Important Information

Thank you for choosing Freenove products!

Getting Started

If you have not yet downloaded the zip file, associated with this kit, please do so now and unzip it.
https://github.com/Freenove/Freenove_ESP32_S3_Display

Get Support and Offer Input

Freenove provides free and responsive product and technical support, including but not limited to:

- Product quality issues
- Product use and build issues
- Questions regarding the technology employed in our products for learning and education
- Your input and opinions are always welcome
- We also encourage your ideas and suggestions for new products and product improvements

For any of the above, you may send us an email to:

support@freenove.com

Safety and Precautions

Please follow the following safety precautions when using or storing this product:

- Keep this product out of the reach of children under 6 years old.
- This product should be **used only when there is adult supervision present** as young children lack necessary judgment regarding safety and the consequences of product misuse.
- This product contains small parts and parts, which are sharp. This product contains electrically conductive parts. **Use caution with electrically conductive parts near or around power supplies, batteries and powered (live) circuits.**
- When the product is turned ON, activated or tested, some parts will move or rotate. **To avoid injuries to hands and fingers keep them away from any moving parts!**
- It is possible that an improperly connected or shorted circuit may cause overheating. **Should this happen, immediately disconnect the power supply or remove the batteries and do not touch anything until it cools down!** When everything is safe and cool, review the product tutorial to identify the cause.
- Only operate the product in accordance with the instructions and guidelines of this tutorial, otherwise parts may be damaged or you could be injured.
- Store the product in a cool dry place and avoid exposing the product to direct sunlight.
- After use, always turn the power OFF and remove or unplug the batteries before storing.

About Freenove

Freenove provides open source electronic products and services worldwide.

Freenove is committed to assist customers in their education of robotics, programming and electronic circuits so that they may transform their creative ideas into prototypes and new and innovative products. To this end, our services include but are not limited to:

- Educational and Entertaining Project Kits for Robots, Smart Cars and Drones
- Educational Kits to Learn Robotic Software Systems for Arduino, Raspberry Pi and micro: bit
- Electronic Component Assortments, Electronic Modules and Specialized Tools
- **Product Development and Customization Services**

You can find more about Freenove and get our latest news and updates through our website:

http://www.freenove.com

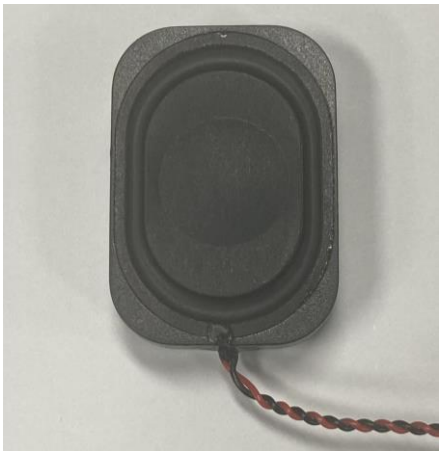
sale@freenove.com

Copyright

Contents

List

If you have any concerns, please feel free to contact us via support@freenove.com

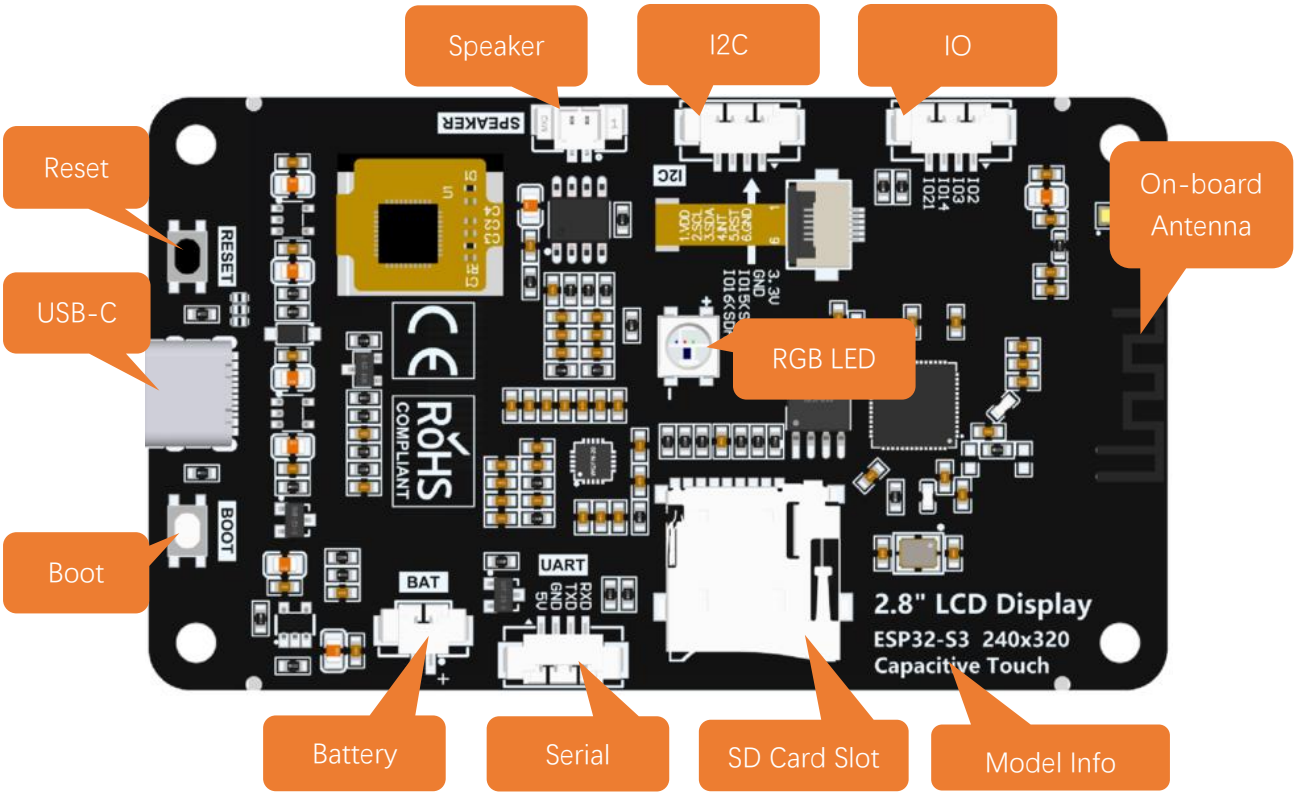
Freenove ESP32 Display x 1 	USB-C Data Cable x 1 
Battery adapter cable x1 	Speaker x1 
Jumper Wire x1 	

Preface

Freenove ESP32S3 Display

Top	Bottom
	

Hardware Interfaces



Battery (Optional)

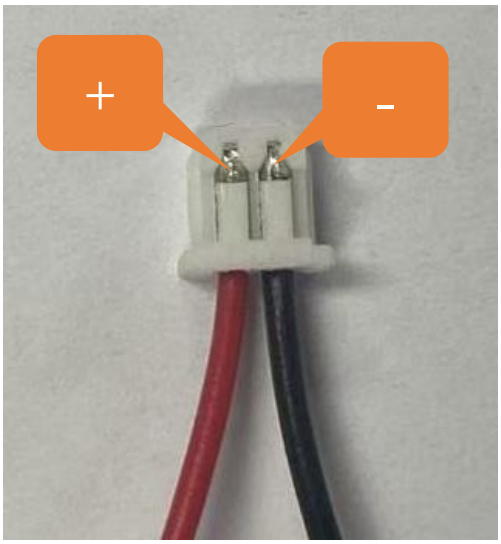
Please note that this product does not come with lithium batteries; please purchase them yourself.

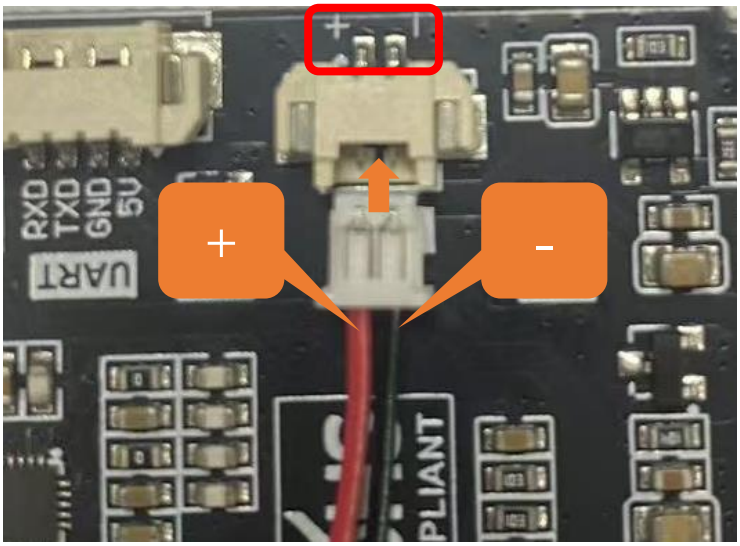
This device supports both **USB-powered and lithium battery-powered operation**. For optimal safety, USB power is recommended. Due to the **hazardous nature of lithium batteries**, we advise against their use unless absolutely necessary.

This device features an **MX1.25mm** connector and supports lithium batteries of various capacities. Note: The input voltage must be maintained within **3.7-4.2V** range.

Market-available batteries may feature **two distinct wiring configurations where the positive (+) and negative (–) terminals are reversed between models**. Please verify the battery's wiring matches the product requirements (refer to the diagram below) to prevent equipment failure or safety risks due to improper connection.

The red cable is the positive terminal while the black one is negative.

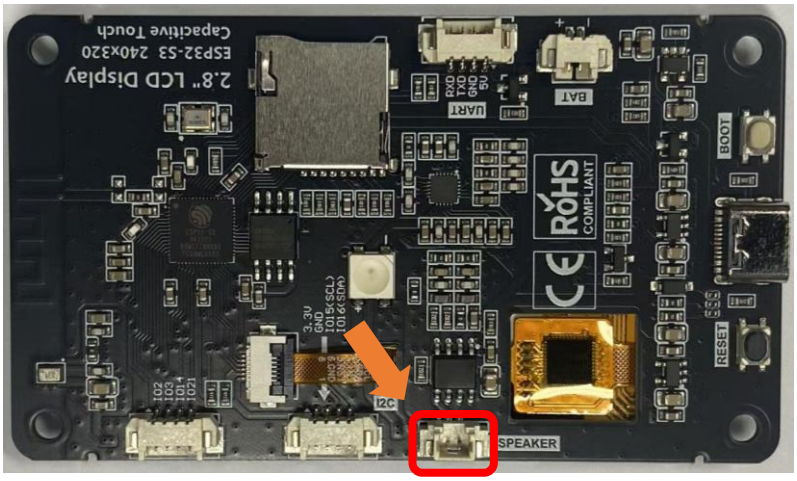




We recommend using a charger specially designed for lithium batteries. Due to various specifications and quality of lithium batteries, using a proper charger helps ensure peak performance, safety, and battery longevity.
While our product also supports USB charging as a backup option, please note that this method does not support fast charging and is limited to standard slow charging.

Speaker

There is a speaker connector (PH1.25mm) on the Freenove ESP32 S3 Display.



SD Card

The connector circuit uses SPI communication and supports high-speed Micro SD card storage.

Item	Pins	Definition
SD Card	GPIO23	SD_CMD
	GPIO18	SD_CLK
	GPIO19	SD_D0
	GPIO5	SD_CS

Note: This product does not include SD cards or SD card readers. Please buy them yourself

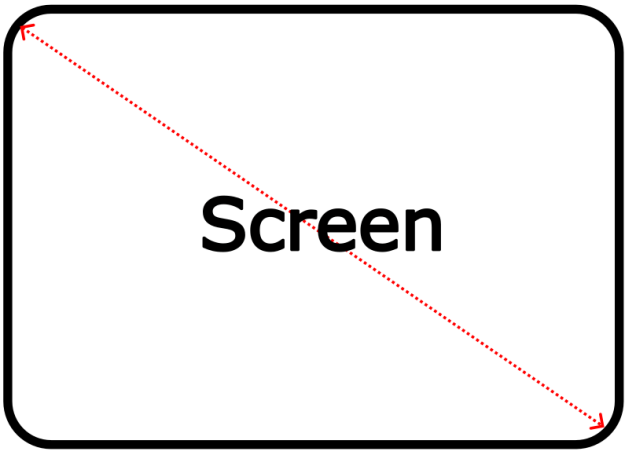
TFT Screen

Item	Pins	Definition
TFT Screen	GPIO13	LCD_MOSI
	GPIO12	LCD_MISO
	GPIO14	LCD_SCK
	GPIO2	LCD_RS
	GPIO15	LCD_CS

Screen Size

An internationally recognized unit of length, the **inch** (equivalent to 2.54 centimeters) has been the standard measurement for screen sizes in the display industry with a long history. This convention originated in the early television era when British manufacturers first adopted inches to measure cathode-ray tube (CRT) dimensions. The practice quickly became the global norm and remains the industry standard today.

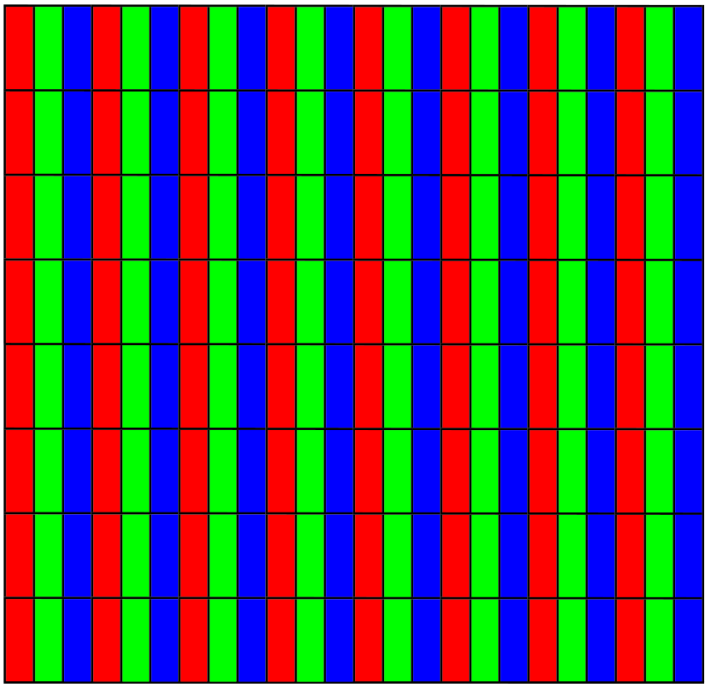
A critical clarification: screen size always refers to **the diagonal measurement of the display** panel. For instance, a 3.5-inch display features a 3.5-inch (8.89 cm) diagonal, ensuring standardized and comparable sizing across all devices.



Resolution

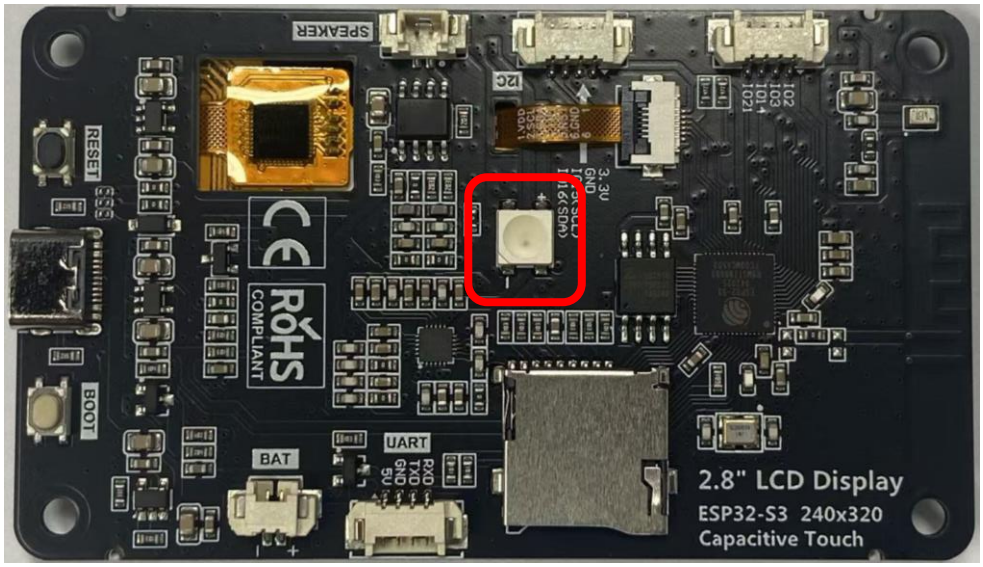
Screen resolution quantifies a display's pixel density along its horizontal and vertical axes, expressed as **width × height** (e.g., 320 × 480). This specification applies universally across displays, such as smartphones, monitors, and television screens.

- **Pixel**: The smallest unit of a display, composed of red (R), green (G), and blue (B) subpixels. Through precise brightness variation, these subpixels generate the complete color spectrum.
- **Resolution vs. Clarity**: Higher resolution means more pixels per unit area, resulting in sharper and more detailed imagery.

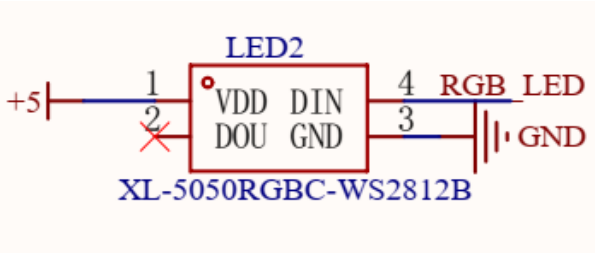


RGB LED

The Freenove ESP32 S3 Display includes an RGB LED (red, green, blue) that can blend colors to create various lighting effects.



Item	Pins
R	GPIO22
G	GPIO16
B	GPIO17



GPIO Pinout Table

To learn what each GPIO corresponds to, please refer to the following table.
The functions of the pins are allocated as follows:

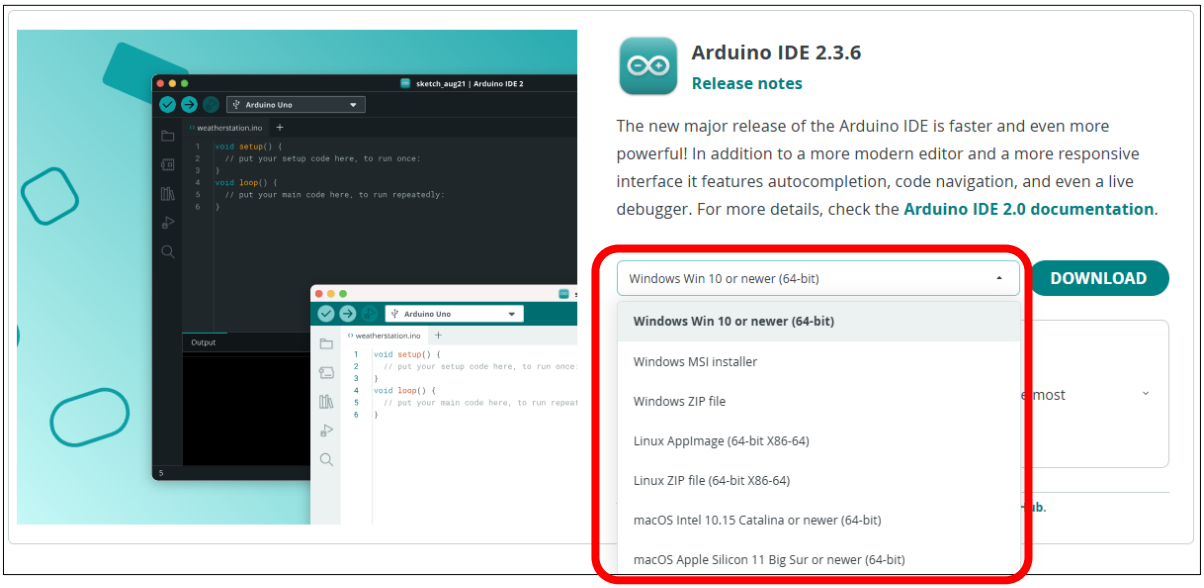
ESP32-S3 N16R8	Functions	Description
GPIO22	R	RGB
GPIO16	G	
GPIO17	B	
GPIO13	LCD_MOSI	TFT_LCD
GPIO12	LCD_MISO	
GPIO14	LCD_SCK	
GPIO2	LCD_RS	
GPIO15	LCD_CS	
GPIO23	SD_CMD	SD Card
GPIO18	SD_CLK	
GPIO19	SD_D0	
GPIO5	SD_CS	

For more information, refer to the schematic.

If you have any concerns, please feel free to contact us via support@freenove.com

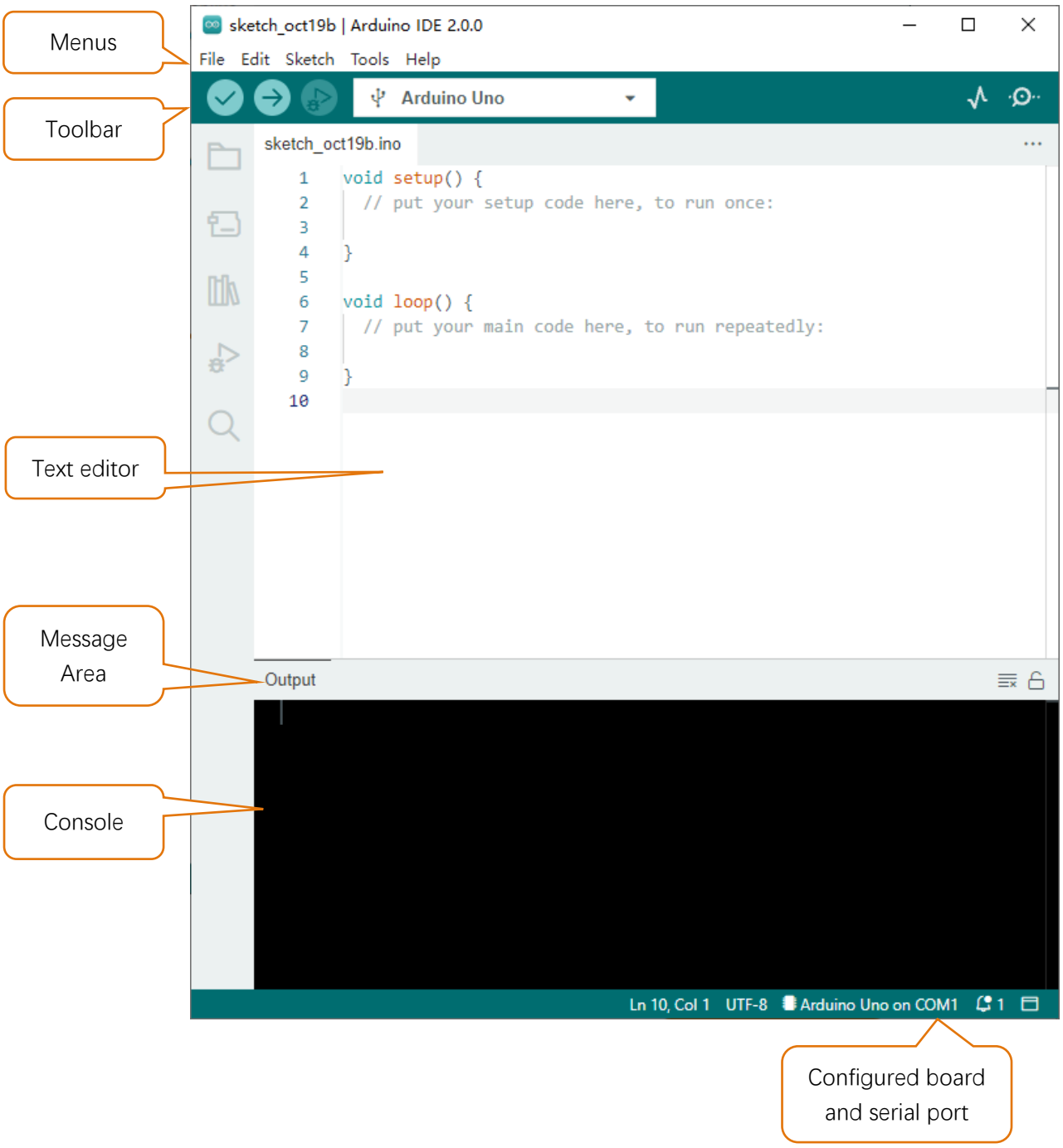
Programming Software

We use the Arduino Software (IDE) to write and upload the code for this product.
First, install Arduino Software (IDE): visit https://www.arduino.cc/en/software/, Select and download corresponding installer according to your operating system. If you are a Windows user, please select the "Windows" to download and install it correctly.

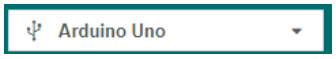


After the download completes, run the installer. For Windows users, there may pop up an installation dialog box of driver during the installation process. When it popes up, please allow the installation.
After installation completes, an Arduino Software shortcut will be generated in the desktop. Run the Arduino Software.

The interface of Arduino Software is as follows:


Menus
Toolbar
Text editor
Message Area
Console
sketch_oct19b | Arduino IDE 2.0.0
File Edit Sketch Tools Help
Arduino Uno
sketch_oct19b.ino
1 void setup() {
2 // put your setup code here, to run once:
3
4 }
5
6 void loop() {
7 // put your main code here, to run repeatedly:
8
9 }
10
Output
Ln 10, Col 1 UTF-8 Arduino Uno on COM1 1
Configured board and serial port

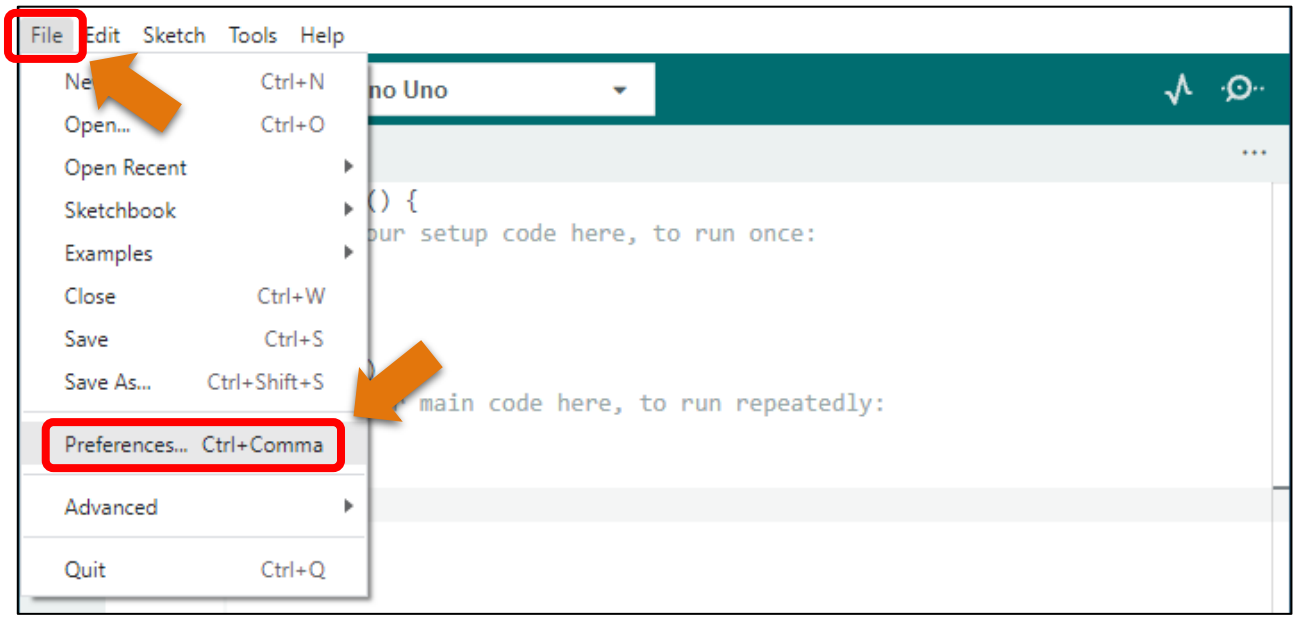
Programs written with Arduino Software (IDE) are called **sketches**. These sketches are written in the text editor and saved with the file extension**.ino**. The editor features text cutting/pasting and searching/replacing. The message area gives feedback while saving and exporting and also displays errors. The console displays text output by the Arduino Software (IDE), including complete error messages and other information. The bottom right-hand corner of the window displays the configured board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.

	Verify Check your code for compile errors.
	Upload Compile your code and upload them to the configured board.
	Debug Debug code running on the board. (Some development boards do not support this function)
	Development board selection Configure the support package and upload port of the development board.
	Serial Plotter Receive serial port data and plot it in a discounted graph.
	Serial Monitor Open the serial monitor.

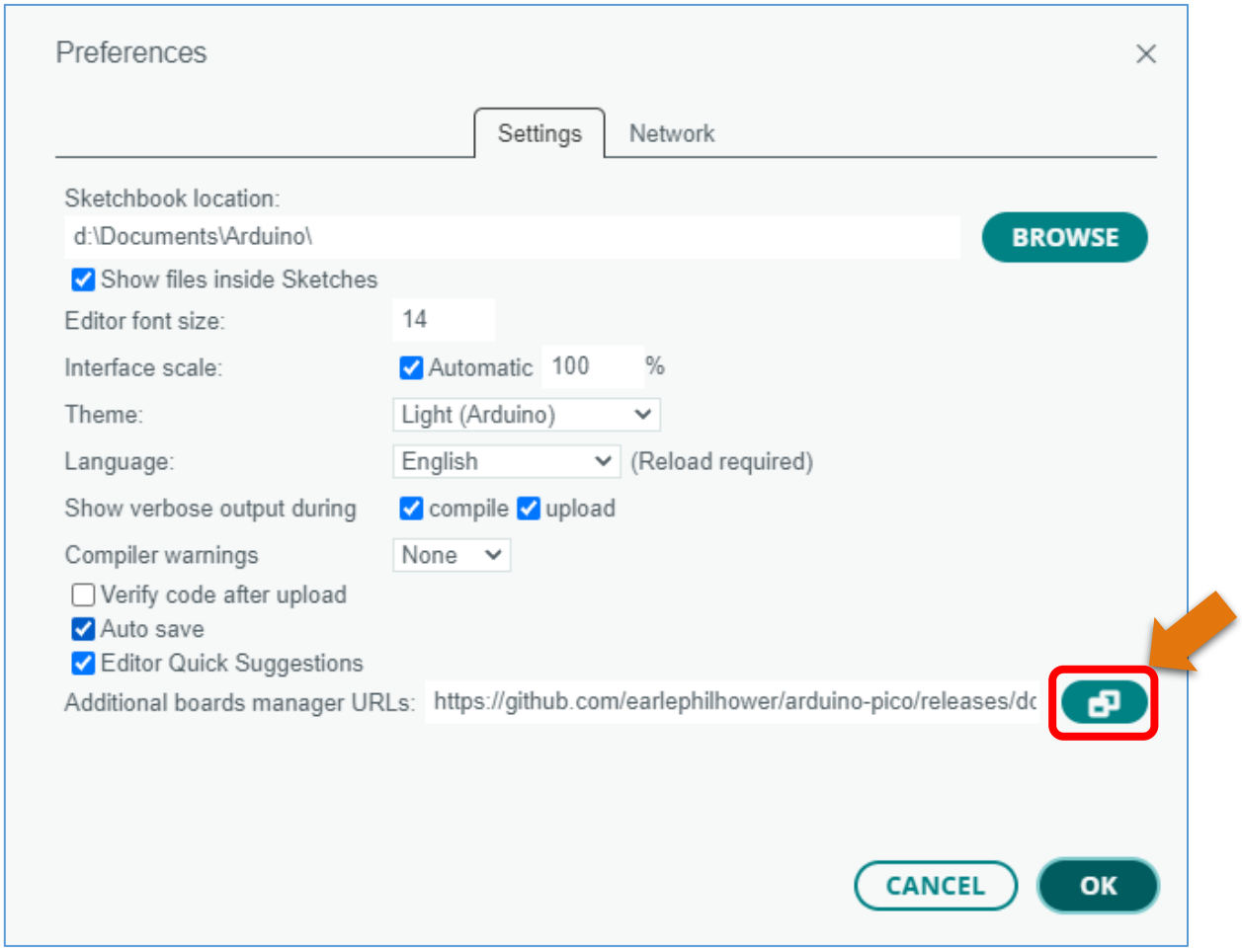
Additional commands are found within the five menus: File, Edit, Sketch, Tools, Help. The menus are context sensitive, which means only those items relevant to the work currently being carried out are available.

Environment Configuration

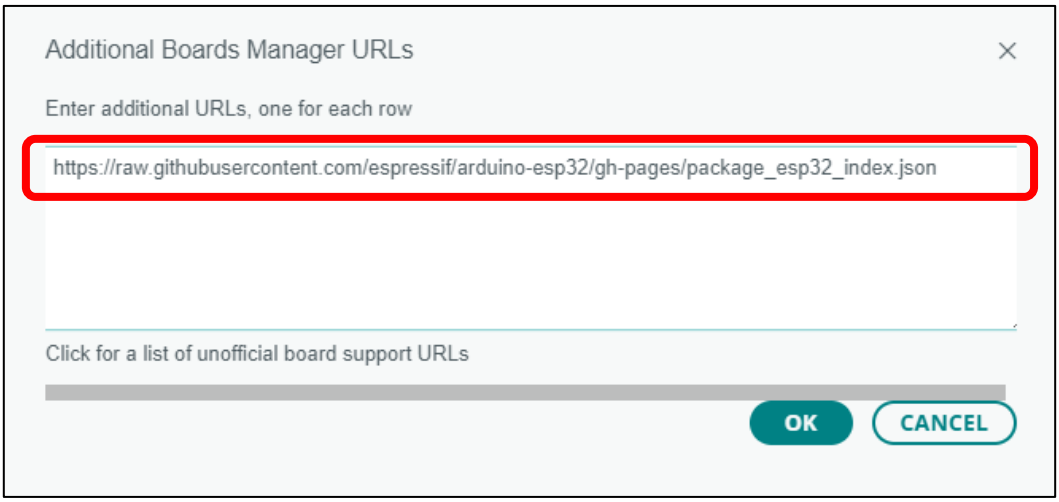
First, open the software platform Arduino, and then click File in Menus and select Preferences.



Second, click on the symbol behind "Additional Boards Manager URLs"

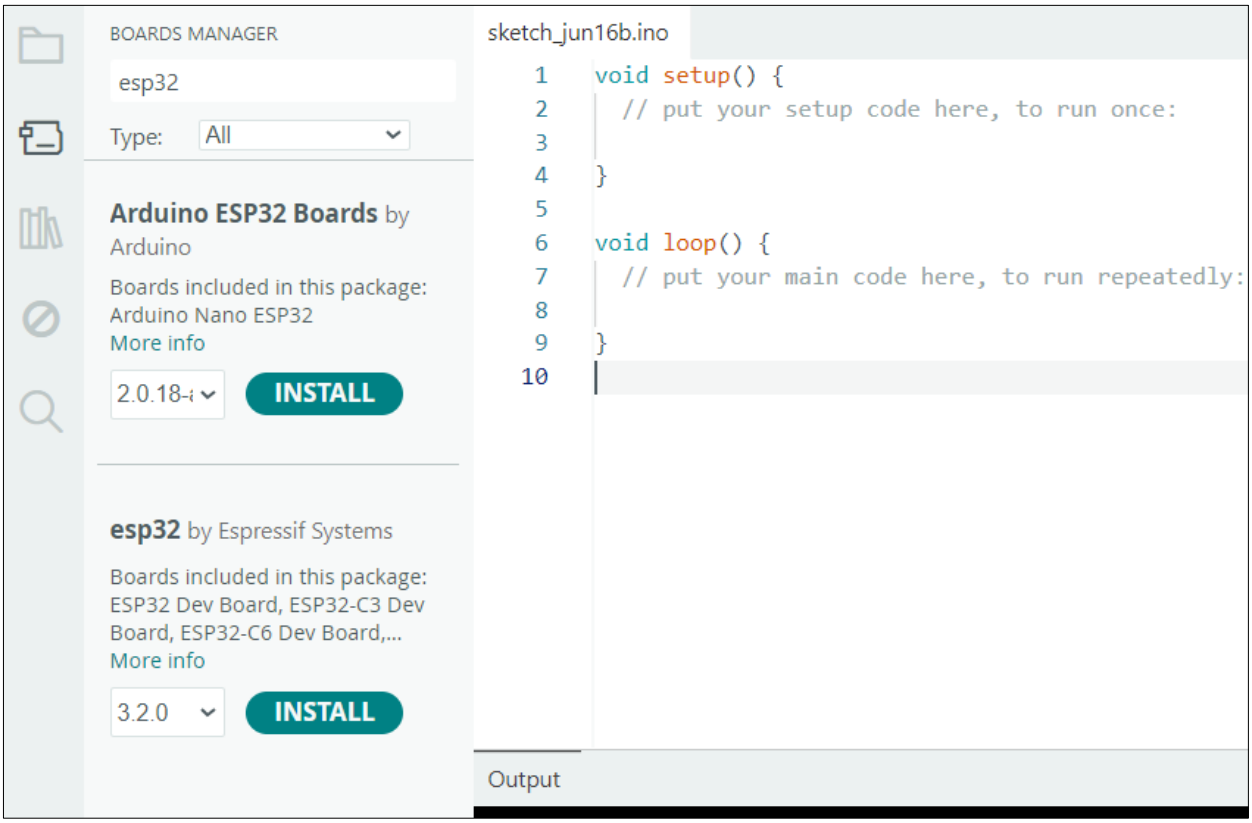


Third, fill in https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json in the new window, click OK, and click OK on the Preferences window again.

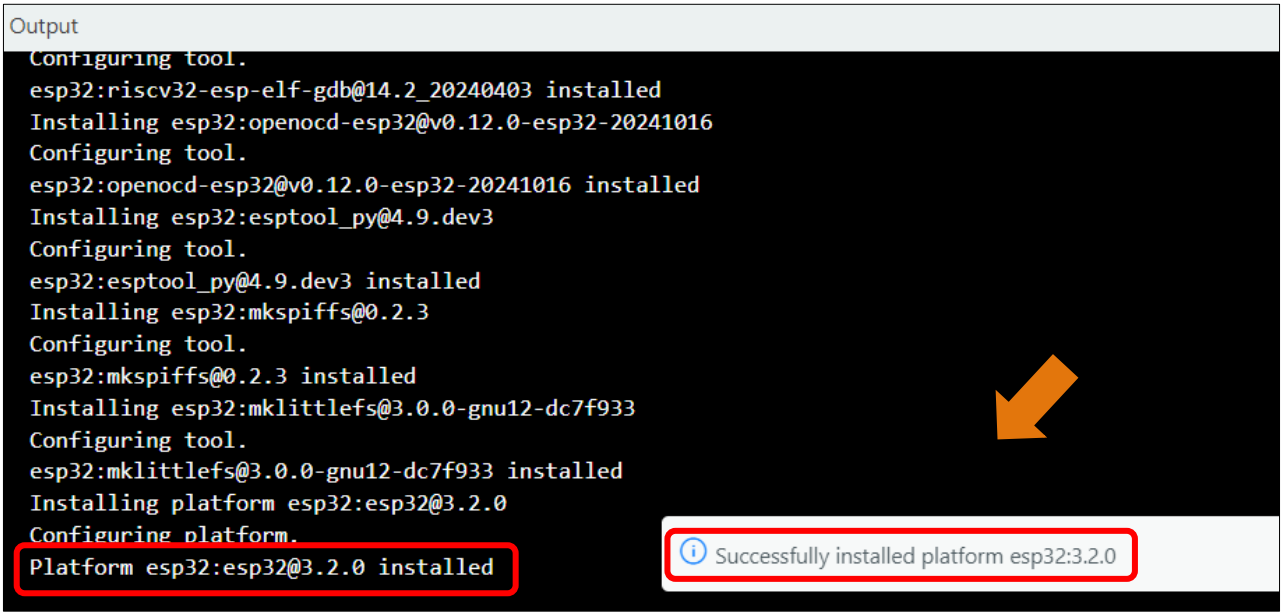


Note: if you copy and paste the URL directly, you may lose the "-". Please check carefully to make sure the link is correct.

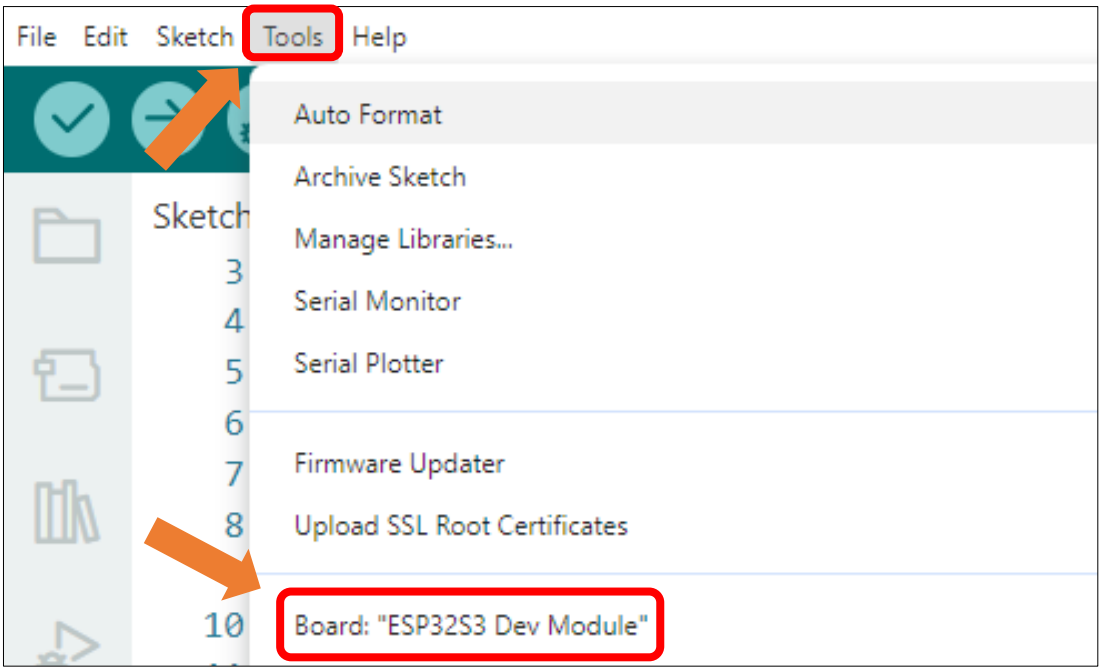
Fourth, click "Boards Manager". Enter “esp32” in Boards manager, select 3.2.0, and click “INSTALL”.



