

KAMAMI

KAmoD ESP32 C3



Rev. 20260404124330

Źródło: https://wiki.kamamilabs.com/index.php?title=KAmoD_ESP32_C3

Table of contents

Description	1
Key Features and Parameters	2
Standard Equipment	3
Electrical Schematic	3
Pinout Description	4
Power Supply	5
Arduino IDE Configuration and Test Program	6
Rust Programming Setup and Test Programs	20
Dimensions	25
External Links	26

Description

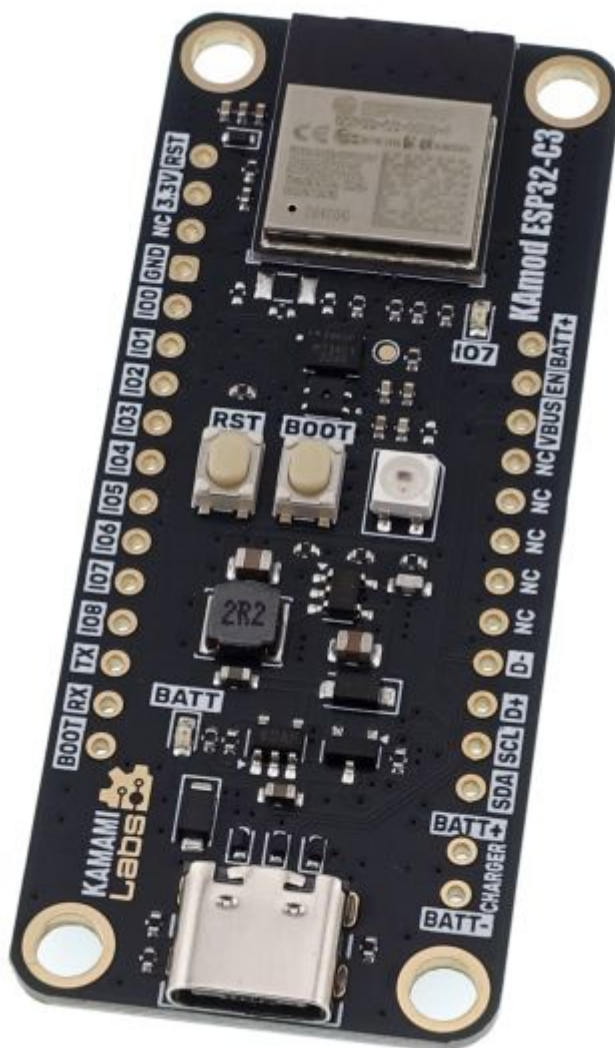
KAmoD ESP32-C3 - Development board with ESP32-C3 Mini-1 module

The KAmoD ESP32-C3 board features the ESP32-C3 Mini-1 module from Espressif, which contains a 32-bit, single-core SoC microcontroller based on the RISC-V architecture. It includes Wi-Fi and Bluetooth 5 (LE) radio interfaces with Long Range (LR) mode support.

The MCU operates at a maximum clock frequency of 160 MHz and is equipped with 400 kB of RAM and 4 MB of Flash memory. It supports low power consumption modes, operates in temperatures from -40 to +85°C, and implements security features such as Secure Boot and Flash encryption with AES-128/256-XTS. These extensive specifications make it ideal for industrial and IoT applications. Additionally, the board includes a high-precision SHTC3 temperature and humidity sensor, an ICM42670 MEMS sensor (3-axis gyroscope and 3-axis accelerometer), a WS2812 RGB LED, and a standard LED connected to one of the microcontroller ports. These components facilitate the rapid building of diverse applications.

The module is powered by +5V via a USB-C connector or by a Li-Ion battery. The +3.3 V voltage for the microcontroller and peripherals is generated by a SY8088 Buck DC/DC converter (input range 2.5-5 V). If USB-C power is disconnected, power is automatically drawn from the battery (if connected). The power system includes an MCP7381 Li-Ion battery charger powered via USB-C.

The KAmoD ESP32-C3 module is designed for developing and testing applications in the Arduino environment (C/C++) as well as in Rust.





