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_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

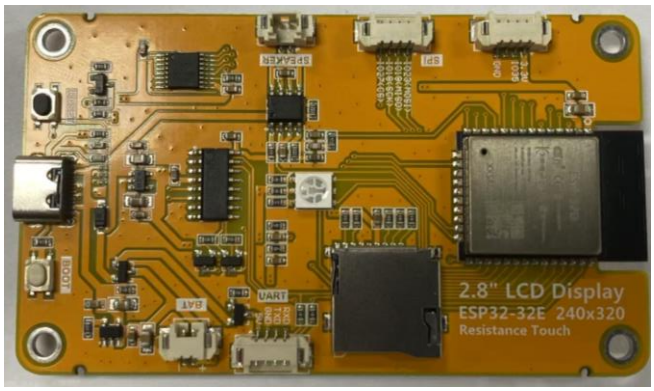
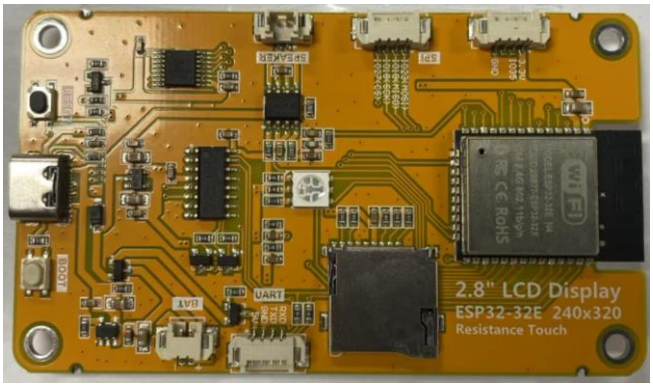
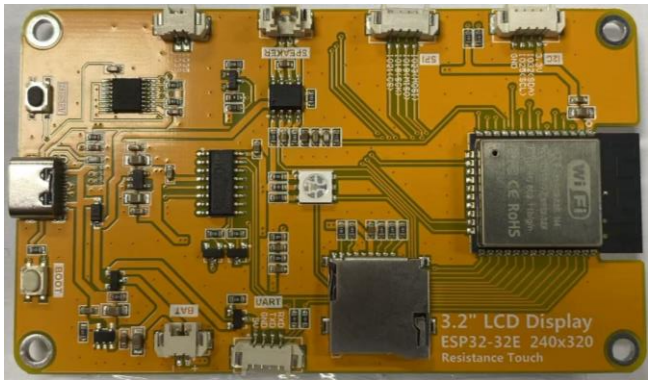
Copyright

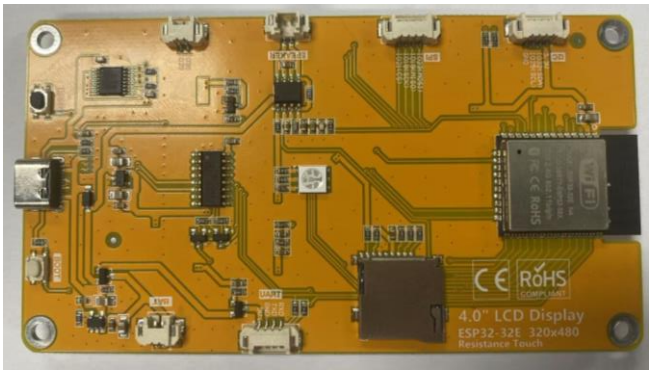
Contents

Freenove ESP32 Display

Freenove ESP32 Display Models

At the time this tutorial was written, Freenove ESP32 Display is available in five different models. While they may vary in drivers, resolution, or screen size, this guide applies to all of them. Below is a list of the supported models, along with corresponding images:

Models	Specifications		Images
FNK0103F_2P8	Size	2.8 inch	
	Resolution	240x320	
	Driver	ILI9341	
FNK0103B_2P8	Size	2.8inch	
	Resolution	240x320	
	Driver	ST7789	
FNK0103L_3P2	Size	3.2inch	
	Resolution	240x320	
	Driver	ST7789	

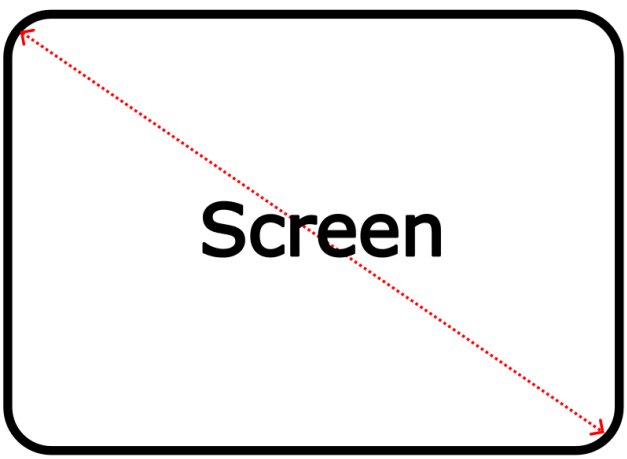
FNK0103N_3P5	Size	3.5inch	
	Resolution	320x480	
	Driver	ST7796	
FNK0103S_4P0	Size	4.0inch	
	Resolution	320x480	
	Driver	ST7796	

Related Knowledge

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.

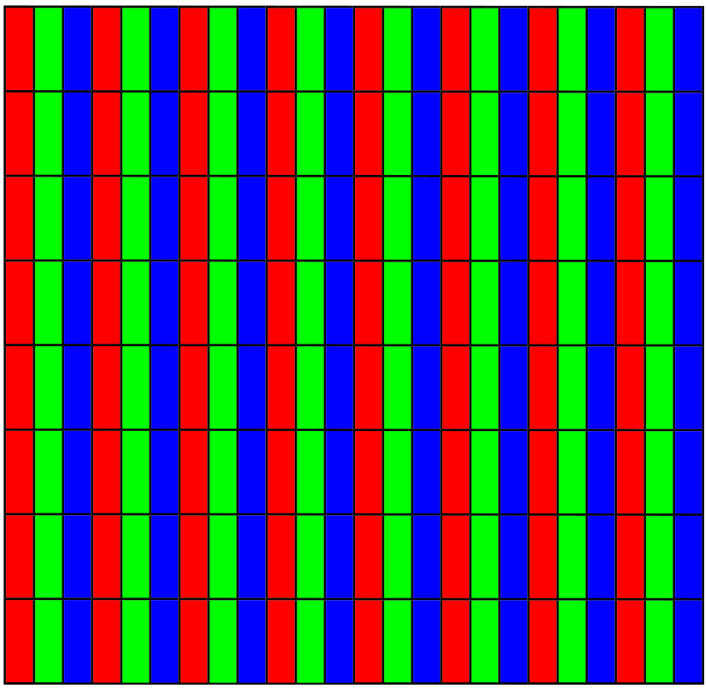
A critical clarification: screen size always refers to **the diagonal measurement of the display** panel. For instance, a 4-inch display features a 4-inch (10.16 cm) diagonal, ensuring standardized and comparable sizing across all devices.



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.



Display Driver

The display driver IC serves as the core component of display panels, processing signals from the main controller and controlling pixel illumination. Below is an overview of three widely used display driver ICs.

Display Driver IC	Resolution	Advantages
ILI9341	240x320	widely used with abundant resources available
ST7789	240x320 or 240x240	Suitable for learning and development
ST7796	320x480	High-definition display with richer colors

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 
Stylus x 1 	Jumper Wire x 1 

Preface

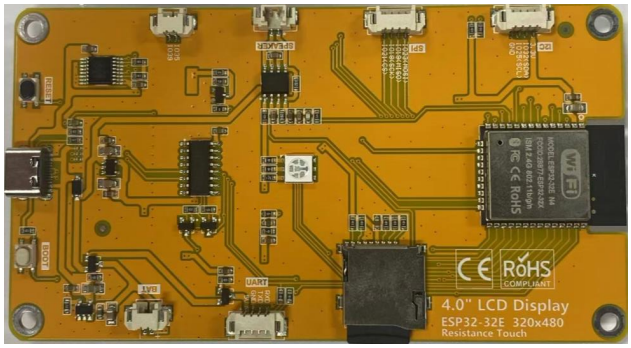
ESP32-WROOM

The ESP32-WROOM offers two antenna options: the PCB on-board antenna and the IPEX antenna.

- The PCB on-board antenna is an integrated antenna within the chip module itself, making it compact and convenient for both portability and design.
- The IPEX antenna is an external metal antenna connected to the module's integrated antenna, providing enhanced signal performance.

PCB on-board antenna	IPEX antenna
	

The ESP32-WROOM of this product is based on the ESP32-WROOM-32E module with built-in PCB on-board antenna.

Top	Bottom
	

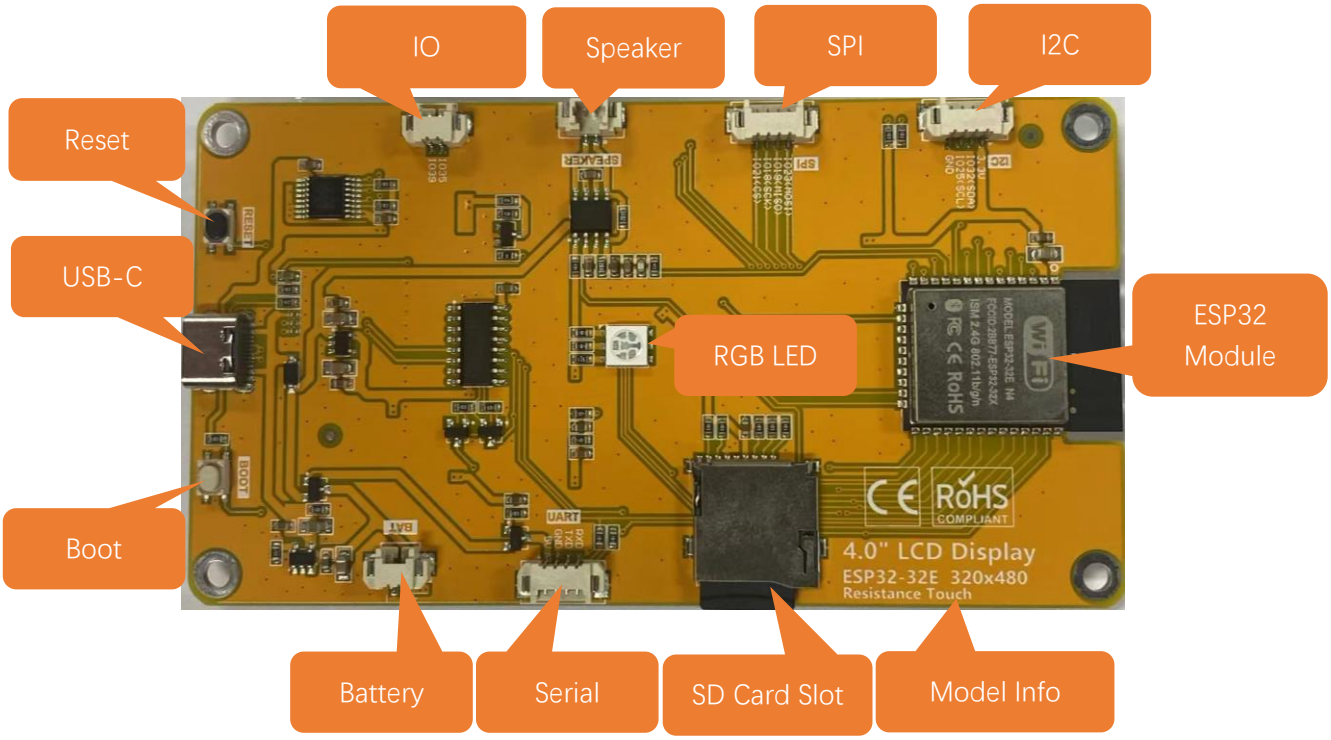
Freenove ESP32 Display is available in five different models. While they may vary in drivers, resolution, or screen size, this guide applies to all of them. For detailed model specifications, please click here.

For datasheet of the ESP32 module, please visit:

https://www.espressif.com.cn/sites/default/files/documentation/esp32-wroom-32e_esp32-wroom-32ue_datasheet_en.pdf

Freenove ESP32 Display

Hardware Interfaces



Note: The 2.8-inch version does not provide access to the I2C interface or IO39 pin.

Battery (Optional)

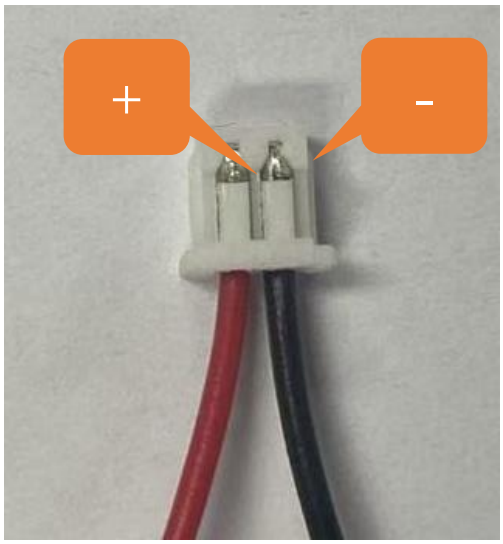
Please note that this product does not come with lithium batteries; please purchase them yourself.

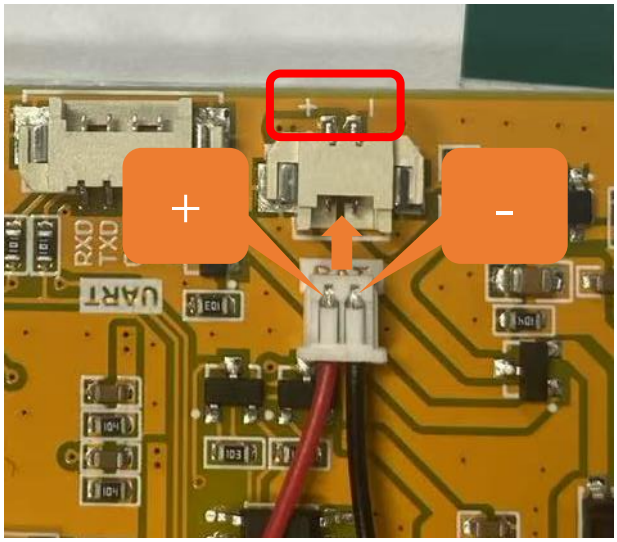
This device supports both **USB-powered and lithium battery-powered operation**. For optimal safety, USB power is recommended. Due to the **hazardous nature of lithium batteries**, we advise against their use unless absolutely necessary.

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.

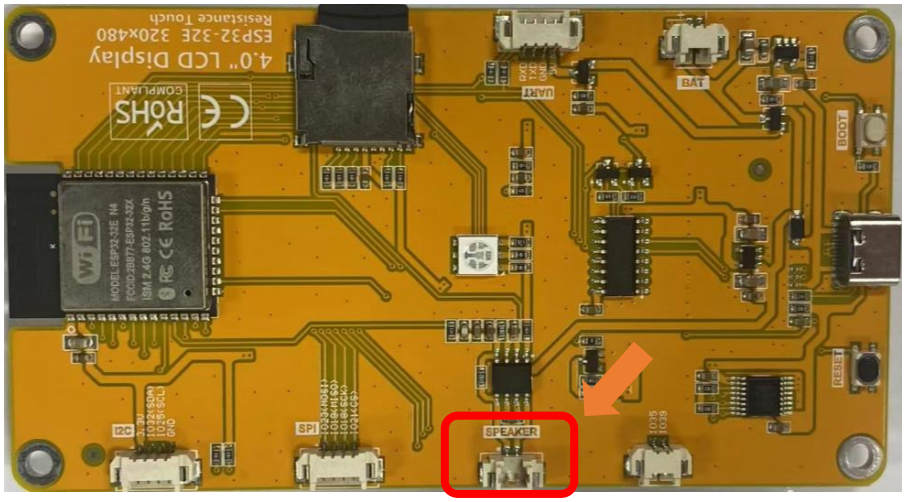


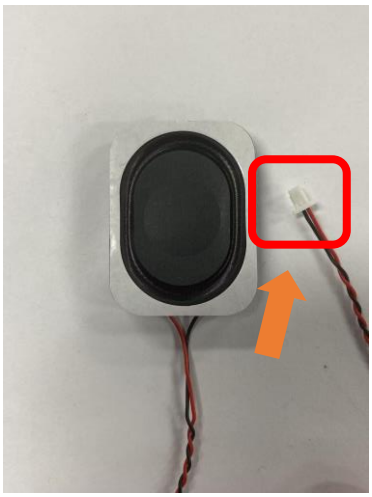


We recommend using a charger specially designed for lithium batteries. Due to various specifications and quality of lithium batteries, using a proper charger helps ensure peak performance, safety, and battery longevity.
While our product also supports USB charging as a backup option, please note that 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 Display. It is recommended to use an **8Ω 1W speaker**.





Please note that this kit does not include a speaker. Please buy one yourself.

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.

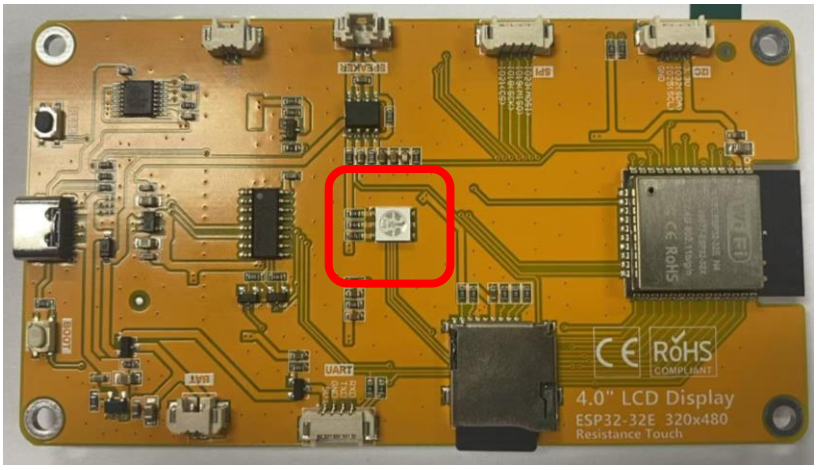
TFT Screen

Freenove ESP32 Display is available in five different models with various TFT screen. This guide applies to all of them. For detailed model specifications, please click here.

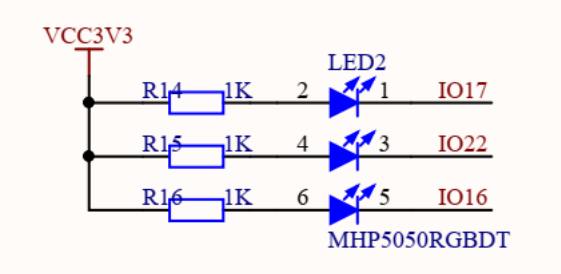
Item	Pins	Definition
TFT Screen	GPIO13	LCD_MOSI
	GPIO12	LCD_MISO
	GPIO14	LCD_SCK
	GPIO2	LCD_RS
	GPIO15	LCD_CS

RGB LED

The Freenove ESP32 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

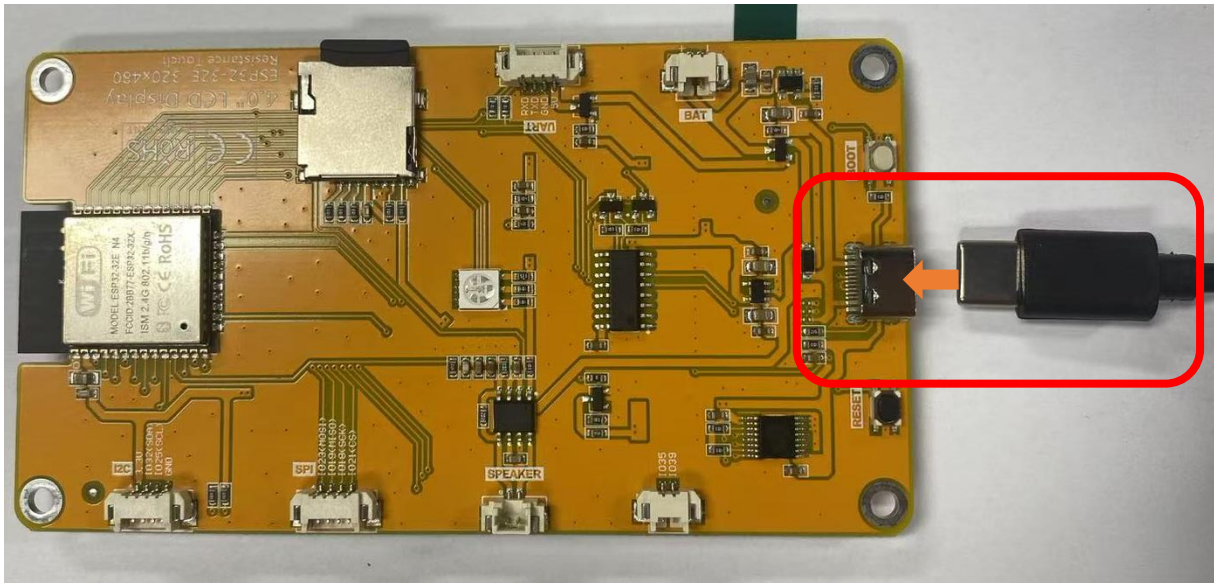
CH340 (Required)

ESP32-WROOM uses CH340 to download code. Therefore, before using the device, it is necessary to install the CH343 driver on your computer.

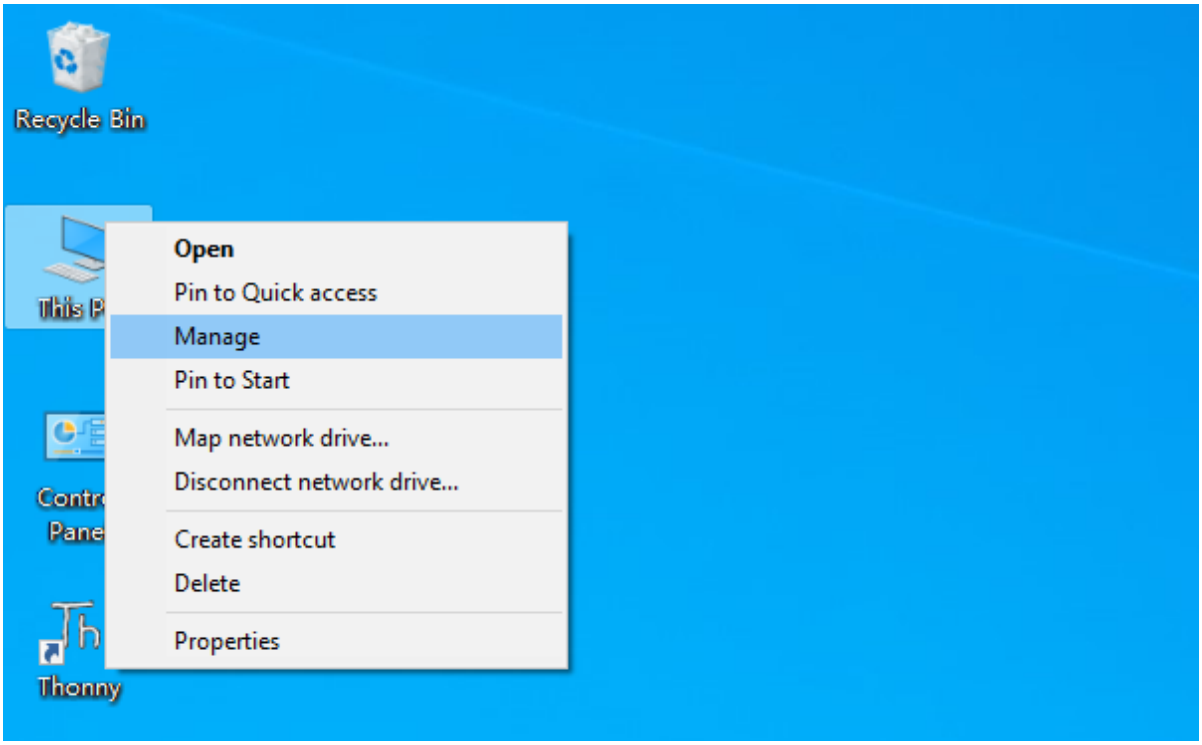
Windows

Check whether CH343 has been installed

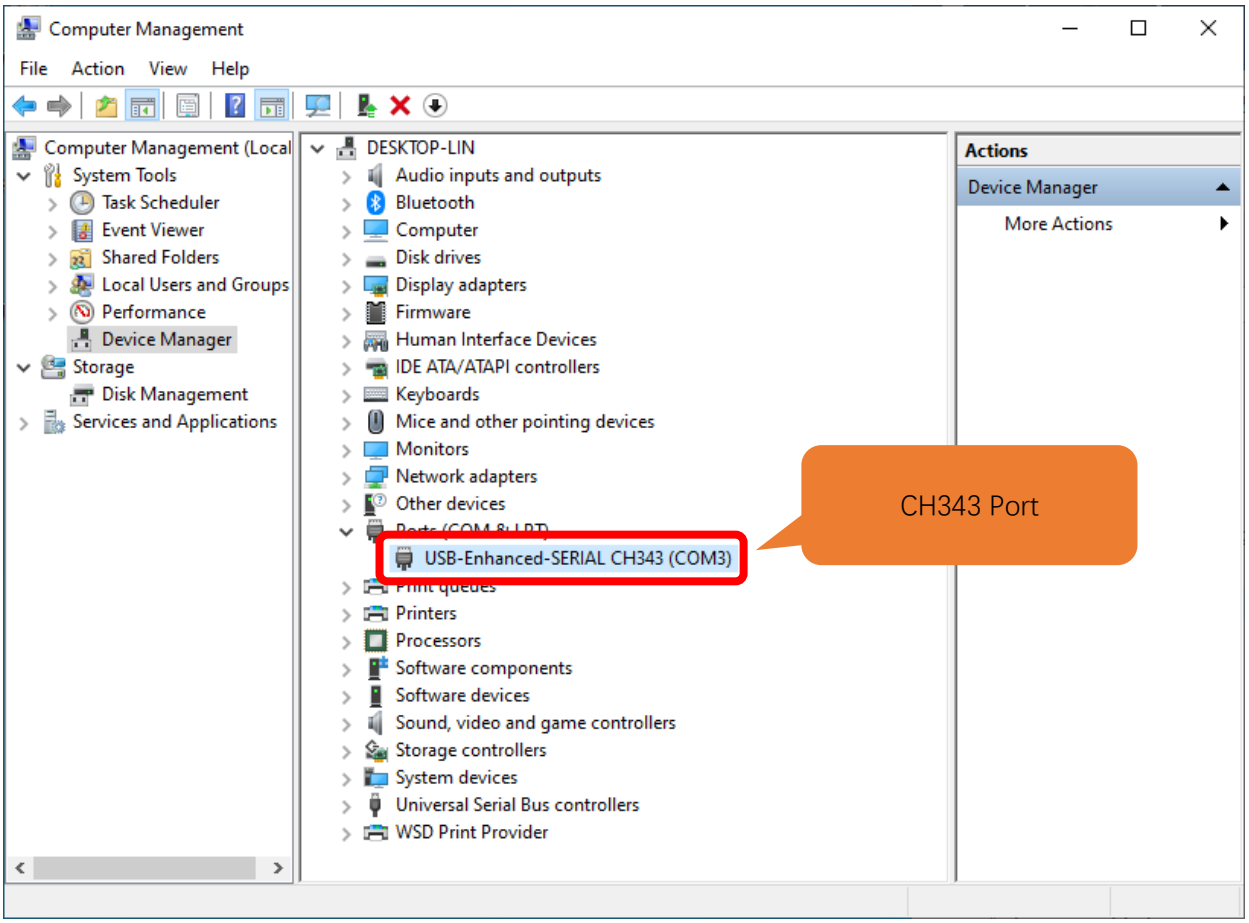
1. Connect your computer and ESP32-WROOM with a USB cable.



2. Turn to the main interface of your computer, select "**This PC**" and right-click to select "**Manage**".

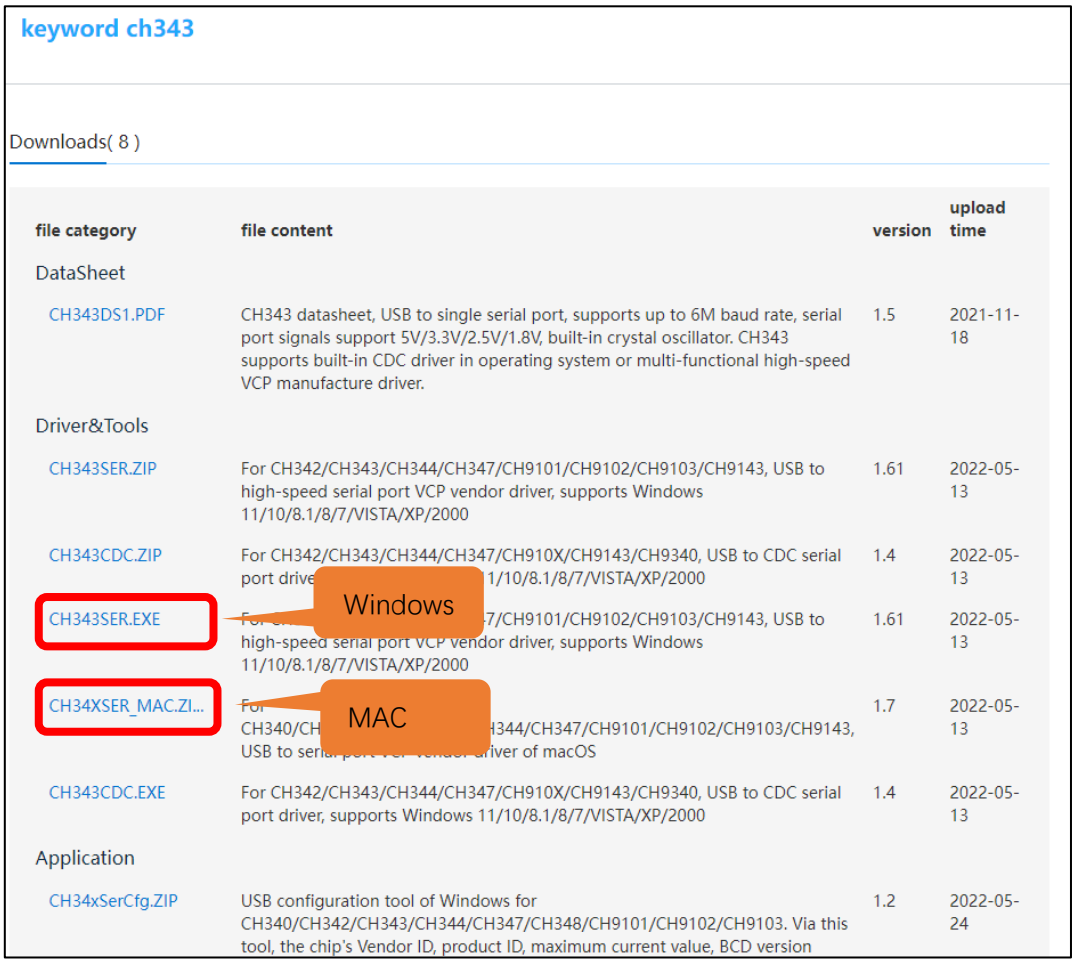


3. Click "Device Manager". If your computer has installed CH343, you can see"USB-Enhances-SERIAL CH343 (COMx)". And you can click here to move to the next step.

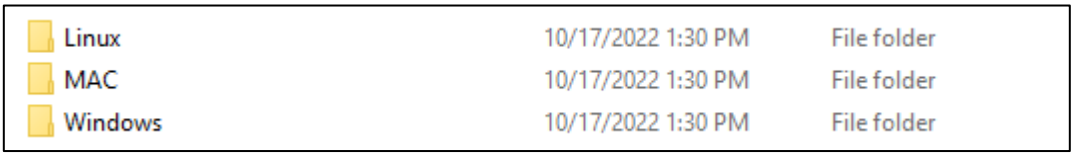


Installing CH343

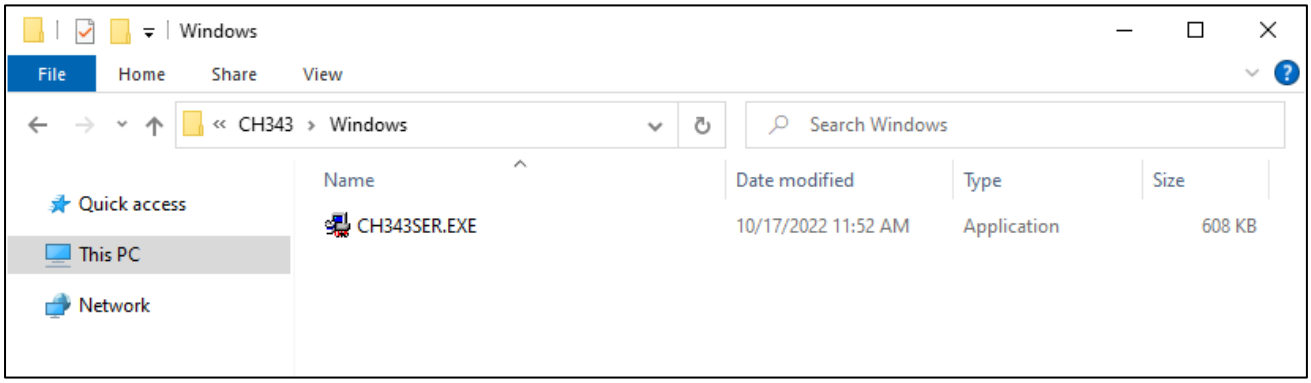
1. First, download CH343 driver, click http://www.wch-ic.com/search?t=all&q=ch343 to download the appropriate one based on your operating system.



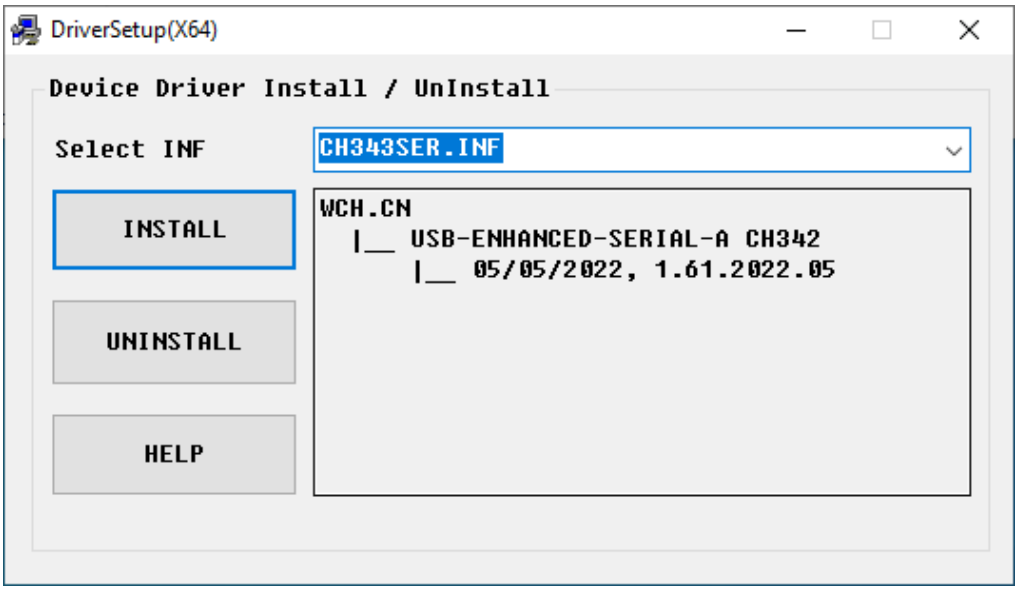
If you would not like to download the installation package, you can open **"Freenove_Media_Kit_for_ESP32-S3/CH343"**, we have prepared the installation package.



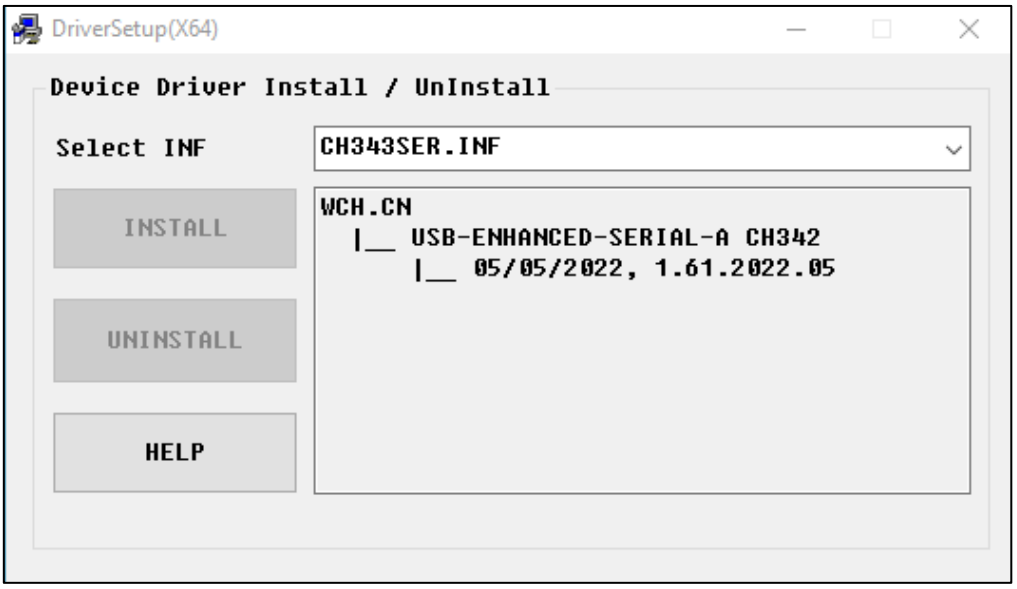
2. Open the folder "**Freenove_Media_Kit_for_ESP32-S3/CH343/Windows/**"



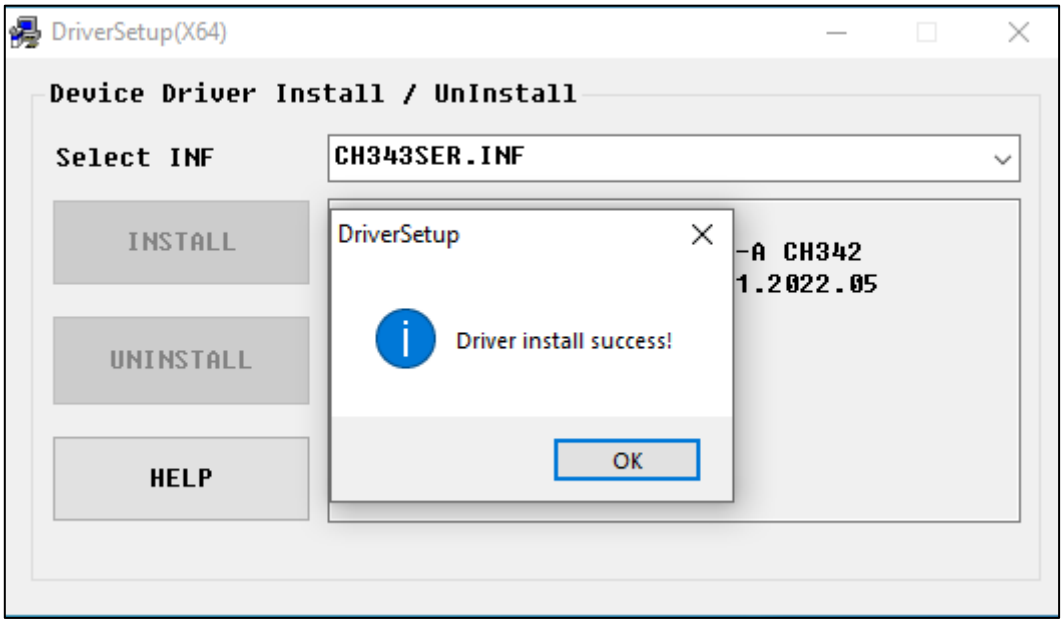
3. Double click "**CH343SER.EXE**".



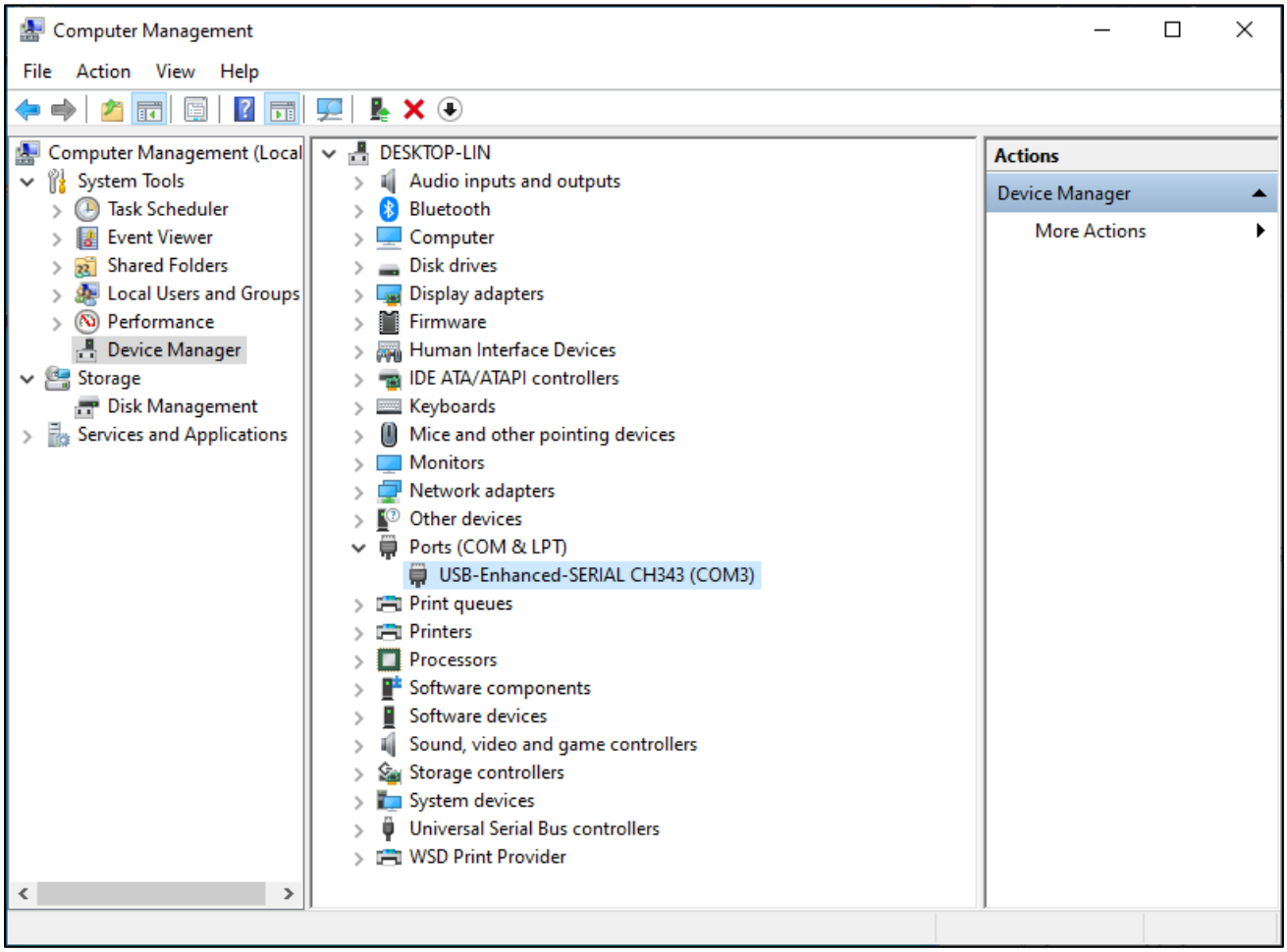
4. Click "INSTALL" and wait for the installation to complete.



5. Install successfully. Close all interfaces.



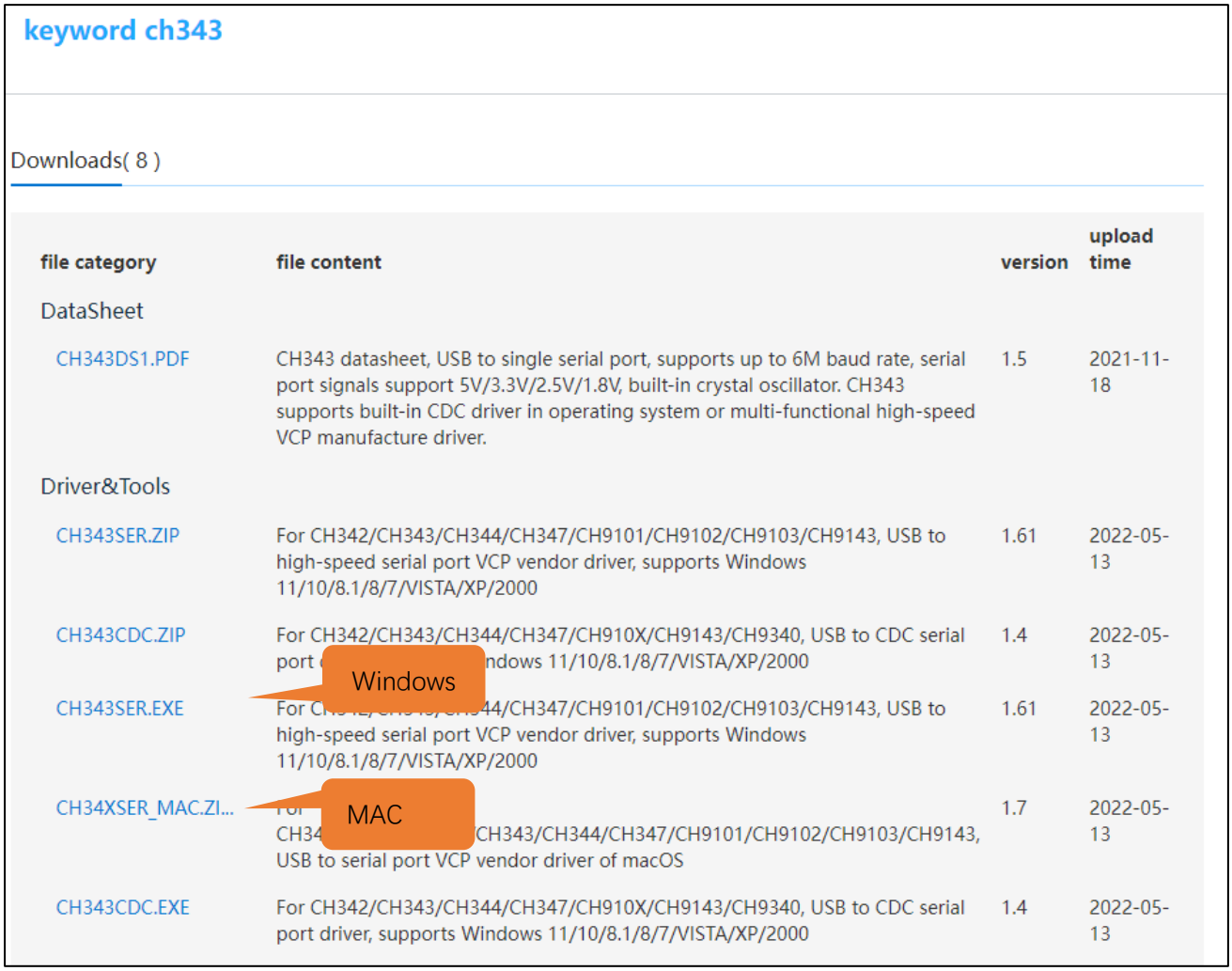
6. When ESP32-S3 WROOM is connected to computer, select "This PC", right-click to select "Manage" and click "Device Manager" in the newly pop-up dialog box, and you can see the following interface.



7. So far, CH343 has been installed successfully. Close all dialog boxes.

MAC

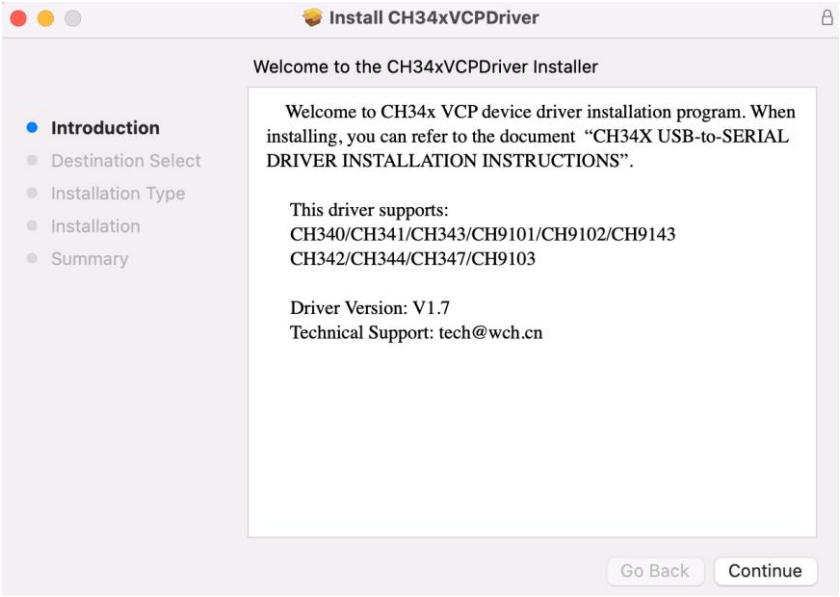
First, download CH343 driver. Click http://www.wch-ic.com/search?t=all&q=ch343 to download the appropriate one based on your operating system.



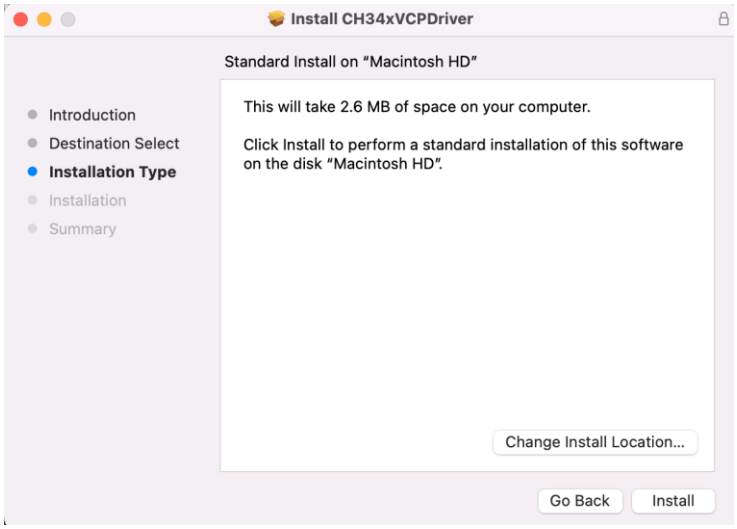
If you would not like to download the installation package, you can open **"Freenove_Media_Kit_for_ESP32-S3/CH343"**. We have prepared the installation package.

Second, open the folder "**Freenove_Media_Kit_for_ESP32-S3/CH343/MAC/**"

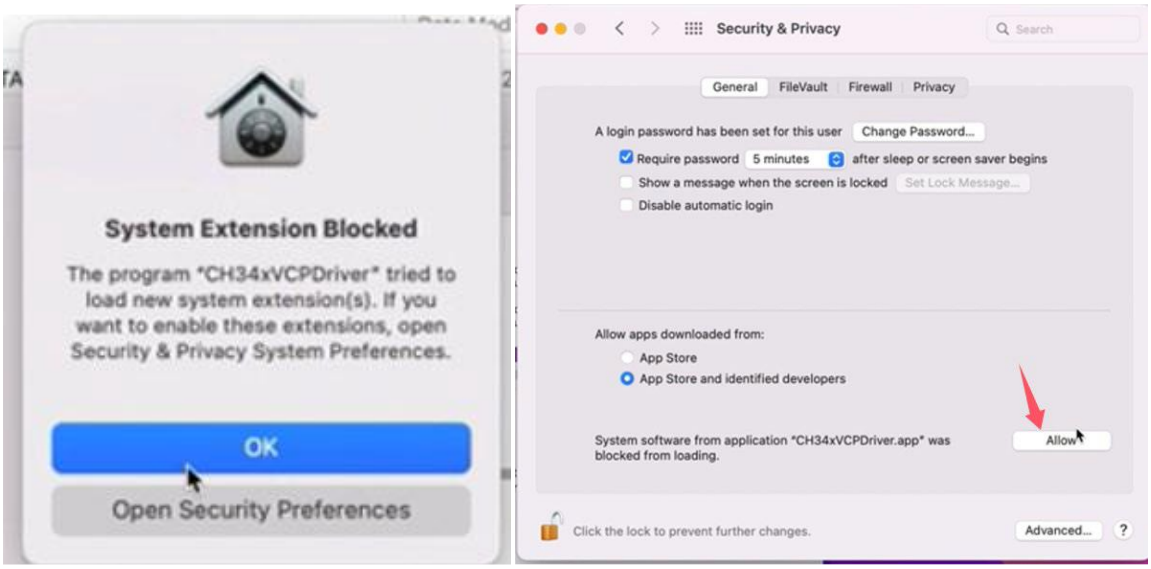
Third, click Continue.



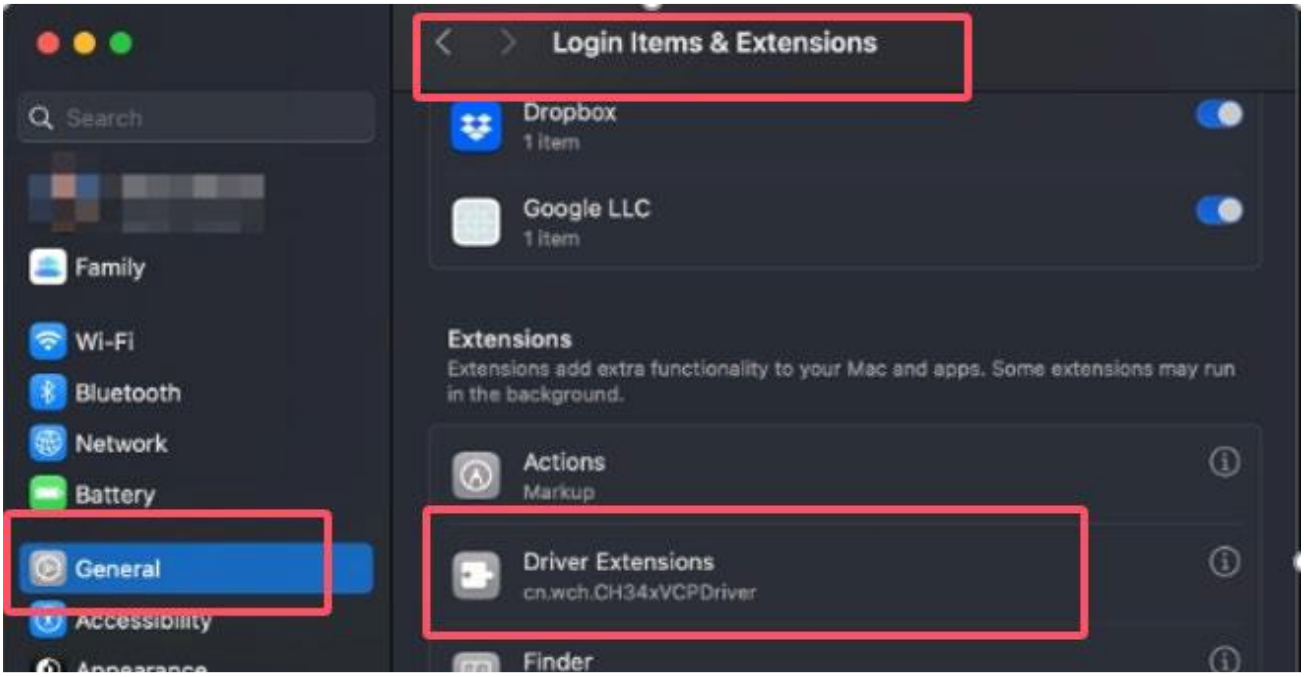
Fourth, click Install.

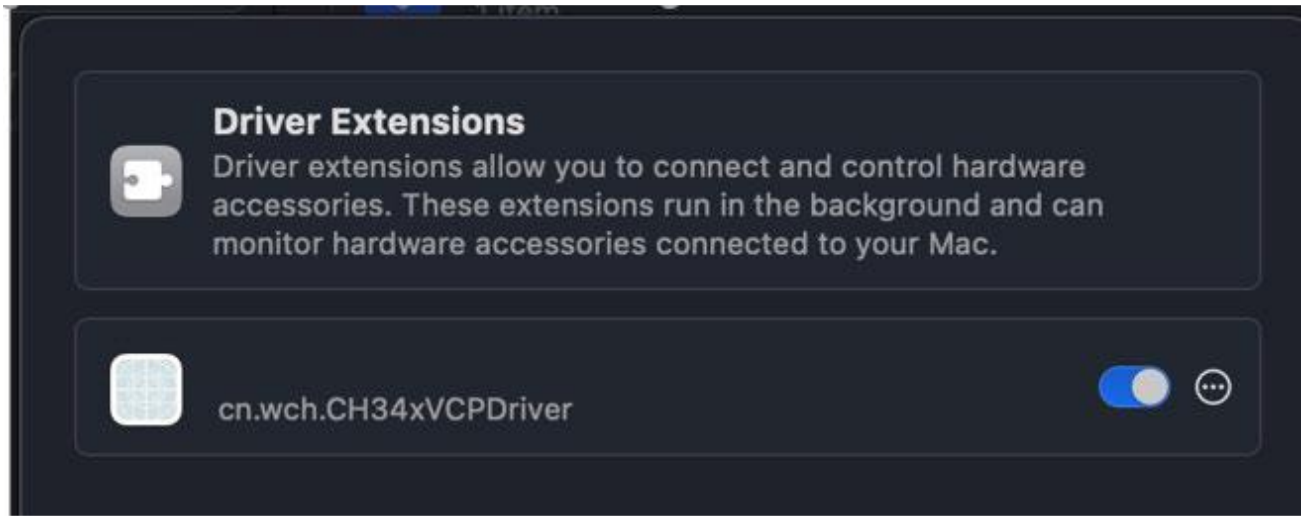


If it is blocked, please open Security Preferences to allow it.

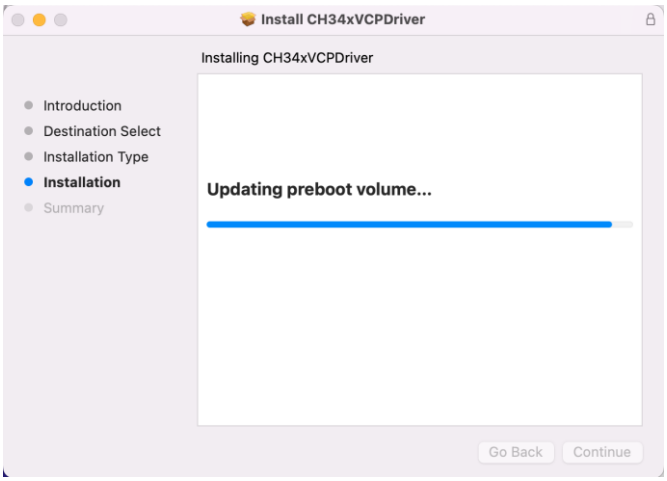


If your OS is Sequoia 15.0.1 and above, please go to System settings - General - login items - extensions to enable the driver.

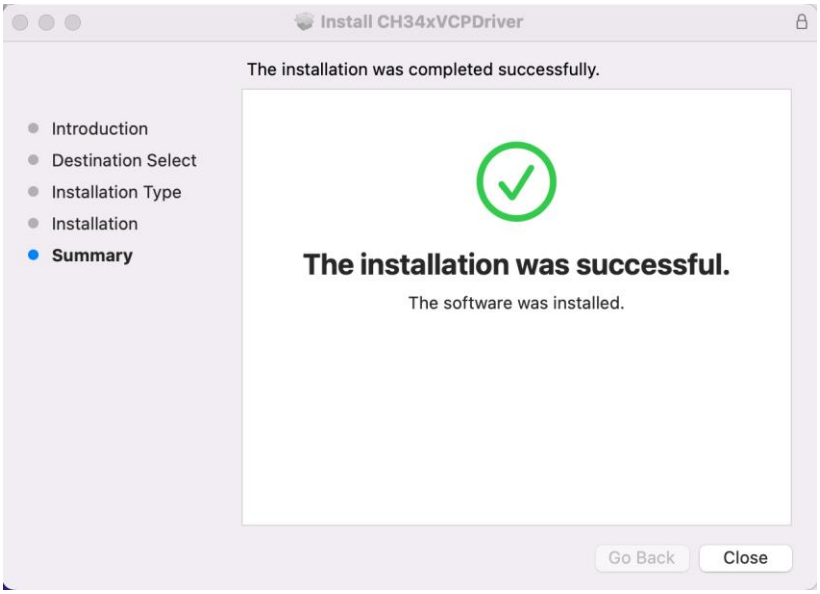




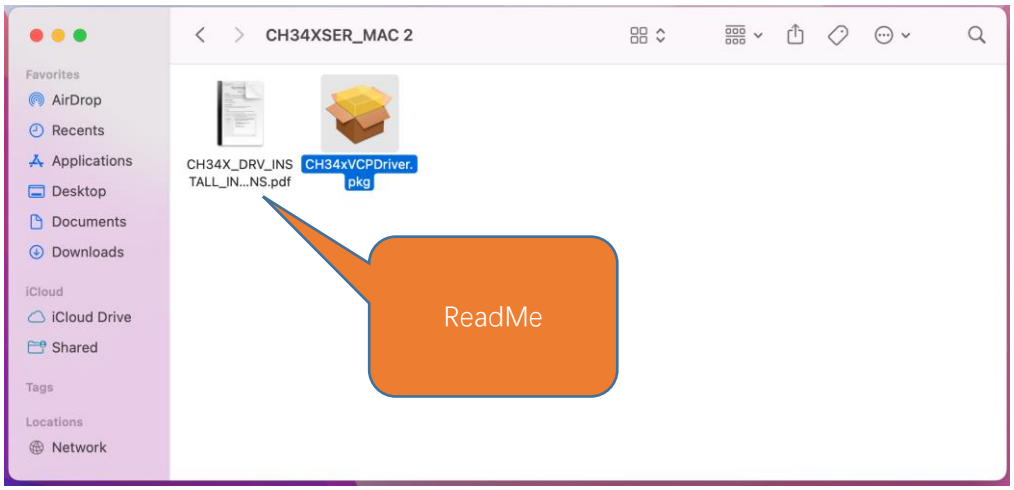
Then, waiting Finish.



