser-Pushs verfügbar. Um einen manuellen TWW-Push auszulösen, muss dazu die TWW-Taste an der ISR-Bedieneinheit gedrückt und für etwa drei Sekunden gehalten werden, bis im Display eine entsprechende Meldung erscheint.

Bei entsprechenden Reglern kann diese Funktion mittels eines SET-Befehls erfolgen: `http://<IP-Adresse>/S10019=1` Der Spezialparameter 10019 muss dazu schreibbar sein (s. Kap. 2.2).

## 6.6 Datum, Uhrzeit und Zeitprogramme verändern

Das Verändern der Uhrzeit und der Zeitprogramme ist nur über einen speziellen URL-Befehl möglich, es ist *nicht* über das Webinterface möglich. Um die Funktion zu nutzen, muss BSB-LAN Schreibzugriff gewährt werden (s. Kap. 2.2).

### Datum und Uhrzeit verändern

Der folgende Befehl stellt das Datum auf den 04.01.2019 und die Uhrzeit auf 20:15 Uhr:

```
/S0=04.01.2019_20:15:00
```

Mit dieser Funktion ist es möglich, die Uhrzeit- und Datumseinstellungen bspw. mit einem NTP Zeitserver abzugleichen.

### Zeitprogramme verändern

Der folgende Befehl setzt das Zeitprogramm für *Mittwoch* beim Heizkreis 1 (Parameter 502) auf 05:00-22:00 Uhr:

```
/S502=05:00-22:00_xx:xx-xx:xx_xx:xx-xx:xx
```

### Zeitprogramm löschen

Um ein Zeitprogramm zu löschen muss man als Uhrzeit „128:00“ eingeben.

Wenn also bspw. für Dienstag drei Zeitprogramme gesetzt wurden, und nun nur das erste von 5-22h behalten möchte, muss folgender Befehl genutzt werden:

```
/S502=05:00-22:00_128:00-128:00_128:00-128:00
```

## 6.7 Übermitteln einer alternativen Außentemperatur

Bei bestimmten Reglermodellen ist es möglich, diverse Funkkomponenten anzuschließen, u.a. auch einen Funk-Außentemperaturfühler. Mittels BSB-LAN ist es bei diesen kompatiblen Reglern möglich, dem Heizungsregler eine anderweitig ermittelte Außentemperatur (AT) zu übermitteln. Dies ist insbesondere für Nutzer komplexerer Hausautomationsinstallationen interessant, die bspw. eine Wetterstation an einem günstigeren Standort als dem des heizungsseitigen Außentemperaturfühlers installiert haben.

Als kompatible Regler sind bisher einige Reglermodelle der Reihen **LMS** und **RVS** gemeldet worden (Stand Oktober 2019). Ältere Reglergenerationen wie bspw. **LMU** oder **RVA** sind anscheinend nicht kompatibel.

Um zu testen, ob der eigene Regler kompatibel ist, kann -zusätzlich neben der Überprüfung des Reglertyps- im Vorfeld `<ip>/Q` oder gezielt ein Abruf der Parameter `<ip>/10003/10004` ausgeführt werden.

Wenn als Rückmeldung bei Parameter 10003 die Außentemperatur (oder "---") angezeigt wird, so ist die Funktion nach bisherigem Kenntnisstand verfügbar.

Wenn hingegen ein "error 7" gemeldet wird, so ist die Funktion leider nicht verfügbar.

Im Zweifelsfall sollte einfach versucht werden, eine alternative AT wie nachfolgend beschrieben zu senden. Ein nachfolgender Abruf des Parameters 8700 gibt Aufschluss darüber, ob der zuvor gesendete Wert übernommen wurde.

Für die Verwendung der Funktion der alternativen Außentemperaturübermittlung mittels BSB-LAN muss der kabelgebundene Außentemperaturfühler der Heizung zwingend vom Regler getrennt werden (da der Regler die alternative AT ansonsten scheinbar nicht annimmt). Die darauf folgende Fehlermeldung des Heizungsreglers "Fehler 10: Aussenfühler" scheint den Betrieb zwar nicht zu stören, kann/sollte aber abgeschaltet werden. Dazu führt man den Parameter 6200 "Fühler speichern" einmal aus (auf JA stellen und bestätigen). Soll der kabelgebundene Fühler irgendwann wieder zum Einsatz kommen, so sollte nach erfolgtem Anschluss erneut Parameter 6200 "Fühler speichern" (-> JA -> bestätigen) ausgeführt werden. Somit ist der kabelgebundene AT-Fühler wieder im Heizungsregler registriert.

Der Funk-Außentemperaturfühler scheint die gemessene AT ca. minütlich zu übermitteln. Bleibt diese Meldung aus, so scheint der Regler nach etwa 10-11 Minuten auf einen intern hinterlegten Wert zurückzugreifen. Zusätzlich erscheint die o.g. Fehlermeldung erneut. Es ist also empfehlenswert, die alternative AT via BSB-LAN etwa alle ein bis zwei Minuten zu übertragen.

Um die Funktion zu nutzen, muss BSB-LAN Schreibzugriff gewährt (s. Kap. 2.2) und die AT mit dem Befehl

`<ip>/I10003=xx`

übermittelt werden, wobei xx die betreffende AT in °C ist. Nachkommawerte sind möglich, als Komma ist ein Punkt einzufügen.

Beispiel:

Mit `<ip>/I10003=16.4` wird dem Heizungsregler die AT von 16.4°C mitgeteilt; `<ip>/I10003=9` übermittelt 9°C AT.

#### Hinweis

Wird nur bei Parameter 10004 die Außentemperatur angezeigt, so ist die Funktion nach bisherigem Kenntnisstand nicht verfügbar. Das Übermitteln der alternativen AT kann in diesem Fall aber trotzdem wie beschrieben getestet werden, allerdings muss dann der Parameter 10004 anstelle von 10003 verwendet werden: `<ip>/10004=xx`.

## 6.8 Eigenen Code in BSB-LAN einbinden

BSB-LAN bietet die Möglichkeit, eigenen Code einzubinden.

Dazu muss das entspr. Definiment in der Datei `BSB_LAN_config.h` aktiviert und der Code entspr. in den Dateien `BSB_LAN_custom.h`, `BSB_LAN_custom_global.h` sowie `BSB_LAN_custom_setup.h` hinzugefügt werden.

Im Ordner **"custom\_functions"** sind Beispiele zu finden.

Bitte lies das entspr. [Readme](#) für weitere Informationen.

FHEM-Forumuser "Scherheinz" hat ein anderes Verwendungsbeispiel zur Verfügung gestellt (siehe [Forumsbeitrag](#)).

Vielen Dank dafür!

Nachfolgend das erwähnte Beispiel:

Beschreibung:

"Alle 20 Sekunden wird über einen Spannungsteiler die Akku Spannung eingelesen. Dann wird aus den letzten 10 Werten ein gleitender Mittelwert ermittelt und per MQTT an FHEM weitergeleitet" (Zitat aus dem oben verlinkten Beitrag).

Einbindung:

Der folgende Code muss in die Datei `BSB_LAN_custom_global.h` eingefügt werden:

```
const int akkuPin = A0;
int akkuWert = 0;
float akkuSpg = 12.00;
char tempBuffer[100];
int j;

void Filtern(float &FiltVal, int NewVal, int FF){ //gleitender Mittelwert bilden aus den 10 letzten Werten
    FiltVal= ((FiltVal * FF) + NewVal) / (FF +1);
}
```

Der folgende Code muss in die Datei `BSB_LAN_custom.h` eingefügt werden:

```

if (custom_timer > custom_timer_compare + 20000) {    // alle 20 Sekunden
    custom_timer_compare = millis();

    akkuWert = analogRead(akkuPin); // Spannung messen

    akkuWert = map(akkuWert, 500, 1023, 0, 150); // umwandeln auf 0 - 15V
    akkuWert = akkuWert / 10.00;

    Filtern(akkuSpg, akkuWert, 9); //gleitender Mittelwert bilden aus den 10 letzten Werten
    if (j++ > 10) akkuWert=1; // nach 10 Werten Sprung auf 1

    if (!MQTTClient.connected()) {
        MQTTClient.setServer(MQTTBroker, 1883);
        int retries = 0;
        while (!MQTTClient.connected() && retries < 3) {
            MQTTClient.connect("BSB-LAN", MQTTUser, MQTTPass);
            retries++;
            if (!MQTTClient.connected()) {
                delay(1000);
                DebugOutput.println(F("Failed to connect to MQTT broker, retrying..."));
            }
            MQTTClient.publish("AkkuSpannung", dtostrf(akkuSpg, 6, 1, tempBuffer));
            MQTTClient.disconnect();
        }
    }
}

```

## 6.9 Verwenden der Webserver-Funktion

**Die Webserver-Funktion wurde von User "dukess" entwickelt, der ebenfalls die nachfolgenden Informationen hinsichtlich der Benutzung zur Verfügung stellte.**

**Vielen Dank!**

Wenn das zugehörige Definement '#define webserver' in *BSB\_LAN\_config.h* aktiviert wurde, kann BSB-LAN als Webserver fungieren, der außerdem statische Komprimierung unterstützt. Um diese Funktion zu verwenden müssen folgende Punkte berücksichtigt werden:

- Alle Dateien werden / müssen auf der microSD-Karte gespeichert sein, können allerdings in verschiedenen Unterverzeichnissen abgelegt werden. Beispiel: `http://<IP-Adresse>/foo/bar.html` liest die Datei `bar.html` aus dem Verzeichnis `foo` von der microSD-Karte.
- Nur statische Inhalte werden unterstützt.
- Unterstützte Dateiformate sind: html, htm, css, js, xml, txt, jpg, gif, svg, png, ico, gz.
- Der Webserver unterstützt die folgenden header: ETag, Last-Modified, Content-Length, Cache-Control.
- Wie bereits erwähnt unterstützt der Webserver statische Komprimierung. Wenn möglich (falls der Browser des Clients gzip unterstützt), werden immer gzip-Inhalte generiert (bspw. `/d3d.js.gz` für die URL `/d3d.js`).

Die folgenden Beispiele verdeutlichen die Benutzung:

- Wenn keine Datei namens `index.html` im Stammverzeichnis der microSD-Karte vorliegt, wird das reguläre Webinterface bei Aufruf von `http://IP-Adresse` dargestellt.
- Wenn eine Datei namens `index.html` im Stammverzeichnis der microSD-Karte vorliegt, wird diese Datei anstelle des regulären Webinterface bei Aufruf von `http://IP-Adresse` dargestellt.
- Wenn die Datei `index.html` in einem Unterverzeichnis auf der microSD-Karte liegt, wird diese nur dann dargestellt, wenn die komplette URL eingegeben und abgerufen wird: `http://<IP-Adresse>/foo/bar/index.html`. Sollte in diesem Fall lediglich `http://<ip-address>/foo/bar/` abgerufen werden, so wird trotzdem das reguläre Webinterface von BSB-LAN dargestellt, da weder eine Verzeichnisauflistung, noch eine URL-Umschreibung in der Webserverfunktion implementiert ist.

### Hinweis

Wenn die optionale PASSKEY-Funktion verwendet wird, muss der PASSKEY wie immer der URL hinzugefügt werden.

## 6.10 Benutzen des alternativen AJAX Webinterface

**Die AJAX Webserverfunktion wurde ebenfalls von User "dukess" entwickelt.**

## 6.11 Raumgerät-Emulation

Mit dem BSB-LAN-Setup lässt sich ein Raumgerät emulieren, dazu ist zusätzliche Hardware erforderlich.

Im Code implementiert sind folgende Funktionen:

- Einbindung angeschlossener Sensoren zur Messung und Übermittlung der Raumtemperatur an den/die gewünschten Heizkreis/e,
- TWW-Push mittels Taster auslösen sowie
- Präsenztastenfunktion für HK1-3 mittels Taster (automatische Erkennung des Ist-Zustandes mit entspr. Umschaltung zwischen Komfort und Reduziert im Automatikmodus).

Um die Funktionen zu nutzen sind die entspr. Einträge in der Konfiguration vorzunehmen, dies kann entweder durch Änderungen in der Datei `BSB_LAN_config.h` oder über das Webinterface (Menüpunkt "Einstellungen") erfolgen.

Nachfolgend einige Hinweise für die jeweiligen Funktionen.

### Raumtemperatur

- Es können bis zu fünf angeschlossene Sensoren für die Raumtemperaturmessungen angegeben werden.
- Kommen mehrere Sensoren zum Einsatz, so wird automatisch ein Mittelwert der Messwerte gebildet und an den Heizungsregler übertragen.
- Um die jeweiligen Sensoren den gewünschten Heizkreisen zuzuordnen, müssen die spezifischen Parameternummern der jeweiligen Sensoren eingetragen werden. Einen Überblick über die angeschlossenen Sensoren samt zugehöriger Parameternummer gibt die Kategorie "One Wire, DHT & MAX! Sensors" (Menüpunkt "Heizungsfunktionen" bzw. direkt per Klick auf den Menüpunkt "Sensoren").
- Bei Eingabe mehrerer Sensoren für einen HK sind die Parameternummern lediglich durch ein Komma von einander zu trennen, es darf kein Leerzeichen nach dem Komma verwendet werden.

### Taster für TWW-Push und Präsenztastenfunktion

- Die verwendeten GPIO-Pins für den Anschluss der Taster (pro Taster ein Pin) sind in der Konfiguration einzustellen.
- Es müssen DIGITALpins genutzt werden!
- Bitte achte darauf, dass du keine anderweitig verwendeten Pins nutzt (bspw. die von angeschlossenen Sensoren)! Für Due-User gilt: explizit *nicht* verwendet werden dürfen die Pins 12, 16-21, 31, 33, 53!
- Die Taster sind mikrocontroller-typisch für HIGH anzuschließen, d.h. du musst zusätzlich zum Taster noch einen PullDown-Widerstand (ca. 100kOhm) für den jeweiligen Pin anschließen.
- Ein Pinout-Diagramm des Due findest du in [Anhang B](#).
- Solltest du dir nicht sicher sein, wie ein Taster generell bei einem Mikrocontroller für HIGH angeschlossen wird, so sieh bitte zusätzlich im Internet nach, dort finden sich unzählige Beispiele.

Trotzdem sei an dieser Stelle kurz erwähnt, wie vorzugehen ist:

- Der Taster mit den beiden Anschlüssen A und B wird an einem Anschluss (A) mit dem gewünschten GPIO-Digitalpin des Due verbunden.
- Zusätzlich wird am selben Anschluss des Tasters (A) der PullDown-Widerstand angeschlossen, welcher wiederum mit GND des Due verbunden wird.  
*Dies ist wichtig, auf den Einsatz des Widerstands darf nicht verzichtet werden!* Durch den PullDown-Widerstand liegt ein definiertes Potential bei nicht-betätigtem Taster am GPIO an und das sog. 'floaten' des Eingangs wird verhindert. Würde man auf den PullDown verzichten und der Eingang würde 'floaten', so könnten ungewollte Leveländerungen am Pin entstehen, die wiederum zur Folge hätten, dass die jeweilige Funktion (also TWW-Push oder BA-Umschaltung) ungewollt ausgelöst wird.
- Der andere Anschluss des Tasters (B) wird an einen **3,3V**-Anschluss des Due angeschlossen.  
**Achtung: Die Eingänge des Due sind nur 3,3V tolerant, verbinde also keinesfalls den Taster mit einem 5V-Anschlusspin des Due!**



Wird der Taster nun betätigt, wird der Stromkreis geschlossen - das Signal wird als HIGH erkannt und der jeweilige Befehl (TWW-Push/Präsenztaste) wird ausgelöst.

- Zusätzlicher Hinweis: Solltest du die Taster entfernen (bspw. weil du die Funktion nicht mehr benutzen willst), so stelle den entspr. Pin in der Konfiguration bitte auf "0" und speichere die neue Einstellung, so dass der zuvor eingestellte Pin nicht 'floaten' kann!

## 6.12 EEPROM-Löschung mittels Pinkontakten

Grundsätzlich kann das EEPROM via Webinterface mit dem Befehl /NE gelöscht werden. Es kann aber in bestimmten Situationen (bspw. wenn kein Zugriff auf das Webinterface möglich ist) nötig sein, das EEPROM auch ohne Nutzung des URL-Befehls zu löschen.

Hierfür müssen

- beim Due die Pins 31 und 33,
- beim ESP32 die Pins 18 und GND,
- beim Olimex die Pins 34 und GND

beim Start oder Reboot kurzzeitig miteinander verbunden werden.

Nach erfolgreichem Löschen blinkt die Arduino-/ESP32-LED vier Sekunden lang.

Beim erneuten Start werden dann die (Vor-)Einstellungen aus der Datei *BSB\_LAN\_config.h* übernommen, eine Anpassung kann danach wie gewohnt bspw. via Webinterface erfolgen.

Die o.g. Pins zum Löschen können bei Bedarf auch individuell in der Datei *BSB\_LAN\_config.h* über das Definiment `#define EEPROM_ERASING_PIN` `xx` gesetzt werden.

Alternativ kann der Microcontroller aber auch über die serielle Schnittstelle angeschlossen werden und über den Serial Monitor die Zeichenkette `/NE` gesendet werden, ggf. mit vorangestelltem Passkey (also dann z.B. `/1234/NE` ). Dann wird ebenfalls das EEPROM gelöscht.

ESP32-Nutzer können außerdem in der ArduinoIDE vor einem erneuten Flashen unter "Werkzeuge" den Eintrag "Erase All Flash Before Sketch Upload" auf "Enabled" stellen, dann wird vor dem Upload der gesamte Flashbereich einmal gelöscht.



[Weiter zu Kapitel 7](#)

[Zurück zum Inhaltsverzeichnis](#)

## 7. BSB-LAN Setup: Optionale Hardware

[Zurück zum Inhaltsverzeichnis](#)

[Zurück zu Kapitel 6](#)

## 7. BSB-LAN Setup: Optionale Hardware

Das BSB-LAN Setup kann durch optionale Hardware in seinem Funktionsumfang erweitert werden. Im Folgenden werden einige Komponenten wie Sensoren und Relais sowie weitere Hardwarelösungen vorgestellt.

*Solltest du ein eigenes, interessantes Projekt umgesetzt haben, was mit dem BSB-LAN Setup zusammenarbeitet und den Funktionsumfang erweitert und möchtest du es gerne auch anderen Usern zur Verfügung stellen, so kontaktiere mich (Ulf) gerne per Email ([adapter@quantentunnel.de](mailto:adapter@quantentunnel.de))!*

### ACHTUNG, wichtiger Hinweis:

Beim Anschließen optionaler Hardware wie bspw. Sensoren, Relais etc. an den Arduino Due bzw. das spezifische ESP32-Board ist unbedingt darauf zu achten, dass **der verwendete Anschlusspin nicht anderweitig belegt ist bzw. nicht bereits boardintern verwendet wird!** Aufschluss hierüber gibt das jeweilige Pinout-Schema des spezifischen Arduino-/ESP-Boards. *Beachte auch die seriellen Pins des Adapters und etwaige weitere Komponenten wie bspw. das LAN-Shield, ein Relais-Shield etc.*

### 7.1 Verwendung optionaler Sensoren: DHT22, DS18B20, BME280

Es besteht die Möglichkeit, zusätzliche Sensoren des Typs

- DHT22 (Temperatur, Luftfeuchtigkeit; Parameternummern 20100-20199),
- DS18B20 (OneWire-Sensor: Temperatur; Parameternummern 20300-20399) sowie
- BME280 (Temperatur, Luftfeuchtigkeit, Luftdruck; Parameternummern 20200-20299) direkt an bestimmte Pins des Adapters bzw. Mikrocontrollers anzuschließen. Die entsprechenden Bibliotheken für die Arduino IDE sind bereits im Softwarepaket des Adapters integriert.

*Solltest du ein ESP32-Board einsetzen, so besteht außerdem die (inoffizielle) Möglichkeit, Xiaomi Bluetooth-Sensoren zu nutzen. Für weitere Informationen diesbzgl. lies bitte [Kap. 7.4.4](#).*

### Hinweise

In der Konfiguration von BSB-LAN ist bei allen Sensoren "Pin 0" voreingestellt. Dies entspricht programmintern der Deaktivierung dieser Funktion und bezeichnet *nicht* den Pin GPIO0! Nach Anschluss eines Sensors muss in der Konfiguration von BSB-LAN der entspr. Pin eingestellt werden - hierfür ist die *GPIO-Pinnummer* einzutragen (bspw. **7** für den Anschluss eines Sensors an GPIO7). Die Lokalisationen und Bezeichnungen der Pins sind dem boardspezifischen Pinout-Schema zu entnehmen.

Der Anschluss der Sensoren kann i.d.R. an GND und +3,3V des Adapters / Mikrocontrollers (ggf. unter zusätzlicher Verwendung der fühlerspezifischen PullUp-Widerstände!) stattfinden.

Zur Nutzung dieser Sensoren muss lediglich die *Konfiguration in der Datei BSB\_LAN\_config.h entsprechend angepasst werden*: Es sind die jeweiligen Definements zu aktivieren und die für DATA genutzten Digitaleingänge bzw. Pins festzulegen (s. hierzu auch [Kap. 2.2](#)).

Auf die Daten der Sensoren kann nach erfolgter Installation über das Webinterface (Button "Sensoren" im oberen Bereich oder Kategorie "One Wire, DHT & MAX! Sensors"), mittels des URL-Befehls mit der entspr. Kategorienummer zugegriffen werden oder auch via direkter sensorspezifischer Parameternummern.

Darüber hinaus werden sie unter `<URL>/ipwe.cgi` standardmäßig mit angezeigt. Voraussetzung hierfür ist jedoch, dass die IPWE-Erweiterung in der Datei *BSB\_LAN\_config.h* durch das entspr. Definement aktiviert wurde (s. [Kap. 2.2](#)).

Sollen die gemessenen Werte geloggt werden oder sind 24h-Mittelwertberechnungen gewünscht, so kann dies mit den jeweiligen Anpassungen in der Datei *BSB\_LAN\_config.h* (s. [Kap. 2.2](#)) ganz einfach realisiert werden.

#### 7.1.1 Hinweise zu DHT22-Temperatur-/Feuchtigkeitssensoren

DHT22-Sensoren werden häufig als „1 wire“ beworben, jedoch handelt es sich hierbei NICHT um den OneWire-Bus von Maxim Integrated oder eine andere Form eines ‚echten‘ Bussystems, bei dem jeder Sensor eine spezifische Adresse aufweist! Die DHT22-Sensoren sind demzufolge auch nicht mit den ‚echten‘ Maxim-OneWire-Sensoren/-Komponenten kompatibel.

Die einzelnen DHT22-Sensoren weisen i.d.R. vier Anschlusspins auf, von denen jedoch der dritte Pin von links (bei Ansicht auf die Vorderseite des Sensors) meistens nicht belegt ist. Im Zweifelsfall sollte dies jedoch nochmal nachgemessen werden! Die Belegung der Pins ist üblicherweise wie folgt:

Pin 1 = VCC (+)

Pin 2 = DATA

Pin 3 = i.d.R. nicht belegt

Pin 4 = GND (-)

Bei Anschluss des Sensors sollte ein PullUp-Widerstand zwischen VCC (Pin 1) und DATA (Pin 2) in der Größe von etwa 4,7kΩ bis 10kΩ hinzugefügt werden. Meist werden 10kΩ empfohlen, die richtige Größe muss im Zweifelsfall ermittelt werden.

**Bitte beachte:**

Kommen mehrere DHT22-Sensoren zum Einsatz, so muss für jeden DATA-Anschluss ein eigener Pin am Mikrocontroller genutzt und in der Datei *BSB\_LAN\_config.h* definiert werden!

Neben den 'nackten' Sensoren gibt es auch noch Ausführungen, die bereits auf einer kleinen Platine angebracht und bei der die drei notwendigen Anschlusspins abgeführt und beschriftet sind. Die folgende Abbildung zeigt ein solches Modell des baugleichen Sensors AM2302.



Die Abfrage der Sensoren/Messwerte kann entweder via direktem Parameterruf (<URL/20100-20199>) oder durch den Aufruf der entspr. Kategorie erfolgen. Der folgende Screenshot zeigt die Webausgabe eines angeschlossenen DHT22-Sensors.

20100 One Wire, DHT & MAX! Sensors - DHT22 Sensor ID #1: 4	4
20100.1 One Wire, DHT & MAX! Sensors - DHT22 Sensor Temperatur #1: 19.90 °C	19.90
20100.2 One Wire, DHT & MAX! Sensors - DHT22 Sensor Luftfeuchtigkeit #1: 65.90 %	65.90
20100.3 One Wire, DHT & MAX! Sensors - DHT22 Sensor Abs Luftfeuchtigkeit #1: 11.30 g/m³	11.30

Darstellung der Messwerte eines DHT22 im Webinterface (Kategorie "One Wire, DHT & MAX! Sensors").

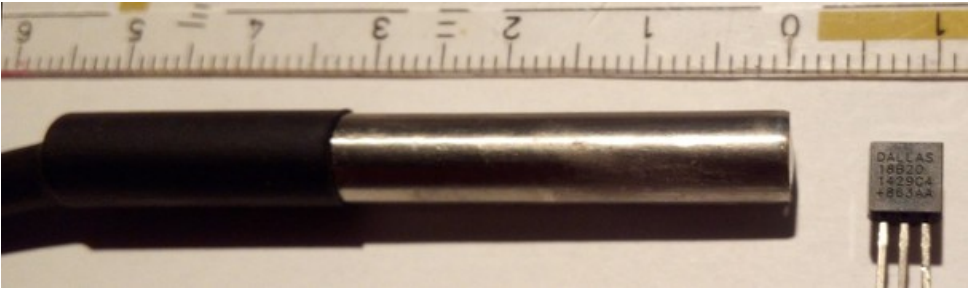
**Tipp:**

Im Internet finden sich zahlreiche Tutorials, Leitfäden und Anwendungsbeispiele für die Anwendung von DHT22-Sensoren.

## 7.1.2 Hinweise zu DS18B20-Tempertursensoren

DS18B20-Sensoren sind 'echte' 1-Wire-/OneWire-Komponenten der Firma Maxim Integrated (ursprünglich Dallas Semiconductor). Jeder Sensor weist eine spezifische interne SensorID auf, die es insbesondere bei größeren Installationen deutlich einfacher macht, einzelne

Sensoren zu identifizieren, sofern man vor der finalen Installation die ID ausgelesen und gut sichtbar auf/an den Sensoren angebracht hat. Neben der üblichen Bauart TO-92 sind die Sensoren auch in wasserdicht gekapselten Ausführungen mit verschiedenen Kabellängen erhältlich.



Die gekapselte Ausführung macht den Einsatz gerade im Bereich der Heizungssteuerung sehr interessant, da hiermit schnell und kostengünstig eine individuelle Installation für diverse Temperaturmessungen realisiert werden kann.

Die Abfrage der Sensoren/Messwerte kann entweder via direktem Parameterruf (`URL/20300-20399`) oder durch den Aufruf der entspr. Kategorie erfolgen. Der folgende Screenshot zeigt die Webausgabe von vier an Pin 7 angeschlossenen DS18B20-Sensoren.

20300.0 One Wire, DHT & MAX! Sensors - DS18B20 Sensor ID #1: 28701A3711220166	28701A3711220166
20300.1 One Wire, DHT & MAX! Sensors - DS18B20 Sensor Temperatur #1: 20.00 °C	20.00
20301.0 One Wire, DHT & MAX! Sensors - DS18B20 Sensor ID #2: 28EAB2311122011C	28EAB2311122011C
20301.1 One Wire, DHT & MAX! Sensors - DS18B20 Sensor Temperatur #2: 19.81 °C	19.81
20302.0 One Wire, DHT & MAX! Sensors - DS18B20 Sensor ID #3: 28CD3233112201E3	28CD3233112201E3
20302.1 One Wire, DHT & MAX! Sensors - DS18B20 Sensor Temperatur #3: 19.81 °C	19.81
20303.0 One Wire, DHT & MAX! Sensors - DS18B20 Sensor ID #4: 2853B935112201F3	2853B935112201F3
20303.1 One Wire, DHT & MAX! Sensors - DS18B20 Sensor Temperatur #4: 19.87 °C	19.87
20304.0 One Wire, DHT & MAX! Sensors - DS18B20 Sensor ID #5: 288B8D2A112201AF	288B8D2A112201AF
20304.1 One Wire, DHT & MAX! Sensors - DS18B20 Sensor Temperatur #5: 19.87 °C	19.87

Darstellung der Messwerte von vier DS18B20 im Webinterface (Kategorie "One Wire, DHT & MAX! Sensors").

**Tipp:**

Werden DS18B20-Sensoren verwendet, so werden in der Kategorie "One Wire, DHT & MAX! Sensors" (und -falls aktiviert- ebenfalls unter `<URL>/ipwe.cgi`) die jeweils **spezifischen internen Hardwarekennungen (SensorID) der DS18B20-Sensoren** aufgeführt. Diese SensorID ist für eine spätere eindeutige Unterscheidung der einzelnen Sensoren notwendig und sollte bspw. bei der weitergehenden Verwendung mit externen Programmen wie FHEM berücksichtigt werden (Stichwort RegEx).

Es ist empfehlenswert, die jeweilige SensorID zu notieren und den entspr. Sensor zu beschriften. Dazu kann ein einzelner Sensor kurz erwärmt oder abgekühlt und durch einen erneuten Aufruf der Sensor-Kategorie anhand der Temperaturschwankung identifiziert werden.

Werden Sensoren ausgetauscht, hinzugefügt oder entfernt, so ändert sich meist auch die Reihenfolge, in der sie in der entspr. Kategorie angezeigt werden (da diese auf der SensorID basiert). Wird das Reading also nicht auf die individuelle SensorID ausgelegt, sondern lediglich auf die Bezeichnung "temp[x]" wie sie in der entspr. Kategorie angezeigt werden, so kommt es früher oder später dazu, dass die entsprechend gemachten Zuordnungen (bspw. VL, RL, Puffer) nicht mehr übereinstimmen.

**Hinweis:**

Werden Änderungen an der Sensorinstallation vorgenommen (Austausch, Hinzufügen, Entfernen), so muss der Mikrocontroller neu gestartet werden, damit die Sensoren initial neu eingelesen werden.

**Tipps für die elektrische Installation:**

Die einzelnen Sensoren weisen i.d.R. drei Pins auf: VCC, DATA und GND.  
Bei den gekapselten Versionen ist die Farbwahl der bereits angeschlossenen Kabel meist wie folgt:  
Rot = VCC (+3,3V)  
Gelb = DATA  
Schwarz = GND (-)

Kommen mehrere DS18B20-Sensoren und/oder größere Leitungslängen zum Einsatz, hat es sich bewährt, pro Sensor je einen 100nF-Keramikkondensator (und ggf. noch einen 10µF-Tantalkondensator zusätzlich) möglichst nah am Sensor in die Leitung zwischen GND und VCC zu positionieren, um einen Spannungsabfall bei der Abfrage zu kompensieren.

#### Anmerkungen:

Kommen die üblichen gekapselten und bereits verkabelten Sensoren zum Einsatz, so reicht es i.d.R. aus, den Kondensator dort anzuschließen, wo auch die Kabel angeschlossen werden - ein Auftrennen des Kabels nahe des Sensors ist -zumindest bei den Versionen mit 1m und 3m Kabellängen- erfahrungsgemäß nicht nötig.

Im Gegensatz zu Keramikkondensatoren ist bei der (zusätzlichen) Verwendung von Tantalkondensatoren auf die Polarität zu achten!

Der Wert des PullUp-Widerstandes am Adapterausgang zwischen DATA und VCC (+3,3V) ist (insbesondere bei großen Leitungslängen und/oder mehreren Sensoren) für einen problemlosen Betrieb u.U. kleiner als die üblicherweise empfohlenen 4,7kΩ zu wählen.

Darüber hinaus scheint es bei komplexeren bzw. größeren Installationen in Einzelfällen so zu sein, dass die Spannungsversorgung mit den 3,3V des Due nicht immer einen problemlosen Betrieb der Sensoren ermöglicht. Da diese OneWire-Sensoren "open drain" sind, können sie auch mit 5V des Due betrieben werden, was einen stabileren Betrieb zur Folge zu haben scheint. Es muss dann allerdings darauf geachtet werden, dass die 5V *nie* an dem GPIO des Due anliegen!

*Für die Installation heißt das konkret, dass VCC der Sensoren am 5V-Pin des Due angeschlossen wird, der zu verwendende PullUp-Widerstand dann jedoch zwischen DATA und einem 3,3V-Pin des Due platziert werden muss!*

Von der Verwendung des sogenannten 'parasitären Modus' ist abzuraten.

Die Verwendung einer geschirmten Steuerleitung ist zu empfehlen. Die Schirmung sollte dabei einseitig an Masse (GND) angeschlossen werden. Um etwaige von der Versorgungsspannung des Mikrocontroller-Netzteils ausgehende Störeinflüsse zu minimieren, kann die Zuleitung der Stromversorgung mikrocontrollerseitig etwa vier bis fünfmal durch einen Ferritring geführt werden.

Kommen *große* Kabellängen zum Einsatz, so ist insbesondere auf eine korrekte Netzwerktopologie zu achten. Hier ist die Lektüre des vom Hersteller herausgegebenen Tutorials "[Guidelines for Reliable Long Line 1-Wire Networks](#)" zu empfehlen.

In diesem Fall sind außerdem weitere Dinge zu beachten, wie bspw. eine empfehlenswerte Hin- und Rückleitung für den Datenkanal, der möglicherweise notwendige Einsatz von zusätzlichen Spannungsquellen, die Verwendung eines dedizierten Busmasters etc.

Als vereinfachte Faustregel kann man sagen, je größer die Leitungslängen und je komplexer die DS18B20-Installationen ausfallen, desto kritischer ist die vorhergehende Planung zu betrachten.

#### Tipp:

Im Internet finden sich zahlreiche Tutorials, Leitfäden und Anwendungsbeispiele zum Thema 1-Wire/OneWire/DS18B20.

#### **Zusammenfassung benötigter Bauteile für eine Installation:**

- Dreiadriges Kabel, idealerweise geschirmt (Schirmung ist dann einseitig an GND anzuschließen)
- PullUp-Widerstand 4,7kΩ oder ggf. kleiner, nur einer notwendig, adapter-/mikrocontrollerseitig zwischen VCC und DATA positionieren
- Keramikkondensator 100nF, pro Sensor einer, zwischen VCC und GND nahe am Sensor positionieren
- optional: Tantalkondensator 10μF, pro Sensor einer (zusätzlich zum Keramikkondensator!), zwischen VCC und GND nahe am Sensor positionieren (bei Tantalkondensatoren bitte die Polarität beachten!)
- optional: Schraublemmen o.ä., Streifen-/Lochrasterplatte, Gehäuse, ...

#### **Tipps für die Verwendung im Bereich der Heizungsinstallation:**

- Werden die gekapselten und bereits mit einem Kabel versehenen Sensoren eingesetzt, so kann es sich bei größeren und verzweigteren Heizungsanlagen lohnen, die Versionen mit 3m anstatt 1m Kabellänge zu nehmen. Sie kosten zwar etwas mehr, bieten jedoch deutlich mehr Spielraum und Bewegungsfreiheit bei der Platzierung der Sensoren.
- Sollen die Sensoren für Temperaturmessungen an Rohren zum Einsatz kommen (bspw. HK-VL/-RL), so ist es empfehlenswert, ein Bett aus Wärmeleitpaste für den Kontaktbereich zu verwenden.
- Darüber hinaus haben Tests gezeigt, dass die Positionierung nach einer Rohrbiegung oder eines 90°-"Knicks" (in Strömungsrichtung) an der 'Außenseite' des "Knicks" ideal zu sein scheint, da hier die Kerntemperatur des Strömungsmediums aufgrund der auftretenden Verwirbelungen nah an die Rohrwand gelangt.
- Die Metallhülse der gekapselten Bauform sollte möglichst mit einer metallenen Rohrschelle am Rohr fixiert werden.
- Das Kabel selbst sollte zusätzlich mit einem Kabelbinder fixiert werden, um Zugkräfte an der Fühlerhülse sowie ein Verrutschen des Fühlers zu vermeiden.



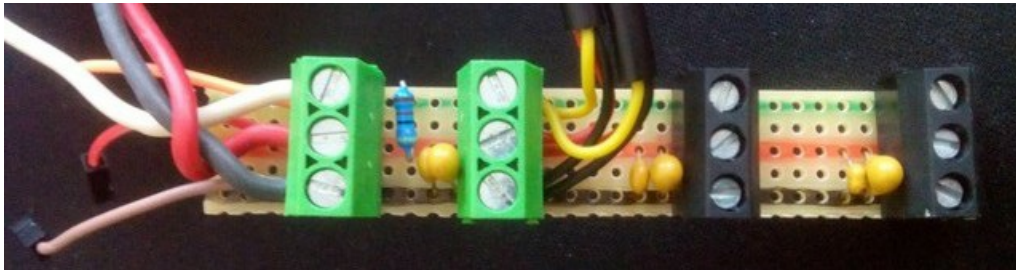
- Die Rohrdämmung sollte nach Anbringen des Fühlers (unter der Dämmung) wieder gewissenhaft verschlossen werden. Löcher, Einschnitte o.ä. in Fühlernähe sind zu vermeiden. Werden Fühler an bisher ungedämmten Rohren montiert, so ist zumindest für den Bereich des Fühlers eine zusätzliche Rohrisolierung empfehlenswert, um Messwertverfälschungen durch bspw. Raum- oder Zugluft zu vermeiden.
- Kommen die Fühler in Tauchhülsen oder Klemmschienen zum Einsatz, ist ggf. auch hier die Verwendung von zusätzlicher Wärmeleitpaste zu empfehlen.
- Im Allgemeinen sollten die Fühler etwa ein bis zwei Meter von einer zusätzlichen Wärmequelle (wie bspw. Heizkessel, Pufferspeicher o.ä.) entfernt montiert werden.

**Bitte beachte:**

**Bereits installierte Fühler (bspw. in Tauchhülsen von Mischern, Pufferspeichern etc.), die an einen Heizungs- oder Solarregler angeschlossen sind, haben immer Vorrang! Keinesfalls sollte deren Installation oder der Kontakt mit dem zu messenden Element durch eine zusätzliche Montage von DS18B20-Sensoren leiden!**

**Bauvorschlag:**

Bei kleineren DS18B20-Installationen im Heizungsbereich mit übersichtlichen Kabellängen kann man sich einen kleinen 'Verteilerkasten' bauen. Dazu kann man die gekapselten Sensoren nacheinander samt vorgeschalteter Kondensatoren auf einer Streifenplatine anschließen. Lötet man die Kabel der Sensoren nicht an, sondern verwendet statt dessen kleine Schraubklemmen, so kann man im Bedarfsfall problemlos einzelne Sensoren austauschen oder auch das System erweitern. Am Anfang dieser Verteilerplatine wird das Kabel angeschlossen, was zum BSB-LAN-Adapter bzw. zum Mikrocontroller geführt wird. Wenn die Optik nicht stört, kann das gesamte Konstrukt kostengünstig in einer Feuchtraum-AP-Verteilerdose untergebracht werden.



### 7.1.3 Hinweise zu BME280-Sensoren

Sensoren des Typs BME280 bieten drei (bzw. fünf) Messgrößen: Temperatur, Luftfeuchtigkeit (zzgl. der errechneten absoluten Luftfeuchtigkeit) sowie Luftdruck (zzgl. der errechneten Höhe). Sie sind klein, i.d.R. unkompliziert anzuschließen und bieten (ausreichend) exakte Messergebnisse.

**Am I2C-Bus des Mikrocontrollers (ebenfalls am veralteten Arduino Mega 2560) können bis zu zwei Sensoren des Typs BME280**

angeschlossen werden.

Zur Verwendung muss das entspr. Definement in der Datei `BSB_LAN_config.h` oder via Webconfig aktiviert und die Anzahl der angeschlossenen Sensoren festgelegt werden (s. [Kap. 2.2.2](#)).

Hinweise
Prinzipiell können BME280 auch an einem SPI angeschlossen werden, jedoch <b>nicht</b> am Mikrocontroller unseres BSB-LAN-Setups!
Wenn mehr als zwei BME280-Sensoren benötigt werden, können diese mittels eines I2C-Multiplexers TCA9548A angeschlossen werden.
Die Verwendung eines BMP280 ist ebenfalls möglich, dieser bietet allerdings keine Feuchtigkeitsmessung. Daher ist der Einsatz eines BME280 zu empfehlen.



Ein BME280-Sensor auf einem typischen Breakout-Board (Clone); links = Vorderseite, rechts = Rückseite.

Folgende Punkte sind dabei zu beachten:

- Stelle sicher, dass es sich um einen Sensor des Typs BME280 handelt (und nicht um bspw. einen BMP180 o.ä.).
- Stelle sicher, dass du möglichst eine Variante verwendest, die auf dem Breakout-Board bereits PullUp-Widerstände verbaut hat (wie auf dem oben gezeigten Bild). Sollte deine Variante *keine* PullUp-Widerstände verbaut haben, so musst du diese beim Anschluss an den Mikrocontroller noch hinzufügen (ca. 10kOhm, zwischen SDA und 3,3V sowie zwischen SCL und 3,3V anschließen)!
- Stelle sicher, dass der erste Sensor die I2C-Adresse 0x76 hat! Dies ist bei dem oben gezeigten Modul üblicherweise der Fall.
- Der zweite Sensor muss die Adresse 0x77 erhalten. Wie dies bei dem oben gezeigten Modul zu erreichen ist, wird nachfolgend beschrieben.
- Stelle sicher, dass du den Sensor an den 3,3V-Pin des Mikrocontrollers anschließt! Das oben gezeigte Modul hat zwar einen Spannungsregler und Levelshifter verbaut, so dass in dem Fall prinzipiell auch ein Anschluss an 5V erfolgen *könnte* - um aber sicherzustellen, dass keinesfalls 5V an SDA/SCL anliegen, solltest du stets den Anschluss an 3,3V vorziehen.

## Anschluss

Der Sensor bzw. das Breakout-Board ist i.d.R. bereits eindeutig beschriftet, so dass die Anschlüsse hier klar identifiziert werden können. Je nach verwendetem Arduino muss ein anderer I2C-Anschluss verwendet werden:

- Der **Due** weist zwei I2C-Busanschlüsse auf: SDA/SCL an den Pins 20/21 sowie SDA1/SCL1. Es ist darauf zu achten, dass die Anschlüsse **SDA1 & SCL1** verwendet werden, da der BSB-LAN-Adapter bereits die Anschlüsse SDA/SCL verwendet. SDA1/SCL1 befinden sich neben dem "AREF"-Pin. Sie werden i.d.R. vom LAN-Shield verdeckt und sind nicht nach oben zum/durchs LAN-Shield durchgeführt. Sie sind jedoch unterhalb des LAN-Shields direkt auf dem Due zugänglich. Für eine genaue Positionsbestimmung von SDA1/SCL1 sieh dir bitte das [Pinoutdiagramm in Anhang B](#) an.
- Der **Mega 2560** weist hingegen nur einen I2C-Busanschluss auf: SDA/SCL an den Pins 20/21. Dieser wird vom alten Adapter v2 nicht belegt, der Anschluss kann für den BME280 verwendet werden.

Die Verkabelung ist wie folgt vorzunehmen:

BME280	DUE	Mega2560
VIN	3,3V	3,3V
GND	GND	GND
SDA	SDA1	SDA 20
SCL	SCL1	SCL 21

### Adressierung

Die üblichen Breakout-Boards wie das oben gezeigte BME280-Modul weisen auf der Vorderseite unterhalb des eigentlichen Sensors drei Lötunkte (oder Lötfelder) auf, bei denen üblicherweise der *linke* und der mittlere Lötunkt durch eine Leiterbahn miteinander verbunden sind. Dies entspricht i.d.R. der Adresse 0x76. Das nachfolgende Bild zeigt gelb eingekreist diese Verbindung.



Adresse 0x76: Leiterbahn zwischen dem linken und dem mittleren Lötunkt.

Soll nun ein zweiter Sensor parallel dazu angeschlossen werden, so ist bei dem zweiten Modul diese Leiterbahn vorsichtig(!) und gewissenhaft mit einem feinen scharfen Gegenstand (bspw. Cutter, Skalpell) zu durchtrennen. Danach müssen der *rechte* und der mittlere Pin durch etwas Lötzinn miteinander verbunden werden. Das nachfolgende Bild zeigt skizzenhaft die notwendigen Schritte: Die rote Linie links kennzeichnet den notwendigen 'Schnitt' auf der Platine, die grüne Linie rechts kennzeichnet die danach vorzunehmende Verbindung mittels Lötzinn.





Adresse 0x77: Die rote Linie markiert die durchgetrennte Leiterbahn, die grüne Linie markiert die neu herzustellende Verbindung.

Auslesen

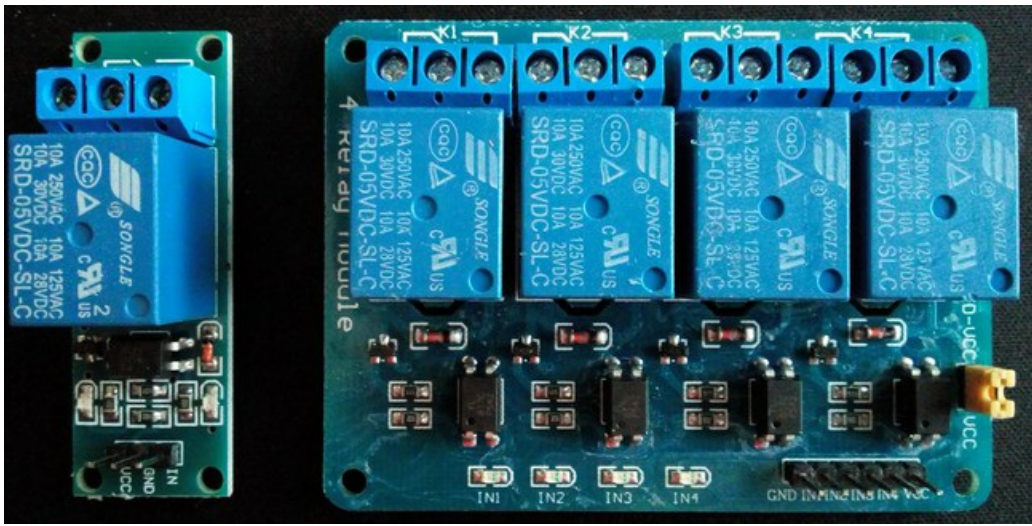
Die Messwerte des/der angeschlossenen BME280 können wie üblich ausgelesen werden, bspw. durch Aufrufen der Kategorie "One Wire, DHT & MAX! Sensors" unter "Heizungsfunktionen", durch einen direkten Klick auf den Button "Sensoren" oder auch durch die Eingabe der spezifischen Parameternummern ( [URL/20200-20299](#) ). Soll ein Loggen, eine Anzeige innerhalb der IPWE-Erweiterung etc. erfolgen, sind die spezifischen Parameternummern der gewünschten Messwerte des jeweiligen Sensors anzugeben.  
Der nachfolgende Screenshot zeigt die entspr. Darstellung eines BME280 innerhalb der Kategorie "One Wire, DHT & MAX! Sensors".

20200.0 One Wire, DHT & MAX! Sensors - BME280 Sensor ID #1: 77	77
20200.1 One Wire, DHT & MAX! Sensors - BME280 Sensor Temperatur #1: 22.82 °C	22.82
20200.2 One Wire, DHT & MAX! Sensors - BME280 Sensor Luftfeuchtigkeit #1: 52.31 %	52.31
20200.3 One Wire, DHT & MAX! Sensors - BME280 Sensor Pressure #1: 1004.47 hPa	1004.47
20200.4 One Wire, DHT & MAX! Sensors - BME280 Sensor Altitude #1: 71.27 m	71.27

Darstellung der Messwerte eines BME280 im Webinterface (Kategorie "One Wire, DHT & MAX! Sensors").

## 7.2 Relais und Relaisboards

Prinzipiell ist es möglich und in der BSB-LAN-Software als Funktion mit den Varianten des [URL-Befehls](#) [/G](#) auch bereits vorgesehen, dass am Arduino/ESP32 zusätzliche Relais oder Relaisboards angeschlossen und mit BSB-LAN gesteuert werden können. Auf diese Weise können nicht nur Verbraucher geschaltet, sondern auch Zustände angeschlossener Verbraucher abgefragt werden.



Ein einzelnes und ein 4-Kanal Relaismodul für den Einsatz an einem Mikrocontroller.

Die oftmals günstig erhältlichen Relaisboards sind dabei bereits mit Relais bestückt, die 230V-Verbraucher direkt schalten können. Leider kann es aufgrund mangelhafter Qualität oder Überlastung zu diversen Schäden und damit einhergehenden größeren Risiken wie bspw. Bränden kommen. Daher ist die zusätzliche Verwendung von entsprechend dimensionierten Koppelrelais oder Solid-State-Relais überlegenswert. Sollten diese jedoch ausschließlich zum Einsatz kommen und mit ihnen Schaltvorgänge ausgelöst werden, so ist ggf. darauf zu achten, dass Strom- und Spannungsstärke des Mikrocontrollers ausreichend sind, um den Schaltvorgang des Relais auszulösen.

**ACHTUNG**

*Es sollte beachtet werden, dass jegliche Installationen und Arbeiten am 230V-Netz nur von zugelassenen Elektrikern vorgenommen werden dürfen! 230V können tödlich sein! Es ist empfehlenswert, einen Elektriker bereits bei der Planung des Vorhabens mit einzubeziehen.*

*Vor der Verwendung eines Relais/Relaisboards sollte sichergestellt werden, dass es für die gewünschte Aufgabe geeignet ist! Bei den schaltbaren multifunktionalen Eingängen der Heizungsregler wird bspw. häufig gefordert, dass das Relais "kleinspannungsgerecht" ist - dieses Kriterium erfüllen nicht alle Relais!*

*Es ist NICHT möglich, den Mikrocontroller direkt an die multifunktionalen Eingänge des Heizungsreglers anzuschließen!*



Ein übliches Koppelrelais. Die entsprechenden Pins am Mikrocontroller werden bei diesem Modell an "14" und "13" angeschlossen.

**Beispiel**

Mittels eines parallel zur Umwälzpumpe einer Solarthermieranlage angeschlossenen Koppelrelais (sofern deren Regelung nicht mit dem Heizungsregler verbunden oder bei diesem integriert ist), wäre es bspw. möglich, den Zustand des mikrocontrollerseitigen Kontaktes (offen/geschlossen) und somit den Betriebsstatus der Pumpe abzufragen.



## 7.3 MAX!-Komponenten

BSB-LAN ist bereits für die Einbindung und Nutzung von MAX!-Komponenten vorbereitet. MAX-Thermostate, die von BSB-LAN verwendet werden sollen, müssen anhand der aufgedruckten Seriennummer in der Datei *BSB\_LAN\_config.h* in das Array `max_device_list[]` eingetragen werden. Nach dem Start von BSB-LAN muss dann an diesen Thermostaten die Pairing-Taste gedrückt werden, um die Verbindung zwischen BSB-LAN und den Thermostaten herzustellen.

In der Datei *BSB\_LAN\_custom.h* werden für die MAX!-Einbindung folgende Variablen bereit gestellt:

- `custom_timer`  
Diese Variable wird bei jedem Durchlauf der loop Funktion des Hauptprogramms auf den Wert von `millis()` gesetzt.
- `custom_timer_compare`  
Diese Variable kann verwendet werden, um im Zusammenhang mit `custom_timer` zeitabhängige Ausführungen zu realisieren, z.B., um eine Aufgabe (Werte abrufen, vergleichen etc.) periodisch auszuführen.

Darüber hinaus stehen alle globalen Variablen aus der Datei *BSB\_LAN.ino* zur Verfügung. Hinsichtlich der MAX!-Funktionalität sind das insbesondere:

- `max_devices[]`  
Dieses Array enthält die DeviceID des jeweiligen MAX!-Gerätes, das sich angemeldet hat. Hiermit kann man ggf. bei Berechnungen bestimmte Thermostate ausblenden.
- `max_cur_temp[]`  
Dieses Array enthält die aktuell gemessene Temperatur des Thermostats. Sinnvoll für Berechnungen sind bei MAX!-Geräten nur die Wandthermostate, weil diese kontinuierlich die Temperatur übermitteln. Heizkörperthermostate tun dies nur bei Ventiländerungen oder Schaltzeitwechseln.
- `max_dst_temp[]`  
Dieses Array enthält die Soll-Temperatur des Thermostats.
- `max_valve[]`  
Dieses Array enthält die momentane Ventilöffnung des Heizkörperthermostats (bei Wandthermostaten nur verfügbar, wenn diese entsprechend mit einem Heizkörperthermostat gekoppelt sind).

Die Reihenfolge zwischen den Arrays ist immer gleich, d.h., wenn `max_devices[3]` der Wandthermostat im Wohnzimmer mit ID xyz ist, dann ist `max_cur_temp[3]` die momentane Temperatur im Wohnzimmer und `max_dst_temp[3]` die entsprechende Solltemperatur usw.

Die Reihenfolge innerhalb `max_devices[]` richtet sich danach, wie sich diese angemeldet haben, bleibt dann aber auch über Neustarts hinweg konstant, da diese im EEPROM abgespeichert werden (bis diese mit `http://<IP-Adresse>/NE` gelöscht werden). Dennoch sollte man sich nicht darauf verlassen, sondern im Zweifelsfall, z.B. beim Ausklammern von bestimmten Thermostaten, immer mit der in `max_device[]` hinterlegten ID vergleichen. Diese kann man der zweiten Spalte der Auflistung unter `http://<IP-Adresse>/X` oder der Ausgabe der Kategorie "Sensoren" entnehmen und ist nicht identisch mit der auf den Geräten aufgedruckten ID.

20500 One Wire, DHT & MAX! Sensors - MAX! Sensor ID #1: KEQ0502326	KEQ0502326
20501 One Wire, DHT & MAX! Sensors - MAX! Sensor Ist Temperatur #1: 21.20 °C	21.20
20502 One Wire, DHT & MAX! Sensors - MAX! Sensor Soll-Temperatur #1: 21.00 °C	21.00
20503 One Wire, DHT & MAX! Sensors - MAX! Sensor Ventilöffnung #1: --- % query failed	---
20504 One Wire, DHT & MAX! Sensors - MAX! Sensor ID #2: KEQ0505080	KEQ0505080
20505 One Wire, DHT & MAX! Sensors - MAX! Sensor Ist Temperatur #2: 21.50 °C	21.50
20506 One Wire, DHT & MAX! Sensors - MAX! Sensor Soll-Temperatur #2: 21.00 °C	21.00
20507 One Wire, DHT & MAX! Sensors - MAX! Sensor Ventilöffnung #2: 0 %	0

Darstellung der MAX!-Sensoren in der Kategorie "One Wire, DHT & MAX! Sensors".

Wichtiger Hinweis für diejenigen, die die MAX!-Thermostate über einen zum CUL/CUNO geflashten Max!Cube (Informationen diesbzgl. s. [hier](#)) verwenden:  
Wenn bei der Einrichtung des CUNO BSB-LAN nicht lief (oder anderweitig beschäftigt war), muss an den betreffenden Geräten nochmals die Pairing-Taste gedrückt werden. Denn nur bei *diesem* Pairing-Prozess wird die auf den Geräten aufgedruckte Seriennummer zusammen mit der sonst intern verwendeten ID (die auch u.a. auch FHEM verwendet) übermittelt und BSB-LAN kann die entsprechende Zuordnung vornehmen. Ansonsten weiß BSB-LAN bei den anderen Telegrammen des Cube nämlich nicht, um welche MAX!-Geräte es geht.

Wird im weiteren Verlauf bspw. mittels FHEM (Hinweise zur Konfiguration des MAX-Moduls unter FHEM siehe [hier](#)) die jeweilige Temperatur

mehrerer Wand- und Heizkörperthermostate erfasst, so lässt sich daraus eine gemittelte Ist- und Soll-Temperatur bilden. Diese kann dann dem Heizungsregler übermittelt werden, um den Wärmeerzeuger bedarfsgerechter zu steuern. Eine solche Lösung lässt sich [hier](#) nachlesen. FHEM-Forumsmitglied „Andreas29“ hat dieses Anwendungsbeispiel ohne FHEM umgesetzt. Eine ausführliche Beschreibung samt der benötigten angepassten Datei `BSB_LAN_custom.h` findet sich [hier](#). Das in dem Zusammenhang dort erwähnte und verwendete „Arduino-Raumgerät light“ ist in Kap. [7.4.2](#) vorgestellt.

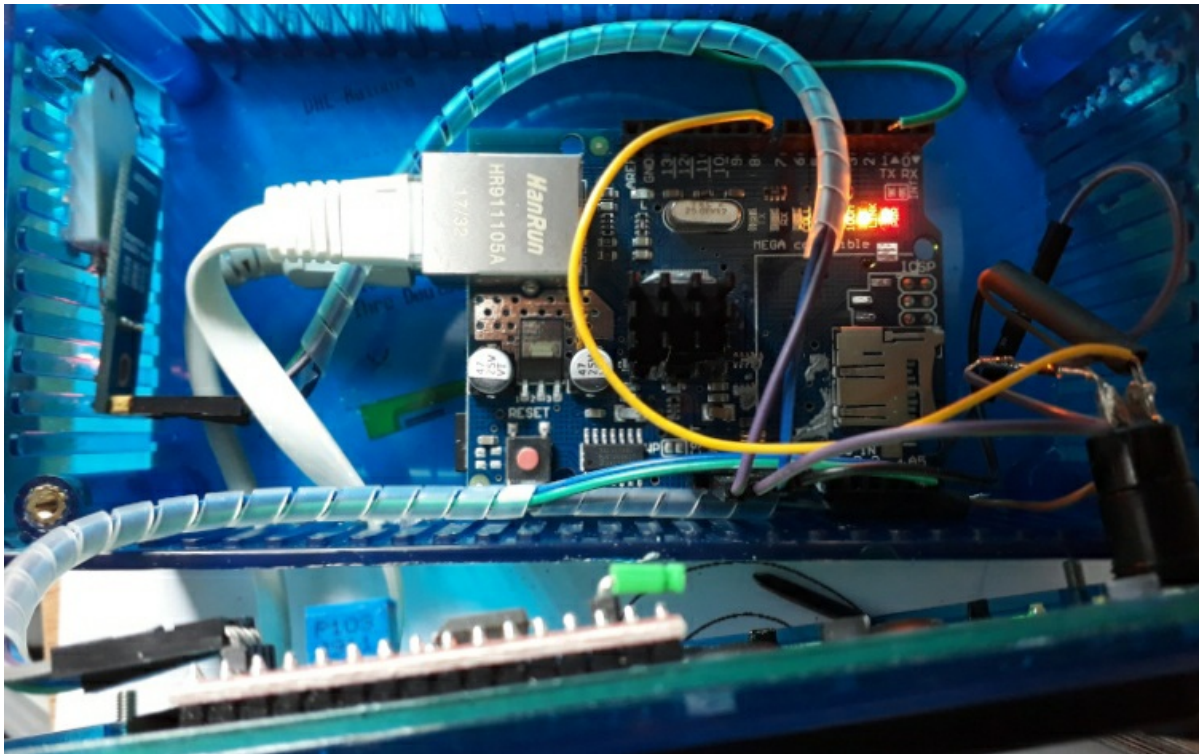
## 7.4 Eigene Hardwarelösungen

Im Folgenden werden Lösungen von Nutzern vorgestellt, die nicht nur zum Nachbau anregen, sondern weitere Nutzungsmöglichkeiten von BSB-LAN aufzeigen und als Inspiration für eigene Projekte dienen sollen.

