Welcome

Thank you for choosing Freenove products!

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, micro: bit and Raspberry Pi Pico W.
- 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

Copyright

FREENOVE™

Contents

Preface

Welcome to the Freenove Computer Case Kit for Raspberry Pi

This product is exclusively designed for the RPi 5, a popular single-board computer (SBC).Follow this tutorial, you caninstall a sleek, multi-functional computer case kit for your Raspberry Pi 5 (RPi 5).

The RPi 5 delivers highly powerful performance with a CPU frequency of up to 2.4GHz. However, this increased power consumption also leads to significant heat generation. Without an appropriate heatsink, thermal throttling may severely limit the Raspberry Pi 5's performance. Additionally, the removal of the 3.5mm audio jack in the RPi 5 has caused inconvenience for many users.

In this context, our case kit came into being. It not only addresses these issues, but also introduces advanced features.

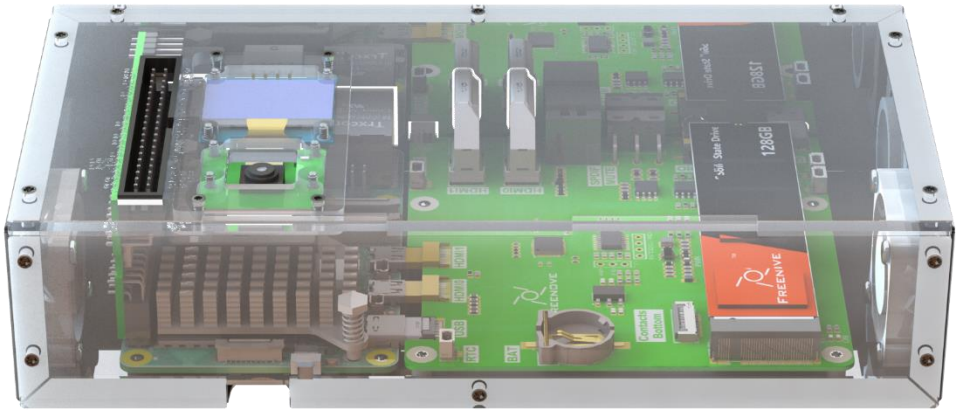
If you encounter any issues, feel free to contact us for prompt and free technical support.

support@freenove.com

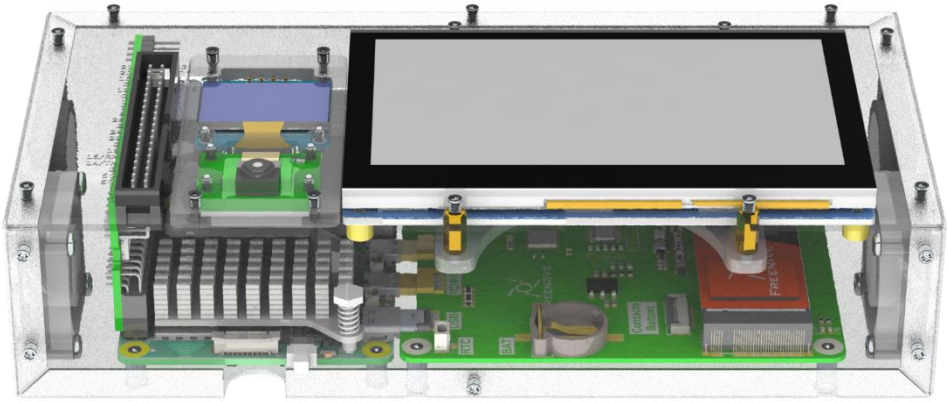
List

The product is currently available in two versions: FNK0100B and FNK0100K. The sole distinction between them lies in the addition of a 4.3-inch IPS screen (with a DSI interface) in the FNK0100K model. Below are their respective images:

FNK0100B:



FNK0100K:

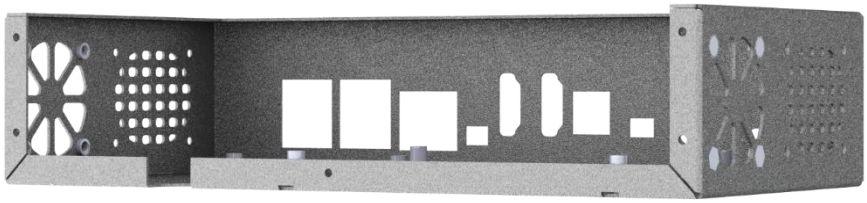


Before getting started, please check the part list. If any component is missing from your kit, please email our support team at

support@freenove.com

Metal Case

Metal case x 1



Machinery Parts

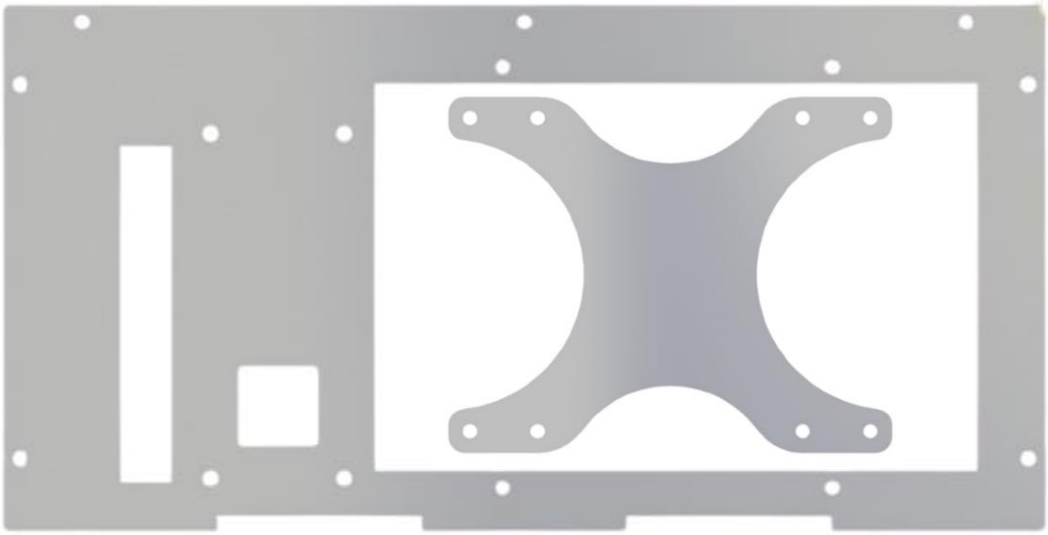
All fasteners come in a larger bag, please open it and check whether they are complete.

M2.5*3 Screw  x4 Freenove	M2.5*5 Brass Standoff  x2 Freenove	M3*12 Screw  x10 Freenove
M2*4 Countersunk Head Screw  x20 Freenove	M2*6 Countersunk Head Screw  x6 Freenove	(Only for FNK0100K) M2.5*9 Brass Standoff  x5 Freenove
(Only for FNK0100B) M2.5*5 Countersunk Head Screw  x25 Freenove	(Only for FNK0100K) M2.5*5 Countersunk Head Screw  x40 Freenove	

Acrylic Parts

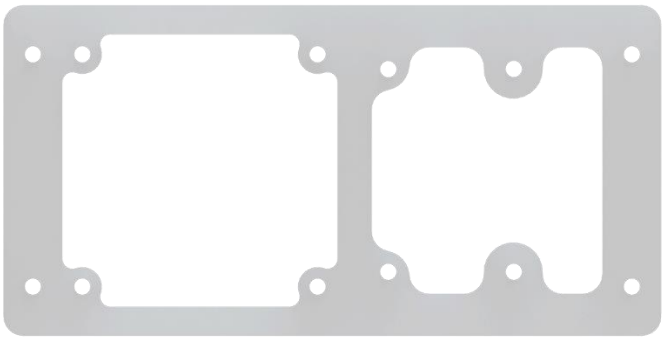
Top Cover x 1 (Only for FNK0100B)

Top Cover X 1, Support Plate for Screen X 1 (Only for FNK0100K)



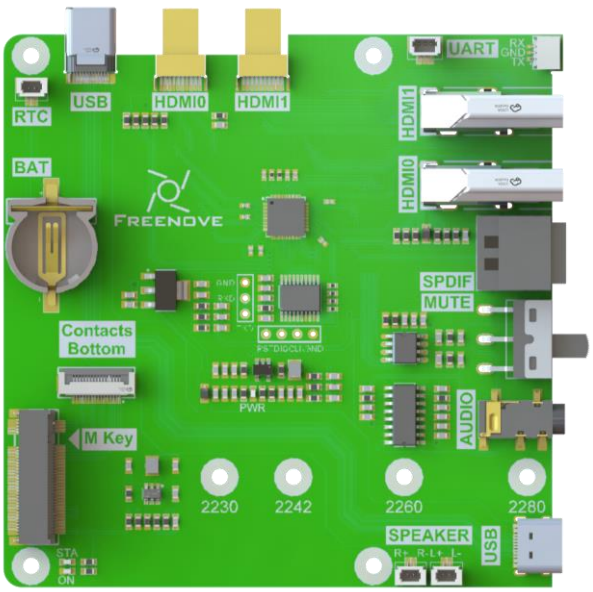
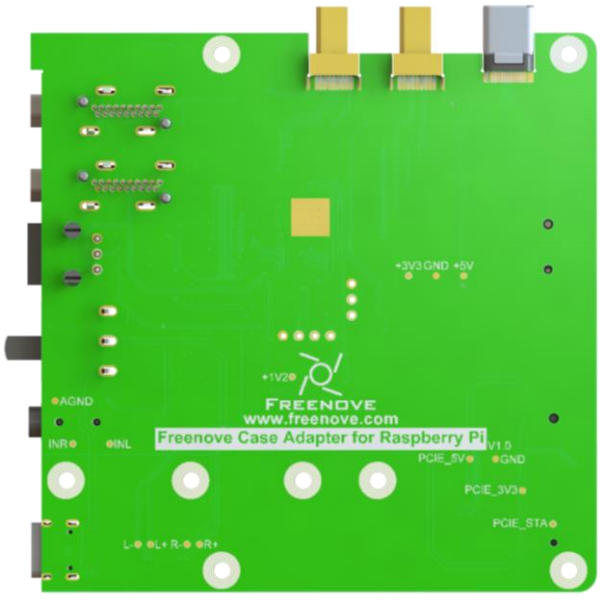
Side Plate x 1

Power Button Caps x 6

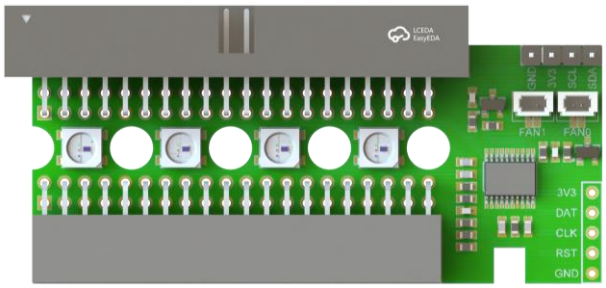
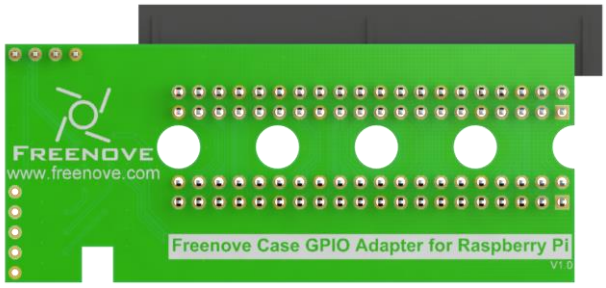
Note: The physical power button caps are very small, so be careful not to lose them.
Acrylic Part for OLED Screen and Camera x 1


Electronic Parts

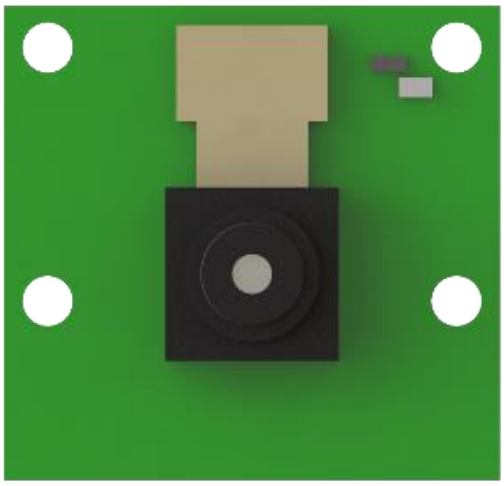
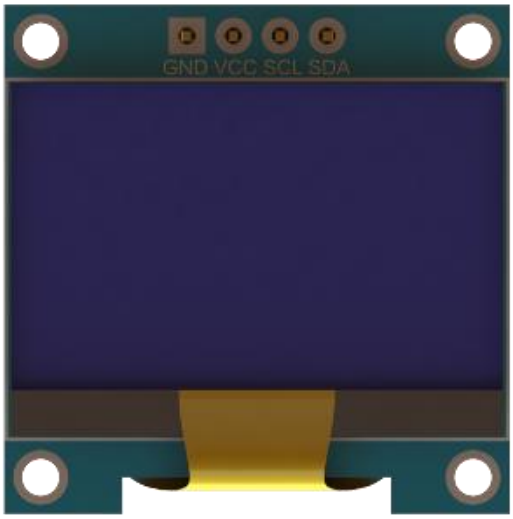
Freenove Case Adapter for Raspberry Pi

Top	Bottom
	

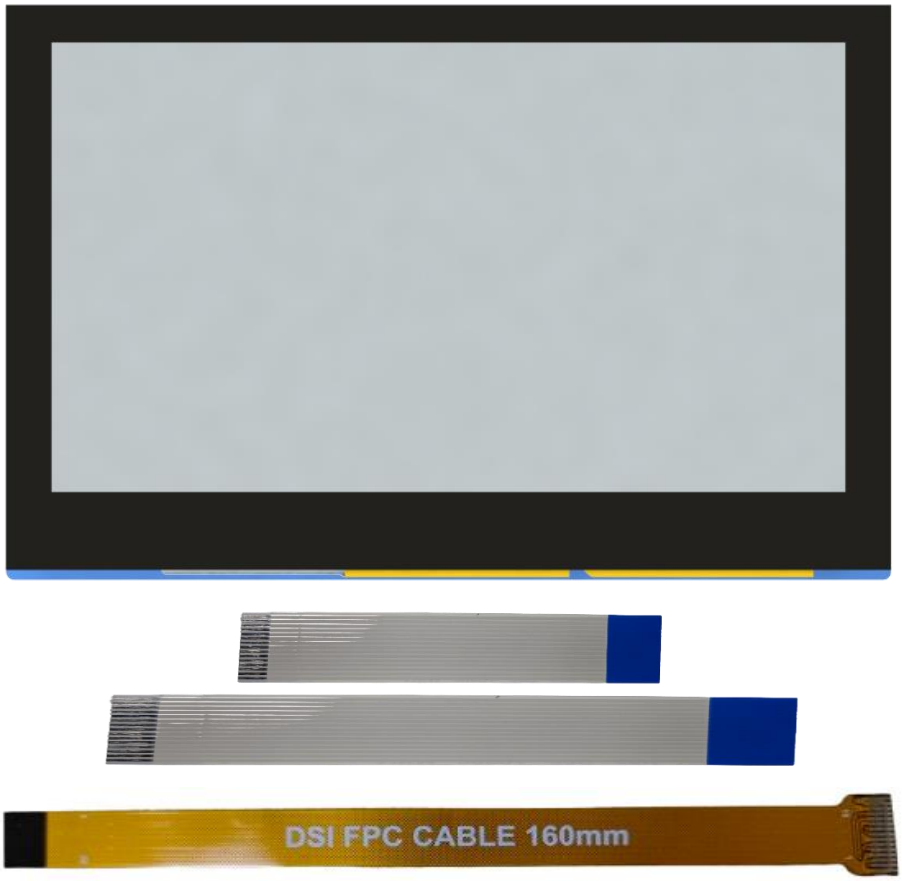
Freenove Case GPIO Adapter for Raspberry Pi

Top	Bottom
	

Electronic Modules

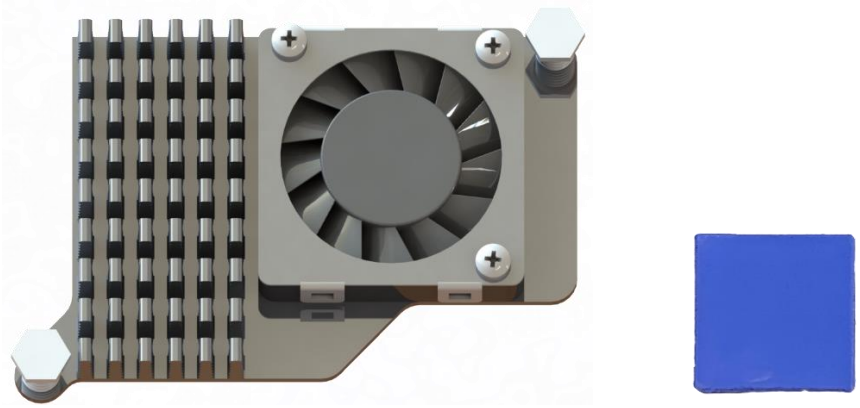
Camera Module x 1	OLED Module x 1
	
3010 Fan x 2	3040 Speaker x 2
	

4.3-inch Screen (with 3 FPC lines) x 1 (Only for FNK0100K)



Caution: Although the Raspberry Pi camera cable and screen cable may look identical in appearance, they are NOT interchangeable.

Pi5 CPU Active Cooler x 1, Heat Dissipation Silicone Sheet x3



NVMe SSD x 1

Wires

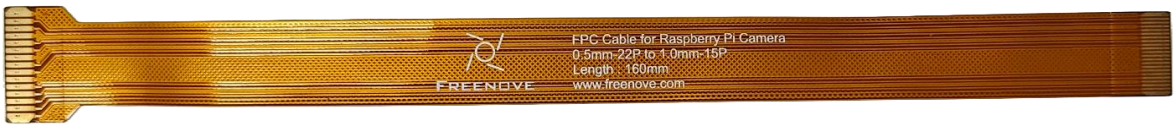
F-F Jumper Wire -4P-20cm x 1

NVMe SSD FPC Line-0.5mm-16P-15cm (same direction) x 1

SH-1.0mm-2P-5cm cable (same direction) x 1

SH-1.0mm-3P-5cm cable (same direction) x 1

Camera FPC line 0.5mm-22P to 1.0mm-15P-16cm line (same direction) x 1



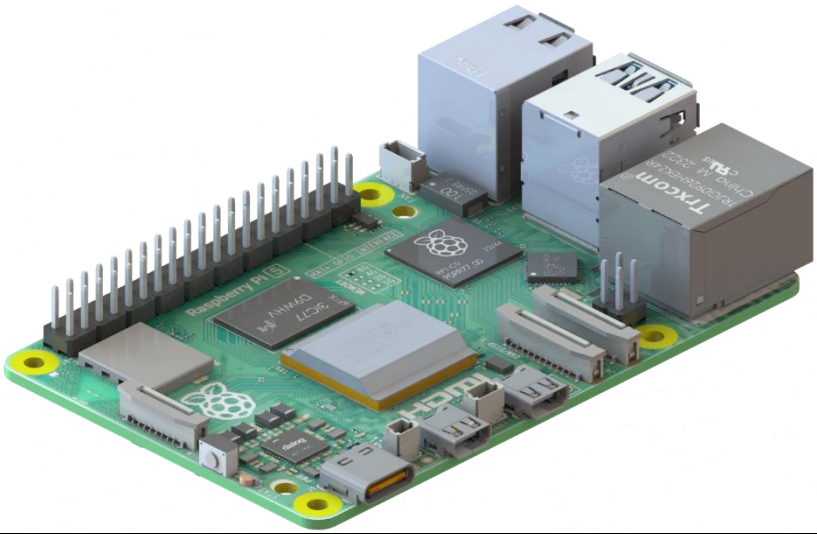
Caution: Although the Raspberry Pi camera cable and screen cable may look identical in appearance, they are NOT interchangeable.

Tools

Cross screwdriver (3mm) x 1

Required but NOT Contained Parts

Raspberry Pi 5 x 1



27W Power Adapter x 1 (or a power adapter compatible with Raspberry Pi official one that can output 5.1V/5A)

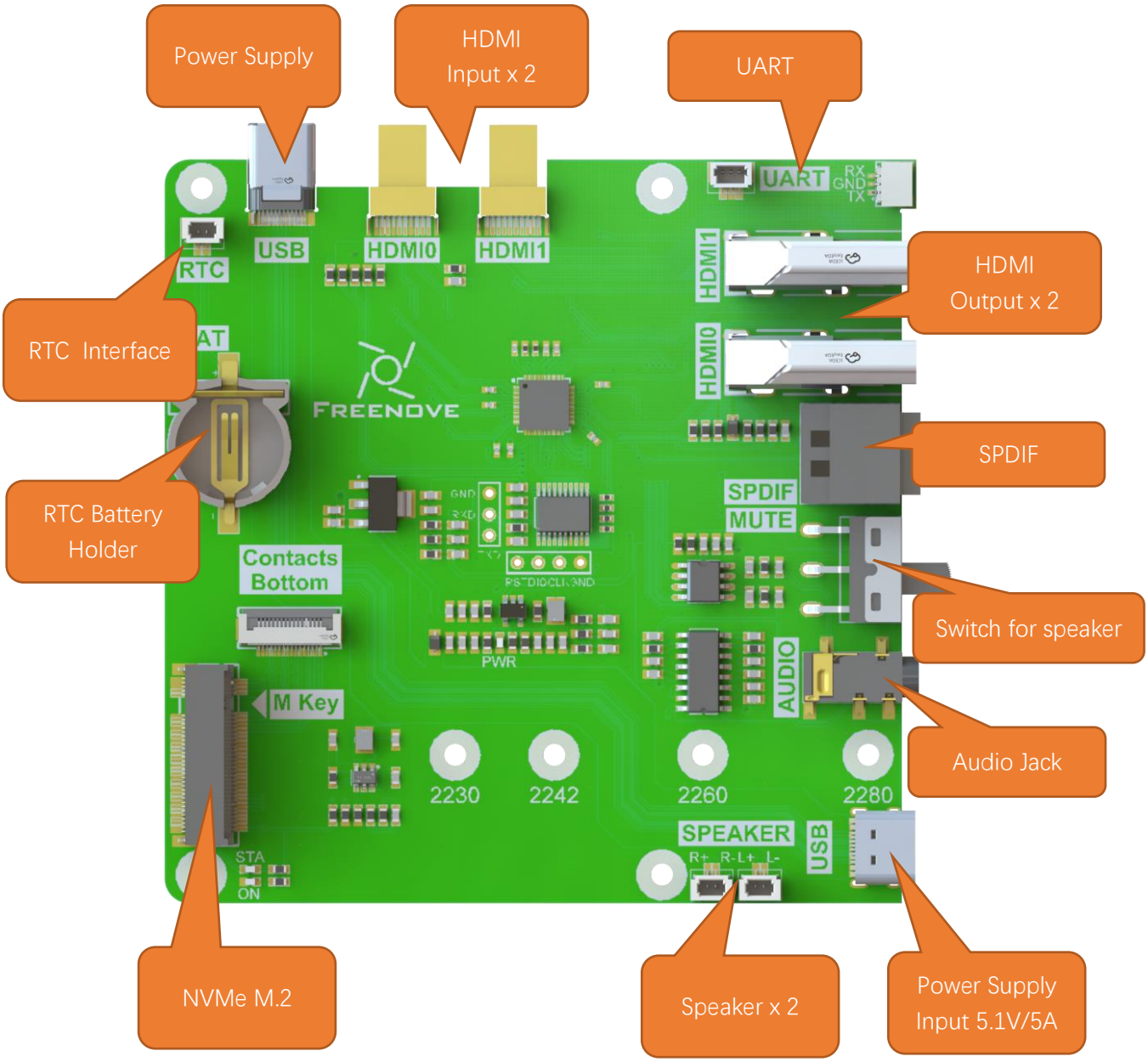
Micro SD Card (TF Card), Card Reader x 1

Chapter 1 Main Components of the Computer Case

In this chapter, we will mainly introduce the main components of this case and their functionalities.