Arduino will download these files automatically. Wait for the installation to complete.



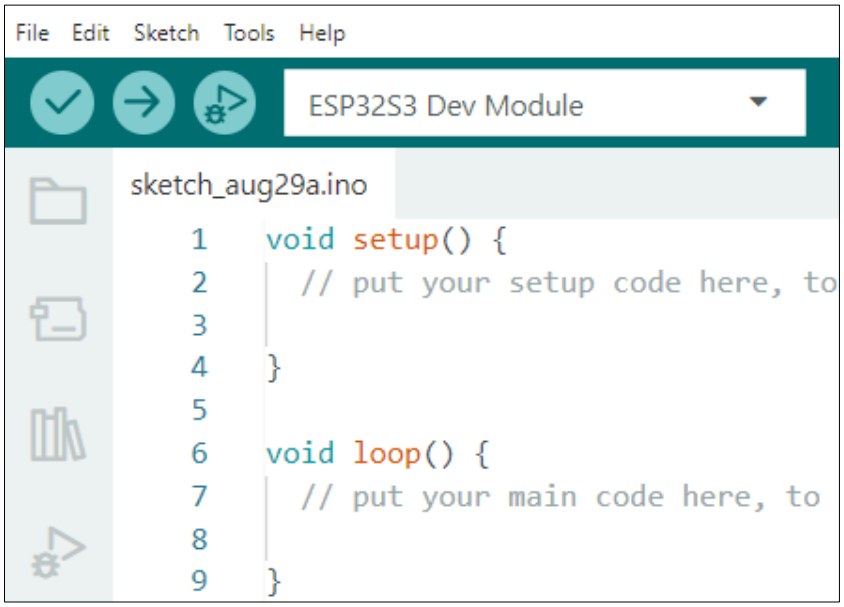
When finishing installation, click Tools in the Menus again and select Board: "ESP32S3 Dev Module", and then you can see information of ESP32S3.



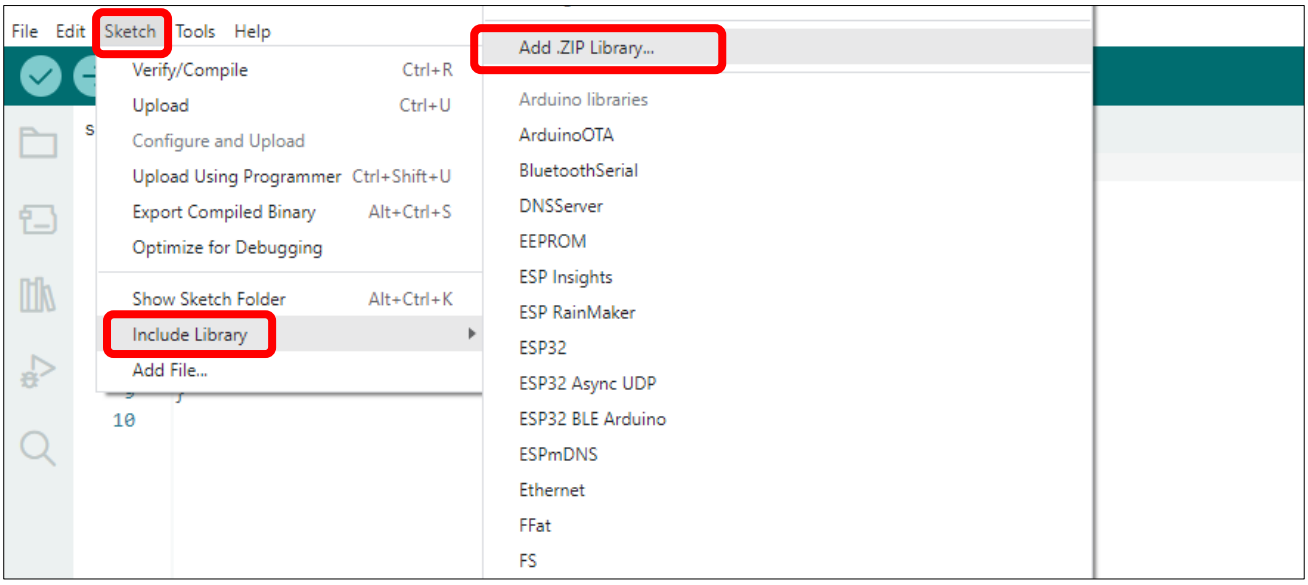
Library Installation

Before starting the learning process, it is necessary to install some libraries in advance to enable the code to be compiled properly. For convenience, we have already packaged these libraries and placed them in the **Freenove_ESP32_S3_Display/Libraries** folder. Please refer to the following steps to install these libraries into the Arduino IDE.

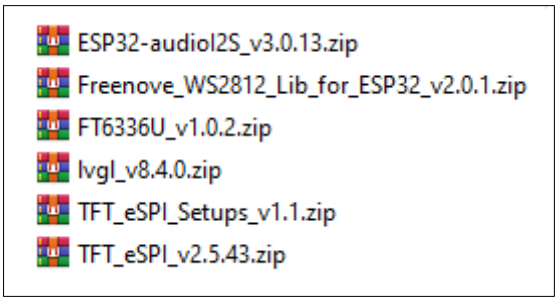
1. Open Arduino IDE.



2. Select Sketch->Include Library->Add .ZIP library….

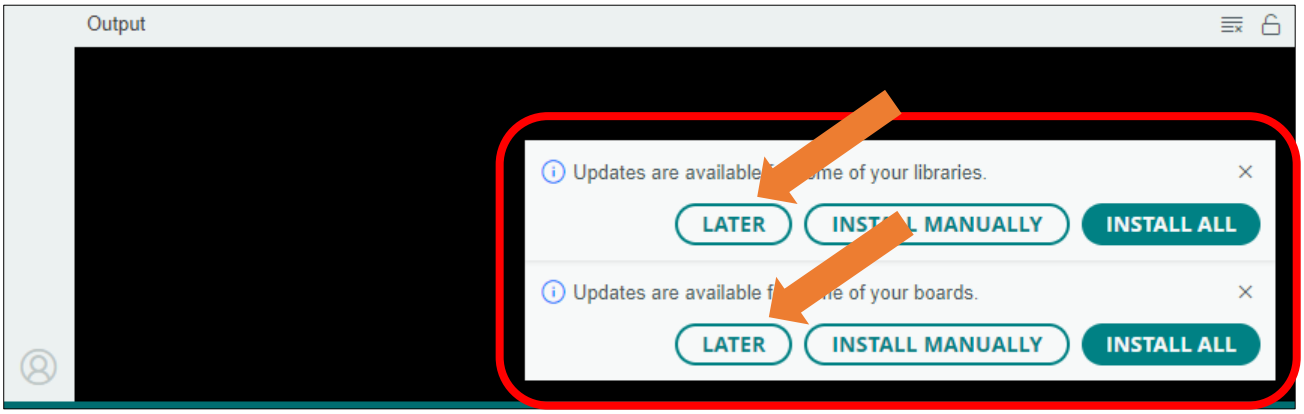


3. On the newly pop-up window, select the files from the **Freenove_ESP32_S3_Display /Libraries**. Click Open to install the library.



4. **Repeat the above steps until all the six libraries are installed to Arduino.** So far, all libraries have been installed.

Note: Some libraries are not the latest version. Please do not update them even if it prompts every time you open the IDE. Just click LATER. Otherwise, it may lead to compilation failure.



Chapter 1 Serial

Project 1.1 SerialRW

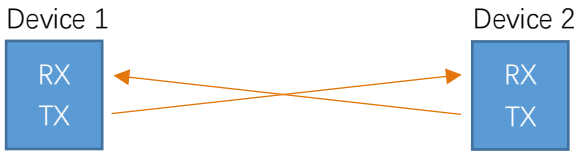
Related Knowledge

Serial and parallel communication

Serial communication uses one data cable to transfer data one bit by another in turn, while parallel communication means that the data is transmitted simultaneously on multiple cables. Serial communication takes only a few cables to exchange information between systems, which is especially suitable for computers to computer, long distance communication between computers and peripherals. Parallel communication is faster, but it requires more cables and higher cost, so it is not appropriate for long distance communication.

Serial communication

Serial communication generally refers to the Universal Asynchronous Receiver/Transmitter (UART), which is commonly used in electronic circuit communication. It has two communication lines, one is responsible for sending data (TX line) and the other for receiving data (RX line). The serial communication connections of two devices use is as follows:



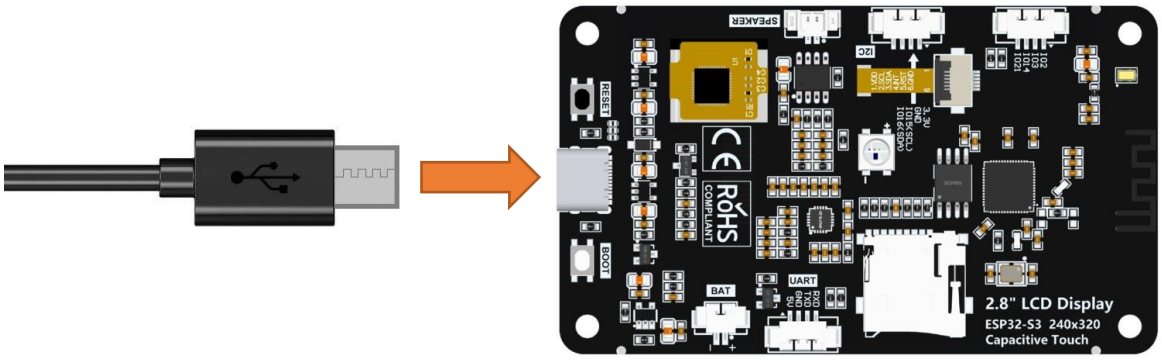
Before serial communication starts, the baud rate in both sides must be the same. Only use the same baud rate can the communication between devices be normal. The baud rates commonly used are 9600 and 115200.

Component List

Freenove ESP32 S3 Display x 1 	USB cable x1 

Circuit

Connect Freenove ESP32 S3 Display to the computer with USB cable.



Sketch

Open "**Sketch_01.1_SerialRW**" folder under **"Freenove_ESP32_S3_Display\NonTouch\Sketches"** and double-click "**Sketch_01.1_SerialRW.ino**".

Sketch_01.1_SerialRW

```
1  /*
2  * @ File:   Sketch_01.1_SerialRW.ino
3  * @ Author: [Zhentao Lin]
4  * @ Date:   [2025-08-19]
5  */
6
7  String inputString = "";      //a String to hold incoming data
8  bool stringComplete = false;  // whether the string is complete
9
10 void setup() {
11    Serial.begin(115200);
12    Serial.println(String("\nESP32-S3 initialization completed!\n")
```

```
13                  + String("Please input some characters,\n")
14                  + String("select \"Newline\" below and click send button. \n"));
15  }
16
17  void loop() {
18    if (Serial.available()) {         // judge whether data has been received
19      char inChar = Serial.read();          // read one character
20      inputString += inChar;
21      if (inChar == '\n') {
22        stringComplete = true;
23      }
24    }
25    if (stringComplete) {
26      Serial.printf("InputString: %s", inputString);
27      inputString = "";
28      stringComplete = false;
29    }
30  }
```

Code Explanation

Set the baud rate to 115200.

```
8   Serial.begin(115200);
```

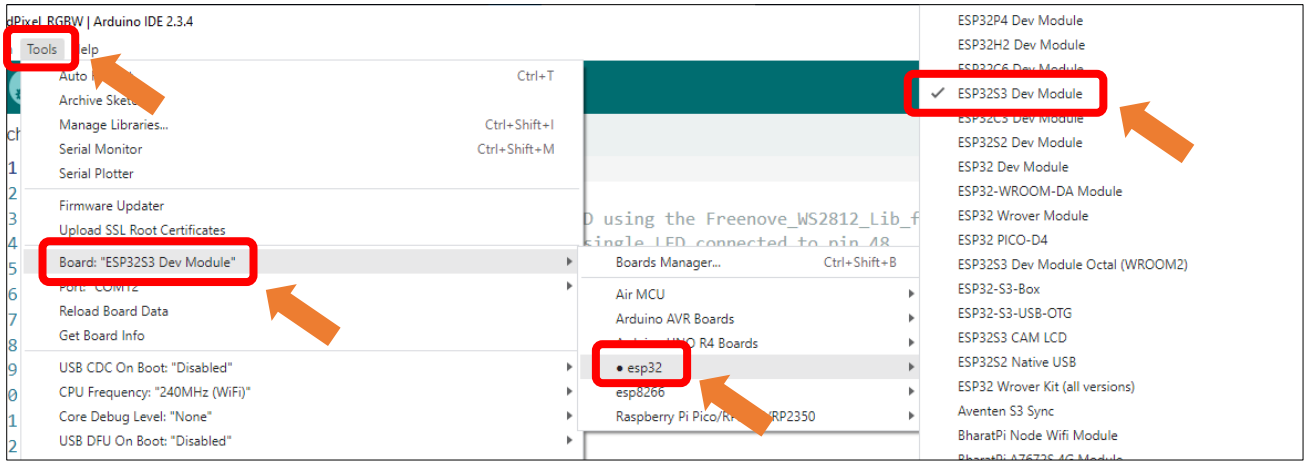
Determine whether there is data in the serial port buffer.

```
8   if (Serial.available())
```

Receive serial port data and save it in the inputString string.

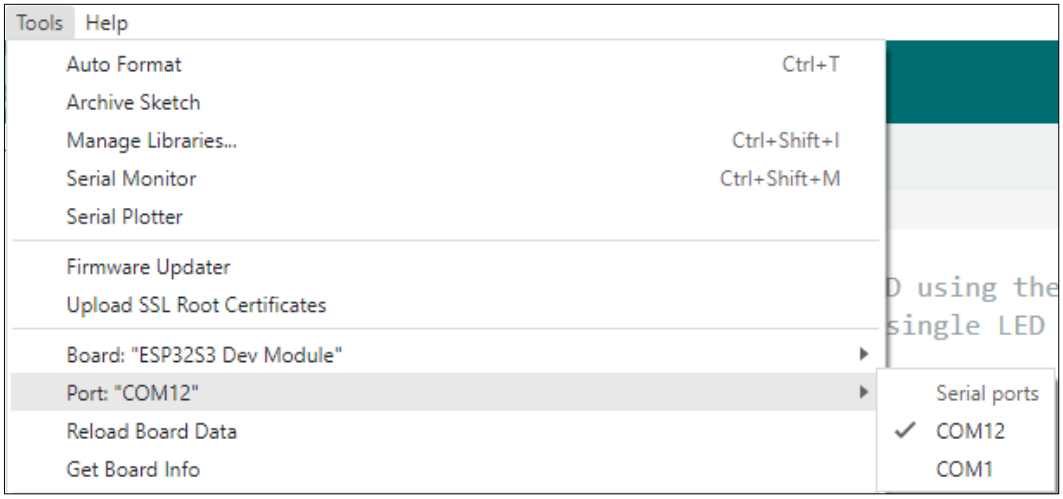
```
19  char inChar = Serial.read();          // read one character
20  inputString += inChar;
```

Select **Tools** –> **Board** -> **esp32** -> ESP32S3 Dev Module.

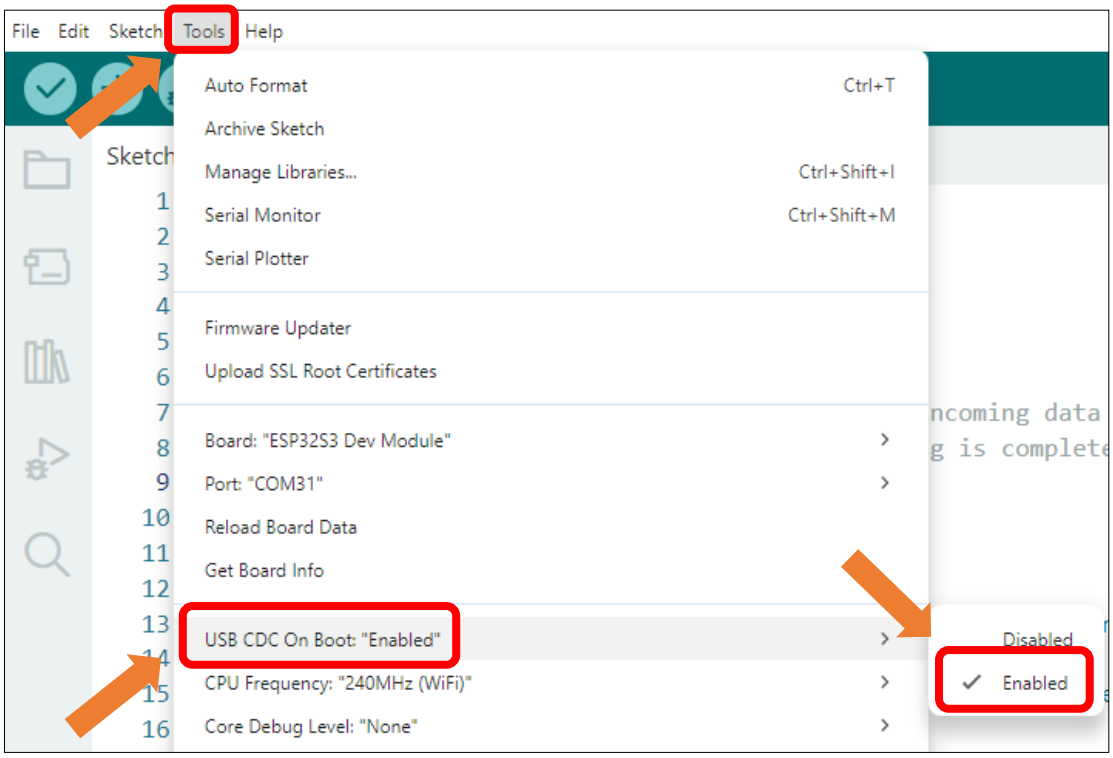


After connecting the Freenove Media Kit for ESP32-S3, the system will assign a serial communication port named in the format 'COMx' (**where 'x' is a numeric ID that may vary across computers**). You must select the correct port under Tools → Port.

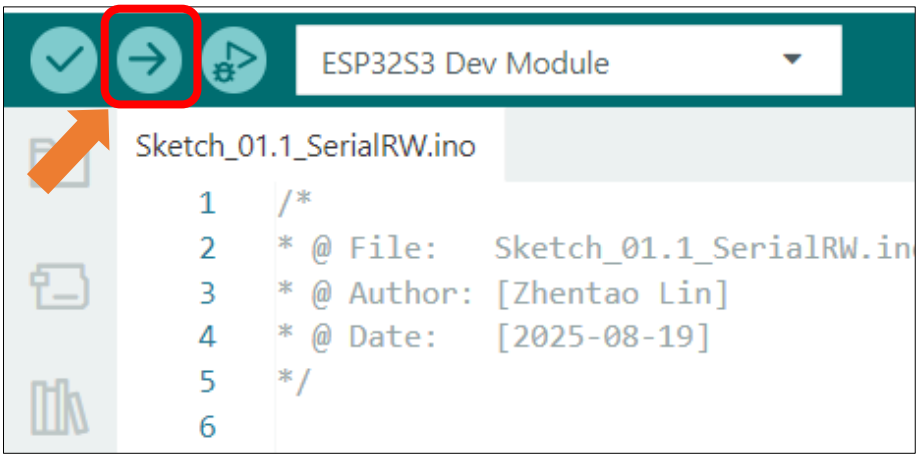
Note: COM1 is typically NOT the port used by the Freenove Media Kit for ESP32-S3.



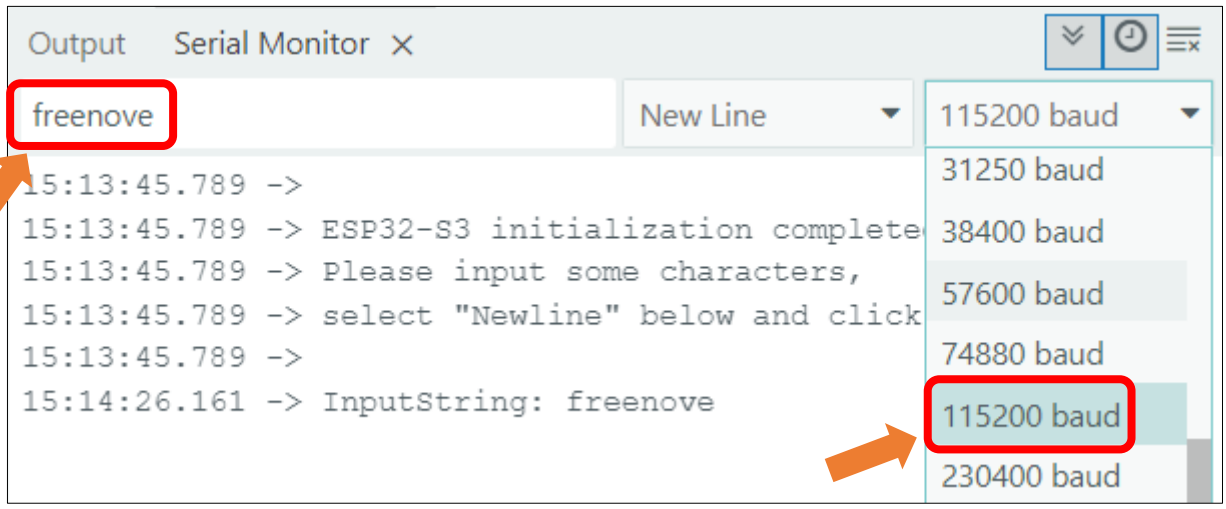
Enable the "USB CDC On Boot" feature.



The purpose of this code is to display data on the serial monitor. Click "**Upload**" to upload the code to Freenove ESP32 S3 Display.



After downloading the code, open the serial port monitor, and set the baud rate to **115200**, input any data in the messages bard and press Enter key, Freenove ESP32 S3 Display will print the received data.



Reference

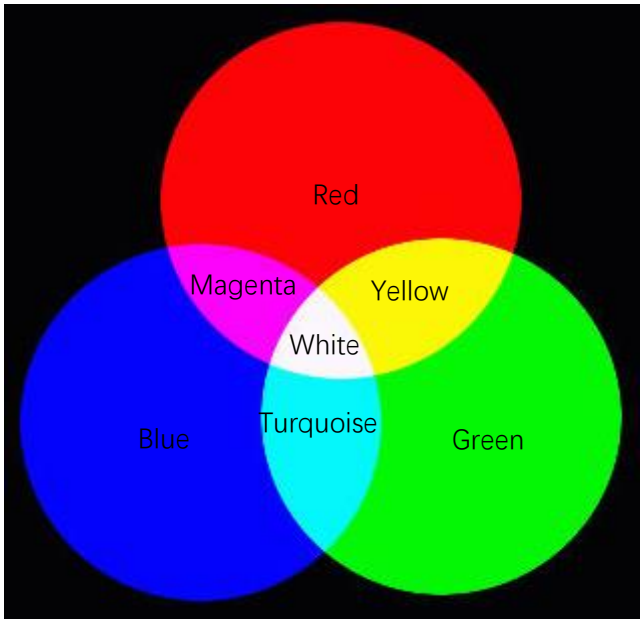
`int available()`
Serial.available() checks the number of bytes currently available to read in the Serial receive buffer. It returns the number of bytes available (int type), or 0 if the buffer is empty.
`int read ()`
Serial.read() reads one byte of data from the Serial receive buffer and returns it as an int. If no data is available to read, it returns -1.

Chapter 2 RGB

Project 2.1 RGB

Related Knowledge

Red, green, and blue are called the three primary colors. When you combine these three primary colors of different brightness, it can produce almost all kinds of visible light.



RGB

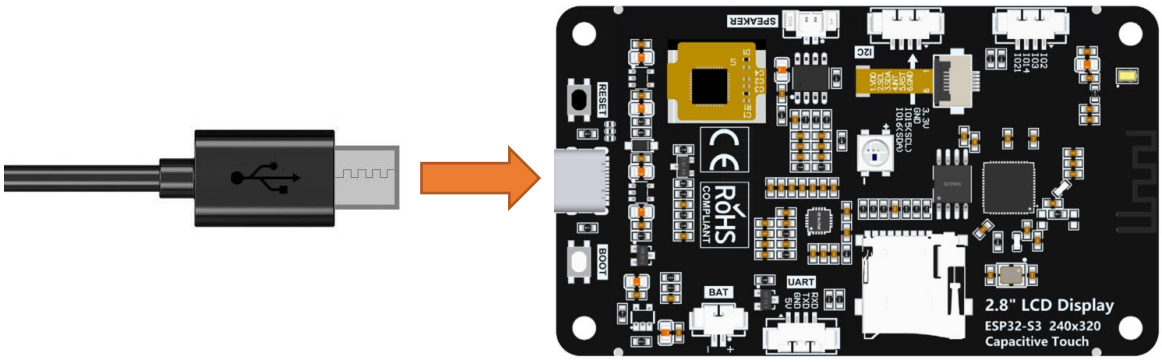
The onboard RGB LED can emit three basic colors of red, green and blue, and supports 256-level brightness adjustment, which means that it can emit 2^24=16,777,216 different colors.

Component List

Freenove ESP32 S3 Display x 1 	USB cable x1 

Circuit

Connect Freenove ESP32 S3 Display to the computer with USB cable.

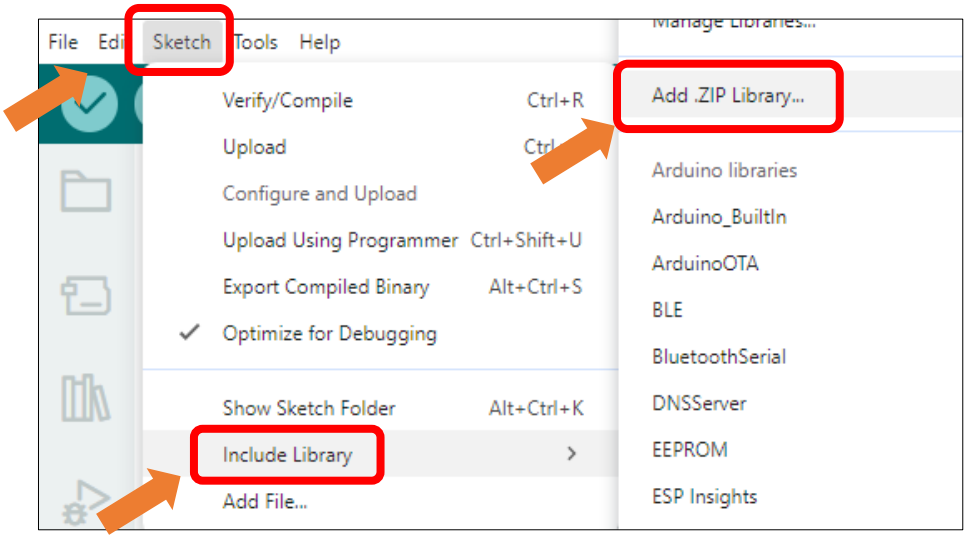


Sketch

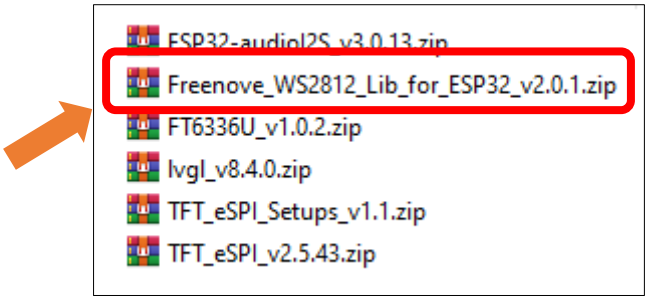
Open "**Sketch_02.1_LedPixel**" folder under **"Freenove_ESP32_S3_Display\NonTouch\Sketches"** and double-click "**Sketch_02.1_LedPixel.ino**".

Install the needed libraries.

Click **Sketch** -> **Include Library** -> **Add .ZIP Library...**



Select **Freenove_WS2812_Lib_for_ESP32_v2.0.1.zip**



Sketch_02.1_LedPixel

The following is the program code:

```
/*
* @ File:   Sketch_02.1_LedPixel.ino
* @ Author: [Zhentao Lin]
* @ Date:   [2025-08-19]
*/

#include "Freenove_WS2812_Lib_for_ESP32.h"

#define LEDS_COUNT 1
#define LEDS_PIN   42
#define CHANNEL    0

Freenove_ESP32_WS2812 strip = Freenove_ESP32_WS2812(LEDS_COUNT, LEDS_PIN, CHANNEL, TYPE_GRB);

uint8_t m_color[5][3] = { { 255, 0, 0 }, { 0, 255, 0 }, { 0, 0, 255 }, { 255, 255, 255 }, { 0,
0, 0 } };
int delayval = 100;

void setup() {
   strip.begin();
   strip.setBrightness(10);
}
void loop() {
   for (int j = 0; j < 5; j++) {
     for (int i = 0; i < LEDS_COUNT; i++) {
       strip.setLedColorData(i, m_color[j][0], m_color[j][1], m_color[j][2]);
       strip.show();
       delay(delayval);
     }
     delay(500);
   }
}
```

Code Explanation

Define the pins for the RGB LED.

```
#define LEDS_COUNT 1
#define LEDS_PIN   42
#define CHANNEL    0
```

Initialize the LED, set the brightness to 10

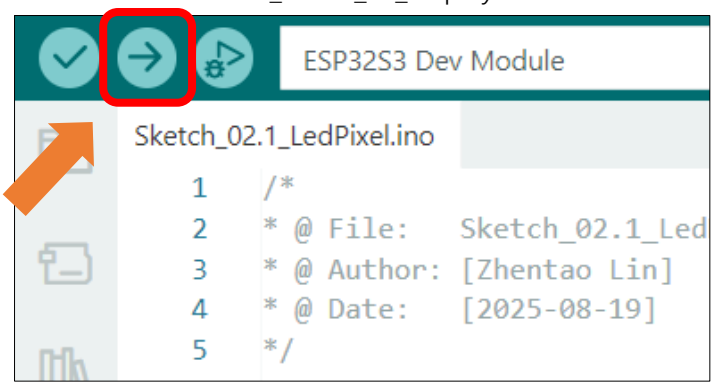
```
strip.begin();
strip.setBrightness(10);
```

Cycle through five different colors.

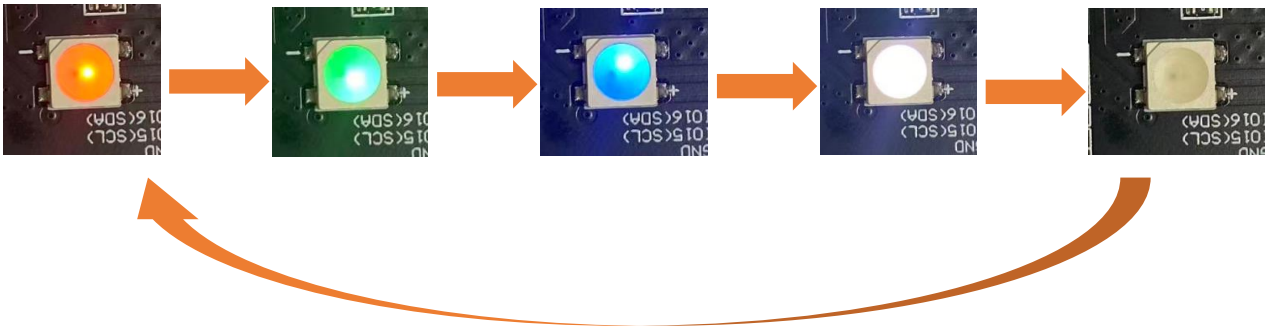
```
for (int j = 0; j < 5; j++) {
```

```
25      for (int i = 0; i < LEDS_COUNT; i++) {
26        strip.setLedColorData(i, m_color[j][0], m_color[j][1], m_color[j][2]);
27        strip.show();
28        delay(delayval);
29      }
30      delay(500);
31    }
```

Click "**Upload**" to upload the code to Freenove_ESP32_S3_Display.



The RGB LEDs will cycle through Red -> Green -> Blue –> White -> Off every 500ms after code upload.



Reference

Freenove_ESP32_WS2812(u16 **n** = 8, u8 **pin_gpio** = 2, u8 **chn** = 0, LED_TYPE **t** = TYPE_GRB)

Constructor to create a ws2812 object.

Before each use of the constructor, please add "**#include "Freenove_WS2812_Lib_for_ESP32.h**"

Parameters

n: The number of led.

pin_gpio: The pin connected to the LED.

Chn: RMT channel, which has eight channels, 0-7, and uses channel 0 by default. This means that you can use eight ws2812 modules for the display at the same time, and these modules do not interfere with each other.

t: Types of LED.

TYPE_RGB: The sequence of ws2812 module loading color is red, green and blue.

TYPE_RBG: The sequence of ws2812 module loading color is red, blue and green.

TYPE_GRB: The sequence of ws2812 module loading color is green, red and blue.

TYPE_GBR: The sequence of ws2812 module loading color is green, blue and red.

TYPE_BRG: The sequence of ws2812 module loading color is blue, red and green.

TYPE_BGR: The sequence of ws2812 module loading color is blue, green and red.

Project 2.2 Rainbow

In the previous project, we have mastered the use of LEDPixel. This project will realize a slightly complicated rainbow light. The component list and the circuit are exactly the same as the project fashionable light.

Sketch

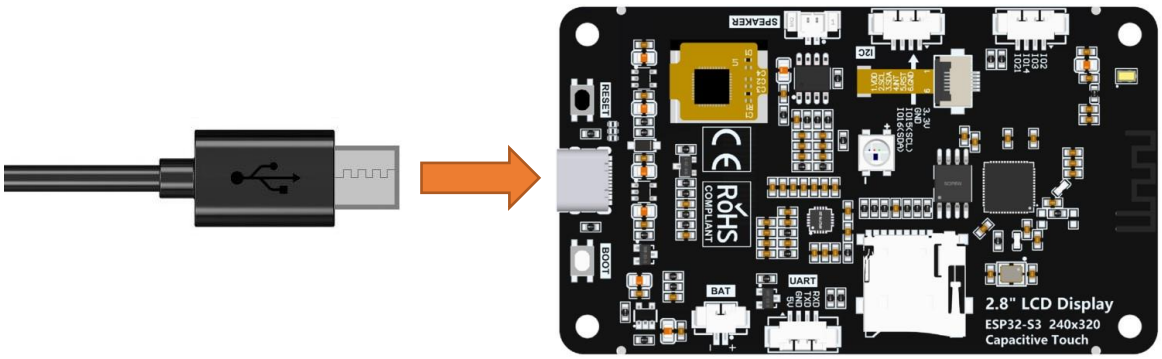
Continue to use the following color model to equalize the color distribution of the LED and gradually change.

Component List

Freenove ESP32 S3 Display x 1 	USB cable x1 

Circuit

Connect Freenove ESP32 S3 Display to the computer with USB cable.



Sketch

Open "**Sketch_02.2_Rainbow**" folder under **"Freenove_ESP32_S3_Display\NonTouch\Sketches"** and double-click "**Sketch_02.2_Rainbow**".

Sketch_02.2_Rainbow

The following is the program code:

```
/*
* @ File:   Sketch_02.2_Rainbow.ino
* @ Author: [Zhentao Lin]
* @ Date:   [2025-06-11]
*/
#include "Freenove_WS2812_Lib_for_ESP32.h"

#define LEDS_COUNT  1
#define LEDS_PIN    42
#define CHANNEL     0

Freenove_ESP32_WS2812 strip = Freenove_ESP32_WS2812(LEDS_COUNT, LEDS_PIN, CHANNEL, TYPE_GRB);

void setup() {
  strip.begin();
  strip.setBrightness(20);
}

void loop() {
  for (int j = 0; j < 255; j += 2) {
    for (int i = 0; i < LEDS_COUNT; i++) {
      strip.setLedColorData(i, strip.Wheel((i * 256 / LEDS_COUNT + j) & 255));
    }
    strip.show();
    delay(10);
  }
}
```

Code Explanation

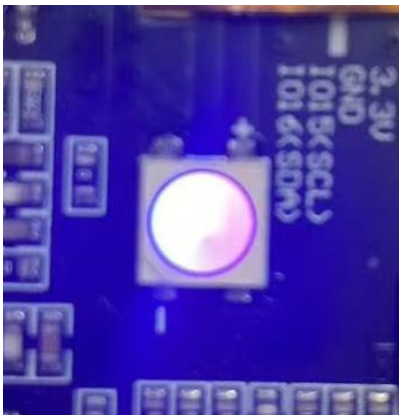
In the loop(), two "for" loops are used, the internal "for" loop(for-j) is used to set the color of each LED, and the external "for" loop(for-i) is used to change the color, in which the self-increment value in i+=2 can be changed to change the color step distance. Changing the delay parameter changes the speed of the color change. `strip.Wheel((i * 256 / LEDS_COUNT + j) & 255)` will take color from the color model at equal intervals starting from i.

```
for (int j = 0; j < 255; j += 2) {
    for (int i = 0; i < LEDS_COUNT; i++) {
      strip.setLedColorData(i, strip.Wheel((i * 256 / LEDS_COUNT + j) & 255));
    }
```

```
23          strip.show();
24          delay(10);
25      }
```

This code achieves the function of printing data on serial monitor. Click "**Upload**" to upload the code to Freenove_ESP32_S3_Display.

After uploading the code, the LED lights will display a smooth rainbow gradient effect.



Reference

```
bool ledcAttachChannel(uint8_t pin, uint32_t freq, uint8_t resolution, uint8_t channel);
```

This function binds the specified PWM channel to a GPIO pin and configures the frequency and resolution of the PWM signal.

Parameters:

pin: GPIO pin number to bind

freq: PWM signal frequency in Hz

resolution: Bit depth for PWM duty cycle resolution (range: 1-16 bits).

For example, 8 represents 2^8 (0~255) levels.

channel: PWM channel number to assign (integer)

```
void ledcWrite(uint8_t channel, uint32_t duty);
```

This function sets the duty cycle for a specified PWM channel to control output signal intensity.

Parameters:

channel: PWM channel number

duty: Duty cycle value (range determined by resolution bit depth)

Chapter 3 Button

The Boot button on the Freenove ESP32 S3 Display can be configured as a regular input button after the program starts.

Project 3.1 Button RGB

In the previous chapter, we learned about RGB LEDs. In this section, we will further explore how to integrate RGB LEDs with buttons.

Related Knowledge

Button

In embedded systems, buttons are one of the most common human-machine input devices. When a button is not pressed, its opposing contacts are connected while adjacent contacts remain disconnected. Conversely, when the button is pressed, adjacent contacts connect while opposing contacts disconnect.

Usage of the Boot Button on the Freenove CYD Board:

Manual Entry into Download Mode:

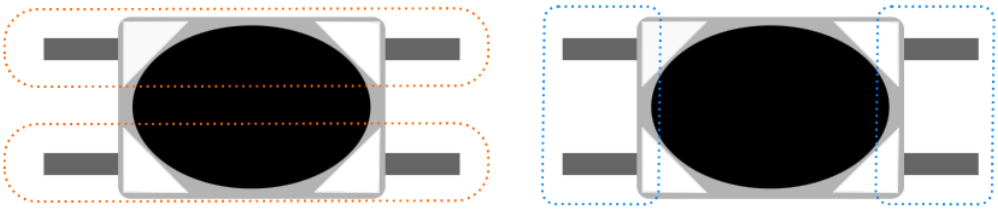
When the board is powered on or reset, pressing and holding the Boot button forces the board into download mode.

In this mode, users can upload programs to the board via the serial port.

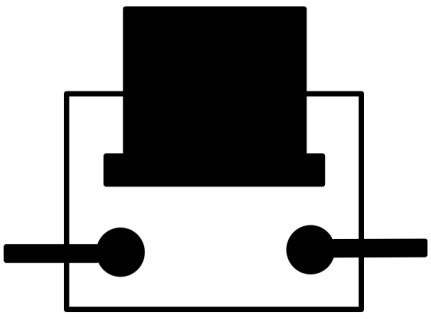
Use as a Regular Input Button:

If the Boot button is not pressed during power-on or reset, the board enters normal operation mode.

Once in operation mode, the Boot button can function as GPIO0, typically configured as a standard input button for user interaction.

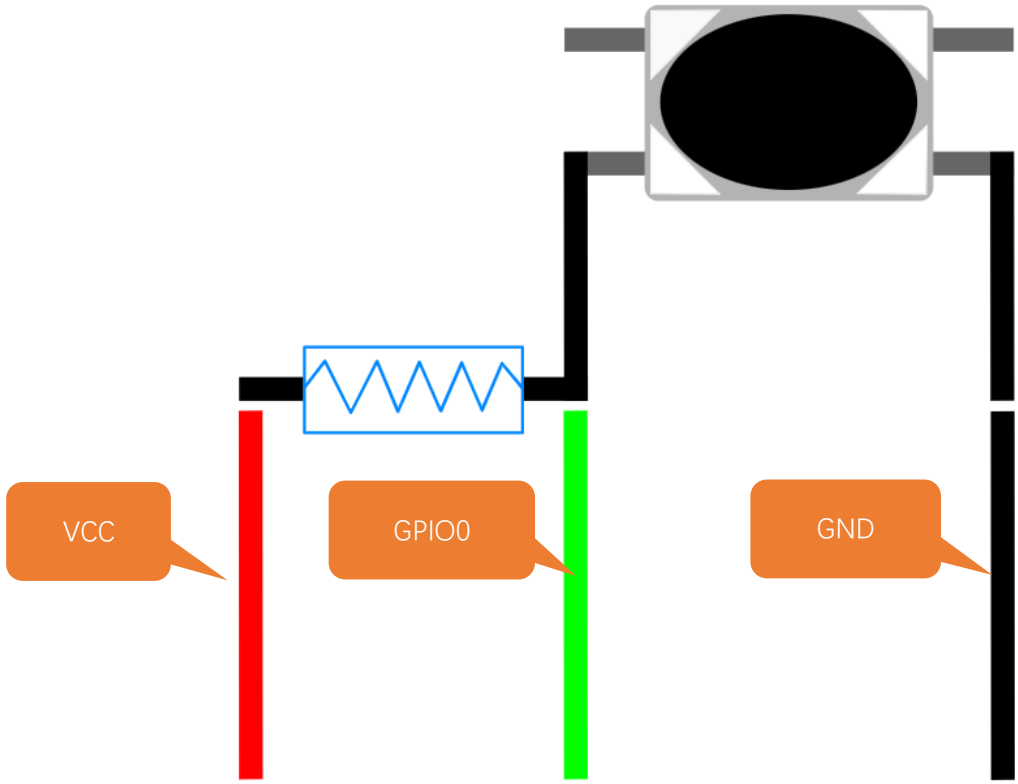


Before pressing After pressing



Pull-up Resistor

The primary function of a pull-up resistor is to ensure that the button maintains a **stable high-level state** when not pressed, preventing false triggering caused by floating pins or signal noise. The pull-up resistor is connected between the GPIO pin and VCC.



