KAMAMI

KAmod ESP32-C3 (PL)

Rev. 20260127080709
Źródło: https://wiki.kamamilabs.com/index.php?title=KAmod_ESP32-C3_(PL)

Spis treści

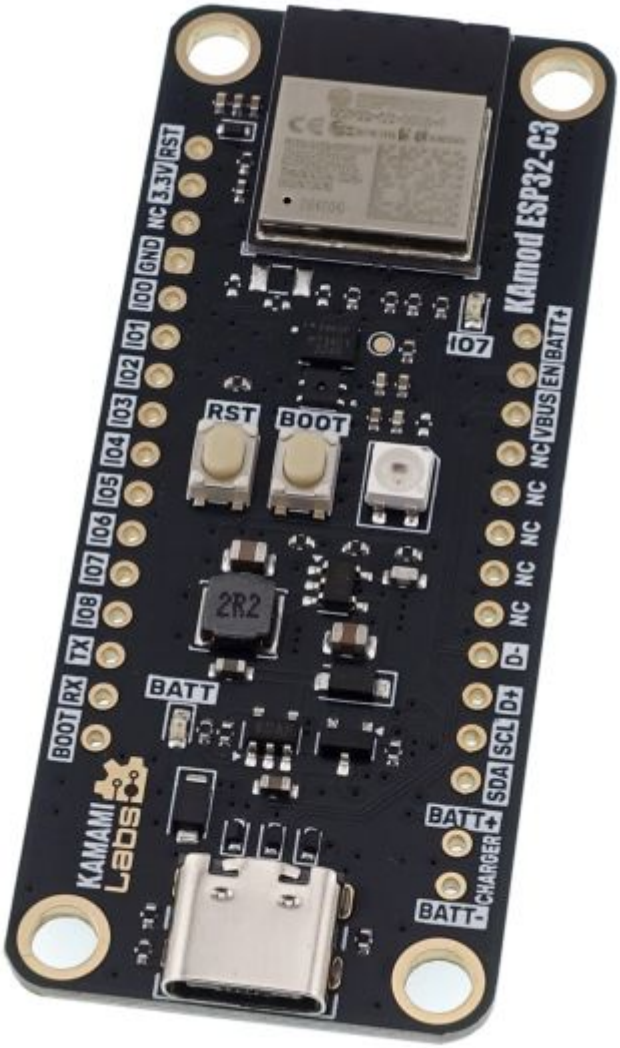
Opis

KAmod ESP32-C3 - Płytka rozwojowa z układem ESP32-C3 Mini-1
Na płytce KAmod ESP32-C3 znajduje się moduł ESP32-C3 Mini-1 firmy Espressif, który zawiera 32-bitowy, jednordzeniowy mikrokontroler SoC o architekturze RISC-V oraz interfejsy radiowe Wi-Fi i Bluetooth 5 (LE) z obsługą trybu dalekiego zasięgu (LR).

Częstotliwość taktowania MCU wynosi maksymalnie 160 MHz i jest wyposażony w aż 400 kB pamięci RAM i 4 MB pamięci Flash. Obsługuje tryby niskiego zużycia energii, działa w temperaturach od -40 do +85□C, realizuje funkcje bezpiecznego rozruchu Secure Boot oraz szyfrowanie flash z AES-128/256-XTS, itd. Dzięki tak rozbudowanej specyfikacji doskonale nadaje się do zastosowań przemysłowych oraz z zakresu IoT. Ponadto, na płytce modułu znajdują się: dokładny czujnik temperatury i wilgotności SHTC3, czujnik MEMS typu ICM42670 zawierający 3-osiowy żyroskop i 3-osiowy akcelerometr, 3-kolorowa dioda typu WS28212, oraz dioda LED podłączona do jednego z portów mikrokontrolera. Komponenty te ułatwią budowę wielu różnych aplikacji.

Moduł jest zasilany napięciem +5V dostarczanym przez złącze USB-C, lub napięciem z akumulatora Li-Ion. Napięcie +3,3 V zasilające mikrokontroler i układy peryferyjne wytwarza przetwornica Buck DC/DC typu SY8088 o napięciu wejściowym z zakresu 2.5….5 V. Po zaniku napięcia ze złącza USB-C zasilanie automatycznie dostarczane jest z akumulatora (jeżeli jest podłączony). Układ zasilania jest uzupełniony o ładowarkę akumulatora Li-Ion na bazie układu MCP7381, zasilanej ze złącza USB-C.

Moduł KAmod ESP32-C3 jest przeznaczony do uruchamiania i testowania aplikacji w środowisku Arduino z użyciem języka C/C++ oraz Rust.


KAmod ESP32-C3
RST
3.3V
GND
IO0
IO1
IO2
IO3
IO4
IO5
IO6
IO7
IO8
TX
RX
BOOT
BATT
2R2
IO7
EN
VBUS
NC
D-
D+
SCL
SDA
BATT+
CHARGER
BATT-
KAMAMI Labs

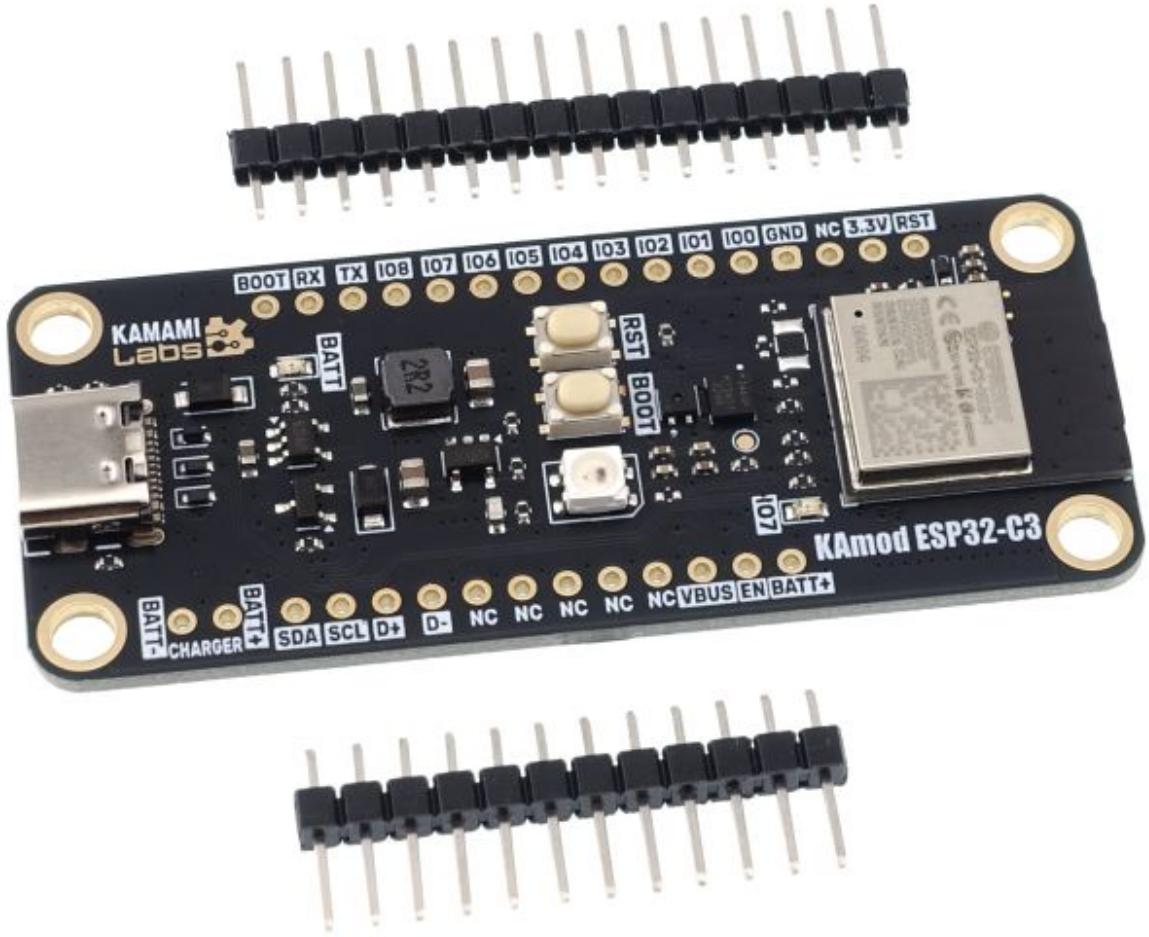
Podstawowe cechy i parametry

- Moduł ESP32C3 Mini-1 (Wi-Fi 802.11 b/g/n, Bluetooth 5 LE)
- Mikrokontroler 32-bit RISC-V jednordzeniowy 160 MHz, 400 kB SRAM, 4 MB Flash
- 15 linii GPIO
- interfejsy komunikacyjne SPI, I2C, I2S, UART, USB
- przetwornik ADC 12-bit SAR do 6 kanałów
- Akcelerometr ICM42670
    - Trójosiowy żyroskop MEMS - czujniki prędkości kątowej osi X, Y i Z
    - Trójosiowy akcelerometr MEMS osi X, Y i Z
    - Interfejs komunikacyjny: I2C
- Termometr/higrometr SHTC3
    - Zakres pomiaru wilgotności 0...100%RH z dokładnością +/- 2%
    - Zakres pomiaru temperatury -40 do +125 °C. Dokładność pomiaru +/-0.2°C w zakresie od 0°C do +60°C
    - Interfejs komunikacyjny I2C
- Przetwornica DC/DC typu SY8088
- Układ ładowarki baterii Li-Ion typu MCP73831
- Programowana dioda WS2812
- Złącze USB-C
    - Zasilanie modułu napięciem +5V