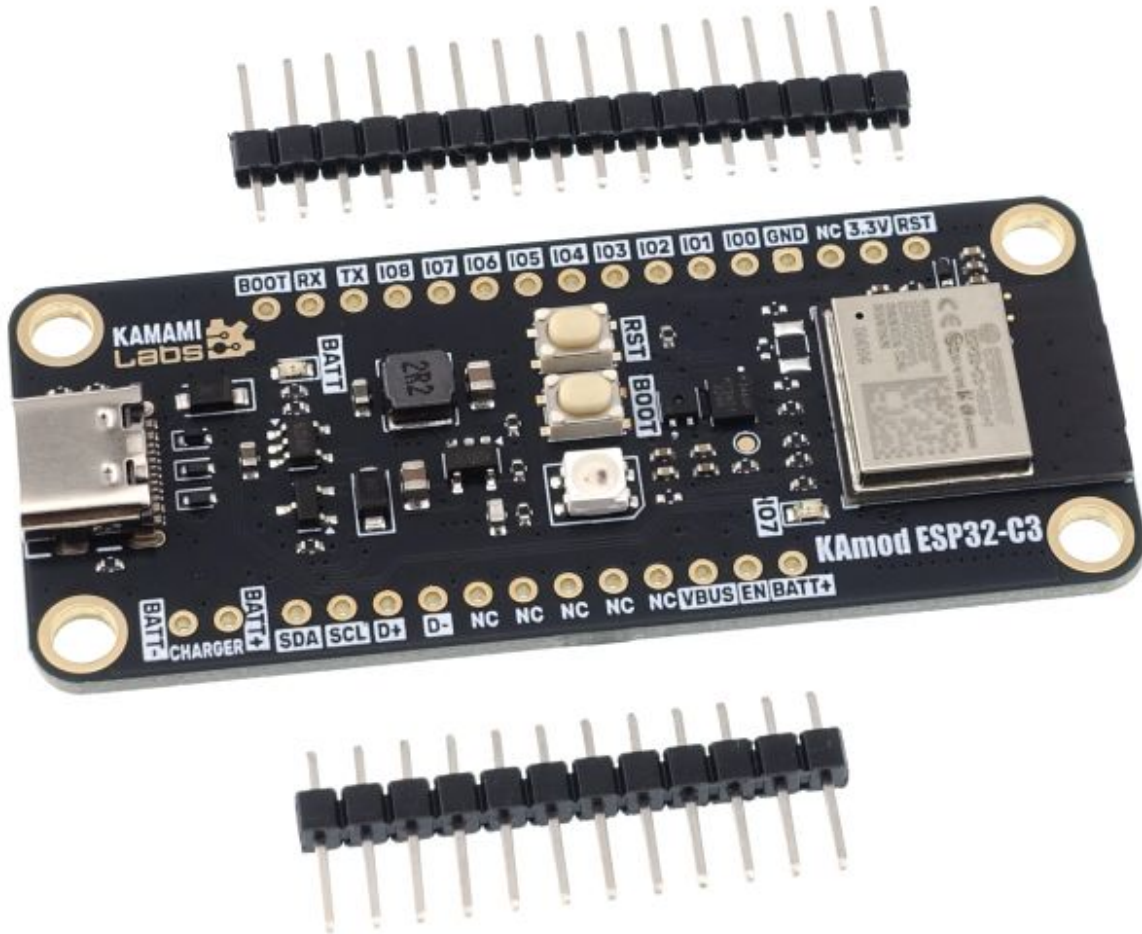
Key Features and Parameters

- ESP32C3 Mini-1 Module (Wi-Fi 802.11 b/g/n, Bluetooth 5 LE)
- 32-bit RISC-V single-core MCU @ 160 MHz, 400 kB SRAM, 4 MB Flash
- 15 GPIO lines
- Communication interfaces: SPI, I2C, I2S, UART, USB
- 12-bit SAR ADC (up to 6 channels)
- ICM42670 Accelerometer/Gyroscope
 - 3-axis MEMS gyroscope (X, Y, Z angular rate)
 - 3-axis MEMS accelerometer (X, Y, Z axis)
 - Communication interface: I2C
- SHTC3 Thermometer/Hygrometer
 - Humidity range: 0...100%RH (+/- 2% accuracy)
 - Temperature range: -40 to +125 °C (+/-0.2°C accuracy from 0°C to +60°C)
 - Communication interface: I2C
- SY8088 DC/DC Converter
- MCP73831 Li-Ion Battery Charger
- Programmable WS2812 RGB LED
- USB-C Connector
 - +5V Power supply

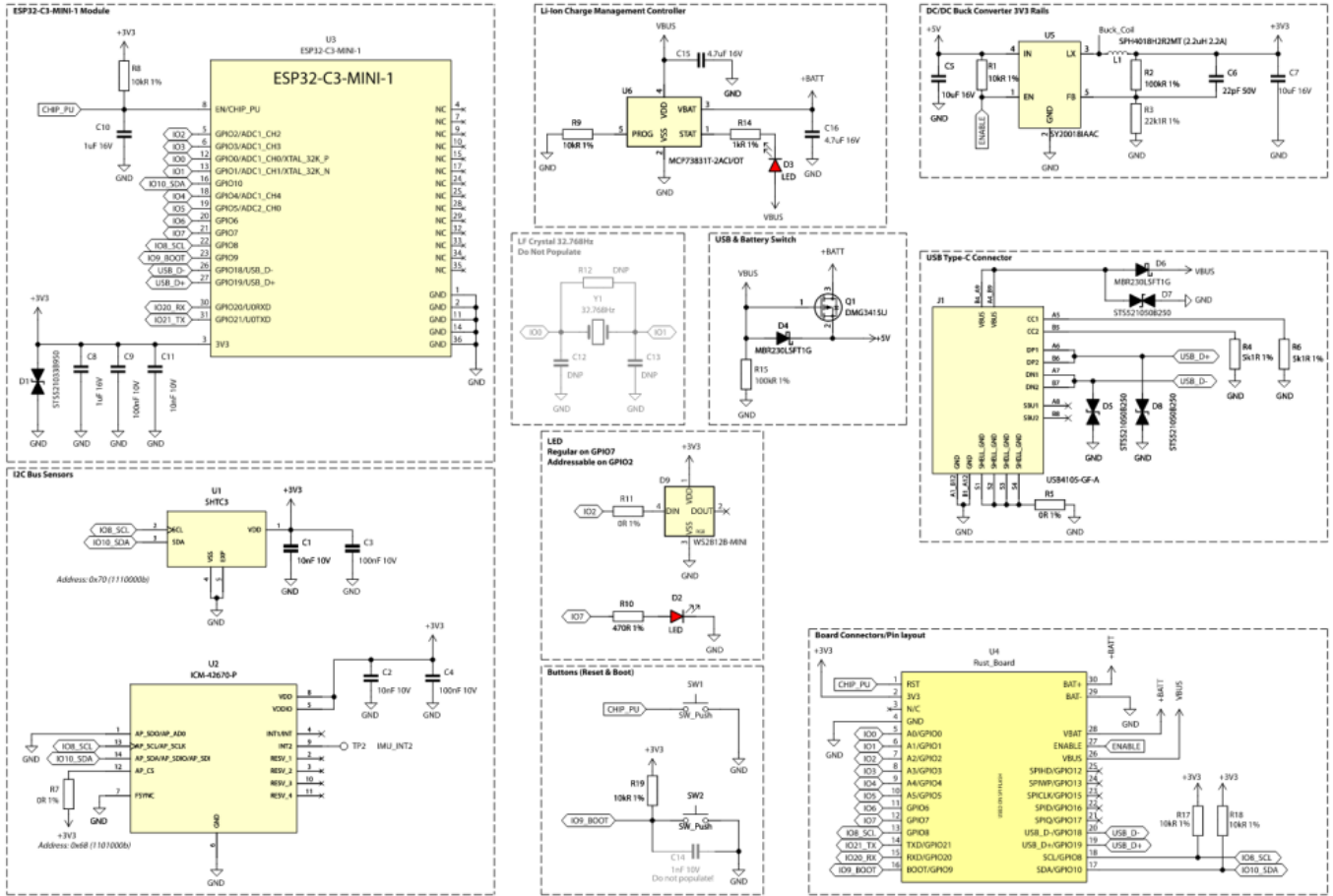
- Flash programming interface
- JTAG interface
- Reset and Boot buttons

