M5STACK

PowerHub

1. Description

PowerHub is an integrated programmable power management controller.

It features the ESP32-S3-WROOM-1U-N16R2 as its main control unit, equipped with a dual-core Xtensa LX7 processor running at 240MHz and supporting 2.4GHz Wi-Fi.

An integrated co-processor, the STM32G031G8U6, works with multiple INA226 voltage/current sensor ICs and electronic switching circuitry. This enables precise power management for multiple expansion interfaces and provides system-wide low-power wake-up capability.

The controller integrates two HY2.0-4P interfaces (supporting I2C + UART), one RS485 communication interface, and one CAN bus communication interface, facilitating easy connection to various sensors and actuators.

It includes both USB-A and USB Type-C ports. Leveraging the ESP32-S3's USB OTG peripheral, these ports support both USB Host and Device functionality.

Designed for applications like industrial automation control, smart home hubs, and IoT edge devices, the PowerHub provides developers with a stable and reliable core control solution.

2. Specifications

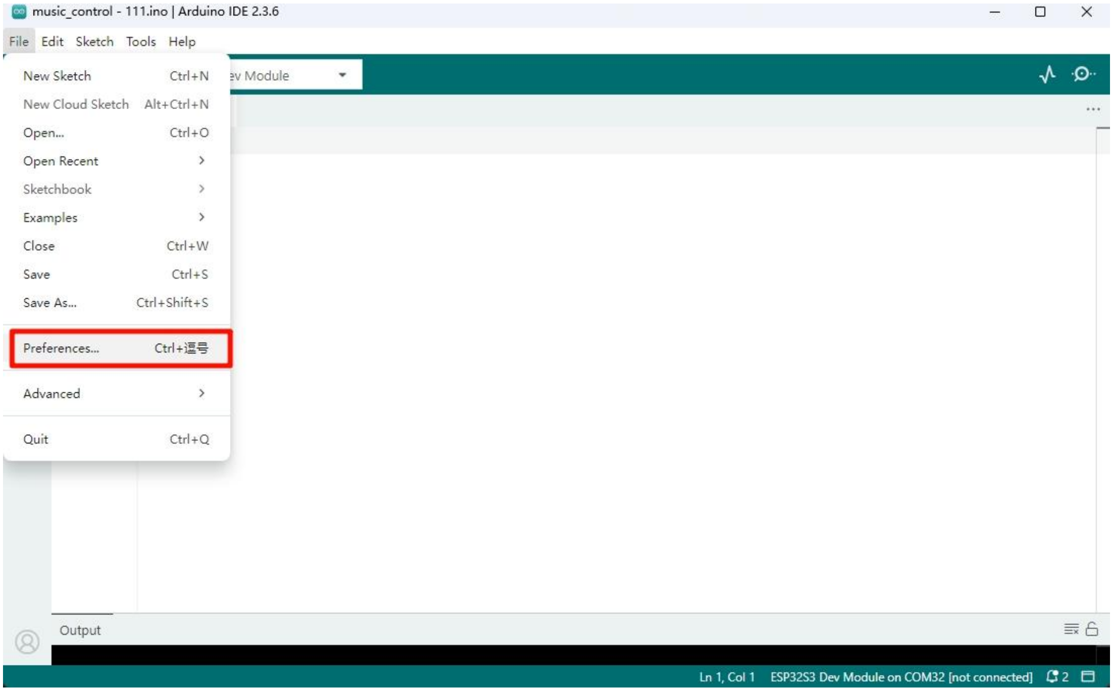
Specification	Parameters
SoC	ESP32-S3-WROOM-1U-N16R2, with Xtensa® 32-bit LX7 dual-core processor (max 240MHz CPU frequency)
PSRAM	2MB
Flash	16MB
Wi-Fi	2.4 GHz
Wireless Antenna	Specifications：15 x 14MM SMA Internal Threaded Hole Frequency：2.4 GHz Gain：-6.41 dBi VSWR (Voltage Standing Wave Ratio)：≤1.8
Dimensions	88.0 x 56.0 x 38.5 mm

