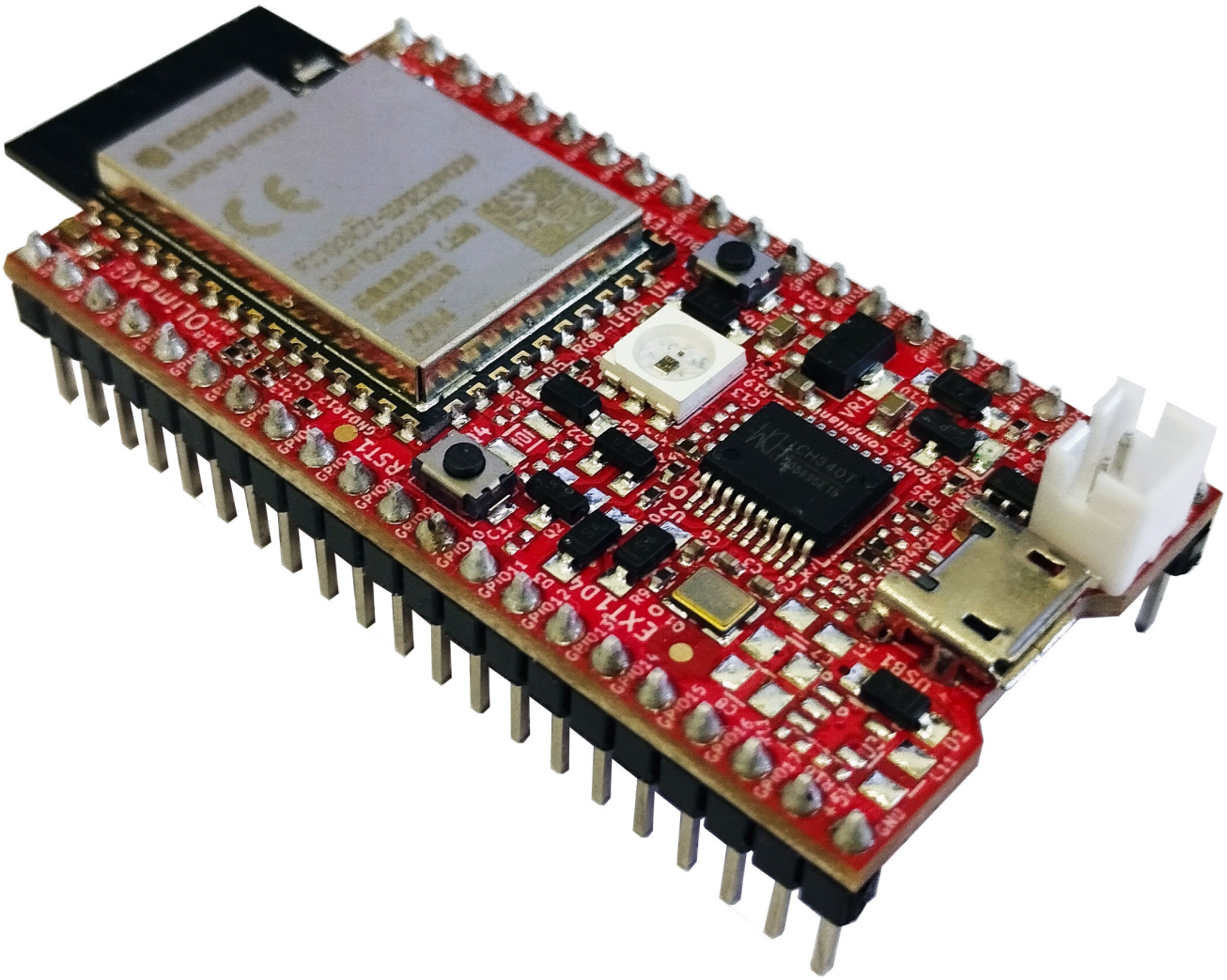


# ESP32-S2-DevKit-LiPo

## User's Manual

**Document revision 1.0, April 2026**

**www.olimex.com**

# Table of Contents

# 1. What is ESP32-S2-DevKit-LiPo

ESP32-S2-DevKit-LiPo is an Open Source Hardware development board that incorporates an ESP32-S2 module. The ESP32-S2-DevKit-LiPo board is designed and manufactured by Olimex, while the ESP32-S2 module is designed and manufactured by Espressif systems. The design is inspired by ESP32-S2-SAOLA-1 and has the same pinout, but adds Li-Po battery connector and charger. The board can operate on a Li-Po power when external power supply goes missing, allowing for handheld applications and increasing availability and reliability.