Standard Equipment

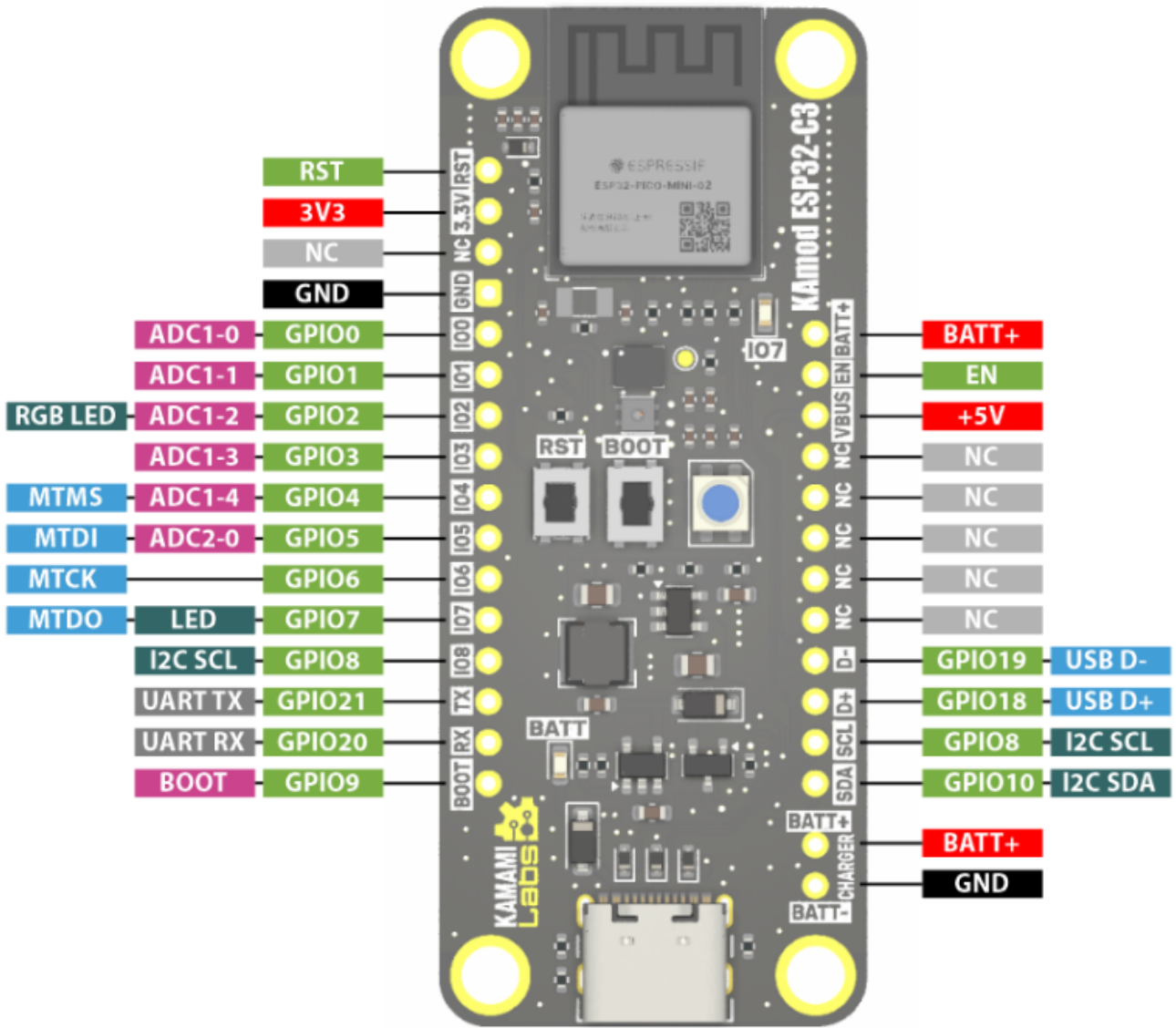
Code	Description
KAmo ESP32-C3	<ul style="list-style-type: none"> • Assembled and tested module • 1 x straight 12-pin goldpin header (2.54 mm pitch) • 1 x straight 16-pin goldpin header (2.54 mm pitch)