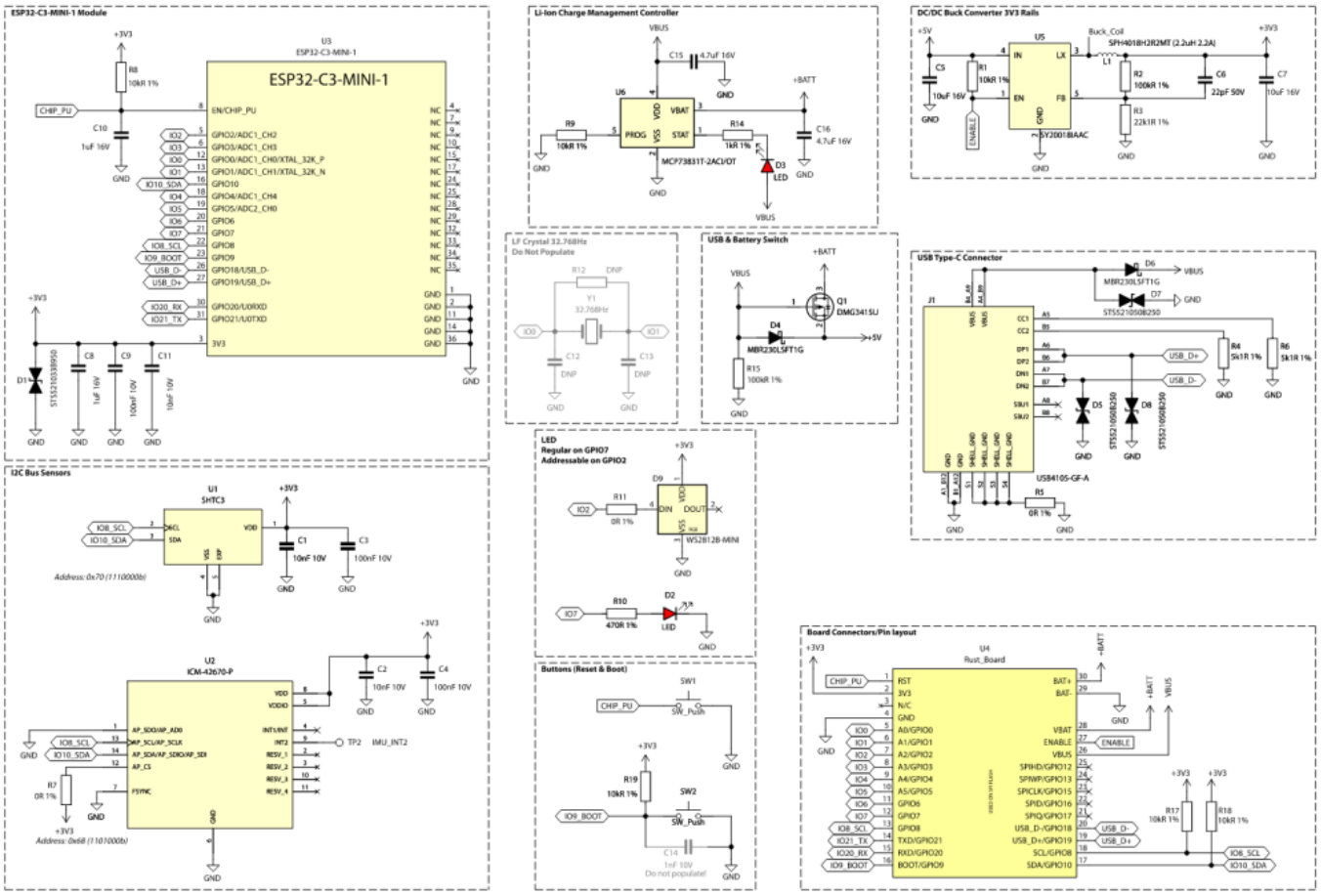
    - Interfejs programujący pamięć Flash mikrokontrolera
    - Interfejs JTAG
- Przyciski Reset i Boot

Wyposażenie standardowe

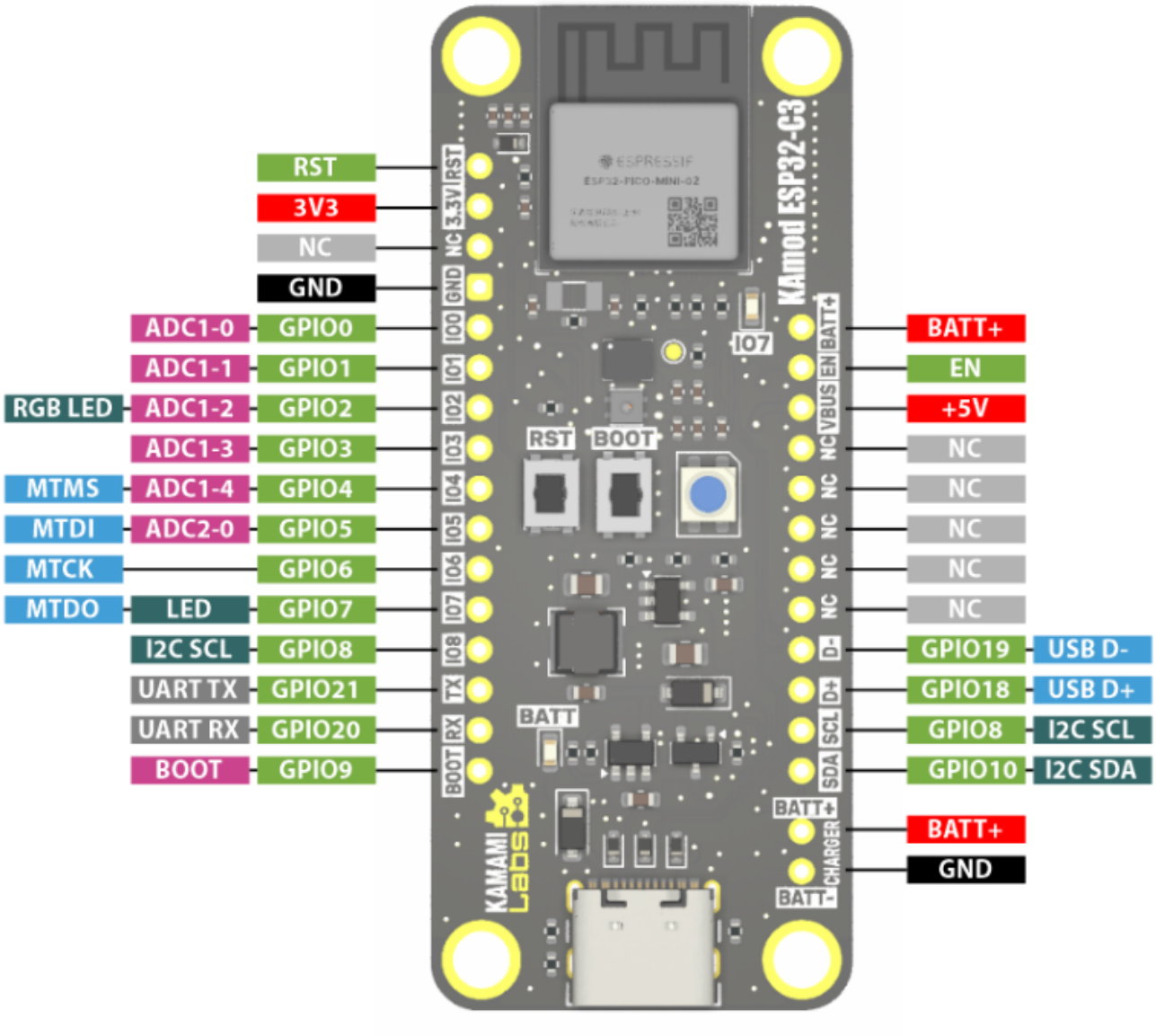
Kod	Opis
KAmod ESP32-C3	• Zmontowany i uruchomiony moduł • 1 x prosta listwa goldpin 12-pin raster 2,54 mm • 1 x prosta listwa goldpin 16-pin raster 2,54 mm



