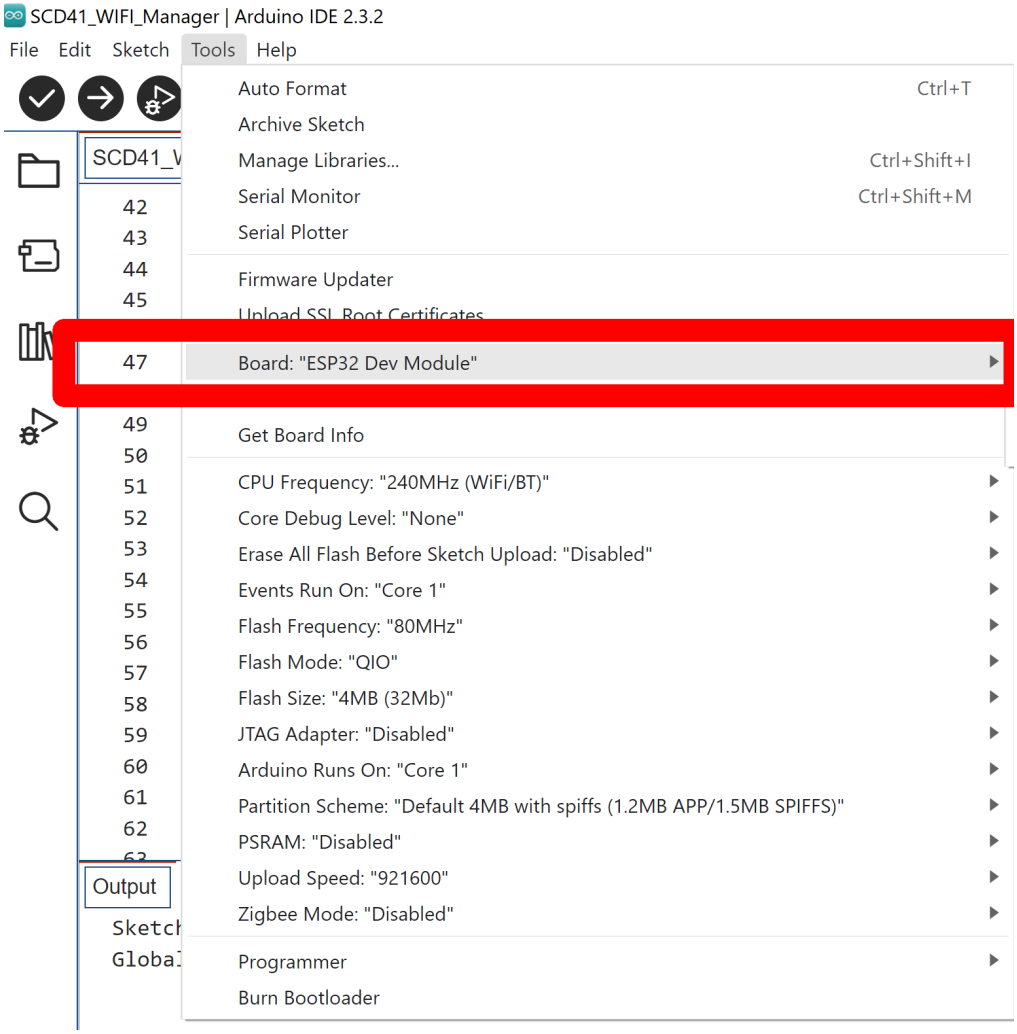
ESP32-DevKitI2C-RGB Getting Started Guide

This guide shows how to start using the ESP32-DevKitI2C-(RGB) development board.

Overview

ESP32-DevKitI2C-RGB is a small-sized ESP32-WROOM-32E based development board using a powerful and affordable WiFi module from Espressif. Some of the I/O pins are broken out to the pin headers on both sides for easy interfacing. Developers can either connect sensors using a 2.54mm pitch connector or a QWIIC/STEMMA/uSup connector or mount ESP32-DevKitI2C on a breadboard. This board is fully SW compatible with ESP32-DevKitC-32E so any code or settings used with ESP32-DevKitC-32E (WS2812B which is missing on the original ESP32-DevKitC-32E board) will work. For Arduino environment please choose ESP32 Dev Module while working with this board:



Other SW platforms like Tasmota or ESPHome are working in the same way as with ESP32-DevKitC-32E board + there is a full color LED available for status indication.