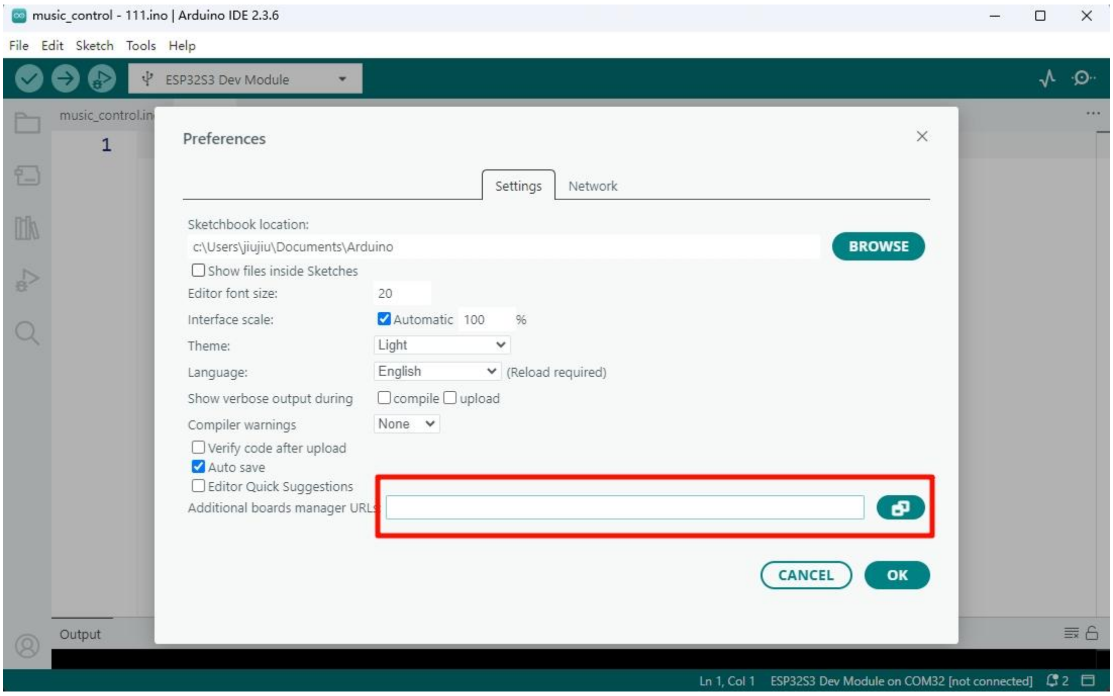
3. Quick Start

3.1 Preparation

1. Visit the official Arduino website and install the Arduino IDE
https://www.arduino.cc/en/Main/Software

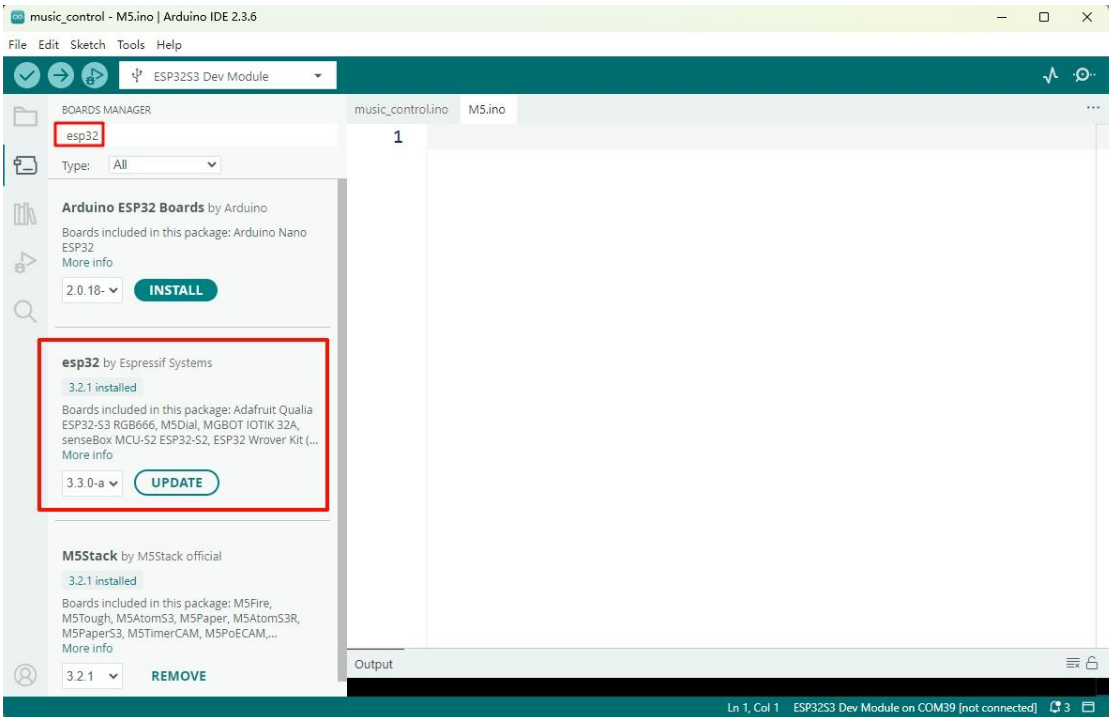
2. Add the following Board Manager URL to File → Preferences → Additional→Boards Manager URLs:
https://espressif.github.io/arduino-esp32/package_esp32_dev_index.json