The ESP32-S2 modules are extremely popular WIFI modules due to their size, price, and very good documentation. The typical use is for WiFi enabled devices. ESP32-S2 supports low power modes including deep sleep that goes as low as 20uA.

Compared to ESP32, ESP32-S2 has only a single core and misses the Ethernet and Bluetooth connectivity, but has much more GPIOs which were missing in ESP32.

The ESP32-S2-DevKit-LiPo board has the following features:

- ESP32-S2-WROOM module or ESP32-S2-WROVER module
- RGB user LED
- User button, Reset button
- Micro USB-OTG connector for powering and programming; connected to CH340 serial-USB converter
- Uses CH340 Serial-USB adapter
- Built-in LiPo battery charger
- LiPo battery connector
- Battery measurement and power detection circuits
- Two columns of pins with headers soldered for easy access to all the board’s GPIOs
- Low-power design for extended operation on battery
- PCB dimensions: (1.9×1.1)" ~ (4.8×2.8)cm
- Operating temperature: -40+85C

## 1.1 Difference between ESP32-S2-DevKit-LiPo and ESP32-S2-DevKit-LiPo-USB

ESP32-S2-DevKit-LiPo-USB uses the built-in ESP32-S2 USB interface while ESP32-S2-DevKit-LiPo uses external USB-serial adapter (CH340) for the USB communication.

## 1.2 ESP32-S2-DevKit-LiPo variants

The board has 4 variants:

- **ESP32-S2-DevKit-LiPo**
- **ESP32-S2-DevKit-LiPo-EA**
- **ESP32-S2-WROVER-DevKit-LiPo**
- **ESP32-S2-WROVER-DevKit-LiPo-EA**

All board variants work in the industrial temperature range (-40+85 degrees Celsius).

ESP32-S2-DevKit-LiPo and ESP32-S2-DevKit-LiPo-USB-EA come with ESP32-WROOM-32E module;

ESP32-S2-WROVER-DevKit-LiPo-USB and ESP32-S2-WROVER-DevKit-LiPo-USB-EA come with ESP32-S2-WROVER-I-4MB;

Both variants with -EA suffix (ESP32-S2-DevKit-LiPo-USB-EA and ESP32-S2-WROVER-DevKit-LiPo-USB-EA) come with external 3dB antenna compatible with the module's u.FL connector.

## 1.2 Board use requirements

You only need a fitting USB cable and a personal computer. The board requires USB micro connector. Usually only such cable is required:

https://www.olimex.com/Products/Components/Cables/USB-CABLE-A-MICRO-1.8M/

The computer needs software compatible with ESP32 modules. Most commonly used tools are ESP-IDF and Arduino IDE with ESP32 package. You can use ESP32-S2-DevKit-LiPo with any software tool that supports the main ESP32 module.