1.1 Introduction to Freenove Case Adapter for Raspberry Pi

Hereinafter we will refer to this component as Case Adapter Board.



The Case Adapter Board is specifically designed for the Raspberry Pi 5 interface, with the goal of transforming the Rpi 5 into a fully functional home computer. This board incorporates a variety of essential interfaces, including a Type-C interface, two HDMI interfaces, an audio output interface, a speaker interface, an M.2 M-Key interface, and an RTC interface.

In the RPi 5 design, the traditional 3.5mm audio output interface has been removed, and audio signals are now solely output through HDMI. To address this, we have developed an audio separation circuit on the Case Adapter Board. This circuit enables audio to be output via the 3.5mm audio interface on the board. Additionally, we have designed an audio amplification circuit. With this circuit, you can control two speakers, each with an impedance of 4Ω and a power rating of 3W. If you prefer not to utilize the speakers within the computer case, you can easily switch them on or off using the MUTE button. For those who have high-fidelity demands, we also offer an SPDIF interface. This allows you to connect your own device for audio playback, ensuring superior sound quality.

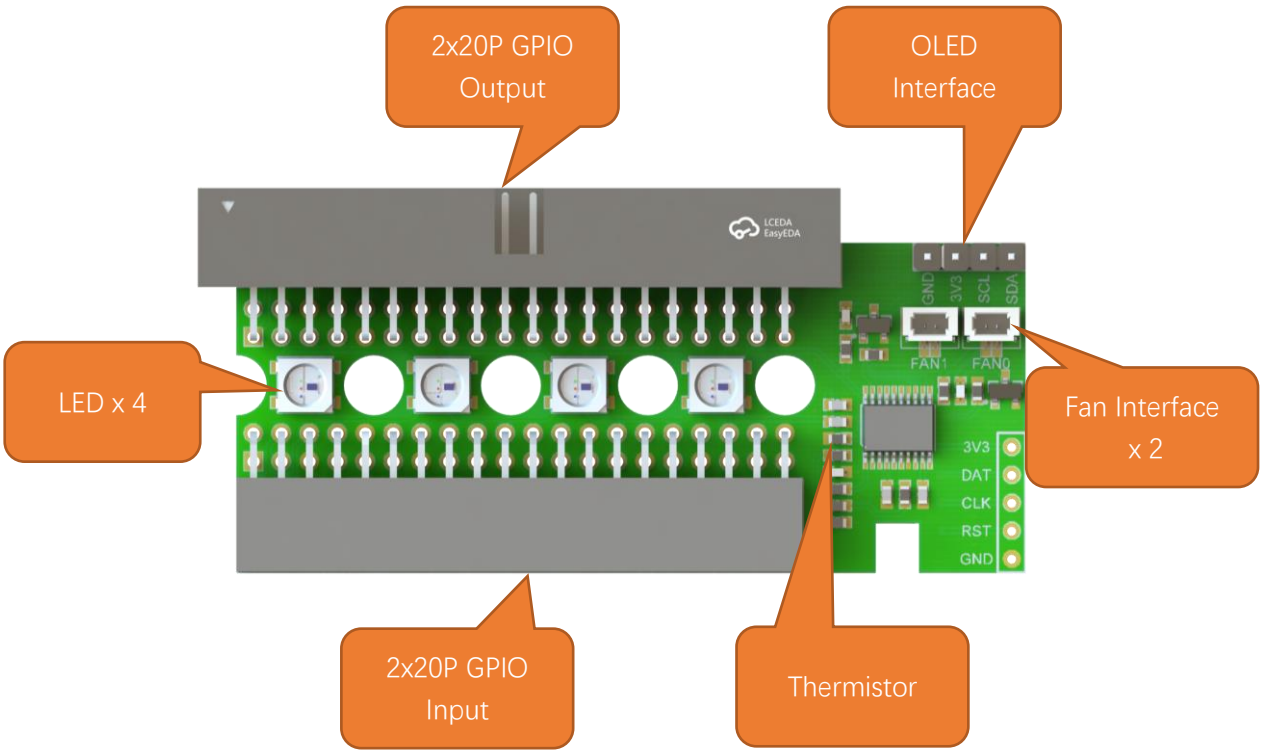
The Case Adapter Board also features a conversion function that transforms the RPi 5's PCIe interface into an M.2 M-Key interface. This conversion provides great convenience when it comes to using NVME SSDs. It supports four different sizes of SSDs, namely 2230, 2242, 2260, and 2280.

To enhance the accuracy of the Raspberry Pi's clock, it is advisable to add a battery to the RTC. For this purpose, we have installed a battery holder on the board. If you decide to add a battery, we recommend either purchasing a rechargeable lithium-manganese button-cell battery (with dimensions of 1220 and a voltage of 3V) or an official battery.
https://www.raspberrypi.com/documentation/computers/raspberry-pi.html#real-time-clock-rtc。

Due to the addition of the audio separation function, there is an increase in power consumption. Only the official 5.1V/5A power adapter (or compatible versions) can be used.

1.2 Introduction to Freenove Case GPIO Adapter for Raspberry Pi

Hereinafter we will refer to this component as GPIO Board.



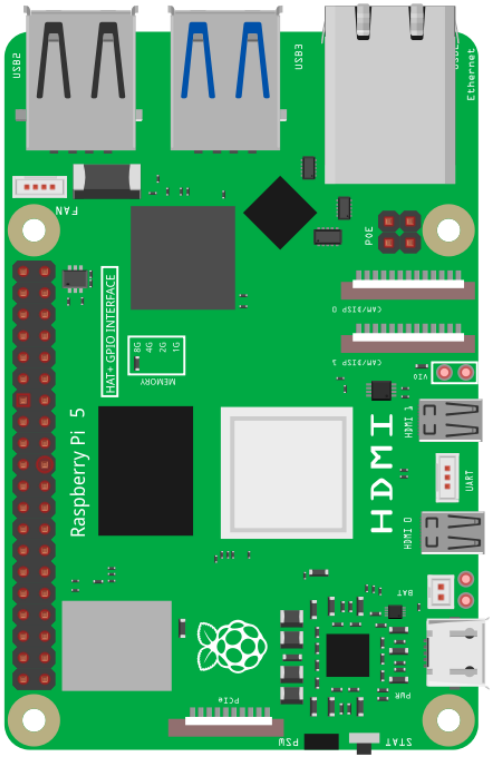
The GPIO board extends the GPIO pins of the Raspberry Pi 5 to the outside of the case. Additionally, it integrates color lights, case fans, and OLED interface, which can be controlled via the IIC of the RPi 5.

The following table shows the power consumption of the GPIO Board.

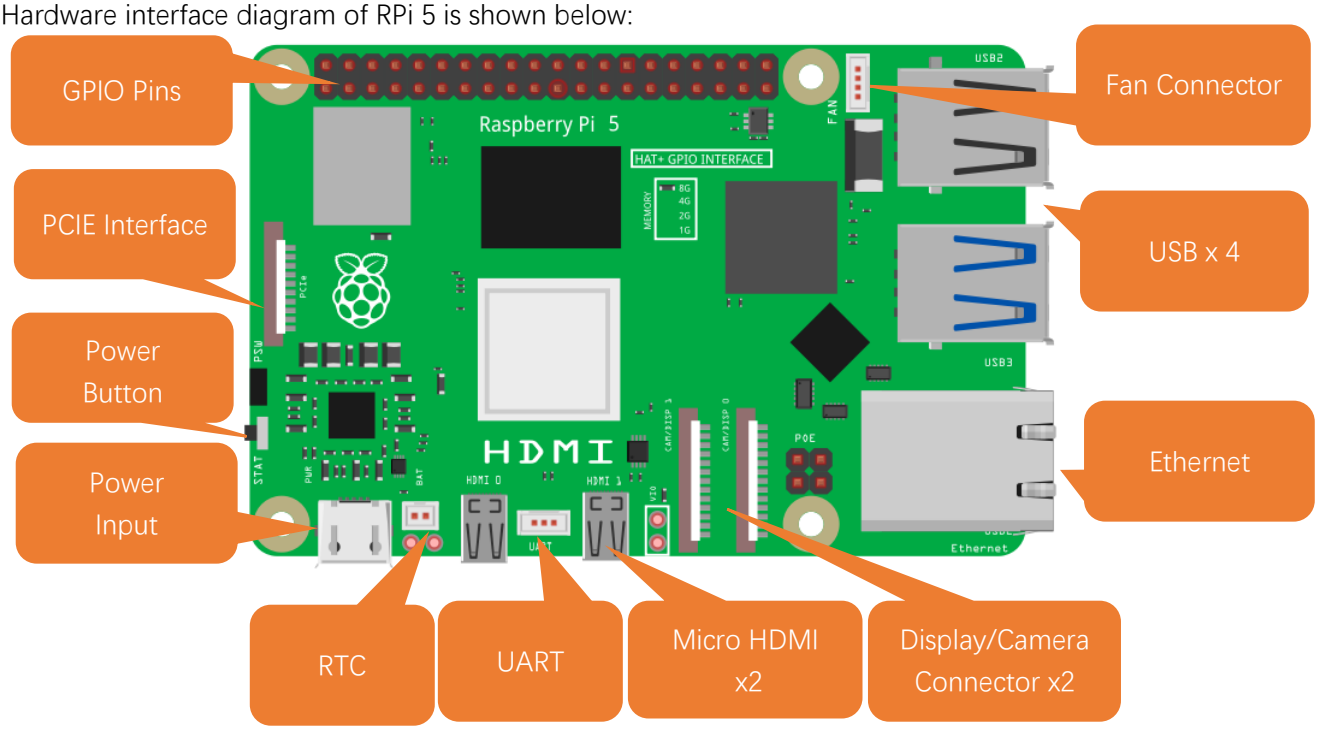
Function	Supply Voltage	Electric current
LED	5V	40*4mA
Fan	3.3V	130mA
OLED	3.3V	22mA

1.3 Introduction to Raspberry Pi 5 (RPi 5)

At the time of this writing, this product only supports RPi5. The following shows the physical and model figures of an RPi 5.

Practicality picture of Raspberry Pi 5:	Model diagram of Raspberry Pi 5:
	

Hardware interface diagram of RPi 5 is shown below:

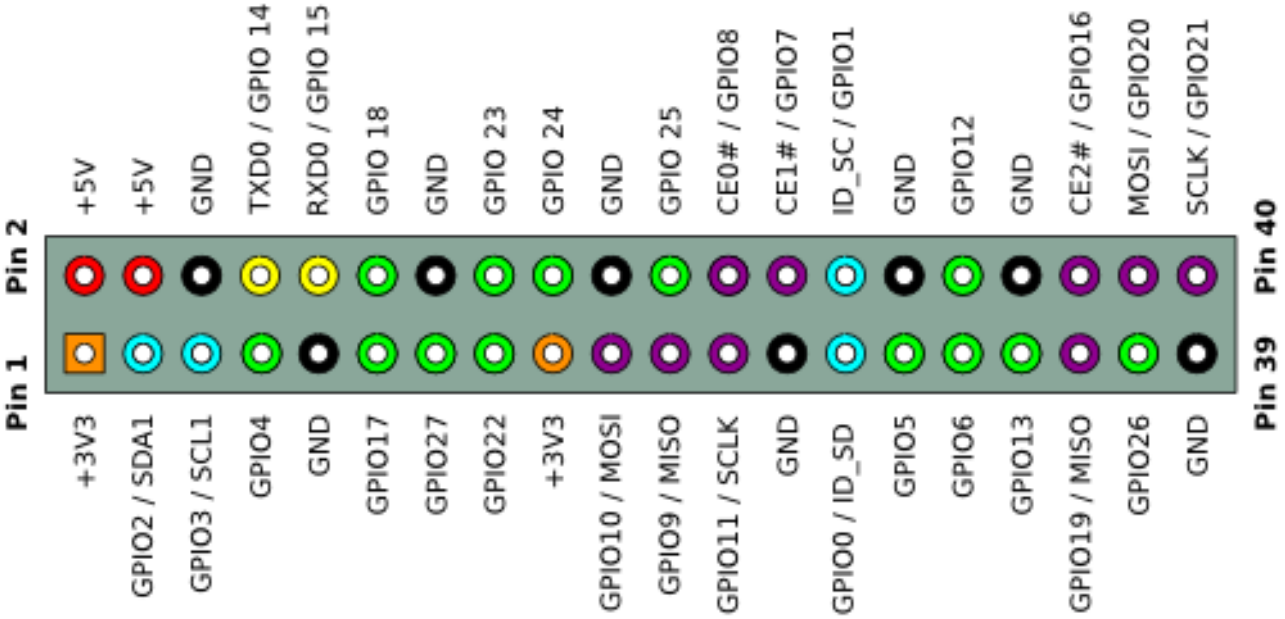


GPIO

GPIO: General purpose input/output. We will introduce the specific feature of the pins on the Raspberry Pi and how you can utilize them in all sorts of ways in your projects. Most RPi Module pins can be used as either an input or output, depending on your program and its functions. When programming the GPIO pins, there are three different ways to reference them: GPIO numbering, physical numbering, WiringPi GPIO Numbering.

BCM GPIO Numbering

The Raspberry Pi CPU uses Broadcom (BCM) processing chips BCM2835, BCM2836 or BCM2837. GPIO pin numbers are assigned by the processing chip manufacturer and are how the computer recognizes each pin. The pin numbers themselves do not make sense or have meaning, as they are only a form of identification. Since their numeric values and physical locations have no specific order, there is no way to remember them, so you will need to have a printed reference or a reference board that fits over the pins.
Each pin is defined as below:



For more details about pin definition of GPIO, please refer to https://pinout.xyz/

Power requirements of various versions of Raspberry Pi are shown in following table:

Product	Recommended PSU current capacity	Maximum total USB peripheral current draw	Typical bare-board active current consumption
Raspberry Pi Model A	700mA	500mA	200mA
Raspberry Pi Model B	1.2A	500mA	500mA
Raspberry Pi Model A+	700mA	500mA	180mA
Raspberry Pi Model B+	1.8A	600mA/1.2A (switchable)	330mA
Raspberry Pi 2 Model B	1.8A	600mA/1.2A (switchable)	350mA
Raspberry Pi 3 Model B	2.5A	1.2A	400mA
Raspberry Pi 3 Model A+	2.5A	Limited by PSU, board, and connector ratings only.	350mA
Raspberry Pi 3 Model B+	2.5A	1.2A	500mA
Raspberry Pi 4 Model B	3.0A	1.2A	600mA
Raspberry Pi 5 Model B	5.0A	1.6A (600mA if using a 3A power supply)	800mA
Raspberry Pi Zero W	1.2A	Limited by PSU, board, and connector ratings only.	150mA
Raspberry Pi Zero	1.2A	Limited by PSU, board, and connector ratings only	100mA

For more details, please refer to
https://www.raspberrypi.com/documentation/computers/raspberry-pi.html#power-supply

In this product, the Raspberry Pi 5 is used and it must be powered by a 5.1V/5A power supply. Insufficient power may cause various functions to operate abnormally, or even permanently damage your Raspberry Pi 5. Therefore, we strongly recommend using a 5.1V/5A power supply to ensure optimal performance and avoid potential hardware failure

Chapter 2 Assembly

If you need any help, please feel free to contact us at support@freenove.com

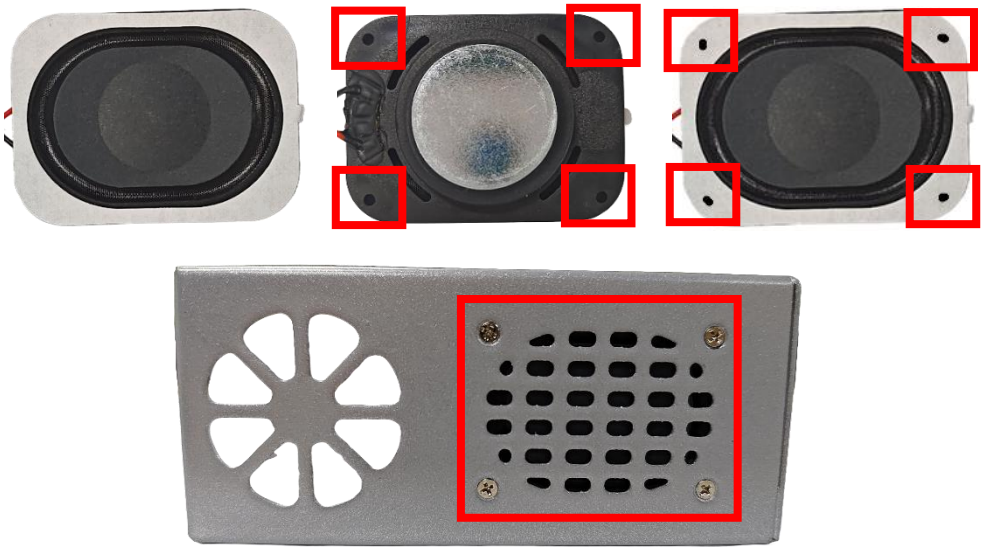
It is recommended to assemble and use the Freenove Computer Case for Raspberry Pi according to this tutorial. Otherwise, it may lead to incorrect device installation or damage. Please check all the parts again. If there are any incorrect or missing parts in your kit, please contact us in time.

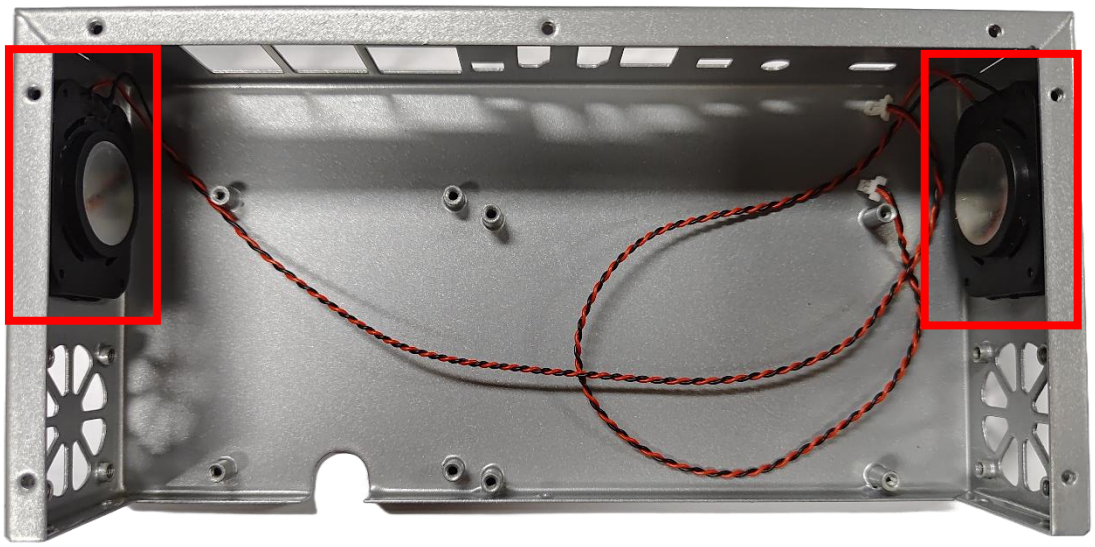
Please assemble the product when it is powered off.

Step 1 Installation of Fans and Speakers

Installing the Speakers

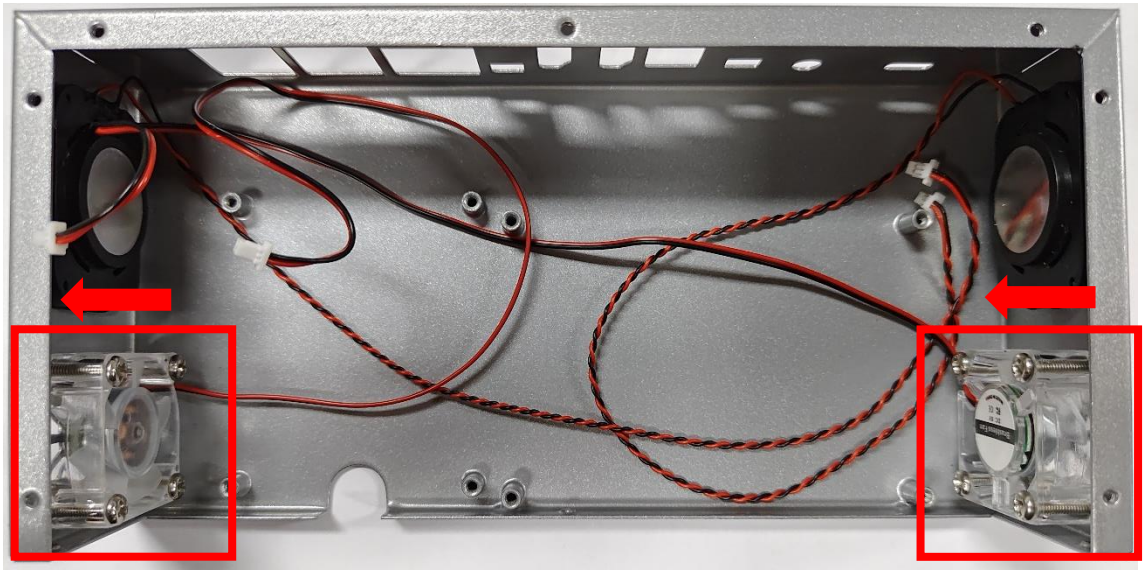
When installing the speakers, the rubber pad might interfere with the installation process. To address this, you can initially use four M2 x 4 Countersunk Head Screws to pre-drill the four mounting holes. Once the holes are prepared, position the speaker inside the case. Afterward, secure the speaker from the exterior of the case using four M2 x 4 Countersunk Head Screws. Ensure that the rubber pad side of the speaker is firmly pressed against the case.