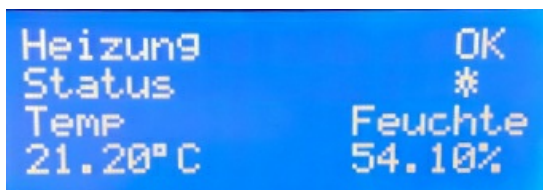
Wenn du ein eigenes interessantes Projekt erfolgreich umgesetzt hast, was auch für andere Nutzer hilfreich sein könnte, so würde ich mich freuen, wenn du dich mit mir in Verbindung setzt. Eventuell kann auch dein Beispiel an dieser Stelle mit aufgeführt werden. Schicke mir dazu gerne eine Email an [adapter \[ät\] quantentunnel.de](mailto:adapter@quantentunnel.de). Vielen Dank!

### 7.4.1 Raumgeräteersatz (Arduino Uno, LAN-Shield, DHT22, Display, Taster)

FHEM-Forumsmitglied „Andreas29“ hat basierend auf einem Arduino Uno einen Raumgeräteersatz realisiert. Der jeweilige Betriebs- und Fehlerstatus des Wärmeerzeugers sowie die aktuellen Daten eines DHT22-Sensors werden auf einem 4x20-LCD dargestellt. Mittels eines Tasters wird die Funktion der Präsenztaste eines echten Raumgerätes nachgebildet.



Das Innenleben des Raumgeräteersatzes.



Das Display des Raumgeräteersatzes.

Eine ausführliche Beschreibung samt Schaltplan und Software ist [hier](#) zu finden.

Andreas29 hat den Funktionsumfang um Push-Benachrichtigung im Fehlerfall (Erreichbarkeit der Heizung, MAX!-Fehler) erweitert, die entspr. Beschreibung sowie die Software sind [hier](#) zu finden.

## 7.4.2 Raumtemperaturfühler (Wemos D1 mini, DHT22, Display)

FHEM-Forumsmitglied „Gizmo\_the\_great“ hat basierend auf einem Wemos D1 mini und einem DHT22-Fühler einen Raumfühler realisiert. Die aktuellen Temperaturen von HK1 und HK2 werden dabei auf einem OLED-Display angezeigt. Auf dem Wemos D1 läuft ESPEasy.

Eine genauere Beschreibung des Projekts „Raumfühler mit OLED“ ist [hier](#) zu finden.

## 7.4.3 Raumgeräteersatz mit UDP-Kommunikation

### 7.4.3.1 UDP mit Arduino Uno + LAN-Shield

FHEM-Forumsmitglied "fabulous" hat in Anlehnung auf die oben genannte Variante von User "Andreas29" einen Raumgeräteersatz realisiert, der mit dem BSB-LAN-Adapter via UDP kommuniziert. Zur Verwendung kommen dabei ein Arduino Uno samt LAN-Shield, ein 20x4 LCD sowie ein Taster. Eine genaue Beschreibung sowie der entspr. Code ist [hier](#) zu finden.

### 7.4.3.2 UDP mit ESP32

BSB-LAN-User "-cr" hat die o.g. Variante von User "fabulous" erweitert und auf einen ESP32 samt ssd1306-Display angepasst. Sein Projekt [BSBmonCR](#) ermöglicht u.a. eine grafische Darstellung ausgewählter Parameter im zeitlichen Verlauf sowie eine Präsenzerkennung. Darüber hinaus kann sogar auf ein Display verzichtet werden, da die grafische Darstellung auch über http abrufbar ist und außerdem das Loggen in einen Dropbox-Account möglich ist.



Das fertige Setup samt Gehäuse.

☀️-12  
🏠45.6  
💧78.9

Grafische Darstellung von drei Parametern im zeitlichen Verlauf.

Die ausführliche Beschreibung ist in [seinem GitHub-Repository](#) zu finden.

## 7.4.4 Xiaomi Mijia BLE Sensoren

Achtung

## Achtung

**Die im Folgenden vorgestellte Lösung ist keine 'offizielle' BSB-LAN-Version!**

Daher sind bestimmte Funktionen nicht verfügbar und wir können keinen Support dafür bieten!

Sollten jedoch Fragen oder Probleme bzgl. der Verwendung auftauchen, so kannst du sie jedoch [in diesem Diskussionsthread](#) (bitte möglichst auf Englisch) stellen.

Die nachfolgend beschriebene Möglichkeit, Xiaomi Bluetooth-Sensoren einzubinden, betrifft **ausschließlich die Verwendung von ESP32-Boards!**

**User DukeSS hat die Unterstützung für BLE (bluetooth low energy) Sensoren entwickelt und stellt dies in einer speziellen BSB-LAN-Version in [seinem GitHub repository](#) zur Verfügung.**

**Vielen Dank!**

Wenn du ein ESP32-Board verwendest, kannst du eine [alternative Version von BSB-LAN](#) nutzen, welche Unterstützung für BLE-Sensoren bietet. Mit dieser speziellen Version ist es möglich, verschiedene BLE-Sensoren in BSB-LAN einzubinden. [Hier](#) findest du eine Liste der unterstützten Sensoren.

Diese Lösung wurde mit *Xiaomi Mijia BLE Sensoren des Typs LYWSD03MMC* getestet.

Derzeit werden nur unverschlüsselte Nachrichten unterstützt, man muss für die jeweiligen Sensoren also eine alternative Firmware verwenden. Für die erwähnten Xiaomi Mijia BLE Sensoren des Typs LYWSD03MMC kannst du eine solche Firmware [hier](#) finden.

Die Einschränkungen bei dieser Lösung bestehen derzeit darin, dass bspw. die OTA-Funktionalität nicht funktioniert, weil die BLE-Implementierung zu viel Speicherplatz benötigt.

Um die Funktion zu nutzen, müssen zwei Einstellungen in dieser speziellen BSB-LAN-Version angepasst werden:

- Aktiviere `EnableBLE` um den BLE-Scan zu aktivieren.
- Füge die MAC-Adressen der gewünschten BLE-Sensoren bei `BLE_sensors_macs` hinzu.  
Geräte, die hier nicht aufgeführt sind, werden ignoriert. Die Reihenfolge in der Auflistung beeinflusst die spätere Reihenfolge in der Kategoriedarstellung.  
Derzeit können bis zu 40 Sensoren gelistet werden (Parameternummern: NN20900-20199).  
Eine Auflistung aller gefundenen Sensoren kann mittels des URL-Befehls `/CO` erfolgen (wenn `EnableBLE` aktiviert ist).

## 7.5 LAN-Optionen für das BSB-LAN-Setup

Obwohl für die Netzwerkanbindung des Adapters definitiv die kabelgebundene Variante zu empfehlen ist, kann es in Einzelfällen jedoch nötig sein, eine alternative LAN-Anbindung für den Adapter zu schaffen, da eine Kabelinstallation (LAN oder Busleitung) bis zum Wärmeerzeuger nicht realisierbar ist. Dafür gibt es mehrere Möglichkeiten, die im Folgenden kurz vorgestellt werden.

*An dieser Stelle sei aber nochmals darauf hingewiesen, dass der Adapter (nur bei Anbindung via BSB!) auch an ein bereits vorhandenes Raumgerät mittels zusätzlicher Busleitung angeschlossen werden kann.*

### 7.5.1 Nutzung eines PowerLANs / dLANs

Die Nutzung von Powerline-Adaptern, bei denen das 230V-Netz als LAN 'missbraucht' wird, ist eine Option, um eine LAN-Anbindung im Heizungskeller zu realisieren.

Probleme können hierbei jedoch von Steckernetzteilen ausgelöst werden, bei denen bestimmte Frequenzen auf die Stromleitung übertragen werden.

Außerdem müssen sich die Powerline-Adapter bzw. die verwendeten Steckdosen an der gleichen Phase des Stromnetzes befinden. Bei Elektroinstallationen, die bspw. über mehrere Stockwerke gehen und jeweils an einen eigenständigen Sicherungskasten angeschlossen sind, kann es daher zu Problemen kommen. Abhilfe können hier sog. Phasenkoppler schaffen, die jedoch zusätzlich angeschafft und vom Elektriker installiert werden müssen.

### 7.5.2 WLAN: Nutzung eines extra Routers

Eine Möglichkeit für eine WLAN-Anbindung ist, den Adapter via LAN an einen ausgemusterten Router (bspw. eine alte FritzBox) anzuschließen, welcher sich wiederum als Client im bestehenden WLAN-Netz anmeldet. Die Übertragungsraten und Latenzen sind normalerweise für die Nutzung von BSB-LAN absolut ausreichend. Sollte das WLAN-Signal am Aufstellort grenzwertig sein, so könnte der Router mit stärkeren

Antennen ausgerüstet werden.

Neben dem Einsatz eines 'normalen' Routers können auch kleine Geräte genutzt werden, die einen WLAN-Client- bzw. einen WLAN-Client-Bridge-Modus anbieten. Diese Geräte stellen (wie die zuvor beschriebene FritzBox-Lösung) per WLAN eine Verbindung zum Netzwerk her und bieten mit einem zusätzlich verbauten LAN-Port die Möglichkeit, den Arduino Due per LAN-Kabel anzuschließen. Geräte dieser Art sind häufig sehr klein und wie ein Steckernetzteil eine Steckdose, so dass die Installation der Hardware i.d.R. recht unkompliziert stattfinden kann.

In jedem Fall sollte eine möglichst stabile WLAN-Verbindung angestrebt werden - insbesondere dann, wenn via FHEM o.ä. Logdateien erstellt oder mit zusätzlicher Hardware (HK-Thermostate o.ä.) der Wärmeerzeuger gesteuert oder dessen Verhalten beeinflusst werden soll.



Support me on Ko-fi

[Weiter zu Kapitel 8](#)

[Zurück zum Inhaltsverzeichnis](#)



## 8. BSB-LAN: Einbindung mittels zusätzlicher Software

[Zurück zum Inhaltsverzeichnis](#)

[Zurück zu Kapitel 7](#)

## 8. BSB-LAN: Einbindung mittels zusätzlicher Software

### **Achtung:**

*Die in den nachfolgenden Unterkapiteln dargestellten Codebeispiele und/oder Module sind u.U. noch nicht an die Änderungen angepasst, die mit der Umstellung der BSB-LAN-Version auf die v3.x einher gingen. Es kann also sein, dass einige Beispiele nicht mehr 1:1 übernommen werden können, sondern angepasst werden müssen. Als Beispiel sei hier der Wegfall des URL-Befehls /T erwähnt, der in vorherigen Versionen die Werte der angeschlossenen Sensoren ausgab und in den nachfolgenden Beispielen hin und wieder noch vorzufinden ist.*

*Solltest du ein Codebeispiel oder ein Modul entdecken, das in der dargestellten Form nicht mit der aktuellen BSB-LAN-Version v3.x lauffähig ist, so informiere bitte den entspr. Autor des Beispiels/Moduls und schicke mir ggf. ein entsprechend angepasstes Beispiel, damit ich die korrigierte Version hier einstellen kann.*

**Danke.**

Da der Adapter lediglich eine Schnittstelle darstellt, mittels derer man Zugriff auf den Heizungsregler via Computer erhält, können selbstverständlich externe Programme zum Einsatz kommen. Somit kann die Heizungssteuerung in komplexe Heimautomationssysteme eingebunden werden. Auch das Erstellen von umfassenden Logdateien und deren grafische Aufbereitung kann auf diese Weise erfolgen.

Da es hierfür verschiedene Softwarelösungen gibt, kann hier weder eine umfangreiche Vorstellung der verschiedenen Systeme erfolgen, noch eine grundsätzliche Anleitung zur Integration des Adapters oder einer Heizungsregelung per se gegeben werden. Sollte jedoch bereits eines der nachfolgend erwähnten Systeme zum Einsatz kommen, so sind die folgenden Beispielkonfigurationen hoffentlich eine kleine Starthilfe, um die eigenen Vorstellungen umzusetzen.

Sollten Fragen bzgl. der folgenden Beispiele oder anderer Konfigurationsmöglichkeiten auftreten, so ist bitte von diesbezüglichen Anfragen an mich abzusehen - die Fragen sollten mittels entsprechender Literatur oder in spezifischen Foren geklärt werden.

### **Hinweis:**

*Selbstverständlich müssen stets sowohl die IP als auch -falls aktiviert- die optionalen Sicherheitsfunktionen bei den folgenden Beispielen entsprechend angepasst werden. Ebenso müssen Parameter, die gesetzt werden sollen, schreibbar sein (s. Kap. 2.2).*

*Solltest du ein anderes System als die im Folgenden aufgeführten verwenden, so würde ich mich über die Zusendung eines Beispielskripts zur Anbindung des Adapters freuen! Sende es mir einfach als .txt-Datei per Email (adapter (ät) quantentunnel.de) zu - danke!*

## 8.1 FHEM

**Die FHEM-Beispielskripte zur Einbindung mittels HTTPMOD stammen vom FHEM-Forumsmitglied „freetz“.**

**Vielen Dank!**

Um auf die Webschnittstelle des Adapters zuzugreifen, kann das Modul HTTPMOD in FHEM genutzt werden.

### **Beispielskript für eine Parameterabfrage und die Übermittlung einer Raumtemperatur:**

Der Beispielcode fragt die Parameter 8700, 8743 und 8314 alle 300 Sekunden ab und weist diese dem Gerät \"THISION\" (Name des Heizungssystems) und den Readings \"Aussentemperatur\", \"Vorlauftemperatur\" und \"Ruecklauftemperatur\" zu.

Darüber hinaus stellt es ein Reading \"Istwert\" bereit, das per FHEM gesetzt werden kann, um dem Heizungssystem die aktuelle Zimmertemperatur mitzuteilen (Parameter 10000).

Zu guter Letzt berechnet es die Differenz zwischen \"Vorlauftemperatur\" und \"Rücklauftemperatur\" und weist diese Differenz dem Reading \"Spreizung\" zu.

### **Bitte beachte:**

*Die RegEx-Bedingungen müssen vom Beginn des Strings an (also der Parameternummer wie bspw. 8700) matchen und nicht erst ab einem späteren Teil des Strings.*

### **Hinweis:**

*Es scheint ein Problem mit dem HTTPMOD-Modul zu geben, aufgrund dessen Readings nicht erfolgen, die `reading0Name` und `reading0Regex`*

heißen. Bitte beginne daher stets mit **1**.

```
define THISION HTTPMOD http://192.168.178.88/8700/8743/8314 300
attr THISION userattr reading1Name reading1Regex reading2Name reading2Regex reading3Name reading3Regex readingOExpr set1Name set1URL
attr THISION event-on-change-reading .*
attr THISION reading1Name Aussentemperatur
attr THISION reading1Regex 8700 .*:[ \t]+([-]?[\d\.]+)
attr THISION reading2Name Vorlauftemperatur
attr THISION reading2Regex 8743 .*:[ \t]+([-]?[\d\.]+)
attr THISION reading3Name Ruecklauftemperatur
attr THISION reading3Regex 8314 .*:[ \t]+([-]?[\d\.]+)
attr THISION readingOExpr $val=~s/[\r\n]//g;;$val
attr THISION set1Name Istwert
attr THISION set1URL http://192.168.178.88/I10000=$val
attr THISION timeout 5
attr THISION userReadings Spreizung { sprintf("%.1f",ReadingsVal("THISION","Vorlauftemperatur",0)-ReadingsVal("THISION","Ruecklaufte
mpemperatur",0));; }
```

Bei dem obigen Beispiel erfolgt die Darstellung der jeweiligen Readings als numerischer Wert.

Soll die Darstellung wie bei den PullDown-Menüs im Webinterface im Klartext erfolgen, so müssen die regulären Ausdrücke entsprechend angepasst werden.

Im Folgenden ein Beispiel für die Parameter '700 - Betriebsart' und '8000 - Status Heizkreis 1'.

```
attr THISION reading4Name Betriebsart
attr THISION reading4Regex 700 .*-[ \t]+(.*?)
attr THISION reading5Name Status Heizkreis 1
attr THISION reading5Regex 8000 .*-[ \t]+(.*?)
```

Die Nummerierung der zuvor aufgeführten Readings wird hierbei weitergeführt, die Readings sind in der Zeile 'attr THISION userattr' hinzuzufügen.

Außerdem ist die URL dabei um die Parameter 700 und 8000 zu ergänzen.

Zusammengefasst sieht das Ganze dann so aus:

```
define THISION HTTPMOD http://192.168.178.88/8700/8743/8314/700/8000 300
attr THISION userattr reading1Name reading1Regex reading2Name reading2Regex reading3Name reading3Regex reading4Name reading4Regex re
ading5Name reading5Regex readingOExpr set1Name set1URL
attr THISION event-on-change-reading .*
attr THISION reading1Name Aussentemperatur
attr THISION reading1Regex 8700 .*:[ \t]+([-]?[\d\.]+)
attr THISION reading2Name Vorlauftemperatur
attr THISION reading2Regex 8743 .*:[ \t]+([-]?[\d\.]+)
attr THISION reading3Name Ruecklauftemperatur
attr THISION reading3Regex 8314 .*:[ \t]+([-]?[\d\.]+)
attr THISION reading4Name Betriebsart
attr THISION reading4Regex 700 .*-[ \t]+(.*?)
attr THISION reading5Name Status Heizkreis 1
attr THISION reading5Regex 8000 .*-[ \t]+(.*?)
attr THISION readingOExpr $val=~s/[\r\n]//g;;$val
attr THISION set1Name Istwert
attr THISION set1URL http://192.168.178.88/I10000=$val
attr THISION timeout 5
attr THISION userReadings Spreizung { sprintf("%.1f",ReadingsVal("THISION","Vorlauftemperatur",0)-ReadingsVal("THISION","Ruecklaufte
mpemperatur",0));; }
```

### **Beispielscript für die Abfrage und Ansteuerung eines Relaisboards:**

Das Folgende ist ein Beispiel für eine FHEM-Konfiguration, bei dem die drei Relais-Ports namens \"Heater\", \"Fan\" und \"Bell\" abgefragt und gesteuert werden, die an die entsprechenden GPIO-Pins 7, 6 und 5 angeschlossen sind.

```

define EthRelais HTTPMOD http://192.168.178.88/G05/G06/G07 30
attr EthRelais userattr reading1Name reading1Regex reading2Name reading2Regex reading3Name reading3Regex readingOExpr readingOMap se
t1Name set1URL set2Name set2URL set3Name set3URL set1Map setParseResponse:0,1 setRegex
attr EthRelais event-on-change-reading .*
attr EthRelais reading1Name Heater
attr EthRelais reading1Regex GPIO7:[ \t](\d)
attr EthRelais reading2Name Fan
attr EthRelais reading2Regex GPIO6:[ \t](\d)
attr EthRelais reading3Name Bell
attr EthRelais reading3Regex GPIO5:[ \t](\d)
attr EthRelais room Heizung
attr EthRelais set1Name Heater
attr EthRelais set1URL http://192.168.178.88/G07=$val
attr EthRelais set2Name Fan
attr EthRelais set2URL http://192.168.178.88/G06=$val
attr EthRelais set3Name Bell
attr EthRelais set3URL http://192.168.178.88/G05=$val
attr EthRelais setParseResponse 1
attr EthRelais setRegex GPIO[0-9]+:[ \t](\d)
attr EthRelais timeout 5

```

## 8.2 openHAB

Bei Verwendung von openHAB >v2.5.4 kann BSB-LAN mittels Binding eingebunden werden - für ältere openHAB-Versionen existiert kein Binding. Mit dem HTTP-Binding und der Javascript Transformation ist es allerdings durchaus möglich, Werte auszulesen und auch zu schreiben. Ein Loggen der Daten kann bspw. mit InfluxDB erfolgen, die Visualisierung mit Grafana.

### 8.2.1 openHAB-Binding

**BSB-LAN-User „hypetsch“ hat ein [Binding für openHAB](#) entwickelt, das offiziell Teil von openHAB ab v2.5.4 ist!**  
**Vielen Dank!**

### 8.2.2 openHAB mit Javascript Transformation

Für openHAB-Versionen <v2.5.4 muss die Einbindung manuell erfolgen, ein Binding existiert für diese alte openHAB-Version nicht.

**Die openHAB-Beispielscripte stammen vom FHEM-Forumsmitglied „acfsicher42“, zwei Korrekturen/Änderungsvorschläge sowie das Skript zum Anzeigen der Werte in einer Sitemap von „sihui“.**  
**Vielen Dank!**

**ACHTUNG:**

Die notwendigen Addons wie bspw. die Javascript Transformation sind vorhergehend zu installieren!

**Beispiel einer Item-Konfiguration:**

```

Number hz_aussentemp "Aussentemperatur [%1f °C]" <temperature> (Heizunglog) { http="<[http://192.168.178.88/8700:60000:JS(bsbinp
ut.js)]" }
String hz_700 "Heizkreis 1 Betriebsart [%s]" <temperature> (Heizunglog){ http="<[http://192.168.178.88/700:1000:JS(bsbinp
ut_string.js)]" }

```

Das folgende Javascript ist als *bsbinput.js* im Verzeichnis *transform* abzulegen.

**Beispielscript für Abfragen von Parametern, bei denen ein Wert ausgegeben wird (bsbinput.js):**

```

(function(i) {
    var outputres;
    var results = [];
    value1 = i;
    // define regex to search for the line in the http response
    var regex = 'input type=text id=\'value[0-9]+\\' VALUE=\'[-]*[0-9]+\.[0-9]+\';
    var re = new RegExp(regex, 'gim');

    do {
        match = re.exec(value1);
        if (match){
            results.push(match[0]);
        }
    } while (match);

    outputres = results[0]
    //extract actual value from the output
    var output=outputres.substr(outputres.indexOf("VALUE=")+7,outputres.length);
    return output;
})(input)

```

**Beispielscript für direkte Abfragen von enum-Werten (bsbinput\_string.js):**

```

(function(i) {
    var outputres;
    var results = [];
    value1 = i;
    // define regex to search for the line in the http response
    var regex = '<option value=\'[0-9]+\\' SELECTED>.*</option>';
    var re = new RegExp(regex, 'gim');

    do {
        match = re.exec(value1);
        if (match){
            results.push(match[0]);
        }
    } while (match);

    outputres = results[0]
    //extract actual value from the output
    var l=outputres.indexOf("</o")-outputres.indexOf(">")-1
    var output=outputres.substr(outputres.indexOf(">")+1,l);
    return output;
})(input)

```

**Das Schreiben von Daten erfolgt über Rules:**

```

rule "RoomTemp"

when
    Item iSet_temp changed
then
    sendHttpRequest("http://192.168.178.88/I10000="+iSet_temp.state.toString)
end

```

**Anzeigen der Werte in einer Sitemap (BasicUI, ClassicUI, iOS und Android App):**

```

sitemap demo label="Mein BSB LAN" {
    Frame label="Heizung" {
        Text item=hz_aussentemp
        Text item=hz_700
    }
}

```

## 8.2.3 openHAB mit Javascript Transformation, MQTT, Network und Expire

**Basierend auf dem vorhergehenden Beispiel hat FHEM-Forumsmitglied „sihui“ (GitHub: [sihui62](#)) ein erweitertes Beispiel erstellt. Vielen Dank!**

**ACHTUNG:**

















Die notwendigen Addons wie bspw. die Javascript Transformation, MQTT, Network und Expire sind vorhergehend zu installieren! Auf das Anlegen der ggf. notwendigen Things über PaperUI wird ebenfalls nicht näher eingegangen.

**Hinweis:**

Das folgende Beispiel muss selbstverständlich individuell angepasst werden. Dabei ist insbesondere darauf zu achten, dass gewisse heizungsseitige Parameter/Funktionen nicht bei jedem Gerät verfügbar sind!

Des Weiteren kann eine Anzeige der Raumtemperatur nur dann erfolgen, wenn diese auch übermittelt wird (bspw. durch ein entspr. Raumgerät). Die Präsenztastenfunktion kann u.U. bei einigen Reglertypen nicht gegeben sein (s. [entspr. Kap.](#)) und sollte daher im Vorfeld überprüft werden. MAX!-Komponenten können selbstverständlich ebenfalls nur genutzt werden, wenn sie vorhandenen und entspr. eingebunden sind (s. [entspr. Kap.](#)).

**Das folgende Beispiel wird als Sitemap in BasicUI wie in folgendem Screenshot angezeigt:**

 IST Betriebsart	Automatik	 SOLL Betriebsart	Automatik ▼
 Gesetzte Temperatur	20,5 °C	 SOLL Temperatur	20,5 °C ^ v
 Status Heizung	Schnellaufheizung	 Status Wasser	Geladen, Nenntemperatur
 Präsenz Komfort	<input type="checkbox"/>	 Präsenz Reduziert	<input type="checkbox"/>
 Partymodus	1h ▼	 Raumtemperatur RGT	20,3 °C
 MAX! Küche	20,5 °C	 Aussentemperatur	5,1 °C
 Vorlauftemperatur	43,6 °C	 Rücklauftemperatur	38,2 °C
 BSB LAN Online Status	ON	 Gebläsedrehzahl	5187 U/min

**Beispiel einer Item-Konfiguration (/items/bsblan.items):**

```

Number hz_mode_cmd <heating> //change heating mode
String hz_mode_state <heating> { http="<[http://192.168.178.88/700:10000:JS(bsbinput_string.js)]" } //read heating mode from BSB LAN Adapter
Number hz_temperature_cmd <temperature> //change target temperature
Number hz_temperature_state <temperature> { http="<[http://192.168.178.88/8741:15000:JS(bsbinput.js)]" } //read current target temperature from BSB LAN Adapter
String hz_status <heating> { http="<[http://192.168.178.88/8000:20000:JS(bsbinput_string.js)]" } //read current heating status from BSB LAN Adapter
String hz_status_water <water> { http="<[http://192.168.178.88/8003:25000:JS(bsbinput_string.js)]" } //read current hot water status from BSB LAN Adapter
Switch hz_mode_komfort <switch> { expire="1s,command=OFF" } //ONLY if Parameter 48 is available on your controller: set temporary Komfort state during Automatik mode, switch item to OFF after one second (momentary switch)
Switch hz_mode_reduziert <switch> { expire="1s,command=OFF" } //ONLY if Parameter 48 is available on your controller: set temporary Reduziert state during Automatik mode, switch item to OFF after one second (momentary switch)
Number hz_temperature_rgt <temperature> { http="<[http://192.168.178.88/8740:25000:JS(bsbinput.js)]" } //read current room temperature for remote RGT from BSB LAN Adapter
Number hz_fan_speed <fan> { http="<[http://192.168.178.88/8323:30000:JS(bsbinput.js)]" } //read current fan speed from BSB LAN Adapter
Number hz_aussentemp <temperature> { http="<[http://192.168.178.88/8700:20000:JS(bsbinput.js)]" } //read current outside temperature from BSB LAN Adapter via Javascript Transformation (not used here)
Number hz_kitchen_maxActual "MAX! Küche [%1f °C]" { channel="max:thermostat:KEQ0565026:KEQ0648949:actual_temp" } //read temperature from MAX!
Number BSBLAN_Aussentemp <temperature> { channel="mqtt:topic:bsblan:aussentemp" } //read current outside temperature from BSB LAN Adapter via MQTT2
Number BSBLAN_Vorlauftemp <temperature> { channel="mqtt:topic:bsblan:vorlauftemp" } //read current flow temperature from BSB LAN Adapter via MQTT2
Number BSBLAN_Ruecklauftemp <temperature> { channel="mqtt:topic:bsblan:ruecklauftemp" } //read current return temperature from BSB LAN Adapter via MQTT2
Switch bsb_lan_presence <presence> { channel="network:pingdevice:192_168_178_88:online" } //check online status of BSB LAN through Network binding
Number hz_mode_party <party> //enable or disable Party mode for 1-5 hours

```

Das folgende Javascript ist als *bsbinput.js* im Verzeichnis *transform* abzulegen.

**Beispielscript für Abfragen von Parametern, bei denen ein Wert ausgegeben wird (/transform/bsbinput.js):**



```

(function(i) {
    var outputres;
    var results = [];
    value1 = i;
    // define regex to search for the line in the http response
    var regex = 'input type=text id=\'value[0-9]+\\' VALUE=\'[-]*[0-9]+\.[0-9]+\';
    var re = new RegExp(regex, 'gim');

    do {
        match = re.exec(value1);
        if (match){
            results.push(match[0]);
        }
    } while (match);

    outputres = results[0]
    //extract actual value from the output
    var output=outputres.substr(outputres.indexOf("VALUE=")+7,outputres.length);
    return output;
})(input)

```

#### Beispielscript für direkte Abfragen von enum-Werten (/transform/bsbinput\_string.js):

```

(function(i) {
    var outputres;
    var results = [];
    value1 = i;
    // define regex to search for the line in the http response
    var regex = '<option value=\'[0-9]+\\' SELECTED>.*</option>';
    var re = new RegExp(regex, 'gim');

    do {
        match = re.exec(value1);
        if (match){
            results.push(match[0]);
        }
    } while (match);

    outputres = results[0]
    //extract actual value from the output
    var l=outputres.indexOf("</o")-outputres.indexOf(">")-1
    var output=outputres.substr(outputres.indexOf(">")+1,l);
    return output;
})(input)

```

#### Das Schreiben und Auslesen von Daten erfolgt über Rules (/rules/bsblan.rules):

```

var Timer PartyModeTimer = null //initialize a timer for party mode

rule "HeatingTempTarget" //change target temperature
when
    Item hz_temperature_cmd changed
then
    sendHttpRequest("http://192.168.178.88/S710="+hz_temperature_cmd.state.toString)
end

rule "HeatingMode" //change heating mode
when
    Item hz_mode_cmd changed
then
    sendHttpRequest("http://192.168.178.88/S700="+hz_mode_cmd.state.toString)
end

rule "UpdateHeatingMode" //reflect manual RGT remote changes on UI
when
    Item hz_mode_state changed
then
    hz_mode_cmd.postUpdate(transform("MAP","heatingmode.map",hz_mode_state.state.toString))
end

rule "SetModeKomfort" //set mode temporary to Komfort during Automatik mode
when
    Item hz_mode_komfort changed to ON
then
    sendHttpRequest("http://192.168.178.88/S701=0")
end

rule "SetModeReduziert" //set mode temporary to Reduziert during Automatik mode
when
    Item hz_mode_reduziert changed to ON
then
    sendHttpRequest("http://192.168.178.88/S701=1")
end

```

```

rule "SetPartyMode" //extends heating Komfort time for 1-5 hours
when
    Item hz_status changed
then
    // to do: read shutdown times for Absenkung Reduziert dynamically from BSB LAN Adapter
    if (hz_status.state.toString=="Absenkung Reduziert" && (now.getHourOfDay()>=22 && (now.getHourOfDay()<=23))) { //only trigger rule content during normal Reduziert shutdown times
        switch (hz_mode_party.state) {
            case 1: {
                if(PartyModeTimer!=null) {
                    PartyModeTimer.cancel
                    PartyModeTimer = null
                }
                PartyModeTimer = createTimer(now.plusHours(1)) [ |
                    hz_mode_cmd.sendCommand(1)
                    logInfo("BSBLAN","Party Mode disabled")
                ]
                hz_mode_cmd.sendCommand(3)
                hz_mode_party.postUpdate(0)
                logInfo("BSBLAN","Party Mode 1h")
            }
            case 2: {
                if(PartyModeTimer!=null) {
                    PartyModeTimer.cancel
                    PartyModeTimer = null
                }
                PartyModeTimer = createTimer(now.plusHours(2)) [ |
                    hz_mode_cmd.sendCommand(1)
                    logInfo("BSBLAN","Party Mode disabled")
                ]
                hz_mode_cmd.sendCommand(3)
                hz_mode_party.postUpdate(0)
                logInfo("BSBLAN","Party Mode 2h")
            }
            case 3: {
                if(PartyModeTimer!=null) {
                    PartyModeTimer.cancel
                    PartyModeTimer = null
                }
                PartyModeTimer = createTimer(now.plusHours(3)) [ |
                    hz_mode_cmd.sendCommand(1)
                    logInfo("BSBLAN","Party Mode disabled")
                ]
                hz_mode_cmd.sendCommand(3)
                hz_mode_party.postUpdate(0)
                logInfo("BSBLAN","Party Mode 3h")
            }
            case 4: {
                if(PartyModeTimer!=null) {
                    PartyModeTimer.cancel
                    PartyModeTimer = null
                }
                PartyModeTimer = createTimer(now.plusHours(4)) [ |
                    hz_mode_cmd.sendCommand(1)
                    logInfo("BSBLAN","Party Mode disabled")
                ]
                hz_mode_cmd.sendCommand(3)
                hz_mode_party.postUpdate(0)
                logInfo("BSBLAN","Party Mode 4h")
            }
            case 5: {
                if(PartyModeTimer!=null) {
                    PartyModeTimer.cancel
                    PartyModeTimer = null
                }
                PartyModeTimer = createTimer(now.plusHours(5)) [ |
                    hz_mode_cmd.sendCommand(1)
                    logInfo("BSBLAN","Party Mode disabled")
                ]
                hz_mode_cmd.sendCommand(3)
                hz_mode_party.postUpdate(0)
                logInfo("BSBLAN","Party Mode 5h")
            }
        }
    }
}

end

rule "ConsiderRoomTempFromKitchen" //feed external temperatures to controller, for example MAX!
when
    Item hz_kitchen_maxActual changed
then
    sendHttpRequest("http://192.168.178.88/I10000="+hz_kitchen_maxActual.state.toString)
end

```

## Transformation von Zahlenwerten zu besser lesbaren Texten (/transform/heatingmode.map):

```
Automatik=1
Reduziert=2
Komfort=3
Schutzbetrieb=0
```

## Anzeigen der Werte in einer Sitemap (/sitemaps/bsblan.sitemap, z.B. für BasicUI, ClassicUI, iOS und Android App):

```
sitemap bsblan label="Mein BSB LAN"
{
  Frame
  {
    Text label="Heizung" icon="heating"
    {
      Text item=h_z_mode_state label="IST Betriebsart [%s]"
      Selection item=h_z_mode_cmd label="SOLL Betriebsart [%s]" mappings=[1="Automatik",3="Komfort",2="Reduziert"]
      Text item=h_z_temperature_state label="Gesetzte Temperatur [%1f °C]"
      Setpoint item=h_z_temperature_cmd label="SOLL Temperatur [%1f °C]" minValue=16 maxValue=24 step=0.5
      Text item=h_z_status label="Status Heizung [%s]"
      Text item=h_z_status_water label="Status Wasser [%s]"
      Switch item=h_z_mode_komfort label="Präsenz Komfort"
      Switch item=h_z_mode_reduziert label="Präsenz Reduziert"
      Selection item=h_z_mode_party label="Partymodus [%s]" mappings=[0="Aus",1="1h",2="2h",3="3h",4="4h",5="5h"]
      Text item=h_z_temperature_rgt label="Raumtemperatur RGT [%1f °C]"
      Text item=h_z_kitchen_maxActual label="MAX! Küche [%1f °C]"
      Text item=BSBLAN_Aussentemp label="Aussentemperatur [%1f °C]"
      Text item=BSBLAN_Vorlauftemp label="Vorlauftemperatur [%1f °C]"
      Text item=BSBLAN_Ruecklauftemp label="Rücklauftemperatur [%1f °C]"
      Text item=bsb_lan_presence label="BSB LAN Online Status [%s]"
      Text item=h_z_fan_speed label="Gebläsedrehzahl [%d U/min]"
    }
  }
}
```

## 8.3 HomeMatic (EQ3)

Die folgenden HomeMatic-Beispielskripte stammen vom FHEM-Forumsmitglied „Bratmaxe“.

Sie sind samt einer genaueren Beschreibung ebenfalls [hier](#) im FHEM-Forum zu finden (die hier eingefügten Beschreibungen wurden von dort größtenteils unverändert übernommen).

Das letzte Beispiel beinhaltet die Abfrage optional angeschlossener DS18B20-Temperatur Sensoren mittels der spezifischen SensorIDs und der Ausgabe von /T.

Im Anschluss daran sind die Beispielskripte vom FHEM-Forumsmitglied "PaulM" aufgeführt.

Vielen Dank!

### Beispielskript für die Abfrage des Adapters:

Es müssen lediglich 6 Parameter eingegeben werden.

CuxGeraetAbfrage = GeräteAdresse des CuxD Execute Gerätes, welches die Abfragen ausführt

CuxGeraetLogging = GeräteAdresse des CuxD Execute Gerätes, welches das Logging ausführt (leer==deaktiviert)

IPAdresseBSB = IP-Adresse des BSB-Adapters

Wort = Parameternummer: Beispiel Außentemperatur = 8700 oder Betriebsmodus = 700

Variablenname = Name der Systemvariable in der CCU

Durchschnitt24h = true == Durchschnittswert 24h holen, false == aktuellen Wert holen

Es muss keine Variable vorher angelegt werden, das erledigt das Skript.

Der Variabeltyp (Zahl, Bool, Werteliste) wird automatisch an den abgefragten Parameter angepasst.

```
! BSB-Adapter Wert abfragen by Bratmaxe
! 29.10.2018 - V0.1 - Erste Version

string CuxGeraetAbfrage = "CUX2801001:1"; ! GeräteAdresse des CuxD Execute Gerätes, welches die Abfragen ausführt
string CuxGeraetLogging = "CUX2801001:1"; ! GeräteAdresse des CuxD Execute Gerätes, welches das Logging ausführt, Leer ("" ) lassen,
wenn kein Cuxd-Highcharts Logging gewünscht
string IPAdresseBSB = "192.168.178.100"; !IP_Adresse des BSB-Adapters
string Wort = "8700"; !Parameternummer: Beispiel Außentemperatur = 8700, Betriebsmodus = 700
string Variablenname = "Wetter_Temperatur_Heizung"; ! Name der Systemvariable
boolean Durchschnitt24h = false; ! true = Durchschnittswert holen, false = aktuellen Wert holen - diese muss vorher in der BSB_lan_c
onfig.h konfiguriert wurden sein!!!

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

```

! URL Zusammenführen
string url="";
if (Durchschnitt24h) { url="http://" # IPAdresseBSB # "/" # Wort; }
else { url="http://" # IPAdresseBSB # "/" # Wort; }
! Variable anlegen, wenn nicht vorhanden:
object svObject = dom.GetObject(Variablename);
object svObjectlist = dom.GetObject(ID_SYSTEM_VARIABLES);
if (!svObject)
{
    svObject = dom.CreateObject(OT_VARDP);
    svObjectlist.Add(svObject.ID());
    svObjectlist = dom.GetObject(ID_SYSTEM_VARIABLES);
    svObject.Name(Variablename);
    svObject.Internal(false);
    svObject.Visible(true);
}

! Werte holen
dom.GetObject("CUxD." # CuxGeraetAbfrage # ".CMD_SETS").State("wget --tries=5 --timeout=20 --quiet --output-document=- '" # url # "'");
;
dom.GetObject("CUxD." # CuxGeraetAbfrage # ".CMD_QUERY_RET").State(1);
var stdout = dom.GetObject("CUxD." # CuxGeraetAbfrage # ".CMD_RETS").State();

! Prüfe, ob eine Ausgabe vorhanden ist, sonst z.B. IP-Adresse falsch, oder Netzwerkfehler
if (stdout != null && stdout != "")
{
    ! Ausgabe filtern
    integer pos = (stdout.Find(Wort# " "));
    if (pos == -1)
    {
        WriteLine("Position vom Wort '" # Wort # "' konnte nicht ermittelt werden");
    }

    stdout = stdout.Substr(pos, stdout.Length());
    pos = stdout.Find("/td");
    stdout = stdout.Substr(0, pos);

    ! Sonderzeichen ersetzen
    if (stdout.Contains("°")){ stdout = stdout.Replace("°", "°"); }
    if (stdout.Contains("&#037;")){ stdout = stdout.Replace("&#037;", "%"); }
    stdout = stdout.ToLatin();
    !WriteLine("Nach Sonderzeichenumwandlung: " # stdout); !Debug: Welchen Wert hat stdout aktuell

    ! Systemvariabel Info ermitteln
    string Info = stdout.Substr(0, stdout.Find(":"));
    !Info = Info.Substr(Wort.Length(), stdout.Length()); !Parameterzahl vor der Info entfernen
    !WriteLine("DPInfo = " # Info); !Debug: Welcher DPInfo-Wert wurde gefunden

    ! Systemvariabel Wert ermitteln
    string Wert = stdout.Substr(stdout.Find(": ") + 2, stdout.Length());
    Wert = Wert.Substr(0, Wert.Find(" "));
    !WriteLine("Wert = " # Wert); !Debug: Welcher Wert wurde gefunden

    ! Systemvariabel Einheit ermitteln
    string Einheit = stdout.Substr(stdout.Find(Info) + Info.Length() + 1, stdout.Length());
    Einheit = Einheit.Substr(Einheit.Find(Wert) + Wert.Length() + 1, Einheit.Length());
    Einheit = Einheit.RTrim();
    if (Einheit.Contains("- ")) { Einheit = ""; }
    !WriteLine("Einheit = " # Einheit); !Debug: Welche Einheit wurde gefunden

    ! Systemvariable Typ und Werte setzen
    svObject.DPInfo(Info);
    svObject.ValueUnit(Einheit);

    ! Enums des Parameters ermitteln, wenn vorhanden
    url="http://" # IPAdresseBSB # "/" # Wort;

    dom.GetObject("CUxD." # CuxGeraetAbfrage # ".CMD_SETS").State("wget --tries=5 --timeout=20 --quiet --output-document=- '" # url # "'");
    dom.GetObject("CUxD." # CuxGeraetAbfrage # ".CMD_QUERY_RET").State(1);
    stdout = dom.GetObject("CUxD." # CuxGeraetAbfrage # ".CMD_RETS").State();
    ! Prüfe, ob es sich um einen Parameter mit Enum-Werten handelt.
    if (!stdout.Contains("FEHLER: Falscher Typ!"))
    {
        ! Setzen des Systemvariabel Wertetyp und Ermitteln der Enum-Werte des Parameters
        stdout = (stdout.Substr(stdout.Find("0 - "), stdout.Length())).ToLatin();
        string value = "";
        string newvalues = "";
        integer inewvalues=0;
        foreach (value, stdout.Split("\r"))
        {
            if (value.Contains(" - "))
            {
                if (newvalues == "") { newvalues = newvalues # value.Substr(value.Find(" - ") + 3, value.Length()); }
                else { newvalues = newvalues # ";" # value.Substr(value.Find(" - ") + 3, value.Length()); }
                inewvalues = inewvalues + 1;
            }
        }
    }
}

```

```

    }
}

svObject.ValueType(ivtInteger);
svObject.ValueSubType(istEnum);
svObject.ValueList(newvalues);
!prüft, ob der ermittelte Wert innerhalb der möglichen Werte liegt
if (Wert < inewvalues) { if (Wert != svObject.Value()) { svObject.State(Wert); } }
else { WriteLine("Der ermittelte Wert entspricht keinem gültigen Enum-Wert. Bitte Ausgabe prüfen!") }
}
elseif (Einheit.Contains("- Aus") || Einheit.Contains("- Ein"))
{
    ! Setzen des Systemvariabel Wertetyp
    svObject.ValueType(ivtBinary);
    svObject.ValueSubType(istBool);
    svObject.ValueName0("Aus");
    svObject.ValueName1("Ein");
    if (Wert != svObject.Value()) { svObject.State(Wert); }
}
elseif (Einheit.Contains("°"))
{
    ! Setzen des Systemvariabel Wertetyp
    svObject.ValueType(ivtFloat);
    svObject.ValueSubType(istGeneric);
    svObject.ValueMin(-50);
    svObject.ValueMax(100);
    if (Wert != svObject.Value()) { svObject.State(Wert); }
}
elseif (Einheit.Contains("%"))
{
    ! Setzen des Systemvariabel Wertetyp
    svObject.ValueType(ivtFloat);
    svObject.ValueSubType(istGeneric);
    svObject.ValueMin(0);
    svObject.ValueMax(100);
    if (Wert != svObject.Value()) { svObject.State(Wert); }
}
else
{
    ! Setzen des Systemvariabel Wertetyp
    svObject.ValueType(ivtFloat);
    svObject.ValueSubType(istGeneric);
    if (Wert != svObject.Value()) { svObject.State(Wert); }
}
dom.RTUpdate(0); ! Interne Aktualisierung der Systemvariablen

! Logging
if (CuxGeraetLogging != "") { dom.GetObject("CUXD."#CuxGeraetLogging#.LOGIT").State(dom.GetObject(ID_SYSTEM_VARIABLES).Get(Variablenname).Name()#";"#dom.GetObject(ID_SYSTEM_VARIABLES).Get(Variablenname).Value()); }
}

```

### Skript zum Setzen von Parametern:

Ein Programm, wo alle Systemvariablen die überwacht werden sollen mit ODER Verknüpft und größer oder gleich 0 und "bei Aktualisierung auslösen", anlegen.

Beispiel:

```

WENN Variablenname größer oder gleich 0 "bei Aktualisierung auslösen"
DANN Dieses SKRIPT sofort ausführen

```

Die Variable muss in der Info zuerst den ParameterWert enthalten (wird vom obigen Auslese-Skript automatisch so benannt). Beispiel: 700 Heizkreis 1 - Betriebsart

Die Parameternummer wird dann automatisch aus der Systemvariable Info ermittelt.

Wird die Variable geändert, so wird der geänderte Wert automatisch an den BSB-Adapter übermittelt und aktualisiert!

```

! BSB-Adapter Wert setzen by Bratmaxe
! 29.10.2018 - V0.1 - Erste Version

! Funktionsbeschreibung:
! Ein Programm, wo alle Systemvariablen die überwacht werden sollen mit ODER Verknüpft und größer oder gleich 0 und "bei Aktualisierung auslösen", anlegen.
! Beispiel:
! WENN Variablenname größer oder gleich 0 "bei Aktualisierung auslösen"
! DANN Dieses SKRIPT sofort ausführen
! die Variable muss in der Info zuerst den Parameter-Wert enthalten (wird von meinem Auslese Skript automatisch so benannt. Beispiel : 700 Heizkreis 1 - Betriebsart
! Die Parameternummer wird dann automatisch aus der Systemvariable Info ermittelt.
! Wird die Variable geändert, so wird der geänderte Wert automatisch an den BSB-Adapter übermittelt und aktualisiert!

string CuxGeraetSetzen = "CUX2801001:12"; ! GeräteAdresse des CuxD Execute Gerätes
string IPAdresseBSB = "192.168.2.200"; !IP_Adresse des BSB-Adapters

```



```

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!Ab hier keine Anpassungen mehr notwendig!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

! Hole Auslösende Variabel
var source = dom.GetObject("$src$"); !Funktioniert nur beim automatischen Aufruf
! Zum manuellen Aufruf/testen nächste Zeile einkommentieren
!source = dom.GetObject(ID_SYSTEM_VARIABLES).Get("VARIABLENAMEN");

if (source)
{
    ! Wort ermitteln
    string Wort = source.DPInfo().ToString().Substr(0,source.DPInfo().Find(" "));
    !WriteLine("Wort: " #Wort);
    if (Wort != null && Wort != "")
    {
        string Wert = source.Value().ToString();
        !WriteLine("Wert: " #Wert);
        if (Wert != null && Wert != "")
        {
            ! Anweisung senden
            string urlset="http://" # IPAdresseBSB # "/"S" # Wort # "=" # Wert;
            dom.GetObject("CUXD." # CuxGeraetSetzen # ".CMD_SETS").State("wget -t 5 -T 20 -q -O - '"# urlset #"");
            dom.GetObject("CUXD." # CuxGeraetSetzen # ".CMD_QUERY_RET").State(1);
            var stdout = dom.GetObject("CUXD." # CuxGeraetSetzen # ".CMD_RETS").State();
            if (stdout != null && stdout != "")
            {
                if (stdout.Contains("FEHLER: "))
                {
                    stdout = stdout.Substr(stdout.Find("FEHLER: "), stdout.Length());
                    stdout = stdout.Substr(0, stdout.Find("/td"));
                    WriteLine("Fehlermeldung: " # stdout);
                    WriteLine("Wurde der BSB-Adapter zum Schreiben berechtigt? Handbuch Seite 26 beachten...");
                }
                else
                {
                    ! Kontrollabfrage
                    string url="http://" # IPAdresseBSB # "/" # Wort;
                    dom.GetObject("CUXD." # CuxGeraetSetzen # ".CMD_SETS").State("wget -t 5 -T 20 -q -O - '"# url #"");
                    dom.GetObject("CUXD." # CuxGeraetSetzen # ".CMD_QUERY_RET").State(1);
                    stdout = dom.GetObject("CUXD." # CuxGeraetSetzen # ".CMD_RETS").State();

                    ! Ausgabe filtern
                    integer pos = (stdout.Find("tr td \r\n" # Wort # " ") + 9);
                    stdout = stdout.Substr(pos, stdout.Length());
                    pos = stdout.Find("/td");
                    stdout = stdout.Substr(0, pos);

                    ! Sonderzeichen ersetzen
                    if (stdout.Contains("°")){ stdout = stdout.Replace("°","°"); }
                    if (stdout.Contains("#037;")){ stdout = stdout.Replace("#037;","%"); }
                    !WriteLine("Nach Sonderzeichenumwandlung: " # stdout); !Debug: Welchen Wert hat stdout aktuell

                    ! Systemvariabel oldWert ermitteln
                    string oldWert = stdout.Substr(stdout.Find(": ") + 2,stdout.Length());
                    oldWert = oldWert.Substr(0,oldWert.Find(" "));
                    !WriteLine("oldWert = " # oldWert.ToFloat()); !Debug: Welcher oldWert wurde gefunden
                    !WriteLine("newWert = " # Wert.ToFloat()); !Debug: Welcher oldWert wurde gefunden

                    if (Wert.ToFloat() != oldWert.ToFloat()) { WriteLine("Fehler: Werte stimmen nach setzen nicht überein!"); }
                    else { WriteLine("Wert wurde erfolgreich gesetzt"); }
                }
            }
            else { WriteLine("Keine Ausgabe gefunden. IP-Adresse und Verkabelung prüfen."); }
        }
        else { WriteLine("Der neue Wert konnte nicht ermittelt werden."); }
    }
    else { WriteLine("Wort konnte nicht ermittelt werden, Steht der Wert in der SystemvariableInfo am Anfang gefolgt von einem Leerzeichen?"); }
}
else { WriteLine("Auslösende Variable nicht erkannt! - Skript wird nicht ausgeführt."); }

```

#### Abfrage von heizungsseitigen Fehlermeldungen zwecks Benachrichtigung im Störfall:

```

! BSB-Adapter Wert abfragen Fehlercodes by Bratmaxe
! 05.11.2018 - V0.1 - Erste Version

string CuxGeraetAbfrage = "CUX2801001:1"; ! GeräteAdresse des CuxD Execute Gerätes, welches die Abfragen ausführt
string IPAdresseBSB = "192.168.178.100"; !IP_Adresse des BSB-Adapters
string Variablename = "Heizung_Fehlercodes"; ! Name der Systemvariable
integer AnzahlFehler = 10;

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!Ab hier keine Anpassungen mehr notwendig!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

```

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

```
! Parameter Zusammenbauen
integer i =0;
string Woerter ="";
while (i < AnzahlFehler)
{
    if (Woerter != "")
    {
        Woerter = Woerter + "," + ((6801) + (10 * i)).ToString();
    }
    else { Woerter = Woerter + ((6801) + (10 * i)).ToString(); }
    i = i + 1;
}

! URL Zusammenführen
string Ergebnis = "";
string Wort = "";

foreach(Wort, Woerter.Split(","))
{
    string url="http://" # IPAdresseBSB # "/" # ((Wort.ToInteger() - 1).ToString());

    ! Werte holen
    dom.GetObject("CUxD." # CuxGeraetAbfrage # ".CMD_SETS").State("wget --tries=5 --timeout=20 --quiet --output-document=- '"# url # """);
    dom.GetObject("CUxD." # CuxGeraetAbfrage # ".CMD_QUERY_RET").State(1);
    var stdout = dom.GetObject("CUxD." # CuxGeraetAbfrage # ".CMD_RETS").State();

    ! Prüfe, ob eine Ausgabe vorhanden ist, sonst z.B. IP-Adresse falsch, oder Netzwerkfehler
    if (stdout != null && stdout != "")
    {
        ! Ausgabe filtern
        integer pos = (stdout.Find((Wort.ToInteger() - 1).ToString() # " "));
        stdout = stdout.Substr(pos, stdout.Length());
        pos = stdout.Find("/td");
        stdout = stdout.Substr(0, pos);

        ! Sonderzeichen ersetzen
        if (stdout.Contains(""")){ stdout = stdout.Replace(""", ""); }
        if (stdout.Contains("%")){ stdout = stdout.Replace("%", "%"); }
        stdout = stdout.ToLatin();
        Ergebnis = Ergebnis # stdout.RTrim() # "\n\r";
    }

    url="http://" # IPAdresseBSB # "/" # Wort;
    dom.GetObject("CUxD." # CuxGeraetAbfrage # ".CMD_SETS").State("wget --tries=5 --timeout=20 --quiet --output-document=- '"# url # """);
    dom.GetObject("CUxD." # CuxGeraetAbfrage # ".CMD_QUERY_RET").State(1);
    stdout = dom.GetObject("CUxD." # CuxGeraetAbfrage # ".CMD_RETS").State();

    ! Prüfe, ob eine Ausgabe vorhanden ist, sonst z.B. IP-Adresse falsch, oder Netzwerkfehler
    if (stdout != null && stdout != "")
    {
        ! Ausgabe filtern
        integer pos = (stdout.Find(Wort# " "));
        stdout = stdout.Substr(pos, stdout.Length());
        pos = stdout.Find("/td");
        stdout = stdout.Substr(0, pos);

        ! Sonderzeichen ersetzen
        if (stdout.Contains(""")){ stdout = stdout.Replace(""", ""); }
        if (stdout.Contains("%")){ stdout = stdout.Replace("%", "%"); }
        stdout = stdout.ToLatin();
        Ergebnis = Ergebnis # stdout.RTrim() # "\n\r\n\r";
    }
}

!Wenn noch keine Systemvariable vorhanden, diese anlegen
object svObject = dom.GetObject(Variablename);
object svObjectlist = dom.GetObject(ID_SYSTEM_VARIABLES);
if (!svObject)
{
    svObjectlist = dom.GetObject(ID_SYSTEM_VARIABLES);
    svObject = dom.CreateObject(OT_VARDP);
    svObjectlist.Add(svObject.ID());
    svObject.Name(Variablename);
    svObject.ValueType(ivtString);
    svObject.ValueSubType(istChar8859);
    svObject.DPInfo("Die letzten 20 Fehlercodes der Heizung");
    svObject.Internal(false);
    svObject.Visible(true);
    dom.RTUpdate(0);
}

if (Ergebnis.ToLatin() != svObject.Value().ToLatin()) { svObject.State(Ergebnis); }
```

```

! BSB-Adapter Wert abfragen by Bratmaxe
! 29.10.2018 - V0.1 - Erste Version
! 11.11.2019 - V0.2 - Auslesen von Temperatursensoren hinzugefügt
! 15.11.2019 - V0.3 - Änderung der Ausleseart der Temperatursensoren mithilfe der ID

string CuxGeraetAbfrage = "CUX2801001:11"; ! GeräteAdresse des CuxD Execute Gerätes, welches die Abfragen ausführt
string CuxGeraetLogging = "CUX2801001:10"; ! GeräteAdresse des CuxD Execute Gerätes, welches das Logging ausführt, Leer (") lassen,
wenn kein Cuxd-Highcharts Logging gewünscht
string IPAdresseBSB = "192.168.178.88"; !IP_Adresse des BSB-Adapters
string Wort = "T"; !Parameternummer: Beispiel Außentemperatur = 8700, Betriebsmodus = 700, eigene Temperatursensoren = T
string TemperatursensorID = "28aa44085414010b"; !Wenn Wort = "T", dann hier die ID des auszulesenden Temperatursensors eingeben, wir
d sonst ignoriert!
string Variablename = "Wetter_Temperatur"; ! Name der Systemvariable
boolean Durchschnitt24h = false; ! true = Durchschnittswert holen, false = aktuellen Wert holen - diese muss vorher in der BSB_lan_c
onfig.h konfiguriert wurden sein!!! (Bei Wort = T wird dieser Parameter ignoriert)

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

! URL Zusammenführen
string url="";
if (Durchschnitt24h && Wort != "T")
{
    url="http://" # IPAdresseBSB # "/"A" # Wort;
}
else
{
    url="http://" # IPAdresseBSB # "/" # Wort;
}

! Variable anlegen, wenn nicht vorhanden:
object svObject = dom.GetObject(Variablename);
object svObjectlist = dom.GetObject(ID_SYSTEM_VARIABLES);
if (!svObject)
{
    svObject = dom.CreateObject(OT_VARDP);
    svObjectlist.Add(svObject.ID());
    svObjectlist = dom.GetObject(ID_SYSTEM_VARIABLES);
    svObject.Name(Variablename);
    svObject.Internal(false);
    svObject.Visible(true);
}

! Werte holen
dom.GetObject("CUXxD." # CuxGeraetAbfrage # ".CMD_SETS").State("wget --tries=5 --timeout=20 --quiet --output-document=- '"# url #"'"");
;
dom.GetObject("CUXxD." # CuxGeraetAbfrage # ".CMD_QUERY_RET").State(1);
var stdout = dom.GetObject("CUXxD." # CuxGeraetAbfrage # ".CMD_RET").State();

! Prüfe, ob eine Ausgabe vorhanden ist, sonst z.B. IP-Adresse falsch, oder Netzwerkfehler
if (stdout != null && stdout != "")
{
    integer pos = (stdout.Find(Wort# " "));

    ! Ausgabe filtern
    if (Wort == "T")
    {
        pos = (stdout.Find(TemperatursensorID # ": "));
    }

    if (pos == -1)
    {
        WriteLine("Position vom Wort '" # Wort # "' konnte nicht ermittelt werden");
    }

    stdout = stdout.Substr(pos, stdout.Length());
    pos = stdout.Find("/td");
    stdout = stdout.Substr(0, pos);

    ! Sonderzeichen ersetzen
    if (stdout.Contains("°")){ stdout = stdout.Replace("°", "°"); } !& d e g ; ohne Leerzeichen
    if (stdout.Contains("%")){ stdout = stdout.Replace("%", "%"); } !& # 0 3 7 ; ohne Leerzeichen
    !WriteLine("Nach Sonderzeichenumwandlung: " # stdout); !Debug: Welchen Wert hat stdout aktuell

    ! Systemvariabel Info ermitteln
    string Info = "";
    if (Wort == "T")
    {
        Info = "SensorID: " + TemperatursensorID;
    }
    else
    {
        Info = stdout.Substr(0, stdout.Find(":"));
    }
}

```

```

}
!Info = Info.Substr(Wort.Length(), stdout.Length());
!WriteLine("DPInfo = " # Info); !Debug: Welcher DPInfo-Wert wurde gefunden

! Systemvariabel Wert ermitteln
string Wert = stdout.Substr(stdout.Find(": ") + 2, stdout.Length());
Wert = Wert.Substr(0, Wert.Find(" "));
!WriteLine("Wert = " # Wert); !Debug: Welcher Wert wurde gefunden

! Systemvariabel Einheit ermitteln
string Einheit = stdout.Substr(stdout.Find(Info) + Info.Length() + 1, stdout.Length());
Einheit = Einheit.Substr(Einheit.Find(Wert) + Wert.Length() + 1, Einheit.Length());
Einheit = Einheit.RTrim();
if (Einheit.Contains("- ")) { Einheit = ""; }
!WriteLine("Einheit = " # Einheit); !Debug: Welche Einheit wurde gefunden

