

I2C Workshop

I2C Workshop

LCDproc + Digispark + I2C



Vortrag zu den Labortagen 2021
Mark Hoffmann, B.Eng.



I2C Workshop

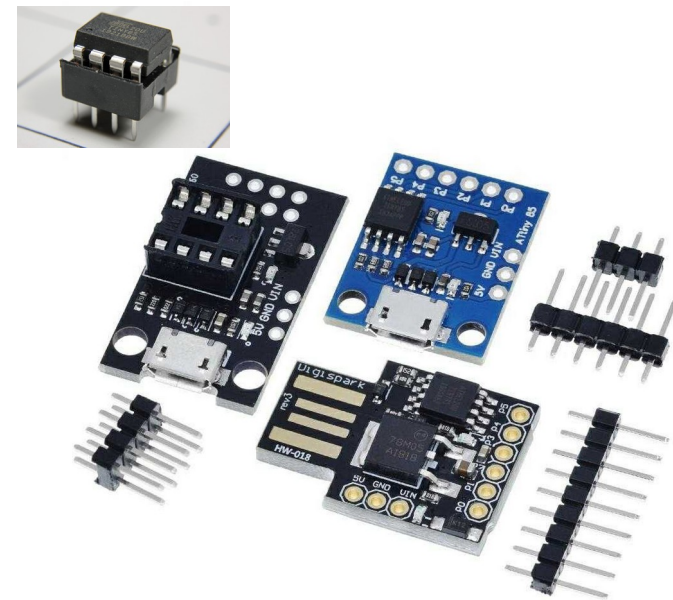
Inhalt

- Was ist ein **Digispark**?
- Was ist **I2C**?
- Dr. Till Harbaums „**I2C-Tiny-USB**“
- **Linux** Anbindung
- Statistiken mit **LCDproc**
- Auslesen von **I2C-Sensoren**
- Andere **Anwendungen von I2C**
- **Bad USB**
- **Demonstration**
- **Lötübung** - Digispark mit Display zum Mitnehmen

I2C Workshop

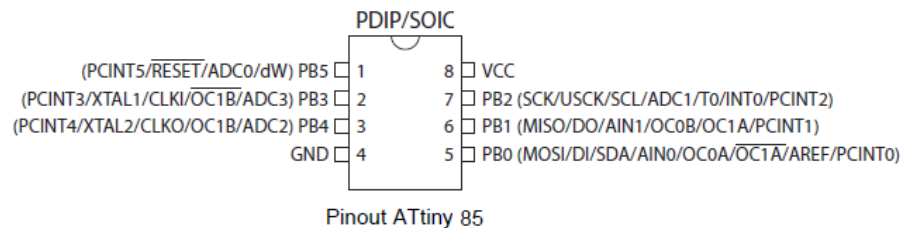
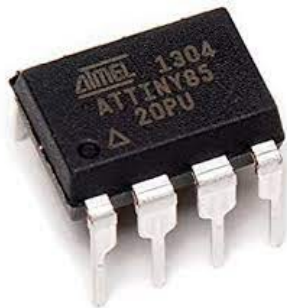
Digisparks

- Drei verschiedene Varianten
- ATtiny85 Mikrocontroller (an USB)
- Für I2C-Bus Konnektivität braucht es zusätzliche Pull-Up Widerstände



I2C Workshop

ATtiny85 - „Ein kleiner ATmega“



Spezifikation

Unterstützt die Arduino IDE 1.0+ (OSX/Win/Linux)

Stromversorgung über USB oder eine externe Quelle – 5V oder 7-35V (VIN Pin, automatische Auswahl)

On-board 500mA 5V Regler

Built-in USB 2.0 (und Serial-Debugging)

6 I/O Pins (**2 werden für USB benötigt wenn das Programm über USB kommuniziert, ansonsten kann man alle 6 verwenden**)

8kB Flash-Speicher (ca. 6kB nach Bootloader)

I2C und SPI

PWM auf 3 Pins (mehr über Software PWM möglich)

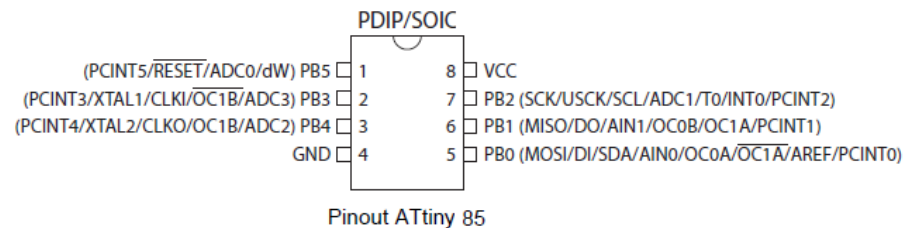
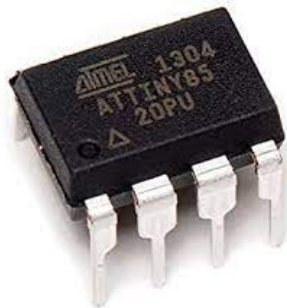
ADC auf 4 Pins

Power LED and Test/Status LED (auf Pin 0)

(Näheres unter: <https://is.gd/ovuhij>)

I2C Workshop

ATtiny85 - „Ein kleiner ATmega“



Pinbelegung

Pin 0 bis Pin 5

P0: I2C SDA, PWM

P1: PWM

P2: I2C SCK, Analog In

P3: Analog In, USB+

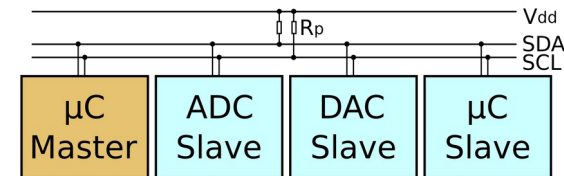
P4: PWM, Analog In, USB-

P5: Analog In (liefert 3V, wenn auf High)

I2C Workshop

I2C-Bus

- Inter-Integrated Circuit = IIC = I²C = I2C
- Philips / NXP Semiconductors: Entwickelt 1982 serieller Datenbus zur Steuerung bzw. Konfiguration von Chips in Fernsehern und CD-Playern
- Single Master / Multi Slave Aufbau / aber auch Multi Master möglich
- Eine Teilnehmeradresse pro Chip, z.B. „0x32“
- Suche nach Teilnehmern möglich per General Call Adresse „0x00“



I2C Workshop

I2C-Bus

- Vier Leitungen – SCL (Serial Clock – vom Master vorgegeben) und SDA (Serial Data), Vcc (Versorgungsspannung) und GND (Ground)
- 7-bit oder 10-bit Adressraum (Abhängig von der genutzten Hardware) - bis zu 112 oder 1136 Nodes an einem Bus
- Geschwindigkeit anpassbar
- Arduinos können 100kHz Standard Mode und 400kHz Fast Mode

Modus	Maximale Taktrate
Standard Mode	100 kHz
Fast Mode	400 kHz
Fast Mode Plus	1 MHz
High Speed Mode	3,4 MHz

I2C Workshop

I2C-Bus

- Zum Sparen von Lizenzkosten auch „Two Wire Interface (TWI)“ genannt
- NXP Patent 01.10.2006 abgelaufen
- Spezifikation von NXP → <https://is.gd/sumuwe> (2014)
- Auf der NXP Seite sind die entsprechenden Seiten nicht mehr verfügbar
- Empfehlung: Vortrag - „Auslesen eines Temperatursensors LM75 mit Raspberry Pi“ → <https://is.gd/doriwa>

I2C Workshop

Elektrische Eigenschaften

- Logikpegel orientieren sich an der Versorgungsspannung des Busses, d.h. 1,1V wird an einem mit 5V versorgten Bus als Low, in einem mit 1,2V versorgten Bus als High interpretiert

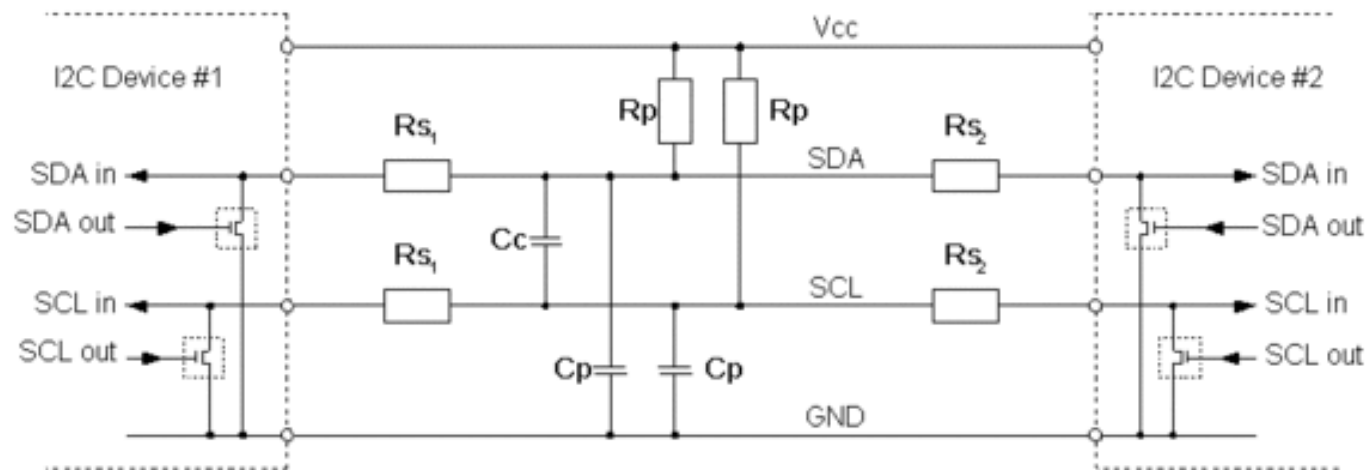
Die “Low Input Level Voltage”, r_L , liegt bei $0.3 \times V_{dd}$

Die “High Input Level Voltage”, r_H , liegt bei $0.7 \times V_{dd}$

- Das Entfernen von Steuerleitungen (Reduktion auf vier pro Knoten) reduziert die Pin-Anzahl und spart Kosten

I2C Workshop

Elektrischer Aufbau



VCC I2C-Versorgungsspannung

GND Masse

SDA I2C-Datenleitung

SCL Taktleitung

R_p Pull-Up-Widerstand / Terminierungswdstd.

