Training about ESP32-C5

Content

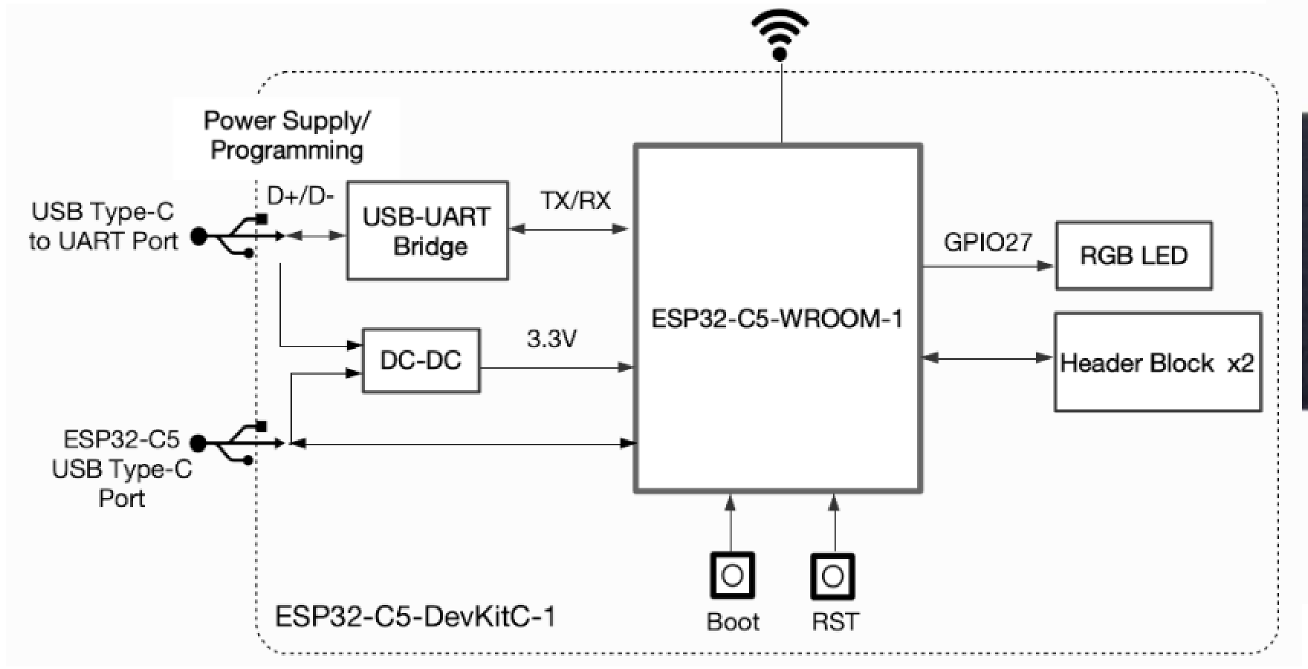
- Background
- Prepare
- Peripheral
- How to build code & Apply

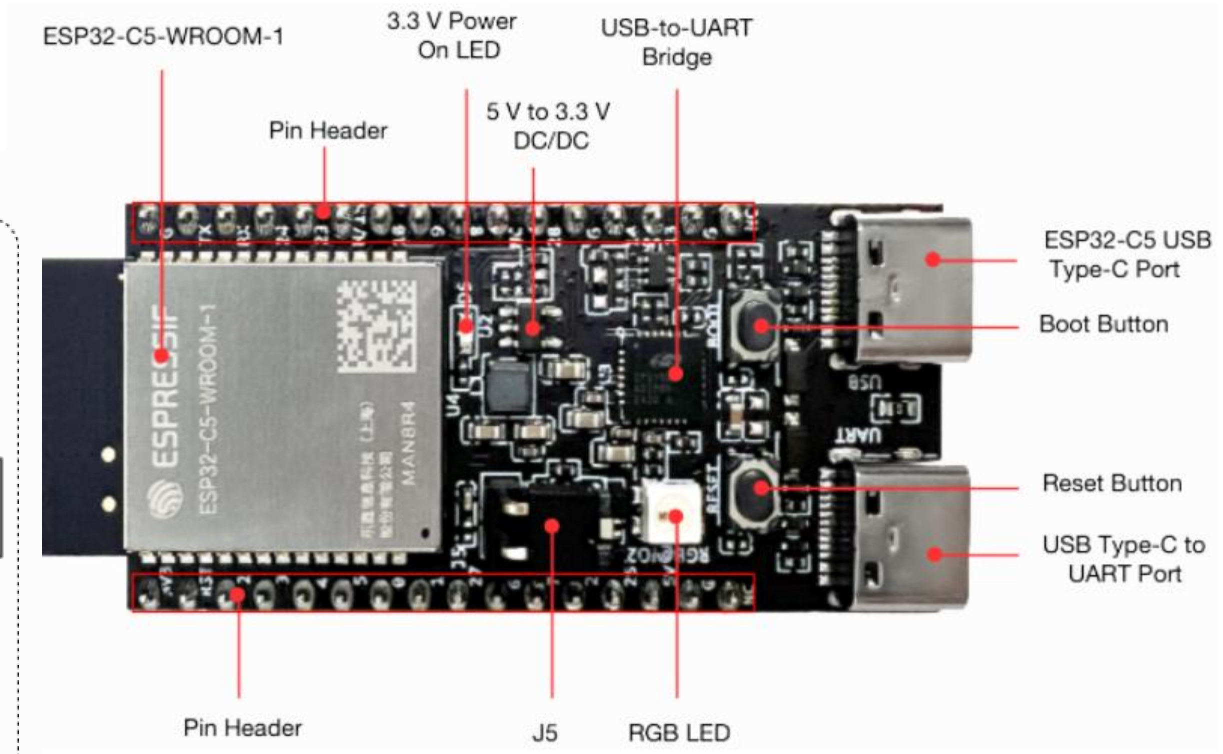
Background

Introduce ESP32-C5

ESP32-C5 is a system on a chip that integrates the following features:

- Wi-Fi (2.4 GHz band & 5 GHz)
- Bluetooth
- Dual high performance Xtensa® 32-bit LX6 CPU cores
- Ultra Low Power co-processor
- Multiple peripherals





Background

Advantage of ESP32-C5

1. **Dual-Band Wi-Fi 6 Support (2.4 GHz & 5 GHz)**
- Supports Wi-Fi 6 and both bands (2.4 GHz & 5 GHz).
- This makes the network more stable, less speed**.**

2. **Multiple protocol connections**
- In addition to Wi-Fi, there is also Bluetooth 5 and Zigbee/Thread/Matter support → suitable for smart factory**.**

3. **Good activation**
- Uses 32-bit RISC-V core, speed up to 240 MHz, fast and stable processing**.**

4. **Rich Peripheral Support**
- Up to 29 GPIOs, plus interfaces like SPI, I2C, I2S, UART, PWM, and USB OTG.
- Supports external PSRAM for memory expansion.

5. **Open-Source SDK & Ecosystem**
- Fully supported by ESP-IDF, ensuring easy migration and integration with existing ESP32 projects. Arduino support is also in development

Prepare

Software need install:

- Adruino IDE: coding, edit, debug, and build software applications

Knowledge:

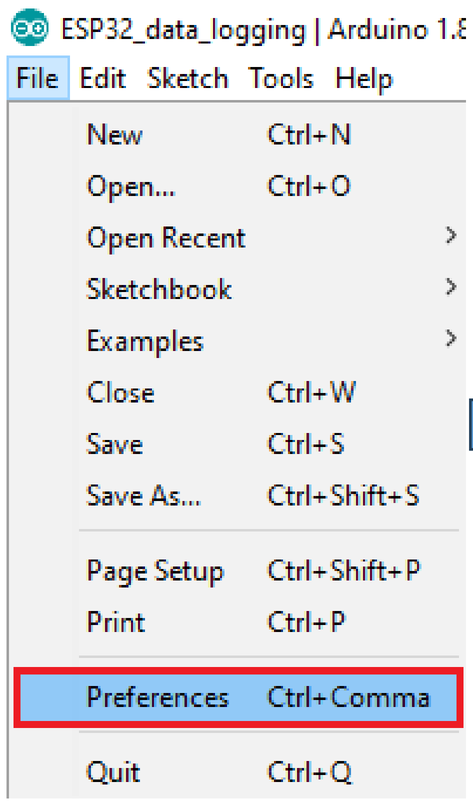
- Basic about C/C++
- Hardware/ firmware
- Peripheral
- Wireless communication network

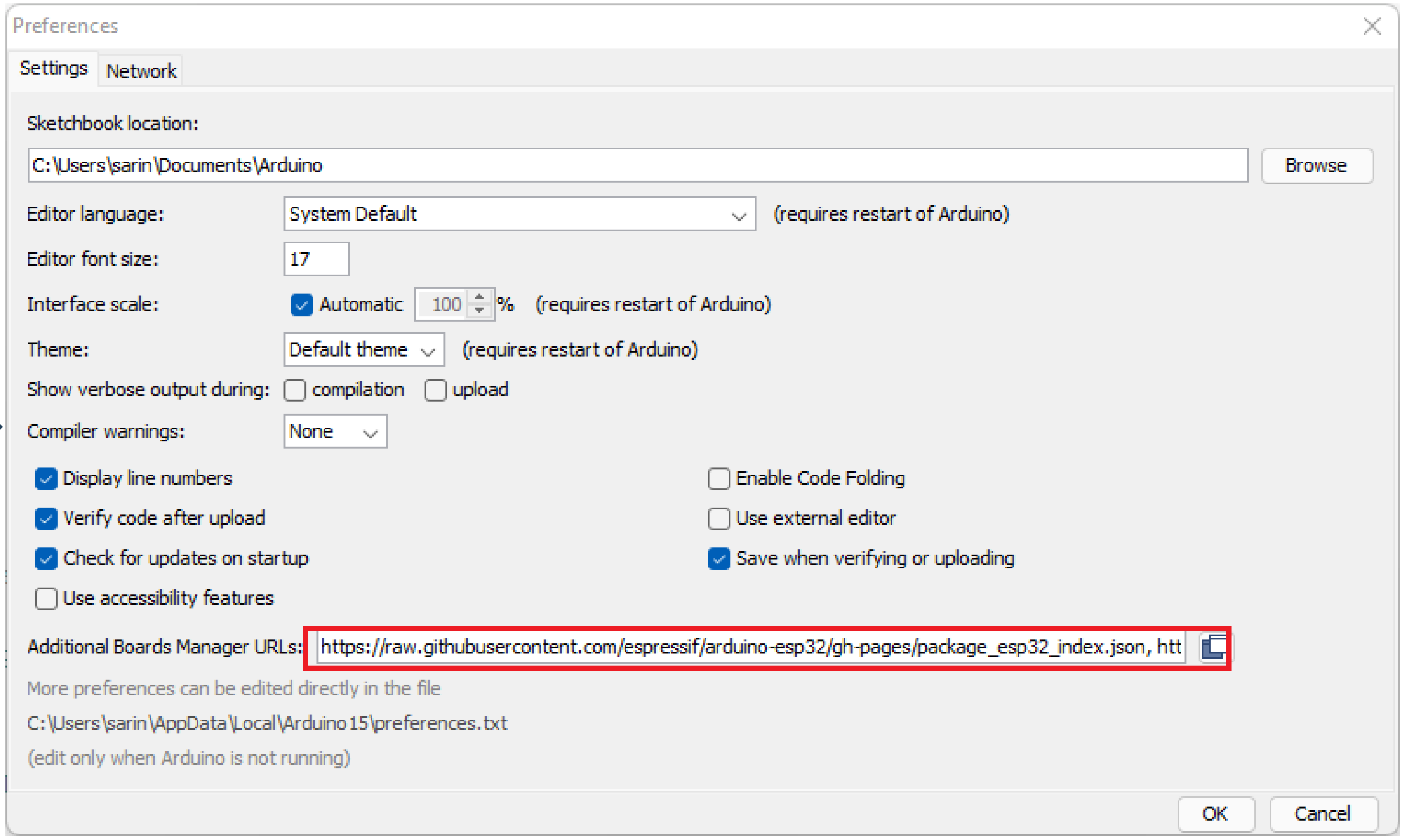
Prepare

Setup board ESP32-C5

1. In your Arduino IDE, go to **File**> **Preferences**

2. Enter the following into the “Additional Board Manager URLs” field:
https://raw.githubusercontent.com/espressif/arduino-esp32/ghpages/package_esp32_index.json





Sketchbook location:
C:\Users\sarin\Documents\Arduino

Editor language: System Default (requires restart of Arduino)
Editor font size: 17
Interface scale: ✓ Automatic 100 % (requires restart of Arduino)
Theme: Default theme (requires restart of Arduino)
Show verbose output during: ☐ compilation ☐ upload
Compiler warnings: None

✓ Display line numbers
✓ Verify code after upload
✓ Check for updates on startup
☐ Use accessibility features
☐ Enable Code Folding
☐ Use external editor
✓ Save when verifying or uploading

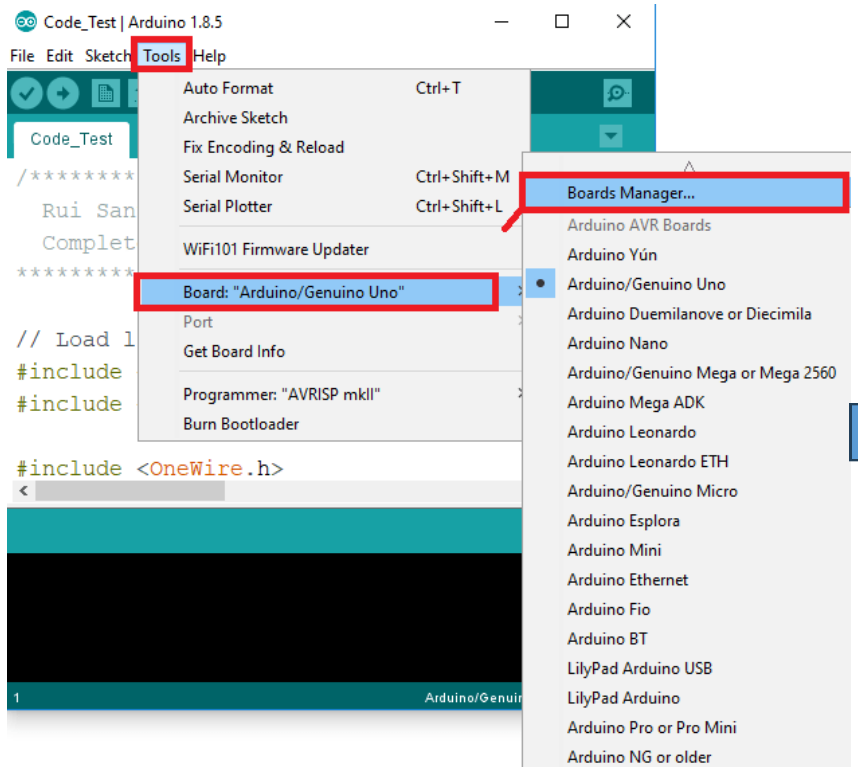
Additional Boards Manager URLs: https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json, htt

