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.

Get Support and Offer Input

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

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

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

support@freenove.com

Safety and Precautions

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

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

About Freenove

Freenove provides open source electronic products and services worldwide.

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

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

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

http://www.freenove.com

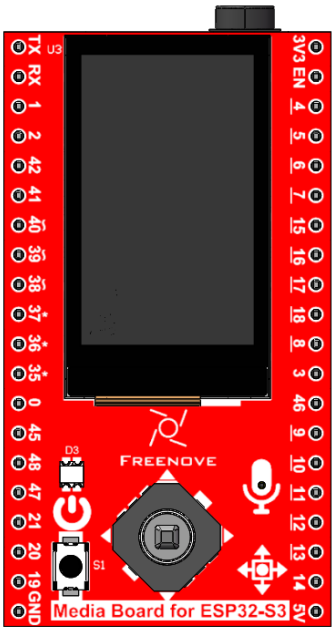
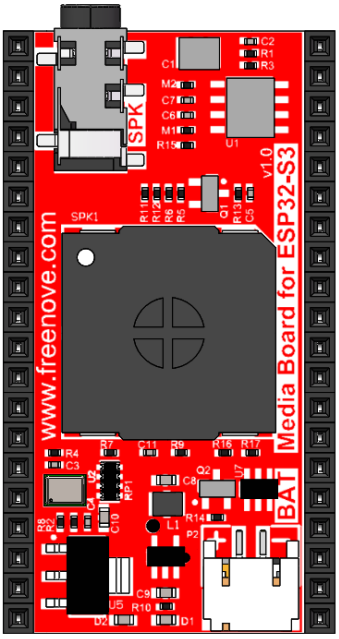
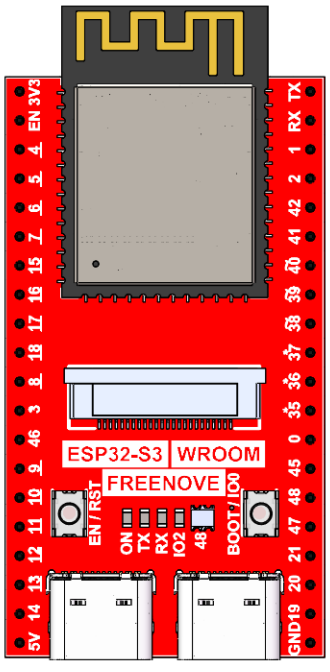
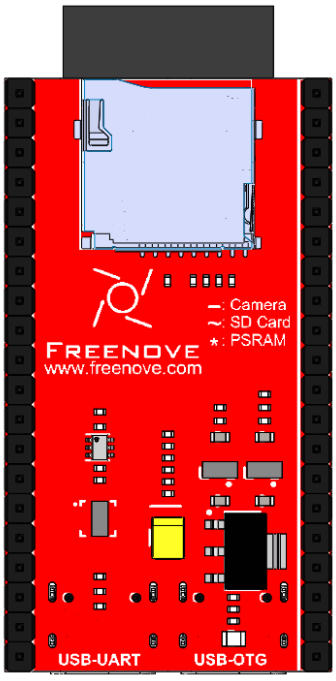
sale@freenove.com

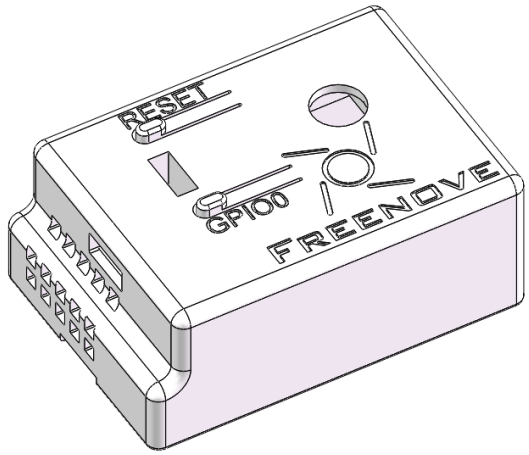
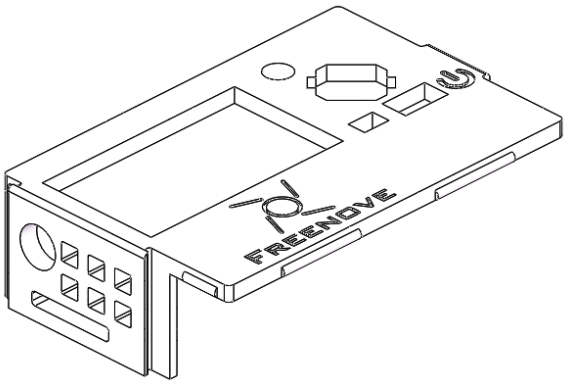
Copyright

Contents

List

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

Top of the Extension Board x1	Bottom of the Extension Board x1
	
Top of ESP32-S3 N16R8 x1	Bottom of ESP32-S3 N16R8 x1
	

Housing Base (3D Printed) x1 	Housing Cover (3D Printed) x1 
SD card reader x1 (random color)  (Not a USB flash drive.)	SDcard x1 

Important Notes:

1. Surface Finish: The housing is 3D-printed using FDM technology with a layer resolution of 0.1–0.2 mm. As a result, minor surface roughness and visible layer lines are normal.
2. Color Change: The white PLA material may gradually develop a beige/yellowish tint over time due to environmental exposure. This is an inherent characteristic of the material.
3. Handling Care: PLA is less impact-resistant than conventional plastics. Please handle the housing gently and avoid excessive force during assembly.

Replacement Options:

1. 3D model files are available in Freenove_Media_Kit_for_ESP32-S3\3D_Models for self-printing if needed.
2. If you don't have a 3D printer, you can order a replacement by uploading the STL file at JLCPCB (https://jlc3dp.com/?href=easyeda-home).

Preface

ESP32-S3 WROOM

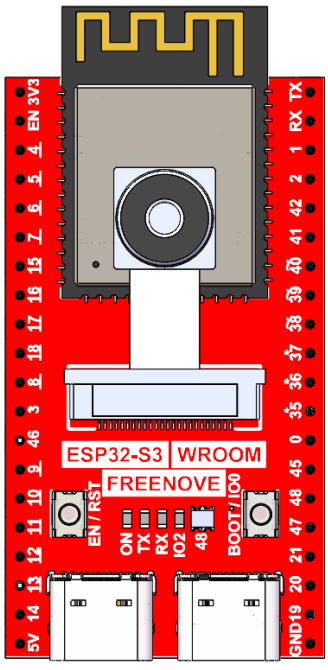
The ESP32-S3-WROOM-1 offers two antenna options: the PCB on-board antenna and the IPEX antenna.

- 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.

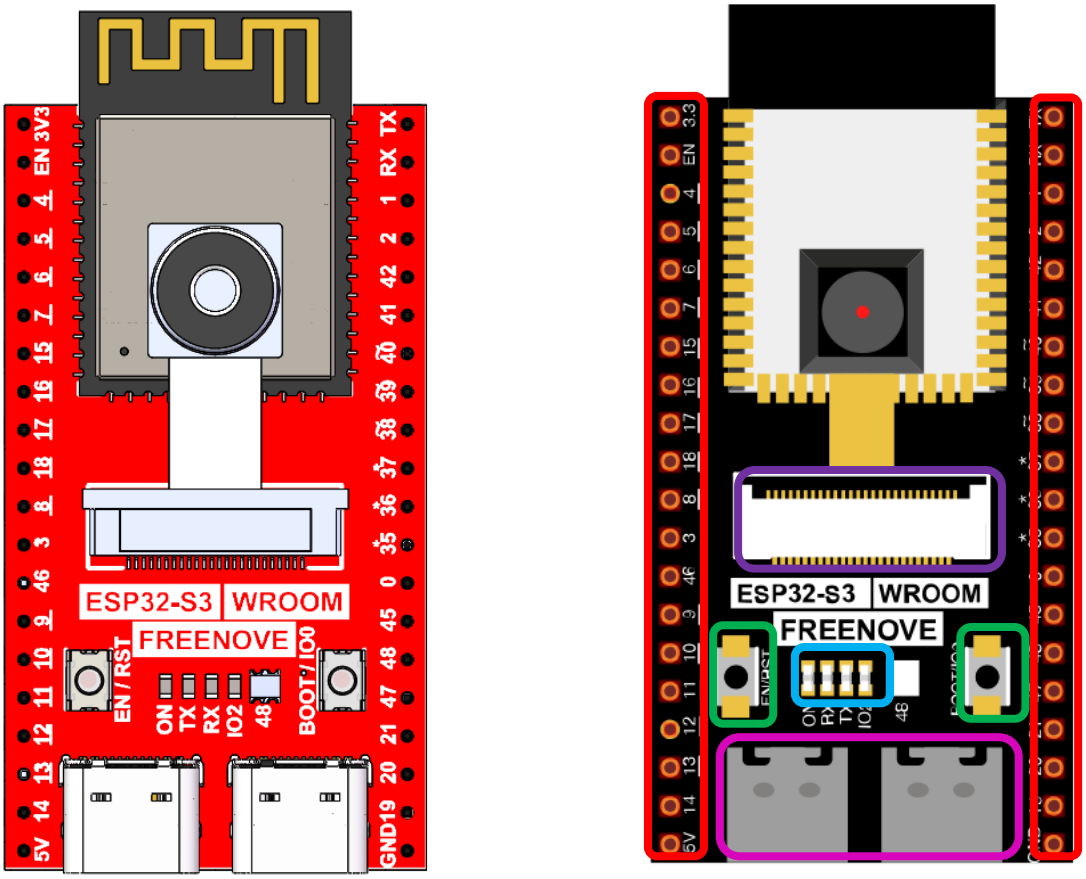
PCB on-board antenna	IPEX antenna
	

The ESP32-S3 WROOM of this product is based on the ESP32-S3-WROOM-1 module with built-in PCB on-board antenna.

ESP32-S3 WROOM



The hardware interfaces of ESP32-S3 WROOM are distributed as follows:



Compare the left and right images. We've boxed off the resources on the ESP32-S3 WROOM in different colors to facilitate your understanding of the board.

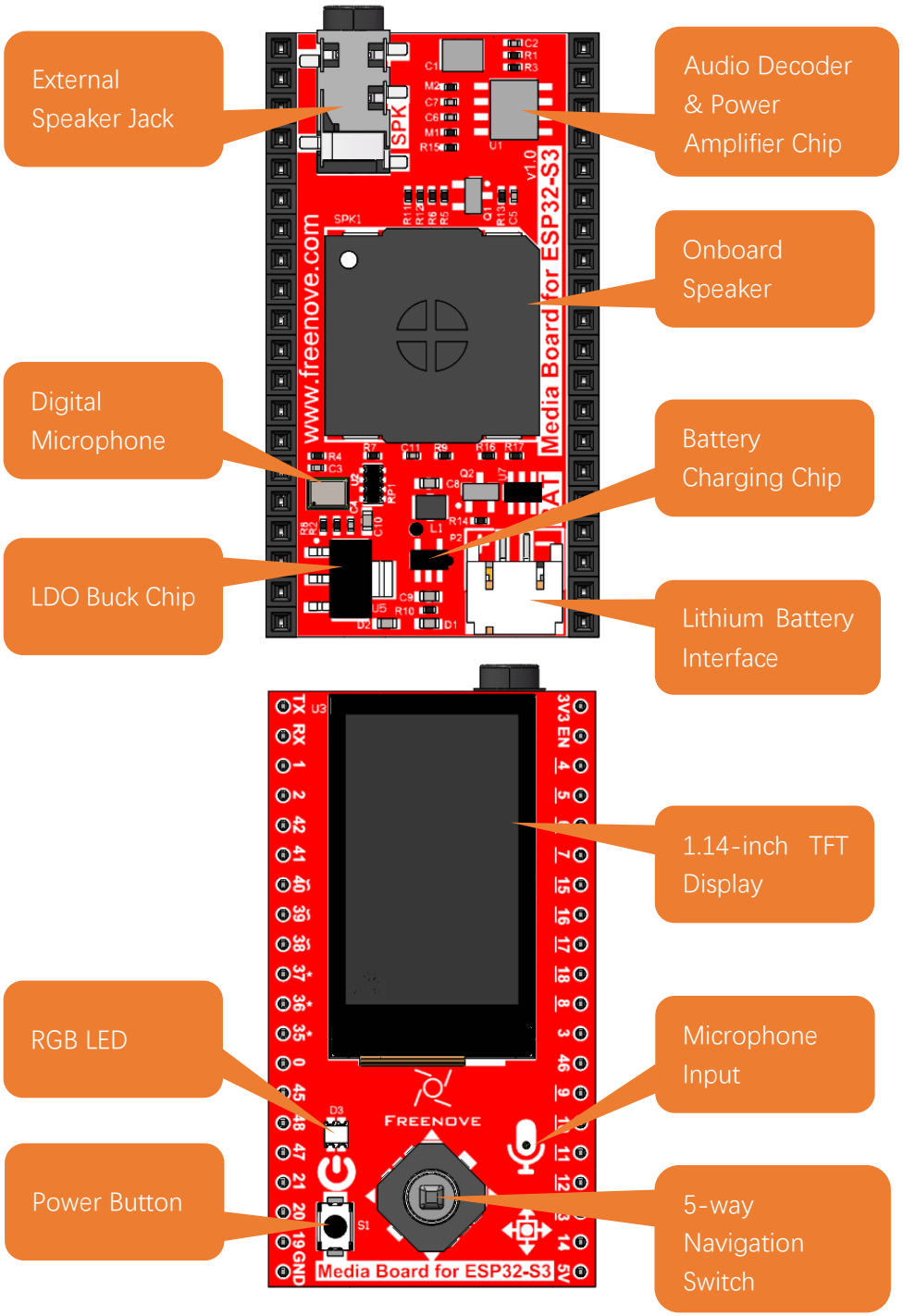
Box color	Corresponding resources introduction
	GPIO pins
	LED indicators
	Camera interface
	Reset button, Boot mode selection button
	USB ports

For more information, please visit: https://www.espressif.com.cn/sites/default/files/documentation/esp32-s3-wroom-1_wroom-1u_datasheet_en.pdf.

GPIO pins of ESP32-S3 WROOM can be used to interface with external devices and control peripheral circuits.

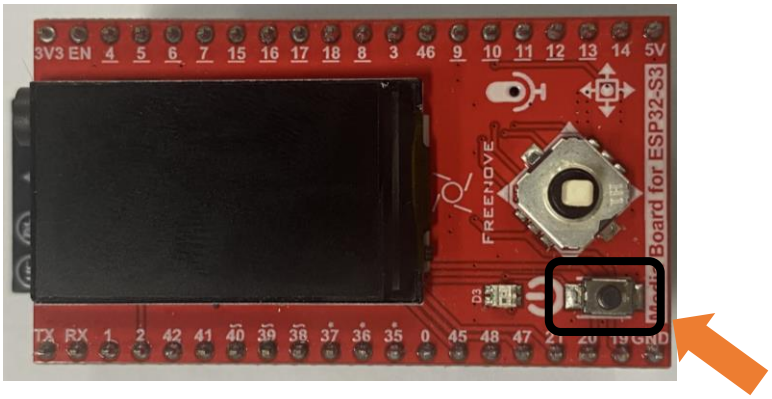
Freenove Media Kit for ESP32-S3

Freenove Media Kit for ESP32-S3 is an expansion board designed for the Freenove ESP32-S3 WROOM Board. Its key features are illustrated below.



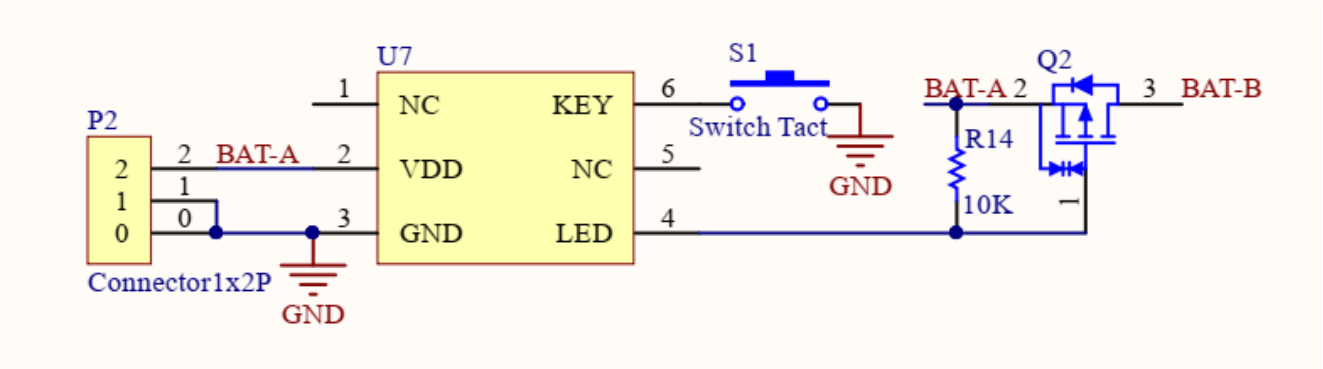
Power Button

The power button on the Freenove Media Kit for ESP32-S3, as shown in the diagram below, only works when powered by a battery (used to control battery connection/disconnection). **It has no effect when powered via USB.**



Schematic

The schematic of the power button is as shown below:



Battery (Optional)

This product does not include a lithium battery. Please purchase one separately.

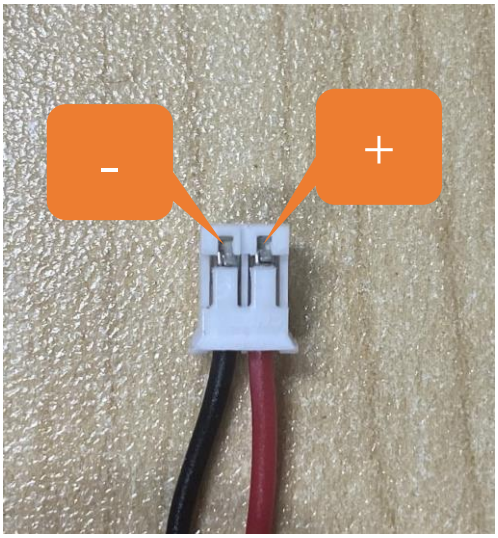
The device supports both **USB direct power supply** and **lithium battery power supply**.
We strongly recommend using USB power whenever possible, as lithium batteries can be hazardous and require careful handling. Avoid using a battery unless absolutely necessary.

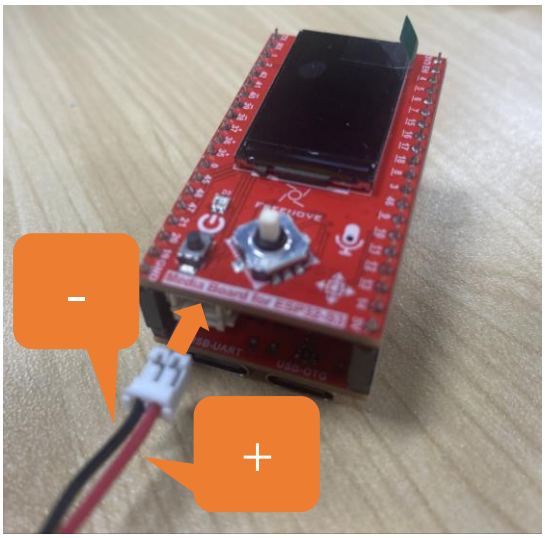
The recommended size of lithium batteries for this product is: 7.5mmx20mmx35mm. Search for "702035 battery" on any shopping platform.

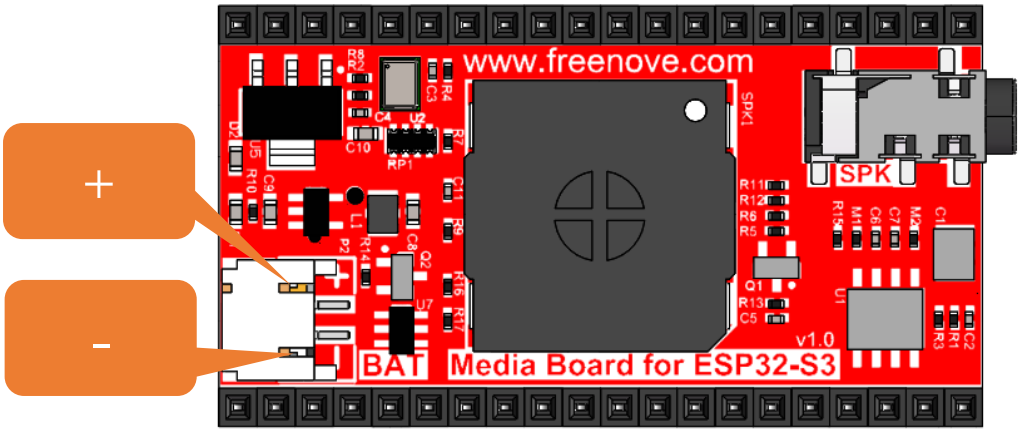
This product uses **a PH2.0mm 2P connector** for power supply. You may purchase lithium batteries of any capacity, but please note that the rated input voltage range must be **3.7V–4.2V**.

Commercially available batteries may have two different wiring polarities (positive/negative reversed). You must verify that the purchased battery's pinout matches the product's requirements (as shown in the diagram below). Incorrect polarity may cause device damage or safety hazards.

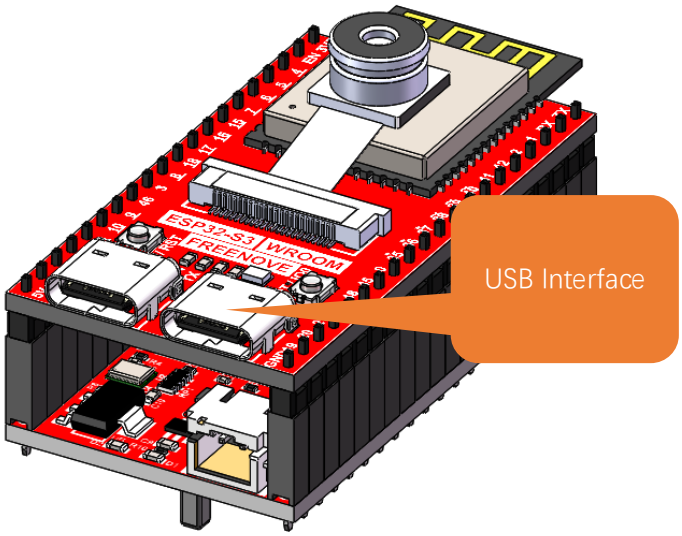
The red wire is Positive (+) and black wire Negative (-)







We recommend using the dedicated charger designed for your lithium battery.
Since lithium batteries vary in specifications and quality, using the correct charger helps ensure optimal performance, safety, and longevity.
While our product also supports USB charging as a backup option, please note that this method does not support fast charging and provides only a slow, standard charge.



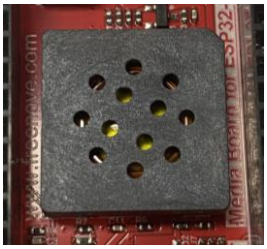
Charging & Power Indicators:

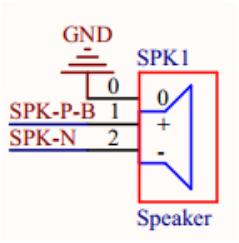
When using the USB port on the board to charge the battery:
While charging, the blue LED will blink.
When charging is complete, the blue LED will stay lit.
If no battery is connected, the blue LED will keep blinking.

When the device is not connected to USB, it runs on battery power, and the green LED remains steadily lit.

Speaker and Headphone Connector

The Freenove Media Kit for ESP32-S3 is equipped with a speaker that supports high-quality audio output, capable of meeting the playback requirements for various sound effects and music. You can learn how to generate simple tones in Chapter 5: Simple Tone Test, and how to play music in Chapter 6: Play MP3 Test.



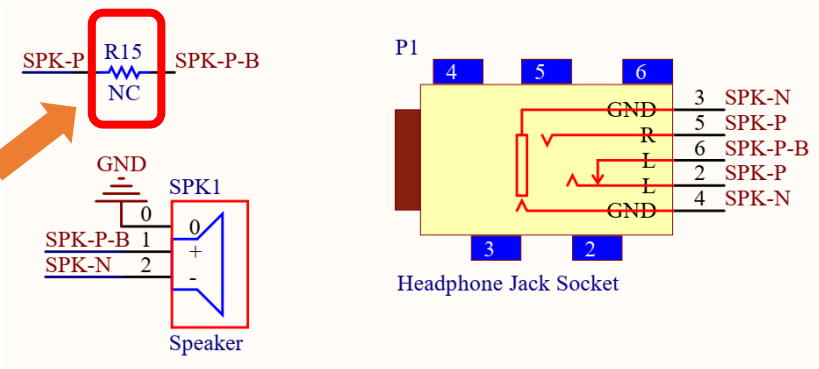


The following table shows the specifications of the speaker.

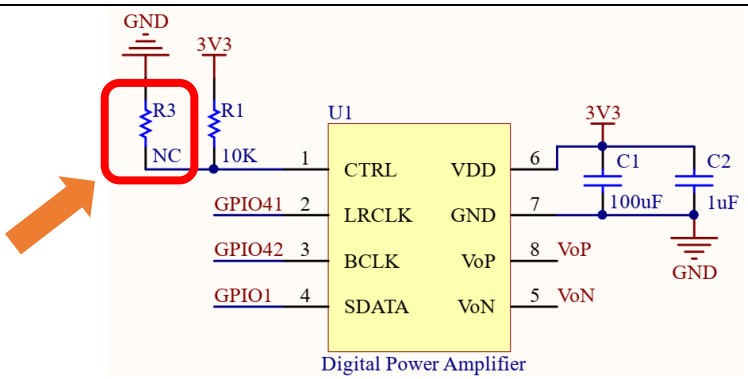
Specifications	Description	Typical value
Rated Impedance	The resistance of the speaker to AC current, affecting amplifier matching.	8Ω
Rated Power	The maximum continuous power the speaker can handle.	1W
Frequency Range	The effective operating frequency range of the speaker.	550~20kHz
Maximum Sound Pressure Level (SPL)	The highest sound intensity the speaker can produce under **specific conditions**.	96±3dB
Dimensions	The length and width of the speaker.	18mm * 18mm

Notes:

1. **Resistor R15 is not soldered by default** (as shown in the figure below).
   - When R15 is not soldered: Inserting a headphone plug disconnects the speaker.
   - When R15 is soldered: Inserting a headphone plug allows both the speaker and headphone jack to output audio simultaneously.



2. Resistor R1 is soldered by default**, while R3 is not soldered by default.**
   - When only R1 is soldered: The right channel is active.
   - When only R3 is soldered: The left channel is active.



The onboard speaker can be used to play any audio, but the volume may be relatively low. If you wish to use external speakers, you can connect them via the headphone jack.

This product features a 3.5mm TRS headphone socket. If you intend to connect your own speakers using an audio cable, please ensure you purchase a compatible 3.5mm TRS plug.

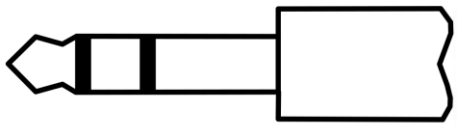
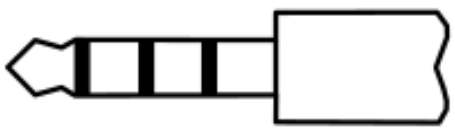
Common 3.5mm male plugs include TRS Jack and TRRS Jack, which are easily distinguishable:

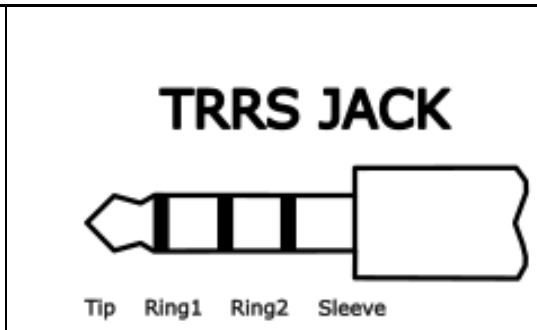
TRS Jack has two metal rings (Tip, Ring, Sleeve).

TRRS Jack has three metal rings (Tip, Ring, Ring, Sleeve). (See figure below.)

The TRS Jack (2-ring) headphone plug is compatible with the 3.5mm female socket on the Freenove Media Kit for ESP32-S3.

TRRS Jacks (3-ring) come in two common standards: OMTP and CTIA, differing in the wiring of Microphone (MIC) and Ground (GND) on the Ring2 and Sleeve contacts (see figure below).

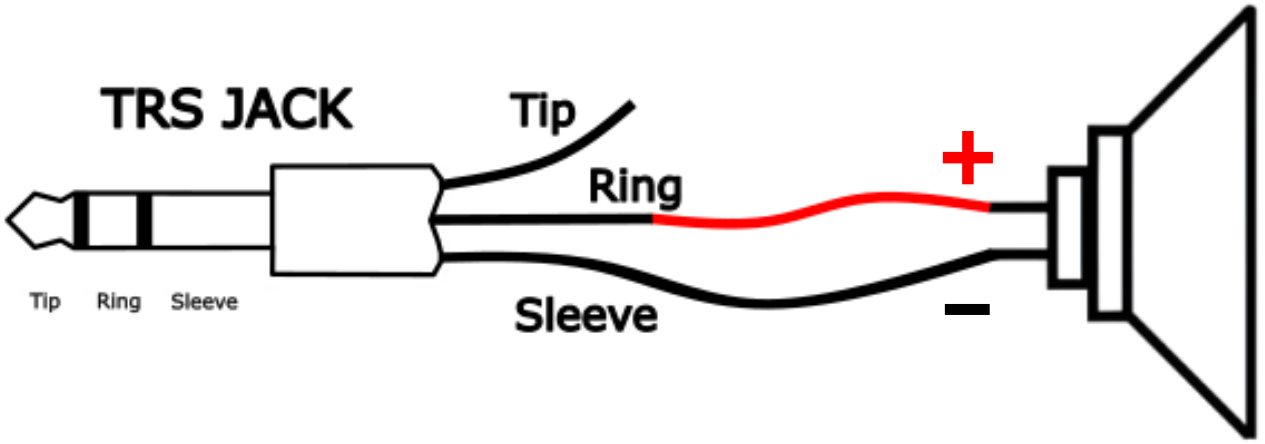
<table>
<tr><td colspan="2" rowspan="3">TRS JACK
 
</td><td>Tip</td><td>Left channel</td></tr>
<tr><td>Ring</td><td>Right channel</td></tr>
<tr><td>Sleeve</td><td>GND</td></tr>
<tr><td rowspan="4">OMTP</td><td rowspan="4">TRRS JACK
 
</td><td>Tip</td><td>Left channel</td></tr>
<tr><td>Ring1</td><td>Right channel</td></tr>
<tr><td>Ring2</td><td>Microphone</td></tr>
<tr><td>Sleeve</td><td>GND</td></tr>
</table>

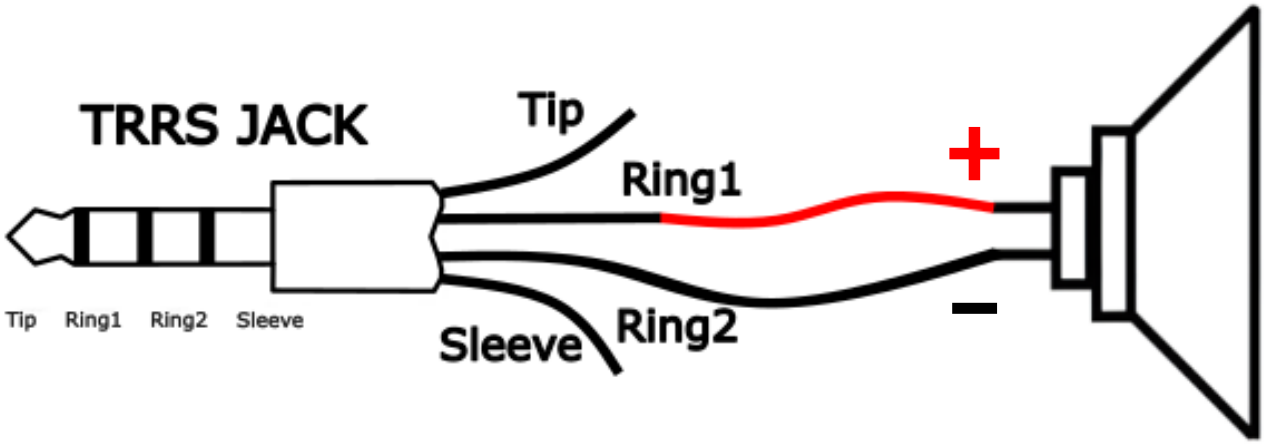
CTIA		Tip	Left channel
		Ring1	Right channel
		Ring2	GND
		Sleeve	Microphone

Important Notes:

1. If you connect headphones to the 3.5mm female socket, set the volume to 5 or lower (max volume = 21) in your code. Excessive volume may damage your headphones.
   The Freenove Media Kit for ESP32-S3 is fully compatible with CTIA-standard (modern) headphones, but only partially supports OMTP-standard (older) headphones. Please verify compatibility before use.
2. If you connect a speaker to the 3.5mm socket, refer to the wiring diagram below for proper connection.

- Maximum Supported Speaker Power: The built-in audio output supports up to 2.5W speakers. For higher-power speakers, an external amplifier is required.

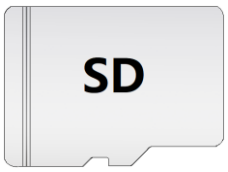




SD Card

Freenove Media Kit for ESP32-S3 comes with a 1GB SD card and a SD card reader (see the figure below). The SD card uses **SDMMC** communication protocol, providing faster speeds and better performance compared to SPI proocol.

Please note that the included card reader cannot be used as a USB flash driver. It is specifically designed for SD card access only.

SD card x1	Card reader x1 (random color)
	

For more information and instuctions for using the SD card, please refer to Chapter 4.

The following table shows the pin definition of the SD card.

Item	Pins	Definition
SD Card	GPIO38	SD_CMD
	GPIO39	SD_CLK
	GPIO40	SD_D0

TFT Display

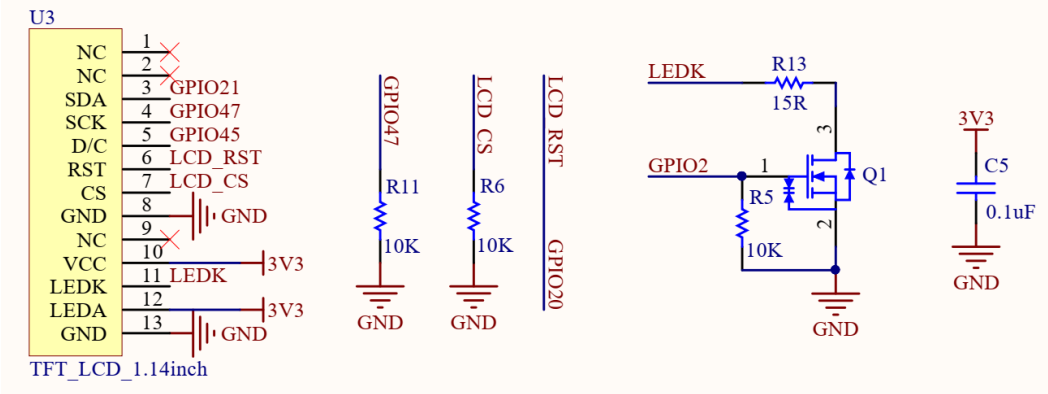
Freenove Media Kit for ESP32-S3 features a **1.14-inch** TFT display. With each pixel is individually controlled by a tiny transistor, TFT (Thin-Film Transistor) displays, a common type of LCD screen, offer high responsiveness, brightness, and contrast.

Specifications of the TFT Display

Specifications	Description
Dimensions	1.14 inch
Resolution	135×240 pixel
Drvier	ST7789
Display Area	14.9mm * 24.9mm
Rated Voltage	3.3V
Communication	SPI

The 1.14-inch screen size specification refers to the diagonal measurement of the display's active viewing area, which equals approximately 2.6 centimeters (1.14 inches).

Schematic of the TFT Display



Note: During display reset operations, GPIO20 must be configured in output mode.
The reset sequence requires:
First output a low level and maintain it for a delay period
Then switch to high level to complete the reset timing

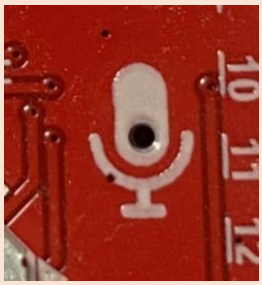
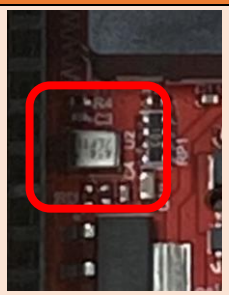
The following table shows the pin definition of the TFT display.

Item	Pins	Definition
TFT display	GPIO21	LCD_SDA
	GPIO47	LCD_SCK
	GPIO45	LCD_D/C
	GPIO20	LCD_RST

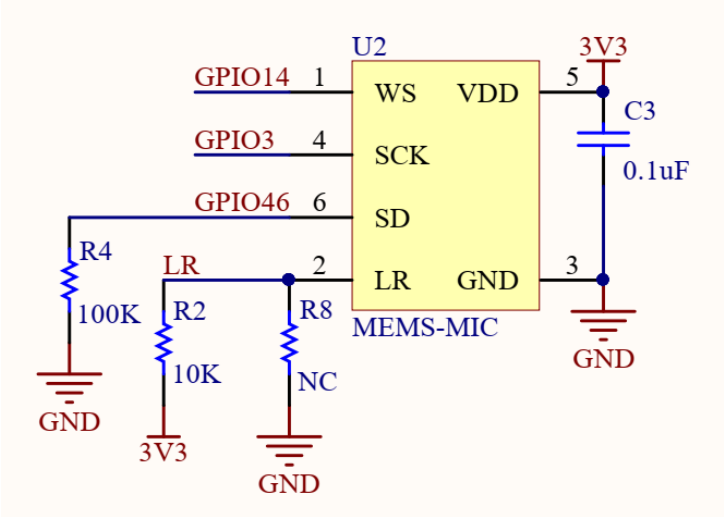
Microphone

The Freenove Media Kit for ESP32-S3 includes a MEMS microphone (Micro-Electro-Mechanical Systems microphone), which offers several advantages over traditional ECM (Electret Condenser Microphones) including smaller size, higher sensitivity and signal-to-noise ratio (SNR), and superior noise immunity, making them ideal for **compact embedded applications**.

For further details on MEMS microphone technology, refer to Chapter 10

Top	Bottom
	

Shematic

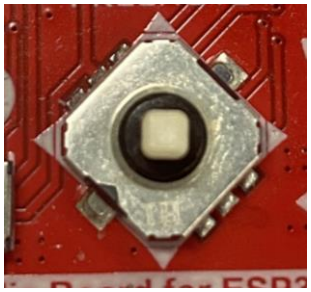


The following table shows the pin definition of the microphone

Item	Pins	Definition
Mic	GPIO14	MIC_WS
	GPIO3	MIC_SCK
	GPIO46	MIC_SD

5-way Navigation Switch

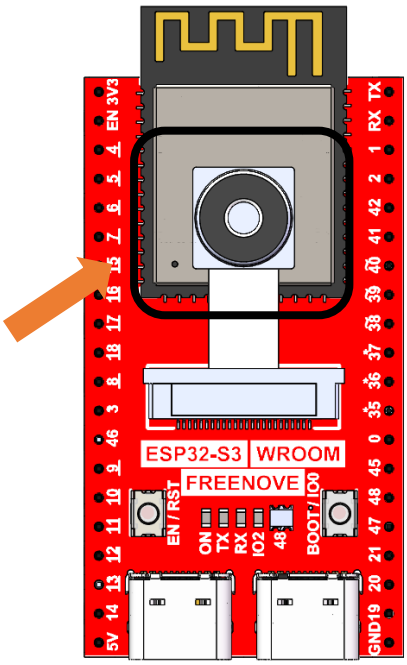
Freenove Media Kit for ESP32-S3 utilizes a 5-way navigation button for human-machine interaction. For more details about the 5-way button, please refer to Chapter 3.



Item	Pin	Definition
Power Button	GPIO19	PowerButton_COM

Camera

Freenove Media Kit for ESP32-S3 integrates a camera module.



The following table shows the pin definition of the camera.

Item	Pins	Definition
Camera	GPIO4	SIOD
	GPIO5	SIOC
	GPIO6	CSI_VYSNC
	GPIO7	CSI_HREF
	GPIO16	CSI_Y9
	GPIO15	XCLK
	GPIO17	CSI_Y8
	GPIO18	CSI_Y7
	GPIO13	CSI_PCLK
	GPIO12	CSI_Y6
	GPIO11	CSI_Y2
	GPIO10	CSI_Y5
	GPIO9	CSI_Y3
	GPIO8	CSI_Y4

Notes for GPIO

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	Funtions	Description
GPIO48	WS2812_DIN	WS2812
GPIO21	LCD_SDA	TFT_LCD
GPIO47	LCD_SCK	
GPIO45	LCD_D/C	
GPIO20	LCD_RST	
GPIO14	MIC_WS	Mic
GPIO3	MIC_SCK	
GPIO46	MIC_SD	
GPIO19	PowerButton_COM	Power Button
GPIO41	NS4168_LRCLK	Digital Power Amplifier
GPIO42	NS4168_BCLK	
GPIO1	NS4168_SDATA	
GPIO4	SIOD	Camera
GPIO5	SIOC	
GPIO6	CSI_VYSNC	
GPIO7	CSI_HREF	
GPIO16	CSI_Y9	
GPIO15	XCLK	
GPIO17	CSI_Y8	
GPIO18	CSI_Y7	
GPIO13	CSI_PCLK	
GPIO12	CSI_Y6	
GPIO11	CSI_Y2	
GPIO10	CSI_Y5	
GPIO9	CSI_Y3	
GPIO8	CSI_Y4	
GPIO38	SD_CMD	SD Card
GPIO39	SD_CLK	
GPIO40	SD_D0	

For more information, refer to the schematic.

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

PSRAM Pin

The module on the ESP32-S3-WROOM board utilizes the ESP32-S3R16 chip, which comes with 8MB of external Flash. When using the OPI PSRAM, it should be noted that GPIO35-GPIO37 on the ESP32-S3-WROOM board will not be available for other purposes. However, when OPI PSRAM is not used, GPIO35-GPIO37 on the board can be used as normal GPIO.

ESP32-S3R8 / ESP32-S3R8V	In-package PSRAM (8 MB, Octal SPI)
SPICLK	CLK
SPICS1	CE#
SPID	DQ0
SPIQ	DQ1
SPIWP	DQ2
SPIHD	DQ3
GPIO33	DQ4
GPIO34	DQ5
GPIO35	DQ6
GPIO36	DQ7
GPIO37	DQS/DM

SDcard Pin

An SDcard slot is integrated on the back of the ESP32-S3-WROOM board, and we can use GPIO38-GPIO40 of ESP32-S3-WROOM to drive SD card.

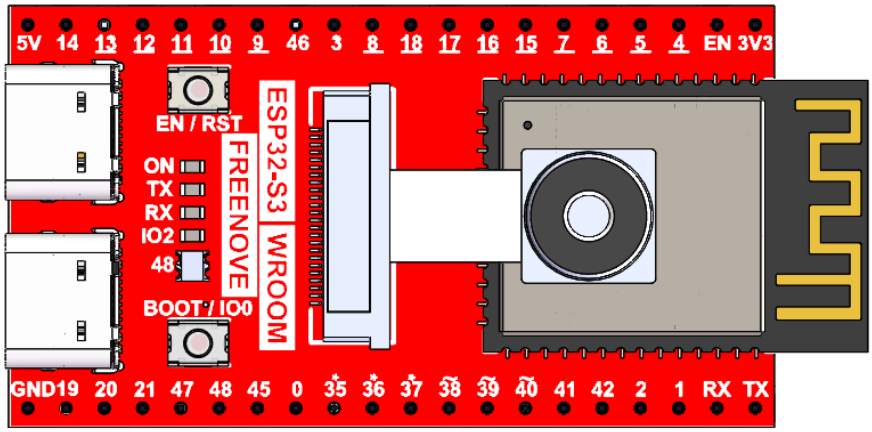
The SDcard of ESP32-S3-WROOM uses SDMMC, a 1-bit bus driving method, which is integrated in the Arduino IDE, and we can call the "SD_MMC.h" library to drive it. For more details, please refer to the SDcard chapter in this tutorial.

USB Pin

Please note that in this product, GPIO20 is used for both battery voltage sampling (ADC) and LCD reset signal (RST). Therefore, it must not be configured for USB functions to avoid signal interference.

Cam Pin

When using the camera on our ESP32-S3-WROOM, please check its pin assignments. Pins marked with underlined numbers are dedicated to the camera function. If you intend to use additional functions alongside the camera, avoid using these pins to prevent conflicts.



If you have any questions regarding GPIO information, you can click here to navigate back to the ESP32-S3 WROOM and view specific GPIO details.
Or check: https://www.espressif.com/sites/default/files/documentation/esp32-s3_datasheet_en.pdf.

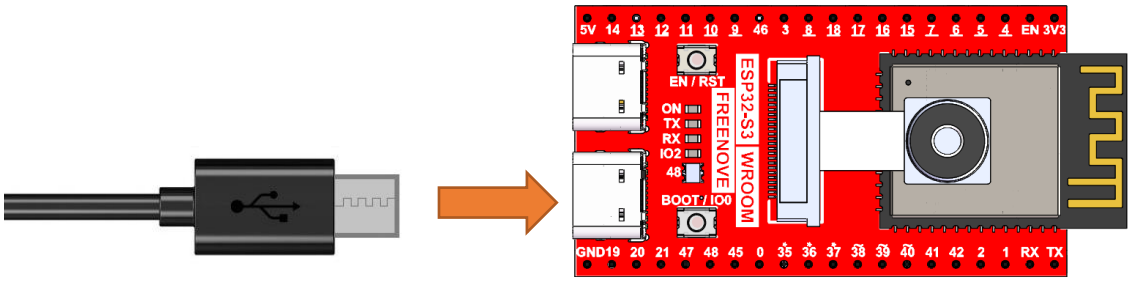
CH343 (Required)

ESP32-S3 WROOM uses CH343 to download code. Therefore, before using the device, it is necessary to install the CH343 driver on your computer.

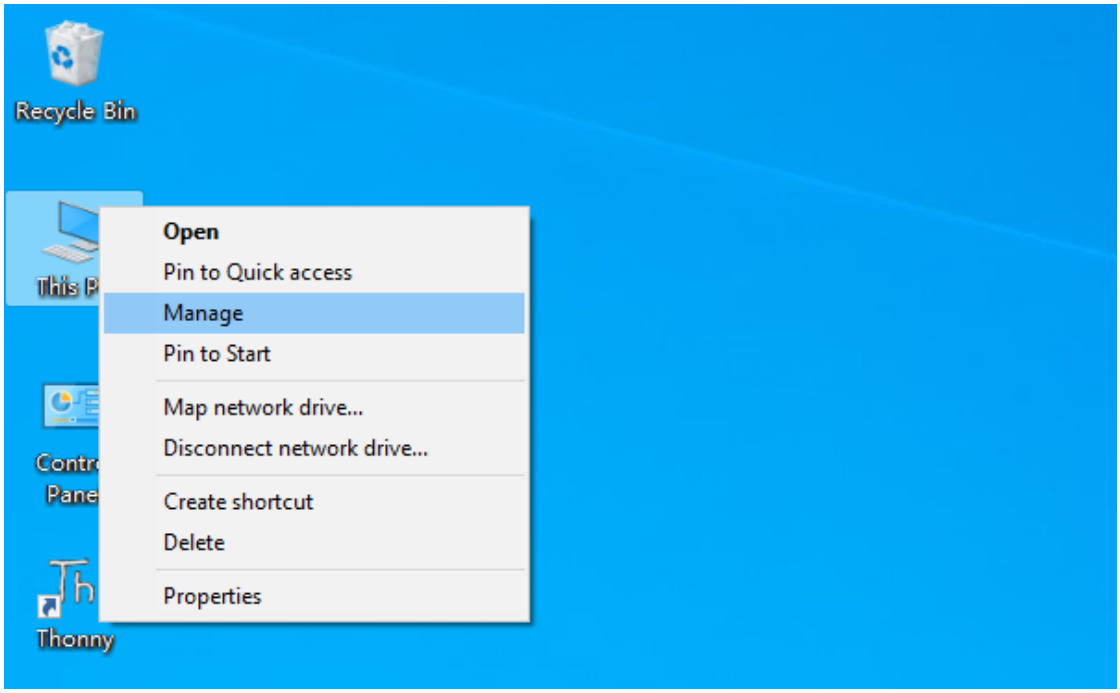
Windows

Check whether CH343 has been installed

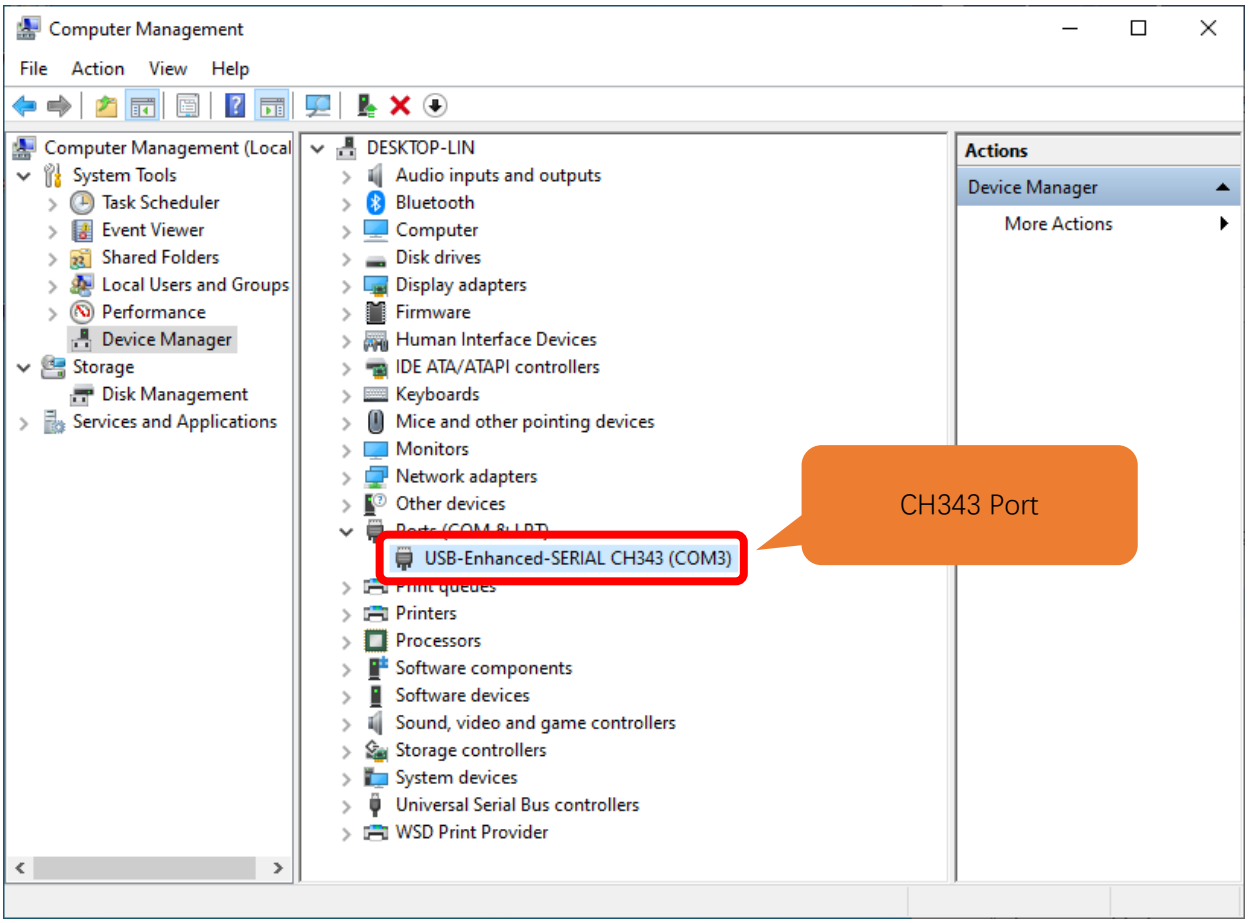
1. Connect your computer and ESP32-S3 WROOM with a USB cable.



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

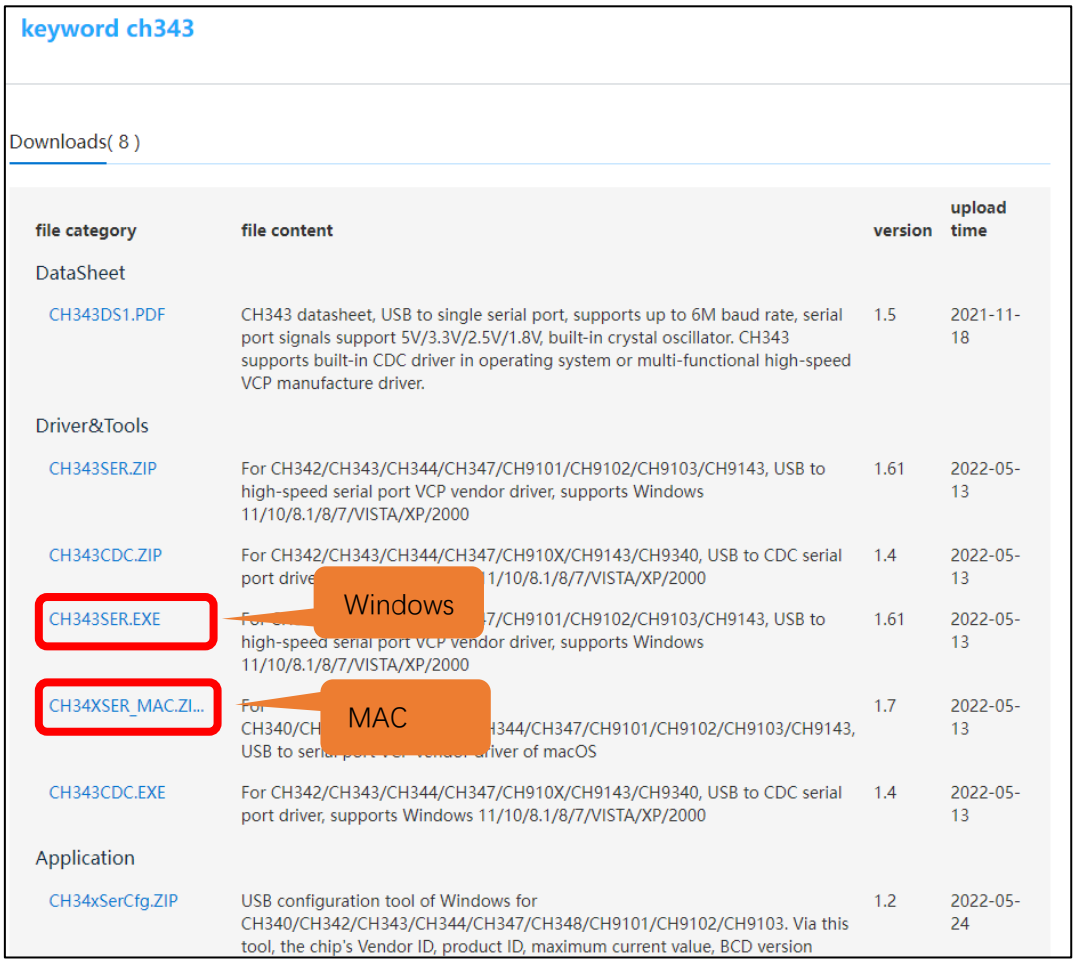


3. Click "Device Manager". If your computer has installed CH343, you can see"USB-Enhances-SERIAL CH343 (COMx)". And you can click here to move to the next step.

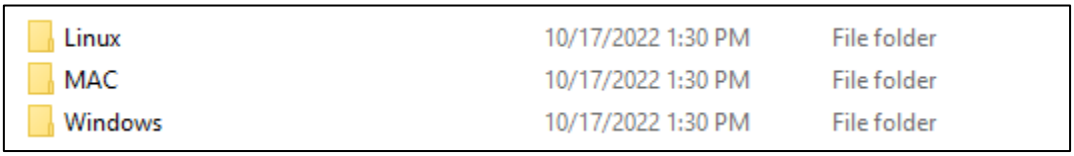


Installing CH343

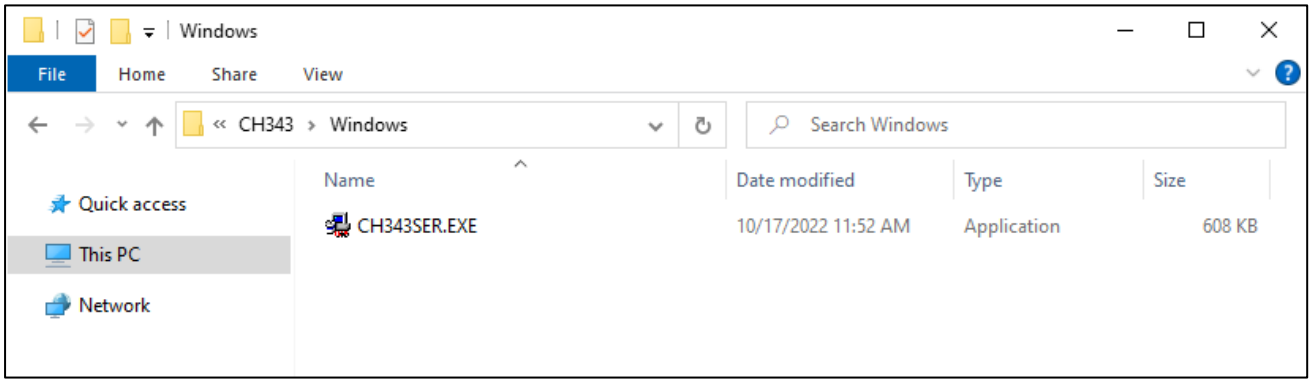
1. First, download CH343 driver, click http://www.wch-ic.com/search?t=all&q=ch343 to download the appropriate one based on your operating system.



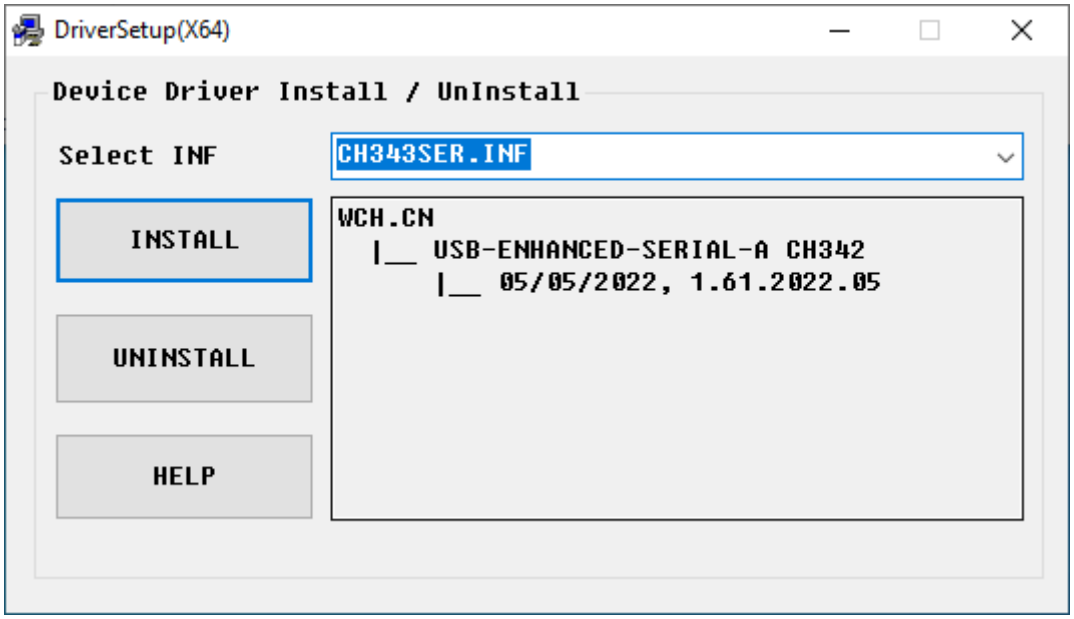
If you would not like to download the installation package, you can open **"Freenove_Media_Kit_for_ESP32-S3/CH343"**, we have prepared the installation package.



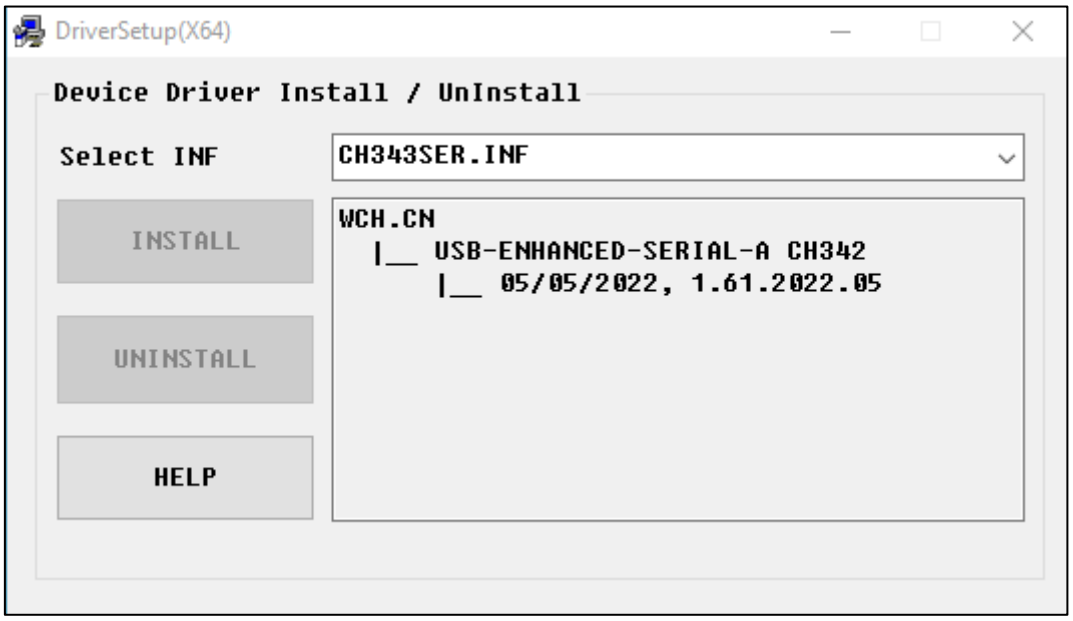
2. Open the folder "**Freenove_Media_Kit_for_ESP32-S3/CH343/Windows/**"



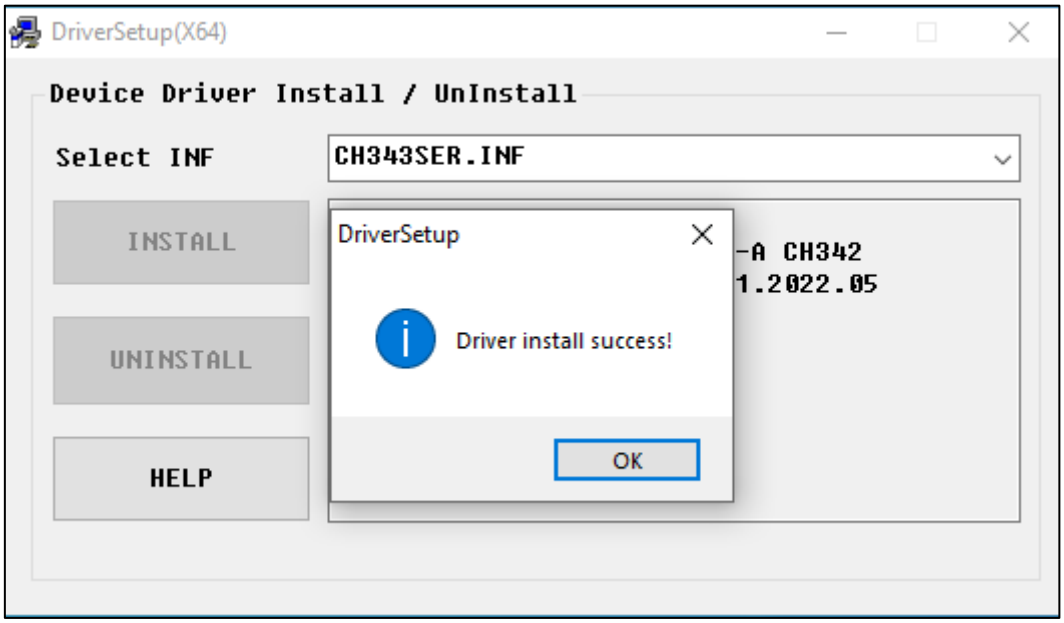
3. Double click "**CH343SER.EXE**".



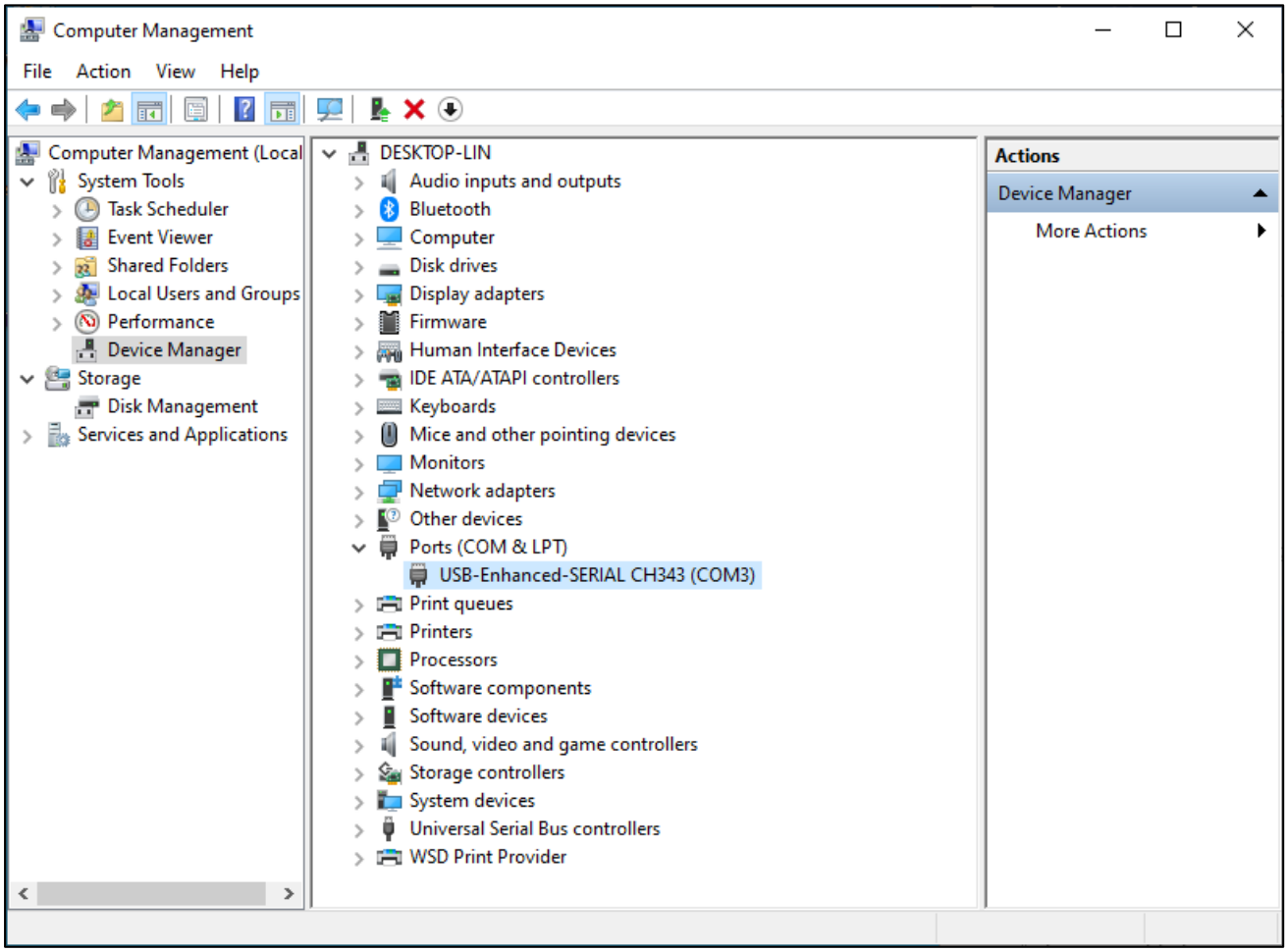
4. Click "INSTALL" and wait for the installation to complete.



5. Install successfully. Close all interfaces.



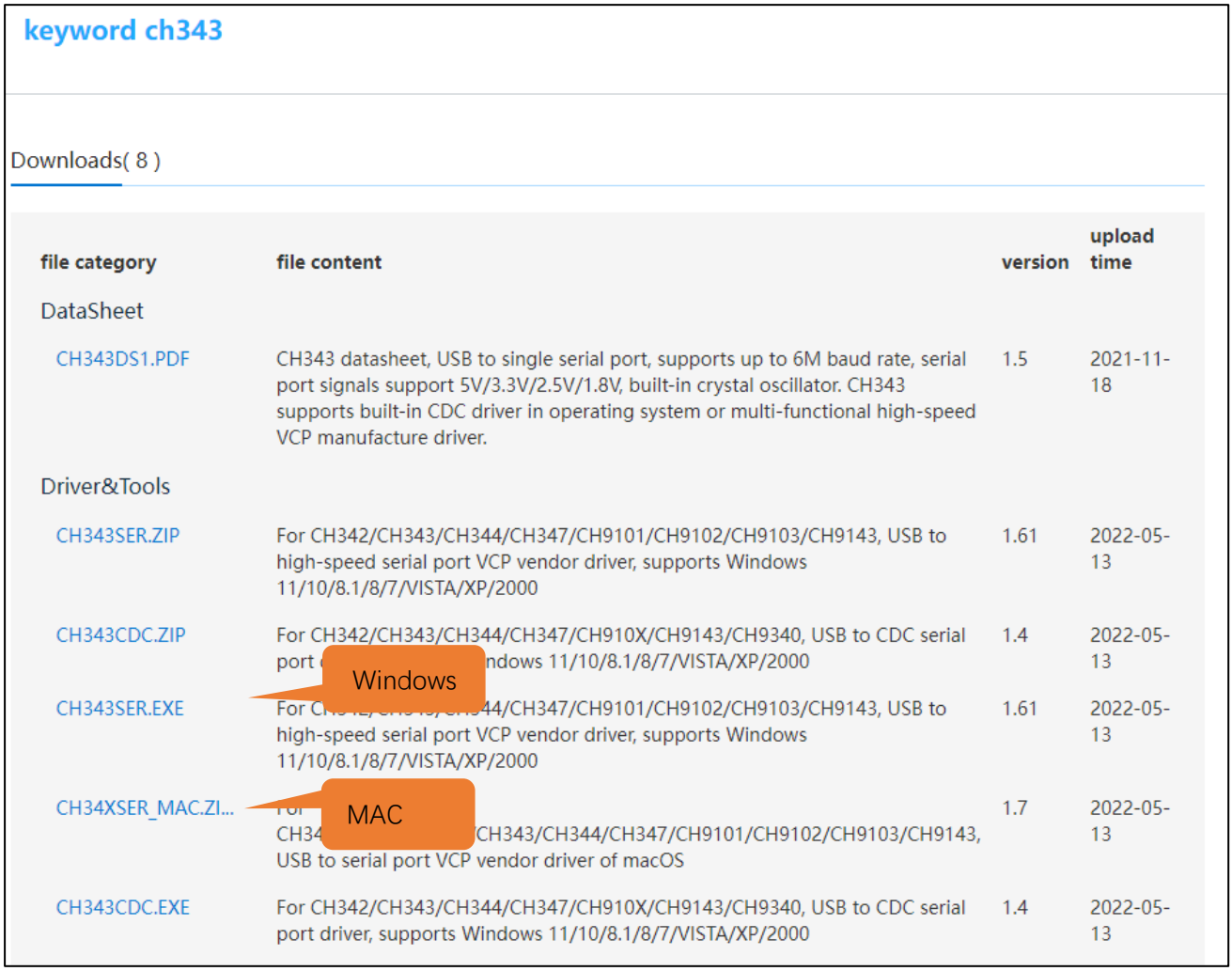
6. When ESP32-S3 WROOM is connected to computer, select "This PC", right-click to select "Manage" and click "Device Manager" in the newly pop-up dialog box, and you can see the following interface.