Finally, restart your PC.

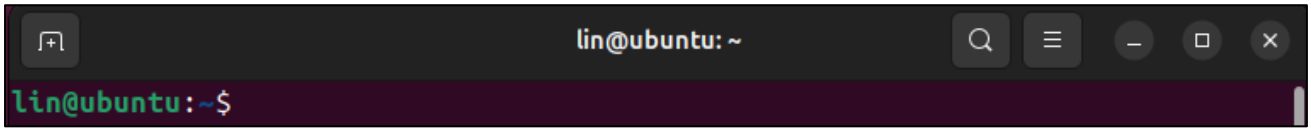


If it fails to be installed with the above steps, you can refer to readme.pdf to install it.



Linux

Here we take Ubuntu system as an example. Open the Terminal.



Run “lsusb” to check the port.

```
lsusb
ls /dev/tty*
```

```
lin@lin-virtual-machine:~$ lsusb
Bus 002 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
Bus 001 Device 004: ID 1a86:55d3 QinHeng Electronics USB Single Serial
Bus 001 Device 003: ID 0e0f:0002 VMware, Inc. Virtual USB Hub
Bus 001 Device 002: ID 0e0f:0003 VMware, Inc. Virtual Mouse
Bus 001 Device 001: ID 1d6b:0001 Linux Foundation 1.1 root hub
```

```
lin@lin-virtual-machine:~$ ls /dev/tty*
/dev/tty    /dev/tty23  /dev/tty39  /dev/tty54      /dev/ttyS1   /dev/ttyS25
/dev/tty0   /dev/tty24  /dev/tty4   /dev/tty55      /dev/ttyS10  /dev/ttyS26
/dev/tty1   /dev/tty25  /dev/tty40  /dev/tty56      /dev/ttyS11  /dev/ttyS27
/dev/tty10  /dev/tty26  /dev/tty41  /dev/tty57      /dev/ttyS12  /dev/ttyS28
/dev/tty11  /dev/tty27  /dev/tty42  /dev/tty58      /dev/ttyS13  /dev/ttyS29
/dev/tty12  /dev/tty28  /dev/tty43  /dev/tty59      /dev/ttyS14  /dev/ttyS3
/dev/tty13  /dev/tty29  /dev/tty44  /dev/tty6       /dev/ttyS15  /dev/ttyS30
/dev/tty14  /dev/tty3   /dev/tty45  /dev/tty60      /dev/ttyS16  /dev/ttyS31
/dev/tty15  /dev/tty30  /dev/tty46  /dev/tty61      /dev/ttyS17  /dev/ttyS4
/dev/tty16  /dev/tty31  /dev/tty47  /dev/tty62      /dev/ttyS18  /dev/ttyS5
/dev/tty17  /dev/tty32  /dev/tty48  /dev/tty63      /dev/ttyS19  /dev/ttyS6
/dev/tty18  /dev/tty33  /dev/tty49  /dev/tty7       /dev/ttyS2   /dev/ttyS7
/dev/tty19  /dev/tty34  /dev/tty5   /dev/tty8       /dev/ttyS20  /dev/ttyS8
/dev/tty2   /dev/tty35  /dev/tty50  /dev/tty9       /dev/ttyS21  /dev/ttyS9
/dev/tty20  /dev/tty36  /dev/tty51  /dev/ttyACM0    /dev/ttyS22
/dev/tty21  /dev/tty37  /dev/tty52  /dev/ttyprintk  /dev/ttyS23
/dev/tty22  /dev/tty38  /dev/tty53  /dev/ttyS0      /dev/ttyS24
lin@lin-virtual-machine:~$
```

CH343 is fully compliant to the Communications Device Class (CDC) standard, they will work with a standard CDC-ACM driver (CDC - Abstract Control Model). Linux operating systems supply a default CDC-ACM driver that can be used with these USB UART devices. In Linux, this driver file name is cdc-acm.

If your computer does not recognize the ESP32S3's port, you can do as follows to install the ch343 driver.

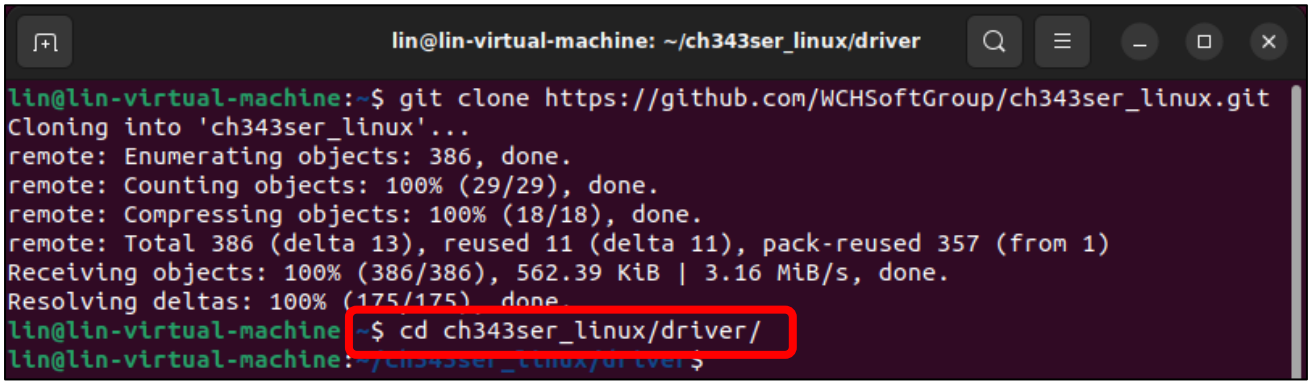
Install the CH343 driver with the following command.

```
git clone https://github.com/WCHSoftGroup/ch343ser_linux.git
```

```
lin@lin-virtual-machine: ~
lin@lin-virtual-machine:~$ git clone https://github.com/WCHSoftGroup/ch343ser_linux.git
Cloning into 'ch343ser_linux'...
remote: Enumerating objects: 386, done.
remote: Counting objects: 100% (29/29), done.
remote: Compressing objects: 100% (18/18), done.
remote: Total 386 (delta 13), reused 11 (delta 11), pack-reused 357 (from 1)
Receiving objects: 100% (386/386), 562.39 KiB | 3.16 MiB/s, done.
Resolving deltas: 100% (175/175), done.
lin@lin-virtual-machine:~$
```

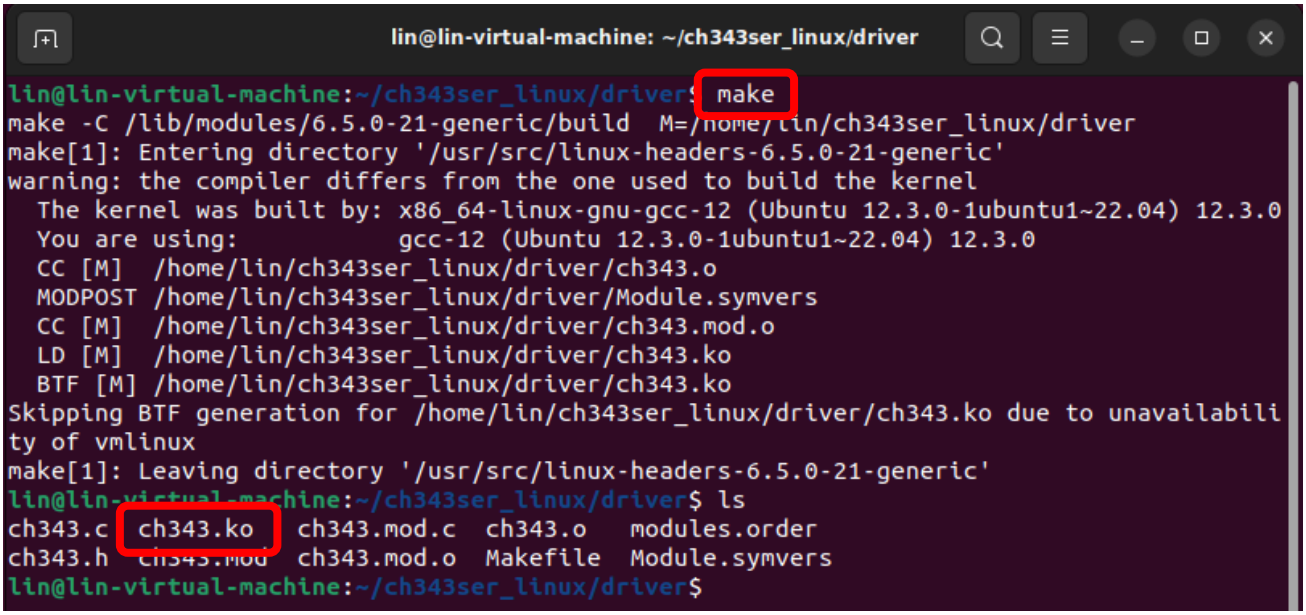
Enter the folder.

```
cd ch343ser_linux/driver/
```



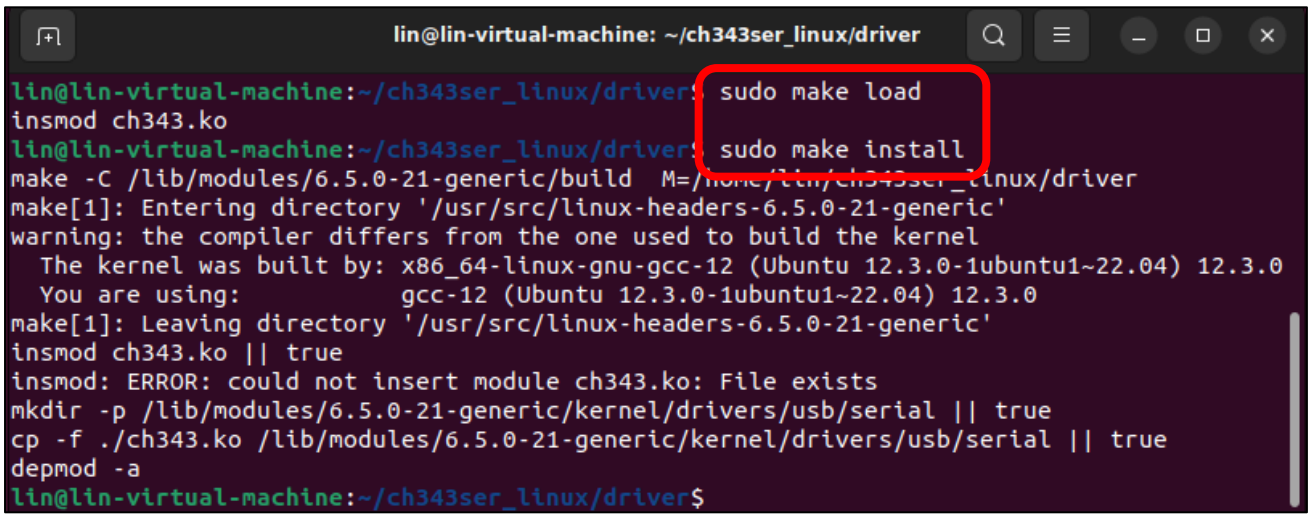
Compile and generate the ch343.ko file.

```
make
```



Load the generated file to the system.

```
sudo make load
sudo make install
```

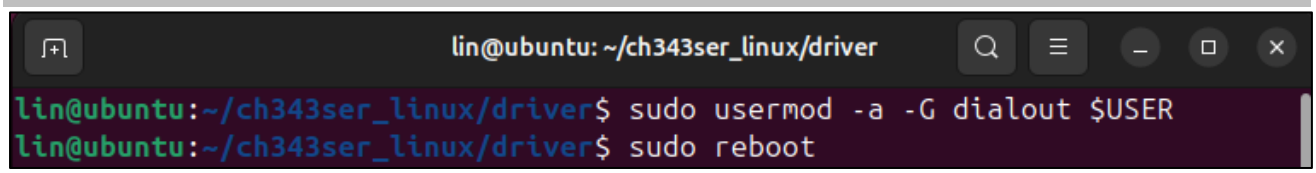


Connect the ESP32S3 to your computer, check the port with the following command and you should see the port.

```
ls /dev/tty*
```

Accessing "ttyACM0" in Ubuntu requires higher privileges, so permission escalation via command is mandatory.

```
sudo usermod -a -G dialout $USER
sudo reboot
```

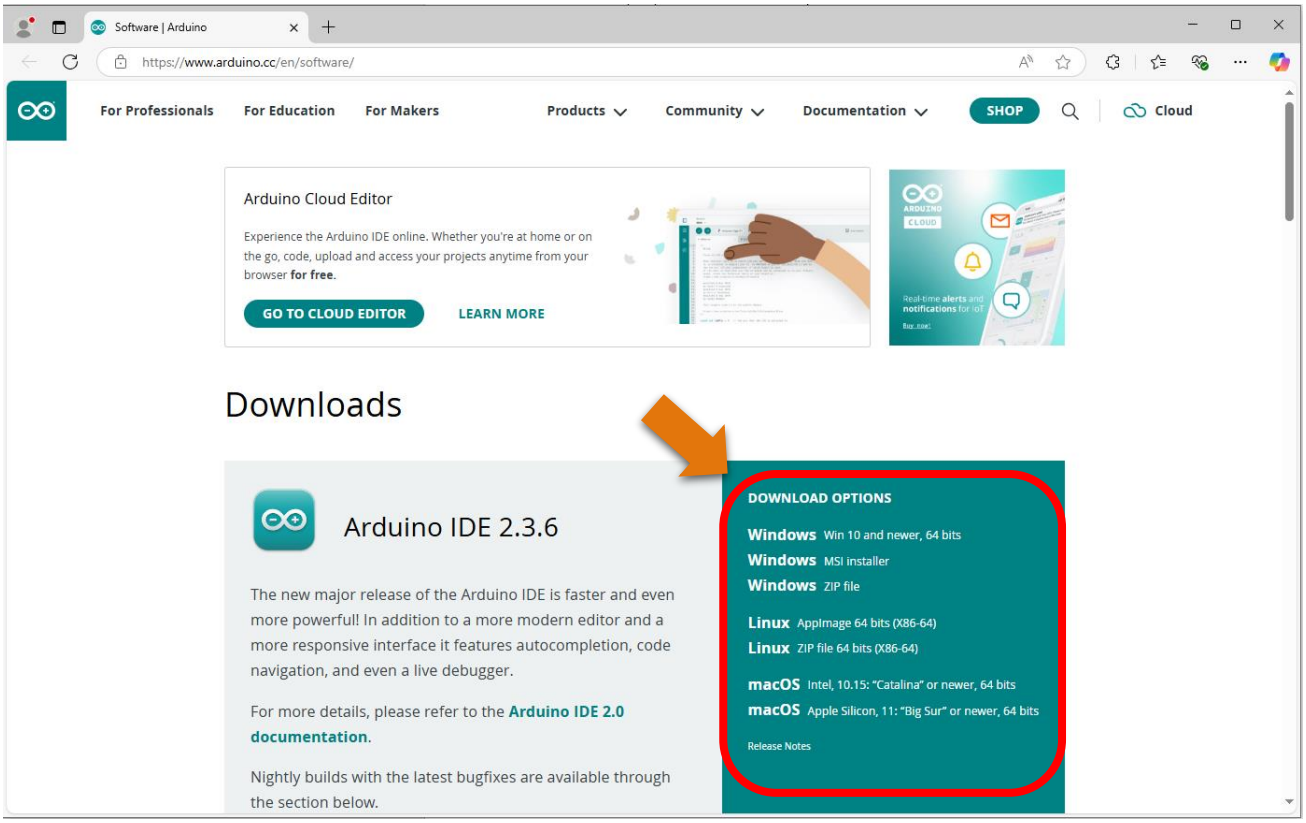


Please note that the configure takes effect after rebooting.

Programming Software

We use the Arduino Software (IDE) to write and upload the code for this product.

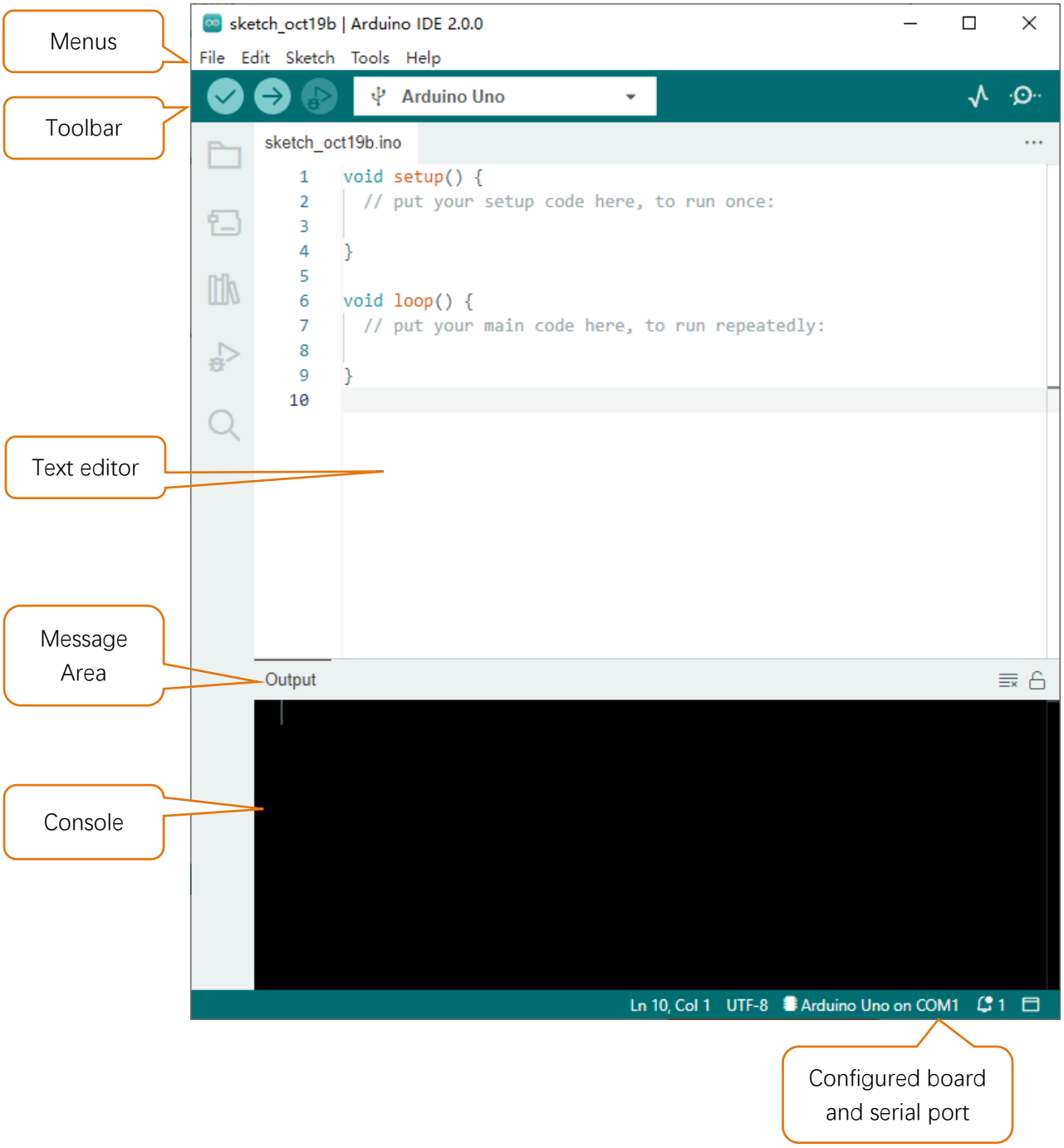
First, install Arduino Software (IDE): visit https://www.arduino.cc/en/software/, Select and download corresponding installer according to your operating system. If you are a Windows user, please select the "Windows" to download and install it correctly.



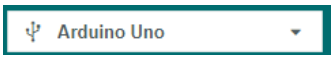
After the download completes, run the installer. For Windows users, there may pop up an installation dialog box of driver during the installation process. When it popes up, please allow the installation.

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:



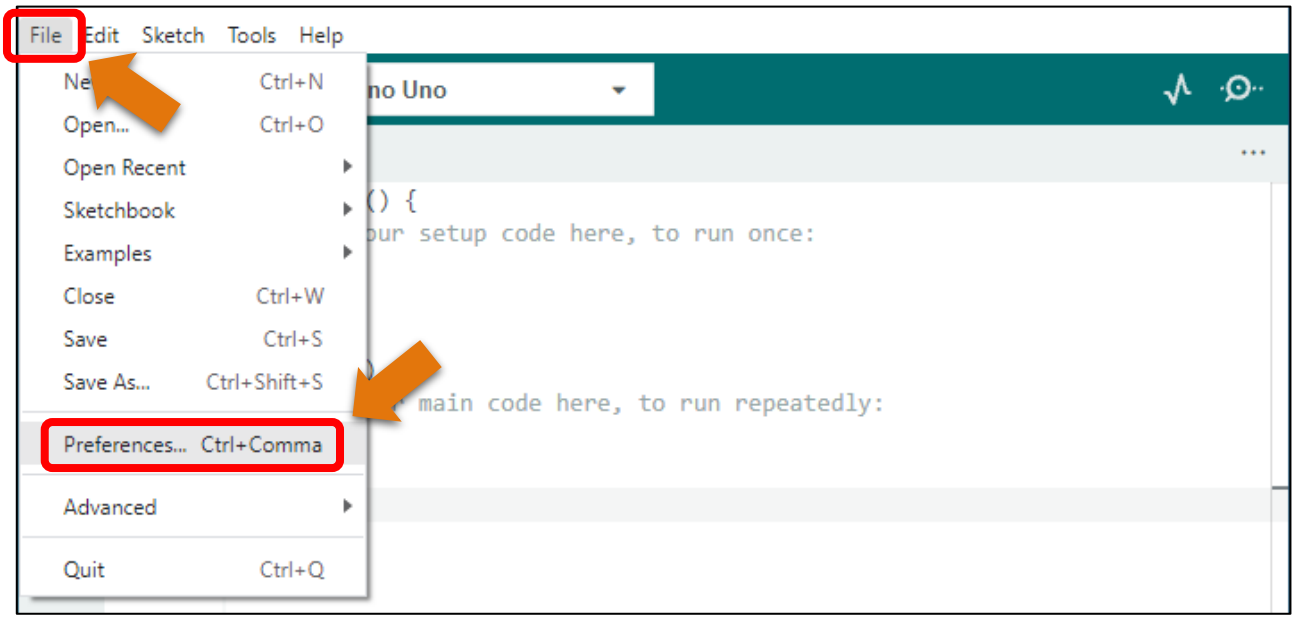
Programs written with Arduino Software (IDE) are called **sketches**. These sketches are written in the text editor and saved with the file extension**.ino**. The editor features text cutting/pasting and searching/replacing. The message area gives feedback while saving and exporting and also displays 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.

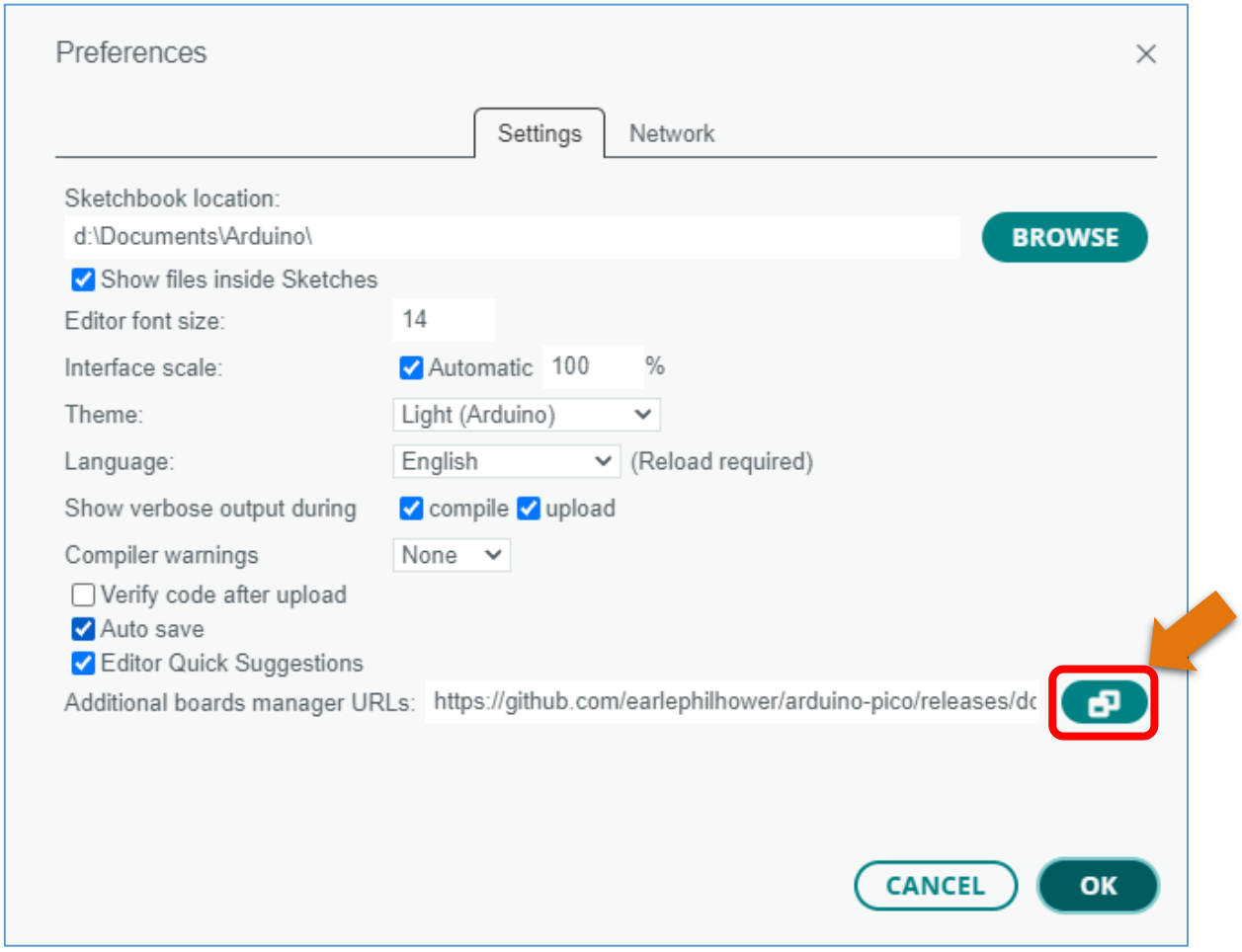
Additional commands are found within the five menus: File, Edit, Sketch, Tools, Help. The menus are context sensitive, which means only those items relevant to the work currently being carried out are available.

Environment Configuration

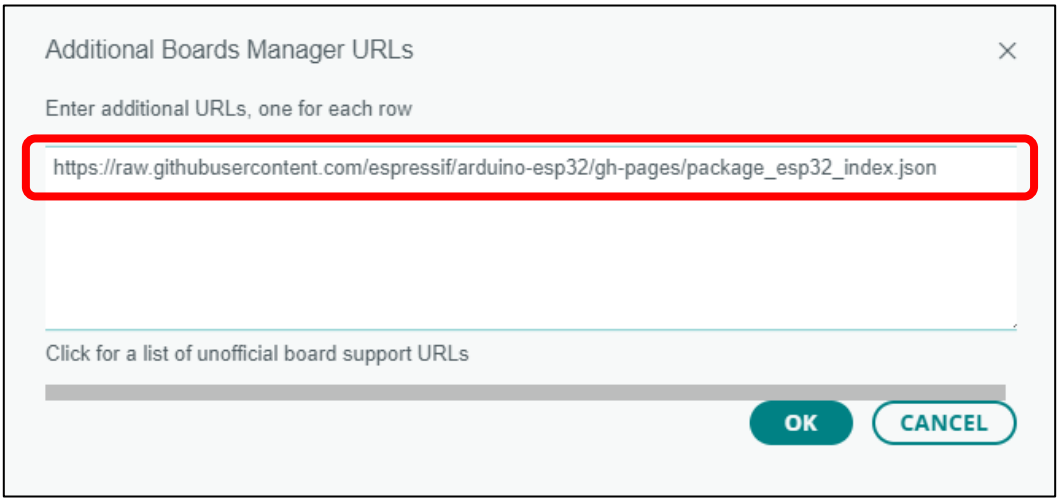
First, open the software platform Arduino, and then click File in Menus and select Preferences.



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

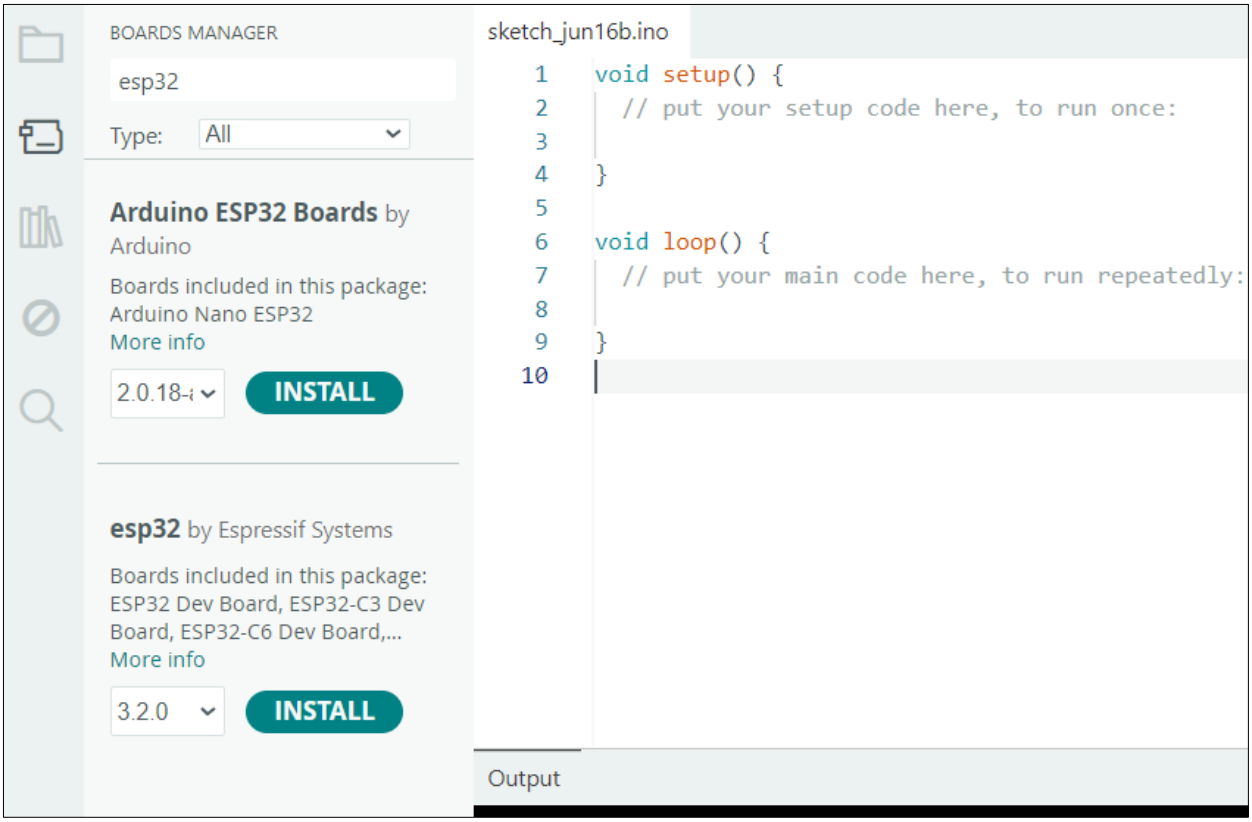


Third, fill in https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json in the new window, click OK, and click OK on the Preferences window again.

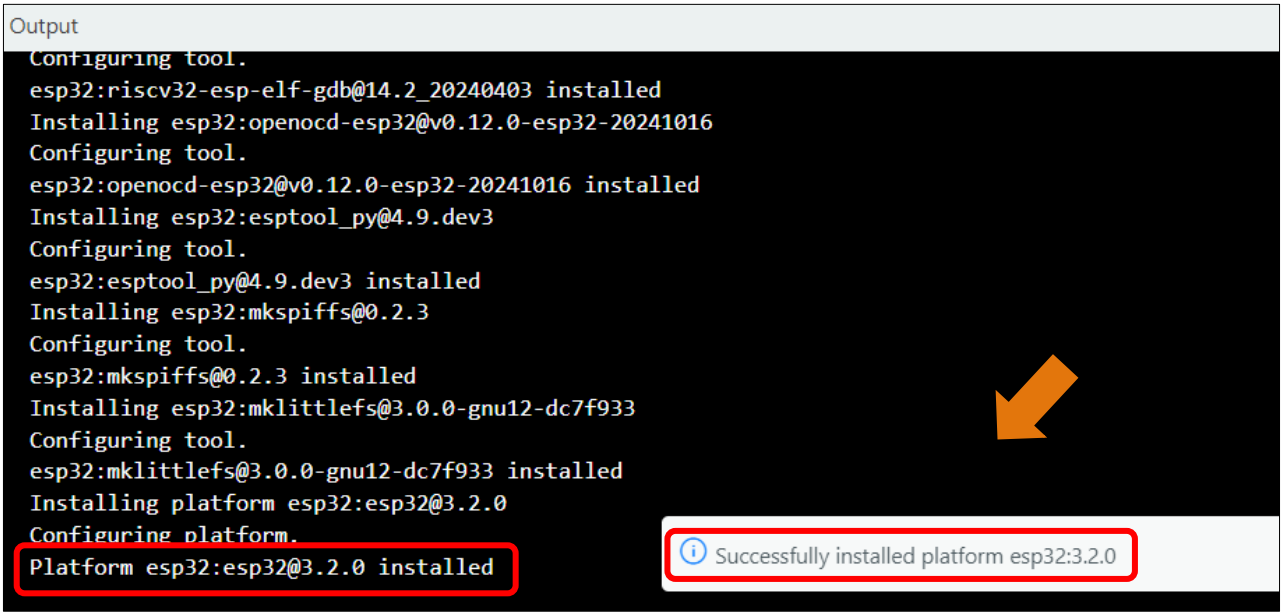


Note: if you copy and paste the URL directly, you may lose the "-". Please check carefully to make sure the link is correct.

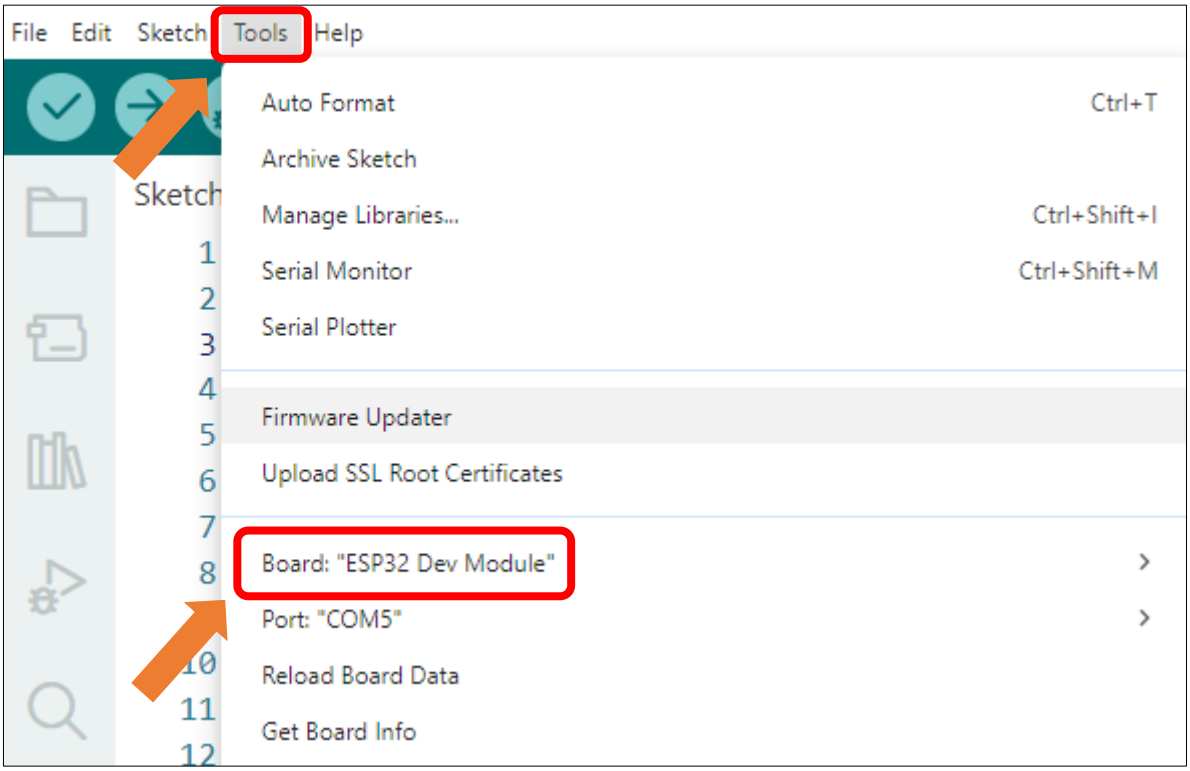
Fourth, click "Boards Manager". Enter “esp32” in Boards manager, select 3.2.0, and click “INSTALL”.



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



When finishing installation, click Tools in the Menus again and select Board: "ESP32 Dev Module", and then you can see information of ESP32.

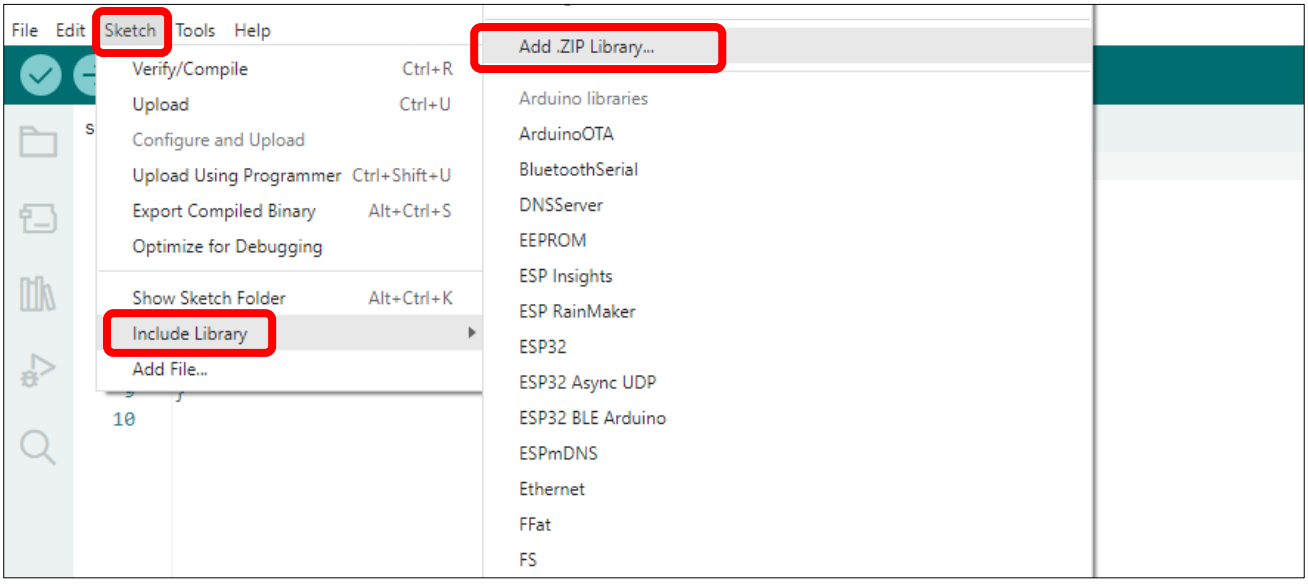


Library Installation

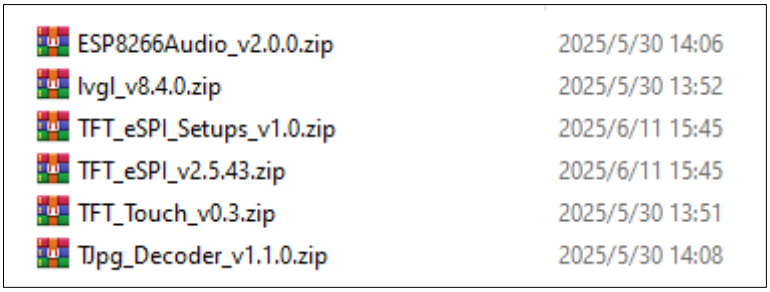
Before starting the learning process, it is necessary to install some libraries in advance to enable the code to be compiled properly. For convenience, we have already packaged these libraries and placed them in the **Freenove_ESP32_Display/Libraries** folder. Please refer to the following steps to install these libraries into the Arduino IDE.

1. Open Arduino IDE.

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

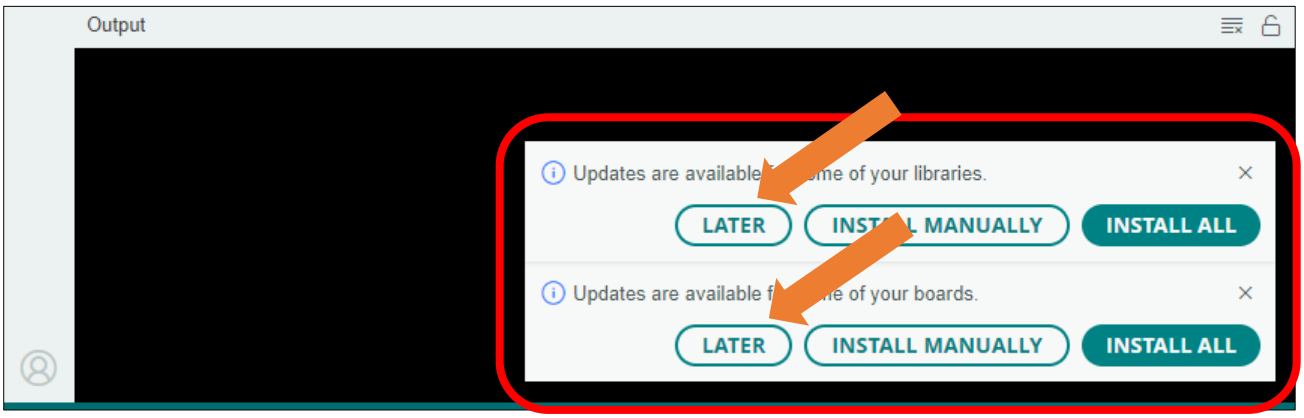


3. On the newly pop-up window, select the files from the **Freenove_ESP32_Display /Libraries**. Click Open to install the library.



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

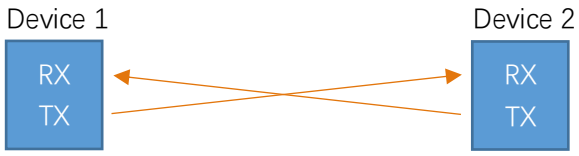
Related Knowledge

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:



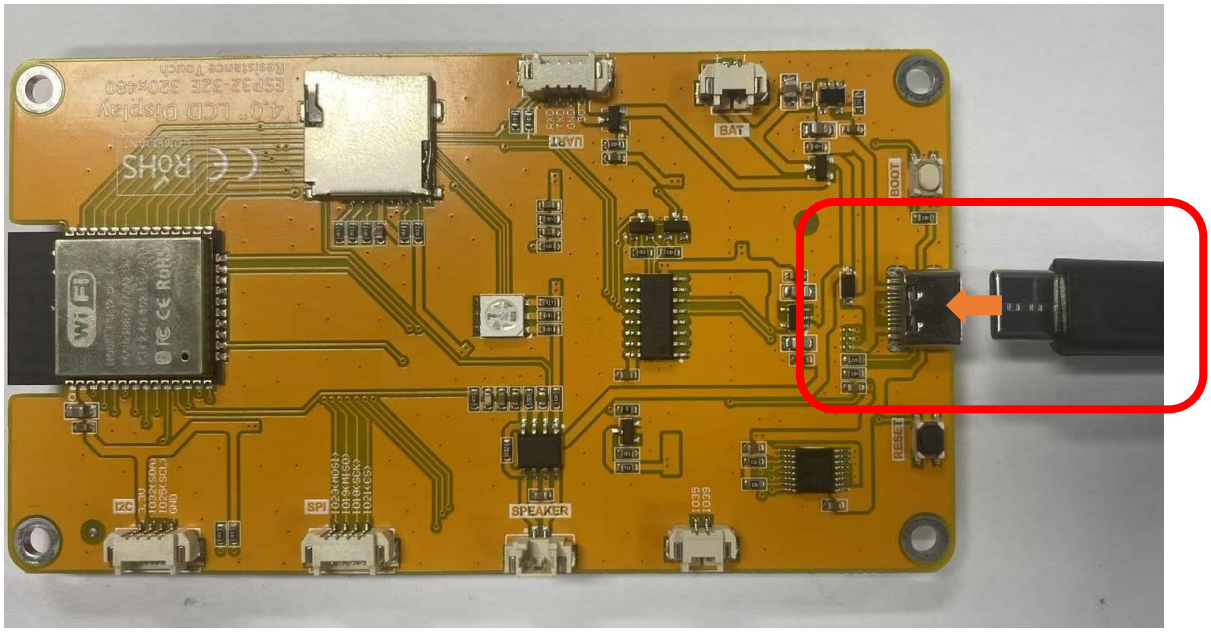
Before serial communication starts, the baud rate in both sides must be the same. Only use the same baud rate can the communication between devices be normal. The baud rates commonly used are 9600 and 115200.

Component List

Freenove ESP32 Display x 1 	USB cable x1 

Circuit

Connect Freenove ESP32 Display to the computer with USB cable.



Sketch

Open "**Sketch_01.1_SerialRW**" folder under **"Freenove_ESP32_Display \Sketch"** 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:   [2025-06-11]
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 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);
```

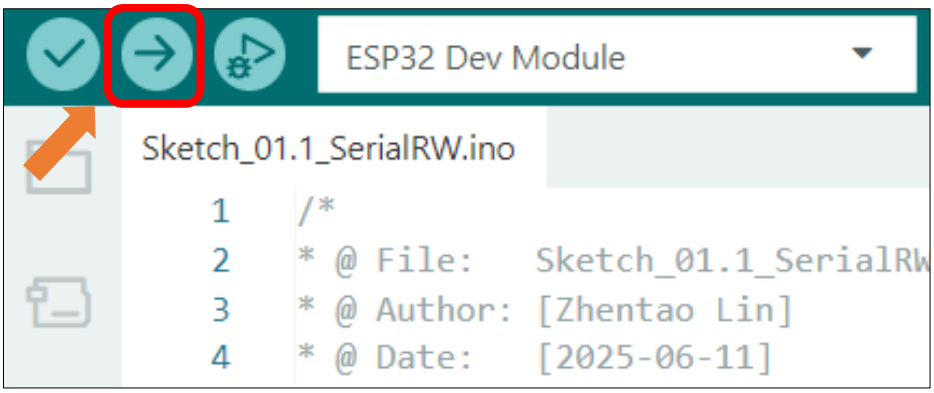
Determine whether there is data in the serial port buffer.

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

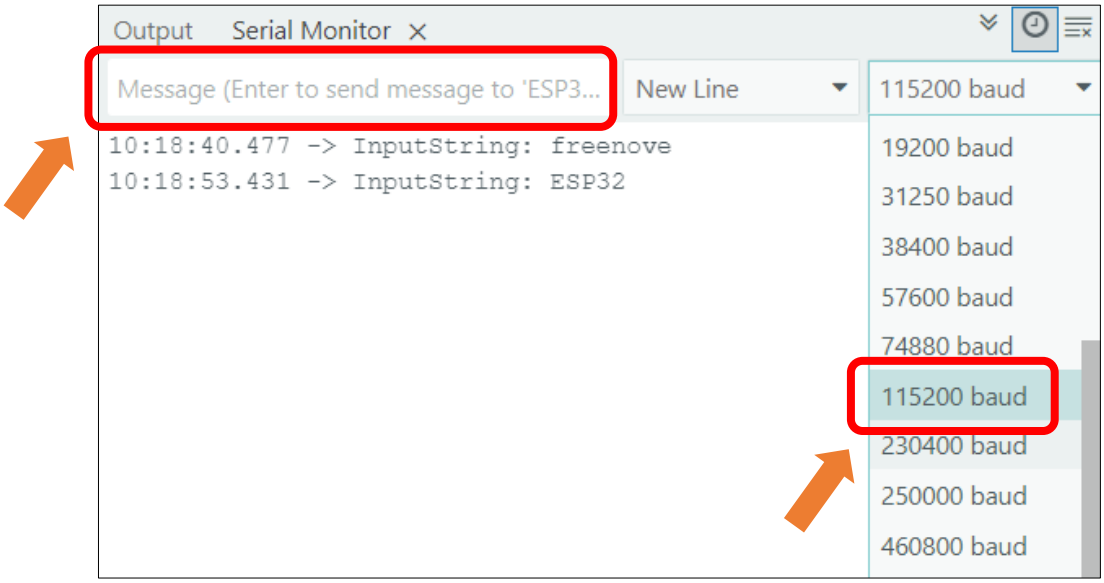
Receive serial port data and save it in the inputString string.

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

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



Reference

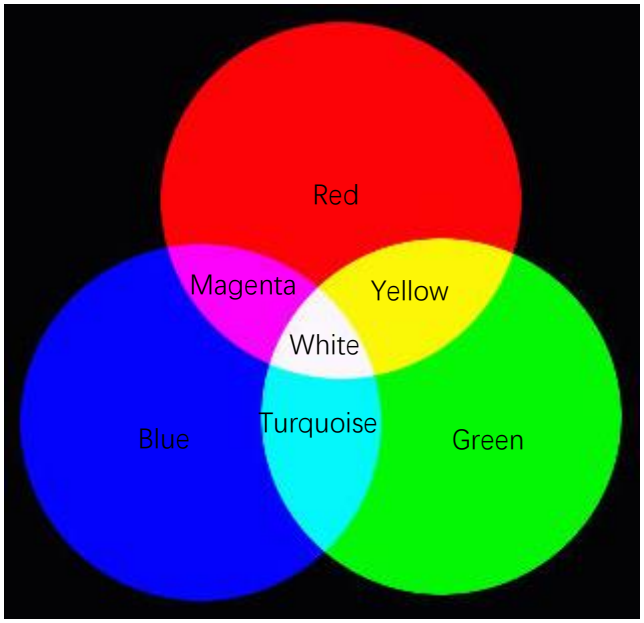
`int available()`
Serial.available() checks the number of bytes currently available to read in the Serial receive buffer. It returns the number of bytes available (int type), or 0 if the buffer is empty.
`int read ()`
Serial.read() reads one byte of data from the Serial receive buffer and returns it 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

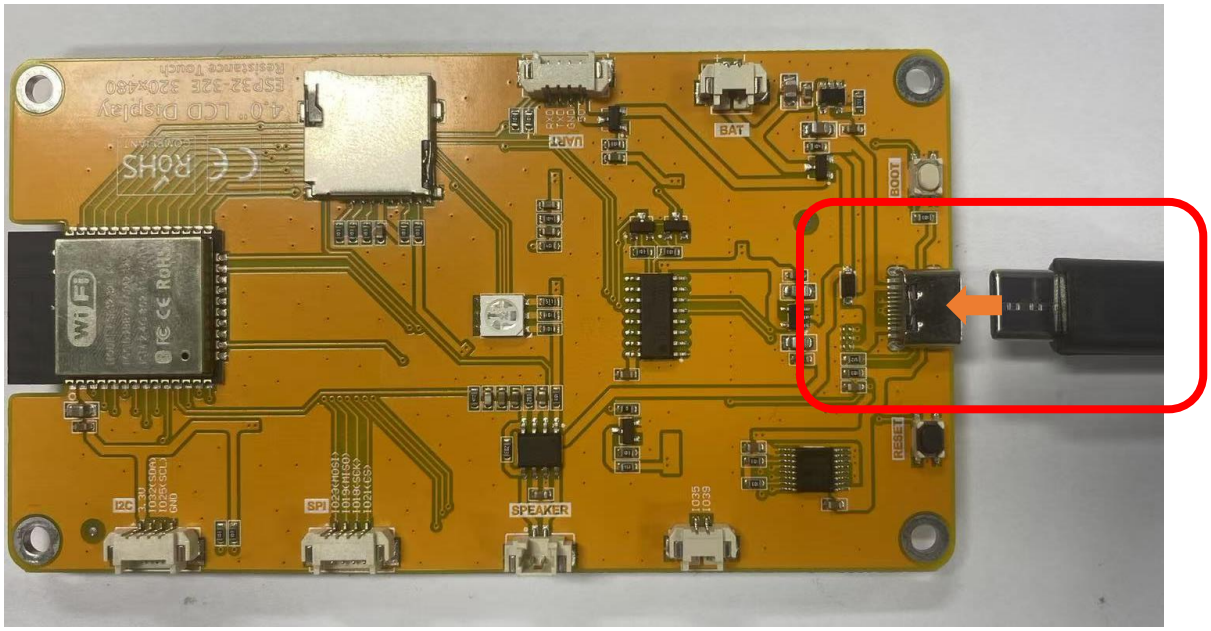
The onboard RGB LED can emit three basic colors of red, green and blue, and supports 256-level brightness adjustment, which means that it can emit 2^24=16,777,216 different colors.

Component List

Freenove ESP32 Display x 1 	USB cable x1 

Circuit

Connect Freenove ESP32 Display to the computer with USB cable.



Sketch

Open "**Sketch_02.1_RGB**" folder under **"Freenove_ESP32_Display\Sketch"** and double-click "**Sketch_02.1_RGB.ino**".

Sketch_02.1_RGB

The following is the program code:

```
/*
* @ File:   Sketch_02.1_RGB.ino
* @ Author: [Zhentao Lin]
* @ Date:   [2025-06-11]
*/

#include <Arduino.h>
#define RED_PIN 22
#define GREEN_PIN 16
#define BLUE_PIN 17

void rgbInit(void) {
    pinMode(RED_PIN, OUTPUT);
    pinMode(GREEN_PIN, OUTPUT);
    pinMode(BLUE_PIN, OUTPUT);
}

```

```
18  void setRGB(bool redLevel, bool greenLevel, bool blueLevel) {
19      digitalWrite(RED_PIN, !redLevel);
20      digitalWrite(GREEN_PIN, !greenLevel);
21      digitalWrite(BLUE_PIN, !blueLevel);
22  }
23
24  void setup() {
25      rgbInit();
26  }
27
28  void loop() {
29      setRGB(1, 0, 0);
30      delay(500);
31      setRGB(0, 1, 0);
32      delay(500);
33      setRGB(0, 0, 1);
34      delay(500);
35      setRGB(0, 0, 0);
36      delay(500);
37  }
```

Code Explanation

Define the pins for the RGB LED.

```
8   #define RED_PIN 22
9   #define GREEN_PIN 16
10  #define BLUE_PIN 17
```

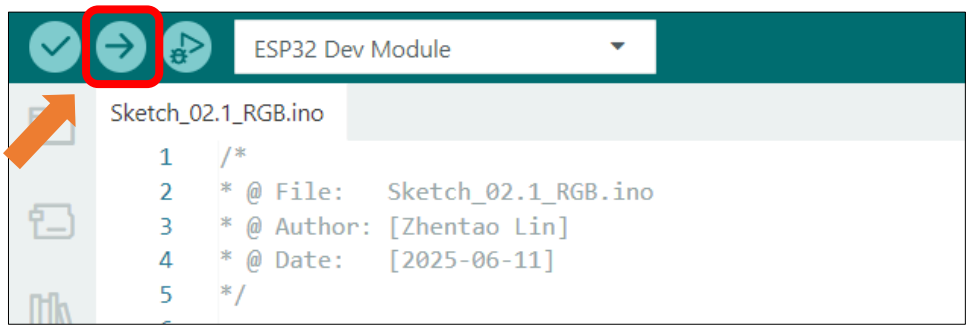
Initialize the LED.

```
25  rgbInit();
```

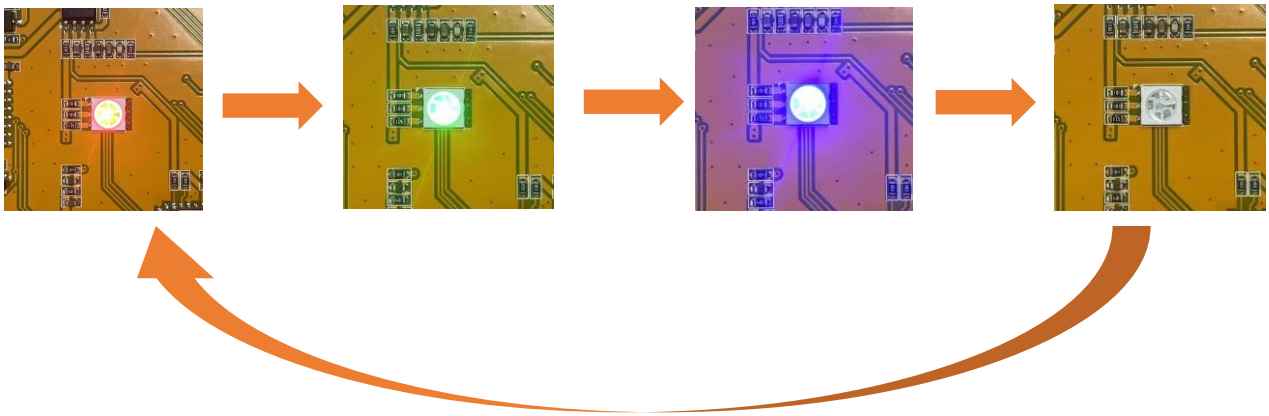
Change the color of the RGB LED continuously with a cycle period of 500ms.