! Systemvariable Typ und Werte setzen
svObject.DPInfo(Info);
svObject.ValueUnit(Einheit);

! Enums des Parameters ermitteln, wenn vorhanden
url="http://" # IPAdresseBSB # "/"E" # Wort;

dom.GetObject("CUXD." # CuxGeraetAbfrage # ".CMD_SETS").State("wget --tries=5 --timeout=20 --quiet --output-document=- '"# url #
""");
dom.GetObject("CUXD." # CuxGeraetAbfrage # ".CMD_QUERY_RET").State(1);
stdout = dom.GetObject("CUXD." # CuxGeraetAbfrage # ".CMD_RETS").State();
! Prüfe, ob es sich um einen Parameter mit Enum-Werten handelt.
if (!stdout.Contains("FEHLER: Falscher Typ!"))
{
    ! Setzen des Systemvariabel Wertetyp und Ermitteln der Enum-Werte des Parameters
    stdout = (stdout.Substr(stdout.Find("0 - "), stdout.Length())).ToLatin();
    string value = "";
    string newvalues = "";
    integer inewvalues=0;
    foreach (value, stdout.Split("\r"))
    {
        if (value.Contains(" - "))
        {
            if (newvalues == "") { newvalues = newvalues # value.Substr(value.Find(" - ") + 3, value.Length()); }
            else { newvalues = newvalues # ";" # value.Substr(value.Find(" - ") + 3, value.Length()); }
            inewvalues = inewvalues + 1;
        }
    }

    svObject.ValueType(ivtInteger);
    svObject.ValueSubType(istEnum);
    svObject.ValueList(newvalues);
    !prüft, ob der ermittelte Wert innerhalb der möglichen Werte liegt
    if (Wert < inewvalues) { if (Wert != svObject.Value()) { if (Wert != "") { svObject.State(Wert); } } }
    else { WriteLine("Der ermittelte Wert entspricht keinem gültigen Enum-Wert. Bitte Ausgabe prüfen!") }
}
elseif (Einheit.Contains("-- Aus") || Einheit.Contains("-- Ein"))
{
    ! Setzen des Systemvariabel Wertetyp
    svObject.ValueType(ivtBinary);
    svObject.ValueSubType(istBool);
    svObject.ValueName0("Aus");
    svObject.ValueName1("Ein");
    if (Wert != svObject.Value()) { if (Wert != "") { svObject.State(Wert); } }
}
elseif (Einheit.Contains("°"))
{
    ! Setzen des Systemvariabel Wertetyp
    svObject.ValueType(ivtFloat);
    svObject.ValueSubType(istGeneric);
    svObject.ValueMin(-50);
    svObject.ValueMax(100);
    if (Wert != svObject.Value()) { if (Wert != "") { svObject.State(Wert); } }
}
elseif (Einheit.Contains("%"))
{
    ! Setzen des Systemvariabel Wertetyp
    svObject.ValueType(ivtFloat);
    svObject.ValueSubType(istGeneric);
    svObject.ValueMin(0);
    svObject.ValueMax(100);
    if (Wert != svObject.Value()) { if (Wert != "") { svObject.State(Wert); } }
}
else
{
    ! Setzen des Systemvariabel Wertetyp
    svObject.ValueType(ivtFloat);
    svObject.ValueSubType(istGeneric);
    if (Wert != svObject.Value()) { if (Wert != "") { svObject.State(Wert); } }
}
dom.RTUpdate(0); ! Interne Aktualisierung der Systemvariablen

```

```

! Logging
if (CuxGeraetLogging != "") { dom.GetObject("CUxD."#CuxGeraetLogging#.LOGIT").State(dom.GetObject(ID_SYSTEM_VARIABLES).Get(Variablename).Name()#";"#dom.GetObject(ID_SYSTEM_VARIABLES).Get(Variablename).Value()); }
}

```

**Die folgenden HomeMatic-Beispielscripte stammen vom FHEM-Forumsmitglied „PaulM“.**

**Sie sind ebenfalls [hier](#) im FHEM-Forum zu finden.**

**Vielen Dank!**

Zur Einbindung in HomeMatic bietet sich die Verwendung von CuxD und wget an.

**Beispiel zur Abfrage des Parameters ‚8326 Brennermodulation‘ mittels CuxD:**

```

! Skriptanfang:
! BSB-Abfrage
string url='http://192.168.178.88/8326'; !IP anpassen
! WriteLine("URL: " # url); !nur Kontrolle
! --timeout=seconds ! während der Dauer der Abfrage werden von der CCU
! keine anderen Skripte ausgeführt !!!
! siehe CUXD-Handbuch 5.8.2 System.Exec
! dom.GetObject("CUxD.CUX2801001:1.CMD_SETS").State
! ("wget -t 5 -T 20 -q -O - '"# url #""); ! abgekürzte Wget Syntax
dom.GetObject("CUxD.CUX2801001:1.CMD_SETS").State("wget --tries=5 --timeout=20 --quiet --output-document=- '"# url #"");
! ausführliche Wget Syntax
dom.GetObject("CUxD.CUX2801001:1.CMD_QUERY_RET").State(1);
var stdout = dom.GetObject("CUxD.CUX2801001:1.CMD_RETS").State();
! WriteLine("Antwort: " # stdout);
! Beim stdout den ueberfluessigen Vorspann entfernen
integer laenge = stdout.Length();
integer pos = stdout.Find("body");
! ab hier kommen auswertbare Informationen
pos = pos + 10;
stdout = stdout.Substr(pos, (laenge - pos));
! wenn Rückmeldung mit allen Daten angezeigt werden soll,
! Ausrufezeichen nächste Zeile entfernen
! WriteLine("Antwort ohne Vorspann: " # stdout);
string wort = "8326"; !Brennermodulation
integer laenge = wort.Length();
! WriteLine("laenge: " # laenge); zum Testen für andere Parameter
! für Skripttest Ausrufezeichen entfernen
integer pos = stdout.Find(wort);
! WriteLine("pos:" #pos);
pos = pos + 39; !bei anderen Parametern meist zwischen 25 und 50
string daten = stdout.Substr((pos + laenge +1), 100);
! WriteLine("daten :"#daten);
integer pos = daten.Find(wort);
daten = daten.Substr(0, (pos));
integer zahl = daten.ToFloat();
! keine Prüfung, da auch 0 sein kann
dom.GetObject("H_Brennermodulation").State(zahl);
WriteLine("H_Brennermodulation: "#zahl);
! nur für Chart CUxD
dom.GetObject("CUxD.CUX2801001:1.LOGIT").State
("H_Brennermodulation"#";"#zahl);
WriteLine("Hallo Welt!");
! Skriptende:

```

**Beispiel zum Setzen der Betriebsart auf Komfort-Betrieb mit ‚S700=3‘ mittels CuxD:**



```

! Skriptanfang:
! Heizung KOMFORT (=AN - keine Nachtabsenkung)
! Anweisung senden
string urlset ='http://192.168.178.88/S700=3'; !IP anpassen
WriteLine("Befehl: " # urlset);
dom.GetObject("CUxD.CUX2801001:1.CMD_SETS").State
("wget -t 5 -T 20 -q -O - '"# urlset #'");
dom.GetObject("CUxD.CUX2801001:1.CMD_QUERY_RET").State(1);
var stdout = dom.GetObject("CUxD.CUX2801001:1.CMD_RETS").State();
! Kontrollabfrage
string url='http://192.168.178.88/700'; !IP anpassen
! WriteLine("URL: " # url);
dom.GetObject("CUxD.CUX2801001:1.CMD_SETS").State
("wget -t 5 -T 20 -q -O - '"# url #'");
dom.GetObject("CUxD.CUX2801001:1.CMD_QUERY_RET").State(1);
var stdout = dom.GetObject("CUxD.CUX2801001:1.CMD_RETS").State();
! WriteLine("Antwort: " # stdout);
! Beim stdout den ueberfluessigen Vorspann entfernen
integer laenge = stdout.Length();
integer pos = stdout.Find("body");
pos = pos + 10;
stdout = stdout.Substr(pos, (laenge - pos));
! wenn Rückmeldung mit allen Daten angezeigt werden soll,
! Ausrufezeichen nächste Zeile entfernen
! WriteLine("Antwort ohne Vorspann: " # stdout);
string wort = "700"; !Heizbetrieb
integer laenge = wort.Length();
! WriteLine("laenge: " # laenge);
integer pos = stdout.Find(wort);
! WriteLine("pos:" #pos);
pos = pos + 28;
! WriteLine("pos: " # pos);
string daten = stdout.Substr((pos + laenge +1), 100);
! WriteLine("daten :"#daten);
string substring = daten.Substr(0, 1);
integer zahl = substring.ToInteger();
WriteLine("aktueller Heizbetrieb (0 bis 3): " # zahl);
if (zahl == 0) {dom.GetObject('H_Heizbetrieb').State("Heizung AUS");}
if (zahl == 1) {dom.GetObject('H_Heizbetrieb').State("Heizung Automatik");}
if (zahl == 2) {dom.GetObject('H_Heizbetrieb').State("Heizung Nachtabsenkung");}
if (zahl == 3) {dom.GetObject('H_Heizbetrieb').State("Heizung Komfort");}
! nur für Chart CUxD
!dom.GetObject("CUxD.CUX2801001:1.LOGIT").State("H_Heizbetrieb"#";"#zahl);
! alle Werte aktualisieren
var programObj = dom.GetObject("Heizungswerte abfragen");
! Programmnamen ggf. anpassen
programObj.ProgramExecute();
WriteLine("Hallo Welt!");
! Skriptende:

```

### Heizung Abfrage und speichern als Systemvariable:

Ferienbetrieb von-bis / Absenkniveau (632/633/648)

Heiz- und Warmwasserbetrieb (700/1600/8700/8326/8743/8314/8830)

Übertrag der von einem One-Wire Sensor gemessenen Raumtemperatur zum BSB (I10000)

Protokollieren (für Auswertungen mit CUxD Highcharts)

```

!Heizung Abfrage V17 mit CUxD 2018-02-17

!wget mit ausführl. Syntax 2018-02-17
!inkl. Daten Ferienbetrieb und -niveau 2018-01-31

!Prüfung of Rückgabewerte ungleich Null
!auch senden an CUxD-Geräte Tempsens_H_*

!H_Trinkwassertemperatur korrigiert +1.3 2017-08-11
!mit Übertrag der Raumtemperatur an die Heizung Parameter 8740 2017-05-13
!mit OneWire Sensoren 2017-04-26

!als Systemvariablen sind in Homematic angelegt:
!H_Ferien_Beginn Zeichenkette
!H_Ferien_Ende Zeichenkette
!H_Ferienniveau Zeichenkette

!H_Aussentemperatur      Zahl
!H_Brennermodulation     Zahl
!H_Vorlauftemperatur     Zahl
!H_Kesselruecklauftemperatur Zahl
!H_Trinkwassertemperatur Zahl
!H_Heizbetrieb          Zeichenkette
!H_Trinkwasserbetrieb    Zeichenkette

!632   Beginn Ferienbetrieb    TT.MM
!633   Ende Ferienbetrieb      TT.MM

```

```

!648 Frostschutz / Reduziert

string url='http://192.168.10.13/632/633/648/700/1600/8700/8326/8743/8314/8830/T';

!WriteLine("URL: " # url);

!--timeout=seconds ! während der Dauer der Abfrage werden von der CCU keine anderen Skripte ausgeführt
! CUXD-Handbuch 5.8.2 System.Exec "Es ist zu beachten, dass die Verarbeitung des HM-Scripts erst fortgesetzt wird, nachdem das aufge
rufene Programm beendet wurde. Während dieser Zeit werden keine anderen HM-Scripts ausgeführt!"

!dom.GetObject("CUXD.CUX2801001:1.CMD_SETS").State("wget -t 30 -q -O - '"# url #'"); ! abgekürzte Wget Syntax
dom.GetObject("CUXD.CUX2801001:1.CMD_SETS").State("wget --tries=5 --timeout=20 --quiet --output-document=- '"# url #'"); ! ausführ
liche Wget Syntax
dom.GetObject("CUXD.CUX2801001:1.CMD_QUERY_RET").State(1);
var stdout = dom.GetObject("CUXD.CUX2801001:1.CMD_RETs").State();
!WriteLine("Antwort: " # stdout);

!Beim stdout den ueberfluessigen Vorspann entfernen
integer laenge = stdout.Length();
integer pos = stdout.Find("body");
pos = pos + 10;
stdout = stdout.Substr(pos, (laenge - pos));
!wenn Rückmeldung mit allen Daten angezeigt werden soll, Ausrufezeichen nächste Zeile entfernen
!WriteLine("Antwort ohne Vorspann: " # stdout);

string wort = "632"; !Ferien Heizkreis 1 Beginn TT.MM
integer laenge = wort.Length();
!WriteLine("laenge: " # laenge);
integer pos = stdout.Find(wort);
!WriteLine("pos: " #pos);
pos = pos + 48;
!WriteLine("pos: " # pos);
string daten = stdout.Substr((pos + laenge +1), 13);
daten = daten.Substr(2,5);
!WriteLine("daten :"#daten);
dom.GetObject('H_Ferien_Beginn').State("Ferien ab: ???"); ! setzt vorherigen Wert zurück
dom.GetObject('H_Ferien_Beginn').State("Ferien ab: " # daten);
var temp = dom.GetObject('H_Ferien_Beginn').Value();
WriteLine("Ferien Heizkreis 1 Beginn TT.MM: " # temp);

string wort = "633"; !Ferien Heizkreis 1 Ende TT.MM
integer laenge = wort.Length();
!WriteLine("laenge: " # laenge);
integer pos = stdout.Find(wort);
!WriteLine("pos: " #pos);
pos = pos + 48;
!WriteLine("pos: " # pos);
string daten = stdout.Substr((pos + laenge +1), 15);
daten = daten.Substr(0,5);
!WriteLine("daten :"#daten);
dom.GetObject('H_Ferien_Ende').State("Ferien bis: ???"); ! setzt vorherigen Wert zurück
dom.GetObject('H_Ferien_Ende').State("Ferien bis: " # daten);
var temp = dom.GetObject('H_Ferien_Ende').Value();
WriteLine("Ferien Heizkreis 1 Ende TT.MM: " # temp);

string wort = "648"; !Betriebsniveau Ferien
integer laenge = wort.Length();
!WriteLine("laenge: " # laenge);
integer pos = stdout.Find(wort);
!WriteLine("pos: " #pos);
pos = pos + 48;
!WriteLine("pos: " # pos);
string daten = stdout.Substr((pos + laenge +1), 15);
daten = daten.Substr(0,12);
!WriteLine("daten :"#daten);
dom.GetObject('H_Ferienniveau').State("Ferienniveau: ???"); ! setzt vorherigen Wert zurück
dom.GetObject('H_Ferienniveau').State("Ferienniveau: " # daten);
var temp = dom.GetObject('H_Ferienniveau').Value();
WriteLine("Betriebsniveau Ferien: " # temp);

string wort = "700"; !Heizbetrieb
integer laenge = wort.Length();
!WriteLine("laenge: " # laenge);
integer pos = stdout.Find(wort);
!WriteLine("pos: " #pos);
pos = pos + 28;
!WriteLine("pos: " # pos);
string daten = stdout.Substr((pos + laenge +1), 100);
!WriteLine("daten :"#daten);
string substring = daten.Substr(0, 1);
integer zahl = substring.ToInteger();
!WriteLine("zahl: " # zahl);
if (zahl == 0) {dom.GetObject('H_Heizbetrieb').State("Heizung AUS");}
if (zahl == 1) {dom.GetObject('H_Heizbetrieb').State("Heizung Automatik");}
if (zahl == 2) {dom.GetObject('H_Heizbetrieb').State("Heizung Nachtabsenkung");}
if (zahl == 3) {dom.GetObject('H_Heizbetrieb').State("Heizung Komfort");}

```

```

! nur für Chart CUxD
!dom.GetObject("CUxD.CUX2801001:1.LOGIT").State("H_Heizbetrieb"#"#";"#zahl);

string wort = "1600"; !Trinkwasserbetrieb
integer laenge = wort.Length();
!WriteLine("laenge: " # laenge);
integer pos = stdout.Find(wort);
!WriteLine("pos:" #pos);
pos = pos + 35;
string daten = stdout.Substr((pos + laenge +1), 100);
!WriteLine("daten :"#daten);
string substring = daten.Substr(0, 1);
integer zahl = substring.ToInteger();
!WriteLine("zahl: " # zahl);
    if (zahl == 0) {dom.GetObject('H_Trinkwasserbetrieb').State("Warmwasserbetrieb - AUS");}
    if (zahl == 1) {dom.GetObject('H_Trinkwasserbetrieb').State("Warmwasserbetrieb - EIN");}

! nur für Chart CUxD
!dom.GetObject("CUxD.CUX2801001:1.LOGIT").State("H_Trinkwasserbetrieb"#"#";"#zahl);

string wort = "8700"; !Aussentemperatur
integer laenge = wort.Length();
!WriteLine("laenge: " # laenge);
integer pos = stdout.Find(wort);
!WriteLine("pos:" #pos);
pos = pos + 41;
string daten = stdout.Substr((pos + laenge +1), 100);
!WriteLine("daten :"#daten);
integer pos = daten.Find(wort);
daten = daten.Substr(0, (pos));
integer zahl = daten.ToFloat();
!Korrektur des angezeigten Wertes auf das Niveau der übrige Thermometer
if (zahl<>0)
{
    zahl = zahl - 3.5;
    dom.GetObject("H_Aussentemperatur").State(zahl);
    WriteLine("H_Aussentemperatur: "#zahl);

! nur für Chart CUxD
dom.GetObject("CUxD.CUX2801001:1.LOGIT").State("H_Aussentemperatur"#"#";"#zahl);

! für CUxD Gerät Tempsens_H_Aussen
dom.GetObject("CUxD.CUX9002012:1.SET_TEMPERATURE").State(zahl);
}

string wort = "8326"; !Brennermodulation
integer laenge = wort.Length();
!WriteLine("laenge: " # laenge);
integer pos = stdout.Find(wort);
!WriteLine("pos:" #pos);
pos = pos + 39;;
string daten = stdout.Substr((pos + laenge +1), 100);
!WriteLine("daten :"#daten);
integer pos = daten.Find(wort);
daten = daten.Substr(0, (pos));
integer zahl = daten.ToFloat();
!keine Prüfung, da auch 0 sein kann
dom.GetObject("H_Brennermodulation").State(zahl);
WriteLine("H_Brennermodulation: "#zahl);

! nur für Chart CUxD
dom.GetObject("CUxD.CUX2801001:1.LOGIT").State("H_Brennermodulation"#"#";"#zahl);

string wort = "8743"; !Vorlauftemperatur 1:
integer laenge = wort.Length();
!WriteLine("laenge: " # laenge);
integer pos = stdout.Find(wort);
!WriteLine("pos:" #pos);
pos = pos + 45;
string daten = stdout.Substr((pos + laenge +1), 100);
!WriteLine("daten :"#daten);
integer pos = daten.Find(wort);
daten = daten.Substr(0, (pos));
integer zahl = daten.ToFloat();
if (zahl<>0)
{
    dom.GetObject("H_Vorlauftemperatur").State(zahl);
    WriteLine("H_Vorlauftemperatur: "#zahl);

! nur für Chart CUxD
dom.GetObject("CUxD.CUX2801001:1.LOGIT").State("H_Vorlauftemperatur"#"#";"#zahl);

! für CUxD Gerät Tempsens_H_KesselVor
dom.GetObject("CUxD.CUX9002014:1.SET_TEMPERATURE").State(zahl);
}

```

```

string wort = "8314"; !Kesselrücklauftemperatur
integer laenge = wort.Length();
!WriteLine("laenge: " # laenge);
integer pos = stdout.Find(wort);
!WriteLine("pos:" #pos);
pos = pos + 52;
string daten = stdout.Substr((pos + laenge +1), 100);
!WriteLine("daten :"#daten);
integer pos = daten.Find(wort);
daten = daten.Substr(0, (pos));
integer zahl = daten.ToFloat();
if (zahl<>0)
{
    dom.GetObject("H_Kesselruecklauftemperatur").State(zahl);
    WriteLine("H_Kesselruecklauftemperatur: "#zahl);

    ! nur für Chart CUxD
    dom.GetObject("CUxD.CUX2801001:1.LOGIT").State("H_Kesselruecklauftemperatur"#";"#zahl);

    ! für CUxD Gerät Tempsens_H_KesselRue
    dom.GetObject("CUxD.CUX9002013:1.SET_TEMPERATURE").State(zahl);
}

string wort = "8830"; !Trinkwassertemperatur
integer laenge = wort.Length();
!WriteLine("laenge: " # laenge);
integer pos = stdout.Find(wort);
!WriteLine("pos:" #pos);
pos = pos + 48;
string daten = stdout.Substr((pos + laenge +1), 100);
!WriteLine("daten :"#daten);
integer pos = daten.Find(wort);
daten = daten.Substr(0, (pos));
integer zahl = daten.ToFloat();
!Korrektur relativ zur Kesselvorlauftemperatur
if (zahl<>0)
{
    zahl = zahl + 1.3;
    dom.GetObject("H_Trinkwassertemperatur").State(zahl);
    WriteLine("H_Trinkwassertemperatur "#zahl);

    ! nur für Chart CUxD
    dom.GetObject("CUxD.CUX2801001:1.LOGIT").State("H_Trinkwassertemperatur"#";"#zahl);

    ! für CUxD Gerät Tempsens_H_Trinkw
    dom.GetObject("CUxD.CUX9002015:1.SET_TEMPERATURE").State(zahl);
}

!Übertrag der Raumtemperatur Esszimmer ...4d6 an die Heizung als Parameter 8740

real temp = dom.GetObject("T_Innentemperatur_Esszimmer").Value();
temp = temp.ToString();

string urlset = 'http://192.168.10.13/I10000='# temp;
!WriteLine("url " # urlset);

dom.GetObject("CUxD.CUX2801001:1.CMD_SETS").State("wget -t 30 -q -O - '"# urlset #'");
dom.GetObject("CUxD.CUX2801001:1.CMD_QUERY_RET").State(1);
var stdout = dom.GetObject("CUxD.CUX2801001:1.CMD_RETS").State();
!WriteLine("stdout: " # stdout);

!Abfrage, ob Setzen des Wertes ok
string url='http://192.168.10.13/8740';

dom.GetObject("CUxD.CUX2801001:1.CMD_SETS").State("wget -t 30 -q -O - '"# url #'");
dom.GetObject("CUxD.CUX2801001:1.CMD_QUERY_RET").State(1);
var stdout = dom.GetObject("CUxD.CUX2801001:1.CMD_RETS").State();
!WriteLine("stdout: " # stdout);

string wort = "8740"; !Raumtemperatur 1
integer laenge = wort.Length();
!WriteLine("laenge: " # laenge);
integer pos = stdout.Find(wort);
!WriteLine("pos:" #pos);
pos = pos + 42;
string daten = stdout.Substr((pos + laenge +1), 5);
!WriteLine("daten :"#daten);
integer pos = daten.Find(wort);
daten = daten.Substr(0, (pos));
!integer zahl = daten.ToFloat();
WriteLine("an die Heizung übertragene Raumtemperatur (8740) "#daten #"°C");

WriteLine("Hallo Welt!");

```

Damit syntax-sichere Anweisungen von CCU an BSB gegeben werden können (wichtig z.B. auch wenn via VPN kein direkter Zugang zum BSB-Adapter möglich ist).

*Heizung AUS (= Frostschutzbetrieb):*

```
!Heizung AUS (=Frostschutzbetrieb) 2017-03-09

string urlset ='http://192.168.10.13/S700=0';
WriteLine("Befehl: " # urlset);

dom.GetObject("CUxD.CUX2801001:1.CMD_SETS").State("wget -t 30 -q -O - '"# urlset #"");
dom.GetObject("CUxD.CUX2801001:1.CMD_QUERY_RET").State(1);
var stdout = dom.GetObject("CUxD.CUX2801001:1.CMD_RETS").State();

string url='http://192.168.10.13/700';

WriteLine("URL: " # url);

!--timeout=seconds

dom.GetObject("CUxD.CUX2801001:1.CMD_SETS").State("wget -t 30 -q -O - '"# url #"");
dom.GetObject("CUxD.CUX2801001:1.CMD_QUERY_RET").State(1);
var stdout = dom.GetObject("CUxD.CUX2801001:1.CMD_RETS").State();
WriteLine("Antwort: " # stdout);

!Beim stdout den ueberfluessigen Vorspann entfernen
integer laenge = stdout.Length();
integer pos = stdout.Find("body");
pos = pos + 10;
stdout = stdout.Substr(pos, (laenge - pos));
!wenn Rückmeldung mit allen Daten angezeigt werden soll, Ausrufezeichen nächste Zeile entfernen

!Werte aktualisieren
var programObj = dom.GetObject("Heizungswerte abfragen");
programObj.ProgramExecute();

WriteLine("Antwort ohne Vorspann: " # stdout);
WriteLine("Hallo Welt!");
```

*Heizung Automatik (= AN - mit Nachtabsenkung):*

```
!Heizung Automatik (=AN - mit Nachtabsenkung) 2017-03-09

!http://192.168.10.13/Passwort/S700=0 (Schutzbetrieb)
!http://192.168.10.13/Passwort/S700=1 (Automatik)
!http://192.168.10.13/Passwort/S700=2 (Reduziert)
!http://192.168.10.13/Passwort/S700=3 (Komfort)

string urlset ='http://192.168.10.13/S700=1';
WriteLine("Befehl: " # urlset);

dom.GetObject("CUxD.CUX2801001:1.CMD_SETS").State("wget -t 30 -q -O - '"# urlset #"");
dom.GetObject("CUxD.CUX2801001:1.CMD_QUERY_RET").State(1);
var stdout = dom.GetObject("CUxD.CUX2801001:1.CMD_RETS").State();

string url='http://192.168.10.13/700';

WriteLine("URL: " # url);

!--timeout=seconds

dom.GetObject("CUxD.CUX2801001:1.CMD_SETS").State("wget -t 30 -q -O - '"# url #"");
dom.GetObject("CUxD.CUX2801001:1.CMD_QUERY_RET").State(1);
var stdout = dom.GetObject("CUxD.CUX2801001:1.CMD_RETS").State();
WriteLine("Antwort: " # stdout);

!Beim stdout den ueberfluessigen Vorspann entfernen
integer laenge = stdout.Length();
integer pos = stdout.Find("body");
pos = pos + 10;
stdout = stdout.Substr(pos, (laenge - pos));
!wenn Rückmeldung mit allen Daten angezeigt werden soll, Ausrufezeichen nächste Zeile entfernen
WriteLine("Antwort ohne Vorspann: " # stdout);

!Werte aktualisieren
var programObj = dom.GetObject("Heizungswerte abfragen");
programObj.ProgramExecute();

WriteLine("Hallo Welt!");
```

*Heizung KOMFORT (= AN - keine Nachtabsenkung):*



```
!Heizung KOMFORT (=AN - keine Nachtabenkung) 2017-03-09

!http://192.168.10.13/Passwort/S700=0 (Schutzbetrieb)
!http://192.168.10.13/Passwort/S700=1 (Automatik)
!http://192.168.10.13/Passwort/S700=2 (Reduziert)
!http://192.168.10.13/Passwort/S700=3 (Komfort)

! Anweisung senden
string urlset ='http://192.168.10.13/S700=3';
WriteLine("Befehl: " # urlset);

dom.GetObject("CUxD.CUX2801001:1.CMD_SETS").State("wget -t 30 -q -O - '"# urlset #'");
dom.GetObject("CUxD.CUX2801001:1.CMD_QUERY_RET").State(1);
var stdout = dom.GetObject("CUxD.CUX2801001:1.CMD_RETS").State();

! Kontrollabfrage
string url='http://192.168.10.13/700';

WriteLine("URL: " # url);

!--timeout=seconds

dom.GetObject("CUxD.CUX2801001:1.CMD_SETS").State("wget -t 30 -q -O - '"# url #'");
dom.GetObject("CUxD.CUX2801001:1.CMD_QUERY_RET").State(1);
var stdout = dom.GetObject("CUxD.CUX2801001:1.CMD_RETS").State();
WriteLine("Antwort: " # stdout);

!Beim stdout den ueberfluessigen Vorspann entfernen
integer laenge = stdout.Length();
integer pos = stdout.Find("body");
pos = pos + 10;
stdout = stdout.Substr(pos, (laenge - pos));
!wenn Rückmeldung mit allen Daten angezeigt werden soll, Ausrufezeichen nächste Zeile entfernen
WriteLine("Antwort ohne Vorspann: " # stdout);

!alle Werte aktualisieren
var programObj = dom.GetObject("Heizungswerte abfragen");
programObj.ProgramExecute();

WriteLine("Hallo Welt!");
```

**Heizung NACHTABSENKUNG (dauernd, d.h. auch tagsüber!):**

```
!Heizung Nachtabenkung (dauernd, d.h. auch tagsüber Nachtabenkung) 2017-03-09

!http://192.168.10.13/Passwort/S700=0 (Schutzbetrieb)
!http://192.168.10.13/Passwort/S700=1 (Automatik)
!http://192.168.10.13/Passwort/S700=2 (Reduziert)
!http://192.168.10.13/Passwort/S700=3 (Komfort)

string urlset ='http://192.168.10.13/S700=2';
WriteLine("Befehl: " # urlset);

dom.GetObject("CUxD.CUX2801001:1.CMD_SETS").State("wget -t 30 -q -O - '"# urlset #'");
dom.GetObject("CUxD.CUX2801001:1.CMD_QUERY_RET").State(1);
var stdout = dom.GetObject("CUxD.CUX2801001:1.CMD_RETS").State();

string url='http://192.168.10.13/700';

WriteLine("URL: " # url);

!--timeout=seconds

dom.GetObject("CUxD.CUX2801001:1.CMD_SETS").State("wget -t 30 -q -O - '"# url #'");
dom.GetObject("CUxD.CUX2801001:1.CMD_QUERY_RET").State(1);
var stdout = dom.GetObject("CUxD.CUX2801001:1.CMD_RETS").State();
WriteLine("Antwort: " # stdout);

!Beim stdout den ueberfluessigen Vorspann entfernen
integer laenge = stdout.Length();
integer pos = stdout.Find("body");
pos = pos + 10;
stdout = stdout.Substr(pos, (laenge - pos));
!wenn Rückmeldung mit allen Daten angezeigt werden soll, Ausrufezeichen nächste Zeile entfernen
WriteLine("Antwort ohne Vorspann: " # stdout);

!Werte aktualisieren
var programObj = dom.GetObject("Heizungswerte abfragen");
programObj.ProgramExecute();

WriteLine("Hallo Welt!");
```

## Abfrage der Tagesdurchschnitte /A bestimmter Parameter und speichern als Systemvariable:

```
!Heizung Abfrage Tagesdurchschnitte V5 2017-10-07

!Raumtemperatur Ist und Trinkwassertemperatur ergänzt
!Variablen reduziert

!als Systemvariablen sind in Homematic angelegt:
!H_Aussentemperatur_24h          Zahl    8700
!H_Brennermodulation_24h        Zahl    8326
!H_Vorlauftemperatur_24h        Zahl    8743
!H_Kesselruecklauftemperatur_24h Zahl    8314

!H_Raumtemperatur_Ist_24h        Zahl    8740
!H_Raumtemperatur_Soll_24h      ???
!H_Trinkwassertemperatur_24h    Zahl    8830

string url='http://192.168.10.13/A';

WriteLine("URL: " # url);

!--timeout=seconds

dom.GetObject("CUxD.CUX2801001:1.CMD_SETS").State("wget -t 30 -qO- '"# url #'");
dom.GetObject("CUxD.CUX2801001:1.CMD_QUERY_RET").State(1);
var stdout = dom.GetObject("CUxD.CUX2801001:1.CMD_RETS").State();
!WriteLine("Antwort: " # stdout);

!Beim stdout den ueberfluessigen Vorspann entfernen
integer laenge = stdout.Length();
integer pos = stdout.Find("body");
pos = pos + 10;
stdout = stdout.Substr(pos, (laenge - pos));
!wenn Rückmeldung mit allen Daten angezeigt werden soll, Ausrufezeichen nächste Zeile entfernen
WriteLine("Antwort ohne Vorspann: " # stdout);

!-----

string wort = "8700"; !Aussentemperatur
integer laenge = wort.Length();
!WriteLine("laenge: " # laenge);
integer pos = stdout.Find(wort);
!WriteLine("pos:" #pos);
pos = pos + 20;
string daten = stdout.Substr((pos + laenge +1), 100);
!WriteLine("daten :"#daten);
integer pos = daten.Find(wort);
daten = daten.Substr(0, (pos));
integer zahl = daten.ToFloat();
!Korrektur des angezeigten Wertes auf das Niveau der übrige Thermometer
zahl = zahl - 3.5;
dom.GetObject("H_Aussentemperatur_24h").State(zahl);
WriteLine("H_Aussentemperatur_24h: "#zahl);

! nur für Chart CUxD
!dom.GetObject("CUxD.CUX2801001:1.LOGIT").State("H_Aussentemperatur_24h"#;"#zahl);

!-----

string wort = "8326"; !Brennermodulation
integer laenge = wort.Length();
!WriteLine("laenge: " # laenge);
integer pos = stdout.Find(wort);
!WriteLine("pos:" #pos);
pos = pos + 21;
string daten = stdout.Substr((pos + laenge +1), 100);
!WriteLine("daten :"#daten);
integer pos = daten.Find(wort);
daten = daten.Substr(0, (pos));
integer zahl = daten.ToFloat();
dom.GetObject("H_Brennermodulation_24h").State(zahl);
WriteLine("H_Brennermodulation_24h: "#zahl);

! nur für Chart CUxD
!dom.GetObject("CUxD.CUX2801001:1.LOGIT").State("H_Brennermodulation_24h"#;"#zahl);

!-----

string wort = "8743"; !Vorlauftemperatur 1:
integer laenge = wort.Length();
!WriteLine("laenge: " # laenge);
integer pos = stdout.Find(wort);
!WriteLine("pos:" #pos);
pos = pos + 23;
string daten = stdout.Substr((pos + laenge +1), 100);
!WriteLine("daten :"#daten);
```

```

integer pos = daten.Find(wort);
!daten = daten.Substr(0, (pos));
integer zahl = daten.ToFloat();
dom.GetObject("H_Vorlauftemperatur_24h").State(zahl);
WriteLine("H_Vorlauftemperatur_24h: "#zahl);

! nur für Chart CUxD
!dom.GetObject("CUxD.CUX2801001:1.LOGIT").State("H_Vorlauftemperatur_24h"#";"#zahl);

!-----

string wort = "8314"; !Kesselrücklauftemperatur
integer laenge = wort.Length();
!WriteLine("laenge: " # laenge);
integer pos = stdout.Find(wort);
!WriteLine("pos: " #pos);
pos = pos + 33;
string daten = stdout.Substr((pos + laenge +1), 100);
!WriteLine("daten :"#daten);
integer pos = daten.Find(wort);
daten = daten.Substr(0, (pos));
integer zahl = daten.ToFloat();
dom.GetObject("H_Kesselruecklauftemperatur_24h").State(zahl);
WriteLine("H_Kesselruecklauftemperatur_24h: "#zahl);

! nur für Chart CUxD
!dom.GetObject("CUxD.CUX2801001:1.LOGIT").State("H_Kesselruecklauftemperatur_24h"#";"#zahl);

!-----

string wort = "8740"; !Raumtemperatur 1
integer laenge = wort.Length();
!WriteLine("laenge: " # laenge);
integer pos = stdout.Find(wort);
!WriteLine("pos: " #pos);
pos = pos + 21;
string daten = stdout.Substr((pos + laenge +1), 100);
!WriteLine("daten :"#daten);
integer pos = daten.Find(wort);
daten = daten.Substr(0, (pos));
integer zahl = daten.ToFloat();
dom.GetObject("H_Raumtemperatur_Ist_24h").State(zahl);
WriteLine("H_Raumtemperatur_Ist_24h: "#zahl);

! nur für Chart CUxD
!dom.GetObject("CUxD.CUX2801001:1.LOGIT").State("H_Raumtemperatur_Ist_24h"#";"#zahl);

!-----

string wort = "8830"; !Trinkwassertemperatur
integer laenge = wort.Length();
!WriteLine("laenge: " # laenge);
integer pos = stdout.Find(wort);
!WriteLine("pos: " #pos);
pos = pos + 28;
string daten = stdout.Substr((pos + laenge +1), 100);
!WriteLine("daten :"#daten);
integer pos = daten.Find(wort);
daten = daten.Substr(0, (pos));
integer zahl = daten.ToFloat();
dom.GetObject("H_Trinkwassertemperatur_24h").State(zahl);
WriteLine("H_Trinkwassertemperatur_24h: "#zahl);

! nur für Chart CUxD
!dom.GetObject("CUxD.CUX2801001:1.LOGIT").State("H_Trinkwassertemperatur_24h"#";"#zahl);

WriteLine("Hallo Welt!");

```

**Abfrage der angeschlossenen OneWire-Sensoren und speichern als Systemvariable:**

```

!Temperatursensoren Abfrage CUXD 2017-10-13 (Auszug)

!als Systemvariablen sind in Homematic angelegt:
!28ff99e890160456 T_Aussentemperatur_Nord
!28ffff85901604d6 T_Innentemperatur_Esszimmer

string url='http://192.168.10.13/ipwe.cgi';

!WriteLine("URL: " # url);

!--timeout=seconds

dom.GetObject("CUXD.CUX2801001:1.CMD_SETS").State("wget -t 30 --timeout=20 -q -O - '"# url #'");
dom.GetObject("CUXD.CUX2801001:1.CMD_QUERY_RET").State(1);
var stdout = dom.GetObject("CUXD.CUX2801001:1.CMD_RETS").State();
!WriteLine("Antwort: " # stdout);

!Beim stdout den ueberfluessigen Vorspann entfernen
integer laenge = stdout.Length();
integer pos = stdout.Find("body");
pos = pos + 10;
stdout = stdout.Substr(pos, (laenge - pos));
!wenn Rückmeldung mit allen Daten angezeigt werden soll, Ausrufezeichen nächste Zeile entfernen
!WriteLine("Antwort ohne Vorspann: " # stdout);

string wort = "28ff99e890160456"; !T_Aussentemperatur_Nord
integer laenge = wort.Length();
!WriteLine("laenge: " # laenge);
integer pos = stdout.Find(wort);
!WriteLine("pos:" #pos);
pos = pos + 11;
string daten = stdout.Substr((pos + laenge +1), 100);
!WriteLine("daten :"#daten);
integer pos = daten.Find(wort);
daten = daten.Substr(0, (pos));
integer zahl = daten.ToFloat();
if (zahl<>0)
{
WriteLine("T_Aussentemperatur_Nord gemessen: "#zahl);
dom.GetObject("T_Aussentemperatur_Nord").State(zahl);
! nur für Chart CUXD
dom.GetObject("CUXD.CUX2801001:1.LOGIT").State("T_Aussentemperatur_Nord"#"#"#zahl);
}

string wort = "28ffff85901604d6"; !T_Innentemperatur_Esszimmer
integer laenge = wort.Length();
!WriteLine("laenge: " # laenge);
integer pos = stdout.Find(wort);
!WriteLine("pos:" #pos);
pos = pos + 11;
string daten = stdout.Substr((pos + laenge +1), 100);
!WriteLine("daten :"#daten);
integer pos = daten.Find(wort);
daten = daten.Substr(0, (pos));
integer zahl = daten.ToFloat();
if (zahl<>0)
{
dom.GetObject("T_Innentemperatur_Esszimmer").State(zahl);
WriteLine("T_Innentemperatur_Esszimmer: "#zahl);
! nur für Chart CUXD
dom.GetObject("CUXD.CUX2801001:1.LOGIT").State("T_Innentemperatur_Esszimmer"#"#"#zahl);
}

WriteLine("Hallo Welt!");

```

## 8.4 ioBroker

**BSB-LAN-User „hacki11“ hat einen Adapter für ioBroker entwickelt, den er freundlicherweise in seinem [GitHub-Repo](#) zur Verfügung stellt.**

**Vielen Dank!**







**Die folgenden ioBroker-Beispiele stammen vom FHEM-Forumsmitglied „Thomas\_B“.**

**Vielen Dank!**

**Werte abrufen und anzeigen (exemplarisch ,Warmwasser Solltemperatur'):**

Steuerung	Status
 Heizung Automatik (AN)	WW Temperatur Soll 50,0 °C
 Heizung (AUS)	WW Temperatur IST 60,8 °C
	Vorlauftemperatur 22,5 °C

Unter ‚ioBroker Admin → Adapter‘ eine ‚Parser‘-Instanz hinzufügen:

 Parser	Dieser Adapter ermöglicht d...	url html file parser	1	1.0.5	1.0.5	MIT	    
------------------------------------------------------------------------------------------	--------------------------------	----------------------	---	-------	-------	-----	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Danach unter ‚ioBroker Admin → Instanzen‘ die hinzugefügte Adapterinstanz ‚parser.0‘ zum Konfigurieren öffnen:

 parser.0	   	Parser		warn	→3 / →11	82.3 MB
--------------------------------------------------------------------------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------	--------	-----------------------------------------------------------------------------------	------	----------	---------

Dort auf das ‚+‘ klicken, danach unter ‚Namen‘ den Namen ‚TWW\_Nennsollwert‘ vergeben. Unter ‚URL oder Dateiname‘ die IP des BSB-LAN-Adapters samt Parameternummer angeben. Anschließend auf das ‚Bearbeiten‘-Icon klicken.

Adapterkonfiguration: parser.0												
Regel hinzufügen: 												
Name	URL oder Dateiname	RegEx	Num	Rolle	Typ	Einheit	Alt	Ersatz	Faktor	Offset	Intervall	
1 Betriebsart	http://192.168.178.88/700	(\w+:\s	0	defau	stir		<input type="checkbox"/>	0			15000	    
2 Komfortsollwert	http://192.168.178.88/710	(\w+:\s	0	defau	stir	°C	<input type="checkbox"/>				180000	    
3 Reduziertssollwe	http://192.168.178.88/712	kreis 1\	0	defau	nom	°C	<input type="checkbox"/>		1	0	120000	    
4 Trinkwasserbetr	http://192.168.178.88/1600	(\w+:\s	0	defau	stir		<input type="checkbox"/>				17000	    
5 <u>TWW_Nennsollh</u>	http://192.168.178.88/1610	asser\s	0	defau	nom	°C	<input type="checkbox"/>		1	0	130000	    

Es öffnet sich die Eingabemaske, in der unter ‚RegEx‘ folgende Zeichenfolge eingegeben werden muß:

```
asser\s+--\s+TWW Nennsollwert\:\s+(\d{2}.\d)
```

Test regex: **TWW\_Nennsollwert**

Typ  
number(.)

☐ Letztes Wert

Ersatzwert

Num

Faktor


Offset

0

1

0

RegEx  
asser\s+--\s+TWW Nennsollwert\:\s+(\d{2}.\d)



Danach kann die Eingabemaske mit ‚Speichern‘ geschlossen werden.

Das Abfrageintervall kann man nach Bedarf einstellen, danach auf ‚Speichern und Schließen‘ klicken. Damit ist die Adapterkonfiguration abgeschlossen.

Unter ‚ioBroker Admin → Objekte‘ findet sich nun der Ordner ‚parser.0‘ und die unter der Instanz ‚parser.0‘ angelegten Datennamen und deren Werte:



node-red.v	States Created by node-red.v	Channel	Info		
parser.0					
Betriebsart	Betriebsart	state	state	Betriebsart: 0 - Schut	
Betriebsstunden_Brenner	Betriebsstunden Brenner	state	state	1814 h	
Brennermodulation	Brennermodulation	state	state	0 %	
Komfortsollwert	Komfortsollwert	state	state	Komfortsollwert: 19.!	
Reduziert	Reduziert	state	state	15 °C	
Rücklaufftemperatur	Rücklaufftemperatur	state	state	22 °C	
Startzähler_Brenner	Startzähler Brenner	state	state	62845	
TWW_Nennsollwert	TWW Nennsollwert	state	state	50 °C	
Trinkwasserbetrieb	Trinkwasserbetrieb	state	state	Trinkwasserbetrieb: C	
Trinkwassertemperatur_	Trinkwassertemperatur	state	state	60.5 °C	
Vorlaufftemperatur	Vorlaufftemperatur	state	state	22 °C	

Die Werte können unter VIS mittels eines ‚Basic-Number‘-Widgets mit folgenden Einstellungen angezeigt werden:

**Allgemein**

Schalter: 
  
TWW Nennsollwert

Voranstellen

HTML:

HTML anhängen (Singular):

HTML anhängen(Plural):

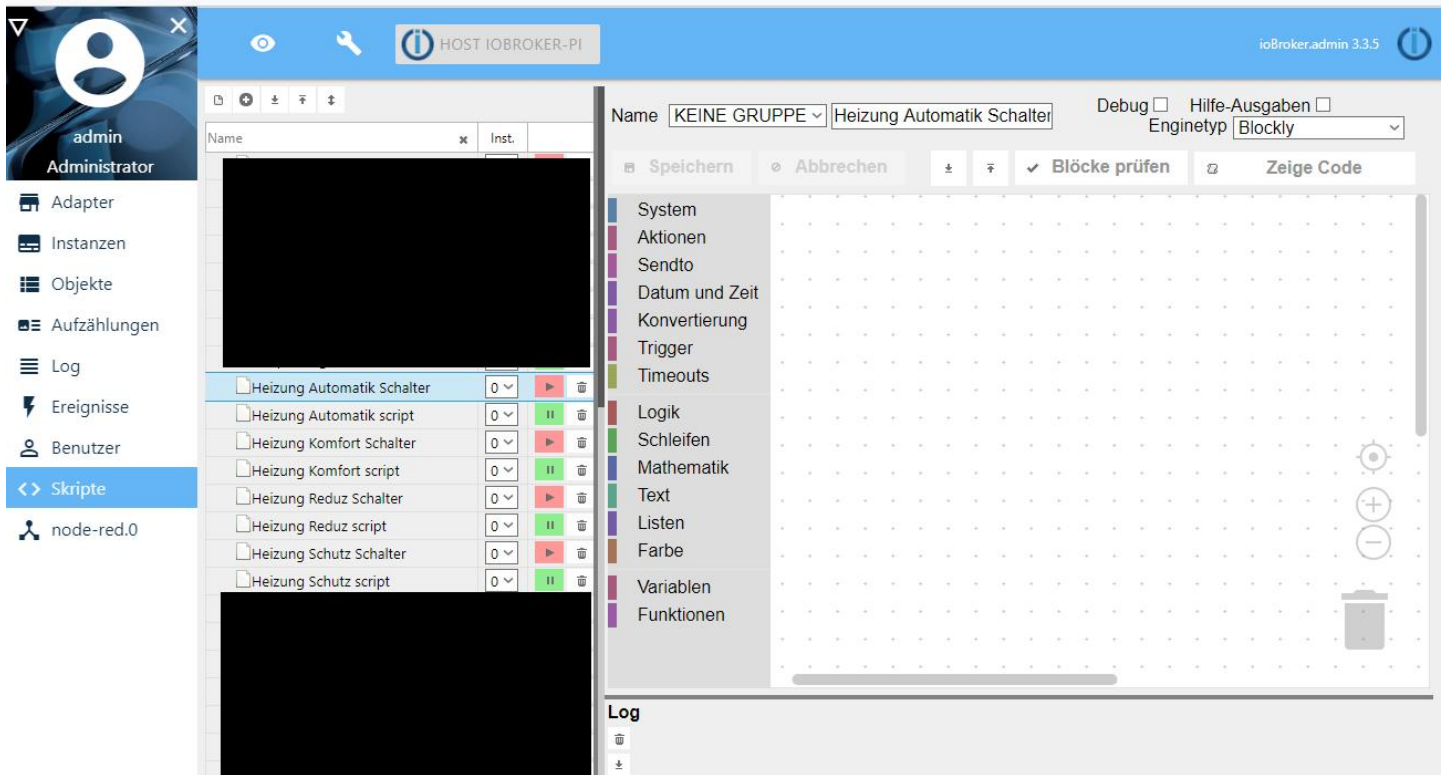
Widgetcode zum Importieren:

```
[{"tpl":"tplValueFloat","data":{"oid":"parser.0.TWW_Nennsollwert","visibility-cond":"==","visibility-val":1,"is_comma":"true","factor":"1","html_append_singular":" °C","html_append_plural":" °C","gestures-offsetX":0,"gestures-offsetY":0,"is_tdp":false,"signals-cond-0":"==","signals-val-0":true,"signals-icon-0":"/vis.0/bluefox_ehome/lowbattery.png","signals-icon-size-0":0,"signals-blink-0":false,"signals-horz-0":0,"signals-vert-0":0,"signals-hide-edit-0":false,"signals-cond-1":"==","signals-val-1":true,"signals-icon-1":"/vis.0/bluefox_ehome/lowbattery.png","signals-icon-size-1":0,"signals-blink-1":false,"signals-horz-1":0,"signals-vert-1":0,"signals-hide-edit-1":false,"signals-cond-2":"==","signals-val-2":true,"signals-icon-2":"/vis.0/bluefox_ehome/lowbattery.png","signals-icon-size-2":0,"signals-blink-2":false,"signals-horz-2":0,"signals-vert-2":0,"signals-hide-edit-2":false,"visibility-groups-action":"hide","lc-type":"last-change","lc-is-interval":true,"lc-is-moment":false,"lc-format":"","lc-position-vert":"top","lc-position-horz":"right","lc-offset-vert":0,"lc-offset-horz":0,"lc-font-size":"12px","lc-font-family":"","lc-font-style":"","lc-bkg-color":"","lc-color":"","lc-border-width":"0","lc-border-style":"","lc-border-color":"","lc-border-radius":10,"lc-zindex":0,"digits":"1"},"style":{"left":"398px","top":"428px","width":"59px","height":"18px","color":"white","text-align":"right","font-family":"Arial, Helvetica, sans-serif","font-style":"normal","font-variant":"normal","font-weight":"bold","font-size":"","z-index":"20"},"widgetSet":"basic"}]
```

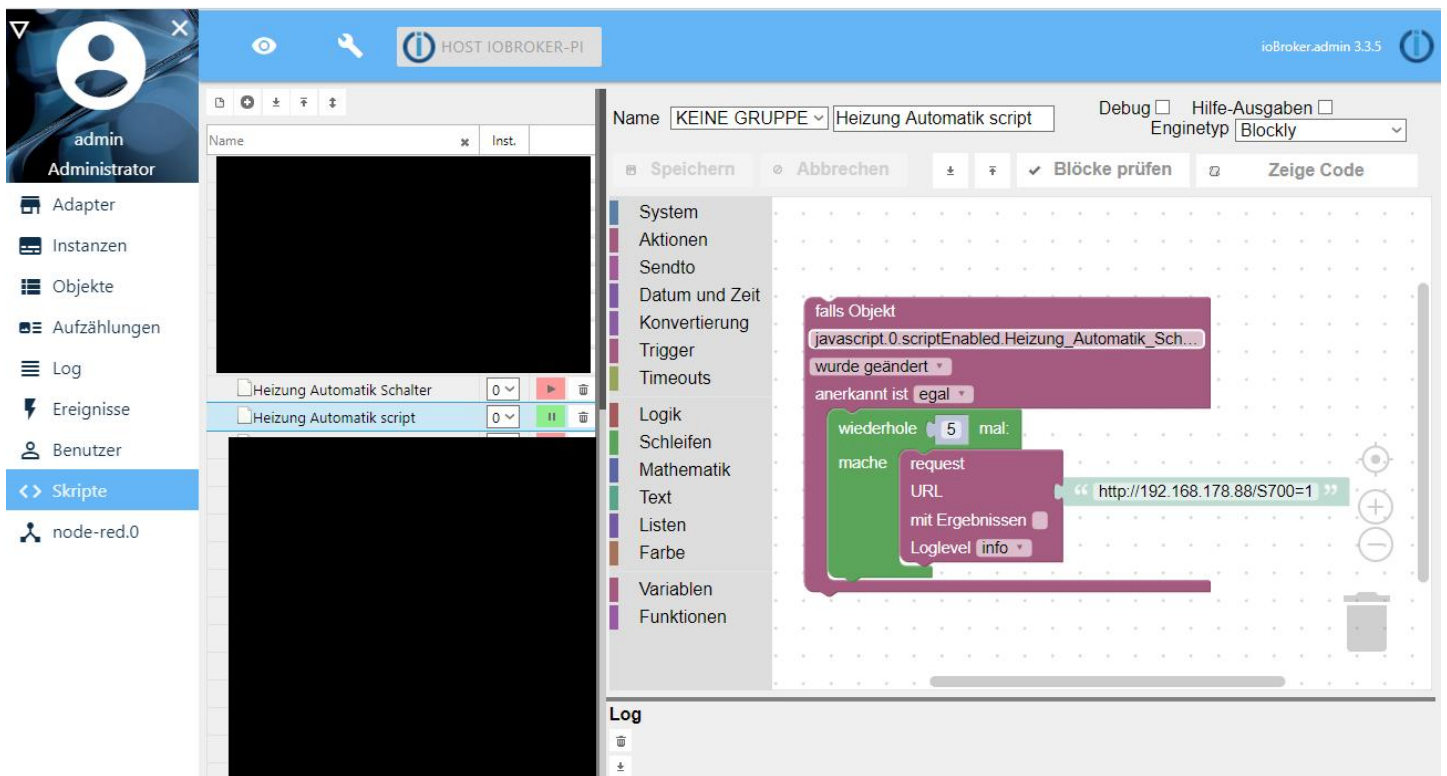
**Schalter anlegen (exemplarisch ‚Heizung Automatik (AN)‘-Schalter):**

Steuerung	Status
Heizung Automatik (AN)	WW Temperatur Soll 50,0 °C WW Temperatur IST 60,8 °C
Heizung (AUS)	Vorlaufftemperatur 22,5 °C Rücklaufftemperatur 22,5 °C
Heizung immer Komfort	Brennermodulation 0,0 %
Heizung immer Reduziert	Betriebsst. Brenner 1814 h
Warmwasser Automatik	Brennerstarts 62845

Zunächst ein leeres Script ‚Heizung Automatik Schalter‘ anlegen:



Dann ein Blockly-Script 'Heizung Automatik script' mit folgendem Inhalt anlegen (der nachfolgende Code kann in Blockly importiert werden):



```
on({id: "javascript.0.scriptEnabled.Heizung_Automatik_Schalter", change: "ne"}, function (obj) {
  var value = obj.state.val;
  var oldValue = obj.oldState.val;
  for (var count = 0; count < 5; count++) {
    try {
      require("request")('http://<IP des BSB-LAN Adapters>/S700=1').on("error", function (e) {console.error(e)});
    } catch (e) { console.error(e); }
    console.log("request: " + 'http:// ://<IP des BSB-LAN Adapters>/S700=1');
  }
});
```

Dann ein 'jquery -- Button State'-Widget in VIS anlegen:



In den Eigenschaften unter ‚Schalter‘ das anfangs angelegte leere Script angeben:

**Eigenschaften**

View w00414 CSS Skripte



**Generell**

**Sichtbarkeit**

**Allgemein**

Schalter: javascript.0.scriptEnab  
scriptEnabled.Heizung\_Aut

Damit lässt sich die Betriebsart ‚Heizung Automatik‘ einschalten.

Damit der Schalterzustand durch  oder  entsprechend visualisiert wird, müssen noch folgende Signalbilder in dem Widget hinzugefügt werden, wobei die Bilder „on.png“ und „off.png“ in dem genannten Verzeichnispfad abgespeichert werden müssen.

**Signalbilder**

Objekt ID [0]: parser.0.Betriebsart  
Betriebsart

Bedingung [0]: ==  
Wert für die Bedingung [0]: Betriebsart: 1 - Automatik

Bild [0]: /vis.0/main/img/on.png

Bildgröße in px [0]: 33

CSS Bildstil [0]:

Beschreibung [0]:

CSS Textstil [0]:

Klassen [0]:

Blinken [0]: ☐

Horizontale [0]: -1

Vertikale [0]: 6

Nicht zeigen beim Editieren [0]: ☐

---

Objekt ID [1]: parser.0.Betriebsart  
Betriebsart

Bedingung [1]: !=  
Wert für die Bedingung [1]: Betriebsart: 1 - Automatik

Bild [1]: /vis.0/main/img/off.png

Bildgröße in px [1]: 20

CSS Bildstil [1]:

Beschreibung [1]:

CSS Textstil [1]:

Klassen [1]:

Blinken [1]: ☐

Horizontale [1]: 2

Vertikale [1]: 17

Nicht zeigen bei Editieren [1]: ☐

Widgetcode zum Importieren:

```
[{"tpl": "tplJquiButtonState", "data": {"oid": "javascript.0.scriptEnabled.Heizung_Automatik_Schalter(2)", "g_fixed": false, "g_visibility": false, "g_css_font_text": true, "g_css_background": true, "g_css_shadow_padding": false, "g_css_border": false, "g_gestures": false, "g_signals": true, "g_last_change": false, "visibility-cond": "=", "visibility-val": 1, "visibility-groups-action": "hide", "buttontext": "Heizung Automatik (AN)", "signals-cond-0": "=", "signals-val-0": "Betriebsart: 1 - Automatik", "signals-icon-0": "/vis.0/main/img/on.png", "signals-icon-size-0": "33", "signals-blink-0": false, "signals-horz-0": "-1", "signals-vert-0": "6", "signals-hide-edit-0": false, "signals-cond-1": "!=", "signals-val-1": "Betriebsart: 1 - Automatik", "signals-icon-1": "/vis.0/main/img/off.png", "signals-icon-size-1": "20", "signals-blink-1": false, "signals-horz-1": "2", "signals-vert-1": "17", "signals-hide-edit-1": false, "signals-cond-2": "=", "signals-val-2": true, "signals-icon-2": "/vis/signals/lowbattery.png", "signals-icon-size-2": 0, "signals-blink-2": false, "signals-horz-2": 0, "signals-vert-2": 0, "signals-hide-edit-2": false, "lc-type": "last-change", "lc-is-interval": true, "lc-is-moment": false, "lc-format": "", "lc-position-vert": "top", "lc-position-horz": "right", "lc-offset-vert": 0, "lc-offset-horz": 0, "lc-font-size": "12px", "lc-font-family": "", "lc-font-style": "", "lc-bkg-color": "", "lc-color": "", "lc-border-width": "0", "lc-border-style": "", "lc-border-color": "", "lc-border-radius": 10, "lc-zindex": 0, "value": "", "signals-oid-1": "parser.0.Betriebsart", "signals-oid-0": "parser.0.Betriebsart"}, "style": {"left": "14px", "top": "426px", "width": "219px", "height": "27px", "z-index": "1", "font-size": "x-small"}, "widgetSet": "jquery"}
```

Die Einbindung der jeweiligen Werte bei ‚Objekt ID [0]‘ und ‚Objekt ID [1]‘ (‚parser.0.Betriebsart‘) wird nachfolgend erklärt.

#### Abfrage für die Betriebsart (für die On/Off-Visualisierung des Heizungsschalters):

Bei der Adapterkonfiguration für ‚parser.0‘ eine Regel mit der Bezeichnung ‚Betriebsart‘ erstellen, dann die IP (samt Parameternummer) des BSB-LAN-Adapters angeben und zum Bearbeiten öffnen.



Unter ‚RegEx‘ folgende Syntax angeben:

```
(\w+:\s+\d\s+-\s+\w+)
```

**Test regex: Betriebsart**

Typ

string

☐ Letztes Wert

Ersatzwert

0

Num

0

RegEx

(\w+:\s+\d\s+-\s+\w+)

## 8.5 Loxone

Die Loxone-Beispiele stammen vom FHEM-Forumsmitglied „Loxonaut“.

Vielen Dank!

#### Abfrage von Parametern:

Die Abfrage von Parametern erfolgt in Loxone mittels virtuellen HTTP Eingängen und der JSON-Funktion von BSB-LAN (URL-Befehl `/JQ=`). Eine detailliertere Beschreibung der virtuellen HTTP Eingänge und der Befehle ist in den Dokumentationen zu Loxone und im Loxone-Wiki zu finden.

*Achtung: Die Entwicklung der JSON-Funktion in BSB-LAN ist noch nicht endgültig abgeschlossen, es kann jederzeit zu Änderungen kommen. Die Konfiguration ist dann entsprechend anzupassen.*

Das folgende Beispiel zeigt die Einrichtung anhand des Parameters "8700 Außentemperatur".

Zum Hinzufügen eines virtuellen HTTP Eingangs muss zunächst im Fenster "Peripherie" die Zeile "Virtuelle Eingänge" markiert werden. Nun klickt man auf die oben erschienene Schaltfläche "Virtueller HTTP Eingang":





Bei den Eigenschaften trägt man die Bezeichnung und die entsprechenden Werte ein (beim Abfragezyklus sollte ein entsprechend sinnvoller Wert gewählt werden), die URL des BSB-LAN-Adapters ist hierbei um den Befehl

`/JQ=8700`

für die Abfrage der Außentemperatur zu erweitern:

Digitale Ausgänge  
Analoge Ausgänge  
KNX/EIB Linie  
Virtuelle Eingänge  
Fehlerstatus Gastherme (VI)  
**Außenfühler Temp Gastherme (VI)**  
Außentemperatur (VI) (Außenbereich, Wetter)

Peripherie Programm Räume Geräte

Eigenschaften (Virtueller HTTP Eingang)

Eigenschaft	Wert
Bezeichnung	Außenfühler Temp Gastherme
Beschreibung	
Objekttyp	Virtueller HTTP Eingang
<b>Einstellungen</b>	
URL	<b>http://192.168.178.88:80/JQ=8700</b>
Abfragezyklus [s]	600
Timeout [ms]	4000

Anschließend fügt man dem virtuellen HTTP Eingang noch einen virtuellen HTTP Eingang Befehl hinzu:



Hier definiert man, was aus dem JSON-Export ausgelesen werden soll. Der JSON-Export ist wie folgt aufgebaut:

```
{
  "8700": {
    "name": "Aussentemperatur", "value": "16.6",
    "unit": "&deg;C",
    "desc": "",
    "dataType": 0
  }
}
```

Mittels Loxone-Befehlserkennung

`value": "\v`

wird der Wert bei "value" des JSON-Exports ausgelesen:

Virtuelle Eingänge  
Fehlerstatus Gastherme (VI)  
Außenfühler Temp Gastherme (VI)  
**Außentemperatur (VI) (Außenbereich, Wetter)**

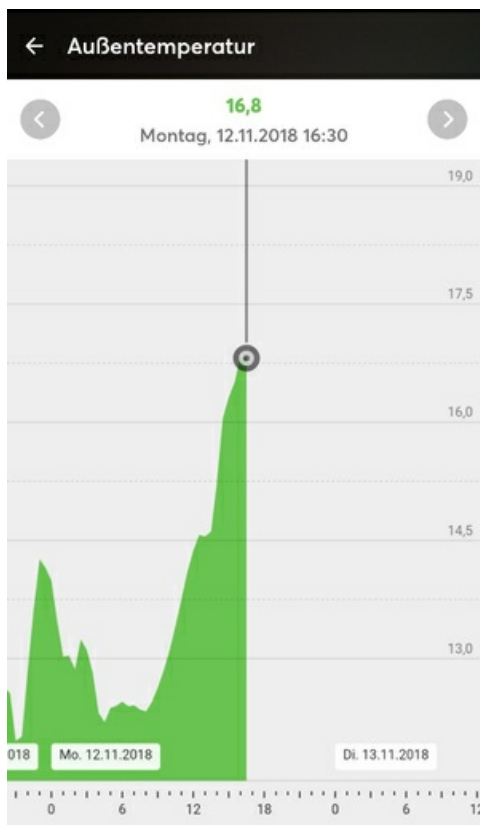
Peripherie Programm Räume Geräte

Eigenschaften (Virtueller HTTP Eingang Befehl)

Eigenschaft	Wert
Befehlserkennung	<b>value": "\v</b>
<input type="checkbox"/> Fehlerausgang anzeigen	
<input type="checkbox"/> Werteinterpretation mit Vorzeichen	



Unter "Visualisierung" bei den Eigenschaften sollte bei "Kategorie" und "Raum" jeweils eine Bezeichnung eingetragen werden, damit die spätere Darstellung in der Loxone-App entsprechend funktioniert (hier: Außenbereich, Wetter). Die nun ausgelesenen Werte des Außentemperaturfühlers können dann in der Loxone-App angezeigt und die Statistik per Graph visualisiert werden.



Hinweis:

Das Setzen von Parametern/Werten könnte analog zu obigem Beispiel mit der Funktion "virtueller Ausgang" und dem URL-Befehl `/JS` (JSON) oder via regulärem URL-Befehl `/S<x>=<y>` möglich sein (s. entspr. Kapitel), wurde allerdings noch nicht getestet.

## 8.6 IP-Symcon

Die BSB-LAN-User „sokkederheld“ und „Joachim“ haben ihre IPS-Skripte samt Screenshots im Symcon-Forum vorgestellt: [Hier](#) und [hier](#). Auf eine weitere/erneute Beschreibung wird daher an dieser Stelle verzichtet.

Vielen Dank!

## 8.7 MQTT, InfluxDB, Telegraf und Grafana

Ein vom FHEM-Forumsmitglied „cubase“ sehr gut dokumentiertes Beispiel für die Verwendung von MQTT, InfluxDB, Telegraf und Grafana findet sich [hier](#). Auf eine weitere/erneute Beschreibung wird daher an dieser Stelle verzichtet.

Vielen Dank!

BSB-LAN-User Ronald hat hierzu ebenfalls ein sehr ausführliches und gut dokumentiertes Beispiel erstellt. Es ist [hier](#) zu finden.

Vielen Dank!

## 8.8 MQTT und FHEM

Das folgende Beispiel stammt vom FHEM-Forumsmitglied „mifh“, der originale FHEM-Forumsbeitrag ist [hier](#) zu finden.

Vielen Dank!

Das folgende Beispiel nutzt den FHEM-eigenen MQTT-Server und ist für eine Gastherme samt Heizleistungsberechnung konzipiert. Letzteres ist bei Ölbrennern und Wärmepumpen hinfällig.

Bitte beachte die Anmerkungen im originalen FHEM-Forumsbeitrag (s.o.).

Auf die notwendigen Anpassungen in der Datei BSB\_lan\_config.h wird an dieser Stelle nicht weiter eingegangen, bitte beachte dazu die entspr. Punkte in Kap. [5](#)!

Parametrierung MQTT-Server in FHEM:

```
define MQTT_TST MQTT2_SERVER 1883 global
define MQTT_2 MQTT <IP-Adresse>:1883
```

Heizung als MQTT-Device in FHEM darstellen:

```
define Hzg_Therme MQTT_DEVICE
attr Hzg_Therme IODev MQTT_2
attr Hzg_Therme alias Brötje Heizung
attr Hzg_Therme group Heizung
attr Hzg_Therme room Heizung
attr Hzg_Therme subscribeReading_Kesseltemperatur Zuhause/Heizungsraum/BSB-LAN/8310
attr Hzg_Therme subscribeReading_Ruecklaufttemperatur Zuhause/Heizungsraum/BSB-LAN/8314
attr Hzg_Therme subscribeReading_Geblaesedrehzahl Zuhause/Heizungsraum/BSB-LAN/8323
attr Hzg_Therme subscribeReading_Brennermodulation Zuhause/Heizungsraum/BSB-LAN/8326
attr Hzg_Therme subscribeReading_BetriebsstundenStufel Zuhause/Heizungsraum/BSB-LAN/8330
attr Hzg_Therme subscribeReading_StartzaehlerBrenner Zuhause/Heizungsraum/BSB-LAN/8331
attr Hzg_Therme subscribeReading_BetriebsstundenHeizbetrieb Zuhause/Heizungsraum/BSB-LAN/8338
attr Hzg_Therme subscribeReading_BetriebsstundenTWW Zuhause/Heizungsraum/BSB-LAN/8339
attr Hzg_Therme subscribeReading_Gesamt_Gasenergie_Heizen Zuhause/Heizungsraum/BSB-LAN/8378
attr Hzg_Therme subscribeReading_Gesamt_Gasenergie_TWW Zuhause/Heizungsraum/BSB-LAN/8379

attr Hzg_Therme subscribeReading_Aussentemperatur Zuhause/Heizungsraum/BSB-LAN/8700
attr Hzg_Therme subscribeReading_DrehzahlHeizkreispumpe Zuhause/Heizungsraum/BSB-LAN/8735

attr Hzg_Therme stateFormat {sprintf("Leistung: %.1f kW",ReadingsVal($name,"Leistung",0))}
attr Hzg_Therme verbose 3 Hzg_Therme

define Hzg_Therme_NF1 notify Hzg_Therme:Geblaesedrehzahl.* {setHzgLeistung()}
```

Das Notify setzt mit einer Perl-Funktion in 99\_myUtils.pm das Reading Leistung:

```

sub setHzgLeistung{
  my $drehzahl=ReadingsVal("Hzg_Therme", "Geblaesedrehzahl",0);
  my $leistung;
  if ($drehzahl > 0) {
    $leistung = ($drehzahl- 1039.1)/383.1; # Heizungsspezifische Parameter
  }
  else {
    $leistung = 0;
  }
  fhem("setreading Hzg_Therme Leistung $leistung");
}

```

## 8.9 MQTT2 und FHEM

**Das folgende Beispiel stammt vom FHEM-Forumsmitglied „FunkOdyssey“, der originale FHEM-Forumsbeitrag ist [hier](#) zu finden. Vielen Dank!**

Das folgende Beispiel nutzt den FHEM-eigenen MQTT2-Server, die Readings erscheinen nach der korrekten Einrichtung automatisch.

Auf die notwendigen Anpassungen in der Datei `BSB_lan_config.h` wird an dieser Stelle nicht weiter eingegangen, bitte beachte dazu die entspr. Punkte in Kap. 5!

**Einrichten des MQTT2-Server in FHEM gemäß CommandRef:**

```

defmod mqtt2Server MQTT2_SERVER 1883 global
attr mqtt2Server autocreate 1

```

Sobald man in der Datei `BSB_LANn_config.h` die IP des FHEM-Servers angegeben hat, erscheint das MQTT2-Device mitsamt aller Readings:

```

defmod MQTT2_BSB_LAN MQTT2_DEVICE BSB_LAN
attr MQTT2_BSB_LAN IODev mqtt2Server
attr MQTT2_BSB_LAN readingList BSB_LAN:BSB/8314:.* Kesselruecklauftemperatur\
BSB_LAN:BSB/8700:.* Aussentemperatur\
BSB_LAN:BSB/8323:.* Geblaesedrehzahl\
BSB_LAN:BSB/8324:.* Brennergeblaesesollwert\
BSB_LAN:BSB/700:.* Betriebsart\
...

```

**Das folgende Beispiel stammt vom FHEM-Forumsmitglied „Luposoft“, der originale FHEM-Forumsbeitrag ist [hier](#) zu finden. Vielen Dank!**

```

define mqtt2Server MQTT2_SERVER 1883 global
define MQTT2_BSB_LAN MQTT2_DEVICE BSB_LAN
define FileLog_MQTT2 FileLog ./log/%V-%G-MQTT2.log MQTT2_BSB_LAN

```

Diese publish-Ausgabe sendet ungerichtet das MQTT-Telegramm in die Welt, ohne zu wissen, ob jemand zuhört (ein Grundprinzip von MQTT).

`set mqtt2Server publish BSB-LAN S5890=0` → Ausgabe des Steuerbefehls (manche brauchen auch ein I statt dem S) und unser Arduino hört genau auf BSB-LAN (zumindest in der Standardconfig).

Die dazugehörigen Einträge in `FileLog_MQTT2`:

`2021-02-03_16:56:28 MQTT2_BSB_LAN MQTT: ACK_S5890=0` → Hier bestätigt BSB-LAN den Empfang.

`2021-02-03_16:56:29 MQTT2_BSB_LAN BSB-LAN_5890: 0 - Kein` → Das ist die Ausgabe nach der Abfrage der Heizung.

`set mqtt2Server publish BSB-LAN 5890` → Einfache Wertabfrage.

`2021-02-04_13:24:15 MQTT2_BSB_LAN MQTT: ACK_5890` → Hier bestätigt BSB-LAN den Empfang.

`2021-02-04_13:24:16 MQTT2_BSB_LAN BSB-LAN_5890: 1 - Zirkulationspumpe Q4` → Das ist die Ausgabe nach der Abfrage der Heizung.

## 8.10 EDOMI

**Das folgende Beispiel stammt vom BSB-LAN-User Lutz. Vielen Dank!**

Die Abfrage von Werten aus BSB-LAN erfolgt mittels des erstellten [Logikbausteins 19001820](#).

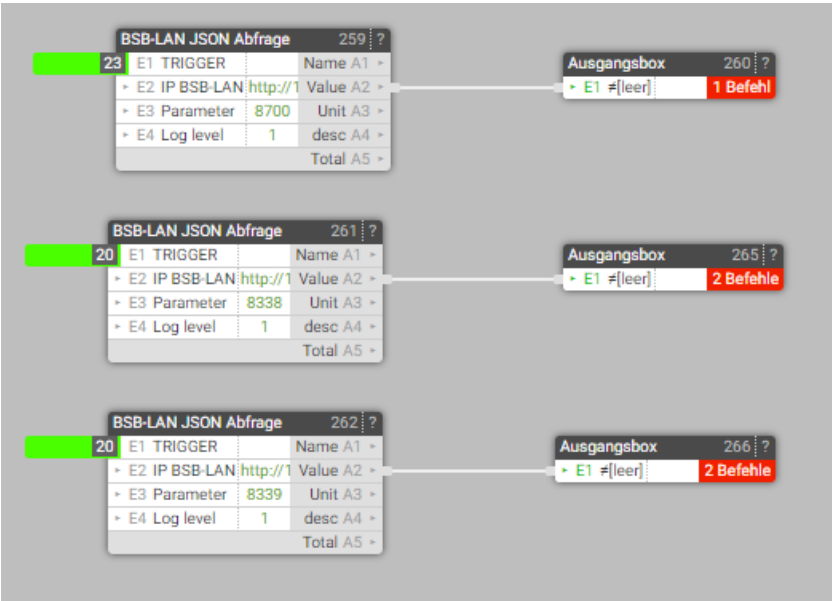
Der Baustein greift über die JSON Schnittstelle von BSB LAN auf das Gateway zu und liefert je nach angegebenem Parameter die entsprechenden Werte.

Dabei sind folgende Eingangswerte einzutragen:

- E1 = Trigger, nicht gleich Null
- E2 = IP Adresse BSB-LAN Gateway
- E3 = Parameter, z.B. Wert 8700 für Außentemperatur
- E4 = Log Level

Als Ergebnis erhält EDOMI folgende Werte zurück:

- A1 = Name des Parameters, z.B. "Aussentemperatur"
- A2 = Wert des Parameters, z.B. "10,5"
- A3 = Einheit des Parameters z.B. "°C"
- A4 = Beschreibung von A2, wenn als Code ausgegeben.
- A5 = Verkettung von A1 bis A4




Die Werte A1 bis A5 können dann über andere Bausteine weiterverarbeitet werden. In diesem Beispiel wird die Außentemperatur in ein internes Kommunikationsobjekt geschrieben, um den Inhalt z.B. in der Visu auszugeben oder die Werte für die Betriebszeit in ein Datenarchiv gespeichert, um daraus später Laufzeiten der Heizung zu ermitteln.

## 8.11 Home Assistant

**ACHTUNG**

Home Assistant ab der Version 2024.2.0 unterstützt keine selbst konfigurierten (MQTT) Entities die den Device Namen im Namen verwenden!

# Manual configured MQTT entities with a name that starts with the device name

 This stops working in version 2024.2.0. Please address before upgrading.

Some MQTT entities have an entity name that starts with the device name. This is not expected. To avoid a duplicate name the device name prefix is stripped off the entity name as a work-a-round. Please update your configuration and restart Home Assistant to fix this issue.

List of affected entities:

- climate.bsb\_lan\_heizungstherme
- sensor.bsb\_lan\_historia01\_fa\_phase

**BSB-LAN-User herr.vorragend hat eine ausführliche Beschreibung für die Einbindung in HomeAssistant via MQTT erstellt. Vielen Dank dafür!**

Die folgende Beschreibung zeigt auf, wie BSB-LAN via MQTT und ohne zusätzliche Automatisierungs-Workarounds in Home Assistant eingebunden werden kann. Diese Verfahren benötigen keine REST-API und schreibt Konfigurationsänderungen unmittelbar mit den verfügbaren Bordmitteln in den Heizungsregler.

Es ist ratsam, an dieser Stelle mit [Packages](#) zu arbeiten. Somit können sämtliche Konfigurationen aller Domänen in einer YAML-Datei verwaltet und bearbeitet werden.

Es lassen sich wie folgt viele verschiedene Einzelwerte in einer Climate-Entität zusammenfassen.

Man erkennt somit auf Anhieb die aktuelle Betriebsart, die aktuelle Soll- wie auch Ist-Temperatur. Zudem lassen sich die Werte auch direkt über die grafische Oberfläche in die Therme zurückschreiben.

```
mqtt:
  climate:
    - name: "BSB-LAN Heizungstherme"
      unique_id: bsb_lan_climate_heizungstherme
      availability_topic: "BSB/status"
      payload_on: 1
      payload_off: 0
      modes:
        - auto
        - heat
        - cool
        - "off"
      mode_state_topic: "BSB/700"
      mode_state_template: >-
        {% set values = { '0 - Schutzbetrieb':'off', '1 - Automatik':'auto', '2 - Reduziert':'cool', '3 - Komfort':'heat' } %}
        {{ values[value] if value in values.keys() else 'off' }}
      mode_command_topic: "BSB"
      mode_command_template: >-
        {% set values = { 'off':'S700=0', 'auto':'S700=1', 'cool':'S700=2', 'heat':'S700=3' } %}
        {{ values[value] if value in values.keys() else '0' }}
      current_temperature_topic: "BSB/8740"
      current_temperature_template: >-
        {% if value == '---' %}
          {{ 'None' }}
        {% else %}
          {{ value }}
        {% endif %}
      min_temp: 17
      max_temp: 24
      temp_step: 0.1
      temperature_state_topic: "BSB/710"
      temperature_command_topic: "BSB"
      temperature_command_template: "{{ 'S710=' + (value| string) }}"
      device:
        {
          identifiers: ["00000002"],
          name: "Heizung",
          model: "Arduino Due",
          manufacturer: "Github",
        }
    }
```

Für den manuellen Trinkwasserpump gibt es mehrere Möglichkeiten. Entweder über die Switch-Domain oder auch nur ein Button, der den TWW-



Push auslöst. Da es von der Therme keine Rückmeldung gibt, dürfte der Button vermutlich die bessere Lösung sein. Der Schalter wird nie wissen, in welchem Status sich der Trinkwasserpush aktuell befindet.

```
switch:
- name: "BSB-LAN Manueller TWW-Push"
  unique_id: bsb_lan_switch_manueller_tww_push
  state_topic: "BSB/10019"
  command_topic: "BSB"
  payload_on: "S10019=1"
  payload_off: "S10019=0"
  state_on: "1 - Ein"
  state_off: "0 - Aus"
  availability_topic: "BSB/status"
  device:
    {
      identifiers: ["00000002"],
      name: "Heizung",
      model: "Arduino Due",
      manufacturer: "Github",
    }

button:
- name: BSB-LAN Trinkwasserpush #war in v2.x noch 1603
  unique_id: bsb_lan_button_trinkwasserpush
  command_topic: "BSB"
  payload_press: "S10019=1"
  qos: 0
  retain: false
  availability_topic: "BSB/status"
  device:
    {
      identifiers: ["00000002"],
      name: "Heizung",
      model: "Arduino Due",
      manufacturer: "Github",
    }
```

Möchte man nicht nur die Climate-Entität nutzen, sondern auch Select-Entitäten für die Betriebsart, so geht das wie folgt:

```
select:
- name: BSB-LAN Betriebsart # Heizkreis 1
  unique_id: bsb_lan_select_betriebsart
  state_topic: "BSB/700"
  command_topic: "BSB"
  value_template: >
    {% set mapping = {0: 'Schutzbetrieb', 1: 'Automatik', 2: 'Reduziert', 3: 'Komfort'} %}
    {% set idx = value.split() | first | int %}
    {{ mapping[idx] }}
  command_template: >
    {% set mapping = {'Schutzbetrieb': 0, 'Automatik': 1, 'Reduziert': 2, 'Komfort': 3} %}
    S700={{ mapping[value] }}
  options:
    - Schutzbetrieb
    - Automatik
    - Reduziert
    - Komfort
  icon: mdi:list-box
  availability_topic: "BSB/status"
  entity_category: "config"
  device:
    {
      identifiers: ["00000002"],
      name: "Heizung",
      model: "Arduino Due",
      manufacturer: "Github",
    }
```

Zustände lassen sich sehr gut über binäre Sensoren anzeigen:

```

binary_sensor:
- name: BSB-LAN Kesselpumpe Q1
  state_topic: "BSB/8304"
  payload_on: "255 - Ein"
  payload_off: "---"
  unique_id: bsb_lan_kesselpumpe_q1
  availability_topic: "BSB/status"
  device:
    {
      identifiers: ["00000002"],
      name: "Heizung",
      model: "Arduino Due",
      manufacturer: "Github",
    }

- name: BSB-LAN Zustand Trinkwasserpumpe
  state_topic: "BSB/8820"
  payload_on: "255 - Ein"
  payload_off: "0 - Aus"
  unique_id: bsb_lan_zustand_trinkwasserpumpe
  availability_topic: "BSB/status"
  device:
    {
      identifiers: ["00000002"],
      name: "Heizung",
      model: "Arduino Due",
      manufacturer: "Github",
    }

```

Die Number-Domains ist ausgesprochen hilfreich, wenn man numerische Werte über MQTT direkt in the Therme zurückschreiben möchte:

```

number:
- name: BSB-LAN Heizkreis Komfortsollwert # Heizkreis 1 - Komfortsollwert
  unique_id: bsb_lan_heizkreis_komfortsollwert
  state_topic: "BSB/710"
  command_topic: "BSB"
  command_template: "S710={{ value }}"
  mode: slider
  min: 12
  max: 26
  step: 0.1
  unit_of_measurement: °C
  device_class: temperature
  icon: mdi:temperature-celsius
  availability_topic: "BSB/status"
  entity_category: "config"
  device:
    {
      identifiers: ["00000002"],
      name: "Heizung",
      model: "Arduino Due",
      manufacturer: "Github",
    }

- name: BSB-LAN Heizkreis Reduziertsollwert # Heizkreis 1 - Raumtemperatur-Reduziertsollwert
  unique_id: bsb_lan_heizkreis_reduziertsollwert
  state_topic: "BSB/712"
  command_topic: "BSB"
  command_template: "S712={{ value }}"
  mode: slider
  min: 12
  max: 26
  step: 0.1
  unit_of_measurement: °C
  device_class: temperature
  icon: mdi:temperature-celsius
  availability_topic: "BSB/status"
  entity_category: "config"
  device:
    {
      identifiers: ["00000002"],
      name: "Heizung",
      model: "Arduino Due",
      manufacturer: "Github",
    }

- name: BSB-LAN TWW Nennsollwert # Trinkwassertemperatur-Nennsollwert
  unique_id: bsb_lan_tww_nennsollwert
  state_topic: "BSB/1610"
  command_topic: "BSB"
  command_template: "S1610={{ value }}"
  mode: slider
  min: 40
  max: 65
  step: 0.5

```

```

step: 0.5
unit_of_measurement: °C
device_class: temperature
icon: mdi:temperature-celsius
availability_topic: "BSB/status"
entity_category: "config"
device:
  {
    identifiers: ["00000002"],
    name: "Heizung",
    model: "Arduino Due",
    manufacturer: "Github",
  }

- name: BSB-LAN TWW Reduziertersollwert # Trinkwassertemperatur-Reduziertersollwert
unique_id: bsb_lan_tww_reduziertsollwert
state_topic: "BSB/1612"
command_topic: "BSB"
command_template: "S1612={{ value }}"
mode: slider
min: 40
max: 65
step: 0.5
unit_of_measurement: °C
device_class: temperature
icon: mdi:temperature-celsius
availability_topic: "BSB/status"
entity_category: "config"
device:
  {
    identifiers: ["00000002"],
    name: "Heizung",
    model: "Arduino Due",
    manufacturer: "Github",
  }

- name: BSB-LAN TWW Nennsollwertmaximum # Trinkwassertemperatur-Nennsollwertmaximum
unique_id: bsb_lan_tww_nennsollwertmaximum
state_topic: "BSB/1614"
command_topic: "BSB"
command_template: "S1614={{ value }}"
mode: slider
min: 40
max: 65
step: 0.5
unit_of_measurement: °C
device_class: temperature
icon: mdi:temperature-celsius
availability_topic: "BSB/status"
entity_category: "config"
device:
  {
    identifiers: ["00000002"],
    name: "Heizung",
    model: "Arduino Due",
    manufacturer: "Github",
  }

- name: BSB-LAN Heizkennlinien-Steilheit
unique_id: bsb_lan_heizkennlinien_steilheit
state_topic: "BSB/720"
command_topic: "BSB"
command_template: "S720={{ value }}"
mode: slider
min: 0.1
max: 2.0
step: 0.01
availability_topic: "BSB/status"
entity_category: "config"
device:
  {
    identifiers: ["00000002"],
    name: "Heizung",
    model: "Arduino Due",
    manufacturer: "Github",
  }

- name: BSB-LAN Heizkennlinien-Parallelverschiebung
unique_id: bsb_lan_heizkennlinien_parallelverschiebung
state_topic: "BSB/721"
command_topic: "BSB"
command_template: "S721={{ value }}"
mode: slider
min: 0
max: 2
step: 0.1
unit_of_measurement: °C
device_class: temperature

```

```
availability_topic: "BSB/status"
entity_category: "config"
device:
  {
    identifiers: ["00000002"],
    name: "Heizung",
    model: "Arduino Due",
    manufacturer: "Github",
  }
}
```

Hinzufügen der üblichen Sensoren:

```
sensor:
- name: "BSB-LAN Aussentemperatur"
  state_topic: "BSB/8700"
  unique_id: bsb_lan_aussentemperatur
  unit_of_measurement: °C
  device_class: temperature
  state_class: measurement
  availability_topic: "BSB/status"
  device:
    {
      identifiers: ["00000002"],
      name: "Heizung",
      model: "Arduino Due",
      manufacturer: "Github",
    }
}

- name: "BSB-LAN Aussentemperatur gedaempft"
  state_topic: "BSB/8703"
  unique_id: bsb_lan_aussentemperatur_gedaempft
  unit_of_measurement: °C
  device_class: temperature
  state_class: measurement
  availability_topic: "BSB/status"
  device:
    {
      identifiers: ["00000002"],
      name: "Heizung",
      model: "Arduino Due",
      manufacturer: "Github",
    }
}

- name: "BSB-LAN Außentemperatur gemischt"
  state_topic: "BSB/8704"
  unique_id: bsb_lan_aussentemperatur_gemischt
  unit_of_measurement: °C
  device_class: temperature
  state_class: measurement
  availability_topic: "BSB/status"
  device:
    {
      identifiers: ["00000002"],
      name: "Heizung",
      model: "Arduino Due",
      manufacturer: "Github",
    }
}

- name: BSB-LAN Kesseltemperatur-Istwert
  state_topic: "BSB/8310"
  value_template: >-
    {% if value == '---' %}
      {{ '0.0' | float }}
    {% else %}
      {{ value }}
    {% endif %}
  unique_id: bsb_lan_kesseltemperatur_istwert
  unit_of_measurement: °C
  device_class: temperature
  state_class: measurement
  availability_topic: "BSB/status"
  device:
    {
      identifiers: ["00000002"],
      name: "Heizung",
      model: "Arduino Due",
      manufacturer: "Github",
    }
}

- name: BSB-LAN Kesseltemperatur-Sollwert
  state_topic: "BSB/8311"
  value_template: >-
    {% if value == '---' %}
      {{ '0.0' | float }}
    {% else %}
      {{ value }}
```

```

    {{ value }}
  {% endif %}
unique_id: bsb_lan_kesseltemperatur_sollwert
unit_of_measurement: °C
device_class: temperature
availability_topic: "BSB/status"
device:
  {
    identifiers: ["00000002"],
    name: "Heizung",
    model: "Arduino Due",
    manufacturer: "Github",
  }

- name: BSB-LAN Kesselschaltpunkt
state_topic: "BSB/8312"
value_template: >-
  {% if value == '---' %}
    {{ '0.0' | float }}
  {% else %}
    {{ value }}
  {% endif %}
unique_id: bsb_lan_kesselschaltpunkt
unit_of_measurement: °C
device_class: temperature
state_class: measurement
availability_topic: "BSB/status"
device:
  {
    identifiers: ["00000002"],
    name: "Heizung",
    model: "Arduino Due",
    manufacturer: "Github",
  }

- name: BSB-LAN Ruecklauftemperatur-Istwert # Kesselrücklauftemperatur
state_topic: "BSB/8314"
unique_id: bsb_lan_ruecklauftemperatur_istwert
unit_of_measurement: °C
device_class: temperature
state_class: measurement
availability_topic: "BSB/status"
device:
  {
    identifiers: ["00000002"],
    name: "Heizung",
    model: "Arduino Due",
    manufacturer: "Github",
  }

- name: BSB-LAN Trinkwassertemperatur-Istwert Oben #B3
state_topic: "BSB/8830"
unique_id: bsb_lan_trinkwassertemperatur_istwert_oben
unit_of_measurement: °C
device_class: temperature
state_class: measurement
availability_topic: "BSB/status"
device:
  {
    identifiers: ["00000002"],
    name: "Heizung",
    model: "Arduino Due",
    manufacturer: "Github",
  }

- name: BSB-LAN Trinkwassertemperatur-Sollwert
state_topic: "BSB/8831"
unique_id: bsb_lan_trinkwassertemperatur_sollwert
unit_of_measurement: °C
device_class: temperature
state_class: measurement
availability_topic: "BSB/status"
device:
  {
    identifiers: ["00000002"],
    name: "Heizung",
    model: "Arduino Due",
    manufacturer: "Github",
  }

- name: BSB-LAN Geblaesedrehzahl
state_topic: "BSB/8323"
unique_id: bsb_lan_geblaesedrehzahl
state_class: measurement
unit_of_measurement: "rpm"
availability_topic: "BSB/status"
device:
  {

```



```

        identifiers: ["00000002"],
        name: "Heizung",
        model: "Arduino Due",
        manufacturer: "Github",
    }

- name: BSB-LAN Brennergeblaesesollwert
  state_topic: "BSB/8324"
  unit_of_measurement: "rpm"
  unique_id: bsb_lan_brennergeblaesesollwert
  availability_topic: "BSB/status"
  device:
    {
      identifiers: ["00000002"],
      name: "Heizung",
      model: "Arduino Due",
      manufacturer: "Github",
    }

- name: BSB-LAN Aktuelle Geblaeseansteuerung
  state_topic: "BSB/8325"
  unique_id: bsb_lan_aktuelle_geblaeseansteuerung
  unit_of_measurement: "%"
  icon: "mdi:fire"
  state_class: measurement
  device_class: power_factor
  availability_topic: "BSB/status"
  value_template: >-
    {% if value == '---' %}
      {{ '0.0' | float }}
    {% else %}
      {{ value }}
    {% endif %}
  device:
    {
      identifiers: ["00000002"],
      name: "Heizung",
      model: "Arduino Due",
      manufacturer: "Github",
    }

- name: BSB-LAN Brennermodulation
  state_topic: "BSB/8326"
  unique_id: bsb_lan_brennermodulation
  unit_of_measurement: "%"
  icon: "mdi:fire"
  state_class: measurement
  device_class: power_factor
  availability_topic: "BSB/status"
  value_template: >-
    {% if value == '---' %}
      {{ '0.0' | float }}
    {% else %}
      {{ value }}
    {% endif %}
  device:
    {
      identifiers: ["00000002"],
      name: "Heizung",
      model: "Arduino Due",
      manufacturer: "Github",
    }

- name: BSB-LAN Raumtemperatur-Istwert
  state_topic: "BSB/8740"
  value_template: >-
    {% if value == '---' %}
      {{ 'None' }}
    {% else %}
      {{ value }}
    {% endif %}
  unique_id: bsb_lan_raumtemperatur_istwert
  unit_of_measurement: °C
  device_class: temperature
  state_class: measurement
  availability_topic: "BSB/status"
  device:
    {
      identifiers: ["00000002"],
      name: "Heizung",
      model: "Arduino Due",
      manufacturer: "Github",
    }

- name: BSB-LAN Raumtemperatur-Sollwert
  state_topic: "BSB/8741"
  value_template: >-
    {% if value == '---' %}

```

```

    {% if value == '----' %}
        {{ 'None' }}
    {% else %}
        {{ value }}
    {% endif %}
unique_id: bsb_lan_raumtemperatur_sollwert
unit_of_measurement: °C
device_class: temperature
state_class: measurement
availability_topic: "BSB/status"
device:
{
    identifiers: ["00000002"],
    name: "Heizung",
    model: "Arduino Due",
    manufacturer: "Github",
}

- name: BSB-LAN Raumtemperatur Modell
state_topic: "BSB/8742"
value_template: >-
    {% if value == '----' %}
        {{ 'None' }}
    {% else %}
        {{ value }}
    {% endif %}
unique_id: bsb_lan_raumtemperatur_modell
unit_of_measurement: °C
device_class: temperature
state_class: measurement
availability_topic: "BSB/status"
device:
{
    identifiers: ["00000002"],
    name: "Heizung",
    model: "Arduino Due",
    manufacturer: "Github",
}

- name: BSB-LAN Vorlauftemperatur-Sollwert
state_topic: "BSB/8744"
value_template: >-
    {% if value == '----' %}
        {{ '0.0' | float }}
    {% else %}
        {{ value }}
    {% endif %}
unique_id: bsb_lan_vorlauftemperatur_sollwert
unit_of_measurement: °C
device_class: temperature
state_class: measurement
availability_topic: "BSB/status"
device:
{
    identifiers: ["00000002"],
    name: "Heizung",
    model: "Arduino Due",
    manufacturer: "Github",
}

- name: BSB-LAN Schienenvorlauftemperatur-Istwert
state_topic: "BSB/8950"
unique_id: bsb_lan_schienenvorlauftemperatur_istwert
unit_of_measurement: °C
device_class: temperature
state_class: measurement
availability_topic: "BSB/status"
device:
{
    identifiers: ["00000002"],
    name: "Heizung",
    model: "Arduino Due",
    manufacturer: "Github",
}

- name: BSB-LAN Schienenvorlauftemperatur-Sollwert
state_topic: "BSB/8951"
value_template: >-
    {% if value == '----' %}
        {{ '0.0' | float }}
    {% else %}
        {{ value }}
    {% endif %}
unique_id: bsb_lan_schienenvorlauftemperatur_sollwert
unit_of_measurement: °C
device_class: temperature
state_class: measurement
availability_topic: "BSB/status"

```

```

device:
  {
    identifiers: ["00000002"],
    name: "Heizung",
    model: "Arduino Due",
    manufacturer: "Github",
  }

- name: BSB-LAN Status Heizkreis
  state_topic: "BSB/8000"
  value_template: "{{value | regex_findall_index('-[ \t]+(.*?)')}}"
  unique_id: bsb_lan_status_heizkreis
  availability_topic: "BSB/status"
  device:
    {
      identifiers: ["00000002"],
      name: "Heizung",
      model: "Arduino Due",
      manufacturer: "Github",
    }

- name: BSB-LAN Status Trinkwasserbetrieb
  state_topic: "BSB/8003"
  value_template: "{{value | regex_findall_index('-[ \t]+(.*?)')}}"
  unique_id: bsb_lan_status_trinkwasserbetrieb
  availability_topic: "BSB/status"
  device:
    {
      identifiers: ["00000002"],
      name: "Heizung",
      model: "Arduino Due",
      manufacturer: "Github",
    }

- name: BSB-LAN Status Kessel
  state_topic: "BSB/8005"
  value_template: "{{value | regex_findall_index('-[ \t]+(.*?)')}}"
  unique_id: bsb_lan_status_kessel
  availability_topic: "BSB/status"
  device:
    {
      identifiers: ["00000002"],
      name: "Heizung",
      model: "Arduino Due",
      manufacturer: "Github",
    }

- name: BSB-LAN Status Brenner
  state_topic: "BSB/8009"
  value_template: "{{value | regex_findall_index('-[ \t]+(.*?)')}}"
  unique_id: bsb_lan_status_brenner
  availability_topic: "BSB/status"
  device:
    {
      identifiers: ["00000002"],
      name: "Heizung",
      model: "Arduino Due",
      manufacturer: "Github",
    }

- name: BSB-LAN Drehzahl Kesselpumpe
  state_topic: "BSB/8308"
  value_template: >-
    {% if value == '---' %}
      {{ '0.0' | float }}
    {% else %}
      {{ value }}
    {% endif %}
  unique_id: bsb_lan_drehzahl_kesselpumpe
  state_class: measurement
  unit_of_measurement: "%"
  availability_topic: "BSB/status"
  device:
    {
      identifiers: ["00000002"],
      name: "Heizung",
      model: "Arduino Due",
      manufacturer: "Github",
    }

# =====
# FEHLERHISTORIE 1
# =====

- name: BSB-LAN Historie01 DatumZeit
  state_topic: "BSB/6800"
  unique_id: bsb_lan_historie01_datumzeit
  availability_topic: "BSB/status"

```

```

availability_topic: "BSB/status"
device:
  {
    identifiers: ["00000002"],
    name: "Heizung",
    model: "Arduino Due",
    manufacturer: "Github",
  }

- name: BSB-LAN Historie01 Fehlercode
state_topic: "BSB/6803"
unique_id: bsb_lan_historie01_fehlercode
availability_topic: "BSB/status"
device:
  {
    identifiers: ["00000002"],
    name: "Heizung",
    model: "Arduino Due",
    manufacturer: "Github",
  }

- name: BSB-LAN Historie01 SW Diagnosecode
state_topic: "BSB/6805"
unique_id: bsb_lan_historie01_sw_diagnosecode
availability_topic: "BSB/status"
device:
  {
    identifiers: ["00000002"],
    name: "Heizung",
    model: "Arduino Due",
    manufacturer: "Github",
  }

- name: BSB-LAN Historie01 FA Phase
state_topic: "BSB/6806"
unique_id: bsb_lan_historie01_faphase
availability_topic: "BSB/status"
device:
  {
    identifiers: ["00000002"],
    name: "Heizung",
    model: "Arduino Due",
    manufacturer: "Github",
  }

- name: BSB-LAN SW Diagnosecode
state_topic: "BSB/6705"
unique_id: bsb_lan_sw_diagnosecode
availability_topic: "BSB/status"
device:
  {
    identifiers: ["00000002"],
    name: "Heizung",
    model: "Arduino Due",
    manufacturer: "Github",
  }

- name: BSB-LAN Brennerstarts Stufe 1 # Startzähler 1.Stufe
unique_id: bsb_lan_brennerstarts_stufe1
state_topic: "BSB/8331"
availability_topic: "BSB/status"
device:
  {
    identifiers: ["00000002"],
    name: "Heizung",
    model: "Arduino Due",
    manufacturer: "Github",
  }

```

Neben den obigen MQTT-Sensoren könnte man noch die Spreizung über einen Template-Sensor berechnen:

```

template:
- sensor:
  - name: bsb_lan_temperaturspreizung
    unique_id: bsb_lan_temperaturspreizung
    state: "{ (states('sensor.bsb_lan_kesseltemperatur_istwert') | float(0) - states('sensor.bsb_lan_ruecklauftemperatur_istwert') | float(0) ) | round(1) }"
    unit_of_measurement: °C
    device_class: temperature

```

Hinweis am Rande:

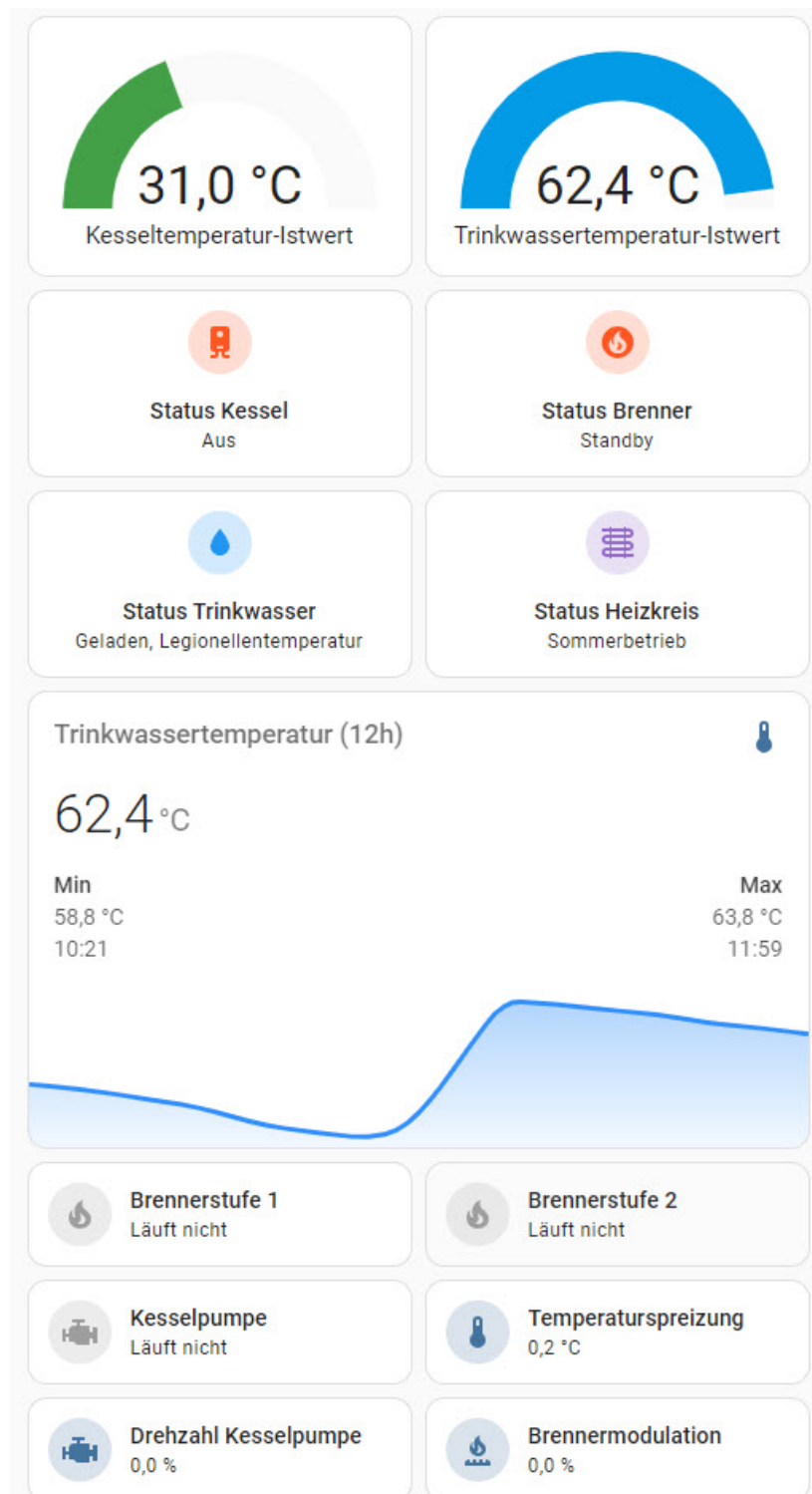
Bei allen Konfigurationen wurde hier folgende Zeilen ergänzt. Das sieht ungewöhnlich aus und verursacht nur viele redundanten Code-Zeilen. Es

hat aber den Vorteil, dass sämtliche Entitäten in Home Assistant zu einem Device zusammengeführt werden.

```
device:
  {
    identifiers: ["00000002"],
    name: "Heizung",
    model: "Arduino Due",
    manufacturer: "Github",
  }
```

Man kann es also löschen, aber dann gibt es nur viele Einzelentitäten.

Die folgenden Screenshots zeigen die Darstellung in HomeAssistant.



×

Heizungstherme

⚙

⋮

🟢

Heizungstherme  
Vorgestern

Auto 20 °C

Aktuell: 24,9 °C

Zieltemperatur

20 °C

^

v

Aktion

Auto

▼

🟢

Heizungstherme

Auto - 24,9 °C

🌡

Raummessung

24,9 °C

🌡

Raumtemperatur-Sollwert

20,0 °C

Verteiler Fußbodenheizung

🌡	Erdgeschoss Vor 4 Stunden	Vorlauf 22,9 °C	Rücklauf 22,4 °C	Spreizung 0,5 °C
🌡	Obergeschoss Vor 23 Minuten	Vorlauf 25,3 °C	Rücklauf 25,0 °C	Spreizung 0,3 °C

Verlauf

26. Juni 2023

💧

Status Trinkwasserbetrieb wechselte zu Geladen,  
Legionellentemperatur  
11:11:07 - Vor 6 Stunden

💧

Status Trinkwasserbetrieb wechselte zu Geladen,  
Nenntemperatur  
07:47:07 - Vor 9 Stunden

25. Juni 2023





💧

Status Trinkwasserbetrieb wechselte zu Geladen,  
Legionellentemperatur  
17:55:08 - Gestern



## Fehlerhistorie

### Aktuellster Fehler Nr.1



	Zeitpunkt	10.03.2023 14:50:00
	FA Phase	16
	Fehlercode	83 - 83:BSB-Draht Kurzschluss / keine Kommunikation
	SW Diagnosecode	---

Weitere



## Therme


 Heizungstherme **Auto 20 °C**  
Aktuell: 24,9 °C

 Betriebsart  
Automatik 

 Raumtemperatur-Sollwert 20,0 °C

Heizkreis 

°C Komfortsollwert  20,0 °C


°C Reduziertsollwert  18,0 °C

 Kennlinie-Steilheit  0,5

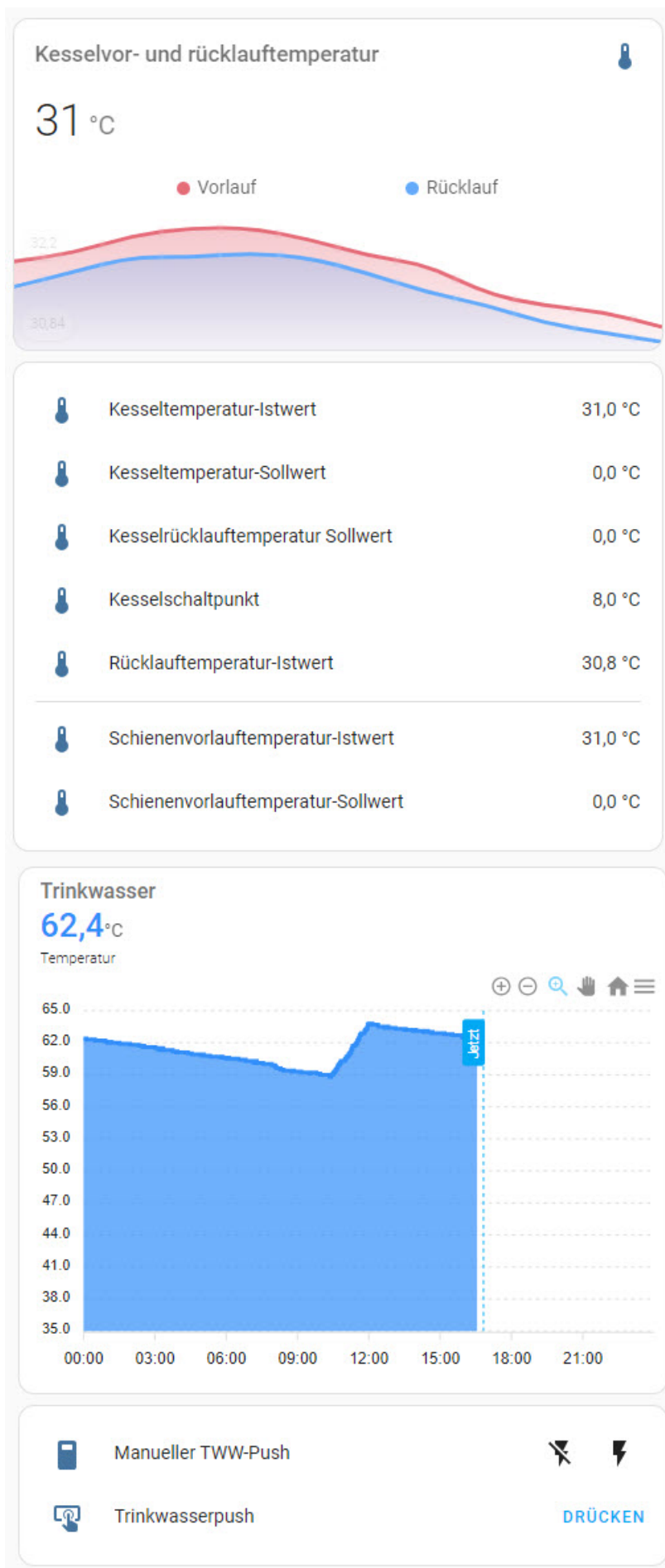
 Kennlinie-Verschiebung  0,0 °C

 Trinkwassertemperatur-Sollwert 55,0 °C

Trinkwasser 

°C Trinkwasser Nennsollwert  55,0 °C

°C Trinkwasser Reduziertsollw...  45,0 °C



BSB-LAN-User Yann hat eine ausführliche Beschreibung für die Einbindung in HomeAssistant in Verbindung mit Mosquitto erstellt (Englisch), sie ist [hier](#) zu finden.

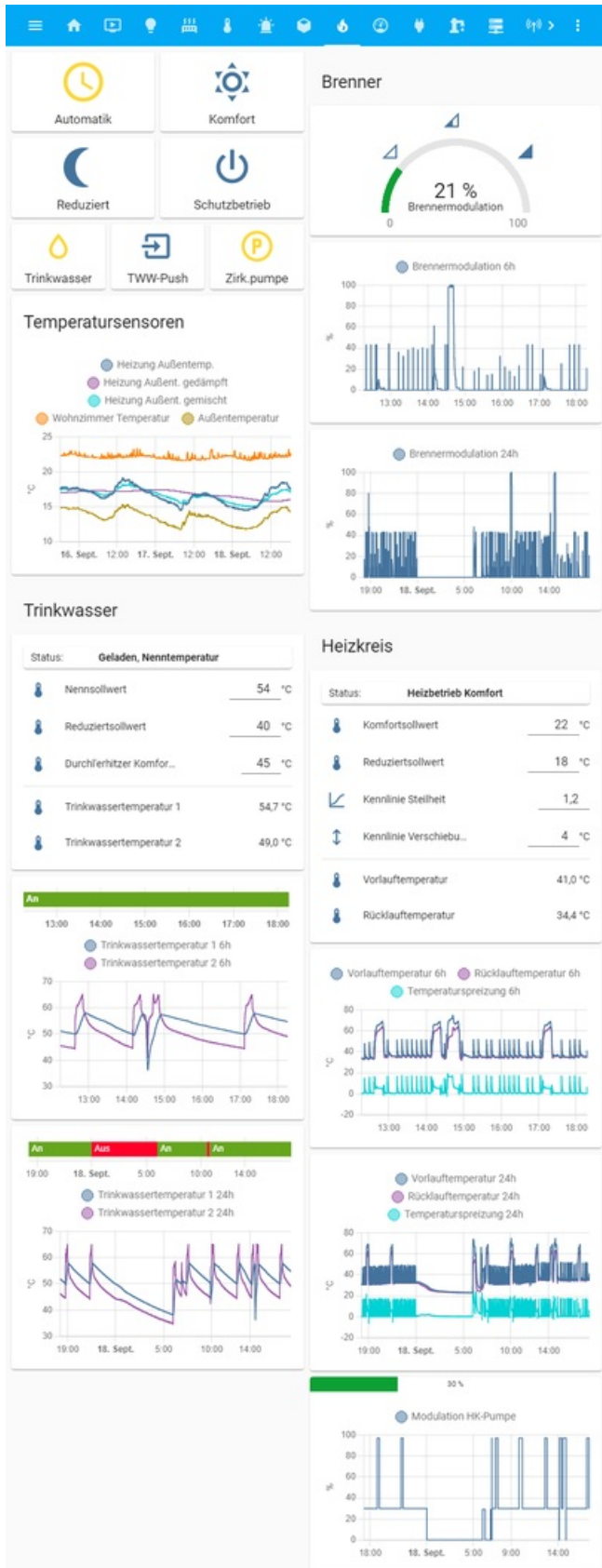
Dies Beschreibung ist ebenfalls [in Französisch](#) verfügbar.

Vielen Dank!

**BSB-LAN-User tiger42** hat im [HomeAssistant-Forum](#) eine Einbindungsmöglichkeit mittels **JSON** und **MQTT** beschrieben. Darüber hinaus hat er ein Einbindungsbeispiel für dieses Handbuch geschrieben, das im Folgenden dargestellt wird. Vielen Dank!

Die folgenden Beispiele sollen zeigen wie es möglich ist, BSB-LAN individuell in Home Assistant einzubinden, indem man Sensor- und Schalter-Entitäten selber definiert. Wie immer wenn es um Home Assistant geht, gilt auch hier: Viele Wege führen zum Ziel, es gibt nicht die "eine" beste Lösung. Deshalb sind alle Beispiele als Anregung zum selber Weiterentwickeln zu sehen.

Ein umfangreiches BSB-LAN Dashboard in Home Assistant könnte z.B. so aussehen:



Aller folgender Code muss in die YAML-Datei *configuration.yaml* eingefügt werden, wenn nicht anders angegeben. Hat man seine Konfiguration aufgesplittet und z.B. die Sensordefinitionen in eine Datei *sensors.yaml* ausgelagert, sind die Anpassungen natürlich entsprechend dort vorzunehmen.

### MQTT-Sensor

Das Auslesen von Daten per MQTT empfiehlt sich für alle Werte, die sich laufend ändern, wie z.B. Temperaturwerte. Voraussetzung dafür ist natürlich, dass man einen MQTT Broker einsetzt und die auszulesenden Werte auch per MQTT gepublikt werden.

Beispiel für einen Sensor, der die Vorlauftemperatur des HK ausliest:

```
mqtt:
  sensor:
    - state_topic: "bsb-lan/8310"
      name: BSB-LAN Vorlauftemperatur
      unit_of_measurement: °C
      device_class: temperature
```

Dieser Sensor wird in Home Assistant unter dem Namen *sensor.bsb\_lan\_vorlauftemperatur* erscheinen.

### REST-Sensor & JSON

Wenn man MQTT nicht nutzen will oder kann, lassen sich Werte auch mittels REST-Sensor und JSON auslesen.

Die folgende Sensordefinition erzeugt einen Sensor zum Auslesen verschiedener Heizungsparameter, welche sich selten oder fast nie ändern (Betriebsart, Komfortsollwert etc.) und als Block abgefragt werden. Der Zustand (Wert) des Sensors enthält in diesem Beispiel den "SW Diagnosecode", alle weiteren Werte werden als Attribute des Sensors gesetzt. Der Sensor macht alle 60 Sekunden einen Request gegen BSB-LAN.