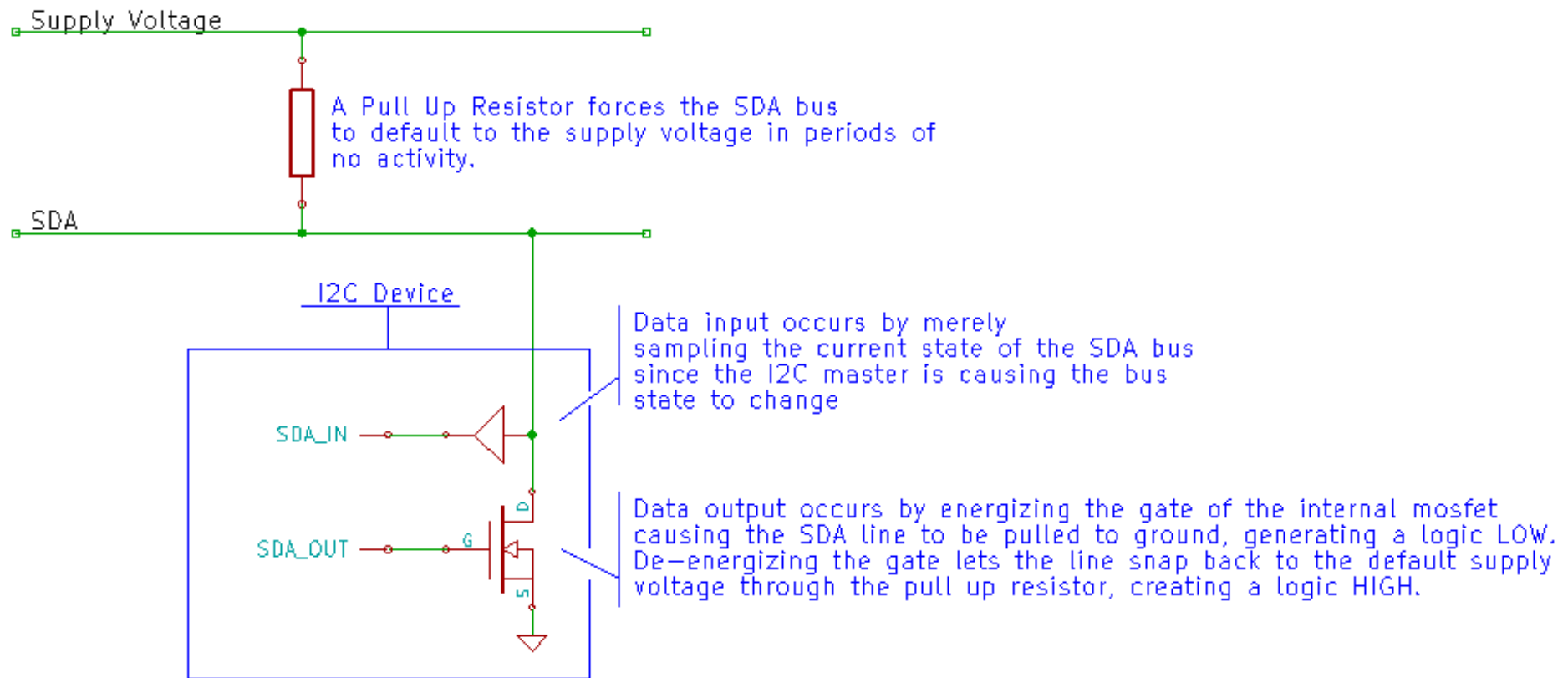
R_s Serienwiderstand

C_p Leitungskapazität

C_c Kapazität zwischen den Leitungen

I2C Workshop

I2C SDA und SCL Ein- / Ausgang (jeweils an jedem Busteilnehmer)



I2C Workshop

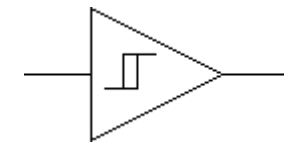
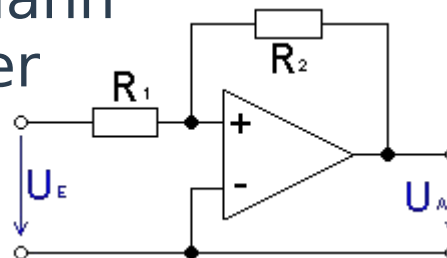
Schmitt Trigger Eingänge und Mosfet Ausgänge

- SCL und SDA Zustände müssen über Schmitt Trigger Eingänge eingelesen werden - mit Hysterese
- Es werden Feldeffekttransistoren eingesetzt in einer „Open-Drain“ Schaltung (das Bipolartransistor Äquivalent ist die „Open-Collector“ Schaltung)
- Open-Drain-Technologie bedeutet, daß Teilnehmer die Leitungen gegen 0 ziehen oder sich passiv verhalten können. Die Pull-Up / Terminierungswiderstände ziehen den Bus in Ruhe auf den High Pegel; die Transistoren dürfen den Bus nicht aktiv gegen High treiben

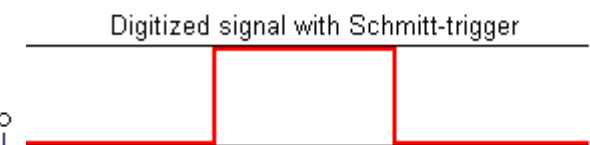
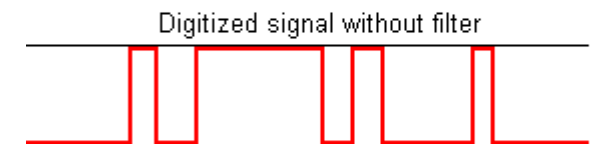
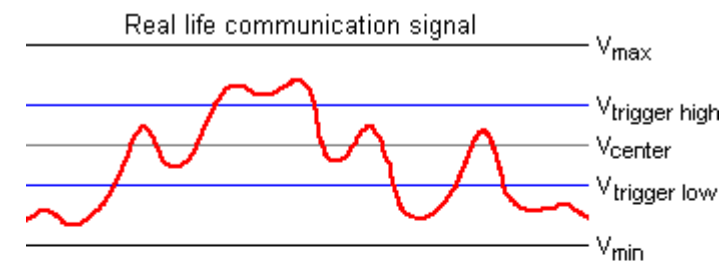
I2C Workshop

Schmitt Trigger Eingänge

- Schmitt Trigger dienen der „Noise“-Störungsreduktion; Bsp. Fahrstuhlknopf, der nicht bei Windhauch auslöst
- Bei langen Leitungen verfälscht das Tiefpassverhalten digitale Signale → es wird die ursprüngliche binäre Form wiederhergestellt
- Realisierung per Operationsverstärker → die Schaltung arbeitet dann als Sinus-Rechteck-Wandler vgl. <https://is.gd/yoxoto>



Schaltzeichen



<https://is.gd/zopisu>

I2C Workshop

Mosfet Ausgänge

In einer Open-Drain Schaltung lassen Feldeffekttransistoren im 1. Modus den Strom „verschwinden“; speisen keinen neuen ein; wird ein Strom „abgeleitet“, so ergibt dies ein logisches „Low“ - oder haben im 2. Modus keinen Stromfluss, geben einen hochohmigen „High“ Zustand aus