Electrical Schematic

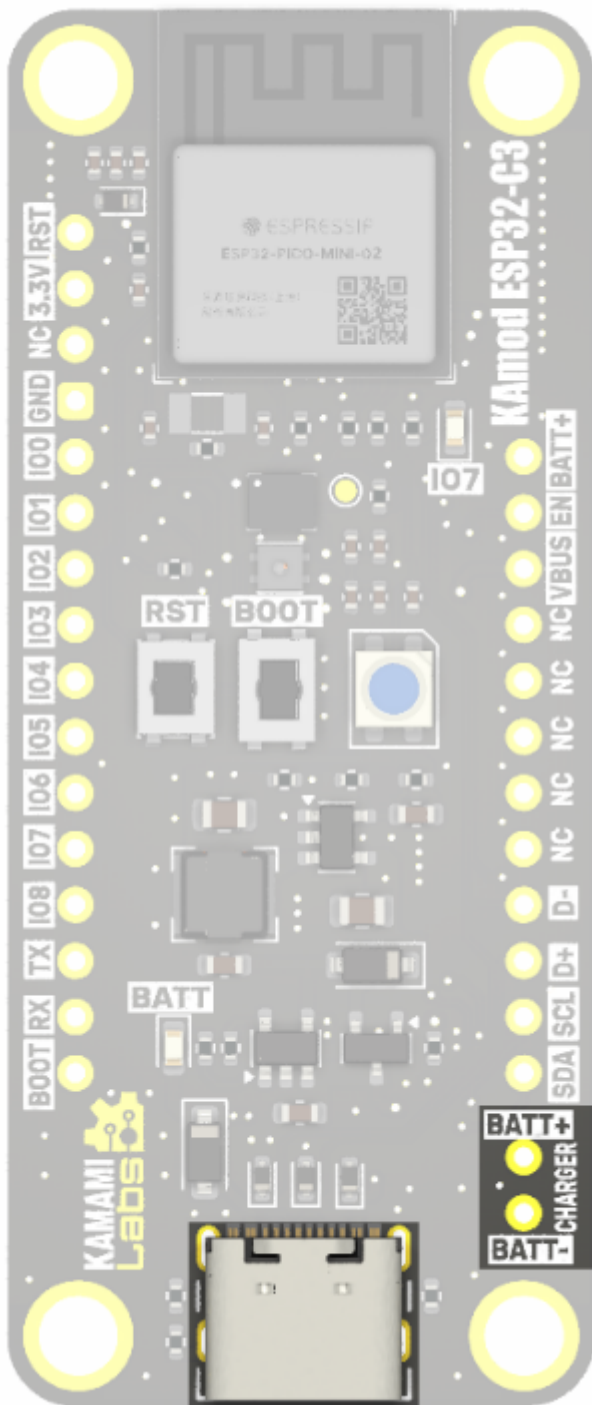


Pinout Description



Power Supply

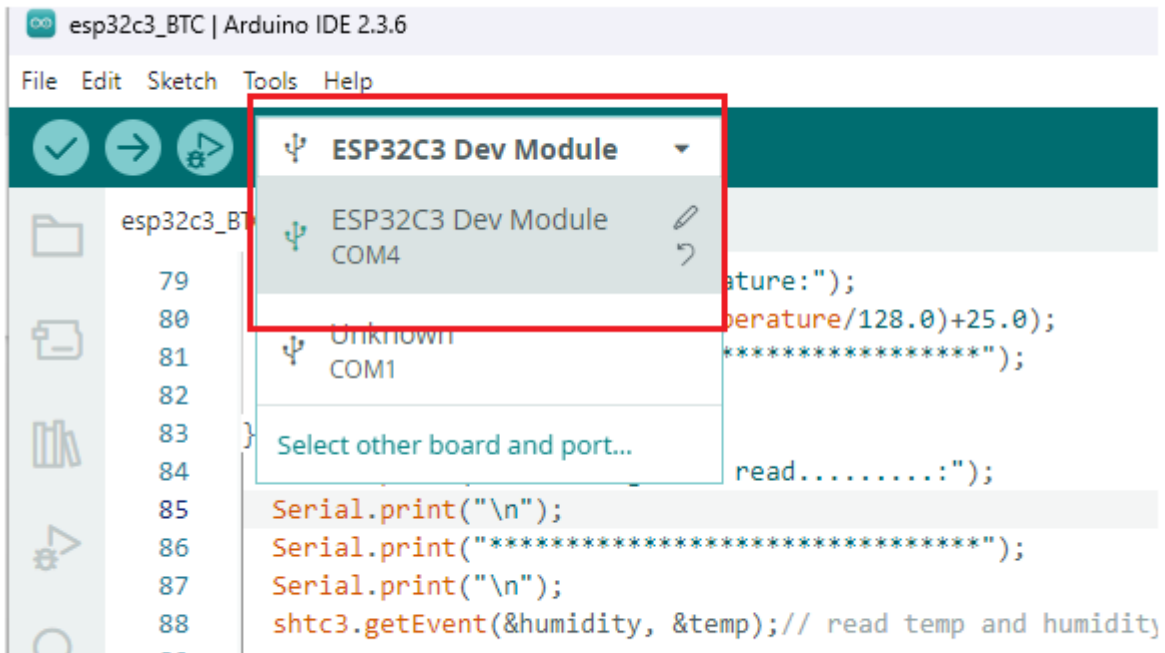
The module is powered by VBUS (+5 V) from the USB-C connector. This voltage also charges the 4.2 V Li-Ion battery if connected to the Charger output. The MCP7381 chip manages the charging process using a Constant Current/Constant Voltage (CC/CV) algorithm, with the charging current set to 100 mA and the termination voltage to 4.20 V.



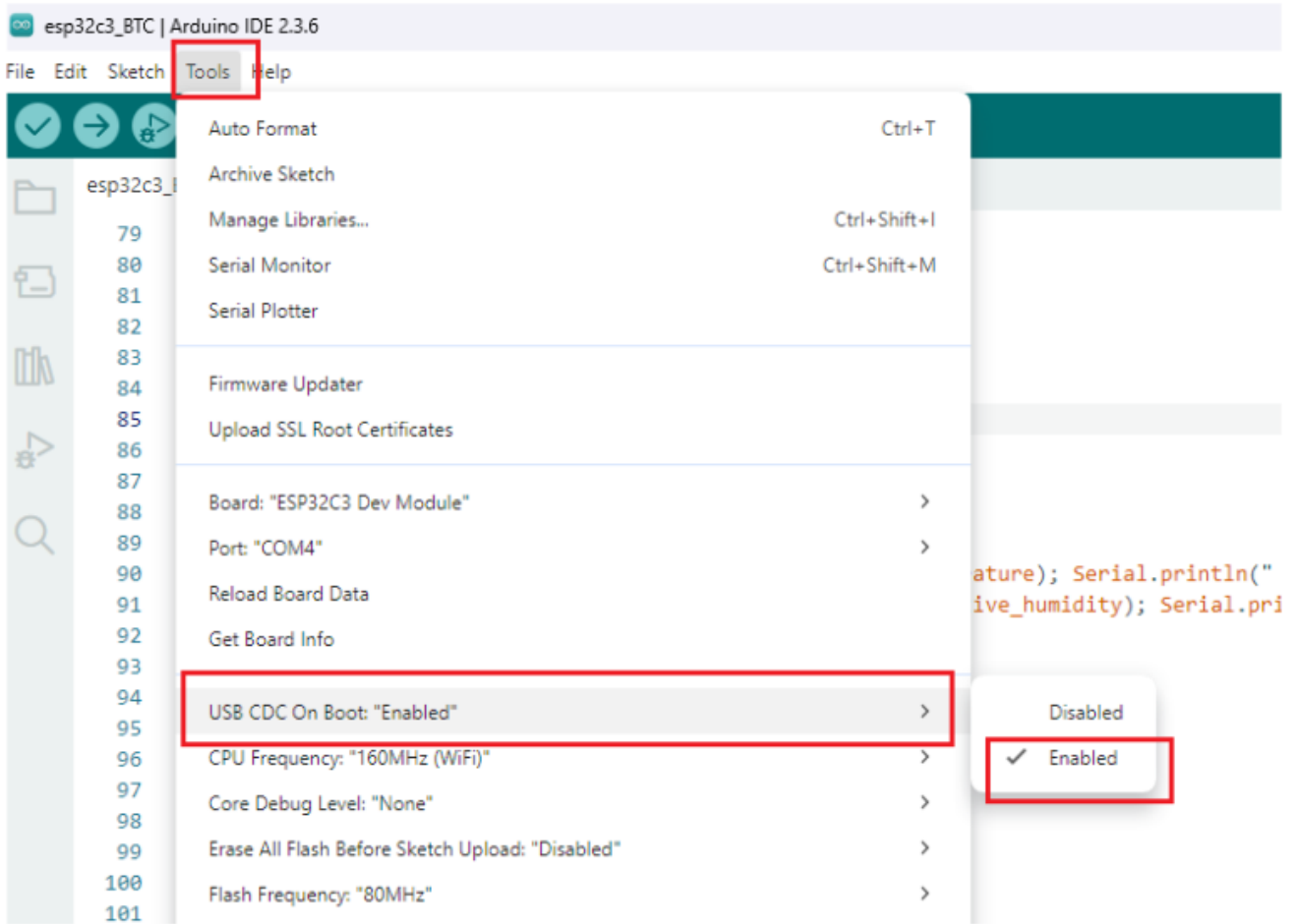
Arduino IDE Configuration and Test Program

Preliminary Steps

Connect the KAMod ESP-C3 module to a computer with Arduino IDE installed. In the board selection menu, choose *ESP32C3 Dev Module* and the corresponding virtual COMx port.