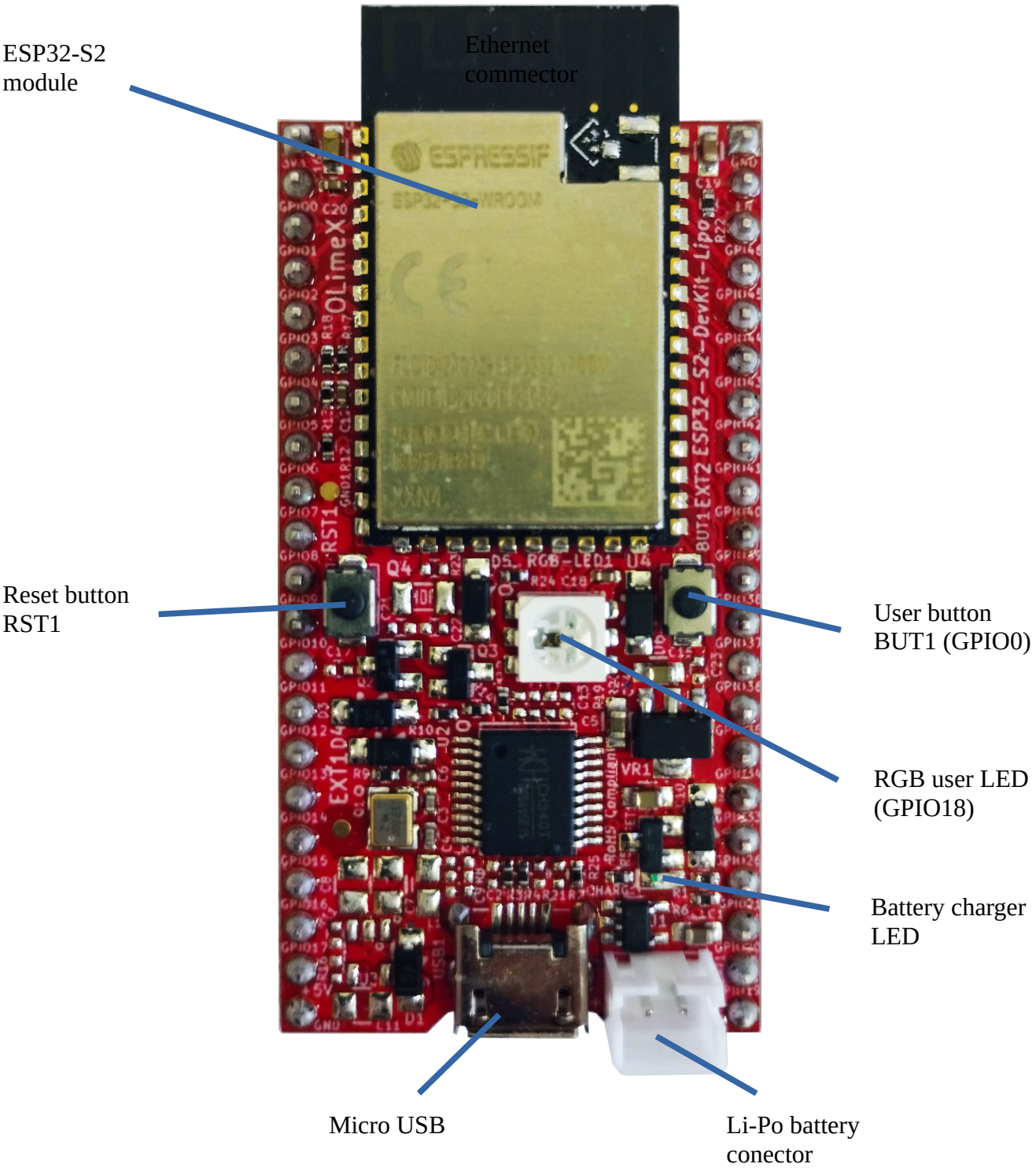
## 1.3 ESP32-S2-DevKit-LiPo-USB Open Source Licenses

ESP32-S2-DevKit-LiPo is Open Source Hardware, listed in OSHWA.org here:

https://certification.oshwa.org/bg000106.html

The hardware files are released under CERN OSHW license.
The software is released under GPL 3 license.
The documentation is released under CC BY-SA 3.0 license.

## 2. General board layout



# 3. Power supply and consumption

ESP32-S2-DevKit-LiPo-USB typically consumes around 50mA of current depending on the software. The board can consume much less using the power saving modes – down to around 20uA in deep sleep mode.