```
29  setRGB(1, 0, 0);
30  delay(500);
31  setRGB(0, 1, 0);
32  delay(500);
33  setRGB(0, 0, 1);
34  delay(500);
35  setRGB(0, 0, 0);
36  delay(500);
```

Click "**Upload**" to upload the code to Freenove_ESP32_Display.



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



Reference

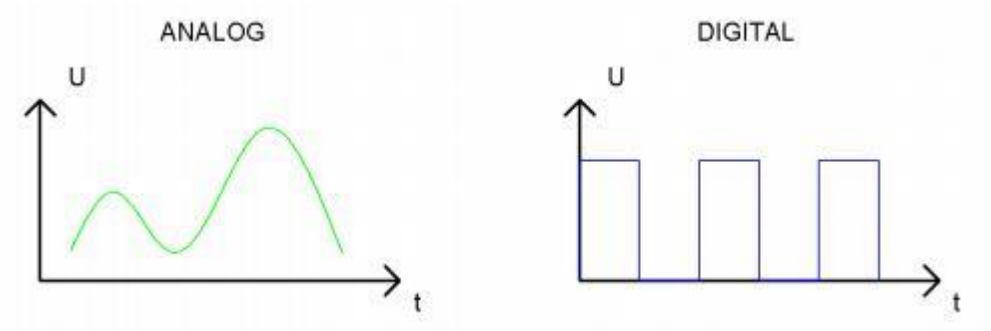
`void setRGB(bool redLevel, bool greenLevel, bool blueLevel)`	
This function controls the RGB light to display different colors. Parameters: redLevel: When set to 1, the red light turns on; when 0, it turns off. greenLevel: When set to 1, the green light turns on; when 0, it turns off. blueLevel: When set to 1, the blue light turns on; when 0, it turns off.	

Project 2.2 PWM RGB

Related Knowledge

Analog & Digital

An Analog Signal is a continuous signal in both time and value. On the contrary, a Digital Signal or discrete - time signal is a time series consisting of a sequence of quantities. Most signals in life are analog signals. A familiar example of an Analog Signal would be how the temperature throughout the day is continuously changing and could not suddenly change instantaneously from 0℃ to 10℃. However, Digital Signals can instantaneously change in value. This change is expressed in numbers as 1 and 0 (the basis of binary code). Their differences can more easily be seen when compared when graphed as below.

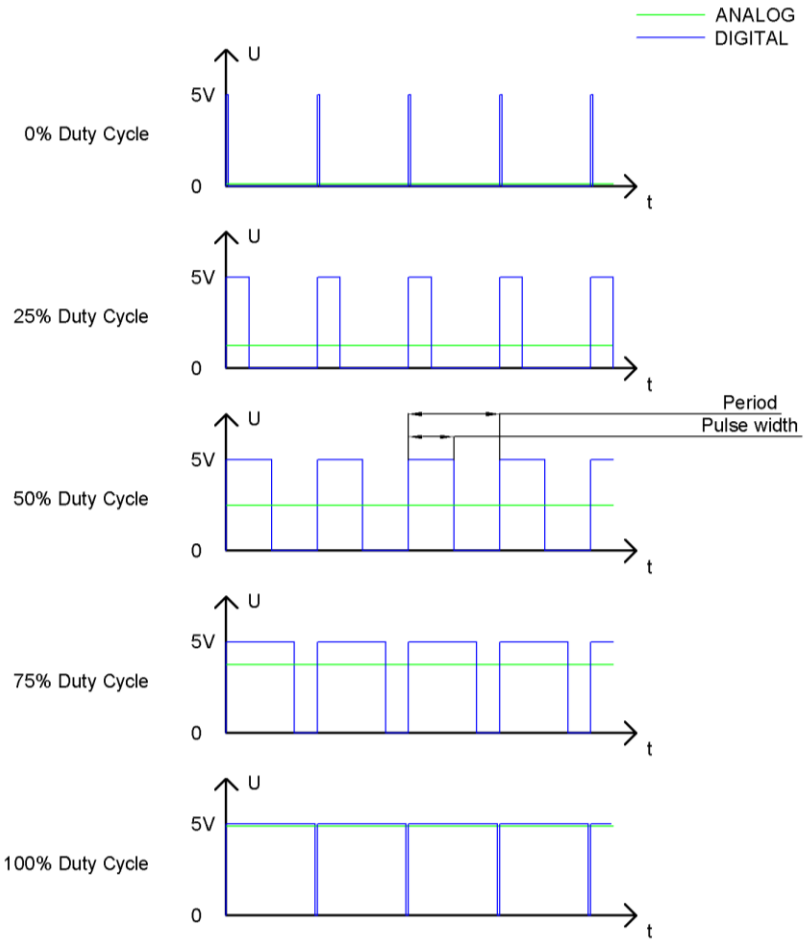


In practical application, we often use binary as the digital signal, which is a series of 0's and 1's. Since a binary signal only has two values (0 or 1), it has great stability and reliability. Lastly, both analog and digital signals can be converted into the other.

PWM

PWM, Pulse Width Modulation, uses digital pins to send certain frequencies of square waves, that is, the output of high levels and low levels, which alternately last for a while. The total time for each set of high levels and low levels is generally fixed, which is called the period (the reciprocal of the period is frequency). The time of high level outputs are generally called "pulse width", and the duty cycle is the percentage of the ratio of pulse duration, or pulse width (PW) to the total period (T) of the waveform.

The longer the output of high levels last, the larger the duty cycle and the higher the corresponding voltage in analog signal will be. The following figures show how the analogs signal voltage vary between 0V-5V (high level is 5V) corresponding to the pulse width 0%-100%:

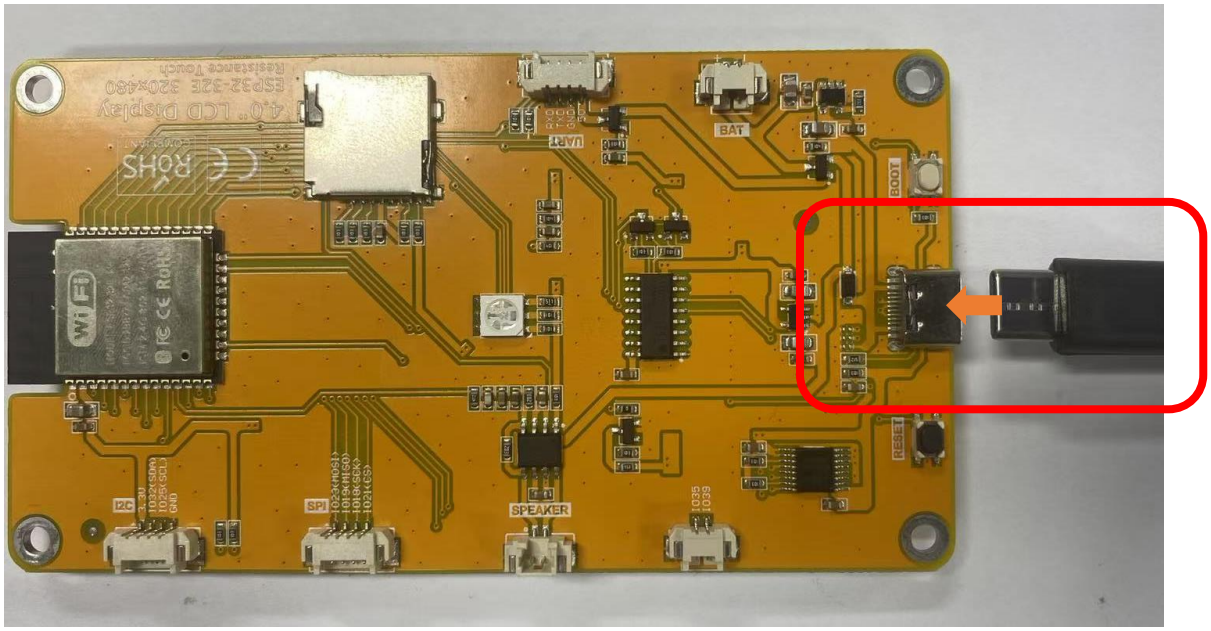


Component List

Freenove ESP32 Display x 1	USB cable x1
	

Circuit

Connect Freenove ESP32 Display to the computer with USB cable.



Sketch

Open "**Sketch_02.2_PWM_RGB**" folder under **"Freenove_ESP32_Display\Sketch"** and double-click "**Sketch_02.2_PWM_RGB.ino**".

Sketch_02.2_PWM_RGB

The following is the program code:

```
/*
* @ File:   Sketch_02.2_PWM_RGB.ino
* @ Author: [Zhentao Lin]
* @ Date:   [2025-06-11]
*/

#include <Arduino.h>

#define RED_PIN 22
#define GREEN_PIN 16
#define BLUE_PIN 17
#define LEDC_RESOLUTION 8
#define LEDC_FREQ 2000
#define LEDC_RED_CHANNEL 0
#define LEDC_GREEN_CHANNEL 1
#define LEDC_BLUE_CHANNEL 2

```

```
#define PWM_BRIGHTNESS 255
#define DELAY_TIME 2

void pwmInit(void) {
    ledcAttachChannel(RED_PIN, LEDC_FREQ, LEDC_RESOLUTION, LEDC_RED_CHANNEL);
    ledcAttachChannel(GREEN_PIN, LEDC_FREQ, LEDC_RESOLUTION, LEDC_GREEN_CHANNEL);
    ledcAttachChannel(BLUE_PIN, LEDC_FREQ, LEDC_RESOLUTION, LEDC_BLUE_CHANNEL);
}

void setPwm(int redValue, int greenValue, int blueValue) {
    ledcWrite(RED_PIN, 255 - redValue);
    ledcWrite(GREEN_PIN, 255 - greenValue);
    ledcWrite(BLUE_PIN, 255 - blueValue);
}

void setup() {
    pwmInit();
}

void loop() {
    for (int i = 0; i <= PWM_BRIGHTNESS; i++) {
      setPwm(i, 0, 0);
      delay(DELAY_TIME);
    }
    for (int i = PWM_BRIGHTNESS; i >= 0; i--) {
      setPwm(i, 0, 0);
      delay(DELAY_TIME);
    }

    for (int i = 0; i <= PWM_BRIGHTNESS; i++) {
      setPwm(0, i, 0);
      delay(DELAY_TIME);
    }
    for (int i = PWM_BRIGHTNESS; i >= 0; i--) {
      setPwm(0, i, 0);
      delay(DELAY_TIME);
    }

    for (int i = 0; i <= PWM_BRIGHTNESS; i++) {
      setPwm(0, 0, i);
      delay(DELAY_TIME);
    }
    for (int i = PWM_BRIGHTNESS; i >= 0; i--) {
      setPwm(0, 0, i);
```

```
62          delay(DELAY_TIME);
63      }
64  }
```

Code Explanation

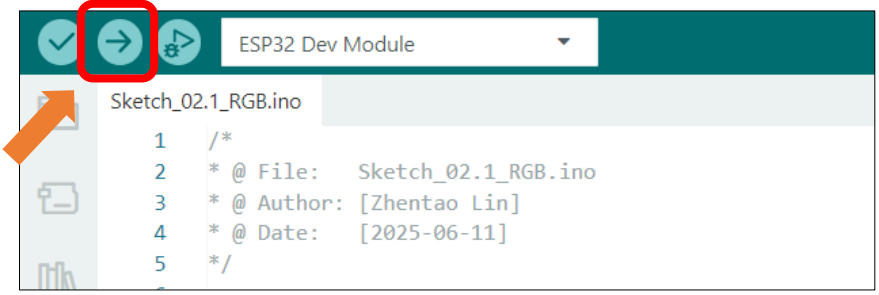
Write a PWM initialization function to configure the pins controlling the RGB lights as PWM output mode.

```
8   pwmInit();
```

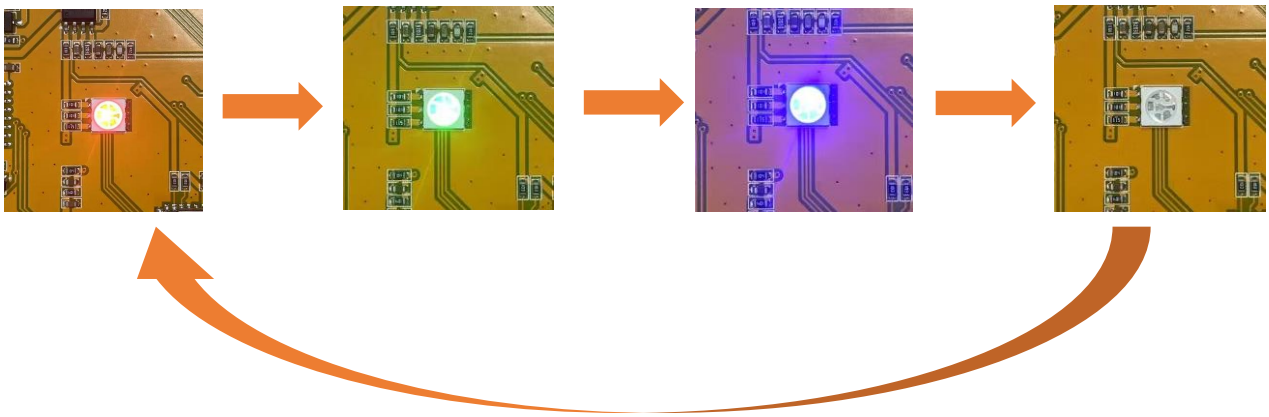
Use a “for” loop to create a smooth breathing light effect on the RGB LED.

```
32  for (int i = 0; i <= PWM_BRIGHTNESS; i++) {
33      setPwm(i, 0, 0);
34      delay(DELAY_TIME);
35  }
36  for (int i = PWM_BRIGHTNESS; i >= 0; i--) {
37      setPwm(i, 0, 0);
38      delay(DELAY_TIME);
39  }
40
41  for (int i = 0; i <= PWM_BRIGHTNESS; i++) {
42      setPwm(0, i, 0);
43      delay(DELAY_TIME);
44  }
45  for (int i = PWM_BRIGHTNESS; i >= 0; i--) {
46      setPwm(0, i, 0);
47      delay(DELAY_TIME);
48  }
49
50  for (int i = 0; i <= PWM_BRIGHTNESS; i++) {
51      setPwm(0, 0, i);
52      delay(DELAY_TIME);
53  }
54  for (int i = PWM_BRIGHTNESS; i >= 0; i--) {
55      setPwm(0, 0, i);
56      delay(DELAY_TIME);
57  }
```

This code achieves the function of printing data on serial monitor. Click “**Upload**” to upload the code to Freenove_ESP32_Display.



After uploading the code, the RGB LED produces a smooth breathing effect, gradually increasing in brightness while cycling through colors in the sequence: red → green → blue → off (repeat).



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 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:

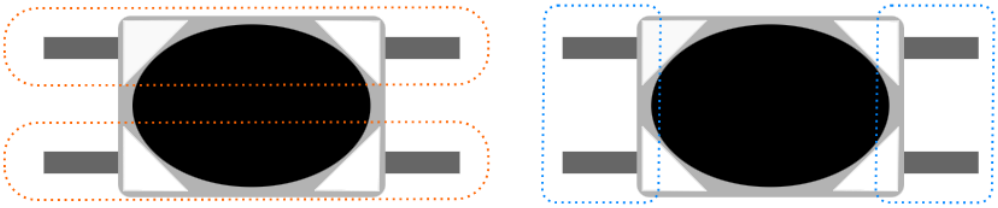
When the board is powered on or reset, pressing and holding the Boot button forces the board into download mode.

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

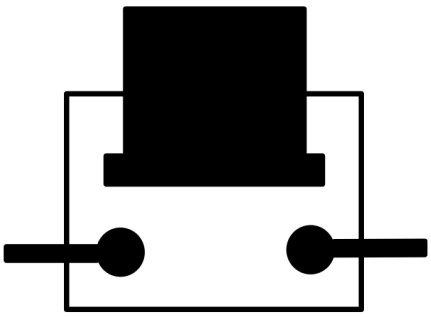
Use as a Regular Input Button:

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.

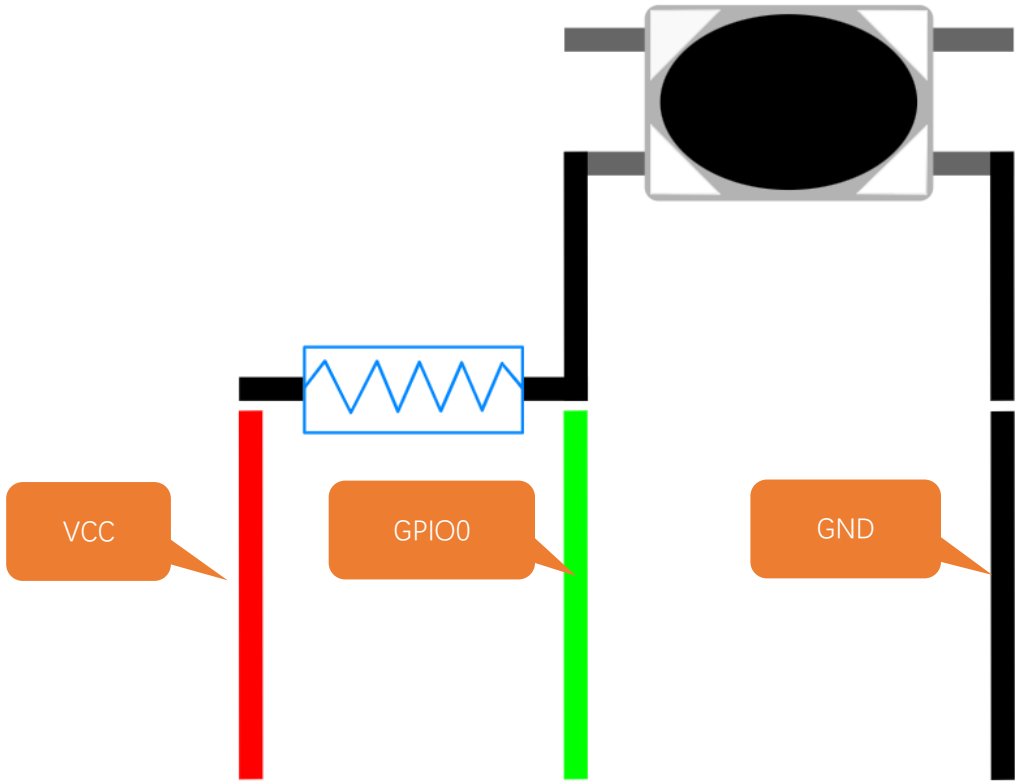


Before pressing After pressing



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.

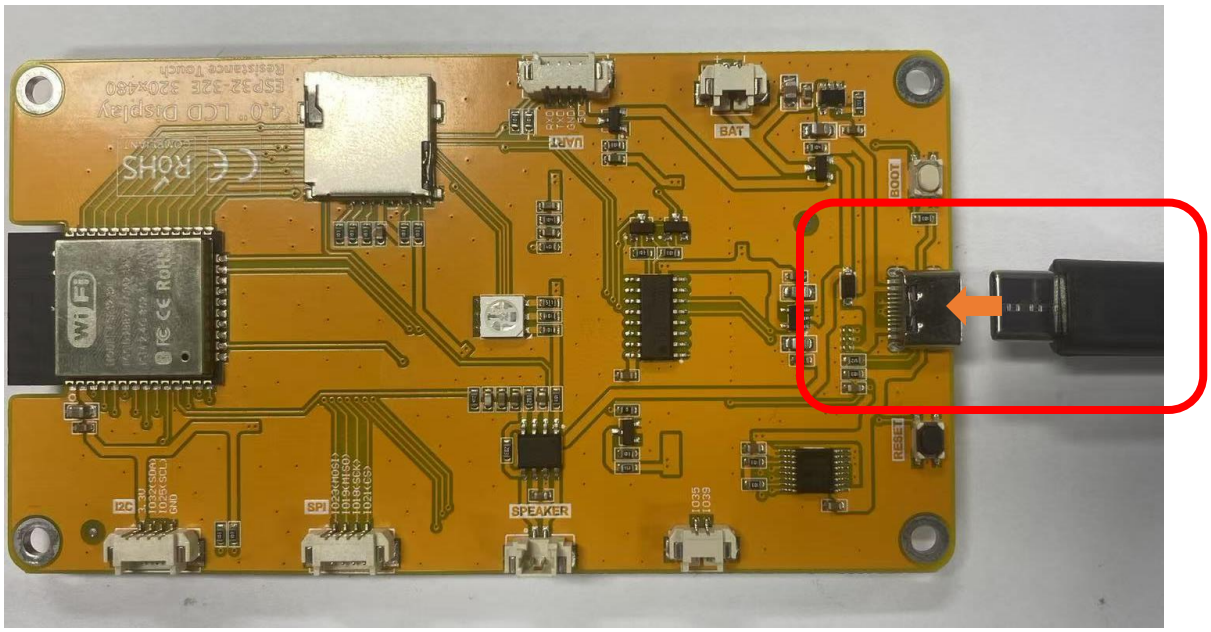
Therefore, by reading the GPIO0 pin state, we can determine whether the button is pressed or not.

Component List

Freenove ESP32 Display x 1 	USB cable x1 

Circuit

Connect Freenove ESP32 Display to the computer with USB cable.



Sketch

Open "**Sketch_03.1_Button_RGB**" folder under **"Freenove_ESP32_Display\Sketch"** 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:   [2025-06-13]
*/

#include <Arduino.h>
#define RED_PIN 22
#define GREEN_PIN 16
#define BLUE_PIN 17
#define KEY_PIN 0

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

void buttonInit(void) {
    pinMode(KEY_PIN, INPUT_PULLUP);
}

bool readButton(void) {
    return digitalRead(KEY_PIN);
}

void rgbInit(void) {
    pinMode(RED_PIN, OUTPUT);
    pinMode(GREEN_PIN, OUTPUT);
    pinMode(BLUE_PIN, OUTPUT);
}

void setRGB(bool redLevel, bool greenLevel, bool blueLevel) {
    digitalWrite(RED_PIN, !redLevel);
    digitalWrite(GREEN_PIN, !greenLevel);
    digitalWrite(BLUE_PIN, !blueLevel);
}

void switchRGB(int value) {
    int colorIndex = value % 4;
```

```
38      switch (colorIndex) {
39        case 1:
40          setRGB(1, 0, 0);
41          break;
42        case 2:
43          setRGB(0, 1, 0);
44          break;
45        case 3:
46          setRGB(0, 0, 1);
47          break;
48        default:
49          setRGB(0, 0, 0);
50          break;
51      }
52  }
53
54  void setup() {
55      rgbInit();
56      buttonInit();
57      Serial.println("ESP32 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.

```
8   #define RED_PIN 22
9   #define GREEN_PIN 16
10  #define BLUE_PIN 17
11  #define KEY_PIN 0
```

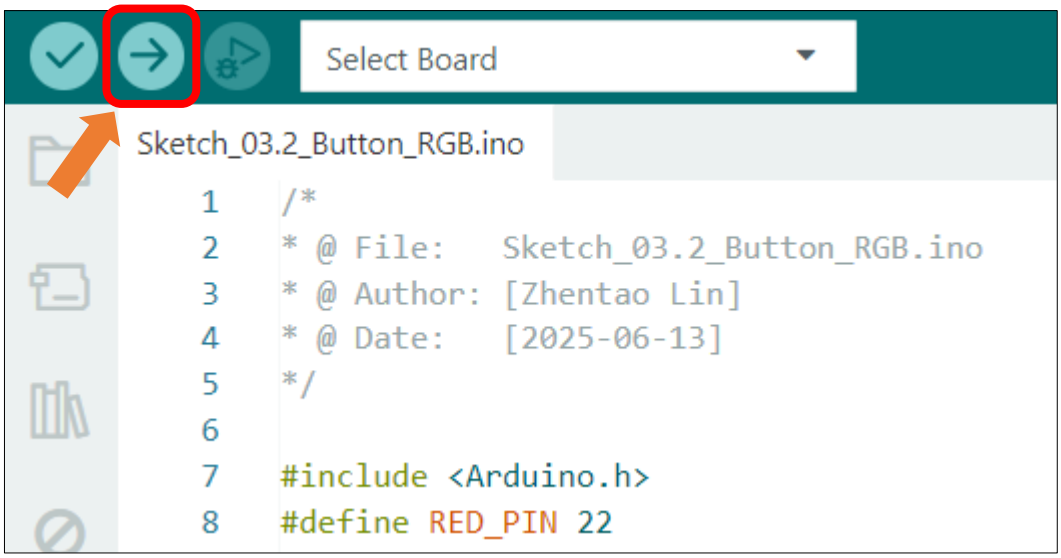
Initialize the RGB LED and the button.

```
55  rgbInit();
56  buttonInit();
```

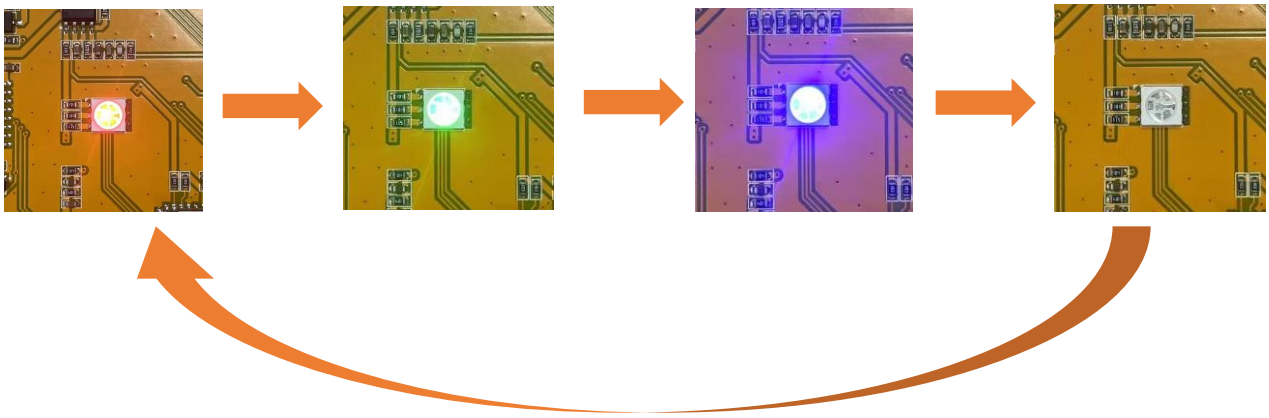
Control the color of the RGB LED through the button.

```
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_Display.



Press the button and the RGB LED will change color in the sequence of red -> blue -> green -> OFF.



Chapter 4 Button Interrupt

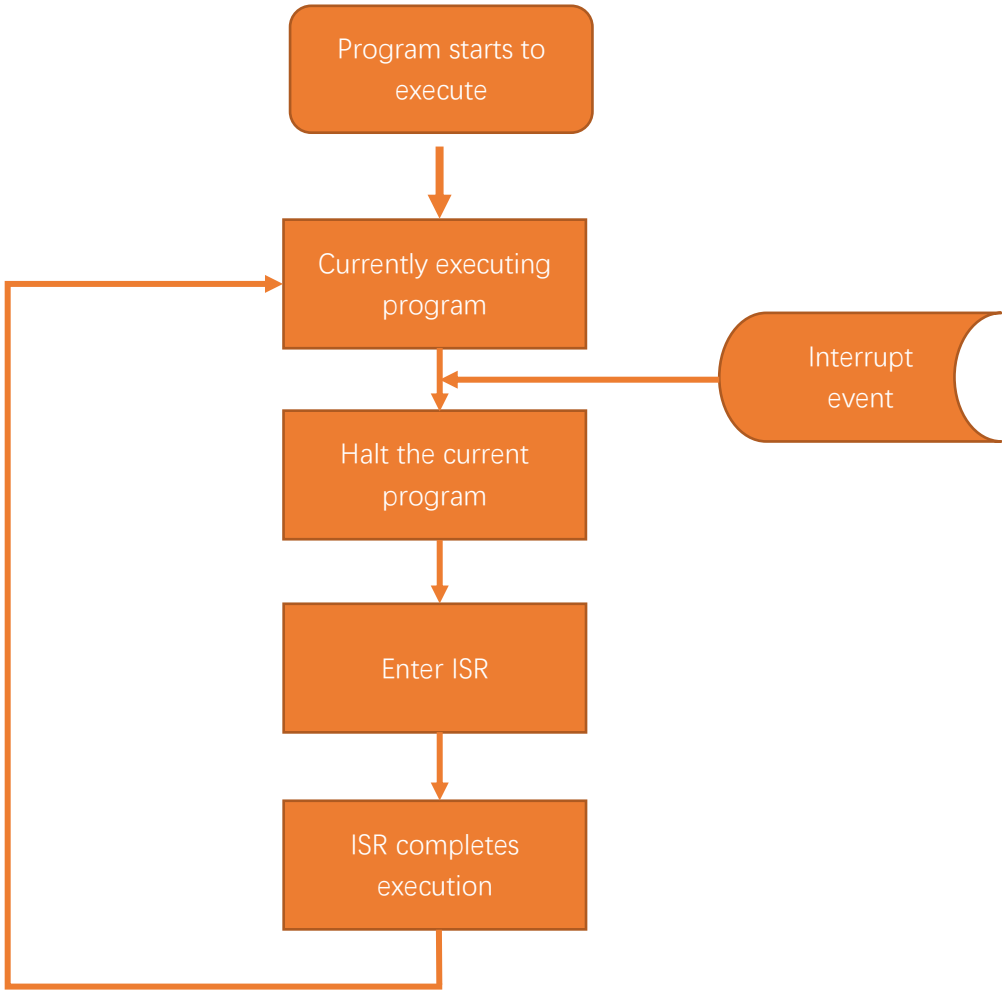
In this chapter will learn to use interrupt to detect the button state.

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.

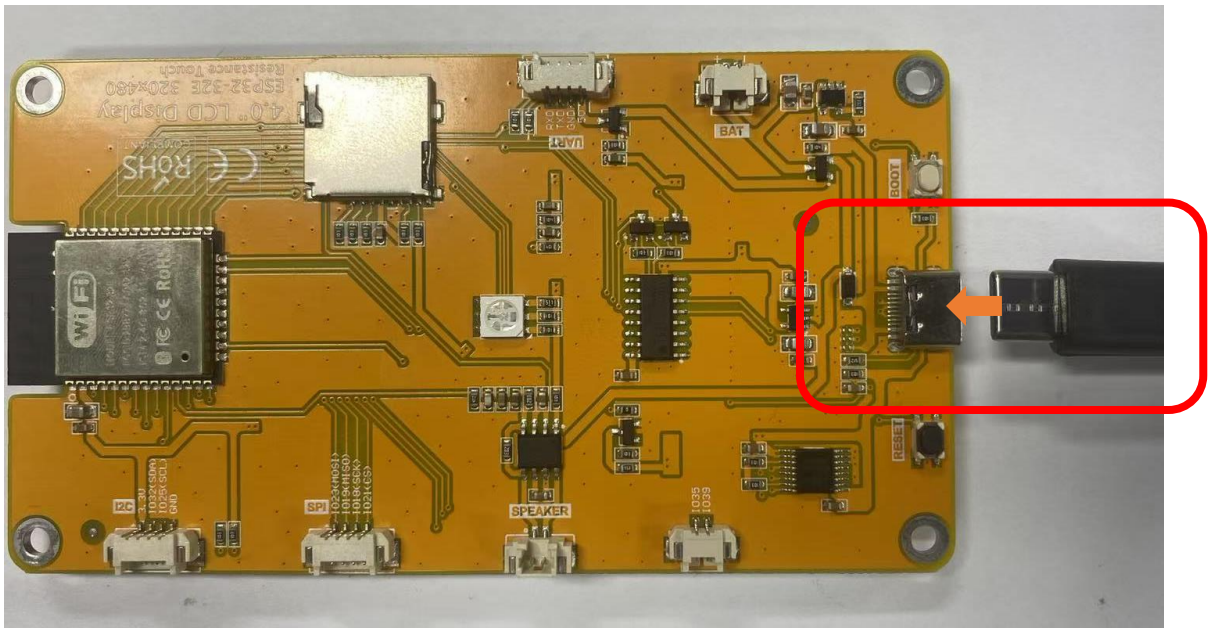


Component List

Freenove ESP32 Display x 1 	USB cable x1 

Circuit

Connect Freenove ESP32 Display to the computer with USB cable.



Sketch

Open "**Sketch_04.1_Button_Interrupt_UART**" folder under **"Freenove_ESP32_Display\Sketch"** 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:   [2025-06-13]
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 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);
```

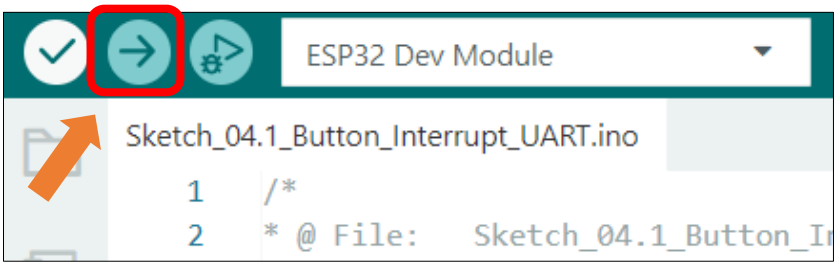
The interrupt processing function, counting how many times the button is pressed.

```
28  void handleInterrupt(void) {
29    portENTER_CRITICAL_ISR(&mux);
30    interruptCounter++;
31    delayDebounce(20);
32    portEXIT_CRITICAL_ISR(&mux);
33  }
```

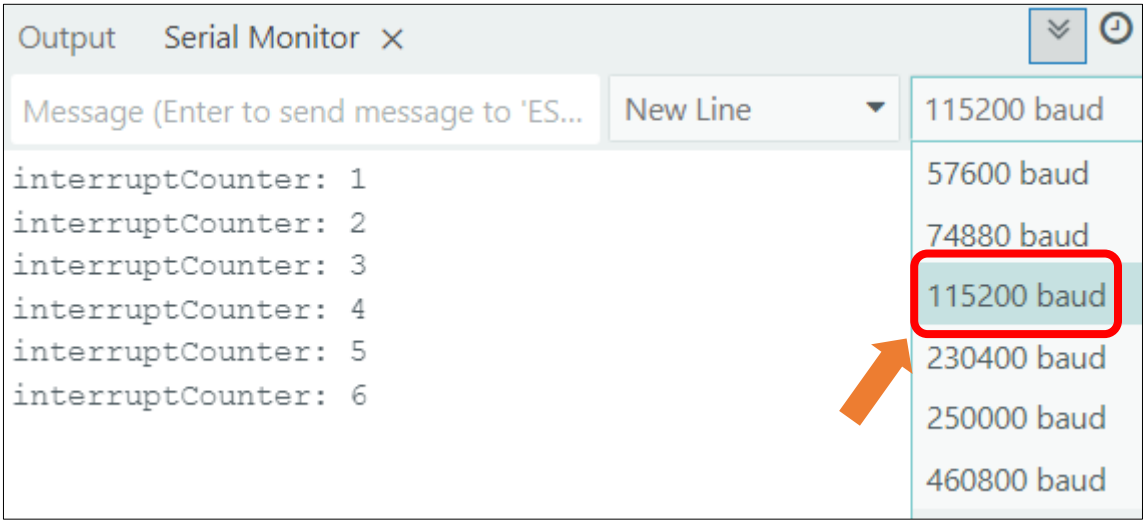
When the number of button presses changes, print new data in the serial monitor.

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

Click "**Upload**" to upload the code to Freenove_ESP32_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)

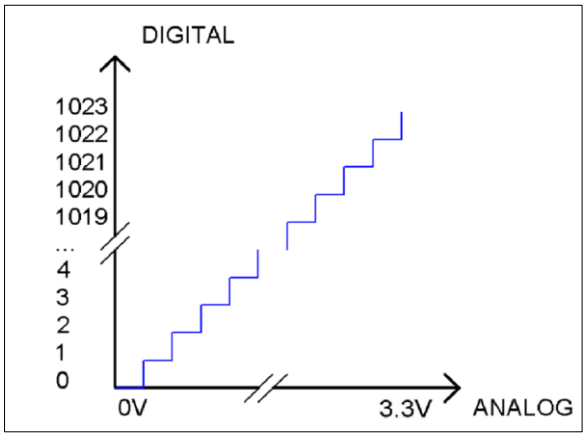
Chapter 5 Battery Voltage

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.

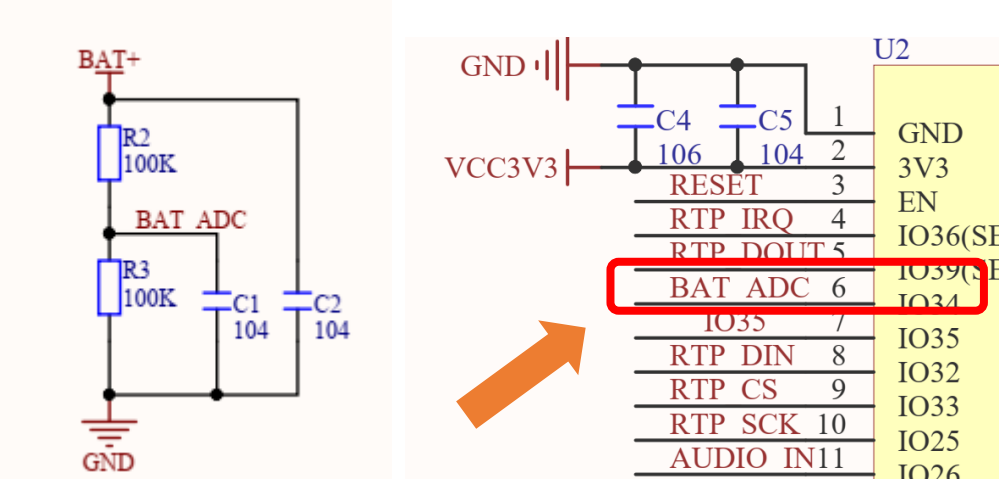


Subsection 1: the analog in rang of 0V---3.3/1023 V corresponds to digital 0;
Subsection 2: the analog in rang of 3.3/1023 V---2*3.3 /1023V corresponds to digital 1;
The following analog will be divided accordingly.
The conversion formula is as follows:

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

Battery Voltage

From the schematic diagram, we can know that Freenove_ESP32_Display reads the battery voltage through the ADC pin (GPIO34).



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.

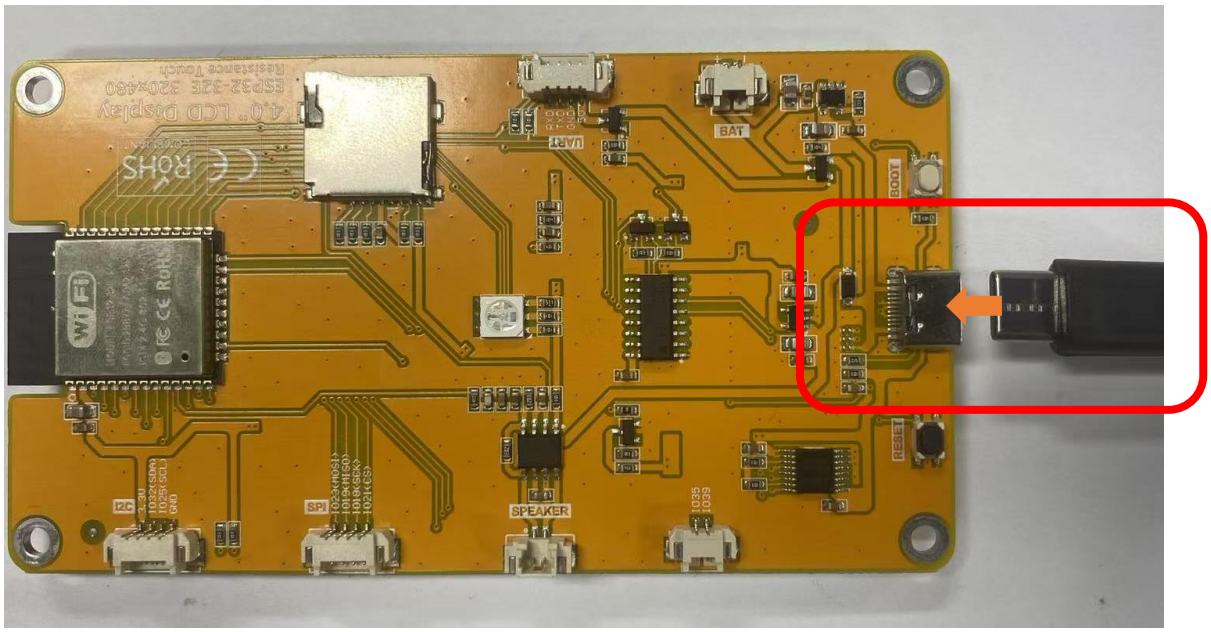
Please note that this kit does not include a lithium battery, please buy one yourself. For more information about battery, please refer to Battery.

Component List

Freenove ESP32 Display x 1 	USB cable x1 

Circuit

Connect Freenove ESP32 Display to the computer with USB cable.



Sketch

Next, we download the code to Freenove_ESP32_Display to test Serial. Open "**Sketch_05.1_Battery_Voltage.ino**" folder under **"Freenove_ESP32_Display\Sketch"** and double-click "**Sketch_05.1_Battery_Voltage.ino**".

Sketch_05.1_Battery_Voltage

The following is the program code:

```
/*
* @ File:    Sketch_05.1_Battery_Voltage.ino
* @ Author: [Zhentao Lin]
* @ Date:    [2025-06-13]
*/

#define BAT_ADC_PIN  34

void setup(){
   Serial.begin(115200);
   pinMode(BAT_ADC_PIN, INPUT);
}

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

```
20  }
```

Code Explanation

Define the pins for the button and RGB LED.

```
7   #define BAT_ADC_PIN  34
```

Set the baud rate to 115200.

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

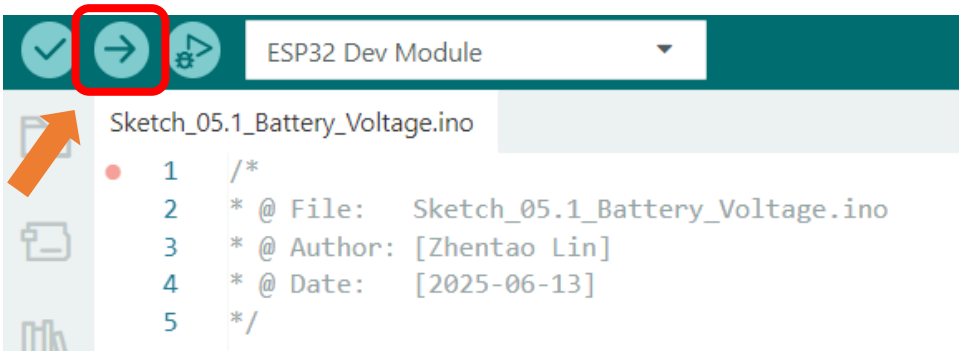
Set the battery voltage detecting pin to input mode.

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

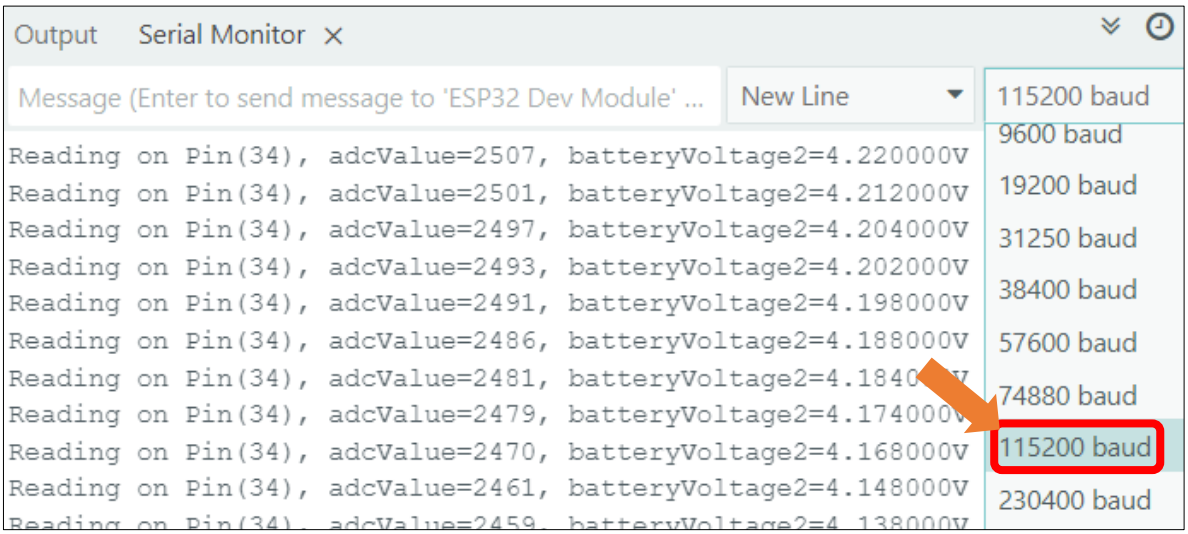
Measure the battery voltage every 300ms and outputs the reading to the serial monitor.

```
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  }
```

Click "**Upload**" to upload the code to Freenove_ESP32_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 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

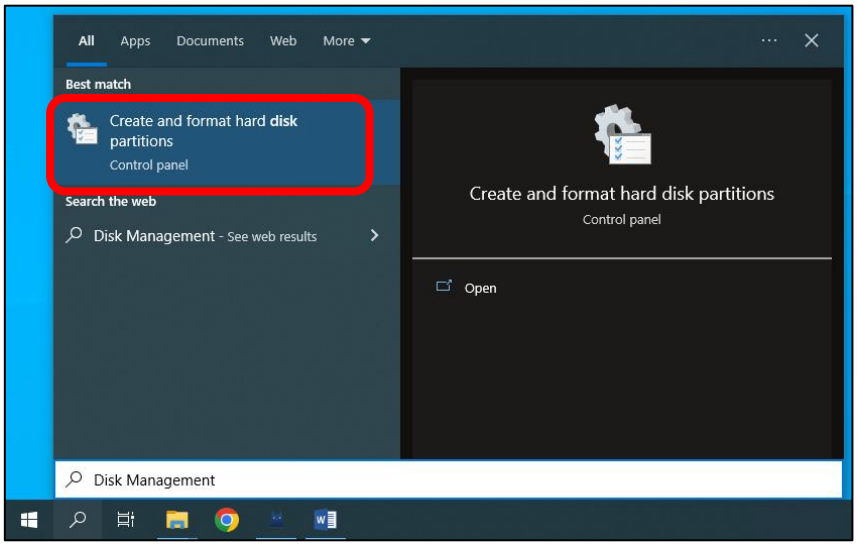
Format SD card

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.

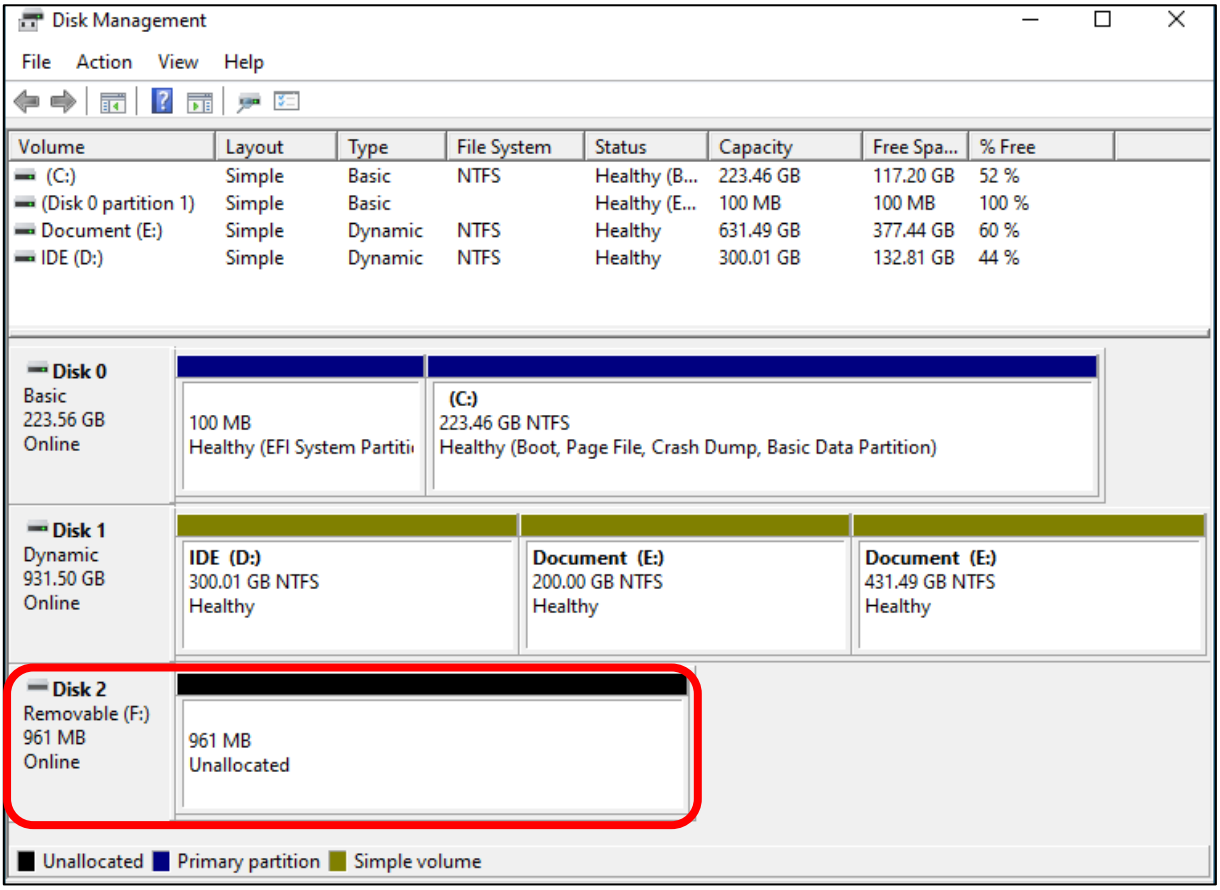
Windows

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

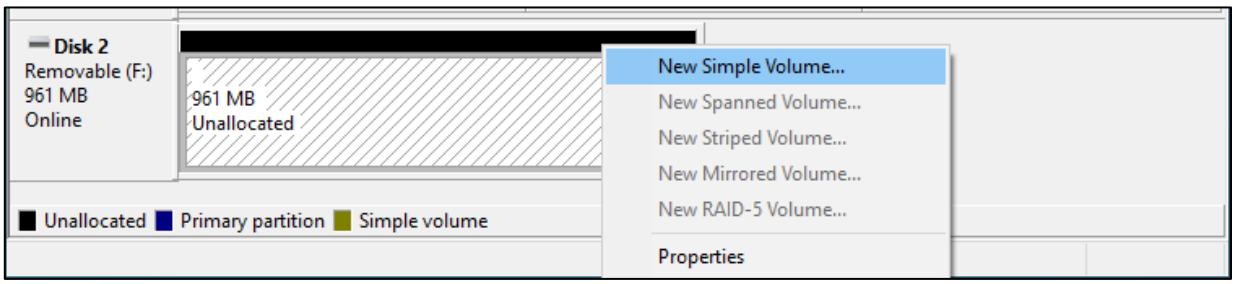
In the Windows search box, enter "Disk Management" and select "Create and format hard disk partitions".



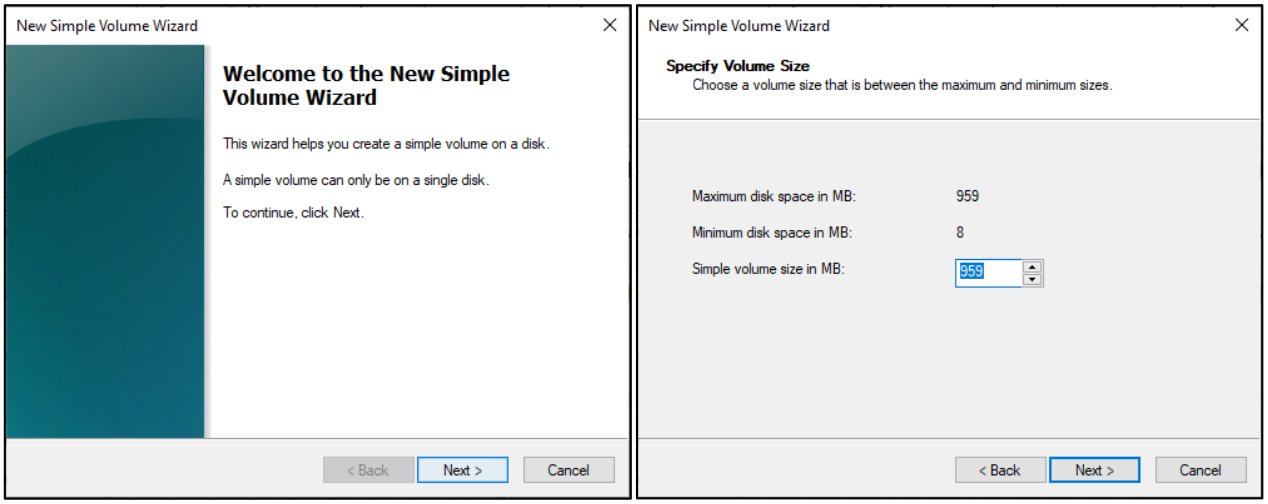
In the newly opened window, locate an unallocated volume with a size close to 1GB.



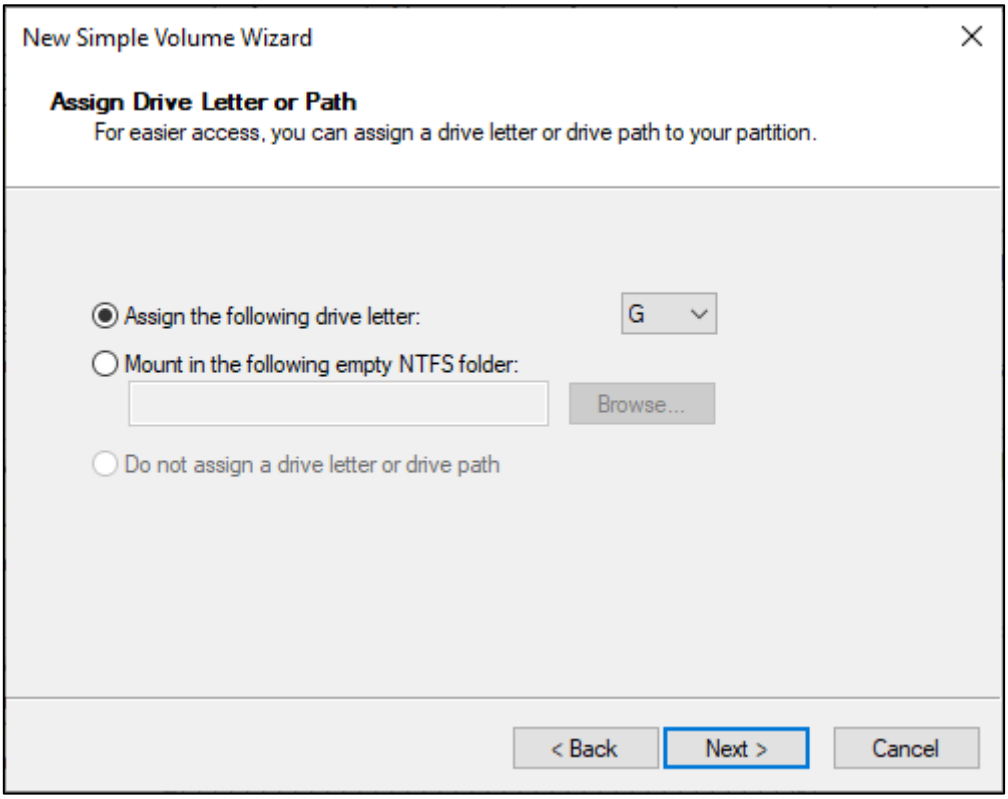
Click to select the volume, right-click and select "New Simple Volume".



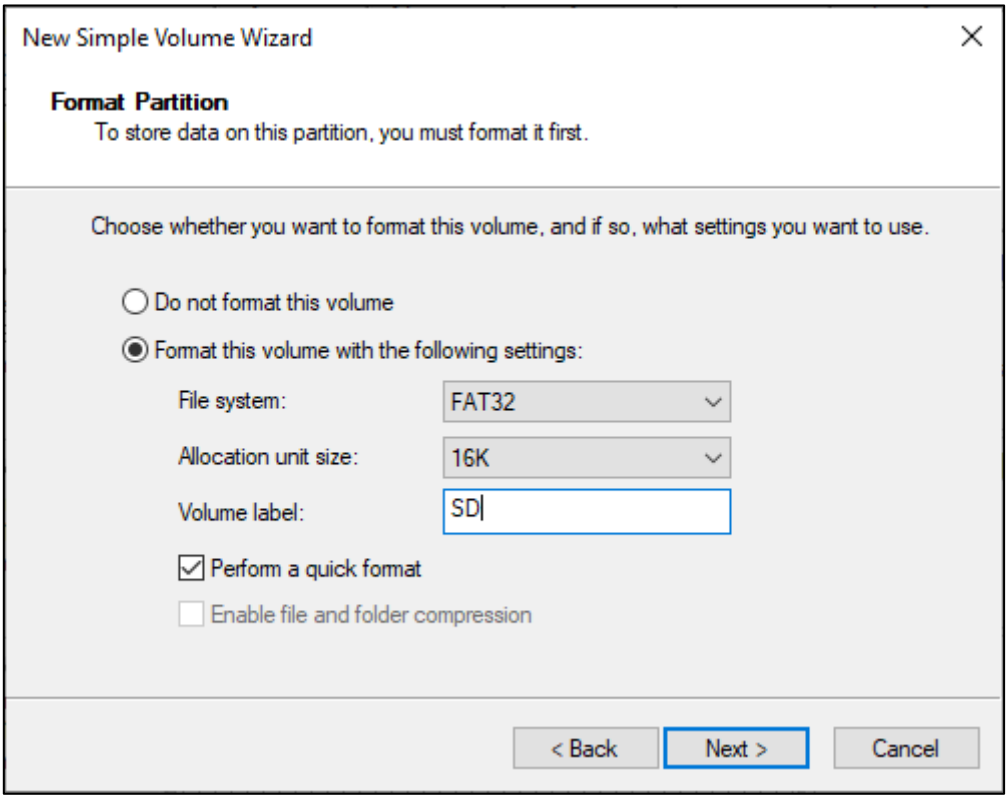
Click Next.



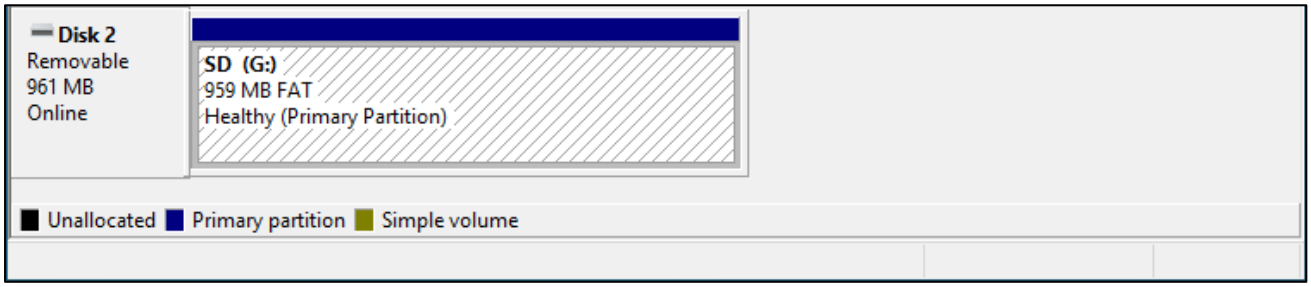
On the right-hand side, you can select a preferred drive letter for the SD card or simply proceed with the default setting by clicking on the "Next" button.



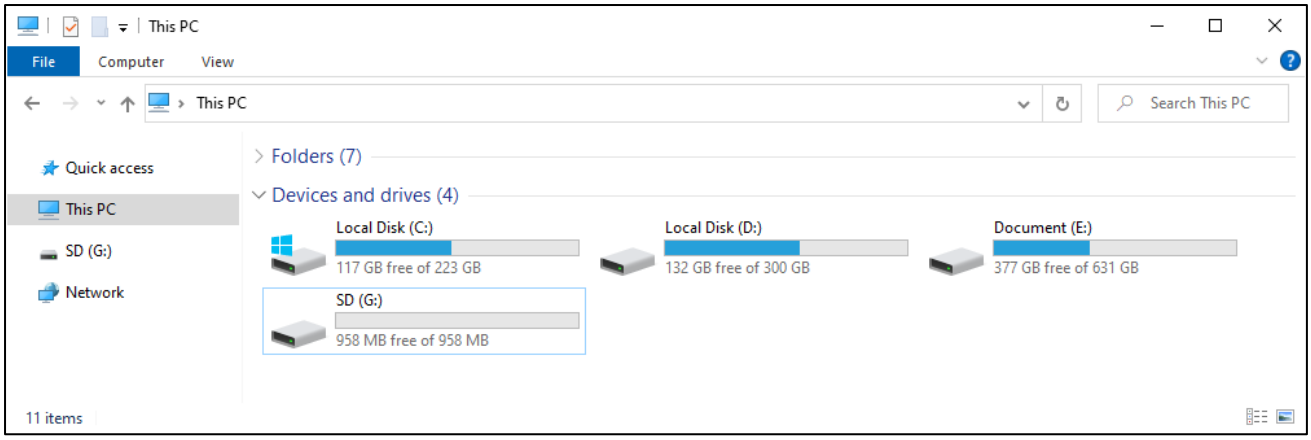
When formatting the SD card, select the file system as FAT (or FAT32) and set the allocation unit size to 16K. You can set the volume label to any name of your choice. Once you have made these selections, click on the "Next" button to proceed.



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

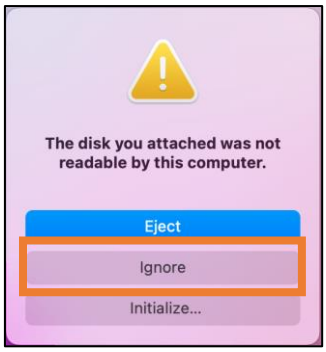


After completing the formatting process, you should be able to see the SD card in the "This PC" section of your computer.

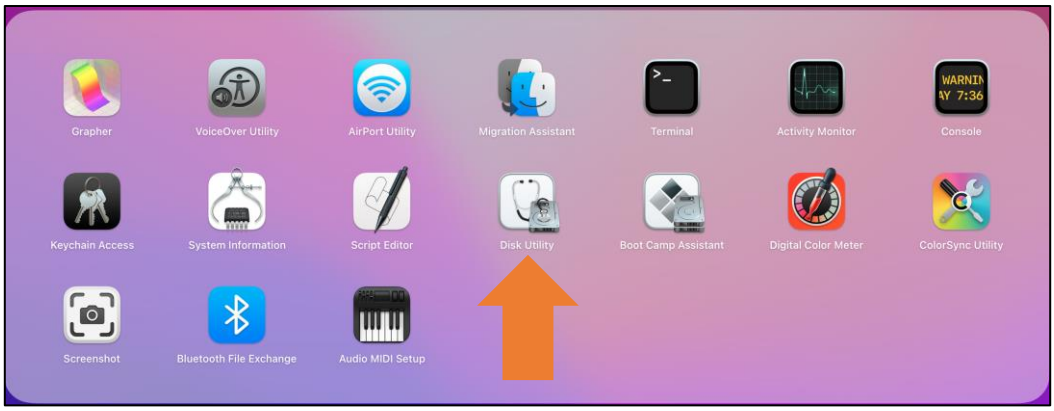


MAC

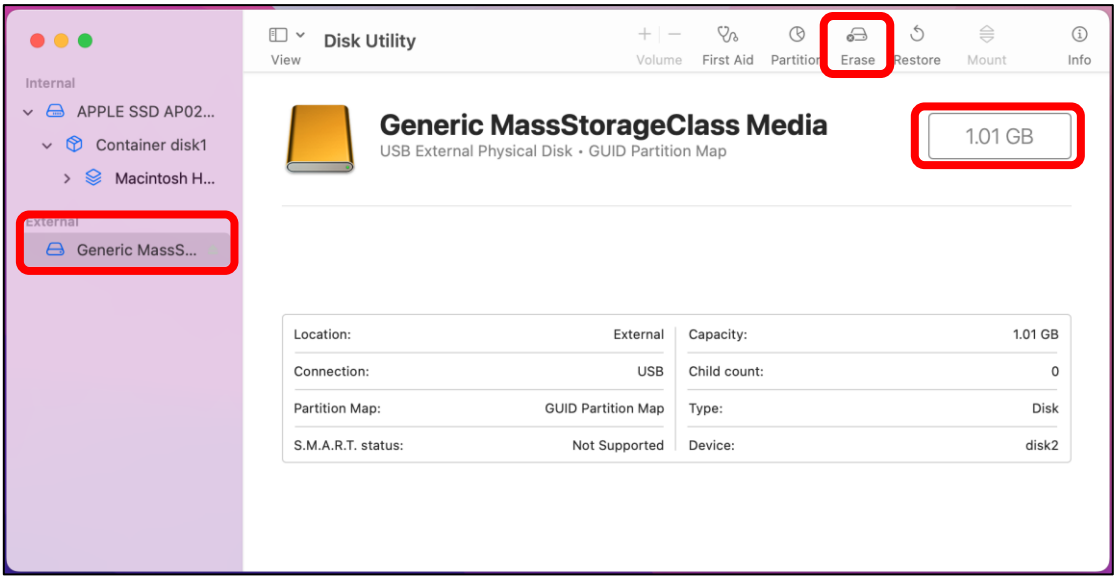
Insert the SD card into the card reader and then insert the card reader into your computer. Some computers may display a prompt with the following message. In this case, please click on the "Ignore" option to proceed.