Schemat elektryczny

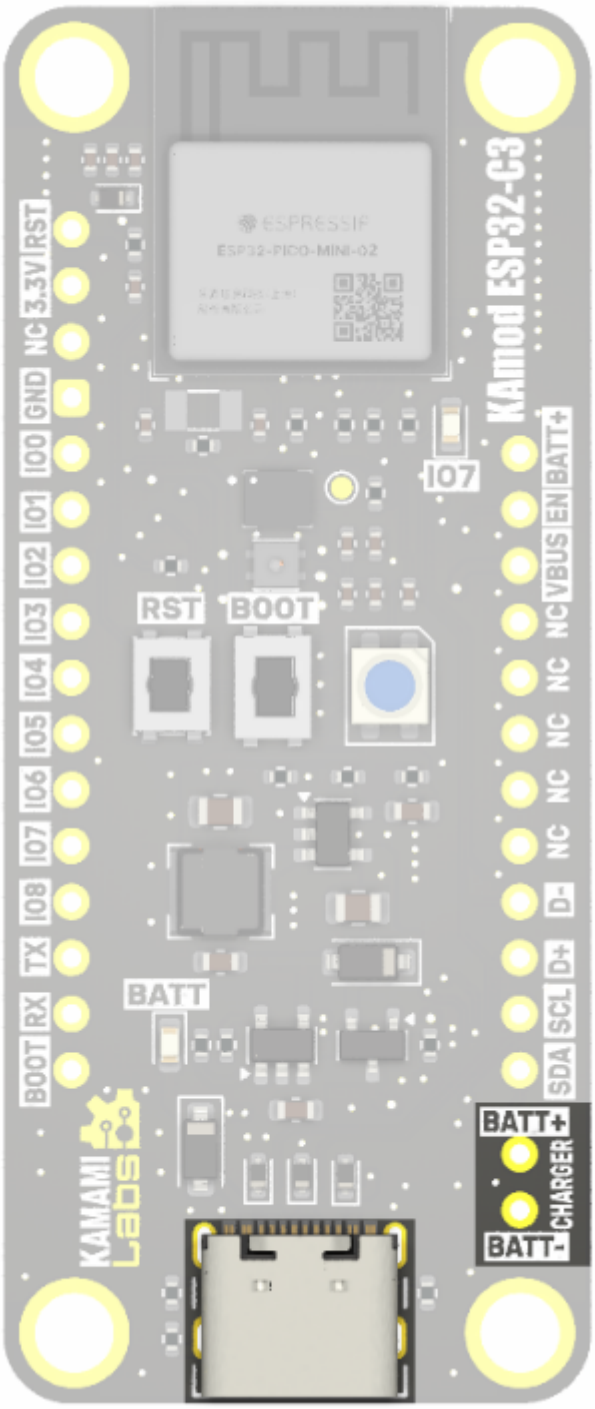


Opis wyprowadzeń



Zasilanie układu

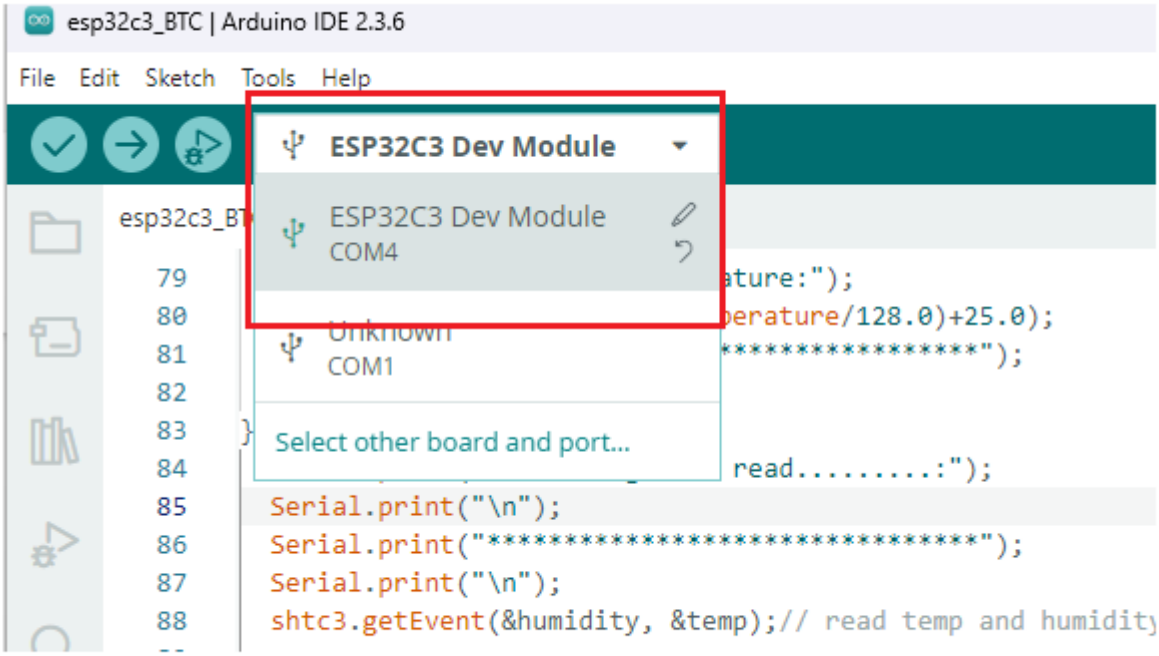
Moduł jest zasilany napięciem VBUS o wartości +5 V ze złącza USB-C. To napięcie służy również do ładowania akumulatora Li-Ion 4,2 V, jeśli jest podłączony do wyjścia Charger. Za proces ładowania odpowiada układ MCP7381 realizujący algorytm ładowania o stałym natężeniu prądu/stałym napięciu, gdzie prąd ładowania jest ustawiony na 100 mA, a napięcie końcowe na 4,20 V.



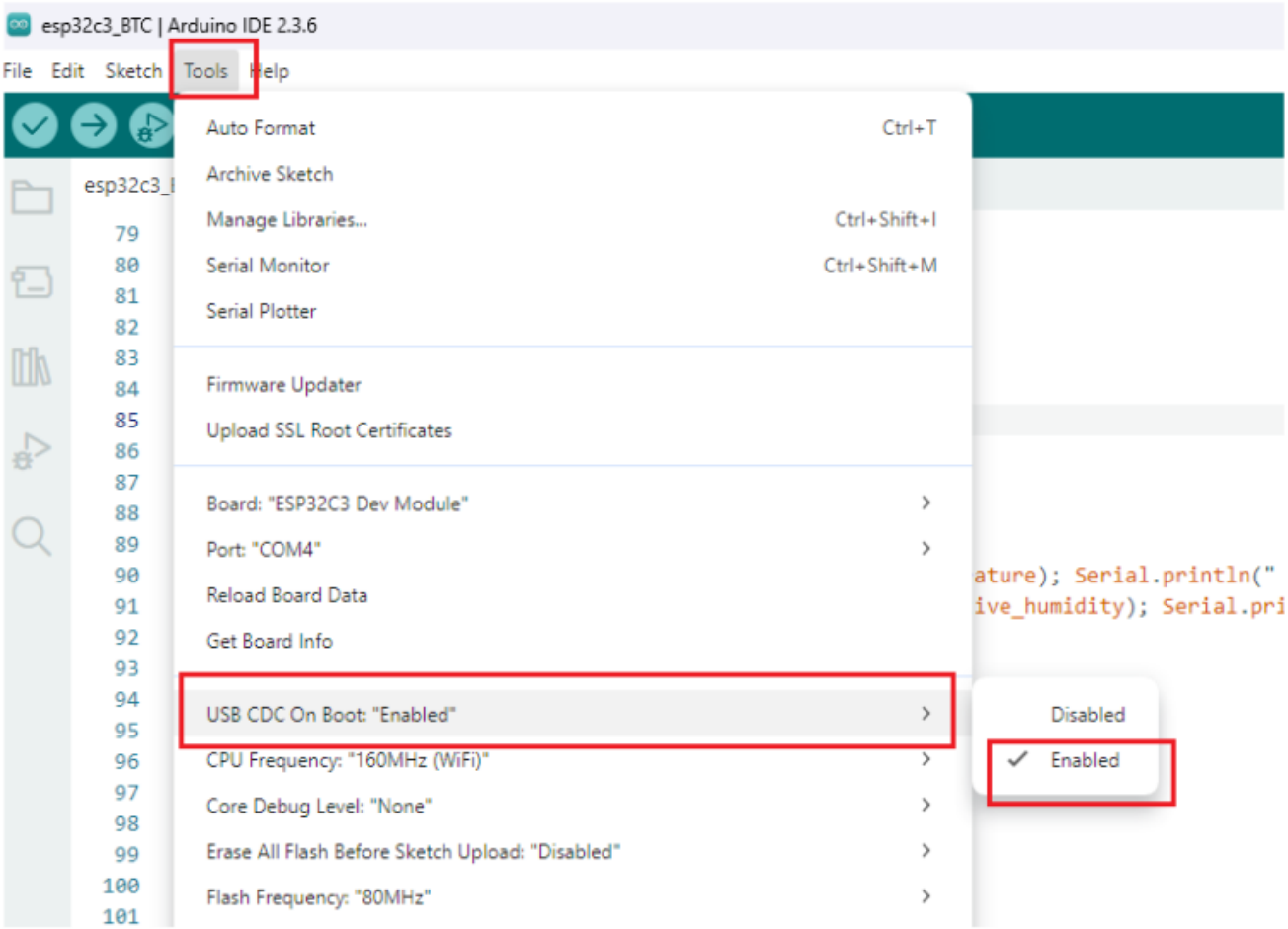
Konfiguracja środowiska Arduino i program testowy

Czynności wstępne

Podłączamy moduł Kamod ESP-C3 do komputera z zainstalowanym środowiskiem Arduino IDE. W oknie wyboru modułu wybieramy moduł *ESP32C3 Dev Module* i wirtualny port szeregowy COMx, poprzez który jest połączony moduł.