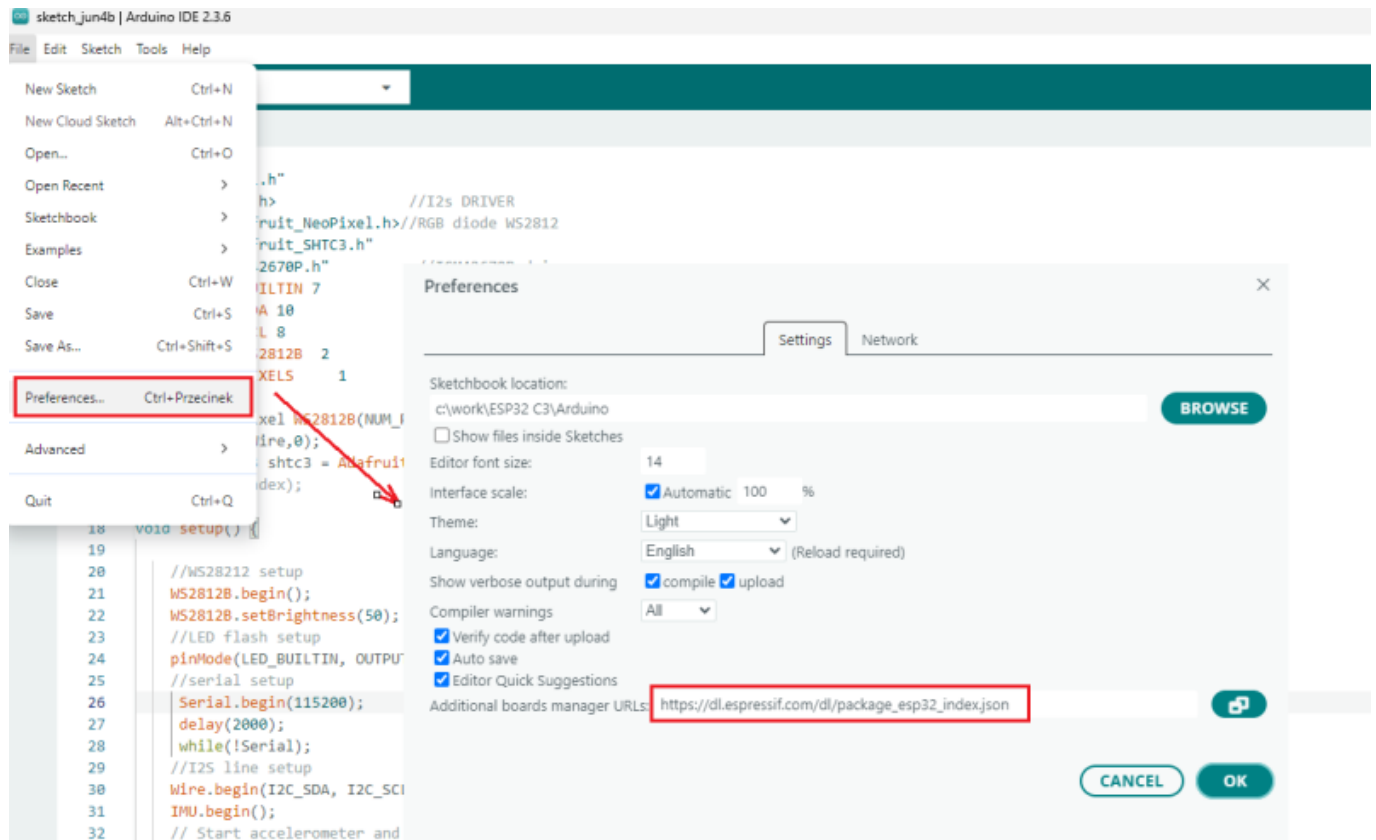


For the test procedure, we will use the Serial Monitor to display results. To enable this, you must unlock the *USB CDC On Boot* option (disabled by default).



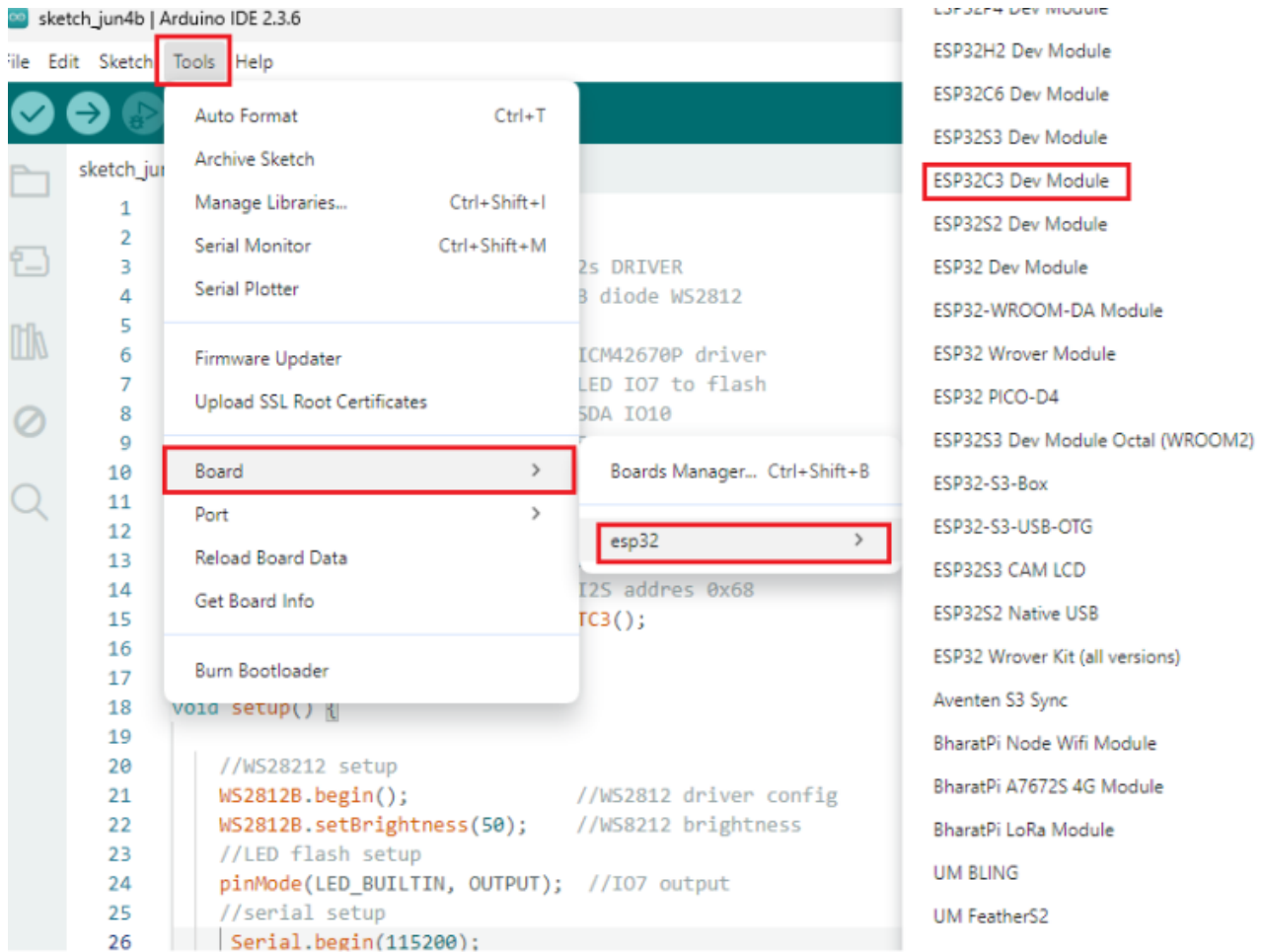
Preferences Configuration

Open *File -> Preferences*. In the *Additional boards manager URLs* field, enter:
https://dl.espressif.com/dl/package_esp32_index.json



Selecting the Processor Type

Navigate to *Tools* -> *Board* -> *esp32* -> *ESP32C3 Dev Module*. This step ensures the project compiles for the correct architecture.