music_control - 111.ino | Arduino IDE 2.3.6
File Edit Sketch Tools Help
ESP32S3 Dev Module
music_control.in
Preferences
Settings
Network
BROWSE
CANCEL
OK
Output
Ln 1, Col 1 ESP32S3 Dev Module on COM32 [not connected] 2

Sketchbook location:
c:\Users\jiujiu\Documents\Arduino
□ Show files inside Sketches
Editor font size: 20
Interface scale: ☑ Automatic 100 %
Theme: Light
Language: English (Reload required)
Show verbose output during □ compile □ upload
Compiler warnings None
□ Verify code after upload
☑ Auto save
□ Editor Quick Suggestions
Additional boards manager URLs

3.Open the Boards Manager, search for "ESP32" ,and click install.


music_control - M5.ino | Arduino IDE 2.3.6
File Edit Sketch Tools Help
ESP32S3 Dev Module
BOARDS MANAGER
music_control.ino
M5.ino
INSTALL
UPDATE
REMOVE
Output
Ln 1, Col 1 ESP32S3 Dev Module on COM39 [not connected] 3

esp32
Type: All

Arduino ESP32 Boards by Arduino
Boards included in this package: Arduino Nano ESP32
More info
2.0.18-

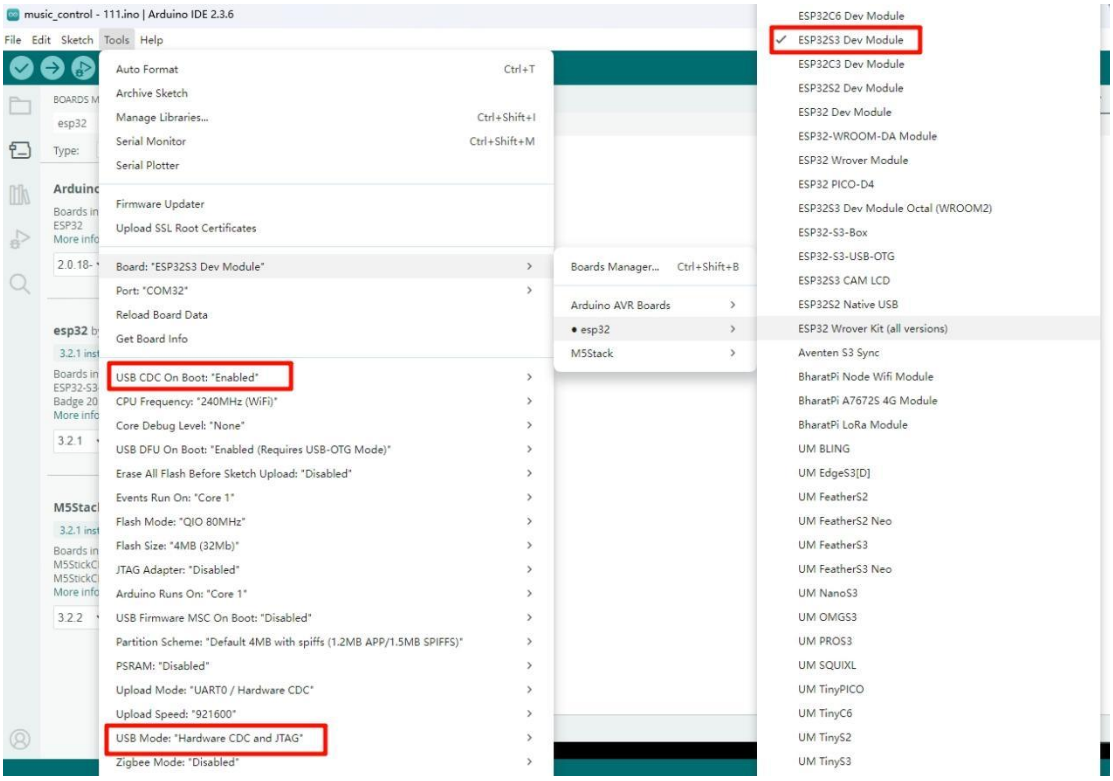
esp32 by Espressif Systems
3.2.1 installed
Boards included in this package: Adafruit Qualia ESP32-S3 RGB666, M5Dial, MGBOT IOTIK 32A, senseBox MCU-S2 ESP32-S2, ESP32 Wrover Kit (...
More info
3.3.0-a