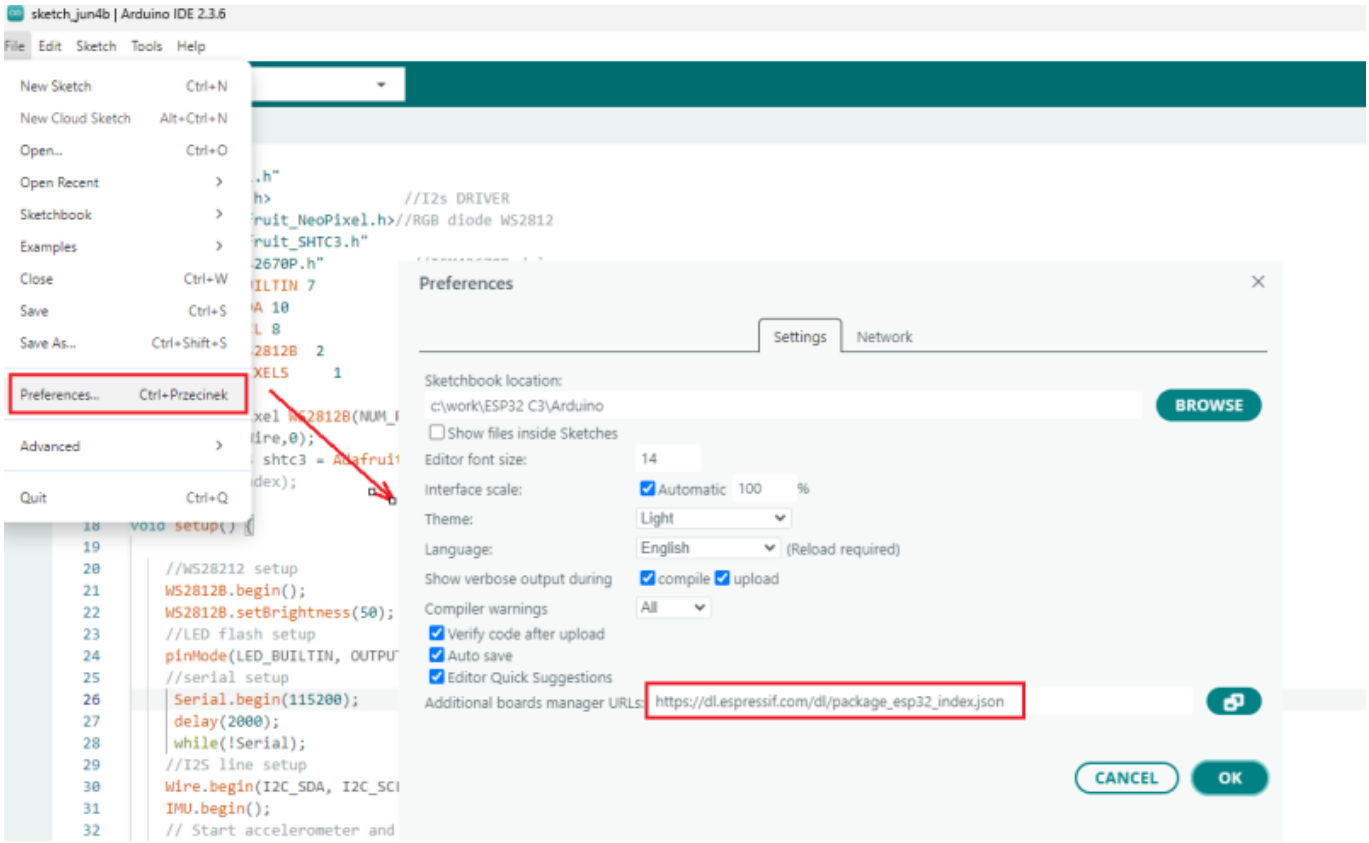


W naszej procedurze testowej będziemy używali okna terminala, w którym będą wyświetlane wyniki działania programu. Żeby to było możliwe trzeba odblokować domyślnie zablokowaną opcję *USB CDC On Boot*


esp32c3_BTC | Arduino IDE 2.3.6
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter
Firmware Updater
Upload SSL Root Certificates
Board: "ESP32C3 Dev Module"
Port: "COM4"
Reload Board Data
Get Board Info
USB CDC On Boot: "Enabled"
CPU Frequency: "160MHz (WiFi)"
Core Debug Level: "None"
Erase All Flash Before Sketch Upload: "Disabled"
Flash Frequency: "80MHz"
Disabled
Enabled

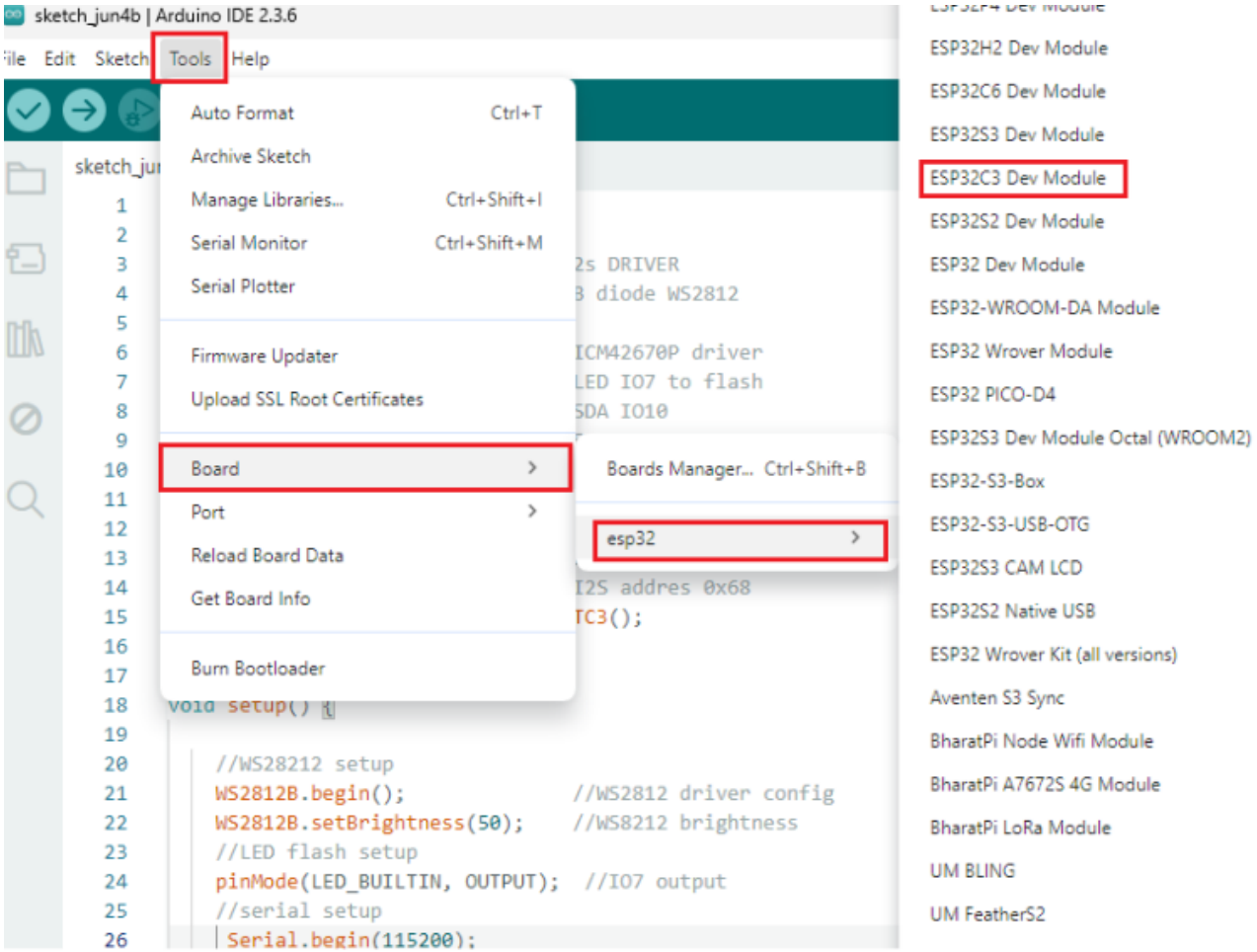
Konfiguracja okna preferences

Otwieramy okno preferencji *File->Preferences*. W polu *Additional boards manager ULRs* zakładki *Settings* wpisujemy adres https://dl.espressif.com/dl/package_esp32_index.json



Wybór typu procesora dla programowanego modułu

Wybieramy kolejno zakładkę *Tools*, następnie *Board*, rodzinę procesorów *esp32* i moduł z procesorem *ESP32C3*. Każda zmiana modułu podłączonego do komputera przez USB będzie wymagała powtórzenia tego kroku. Jeżeli nie wykonamy go na początku, to można go wykonać przed kompilacją. W przeciwnym razie projekt nie zostanie prawidłowo skompilowany.