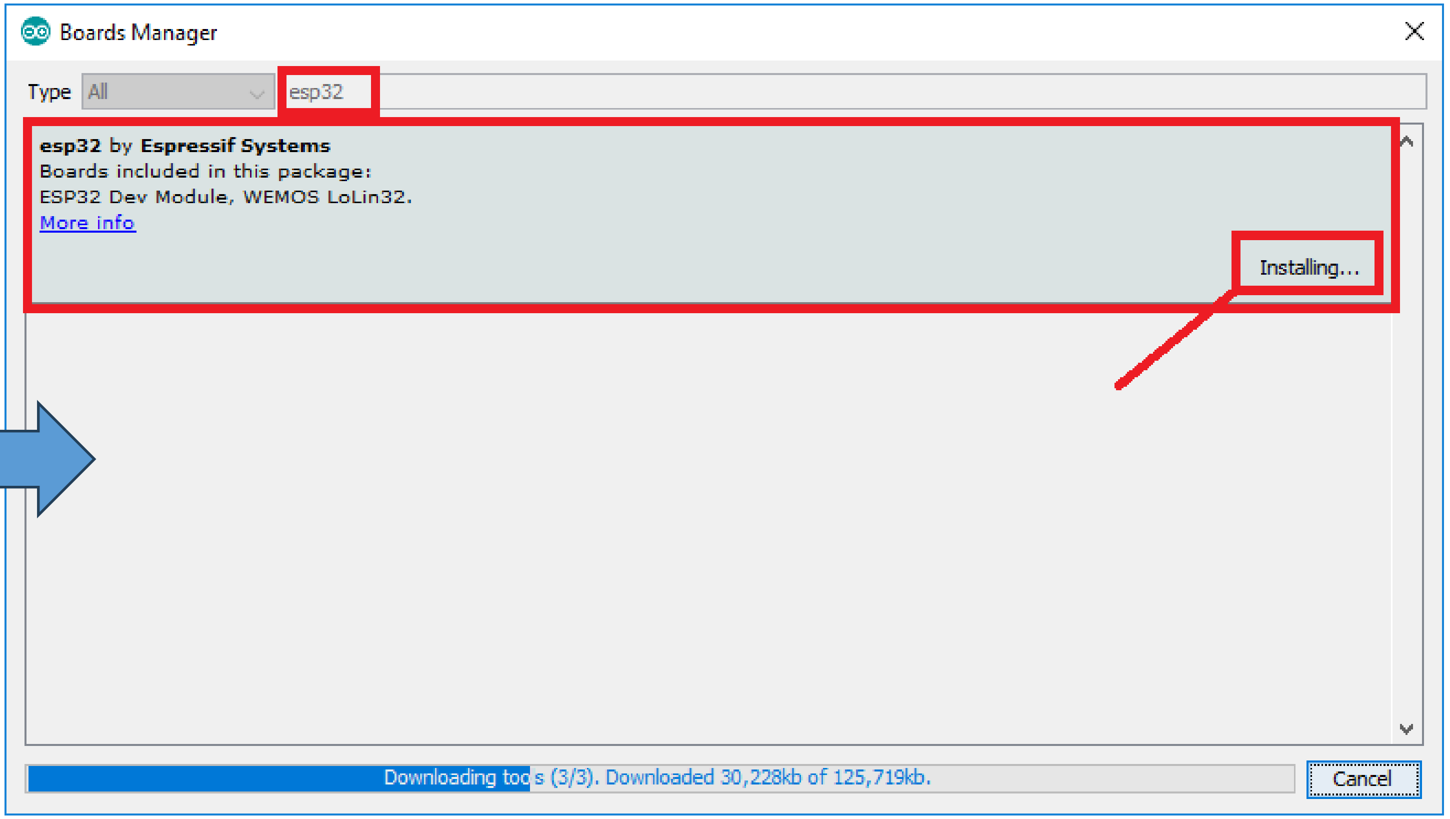
More preferences can be edited directly in the file
C:\Users\sarin\AppData\Local\Arduino15\preferences.txt
(edit only when Arduino is not running)

Prepare

Setup board ESP32-C5

3. Open the Boards Manager. Go to **Tools** > **Board** > **Boards Manager…**
4. Search for **ESP32** and press install button for the "**ESP32 by Espressif Systems**":





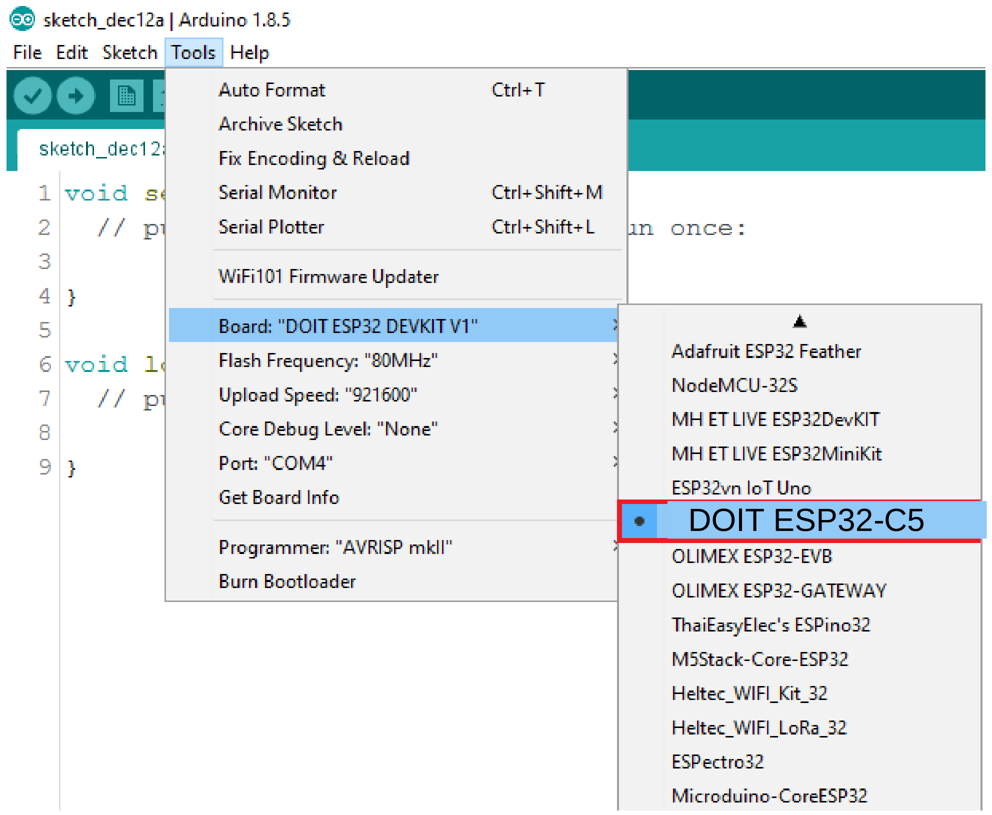
esp32 by **Espressif Systems**
Boards included in this package:
ESP32 Dev Module, WEMOS LoLin32.
More info

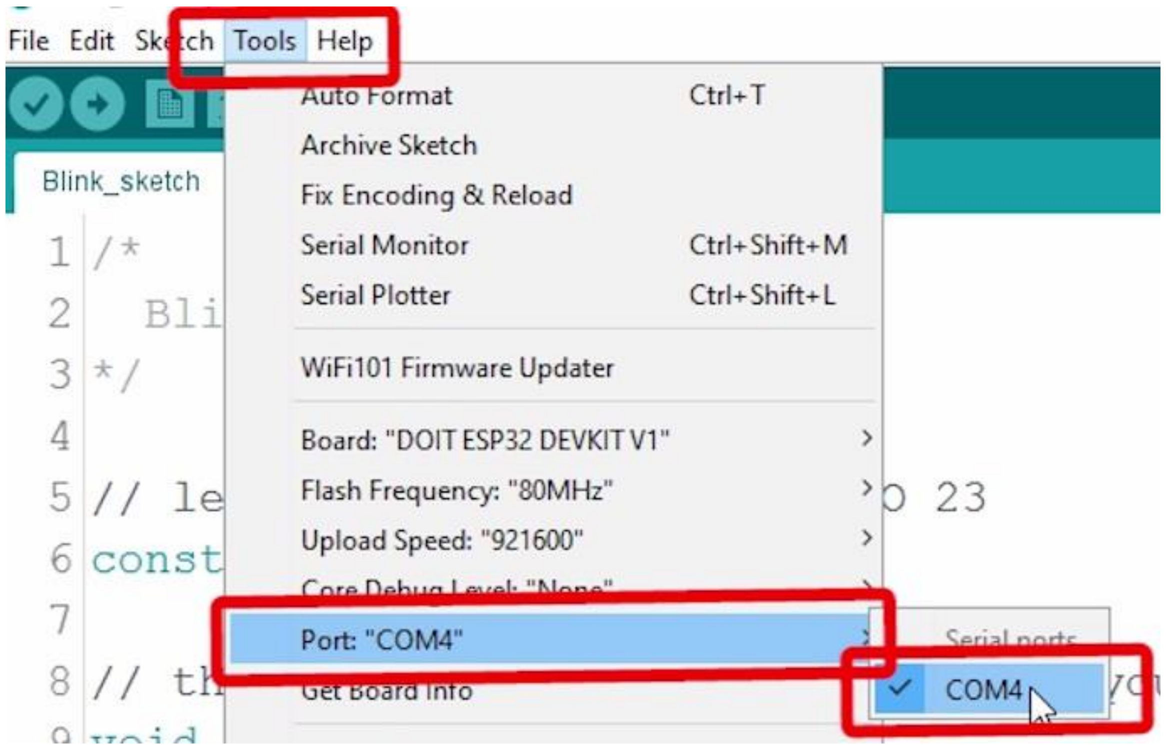
Downloading tools (3/3). Downloaded 30,228kb of 125,719kb.

Prepare

Setup board ESP32-C5

1. Select your Board in **Tools** > **Board** menu (in my case it's the **DOIT ESP32-C5**)
2. Select your Board

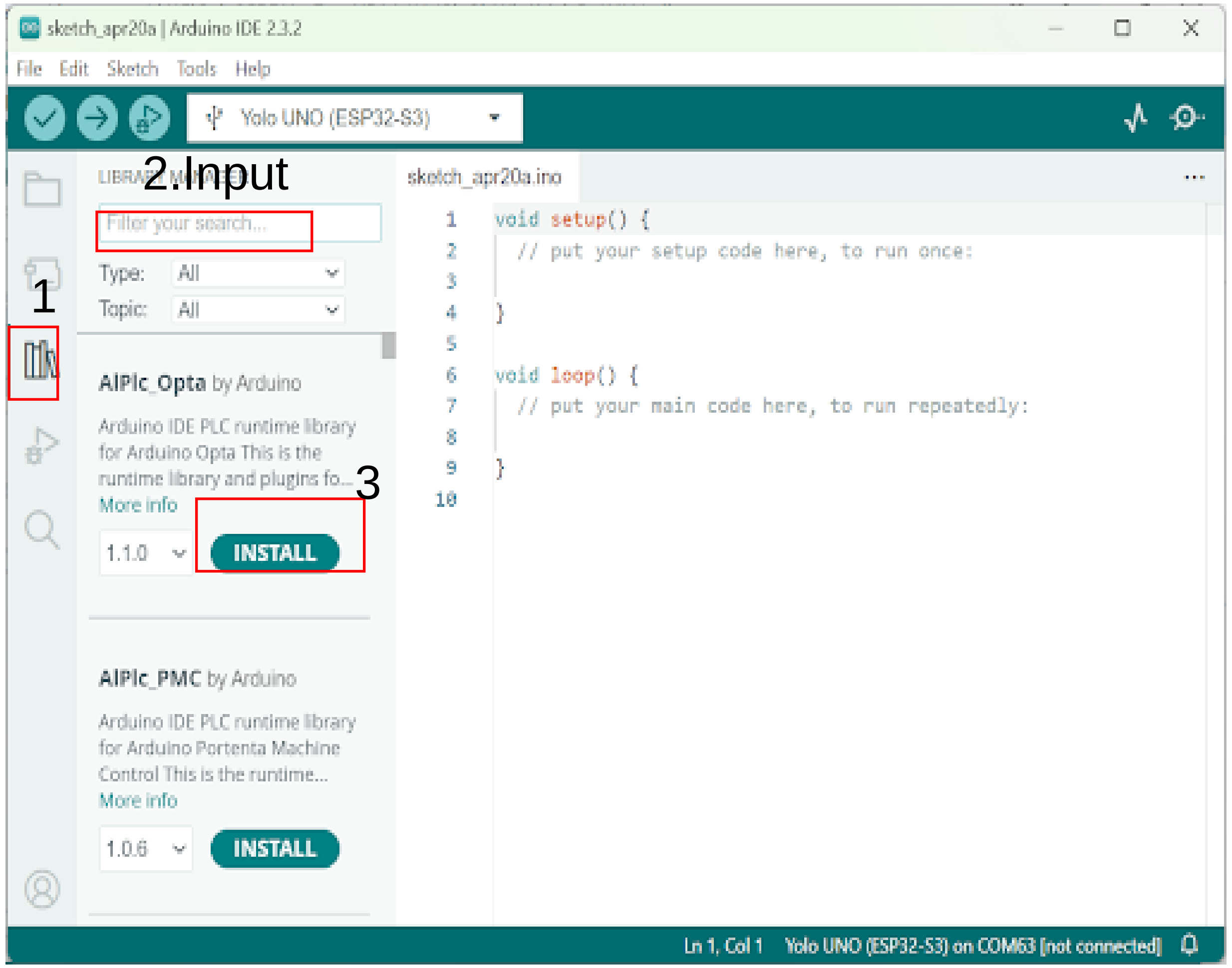




Prepare

How to install library

➢ Install manufacturer's library



Type: All

Topic: All

AIPlc_Opta by Arduino

Arduino IDE PLC runtime library for Arduino Opta This is the runtime library and plugins fo...