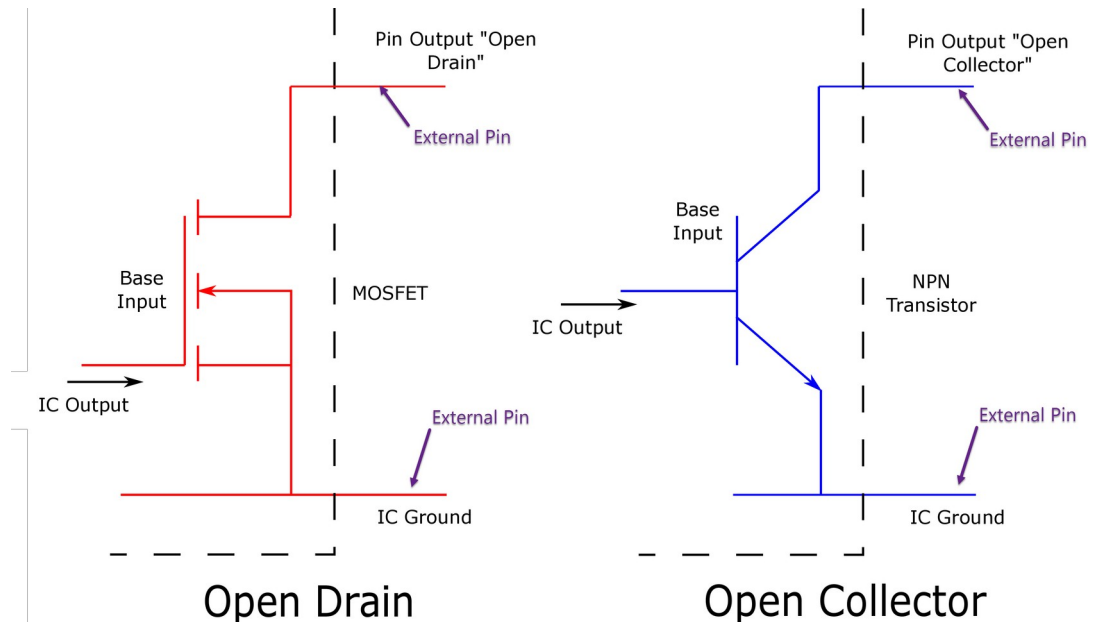
- Englische Beschreibung: „current sink (on an ic output pin)“; im Gegensatz zur „current source“
- Eine Stromquelle muss zu jeder Zeit verfügbar sein, um ein logisches „High“ zu erzeugen. Dies ist bei den SCL und SDA Leitungen gegeben; Anschluss über die Pull-Up Widerstände
- Im Gegensatz zu stromgesteuerten Bipolartransistoren sind Feldeffekttransistoren spannungsgesteuerte Schaltungselemente

I2C Workshop

Mosfet Ausgänge

Vgl. <https://is.gd/nelemo>

- Wenn Strom in an den Base (Gate) Anschluss eine Spannung angelegt wird, fließt Strom durch Source und Drain, der Transistor ist "an", wenn zu wenig davon (oder gar keine), leitet der Transistor nicht, er ist "aus"



I2C Workshop

Pull-Up Widerstände

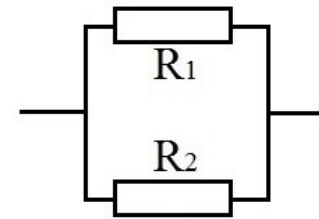
(auch genannt Terminierungswiderstände)

- Bei I2C Eigenbau Schaltungen betragen diese meist $4,7k\Omega$, $10k\Omega$ oder fehlen ganz (je kürzer der Bus, je weniger Widerstände benötigt)
- Ziehen SDA bzw. SCL jeweils auf den Standard-Pegel High, damit ein Schwingen bzw. Oszillieren nicht auftritt
- Teilweise haben Busteilnehmer wie Displays integrierte Pull-Up Widerstände, z.B. die hier eingesetzten Displays ... $6,84k\Omega$ (ausgemessen, sollte per Datenblatt ermittelt werden)
- Die internen Pull-Up Widerstände von Microcontrollern können per Software aktiviert werden (mit Problemen)
- Geräte nehmen keinen Schaden, wenn der Pull-Up Widerstand zu gering ist bzw. fehlt ... nur findet dann keine Kommunikation statt

I2C Workshop

Pull-Up Widerstände

- Pull-Up Widerstände für jeden Knoten werden addiert als parallele Widerstände zum Gesamt Widerstand
- Gesamter Bus-weiter Pull-Up Widerstand sollte nicht weniger als 2,2kΩ betragen
- TI „I2C Bus Pullup Resistor Calculation“
Application Note: <https://is.gd/tuhobu>



$$R_{Ges} = \frac{R_1 \cdot R_2}{R_1 + R_2}$$

Addieren
paralleler
Widerstände

I2C Workshop

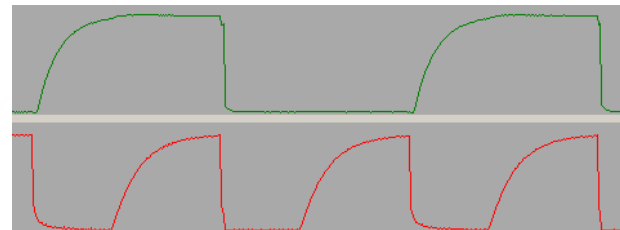
Kapazität

- Das Maximum aller Kapazitäten auf einem I2C Bus ist auf 400pF begrenzt gemäß dem I2C-Standard
- Kapazitäten werden in einer Parallelschaltung einfach vom Wert her addiert
- Während Mosfets bzw. Open-Drain-Treiber zumeist 10 mA und mehr verarbeiten und das Signal gegen 0 ziehen können, sind die Pull-Up Widerstände für die Rückführung des Signals auf den High-Pegel verantwortlich

I2C Workshop

Kapazität

- Der gesamte Pull-Up Widerstand liegt meist zwischen 2,2 k Ω und 10 k Ω , was zu Strömen von 1 mA und weniger führt
- Je größer der Widerstand, desto länger dauert es, bis der Kondensator geladen ist, desto länger dauert es, dass das Signal auf High-Pegel ansteigt
- Dieses ist die Ursache für das sägezahnförmige I2C-Signal. Jeder Zahn zeigt das Lade- bzw. Entladeverhalten der Leitung



SDA (oben) und SCL (unten) mit $R_p = 10 \text{ k}\Omega$ und $C_p = 300 \text{ pF}$. Der SCL-Takt beträgt 100 kHz

I2C Workshop

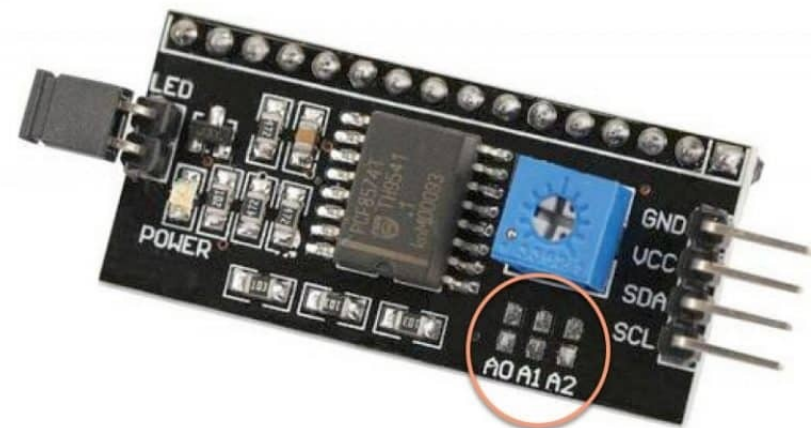
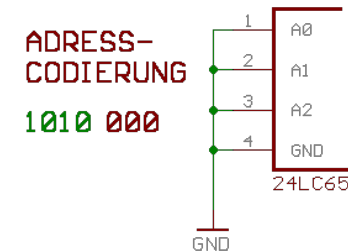
Logische Eigenschaften I2C

- Die Busteilnehmeradressen sind 7 Bit lang, gefolgt von einem Richtungsbit, MSB-first
- Jede Adresse nur ein mal pro Bus möglich (!)
- Eine 7 Bit lange Adresse erlaubt theoretisch bis zu 128 I2C Busteilnehmer
- Manche Adressen sind reserviert für spezielle Zwecke
- Also sind nur 112 Adressen für Busteilnehmer verfügbar, 0x08 bis 0x7F
- General Call Adresse mit dem wert 0x00, damit lassen sich alle Knoten ansprechen (und suchen)

I2C Workshop

Logische Eigenschaften I2C

- Adressen meist bei Slave Knoten per Lötbrücken einstellbar, Grundadresse, z.B. 0x27 + 0,1,2,3,4,5,6,7
→ man kann 8 dieser Knoten an dem gleichen I2C-Bus betreiben
- 10 Bit Adressen als Erweiterung möglich, kompatibel - jedoch nicht mit Arduinos / ATmegas und Linux Paket i2c-tools, vgl. <https://is.gd/kacubi>
- Vergabe früher durch NXP
- Liste aller vergebenen Adressen mit Namen der Bauteile: <https://is.gd/inifub>

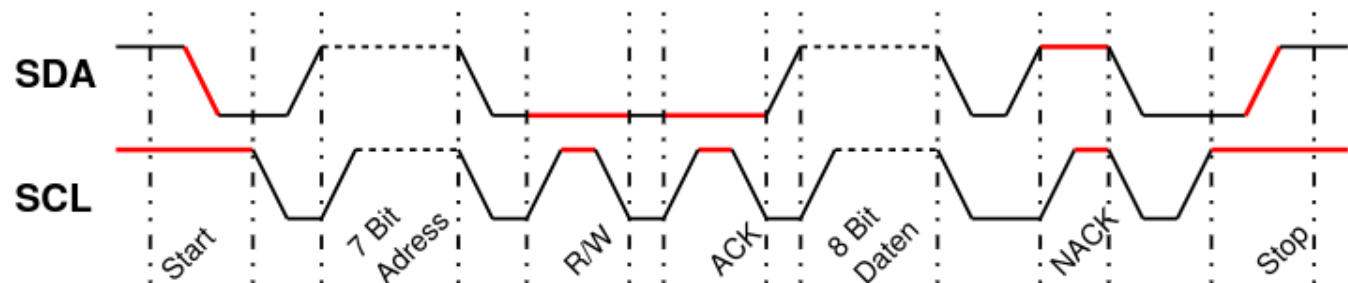


I2C Workshop

Logische Eigenschaften I2C

Zur Übertragung eines Bytes sind folgende Schritte zu durchlaufen (Auslesen eines Sensors)