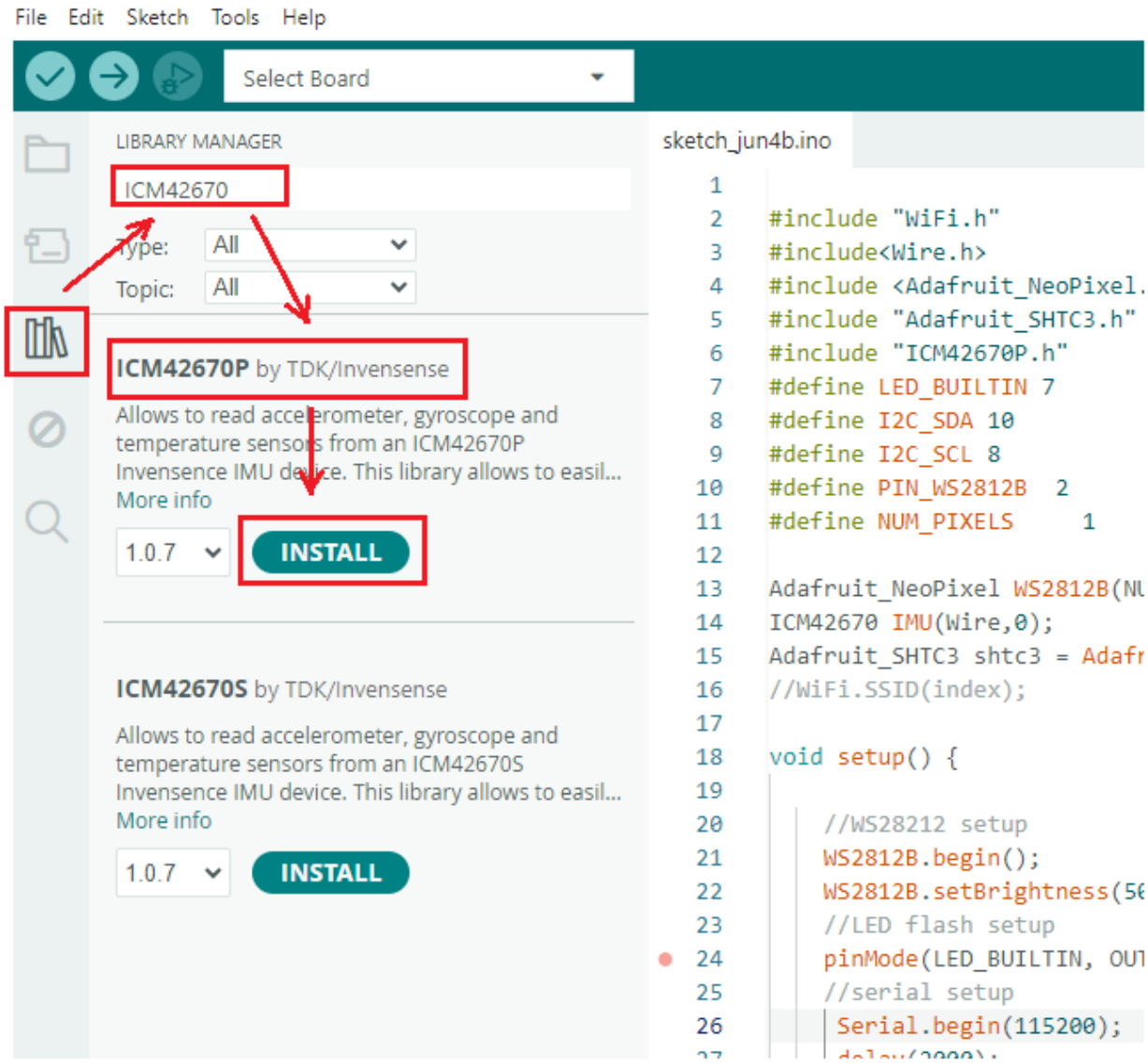


Instalowanie bibliotek

Program testowy wymaga zainstalowania bibliotek obsługujących układy peryferyjne: akcelerometr ICM42670P, termometr/higrometr SHTC3 i 3-kolorową diodę LED typu WS2812B.

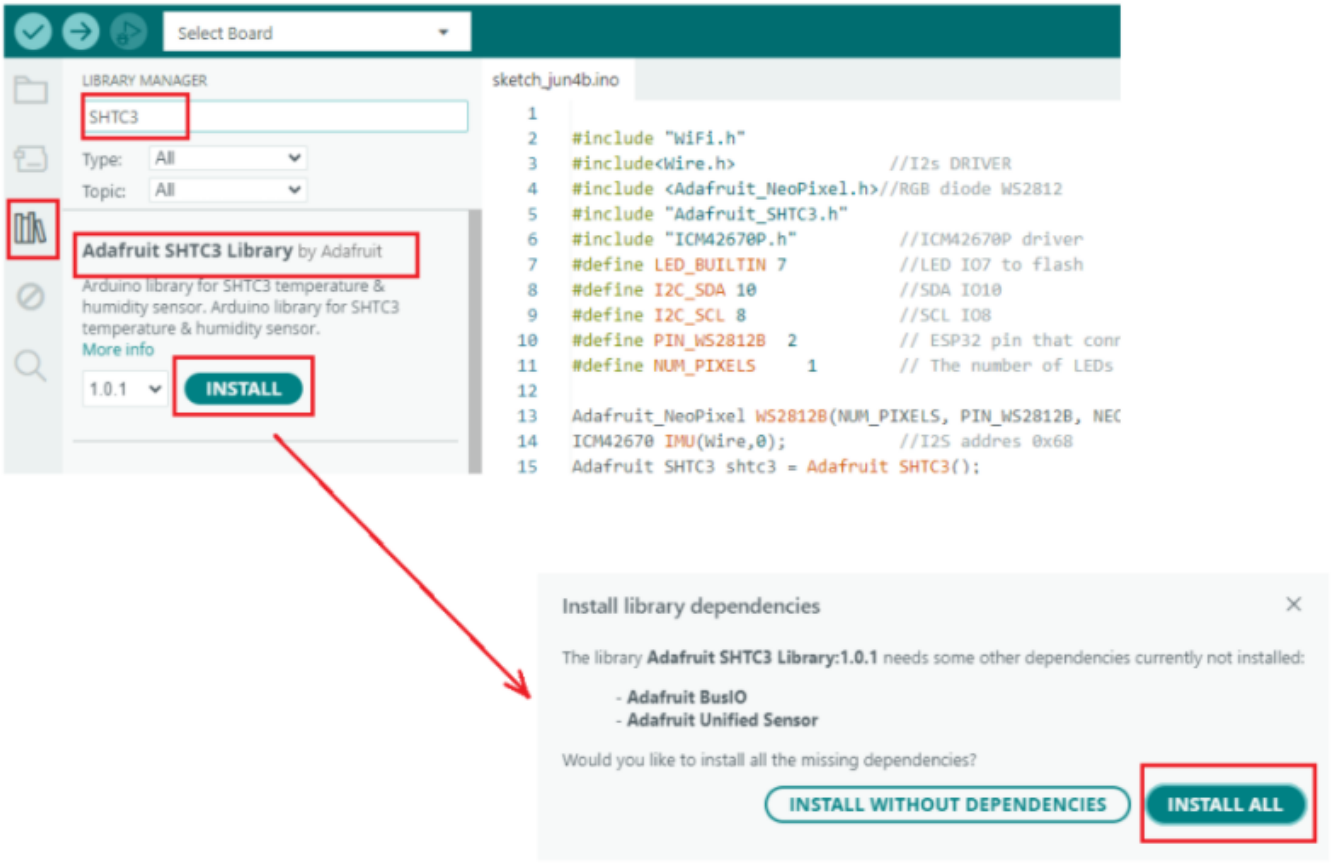
Instalowanie biblioteki do obsługi układu ICM42670P

Klikamy na pionowym pasku narzędzi ikonę bibliotek. W oknie wyszukiwanie wpisujemy ICM42670, wybieramy *ICM42670P by TDK/Invensense* i klikamy *Install*.



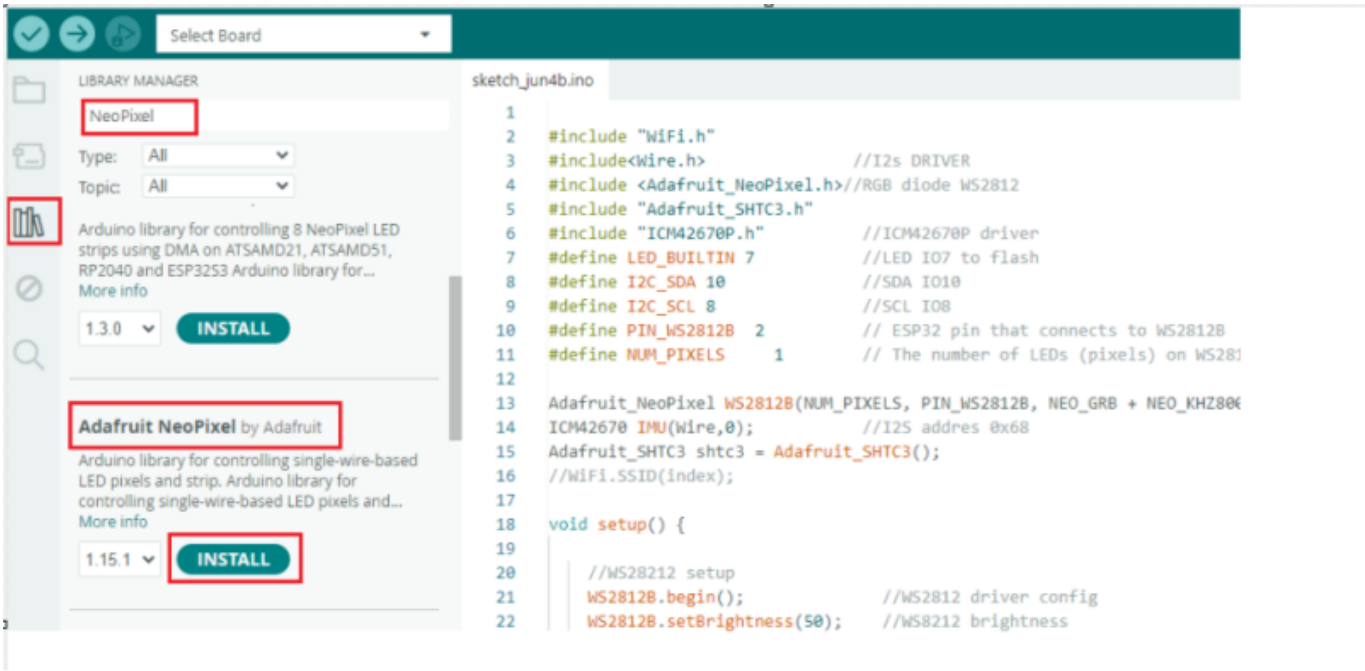
Instalowanie biblioteki do obsługi układu SHTC3