M5Stack by M5Stack official
3.2.1 installed
Boards included in this package: M5Fire, M5Tough, M5AtomS3, M5Paper, M5AtomS3R, M5PaperS3, M5TimerCAM, M5PoECAM,...
More info
3.2.1

4.After installation, select the board "ESP32S3 DevModule".

5.Configure the following options:

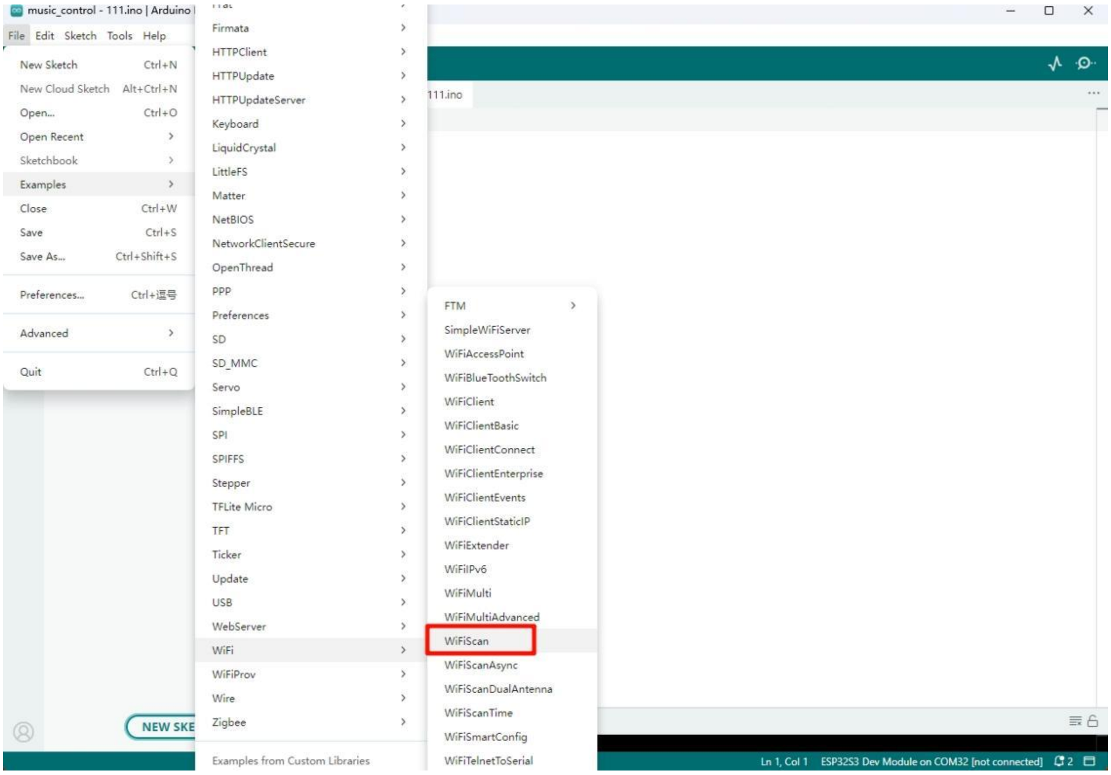
1. **USB CDC On Boot**: "Enabled"
2. **USB Mode**: "Hardware CDC and JTAG"