```
sensor:
  - platform: rest
    name: BSB-LAN Status
    resource: "http://<BSB-LAN-IP>/JQ=700,1600,8000,8003,6705,710,712,720,721,1610,1612,5400"
    method: POST
    username: !secret bsb_lan_user
    password: !secret bsb_lan_pass
    scan_interval: 60
    value_template: "{{ value_json['6705'].value }}"
    json_attributes:
      - "700"
      - "1600"
      - "6705"
      - "8000"
      - "8003"
      - "710"
      - "712"
      - "720"
      - "721"
      - "1610"
      - "1612"
      - "5400"
```

Der Sensor taucht in Home Assistant unter dem Namen *sensor.bsb\_lan\_status* auf.

Um die Attribute dieses Sensors wiederum als separate Sensoren verfügbar zu machen, die sich komfortabel in die Oberfläche integrieren lassen, sind weitere Definitionen notwendig. Im folgenden Beispiel anhand der Parameter 700 (Betriebsart) und 1610 (TWW Nennsollwert) gezeigt:

```

sensor:
  - platform: template
    sensors:
      bsb_lan_betriebsart:
        unique_id: bsb_lan_betriebsart
        friendly_name: BSB-LAN Betriebsart
        value_template: "{% if states('sensor.bsb_lan_status') is defined and states('sensor.bsb_lan_status') != 'unavailable' %}{{
state_attr('sensor.bsb_lan_status', '700')['value'] }}{% else %}{{ states('sensor.bsb_lan_betriebsart') }}{% endif %}"
        attribute_templates:
          desc: "{% if states('sensor.bsb_lan_status') is defined and states('sensor.bsb_lan_status') != 'unavailable' %}{{ state_attr('sensor.bsb_lan_status', '8000')['desc'] }}{% else %}{{ state_attr('sensor.bsb_lan_betriebsart', 'desc') }}{% endif %}"
      bsb_lan_tww_nennsollwert:
        unique_id: bsb_lan_tww_nennsollwert
        friendly_name: BSB-LAN TWW Nennsollwert
        value_template: "{% if states('sensor.bsb_lan_status') is defined and states('sensor.bsb_lan_status') != 'unavailable' %}{{
state_attr('sensor.bsb_lan_status', '1610')['value'] }}{% else %}{{ states('sensor.bsb_lan_tww_nennsollwert') }}{% endif %}"
        unit_of_measurement: °C
        device_class: temperature

```

Die *if* Abfragen im Code sorgen dafür, dass die Sensoren ihren vorigen Wert behalten, auch wenn der "BSB-LAN Status" Sensor einmal kurzzeitig nicht verfügbar ist (z.B. beim Neustart von HA). Obiges Beispiel würde in Home Assistant die Sensoren *sensor.bsb\_lan\_betriebsart* und *sensor.bsb\_lan\_tww\_nennsollwert* erzeugen.

### Setzen von Parametern per REST & JSON

Für das Setzen von Werten empfiehlt es sich, zuerst ein allgemeines parametrisierbares RESTful Command zu definieren:

```

rest_command:
  bsb_lan_set_parameter:
    url: http://<BSB-LAN-IP>/JS
    method: POST
    username: !secret bsb_lan_user
    password: !secret bsb_lan_pass
    # Parameter "type": 1 = SET (default), 0 = INF
    payload: '{"Parameter": "{{ parameter }}", "Value": "{{ value }}", "Type": "{% if type is defined %}{{ type }}{% else %}1{% endif %}"}'

```

Dies erzeugt einen Service mit dem Namen *rest\_command.bsb\_lan\_set\_parameter*. Dieser Service lässt sich nun zum Setzen beliebiger Parameter nutzen.

Folgendes Beispiel erzeugt einen Schalter, mit dem man die Automatik-Betriebsart der Heizung an- und ausschalten kann:

```

switch:
  - platform: template
    switches:
      bsb_lan_betriebsart_automatik:
        friendly_name: BSB-LAN Betriebsart Automatik
        value_template: "{{ is_state('sensor.bsb_lan_betriebsart', '1') }}"
        turn_on:
          service: rest_command.bsb_lan_set_parameter
          data:
            parameter: 700
            value: 1
        turn_off:
          service: rest_command.bsb_lan_set_parameter
          data:
            parameter: 700
            value: 0

```

In Home Assistant ist dieser Schalter nun als *switch.bsb\_lan\_betriebsart\_automatik* nutzbar. Wird er aktiviert, wird für den Parameter 700 der Wert 1 ("Automatik") gesetzt. Deaktivieren setzt den Wert 0 ("Schutzbetrieb"). Wie man sieht, nutzt der Switch den weiter oben definierten Sensor *sensor.bsb\_lan\_betriebsart*, um seinen aktuellen Zustand (an/aus) zu ermitteln.

Folgender Code erzeugt zwei Automatisierungen, die man als Basis für ein Eingabefeld für den TWW Nennsollwert nutzen kann. Das Eingabefeld zeigt natürlich auch den aktuell eingestellten Wert an. **Achtung:** Der Code muss in die Datei *automations.yaml* eingefügt werden! Das Eingabefeld muss man zuvor per YAML definiert oder manuell in der Oberfläche unter "Einstellungen/Geräte & Dienste/Helfer" angelegt haben:

## EINSTELLUNGEN

## VERWANDTE

Name

BSB-LAN TWW Nennsollwert Input

Symbol

mdi:thermometer



Minimaler Wert

20

Maximaler Wert

65

Anzeigemodus



Schieberegler



Eingabefeld

Schrittgröße

0,5

Maßeinheit

°C

Entitäts-ID

input\_number.bsb\_lan\_tww\_nennsollwert

Bereich



Entität aktivieren



Deaktivierte Entitäten werden nicht zu Home Assistant hinzugefügt.

Hinweis: Dies funktioniert möglicherweise noch nicht bei allen Integrationen.

LÖSCHEN

AKTUALISIEREN

Alternativ das gleiche als YAML:

```
input_number:
  bsb_lan_tww_nennsollwert:
    name: BSB-LAN TWW Nennsollwert Input
    icon: mdi:thermometer
    mode: box
    min: 20
    max: 65
    step: 0.5
    unit_of_measurement: °C
```

Datei *automations.yaml*:



```

- id: bsb_lan_set_tww_nennsollwert
  alias: BSB-LAN TWW Nennsollwert setzen
  trigger:
    - platform: state
      entity_id: input_number.bsb_lan_tww_nennsollwert
  condition: []
  action:
    - data_template:
        parameter: 1610
        value: "{{ states('input_number.bsb_lan_tww_nennsollwert') }}"
      service: rest_command.bsb_lan_set_parameter
- id: bsb_lan_get_tww_nennsollwert
  alias: BSB-LAN TWW Nennsollwert auslesen
  trigger:
    - platform: state
      entity_id: sensor.bsb_lan_tww_nennsollwert
  condition: []
  action:
    - data_template:
        entity_id: input_number.bsb_lan_tww_nennsollwert
        value: "{{ states('sensor.bsb_lan_tww_nennsollwert') | float(0) }}"
      service: input_number.set_value

```

Der erste Trigger reagiert auf eine Änderung des Eingabefeldes und setzt entsprechend den Heizungsparamater mit Hilfe des oben definierten REST Commands. Der zweite Trigger reagiert auf eine Änderung des Parameters seitens der Heizung und aktualisiert entsprechend den Inhalt des Eingabefeldes.

**BSB-LAN ist jetzt eine offizielle Home-Assistant-Integration, die von [Willem-Jan](#) betreut wird.**

**Die Dokumentation der Integration ist [hier](#) zu finden.**

**Vielen Dank!**

**BSB-LAN-User Florian hat die o.g. Lösung erweitert, so dass zwei Regler mit insgesamt vier Heizkreisen bedient werden können. [Hier](#) hat er seine Lösung zur Verfügung gestellt.**

**Vielen Dank!**

#### Achtung

Die o.g. Integration ist z.Zt. nur mit der BSB-LAN-Version 1.0 kompatibel, NICHT mit der aktuellen Version! Für die Einbindung der aktuellen BSB-LAN-Version ist es daher empfehlenswert, die im Folgenden dargestellten Einbindungsmöglichkeiten mittels MQTT oder JSON zu nutzen.

## 8.12 SmartHomeNG

**BSB-LAN-User Thomas hat ein Plugin für SmartHomeNG geschrieben und in [seinem GitHub Repo](#) zur Verfügung gestellt.**

**Vielen Dank!**

## 8.13 Node-RED

**BSB-LAN-User Konrad hat ein [Modul für Node-RED](#) geschrieben, das eine einfache Einbindung von BSB-LAN ermöglicht.**

**Vielen Dank!**

**BSB-LAN-User n300 hat eine "Queueing-Flow"-Erweiterung für das o.g. Node-RED-Modul geschrieben. Hierdurch werden Timeouts und/oder Connection Refuses (ECONNECTREFUSE) weitestgehend vermieden, die durch zeitgleiche Requests entstehen, da Requests nun zeitlich geordnet nach einander abgearbeitet werden. Der Flow und eine weitergehende Beschreibung sind [hier](#) zu finden.**

**Vielen Dank!**

## 8.14 Datenverarbeitung mittels Bash-Skript

**BSB-LAN-User Karl-Heinz hat zwei Bash-Skripte geschrieben, die unter Linux dazu verwendet werden können, Daten vom Heizungsregler auszulesen und mittels gnuplot graphisch darzustellen.**

**Vielen Dank!**

Die Lösung von Karl-Heinz ist für diejenigen Nutzer interessant, die keine komplexe Heimautomatisierungssoftware einsetzen (wollen), um Daten unter Linux vom Heizungsregler abzurufen und graphisch darstellen zu lassen. Die Skripte stellt er freundlicherweise in [seinem GitHub-Repo](#) zur Verfügung.

## 8.15 Volkszaehler

---

*BSB-LAN-User Michael hat ein Script geschrieben, mit dem sich die Daten von BSB-LAN in das [Volkszaehler-Projekt](#) einbinden lassen. Er stellt das Script samt einer Beschreibung in [seinem GitHub-Repo](#) zur Verfügung.*  
*Vielen Dank!*

## 8.16 Homebridge

---

*BSB-LAN-User Michael hat ein [Plugin für Homebridge](#) geschrieben, das eine einfache Einbindung von BSB-LAN ermöglicht.*  
*Vielen Dank!*

## 8.17 Jeedom

---

*BSB-LAN-User [bernard-dandrea](#) hat ein [Plugin für Jeedom](#) geschrieben (Beschreibung auf Französisch), das eine einfache Einbindung von BSB-LAN ermöglicht.*  
*Vielen Dank!*



Support me on Ko-fi

[Weiter zu Kapitel 9](#)

[Zurück zum Inhaltsverzeichnis](#)

## 9. Exkurs: Auslesen neuer Parameter-Telegramme

[Zurück zum Inhaltsverzeichnis](#)

[Zurück zu Kapitel 8](#)

## 9. Exkurs: Auslesen neuer Parameter-Telegramme

In diesem Kapitel wird die Vorgehensweise des Auslesens neuer Parameter-Telegramme beschrieben. Es ist dabei in eine ausführliche Beschreibung für Einsteiger, eine kurzgefasste Beschreibung für erfahrene Nutzer sowie eine Beschreibung des Implementierens neuer Command IDs für Nutzer mit Programmierkenntnissen auf gegliedert.

Damit dein Einsatz von Erfolg gekrönt ist und verwertbare Daten liefert, beachte bitte das Folgende: Sollte ein Parameter mehrere Einstellungsoptionen bieten, so wären idealerweise alle Optionen durchzugehen und die einzelnen Telegramme mit der eindeutigen Bezeichnung der jeweiligen Option zu notieren. Erfahrungsgemäß muss hierfür jedoch jede Option einmal mittels „OK“ der Bedieneinheit bestätigt und somit eingestellt werden, nach Beendigung sollte dann wieder die ursprüngliche Einstellung vorgenommen werden. Solltest du dir bzgl. des Verstellens bestimmter Reglereinstellungen hinsichtlich möglicher Folgen unsicher sein, belasse es bitte im Zweifelsfall beim Auslesen der aktuell eingestellten Option.

Für den Fall, dass mehrere Einstellungsoptionen verfügbar sind, ist es jedoch wichtig, dass in jedem Fall die aufgezählte Reihenfolge der Optionen mit der in der Bedieneinheit angezeigten Reihenfolge übereinstimmt und die aktuelle Einstellung notiert wird!

Rufe nun bitte zusätzlich einmal /Q ( <http://<IP-Adresse>/Q> ) im Webinterface auf und füge die Ausgabe zu deinen Notizen hinzu, ebenso wie den Hersteller, die genaue Modellbezeichnung des Heizungssystems sowie den von dir verwendeten Bustyp (BSB/LPB/PPS).

*Ein Beispiel für eine verwertbare Auflistung neuer Parameter und den entsprechenden Zusatzangaben findest du weiter unten.*

Wenn du alles abgeschlossen hast, speichere deine Auflistung bitte als einfache .txt-Datei. Schicke die Datei nun entweder an Frederik (bsb @ code-it.de) oder melde dich [hier](#) im FHEM-Forum.

Vielen Dank für deine Unterstützung!

### Auslesen

Um die entsprechenden Telegramme neuer Parameter auszulesen, muss das Hardware-Setup (Mikrocontroller, Adapter und die Verbindung mit dem Heizungsregler) sowie die Installation der BSB-LAN-Software abgeschlossen und verwendungsfähig sein. Der Zugriff auf das Webinterface von BSB-LAN muss ebenfalls gegeben sein.

Zusätzlich zur bestehenden Verkabelung muss nun der Mikrocontroller mit deinem Laptop/PC via USB verbunden werden.

Starte dann die Arduino IDE und wähle unter „Menü Werkzeuge → Port“ den korrekten USB-Anschluss aus.

Falls du einen Arduino Due nutzt, überprüfe außerdem, ob unter „Menü Werkzeuge → Board → Due (Programming Port)“ ausgewählt ist.

Starte nun den seriellen Monitor unter „Menü Werkzeuge → Serieller Monitor“ und überprüfe, ob unten rechts in der Fußzeile „115200 Baud“ eingestellt ist.

Falls du den Verboseitätsmodus deaktiviert hast, aktiviere ihn bitte mit dem URL-Befehl <http://<IP-Adresse>/V1> im Webinterface von BSB-LAN (falls gewünscht, kann nach Beenden des Auslesen der Verboseitätsmodus mit dem Befehl <http://<IP-Adresse>/V0> wieder deaktiviert werden). Bereits jetzt werden schon einige Informationen und Telegramme im seriellen Monitor dargestellt.

Schalte als nächstes beim Heizungssystem über die integrierte Bedieneinheit zu dem Parameter, den du analysieren möchtest (mittels des Drehrades, der Pfeiltasten oder der spezifischen Eingabemöglichkeiten deiner Heizungssteuerung).

Warte auf 'Ruhe' auf dem Bus, dann schalte einen Parameter weiter vor und gleich wieder zurück zu dem Parameter, den du analysieren möchtest. Das Hin- und Herschalten soll nur sicherstellen, dass die letzte Nachricht auf dem Bus wirklich der Parameter ist, den du suchst. Nun solltest du das spezifische Parameter-Telegramm in der Ausgabe des seriellen Monitors sehen.

Solange in der Bedieneinheit der aufgerufene Parameter angezeigt wird, kommt in etwa zehntsekündigem Abstand regelmäßig das entsprechende Telegramm über den Bus.

Aber Achtung: Der Heizungsregler sendet zwischendurch automatisch auch andere Status-Telegramme (die sogenannten ‚broadcasts‘) - sei daher bitte aufmerksam beim Auslesen des gewünschten neuen Parameter-Telegramms!

### Beispiel:

```
DSP1->HEIZ QUR 113D305F
DC 8A 00 0B 06 3D 11 30 5F AB EC

HEIZ->DSP1 ANS 113D305F 00 00
DC 80 0A 0D 07 11 3D 30 5F 00 00 03 A1

**DSP1->HEIZ QUR 113D3063**
**DC 8A 00 0B 06 3D 11 30 63 5C 33**

**HEIZ->DSP1 ANS 113D3063 00 00 16**
**DC 80 0A 0E 07 11 3D 30 63 00 00 16 AD 0B**
```

Die ersten vier Zeilen in obigem Beispiel sind von dem Parameter, zu dem hingeschaltet wurde. Die letzten vier Zeilen (hier im Beispiel mit zusätzlichen **\*\*** Sternchen **\*\*** gekennzeichnet) stammen von dem Parameter, den du nun analysieren möchtest.

QUR bedeutet Anfrage, ANS die zugehörige Antwort. Anstelle von DSP1 wird eventuell RGT1 oder RGT2 angezeigt, dies ist abhängig vom jeweiligen Gerät, mit dem du die Eingaben am Heizungssystem tätigst (DSP1 = Bedieneinheit, RGTx = Raumgerät) und nicht weiter von Interesse.

Hast du nun also das betreffende Telegramm entdeckt, bietet es sich an, in der Fußzeile des seriellen Monitors unten links den Haken bei „Autoscroll“ durch einen Klick auf denselben zu entfernen. Somit bleibt die Auflistung des seriellen Monitors an dieser Stelle stehen und wird nicht durch weitere eintreffende Telegramme immer wieder verschoben. Nun ist es auch möglich, mit copy&paste zu arbeiten.

Markiere und kopiere nun also das entsprechende Frage- und Antworttelegramm (also die genannten letzten vier Zeilen in obigem Beispiel) und füge sie in eine Textdatei ein.

#### Hinweis:

*Copy&paste funktioniert im seriellen Monitor erfahrungsgemäß nicht immer zuverlässig, also überprüfe bitte jedes Mal beim Einfügen in die Textdatei, ob es sich auch wirklich um das gewünschte Telegramm handelt!*

#### WICHTIG:

**Schreibe nun zusätzlich zum Telegramm die entsprechende Parameternummer, die genaue Parameterbezeichnung sowie ggf. die jeweilige Werte-Einheit hinzu!**

**Notiere bitte außerdem die jeweils angezeigte aktive Einstellung bzw. den im Moment der Abfrage angezeigten Wert - dies ist zwingend notwendig, da ohne die aufgezählten zusätzlichen Informationen das reine Telegramm nutzlos ist!**

Wenn du einen weiteren Parameter auslesen möchtest, wiederhole das beschriebene Procedere entsprechend. Es bietet sich an, vorher wieder den Haken bei „Autoscroll“ zu setzen, bis du den nächsten zu notierenden Parameter aufgerufen hast.

Wenn du sämtliche neuen Parameter (und/oder Einstellungsoptionen neuer Parameter) samt Telegrammen und Beschreibungen etc. notiert und das Auslesen somit beendet hast, kannst du den seriellen Monitor schließen.

#### **Beispiel für eine ‚Meldedatei‘**

Hier ein Beispiel für eine erstellte ‚Meldedatei‘, die alle notwendigen Informationen für eine weitere Verarbeitung und Implementierung der neuen Parameter enthält (*Achtung: Dies ist noch ein altes Beispiel, aktuell rufe bitte /Q auf! Ein aktuelles Beispiel folgt!*):

```
Brötje NovoCondens SOB 26 C (Öl)
Anschluss: BSB

6220 Konfiguration - Software- Version: 1.3
6221 Konfiguration - Entwicklungs-Index: error 7 (parameter not supported)
6222 Konfiguration - Gerätebetriebsstunden: 12345 h
6223 Konfiguration - Bisher unbekannte Geräteabfrage: unknown type 000014
6224 Konfiguration - Geräte-Identifikation: RVS43.222/100
6225 Konfiguration - Gerätefamilie: 96
6226 Konfiguration - Gerätevariante: 100
6227 Konfiguration - Objektverzeichnis-Version: 1.0
6228 Konfiguration - Bisher unbekannte Geräteabfrage: unknown type 000014
```

```
Parameter 2270 Kessel -- Rücklaufsollwert Minimum °C
→ wird vom Mikrocontroller/BSB bei Abfrage mit 60°C angezeigt,
angezeigter Ist-Wert laut RGT-Bedieneinheit: 8°C
RGT1->HEIZ QUR 053D0908
DC 86 00 0B 06 3D 05 09 08 B0 E7
HEIZ->RGT1 ANS 053D0908 00 02 00
DC 80 06 0E 07 05 3D 09 08 00 02 00 4B 02
```

```
Parameter 5010 Trinkwasserspeicher -- Ladung
Mögliche Parameteroptionen: [Einmal/Tag | Mehrmals/Tag]
Ist: Mehrmals/Tag
RGT1->HEIZ QUR 253D0737
DC 86 00 0B 06 3D 25 07 37 D2 92
HEIZ->RGT1 ANS 253D0737 00 FF
DC 80 06 0D 07 25 3D 07 37 00 FF CE 62
```

```
Parameter 5050 Trinkwasserspeicher -- Ladetemperatur Maximum °C
Mögliche Einstelloptionen: [8°C - 90°C]
Ist: 60°C
RGT1->HEIZ QUR 253D08A3
DC 86 00 0B 06 3D 25 08 A3 01 91
HEIZ->RGT1 ANS 253D08A3 00 0F 00
DC 80 06 0E 07 25 3D 08 A3 00 0F 00 0D 90
```

 Support me on Ko-fi

[Weiter zu Kapitel 10](#)

[Zurück zum Inhaltsverzeichnis](#)

# 10. Exkurs: Heizungsregler und Zubehör

[Zurück zum Inhaltsverzeichnis](#)

[Zurück zu Kapitel 9](#)

## 10. Exkurs: Heizungsregler und Zubehör

In den folgenden Kapiteln werden die verschiedenen Heizungsregler sowie deren Bussysteme und Zubehör vorgestellt. Es ist empfehlenswert, zumindest die Kapitel über die Bussysteme und den spezifischen Regler des Wärmeerzeugers zu lesen, an den das BSB-LAN Setup angeschlossen werden soll.

Prinzipiell unterstützt BSB-LAN von der Firma SIEMENS hergestellte (Heizungs-)Regler, die einen BSB und/oder LPB aufweisen. Diese werden von verschiedenen Heizungsherstellern 'gebrandet' und verbaut.

*Klarstellung:*

*Wann immer ich von "Regler" spreche, dann meine ich die sog. "BMU" (boiler management unit). Das ist das Gerät im Inneren des Gehäuses des Wärmeerzeugers, das die Funktionsweise des Wärmeerzeugers steuert und an dem die Sensoren, Pumpen, die Bedieneinheit und die Raumgeräte angeschlossen sind.*

*Die 'Bedieneinheit' und die Raumgeräte hingegen sind die Komponenten, die außerhalb des Gehäuses angebracht und mit einem Display und Knöpfen zur Bedienung des Wärmeerzeugers ausgestattet sind.*

*Hinweise:*

Die folgende Aufzählung der verschiedenen Reglertypen mag im ersten Moment ein wenig verwirrend erscheinen, doch im Grunde kann man sich eine vereinfachte Regel merken:

Wenn die Reglerbezeichnung mit einem "S" endet (RVS und LMS), dann gehört der Regler zur 'aktuellen' Generation.

Die Gerätefamilie LMS ist dabei die Modellreihe für Gasgeräte, alle anderen Systeme nutzen RVS-Regler.

Je größer die darauffolgende Nummer in der Reglerbezeichnung ist, desto 'größer' vom (internen) Funktionsumfang her und meist auch hinsichtlich der Anzahl der Anschlüsse ist das jeweilige Modell.

Je nach 'Größe', Typ und vorgesehenem Verwendungsumfang sind somit auch unterschiedliche Parameter verfügbar.

**ACHTUNG:**

**Aus aktuellem Anlass sei bereits hier darauf hingewiesen, dass die Heizungshersteller offensichtlich eine neue Heizungs- und Reglergeneration auf den Markt gebracht haben, die nach bisherigem Wissensstand NICHT kompatibel mit BSB-LAN ist (s. Kap. 10.2.3)!**

### 10.1 Bussysteme der Heizungsregler: BSB, LPB, PPS

Bei BSB (Boiler System Bus), LPB (Local Process Bus) und PPS (Point-to-Point Schnittstelle) handelt es sich um jeweils verschiedene und untereinander nicht kompatible Bussysteme bzw. Schnittstellen. Es können also jeweils nur Geräte angeschlossen werden, die den gleichen Bus- bzw. Schnittstellen-Typ aufweisen.

**BSB (Boiler System Bus)** und **LPB (Local Process Bus)** sind zwei verschiedene "echte" Bussysteme, die sich vereinfacht in zwei Nutzungsklassen unterscheiden lassen:

- Der BSB ist im Grunde ein 'Regler-lokaler' Bus, er kommt bei einem Regler und 'direktem' Zubehör wie bspw. Bedieneinheit, Raumgerät, Erweiterungsmodul zum Einsatz.
- Der LPB ist im Grunde ein 'Regler-übergreifender' Bus, er kommt zum Einsatz, wenn mehrere Regler miteinander verbunden werden sollen (bspw. bei einer Koppelung von mehreren Wärmeerzeugern, um eine Kaskadenschaltung zu realisieren).

*Beispiel:*

Vorhanden sind eine Öl- oder Gasheizung, ein nachgerüsteter wasserführender Kamin und eine thermische Solaranlage zur Unterstützung des Heizkreises oder der Warmwasserbereitung. Alle drei Wärmeerzeuger sind hydraulisch an einem Pufferspeicher angeschlossen. Die Wärme für den Heizkreis soll vom Pufferspeicher bezogen werden. Die Regelung der Solaranlage und des Feststoffkessels übernimmt ein Solarsystemregler (SSR), die Kesselsteuerung der Heizung übernimmt in diesem Beispiel der interne Heizungsregler. Alle Sensoren, Pumpen, Mischer etc. sind am SSR angeschlossen, welcher jedoch via LPB mit dem Heizungsregler verbunden ist. Durch diese Verbindung der beiden Regler kann somit bspw. eine Pufferspeicherladung geregelt werden, bei der die Heizung nur aktiv wird, wenn weder Solar noch Feststoffkessel den Puffer laden / geladen haben.

\*Wenn das BSB-LAN Setup via BSB an einem der beiden Regler aus oben genanntem Beispiel angeschlossen ist, kann er nur auf den jeweiligen



Regler 'lokal' zugreifen, an dem er angeschlossen ist (also bspw. Heizungsregler oder SSR). Pro Regler muss in dem Fall ein BSB-LAN Setup am jeweiligen BSB angeschlossen werden, wenn Zugriff auf beide Regler gewünscht ist.

- Wenn ein Adapter via LPB an einem der beiden Regler aus oben genanntem Beispiel angeschlossen ist, müssen
1. die Geräte- und Segmentadressen entsprechend der LPB-Konfigurationsanforderungen eingestellt werden, und
  2. beim Adapter eine Zieladresse eingestellt werden, an die die jeweiligen Anfragen des Adapters geschickt werden.

**PPS (Point-to-Point-Schnittstelle)** hingegen ist kein "echter" Bus, es können also nicht mehrere Teilnehmer mit spezifischen Adressen angeschlossen werden. PPS ist eher (wie der Name schon sagt) eine Schnittstelle, um ein weiteres Gerät (i.d.R. ein Raumgerät) anzuschließen.

Bei allen in diesem Handbuch aufgezählten (aktuellen) Reglern des Typs RVS, LMS1x und LMU7x ist ein BSB-Anschluss am Regler zu finden. Ein LPB ist standardmäßig nur bei Reglern des Typs RVS (nur Modellreihen 4x und 6x, Ölheizungen und SolarSystemRegler) vorhanden. Bei Reglern des Typs RVS21 (Wärmepumpen), LMS14/15 sowie LMU74/75 (Gasheizungen) ist ein LPB i.d.R. mittels eines ClipIn-Moduls nachrüstbar (s. [Kap. 10.2.6](#)). PPS ist nur bei alten Reglern vorzufinden und wird heutzutage nicht mehr verbaut.

### 10.1.1 BSB

Der BSB (Boiler System Bus) ist im Grunde ein 'lokaler' Bus. Mit 'lokal' meine ich in diesem Fall den spezifischen Regler, der im Wärmeerzeuger zum Einsatz kommt.

An den BSB werden bspw. die Bedieneinheit, Raumgeräte und Erweiterungsmodule angeschlossen. Diese Geräte haben dann 'lokal' Zugriff auf den Regler. Schließt man das BSB-LAN Setup an den BSB an, so hat man aufgrund der eindeutigen Adressierung Zugriff auf alle Busteilnehmer dieses Reglers.

Der BSB ist bei den im Folgenden vorgestellten Reglern des Typs RVS, LMS1x sowie LMU7x vorhanden. Sollte dein Regler einen BSB- und einen LPB-Anschluss aufweisen und kein weiterer Regler via LPB angeschlossen sein (bspw. eines weiteren Wärmeerzeugers bei einer Kaskadenschaltung, ein Solaranlagenregler o.ä.) und möchtest du nur Zugriff auf diesen einen Regler, dann ist der Anschluss des BSB-LAN Setups an den BSB-Anschluss empfehlenswert.

#### Adressierung beim BSB

Beim BSB wird aufgrund des Bussystems jedem Teilnehmer eine spezifische Adresse zugeteilt. Folgende Adressen sind bereits festgelegt:

Bus-Adresse	Geräteadresse	Gerät (Bezeichnung im Seriellen Monitor)
0x00	0	Heizungsregler („HEIZ“)
0x03	3	Erweiterungsmodul 1 („EM1“) / Mischer-ClipIn AGU
0x04	4	Erweiterungsmodul 2 („EM2“) / Mischer-ClipIn AGU
0x06	6	Raumgerät 1 („RGT1“: QAA55, QAA75, IDA)
0x07	7	Raumgerät 2 („RGT2“: QAA55, QAA75)
0x08	8	Raumgerät 3/P und/oder OCI700 Servicetool („RGT3“)
0x0A	10	reglerseitige Bedieneinheit / Display 1 („DSP1“)
0x0B	11	Servicegerät (QAA75 als Servicegerät parametrier) („SRVC“)
0x0C	12	reglerseitige Bedieneinheit / Display 2 („DSP2“)
0x0D	13	reglerseitige Bedieneinheit / Display 3 („DSP3“)
0x31	49	OZW672 Webserver
0x32	50	(vermutlich) Funkempfänger („FUNK“)
0x36	54	Remocon Net B („REMO“)
<b>0x42</b>	<b>66</b>	<b>BSB-LPB-LAN-Adapter („LAN“)</b>
0x7F	127	Broadcast („INF“-Meldungen)

Dem BSB-LAN Adapter wird in der Voreinstellung die Busadresse **0x42** zugeteilt, was der BSB-Adresse 66 entspricht. Die Adresse wird in der

Datei `BSB_LAN_config.h` festgelegt.

### 10.1.2 LPB

Der LPB (Local Process Bus) ist ein ‚übergreifender‘ Bus zur Nutzung mehrerer angeschlossener Regler in einem kommunikationsfähigen Verbund.

Über den LPB können mehrere Regler miteinander verbunden werden und bei korrekter Parametrierung gewisse Werte miteinander teilen bzw. sich gegenseitig beeinflussen.

Auf diese Weise kann bspw. eine Kaskadenschaltung von mehreren Brennern realisiert werden oder eine Gas- oder Öl-Heizung mit einer Solaranlage und einem Feststoffkessel regelungstechnisch 'verbunden' werden.

Der korrekte Anschluss der einzelnen Komponenten sowie die korrekte Parametrierung der jeweiligen Regler sollte im Normalfall bereits bei der Installation der Anlage durch den Heizungsinstallateur erfolgt sein.

Eine übergreifende Abfrage von Werten oder Parametern zweier oder mehrerer Regler im LPB-Verbund via BSB-LAN Setup kann durch Hinzufügen der spezifischen Adresse des Busteilnehmers erfolgen.

Die spezifischen technischen Daten, Leistungsmerkmale und Anforderungen an entsprechende Installationen und Parametrierungen hinsichtlich der Geräte- und Segmentadressen sind den jeweiligen technischen Dokumentationen der Hersteller zu entnehmen. Da die teilweise doch recht komplexe Installation i.d.R. bereits bei der Installation vom jeweiligen Monteur vorgenommen wird, wird an dieser Stelle nicht auf weitere Besonderheiten eingegangen.

Dem interessierten Anwender seien an dieser Stelle insbesondere zwei Dokumente von „Siemens Building Technologies - Landis & Staefa Division“ empfohlen:

- CE1N2030D Local Process Bus LPB Systemgrundlagen
- CE1N2032D Local Process Bus LPB Projektierungsgrundlagen

#### Adressierung beim LPB

Beim LPB ist die Adressierung anders als beim BSB. Prinzipiell gibt es verschiedene Segmente (bzw. Segmentadressen) und Geräteadressen. Den Segmentadressen kommt eine andere Bedeutung zu, als den Geräteadressen.

In diesem Zusammenhang sei lediglich darauf hingewiesen, dass zusätzlich zu diesem Unterschied auch die jeweiligen Adressvergaben selbst beim LPB anders gestaltet sind. Bei der Busadresse `0x00` beispielsweise ist die erste Ziffer hinter dem x die Segmentadresse 0 (also 0=0, 1=1 etc.), die zweite 0 hingegen ist Busadresse des Gerätes plus eins (also 0=1, 1=2 etc.).

Beispiel:

Das Gerät im obigen Beispiel `0x00` befindet sich im Segment 0 mit der Adresse 1.

Die bei BSB-LAN in der Datei `BSB_LAN_config.h` voreingestellte Adresse `0x42` bedeutet somit, dass der Adapter im Segment 4 mit der Adresse 3 angemeldet wird.

### 10.1.3 PPS-Schnittstelle

Die PPS-Schnittstelle findet sich bei *älteren* Reglern und stellt eine Punkt-zu-Punkt-Schnittstelle dar, mittels derer digitale Bedieneinheiten/Raumgeräte wie das [QAA70](#) angeschlossen werden können. An demjenigen Anschluss wird analog zum QAA auch der Adapter angeschlossen. Die Anschlüsse sind dem jeweiligen Handbuch zu entnehmen, häufig sind dies jedoch die Pins "A6" und "MD" (oder auch "M") (in dem Fall dann "A6" → CL+ und "M"/"MD" → CL-).



Die Anschlüsse "A6" und "MD" bei einem Siemens RVA53 Regler.

PPS scheint bei folgenden Reglern zum Einsatz gekommen zu sein (siehe „Siemens Raumgerät QAA70 Basisdokumentation“, CE1P1638D): RVP digital Serie D, RVP54..., ALBATROS RVA..., LGM11...; bzw. u.a. bei folgenden Heizungen: Brötje BBS/WGB 2N, Weishaupt WRD 0.2 / 1.1, Sieger TG11 (mit Siegermatic S42DB), Olymp THR 5-25C, Schäfer Interdomo (mit DomoCommand DC 225).

Bei den „Bedieneinheiten“/Reglern handelt es sich (bei Brötje) vermutlich meist um Eurocontrol-Varianten, manchmal auch um Eurotronic-Varianten (anscheinend NICHT Eurotronic A, nur Eurotronic D aufwärts). *Als Hinweis kann die Anschlussmöglichkeit einer [QAA70-Raumeinheit](#) überprüft werden - ist diese anschließbar, so sollte auch der Anschluss des Adapters möglich sein.*

Die beiden Geräte (Raumgerät und Regler) kommunizieren nur bedingt miteinander. Der Regler sendet Infos, schickt dann später mit einem einzigen Byte (0x17) eine Anforderung an das Raumgerät, das dann teilweise auf vorhergehende Regler-Infos reagiert, andererseits aber auch nach eigenem Rhythmus seine Infos sendet. Und das teilweise in unterschiedlicher Häufigkeit. Der Bus kommt so kaum zur Ruhe, i.d.R. werden bis zu zwei Telegramme pro Sekunde ausgetauscht, entsprechend schnell muss die Software dann auch antworten. Kommt auf bestimmte Anfragen des Reglers keine oder nicht die richtige Antwort, wird angenommen, dass es kein Raumgerät mehr gibt und der Regler verfällt wieder in einen Suchmodus.

Der Funktionsumfang ist hierbei nur rudimentär und beschränkt sich derzeit mittels BSB-LAN derzeit auf etwa ein Dutzend Parameter, die man lesen/schreiben kann (*Anm.: Die folgende Auflistung ist u.U. nicht komplett - ausschlaggebend ist Kategorie "PPS-Bus" im Webinterface und die von Regler unterstützten Parameter!*):

- Raumtemperatur Ist
- Raumtemperatur Soll
- Außentemperatur (read-only)
- Außentemperatur gemischt (read-only)
- Position Drehknopf
- Kesselvorlauftemperatur (read-only)
- Mischervorlauftemperatur (read-only)
- Status Trinkwasserbetrieb (read-only)
- Trinkwassertemperatur Ist (read-only)
- Trinkwassertemperatur Soll
- Betriebsart
- Anwesenheit

**Im Webinterface von BSB-LAN ist die einzig verfügbare Kategorie bei der Verwendung von PPS die Kategorie "PPS-Bus"! Aus den anderen Kategorien sind keinerlei Parameter abrufbar!**

Somit entfällt auch die Abfrage von `URL/Q` zur Kontrolle auf nicht-freigegebene Parameter!

Immerhin lassen sich damit aber die wichtigsten Funktionen einer intelligenten Heizungssteuerung umsetzen, indem man z.B. gewichtete Raumtemperaturen sendet und die Solltemperaturen nach vielfältigeren Kriterien steuern kann.

#### **Hinweise:**

Sollte bereits ein QAA70 angeschlossen sein, so ist der Zugriff mittels BSB-LAN nur lesend möglich! Soll BSB-LAN schreibend einwirken, also aktiv Werte und Einstellungen verändern, so muss ein vorhandenes QAA70 dauerhaft deinstalliert werden, da es ansonsten mit den eigenen Werten alles wieder überschreibt!

Bzgl. der spezifischen Bus-Einstellungen in der Datei `BSB_LAN_config.h` beachte die dortigen Hinweise in [Kap. 2.2](#).

Über PPS tauschen Heizung und Raumgerät bzw. BSB-LAN permanent Daten aus. Das Protokoll ist sehr zeitkritisch. Das Aufrufen von längeren Webseiten führt dazu, dass der Mikrocontroller nicht rechtzeitig auf entsprechende Anfragen der Heizung reagieren kann, weswegen die Heizung dann denkt, dass die Gegenseite ausgefallen ist. Das ist an sich kein Problem, nach ca. 10-20 Sekunden, nachdem der Mikrocontroller wieder „ansprechbar“ ist, haben sich beide wieder verständigt. Bis dann aber wieder alle Werte ausgetauscht bzw. aktualisiert sind, kann es noch mal 1-2 Minuten dauern, so dass sich Änderungen dann erst entsprechend verzögert zeigen. Von zu vielen Anfragen auf den Mikrocontroller sollte daher bei PPS abgesehen werden und etwaige Sensoren etc. dann ggf. auf einen zweiten Mikrocontroller ausgelagert werden.

Bei der ersten Verwendung bzw. nach einem Reboot des Mikrocontroller muss man (anders als bspw. beim BSB) einige Zeit abwarten, bis die Parameter abrufbar/verfügbar sind.

### Wichtiger Hinweis für Nutzer des (veralteten) Setups Adapter v2 + Arduino Mega 2560:

Aufgrund der zeitkritischen Kommunikation bei PPS ist es sinnvoll, das Setup auf die Nutzung der Hardware-Serial umzustellen. Dazu sind folgende Änderungen vorzunehmen:

- Die Adapterplatine v2 muss *komplett* bestückt sein.
- Es darf nur die Lötbrücke SJ1 gesetzt sein.
- Die Platine muss um eine Pin-Reihe versetzt in Richtung Mitte des Arduino eingesetzt werden.
- Die Konfiguration muss entsprechend geändert werden, indem die BSB-bus-Variable auf die Pins 19 (RX) und 18 (TX) gesetzt wird.

## 10.2 Detaillierte Beschreibung der kompatiblen Regler

Die folgende Reglerauflistung und -beschreibung soll einen kurzen Überblick über eine Auswahl der bereits von BSB-LAN unterstützten Geräte und deren rudimentären Unterschiede geben. Auf die unterschiedliche reglerspezifische Verfügbarkeit von speziellen Parametern wird nicht weiter eingegangen.

Mittels BSB-LAN steht i.d.R. der gesamte Funktionsumfang der jeweiligen Reglertypen zur Verfügung. Dieser ist jedoch hinsichtlich der verfügbaren Parameter naturgemäß unterschiedlich: Ein Regler der neusten Generation weist mehr Parameter und Einstelloptionen als ein Regler der ältesten Generation auf. Die Heizungsanlage wird dadurch jedoch nicht zwingend ineffizienter oder ist per se 'veraltet' und unbrauchbar! Dank BSB-LAN können jedoch i.d.R. auch die ältesten unterstützten Regler noch etwas 'smarter' gemacht und in die Hausautomatisierung mit eingebunden werden.

### Hinweis

Im Folgenden werden die grundsätzlichen Reglerserien wie bspw. LMU, LMS, RVS grob vorgestellt. Dies soll lediglich dem grundsätzlichen Verständnis und Überblick über die unterschiedlichen Reglerserien dienen.

Als weitere Unterteilung erfolgt dazu i.d.R. ebenfalls die Nennung der beiden Ziffern nach der Buchstabenkombination, also bspw.

"RVS43.xxx". Auf eine weitere, genauere Unterteilung der jeweiligen Reglerserien mit Nennung der *kompletten* Reglerbezeichnung wie bspw. "RVS43.222/xyz" oder "RVS43.325/xyz" und deren spezifischer Unterschiede zu einander wird jedoch weitestgehend verzichtet, um den Umfang dieses Kapitels in einem überschaubaren Ausmaß zu halten.

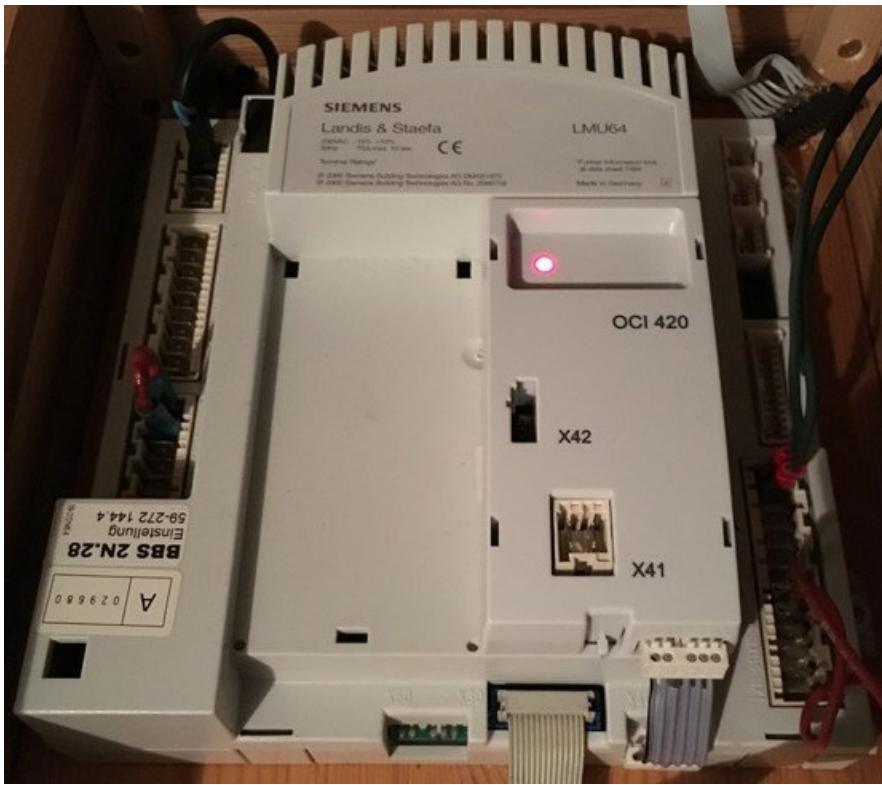
Sollten tiefergehende, reglerspezifische Informationen zum jeweiligen Reglertyp benötigt werden, so sind die entspr. Anleitungen des Heizungsherstellers heranzuziehen. Sollten diese nicht verfügbar oder inhaltlich nicht umfassend genug sein, so empfiehlt es sich, unter Angabe der vollständigen Reglerkennung nach Anleitungen des Reglerherstellers "Siemens Albatros" zu suchen.

### 10.2.1 LMx-Regler

Im Folgenden werden die Regler des Typs LMU und LMS aufgeführt. Diese sind erfahrungsgemäß bei Gasheizungen/-thermen verbaut.

#### 10.2.1.1 LMU-Regler

Regler der Serie **LMU54/LMU64** sind in älteren Heizungssystemen vorzufinden, in aktuellen Modellen werden sie nicht mehr verbaut. Diese Regler basieren auf dem OpenTherm Protokoll, welches inkompatibel mit dem BSB-LAN Projekt ist. BSB/LPB/PPS ist bei diesem Reglertyp nicht vorhanden, LPB kann jedoch mittels eines ClipIn-Moduls (OCI420) nachgerüstet werden.



Ein Regler des Typs LMU64 samt installiertem OCI420 ClipIn-Modul.

Genauere Hinweise bezüglich dieses Reglertyps sind in [Kap. 10.2.4](#) zu finden.

Weitere Hinweise zum Nachrüsten eines LPB mittels ClipIn (OCI420) sind in [Kap. 10.2.6](#) zu finden.

Regler der Serie **LMU74/LMU75** scheinen die Nachfolger der LMU54/LMU64-Reglerreihe zu sein und werden ebenfalls nicht mehr verbaut.



Ein Regler des Typs LMU7x.

Der LMU7x-Reglertyp weist i.d.R. nur einen BSB-Anschluss auf, an dem der Adapter angeschlossen wird. LPB muss bei Bedarf mittels eines ClipIn-Moduls (OCI420) nachgerüstet werden (für die Nutzung von BSB-LAN ist dies jedoch nicht notwendig).

Als Bedieneinheit kommt i.d.R. eine Variante des Siemens AVS37.294 zum Einsatz (Bezeichnung bspw. „ISR Plus“ bei Brötje).

Als Fühler kommen i.d.R. NTC10k (QAD36, QAZ36) und NTC1k (QAC34 = Außentemperaturfühler) zum Einsatz.

Der Feuerungsautomat befindet sich bei diesen Reglern auf der Platine selbst.

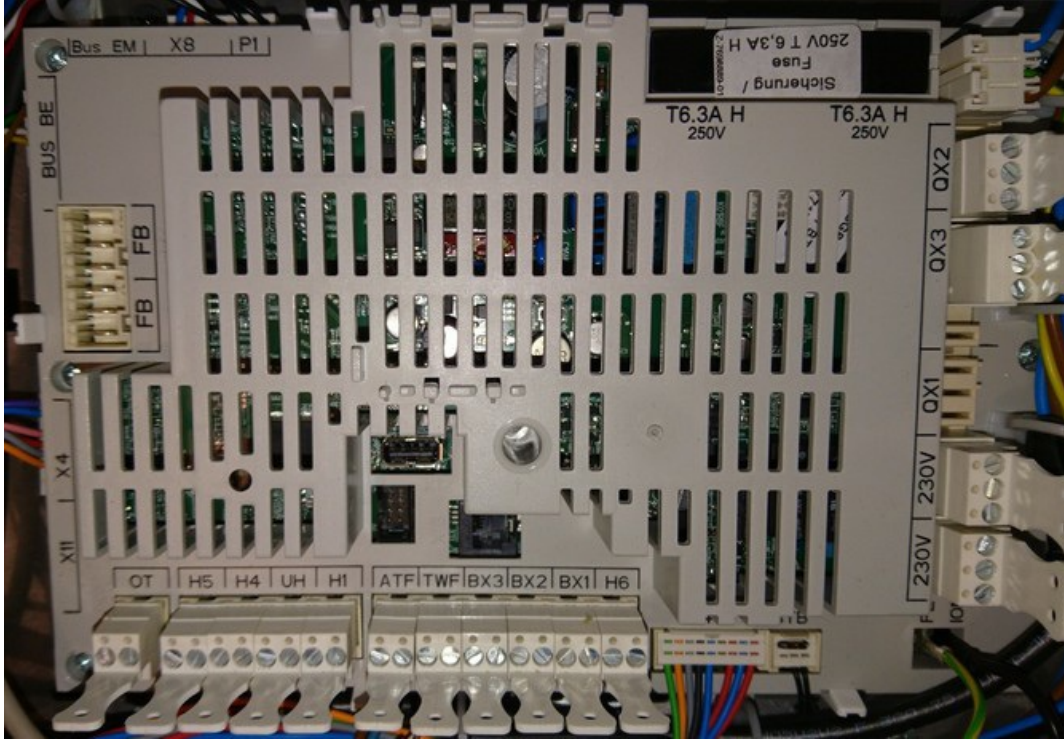


### 10.2.1.2 LMS-Regler

Regler der Serie **LMS** scheinen die Nachfolger der LMU-Serie und somit die aktuelle Reglergeneration zu sein.

Der (Funktions-)Unterschied zwischen dem LMS14 und dem LMS15 scheint in der „Sitherm Pro“-Anwendung zur Optimierung des gesamten Verbrennungsprozesses zu liegen, die anscheinend nur die LMS15-Regler aufweisen.

Der LMS-Reglertyp weist i.d.R. nur einen BSB-Anschluss auf, an dem der Adapter angeschlossen wird. LPB muss bei Bedarf mittels eines ClipIn-Moduls (OCI345) nachgerüstet werden (für die Nutzung von BSB-LAN ist dies jedoch nicht notwendig).



Als Bedieneinheit kommt i.d.R. eine Variante des Siemens AVS37.294 zum Einsatz (Bezeichnung bspw. „ISR Plus“ bei Brötje).

Als Fühler kommen i.d.R. NTC10k (QAD36, QAZ36) und NTC1k (QAC34 = Außentemperaturfühler) zum Einsatz.

Der Feuerungsautomat befindet sich bei diesen Reglern auf der Platine selbst.

Service-/Ersatzplatinen des LMS1x-Reglers sind im Falle eines Defekts erhältlich, diese müssen jedoch mittels eines speziellen USB-Sticks gerätespezifisch geflasht werden.

### 10.2.2 RVx-Regler

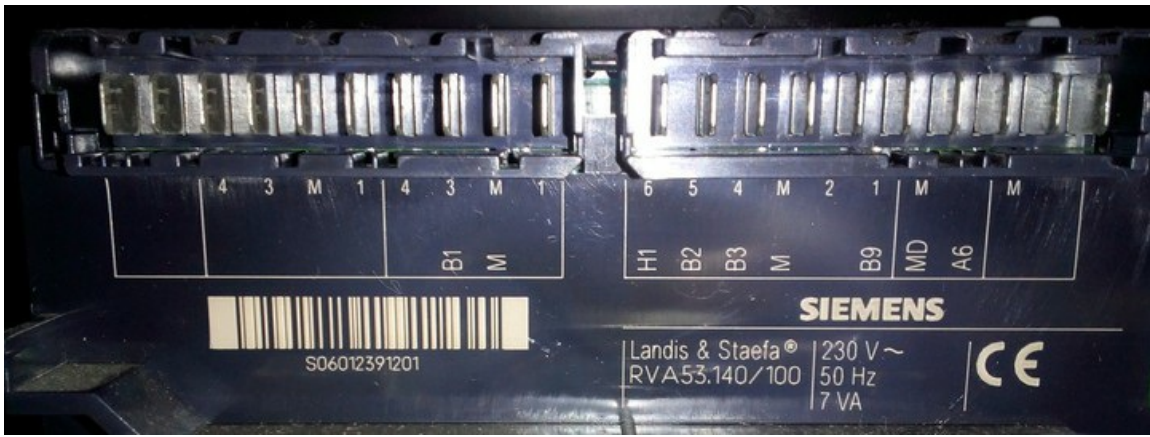
Im Folgenden werden die Regler des Typs RVA, RVP und *RVS (aktueller Reglertyp)* aufgeführt. Diese scheinen i.d.R. bei Ölheizungen, Wärmepumpen und verschiedenen ‚alleinstehenden‘ Reglern (Zonenregler, Solarsystemregler) zum Einsatz zu kommen.

#### 10.2.2.1 RVA- und RVP-Regler

Regler des Typs **RVA** sind in älteren Heizungssystemen vorzufinden, in aktuellen Modellen werden sie nicht mehr verbaut. Je nach Modell weisen sie nur einen PPS oder einen PPS- und LPB-Anschluss auf, jedoch keinen BSB.

Als (integrierte) Bedieneinheit ist meist eine Variante der "Eurocontrol" (Brötje) verbaut.





Ein Regler des Typs RVA53.



Vorderansicht: Bedieneinheit eines Reglers des Typs RVA53.

Regler des Typs **RVP** scheinen noch älter als RVA-Regler zu sein und weisen lediglich eine PPS-Schnittstelle auf.

### 10.2.2.2 RVS-Regler

Regler des Typs **RVS** scheinen die 'aktuelle' Reglergeneration darzustellen. Sie weisen i.d.R. sowohl einen LPB-, als auch mehrere BSB-Anschlüsse auf.

Ausnahmen scheinen die Regler der Reihen RVS21, RVS41, RVS51, RVS61 und RVS23 zu sein:

- RVSx1-Regler kommen bei Wärmepumpen zum Einsatz, der RVS21 scheint nur einen BSB aufzuweisen.
- RVS23-Regler kommen bei einer bestimmten Weishaupt-Modellreihe (WTU) zum Einsatz und scheinen nur einen LPB aufzuweisen. Bei Weishaupt scheinen diese Regler als "WRS-CPU-Bx" bezeichnet zu werden. Weitere Hinweise zu diesem Reglermodell finden sich in [Kap. 10.2.5](#).

Als Bedieneinheit kommt hier i.d.R. eine Variante des Siemens AVS37.294 zum Einsatz (Bezeichnung bspw. „ISR Plus" bei Brötje).

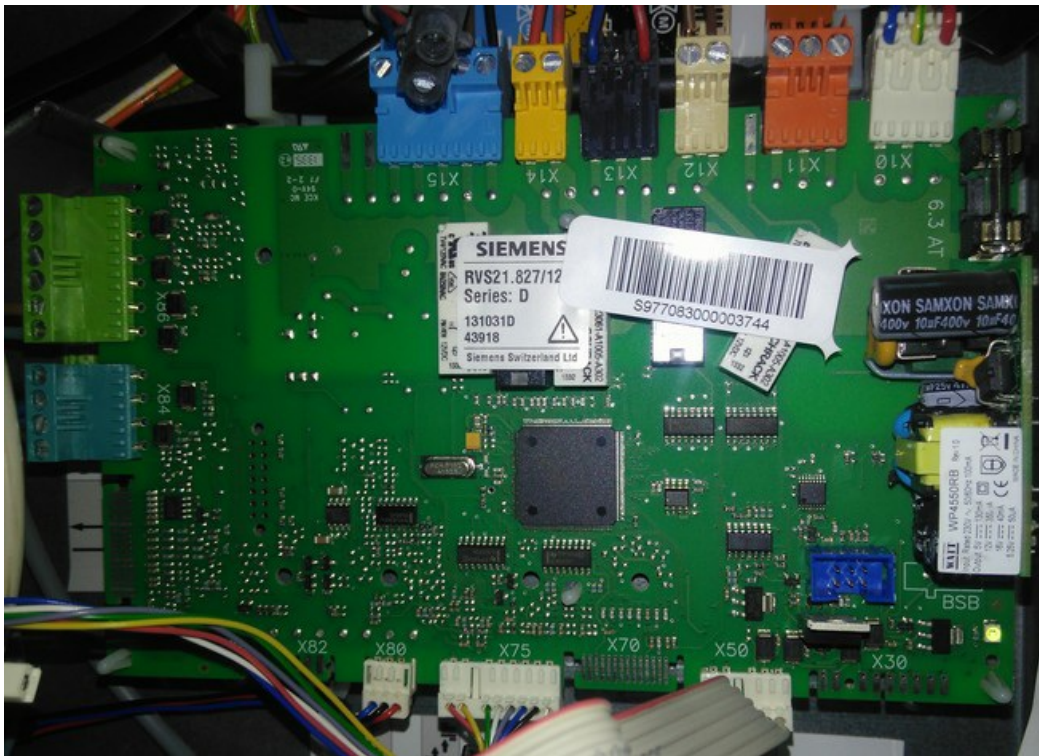
Als Fühler kommen i.d.R. NTC10k (QAD36, QAZ36) und NTC1k (QAC34 = Außentemperaturfühler) zum Einsatz.

Der Feuerungsautomat befindet sich bei diesen Reglern nicht auf der Platine, sondern ist als zusätzliches Bauteil im System verbaut.

Die folgende grobe Darstellung der Gerätefamilie zeigt wesentliche Unterschiede auf.

#### RVS21.xxx

Der RVS21 ist der Reglertyp, der in Wärmepumpen Verwendung findet. Er bietet einen BSB und Anschlüsse für ein optionales Raumgerät.



Ein RVS21 Regler.

LPB ist bei einem RVS21 im Bedarfsfall via OCI345 nachzurüsten (für die Nutzung von BSB-LAN ist dies jedoch nicht notwendig, da ein BSB-Anschluss vorhanden ist).

#### **RVS41.xxx**

Der RVS41 ist ebenfalls ein Reglertyp, der in Wärmepumpen Verwendung findet. Er bietet BSB und LPB und scheint (zumindest äußerlich betrachtet) dem RVS43 recht ähnlich zu sein.

#### **RVS43.xxx**

Der RVS43 ist die Serie, die bspw. in Ölbrennwertanlagen, dem Brötje Pelletkessel SPK, bei dem Brötje Kaskadenregler "ISR-BCA" sowie dem Brötje Heizungssystemmanager "ISR-HSM" zum Einsatz kommt, bei vereinzelt Herstellern (bspw. Bösch) kommt dieser Reglertyp auch in Wärmepumpen vor. Abhängig vom jeweiligen Wärmeerzeuger bzw. Gerät (bspw. BCA, HSM) gibt es jedoch modellspezifische Unterschiede hinsichtlich der Anschlüsse und des Funktionsumfangs.

Regler der Serie RVS43 sind i.d.R. mit (mindestens) einem BSB- und einem LPB-Anschluss ausgestattet.

Die Anzahl der Anschlüsse und Funktionen kann mit einem Erweiterungsmodul AVS75.xxx vergrößert werden.

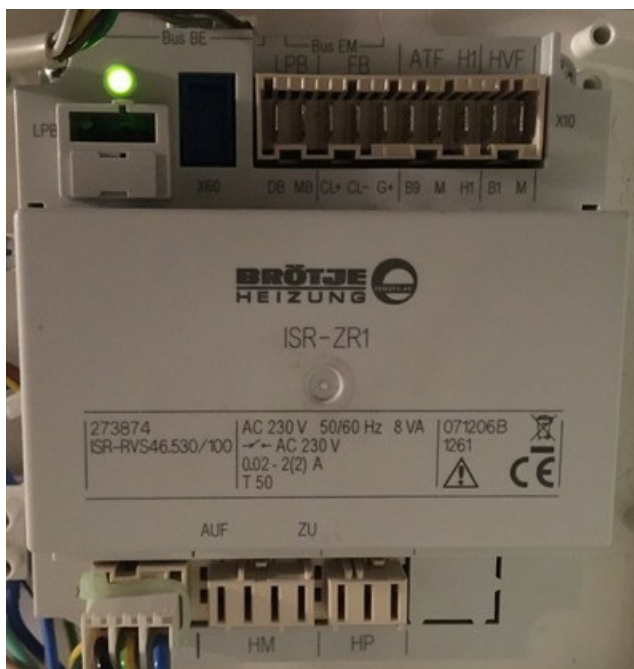


Ein Regler des Typs RVS43.222.

Das Modell RVS43.325 wird von Brötje als Ersatzregler ausgewiesen und ist u.a. für Ölbrennwertanlagen der Serien BOB, SOB, WOB und den Pelleter SPK einsetzbar. Dieser Regler verfügt bspw. im direkten Vergleich zum oben abgebildeten RVS43.222 aus einem Brötje SOB C u.a. über zusätzliche Anschlüsse. Bei einem Einsatz als Ersatzregler muss er je nach Heizungsmodell bei einem Austausch entspr. parametrieren werden. Genauere Informationen hierzu finden sich in der entspr. Montageanleitung dieses Ersatzreglers.

#### RVS46.xxx

Der RVS46 ist ein kleiner Zonenregler, der je nach Ausführung (ZR1 / ZR2) Anschlüsse für ein oder zwei Pumpen- / Mischerkreise hat. Er kann einzelne Zonen eigenständig oder auch als Erweiterung eines vorhandenen Reglers im LPB-Verbund eingesetzt zu werden. Er bietet sowohl eine BSB- als auch einen LPB-Anschluss.



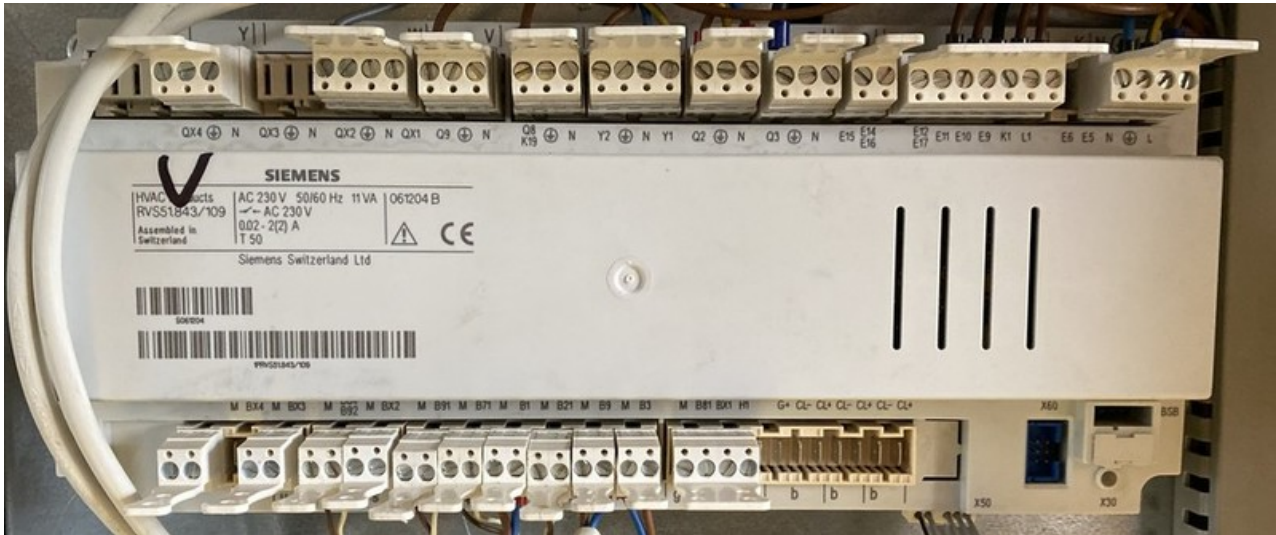
Der kleine Zonenregler ZR1.

Der ZR1 ist nicht dafür gedacht oder geeignet, bspw. den Verbrennungsprozess eines Ölbrenners zu steuern.

#### RVS51.xxx

Der RVS51 ist der 'große' Reglertyp, der in Wärmepumpen Verwendung findet. Er bietet BSB und LPB und scheint (zumindest äußerlich betrachtet) dem RVS63 recht ähnlich zu sein.





Ein Regler des Typs RVS51.843.

### RVS61.xxx

Der RVS61 ist der 'große' Reglertyp, der in Wärmepumpen Verwendung findet. Er bietet BSB und LPB und scheint (zumindest äußerlich betrachtet) dem RVS63 recht ähnlich zu sein.

### RVS63.xxx

Der RVS63 ist der größte Regler mit den meisten Anschlüssen und kann aufgrund seines Funktionsumfanges vielfältig eingesetzt werden. Er ist in erster Linie dafür vorgesehen, komplexere Anlagen mit einer zusätzlichen Solarthermieanlage zu steuern. Bei Brötje wird er daher auch als "Solar System Regler" bezeichnet. Er ist sowohl als optionaler Nachrüstregler in einem Wandgehäuse erhältlich, wird aber auch als bereits in den Wärmerzeuger eingebauter interner Regler verwendet.



Ein Regler des Typs RVS63.

Der **RVS65.xxx** scheint relativ identisch zum RVS63 zu sein und wurde bisher lediglich einmal von einem User gemeldet, bei dem der Regler als SSR von Brötje in einem Wandgehäuse zum Einsatz kam.

## 10.2.3 Hinweis: Inkompatible Systeme von Brötje und Elco

Aus aktuellem Anlass sei an dieser Stelle darauf hingewiesen, dass die genannten Heizungshersteller neue Gerätemodelle auf den Markt gebracht haben, deren Regler NICHT mit BSB-LAN kompatibel sind.

Bei Brötje scheint es sich hierbei um die Modellreihen

- WGB-K, WGB 14.1/22.1/28.1/38.1 (Gas),
- WHC, WHS, WLC, WLS (Gas),
- BOK (Öl) sowie
- BLW Split-P, BLW Split C und BLW Split-K C (Wärmepumpen) zu handeln.

Bei Elco scheint es sich um die Modellreihe "Thision Mini" zu handeln.

Bei diesen Modellen sind scheinbar ,IWR CAN'-basierte Regler verbaut (geräteseitig "IWR Alpha", kompatibles Raumgerät "IWR IDA"), die weder einen LPB noch einen BSB aufweisen.

Das folgende Bild einer WLC24-Platine zeigt die dort vorhandenen Anschlüsse.



*Anschlüsse des neuen Reglertyps einer Brötje WLC24 - dieser Regler ist inkompatibel mit BSB-LAN!*

Neben einer Servicebuchse (vermutlich IWR CAN) sind dort ein nicht weiter dokumentierter ,L-Bus' und ein ,R-Bus' zugänglich. Am ,R-Bus' (Raumgeräte-Bus) kann bei Bedarf entweder ein Raumthermostat (On/Off) oder das neue ,smarte' Raumgerät „Brötje IDA" angeschlossen werden.

**ACHTUNG: An keinem dieser Anschlüsse ist der BSB-LPB-LAN-Adapter anschließbar!**

#### 10.2.4 Hinweis: Spezialfall LMU54/LMU64-Regler

Regler des Typs LMU54/LMU64 basieren auf OpenTherm, das andere Bus-Spezifikationen und auch ein anderes Kommunikationsprotokoll aufweist. Daher ist OpenTherm nicht kompatibel mit BSB-LAN.

Es gibt jedoch eine Möglichkeit, diesen Reglertyp trotzdem anzubinden: Wie auch bei den BSB-Reglern LMU7x und LMS1x kann man mittels eines sog. ClipIn-Moduls (Typ OCI420) einen LPB nachrüsten. An diesem wiederum ist der Adapter anschließbar.



Ein Regler des Typs LMU64 samt installiertem OCI420 ClipIn-Modul.

Der Funktionsumfang ist bei diesem Regler (auch bei der Nutzung von BSB-LAN) jedoch relativ eingeschränkt und außerdem in gewissem Ausmaß von der Softwareversion des Reglers abhängig (getestet mit LMU64, SW v2.08 vs. SW v3.0 vs. SW v3.03): Regler mit SW ab v3.0 scheinen mehr (via BSB-LAN steuerbare) Funktionen aufzuweisen als Regler mit SW <v3.0. Insbesondere seien hier die beiden Sollwert-Temperaturparameter 709 und 711 genannt, anhand derer das Brennerverhalten in gewissem Umfang bestimmt werden könnte – diese können nur mit SW ab v3.0 genutzt bzw. verändert werden. (Hinweis: Derzeit läuft noch ein Versuch, ob das Brennerverhalten zufriedenstellend via Relais an einem anderen Kontakt beeinflusst werden kann.)

Auf Parameter wie Außentemperatur, Kesseltemperatur, TWW-Temperatur, Vorlauftemperatur etc. kann jedoch nach bisherigem Kenntnisstand bei allen erwähnten Softwareversionen zugegriffen werden.

Fairerweise muss man an dieser Stelle sagen, dass sich der finanzielle Aufwand, der für den Kauf eines LPB-ClipIn-Moduls des Typs OCI420 zusätzlich geleistet werden muss, u.U. nicht 'lohnt'. Dies ist jedoch abhängig vom verfolgten Ziel. Will man nur Temperaturen loggen um einen groben Überblick über den Ist-Zustand des Heizungssystems zu erhalten, so ist u.U. eine günstigere Lösung mit einer entsprechenden DS18B20-Temperatursensoren-Installation ausreichend. Will man hingegen auch bspw. Fehlereinträge aus der Ferne abrufen können, so lässt sich dies nur über die Lösung OCI420+BSB-LAN-Setup erreichen.

Es sei an dieser Stelle aber nochmals ausdrücklich darauf hingewiesen, dass eine Beeinflussung des Brennerverhaltens (wenn überhaupt) nur in sehr eingeschränktem Umfang möglich ist!

Hinweise zum Anschluss und der Konfiguration des OCI420-ClipIns sind im [Kap. 10.2.6](#) zu finden.

Im Folgenden wird exemplarisch eine Komplettabfrage einer LMU64 mit SW v3.03 (also der 'aktuellsten' Version, bei der am meisten Parameter verfügbar sind) dargestellt - so bekommt man einen Eindruck, was bestenfalls via BSB-LAN abrufbar ist.