Installing Libraries

The test program requires libraries for the ICM42670P accelerometer, SHTC3 thermometer/hygrometer, and WS2812B RGB LED.

ICM42670P Library

Search for "ICM42670" in the Library Manager, select *ICM42670P* by *TDK/Invensense* and click *Install*.

File Edit Sketch Tools Help

Select Board

LIBRARY MANAGER

ICM42670

Type: All

Topic: All

ICM42670P by TDK/Invensense

Allows to read accelerometer, gyroscope and temperature sensors from an ICM42670P Invensense IMU device. This library allows to easil...
More info

1.0.7

INSTALL

ICM42670S by TDK/Invensense

Allows to read accelerometer, gyroscope and temperature sensors from an ICM42670S Invensense IMU device. This library allows to easil...
More info

1.0.7

INSTALL

sketch_jun4b.ino

```

1
2 #include "WiFi.h"
3 #include <Wire.h>
4 #include <Adafruit_NeoPixel.h>
5 #include "Adafruit_SHTC3.h"
6 #include "ICM42670P.h"
7 #define LED_BUILTIN 7
8 #define I2C_SDA 10
9 #define I2C_SCL 8
10 #define PIN_WS2812B 2
11 #define NUM_PIXELS 1
12
13 Adafruit_NeoPixel WS2812B(NUM_PIXELS);
14 ICM42670 IMU(Wire,0);
15 Adafruit_SHTC3 shtc3 = Adafruit_SHTC3(I2C_SDA,I2C_SCL);
16 //WiFi.SSID(index);
17
18 void setup() {
19
20     //WS2812 setup
21     WS2812B.begin();
22     WS2812B.setBrightness(50);
23     //LED flash setup
24     pinMode(LED_BUILTIN, OUTPUT);
25     //serial setup
26     Serial.begin(115200);
27     delay(1000);

```

SHTC3 Library

Search for *Adafruit SHTC3 Library*. When prompted, click *Install All* to include necessary dependencies.

The screenshot shows the Arduino IDE interface. In the Library Manager, the search term "SHTC3" is entered. The results show the "Adafruit SHTC3 Library by Adafruit" with version 1.0.1 selected. The "INSTALL" button is highlighted with a red box. A red arrow points from this button to a dialog box titled "Install library dependencies".

The dialog box contains the following text:

Install library dependencies

The library **Adafruit SHTC3 Library:1.0.1** needs some other dependencies currently not installed:

- Adafruit BusIO
- Adafruit Unified Sensor

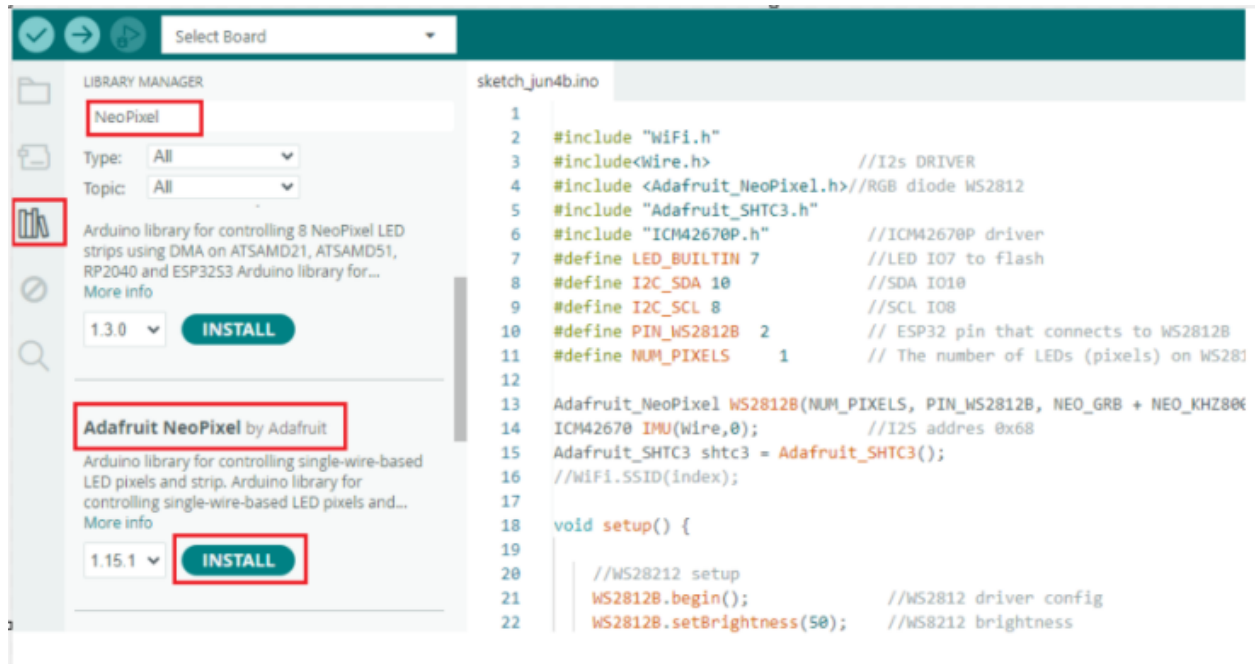
Would you like to install all the missing dependencies?

Two buttons are shown at the bottom: "INSTALL WITHOUT DEPENDENCIES" and "INSTALL ALL". The "INSTALL ALL" button is highlighted with a red box.

```
1
2 #include "WiFi.h"
3 #include<Wire.h> //I2s DRIVER
4 #include <Adafruit_NeoPixel.h>//RGB diode WS2812
5 #include "Adafruit_SHTC3.h"
6 #include "ICM42670P.h" //ICM42670P driver
7 #define LED_BUILTIN 7 //LED IO7 to flash
8 #define I2C_SDA 10 //SDA IO10
9 #define I2C_SCL 8 //SCL IO8
10 #define PIN_WS2812B 2 // ESP32 pin that contr
11 #define NUM_PIXELS 1 // The number of LEDs
12
13 Adafruit_NeoPixel WS2812B(NUM_PIXELS, PIN_WS2812B, NEC
14 ICM42670 IMU(Wire,0); //I2S address 0x68
15 Adafruit_SHTC3 shtc3 = Adafruit_SHTC3();
```

WS2812 RGB LED Library

Install the *Adafruit NeoPixel* library.