The absolute maximum power ESP32-S2-DevKit-LiPo-USB can draw from the power supply would be the determined by the maximum input of the regulator on the power input line. Of course, consider that on-board peripherals and the main module would use some of that current. The board uses low power regulator MCP1700T, it was chosen since it consumes only 1.5uA in power down mode. The downside is that it doesn't provide a lot of current, up to a maximum of 500mA. Leave 100-150 for the ESP32-S2 module, you are left with 350-400mA for additional circuits.

There is an option to disable the RGB LED via PTH jumper at the bottom. The jumper is labelled "LED_ENABLE1".

# 4. Schematics and dimensions

ESP32-S2-DevKit-LiPo-USB was designed with KiCAD (free and open-source CAD tool). ESP32-S2-DevKit-LiPo-USB schematics and sources can be found at GitHub here:

https://github.com/OLIMEX/ESP32-S2-DevKit-LiPo-USB/tree/main/HARDWARE

There are also PDF exports if you don't want to install KiCAD.

# 5. ESP32-S2-DevKit-LiPo pinout description

All pin names and functions are printed in white print at the bottom of the board. The board's pinout can also be seen in the schematic's bottom left corner.

The ESP32 chip has very good multiplexer so you can set the free GPIO pins for alternative functions via software means.

The board has battery power sense and battery power measurement optional feature, that can be enabled by modifying the state jumpers PWR_SENS_E1 and BAT_SENS_E1. Close them (solder the pads together) to enable both functions, on pins GPIO7 and GPIO8.

The board also has an option for external quartz, there are pads provided (located just under the RST1 reset button).

# 6. Software installation

Espressif guide for Arduino IDE installation – after installation – there is own entry for the board, it should be listed as OLIMEX ESP32-S2-DevKit-LiPo in the board selection.

Olimex provides an Arduino example here:

https://github.com/OLIMEX/ESP32-S2-DevKit-LiPo-USB/blob/main/SOFTWARE/ESP32-S2-DevKit-USB_RGB_LED.ino

Espressif ESP-IDF installation.

Espressif guide for PlatformIO installation.

# 7. Step-by-step Arduino IDE installation

The instructions were made under Windows 10 but should be pretty similar for different operating systems.

## 7.1. Download and install Arduino IDE

If you still don't have Arduino IDE, navigate to the following web-address to download it:

https://www.arduino.cc/en/software/

Download the version suitable for your operating system and install it (or extract it if you use the stand-alone version). After the installation is complete, launch the Arduino IDE application.

## 7.2. Install the ESP32-S2 support for Arduino IDE

**7.2.1.** Install Espressif Arduino-ESP32 stable release. To do so follow the detailed instructions here:

https://docs.espressif.com/projects/arduino-esp32/en/latest/installing.html

Short summary of what needs to be done:

- In Arduino IDE navigate to "File" → "Preferences" and in the filed that says "Additional Boards Manager URLs" append the following json link (copy-paste):

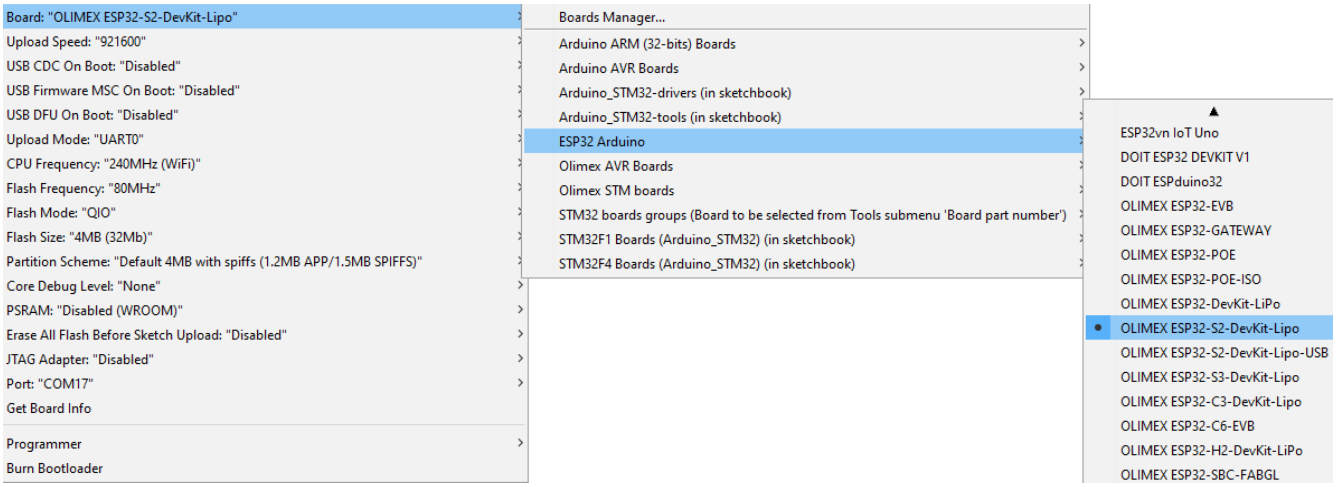
***https://espressif.github.io/arduino-esp32/package_esp32_index.json***