Wybieramy bibliotekę *Adafruit SHTC3 Library* dostarczaną przez Adafruit. Wymaga ona doinstalowania powiązanych bibliotek - aby to zrobić w wyskakującym dodatkowym oknie *Install library dependencies* klikamy *Install All*.



Instalowanie biblioteki obsługującej 3-kolorową diodę RGB typu WS2812

Do sterowania 3-kolorowej diody RGB zastosujemy bibliotekę dostarczana przez Adafruit - *Adafruit NeoPixel*



Obsługa akcelerometru ICM42670P

ICM42670P jest połączony z mikrokontrolerem za pomocą magistrali I2C. Do jej obsługi jest przeznaczona biblioteka Wire. Do inicjowania interfejsu I2S jest używana metoda begin z argumentami określającymi linie portów przypisane do sygnałów SDA i SCL

```
#define I2C_SDA 10              //SDA IO10
#define I2C_SCL 8               //SCL IO8
Wire.begin(I2C_SDA, I2C_SCL);
```

Obsługa termometru/higrometru SHTC3

Układ SHTC3 łączy się z mikrokontrolerem również za pomocą interfejsu I2C. Konfiguracja i inicjacja tego interfejsu została pokazana powyżej. Inicjalizacja biblioteki układu SHTC3 jest wykonywana standardowo przez metodę begin.

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

Odczytywanie i konwersję wartości temperatury wykonuje metoda htc3.getEvent

```
htc3.getEvent(&humidity, &temp); // read temp and humidity
```

Dane wyjściowe są umieszczane w zmiennych temp.temperature i humidity.relative_humidity i można je bezpośrednio wyświetlić.

Przykładowy wydruk odczytanych danych z czujnika SHTC3 na konsoli znakowej Arduino IDE

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

Obsługa diody RGB WS2812B

Tworzymy instancję WS2812B, w której określamy: ilość diod WS2812 połączonych w łańcuchu (NUM_PIXELS), oraz numer portu linii danych (PIN_WS2812B). Pozostałe parametry można zostawić domyślnie. U nas jest jedna dioda (NUM_PIXELS = 1) podłączona do linii poru DO2 (PIN_WS2812B =2)

```
#define PIN_WS2812B  2          // ESP32 pin that connects to WS2812B
#define NUM_PIXELS     1        // The number of LEDs (pixels) on WS2812B
Adafruit_NeoPixel WS2812B(NUM_PIXELS, PIN_WS2812B, NEO_GRB + NEO_KHZ800);
```

Inicjalizacja drivera jest standardowo wykonywana przez metodę begin:

```
WS2812B.begin();              //WS2812 driver config
```

Metoda setBrightness służy do ustawiania jasności świecenia wszystkich diod

```
WS2812B.setBrightness(50);    //WS8212 brightness
```

Metoda clear wygasza wszystkie diody, a metoda setPixelColor określa składowe koloru dla każdej z diod.

```
WS2812B.setPixelColor(0, WS2812B.Color(255, 0, 0)); //kolor czerwony
```

Skutki działania metod clear i setPixelColor są widoczne po wywołaniu metody show:

```
WS2812B.show();
```

Sprawdzanie działania modułu WiFi

Procedura testowa sprawdzająca działanie modułu WiFi polega na skanowaniu radiowej sieci WiFi i wyświetlaniu znalezionych identyfikatorów SSID lokalnych sieci wraz z poziomem sygnału radiowego, kanałem radiowym oraz rodzajami szyfrowania danych.
Najpierw jest ustawiany tryb WIFI_STA, czyli tryb stacji. W tym trybie moduł ESP32 może się łączyć z sieciami WIFI. Po połączeniu z ruterem moduł może żądać informacji z Internetu (jeżeli ruter jest połączony z Internetem), lub z urządzeń z lokalnej sieci rutera. Po ustawieniu tego trybu wykonujemy ewentualne rozłączenie z siecią (jeżeli moduł był wcześniej połączony):

```
WiFi.mode(WIFI_STA); //station mode

WiFi.disconnect();         //disconnect WIFI network
```

Możemy teraz skanować identyfikatory sieci WIFI za pomocą metody scanNetworks
Metoda zwraca ilość wykrytych sieci wspomniane już parametry: poziom sygnału radiowego, kanał radiowy oraz rodzaj szyfrowania danych.

```
numNetworks = WiFi.scanNetworks(); //scan WIFI networks </nowiki>
  Serial.println("Scan done");
  if (numNetworks == 0)
  {
    Serial.println("no networks found");
  }
```

Po wykonaniu skanowania można wyświetlić odczytane dane w konsoli i wykorzystać do w naszej aplikacji na przykład w celu połączenia z wybraną siecią.

```
for (int i = 0; i < numNetworks; ++i) {
      // Print SSID and RSSI for each network found
      Serial.printf("%2d", i + 1);
      Serial.print(" | ");
      Serial.printf("%-32.32s", WiFi.SSID(i).c_str());
      Serial.print(" | ");
      Serial.printf("%4ld", WiFi.RSSI(i));
      Serial.print(" | ");
      Serial.printf("%2ld", WiFi.channel(i));
      Serial.print(" | ");
      switch (WiFi.encryptionType(i)) {
        case WIFI_AUTH_OPEN:            Serial.print("open"); break;
        case WIFI_AUTH_WEP:             Serial.print("WEP"); break;
        case WIFI_AUTH_WPA_PSK:         Serial.print("WPA"); break;
        case WIFI_AUTH_WPA2_PSK:        Serial.print("WPA2"); break;
        case WIFI_AUTH_WPA_WPA2_PSK:    Serial.print("WPA+WPA2"); break;
        case WIFI_AUTH_WPA2_ENTERPRISE: Serial.print("WPA2-EAP"); break;
        case WIFI_AUTH_WPA3_PSK:        Serial.print("WPA3"); break;
        case WIFI_AUTH_WPA2_WPA3_PSK:   Serial.print("WPA2+WPA3"); break;
        case WIFI_AUTH_WAPI_PSK:        Serial.print("WAPI"); break;
        default:                        Serial.print("unknown");
      }
      Serial.println();
      delay(10);
    }
  }
```

Przykładowy wydruk działania programu skanującego sieć WIFI na konsoli znakowej Arduino IDE

```
*********************************
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
```

Po wykonaniu skanowania i wyświetleniu danych trzeba wyczyścić dane, żeby przygotować moduł WiFi do kolejnego skanowania – metoda scanDelete

```
WiFi.scanDelete();
```

Program testowy

Program testowy ma zadanie przetestować wszystkie komponenty umieszczone na płytce modułu. Test składa się z kolejnych kroków:

- Zapaleniu diody czerwonej LED podłączonej do linii portu IO7
- Odczytaniu i wyświetleniu w konsoli Arduino IDE odczytanych danych z ICM42670P
- Odczytaniu i wyświetleniu w konsoli Arduino IDE odczytanych danych z SHTC3
- Skanowaniu sieci WIFI i wyświetleniu w konsoli Arduino IDE identyfikatorów SSID siłę sygnału radiowego, numer kanału WIFI i rodzaj kodowania danych w sieci dla każdej ze znalezionych sieci
- Zgaszeniu czerwonej diody LED
- Zapaleniu co 0,5 sekundy kolejnych diod WS2812B czerwonej, zielonej i niebieskiej
- Zgaszeniu wszystkich diod WS2812B i rozpoczęciu testu od nowa

Poniżej pokazany jest fragment ekranu konsoli znakowej Arduino IDE wyświetlający jeden przebieg programu testowego.

```
*******************************
ICM42670P AccelX:0.82
ICM42670P AccelY:0.50
ICM42670P AccelZ:0.29
*******************************
ICM42670P GyroX:-1.04
ICM42670P GyroY:-0.06
ICM42670P GyroZ:-0.37
*******************************
ICM42670PTemperature:32.00
*******************************
SHTC3P register read.........:
*******************************
SHTC3 Temperature: 31.54 deg C
SHTC3 Humidity: 46.78% rH
*******************************
Scan done
4 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 | TP-Link_Extender                 |  -94 |  2 | WPA2
```