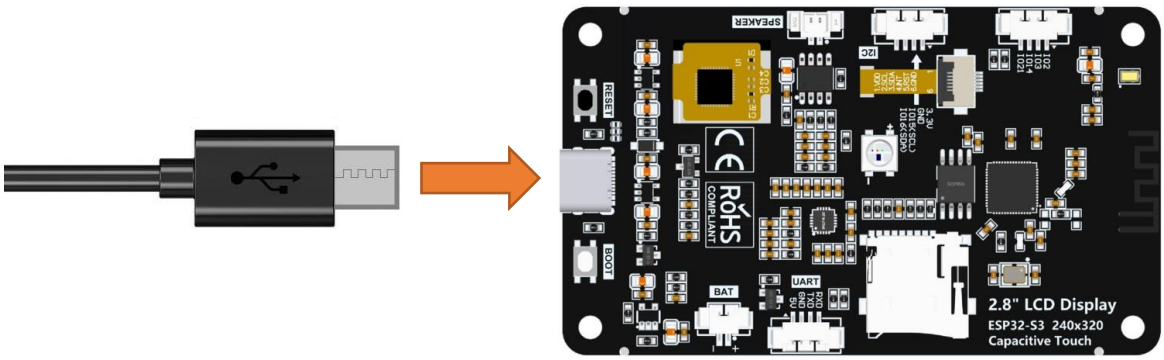
When the button is not pressed, Vcc is connected to GPIO0 through the resistor, resulting in a high-level signal at this pin.
When the button is pressed, GPIO0 is connected to GND, resulting in a low-level signal.
Therefore, by reading the GPIO0 pin state, we can determine whether the button is pressed or not.

Component List

Freenove ESP32 S3 Display x 1 	USB cable x1 

Circuit

Connect Freenove ESP32 S3 Display to the computer with USB cable.



Sketch

Open "**Sketch_03.1_Button_RGB**" folder under **"Freenove_ESP32_S3_Display\NonTouch\Sketches"** and double-click "**Sketch_03.1_Button_RGB.ino**".

Sketch_03.1_Button_RGB

The following is the program code:

```
/*
* @ File:   Sketch_03.1_Button_RGB.ino
* @ Author: [Zhentao Lin]
* @ Date:   [2025-08-19]
*/

#include <Arduino.h>
#include "Freenove_WS2812_Lib_for_ESP32.h"

#define KEY_PIN 0
#define LEDS_COUNT 1
#define LEDS_PIN   42
#define CHANNEL    0

Freenove_ESP32_WS2812 strip = Freenove_ESP32_WS2812(LEDS_COUNT, LEDS_PIN, CHANNEL, TYPE_GRB);

uint8_t m_color[5][3] = { { 255, 0, 0 }, { 0, 255, 0 }, { 0, 0, 255 }, { 255, 255, 255 }, { 0,
0, 0 } };

static unsigned int keyCount = 0;
static unsigned int lastKeyCount = 1;

void buttonInit(void) {
```

```
24      pinMode(KEY_PIN, INPUT_PULLUP);
25  }
26
27  bool readButton(void) {
28      return digitalRead(KEY_PIN);
29  }
30
31  void switchRGB(int value) {
32      int colorIndex = value % 4;
33      switch (colorIndex) {
34        case 1:
35          strip.setLedColorData(0, m_color[0][0], m_color[0][1], m_color[0][2]);
36          strip.show();
37          break;
38        case 2:
39          strip.setLedColorData(0, m_color[1][0], m_color[1][1], m_color[1][2]);
40          strip.show();
41          break;
42        case 3:
43          strip.setLedColorData(0, m_color[2][0], m_color[2][1], m_color[2][2]);
44          strip.show();
45          break;
46        default:
47          strip.setLedColorData(0, m_color[4][0], m_color[4][1], m_color[4][2]);
48          strip.show();
49          break;
50      }
51  }
52
53  void setup() {
54      strip.begin();
55      strip.setBrightness(10);
56      buttonInit();
57      Serial.println("ESP32-S3 initialization completed!");
58  }
59
60  void loop() {
61      if (readButton() == 0) {
62        delay(20);
63        if (readButton() == 0) {
64          keyCount++;
65          while (!readButton());
66        }
67      }
```

```
68      if (keyCount != lastKeyCount) {
69        switchRGB(keyCount);
70        lastKeyCount = keyCount;
71      }
72  }
```

Code Explanation

Define the pins for the button and the RGB LED.

```
10  #define KEY_PIN 0
11  #define LEDS_COUNT 1
12  #define LEDS_PIN    42
13  #define CHANNEL     0
```

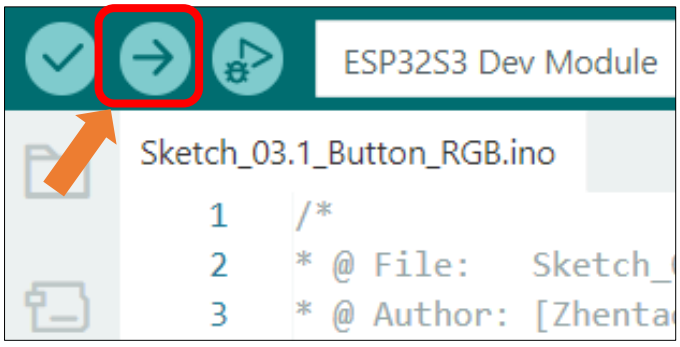
Initialize the RGB LED and the button.

```
54  strip.begin();
55  strip.setBrightness(10);
56  buttonInit()
```

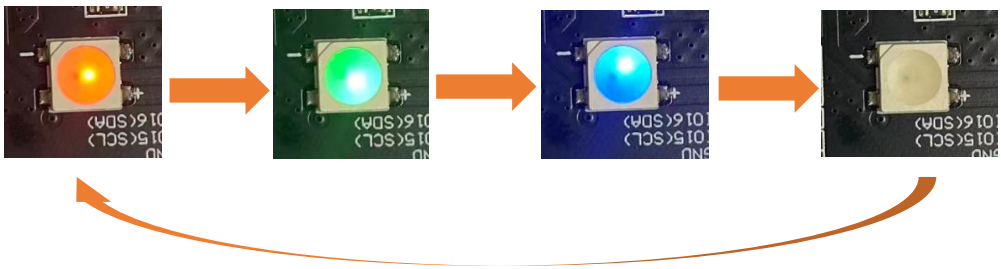
Control the color of the RGB LED through the button.

```
61   if (readButton() == 0) {
62     delay(20);
63     if (readButton() == 0) {
64       keyCount++;
65       while (!readButton());
66     }
67   }
68   if (keyCount != lastKeyCount) {
69     switchRGB(keyCount);
70     lastKeyCount = keyCount;
71   }
```

Click "**Upload**" to upload the code to Freenove_ESP32_S3_Display.



The RGB LEDs will cycle through Red -> Green -> Blue -> OFF every 500ms after code upload.



Chapter 4 Button Interrupt

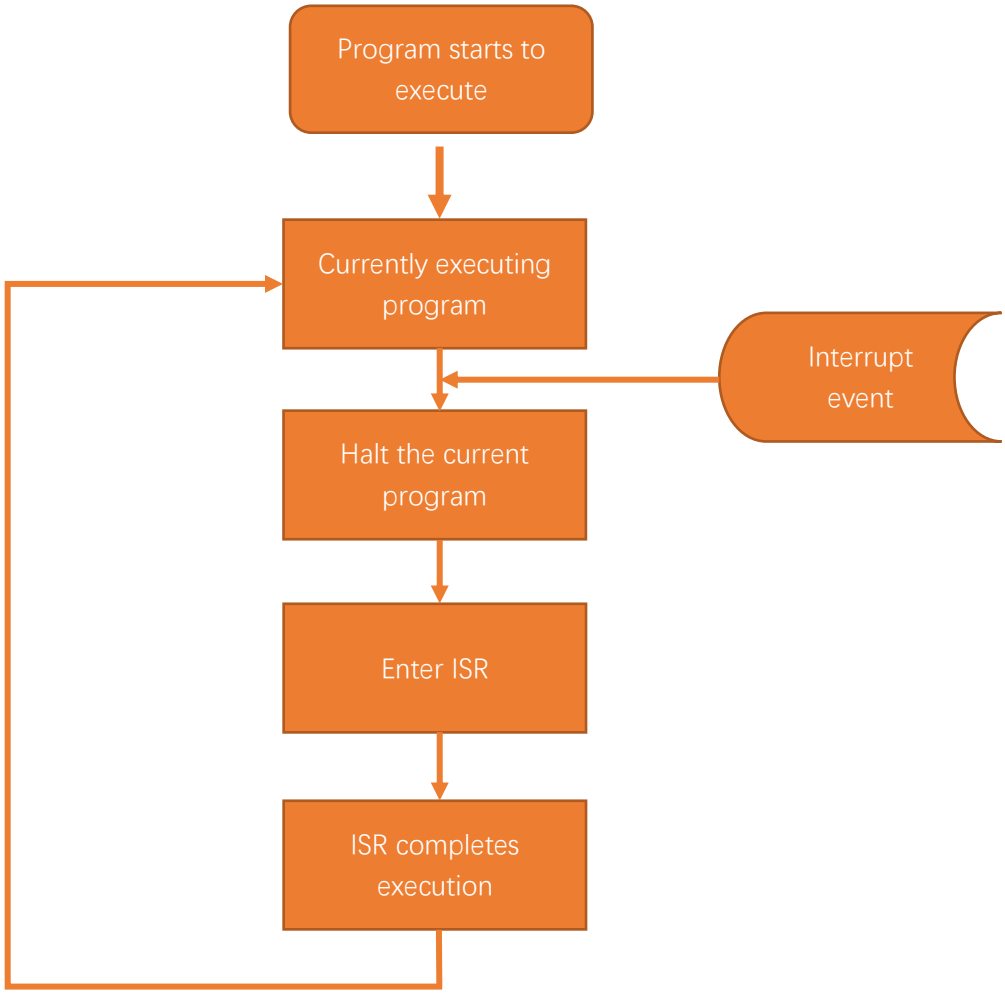
In this chapter will learn to use interrupt to detect the button state.

Project 4.1 Button Interrupt UART

Related Knowledge

How Interrupts Work

When an interrupt event is triggered, the Raspberry Pi Pico (W) receives an interrupt signal. At this moment, the Raspberry Pi Pico (W) pauses the currently executing program and instead executes the Interrupt Service Routine (ISR). The ISR contains the code that needs to be processed after the interrupt event occurs. Once the ISR execution is complete, the Raspberry Pi Pico will return to the state it was in before the interrupt occurred and continue executing the paused program, as shown in the diagram below.

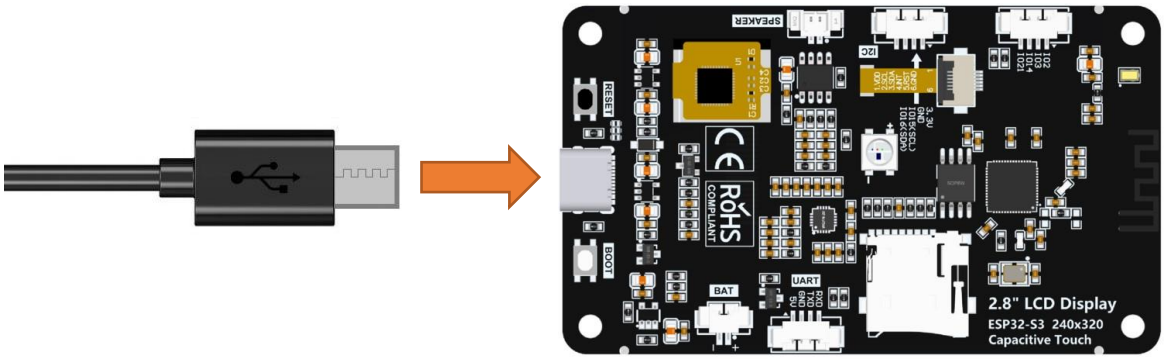


Component List

Freenove ESP32 S3 Display x 1 	USB cable x1 

Circuit

Connect Freenove ESP32 S3 Display to the computer with USB cable.



Sketch

Open "**Sketch_04.1_Button_Interrupt_UART**" folder under **"Freenove_ESP32_S3_Display\NonTouch\Sketches"** and double-click "**Sketch_04.1_Button_Interrupt_UART.ino**".

Sketch_04.1_Button_Interrupt_UART

The following is the program code:

```
1  /*
2  * @ File:    Sketch_04.1_Button_Interrupt_UART.ino
3  * @ Author: [Zhentao Lin]
4  * @ Date:    [2025-08-19]
5  */
6
7  #include <Arduino.h>
8
9  #define KEY_PIN 0
10 volatile int interruptCounter = 0;
```

```
11  int lastValue = 0;
12  portMUX_TYPE mux = portMUX_INITIALIZER_UNLOCKED;
13
14  void buttonInit(void) {
15    pinMode(KEY_PIN, INPUT_PULLUP);
16    attachInterrupt(digitalPinToInterrupt(KEY_PIN), handleInterrupt, FALLING);
17  }
18
19  void delayDebounce(unsigned int delayTime) {
20    unsigned int i = delayTime;
21    while (--i > 0) {
22      if (digitalRead(KEY_PIN) == HIGH) {
23        return;
24      }
25    }
26  }
27
28  void handleInterrupt(void) {
29    portENTER_CRITICAL_ISR(&mux);
30    interruptCounter++;
31    delayDebounce(20);
32    portEXIT_CRITICAL_ISR(&mux);
33  }
34
35  void setup() {
36    Serial.begin(115200);
37    buttonInit();
38    Serial.println("ESP32-S3 initialization completed!");
39  }
40
41  void loop() {
42    if (interruptCounter != lastValue) {
43      Serial.printf("interruptCounter: %d\r\n", interruptCounter);
44      lastValue = interruptCounter;
45    }
46  }
```

Code Explanation

Define the pin of the button.

```
9   #define KEY_PIN 0
```

Configure the button pin hardware interrupt to trigger on the falling edge signal.

```
16  attachInterrupt(digitalPinToInterrupt(KEY_PIN), handleInterrupt, FALLING);
```

The interrupt processing function, counting how many times the button is pressed.

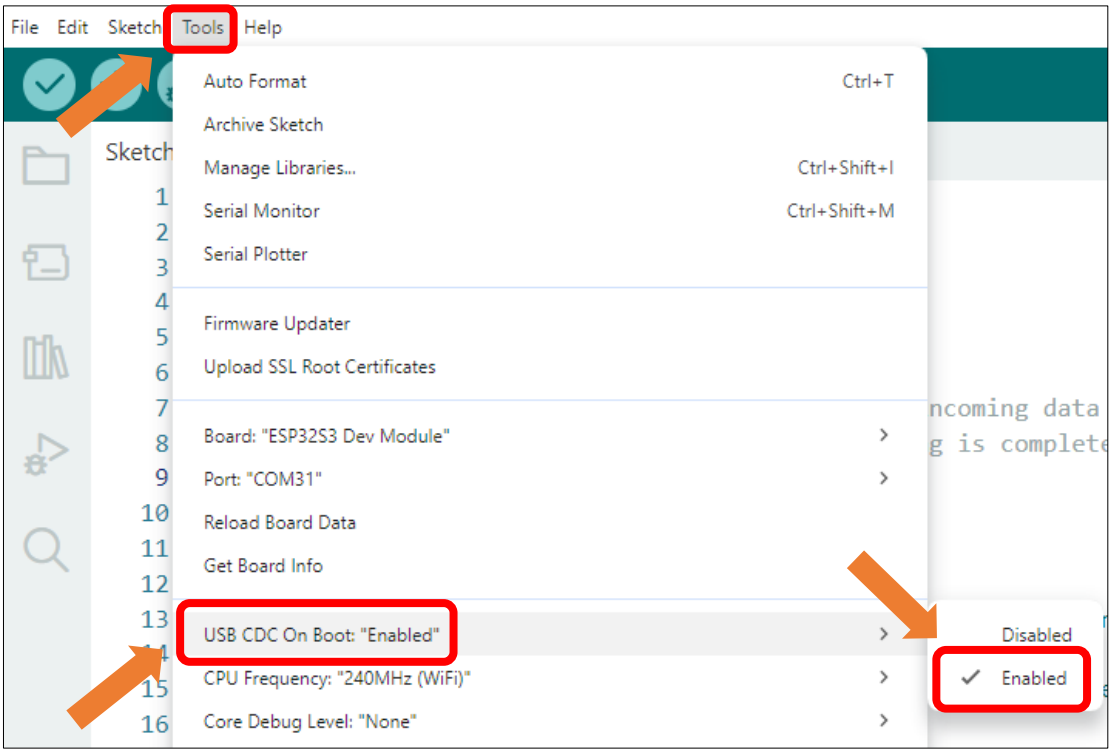
```
28  void handleInterrupt(void) {
29    portENTER_CRITICAL_ISR(&mux);
```

```
30      interruptCounter++;
31      delayDebounce(20);
32      portEXIT_CRITICAL_ISR(&mux);
33  }
```

When the number of button presses changes, print new data in the serial monitor.

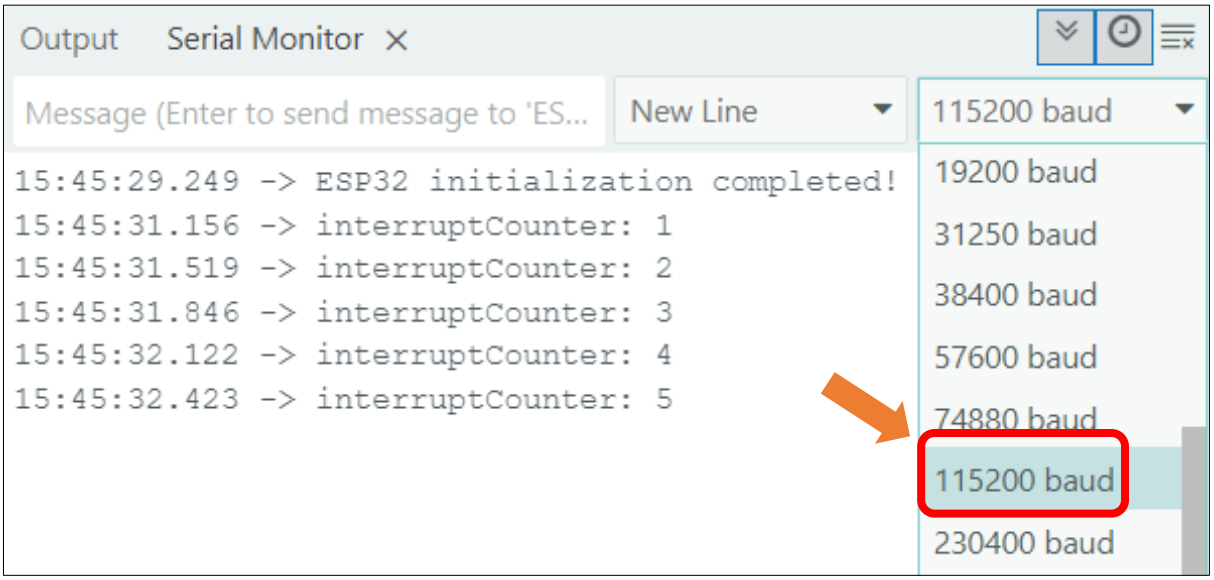
```
41  void loop() {
42      if (interruptCounter != lastValue) {
43        Serial.printf("interruptCounter: %d\r\n", interruptCounter);
44        lastValue = interruptCounter;
45      }
46  }
```

Enable the "USB CDC On Boot" feature.



Click "**Upload**" to upload the code to Freenove_ESP32_S3_Display.

When the code finishes uploading, open the serial monitor and set the baud rate to 115200. The serial monitor will print the number of button presses.



Reference

```
void attachInterrupt(uint8_t pin, void (*)(void), int mode);
```

external interrupts

Parameters:

pin: The digital pin number for the interrupt

void: The name of the interrupt function

mode:

- LOW (trigger on low level)
- CHANGE (trigger on change)
- RISING (trigger on rising edge)
- FALLING (trigger on falling edge)

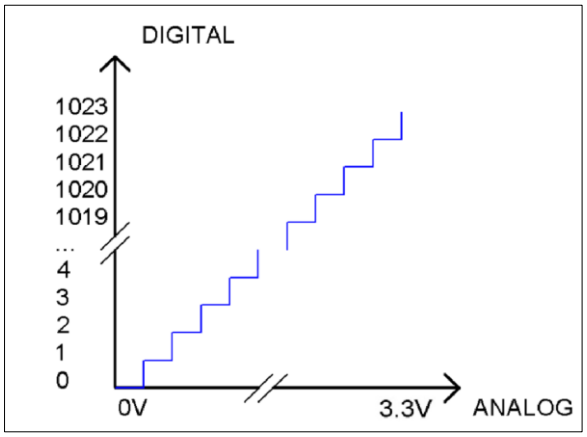
Chapter 5 Battery Voltage

Project 5.1 Battery Voltage

Related Knowledge

ADC

ADC is an electronic integrated circuit used to convert analog signals such as voltages to digital or binary form consisting of 1s and 0s. The range of our ADC on Raspberry Pi Pico (W) is 10 bits, which means the resolution is 2^10=1024, and it represents a range (at 3.3V) will be divided equally to 1024 parts. The rage of analog values corresponds to ADC values. So the more bits the ADC has, the denser the partition of analog will be and the greater the precision of the resulting conversion.

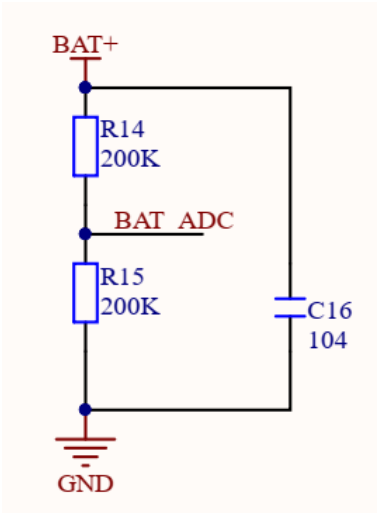


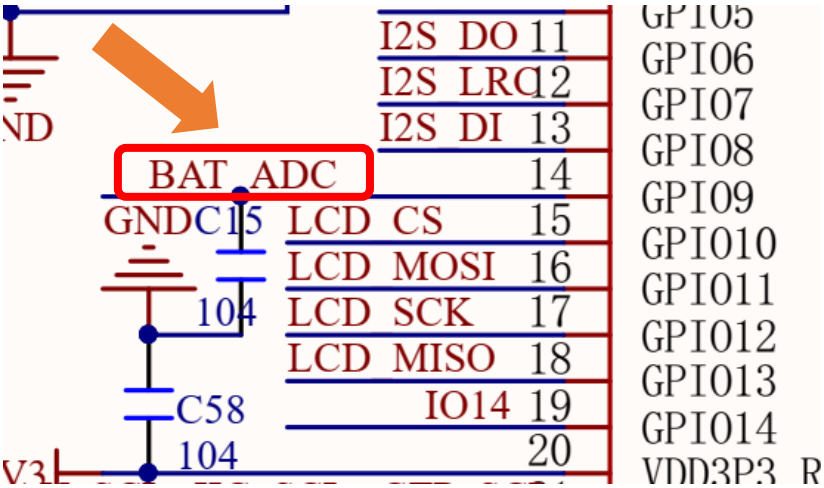
Subsection 1: the analog in rang of 0V---3.3/1023 V corresponds to digital 0;
Subsection 2: the analog in rang of 3.3/1023 V---2*3.3 /1023V corresponds to digital 1;
The following analog will be divided accordingly.
The conversion formula is as follows:

$$ADC\ Value = \frac{\text{Analog Voltage}}{3.3} * 1023$$

Battery Voltage

From the schematic diagram, we can know that Freenove_ESP32_S3_Display reads the battery voltage through the ADC pin (GPIO9).





The ADC pin on this board has a voltage sampling range of 0~3.3V. However, when powered by a 3.7V lithium battery, the output voltage in a fully charged state can reach 4.2V, exceeding the ADC's rated input range.

To ensure accurate voltage measurement and ADC pin protection, the circuit employs a resistive voltage divider that scales the battery voltage down by a factor of 0.5. At a full charge (4.2V), the divided voltage at the ADC input is 2.1V, well within the safe operating range. This design enables reliable battery monitoring while preventing overvoltage damage to the ADC.

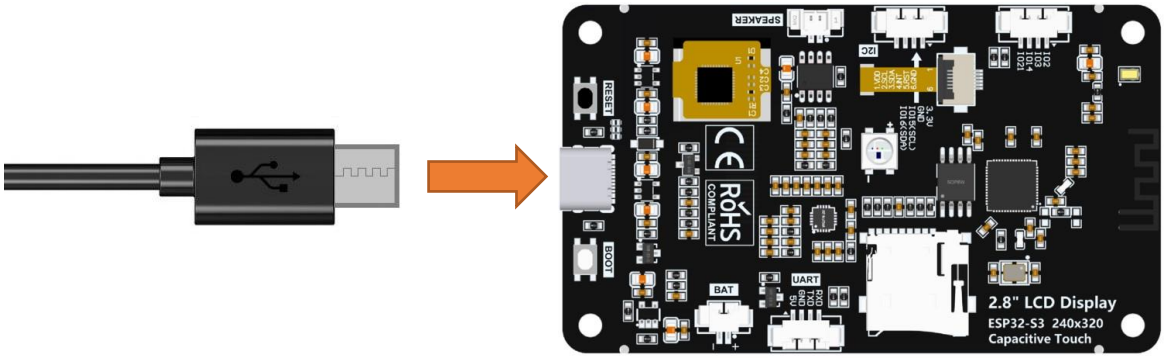
Please note that this kit does not include a lithium battery, please buy one yourself. For more information about battery, please refer to Battery.

Component List

Freenove ESP32 S3 Display x 1 	USB cable x1 

Circuit

Connect Freenove ESP32 S3 Display to the computer with USB cable.



Sketch

Next, we download the code to Freenove_ESP32_S3_Display to test Serial. Open "**Sketch_05.1_Battery_Voltage.ino**" folder under **"Freenove_ESP32_S3_Display\NonTouch\Sketches"** and double-click "**Sketch_05.1_Battery_Voltage.ino**".

Sketch_05.1_Battery_Voltage

The following is the program code:

```
1  /*
2  * @ File:    Sketch_05.1_Battery_Voltage.ino
3  * @ Author: [Zhentao Lin]
4  * @ Date:    [2025-08-19]
5  */
6
7  #define BAT_ADC_PIN  9
8
9  void setup(){
10    Serial.begin(115200);
11    pinMode(BAT_ADC_PIN, INPUT);
12 }
13
14 void loop() {
15    int adcValue= analogRead(BAT_ADC_PIN);
16    float batteryVoltage = analogReadMilliVolts(BAT_ADC_PIN) * 2.0 / 1000;
17    Serial.printf("Reading on Pin(%d), adcValue=%d, batteryVoltage2=%fV\n", BAT_ADC_PIN,
18 adcValue, batteryVoltage);
19    delay(300);
20 }
```

Code Explanation

Define the pins for the button and RGB LED.

```
7  #define BAT_ADC_PIN  9
```

Set the baud rate to 115200.

```
10  Serial.begin(115200);
```

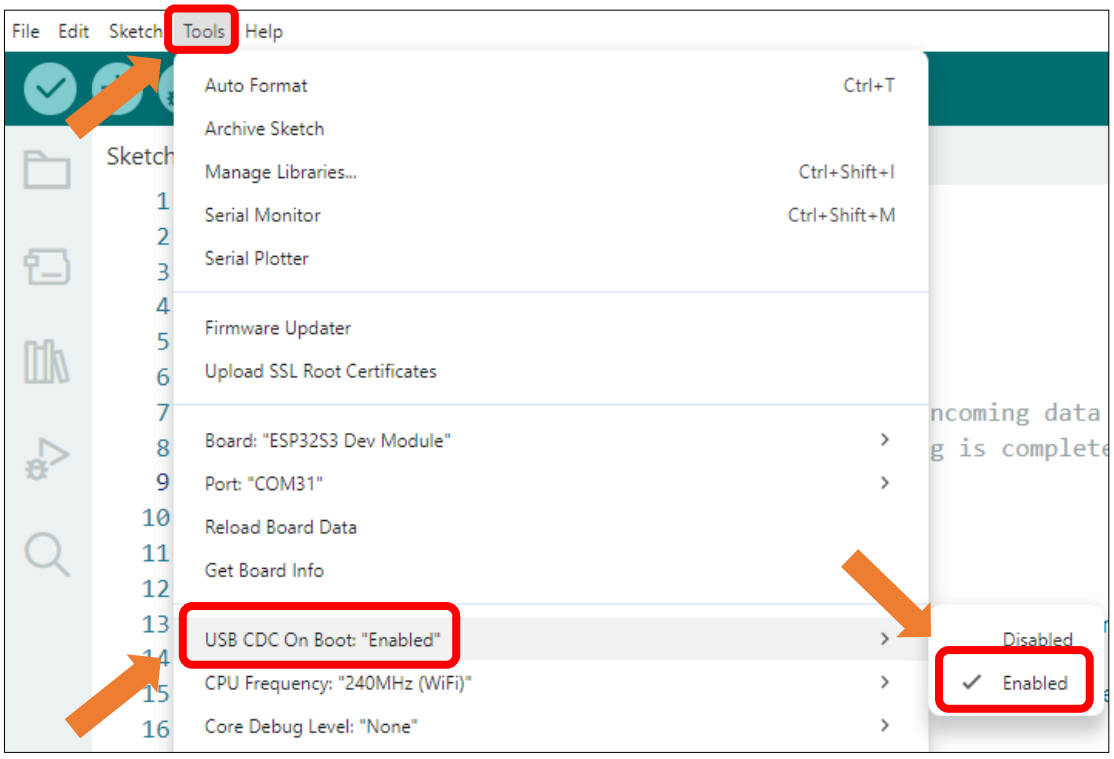
Set the battery voltage detecting pin to input mode.

```
11  pinMode(BAT_ADC_PIN, INPUT);
```

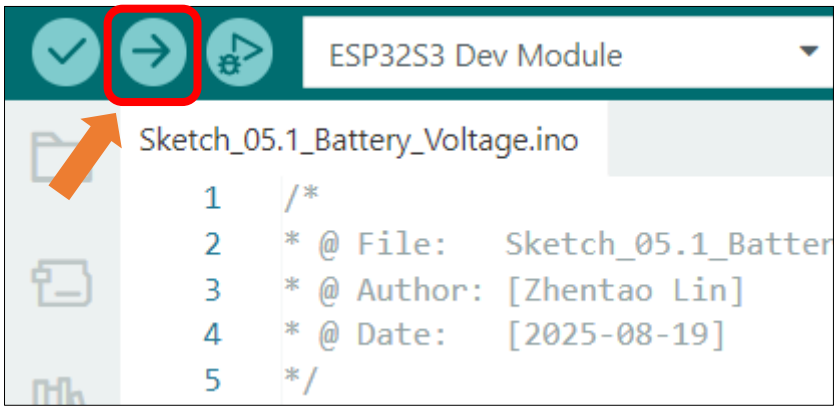
Measure the battery voltage every 300ms and outputs the reading to the serial monitor.

```
14  void loop() {
15      int adcValue= analogRead(BAT_ADC_PIN);
16      float batteryVoltage = analogReadMilliVolts(BAT_ADC_PIN) * 2.0 / 1000;
17      Serial.printf("Reading on Pin(%d), adcValue=%d, batteryVoltage2=%fV\n", BAT_ADC_PIN,
18  adcValue, batteryVoltage);
19      delay(300);
20  }
```

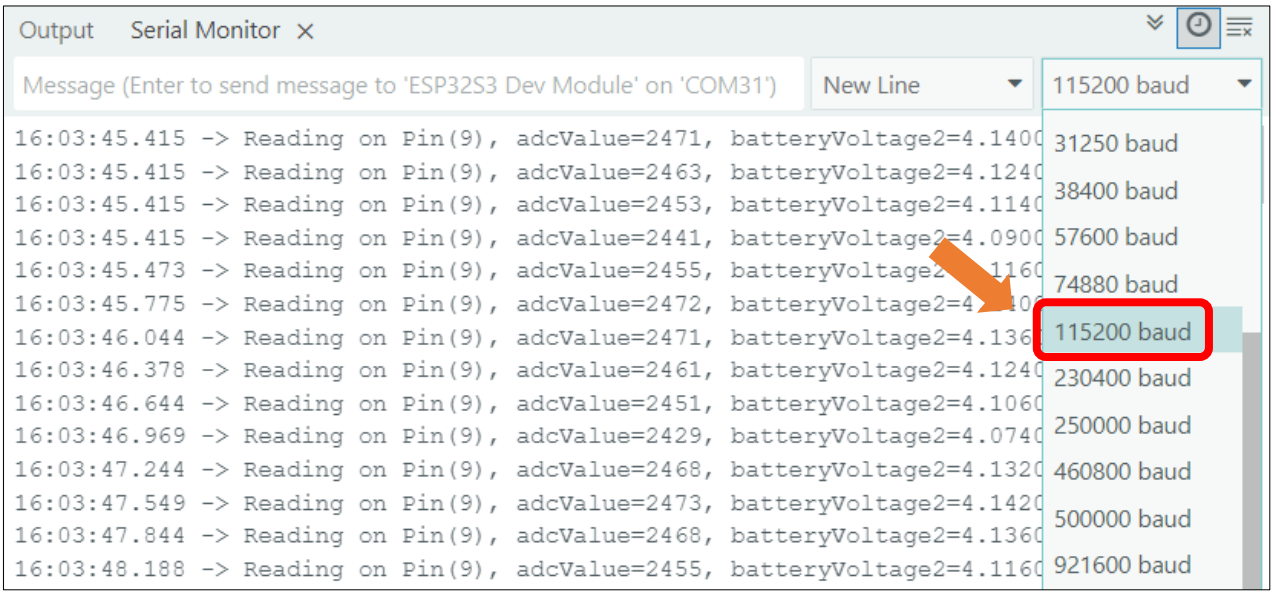
Enable the "USB CDC On Boot" feature.



Click "**Upload**" to upload the code to Freenove_ESP32_S3_Display.



Once the program starts, open the serial monitor to view the battery voltage readings, updated every 300ms.



Reference

uint32_t analogReadMilliVolts(uint8_t pin)

This function reads the voltage directly from the analog pin and returns the value in millivolts (mV), with a maximum return value of 3300mV (3.3V)

Chapter 6 SD Card

Project 6.1 SD Test

Component List

Freenove ESP32 S3 Display x 1	USB cable x1
	

Please note that this kit does not include SD card and card reader, please buy them by yourself.

Component Knowledge

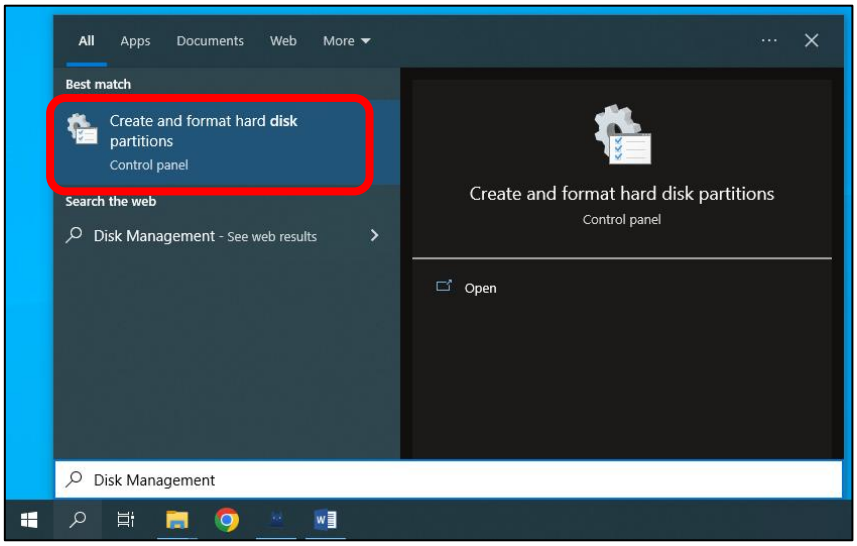
Format SD card

To initiate the project, you'll first need to prepare a blank SD card and a compatible card reader. The initial setup involves assigning a drive letter to the SD card and formatting it properly. Below you'll find system-specific instructions to complete these preparatory steps. Please follow the guide corresponding to your operating system.

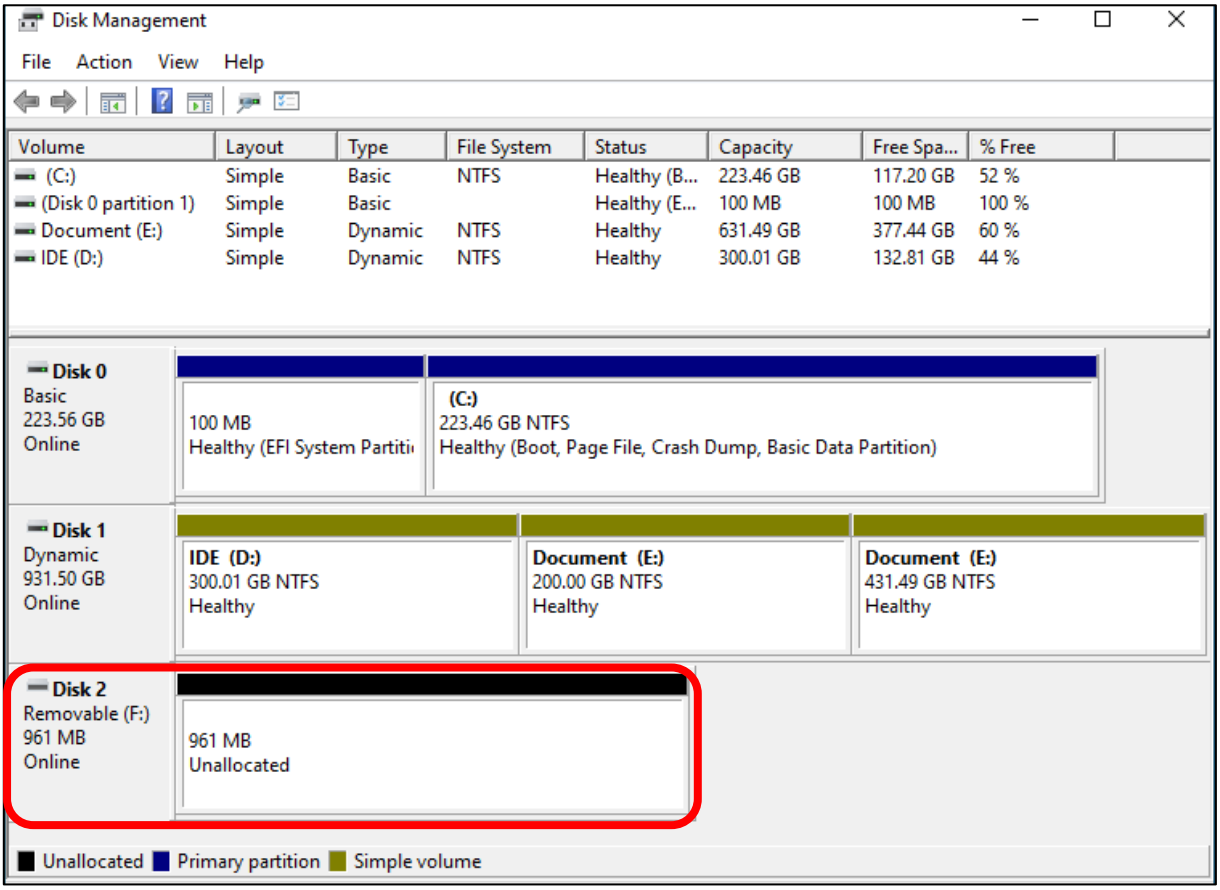
Windows

Insert the SD card into the card reader, and then insert the card reader into the computer.

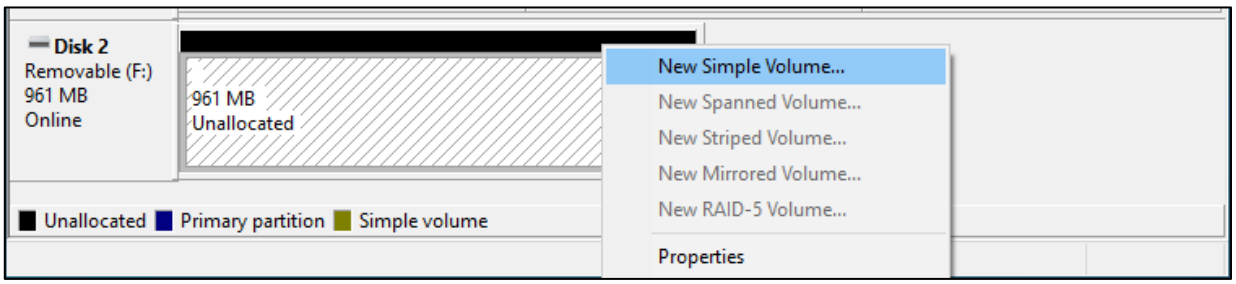
In the Windows search box, enter "Disk Management" and select "Create and format hard disk partitions".