- Zuerst muss der Bus im Ruhe Modus sein (SCL und SDA beide auf High-Pegel)
- Eine Start-Bedingung wird vom I2C-Master ausgelöst
- Eine Adresse eines Sensors wird vom I2C Master gesendet (7 Bit, z.B. 0x27 → 0100111b)
- Ein R/W (Lese-/Schreib) Merker wird vom I2C Master gesendet (gibt an, ob der Master Lesen oder Schreiben will - hier Schreiben)
- Ein ACK/NACK (hoffentlich ein ACK) wird vom I2C Slave gesendet; ohne Antwort zählt's als NACK
- 8 Bit bzw. 1 Byte an Daten werden übertragen

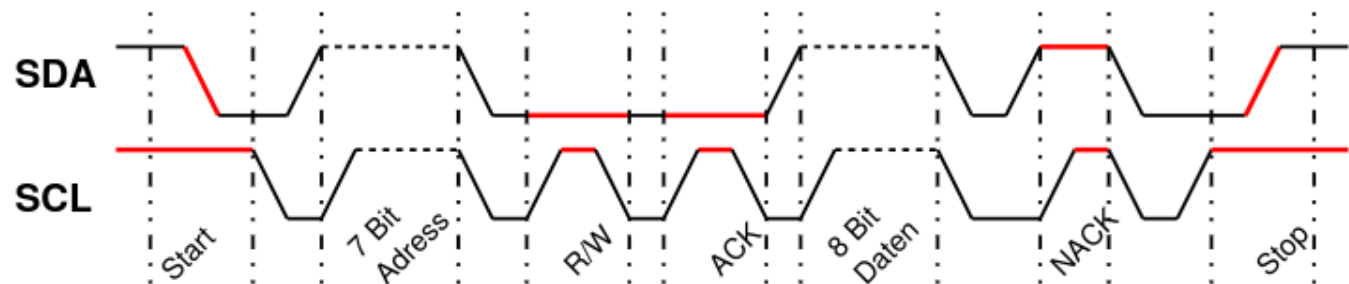


I2C Workshop

Logische Eigenschaften I2C

Zur Übertragung eines Bytes sind folgende Schritte zu durchlaufen (Auslesen eines Sensors)

- Ein ACK/NACK (hoffentlich ein ACK) wird vom I2C Slave gesendet; ohne Antwort zählt's als NACK
- 8 Bit bzw. 1 Byte an Daten werden übertragen (Übertragung mehrerer Bytes: <https://is.gd/nilluke>)
- Ein ACK/NACK (hoffentlich ein NACK) wird vom I2C Master gesendet (NACK heisst entweder, dass alle Daten übertragen wurden, oder ein Fehler aufgetreten ist)
- Eine Stop-Bedingung wird vom I2C Master ausgelöst
- Der Bus wechselt wieder in den Ruhe Modus

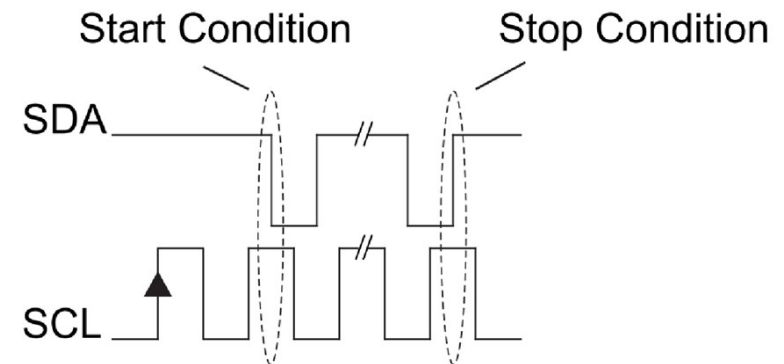


I2C Workshop

Logische Eigenschaften I2C

Start / Stop Condition

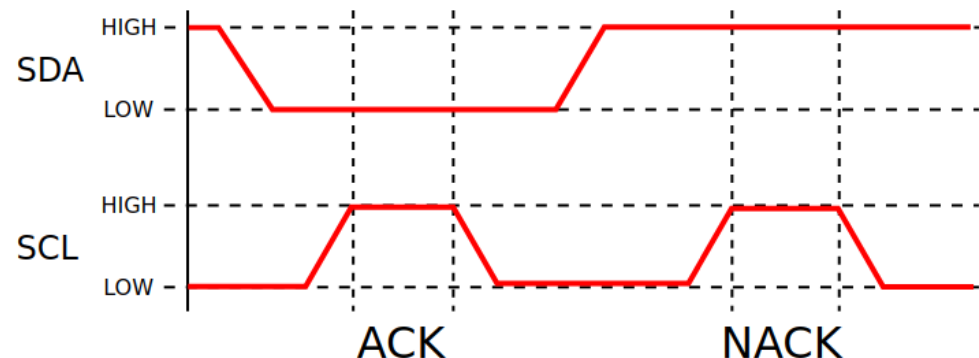
- Ein High-zu-Low Übergang auf der SDA-Leitung während SCL auf High liegt beschreibt eine Start-Condition
- Eine Low-zu-High Übergang auf der SDA-Leitung während SCL auf High ist beschreibt eine Stop-Condition



I2C Workshop

Logische Eigenschaften I2C ACK / NACK

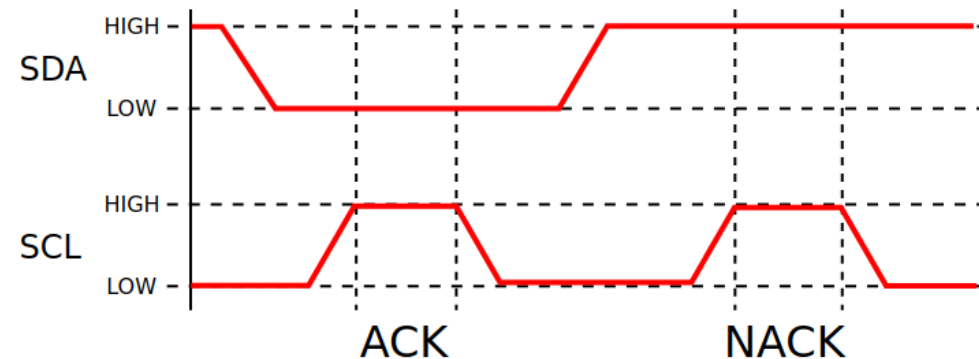
- Jedes gesendete Byte muss von dem Empfänger bestätigt werden mit einem einzelnen Bit: 0 für ACK und 1 für NACK (Acknowledgment / Not-Acknowledgement)
- Eine Dateneinheit besteht aus 8 Datenbits, welche entweder als Wert oder als Adresse interpretiert werden) und einem ACK-Bit



I2C Workshop

Logische Eigenschaften I2C ACK / NACK

- Das ACK wird vom Slave durch einen Low-Pegel auf der Datenleitung SDA während der neunten Takt-High-Phase (welche nach wie vor vom Master generiert wird)
- NACK wird durch einen High-Pegel signalisiert. Der Slave muss den Low-Pegel an der Datenleitung anlegen, bevor SCL auf High geht, andernfalls lesen weitere eventuelle Teilnehmer ein Start-Signal