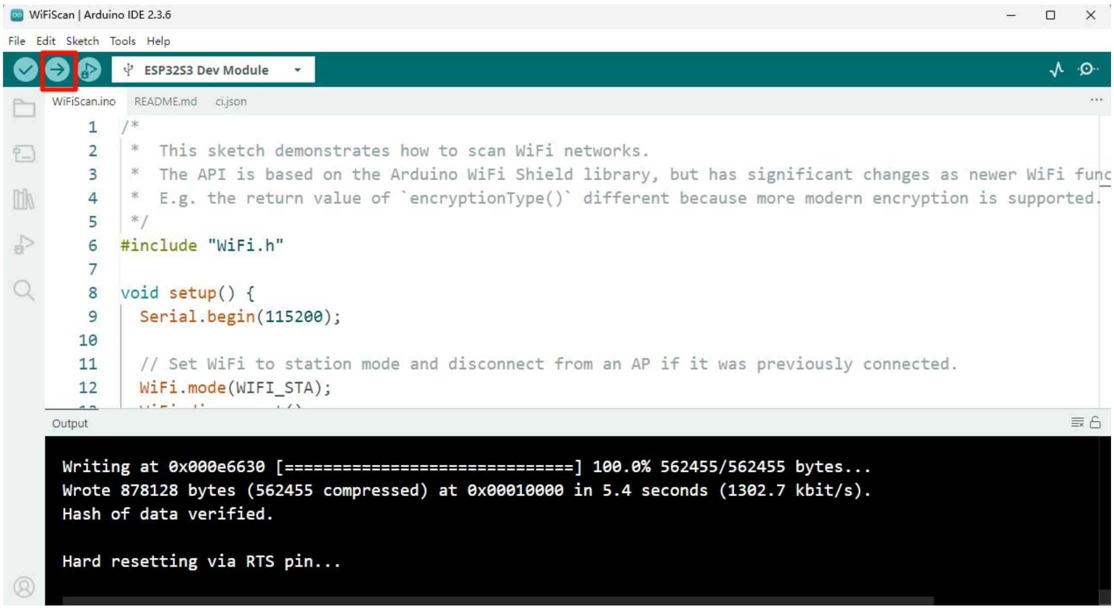


Board: "ESP32S3 Dev Module"
Port: "COM32"
USB CDC On Boot: "Enabled"
CPU Frequency: "240MHz (WiFi)"
Core Debug Level: "None"
USB DFU On Boot: "Enabled (Requires USB-OTG Mode)"
Erase All Flash Before Sketch Upload: "Disabled"
Events Run On: "Core 1"
Flash Mode: "QIO 80MHz"
Flash Size: "4MB (32Mb)"
JTAG Adapter: "Disabled"
Arduino Runs On: "Core 1"
USB Firmware MSC On Boot: "Disabled"
Partition Scheme: "Default 4MB with spiffs (1.2MB APP/1.5MB SPIFFS)"
PSRAM: "Disabled"
Upload Mode: "UART0 / Hardware CDC"
Upload Speed: "921600"
USB Mode: "Hardware CDC and JTAG"
Zigbee Mode: "Disabled"

3.2 Wi-Fi Test

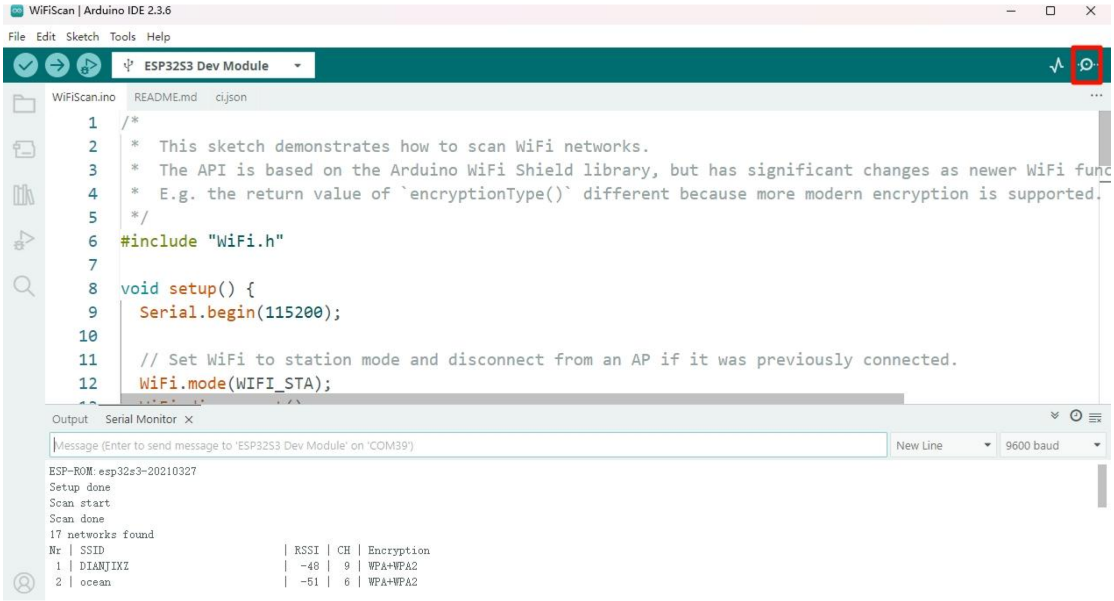
Select the example program "Examples" → "WiFi" → "WiFiScan", choose the port corresponding to your device, and click the compile and upload button in the top-left corner. After uploading is complete, open the Serial Monitor to view Wi-Fi scan information.