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

MAC

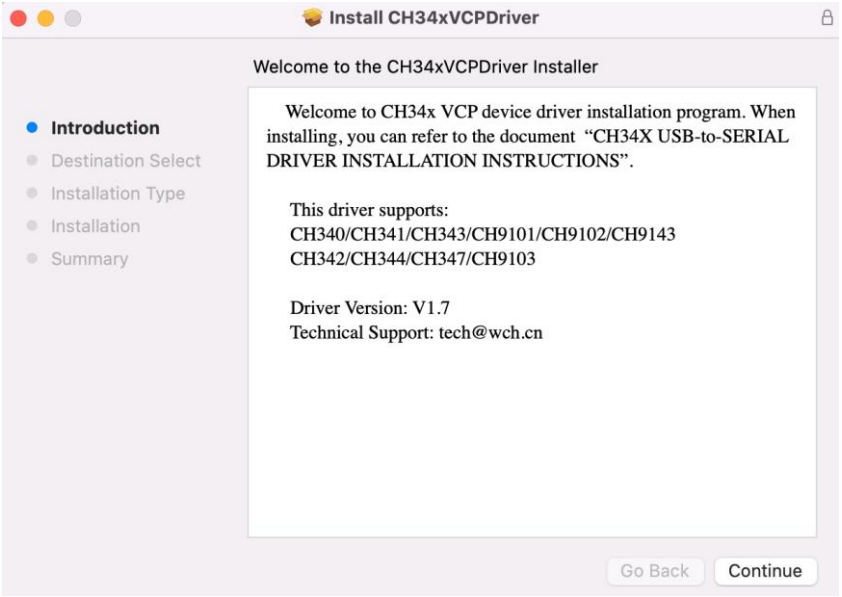
First, download CH343 driver. Click http://www.wch-ic.com/search?t=all&q=ch343 to download the appropriate one based on your operating system.



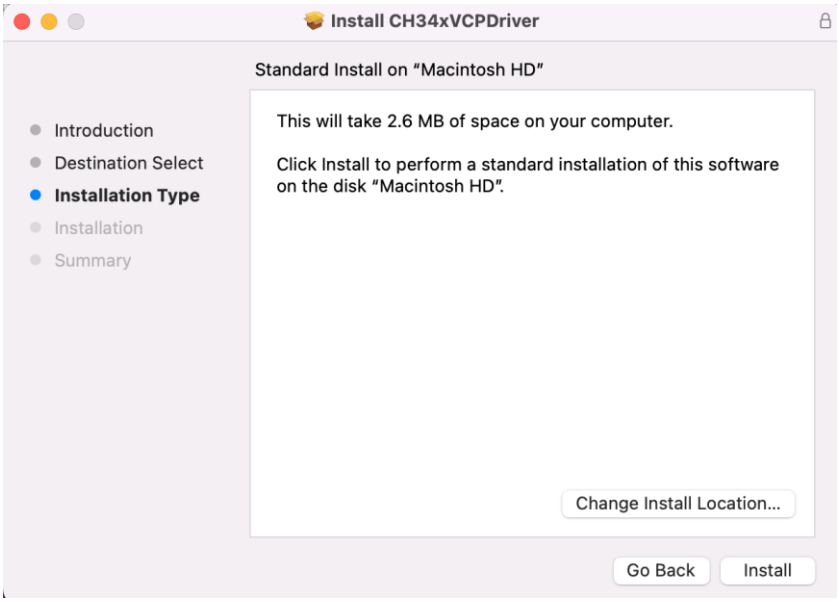
If you would not like to download the installation package, you can open **"Freenove_Media_Kit_for_ESP32-S3/CH343"**. We have prepared the installation package.

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

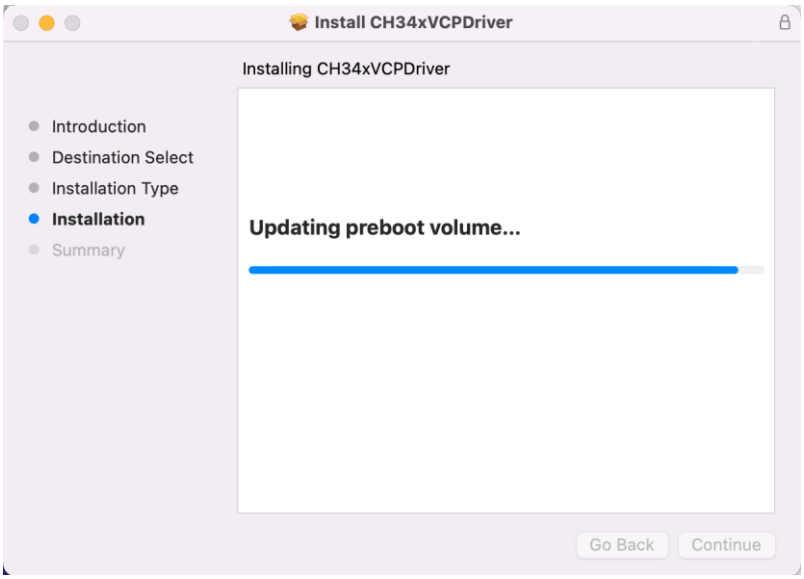
Third, click Continue.



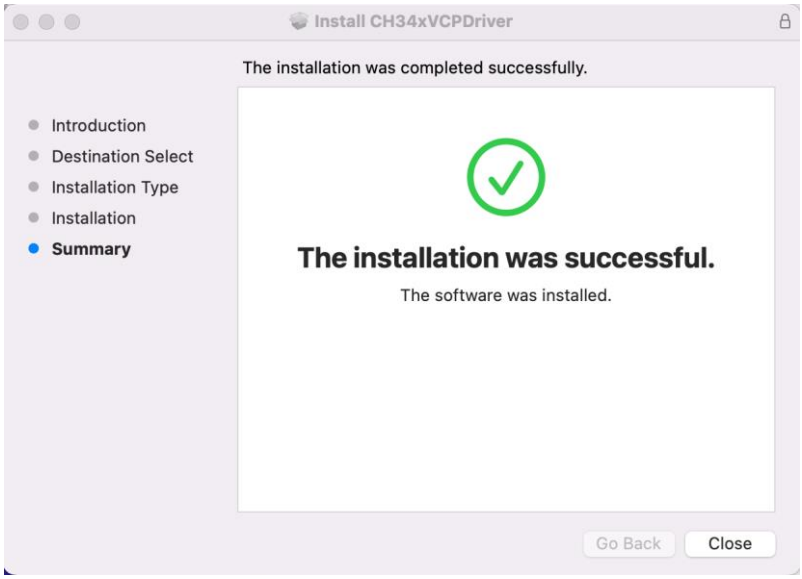
Fourth, click Install.



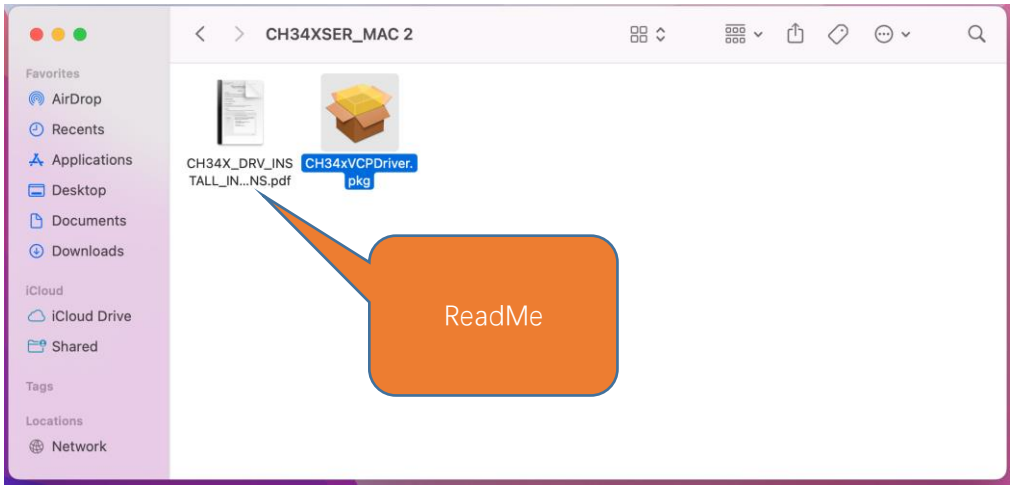
Then, waiting Finsh.



Finally, restart your PC.

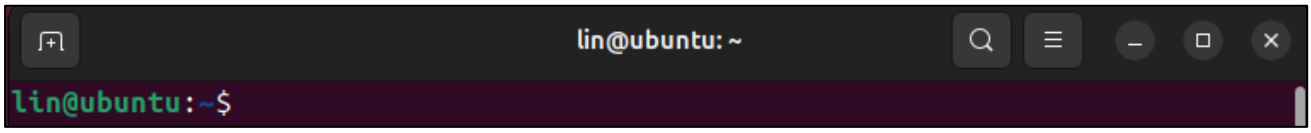


If it fails to be installed with the above steps, you can refer to readme.pdf to install it.



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
```

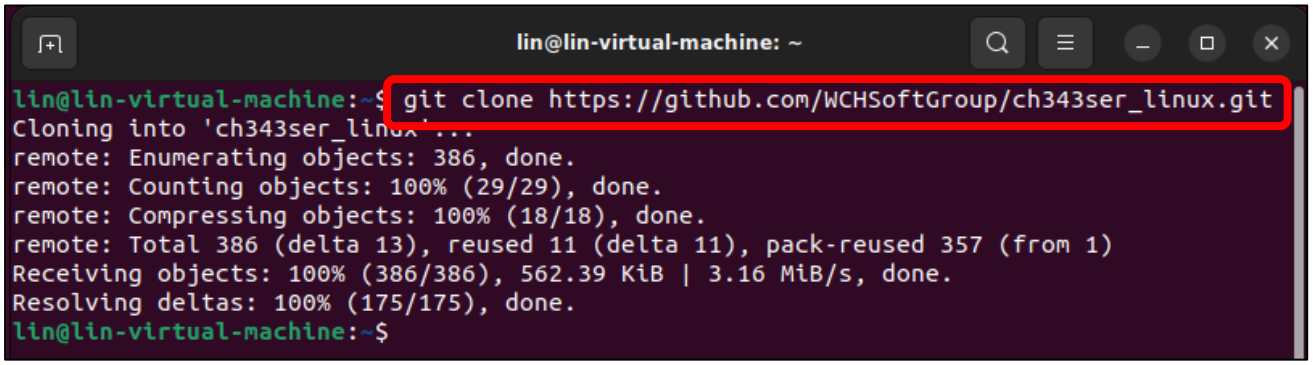
```
lin@lin-virtual-machine:~$ ls /dev/tty*
/dev/tty    /dev/tty23  /dev/tty39  /dev/tty54      /dev/ttyS1   /dev/ttyS25
/dev/tty0   /dev/tty24  /dev/tty4   /dev/tty55      /dev/ttyS10  /dev/ttyS26
/dev/tty1   /dev/tty25  /dev/tty40  /dev/tty56      /dev/ttyS11  /dev/ttyS27
/dev/tty10  /dev/tty26  /dev/tty41  /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 recognise the ESP32S3's port, you can do as follows to install the ch343 driver.

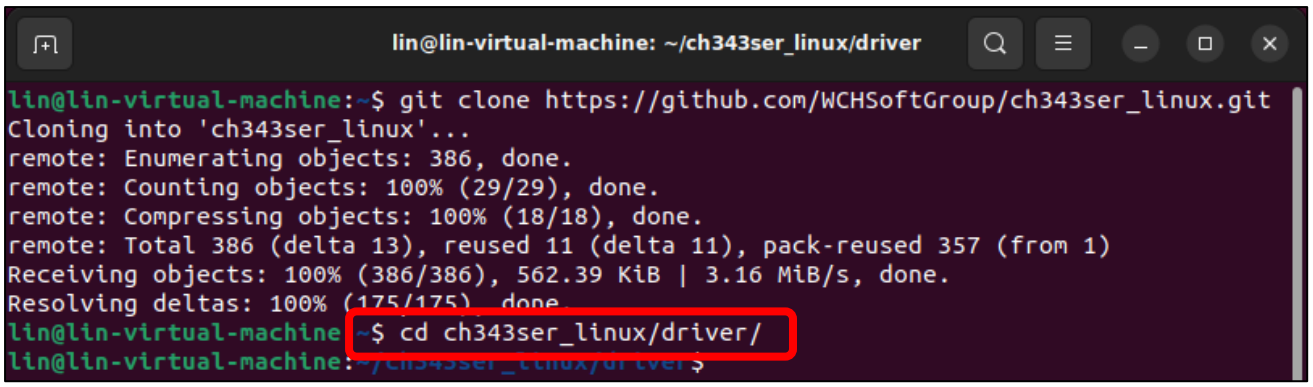
Install the CH343 driver with the following command.

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



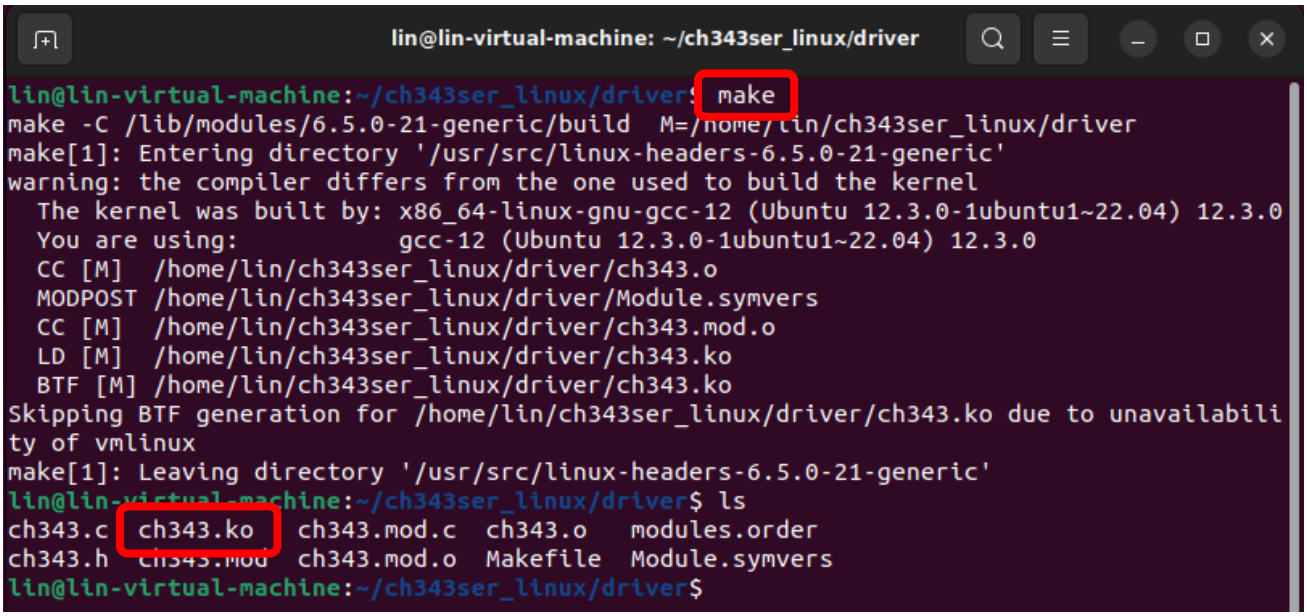
Enter the folder.

```
cd ch343ser_linux/driver/
```



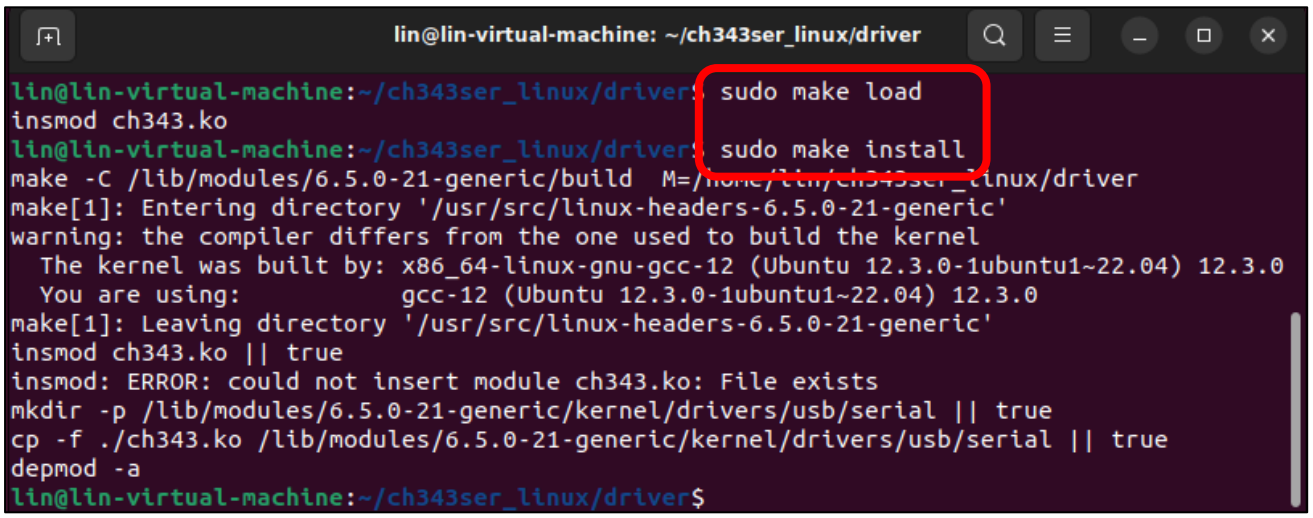
Compile and generate the ch343.ko file.

```
make
```



Load the generated file to the system.

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

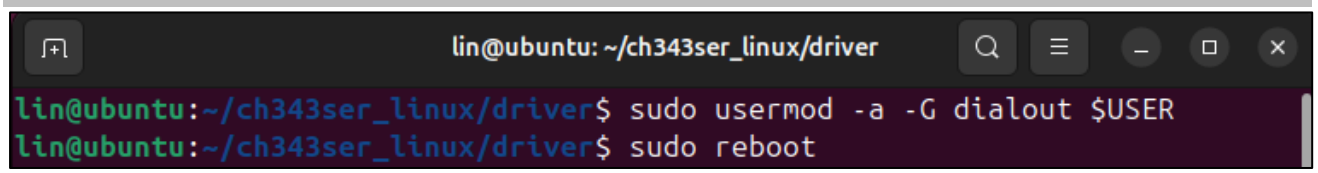


Connect the ESP32S3 to your computer, check the port with the following command and you should see the port.

```
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
```

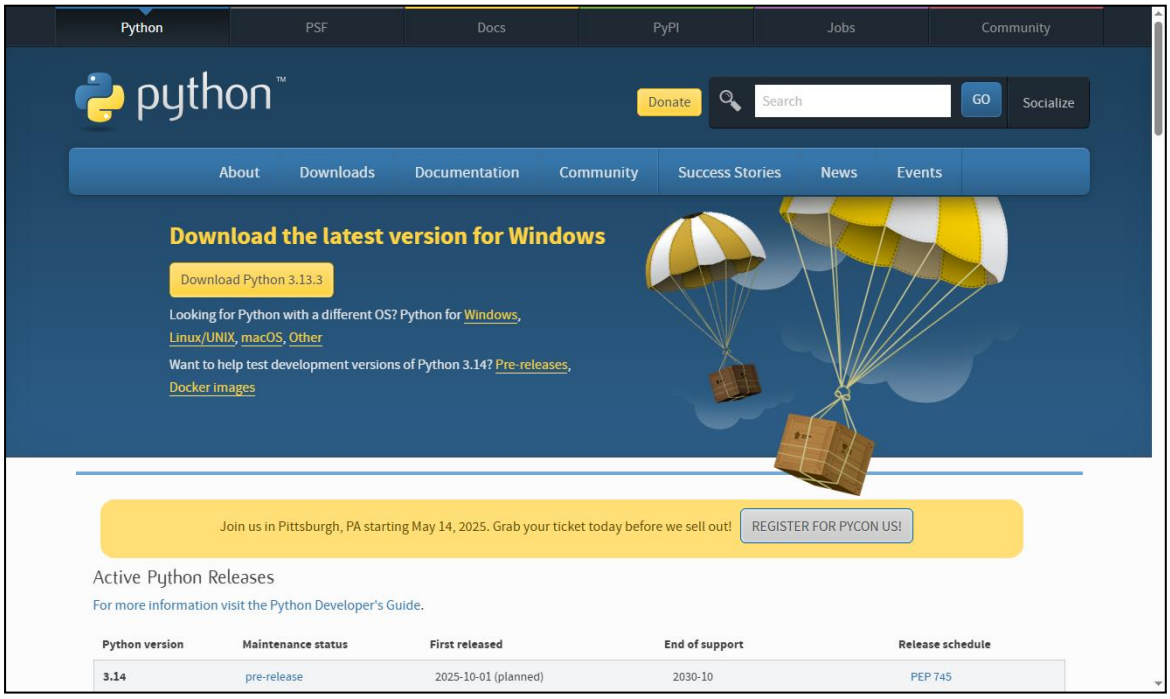


Please note that the configure takes effect after rebooting.

Installing Python (Required)

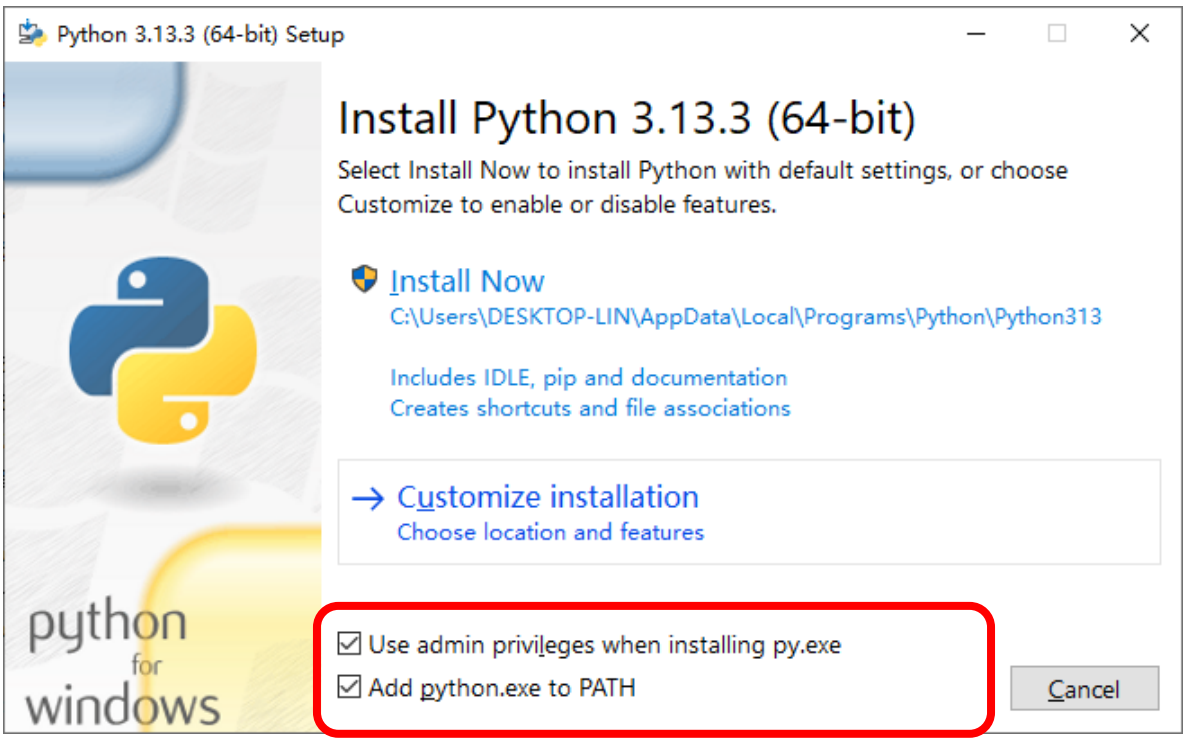
Windows

Download and install Python3 package.
https://www.python.org/downloads/windows/

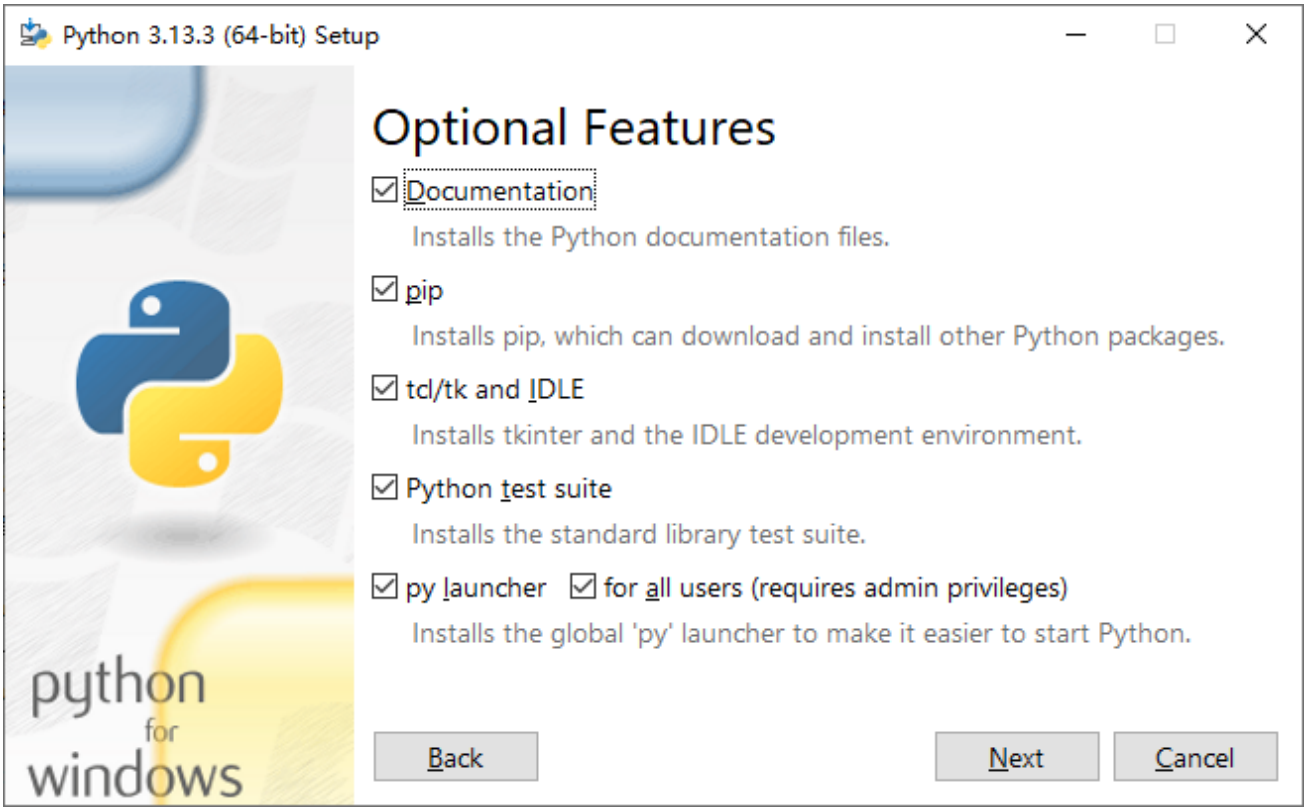


Click Download Python 3.13.3

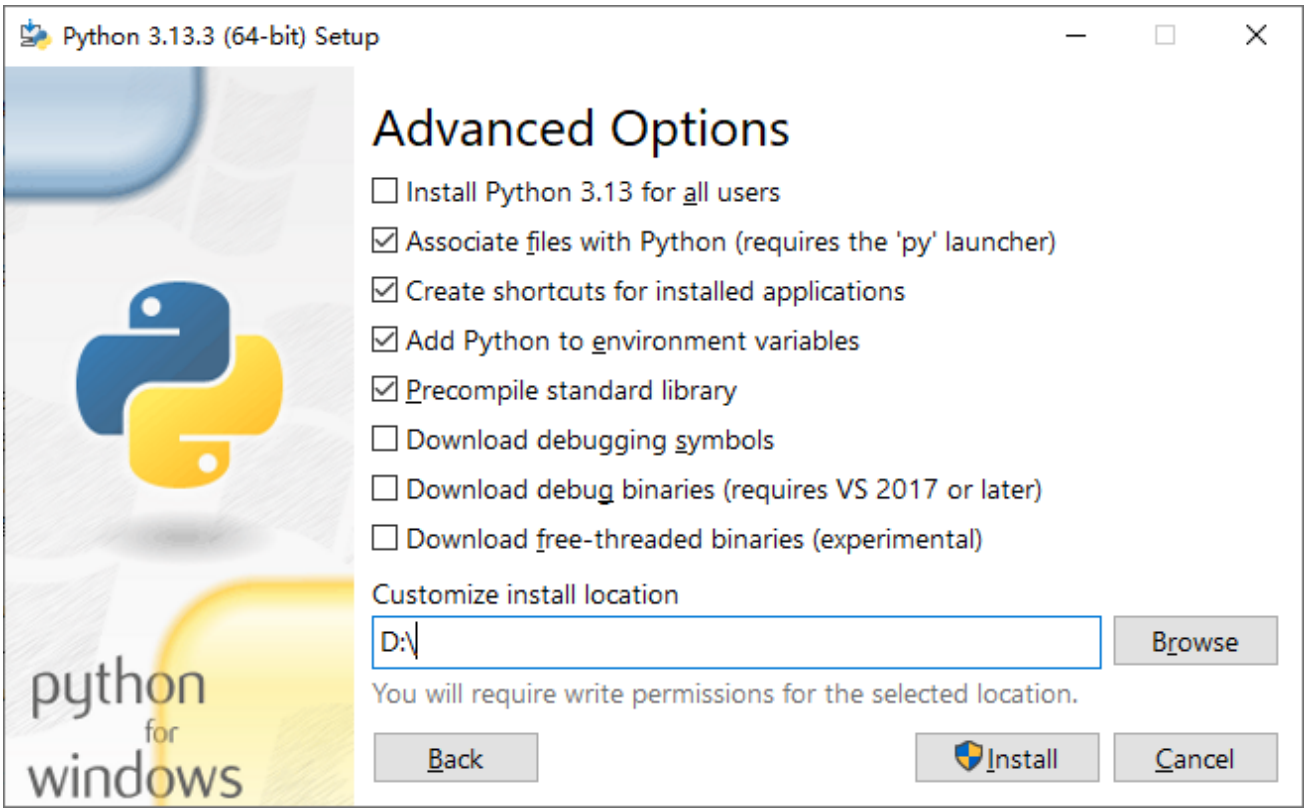
Please note that "Add Python 3.13 to PATH" MUST be check.



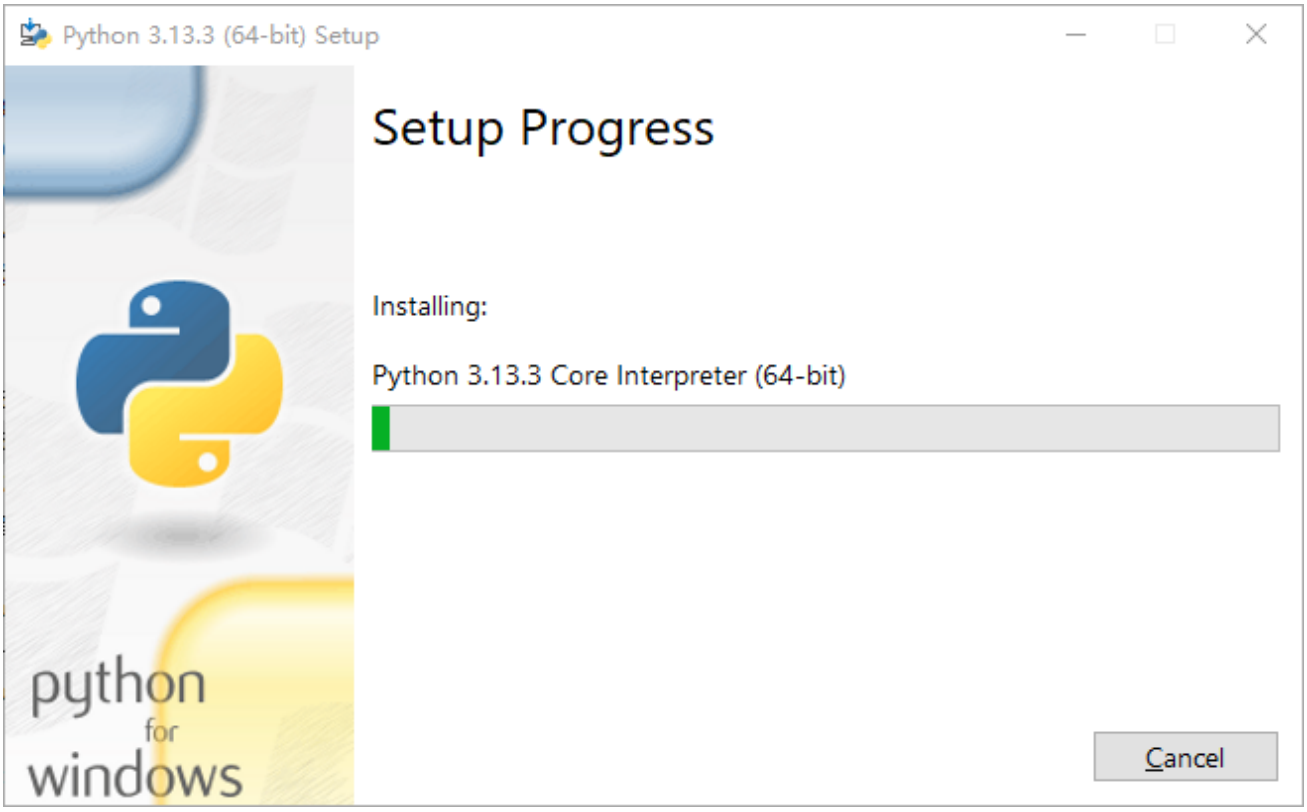
Check all the options and then click "Next".



Here you can select the installation path of Python. We install it at D drive. If you are a novice, you can select the default path.



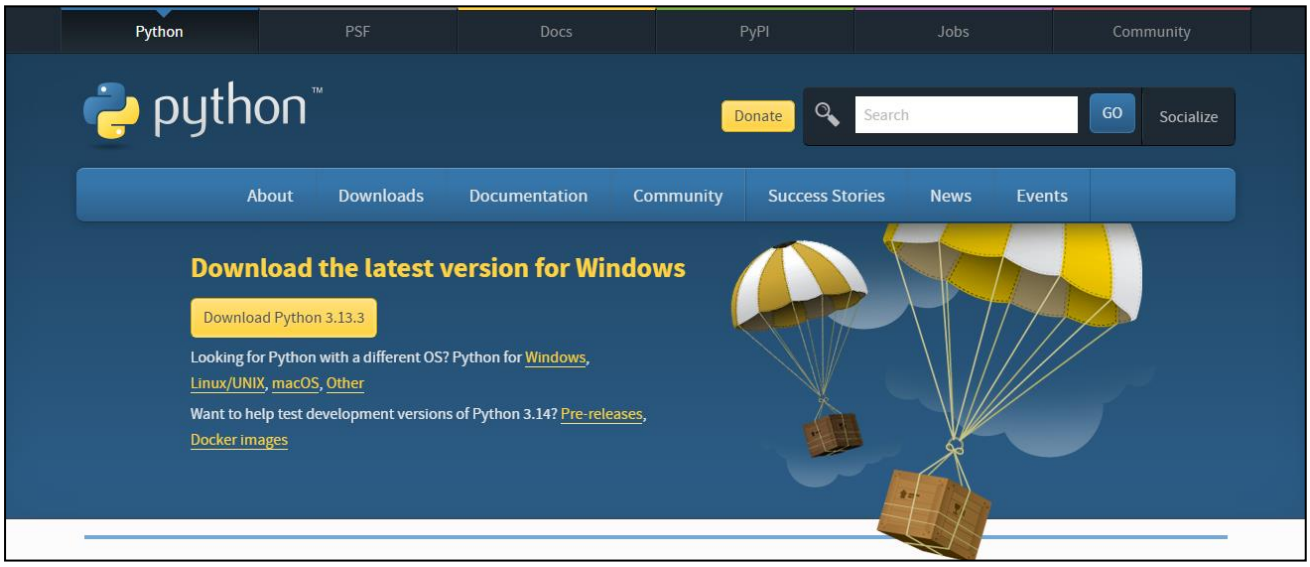
Wait for it to finish installing.



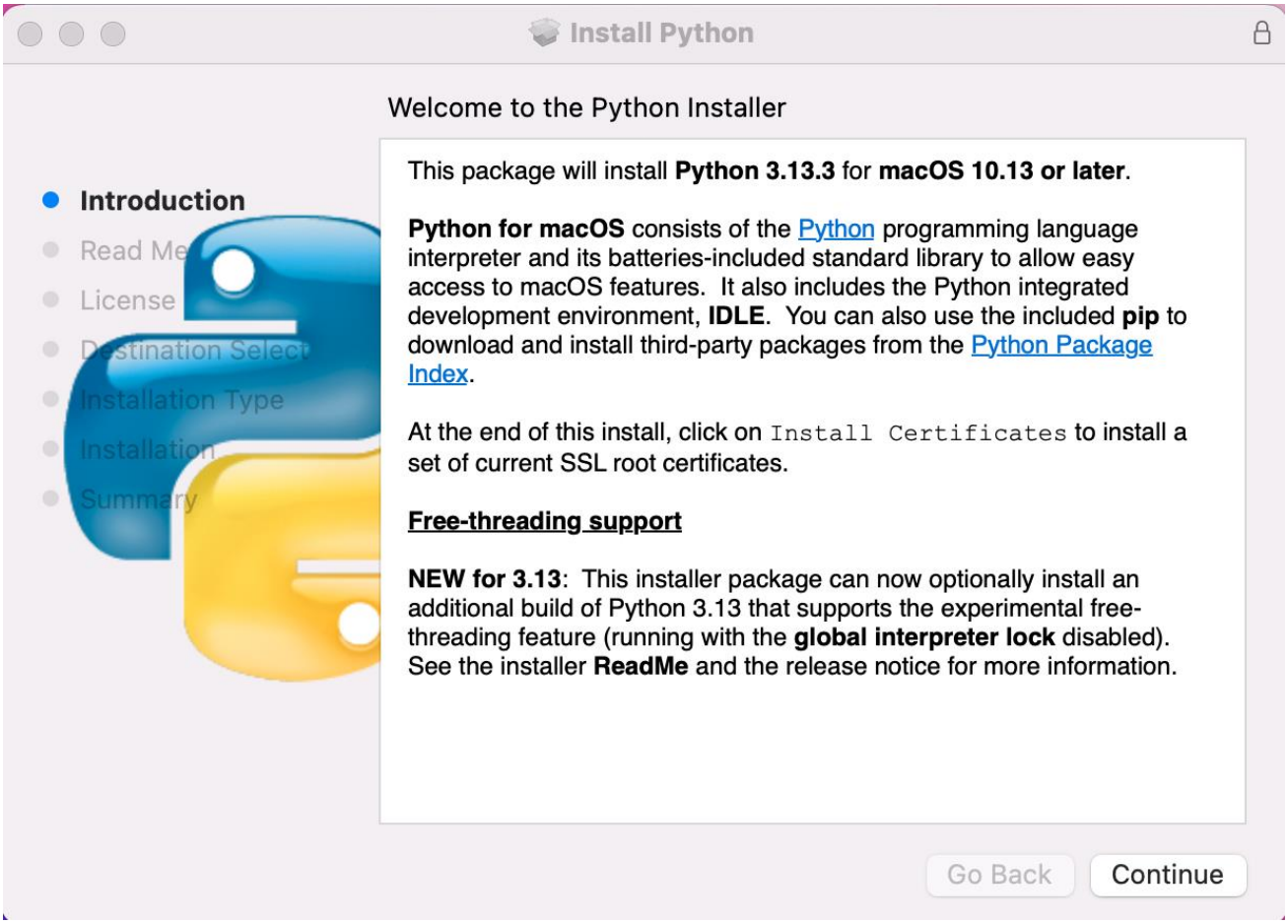
Now the installation is finished.

MAC

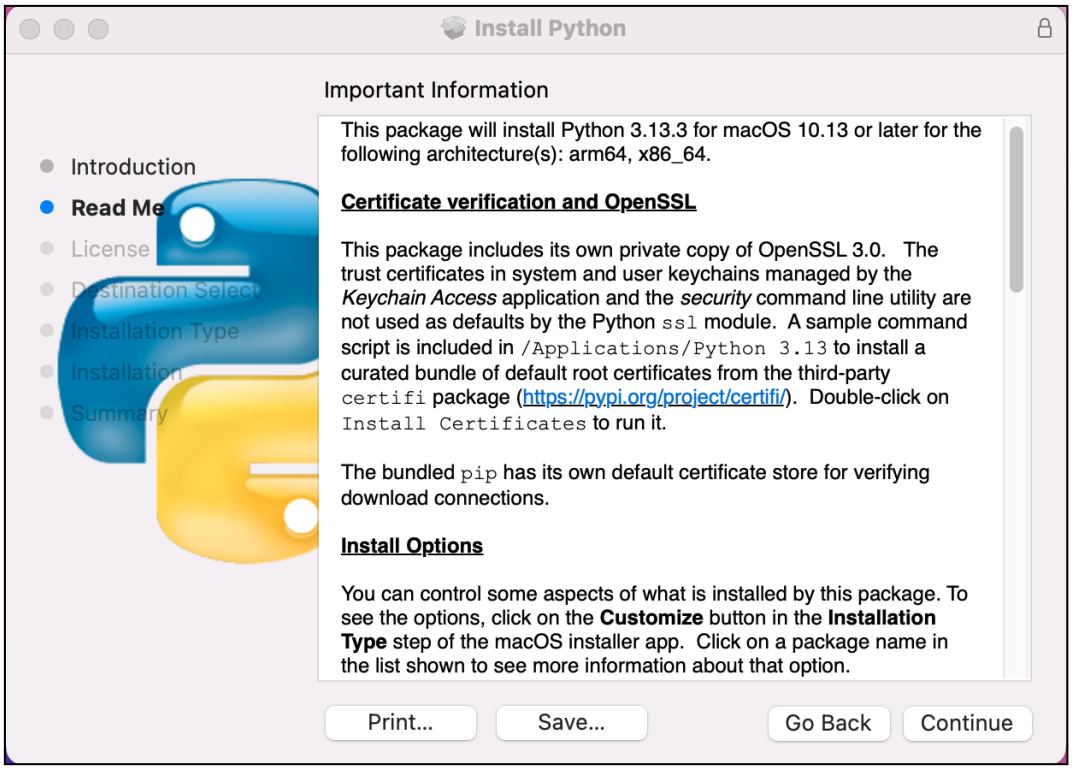
Download installation package, link: https://www.python.org/downloads/
Click Download Python 3.13.3



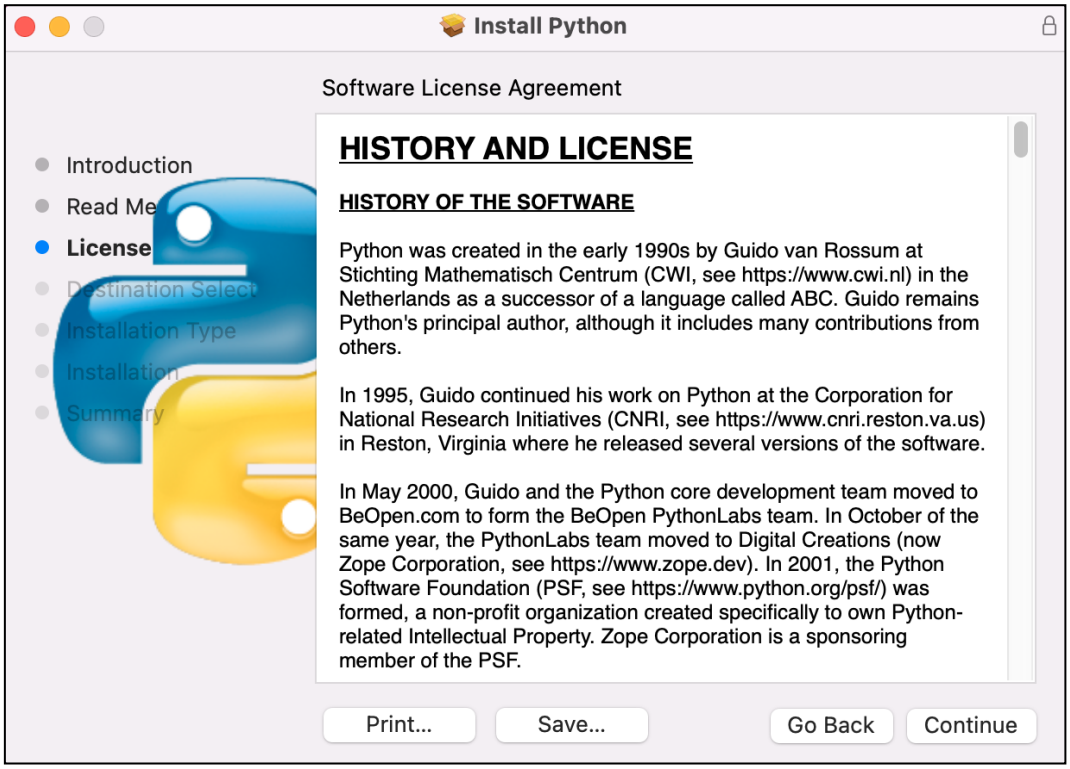
Run the downloaded installation package. Click Continue



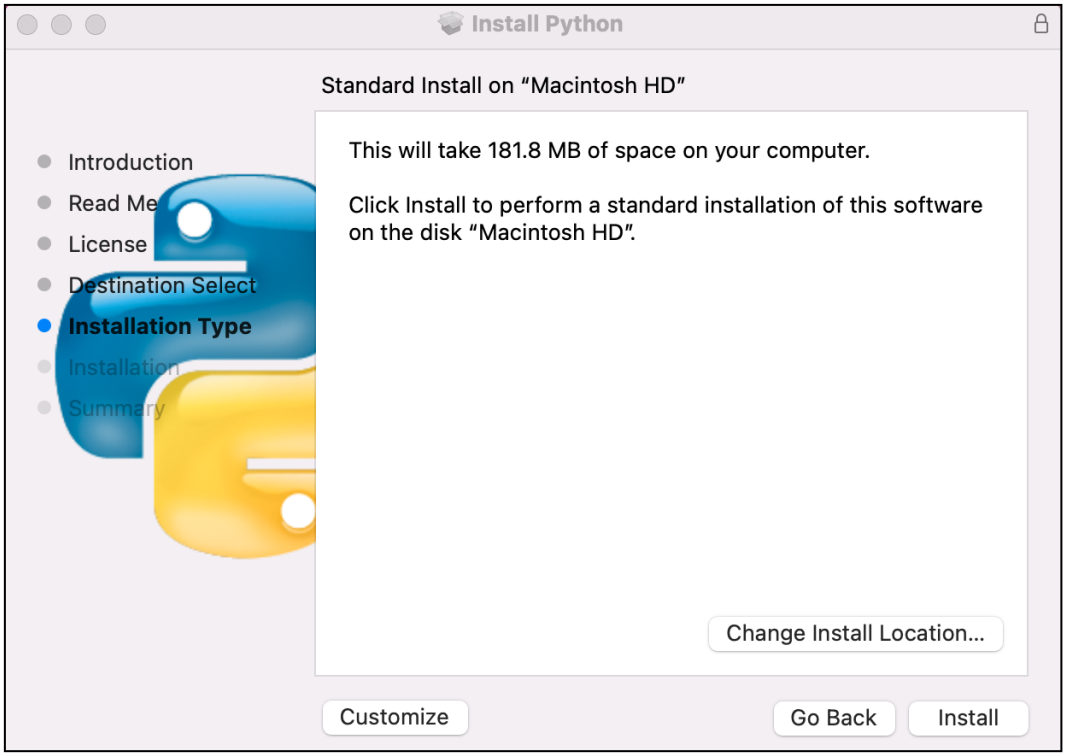
Click Continue



Click Continue



Click Install. If your computer has a password, enter the password and Install Software.



Now the installation succeeds.

Linux

Check whether Python3 is installed in the system.

python –version

python3 –version

```
lin@ubuntu:~$ python --version
python: command not found
lin@ubuntu:~$ python3 --version
bash: /usr/bin/python3: No such file or directory
lin@ubuntu:~$
```

If it has not been installed, do it by running the following command. This will install the latest version by default.

sudo apt install python3

```
lin@ubuntu: ~
python: command not found
lin@ubuntu:~$ python3 --version
python3: command not found
lin@ubuntu:~$ sudo apt install python3
Installing:
  python3
```

Link python to python3.

sudo rm /usr/bin/python

sudo ln -s /usr/bin/python3 /usr/bin/python

```
lin@ubuntu: ~
lin@ubuntu:~$ sudo rm /usr/bin/python
lin@ubuntu:~$ sudo ln -s /usr/bin/python3 /usr/bin/python
lin@ubuntu:~$ python --version
Python 3.13.3
```

Install python3.13-venv virtual environment.

sudo apt install python3-venv

```
lin@ubuntu: ~/Upload_Xiaozhi_Bin
lin@ubuntu:~/Upload_Xiaozhi_Bin$ sudo apt install python3-venv
```

Install python3-pip.

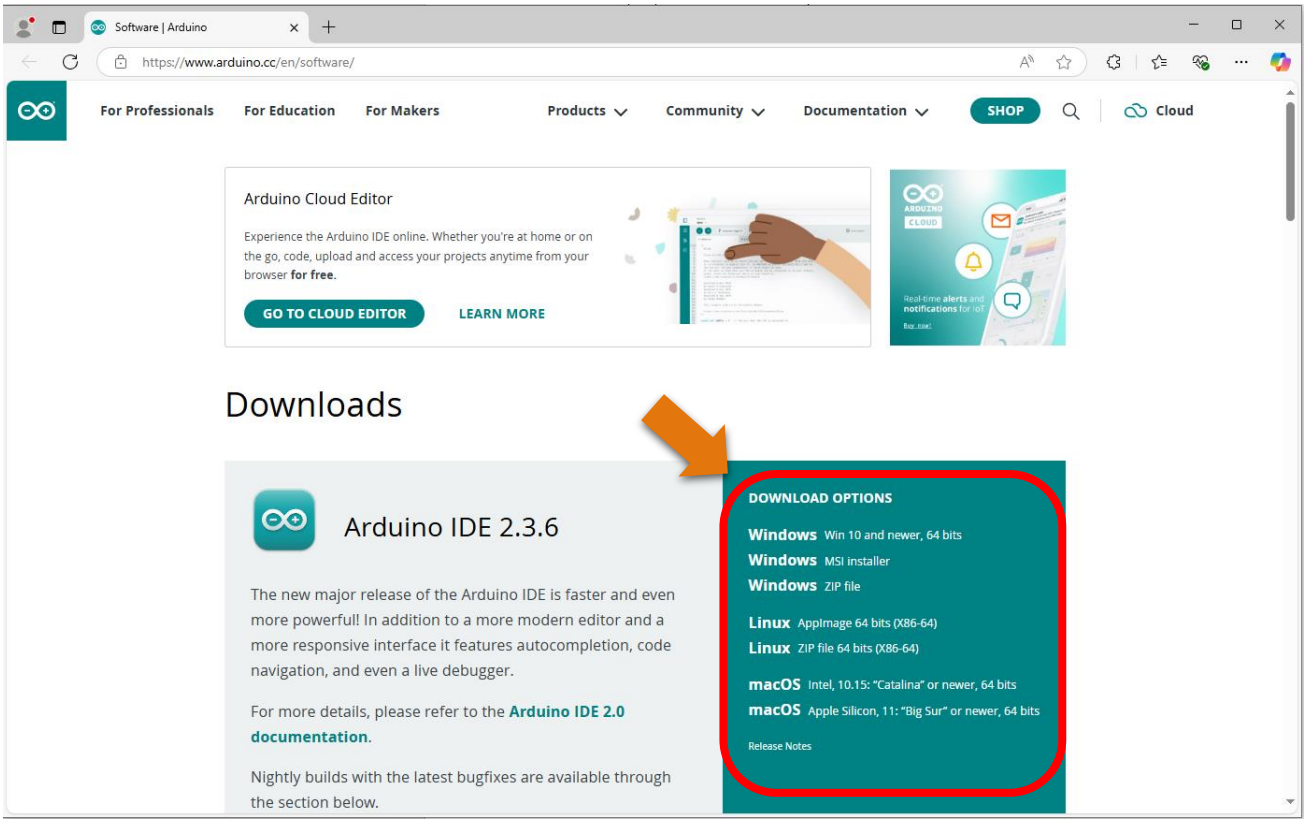
sudo apt install python3-pip

```
lin@ubuntu: ~
lin@ubuntu:~$ sudo apt install python3-pip
python3-pip is already the newest version (25.0+dfsg-1).
Summary:
  Upgrading: 0, Installing: 0, Removing: 0, Not Upgrading: 0
lin@ubuntu:~$ pip --version
pip 25.0 from /usr/lib/python3/dist-packages/pip (python 3.13)
lin@ubuntu:~$
```

Programming Software

Arduino Software (IDE) is used to write and upload the code for Arduino Board.

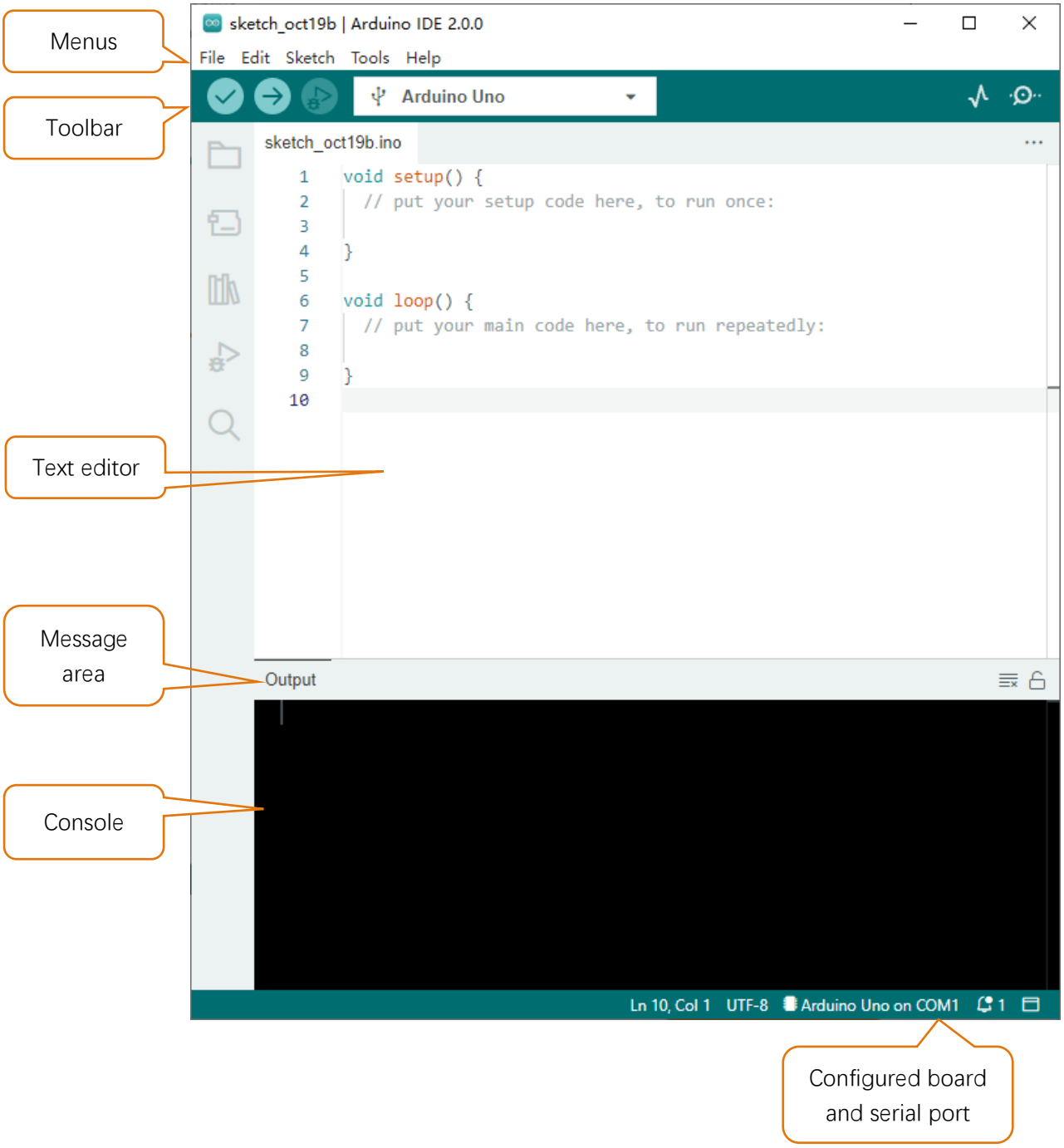
First, install Arduino Software (IDE): visit https://www.arduino.cc/en/software/, Select and download corresponding installer according to your operating system. If you are a Windows user, please select the "Windows" to download and install it correctly.



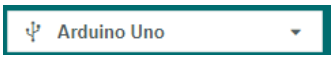
After the download completes, run the installer. For Windows users, there may pop up an installation dialog box of driver during the installation process. When it popes up, please allow the installation.

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:



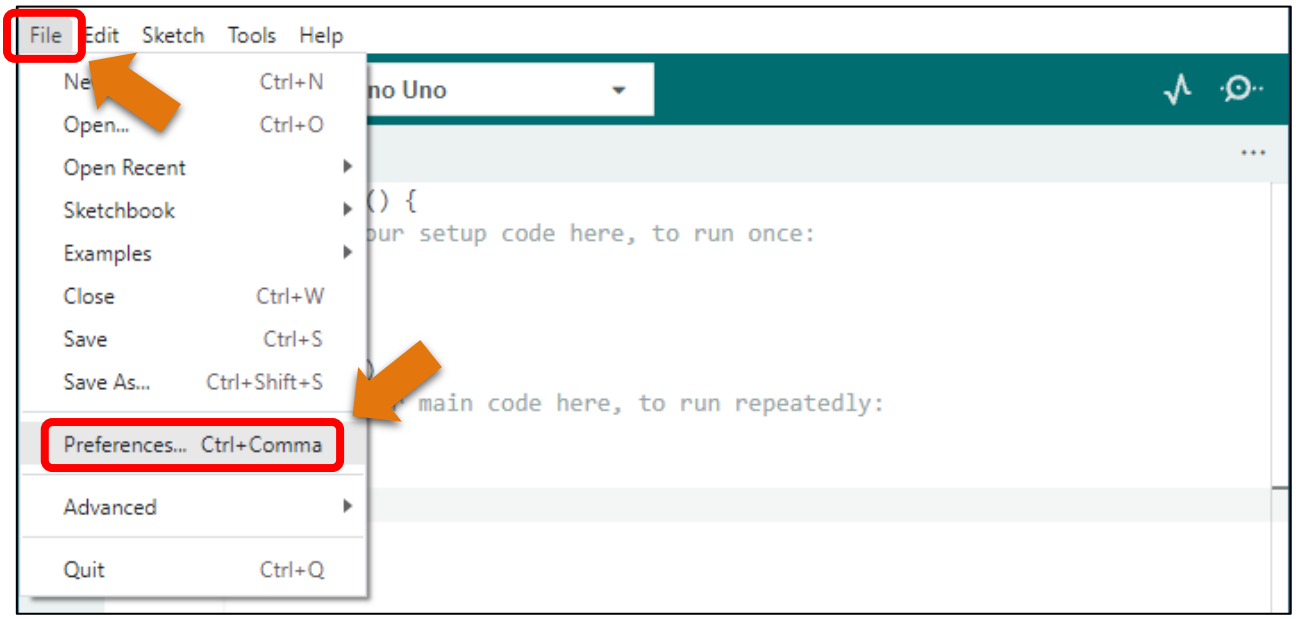
Programs written with Arduino Software (IDE) are called **sketches**. These sketches are written in the text editor and saved with the file extension**.ino**. The editor features text cutting/pasting and searching/replacing. The message area gives feedback while saving and exporting and also displays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

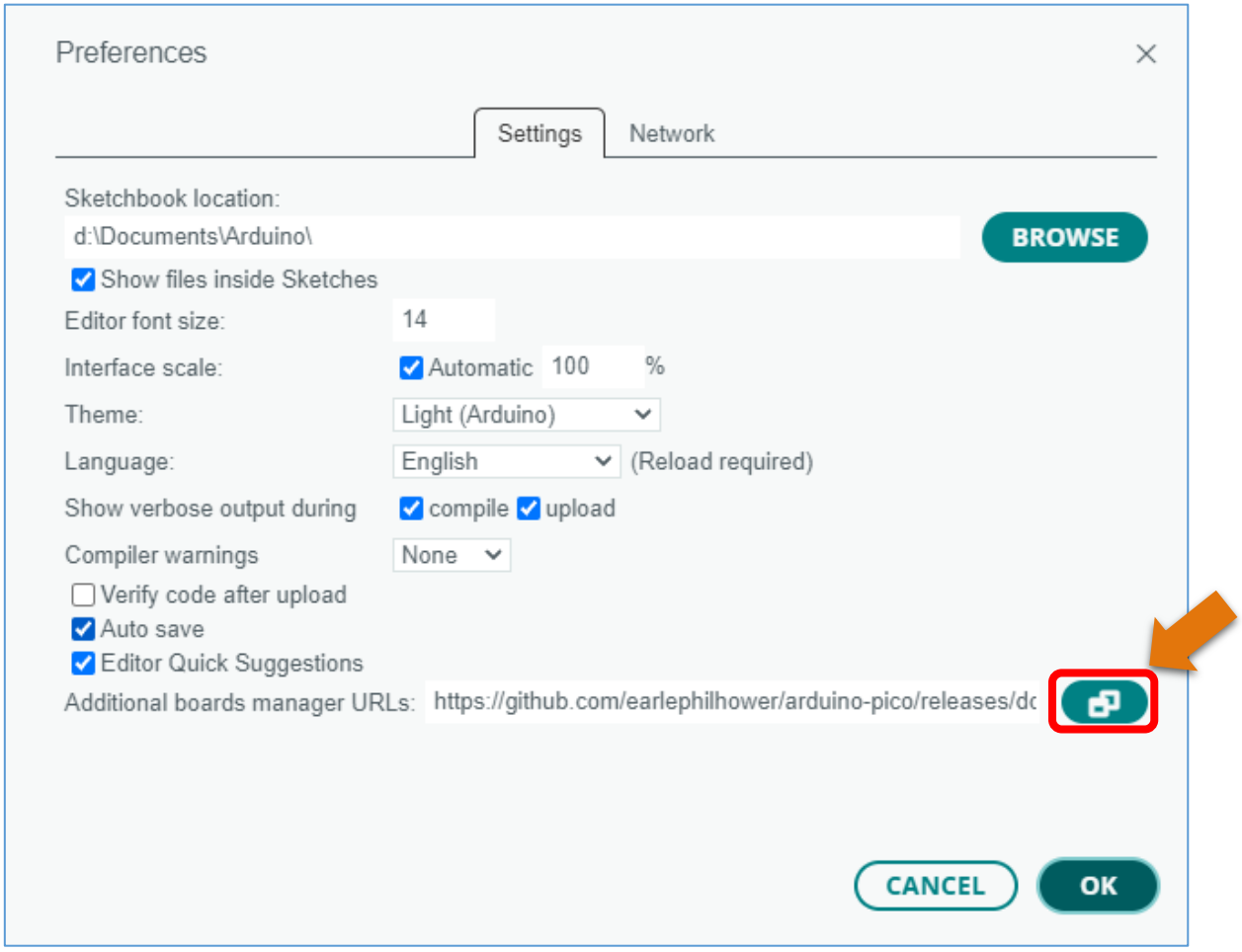
Additional commands are found within the five menus: File, Edit, Sketch, Tools, Help. The menus are context sensitive, which means only those items relevant to the work currently being carried out are available.

Environment Configuration

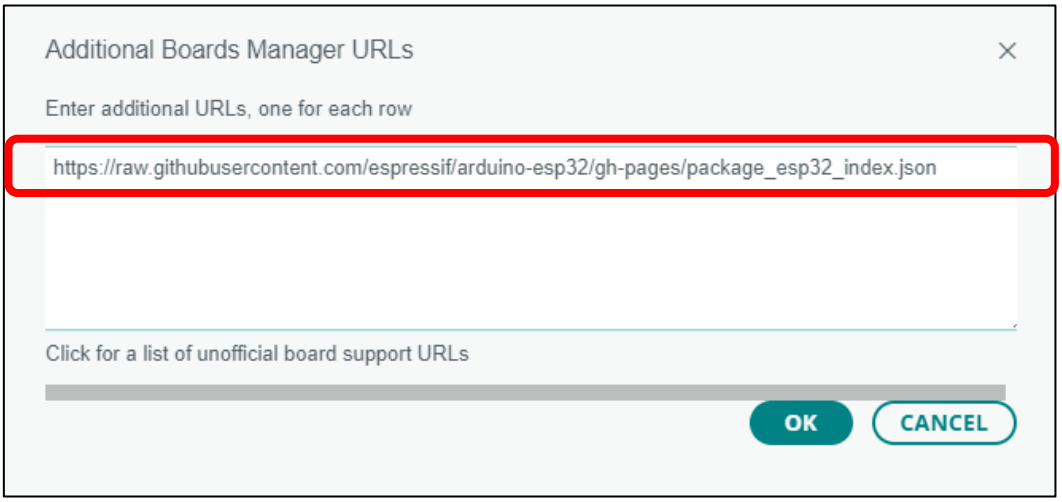
First, open the software platform arduino, and then click File in Menus and select Preferences.



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.

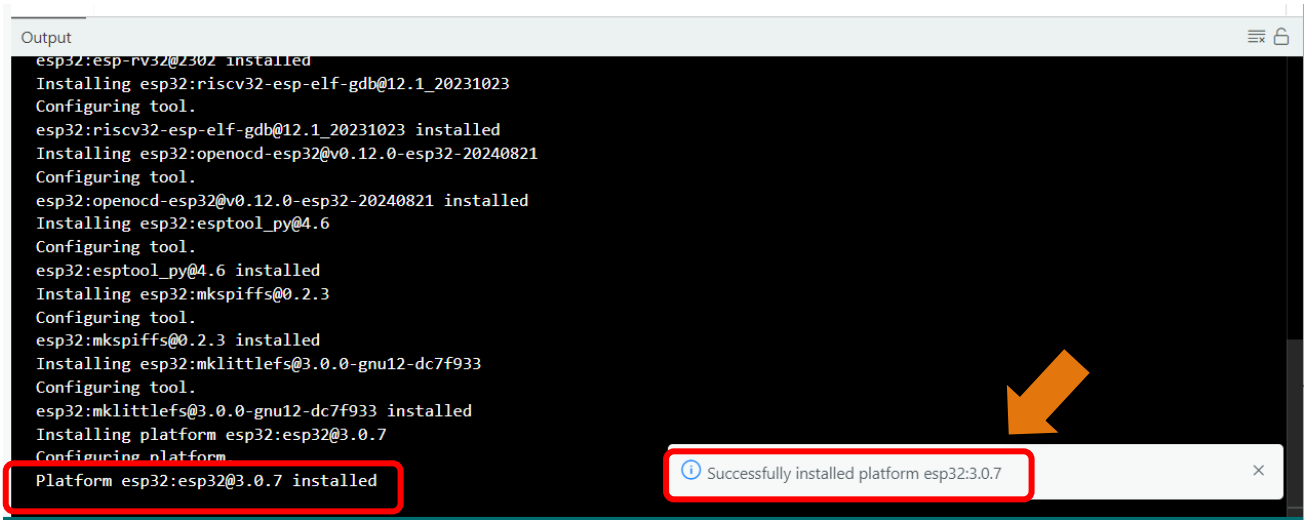


Note: if you copy and paste the URL directly, you may lose the "-". Please check carefully to make sure the link is correct.