1.1.0

AIPlc_PMC by Arduino

Arduino IDE PLC runtime library for Arduino Portenta Machine Control This is the runtime...

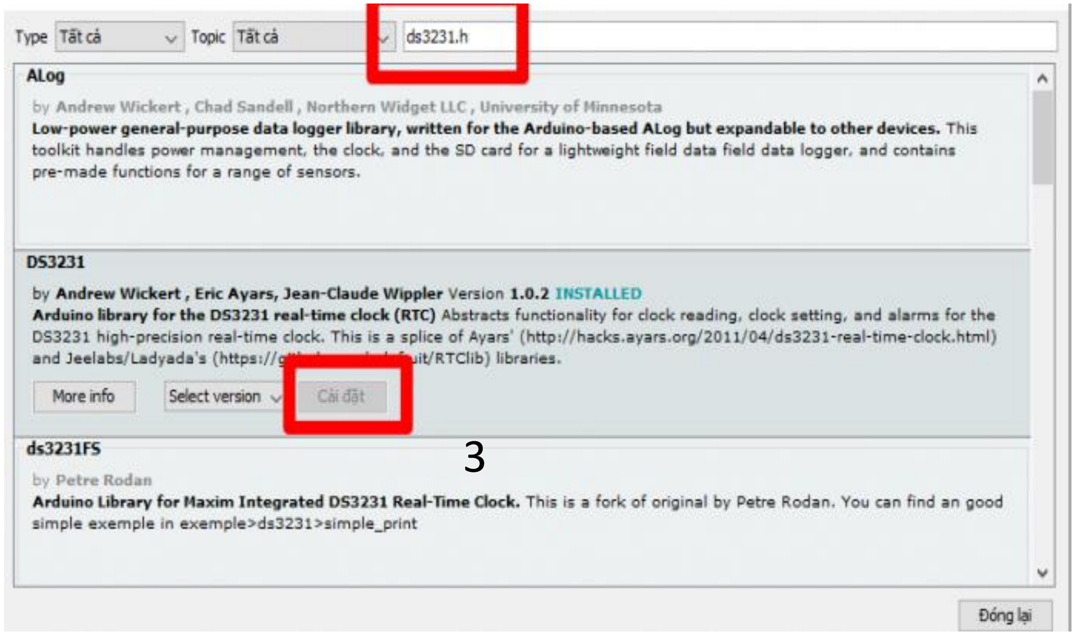
1.0.6

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Ln 1, Col 1 Yolo UNO (ESP32-S3) on COM63 [not connected]

For example

2.Input



Type: Tất cả Topic: Tất cả ds3231.h

ALog

by Andrew Wickert , Chad Sandell , Northern Widget LLC , University of Minnesota

Low-power general-purpose data logger library, written for the Arduino-based ALog but expandable to other devices. This toolkit handles power management, the clock, and the SD card for a lightweight field data field data logger, and contains pre-made functions for a range of sensors.

DS3231

by **Andrew Wickert , Eric Ayars, Jean-Claude Wippler** Version **1.0.2** INSTALLED

Arduino library for the DS3231 real-time clock (RTC) Abstracts functionality for clock reading, clock setting, and alarms for the DS3231 high-precision real-time clock. This is a splice of Ayars' (http://hacks.ayars.org/2011/04/ds3231-real-time-clock.html) and Jeelabs/Ladyada's (https://g[illegible]it/RTClib) libraries.

ds3231FS

by Petre Rodan

Arduino Library for Maxim Integrated DS3231 Real-Time Clock. This is a fork of original by Petre Rodan. You can find an good simple exemple in exemple>ds3231>simple_print

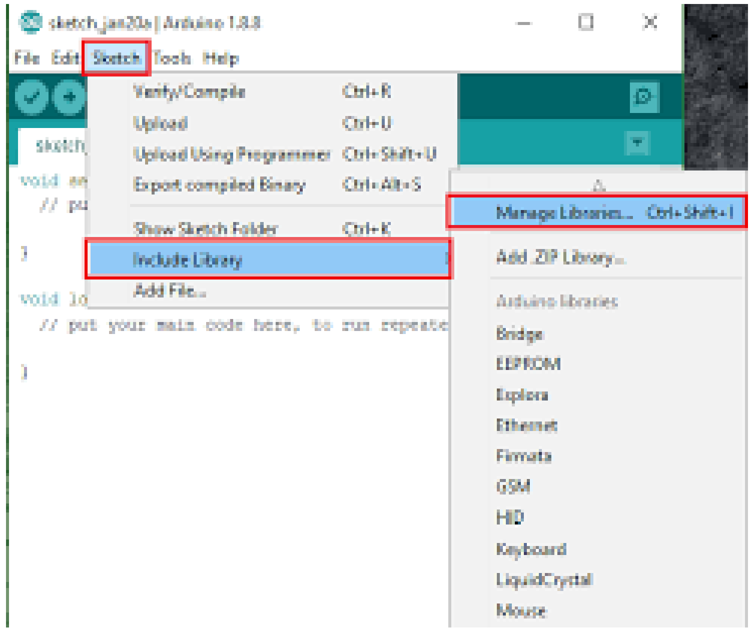
Prepare

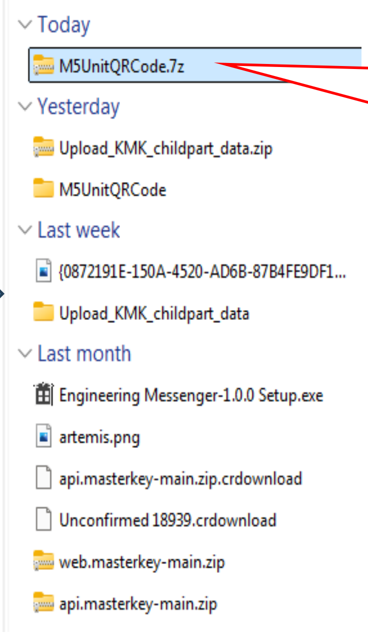
How to install library

➢ Install other peripheral libraries

If you cannot find the desired library in the library manager, you can do the following:

1. Install peripheral libraries
2. Add library to AdruinoIDE:


sketch_jan20a | Arduino 1.8.8
File Edit Sketch Tools Help
Verify/Compile Ctrl+R
Upload Ctrl+U
Upload Using Programmer Ctrl+Shift+U
Export compiled Binary Ctrl+Alt+S
Show Sketch Folder Ctrl+K
Include Library
Add File...
Manage Libraries... Ctrl+Shift+I
Add .ZIP Library...
Arduino libraries
Bridge
EEPROM
Esplora
Ethernet
Firmata
GSM
HID
Keyboard
LiquidCrystal
Mouse

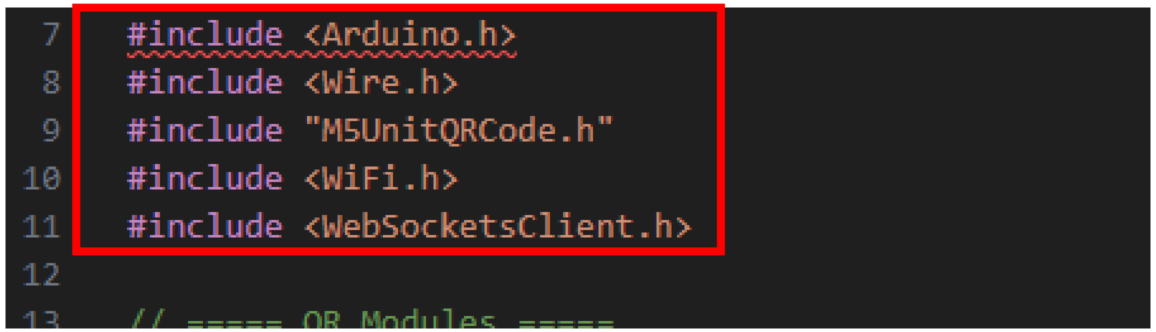

Today
M5UnitQRCode.7z
Yesterday
Upload_KMK_childpart_data.zip
M5UnitQRCode
Last week
{0872191E-150A-4520-AD6B-87B4FE9DF1...
Upload_KMK_childpart_data
Last month
Engineering Messenger-1.0.0 Setup.exe
artemis.png
api.masterkey-main.zip.crdownload
Unconfirmed 18939.crdownload
web.masterkey-main.zip
api.masterkey-main.zip

Choose file to add library

Prepare

Declare required libraries

After installing the library, you need to declare the library to use it.

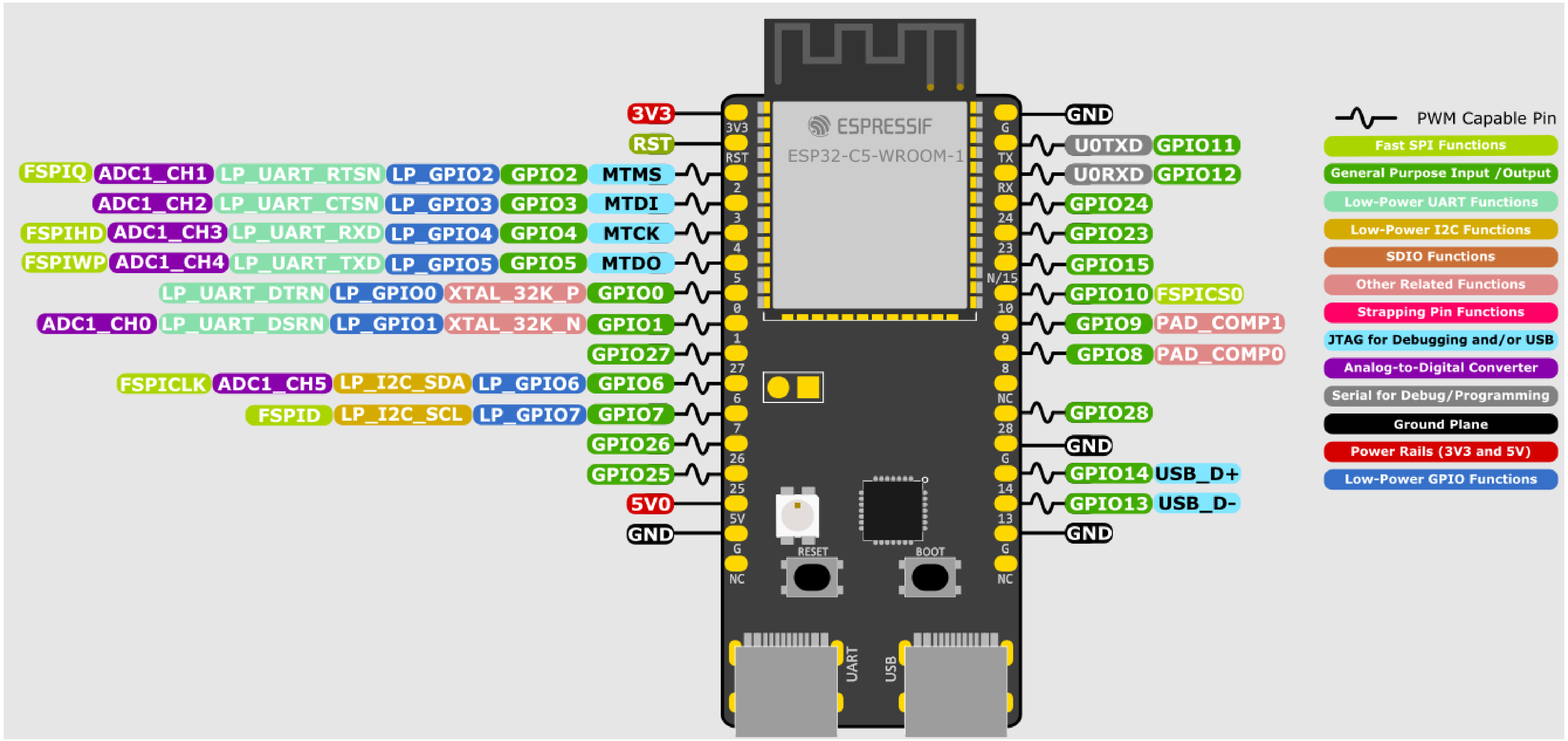


```
#include <Arduino.h>
#include <Wire.h>
#include "M5UnitQRCode.h"
#include <WiFi.h>
#include <WebSocketsClient.h>

// ===== QR Modules =====
```

Peripheral

Overview

- The ESP32-C5 module has multiple pins (GPIOs) that can perform different functions. These pins are color-coded in the diagram for easy understanding:


3V3
RST
FSPIQ ADC1_CH1 LP_UART_RTSN LP_GPIO2 GPIO2 MTMS
ADC1_CH2 LP_UART_CTSN LP_GPIO3 GPIO3 MTDI
FSPIHD ADC1_CH3 LP_UART_RXD LP_GPIO4 GPIO4 MTCK
FSPIWP ADC1_CH4 LP_UART_TXD LP_GPIO5 GPIO5 MTDO
LP_UART_DTRN LP_GPIO0 XTAL_32K_P GPIO0
ADC1_CH0 LP_UART_DSRN LP_GPIO1 XTAL_32K_N GPIO1
GPIO27
FSPICLK ADC1_CH5 LP_I2C_SDA LP_GPIO6 GPIO6
FSPID LP_I2C_SCL LP_GPIO7 GPIO7
GPIO26
GPIO25
5V0
GND
3V3 RST 2 3 4 5 0 1 27 6 7 26 25 5V G NC
ESPRESSIF
ESP32-C5-WROOM-1
RESET
BOOT
UART
USB
G TX RX 24 23 N/15 10 9 8 NC 28 G 14 13 G NC
GND
U0TXD GPIO11
U0RXD GPIO12
GPIO24
GPIO23
GPIO15
GPIO10 FSPICS0
GPIO9 PAD_COMP1
GPIO8 PAD_COMP0
GPIO28
GND
GPIO14 USB_D+
GPIO13 USB_D-
GND
PWM Capable Pin
Fast SPI Functions
General Purpose Input /Output
Low-Power UART Functions
Low-Power I2C Functions
SDIO Functions
Other Related Functions
Strapping Pin Functions
JTAG for Debugging and/or USB
Analog-to-Digital Converter
Serial for Debug/Programming
Ground Plane
Power Rails (3V3 and 5V)
Low-Power GPIO Functions

Peripheral

Overview

1. **General GPIO (Green)**

- These are basic input/output pins.
- You can use them to read signals (like from a button) or send signals (like turning on an LED).
- Many of them support PWM for controlling brightness or motor speed.

