M5STACK

NanoC6

2023

CONTENTS

1. OUTLINE

M5NanoC6 is a miniature, low-power IoT development board within the M5Stack development kit series. Powered by the ESP32-C6 MCU, it boasts advanced wireless communication support, including Wi-Fi 6 and Zigbee, facilitating seamless control of infrared IoT devices through its built-in infrared transmitter. The onboard ceramic antenna ensures a stable wireless communication connection. Additionally, the device features programmable RGB LEDs, adding a personalized visual touch to projects. The inclusion of Grove interfaces allows M5NanoC6 to flexibly expand with various M5 devices, supporting the connection of different device types through protocols such as UART and I2C. This provides developers with abundant hardware expansion possibilities. Suitable for applications in smart homes, industrial automation, health monitoring, and IoT devices, M5NanoC6 delivers a comprehensive development solution for innovative projects.

1.1. ESP32-C6

1. Communication Capabilities:
    - Supports 2.4 GHz Wi-Fi 6 (802.11 ax): Provides high-speed and efficient Wi-Fi communication.
    - Bluetooth® 5 (LE): Incorporates Bluetooth 5.0 technology for extended wireless coverage and faster data transmission.
    - Zigbee and Thread (802.15.4): Supports Zigbee and Thread communication protocols, offering flexible connectivity for IoT applications.
2. Processor and Performance:
    - RISC-V 32-bit single-core processor: Delivers a highly flexible and scalable processor architecture.
    - Up to 160 MHz clock frequency: Ensures the device has fast and efficient data processing capabilities.
    - Industry-leading low-power and RF performance: Achieves a leading position in both power consumption and RF performance.
3. Memory:
    - Built-in 320 KB ROM: Used for storing firmware or program code.
    - 512 KB SRAM: Utilized for runtime data storage.
    - 16 KB low-power SRAM: Specifically designed for low-power operations.
    - Supports external Flash: Can extend the storage space for program code through an external memory.
4. GPIO Pins and Programmable Interfaces:
    - Supports SPI, UART, I2C, I2S, RMT, TWAI, and PWM: Multiple communication interfaces, enhancing flexibility for data exchange with other devices.

2. SPECIFICATIONS

Specification	Parameter
MCU	ESP32-C6FH4@RISC-V 32-bit single-core processor 160 MHZ, 320 KB ROM, 512 KB SRAM, 16 KB low-power SRAM, supports external flash
Programmable RGB	WS2812-2020
Input voltage	5V
Communication mode	2.4 GHz Wi-Fi 6 (802.11 ax)、Zigbee And Thread (802.15.4)And other mainstream communication methods
Communication Protocol	Support SPI、UART、I2C、I2S、RMT、TWAI And PWM
Antenna type	Onboard ceramic antenna
Other peripherals	Board infrared transmitter, programmable onboard control buttons
Operating temperature	0-40°C

FCC Caution:
Any changes or modifications not expressly approved by the party responsible for compliance could void the user's authority to operate the equipment.
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.
IMPORTANT NOTE:
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.

FCC Radiation Exposure Statement :
This equipment complies with FCC radiation exposure limits set forth for an uncontrolled environment. The SAR was tested for the device in thebody worn mode with 5mm distance. And it can meet the SAR limit of FCC.