Installing the Fans

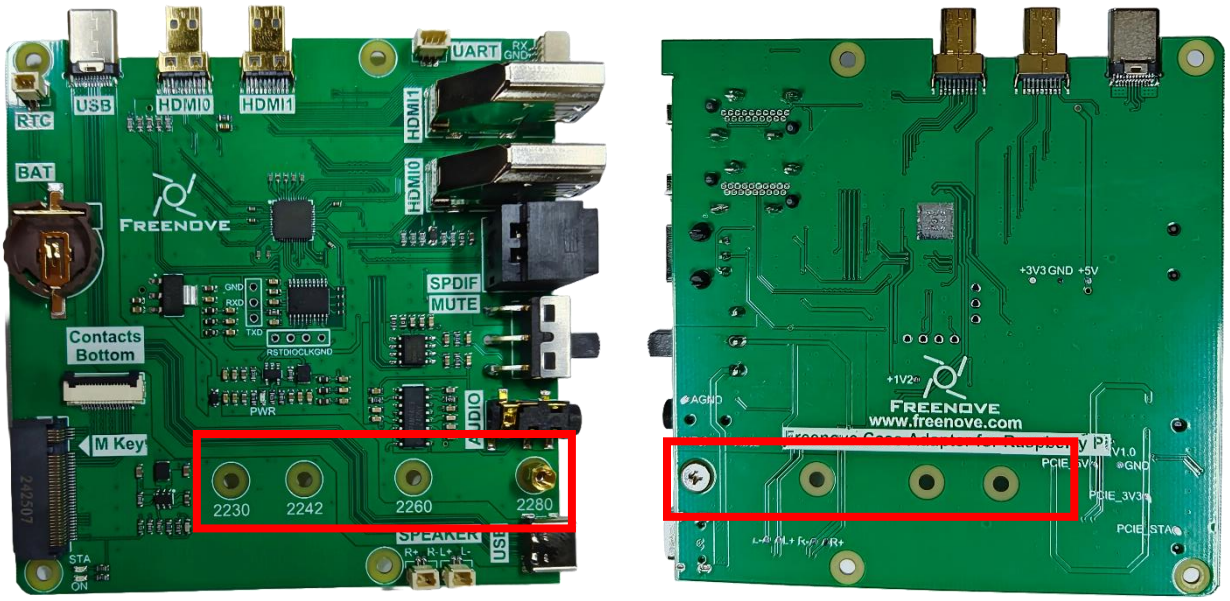
Use the M3 x 12 screws to secure the fans to the inside of the computer case. Pay close attention to the assembly orientation of the fan as they should form an air duct (indicated by the red arrows), which shows the direction of airflow. The fan on the left side is responsible for blowing the hot air out of the computer case's interior, while the fan on the right side draws in cool air.



Step 2 Installing Rpi 5 and Case Adapter Board

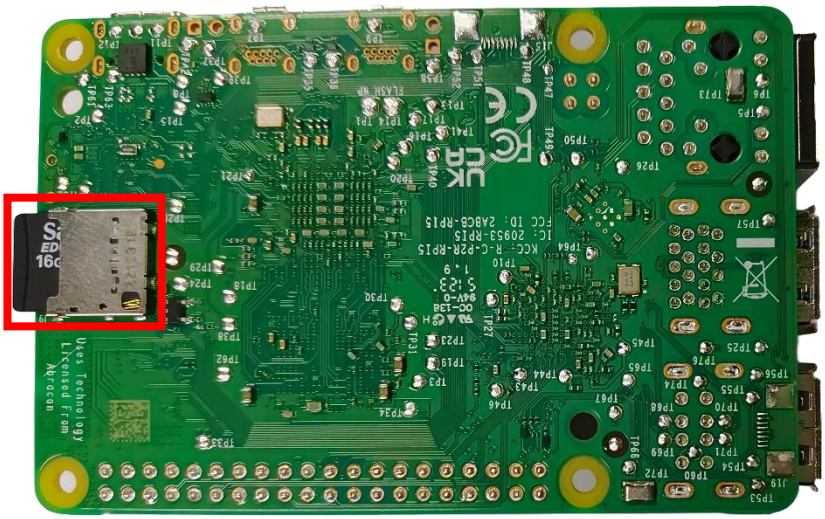
Installing the NVMe SSD Securing Standoffs

Install the brass standoffs for securing the NVMe SSD on the Case Adapter Board as illustrated below. Utilize M2.5x3 Screws to affix the M2.5x5 Brass Standoffs to the 2280 mounting holes. Depending on the length of your NVMe SSD, please install them on the appropriate holes.

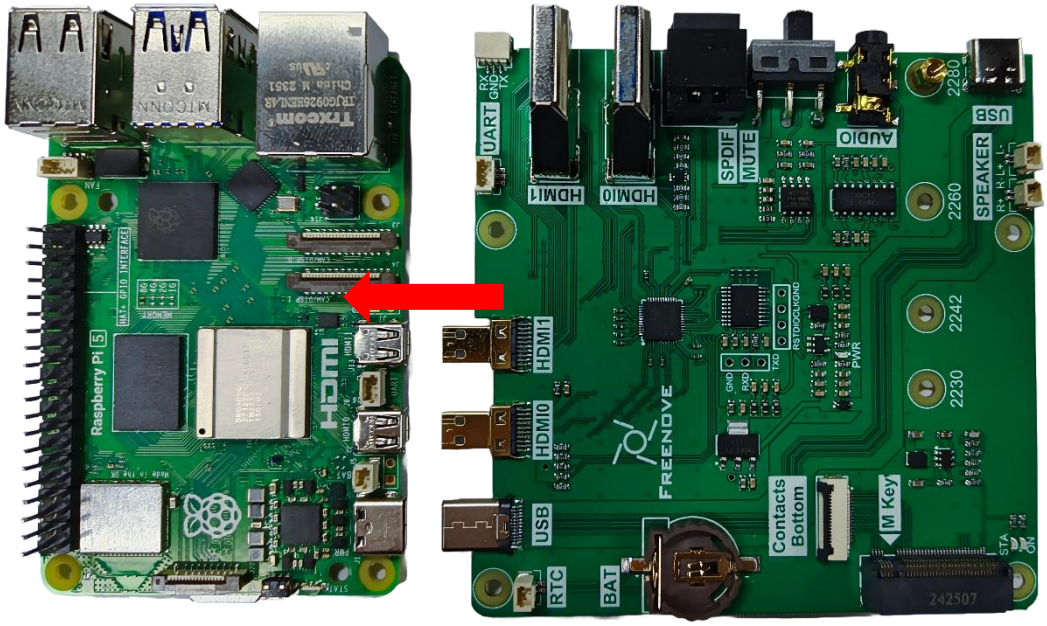


Installing RPi 5 and Case Adapter Board

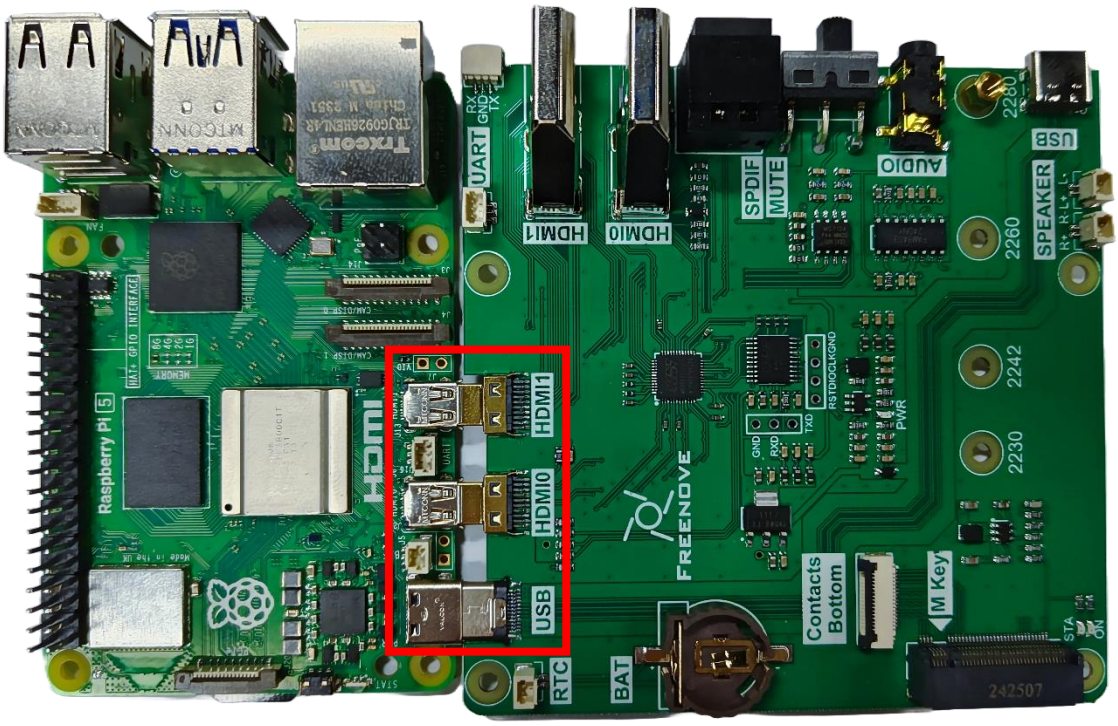
Inserting the SD card to the SD card slot on the back of RPi 5.



Align the Type-C interface, HDMI0 interface, and HDMI1 interface of the RPi 5 precisely with the corresponding interfaces on the Case Adapter Board. Then, connect the two boards together. Please follow the arrow directions for assembly.

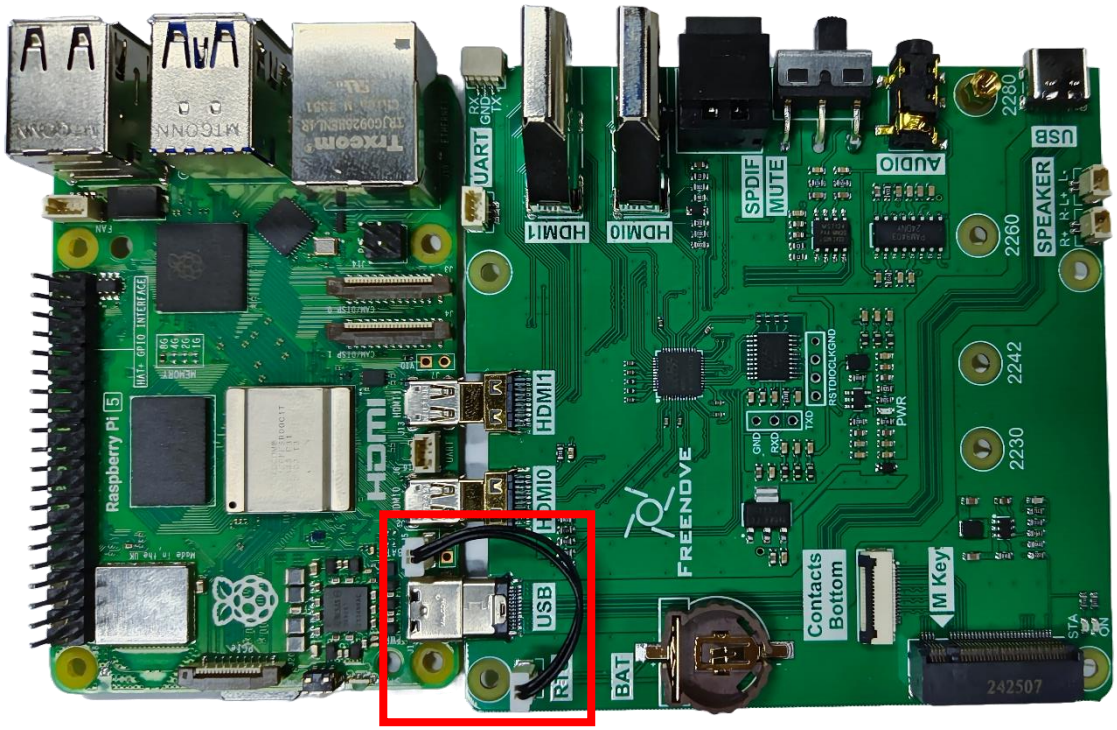


The following image shows the two boards after connection.

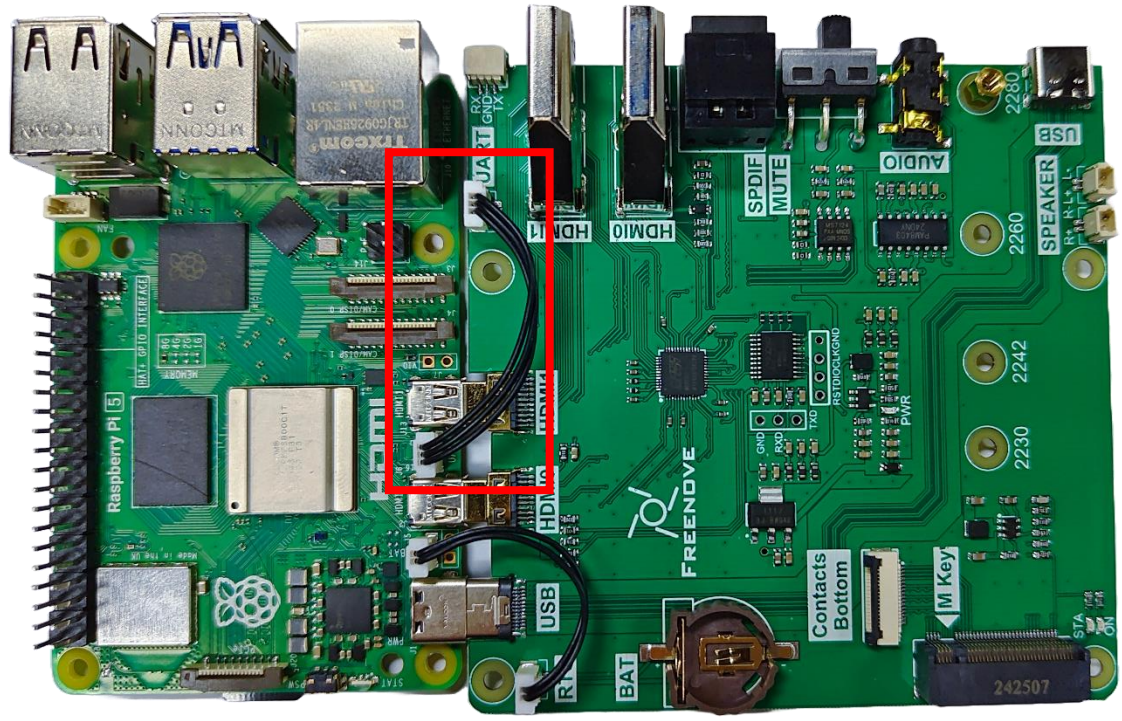


Connecting Cables of RTC, UART and the Speakers

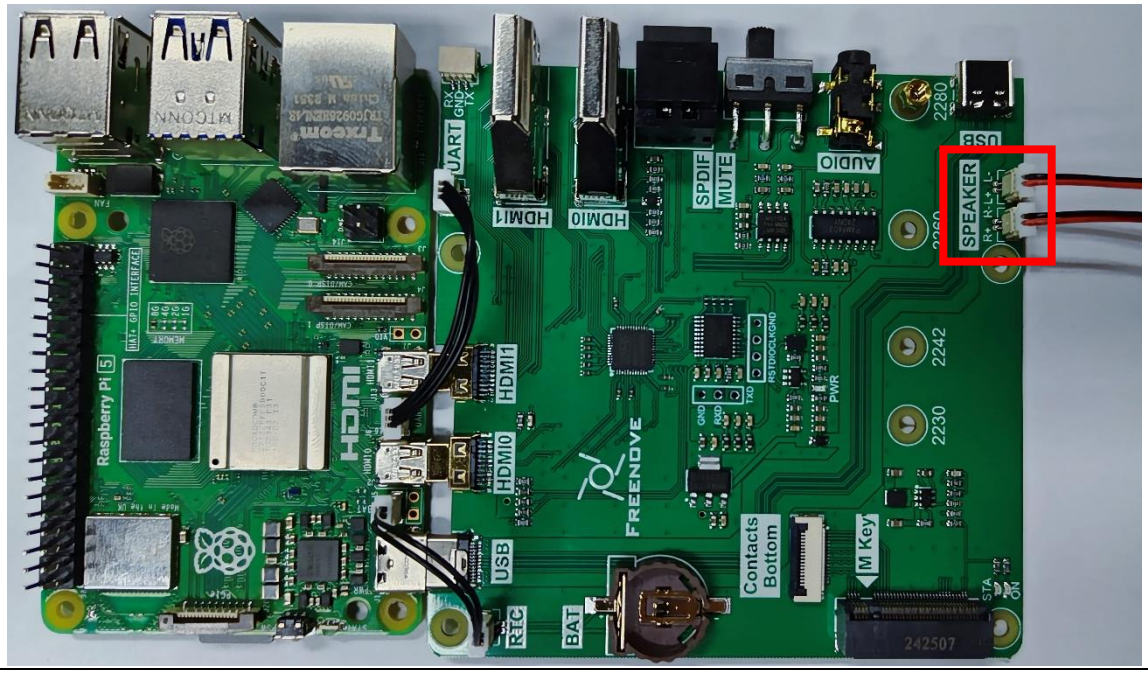
Connect the RPi 5's BAT connector to the RTC connector of the Case Adapter Board with the SH-1.0mm-2P cable.



Connect the RPi 5's UART connector to the UART connector of the Case Adapter Board with the SH-1.0mm-3P cable.

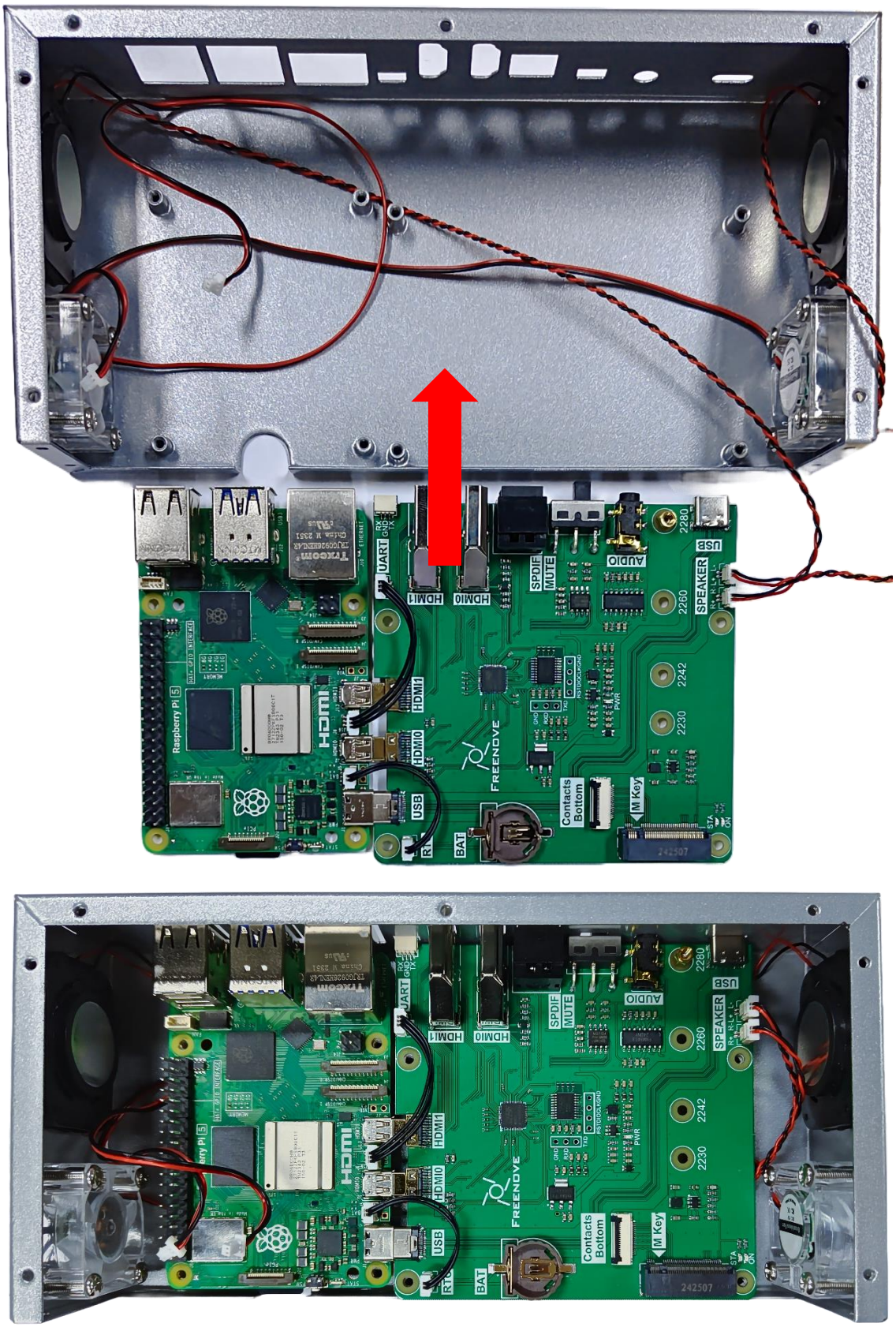


Connect the cables of the two speakers to the SPEAKER interface on the Case Adapter Board.