2. **Communication Interfaces**

- **UART (Purple):** For serial communication with other devices (e.g., sensors, GPS).
- **SPI (Light Green):** High-speed communication with displays or memory chips.
- **I2C (Yellow):** Commonly used for sensors and small modules.

3. **ADC (Pink)**

- Analog-to-Digital Converter pins.
- Used to read analog signals (e.g., from a temperature sensor or potentiometer).

4. **USB & Debug**

- USB D+ / D- (Blue): For programming and USB communication.
- JTAG (Green): For advanced debugging.

5. **Special Functions**

- PAD_COMP (Red): Comparator-related functions.
- XTAL: Crystal oscillator pins for clock.

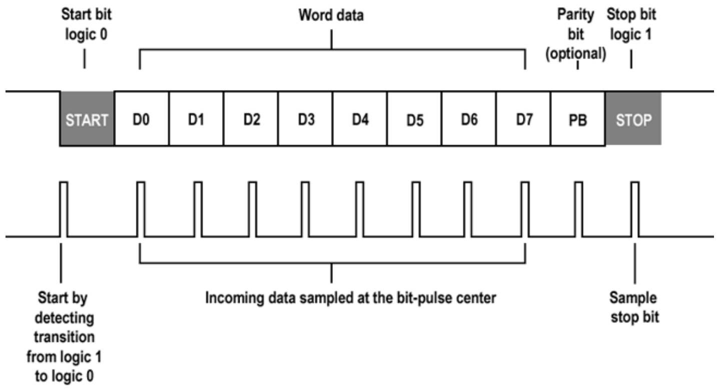
6. **Power**

- Pins3V3 and GND: Power supply for the board.

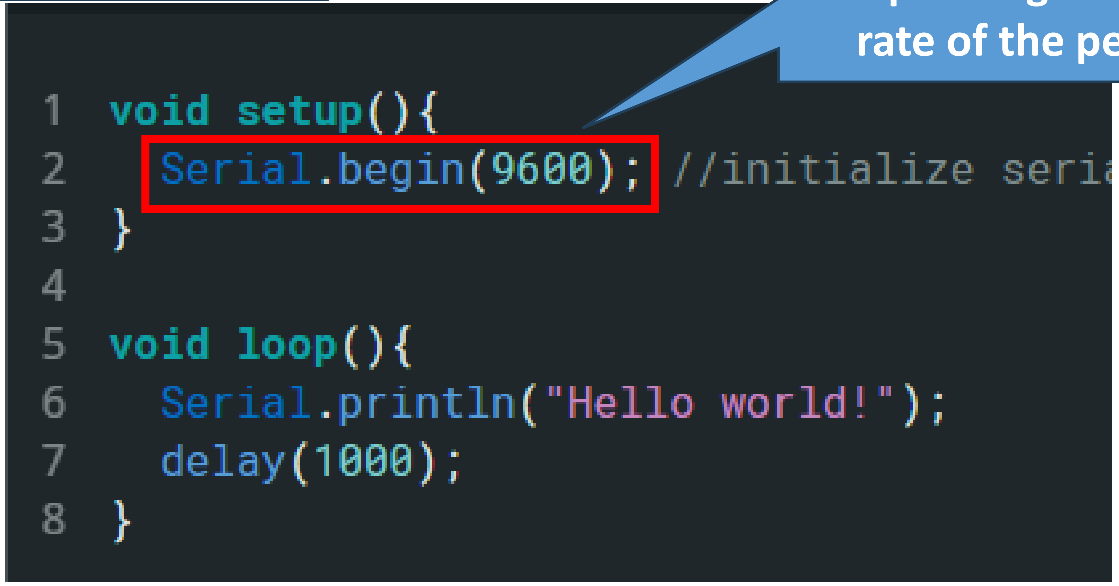
Peripheral

UART communication

- UART is an asynchronous communication, meaning it does not have a dedicated clock line.
- Both devices must agree on the baud rate to properly interpret the data.
- Data is sampled between each bit to avoid noise.

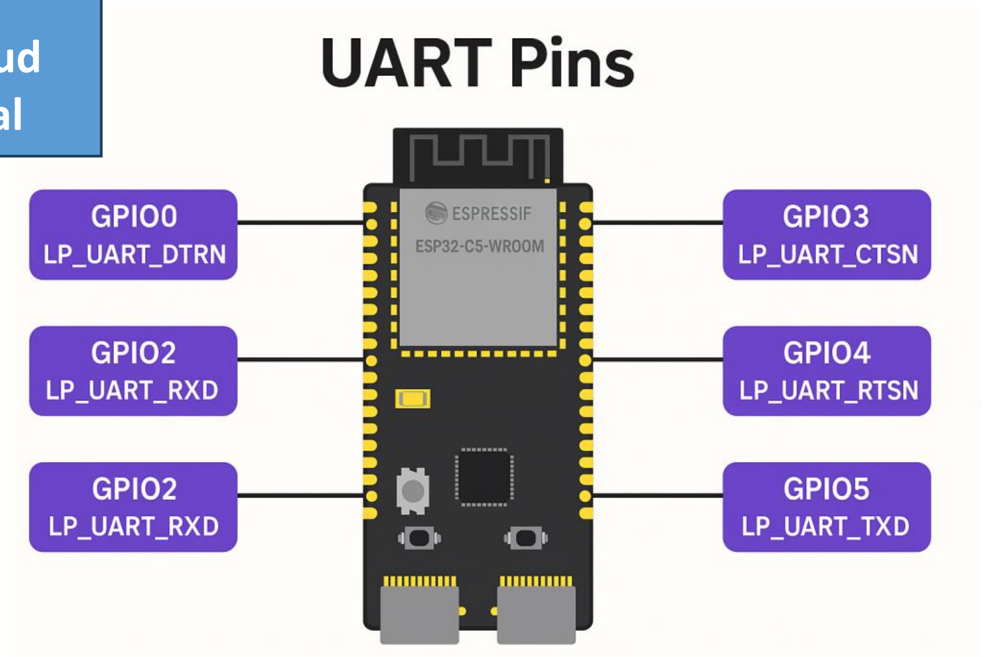


For example



```
void setup(){
  Serial.begin(9600); //initialize seria
}

void loop(){
  Serial.println("Hello world!");
  delay(1000);
}
```

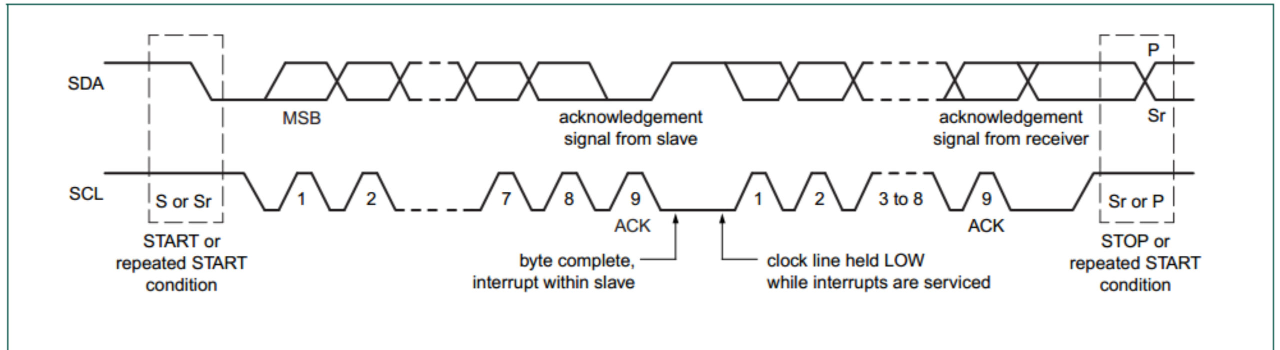
Set up baud rate depending on the baud rate of the peripheral



Peripheral

I2C communication

I2C communication protocol uses two wires to share information. One is used for the clock signal (**SCL**) and the other is used to send and receive data (**SDA**).


SDA
SCL
MSB
acknowledgement signal from slave
acknowledgement signal from receiver
P
Sr
S or Sr
1
2
7
8
9
ACK
1
2
3 to 8
9
ACK
Sr or P
START or repeated START condition
byte complete, interrupt within slave
clock line held LOW while interrupts are serviced
STOP or repeated START condition

Data transfer on the I²C-bus

I2C Device	**ESP32**
SDA	SDA (default is GPIO 21)
SCL	SCL (default is GPIO 22)
GND	GND
VCC	usually 3.3V or 5V

When using the ESP32 with Arduino IDE, the default I2C pins are (SCL) and (SDA) but you can configure your code to use any other pins.

For example

Declare I2C's library

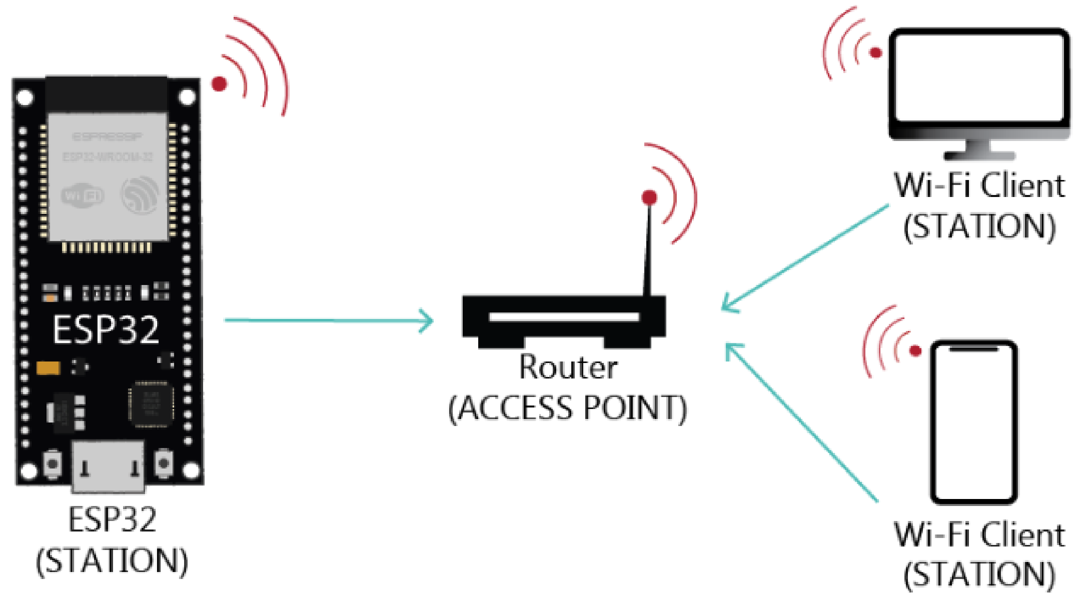
Call I2C's library

```
#include <Wire.h>

void setup() {
  Wire.begin();
  Serial.begin(115200);
  Serial.println("\nI2C Scanner");
}

void loop() {
  byte error, address;
  int nDevices;
  Serial.println("Scanning...");
  nDevices = 0;
  for(address = 1; address < 127; address++ ) {
    Wire.beginTransmission(address);
```

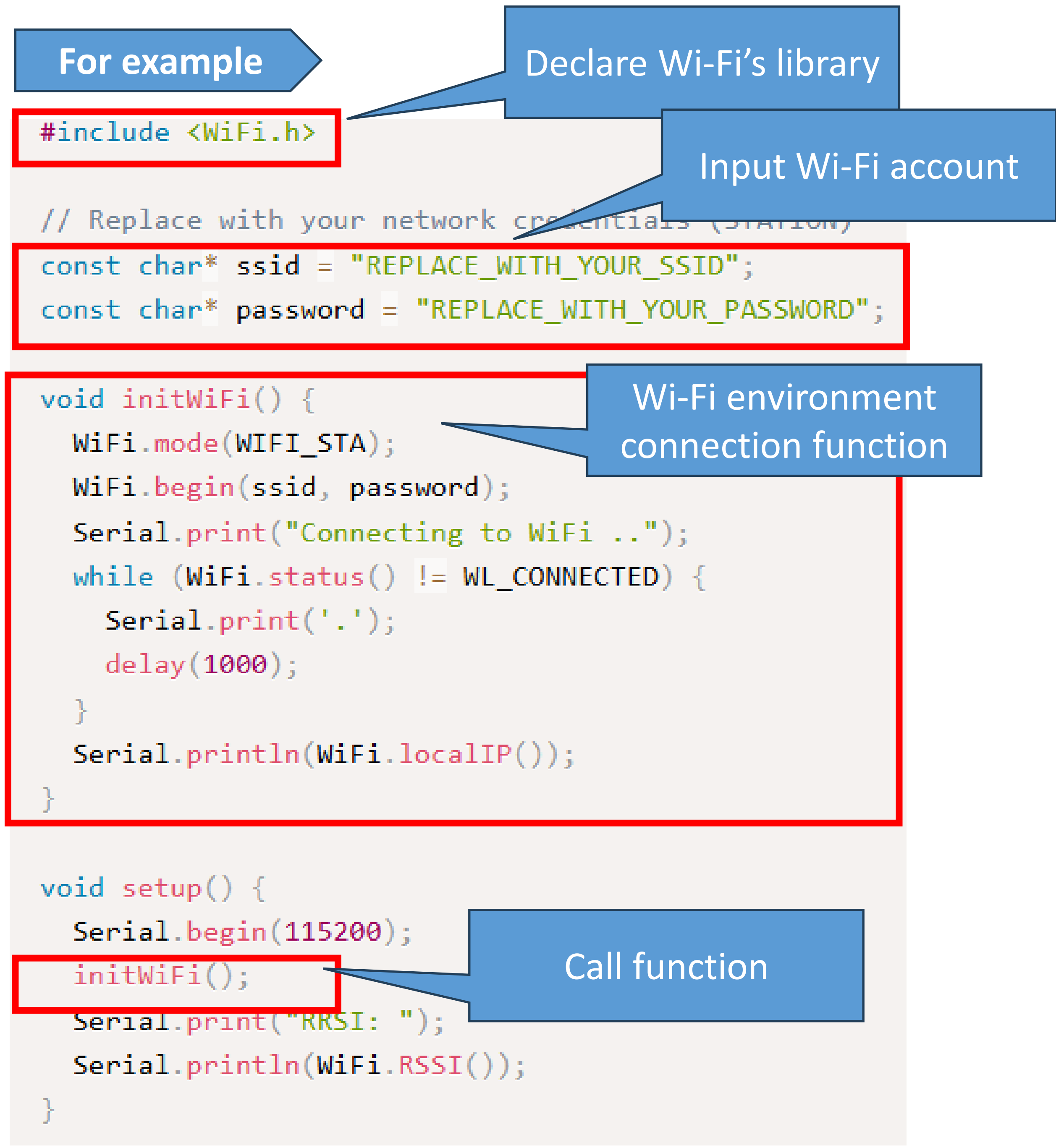
Peripheral

Wi-Fi

When the ESP32 is set as a Wi-Fi station, it can connect to other networks (like your router). In this scenario, the router assigns a unique IP address to your ESP board. You can communicate with the ESP using other devices (stations) that are also connected to the same network by referring to the ESP unique IP address.