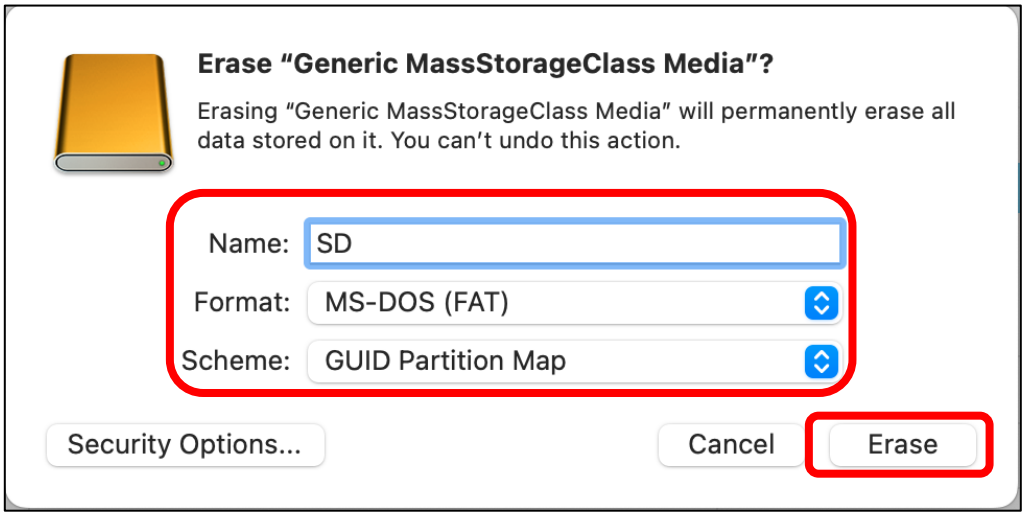
Find "Disk Utility" in the MAC system and click to open it.



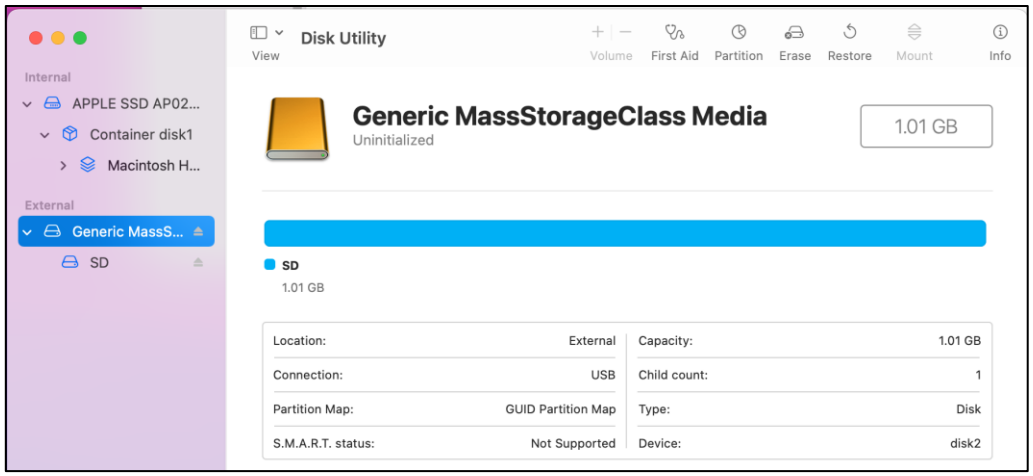
Select "Generic MassStorageClass Media", note that its size is about 1G. It is important to select the correct item to avoid accidentally erasing other device. Once you have verified that you have selected the correct device, click on the "Erase" button to proceed with erasing the SD card.



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



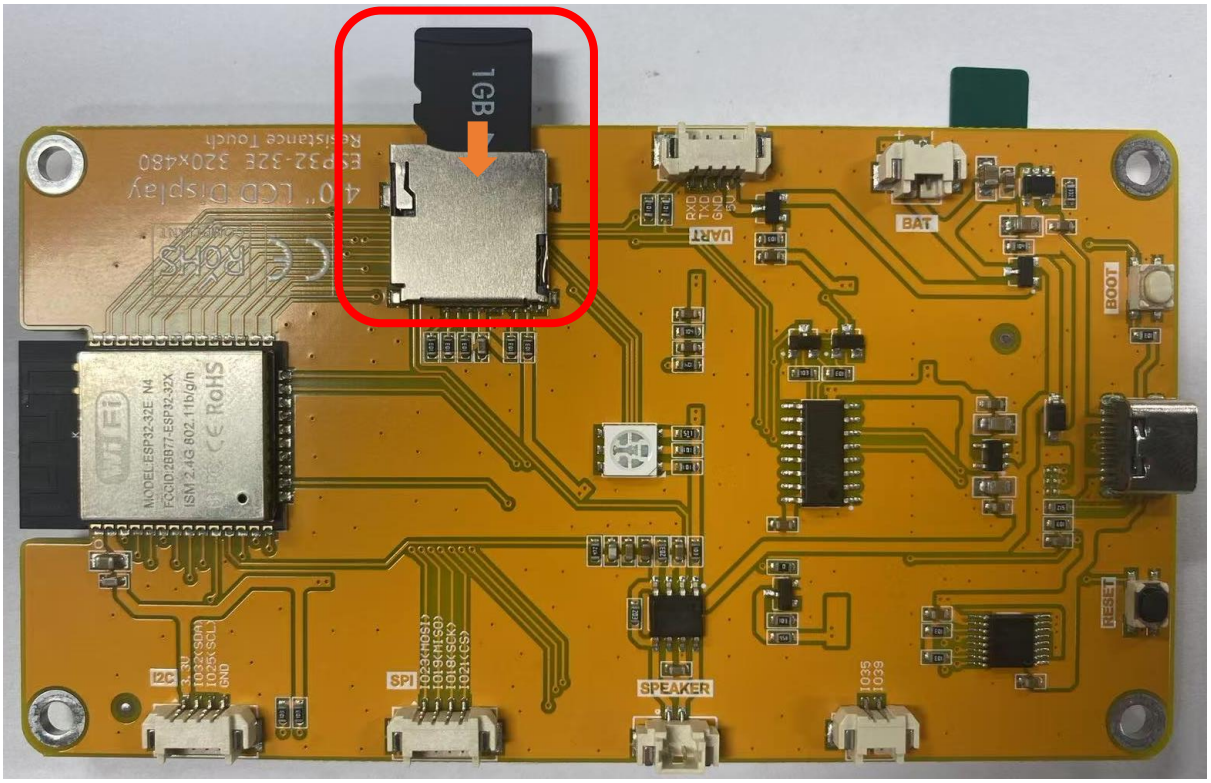
Please wait for the formatting process to complete. Once it is finished, the interface should resemble the image below. You should now be able to see a new disk named "SD" on your desktop.



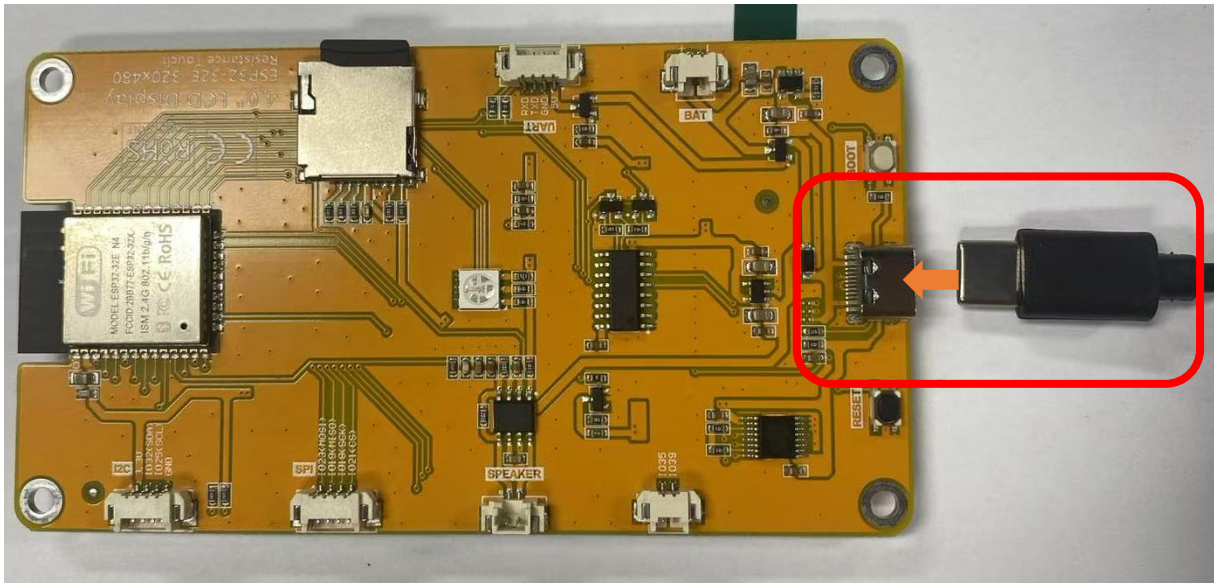
Circuit

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_Display to test Serial. Open "**Sketch_06.1_SD_Test.ino**" folder under **"Freenove_ESP32_Display\Sketch"** and double-click "**Sketch_06.1_SD_Test.ino**".

Sketch_06.1_SD_Test

The following is the program code:

```
/*
* @ File:   Sketch_07.1_SD_Test.ino
* @ Author: [Zhentao Lin]
* @ Date:   [2025-06-14]
*/

#include "sd_read_write.h"
#include "SPI.h"

#define SD_VSPI_SS 5
#define SD_VSPI_SCK 18
#define SD_VSPI_MISO 19
#define SD_VSPI_MOSI 23

SPIClass spi(VSPI);

void setup() {
  Serial.begin(115200);
  spi.begin(SD_VSPI_SCK, SD_VSPI_MISO, SD_VSPI_MOSI, SD_VSPI_SS);
  if (!SD.begin(SD_VSPI_SS, spi)) {
    Serial.println(F("SD.begin failed!"));
    while (1) delay(1);
  }
  Serial.println("\r\nInitialisation done.");
  uint8_t cardType = SD.cardType();
  if (cardType == CARD_NONE) {
    Serial.println("No SD card attached");
    return;
  }

  Serial.print("SD Card Type: ");
  if (cardType == CARD_MMC) {
    Serial.println("MMC");
  } else if (cardType == CARD_SD) {
    Serial.println("SDSC");
  } else if (cardType == CARD_SDHC) {
    Serial.println("SDHC");
```

```
38      } else {
39        Serial.println("UNKNOWN");
40      }
41
42      uint64_t cardSize = SD.cardSize() / (1024 * 1024);
43      Serial.printf("SD Card Size: %lluMB\n", cardSize);
44
45      listDir(SD, "/", 0);
46
47      createDir(SD, "/mydir");
48      listDir(SD, "/", 0);
49
50      removeDir(SD, "/mydir");
51      listDir(SD, "/", 2);
52
53      writeFile(SD, "/hello.txt", "Hello ");
54      appendFile(SD, "/hello.txt", "World!\n");
55      readFile(SD, "/hello.txt");
56
57      deleteFile(SD, "/foo.txt");
58      renameFile(SD, "/hello.txt", "/foo.txt");
59      readFile(SD, "/foo.txt");
60
61      testFileIO(SD, "/test.txt");
62
63      Serial.printf("Total space: %lluMB\r\n", SD.totalBytes() / (1024 * 1024));
64      Serial.printf("Used space: %lluMB\r\n", SD.usedBytes() / (1024 * 1024));
65  }
66
67  void loop() {
68      delay(10000);
69  }
```

Code Explanation

Include necessary header files.

```
7   #include "sd_read_write.h"
8   #include "SPI.h"
```

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

```
10  #define SD_VSPI_SS 5
11  #define SD_VSPI_SCK 18
12  #define SD_VSPI_MISO 19
13  #define SD_VSPI_MOSI 23
```

Initialize SPI bus.

```
76  SPIClass spi(VSPI);
```

Set the baud rate to 115200.

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

Start SPI communication and try to mount the SD card.

```
19  spi.begin(SD_VSPI_SCK, SD_VSPI_MISO, SD_VSPI_MOSI, SD_VSPI_SS);
20  if (!SD.begin(SD_VSPI_SS, spi)) {
21      Serial.println(F("SD.begin failed!"));
22      while (1) delay(1);
23  }
```

List all files under the root directory.

```
45  listDir(SD, "/", 0);
```

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

```
47  createDir(SD, "/mydir");
48  listDir(SD, "/", 0);
```

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

```
50  removeDir(SD, "/mydir");
51  listDir(SD, "/", 0);
```

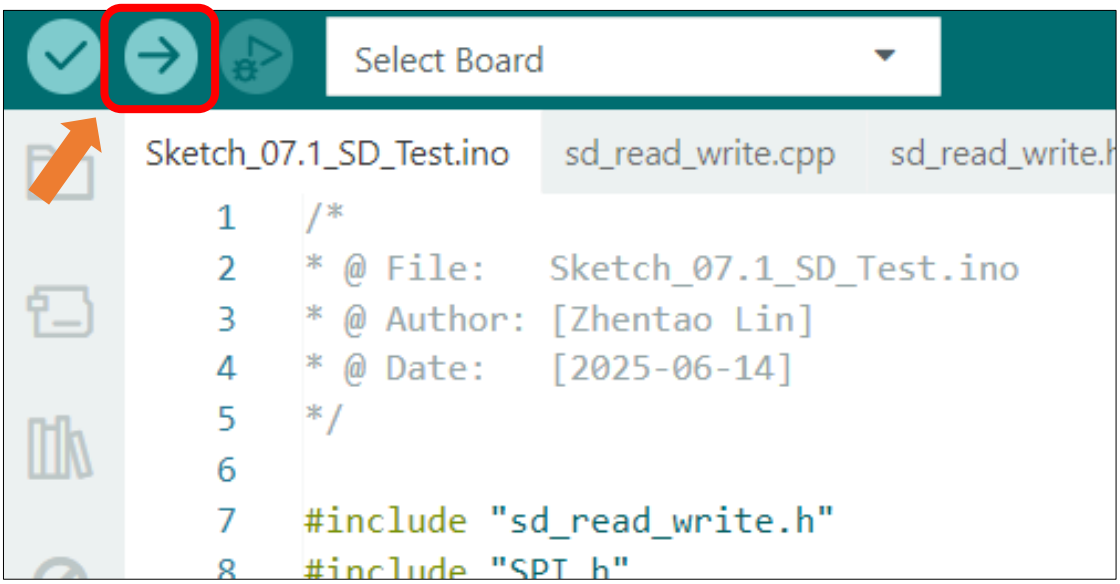
Write "Hello" in the hello.txt file, append "World!" to it, then read and print the file contents.

```
53  writeFile(SD, "/hello.txt", "Hello ");
54  appendFile(SD, "/hello.txt", "World!\n");
55  readFile(SD, "/hello.txt");
```

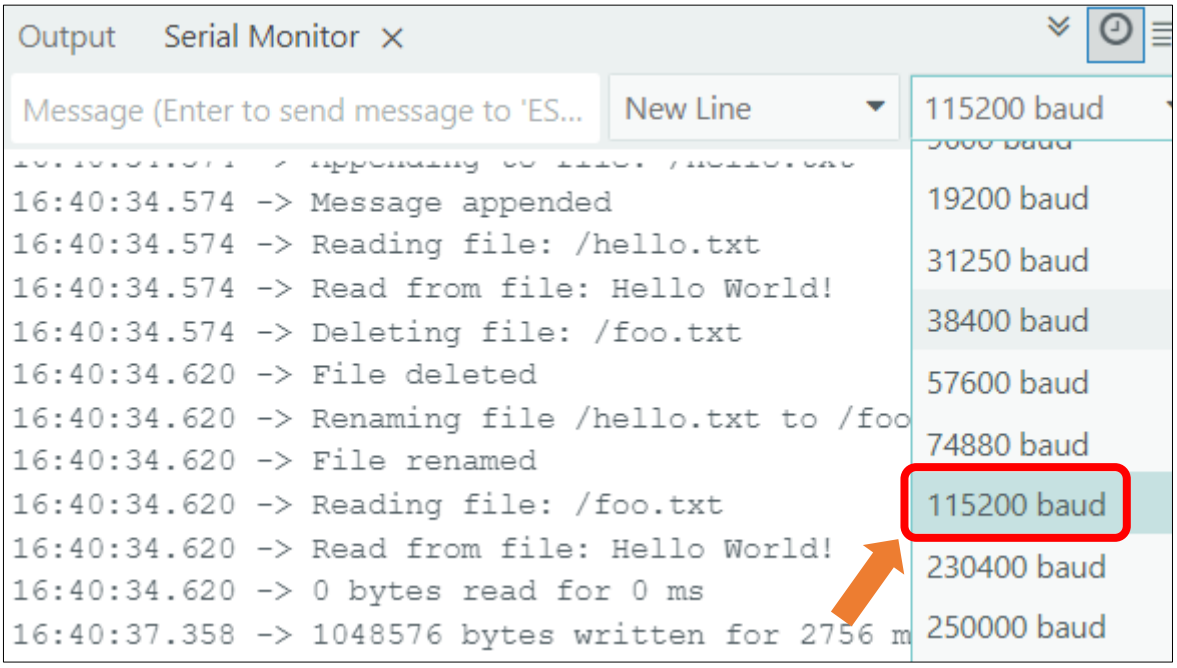
Dele the file “foo.txt”. Rename hello.txt to foo.txt and read the file.

```
53  deleteFile(SD, "/foo.txt");
54  renameFile(SD, "/hello.txt", "/foo.txt");
55  readFile(SD, "/foo.txt");
```

Click “**Upload**” to upload the code to Freenove_ESP32_Display.



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



Chapter 7 Play MP3

Project 7.1 Play MP3 SD by DAC

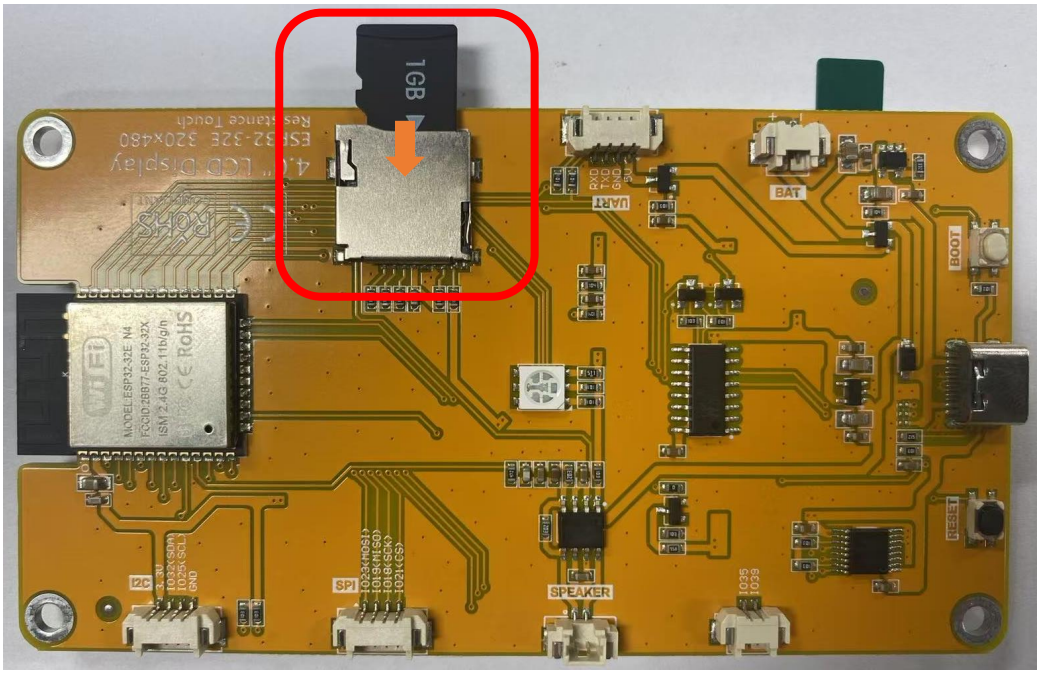
Component List

Freenove ESP32 Display x 1 	USB cable x1 

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

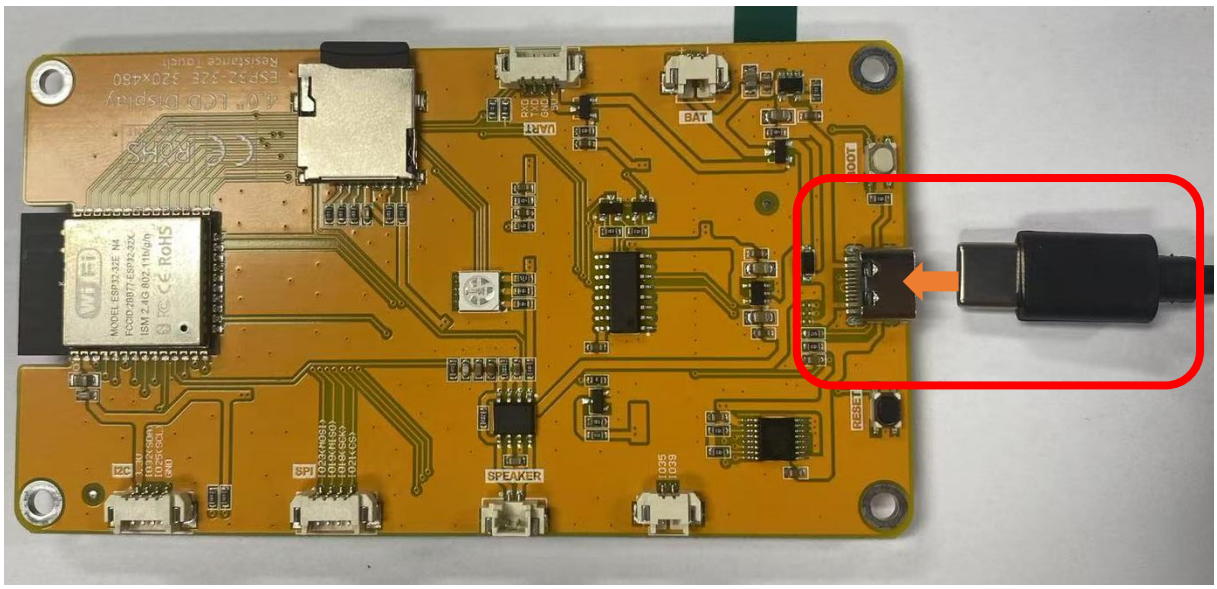
Circuit

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



Connect the speaker.

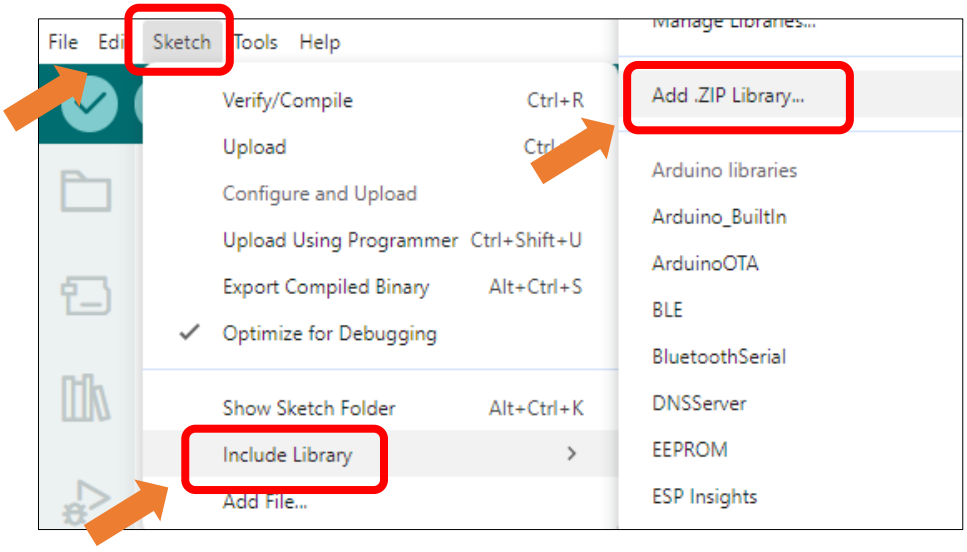
Connect Freenove ESP32-S3 to the computer using the USB cable.



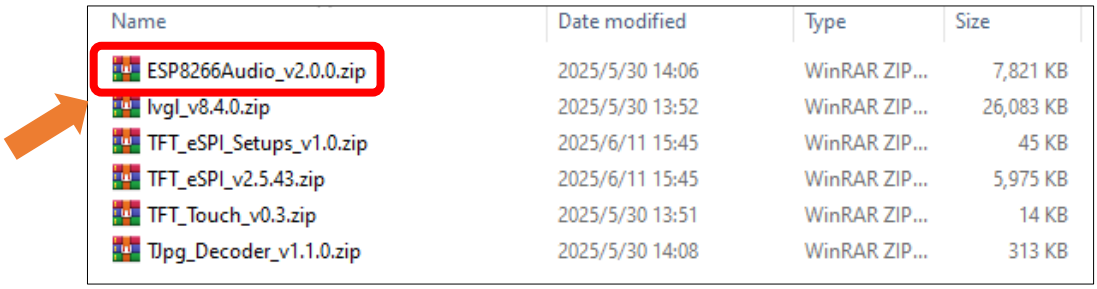
Sketch

Install the needed libraries.

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



Select **ESP8266Audio_v2.0.0.zip**



Next, we download the code to Freenove_ESP32_Display to test. Open "**Sketch_07.1_Play_MP3_SD_by_DAC**" folder under **"Freenove_ESP32_Display\Sketch"** and double-click "**Sketch_07.1_Play_MP3_SD_by_DAC.ino**".

Sketch_07.1_Play_MP3_SD_by_DAC

The following is the program code:

```
/*
* @ File:   Sketch_08.1_Play_MP3_SD_by_DAC.ino
* @ Author: [Zhentao Lin]
* @ Date:   [2025-06-14]
*/
#include <Arduino.h>
#include <WiFi.h>
#include "AudioFileSourceSD.h"
#include "AudioFileSourceID3.h"
#include "AudioGeneratorMP3.h"
#include "AudioOutputI2S.h"

#define AUDIO_EN 4
#define SD_VSPI_SS 5
#define SD_VSPI_MOSI 23
#define SD_VSPI_MISO 19
#define SD_VSPI_SCK 18

AudioFileSourceSD *file;
AudioFileSourceID3 *id3;
AudioOutputI2S *out;
AudioGeneratorMP3 *mp3;

SPIClass *vspi = NULL;

void playMP3(char *filename) {
   file = new AudioFileSourceSD(filename);
   id3 = new AudioFileSourceID3(file);
   out = new AudioOutputI2S(0, 1);
   out->SetGain(1);
```

```
31      mp3 = new AudioGeneratorMP3();
32      mp3->begin(id3, out);
33  }
34
35  void setup() {
36      Serial.begin(115200);
37      Serial.println();
38      WiFi.mode(WIFI_OFF);
39      pinMode(AUDIO_EN, OUTPUT);
40      digitalWrite(AUDIO_EN, LOW);
41      delay(500);
42      vspi = new SPIClass(VSPI);
43      vspi->begin(SD_VSPI_SCK, SD_VSPI_MISO, SD_VSPI_MOSI, SD_VSPI_SS);
44      if (!SD.begin(SD_VSPI_SS, *vspi)) {
45        Serial.println(F("SD.begin failed!"));
46        while (1) delay(1);
47      }
48      playMP3("/Olsen-Banden.mp3");
49  }
50
51  void loop() {
52      while (mp3->isRunning()) {
53        if (!mp3->loop()) mp3->stop();
54      }
55  }
```

Code Explanation

Include necessary header files.

```
6   #include <Arduino.h>
7   #include <WiFi.h>
8   #include "AudioFileSourceSD.h"
9   #include "AudioFileSourceID3.h"
10  #include "AudioGeneratorMP3.h"
11  #include "AudioOutputI2S.h"
```

Define the pins.

```
13  #define AUDIO_EN 4
14  #define SD_VSPI_SS 5
15  #define SD_VSPI_MOSI 23
16  #define SD_VSPI_MISO 19
17  #define SD_VSPI_SCK 18
```

Declare audio objects.

```
19  AudioFileSourceSD *file;
20  AudioFileSourceID3 *id3;
21  AudioOutputI2S *out;
22  AudioGeneratorMP3 *mp3;
```

Set the baud rate to 115200

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

Initialize MP3 decoder and play the music. To play other music, please change the name in the code.

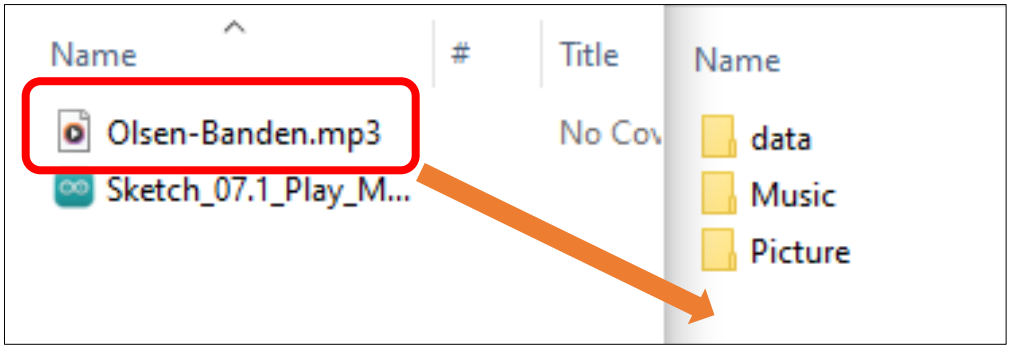
```
19  playMP3("/Olsen-Banden.mp3");
```

Read audio data and play it.

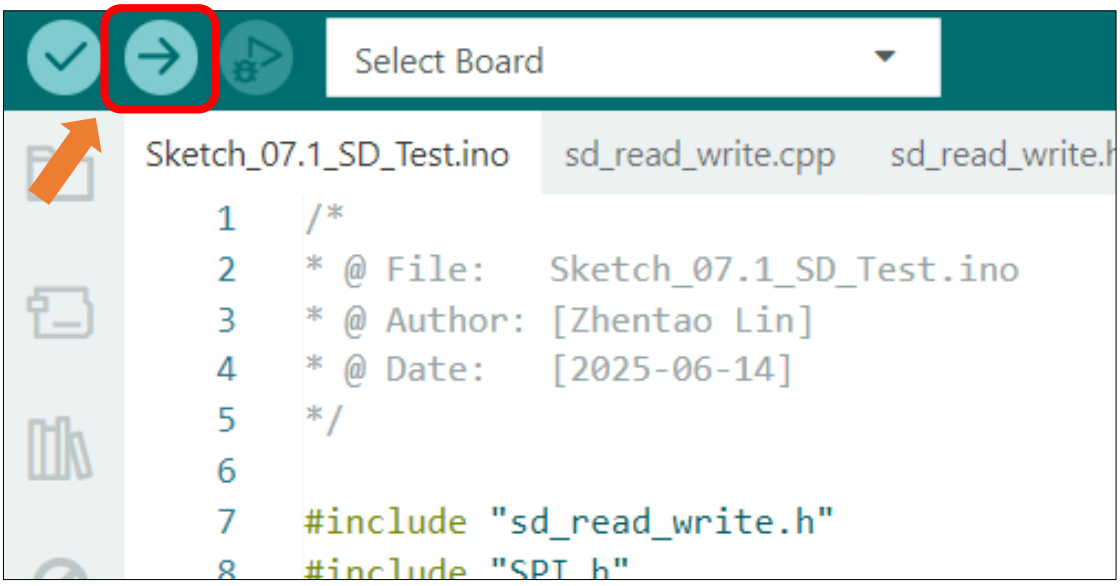
```
51  void loop() {
52    while (mp3->isRunning()) {
53      if (!mp3->loop()) mp3->stop();
54    }
55  }
```

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

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



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



The speaker plays the music in the SD card.

Chapter 8 BLE

Project 8.1 BLE USART

Component List

Freenove ESP32 Display x 1 	USB cable x1 

Component Knowledge

BLE (Bluetooth Low Energy)

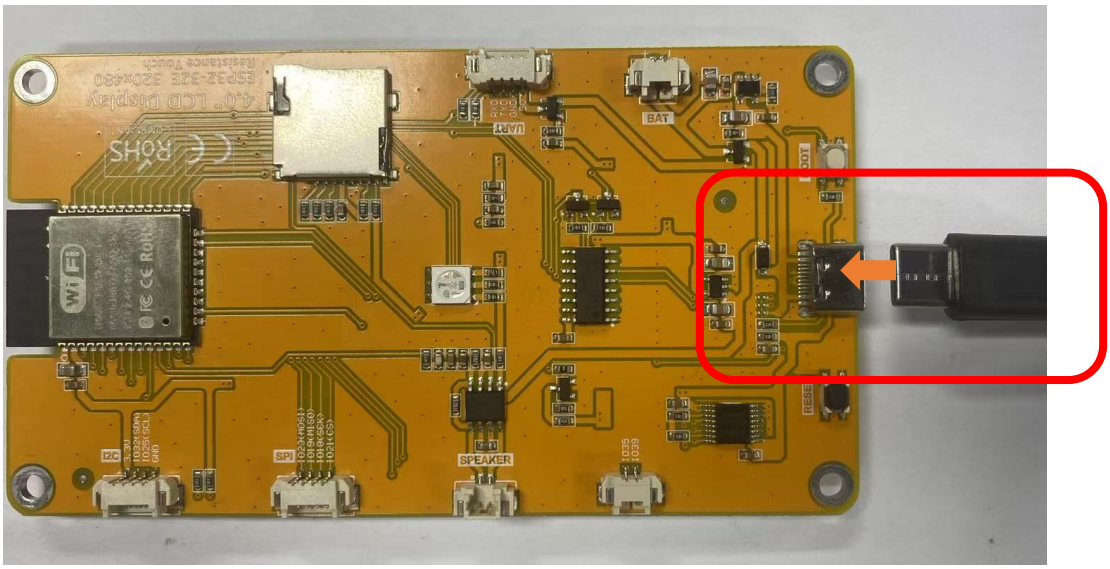
Low Energy Bluetooth (BLE) is a new feature introduced in the Bluetooth 4.0 specification, specifically designed for low-power devices and suitable for applications involving intermittent transmission of small amounts of data.

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_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_Display to test. Open"**Sketch_08.1_BLE_USART**" folder under **"Freenove_ESP32_Display\Sketch"** and double-click "**Sketch_08.1_BLE_USART.ino**".

Sketch_08.1_BLE_USART

The following is the program code:

```
/*
* @ File:   Sketch_09.1_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"

class MyServerCallbacks: public BLEServerCallbacks {
```

```
24        void onConnect(BLEServer* pServer) {
25          deviceConnected = true;
26        };
27        void onDisconnect(BLEServer* pServer) {
28          deviceConnected = false;
29        }
30    };
31
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++){
38              rxload +=(char)rxValue[i];
39            }
40          }
41        }
42    };
43
44    void setupBLE(String BLEName){
45      const char *ble_name=BLEName.c_str();
46      BLEDevice::init(ble_name);
47      BLEServer *pServer = BLEDevice::createServer();
48      pServer->setCallbacks(new MyServerCallbacks());
49      BLEService *pService = pServer->createService(SERVICE_UUID);
50      pCharacteristic=
51    pService->createCharacteristic(CHARACTERISTIC_UUID_TX,BLECharacteristic::PROPERTY_NOTIFY);
52      pCharacteristic->addDescriptor(new BLE2902());
53      BLECharacteristic *pCharacteristic =
54    pService->createCharacteristic(CHARACTERISTIC_UUID_RX,BLECharacteristic::PROPERTY_WRITE);
55      pCharacteristic->setCallbacks(new MyCallbacks());
56      pService->start();
57      pServer->getAdvertising()->start();
58      Serial.println("Waiting a client connection to notify...");
59    }
60
61    void setup() {
62      Serial.begin(115200);
63      setupBLE("ESP32_BLE");
64    }
65
66    void loop() {
67      long now = millis();
```

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

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++){
38            rxload +=(char)rxValue[i];
39          }
```

```
        }
    }
};
```

Set the baud rate to 115200.

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

BLE Device Initialization, Service Creation, and Characteristic Setup

```
setupBLE("ESP32_BLE");
```

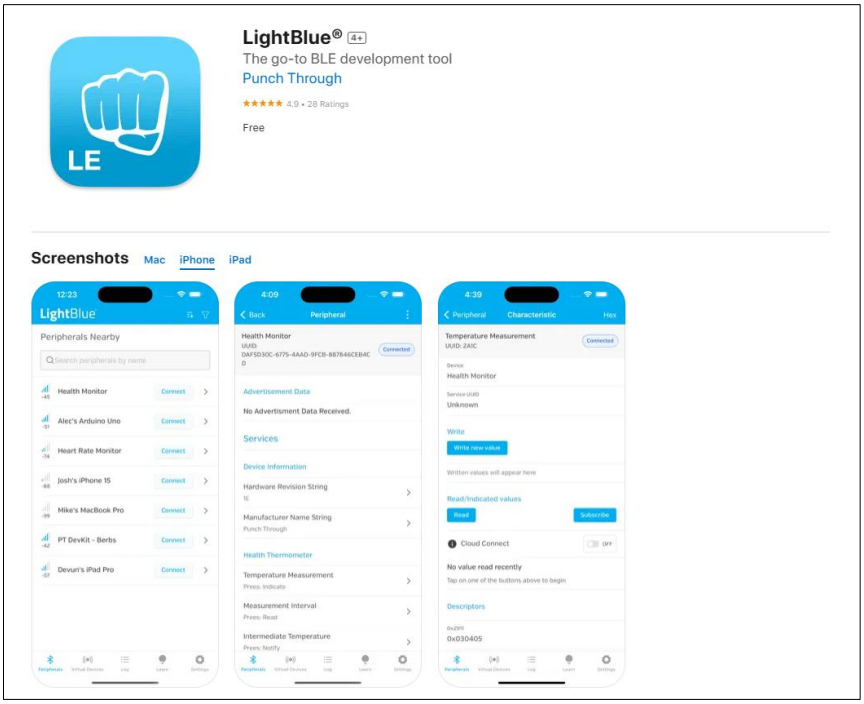
The Loop function check the connection status and serial data every 100 milliseconds.

```
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;
  }
}
```

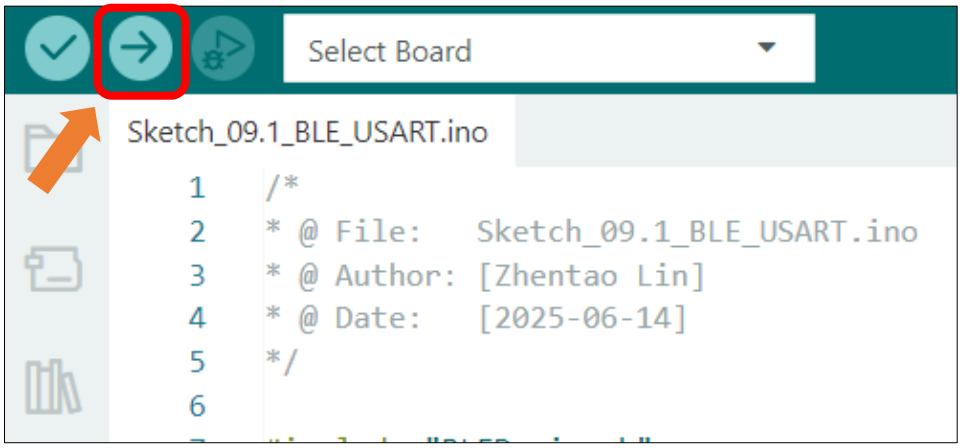
LightBlue

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

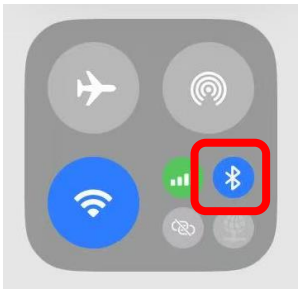
https://apps.apple.com/us/app/lightblue/id557428110



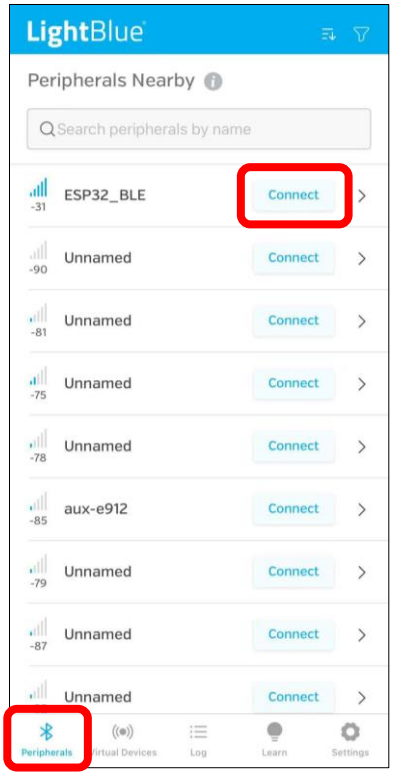
Click "**Upload**" to upload the code to Freenove ESP32 Display.

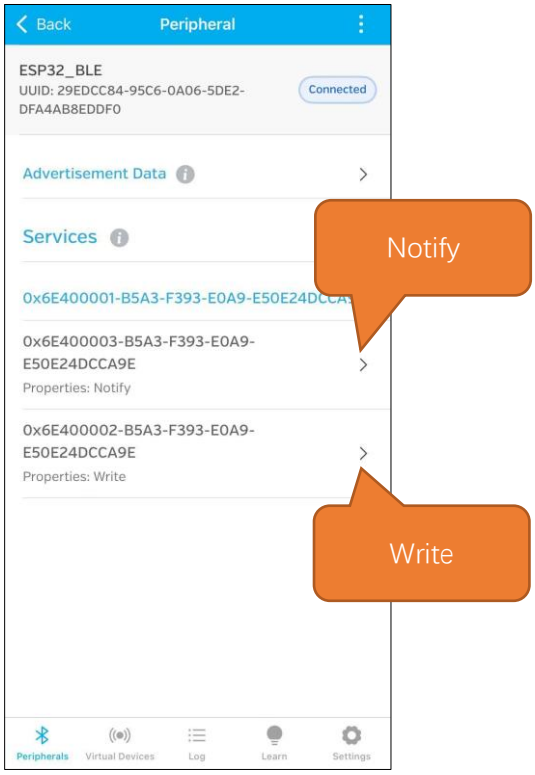


Turn ON Bluetooth on your phone, and open the LightBlue APP.



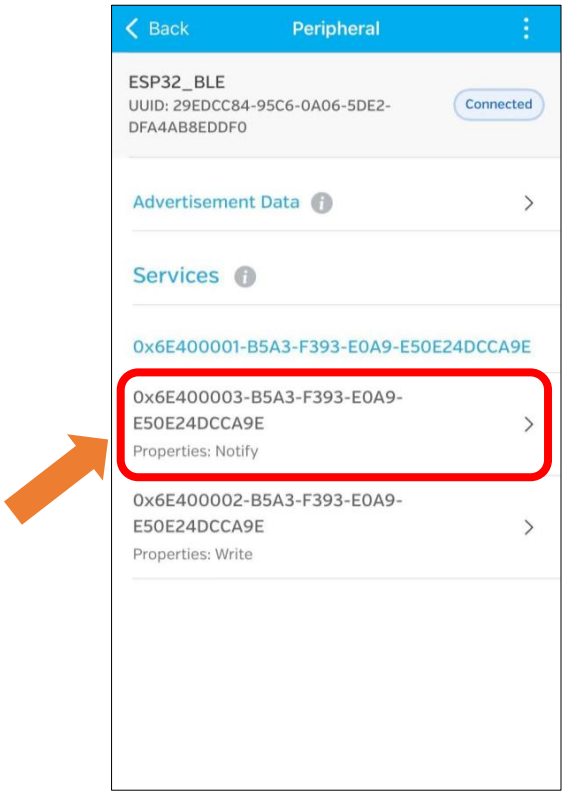
In the Scan page, swipe down to refresh the name of Bluetooth that the phone searches for. Click the Connection button of ESP32_BLE.



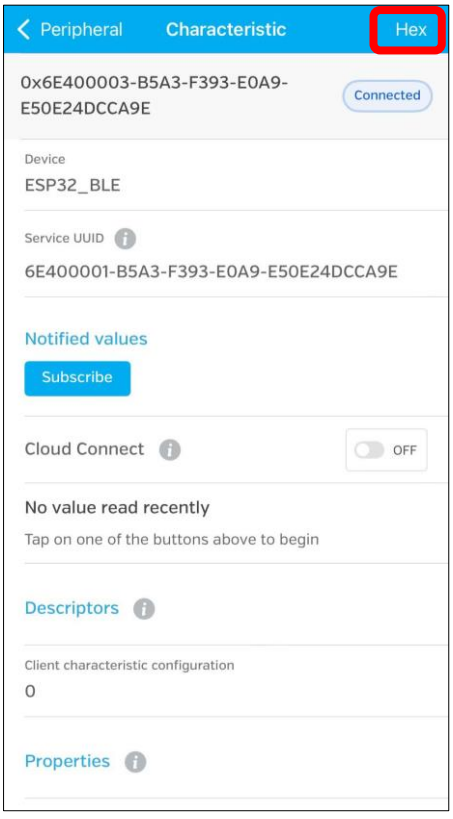


Receiving Data

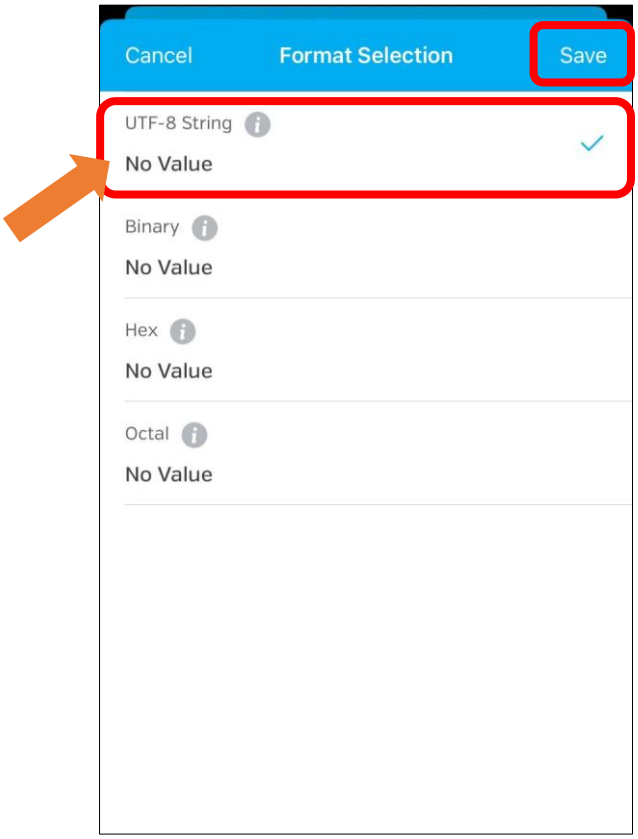
Click Notify



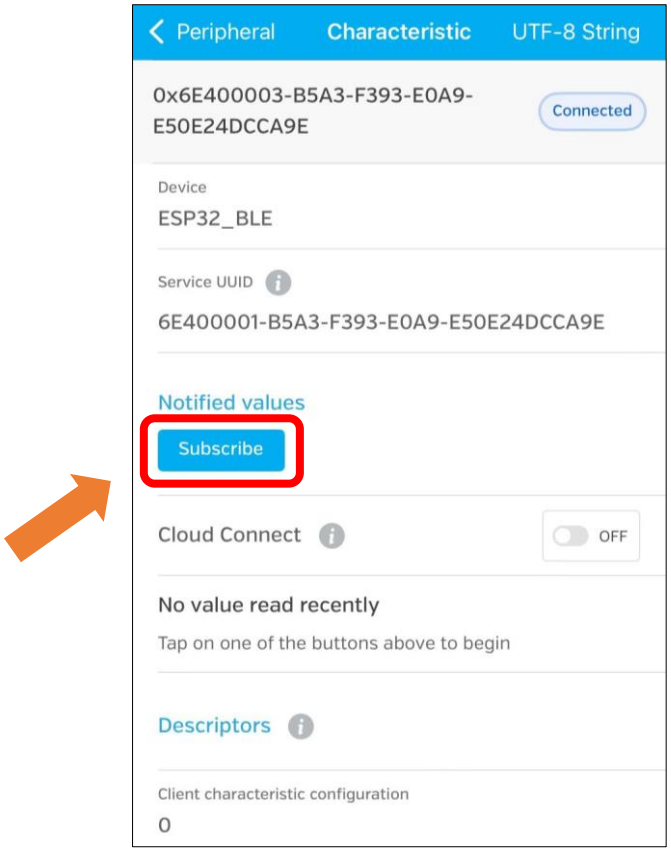
Click the top right corner to change the string type.



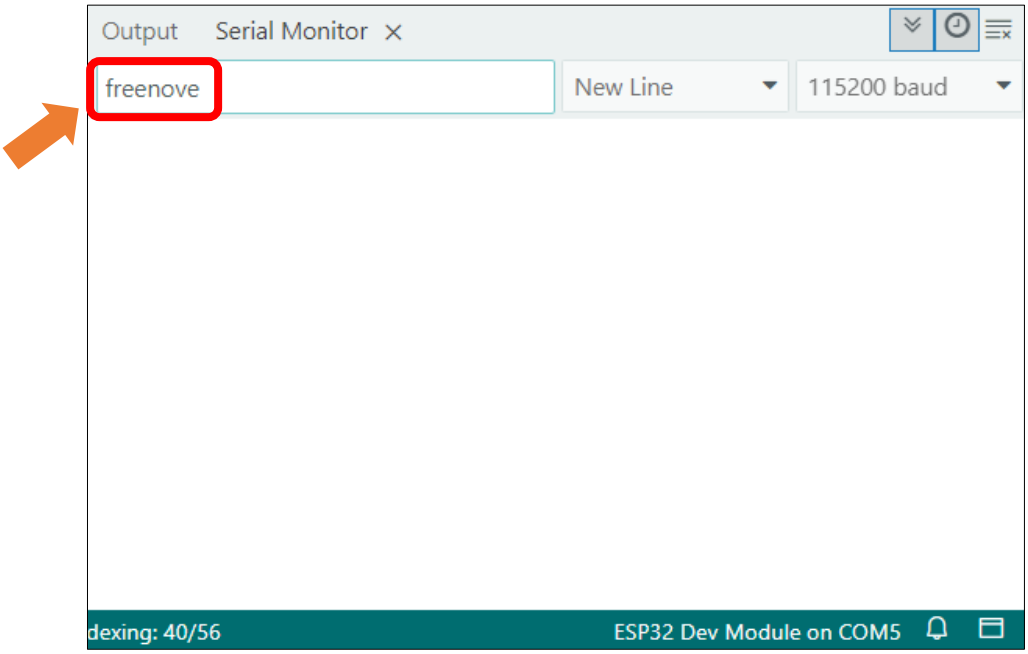
Select UTF-8 String, and click “Save”.



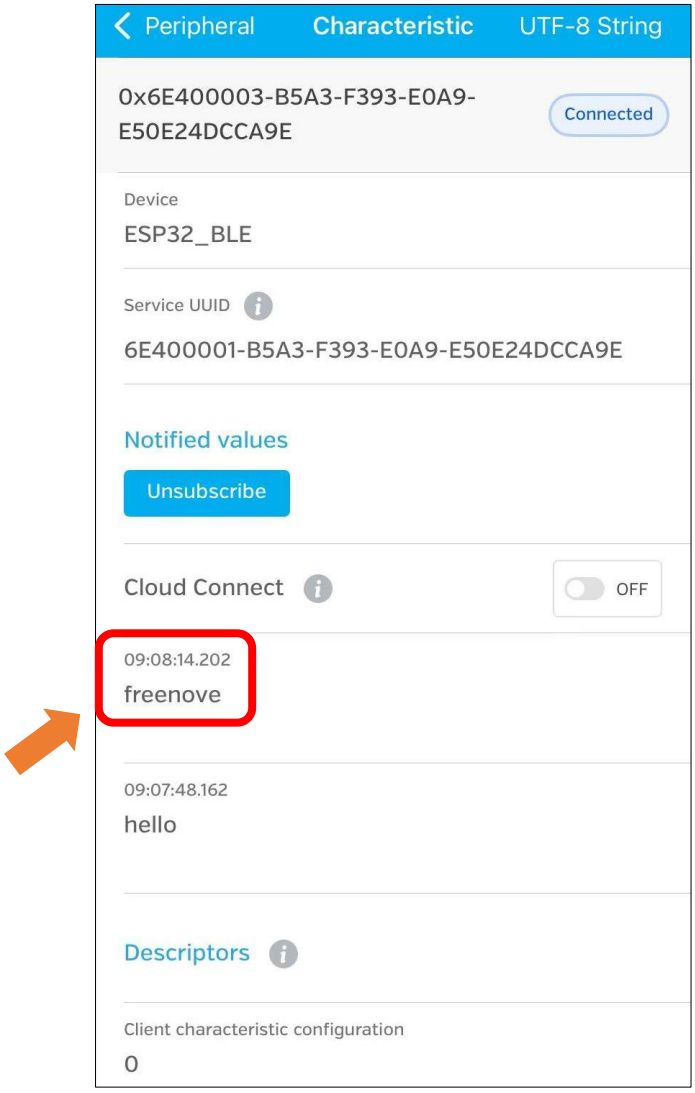
Click Subscribe



Send data on the serial monitor.

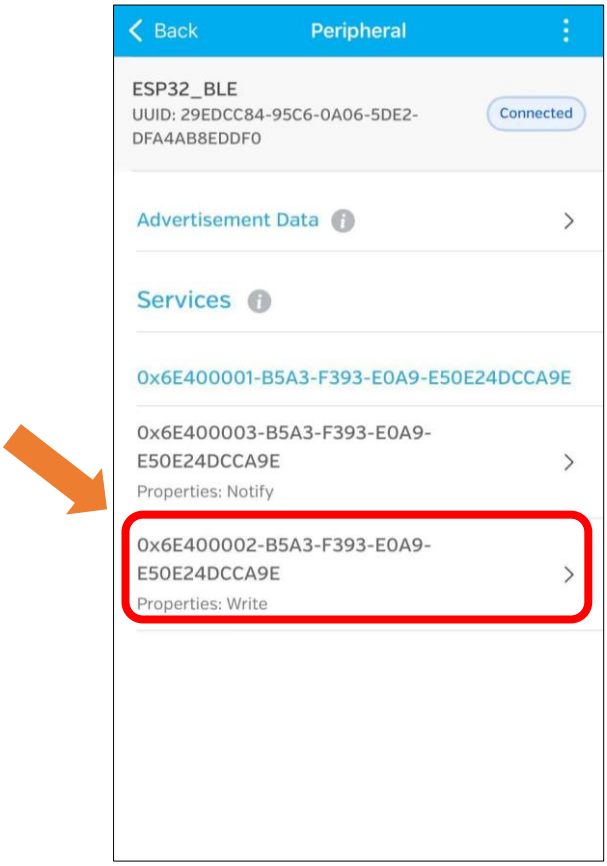


Data will be received on the LightBlue app.

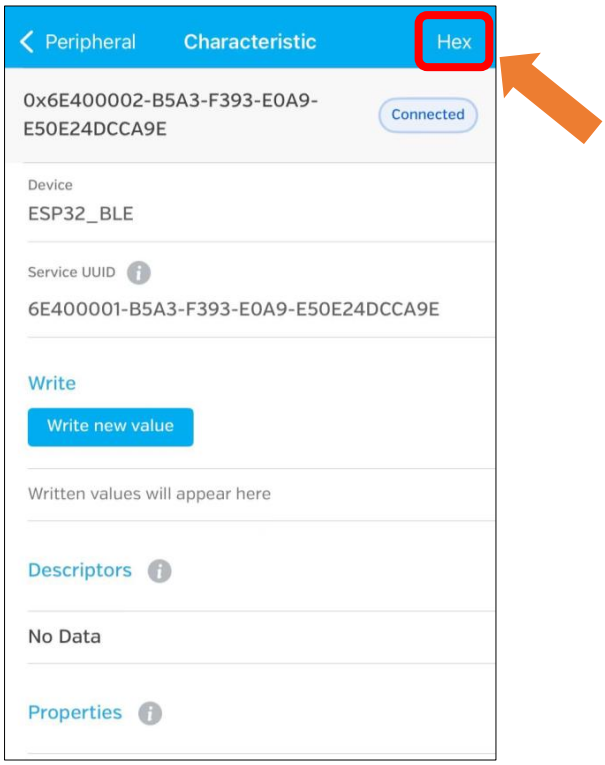


Sending Data

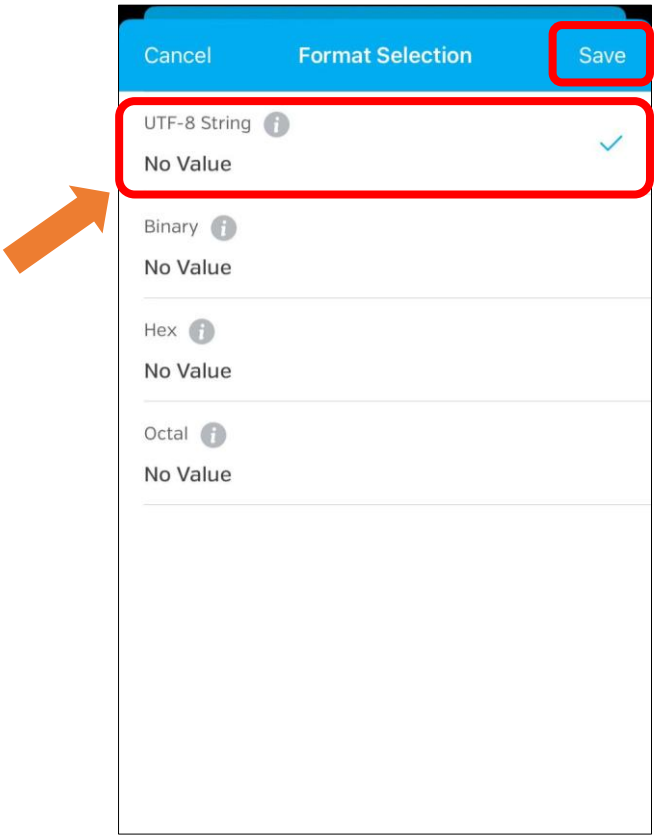
Click Write



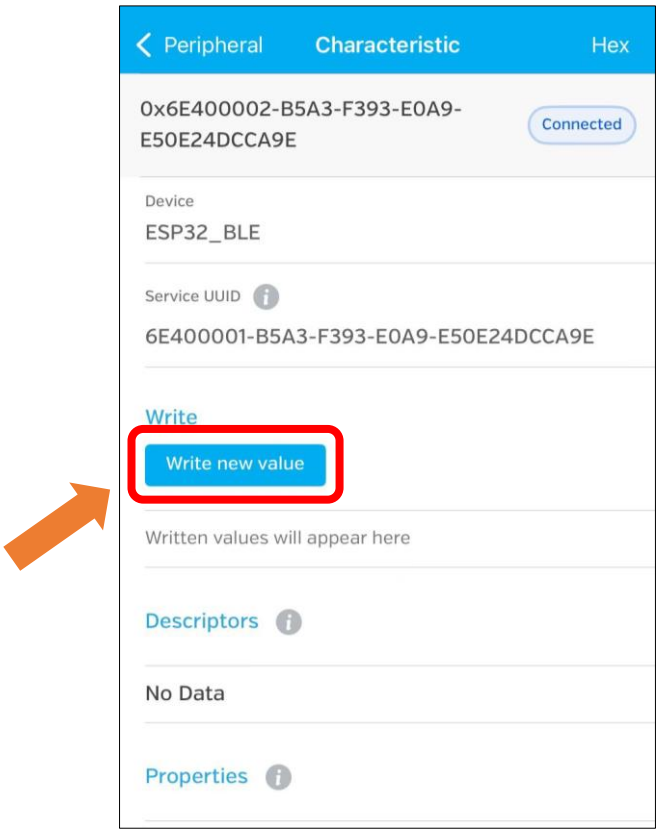
Click the top right corner to change the string type.



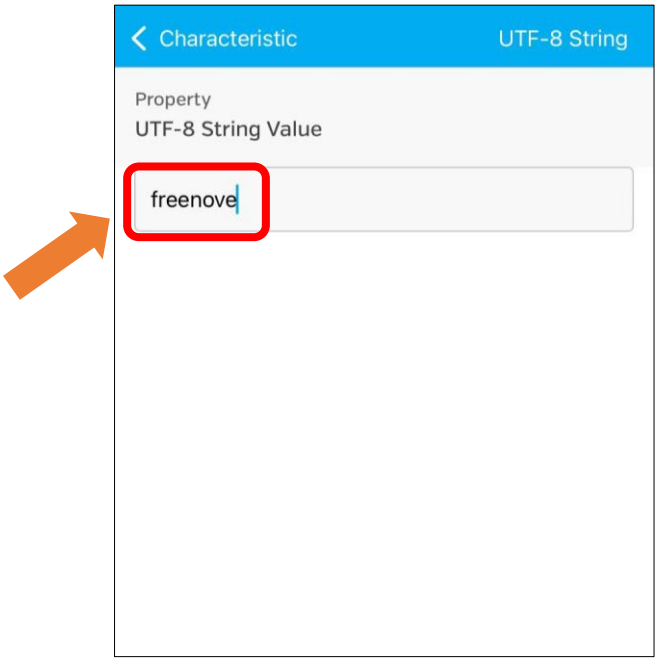
Select UTF-8 String and click “Save”.



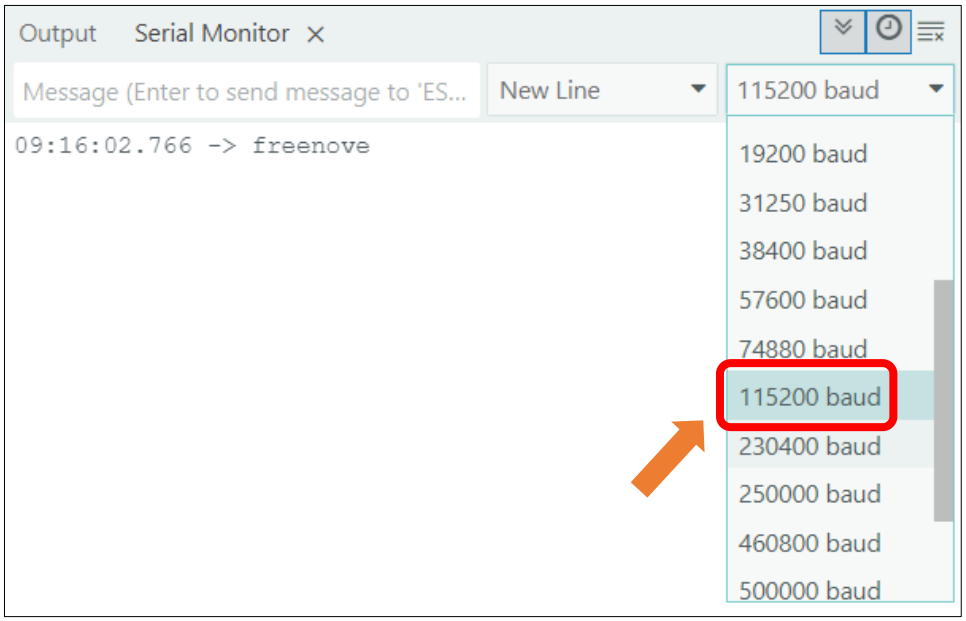
Click “Write new value”



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

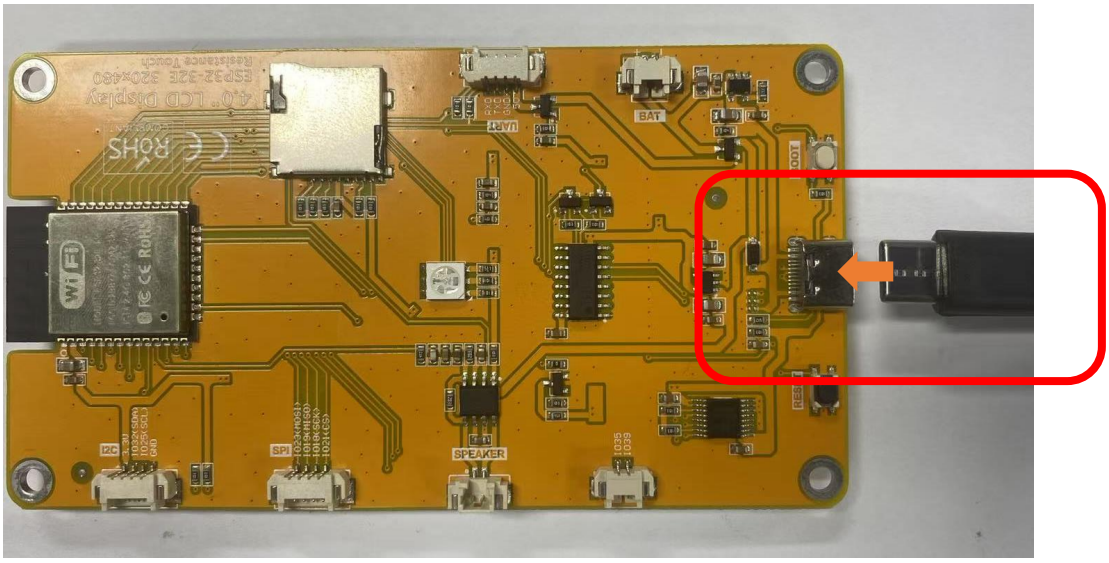
In this section we will control the RGB LED via BLE.

Component List

Freenove ESP32 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_Display to test. Open"**Sketch_08.2_BLE_RGB**" folder under **"Freenove_ESP32_Display\Sketch"** and double-click "**Sketch_08.2_BLE_RGB.ino**".

Sketch_08.2_BLE_RGB

The following is the program code:

```
1  /*
2  * @ File:    Sketch_09.2_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"

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 RED_PIN 22
#define GREEN_PIN 16
#define BLUE_PIN 17

void rgbInit(void) {
   pinMode(RED_PIN, OUTPUT);
   pinMode(GREEN_PIN, OUTPUT);
   pinMode(BLUE_PIN, OUTPUT);
}

void setRGB(bool redLevel, bool greenLevel, bool blueLevel) {
   digitalWrite(RED_PIN, !redLevel);
   digitalWrite(GREEN_PIN, !greenLevel);
   digitalWrite(BLUE_PIN, !blueLevel);
}

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

```
47
48  class MyCallbacks : public BLECharacteristicCallbacks {
49      void onWrite(BLECharacteristic *pCharacteristic) {
50        String rxValue = pCharacteristic->getValue();
51        if (rxValue.length() > 0) {
52          rxload = "";
53          for (int i = 0; i < rxValue.length(); i++) {
54            rxload += (char)rxValue[i];
55          }
56        }
57      }
58  };
59
60  void setupBLE(String BLEName) {
61      const char *ble_name = BLEName.c_str();
62      BLEDevice::init(ble_name);
63      BLEServer *pServer = BLEDevice::createServer();
64      pServer->setCallbacks(new MyServerCallbacks());
65      BLEService *pService = pServer->createService(SERVICE_UUID);
66      pCharacteristic = pService->createCharacteristic(CHARACTERISTIC_UUID_TX,
67  BLECharacteristic::PROPERTY_NOTIFY);
68      pCharacteristic->addDescriptor(new BLE2902());
69      BLECharacteristic *pCharacteristic = pService->createCharacteristic(CHARACTERISTIC_UUID_RX,
70  BLECharacteristic::PROPERTY_WRITE);
71      pCharacteristic->setCallbacks(new MyCallbacks());
72      pService->start();
73      pServer->getAdvertising()->start();
74      Serial.println("Waiting a client connection to notify...");
75  }
76
77  void setup() {
78      Serial.begin(115200);
79      rgbInit();
80      setRGB(0, 0, 0);
81      setupBLE("ESP32_BLE");
82  }
83
84  void loop() {
85      long now = millis();
86      if (now - lastMsg > 100) {
87        if (deviceConnected && rxload.length() > 0) {
88          Serial.println(rxload);
89          if (strncmp(rxload.c_str(), "red_on", 6) == 0) {
90            setRGB(1, 0, 0);
```

```
91          } else if (strncmp(rxload.c_str(), "red_off", 7) == 0) {
92            setRGB(0, 0, 0);
93          } else if (strncmp(rxload.c_str(), "green_on", 8) == 0) {
94            setRGB(0, 1, 0);
95          } else if (strncmp(rxload.c_str(), "green_off", 9) == 0) {
96            setRGB(0, 0, 0);
97          } else if (strncmp(rxload.c_str(), "blue_on", 7) == 0) {
98            setRGB(0, 0, 1);
99          } else if (strncmp(rxload.c_str(), "blue_off", 8) == 0) {
100           setRGB(0, 0, 0);
101         }
102         rxload = "";
103       }
104       if (Serial.available() > 0) {
105         String str = Serial.readString();
106         const char *newValue = str.c_str();
107         pCharacteristic->setValue(newValue);
108         pCharacteristic->notify();
109       }
110       lastMsg = now;
111     }
112 }
```

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"
```

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"
```

Define pins for the RGB LED.

```
23 #define RED_PIN 22
24 #define GREEN_PIN 16
25 #define BLUE_PIN 17
```

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) {
```

```
        deviceConnected = false;
      }
};
```

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

```
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];
          }
        }
      }
};
```

Set the baud rate to 115200

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

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

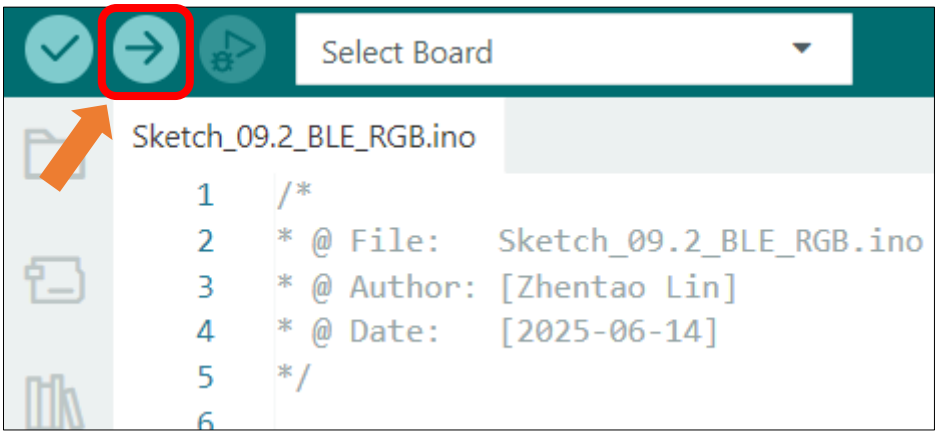
```
rgbInit();
setRGB(0, 0, 0);
setupBLE("ESP32_BLE");
```

The Loop function checks the sending command every 100 milliseconds.

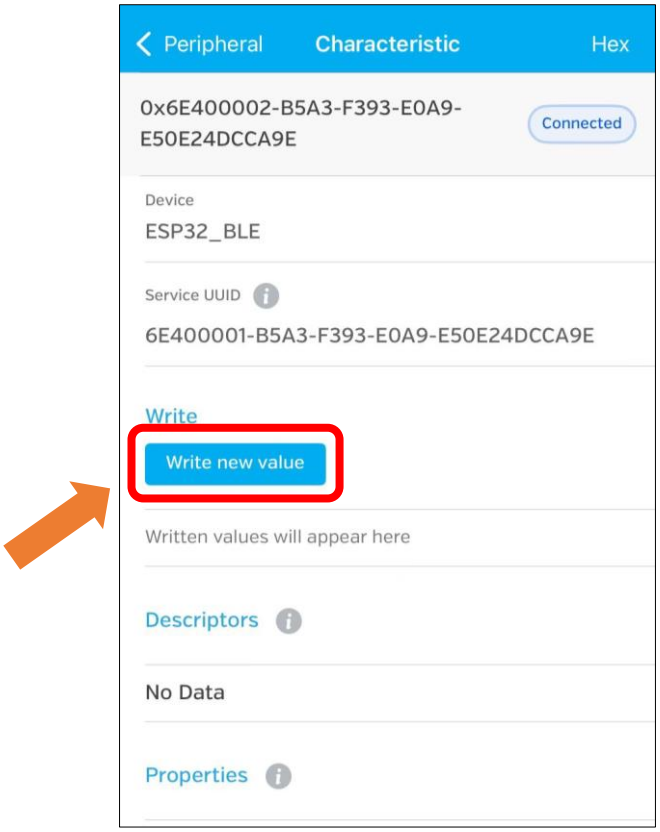
```
void loop() {
   long now = millis();
   if (now - lastMsg > 100) {
     if (deviceConnected && rxload.length() > 0) {
       Serial.println(rxload);
       if (strncmp(rxload.c_str(), "red_on", 6) == 0) {
         setRGB(1, 0, 0);
       } else if (strncmp(rxload.c_str(), "red_off", 7) == 0) {
         setRGB(0, 0, 0);
       } else if (strncmp(rxload.c_str(), "green_on", 8) == 0) {
         setRGB(0, 1, 0);
       } else if (strncmp(rxload.c_str(), "green_off", 9) == 0) {
         setRGB(0, 0, 0);
       } else if (strncmp(rxload.c_str(), "blue_on", 7) == 0) {
         setRGB(0, 0, 1);
       } else if (strncmp(rxload.c_str(), "blue_off", 8) == 0) {
         setRGB(0, 0, 0);
       }
       rxload = "";
     }
     if (Serial.available() > 0) {
       String str = Serial.readString();
```

```
106             const char *newValue = str.c_str();
107             pCharacteristic->setValue(newValue);
108             pCharacteristic->notify();
109         }
110         lastMsg = now;
111     }
112 }
```

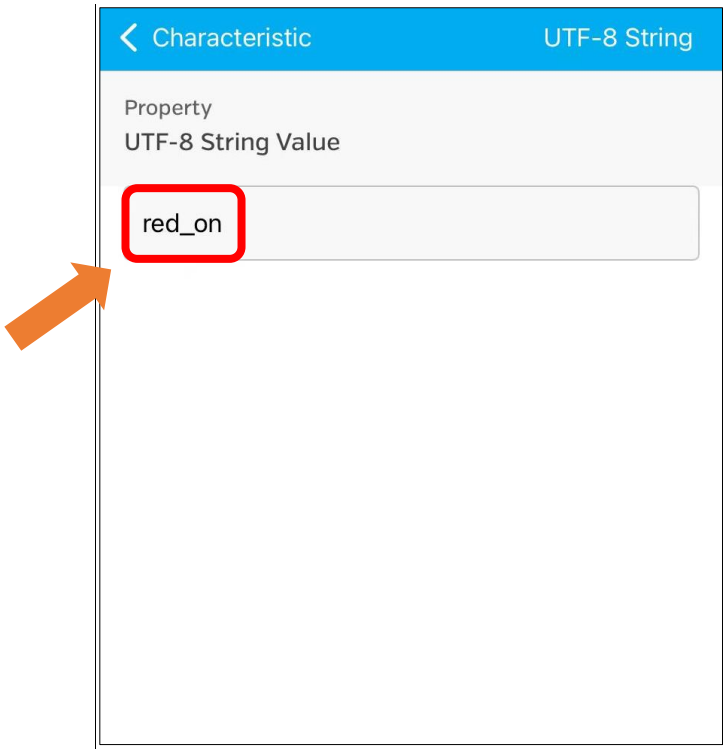
Click "**Upload**" to upload the code to Freenove ESP32 Display.



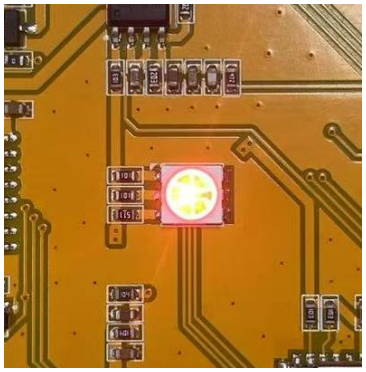
Click "Write new value"



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



The RGB LED on the Freenove ESP32 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 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

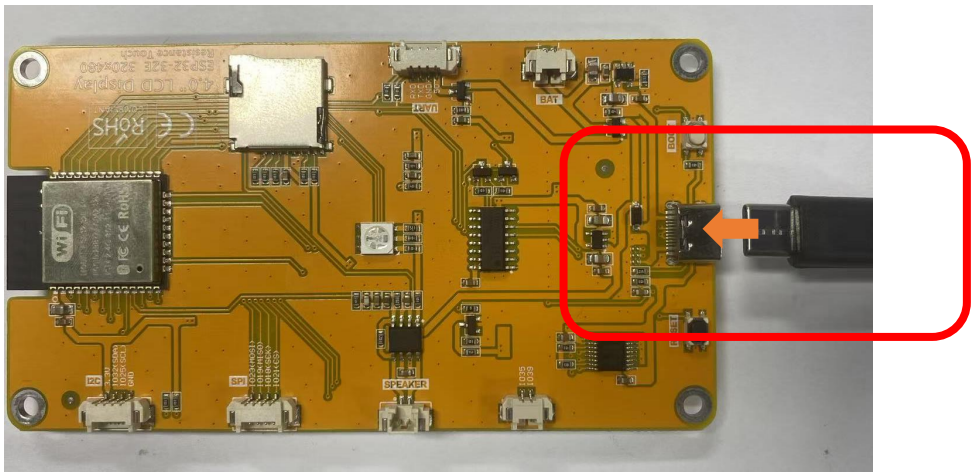
The Web (World Wide Web) is an internet-based information system that integrates global resources through hypertext links. In the embedded field, Web technology refers to devices equipped with lightweight built-in web servers, allowing users to remotely access the device interface via browsers (such 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 Display to the computer with USB cable.



Sketch

Next, we download the code to Freenove_ESP32_Display to test. Open "**Sketch_09.1_WiFi_Web_Servers_LED**" folder under **"Freenove_ESP32_Display\Sketch"** 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 /*
```

```
* @ File:   Sketch_12.1_WiFi_Web_Servers_LED.ino
* @ Author: [Zhentao Lin]
* @ Date:   [2025-06-14]
*/

#include <WiFi.h>

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

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

// HTTP request buffer
String request;

// LED pin definitions
const int redLedPin = 22;
const int greenLedPin = 16;
const int blueLedPin = 17;

// LED state tracking
String greenLedState = "OFF";
String blueLedState = "OFF";
String redLedState = "OFF";

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

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

   // Initialize LED pins
   pinMode(redLedPin, OUTPUT);
   pinMode(greenLedPin, OUTPUT);
   pinMode(blueLedPin, OUTPUT);

   digitalWrite(redLedPin, HIGH);
   digitalWrite(greenLedPin, HIGH);
   digitalWrite(blueLedPin, HIGH);

```

```
46    // Connect to WiFi
47    Serial.print("Connecting to ");
48    Serial.println(ssid);
49    WiFi.begin(ssid, password);
50    while (WiFi.status() != WL_CONNECTED) {
51      delay(500);
52      Serial.print(".");
53    }
54
55    Serial.println("");
56    Serial.println("WiFi connected.");
57    Serial.println("IP address: ");
58    Serial.println(WiFi.localIP());
59
60    server.begin();
61  }
62
63  void loop() {
64    WiFiClient client = server.available();
65
66    if (client) {
67      currentTime = millis();
68      previousTime = currentTime;
69      Serial.println("New Client.");
70      String currentLine = "";
71
72      while (client.connected() && currentTime - previousTime <= timeoutTime) {
73        currentTime = millis();
74        if (client.available()) {
75          char c = client.read();
76          Serial.write(c);
77          request += c;
78
79          if (c == '\n') {
80            if (currentLine.length() == 0) {
81              // HTTP response headers
82              // Handle incoming commands first. These are AJAX requests.
83              if (request.indexOf("GET /22/ON") >= 0) {
84                Serial.println("Red LED ON");
85                redLedState = "ON";
86                digitalWrite(redLedPin, LOW);
87              } else if (request.indexOf("GET /22/OFF") >= 0) {
88                Serial.println("Red LED OFF");
89                redLedState = "OFF";
```

```
                digitalWrite(redLedPin, HIGH);
              } else if (request.indexOf("GET /16/ON") >= 0) {
                Serial.println("Green LED ON");
                greenLedState = "ON";
                digitalWrite(greenLedPin, LOW);
              } else if (request.indexOf("GET /16/OFF") >= 0) {
                Serial.println("Green LED OFF");
                greenLedState = "OFF";
                digitalWrite(greenLedPin, HIGH);
              } else if (request.indexOf("GET /17/ON") >= 0) {
                Serial.println("Blue LED ON");
                blueLedState = "ON";
                digitalWrite(blueLedPin, LOW);
              } else if (request.indexOf("GET /17/OFF") >= 0) {
                Serial.println("Blue LED OFF");
                blueLedState = "OFF";
                digitalWrite(blueLedPin, HIGH);
              } else {
                // Send the Complete HTML page
                client.println("HTTP/1.1 200 OK");
                client.println("Content-type:text/html");
                client.println("Connection: close");
                client.println();

                // Generate HTML response
                client.println("<!DOCTYPE html><html>");
                client.println("<head><meta charset=\"UTF-8\">");
                client.println("<meta name=\"viewport\" content=\"width=device-width, initial-
scale=1.0\">");
                client.println("<title>ESP32 Web Server LED</title>");
                client.println("<style>");
                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;}}");
                client.println("</style></head>");
                client.println("<body>");
                client.println("<h1>ESP32 Web Server LED</h1>");
                client.println("<div class=\"container\">");
                client.println("");
                // 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>");
                client.println("");
                // 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>");
                client.println("");
                // 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>");
                client.println("");
                client.println("</div>");
```

```
178                 client.println("<div class=\"device-info\">Device IP: " +
179 WiFi.localIP().toString() + " | ESP32 Web Server LED</div>");
180                 client.println("<script>");
181                 client.println("function toggleLED(pin, element) {");
182                 client.println("  const statusElement = document.getElementById('status-' +
183 pin);");
184                 client.println("  const currentState = statusElement.innerText;");
185                 client.println("  const newState = currentState === 'ON' ? 'OFF' : 'ON';");
186                 client.println("  fetch('/' + pin + '/' + newState);");
187                 client.println("  statusElement.innerText = newState;");
188                 client.println("  if (newState === 'ON') { element.classList.remove('off-
189 state'); } else { element.classList.add('off-state'); }");
190                 client.println("}");
191                 client.println("</script>");
192                 client.println("</body></html>");
193               }
194               break;
195             } else {
196               currentLine = "";
197             }
198           } else if (c != '\r') {
199             currentLine += c;
200           }
201         }
202       }
203
204       request = "";
205       client.stop();
206       Serial.println("Client disconnected.");
207       Serial.println("");
208     }
209 }
```

Code Explanation

Include necessary header files.

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

Configure WiFi SSID and password.

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

Define pins for the RGB LED.

```
19 // LED pin definitions
20 const int redLedPin = 22;
21 const int greenLedPin = 16;
22 const int blueLedPin = 17;
```

Initialize the configuration related to RGB LED.

```
37  // Initialize LED pins
38  pinMode(redLedPin, OUTPUT);
39  pinMode(greenLedPin, OUTPUT);
40  pinMode(blueLedPin, OUTPUT);
41
42  digitalWrite(redLedPin, HIGH);
43  digitalWrite(greenLedPin, HIGH);
44  digitalWrite(blueLedPin, HIGH);
```

Connect to WIFI.

```
46  // Connect to WiFi
47  Serial.print("Connecting to ");
48  Serial.println(ssid);
49  WiFi.begin(ssid, password);
50  while (WiFi.status() != WL_CONNECTED) {
51     delay(500);
52     Serial.print(".");
53  }
54
55  Serial.println("");
56  Serial.println("WiFi connected.");
57  Serial.println("IP address: ");
58  Serial.println(WiFi.localIP());
59
60  server.begin();
```

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

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

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

```
87  // Handle incoming commands
88  if (request.indexOf("GET /22/on") >= 0) {
89    Serial.println("Red LED ON");
90    redLedState = "on";
91    digitalWrite(redLedPin, LOW);
92  } else if (request.indexOf("GET /22/off") >= 0) {
93    Serial.println("Red LED OFF");
94    redLedState = "off";
95    digitalWrite(redLedPin, HIGH);
96  } else if (request.indexOf("GET /16/on") >= 0) {
97    Serial.println("Green LED ON");
98    greenLedState = "on";
99    digitalWrite(greenLedPin, LOW);
100 } else if (request.indexOf("GET /16/off") >= 0) {
101   Serial.println("Green LED OFF");
102   greenLedState = "off";
103   digitalWrite(greenLedPin, HIGH);
```

```
} else if (request.indexOf("GET /17/on") >= 0) {
  Serial.println("Blue LED ON");
  blueLedState = "on";
  digitalWrite(blueLedPin, LOW);
} else if (request.indexOf("GET /17/off") >= 0) {
  Serial.println("Blue LED OFF");
  blueLedState = "off";
  digitalWrite(blueLedPin, HIGH);
}
```

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>");
```

```
159  client.println("</div>");
160
161  // Green LED control
162  client.println("<div class=\"card " + String(greenLedState == "OFF" ? "off-state" : "") + "\"
163  id=\"green-card\" onclick=\"toggleLED('16', this)\">");
164  client.println("<h2>Green LED</h2>");
165  client.println("<p class=\"status\" id=\"status-16\">" + greenLedState + "</p>");
166  client.println("</div>");
167
168  // Blue LED control
169  client.println("<div class=\"card " + String(blueLedState == "OFF" ? "off-state" : "") + "\"
170  id=\"blue-card\" onclick=\"toggleLED('17', this)\">");
171  client.println("<h2>Blue LED</h2>");
172  client.println("<p class=\"status\" id=\"status-17\">" + blueLedState + "</p>");
173  client.println("</div>");
```

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

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

Input correct WiFi(2.4GHz) SSID and password.

```
9   // WiFi credentials
10  const char* ssid     = "***************";
11  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.

Reference

`server.begin()` This function starts the server to monitor the port.
`server.available()` This function is used to detect and return the connected client object.

Chapter 10 TFT Display

Project 10.1 TFT_Rainbow

Freenove ESP32 Display is available in five different models, each featuring a unique TFT display. This guide applies to all of them. For detailed specifications of each model, please click here.

Component List

Freenove ESP32 Display x 1 	USB cable x1 

Component Knowledge

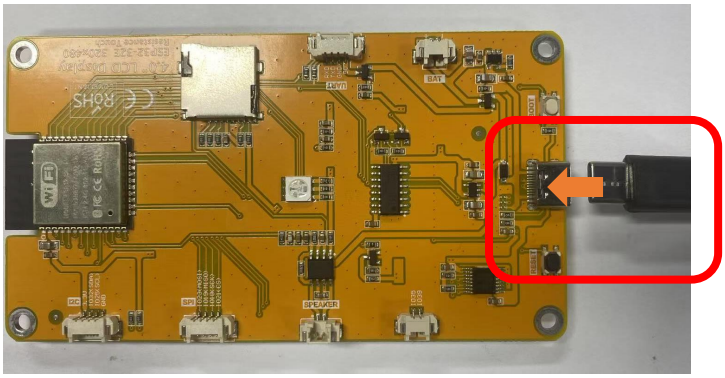
TFT Display

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.

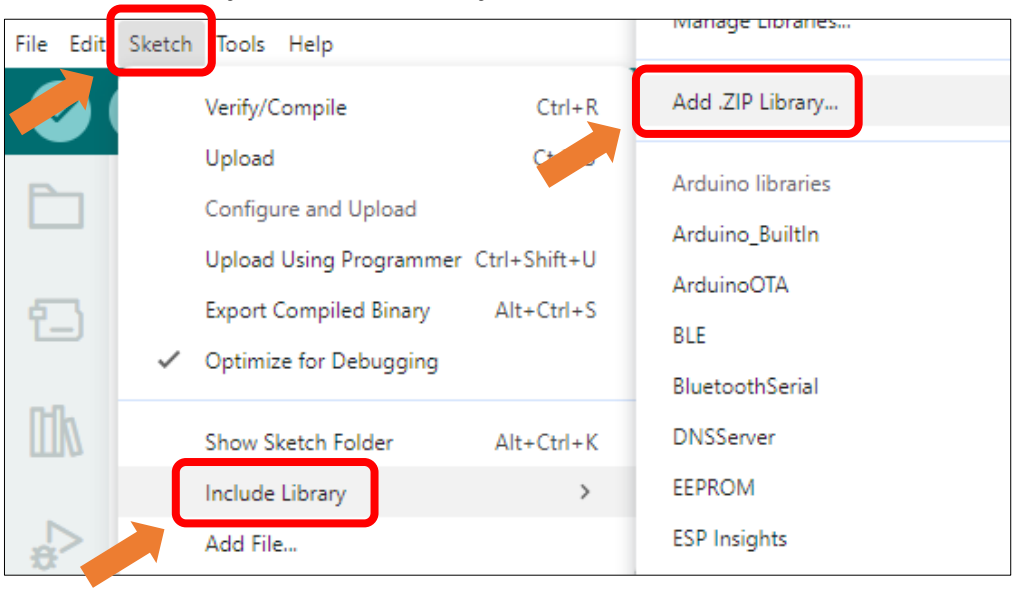


Sketch

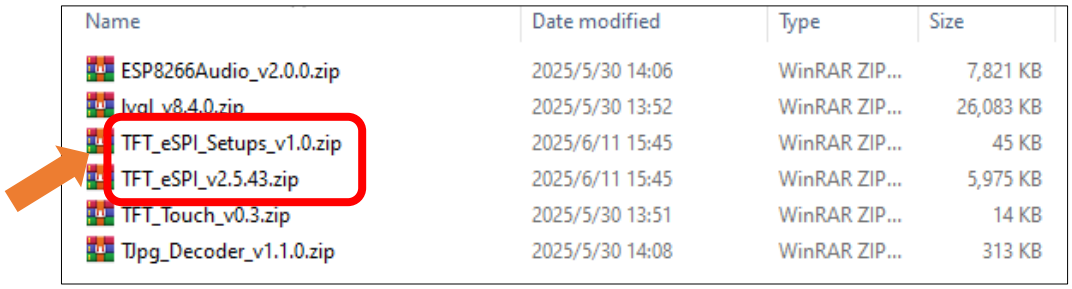
Open "**Sketch_10.1_TFT_Rainbow.ino**" folder under **"Freenove_ESP32_Display\Sketch"** and double-click "**Sketch_10.1_TFT_Rainbow.ino**".

Install Libraries

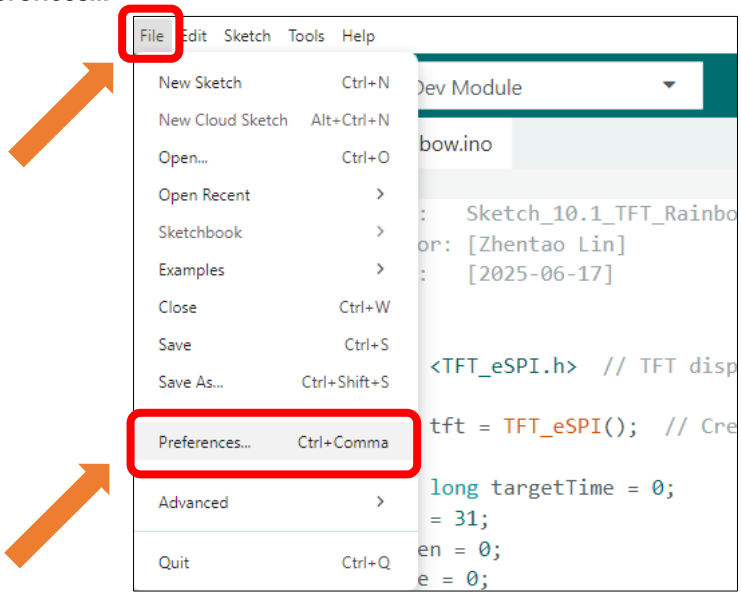
Click **Sketch** -> **Include Library** -> **Add .ZIP Library...**



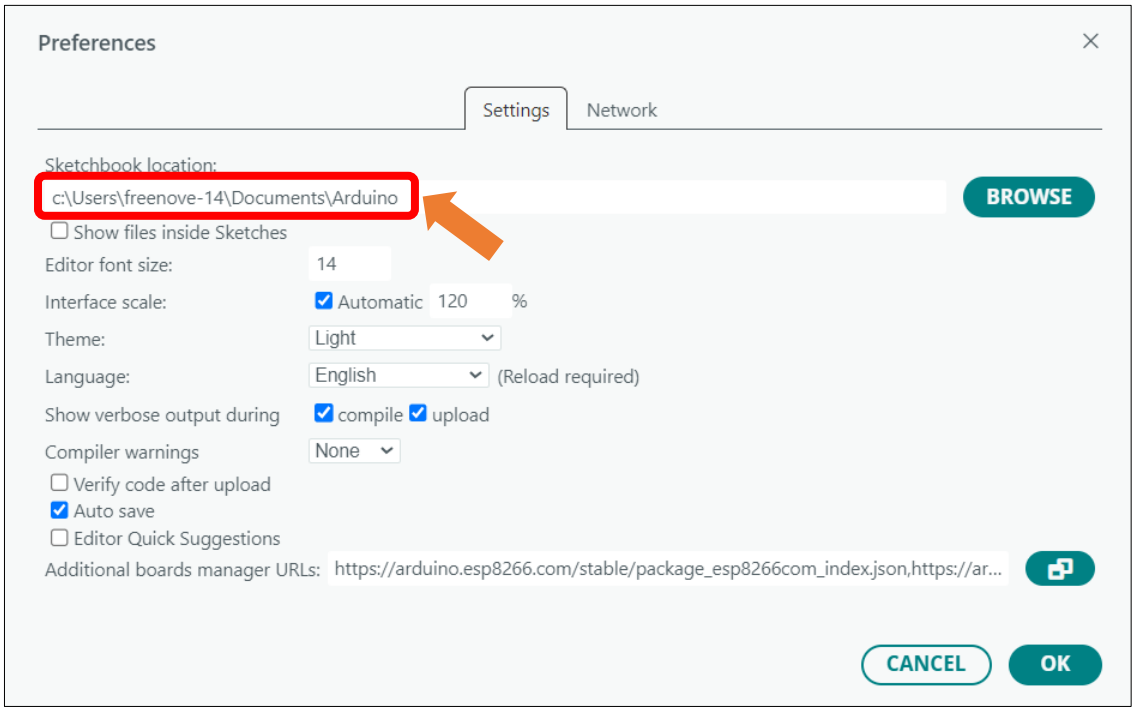
Install **TFT_eSPI_v2.5.43.zip** and **TFT_eSPI_Setups_v1.0.zip**



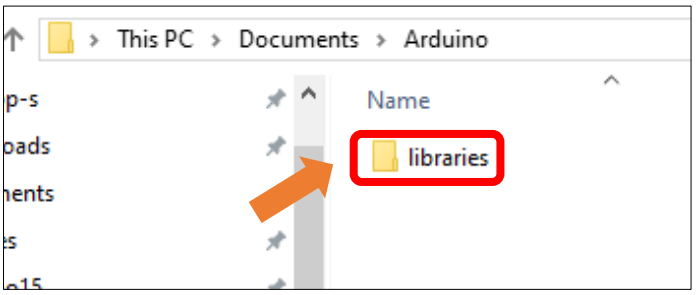
Click **File** -> **Preferences...**



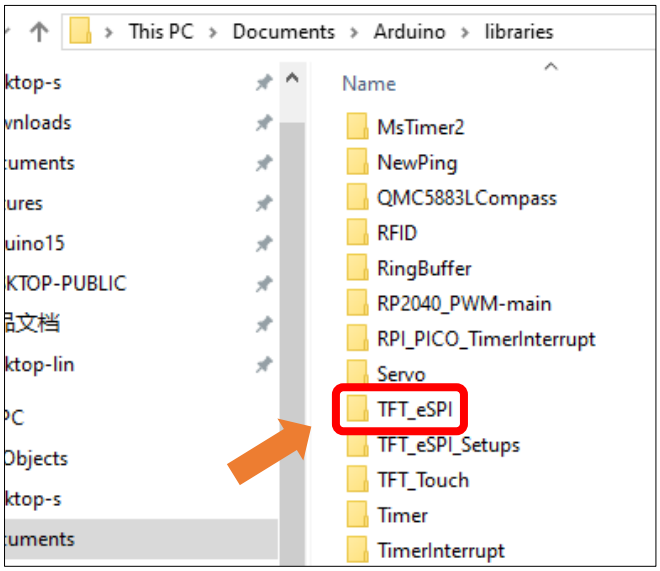
Linux and Mac users can use the **cd** command on Terminal to enter the sketchbook location, and Windows users can copy and paste it in the file explorer.



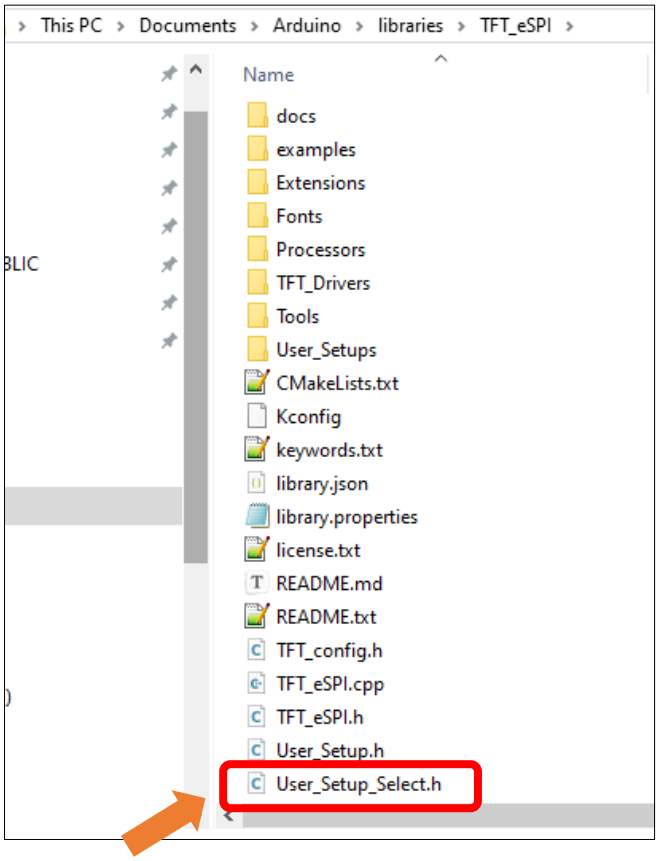
Double click **libraries**



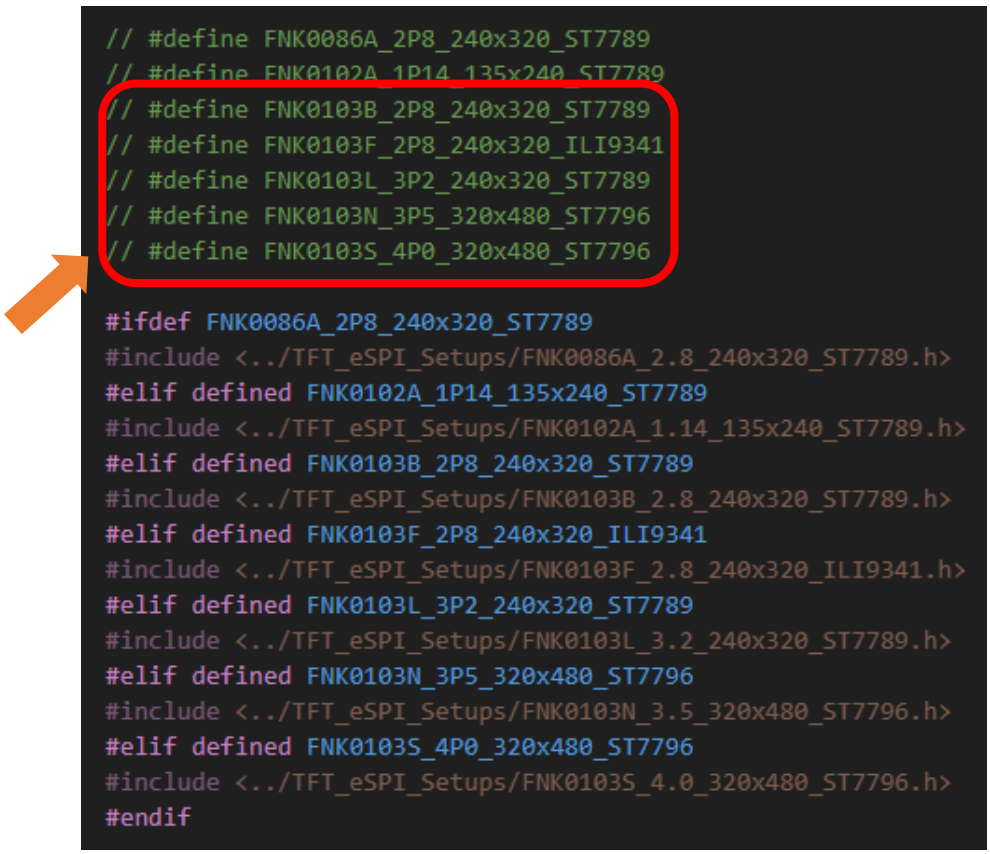
Double click **TFT_eSPI**



Open the **User_Setup_Select.h** file

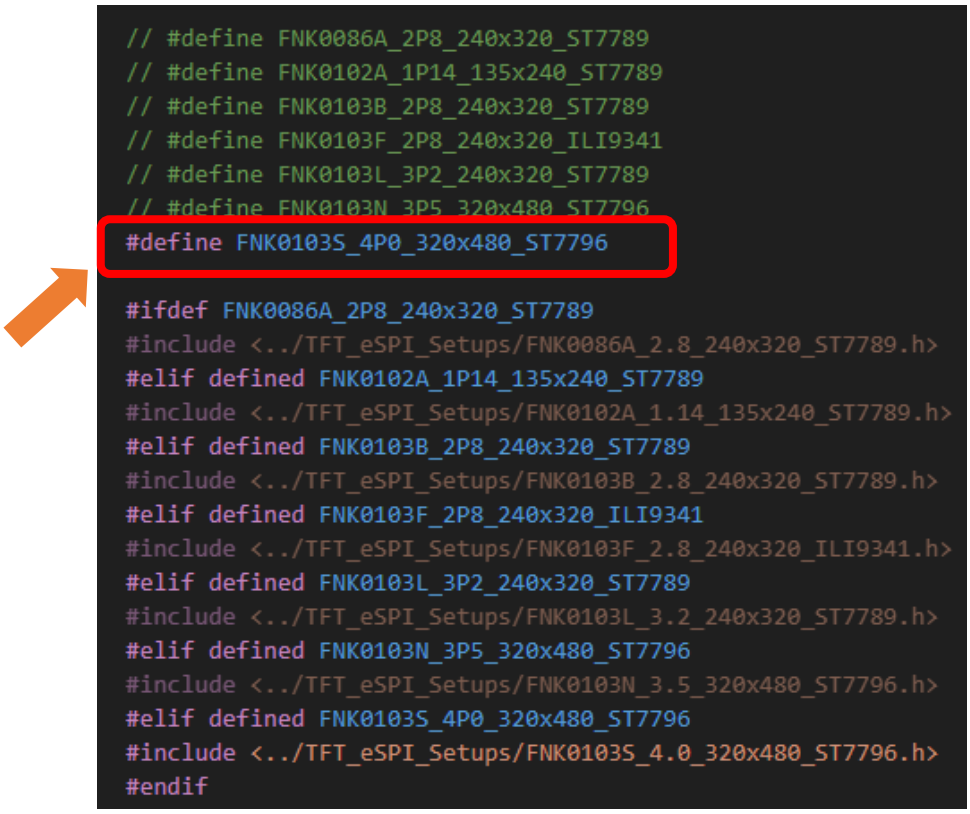


Remove the "//" comment markers from the macro definition line **corresponding to your Freenove ESP32 Display model**.



```
// #define FNK0086A_2P8_240x320_ST7789
// #define FNK0102A_1P14_135x240_ST7789
// #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

#ifdef FNK0086A_2P8_240x320_ST7789
#include <../TFT_eSPI_Setups/FNK0086A_2.8_240x320_ST7789.h>
#elif defined FNK0102A_1P14_135x240_ST7789
#include <../TFT_eSPI_Setups/FNK0102A_1.14_135x240_ST7789.h>
#elif defined FNK0103B_2P8_240x320_ST7789
#include <../TFT_eSPI_Setups/FNK0103B_2.8_240x320_ST7789.h>
#elif defined FNK0103F_2P8_240x320_ILI9341
#include <../TFT_eSPI_Setups/FNK0103F_2.8_240x320_ILI9341.h>
#elif defined FNK0103L_3P2_240x320_ST7789
#include <../TFT_eSPI_Setups/FNK0103L_3.2_240x320_ST7789.h>
#elif defined FNK0103N_3P5_320x480_ST7796
#include <../TFT_eSPI_Setups/FNK0103N_3.5_320x480_ST7796.h>
#elif defined FNK0103S_4P0_320x480_ST7796
#include <../TFT_eSPI_Setups/FNK0103S_4.0_320x480_ST7796.h>
#endif
```

Here we take the 4-in display as an example, modifying it as shown below:



```
// #define FNK0086A_2P8_240x320_ST7789
// #define FNK0102A_1P14_135x240_ST7789
// #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

#ifdef FNK0086A_2P8_240x320_ST7789
#include <../TFT_eSPI_Setups/FNK0086A_2.8_240x320_ST7789.h>
#elif defined FNK0102A_1P14_135x240_ST7789
#include <../TFT_eSPI_Setups/FNK0102A_1.14_135x240_ST7789.h>
#elif defined FNK0103B_2P8_240x320_ST7789
#include <../TFT_eSPI_Setups/FNK0103B_2.8_240x320_ST7789.h>
#elif defined FNK0103F_2P8_240x320_ILI9341
#include <../TFT_eSPI_Setups/FNK0103F_2.8_240x320_ILI9341.h>
#elif defined FNK0103L_3P2_240x320_ST7789
#include <../TFT_eSPI_Setups/FNK0103L_3.2_240x320_ST7789.h>
#elif defined FNK0103N_3P5_320x480_ST7796
#include <../TFT_eSPI_Setups/FNK0103N_3.5_320x480_ST7796.h>
#elif defined FNK0103S_4P0_320x480_ST7796
#include <../TFT_eSPI_Setups/FNK0103S_4.0_320x480_ST7796.h>
#endif
```

Important Note: Only one macro definition should be uncommented.

Sketch_10.1_TFT_Rainbow

The following is the program code:

```
/*
* @ File:   Sketch_13.1_TFT_Rainbow.ino
* @ Author: [Zhentao Lin]
* @ Date:   [2025-06-17]
*/

#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
```

```
20      tft.setRotation(1);           // Set screen rotation (landscape mode)
21      targetTime = millis() + 100;  // Set initial target time for rainbow effect
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);
32  }
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:
```

```
            green -= 2;  // Transition from cyan to blue (decrease green)
            if (green == 255) {
              green = 0;
              state = 4;
            }
            break;
          case 4:
            red++;  // Transition from blue to magenta (increase red)
            if (red == 32) {
              red = 31;  // Cap at max red value
              state = 5;
            }
            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
      }
      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
      float pi = 3.14159;                                    // Value to print
      int precision = 3;                                     // Number of decimal digits
      int xpos = 50;                                         // X position
      int ypos = 110;                                        // Y position
      int font = 2;                                          // Font type
      xpos += tft.drawFloat(pi, precision, xpos, ypos, font);  // Draw float and update x-
position
      tft.drawString(" is pi", xpos, ypos, font);              // Continue drawing string from
updated x position
      delay(6000);  // Wait before next cycle
    }
}
```

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.

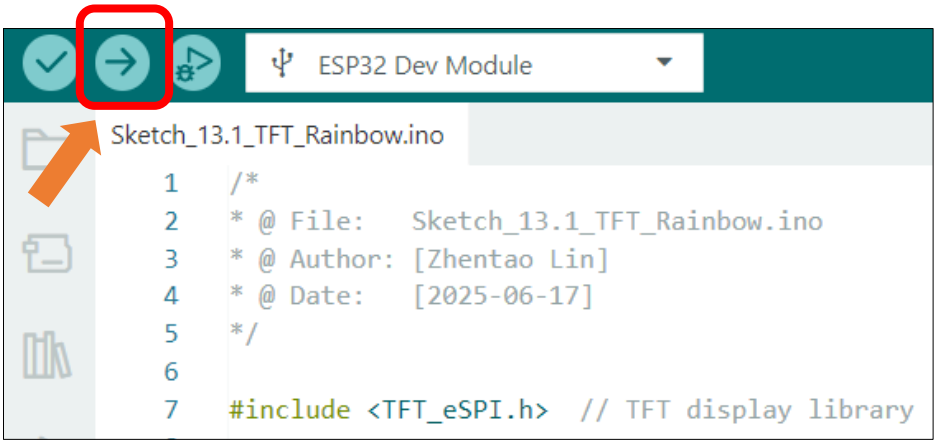
```
38  for (int i = 0; i < 320; i++) {           // Generate horizontal rainbow using vertical lines
39     tft.drawFastVLine(i, 0, tft.height(), colour);  // Draw one vertical line
40     switch (state) {
41       case 0:
42          green += 2;  // Transition from red to yellow (increase green)
43          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;
76        case 5:
77          blue--;  // Transition from magenta to red (decrease blue)
78          if (blue == 255) {
79            blue = 0;
80            state = 0;
81          }
82          break;
83      }
84      colour = red << 11 | green << 5 | blue;  // Combine RGB values into 16-bit color
85  }
```

Text display.

```
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
```

Click “**Upload**” to upload the code to Freenove_ESP32_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 Display x 1 	USB cable x1 

Component Knowledge

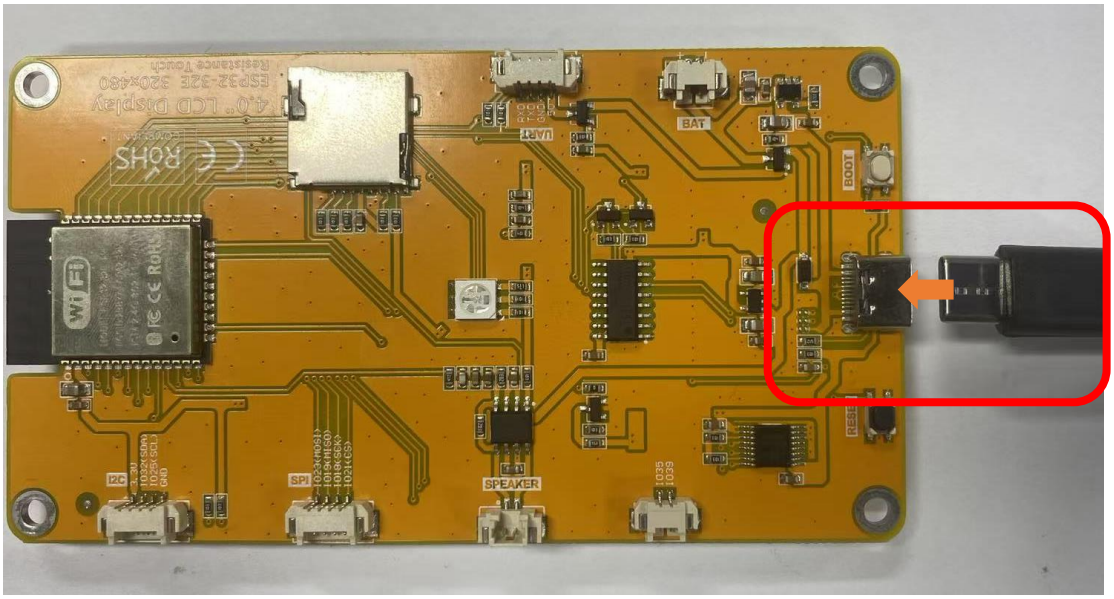
DMA

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.

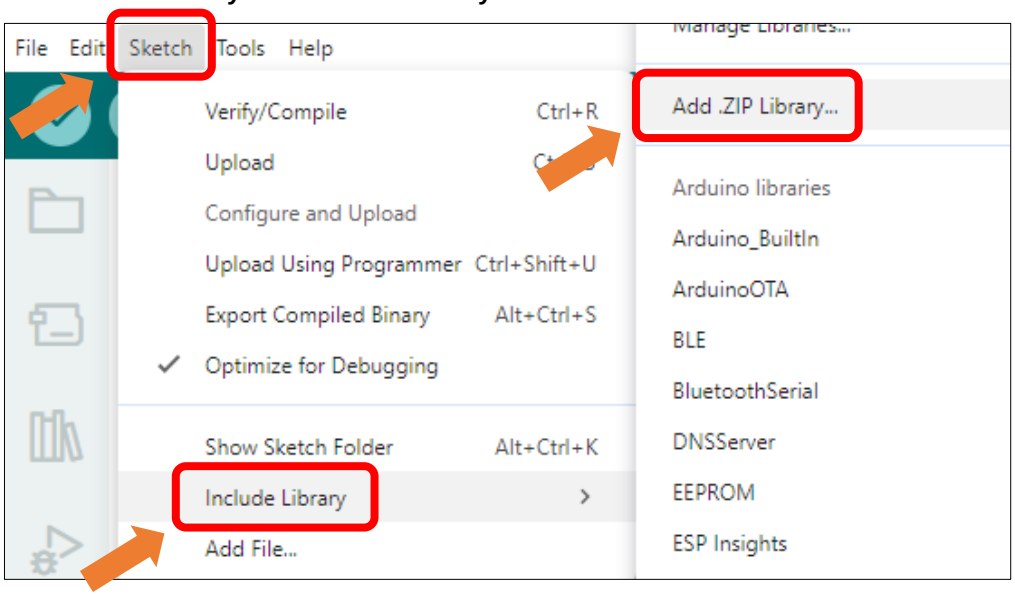
Circuit

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

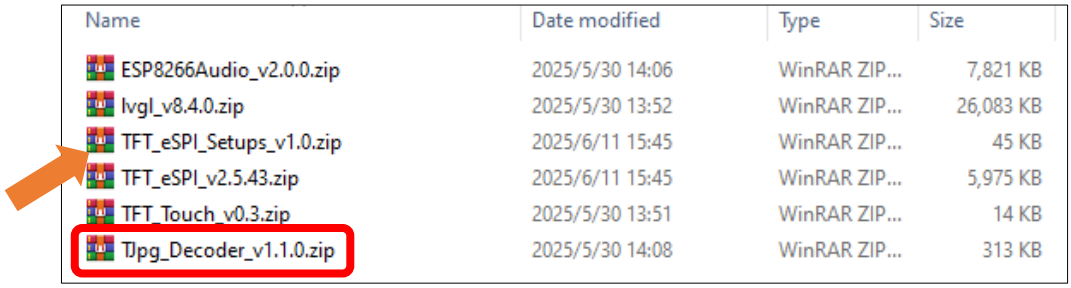


Sketch

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_Display\Sketch"** 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  #define USE_DMA
7  #include "panda.h"          // Include raw JPEG image data array
8  #include <TJpg_Decoder.h>  // Include JPEG decoder library
9
10 #ifdef USE_DMA
11 uint16_t dmaBuffer1[16 * 16];  // DMA buffer for 16x16 pixel block
12 uint16_t dmaBuffer2[16 * 16];  // Second DMA buffer for toggling
13 uint16_t* dmaBufferPtr = dmaBuffer1;
14 bool dmaBufferSel = 0;
```

```
#endif

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

// Callback function to draw decoded JPEG blocks on screen
bool tft_output(int16_t x, int16_t y, uint16_t w, uint16_t h, uint16_t* bitmap) {
    if (y >= tft.height()) return 0;  // Stop decoding if off-screen

#ifdef USE_DMA
    if (dmaBufferSel) dmaBufferPtr = dmaBuffer2;
    else dmaBufferPtr = dmaBuffer1;
    dmaBufferSel = !dmaBufferSel;
    tft.pushImageDMA(x, y, w, h, bitmap, dmaBufferPtr);  // Start DMA transfer
#else
    tft.pushImage(x, y, w, h, bitmap);  // Draw without DMA
#endif

    return 1;  // Continue decoding next block
}

void setup() {
    Serial.begin(115200);
    Serial.println("\n\n Testing TJpg_Decoder library");

    tft.begin();  // Initialize TFT display
    tft.setTextColor(TFT_WHITE, TFT_BLACK);
    tft.fillScreen(TFT_BLACK);  // Clear screen with black background

#ifdef USE_DMA
    tft.initDMA();  // Setup SPI DMA engine
#endif

    TJpgDec.setJpgScale(1);           // Set scale factor (1=full size)
    tft.setSwapBytes(true);           // Match color byte order
    TJpgDec.setCallback(tft_output);  // Register drawing callback
}

void loop() {
    uint16_t w = 0, h = 0;
    TJpgDec.getJpgSize(&w, &h, img_asset, sizeof(img_asset));  // Get image dimensions
    Serial.print("Width = ");
    Serial.print(w);
    Serial.print(", height = ");
    Serial.print(h);
```

```
59
60      uint32_t dt = millis();
61      tft.startWrite();
62      TJpgDec.drawJpg(0, 0, img_asset, sizeof(img_asset));  // Draw the JPEG image
63      tft.endWrite();
64
65      dt = millis() - dt;
66      Serial.print(", dt = ");
67      Serial.print(dt);
68      Serial.println(" ms");
69
70      delay(2000);  // Wait before redraw
71  }
```

Code Explanation

Include necessary header files.

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

Configure DMA buffer

```
7   uint16_t dmaBuffer1[16 * 16];  // DMA buffer for 16x16 pixel block
8   uint16_t dmaBuffer2[16 * 16];  // Second DMA buffer for toggling
9   uint16_t* dmaBufferPtr = dmaBuffer1;
10  bool dmaBufferSel = 0;
```

Create TFT object instance.

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

JPEG decoding callback function

```
15  // Callback function to draw decoded JPEG blocks on screen
16  bool tft_output(int16_t x, int16_t y, uint16_t w, uint16_t h, uint16_t* bitmap) {
17      if (y >= tft.height()) return 0;  // Stop decoding if off-screen
18
19  #ifdef USE_DMA
20      if (dmaBufferSel) dmaBufferPtr = dmaBuffer2;
21      else dmaBufferPtr = dmaBuffer1;
22      dmaBufferSel = !dmaBufferSel;
23      tft.pushImageDMA(x, y, w, h, bitmap, dmaBufferPtr);  // Start DMA transfer
24  #else
25      tft.pushImage(x, y, w, h, bitmap);  // Draw without DMA
26  #endif
27
28      return 1;  // Continue decoding next block
29  }
```

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();
```

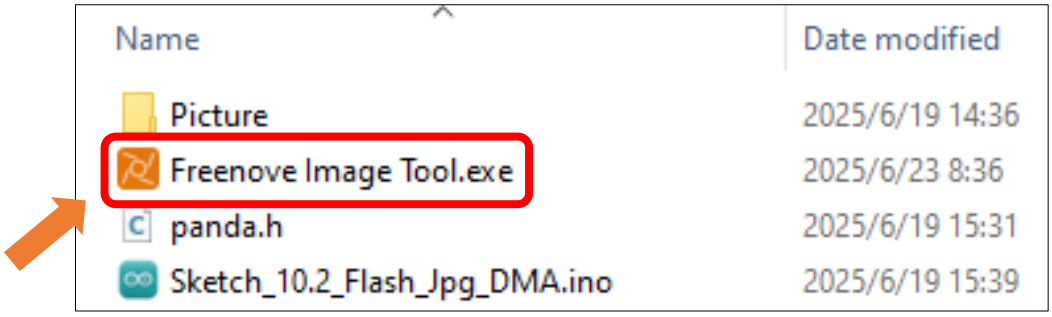
Click "**Upload**" to upload the code to Freenove ESP32 Display

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

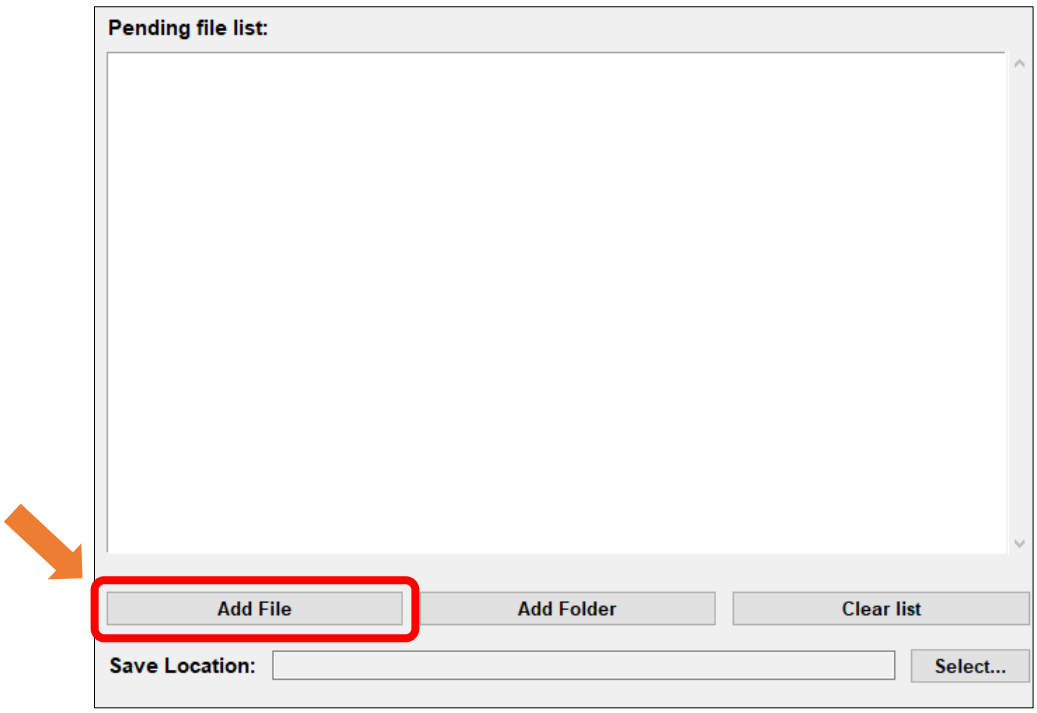
Custom image display

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

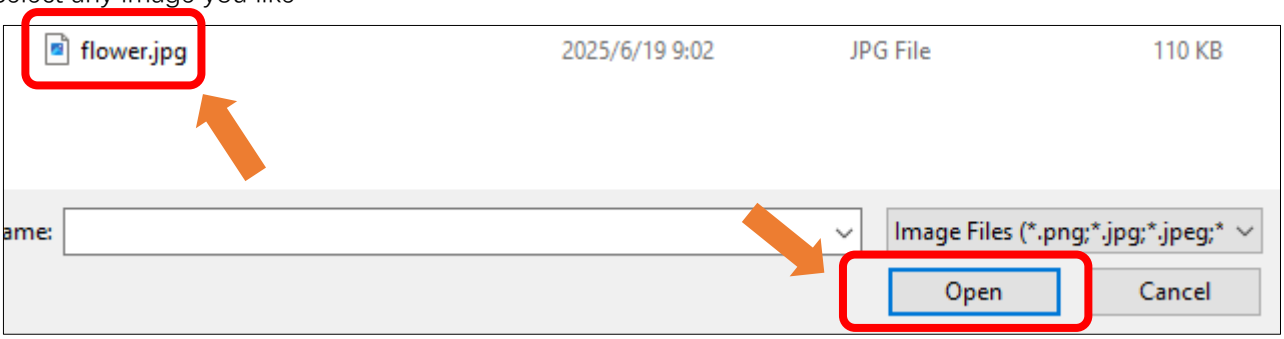
First, open **Freenove_ESP32_Display\Sketch\Sketch_10.2_Flash_Jpg_DMA\Freenove Image Tool.exe**



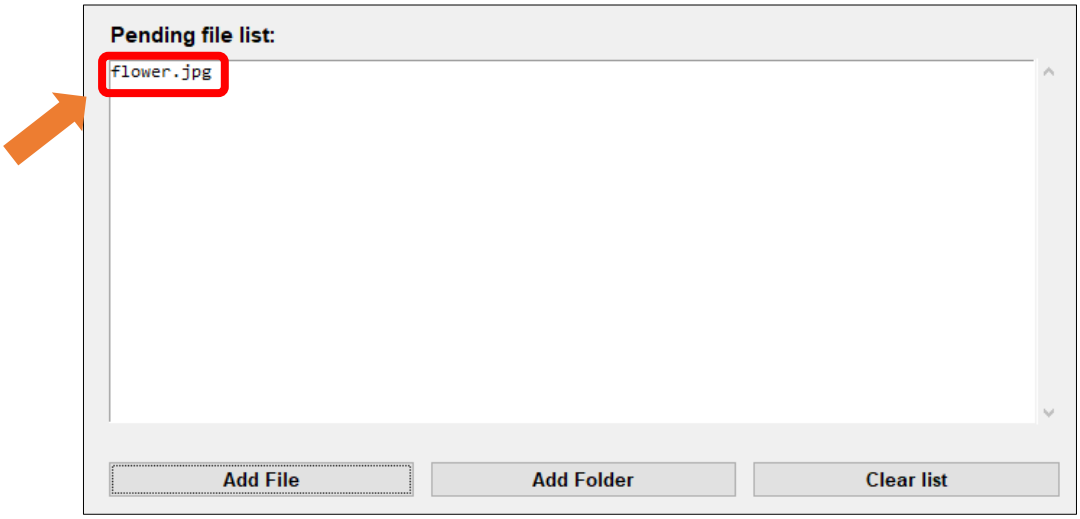
Click **"Add File"**



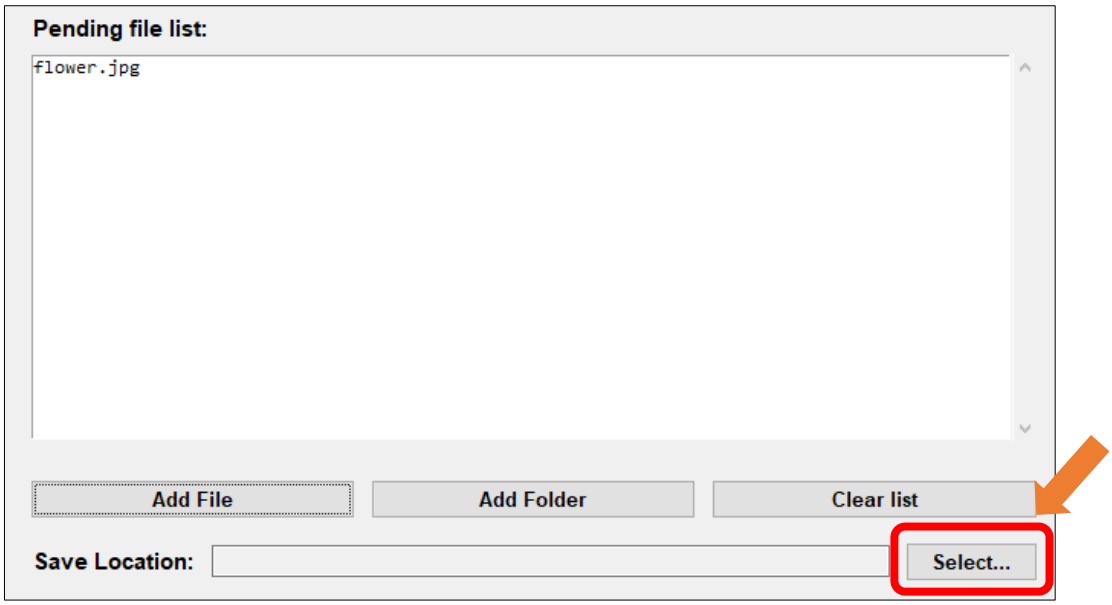
Select any image you like



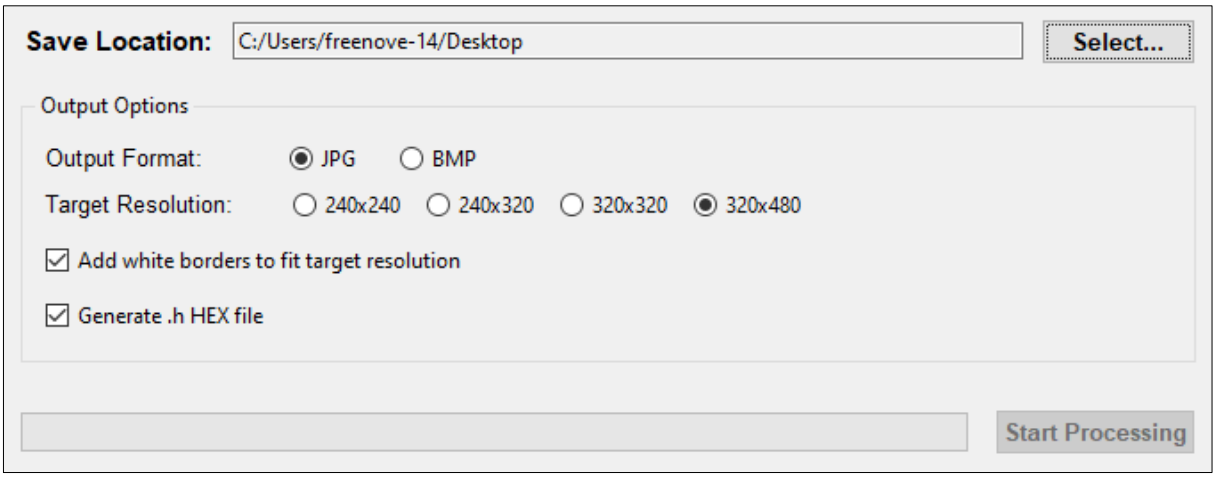
The image files from your folder will now appear in the **Pending File List**.