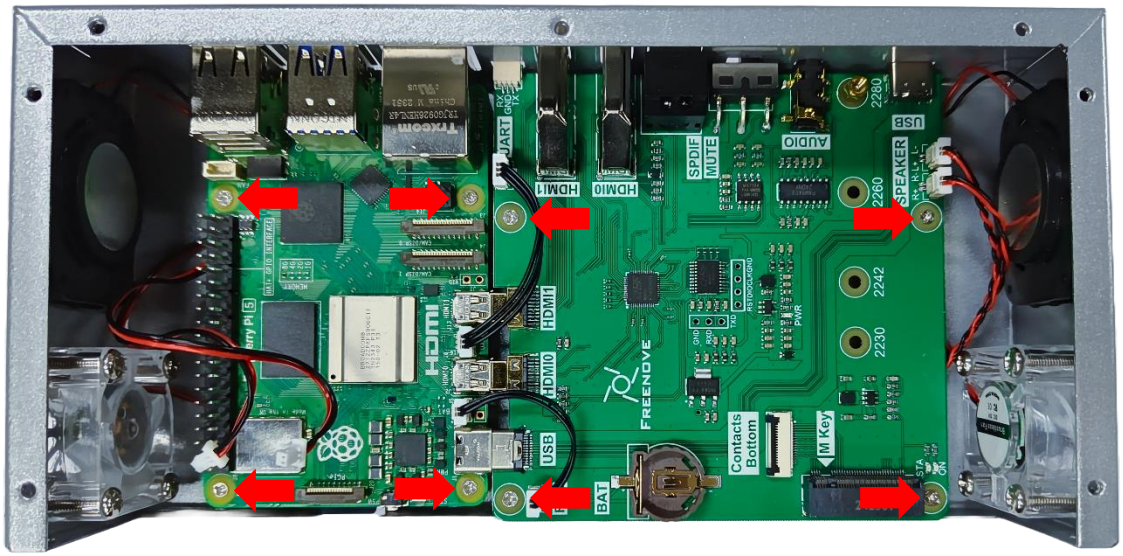


Assembly with the Case

Place the combined Raspberry Pi 5 and Case Adapter Board into the case.



Fix them to the case with M2.5 x 5 Countersunk Head Screws.



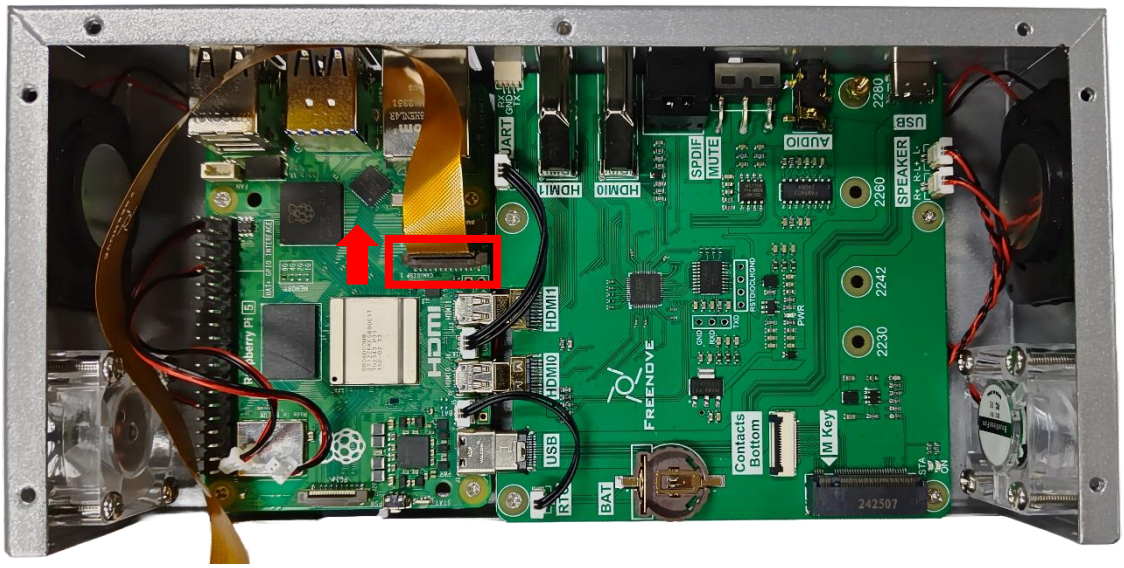
Step 3 Connecting Cables

Connecting Cables

1. Connecting the camera cable

   Connect the shorter end of the camera to the CAM/DISP 1 interface on the RPi 5.

   Pay attention to the orientation of the contacts. (The red arrow below indicates the orientation of the contacts).

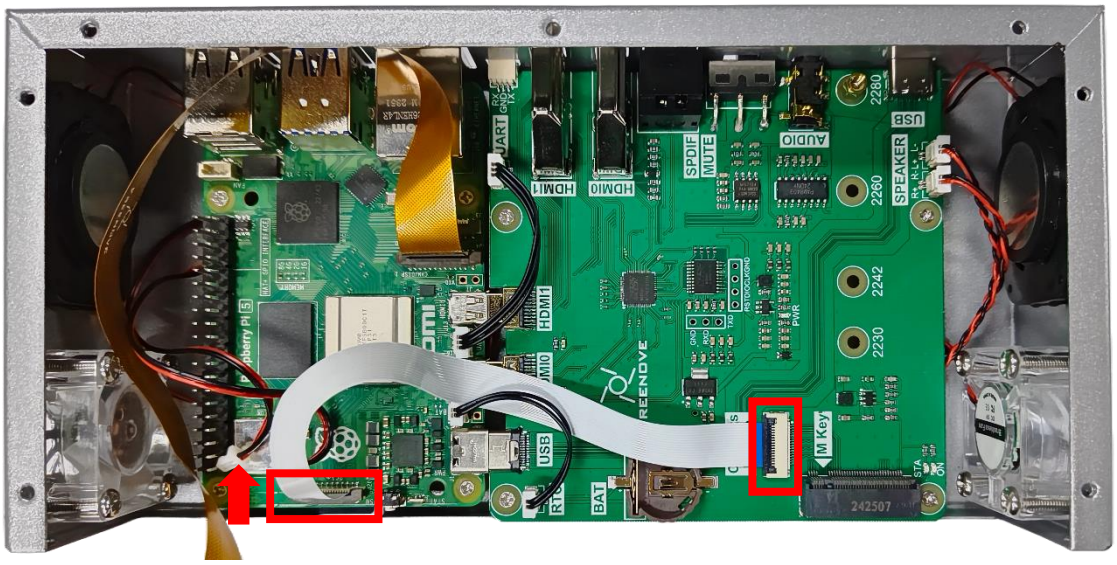


Caution: Although the Raspberry Pi camera cable and screen cable may look identical in appearance, they are NOT interchangeable.

2. Connect the NVMe SSD cable

Connect the cable for NVME SSD to the PCIe interface on the RPi 5. Pay attention to the orientation of the contacts (The red arrow below indicates the contacts orientation.)

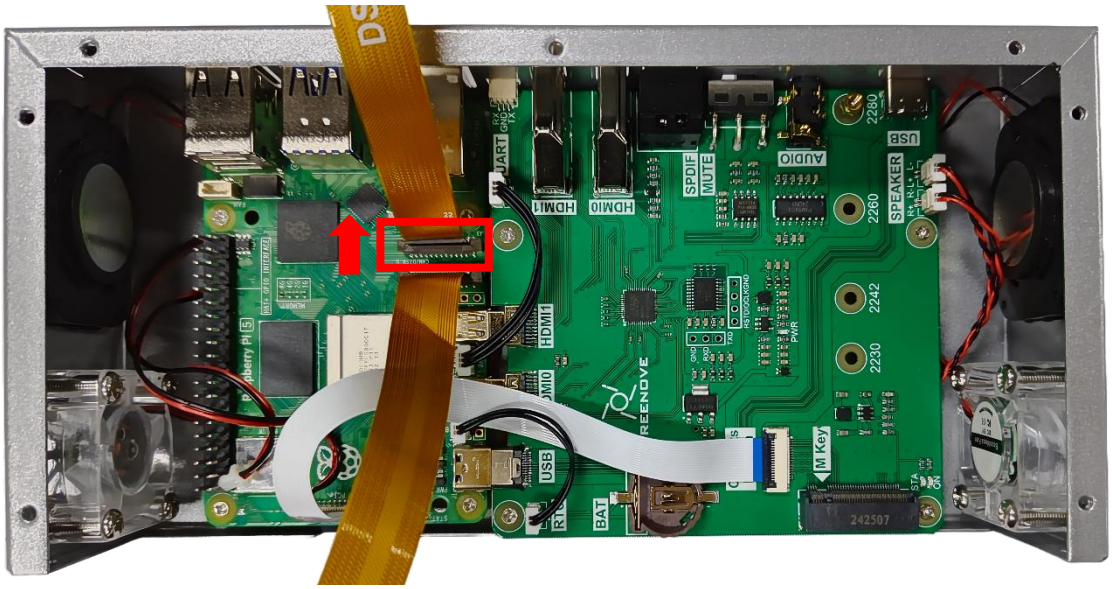
Connect the other end of the cable to the FPC connector of the Case Adapter Board (Contacts Bottom).



3. Connect the screen cable

   If your purchase is not an FNK0100K, please skip this.

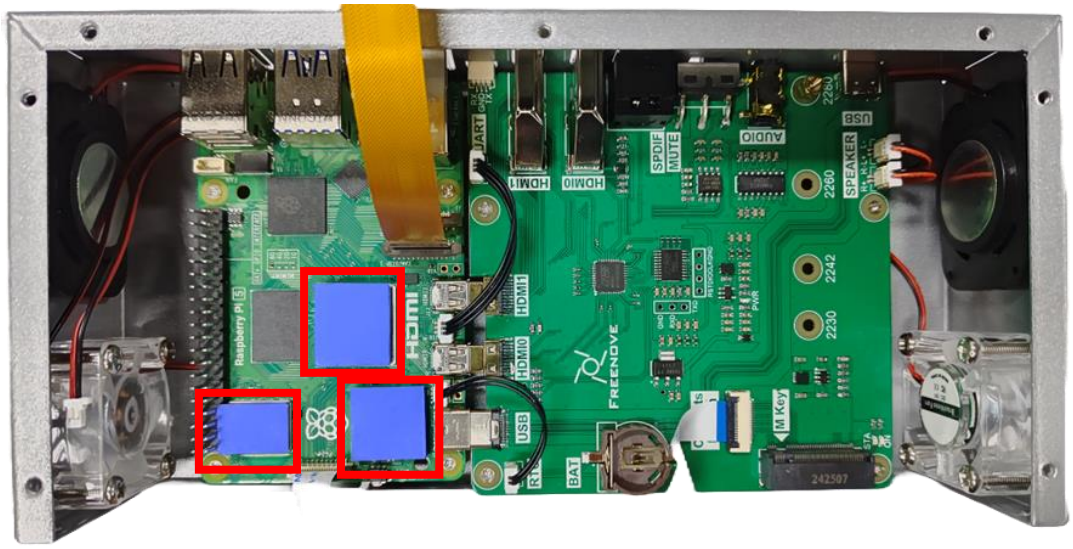
   Connect the shorter end of the cable to the CAM/DISP 0 inferface of RPi 5. Pay attention to the contacts orientation (the red arrow below indicates the contacts orientation).



Caution: Although the Raspberry Pi camera cable and screen cable may look identical in appearance, they are NOT interchangeable.

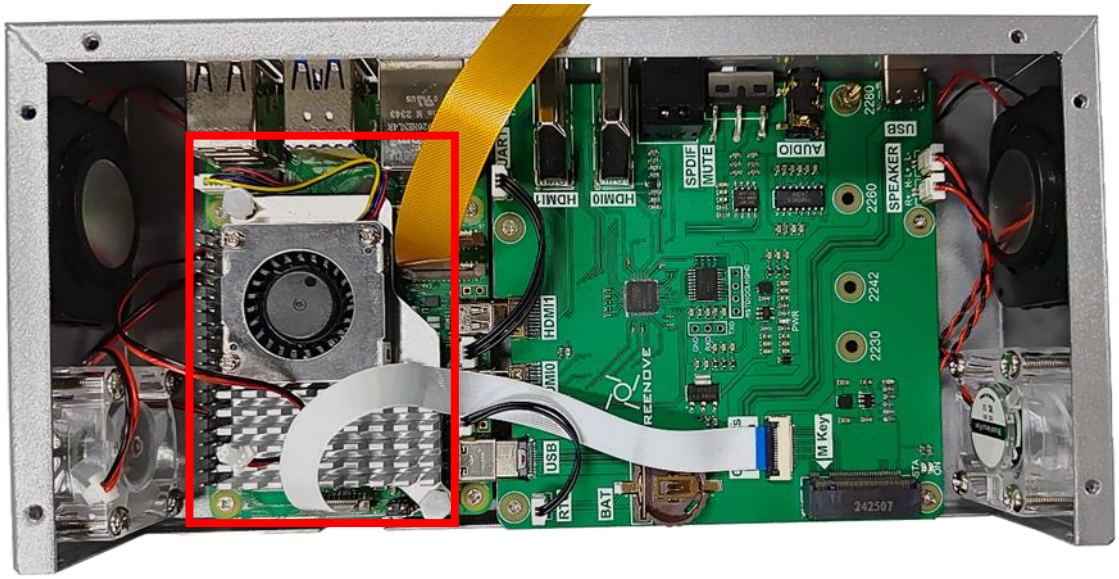
Installing the RPi 5 CPU cooler

Attach the heat dissipation silicone pads to the power management chip, CPU, and WIFI module of the Raspberry Pi 5 respectively. (Note: Peel off the adhesive tape on the heat sink pad.)



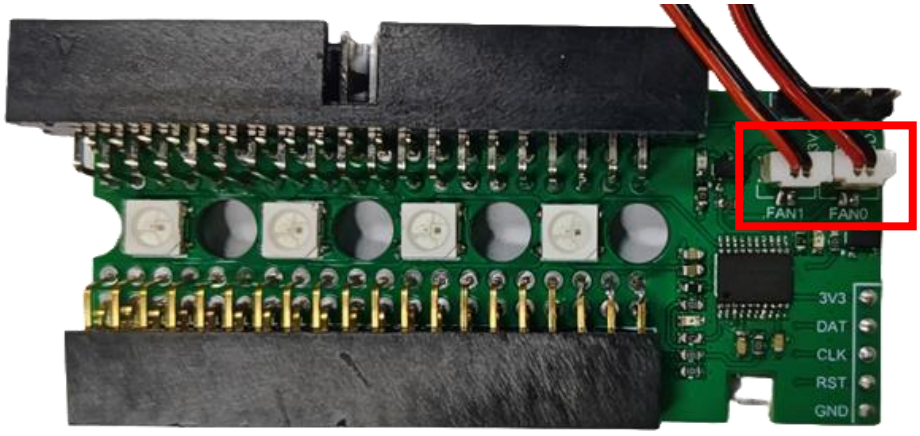
Connect the cable of the active cooler to the fan interface on the RPi 5.

Fix the radiator on the Raspberry Pi 5 with nylon fixing pins.

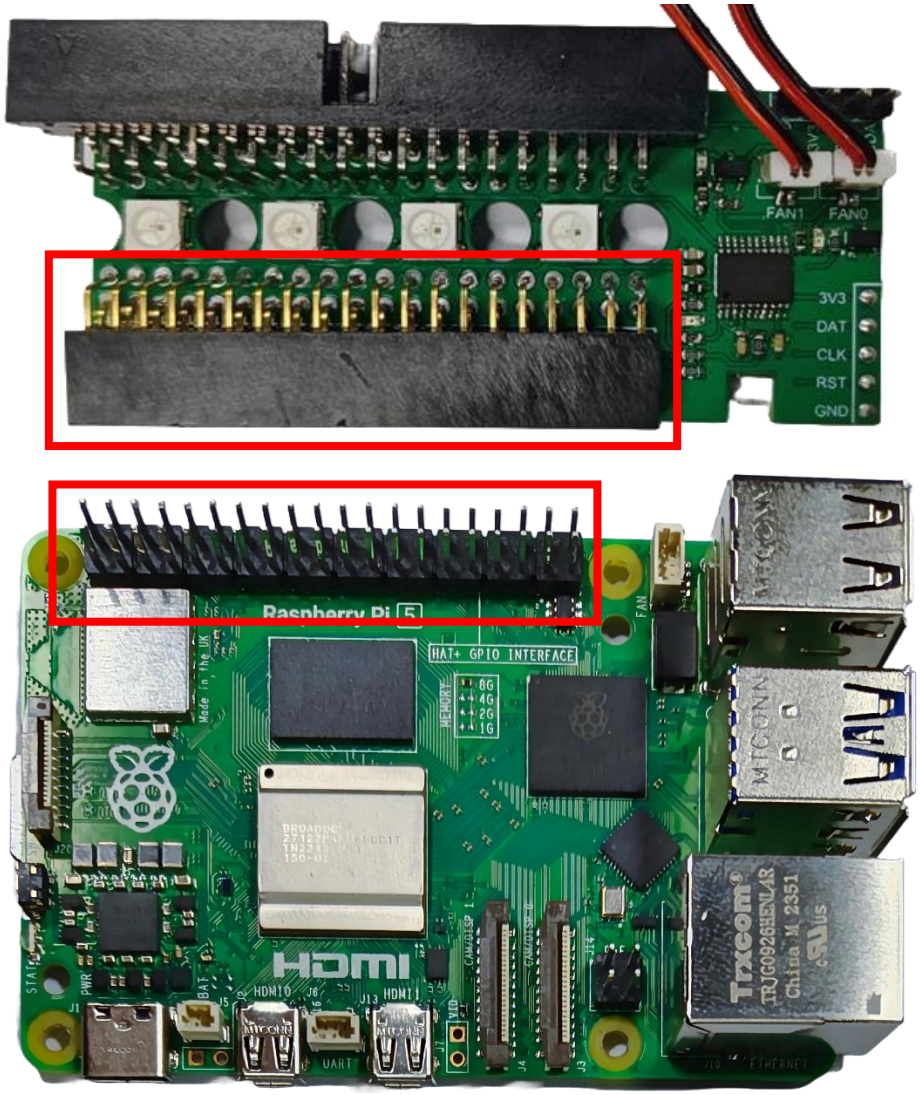


Installing GPIO Board

Connect the two fans on the computer case to the FAN0 and FAN1 interfaces on the GPIO Board respectively.



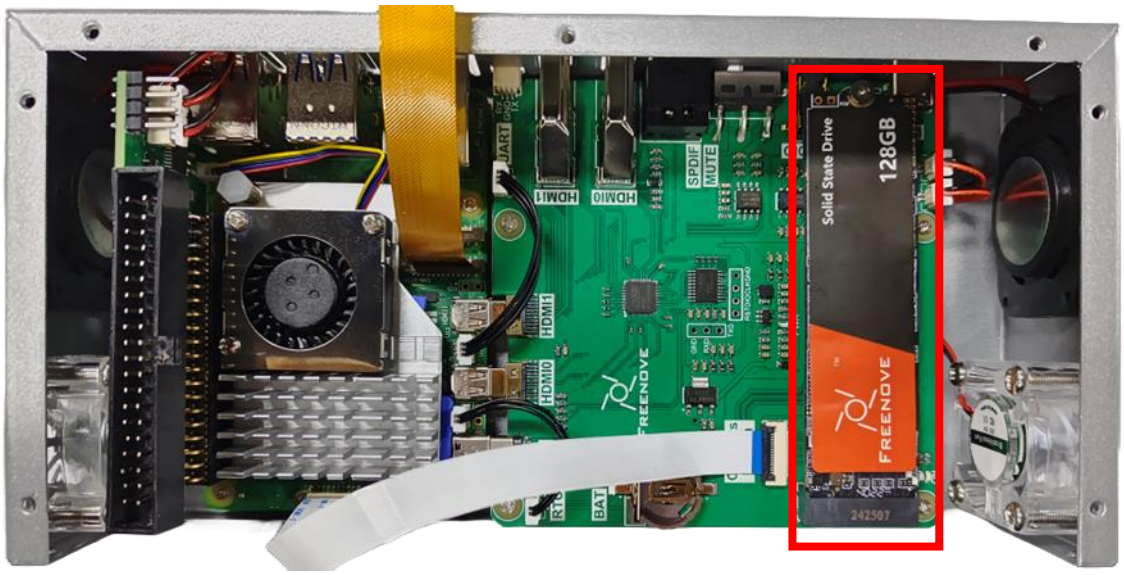
Install the GPIO Board on the GPIO interface of the Raspberry Pi. Please pay attention to the assembly direction and avoid misalignment during installation.



After installation, it should look like the figure below.

Installing NVMe SSD

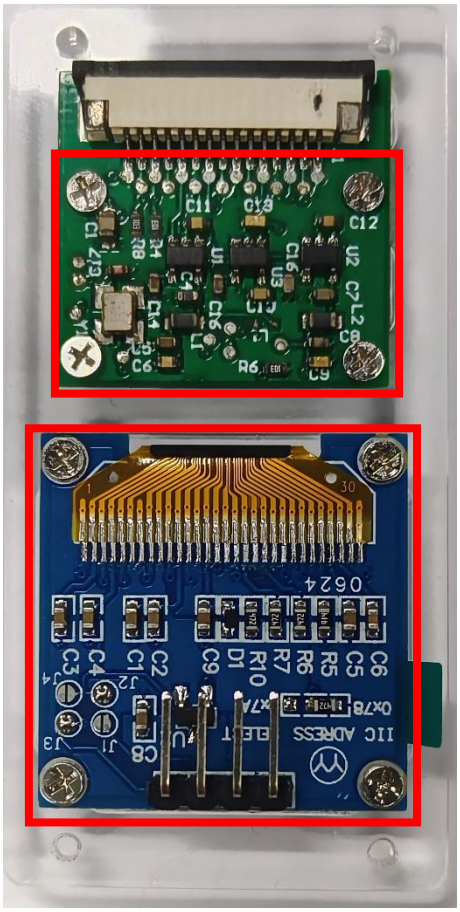
Insert the NVMe SSD diagonally into the M.2 interface, and use the M2.5x3 Screws to fix it onto the brass standoffs for the NVMe SSD.