For example

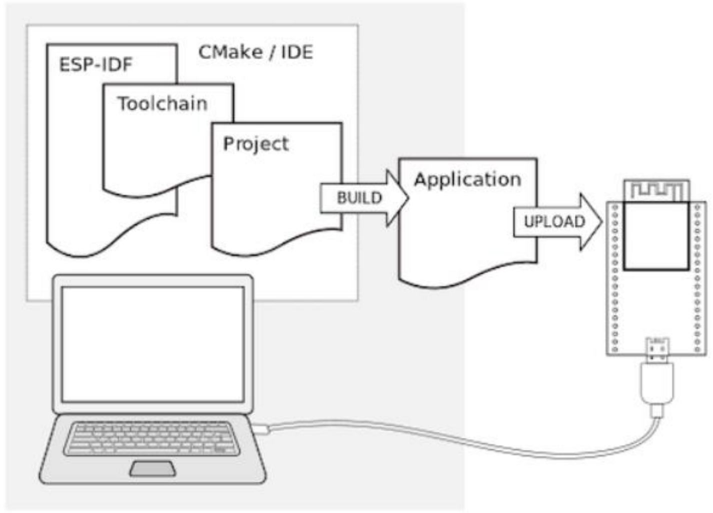


```
#include <WiFi.h>

// Replace with your network credentials (STATION)
const char* ssid = "REPLACE_WITH_YOUR_SSID";
const char* password = "REPLACE_WITH_YOUR_PASSWORD";

void initWiFi() {
  WiFi.mode(WIFI_STA);
  WiFi.begin(ssid, password);
  Serial.print("Connecting to WiFi ..");
  while (WiFi.status() != WL_CONNECTED) {
    Serial.print('.');
    delay(1000);
  }
  Serial.println(WiFi.localIP());
}

void setup() {
  Serial.begin(115200);
  initWiFi();
  Serial.print("RRSI: ");
  Serial.println(WiFi.RSSI());
}
```

How to build code

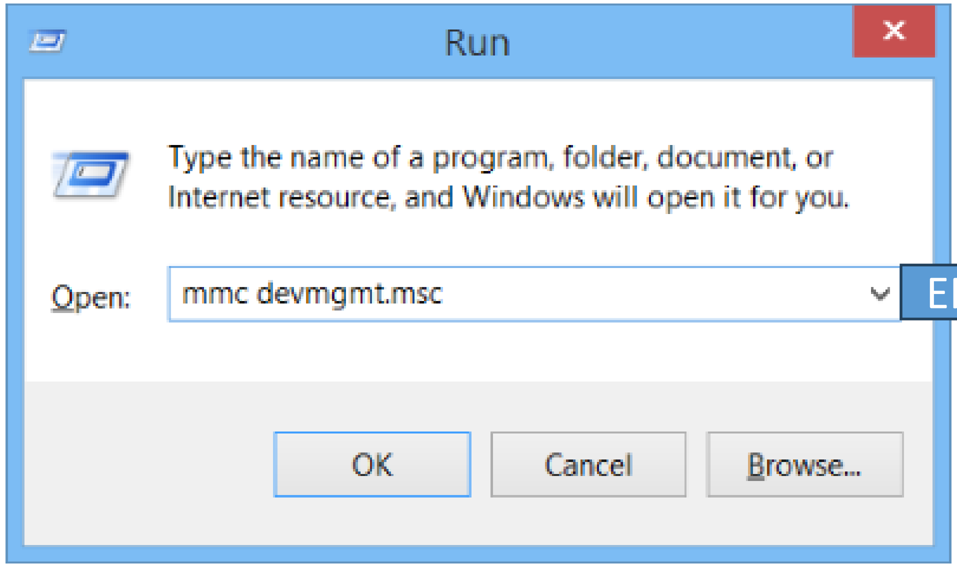
Set up hardware

Step 1: Connect ESP32-C5 to PC



Step 2: Find the connection port of ESP32-C5 to the PC

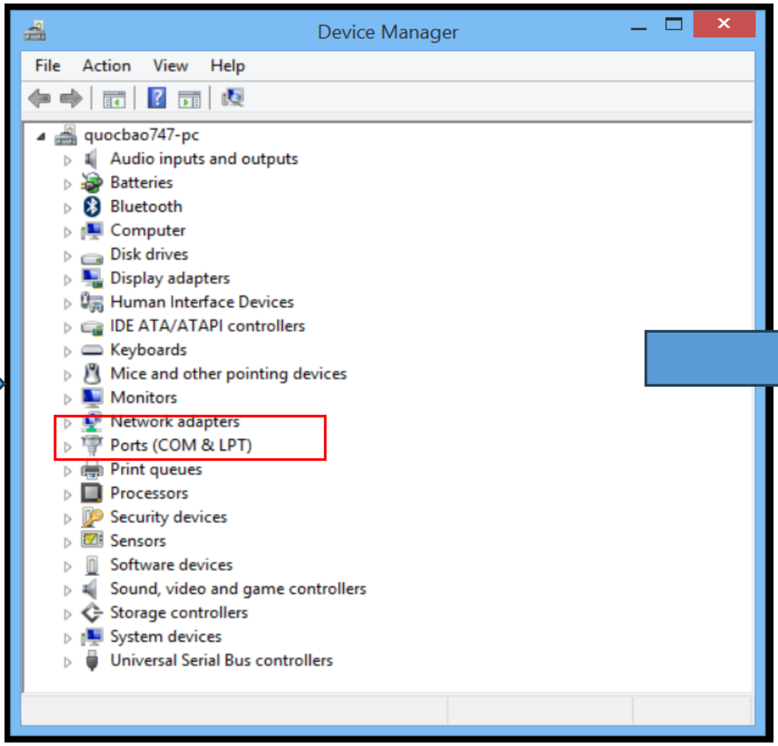
You open the Run window and type the command "mmc devmgmt.msc".


Run

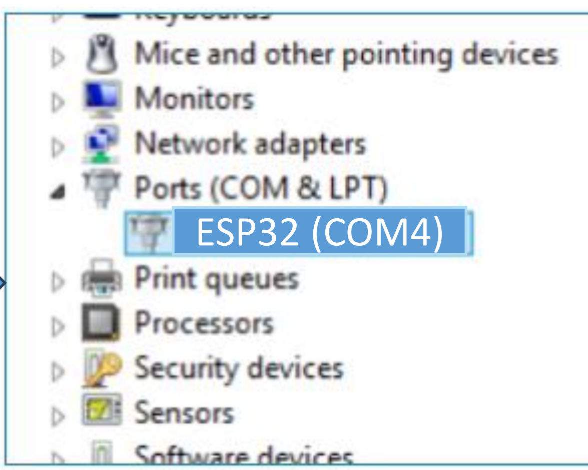
Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.

Open: mmc devmgmt.msc

ENTER


Device Manager

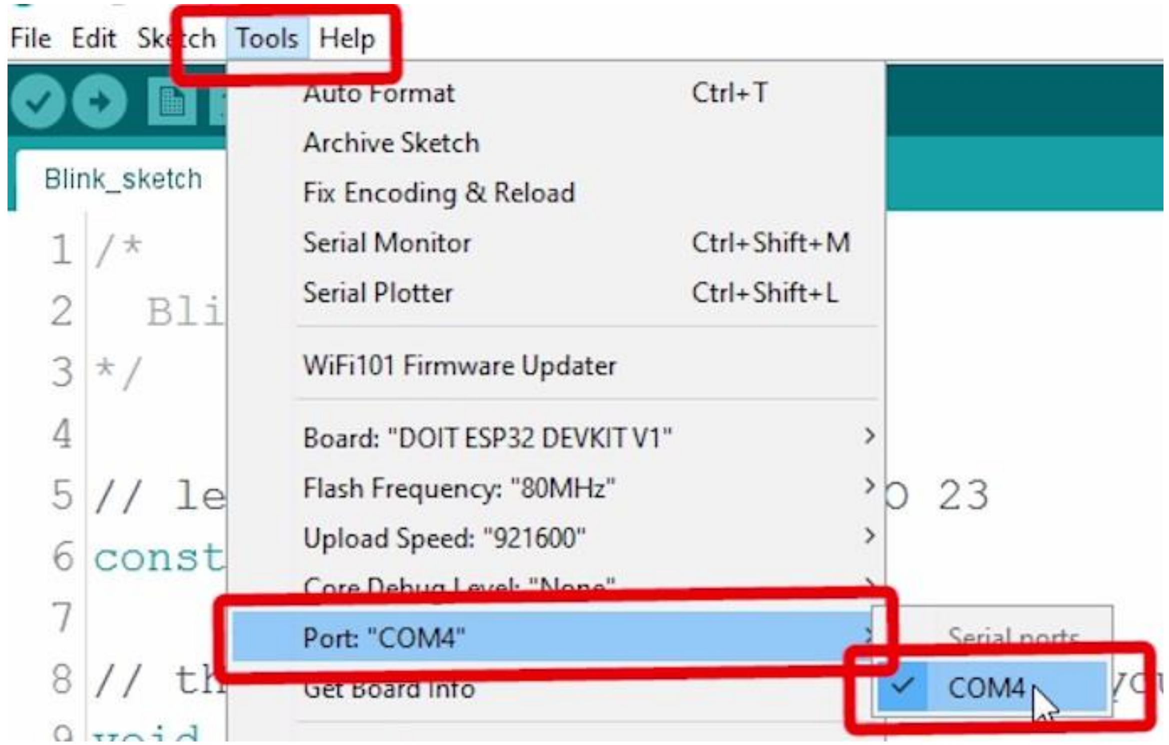
quocbao747-pc
- Audio inputs and outputs
- Batteries
- Bluetooth
- Computer
- Disk drives
- Display adapters
- Human Interface Devices
- IDE ATA/ATAPI controllers
- Keyboards
- Mice and other pointing devices
- Monitors
- Network adapters
- Ports (COM & LPT)
- Print queues
- Processors
- Security devices
- Sensors
- Software devices
- Sound, video and game controllers
- Storage controllers
- System devices
- Universal Serial Bus controllers


- Mice and other pointing devices
- Monitors
- Network adapters
- Ports (COM & LPT)
  - ESP32 (COM4)
- Print queues
- Processors
- Security devices
- Sensors
- Software devices

How to build code

Set up hardware

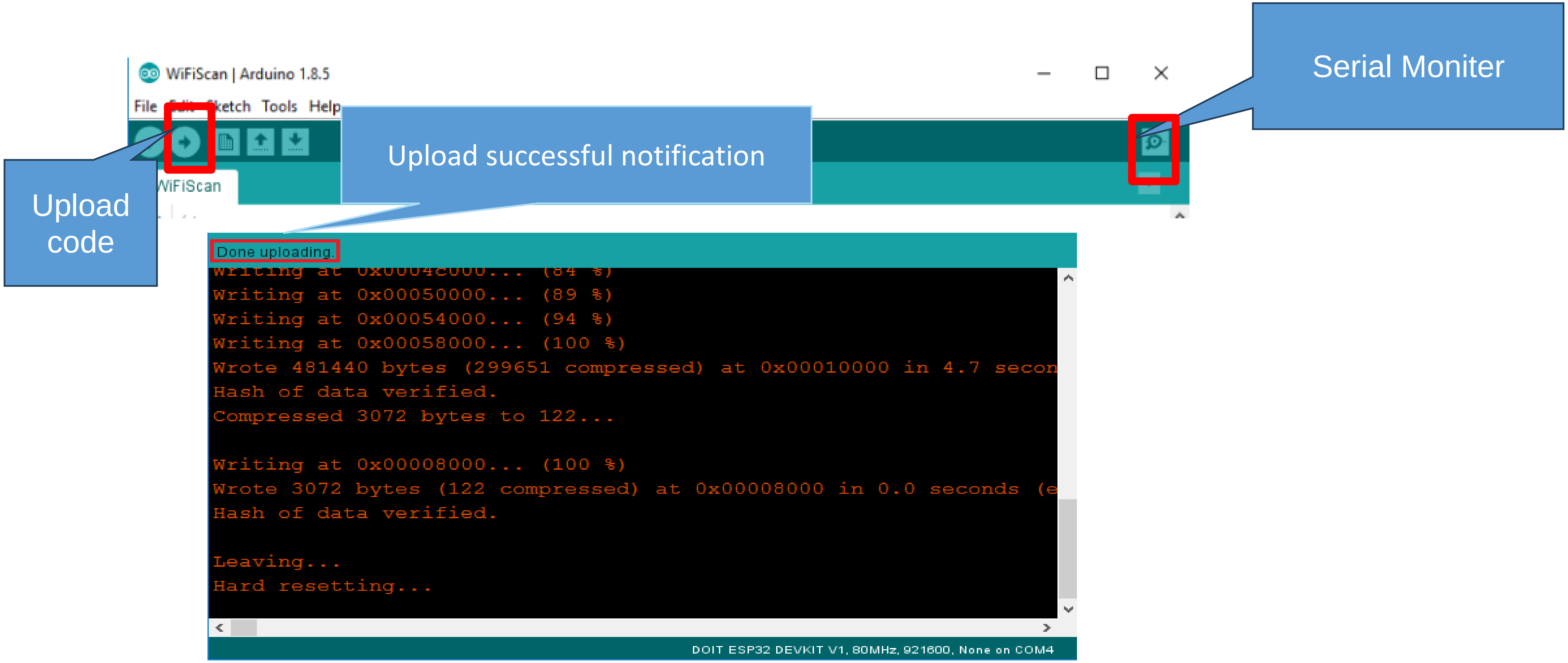
Step3: Go to menu Tools -> Serial Port -> select the Arduino port connected to the computer.