Kod programu testowego znajduje się poniżej, można go skompilować w środowisku Arduino.

```
#include "WiFi.h"
#include<Wire.h>               //I2s DRIVER
#include <Adafruit_NeoPixel.h>//RGB diode WS2812
#include "Adafruit_SHTC3.h"
#include "ICM42670P.h"          //ICM42670P driver

#define LED_BUILTIN 7           //LED IO7 to flash

#define I2C_SDA 10              //SDA IO10
#define I2C_SCL 8               //SCL IO8
#define PIN_WS2812B  2          // ESP32 pin that connects to WS2812B
```

```
#define NUM_PIXELS     1        // The number of LEDs (pixels) on WS2812B

Adafruit_NeoPixel WS2812B(NUM_PIXELS, PIN_WS2812B, NEO_GRB + NEO_KHZ800);
ICM42670 IMU(Wire,0);           //I2S addres 0x68
Adafruit_SHTC3 shtc3 = Adafruit_SHTC3();


void setup() {

    //WS28212 setup
    WS2812B.begin();              //WS2812 driver config
    WS2812B.setBrightness(50);    //WS8212 brightness
    //LED flash setup
    pinMode(LED_BUILTIN, OUTPUT);  //IO7 output
    //serial setup
     Serial.begin(115200);
     delay(2000);
     while(!Serial);                 //wait to serial ready
    //I2S line setup
    Wire.begin(I2C_SDA, I2C_SCL);                    //setup I2S driver
    IMU.begin();                                      //setup ICM42670P driver
    // Start accelerometer and gyroscope
    IMU.startAccel(100, 16); // 100 Hz, ±16g
    IMU.startGyro(100, 2000); // 100 Hz, ±2000 dps
    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);
  //ICM42670P register read, convert and
  Serial.print("\n");
   Serial.print("*********************************");
   Serial.print("\n");
  int ret = IMU.getDataFromRegisters(imu_event);   //ICM42670P register read
  Serial.println(ret);
  if (ret == 0) {
// Convert acceleration to g
  float accelX = imu_event.accel[0] / 2048.0;
  float accelY = imu_event.accel[1] / 2048.0;
  float accelZ = imu_event.accel[2] / 2048.0;

// Convert gyroscope to dps
  float gyroX = imu_event.gyro[0] / 16.4;
  float gyroY = imu_event.gyro[1] / 16.4;
  float gyroZ = imu_event.gyro[2] / 16.4;
//print converted result
  Serial.print("ICM42670P AccelX:");
  Serial.println(accelX);
  Serial.print("ICM42670P AccelY:");
  Serial.println(accelY);
  Serial.print("ICM42670P AccelZ:");
  Serial.println(accelZ);
  Serial.print("*********************************");
```

```
  Serial.print("\n");
  Serial.print("ICM42670P GyroX:");
  Serial.println(gyroX);
  Serial.print("ICM42670P GyroY:");
  Serial.println(gyroY);
  Serial.print("ICM42670P GyroZ:");
  Serial.println(gyroZ);
  Serial.print("*********************************");
   Serial.print("\n");
  Serial.print("ICM42670PTemperature:");
  Serial.println((imu_event.temperature/128.0)+25.0);
  Serial.print("*********************************");
  Serial.print("\n");
}
  Serial.print("SHTC3P register read.........:");
  Serial.print("\n");
  Serial.print("*********************************");
  Serial.print("\n");
  shtc3.getEvent(&humidity, &temp);// read temp and humidity
  Serial.print("SHTC3 Temperature: "); Serial.print(temp.temperature); Serial.println(" deg
C");
  Serial.print("SHTC3 Humidity: "); Serial.print(humidity.relative_humidity);
Serial.println("% rH");
  Serial.print("*********************************");
  Serial.print("\n");
      numNetworks = WiFi.scanNetworks();
    Serial.println("Scan done");
  if (numNetworks == 0) {
    Serial.println("no networks found");
  } else {
    Serial.print(numNetworks);
    Serial.println(" networks found");
    Serial.println("Nr | SSID                                | RSSI | CH | Encryption");
    for (int i = 0; i < numNetworks; ++i) {
      // Print SSID and RSSI for each network found
      Serial.printf("%2d", i + 1);
      Serial.print(" | ");
      Serial.printf("%-32.32s", WiFi.SSID(i).c_str());
      Serial.print(" | ");
      Serial.printf("%4ld", WiFi.RSSI(i));
      Serial.print(" | ");
      Serial.printf("%2ld", WiFi.channel(i));
      Serial.print(" | ");
      switch (WiFi.encryptionType(i)) {
        case WIFI_AUTH_OPEN:            Serial.print("open"); break;
        case WIFI_AUTH_WEP:             Serial.print("WEP"); break;
        case WIFI_AUTH_WPA_PSK:         Serial.print("WPA"); break;
        case WIFI_AUTH_WPA2_PSK:        Serial.print("WPA2"); break;
        case WIFI_AUTH_WPA_WPA2_PSK:    Serial.print("WPA+WPA2"); break;
        case WIFI_AUTH_WPA2_ENTERPRISE: Serial.print("WPA2-EAP"); break;
        case WIFI_AUTH_WPA3_PSK:        Serial.print("WPA3"); break;
        case WIFI_AUTH_WPA2_WPA3_PSK:   Serial.print("WPA2+WPA3"); break;
        case WIFI_AUTH_WAPI_PSK:        Serial.print("WAPI"); break;
        default:                        Serial.print("unknown");
      }
      Serial.println();
      delay(10);
    }
  }
```

```
    WiFi.scanDelete();
    digitalWrite (LED_BUILTIN, LOW);
    WS2812B.clear();  // set all pixel colors to 'off'. It only takes effect if pixels.show()
is called
    WS2812B.setPixelColor(0, WS2812B.Color(255, 0, 0));  // it only takes effect if
pixels.show() is called
    WS2812B.show();
    delay(500);
    WS2812B.setPixelColor(0, WS2812B.Color(0, 255, 0));  // it only takes effect if
pixels.show() is called
    WS2812B.show();
    delay(500);
    WS2812B.setPixelColor(0, WS2812B.Color(0, 0, 255));  // it only takes effect if
pixels.show() is called
    WS2812B.show();
    delay(500);
     WS2812B.clear();  // set all pixel colors to 'off'. It only takes effect if pixels.show()
is called
     WS2812B.show();
    delay(100);  // 500ms pause between each pixel
  //}
}
```

Konfiguracja środowiska do programowania w języku Rust i programy testowe

Najpierw pokażemy, jak szybko pobrać i skonfigurować niezbędne komponenty i narzędzia programowe do wygenerowania prostego projektu w języku Rust. Po skonfigurowaniu narzędzi będzie można projekt skompilować i przesłać do pamięci flash. Wszystkie czynności są wykonywane w systemie operacyjnym Linux Ubuntu wersja 24.04.2 LTS. Komputer musi być podłączony do Internetu.

Włączenie repozytorium Universe

"Universe" to standardowe repozytorium dla Ubuntu. Repozytorium jest utrzymywane przez społeczność i zapewnia bezpłatne oprogramowanie typu open source. Domyślnie jest włączone w najnowszych wersjach Ubuntu. Jednak, gdyby tak nie było to możemy go włączyć za pomocą wiersza poleceń:

```
sudo add-apt- repository universe
```

Instalacja pakietów libssl-dev, libudev, kompilatora C/C++ clang i interpretera Python 3

Pakiet libssl-dev zawiera protokoły kryptograficzne SSL i TLS do bezpiecznej komunikacji przez Internet. Pakiet libudev udostępnia interfejs API do introspekcji i wyliczania urządzeń na poziomie lokalnym system. Clang jest kompilatorem języków programowania C, C++ i Objective-C i jest częścią projektu LLVM. Znany z szybkiego czasu kompilacji i doskonałej diagnostyki, Clang może również działać jako zamiennik GCC. W projekcie jest używany linker Clang. Python3-pip jest pakietem instalacyjnym języka Python ver3, a python3-venv instaluje środowisko wirtualne.

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

Instalacja pakietów Rust i menadżera Cargo

Rust to język programowania ogólnego przeznaczenia, kładący nacisk na wydajność, bezpieczeństwo typów i współbieżność. Cargo to menedżer pakietów Rust. Cargo pobiera zależności pakietu Rust, kompiluje pakiety, tworzy pakiety dystrybucyjne i przesyła je do crates.io, rejestru pakietów społeczności Rust.