To cover a wide range of user requirements, the following versions of ESP32-DevKitI2C are available:

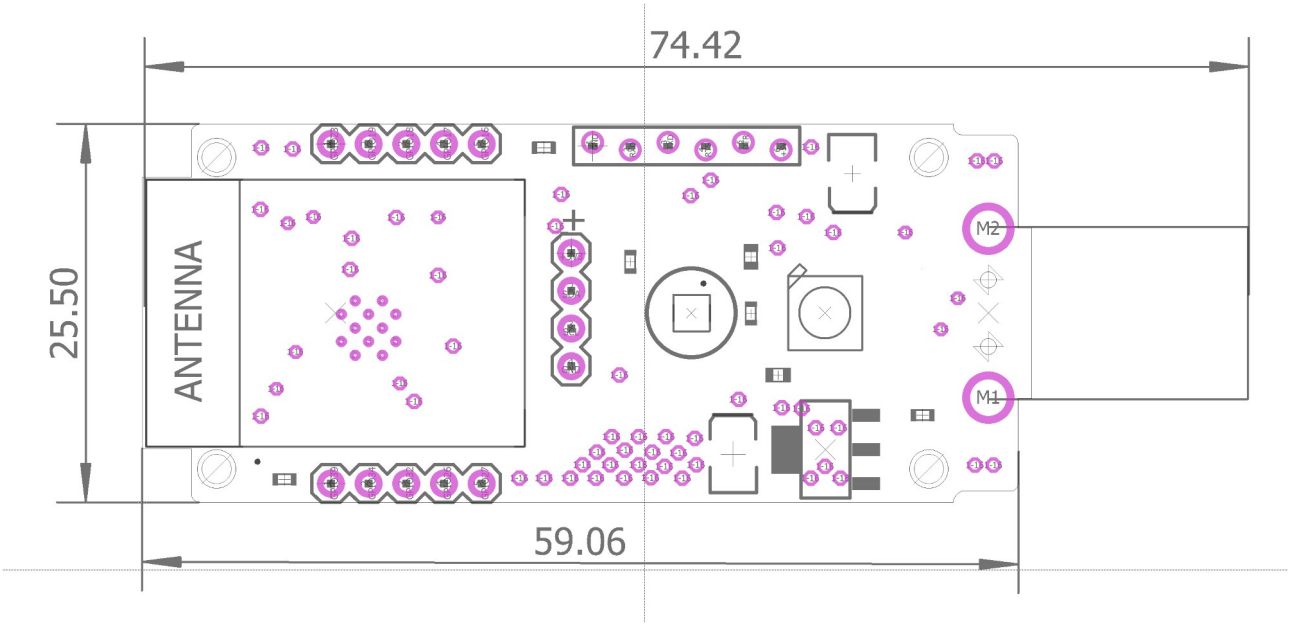
ESP32-DevKitI2C-RGB (Basic Version WS2812B LED)

ESP32-DevKitI2C-DARGB (Version with Dual Antenna ESP32-WROOM-DA)