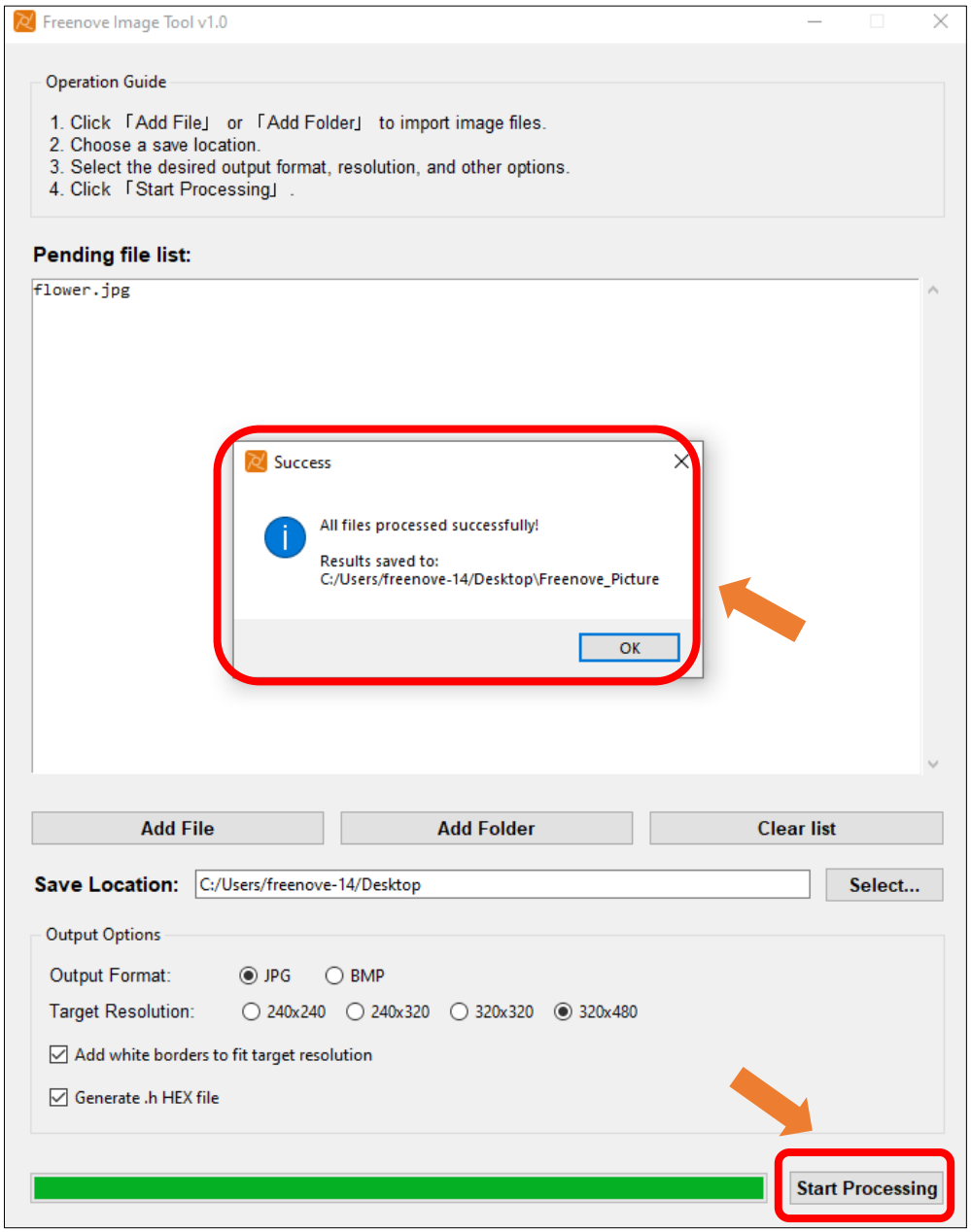
Click "Select…" to change the save location.



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.

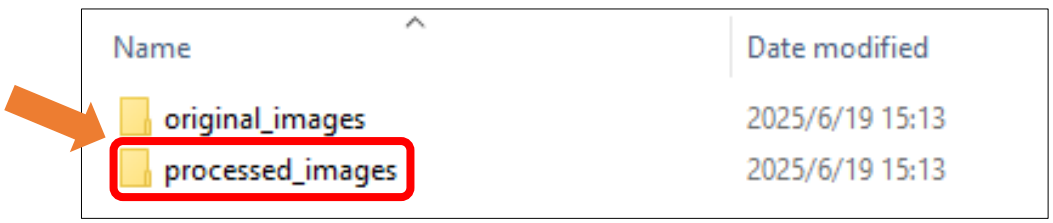


Click **Start Processing**. Wait for the progress bar to complete and the target folder will be generated.

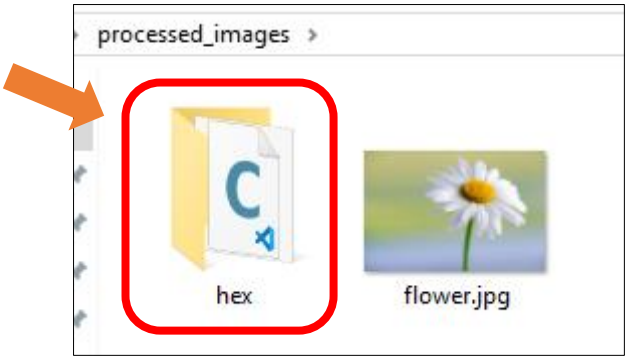


There will be three folders in the generated folder
original_images: backup of images before processing
processed_images: processed images
processed_images\hex: generated .h files corresponding to the images

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

Project 10.3 SD JPG

Having learned SD card operations in earlier chapters, this section will demonstrate how to render images stored on the SD card to the display.

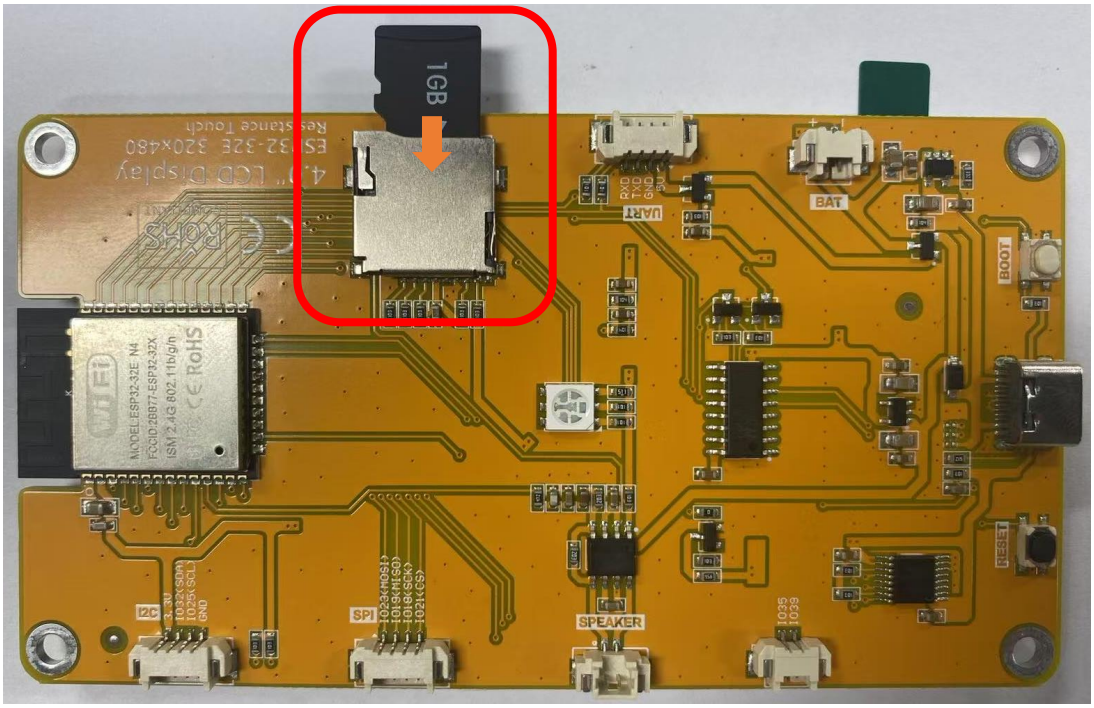
Component List

Freenove ESP32 Display x 1 	USB cable x1 

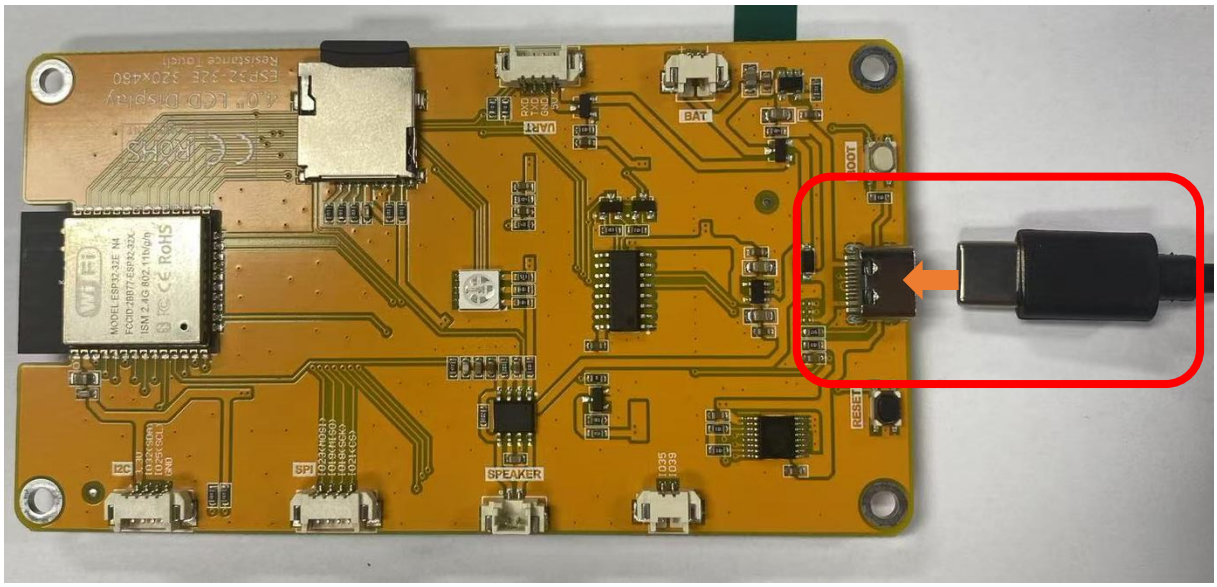
This kit does not include an SD card and the card reader, please buy them yourself. For more details, please refer to SD card.

Circuit

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



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



Sketch

Open“**Sketch_10.3_SD_Jpg**” folder under **“Freenove_ESP32_Display\Sketch”** and double-click “**Sketch_10.3_SD_Jpg.ino**”.

Sketch_10.3_SD_Jpg

The following is the program code:

```
/*
* @ File:   Sketch_11.3_SD_Jpg.ino
* @ Author: [Zhentao Lin]
* @ Date:   [2025-06-19]
*/

#include <TJpg_Decoder.h>  // JPEG decoder library
#include <TFT_eSPI.h>  // TFT display library
TFT_eSPI tft = TFT_eSPI();

#define SD_VSPI_SS 5     // SD card chip select pin
#define SD_VSPI_MOSI 23  // SD card SPI MOSI pin
#define SD_VSPI_MISO 19  // SD card SPI MISO pin
#define SD_VSPI_SCK 18   // SD card SPI SCK pin
SPIClass* vspi = NULL;   // VSPI

bool tft_output(int16_t x, int16_t y, uint16_t w, uint16_t h, uint16_t* bitmap) {
   if (y >= tft.height()) return 0;
   tft.pushImage(x, y, w, h, bitmap);
   return 1;
}

```

```
23  void setup() {
24    Serial.begin(115200);
25    Serial.println("\n\n Testing TJpg_Decoder library");
26    tft.begin();
27    // tft.setRotation(1);
28    vspi = new SPIClass(VSPI);
29    vspi->begin(SD_VSPI_SCK, SD_VSPI_MISO, SD_VSPI_MOSI, SD_VSPI_SS);
30    if (!SD.begin(SD_VSPI_SS, *vspi, 1000000)) {
31      Serial.println(F("SD.begin failed!"));
32      while (1) delay(0);
33    }
34    Serial.println("\r\nInitialisation done.");
35    tft.setTextColor(0xFFFF, 0x0000);
36    tft.fillScreen(TFT_BLACK);
37    TJpgDec.setJpgScale(1);
38    tft.setSwapBytes(true);
39    TJpgDec.setCallback(tft_output);
40  }
41
42  void loop() {
43    uint32_t t = millis();
44    uint16_t w = 0, h = 0;
45    TJpgDec.getSdJpgSize(&w, &h, "/Board.jpg");
46    Serial.print("Width = ");
47    Serial.print(w);
48    Serial.print(", height = ");
49    Serial.println(h);
50    TJpgDec.drawSdJpg(0, 0, "/Board.jpg");
51    t = millis() - t;
52    Serial.print(t);
53    Serial.println(" ms");
54    delay(2000);
55  }
```

Code Explanation

Included the necessary header files.

```
7  #include <TFT_eSPI.h>      // TFT display library
8  #include <TJpg_Decoder.h>  // Include JPEG decoder library
```

Create TFT object instance.

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

Define pins for the SD card.

```
11  #define SD_VSPI_SS 5     // SD card chip select pin
12  #define SD_VSPI_MOSI 23  // SD card SPI MOSI pin
13  #define SD_VSPI_MISO 19  // SD card SPI MISO pin
14  #define SD_VSPI_SCK 18   // SD card SPI SCK pin
```

JPEG decoding callback function

```
// Callback function to draw decoded JPEG blocks on screen
bool tft_output(int16_t x, int16_t y, uint16_t w, uint16_t h, uint16_t* bitmap) {
    if (y >= tft.height()) return 0;  // Stop decoding if off-screen

#ifdef USE_DMA
    if (dmaBufferSel) dmaBufferPtr = dmaBuffer2;
    else dmaBufferPtr = dmaBuffer1;
    dmaBufferSel = !dmaBufferSel;
    tft.pushImageDMA(x, y, w, h, bitmap, dmaBufferPtr);  // Start DMA transfer
#else
    tft.pushImage(x, y, w, h, bitmap);  // Draw without DMA
#endif

    return 1;  // Continue decoding next block
}
```

Obtain image dimensions.

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

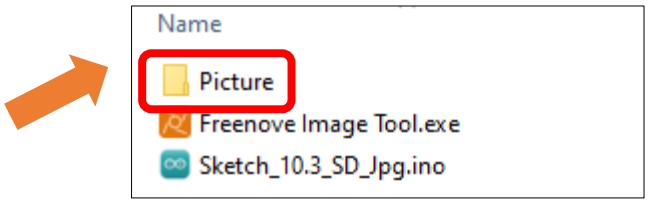
Draw an image on the TFT screen.

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

"Before uploading, remove the SD card from the Freenove ESP32 Display. Use a card reader to copy your image files to the root directory of the SD card.

WARNING: To ensure stable operation and prevent SD card damage, never insert or remove the SD card while the Freenove ESP32 Display is powered on. Sudden removal may cause data corruption or permanent hardware failure.

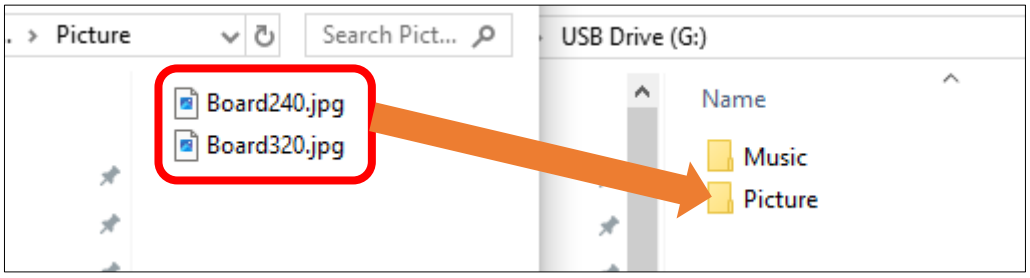
Double click to open **Sketch_10.3_SD_Jpg\Picture**



Copy the image files from the Picture folder to your SD card. Please select the appropriate files based on your Freenove ESP32 Display's screen resolution:

Files containing '240' in their names are for 240×320 resolution displays

Files containing '320' in their names are for 320×480 resolution displays"



Modify the corresponding filenames in the code. Here we take 'Board320.jpg' as an example:

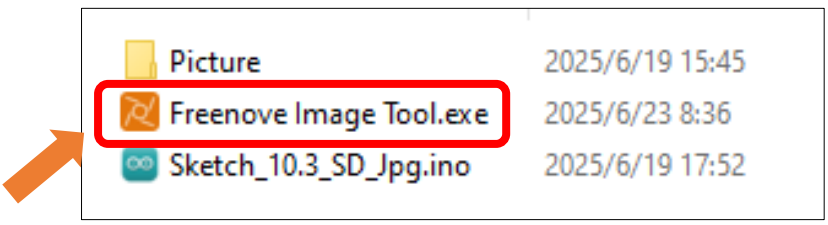
```
42  void loop() {
43    uint32_t t = millis();
44    uint16_t w = 0, h = 0;
45    TJpgDec.getSdJpgSize(&w, &h, "/Board320.jpg");
46    Serial.print("Width = ");
47    Serial.print(w);
48    Serial.print(", height = ");
49    Serial.println(h);
50    TJpgDec.drawSdJpg(0, 0, "/Board320.jpg");
51    t = millis() - t;
52    Serial.print(t);
```

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

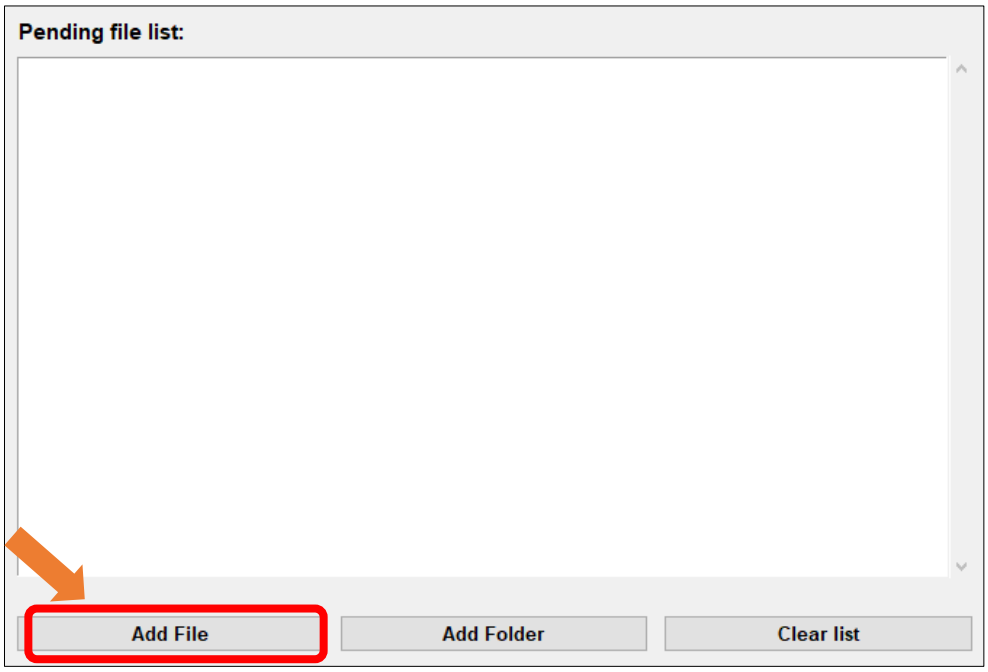
Custom Image Display

You can customize display images according to your preferences. **Ensure all custom images use JPG format.**

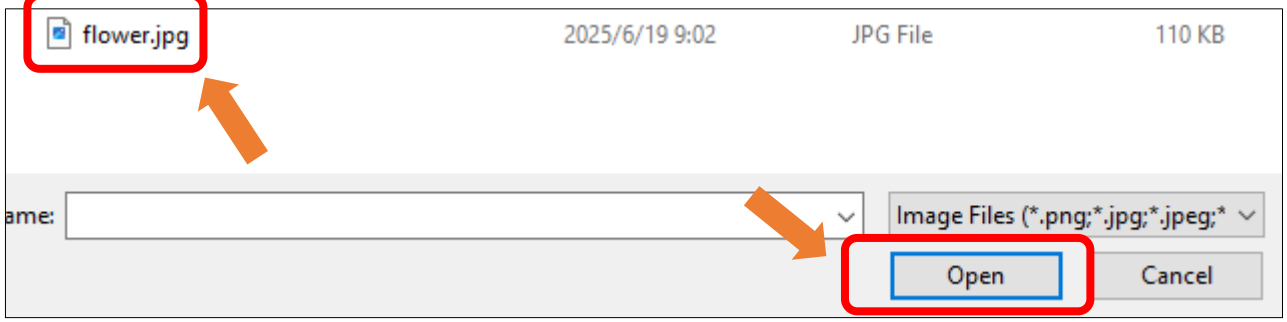
First, open **Freenove_ESP32_Display\Sketch\Sketch_10.3_SD_Jpg\Freenove Image Tool.exe**



Click **"Add File"**

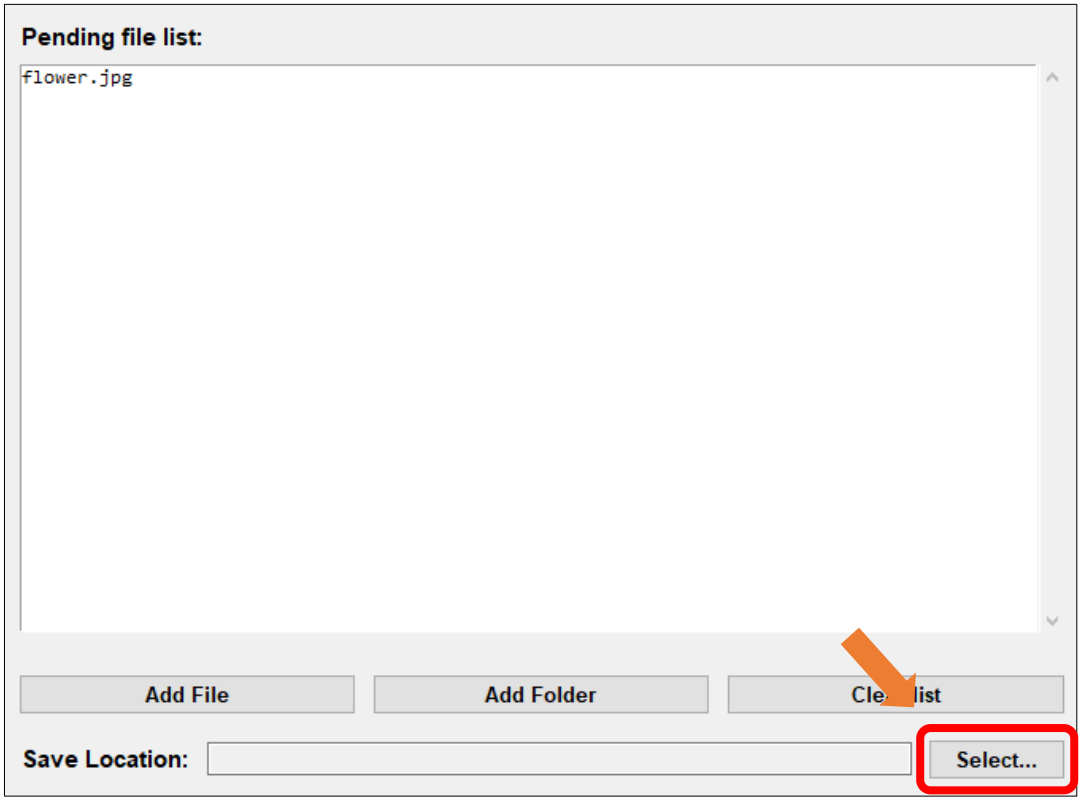


Select any JPG image you like.

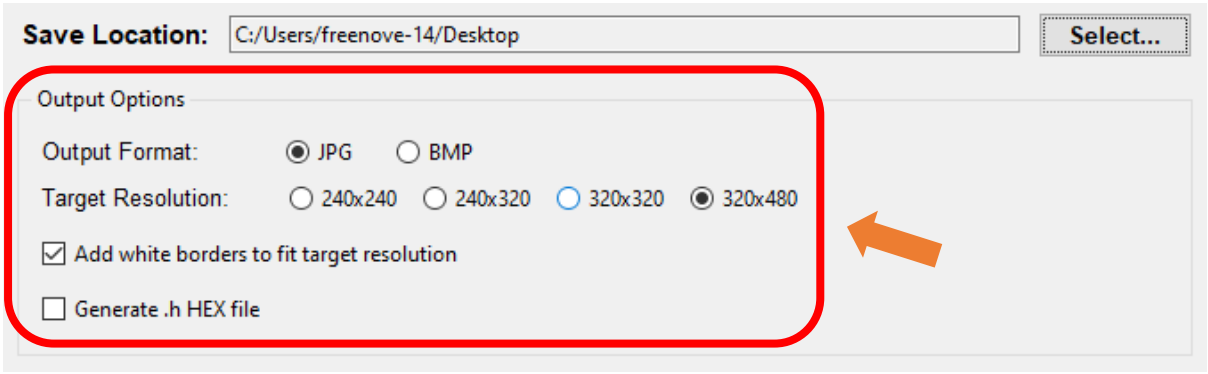


The image files from your folder will now appear in the **Pending File List**.

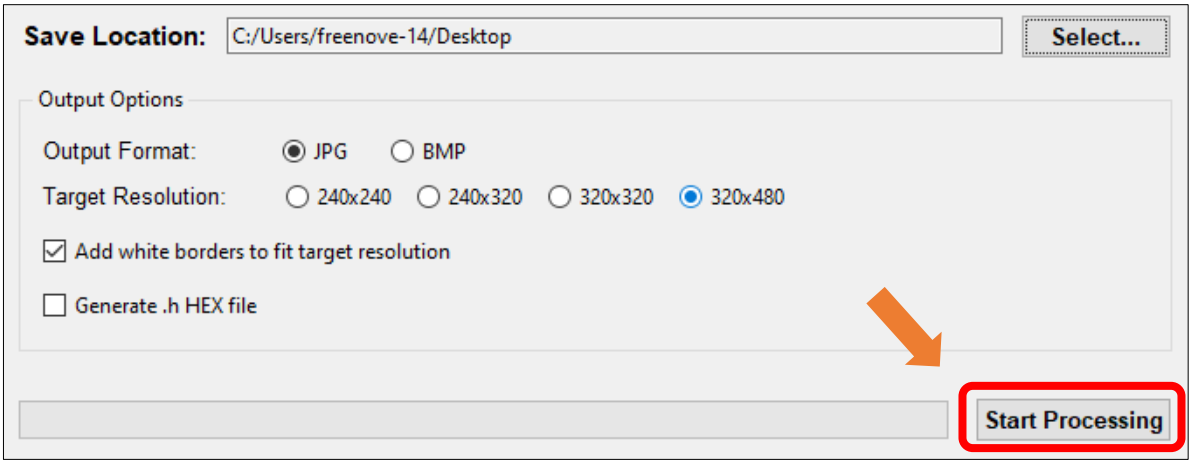
Click "**Select…**" to select the image save location.

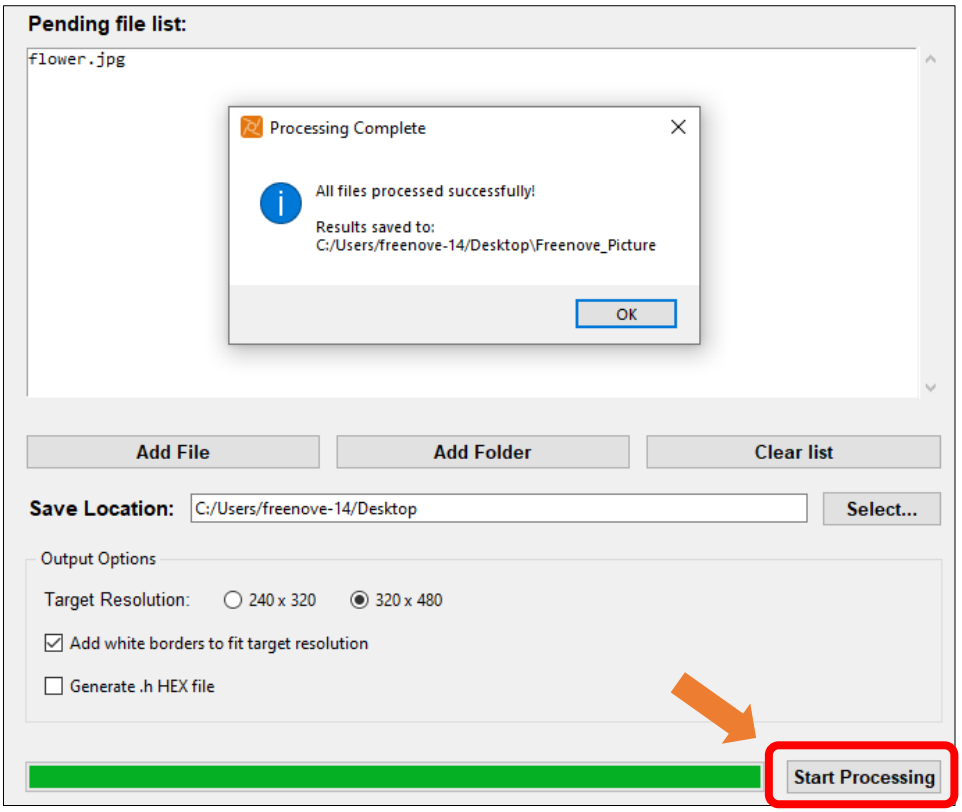


The resolution size is selected according to the screen resolution. Check the "**Add white borders to fit target resolution**"



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

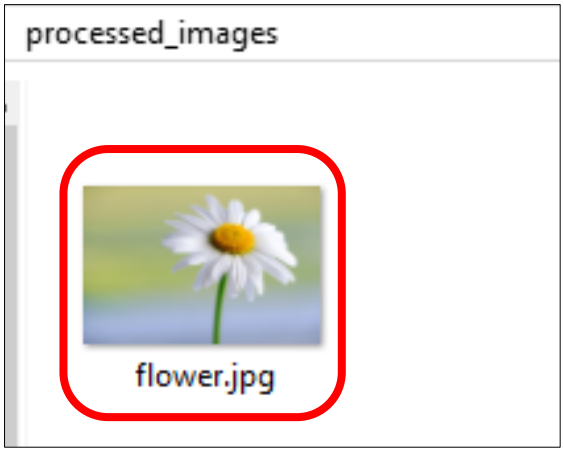




There will be two folders in the generated folder
original_images: backup of images before processing
processed_images: processed images

Double click to open **processed_images**

Upload the generated image file to the root directory of the SD card.



Change the file name in the code.

```
42  void loop() {
43    uint32_t t = millis();
44    uint16_t w = 0, h = 0;
45    TJpgDec.getSdJpgSize(&w, &h, "/flower.jpg");
46    Serial.print("Width = ");
47    Serial.print(w);
48    Serial.print(", height = ");
49    Serial.println(h);
50    TJpgDec.drawSdJpg(0, 0, "/flower.jpg");
51    t = millis() - t;
52    Serial.print(t);
```

Upload the sketch and the image will display on the screen.

Chapter 11 TFT Touch Calibration

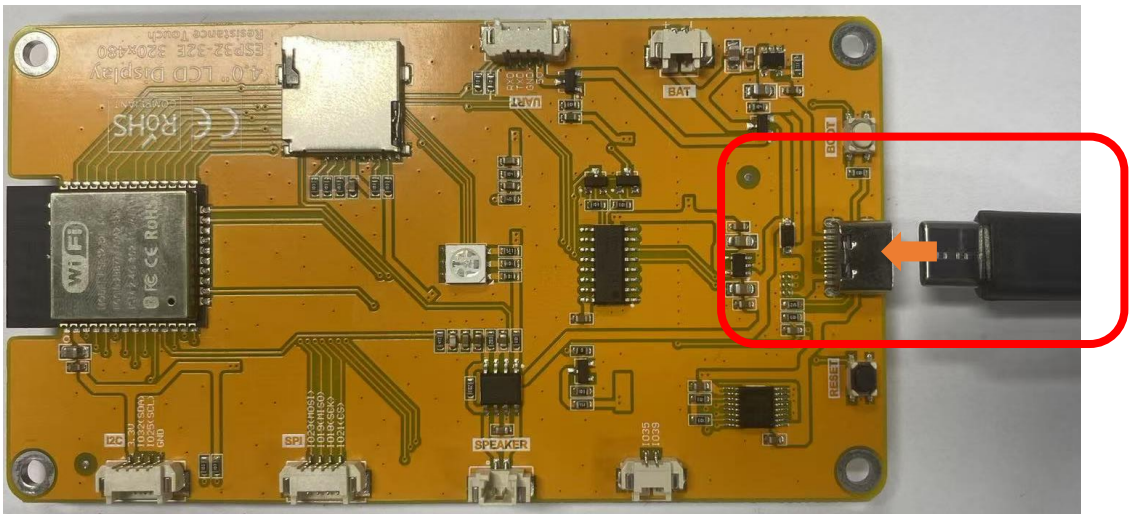
Project 11.1 TFT Touch Calibration

Component List

Freenove ESP32 Display x 1 	USB cable x1 
Stylus x 1 	

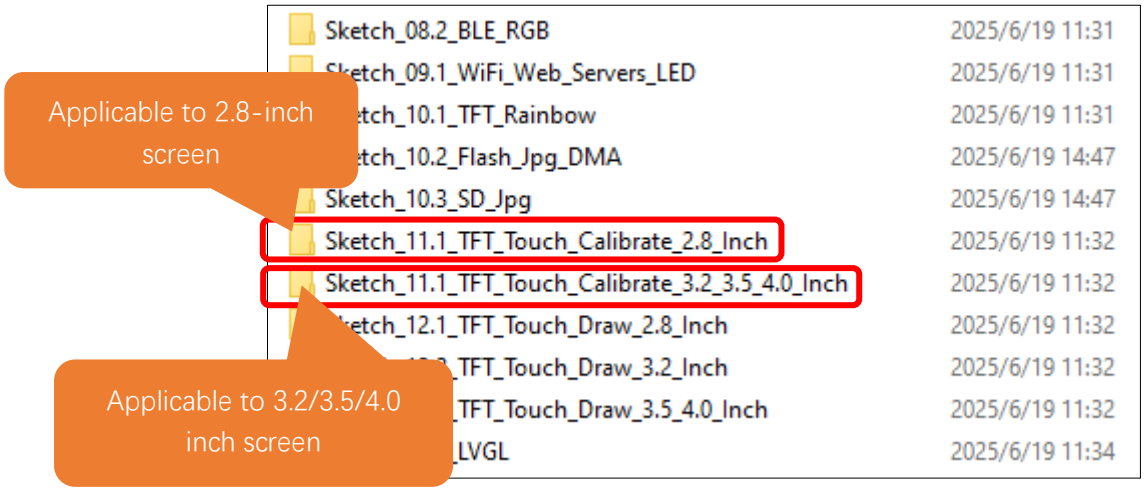
Circuit

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



Sketch

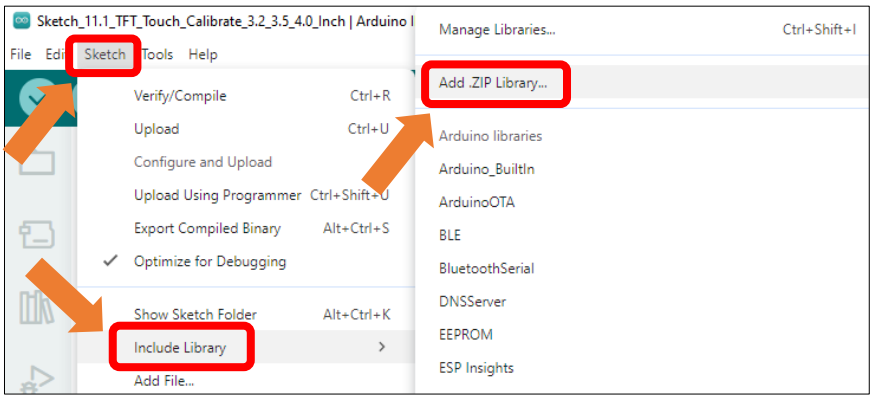
Open“**Sketch_11.1_TFT_Touch_Calibrate**” folder under **“Freenove_ESP32_Display\Sketch”** and double-click “**Sketch_11.1_TFT_Touch_Calibrate.ino**”.
The sample code provides two codes for different models of Freenove ESP32 Display. You can upload the corresponding sample code according to the actual model.



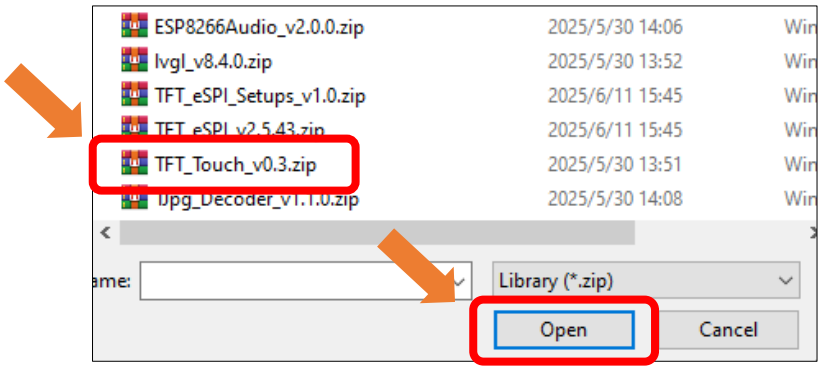
Sketch_11.1_TFT_Touch_Calibrate_2.8_Inch

Install Necessary Libraries

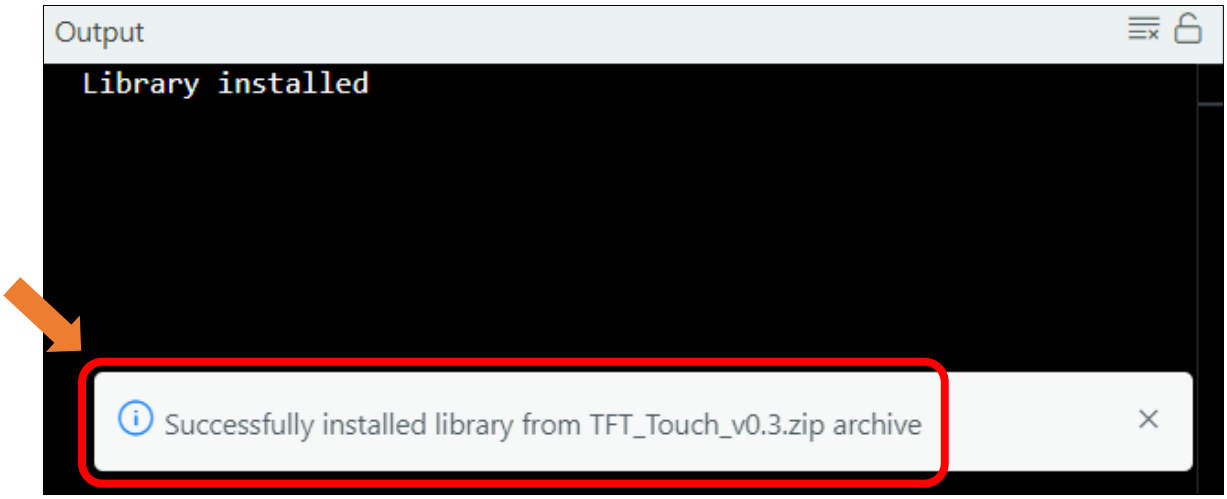
If your screen is a **2.8-inch** one, you also need to install the TFT_Touch library.
Click **Sketch** -> **Include Library** -> **Add .ZIP Library...**



Select “**TFT_Touch_v0.3.zip**” in the **Freenove_ESP32_Display\Libraries** folder and click Open.



The following message indicates the successful installation of the library.



The following is the program code:

```
/*
* @ File:   Sketch_11.1_TFT_Touch_Calibrate_3.2_3.5_4.0_Inch.ino
* @ Author: [Zhentao Lin]
* @ Date:   [2025-06-20]
*/

#include <SPI.h>
#include <TFT_eSPI.h>
TFT_eSPI tft = TFT_eSPI();

void setup() {
   // Use serial port
   Serial.begin(115200);

   // Initialise the TFT screen
   tft.init();

   // Set the rotation to the orientation you wish to use in your project before calibration
   // (the touch coordinates returned then correspond to that rotation only)
   tft.setRotation(1);

```

```
22      // Calibrate the touch screen and retrieve the scaling factors
23      touch_calibrate();
24
25  /*
26      // Replace above line with the code sent to Serial Monitor
27      // once calibration is complete, e.g.:
28      uint16_t calData[5] = { 286, 3534, 283, 3600, 6 };
29      tft.setTouch(calData);
30  */
31
32      // Clear the screen
33      tft.fillScreen(TFT_BLACK);
34      tft.drawCentreString("Touch screen to test!",tft.width()/2, tft.height()/2, 2);
35  }
36
37  void loop(void) {}
38
39  // Code to run a screen calibration, not needed when calibration values set in setup()
40  void touch_calibrate()
41  {
42      uint16_t calData[5];
43
44      // Calibrate
45      tft.fillScreen(TFT_BLACK);
46      tft.setCursor(20, 0);
47      tft.setTextFont(2);
48      tft.setTextSize(1);
49      tft.setTextColor(TFT_WHITE, TFT_BLACK);
50
51      tft.println("Touch corners as indicated");
52
53      tft.setTextFont(1);
54      tft.println();
55
56      tft.calibrateTouch(calData, TFT_MAGENTA, TFT_BLACK, 15);
57
58      Serial.println(); Serial.println();
59      Serial.println("// Use this calibration code in setup():");
60      Serial.print("  uint16_t calData[5] = ");
61      Serial.print("{ ");
62
63      for (uint8_t i = 0; i < 5; i++)
64      {
65        Serial.print(calData[i]);
```

```
66        if (i < 4) Serial.print(", ");
67      }
68
69      Serial.println(" };");
70      Serial.print("  tft.setTouch(calData);");
71      Serial.println(); Serial.println();
72
73      tft.fillScreen(TFT_BLACK);
74
75      tft.setTextColor(TFT_GREEN, TFT_BLACK);
76      tft.println("Calibration complete!");
77      tft.println("Calibration code sent to Serial port.");
78
79      delay(4000);
80  }
```

Code Explanation

Include the necessary header files.

```
7   #include <SPI.h>
8   #include <TFT_eSPI.h>
```

Create TFT object instance.

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

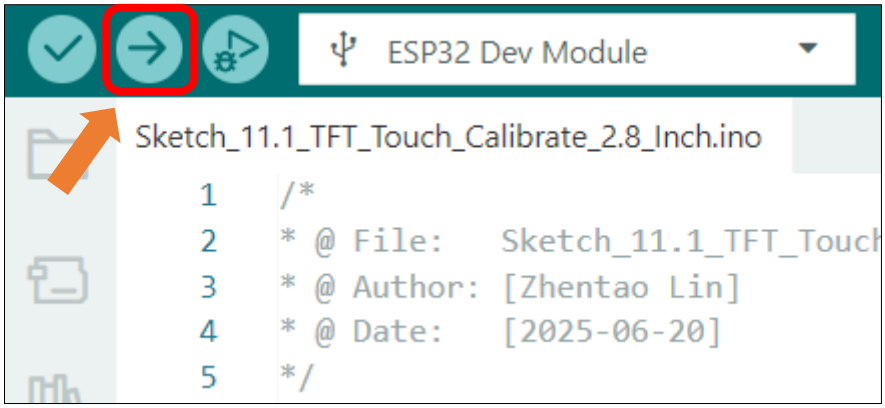
Adjust the display direction of the screen.

```
20  tft.setRotation(1);
```

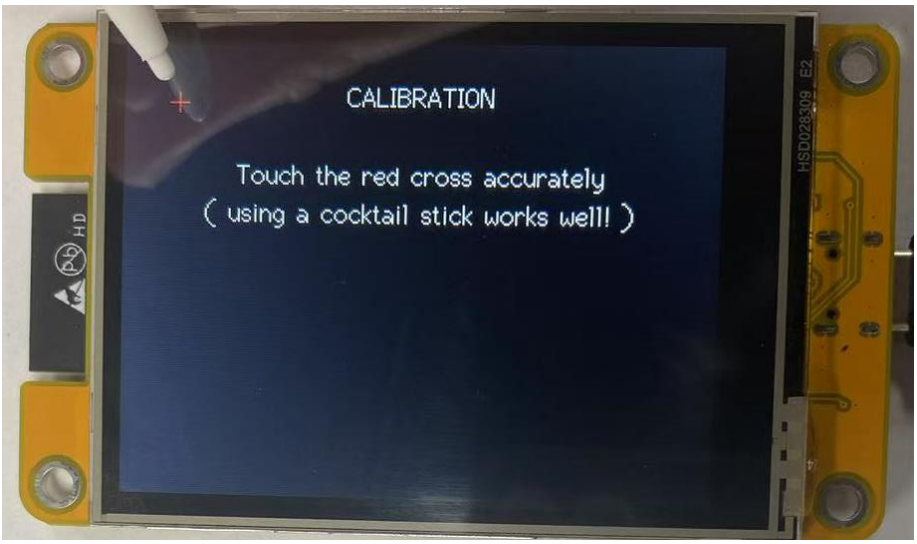
The touch calibration function.

```
56  tft.calibrateTouch(calData, TFT_MAGENTA, TFT_BLACK, 15);
```

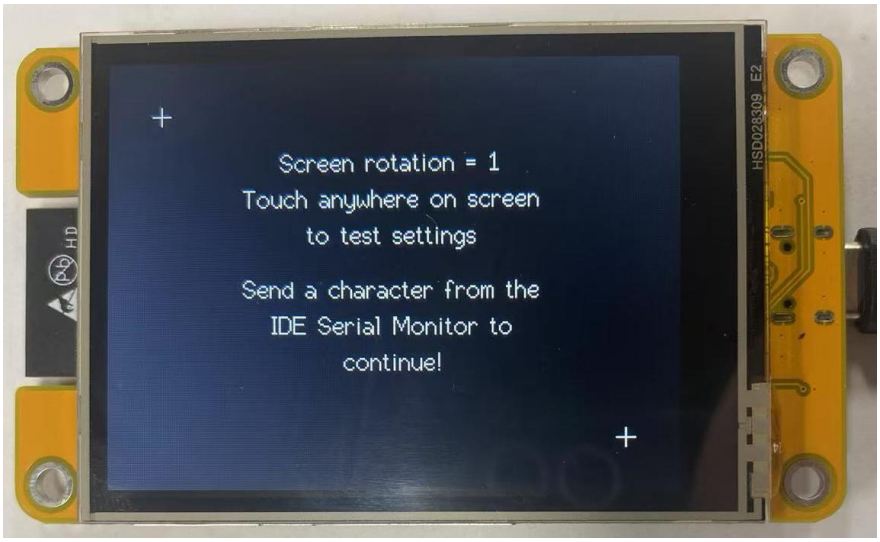
Click "**Upload**" to upload the code to Freenove ESP32 Display, set the baud rate to 115200.



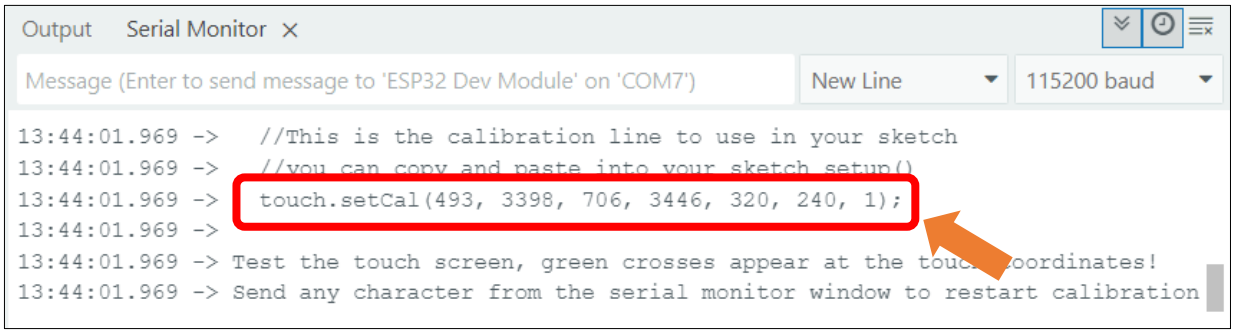
Click the positions of the red cross cursor on the screen in sequence according to the prompts on the display



When the following message appears on the screen, the calibration is complete.



Save the calibration results printed in the serial monitor.



Note: This calibration code only needs to run once during initial setup to obtain screen calibration parameters. After successful calibration, please save the calibration values for future use – no need to repeat this calibration procedure afterward.

Sketch_11.1_TFT_Touch_Calibrate_3.2_3.5_4.0_Inch

The following is the program code:

```
/*
* @ File:   Sketch_11.1_TFT_Touch_Calibrate_3.2_3.5_4.0_Inch.ino
* @ Author: [Zhentao Lin]
* @ Date:   [2025-06-20]
*/

#include <SPI.h>
#include <TFT_eSPI.h>
TFT_eSPI tft = TFT_eSPI();

void setup() {
   // Use serial port
   Serial.begin(115200);

   // Initialise the TFT screen
   tft.init();

   // Set the rotation to the orientation you wish to use in your project before calibration
   // (the touch coordinates returned then correspond to that rotation only)
   tft.setRotation(1);

   // Calibrate the touch screen and retrieve the scaling factors
   touch_calibrate();

/*
   // Replace above line with the code sent to Serial Monitor
   // once calibration is complete, e.g.:
   uint16_t calData[5] = { 286, 3534, 283, 3600, 6 };
   tft.setTouch(calData);
*/

   // Clear the screen
   tft.fillScreen(TFT_BLACK);
   tft.drawCentreString("Touch screen to test!",tft.width()/2, tft.height()/2, 2);
}

void loop(void) {}

// Code to run a screen calibration, not needed when calibration values set in setup()
void touch_calibrate()
{
   uint16_t calData[5];
```

```
43
44      // Calibrate
45      tft.fillScreen(TFT_BLACK);
46      tft.setCursor(20, 0);
47      tft.setTextFont(2);
48      tft.setTextSize(1);
49      tft.setTextColor(TFT_WHITE, TFT_BLACK);
50
51      tft.println("Touch corners as indicated");
52
53      tft.setTextFont(1);
54      tft.println();
55
56      tft.calibrateTouch(calData, TFT_MAGENTA, TFT_BLACK, 15);
57
58      Serial.println(); Serial.println();
59      Serial.println("// Use this calibration code in setup():");
60      Serial.print("  uint16_t calData[5] = ");
61      Serial.print("{ ");
62
63      for (uint8_t i = 0; i < 5; i++)
64      {
65        Serial.print(calData[i]);
66        if (i < 4) Serial.print(", ");
67      }
68
69      Serial.println(" };");
70      Serial.print("  tft.setTouch(calData);");
71      Serial.println(); Serial.println();
72
73      tft.fillScreen(TFT_BLACK);
74
75      tft.setTextColor(TFT_GREEN, TFT_BLACK);
76      tft.println("Calibration complete!");
77      tft.println("Calibration code sent to Serial port.");
78
79      delay(4000);
80  }
```

Code Explanation

Include necessary header files.

```
7   #include <SPI.h>
8   #include <TFT_eSPI.h>
```

Create TFT object instance.

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

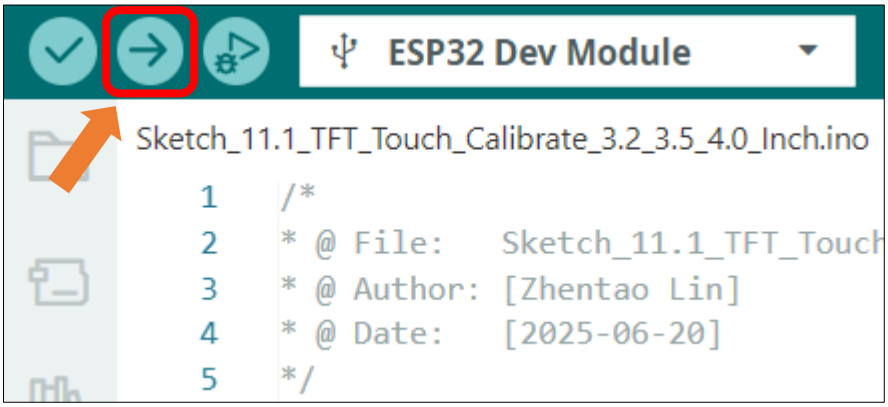
Adjust the display direction.

```
20  tft.setRotation(1);
```

Touch calibrate function.

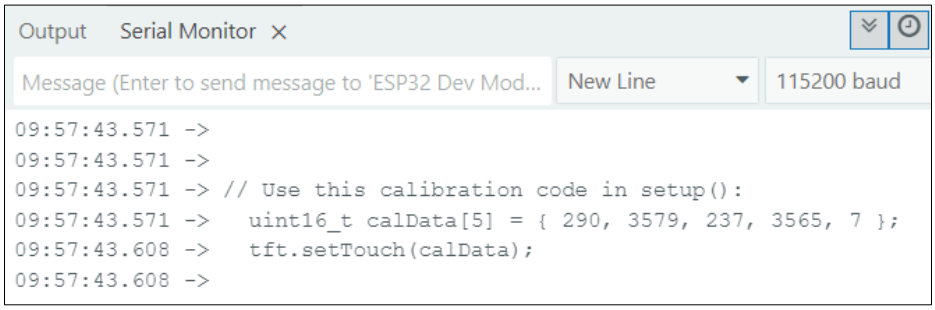
```
56  tft.calibrateTouch(calData, TFT_MAGENTA, TFT_BLACK, 15);
```

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



Follow the on-screen instructions and sequentially tap the positions indicated by the purple arrows on the display.