I2C Workshop

Logische Eigenschaften I2C

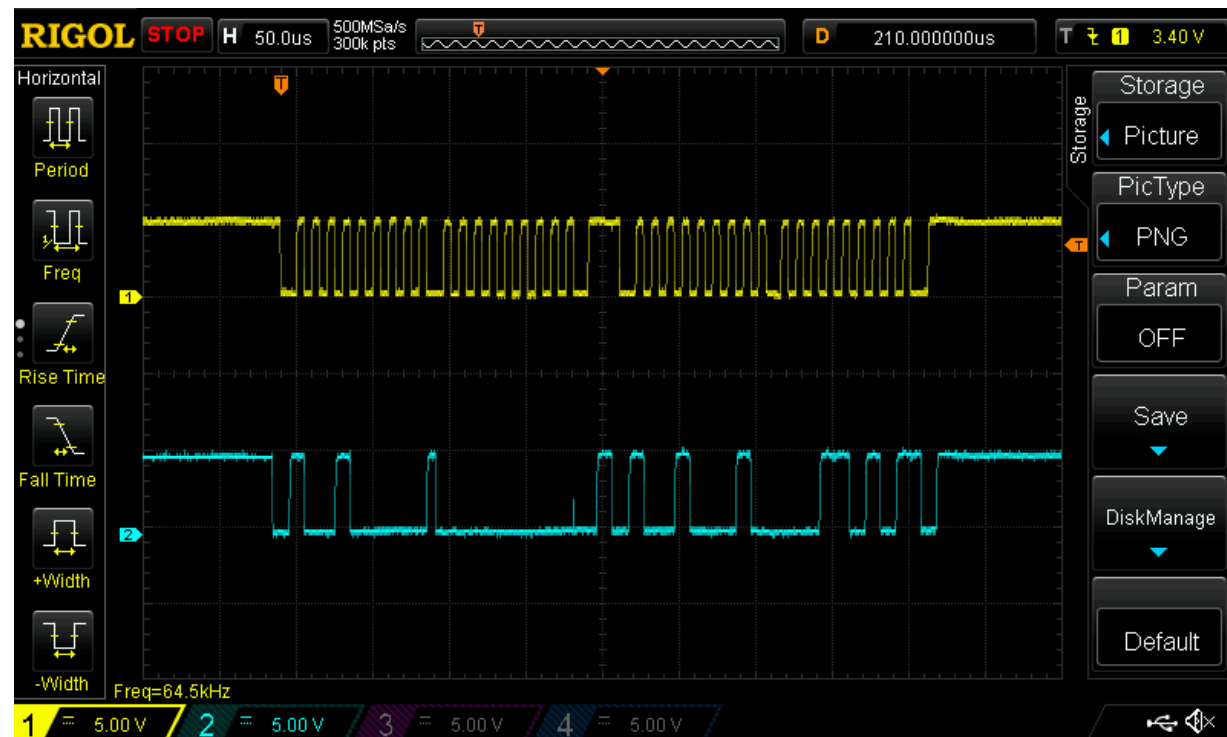
ACK / NACK

- Ein NACK nach einer Adresse wird gesendet, wenn keine Slave auf diese Adresse geantwortet hat (sendet wer?)
- Ein NACK nach Schreiben bedeutet, dass der Slave entweder den Befehl nicht verstanden hat oder nicht mehr Daten verarbeitet werden können
- Ein NACK während eines Lesevorgangs bedeutet, dass der Master keine weiteren Daten empfangen will vom SLave

I2C Workshop

Logische Eigenschaften I2C

- I2C Signal Reverse Engineering mit Oszilloskop:
“Wer es genau wissen will“ ... → <https://is.gd/quwocu>



I2C Workshop

Logische Eigenschaften I2C

Clock Stretching

- Es kann sein, dass der Slave, der angesprochen werden soll, gerade nicht in der Lage ist zu antworten
- In solchen Fällen ist es dem Slave erlaubt die SCL-Leitung auf LOW-Level zu halten bis er reagieren kann
- Beispiele sind EEPROMs mit langsamen Zugriffszeiten oder softwarebasierenden Slaves; sonst eher unüblich heute
- Kann Synchronisierungsprobleme lösen
- Problem: Es wird der gesamte Bus ausgebremst

I2C Workshop

Bus Arbitrierung

- Nur problematisch bei Multi Master Systemen
- Bei Multi Master Systemen verhält sich nur einer wie ein Master, die anderen wie Slaves

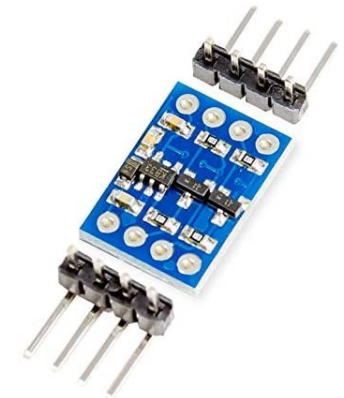
Kommunikationsfehler

- Werden die Logikpegel nicht schnell genug erreicht, bricht die Kommunikation zusammen

I2C Workshop

Pegelwandler

- Hat man Busteilnehmer mit unterschiedlicher Spannungsspezifikation empfehlen sich Pegelwandler
- Diese setzen bei den üblichen I2C Komponenten die Pegel 3,3V zu 5V um

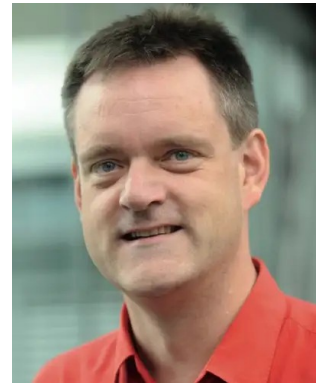


I2C 3,3V ↔ 5V
Pegelwandler

I2C Workshop

Dr. Till Harbaum

Mitarbeiter am
Karlsruher Institut für Technologie (KIT) 
Institut für Telematik - Lehrstuhl Prof. Zitterbart



Dr. Till Harbaum

Frage zur Verwendung der Software für diesen Workshop – „Ok“
→ <http://www.harbaum.org/till>

I2C Workshop

Dr. Till Harbaum

i2c-tiny-usb (Digispark Firmware)

- Bietet neuen „Out-of-the-box“ I2C-Bus unter Linux
- Support ab Linux Kernel 2.6 bereits integriert
→ <https://is.gd/lebusi> (Kernel Sourcecode)
- Support für Windows
→ <https://is.gd/evuwul> (Installation) und <https://is.gd/uhiolol> (Code für DS1621 und PCF8574)
- I2c-tiny-usb macht bei Raspberry Pi keinen Sinn, da dort schon von SoC angeboten → vgl. <https://is.gd/usowic>
- Quelltext für Firmware frei (und vorkompiliert) verfügbar
→ <https://is.gd/ezohup> bzw. <https://is.gd/zaloxa> (Till Harbaums GitHub Account)

I2C Workshop

Digispark Firmware Einspielen

- Programmieren per Micronucleus Flashertool (ohne Zusatz Hardware wie SPI Programmer bei Kauf über z.B. eBay! Anschluss nur über USB)
- Tool starten, Einstecken ...
- **... Löschen**

```
Datei  Bearbeiten  Darstellung  Suchen  Terminal  Hilfe
root@x220:/home/mongoq# micronucleus --erase-only
> Please plug in the device ...
> Device is found!
connecting: 50% complete
> Device has firmware version 1.6
> Available space for user applications: 6012 bytes
> Suggested sleep time between sending pages: 8ms
> Whole page count: 94  page size: 64
connecting: 50% complete
> Erasing the memory ...
erasing: 100% complete
>> Micronucleus done. Thank you!
root@x220:/home/mongoq#
```

Löschen: „micronucleus --erase-only“

I2C Workshop

Digispark Firmware Einspielen

- ... Flashen

```
Datei  Bearbeiten  Darstellung  Suchen  Terminal  Hilfe
root@x220:/home/mongoq# micronucleus --run --type intel-hex main.hex
> Please plug in the device ...
> Device is found!
connecting: 33% complete
> Device has firmware version 1.6
> Available space for user applications: 6012 bytes
> Suggested sleep time between sending pages: 8ms
> Whole page count: 94  page size: 64
> Erase function sleep duration: 752ms
parsing: 50% complete
> Erasing the memory ...
erasing: 66% complete
> Starting to upload ...
writing: 83% complete
> Starting the user app ...
running: 100% complete
>> Micronucleus done. Thank you!
root@x220:/home/mongoq#
```

Flashen: „micronucleus --run --type intel-hex main.hex“

I2C Workshop

I2C unter Linux

- Nachdem der präparierte Digispark eingesteckt wurde sieht die Ausgabe zu „lsusb“ wie folgt aus:

```
Datei  Bearbeiten  Darstellung  Suchen  Terminal  Hilfe
Bus 002 Device 019: ID 0bdb:1911 Ericsson Business Mobile Networks BV F5521gw
Bus 002 Device 002: ID 8087:0024 Intel Corp. Integrated Rate Matching Hub
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 004 Device 001: ID 1d6b:0003 Linux Foundation 3.0 root hub
Bus 003 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 004: ID 04f2:b217 Chicony Electronics Co., Ltd Lenovo Integrated Camera (0.3MP)
Bus 001 Device 003: ID 0a5c:217f Broadcom Corp. BCM2045B (BDC-2.1)
Bus 001 Device 009: ID 0403:c631 Future Technology Devices International, Ltd i2c-tiny-usb interface
Bus 001 Device 002: ID 8087:0024 Intel Corp. Integrated Rate Matching Hub
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
mongoq@x220:~$
```

„lsusb“

I2C Workshop

I2C unter Linux


- Sowie zu „dmesg“ so:

```
Datei  Bearbeiten  Darstellung  Suchen  Terminal  Hilfe
[103165.896322] usb 1-1.2: new low-speed USB device number 9 using ehci-pci
[103166.025805] usb 1-1.2: New USB device found, idVendor=0403, idProduct=c631, bcdDevice= 2.01
[103166.025815] usb 1-1.2: New USB device strings: Mfr=1, Product=2, SerialNumber=3
[103166.025818] usb 1-1.2: Product: i2c-tiny-usb
[103166.025820] usb 1-1.2: Manufacturer: Till Harbaum
[103166.025822] usb 1-1.2: SerialNumber: 0165
[103166.079359] i2c-tiny-usb 1-1.2:1.0: version 2.01 found at bus 001 address 009
[103166.082742] i2c i2c-10: connected i2c-tiny-usb device
[103166.082803] usbcore: registered new interface driver i2c-tiny-usb
mongoq@x220:~$
```

„dmesg“

I2C Workshop

I2C unter Linux

- Nun können die Debian Pakete „i2c-tools“, „libftdi1“ und „python-smbus“ installiert werden (Trick für Fehlen von python-smbus in aktuellen Repositories → <https://is.gd/zuyato> als Bezugsquelle)
- Kernelmodul laden per „sudo modprobe i2c-dev“ 
- Mit „i2cdetect -l“ kann man die I2C-Bus Nummer ermitteln, in einem normalen PC gibt es bereits etwa 10
- I2C Linux Programmen muss dieser Wert als Parameter / Config-Datei mitgegeben werden