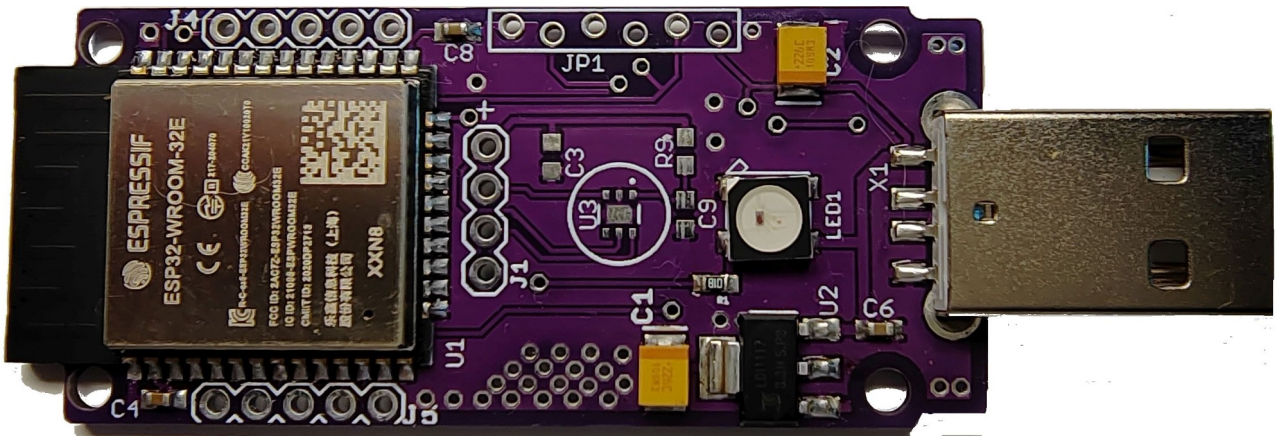
ESP32-DevKitI2C-VOC (Version with VOC/NOx sensor SGP41)

All versions are equipped with QWIIC/uŠup/STEMMA compatible connector in order to allow developers use a wide range of available sensors from different suppliers (Adafruit, Sparkfun, Laskakit, etc)

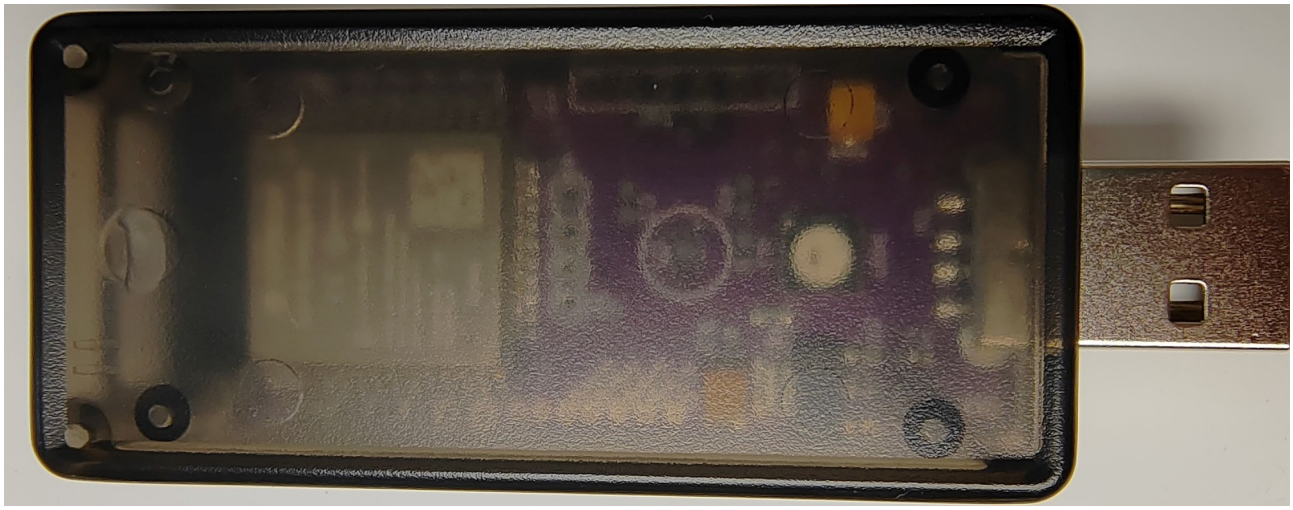
Key Component	Description
ESP32-WROOM-32	A module with ESP32 at its core. For more information, see ESP32-WROOM-32 Datasheet.
EN	Reset button.
USB-to-UART Bridge	Single USB-UART bridge chip provides transfer rates of up to 3 Mbps. FT231Q is used
USB A Port	USB interface. Power supply for the board as well as the communication interface between a computer and the ESP32-WROOM-32 module.
Neopixel WS2812B LED	For details see the schematics in Related Documents and sample code at the end of this User Guide
I/O	Some of the pins on the ESP module are broken out to the pin headers on the board. GPIO16-19, 23, 26, 27, 32, 34, 39 are broken out for further use.