Step 4 Installing the Top Cover

Installing the Camera and the OLED Display

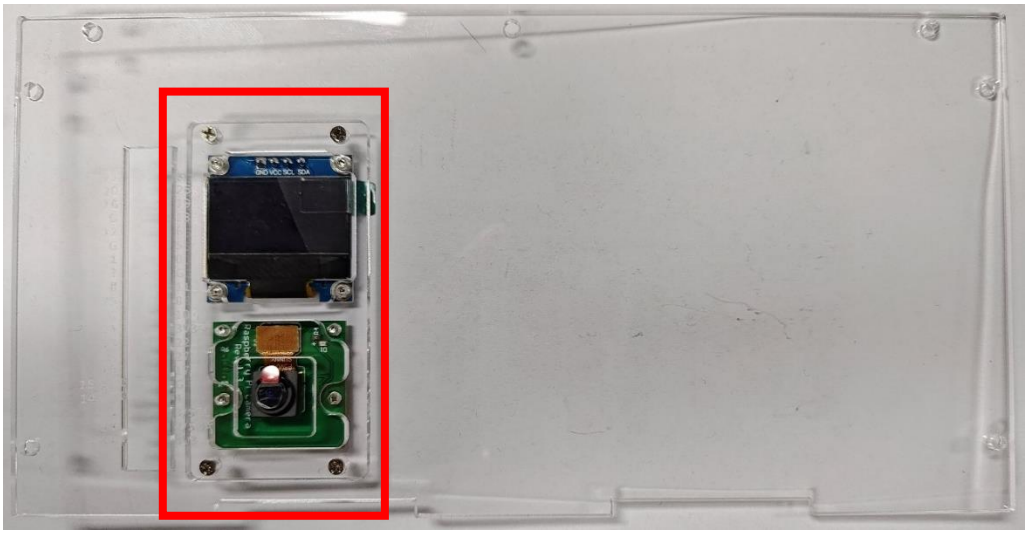
Fix the OLED display and the camera module on the acrylic part respectively with M2 x 4 Countersunk Head Screws.



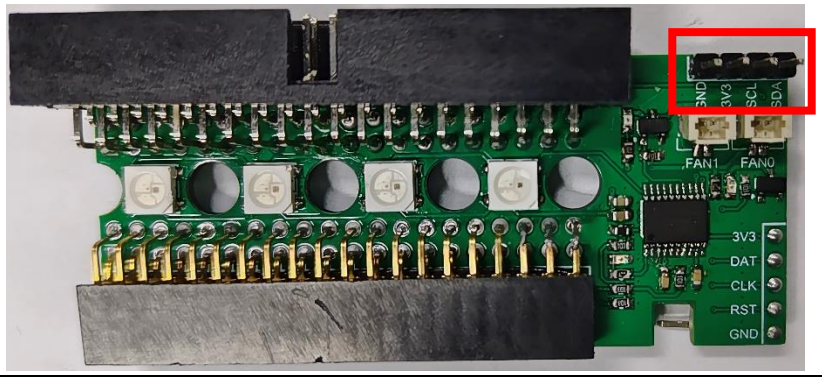
Assembly of FNK0100B Top Cover

If your purchase is FNK0100K, please skip to the next step.

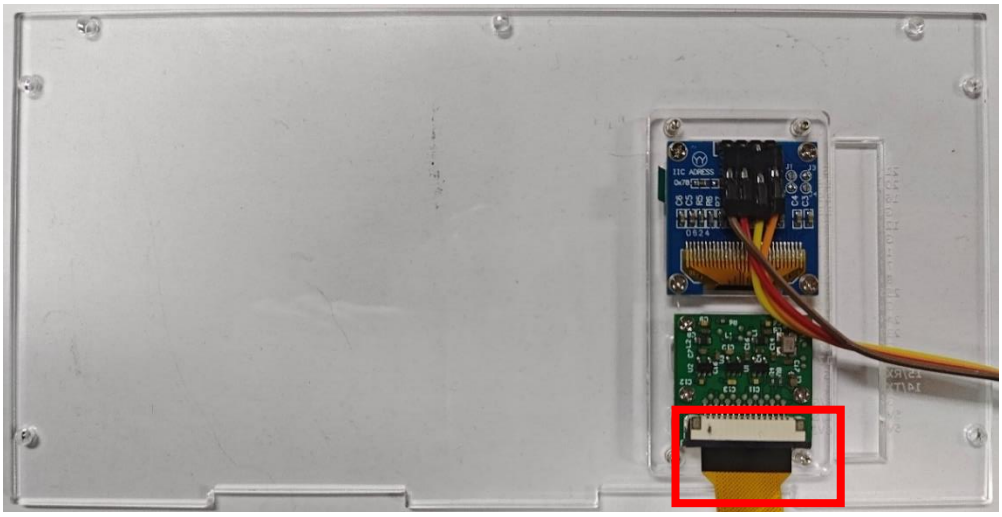
Attach the assembled OLED screen and camera module beneath the top cover. Secure them from above using M2 x 6 Countersunk Head Screws, positioning them specifically at the left side of the cover.



Use the F-F jumper cable to connect the interface of the OLED display to the 1x4P pin header interface on the GPIO Board. Please note the pin numbering carefully. Incorrect connection may result in damage to the components.

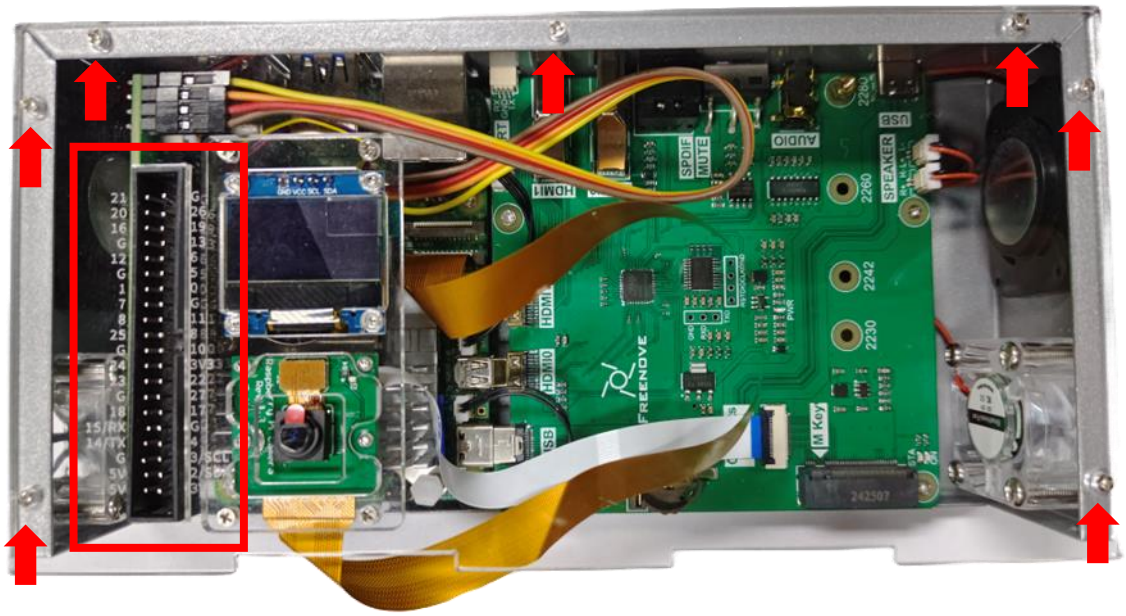


Connect the other end of the camera cable to the camera module, with the contacts facing downward.

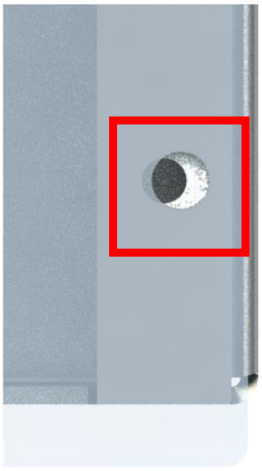


Carefully install the top cover onto the case, ensuring that none of the cables come into contact with the CPU cooler. Failure to do so may result in damage to the CPU cooler.

Align the holes on the left side of the cover with the horn seats of the GPIO Board. Then, use M2.5 x 5 Countersunk Head Screws to secure the top cover firmly in place.



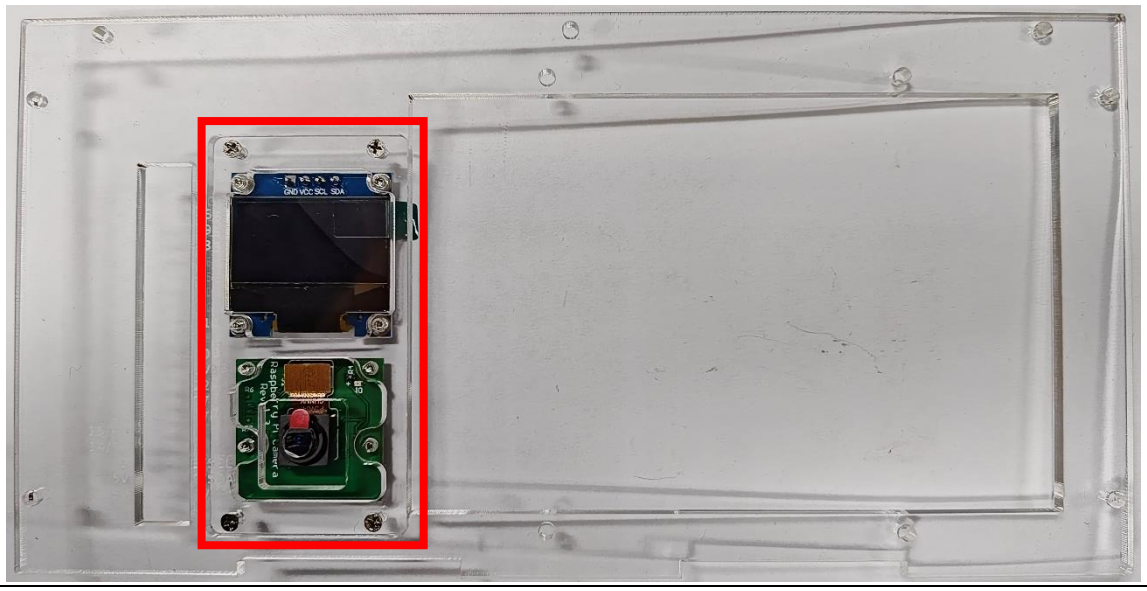
Note: Owing to the machining precision limitations of sheet metal parts, if you notice that certain screw holes do not align properly, this is not an indication of an error. In such cases, you can apply force to bend and align them as needed.



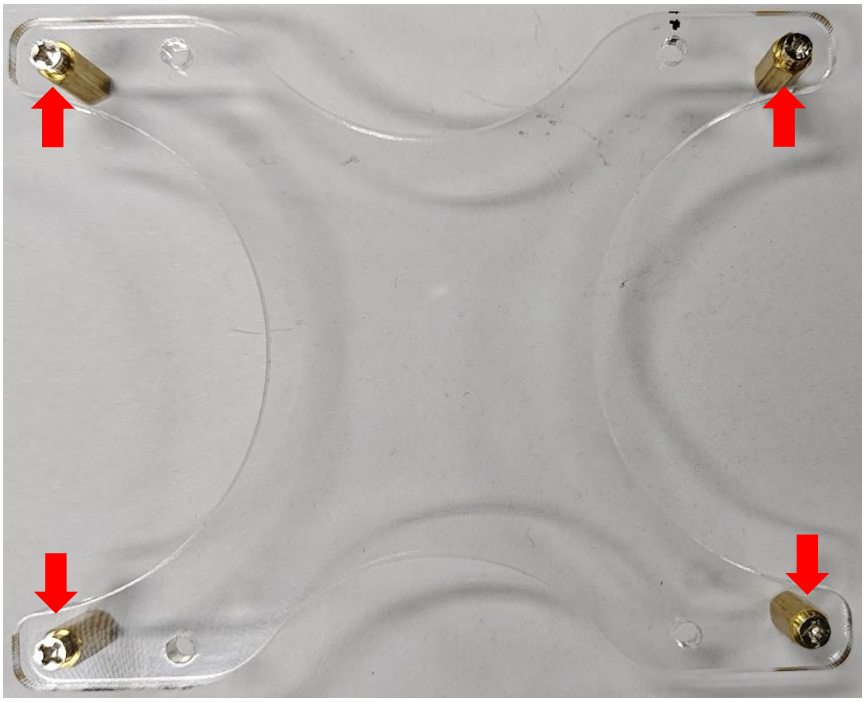
Assembly of FNK0100K Top Cover

If your purchase is an FNK0100B and you have assembled to top cover, you can skip to the next step.

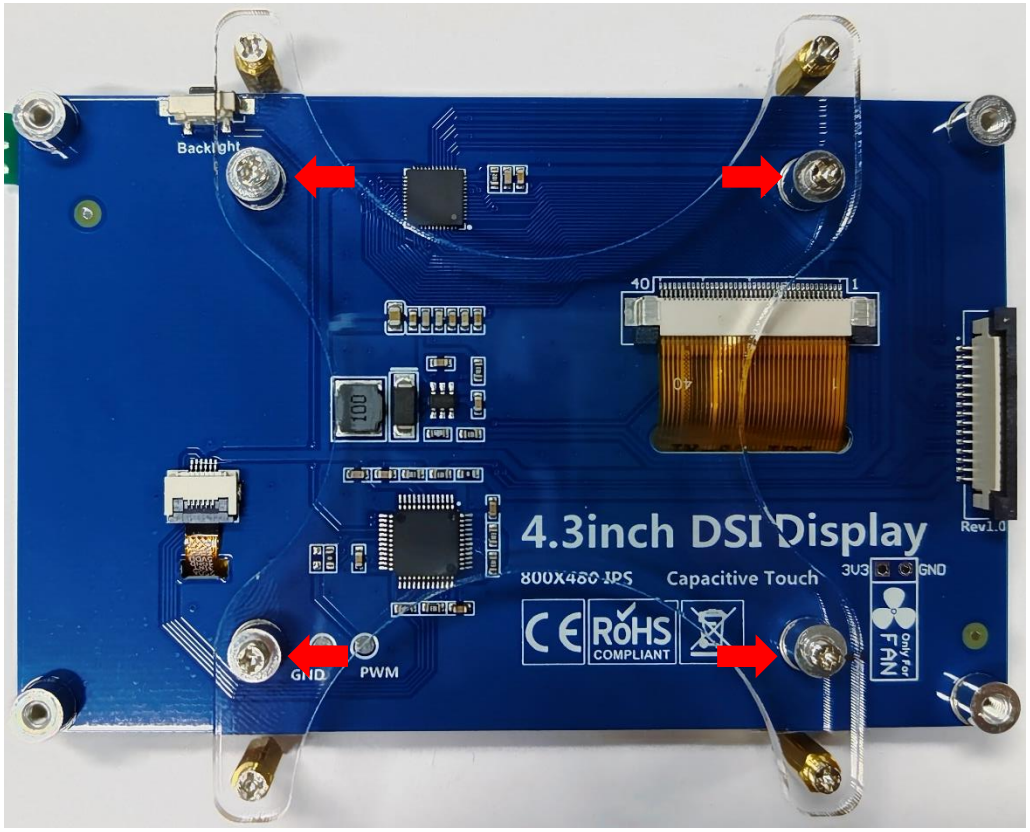
Attach the assembled OLED screen and camera module beneath the top cover. Secure them from above using M2 x 6 Countersunk Head Screws, positioning them specifically at the left side of the cover.



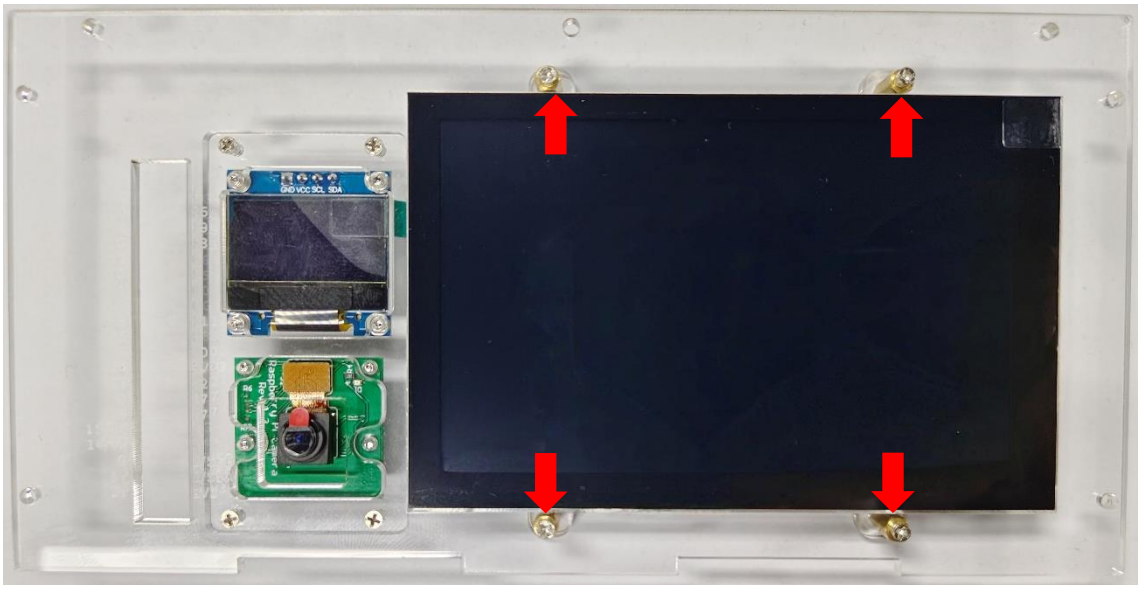
Use a combination of M2.5 x 5 Countersunk Head Screws and M2.5 x 9 Brass Standoffs in the outer holes at the four corners of the screen support plate.



Install the screen on the screen support plate and fix it in the inner holes at the four corners with M2.5 x 5 Countersunk Head Screws.

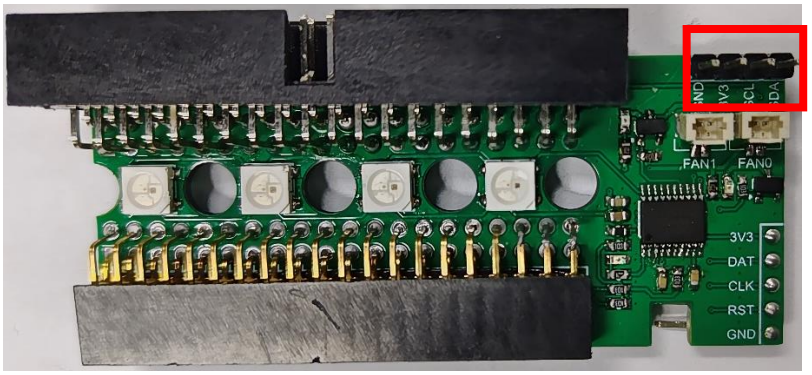


Carefully insert the assembled screen and screen support plate from the underside into the rectangular groove located on the right side of the top cover. Once properly positioned, secure them firmly from above using M2.5 x 5 Countersunk Head Screws.

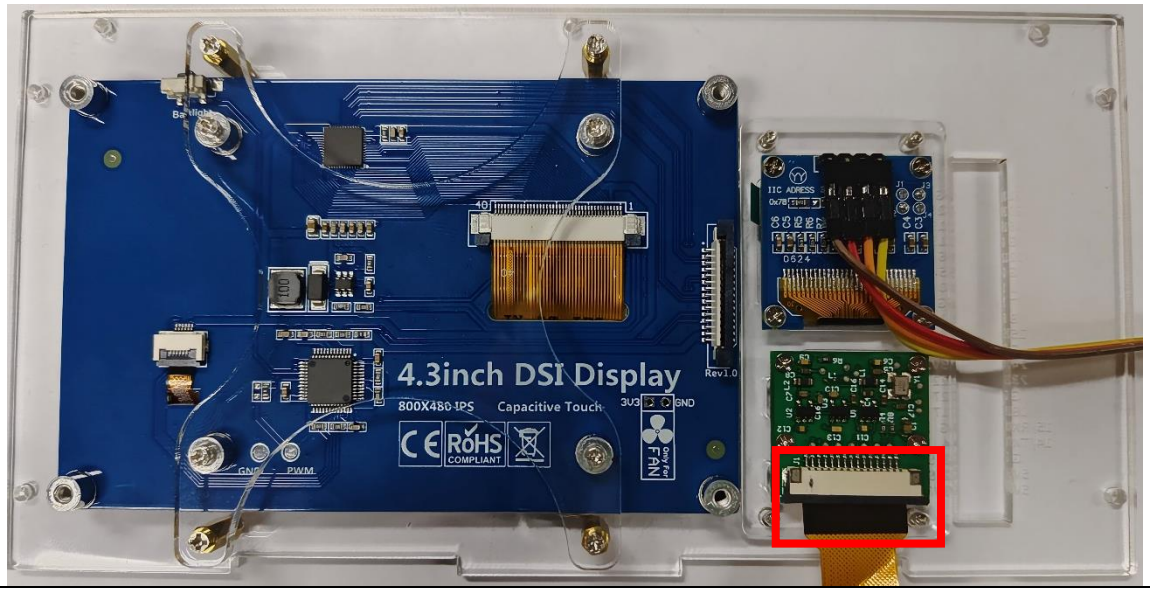


Connect the interface of the OLED display and the 1x4P pin header interface on the GPIO Board with the F-F jumper cable.

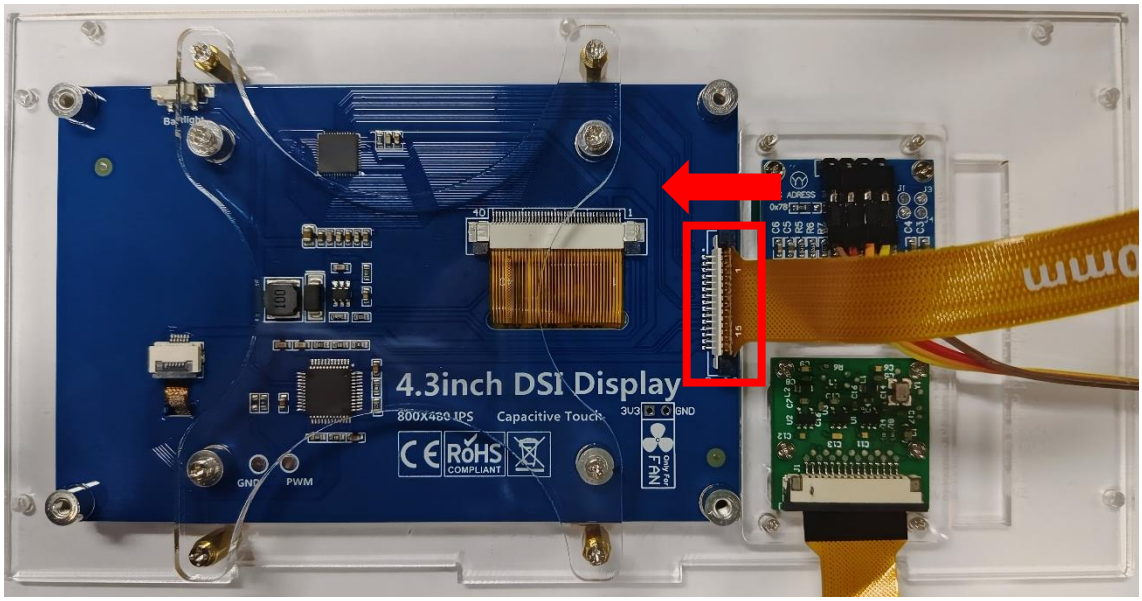
Pay close attention to the pin numbering. Incorrect connection may cause damage to the components.