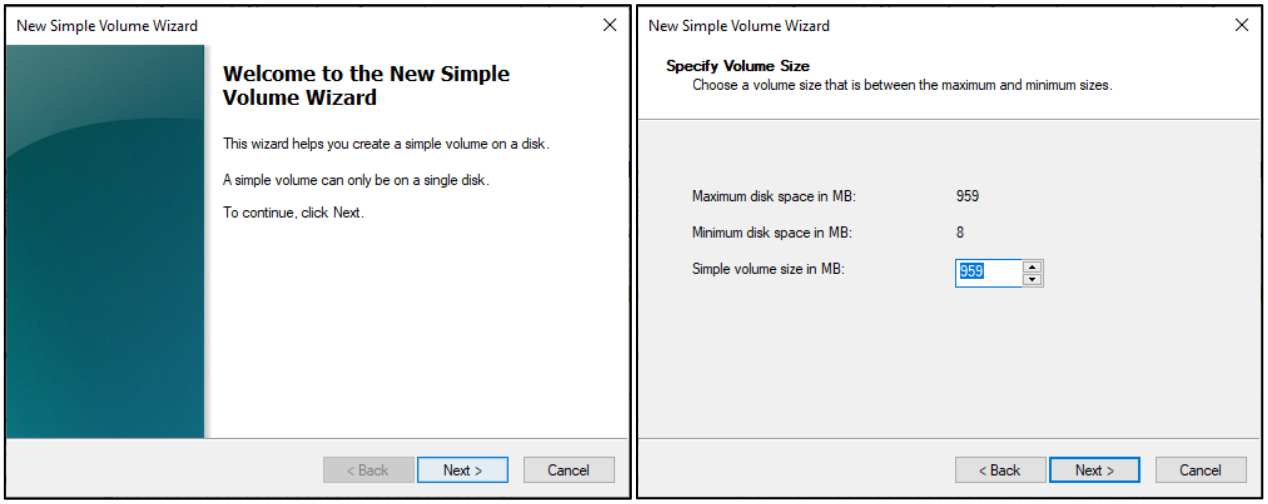
In the newly opened window, locate an unallocated volume with a size close to 1GB.



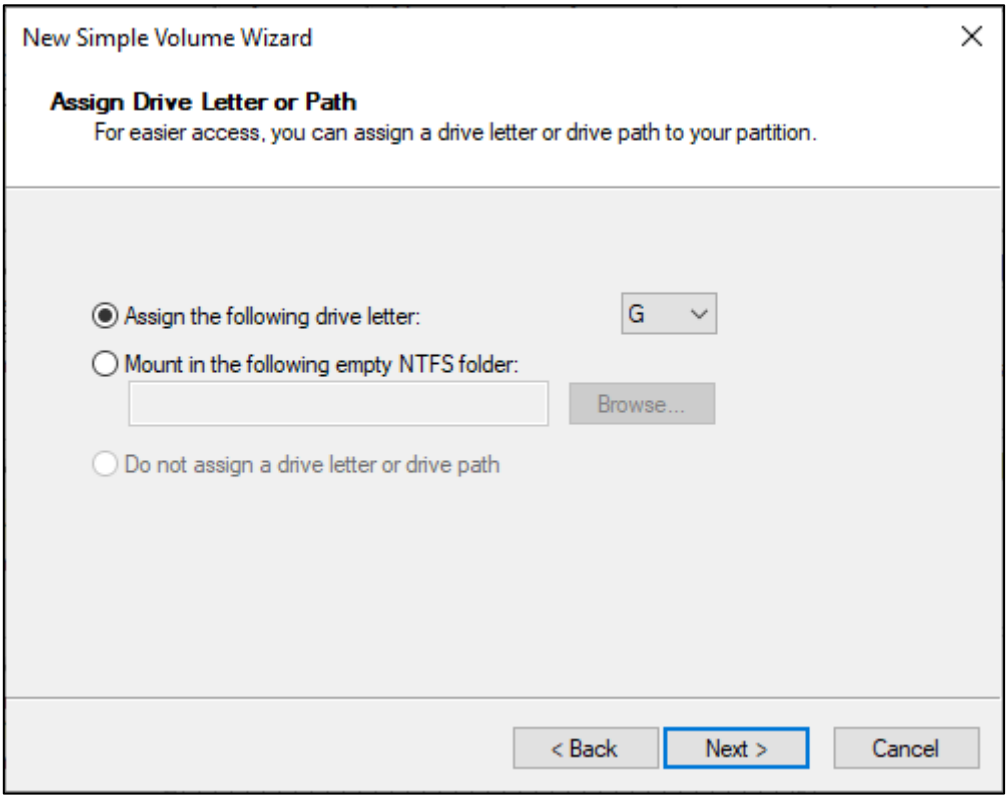
Click to select the volume, right-click and select "New Simple Volume".



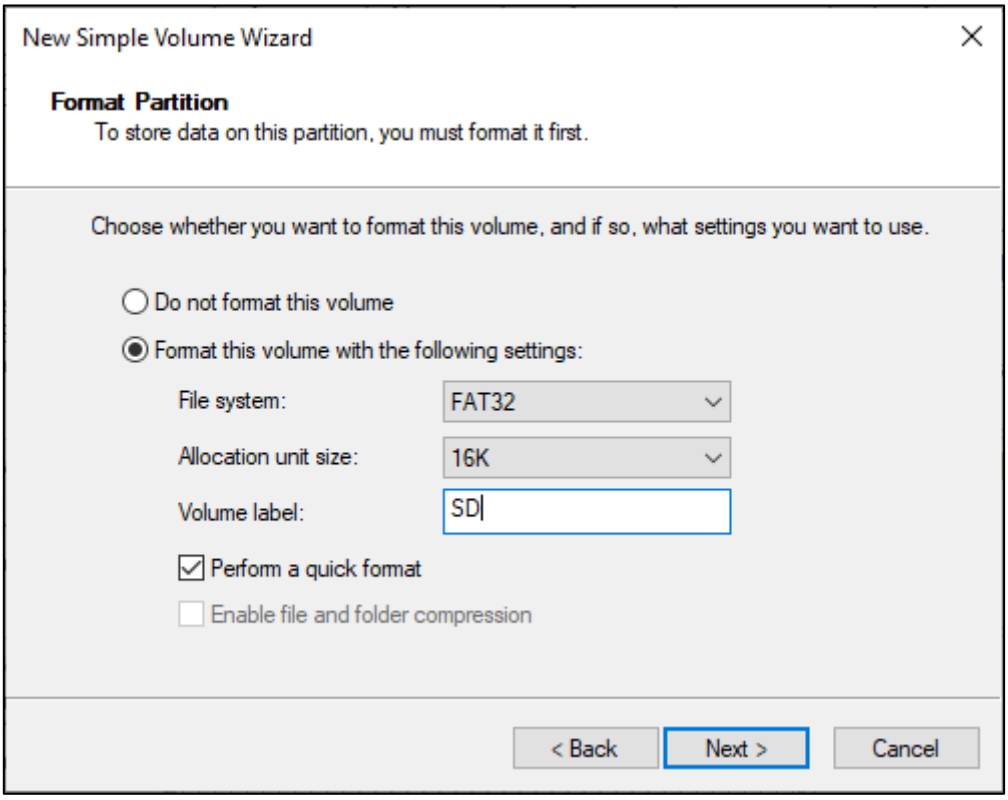
Click Next.



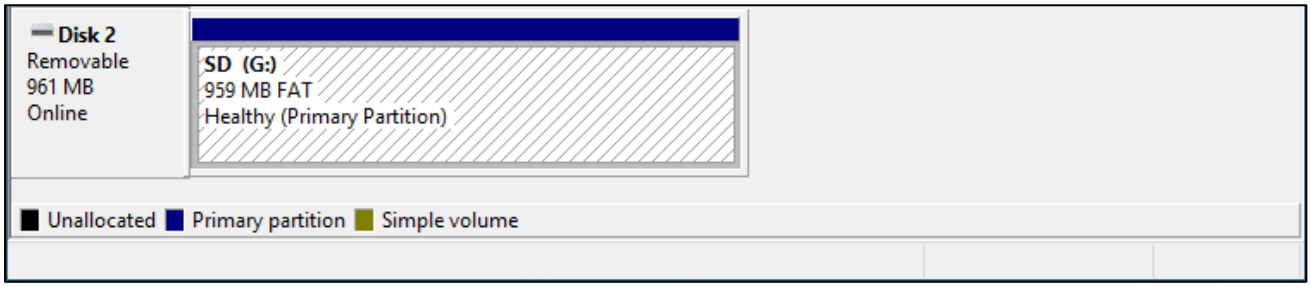
On the right-hand side, you can select a preferred drive letter for the SD card or simply proceed with the default setting by clicking on the "Next" button.



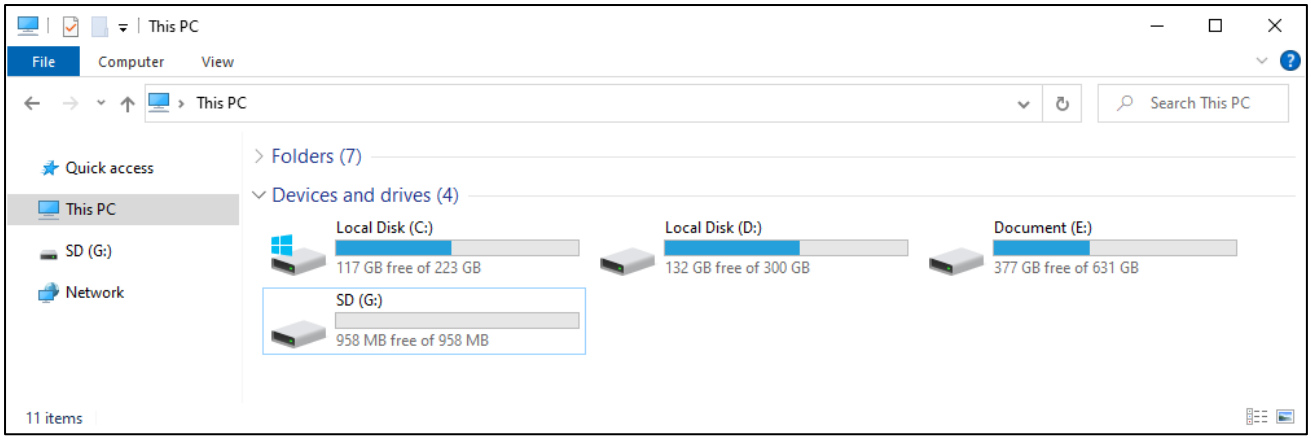
When formatting the SD card, select the file system as FAT (or FAT32) and set the allocation unit size to 16K. You can set the volume label to any name of your choice. Once you have made these selections, click on the "Next" button to proceed.



Click Finish. Wait for the SD card initialization to complete.

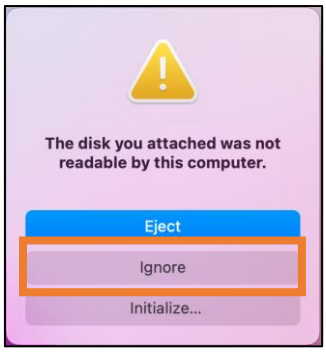


After completing the formatting process, you should be able to see the SD card in the "This PC" section of your computer.

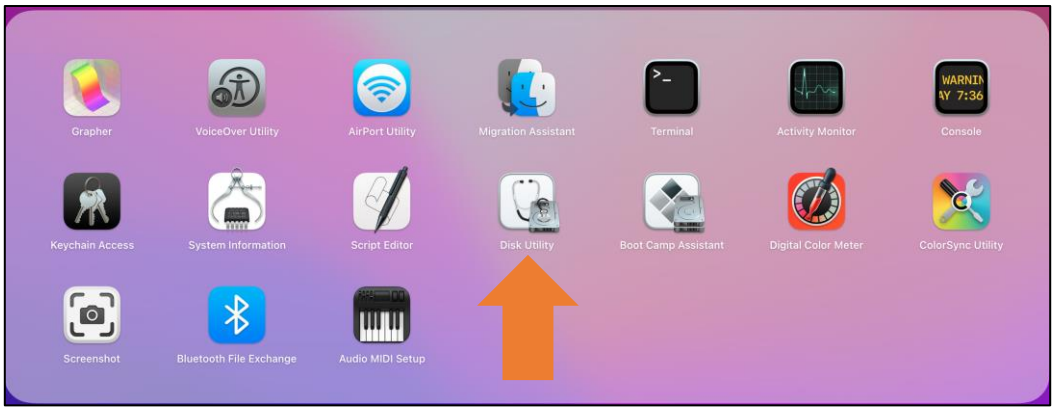


MAC

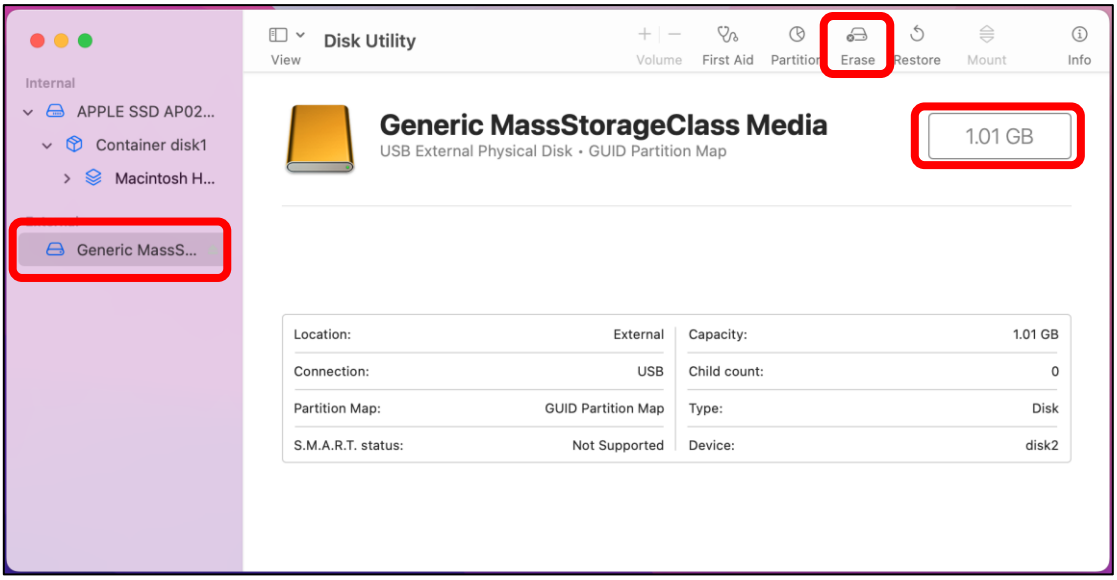
Insert the SD card into the card reader and then insert the card reader into your computer. Some computers may display a prompt with the following message. In this case, please click on the "Ignore" option to proceed.



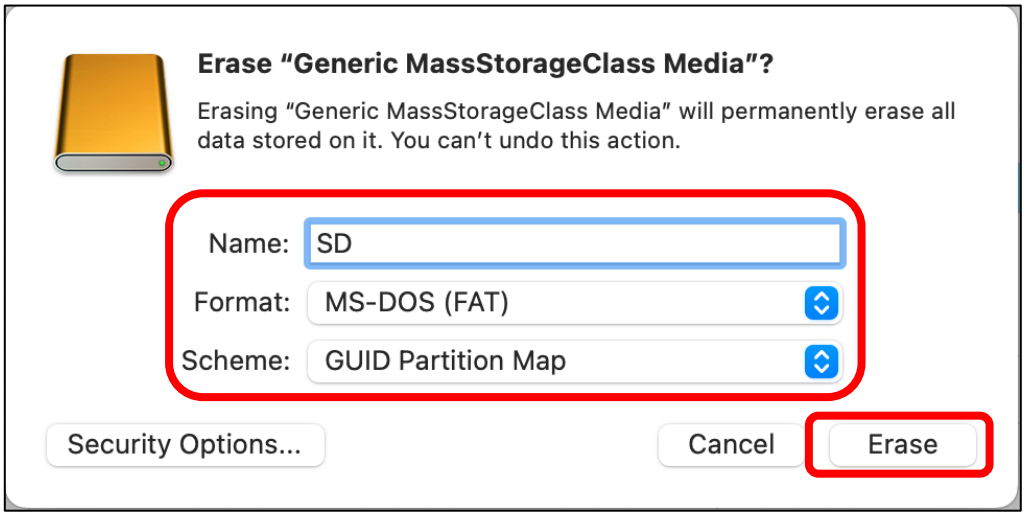
Find "Disk Utility" in the MAC system and click to open it.



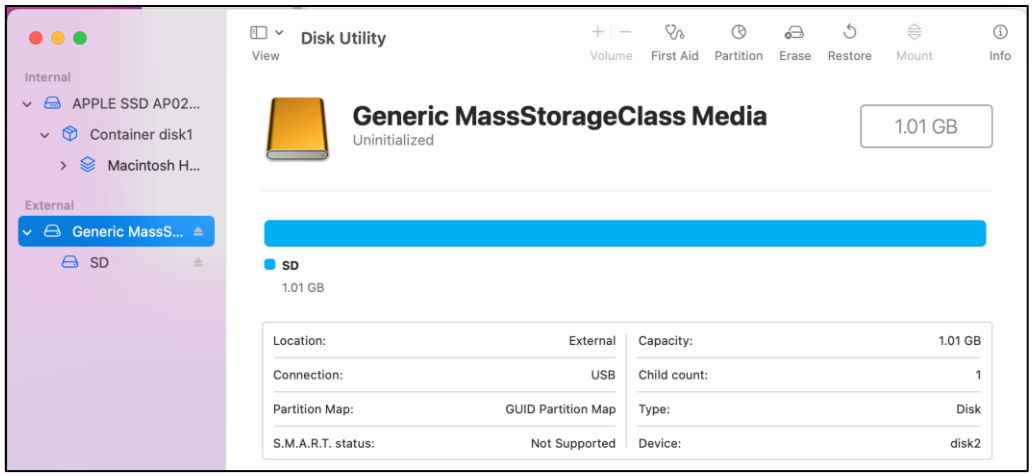
Select "Generic MassStorageClass Media", note that its size is about 1G. It is important to select the correct item to avoid accidentally erasing other device. Once you have verified that you have selected the correct device, click on the "Erase" button to proceed with erasing the SD card.



Select the configuration as shown in the figure below, and then click "Erase".



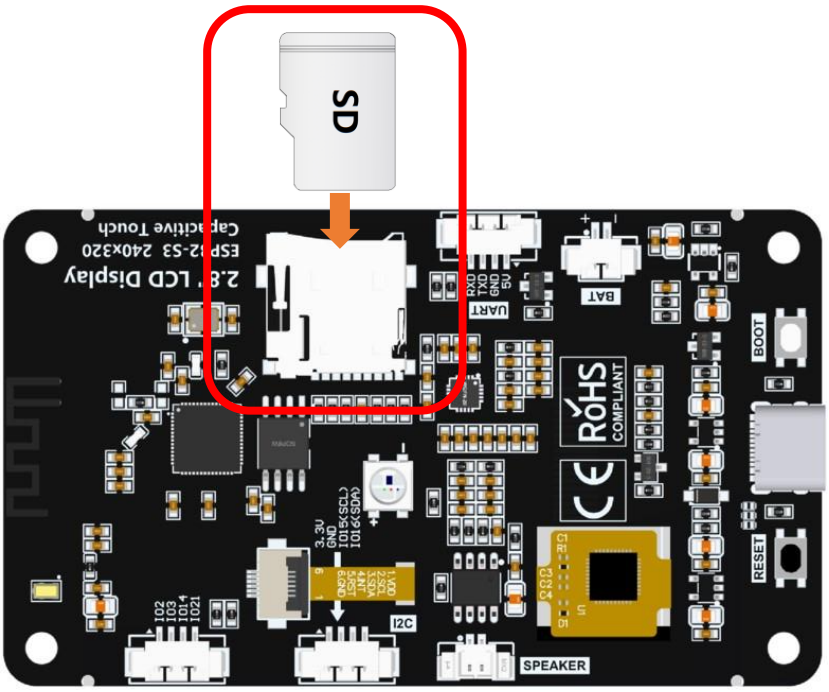
Please wait for the formatting process to complete. Once it is finished, the interface should resemble the image below. You should now be able to see a new disk named "SD" on your desktop.



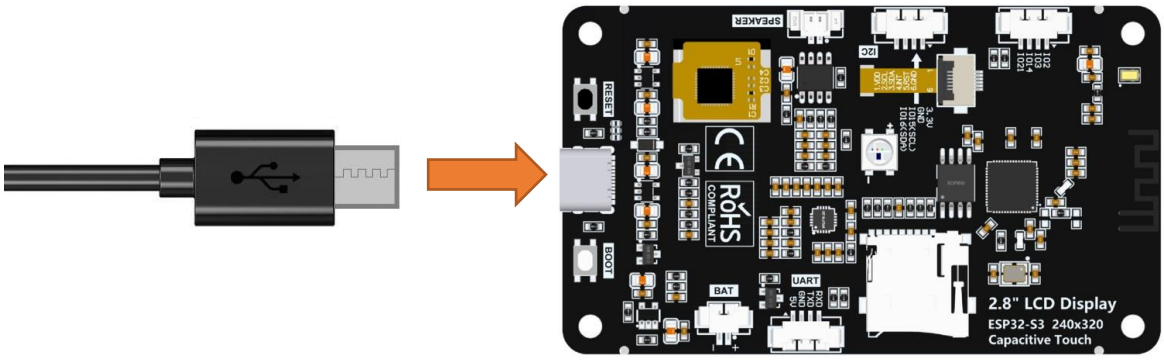
Circuit

Before connecting the USB cable, insert the SD card into the SD card slot on the back of the ESP32-S3.

Please note that this kit does not include SD card and card reader; please buy them yourself.



Connect Freenove ESP32-S3 to the computer using the USB cable.



If you have any concerns, please feel free to contact us via support@freenove.com

Sketch

Next, we download the code to Freenove_ESP32_S3_Display to test Serial. Open "**Sketch_06.1_SD_Test.ino**" folder under **"Freenove_ESP32_S3_Display\NonTouch\Sketches"** and double-click "**Sketch_06.1_SD_Test.ino**".

Sketch_06.1_SD_Test

The following is the program code:

```
/*
* @ File:   Sketch_06.1_SDMMC_Test.ino
* @ Author: [Zhentao Lin]
* @ Date:   [2025-08-19]
*/

#include "driver_sdmmc.h"

#define SD_MMC_CMD 40  // Please do not modify it.
#define SD_MMC_CLK 38  // Please do not modify it.
#define SD_MMC_D0  39  // Please do not modify it.
#define SD_MMC_D1  41  // Please do not modify it.
#define SD_MMC_D2  48  // Please do not modify it.
#define SD_MMC_D3  47  // Please do not modify it.

void setup() {
    // Initialize serial communication at 115200 bits per second
    Serial.begin(115200);
    delay(3000);

    // Initialize the SDMMC interface with the specified pins
    sdmmc_init(SD_MMC_CLK, SD_MMC_CMD, SD_MMC_D0, SD_MMC_D1, SD_MMC_D2, SD_MMC_D3);

    // List files in the root directory and print them
    std::list<File_Entry> fileList = list_dir("/", 0);
    print_file_list(fileList);

    // Create a new directory and list files again
    create_dir("/mydir");
    fileList = list_dir("/", 0);
    print_file_list(fileList);

    // Remove the directory and list files again
    remove_dir("/mydir");
    fileList = list_dir("/", 2);
    print_file_list(fileList);
```

```
37
38      // Write text to a new file
39      const char* text = "Hello ";
40      write_file("/foo.txt", (const uint8_t*)text, strlen(text));
41
42      // Append text to the existing file
43      text = "World!\n";
44      append_file("/foo.txt", (const uint8_t*)text, strlen(text));
45
46      // Read the file content into a buffer and print it
47      uint8_t buffer[100];
48      read_file("/foo.txt", buffer, sizeof(buffer));
49      Serial.write(buffer, strlen((char*)buffer));
50
51      // Rename the file and read the new file content
52      rename_file("/foo.txt", "/hello.txt");
53      read_file("/hello.txt", buffer, sizeof(buffer));
54      Serial.write(buffer, strlen((char*)buffer));
55
56      // Print SD card information
57      Serial.println("\r\nSD card info:");
58      Serial.printf("Total space: %lluMB\r\n", SD_MMC.totalBytes() / (1024 * 1024));
59      Serial.printf("Used space: %lluMB\r\n", SD_MMC.usedBytes() / (1024 * 1024));
60  }
61
62  void loop() {
63      // Delay for 10 seconds before the next iteration
64      delay(10000);
65  }
```

Code Explanation

Include necessary header files.

```
7   #include "driver_sdmmc.h"
```

Define SD card pins. In this example, we read and write the SD card via SPI.

```
9   #define SD_MMC_CMD 40  // Please do not modify it.
10  #define SD_MMC_CLK 38  // Please do not modify it.
11  #define SD_MMC_D0  39  // Please do not modify it.
12  #define SD_MMC_D1  41  // Please do not modify it.
13  #define SD_MMC_D2  48  // Please do not modify it.
14  #define SD_MMC_D3  47  // Please do not modify it.
```

Set the baud rate to 115200.

```
18  Serial.begin(115200);
```

Initialize the SD card

```
19  sdmmc_init(SD_MMC_CLK, SD_MMC_CMD, SD_MMC_D0, SD_MMC_D1, SD_MMC_D2, SD_MMC_D3);
```

List files in the root directory and print them

```
25  std::list<File_Entry> fileList = list_dir("/", 0);
26  print_file_list(fileList);
```

Create a directory named “mydir” and list all files under the root directory.

```
29  create_dir("/mydir");
30  fileList = list_dir("/", 0);
```

Delete the “mydir” directory and list all files under the root directory.

```
34  remove_dir("/mydir");
35  fileList = list_dir("/", 2);
```

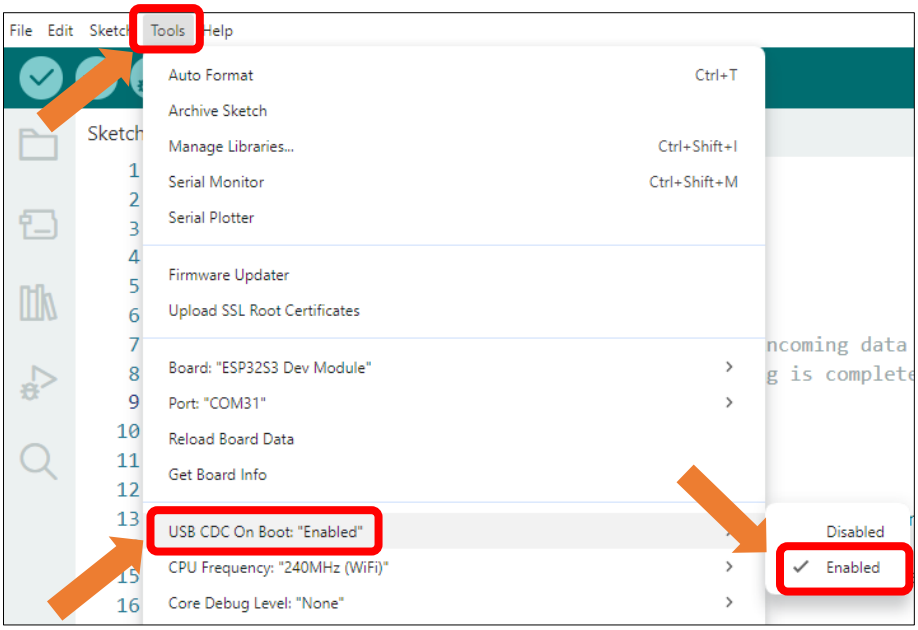
Write "Hello" in the hello.txt file, append "World!" to it, then read and print the file contents.

```
38  // Write text to a new file
39  const char* text = "Hello ";
40  write_file("/foo.txt", (const uint8_t*)text, strlen(text));
41
42  // Append text to the existing file
43  text = "World!\n";
44  append_file("/foo.txt", (const uint8_t*)text, strlen(text));
45
46  // Read the file content into a buffer and print it
47  uint8_t buffer[100];
48  read_file("/foo.txt", buffer, sizeof(buffer));
49  Serial.write(buffer, strlen((char*)buffer));
```

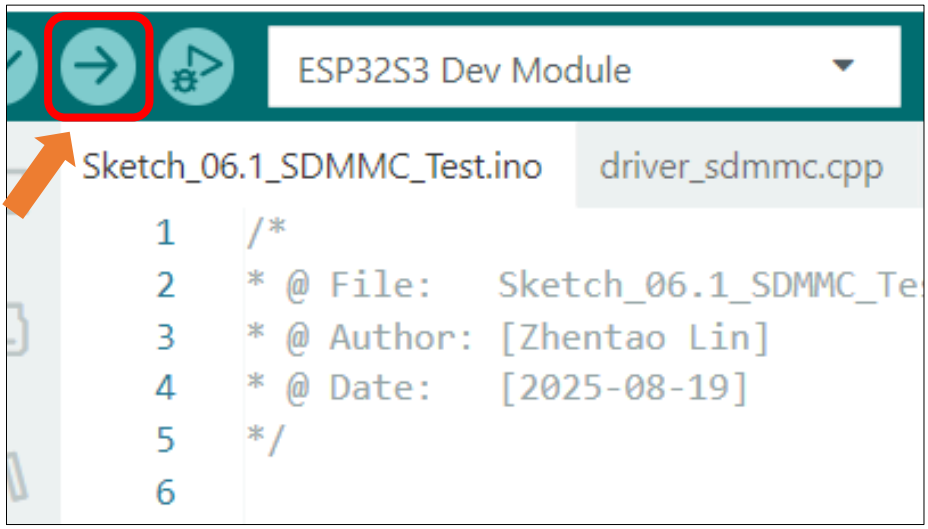
Rename foo.txt to hello.txt and read the file.

```
51  // Rename the file and read the new file content
52  rename_file("/foo.txt", "/hello.txt");
53  read_file("/hello.txt", buffer, sizeof(buffer));
54  Serial.write(buffer, strlen((char*)buffer));
```

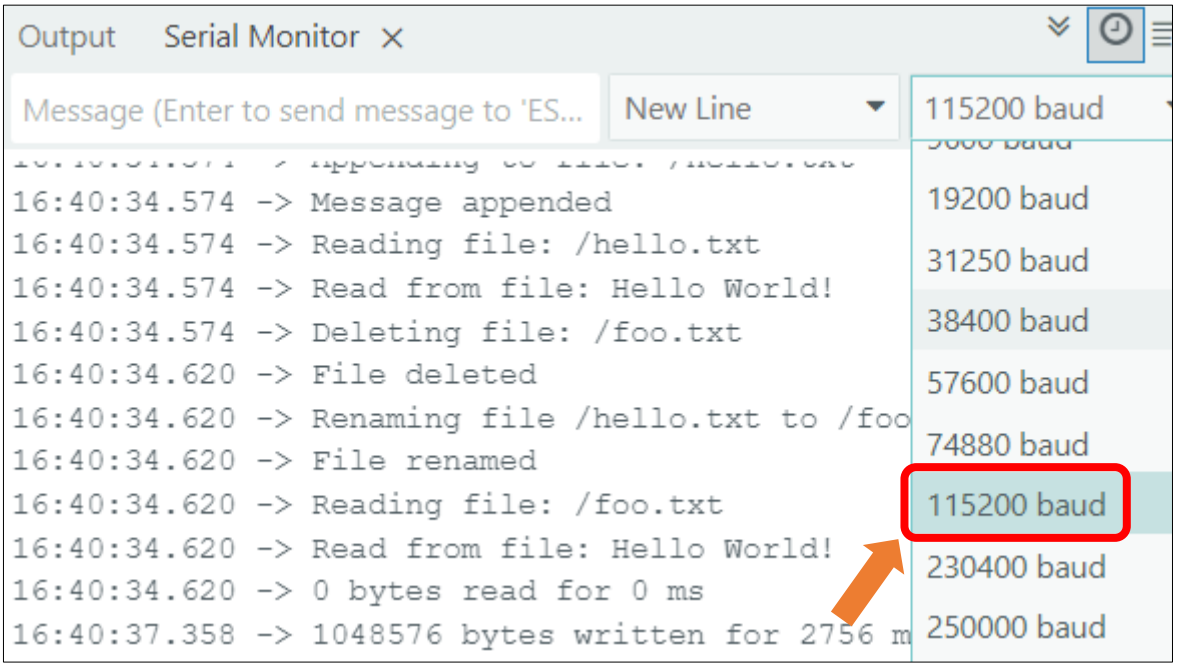
Enable the "USB CDC On Boot" feature.



Click “**Upload**” to upload the code to Freenove_ESP32_S3_Display.



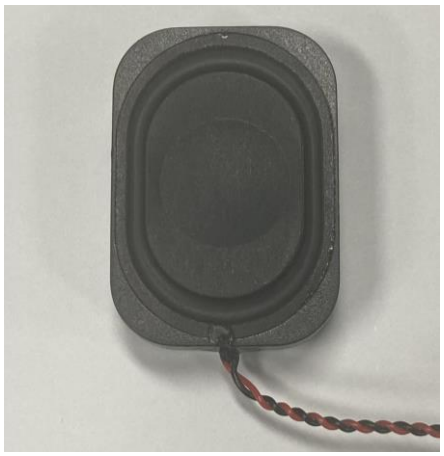
Upon the code runs, open the serial and set the baud rate to 115200.



Chapter 7 Music

Project 7.1 Music

Component List

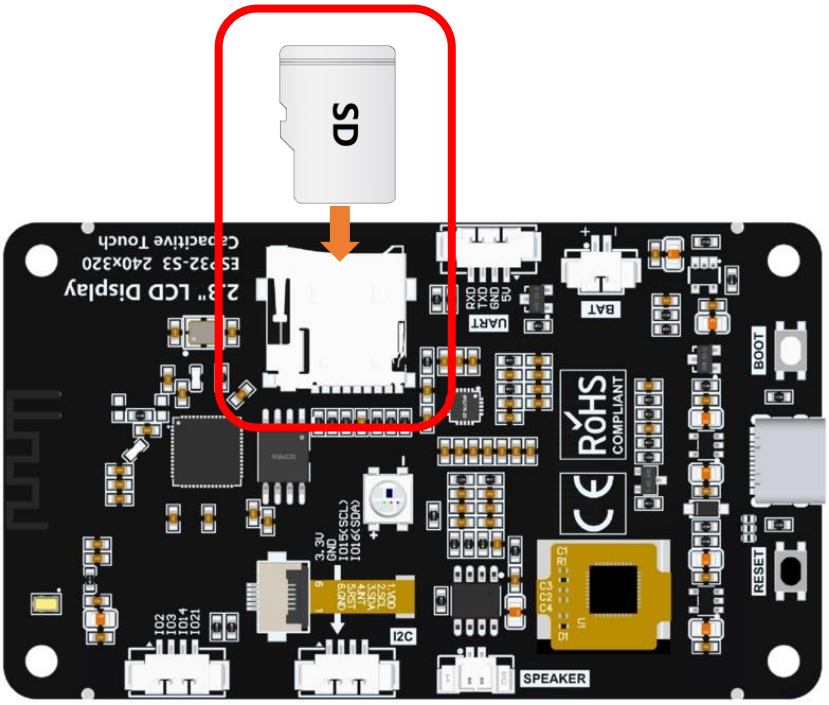
Freenove ESP32 S3 Display x 1 	USB cable x1 
Speaker x1 	

Please note that this kit does not include SD card and card reader, please buy them by yourself.

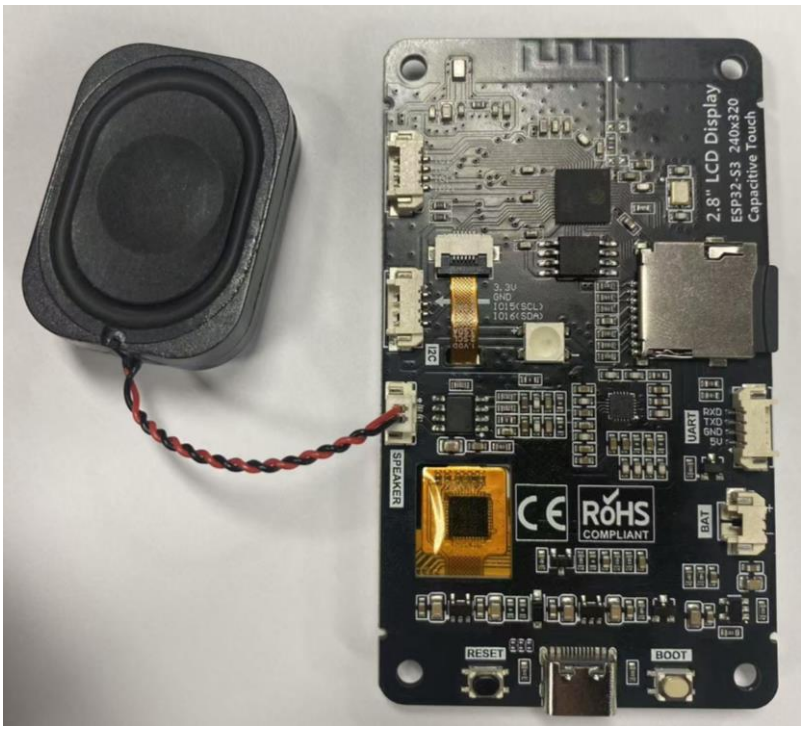
Circuit

Before connecting the USB cable, insert the SD card into the SD card slot on the back of the ESP32-S3.

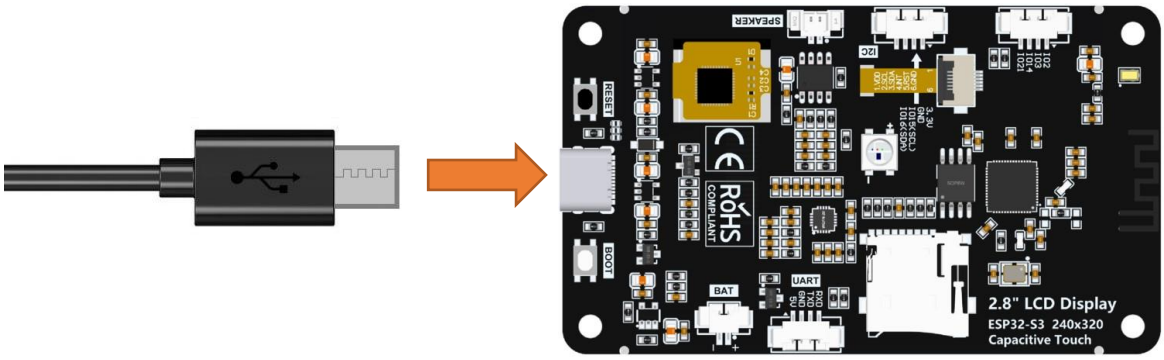
Please note that this kit does not include SD card and card reader; please buy them yourself.



Connect speaker



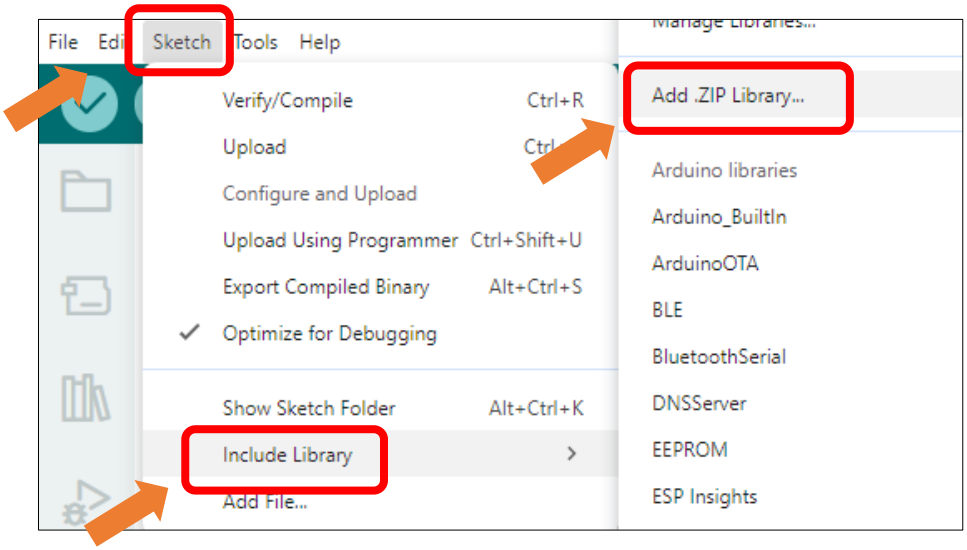
Connect Freenove ESP32-S3 to the computer using the USB cable.



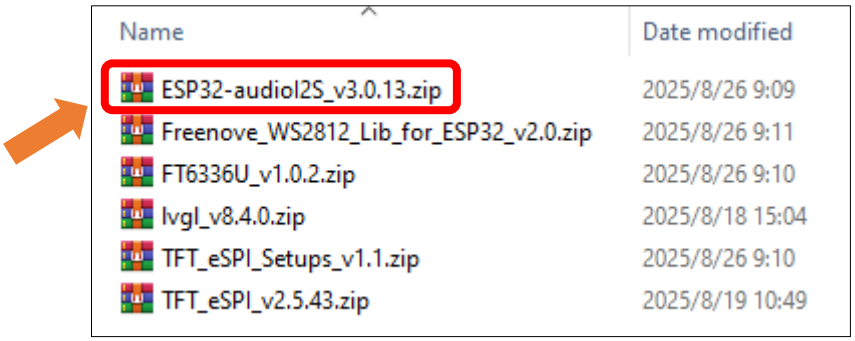
Sketch

Install the needed libraries.

Click **Sketch** -> **Include Library** -> **Add .ZIP Library...**



Select **ESP32-audioI2S_v3.0.13.zip**



Next, we download the code to Freenove_ESP32_S3_Display to test. Open"**Sketch_07.1_Music**" folder under **"Freenove_ESP32_S3_Display\NonTouch\Sketches"** and double-click "**Sketch_07.1_Music.ino**".