3. QUICK START

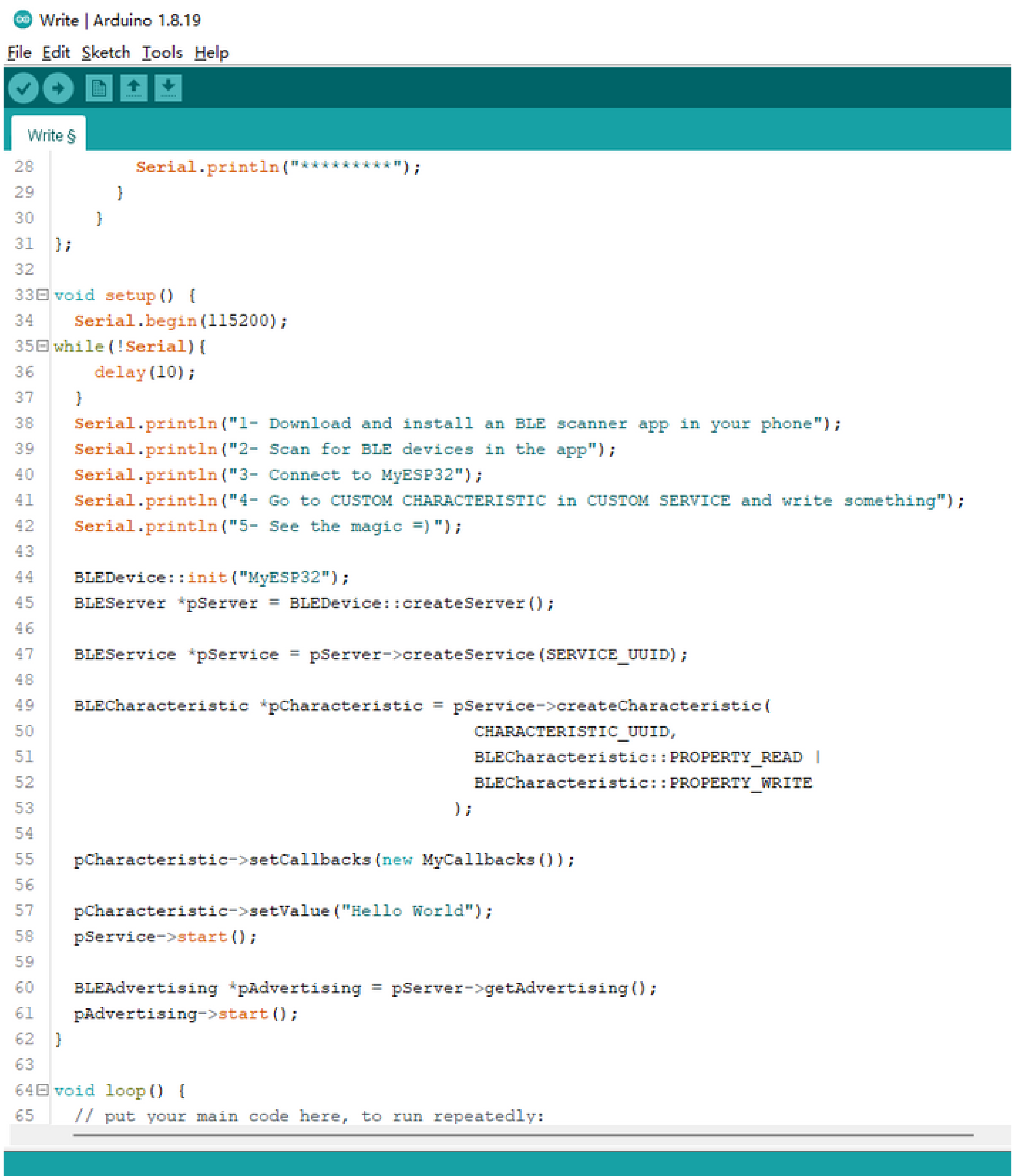
3.1. ARDUINO IDE

To access the Arduino official website (https://www.arduino.cc/en/Main/Software) and download the installation package for your operating system, follow these steps:

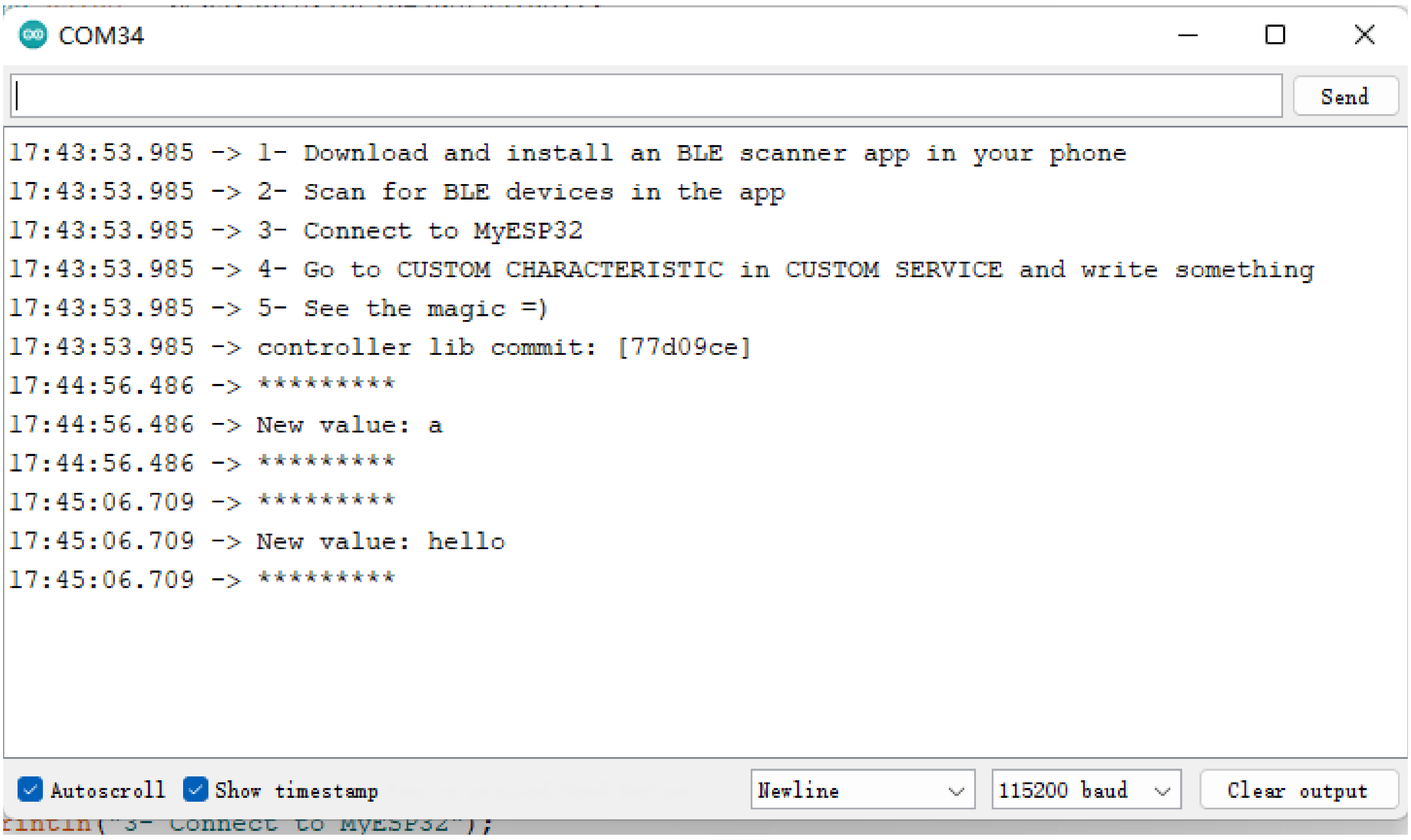
1. Open the Arduino IDE and navigate to **File** -> **Preferences** -> **Settings**.
2. Copy the following M5Stack Boards Manager URL and paste it into "Additional Boards Manager URLs": https://m5stack.oss-cn-shenzhen.aliyuncs.com/resource/arduino/package_m5stack_index.json
3. Go to: **Tools** -> **Board:** -> **Boards Manager...**
4. Search for **M5Stack**, find it, and click "Install."
5. Choose **Tools** -> **Board:** -> **M5Stack Arduino**
6. Select **M5NanoC6**

3.2. BLUETOOTH SERIAL

- Open the Arduino IDE and load the example program: **File** -> **Examples** -> **BLE** -> **Write**.
- Connect the device to the computer.
- Choose the appropriate port for uploading. Once completed, the device will automatically enable Bluetooth.
- The device name is set to My**ESP32**. Now, use a Bluetooth serial communication tool on your PC to achieve transparent transmission of Bluetooth serial data.