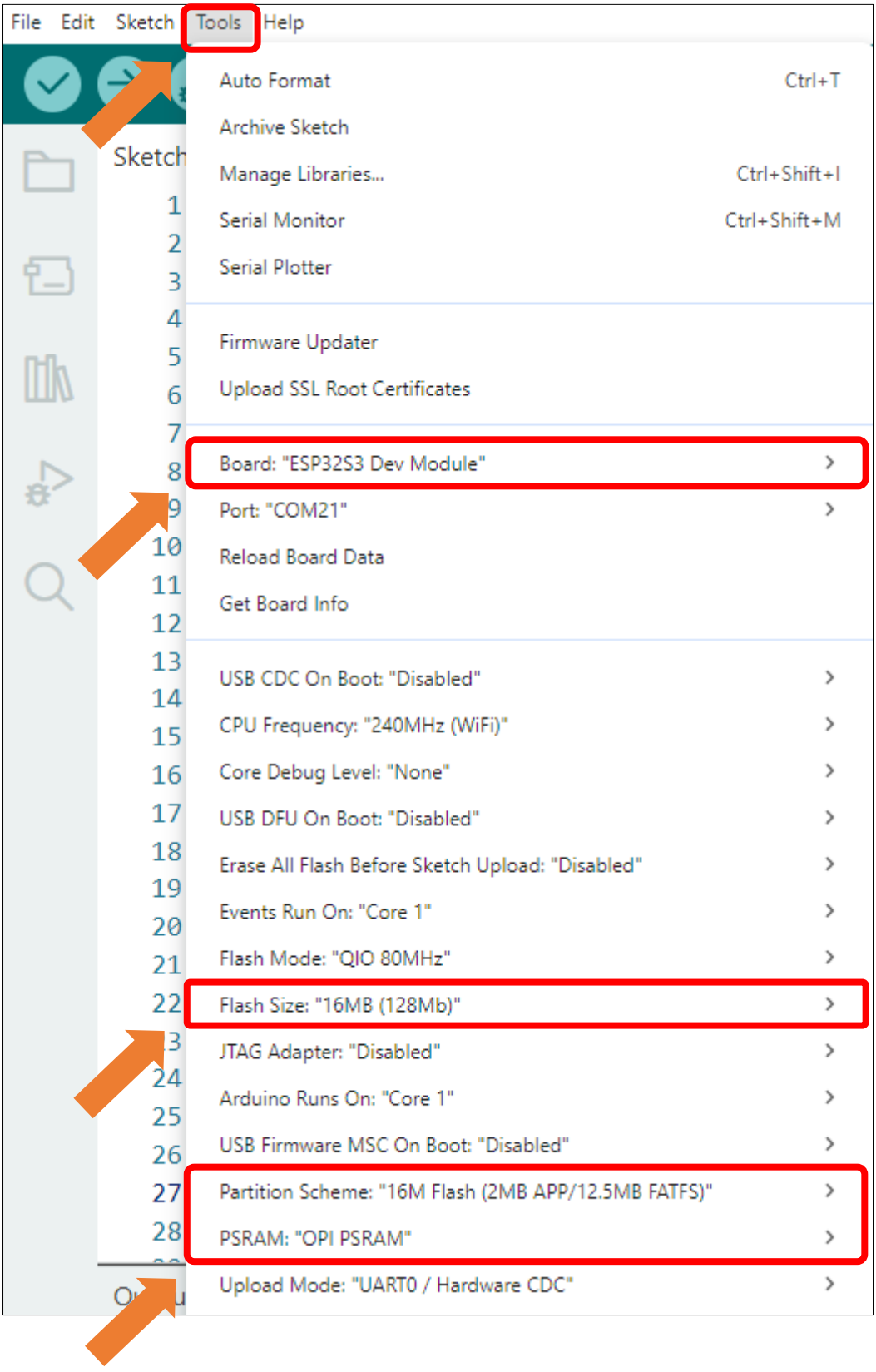
Fourth, click "Boards Manager". Enter “esp32” in Boards manager, select 3.2.0, and click “INSTALL”.

Note: Currently only version 3.0.7 is supported. Higer versions may lead to code running failure.

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



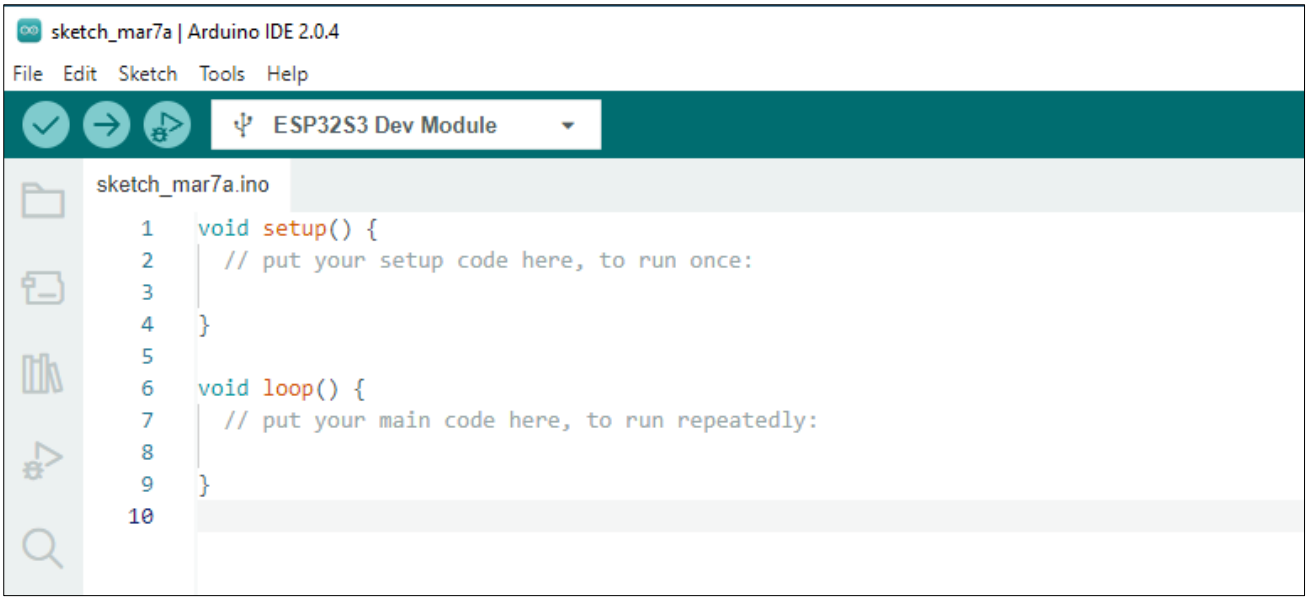
When finishing installation, click Tools in the Menus again and select Board: "ESP32S3 Dev Module", and then you can see information of ESP32-S3.



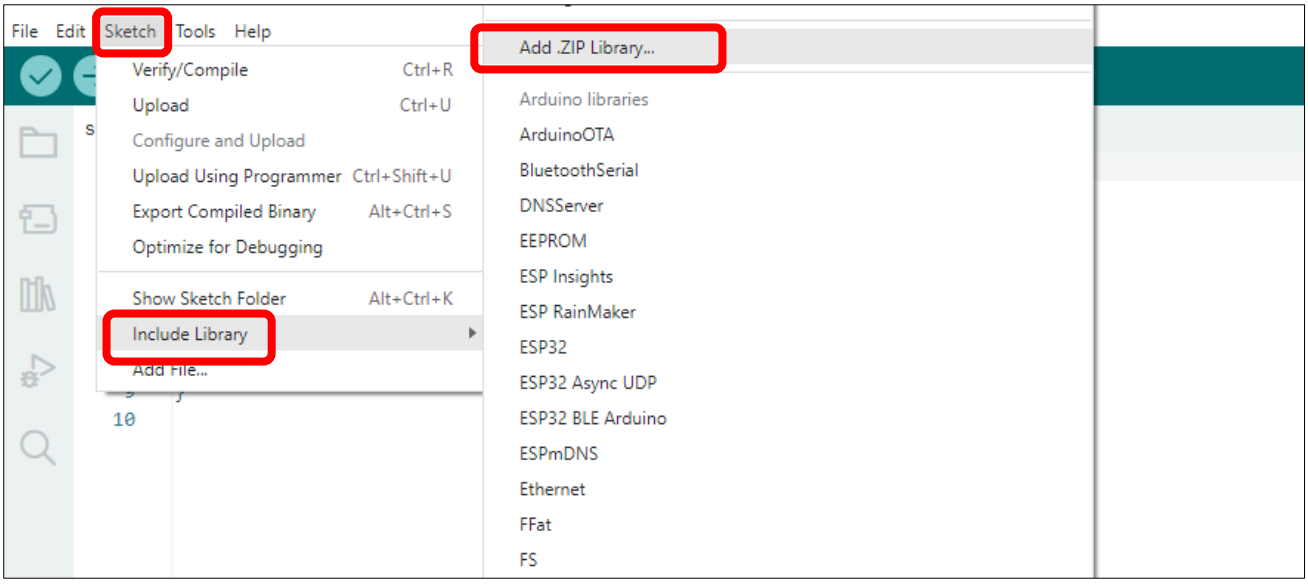
Library Installation

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_Media_Kit_for_ESP32-S3/Libraries** folder. Please refer to the following steps to install these libraries into the Arduino IDE.

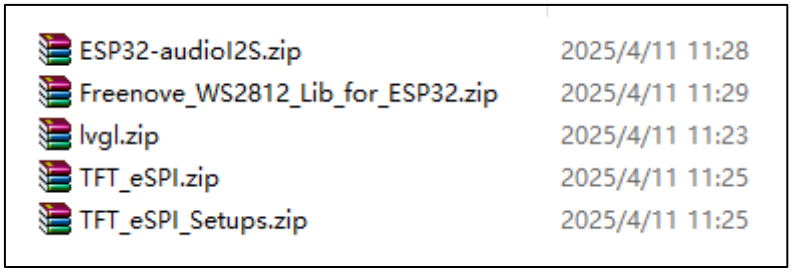
1. Open Arduino IDE.



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

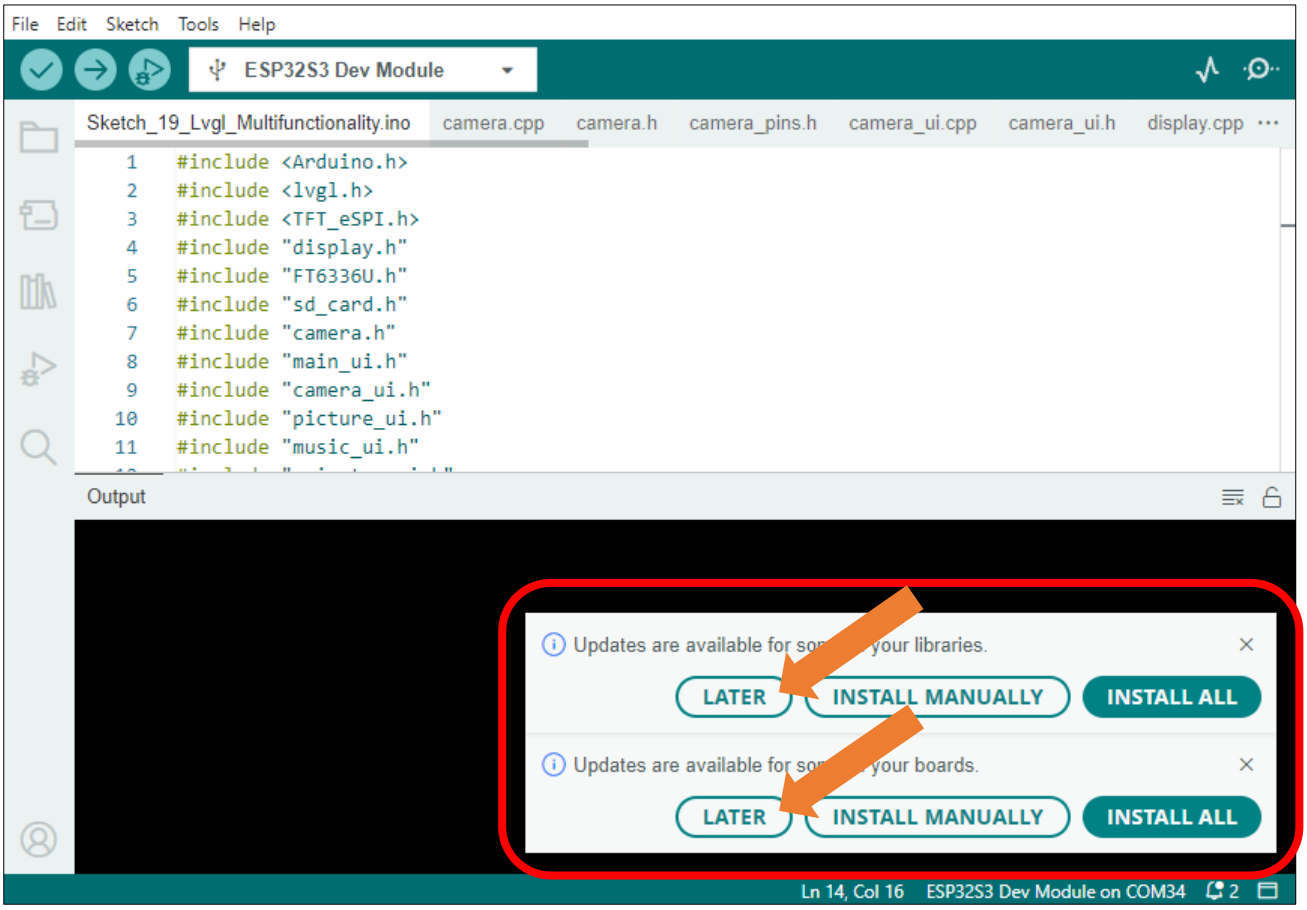


3. On the newly pop-up window, select the files from the **Freenove_Media_Kit_for_ESP32-S3/Libraries**. Click Open to install the library.



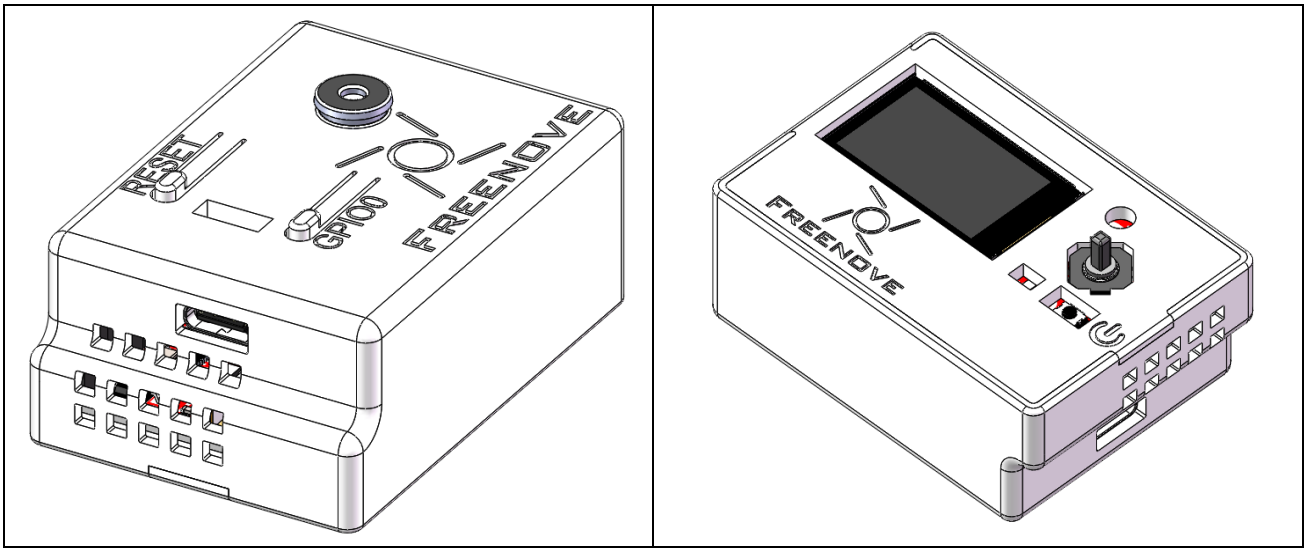
4. **Repeat the above steps until all the six libraries are installed to Arduino.** So far, all libraries have been installed.

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.



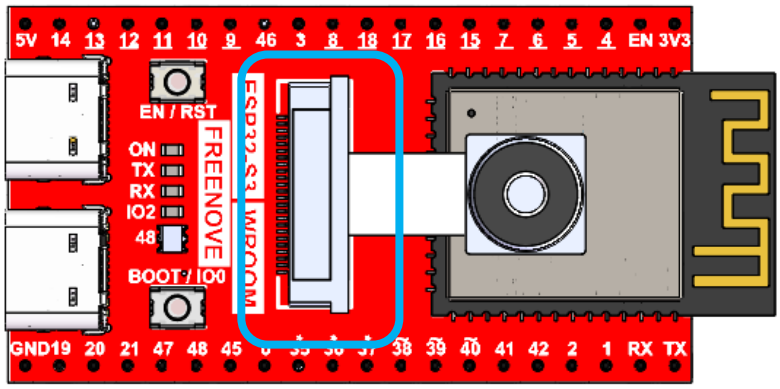
Chapter 0 Assembly

If the product you received is pre-assembled, you can skip this chapter.



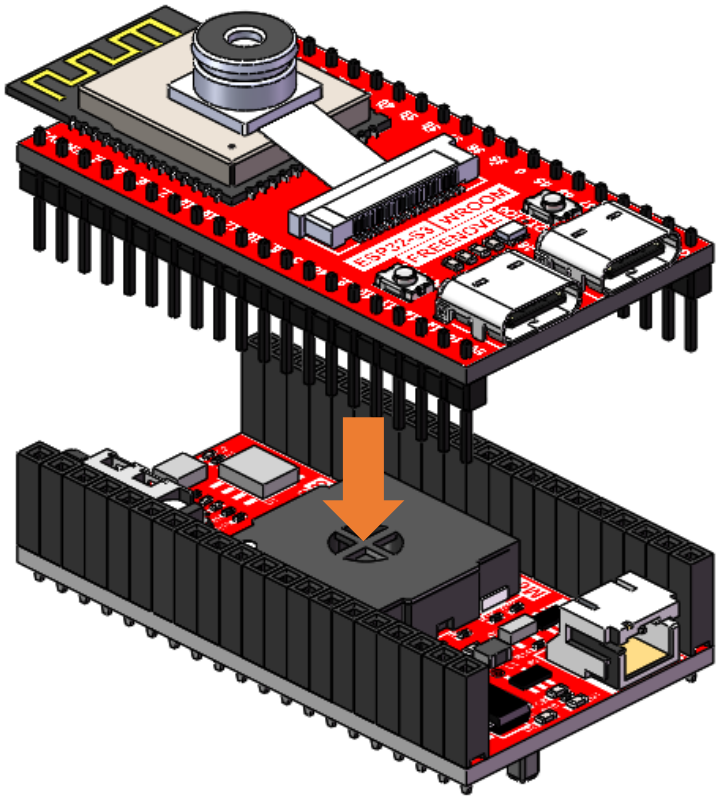
If you need to assemble it, please follow the following instructions step by step to proceed.

First, connect the camera to the board.



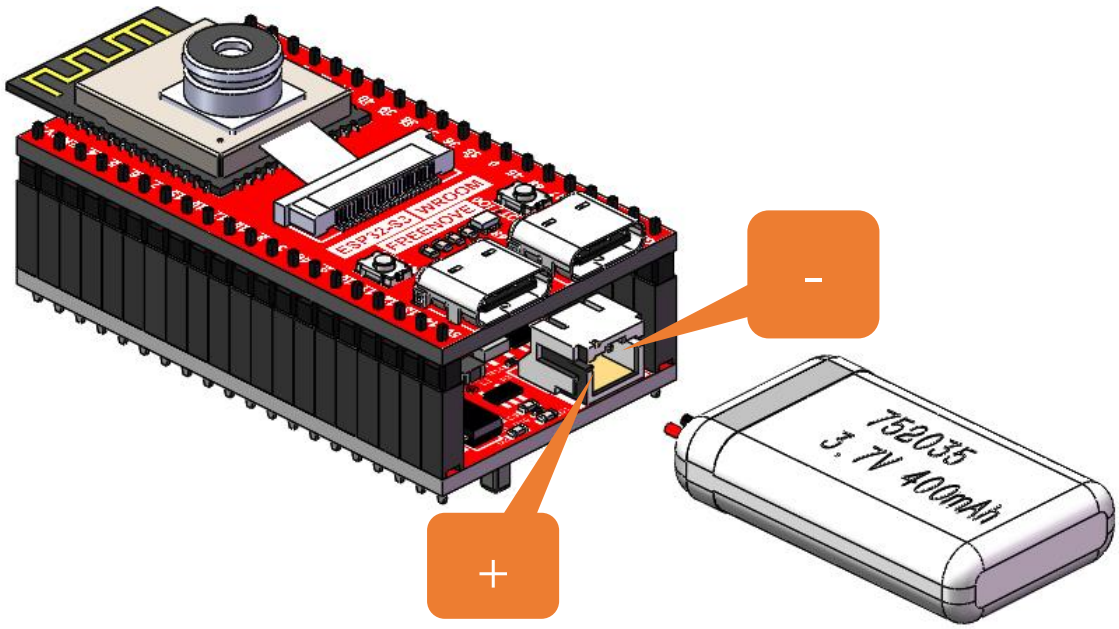
1. Do not remove or install the camera while power is on to avoids potential short circuits or damage during hot-plugging.
2. The camera socket features a flip-top design—gently lift the cover to install the camera. Avoid forceful handling.
3. Fully insert the camera into the socket until it reaches the end.

Plug the ESP32S3 onto the extension board.



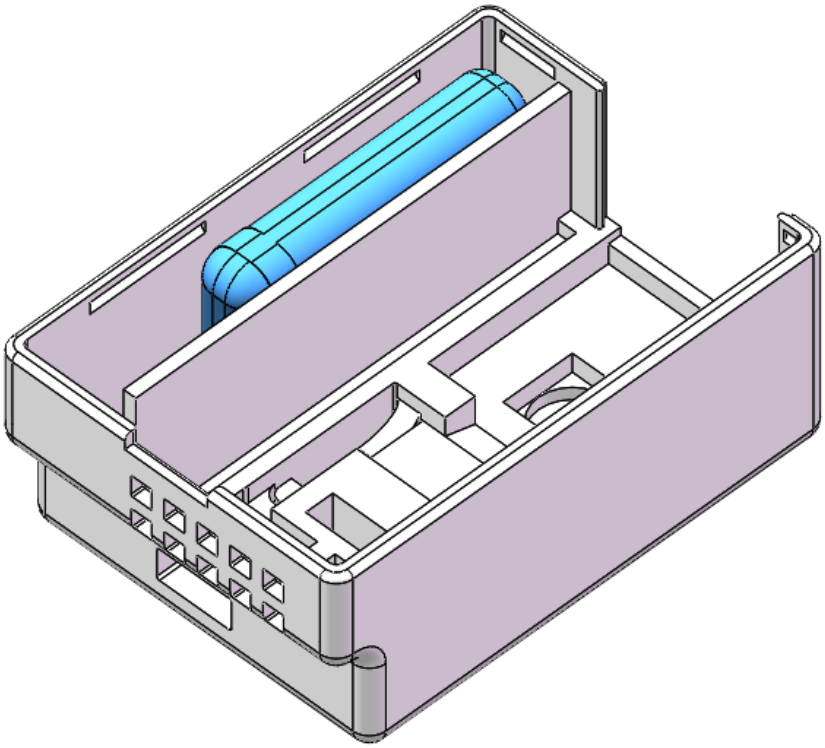
1. Pay attention to the orientation of the board. Installing reversely may damage the board.
2. Align the two board pin to pin.

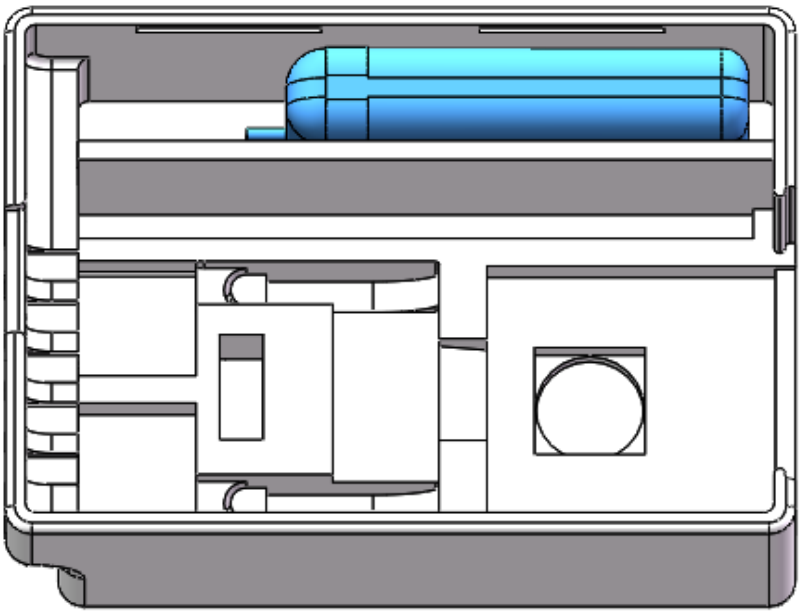
Connect the battery to the battery interface.
(Note: The battery is optional. You can also power the device via USB cable without installing a battery.)



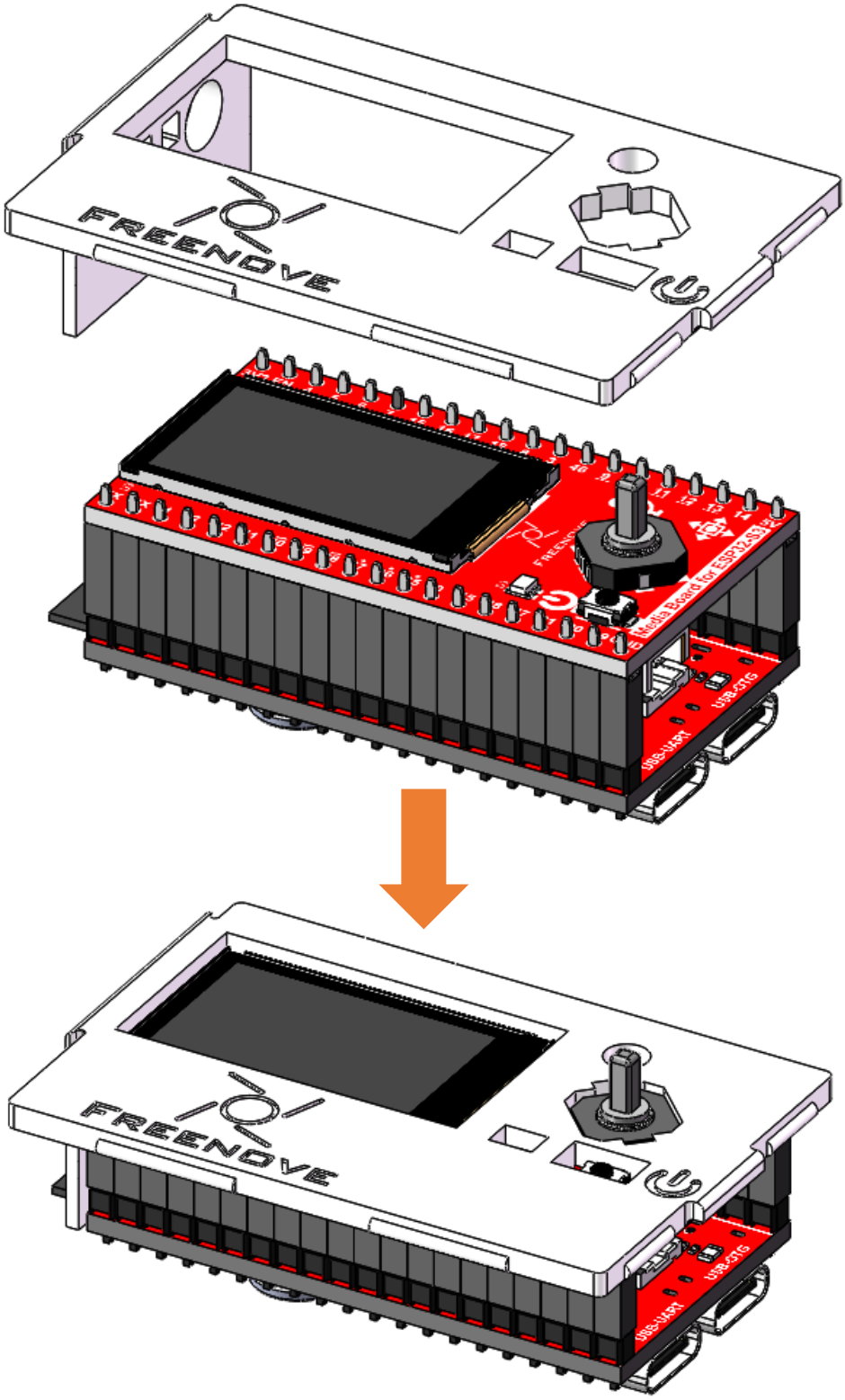
Caution: Observe correct battery polarity during connection. Reverse polarity may damage the product.
(If you do not use the battery, you may skip this step.)

Place the battery as shwon below. (Skip this if you do not have a battery.)

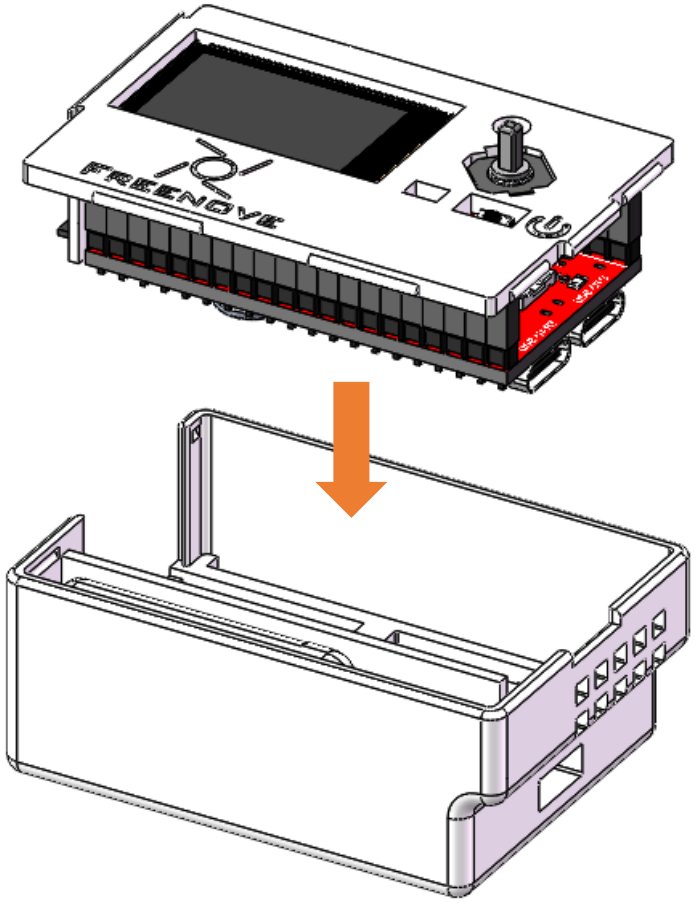




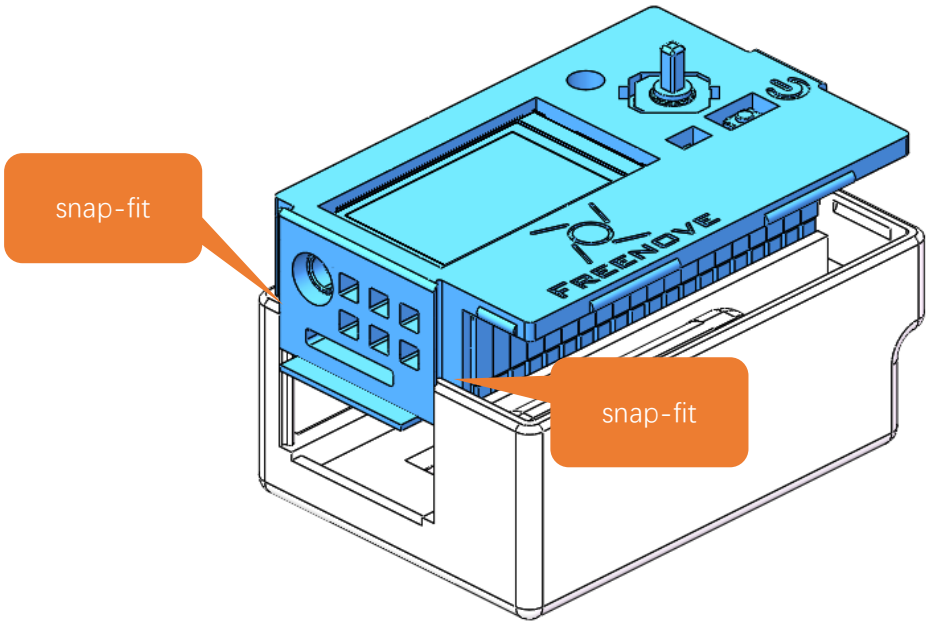
Place the board upside down with the extension board facing up, then secure the housing cover on it.



Align the housing cover with the base and secure both components together.

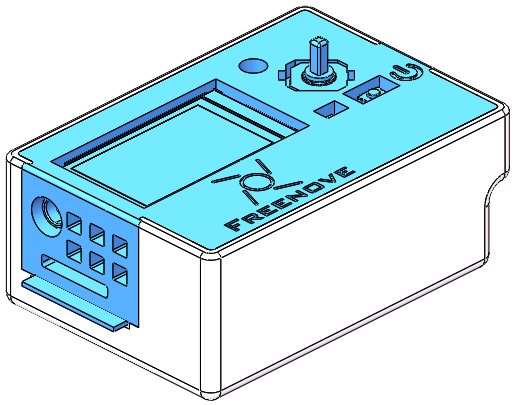


Align the housing cover with the snap-fit on the base.
Gently press down until fully seated.



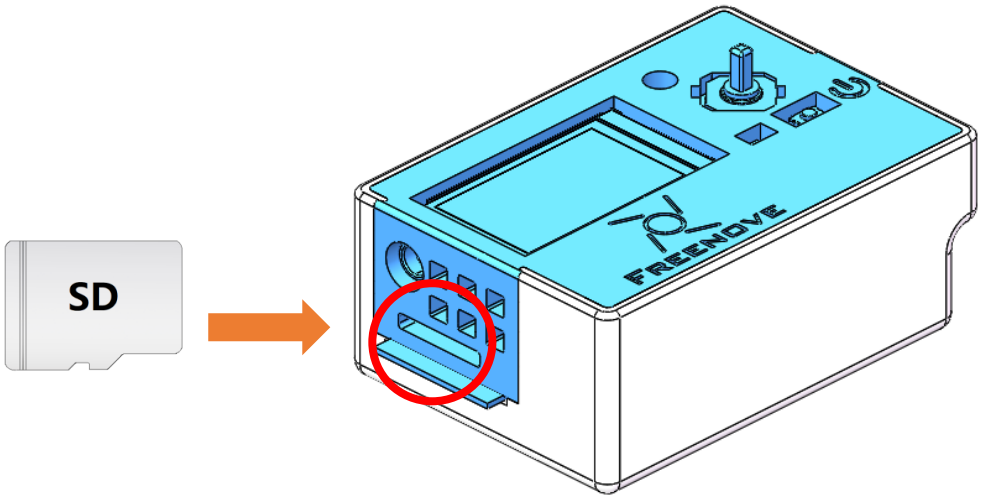
Note: The housing is fragile. Avoid forceful installation.

The installation should look like the image below once completed.



Please note that the typical accuracy of FDM-type 3D printers is 0.1–0.2 mm, so snap-fit structures may have slight looseness, which is normal.

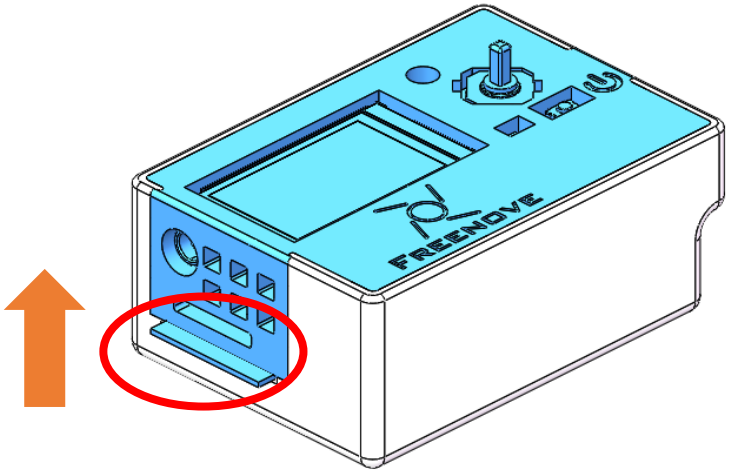
Finally, insert the SD card to the ESP32S3.



To depart the housing, please do as following:

Caution: The housing is fragile. Avoid using excessive force during disassembly

Gently push upward on the antenna area of the ESP32-S3 (highlighted in the circled section below) with your finger.



Chapter 1 LEDPixel Test

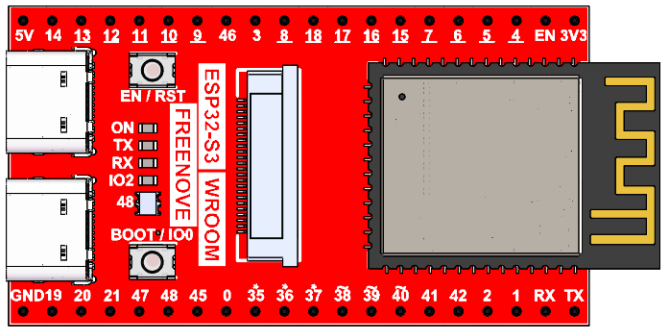
In this chapter, we will learn how to drive an RGB LED by assigning specific pins to control its color output.

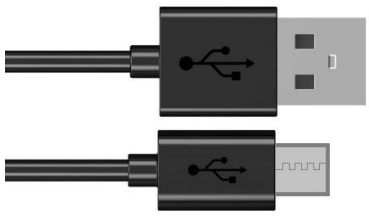
Project 1.1 LEDPixel

Learn the basic usage of WS2812 and use it to flash red, green, blue and white.
Note: Both the LED Pixels on the ESP32S3 board and on the extension board are connected to IO48 pin, so they will always maintain perfectly synchronized lighting effects.

Component List

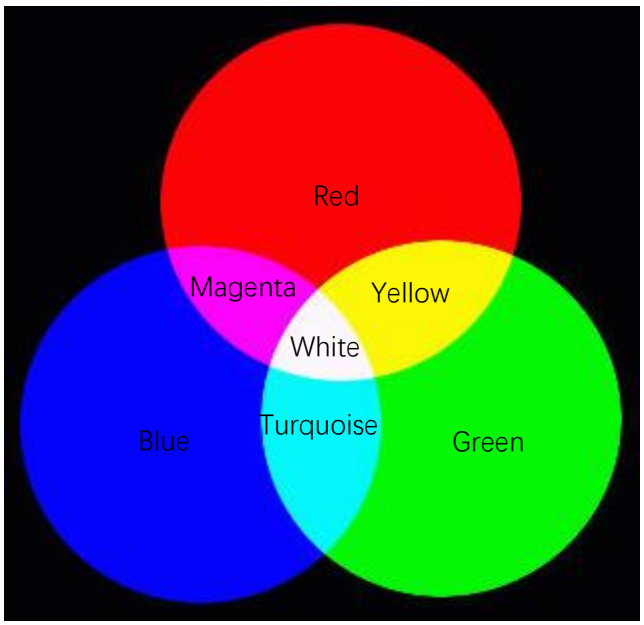
Freenove Media Kit for ESP32-S3 x1	USB cable x1

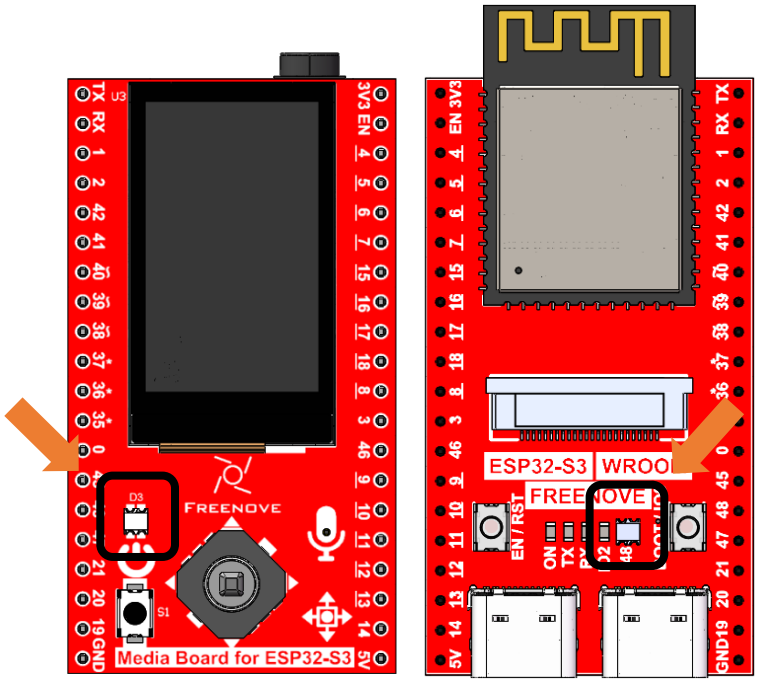




Related Knowledge

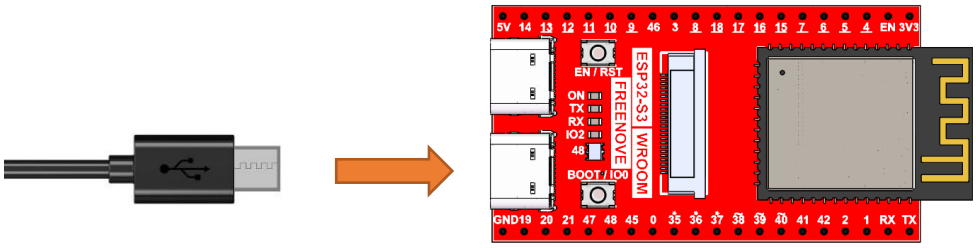
The three primary colors of light are red, green, and blue. When these colors are combined in different proportions and brightness levels, they can generate almost all colors we can see.





Circuit

Connect Freenove Media Kit for ESP32-S3 to your computer using the USB cable.

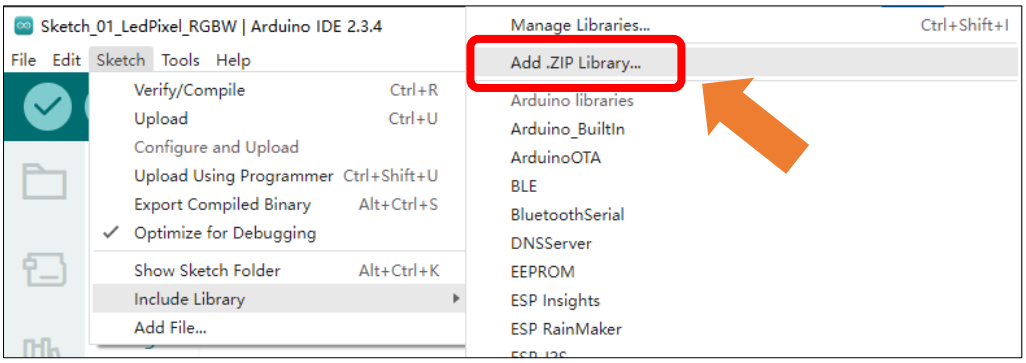


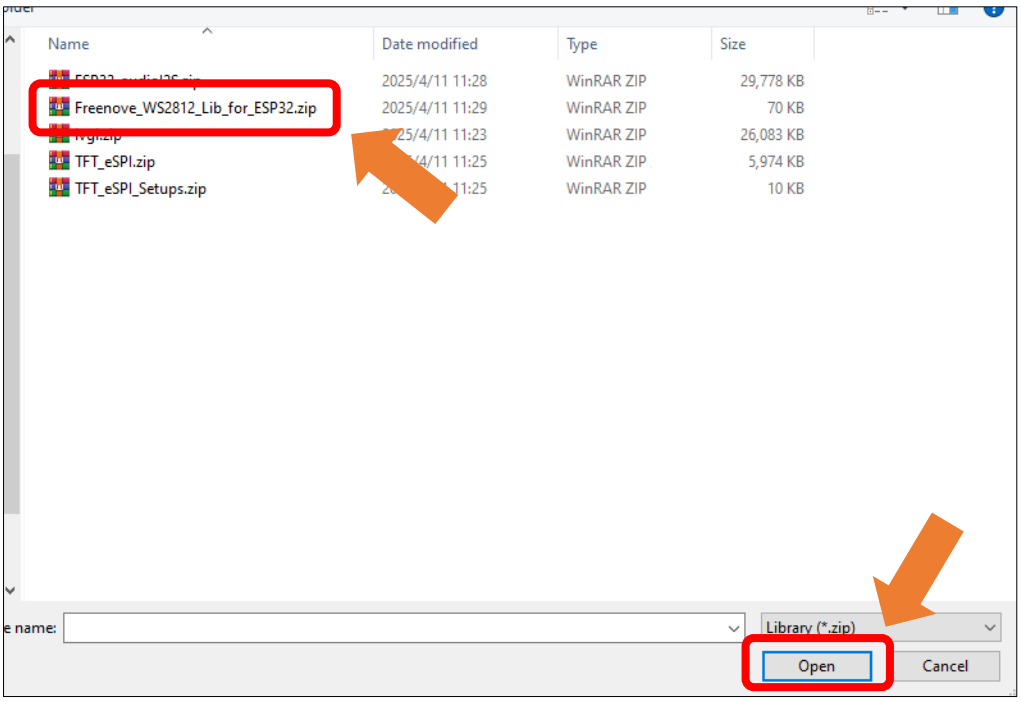
Sketch

This code uses a library named "Freenove_WS2812_Lib_for_ESP32". If you have not yet installed it, please do so first.

How to install the library

Open Arduino IDE, click **Sketch** -> **Include Library** -> **Add .ZIP Library**. In the pop-up window, find the file named "./Libraries/Freenove_WS2812_Lib_for_ESP32.Zip" which locates in this directory, and click **OPEN.**





Sketch_01_LedPixel_RGBW

Sketch_01_LedPixel_RGBW.ino

```
#include "Freenove_WS2812_Lib_for_ESP32.h"

#define LEDS_COUNT 1  // Number of LEDs in the strip
#define LEDS_PIN 48   // GPIO pin connected to the LED strip
#define CHANNEL 0     // PWM channel for LED control

// Initialize the LED strip with the specified parameters
Freenove_ESP32_WS2812 strip = Freenove_ESP32_WS2812(LEDS_COUNT, LEDS_PIN, CHANNEL, TYPE_GRB);

// Array of colors to cycle through: Red, Green, Blue, White, Off
uint8_t m_color[5][3] = { { 255, 0, 0 }, { 0, 255, 0 }, { 0, 0, 255 }, { 255, 255, 255 }, { 0, 0, 0 } };
int delayval = 100;  // Delay between color changes in milliseconds

// Setup function runs once when the program starts
void setup() {
  strip.begin();           // Initialize the LED strip
  strip.setBrightness(10);  // Set the brightness of the LED strip (0-100)
}

// Loop function runs continuously after setup
void loop() {
  for (int j = 0; j < 5; j++) {                                    // Loop through each color
    for (int i = 0; i < LEDS_COUNT; i++) {                         // Loop through each LED (only one in this case)
      strip.setLedColorData(i, m_color[j][0], m_color[j][1], m_color[j][2]);  // Set the color of the LED
      strip.show();                                                //
      Update the LED strip with the new color
      delay(delayval);                                             // Wait for a short period before changing the color again
    }
    delay(500);  // Wait for half a second before moving to the next color
  }
}
```

The following is the program code:

```
#include "Freenove_WS2812_Lib_for_ESP32.h"

#define LEDS_COUNT 1  // Number of LEDs in the strip
#define LEDS_PIN 48   // GPIO pin connected to the LED strip
#define CHANNEL 0     // PWM channel for LED control

// Initialize the LED strip with the specified parameters
Freenove_ESP32_WS2812 strip = Freenove_ESP32_WS2812(LEDS_COUNT, LEDS_PIN, CHANNEL, TYPE_GRB);

// Array of colors to cycle through: Red, Green, Blue, White, Off
uint8_t m_color[5][3] = { {255, 0, 0}, {0, 255, 0}, {0, 0, 255}, {255, 255, 255}, {0, 0, 0} };
int delayval = 100;  // Delay between color changes in milliseconds

// Setup function runs once when the program starts
void setup() {
  strip.begin();           // Initialize the LED strip
  strip.setBrightness(10);  // Set the brightness of the LED strip (0-100)
}

// Loop function runs continuously after setup
void loop() {
  for (int j = 0; j < 5; j++) {
    for (int i = 0; i < LEDS_COUNT; i++)
      strip.setLedColorData(i, m_color[j][0], m_color[j][1], m_color[j][2]);
      strip.show();
      delay(delayval);
    }
    delay(500);  // Wait for half a second before moving to the next color
  }
}
```

To use some libraries, first you need to include their header file.

```
#include "Freenove_WS2812_Lib_for_ESP32.h"
```

Define the pin connected to the WS2812, the number of WS2812, and RMT channel values.

```
#define LEDS_COUNT 1  // Number of LEDs in the strip
#define LEDS_PIN 48   // GPIO pin connected to the LED strip
#define CHANNEL 0     // PWM channel for LED control
```

Use the above parameters to create a WS2812 object strip.

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

Define the color values to be used, such as red, green, blue, white, and black.

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

Define a variable to set the time interval for each led to light up. The smaller the value is, the faster it will light up.

```
int delayval = 100;  // Delay between color changes in milliseconds
```

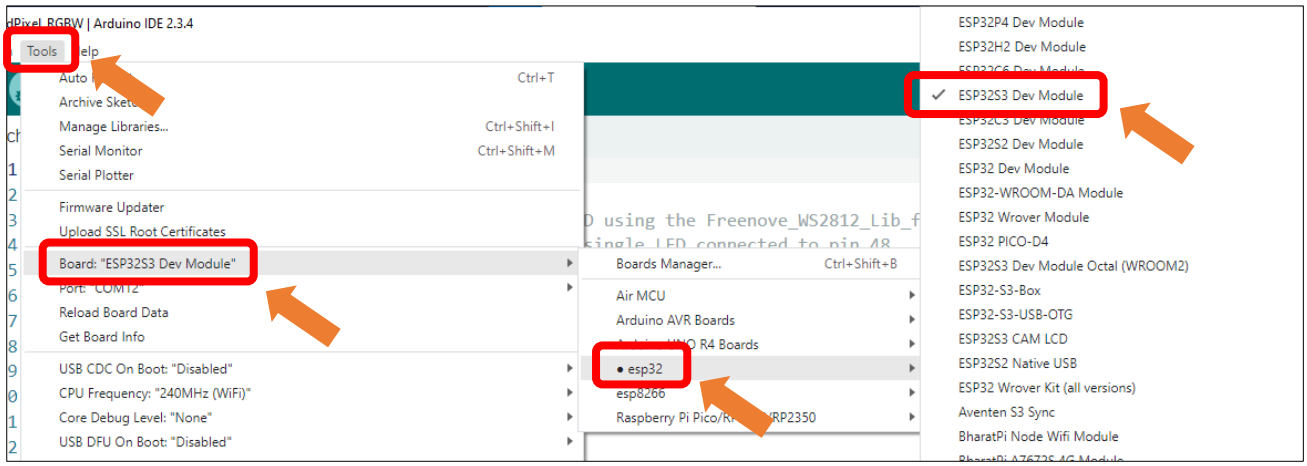
Initialize strip() in setup() and set the brightness.

```
13  strip.begin();           // Initialize the LED strip
14  strip.setBrightness(10); // Set the brightness of the LED strip (0-100)
```

In the loop(), there are two "for" loops, the internal for loop is to light the LED one by one, and the external one to switch colors. strip.setLedColorData() is used to set the color, but it does not change immediately. Only when strip.show() is called will the color data be sent to the LED to change the color.

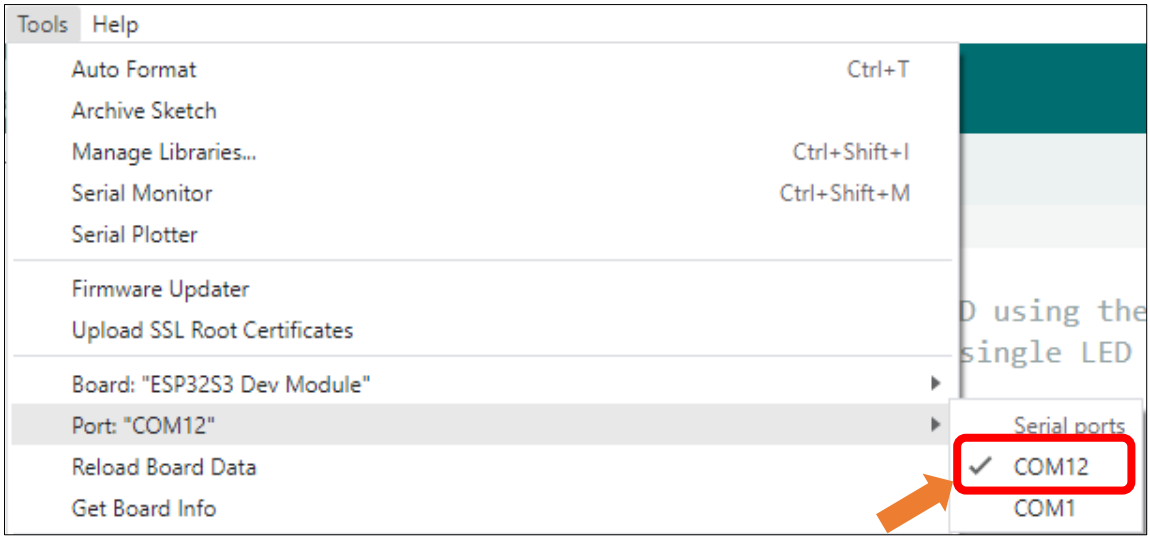
```
17  for (int j = 0; j < 5; j++) {
18    for (int i = 0; i < LEDS_COUNT; i++)
19      strip.setLedColorData(i, m_color[j][0], m_color[j][1], m_color[j][2]);
20      strip.show();
21      delay(delayval);
22    }
23    delay(500);  // Wait for half a second before moving to the next color
24  }
```

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

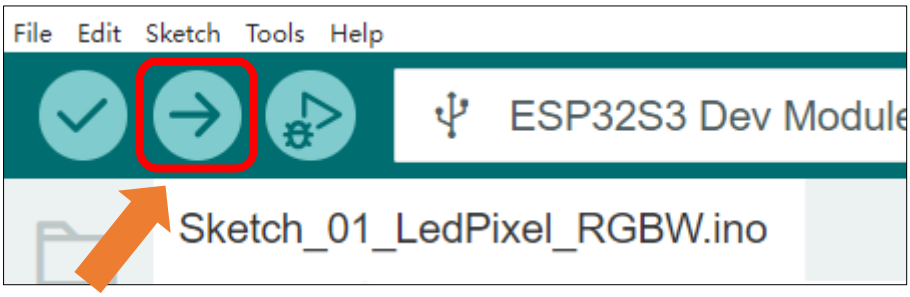


After connecting the Freenove Media Kit for ESP32-S3, the system will assign a serial communication port named in the format 'COMx' (**where 'x' is a numeric ID that may vary across computers**). You must select the correct port under Tools → Port.

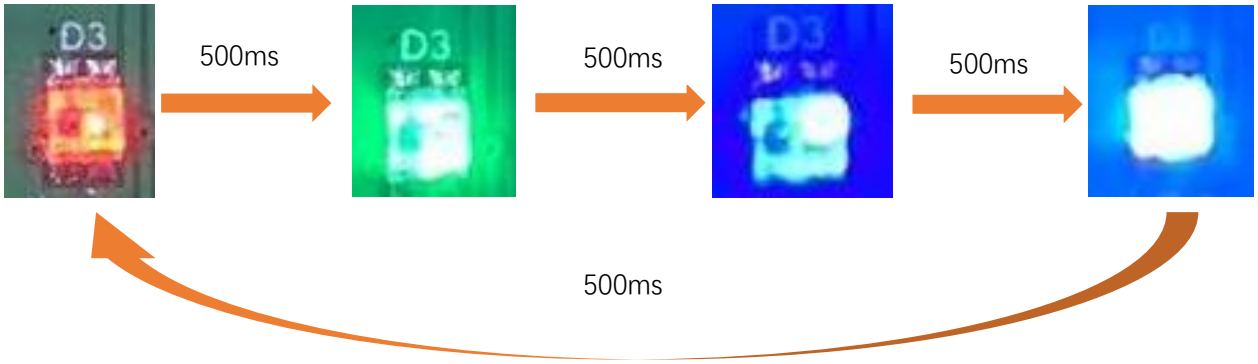
Note: COM1 is typically NOT the port used by the Freenove Media Kit for ESP32-S3.



Click **upload** to upload the sketch to ESP32S3 board.



After successfully uploading the code, you will observe the WS2812 LEDs cycling through predefined colors (red → green → blue → white → red) in sequence. The color transitions occur at 500ms intervals, creating a smooth color-changing loop effect.



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

Reference

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

Constructor to create a ws2812 object.

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

Parameters

n: The number of led.

pin_gpio: The pin connected to the LED.

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

t: Types of LED.

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

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

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

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

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

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

`void begin(void);`
Initialize the LEDPixel object

`void setLedColorData (u8 index, u8 r, u8 g, u8 b);`
`void setLedColorData (u8 index, u32 rgb);`
`void setLedColor (u8 index, u8 r, u8 g, u8 b);`
`void setLedColor (u8 index, u32 rgb);`

Set the color of led with order number n.
`void show(void);`
Send the color data to the led and display the set color immediately.
`void setBrightness(uint8_t);`
Set the brightness of the LED.

To learn more about the library, you may visit the following website:
https://github.com/Freenove/Freenove_WS2812_Lib_for_ESP32

Chapter 2 Battery Voltage Detection

ADC is used to convert analog signals into digital signals. Control chip on the control board has integrated this function. Now let us try to use this function to convert analog signals into digital signals

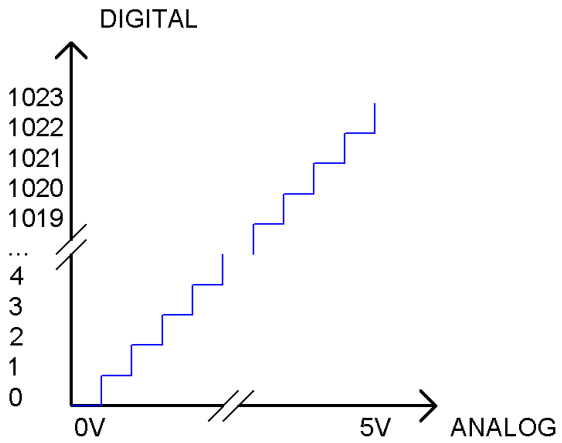
For battery precautions and selection guidelines, please refer to Battery.

Project 2.1 Battery Voltage Value

Related Knowledge

ADC

An 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 onboard ADC module is set to 10 bits by default, which means the resolution is 2^10=1024, so its range (at 5V) will be evenly divided into 1024 parts. By default, the resolution of the onboard ADC is set to 10 bits, which can be updated to 12 bits (0-4096) and 14 bits (0-16383) resolution to improve the accuracy of the analog readings. Any analog value can be mapped to one digital value using the resolution of the converter. 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-5/1024V corresponds to digital 0;
Subsection 2: the analog in rang of 5 /1024V-2*5/1024V corresponds to digital 1;

The following analog signal will be divided accordingly.
The conversion formula is as follows:

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

Kirchhoff's Laws

Kirchhoff's Laws are the two fundamental principles of circuit analysis, formulated by German physicist Gustav Kirchhoff in 1845. Consisting of Kirchhoff's **Current Law (KCL) and Voltage Law (KVL)**, they are applicable to all lumped-parameter circuits — regardless of whether they are linear or nonlinear, time-varying or time-invariant. These laws represent one of the key manifestations of the conservation of energy in electrical circuits.

Kirchhoff's Current Law (KCL) states:

At any given node in a lumped-parameter circuit, the algebraic sum of all currents entering (or leaving) the node at any instant is always zero.

Mathematically, this is expressed as:

$$\sum_{k=1}^{n} I_k = 0$$

Kirchhoff's Voltage Law (KVL) states:

In any closed loop of a circuit, the algebraic sum of all voltage drops is always equal to zero.

Mathematically, this is expressed as:

$$\sum_{k=1}^{n} U_k = 0$$

Ohm's Law

Ohm's Law, formulated by the German physicist Georg Simon Ohm in 1827, is one of the most fundamental laws in circuit theory. It defines the basic relationship between voltage and current in a linear resistive element:

Under constant temperature conditions, the current (I) flowing through a conductor is directly proportional to the voltage (U) across it and inversely proportional to the conductor's resistance (R).

Mathematically, it is expressed as:

$$U = I \times R$$

Where:

- U represents Voltage （Unit: Volt,V）
- I refer to current （Unit: Ampere, A）
- R denotes resistance （Unit:: Ohm, Ω）

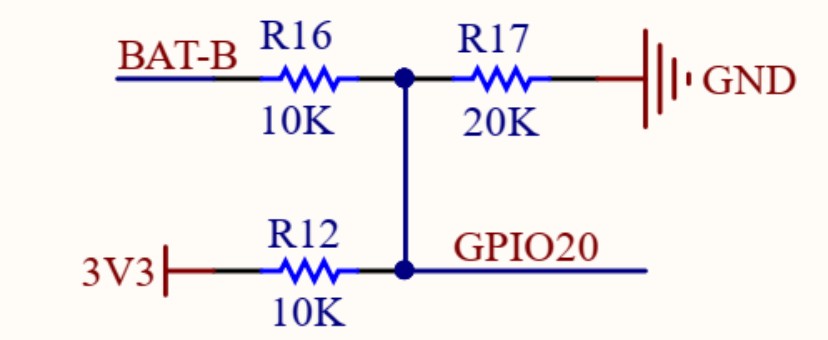
Based on the above, the following relationships can be derived from the formula:

$$I = \frac{U}{R} \qquad\qquad R = \frac{U}{I}$$

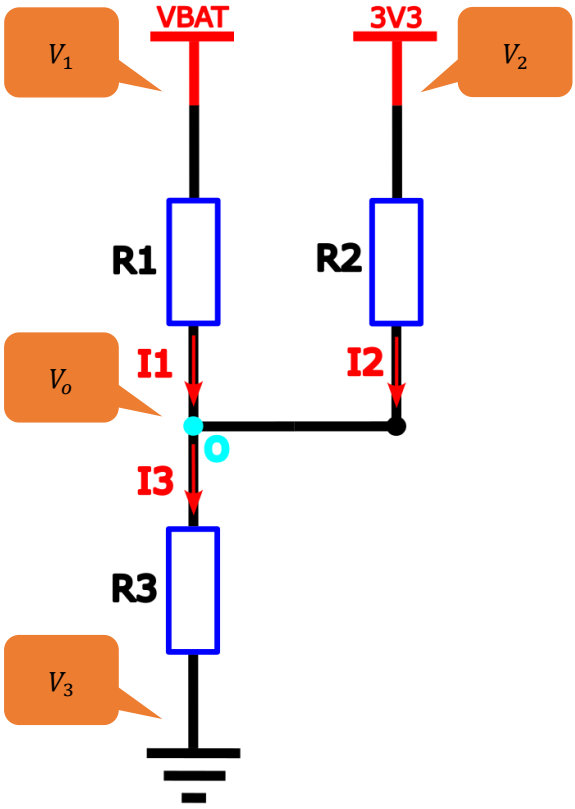
Battery Voltage Detection

The maximum **input voltage** for the GPIO pins of the Freenove Media Kit for ESP32-S3 is 3.3V. Exceeding this limit may cause permanent damage to the device. Since the full charge voltage of a 3.7V lithium battery can theoretically reach **4.2V**, it is strictly prohibited to directly connect the lithium battery to the GPIO pins of the Freenove Media Kit for ESP32-S3. The input voltage must be regulated within the safe range through circuit

design (as shown in the figure below).



The schematic above can be simplified as the following equivalent circuit for easier understanding.



Based on Kirchhoff's Current Law (KCL), we can derive that:

$$\boldsymbol{I_1 + I_2 + I_3 = 0}$$

Accoding to the following formula of Ohm's Law

$$\boldsymbol{I = \frac{U}{R}}$$

It can be further deduced that:

$$\boldsymbol{\frac{U_1}{R_1} + \frac{U_2}{R_2} + \frac{U_3}{R_3} = 0}$$

Since voltage is the difference in potential, it can be converted into:

$$\frac{V_1 - V_o}{R_1} + \frac{V_2 - V_o}{R_2} + \frac{V_3 - V_o}{R_3} = 0$$

By simplifying it, we can get:

$$V_1 = V_o\left(1 + \frac{R_1}{R_2} + \frac{R_1}{R_3}\right) - V_2 \times \frac{R_1}{R_2} - V_3 \times \frac{R_1}{R_3}$$

Substituting the actual resistance value, we can finally get

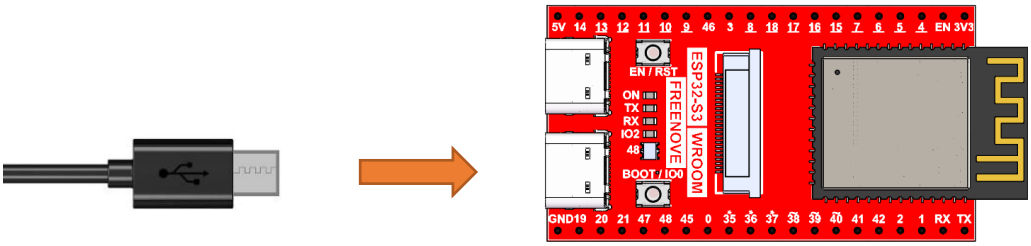
$$V_1 = V_o \times 2.5 - 3300$$

Where:

R_1 = 10KΩ, R_2 = 10KΩ, R_3 = 20KΩ, The unit of V_1 is millivolt(mV).

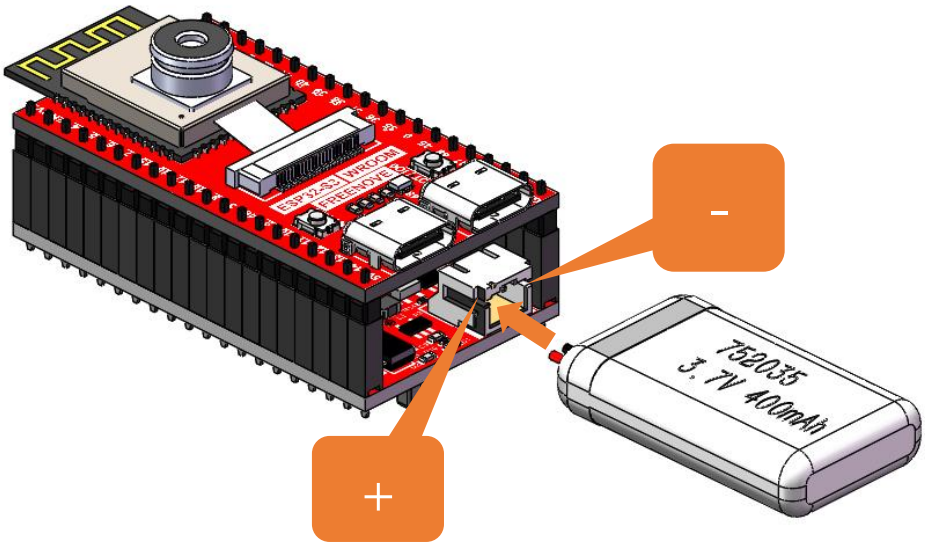
Circuit

Connect Freenove Media Kit for ESP32-S3 to your computer using the USB cable.



Connect the battery to Freenove Media Kit for ESP32-S3.

For battery precautions and selection guidelines, please refer to Battery.



Sketch

```
#define BATTERY_MIN 3200
#define BATTERY_MAX 4200

const uint8_t BATTERY_PIN = 20;

void setup() {
  // Setup code to run once:
  Serial.begin(115200);          // Set baud rate to 115200
  pinMode(BATTERY_PIN, INPUT);   // Set battery voltage detection pin to input mode
  analogReadResolution(12);      // Set ADC resolution to 12 bits
  analogSetAttenuation(ADC_11db);// Set attenuation to 11dB
}

void loop() {
  // Main code to run repeatedly:
  battery_read();
}

void battery_read() {
  int total_value = 0, valid_time = 0, adc_value = 0, battery_volts = 0;
  // Sample 20 times for filtering
  for(int i = 0; i < 20; i++) {
    // Read ADC value
    adc_value = analogReadMilliVolts(BATTERY_PIN);
    // Calculate battery voltage using voltage divider (unit: mV)
    battery_volts = (adc_value * 2.5) - 3300;
    // Filter out abnormal values
    if(battery_volts > BATTERY_MIN && battery_volts < BATTERY_MAX) {
      total_value += battery_volts;
      valid_time++;
    }
    delay(50);
  }
  // Calculate average of valid measurements
  if(valid_time != 0)
    battery_volts = total_value / valid_time;

  // Output results
  Serial.printf("valid time: %d, battery volts: %.2fV\n", valid_time, battery_volts/1000.0);
}
```

Define the maximum and minimum voltage range

```
#define BATTERY_MIN 3200
#define BATTERY_MAX 4200
```

Define battery voltage detection pin.

```
const uint8_t BATTERY_PIN = 20;
```

Set the serial baud rate to 115200.

```
Serial.begin(115200); // Set baud rate to 115200
```

Set the ADC resolution to 12-bit, providing a measurement range of 0 to 4095 (2^{12} - 1).

```
analogReadResolution(12);
```

Set the ADC attenuation factor to 11db

```
analogSetAttenuation(ADC_11db);// Set attenuation to 11dB
```

Perform cyclic sampling of battery voltage 20 times and apply filtering processing.

```
for(int i = 0; i < 20; i++) {
  // Read ADC value
  adc_value = analogReadMilliVolts(BATTERY_PIN);
  // Calculate battery voltage using voltage divider (unit: mV)
  battery_volts = (adc_value * 2.5) - 3300;
  // Filter out abnormal values
  if(battery_volts > BATTERY_MIN && battery_volts < BATTERY_MAX) {
    total_value += battery_volts;
    valid_time++;
  }
```

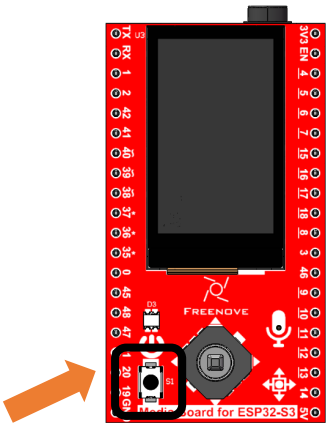
Calculate the average effective battery voltage

```
// Calculate average of valid measurements
if(valid_time != 0)
   battery_volts = total_value / valid_time;
```

Output battery voltage value and sampling times.

```
// Output results
Serial.printf("valid time: %d, battery volts: %.2fV\n", valid_time, battery_volts / 1000.0);
```

Upload the code to the board and press ON the power button.