Sketch_07.1_Music

The following is the program code:

```
1  #include <Arduino.h>
2  #include "FS.h"
```

```
3  #include "SD_MMC.h"
4  #include "SPI.h"
5  #include "es8311.h"
6  #include "Audio.h"
7  #include "Wire.h"
8  #include "ESP_I2S.h"
9  #include "demo_music.h"
10
11 //ESP32-S3 IO Pin define
12 #define SD_SCK 38
13 #define SD_CMD 40
14 #define SD_D0 39
15 #define SD_D1 41
16 #define SD_D2 48
17 #define SD_D3 47
18
19 //I2S IO Pin define
20 #define I2S_MCK 4
21 #define I2S_BCK 5
22 #define I2S_DINT 6
23 #define I2S_DOUT 8
24 #define I2S_WS 7
25 #define AP_ENABLE 1
26 #define I2C_SCL 15        /*!< GPIO number used for I2C master clock */
27 #define I2C_SDA 16        /*!< GPIO number used for I2C master data  */
28 #define I2C_SPEED 400000  /*!< I2C master clock frequency */
29
30 Audio audio;
31 I2SClass es8311_i2s;
32
33 void driver_es8311_init(void) {
34    pinMode(AP_ENABLE, OUTPUT);
35    digitalWrite(AP_ENABLE, LOW);
36
37    Wire.begin(I2C_SDA, I2C_SCL, I2C_SPEED);
38
39    es8311_i2s.setPins(I2S_BCK, I2S_WS, I2S_DOUT, I2S_DINT, I2S_MCK);
40    if (!es8311_i2s.begin(I2S_MODE_STD, 44100, I2S_DATA_BIT_WIDTH_16BIT, I2S_SLOT_MODE_STEREO,
41 I2S_STD_SLOT_LEFT)) {
42      Serial.println("Failed to initialize I2S bus!");
43    }
44 }
45
46 void setup() {
```

```
47      Serial.begin(115200);
48
49      //SD card init
50      if (!SD_MMC.setPins(SD_SCK, SD_CMD, SD_D0, SD_D1, SD_D2, SD_D3)) {
51        Serial.println("Pin change failed!");
52        return;
53      }
54      while (!SD_MMC.begin()) {
55        Serial.println("SD card does not exist,please insert SD card.");
56        delay(100);
57      }
58
59      driver_es8311_init();
60      if (es8311_codec_init() != ESP_OK) {
61        Serial.println("ES8311 init failed!");
62        return;
63      }
64      //audio init
65      audio.setPinout(I2S_BCK, I2S_WS, I2S_DOUT, I2S_MCK);
66      audio.setVolume(10);
67      if (!demo_music()) {
68        return;
69      }
70
71      //play music
72      demo_music_play(0);
73  }
74
75  void loop() {
76      audio.loop();
77  }
```

Code Explanation

Include necessary header files.

```
1   #include <Arduino.h>
2   #include "FS.h"
3   #include "SD_MMC.h"
4   #include "SPI.h"
5   #include "es8311.h"
6   #include "Audio.h"
7   #include "Wire.h"
8   #include "ESP_I2S.h"
```

Define the pins.

```
10  //ESP32-S3 IO Pin define
11  #define SD_SCK 38
```

```
#define SD_CMD 40
#define SD_D0 39
#define SD_D1 41
#define SD_D2 48
#define SD_D3 47

//I2S IO Pin define
#define I2S_MCK 4
#define I2S_BCK 5
#define I2S_DINT 6
#define I2S_DOUT 8
#define I2S_WS 7
#define AP_ENABLE 1
#define I2C_SCL 15
#define I2C_SDA 16
#define I2C_SPEED 400000
```

Declare an I2S object

```
Audio audio;
I2SClass es8311_i2s;
```

Set the baud rate to 115200

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

SD card init

```
if (!SD_MMC.setPins(SD_SCK, SD_CMD, SD_D0, SD_D1, SD_D2, SD_D3)) {
    Serial.println("Pin change failed!");
    return;
}
while (!SD_MMC.begin()) {
    Serial.println("SD card does not exist,please insert SD card.");
    delay(100);
}

driver_es8311_init();
if (es8311_codec_init() != ESP_OK) {
    Serial.println("ES8311 init failed!");
    return;
}
```

Read audio data and play it.

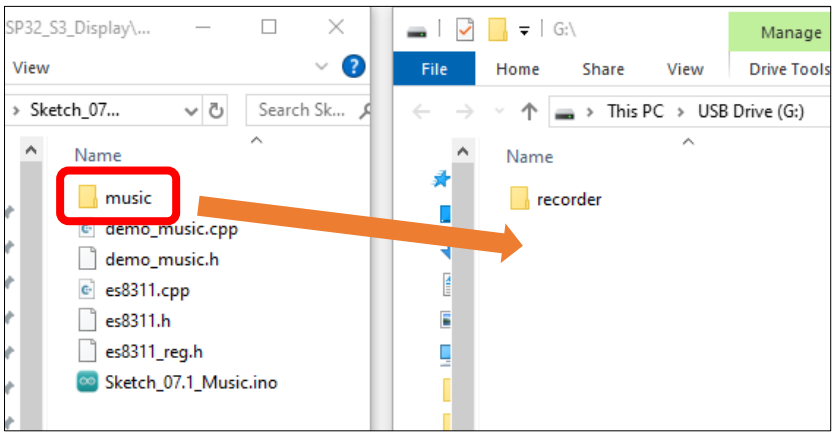
```
audio.setPinout(I2S_BCK, I2S_WS, I2S_DOUT, I2S_MCK);
audio.setVolume(10);
if (!demo_music()) {
    return;
}
```

play music

```
demo_music_play(1);
```

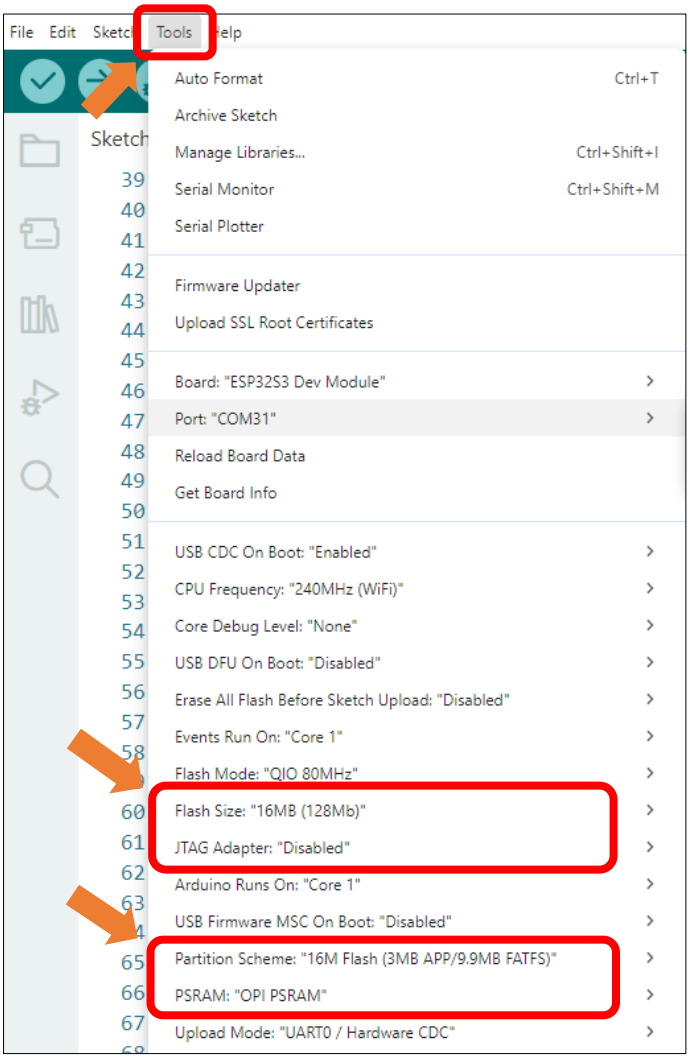
This product does not include SD card, and SD card reader, please buy them by yourself. For more information, please refer to SD card sections.

Before uploading the code, copy the music to the root directory of the SD card with the SD card reader.

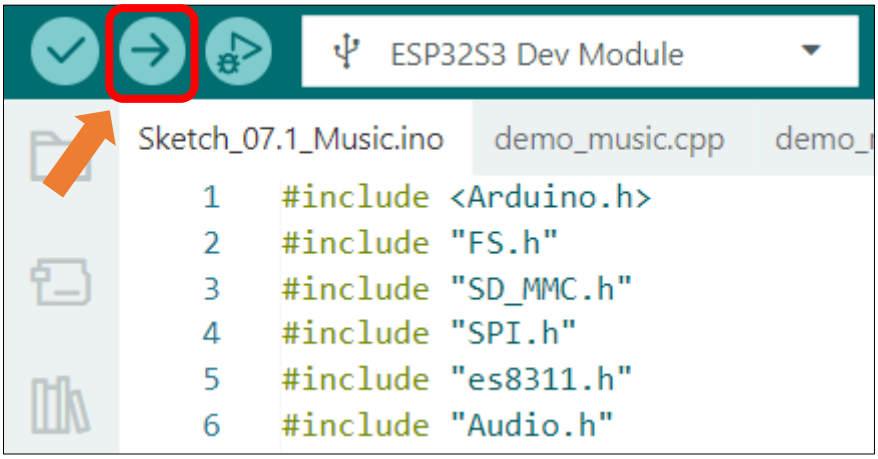


It is necessary to change the settings in Arduino IDE before clicking the Uploading button, as shown below.

Caution: Incorrect settings will result in compilation error or uploading failure. To achieve desired result, please configure exactly the same as below.



Click "**Upload**" to upload the code to Freenove ESP32 S3 Display.



The speaker plays the music in the SD card.

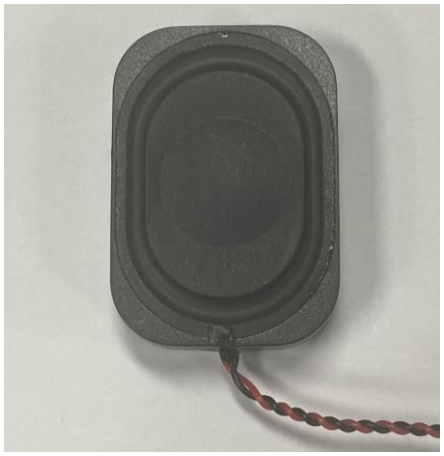
Project 7.2 Echo

Component List

Freenove ESP32 S3 Display x 1

USB cable x1

Speakerx1



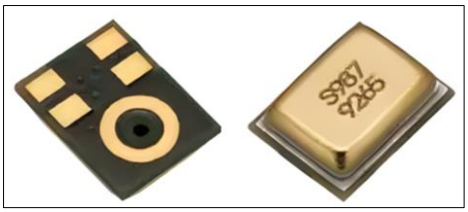
Please note that this kit does not include SD card and card reader, please buy them by yourself.

Component knowledge

MEMS-MIC

A MEMS Microphone (Micro-Electro-Mechanical Systems Microphone) is a miniature microphone manufactured using MEMS technology. It integrates mechanical sensing elements and electronic circuits on the same chip to achieve sound signal acquisition and conversion. Its working principle primarily involves a tiny vibrating diaphragm that detects sound pressure changes, then converts these mechanical vibrations into electrical signals, enabling sound capture and transmission.

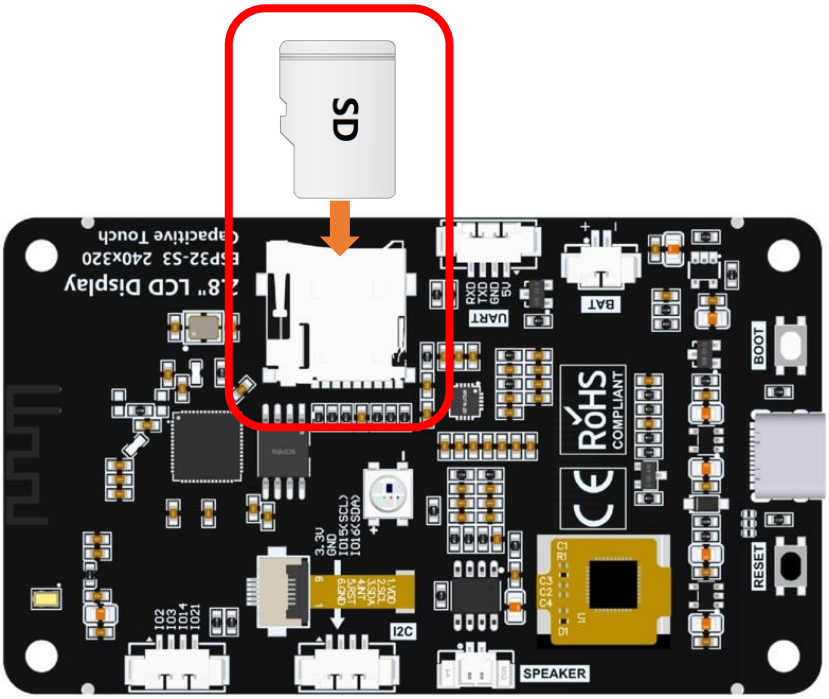
MEMS microphones are characterized by their compact size, high sensitivity, excellent stability, and ease of mass production. They are widely used in electronic devices such as smartphones, earphones, and smart speakers. Compared to traditional microphones, MEMS microphones better meet the dual demands of modern electronic products for both miniaturization and performance.



Circuit

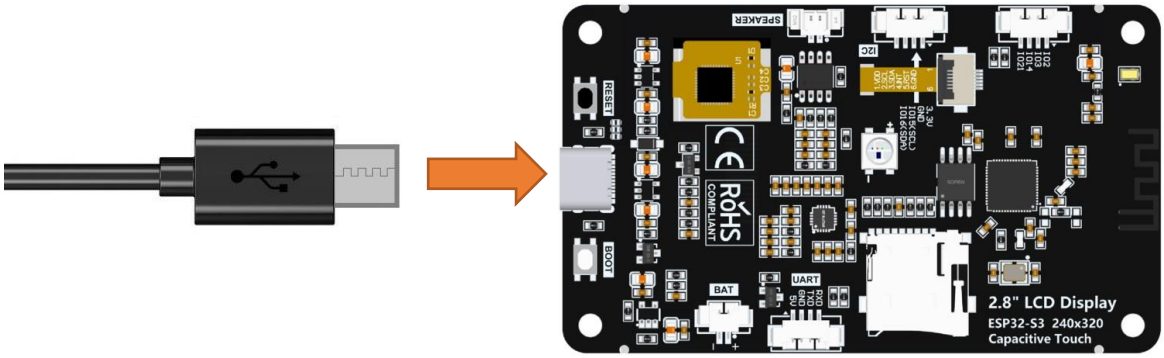
Before connecting the USB cable, insert the SD card into the SD card slot on the back of the ESP32-S3.

Please note that this kit does not include SD card and card reader; please buy them yourself.



Connect speaker

Connect Freenove ESP32 S3 Display to the computer using the USB cable.



Sketch

Sketch_07.2_Echo

The following is the program code:

```
#include <Arduino.h>
#include "FS.h"
#include "SD_MMC.h"
#include "SPI.h"
#include "es8311.h"
#include "Audio.h"
#include "Wire.h"
#include "ESP_I2S.h"

//ESP32-S3 IO Pin define
#define SD_SCK 38
#define SD_CMD 40
#define SD_D0 39
```

```
#define SD_D1 41
#define SD_D2 48
#define SD_D3 47

//I2S IO Pin define
#define I2S_MCK 4
#define I2S_BCK 5
#define I2S_DINT 6
#define I2S_DOUT 8
#define I2S_WS 7
#define AP_ENABLE 1
#define I2C_SCL 15
#define I2C_SDA 16
#define I2C_SPEED 400000

I2SClass es8311_i2s;

void driver_es8311_init(void) {
   pinMode(AP_ENABLE, OUTPUT);
   digitalWrite(AP_ENABLE, LOW);

   Wire.begin(I2C_SDA, I2C_SCL, I2C_SPEED);

   es8311_i2s.setPins(I2S_BCK, I2S_WS, I2S_DOUT, I2S_DINT, I2S_MCK);
   if (!es8311_i2s.begin(I2S_MODE_STD, 44100, I2S_DATA_BIT_WIDTH_16BIT, I2S_SLOT_MODE_MONO,
I2S_STD_SLOT_LEFT)) {
     Serial.println("Failed to initialize I2S bus!");
   }
}

void setup()
{
   Serial.begin(115200);
   while (!Serial) {
     delay(10);
   }

   Serial.println("Start initializing the audio device...");
   driver_es8311_init();
   if (es8311_codec_init() != ESP_OK) {
     Serial.println("ES8311 init failed!");
     return;
   }
   delay(3000);
```

```
58      Serial.println("Initialization completed.");
59  }
60
61  void loop()
62  {
63      uint8_t *wav_buffer;
64      size_t wav_size;
65      // Record 5 seconds of audio data
66      Serial.println("Start recording for 5 seconds...");
67      wav_buffer = es8311_i2s.recordWAV(5, &wav_size);
68      Serial.println("Recording completed.");
69      delay(1000);
70
71      Serial.println("Start playing the recording...");
72      es8311_i2s.playWAV(wav_buffer, wav_size);
73      Serial.println("Playback has been completed.");
74      free(wav_buffer);
75      delay(1000);
76  }
```

Code Explanation

Include necessary header files.

```
1  #include <Arduino.h>
2  #include "FS.h"
3  #include "SD_MMC.h"
4  #include "SPI.h"
5  #include "es8311.h"
6  #include "Audio.h"
7  #include "Wire.h"
8  #include "ESP_I2S.h"
```

Define the pins.

```
10  //ESP32-S3 IO Pin define
11  #define SD_SCK 38
12  #define SD_CMD 40
13  #define SD_D0 39
14  #define SD_D1 41
15  #define SD_D2 48
16  #define SD_D3 47
17
18  //I2S IO Pin define
19  #define I2S_MCK 4
20  #define I2S_BCK 5
21  #define I2S_DINT 6
22  #define I2S_DOUT 8
23  #define I2S_WS 7
```

```
24 #define AP_ENABLE 1
25 #define I2C_SCL 15
26 #define I2C_SDA 16
27 #define I2C_SPEED 400000
```

Declare an I2S object

```
19 I2SClass es8311_i2s;
```

Set the baud rate to 115200

```
36 Serial.begin(115200);
```

Initialize the audio device.

```
51 Serial.println("Start initializing the audio device...");
52 driver_es8311_init();
53 if (es8311_codec_init() != ESP_OK) {
54     Serial.println("ES8311 init failed!");
55     return;
56 }
```

Implement audio recording and playback functionality

```
61 void loop()
62 {
63     uint8_t *wav_buffer;
64     size_t wav_size;
65     // Record 5 seconds of audio data
66     Serial.println("Start recording for 5 seconds...");
67     wav_buffer = es8311_i2s.recordWAV(5, &wav_size);
68     Serial.println("Recording completed.");
69     delay(1000);
70
71     Serial.println("Start playing the recording...");
72     es8311_i2s.playWAV(wav_buffer, wav_size);
73     Serial.println("Playback has been completed.");
74     free(wav_buffer);
75     delay(1000);
76 }
```

Chapter 8 BLE

Project 8.1 BLE USART

Component List

Freenove ESP32 S3 Display x 1 	USB cable x1 

Component Knowledge

BLE (Bluetooth Low Energy)

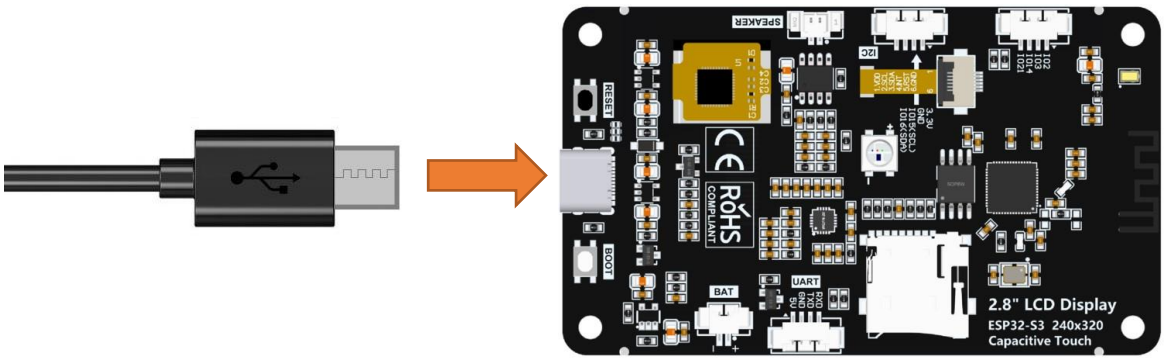
Low Energy Bluetooth (BLE) is a new feature introduced in the Bluetooth 4.0 specification, specifically designed for low-power devices and suitable for applications involving intermittent transmission of small amounts of data.

The BLE architecture follows a client-server model and consists of the following key components:
GATT (Generic Attribute Protocol): The foundational protocol for BLE communication, defining a hierarchical data structure of services and characteristics.
Service: A collection of data that performs a specific function or feature, containing one or more characteristics.
Characteristic: A specific data point within a service, consisting of a value and descriptors.
UUID: A 128-bit universally unique identifier used to distinguish services and characteristics.

BLE device can function simultaneously as a Peripheral (providing services) and a Central (connecting to peripherals). In this section, we will learn how to configure the Freenove_ESP32_S3_Display as a peripheral device to provide services.

Circuit

Connect Freenove ESP32-S3 to the computer using the USB cable.



Sketch

Next, we download the code to Freenove_ESP32_S3_Display to test. Open"**Sketch_08.1_BLE_USART**" folder under **"Freenove_ESP32_S3_Display\NonTouch\Sketches"** and double-click "**Sketch_08.1_BLE_USART.ino**".

Sketch_08.1_BLE_USART

The following is the program code:

```
/*
* @ File:   Sketch_08.1_BLE_USART.ino
* @ Author: [Zhentao Lin]

* @ Date:   [2025-06-14]
*/

#include "BLEDevice.h"
#include "BLEServer.h"
#include "BLEUtils.h"
#include "BLE2902.h"
#include "String.h"

BLECharacteristic *pCharacteristic;
bool deviceConnected = false;
uint8_t txValue = 0;
long lastMsg = 0;
String rxload="Test\n";

#define SERVICE_UUID           "6E400001-B5A3-F393-E0A9-E50E24DCCA9E"
#define CHARACTERISTIC_UUID_RX "6E400002-B5A3-F393-E0A9-E50E24DCCA9E"
#define CHARACTERISTIC_UUID_TX "6E400003-B5A3-F393-E0A9-E50E24DCCA9E"
```

```
23
24  class MyServerCallbacks: public BLEServerCallbacks {
25        void onConnect(BLEServer* pServer) {
26          deviceConnected = true;
27        };
28        void onDisconnect(BLEServer* pServer) {
29          deviceConnected = false;
30        }
31  };
32
33  class MyCallbacks: public BLECharacteristicCallbacks {
34        void onWrite(BLECharacteristic *pCharacteristic) {
35          String rxValue = pCharacteristic->getValue();
36          if (rxValue.length() > 0) {
37            rxload="";
38            for (int i = 0; i < rxValue.length(); i++){
39              rxload +=(char)rxValue[i];
40            }
41          }
42        }
43  };
44
45  void setupBLE(String BLEName){
46     const char *ble_name=BLEName.c_str();
47     BLEDevice::init(ble_name);
48     BLEServer *pServer = BLEDevice::createServer();
49     pServer->setCallbacks(new MyServerCallbacks());
50     BLEService *pService = pServer->createService(SERVICE_UUID);
51     pCharacteristic=
52  pService->createCharacteristic(CHARACTERISTIC_UUID_TX,BLECharacteristic::PROPERTY_NOTIFY);
53     pCharacteristic->addDescriptor(new BLE2902());
54     BLECharacteristic *pCharacteristic =
55  pService->createCharacteristic(CHARACTERISTIC_UUID_RX,BLECharacteristic::PROPERTY_WRITE);
56     pCharacteristic->setCallbacks(new MyCallbacks());
57     pService->start();
58     pServer->getAdvertising()->start();
59     Serial.println("Waiting a client connection to notify...");
60  }
61
62  void setup() {
63     Serial.begin(115200);
64     setupBLE("ESP32S3_BLE");
65  }
66
```

```
67  void loop() {
68     long now = millis();
69     if (now - lastMsg > 100) {
70       if (deviceConnected&&rxload.length()>0) {
71           Serial.println(rxload);
72           rxload="";
73       }
74       if(Serial.available()>0){
75           String str=Serial.readString();
76           const char *newValue=str.c_str();
77           pCharacteristic->setValue(newValue);
78           pCharacteristic->notify();
79       }
80       lastMsg = now;
81     }
82  }
```

Code Explanation

Include the necessary header libraries.

```
6   #include "BLEDevice.h"
7   #include "BLEServer.h"
8   #include "BLEUtils.h"
9   #include "BLE2902.h"
10  #include "String.h"
```

Define service UUID and characteristic UUID.

```
19  #define SERVICE_UUID           "6E400001-B5A3-F393-E0A9-E50E24DCCA9E"
20  #define CHARACTERISTIC_UUID_RX "6E400002-B5A3-F393-E0A9-E50E24DCCA9E"
21  #define CHARACTERISTIC_UUID_TX "6E400003-B5A3-F393-E0A9-E50E24DCCA9E"
```

The MyServerCallbacks class handles device connection and disconnection events and updates the deviceConnected status.

```
23  class MyServerCallbacks: public BLEServerCallbacks {
24       void onConnect(BLEServer* pServer) {
25         deviceConnected = true;
26       };
27       void onDisconnect(BLEServer* pServer) {
28         deviceConnected = false;
29       }
30  };
```

The MyServerCallbacks class handles the received data and save them to the rxload string.

```
32  class MyCallbacks: public BLECharacteristicCallbacks {
33       void onWrite(BLECharacteristic *pCharacteristic) {
34         String rxValue = pCharacteristic->getValue();
35         if (rxValue.length() > 0) {
36           rxload="";
37           for (int i = 0; i < rxValue.length(); i++){
```

```
            rxload +=(char)rxValue[i];
          }
        }
      }
};
```

Set the baud rate to 115200.

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

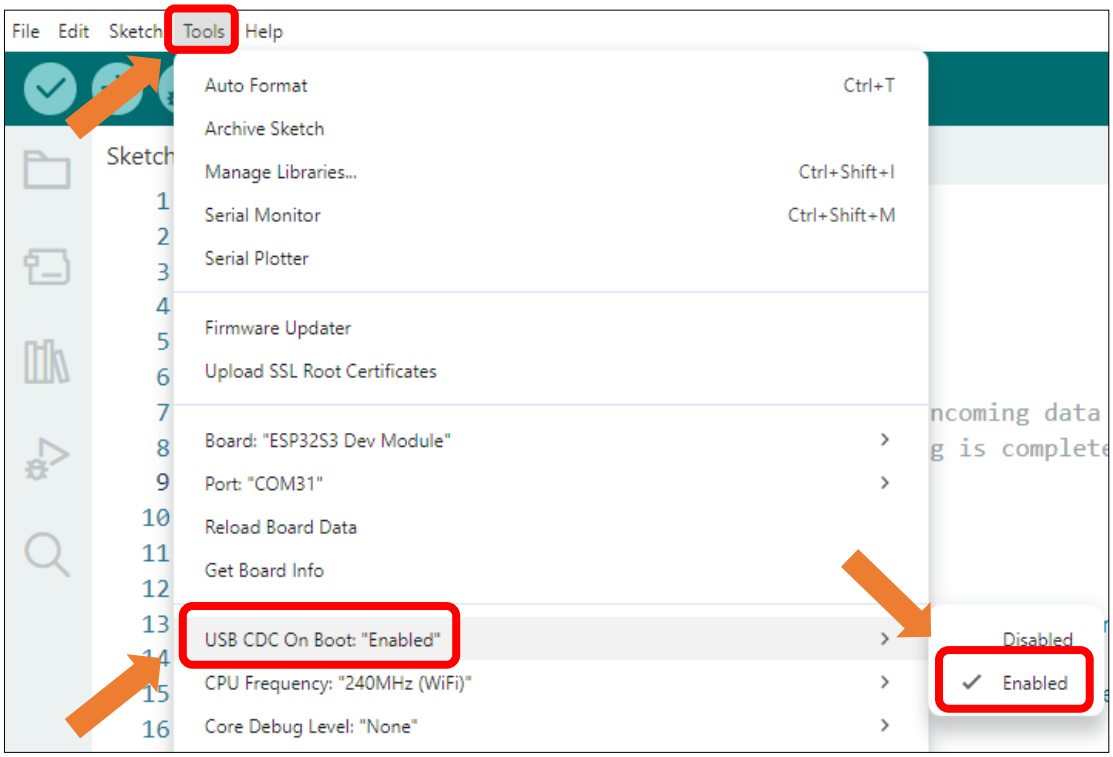
BLE Device Initialization, Service Creation, and Characteristic Setup

```
setupBLE("ESP32S3_BLE");
```

The Loop function check the connection status and serial data every 100 milliseconds.

```
void loop() {
  long now = millis();
  if (now - lastMsg > 100) {
    if (deviceConnected&&rxload.length()>0) {
        Serial.println(rxload);
        rxload="";
    }
    if(Serial.available()>0){
        String str=Serial.readString();
        const char *newValue=str.c_str();
        pCharacteristic->setValue(newValue);
        pCharacteristic->notify();
    }
    lastMsg = now;
  }
}
```

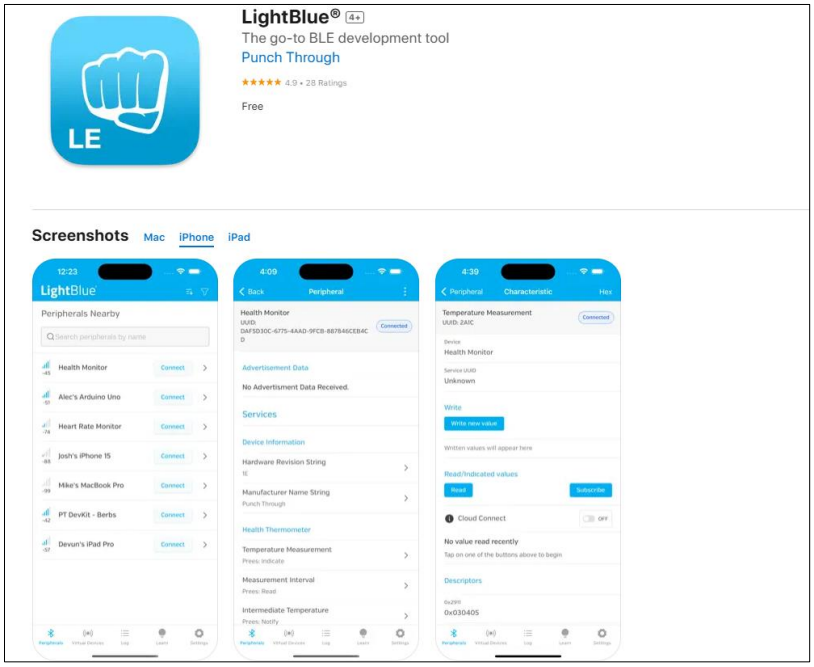
Enable the "USB CDC On Boot" feature.



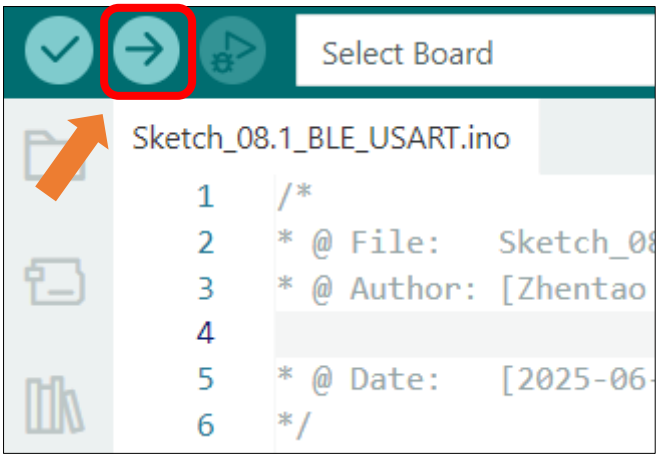
LightBlue

If you do not have this software installed on your phone, you can refer to this link:

https://apps.apple.com/us/app/lightblue/id557428110

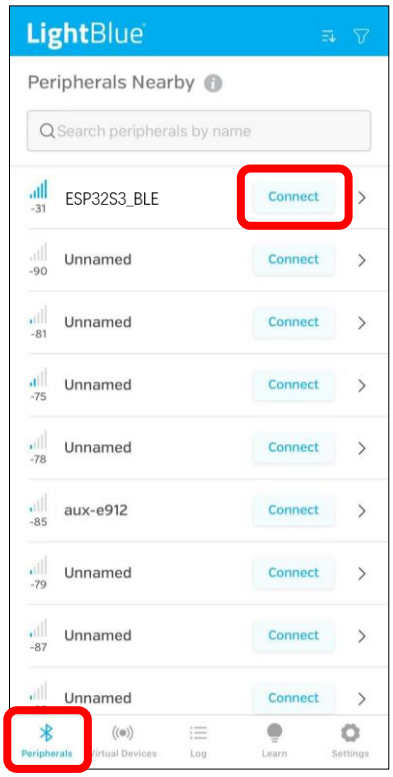


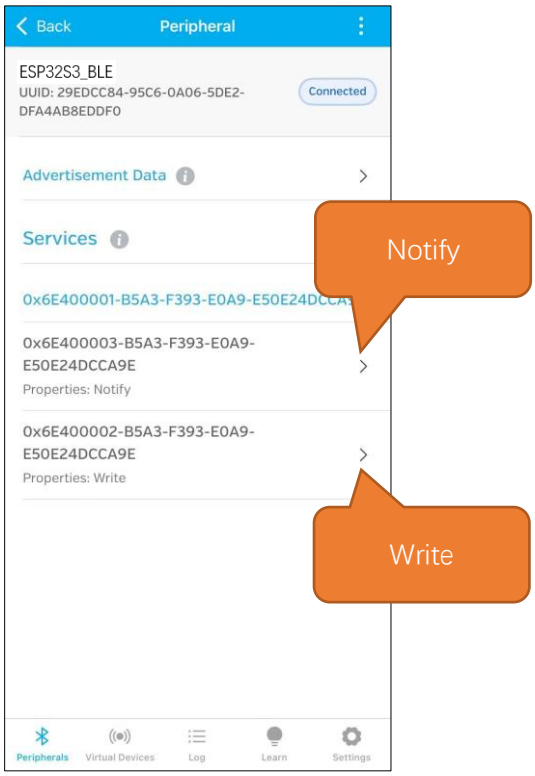
Click "**Upload**" to upload the code to Freenove ESP32 S3 Display.



Turn ON Bluetooth on your phone, and open the LightBlue APP.

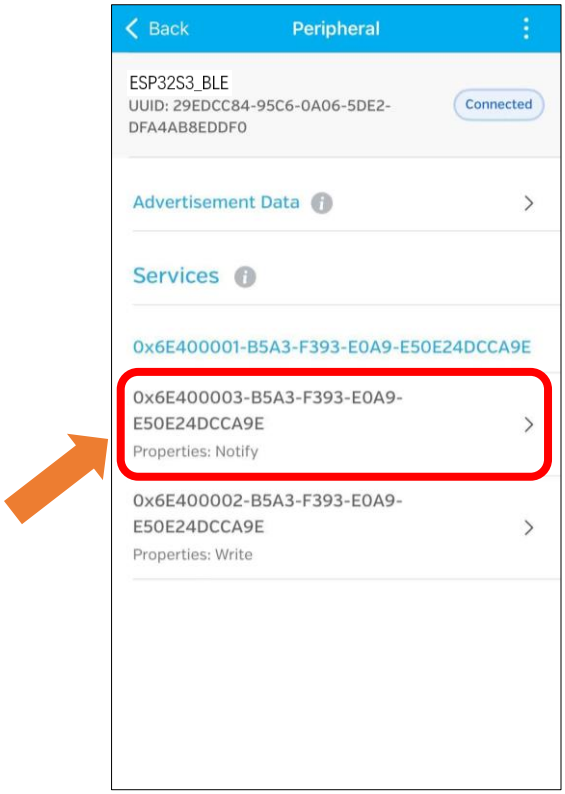
In the Scan page, swipe down to refresh the name of Bluetooth that the phone searches for. Click the Connection button of ESP32S3_BLE



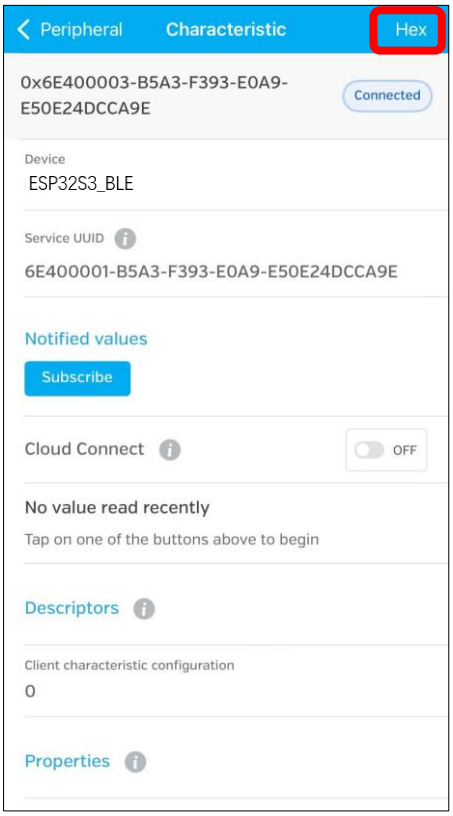


Receiving Data

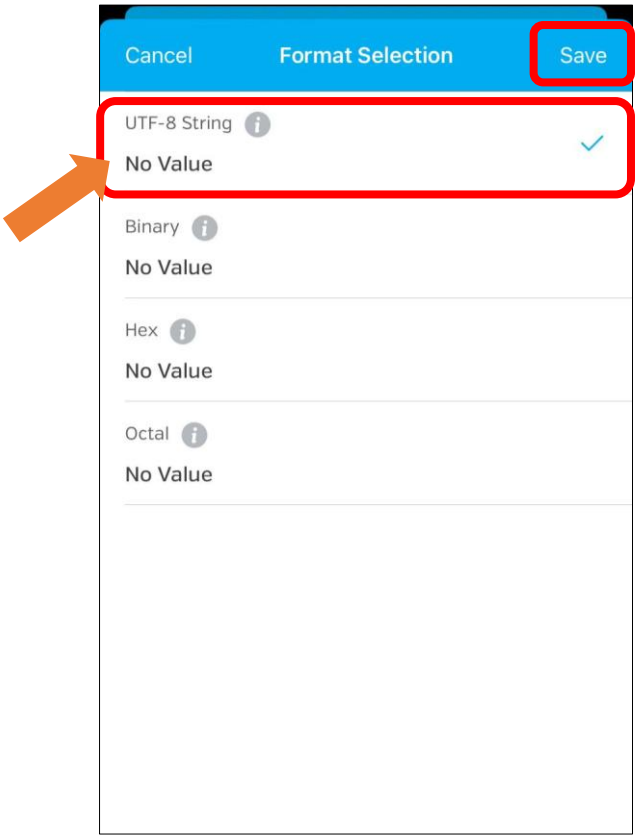
Click Notify


Back
Peripheral
ESP32S3_BLE
UUID: 29EDCC84-95C6-0A06-5DE2-DFA4AB8EDDF0
Connected
Advertisement Data
Services
0x6E400001-B5A3-F393-E0A9-E50E24DCCA9E
0x6E400003-B5A3-F393-E0A9-E50E24DCCA9E
Properties: Notify
0x6E400002-B5A3-F393-E0A9-E50E24DCCA9E
Properties: Write

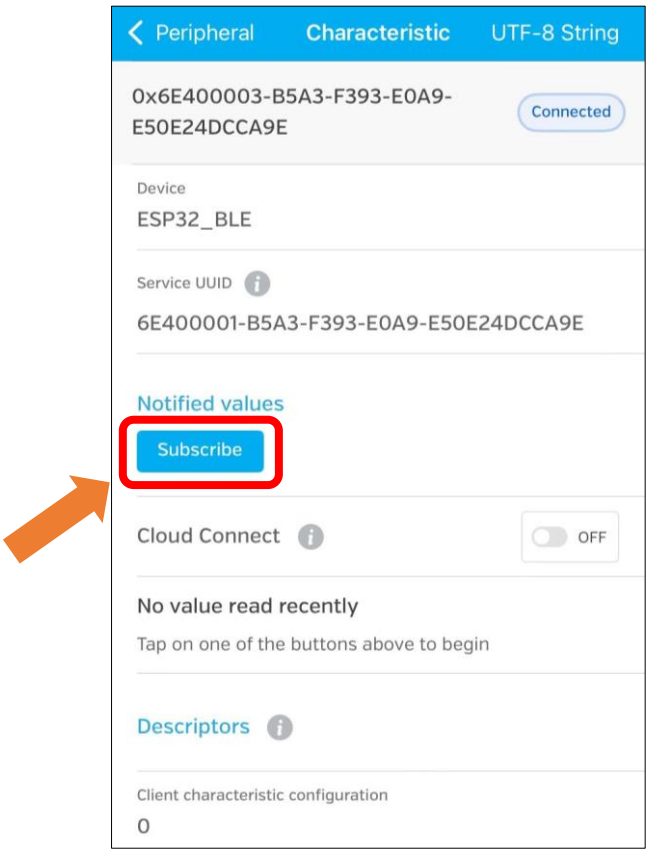
click the top right corner to change the string type.



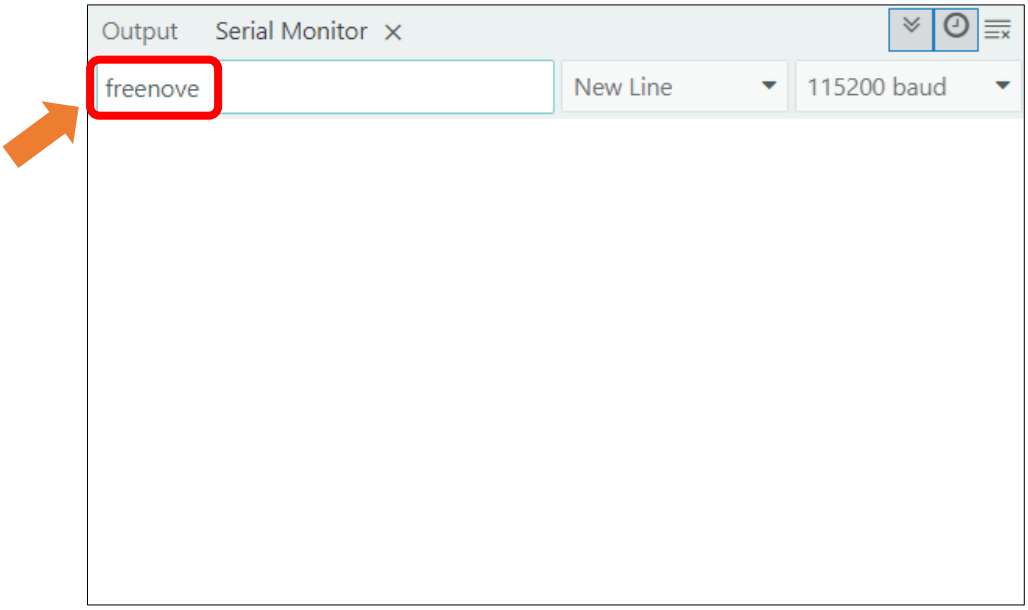
Select UTF-8 String, and click “Save”.