The calibration results will be printed in the Serial Monitor. After calibration, you must **save** these values, as they will be used in the code implementation in the next section.



This calibration routine should be executed only once during initial device setup to acquire screen calibration parameters.

Chapter 12 TFT Touch Drawing

After learning this chapter, you will be able to draw freely on the screen.

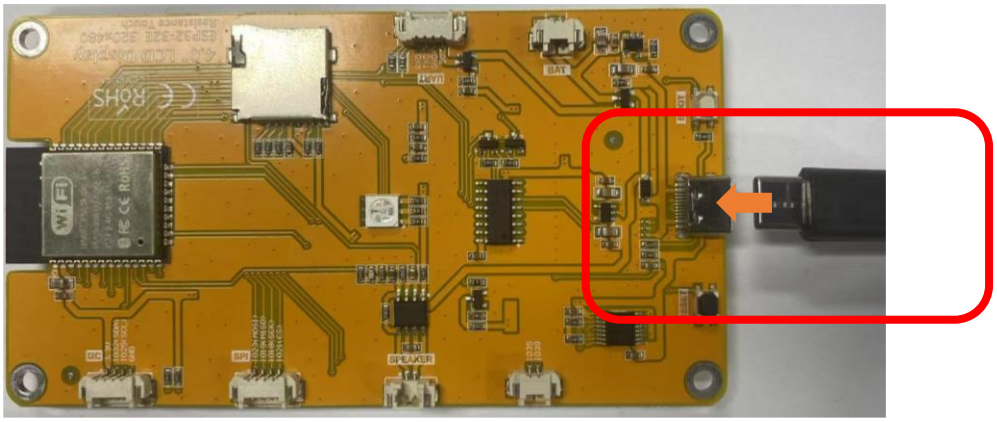
Project 12.1 TFT Touch Drawing

Component List

Freenove ESP32 Display x 1 	USB cable x1 
Style x 1 	

Circuit

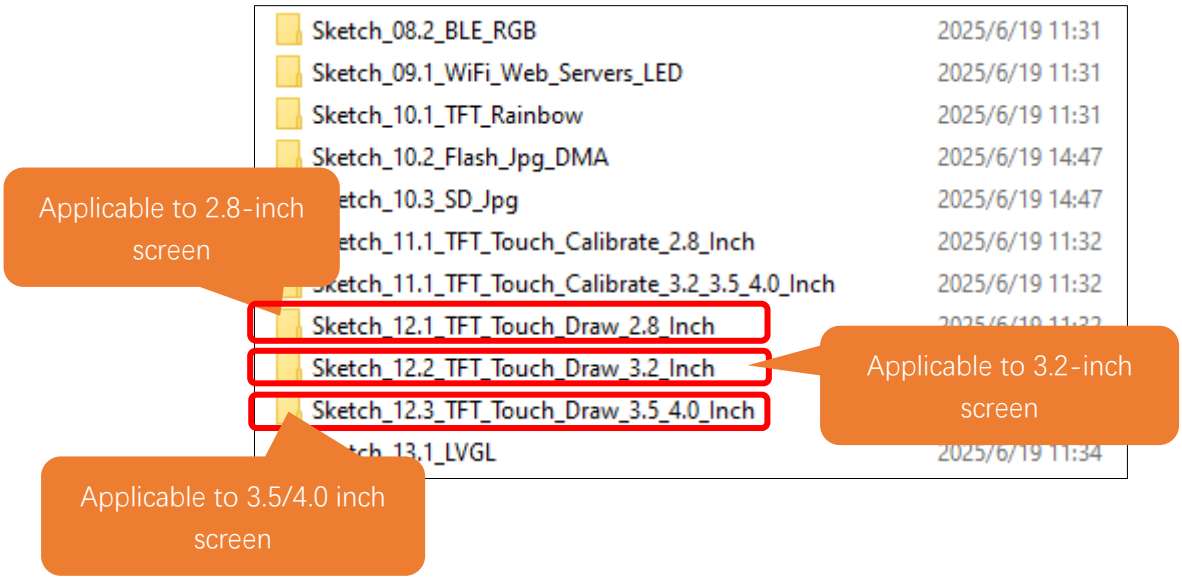
Connect Freenove ESP32-S3 to the computer using the USB cable.



Sketch

Open“**Sketch_12.1_TFT_Touch_Calibrate**” folder under **“Freenove_ESP32_Display\Sketch”** and double-click “**Sketch_12.1_TFT_Touch_Calibrate.ino**”.

The sample code provides two codes for different models of Freenove ESP32 Display. You can upload the corresponding sample code according to the actual model.



Sketch_12.1_TFT_Touch_Draw_2.8_Inch

The following is the program code:

```
/*
* @ File:   Sketch_12.1_TFT_Touch_Draw_2.8_Inch.ino
* @ Author: [Zhentao Lin]
* @ Date:   [2025-06-19]
*/

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

// Call up touch screen library
#include <TFT_Touch.h>

// Invoke custom TFT driver library
TFT_eSPI tft = TFT_eSPI();       // Invoke custom library

#define SCREEN_WIDTH 240
#define SCREEN_HEIGHT 320

#define DOUT 39  /* Data out pin (T_DO) of touch screen */
#define DIN  32  /* Data in pin (T_DIN) of touch screen */
#define DCS  33  /* Chip select pin (T_CS) of touch screen */
#define DCLK 25  /* Clock pin (T_CLK) of touch screen */
```

```
22
23  /* Create an instance of the touch screen library */
24  TFT_Touch touch = TFT_Touch(DCS, DCLK, DIN, DOUT);
25
26  // Define coordinate variables
27  uint16_t x, y;
28  uint16_t last_x, last_y;
29
30  const uint8_t ColorPalette = 21;      // Height of the color palette area
31  const uint8_t BrushSize = 2;          // Brush thickness (radius)
32  const uint8_t LineSize = 5;           // Line thickness (radius)
33  const uint8_t color_num = 12;
34  int color = TFT_WHITE;
35
36  unsigned int colors[color_num] = { TFT_RED, TFT_PINK, TFT_GREEN, TFT_BLUE, TFT_OLIVE ,
37  TFT_CYAN, TFT_YELLOW, TFT_WHITE, TFT_MAGENTA, TFT_ORANGE, TFT_BLACK, TFT_SILVER};
38
39  void drawUI()
40  {
41    // Draw color palette
42    for (int i = 0; i < color_num; i++) {
43      tft.fillRect(i * ColorPalette, 0, ColorPalette, ColorPalette, colors[i]);
44    }
45
46    // Draw clear screen button
47    tft.setCursor(260 , 3, 2);
48    tft.setTextColor(TFT_WHITE);
49    tft.print("Clear");
50
51    // Display current color
52    tft.fillRect(300, 5, 12, 12, color);
53
54    // Add palette border/outline
55    tft.drawRect(0, 0, SCREEN_HEIGHT, ColorPalette, TFT_WHITE);
56  }
57
58  void setup()
59  {
60    Serial.begin(115200);
61
62    tft.init();
63
64    //This is the calibration line produced by the TFT_Touch_Calibrate_v2 sketch
65    // touch.setCal(493, 3398, 706, 3446, 320, 240, 1);
```

```
66
67      // Set the TFT and touch screen to landscape orientation
68      tft.setRotation(1);
69      touch.setRotation(1);
70
71      tft.fillScreen(TFT_BLACK);
72
73      drawUI();
74  }
75
76  /* Main program */
77  void loop()
78  {
79      // Check if the touch screen is currently pressed
80      // Raw and coordinate values are stored within library at this instant
81      // for later retrieval by GetRaw and GetCoord functions.
82
83      if (touch.Pressed()) // Note this function updates coordinates stored within library
84  variables
85      {
86        // Read the current X and Y axis as co-ordinates at the last touch time
87        // The values were captured when Pressed() was called!
88        x = touch.X();
89        y = touch.Y();
90
91        Serial.print(x); Serial.print(","); Serial.println(y);
92
93        /* Color palette area logic */
94        if(y < ColorPalette + 3)
95        {
96          // Enter color selection area, stroke interrupted
97          last_x = 0;
98          last_y = 0;
99
100         /* Determine stylus click area */
101         // Clicked Clear button
102         if(x >= ColorPalette * color_num)
103         {
104           tft.fillRect(0, ColorPalette, SCREEN_HEIGHT, SCREEN_WIDTH - ColorPalette, color);
105         }
106         // Clicked color palette
107         else{
108           // Calculate current color based on x-coordinate
109           color = colors[x / ColorPalette];
```

```
110
111             // Update display of current color
112             tft.fillRect(300, 5, 12, 12, color);
113           }
114           delay(5);
115           return;
116         }
117
118         /* Drawing logic */
119         // Start of a new stroke
120         if(last_x == 0 && last_y == 0)
121         {
122           // Record current position
123           last_x = x;
124           last_y = y;
125         }
126
127         // Draw a line between the previous position and current position
128         tft.drawWideLine(last_x, last_y, x, y, LineSize, color, color);
129
130         // Update coordinates
131         last_x = x;
132         last_y = y;
133       }
134       else{
135         // Reset coordinates when stylus leaves the screen
136         last_x = 0;
137         last_y = 0;
138       }
139   }
```

Code Explanation

Include the necessary header files.

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

Create TFT object instance.

```
12  // Invoke custom TFT driver library
13  TFT_eSPI tft = TFT_eSPI();       // Invoke custom library
```

Define screen resolution.

```
15  #define SCREEN_WIDTH 240
16  #define SCREEN_HEIGHT 320
```

Draw color palette.

```
// Draw color palette
for (int i = 0; i < color_num; i++) {
    tft.fillRect(i * ColorPalette, 0, ColorPalette, ColorPalette, colors[i]);
}
```

Detect whether the screen is pressed.

```
if (touch.Pressed())
```

Store the x and y coordinates of the touch point in variables.

```
x = touch.X();
y = touch.Y();
```

When the Clear button is pressed, fill the screen with the current color.

```
// Clicked Clear button
if(x >= ColorPalette * color_num)
{
  tft.fillRect(0, ColorPalette, SCREEN_HEIGHT, SCREEN_WIDTH - ColorPalette, color);
}
```

Calculate the selected color based on the coordinates of the touch point

```
// Clicked color palette
else{
  // Calculate current color based on x-coordinate
  color = colors[x / ColorPalette];

  // Update display of current color
  tft.fillRect(300, 5, 12, 12, color);
}
```

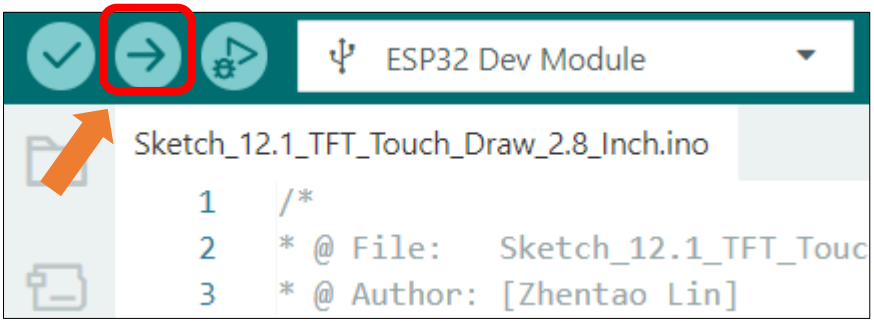
Draw a line based on coordinates

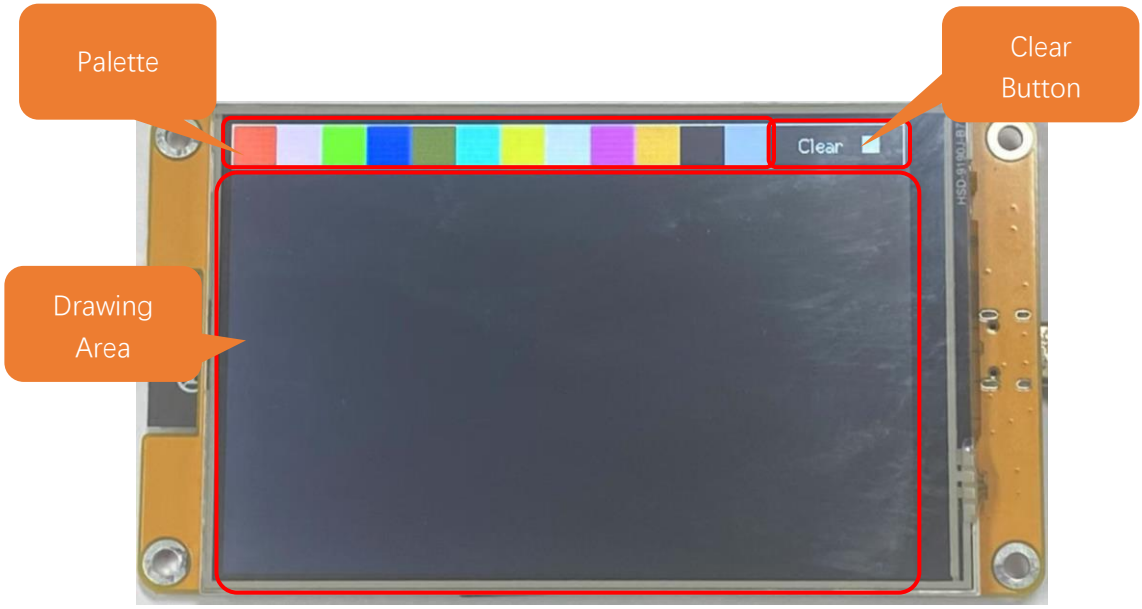
```
/* Drawing logic */
// Start of a new stroke
if(last_x == 0 && last_y == 0)
{
  // Record current position
  last_x = x;
  last_y = y;
}

// Draw a line between the previous position and current position
tft.drawWideLine(last_x, last_y, x, y, LineSize, color, color);

// Update coordinates
last_x = x;
last_y = y;
```

Add the calibration values obtained in the previous section to the code.

```
57  void setup()
58  {
59    Serial.begin(115200);
60
61    tft.init();
62
63    //This is the calibration line produced by the TFT_Touch_Calibrate_v2 sketch
64    touch.setCal(493, 3398, 706, 3446, 320, 240, 1);
65
66    // Set the TFT and touch screen to landscape orientation
```

Click "**Upload**" to upload the code to Freenove ESP32 Display. Set the baud rate to115200





Sketch_12.2_TFT_Touch_Draw_3.2_Inch

The following is the program code:

```
/*
* @ File:   Sketch_12.2_TFT_Touch_Draw_3.2_Inch.ino
* @ Author: [Zhentao Lin]
* @ Date:   [2025-06-20]
*/

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

#define SCREEN_WIDTH 240
#define SCREEN_HEIGHT 320

TFT_eSPI tft = TFT_eSPI(SCREEN_WIDTH, SCREEN_HEIGHT);

// Define coordinate variables
uint16_t x, y;
uint16_t last_x, last_y;

const uint8_t ColorPalette = 21;      // Height of the color palette area
const uint8_t BrushSize = 3;          // Brush thickness (radius)
const uint8_t LineSize = 6;           // Line thickness (radius)
const uint8_t color_num = 12;
int color = TFT_WHITE;

unsigned int colors[color_num] = { TFT_RED, TFT_PINK, TFT_GREEN, TFT_BLUE, TFT_OLIVE ,
TFT_CYAN, TFT_YELLOW, TFT_WHITE, TFT_MAGENTA, TFT_ORANGE, TFT_BLACK, TFT_SILVER};

void drawUI()
{
   // Draw color palette
   for (int i = 0; i < color_num; i++) {
     tft.fillRect(i * ColorPalette, 0, ColorPalette, ColorPalette, colors[i]);
   }

   // Draw clear screen button
   tft.setCursor(260 , 3, 2);
   tft.setTextColor(TFT_WHITE);
   tft.print("Clear");

   // Display current color
   tft.fillRect(300, 5, 12, 12, color);

   // Add palette border/outline
```

```
    tft.drawRect(0, 0, SCREEN_HEIGHT, ColorPalette, TFT_WHITE);
}

void setup() {
    Serial.begin(115200);
    tft.init();
    uint16_t calData[5] = { 412, 3502, 262, 3596, 3 };
    tft.setTouch(calData);
    tft.setRotation(1);
    tft.fillScreen(TFT_BLACK);

    drawUI();
}

void loop() {
    bool pressed = tft.getTouch(&x, &y);

    if (pressed) {
      Serial.printf("x: %d, y: %d\r\n", x, y);

      /* Color palette area logic */
      if(y < ColorPalette + 3)
      {
        // Enter color selection area, stroke interrupted
        last_x = 0;
        last_y = 0;

        /* Determine stylus click area */
        // Clicked Clear button
        if(x >= ColorPalette * color_num)
        {
          tft.fillRect(0, ColorPalette, SCREEN_HEIGHT, SCREEN_WIDTH - ColorPalette, color);
        }
        // Clicked color palette
        else{
          // Calculate current color based on x-coordinate
          color = colors[x / ColorPalette];

          // Update display of current color
          tft.fillRect(300, 5, 12, 12, color);
        }
        delay(5);
        return;
      }
```

```
87
88        /* Drawing logic */
89        // Start of a new stroke
90        if(last_x == 0 && last_y == 0)
91        {
92          // Record current position
93          last_x = x;
94          last_y = y;
95        }
96
97        // Draw a line between the previous position and current position
98        tft.drawWideLine(last_x, last_y, x, y, LineSize, color, color);
99
100       // Update coordinates
101       last_x = x;
102       last_y = y;
103     }
104     else{
105       // Reset coordinates when stylus leaves the screen
106       last_x = 0;
107       last_y = 0;
108     }
109 }
```

Code Explanation

Include the necessary library.

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

Create TFT object instance.

```
12  // Invoke custom TFT driver library
13  TFT_eSPI tft = TFT_eSPI();       // Invoke custom library
```

Define screen resolution.

```
15  #define SCREEN_WIDTH 240
16  #define SCREEN_HEIGHT 320
```

Apply the calibration data.

```
34  uint16_t calData[5] = { 412, 3502, 262, 3596, 3 };
35  tft.setTouch(calData);
```

Draw color palette.

```
28  // Draw color palette
29  for (int i = 0; i < color_num; i++) {
30     tft.fillRect(i * ColorPalette, 0, ColorPalette, ColorPalette, colors[i]);
31  }
```

Detect whether the touch screen is pressed and store the x and y coordinates of the touch point in variables.

```
58  bool pressed = tft.getTouch(&x, &y);
```

When the Clear button is pressed, fill the screen with the current color.

```
71  // Clicked Clear button
72  if(x >= ColorPalette * color_num)
73  {
74    tft.fillRect(0, ColorPalette, SCREEN_HEIGHT, SCREEN_WIDTH - ColorPalette, color);
75  }
```

Calculate the selected color based on the coordinates of the touch point

```
77  // Clicked color palette
78  else{
79    // Calculate current color based on x-coordinate
80    color = colors[x / ColorPalette];
81
82    // Update display of current color
83    tft.fillRect(300, 5, 12, 12, color);
84  }
```

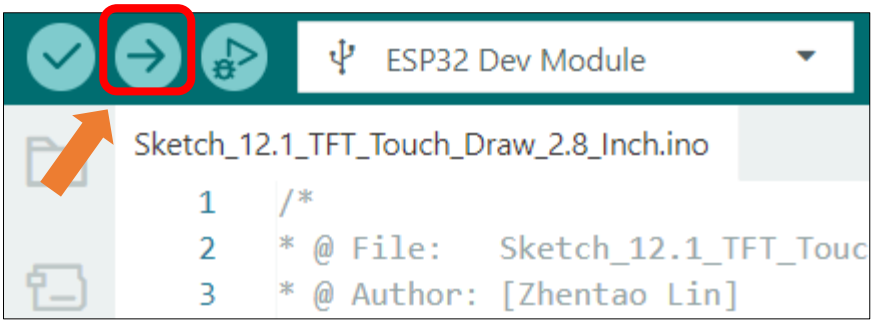
Draw a line based on coordinates

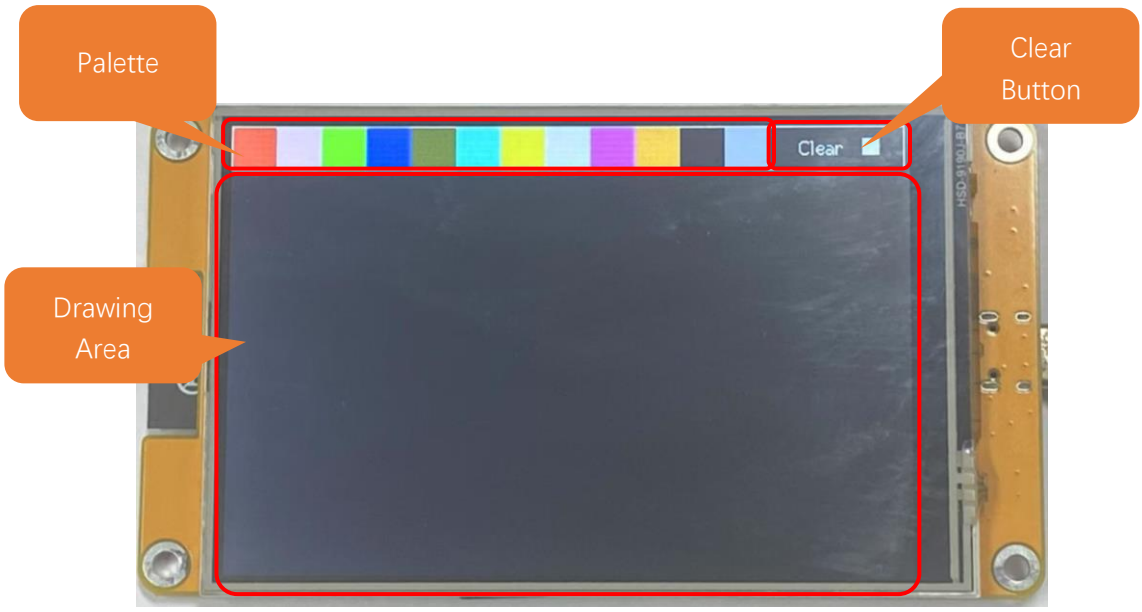
```
88  /* Drawing logic */
89  // Start of a new stroke
90  if(last_x == 0 && last_y == 0)
91  {
92    // Record current position
93    last_x = x;
94    last_y = y;
95  }
96
97  // Draw a line between the previous position and current position
98  tft.drawWideLine(last_x, last_y, x, y, LineSize, color, color);
99
100 // Update coordinates
101 last_x = x;
102 last_y = y;
```

Add the calibration value obtained from the previous section to the code.

```
57  void setup()
58  {
59    Serial.begin(115200);
60
61    tft.init();
62
63    //This is the calibration line produced by the TFT_Touch_Calibrate_v2 sketch
64    touch.setCal(493, 3398, 706, 3446, 320, 240, 1);
65
66    // Set the TFT and touch screen to landscape orientation
```

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





Sketch_12.3_TFT_Touch_Draw_3.5_4.0_Inch

The following is the program code:

```
/*
* @ File:   Sketch_12.3_TFT_Touch_Draw_3.5_4.0_Inch.ino
* @ Author: [Zhentao Lin]
* @ Date:   [2025-06-20]
*/

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

#define SCREEN_WIDTH 320
#define SCREEN_HEIGHT 480

TFT_eSPI tft = TFT_eSPI(SCREEN_WIDTH, SCREEN_HEIGHT);

// Define coordinate variables
uint16_t x, y;
uint16_t last_x, last_y;

const uint8_t ColorPalette = 32;      // Height of the color palette area
const uint8_t LineSize = 7;           // Line thickness (radius)
const uint8_t color_num = 12;
int color = TFT_WHITE;

unsigned int colors[color_num] = { TFT_RED, TFT_PINK, TFT_GREEN, TFT_BLUE, TFT_OLIVE ,
TFT_CYAN, TFT_YELLOW, TFT_WHITE, TFT_MAGENTA, TFT_BROWN, TFT_BLACK, TFT_SILVER};

void drawUI()
{
   // Draw color palette
   for (int i = 0; i < color_num; i++) {
     tft.fillRect(i * ColorPalette, 0, ColorPalette, ColorPalette, colors[i]);
   }
   // Draw clear screen button
   tft.setCursor(405 , 8, 2);
   tft.setTextColor(TFT_WHITE);
   tft.print("Clear");

   // Display current color
   tft.fillRect(450, 9, 12, 12, color);

   // Add palette border/outline
   tft.drawRect(0, 0, SCREEN_HEIGHT, ColorPalette, TFT_WHITE);
}
```

```
43
44  void setup() {
45    Serial.begin(115200);
46    tft.init();
47    uint16_t calData[5] = { 303, 3458, 350, 3304, 7 };
48    tft.setTouch(calData);
49    tft.setRotation(1);
50    tft.fillScreen(TFT_BLACK);
51
52    drawUI();
53  }
54
55  void loop() {
56    bool pressed = tft.getTouch(&x, &y);
57
58    if (pressed) {
59      Serial.printf("x: %d, y: %d\r\n", x, y);
60
61      /* Color palette area logic */
62      if(y < ColorPalette + 3)
63      {
64        // Enter color selection area, stroke interrupted
65        last_x = 0;
66        last_y = 0;
67
68        /* Determine stylus click area */
69        // Clicked Clear button
70        if(x >= ColorPalette * color_num)
71        {
72          tft.fillRect(0, ColorPalette, SCREEN_HEIGHT, SCREEN_WIDTH - ColorPalette, color);
73        }
74        // Clicked color palette
75        else{
76          // Calculate current color based on x-coordinate
77          color = colors[x / ColorPalette];
78
79          // Update display of current color
80          tft.fillRect(450, 9, 12, 12, color);
81        }
82        delay(5);
83        return;
84      }
85
86      /* Drawing logic */
```

```
87        // Start of a new stroke
88        if(last_x == 0 && last_y == 0)
89        {
90          // Record current position
91          last_x = x;
92          last_y = y;
93        }
94
95        // Draw a line between the previous position and current position
96        tft.drawWideLine(last_x, last_y, x, y, LineSize, color, color);
97
98        // Update coordinates
99        last_x = x;
100       last_y = y;
101     }
102     else{
103       // Reset coordinates when stylus leaves the screen
104       last_x = 0;
105       last_y = 0;
106     }
107 }
```

Code Explanation

Include the necessary header file.

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

Define screen resolution.

```
9   #define SCREEN_WIDTH 320
10  #define SCREEN_HEIGHT 480
```

Create TFT object instance.

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

Apply the calibration data.

```
34  uint16_t calData[5] = { 303, 3458, 350, 3304, 7 };
35  tft.setTouch(calData);
```

Draw color palette.

```
28  // Draw color palette
29  for (int i = 0; i < color_num; i++) {
30     tft.fillRect(i * ColorPalette, 0, ColorPalette, ColorPalette, colors[i]);
31  }
```

Detect whether the touch screen is pressed and store the x and y coordinates of the touch point in variables.

```
58  bool pressed = tft.getTouch(&x, &y);
```

When the Clear button is pressed, fill the screen with the current color.

```
71  // Clicked Clear button
72  if(x >= ColorPalette * color_num)
```

```
{
  tft.fillRect(0, ColorPalette, SCREEN_HEIGHT, SCREEN_WIDTH - ColorPalette, color);
}
```

Calculate the selected color based on the coordinates of the touch point

```
// Clicked color palette
else{
  // Calculate current color based on x-coordinate
  color = colors[x / ColorPalette];

  // Update display of current color
  tft.fillRect(450, 9, 12, 12, color);
}
```

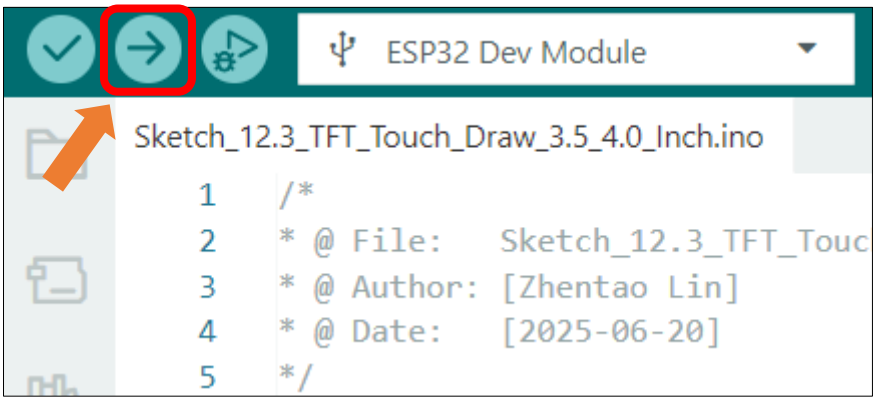
Draw a line based on coordinates

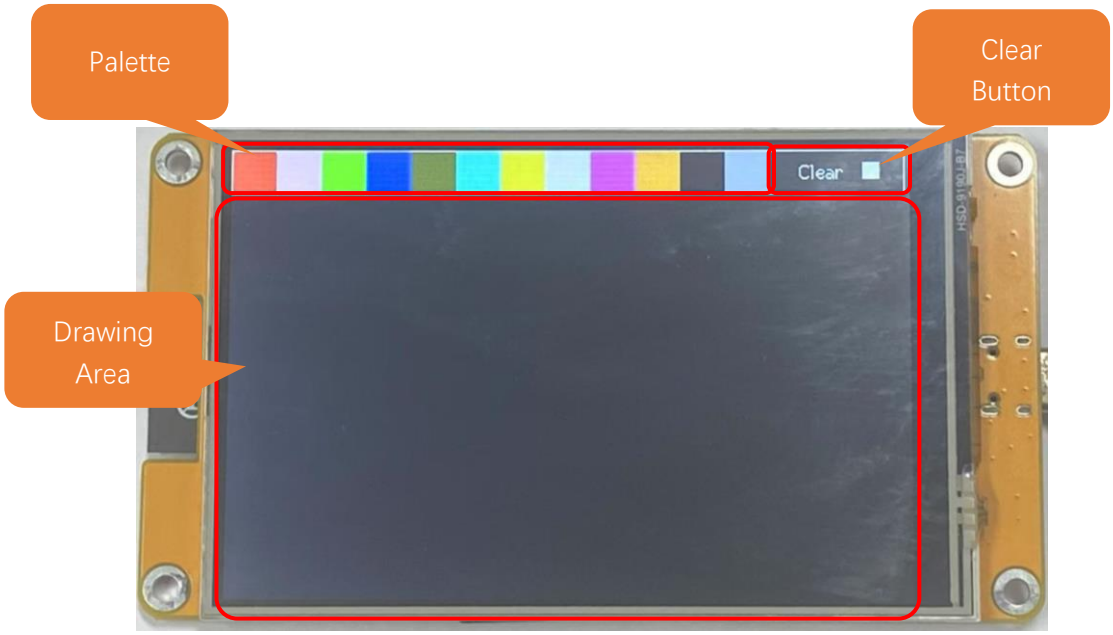
```
/* Drawing logic */
// Start of a new stroke
if(last_x == 0 && last_y == 0)
{
  // Record current position
  last_x = x;
  last_y = y;
}

// Draw a line between the previous position and current position
tft.drawWideLine(last_x, last_y, x, y, LineSize, color, color);

// Update coordinates
last_x = x;
last_y = y;
```

Add the calibration values obtained from the previous section to the code.

```
void setup() {
  Serial.begin(115200);
  tft.init();
  uint16_t calData[5] = { 303, 3458, 350, 3304, 7 };
  tft.setTouch(calData);
  tft.setRotation(1);
  tft.fillScreen(TFT_BLACK);

  drawUI();
}
```

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





Reference

```
void fillRect(int16_t x, int16_t y, int16_t w, int16_t h, uint16_t color);
```

This function is used to draw a solid rectangle

Parameters:

x: x coordinate of the upper left corner of the solid rectangle

y: y coordinate of the upper left corner of the solid rectangle

w: width of the solid rectangle

h: height of the solid rectangle

color: fill color of the solid rectangle

```
void drawWideLine(int16_t x0, int16_t y0, int16_t x1, int16_t y1, uint16_t width, uint16_t color1, uint16_t color2);
```

This function is used to draw lines

Parameters:

x0: x coordinate of the starting point of the line

y0: y coordinate of the starting point of the line

x1: x coordinate of the end point of the line

y1: y coordinate of the end point of the line

width: width of the line

color1: line color

color2: anti-aliasing transition color

Tips: Aliasing & Anti-aliasing

In computer graphics, **aliasing** refers to the **jagged or stair-step appearance** of lines and curves in digital images, particularly noticeable on diagonal lines and edges. This occurs due to the discrete nature of pixel grids - screens compose images from tiny square pixels that cannot perfectly represent continuous geometry. **Anti-aliasing** mitigates these artifacts through technical means to **smooth edges**, achieving more natural-looking graphics. The core technique blends transitional colors at boundary pixels, simulating human visual perception of soft edges.

Chapter 13 LVGL

Project 13.1 LVGL

Component List

Freenove ESP32 Display x 1 	USB cable x1 

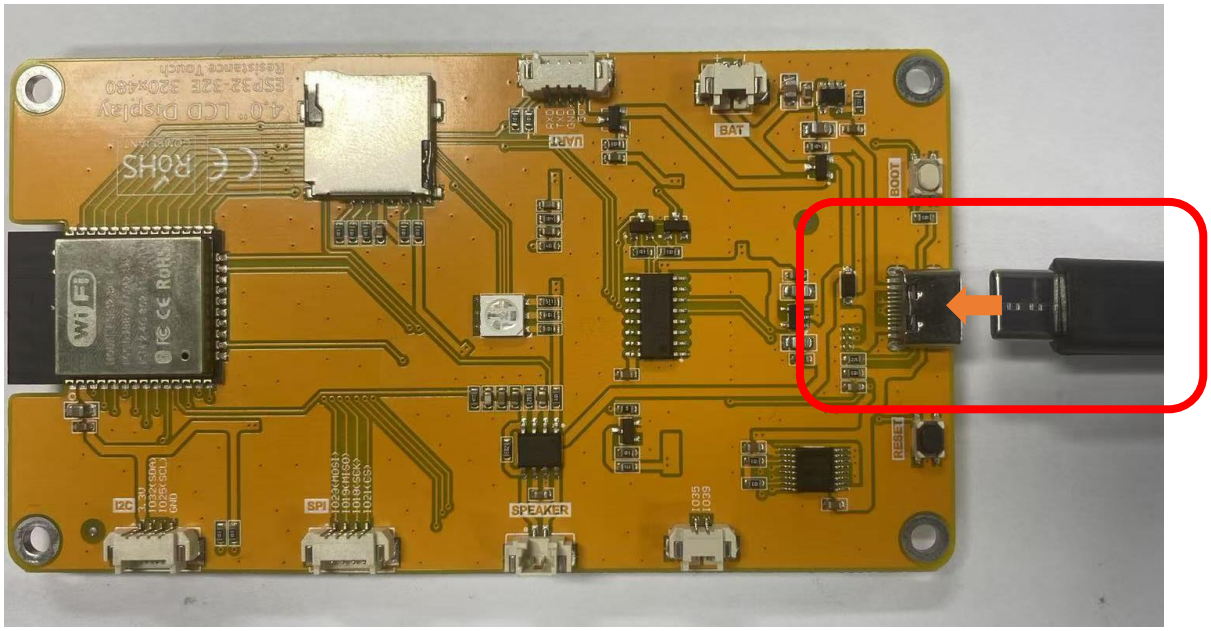
Component Knowledge

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

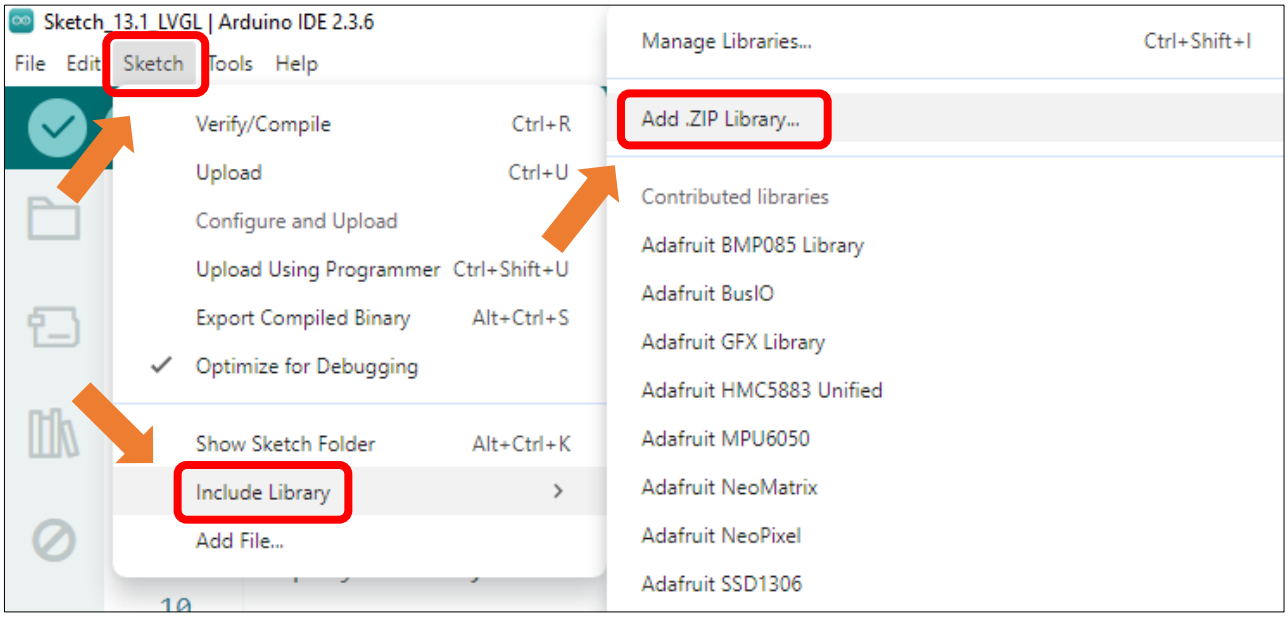
Connect Freenove ESP32-S3 to the computer using the USB cable.



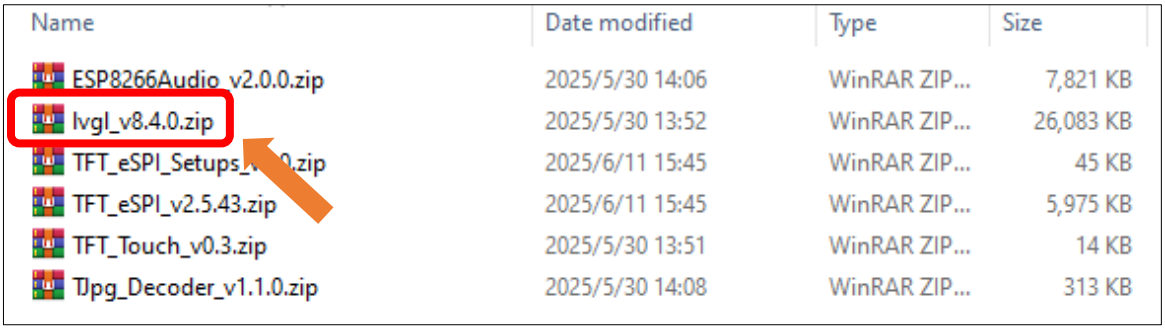
Sketch

Install Libraries

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



Install **lvgl_v8.4.0.zip**



Open"**Sketch_13.1_LVGL**" folder under **"Freenove_ESP32_Display\Sketch"** and double-click "**Sketch_13.1_LVGL.ino**".

Sketch_13.1_LVGL

The following is the program code:

```
/*
* @ File:   Sketch_13.1_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.

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

Set the baud rate to 115200

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

Initialize the screen.

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

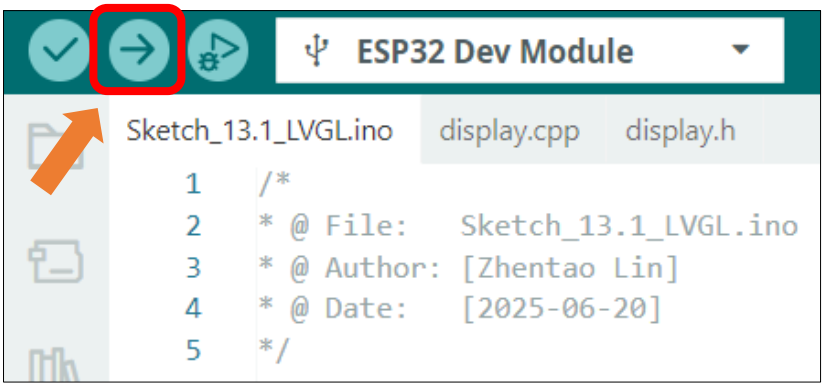
Configure the screen interface.

```
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 );
```

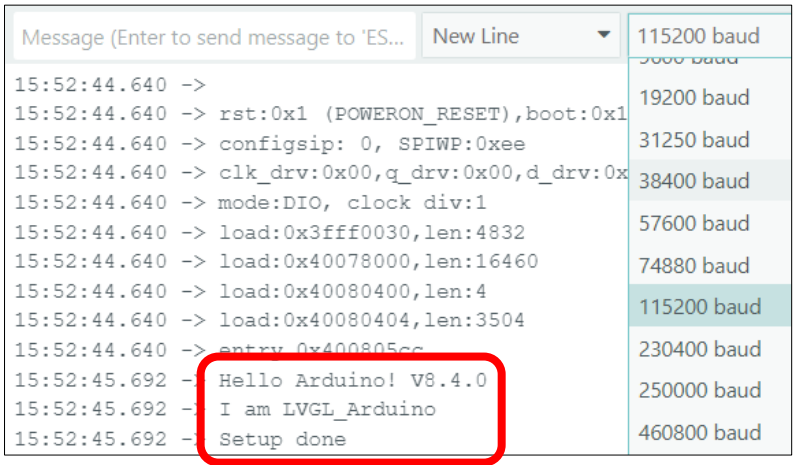
Let the LVGL handle the tasks.

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

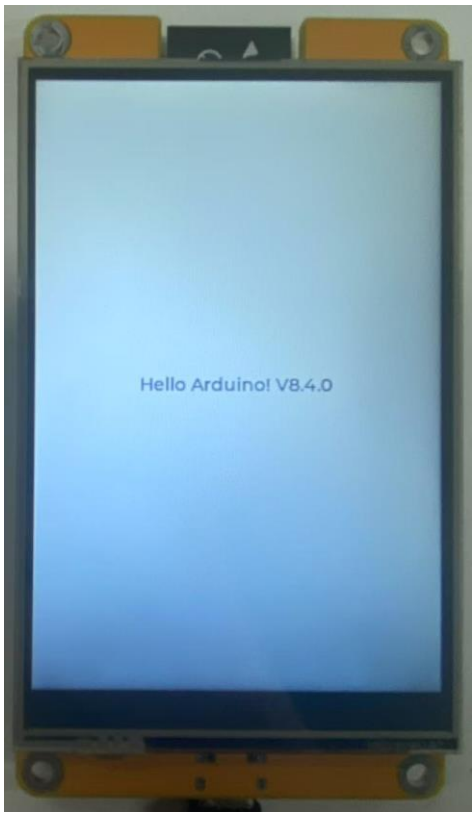
Click "**Upload**" to upload the code to Freenove ESP32 Display. Set the baud rate to 115200.



Data will be printed on the serial monitor.



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



Chapter 14 LVGL Picture

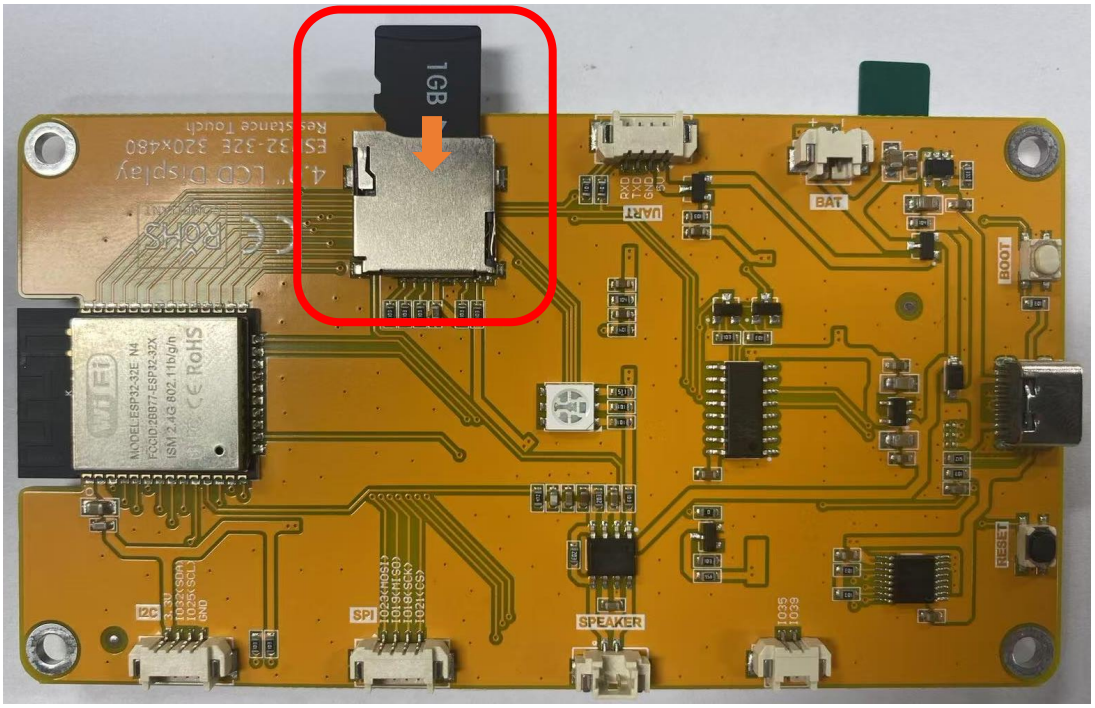
Project 14.1 LVGL Picture

Component List

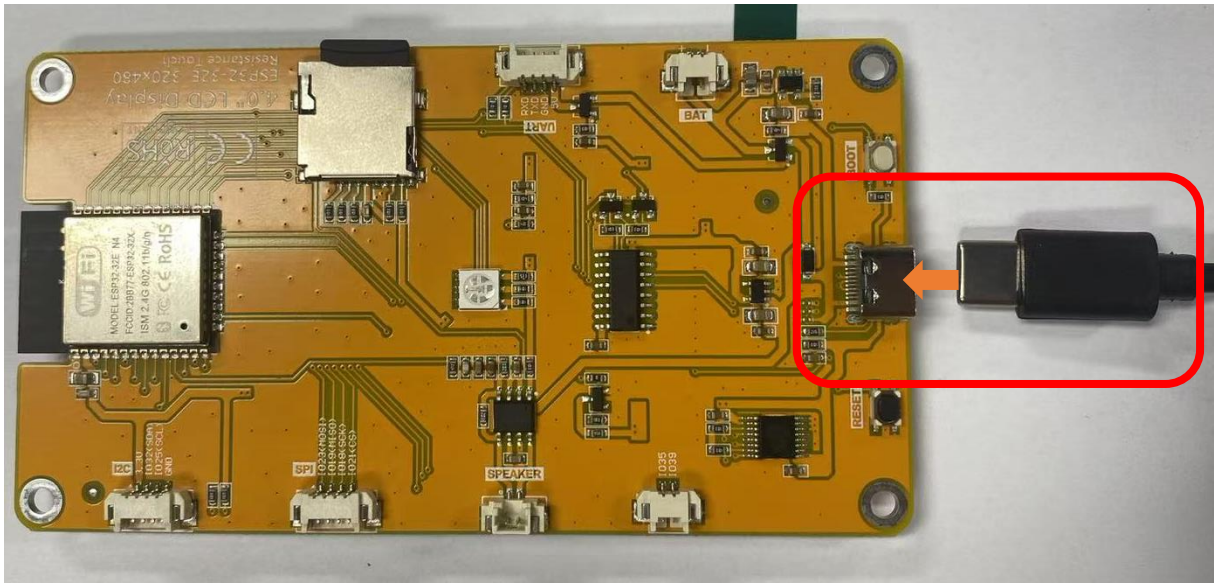
Freenove ESP32 Display x 1 	USB cable x1 
Stylus x 1 	

Circuit

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



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



Sketch

Open“**Sketch_14.1_Lvgl_Picture**” folder under **“Freenove_ESP32_Display\Sketch”** and double-click “**Sketch_14.1_Lvgl_Picture.ino**”.

Sketch_14.1_Lvgl_Picture

The following is the program code:

```
1  /*
```

```
2  * @ File:   Sketch_14.1_LVGL_Picture.ino
3  * @ Author: [Zhentao Lin]
4  * @ Date:   [2025-06-23]
5  */
6
7  #include "display.h"
8  #include "driver_sdspi.h"
9  #include "picture_ui.h"
10
11 Display screen;
12
13 #define SD_SCK   18
14 #define SD_MISO  19
15 #define SD_MOSI  23
16 #define SD_CS    5
17
18 void setup(){
19     /* prepare for possible serial debug */
20     Serial.begin( 115200 );
21
22    /*** Init drivers ***/
23     sdspi_init(SD_SCK, SD_MISO, SD_MOSI, SD_CS);       //Initialize the SD module
24     screen.init();
25
26     String LVGL_Arduino = "Hello Arduino! ";
27     LVGL_Arduino += String('V') + lv_version_major() + "." + lv_version_minor() + "." +
28 lv_version_patch();
29     Serial.println( LVGL_Arduino );
30     Serial.println( "I am LVGL_Arduino" );
31
32     setup_scr_picture(&guider_picture_ui);
33     lv_scr_load(guider_picture_ui.picture);
34
35     Serial.println( "Setup done" );
36 }
37
38 void loop(){
39     screen.routine(); /* let the GUI do its work */
40     delay( 5 );
41 }
```

Code Explanation

Include the header files.

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

```
9  #include "picture_ui.h"
```

Define the pins.

```
13 #define SD_SCK   18
14 #define SD_MISO  19
15 #define SD_MOSI  23
16 #define SD_CS    5
```

Set the baud rate to 115200

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

Initialize configuration.

```
23 sdspi_init(SD_SCK, SD_MISO, SD_MOSI, SD_CS);       //Initialize the SD module
24 screen.init();
```

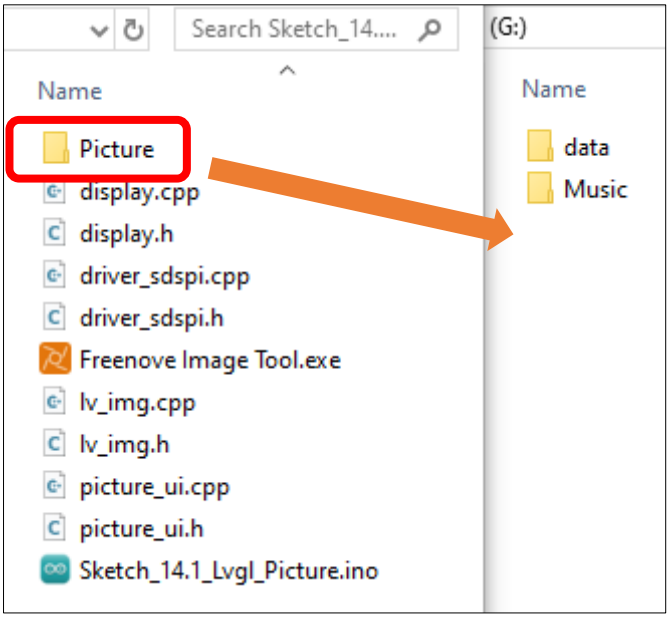
Create and load the interface.