Connect the other end of the camera cable to the camera module, with the contacts facing downward.

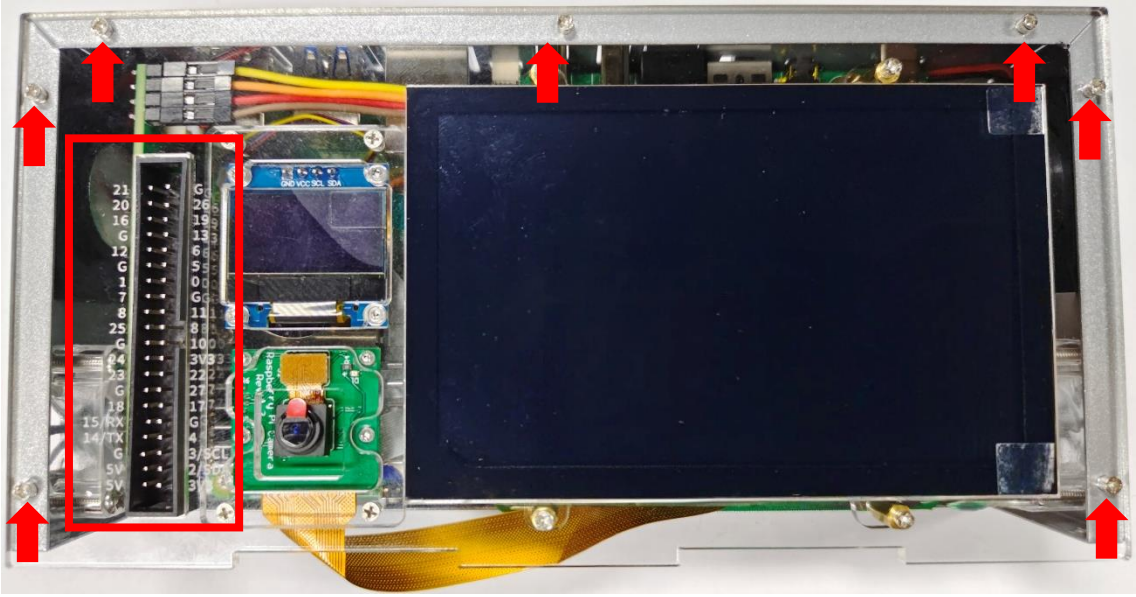


Connect the other end of the screen wire to the screen, paying attention to the orientation of the contacts (the red arrow indicates the direction of the contacts).

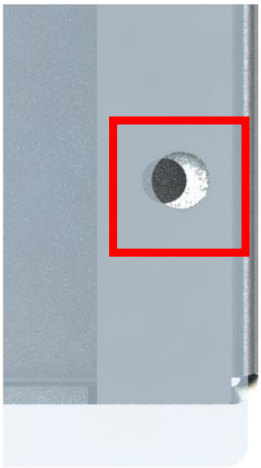


Carefully install the top cover onto the case, ensuring that none of the cables come into contact with the CPU cooler. Failure to do so may result in damage to the CPU cooler.

Align the holes on the left side of the cover with the horn seats of the GPIO Board. Then, use M2.5 x 5 Countersunk Head Screws to secure the top cover firmly in place.



Note: Owing to the machining precision limitations of sheet metal parts, if you notice that certain screw holes do not align properly, this is not an indication of an error. In such cases, you can apply force to bend and align them as needed.



Step 5 Installing the Side Board

Insert the power button cap into the square hole on the side panel, and use the M2.5 x 5 Countersunk Head Screw to fix the side panel to the computer case.

Following the above assembly steps, you have now successfully assembled your computer case. If you encounter any issues during the assembly process, please feel free to contact us at any time.
support@freenove.com

Chapter 3 Install Raspberry Pi OS

After the assembly is complete, we will start installing the system for the Raspberry Pi. Regardless of whether you want to install the system on the NVMe SSD or not, you need to install the system on an SD card or USB flash drive first.

Analysis

1. First of all, make sure you can enter the Raspberry Pi OS via SD card or U drive.
2. After booting the Raspberry Pi, you can use it to flash the OS image directly onto the NVMe SSD. Alternatively, you can purchase an NVMe SSD to USB adapter and flash the image using USB on Windows or macOS, much like you would for an SD card or USB drive.
3. With this analysis in mind, we can systematically carry out the necessary steps.

Raspberry Pi models manufactured at different times might not boot up in the same way as described, but that's okay; just follow our guide to proceed.
There are various ways to burn the Raspberry Pi OS to SSD, each requiring different hardware tools.

Ways	Ways of burning	Requirements
1	Use Raspberry Pi to burn the OS. This requires that you can successfully boot up the Raspberry Pi via SD card or U disk. (**Recommended, described in this tutorial**)	An SD card or a U disk that can access the Raspberry Pi OS.
2	Purchase an NVMe SSD to USB adapter and flash the image just like you would with an SD card or USB drive.	NVME SSD to USB adapter (need to be bought separately)
3	If there are spare M.2 NVME interface on the motherboard of your PC, you can insert the SSD to it to flash the OS.	PC with M.2 NVME interface

Caution: Incompatible SSDs

The recognition and read/write operations of the NVMe SSD are handled by the drivers of the Raspberry Pi 5. If you find that your SSD cannot be recognized by Pi 5 or is not readable/writable, please try to find a driver suitable for your SSD and install it on the Raspberry Pi, or replace the SSD, or purchase the adapter kit that comes with the SSD.

3.1 Flashing OS to SD Card or USB Drive

Based on the analysis above, our first step should be to install the Raspberry Pi operating system onto an SD card or USB drive, with a capacity of at least 16GB. If you are already able to boot the Raspberry Pi using an SD card or USB drive, you can skip this section and move on to the next section.

3.1.1 Component List

Required Components

Raspberry Pi 5 x 1 	One 27W power adapter (or a power adapter compatible with Raspberry Pi official one that can output 5.1V/5A) 
Micro SD Card（TF Card）x1，Card Reader x1 	

3.1.2 Raspberry Pi OS

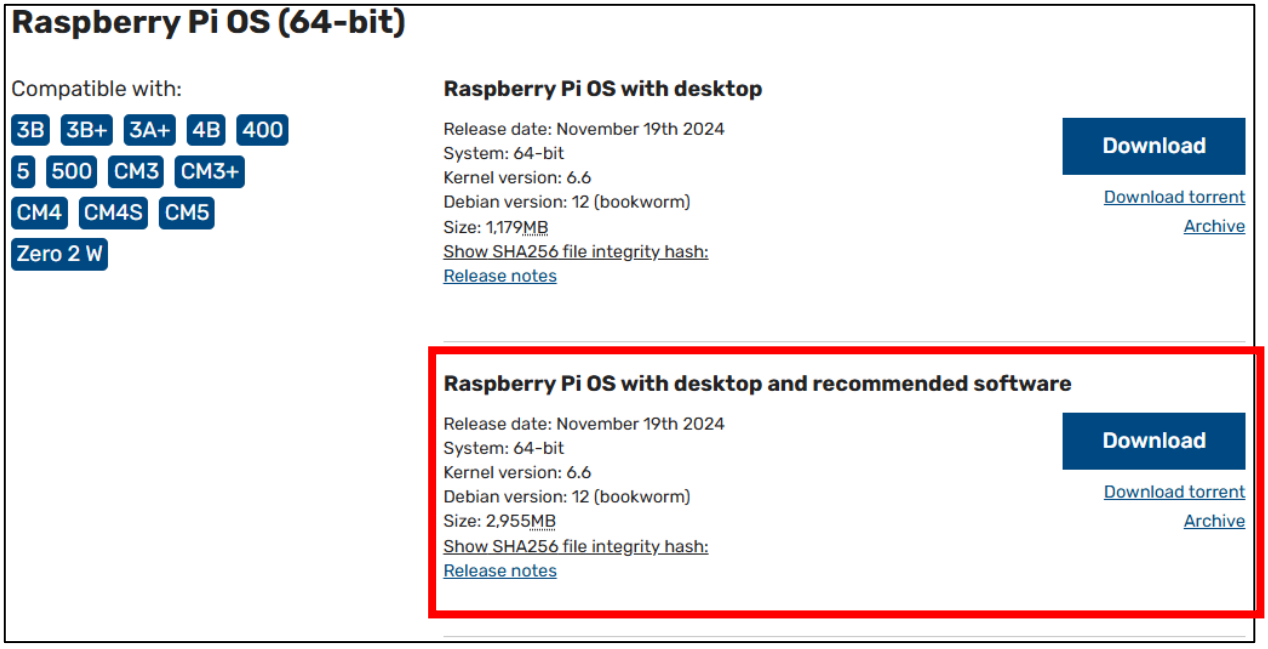
Without Screen - Use Raspberry Pi - under Windows PC: https://youtu.be/XpiT_ezb_7c
With Screen - Use Raspberry Pi - under Windows PC: https://youtu.be/HEywFsFrj3I

Automatically Method

You can follow the official method to install the system for raspberry pi via visiting link below:
https://projects.raspberrypi.org/en/projects/raspberry-pi-setting-up/2
In this way, the system will be downloaded **automatically** via the application.

Manually Method

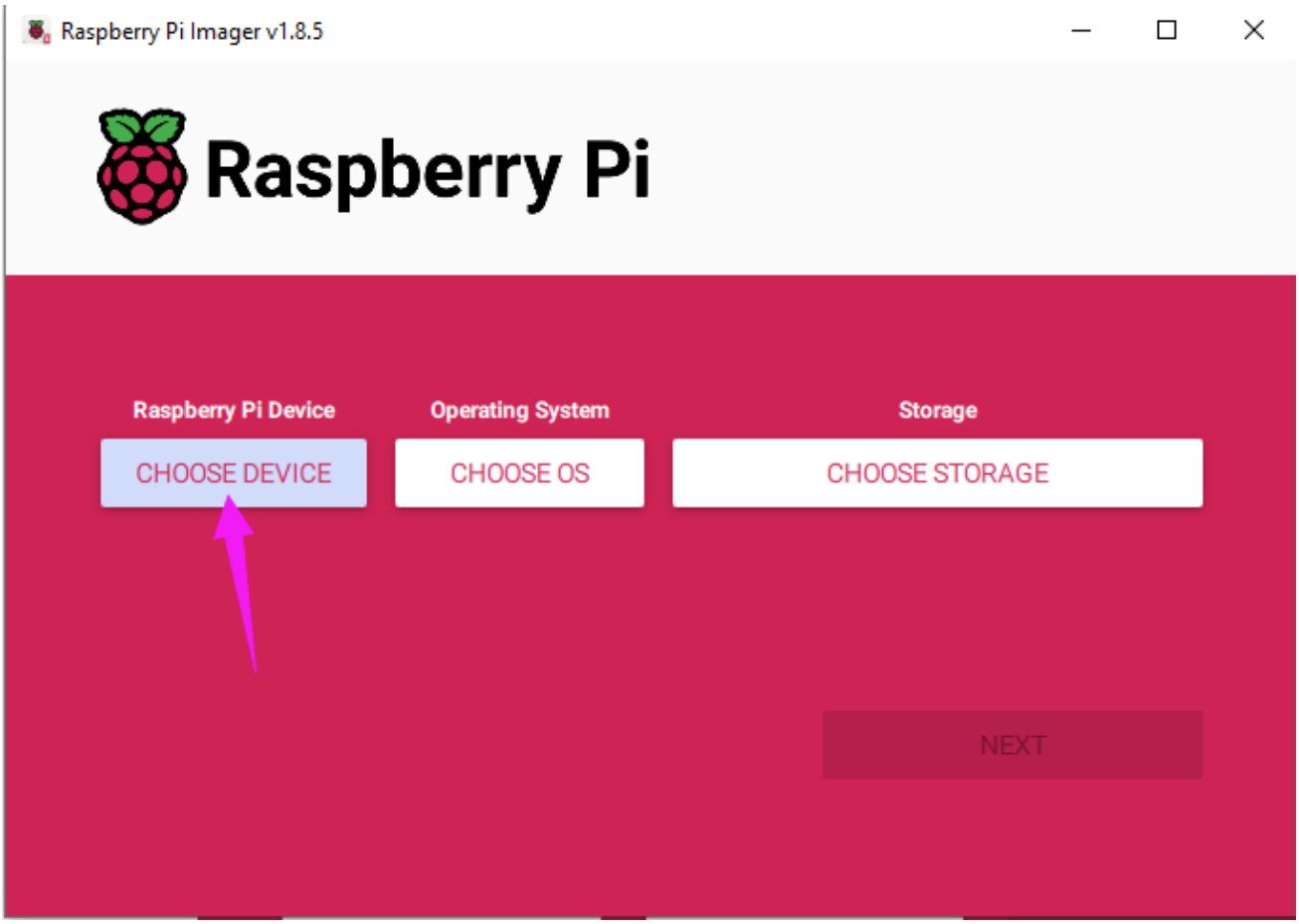
After installing the Imager Tool in the **link above**. You can **also** download the system **manually** first.
Visit https://www.raspberrypi.com/software/operating-systems/



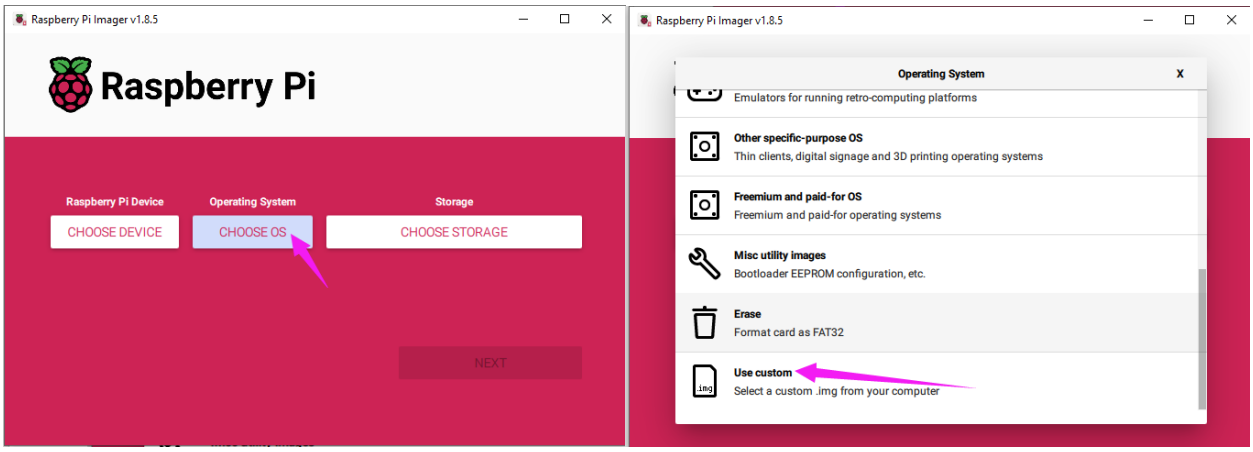
Write System to Micro SD Card

First, put your Micro **SD card** into card reader and connect it to USB port of PC.

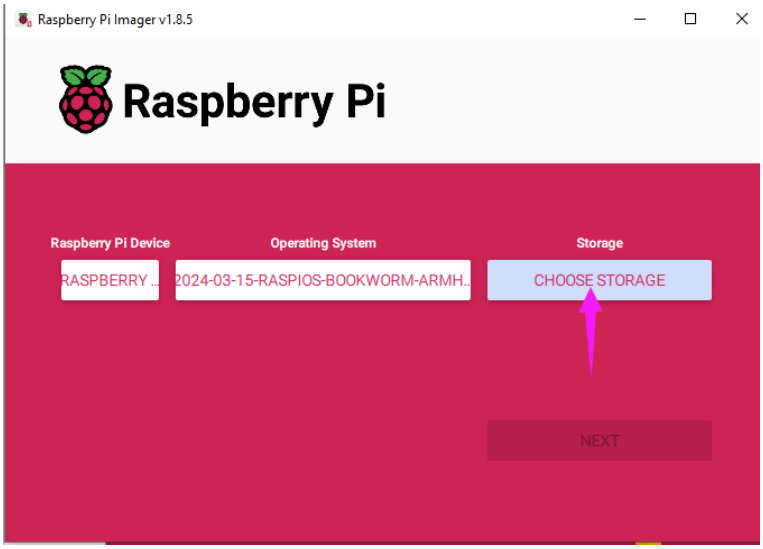
Open Raspberry Pi Imager.
Choose Raspberry Pi 5 as the device.

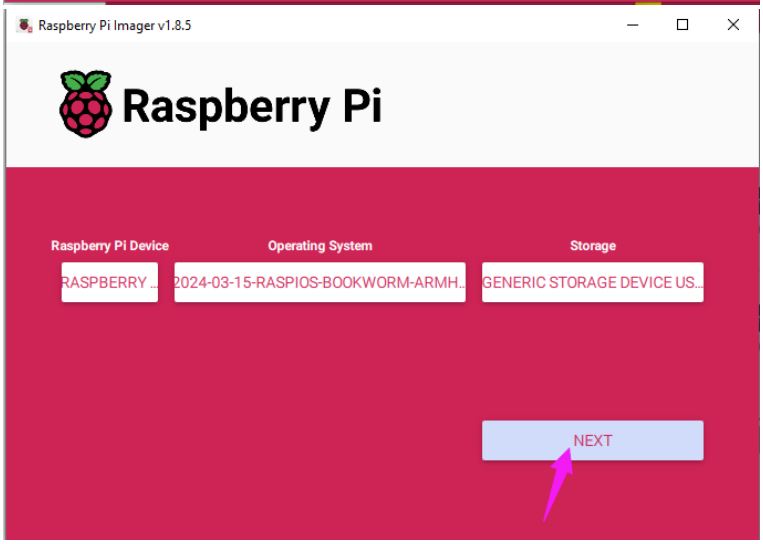


Choose the system that you just downloaded in Use custom.



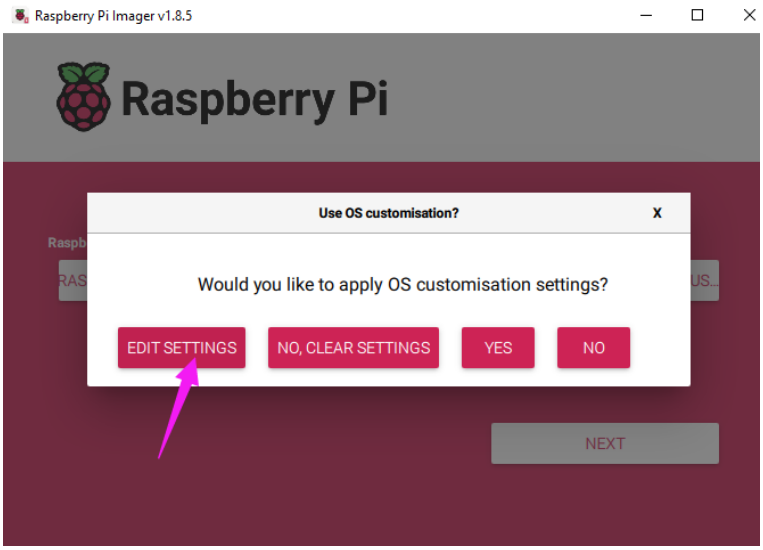
Choose the SD card and click on Next.