How to build code

Build code

1. Press the **Upload** button in the Arduino IDE. Wait a few seconds while the code compiles and uploads to your board.
2. Press the ESP32 on-board **Enable** button and you should see the networks available near your ESP32:

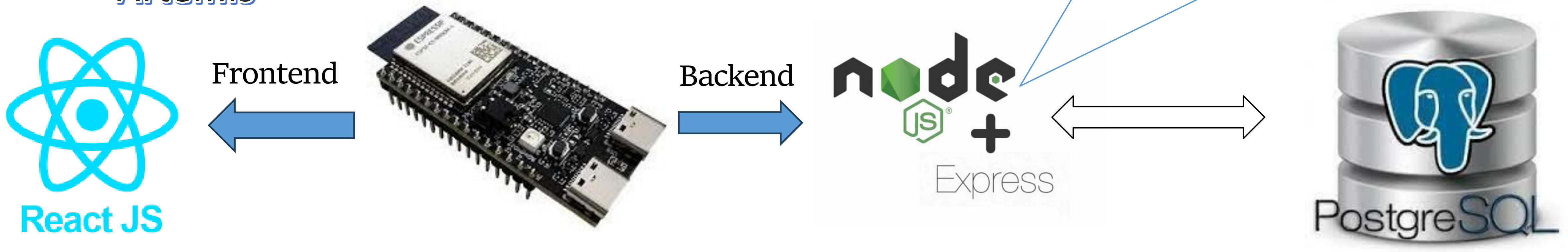

WiFiScan | Arduino 1.8.5
File Edit Sketch Tools Help
WiFiScan
Serial Moniter
Upload successful notification
Upload code
Done uploading.
DOIT ESP32 DEVKIT V1, 80MHz, 921600, None on COM4

```
Writing at 0x0004c000... (84 %)
Writing at 0x00050000... (89 %)
Writing at 0x00054000... (94 %)
Writing at 0x00058000... (100 %)
Wrote 481440 bytes (299651 compressed) at 0x00010000 in 4.7 secon
Hash of data verified.
Compressed 3072 bytes to 122...

Writing at 0x00008000... (100 %)
Wrote 3072 bytes (122 compressed) at 0x00008000 in 0.0 seconds (e
Hash of data verified.

Leaving...
Hard resetting...
```

Apply

Artemis

Express.js, or simply Express, is a back end web application framework for building RESTful APIs with Node.js



Hardware

PIN 2	SDA
PIN 3	SCL
5V	5V
GND	GND

Artemis is a system that scans QR code and sends data to the system in real time

Apply

Start creating embedded using Esp32-c5 + Unit QR Code

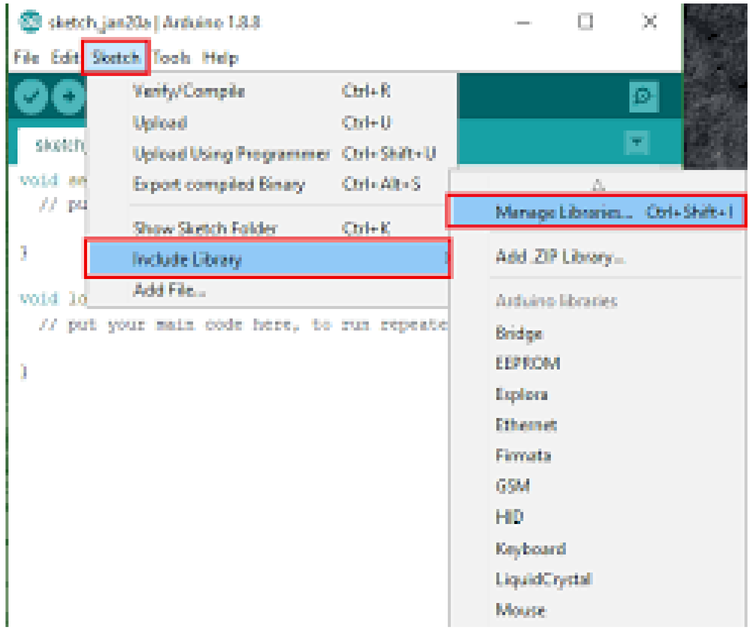
1. Create project Adruino

❖ Create folder project

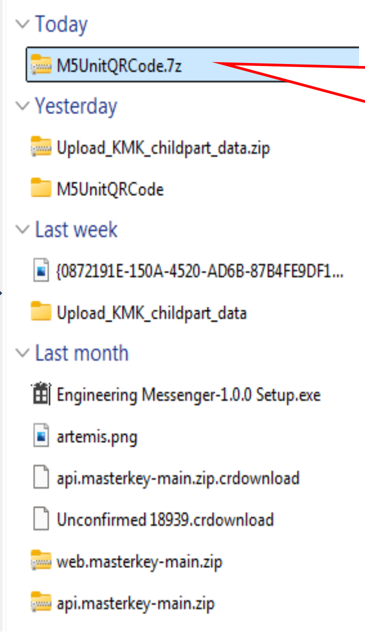
2. Install M5UnitQRCode library

❖ Link install: https://github.com/m5stack/M5Unit-QRCode

3. Add library to AdruinoIDE:


sketch_jan20a | Arduino 1.8.8
File Edit Sketch Tools Help
Verify/Compile Ctrl+R
Upload Ctrl+U
Upload Using Programmer Ctrl+Shift+U
Export compiled Binary Ctrl+Alt+S
Show Sketch Folder Ctrl+K
Include Library
Add File...
Manage Libraries... Ctrl+Shift+I
Add .ZIP Library...
Arduino libraries
Bridge
EEPROM
Esplora
Ethernet
Firmata
GSM
HID
Keyboard
LiquidCrystal
Mouse

```
void setup() {
  // put
}

void loop() {
  // put your main code here, to run repeatedly
}
```


Today
M5UnitQRCode.7z
Yesterday
Upload_KMK_childpart_data.zip
M5UnitQRCode
Last week
{0872191E-150A-4520-AD6B-87B4FE9DF1...
Upload_KMK_childpart_data
Last month
Engineering Messenger-1.0.0 Setup.exe
artemis.png
api.masterkey-main.zip.crdownload
Unconfirmed 18939.crdownload
web.masterkey-main.zip
api.masterkey-main.zip

Choose file to add library

Apply

Import library neccessary

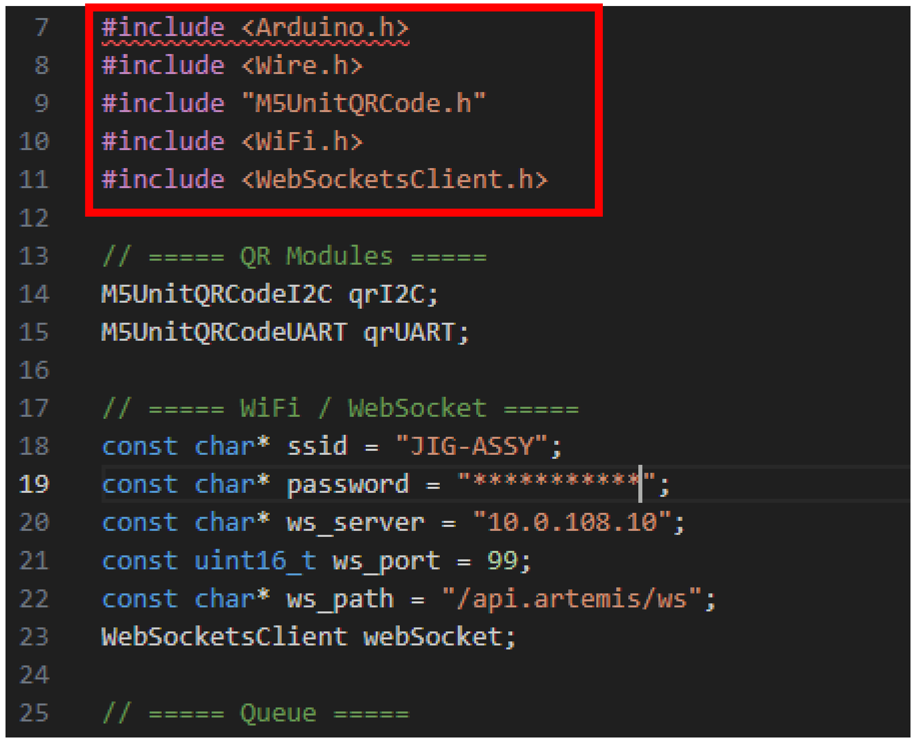
`Arduino.h`: This is the core library of Arduino, providing the most basic functions and constants for programming on the platform.

`Wire.h`: This library allows communication with devices using the I2C (Inter-Integrated Circuit) protocol.

`M5UnitQRCode.h`: This is a library specifically for the M5Stack UNIT QRCode module. This module is often used to read and decode QR codes.

`WiFi.h`: This library is used to connect the ESP32 module (or WiFi capable modules) to the wireless network.

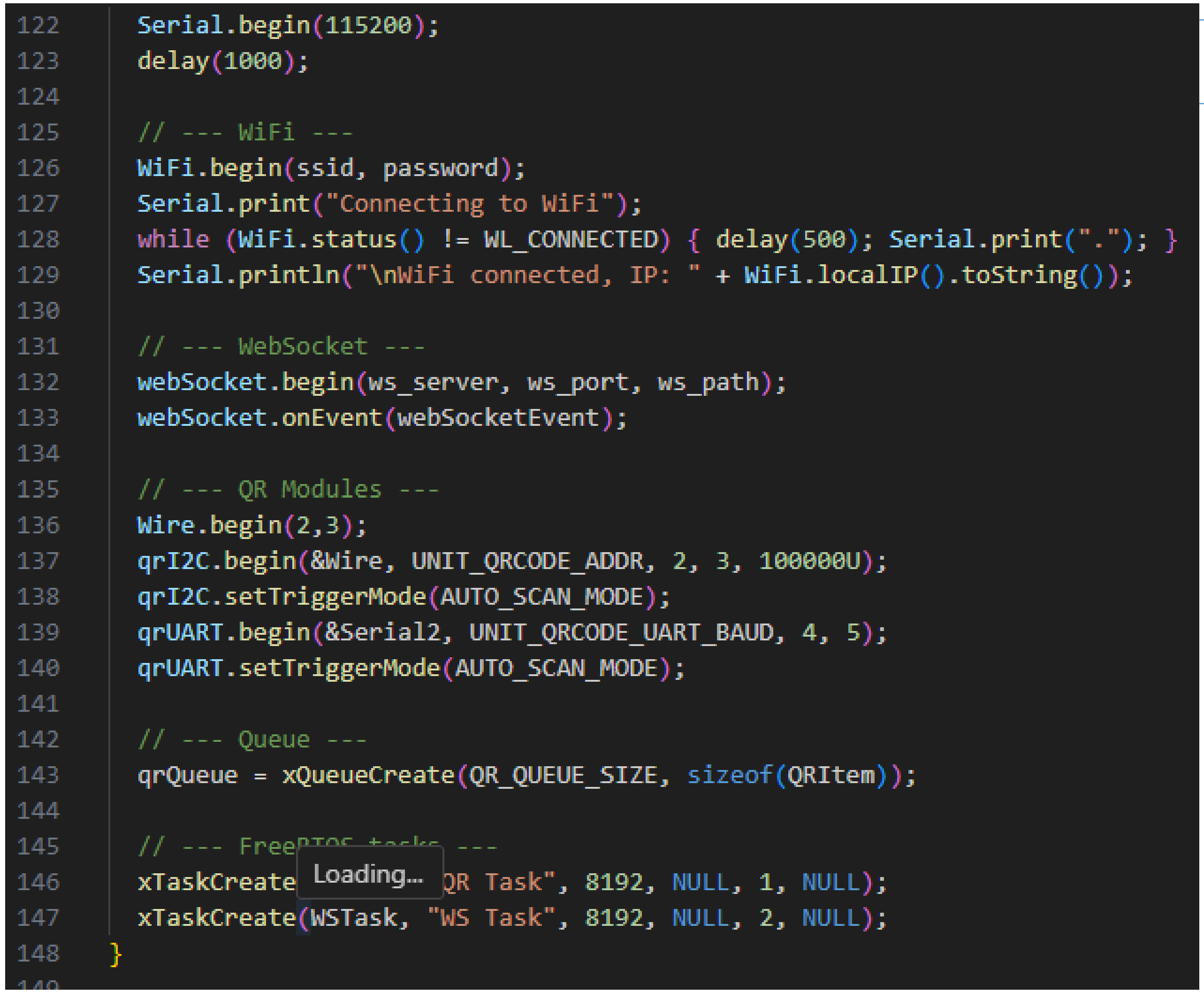
`WebSocketsClient.h`: This library provides the ability to connect devices to a WebSocket server.



```
#include <Arduino.h>
#include <Wire.h>
#include "M5UnitQRCode.h"
#include <WiFi.h>
#include <WebSocketsClient.h>

// ===== QR Modules =====
M5UnitQRCodeI2C qrI2C;
M5UnitQRCodeUART qrUART;

// ===== WiFi / WebSocket =====
const char* ssid = "JIG-ASSY";
const char* password = "***********";
const char* ws_server = "10.0.108.10";
const uint16_t ws_port = 99;
const char* ws_path = "/api.artemis/ws";
WebSocketsClient webSocket;

// ===== Queue =====
```