WiFiScan | Arduino IDE 2.3.6
File Edit Sketch Tools Help
ESP32S3 Dev Module
WiFiScan.ino README.md ci.json
Output

```
/*
 *  This sketch demonstrates how to scan WiFi networks.
 *  The API is based on the Arduino WiFi Shield library, but has significant changes as newer WiFi func
 *  E.g. the return value of `encryptionType()` different because more modern encryption is supported.
 */
#include "WiFi.h"

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

  // Set WiFi to station mode and disconnect from an AP if it was previously connected.
  WiFi.mode(WIFI_STA);
```

```
Writing at 0x000e6630 [==============================] 100.0% 562455/562455 bytes...
Wrote 878128 bytes (562455 compressed) at 0x00010000 in 5.4 seconds (1302.7 kbit/s).
Hash of data verified.

Hard resetting via RTS pin...
```


WiFiScan | Arduino IDE 2.3.6
File Edit Sketch Tools Help
ESP32S3 Dev Module
WiFiScan.ino
README.md
ci.json
Output
Serial Monitor
Message (Enter to send message to 'ESP32S3 Dev Module' on 'COM39')
New Line
9600 baud

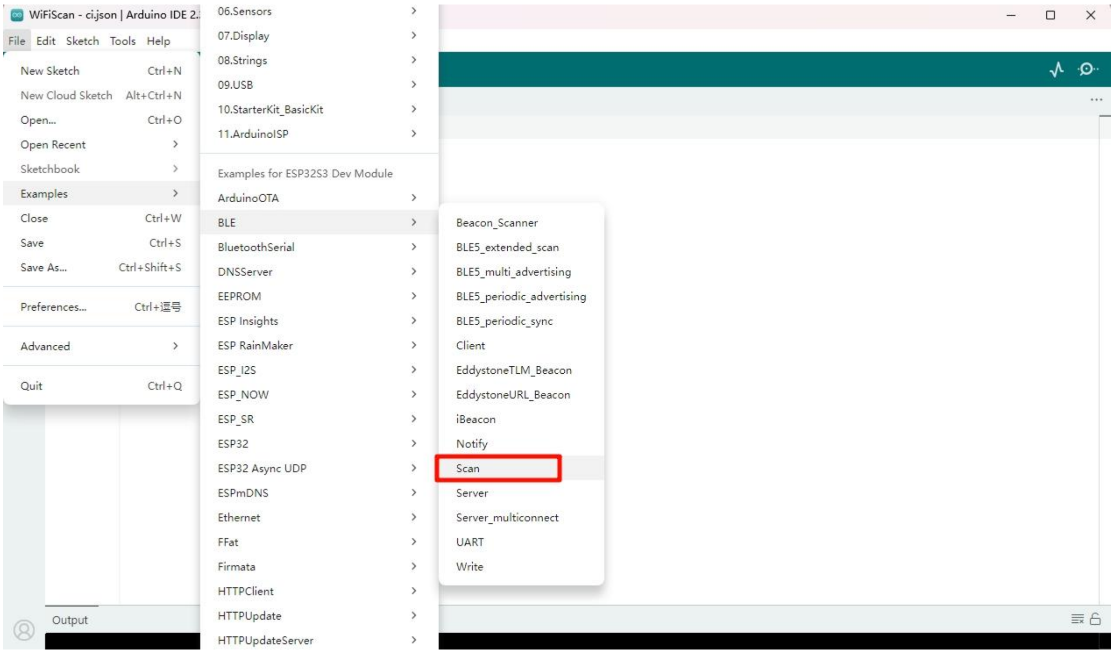
```
/*
 *  This sketch demonstrates how to scan WiFi networks.
 *  The API is based on the Arduino WiFi Shield library, but has significant changes as newer WiFi func
 *  E.g. the return value of `encryptionType()` different because more modern encryption is supported.
 */
#include "WiFi.h"

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

  // Set WiFi to station mode and disconnect from an AP if it was previously connected.
  WiFi.mode(WIFI_STA);
```