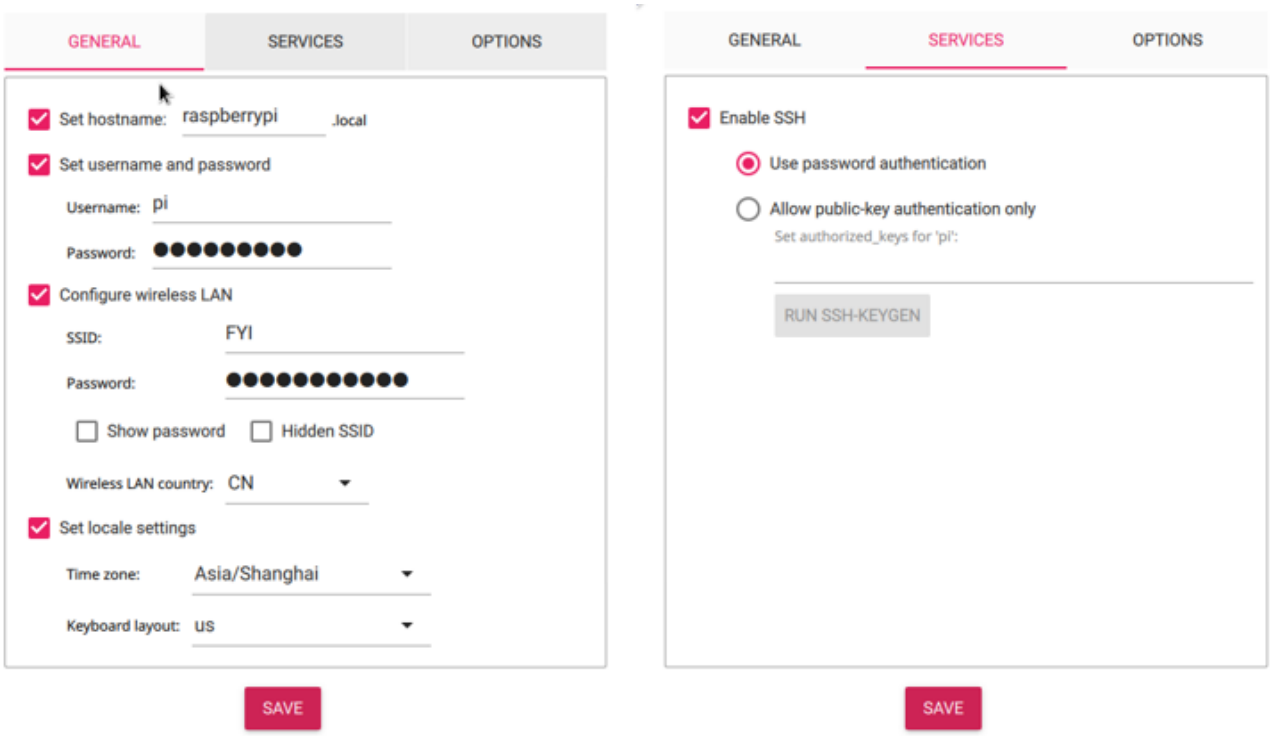


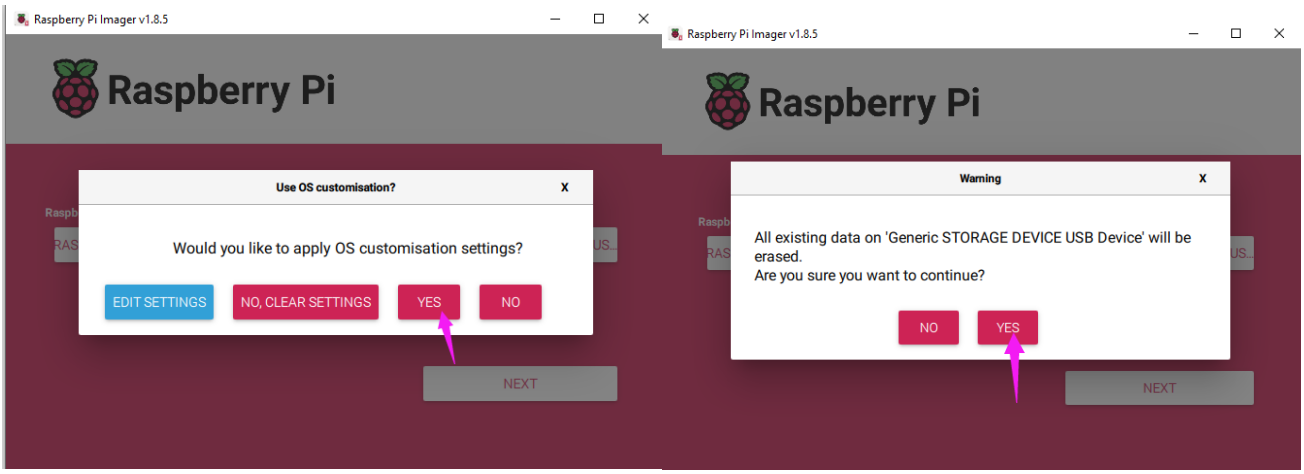
Enable ssh and configure WiFi

Click EDIT SETTINGS.



Configure wireless LAN, enable SSH and click Save.



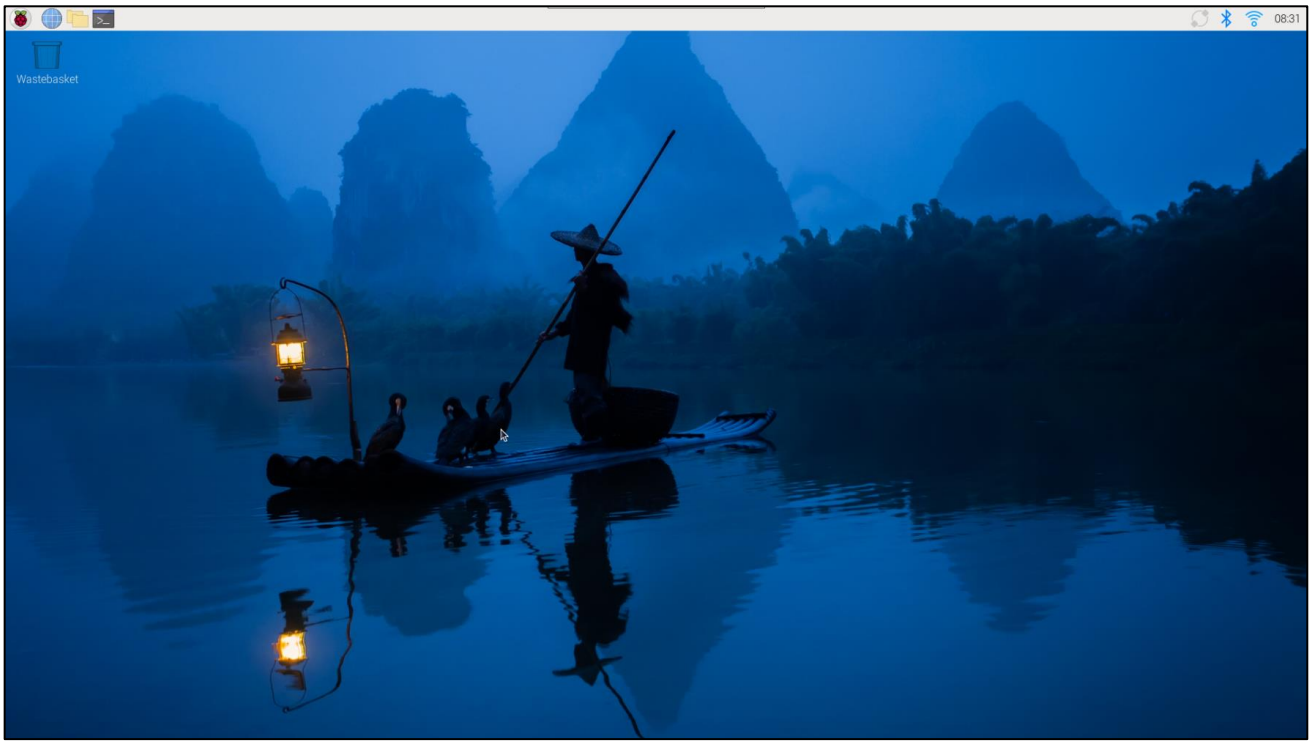


Wait for it to finish writing and verifying.

3.1.3 Monitor desktop

If you do not have a spare monitor, please skip to next section Remote desktop & VNC. If you have a spare monitor, please follow the steps in this section.

After the system is written successfully, take out Micro SD Card and put it into the SD card slot of RPi. Then connect your RPi to the monitor through the HDMI port, attach your mouse and keyboard through the USB ports, attach a network cable to the network port and finally, connect your power supply (making sure that it meets the specifications required by your RPi Module Version. Your RPi should start (power up). Later, after setup, you will need to enter your user name and password to login. The default user name: pi; password: raspberry. After login, you should see the following screen.



Congratulations! You have successfully installed the RASPBERRY PI OS operating system on your RPi.

Raspberry Pi 5 integrates a Wi-Fi adaptor. You can use it to connect to your Wi-Fi. Then you can use the wireless remote desktop to control your RPi. This will be helpful for the following work. Raspberry Pi of other models can use wireless remote desktop through accessing an external USB wireless card.

3.1.4 Remote desktop & VNC

If you have logged in Raspberry Pi via display, you can skip to VNC Viewer.

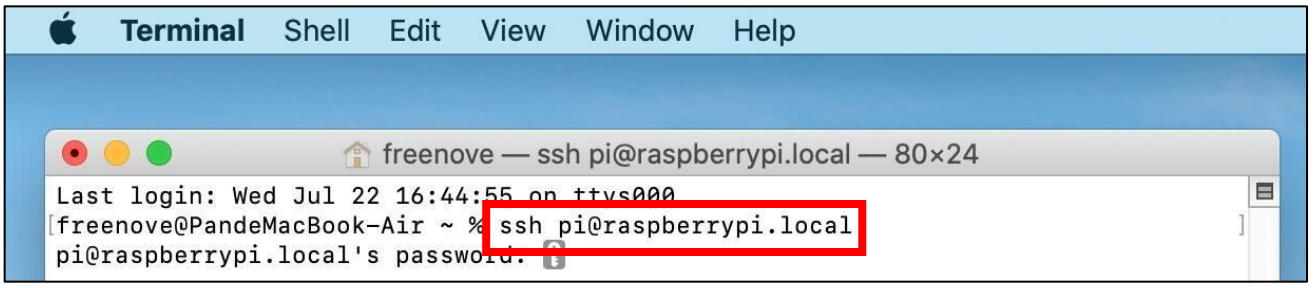
If you don't have a spare display, mouse and keyboard for your RPi, you can use a remote desktop to share a display, keyboard, and mouse with your PC. Below is how to use:
MAC OS remote desktop and Windows OS remote desktop.

MAC OS Remote desktop

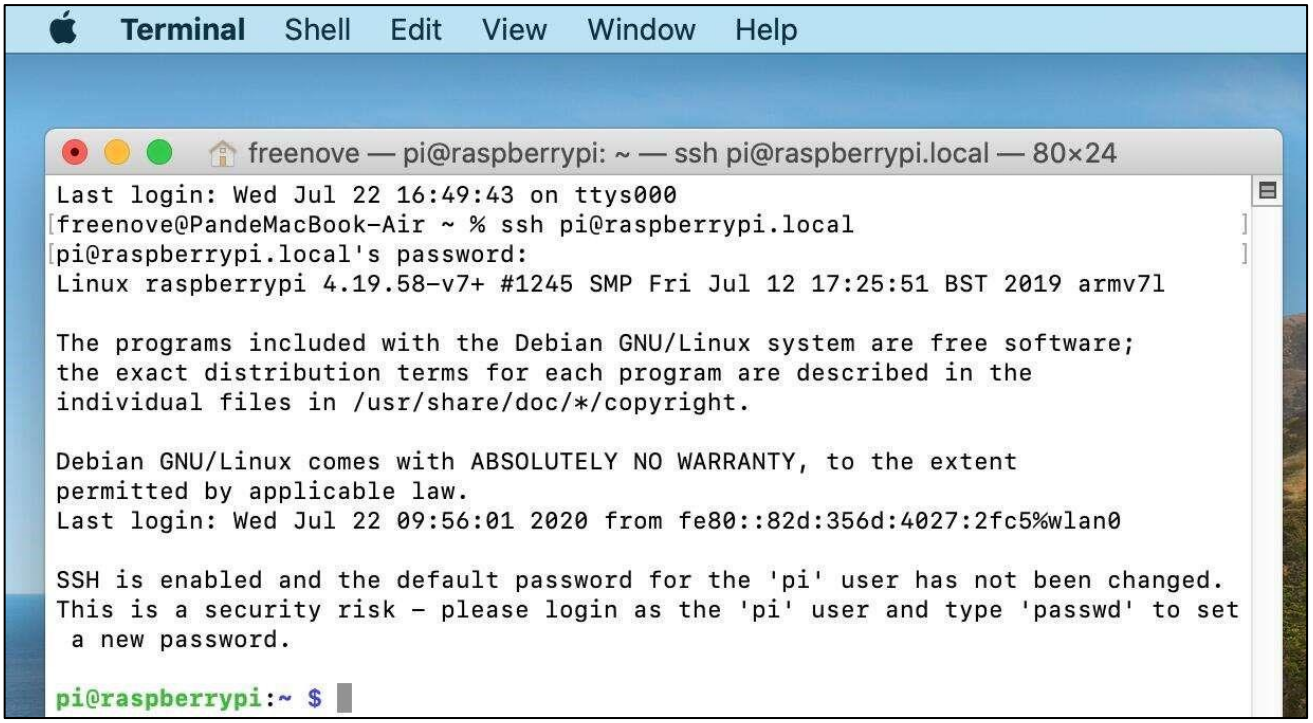
Open the terminal and type following command. **If this command doesn't work, please move to next page.**

```
ssh pi@raspberrypi.local
```

The password is **raspberry** by default, case sensitive.



You may need to type **yes** during the process.

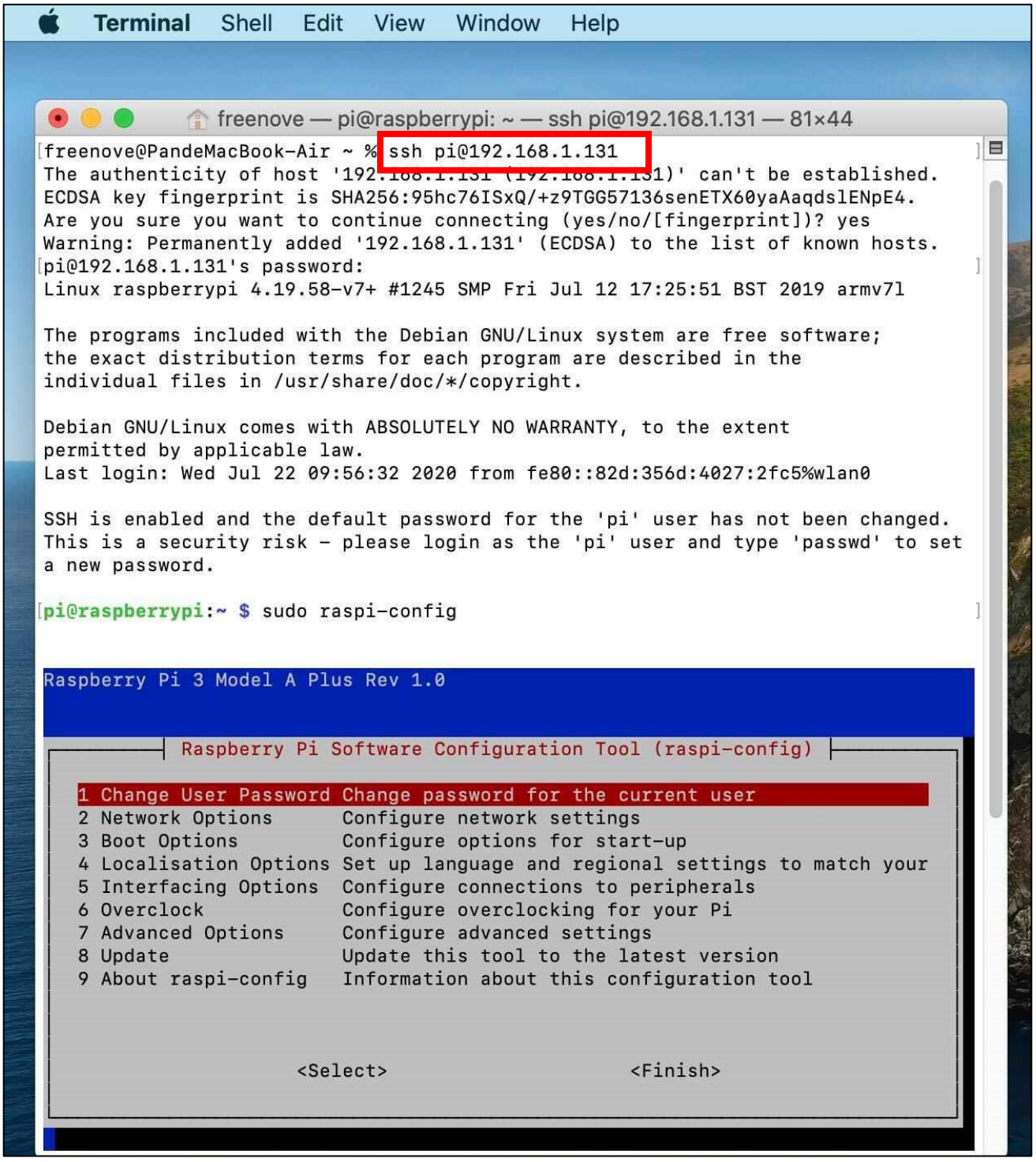


You can also use the IP address to log in Pi.
Enter **router** client to **inquiry IP address** named "raspberry pi". For example, I have inquired to **my RPi IP address, and it is "192.168.1.131".**

Open the terminal and type following command.

```
ssh pi@192.168.1.131
```

When you see pi@raspberrypi:~ $, you have logged in Pi successfully. Then you can skip to next section.



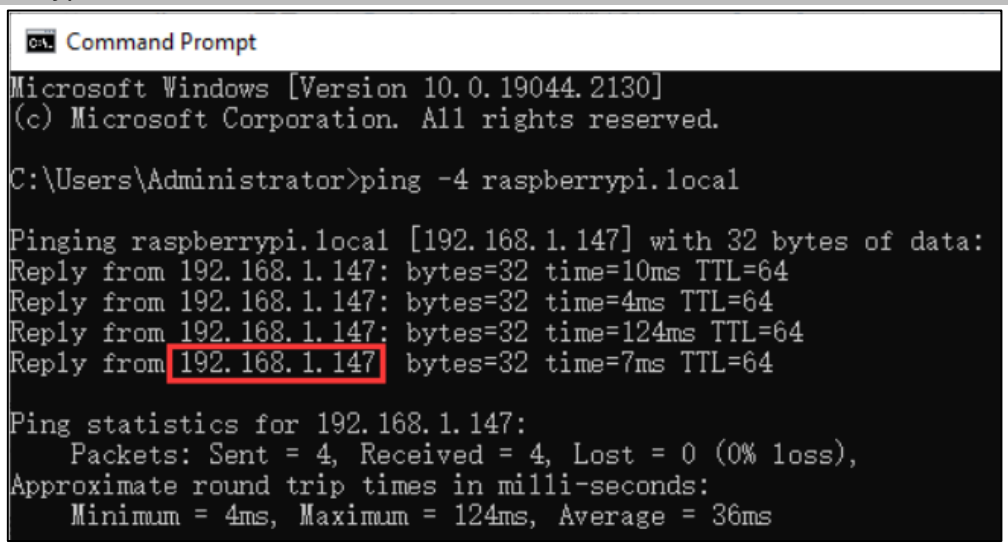
Then you can skip to VNC Viewer.

Windows OS remote desktop

If you are using win10, you can use follow way to login Raspberry Pi without desktop.

Press **Win+R**. Enter **cmd**. Then use this command to check IP:

```
ping -4 raspberrypi.local
```



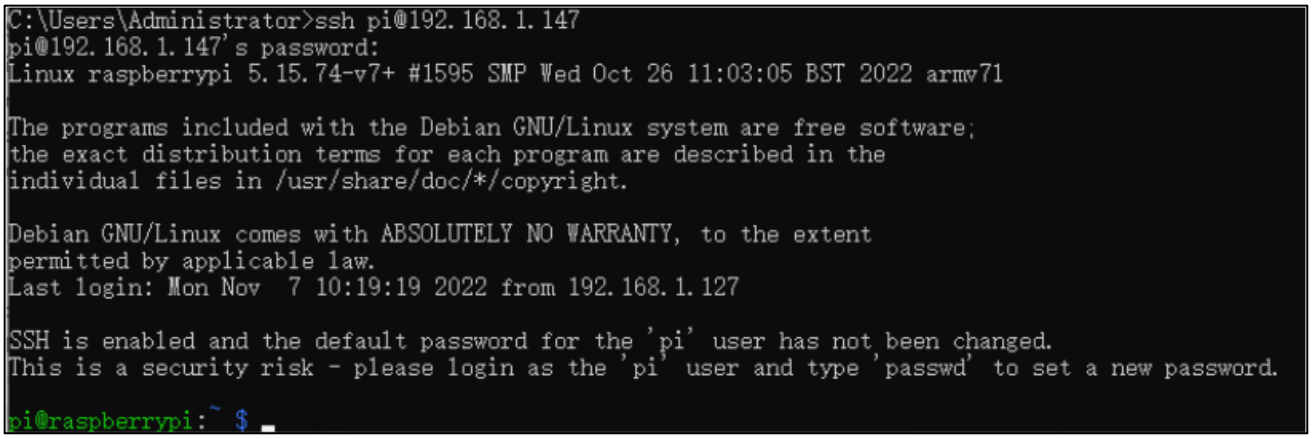
Then 192.168.1.147 is my Raspberry Pi IP.

Or enter **router** client to **inquiry IP address** named "**raspberrypi**". For example, I have inquired to **my RPi IP address**, **and it is "192.168.1.147".**

```
ssh pi@xxxxxxxxxx(IP address)
```

Enter the following command:

```
ssh pi@192.168.1.147
```

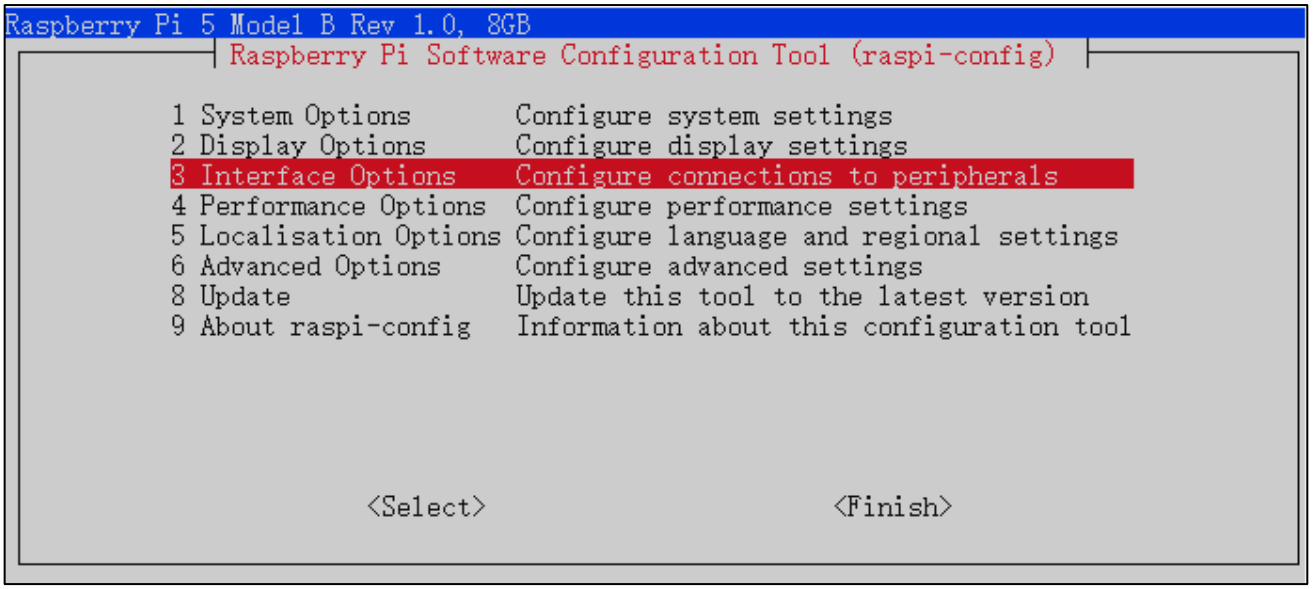


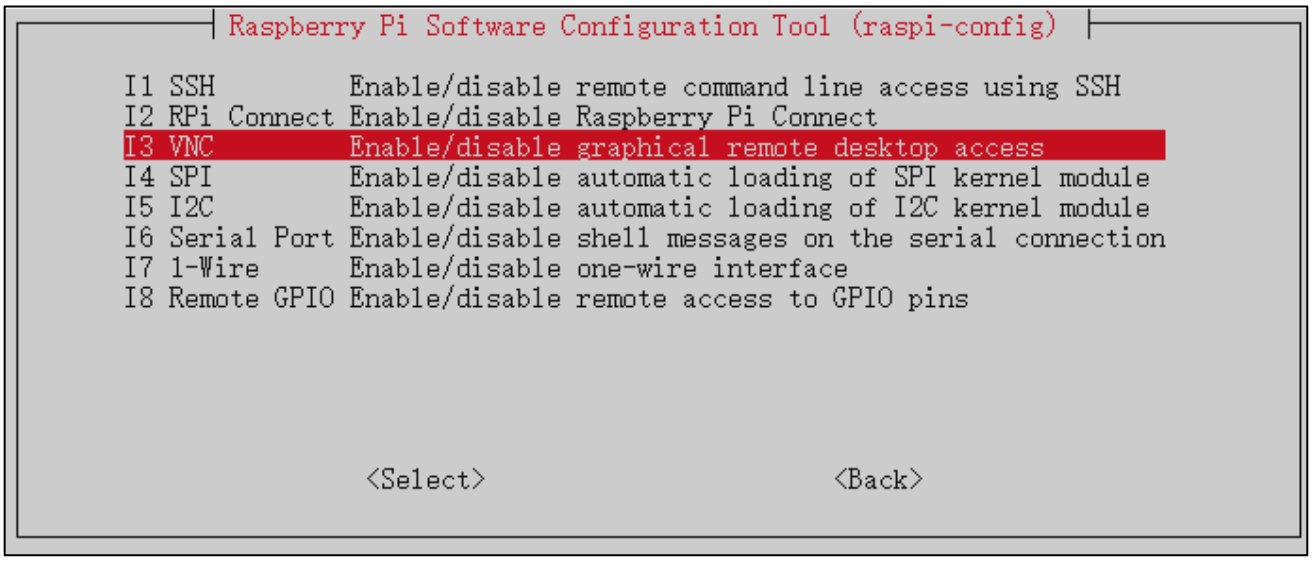
3.1.5 VNC Viewer & VNC

Enable VNC

Enter the following commands: select Interface Options -> I3 VNC -> Enter -> Yes -> OK. You may need to restart the Raspberry Pi 5 here, select OK, and then open the VNC interface.