I2C Workshop

I2C unter Linux

- Für Arch Linux sind das respektive die Pakete „i2c-tools“, „lm_sensors“ und „libftdi“- „python-smbus“ ist in diesen Paketen bereits enthalten



I2C Workshop

I2C unter Linux

- Mit „i2cdetect -l“ kann man die I2C-Bus Nummer des Digisparks „i2c-tiny-usb“ Buses ermitteln. Hier ist die Bus Interface Nummer i2c-10 (also 10).

```
Datei  Bearbeiten  Darstellung  Suchen  Terminal  Hilfe
mongoq@x220:~$ i2cdetect -l
i2c-3    unknown      i915 gmbus panel          N/A
i2c-10   unknown      i2c-tiny-usb at bus 001 device 006  N/A
i2c-1    unknown      i915 gmbus ssc            N/A
i2c-8    unknown      AUX C/DP C                N/A
i2c-6    unknown      i915 gmbus dpd            N/A
i2c-4    unknown      i915 gmbus dpc            N/A
i2c-2    unknown      i915 gmbus vga            N/A
i2c-0    unknown      SMBus I801 adapter at efa0    N/A
i2c-9    unknown      AUX D/DP D                N/A
i2c-7    unknown      AUX B/DP B                N/A
i2c-5    unknown      i915 gmbus dpb            N/A
mongoq@x220:~$
```

„i2cdetect -l“

I2C Workshop

I2C unter Linux

- Nun das kann man zum Testen des I2C Display anschließen
- Per „i2cdetect -y 10“ findet sich das Display mit der Adresse 0x27
- → Das Display kann angesprochen werden

```
Datei  Bearbeiten  Darstellung  Suchen  Terminal  Hilfe
root@x220:/# i2cdetect -y 10
      0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:                -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- 27 -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- -- -- -- -- -- --
root@x220:/#
```

„i2cdetect -y 10“

I2C Workshop

SMBus System Management Bus



- Der Bus wurde 1995 von Intel und Duracell definiert
- Abwärtskompatibel zu I2C
(Gegenüberstellung I2C ↔ SMBus: <https://is.gd/ucebug>)
- Die Bitrate beim SMBus beträgt zwischen 10 kHz und 100 kHz
- Einstaz bei Mainboards: Herstellerinformationen, Ausgabe der Modell- oder Seriennummer, u.a. Status des Energiesparmodus, Fehlermeldungen, Steuerbefehle
- Zugriff nicht für den Endverbraucher „gedacht“
CPU-Temperatur und Akku-Status Auslesen ist jedoch unkritisch

I2C Workshop

SMBus System Management Bus



- Mit Hilfe des Pakets „lm-sensors“ kann man per “sensors-detect” nach SMBus und I2C-Teilnehmern suchen
- Findet leider nicht beliebige Busteilnehmer (!), nur bekannte Sensoren

```
Datei  Bearbeiten  Darstellung  Suchen  Terminal  Hilfe
root@x220:/home/mongoq# sensors-detect
# sensors-detect version 3.6.0
# System: LENOVO 429153G [ThinkPad X220] (laptop)
# Kernel: 5.11.0-38-generic x86_64
# Processor: Intel(R) Core(TM) i5-2450M CPU @ 2.50GHz (6/42/7)

This program will help you determine which kernel modules you need
to load to use lm_sensors most effectively. It is generally safe
and recommended to accept the default answers to all questions,
unless you know what you're doing.

Some south bridges, CPUs or memory controllers contain embedded sensors.
Do you want to scan for them? This is totally safe. (YES/no):
```

I2C Workshop

SMBus System Management Bus



- Ein einfacher Befehl zum Auslesen der CPU Temperatur: “sensors”

```
Datei Bearbeiten Darstellung Suchen Terminal Hilfe
root@x220:/home/mongoq# sensors
thinkpad-isa-0000
Adapter: ISA adapter
fan1:          3644 RPM

BAT0-acpi-0
Adapter: ACPI interface
in0:           10.99 V

coretemp-isa-0000
Adapter: ISA adapter
Package id 0:  +60.0°C (high = +86.0°C, crit = +100.0°C)
Core 0:         +55.0°C (high = +86.0°C, crit = +100.0°C)
Core 1:         +60.0°C (high = +86.0°C, crit = +100.0°C)

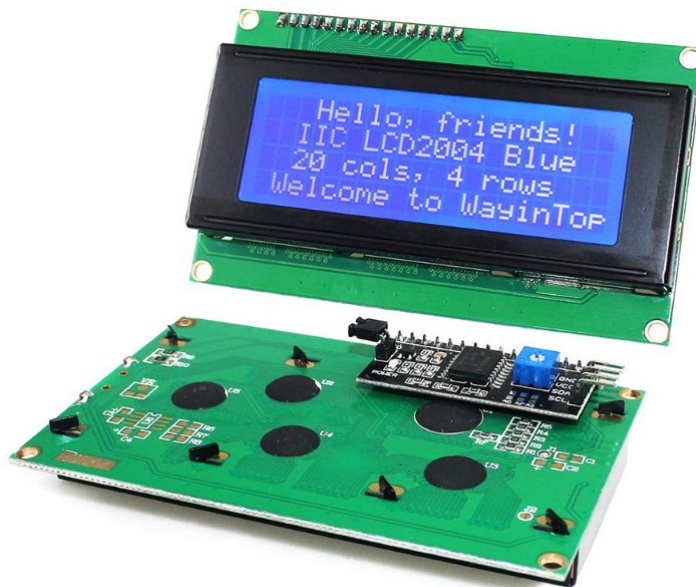
BAT1-acpi-0
Adapter: ACPI interface
in0:           11.98 V

acpitz-acpi-0
Adapter: ACPI interface
temp1:         +59.0°C (crit = +99.0°C)

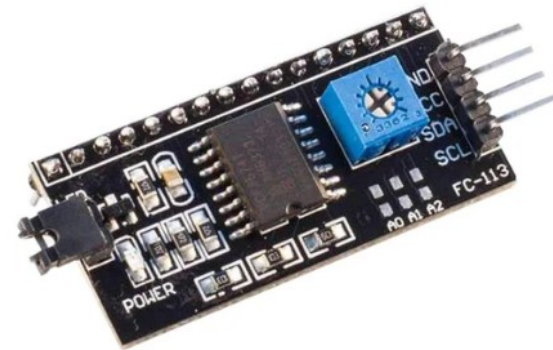
root@x220:/home/mongoq#
```