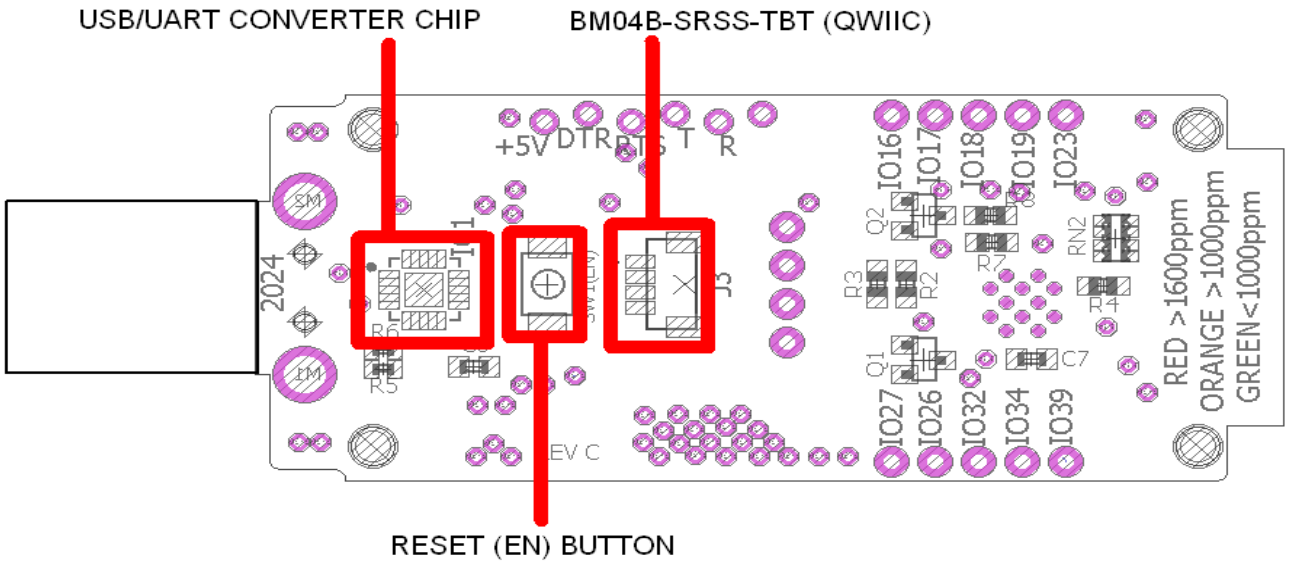
ESP32-DEVKITI2C-(RGB) dimensions and component placement



ESP32-DEVKITI2C-RGB top view (Basic version)