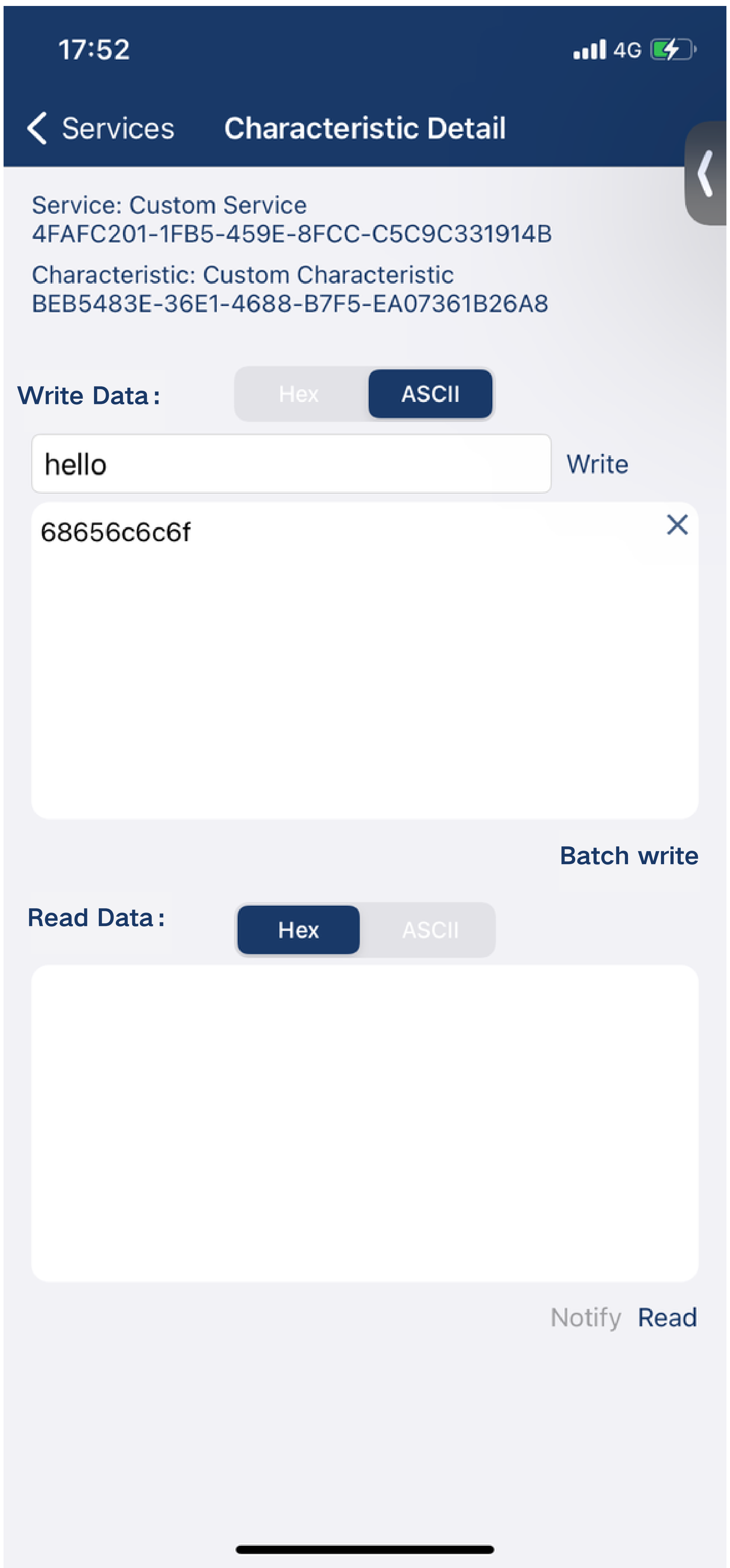
```
        Serial.println("*********");
      }
    }
};

void setup() {
  Serial.begin(115200);
while(!Serial){
    delay(10);
  }
  Serial.println("1- Download and install an BLE scanner app in your phone");
  Serial.println("2- Scan for BLE devices in the app");
  Serial.println("3- Connect to MyESP32");
  Serial.println("4- Go to CUSTOM CHARACTERISTIC in CUSTOM SERVICE and write something");
  Serial.println("5- See the magic =)");

  BLEDevice::init("MyESP32");
  BLEServer *pServer = BLEDevice::createServer();

  BLEService *pService = pServer->createService(SERVICE_UUID);

  BLECharacteristic *pCharacteristic = pService->createCharacteristic(
                                         CHARACTERISTIC_UUID,
                                         BLECharacteristic::PROPERTY_READ |
                                         BLECharacteristic::PROPERTY_WRITE
                                       );

  pCharacteristic->setCallbacks(new MyCallbacks());

  pCharacteristic->setValue("Hello World");
  pService->start();

  BLEAdvertising *pAdvertising = pServer->getAdvertising();
  pAdvertising->start();
}

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



```
17:43:53.985 -> 1- Download and install an BLE scanner app in your phone
17:43:53.985 -> 2- Scan for BLE devices in the app
17:43:53.985 -> 3- Connect to MyESP32
17:43:53.985 -> 4- Go to CUSTOM CHARACTERISTIC in CUSTOM SERVICE and write something
17:43:53.985 -> 5- See the magic =)
17:43:53.985 -> controller lib commit: [77d09ce]
17:44:56.486 -> *********
17:44:56.486 -> New value: a
17:44:56.486 -> *********
17:45:06.709 -> *********
17:45:06.709 -> New value: hello
17:45:06.709 -> *********
```

☑ Autoscroll ☑ Show timestamp Newline 115200 baud