```
1 Uhrzeit und Datum - Uhrzeit: 12:30:55
2 Uhrzeit und Datum - Tag/Monat: 24.01.
3 Uhrzeit und Datum - Jahr: 2022
70 Bedieneinheit - Geräte-Version Bedienteil: 30.3
701 Heizkreis 1 - Präsenztaste (temporäre Abwesenheit): ---
709 Heizkreis 1 - Komfortsollwert Minimum: 10.0 °C
711 Heizkreis 1 - Komfortsollwert Maximum: 30.0 °C
720 Heizkreis 1 - Kennlinie Steilheit: 18
721 Heizkreis 1 - Kennlinie Verschiebung: 0.0 °C
730 Heizkreis 1 - Sommer-/ Winterheizgrenze: 21 °C
734 Heizkreis 1 - Raumsollabsenkung mit Schaltuhr: 10.0 °C
781 Heizkreis 1 - Schnellabsenkung Faktor: 4
850 Heizkreis 1 - Estrichfunktion: 0 - Aus
884 Heizkreis 1 - Drehzahlstufe Auslegungspunkt: 17
885 Heizkreis 1 - Pumpe-PWM Minimum: 41 %
886 Heizkreis 1 - Norm Außentemperatur: -20 °C
887 Heizkreis 1 - Vorlaufsoll Norm Außentemperatur: 72.0 °C
888 Heizkreis 1 - dt Überhöhungsfaktor: 25.0 %
889 Heizkreis 1 - Filterzeitkonstant Drehzahlregler: 99 %
```



```

891 Heizkreis 1 - dT Vorhaltezeit Tv: 0 s
892 Heizkreis 1 - dT Nachstellzeit Tv: 50 s
893 Heizkreis 1 - dT Abtastfaktor: 10
894 Heizkreis 1 - dT Spreizung Norm Außentemperatur: 20.0 °C
895 Heizkreis 1 - dT Spreizung Maximum: 20.0 °C
1001 Heizkreis 2 - Präsenztaste (temporäre Abwesenheit): ---
1020 Heizkreis 2 - Kennlinie Steilheit: 14
1021 Heizkreis 2 - Kennlinie Verschiebung: 0.0 °C
1130 Heizkreis 2 - Mischerüberhöhung: 10.0 °C
1134 Heizkreis 2 - Antrieb Laufzeit: 150 s
1150 Heizkreis 2 - Estrichfunktion: 0 - Aus
1301 Heizkreis 3/P - Präsenztaste (temporäre Abwesenheit): ---
1603 Trinkwasser - Manueller TWW-Push: Aus
1614 Trinkwasser - TWW Nennsollwert Maximum: 2.2 °C
1615 Trinkwasser - TWW Reduziertollsollwert Minimum: 0.3 °C
2210 Kessel - Sollwert Minimum: 20.0 °C
2212 Kessel - Sollwert Maximum: °C 000084 - decoding error
2215 Kessel - Max. Regeldifferenz ohne Abbruch der Mindestpause (°K): 60.0 °C
2250 Kessel - Pumpennachlaufzeit: 10 min
2440 Kessel - Gebläse-PWM Hz Maximum: 70 %
2441 Kessel - Gebläsedrehzahl Hz Maximum: 5400 U/min
2442 Kessel - Gebläse-PWM Reglerverzögerung: 19 %
2443 Kessel - Gebläse-PWM Startwert DLH: 40 %
2444 Kessel - Leistung Minimum: 4.0 kW
2445 Kessel - Nennleistung Kessel: 15.0 kW
2451 Kessel - Brennerpausenzeit Minimum: 120 s
2452 Kessel - Schaltdifferenz Brennerpause: 30.0 °C
2453 Kessel - Reglerverzögerung Dauer: 60 s
2454 Kessel - Schaltdifferenz ein HK: 4.0 °C
2455 Kessel - Schaltdifferenz aus Min HK: 5.0 °C
2456 Kessel - Schaltdifferenz aus Max HK: 7.0 °C
2458 Kessel - Einschwingzeit TWW Max: 3 min
2459 Kessel - Sperrzeit dynamische Schaltdifferenz: 0 s
2471 Kessel - Pumpennachlaufzeit HK: 10 min
2472 Kessel - Pumpennachlauftemperatur TWW: 72.0 °C
2473 Kessel - Abgastemperatur Leistungsreduktion: 80 °C
2521 Kessel - Frostschutz Einschalttemperatur: 5.0 °C
2522 Kessel - Frostschutz Ausschalttemperatur: 10.0 °C
2547 Kessel - Abtastintervall DLH: 0.2 s
2730 Sitherm Pro - Ionisationsstrom: 131.80 µA
2792 Wärmepumpe - Pumpendrehzahl Minimum: 41 %
3810 Solar - Temperaturdifferenz ein: 8.0 °C
3811 Solar - Temperaturdifferenz aus: 4.0 °C
3812 Solar - Ladetemperatur Min TWW-Speicher: 30.0 °C
3831 Solar - Mindestlaufzeit Kollektorpumpe: 60 s
3840 Solar - Kollektortemperatur Frostschutz: 0.0 °C
3850 Solar - Kollektorüberhitzschutz: 120.0 °C
5019 Trinkwasserspeicher - TWW Nachladeüberhöhung Schichtenspeicher: 5.0 °C
5020 Trinkwasserspeicher - TWW Vorlaufsollwertüberhöhung: 18.0 °C
5050 Trinkwasserspeicher - TWW Ladetemperatur Maximum: 80.0 °C
5055 Trinkwasserspeicher - TWW Rückkühltemperatur: 80.0 °C
5100 Trinkwasserspeicher - TWW Pumpe-PWM Durchladung: 18 %
5126 Trinkwasserspeicher - Mischer Nachstellzeit Tn: 100 s
5400 Trinkwasser Durchlauferhitzer - Komfortsollwert: 45.0 °C
5420 Trinkwasser Durchlauferhitzer - Vorlauf-Sollwertüberhöhung: °C 0024 - decoding error
5430 Trinkwasser Durchlauferhitzer - Einschaltdifferenz im BW-Betrieb (Fühler 1): 4.0 °C
5431 Trinkwasser Durchlauferhitzer - Min. Ausschaltdifferenz im BW-Betrieb (Fühler 1): 2.0 °C
5450 Trinkwasser Durchlauferhitzer - Schwelle zum Beenden einer BW-Zapfung bei DLH: 0.20313 K/s
5451 Trinkwasser Durchlauferhitzer - Schwelle für Bw-Zapfung bei DLH in Komfort: -0.20313 K/s
5452 Trinkwasser Durchlauferhitzer - Schwelle für Bw-Zapfung bei DLH in Heizbetrieb: -0.29688 K/s
5453 Trinkwasser Durchlauferhitzer - Sollwertkorrektur bei Komfortregelung mit 40°C (°K): 0.0 °C
5454 Trinkwasser Durchlauferhitzer - Sollwertkorrektur bei Komfortregelung mit 60°C (°K): 0.0 °C
5455 Trinkwasser Durchlauferhitzer - Sollwertkorrektur bei Auslaufregelung mit 40°C (°K): 0.0 °C
5456 Trinkwasser Durchlauferhitzer - Sollwertkorrektur bei Auslaufregelung mit 60°C (°K): 0.0 °C
5480 Trinkwasser Durchlauferhitzer - Komfortzeit ohne Hz-Anforderung: min 000000 - unknown type
5481 Trinkwasser Durchlauferhitzer - Komfortzeit mit Hz-Anforderung: 0 min
5482 Trinkwasser Durchlauferhitzer - Zeit TWW-FlowSwitch geschlossen: 0 s
5483 Trinkwasser Durchlauferhitzer - Zeit TWW-Komfort FlowSwitch geschlossen: 0 s
5486 Trinkwasser Durchlauferhitzer - Reglerverzögerung bei Takten in DLH Auslaufbetrieb: 0 s
5487 Trinkwasser Durchlauferhitzer - Pumpennachlauf Komfort in Minuten: 255 min
5488 Trinkwasser Durchlauferhitzer - Pumpennachlauf Komfort in Sekunden: s 0000000000 - decoding error
5701 Konfiguration - Hydraulisches Schema: 58
5732 Konfiguration - TWW Pumpenpause Umsch UV: 4 s
5733 Konfiguration - TWW Pumpenpause Verzögerung: 1 s
5761 Konfiguration - Zubringerpumpe Q8 Bit 0-3: 00000000 Zonen mit ZubringerpumpeHK1 mit ZubringerpumpeHK2 mit ZubringerpumpeTWW
mit Zubringerpumpe
5920 Konfiguration - Programmierbarer Ausgang K2: 0 - Default, Keine Funktion
5922 Konfiguration - Relaisausgang 1 RelCl: 0 - Default, Keine Funktion
5923 Konfiguration - Relaisausgang 2 RelCl: 0 - Default, Keine Funktion
5924 Konfiguration - Relaisausgang 3 RelCl: 0 - Default, Keine Funktion
5926 Konfiguration - Relaisausgang 1 SolCl: 0 - Default, Keine Funktion
5927 Konfiguration - Relaisausgang 2 SolCl: 0 - Default, Keine Funktion
5928 Konfiguration - Relaisausgang 3 SolCl: 0 - Default, Keine Funktion
5950 Konfiguration - Funktion Eingang H1: 0 - Default
5973 Konfiguration - Funktion Eingang RelCl: 0 - Keine Funktion
5975 Konfiguration - Externer Vorlaufsollwert Maximum: 100.0 °C
5976 Konfiguration - Externe Leistungsvorgabe Schwelle: 5 %
5978 Konfiguration - Funktion Eingang SolCl: 0 - kein

```

```

6089 Konfiguration - Mod Pumpe Drehzahlstufen: 30
6092 Konfiguration - Mod Pumpe PWM: 3 %
6093 Konfiguration - Mod Pumpe PWM Max: 85 %
6127 Konfiguration - Pumpen/Ventilkick Dauer: 5 s
6140 Konfiguration - Wasserdruck Maximum: 2.5 bar
6141 Konfiguration - Wasserdruck Minimum: 0.7 bar
6143 Konfiguration - Wasserdruckschwelle für Kessel und Pumpe aus: 0.5 bar
6144 Konfiguration - Schaltdifferenz Wasserdruck: 0.3 bar
6146 Konfiguration - Druckwert 3.5V: 0.0 bar
6220 Konfiguration - Software-Version: 30.3
6222 Konfiguration - Gerätebetriebsstunden: 19042 h
6223 Konfiguration - Bisher unbekannte Geräteabfrage: 003A - decoding error
6224 Konfiguration - Geräte-Identifikation: LMU54/64
6225 Konfiguration - Gerätefamilie: 64
6226 Konfiguration - Gerätevariante: 100
6227 Konfiguration - Objektverzeichnis-Version: 202.0
6229 Konfiguration - EEPROM-Version: 15.0
6230 Konfiguration - KonfigRg0 Bit 0-7: 00100011 Fühlerunterbruch Rücklauffühler unterdrückenFühlerunterbruch Rücklauffühler ausgebenFühlerunterbruch Trinkwasserfühler unterdrückenFühlerunterbruch Trinkwasserfühler ausgebenFühlerunterbruch Abgastemperaturfühler unterdrückenFühlerunterbruch Abgastemperaturfühler ausgebenFühlerunterbruch Außentemperaturfühler unterdrückenFühlerunterbruch Außentemperaturfühler ausgebenFühlerunterbruch Wasserdruckfühler unterdrückenFühlerunterbruch Wasserdruckfühler ausgebenFühlerunterbruch Vorlauffühler unterdrückenFühlerunterbruch Vorlauffühler ausgebenFühlerunterbruch Fühler am ClipIn unterdrückenFühlerunterbruch Fühler am ClipIn ausgeben
6240 Konfiguration - KonfigRg1 Bit 0-7: 00110100 Vorrang Trinkwasser: absolutVorrang Trinkwasser: gleitendVorrang Trinkwasser: keinKlemmenbelegung X10-02: RTKlemmenbelegung X10-02: SchaltuhrKlemmenbelegung X10-01: RTKlemmenbelegung X10-01: SchaltuhrAnlagenfrostschutz ausAnlagenfrostschutz einWirkung Schaltuhr an X10-01: HeizkreisWirkung Schaltuhr an X10-01: TWWBetriebsart des Heizkreises bei Modemfunktion: StandbyBetriebsart des Heizkreises bei Modemfunktion: Reduziert
6250 Konfiguration - KonfigRg2 Bit 0-7: 10000000 Pumpennachlauf TWW in den HeizkreisPumpennachlauf TWW in den TWW-WärmetauscherKomforttemperaturniveau wie AuslaufftemperaturKomforttemperaturniveau wie BereitschaftstemperaturKomfort PID Regelungsfühler: KesselvorlaufKomfort PID Regelungsfühler: TWW-FühlerKomfort PID Regelungsfühler: KesselrücklaufFühleranordnung Schichtenspeicher SSP: KVFFühleranordnung Schichtenspeicher SSP: KRF, nicht bei SSPFühleranordnung Schichtenspeicher SSP: KRF, nur SSP
6270 Konfiguration - KonfigRg4 Bit 0-7: 01000000 Zubringerfunktion Q8: ausZubringerfunktion Q8: einGebäudebauweise: leichtGebäudebauweise: schwerAnschlussklemme TW-Thermostat: Fühler an X10/TWFAanschlussklemme TW-Thermostat: Thermostat an X10/TWFKonfiguration 3-Wege-Ventil: keinKonfiguration 3-Wege-Ventil: MagnetventilKonfiguration 3-Wege-Ventil: MotorventilKonfiguration 3-Wege-Ventil: Motorventil invers
6280 Konfiguration - KonfigRg5 Bit 0-7: 00000111 Wassermangelsicherung Strömungswächter->StörstellungWassermangelsicherung Strömungswächter->StartverhinderungWassermangelsicherung Wasserdrukchwächter->StörstellungWassermangelsicherung Wasserdrukchwächter->StartverhinderungWassermangelsicherung Drehzahlbegrenzung ausWassermangelsicherung Drehzahlbegrenzung einBits 3-7: Grundeinstellung; nicht verändern!
6290 Konfiguration - KonfigRg6 Bit 0-7: 11100000 Initialisierung ISR: Sofortige Übernahme der Stellgröße nach FreigabeInitialisierung ISR: Stoßfreier Übergang nach FreigabeVerriegelung Fremd-Room Units: ausVerriegelung Fremd-Room Units: einQuelle BW-Sollwert: R UQuelle BW-Sollwert: HMISperrsignalberechnung deaktiviertSperrsignalberechnung aktiviertSchnelle Wechsel der Drehzahlgrenzen: Normal e AbarbeitungSchnelle Wechsel der Drehzahlgrenzen: BeschleunigtInitialisierung ISR: Nur bei BetriebsartwechselInitialisierung ISR: Immer bei BrennerstartPWM-Gebläserampen: ausPWM-Gebläserampen: ein
6300 Konfiguration - KonfigRg7 Bit 0-7: 00001111 Heizkreispumpe: stufigHeizkreispumpe: modulierendDelta-T-Begrenzung: ausDelta-T-Begrenzung: einDelta-T-Überwachung: ausDelta-T-Überwachung: einAnlagenvolumen: kleinAnlagenvolumen: mittelAnlagenvolumen: großDelta-T-Überwachung im Reduziert-Betrieb: ausDelta-T-Überwachung im Reduziert-Betrieb: einReglerverzögerung: nur im Heizbetrieb aktivReglerverzögerung: in allen Betriebsarten aktivPumpe Q1 Modulation in Systemen 51,54,55,67,70,71: ausPumpe Q1 Modulation in Systemen 51,54,55,67,70,71: ein
6310 Konfiguration - KonfigRg8 Bit 0-7: 00000000 Sekundärtauscher: PlattenwärmetauscherSekundärtauscher: Wendelwärmetauscher primärseitigSekundärtauscher: Wendelwärmetauscher sekundärseitigl. Maximum Durchlauferhitzer-Regelung: auswertenl. Maximum Durchlauferhitzer-Regelung: ignorierenDurchlauferhitzer Anfrage über DLH1 Sensor oder Flow-SwitchDurchlauferhitzer Anfrage nur über Flow-SwitchPosition von Verteilventil nach Durchlauferhitzer: LetztePosition von Verteilventil nach Durchlauferhitzer: Heizposition
6330 Konfiguration - KonfigRg10 Bit 0-7: 00000000 Kesselpumpe bei Brennersperre: keine AbschaltungKesselpumpe bei Brennersperre: AbschaltungBrennersperre nur bei HeizanforderungBrennersperre nur bei Heiz- und TWW-AnforderungDurchlauferhitzertaktschutz durch Temperaturerhöhung: ausDurchlauferhitzertaktschutz durch Temperaturerhöhung: einPosition des TWW-Speichers: vor hydraulischer WeichePosition des TWW-Speichers: nach hydraulischer WeicheZubringerpumpe bei Brennersperre: keine AbschaltungZubringerpumpe bei Brennersperre: AbschaltungGebläseabschaltung bei Heizanforderung: FreigabeGebläseabschaltung bei Heizanforderung: keine FreigabePumpennachlauf TWW bei weiteren Wärmeanforderungen: UnterdrückenPumpennachlauf TWW bei weiteren Wärmeanforderungen: Durchführen
6343 Konfiguration - Bisher unbekannte Geräteabfrage: 0001260096 - unknown type
6344 Konfiguration - Hersteller-ID (letzten vier Bytes): 1012140155
1012140155
6600 LPB-System - Geräteadresse: 00.01
6604 LPB-System - Busspeisung Funktion: 1 - Automatisch
6605 LPB-System - Busspeisung Status: 255 - Ein
6610 LPB-System - Anzeige Systemmeldungen: 255 - Ja
6625 LPB-System - Trinkwasserzuordnung: 0 - Lokale Heizkreise
6640 LPB-System - Uhrbetrieb: 2 - Slave mit Fernverstellung
6650 LPB-System - Außentemperatur Lieferant: 00.01
6699 LPB-System - Software Version Einschub: 0.7
6701 Fehler - Unknown command: 0000 - unknown type
6705 Fehler - SW Diagnosecode: 0 - not found
6800 Fehler - Historie 1: 5
6803 Fehler - Historie 1 Fehlermeldung: 153 - Entriegelungstaste wurde betätigt
6805 Fehler - SW Diagnosecode 1: 259
6806 Fehler - FA Phase 1: 0 - Standby
6810 Fehler - Historie 2: 4
6813 Fehler - Historie 2: 133 - Keine Flammenbildung nach Ablauf der Sicherheitszeit
6815 Fehler - SW Diagnosecode 2: 102 - Uhrzeitmaster ohne Gangreserve (LPB)
6816 Fehler - FA Phase 2: 14 - not found
6820 Fehler - Historie 3: 0
6823 Fehler - Historie 3 Fehlermeldung: 0 - kein Fehler
6825 Fehler - SW Diagnosecode 3: 0 - not found
6826 Fehler - FA Phase 3: 0 - Standby
6830 Fehler - Historie 4: 0
6833 Fehler - Historie 4 Fehlermeldung: 0 - kein Fehler
6835 Fehler - SW Diagnosecode 4: 0 - not found

```

```

6836 Fehler - FA Phase 4: 0 - Standby
6840 Fehler - Historie 5: 0
6843 Fehler - Historie 5 Fehlermeldung: 0 - kein Fehler
6845 Fehler - SW Diagnosecode 5: 0 - not found
6846 Fehler - FA Phase 5: 0 - Standby
7001 Wartung/Sonderbetrieb - Meldung: 0
7010 Wartung/Sonderbetrieb - Quittierung Meldung: 1 - Ein
7011 Wartung/Sonderbetrieb - Repetitionszeit Meldung: 14 Tage
7040 Wartung/Sonderbetrieb - Repetitionszeit Meldung: h 0001499700 - decoding error
7041 Wartung/Sonderbetrieb - Brennerstunden seit Wartung: h 0002255100 - decoding error
7042 Wartung/Sonderbetrieb - Brennerstarts Intervall: 0
7043 Wartung/Sonderbetrieb - Brennerstarts seit Wartung: 400
7044 Wartung/Sonderbetrieb - Wartungsintervall: 6000 h
7045 Wartung/Sonderbetrieb - Zeit seit Wartung: 0 Monate
7050 Wartung/Sonderbetrieb - Gebläsedrehzahl Ion Strom: 4000 U/min
7051 Wartung/Sonderbetrieb - Meldung Ion Strom: 0
7700 Ein-/Ausgangstest - Relaistest: 0 - Kein Test
8310 Diagnose Erzeuger - Kesseltemperatur: 76.0 °C
8311 Diagnose Erzeuger - Kesselsollwert: 2.3 °C
8314 Diagnose Erzeuger - Kesselrücklauftemperatur Ist: 57.5 °C
8321 Diagnose Erzeuger - Aktuelle Regeldifferenz: 4.4 °C
8324 Diagnose Erzeuger - Brennergebläsesollwert: 0 U/min
8325 Diagnose Erzeuger - PWM Drehzahlregler (Proz): 0 %
8326 Diagnose Erzeuger - Brennermodulation: 0 %
8329 Diagnose Erzeuger - Ionisationsstrom: 0.00 µA
8336 Diagnose Erzeuger - Betriebsstunden Brenner: 19042 h
8337 Diagnose Erzeuger - Startzähler Brenner: 3467
8338 Diagnose Erzeuger - Betriebsstunden Heizbetrieb: 57508 h
8339 Diagnose Erzeuger - Betriebsstunden TWW: 2680 h
8340 Diagnose Erzeuger - Betriebsstunden externe Wärmeanforderung: 0 h
8530 Diagnose Erzeuger - Betriebsstunden Solarertrag: 0 h
8700 Diagnose Verbraucher - Außentemperatur: 6.5 °C
8705 Diagnose Verbraucher - Außentemperatur LPB: --- °C
8741 Diagnose Verbraucher - Raumsollwert 1: 30.0 °C
8750 Diagnose Verbraucher - Mod Pumpe Sollwert: 60 %
8773 Diagnose Verbraucher - Vorlauftemperatur 2: 36.0 °C
8830 Diagnose Verbraucher - Trinkwassertemperatur 1: --- °C
8831 Diagnose Verbraucher - Trinkwassersollwert: 55.0 °C
8832 Diagnose Verbraucher - Trinkwassertemperatur 2: 52.0 °C
8950 Diagnose Verbraucher - Schienenvorlauftemperatur: 77.5 °C
8952 Diagnose Verbraucher - Schienenrücklauftemperatur: 58.0 °C
9500 Feuerungsautomat - Vorlüftzeit: 3.0 s
9502 Feuerungsautomat - Gebläseansteuerung Vorlüftung: 40 %
9504 Feuerungsautomat - Solldrehzahl Vorlüftung: 3500 U/min
9510 Feuerungsautomat - Gebläseansteuerung Zündung: 40 %
9512 Feuerungsautomat - Solldrehzahl Zündung: 3500 U/min
9520 Feuerungsautomat - Gebläseansteuerung Betrieb. Min: 14 %
9522 Feuerungsautomat - Gebläseansteuerung Betrieb. Max: 70 %
9524 Feuerungsautomat - Solldrehzahl Betrieb Min: 1450 U/min
9527 Feuerungsautomat - Solldrehzahl Betrieb Max: 5400 U/min
9540 Feuerungsautomat - Nachlüftzeit: 2.0 s
9560 Feuerungsautomat - Gebläseansteuerung Durchladung: 24.0 %
9563 Feuerungsautomat - Solldrehzahl Durchladung: 3950 U/min

```

## 10.2.5 Hinweis: Spezialfall Weishaupt-Geräte

Einige Weishaupt-Geräte (s. Auflistung der erfolgreich getesteten Systeme: Weishaupt WTU mit Bedieneinheit WRS-CPU) haben einen Regler des Typs RVS23 verbaut. Dieser Reglertyp weist einen LPB auf, auf dem bereits die bestehende Installation dieser Weishaupt-Anlagen basiert: Raumgeräte, Bedieneinheiten und Erweiterungsmodule sind bereits miteinander via LPB verbunden.

An diesem LPB ist ebenfalls der Adapter anschließbar, er muss jedoch korrekt in die bestehende Installation eingebunden werden. In der Regel stellt dies mit der voreingestellten LPB-Adresse des Adapters (Segment 4, Adresse 3) kein Problem dar, sollte aber bei etwaigen Kommunikationsproblemen ggf. nochmal überprüft werden.

Auch bei den Weishaupt-Geräten scheint es neben der kesselseitigen Bedieneinheit eine Servicebuchse zu geben, bei der von den vier vorgesehenen Pins zwei belegt und herausgeführt sind. Laut der Aussage eines Weishaupt-Nutzers (*Danke an BSB-LAN-User Philippe!*) scheint hier der obere der beiden Pins MB und der untere der beiden Pins DB zu sein.

## 10.2.6 Hinweis: LPB nachrüsten mittels OCI420 ClipIn-Modul

Soll ein OCI420 an einem LMU-Regler (LMU64/LMU7x) angeschlossen und verwendet werden, so ist die Installation und der Anschluss prinzipiell gemäß den jeweiligen Bedienungsanleitungen vorzunehmen.

Es gibt jedoch ein paar wichtige Punkte, die i.d.R. nicht in den jeweiligen Anleitungen zu finden sind, obwohl sie für einen erfolgreichen Betrieb entscheidend sind. Dies betrifft vor allem die Einstellungen, die für die LPB-Spannungsversorgung vorzunehmen sind. Des Weiteren ist die LPB-Geräteadresse 1 mit Segmentadresse 0 einzustellen und die Einstellung als Uhrzeit-Master vorzunehmen.

Die folgenden Angaben sind wie immer ohne Gewähr – darauf sei an dieser Stelle nochmal explizit hingewiesen.

Schließt man das OCI420 den Anleitungen folgend an, so wird höchstwahrscheinlich der Fehler 81 an der Bedieneinheit angezeigt werden, welcher „Kurzschluss im LPB Bus oder *fehlende LPB-Busspeisung*“ bedeutet. Sofern man das OCI420 korrekt angeschlossen hat, muss in dem Fall die LPB-Busspeisung des Reglers via Bedieneinheit aktiviert werden. Der Parameter dazu ist „LPBKongig0“.

Die folgenden Einstellungen sind für Regler des Typs LMU64 beschrieben, bis auf die Parameternummer sind die Einstellungen der Bits bei anderen LMx-Reglern identisch.

Die Einstellungen müssen an der Bedieneinheit des Reglers vorgenommen werden.

Bei der LMU64 hat der betreffende Parameter die Nummer 604 (bei LMU74: Parameternummer 6006). Hier sind acht Bits (604.0 bis 604.7) verfügbar, die wie folgt einzustellen sind (dabei bedeutet „0“=AUS und „1“=EIN):

604.0 = 0 → Uhrzeitmaster

604.1 = 1 → Uhrzeitmaster

**604.2 = 1 → Verteilte Busspeisung AUTOMATIK**

604.3 = 1 → Status LPB-Busspeisung: 1 = aktiv

604.4 = 1 → Ereignisverhalten erlaubt

604.5 = 0 → Brauchwasserzuordnung lokal

604.6 = 0 → Brauchwasserzuordnung lokal

604.7 = 0 → Kein Vorrang LMU-Anforderung vor externer Leistungsvorgabe

Ruft man die ‚Übersicht‘ der LPBKongig0-Einstellungen auf, so wird dort jedoch die Bit-Reihenfolge von hinten nach vorne (also von Bit 7 bis Bit 0!) dargestellt und sollte nach erfolgreicher Einstellung folgendermaßen lauten: 00011110. Des Weiteren sind folgende Einstellungen vorzunehmen:

605 LPB-Geräteadresse = 1

606 LPB-Segmentadresse = 0

Nach erfolgreicher Einstellung sollte kein Fehlercode mehr auftreten und die rote LED am OCI420 in regelmäßigen Abständen blinken.

Sollte hingegen Fehler 82 angezeigt werden, so müssen die LPB-Adressen überprüft werden, da in dem Fall eine LPB-Adresskollision vorliegt (was aber bei den o.g. Einstellungen und der Standardadresse von BSB-LAN nicht auftreten sollte).

## 10.3 Erweiterungs- und Cliquin-Module

Sollten die Anschlussmöglichkeiten und der Funktionsumfang der genannten Regler im Einzelfall nicht ausreichen, bspw. weil ein zusätzlicher Pumpenkreis nachträglich installiert wird, so lassen sich jeweils spezielle Erweiterungs-/Cliquin-Module (im Folgenden EWM) verbauen. Die EWM bieten Anschlussmöglichkeiten für einen Pumpen- bzw. Mischerkreis samt zugehöriger Sensoren.

Diese EWM werden am Regler mittels eines speziellen Bus-Kabels an einem dedizierten Anschluss angeschlossen und kommunizieren intern über den BSB (eine Ausnahme scheint Weishaupt zu sein, worauf in diesem Kapitel jedoch nicht weiter eingegangen wird). Die EWM selbst weisen keinen eigenen BSB- oder LPB-Anschluss auf. Die Parametrierung erfolgt über die Bedieneinheit des Reglers.

Der Zugriff auf ein EWM ist somit nur indirekt über die jeweils spezifischen Parameter im eigentlichen Regler möglich, die die Einstellungen und Funktionen des EWMs definieren und beschreiben. Da sie jedoch bspw. beim Aufruf von `ip/o` mit aufgelistet werden, stelle ich sie im Folgenden kurz vor.

*Hinweis:*

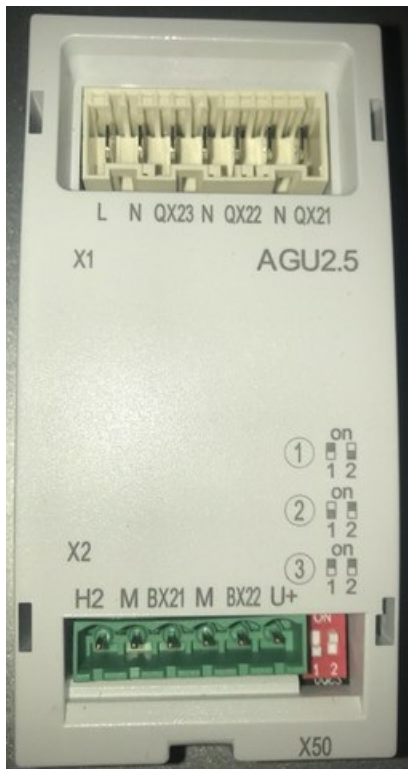
*Soll ein Erweiterungs-/Cliquin-Modul nachgerüstet werden, so sollte selbstverständlich das anlagenspezifische Handbuch berücksichtigt sowie ein Heizungsinstallateur beauftragt werden.*

Erweiterungsmodule des Typs **AVS75.xxx** kommen bei den Reglerreihen des Typs RVS und LMS zum Einsatz. Die Busanbindung erfolgt i.d.R. über den Anschluss "Bus-EM".



Erweiterungsmodul des Typs AVS75.390.

Erweiterungsmodule für LMU-Regler werden als "ClipIn-Module" bezeichnet. Je nach Verwendungszweck scheint es unterschiedliche Ausführungen zu geben (bspw. Relaismodul, Solarmodul). Generell scheinen sie jedoch die Typenbezeichnung **AGU2.5x** zu tragen (das "x" scheint dann die jeweilige Ausführung zu kennzeichnen), die Busanbindung erfolgt i.d.R. über den Anschluss "X50".



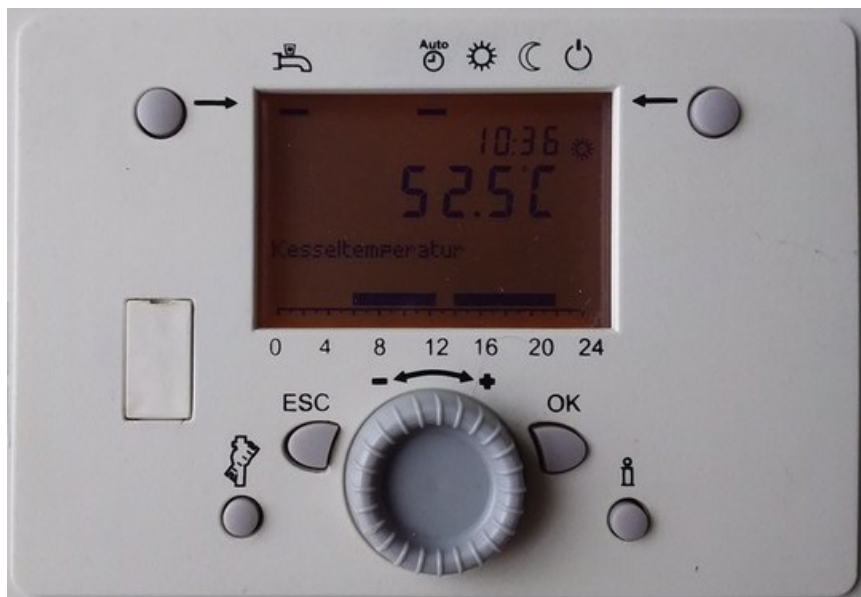
ClipIn-Modul des Typs AGU2.55.

## 10.4 Bedieneinheiten

Die Bedieneinheit (am Wärmeerzeuger selbst) der Systeme der letzten Jahre (mit den Reglertypen LMU7x, LMS1x, RVS) sind üblicherweise Modelle des Typs **AVS37.xxx**. Sie sehen herstellerübergreifend recht identisch aus, können aber bei bestimmten Systeme (bspw. Wärmepumpen) zusätzliche Bedienelemente oder Funktionen aufweisen. Wenn man das Aussehen dieser AVS37-Bedieneinheiten und der QAA75.61x-Raumgeräte vergleicht, so kann man feststellen, dass sich die



beiden Geräte sehr ähneln. Die Art der Bedienung ist in dem meisten Fällen ebenso identisch. Die heizungsseitigen Bedieneinheiten stellen i.d.R. die Temperatur des Wärmeerzeugers (bspw. Kesseltemperatur) dauerhaft dar, die Raumgeräte hingegen üblicherweise die Raumtemperatur. Beide Geräte senden den jeweiligen Wert regelmäßig (etwa alle zehn Sekunden) als Broadcast (INF-Nachricht) über den BSB.



Eine Bedieneinheit des Typs AVS37.xxx.

Als Nachfolger des Modells AVS37 gilt das **AVS74.xxx**. Es ist mit einem 3,8" LCD-Display und einem Dreh-/Drückknopf ausgestattet, mit dem sämtliche Einstellungen vorgenommen werden. Es kommt ebenfalls als Raumgerät unter der Bezeichnung [QAA74](#) zum Einsatz.



Eine AVS74.xxx Bedieneinheit.

In jüngerer Zeit wird von einigen Herstellern bei bestimmten Geräteserien ein neuer Typ von Bedieneinheit verbaut, die Modellbezeichnung lautet **QAA75.91x**. Die Bedieneinheit scheint (zumindest bei bestimmten Herstellern) abnehmbar und mithilfe eines Adapters (Brötje: "ISR RGA") im Wohnraum installiert und zusätzlich als Raumgerät genutzt werden kann. Die Bedienung des Wärmeerzeugers erfolgt in dem Fall weiterhin über diese Komponente.





Eine Bedieneinheit des Typs QAA75.91x.

Darüber hinaus gibt es ein weiteres Modell, das **AVS77.xxx**. Diese Bedieneinheit ist uns bisher nur bei einem Baxi-Modell (Baxi Luna Duo Tec MP) gemeldet worden. Dieses Modell weist u.a. Knöpfe für die gradweise Veränderung des TWW- und HK-Sollwertes auf, verfügt jedoch nicht mehr über einen Drehknopf.



Eine Bedieneinheit des Typs AVS77.xxx.

#### Hinweis

Es scheint, als ob eine Änderung der Betriebsart mittels der TWW-/HK-Taste (in der Mitte der Bedieneinheit) bei bestimmten Einstellungen zur Folge hat, dass eine Änderung der Betriebsart mittels BSB-LAN und den 'regulären' Parametern 1600 (TWW) und 7xx (HK) nicht mehr möglich ist, wenn eine andere Einstellung als "beide" mittels erwähnter Taste ausgewählt wurde. Die Funktion dieser neuen Taste ist in BSB-LAN (bisher) nicht hinterlegt. Soll eine Änderung der Betriebsarten via BSB-LAN gewünscht sein, so ist darauf zu achten, dass die Betriebsart mittels erwähnter Taste auf "beide" eingestellt ist.

## 10.5 Konventionelle Raumgeräte für die aufgeführten Reglertypen

Im Folgenden wird kurz auf die unterschiedlichen Raumgeräte eingegangen. Auch diese werden prinzipiell von SIEMENS hergestellt und von den verschiedenen Heizungsherstellern gebrandet. Somit sind sie herstellerübergreifend einsetzbar, d.h. ein entsprechendes QAA-Raumgerät von bspw. Elco kann prinzipiell an einer Brötje-Heizung eingesetzt werden (natürlich immer vorausgesetzt, dass es sich um das richtige Modell handelt). Ob dabei in Einzelfällen gewisse Einschränkungen bestehen, ist bisher nicht bekannt bzw. bei Tests nicht aufgefallen.

Die nachfolgende Beschreibung beginnt dabei mit den Raumgeräten für die aktuellen Heizungsregler (RVS und LMS), die auch von BSB-LAN voll unterstützt werden (sog. "Broetje ISR").

Anmerkung: Es scheint, als wenn das Produktportfolio um neue Raumgeräte und weiteres Zubehör ergänzt wurde. Bei Gelegenheit werde ich die m.E. relevanten Produkte hier hinzufügen.

### 10.5.1 QAA55 / QAA58

Das QAA55 ist das ‚kleinste‘ und günstigste ISR-Raumgerätemodell. Bei Brötje wird es als „RGB B“ geführt, manchmal ist es auch als „Raumgerät Basic“, „ISR RGB“ o.ä. zu finden. Es ist im Funktionsumfang recht eingeschränkt und ist im Grunde mehr als Raumtemperaturfühler mit einigen wenigen zusätzlichen Bedienoptionen anzusehen.



*Das QAA55 Raumgerät.*

Neben der optionalen Messung der Raumtemperatur bietet es eine Präsenztaste und die Möglichkeiten zur Umschaltung der Betriebsart sowie zur Veränderung der Raumsolltemperatur. Es verfügt lediglich über ein kleines LCD-Display, das die aktuelle Raumtemperatur anzeigt. Angeschlossen wird es über ein zweipoliges Kabel am BSB.

Das QAA58 ist die Funkvariante des QAA55. Es ist batteriebetrieben, der Funkempfänger AVS71.390 (Frequenz 868 MHz) muss wiederum per Kabel am Anschluss X60 des Kesselreglers angeschlossen werden.

### 10.5.2 QAA75 / QAA78

Das QAA75.61x ist das ‚große‘ ISR-Raumgerät. Es weist neben dem integrierten Temperaturfühler den vollen Funktionsumfang der kesselseitigen Bedieneinheit auf. Zusätzlich ist eine Präsenztaste vorhanden, ein manueller TWW-Push kann bei Bedarf i.d.R. durch längeres Drücken der TWW-Betriebsarttaste ausgelöst werden.



Das QAA75.61x Raumgerät.

Das QAA75.61x heißt bei Brötje „Raumgerät RGT“, manchmal ist es auch als „Raumgerät RGT B Top“, „ISR RGT“ o.ä. zu finden. Es wird ebenfalls per Kabel am BSB angeschlossen, wobei ein dritter Anschluss für die optional nutzbare Hintergrundbeleuchtung vorhanden ist (Klemme „G+“ am Regler).

Das QAA78.61x ist die Funkvariante des QAA75.61x. Es ist batteriebetrieben, der Funkempfänger AVS71.390 (Frequenz 868 MHz) muss wiederum per Kabel am Anschluss X60 des Kesselreglers angeschlossen werden. Die oben genannte Bezeichnung „RGT“ wird bei Brötje um ein „F“ erweitert, also „RGTF“.

#### Hinweis

An dieser Stelle muss zusätzlich erwähnt werden, dass es offenbar zwei verschiedene Ausführungen des QAA75 gibt: Das bereits erwähnte Raumgerät QAA75.61x und ein optisch nicht identisches QAA75.91x. Wann immer ich in diesem Handbuch das "QAA75" erwähne, so beziehe ich mich dabei auf das bereits vorgestellte Modell QAA75.61x.

Das QAA75.91x scheint im Bedienungsumfang identisch zum QAA75.61x zu sein, jedoch nur bei bestimmten Modellreihen einiger Hersteller (bspw. Brötje WMS/WMC C, BMK B, BMR B und Baxi Luna Platinum+) zum Einsatz zu kommen. Es scheint die 'heizungsseitige' Bedieneinheit zu sein, die jedoch mittels eines Adapters (Brötje: "ISR RGA") zusätzlich als Raumgerät genutzt werden kann. Die Bedienung des Wärmeerzeugers erfolgt in dem Fall weiterhin über diese Komponente, nur mit dem Vorteil, dass man sie im Wohnbereich installieren und sich ein zusätzliches Raumgerät sparen kann.



Eine QAA75.91x Bedieneinheit, mit Zubehör optional nutzbar als Raumgerät.

### 10.5.3 QAA74

Das QAA74 ist ein relativ neues Raumgerät, welches das QAA75 ablösen soll/wird. Bei Brötje heißt es "ISR RGP" ("Raumgerät Premium"), bei Siemens "UI400". Es ist mit einem 3,8" LCD-Display und einem Dreh-/Drückknopf ausgestattet, mit dem sämtliche Einstellungen vorgenommen werden. Es kommt bei einigen Modellen ebenfalls als heizungsseitige Bedieneinheit unter der Bezeichnung AVS74 zum Einsatz.



Das QAA74 Raumgerät.

### 10.5.4 Brötje IDA

Die „Brötje IDA“ ist eine Raumeinheit, die neben einem integrierten Temperaturfühler und einigen Funktionen zusätzlich einen gewissen Funktionsumfang für die Steuerung via App mit Smartphone bietet. Eine Präsenztaste ist nicht vorhanden.

IDA wird ins heimische WLAN integriert und benötigt Internetzugriff, falls man die Steuerung per App nutzen möchte. Bei einer rein lokalen Nutzung des Raumgerätes (ohne Fernzugriff via App) ist kein WLAN-Zugang erforderlich. Über den WLAN-Zugang erfolgt im Übrigen auch die Aktualisierung der IDA-Firmware.

Eine interessante Analyse des Datenverkehrs wurde [hier](#) von FHEM-Forumsmitglied „freetz“ vorgenommen.

Für den Anschluss am BSB des Kesselreglers muss ein BSB-Interface (GTW17) angeschlossen werden. Interessenten müssen in diesem Fall nach „ISR IDA“ Ausschau halten, damit das GTW17 im Paket enthalten ist.

Bei Reglern mit dem Kommunikationsprotokoll OpenTherm (bspw. die ältere Reglergeneration Brötje LMU6x) muss das OT-Interface (GTW16) verwendet werden.

IWR-CAN-basierte Regler (s. [Kap. 3.3](#)) werden direkt an das Service Dongle GW05 (WLAN-Gateway) angeschlossen.

Der genaue Funktionsumfang und die Installationsschritte von IDA sind bitte den entsprechenden Anleitungen des Herstellers zu entnehmen. Eine Übersicht ist bspw. unter der URL <https://www.broetje.de/de/produkte/regelung-und-vernetzte-heizung/isr/raumgeraet-isr-ida> verfügbar.

Der gleichzeitige Einsatz von IDA und BSB-LAN ist prinzipiell möglich, jedoch sind aufgrund des Erfahrungsberichtes eines Nutzers (*Danke an FHEM-Foumsmitglied „mifh“!*) ein paar Einschränkungen hinsichtlich des Funktionsumfangs von BSB-LAN bekannt:

Ist IDA am BSB angeschlossen, so ist es der Master für die Einstellungen bzw. Werte von

- Uhrzeit und Datum,
- Heiz- bzw. Schaltprogrammen sowie der
- Raumtemperatur. Werden diese Einstellungen / Werte via BSB-LAN geändert, so werden sie nach kurzer Zeit wieder mit den Einstellungen / Werten aus IDA überschrieben. Es ist somit also nicht mehr möglich, bspw. die Raumtemperaturen aus verschiedenen Räumen zu erfassen und mittels BSB-LAN an den Regler zu übermitteln, da IDA dies überschreibt.

Die Funktion der Präsenztaste ist via BSB-LAN i.d.R. nach wie vor gegeben.

### 10.5.5 QAA53 / QAA73

Die Raumgeräte QAA 53 und QAA 73 unterscheiden sich ebenfalls im Funktionsumfang. Zum Einsatz kommen sie bei den OpenTherm-basierten Reglern des Typs LMU6x. Sollte dieser Raumgerätetyp angeschlossen oder anschließbar sein, so ist der Regler nicht ohne Weiteres kompatibel mit dem BSB-LPB-LAN-Adapter! Für weitere Informationen diesbzgl. beachte bitte das [Kapitel 10.2.4](#). Weitere Informationen zu diesen Raumgeräten sind bitte den entsprechenden Anleitungen zu entnehmen.

### 10.5.6 QAA50 / QAA70

Auch beim QAA50 und QAA70 besteht prinzipiell der Unterschied im Funktionsumfang. Diese Raumgeräte kommen bei den alten Reglergenerationen zum Einsatz, die lediglich eine PPS-Schnittstelle aufweisen und somit prinzipiell kompatibel mit dem BSB-LPB-LAN-Adapter sind. Der Einsatz von BSB-LAN parallel zu einem vorhandenen Raumgerät ist in diesem Fall nur lesend möglich, Werte und Einstellungen des Heizungsreglers können also nicht via BSB-LAN verändert werden.



Ein QAA70 Raumgerät.

Weitere Informationen zu diesen Raumgeräten sind bitte den entsprechenden Anleitungen zu entnehmen.

## 10.6 Sonderzubehör: Webserver OZW672 und Servicetool OCI700

An dieser Stelle seien der Vollständigkeit halber noch zwei kommerzielle Lösungen erwähnt, mittels derer man Zugriff auf den Heizungsregler via Computer bekommen kann.

Dies ist zum einen der Webserver OZW672 und zum anderen das Servicetool OCI700.

Der Webserver OZW672 (Brötje: "ISR OZW") wird per Busleitung an den Regler angeschlossen und mit einem Netzwerkanschluss mit dem heimischen Netzwerk und ggf. dem Internet verbunden. Er stellt bei Bedarf eine Verbindung zum (kostenpflichtigen) 'Brötje Datenportal' her und bietet dann mittels Fernzugriff (via PC, Tablet oder Smartphone+App) Möglichkeiten wie Ferndiagnose oder auch Benachrichtigungen im



Störfall.

Das OCI700 ist das Servicetool, das überwiegend vom Fachhandwerker eingesetzt wird. Es wird lokal mit einem Rechner verbunden, auf dem eine spezielle Software installiert ist und ermöglicht einen Überblick über die Einstellungen des Reglers.

## 10.7 Fühlertypen

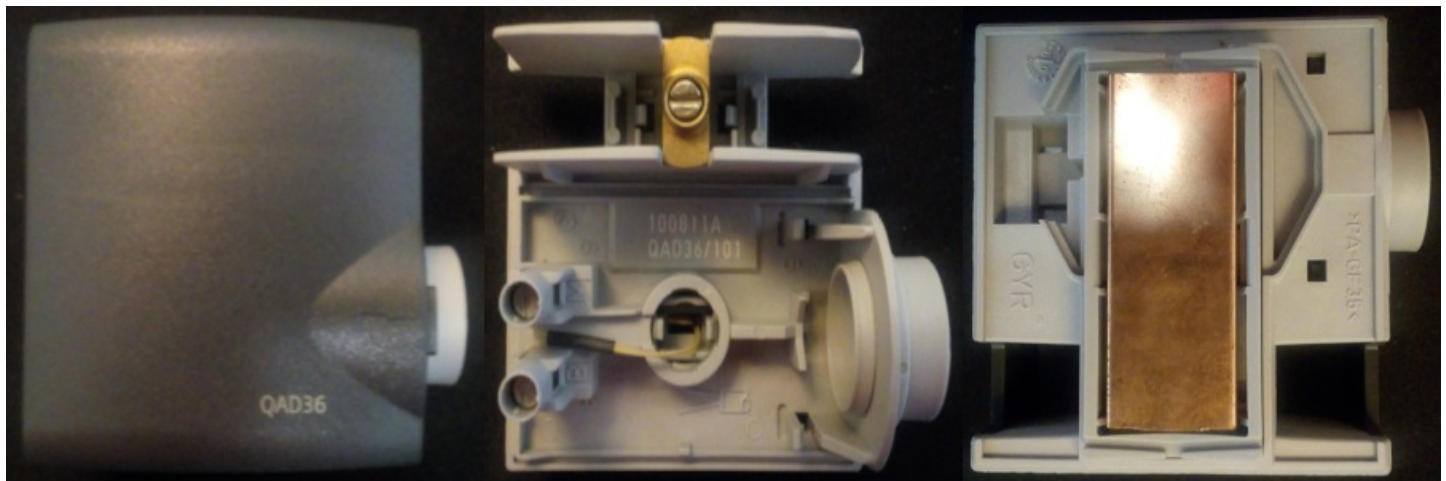
Hinweis
Die nachfolgende Auflistung ersetzt <i>nicht</i> die spezifische Beschreibung der verwendeten anlagenspezifischen Fühlertypen! Sie dient lediglich der Veranschaulichung und soll aufzeigen, dass u.U. bspw. auch die Verwendung von kostengünstigeren, universellen Anlegesensoren als Vor- oder Rücklauffühler möglich ist! Bei einem Austausch von Fühlern oder einer Erweiterung des Systems mit zusätzlichen Fühlern muss daher <i>immer</i> in der spezifischen Anlagendokumentation nachgesehen werden, welche Fühler anlagenspezifisch zum Einsatz kommen!

Bei den aktuellen Reglerreihen kommen (gelabelte) Siemens-Fühler zum Einsatz, die man wie folgt grob in zwei unterschiedliche Fühlertypen unterscheiden kann:

- der Außentemperaturfühler **QAC34** ist ein Fühler des Typs **NTC 1k Ohm** (NTC 1000 Ohm Widerstand bei 25°C, Widerstandskennlinie sinkend mit ca. 4%/K bei steigender Temperatur, Messbereich -50...70°C, Toleranz +/- 1K),
- die anderen Fühler **QAD36**, **QAL36**, **QAK36**, **QAR34**, **QAZ36** sind Fühler des Typs **NTC 10k Ohm** (NTC 10k Ohm Widerstand bei 25°C; Widerstandskennlinie sinkend bei steigender Temperatur).  
Diese Fühler unterscheiden sich jedoch u.a. in ihrer Bauform und dem Messbereich des eingesetzten Fühlerelements, sogar innerhalb einer Produktkategorie (bspw. QAD36 in der Ausführung als BMU- oder Kollektorfühler), so dass ein Ersatz durch universelle und kostengünstigere Standardfühler des Typs NTC 10k Ohm (bei 25°C) nicht immer möglich ist. Speziell bei zwei bzw. drei Fühlertypen/Einsatzbereichen ist jedoch grundsätzlich der Einsatz von kostengünstigeren Standardfühlern des Typs NTC 10k Ohm (bei 25°C) möglich, daher werden diese im Folgenden kurz dargestellt.

### Siemens QAD36 / Brötje UAF6C

Der QAD36 ist ein sogenannter "Anlegefühler" (Brötje: "Universalanlegefühler UAF6C") und kann zur nachträglichen Erweiterung als Vor- und Rücklauffühler eingesetzt werden. Der Fühler ist in einem Gehäuse untergebracht, das auf das entspr. Rohr (bspw. Vorlauf) montiert wird und die Temperatur des Mediums indirekt über die Temperatur der Rohrwand misst. Als Messelement kommt ein NTC 10k Ohm (bei 25°C) mit einem Messbereich von -30...125°C, einer Toleranz von +/- 0,5K und einer Zeitkonstante von 6s zum Einsatz.



Der Siemens QAD36-Anlegefühler, bei Brötje "Universalanlegefühler UAF6C" genannt.

Als kostengünstige Alternative zu diesem Fühler kann ein universeller NTC 10k Ohm (bei 25°C) Rohranlegefühler verwendet werden. Diese Standardfühler gibt es in verschiedenen Ausführungen, die alle eine einseitige konkave Wölbung der Metallhülse aufweisen, wodurch das Anbringen des Fühlers an einem Rohr erleichtert und die wärmeübertragende Fläche vergrößert wird. Empfehlenswert ist hier m.E. die Variante mit Messinghülse und einem zusätzlichen, leicht gebogenen Messingstreifen, da diese Ausführung aufgrund der schlanken Bauform gut unter eine Rohrisolierung geschoben, der Biegeradius des Messingstreifens etwas an das Rohr angepasst werden kann und das Messing die Wärme gut leitet.

Es ist empfehlenswert, für die Kontaktfläche eine entspr. Wärmeleitpaste zu verwenden. Zur Sicherung gegen ein Verrutschen ist eine Fixierung mittels Kabelbinder oder Schlauchschelle anzuraten.

### Siemens QAZ36 / Brötje KF ISR



Der QAZ36 in der Ausführung als Sonnenkollektorfühler (Brötje: "ISR Kollektorfühler / KF ISR") ist ein Tauchfühler mit einem ca. 1,5m langen *Silikonkabel*. Als Fühlerelement kommt ein NTC 10k Ohm (bei 25°C) mit einem Messbereich von -30...200°C, einer Toleranz von +/- 0,5K und einer Zeitkonstante von ca. 30s zum Einsatz. Der Durchmesser der Hülse beträgt 6mm, die Länge 40,5mm.



Ein Siemens QAZ36-Kollektorfühler, bei Brötje "ISR Kollektorfühler / KF ISR" genannt.

Aufgrund der höheren Umgebungstemperaturen ist bei der Verwendung eines kostengünstigeren NTC 10k Ohm (bei 25°C) Standardfühlers unbedingt auf den höheren Messbereich und das *Silikonkabel* zu achten - PVC-Kabeltypen sind nicht geeignet!

#### Siemens QAZ36 / Brötje UF6C

Der QAZ36 in der Ausführung als Tauchfühler mit *PVC-Kabel* (Brötje: "Universaltauchfühler UF6C" mit 6m langem PVC-Kabel) kommt bspw. bei Puffer- oder Brauchwasserspeichern zum Einsatz. Als Fühlerelement kommt ein NTC 10k Ohm (bei 25°C) mit einem Messbereich von 0...95°C, einer Toleranz von +/- 0,5K und einer Zeitkonstante von ca. 30s zum Einsatz. Der Durchmesser der Hülse beträgt 6mm, die Länge 40,5mm.



Ein Siemens QAZ36-Tauchfühler, bei Brötje "Universaltauchfühler UF6C" genannt.

Dieser Fühler kann ebenfalls durch einen entspr. kostengünstigeren Standardfühler des Typs NTC 10kOhm (bei 25°C) ersetzt werden.

#### Achtung

Die o.g. 'Tauchfühler' sind *nicht* für das direkte Eintauchen in flüssige Medien geeignet! Der Name bezieht sich auf den Einsatz in sog. 'Tauchhülsen', wie sie bspw. bei Puffer- oder Brauchwasserspeichern vorzufinden sind.

### Achtung

*Bei älteren Reglertypen kommen u.U. andere Fühlertypen zum Einsatz, erkennbar an der Produktbezeichnung (bei den Außentemperaturfühlern lauten diese bspw. "QAC21" und "QAC31") - diese Fühler weisen andere Widerstandswerte und -kennlinien auf und dürfen nicht mit den o.g. Fühlertypen verwechselt werden!*  
*Informationen zu den verwendeten Fühlertypen finden sich in der spezifischen Anlangendokumentation.*



Support me on Ko-fi

[Weiter zu Kapitel 11](#)

[Zurück zum Inhaltsverzeichnis](#)

# 12. Exkurs: Arduino IDE

[Zurück zum Inhaltsverzeichnis](#)

[Zurück zu Kapitel 11](#)

## 12. Exkurs: Arduino IDE

Im Folgenden wird kurz auf die Verwendung der Arduino IDE v1.x sowie die Installation der benötigten Boardbibliotheken für den Arduino Due und den ESP32 eingegangen.

Abschließend wird kurz die Verwendung des Seriellen Monitors (SerMo) der Arduino IDE vorgestellt, mit dem man u.a. das Startverhalten des Arduino Due/ESP32 beobachten und etwaige Fehler leichter identifizieren kann.

### 12.1 Installation

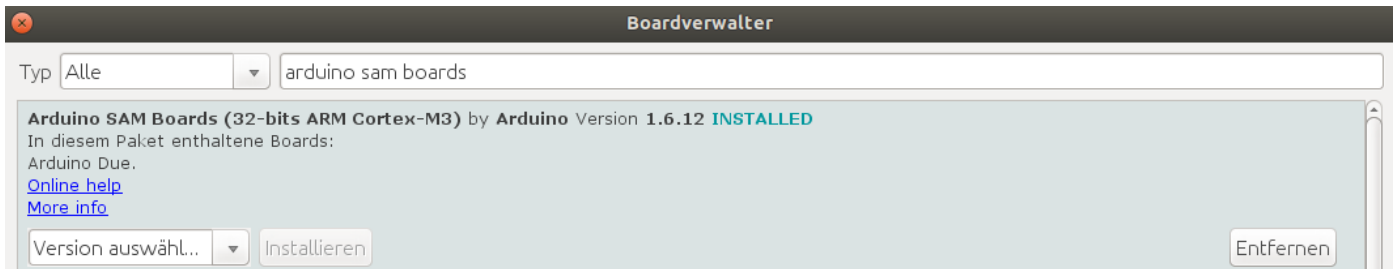
#### Installation der Arduino IDE

Downloade und installiere die aktuelle Version der Arduino IDE von <https://www.arduino.cc/en/Main/Software> für dein Betriebssystem (Windows-, Mac- und Linux-Version verfügbar).

#### 12.1.1 Arduino Due

##### Installation der spezifischen Boardbibliotheken

1. Starte die Arduino IDE und öffne den "Boardverwalter" unter "Werkzeuge/Board".
2. In dem sich nun öffnenden Dialogfenster gib oben in der Suchzeile "Arduino SAM Boards" ein, wo der Due enthalten ist.
3. Klicke auf den Eintrag "Arduino SAM Boards (32-bits ARM Cortex-M3) by Arduino" und dann auf die Schaltfläche "Installieren".



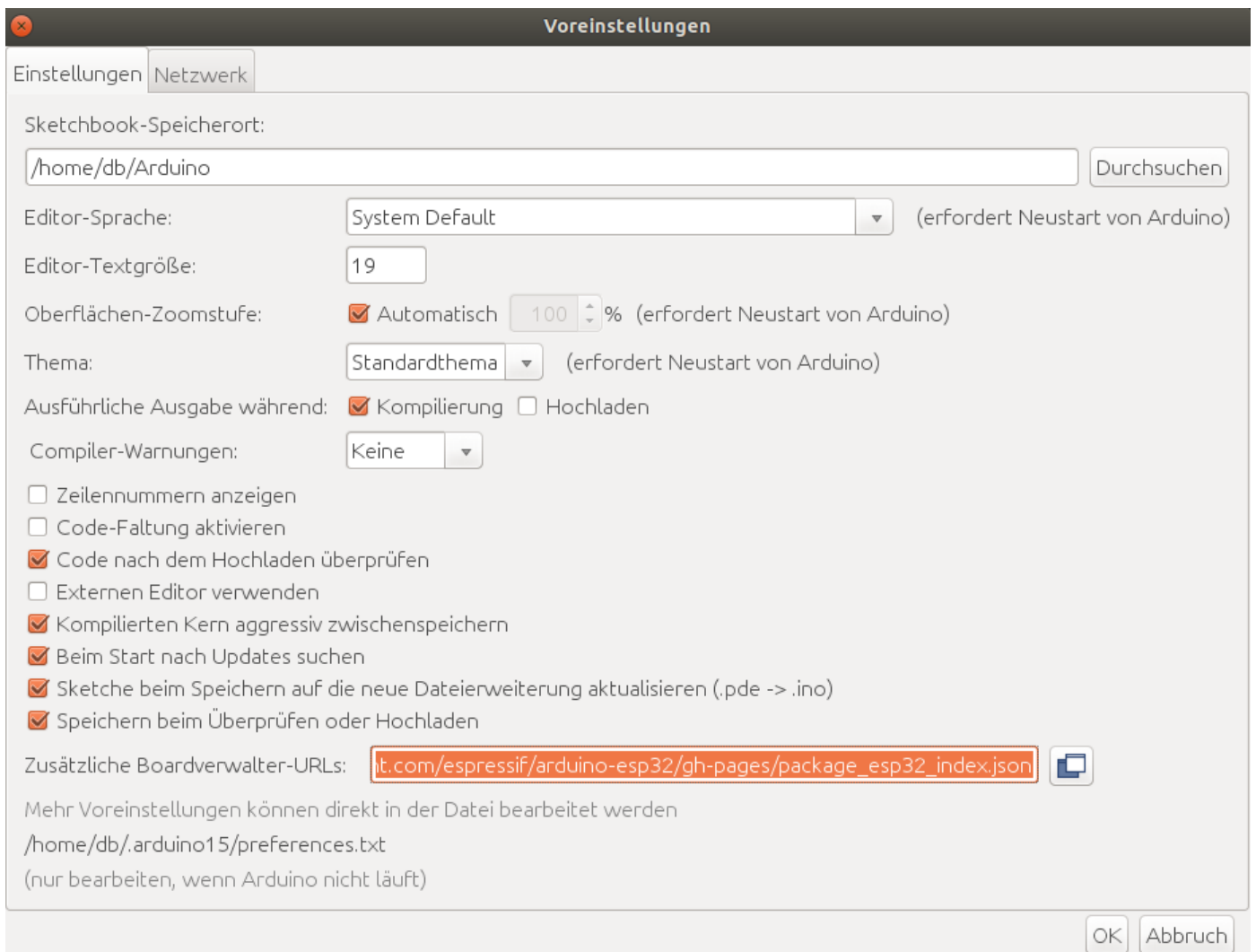
*Das korrekt installierte SAM-Framework (ARM Cortex-M3) für den Arduino Due im Boardverwalter.*

Nun solltest du den Due in der Auflistung bei "Werkzeuge/Board" finden und auswählen können.

#### 12.1.2 ESP32

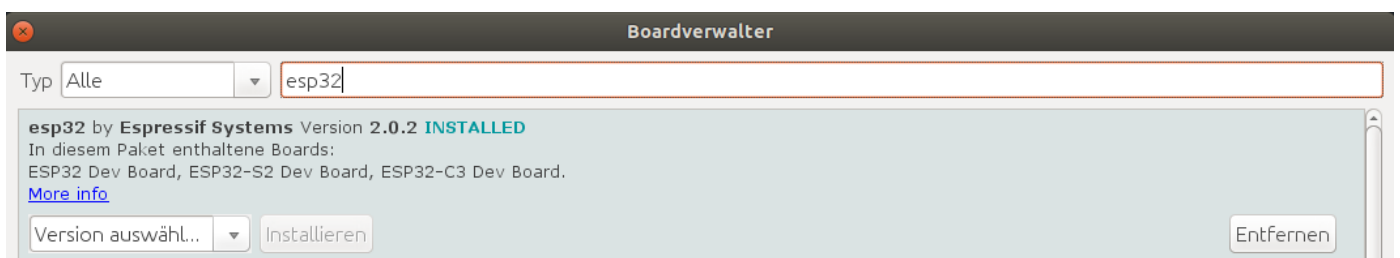
##### Installation der spezifischen Boardbibliotheken

1. Starte die Arduino IDE und klicke auf "Datei/Voreinstellungen" (Shortcut: Strg+Komma).
2. Bei dem sich nun öffnenden Dialogfenster füge unten im Eingabefeld bei "Zusätzliche Boardverwalter-URLs:" folgenden Link ein: [https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package\\_esp32\\_index.json](https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json). Sollte in dem Feld bereits eine oder mehrere URLs stehen, so kann der zusätzliche Eintrag einfach durch ein Komma getrennt zu den bestehenden Einträgen hinzugefügt werden.
3. Klicke dann auf "OK".



Das Dialogfenster "Voreinstellungen" mit dem hinzugefügten Link in der Zeile "Zusätzliche Boardverwalter-URLs".

4. Als nächstes öffne den "Boardverwalter" unter "Werkzeuge/Board".
5. In dem sich nun öffnenden Dialogfenster gib oben in der Suchzeile "ESP32" ein.
6. Suche dann den Eintrag "esp32 by Espressif Systems".
7. Klicke auf den Eintrag, wähle Version 2.0.2 aus (oder höher, falls verfügbar) und klicke dann auf die Schaltfläche "Installieren". **Sollte eine Version kleiner als 2.0.2 installiert sein, führe bitte ein Update auf 2.0.2 (oder höher) aus.**



Das korrekt installierte ESP32-Framework im Boardverwalter.

Nun solltest du das ESP32-Board in der Auflistung bei "Werkzeuge/Board" finden und auswählen können.

## 12.2 Serieller Monitor

### Verwenden des Seriellen Monitors

Der Serieller Monitor (kurz: SerMo) ist ein nützliches Tool, um bspw. das Startverhalten und den Datenverkehr des Mikrocontrollers zu beobachten. So lassen sich bspw. Fehler eingrenzen und finden oder auch unbekannte Telegramme mitschneiden.

Zum Verwenden des SerMo gehe bitte wie folgt vor:

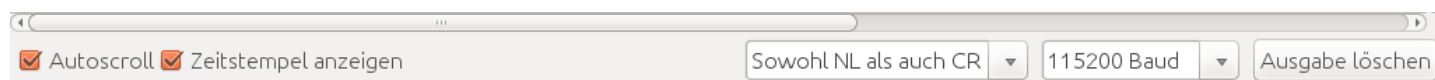
- Verbinde dein BSB-LAN-Setup per USB mit deinem Rechner verbindest.
- Starte die Arduino IDE per Doppelklick auf die Datei *BSB\_LAN.ino*.
- Wähle nun deinen Boardtyp, den Port etc. entsprechend aus.
- Starte nun den "Seriellen Monitor". Dies kann entweder über "Werkzeuge/Serieller Monitor" (Shortcut: Strg+Umschalt+M) oder einfach durch einen Klick auf das Lupensymbol oben rechts in der Symbolleiste der Arduino IDE erfolgen.

In dem Moment, in dem du den Seriellen Monitor startest, wird der angeschlossene Mikrocontroller (Due/ESP32) neu gestartet.

Hast du alles korrekt konfiguriert, kannst du den Startvorgang und das Senden und Empfangen von Telegrammen beobachten (eine exemplarische Ausgabe findest du am Ende dieses Kapitels).

Sollten jedoch nur unleserliche kryptische Zeichen auftauchen, so überprüfe die Einstellung der Übertragungsrate: Diese steht ganz unten rechts und sollte auf 115200 Baud eingestellt sein.

Es bietet sich außerdem an, einen Haken bei "Zeitstempel" zu setzen und im Feld links von der Übertragungsrate "Sowohl NL als auch CR" einzustellen.



Die untere Zeile des SerMo mit den entspr. Einstellungen.

Wenn du nun bspw. via Webinterface URL-Befehle abschickst, so wirst du die entspr. Befehle bzw. Telegramme in der Ausgabe des SerMo sehen können. Regelmäßig eintreffende INF-Telegramme sind Broadcasts, die vom Heizungsregler bzw. von der angeschlossenen Bedieneinheit und ggf. auch von einem zusätzlichen Raumgerät geschickt werden. Von der Bedieneinheit wird etwa alle zehn Sekunden die Kesseltemperatur gesendet, von einem Raumgerät i.d.R. die Raumtemperatur.

Wenn du nun bspw. an der Bedieneinheit einen bestimmten Parameter aufrufst, dann wird dieser samt zugehörigem Wert nicht nur im Display der Bedieneinheit angezeigt, sondern auch im SerMo. Auf diese Weise können bspw. auch unbekannte, neue Parameter eines Heizungsreglers und deren zugehörige Telegramme dekodiert werden (siehe hierzu [Kap. 09](#)).

#### Hinweis:

Wenn du dich mit Fragen oder Problemen an uns (Frederik und mich, Ulf) wendest, so wirst du sehr wahrscheinlich die Aufforderung zum Schicken eines "SerMo-Logs" erhalten. Damit ist gemeint, dass du einen Mitschnitt der Ausgabe des SerMo erstellen sollst.

Dazu entfernst du den Haken unten links bei "Autoscroll" im Fenster des SerMo (damit sich die Ausgabe nicht regelmäßig verschiebt) und markierst dann mit der Maus die gewünschten Zeilen. Mittels copy&paste kannst du die Ausgabe dann in einen Texteditor einfügen und die Datei als txt-file speichern (oder bspw. im Forum in Codetags eingefügt posten).

Nachfolgend ein exemplarischer Mitschnitt einer SerMo-Ausgabe eines erfolgreichen Starts eines BSB-LAN-Setups mit einem Arduino Due und einem angeschlossenen RVS43-Regler samt INF-Meldungen der angeschlossenen Bedieneinheit, die die Kesseltemperatur ca. alle zehn Sekunden als Broadcast sendet:

```
12:25:46.361 -> READY
12:25:46.388 -> Reading EEPROM
12:25:47.084 -> Reading done.
12:25:47.084 -> EEPROM schema v.5 Program schema v.5
12:25:47.084 -> Address EEPROM option 0: 0
12:25:47.084 -> Address EEPROM option 1: 1
12:25:47.084 -> Address EEPROM option 2: 2
12:25:47.084 -> Address EEPROM option 3: 6
12:25:47.084 -> Address EEPROM option 4: 226
12:25:47.084 -> Address EEPROM option 5: 306
12:25:47.084 -> Address EEPROM option 6: 490
12:25:47.118 -> Address EEPROM option 7: 491
12:25:47.118 -> Address EEPROM option 8: 492
12:25:47.118 -> Address EEPROM option 9: 493
12:25:47.118 -> Address EEPROM option 10: 494
12:25:47.118 -> Address EEPROM option 11: 495
12:25:47.118 -> Address EEPROM option 12: 499
12:25:47.118 -> Address EEPROM option 13: 500
12:25:47.118 -> Address EEPROM option 14: 660
12:25:47.118 -> Address EEPROM option 15: 661
12:25:47.118 -> Address EEPROM option 16: 665
12:25:47.118 -> Address EEPROM option 17: 825
12:25:47.118 -> Address EEPROM option 18: 831
12:25:47.118 -> Address EEPROM option 19: 832
12:25:47.118 -> Address EEPROM option 20: 836
12:25:47.151 -> Address EEPROM option 21: 840
```

```

12:25:47.151 -> Address EEPROM option 22: 844
12:25:47.151 -> Address EEPROM option 23: 848
12:25:47.151 -> Address EEPROM option 24: 850
12:25:47.151 -> Address EEPROM option 25: 854
12:25:47.151 -> Address EEPROM option 26: 858
12:25:47.151 -> Address EEPROM option 27: 922
12:25:47.151 -> Address EEPROM option 28: 986
12:25:47.151 -> Address EEPROM option 29: 987
12:25:47.151 -> Address EEPROM option 30: 988
12:25:47.151 -> Address EEPROM option 31: 998
12:25:47.151 -> Address EEPROM option 32: 999
12:25:47.239 -> Address EEPROM option 33: 1159
12:25:47.239 -> Address EEPROM option 34: 1160
12:25:47.239 -> Address EEPROM option 35: 1164
12:25:47.239 -> Address EEPROM option 36: 1165
12:25:47.239 -> Address EEPROM option 37: 1166
12:25:47.239 -> Address EEPROM option 38: 1167
12:25:47.239 -> Address EEPROM option 39: 1171
12:25:47.239 -> Address EEPROM option 40: 1203
12:25:47.239 -> Address EEPROM option 41: 1235
12:25:47.239 -> Address EEPROM option 42: 1267
12:25:47.239 -> Address EEPROM option 43: 1299
12:25:47.239 -> Address EEPROM option 44: 1301
12:25:47.239 -> Address EEPROM option 45: 1302
12:25:47.239 -> Address EEPROM option 46: 1303
12:25:47.239 -> Address EEPROM option 47: 1304
12:25:47.239 -> Address EEPROM option 48: 1336
12:25:47.239 -> Address EEPROM option 49: 1400
12:25:47.239 -> Address EEPROM option 50: 1420
12:25:47.239 -> Address EEPROM option 51: 1440
12:25:47.239 -> Address EEPROM option 52: 1460
12:25:47.239 -> Address EEPROM option 53: 1461
12:25:47.239 -> Address EEPROM option 54: 1462
12:25:47.239 -> Address EEPROM option 55: 1463
12:25:47.239 -> Size of cmdtbl1: 29568
12:25:47.239 -> Size of cmdtbl2: 38616
12:25:47.239 -> Size of cmdtbl3: 26496
12:25:47.239 -> free RAM: 81703
12:25:47.239 -> Init One Wire bus...
12:25:47.239 -> numSensors: 0
12:25:47.239 -> PPS settings:
12:25:47.239 -> Starting SD..failed
12:25:52.197 -> 192.168.178.37
12:25:52.197 -> 255.255.255.0
12:25:52.197 -> 192.168.178.1
12:25:52.197 -> Waiting 3 seconds to give Ethernet shield time to get ready...
12:25:52.197 -> Calculating free space on SD...0 MB free
12:25:55.387 -> Start network services
12:25:55.751 -> LAN->HEIZ QUR 6225 Konfiguration - Gerätefamilie:
12:25:55.751 -> DC C2 00 0B 06 3D 05 00 02 52 88
12:25:55.751 -> HEIZ->LAN ANS 6225 Konfiguration - Gerätefamilie: 96
12:25:55.784 -> DC 80 42 0E 07 05 3D 00 02 00 00 60 5E 3E
12:25:55.784 -> #6225: 96
12:25:55.950 -> LAN->HEIZ QUR 6226 Konfiguration - Gerätevariante:
12:25:55.950 -> DC C2 00 0B 06 3D 05 00 03 42 A9
12:25:55.983 -> HEIZ->LAN ANS 6226 Konfiguration - Gerätevariante: 100
12:25:55.983 -> DC 80 42 0E 07 05 3D 00 03 00 00 64 68 0E
12:25:55.983 -> #6226: 100
12:25:55.983 -> Device family: 96
12:25:55.983 -> Device variant: 100
12:25:55.983 ->
12:25:55.983 -> Setup complete
12:26:00.698 -> DSP1->HEIZ QUR 8310 Diagnose Erzeuger - Kesseltemperatur:
12:26:00.698 -> DC 8A 00 0B 06 3D 0D 05 19 4F 8C
12:26:00.764 -> HEIZ->DSP1 ANS 8310 Diagnose Erzeuger - Kesseltemperatur: 41.0 &deg;C
12:26:00.797 -> DC 80 0A 0E 07 0D 3D 05 19 00 0A 41 08 A5
12:26:10.889 -> DSP1->HEIZ QUR 8310 Diagnose Erzeuger - Kesseltemperatur:
12:26:10.889 -> DC 8A 00 0B 06 3D 0D 05 19 4F 8C
12:26:10.989 -> HEIZ->DSP1 ANS 8310 Diagnose Erzeuger - Kesseltemperatur: 41.0 &deg;C
12:26:10.989 -> DC 80 0A 0E 07 0D 3D 05 19 00 0A 41 08 A5
12:26:21.116 -> DSP1->HEIZ QUR 8310 Diagnose Erzeuger - Kesseltemperatur:
12:26:21.116 -> DC 8A 00 0B 06 3D 0D 05 19 4F 8C
12:26:21.182 -> HEIZ->DSP1 ANS 8310 Diagnose Erzeuger - Kesseltemperatur: 41.0 &deg;C
12:26:21.215 -> DC 80 0A 0E 07 0D 3D 05 19 00 0A 41 08 A5

```

 Support me on Ko-fi

[Weiter zu Kapitel 13](#)

[Zurück zum Inhaltsverzeichnis](#)



# 13. Exkurs: BSB-LAN sicher aus dem Internet heraus erreichen

[Zurück zum Inhaltsverzeichnis](#)

[Zurück zu Kapitel 12](#)

## 13. Exkurs: BSB-LAN sicher aus dem Internet heraus erreichen

In diesem Abschnitt werden die grundsätzlichen Möglichkeiten für eine sichere Erreichbarkeit von BSB-LAN aus dem Internet beschrieben. Aufgrund der Vielzahl von erhältlichen Routern können hier nur die wichtigsten Schritte beschrieben werden, für weitere Details ist das Handbuch des jeweiligen Routers hinzuzuziehen. Wir können für die Einrichtung dieser Schritte auch keinen Support bieten, bitte dafür in passenden Internet-Foren um Rat fragen.

### Grundvoraussetzung: (Sub-)Domain mit dynamischem DNS einrichten

Um den externen Zugriff zu ermöglichen, benötigt man eine eigene (Sub-)Domain, die man über einen dynamischen DNS-Dienst vom Internet aus erreichen kann. Manche Router- oder NAS-Anbieter wie AVM oder Synology bieten solch einen Service direkt für ihre Kunden an, ansonsten muss man eine eigene Domain (z.B. `mein-zuhause.de`) besitzen, bei der man sich dann eine Subdomain einrichtet (hier im Beispiel `bsb-lan.mein-zuhause.de`), die dann zusammen mit dem dynamischen DNS-Anbieter entsprechend konfiguriert werden muss.

### Variante 1: Virtuelles Privates Netzwerk (VPN)

Viele Router stellen von Haus aus einen Server für ein Virtuelles Privates Netzwerk (VPN) bereit. Dies ist die sicherste Variante, weil auf diese Weise der anderweitige Zugang zum Heimnetz generell gesperrt ist. Ist so ein VPN Server z.B. auf dem Router eingerichtet und aktiviert, kann man mit einem ebenfalls VPN-fähigen Gerät auf BSB-LAN so zugreifen, wie man es auch sonst tun würde, also ganz normal über die heimatische IP-Adresse.

Der Nachteil liegt allerdings darin, dass es ohne ein VPN-fähiges Endgerät nicht möglich ist, auf BSB-LAN zuzugreifen. Ebenso kann der Internetzugang, mit dem man unterwegs aus das Internet nutzt, so konfiguriert sein, dass VPN nicht möglich ist. In diesen Fällen gibt es dann keine Möglichkeit auf die heimischen Ressourcen zuzugreifen.

### Variante 2: Reverse Proxy

Ein Reverse Proxy bietet u.a. die Möglichkeit, mehrere Geräte im Heimatnetz über ein einziges, von außen sichtbares Gerät, auf dem ein Reverse Proxy Server läuft, zu erreichen. Hierfür sind die folgenden Schritte nötig:

#### 1. Portweiterleitung einrichten

Für das Gerät, auf dem der Reverse Proxy läuft, muss im lokalen Netzwerk eine Portfreigabe eingerichtet werden. Damit dieser Zugriff über SSL/TLS abgesichert werden kann, ist hierfür Port 443 zu verwenden. Hierbei ist aufzupassen, dass dann ggf. nicht mehr über Port 443 auf den eigentlichen Router zugegriffen werden kann. Bei einigen Routern lässt sich aber der SSL-Port verändern, so dass das kein grundsätzliches Problem sein muss. Für die Nutzung von SSL/TLS / Port 443 muss natürlich auf dem Gerät dann auch ein entsprechendes (ggf. selbstsigniertes) Zertifikat installiert sein. Viele Router- oder NAS-Hersteller bieten dafür aber schon von Werk aus die Einrichtung von kostenlosen Let's Encrypt-Zertifikaten an.

#### 2. Reverse Proxy installieren und einrichten

Das Gerät, auf dem der Reverse Proxy läuft, kann irgendein Rechner sein, der dauerhaft erreichbar ist, z.B. ein File-Server/NAS. Auf diesem wird der ReverseProxy-Server installiert und eingerichtet. Nutzt man hierfür eine Synology NAS, so ist hier ab DSM 7 bereits solch eine Funktion mit eingebaut (siehe Systemsteuerung / Anmeldeportal / Erweitert).

Man konfiguriert den Reverse Proxy nun so, dass er die Anfragen für die gewählte (Sub-)Domain über **HTTPS(!)** auf Port 443 annimmt und diese dann über **HTTP(!)** auf Port 80 des BSB-LAN Adapters weiterleitet. Der Weg zurück erfolgt dann genau umgekehrt: Von BSB-LAN über ungesichertes HTTP zum Reverse Proxy und von dort aus über HTTPS wieder raus ins Internet.

Nun ist BSB-LAN über den HTTPS-Aufruf der (Sub-)Domain direkt erreichbar. Es empfiehlt sich nun auf jeden Fall die HTTP-Authentifizierung in BSB-LAN zu aktivieren, da ansonsten jeder Zugriff auf BSB-LAN hätte.



[Weiter zu Kapitel 14](#)

[Zurück zum Inhaltsverzeichnis](#)

# 14. Etwaige Fehlermeldungen und deren mögliche Ursachen

[Zurück zum Inhaltsverzeichnis](#)

[Zurück zu Kapitel 13](#)

## 14. Etwaige Fehlermeldungen und deren mögliche Ursachen

### 14.1 Fehlermeldung "unknown type \<xxxxxxx>"

---

Dieser Fehler sagt aus, dass für diesen Parameter keine Umrechnungsanweisung vorliegt, um die Rohdaten in eine entsprechende Einheit (Zeit, Temperatur, Prozent, Druck etc.) umzuwandeln.

Um den Fehler zu beheben, sollte das jeweilige Telegramm / die Command ID des betreffenden Parameters sowie der zugehörige Wert ausgelesen und gemeldet werden. Sollten mehrere Einstellungsoptionen für einen Parameter verfügbar sein, muss zusätzlich jede Option ausgelesen werden, damit eine eindeutige Zuordnung stattfinden kann. Siehe hierzu auch das [Kap. 9](#).

### 14.2 Fehlermeldung "error 7 (parameter not supported)"

---

Die zugehörige Command ID wird nicht erkannt oder der entsprechende Parameter wird vom Regler nicht unterstützt (bspw. spezifische Parameter, die eine Gasheizung betreffen und bei einer Ölheizung dementsprechend nicht verfügbar sind).

Fehlermeldungen dieses Typs werden seit v0.41 der Übersichtlichkeit halber per default ausgeblendet (die entsprechenden Parameter bspw. bei einer Komplettabfrage aber dennoch abgefragt). Möchtest du sie dennoch angezeigt bekommen, so ist das entsprechende Definiment `#define HIDE_UNKNOWN` in der Datei `BSB_lan_config.h` auszukommentieren (`//#define HIDE_UNKNOWN`).

Zur Überprüfung, ob die CommandID vom Regler prinzipiell unterstützt wird, jedoch für diese Gerätefamilie nicht freigegeben ist, führe bitte den URL-Befehl /Q aus (s. hierzu auch [Kap. 8.2.5](#)). Sollten bei dieser Abfrage 'error 7'-Meldungen angezeigt werden, melde sie bitte unter Angabe des kompletten Outputs von /Q.

Sollte ein Parameter an der heizungsseitigen Bedieneinheit jedoch definitiv verfügbar sein und dennoch bei der /Q-Abfrage nicht als 'error 7'-Parameter aufgelistet werden, so sollte der entspr. Parameter gemäß der Beschreibung in [Kap. 9](#) decodiert und gemeldet werden.

### 14.3 Fehlermeldung "query failed"

---

Diese Meldung erscheint, wenn auf die Anfrage des Adapters keine (sinnvolle) Antwort des Reglers kommt.

Mögliche Ursachen sind meist hardwareseitig zu suchen (bspw. fehlerhafte RX- und/oder TX-Verbindung, falsch verbaute Komponenten oder auch ein timeout aufgrund eines ausgeschalteten oder nicht angeschlossenen Reglers).

### 14.4 Fehlermeldung "FEHLER: Setzen fehlgeschlagen! - Parameter ist nur lesbar"

---

Diese Meldung erscheint bei dem Versuch, Werte zu schreiben bzw. zu übermitteln (bspw. die Raumtemperatur) oder Parameter zu verändern, während der Zugriff des Adapters nur auf Lesen beschränkt ist.

Sollte es sich nicht um einen Parameter handeln, der per se nur lesbar ist, muss BSB-LAN Schreibzugriff gewährt werden.

### 14.5 Fehlermeldung "decoding error"

---

Die Fehlermeldung "decoding error" bedeutet, dass der Parameter und die Command ID bekannt sind bzw. passen, aber dass das Datenpaket nicht der bisher bekannten Dekodierung entspricht. Das kann eine andere Länge oder eine andere Einheit als Grund haben.

Um das für die entsprechende Heizung zu aktualisieren, wird das zu dem Fehler ausgegebene Datenpaket und der exakt zu diesem Moment angezeigte Wert an der Therme und die Einheit benötigt. Siehe hierzu auch das [Kap. 9](#).



Support me on Ko-fi

[Weiter zu Kapitel 15](#)

[Zurück zum Inhaltsverzeichnis](#)

# 15. Etwaige Probleme und deren mögliche Ursachen

[Zurück zum Inhaltsverzeichnis](#)

[Zurück zu Kapitel 14](#)

## 15. Etwaige Probleme und deren mögliche Ursachen

### 15.1 Arduino IDE stoppt beim Kompilieren

---

Es gibt viele Gründe, dass die Arduino IDE beim Kompilervorgang mit einer Fehlermeldung abbricht, bspw. weil ein falscher Boardtyp, Anschluss oder eine falsche Geschwindigkeit ausgewählt wurde. Es gibt jedoch drei Typen von Fehlermeldungen, die beim Kompilieren für ESP32 basierte Boards auftreten können, die hier kurz erwähnt werden sollten:

- Die Fehlermeldung erwähnt etwas bzgl. "WiFiSPI"?  
→ Wenn auf ESP32, entferne den `WiFiSPI`-Ordner aus dem Ordner `src` - s. Schritt 5 in [Kap. 2.1.2](#).
- Die Fehlermeldung erwähnt etwas bzgl. "ArduinoMDNS"?  
→ Wenn auf ESP32, entferne den `ArduinoMDNS`-Ordner aus dem Ordner `src` - s. Schritt 5 in [Kap. 2.1.2](#).
- Die Fehlermeldung erwähnt etwas bzgl. "EEPROMClass"?  
→ Stelle sicher, dass du das korrekte ESP32 framework installiert hast (1.0.6 ist zu alt) - s. [Kap. 12.1.2](#).

### 15.2 Die rote LED des Adapters leuchtet nicht

---

- Regler ist ausgeschaltet
- Adapter ist nicht mit dem Regler via BSB oder LPB verbunden
- Adapter ist falsch mit dem Regler verbunden (CL+/CL- bzw. DB/MB vertauscht)
- Evtl. Hardwarefehler des Adapters (bspw. defektes Bauteil, Fehler im Aufbau)
- Evtl. Wackelkontakt beim Bus-Anschluss (Rx/Tx oder CL+/CL-)

### 15.3 Die rote LED leuchtet, aber es ist keine Abfrage möglich

---

- Evtl. Adapter falsch angeschlossen (an G+ statt an CL+)
- Evtl. Wackelkontakt beim Busanschluss (Rx/Tx oder CL+/CL-)
- Evtl. falsche Pinbelegung (Rx/Tx)
- Evtl. Transistoren Q1/Q2 vertauscht
- Evtl. kalte Lötstellen
- Siehe Punkt „[Keine Parameterabfrage möglich](#)“

### 15.4 Zugriff auf das Webinterface nicht möglich

---

- Adapter hat keine, keine ausreichende oder eine unzuverlässige Stromversorgung
- Adapter bzw. LAN-Shield ist nicht mit dem LAN verbunden
- IP- und/oder MAC-Adresse des Adapters ist nicht korrekt
- Sicherheitsfunktionen `Passkey`, `TRUSTED_IP` und/oder `USER_PASS_B64` aktiviert/deaktiviert → URL nicht angepasst, Zugriff von falscher IP etc.

- Router- und/oder Firewall-Einstellungen überprüfen
- Zugriff nach Stromausfall und/oder Neustart nicht möglich → Reset-Knopf des Mikrocontrollers drücken
- Wird eine microSD-Karte zum Loggen verwendet? → FAT32-formatieren, URL-Befehl `/D0` ausführen, evtl. andere/kleinere Karte testen → s. Kap. 6.1
- (Adapter,) LAN-Shield und/oder Arduino/Mikrocontroller fehlerhaft (→ vereinzelt kam es zu diffusen Problemen bei der Verwendung von günstigen Clones; im Zweifelsfall ist ein Test mit einem anderen LAN-Shield zu empfehlen)

## 15.5 Keine Verbindung zum WLAN möglich

Bitte überprüfe, ob das Definiment `#define WIFI` aktiviert ist, also dass die führenden Schrägstriche entfernt wurden. Ein Hinweis darauf, dass das Definiment nicht aktiviert wurde, sind u.a. diese Fehlermeldungen im Seriellen Monitor direkt nach dem Neustart:

```
E (1229) esp.emac: emac_esp32_init(349): reset timeout
E (1229) esp_eth: esp_eth_driver_install(214): init mac failed
```

BSB-LAN kann sich darüber hinaus nicht mit versteckten WLAN-Netzwerken verbinden. Dies geht nur, wenn man die BSSID des WLAN-Netzwerks in der Datei `BSB_LAN_config.h` fest in die Variable `bssid` einträgt.

## 15.6 Keine Parameterabfrage möglich

- Siehe Punkt „Die rote LED des Adapters leuchtet nicht“
- Siehe Punkt „Die rote LED leuchtet, aber es ist keine Abfrage möglich“
- Siehe Punkt „Zugriff auf das Webinterface nicht möglich“
- Rx- und/oder Tx-Belegung nicht korrekt, Pinbelegung und/oder Adapteranschluss stimmt nicht mit der Angabe in der Datei `BSB_LAN_config.h` überein
- Falscher Bus-Typ (BSB/LPB)

## 15.7 Regler wird nicht korrekt erkannt

- Siehe Punkt „Die rote LED leuchtet, aber es ist keine Abfrage möglich“
- Siehe Punkt „Keine Parameterabfrage möglich“
- Regler ist ausgeschaltet
- Regler wurde erst nach dem Mikrocontroller angeschaltet (automatische Reglererkennung funktioniert dann nicht)
- Regler ist nicht oder falsch mit dem Adapter verbunden
- Gerätefamilie und -variante ( `http://<IP-Adresse>/6225/6226` ) des Reglers unbekannt

## 15.8 HK1 kann nicht bedient werden

- Adapter ist evtl. als RGT2 konfiguriert

## 15.9 Es kann keine Raumtemperatur an einen HK1 gesendet werden

- Adapter ist evtl. als RGT2 konfiguriert
- Zugriff des Adapters ist auf Lesen beschränkt → Schreibzugriff muss gewährt werden (Webconfig `/c`: "Schreibzugriff" auf "Standard" oder "Komplett" stellen)

## 15.10 HK2 kann nicht bedient werden

---

- Adapter ist evtl. als RGT1 konfiguriert

## 15.11 Es kann keine Raumtemperatur an einen HK2 gesendet werden

---

- Adapter ist evtl. als RGT1 konfiguriert
- Zugriff des Adapters ist auf Lesen beschränkt → Schreibzugriff muss gewährt werden (Webconfig : "Schreibzugriff" auf "Standard" oder "Komplett" stellen)

## 15.12 Einstellungen des Reglers können nicht via Adapter verändert werden

---

- Zugriff des Adapters ist auf Lesen beschränkt → Schreibzugriff muss gewährt werden (Webconfig : "Schreibzugriff" auf "Standard" oder "Komplett" stellen)

## 15.13 Der Adapter reagiert manchmal nicht auf Abfragen oder SET-Befehle

---

- Der Mikrocontroller ist nicht multitaskingfähig - warte, bis eine Abfrage abgeschlossen ist (insbesondere umfangreichere Abfragen wie bspw. ganze Kategorien oder auch die Darstellung des Logfiles dauern u.U. recht lange)

## 15.14 Bei der Abfrage der Logdatei passiert ,nichts'

---

- Es ist keine microSD-Karte eingelegt
- Das Loggen auf microSD-Karte war oder ist deaktiviert
- Die Logdatei ist sehr groß, jegliche Darstellung dauert entsprechend länger
- Die grafische Darstellung ( <http://<IP-Adresse>/DG> ) der Logdatei kann aufgrund von JavaScript-Blockern nicht erfolgen

## 15.15 Es werden keine 24h-Durchschnittswerte angezeigt

---

- Das entsprechende Definiment ist nicht aktiviert
- Es sind keine zu berechnenden Parameter angegeben

## 15.16 Bei der Abfrage der Daten von DS18B20-/DHT22-Sensoren passiert ,nichts'

---

- Es sind keine Sensoren angeschlossen
- Die entsprechenden Definements sind nicht aktiviert
- Die Pinbelegung ist nicht korrekt eingestellt
- Die Sensoren sind fehlerhaft installiert oder defekt

## 15.17 Die DS18B20-Sensoren zeigen falsche Werte an

---

- Die Stromversorgung und Installation prüfen (Größe des PullUp-Widerstands prüfen, Kondensatoren verbauen, Verkabelung prüfen, richtige Topologie verwenden etc.)

## 15.18 Der ,Serielle Monitor' der Arduino IDE liefert keine Daten

---



- Der Adapter ist nicht zusätzlich via USB angeschlossen
- Falscher Anschluss (COM-Port) oder falsches Board in der Arduino IDE ausgewählt
- Falsche Baudrate eingestellt → auf 115200 Baud einstellen
- Adapter nicht am Regler angeschlossen, Regler ist ausgeschaltet → siehe o.g. Punkte



Support me on Ko-fi

[Weiter zu Kapitel 16](#)

[Zurück zum Inhaltsverzeichnis](#)

## 16. FAQ

[Zurück zum Inhaltsverzeichnis](#)

[Zurück zu Kapitel 15](#)

## 16. FAQ

### 16.1 Kann ich Adapter & Software mit einem Raspberry Pi nutzen?

---

Ja und nein.

Der Adapter kann mit einem Raspberry Pi verwendet werden, wenn andere Pinheader genutzt werden (weibliche statt männliche). Siehe hierzu bitte die folgenden Kapitel: [Kap. 1.4](#) sowie [Anhang A2.2](#).

**Die BSB-LAN-Software kann NICHT mit einem RPi verwendet werden, sie ist ausschließlich auf dem hier vorgestellten Mikrocontrollersystem lauffähig!**

Zur Nutzung des Adapters mit einem RPi muss eine vollkommen andere Software genutzt werden. Weitere Informationen diesbezüglich sind in [1.4](#) zu finden.

### 16.2 Kann ich einen Adapter gleichzeitig an zwei Regler anschließen?

---

Nein, das geht leider nicht.

Derzeit benötigt man für jeden Regler einen Adapter bzw. ein komplettes Hardware-Setup (Mikrocontroller + Adapter), um die jeweiligen reglerspezifischen Parameter via BSB abrufen zu können.

Sollten jedoch mehrere Regler vorhanden und bereits miteinander via LPB verbunden sein, beachte bitte die folgende FAQ.

### 16.3 Kann ich einen Adapter via LPB anschließen und mehrere Regler abfragen?

---

Ja, wenn die vorhandenen Regler bereits korrekt via LPB miteinander verbunden und entsprechend konfiguriert sind (korrekte LPB-Adressvergabe).

Die Möglichkeit, Abfragen fallweise an unterschiedliche Regler zu senden, ist mittlerweile gegeben, jedoch ist diese Funktion noch nicht ausgiebig getestet. Siehe hierzu den entsprechenden Punkt in Kapitel [5.1](#).

### 16.4 Ist ein multifunktionaler Eingang des Reglers direkt via Adapter schaltbar?

---

Nein!

Die multifunktionalen Eingänge der Regler (bspw. H1, H2, H3 etc.) sind nicht direkt an den Adapter anzuschließen!

Soll bspw. eine Betriebsartumschaltung oder Erzeugersperre mittels H1 als Arbeitskontakt realisiert werden, so muss der jeweilige Eingang den Herstellerangaben entsprechend parametrisiert und belegt werden. Eine Steuerung dieser Art muss mittels eines anzuschließenden Relais erfolgen, dessen reglerseitiger Ausgang unbedingt potentialfrei sein muss, d.h. es darf keinerlei Fremdspannung anliegen! Das Relais hat in dem Fall lediglich die Aufgabe, den Kontakt zu schließen (oder zu öffnen).

Das Relais wiederum kann jedoch unter bestimmten Umständen vom Mikrocontroller gesteuert werden (bspw. mittels eines Relaisboards). Siehe hierzu auch [Kap. 7.2](#).

Entsprechende Relais findest du im Internet, bei Unsicherheiten solltest du deinen Elektriker und/oder Heizungsinstallateur zu Rate ziehen. Eine falsche Belegung und/oder Parametrierung kann den Regler u.U. zerstören!

### 16.5 Ist zusätzlich ein Relaisboard am Mikrocontroller anschließ- und steuerbar?

---

Ja. Siehe diesbezüglich den entsprechenden Punkt in Kap. 5.1 sowie Kap. 7.2.

## 16.6 Kann ich bspw. den Zustand eines angeschlossenen Koppelrelais abfragen?

---

Ja. Siehe diesbezüglich den entsprechenden Punkt in Kap. 5.1.

## 16.7 Kann ich behilflich sein, um bisher nicht unterstützte Parameter hinzuzufügen?

---

Ja! Wenn dein Heizungssystem über Parameter verfügt, die von der Software bisher nicht unterstützt werden, würden wir uns sehr freuen, wenn du uns unterstützt! Genauere Informationen zur Vorgehensweise sind in Kap. 9 zu finden.

## 16.8 Warum erscheinen bei einer Komplettabfrage einige Parameter doppelt?

---

Wenn du eine Komplettabfrage aller Parameter via URL-Befehl machst ( `http://<IP-Adresse>/0-10000` ) kann es sein, dass sich einige Parameter bzw. Programmnummern in der Auflistung wiederholen. Dies kommt daher, dass es es zwar unterschiedliche Parameter sind, diese aber die gleiche Command ID haben. Dies stellt nur einen 'optischen Mangel' dar, der die Funktionalität nicht negativ beeinflusst.

## 16.9 Warum werden manchmal bestimmte Parameter nicht angezeigt?

---

Wenn der Regler nach erfolgreichem Adapteranschluss angeschaltet wird und der Arduino zu diesem Zeitpunkt bereits lief, funktioniert die automatische Reglererkennung nicht. Der Arduino muss dann lediglich resettet bzw. aus- und wieder angeschaltet werden.

Sollten dann bestimmte Parameter noch immer nicht erscheinen, so sollte bitte einmal /Q ausgeführt und die Webausgabe gemeldet werden.

## 16.10 Warum ist kein Zugriff auf angeschlossene Sensoren möglich?

---

Wenn du DHT22, BME280 und/oder DS18B20-Sensoren korrekt am angeschlossen hast, sie allerdings keine Werte liefern, hast du vermutlich die betreffenden Einträge in der Datei `BSB_LAN_config.h` oder in der Konfiguration via Webinterface nicht entsprechend angepasst.

Siehe hierzu auch die Kapitel 2.2 und 7.1.

## 16.11 Ich nutze ein W5500-LAN-Shield, was muss ich tun?

---

Darauf achten, dass die aktuelle Version der Ethernet Bibliothek (mindestens Version 2.0) in der Arduino IDE vorhanden ist. Dies ist i.d.R. der Fall. Sollten jedoch Probleme mit dem LAN-Shield auftauchen, so solltest du sicherheitshalber die Version überprüfen.

## 16.12 Können Stati oder Werte als Push-Mitteilungen abgesetzt werden?

---

Nein, nicht ohne weitere Software wie z.B. FHEM. Dafür müsste ansonsten der Regler des Wärmeerzeugers ständig abgefragt werden, was den Bus (und die Erreichbarkeit des Arduino) stark belasten würde. Die sinnvollere Variante wäre, bestimmte Werte z.B. alle 60 Sekunden abzurufen und dann anhand bestimmter Kriterien weitere Aktionen auszulösen.

Bei FHEM wäre das mit DOIF oder NOTIFY möglich.

## 16.13 Kann bspw. FHEM auf bestimmte Broadcasts ,lauschen'?

---

FHEM kann zwar lauschen, aber BSB-LAN kann bisher keine eigenständigen Nachrichten absetzen. Dazu müsste ein Hintergrundprozess die auflaufenden Broadcast-Meldungen anhand konfigurierbarer Schwellenwerte auswerten und über einen HTTP-Client-Aufruf an eine definierbare Zieladresse absetzen. Ob dies mit dem begrenzten Speicherplatz des Arduino noch umsetzbar ist, wäre fraglich. Wer sich aber daran probieren möchte, ist herzlich eingeladen, dies zu tun!

## 16.14 Warum kommt es manchmal zu timeout-Problemen bei FHEM?

---

Das könnte an der Dauer des Sende-/Empfangsvorgangs liegen. Man sollte den timeout-Wert in FHEM so bemessen, dass für jeden Parameter pro Setzbefehl zwei und pro Abfragebefehl eine Sekunde angesetzt werden.

Sind mehrere einzelne (BSB-LAN-spezifische) HTTPMOD-Abfragen definiert und werden diese zum gleichen Zeitpunkt ausgeführt, kann es außerdem vorkommen, dass es zu Kollisionen kommt und sie sich somit gegenseitig ‚behindern‘, da der Bus bereits von einer Abfrage belegt ist. Als Abhilfe können hier entweder unterschiedliche Abfrageintervalle gewählt oder alle Abfragen in eine HTTPMOD-Abfrage gelegt werden.

FHEM-Forumsmitglied „frank“ hat den [Tipp](#) gegeben, bei der Einbindung in FHEM ‚attr alignTime‘ zu nutzen, um Kollisionen bei den Abfragen zu verhindern.

## 16.15 Gibt es ein Modul für FHEM?

---

Jein. FHEM-Forumsmitglied „justme1968“ hatte in der Vergangenheit angefangen, ein Modul zu entwickeln:

<https://forum.fhem.de/index.php/topic,84381.0.html>

Die Entwicklung ist jedoch noch nicht abgeschlossen, so dass ein zuverlässiger und problemloser Einsatz mit der aktuellen BSB-LAN-Version nicht garantiert werden kann.

## 16.16 Warum werden unter /B bei Stufe 2 keine Werte angezeigt?

---

Wenn du einen Gasbrenner hast, so wird dieser höchstwahrscheinlich modulieren und generell nicht über ein zweistufiges Brennersystem verfügen; zweistufige Brenner kommen meist nur bei Ölbrennern zum Einsatz. Die Unterscheidung der Brennerstufen wird mittels spezifischer Broadcasts vorgenommen, die jedoch nicht jeder Regler sendet. In dem Fall werden die Brennerstarts und -laufzeiten kumuliert unter Stufe 1 dargestellt. Bitte beachte diesbezüglich auch den Hinweis unter „/B“ in Kap. 5.1.

## 16.17 Ich habe den Eindruck, die angezeigten Werte bei /B sind nicht korrekt.

---

Das kann durchaus sein. Die jeweiligen Starts und Laufzeiten werden anhand von Broadcasts ermittelt, die automatisch vom Regler gesendet werden. Manchmal kann es vorkommen, dass einzelne BCs nicht ankommen, bspw. wenn zeitgleich eine Abfrage gestartet wird oder der Arduino das Logfile lädt.

## 16.18 Was ist der genaue Unterschied zwischen /M1 und /V1?

---

Mit dem URL-Befehl /M1 aktivierst du den Monitor-Modus, mit /V1 den Verbotitäts-Modus.

Mit aktivierter Monitor-Funktion (/M1) werden alle Daten, die über den Bus gehen und nicht von BSB-LAN aus initiiert wurden, „roh“ auf dem seriellen Monitor ausgegeben.

Dies kann sinnvoll sein, um Fehlfunktionen in der Datenübertragung ausfindig zu machen, da ansonsten nur Meldungen von BSB-LAN verarbeitet werden, die von ihrem Aufbau her korrekt sind. Das schließt auch die Verarbeitung von Broadcast-Nachrichten ein, d.h. mit aktivierter Monitor-Funktion findet keine Auswertung dieser Nachrichten statt.

Die Monitor-Funktion erlaubt es z.B. bei Fehlermeldungen genauer zu sehen, ob eine Nachricht schlichtweg nicht auf dem Bus angekommen ist oder ob BSB-LAN sie wegen fehlerhafter Übertragung verworfen hat. Die volle Kontrolle hätte man mit einem zweiten BSB-LAN-Adapter, der auf dem Bus lauscht und dann alle ein- und ausgehenden Nachrichten protokolliert.

Mit (seit v0.41 per default) aktiviertem Verbotitäts-Modus (/V1) werden zu jedem von BSB-LAN initiierten Aufruf und der entsprechenden Antwort neben dem Klartext auch die entsprechenden Rohdaten auf dem seriellen Monitor ausgegeben, wenn die Nachricht von ihrem Aufbau her korrekt sind und fehlerfrei übertragen wurden.

Eine Auswertung von (fehlerfreien) Broadcasts findet hier weiterhin statt. Es werden hier beim Senden aber nur die Daten ausgegeben, die BSB-LAN vorbereitet hat. Dies muss nicht bedeuten, dass diese Daten - z.B. bei Hardwarefehlern - auch auf dem Bus ankommen. Umgekehrt werden beim Auswerten der Rückmeldung auf einen Befehl zwar die Daten ausgegeben, die auf dem Bus zurück gekommen sind, aber nur dann, wenn die Nachricht auch korrekt aufgebaut war.

Eine Kombination aus beiden Parametern ist möglich und führt dazu, dass im Monitor-Modus auch bei von BSB-LAN initiierten Nachrichten die Rohdaten ausgegeben werden - mit den bereits erwähnten Einschränkungen des Verbotitäts-Modus bezüglich des Verwerfens von nicht korrekt aufgebauten Nachrichten.

## 16.19 Kann ich eigenen Code in BSB-LAN einbinden?

---

Ja, dafür gibt es die Datei `BSB_LAN_custom.h`. Hier können eigene Programmteile geschrieben werden, die bei jedem Schleifendurchlauf (loop) aufgerufen werden. Damit kann man z.B. unabhängig von externen Home-Automations-Systemen Sensoren auswerten und/oder Relais schalten. Die Beispieldatei wertet z.B. zwei DHT22-Feuchtigkeits-/Temperatursensoren aus und schaltet beim Unter- bzw. Überschreiten ein Relais, das an einem digitalen Ausgang angeschlossen ist.

Siehe hierzu auch Kap. 6.8.

## 16.20 Kann ich MAX!-Thermostate einbinden?

---

Ja, das ist möglich. Dazu musst du das entsprechende Definiment in der Datei `BSB_LAN_config.h` aktivieren und anpassen. Siehe hierzu auch das Kap. sowie 7.3.

Mittels entsprechender Modifikationen in der Datei `BSB_LAN_custom.h` (s. Kap. 6.8) können weitere Funktionen realisiert werden, mit der derzeitigen Programmierung ist eine eigenständige Raum-Ist-Wert-Übermittlung (ohne FHEM) möglich.

Siehe auch die jeweiligen Punkte in den Kapiteln 2.2 und 5.1.

## 16.21 Warum ist der Adapter nach einem Stromausfall nicht mehr erreichbar?

---

Dieses Verhalten wurde des Öfteren bei den günstigen LAN-Shield-Clones beobachtet, mit einem originalen Arduino-LAN-Shield scheint dieses Problem nicht aufzutreten. Eine konkrete Erklärung hierfür gibt es bisher nicht.

Ein weiterer Grund kann sein, dass der Mikrocontroller nach einem Stromausfall nicht korrekt gestartet ist.

Abhilfe: Nach Drücken des Reset-Knopfes am Mikrocontroller ist der Adapter wieder wie gewohnt erreichbar. Generelle Abhilfe könnte eine kleine USV für den Mikrocontroller schaffen, so dass der Mikrocontroller nicht stromlos wird. Andere Lösungen sind bisher nicht bekannt.

## 16.22 Warum ist der Adapter (ohne Stromausfall) manchmal nicht mehr erreichbar?

---

Dieses Problem ist bisher nur vereinzelt aufgetreten, eine eindeutige Lösung oder Erklärung für dieses Verhalten gibt es bisher noch nicht. Laut Mitschnitt des Seriellen Monitors lief der BSB-LAN-Sketch ohne Probleme weiter, lediglich das LAN-Shield war nicht mehr erreichbar. Abhilfe schaffte nur ein Reset. Sollte dieses Verhalten auftreten, ist das Testen eines weiteren LAN-Shields zu empfehlen, da Hardwareprobleme des betroffenen LAN-Shields nicht auszuschließen sind.

## 16.23 Warum kommen beim Senden manchmal ‚query failed‘-Meldungen?

---

Wenn Befehle, die in der Regel problemlos gesendet werden können, plötzlich ‚query failed‘-Fehlermeldungen auslösen, könnte dies in der eingesetzten Hardware begründet sein. Es scheint, als wenn einige günstige Mikrocontroller zeitweise unzuverlässig arbeiten und diffuse Probleme verursachen. Abhilfe könnte ein Austausch des Mikrocontrollers schaffen, der Einsatz eines originalen Mikrocontrollers ist selbstverständlich eine weitere Option.

Ein Nutzer berichtete von erfolgreichen Änderungen an der Adapter-Hardware selbst, die er zur Eingrenzung des Problems vornahm.

Dieses Problem wird aktuell verfolgt und es wird aktiv nach einer Lösung gesucht. Sollte sich der Austausch von Hardwarekomponenten des Adapters bei solchen ‚Problem-Clones‘ als dauerhaft erfolgreich zeigen, so wird dies kommuniziert und im Platinenlayout berücksichtigt werden.

## 16.24 Ich finde keinen LPB- oder BSB-Anschluss, nur L-BUS und R-BUS?!

---

In diesem Fall schließe bitte den Adapter *NICHT* an und beachte das Kap. 10.2.3.

## 16.25 Gibt es eine (W)LAN-Option für den Adapter?

---

Ja, siehe die Kapitel 1.2.2 und 7.5, wenn du bereits ein Due-Setup benutzt. Solltest du noch keine Hardware angeschafft haben, so könntest du auch einen ESP32 nutzen.

## 16.26 Ich nutze das veraltete Setup Adapter v2 + Arduino Mega 2560 - muss ich irgendetwas beachten?

Ja! Siehe hierzu bitte [Anhang D](#).

## 16.27 Ich bekomme Fehlermeldungen von der Arduino IDE, was kann ich tun?

Fehlermeldungen seitens der Arduino IDE können vielfältig sein und verschiedene Gründe haben, daher kann hier nicht tiefer darauf eingegangen werden. Grundsätzlich sollten in dem Fall nochmals alle Einstellungen hinsichtlich Port, Boardtyp etc. überprüft werden. Wenn auch Google keine weiteren Hinweise liefert, kann selbstverständlich auch im Forum nachgefragt werden.

Exemplarisch sei hier jedoch ein Typ einer Fehlermeldung genannt, die auftritt, wenn der falsche Boardtyp bei Verwendung eines Arduino DUE eingestellt wurde:

```
BSB_lan:802:27: error: 'pgm_read_byte_far' was not declared in this scope

uint8_t second_char = pgm_read_byte_far(enum_addr + page + 1);

                        ^~~~~~
```

## 16.28 Es kann keine Verbindung zum WLAN-Netzwerk hergestellt werden

In diesem Fall sind grundsätzlich die WLAN-Zugangsdaten, die in der Datei *BSB\_LAN\_config.h* eingetragen werden, zu überprüfen. Ebenso die weiteren Netzwerkeinstellungen.

Ggf. ist auch die Routerkonfiguration zu überprüfen, ob sich neue WLAN-Geräte anmelden dürfen.

Wenn beim Starten eines ESP32-basierten Microcontrollers, bei dem man WiFi verwenden möchte, allerdings die folgenden zwei Fehlermeldungen auftreten