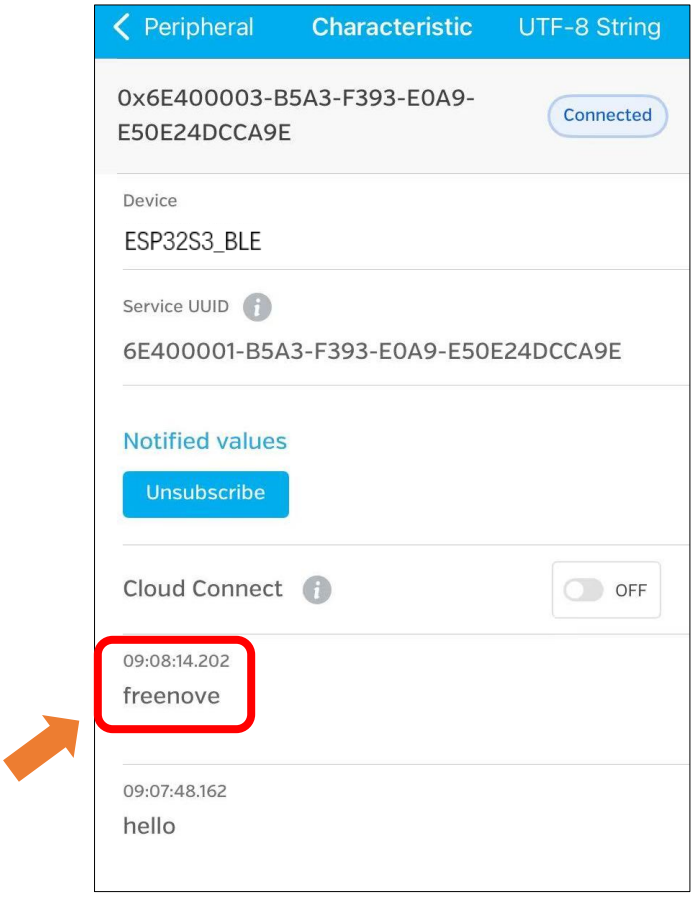
Click Subscribe



Send data on the serial monitor.

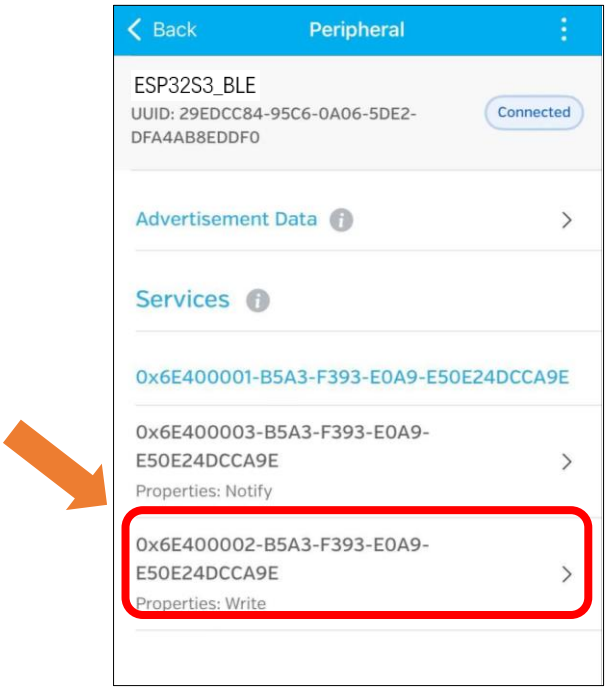


Data will be received on the LightBlue app.

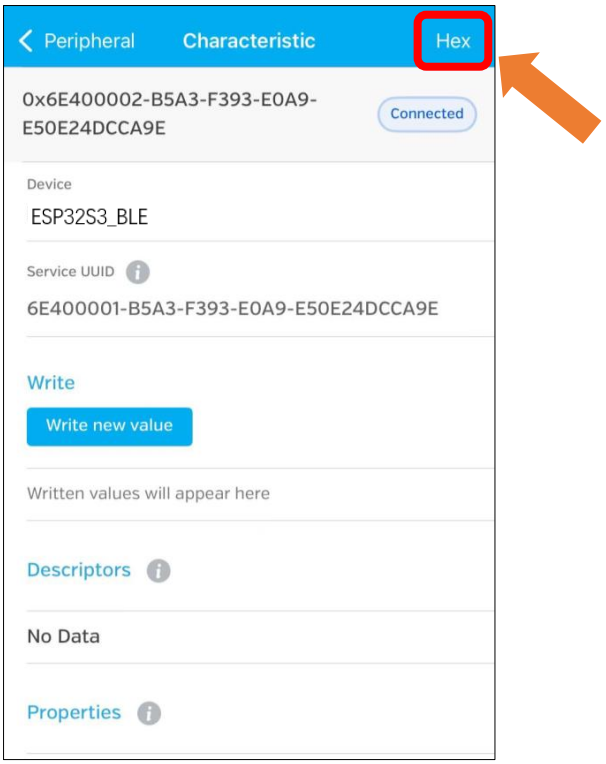


Sending Data

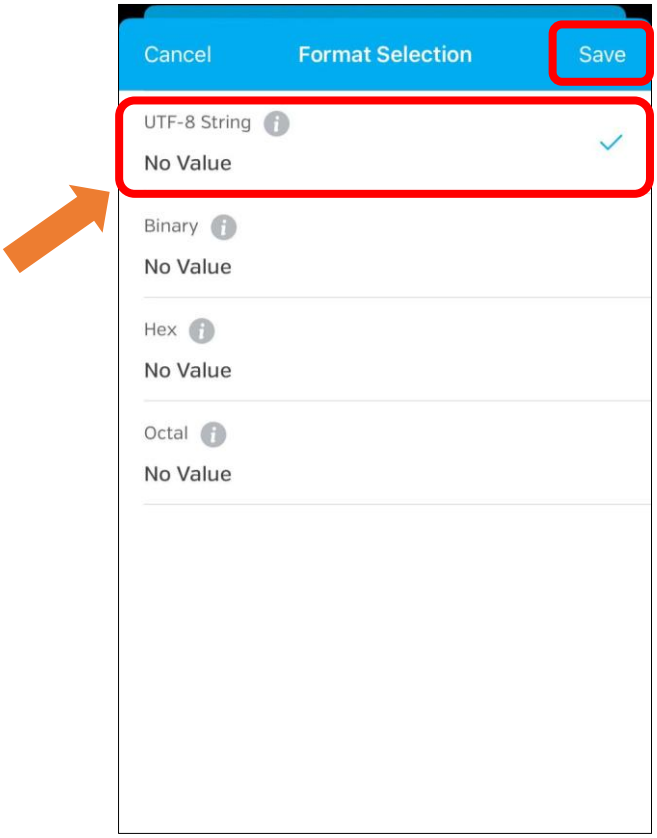
Click Write



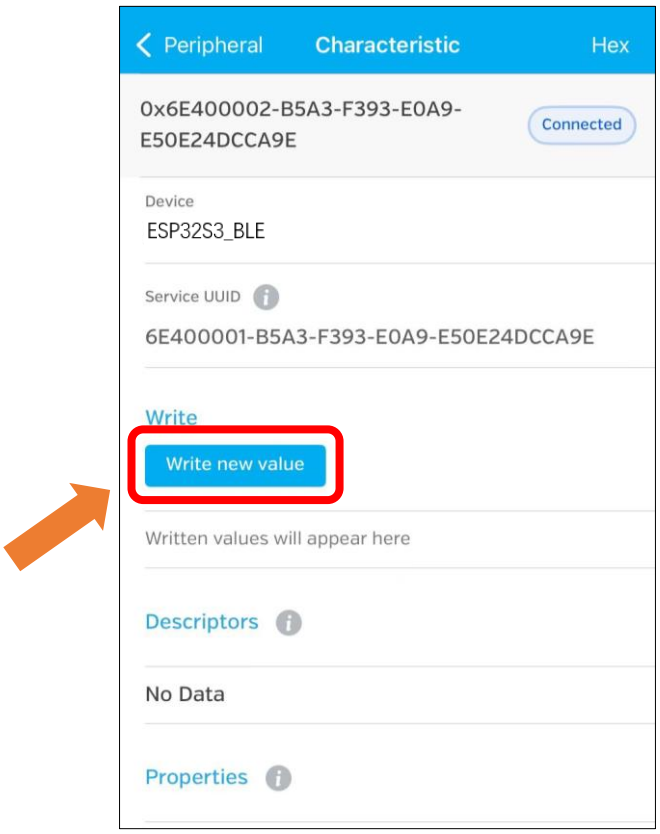
Click the top right corner to change the string type.



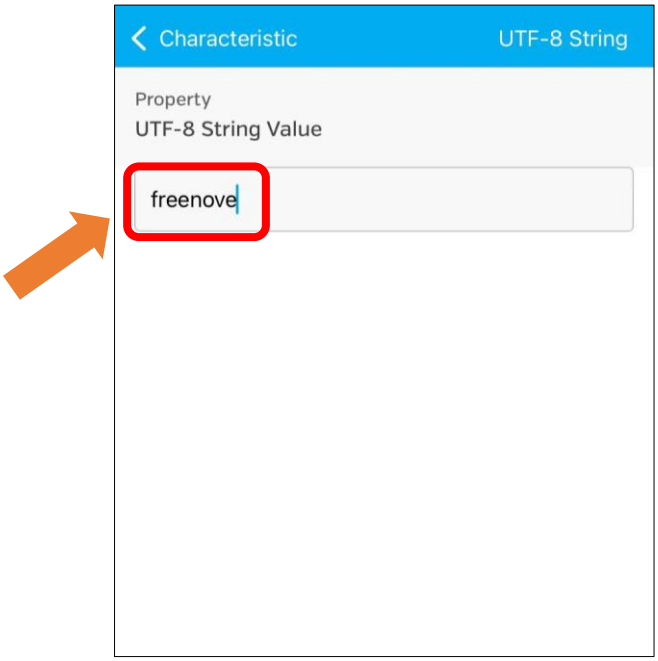
Select UTF-8 String and click “Save”.



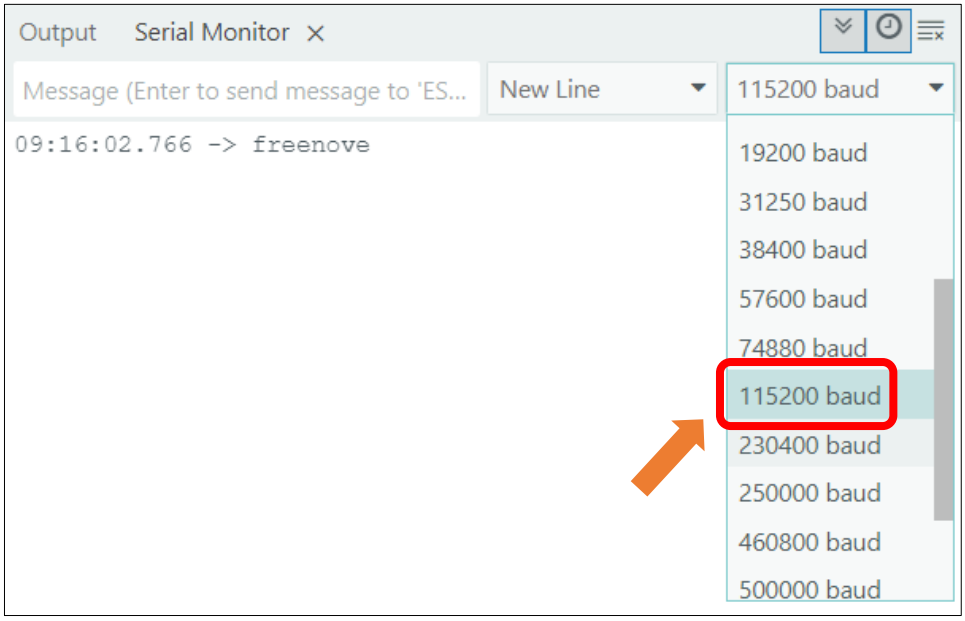
Click "Write new value"



Enter the messages to send.



Set the baud rate to 115200, and the serial monitor will print the data received.



Project 8.2 BLE RGB

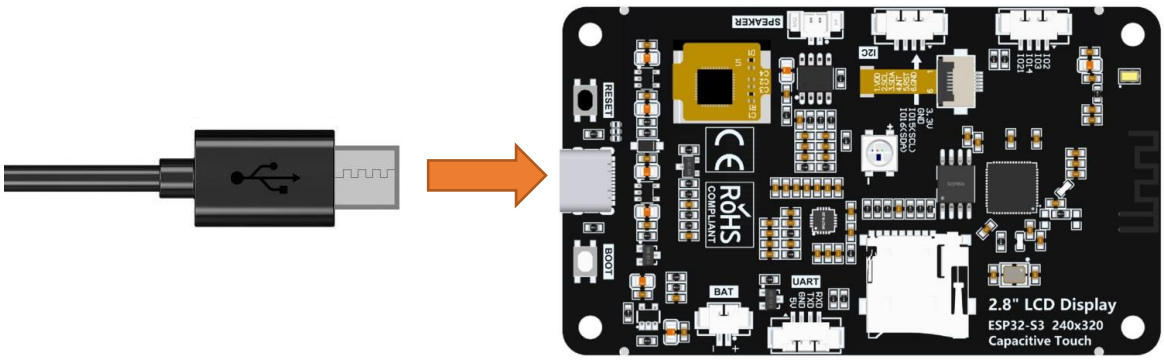
In this section we will control the RGB LED via BLE.

Component List

Freenove ESP32 S3 Display x 1 	USB cable x1 

Circuit

Connect Freenove ESP32-S3 to the computer using the USB cable.



Sketch

Next, we download the code to Freenove_ESP32_S3_Display to test. Open"**Sketch_08.2_BLE_RGB**" folder under **"Freenove_ESP32_S3_Display\NonTouch\Sketches"** and double-click "**Sketch_08.2_BLE_RGB.ino**".

Sketch_08.2_BLE_RGB

The following is the program code:

```
/*
* @ File:    Sketch_08.2_BLE_RGB.ino
* @ Author: [Zhentao Lin]
* @ Date:    [2025-06-14]
*/

#include "BLEDevice.h"
#include "BLEServer.h"
#include "BLEUtils.h"
#include "BLE2902.h"
#include "String.h"
#include "Freenove_WS2812_Lib_for_ESP32.h"

BLECharacteristic *pCharacteristic;
bool deviceConnected = false;
uint8_t txValue = 0;
long lastMsg = 0;
String rxload = "Test\n";

#define SERVICE_UUID "6E400001-B5A3-F393-E0A9-E50E24DCCA9E"
#define CHARACTERISTIC_UUID_RX "6E400002-B5A3-F393-E0A9-E50E24DCCA9E"
#define CHARACTERISTIC_UUID_TX "6E400003-B5A3-F393-E0A9-E50E24DCCA9E"

```

```
#define KEY_PIN 0
#define LEDS_COUNT 1
#define LEDS_PIN   42
#define CHANNEL    0

Freenove_ESP32_WS2812 strip = Freenove_ESP32_WS2812(LEDS_COUNT, LEDS_PIN, CHANNEL, TYPE_GRB);

uint8_t m_color[5][3] = { { 255, 0, 0 }, { 0, 255, 0 }, { 0, 0, 255 }, { 255, 255, 255 }, { 0,
0, 0 } };

void stripInit(void){
   strip.begin();
   strip.setBrightness(10);
}

void switchRGB(int value) {
   int colorIndex = value % 4;
   switch (colorIndex) {
     case 1:
       strip.setLedColorData(0, m_color[0][0], m_color[0][1], m_color[0][2]);
       strip.show();
       break;
     case 2:
       strip.setLedColorData(0, m_color[1][0], m_color[1][1], m_color[1][2]);
       strip.show();
       break;
     case 3:
       strip.setLedColorData(0, m_color[2][0], m_color[2][1], m_color[2][2]);
       strip.show();
       break;
     default:
       strip.setLedColorData(0, m_color[4][0], m_color[4][1], m_color[4][2]);
       strip.show();
       break;
   }
}

class MyServerCallbacks : public BLEServerCallbacks {
   void onConnect(BLEServer *pServer) {
     deviceConnected = true;
   };
   void onDisconnect(BLEServer *pServer) {
     deviceConnected = false;
   }
```

```
};

class MyCallbacks : public BLECharacteristicCallbacks {
    void onWrite(BLECharacteristic *pCharacteristic) {
      String rxValue = pCharacteristic->getValue();
      if (rxValue.length() > 0) {
        rxload = "";
        for (int i = 0; i < rxValue.length(); i++) {
          rxload += (char)rxValue[i];
        }
      }
    }
};

void setupBLE(String BLEName) {
    const char *ble_name = BLEName.c_str();
    BLEDevice::init(ble_name);
    BLEServer *pServer = BLEDevice::createServer();
    pServer->setCallbacks(new MyServerCallbacks());
    BLEService *pService = pServer->createService(SERVICE_UUID);
    pCharacteristic = pService->createCharacteristic(CHARACTERISTIC_UUID_TX,
BLECharacteristic::PROPERTY_NOTIFY);
    pCharacteristic->addDescriptor(new BLE2902());
    BLECharacteristic *pCharacteristic = pService->createCharacteristic(CHARACTERISTIC_UUID_RX,
BLECharacteristic::PROPERTY_WRITE);
    pCharacteristic->setCallbacks(new MyCallbacks());
    pService->start();
    pServer->getAdvertising()->start();
    Serial.println("Waiting a client connection to notify...");
}

void setup() {
    Serial.begin(115200);
    stripInit();
    switchRGB(0);
    setupBLE("ESP32S3_BLE");
}

void loop() {
    long now = millis();
    if (now - lastMsg > 100) {
      if (deviceConnected && rxload.length() > 0) {
        Serial.println(rxload);
        if (strncmp(rxload.c_str(), "red_on", 6) == 0) {
```

```
112           switchRGB(1);
113         } else if (strncmp(rxload.c_str(), "red_off", 7) == 0) {
114           switchRGB(0);
115         } else if (strncmp(rxload.c_str(), "green_on", 8) == 0) {
116           switchRGB(2);
117         } else if (strncmp(rxload.c_str(), "green_off", 9) == 0) {
118           switchRGB(0);
119         } else if (strncmp(rxload.c_str(), "blue_on", 7) == 0) {
120           switchRGB(3);
121         } else if (strncmp(rxload.c_str(), "blue_off", 8) == 0) {
122           switchRGB(0);
123         }
124         rxload = "";
125       }
126       if (Serial.available() > 0) {
127         String str = Serial.readString();
128         const char *newValue = str.c_str();
129         pCharacteristic->setValue(newValue);
130         pCharacteristic->notify();
131       }
132       lastMsg = now;
133     }
134 }
```

Code Explanation

Include the necessary header files.

```
7  #include "BLEDevice.h"
8  #include "BLEServer.h"
9  #include "BLEUtils.h"
10 #include "BLE2902.h"
11 #include "String.h"
12 #include "Freenove_WS2812_Lib_for_ESP32.h"
```

Define Service UUID and Characteristic UUID.

```
20 #define SERVICE_UUID "6E400001-B5A3-F393-E0A9-E50E24DCCA9E"
21 #define CHARACTERISTIC_UUID_RX "6E400002-B5A3-F393-E0A9-E50E24DCCA9E"
22 #define CHARACTERISTIC_UUID_TX "6E400003-B5A3-F393-E0A9-E50E24DCCA9E"
```

Define pins for the RGB LED.

```
24 #define KEY_PIN 0
25 #define LEDS_COUNT 1
26 #define LEDS_PIN   42
27 #define CHANNEL    0
```

The MyServerCallbacks class handles device connection and disconnection events and updates the deviceConnected status.

```
61 class MyServerCallbacks: public BLEServerCallbacks {
62     void onConnect(BLEServer* pServer) {
```

```
63          deviceConnected = true;
64        };
65        void onDisconnect(BLEServer* pServer) {
66          deviceConnected = false;
67        }
68    };
```

The MyCallbacks class handles the receiving data and save them to the rxload string.

```
70    class MyCallbacks: public BLECharacteristicCallbacks {
71        void onWrite(BLECharacteristic *pCharacteristic) {
72          String rxValue = pCharacteristic->getValue();
73          if (rxValue.length() > 0) {
74            rxload="";
75            for (int i = 0; i < rxValue.length(); i++){
76              rxload +=(char)rxValue[i];
77            }
78          }
79        }
80    };
```

Set the baud rate to 115200

```
97    Serial.begin(115200);
```

Initialize RGB LED setting, initialize the BLE device, create services, and set up characteristics.

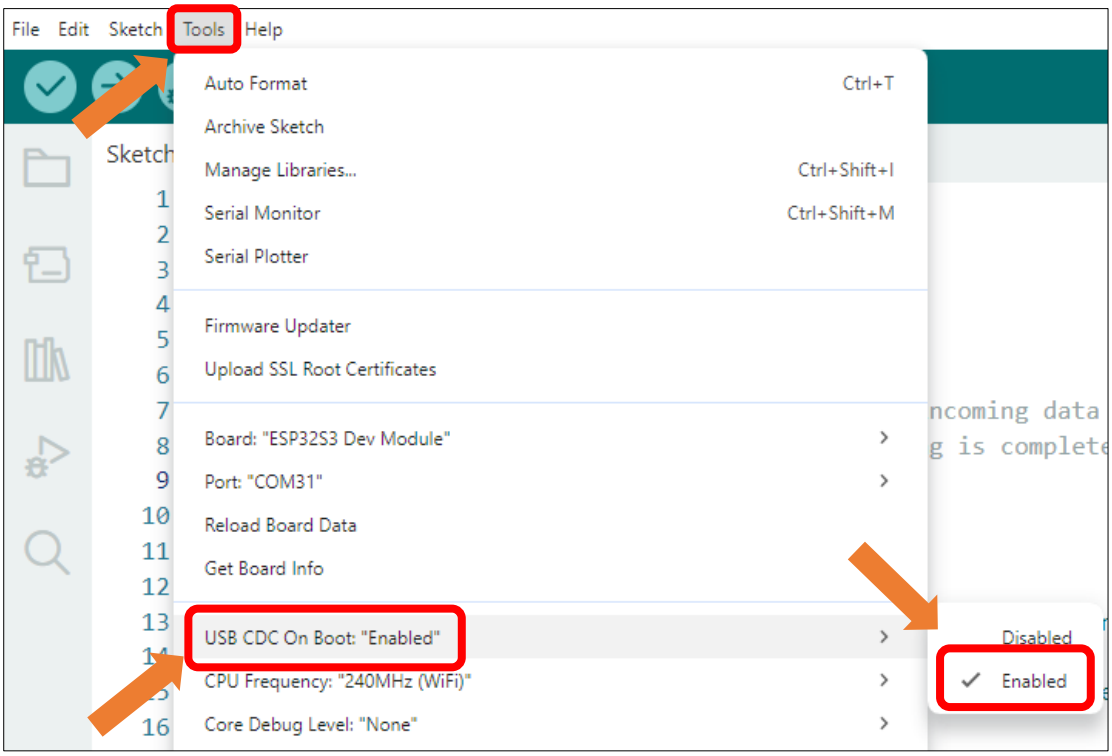
```
101   stripInit();
102   switchRGB(0);
103   setupBLE("ESP32S3_BLE");
```

The Loop function checks the sending command every 100 milliseconds.

```
106   void loop() {
107     long now = millis();
108     if (now - lastMsg > 100) {
109       if (deviceConnected && rxload.length() > 0) {
110         Serial.println(rxload);
111         if (strncmp(rxload.c_str(), "red_on", 6) == 0) {
112           setRGB(1, 0, 0);
113         } else if (strncmp(rxload.c_str(), "red_off", 7) == 0) {
114           setRGB(0, 0, 0);
115         } else if (strncmp(rxload.c_str(), "green_on", 8) == 0) {
116           setRGB(0, 1, 0);
117         } else if (strncmp(rxload.c_str(), "green_off", 9) == 0) {
118           setRGB(0, 0, 0);
119         } else if (strncmp(rxload.c_str(), "blue_on", 7) == 0) {
120           setRGB(0, 0, 1);
121         } else if (strncmp(rxload.c_str(), "blue_off", 8) == 0) {
122           setRGB(0, 0, 0);
123         }
124         rxload = "";
```

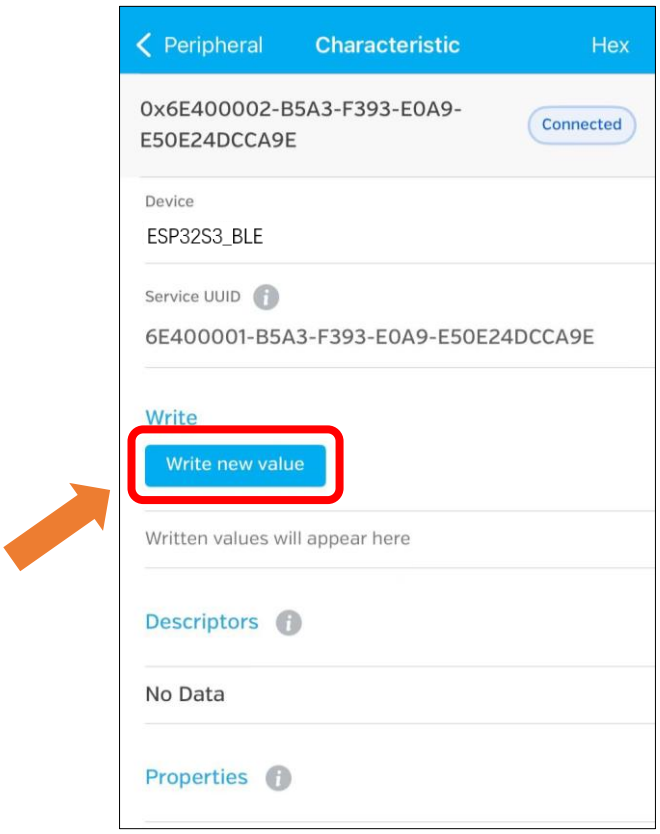
```
125         }
126         if (Serial.available() > 0) {
127           String str = Serial.readString();
128           const char *newValue = str.c_str();
129           pCharacteristic->setValue(newValue);
130           pCharacteristic->notify();
131         }
132         lastMsg = now;
133       }
134   }
```

Enable the "USB CDC On Boot" feature.

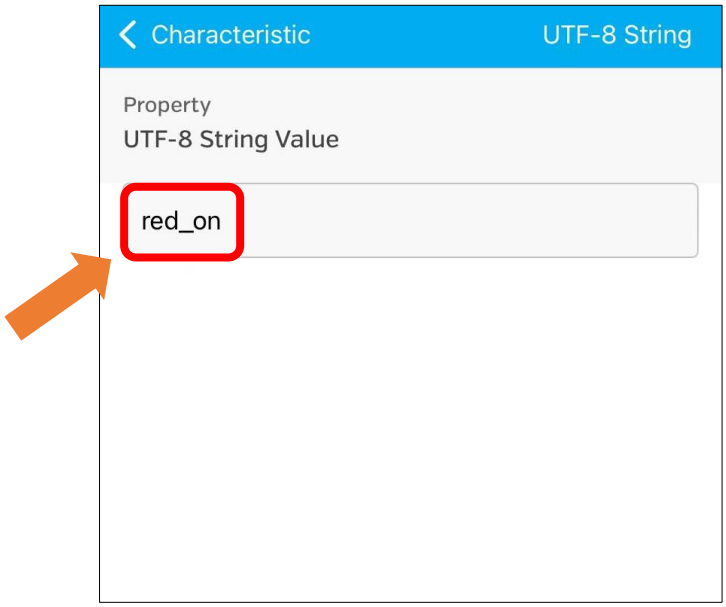


Click "**Upload**" to upload the code to Freenove ESP32 S3 Display.

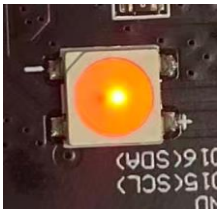
Click "Write new value"



Enter the messages to send. Here we take "red_on" as an example.



The RGB LED on the Freenove ESP32 S3 Display emits red light.



You can also use the following instructions to control the RGB LED:

red_off: red light off
green_on: green light on
green_off: green light off
blue_on: blue light on
blue_off: blue light off

Chapter 9 WIFI Web Server

Project 9.1 WIFI Web Servers LED

Component List

Freenove ESP32 S3 Display x 1 	USB cable x1 

Component Knowledge

Wi-Fi

Wi-Fi is a wireless LAN (WLAN) technology compliant with the IEEE 802.11 standard, enabling devices to transmit data or connect to the internet via radio waves within short-range coverage (typically up to tens of meters). In embedded systems, Wi-Fi modules provide wireless communication capabilities, allowing devices to connect to routers, access cloud services, or interact with other devices.

Its core features include:

- Wireless connectivity (eliminating the need for physical cabling)
- Moderate to high-speed data transfer (depending on protocol versions such as 802.11n/ac)
- Secure communication (ensured by encryption protocols like WPA2/WPA3)

In embedded development, Wi-Fi is one of the key technologies for enabling IoT (Internet of Things) device connectivity.

Web

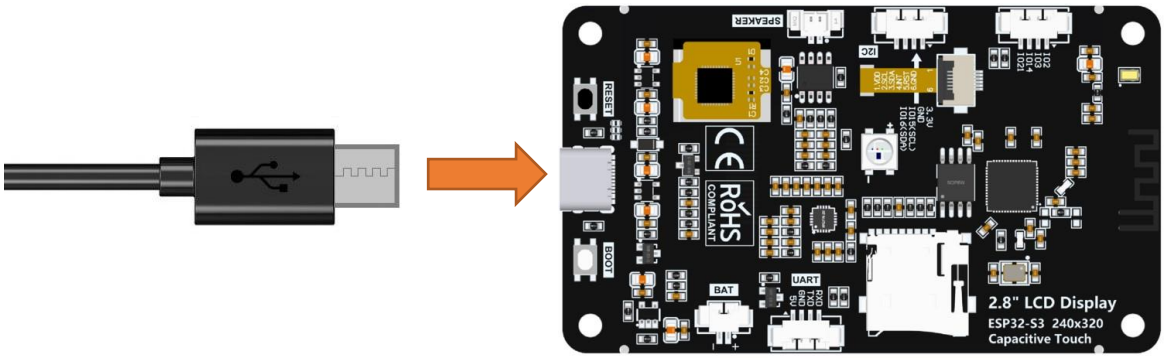
The Web (World Wide Web) is an internet-based information system that integrates global resources through hypertext links. In the embedded field, Web technology refers to devices equipped with lightweight built-in web servers, allowing users to remotely access the device interface via browsers (such as Chrome or Edge). By entering the device's IP address, users can open an interactive page to perform functions like status monitoring, parameter configuration, or firmware upgrades. Its core value lies in cross-platform compatibility (no need for dedicated software installation) and standardized protocols (HTTP/HTTPS), making it a mainstream solution for remote management of embedded devices.

HTML & CSS & JavaScript

- **HTML (HyperText Markup Language)** is the standard language for structuring web pages. It uses tags to define page elements, forming the foundational framework. In embedded web interfaces, HTML describes the layout of components such as device status displays and configuration forms.
- **CSS (Cascading Style Sheets)** controls the visual presentation of web pages, including colors, fonts, spacing, and responsive layouts. Through CSS rules, developers style HTML elements into intuitive interactive interfaces. The combination of HTML and CSS enables the creation of lightweight device control pages without complex graphics libraries, reducing resource overhead in embedded systems.
- **JavaScript (a scripting language)** adds dynamic behavior and interaction logic to web pages. In embedded web interfaces, JavaScript plays a crucial role: it responds to user actions and enables asynchronous communication with the device backend through technologies like **AJAX** or **WebSocket**. This allows the webpage to fetch real-time device status data, update displays without refreshing, and send control commands or configuration parameters—delivering a smooth and efficient remote management experience.

Circuit

Connect Freenove ESP32 -S3 to the computer using the USB cable.



Sketch

Next, we download the code to Freenove_ESP32_S3_Display to test. Open "**Sketch_09.1_WiFi_Web_Servers_LED**" folder under **"Freenove_ESP32_S3_Display\NonTouch\Sketches"** and double-click "**Sketch_09.1_WiFi_Web_Servers_LED.ino**".

Important Note: Before upload the code, please ensure that the Wi-Fi SSID and Password are correctly configured in the code, and that the Wi-Fi network to connect is 2.4GHz.

```
10 const char* ssid     = "**************";
11 const char* password = "**************";
```

Sketch_09.1_WiFi_Web_Servers_LED

The following is the program code:

```
1 /*
2 * @ File:   Sketch_09.1_WiFi_Web_Servers_LED.ino (modified to use WS2812 strip with color
3 names)
4 * @ Author: [Zhentao Lin]
```

```
* @ Date:   [2025-08-23]
*/

#include <WiFi.h>
#include "Freenove_WS2812_Lib_for_ESP32.h"

// WiFi credentials
const char* ssid     = "**************";
const char* password = "**************";

// Web server on port 80
WiFiServer server(80);

// HTTP request buffer
String request;

// WS2812 LED strip configuration
#define LEDS_COUNT 1
#define LEDS_PIN   42
#define CHANNEL    0

Freenove_ESP32_WS2812 strip = Freenove_ESP32_WS2812(LEDS_COUNT, LEDS_PIN, CHANNEL, TYPE_GRB);

// LED state tracking (0=Red, 1=Green, 2=Blue)
bool ledStates[3] = {false, false, false}; // All OFF initially
String redLedState = "OFF";
String greenLedState = "OFF";
String blueLedState = "OFF";

// Timeout settings
unsigned long currentTime = millis();
unsigned long previousTime = 0;
const long timeoutTime = 2000;  // 2 seconds

void setupStrip(void) {
    strip.begin();
    strip.setBrightness(10);
    strip.setLedColorData(0, 0, 0, 0); // Turn off all LEDs
    strip.show();
}

void updateLedStrip(void) {
    uint8_t r = ledStates[0] ? 255 : 0;
    uint8_t g = ledStates[1] ? 255 : 0;
```

```
49      uint8_t b = ledStates[2] ? 255 : 0;
50
51      strip.setLedColorData(0, r, g, b);
52      strip.show();
53  }
54
55  void setup() {
56      Serial.begin(115200);
57
58      // Initialize LED strip
59      setupStrip();
60
61      // Connect to WiFi
62      Serial.print("Connecting to ");
63      Serial.println(ssid);
64      WiFi.begin(ssid, password);
65      while (WiFi.status() != WL_CONNECTED) {
66        delay(500);
67        Serial.print(".");
68      }
69
70      Serial.println("");
71      Serial.println("WiFi connected.");
72      Serial.println("IP address: ");
73      Serial.println(WiFi.localIP());
74
75      server.begin();
76  }
77
78  void loop() {
79      WiFiClient client = server.available();
80
81      if (client) {
82        currentTime = millis();
83        previousTime = currentTime;
84        Serial.println("New Client.");
85        String currentLine = "";
86
87        while (client.connected() && currentTime - previousTime <= timeoutTime) {
88          currentTime = millis();
89          if (client.available()) {
90            char c = client.read();
91            Serial.write(c);
92            request += c;
```

```
93
94          if (c == '\n') {
95            if (currentLine.length() == 0) {
96              // HTTP response headers
97              // Handle incoming commands first. These are AJAX requests.
98              if (request.indexOf("GET /red/ON") >= 0) {
99                Serial.println("Red LED ON");
100               redLedState = "ON";
101               ledStates[0] = true;
102             } else if (request.indexOf("GET /red/OFF") >= 0) {
103               Serial.println("Red LED OFF");
104               redLedState = "OFF";
105               ledStates[0] = false;
106             } else if (request.indexOf("GET /green/ON") >= 0) {
107               Serial.println("Green LED ON");
108               greenLedState = "ON";
109               ledStates[1] = true;
110             } else if (request.indexOf("GET /green/OFF") >= 0) {
111               Serial.println("Green LED OFF");
112               greenLedState = "OFF";
113               ledStates[1] = false;
114             } else if (request.indexOf("GET /blue/ON") >= 0) {
115               Serial.println("Blue LED ON");
116               blueLedState = "ON";
117               ledStates[2] = true;
118             } else if (request.indexOf("GET /blue/OFF") >= 0) {
119               Serial.println("Blue LED OFF");
120               blueLedState = "OFF";
121               ledStates[2] = false;
122             }
123
124             // Update LED strip based on states
125             updateLedStrip();
126
127             // Send HTML page for non-AJAX requests
128             if (request.indexOf("/red/") < 0 && request.indexOf("/green/") < 0 &&
129 request.indexOf("/blue/") < 0) {
130               // Send the Complete HTML page
131               client.println("HTTP/1.1 200 OK");
132               client.println("Content-type:text/html");
133               client.println("Connection: close");
134               client.println();
135
136               // Generate HTML response
```

```
137                 client.println("<!DOCTYPE html><html>");
138                 client.println("<head><meta charset=\"UTF-8\">");
139                 client.println("<meta name=\"viewport\" content=\"width=device-width, initial-
140 scale=1.0\">");
141                 client.println("<title>ESP32 Web Server LED</title>");
142                 client.println("<style>");
143                 client.println("html{font-family:'Segoe UI',Roboto,sans-
144 serif;background:#f5f5f5;height:100vh;display:flex;justify-content:center;align-
145 items:center;}");
146                 client.println("body{margin:0;padding:20px;background:white;border-
147 radius:15px;box-shadow:0 10px 30px rgba(0,0,0,0.1);max-width:800px;width:90vw;}");
148                 client.println(".container{display:grid;grid-template-columns:repeat(3,
149 1fr);gap:20px;margin-top:20px;}");
150                 client.println(".card{border-radius:15px;padding:25px;text-align:center;min-
151 width:180px;transition:all 0.3s;cursor:pointer;border:2px
152 solid;position:relative;overflow:hidden;}");
153                 client.println(".card:hover{transform:translateY(-5px);box-shadow:0 15px 35px
154 rgba(0,0,0,0.15);}");
155                 client.println(".card:active{transform:translateY(0) scale(0.98);}");
156                 client.println("#red-card{background:linear-
157 gradient(145deg,#ffcdd2,#ef9a9a);border-color:#f44336;}");
158                 client.println("#green-card{background:linear-
159 gradient(145deg,#c8e6c9,#a5d6a7);border-color:#4caf50;}");
160                 client.println("#blue-card{background:linear-
161 gradient(145deg,#bbdefb,#90caf9);border-color:#2196f3;}");
162                 client.println(".card.off-state{filter:grayscale(70%) brightness(0.85);}");
163                 client.println("h1{color:#333;text-align:center;font-size:2.4rem;margin-
164 bottom:10px;border-bottom:2px solid #eee;padding-bottom:15px;}");
165                 client.println(".status{font-size:1.4rem;font-weight:600;margin:25px
166 0;color:#333;position:relative;z-index:2;}");
167                 client.println(".device-info{text-align:center;margin-top:25px;color:#777;font-
168 size:0.9rem;padding-top:15px;border-top:1px solid #eee;}");
169                 client.println("@media (max-width:768px){.container{grid-template-columns:1fr;}
170 body{padding:15px;} h1{font-size:2rem;}}");
171                 client.println("</style></head>");
172                 client.println("<body>");
173                 client.println("<h1>ESP32 Web Server LED</h1>");
174                 client.println("<div class=\"container\">");
175 
176                 // Red LED control
177                 client.println("<div class=\"card " + String(redLedState == "OFF" ? "off-
178 state" : "") + "\" id=\"red-card\" onclick=\"toggleLED('red', this)\">");
179                 client.println("<h2>Red LED</h2>");
180 
```

```
181                 client.println("<p class=\"status\" id=\"status-red\">" + redLedState +
182 "</p>");
183                 client.println("</div>");
184 
185                 // Green LED control
186                 client.println("<div class=\"card " + String(greenLedState == "OFF" ? "off-
187 state" : "") + "\" id=\"green-card\" onclick=\"toggleLED('green', this)\">");
188                 client.println("<h2>Green LED</h2>");
189                 client.println("<p class=\"status\" id=\"status-green\">" + greenLedState +
190 "</p>");
191                 client.println("</div>");
192 
193                 // Blue LED control
194                 client.println("<div class=\"card " + String(blueLedState == "OFF" ? "off-
195 state" : "") + "\" id=\"blue-card\" onclick=\"toggleLED('blue', this)\">");
196                 client.println("<h2>Blue LED</h2>");
197                 client.println("<p class=\"status\" id=\"status-blue\">" + blueLedState +
198 "</p>");
199                 client.println("</div>");
200 
201                 client.println("</div>");
202                 client.println("<div class=\"device-info\">Device IP: " +
203 WiFi.localIP().toString() + " | ESP32 Web Server LED</div>");
204                 client.println("<script>");
205                 client.println("function toggleLED(color, element) {");
206                 client.println("  const statusElement = document.getElementById('status-' +
207 color);");
208                 client.println("  const currentState = statusElement.innerText;");
209                 client.println("  const newState = currentState === 'ON' ? 'OFF' : 'ON';");
210                 client.println("  fetch('/' + color + '/' + newState);");
211                 client.println("  statusElement.innerText = newState;");
212                 client.println("  if (newState === 'ON') { element.classList.remove('off-
213 state'); } else { element.classList.add('off-state'); }");
214                 client.println("}");
215                 client.println("</script>");
216                 client.println("</body></html>");
217               }
218               break;
219             } else {
220               currentLine = "";
221             }
222           } else if (c != '\r') {
223             currentLine += c;
224           }
```

```
225         }
226       }
227
228       request = "";
229       client.stop();
230       Serial.println("Client disconnected.");
231       Serial.println("");
232     }
233 }
```