The current battery voltage value will be printed on the serial monitor.

```
11:34:48.273 -> valid time: 20, battery volts: 3.76V
11:34:49.282 -> valid time: 20, battery volts: 3.76V
11:34:50.249 -> valid time: 20, battery volts: 3.76V
11:34:51.267 -> valid time: 20, battery volts: 3.76V
11:34:52.276 -> valid time: 20, battery volts: 3.76V
11:34:53.249 -> valid time: 20, battery volts: 3.76V
11:34:54.249 -> valid time: 20, battery volts: 3.76V
11:34:55.280 -> valid time: 20, battery volts: 3.76V
```

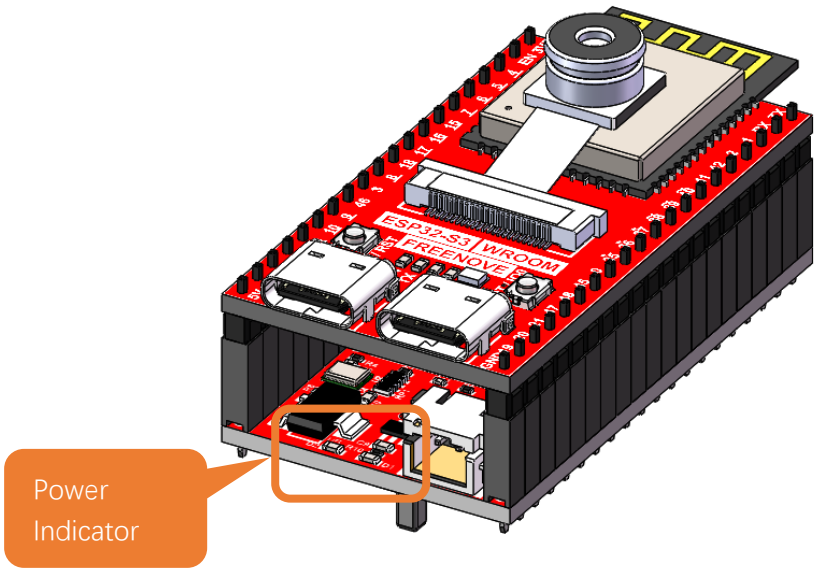
When both the battery and USB are connected, the Freenove Media Kit for ESP32-S3 will be powered by USB and the battery will charge slowly

```
11:35:08.249 -> valid time: 20, battery volts: 3.76V
11:35:09.286 -> valid time: 20, battery volts: 3.76V
11:35:10.295 -> valid time: 20, battery volts: 3.77V
11:35:11.271 -> valid time: 20, battery volts: 3.77V
```

If you have any concerns, please feel free to contact us via support@freenove.com
Charging & Power Indicators:

When using the USB port on the board to charge the battery:
While charging, the blue LED will blink.
When charging is complete, the blue LED will stay lit.
If no battery is connected, the blue LED will keep blinking.

When the device is not connected to USB, it runs on battery power, and the green LED remains steadily lit.



Reference

analogReadResolution(uint8_t bits)
This function is used to configure the ADC's conversion resolution and measurement range. **Parameters** **bits:** The resolution in bits. Higher bit values result in greater measurement precision. ADC sampling range: 0 to (2^bits − 1).
analogSetAttenuation(adc_attenuation_t attenuation)
This function configures the input signal attenuation factor for the ADC. **Parameters** **attenuation:** Attenuation multiple ADC_0dB: Voltage measurement range: 0mV ~ 950mV. ADC_2_5dB: Voltage measurement range: 0mV ~ 1250mV. ADC_6dB: Voltage measurement range: 0mV ~ 1750mV. ADC_11dB: Voltage measurement range: 0mV ~ 3100mV.
analogReadMilliVolts(uint8_t pin)
This function reads the voltage value (in millivolts, mV) from an analog pin. **Parameters** **pin:** Pin number.

Chapter 3 5-Way Navigation Switch Test

In the previous section, we covered ADC concepts. This section will discuss the working principles and operation of a 5-way navigation switch, as there is some relevance between the two topics.

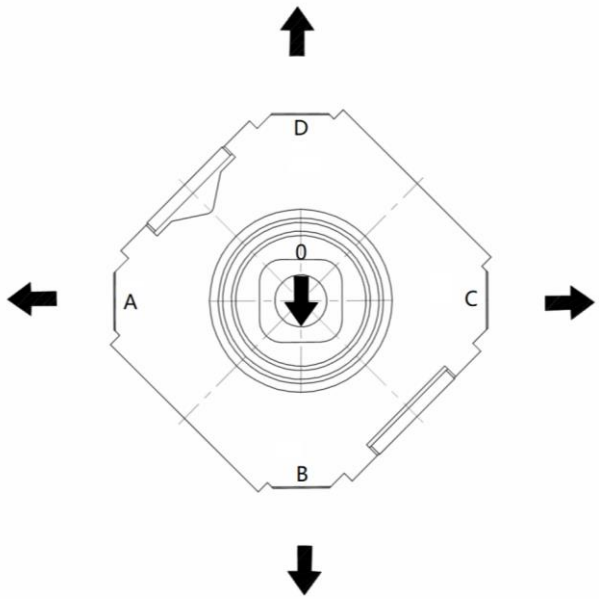
Project 3.1 Button Value

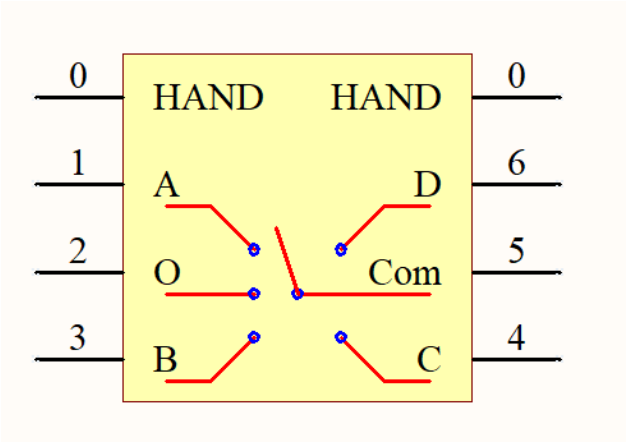
Component Knowledge

5-way Navigation Switch

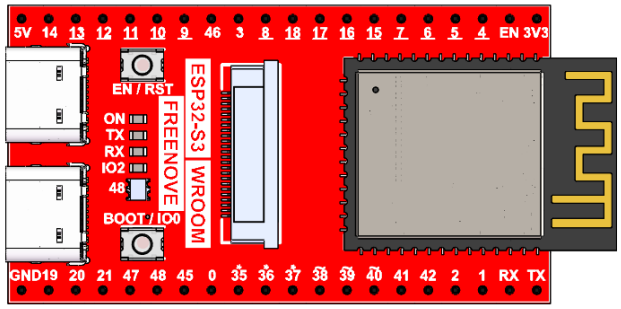
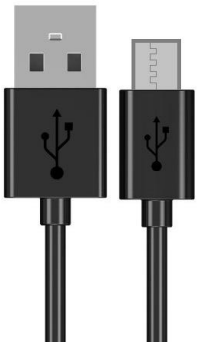
The 5-way navigation switch is a digital switch that supports actuation in five directions: up, down, left, right, and center (press).

Its operation is as follows:
No press: All terminals (A, B, C, D, O, Com) remain open (non-conducting).
Center press: O and Com make contact (conduct).
Upward push: D and Com make contact.
Downward push: B and Com make contact.
Leftward push: A and Com make contact.
Rightward push: C and Com make contact.
This switch is widely used in applications requiring directional input, such as game controllers, remote controls, and navigation interfaces.



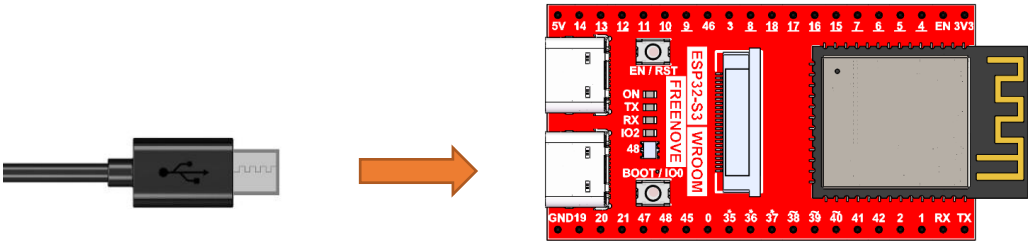


Component List

Freenove Media Kit for ESP32-S3 x1	USB cable x1
	

Circuit

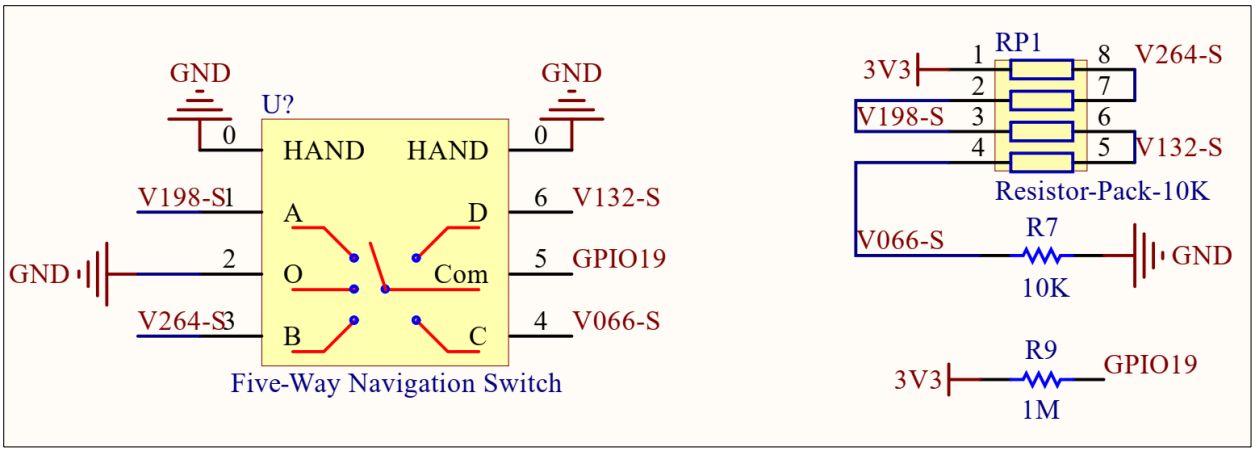
Connect Freenove Media Kit for ESP32-S3 to your computer using the USB cable.



Schematic

We use the ADC function of GPIO19 to read the voltage value of Com and determine which direction of the 5-way button is pressed based on different voltage values.

The circuit schematic is as follows.



Typically, we read the ADC value from GPIO19 and treat any value within the range of [(ADC - 50), (ADC + 50)] as a valid button press. Please refer to the code in Sketch_02_1_Button_Value and Sketch_02_2_Button to better understand this logic.

Sketch

Sketch_03_1_Button_Value

The following is the program code:

```
#define BUTTON_PIN 19  // GPIO pin connected to the button

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

  // Set the ADC resolution to 12 bits (0-4095)
  analogReadResolution(12);
  // Set the ADC attenuation to 11 dB
  analogSetAttenuation(ADC_11db);
}

void loop() {
  // Read the analog/millivolts value for pin 20
  int analogValue = analogReadMilliVolts(BUTTON_PIN);
  // Print the ADC value to the serial monitor
  Serial.printf("ADC value = %d\n", analogValue);

  // Delay in between reads for clear read from serial
  delay(100);
}
```

Define the pin connected to the 5-way navigation button.

```
#define BUTTON_PIN 19  // GPIO pin connected to the button
```

Set the serial baud rate to 115200.

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

Set the ADC resolution to 12-bit, providing a measurement range of 0 to 4095 (2^{12} - 1).

```
analogReadResolution(12);
```

Set the ADC input attenuation level to define the measurable voltage range

```
analogSetAttenuation(ADC_11db);
```

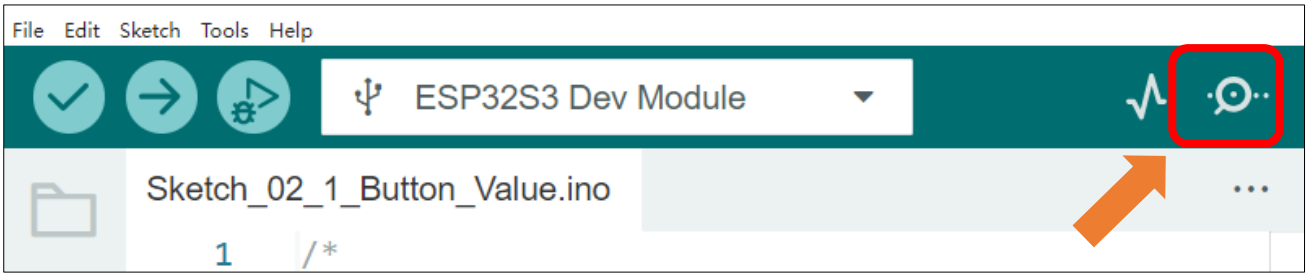
Read the voltage value from GPIO19 in millivolts (mV).

```
int analogValue = analogReadMilliVolts(BUTTON_PIN);
```

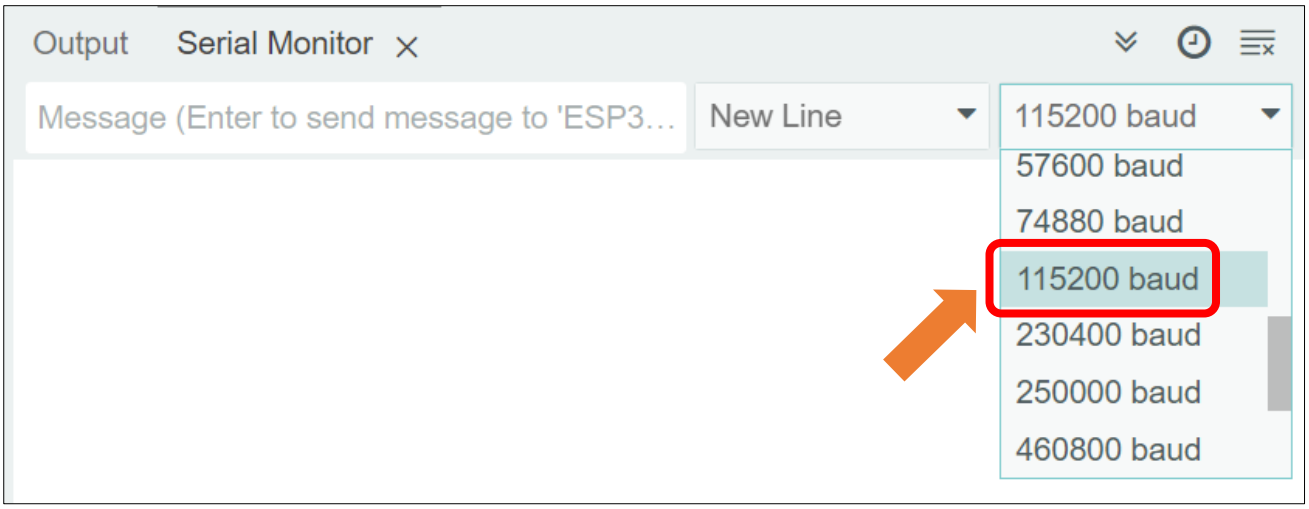
Print the GPIO19 voltage reading to the serial monitor.

```
Serial.printf("ADC value = %d\n", analogValue);
```

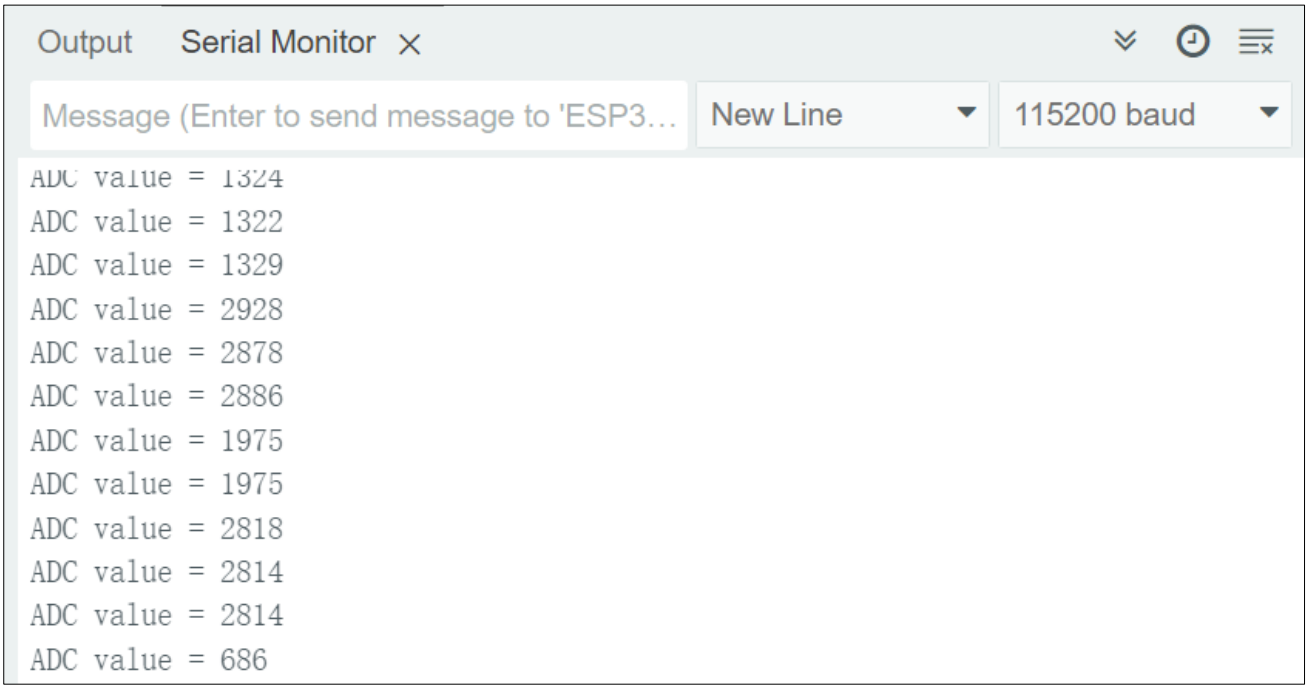
Open the serial monitor upon the code upload completes.



Select the correct baud rate (115200).



The serial monitor will display the ADC sampling values of the GPIO19 pin in real time. When operating the 5-way switch, the corresponding ADC values will dynamically update according to the switch state.



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

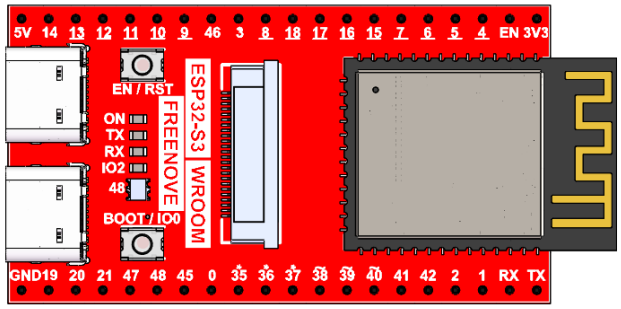
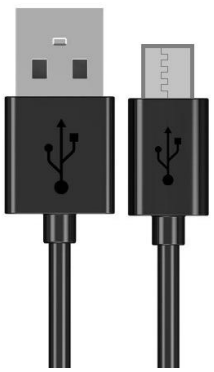
Reference

analogReadResolution(uint8_t bits)
This function is used to configure the ADC's conversion resolution and measurement range. **Parameters** **bits:** The resolution in bits. Higher bit values result in greater measurement precision. ADC sampling range: 0 to (2^bits – 1).
analogSetAttenuation(adc_attenuation_t attenuation)
This function configures the input signal attenuation factor for the ADC. **Parameters** **attenuation:** Attenuation multiple ADC_0dB: Voltage measurement range: 0mV ~ 950mV. ADC_2_5dB: Voltage measurement range: 0mV ~ 1250mV. ADC_6dB: Voltage measurement range: 0mV ~ 1750mV. ADC_11dB: Voltage measurement range: 0mV ~ 3100mV.
analogReadMilliVolts(uint8_t pin)
This function reads the voltage value (in millivolts, mV) from an analog pin. **Parameters** **pin:** Pin number.

Project 3.2 Button

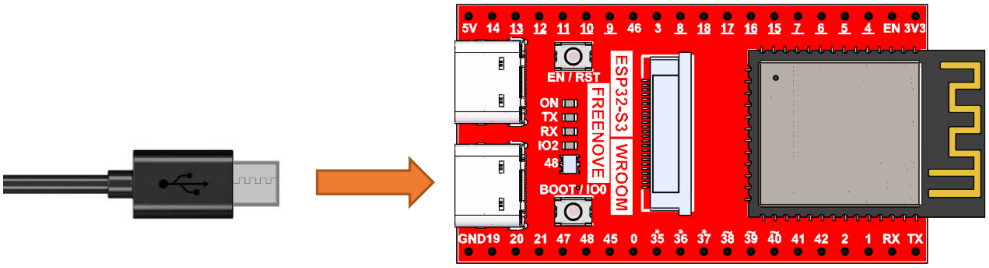
In the previous section, we learned how to read the ADC values of the 5-way navigation button. In this section, we will learn how to accurately determine the specific direction that the button moves based on these ADC values.

Component List

Freenove Media Kit for ESP32-S3 x1	USB cable x1
	

Circuit

Connect Freenove Media Kit for ESP32-S3 to your computer using the USB cable.

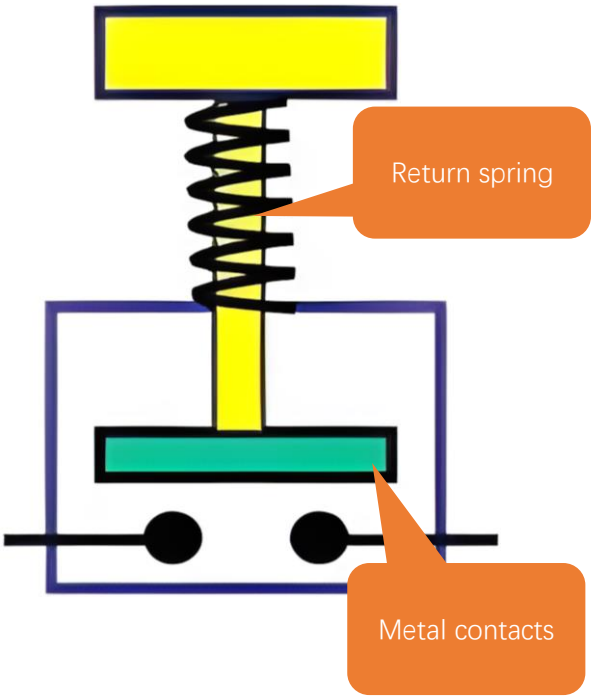


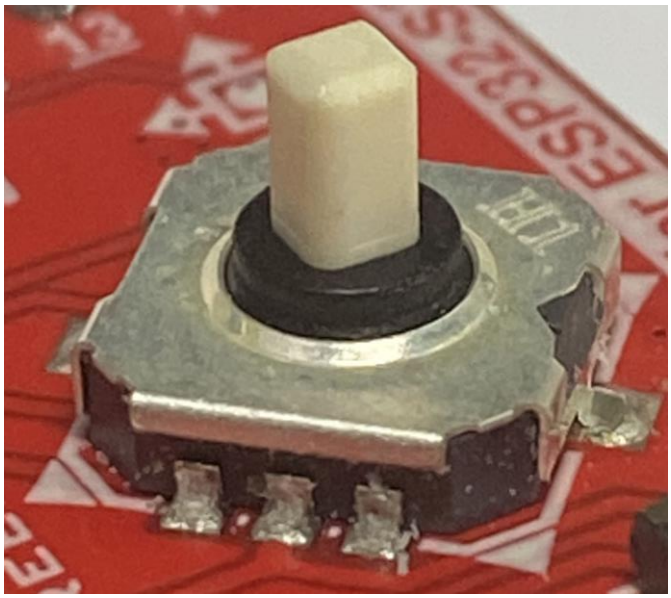
Component Knowledge

Button Bouncing

Push buttons, as common mechanical input devices, utilize metal contacts and a return spring in their internal structure. However, due to the inherent physical properties of mechanical contacts, instantaneous stable conduction or disconnection cannot be achieved upon pressing or releasing. Instead, rapid mechanical bouncing occurs, generating a series of unstable on-off signals during the transition. This phenomenon, known as contact bounce, is also observed in 5-way navigation switches.

To ensure signal accuracy and reliability, effective debounce strategies must be incorporated into the system design. These measures mitigate the effects of mechanical bounce, providing clean and consistent input signals for downstream processing.





Sketch

Sketch_03_2_Button

The following is the program code:

```
#include "driver_button.h"

#define BUTTON_PIN 19  // GPIO pin connected to the button
Button button(BUTTON_PIN);

void setup() {
    // Initialize serial communication at 115200 bits per second
    Serial.begin(115200);
    // Initialize the button
    button.init();
}

void loop() {
    // Scan for button events
    button.key_scan();
    // Handle button events
    handleButtonEvents();
    // Delay to prevent excessive CPU usage
    delay(50);
}

void handleButtonEvents() {
    // Get the current state of the button
    int buttonState = button.get_button_state();
    // Get the key value of the button
    int buttonKeyValue = button.get_button_key_value();
    // Handle different button states
    switch (buttonState) {
        case Button::KEY_STATE_PRESSED:
            Serial.println(String(buttonKeyValue) + " Pressed.");
            break;
        case Button::KEY_STATE_RELEASED:
            Serial.println(String(buttonKeyValue) + " Released.");
            break;
        default:
            break;
    }
}
```

Define the pin connecting to the switch.

```
1  #define BUTTON_PIN 19  // GPIO pin connected to the button
```

Define the object of the switch.

```
8  Button button(BUTTON_PIN);
```

Set the baud rate to 115200.

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

Initialize the switch.

```
10  button.init();
```

Scan the state of the button periodically.

```
15  button.key_scan();
```

Handle button events.

```
17  handleButtonEvents();
```

Reference

Button:init();

This function is used to initialize the button.

Button:key_scan();

The function is used to scan the state of the switch and needs to be placed in the loop() function for periodic scanning.

Button:get_button_state();

This function is used to read the state of the 5-way navigation switch.

Switch states:

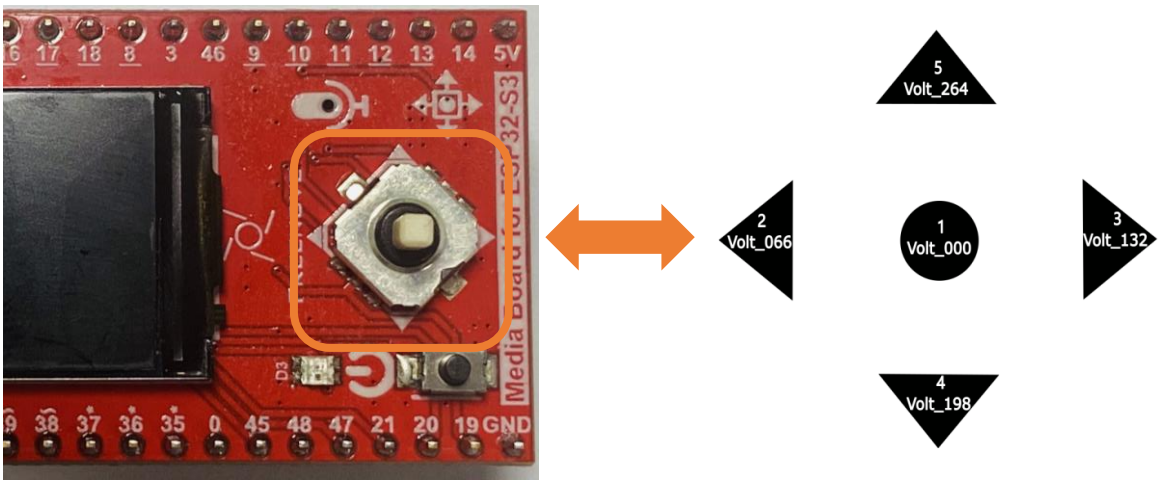
KEY_STATE_IDLE: The switch is inactive (no trigger detected).
KEY_STATE_PRESSED_BOUNCE_TIME: Button press detected, debouncing in progress (filtering contact bounce).
KEY_STATE_PRESSED: Button press confirmed (debounce complete, valid press event).
KEY_STATE_RELEASE_BOUNCE_TIME: Button release detected, debouncing in progress (filtering release bounce).
KEY_STATE_RELEASED: Button y release confirmed (debounce complete, valid release event).

Button:get_button_key_value();

The function is used to obtain the key values of a 5-way button.

The key values are as follows:

Volt_330: Not pressed (0)
Volt_000: Center (1)
Volt_264: Forward (5)
Volt_198: Backward (4)
Volt_066: Left (2)
Volt_132: Right (3)



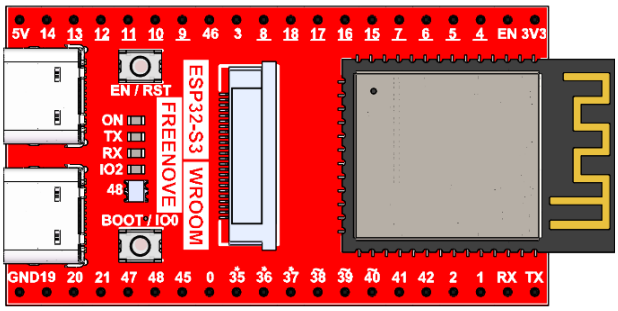
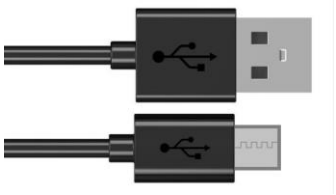
If you have any concerns, please feel free to contact us via support@freenove.com

Chapter 4 SD Card Read & Write Test

An SD card slot is integrated on the back of the ESP32-S3 WROOM. In this chapter, we learn how to use ESP32-S3 to read and write SDcard.

Project 4.1 SDMMC Test

Component List

Freenove Media Kit for ESP32-S3 x1 	USB cable x1 
SD card x1 	Card reader x1 (random color) 

Component knowledge

SD card read and write method

The ESP32-S3 provides two primary interfaces for SD card communication: SPI and SDMMC. Each interface offers distinct advantages in terms of pin usage and performance:

SPI Interface:

Requires 4 I/O pins

Delivers the slowest performance (approximately 50% of SDMMC 4-bit mode speed)

SDMMC Interface:

Offers two operational modes:

1-bit bus mode: Uses only 3 I/O pins while maintaining about 80% of the 4-bit mode's performance

4-bit bus mode: Requires 6 I/O pins but delivers maximum throughput

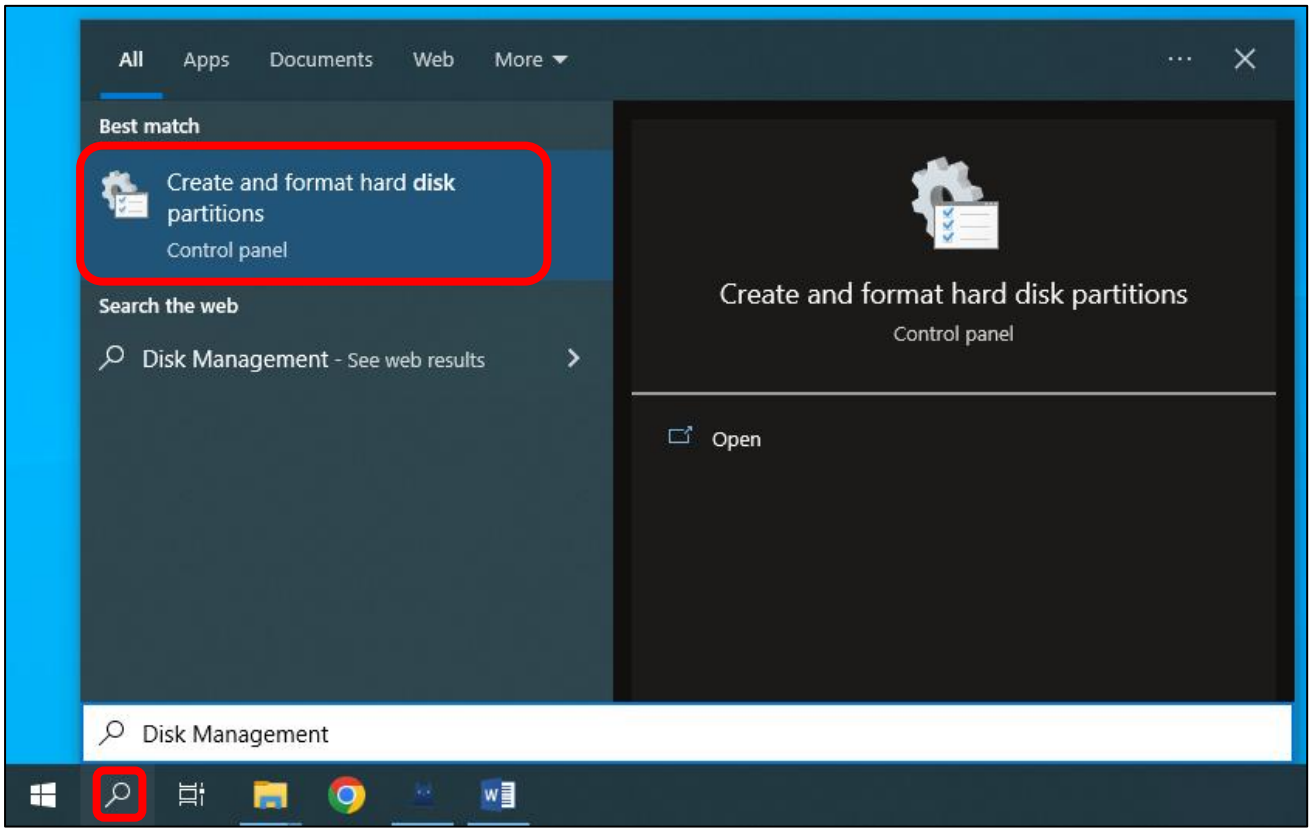
For most applications, we recommend the SDMMC 1-bit bus mode as it provides an optimal balance between pin efficiency and performance. This configuration minimizes I/O usage while still delivering satisfactory speed for typical use cases.

Format SD card

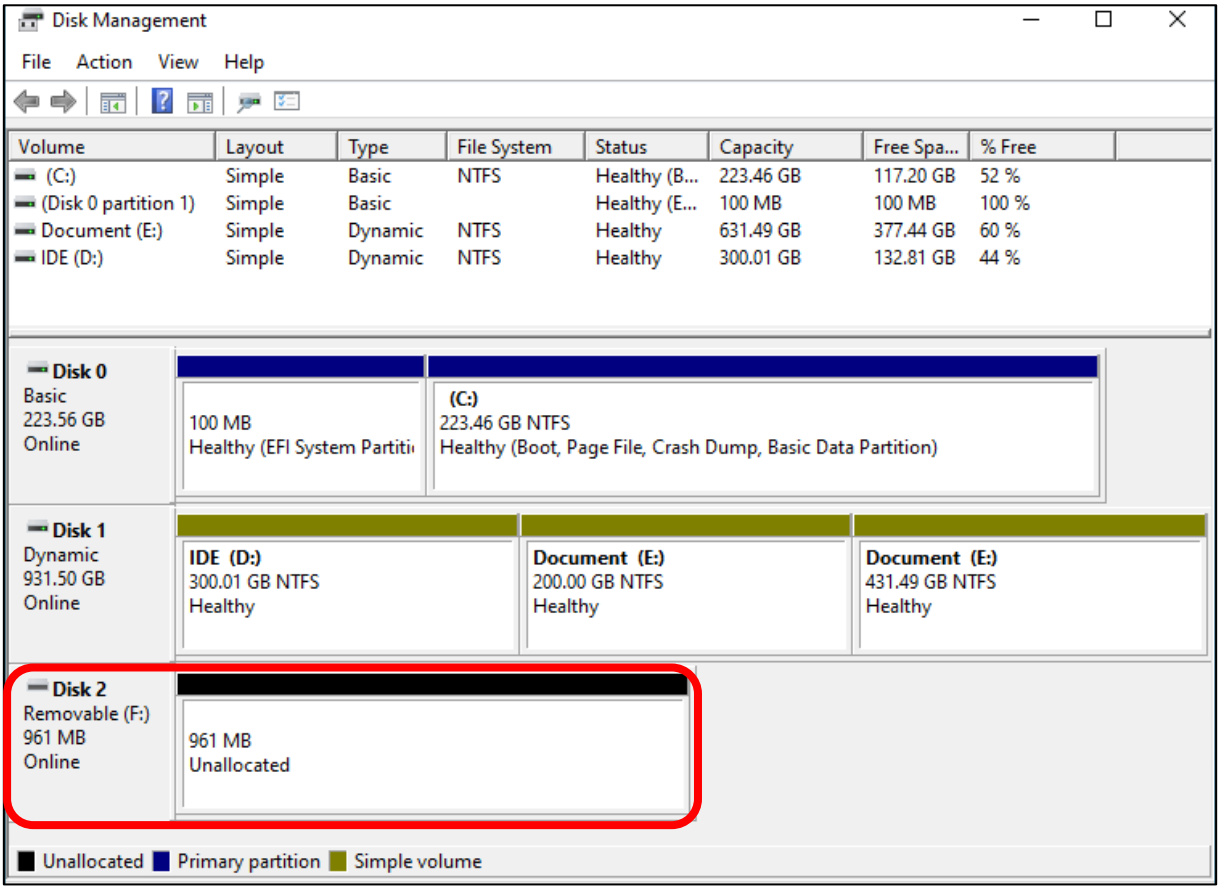
To initiate the project, you'll first need to prepare a blank SD card and a compatible card reader. The initial setup involves assigning a drive letter to the SD card and formatting it properly. Below you'll find system-specific instructions to complete these preparatory steps. Please follow the guide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

Windows

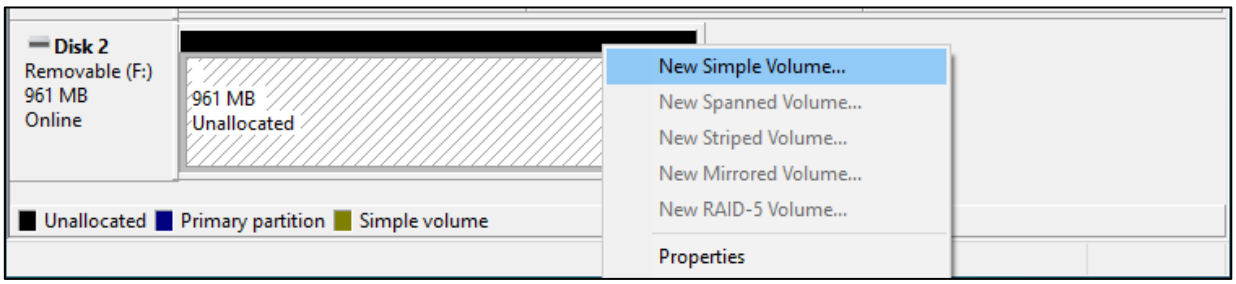
Insert the SD card into the card reader, and then insert the card reader into the computer.
In the Windows search box, enter "Disk Management" and select "Create and format hard disk partitions".



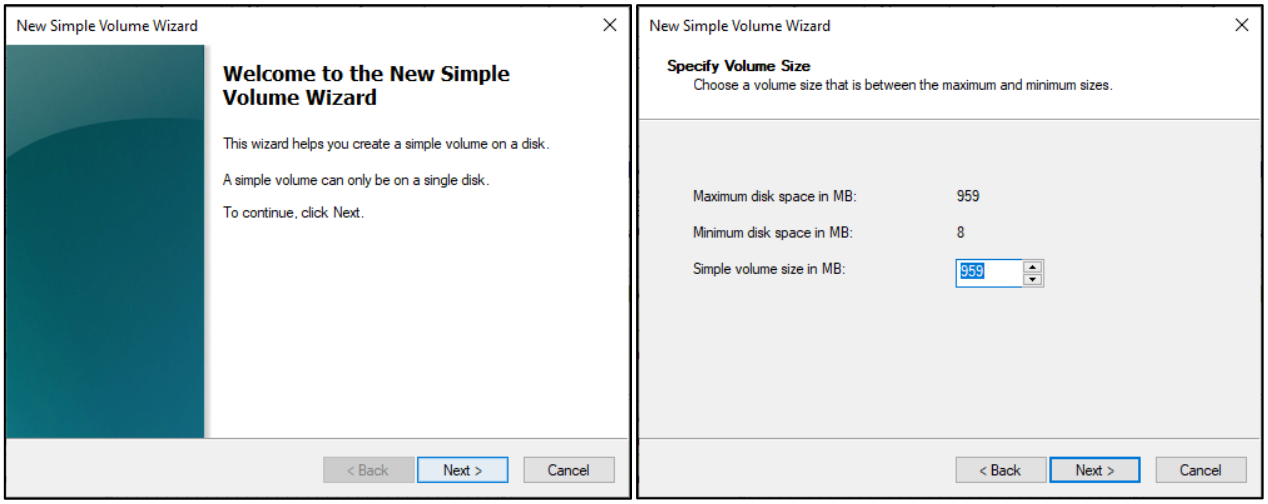
In the newly opened window, locate an unallocated volume with a size close to 1GB.



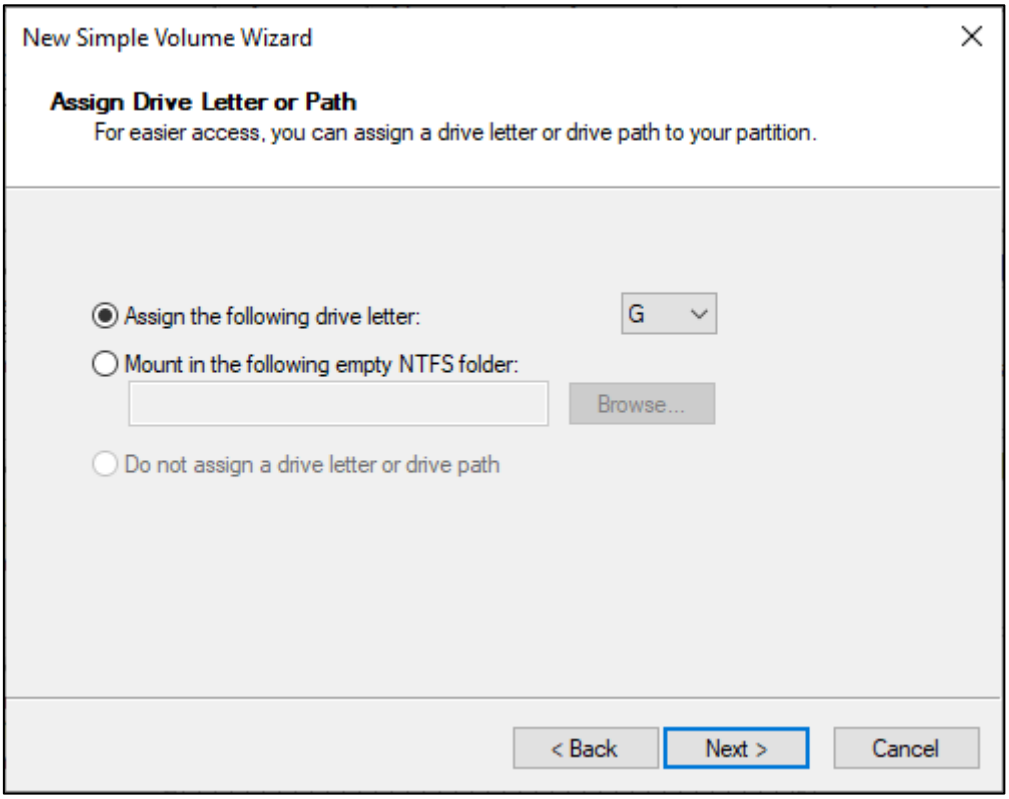
Click to select the volume, right-click and select "New Simple Volume".



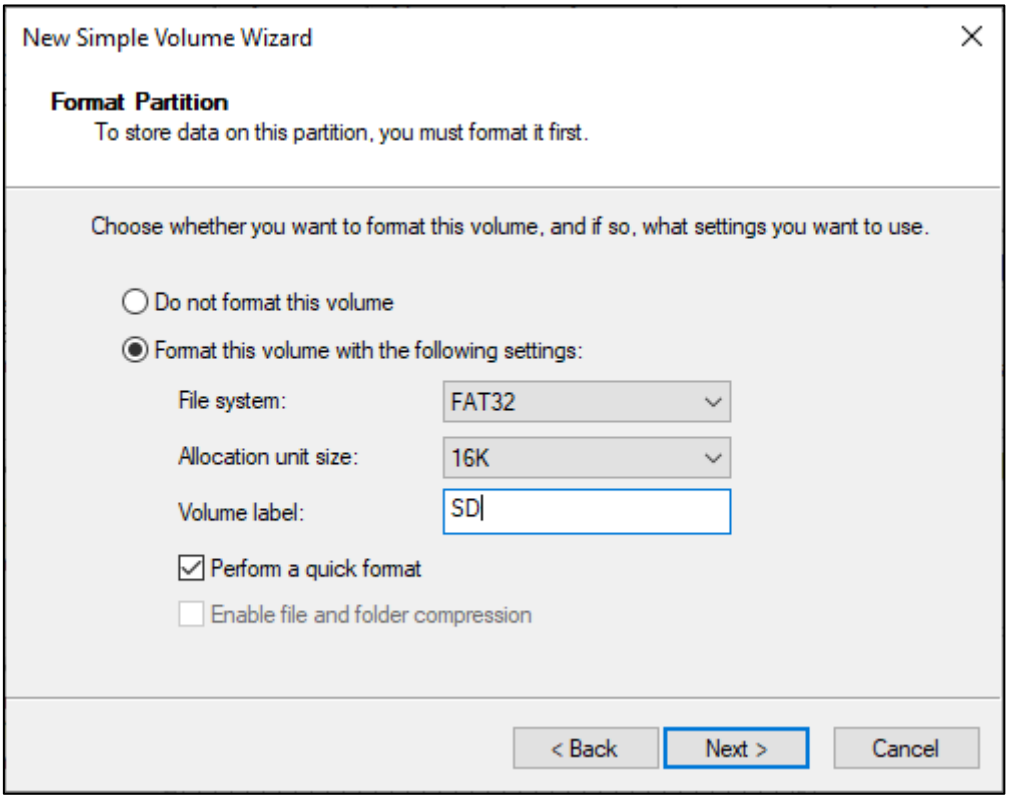
Click Next.



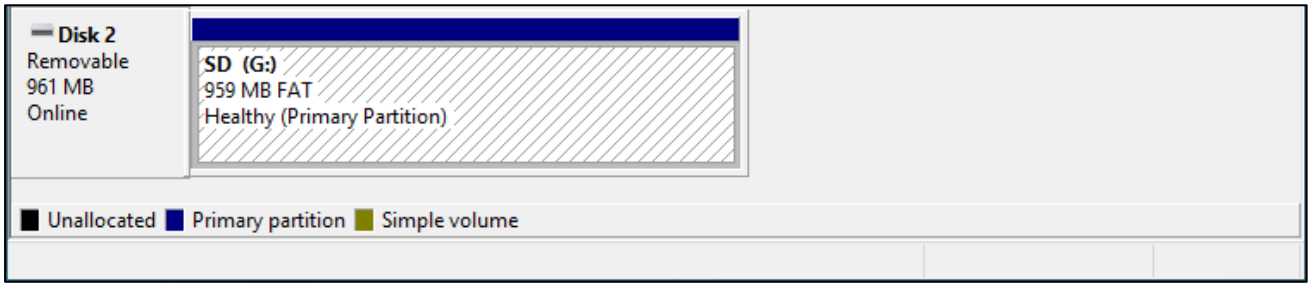
On the right-hand side, you can select a preferred drive letter for the SD card or simply proceed with the default setting by clicking on the "Next" button.



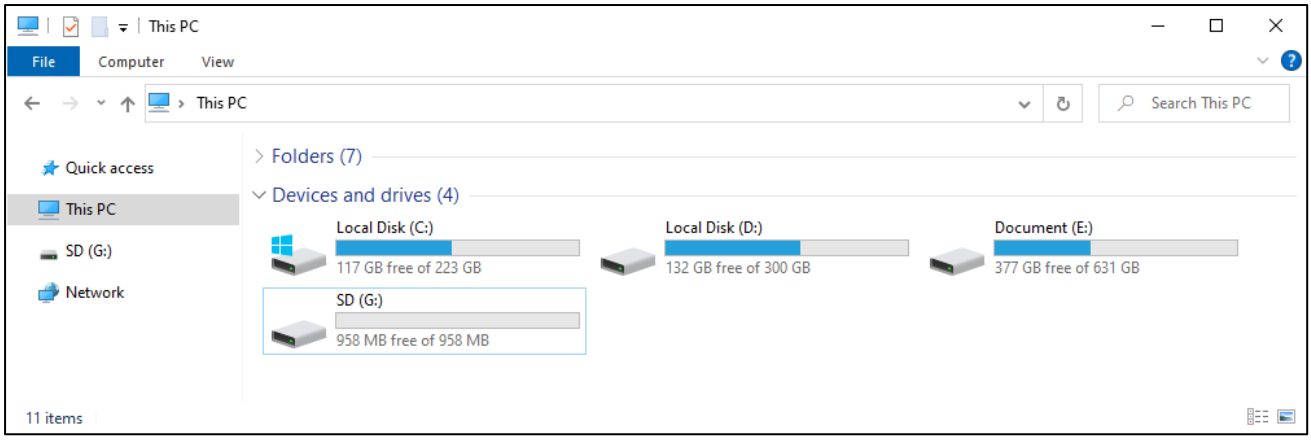
When formatting the SD card, select the file system as FAT (or FAT32) and set the allocation unit size to 16K. You can set the volume label to any name of your choice. Once you have made these selections, click on the "Next" button to proceed.



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

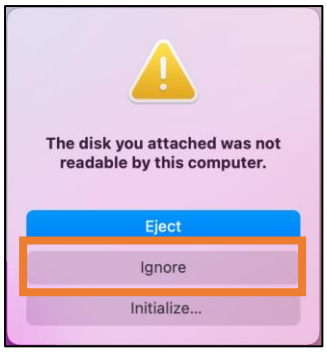


After completing the formatting process, you should be able to see the SD card in the "This PC" section of your computer.

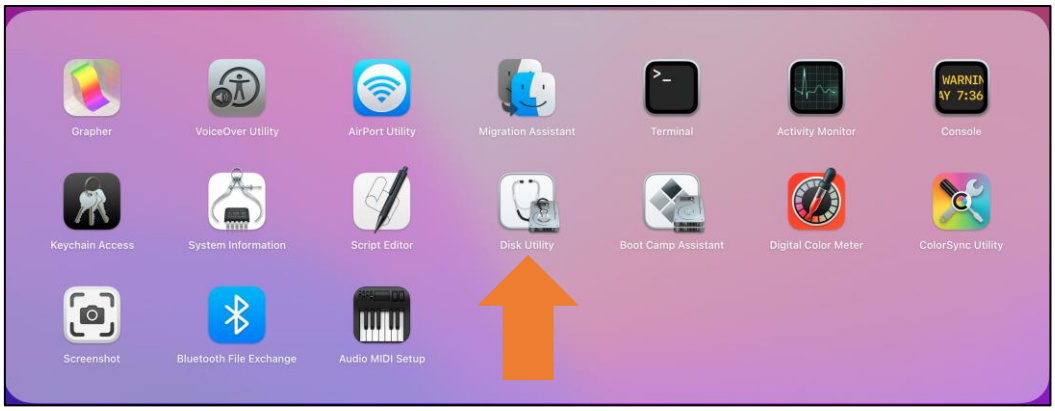


MAC

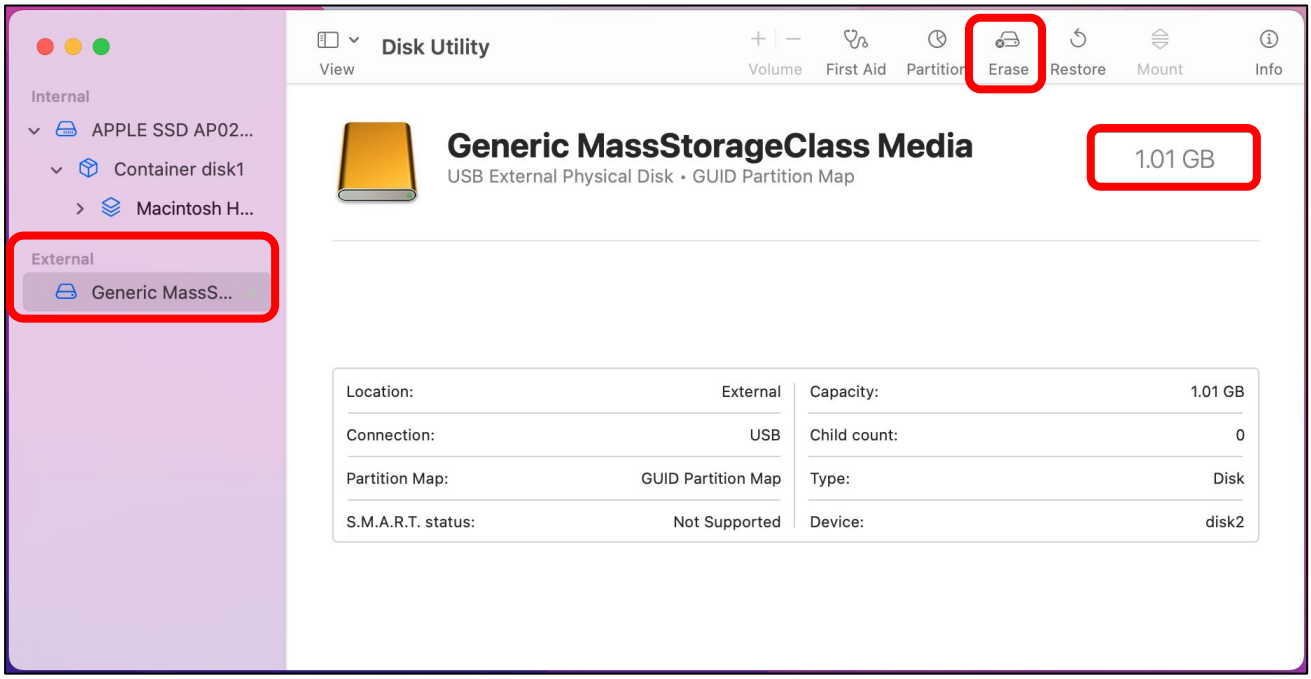
Insert the SD card into the card reader and then insert the card reader into your computer. Some computers may display a prompt with the following message. In this case, please click on the "Ignore" option to proceed.



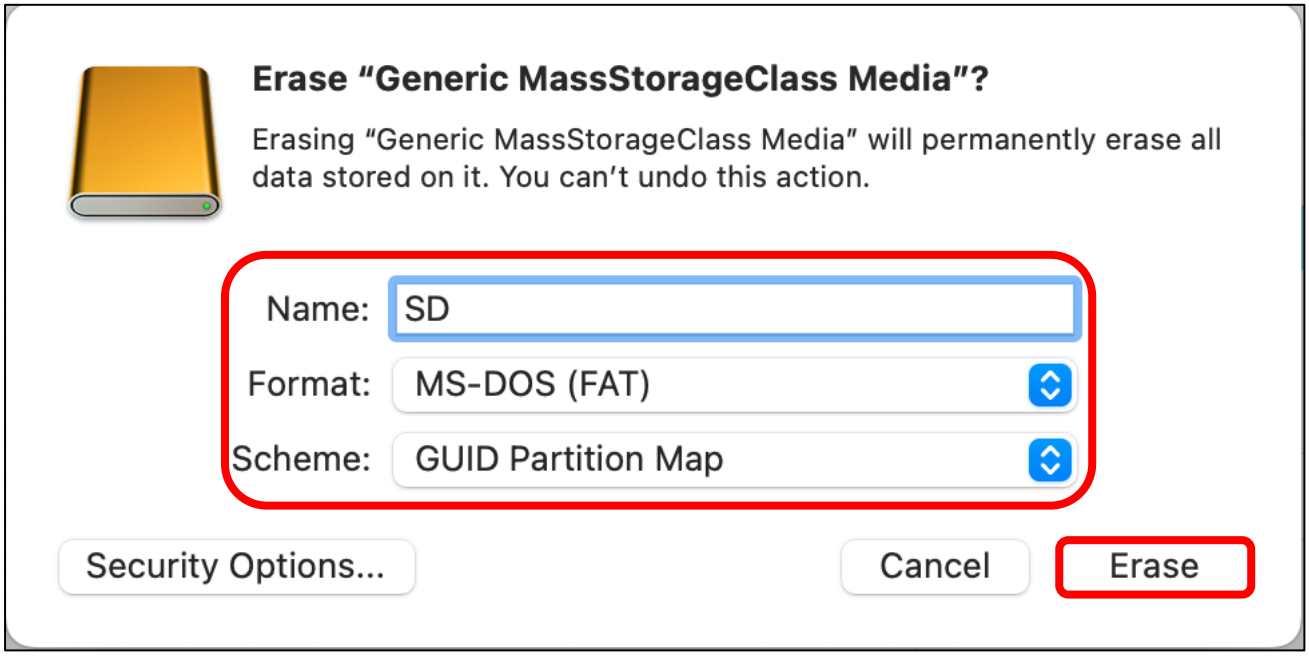
Find "Disk Utility" in the MAC system and click to open it.



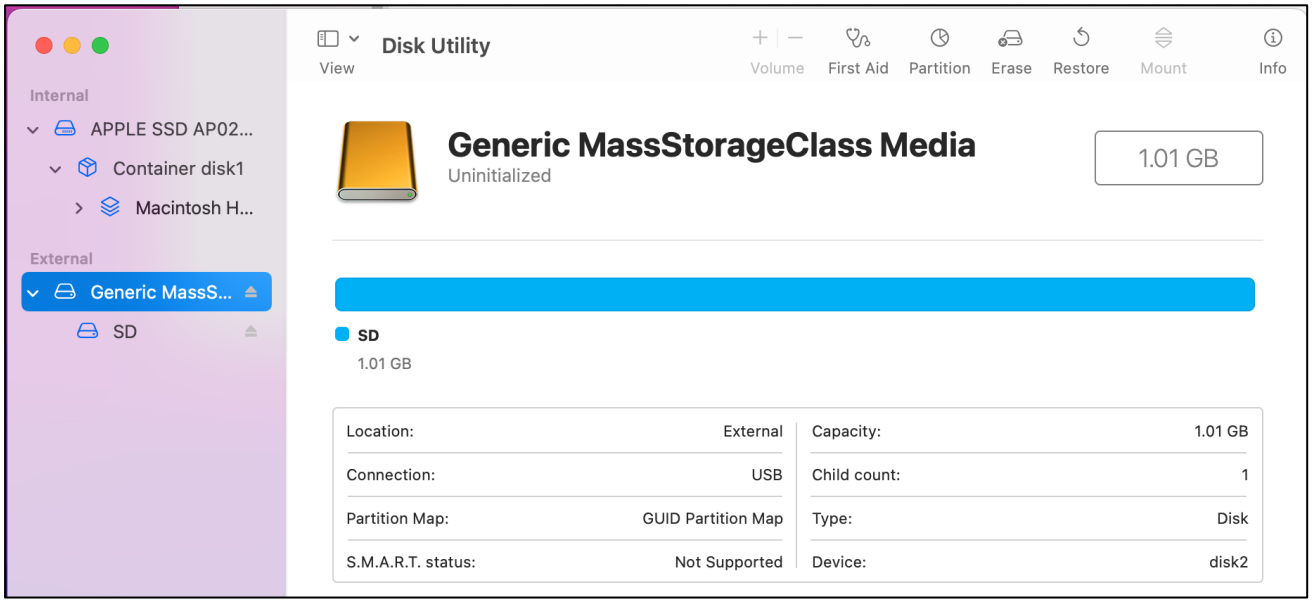
Select "Generic MassStorageClass Media", note that its size is about 1G. It is important to select the correct item to avoid accidentally erasing other device. Once you have verified that you have selected the correct device, click on the "Erase" button to proceed with erasing the SD card.



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

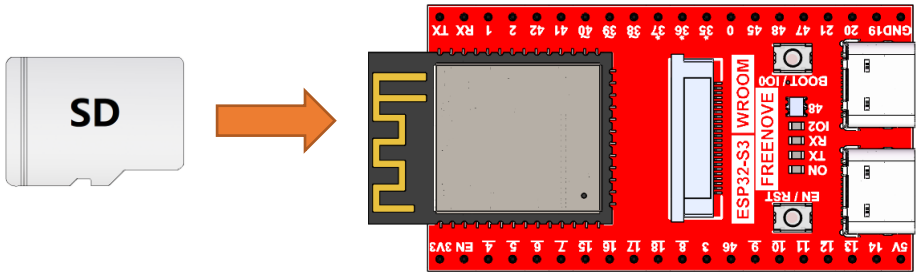


Please wait for the formatting process to complete. Once it is finished, the interface should resemble the image below. You should now be able to see a new disk named "SD" on your desktop.

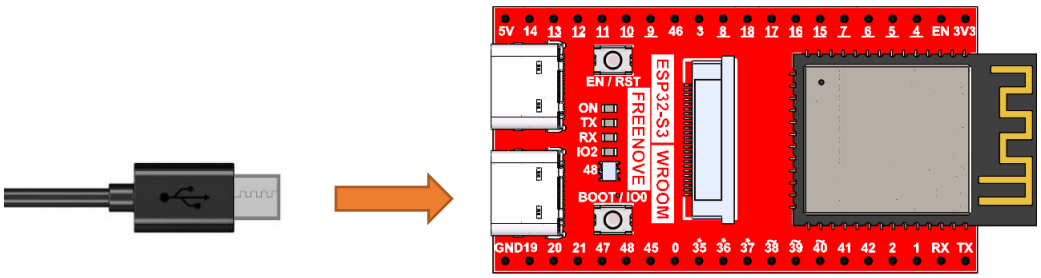


Circuit

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



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

The following is the program code:

```
#include "driver_sdmmc.h"

#define SD_MMC_CMD 38  // Please do not modify it.
#define SD_MMC_CLK 39  // Please do not modify it.
#define SD_MMC_D0 40   // Please do not modify it.

void setup() {
  // Initialize serial communication at 115200 bits per second
  Serial.begin(115200);
  // Initialize the SDMMC interface with the specified pins
  sdmmc_init(SD_MMC_CLK, SD_MMC_CMD, SD_MMC_D0);

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

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

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

  // Write text to a new file
  const char* text = "Hello ";
  write_file("/foo.txt", (const uint8_t*)text, strlen(text));

  // Append text to the existing file
  text = "World!\n";
  append_file("/foo.txt", (const uint8_t*)text, strlen(text));

  // Read the file content into a buffer and print it
  uint8_t buffer[100];
  read_file("/foo.txt", buffer, sizeof(buffer));
  Serial.write(buffer, strlen((char*)buffer));

  // Rename the file and read the new file content
```

```
41    rename_file("/foo.txt", "/hello.txt");
42    read_file("/hello.txt", buffer, sizeof(buffer));
43    Serial.write(buffer, strlen((char*)buffer));
44
45    // Print SD card information
46    Serial.println("\r\nSD card info:");
47    Serial.printf("Total space: %lluMB\r\n", SD_MMC.totalBytes() / (1024 * 1024));
48    Serial.printf("Used space: %lluMB\r\n", SD_MMC.usedBytes() / (1024 * 1024));
49  }
50
51  void loop() {
52    delay(10000);
53  }
```

Add the SD card drive header file.

```
1   #include "driver_sdmmc.h"
```

The drive pins of the SD card are pre-defined and should not be modified as they are fixed. Altering the pins may result in errors or malfunctions while accessing the SD card.

```
3   #define SD_MMC_CMD 38  // Please do not modify it.
4   #define SD_MMC_CLK 39  // Please do not modify it.
5   #define SD_MMC_D0 40   // Please do not modify it.
```

Initialize the serial port function. Sets the drive pin for SDMMC one-bit bus mode.

```
9   Serial.begin(115200);
11  sdmmc_init(SD_MMC_CLK, SD_MMC_CMD, SD_MMC_D0);
```

Call the listDir() function to read the folder and file names in the SD card, and print them out through the serial port. This function can be found in " driver_sdmmc.cpp".

```
14  list_dir("/", 0);
```

Call create_dir () to create a folder, and call remove_dir () to delete a folder.

```
18  create_dir("/mydir");
23  remove_dir("/mydir");
```

Call write_file () to write any content to the txt file. If there is no such file, create this file first.

Call append_file () to append any content to txt.

Call read_file () to read the content in txt and print it via the serial port.

```
29  write_file("/foo.txt", (const uint8_t*)text, strlen(text));
33  append_file("/foo.txt", (const uint8_t*)text, strlen(text));
37  read_file("/foo.txt", buffer, sizeof(buffer));
```

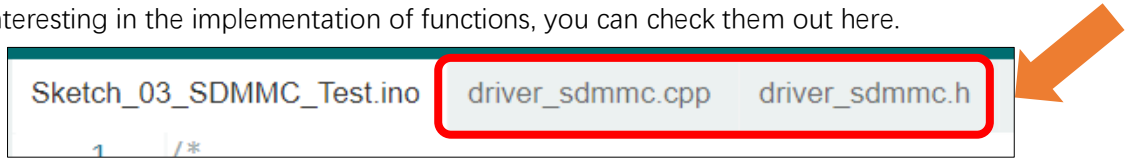
Call renameFile() to copy a file and rename it.

```
41  rename_file("/foo.txt", "/hello.txt");
```

Print the total size and used size of the SD card via the serial port.

```
50  Serial.printf("Total space: %lluMB\r\n", SD_MMC.totalBytes() / (1024 * 1024));
51  Serial.printf("Used space: %lluMB\r\n", SD_MMC.usedBytes() / (1024 * 1024));
```

If you are interesting in the implementation of functions, you can check them out here.



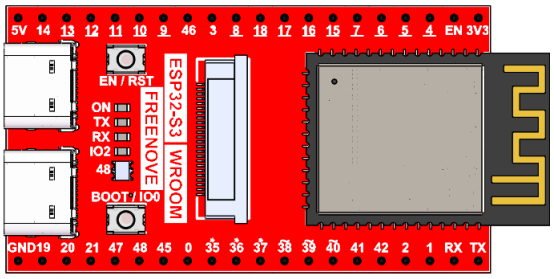
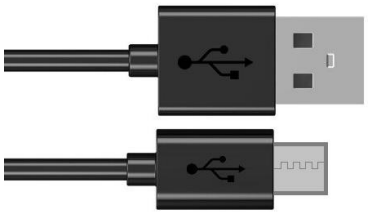
Chapter 5 Simple Tone Test

This chapter will explain how to use the ESP_I2S library to implement I2S audio output and achieve looped playback of a sequence of musical notes.

Project 5.1 Simple Tone

I2S (Inter-IC Sound) is a standardized synchronous serial interface dedicated to digital audio data transfer. Since the microcontroller on the board includes built-in I2S support, we will leverage this capability to produce simple tones.

Component List

Freenove Media Kit for ESP32-S3 x1 	USB cable x1 

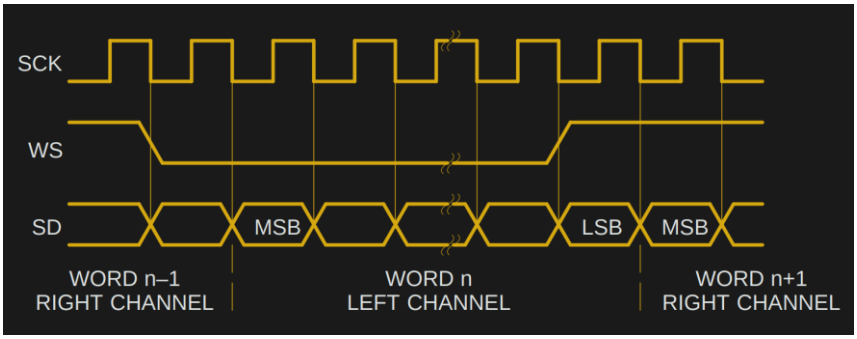
Circuit Knowledge

I2S

I2S achieves high-efficiency data transmission through three signal lines:

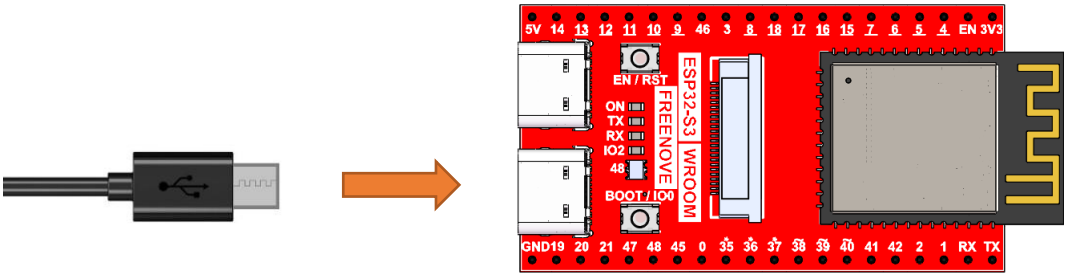
- **SCK** (Serial Clock): Provides synchronization timing, with each bit of audio data corresponding to one SCK pulse.
- **WS** (Word Select): Distinguishes between left and right audio channels. A high WS level indicates left-channel data transmission, while a low WS level represents right-channel transmission.
- **SD** (Serial Data): Carries the actual audio data. The Most Significant Bit (MSB) is transmitted first and always appears at the second SCK pulse after a frame begins (following a WS transition).

The I2S protocol is widely used in audio devices such as DACs, ADCs, and digital microphones, offering low latency and high-fidelity performance.



Circuit

Connect Freenove Media Kit for ESP32-S3 to your computer using the USB cable.