```
rintln("3- Connect to MyESP32");
```



Service: Custom Service
4FAFC201-1FB5-459E-8FCC-C5C9C331914B

Characteristic: Custom Characteristic
BEB5483E-36E1-4688-B7F5-EA07361B26A8

Write Data : Hex [✓ ASCII]

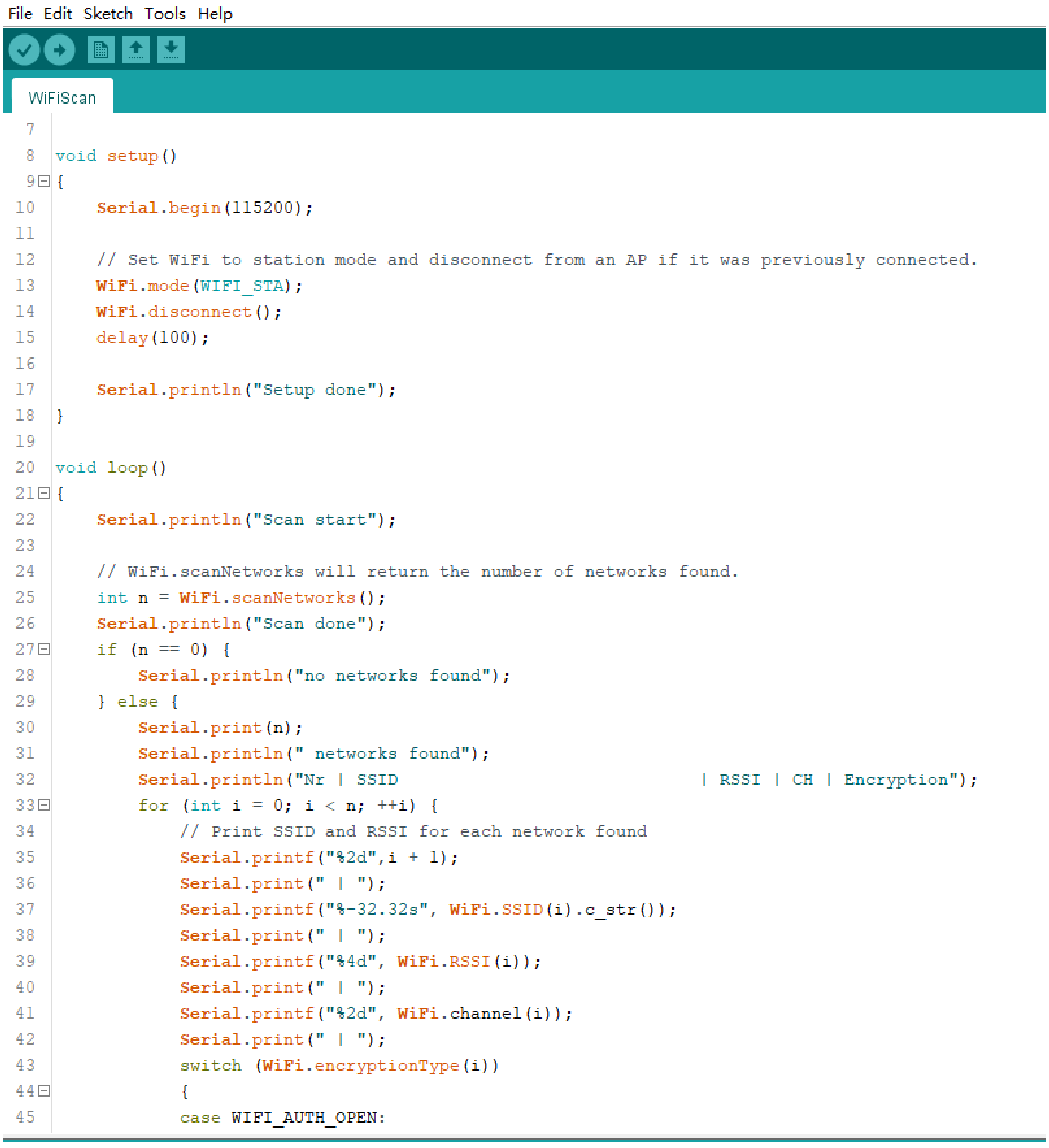
hello

68656c6c6f

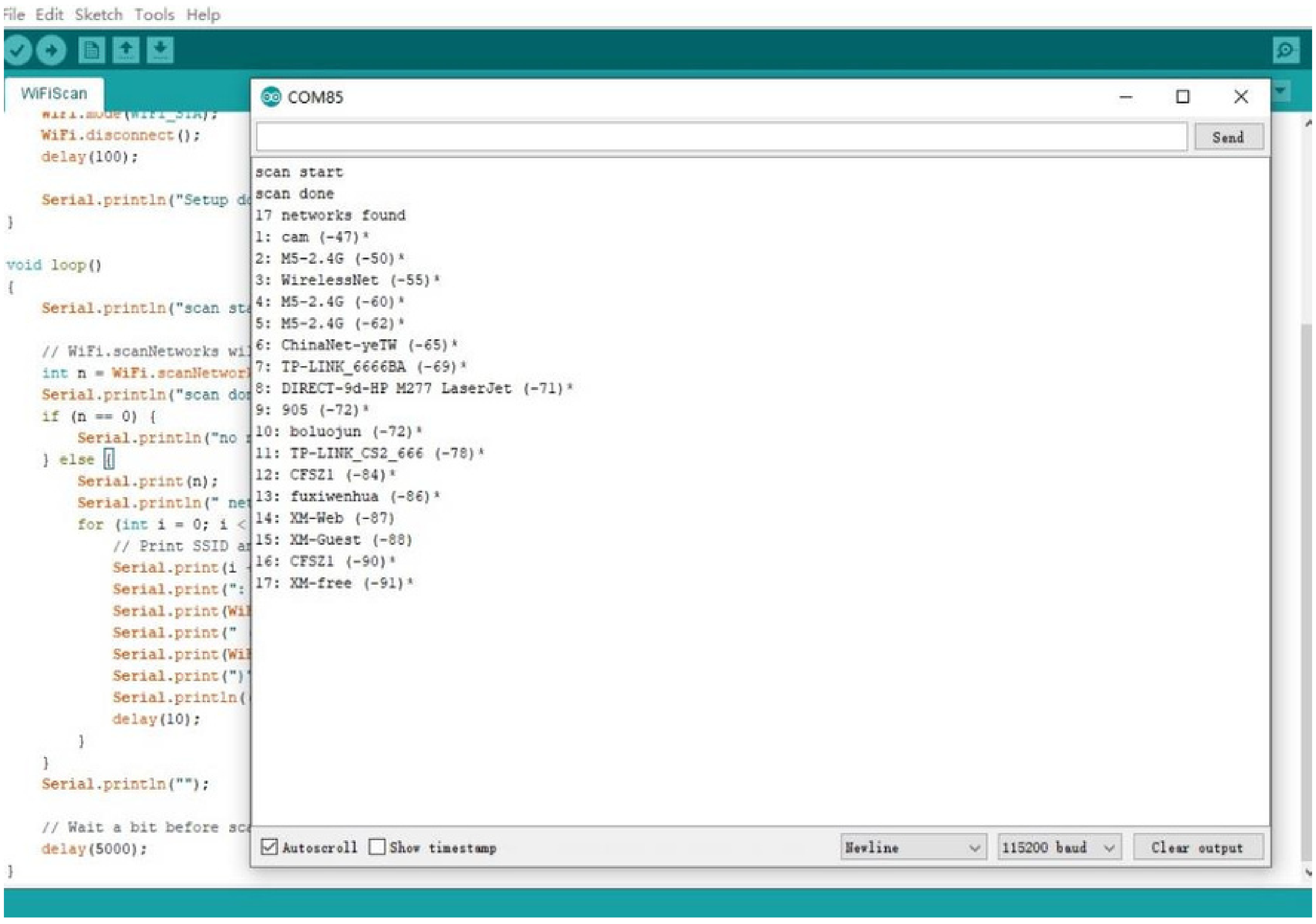
Read Data : [✓ Hex] ASCII

3.3. WIFI SCANNING

1. Open the Arduino IDE and load the example program: File -> Examples -> WiFi -> WiFiScan.
2. Connect the device to the computer and select the appropriate port for uploading.
3. Once completed, the device will automatically execute a WiFi scan. You can obtain the current WiFi scan results through the built-in serial monitor in Arduino.



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

    // Set WiFi to station mode and disconnect from an AP if it was previously connected.
    WiFi.mode(WIFI_STA);
    WiFi.disconnect();
    delay(100);

    Serial.println("Setup done");
}

void loop()
{
    Serial.println("Scan start");

    // WiFi.scanNetworks will return the number of networks found.
    int n = WiFi.scanNetworks();
    Serial.println("Scan done");
    if (n == 0) {
        Serial.println("no networks found");
    } else {
        Serial.print(n);
        Serial.println(" networks found");
        Serial.println("Nr | SSID                             | RSSI | CH | Encryption");
        for (int i = 0; i < n; ++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("%4d", WiFi.RSSI(i));
            Serial.print(" | ");
            Serial.printf("%2d", WiFi.channel(i));
            Serial.print(" | ");
            switch (WiFi.encryptionType(i))
            {
            case WIFI_AUTH_OPEN:
```