- Navigate to "Tools" → "Boards" → "Board Manager" and search for the *esp32* platform. Install the latest version (3.0.4 at the time of writing this document).


If you encounter problems during the package installation make sure to check all details on the official guide from the link on the previous page.

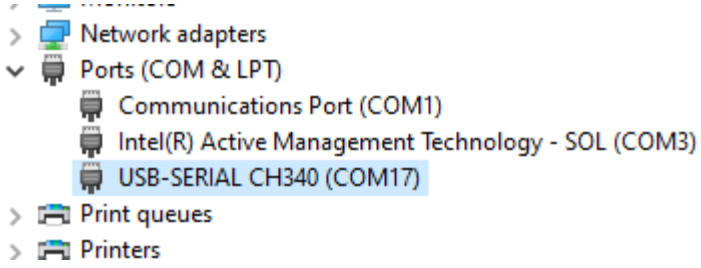
**7.2.2.** Restart Arduino IDE and select the configuration for the board. Navigate to “Tools” → “Board” →under “ESP32 Arduino” tab. Find in the list “OLIMEX ESP32-S2-DevKit-Lipo-USB” and select it.

You can also change the properties (ESP32-WROOM and ESP32-WROVER require slightly different settings, since ESP32-WROVER has extra 8MB PSRAM). If you have WROVER version, enable PSRAM. Remember to have “Internal USB” as “Upload Mode:” selected and to select the proper COM port.




## 7.3. Put the board in bootloader mode

First connect the ESP32-S2 board to your computer via the micro USB connector using USB cable. The LED should stop blinking and new device would be visible in “Windows Device Manager”. In the “Windows Device Manager” you should be seeing a new device. The one we need is called “USB-SERIAL CH340” (make sure to identify it, it will be different):



## 7.4. Upload a sketch

**7.4.1.** After you enter the bootloader mode as mentioned above, navigate to “Tools” → “Port…” select the COM of your device (make sure to identify it from “Windows Device Manager”):

**7.4.2.** Compile and upload your sketch – there is an example provided for the RGB LED at the end of this document;

**7.4.3.** If everything is alright the demo would be uploaded successfully.

**7.4.4.** Press button BUT1 to change LED mode, and also open the serial monitor from Tools and check the status of the LED.

## 7.5. RGB LED demo code

The demo is based on Adafruit demo. You can find the demo code below or at the link here:

https://github.com/OLIMEX/ESP32-S2-DevKit-LiPo/blob/main/SOFTWARE/ARDUINO/ESP32-S2-DevKit-RGB-LED-button-serial/ESP32-S2-DevKit-RGB-LED-button-serial.ino

```
//Serial/button/RGB LED demo for Olimex ESP32-S2-DevKit-Lipo
//used ESP Arduino package v3.3.7 and Arduino IDE 1.8.19
//open serial to see output, press button to change mode

#include <Arduino.h>
#include "driver/rmt_tx.h"
#include "driver/gpio.h"

#define LED_PIN 18
#define BUTTON_PIN 0
#define RMT_RESOLUTION_HZ 10000000

rmt_channel_handle_t tx_chan = NULL;
rmt_encoder_handle_t encoder = NULL;

bool lastButton = HIGH;
uint8_t mode = 0;

uint32_t lastAnim = 0;
uint8_t rainbow = 0;

const char* modeDescription[] = {
  "Mode 0: LED OFF",
  "Mode 1: Red LED always ON",
  "Mode 2: Green LED always ON",
  "Mode 3: Blue LED always ON",
  "Mode 4: Rainbow animation"
};

void ws2812_init()
{
  rmt_tx_channel_config_t tx_config = {
```

```
      .gpio_num = (gpio_num_t)LED_PIN,
      .clk_src = RMT_CLK_SRC_DEFAULT,
      .resolution_hz = RMT_RESOLUTION_HZ,
      .mem_block_symbols = 64,
      .trans_queue_depth = 4
  };

  rmt_new_tx_channel(&tx_config, &tx_chan);

  rmt_bytes_encoder_config_t bytes_config;

  bytes_config.bit0.duration0 = 4;
  bytes_config.bit0.level0 = 1;
  bytes_config.bit0.duration1 = 8;
  bytes_config.bit0.level1 = 0;

  bytes_config.bit1.duration0 = 8;
  bytes_config.bit1.level0 = 1;
  bytes_config.bit1.duration1 = 4;
  bytes_config.bit1.level1 = 0;

  bytes_config.flags.msb_first = 1;

  rmt_new_bytes_encoder(&bytes_config, &encoder);

  rmt_enable(tx_chan);
}

void ws2812_write(uint8_t r, uint8_t g, uint8_t b)
{
  uint8_t data[3] = {g,r,b};

  rmt_transmit_config_t tx_config = {0};

  rmt_transmit(tx_chan, encoder, data, sizeof(data), &tx_config);
  rmt_tx_wait_all_done(tx_chan, portMAX_DELAY);
}

void rainbowStep()
{
  uint8_t r,g,b;
  uint8_t pos = rainbow;

  if(pos < 85) {
    r = pos * 3;
    g = 255 - pos * 3;
    b = 0;
  }
  else if(pos < 170) {
    pos -= 85;
    r = 255 - pos * 3;
    g = 0;
    b = pos * 3;
  }
  else {
    pos -= 170;
    r = 0;
    g = pos * 3;
```

```
    b = 255 - pos * 3;
  }

  ws2812_write(r,g,b);
  rainbow++;
}

void setup()
{
  Serial.begin(115200);

  pinMode(BUTTON_PIN, INPUT_PULLUP);

  ws2812_init();

  Serial.println();
  Serial.println("ESP32-S2 RGB LED Demo");
  Serial.println("Press USER button to change LED mode");
  Serial.println(modeDescription[mode]);
}

void loop()
{
  bool button = digitalRead(BUTTON_PIN);

  if(lastButton == HIGH && button == LOW)
  {
    mode++;
    if(mode > 4) mode = 0;

    Serial.println();
    Serial.print("Button pressed -> ");
    Serial.println(modeDescription[mode]);
  }

  lastButton = button;

  if(mode == 0) ws2812_write(0,0,0);
  if(mode == 1) ws2812_write(40,0,0);
  if(mode == 2) ws2812_write(0,40,0);
  if(mode == 3) ws2812_write(0,0,40);

  if(mode == 4)
  {
    if(millis() - lastAnim > 30)
    {
      rainbowStep();
      lastAnim = millis();
    }
  }
}
```

### 7.6. Serial control demo code

A more complex Arduino demo that allows you to control the LED by commands send over the serial can be found here:

https://github.com/OLIMEX/ESP32-S2-DevKit-LiPo/blob/main/SOFTWARE/ARDUINO/Serial-control-RGB-led/Serial-control-RGB-led.ino

## 8. Document revision history

- Revision 1.0 April 2026

  - Initial release