Sketch

Sketch_05_Simple_Tone

The following is the program code:

```
#include <ESP_I2S.h>  // Include ESP_I2S library for I2S audio output

#define I2S_BCLK 42
#define I2S_DOUT 1
#define I2S_LRC 41

// Define note frequencies in Hz for "Do Re Mi Fa So La Ti"
const int frequencies[] = { 261, 293, 329, 349, 392, 440, 493 };
const int amplitude = 500;     // Amplitude of the square wave, controls volume
const int sampleRate = 8000;  // Sampling rate, number of samples per second in Hz

// I2S configuration parameters
i2s_data_bit_width_t bps = I2S_DATA_BIT_WIDTH_16BIT;  // 16 bits per sample
i2s_mode_t mode = I2S_MODE_STD;                       // Use standard I2S mode
i2s_slot_mode_t slot = I2S_SLOT_MODE_STEREO;          // Use stereo mode (left and right
channels)

int currentNote = 0;  // Index of the current note being played, starts at 0 (Do)
int halfWavelength;   // Half wavelength of the square wave, controls the flipping point

int32_t sample = amplitude;  // Current sample value, initialized to amplitude
int count = 0;               // Sample counter, tracks progress of the current note

I2SClass i2s;  // Create I2S object for controlling I2S output

void setup() {

```

```
28    Serial.begin(115200);                       // Initialize serial communication at 115200
29  baud rate
30    Serial.println("I2S simple tone");          // Print debug message
31    i2s.setPins(I2S_BCLK, I2S_LRC, I2S_DOUT);  // Set I2S pins (BCLK, LRCK, DATA)
32
33    // Initialize I2S
34    if (!i2s.begin(mode, sampleRate, bps, slot)) {
35      Serial.println("Failed to initialize I2S!");  // Print error message if initialization
36  fails
37      while (1)
38        ;  // Stop program execution
39    }
40    // // Calculate initial half wavelength for generating the square wave
41    halfWavelength = (sampleRate / frequencies[currentNote]);
42
43    while(currentNote < 7)
44    {
45      if (count % halfWavelength == 0) {
46        sample = -1 * sample;
47      }
48
49      // Write the current sample value to I2S, output to right and left channels
50      i2s.write(sample);  // Right channel
51      i2s.write(sample);  // Left channel
52
53      // Increment the sample counter
54      count++;
55
56      // If the current note finishes playing (1 second), switch to the next note
57      if (count >= sampleRate) {  // sampleRate represents the number of samples in 1 second
58        count = 0;                // Reset the sample counter
59        currentNote++;            // Switch to the next note
60
61        // Update the half wavelength to match the new note frequency
62        if(currentNote < 7)
63          halfWavelength = (sampleRate / frequencies[currentNote]);
64      }
65    }
66  }
67  void loop() {
68    // If the sample counter reaches half wavelength, flip the sample value to generate a
69  square wave
70    delay(1000);
71  }
```

Include the header file for the I2S library.

```
#include <ESP_I2S.h>  // Include ESP_I2S library for I2S audio output
```

Set the frequency, volume and sampling rate of the sound.

```
// Define note frequencies in Hz for "Do Re Mi Fa So La Ti"
const int frequencies[] = { 261, 293, 329, 349, 392, 440, 493 };
const int amplitude = 500;    // Amplitude of the square wave, controls volume
const int sampleRate = 8000;  // Sampling rate, number of samples per second in Hz
```

Configure I2S parameters.

```
// I2S configuration parameters
i2s_data_bit_width_t bps = I2S_DATA_BIT_WIDTH_16BIT;  // 16 bits per sample
i2s_mode_t mode = I2S_MODE_STD;                       // Use standard I2S mode
i2s_slot_mode_t slot = I2S_SLOT_MODE_STEREO;      // Use stereo mode (left and right channels)
```

Set I2S pins.

```
i2s.setPins(I2S_BCLK, I2S_LRC, I2S_DOUT);  // Set I2S pins (BCLK, LRCK, DATA)
```

The I2S initialization function.

```
i2s.begin(mode, sampleRate, bps, slot)
```

Output left and right channel audio samples through I2S interface

```
// Write the current sample value to I2S, output to right and left channels
i2s.write(sample);  // Right channel
i2s.write(sample);  // Left channel
```

Calculate the half wavelength corresponding to the frequency based on the frequency and sampling rate of the current note,

```
halfWavelength = (sampleRate / frequencies[currentNote]);
```

Click the Upload button to upload the sketch.

Once the code is uploaded, the speaker will play the Do-Re-Mi-Fa-Sol-La-Si-Do scale sequence

Reference

```
I2SClass:setPins (int8_t bclk, int8_t ws, int8_t dout, int8_t din = -1, int8_t mclk = -1);
```

This function configures the I2S interface pins

Parameters:

bclk: Bit clock pin

ws: Word select (channel selection) pin

dout: Data output pin

din: Data input pin (optional)

mclk: Master clock pin (optional)

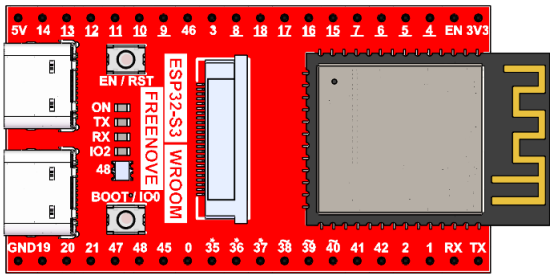
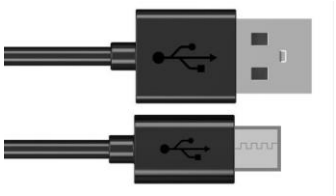
I2SClass:begin (i2s_mode_t mode, uint32_t rate, i2s_data_bit_width_t bits_cfg, i2s_slot_mode_t ch, int8_t slot_mask = -1);
I2SThis function initialize I2S. **Paramters：** Mode: I2S working mode Rate: sampling rate bits_cfg: data bit width I2S_DATA_BIT_WIDTH_8BIT = 8, /*!< I2S channel data bit-width: 8 */ I2S_DATA_BIT_WIDTH_16BIT = 16, /*!< I2S channel data bit-width: 16 */ I2S_DATA_BIT_WIDTH_24BIT = 24, /*!< I2S channel data bit-width: 24 */ I2S_DATA_BIT_WIDTH_32BIT = 32, /*!< I2S channel data bit-width: 32 */ Ch: channel modes I2S_SLOT_MODE_MONO = 1, I2S_SLOT_MODE_STEREO = 2, slot_mask： slot mask (optional)

Chapter 6 Play MP3 Test

In the previous study, we have learned how to use the SD card. Now we are going to learn to play the music in the SD card.

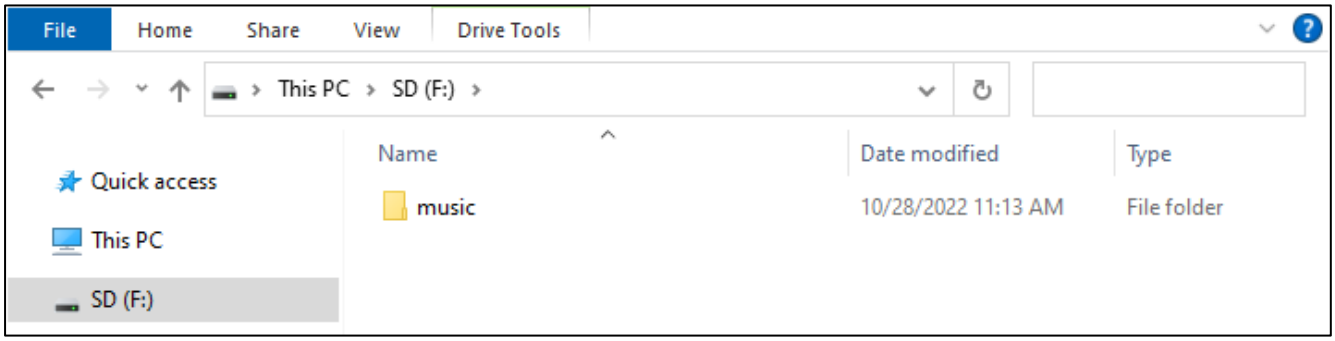
Project 6.1 Play MP3

In this project, we will read files in mp3 format from SD card, decode them through ESP32-S3, and use Audio Converter & Amplifier module to transcode into stereo output.

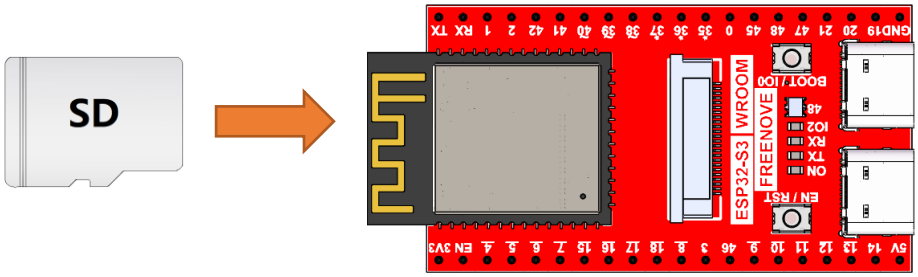
Freenove Media Kit for ESP32-S3 x1 	USB cable x1 
SD card x1 	Card reader x1 (random color) 

Circuit

Before compling code, we should copy music to SD card first, as illustrated below.
We have placed a folder called "music" in:
Freenove_Media_Kit_for_ESP32-S3\Sketch\Sketch_05_I2S_Audio,
please copy this folder to the SD card.



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

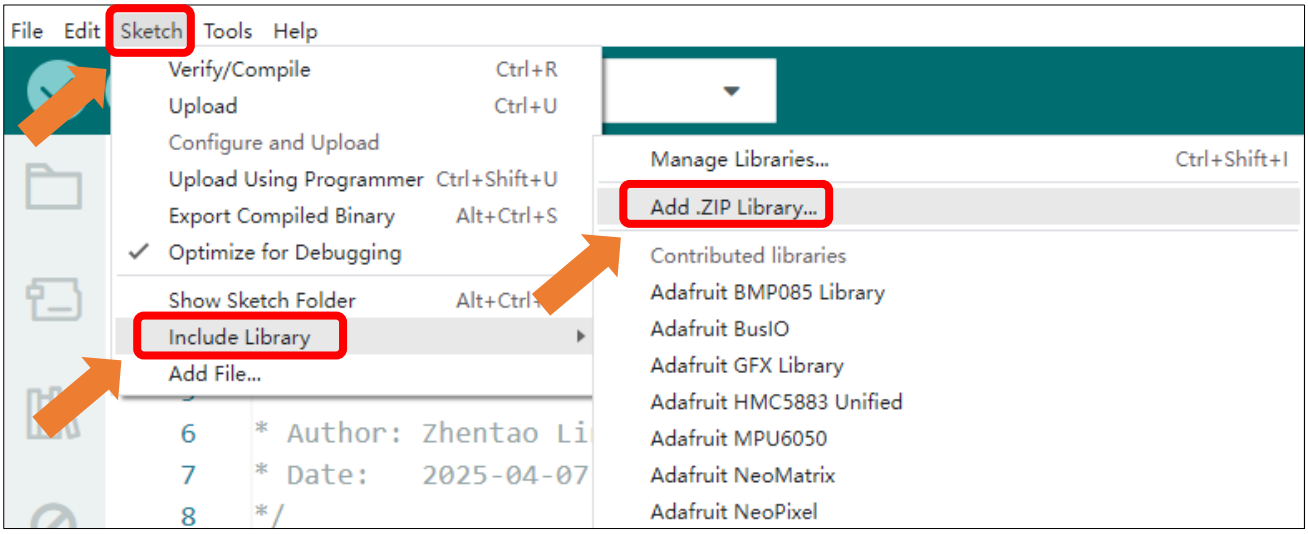


Sketch

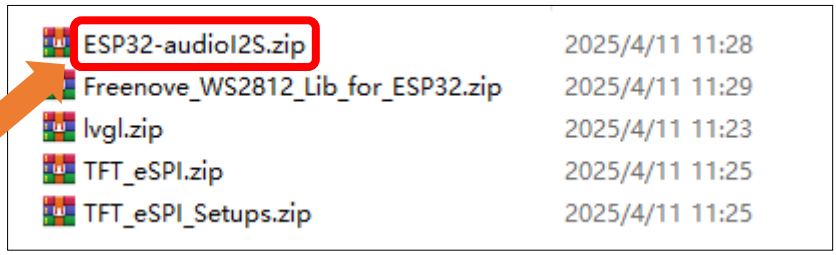
How to install the library

In this project, we will use the ESP32-audioI2S.zip library to decode the audio files in the SD card, and then output the audio signal through IIS. If you have not installed this library, please follow the steps below to install it.

Open **Arduino IDE->Sketch->Include library-> Add .ZIP Library**.



In the newly opened window, select " **Freenove_Media_Kit_for_ESP32-S3\Libraries\ESP32-audioI2S.zip**" and click "Open".



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

Sketch_05_Play_MP3

The following is the program code:

```
#include "Arduino.h"
#include "Audio.h"
#include "FS.h"
#include "SD_MMC.h"

#define SD_MMC_CMD 38  // Command pin for SDMMC
#define SD_MMC_CLK 39  // Clock pin for SDMMC
#define SD_MMC_D0  40  // Data pin for SDMMC

#define I2S_BCLK   42  // Bit clock pin for I2S
#define I2S_DOUT   1   // Data out pin for I2S
#define I2S_LRC    41  // Left/Right clock pin for I2S

Audio audio;

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

 SD_MMC.setPins(SD_MMC_CLK, SD_MMC_CMD, SD_MMC_D0);
 if (!SD_MMC.begin("/sdcard", true, true, SDMMC_FREQ_DEFAULT, 5)) {
   Serial.println("Card Mount Failed");
   return;
 }
 uint8_t cardType = SD_MMC.cardType();
 if (cardType == CARD_NONE) {
   Serial.println("No SD_MMC card attached");
   return;
 }
 if (cardType == CARD_MMC) {
   Serial.println("MMC");
 } else if (cardType == CARD_SD) {
   Serial.println("SDSC");
 } else if (cardType == CARD_SDHC) {
   Serial.println("SDHC");
 } else {
   Serial.println("UNKNOWN");
 }
 uint64_t cardSize = SD_MMC.cardSize() / (1024 * 1024);
 Serial.printf("SD_MMC Card Size: %lluMB\n", cardSize);

 audio.setPinout(I2S_BCLK, I2S_LRC, I2S_DOUT);
 audio.setVolume(2);  // 0...21
```

```
43
44    audio.connecttoFS(SD_MMC, "/music/Jingle Bells.mp3");
45  }
46
47  void loop() {
48    audio.loop();
49    if (Serial.available()) {  // put streamURL in serial monitor
50      audio.stopSong();
51      String r = Serial.readString();
52      r.trim();
53      if (r.length() > 5) audio.connecttoFS(SD_MMC, r.c_str());
54      log_i("free heap=%i", ESP.getFreeHeap());
55    }
56  }
57
58  // optional
59  void audio_info(const char *info) {
60    Serial.print("info        ");
61    Serial.println(info);
62  }
63  void audio_id3data(const char *info) {  //id3 metadata
64    Serial.print("id3data     ");
65    Serial.println(info);
66  }
67  void audio_eof_mp3(const char *info) {  //end of file
68    Serial.print("eof_mp3     ");
69    Serial.println(info);
70  }
71  void audio_showstation(const char *info) {
72    Serial.print("station     ");
73    Serial.println(info);
74  }
75  void audio_showstreamtitle(const char *info) {
76    Serial.print("streamtitle ");
77    Serial.println(info);
78  }
79  void audio_bitrate(const char *info) {
80    Serial.print("bitrate     ");
81    Serial.println(info);
82  }
83  void audio_commercial(const char *info) {  //duration in sec
84    Serial.print("commercial  ");
85    Serial.println(info);
86  }
```

```
87  void audio_icyurl(const char *info) {  //homepage
88   Serial.print("icyurl      ");
89   Serial.println(info);
90  }
91  void audio_lasthost(const char *info) {  //stream URL played
92   Serial.print("lasthost    ");
93   Serial.println(info);
94  }
```

Include the required header files.

```
1  #include "Arduino.h"
2  #include "Audio.h"
3  #include "FS.h"
4  #include "SD_MMC.h"
```

Macro definitions for SD Card pin configuration

```
6  #define SD_MMC_CMD 38  // Command pin for SDMMC
7  #define SD_MMC_CLK 39  // Clock pin for SDMMC
8  #define SD_MMC_D0  40  // Data pin for SDMMC
```

Macro definitions for I2S pin configuration

```
10  #define I2S_BCLK   42  // Bit clock pin for I2S
11  #define I2S_DOUT   1   // Data out pin for I2S
12  #define I2S_LRC    41  // Left/Right clock pin for I2S
```

SD card initialization

```
19  SD_MMC.setPins(SD_MMC_CLK, SD_MMC_CMD, SD_MMC_D0);
20  if (!SD_MMC.begin("/sdcard", true, true, SDMMC_FREQ_DEFAULT, 5)) {
21   Serial.println("Card Mount Failed");
22   return;
23  }
```

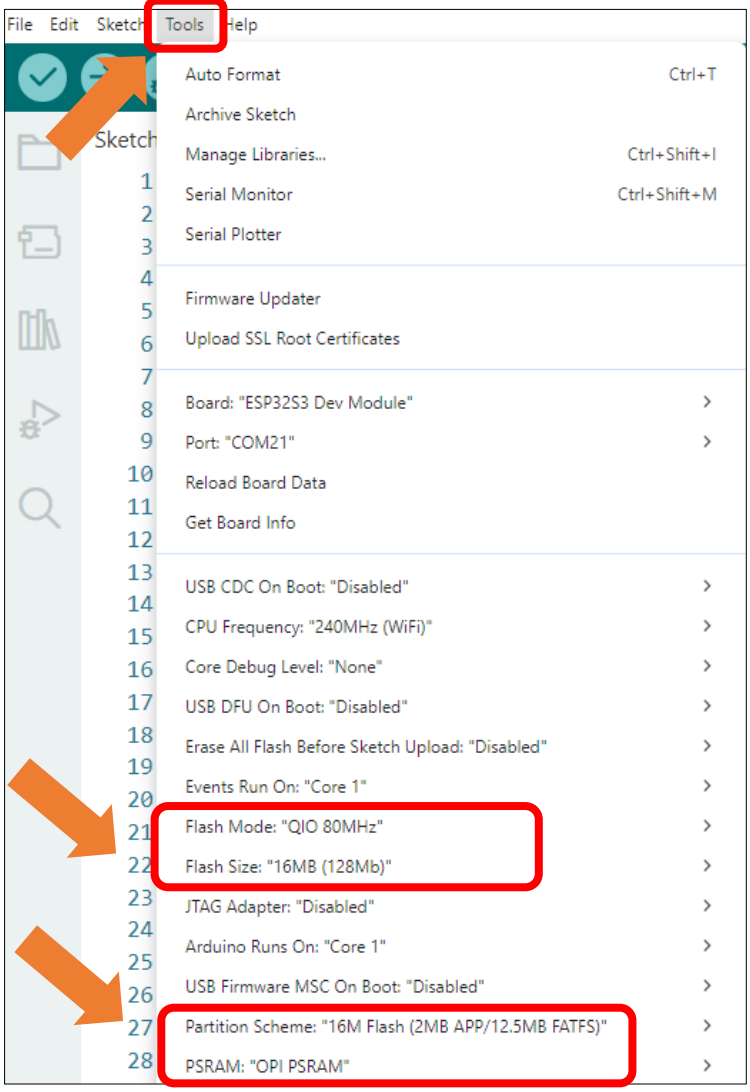
Declare an audio decoding object, associate it with the pin, set the volume, and set the decoding object.

```
14  Audio audio;
    ...
41  audio.setPinout(I2S_BCLK, I2S_LRC, I2S_DOUT);
42  audio.setVolume(2);  // 0...21
43
44  audio.connecttoFS(SD_MMC, "/music/Jingle Bells.mp3");
```

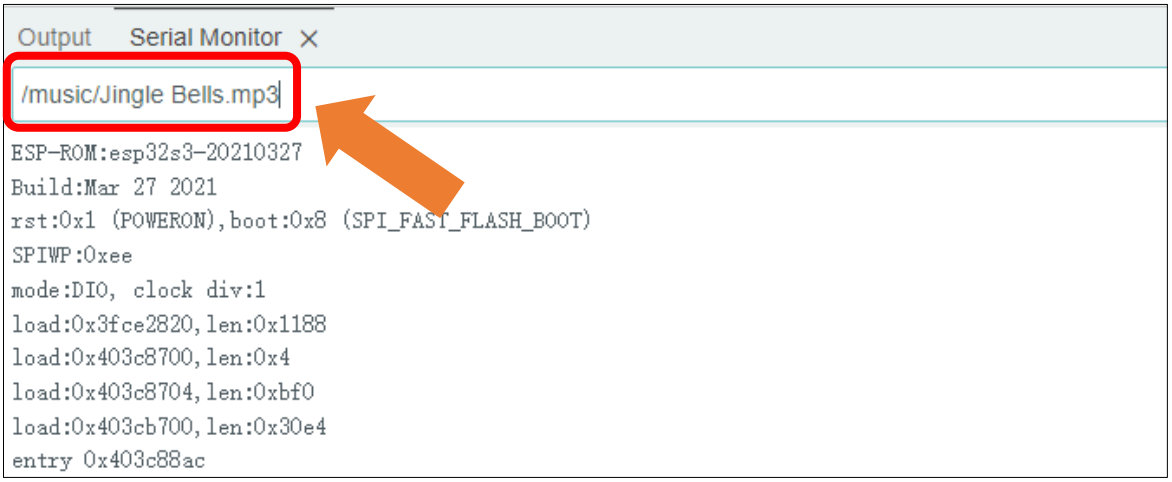
Continuously play music until the current track finishes. Upon receiving data through the serial port, remove any leading or trailing spaces and use the audio object to decode the data.

```
45  audio.loop();
46  if (Serial.available()) {  // put streamURL in serial monitor
47    audio.stopSong();
48    String r = Serial.readString();
49    r.trim();
50    if (r.length() > 5) audio.connecttoFS(SD_MMC, r.c_str());
51    log_i("free heap=%i", ESP.getFreeHeap());
52  }
```

It is necessary to change the settings in Arduino IDE before clicking the Uploading button, as shown below.
Caution: Incorrect settings will result in compilation error or uploading failure. To achieve desired result, please configure exactly the same as below.



If you want to switch the music in the SD card, you can directly input the song through the serial port.



To adjust the volume, you can modify the parameter in the code, as shown below.

The range of the volume is between 0 to 21.

```
51  audio.setVolume(2);  // 0...21
```

These functions are used to print information about audio decoding. If you don't want to see this information in the serial port, you can simply comment out these functions."

```
59  void audio_info(const char *info);
63  void audio_id3data(const char *info);
67  void audio_eof_mp3(const char *info);
71  void audio_showstation(const char *info);
75  void audio_showstreamtitle(const char *info);
79  void audio_bitrate(const char *info);
83  void audio_commercial(const char *info);
87  void audio_icyurl(const char *info);
91  void audio_lasthost(const char *info);
```

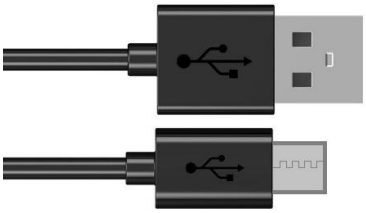
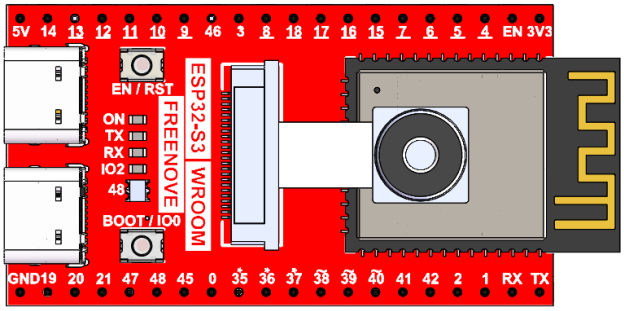
Chapter 7 Video Web Server

In this section, we take ESP32-S3's video function as an example to study.

Project 7.1 Camera Web Server

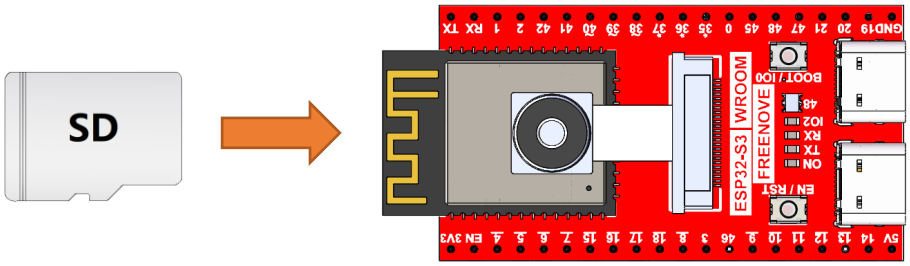
Connect the ESP32-S3 to your computer with the USB cable. Check its IP address through the serial monitor upon the code upload completes. Access the IP address via a browser to view the video and image data.

Component List

SD card x1 	USB cable x1 
Freenove Media Kit for ESP32-S3 x1 	

Circuit

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



Sketch

Sketch_06_Video_Web_Server

The following is the program code:

```
#include "esp_camera.h"
#include <WiFi.h>
#include "driver_sdmmc.h"

// Select camera model
#define CAMERA_MODEL_ESP32S3_EYE  // Has PSRAM
#include "camera_pins.h"

const char* ssid = "********";         // Input your Wi-Fi name
const char* password = "**********";  // Input your Wi-Fi password

#define SD_MMC_CMD 38  // Please do not modify it.
#define SD_MMC_CLK 39  // Please do not modify it.
#define SD_MMC_D0 40   // Please do not modify it.

void camera_init(void);
void startCameraServer();

void setup() {
 Serial.begin(115200);
 while (!Serial);
 Serial.setDebugOutput(true);
 Serial.println();

 // Initialize the camera
 camera_init();
 // Initialize the SDMMC interface for SD card operations
 sdmmc_init(SD_MMC_CLK, SD_MMC_CMD, SD_MMC_D0);
 // Remove and create a new directory for video storage
 remove_dir("/video");
 create_dir("/video");

 // Connect to Wi-Fi network
 WiFi.begin(ssid, password);

 // Wait for Wi-Fi connection
 while (WiFi.status() != WL_CONNECTED) {
   delay(500);
   Serial.print(".");
```

```
  }
  Serial.println("");
  Serial.println("WiFi connected");

  // Start the camera web server
  startCameraServer();

  // Print the IP address to connect to the web server
  Serial.print("Camera Ready! Use 'http://");
  Serial.print(WiFi.localIP());
  Serial.println("' to connect");
}

void loop() {
  // Main loop code, can be used for additional tasks
  delay(10000);
}

void camera_init(void) {
  camera_config_t config;
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = Y2_GPIO_NUM;
  config.pin_d1 = Y3_GPIO_NUM;
  config.pin_d2 = Y4_GPIO_NUM;
  config.pin_d3 = Y5_GPIO_NUM;
  config.pin_d4 = Y6_GPIO_NUM;
  config.pin_d5 = Y7_GPIO_NUM;
  config.pin_d6 = Y8_GPIO_NUM;
  config.pin_d7 = Y9_GPIO_NUM;
  config.pin_xclk = XCLK_GPIO_NUM;
  config.pin_pclk = PCLK_GPIO_NUM;
  config.pin_vsync = VSYNC_GPIO_NUM;
  config.pin_href = HREF_GPIO_NUM;
  config.pin_sccb_sda = SIOD_GPIO_NUM;
  config.pin_sccb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn = PWDN_GPIO_NUM;
  config.pin_reset = RESET_GPIO_NUM;
  config.xclk_freq_hz = 10000000;
  config.frame_size = FRAMESIZE_VGA;
  config.pixel_format = PIXFORMAT_JPEG;  // for streaming
  config.grab_mode = CAMERA_GRAB_WHEN_EMPTY;
  config.fb_location = CAMERA_FB_IN_PSRAM;
  config.jpeg_quality = 12;
```

```
  config.fb_count = 1;

  // If PSRAM IC is present, use higher resolution and quality
  if (psramFound()) {
    config.jpeg_quality = 10;
    config.fb_count = 2;
    config.grab_mode = CAMERA_GRAB_LATEST;
  } else {
    // Limit the frame size when PSRAM is not available
    config.frame_size = FRAMESIZE_VGA;
    config.fb_location = CAMERA_FB_IN_DRAM;
  }

  // Initialize the camera with the specified configuration
  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    Serial.printf("Camera init failed with error 0x%x", err);
    return;
  }

  // Get the camera sensor and adjust settings
  sensor_t* s = esp_camera_sensor_get();
  s->set_vflip(s, 0);       // Flip the image vertically
  s->set_brightness(s, 1);  // Increase brightness
  s->set_saturation(s, 0);  // Decrease saturation
}
```

Add procedure files and API interface files related to ESP32-S3 camera.

```
// Start the camera web server
startCameraServer();

// Print the IP address to connect to the web server
Serial.print("Camera Ready! Use 'http://");
Serial.print(WiFi.localIP());
Serial.println("' to connect");
```

Enter the name and password of your router.

```
const char* ssid = "********";         // Input your Wi-Fi name
const char* password = "**********";  // Input your Wi-Fi password
```

Initialize serial port, set baud rate to 115200; open the debug and output function of the serial.

```
  Serial.begin(115200);
  while (!Serial);
  Serial.setDebugOutput(true);
  Serial.println();
```

ESP32-S3 connects to the router and prints a successful connection prompt. If it does not connect succssfully, press the reset key on the ESP32-S3 WROOM.

```
// Connect to Wi-Fi network
WiFi.begin(ssid, password);

// Wait for Wi-Fi connection
while (WiFi.status() != WL_CONNECTED) {
  delay(500);
  Serial.print(".");
}
Serial.println("");
Serial.println("WiFi connected");
```

Start the camera web server and print its IP address via serial port.

```
 // Start the camera web server
 startCameraServer();

 // Print the IP address to connect to the web server
 Serial.print("Camera Ready! Use 'http://");
 Serial.print(WiFi.localIP());
 Serial.println("' to connect");
```

Configure parameters including interface pins of the camera. Note: Changing the pins is not recommended.

```
camera_config_t config;
config.ledc_channel = LEDC_CHANNEL_0;
config.ledc_timer = LEDC_TIMER_0;
config.pin_d0 = Y2_GPIO_NUM;
config.pin_d1 = Y3_GPIO_NUM;
config.pin_d2 = Y4_GPIO_NUM;
config.pin_d3 = Y5_GPIO_NUM;
config.pin_d4 = Y6_GPIO_NUM;
config.pin_d5 = Y7_GPIO_NUM;
config.pin_d6 = Y8_GPIO_NUM;
config.pin_d7 = Y9_GPIO_NUM;
config.pin_xclk = XCLK_GPIO_NUM;
config.pin_pclk = PCLK_GPIO_NUM;
config.pin_vsync = VSYNC_GPIO_NUM;
config.pin_href = HREF_GPIO_NUM;
config.pin_sccb_sda = SIOD_GPIO_NUM;
config.pin_sccb_scl = SIOC_GPIO_NUM;
config.pin_pwdn = PWDN_GPIO_NUM;
config.pin_reset = RESET_GPIO_NUM;
config.xclk_freq_hz = 10000000;
config.frame_size = FRAMESIZE_VGA;
config.pixel_format = PIXFORMAT_JPEG;  // for streaming
config.grab_mode = CAMERA_GRAB_WHEN_EMPTY;
```

```
82  config.fb_location = CAMERA_FB_IN_PSRAM;
83  config.jpeg_quality = 12;
84  config.fb_count = 1;
```

Configure the display image information of the camera.

The set_vflip() function sets whether the image is flipped 180°, with 0 for no flip and 1 for flip 180°.

The set_brightness() function sets the brightness of the image, with values ranging from -2 to 2.

The set_saturation() function sets the color saturation of the image, with values ranging from -2 to 2.

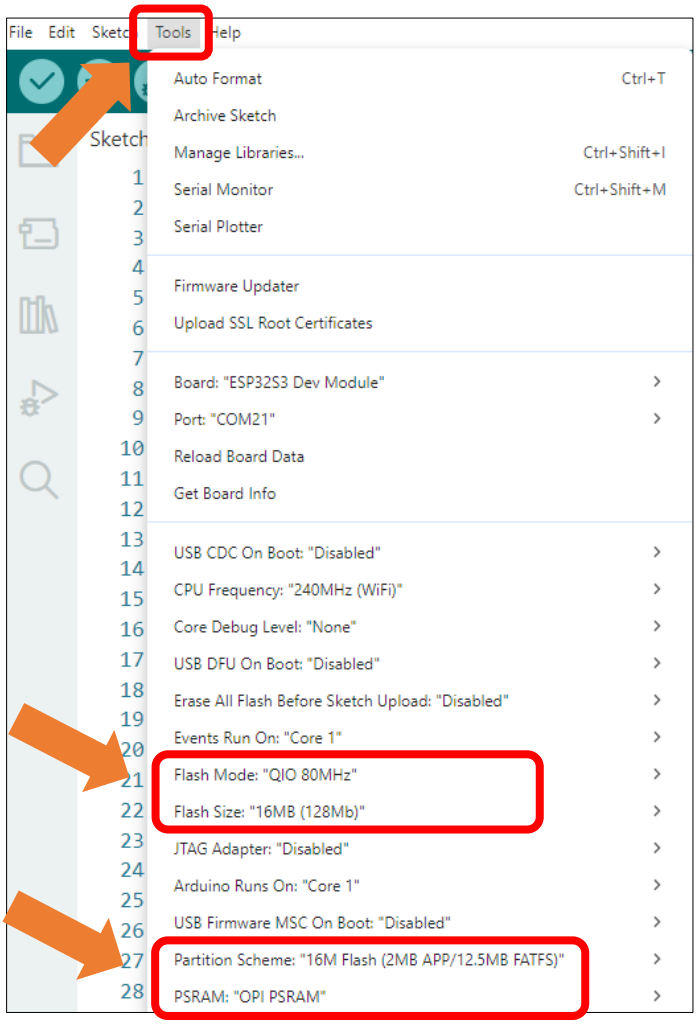
```
104 // Get the camera sensor and adjust settings
105 sensor_t* s = esp_camera_sensor_get();
106 s->set_vflip(s, 0);       // Flip the image vertically
107 s->set_brightness(s, 1);  // Increase brightness
108 s->set_saturation(s, 0);  // Decrease saturation
```

Modify the resolution and sharpness of the images captured by the camera. The sharpness ranges from 10 to 63, and the smaller the number, the sharper the picture. The larger the number, the blurrier the picture. Please refer to the table below.

```
79  config.frame_size = FRAMESIZE_VGA;
83  config.jpeg_quality = 10;
```

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

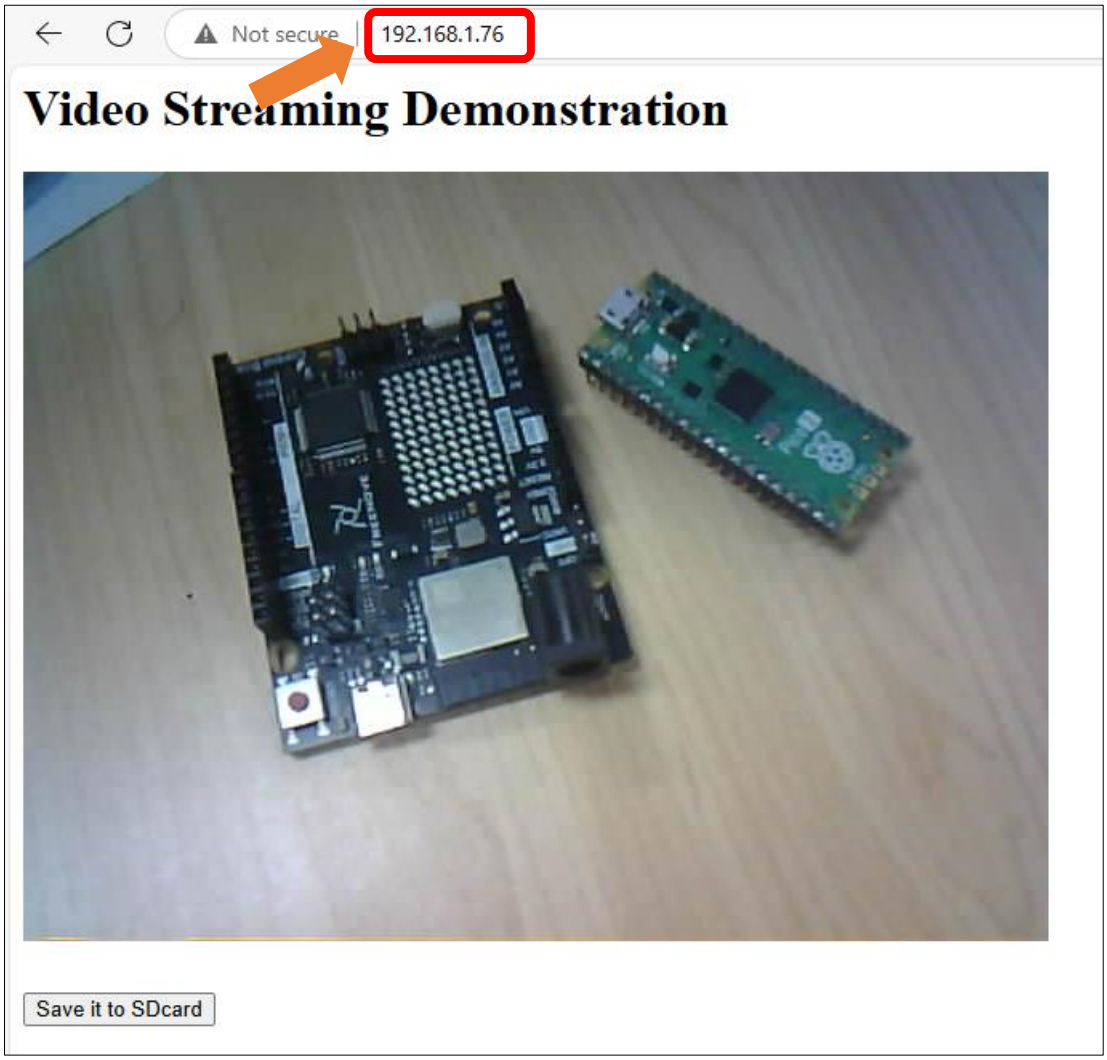
Caution: Incorrect settings will result in compilation error or uploading failure. To achieve desired result, please configure exactly the same as below.



After uploading the code, an IP address will be printed in the Serial Monitor. The IP address is automatically assigned by the system (the specific IP address may vary depending on the network environment).

```
13:47:43.787 -> ESP-ROM:esp32s3-20210327
13:47:43.787 -> Build:Mar 27 2021
13:47:43.787 -> rst:0x1 (POWERON),boot:0x8 (SPI_FAST_FLASH_BOOT)
13:47:43.787 -> SPIWP:0xee
13:47:43.787 -> mode:DIO, clock div:1
13:47:43.787 -> load:0x3fce2820,len:0x1188
13:47:43.787 -> load:0x403c8700,len:0x4
13:47:43.787 -> load:0x403c8704,len:0xbf0
13:47:43.787 -> load:0x403cb700,len:0x30e4
13:47:43.787 -> entry 0x403c88ac
13:47:44.006 ->
13:47:44.280 -> SD_MMC Card Type: SDSC
13:47:44.280 -> SD_MMC Card Size: 960MB
13:47:44.370 -> Total space: 959MB
13:47:44.370 -> Used space: 14MB
13:47:44.951 -> ...
13:47:45.989 -> WiFi connected
13:47:45.989 -> Camera Ready! Use 'http://192.168.1.76' to connect
```

Open the IP address on a browser and you will see the video streaming.



Note: Display performance may differ across camera models, with some devices showing a flipped image. If this occurs, configure the horizontal/vertical mirroring settings.

```
104  s->set_hmirror(s, 1);      // Mirror the image horizontally
105  s->set_vflip(s, 0);        // Restore vertical orientation
```

Parameter Description:

- 0: Normal display
- 1: Flip (mirror)

To achieve the desired display, configure the settings according to real-time preview feedback during setup.

Reference

Image resolution	Sharpness	Image resolution	Sharpness
FRAMESIZE_96X96	96x96	FRAMESIZE_HVGA	480x320
FRAMESIZE_QQVGA	160x120	FRAMESIZE_VGA	640x480
FRAMESIZE_QCIF	176x144	FRAMESIZE_SVGA	800x600
FRAMESIZE_HQVGA	240x176	FRAMESIZE_XGA	1024x768
FRAMESIZE_240X240	240x240	FRAMESIZE_HD	1280x720
FRAMESIZE_QVGA	320x240	FRAMESIZE_SXGA	1280x1024
FRAMESIZE_CIF	400x296	FRAMESIZE_UXGA	1600x1200

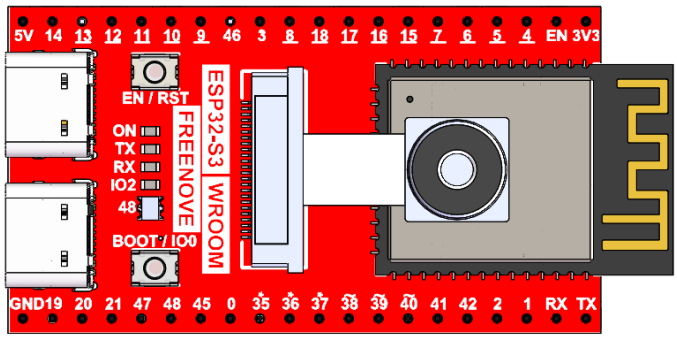
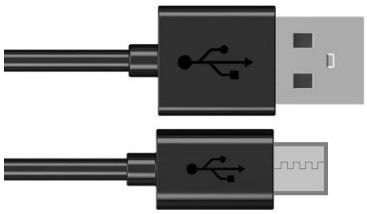
If you have any concerns, please feel free to contact us via support@freenove.com

Chapter 8 TFT Clock

In this section, we will learn how to use the TFT display.

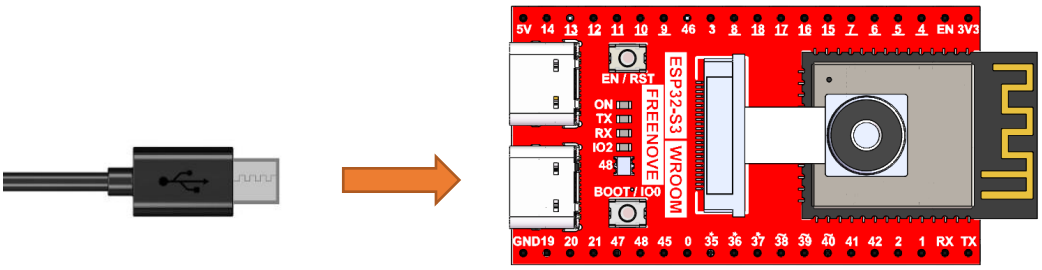
Project 8.1 TFT Clock

Component List

Freenove Media Kit for ESP32-S3 x1 	USB cable x1 

Circuit

Connect Freenove Media Kit for ESP32-S3 to your computer using the USB cable.

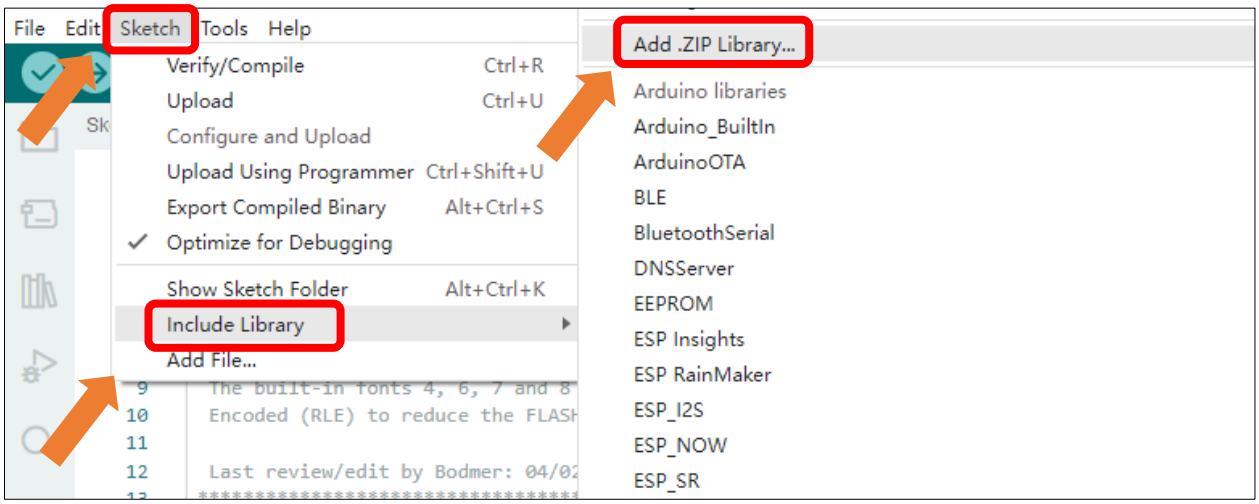


Sketch

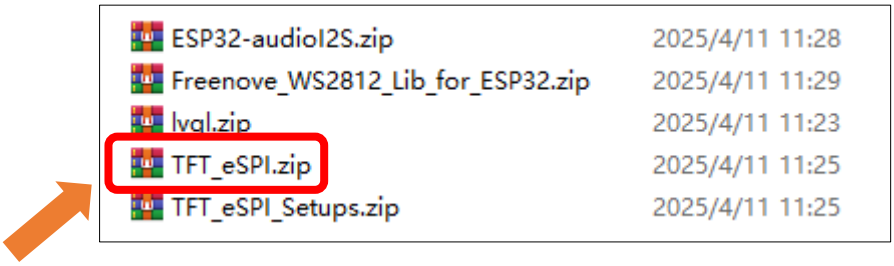
Before uploading the code, you need to isntall the TFT_eSPI library and make some configuration.

How to install the library

Open Arduino IDE, click Sketch->Include Library->Add .ZIP Library. In the pop-up window, find the file named "**Freenove_Media_Kit_for_ESP32-S3\Libraries\TFT_eSPI.Zip**", which locates in this directory, and click OPEN.

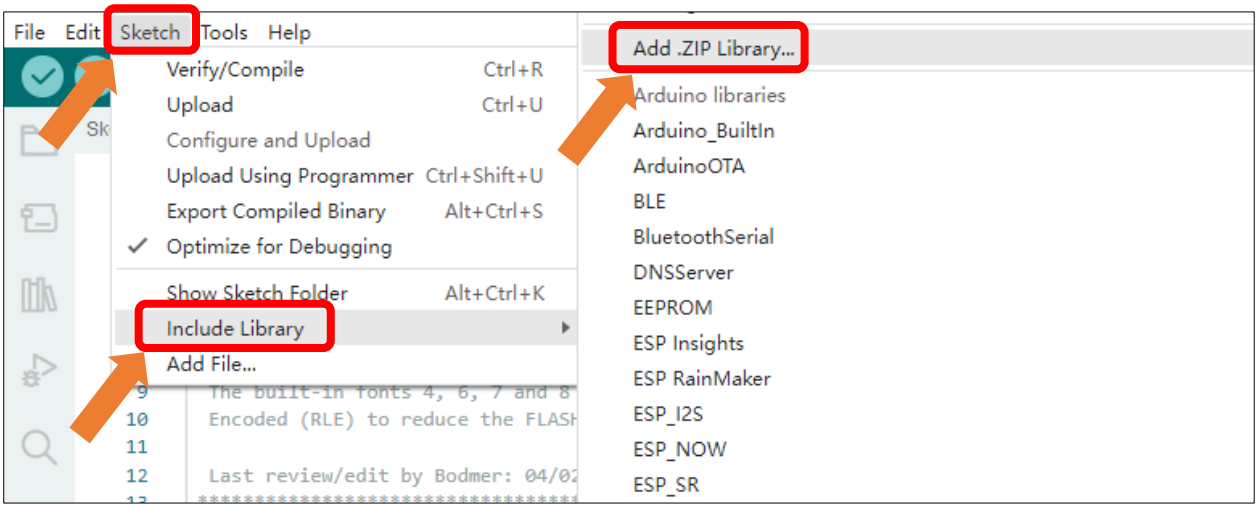


Select **TFT_eSPI.Zip** and click Open.

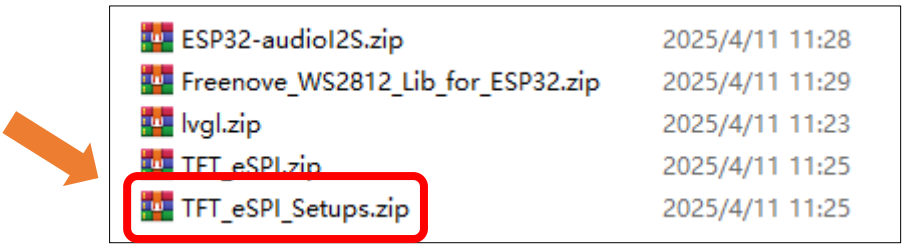


Install the **TFT_eSPI_Setups** library with the same approach. Select TFT_eSPI_Setups.zip that locates in this directory, and click OPEN.

Note: TFT_eSPI_Setups.Zip and TFT_eSPI.Zip are different and both are needed.

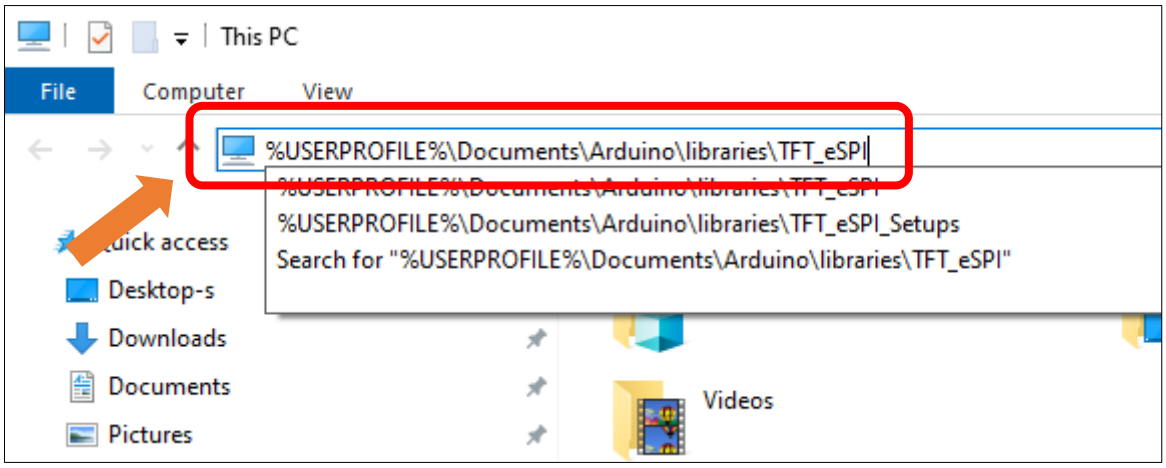


Select **TFT_eSPI_Setups.Zip** and click Open.

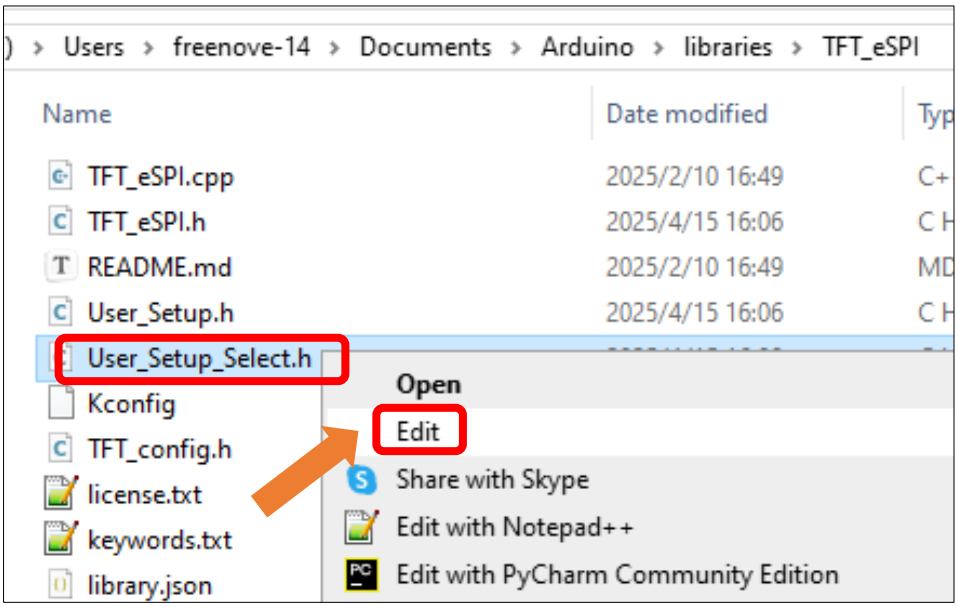


How to configure (Important)

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



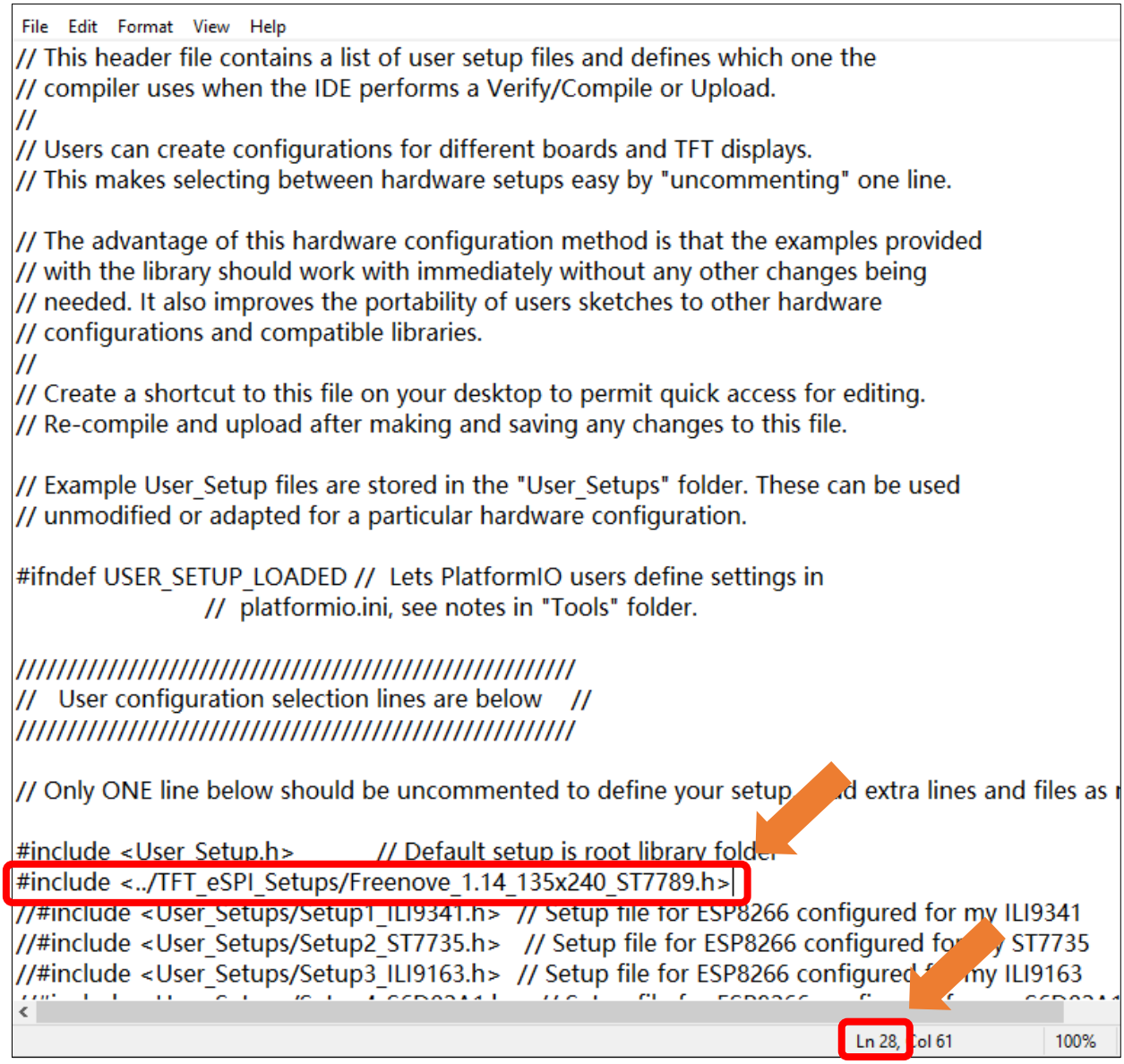
Right click User_Setup_Select.h, click Edit.



Add the following contents to **Line 28** of the file.

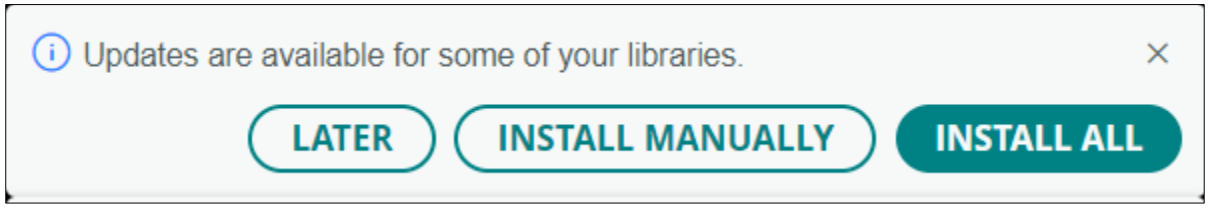
```
1  #include <../TFT_eSPI_Setups/Freenove_1.14_135x240_ST7789.h>
```

Save the change and exit the file.



Warning:

If the following updating message shows up, click LATER. Updating the TFT_eSPI library will reset all related configurations. If you click INSTALL, follow the aforementioned steps to re-add the header file to ensure proper project operation.



Sketch_07_TFT_Clock

The following is the program code:

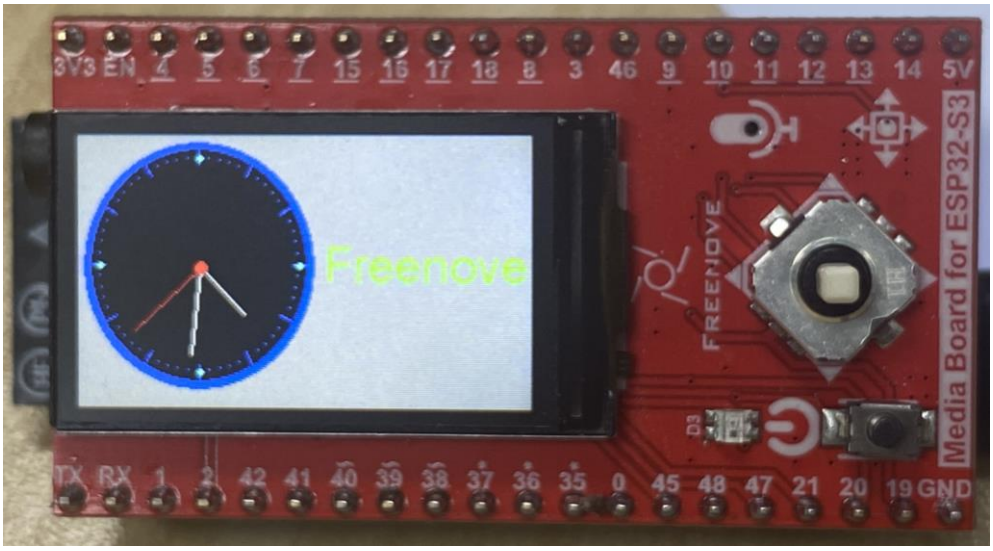
```
#include <TFT_eSPI.h>  // Graphics and font library for ST7735 driver chip
#include <SPI.h>

#define TFT_DIRECTION 1  // Define the direction of the TFT display (0, 1, 2, or 3)

TFT_eSPI tft = TFT_eSPI();  // Invoke library, pins defined in User_Setup.h

#define TFT_GREY 0xBDF7  // Define a custom grey color for the background

// Variables to store the positions and angles of the clock hands
float sx = 0, sy = 1, mx = 1, my = 0, hx = -1, hy = 0;                  // Saved H, M, S x & y multipliers
float sdeg = 0, mdeg = 0, hdeg = 0;                                     // Degrees for second, minute, and hour hands
uint16_t osx = 64, osy = 64, omx = 64, omy = 64, ohx = 64, ohy = 64;  // Saved H, M, S x & y coords
uint16_t x0 = 0, x1 = 0, yy0 = 0, yy1 = 0;                              // Temporary variables for drawing lines
uint32_t targetTime = 0;                                                // Time for the next second timeout

// Function to convert a two-character string to an integer
static uint8_t conv2d(const char *p) {
  uint8_t v = 0;
  if ('0' <= *p && *p <= '9')
    v = *p - '0';
  return 10 * v + *++p - '0';
}

// Get the current time from the compile time
uint8_t hh = conv2d(__TIME__), mm = conv2d(__TIME__ + 3), ss = conv2d(__TIME__ + 6);

bool initial = 1;  // Flag to indicate initial setup

void setup(void) {
  tft.init();                           // Initialize the TFT display
  tft.setRotation(TFT_DIRECTION);       // Set the rotation of the display
  tft.fillScreen(TFT_GREY);             // Fill the screen with grey color
  tft.setTextColor(TFT_GREEN, TFT_GREY);  // Set text color and background color

  // Draw the clock face
  tft.fillCircle(64, 64, 61, TFT_BLUE);   // Outer circle
```

```
  tft.fillCircle(64, 64, 57, TFT_BLACK);  // Inner circle

  // Draw 12 lines for the clock marks
  for (int i = 0; i < 360; i += 30) {
    sx = cos((i - 90) * 0.0174532925);  // Calculate x multiplier
    sy = sin((i - 90) * 0.0174532925);  // Calculate y multiplier
    x0 = sx * 57 + 64;                  // Calculate x start point
    yy0 = sy * 57 + 64;                 // Calculate y start point
    x1 = sx * 50 + 64;                  // Calculate x end point
    yy1 = sy * 50 + 64;                 // Calculate y end point

    tft.drawLine(x0, yy0, x1, yy1, TFT_BLUE);  // Draw the line
  }

  // Draw 60 dots for the clock marks
  for (int i = 0; i < 360; i += 6) {
    sx = cos((i - 90) * 0.0174532925);  // Calculate x multiplier
    sy = sin((i - 90) * 0.0174532925);  // Calculate y multiplier
    x0 = sx * 53 + 64;                  // Calculate x point
    yy0 = sy * 53 + 64;                 // Calculate y point

    tft.drawPixel(x0, yy0, TFT_BLUE);  // Draw the dot
    // Draw larger dots at 12, 3, 6, 9 o'clock positions
    if (i == 0 || i == 180)
      tft.fillCircle(x0, yy0, 1, TFT_CYAN);
    if (i == 0 || i == 180)
      tft.fillCircle(x0 + 1, yy0, 1, TFT_CYAN);
    if (i == 90 || i == 270)
      tft.fillCircle(x0, yy0, 1, TFT_CYAN);
    if (i == 90 || i == 270)
      tft.fillCircle(x0 + 1, yy0, 1, TFT_CYAN);
  }

  tft.fillCircle(65, 65, 3, TFT_RED);  // Draw the center dot

  // Draw text at the center of the display
  tft.setTextColor(TFT_GREEN);  // Set text color

  // Position text based on the display rotation
#if TFT_DIRECTION == 0 || TFT_DIRECTION == 2
  tft.drawCentreString("Freenove", 64, 130, 4);
#elif TFT_DIRECTION == 1 || TFT_DIRECTION == 3
  tft.drawCentreString("Freenove", 184, 54, 4);
#endif
```

```
87
88    targetTime = millis() + 1000;  // Set the target time for the next second
89  }
90
91  void loop() {
92    if (targetTime < millis()) {
93      targetTime = millis() + 1000;  // Update the target time for the next second
94      ss++;                          // Increment seconds
95      if (ss == 60) {
96        ss = 0;  // Reset seconds
97        mm++;    // Increment minutes
98        if (mm > 59) {
99          mm = 0;  // Reset minutes
100         hh++;    // Increment hours
101         if (hh > 23) {
102           hh = 0;  // Reset hours
103         }
104       }
105     }
106
107     // Pre-compute hand degrees and positions for a fast screen update
108     sdeg = ss * 6;                          // Calculate second hand angle (0-59 -> 0-354)
109     mdeg = mm * 6 + sdeg * 0.01666667;      // Calculate minute hand angle (0-59 -> 0-360)
110 including seconds
        hdeg = hh * 30 + mdeg * 0.0833333;      // Calculate hour hand angle (0-11 -> 0-360)
111 including minutes and seconds
        hx = cos((hdeg - 90) * 0.0174532925);  // Calculate x multiplier for hour hand
112     hy = sin((hdeg - 90) * 0.0174532925);  // Calculate y multiplier for hour hand
113     mx = cos((mdeg - 90) * 0.0174532925);  // Calculate x multiplier for minute hand
114     my = sin((mdeg - 90) * 0.0174532925);  // Calculate y multiplier for minute hand
115     sx = cos((sdeg - 90) * 0.0174532925);  // Calculate x multiplier for second hand
116     sy = sin((sdeg - 90) * 0.0174532925);  // Calculate y multiplier for second hand
117
118     if (ss == 0 || initial) {
119       initial = 0;  // Reset initial flag
120       // Erase hour and minute hand positions every minute
121       tft.drawLine(ohx, ohy, 65, 65, TFT_BLACK);  // Erase old hour hand position
122       ohx = hx * 33 + 65;                         // Calculate new hour hand x position
123       ohy = hy * 33 + 65;                         // Calculate new hour hand y position
124       tft.drawLine(omx, omy, 65, 65, TFT_BLACK);  // Erase old minute hand position
125       omx = mx * 44 + 65;                         // Calculate new minute hand x position
126       omy = my * 44 + 65;                         // Calculate new minute hand y position
127     }
128
```

```
129       // Redraw new hand positions, hour and minute hands not erased here to avoid flicker
130       tft.drawLine(osx, osy, 65, 65, TFT_BLACK);  // Erase old second hand position
131       tft.drawLine(ohx, ohy, 65, 65, TFT_WHITE);  // Draw new hour hand position
132       tft.drawLine(omx, omy, 65, 65, TFT_WHITE);  // Draw new minute hand position
133       osx = sx * 47 + 65;                         // Calculate new second hand x position
134       osy = sy * 47 + 65;                         // Calculate new second hand y position
135       tft.drawLine(osx, osy, 65, 65, TFT_RED);    // Draw new second hand position
136
137       tft.fillCircle(65, 65, 3, TFT_RED);  // Draw the center dot
138     }
139   }
```

After uploading the code, the TFT screen will display a real-time clock along with the text "Freenove", as shown in the image below:



For a more in-depth understanding of how the SPI protocol drives the TFT screen, please refer to TFT_eSPI.

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

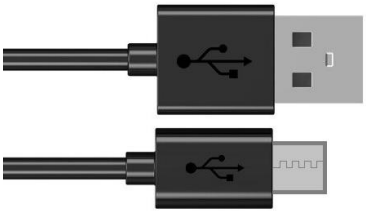
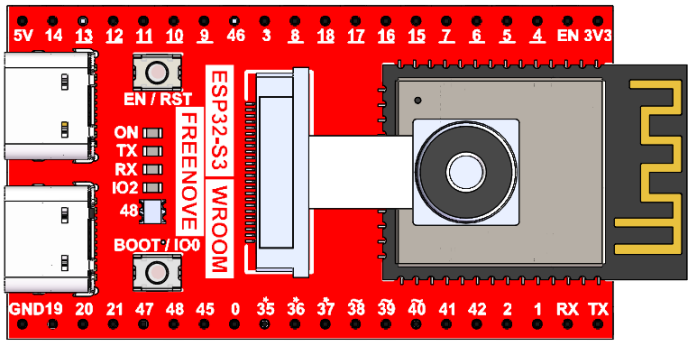
Chapter 9 Camera TFT Test

In the previous two chapters, we covered the basic usage of both the camera module and TFT display separately. This chapter will focus on integrating these two components for combined applications.

Project 9.1 Camera TFT Show

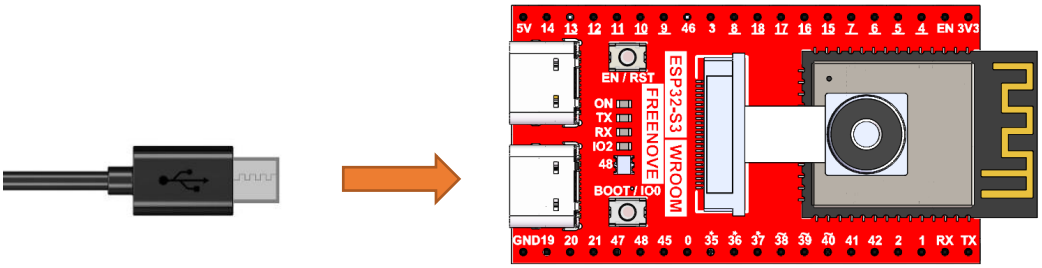
Capture image data using the camera module and display it on the TFT screen.

Component List

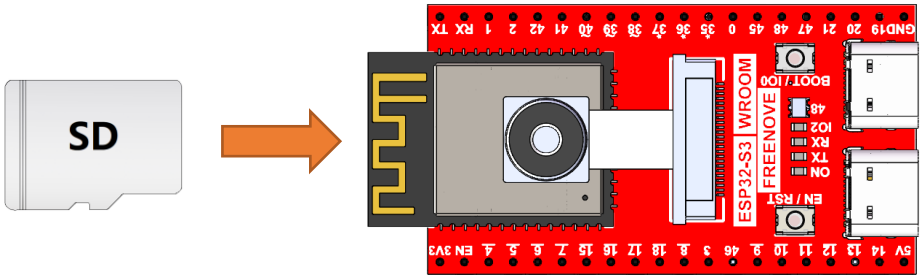
SD card x1 	USB cable x1 
Freenove Media Kit for ESP32-S3 x1 	

Circuit

Connect Freenove Media Kit for ESP32-S3 to your computer using the USB cable.