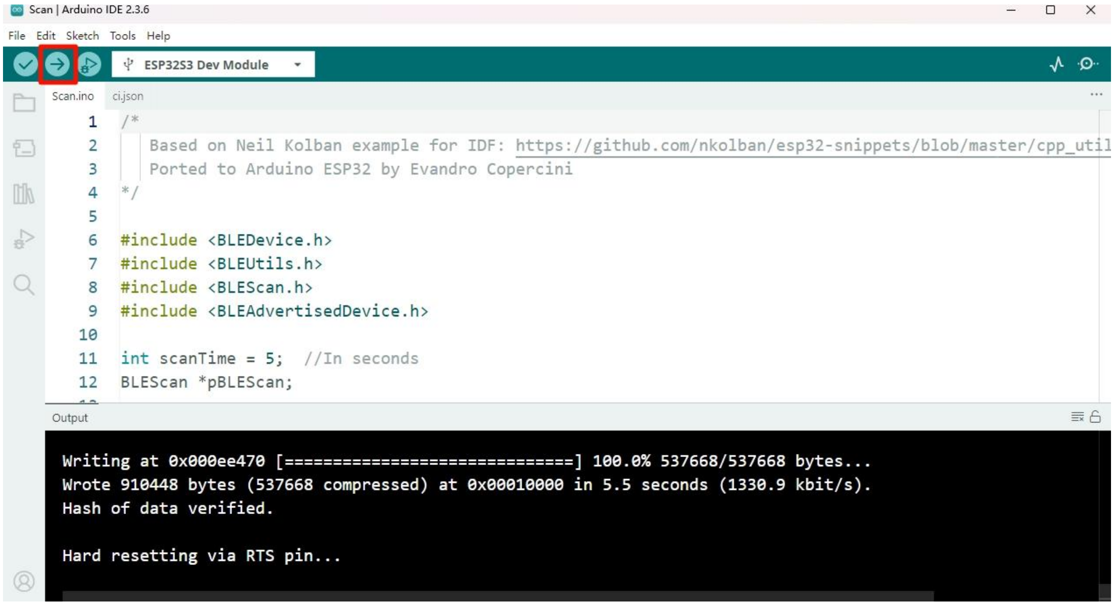
```
ESP-ROM:esp32s3-20210327
Setup done
Scan start
Scan done
17 networks found
Nr | SSID                             | RSSI | CH | Encryption
 1 | DIANJIXZ                         |  -48 |  9 | WPA+WPA2
 2 | ocean                            |  -51 |  6 | WPA+WPA2
```

3.3 BLE Test

Select the example program "Examples"-"BLE"-"Scan", choose the port corresponding to your device, and click the compile and upload button in the top-left corner.