Apply

Set up object



```
Serial.begin(115200);
delay(1000);

// --- WiFi ---
WiFi.begin(ssid, password);
Serial.print("Connecting to WiFi");
while (WiFi.status() != WL_CONNECTED) { delay(500); Serial.print("."); }
Serial.println("\nWiFi connected, IP: " + WiFi.localIP().toString());

// --- WebSocket ---
webSocket.begin(ws_server, ws_port, ws_path);
webSocket.onEvent(webSocketEvent);

// --- QR Modules ---
Wire.begin(2,3);
qrI2C.begin(&Wire, UNIT_QRCODE_ADDR, 2, 3, 100000U);
qrI2C.setTriggerMode(AUTO_SCAN_MODE);
qrUART.begin(&Serial2, UNIT_QRCODE_UART_BAUD, 4, 5);
qrUART.setTriggerMode(AUTO_SCAN_MODE);

// --- Queue ---
qrQueue = xQueueCreate(QR_QUEUE_SIZE, sizeof(QRItem));

// --- FreeRTOS tasks ---
xTaskCreate Loading... QR Task", 8192, NULL, 1, NULL);
xTaskCreate(WSTask, "WS Task", 8192, NULL, 2, NULL);
}
```

Set up Buadrate

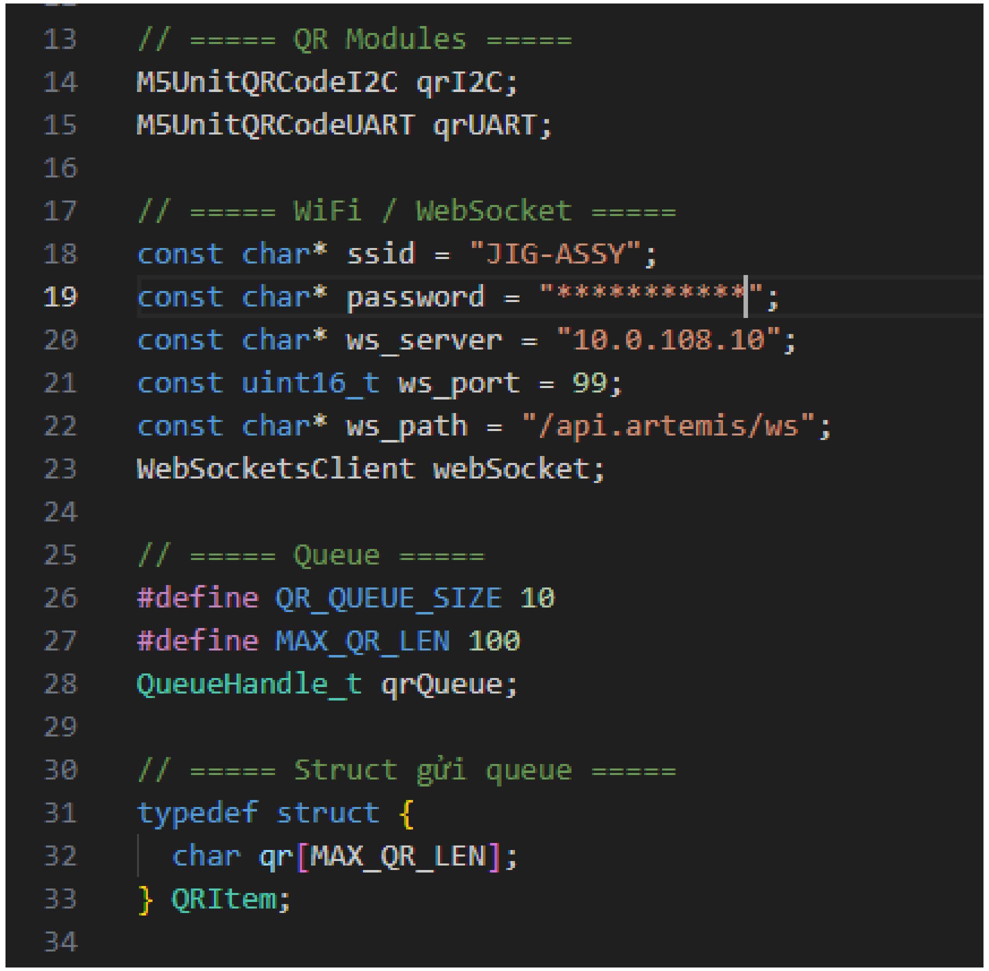
Set up connect Wifi

Set up UnitQRCode

Set up FreeRTOS

Apply

Define object



```
// ===== QR Modules =====
M5UnitQRCodeI2C qrI2C;
M5UnitQRCodeUART qrUART;

// ===== WiFi / WebSocket =====
const char* ssid = "JIG-ASSY";
const char* password = "***********";
const char* ws_server = "10.0.108.10";
const uint16_t ws_port = 99;
const char* ws_path = "/api.artemis/ws";
WebSocketsClient webSocket;

// ===== Queue =====
#define QR_QUEUE_SIZE 10
#define MAX_QR_LEN 100
QueueHandle_t qrQueue;

// ===== Struct gửi queue =====
typedef struct {
  char qr[MAX_QR_LEN];
} QRItem;

```

Declare and create objects (variables) to represent the QR code scanning module

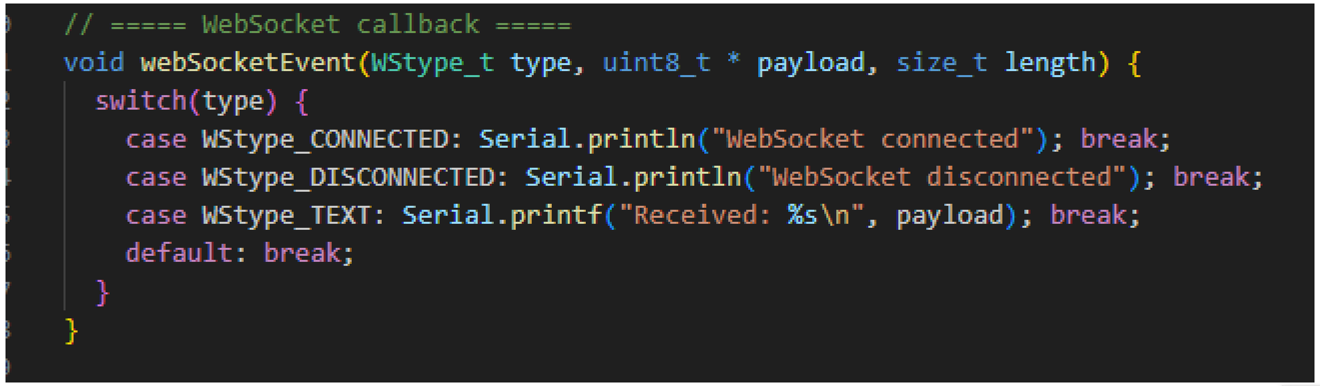
Declares the constants and an object needed so that the ESP32 can connect to a Wi-Fi network and then establish a WebSocket connection with a specific server.

The use of real-time operating system (RTOS) FreeRTOS, built into ESP32 microcontrollers, to manage and process QR code data effectively and safely.

Defines a structure named QRItem to contain the data of a QR code. This structure will then be used as the data type for the elements in the FreeRTOS queue.

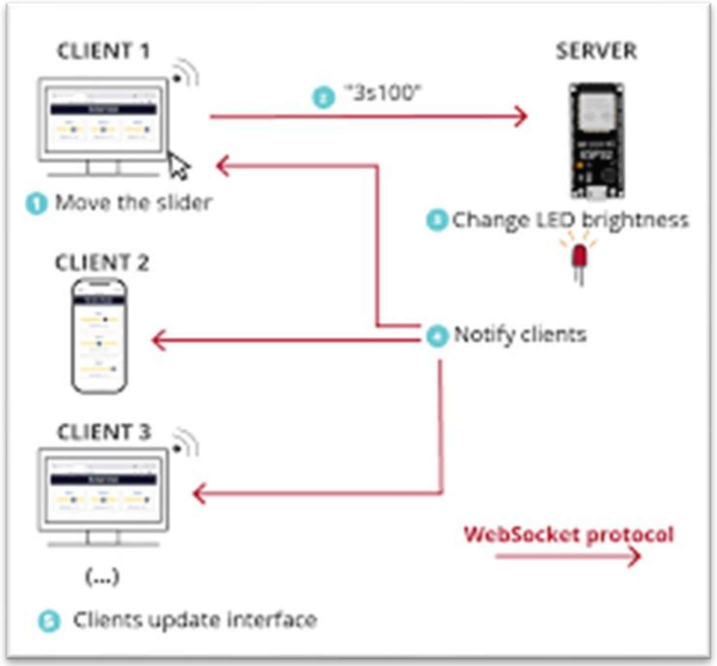
Apply

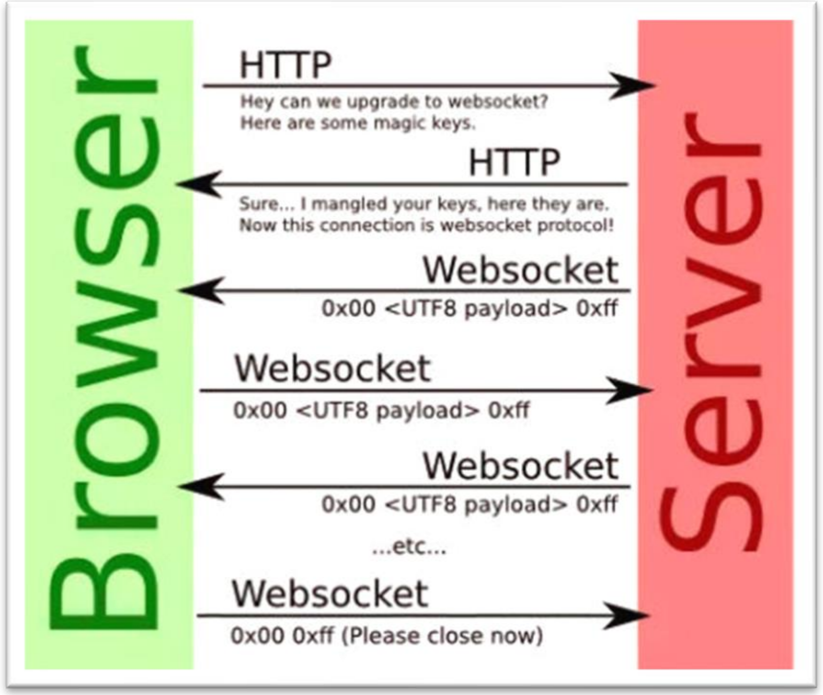
WebSocket callback



```
// ===== WebSocket callback =====
void webSocketEvent(WStype_t type, uint8_t * payload, size_t length) {
  switch(type) {
    case WStype_CONNECTED: Serial.println("WebSocket connected"); break;
    case WStype_DISCONNECTED: Serial.println("WebSocket disconnected"); break;
    case WStype_TEXT: Serial.printf("Received: %s\n", payload); break;
    default: break;
  }
}
```

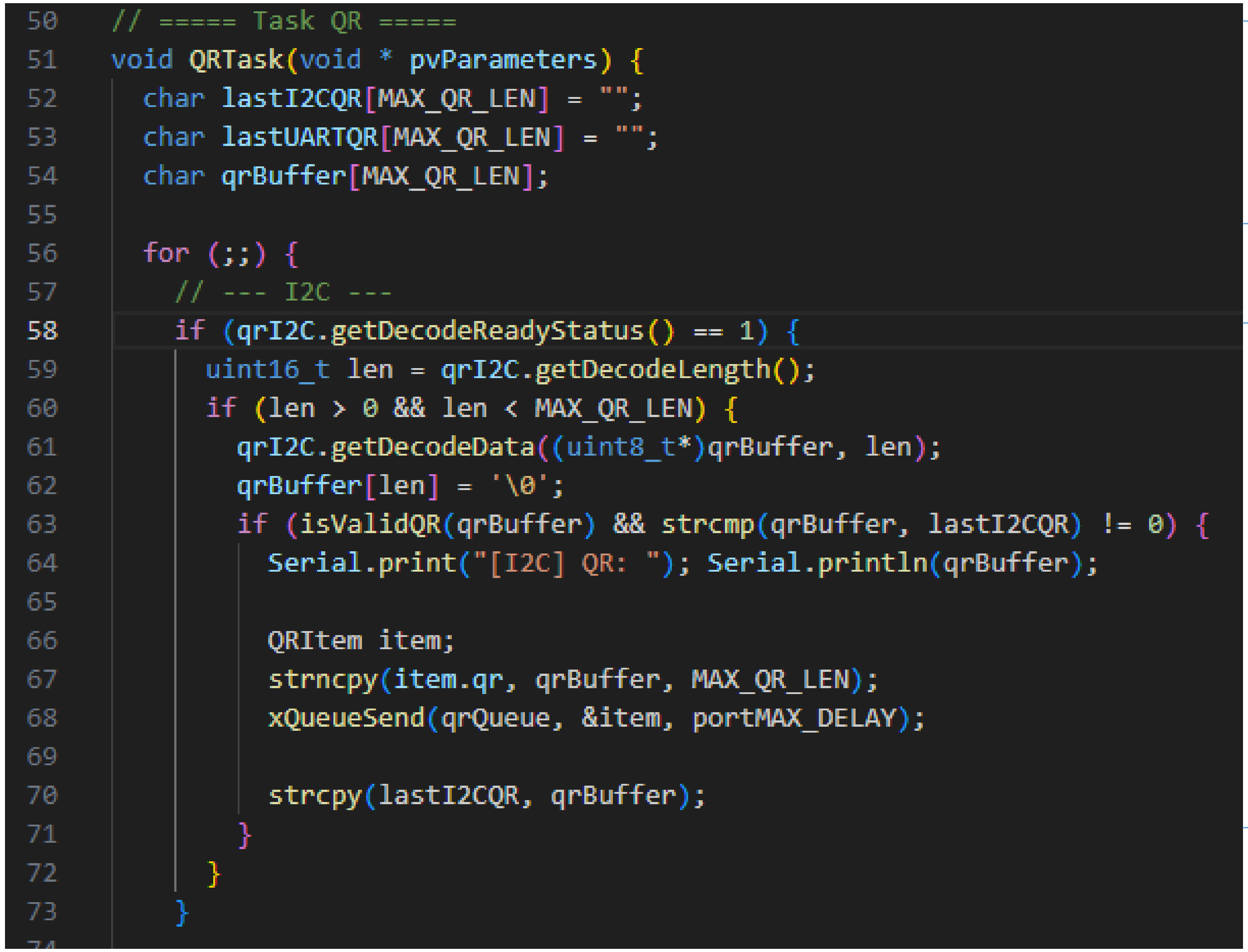
The webSocketEvent function is an event handler registered with the WebSocketsClient library to listen and react to various events that occur during WebSocket communication.