The screenshot shows the Arduino IDE interface. On the left, the Library Manager is open, displaying the search results for 'NeoPixel'. The 'NeoPixel' library by Adafruit is highlighted, with version 1.3.0 selected and the 'INSTALL' button visible. Below it, the 'Adafruit NeoPixel' library is also shown, with version 1.15.1 selected and the 'INSTALL' button visible. On the right, the sketch editor shows the code for 'sketch_jun4b.ino'. The code includes the necessary headers and defines the pin and number of LEDs. The setup function is also shown, initializing the WS2812 driver and setting the brightness to 50.

```
1
2 #include "WiFi.h"
3 #include <Wire.h> //I2s DRIVER
4 #include <Adafruit_NeoPixel.h> //RGB diode WS2812
5 #include "Adafruit_SHTC3.h"
6 #include "ICM42670P.h" //ICM42670P driver
7 #define LED_BUILTIN 7 //LED IO7 to flash
8 #define I2C_SDA 10 //SDA IO10
9 #define I2C_SCL 8 //SCL IO8
10 #define PIN_WS2812B 2 // ESP32 pin that connects to WS2812B
11 #define NUM_PIXELS 1 // The number of LEDs (pixels) on WS2812B
12
13 Adafruit_NeoPixel WS2812B(NUM_PIXELS, PIN_WS2812B, NEO_GRB + NEO_KHZ800);
14 ICM42670 IMU(Wire,0); //I2S address 0x68
15 Adafruit_SHTC3 shtc3 = Adafruit_SHTC3();
16 //WiFi.SSID(index);
17
18 void setup() {
19
20 //WS2812 setup
21 WS2812B.begin(); //WS2812 driver config
22 WS2812B.setBrightness(50); //WS2812 brightness
```

Handling the ICM42670P Accelerometer

The sensor uses the I2C bus. We initialize the interface using the Wire library:

1. `define I2C_SDA 10 // SDA on IO10`
2. `define I2C_SCL 8 // SCL on IO8`

```
Wire.begin(I2C_SDA, I2C_SCL);
```

Handling the SHTC3 Sensor

Initialized via the standard begin method:

```
shtc3.begin();
```

Reading data:

```
shtc3.getEvent(&humidity, &temp); // read temp and humidity
```

Results are stored in `temp.temperature` and `humidity.relative_humidity`.

```
*****  
SHTC3 Temperature: 31.54 deg C  
SHTC3 Humidity: 46.78% rH  
*****
```

Handling the WS2812B RGB LED

Define the number of pixels and the data pin:

1. `define PIN_WS2812B 2 // Data pin`
2. `define NUM_PIXELS 1 // One LED`

```
Adafruit_NeoPixel WS2812B(NUM_PIXELS, PIN_WS2812B, NEO_GRB + NEO_KHZ800);
```

Wi-Fi Module Testing

The test scans local networks and displays the SSID, signal strength (RSSI), channel, and encryption type in the console.

```
WiFi.mode(WIFI_STA); // Station mode
WiFi.disconnect(); // Clear previous connections
numNetworks = WiFi.scanNetworks();
```

```
*****
Scan done
5 networks found
Nr | SSID | RSSI | CH | Encryption
 1 | ZTE-SKOK | -56 | 5 | WPA+WPA2
 2 | NET_SKOK | -77 | 11 | WPA
 3 | Derkom52 | -81 | 5 | WPA+WPA2
 4 | MIK_ZAM | -90 | 7 | WPA2
 5 | TP-Link_Extender | -93 | 2 | WPA2
```

Arduino Test Program Code

The full test program blinks the LED, reads IMU and SHTC3 data, scans Wi-Fi, and cycles the RGB LED colors.

```
#include "WiFi.h"
#include <Wire.h>
#include <Adafruit_NeoPixel.h>
#include "Adafruit_SHTC3.h"
#include "ICM42670P.h"

#define LED_BUILTIN 7
#define I2C_SDA 10
#define I2C_SCL 8
#define PIN_WS2812B 2
#define NUM_PIXELS 1

Adafruit_NeoPixel WS2812B(NUM_PIXELS, PIN_WS2812B, NEO_GRB + NEO_KHZ800);
ICM42670 IMU(Wire,0);
Adafruit_SHTC3 shtc3 = Adafruit_SHTC3();

void setup() {
  WS2812B.begin();
  WS2812B.setBrightness(50);
  pinMode(LED_BUILTIN, OUTPUT);
  Serial.begin(115200);
  delay(2000);
  while(!Serial);
  Wire.begin(I2C_SDA, I2C_SCL);
  IMU.begin();
  IMU.startAccel(100, 16);
  IMU.startGyro(100, 2000);
  shtc3.begin();
  WiFi.mode(WIFI_STA);
  WiFi.disconnect();
  delay(100);
}

void loop() {
  int numNetworks;
  sensors_event_t humidity, temp;
  inv_imu_sensor_event_t imu_event;
  digitalWrite(LED_BUILTIN, HIGH);
  Serial.println("\n*****");
  if (IMU.getDataFromRegisters(imu_event) == 0) {
    Serial.print("Accel X: "); Serial.println(imu_event.accel[0] / 2048.0);
    Serial.print("Gyro X: "); Serial.println(imu_event.gyro[0] / 16.4);
  }
  shtc3.getEvent(&humidity, &temp);
  Serial.print("SHTC3 Temp: "); Serial.println(temp.temperature);
  numNetworks = WiFi.scanNetworks();
  Serial.print(numNetworks); Serial.println(" networks found.");
  digitalWrite(LED_BUILTIN, LOW);
  WS2812B.setPixelColor(0, WS2812B.Color(255, 0, 0)); WS2812B.show(); delay(500);
  WS2812B.setPixelColor(0, WS2812B.Color(0, 255, 0)); WS2812B.show(); delay(500);
  WS2812B.setPixelColor(0, WS2812B.Color(0, 0, 255)); WS2812B.show(); delay(500);
  WS2812B.clear(); WS2812B.show();
}
```

Rust Programming Setup and Test Programs

This section describes how to configure the environment on Linux Ubuntu (24.04.2 LTS) to develop for ESP32-C3 in Rust.

Step 1: Install Prerequisites

Install essential libraries, Clang compiler, and Python tools:

```
sudo apt install --yes clang curl git libssl-dev libudev-dev=251.4-1ubuntu7  
pkg-config python3-pip
```

Step 2: Install Rust and Cargo Tools

```
curl --proto '=https' --tlsv1.2 --fail --show-error --silent https://sh.rustup.rs | sh -s -- -y  
source "$HOME/.cargo/env"
```

Install Cargo modules for ESP development:

- `espflash` for flashing memory
- `ldproxy` for linker arguments
- `cargo-generate` for project templates

```
cargo install espflash ldproxy cargo-generate
```