```
E (1593) esp.emac: emac_esp32_init(349): reset timeout
E (1594) esp_eth: esp_eth_driver_install(214): init mac failed
```

dann wurde das Definiment `#define WIFI` in der Datei *BSB\_LAN\_config.h* nicht aktiviert. Um es zu aktivieren, müssen die beiden Schrägstriche `//` davor entfernt und BSB-LAN erneut geflasht werden (siehe auch Kap. [2.2.2](#)).

## 16.29 BSB-LAN stürzt häufig ab oder die WLAN-Verbindung ist instabil.

Probleme mit Abstürzen oder instabilem WLAN sind häufig (auch) auf nicht ausreichende Stromversorgung zurückzuführen. Bei den Olimex Boards ist dies hin und wieder auch bei Stromversorgung über die DC-Buchse aufgefallen, eine Stromversorgung über die USB-Schnittstelle war in den Fällen stabiler.

## 16.30 Ich finde die Einstellung zum Schreiben der Parameter nicht / Es fehlt der „set“ Button

In der webbasierten Konfiguration muss der Schreibmodus erlaubt werden, dafür müssen zuvor die erweiterten Einstellungen im Webinterface aktiviert werden.

## 16.31 Warum finde ich nirgendwo den Parameter XYZ?

BSB-LAN kann nur die Parameter anzeigen, die auch als solche am Heizungsregler vorhanden sind. Bitte also nur dann eine Support-Anfrage stellen, wenn eine Parameternummer am Regler angezeigt wird, aber diese nicht in BSB-LAN auftaucht.

## 16.32 Die LED flackert, aber es kommt keine Verbindung zur Heizung zustande.

Insbesondere beim Olimex noch einmal die richtige Orientierung der Platine und deren mittigen Sitz überprüfen.



## 16.33 Ich habe weitere Fragen, an wen kann ich mich wenden?

---

Das Beste wäre, wenn du dich dafür im FHEM-Forum (<https://forum.fhem.de/>) anmelden würdest, da dort speziell für diesen Adapter ein eigener Thread existiert und sich dort eine nette und hilfsbereite Community findet. Hier findet ein reger Austausch über die Hard- und Software statt, Fragen werden meist zügig beantwortet und auf Updates wird hingewiesen.

Hier findest du den entsprechenden Thread: <https://forum.fhem.de/index.php/topic,29762.0.html>

Wenn du dich mit deinen Fragen vorstellst, gib uns bitte zuerst genaue Informationen bzgl. des von dir verwendeten Heizungstyps, des Reglers, des verwendeten Bus-Typs etc.

Wenn du den Adapter bereits erfolgreich angeschlossen und in Verwendung hast, rufe bitte einmal /Q ( <http://<IP-Adresse>/Q> ) auf und schreibe die Ausgabe zusätzlich mit in deine Beschreibung.

Prinzipiell kann man sagen: Lieber erst einmal zu viele Informationen, als zu wenige.

Fragen, deren Antworten sich aus dem gründlichen Lesen dieses Handbuchs ergeben, werden ab einem gewissen Punkt lediglich mit einem Verweis hierauf beantwortet.

Bitte bedenke, dass dies für jeden von uns nur ein Hobby-Projekt ist.



Support me on Ko-fi

[Weiter zu Kapitel 17](#)

[Zurück zum Inhaltsverzeichnis](#)

# 17. Weiterführende Informationen und Quellen

[Zurück zum Inhaltsverzeichnis](#)

[Zurück zu Kapitel 16](#)

## 17 Weiterführende Informationen und Quellen

Ein reger Austausch bzgl. der hier vorgestellten Hard- und Software findet in folgendem Forum statt:

<https://forum.fhem.de/index.php/topic,29762.0.html>

Dies ist auch eine gute Anlaufstelle für Fragen, Erfahrungsaustausch und Support, wo auch regelmäßig über Neuerungen und erfolgte Updates informiert wird.

Sämtliche Dokumentationen zur hier vorgestellten Hard- und Software sowie die verschiedenen Software-Versionen sind [hier](#) zu finden.

Dieses Handbuch ist zudem [hier](#) als PDF-Version erhältlich.

Die Software und die dazugehörigen Dokumentationen für den Einsatz des hier vorgestellten Adapters in Verbindung mit einem Raspberry Pi 2 ist [hier](#) zu finden.

Für die Nutzung des Adapters mit einem RPi an der PPS-Schnittstelle kann das Python-Script „[PPS-monitor](#)“ genutzt werden.

Die initiale Idee der Regleranbindung via BSB/LPB kann [hier](#) und [hier](#) nachvollzogen werden.

Als relativ umfangreiche Quelle mit vielen Parameterbeschreibungen sei das „Brötje Systemhandbuch ISR Plus“ empfohlen. Es stellt neben zahlreichen anderen und modellspezifischen Anleitungen die zugrunde liegende ‚Referenz‘ für die Parameterdefinitionen des hier vorgestellten Projekts dar.

Tieferegehende Informationen wie Spezifikationen und technische Anforderungen der Bus-Typen sind den jeweiligen Dokumenten der Hersteller zu entnehmen.

Speziell hinsichtlich des LPB seien zwei Dokumente von „Siemens Building Technologies - Landis & Staefa Division“ empfohlen:

- CE1N2030D Local Process Bus LPB Systemgrundlagen
- CE1N2032D Local Process Bus LPB Projektierungsgrundlagen

Hinsichtlich der Installation und Verwendung von DHT22- und OneWire-Sensoren wie dem DS18B20 gibt es zahlreiche Informationsquellen. Im Internet finden sich etliche kostenlose Anleitungen, Beispielinstallationen und Skripte.

Bei Problemen mit einer umfangreicheren OneWire-Installation sei ein Blick auf die technischen Anforderungen von OneWire (speziell hinsichtlich der Bus-Typologie und der Leitungslängen) und auf die Hinweise im entsprechenden Kapitel empfohlen.

 [Support me on Ko-fi](#)

[Weiter zu Anhang A1](#)

[Zurück zum Inhaltsverzeichnis](#)



# Anhang A2: Anmerkungen zum Schaltplan

[Zurück zum Inhaltsverzeichnis](#)

[Zurück zu Anhang A1](#)

## Anhang A2: Anmerkungen zum Schaltplan

Hinweis: Der Schaltplan für die ESP32-Variante des Adapters ist prinzipiell identisch, lediglich das EEPROM entfällt!

### A2.1 Kurze Legende zum Schaltplan

D1 = Leuchtdiode

D2 = Diode

EEPROM = EEPROM

OK(x) = Optokoppler

Q(x) = Transistor

R(x) = Widerstand

ARD = Arduino

RPI = Raspberry Pi

CL+/- = BSB-Anschluss

DB/MB = LPB-Anschluss

TXD = Digitalpin Senden

RXD = Digitalpin Empfangen

### A2.2 Teileliste

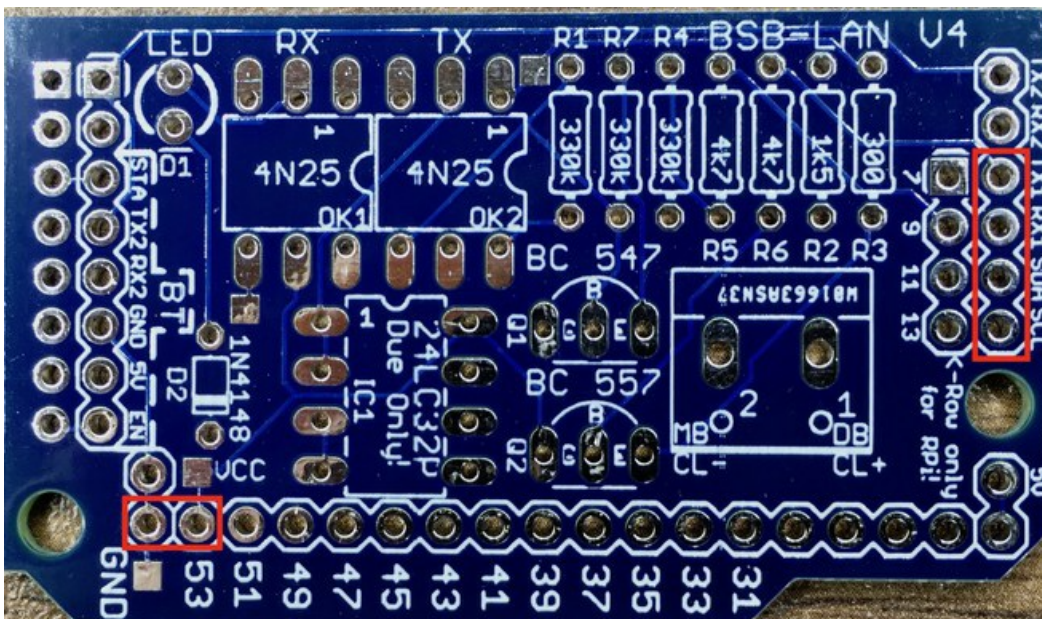
Anzahl	Komponente	Bezeichnung	Abbildung	Anmerkung
1	LED (rot)	D1		Betriebsspannung max. 2,8V, Sperrspannung 5V <b>Ausrichtung beachten!</b>
1	Diode 1N4148	D2		<b>Ausrichtung beachten!</b>
1	EEPROM 24LC32A-I/P	EEPROM		nicht benötigt für die ESP32-Boardvariante <b>Ausrichtung beachten!</b>
2	Optokoppler 4N25	OK1, OK2		<b>Ausrichtung beachten!</b>
1	Transistor BC547	Q1		<b>Achtung: Nicht mit Q2 verwechseln!</b> <b>Ausrichtung beachten!</b>

Anzahl	Komponente	Bezeichnung	Abbildung	Anmerkung
1	Transistor BC557	Q2		<b>Achtung: Nicht mit Q1 verwechseln!</b> <b>Ausrichtung beachten!</b>
3	Widerstand 330kΩ	R1, R4, R7		orange, orange, schwarz, orange, braun
1	Widerstand 1.5kΩ	R2		braun, grün, schwarz, braun, braun
1	Widerstand 300Ω	R3		orange, orange, schwarz, schwarz, braun
2	Widerstand 4.7kΩ	R5, R6		gelb, violett, schwarz, braun, braun
1	Anschlussklemme	CL+/CL		RM 5,08mm

#### Arduino Due:

Pinleisten (RM 2,54mm), ggf. IC-Sockel für Optokoppler und/oder EEPROM etc.

Für den Einsatz des Adapters v4 an einem *Arduino Due* werden letztlich lediglich die Pins RX1, TX1, SDA, SCL, GND sowie Pin 53 benötigt und müssen daher zwingend mit entspr. Pinleisten bestückt werden. Die anderen Pins können optional zur Verbesserung der Stabilität bestückt und/oder anderweitig genutzt werden.



Zwingend zu bestückende Pins für die Verwendung mit dem Arduino Due.

#### Raspberry Pi:

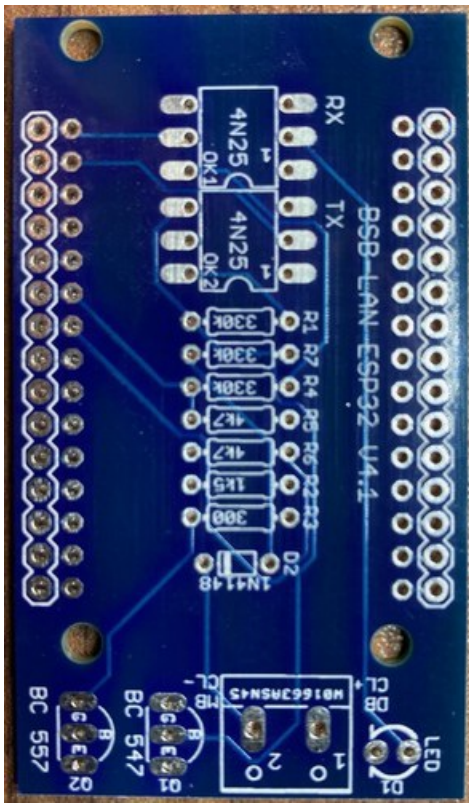
Buchsenleiste / weibliche Pinheader (doppelreihig, RM 2,54mm), ggf. IC-Sockel für Optokoppler und/oder EEPROM etc.

Für den Einsatz des Adapters v4 an einem *Raspberry Pi* sind andere Dinge zu beachten, die im [Kapitel 1.4](#) gesammelt aufgeführt sind.

#### ESP32:

Buchsenleiste / weibliche Pinheader (RM 2,54mm; ESP32: einreihig; Olimex: doppelreihig 2x5), ggf. IC-Sockel für Optokoppler etc.

Für den Einsatz des ESP32-spezifischen Adapters v4 an dem empfohlenen *ESP32 NodeMCU von Joy-It* werden letztlich lediglich die Pins RX2, TX2, GND sowie 3,3V benötigt und müssen daher zwingend mit entspr. Pinheadern bestückt werden. Aus Stabilitätsgründen ist es jedoch empfehlenswert, beide Seiten komplett mit je einer Reihe Pinheadern zu bestücken.



Die unbestückte ESP32-spezifische Adapterplatine.

## A2.3 Generelle Hinweise

**Vor dem Löten gilt: Bitte den Schaltplan aufmerksam studieren!**

Die folgenden Hinweise ersetzen kein grundsätzliches Elektronik-Vorwissen, könnten aber vielleicht dem einen oder anderen Elektronik-Anfänger eine kleine Hilfe sein.

Generell ist es hilfreich, die Bauteile erst einmal zu positionieren und die 'Beine' der Komponenten leicht umzubiegen, so dass sie nicht herausfallen können. Dabei empfiehlt es sich, mit den kleinsten Bauteilen zu beginnen. Wenn du die Platine fertig bestückt hast, prüfe den Aufbau erneut. Insbesondere die korrekte Ausrichtung von Bauteilen wie Dioden, Transistoren und der ICs solltest du nochmals überprüfen. Sieht soweit alles gut aus, kannst du die Platine umdrehen und mit dem Verlöten beginnen. Benutze dabei nicht zu viel Lötzinn für die einzelnen Lötstellen und achte darauf, dass du nicht versehentlich Kurzschlüsse produzierst, indem du Lötstellen ungewollt mit einander verbindest. Ein vorheriger Breadboard-Testaufbau ist natürlich eine Option, aber aufgrund nicht auszuschließender Problemquellen (Nutzung einer falschen Steckreihe, eventuelle Wackelkontakte o.ä.) nicht unbedingt empfehlenswert.

Bitte achte darauf, dass die Bauteile beim Löten nicht zu heiß werden, da sie u. U. Schaden nehmen können. Für den Einsatz der Optokoppler-ICs OK1 und OK2 sowie des EEPROMs bietet es sich daher an, einen entsprechenden IC-Sockel zu verwenden und die Optokoppler und das EEPROM erst nach Fertigstellung der Lötarbeiten in die Sockel zu stecken. Achte dabei auf die korrekte Ausrichtung der Sockel und Optokoppler/EEPROM; ebenso ist auf die korrekte Ausrichtung von bspw. Dioden und Transistoren zu achten!

Vor der Inbetriebnahme des Adapters ist es ratsam, die komplette Bestückung nochmals gründlich zu überprüfen und (falls möglich) durchzumessen. Kalte Lötstellen, versehentlich überbrückte Kontakte etc. können ein unerklärliches und schwer zu diagnostizierendes Fehlverhalten des Adapters bis hin zu einem eventuellen Reglerdefekt nach sich ziehen!

Viel Erfolg!

 Support me on Ko-fi

[Weiter zu Anhang B](#)

[Zurück zum Inhaltsverzeichnis](#)



# Anhang B: Pinouts

[Zurück zum Inhaltsverzeichnis](#)

[Zurück zu Anhang A2](#)

## Anhang B: Pinouts

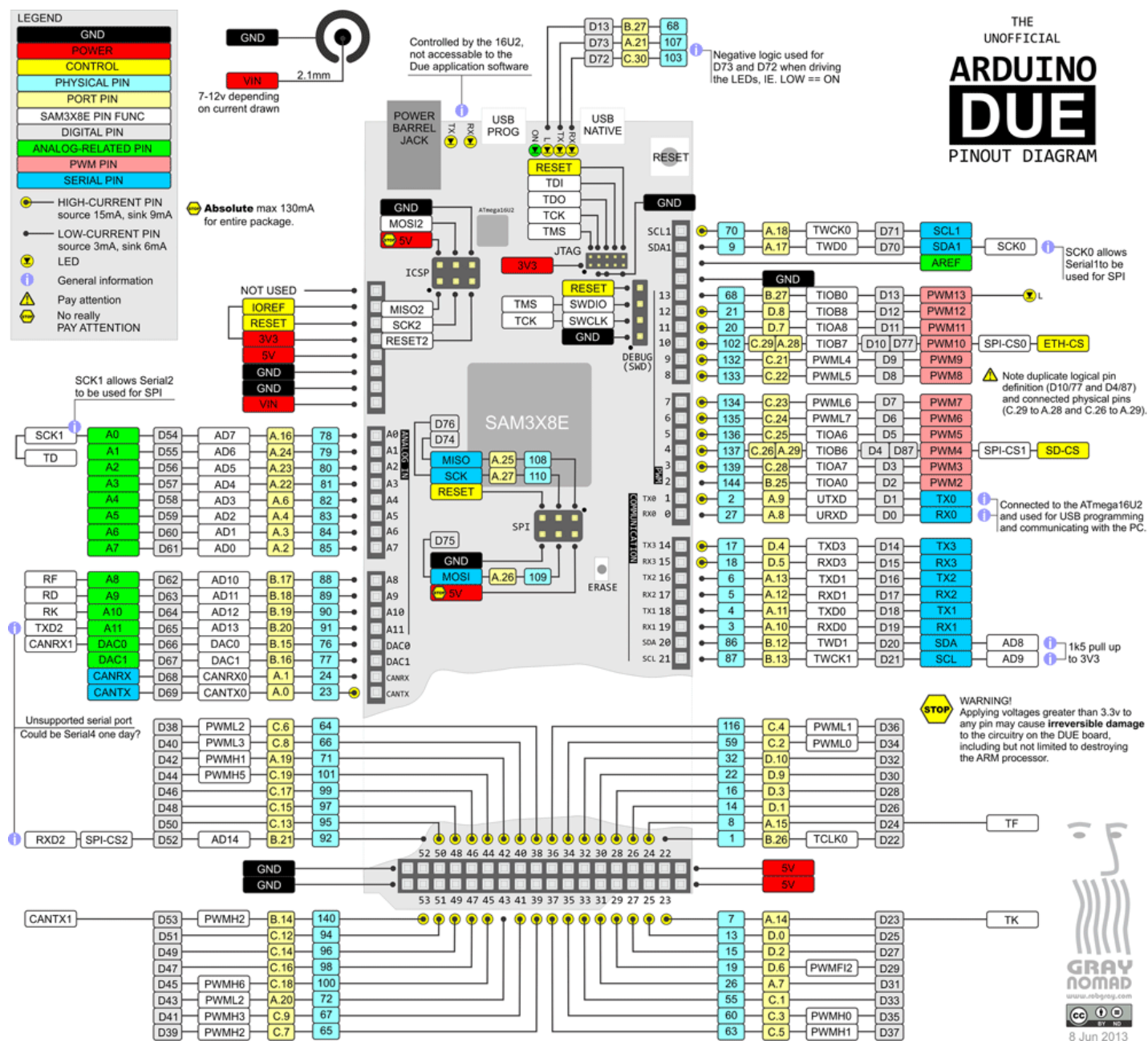
### B1 Arduino DUE

Das nachfolgend abgebildete 'inoffizielle' Arduino DUE Pinout-Schema stammt von [Rob Gray](#).

Es ist u.a. auch direkt als [PDF](#) verfügbar.

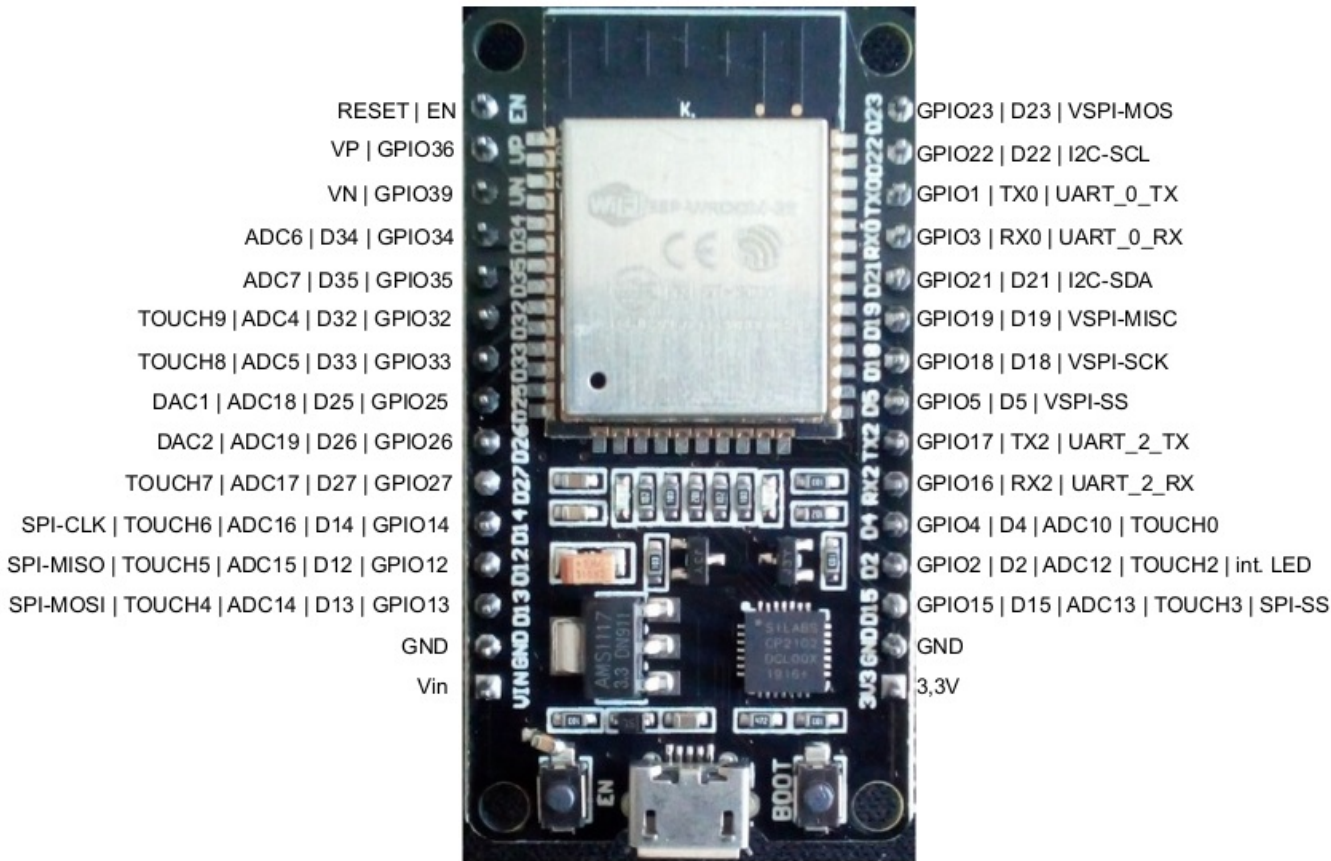
Vielen Dank für die großartige Arbeit!

Hinweis: Bitte beachte, dass das Pinout bei einigen günstigen Clones vom originalen Pinout abweichen kann!



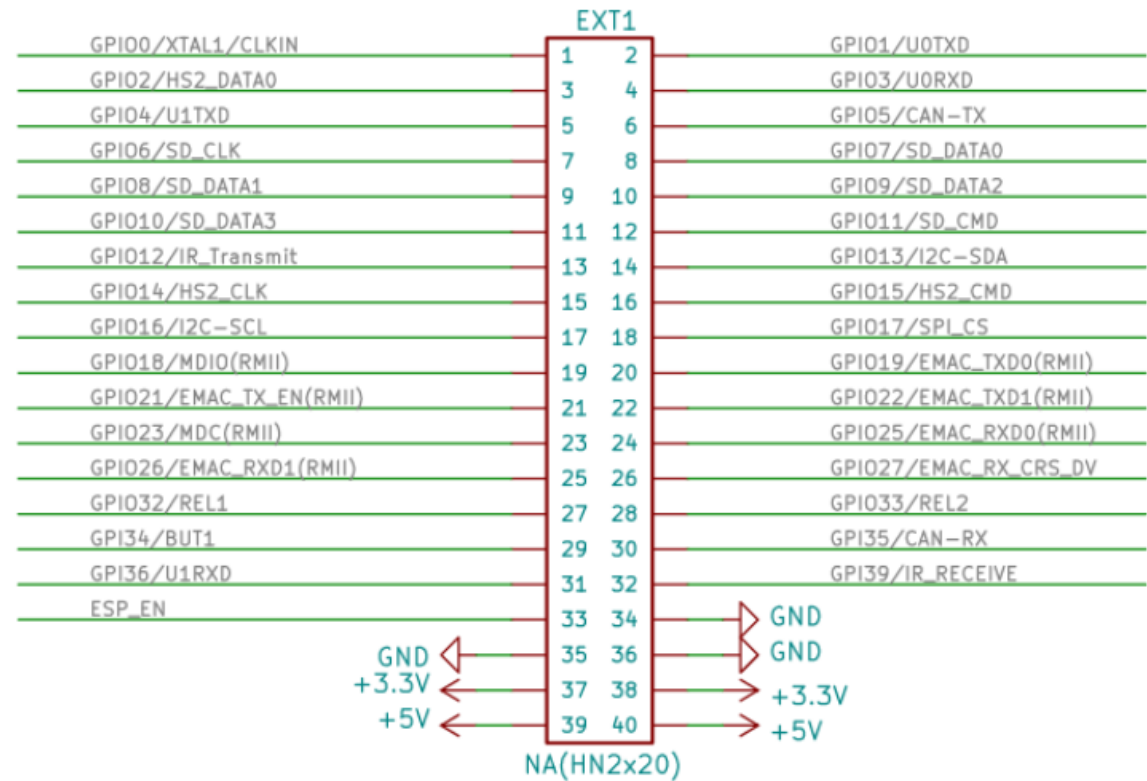
## B2 Joy-It ESP32 NodeMCU

Das Pinout-Schema für den Joy-It ESP32 NodeMCU ist in der [Bedienungsanleitung](#) des Herstellers abgebildet.



## B3 Olimex ESP32-EVB

Das Pinout für den Olimex ESP32-EVB ist im [Schaltplan](#) einsehbar. Die nachfolgende Abbildung zeigt den entspr. Ausschnitt daraus und bezieht sich auf die 40 polige Stiftleiste.



Pinout der 40 poligen Pinleiste des Olimex ESP32-EVB.



Support me on Ko-fi

[Weiter zu Anhang C](#)

[Zurück zum Inhaltsverzeichnis](#)

# Anhang C: Changelog BSB-LAN-Software

[Zurück zum Inhaltsverzeichnis](#)

[Zurück zu Anhang B](#)

## Anhang C: Changelog BSB-LAN-Software

### Version 2.2

- ATTENTION: Several variables in BSB\_LAN\_config.h.default have changed their variable type, it's probably best to re-create your BSB\_LAN\_config.h from scratch.
- Parameter numbers are now floating point (i.e. XXXX.Y) because some parameters contain two different kinds of information. These are now shown in decimal increments of 0.1. You can still query the "main" parameter via XXXX (without .Y)
- Lots of bugfixes and new data types
- Device-specific parameter lists supported

### Version 2.1

- Many new parameters for LMU64
- ATTENTION: New categories for LMU64 and RVD/RVP controllers due to their different numbering schemes. Will be filled over time. PPS and sensor categories have moved up by two.
- ESP32: OTA now uses system-wide HTTP AUTH authentication credentials
- Improved built-in chart display (/DG), new configuration definement #define USE\_ADVANCED\_PLOT\_LOG\_FILE - thanks to Christian Ramharter
- Lots of bugfixes

### Version 2.0

- ATTENTION: LOTS of new functionalities, some of which break compatibility with previous versions, so be careful and read all the docs if you make the upgrade!
- ATTENTION: Added and reorganized PPS parameters, almost all parameter numbers have changed!
- ATTENTION: Change of EEPROM layout will lead to loading of default values from BSB\_LAN\_config.h! You need to write settings to EEPROM in configuration menu again!
- ATTENTION: Folder locations and filenames have been adjusted for easier installation! If you update your installation, please take note that the configuration is now in BSB\_LAN\_config.h (LAN in caps), and no longer in BSB\_lan\_config.h (lower-caps "lan")
- ATTENTION: HTTP-Authentication configuration has changed and now uses plain text instead of Base64 encoded strings!
- Thanks to GitHub user do13, this code now also compiles on a ESP32, tested on NodeMCU-ESP32, Olimex ESP32-POE and Olimex ESP32-EVB boards. ESP32 code uses SDK version 2.0.2, please take note when configuring Arduino IDE!
- OTA Updates now possible for ESP32-based devices
- Support for special PPS devices (based on DC225/Honeywell MCBA) added
- Webinterface allows for configuration of most settings without the need to re-flash, also split into basic and extended configuration
- Added better WiFi option for Arduinos through Jiri Bilek's WiFiSpi library, using an ESP8266-based microcontroller like Wemos D1 mini or LoLin NodeMCU. Older WiFi-via-Serial approach no longer supported.
- Added MDNS\_SUPPORT definement in config so that BSB-LAN can be discovered through mDNS
- If BSB-LAN cannot connect to WiFi on ESP32, it will set up its own access point "BSB-LAN" with password "BSB-LPB-PPS-LAN" for 30 minutes. After that, it will reboot and try to connect again.
- New MQTT functions, including allowing any parameter to be set by an MQTT message and actively query any parameter once by sending

an MQTT message

- Added support for BME280 sensors
- Setting a temporary destination address for querying parameters by adding !x (where x is the destination id), e.g. /6224!10 to query the identification of the display unit
- URL commands /A, /B, /T and /JA have been removed as all sensors can now be accessed via parameter numbers 20000 and above as well as (currently) under new category K49.
- New categories added, subsequent categories have been shifted up
- HTTP Authentification now uses clear text username and password in configuration
- PPS users can now send time and day of week to heater
- Lots of new parameters added
- URL command /JR allows for querying the standard (reset) value of a parameter in JSON format
- URL command /JB allows for backing up parameters to JSON file
- New library for DHT22 should provide more reliable results
- Consolidated data and value types: New data types VT\_YEAR, VT\_DAYMONTH, VT\_TIME as subsets of VT\_DATETIME for parameters 1-3, replacing VT\_SUMMERPERIOD and adjusting VT\_VACATIONPROG. New value types DT\_THMS for time consisting of hour:minutes:seconds
- MQTT: Use MQTTDeviceID as a client ID for the broker, still defaults to BSB-LAN. ATTENTION: Check your config if you're broker relies on the client ID in any way for authorization etc.

#### Version 1.1

- ATTENTION: DHW Push ("Trinkwasser Push") parameter had to be moved from 1601 to 1603 because 1601 has a different "official" meaning on some heaters. Please check and change your configuration if necessary
- ATTENTION: New categories added, most category numbers (using /K) will be shifted up by a few numbers.
- /JA URL command outputs average values
- Many new parameters decoded
- New parameters for device families 25, 44, 51, 59, 68, 85, 88, 90, 96, 97, 108, 134, 162, 163, 170, 195, 209, 211
- Improved mobile display of webinterface
- Added definement "BtSerial" for diverting serial output to Serial2 where a Bluetooth adapter can be connected (5V->5V, GND->GND, RX->TX2, TX->RX2). Adapter has to be in slave mode and configured to 115200 bps, 8N1.
- Lots of added Polish translations
- New data types VT\_BYTE10, VT\_SPF
- Bugfix for PPS bus regarding display of heating time programs
- Bugfix for MQTT

#### Version 1.0

- /JI URL command outputs configuration in JSON structure
- /JC URL command gets list of possible values from user-defined list of functions. Example: /JC=505,700,701,702,711,1600,1602
- Logging telegrams (log parameter 30000) now writes to separate file (journal.txt). It can be reset with /D0 (same time with datalog.txt) command and dumped with /DJ command.
- removed WIFI configuration as it is no longer applicable for the Due
- lots of new parameters for various device families

- Code optimization and restructuring, general increase of speed
- new schematics for board layout V3
- lots of bugfixes

#### Version 0.44

- Added webserver functionality via SD card and various other improvements from GitHub user dukess
- Added JSON output for MQTT
- mobile friendlier web interface
- more parameters and device families
- last version completely tested on Mega 2560. Future versions may still run on the Mega, but will only be tested on the Arduino Due.

#### Version 0.43

- Added support for HardwareSerial (Serial1) connection of the adapter. Use RX pin 19 in bus() definition to activate. See manual/forum for hardware details.
- Added definement DebugTelnet to divert serial output to telnet client (port 23, no password) in BSB\_lan\_config.h
- Added possibility to control BSB-LAN (almost?) completely via USB-serial port. Most commands supported like their URL-counterparts, i.e. //xxx to query parameter xxx or //N to restart Arduino.
- Changed default device ID from 6 (room controller "RGT1") to unused ID 66 ("LAN")
- Many new parameters, please run /Q to see any possible changes for your device family and report back to us!
- Added global variables (arrays of 20 bytes) custom\_floats[] and custom\_longs[] for use with BSB\_lan\_custom.h, for example to read sensors etc. Output of these variables is done via new URL command /U
- Added device families 23 and 29 (Grünenwald heaters)
- Added device families 49, 52, 59 (Weishaupt heaters)
- Added device families 91, 92, 94, 118, 133, 136, 137, 165, 184, 188 (various controllers like QAA75 or AVS37)
- Added device family 171 (Bösch wood pellet system)
- Added device family 172 (SensoTherm BLW Split B (RVS21.826F/200))
- Added device families 186 and 164 (Olymp WHS-500)
- Added device family 195 variant 2 (Thision 19 Plus / LMS14.111B109)
- Including DHT, 1Wire and burner status parameters (>20000) to MQTT
- English is now default language
- Updated various translations
- Added STL files to print a case with a 3D printer (thanks to FHEM user EPo!)
- Moved all sensors to /T , /H is now no longer used
- New virtual parameters 702/703 for Weishaupt room controller
- New virtual parameter 10003 to set outside temperature on newer systems
- Added text descriptions for error phases (6706 ff.)
- /Q is now more comprehensive
- New data types VT\_CUSTOM\_ENUM and VT\_CUSTOM\_BYTE to extract information from non-standard telegrams (such as 702/703)
- Bugfix: DHCP (ethernet) implementation



- Added localization! Now you can help translate BSB-LAN into your language! Simply copy one of the language files from the localization folder (LANG\_DE.h is the most complete) and translate whatever you can. Non-translated items will be displayed in German. Attention: Language definition in BSB\_lan\_config.h is now `#define LANG` For example: `#define LANG DE`
- Added export to MQTT broker, use `log_parameters[]` in BSB\_lan\_config.h to define parameters and activate MQTTBrokerIP definement.
- Added support for WiFi modules such as an ESP8266 or a Wemos Mega connected to Serial3 (RX:15/TX:14) of the Arduino. The ESP8266 has to be flashed with the AT firmware from Espressif to work. Please take note that WiFi over serial is by design much slower (only 115kpbs) than "pure" TCP/IP connections.
- Added new category "34 - Konfiguration / Erweiterungsmodule". All subsequent categories move one number up!
- Lots of new parameters coming from device family 123, please run `/Q` to see if some parameters also work for your heater!
- Lots of new yet unknown parameters through brute force querying :) (parameter numbers 10200 and above)
- Added further PPS-Bus commands, moved parameter numbers to 15000 and above
- Default PPS mode now "listening". Use third parameter of bus definition to switch between listening and controlling, 1 stands for controlling, everything else for listening, i.e. BSB bus(68,67,1) sends data to the heater, BSB bus(68,67) only receives data from heater / room controller. You can switch between modes at run-time with URL command `/P2,x` where x is either 1 (for controlling) or not 1 (for listening only)
- Fixed bug that crashed PPS bus queries
- Stability improvements for PPS bus
- Improved graph legend when plotting several parameters
- Added JSON export; query with `/JQ=a,b,c,d...` or push queries to `/JQ` or push set commands to `/JS`
- Logging of MAX! parameters now possible with logging parameter 20007
- Added Waterstage WP device family (119)
- Added WHG Procon device family (195)
- Added unit to log file as well as average output
- Rewrote device matching in `cmd_tbl` to accomodate also device variant (Gerätevariante). Run `/Q` to see if transition has worked for your device!
- Added `BSB_lan_custom_setup.h` and `BSB_lan_custom_global.h` for you to add individual code (best used in conjunction with `BSB_lan_custom.h`)
- Marked all (known) OEM parameters with flag `FL_OEM`. OEM parameters are set by default as read-only. To make them writeable, change `FL_OEM` from 5 to 4 in `BSB_lan_defs.h`
- Increased performance for querying several parameters at once (similar to category query)
- Added config option to define subnet.
- `/Q` no longer needs `#define DEBUG`
- Bugfix ENUM memory adresssing
- Bugfix in reset function (`/N`), clear EEPROM during reset with `/NE`
- Added `favicon.ico`
- Split of `cmdtbl` into `cmdtbl1` and `cmdtbl2` due to Arduino's(?) limit of 32kB size of struct, opening up more space for new parameters.

- Interim release containing all changes from 0.42 above, except locaization, i.e. all text fragments are still part of the main code.

- Implemented polling of MAX! heating thermostats, display with URL command /X. See BSB\_lan\_custom.h for an example to transmit average room temperature to heating system.
- Added new category "22 - Energiezähler" - please note that all subsequent categories move one up!
- New virtual parameter 1601 (manual TWW push)
- Added Fujitsu Waterstage WSP100DG6 device family (211)
- Added CTC device family (103)
- New definement "#define TRUSTED\_IP2" to grant access to a second local IP address
- Added optional definement "#define GatewayIP" in BSB\_lan\_config.h to enable setting router address different from x.x.x.1
- Removed parameter 10109 because it is the same as 10000
- Added function to check all known CommandIDs on your own heating system. Use /Q after enabling definement "#define DEBUG" in BSB\_lan\_config.h
- Added parameter numbers to category menu
- Updated analyze.sh
- hopefully fixing the memory issue
- Moved HTML strings to html\_strings.h

Version 0.39 -- 02.01.2018

- Implementation of PPS-Bus protocol. See /K40 for the limited commands available for this bus. Use setBusType(2) to set to PPS upon boot or /P2 to switch temporarily.
- Set GPIOs to input by using /Gxx,I
- Definement "#define CUSTOM\_COMMANDS" added. Use this in your configuration to include individual code from "BSB\_lan\_custom.h" (needs to be created by you!) which is executed at the end of each main loop. Variables "custom\_timer" and "custom\_timer\_compare" have been added to execute code at arbitrary intervals.
- Added LogoBloc Unit L-UB 25C device family (95)
- several new parameters added
- Bugfix for logging Brennerlaufzeit Stufe 2

Version 0.38 -- 22.11.2017 ATTENTION: New BSB\_lan\_config.h configurations! You need to adjust your configuration when upgrading to this version!

- Webserver port is now defined in #define Port xx
- IP address is now defined in #define IPAddr 88,88,88,88 form - note the commas instead of dots!
- Special log parameters 20002 to 20006 have changed, see BSB\_lan\_config.h for their new meaning
- Added new virtual parameter 701 (Präsenztaste) which enters reduced temperature mode until next timed switch
- Added Brötje BOB device family (138), including many new parameters!
- Added Brötje SOB26 device family (28)
- Added Elco Aquatop 8es device family (85)
- Added Elco Thision 13 Plus device family (203)
- Added Weishaupt WTU 25-G familiy (50)
- Added output for absolute humidity (g/m3) for DHT22 sensors

- New schematics for Arduino/Raspberry board layout
- Included support for W5500 Ethernet2 shields. Activate definement ETHERNET\_W5500 in BSB\_lan\_config.h
- Including two-stage oil furnaces BC-counters and logging - please note that logging parameters have been adjusted, see BSB\_lan\_config.h for new values!
- Added new options for commands /P and /S to allow specifying a different destination device during runtime
- Added new configuration definement CUSTOM\_COMMANDS which includes BSB\_lan\_custom.h at the end of each main loop. You may use custom\_timer (set to current millis()) and custom\_timer\_compare to execute only every x milliseconds.
- Bugfixing SD-card logging in monitor mode
- Bugfix for setting hour:time parameters via webinterface

#### Version 0.37 -- 08.09.2017

- LPB implementation! More than 450 parameters supported! Switch temporarily between LPB and BSB with the Px command (0=BSB, 1=LPB) or use the setBusType config option to set bus-type at boot-time. Parameter numbers are the same as for BSB.

#### Version 0.36 -- 23.08.2017

- bugfix: brought back VT\_BIT list of options which were erroneously deleted :), fixed/freed several memory issues

#### Version 0.35 -- 25.06.2017

- new category "Sitherm Pro"; caution: category numbers all move up by one, starting from category "Wärmepumpe" (from 20 to 21) onwards.
- graph display of logging data now comes with crosshair and shows detailed values as tooltip
- improved SD-card output by factor 3 (from 16 to 45 kbps), switching SD-card library from SD.h to SdFat.h (<https://github.com/greiman/SdFat>) brings another 10% performance boost
- adjusted paths and directory layout of SdFat to enable compiling from sketch directory.
- new data type vt\_sint for signed int data, currently only used in some Sitherm Pro parameters

#### Version 0.34 -- 29.05.2017

- Log data can now be displayed as graph
- Webinterface can now display and set vt\_bit type parameters in human-readable form
- added KonfigRGx descriptions; caution: various sources used, no guarantee that descriptions match your individual heating system!
- vt\_bit is generally read-only in the webinterface. To set, use URL command /S with decimal representation of value
- fixed a bug with vt\_seconds\_short5, affecting parameters 9500 and 9540.
- fixed bug regarding Fujitsu's device family (from 127 to 170)
- moved libraries from folder libraries to src so they can be included without copying them to the Arduino libraries folder
- modified DallasTemperature.h's include path for OneWire.h

#### Version 0.33 -- 09.05.2017

- no more heating system definements anymore due to new autodetect function based on device family (parameter 6225), or set device\_id variable to parameter value directly
- two more security options: TRUSTED\_IP to limit access to one IP address only, and HTTP authentication with username and password
- Average values are saved on SD-card if present and LOGGER definement is activated
- deactivate logging by setting /L0=0 - this way you can enable LOGGER definement without filling up SD card but still save average values
- new error codes for THISION

- added dump of data payload on website for commands of unknown type, greyed out unsupported parameters
- enable logging of telegrams (log parameter 30000) also in monitor mode (bsb.cpp and bsb.h updated)
- time from heating system is now retrieved periodically from broadcast telegrams, further reducing bus activity
- new data type vt\_bit for parameters that set individual bits. Display as binary digits, setting still using decimal representation
- new data type vt\_temp\_short5\_us for unsigned one byte temperatures divided by 2 (so far only 887 Vorlaufsoll NormAusstemp)
- new data type vt\_percent5 for unsigned one byte temperatures divided by 2 (so far only 885 Pumpe-PWM Minimum)
- new data type vt\_seconds\_word5 for two byte seconds divided by 2 (so far only 2232, 9500 and 9540)
- new data type vt\_seconds\_short4 for (signed?) one byte seconds divided by 4 (so far only 2235)
- new data type vt\_seconds\_short5 for (signed?) one byte seconds divided by 5 (so far only 9500, 9540)
- new data type vt\_speed2 for two byte rpm (so far only 7050)
- cleaned up set() function from apparent duplicate cases
- added cases for vt\_temp\_word, vt\_seconds\_word5, vt\_temp\_short, vt\_temp\_short5, vt\_seconds\_short4 to set() function

Version 0.32 -- 18.04.2017

- lots of new parameters supported
- newly designed webinterface allows control over heating system without any additional software or cryptic URL commands. URL commands of course are still available, so no need to change anything when using FHEM etc.
- German webinterface available with definement LANG\_DE
- new URL-command /LB=x to log only broadcast messages (x=1) or all bus messages (x=0)
- new URL-command /X to reset the Arduino (need to enable RESET definement in BSB\_lan\_config.h)
- new logging parameters 20002 and 20003 for hot water loading times and cycles
- moved DS18B20 logging parameters from 20010-20019 to 20200-20299 and DHT22 logging parameters from 20020-20029 to 20100 to 20199
- moved average logging parameter from 20002 to 20004
- set numerous parameters to read-only because that's what they obviously are (K33-36)
- various bugfixes

Version 0.31 -- 10.04.2017

- increased dumping of logfile by factor 5 / as long as we still have memory left, you can increase logbuflen from 100 to 1000 to increase transfer speed from approx. 16 to 18 kB/s
- adjusted burner activity monitoring based on broadcast messages for Brötje systems
- removed definement PROGNR\_5895 because so far, it has only disabled an ENUM definition.
- removed definement PROGNR\_6030 because double command ID could be resolved via BROETJE / non-BROETJE definements
- renamed BROETJE\_SOB to BROETJE in order to allow for fine-grained distinction between different BROETJE cases (e.g. 6800ff). This means you have to activate TWO definements when using a Brötje system now: The general BROETJE as well as BROETJE\_SOB or BROETJE\_BSW. Have a look at your serial log for parameters 6800 to see which command IDs fit your system and activate one of both accordingly.
- changed 16-Bit addressing of flash memory to 32-Bit to address crashes due to ever growing PROGMEM tables - now we have lots of air to breathe again for new command IDs :)
- removed trailing \0 string from several ENUMs that led to wrong ENUM listings. Please keep in mind not to end ENUMs with a trailing \0 !

#### Version 0.30 -- 22.03.2017

- Time library by Paul Stoffregen (<https://github.com/PaulStoffregen/Time>) is now required and included in the library folder.
- adds logging of raw telegram data to SD card with logging parameter
- Logging telegram data is affected by commands /V and /LU
- adds command /LU=x to log only known (x=0) or unknown (x=1) command IDs when logging telegram data
- removed define USE\_BROADCAST, broadcast data is now always processed
- new internal functions GetDateTime, TranslateAddr, TranslateType

#### Version 0.29 -- 07.03.2017

- adds command /C to display current configuration
- adds command /L to configure logging interval and parameters
- adds option for command /A to set 24h average parameters during runtime
- adds special parameter 20002 for logging /A command (24h averages, only makes sense for long logging intervals)
- bugfixes for logging DS18B20 sensors

#### Version 0.28 -- 05.03.2017

- adds special parameters 20000++ for SD card logging of /B, /T and /H commands (see BSB\_lan\_config.h for examples)
- adds version info to BSB\_LAN web interface

#### Version 0.27 -- 01.03.2017

- adds date field to log file (requires exact time to be sent by heating system)
- /D0 recreates datalog.txt file with table header
- added "flags" field to command table structure. Currently, only FL\_RDONLY is supported to make a parameter read-only
- added DEFAULT\_FLAG in config. Defaults to NULL, i.e. all fields are read/writeable. Setting it to FL\_RDONLY makes all parameters read-only, e.g. for added level of security. Individual parameters can be set to NULL/FL\_RDONLY to make only these parameters writable/read-only.

#### Version 0.26 -- 27.02.2017

- added functionality for logging on micro SD card, using the slot of the w5100 Ethernet shield
- more parameters added (e.g. 8009)

#### Version 0.25 -- 21.02.2017

- more FUJITSU parameters added

#### Version 0.24 -- 14.02.2017

- updated README with added functions
- added German translations of FAQ and README, courtesy of Ulf Dieckmann

#### Version 0.23 -- 12.02.2017

- minor bugfix

#### Version 0.22 -- 07.02.2017

- more FUJITSU parameters
- (hopefully) correct implementation of VT\_VOLTAGE readings

- minor bugfixes

Version 0.21 -- 06.02.2017

- added numerous parameters for Fujitsu Wärmepumpe, including new #define FUJITSU directive to activate these parameters due to different parameter numbers
- minor bugfixes

Version 0.20 -- 27.01.2017

- added more parameters for Feststoffkessel
- minor bugfixes

Version 0.19 -- 01.01.2017

- added humidity command \"H\", currently for DHT22 sensors
- added 24h average command \"A\", define parameters in BSB\_lan\_config.h
- removed trailing whitespace from menu strings
- fixed command id 0x053D04A2 for THISION heaters
- included Rob Tillaart's DHT library because there are various libraries implementing the protocol and this one is used in the code for its ability to address multiple sensors with one object.
- removed /temp URL parameter as it is a duplicate of /T
- included loop to display DHT22 sensors in IPWE
- making compiling IPWE extensions optional (#define IPWE)

Version 0.18 -- 22.12.2016

- split off configuration into bsb\_lan\_config.h
- split off command definitions into bsb\_lan\_defs.h
- changed GPIO return values from LOW/HIGH to 1/0
- reactivated and updated IPWE (define parameters in config)
- check for protected pins when accessing GPIO (define in config)
- added schematics and PCB files to new subfolder \"schematics\"

Version 0.17a -- 20.12.2016

- minor errors corrected

Version 0.17 -- 20.12.2016

- merged v0.16 with FHEM user miwi's changes

Version 0.16 -- 20.11.2016

- removed IPWE and EthRly interface
- added GPIO interface
- merged parameters from J.Weber
- resolved duplicate command IDs

Version 0.15a -- 25.07.2016

- collated the commands from a Python project and this project, merged the two versions, corrected obvious errors. Inserted hypothetical numerical values in ENUM definitions where Broetje manuals documented only the message texts.



- added information from traces in a Broetje installation with an ISR-SSR controller and a WOB 25C oil furnace.

Version 0.15 -- 21.04.2016

- added Solar and Pufferspeicher from Elco Logon B & Logon B MM

Version 0.14 -- 04.04.2016

- minor bugfixes for Broetje SOB
- extended broadcast handling (experimental)

Version 0.13 -- 31.03.2016

- change resistor value in receiving path from 4k7 to 1k5
- added values 0x0f and 0x10 to Enum8005
- fixed strings for Zeitprogramme
- added timeout for sending a message (1 second)
- added option T for querying one wire temperature sensors in mixed queries
- added special handling for Broetje SOB
- simplified settings

Version 0.12 -- 09.04.2015

- added ONEWIRE\_SENSORS to ipwe
- fixed parameter decoding for ELCO Thision heating system

Version 0.11 -- 07.04.2015

- fixed parameter decoding for ELCO Thision heating system

Version 0.10 -- 15.03.2015

- added more parameters for ELCO Thision heating system

Version 0.9 -- 09.03.2015

- added more parameters for ELCO Thision heating system
- printTelegramm returns value string for further processing

Version 0.8 -- 05.03.2015

- added parameters for ELCO Thision heating system
- added IPWE extension
- minor bugfixes

Version 0.7 -- 06.02.2015

- added bus monitor functionality

Version 0.6 -- 02.02.2015

- renamed SoftwareSerial to BSBSoftwareSerial
- changed folder structure to enable simple build with arduino sdk

Version 0.5 -- 02.02.2015

- bugfixes

- added documentation (README)
- added passkey feature
- added R feature (query reset value)
- added E feature (list enum values)
- added setter for almost all value types
- fixed indentation
- added V feature to set verbosity for serial output
- set baudrate to 115200 for serial output
- redirecting favicon request
- added some images of the BSB adapter

Version 0.1 -- 21.01.2015 -- initial version

 Support me on Ko-fi

[Weiter zu Anhang D](#)

[Zurück zum Inhaltsverzeichnis](#)

# Anhang D: Hinweise für Nutzer des veralteten Setups Adapter v2 + Arduino Mega 2560

[Zurück zum Inhaltsverzeichnis](#)

[Zurück zu Anhang C](#)

## Anhang D: Hinweise für Nutzer des veralteten Setups Adapter v2 + Arduino Mega 2560

Für Nutzer des veralteten Setups sind im Folgenden einige Fragen und Punkte aufgeführt, die evtl. der Klärung bedürfen oder die es zu beachten gilt. Etwaige weitere Fragen diesbzgl. stelle bitte im entspr. [Thread des FHEM Forums](#).

Bitte habe jedoch Verständnis, dass wir nicht auf Fragen eingehen werden, die sich bspw. darauf beziehen, sich nach der erfolgten Umstellung auf den Adapter v3/v4 jetzt noch einen Adapter v2 zu bauen.

**PCBs v2 sind nicht mehr verfügbar, Stand der Technik ist die Kombination Adapter v4.x + Due/ESP32.**

### *Hinweis:*

*Es ist möglich, den Adapter v2 durch eine Vollbestückung und kleinere Anpassungen mit einem ESP32 zu verwenden. Auf diese Weise könnte die aktuelle BSB-LAN-Version genutzt werden, ohne auf den aktuellen Adapter wechseln zu müssen. Für weitere Informationen lies bitte das [Kap. 1.3.3](#).*

- **Muss ich zwingend auf das neue Setup Adapter v3/v4 + Due wechseln?**

Nein, wenn du zufrieden mit dem veralteten Setup bist und der Funktionsumfang von BSB-LAN deinen Ansprüchen bisher genügt, dann kannst du das alte Setup natürlich weiterhin verwenden.

- **Gibt es bzgl. der BSB-LAN-Versionen etwas zu beachten?**

Ja. Die letzte 'offiziell' getestete und empfohlene Version für dein Setup ist die [Version v0.44](#). Im zip-file befindet sich auch die letzte 'Mega2560-spezifische' Version des Handbuchs (als PDF).

Aber: Es hat sich bei mehreren Usern gezeigt, dass auch die [v1.1](#) noch ohne große Einschränkungen läuft, aufgrund des Speichermangels des Mega 2560 vermutlich aber schon nicht mehr mit allen verfügbaren Optionen, die BSB-LAN bietet.

Ab **v2.x** ist es dann definitiv nötig, einzelne Module zu deaktivieren und somit auf spezifische Funktionen zu verzichten, die BSB-LAN bietet. Hinweise diesbzgl. findest du in [Kap. 2.2.2](#) bzw in den Kommentaren der Datei `BSB_LAN_config.h`. Besonderes Augenmerk ist auf die letzten Punkte zu richten, die u.a. ein komfortables Deaktivieren einzelner Module (bspw. Webconfig, MQTT, IPWE etc.) an zentraler Stelle ermöglicht.

- **Was gilt es zu beachten, wenn ich die aktuelle v2.x nutzen möchte?**

Wie bereits erwähnt muss die v2.x in der Konfiguration prinzipiell so angepasst werden, dass sie mit dem geringeren Speicher des Mega problemlos kompiliert und lauffähig ist. Neben dem bereits erwähnten Deaktivieren einzelner Module gibt es weitere Möglichkeiten:

1.) Die Größe der Variablen von bspw. `PASSKEY[]`, `avg_parameters[]`, `log_parameters[]`, `ipwe_parameters[]`, `max_device_list[]` kann verkleinert werden (wenn bspw. weniger Parameter als maximal möglich verwendet werden), um ein wenig Speicher einzusparen.

2.) In der Datei `BSB_LAN_config.h` finden sich im unteren Abschnitt verschiedene Definements für Mega-User, die bei bestimmten Einsatzszenarien aktiviert werden können, um nochmals Speicher zu sparen. Hinweise diesbzgl. findest du in der Datei selbst.

3.) Erstellung einer reglerspezifischen `BSB_LAN_defs.h`:

Im Repo liegt ein Perlscript namens `selected_defs.pl` sowie ein Windows-Executable namens `selected_defs.exe`, das die Datei `BSB_LAN_defs.h` nach ausgewählten Gerätefamilien filtert und eine spezifische Datei für den eigenen Reglertyp erstellt. Die Ersparnis beträgt im Schnitt etwa 20 bis 25 kB Flash-Speicher, den man dann für die (Re-)Aktivierung von anderen Funktionen nutzen kann. Im Falle eines Reglerwechsels (= andere Gerätefamilie) muss die Datei natürlich entsprechend neu generiert werden.

Das Script läuft unter Perl, was auf Mac- und Linux-Rechnern standardmäßig installiert ist, lässt sich aber auch auf Windows nachinstallieren.

Vorgehensweise zur Erstellung einer reglerspezifischen defs-Datei:

- Parameter 6225 "Gerätefamilie" via BSB-LAN abrufen und den Wert notieren.
- Datei `selected_defs.pl` bzw. `selected_defs.exe` vor dem Ausführen in den gleichen Ordner kopieren, in dem auch die Datei

*BSB\_LAN\_defs.h* liegt.

- Öffne ein Terminal, wechsele in den entspr. Ordner und erstelle die reduzierte Datei namens *BSB\_LAN\_defs\_filtered.h* mit Hilfe des Perlscripts bzw. des Windows-Executables, die nur die für die spezifische Gerätefamilie(n) relevanten Parameter enthält. Bei nur einem angeschlossenen Regler, bspw. mit der Gerätefamilie 162, lautet der Befehl

```
./selected_defs.pl 162 > BSB_LAN_defs_filtered.h bzw.
```

```
selected_defs.exe 162 > BSB_LAN_defs_filtered.h .
```

Wenn man bspw. zwei Geräte am Bus mit den Gerätefamilien 162 und 90 hat, kann man den Befehl um den zweiten Wert erweitern:

```
./selected_defs.pl 162 90 > BSB_LAN_defs_filtered.h bzw.
```

```
selected_defs.exe 162 90 > BSB_LAN_defs_filtered.h .
```

- Verschiebe die originale Datei *BSB\_LAN\_defs.h* aus dem "BSB\_LAN"-Verzeichnis an einen beliebigen Ort. Verschiebe dann die neu erzeugte Datei *BSB\_LAN\_defs\_filtered.h* in das Verzeichnis "BSB\_LAN" (falls du die Datei nicht bereits im Ordner "BSB\_LAN" erstellt hast).
- Wichtig: Die neu erzeugte Datei nun in "*BSB\_LAN\_defs.h*" umbenennen!

- **Gibt es bzgl. der zu verwendenden Pineinstellungen etwas zu beachten?**

Ja! Solltest du eine neuere Version als v0.44 auf dem Mega testen wollen, so achte darauf, dass du die zur jeweiligen Version zugehörige Datei *BSB\_LAN\_config.h.default* verwendest und entsprechend anpasst:

- Bei BSB-LAN-Versionen **vor v2.x** ist die Anpassung der Zeile `BSB_bus(19,18);` zwingend notwendig: Der DUE verwendet (im Gegensatz zum Mega) die HardwareSerial-Schnittstelle und andere RX-/TX-Pins als der Mega, was hier bereits voreingestellt ist. Bei Verwendung mit dem Mega muss die Zeile daher in `BSB_bus(68,69);` geändert werden!
- Bei BSB-LAN-Versionen **ab v2.x** ist in der Datei *BSB\_LAN\_config.h* eine automatische Erkennung der verwendeten Pins voreingestellt. Somit wird automatisch erkannt, ob ein Mega (= software serial) oder ein Due (= hardware serial) zum Einsatz kommt.

- **Warum gibt es überhaupt einen Umstieg auf den Due?**

Der Mega 2560 bot einfach nicht mehr genügend Speicher, um auch in Zukunft das stetig wachsende BSB-LAN zu beherbergen! ;)

- **Warum gibt es überhaupt einen neuen Adapter v3/v4?**

Das war nötig, da der bisherige Adapter v2 aus verschiedenen Gründen nicht kompatibel mit dem Due ist.

- **Kann ich den Adapter v2 an einem Due weiterverwenden?**

Nein! Der Grund dafür liegt primär darin, dass sowohl der Adapter v2 als auch der Due kein EEPROM aufweist, was für BSB-LAN jedoch notwendig ist.

Möchtest du also auch in Zukunft von den neuen Funktionen von BSB-LAN profitieren, musst du dir einen Adapter v3 besorgen oder selbst herstellen und ihn mit einem Arduino Due verwenden.

- **Kann ich den Adapter v2 zu einem Adapter v3/v4 'umbauen'?**

Nein! Der primäre Grund hierfür liegt (neben weiteren Gründen) auch wieder im fehlenden EEPROM des Due.

- **Kann ich den Adapter v3/v4 mit meinem bisherigen Mega 2560 weiterverwenden?**

Nein! Auch wenn es vielleicht nach gewissen Änderungen am Adapter v3/v4 möglich wäre, so würde es keinerlei Mehrwert gegenüber dem Adapter v2 bieten. Neue Funktionen von BSB-LAN würden aufgrund des mangelnden Speicher des Mega 2560 trotzdem nicht genutzt werden können. Wenn du also den neuen Adapter v3/v4 einsetzen möchtest, dann nur in Verbindung mit einem Arduino Due.

- **Warum ist auf der Platine v3/v4 ein EEPROM?**

Der Arduino Due weist kein EEPROM auf, was jedoch für BSB-LAN notwendig ist.

- **Kann ich das LAN-Shield bei einem Umstieg auf den Due weiterverwenden?**

Ja, das ist normalerweise problemlos möglich.

- **Kann ich mein bestehendes Gehäuse weiterverwenden?**

Jein. Der Due weist prinzipiell den gleichen Formfaktor auf wie der Mega 2560, insofern sollte das Gehäuse von den Abmessungen her passen. Allerdings musst du vermutlich dein Gehäuse etwas anpassen und einen Ausschnitt oder eine große Bohrung für den mittleren USB-Port des Due ('Programming Port') hinzufügen, damit du auch weiterhin bequem das entspr. USB-Kabel anschließen kannst.