Browser
Server
HTTP
Hey can we upgrade to websocket?
Here are some magic keys.
HTTP
Sure... I mangled your keys, here they are.
Now this connection is websocket protocol!
Websocket
0x00 <UTF8 payload> 0xff
Websocket
0x00 <UTF8 payload> 0xff
Websocket
0x00 <UTF8 payload> 0xff
...etc...
Websocket
0x00 0xff (Please close now)

Apply

Task scan QR Code of FreeRTOS



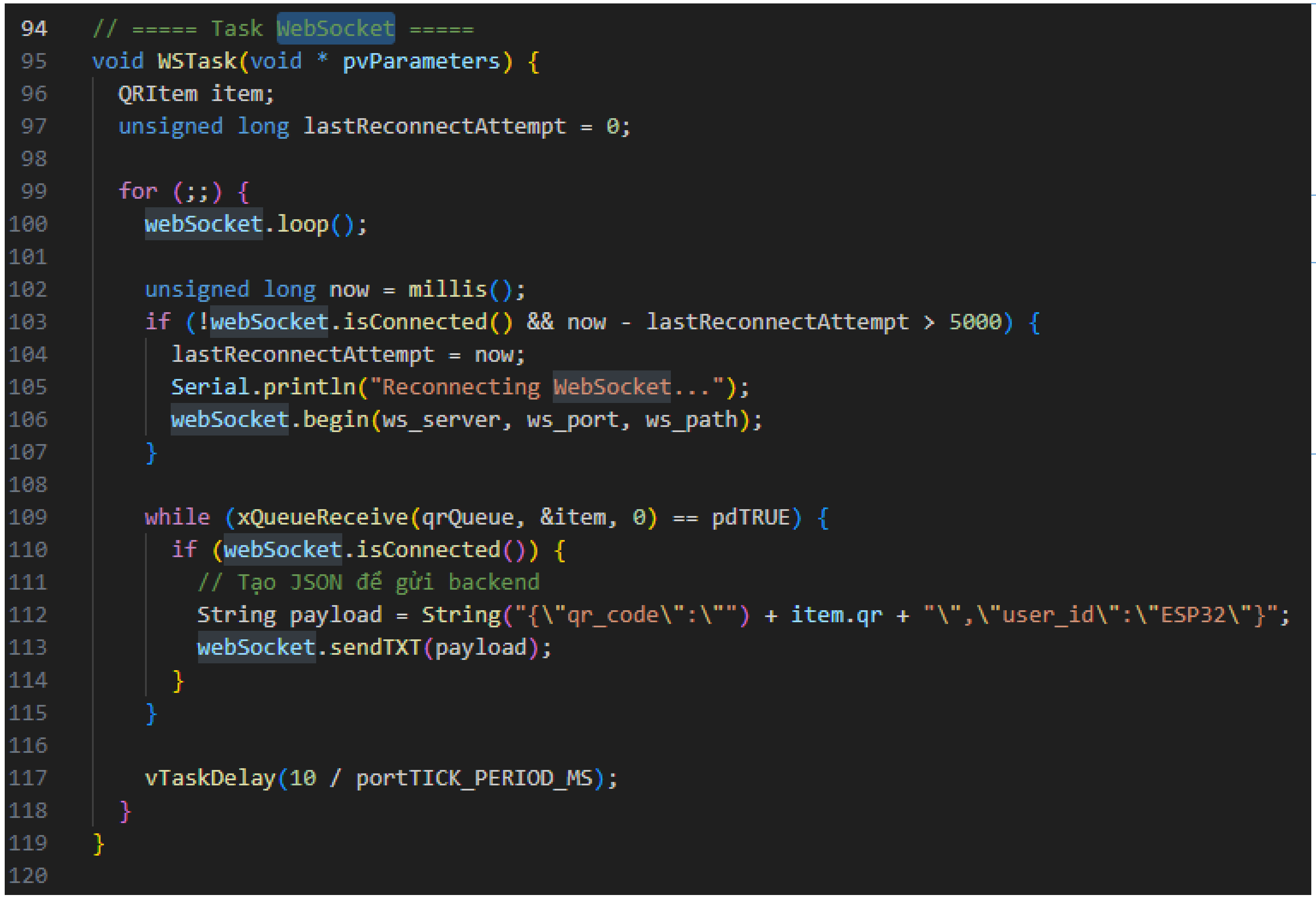
```
// ===== Task QR =====
void QRTask(void * pvParameters) {
  char lastI2CQR[MAX_QR_LEN] = "";
  char lastUARTQR[MAX_QR_LEN] = "";
  char qrBuffer[MAX_QR_LEN];

  for (;;) {
    // --- I2C ---
    if (qrI2C.getDecodeReadyStatus() == 1) {
      uint16_t len = qrI2C.getDecodeLength();
      if (len > 0 && len < MAX_QR_LEN) {
        qrI2C.getDecodeData((uint8_t*)qrBuffer, len);
        qrBuffer[len] = '\0';
        if (isValidQR(qrBuffer) && strcmp(qrBuffer, lastI2CQR) != 0) {
          Serial.print("[I2C] QR: "); Serial.println(qrBuffer);

          QRItem item;
          strncpy(item.qr, qrBuffer, MAX_QR_LEN);
          xQueueSend(qrQueue, &item, portMAX_DELAY);

          strcpy(lastI2CQR, qrBuffer);
        }
      }
    }
```

Create variables to store data

Retrieve and perform data processing

Apply

Task Websocket of FreeRTOS



```
// ===== Task WebSocket =====
void WSTask(void * pvParameters) {
  QRItem item;
  unsigned long lastReconnectAttempt = 0;

  for (;;) {
    webSocket.loop();

    unsigned long now = millis();
    if (!webSocket.isConnected() && now - lastReconnectAttempt > 5000) {
      lastReconnectAttempt = now;
      Serial.println("Reconnecting WebSocket...");
      webSocket.begin(ws_server, ws_port, ws_path);
    }

    while (xQueueReceive(qrQueue, &item, 0) == pdTRUE) {
      if (webSocket.isConnected()) {
        // Tạo JSON để gửi backend
        String payload = String("{\"qr_code\":\"") + item.qr + "\",\"user_id\":\"ESP32\"}";
        webSocket.sendTXT(payload);
      }
    }

    vTaskDelay(10 / portTICK_PERIOD_MS);
  }
}
```

Initial declaration for the task (Task) that handles WebSocket communication (named WSTask)

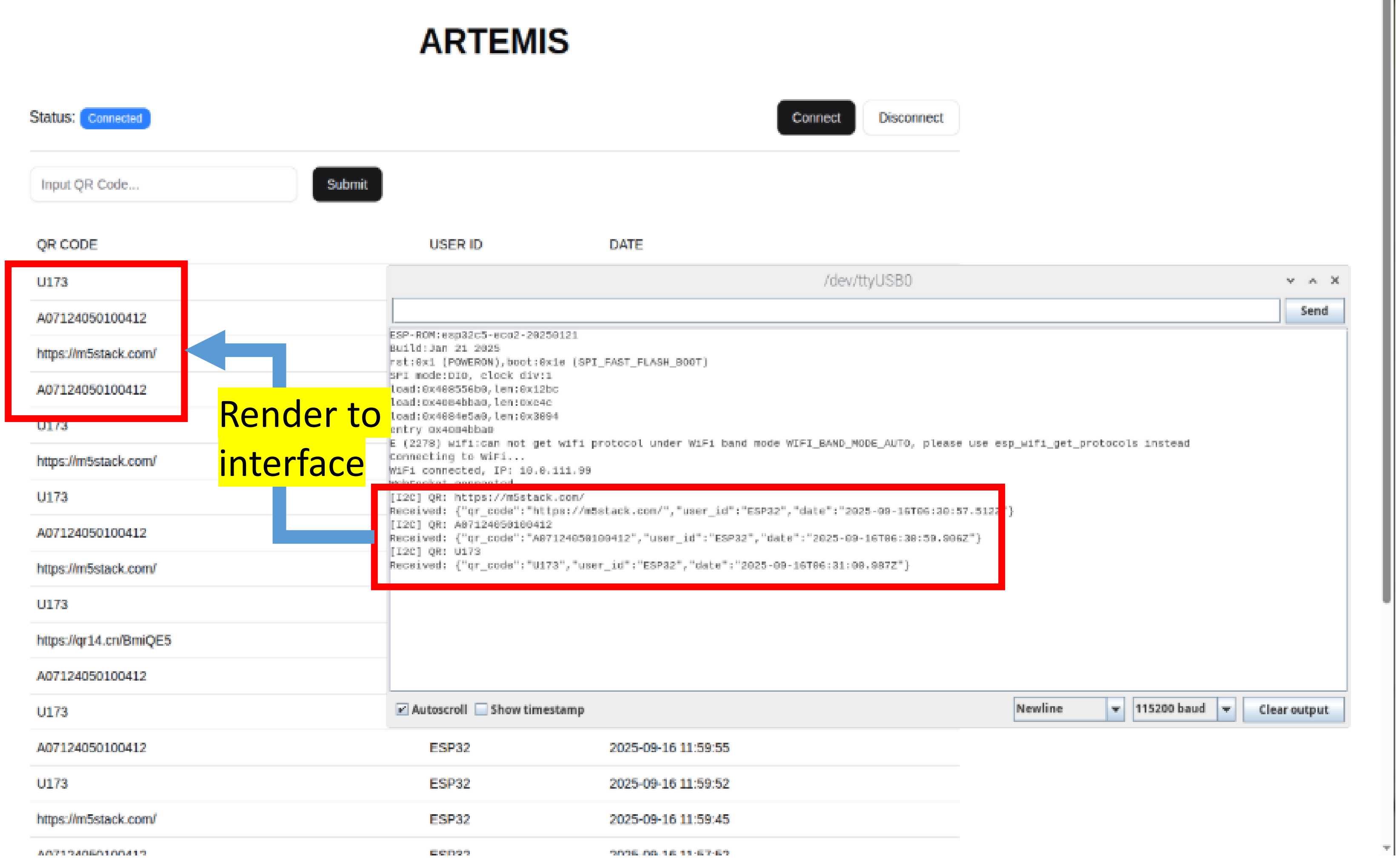
Check connect with Websocket

Send data to back end

Apply

Result

Link project: Artemis_esp32/esp32_C5 at main - Artemis_esp32 - Gitea: Git with a cup of tea



ARTEMIS

Status: Connected

Connect | Disconnect

Input QR Code... | Submit

QR CODE	USER ID	DATE
U173		
A07124050100412		
https://m5stack.com/		
A07124050100412		
U173		
https://m5stack.com/		
U173		
A07124050100412		
https://m5stack.com/		
U173		
https://qr14.cn/BmiQE5		
A07124050100412		
U173		
A07124050100412	ESP32	2025-09-16 11:59:55
U173	ESP32	2025-09-16 11:59:52
https://m5stack.com/	ESP32	2025-09-16 11:59:45
A07124050100412	ESP32	2025-09-16 11:57:52

Render to interface

/dev/ttyUSB0

Send

```
ESP-ROM:esp32c5-eco2-20250121
Build:Jan 21 2025
rst:0x1 (POWERON),boot:0x1e (SPI_FAST_FLASH_BOOT)
SPI mode:DIO, clock div:1
load:0x408556b0,len:0x12bc
load:0x4084bba0,len:0xc4c
load:0x4084e5a0,len:0x3004
entry 0x4084bba0
E (2278) wifi:can not get wifi protocol under WiFi band mode WIFI_BAND_MODE_AUTO, please use esp_wifi_get_protocols instead
connecting to WiFi...
WiFi connected, IP: 10.0.111.99
WebSocket connected
[I2C] QR: https://m5stack.com/
Received: {"qr_code":"https://m5stack.com/","user_id":"ESP32","date":"2025-09-16T06:30:57.512Z"}
[I2C] QR: A07124050100412
Received: {"qr_code":"A07124050100412","user_id":"ESP32","date":"2025-09-16T06:30:59.906Z"}
[I2C] QR: U173
Received: {"qr_code":"U173","user_id":"ESP32","date":"2025-09-16T06:31:08.987Z"}
```

☑ Autoscroll ☐ Show timestamp | Newline | 115200 baud | Clear output thank you!

1. What is a key advantage of ESP32-C5?
A. Only supports 2.4 GHz Wi-Fi
B. Supports Wi-Fi 6 dual-band (2.4 & 5 GHz)
C. Does not support Bluetooth
Correct Answer: B

2. What type of communication is UART on ESP32-C5?
A. Synchronous with clock line
B. Asynchronous without clock line
C. Only for USB
Correct Answer: B

3. In I2C, what do SDA and SCL do?
A. SDA sends data, SCL provides clock
B. SDA provides clock, SCL sends data
C. Both send data
Correct Answer: A

4. What is the main purpose of GPIO pins?
A. Debugging only
B. General input/output for signals
C. Power supply
Correct Answer: B

5. Which library is used for Wi-Fi in Arduino for ESP32?
A. Wire.h
B. WiFi.h
C. WebSocketsClient.h
Correct Answer: B