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



Sketch

Sketch_08_1_Camera_TFT_Show

The following is the program code:

```
#include <TFT_eSPI.h>
#include "esp_camera.h"
#define CAMERA_MODEL_ESP32S3_EYE  // Has PSRAM
#include "camera_pins.h"

#define TFT_DIRECTION 0  // Define the direction of the TFT display (0, 1, 2, or 3)

TFT_eSPI tft = TFT_eSPI();

void setup() {
  Serial.begin(115200);
  Serial.setDebugOutput(true);
  Serial.println();

  tft.init();                       // Initialize the TFT display
  tft.setRotation(TFT_DIRECTION);   // Set the rotation of the TFT display
  camera_init();                    // Initialize the camera
}

void loop() {
  // Capture a frame from the camera
  camera_fb_t *fb = esp_camera_fb_get();
  if (!fb) {
    Serial.println("Camera capture failed");
    return;
  }

  // Define screen and camera dimensions
  int screenWidth = 135;
  int screenHeight = 240;
```

```
  int camWidth = fb->width;
  int camHeight = fb->height;

  // Calculate cropping area
  int cropWidth = screenWidth;
  int cropHeight = screenHeight;
  int cropStartX = (camWidth - cropWidth) / 2;
  int cropStartY = (camHeight - cropHeight) / 2;

  // Check if cropping is needed
  if (camWidth > screenWidth || camHeight > screenHeight) {
    // Allocate memory for cropped image
    uint16_t *croppedBuffer = (uint16_t *)malloc(cropWidth * cropHeight * sizeof(uint16_t));
    if (!croppedBuffer) {
      Serial.println("Failed to allocate memory for cropped image");
      esp_camera_fb_return(fb);
      return;
    }

    // Crop the image
    for (int y = 0; y < cropHeight; y++) {
      for (int x = 0; x < cropWidth; x++) {
        croppedBuffer[y * cropWidth + x] = ((uint16_t *)fb->buf)[(cropStartY + y) * camWidth +
(cropStartX + x)];
      }
    }

    // Display cropped image on the TFT screen
    tft.startWrite();
    tft.pushImage(0, 0, cropWidth, cropHeight, croppedBuffer);
    tft.endWrite();

    // Free the cropped image buffer
    free(croppedBuffer);
  } else {
    // If camera size is less than or equal to screen size, display the image directly
    tft.startWrite();
    tft.pushImage(0, 0, camWidth, camHeight, fb->buf);
    tft.endWrite();
  }

  // Return the frame buffer to the driver for reuse
  esp_camera_fb_return(fb);
}
```

```
75
76  void camera_init(void) {
77    camera_config_t config;
78    config.ledc_channel = LEDC_CHANNEL_0;
79    config.ledc_timer = LEDC_TIMER_0;
80    config.pin_d0 = Y2_GPIO_NUM;
81    config.pin_d1 = Y3_GPIO_NUM;
82    config.pin_d2 = Y4_GPIO_NUM;
83    config.pin_d3 = Y5_GPIO_NUM;
84    config.pin_d4 = Y6_GPIO_NUM;
85    config.pin_d5 = Y7_GPIO_NUM;
86    config.pin_d6 = Y8_GPIO_NUM;
87    config.pin_d7 = Y9_GPIO_NUM;
88    config.pin_xclk = XCLK_GPIO_NUM;
89    config.pin_pclk = PCLK_GPIO_NUM;
90    config.pin_vsync = VSYNC_GPIO_NUM;
91    config.pin_href = HREF_GPIO_NUM;
92    config.pin_sccb_sda = SIOD_GPIO_NUM;
93    config.pin_sccb_scl = SIOC_GPIO_NUM;
94    config.pin_pwdn = PWDN_GPIO_NUM;
95    config.pin_reset = RESET_GPIO_NUM;
96    config.xclk_freq_hz = 10000000;
97    config.frame_size = FRAMESIZE_240X240;
98    config.pixel_format = PIXFORMAT_RGB565;
99    config.grab_mode = CAMERA_GRAB_WHEN_EMPTY;
100   config.fb_location = CAMERA_FB_IN_PSRAM;
101   config.jpeg_quality = 12;
102   config.fb_count = 1;
103
104   // Initialize the camera
105   esp_err_t err = esp_camera_init(&config);
106   if (err != ESP_OK) {
107     Serial.printf("Camera initialization failed, error code 0x%x", err);
108     return;
109   }
110
111   sensor_t *s = esp_camera_sensor_get();
112   // The initial sensor may be vertically flipped and have high color saturation
113   s->set_hmirror(s, 1);      // Mirror the image horizontally
114   s->set_vflip(s, 0);        // Restore vertical orientation
115   s->set_brightness(s, 1);   // Slightly increase brightness
116   s->set_saturation(s, 0);   // Reduce saturation
117 }
```

Include header files and definition.

```
#include <TFT_eSPI.h>
#include "esp_camera.h"
#define CAMERA_MODEL_ESP32S3_EYE  // Has PSRAM
#include "camera_pins.h"
```

Define screen rotation orientation (0-3 correspond to 0°, 90°, 180°, and 270° respectively)

```
#define TFT_DIRECTION 0  // Define the direction of the TFT display (0, 1, 2, or 3)
```

Initialize TFT screen and camera configuration.

```
TFT_eSPI tft = TFT_eSPI();
...
tft.init();                       // Initialize the TFT display
tft.setRotation(TFT_DIRECTION);   // Set the rotation of the TFT display
camera_init();                    // Initialize the camera
```

Capture camera data and store it in the pointer fb.

```
// Capture a frame from the camera
camera_fb_t *fb = esp_camera_fb_get();
if (!fb) {
  Serial.println("Camera capture failed");
  return;
}
```

Crop the image according to the camera's resolution.

```
// Define screen and camera dimensions
int screenWidth = 135;
int screenHeight = 240;
int camWidth = fb->width;
int camHeight = fb->height;

// Calculate cropping area
int cropWidth = screenWidth;
int cropHeight = screenHeight;
int cropStartX = (camWidth - cropWidth) / 2;
int cropStartY = (camHeight - cropHeight) / 2;
// Check if cropping is needed
if (camWidth > screenWidth || camHeight > screenHeight) {
  // Allocate memory for cropped image
  uint16_t *croppedBuffer = (uint16_t *)malloc(cropWidth * cropHeight * sizeof(uint16_t));
  if (!croppedBuffer) {
   Serial.println("Failed to allocate memory for cropped image");
    esp_camera_fb_return(fb);
    return;
}

// Crop the image
for (int y = 0; y < cropHeight; y++) {
```

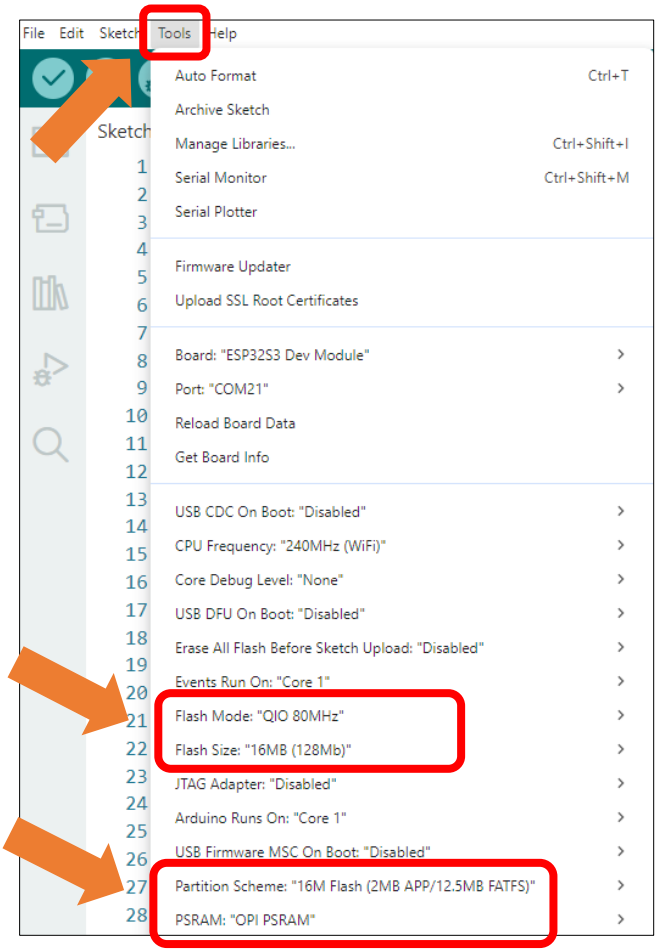
```
    for (int x = 0; x < cropWidth; x++) {
      croppedBuffer[y * cropWidth + x] = ((uint16_t *)fb->buf)[(cropStartY + y) * camWidth +
(cropStartX + x)];
    }
}
```

Initialize camera configuration and adjust image parameters.

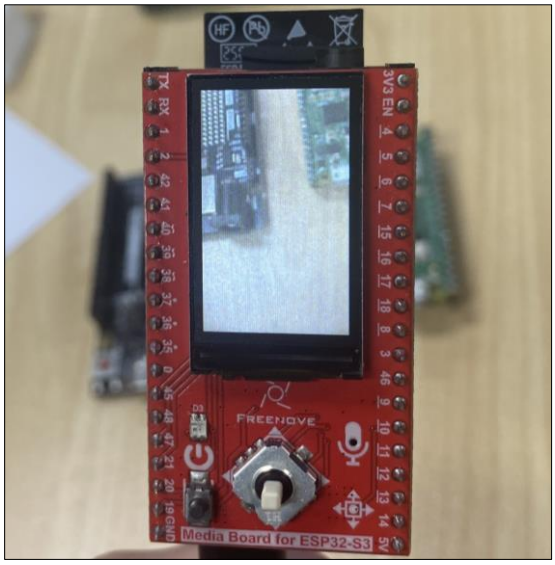
```
// Initialize the camera
esp_err_t err = esp_camera_init(&config);
if (err != ESP_OK) {
  Serial.printf("Camera initialization failed, error code 0x%x", err);
  return;
}

sensor_t *s = esp_camera_sensor_get();
// The initial sensor may be vertically flipped and have high color saturation
s->set_hmirror(s, 1);      // Mirror the image horizontally
s->set_vflip(s, 0);        // Restore vertical orientation
s->set_brightness(s, 1);   // Slightly increase brightness
s->set_saturation(s, 0);   // Reduce saturation
```

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

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



After uploading the code, the TFT screen will display the camera's captured images in real-time, as shown in the following illustration:



Note: Display performance may differ across camera models, with some devices showing a flipped image. If this occurs, configure the horizontal/vertical mirroring settings.

```
104  s->set_hmirror(s, 1);      // Mirror the image horizontally
105  s->set_vflip(s, 0);        // Restore vertical orientation
```

Parameter Description:

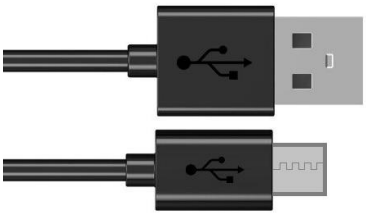
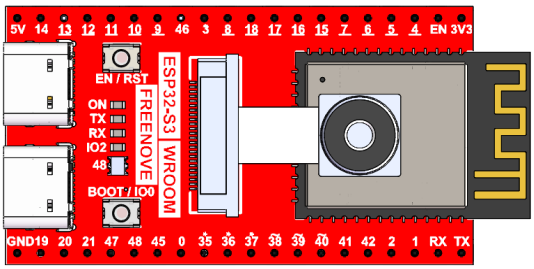
- 0: Normal display
- 1: Flip (mirror)

To achieve the desired display, configure the settings according to real-time preview feedback during setup.

Project 9.2

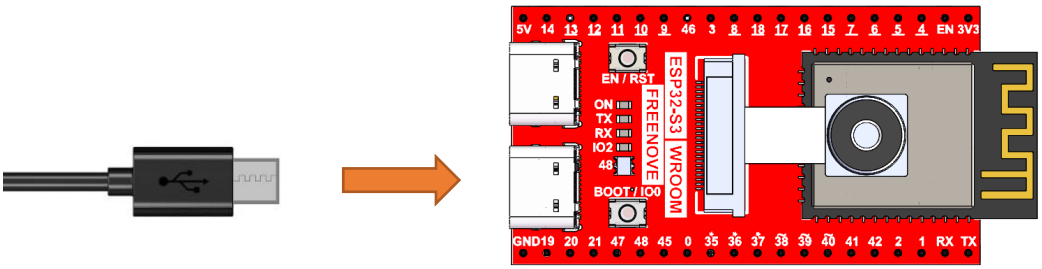
In the previous section, we learned how to capture camera images and display them on a TFT screen. This section will explain how to save the images captured by the camera to an SD card.

Component List

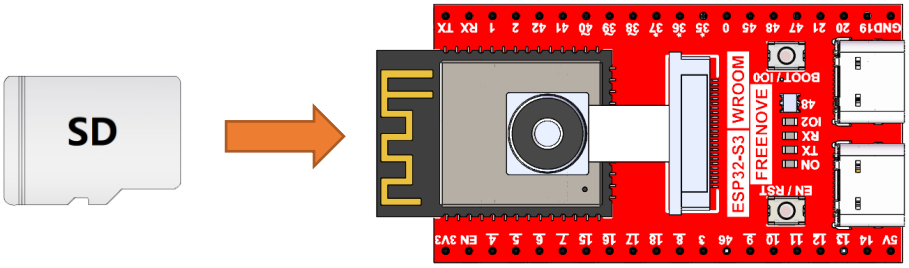
SD card x1 	USB cable x1 
Freenove Media Kit for ESP32-S3 x1 	Card reader x1 (random color) 

Circuit

Connect Freenove Media Kit for ESP32-S3 to your computer using the USB cable.



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



Sketch

Sketch_08_2_Take_A_Photo

The following is the program code:

```
#include <TFT_eSPI.h>
#include "esp_camera.h"
#include "driver_sdmmc.h"
#define CAMERA_MODEL_ESP32S3_EYE  // Has PSRAM
#include "camera_pins.h"

#define BUTTON_PIN 19  // Please do not modify it.
#define SD_MMC_CMD 38  // Please do not modify it.
#define SD_MMC_CLK 39  // Please do not modify it.
#define SD_MMC_D0 40   // Please do not modify it.

TFT_eSPI tft = TFT_eSPI();

void camera_init(int state);
void cameraShow(void);
void cameraPhoto(void);

void setup() {
  Serial.begin(115200);
  Serial.setDebugOutput(true);
  Serial.println();
  analogReadResolution(12);
  analogSetAttenuation(ADC_11db);
  pinMode(BUTTON_PIN, INPUT_PULLUP);
  tft.init();
  tft.setRotation(0);
  camera_init(0);
  sdmmc_init(SD_MMC_CLK, SD_MMC_CMD, SD_MMC_D0);
  remove_dir("/video");
  create_dir("/video");
}

void loop() {
  cameraShow();   // Continuously display the camera feed on the TFT screen
  cameraPhoto();  // Check for button press to take a photo
}

void camera_init(int state) {
  camera_config_t config;
```

```
  config.ledc_channel = LEDC_CHANNEL_0;
  config.ledc_timer = LEDC_TIMER_0;
  config.pin_d0 = Y2_GPIO_NUM;
  config.pin_d1 = Y3_GPIO_NUM;
  config.pin_d2 = Y4_GPIO_NUM;
  config.pin_d3 = Y5_GPIO_NUM;
  config.pin_d4 = Y6_GPIO_NUM;
  config.pin_d5 = Y7_GPIO_NUM;
  config.pin_d6 = Y8_GPIO_NUM;
  config.pin_d7 = Y9_GPIO_NUM;
  config.pin_xclk = XCLK_GPIO_NUM;
  config.pin_pclk = PCLK_GPIO_NUM;
  config.pin_vsync = VSYNC_GPIO_NUM;
  config.pin_href = HREF_GPIO_NUM;
  config.pin_sccb_sda = SIOD_GPIO_NUM;
  config.pin_sccb_scl = SIOC_GPIO_NUM;
  config.pin_pwdn = PWDN_GPIO_NUM;
  config.pin_reset = RESET_GPIO_NUM;
  config.xclk_freq_hz = 10000000;
  config.grab_mode = CAMERA_GRAB_LATEST;
  config.fb_location = CAMERA_FB_IN_PSRAM;
  config.jpeg_quality = 12;
  config.fb_count = 1;
  if (state == 0) {
    config.frame_size = FRAMESIZE_240X240;
    config.pixel_format = PIXFORMAT_RGB565;
  } else {
    config.frame_size = FRAMESIZE_VGA;
    config.pixel_format = PIXFORMAT_JPEG;
  }
  // Deinitialize and reinitialize the camera with the new configuration
  esp_camera_deinit();
  esp_camera_return_all();
  esp_err_t err = esp_camera_init(&config);
  if (err != ESP_OK) {
    Serial.printf("Camera initialization failed, error code 0x%x", err);
    return;
  }
  sensor_t *s = esp_camera_sensor_get();
  // The initial sensor may be vertically flipped and have high color saturation
  s->set_hmirror(s, 0);      // Mirror the image horizontally
  s->set_vflip(s, 0);        // Restore vertical orientation
  s->set_brightness(s, 1);   // Slightly increase brightness
  s->set_saturation(s, 0);   // Reduce saturation
```

```
}

void cameraShow(void) {
  // Capture a frame from the camera
  camera_fb_t *fb = esp_camera_fb_get();
  if (!fb) {
    Serial.println("Camera capture failed");
    return;
  }

  // Define screen and camera dimensions
  int screenWidth = 135;
  int screenHeight = 240;
  int camWidth = fb->width;
  int camHeight = fb->height;

  // Calculate cropping area
  int cropWidth = screenWidth;
  int cropHeight = screenHeight;
  int cropStartX = (camWidth - cropWidth) / 2;
  int cropStartY = (camHeight - cropHeight) / 2;

  // Check if cropping is needed
  if (camWidth > screenWidth || camHeight > screenHeight) {
    // Allocate memory for cropped image
    uint16_t *croppedBuffer = (uint16_t *)malloc(cropWidth * cropHeight * sizeof(uint16_t));
    if (!croppedBuffer) {
      Serial.println("Failed to allocate memory for cropped image");
      esp_camera_fb_return(fb);
      return;
    }

    // Crop the image
    for (int y = 0; y < cropHeight; y++) {
      for (int x = 0; x < cropWidth; x++) {
        croppedBuffer[y * cropWidth + x] = ((uint16_t *)fb->buf)[(cropStartY + y) * camWidth +
(cropStartX + x)];
      }
    }

    // Display cropped image on the TFT screen
    tft.startWrite();
    tft.pushImage(0, 0, cropWidth, cropHeight, croppedBuffer);
    tft.endWrite();
```

```
    // Free the cropped image buffer
    free(croppedBuffer);
  } else {
    // If camera size is less than or equal to screen size, display the image directly
    tft.startWrite();
    tft.pushImage(0, 0, camWidth, camHeight, fb->buf);
    tft.endWrite();
  }

  // Return the frame buffer to the driver for reuse
  esp_camera_fb_return(fb);
}

void cameraPhoto(void) {
  static int fileCounter = 0;
  int analogValue = analogRead(BUTTON_PIN);
  if (analogValue < 100) {
    camera_init(1);  // Reinitialize camera for photo capture
    camera_fb_t *fb = esp_camera_fb_get();
    if (!fb) {
      Serial.println("Camera capture failed");
      return;
    }
    char filename[32];
    snprintf(filename, sizeof(filename), "/video/photo_%04d.jpg", fileCounter);
    write_jpg(filename, fb->buf, fb->len);  // Save the photo to the SD card
    fileCounter++;
    camera_init(0);         // Reinitialize camera for live view
    list_dir("/video", 0);  // List the contents of the /video directory
    while (analogRead(BUTTON_PIN) < 3000)
      ;  // Wait for button release
  }
}
```

Include the needed hearder files and definition.

```
#include <TFT_eSPI.h>
#include "esp_camera.h"
#include "driver_sdmmc.h"
#define CAMERA_MODEL_ESP32S3_EYE  // Has PSRAM
#include "camera_pins.h"
```

Define screen rotation orientation (0-3 correspond to 0°, 90°, 180°, and 270° respectively)

```
#define BUTTON_PIN 19  // Please do not modify it.
#define SD_MMC_CMD 38  // Please do not modify it.
#define SD_MMC_CLK 39  // Please do not modify it.
```

```
10  #define SD_MMC_D0 40   // Please do not modify it.
```

Initialize TFT screen and camera configuration.

```
8   TFT_eSPI tft = TFT_eSPI();
    ...
15  tft.init();                      // Initialize the TFT display
16  tft.setRotation(TFT_DIRECTION);  // Set the rotation of the TFT display
17  camera_init();                   // Initialize the camera
```

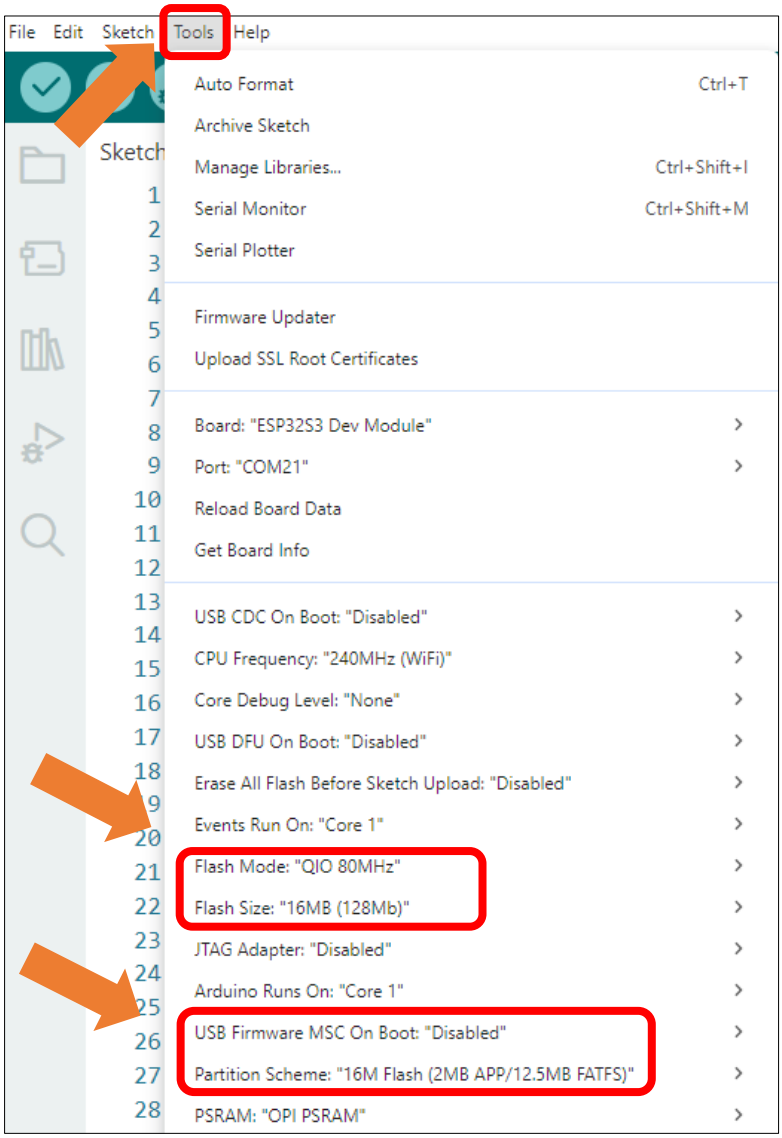
Obtain the frame buffer from the camera

```
21  // Capture a frame from the camera
22  camera_fb_t *fb = esp_camera_fb_get();
23  if (!fb) {
24    Serial.println("Camera capture failed");
25    return;
26  }
```

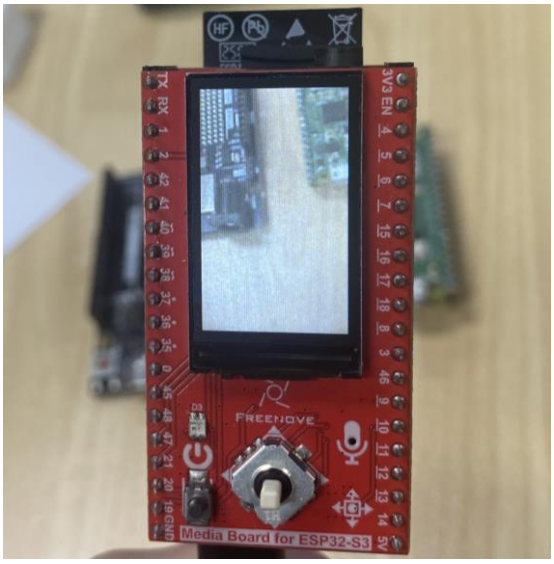
Crop the image captured from the camera

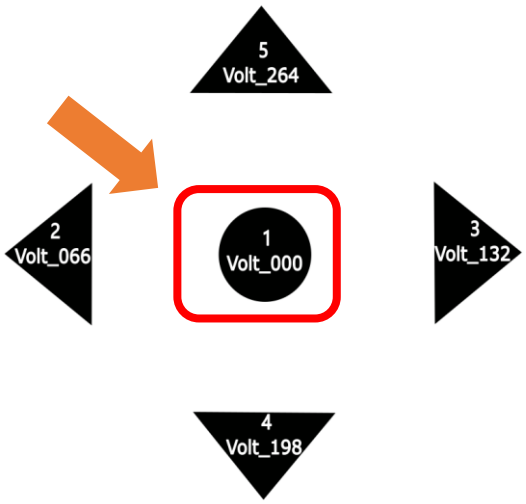
```
94  // Define screen and camera dimensions
95  int screenWidth = 135;
96  int screenHeight = 240;
97  int camWidth = fb->width;
98  int camHeight = fb->height;
99  // Calculate cropping area
100 int cropWidth = screenWidth;
101 int cropHeight = screenHeight;
102 int cropStartX = (camWidth - cropWidth) / 2;
103 int cropStartY = (camHeight - cropHeight) / 2;
104 // Check if cropping is needed
105 if (camWidth > screenWidth || camHeight > screenHeight) {
106   // Allocate memory for cropped image
107   uint16_t *croppedBuffer = (uint16_t *)malloc(cropWidth * cropHeight * sizeof(uint16_t));
108   if (!croppedBuffer) {
109     Serial.println("Failed to allocate memory for cropped image");
110     esp_camera_fb_return(fb);
111     return;
112   }
113   // Crop the image
114   for (int y = 0; y < cropHeight; y++) {
115     for (int x = 0; x < cropWidth; x++) {
116       croppedBuffer[y * cropWidth + x] = ((uint16_t *)fb->buf)[(cropStartY + y) * camWidth +
117 (cropStartX + x)];
118     }
119   }
```

It is necessary to change the settings in Arduino IDE before clicking the Uploading button, as shown below.
Caution: Incorrect settings will result in compilation error or uploading failure. To achieve desired result, please configure exactly the same as below.



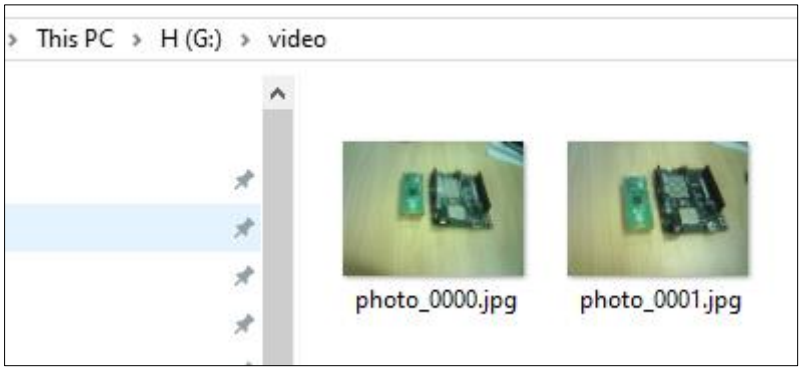
After uploading the code, the image from the camera will be displayed on the TFT screen. Pressing the button down (center) will automatically save the photo to the SD card





After taking photos, please remove the SD card and insert it into a card reader, then connect the reader to your computer.

In the SD card's directory, there is a folder named 'Video' which contains the pictures you just captured.



Note: Display performance may differ across camera models, with some devices showing a flipped image. If this occurs, configure the horizontal/vertical mirroring settings.

```
104  s->set_hmirror(s, 1);      // Mirror the image horizontally
105  s->set_vflip(s, 0);        // Restore vertical orientation
```

Parameter Description:

- 0: Normal display
- 1: Flip (mirror)

To achieve the desired display, configure the settings according to real-time preview feedback during setup.

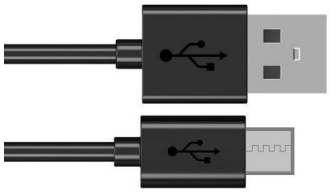
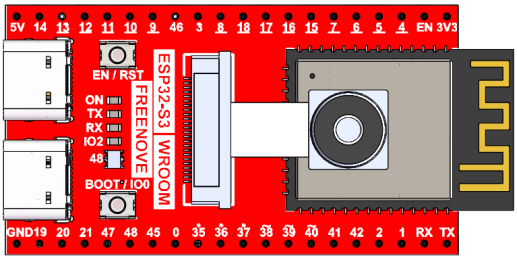
If you have any concerns, please feel free to contact us via support@freenove.com

Chapter 10 Record Test

In this chapter, we will explore the audio recording functionality of the Freenove Media Kit for ESP32.

Project 10.1 Record To WAV

Component List

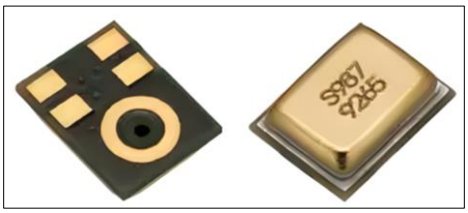
SD card x1 	USB cable x1 
Freenove Media Kit for ESP32-S3 x1 	Card reader x1 (random color) 

Component knowledge

MEMS-MIC

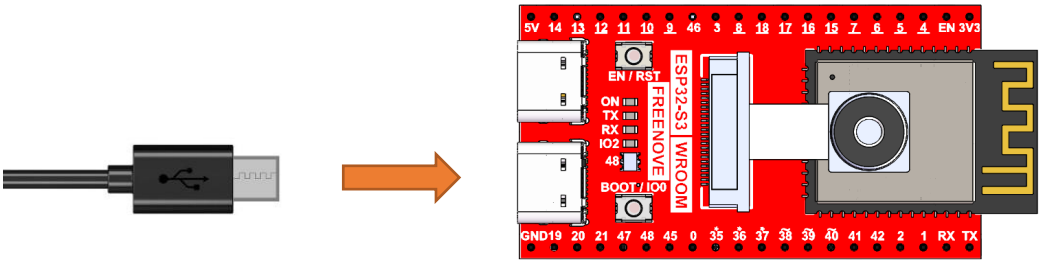
A MEMS Microphone (Micro-Electro-Mechanical Systems Microphone) is a miniature microphone manufactured using MEMS technology. It integrates mechanical sensing elements and electronic circuits on the same chip to achieve sound signal acquisition and conversion. Its working principle primarily involves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

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

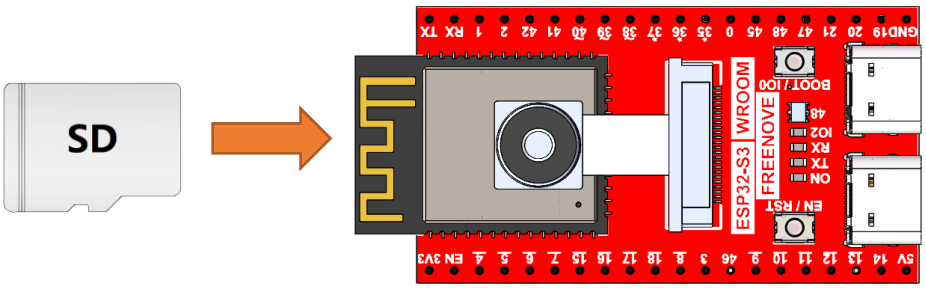


Circuit

Connect Freenove Media Kit for ESP32-S3 to your computer using the USB cable.



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



Sketch

Sketch_09_1_Record_To_WAV

The following is the program code:

```
#include "driver_audio_input.h"
#include "driver_sdmmc.h"

#define RECORDER_FOLDER "/recorder"

#define SD_MMC_CMD      38  // Please do not modify it.
#define SD_MMC_CLK      39  // Please do not modify it.
#define SD_MMC_DO       40  // Please do not modify it.
#define AUDIO_INPUT_SCK  3  // Please do not modify it.
#define AUDIO_INPUT_WS  14  // Please do not modify it.
#define AUDIO_INPUT_DIN 46  // Please do not modify it.

void setup() {
    // Initialize the serial port
    Serial.begin(115200);
    while (!Serial) {
        delay(10);
    }
```

```
19
20      // Initialize I2S bus for audio input
21      audio_input_init(AUDIO_INPUT_SCK, AUDIO_INPUT_WS, AUDIO_INPUT_DIN);
22
23      // Initialize SD card for storing audio files
24      sdmmc_init(SD_MMC_CLK, SD_MMC_CMD, SD_MMC_DO);
25      remove_dir(RECORDER_FOLDER);  // Remove the existing recorder folder if it exists
26      create_dir(RECORDER_FOLDER);   // Create a new recorder folder
27
28      Serial.println("Recording 5 seconds of audio data...");
29
30      // Record 5 seconds of audio data and store it in a buffer
31      size_t wav_size;
32      uint8_t* wav_buffer = audio_input_record_wav(5, wav_size);
33
34      // Print the first 1024 bytes of the recorded audio buffer for debugging
35      audio_input_print_buffer(wav_buffer, 1024);
36
37      // Construct the file name for the recorded audio file
38      String file_name = String(RECORDER_FOLDER) + "/recorder_0.wav";
39      Serial.println(file_name);
40
41      // Write the recorded audio data to the SD card
42      write_file(file_name.c_str(), wav_buffer, wav_size);
43      Serial.println("Application complete.");
44  }
45
46  void loop() {
47      // Main loop can be used for additional tasks if needed
48  }
```

Add required header files and definitions.

```
1   #include "driver_audio_input.h"
2   #include "driver_sdmmc.h"
```

Define SD card and microphone pins using macros.

```
6   #define SD_MMC_CMD      38  // Please do not modify it.
7   #define SD_MMC_CLK      39  // Please do not modify it.
8   #define SD_MMC_DO       40  // Please do not modify it.
9   #define AUDIO_INPUT_SCK  3  // Please do not modify it.
10  #define AUDIO_INPUT_WS  14  // Please do not modify it.
11  #define AUDIO_INPUT_DIN 46  // Please do not modify it.
```

Initialize microphone and SD card.

```
20  // Initialize I2S bus for audio input
21  audio_input_init(AUDIO_INPUT_SCK, AUDIO_INPUT_WS, AUDIO_INPUT_DIN);
22
```

```
// Initialize SD card for storing audio files
sdmmc_init(SD_MMC_CLK, SD_MMC_CMD, SD_MMC_DO);
remove_dir(RECORDER_FOLDER);  // Remove the existing recorder folder if it exists
create_dir(RECORDER_FOLDER);   // Create a new recorder folder
```

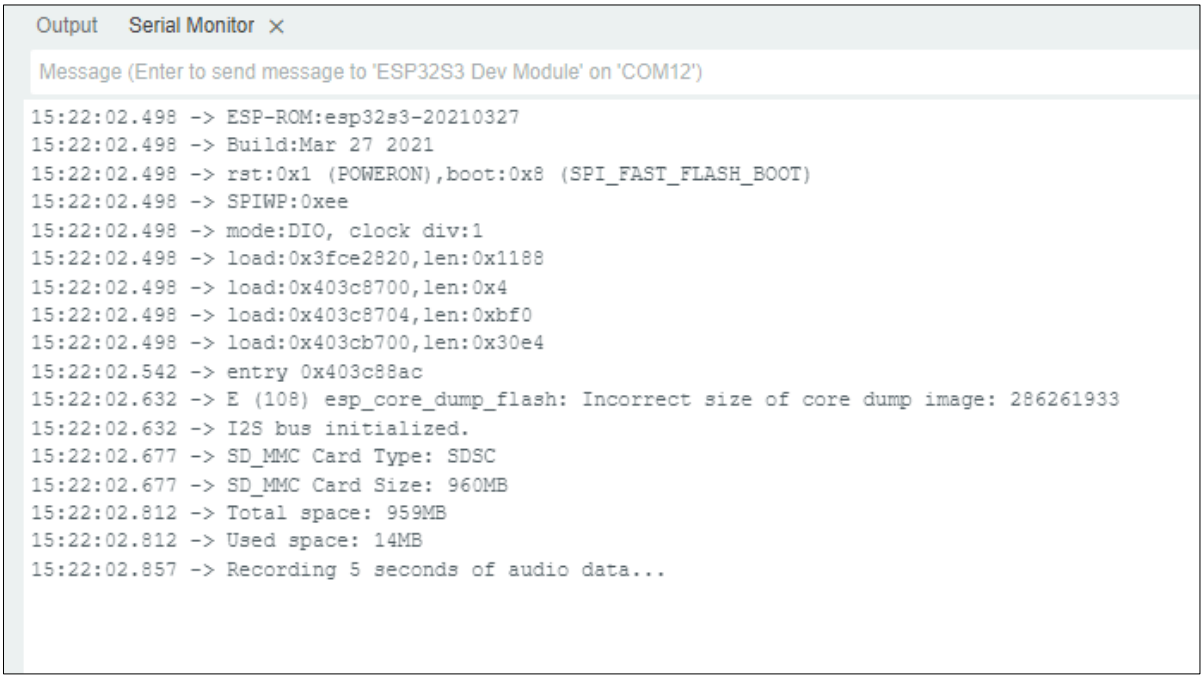
Acquire frame buffer from camera.

```
// Capture a frame from the camera
camera_fb_t *fb = esp_camera_fb_get();
if (!fb) {
  Serial.println("Camera capture failed");
  return;
}
```

Record 5 seconds of audio and save.

```
// Record 5 seconds of audio data and store it in a buffer
size_t wav_size;
uint8_t* wav_buffer = audio_input_record_wav(5, wav_size);

// Print the first 1024 bytes of the recorded audio buffer for debugging
audio_input_print_buffer(wav_buffer, 1024);

// Construct the file name for the recorded audio file
String file_name = String(RECORDER_FOLDER) + "/recorder_0.wav";
Serial.println(file_name);

// Write the recorded audio data to the SD card
write_file(file_name.c_str(), wav_buffer, wav_size);
Serial.println("Application complete.");
```

After uploading the code, a 5-second audio clip will be recorded and saved in WAV format on the SD card.



After recording is complete, remove the SD card and insert it into a card reader. Connect the card reader to your computer.

In the SD card directory, you will find a folder named "recorder", which contains the recorded audio file.

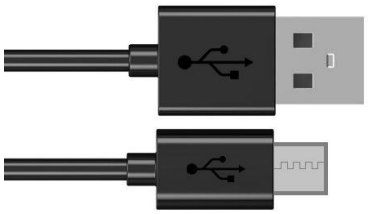
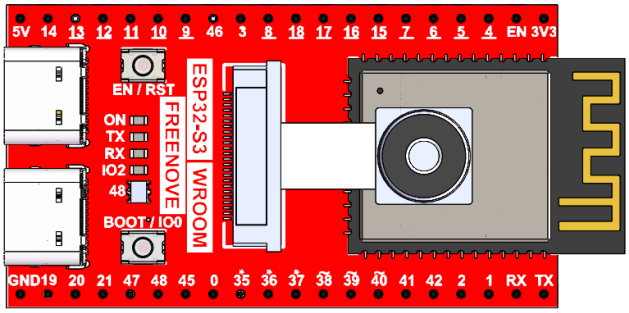
Reference

`audio_input_init(uint8_t sck, uint8_t ws, uint8_t din)`
This functions is used to initialize I2S pins.
`uint8_t* audio_input_record_wav(uint32_t duration, size_t& wav_size)`
This function is used to record audio data and returns a pointer to the audio data buffer. **Parameters:** duration: Recording duration (in seconds) wav_size: Total size of the audio data in bytes
`void write_file(const char *path, const uint8_t *buffer, size_t size)`
This function writes data from memory to the SD card. **Parameters:** path: Target file path buffer: Data buffer to be written size: Number of bytes to write
`uint8_t* audio_input_record_wav(uint32_t duration, size_t& wav_size)`
This function captures audio samples and returns a pointer to the audio buffer. **Parameters:** duration: Recording duration (seconds) wav_size: Total bytes of generated WAV data

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

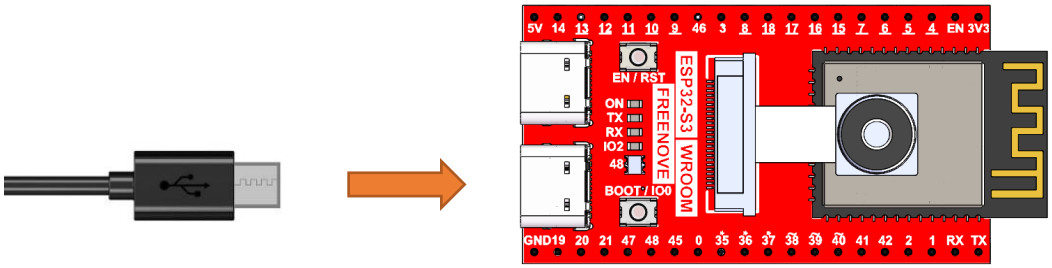
Project 10.2 Record and Play

Component List

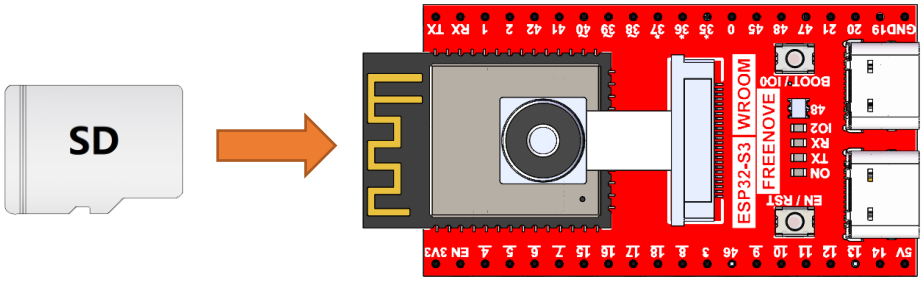
SD card x1 	USB cable x1 
Freenove Media Kit for ESP32-S3 x1 	Card reader x1 (random color) 

Circuit

Connect Freenove Media Kit for ESP32-S3 to your computer using the USB cable.



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



Sketch

Sketch_09_2_Record_And_Play

The following is the program code:

```
#include "driver_audio_input.h"
#include "driver_audio_output.h"
#include "driver_sdmmc.h"

#define RECORDER_FOLDER "/recorder"
#define BUTTON_PIN 20          // Please do not modify it.
#define SD_MMC_CMD 38          // Please do not modify it.
#define SD_MMC_CLK 39          // Please do not modify it.
#define SD_MMC_DO 40           // Please do not modify it.
#define AUDIO_INPUT_SCK 3      // Please do not modify it.
#define AUDIO_INPUT_WS 14      // Please do not modify it.
#define AUDIO_INPUT_DIN 46     // Please do not modify it.
#define AUDIO_OUTPUT_BCLK 42   // Please do not modify it.
#define AUDIO_OUTPUT_LRC 41    // Please do not modify it.
#define AUDIO_OUTPUT_DOUT 1    // Please do not modify it.

int recorder_task_flag = 0;  // Task status flag (0=stopped, 1=running)

void setup() {
  // Initialize the serial port
  Serial.begin(115200);
  while (!Serial) {
    delay(10);
  }

  pinMode(BUTTON_PIN, INPUT_PULLUP);

  // Initialize I2S bus for audio input and output
  audio_input_init(AUDIO_INPUT_SCK, AUDIO_INPUT_WS, AUDIO_INPUT_DIN);
  audio_output_init(AUDIO_OUTPUT_BCLK, AUDIO_OUTPUT_LRC, AUDIO_OUTPUT_DOUT);
  audio_output_set_volume(21);  // Set volume to maximum (0...21)

  // Initialize SD card for storing audio files
  sdmmc_init(SD_MMC_CLK, SD_MMC_CMD, SD_MMC_DO);
  create_dir(RECORDER_FOLDER);  // Create a new recorder folder if it doesn't exist
}

void loop() {
  if (analogRead(BUTTON_PIN) <= 100) {
```

```
    start_recorder_task();  // Start recording if button is pressed
  } else {
    if (recorder_task_is_running())
      stop_recorder_task();  // Stop recording if button is released
    audio_output_loop();     // Loop to play audio if available
    if (!audio_output_is_running()) {
      int file_count = read_file_num(RECORDER_FOLDER) - 1;
      if (file_count >= 0) {
        String file_name = String(RECORDER_FOLDER) + "/recorder_" + String(file_count) +
".wav";
        Serial.println(file_name);
        audio_output_load_music(file_name.c_str());  // Load and play the last recorded file
      }
    }
  }
}

/* Start recording task */
void start_recorder_task(void) {
  if (recorder_task_flag == 0) {
    recorder_task_flag = 1;
    xTaskCreate(loopTask_sound_recorder, "loopTask_sound_recorder", 4096, NULL, 1, NULL);  //
Create a new task for recording
  }
}

/* Stop recording task */
void stop_recorder_task(void) {
  if (recorder_task_flag == 1) {
    recorder_task_flag = 0;
    Serial.println("loopTask_sound_recorder deleted!");  // Indicate that the recording task
is stopped
  }
}

/* Check if recording task is active */
int recorder_task_is_running(void) {
  return recorder_task_flag;  // Return the status of the recording task
}

/* Main recording task loop */
void loopTask_sound_recorder(void *pvParameters) {
  Serial.println("loopTask_sound_recorder start...");
  while (recorder_task_flag == 1) {
```

```
    size_t wav_size;
    uint8_t *wav_buffer = audio_input_record_wav(5, wav_size);  // Record 5 seconds of audio
    int file_count = read_file_num(RECORDER_FOLDER);
    String file_name = String(RECORDER_FOLDER) + "/recorder_" + String(file_count) + ".wav";
    Serial.println(file_name);
    write_file(file_name.c_str(), wav_buffer, wav_size);  // Save the recorded audio to the SD
card
    free(wav_buffer);                                     // Free the allocated memory for the
audio buffer
    recorder_task_flag = 0;                               // Stop the recording task
  }
  Serial.println("loopTask_sound_recorder stop...");
  vTaskDelete(NULL);  // Delete the recording task
}


```

Include the required header files.

```
#include "driver_audio_input.h"
#include "driver_audio_output.h"
#include "driver_sdmmc.h"
```

Macro definitions for button, SD card, and microphone pins, and for the audio file storage directory.

```
#define RECORDER_FOLDER "/recorder"
#define BUTTON_PIN 20          // Please do not modify it.
#define SD_MMC_CMD 38          // Please do not modify it.
#define SD_MMC_CLK 39          // Please do not modify it.
#define SD_MMC_D0 40           // Please do not modify it.
#define AUDIO_INPUT_SCK 3      // Please do not modify it.
#define AUDIO_INPUT_WS 14      // Please do not modify it.
#define AUDIO_INPUT_DIN 46     // Please do not modify it.
#define AUDIO_OUTPUT_BCLK 42   // Please do not modify it.
#define AUDIO_OUTPUT_LRC 41    // Please do not modify it.
#define AUDIO_OUTPUT_DOUT 1    // Please do not modify it.
```

Initialize the microphone and SD card.

```
// Initialize I2S bus for audio input and output
audio_input_init(AUDIO_INPUT_SCK, AUDIO_INPUT_WS, AUDIO_INPUT_DIN);
audio_output_init(AUDIO_OUTPUT_BCLK, AUDIO_OUTPUT_LRC, AUDIO_OUTPUT_DOUT);
audio_output_set_volume(21);  // Set volume to maximum (0...21)

// Initialize SD card for storing audio files
sdmmc_init(SD_MMC_CLK, SD_MMC_CMD, SD_MMC_D0);
create_dir(RECORDER_FOLDER);  // Create a new recorder folder if it doesn't exist
```

Record an audio for five seconds.

```
/* Main recording task loop */
```

```
81  void loopTask_sound_recorder(void *pvParameters) {
82    Serial.println("loopTask_sound_recorder start...");
83    while (recorder_task_flag == 1) {
84      size_t wav_size;
85      uint8_t *wav_buffer = audio_input_record_wav(5, wav_size);  // Record 5 seconds of audio
86      int file_count = read_file_num(RECORDER_FOLDER);
87      String file_name = String(RECORDER_FOLDER) + "/recorder_" + String(file_count) + ".wav";
88      Serial.println(file_name);
89      write_file(file_name.c_str(), wav_buffer, wav_size);  // Save the recorded audio to the SD
90  card
91      free(wav_buffer);                                     // Free the allocated memory for the
92  audio buffer
93      recorder_task_flag = 0;                               // Stop the recording task
94    }
95    Serial.println("loopTask_sound_recorder stop...");
96    vTaskDelete(NULL);  // Delete the recording task
97  }
```

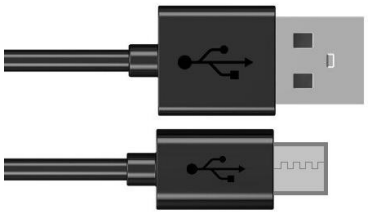
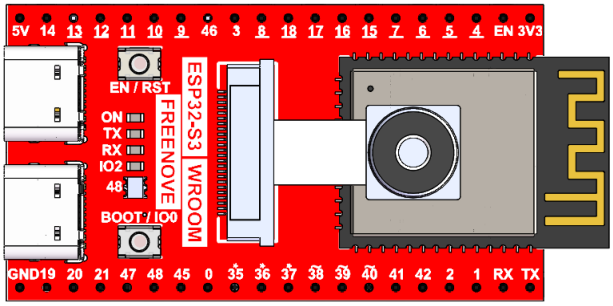
fter uploading the code, press and hold the button to start recording; release the button to automatically stop. Once recording is complete, the system will automatically play back the recorded audio, and the file will be saved in the '**recorder**' folder

Note: For optimal recording quality, please stay as close to the microphone as possible during recording (the microphone location is shown in the figure below).

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

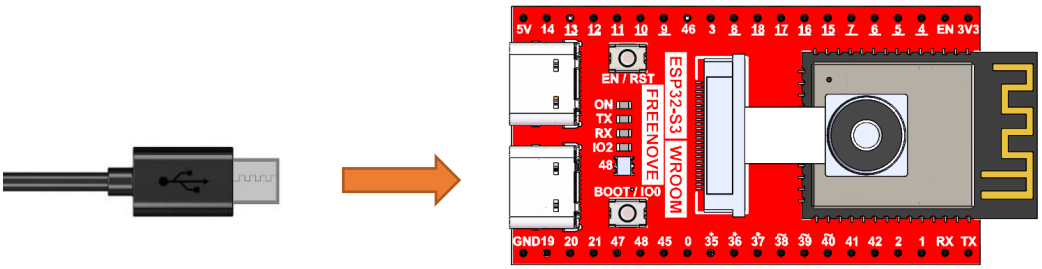
Project 10.3 Record and Play

Component List

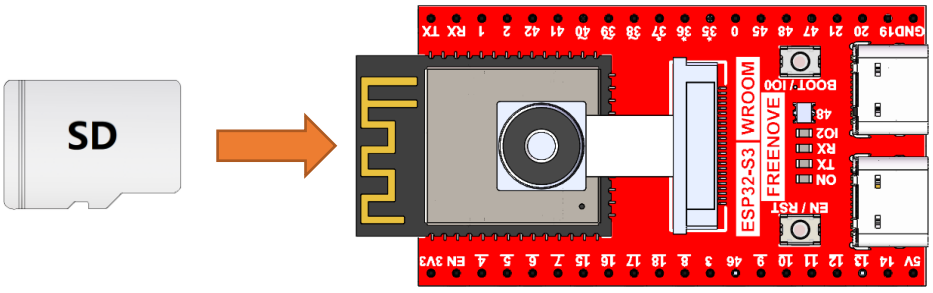
SD card x1 	USB cable x1 
Freenove Media Kit for ESP32-S3 x1 	Card reader x1 (random color) 

Circuit

Connect Freenove Media Kit for ESP32-S3 to your computer using the USB cable.



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



Sketch

Sketch_09_3_Record_And_Play

The following is the program code:

```
#include "driver_audio_input.h"
#include "driver_audio_output.h"
#include "driver_button.h"
#include "driver_sdmmc.h"
#include <esp_heap_caps.h>

// Define the folder path for recording files
#define RECORDER_FOLDER "/recorder"
// Define the pin number for the button (do not modify)
#define BUTTON_PIN 19
// Define the pin numbers for SDMMC interface (do not modify)
#define SD_MMC_CMD 38
#define SD_MMC_CLK 39
#define SD_MMC_D0 40
// Define the pin numbers for audio input (do not modify)
#define AUDIO_INPUT_SCK 3
#define AUDIO_INPUT_WS 14
#define AUDIO_INPUT_DIN 46
// Define the pin numbers for audio output (do not modify)
#define AUDIO_OUTPUT_BCLK 42
#define AUDIO_OUTPUT_LRC 41
#define AUDIO_OUTPUT_DOUT 1

// Define the size of PSRAM in bytes
#define MOLLOC_SIZE (1024 * 1024)

// Create a button object with the specified pin
Button button(BUTTON_PIN);

// Flag to indicate the status of the recorder task (0=stopped, 1=running)
int recorder_task_flag = 0;
// Flag to indicate the status of the player task (0=stopped, 1=running)
int player_task_flag = 0;

// Setup function to initialize the hardware and software components
void setup() {
  // Initialize the serial communication at 115200 baud rate
  Serial.begin(115200);
  // Wait for the serial port to be ready
```

```
  while (!Serial) {
    delay(10);
  }
  // Initialize the button
  button.init();

  // Initialize the I2S bus for audio input
  audio_input_init(AUDIO_INPUT_SCK, AUDIO_INPUT_WS, AUDIO_INPUT_DIN);
  // Initialize the I2S bus for audio output
  audio_output_init(AUDIO_OUTPUT_BCLK, AUDIO_OUTPUT_LRC, AUDIO_OUTPUT_DOUT);
  // Set the volume of the audio output (range 0...21)
  audio_output_set_volume(21);

  // Initialize the SD card
  sdmmc_init(SD_MMC_CLK, SD_MMC_CMD, SD_MMC_D0);
  // Remove the existing recorder folder if it exists
  remove_dir(RECORDER_FOLDER);
  // Create a new recorder folder
  create_dir(RECORDER_FOLDER);
}

// Main loop function that runs continuously
void loop() {
  // Scan the button state
  button.key_scan();
  // Handle button events
  handle_button_events();
  // Delay for 10 milliseconds
  delay(10);
}

// Function to handle button events
void handle_button_events() {
  // Get the current state of the button
  int button_state = button.get_button_state();
  // Get the key value associated with the button press
  int button_key_value = button.get_button_key_value();
  // Switch case based on the button key value
  switch (button_key_value) {
    case 1:
      // If the button is pressed, start the recorder task
      if (button_state == Button::KEY_STATE_PRESSED) {
        start_recorder_task();
      }
```

```
84          // If the button is released, stop the recorder task
85          else if (button_state == Button::KEY_STATE_RELEASED) {
86            stop_recorder_task();
87          }
88          break;
89        case 2:
90          // If the button is pressed, start the player task
91          if (button_state == Button::KEY_STATE_PRESSED) {
92            start_player_task();
93          }
94          break;
95        case 3:
96          // If the button is pressed, stop the player task
97          if (button_state == Button::KEY_STATE_PRESSED) {
98            stop_player_task();
99          }
100       case 4:
101       case 5:
102       default:
103         // Default case for other button key values
104         break;
105     }
106 }
107
108 /* Start recording task */
109 void start_recorder_task(void) {
110   // Check if the recorder task is not already running
111   if (recorder_task_flag == 0) {
112     // Set the recorder task flag to running
113     recorder_task_flag = 1;
114     // Create a new task for recording sound
115     xTaskCreate(loop_task_sound_recorder, "loop_task_sound_recorder", 4096, NULL, 1, NULL);
116   }
117 }
118
119 /* Stop recording task */
120 void stop_recorder_task(void) {
121   // Check if the recorder task is running
122   if (recorder_task_flag == 1) {
123     // Set the recorder task flag to stopped
124     recorder_task_flag = 0;
125     // Print a message indicating the deletion of the recorder task
126     Serial.println("loop_task_sound_recorder deleted!");
127   }
```

```
}

/* Check if recording task is active */
int is_recorder_task_running(void) {
  // Return the status of the recorder task
  return recorder_task_flag;
}

/* Main recording task loop */
void loop_task_sound_recorder(void *pvParameters) {
  // Print a message indicating the start of the recording task
  Serial.println("loop_task_sound_recorder start...");
  // Initialize the total size of recorded data
  int total_size = 0;
  // Allocate memory in PSRAM for storing audio data
  char *buffer = (char *)heap_caps_malloc(MOLLOC_SIZE, MALLOC_CAP_SPIRAM);

  // Get the index for the next recording file
  int wav_index = read_file_num(RECORDER_FOLDER);
  // Generate the file name for the new recording
  String file_name = String(RECORDER_FOLDER) + "/recording_" + String(wav_index) + ".wav";
  // Write the WAV header to the file
  write_wav_header(file_name.c_str(), total_size);
  // Loop while the recorder task is running
  while (recorder_task_flag == 1) {
    // Get the available IIS data size
    int iis_buffer_size = audio_input_get_iis_data_available();
    // Loop while there is IIS data available
    while (iis_buffer_size > 0) {
      // Check if the buffer is full
      if ((total_size + 512) >= MOLLOC_SIZE) {
        // Stop the recorder task if the buffer is full
        recorder_task_flag = 0;
        break;
      }
      // Read IIS data into the buffer
      int real_size = audio_input_read_iis_data(buffer + total_size, 512);
      // Update the total size of recorded data
      total_size += real_size;
      // Decrease the available IIS data size
      iis_buffer_size -= real_size;
    }
  }
  // Append the recorded data to the file
```

```
172     append_file(file_name.c_str(), (uint8_t *)(buffer), total_size);
173     // Update the WAV header with the final file size
174     write_wav_header(file_name.c_str(), total_size);
175     // Load the recorded file into the audio output
176     audio_output_load_music(file_name.c_str());
177     // Free the allocated memory in PSRAM
178     heap_caps_free(buffer);
179     // Print a message indicating the end of the recording task
180     Serial.println("loop_task_sound_recorder stop...");
181     // Delete the current task
182     vTaskDelete(NULL);
183 }
184
185 /* Start player task */
186 void start_player_task(void) {
187     // Check if the player task is not already running
188     if (player_task_flag == 0) {
189         // Set the player task flag to running
190         player_task_flag = 1;
191         // Create a new task for playing sound
192         xTaskCreate(loop_task_play_handle, "loop_task_play_handle", 4096, NULL, 1, NULL);
193     }
194 }
195
196 /* Stop player task */
197 void stop_player_task(void) {
198     // Check if the player task is running
199     if (player_task_flag == 1) {
200         // Set the player task flag to stopped
201         player_task_flag = 0;
202         // Print a message indicating the deletion of the player task
203         Serial.println("loop_task_play_handle deleted!");
204     }
205 }
206
207 /* Check if player task is active */
208 int is_player_task_running(void) {
209     // Return the status of the player task
210     return player_task_flag;
211 }
212
213 /* Main player task loop */
214 void loop_task_play_handle(void *pvParameters) {
215     // Print a message indicating the start of the player task
```

```
216   Serial.println("loop_task_play_handle start...");
217   // Loop while the player task is running
218   while (player_task_flag == 1) {
219     // Handle the audio output loop
220     audio_output_loop();
221     // Check if the audio output is not running
222     if (!audio_output_is_running()) {
223       // Get the number of recorded files
224       int file_count = read_file_num(RECORDER_FOLDER);
225       // Generate the file name for the last recorded file
226       String file_name = String(RECORDER_FOLDER) + String("/") +
227 String(get_file_name_by_index(RECORDER_FOLDER, (file_count - 1)));
228       // Load the last recorded file into the audio output
229       audio_output_load_music(file_name.c_str());
230     }
231   }
232   // Print a message indicating the end of the player task
233   Serial.println("loop_task_play_handle stop...");
234   // Delete the current task
235   vTaskDelete(NULL);
236 }
```

Pressing different directions of the 5-way navigation switch will trigger corresponding function events.

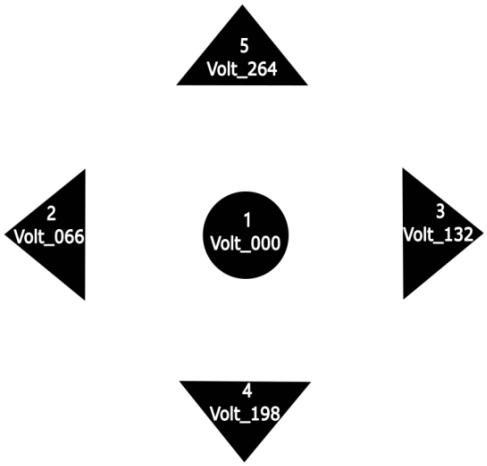
```
78  switch (button_key_value) {
79    case 1:
80      // If the button is pressed, start the recorder task
81      if (button_state == Button::KEY_STATE_PRESSED) {
82        start_recorder_task();
83      }
84      // If the button is released, stop the recorder task
85      else if (button_state == Button::KEY_STATE_RELEASED) {
86        stop_recorder_task();
87      }
88      break;
89    case 2:
90      // If the button is pressed, start the player task
91      if (button_state == Button::KEY_STATE_PRESSED) {
92        start_player_task();
93      }
94      break;
95    case 3:
96      // If the button is pressed, stop the player task
97      if (button_state == Button::KEY_STATE_PRESSED) {
98        stop_player_task();
99      }
```

```
100     case 4:
101     case 5:
102     default:
103         // Default case for other button key values
104       break;
105 }
```

After uploading the code, pressing different directions of the 5-way navigation switch will trigger corresponding function events. Note that directions 4 and 5 currently have no assigned functions.

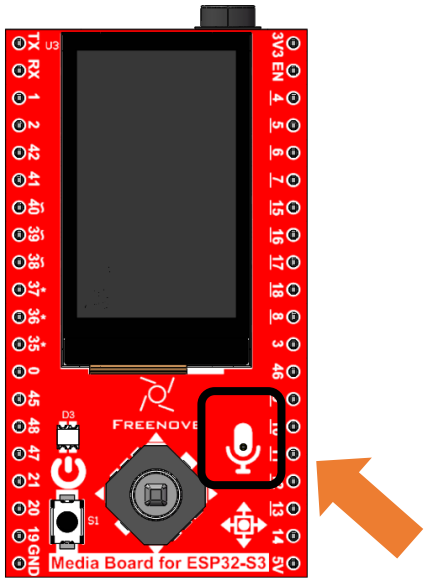
- Long-pressing Button 1 starts recording, and releasing it stops recording.
- Pushing Forward (2) plays back the recording.
- Pushing Backward (3) stops playback.

(**Refer to the diagram below for button numbering**.)



Notes:

1. **In the initial state, playback and stop functions (Directions 2 & 3) are disabled. They become active only after a recording is initiated via Button 1.**
2. **For optimal recording quality, please stay as close to the microphone as possible during recording (the microphone location is shown in the figure below).**



Chapter 11 ESP32_SR

Project 11.1 ESP32_SR

Component knowledge

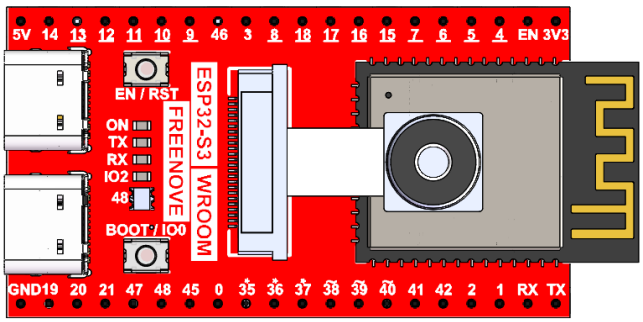
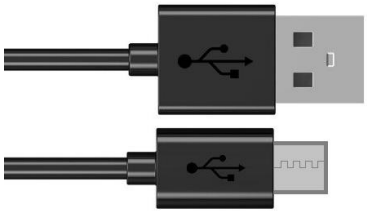
ESP32_SR

ESP32_SR, also known as the ESP-SR Speech Recognition Framework, is a voice recognition solution developed by Espressif Systems for its ESP32 series chips. Designed specifically for low-power, high-efficiency on-device speech processing, this technology integrates core functionalities such as acoustic front-end processing, voice wake-up, and command recognition.

By leveraging built-in hardware acceleration modules and optimized algorithms, the system can perform full-process audio operations—including signal acquisition, noise reduction, feature extraction, and semantic parsing—directly on embedded devices without relying on cloud servers.

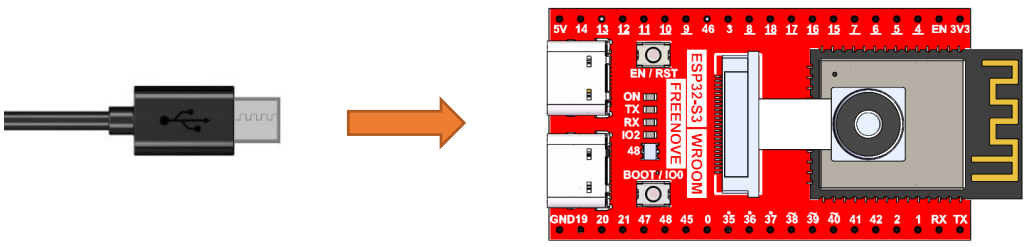
The framework supports custom wake words and multilingual command sets, allowing developers to quickly train and deploy models using the provided toolchain, enabling offline voice interaction capabilities for embedded devices.

Component List

Freenove Media Kit for ESP32-S3 x1 	USB cable x1 

Circuit

Connect Freenove Media Kit for ESP32-S3 to your computer using the USB cable.

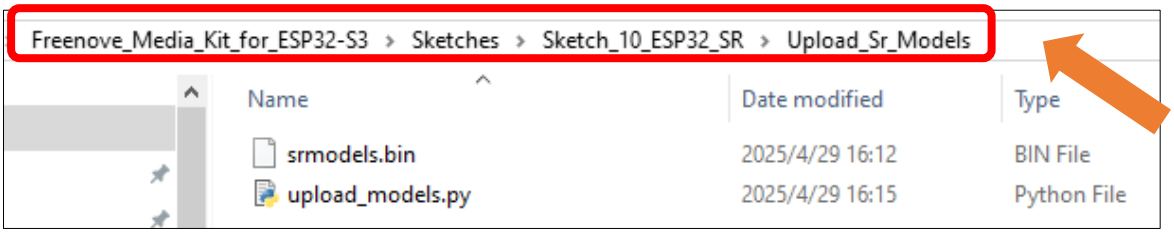


Sketch

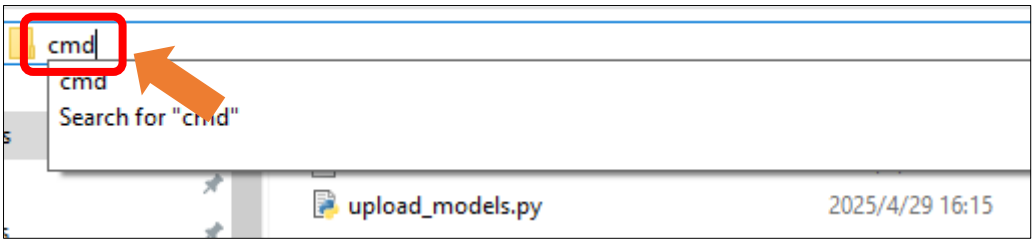
Uploading Voice Models

Connect Freenove Media Kit for ESP32-S3 to your computer using the USB cable.

Enter the path **Freenove_Media_Kit_for_ESP32-S3\Sketches\Sketch_10_ESP32_SR\Upload_Sr_Models**



Type cmd in the input bar and press Enter to open the Terminal.



Enter the following command in the terminal and press Enter to begin uploading the model to the Freenove Media Kit for ESP32-S3. During this process, **do not disconnect the USB cable** to avoid write failures or device damage.

```
python upload_models.py
```

Or

```
python3 upload_models.py
```

Important Notes:

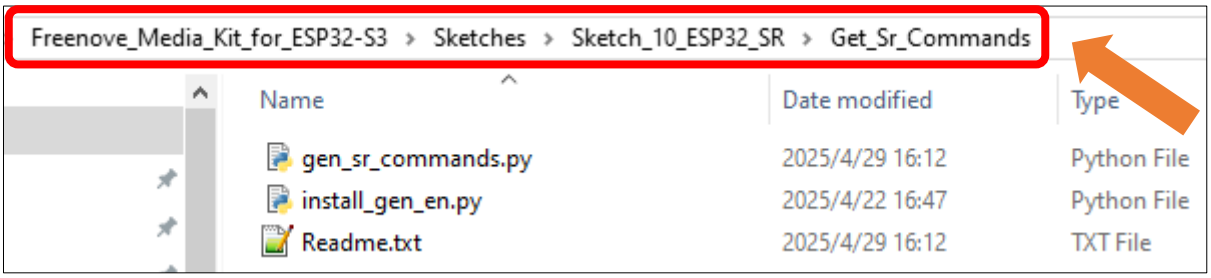
1. **Close the Arduino IDE Serial Monitor before running the upload_models.py script.**
2. **If the prompt shown in the right-side diagram appears, please verify whether the serial port is already in use.**

Generating Custom Voice Commands (On-Device)

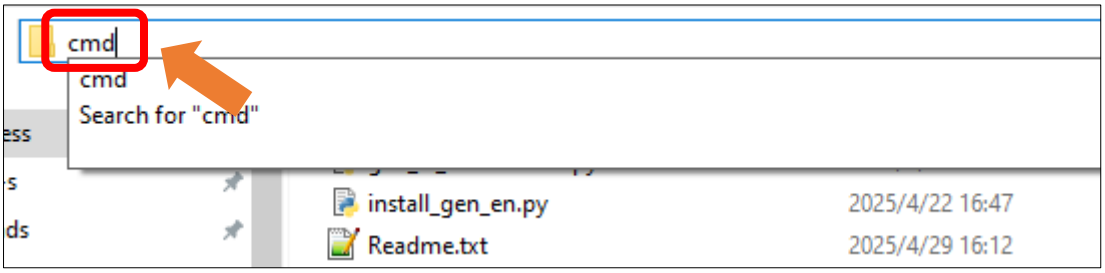
The system supports fully customizable voice commands that can be generated locally. Follow these steps to create your own:

Windows

Enter the path **Freenove_Media_Kit_for_ESP32-S3\Sketches\Sketch_10_ESP32_SR\ Get_Sr_Commands**



Type **cmd** in the input bar and press Enter to open the Terminal.