Code Explanation

Include necessary header files.

```
7  #include <WiFi.h>
```

Configure WiFi SSID and password.

```
12 const char* ssid     = "**************";
13 const char* password = "**************";
```

Define pins for the RGB LED.

```
21 // WS2812 LED strip configuration
22 #define LEDS_COUNT 1
23 #define LEDS_PIN   42
24 #define CHANNEL    0
```

Initialize the configuration related to RGB LED.

```
40 strip.begin();
41 strip.setBrightness(10);
42 strip.setLedColorData(0, 0, 0, 0); // Turn off all LEDs
43 strip.show();
```

Connect to WIFI.

```
61 // Connect to WiFi
62 Serial.print("Connecting to ");
63 Serial.println(ssid);
64 WiFi.begin(ssid, password);
65 while (WiFi.status() != WL_CONNECTED) {
66    delay(500);
67    Serial.print(".");
68 }
69
70 Serial.println("");
71 Serial.println("WiFi connected.");
72 Serial.println("IP address: ");
73 Serial.println(WiFi.localIP());
74
75 server.begin();
```

Check if there are any new clients connected to the server.

```
78 WiFiClient client = server.available();
```

Parse the URL command and control the corresponding LED on and off.

```
// Handle incoming commands
if (request.indexOf("GET /red/ON") >= 0) {
   Serial.println("Red LED ON");
   redLedState = "ON";
   ledStates[0] = true;
} else if (request.indexOf("GET /red/OFF") >= 0) {
   Serial.println("Red LED OFF");
   redLedState = "OFF";
  ledStates[0] = false;
} else if (request.indexOf("GET /green/ON") >= 0) {
  Serial.println("Green LED ON");
  greenLedState = "ON";
  ledStates[1] = true;
} else if (request.indexOf("GET /green/OFF") >= 0) {
  Serial.println("Green LED OFF");
  greenLedState = "OFF";
  ledStates[1] = false;
} else if (request.indexOf("GET /blue/ON") >= 0) {
  Serial.println("Blue LED ON");
  blueLedState = "ON";
  ledStates[2] = true;
} else if (request.indexOf("GET /blue/OFF") >= 0) {
  Serial.println("Blue LED OFF");
  blueLedState = "OFF";
  ledStates[2] = false;
}
```

Use CSS styles to enhance the visual appeal of web interfaces.

```
client.println("html{font-family:'Segoe UI',Roboto,sans-
serif;background:#f5f5f5;height:100vh;display:flex;justify-content:center;align-
items:center;}");
client.println("body{margin:0;padding:20px;background:white;border-radius:15px;box-shadow:0
10px 30px rgba(0,0,0,0.1);max-width:800px;width:90vw;}");
client.println(".container{display:grid;grid-template-columns:repeat(3, 1fr);gap:20px;margin-
top:20px;}");
client.println(".card{border-radius:15px;padding:25px;text-align:center;min-
width:180px;transition:all 0.3s;cursor:pointer;border:2px
solid;position:relative;overflow:hidden;}");
client.println(".card:hover{transform:translateY(-5px);box-shadow:0 15px 35px
rgba(0,0,0,0.15);}");
client.println(".card:active{transform:translateY(0) scale(0.98);}");
client.println("#red-card{background:linear-gradient(145deg,#ffcdd2,#ef9a9a);border-
color:#f44336;}");
client.println("#green-card{background:linear-gradient(145deg,#c8e6c9,#a5d6a7);border-
color:#4caf50;}");
```

```
client.println("#blue-card{background:linear-gradient(145deg,#bbdefb,#90caf9);border-
color:#2196f3;}");
client.println(".card.off-state{filter:grayscale(70%) brightness(0.85);}");
client.println("h1{color:#333;text-align:center;font-size:2.4rem;margin-bottom:10px;border-
bottom:2px solid #eee;padding-bottom:15px;}");
client.println(".status{font-size:1.4rem;font-weight:600;margin:25px
0;color:#333;position:relative;z-index:2;}");
client.println(".device-info{text-align:center;margin-top:25px;color:#777;font-
size:0.9rem;padding-top:15px;border-top:1px solid #eee;}");
client.println("@media (max-width:768px){.container{grid-template-columns:1fr;}
body{padding:15px;} h1{font-size:2rem;}}");
```

Generate the HTML contents to control the LED.

```
// Red LED control
client.println("<div class=\"card " + String(redLedState == "OFF" ? "off-state" : "") + "\"
id=\"red-card\" onclick=\"toggleLED('22', this)\">");
client.println("<h2>Red LED</h2>");
client.println("<p class=\"status\" id=\"status-22\">" + redLedState + "</p>");
client.println("</div>");

// Green LED control
client.println("<div class=\"card " + String(greenLedState == "OFF" ? "off-state" : "") + "\"
id=\"green-card\" onclick=\"toggleLED('16', this)\">");
client.println("<h2>Green LED</h2>");
client.println("<p class=\"status\" id=\"status-16\">" + greenLedState + "</p>");
client.println("</div>");

// Blue LED control
client.println("<div class=\"card " + String(blueLedState == "OFF" ? "off-state" : "") + "\"
id=\"blue-card\" onclick=\"toggleLED('17', this)\">");
client.println("<h2>Blue LED</h2>");
client.println("<p class=\"status\" id=\"status-17\">" + blueLedState + "</p>");
client.println("</div>");
```

Disconnect from the client and print the disconnection information in the serial monitor

```
client.stop();
Serial.println("Client disconnected.");
Serial.println("");
```

Input correct WiFi(2.4GHz) SSID and password.

```
// WiFi credentials
const char* ssid     = "**************";
const char* password = "**************";
```

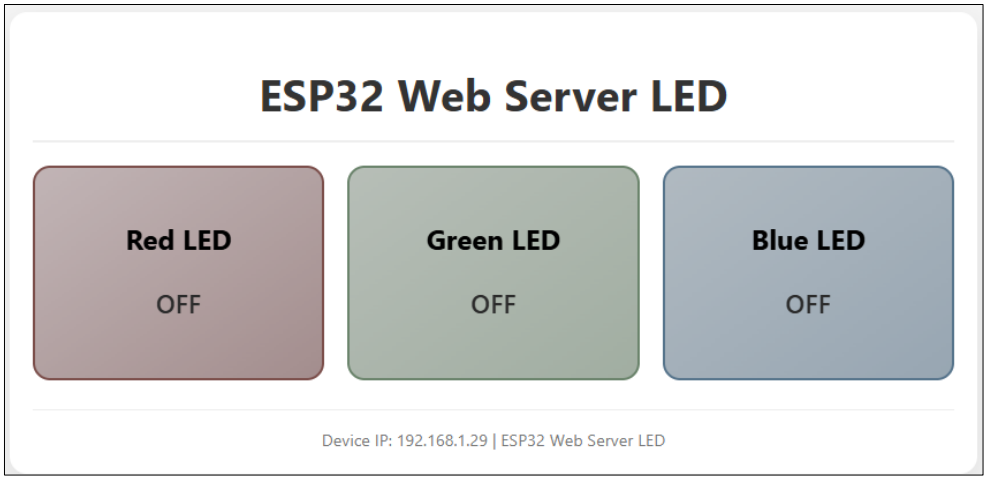
Click "**Upload**" to upload the code to Freenove ESP32 Display.

When the serial monitor prints the following information, it means the network connection is successful. Use a mobile phone or computer browser to open the IP address printed by the serial monitor.

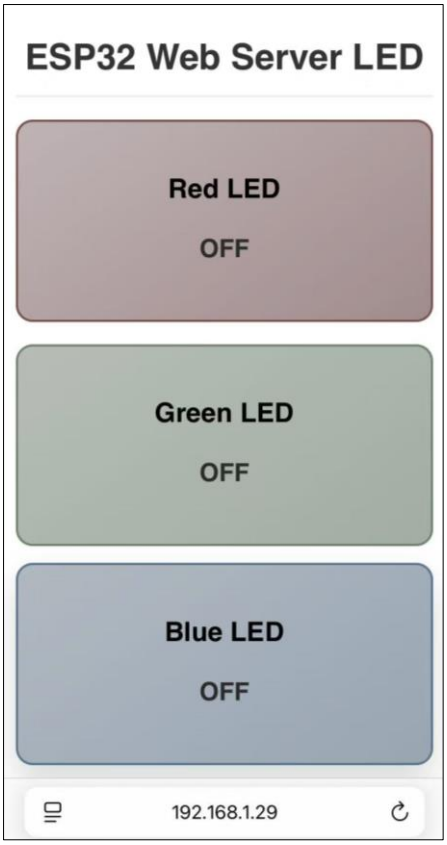
Please note: If it keeps printing dots, please confirm whether the WiFi is in the 2.4G band.

```
14:31:24.553 -> Connecting to FYI_2.4G
14:31:25.115 -> .....
14:31:27.088 -> WiFi connected.
14:31:27.088 -> IP address:
14:31:27.122 -> 192.168.1.29
```

The following screen will be displayed in the computer's browser, where you can control the color of the RGB light by clicking on the three cards.



On the phone web browser, you'll see the following contents. The color of the RGB light can be controlled by clicking on the three cards.


ESP32 Web Server LED
Red LED
OFF
Green LED
OFF
Blue LED
OFF
192.168.1.29

Chapter 10 TFT Display

Project 10.1 TFT_Rainbow

Component List

Freenove ESP32 S3 Display x 1 	USB cable x1 

Component Knowledge

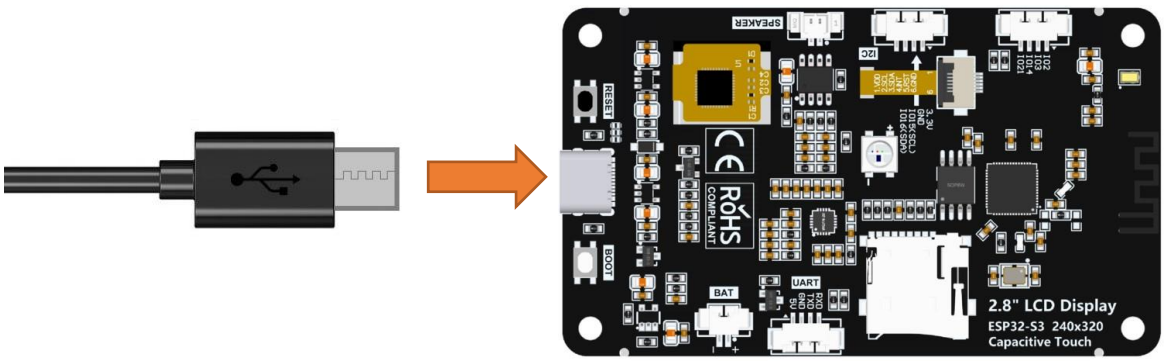
TFT Display

TFT (Thin Film Transistor) is an electronic component that serves as the foundation for TFT displays, the mainstream display technology in modern laptops and desktop computers. In these displays, each individual liquid crystal pixel is controlled by its own dedicated thin-film transistor embedded directly behind it. This architecture classifies TFT screens as a form of active-matrix LCD (AMLCD) technology.

As one of the finest LCD color displays available, TFT screens offer superior performance characteristics including rapid response times, exceptional brightness levels, and outstanding contrast ratios.

Circuit

Connect Freenove ESP32 -S3 to the computer using the USB cable.

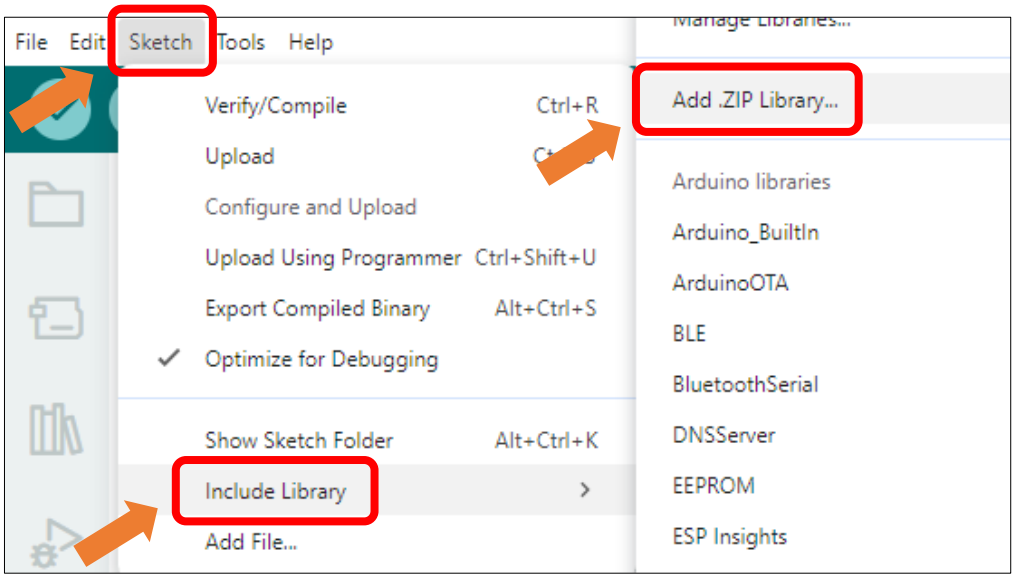


Sketch

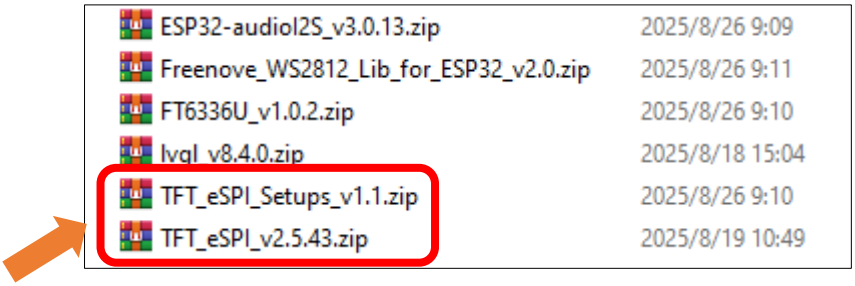
Open "**Sketch_10.1_TFT_Rainbow.ino**" folder under **"Freenove_ESP32_S3_Display\NonTouch\Sketches"** and double-click "**Sketch_10.1_TFT_Rainbow.ino**".

Install Libraries

Click **Sketch** -> **Include Library** -> **Add .ZIP Library...**

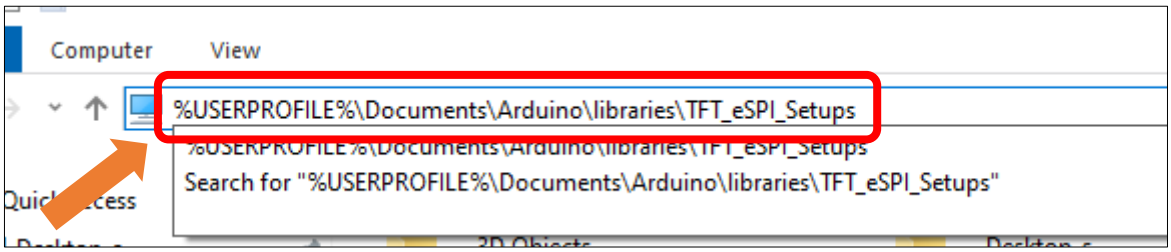


Install **TFT_eSPI_v2.5.43.zip** and **TFT_eSPI_Setups_v1.1.zip**

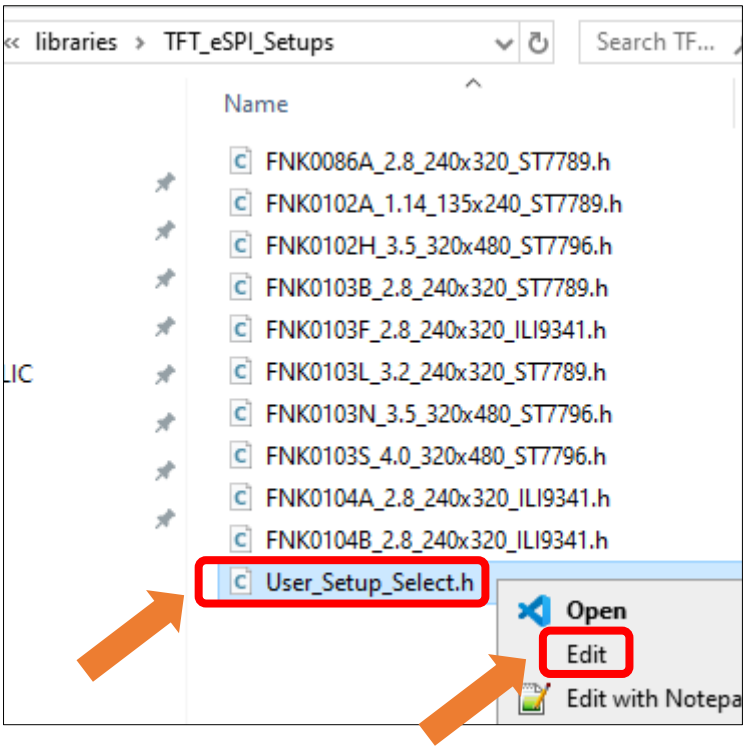


How to configure (Important)

Open This PC, input **%USERPROFILE%\Documents\Arduino\libraries\TFT_eSPI_Setups** and press the **Enter** key.



Right click **User_Setup_Select.h**, click **Edit**.



Uncomment the corresponding macro definition based on the model purchased.
If it is **1.14inch**, configure as below:

```
#define FNK0102A_1P14_135x240_ST7789
//#define FNK0102B_3P5_320x480_ST7796
//#define FNK0103B_2P8_240x320_ST7789
//#define FNK0103F_2P8_240x320_ILI9341
//#define FNK0103L_3P2_240x320_ST7789
//#define FNK0103N_3P5_320x480_ST7796
//#define FNK0103S_4P0_320x480_ST7796
```

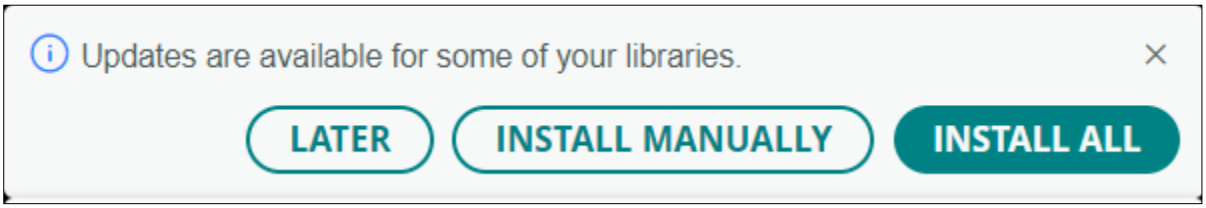
If it is **3.5inch**, configure as below:

```
//#define FNK0086A_2P8_240x320_ST7789
//#define FNK0102A_1P14_135x240_ST7789
#define FNK0102B_3P5_320x480_ST7796
//#define FNK0103B_2P8_240x320_ST7789
//#define FNK0103F_2P8_240x320_ILI9341
//#define FNK0103L_3P2_240x320_ST7789
//#define FNK0103N_3P5_320x480_ST7796
//#define FNK0103S_4P0_320x480_ST7796
```

Important Note: Only one macro definition should be uncommented.
Save the change and exit the file.

Warning:

If the following update prompt appears, click "LATER". Updating the TFT_eSPI library will reset all related configurations. If you click "INSTALL", please reinstall the TFT_eSPI library to ensure proper project operation.



Sketch_10.1_TFT_Rainbow

The following is the program code:

```
/*
* @ File:   Sketch_10.1_TFT_Rainbow.ino
* @ Author: [Zhentao Lin]
* @ Date:   [2025-08-23]
*/

#include <TFT_eSPI.h>  // TFT display library

TFT_eSPI tft = TFT_eSPI();  // Create TFT object instance

unsigned long targetTime = 0;     // Timing variable to control animation intervals
byte red = 31;                    // Start with full red
byte green = 0;                   // No green
byte blue = 0;                    // No blue
byte state = 0;                   // State machine variable
unsigned int colour = red << 11;  // Initial color value: Red only

void setup(void) {
   tft.init();                   // Initialize the TFT screen
   tft.setRotation(1);           // Set screen rotation (landscape mode)
   targetTime = millis() + 100;  // Set initial target time for rainbow effect
   tft.fillScreen(TFT_RED);  // Fill screen with solid red
   delay(1000);
   tft.fillScreen(TFT_GREEN);  // Fill screen with solid green
   delay(1000);
   tft.fillScreen(TFT_BLUE);  // Fill screen with solid blue
   delay(1000);
   tft.fillScreen(TFT_BLACK);  // Fill screen with solid black
   delay(1000);
   tft.fillScreen(TFT_WHITE);  // Fill screen with solid white
   delay(1000);
}
```

```
33
34  void loop() {
35    if (targetTime < millis()) {      // Check if it's time to start the rainbow animation
36      targetTime = millis() + 10000;  // Set next trigger after 10 seconds
37
38      for (int i = 0; i < 320; i++) {                  // Generate horizontal rainbow using
39  vertical lines
40        tft.drawFastVLine(i, 0, tft.height(), colour);  // Draw one vertical line
41        switch (state) {
42          case 0:
43            green += 2;  // Transition from red to yellow (increase green)
44            if (green == 64) {
45              green = 63;  // Cap at max green value
46              state = 1;
47            }
48            break;
49          case 1:
50            red--;  // Transition from yellow to green (decrease red)
51            if (red == 255) {
52              red = 0;
53              state = 2;
54            }
55            break;
56          case 2:
57            blue++;  // Transition from green to cyan (increase blue)
58            if (blue == 32) {
59              blue = 31;  // Cap at max blue value
60              state = 3;
61            }
62            break;
63          case 3:
64            green -= 2;  // Transition from cyan to blue (decrease green)
65            if (green == 255) {
66              green = 0;
67              state = 4;
68            }
69            break;
70          case 4:
71            red++;  // Transition from blue to magenta (increase red)
72            if (red == 32) {
73              red = 31;  // Cap at max red value
74              state = 5;
75            }
76            break;
```

```
77            case 5:
78              blue--;  // Transition from magenta to red (decrease blue)
79              if (blue == 255) {
80                blue = 0;
81                state = 0;
82              }
83              break;
84          }
85          colour = red << 11 | green << 5 | blue;  // Combine RGB values into 16-bit color
86        }
87        tft.setTextColor(TFT_BLACK);           // Set text color to black
88        tft.setCursor(12, 5);                  // Set cursor position
89        tft.print("Original ADAfruit font!");  // Print simple string
90        tft.setTextColor(TFT_BLACK, TFT_BLACK);  // Transparent background
91        tft.drawCentreString("Font size 2", 80, 14, 2);              // Font 2
92        tft.drawCentreString("Font size 4", 80, 30, 4);              // Font 4
93        tft.drawCentreString("12.34", 80, 54, 6);                    // Font 6
94        tft.drawCentreString("12.34 is in font size 6", 80, 92, 2);  // Font 2
95        float pi = 3.14159;                                          // Value to print
96        int precision = 3;                                           // Number of decimal digits
97        int xpos = 50;                                               // X position
98        int ypos = 110;                                              // Y position
99        int font = 2;                                                // Font type
100       xpos += tft.drawFloat(pi, precision, xpos, ypos, font);  // Draw float and update x-
101   position
102       tft.drawString(" is pi", xpos, ypos, font);              // Continue drawing string from
103   updated x position
104       delay(6000);  // Wait before next cycle
105     }
106   }
```

Code Explanation

Include the necessary header file.

```
7   #include <TFT_eSPI.h>  // TFT display library
```

Define TFT display object.

```
9   TFT_eSPI tft = TFT_eSPI();  // Create TFT object instance
```

Initialize the TFT display.

```
19  tft.init();                 // Initialize the TFT screen
20  tft.setRotation(1);         // Set screen rotation (landscape mode)
```

Change the color of the screen in the sequence of red -> green -> blue -> black ->white.

```
22  tft.fillScreen(TFT_RED);  // Fill screen with solid red
23  delay(1000);
24  tft.fillScreen(TFT_GREEN);  // Fill screen with solid green
25  delay(1000);
26  tft.fillScreen(TFT_BLUE);  // Fill screen with solid blue
```

```
27  delay(1000);
28  tft.fillScreen(TFT_BLACK);  // Fill screen with solid black
29  delay(1000);
30  tft.fillScreen(TFT_WHITE);  // Fill screen with solid white
31  delay(1000);
```

Implement the rainbow animation effect.

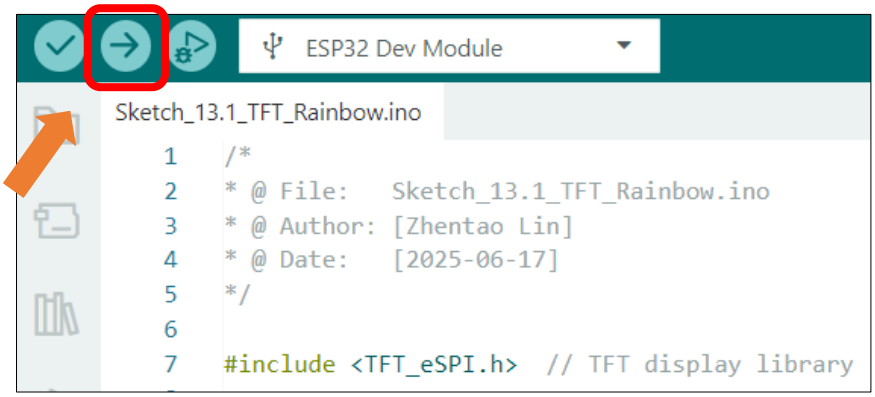
```
38  for (int i = 0; i < 320; i++) {           // Generate horizontal rainbow using vertical lines
39    tft.drawFastVLine(i, 0, tft.height(), colour);  // Draw one vertical line
40    switch (state) {
41      case 0:
42        green += 2;  // Transition from red to yellow (increase green)
43        if (green == 64) {
44          green = 63;  // Cap at max green value
45          state = 1;
46        }
47        break;
48      case 1:
49        red--;  // Transition from yellow to green (decrease red)
50        if (red == 255) {
51          red = 0;
52          state = 2;
53        }
54        break;
55      case 2:
56        blue++;  // Transition from green to cyan (increase blue)
57        if (blue == 32) {
58          blue = 31;  // Cap at max blue value
59          state = 3;
60        }
61        break;
62      case 3:
63        green -= 2;  // Transition from cyan to blue (decrease green)
64        if (green == 255) {
65          green = 0;
66          state = 4;
67        }
68        break;
69      case 4:
70        red++;  // Transition from blue to magenta (increase red)
71        if (red == 32) {
72          red = 31;  // Cap at max red value
73          state = 5;
74        }
75        break;
```

```
      case 5:
        blue--;  // Transition from magenta to red (decrease blue)
        if (blue == 255) {
          blue = 0;
          state = 0;
        }
        break;
    }
    colour = red << 11 | green << 5 | blue;  // Combine RGB values into 16-bit color
}
```

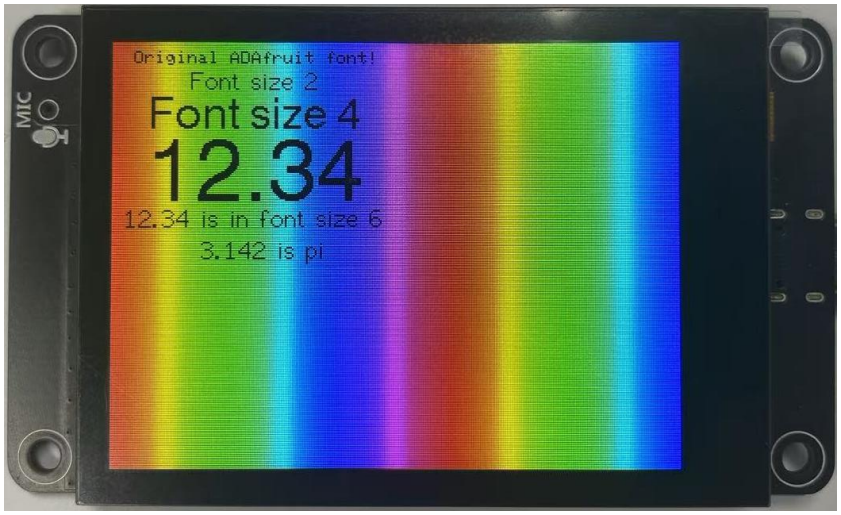
Text display.

```
tft.setTextColor(TFT_BLACK);           // Set text color to black
tft.setCursor(12, 5);                  // Set cursor position
tft.print("Original ADAfruit font!");  // Print simple string
tft.setTextColor(TFT_BLACK, TFT_BLACK);  // Transparent background
tft.drawCentreString("Font size 2", 80, 14, 2);              // Font 2
tft.drawCentreString("Font size 4", 80, 30, 4);              // Font 4
tft.drawCentreString("12.34", 80, 54, 6);                    // Font 6
tft.drawCentreString("12.34 is in font size 6", 80, 92, 2);  // Font 2
```

Click "**Upload**" to upload the code to Freenove_ESP32_S3_Display.



The TFT screen will change colors in the order of red -> green -> blue -> black -> white before displaying text and rainbow effects.



Project 10.2 Flash JPG DMA

Component List

Freenove ESP32 S3 Display x 1 	USB cable x1 

Component Knowledge

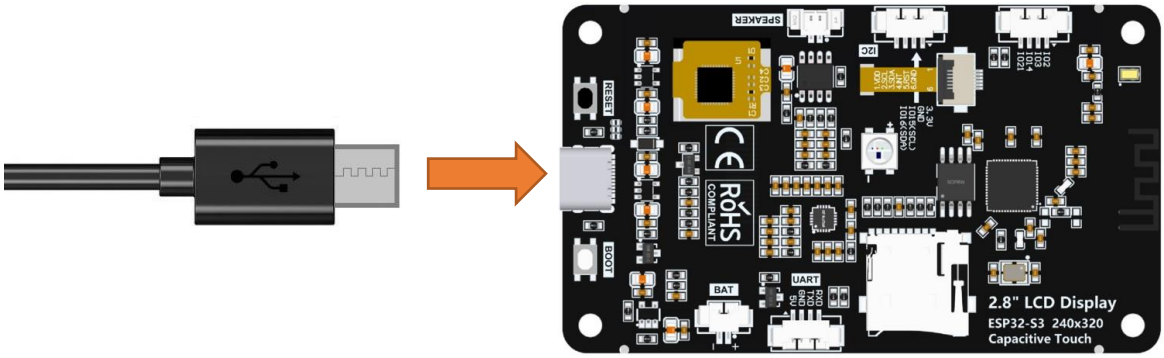
DMA

DMA, or Direct Memory Access, is a hardware feature that allows peripherals to transfer data to and from memory without needing the CPU to be directly involved, dramatically improving overall system efficiency while minimizing processor workload.

The core mechanism of DMA relies on a dedicated DMA controller taking over data transfer tasks. The CPU only needs to initialize the transfer parameters before offloading the operation, allowing computation and I/O operations to proceed in parallel.

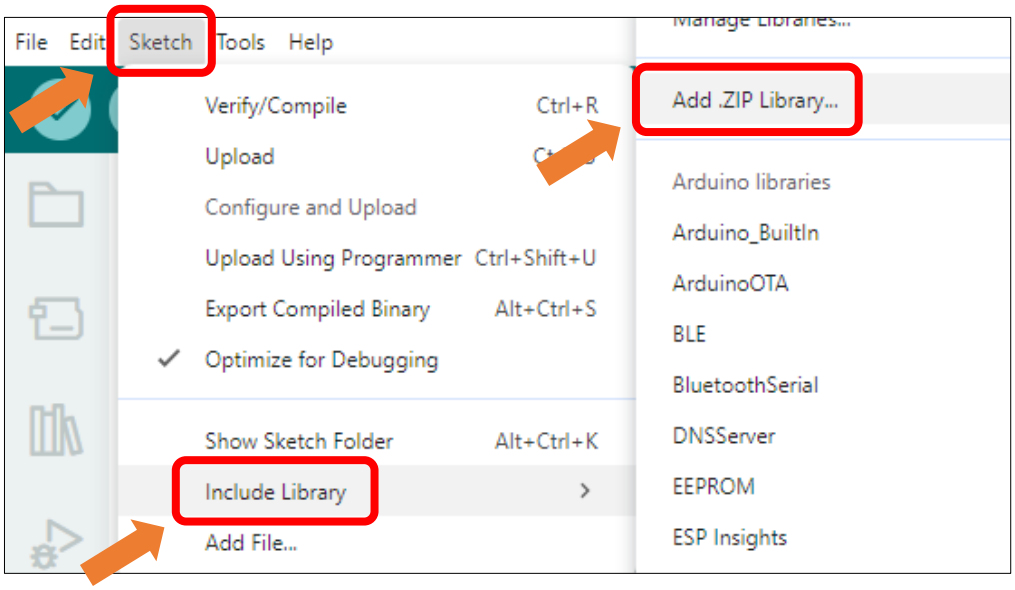
Circuit

Connect Freenove ESP32-S3 to the computer using the USB cable.

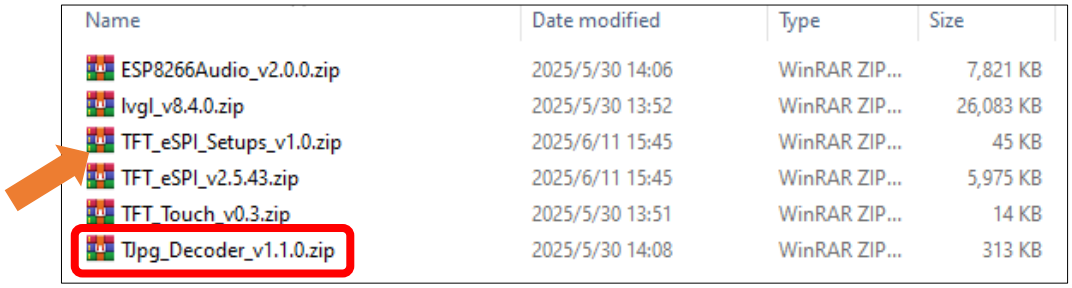


Sketch

Click **Sketch** -> **Include Library** -> **Add .ZIP Library...**



Install **TJpg_Decoder_v1.1.0.zip**



Open"**Sketch_10.2_Flash_Jpg_DMA**" folder under **"Freenove_ESP32_S3_Display\NonTouch\Sketches"** and double-click "**Sketch_10.2_Flash_Jpg_DMA.ino**".

Sketch_10.2_Flash_Jpg_DMA

The following is the program code:

```
1  // Example for library: https://github.com/Bodmer/TJpg_Decoder
2
3  // This example renders a JPEG file stored in Flash memory (see panda.h)
4
5  #include <TFT_eSPI.h>  // TFT display library
6  #include "panda.h"          // Include raw JPEG image data array
7  #include <TJpg_Decoder.h>  // Include JPEG decoder library
8
9  TFT_eSPI tft = TFT_eSPI();  // Create TFT object instance
10
11 // Callback function to draw decoded JPEG blocks on screen
12 bool tft_output(int16_t x, int16_t y, uint16_t w, uint16_t h, uint16_t* bitmap) {
13    if (y >= tft.height()) return 0;  // Stop decoding if off-screen
14    tft.pushImage(x, y, w, h, bitmap);  // Draw without DMA
```

```
15      return 1;  // Continue decoding next block
16  }
17
18  void setup() {
19      Serial.begin(115200);
20      Serial.println("\n\n Testing TJpg_Decoder library");
21
22      tft.begin();  // Initialize TFT display
23      tft.setTextColor(TFT_WHITE, TFT_BLACK);
24      tft.fillScreen(TFT_BLACK);  // Clear screen with black background
25      tft.setRotation(0);
26
27      TJpgDec.setJpgScale(1);           // Set scale factor (1=full size)
28      tft.setSwapBytes(true);           // Match color byte order
29      TJpgDec.setCallback(tft_output);  // Register drawing callback
30  }
31
32  void loop() {
33      uint16_t w = 0, h = 0;
34      TJpgDec.getJpgSize(&w, &h, img_asset, sizeof(img_asset));  // Get image dimensions
35      Serial.print("Width = ");
36      Serial.print(w);
37      Serial.print(", height = ");
38      Serial.print(h);
39
40      uint32_t dt = millis();
41      tft.startWrite();
42      TJpgDec.drawJpg(0, 0, img_asset, sizeof(img_asset));  // Draw the JPEG image
43      tft.endWrite();
44
45      dt = millis() - dt;
46      Serial.print(", dt = ");
47      Serial.print(dt);
48      Serial.println(" ms");
49
50      delay(2000);  // Wait before redraw
51  }
```

Code Explanation

Include necessary header files.

```
1   #include <TFT_eSPI.h>  // TFT display library
2   #include "panda.h"         // Include raw JPEG image data array
3   #include <TJpg_Decoder.h>  // Include JPEG decoder library
```

Create TFT object instance.

```
19  TFT_eSPI tft = TFT_eSPI();  // Create TFT object instance
```

JPEG decoding callback function

```
12  bool tft_output(int16_t x, int16_t y, uint16_t w, uint16_t h, uint16_t* bitmap) {
13      if (y >= tft.height()) return 0;  // Stop decoding if off-screen
14      tft.pushImage(x, y, w, h, bitmap);  // Draw without DMA
15      return 1;  // Continue decoding next block
16  }
```

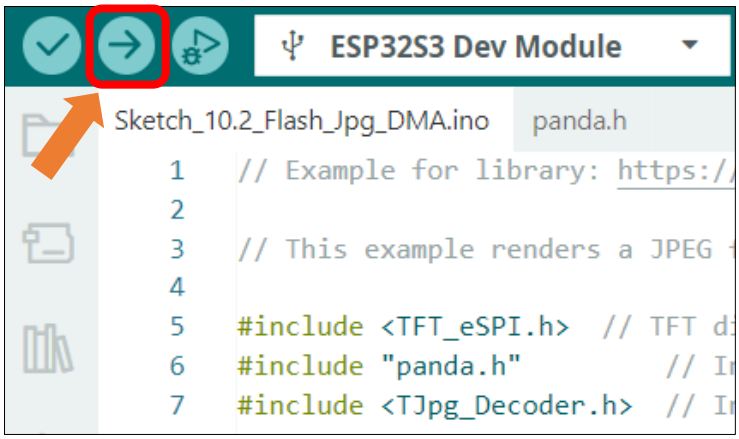
Get JPG size.

```
46  TJpgDec.getJpgSize(&w, &h, img_asset, sizeof(img_asset));  // Get image dimensions
```

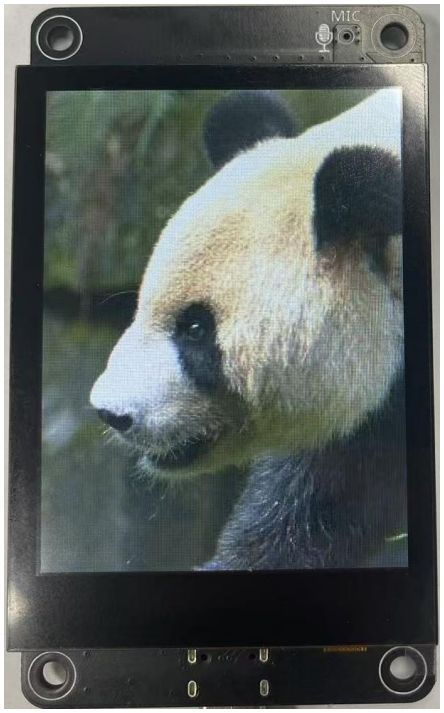
Draw images on the TFT screen.

```
57  tft.startWrite();
58  TJpgDec.drawJpg(0, 0, img_asset, sizeof(img_asset));  // Draw the JPEG image
59  tft.endWrite();
```

Click "**Upload**" to upload the code to Freenove ESP32 Display



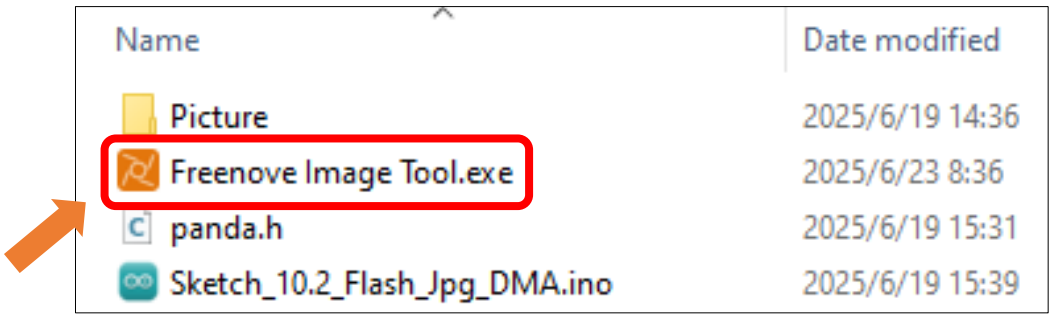
After the code is uploaded, an image will be displayed on the TFT screen. The image is from https://github.com/Bodmer/TJpg_Decoder



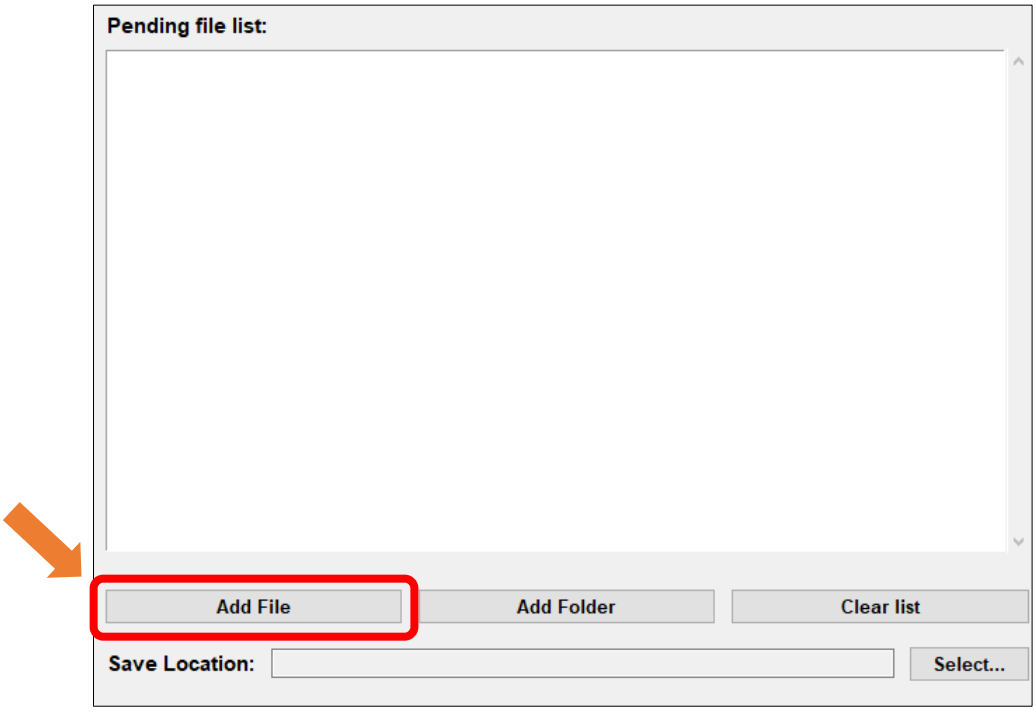
Custom image display

You can customize the image displayed on the display according to your personal preferences.