ESP32-DEVKITI2C-RGB mounted in semitransparent USB enclosure

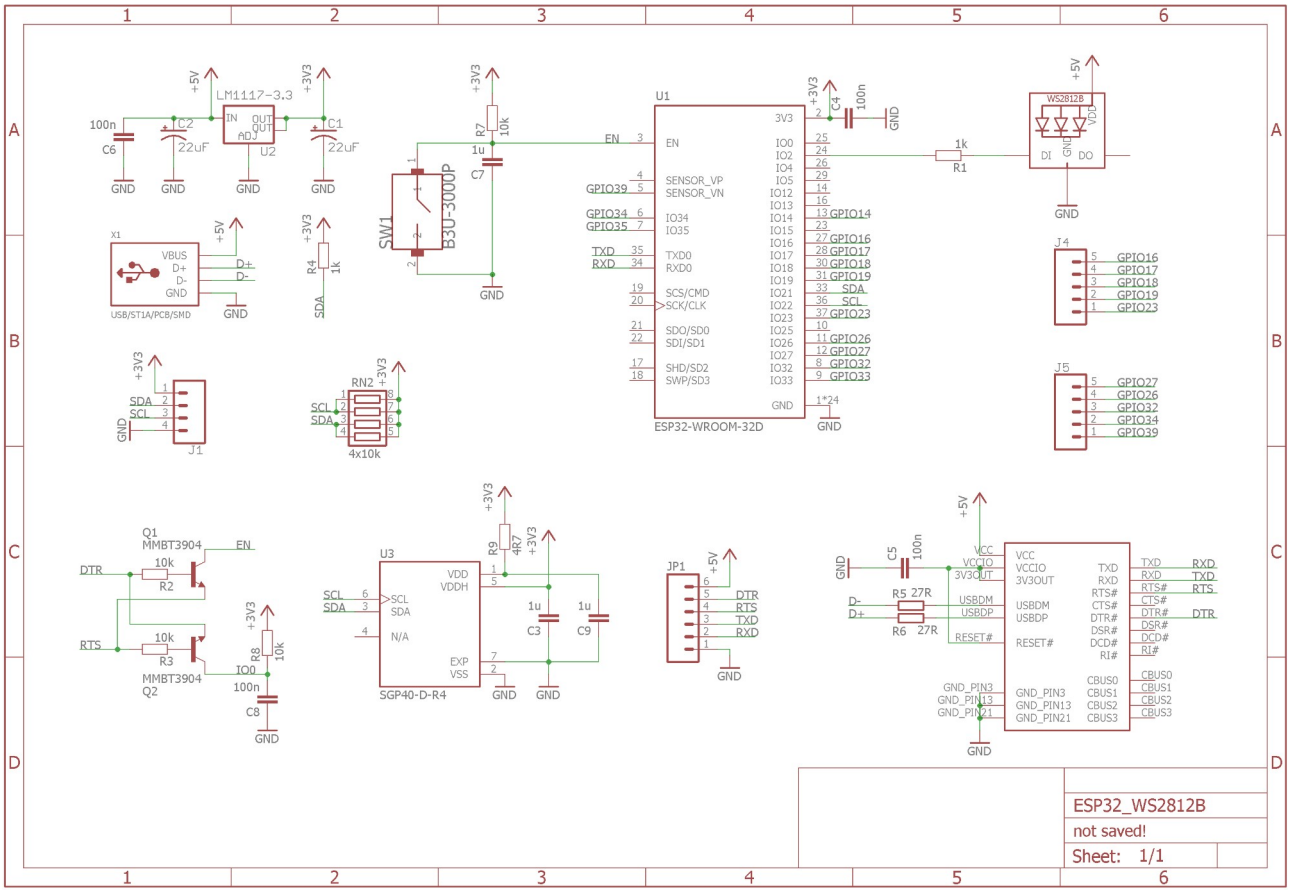


ESP32-DEVKITI2C-RGB bottom wiev

Power Supply

The board is powered from USB port – 5VDC/200mA max – you can use almost any 5V USB A adapter (charger) e.g. from an old smartphone.

Schematic diagram of ESP32-DEVKITI2C-RGB:



ESP32-DEVKITI2C-RGB Schematic (with WS2812B NeoPixel LED)