Input the following command, press Enter, and wait for the gen_en library to finish installing.

```
python3 install_gen_en.py
```

Input the following command and press Enter key to generate voice commands.

```
python3 gen_sr_commands.py "Turn on the light,Switch on the light;Turn off the light,Switch off the light,Go dark;Start fan;Stop fan"
```

Copy the generated commands to the code to use the voice commands.

```
F:\Freenove_Media_Kit_for_ESP32-S3\Sketches\Sketch_10_ESP32_SR\Get_Sr_Commands>python gen_sr_commands.py
ght;Turn off the light,Switch off the light;Go dark;Start fan;Stop fan"
enum {
  SR_CMD_TURN_ON_THE_LIGHT,
  SR_CMD_TURN_OFF_THE_LIGHT,
  SR_CMD_START_FAN,
  SR_CMD_STOP_FAN,
};
static const sr_cmd_t sr_commands[] = {
  { 0, "Turn on the light", "TkN nN jc LiT"},
  { 0, "Switch on the light", "SWgp nN jc LiT"},
  { 1, "Turn off the light", "TkN eF jc LiT"},
  { 1, "Switch off the light", "SWgp eF jc LiT"},
  { 1, "Go dark", "Gb DnRK"},
  { 2, "Start fan", "STnRT FaN"},
  { 3, "Stop fan", "STnP FaN"},
};
```

Important Notes:

1. You can customize the voice commands you need based on your actual requirements.

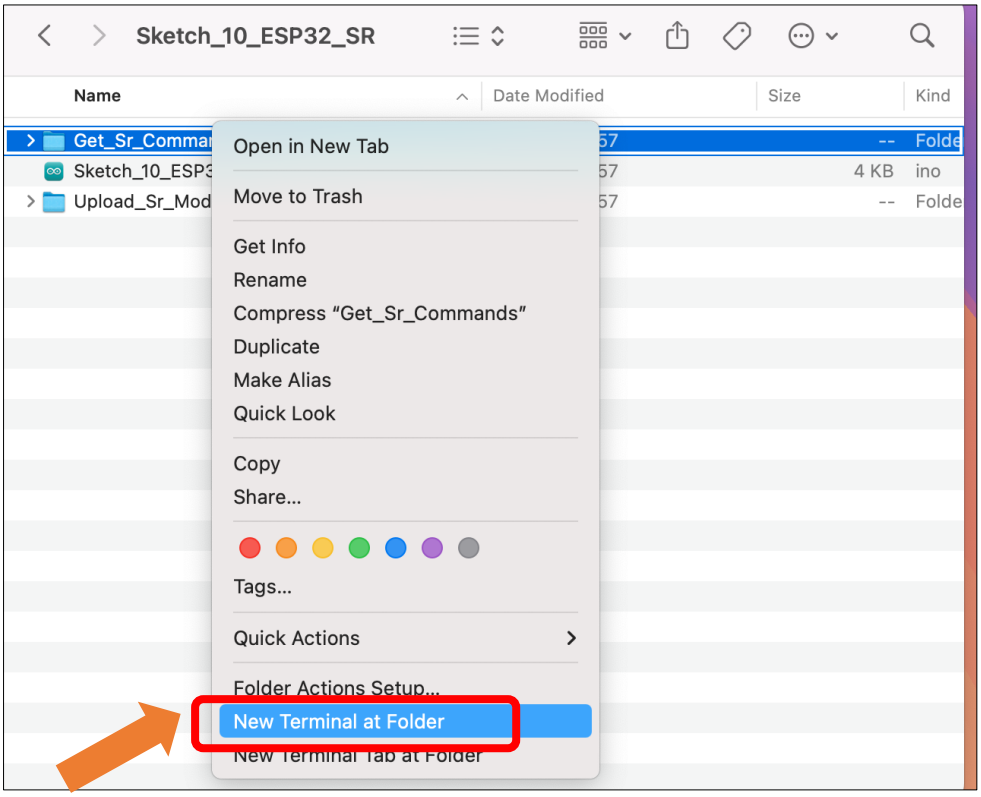
```
python gen_sr_commands.py "command sets"
```

The contents inside the quotation marks can be multiple command. Each segment separated by a comma represents different expressions of the same command (i.e., equivalent voice commands). Different commands are separated by semicolons, representing distinct operational instructions.

2. If you encounter a "command not found" error, please install the Python environment first.

Mac

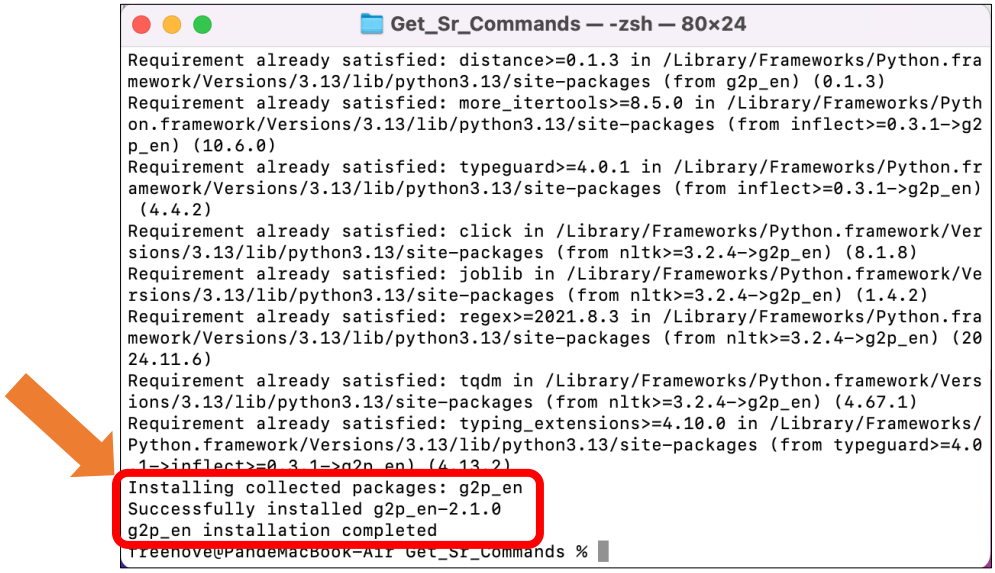
Open **Freenove_Media_Kit_for_ESP32-S3\Sketches\Sketch_10_ESP32_SR**, Press the **Ctrl** key **while right click "Get_Sr_Commands"**, click **New Terminal at Folder**



Run the following comamnd to install the g2p_en library.

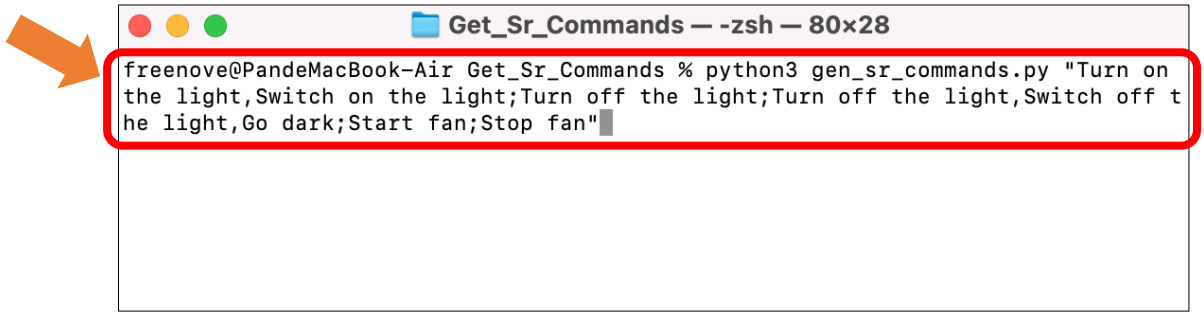
```
python3 install_gen_en.py
```

You will see the following promts upon the g2p_en library finishes installing.



Input the following command and press Enter key to generate the voice commands.

```
python3 gen_sr_commands.py "Turn on the light,Switch on the light;Turn off the light,Switch off the light,Go dark;Start fan;Stop fan"
```



Copy the generated commands to the code to use voice commands.

```
Get_Sr_Commands — -zsh — 92×24
Installing collected packages: g2p_en
Successfully installed g2p_en-2.1.0
freenove@PandeMacBook-Air Get_Sr_Commands % python3 gen_sr_commands.py "Turn on the light,Switch on the light;Turn off the light,Switch off the light,Go dark;Start fan;Stop fan"
[nltk_data] Error loading averaged_perceptron_tagger: <urlopen error
[nltk_data]     [SSL: CERTIFICATE_VERIFY_FAILED] certificate verify
[nltk_data]     failed: unable to get local issuer certificate
[nltk_data]     (_ssl.c:1028)>
enum {
  SR_CMD_TURN_ON_THE_LIGHT,
  SR_CMD_TURN_OFF_THE_LIGHT,
  SR_CMD_START_FAN,
  SR_CMD_STOP_FAN,
};
static const sr_cmd_t sr_commands[] = {
  { 0, "Turn on the light", "TkN nN jc LiT"},
  { 0, "Switch on the light", "SWgp nN jc LiT"},
  { 1, "Turn off the light", "TkN eF jc LiT"},
  { 1, "Switch off the light", "SWgp eF jc LiT"},
  { 1, "Go dark", "Gb DnRK"},
  { 2, "Start fan", "STnRT FaN"},
  { 3, "Stop fan", "STnP FaN"},
};
freenove@PandeMacBook-Air Get_Sr_Commands %
```

Important Notes:

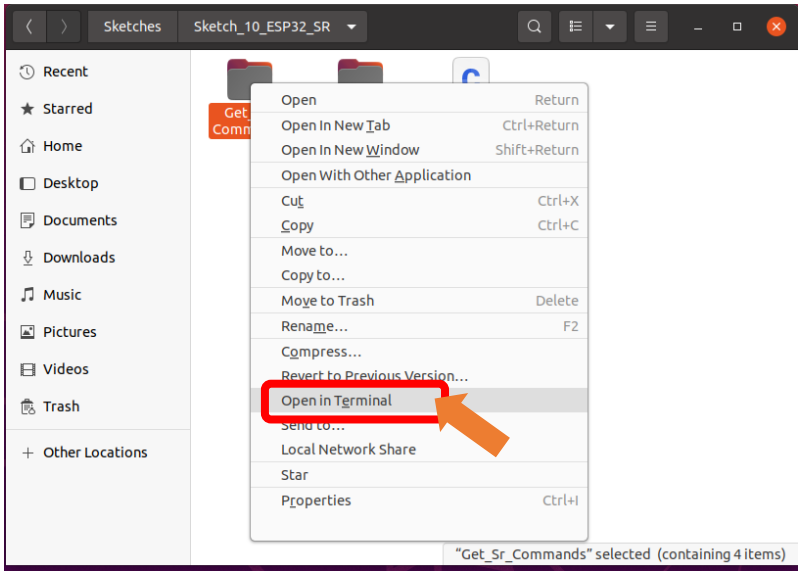
You can customize the voice commands you need based on your actual requirements.

```
python gen_sr_commands.py “command sets”
```

The contents inside the quotation marks can be multiple command. Each segment separated by a comma represents different expressions of the same command (i.e., equivalent voice commands). Different commands are separated by semicolons, representing distinct operational instructions.

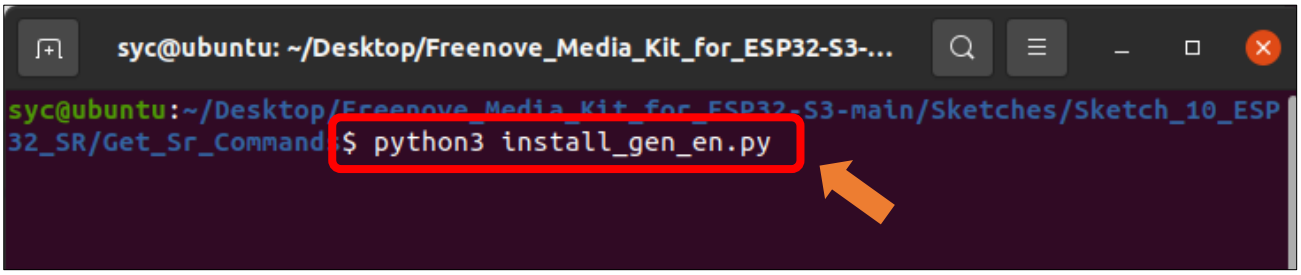
Linux

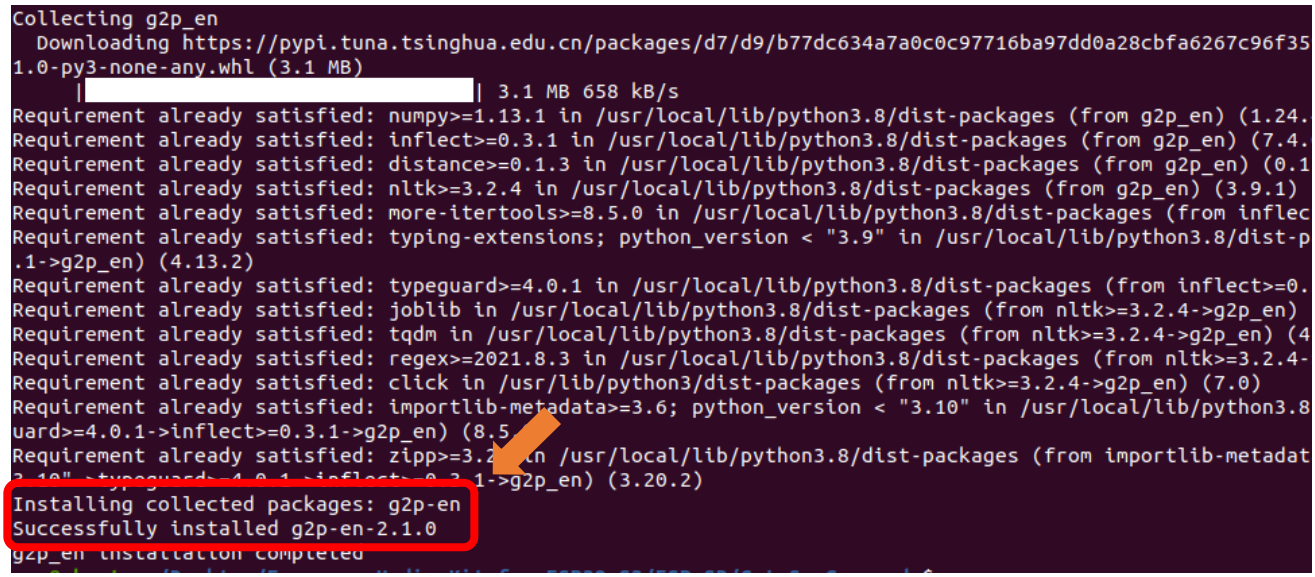
Right click **Freenove_Media_Kit_for_ESP32-S3\Sketches\Sketch_10_ESP32_SR**, right click Get_Sr_Commands, select **Open in Terminal.**



Run the following command to install the g2p_en library.

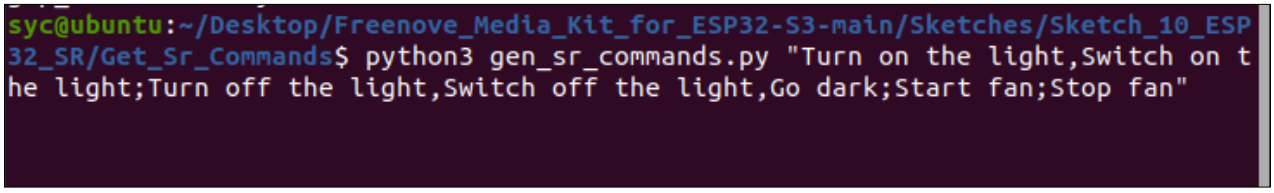
```
python3 install_gen_en.py
```



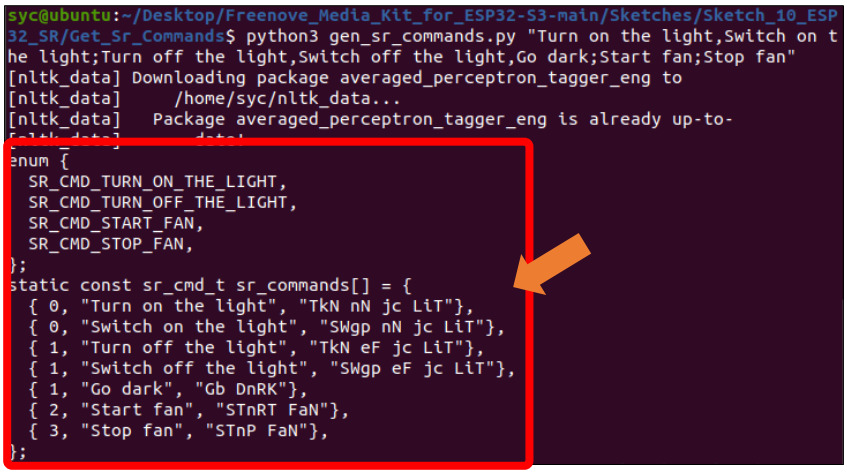


Input the following command and press Enter key to generate the voice commands.

```
python3 gen_sr_commands.py "Turn on the light,Switch on the light;Turn off the light,Switch off the light,Go dark;Start fan;Stop fan"
```



Copy the generated commands to the code to use voice commands.



Important Notes:

You can customize the voice commands you need based on your actual requirements.

```
python gen_sr_commands.py “commands sets”
```

The contents inside the quotation marks can be multiple command. Each segment separated by a comma represents different expressions of the same command (i.e., equivalent voice commands). Different commands are separated by semicolons, representing distinct operational instructions.

Sketch_10_ESP32_SR

The following is the program code:

```
#include "ESP_I2S.h"
#include "ESP_SR.h"

// Define I2S pins for audio input
#define I2S_PIN_BCK 3
#define I2S_PIN_WS 14
#define I2S_PIN_DIN 46

// Define pin for the light
#define LIGHT_PIN 2

// Create an I2S object
I2SClass i2s;

// Generated using the following command:
// python3 tools/gen_sr_commands.py "Turn on the light,Switch on the light;Turn off the
light,Switch off the light,Go dark;Start fan;Stop fan"
enum {
  SR_CMD_TURN_ON_THE_LIGHT,
  SR_CMD_TURN_OFF_THE_LIGHT,
};
static const sr_cmd_t sr_commands[] = {
  { 0, "Turn on the light", "TkN nN jc LiT" },       // Command ID 0: Turn on the light
  { 0, "Switch on the light", "SWgp nN jc LiT" },    // Command ID 0: Switch on the light
  { 1, "Turn off the light", "TkN eF jc LiT" },      // Command ID 1: Turn off the light
  { 1, "Switch off the light", "SWgp eF jc LiT" },  // Command ID 1: Switch off the light
  { 1, "Go dark", "Gb DnRK" },                      // Command ID 1: Go dark
};

/**
 * @brief Callback function for voice recognition events
 * @param event Type of event (e.g., wakeword detected, command recognized)
 * @param command_id ID of the recognized command
 * @param phrase_id ID of the recognized phrase within the command
 */
```

```
36  void onSrEvent(sr_event_t event, int command_id, int phrase_id) {
37    switch (event) {
38      case SR_EVENT_WAKEWORD:
39        Serial.println("WakeWord Detected!");  // Wakeword detected
40        break;
41      case SR_EVENT_WAKEWORD_CHANNEL:
42        Serial.printf("WakeWord Channel %d Verified!\n", command_id);  // Specific wakeword
43  channel verified
44        ESP_SR.setMode(SR_MODE_COMMAND);                              // Switch to command
45  detection mode
46        break;
47      case SR_EVENT_TIMEOUT:
48        Serial.println("Timeout Detected!");  // Timeout occurred
49        ESP_SR.setMode(SR_MODE_WAKEWORD);     // Switch back to wakeword detection mode
50        break;
51      case SR_EVENT_COMMAND:
52        Serial.printf("Command %d Detected! %s\n", command_id, sr_commands[phrase_id].str);  //
53  Command recognized
54        switch (command_id) {
55          case SR_CMD_TURN_ON_THE_LIGHT:
56            digitalWrite(LIGHT_PIN, HIGH);  // Turn on the light
57            break;
58          case SR_CMD_TURN_OFF_THE_LIGHT:
59            digitalWrite(LIGHT_PIN, LOW);  // Turn off the light
60            break;
61          default:
62            Serial.println("Unknown Command!");  // Unknown command received
63            break;
64        }
65        ESP_SR.setMode(SR_MODE_COMMAND);  // Allow for more commands to be given before timeout
66        break;
67      default:
68        Serial.println("Unknown Event!");  // Unknown event received
69        break;
70    }
71  }
72
73  /**
74   * @brief Setup function to initialize hardware and libraries
75   */
76  void setup() {
77    Serial.begin(115200);  // Initialize serial communication for debugging
78
79    pinMode(LIGHT_PIN, OUTPUT);    // Set light pin as output
```

```
80    digitalWrite(LIGHT_PIN, LOW);  // Ensure light is off initially
81
82    // Configure I2S for audio input
83    i2s.setPins(I2S_PIN_BCK, I2S_PIN_WS, -1, I2S_PIN_DIN);
84    i2s.setTimeout(1000);
85    i2s.begin(I2S_MODE_STD, 16000, I2S_DATA_BIT_WIDTH_16BIT, I2S_SLOT_MODE_STEREO);
86
87    // Initialize voice recognition library with commands and event callback
88    ESP_SR.onEvent(onSrEvent);
89    ESP_SR.begin(i2s, sr_commands, sizeof(sr_commands) / sizeof(sr_cmd_t), SR_CHANNELS_STEREO,
90  SR_MODE_WAKEWORD);
91  }
92
93  /**
94   * @brief Main loop function (empty in this sketch)
95   */
96  void loop() {}
97
```

Include the required header files.

```
1  #include "ESP_I2S.h"
2  #include "ESP_SR.h"
```

Define related pins.

```
4  // Define I2S pins for audio input
5  #define I2S_PIN_BCK 3
6  #define I2S_PIN_WS 14
7  #define I2S_PIN_DIN 46
8
9  // Define pin for the light
10 #define LIGHT_PIN 2
```

Custom voice commands, which need to be generated from locally produced voice command sets.

```
18 enum {
19   SR_CMD_TURN_ON_THE_LIGHT,
20   SR_CMD_TURN_OFF_THE_LIGHT,
21 };
22 static const sr_cmd_t sr_commands[] = {
23   { 0, "Turn on the light", "TkN nN jc LiT" },       // Command ID 0: Turn on the light
24   { 0, "Switch on the light", "SWgp nN jc LiT" },    // Command ID 0: Switch on the light
25   { 1, "Turn off the light", "TkN eF jc LiT" },      // Command ID 1: Turn off the light
26   { 1, "Switch off the light", "SWgp eF jc LiT" },  // Command ID 1: Switch off the light
27   { 1, "Go dark", "Gb DnRK" },                      // Command ID 1: Go dark
28 };
```

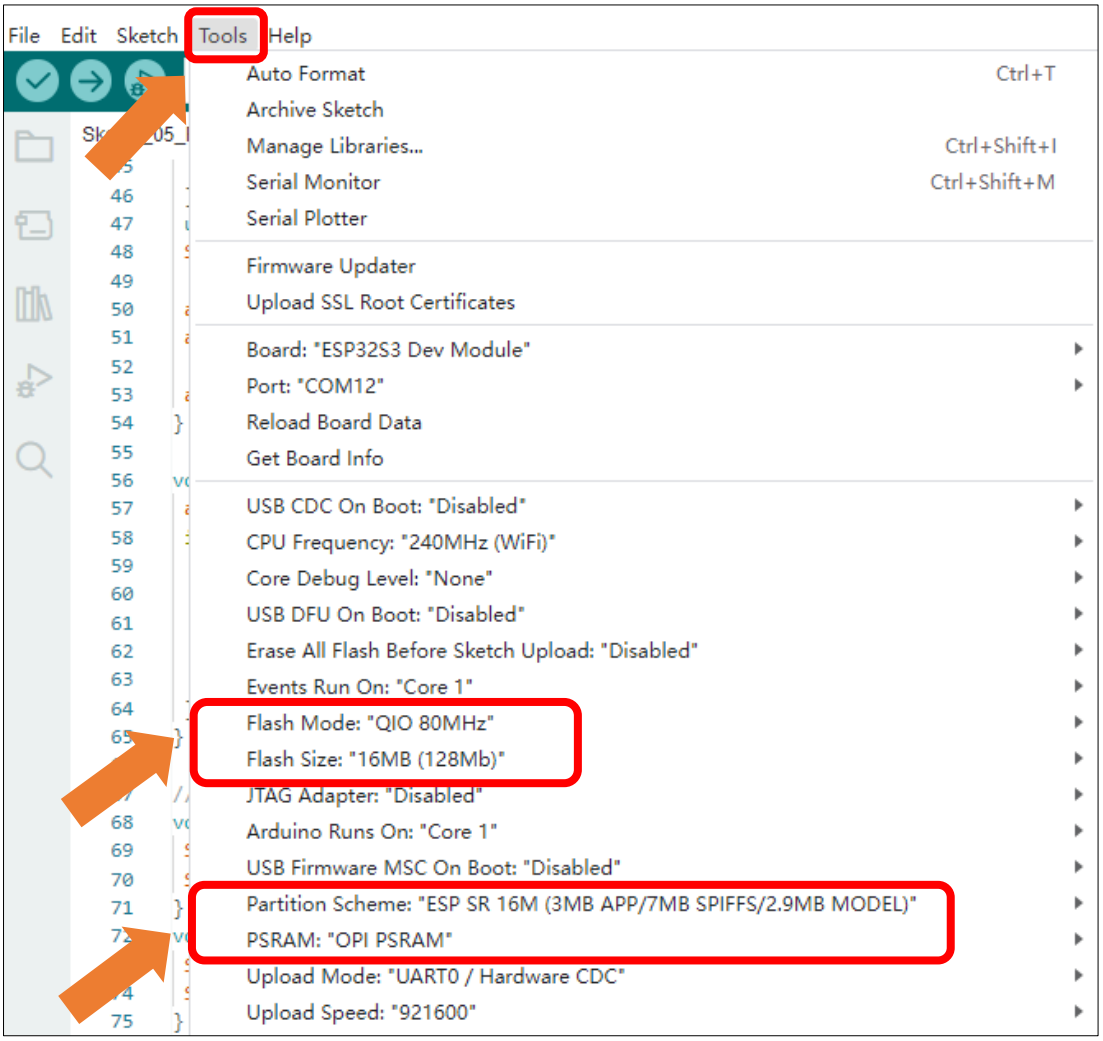
Initialize ESP32-SR and I2S.

```
80  i2s.setPins(I2S_PIN_BCK, I2S_PIN_WS, -1, I2S_PIN_DIN);
    i2s.setTimeout(1000);
```

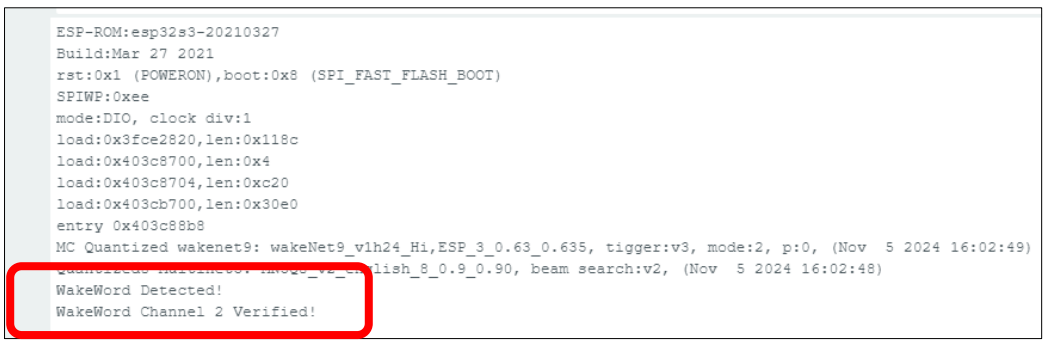
```
81  i2s.begin(I2S_MODE_STD, 16000, I2S_DATA_BIT_WIDTH_16BIT, I2S_SLOT_MODE_STEREO);
82
83  // Initialize voice recognition library with commands and event callback
84  ESP_SR.onEvent(onSrEvent);
85  ESP_SR.begin(i2s, sr_commands, sizeof(sr_commands) / sizeof(sr_cmd_t), SR_CHANNELS_STEREO,
86  SR_MODE_WAKEWORD);
```

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

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



After uploading the code, say the activating command ("Hi ESP") into the microphone. The serial monitor will display the following information:



After waking it, you can use voice commands to control it.

```
WakeWord Detected!
WakeWord Channel 1 Verified!
Command 0 Detected! Turn on the light
Command 1 Detected! Turn off the light
```

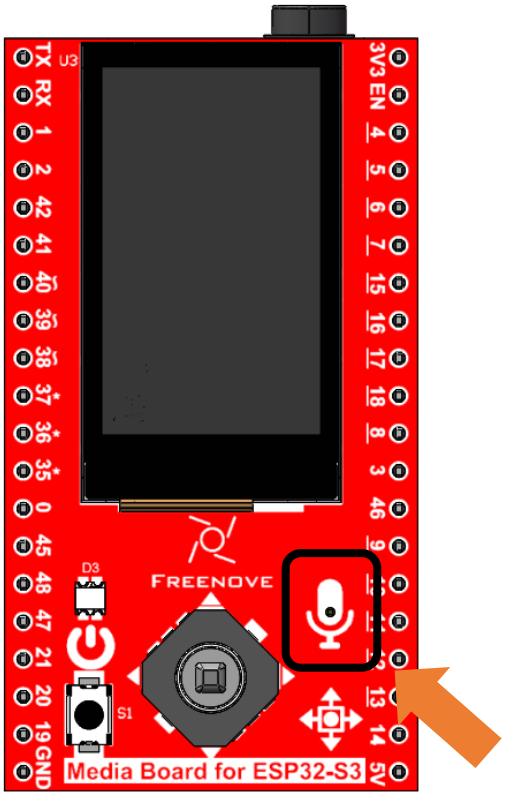
After activation, you can use the following voice commands to control the Freenove Media Kit for ESP32-S3.

Note: Commands with the same number perform the same function.

```
};
static const sr_cmd_t sr_commands[] = {
    { 0, "Turn on the light", "TkN nN jc LiT" },      // Command ID 0: Turn on the light
    { 0, "Switch on the light", "SWgp nN jc LiT" },   // Command ID 0: Switch on the light
    { 1, "Turn off the light", "TkN eF jc LiT" },     // Command ID 1: Turn off the light
    { 1, "Switch off the light", "SWgp eF jc LiT" },  // Command ID 1: Switch off the light
    { 1, "Go dark", "Gb DnRK" },                      // Command ID 1: Go dark
};
```

To ensure accurate voice command recognition for the Freenove Media Kit for ESP32-S3, please follow these guidelines:

- **Microphone Distance: Stay close to the microphone when issuing commands.**
- **Speech Speed: Maintain a moderate pace—avoid speaking too fast or too slow.**
- **Clear Pronunciation: Speak loudly and clearly, ensuring each word is articulated properly.**

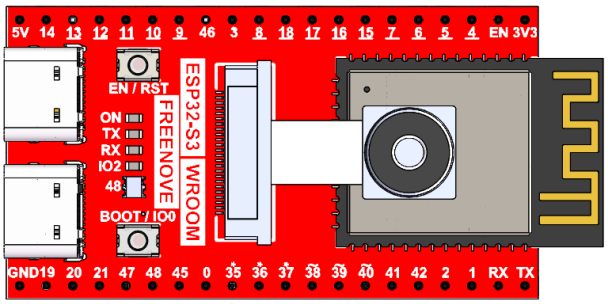
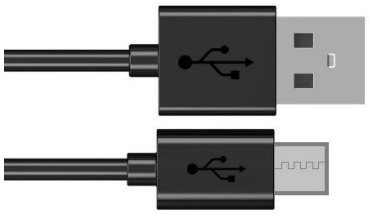


To change the activating word, please refer to the involved documentation and examples. For more detial ,refer to https://github.com/espressif/esp-sr

Chapter 12 LVGL

Project 12.1 LVGL

Component List

Freenove Media Kit for ESP32-S3 x1 	USB cable x1 

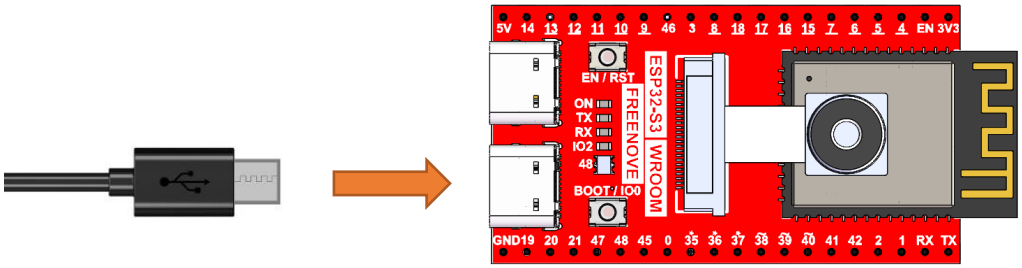
Component knowledge

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 Media Kit for ESP32-S3 to your computer using the USB cable.

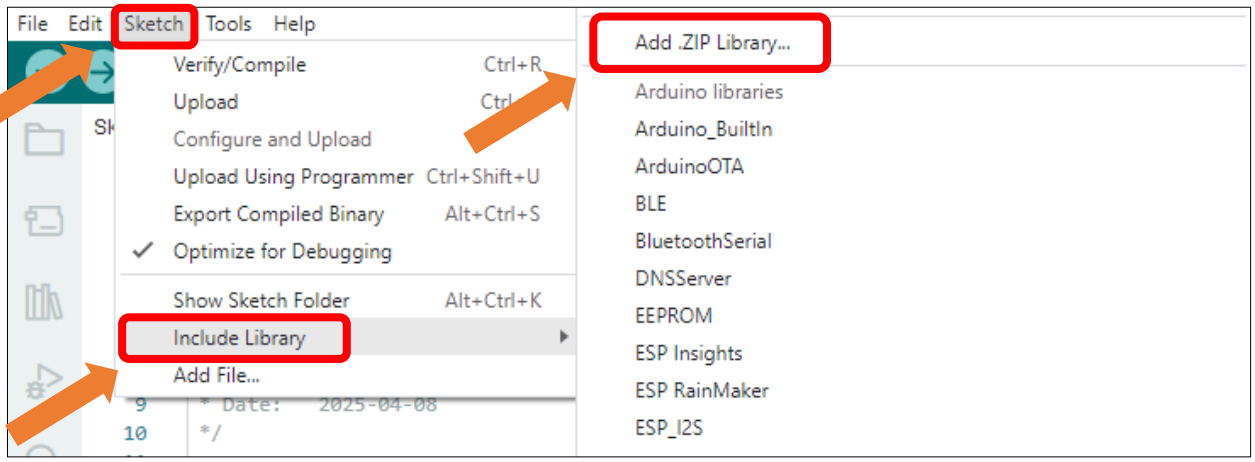


Sketch

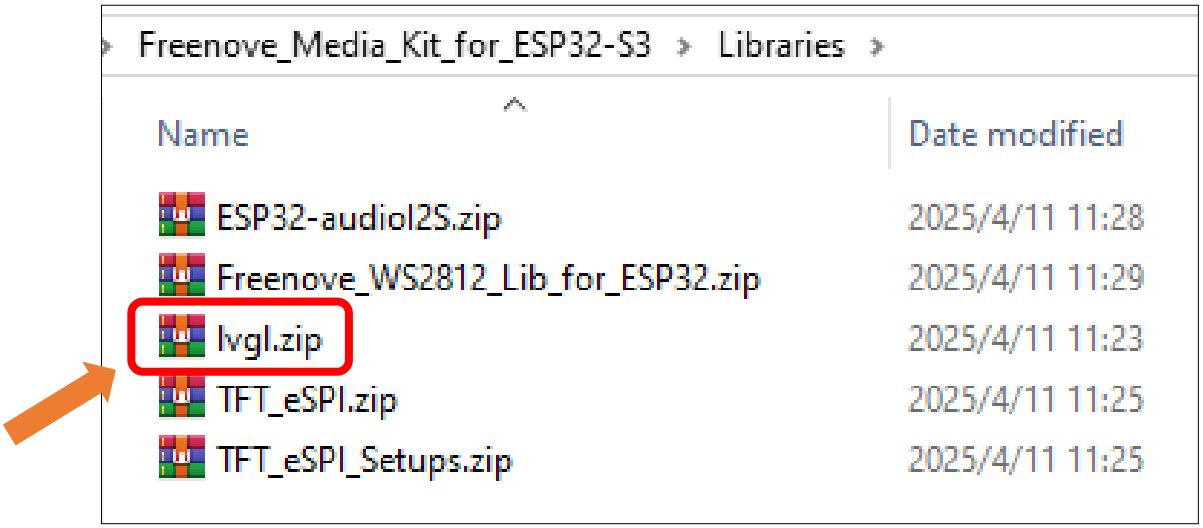
This code uses a library named "**lvgl**". If you have not installed it, please do so first.

How to install the library

Open Arduino IDE, click Sketch -> Include Library -> Add .ZIP Library. In the pop-up window, find the file named "**Freenove_Media_Kit_for_ESP32-S3\Libraries\lvgl.Zip**", which locates in this directory, and click OPEN.



Select lvgl.zip and click Open.



Please be aware that the lvgl.zip file is a pre-configured file library that is specifically designed for our product. Using the online "add library" function to add the lvgl library may cause compilation errors and render our product unusable.

Sketch_11_LVGL

Click on Upload to upload the code to the ESP32-S3. Once the code has completed uploading, you should see the screen displaying as shown below, which indicates successfully library configutaion and you can now begin learning LVGL.

The following is the program code:

```
#include "display.h"
#include <lvgl.h>

#define TFT_BL_PIN 20
Display screen;

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

  /*** Initialize screen ***/
  screen.init(TFT_DIRECTION);

  // Create a string to display the 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");

  // Create a label on the active screen and set its text
  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, -30);
```

```
24
25    // Create a left button on the active screen and set its size and position
26    lv_obj_t *btn_left = lv_btn_create(lv_scr_act());
27    lv_obj_set_size(btn_left, 60, 30);
28    lv_obj_align(btn_left, LV_ALIGN_CENTER, -50, 30);
29
30    // Create a right button on the active screen and set its size and position
31    lv_obj_t *btn_right = lv_btn_create(lv_scr_act());
32    lv_obj_set_size(btn_right, 60, 30);
33    lv_obj_align(btn_right, LV_ALIGN_CENTER, 50, 30);
34
35    // Create a label for the left button and set its text
36    lv_obj_t *label_btn_left = lv_label_create(btn_left);
37    lv_label_set_text(label_btn_left, "Left");
38    lv_obj_center(label_btn_left);
39
40    // Create a label for the right button and set its text
41    lv_obj_t *label_btn_right = lv_label_create(btn_right);
42    lv_label_set_text(label_btn_right, "Right");
43    lv_obj_center(label_btn_right);
44
45    // Create a group to manage the buttons
46    lv_group_t *group = lv_group_create();
47    lv_group_add_obj(group, btn_left);
48    lv_group_add_obj(group, btn_right);
49    lv_group_set_editing(group, true);
50    lv_indev_set_group(indev_keypad, group);
51
52    // Add event callbacks to the buttons
53    lv_obj_add_event_cb(btn_left, btn_event_cb, LV_EVENT_ALL, NULL);
54    lv_obj_add_event_cb(btn_right, btn_event_cb, LV_EVENT_ALL, NULL);
55
56    Serial.println("Setup done");
57  }
58
59  void loop() {
60    screen.routine(); /* Let the GUI do its work */
61    delay(5);
62  }
63
64  /**
65    * @brief Callback function for button events
66    * @param e Pointer to the event data
67    */
```

```
68  void btn_event_cb(lv_event_t *e) {
69    lv_event_code_t code = lv_event_get_code(e);
70    if (code == LV_EVENT_KEY) {
71      uint32_t c = lv_event_get_key(e);
72      if (c == LV_KEY_ENTER) {
73        Serial.printf("LV_KEY_ENTER\r\n");
74      } else if (c == LV_KEY_DOWN) {
75        Serial.printf("LV_KEY_DOWN\r\n");
76      } else if (c == LV_KEY_UP) {
77        Serial.printf("LV_KEY_UP\r\n");
78      } else if (c == LV_KEY_LEFT) {
79        Serial.printf("LV_KEY_LEFT\r\n");
80      } else if (c == LV_KEY_RIGHT) {
81        Serial.printf("LV_KEY_RIGHT\r\n");
82      }
83    }
84    if (code == LV_EVENT_RELEASED) {
85      Serial.printf("LV_EVENT_RELEASED\r\n");
86    }
87  }
```

Include the required header files.

```
1  #include "display.h"
2  #include <lvgl.h>
```

Define the backlight pin for the TFT screen.

```
4  #define TFT_BL_PIN 20
```

Define the TFT screen object.

```
5  Display screen;
```

Obtain the LVGL version and display on the screen.

```
14    // Create a string to display the LVGL version
15    String LVGL_Arduino = "Hello Arduino! ";
16    LVGL_Arduino += String('V') + lv_version_major() + "." + lv_version_minor() + "." +
      lv_version_patch();
17    Serial.println(LVGL_Arduino);
18    Serial.println("I am LVGL_Arduino");
19
20    // Create a label on the active screen and set its text
21    lv_obj_t *label = lv_label_create(lv_scr_act());
22    lv_label_set_text(label, LVGL_Arduino.c_str());
23    lv_obj_align(label, LV_ALIGN_CENTER, 0, -30);
```

Create and display the left and right buttons on the screen

```
25  // Create a left button on the active screen and set its size and position
26  lv_obj_t *btn_left = lv_btn_create(lv_scr_act());
27  lv_obj_set_size(btn_left, 60, 30);
28  lv_obj_align(btn_left, LV_ALIGN_CENTER, -50, 30);
```

```
29
30  // Create a right button on the active screen and set its size and position
31  lv_obj_t *btn_right = lv_btn_create(lv_scr_act());
32  lv_obj_set_size(btn_right, 60, 30);
33  lv_obj_align(btn_right, LV_ALIGN_CENTER, 50, 30);
34
35  // Create a label for the left button and set its text
36  lv_obj_t *label_btn_left = lv_label_create(btn_left);
37  lv_label_set_text(label_btn_left, "Left");
38  lv_obj_center(label_btn_left);
39
40  // Create a label for the right button and set its text
41  lv_obj_t *label_btn_right = lv_label_create(btn_right);
42  lv_label_set_text(label_btn_right, "Right");
43  lv_obj_center(label_btn_right);
```

Create an input device group, add the left and right buttons to the group, and enable keyboard control functionality.

```
45  // Create a group to manage the buttons
46  lv_group_t *group = lv_group_create();
47  lv_group_add_obj(group, btn_left);
48  lv_group_add_obj(group, btn_right);
49  lv_group_set_editing(group, true);
50  lv_indev_set_group(indev_keypad, group);
```

Create event callback functions for the left and right buttons, and configure them to listen for all events.

```
52  // Add event callbacks to the buttons
53  lv_obj_add_event_cb(btn_left, btn_event_cb, LV_EVENT_ALL, NULL);
54  lv_obj_add_event_cb(btn_right, btn_event_cb, LV_EVENT_ALL, NULL);
```

Implement a 5-way directional key event handler to detect and respond to key values from the 5-way navigation button.

```
68  void btn_event_cb(lv_event_t *e) {
69    lv_event_code_t code = lv_event_get_code(e);
70    if (code == LV_EVENT_KEY) {
71      uint32_t c = lv_event_get_key(e);
72      if (c == LV_KEY_ENTER) {
73        Serial.printf("LV_KEY_ENTER\r\n");
74      } else if (c == LV_KEY_DOWN) {
75        Serial.printf("LV_KEY_DOWN\r\n");
76      } else if (c == LV_KEY_UP) {
77        Serial.printf("LV_KEY_UP\r\n");
78      } else if (c == LV_KEY_LEFT) {
79        Serial.printf("LV_KEY_LEFT\r\n");
80      } else if (c == LV_KEY_RIGHT) {
81        Serial.printf("LV_KEY_RIGHT\r\n");
82      }
```

```
83      }
84      if (code == LV_EVENT_RELEASED) {
85        Serial.printf("LV_EVENT_RELEASED\r\n");
86      }
87  }
```

After the code uploads, the TFT screen will display the following contents.

The 5-way switch allows you to cycle through options. You can switch the selected button by moving the focus box.

When the focus is on the left button, push the switch to the right moves it to the right button, and pushing Left brings it back to the left.

If the focus is already on the far right, pushing Right again makes it loop back to the left button.

Similarly, if the focus is on the far left, pushing Left makes it loop to the right button.

Pressing the center button highlights and confirms the selection of the current button.

Reference

`lv_obj_t * lv_label_create(lv_obj_t * parent);`
This function creates a label control.
`lv_obj_t * lv_btn_create(lv_obj_t * parent);`
This function creates a button control.
`void lv_label_set_text(lv_obj_t * obj, const char * text);`
This function sets the corresponding text displayed by the label control **Paramters:** obj: label object text: displayed string

```
void lv_obj_set_size(struct _lv_obj_t * obj, lv_coord_t w, lv_coord_t h);
```

This function sets the width and height of the control

Paramters:

obj: control object

w: control width

h: control height

```
void lv_obj_align(struct _lv_obj_t * obj, lv_align_t align, lv_coord_t x_ofs, lv_coord_t y_ofs);
```

This function controls the alignment of controls

Paramters:

obj: control object

align:

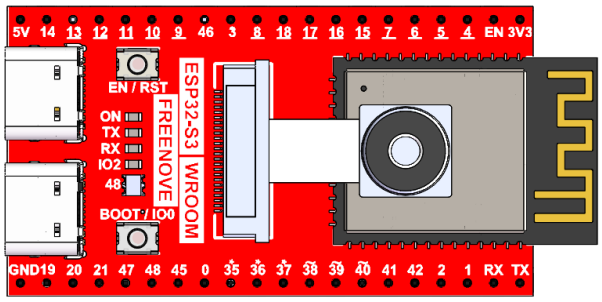
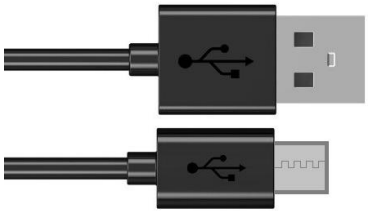
LV_ALIGN DEFAULT=0	// Use default alignment
LV_ALIGN TOP LEFT	// The top left corner of the parent object's content area
LV_ALIGN TOP MID	// The top middle of the parent object's content area
LV _ALIGN TOP RIGHT	// The top right corner of the parent object's content area
LV_ALIGN_BOTTOM_LEFT	// The bottom left corner of the parent object's content
LV ALIGN BOTTOM_MID	// The bottom middle of the parent object's content area
LV_ALIGN_BOTTOM_RIGHT	// The bottom right corner of the parent object's content area
LV_ALIGN LEFT MID	// The left middle of the parent object's content area
LV_ALIGN RIGHT MID	// The right middle of the parent object's content area
LV _ALIGN_CENTER	// Centered horizontally and vertically with the parent object
LV_ALIGN OUT TOP_LEFT	// Aligned above and to the left of the parent object
LV_ALIGN OUT TOP_MID	// Directly above the parent object
LV_ALIGN OUT TOP RIGHT	// Above the parent object and aligned to the right
LV_ALIGN_OUT_BOTTOM_LEFT	// Below the parent object and aligned to the left
LV_ALIGN OUT BOTTOM_MID	// Horizontally centered with the parent object
LV ALIGN OUT BOTTOM RIGHT	// Below the parent object and aligned to the right
LV_ALIGN OUT LEFT TOP	// Left and top aligned with the parent object
LV_ALIGN OUT LEFT_MID	// Vertically centered on the left side of the parent object
LV_ALIGN OUT LEFT BOTTOM	// Left and bottom aligned with the parent object
LV_ALIGN OUT_RIGHT TOP	// Right and top aligned with the parent object
LV_ALIGN OUT RIGHT_MID	// Vertically centered on the right side of the parent object
LV_ALIGN OUT RIGHT BOTTOM	// Right and bottom aligned to the parent object
x_ofs: x-axis offset	y_ofs: y-axis offset

Chapter 13 LVGL LEDPixel

In previous chapters, we learned about LEDPixel and LVGL. This chapter will focus on how to integrate and apply these two technologies together.

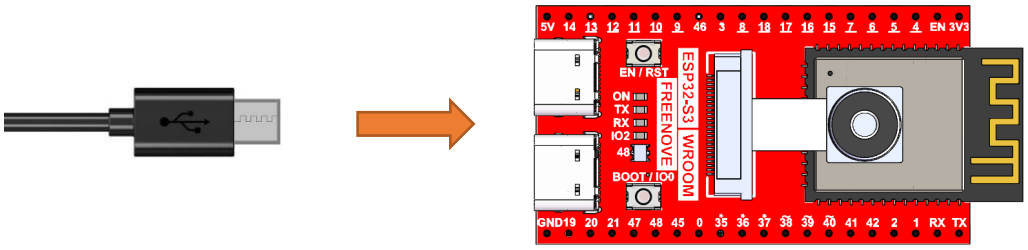
Project 13.1 LVGL LEDPixel

Component List

Freenove Media Kit for ESP32-S3 x1	USB cable x1
	

Circuit

Connect Freenove Media Kit for ESP32-S3 to your computer using the USB cable.



Sketch

Before upload the sketch of this chapter, please ensure that the libraries LedPixel, TFT_eSPI and lvglare already installed.

The following is the program code:

```
#include "display.h"
#include <lvgl.h>
#include "ws2812_ui.h"

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

```

```
7  void setup()
8  {
9    /* Prepare for possible serial debug */
10   Serial.begin(115200); // Initialize serial communication at 115200 baud rate
11
12   /*** Initialize the screen ***/
13   screen.init(TFT_DIRECTION); // Initialize the display
14
15   // Create a string to display LVGL version information
16   String LVGL_Arduino = "Hello Arduino! ";
     LVGL_Arduino += String('V') + lv_version_major() + "." + lv_version_minor() + "." +
17 lv_version_patch();
18
19   // Print LVGL version information to the serial monitor
20   Serial.println(LVGL_Arduino);
21   Serial.println("I am LVGL_Arduino");
22
23   // Setup the WS2812 UI screen
24   setup_scr_ws2812(&guider_ws2812_ui);
25   // Load the WS2812 UI screen
26   lv_scr_load(guider_ws2812_ui.ws2812);
27
28   // Print setup completion message to the serial monitor
29   Serial.println("Setup done");
30 }
31 void loop()
32 {
33   screen.routine(); /* Let the GUI do its work */ // Handle routine display tasks
34   delay(5);         // Add a small delay to prevent the loop from running too fast
35 }
```

Include the required header files.

```
1  #include "display.h"
2  #include <lvgl.h>
3  #include "ws2812_ui.h"
```

Define TFT screen object

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

TFT screen initialization

```
12 /*** Initialize the screen ***/
13 screen.init(TFT_DIRECTION); // Initialize the display
```

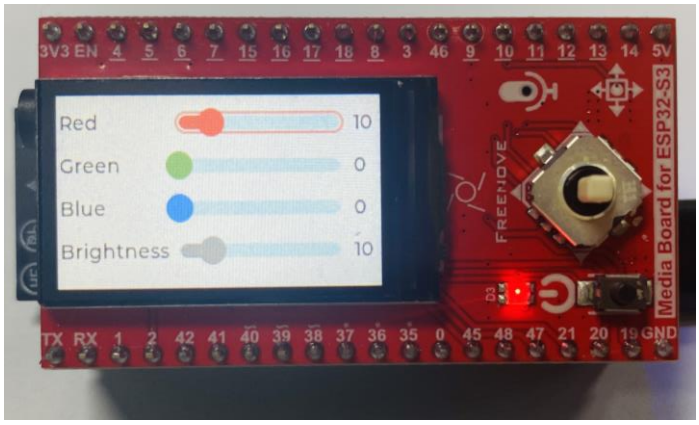
Load LEDPixel UI

```
14 // Setup the WS2812 UI screen
15 setup_scr_ws2812(&guider_ws2812_ui);
```

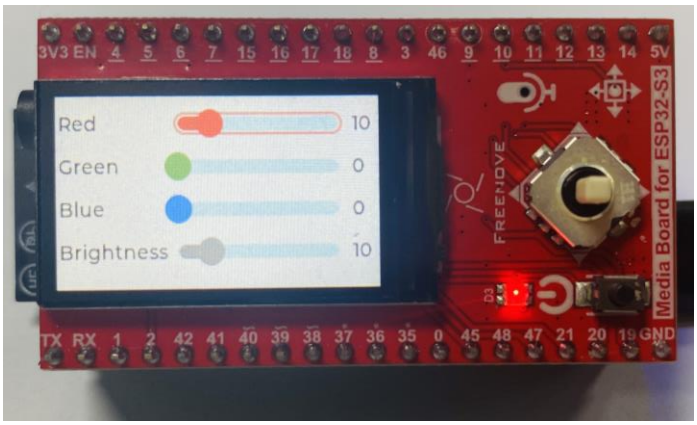
To execute all pending LVGL tasks, this function must be called continuously.

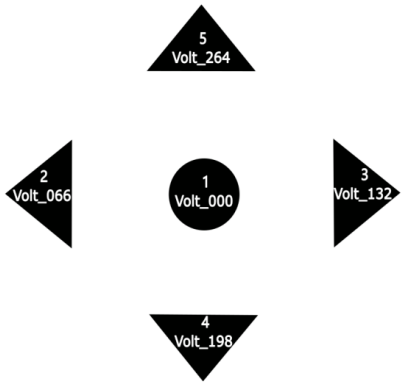
```
25 screen.routine(); /* Let the GUI do its work */ // Handle routine display tasks
```

After the sketch is uploaded, the TFT screen will display the following interface.

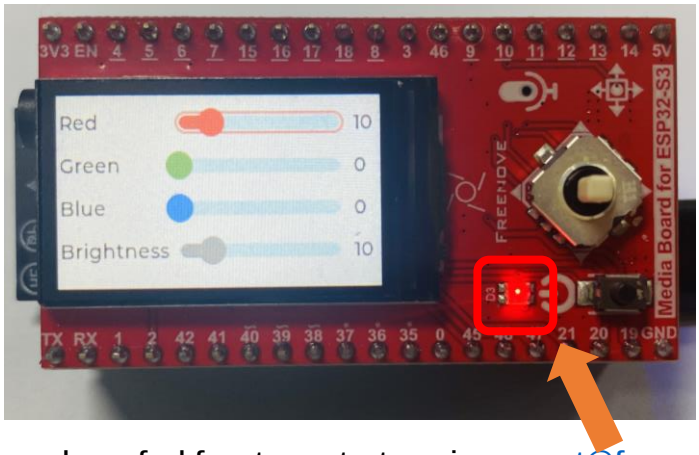


Pressing different directions on the 5-way switch triggers corresponding functional events:
Switches 4 and 5 navigate between adjustable parameter fields
Switches 2 and 3 decrease or increase the threshold value of the currently selected parameter field respectively. (**Refer to the figure below for button numbering definition**)





You can change the color and brightness of the LEDPixel by adjusting the parameters.



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

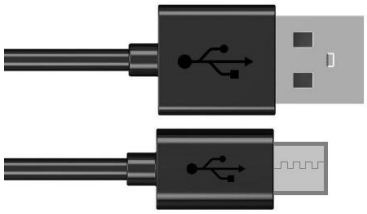
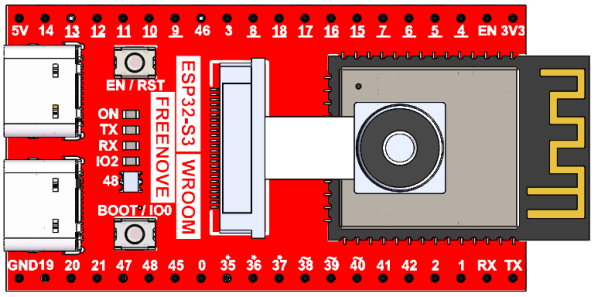
Chapter 14 LVGL Camera

In previous chapters, we learned about Camera and LVGL. This chapter will focus on how to integrate and apply these two technologies together.

Project 14.1 LVGL Camera

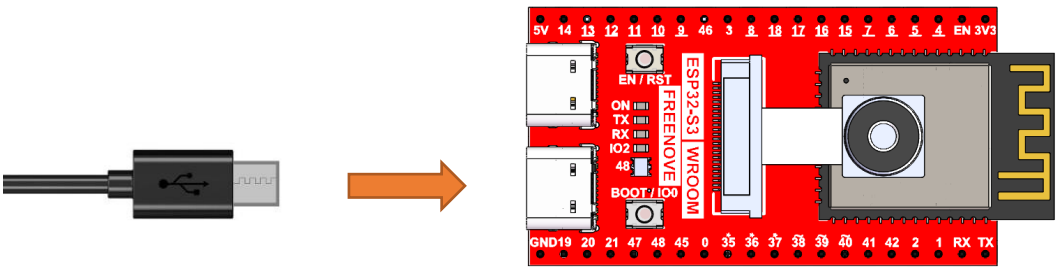
Capture image data using the camera module and display it on the TFT screen.

Component List

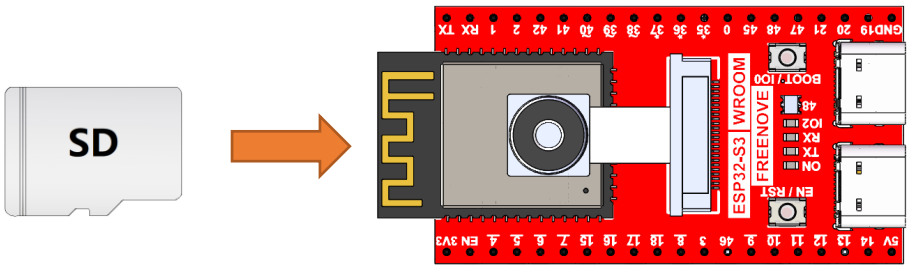
SD card x1 	USB cable x1 
Freenove Media Kit for ESP32-S3 x1 	Card reader x1 (random color) 

Circuit

Connect Freenove Media Kit for ESP32-S3 to your computer using the USB cable.



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



Sketch

Sketch_14_LVGL_Camera

The following is the program code:

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

#define SD_MMC_CMD 38  // Please do not modify it.
#define SD_MMC_CLK 39  // Please do not modify it.
#define SD_MMC_D0 40   // Please do not modify it.

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

void setup() {
  /* Prepare for possible serial debug */
  Serial.begin(115200);  // Initialize serial communication at 115200 baud rate

  camera_init(0);
  sdmmc_init(SD_MMC_CLK, SD_MMC_CMD, SD_MMC_D0);
  // remove_dir(CAMERA_FOLDER);
  // create_dir(CAMERA_FOLDER);

  /*** Initialize the screen ***/
  screen.init(TFT_DIRECTION);  // Initialize the display

  // Create a string to display LVGL version information
  String LVGL_Arduino = "Hello Arduino! ";
  LVGL_Arduino += String('V') + lv_version_major() + "." + lv_version_minor() + "." +
lv_version_patch();

  // Print LVGL version information to the serial monitor
  Serial.println(LVGL_Arduino);
  Serial.println("I am LVGL_Arduino");

```

```
29    setup_scr_camera(&guider_camera_ui);
30    lv_scr_load(guider_camera_ui.camera);
31
32    // Print setup completion message to the serial monitor
33    Serial.println("Setup done");
34  }
35
36  void loop() {
37    screen.routine(); /* Let the GUI do its work */  // Handle routine display tasks
38    delay(5);         // Add a small delay to prevent the loop from running too fast
39  }
```

Include the required libraries.

```
1  #include "display.h"
2  #include "camera_ui.h"
```

Define SD card pins.

```
4  #define SD_MMC_CMD 38  // Please do not modify it.
5  #define SD_MMC_CLK 39  // Please do not modify it.
6  #define SD_MMC_D0 40   // Please do not modify it.
```

Declare TFT screen object.

```
8  Display screen;
```

Initialize SD card and camera.

```
14  camera_init(0);
15  sdmmc_init(SD_MMC_CLK, SD_MMC_CMD, SD_MMC_D0);
```

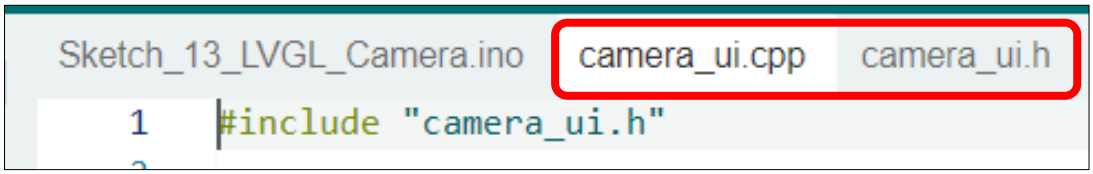
Load the camera interface to the TFT screen.

```
25  setup_scr_camera(&guider_camera_ui);
26  lv_scr_load(guider_camera_ui.camera);
```

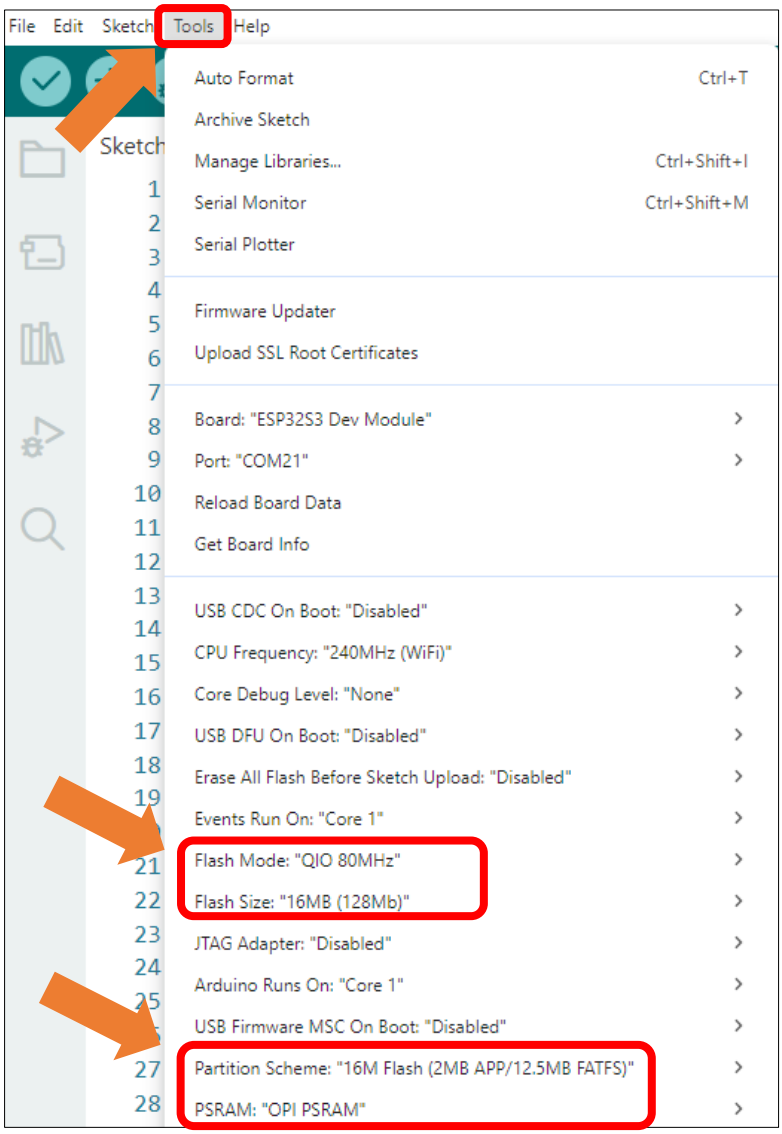
To execute all pending LVGL tasks, this function must be called continuously.

```
45  screen.routine(); /* Let the GUI do its work */  // Handle routine display tasks
```

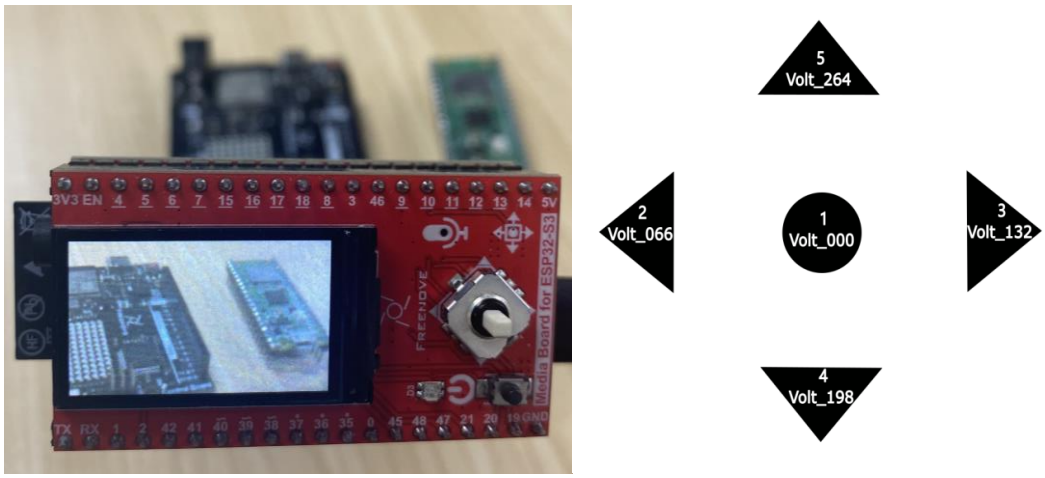
If you are interesting in the implementation of functions, you can check them out here.



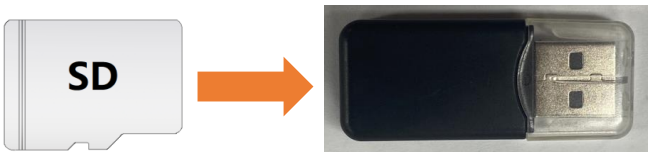
It is necessary to change the settings in Arduino IDE before clicking the Uploading button, as shown below.
Caution: Incorrect settings will result in compilation error or uploading failure. To achieve desired result, please configure exactly the same as below.



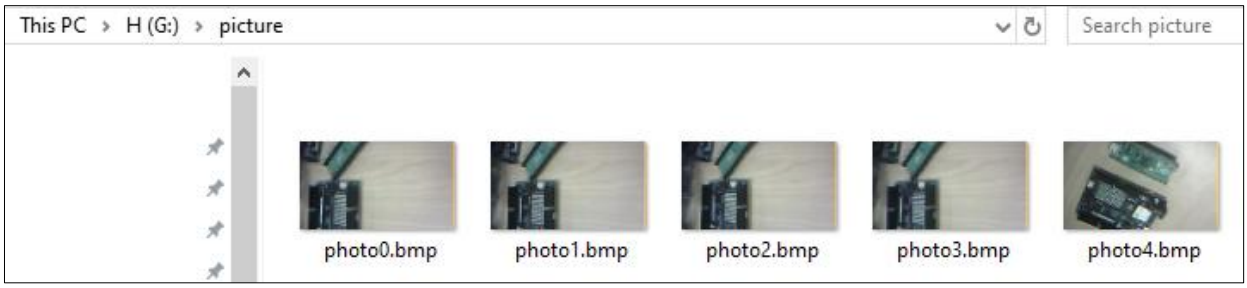
After uploading the code, the image from the camera will be displayed on the TFT screen. Pressing the button1 will automatically save the photo to the SD card.



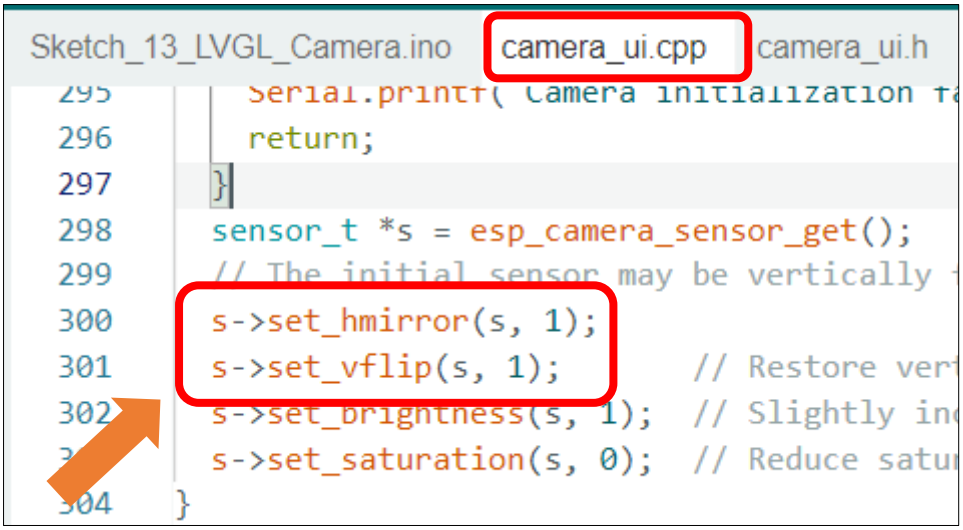
After taking photos, please remove the SD card and insert it into a card reader, then connect the reader to your computer.



In the SD card's directory, there is a folder named 'Video' which contains the pictures you just captured.



Notice: Camera display performance may vary across different module models. Some devices may exhibit mirrored imaging. In such cases, adjust the horizontal flip and vertical flip parameters by modifying the following two lines of code in camera_ui.cpp:



Parameter Description:

- 0: Normal display
- 1: Flip (mirror)

To achieve the desired display, configure the settings according to real-time preview feedback during setup.

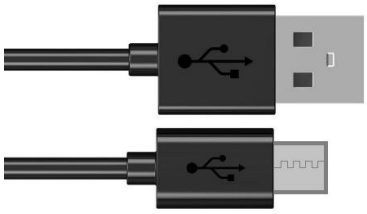
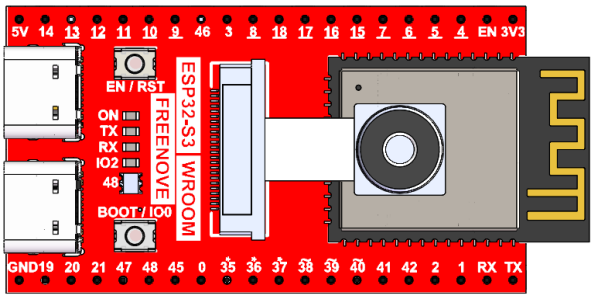
If you have any concerns, please feel free to contact us via support@freenove.com

Chapter 15 LVGL Picture

In this chapter, you will learn how to read image data from the SD card and display it on the TFT screen.

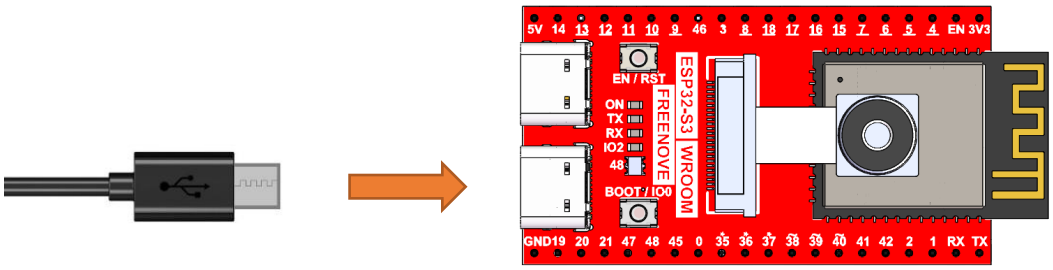
Project 15.1 LVGL Picture

Component List

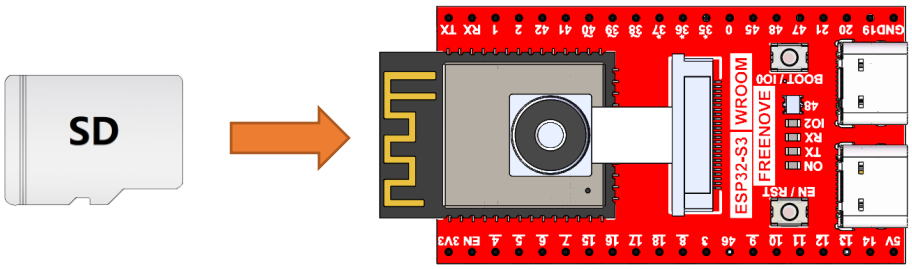
SD card x1 	USB cable x1 
Freenove Media Kit for ESP32-S3 x1 	Card reader x1 (random color) 

Circuit

Connect Freenove Media Kit for ESP32-S3 to your computer using the USB cable.



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



Sketch

Sketch_15_LVGL_Picture

The following is the program code:

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

#define SD_MMC_CMD 38  //Please do not modify it.
#define SD_MMC_CLK 39  //Please do not modify it.
#define SD_MMC_D0 40   //Please do not modify it.

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

void setup() {
  /* Prepare for possible serial debug */
  Serial.begin(115200);  // Initialize serial communication at 115200 baud rate

  /* Initialize SD card */
  sdmmc_init(SD_MMC_CLK, SD_MMC_CMD, SD_MMC_D0);
  // create_dir(PICTURE_FOLDER);


  /*** Initialize the screen ***/
  screen.init(TFT_DIRECTION);  // Initialize the display

  // Create a string to display LVGL version information
  String LVGL_Arduino = "Hello Arduino! ";
  LVGL_Arduino += String('V') + lv_version_major() + "." + lv_version_minor() + "." +
lv_version_patch();

  // Print LVGL version information to the serial monitor
  Serial.println(LVGL_Arduino);
  Serial.println("I am LVGL_Arduino");

```

```
29    setup_scr_picture(&guider_picture_ui);
30    lv_scr_load(guider_picture_ui.picture);
31
32    // Print setup completion message to the serial monitor
33    Serial.println("Setup done");
34  }
35
36  void loop() {
37    screen.routine(); /* Let the GUI do its work */  // Handle routine display tasks
38    delay(5);          // Add a small delay to prevent the loop from running too fast
39  }
```

Include the required libraries.

```
1  #include "display.h"
2  #include "camera_ui.h"
```

Define SD card pins.

```
4  #define SD_MMC_CMD 38  // Please do not modify it.
5  #define SD_MMC_CLK 39  // Please do not modify it.
6  #define SD_MMC_D0 40   // Please do not modify it.
```

Declare TFT screen object.

```
8  Display screen;
```

Initialize the SD card.

```
15  sdmmc_init(SD_MMC_CLK, SD_MMC_CMD, SD_MMC_D0);
```

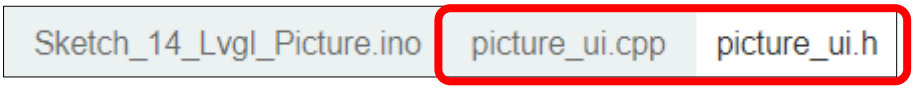
TFT screen initialization.

```
19  screen.init(TFT_DIRECTION);  // Initialize the display
```

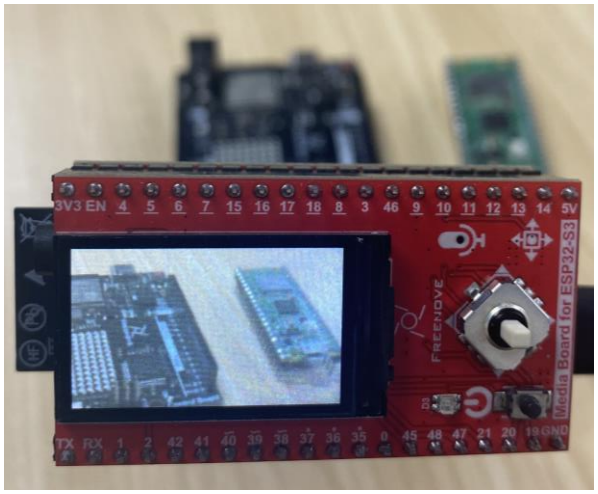
Initialize the UI component for image browsing

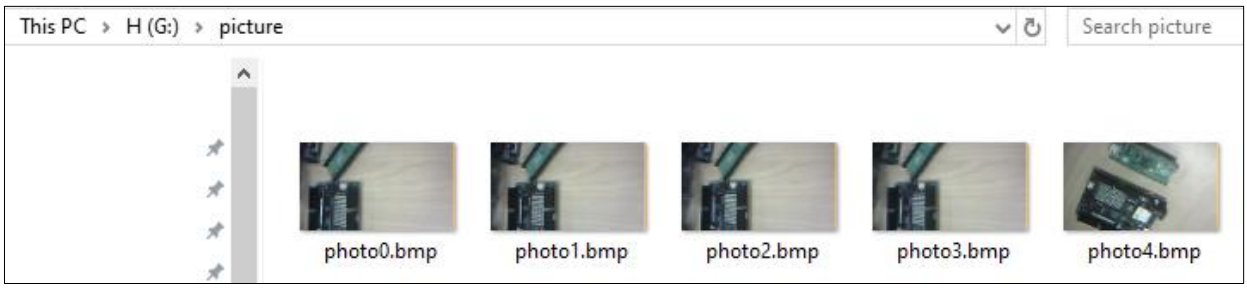
```
29  setup_scr_picture(&guider_picture_ui);
30  lv_scr_load(guider_picture_ui.picture);
```

If you are interesting in the implementation of functions, you can check them out here.