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

Instalacja niezbędnych modułów Cargo:

- espflash potrzebny do programowania pamięci Flash mikrokontrolera
- ldproxy przekazanie argumentów konsolidatora
- cargo-generate do generowania projektów według szablonu

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

Tworzenie projektu

```
cargo generate --git https://github.com/esp-rs/esp-idf-template cargo
```

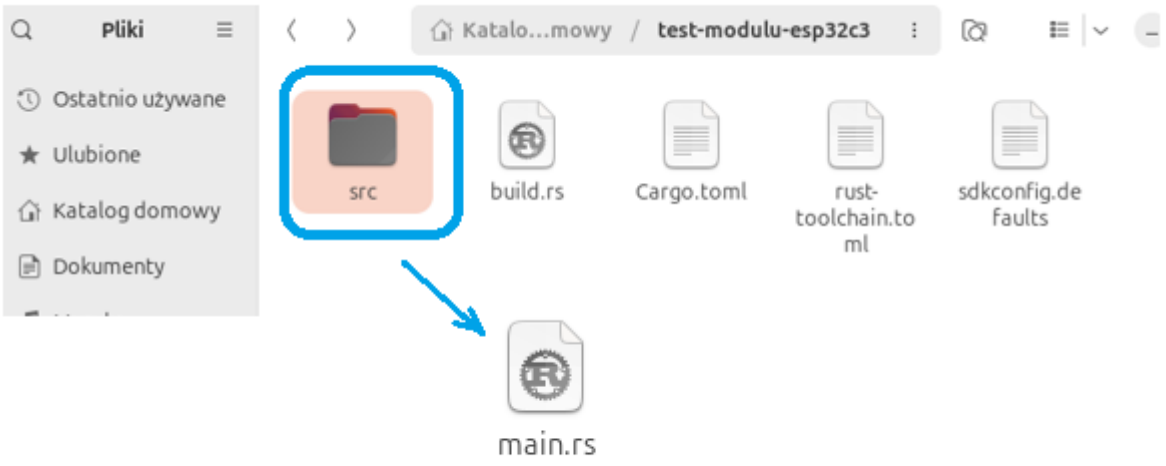
Wykonanie tego polecenia otwiera okno uproszczonego menadżera projektu. Menadżer poprosi nas o:

- wpisanie nazwy projektu na przykład test-modulu-esp32c3
- wybranie z listy typu MCU – w naszym przypadku będzie to ESP32C3
- wybranie wersji ESP-IDF wybieramy v5.3

Pozostałe opcje można wybrać tak jak na rysunku poniżej

```
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
```

Po wybraniu wszystkich opcji menadżer projektu utworzy katalog o nazwie projektu, w którym umieści niezbędne pliki. W katalogu src projektu menadżer umieści plik źródłowy main.rs ze szkieletem wczytanym poleceniem cargo generate.



W tym momencie mamy już wszystko, żeby wyedytowany plik main.rs kompilować, przesyłać swój program do pamięci flash i uruchamiać jego działanie. Wszystkie te czynności są wykonywane automatycznie po wykonaniu polecenie.

```
cargo run
```

Jeżeli wykonujemy to pierwszy raz to polecenie będzie pobierało z sieci szereg niezbędnych komponentów i może to trochę potrwać.

Testowanie modułu – miganie diody D2 podłączonej do portu IO7

Na poniższym listingu jest wersja źródłowa programu cyklicznie zapalającego i gaszącego diodę LED co 500msek. Należy go umieścić w pliku main.rd (katalog src) i wykonać polecenie cargo run.

```
use esp_idf_svc::hal::{delay::FreeRtos, gpio::PinDriver, peripherals::Peripherals};
//use esp_idf_sys as _; // If using the `binstart` feature of `esp-idf-sys`, always keep this
module imported
fn main() {
        // It is necessary to call this function once. Otherwise some patches to the runtime
        // implemented by esp-idf-sys might not link properly. See
https://github.com/esp-rs/esp-idf-template/issues/71
        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);
        }
}
```

Testowanie modułu – odczyt danych z termometru SHTC3 i akceleratora ICM42670

Podobnie jak w poprzednim przykładzie należy wpisać do pliku poniższa wersję źródłową i wykonać polecenia cargo run.

```
use esp_idf_svc::hal::{
        delay::FreeRtos,
        i2c::{I2cConfig, I2cDriver},
        prelude::*,
};
//use esp_idf_sys as _;
use icm42670::{accelerometer::vector::F32x3, prelude::_accelerometer_Accelerometer, Address};
use shtcx::{Measurement, PowerMode::*};

fn main() {
        esp_idf_svc::sys::link_patches();

        let peripherals = Peripherals::take().expect("Failed to take peripherals");

        let i2c_config = I2cConfig::new()
                .baudrate(100.kHz().into())
                .sda_enable_pullup(true)
                .scl_enable_pullup(true);

        let shared_bus = shared_bus::BusManagerSimple::new(
                I2cDriver::new(
                        peripherals.i2c0,
                        peripherals.pins.gpio10,
                        peripherals.pins.gpio8,
                        &i2c_config,
                )
                .expect("Failed to create i2c driver"),
        );

        let mut shtc3 = shtcx::shtc3(shared_bus.acquire_i2c());
        let mut icm42670 = icm42670::Icm42670::new(shared_bus.acquire_i2c(), Address::Primary)
                .expect("Failed to instantiate icm42670");

        loop {
                let Measurement {
                        temperature,
                        humidity,
                } = shtc3
                        .measure(NormalMode, &mut FreeRtos)
                        .expect("Failed to read SHTC3 temp/hum");

                println!(
                        "SHTC3:\n\t- temp: {}\n\t- hum: {}",
                        temperature.as_degrees_celsius(),
                        humidity.as_percent()
                );

                let temp = icm42670
                        .temperature()
                        .expect("Failed to read ICM42670 temp");
```

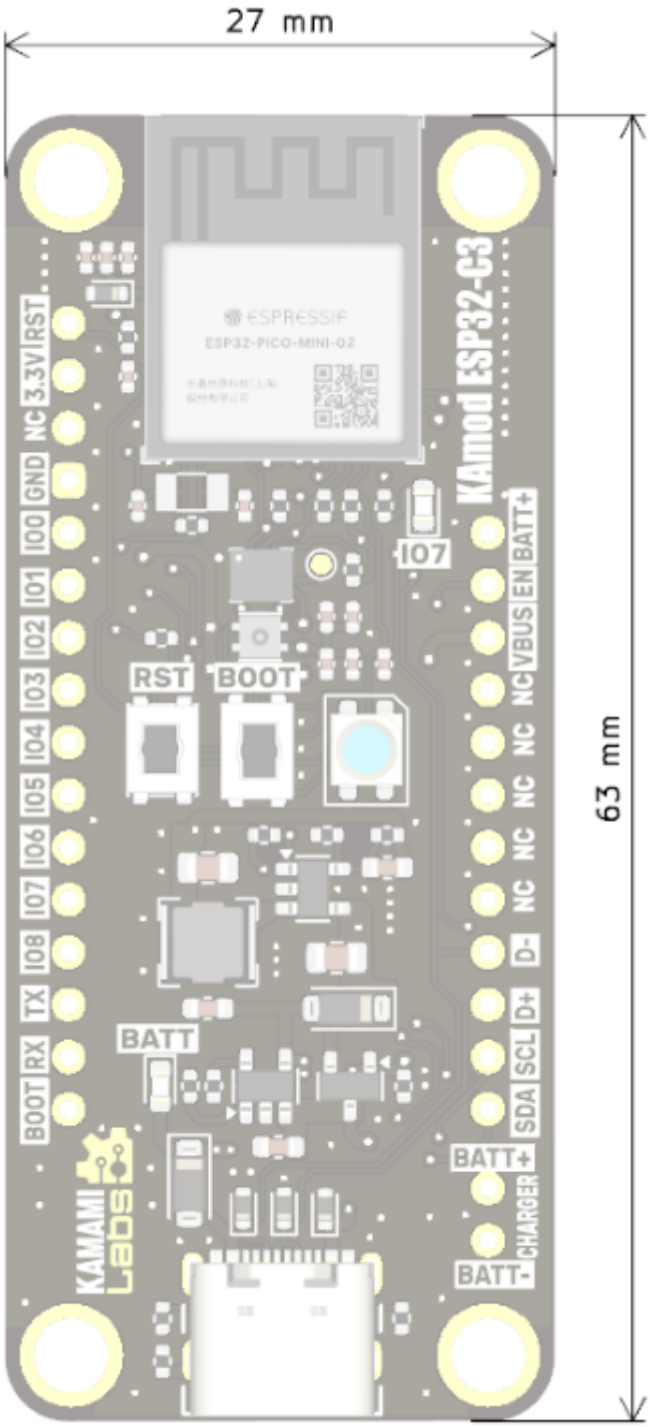
```
                let F32x3 {
                        x: ax,
                        y: ay,
                        z: az,
                } = icm42670
                        .accel_norm()
                        .expect("Failed to read ICM42670 accel data");

                let F32x3 {
                        x: gx,
                        y: gy,
                        z: gz,
                } = icm42670
                        .gyro_norm()
                        .expect("Failed to read ICM42670 gyro data");

                println!(
                        "ICM42670:\n\t- temp: {}\n\t- accel: {}, {}, {}\n\t- gyro: {}, {},
{}",
                        temp, ax, ay, az, gx, gy, gz
                );
                FreeRtos::delay_ms(1000);
        }
}
```

Po wykonaniu komendy cargo run po skompilowaniu i zapisaniu pamięci flash na ekranie terminala będą co sekundę wyświetlane odczytane po magistrali I2C dane z termometru i akceleratora.

```
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
```

Wymiary

Wymiary płytki Kamod ESP32-C3 to 63 x 27 mm.



Linki zewnętrzne

- Dokumentacja układu ESP32-C3
- Dokumentacja układu SHTC3
- Dokumentacja układu ICM42670
- Dokumentacja układu SY8088
- Dokumentacja układu MCP73831
- Get started with Rust
- Cargo Rust dokumentacja
- Rust przykłady

BTC Korporacja
05-120 Legionowo
ul. Lwowska 5
tel.: (22) 767-36-20
faks: (22) 767-36-33
e-mail:
sprzedaz@kamami.pl
https://kamami.pl