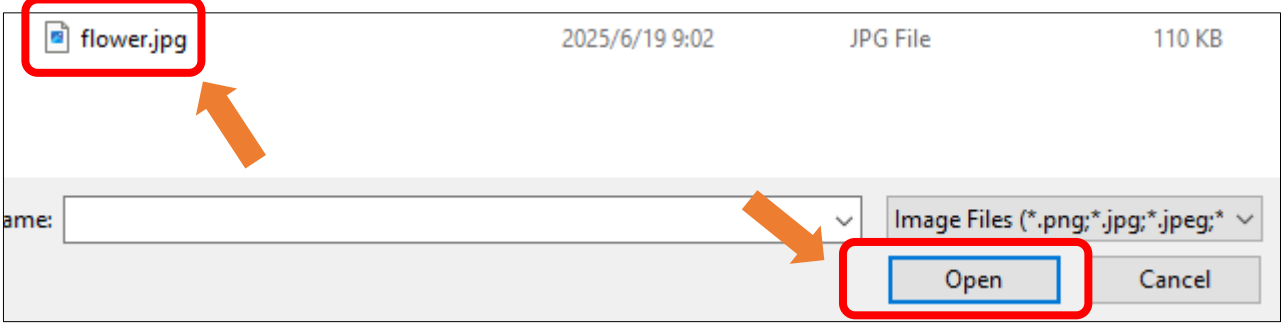
First, open **Freenove_ESP32_S3_Display\NonTouch\Sketches\Sketch_10.2_Flash_Jpg_DMA\Freenove Image Tool.exe**



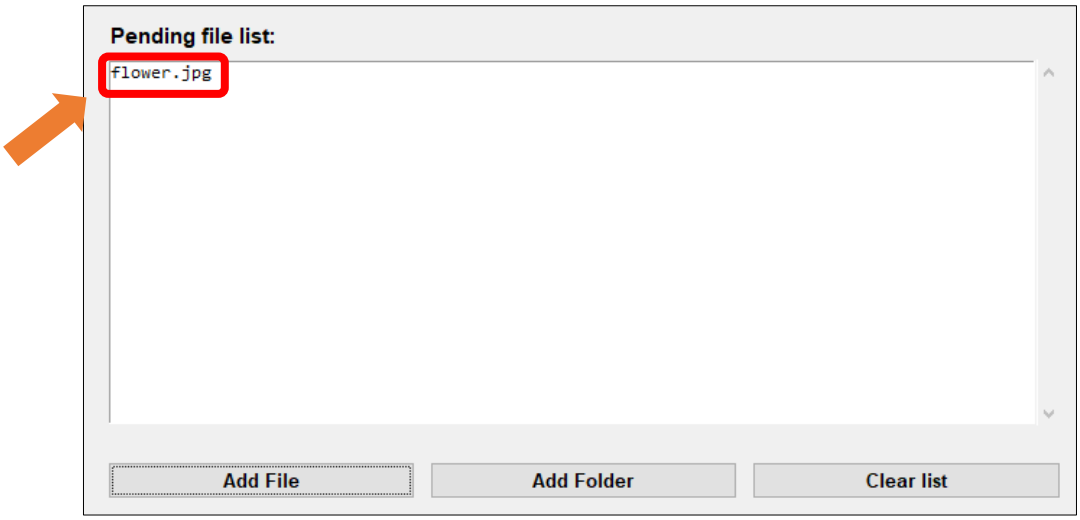
Click **"Add File"**



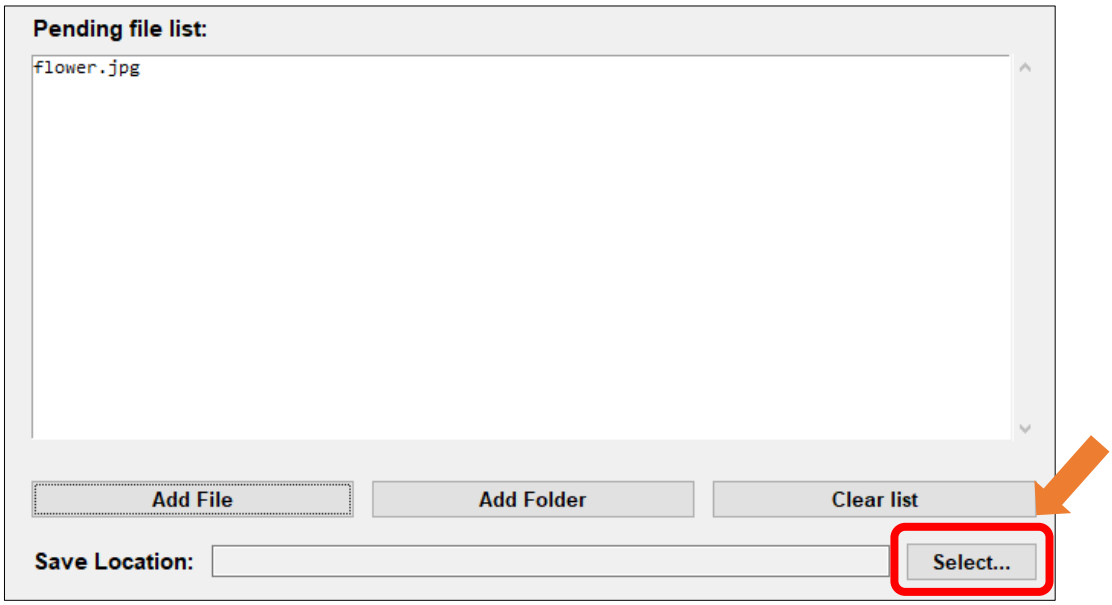
Select any image you like



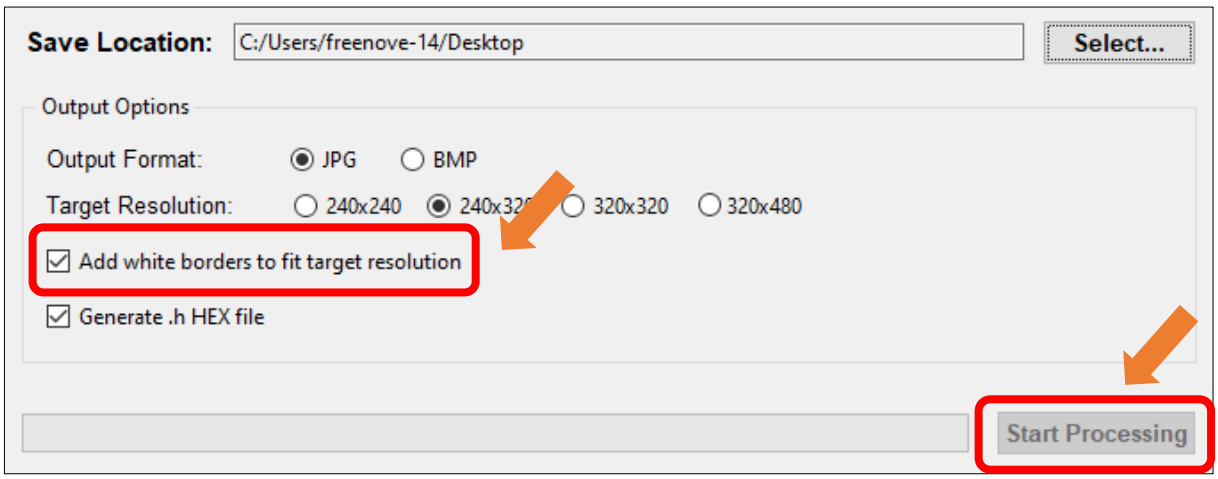
The image files from your folder will now appear in the **Pending File List**.



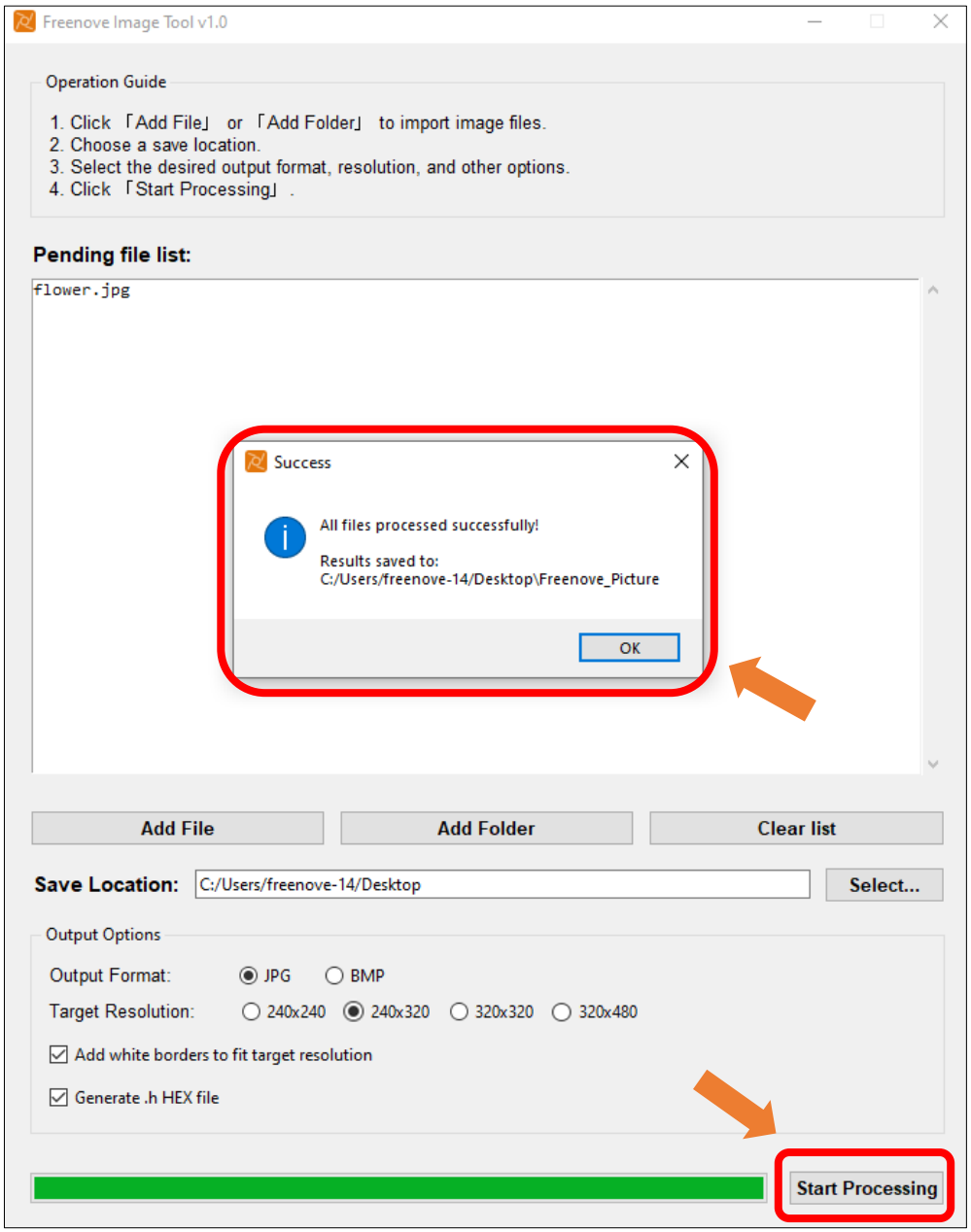
Click "Select…" to change the save location.



The resolution size is selected according to the screen resolution. Check the "**Add white borders to fit target resolution**" and "**Generate .h HEX file**" options.



Click **Start Processing**. Wait for the progress bar to complete and the target folder will be generated.



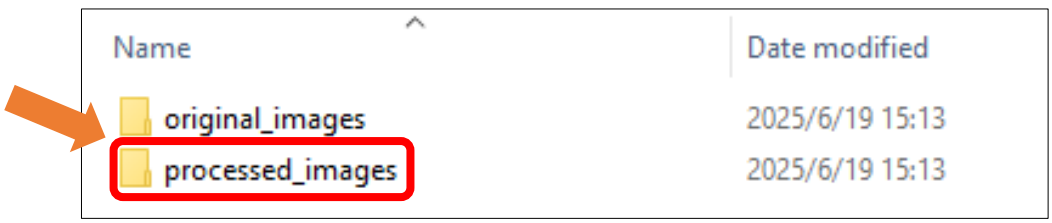
There will be three folders in the generated folder

original_images: backup of images before processing

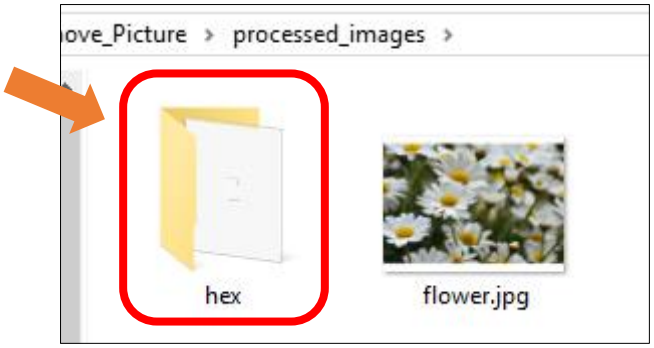
processed_images: processed images

processed_images\hex: generated .h files corresponding to the images

Double click to open **processed_images**



Double click to open **hex**



Replace the entire content of panda.h with the copied data from the .h file, then upload the sketch again.

Sketch_10.2_Flash_Jpg_DMA.ino | panda.h

```
/*
    If you need to customize the displayed image, first use the Freenove Image Tool
    to crop it to the appropriate width and height for your display.

    The tool will automatically adjust the image quality to ensure proper display.

    Paste the generated array into this file, ensuring the correct array format:

        const uint8_t name[] PROGMEM = {

    to start and and end with:

    };
*/

const unsigned char img_asset[] PROGMEM = {
    0xFF, 0xD8, 0xFF, 0xE0, 0x00, 0x10, 0x4A, 0x46, 0x49, 0x46, 0x00, 0x01,
    0x01, 0x00, 0x00, 0x01, 0x00, 0x01, 0x00, 0x00, 0xFF, 0xDB, 0x00, 0x43,
    0x00, 0x05, 0x03, 0x04, 0x04, 0x04, 0x03, 0x05, 0x04, 0x04, 0x04, 0x05,
    0x05, 0x05, 0x06, 0x07, 0x0C, 0x08, 0x07, 0x07, 0x07, 0x07, 0x0F, 0x0B,
    0x0B, 0x09, 0x0C, 0x11, 0x0F, 0x12, 0x12, 0x11, 0x0F, 0x11, 0x11, 0x13,
    0x16, 0x1C, 0x17, 0x13, 0x14, 0x1A, 0x15, 0x11, 0x11, 0x18, 0x21, 0x18,
    0x1A, 0x1D, 0x1D, 0x1F, 0x1F, 0x1F, 0x13, 0x17, 0x22, 0x24, 0x22, 0x1E,
    0x24, 0x1C, 0x1E, 0x1F, 0x1E, 0xFF, 0xDB, 0x00, 0x43, 0x01, 0x05, 0x05,
    0x05, 0x07, 0x06, 0x07, 0x0E, 0x08, 0x08, 0x0E, 0x1E, 0x14, 0x11, 0x14,
    0x1E, 0x1E, 0x1E, 0x1E, 0x1E, 0x1E, 0x1E, 0x1E, 0x1E, 0x1E, 0x1E, 0x1E,
    0x1E, 0x1E, 0x1E, 0x1E, 0x1E, 0x1E, 0x1E, 0x1E, 0x1E, 0x1E, 0x1E, 0x1E,
    0x1E, 0x1E, 0x1E, 0x1E, 0x1E, 0x1E, 0x1E, 0x1E, 0x1E, 0x1E, 0x1E, 0x1E,
    0x1E, 0x1E, 0x1E, 0x1E, 0x1E, 0x1E, 0x1E, 0x1E, 0x1E, 0x1E, 0x1E, 0x1E,
    0x1E, 0x1E, 0xFF, 0xC0, 0x00, 0x11, 0x08, 0x01, 0x40, 0x01, 0xE0, 0x03,
    0x01, 0x22, 0x00, 0x02, 0x11, 0x01, 0x03, 0x11, 0x01, 0xFF, 0xC4, 0x00,
    0x1C, 0x00, 0x00, 0x02, 0x02, 0x03, 0x01, 0x01, 0x00, 0x00, 0x00, 0x00,
```

The image will display on the screen.

Note:

To adjust the image display orientation, modify the following code in Sketch_10.2_Flash_Jpg_DMA.ino:

```
42  tft.setRotation(uint8_t rotation);
```

Value	Rotation Angle	Description
0	0°	Default orientation (no rotation)
1	90°	Clockwise 90-degree rotation
2	180°	Clockwise 180-degree rotation
3	270°	Clockwise 270-degree rotation

Chapter 11 LVGL

Project 11.1 LVGL

Component List

Freenove ESP32 S3 Display x 1 	USB cable x1 

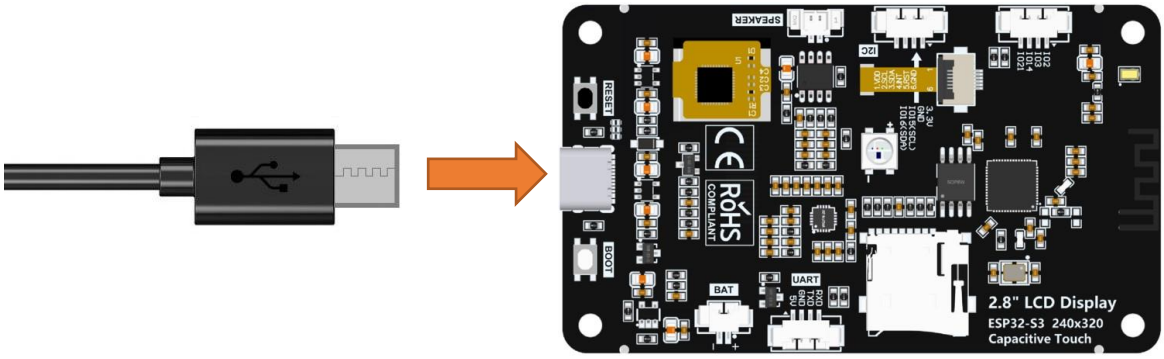
Component Knowledge

LVGL is a widely-used embedded GUI library that is implemented in pure C, making it highly portable and performant. It offers rich features and content, supporting both display and input devices such as touchscreens and keyboards.

	Features supported by LVGL
1	Powerful building blocks such as buttons, charts, lists, sliders, images, and more.
2	Advanced graphics with animation, anti-aliasing, opacity, and smooth scrolling.
3	Various input devices, such as touchpads, mice, keyboards, encoders, and more.
4	Multiple languages with UTF-8 encoding.
5	Multiple display types, including TFT and monochrome displays.
6	Fully customizable graphical elements.
7	LVGL can be used independently of any microcontroller or display hardware.
8	Highly extensible and can be configured to use very little memory (e.g. 64 kB of flash and 16 kB of RAM)
9	It can be used with or without an operating system, and supports external memory and GPUs as optional features.
10	Single-frame buffer operation, even with advanced graphics effects.
11	Written in C language to achieve maximum compatibility (compatible with C++ as well).
12	LVGL has a simulator that allows for embedded GUI design on a PC without any embedded hardware.
13	Resources to help developers quickly get started with the library, including tutorials, examples, and themes.
14	A wide range of resources.

Circuit

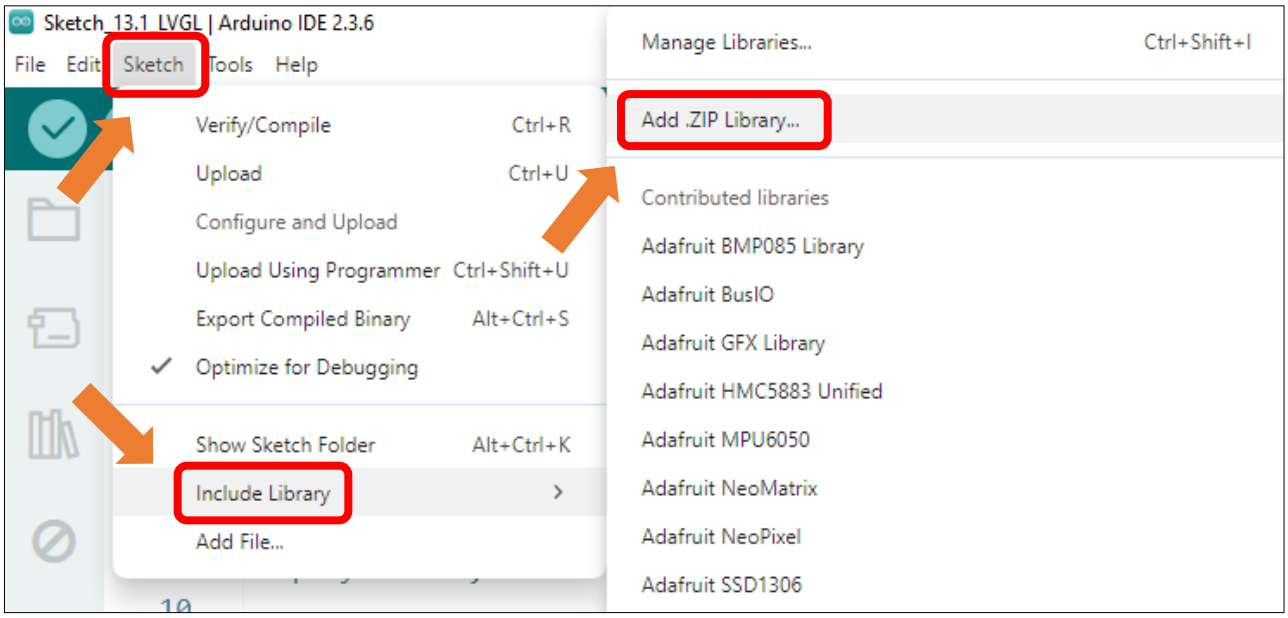
Connect Freenove ESP32 -S3 to the computer using the USB cable.



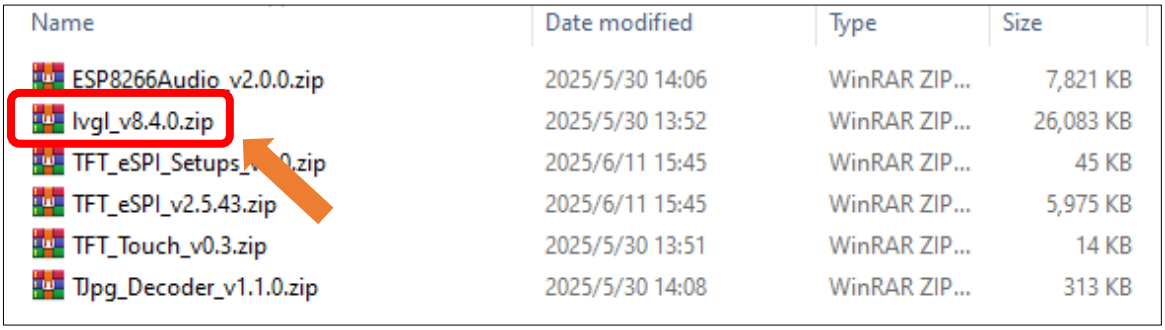
Sketch

Install Libraries

Click **Sketch** -> **Include Library** -> **Add .ZIP Library...**



Install **lvgl_v8.4.0.zip**



Open "**Sketch_11.1_LVGL**" folder under **"Freenove_ESP32_S3_Display\NonTouch\Sketches"** and double-click "**Sketch_11.1_LVGL.ino**".

Sketch_11.1_LVGL

The following is the program code:

```
/*
* @ File:   Sketch_11.1_LVGL.ino
* @ Author: [Zhentao Lin]
* @ Date:   [2025-06-20]
*/

#include "display.h"

Display screen;

void setup()
{
    /* prepare for possible serial debug */
    Serial.begin( 115200 );

    /*** Init screen ***/
    screen.init();

    String LVGL_Arduino = "Hello Arduino! ";
    LVGL_Arduino += String('V') + lv_version_major() + "." + lv_version_minor() + "." +
lv_version_patch();

    Serial.println( LVGL_Arduino );
    Serial.println( "I am LVGL_Arduino" );

    /* Create simple label */
    lv_obj_t *label = lv_label_create( lv_scr_act() );
    lv_label_set_text( label, LVGL_Arduino.c_str() );
    lv_obj_align( label, LV_ALIGN_CENTER, 0, 0 );

    Serial.println( "Setup done" );
}

void loop()
{
    screen.routine(); /* let the GUI do its work */
    delay( 5 );
}
```

Code Explanation

Include the header file.

```
7  #include "display.h"
```

Set the baud rate to 115200

```
19  Serial.begin( 115200 );
```

Initialize the screen.

```
11  screen.init();
```

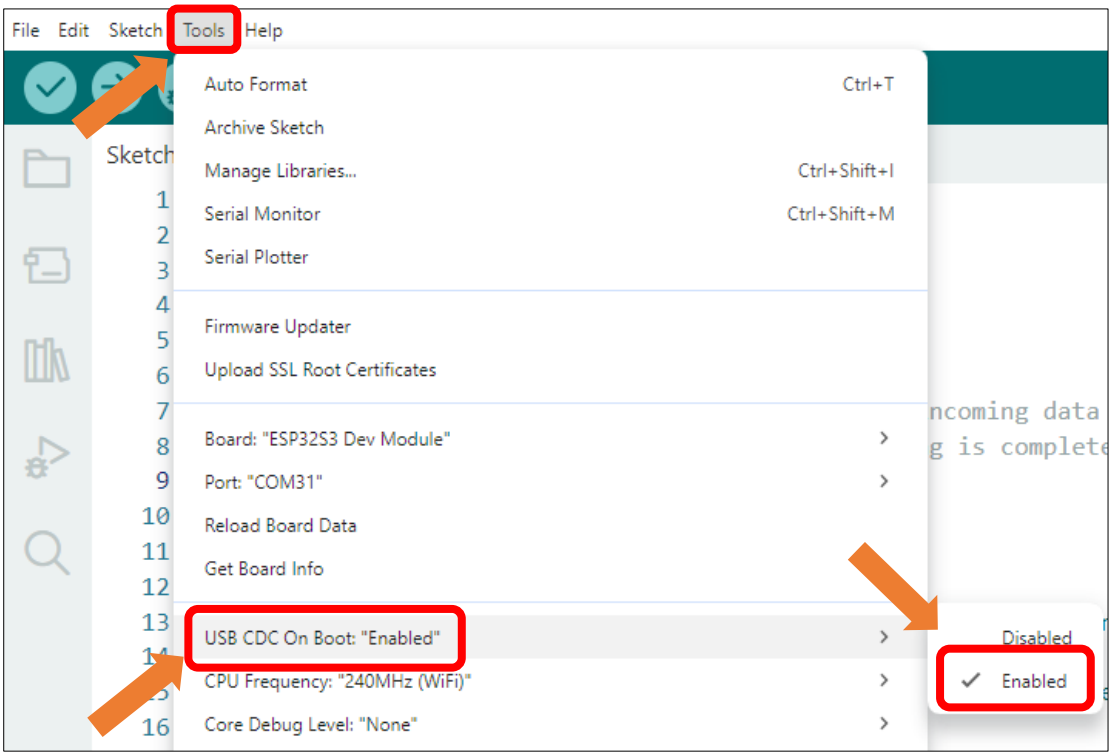
Configure the screen interface.

```
15  String LVGL_Arduino = "Hello Arduino! ";
16  LVGL_Arduino += String('V') + lv_version_major() + "." + lv_version_minor() + "." +
17  lv_version_patch();
18
19  Serial.println( LVGL_Arduino );
20  Serial.println( "I am LVGL_Arduino" );
21
22  /* Create simple label */
23  lv_obj_t *label = lv_label_create( lv_scr_act() );
24  lv_label_set_text( label, LVGL_Arduino.c_str() );
25  lv_obj_align( label, LV_ALIGN_CENTER, 0, 0 );
```

Let the LVGL handle the tasks.

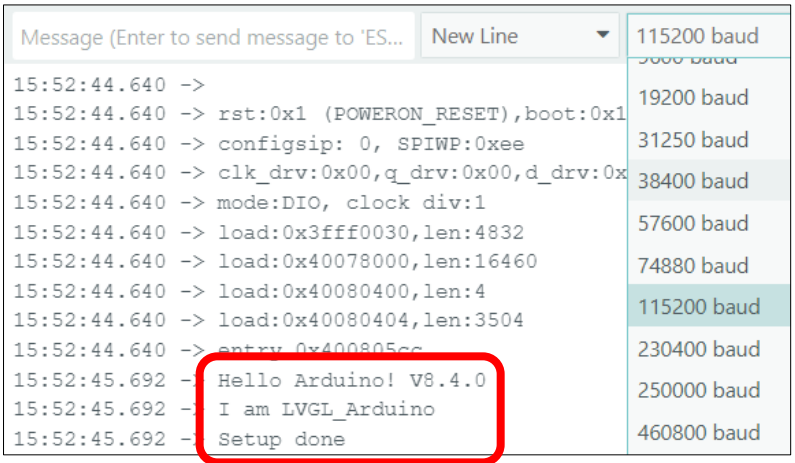
```
46  screen.routine(); /* let the GUI do its work */
```

Enable the "USB CDC On Boot" feature.

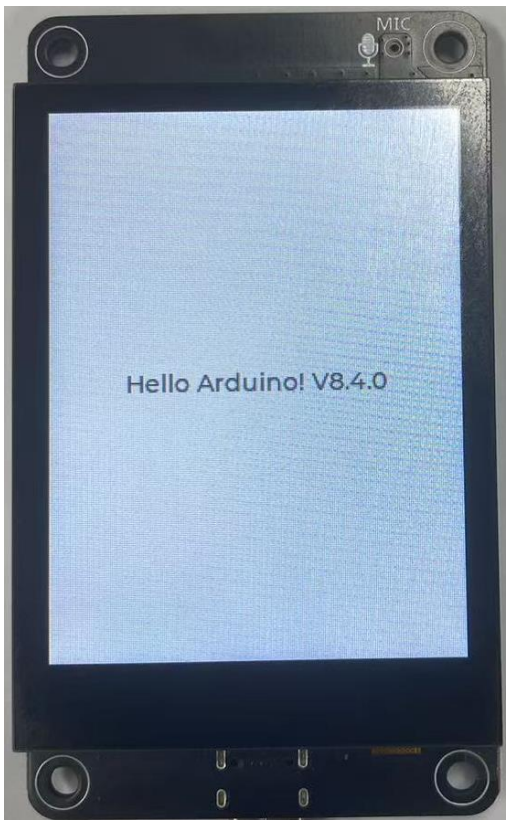


Click "**Upload**" to upload the code to Freenove ESP32 Display. Set the baud rate to 115200.

Data will be printed on the serial monitor.



The text "**Hello Arduino! V8.4.0**" will be displayed on the screen.



Chapter 12 LVGL Arduino

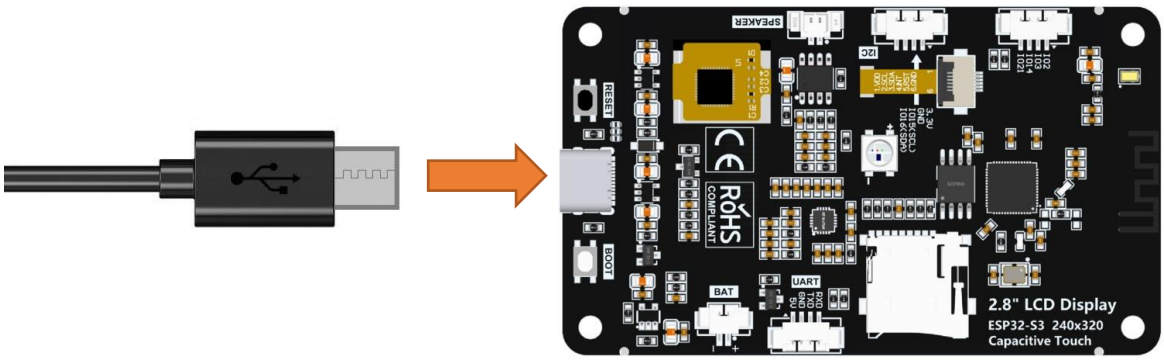
Project 12.1 LVGL Arduino

Component List

Freenove ESP32 S3 Display x 1 	USB cable x1 

Circuit

Connect Freenove ESP32-S3 to the computer using the USB cable.



Sketch

Open“**Sketch_12.1_Lvgl_Arduino**” folder under **“Freenove_ESP32_S3_Display\NonTouch\Sketches”** and double-click “**Sketch_12.1_Lvgl_Arduino.ino**”.

Sketch_12.1_Lvgl_Arduino

The following is the program code:

```
1  /*Using LVGL with Arduino requires some extra steps:
2    *Be sure to read the docs here: https://docs.lvgl.io/master/get-
3  started/platforms/arduino.html  */
4
```

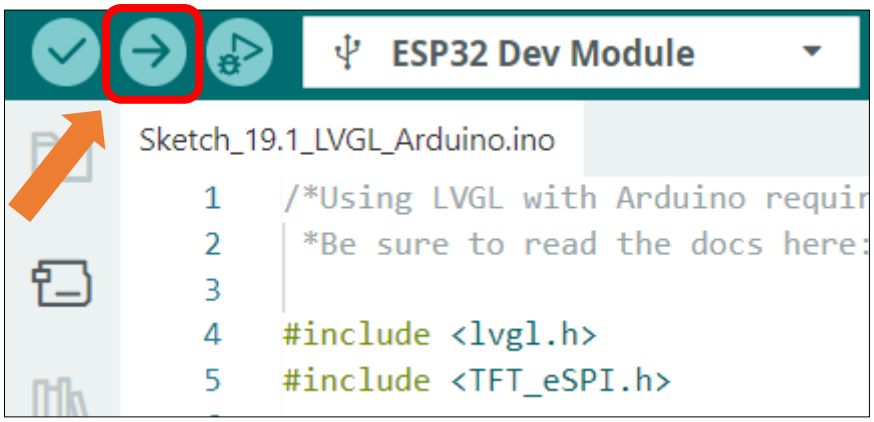
```
#include <lvgl.h>
#include "demos/lv_demos.h"
#include <TFT_eSPI.h>
#include "FT6336U.h"

/*To use the built-in examples and demos of LVGL uncomment the includes below respectively.
  *You also need to copy `lvgl/examples` to `lvgl/src/examples`. Similarly for the demos `lvgl/demos` to `lvgl/src/demos`.
  Note that the `lv_examples` library is for LVGL v7 and you shouldn't install it for this version (since LVGL v8)
  as the examples and demos are now part of the main LVGL library. */

/*Change to your screen resolution*/
#define TFT_DIRECTION 0   //Select TFT Direction (0 - 3)
#define I2C_SCL 15
#define I2C_SDA 16
#define INT_N_PIN 17
#define RST_N_PIN 18
#define TFT_SCREEN_WIDTH 240
#define TFT_SCREEN_HEIGHT 320

static const uint16_t screenWidth = TFT_SCREEN_WIDTH;
static const uint16_t screenHeight = TFT_SCREEN_HEIGHT;
static const uint16_t screenHeightBuf = TFT_SCREEN_HEIGHT / 10;
static lv_disp_draw_buf_t draw_buf;
static lv_color_t draw_buf1[screenWidth * screenHeightBuf];

TFT_eSPI tft = TFT_eSPI(screenWidth, screenHeight); /* TFT instance */
FT6336U ft6336u(I2C_SDA, I2C_SCL, RST_N_PIN, INT_N_PIN);
FT6336U_TouchPointType tp;

#if LV_USE_LOG != 0
/* Serial debugging */
void my_print(const char * buf)
{
    Serial.printf(buf);
    Serial.flush();
}
#endif

/* Display flushing */
void my_disp_flush( lv_disp_drv_t *disp_drv, const lv_area_t *area, lv_color_t *color_p )
{
    uint32_t w = ( area->x2 - area->x1 + 1 );
```

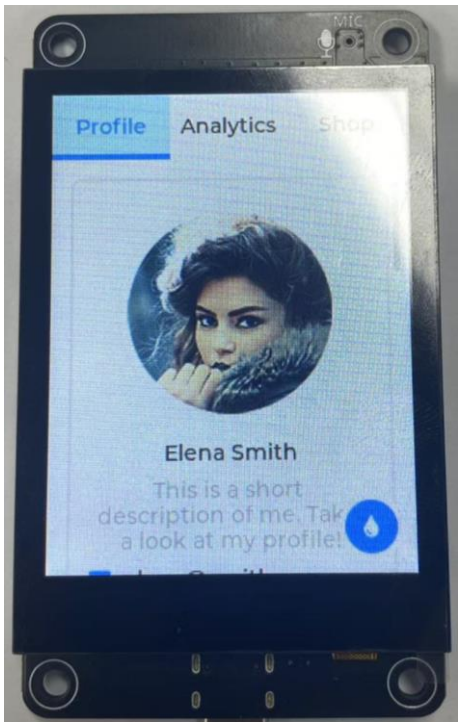
```
    uint32_t h = ( area->y2 - area->y1 + 1 );

    tft.startWrite();
    tft.setAddrWindow( area->x1, area->y1, w, h );
    tft.pushColors( ( uint16_t * )&color_p->full, w * h, true );
    tft.endWrite();

    lv_disp_flush_ready( disp_drv );
}

void my_touchpad_read( lv_indev_drv_t * indev_driver, lv_indev_data_t * data )
{
    uint16_t touchX, touchY;

    //bool touched = tft.getTouch( &touchX, &touchY, 600 );
    tp = ft6336u.scan();
    int touched = tp.touch_count;

    if( !touched )
    {
        data->state = LV_INDEV_STATE_REL;
    }
    else
    {
        int x = tp.tp[0].x;
        int y = tp.tp[0].y;
        if(x >= 0 && x < screenWidth && y >= 0 && y < screenHeight)
        {
            data->state = LV_INDEV_STATE_PR;
            data->point.x = tp.tp[0].x;
            data->point.y = tp.tp[0].y;
        }
    }
}

void setup()
{
    Serial.begin( 115200 ); /* prepare for possible serial debug */

    String LVGL_Arduino = "Hello Arduino! ";
    LVGL_Arduino += String('V') + lv_version_major() + "." + lv_version_minor() + "." +
lv_version_patch();

    Serial.println( LVGL_Arduino );
```

```
93      Serial.println( "I am LVGL_Arduino" );
94
95      tft.begin();          /* TFT init */
96      tft.setRotation(TFT_DIRECTION);
97      ft6336u.begin();
98
99      lv_init();
100
101 #if LV_USE_LOG != 0
102     lv_log_register_print_cb( my_print ); /* register print function for debugging */
103 #endif
104
105     lv_disp_draw_buf_init(&draw_buf, draw_buf1, NULL, screenWidth * screenHeightBuf);
106
107     /*Initialize the display*/
108     static lv_disp_drv_t disp_drv;
109     lv_disp_drv_init( &disp_drv );
110     /*Change the following line to your display resolution*/
111     disp_drv.hor_res = screenWidth;
112     disp_drv.ver_res = screenHeight;
113     disp_drv.flush_cb = my_disp_flush;
114     disp_drv.draw_buf = &draw_buf;
115     lv_disp_drv_register( &disp_drv );
116
117     /*Initialize the (dummy) input device driver*/
118     static lv_indev_drv_t indev_drv;
119     lv_indev_drv_init( &indev_drv );
120     indev_drv.type = LV_INDEV_TYPE_POINTER;
121     indev_drv.read_cb = my_touchpad_read;
122     lv_indev_drv_register( &indev_drv );
123
124     /* Create simple label */
125     // lv_obj_t *label = lv_label_create( lv_scr_act() );
126     // lv_label_set_text( label, "Hello Ardino and LVGL!");
127     // lv_obj_align( label, LV_ALIGN_CENTER, 0, 0 );
128
129     /* Try an example. See all the examples
130      * online: https://docs.lvgl.io/master/examples.html
131      * source codes:
132 https://github.com/lvgl/lvgl/tree/e7f88efa5853128bf871dde335c0ca8da9eb7731/examples */
133      //lv_example_btn_1();
134
135      /*Or try out a demo. Don't forget to enable the demos in lv_conf.h. E.g.
136 LV_USE_DEMOS_WIDGETS*/
```

```
137       lv_demo_widgets();
138       // lv_demo_benchmark();
139       // lv_demo_keypad_encoder();
140       // lv_demo_music();
141       // lv_demo_printer();
142       // lv_demo_stress();
143
144       Serial.println( "Setup done" );
145   }
146
147   void loop()
148   {
149       lv_timer_handler(); /* let the GUI do its work */
150       delay( 5 );
151   }
```

Click "**Upload**" to upload the code to Freenove ESP32 Display





What's next?

Thank you again for choosing Freenove products.

THANK YOU for participating in this learning experience!

We have reached the end of this Tutorial. If you find errors, omissions, or you have suggestions and/or questions about the Tutorial or component contents of this Kit, please feel free to contact us:
support@freenove.com

We will make every effort to make changes and correct errors as soon as feasibly possible and publish a revised version.

If you want to learn more about Arduino, ESP32, Raspberry Pi Pico W, Raspberry Pi, Smart Cars, Robotics and other interesting products in science and technology, please continue to visit our website. We will continue to launch fun, cost effective, innovative and exciting products.

Thank you again for choosing Freenove products.