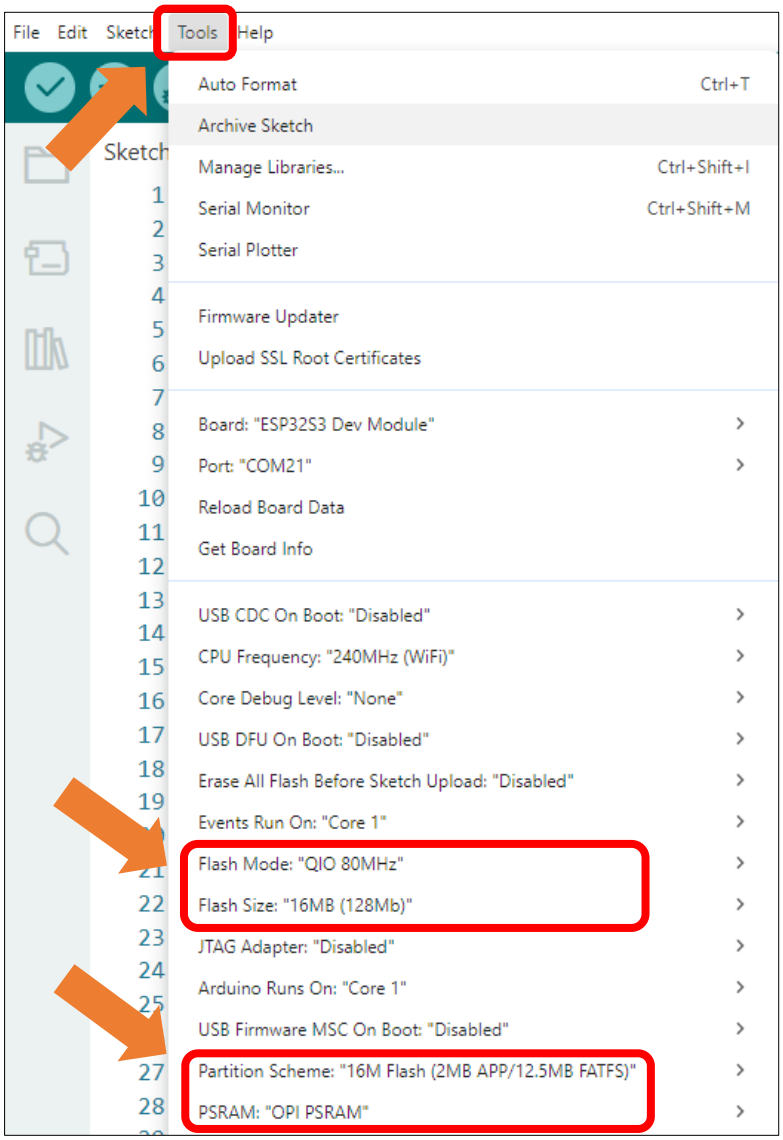
Before uploading the code, you need to use the sample code in Chapter 13 to take a picture and store it in the SD card. Make sure there are image files in the picture folder.





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

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



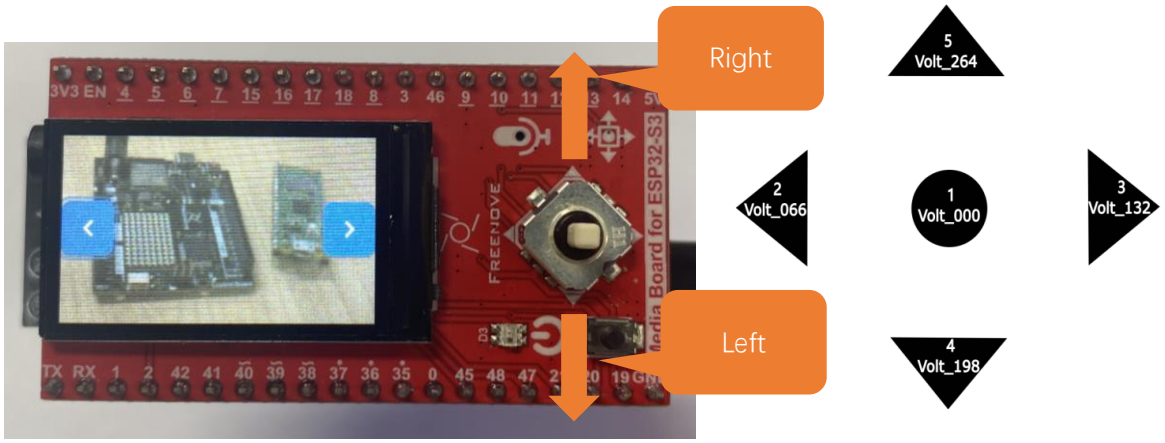
After uploading the code, the TFT display will show images from the SD card. The 5-way navigation key operates as follows:

Switches 5 (Right) and 4 (Left): Navigate horizontally between selection frames

Button 1 (Center): Confirms selection (Enter key)

Switches 2/3: Invalid in this project

Refer to the fiigure below (right) for the button numbering.

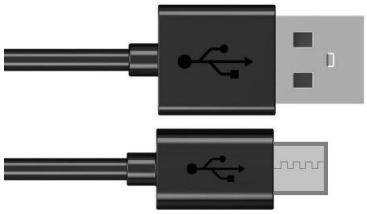
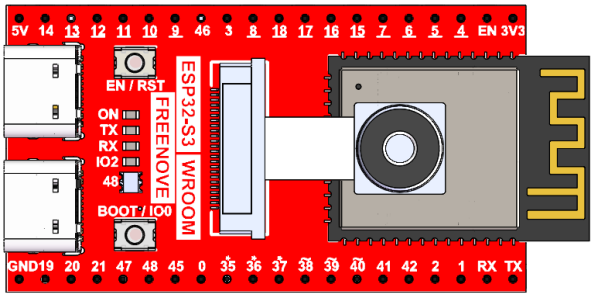


If you have any concerns, please feel free to contact us via support@freenove.com

Chapter 16 LVGL Music

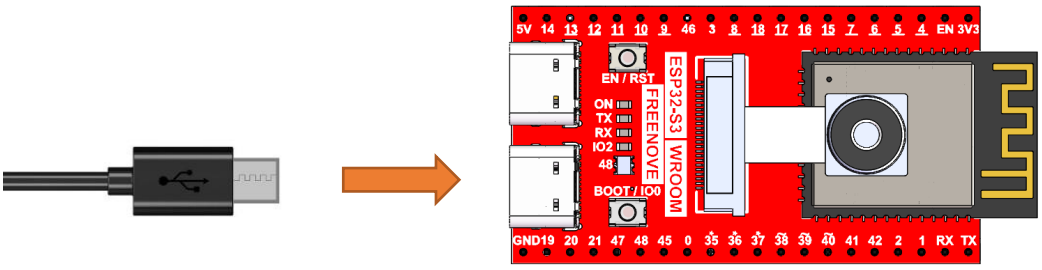
Project 16.1 LVGL Music

Component List

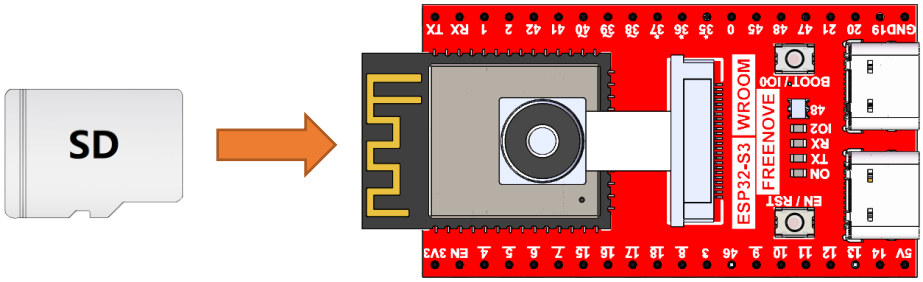
SD card x1 	USB cable x1 
Freenove Media Kit for ESP32-S3 x1 	Card reader x1 (random color) 

Circuit

Connect Freenove Media Kit for ESP32-S3 to your computer using the USB cable.



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



Sketch

Sketch_16_LVGL_Music

The following is the program code:

```
#include "music_ui.h"

#define SD_MMC_CMD 38  //Please do not modify it.
#define SD_MMC_CLK 39  //Please do not modify it.
#define SD_MMC_D0 40   //Please do not modify it.
#define I2S_BCLK 42    //Please do not modify it.
#define I2S_DOUT 1     //Please do not modify it.
#define I2S_LRC 41     //Please do not modify it.

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

void setup() {
  /* Prepare for possible serial debug */
  Serial.begin(115200);  // Initialize serial communication at 115200 baud rate

  /* Initialize SD card */
  sdmmc_init(SD_MMC_CLK, SD_MMC_CMD, SD_MMC_D0);
  audio_output_init(I2S_BCLK, I2S_LRC, I2S_DOUT);

  /*** Initialize the screen ***/
  screen.init(TFT_DIRECTION);  // Initialize the display

  // Create a string to display LVGL version information
  String LVGL_Arduino = "Hello Arduino! ";
  LVGL_Arduino += String('V') + lv_version_major() + "." + lv_version_minor() + "." +
lv_version_patch();

  // Print LVGL version information to the serial monitor
  Serial.println(LVGL_Arduino);
  Serial.println("I am LVGL_Arduino");

  setup_scr_music(&guider_music_ui);
  lv_scr_load(guider_music_ui.music);

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

void loop() {
```

```
40    screen.routine(); /* Let the GUI do its work */  // Handle routine display tasks
41    delay(5);                                        // Add a small delay to prevent the loop
42  from running too fast
43  }
```

Include the required libraries.

```
1   #include "music_ui.h"
```

Define SD card and I2S pins.

```
3   #define SD_MMC_CMD 38  //Please do not modify it.
4   #define SD_MMC_CLK 39  //Please do not modify it.
5   #define SD_MMC_D0 40   //Please do not modify it.
6   #define I2S_BCLK 42    //Please do not modify it.
7   #define I2S_DOUT 1     //Please do not modify it.
8   #define I2S_LRC 41     //Please do not modify it.
```

Declare TFT screen object.

```
10  Display screen;
```

Initialize SD card.

```
16  /* Initialize SD card */
17  sdmmc_init(SD_MMC_CLK, SD_MMC_CMD, SD_MMC_D0);
18  audio_output_init(I2S_BCLK, I2S_LRC, I2S_DOUT);
```

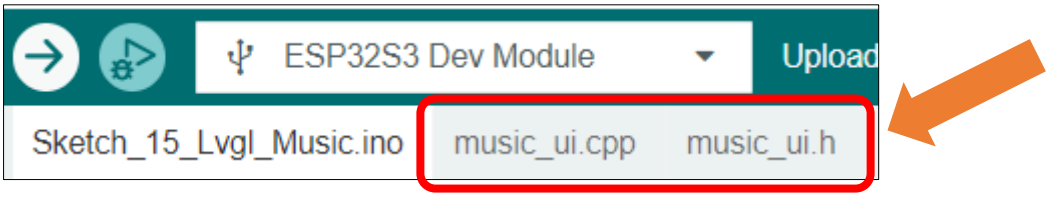
Initialize the TFT screen.

```
21  screen.init(TFT_DIRECTION);  // Initialize the display
```

Initialize the UI component for playing music.

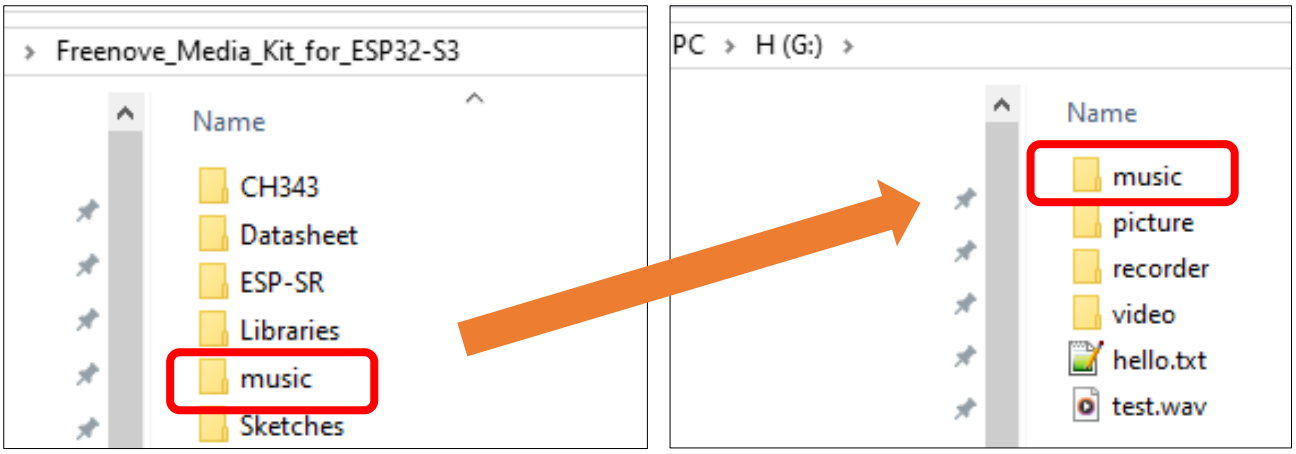
```
32  setup_scr_music(&guider_music_ui);
33  lv_scr_load(guider_music_ui.music);
```

If you are interesting in the implementation of functions, you can check them out here.

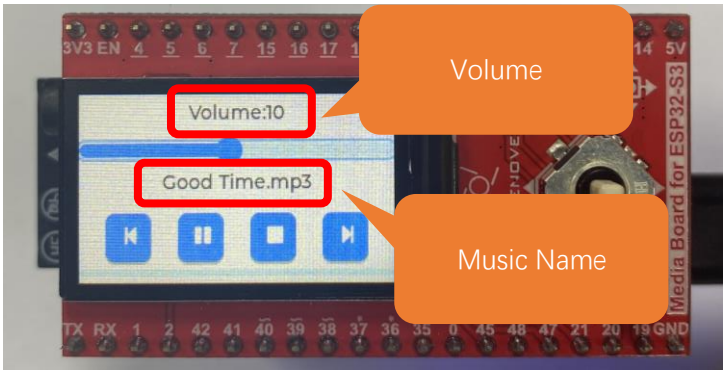


Before uploading the sketch, we need to copy the music files to the SD car. Remove the SD card from the ESP32S3 and insert it into a card reader. Connect the card reader to your computer.

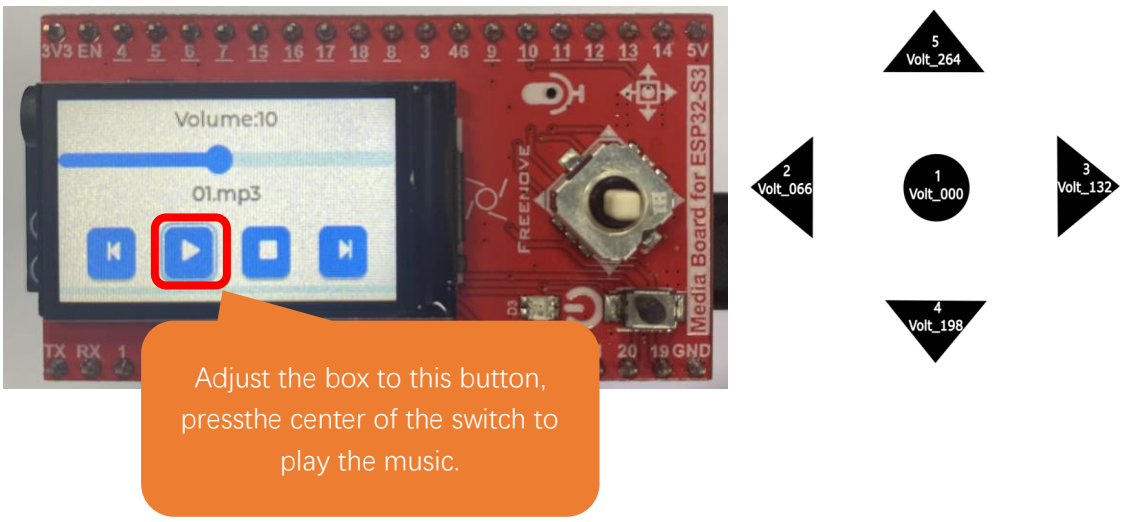
Copy the "**music**" folder from the "**Freenove_Media_Kit_for_ESP32-S3**" kit to the root directory of your SD card. You may add personal MP3 files to this folder. Ensure that the folder name is "**music**".



After uploading the code, you can see the screen displaying the following contents.



Operating different directions of the 5-way navigation switch will trigger theirorresponding functional events:
Switches 5 and 4 are used to switch between selection boxes.
Switches 2 and 3 are only effective within the volume adjustment option box and remain inactive in other cases.
Button 1 (Center) serves as the confirmation key.
(**Please refer to the diagram on the right for the numerical definitions of the buttons.)**



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

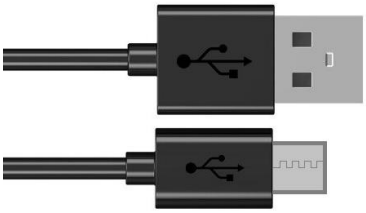
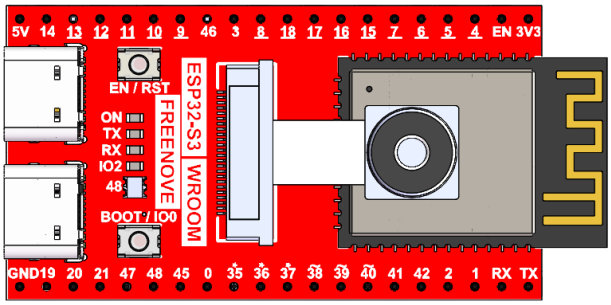
Chapter 17 LVGL Sound Recorder

In previous chapters, we learned about the recording functionality on the Freenove Media Kit for ESP32. In this chapter, we will explore how to integrate it with LVGL.

Project 17.1 LVGL Sound Recorder

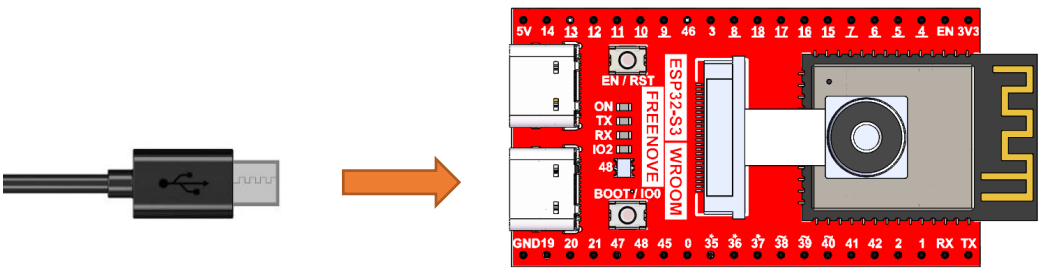
Capture image data using the camera module and display it on the TFT screen.

Component List

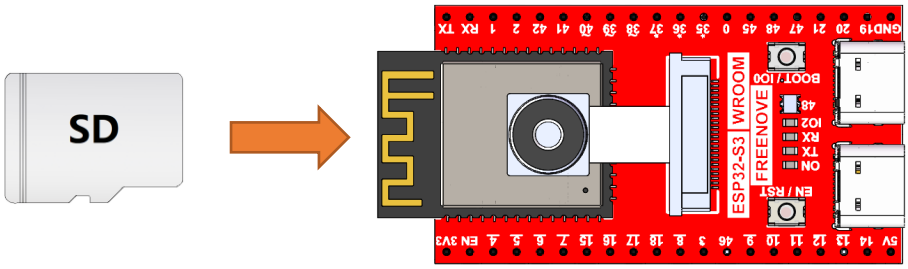
SD card x1 	USB cable x1 
Freenove Media Kit for ESP32-S3 x1 	

Circuit

Connect Freenove Media Kit for ESP32-S3 to your computer using the USB cable.



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



Sketch

Sketch_17_LVGL_Sound_Recorder

The following is the program code:

```
#include "recorder_ui.h"

#define SD_MMC_CMD 38          // Please do not modify it.
#define SD_MMC_CLK 39          // Please do not modify it.
#define SD_MMC_D0 40           // Please do not modify it.
#define AUDIO_INPUT_SCK 3      // Please do not modify it.
#define AUDIO_INPUT_WS 14      // Please do not modify it.
#define AUDIO_INPUT_DIN 46     // Please do not modify it.
#define AUDIO_OUTPUT_BCLK 42   // Please do not modify it.
#define AUDIO_OUTPUT_LRC 41    // Please do not modify it.
#define AUDIO_OUTPUT_DOUT 1    // Please do not modify it.

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

void setup() {
  /* Prepare for possible serial debug */
  Serial.begin(115200);
  while (!Serial) {
    delay(10);
  }

  audio_input_init(AUDIO_INPUT_SCK, AUDIO_INPUT_WS, AUDIO_INPUT_DIN);
  audio_output_init(AUDIO_OUTPUT_BCLK, AUDIO_OUTPUT_LRC, AUDIO_OUTPUT_DOUT);
  sdmmc_init(SD_MMC_CLK, SD_MMC_CMD, SD_MMC_D0);
  audio_output_set_volume(21);

  // Create recording directory if not exists
  create_dir(RECORDER_FOLDER);

  /*** Initialize the screen ***/
  screen.init(TFT_DIRECTION);  // Initialize the display
```

```
32
33    // Create a string to display LVGL version information
34    String LVGL_Arduino = "Hello Arduino! ";
35    LVGL_Arduino += String('V') + lv_version_major() + "." + lv_version_minor() + "." +
36  lv_version_patch();
37
38    // Print LVGL version information to the serial monitor
39    Serial.println(LVGL_Arduino);
40    Serial.println("I am LVGL_Arduino");
41
42    setup_scr_sound_recorder(&guider_recorder_ui);
43    lv_scr_load(guider_recorder_ui.recorder);
44
45    // Print setup completion message to the serial monitor
46    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
52  from running too fast
53  }
```

Include the required libraries.

```
1   #include "recorder_ui.h"
```

Define SD card, I2S and microphone pins.

```
3   #define SD_MMC_CMD 38          // Please do not modify it.
4   #define SD_MMC_CLK 39          // Please do not modify it.
5   #define SD_MMC_DO 40           // Please do not modify it.
6   #define AUDIO_INPUT_SCK 3      // Please do not modify it.
7   #define AUDIO_INPUT_WS 14      // Please do not modify it.
8   #define AUDIO_INPUT_DIN 46     // Please do not modify it.
9   #define AUDIO_OUTPUT_BCLK 42   // Please do not modify it.
10  #define AUDIO_OUTPUT_LRC 41    // Please do not modify it.
11  #define AUDIO_OUTPUT_DOUT 1    // Please do not modify it.
```

Declare TFT screen object.

```
13  Display screen;
```

Initialize the audio input, audio output and SD card interface, and set the output volume to level 21.

```
22  audio_input_init(AUDIO_INPUT_SCK, AUDIO_INPUT_WS, AUDIO_INPUT_DIN);
23  audio_output_init(AUDIO_OUTPUT_BCLK, AUDIO_OUTPUT_LRC, AUDIO_OUTPUT_DOUT);
24  sdmmc_init(SD_MMC_CLK, SD_MMC_CMD, SD_MMC_DO);
25  audio_output_set_volume(21);
```

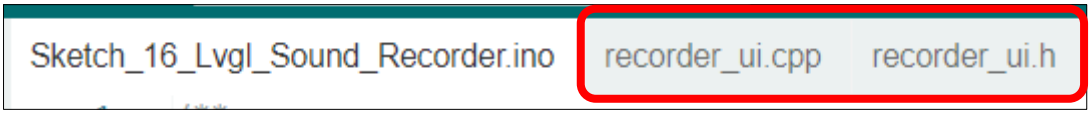
Initialize TFT screen

```
19  screen.init(TFT_DIRECTION);  // Initialize the display
```

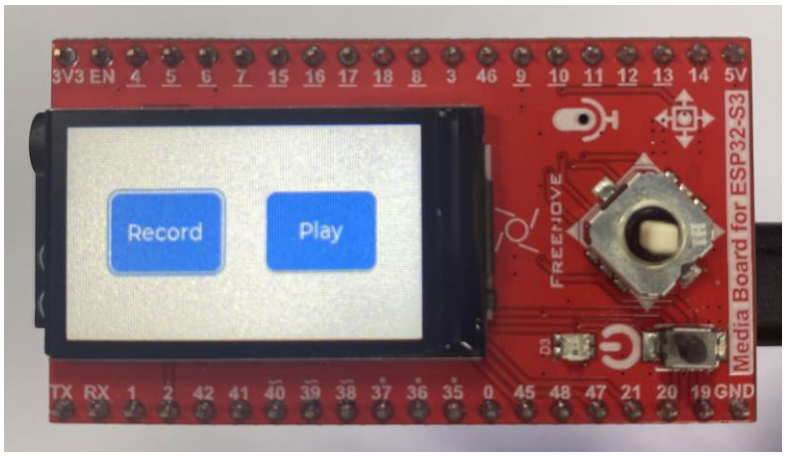
Initialize and load the UI component for sound recording.

```
setup_scr_sound_recorder(&guider_recorder_ui);
lv_scr_load(guider_recorder_ui.recorder);
```

If you are interesting in the implementation of functions, you can check them out here.



After uploading the sketch, you will see the following interface.



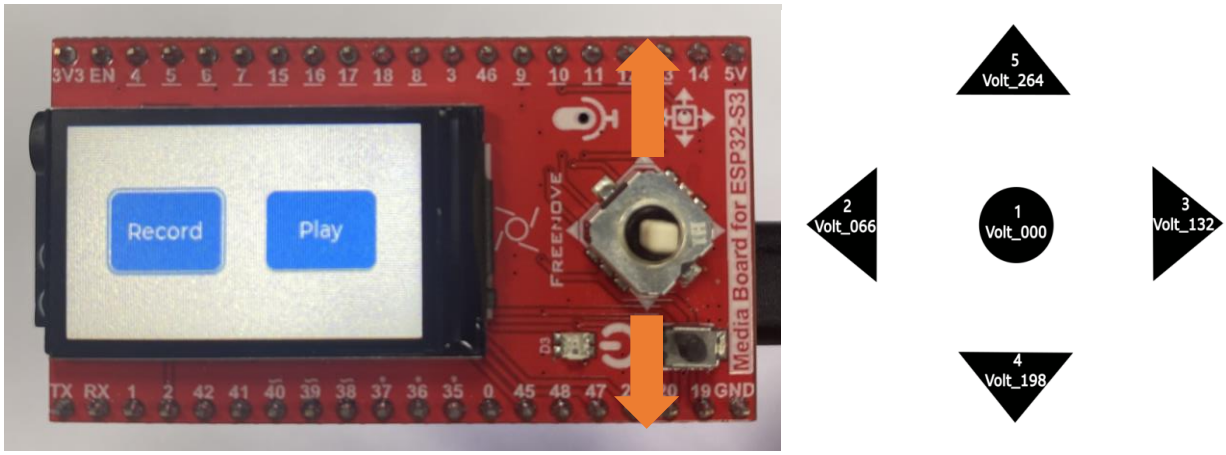
Pressing different directions on the 5-way navigation switch will trigger corresponding functions:

Switches 2 & 3 switch between recording and playback modes.

Button 1 serves as the confirmation key.

Switches 4 & 5 are disabled (no function).

Note: In recording mode, long-press Button 1 to start recording. The recording duration depends on how long Button 1 is held down.

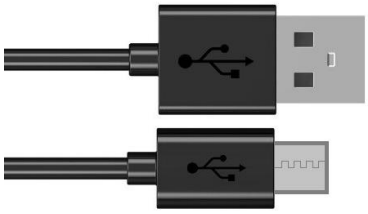
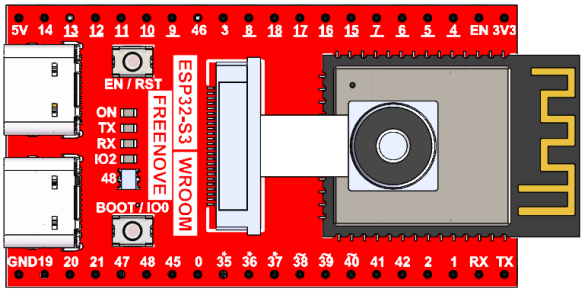


Chapter 18 LVGL Multifunctionality

In this chapter, we will combine the functionalities covered in previous sections.

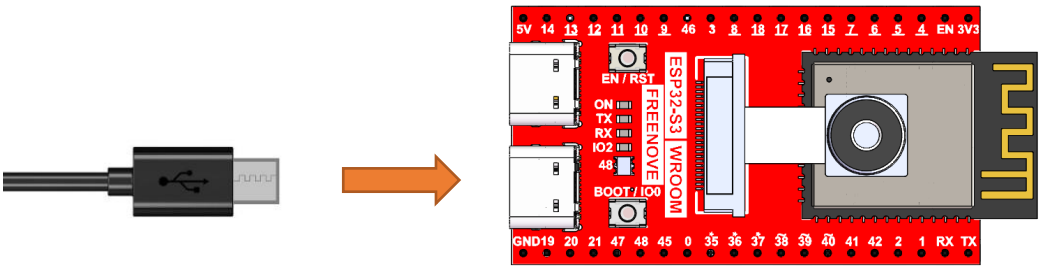
Project 18.1 LVGL Multifunctionality

Component List

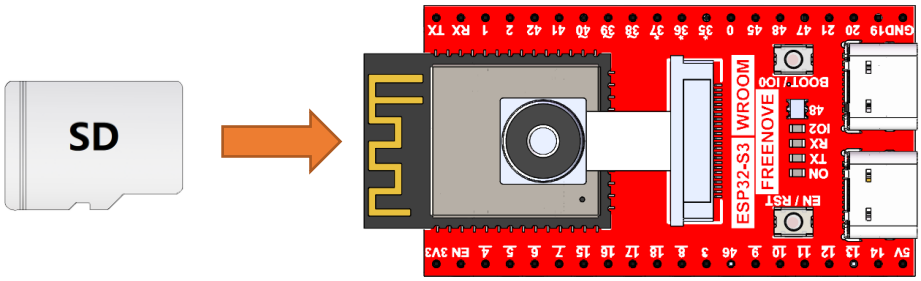
SD card x1 	USB cable x1 
Freenove Media Kit for ESP32-S3 x1 	Card reader x1 (random color) 

Circuit

Connect Freenove Media Kit for ESP32-S3 to your computer using the USB cable.



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



Sketch

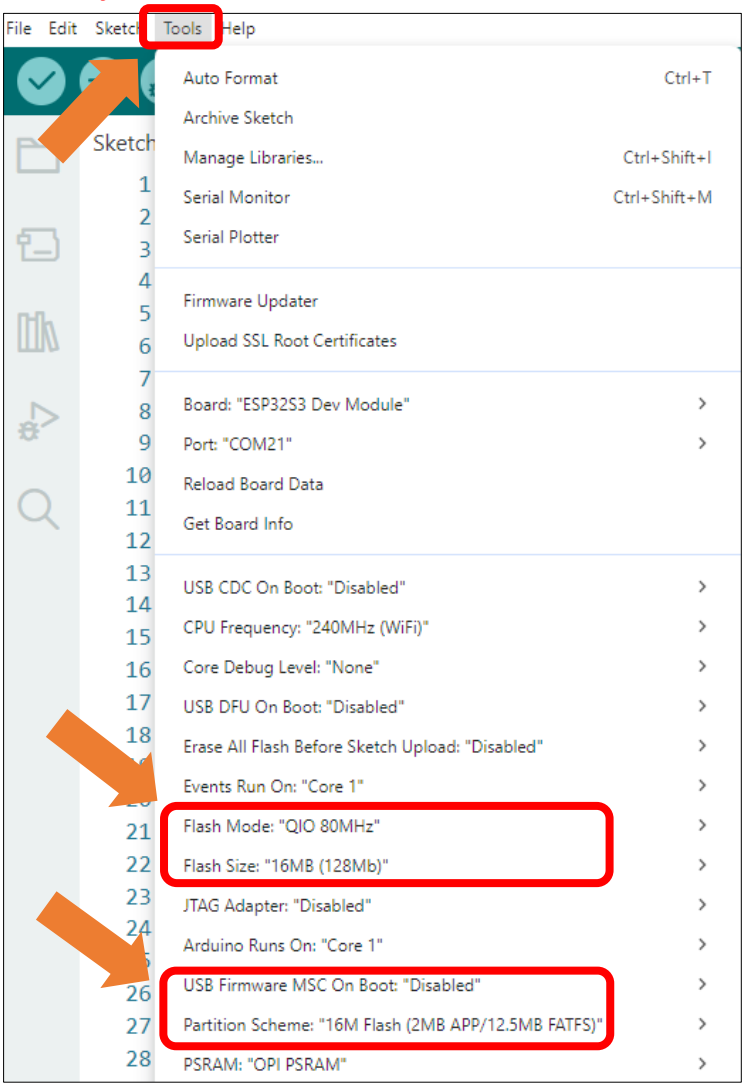
Sketch_17_LVGL_Multifunctionality

The following is the program code:

```
#include "public.h"

#define SD_MMC_CMD 38          //Please do not modify it.
#define SD_MMC_CLK 39          //Please do not modify it.
#define SD_MMC_D0 40           //Please do not modify it.
#define AUDIO_INPUT_SCK 3      //Please do not modify it.
#define AUDIO_INPUT_WS 14      //Please do not modify it.
#define AUDIO_INPUT_DIN 46     //Please do not modify it.
#define AUDIO_OUTPUT_BCLK 42   //Please do not modify it.
#define AUDIO_OUTPUT_LRC 41    //Please do not modify it.
#define AUDIO_OUTPUT_DOUT 1    //Please do not modify it.

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

void setup() {
  /* Prepare for possible serial debug */
  Serial.begin(115200);
  while (!Serial) {
    delay(10);
  }

  camera_init(0);
  audio_input_init(AUDIO_INPUT_SCK, AUDIO_INPUT_WS, AUDIO_INPUT_DIN);
  audio_output_init(AUDIO_OUTPUT_BCLK, AUDIO_OUTPUT_LRC, AUDIO_OUTPUT_DOUT);
  sdmmc_init(SD_MMC_CLK, SD_MMC_CMD, SD_MMC_D0);
  audio_output_set_volume(21);

  create_dir(CAMERA_FOLDER);
  create_dir(MUSIC_FOLDER);
  create_dir(RECORDER_FOLDER);
```

```
31
32    /*** Initialize the screen ***/
33    screen.init(TFT_DIRECTION);  // Initialize the display
34
35    // Create a string to display LVGL version information
36    String LVGL_Arduino = "Hello Arduino! ";
37    LVGL_Arduino += String('V') + lv_version_major() + "." + lv_version_minor() + "." +
38  lv_version_patch();
39
40    // Print LVGL version information to the serial monitor
41    Serial.println(LVGL_Arduino);
42    Serial.println("I am LVGL_Arduino");
43
44    setup_scr_main(&guider_main_ui);
45    lv_scr_load(guider_main_ui.main);
46
47    // Print setup completion message to the serial monitor
48    Serial.println("Setup done");
49  }
50
51  void loop() {
52    screen.routine(); /* Let the GUI do its work */  // Handle routine display tasks
53    delay(5);         // Add a small delay to prevent the loop from running too fast
54  }
```

It is necessary to change the settings in Arduino IDE before clicking the Uploading button, as shown below.
Caution: Incorrect settings will result in compilation error or uploading failure. To achieve desired result, please configure exactly the same as below.



After uploading the code, you can see the following interface on the screen.

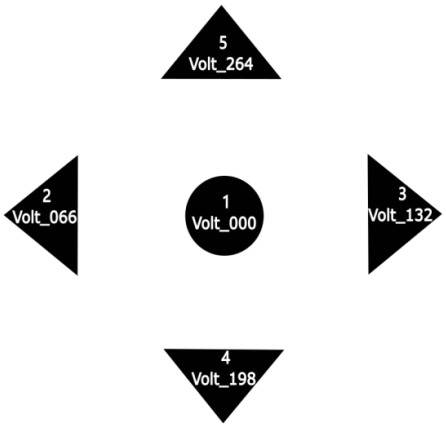
The 5-way navigation switch triggers different functions based on directional input:

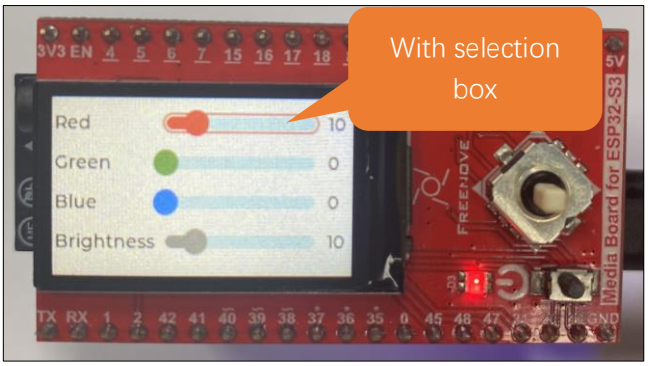
- Switches 2 & 3 – Cycle through available function
- Button 1 – Confirm selection (enters chosen mode)

Note: Operational logic within each function remains unchanged from previous implementations.

To exit current function and return to main menu:

- Deselect all components (no selection box is visible)
- Switch switches 4 or 5 to return to home interface





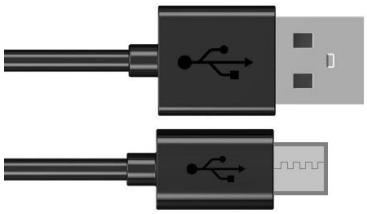
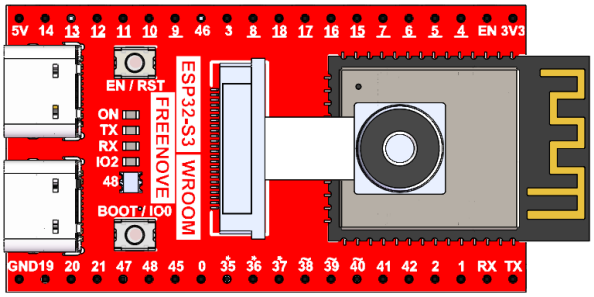
If you have any concerns, please feel free to contact us via support@freenove.com

Chapter 19 LVGL Multifunctionality

The functionality described in this chapter remains consistent with the previous section, but features a redesigned UI interface.

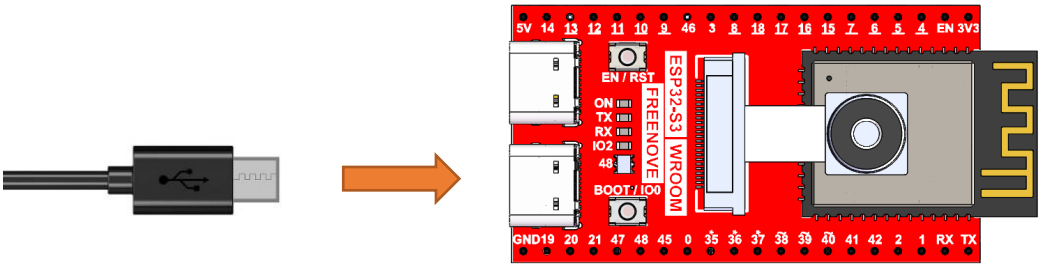
Project 19.1 LVGL Multifunctionality

Component List

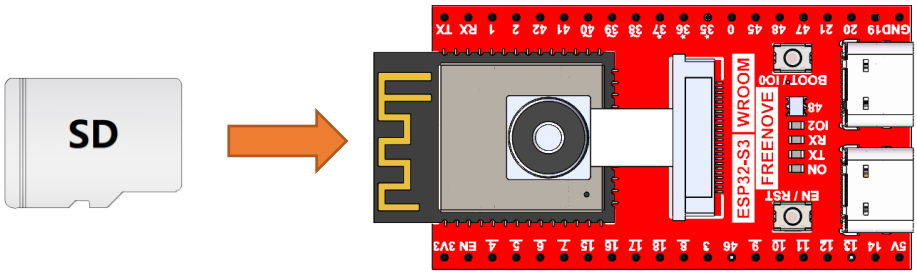
SD card x1 	USB cable x1 
Freenove Media Kit for ESP32-S3 x1 	Card reader x1 (random color) 

Circuit

Connect Freenove Media Kit for ESP32-S3 to your computer using the USB cable.



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



Sketch

Sketch_18_LVGL_Multifunctionality

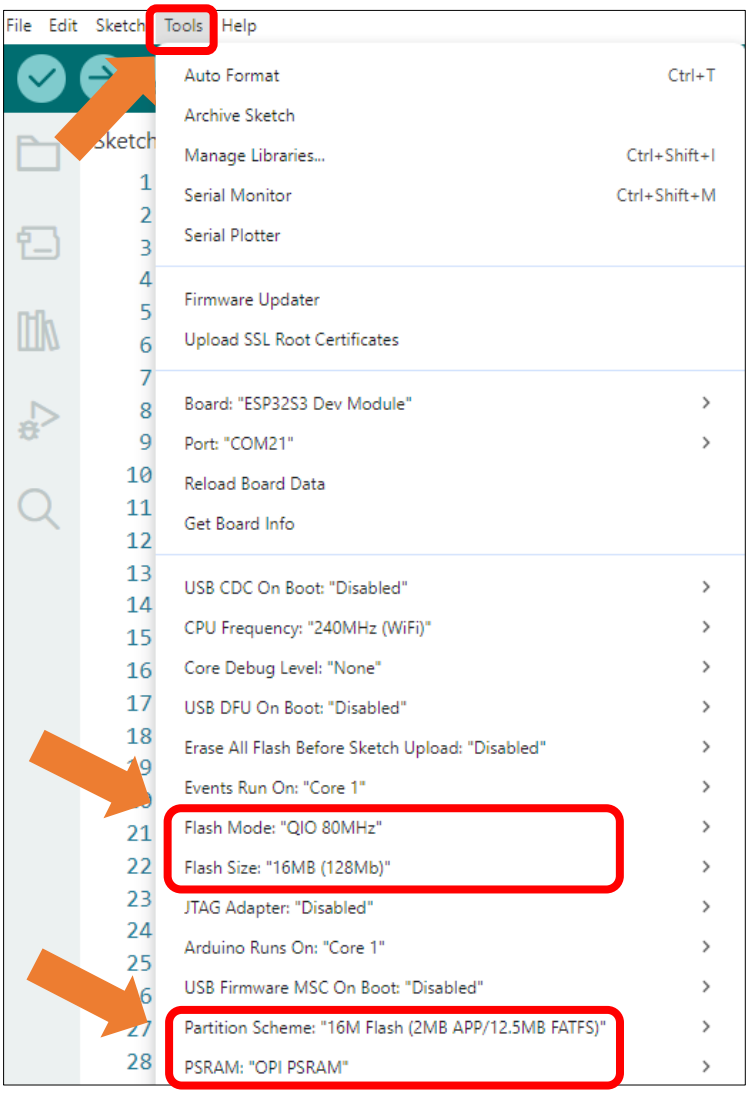
The following is the program code:

```
#include "public.h"

#define SD_MMC_CMD 38          //Please do not modify it.
#define SD_MMC_CLK 39          //Please do not modify it.
#define SD_MMC_D0 40           //Please do not modify it.
#define AUDIO_INPUT_SCK 3      //Please do not modify it.
#define AUDIO_INPUT_WS 14      //Please do not modify it.
#define AUDIO_INPUT_DIN 46     //Please do not modify it.
#define AUDIO_OUTPUT_BCLK 42   //Please do not modify it.
#define AUDIO_OUTPUT_LRC 41    //Please do not modify it.
#define AUDIO_OUTPUT_DOUT 1    //Please do not modify it.

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

void setup() {
  /* Prepare for possible serial debug */
  Serial.begin(115200);
  while (!Serial) {
    delay(10);
  }

  camera_init(0);
  audio_input_init(AUDIO_INPUT_SCK, AUDIO_INPUT_WS, AUDIO_INPUT_DIN);
  audio_output_init(AUDIO_OUTPUT_BCLK, AUDIO_OUTPUT_LRC, AUDIO_OUTPUT_DOUT);
  sdmmc_init(SD_MMC_CLK, SD_MMC_CMD, SD_MMC_D0);
  audio_output_set_volume(21);

  create_dir(CAMERA_FOLDER);
  create_dir(MUSIC_FOLDER);
  create_dir(RECORDER_FOLDER);
```

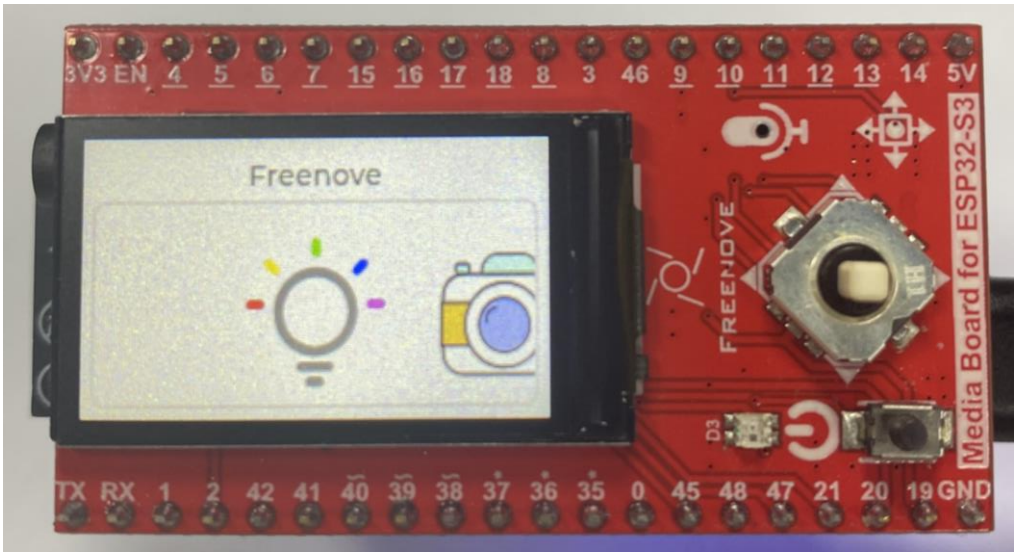
```
31
32    /*** Initialize the screen ***/
33    screen.init(TFT_DIRECTION);  // Initialize the display
34
35    // Create a string to display LVGL version information
36    String LVGL_Arduino = "Hello Arduino! ";
37    LVGL_Arduino += String('V') + lv_version_major() + "." + lv_version_minor() + "." +
38  lv_version_patch();
39
40    // Print LVGL version information to the serial monitor
41    Serial.println(LVGL_Arduino);
42    Serial.println("I am LVGL_Arduino");
43
44    setup_scr_main(&guider_main_ui);
45    lv_scr_load(guider_main_ui.main);
46
47    // Print setup completion message to the serial monitor
48    Serial.println("Setup done");
49  }
50
51  void loop() {
52    screen.routine(); /* Let the GUI do its work */  // Handle routine display tasks
53    delay(5);         // Add a small delay to prevent the loop from running too fast
54  }
```

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

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



After uploading the sketch, you'll see the following interface on the screen.



The 5-way navigation switch triggers different functions based on directional input:

Switches 2 & 3 – Cycle through available function

Button 1 – Confirm selection (enters chosen mode)

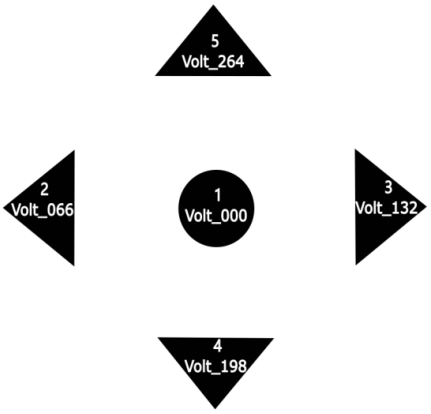
Note: Operational logic within each function remains unchanged from previous implementations.

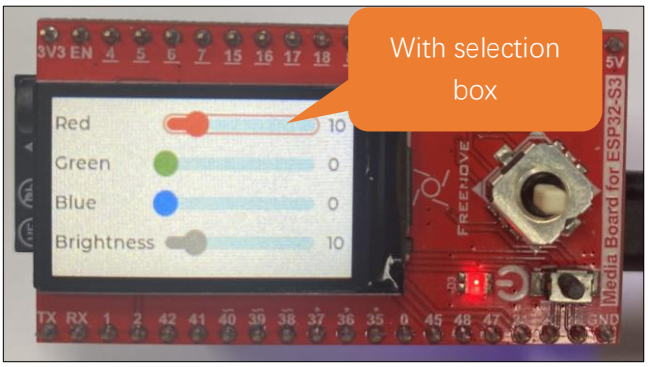
To exit current function and return to main menu

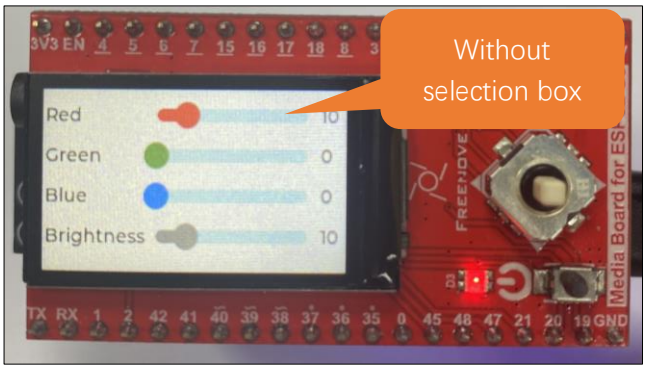
Deselect all components (no selection box is visible)

Switch switches 4 or 5 to return to home interface

(Refer to the figure below for button numbering)bu







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

AI Voice Assistant Based on XiaoZhi AI

This project applies the Media Kit to implement an AI voice assistant, which requires a certain level of programming proficiency as well as familiarity with ESP-IDF and open-source large models.

About the Project

This voice assistant project (https://github.com/Freenove/xiaozhi-esp32) is derived from the open-source project (https://github.com/78/xiaozhi-esp32). It enables the invocation of most mainstream large language models (LLMs) on embedded devices and achieves voice conversation functionality through multiple services, including Voice Activity Detection (VAD), Automatic Speech Recognition (ASR), Speech-to-Text (STT), Text-to-Speech (TTS), Memory Storage, and Intent Recognition. Freenove has adapted this project for its Media Kit product. This article will explain how to run the project on the Media Kit.

There are two ways to run this project – online or offline.

1. **Online**: Connected to the xiaozhi.me server, currently available for free trial to individual users.
2. **Offline**: All the aforementioned services (VAD, ASR, STT, TTS, Memory, Intent Recognition, etc.) must be deployed locally on a personal computer. The user experience depends entirely on the selected models and the performance of the local machine. The local server project (https://github.com/Freenove/xiaozhi-esp32-server) is derived from the open-source project (https://github.com/xinnan-tech/xiaozhi-esp32-server).

For users who prefer AI assistants, we recommend using the online version.
For developer-oriented users, you can try deploying the offline version to gain a deeper understanding of the various services required for an AI assistant. However, it's important to note that personal computers may struggle to run all these services simultaneously—especially the core LLM (Large Language Model) service—which could result in a poor AI assistant experience. Therefore, the offline version is primarily valuable for learning and research purposes.

Cautions

- **Project Copyright**:
  - Voice Assistant Project: Originally developed by "Xiage", this project was forked and adapted by Freenove for the Media Kit, released under MIT License.
  - Local Server Project: Originally created by "xinnan-tech", this project was similarly forked and adapted by Freenove for Media Kit integration, licensed under MIT License.
- **Supprted Countries and Regions:**
  - **Online Version**:
    Service availability is determined by xiaozhi.me server coverage, which may exclude certain regions. For current supported areas, please refer to: https://xiaozhi.me/login?redirect=/console/agents;
    User experience is directly affected by server connectivity quality. Poor network conditions to xiaozhi.me servers may degrade performance.

- **Offline Version**:
  Fully location-independent, with deployment possible in all countries and regions without geographical restrictions.

- **Supported Languages:**
  - Online Version: Currently supports Mandarin Chinese, Cantonese, English, Japanese, Korean, and others. If you do not speak these languages, you may not be able to communicate effectively with XiaoZhi AI.
  - Offline Version: Depending on the ASR model you deploy. The default FunASR model only supports Mandarin Chinese, Cantonese Chinese, English, Japanese and Korean.
- **Pricing:**
  - Online Version: Currently, xiaozhi.me provides free services, but we cannot guarantee that the online server will remain free indefinitely.
  - Offline Version: Among the sub-services mentioned, some are paid while others are free—your choice determines the cost.
- **Seeking Help:**
  If you have followed the tutorial and still encounter issues, please contact us at **support@freenove.com**
  Note: Since the online service is provided by xiaozhi.me, if xiaozhi.me discontinues its service, we will also remove related documentation, tutorials, and code.

Disclaimer

This implementation is an adaptation of the open-source project available at https://github.com/78/xiaozhi-esp32, provided for third-party learning and AI functionality testing purposes, without any promotion or support for commercial applications. This tutorial is intended solely for personal learning and development by technology enthusiasts.

Notes:

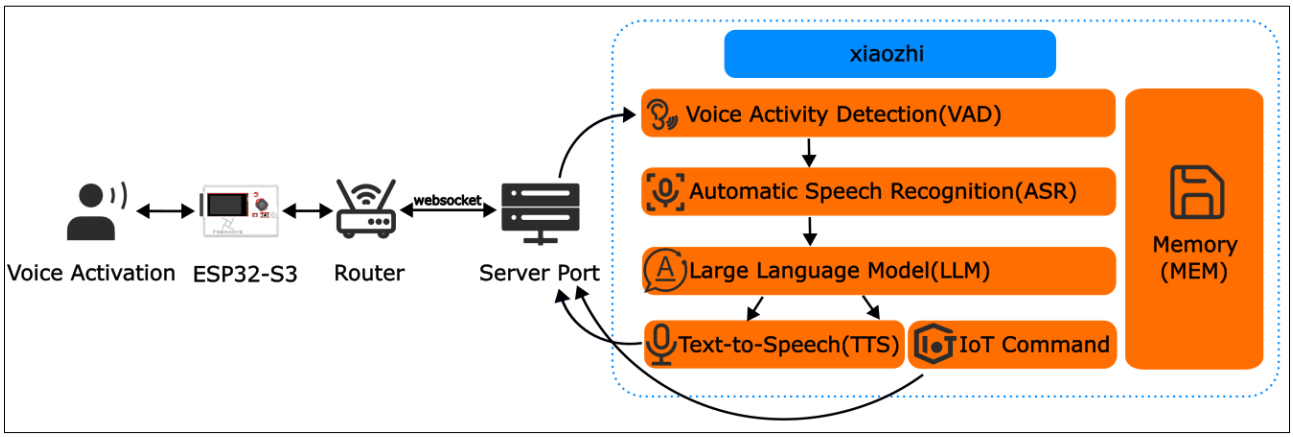
1, As this is a third-party open-source project, if you encounter issues during your learning process, please submit an issue to the original repository: https://github.com/78/xiaozhi-esp32/issues

2, Currently, XiaoZhi AI only supports Mandarin Chinese, Cantonese, English, Korean, and Japanese for speech recognition. Other languages are not yet supported.

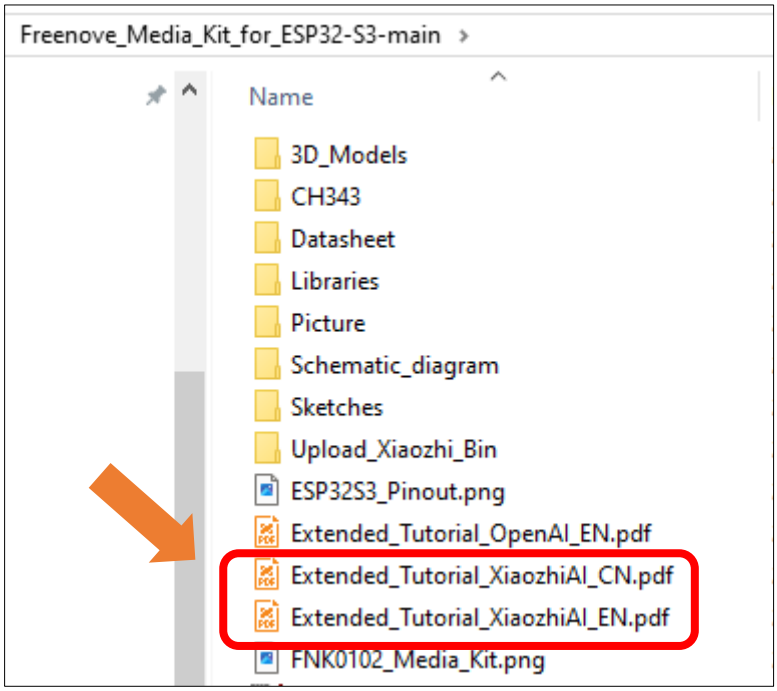
3, The XiaoZhi server interface currently supports English, Chinese, and Japanese only. Additionally, mobile registration is only available for users in the following countries (see the table below). Users from other countries cannot register yet.

Mainland China +86	Germany +49	Philippines +63	Pakistan +92
Hong Kong +852	Poland +48	New Zealand +64	Nigeria +234
Macau +853	Switzerland +41	Singapore +65	Bangladesh +880
Taiwan +886	Spain +34	Thailand +66	Saudi Arabia +966
United States/Canada +1	Denmark +45	Japan +81	United Arab Emirates +971
United Kingdom +44	Malaysia +60	South Korea +82	Brazil +55
France +33	Australia +61	Vietnam +84	Mexico +52
Italy +39	Indonesia +62	India +91	Chile +56
Argentina +54	Egypt +20	South Africa +27	Kenya +254
Tanzania +255	Kazakhstan +7		

In this project, the ESP32-S3 communicates with XiaoZhi AI server through WebSocket protocol for data exchange.



We have provided the tutorial for using the XiaoZhi AI on Freenove Media Kit for ESP32-S3, you can click here to download. The location of the tutorial is as shown below.



For your convenience, XiaoZhi AI's tutorial is available in both Chinese and English versions. You can choose to read either based on your preference:
Files with "EN" suffix -> English version
Files with "CN" suffix -> Chinese version

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

AI Voice Assistant Based on OpenAI Realtime Model

This project applies the Media Kit to develop an AI voice assistant for conversations with OpenAI. It requires basic programming skills and some familiarity with OpenAI.

About the Project

This voice assistant project (https://github.com/Freenove/openai-realtime-embedded) is based on OpenAI's open-source project **openai-realtime-embedded** (https://github.com/openai/openai-realtime-embedded). It enables embedded devices to call OpenAI's Realtime Model APIs, such as GPT-4o Realtime and GPT-4o Mini Realtime, allowing you to build your own AI voice assistant.

Freenove has adapted this project for its Media Kit product. This article will guide you on how to run it on the Media Kit.

Cautions

- **Project Copyright**: The original author of this project is OpenAI. Freenove forked and adapted it for Media Kit, and the project is licensed under the MIT License.
- **Supported Countries/Regions**: This project uses OpenAI's GPT-4o RealTime API, which is not available in all countries/regions. Please check OpenAI's supported countries list: https://platform.openai.com/docs/supported-countries.
  If you cannot access this link, OpenAI's services are likely unavailable in your location.
- **Supported Languages**: OpenAI supports many languages but has no official list. For reference, see: https://platform.openai.com/docs/api-reference/realtime-sessions/create
  https://en.wikipedia.org/wiki/List_of_ISO_639_language_codes)
- **Pricing**: The GPT-4o RealTime API is a paid service, and you must purchase credits from OpenAI to use it.
- **Seeking Help:** If you have followed the tutorial but still encounter issues, contact us at support@freenove.com
  Note: Since both the project and API are provided by OpenAI, if OpenAI discontinues them, we will also remove related documentation, tutorials, and code.

About OpenAI

OpenAI is a leading artificial intelligence research and deployment company committed to ensuring that artificial general intelligence (AGI) benefits all of humanity. Guided by principles of openness, collaboration, and safety, the organization drives the development and practical application of cutting-edge models. These include the renowned GPT series of language models, the DALL·E image generation model, as well as speech recognition and synthesis tools like Whisper.

OpenAI not only provides powerful AI models but also offers API services that enable developers to easily integrate these models into their own products. Recently, OpenAI launched its new Realtime API, which significantly enhances conversational naturalness through **direct streaming of audio input and output**, with automatic interruption handling capabilities.

However, it's important to note that:

1. **OpenAI currently does not offer free services**
2. The Realtime API currently only supports:
   GPT-4o and GPT-4o mini models
   The latest transcription models: GPT-4o Transcribe and GPT-4o mini Transcribe

For more information about Realtime API, please refer to Realtime API - OpenAI API

Openai-realtime-embedded Disclaimer

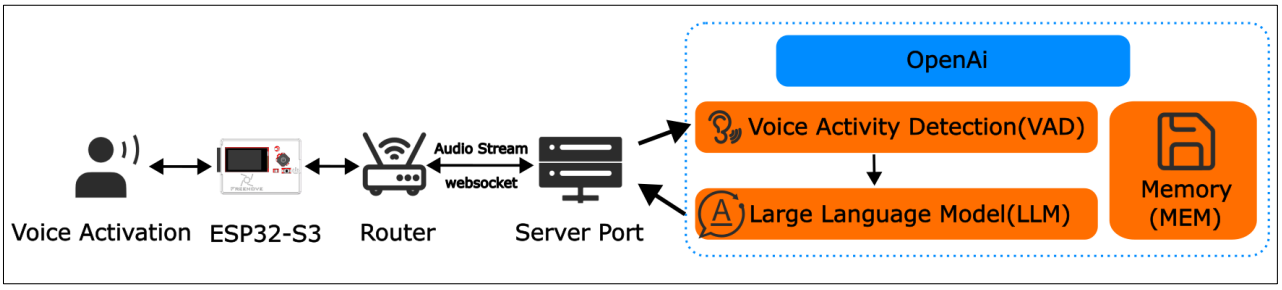
This implementation is an adaptation of the open-source project available at https://github.com/openai/openai-realtime-embedded, provided for third-party learning and AI functionality testing purposes, without any promotion or support for commercial applications. This tutorial is intended solely for personal learning and development by technology enthusiasts.

Notes:

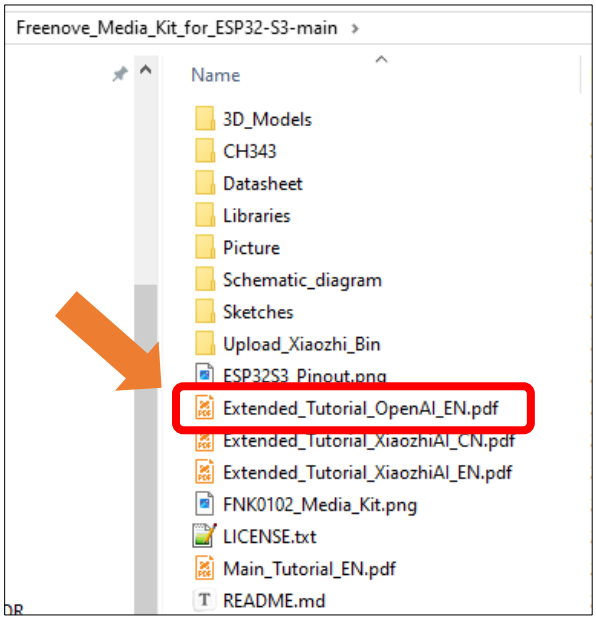
1. As this is a third-party open-source project, if you encounter issues during your learning process, please submit an issue to the original repository https://github.com/openai/openai-realtime-embedded/issues
2. The OpenAI API is a paid service. You must subscribe and enable billing to access any API functionality. Without an active paid account, all OpenAI API features will be unavailable.
3. To use OpenAI services, you must apply for your own API key directly from OpenAI. We do not provide or share any API key information.
   For details about OpenAI API access, please visit:
   https://platform.openai.com/docs/api-reference/introduction

Please see the following flowchart.

- The ESP32-S3 executes the openai-realtime-embedded source code, establishes a WiFi connection, and streams audio data to OpenAI's servers.
- Upon receiving the audio stream, OpenAI's servers process the data, generate text responses, and synthesize corresponding audio, which is then transmitted back to the ESP32-S3 via WiFi.
- In this project, the ESP32-S3 communicates with OpenAI's servers using the WebSocket protocol.



We have provided the tutorial for using the OpenAI on Freenove Media Kit for ESP32-S3, you can click here to download. The location of the tutorial is as shown below.



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

What's Next?

Thanks for your reading.

This book is all over here. If you find any mistakes, missions or you have other ideas and questions about contents of this book or the kit and ect., please feel free to contact us, and we will check and correct it as soon as possible.

After completing the projects in this book, you can further modify and customize the kit—for example, by purchasing and installing additional Freenove electronic modules, or enhancing the code to implement different functions. We will also continue adding new features and updating the code on our GitHub repository: https://github.com/freenove

If you want to learn more about Arduino, Raspberry Pi, smart cars, robots and orther interesting products in science and technology, please continue to focus on our website. We will continue to launch cost-effective, innovative and exciting products.
www.freenove.com

Thank you again for choosing Freenove products.