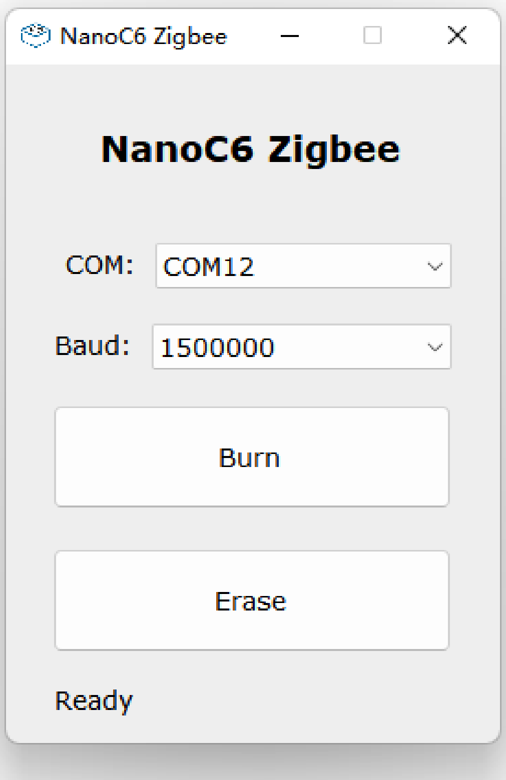


```
scan start
scan done
17 networks found
1: cam (-47)*
2: M5-2.4G (-50)*
3: WirelessNet (-55)*
4: M5-2.4G (-60)*
5: M5-2.4G (-62)*
6: ChinaNet-yeTW (-65)*
7: TP-LINK_6666BA (-69)*
8: DIRECT-9d-HP M277 LaserJet (-71)*
9: 905 (-72)*
10: boluojun (-72)*
11: TP-LINK_CS2_666 (-78)*
12: CFSZ1 (-84)*
13: fuxiwenhua (-86)*
14: XM-Web (-87)
15: XM-Guest (-88)
16: CFSZ1 (-90)*
17: XM-free (-91)*
```