```
32 setup_scr_picture(&guider_picture_ui);
33 lv_scr_load(guider_picture_ui.picture);
```

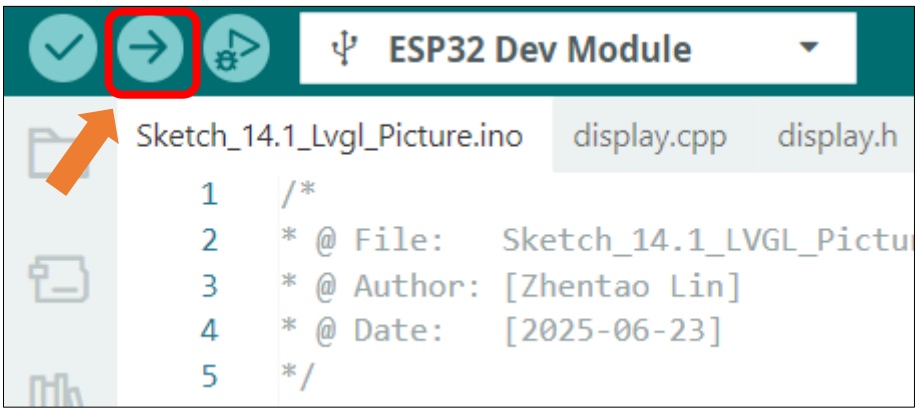
LVGL Task Processor

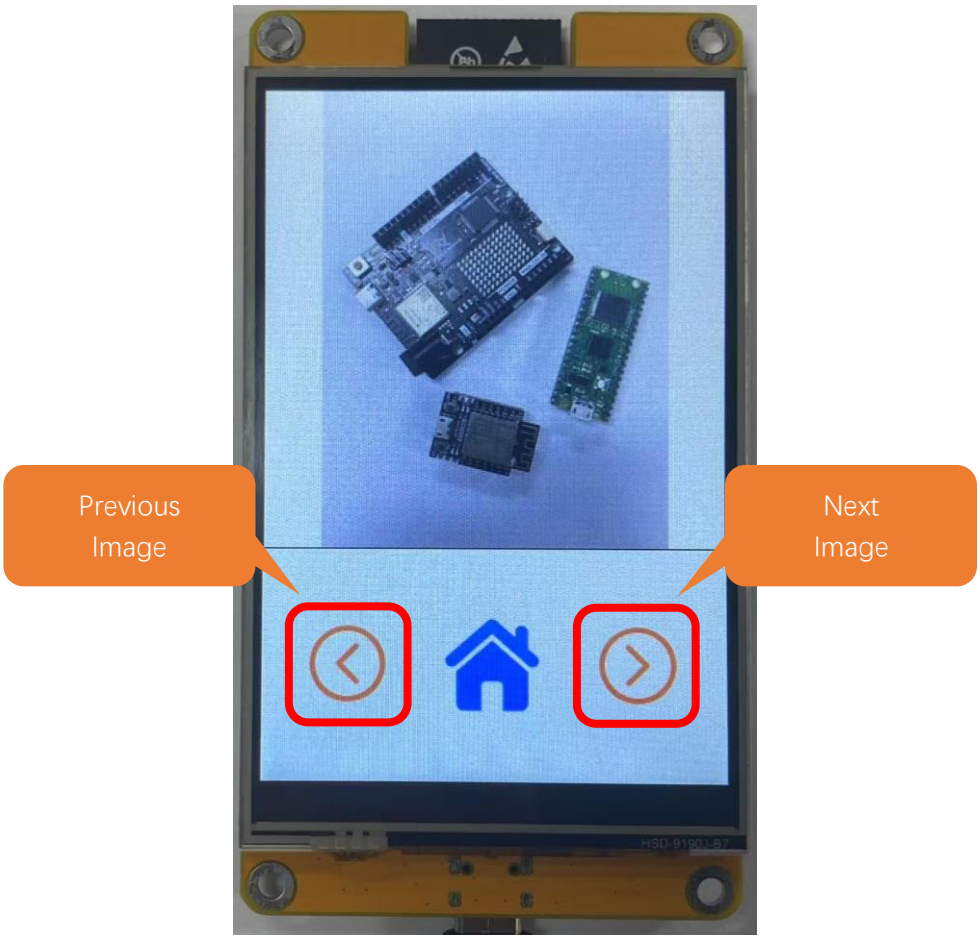
```
39 screen.routine(); /* let the GUI do its work */
```

Copy the Picture folder under the **Freenove_ESP32_Display\Sketch\Sketch_14.1_Lvgl_Picture** directory to the SD card root directory.



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

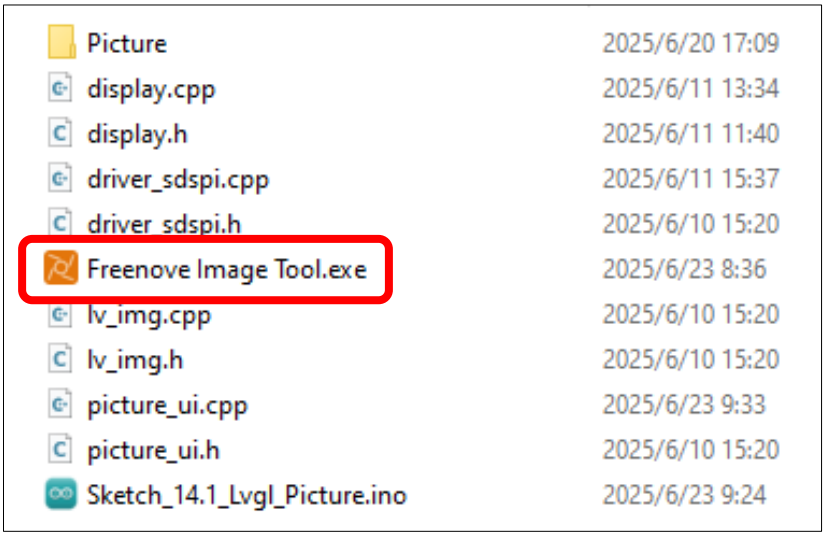




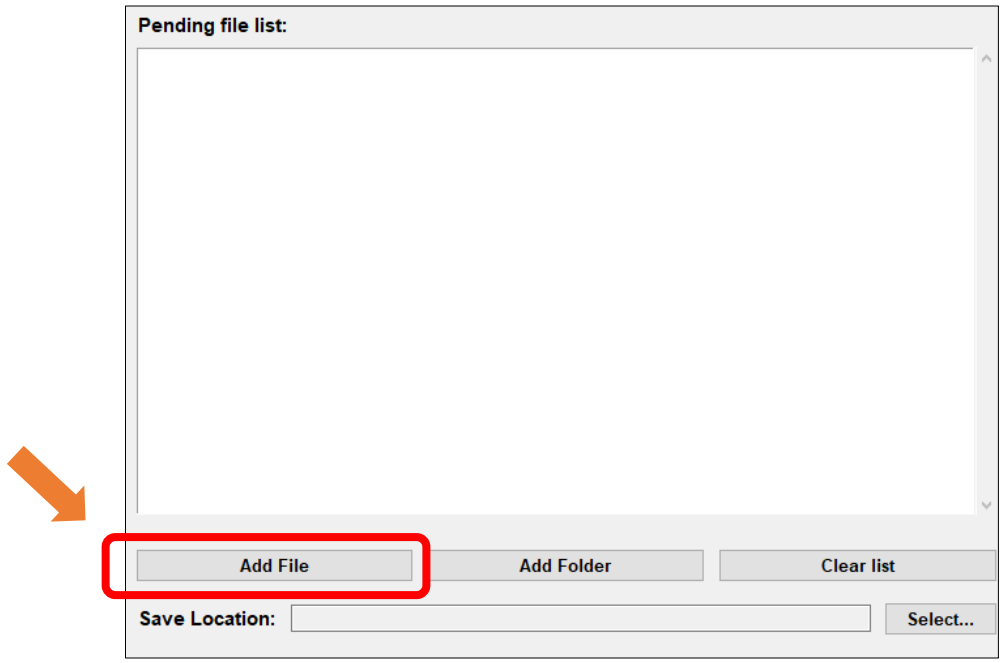
Custom image display

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

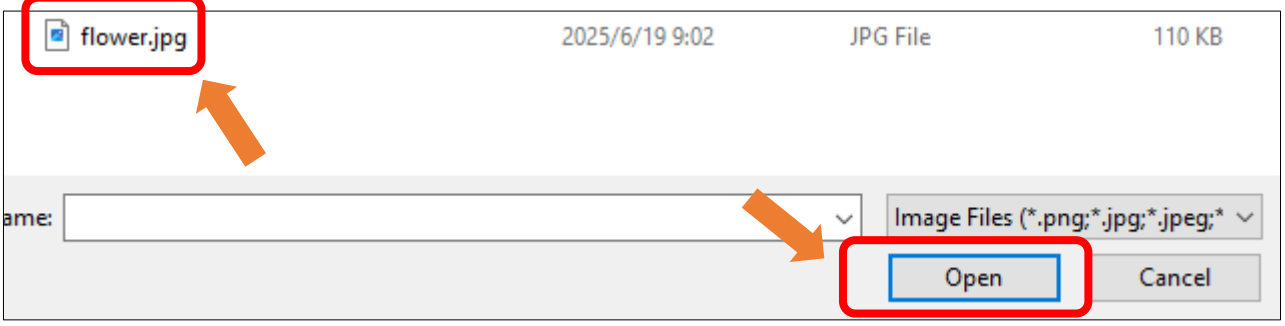
First, open **Freenove_ESP32_Display\Sketch\Sketch_14.1_Lvgl_Picture\Freenove Image Tool.exe**



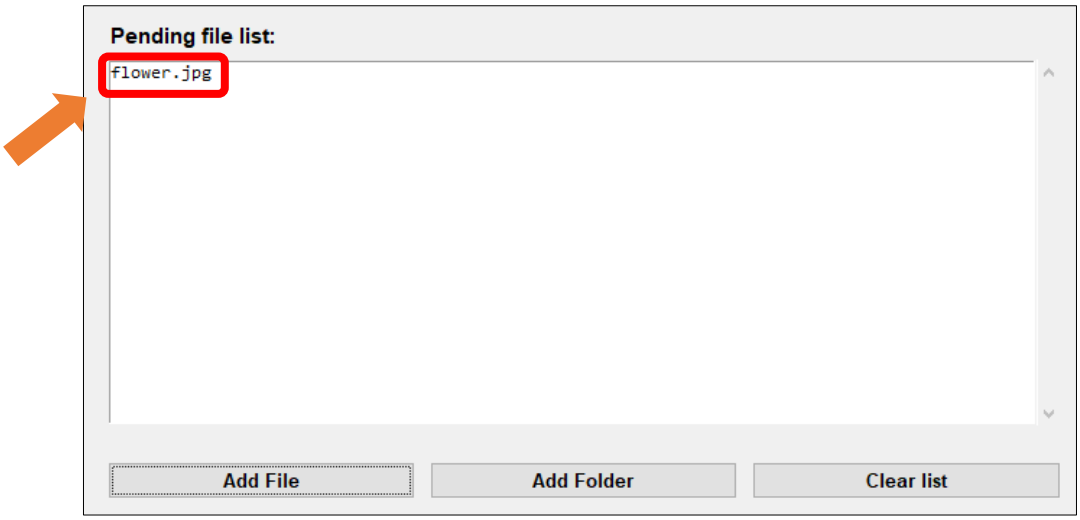
Click **"Add File"**



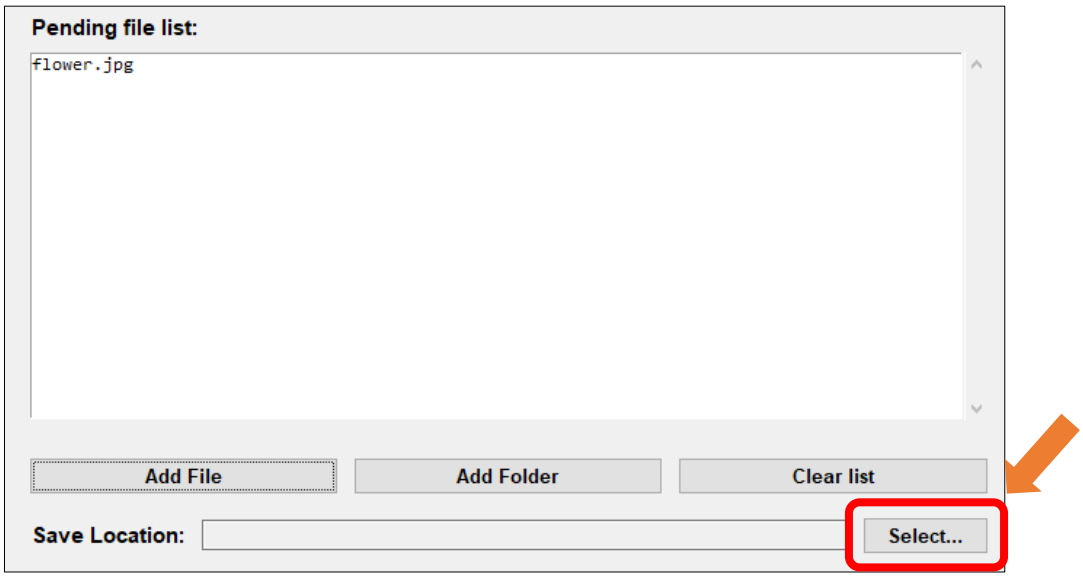
Select any image you like.



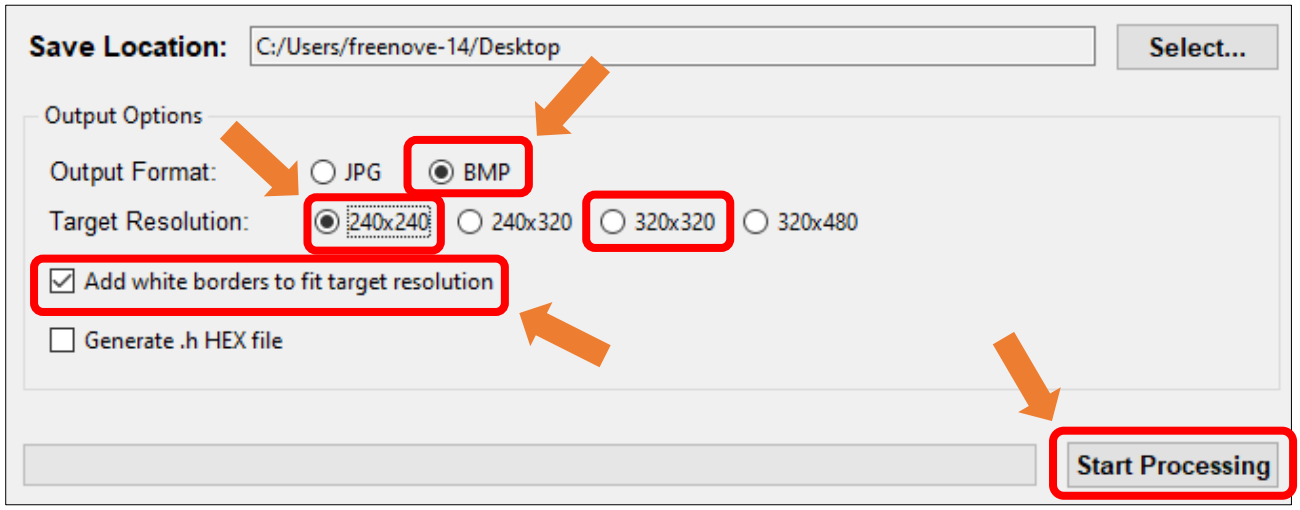
The image files from your folder will now appear in the **Pending File List**.



Click "Select…" to choose the image saving location.

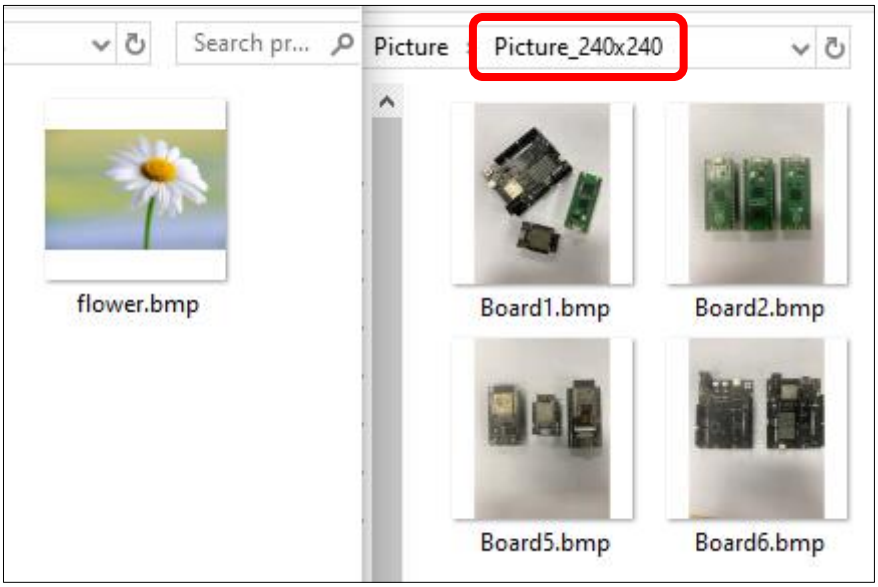


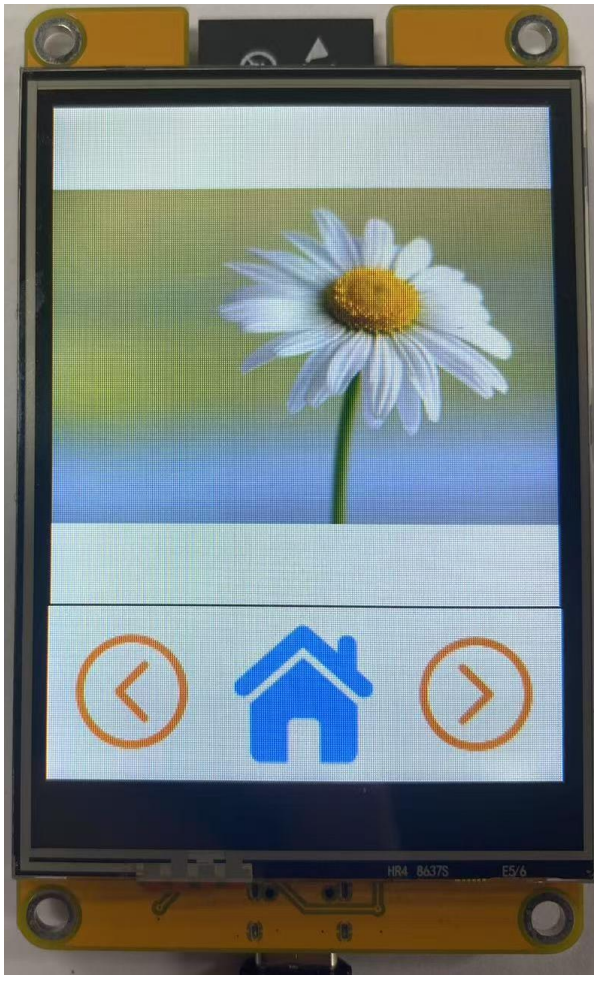
Select **BMP** for the format, set the resolution to **240x240** or **320x320** based on your screen, and then click **Start Processing.**



Wait for the progress bar to complete.

Copy or replace the generated image to the folder corresponding to the Picture folder in the root directory of the SD card.
If the screen resolution is **240x320**, put the image in **Picture_240x240**
If the screen resolution is **320x480**, put the image in **Picture_320x320**





Chapter 15 LVGL Timer

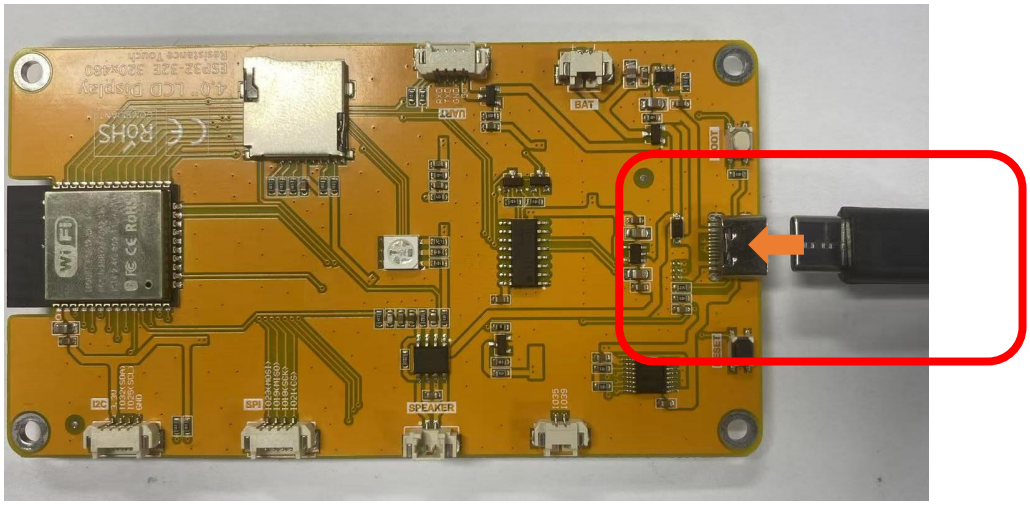
Project 15.1 LVGL Timer

Component List

Freenove ESP32 Display x 1 	USB cable x1 
Stylus x 1 	

Circuit

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



Sketch

Open“**Sketch_15.1_Lvgl_Timer**” folder under **“Freenove_ESP32_Display\Sketch”** and double-click “**Sketch_15.1_Lvgl_Timer.ino**”.

Sketch_15.1_Lvgl_Timer

The following is the program code:

```
/*
* @ File:   Sketch_15.1_LVGL_Timer.ino
* @ Author: [Zhentao Lin]
* @ Date:   [2025-06-23]
*/

#include "display.h"
#include "chronograph_ui.h"

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

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

    /***  Print lvgl version ***/
    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");
    Serial.println("Setup done");

    /*** The custom code ***/
    setup_scr_chronograph(&guider_chronograph_ui);
    lv_scr_load(guider_chronograph_ui.chronograph);
}

void loop() {
    screen.routine();
    delay(5);
}
```

Code Explanation

Include the header files.

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

```
8  #include "chronograph_ui.h"
```

Set the baud rate to 115200.

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

Initialize configuration.

```
23  screen.init();
```

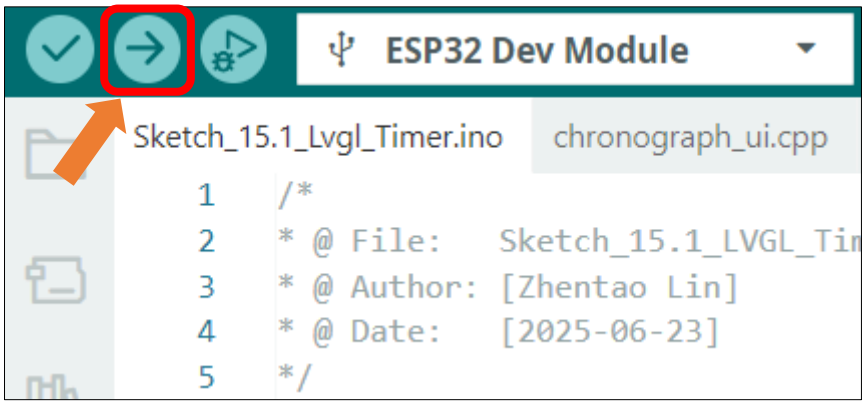
Create and load the interface.

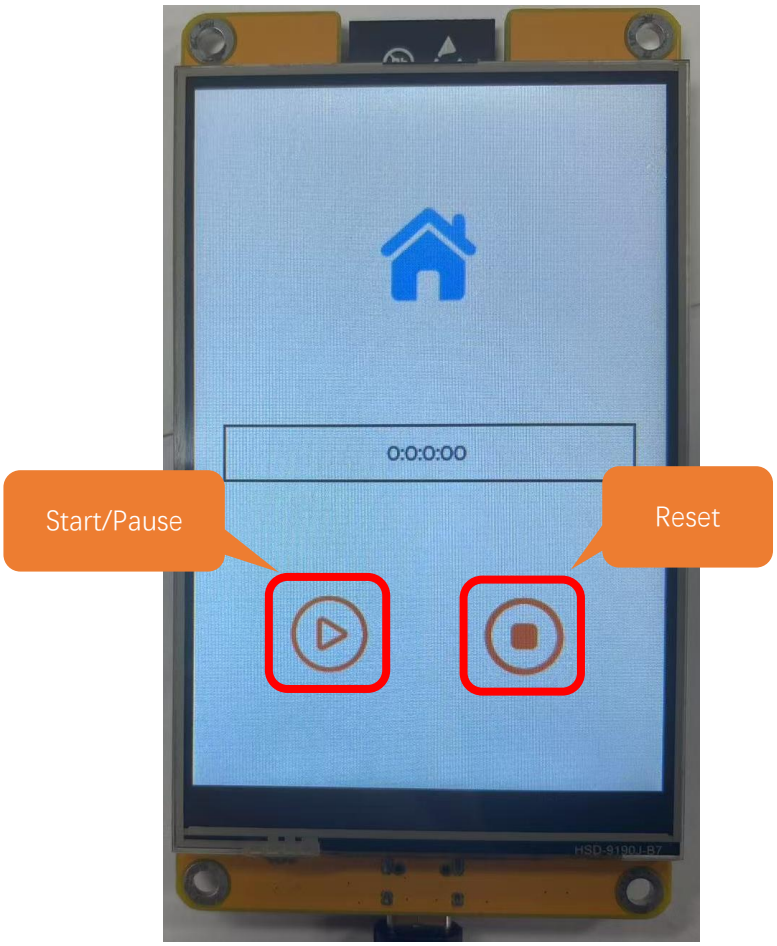
```
32  setup_scr_chronograph(&guider_chronograph_ui);
33  lv_scr_load(guider_chronograph_ui.chronograph);
```

LVGL task processor.

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

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





Chapter 16 Lvgl RGB

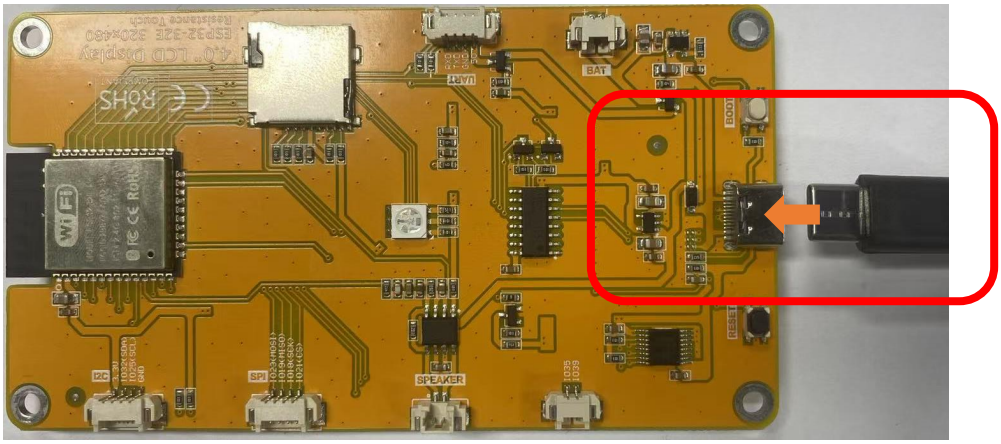
Project 16.1 LVGL RGB

Component List

Freenove ESP32 Display x 1 	USB cable x1 
Stylus x 1 	

Circuit

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



Sketch

Open“**Sketch_16.1_Lvgl_RGB**” folder under **“Freenove_ESP32_Display\Sketch”** and double-click “**Sketch_16.1_Lvgl_RGB.ino**”.

Sketch_16.1_Lvgl_RGB

The following is the program code:

```
/*
* @ File:   Sketch_16.1_LVGL_RGB.ino
* @ Author: [Zhentao Lin]
* @ Date:   [2025-06-23]
*/

#include "display.h"
#include "rgb_ui.h"

#define RED_PIN 22
#define GREEN_PIN 16
#define BLUE_PIN 17

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

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

    /***  Print lvgl version ***/
    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");
    Serial.println("Setup done");

    /*** The custom code ***/
    rgb_init(RED_PIN, GREEN_PIN, BLUE_PIN);
    rgb_set_color(0, 0, 0);
    setup_scr_rgb(&guider_rgb_ui);
    lv_scr_load(guider_rgb_ui.rgb);
}

void loop() {
    screen.routine();
```

```
38      delay(5);
39  }
```

Code Explanation

Include the header files.

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

Define the pins.

```
13  #define RED_PIN 22
14  #define GREEN_PIN 16
15  #define BLUE_PIN 17
```

Set the baud rate to 115200

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

Initialize the RGB LED.

```
23  rgb_init(RED_PIN, GREEN_PIN, BLUE_PIN);
24  rgb_set_color(0, 0, 0);
```

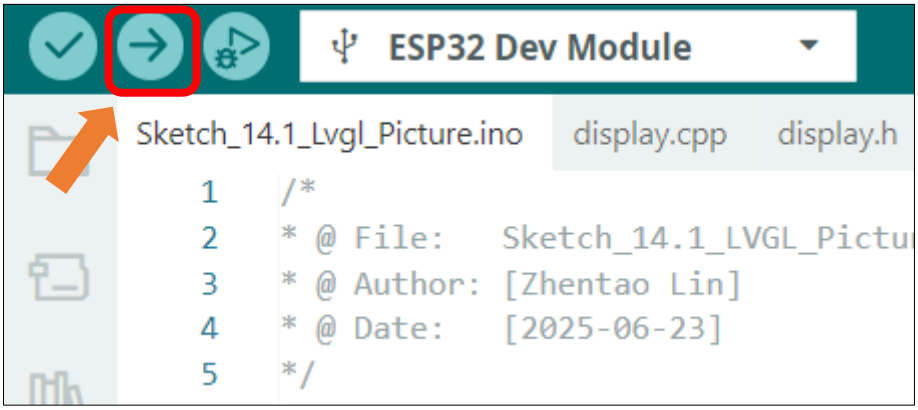
Create and load the interface.

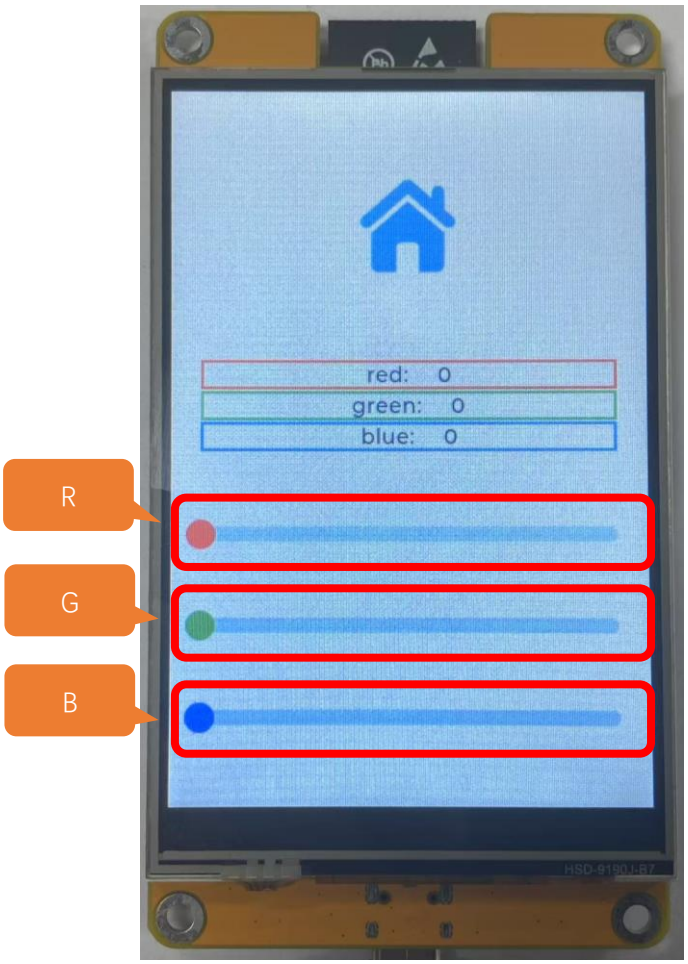
```
32  setup_scr_rgb(&guider_rgb_ui);
33  lv_scr_load(guider_rgb_ui.rgb);
```

LVGL task processor.

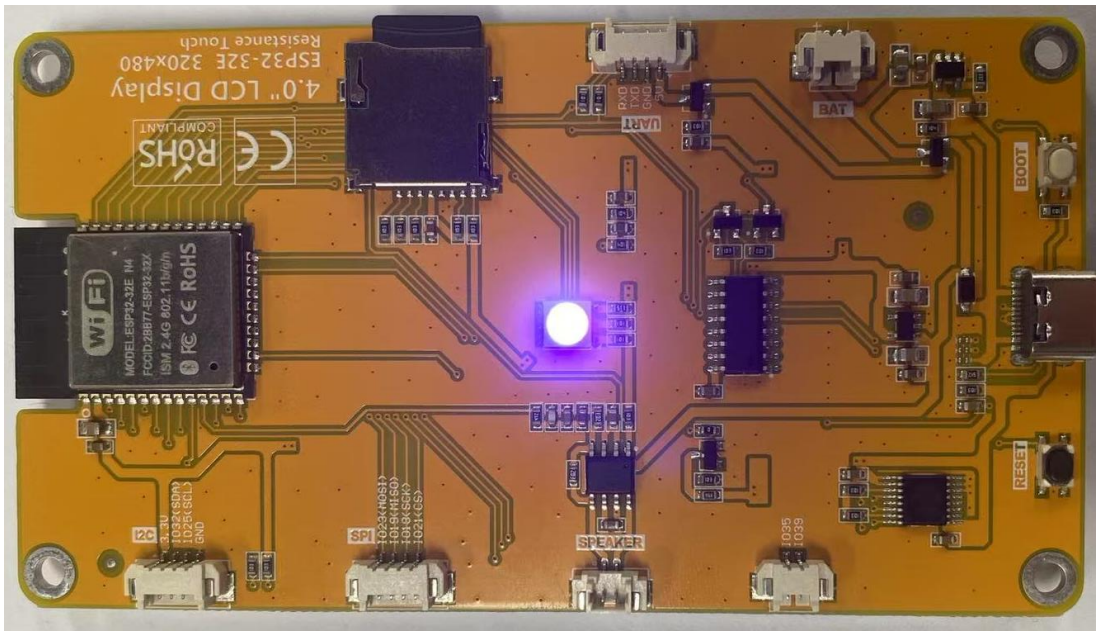
```
39  screen.routine(); /* let the GUI do its work */
```

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





Drag to adjust the value of the red (R), green (G), and blue (B) color, and you will see the color of the LED change.



Chapter 17 LVGL Music

Project 17.1 LVGL Music

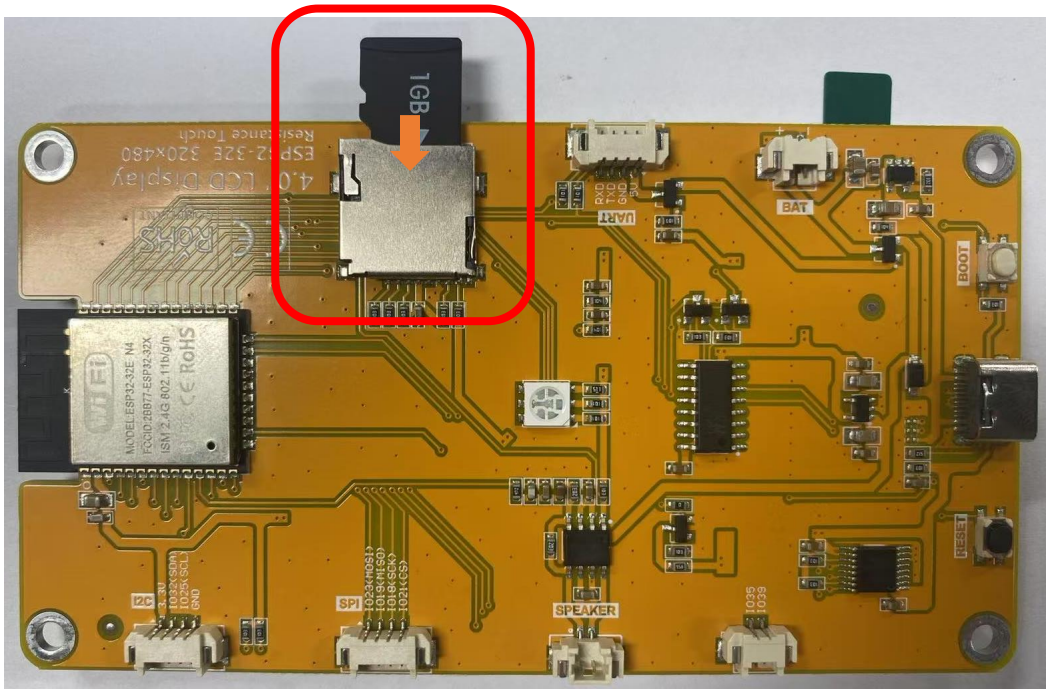
Component List

Freenove ESP32 Display x 1 	USB cable x1 
Stylus x 1 	

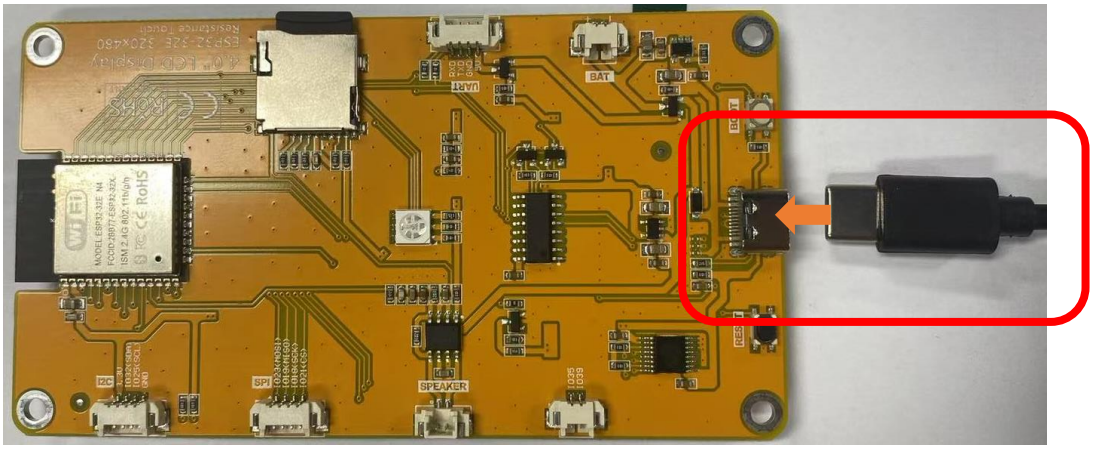
Note: This kit does not include speaker, SD card, or SD card reader. Please buy them yourself!

Circuit

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



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



Sketch

Open“**Sketch_17.1_Lvgl_Music**” folder under **“Freenove_ESP32_Display\Sketch”** and double-click “**Sketch_17.1_Lvgl_Music.ino**”. **Freenove_ESP32_Display\Sketch\Sketch_17.1_Lvgl_Music**

Sketch_17.1_Lvgl_Music

The following is the program code:

```
1  /*
2  * @ File:   Sketch_17.1_Lvgl_Music.ino
3  * @ Author: [Zhentao Lin]
```

```
4   * @ Date:   [2025-06-23]
5   */
6
7   #include "display.h"
8   #include "driver_sdspi.h"
9   #include "music_ui.h"
10
11  Display screen;
12
13  #define SD_SCK   18
14  #define SD_MISO  19
15  #define SD_MOSI  23
16  #define SD_CS    5
17  #define AUDIO_EN 4
18
19  void setup(){
20      /* prepare for possible serial debug */
21      Serial.begin( 115200 );
22
23      pinMode(AUDIO_EN, OUTPUT);
24      digitalWrite(AUDIO_EN, LOW);
25
26      /*** Init drivers ***/
27      sdspi_init(SD_SCK, SD_MISO, SD_MOSI, SD_CS);       //Initialize the SD module
28      screen.init();
29
30      String LVGL_Arduino = "Hello Arduino! ";
31      LVGL_Arduino += String('V') + lv_version_major() + "." + lv_version_minor() + "." +
32  lv_version_patch();
33      Serial.println( LVGL_Arduino );
34      Serial.println( "I am LVGL_Arduino" );
35
36      setup_scr_music(&guider_music_ui);
37      lv_scr_load(guider_music_ui.music);
38
39      Serial.println( "Setup done" );
40  }
41
42  void loop(){
43      screen.routine(); /* let the GUI do its work */
44      delay( 5 );
45  }
```

Code Explanation

Include the header files.

```
7  #include "display.h"
8  #include "driver_sdspi.h"
9  #include "music_ui.h"
```

Define the pins.

```
13 #define SD_SCK   18
14 #define SD_MISO  19
15 #define SD_MOSI  23
16 #define SD_CS    5
17 #define AUDIO_EN 4
```

Set the baud rate to 115200

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

Initialize configuration.

```
23 screen.init();
```

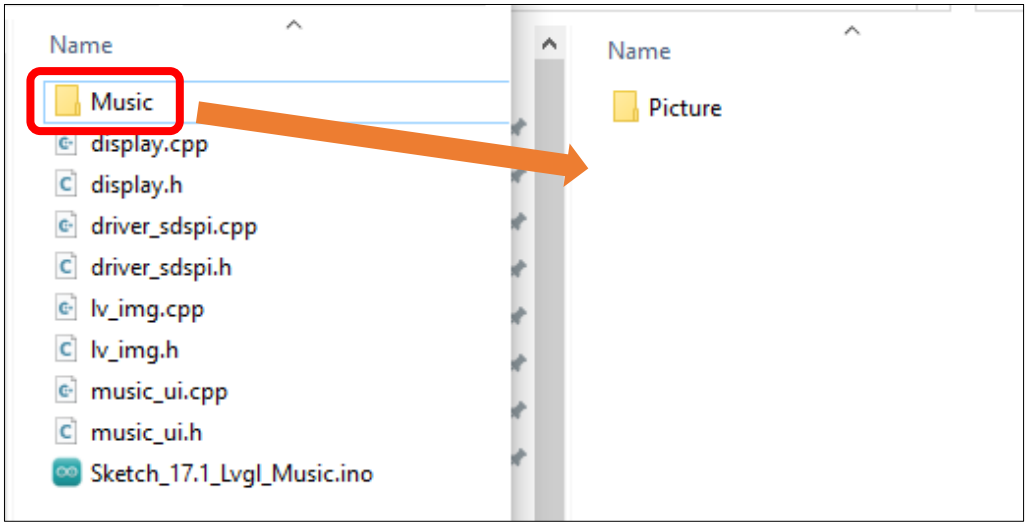
Create and load the interface.

```
32 setup_scr_music(&guider_music_ui);
33 lv_scr_load(guider_music_ui.music);
```

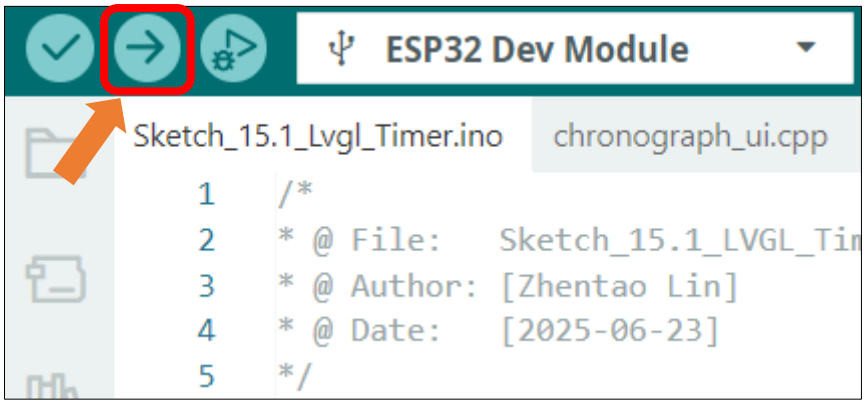
LVGL task processor.

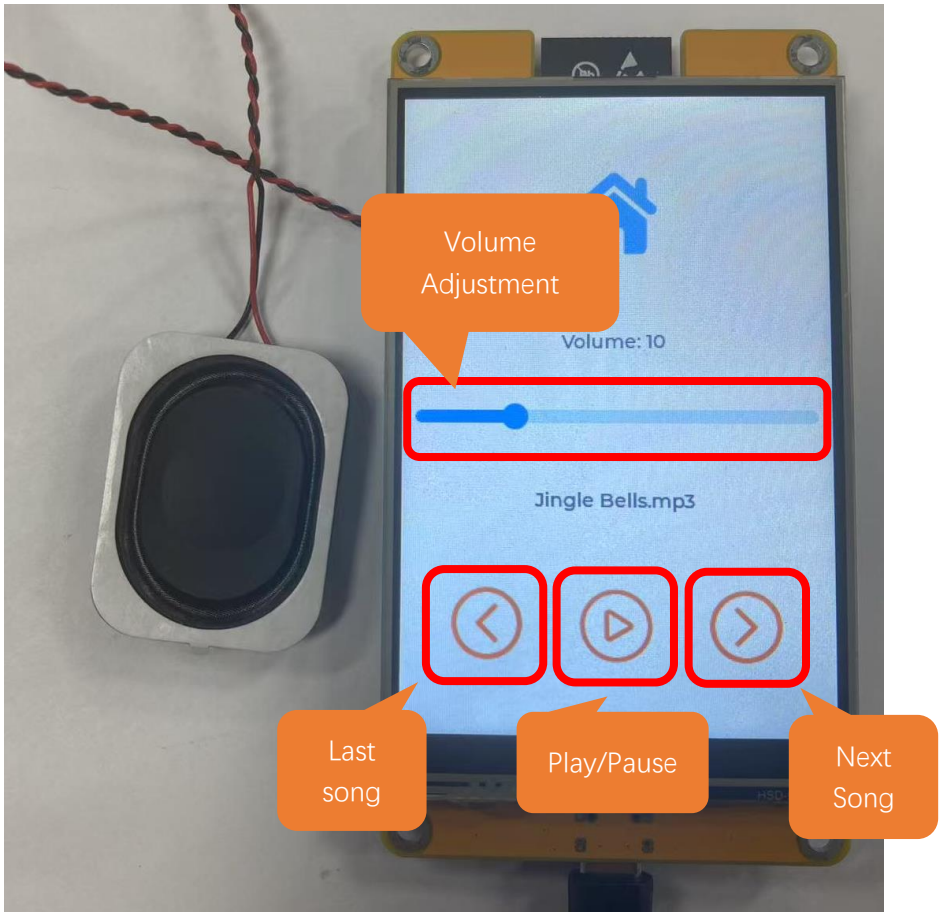
```
39 screen.routine(); /* let the GUI do its work */
```

Insert the SD card to the card reader and plug them to the computer. Copy the **Music** folder under the **Freenove_ESP32_Display\Sketch\Sketch_17.1_Lvgl_Music** directory to the root directory of the SD card.



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





Note: If the screen flickers during playback, it may be due to insufficient power supply. You can try powering with a battery.

Chapter 18 LVGL Multifunctionality

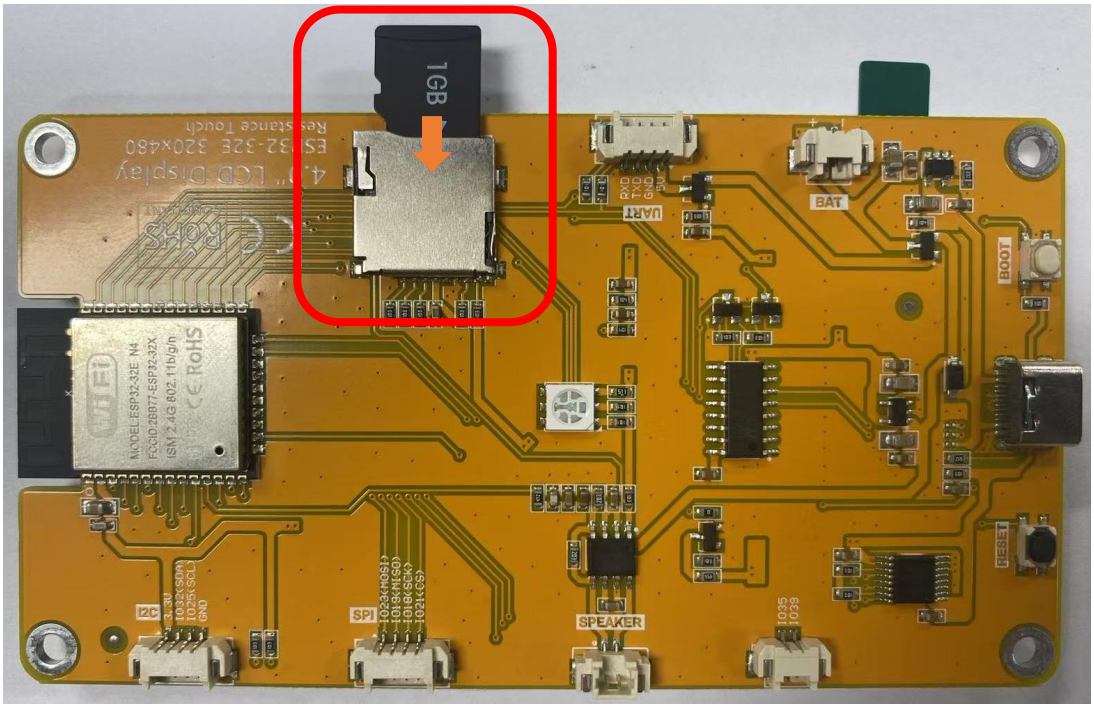
Project 18.1 LVGL Multifunctionality

Component List

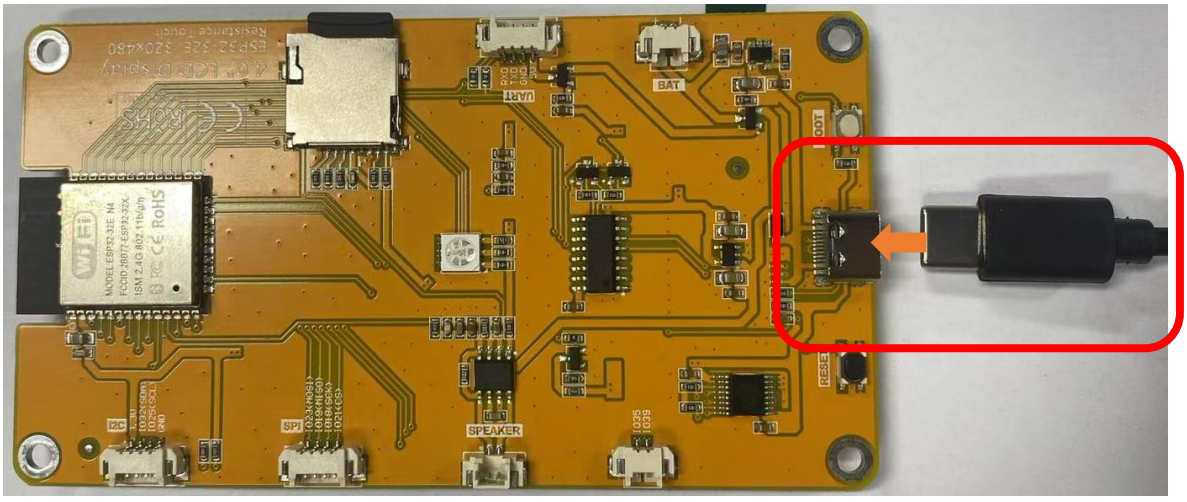
Freenove ESP32 Display x 1 	USB cable x1 
Stylus x 1 	

Circuit

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



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



Sketch

Open“**Sketch_18.1_Lvgl_Multifunctionality**” folder under **“Freenove_ESP32_Display\Sketch”** and double-click “**Sketch_18.1_Lvgl_ Multifunctionality.ino**”.

Sketch_18.1_Lvgl_Multifunctionality

The following is the program code:

```
1  /*
2  * @ File:    Sketch_18.1_LVGL_Multifunctionality.ino
```

```
* @ Author: [Zhentao Lin]
* @ Date:   [2025-06-23]
*/

#include "display.h"
#include "driver_sdspi.h"
#include "main_ui.h"

#define SD_SCK   18
#define SD_MISO  19
#define SD_MOSI  23
#define SD_CS    5
#define AUDIO_EN 4
#define RED_PIN 22
#define GREEN_PIN 16
#define BLUE_PIN 17

Display screen;  // Create an instance of the Display class

void setup() {
   /* Prepare for possible serial debug */
   Serial.begin(115200);
   while (!Serial) {
     delay(10);
   }
   /*** Init drivers ***/
   pinMode(AUDIO_EN, OUTPUT);
   digitalWrite(AUDIO_EN, LOW);
   rgb_init(RED_PIN, GREEN_PIN, BLUE_PIN);
   rgb_set_color(0, 0, 0);
   sdspi_init(SD_SCK, SD_MISO, SD_MOSI, SD_CS);       //Initialize the SD module
   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" );

   setup_scr_main(&guider_main_ui);
   lv_scr_load(guider_main_ui.main);

   // Print setup completion message to the serial monitor
   Serial.println("Setup done");
```

```
47  }
48
49  void loop() {
50      screen.routine(); /* Let the GUI do its work */  // Handle routine display tasks
51      delay(5);         // Add a small delay to prevent the loop from running too fast
52  }
53
```

Code Explanation

Include the header files.

```
7   #include "display.h"
8   #include "driver_sdspi.h"
9   #include "main_ui.h"
```

Define the pins.

```
11  #define SD_SCK   18
12  #define SD_MISO  19
13  #define SD_MOSI  23
14  #define SD_CS    5
15  #define AUDIO_EN 4
16  #define RED_PIN 22
17  #define GREEN_PIN 16
18  #define BLUE_PIN 17
```

Set the baud rate to 115200

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

Initialize RGB LED configuration

```
31  rgb_init(RED_PIN, GREEN_PIN, BLUE_PIN);
32  rgb_set_color(0, 0, 0);
```

Initialize SD card.

```
33  sdspi_init(SD_SCK, SD_MISO, SD_MOSI, SD_CS);       //Initialize the SD module
```

Initialize the screen.

```
34  screen.init();
```

Create and load the interface.

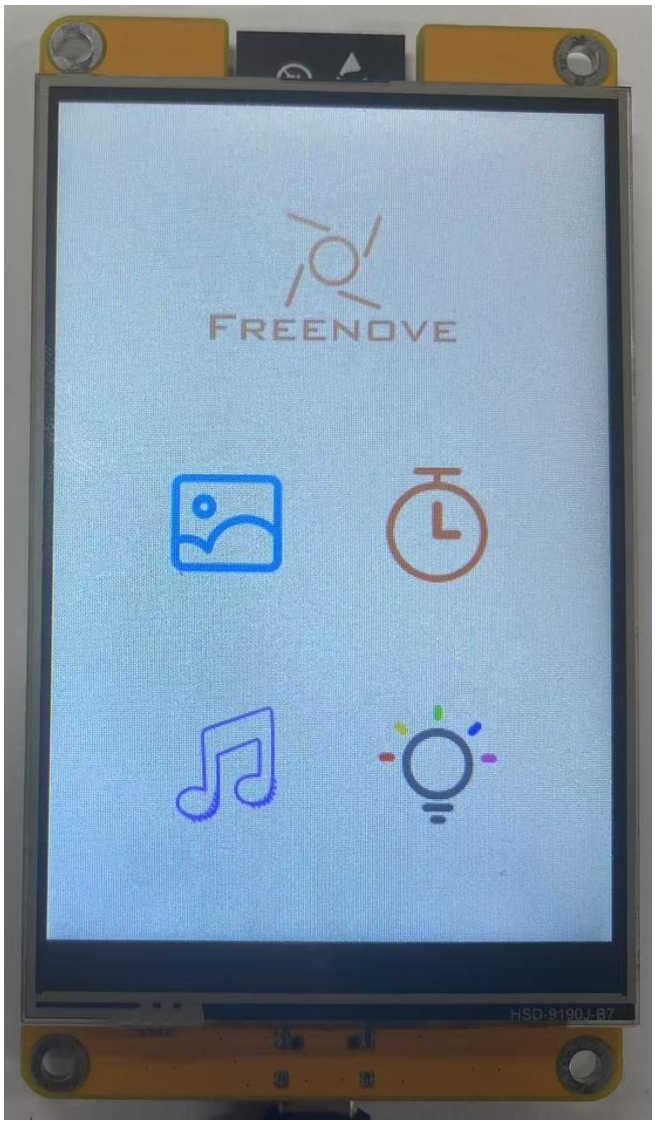
```
43  setup_scr_main(&guider_main_ui);
44  lv_scr_load(guider_main_ui.main);
```

LVGL task processor.

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

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


ESP32 Dev Module
Sketch_18.1_Lvgl_Multifunctionality.ino
chronograp
1 /*
2 * @ File: Sketch_18.1_LVGL_Mult
3 * @ Author: [Zhentao Lin]
4 * @ Date: [2025-06-23]
5 */


FREENOVE

Chapter 19 LVGL Arduino

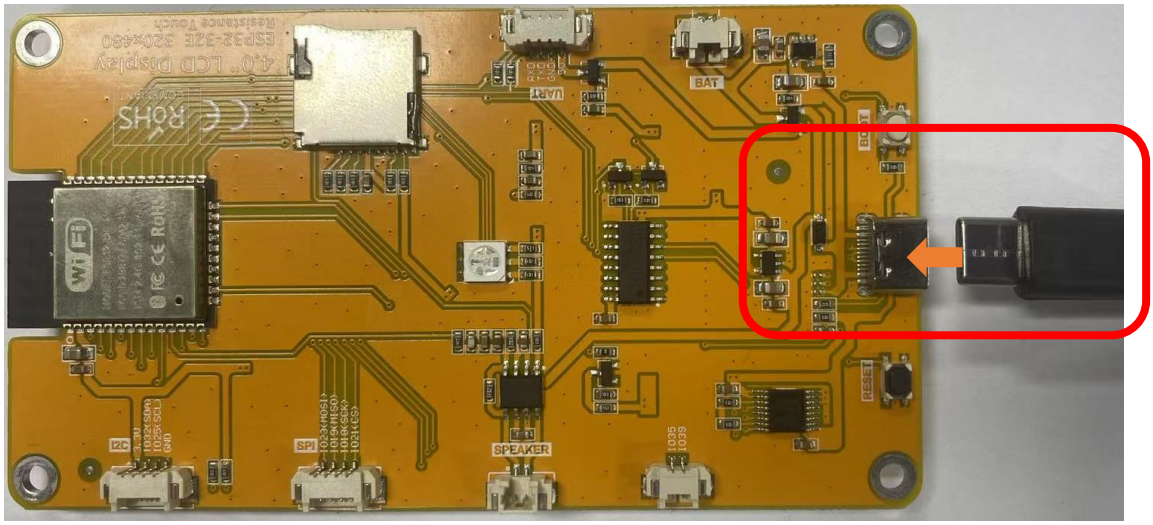
Project 19.1 LVGL Arduino

Component List

Freenove ESP32 Display x 1 	USB cable x1 
Stylus x 1 	

Circuit

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



Sketch

Open“**Sketch_19.1_Lvgl_Arduino**” folder under **“Freenove_ESP32_Display\Sketch”** and double-click “**Sketch_19.1_Lvgl_Arduino.ino**”.

Sketch_19.1_Lvgl_Arduino

The following is the program code:

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

#include <lvgl.h>
#include <TFT_eSPI.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)

#if defined(FNK0103B_2P8_240x320_ST7789) || defined(FNK0103F_2P8_240x320_ILI9341)
   #ifndef _TFT_Touch_H
     #include "TFT_Touch.h"
     #if !defined(_TFT_Touch_H)
       #error "Please install TFT_Touch library before using this library!"
     #endif
   #endif
#define DOUT 39 /* Data out pin (T_DO) of touch screen */
#define DIN 32  /* Data in pin (T_DIN) of touch screen */
#define DCS 33  /* Chip select pin (T_CS) of touch screen */
#define DCLK 25 /* Clock pin (T_CLK) of touch screen */
#define TFT_SCREEN_WIDTH 240
#define TFT_SCREEN_HEIGHT 320
TFT_Touch touch = TFT_Touch(DCS, DCLK, DIN, DOUT);
#elif defined(FNK0103L_3P2_240x320_ST7789)
#define TFT_SCREEN_WIDTH 240
#define TFT_SCREEN_HEIGHT 320
#elif defined(FNK0103N_3P5_320x480_ST7796) || defined(FNK0103S_4P0_320x480_ST7796)
#define TFT_SCREEN_WIDTH 320
```

```
#define TFT_SCREEN_HEIGHT 480
#endif

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 */

#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 );
}

/*Read the touchpad*/
void my_touchpad_read(lv_indev_drv_t *indev_driver, lv_indev_data_t *data) {
  uint16_t touchX, touchY;
  bool touched;
#if defined(FNK0103B_2P8_240x320_ST7789) || defined(FNK0103F_2P8_240x320_ILI9341)
  touched = touch.Pressed();
  touchX = touch.X();
  touchY = touch.Y();
  if (!touched) {
    data->state = LV_INDEV_STATE_REL;
```

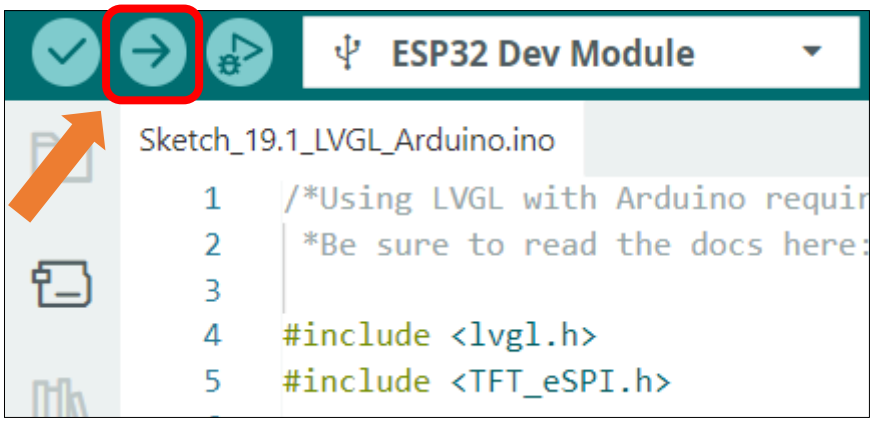
```
81      } else {
82        data->state = LV_INDEV_STATE_PR;
83  #if (TFT_DIRECTION == 0)
84        data->point.x = tft.width() - touchY;
85        data->point.y = touchX;
86  #elif (TFT_DIRECTION == 1)
87        data->point.x = touchX;
88        data->point.y = touchY;
89  #elif (TFT_DIRECTION == 2)
90        data->point.x = touchY;
91        data->point.y = tft.height() - touchX;
92  #elif (TFT_DIRECTION == 3)
93        data->point.x = tft.width() - touchX;
94        data->point.y = tft.height() - touchY;
95  #endif
96        // Serial.print("Data x ");
97        // Serial.print(data->point.x);
98        // Serial.print("\tData y ");
99        // Serial.println(data->point.y);
100     }
101 #elif defined(FNK0103L_3P2_240x320_ST7789)
102     touched = tft.getTouch(&touchX, &touchY, 600);
103     if (!touched) {
104       data->state = LV_INDEV_STATE_REL;
105     } else {
106       data->state = LV_INDEV_STATE_PR;
107 #if (TFT_DIRECTION == 0)
108       data->point.x = touchX;
109       data->point.y = touchY;
110 #elif (TFT_DIRECTION == 1)
111       data->point.x = map(touchY, 0, tft.height(), 0, tft.width());
112       data->point.y = map((tft.width() - touchX), 0, tft.width(), 0, tft.height());
113 #elif (TFT_DIRECTION == 2)
114       data->point.x = tft.width() - touchX;
115       data->point.y = tft.height() - touchY;
116 #elif (TFT_DIRECTION == 3)
117       data->point.x = map((tft.height() - touchY), 0, tft.height(), 0, tft.width());
118       data->point.y = map(touchX, 0, tft.width(), 0, tft.height());
119 #endif
120       // Serial.print("Data x ");
121       // Serial.print(data->point.x);
122       // Serial.print("\tData y ");
123       // Serial.println(data->point.y);
124     }
```

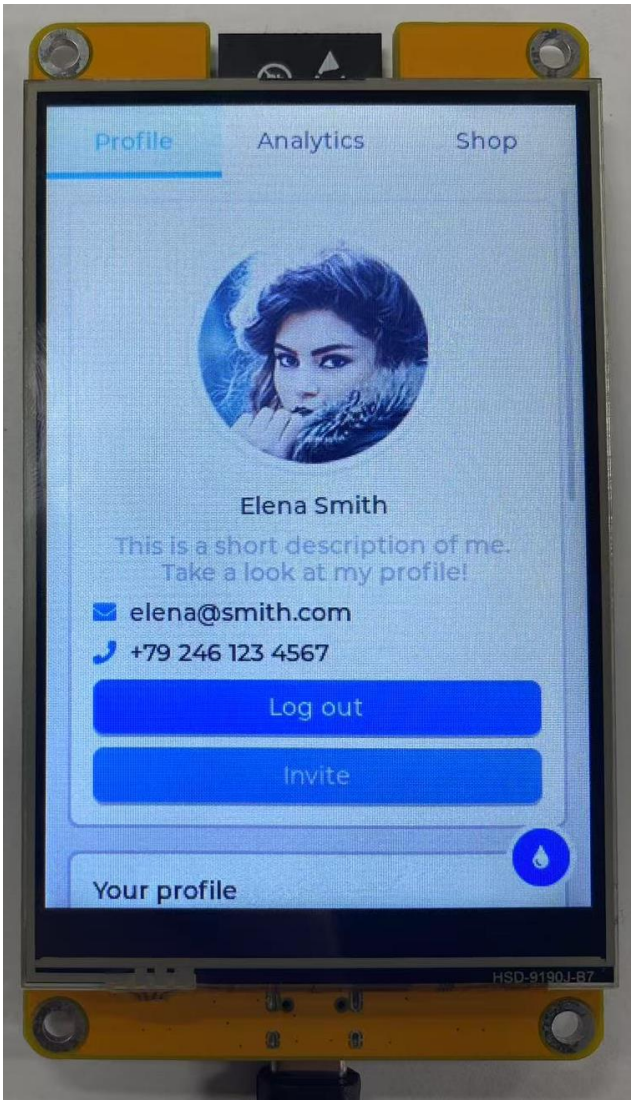
```
#elif defined(FNK0103N_3P5_320x480_ST7796) || defined(FNK0103S_4P0_320x480_ST7796)
    touched = tft.getTouch(&touchX, &touchY, 600);
    if (!touched) {
      data->state = LV_INDEV_STATE_REL;
    } else {
      data->state = LV_INDEV_STATE_PR;
#if (TFT_DIRECTION == 0)
      data->point.x = tft.width() - touchX;
      data->point.y = touchY;
#elif (TFT_DIRECTION == 1)
      data->point.x = map(touchY, 0, tft.height(), 0, tft.width());
      data->point.y = map(touchX, 0, tft.width(), 0, tft.height());
#elif (TFT_DIRECTION == 2)
      data->point.x = touchX;
      data->point.y = tft.height() - touchY;
#elif (TFT_DIRECTION == 3)
      data->point.x = map((tft.height() - touchY), 0, tft.height(), 0, tft.width());
      data->point.y = map(tft.width() - touchX, 0, tft.width(), 0, tft.height());
#endif
      // Serial.print("Data x ");
      // Serial.print(data->point.x);
      // Serial.print("\tData y ");
      // Serial.println(data->point.y);
    }
#endif
}

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

      lv_init();

#if LV_USE_LOG != 0
      lv_log_register_print_cb( my_print ); /* register print function for debugging */
#endif

```

```
169     tft.begin();          /* TFT init */
170     tft.setRotation(TFT_DIRECTION); /* Landscape orientation, flipped
171
172     /*Set the touchscreen calibration data,
173      the actual data for your display can be acquired using
174      the Generic -> Touch_calibrate example from the TFT_eSPI library*/
175     #if defined(FNK0103B_2P8_240x320_ST7789) || defined(FNK0103F_2P8_240x320_ILI9341)
176         touch.setCal(526, 3443, 750, 3377, screenHeight, screenWidth, 1);
177     #elif defined(FNK0103L_3P2_240x320_ST7789)
178         uint16_t calData[5] = { 286, 3534, 283, 3600, 6 };
179         tft.setTouch(calData);
180     #elif defined(FNK0103N_3P5_320x480_ST7796) || defined(FNK0103S_4P0_320x480_ST7796)
181         uint16_t calData[5] = { 286, 3534, 283, 3600, 6 };
182         tft.setTouch(calData);
183     #endif
184
185     lv_disp_draw_buf_init(&draw_buf, draw_buf1, NULL, screenWidth * screenHeightBuf);
186
187     /*Initialize the display*/
188     static lv_disp_drv_t disp_drv;
189     lv_disp_drv_init( &disp_drv );
190     /*Change the following line to your display resolution*/
191     #if (TFT_DIRECTION % 2 == 0)
192     disp_drv.hor_res = screenWidth;
193     disp_drv.ver_res = screenHeight;
194     #else
195     disp_drv.hor_res = screenHeight;
196     disp_drv.ver_res = screenWidth;
197     #endif
198     disp_drv.flush_cb = my_disp_flush;
199     disp_drv.draw_buf = &draw_buf;
200     lv_disp_drv_register( &disp_drv );
201
202     /*Initialize the (dummy) input device driver*/
203     static lv_indev_drv_t indev_drv;
204     lv_indev_drv_init( &indev_drv );
205     indev_drv.type = LV_INDEV_TYPE_POINTER;
206     indev_drv.read_cb = my_touchpad_read;
207     lv_indev_drv_register( &indev_drv );
208
209     /* Create simple label */
210     lv_obj_t *label = lv_label_create( lv_scr_act() );
211     lv_label_set_text( label, "Hello Ardino and LVGL!");
212     lv_obj_align( label, LV_ALIGN_CENTER, 0, 0 );
```

```
213
214       /* Try an example. See all the examples
215        * online: https://docs.lvgl.io/master/examples.html
216        * source codes:
217  https://github.com/lvgl/lvgl/tree/e7f88efa5853128bf871dde335c0ca8da9eb7731/examples */
218        //lv_example_btn_1();
219
220        /*Or try out a demo. Don't forget to enable the demos in lv_conf.h. E.g.
221  LV_USE_DEMOS_WIDGETS*/
222       lv_demo_widgets();
223       // lv_demo_benchmark();
224       // lv_demo_keypad_encoder();
225       // lv_demo_music();
226       // lv_demo_printer();
227       // lv_demo_stress();
228
229       Serial.println( "Setup done" );
230  }
231
232  void loop()
233  {
234       lv_timer_handler(); /* let the GUI do its work */
235       delay( 5 );
236  }
```

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




Profile
Analytics
Shop
Elena Smith
This is a short description of me.
Take a look at my profile!
elena@smith.com
+79 246 123 4567
Log out
Invite
Your profile