Sample code for Arduino to use with SCD4x CO2 sensor and TMEP.CZ cloud service:

```
/*
* Sample code for SCD41 used with ESP32-DEVKITI2C
* Values from CO2 sensor are sent via UART and simultaniosly to TMEP.cz server
* Sending frequency is 60s
*
* SCD41 https://www.soselectronic.com/en/products/various/scd-41-bob-398492
*/

#include <WiFiManager.h>
#include <HTTPClient.h>
#include <Adafruit_NeoPixel.h>
#include <Wire.h>
#include "SparkFun_SCD4x_Arduino_Library.h"
```

```
#define PIN 2
#define NUMPIXELS 1
Adafruit_NeoPixel pixels(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);

// please set up here your address, registered at tmep.cz
// e.g. you registered address "myCO2sensor"
String serverName = "http://myCO2sensor.tmep.cz/index.php?";

// here set your sensor´s GUID
String GUID = "GUID";

/* END OF SETUP */

SCD4x SCD41;

void setup() {

 Serial.begin(115200);
 Wire.begin();
 delay(10);

 WiFiManager manager;
 // manager.resetSettings();
  bool success = manager.autoConnect("ESP32_AP","password");

  if(!success) {
    Serial.println("Failed to connect");
  }
  else {
    Serial.println("Connected");

 // initialisation
 //              begin, autocalibration
 //                |      |
 if (SCD41.begin(false, true) == false)
 {
  Serial.println("SCD41 not found, check your connection");
  while(1)
   ;
 }

 // possible Low Power mode selection true/false
 if (SCD41.startLowPowerPeriodicMeasurement() == false)
 {
  Serial.println("Low power mode enabled");
 }
}
}

void loop() {
 while (!SCD41.readMeasurement()) // waiting for first data (cca 30s)
 {
  delay(100);
 }
 SCD41.readMeasurement(); // reading SCD41 measurement
```

```
 if(SCD41.getCO2()<=1000){ // here you can set threshold for GREEN LED
  pixels.setPixelColor(0, pixels.Color(0, 255, 0)); // here you can set
NEOPIXEL´s LED color (RED, GREEN, BLUE) - GREEN
  pixels.show();
  }
 if(SCD41.getCO2()>1000 && SCD41.getCO2()<=1600){ // here you can set threshold
for ORANGE LED
   pixels.setPixelColor(0, pixels.Color(130, 60, 0)); // here you can set
NEOPIXEL´s LED color (RED=130, GREEN=60, BLUE) - ORANGE
   pixels.show();
  }
 if(SCD41.getCO2()>1600){   // here you can set threshold for RED LED
  pixels.setPixelColor(0, pixels.Color(255, 0, 0)); // here you can set
NEOPIXEL´s LED color (RED=255, GREEN=60, BLUE) - RED
  pixels.show();
  }

 // sending measurments via UART
 Serial.print("Temperature: "); Serial.print(SCD41.getTemperature());
Serial.println(" degC");
 Serial.print("Humidity: "); Serial.print(SCD41.getHumidity());
Serial.println("% rH");
 Serial.print("CO2: "); Serial.print(SCD41.getCO2()); Serial.println(" ppm");

 // sending measurements to TMEP.cz server
 if(WiFi.status()== WL_CONNECTED)
 {
   HTTPClient http;

   //GUID, temperature, humidity, CO2 values sent to server
   String serverPath = serverName + "" + GUID + "=" + SCD41.getTemperature() +
"&humV=" + SCD41.getHumidity() + "&CO2=" + SCD41.getCO2();

   // starting http connection
   http.begin(serverPath.c_str());

   // http get request
   int httpResponseCode = http.GET();

   if (httpResponseCode>0)
   {
    Serial.print("HTTP answer: ");
    Serial.println(httpResponseCode);
    String payload = http.getString();
    Serial.println(payload);
   }
   else
   {
    Serial.print("Error code: ");
    Serial.println(httpResponseCode);
   }
   // Free resources
   http.end();
  }
  else
```

```
  {
   Serial.println("Wi-Fi disconnected");
  }

 delay(60000);
}
```

Sample .yaml configuration file for Home Assistant (ESPHome) and SCD4x CO2 sensor:

```
esphome:

 name: esphome-web-b35cb8
 friendly_name: Kitchen CO2 sensor

esp32:
 board: esp32dev
 framework:
  type: arduino

# Configure I2C bus of ESP32
i2c:
 sda: GPIO21
 scl: GPIO22
 scan: true
 id: bus_a

# Configure Neopixel LED
light:
- platform: neopixelbus
 variant: WS2812
 pin: 2
 num_leds: 1
 flash_transition_length: 100ms
 type: GRB
 id: activity_led
 name: "Neopixel LED"
 restore_mode: ALWAYS_OFF

# Configure SCD4x sensor
sensor:
 - platform: scd4x
  co2:
   name: "CO2 Level"
  temperature:
   name: "Temperature"
  humidity:
   name: "Humidity"

# Enable logging
```

```yaml
logger:

# Enable Home Assistant API
api:
 encryption:
  key: "**********************************"

ota:
 password: "************************"

wifi:
 ssid: !secret wifi_ssid
 password: !secret wifi_password

 # Enable fallback hotspot (captive portal) in case wifi connection fails
 ap:
  ssid: "Esphome-Web-B35Cb8"
  password: "***************"

captive_portal:
```