Step 3: Creating a Project

Run `cargo generate esp-rs/esp-idf-template cargo``. Select MCU `ESP32C3`` and ESP-IDF version `v5.3``.

```
tom-apple@tom-apple-Lenovo-ideapad-100-15IBD:~$ cargo generate --git https://github.com/esp-rs/esp-idf-template cargo
👑 Project Name: test_modulu_ESP32C3
⚠️ Renaming project called 'test_modulu_ESP32C3' to 'test-modulu-esp32c3'...
🔧 Destination: /home/tom-apple/test-modulu-esp32c3 ...
🔧 project-name: test-modulu-esp32c3 ...
🔧 Generating template ...
👑 Which MCU to target? · esp32c3
👑 Configure advanced template options? · true
👑 ESP-IDF version (master = UNSTABLE) · v5.3
👑 Configure project to use Dev Containers (VS Code and GitHub Codespaces)? · false
👑 Configure project to support Wokwi simulation with Wokwi VS Code extension? · false
👑 Add CI files for GitHub Action? · false
[ 1/11] Done: .cargo/config.toml
[ 2/11] Done: .cargo
[ 3/11] Done: .gitignore
[ 4/11] Done: .vscode
[ 5/11] Done: Cargo.toml
[ 6/11] Done: build.rs
[ 7/11] Ignored: pre-script.rhai
[ 8/11] Done: rust-toolchain.toml
[ 9/11] Done: sdkconfig.defaults
[10/11] Done: src/main.rs
[11/11] Done: src
🔧 Moving generated files into: `/home/tom-apple/test-modulu-esp32c3'...
🔧 Initializing a fresh Git repository
🌟 Done! New project created /home/tom-apple/test-modulu-esp32c3
```

To compile and run: `cargo run``.

Rust Example: Blinking LED

```
use esp_idf_svc::hal::{delay::FreeRtos, gpio::PinDriver, peripherals::Peripherals};

fn main() {
    esp_idf_svc::sys::link_patches();
    let peripherals = Peripherals::take().expect("Failed to take peripherals");
    let mut led = PinDriver::output(peripherals.pins.gpio7).expect("Failed to create led
driver");
    loop {
        led.toggle().expect("Failed to toggle LED");
        FreeRtos::delay_ms(500);
    }
}
```

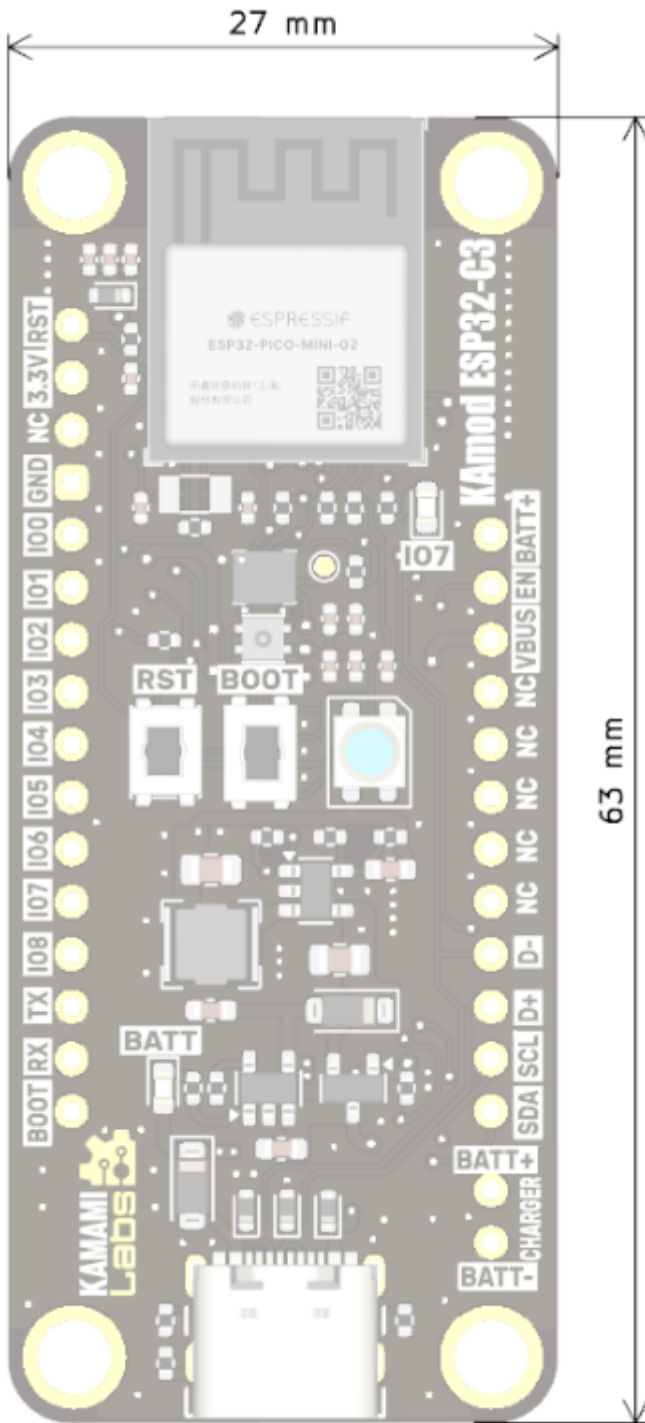
Rust Example: SHTC3 and ICM42670 Data

After running `cargo run`, the terminal will display sensor data every second.

```
SHTC3:
- temp: 26.056
- hum: 40.002
ICM42670:
- temp: 26.1875
- accel: -0.296875, 0.09375, 1.0717773
- gyro: -0.06097561, -10.243902, 21.341463
SHTC3:
- temp: 26.045
- hum: 40.069
ICM42670:
- temp: 26.21875
- accel: -0.30908203, 0.24023438, 0.91552734
- gyro: -8.353659, -13.658537, 7.4390244
SHTC3:
- temp: 26.01
- hum: 40.107
ICM42670:
- temp: 26.1875
- accel: 0.15966797, -0.36621094, 0.8330078
- gyro: -5.8536587, 66.95122, 43.719513
SHTC3:
- temp: 25.944
- hum: 40.089
ICM42670:
- temp: 26.28125
- accel: 0.11621094, -0.053222656, 1.0834961
- gyro: -4.634146, 4.085366, -9.756098
```

Dimensions

The KAMod ESP32-C3 board dimensions are 63 x 27 mm.



External Links

- [ESP32-C3 Documentation](#)
- [SHTC3 Sensor Datasheet](#)
- [ICM42670 Documentation](#)
- [Learn Rust](#)
- [Cargo Documentation](#)



Zastrzegamy prawo do wprowadzania zmian bez uprzedzenia.

Oferowane przez nas płytki drukowane mogą się różnić od prezentowanej w dokumentacji, przy czym zmianom nie ulegają jej właściwości użytkowe.

BTC Korporacja gwarantuje zgodność produktu ze specyfikacją.

BTC Korporacja nie ponosi odpowiedzialności za jakiegokolwiek szkody powstałe bezpośrednio lub pośrednio w wyniku użycia lub nieprawidłowego działania produktu.

BTC Korporacja zastrzega sobie prawo do modyfikacji niniejszej dokumentacji bez uprzedzenia.