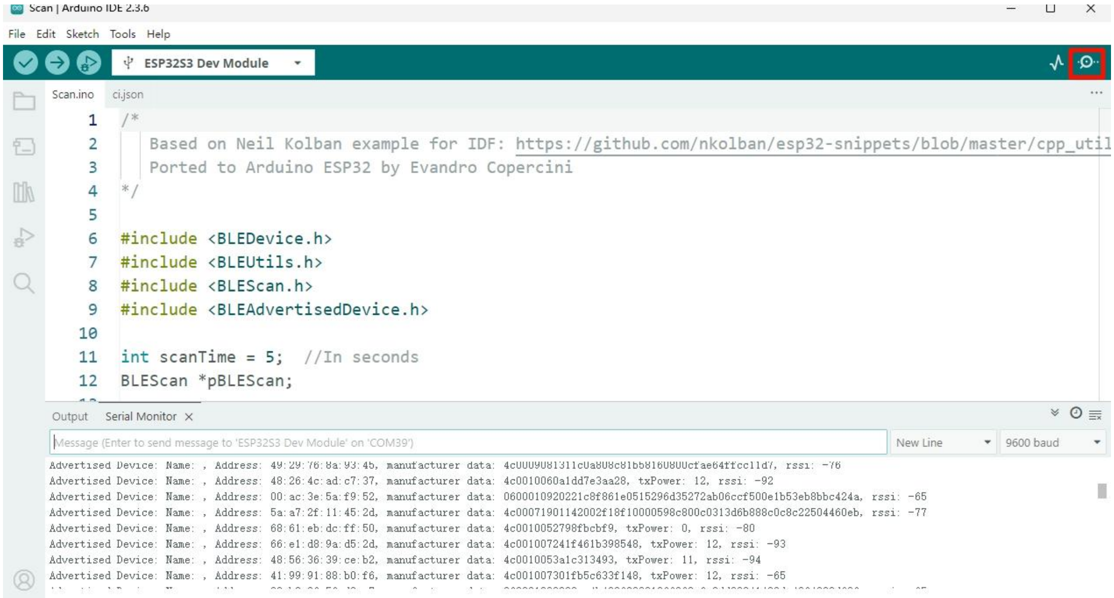
After uploading is complete, open the Serial Monitor to view BLE scan information.





```
/*
  Based on Neil Kolban example for IDF: https://github.com/nkolban/esp32-snippets/blob/master/cpp_util
  Ported to Arduino ESP32 by Evandro Copercini
*/

#include <BLEDevice.h>
#include <BLEUtils.h>
#include <BLEScan.h>
#include <BLEAdvertisedDevice.h>

int scanTime = 5;  //In seconds
BLEScan *pBLEScan;
```

```
Writing at 0x000ee470 [==============================] 100.0% 537668/537668 bytes...
Wrote 910448 bytes (537668 compressed) at 0x00010000 in 5.5 seconds (1330.9 kbit/s).
Hash of data verified.

Hard resetting via RTS pin...
```



```
/*
  Based on Neil Kolban example for IDF: https://github.com/nkolban/esp32-snippets/blob/master/cpp_util
  Ported to Arduino ESP32 by Evandro Copercini
*/

#include <BLEDevice.h>
#include <BLEUtils.h>
#include <BLEScan.h>
#include <BLEAdvertisedDevice.h>

int scanTime = 5;  //In seconds
BLEScan *pBLEScan;
```

```
Advertised Device: Name: , Address: 49:29:76:8a:93:45, manufacturer data: 4c0009081311c0a808c81b58160800cfae64ffcc11d7, rssi: -76
Advertised Device: Name: , Address: 48:26:4c:ad:c7:37, manufacturer data: 4c0010060a1dd7e3aa28, txPower: 12, rssi: -92
Advertised Device: Name: , Address: 00:ac:3e:5a:f9:52, manufacturer data: 0600010920221c8f861e0515296d35272ab06ccf500e1b53eb8bbc424a, rssi: -65
Advertised Device: Name: , Address: 5a:a7:2f:11:45:2d, manufacturer data: 4c00071901142002f18f10000598c800c0313d6b888c0c8c22504460eb, rssi: -77
Advertised Device: Name: , Address: 68:61:eb:dc:ff:50, manufacturer data: 4c0010052798fbcbf9, txPower: 0, rssi: -80
Advertised Device: Name: , Address: 66:e1:d8:9a:d5:2d, manufacturer data: 4c001007241f461b398548, txPower: 12, rssi: -93
Advertised Device: Name: , Address: 48:56:36:39:ce:b2, manufacturer data: 4c0010053a1c313493, txPower: 11, rssi: -94
Advertised Device: Name: , Address: 41:99:91:88:b0:f6, manufacturer data: 4c001007301fb5c633f148, txPower: 12, rssi: -65
```

FCC Warning

This device complies with part 15 of the FCC Rules. Operation is subject to the following two conditions: (1) This device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

Any Changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate the equipment.

Note: This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

-Reorient or relocate the receiving antenna.

-Increase the separation between the equipment and receiver.

-Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.

-Consult the dealer or an experienced radio/TV technician for help.

This equipment complies with FCC radiation exposure limits set forth for an uncontrolled environment. This equipment should be installed and operated with minimum distance 20cm between the radiator & your body.