I2C Workshop

20x04 Display



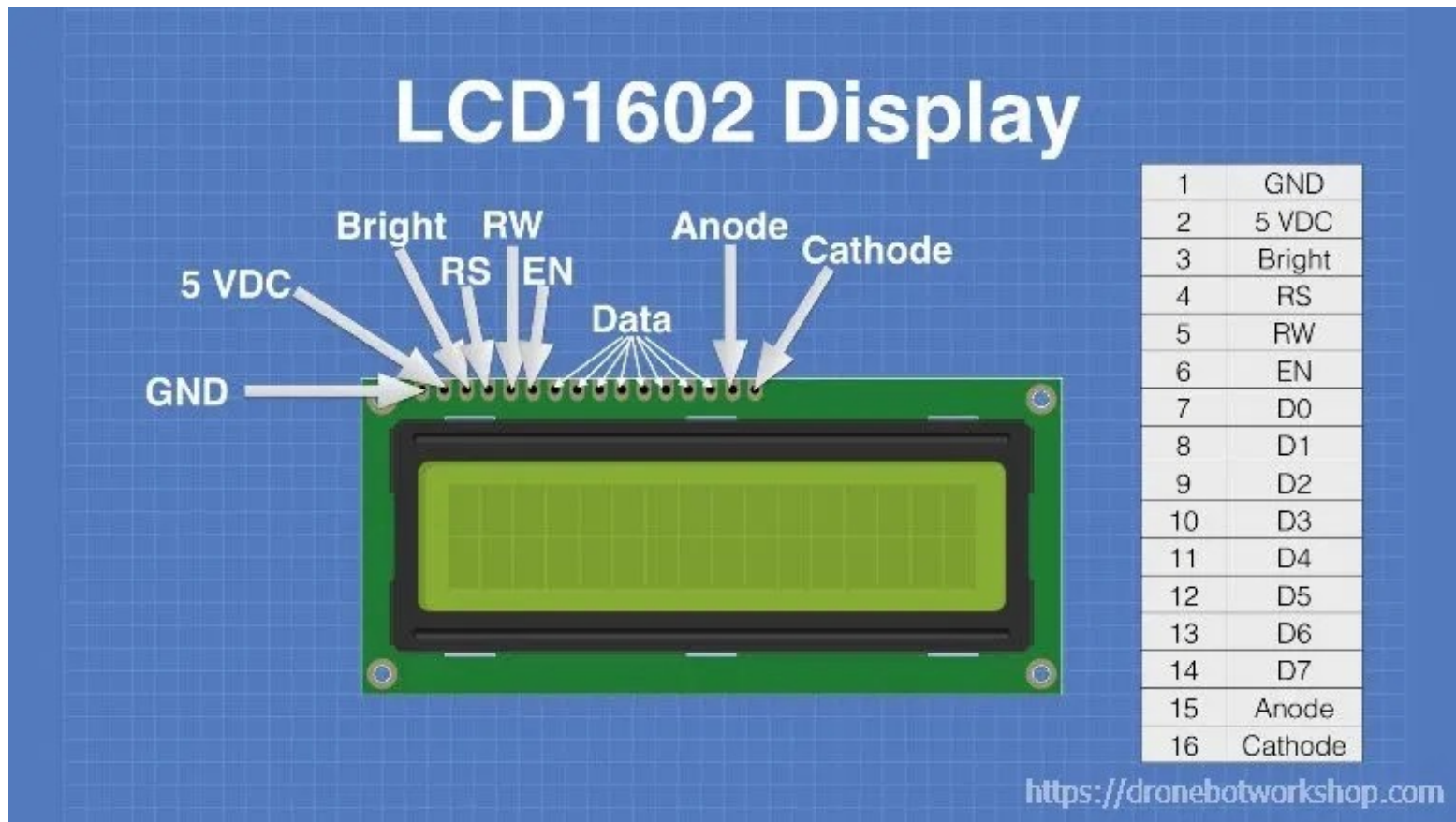
20 Zeichen,
4 Zeilen Display



I2C auf HD44780
PCF8574T Adapter
→ 8-Bit Port
Expander

I2C Workshop

20x04 Display - HD44780



Hitachi HD44780: Standard-IC für LCD Ansteuerung

I2C Workshop

20x04 Display - HD44780

1. **GND** – This is the Ground pin. On some modules it is labeled VSS.
2. **5 VDC** – This is the 5 volt power connection. On some modules it is labeled VDD.
3. **Brightness** – This is the input for the brightness control voltage, which varies between 0 and 5 volts to control the display brightness. On some modules this pin is labeled V0.
4. **RS** – This is the Register Select pin. It controls whether the input data is meant to be displayed on the LCD or used as control characters.
5. **RW** – Puts the LCD in either Read or Write mode. In most cases you'll be using Read mode so this pin can be tied permanently to ground.
6. **EN** – The Enable pin. When High it reads the data applied to the data pins. When low it executes the commands or displays the data.
7. **D0** – Data input 0.
8. **D1** – Data input 1.
9. **D2** – Data input 2.
10. **D3** – Data input 3.
11. **D4** – Data input 4.
12. **D5** – Data input 5.
13. **D6** – Data input 6.
14. **D7** – Data input 7.
15. **A** – The Anode (positive voltage) connection to the backlight LED.
16. **K** – The Cathode (ground or negative voltage) connection to the backlight LCD.

HD44780 Pinbelegung

I2C Workshop

<http://www.lcdproc.org/>



LCDproc

Multi-platform LCD display driver

[HOME](#) - [DOWNLOAD](#) - [SCREENSHOTS](#) - [FAQ](#) - [DOCUMENTS](#) - [FORUMS](#)
[HARDWARE](#) - [CLIENTS](#) - [FUN USES](#) - [DEVELOPMENT](#)
[MAIL](#) - [CREDITS](#) - [HISTORY](#)

Welcome to the Official LCDproc homepage!

LCDproc is a piece of open source software that displays real-time system information from your Linux/*BSD box on a LCD. The server supports several serial and USB devices from Matrix Orbital and CrystalFontz as well as some devices connected to the LPT port: HD44780, T6963, SED1520 and SED1330. Various clients are available that display things like CPU load, system load, memory usage, uptime, and a lot more.



Menu
Home
Screenshots
Hardware
F.A.Q.
Documents
Forums
Download
Clients
Fun Uses!
Development
Mail
Credits
History

LCDproc v0.5.6 released

2012-11-04

LCDproc v0.5.6, the new stable version of LCDproc is ready and available. See the [download section](#) for details.

lcdproc.org domain is back again

2012-06-28

The LCDproc domain names (lcdproc.org, lcdproc.net and lcdproc.com) are pointing to the right page again. This was made possible by a generous donation by [CrystalFontz America, Inc.](#) Thank you very much!



lcdproc.org domain currently unavailable

2012-06-04

I2C Workshop

LCDproc Daemon (LCDd)

- Linux Daemon lauscht auf Port 13666
- Setzt Anfragen auf Displayanzeige um
- Per „lcdproc“ Statistiken vom gleichen Rechner
- Per Netcat anzusprechen per Protokoll: <https://is.gd/inegeg>
- Python Anbindung: <https://is.gd/sojede>
- Zum Debuggen kann man LCDd so starten:
“sudo LCDd -f -r 5 -s 0“



LCDproc Server
ohne Clients

I2C Workshop

LCDproc Daemon (LCDd) Benutzung

- Eigener Text bei eigenem Rechner

Man schicke folgenden Text an den eigenen Rechner per
„nc localhost 13666 < hello_labortage.txt“

hello

screen_add Screen01

widget_add Screen01 Number string

widget_add Screen01 Name string

widget_set Screen01 Number 1 1 "Hello Labortage \'21"

widget_set Screen01 Name 1 2 "... LCDproc Demo ..."

- Über Netzwerk anzusprechen („freizuschalten“) per LCDd -a \$ipdesservers;
dann „nc \$ipdesservers 13666 < hello_labortage.txt“ vom Client

I2C Workshop

LCDproc Daemon (LCDd) Benutzung

- „Hello World“ mit Netcat

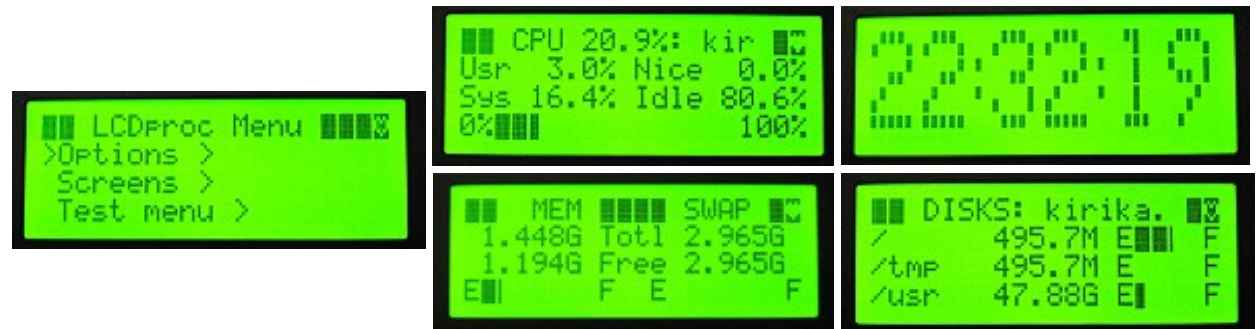


Netcat Einsatz

I2C Workshop

LCDproc Statistiken

- Das beiliegende Programm „lcdproc“ kann verschiedene Betriebssystem-Statistiken anzeigen
- Ausgaben u.a CPU Auslastung / RAM / HDDs / LAN / Uhrzeit / Uptime
- Es ist möglich, ein Menü anzusprechen → Interaktivität u.a. in Kombination mit LIRC, vgl. LCDproc User's Guide (<https://is.gd/qufaper>)



Output LCDproc eigener lcdproc Client

I2C Workshop

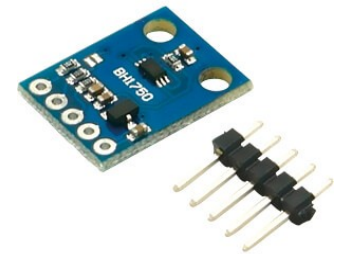
LCDproc Display Treiber → hd44780.so

- Das Standard Ubuntu Paket lcdproc erwartet andere Pinbelegung des Displays
→ vgl. <https://is.gd/soxen> bzw. → <https://is.gd/ehojow>
- Die fertige passende (64 Bit x86 Architektur) hd44780.so Datei wird auf der Webseite bereitgestellt → <https://wt.mongoq.de/lt/files/lcdproc/>
- Per „file hd44780.so“ ergibt sich, dass sie für „herkömmliche PCs“ ausführbar ist
- Kopie der Datei nach /usr/lib/x86_64-linux-gnu/lcdproc/hd44780.so (oder Anpassen des „DriverPath=“ Eintrags in der Config-Datei /etc/LCDd.conf)
- Ein „md5sum hd44780.so“ führt zu „b9313acca34b40b01870bc265f910ad6“ (bei mir)
- Es ist auch möglich lcdproc selber zu kompilieren → <https://is.gd/celiju> bzw. <https://is.gd/joquge> (Dank an CyReVolt!)