☑ Autoscroll □ Show timestamp Newline 115200 baud Clear output

3.4. Zigbee (The Arduino program will be supported later.)

1.We can verify Zigbee functionality by running the firmware version of the program first.

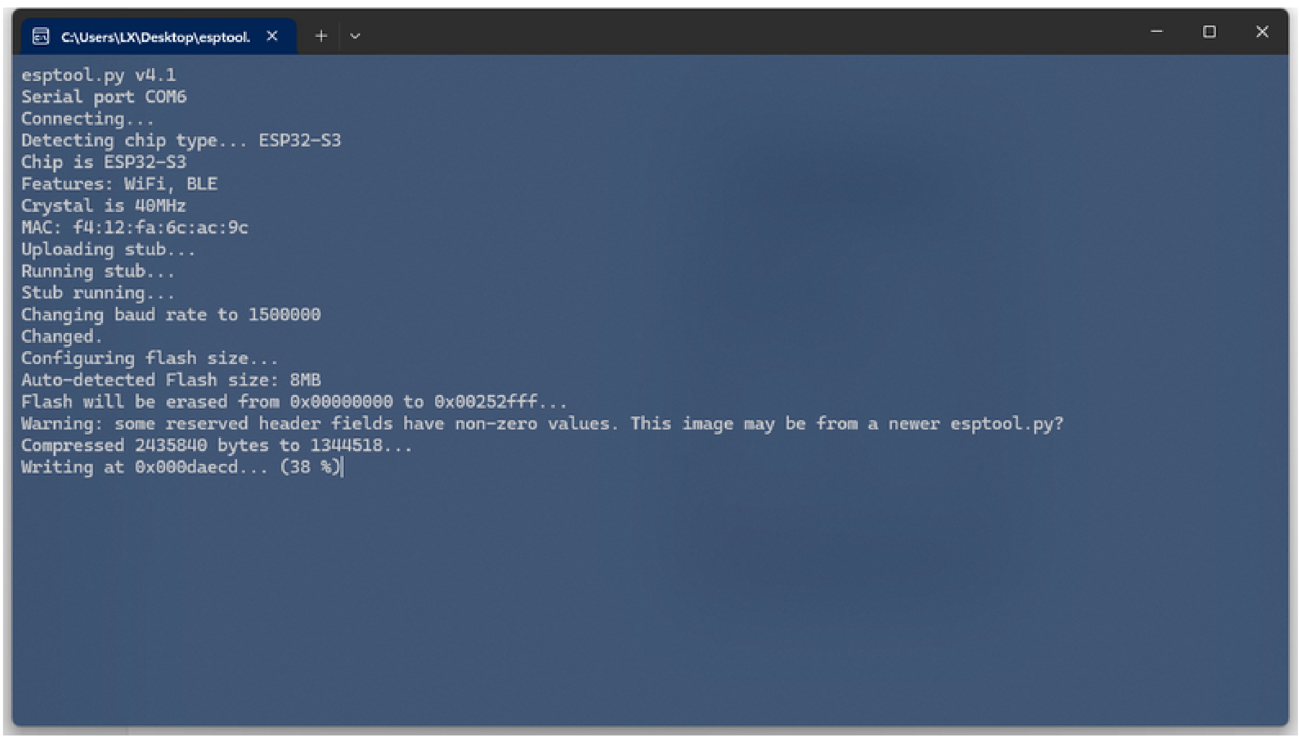


COM: COM12

Baud: 1500000

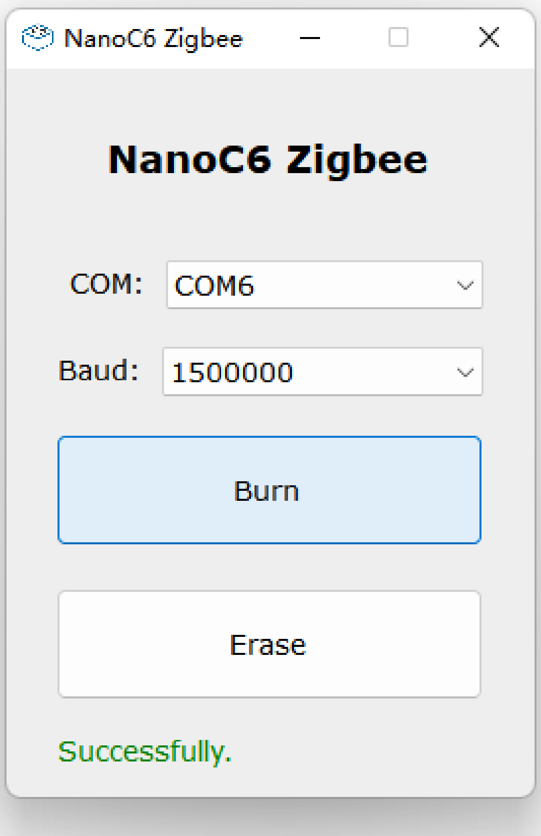
Ready

2.Click Burn to write the program to the device.



```
esptool.py v4.1
Serial port COM6
Connecting...
Detecting chip type... ESP32-S3
Chip is ESP32-S3
Features: WiFi, BLE
Crystal is 40MHz
MAC: f4:12:fa:6c:ac:9c
Uploading stub...
Running stub...
Stub running...
Changing baud rate to 1500000
Changed.
Configuring flash size...
Auto-detected Flash size: 8MB
Flash will be erased from 0x00000000 to 0x00252fff...
Warning: some reserved header fields have non-zero values. This image may be from a newer esptool.py?
Compressed 2435840 bytes to 1344518...
Writing at 0x000daecd... (38 %)
```

3.Upload completed, you can see the effect.



COM: COM6

Baud: 1500000

Successfully.