I2C Workshop

Ansprechen eines Helligkeitssensors

- Die Umgebungshelligkeit kann von sensorinternen Registern (immer das Datenblatt lesen - <https://is.gd/bekixi>) auslesen
- Dafür gibt es die Tools „i2cget, i2cset, i2cdump und i2ctransfer“
- Beispiel: „i2cget 10 0x68 0x20“ → Lese das 8-Bit Register 0x20 an 7-Bit Adresse 0x68 auf I2C-Bus 10 aus
- „i2cset 10 0x68 0x20 0x23“ → Schreibe in das Register 0x20 den Wert 0x23 auf I2C-Bus 10
- Shell Skript dazu und Lösung für Python mit SMBus findet sich unter <https://is.gd/irucut>
- Das Debian Paket Python SMBus wird benötigt
- Es werden Pull-Up Widerstände benötigt

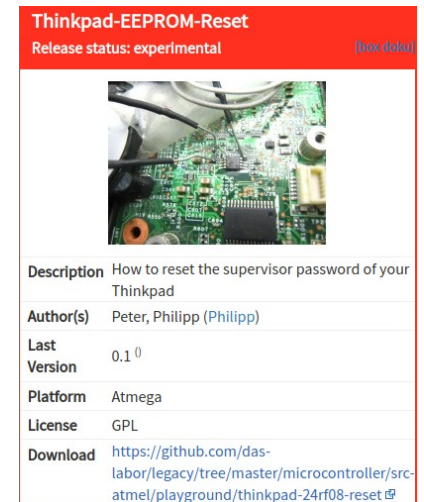


BH1750
Helligkeitssensor

I2C Workshop

Löschen von Thinkpad Bios Passwords per I2C

- Das Bios Passwort liegt bei Thinkpads bei bestimmten Modellen in einem I2C-EEPROM (24RF08)
- Lösung zum Zurückstellen / Resetten eines unbekannten Werts von Laboranten mit Hilfe eines ATmegas möglich gemacht
→ <https://is.gd/helopu>
- Wird Linux zum Überschreiben des EEPROMs genutzt, wird das Tool „eeprog“ empfohlen
→ <https://is.gd/aqudik>
- Passwörter werden teilweise auch in TPM-Chips gespeichert, da ist kein Ansprechen per I2C möglich






Laboranten Lösung
(Tixiv, Zaolin)

I2C Workshop

I2C per VGA- / DVI- / HDMI-Anschluss

- Pull-Up Widerstände werden benötigt für SCL und SDA (z.B. 4,7kΩ)
- Anbindung mit i2c-tools unter Linux „wie üblich“ z.B. mit „i2cdetect -l“
- Eigentlich genutzt zum Auslesen der Monitor Eigenschaften „EDID“ “Extended Display Identification Data” → <https://is.gd/igaxox>
- Vgl. alleinige Nutzung des VGA-Steckers - <https://is.gd/tucuga>

VGA:			DVI:			HDMI (TYPE A):		
								
PIN	DESCRIPTION		PIN	DESCRIPTION		PIN	DESCRIPTION	
5	Ground	●	6	Clock	●	15	Clock	●
9*	+5VDC	●	7	Data	●	16	Data	●
12	Data	●	14	+5VDC	●	17	Ground	●
15	Clock	●	15	Ground	●	18	+5VDC	●

Grafikkarten Ausgänge

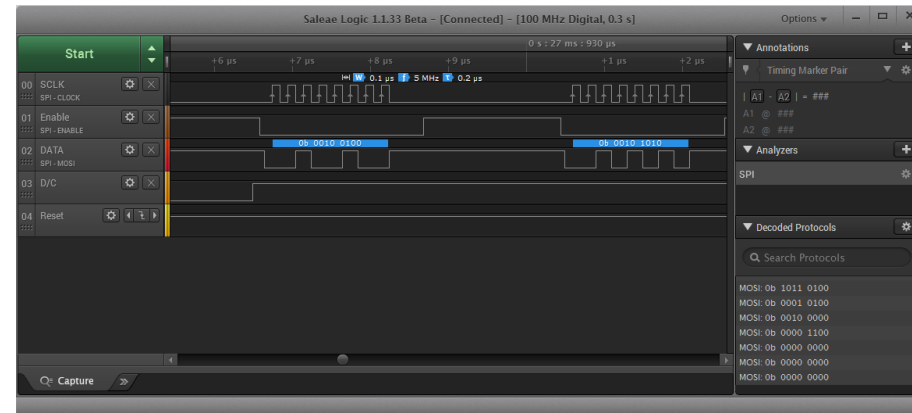
I2C Workshop

Reverse Engineering für FPGAs

- TM1637 Display mit „TWI“ Interface (ohne Busteilnehmeradresse)
- Datenblatt TM1637: <https://is.gd/ikawuf>
- OpenCores VHDL Projekt: <https://is.gd/cuvevu>



TM1637 Modul Logic Analyzer (eBay 10€)
(Saleae kompatibel)



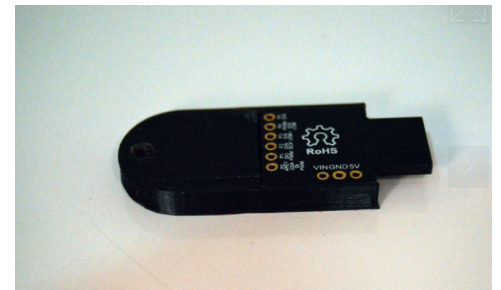
Saleae Logic Software
(kostenlos)

I2C Workshop

BadUSB

Vgl. <https://www.itsb.ruhr-uni-bochum.de/themen/badusb.html>

- Digispark gibt sich als Tastatur (oder Maus) aus, öffnet ein Terminal Fenster / Powershell Fenster, lädt Schadcode nach und führt diesen aus
- Ein Beispiel findet sich in der Zeitschrift iX Ausgabe 7/2017 S.119 - 121 für Einsatz von Metasploit für eine Reverse Shell unter Windows und Linux
- Für eine Metasploit Lösung sollten diese Anleitungen befolgt werden: <https://is.gd/vocelo> und <https://is.gd/ozajay>
- Evtl. gibt es eine bessere Lösung mit Netcat



Digispark und Thingiverse Gehäuse:
<https://is.gd/uhojan>

I2C Workshop

Mouse Jiggler

→ <https://is.gd/satodo>

- Arduino Lösung, die die Maus „wackeln“ lässt, damit der Bildschirmschoner nicht einsetzt
- Der Workflow für das Kompilieren von Arduino C-Programmen (.ino) für den Einsatz als Maus kann so nachgehalten werden (unter Windows)
- Die Inbetriebnahme (unter Linux) findet sich unter <https://is.gd/arijek>



I2C Workshop

Live!

Demonstration

Linux I2C-Tools
Display mit Netcat
LCDproc Statistiken
BH1750 Helligkeitssensor
BadUSB (Metasploit)



I2C Workshop

Vielen Dank für Eure Aufmerksamkeit!

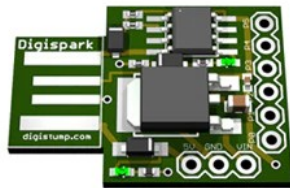
Evtl. habt Ihr noch Fragen ...



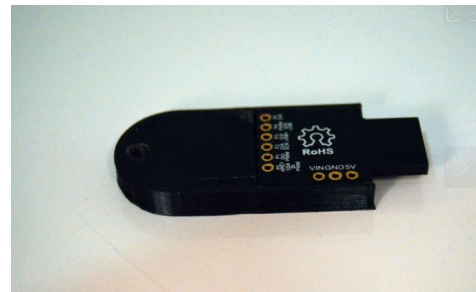
I2C Workshop

Lötworkshop

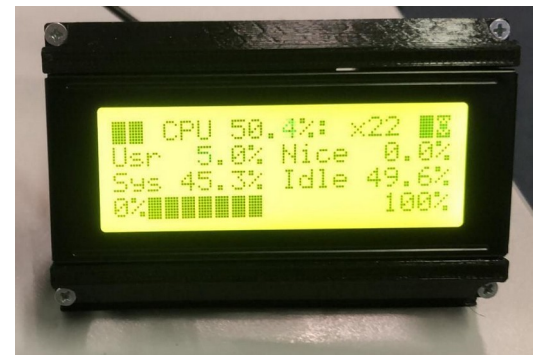
- Es wurden **10 Display Bausätze** zusammengestellt (und verkauft)
- Ebenso wurden **10 Bad USB Sets** zusammengestellt – Digispark mit Gehäuse (diese sind noch zu haben)
- Für die Personen, die beides nicht erhalten können, gibt es zumindest die **Digispark Bad USB Gehäuse** (3D-gedruckt)



Digispark



Bad USB mit Gehäuse



Display mit Gehäuse

I2C Workshop

Lötworkshop

- Nun Verlöten wir das Display und nehmen es in Betrieb
- Anleitung unter → wt.mongoq.de/lt/files/loetanleitung ←
- Lötzinn Pb60Sn40 mit Blei: „macht's einfacher“
- Abstandshalter zwischen Display und Adapterplatine mit Blu-Tack notwendig. **ACHTUNG: ENTWEDER KURZSCHLUSS ODER NICHTPASSEN INS GEHÄUSE, WENN FALSCH VERLÖTET!** (Sonst ist ein Entlöten notwendig!)



I2C Workshop

Weiterführendes: LCDproc Menü

- Beim LCDd Daemon besteht die Möglichkeit, ein Menü aufzurufen über Hardware Buttons
- Idee: Adafruit Pi Plate, für das es einen Treiber gibt → <https://is.gd/obaluv> ... reverse engineeren
→ der MCP23017 ist ein 16-Bit I/O Portexpander; gemäß <https://is.gd/ediloj> die Schematics sichten, C-Code gemäß <https://is.gd/xabace> (!)
- IMON LIRC Anbindung: <https://is.gd/cuhago>
- Lange LCDd.conf → <https://is.gd/igoteb>
- HD44780 (aber ohne I2C) → <https://is.gd/abepoz>
- Arduino LIRC Empfänger: <https://is.gd/gitavo>