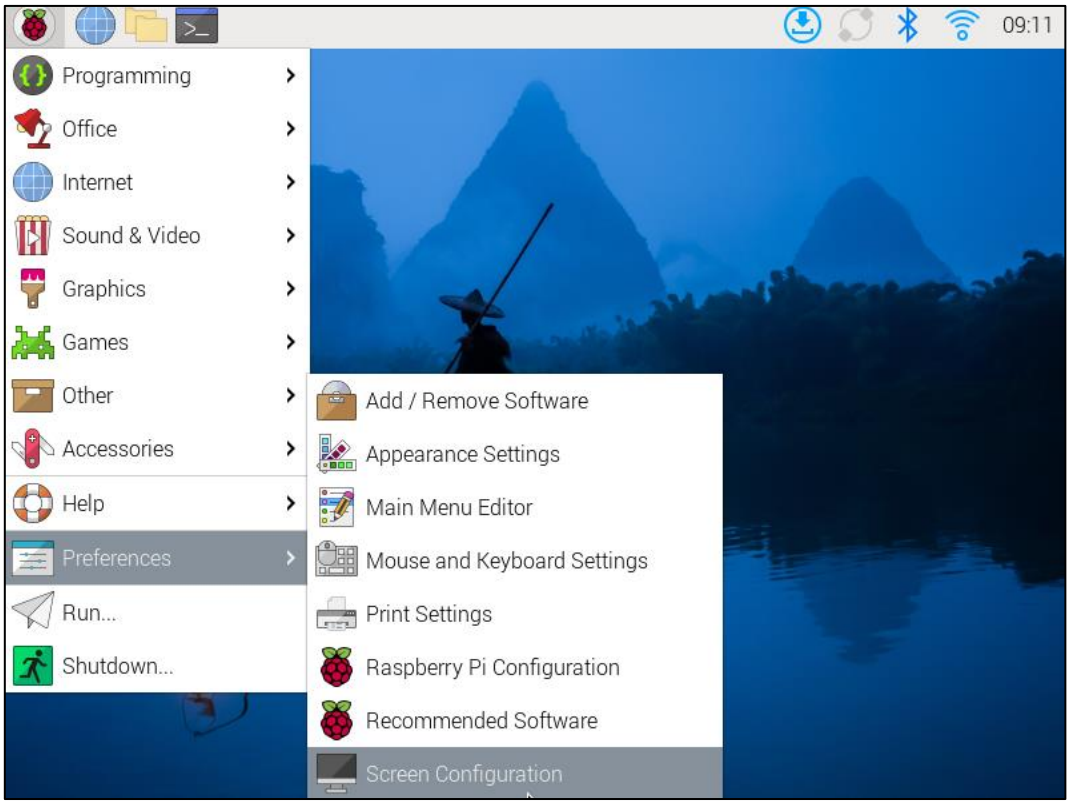
```
sudo raspi-config
```

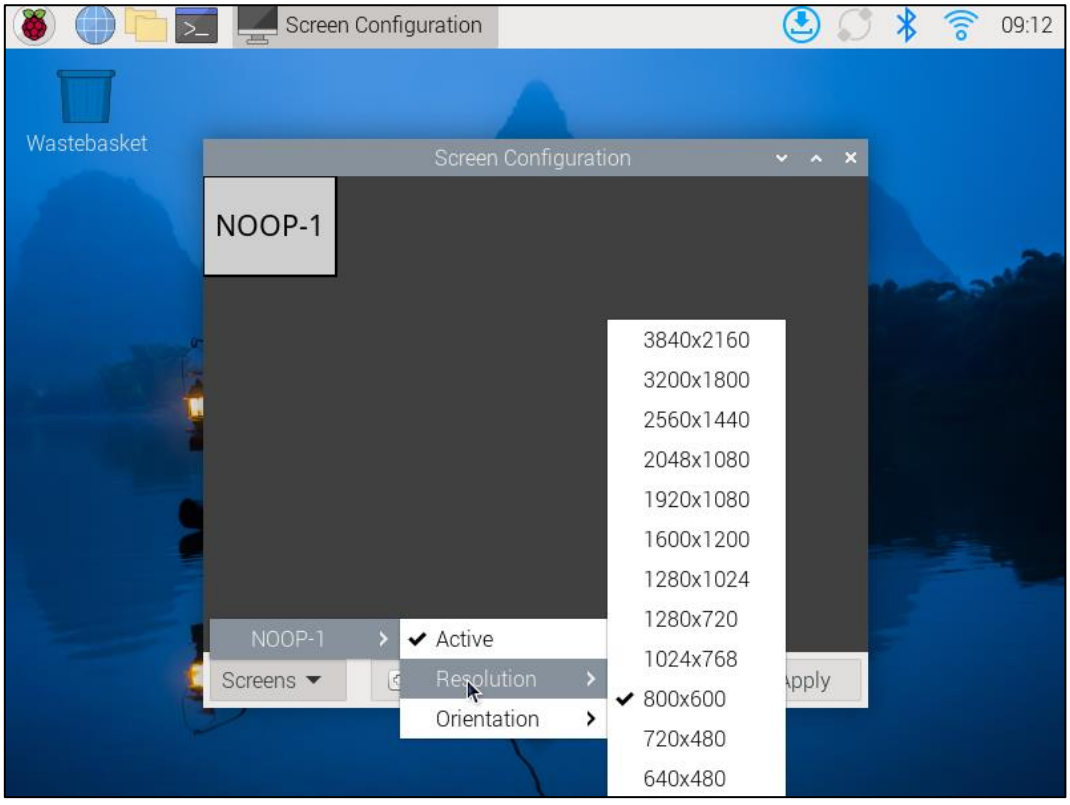




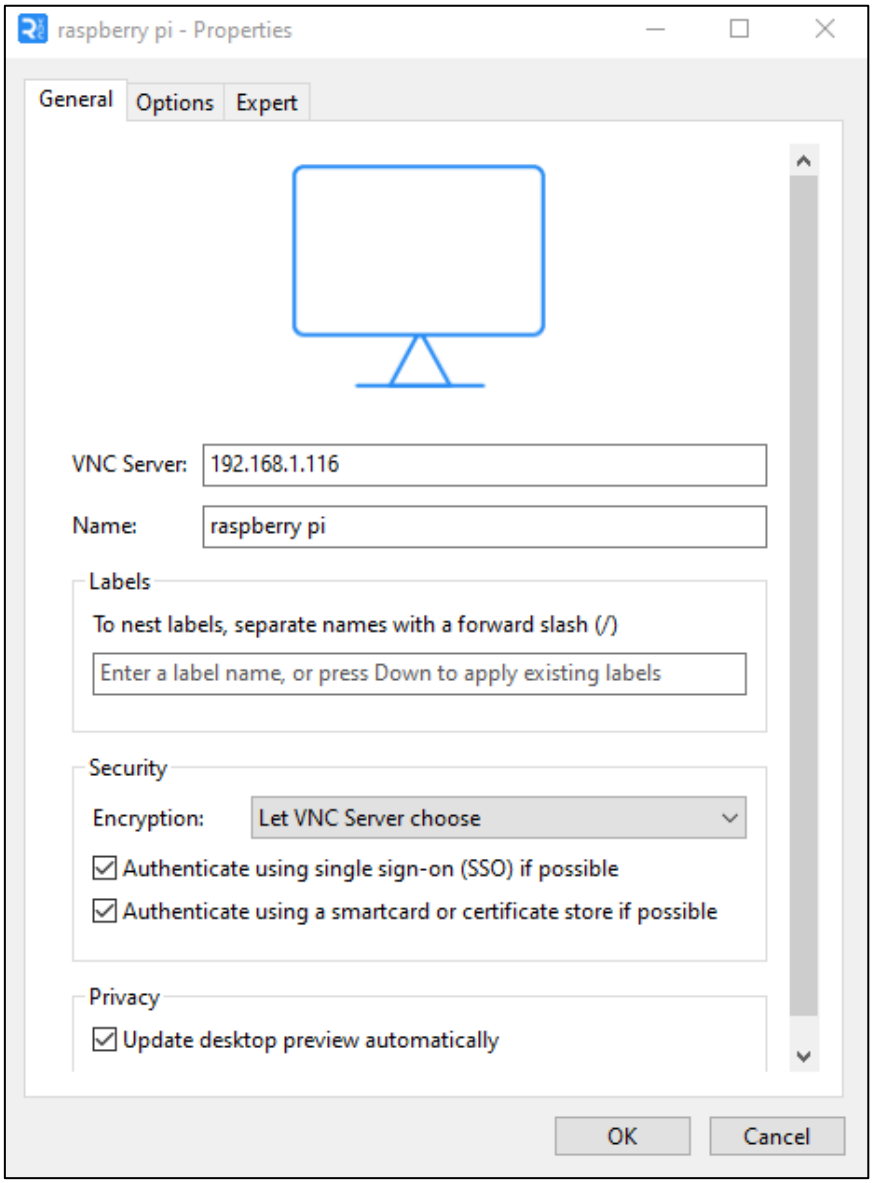
Set Resolution

You can also set other resolutions. If you don't know what to set, you can set it as 800x600 first.

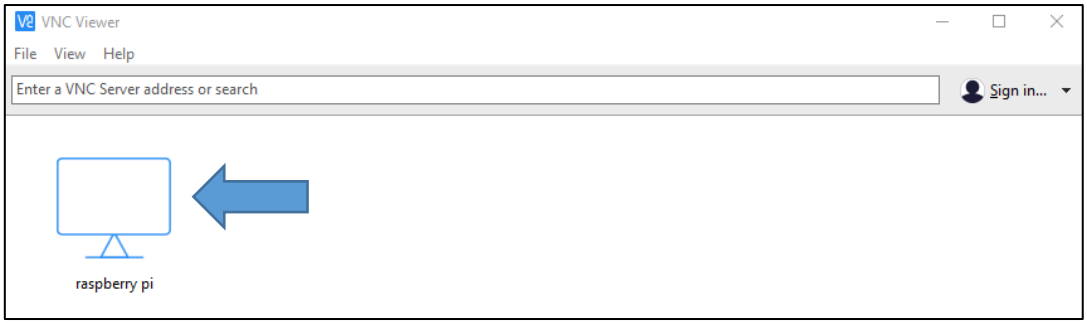




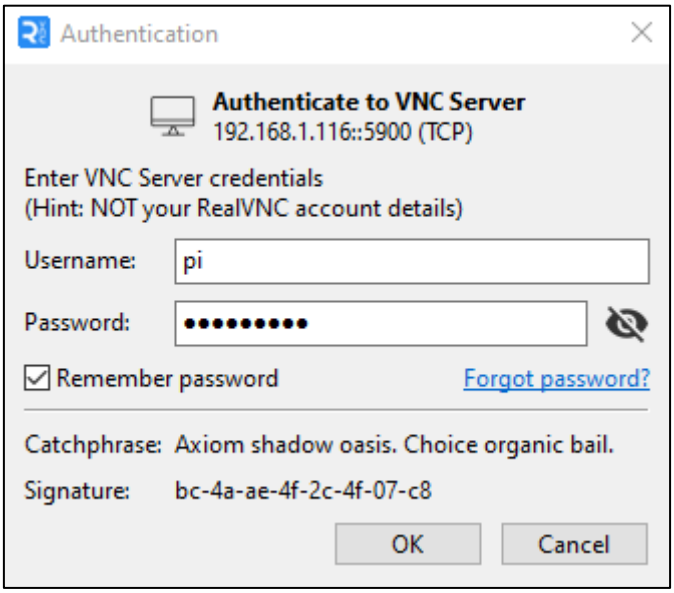
Then download and install VNC Viewer according to your computer system by click following link:
https://www.realvnc.com/en/connect/download/viewer/
After installation is completed, open VNC Viewer. And click File -> New Connection. Then the interface is shown below.



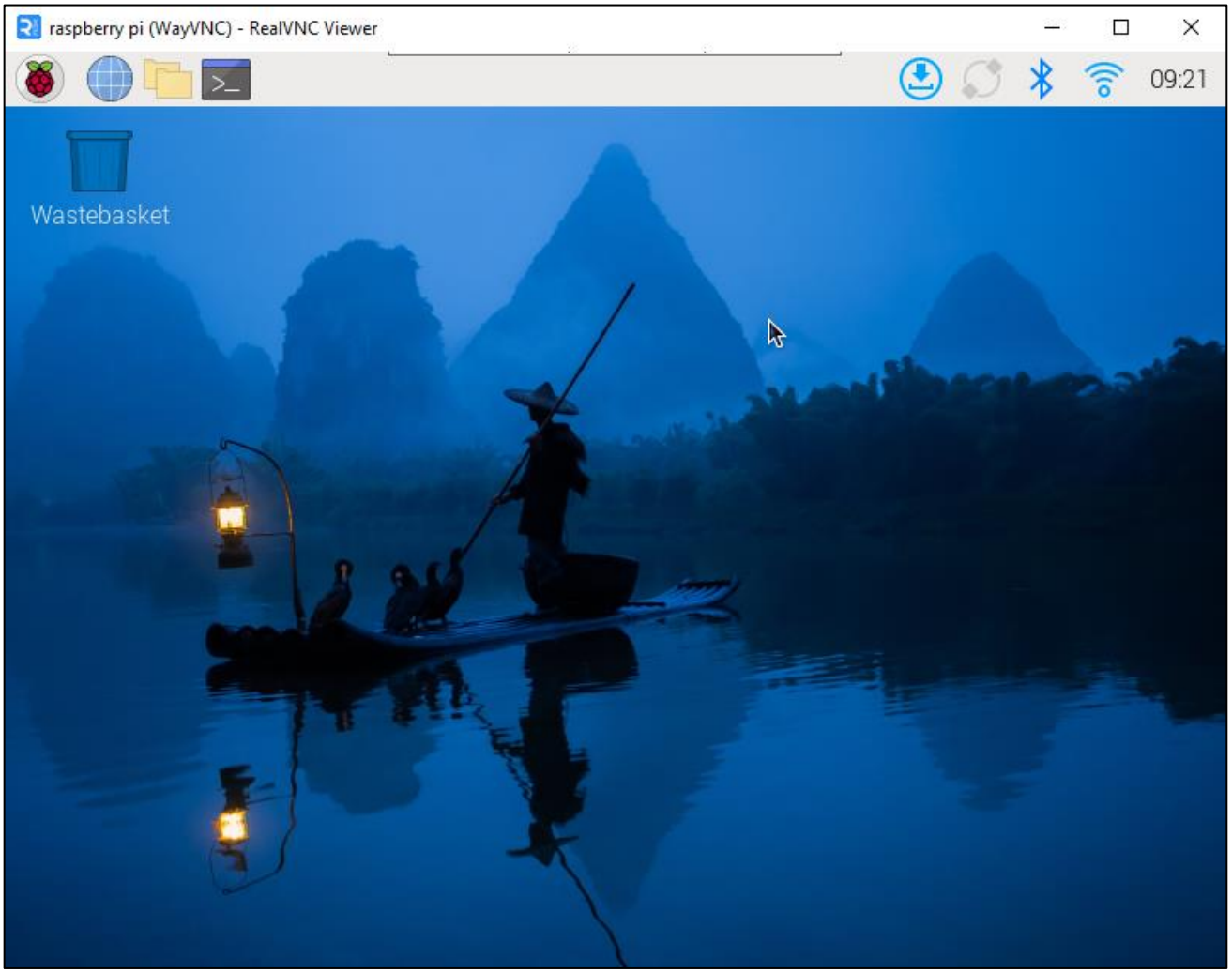
Enter ip address of your Raspberry Pi and fill in a name. Then click OK.
Then on the VNC Viewer panel, double-click new connection you just created,



Then the following dialog box pops up.

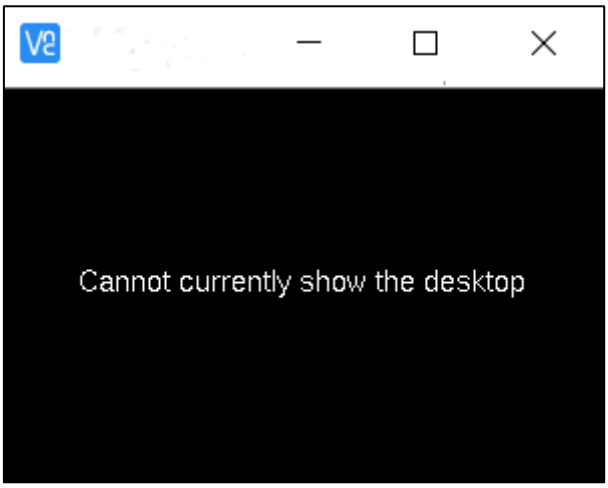


Enter username: **pi** and Password: **raspberry**. And click OK.

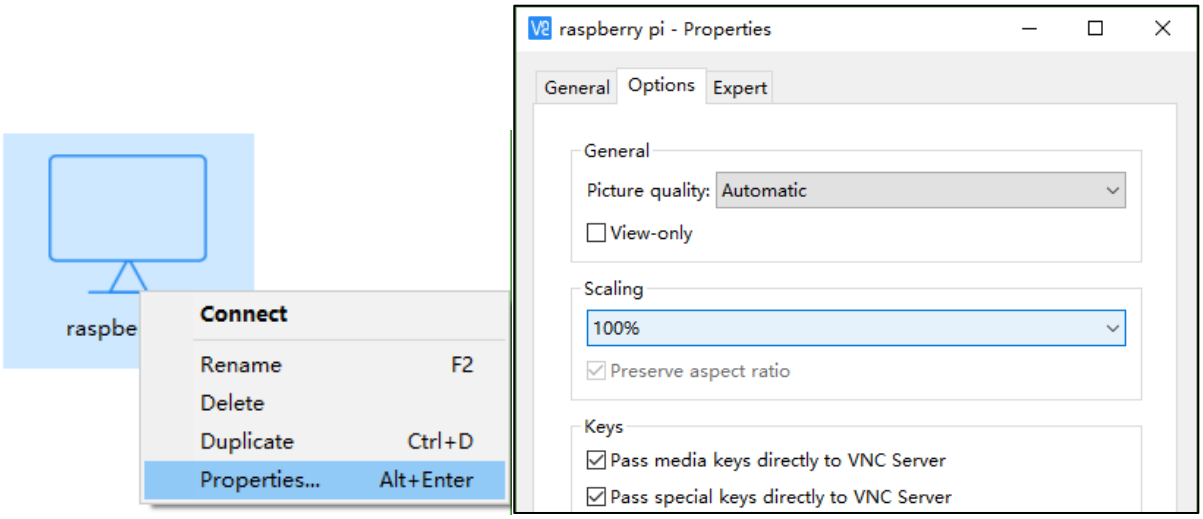


Here, you have logged in to Raspberry Pi successfully by using VNC Viewer.

If there is black window, please set another resolution.



In addition, your VNC Viewer window may zoom your Raspberry Pi desktop. You can change it. On your VNC View control panel, click right key. And select Properties->Options label->Scaling. Then set proper scaling.



Here, you have logged in to Raspberry Pi successfully by using VNC Viewer and operated proper setting.

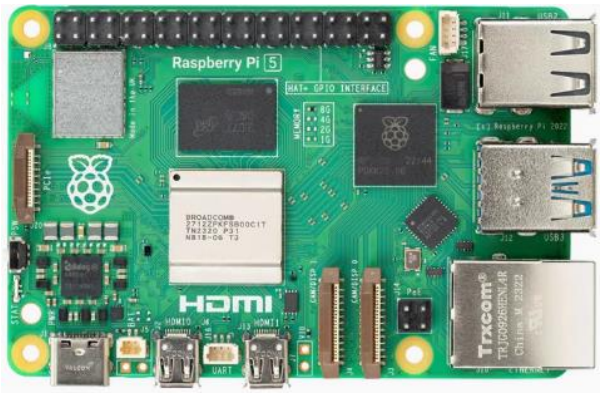
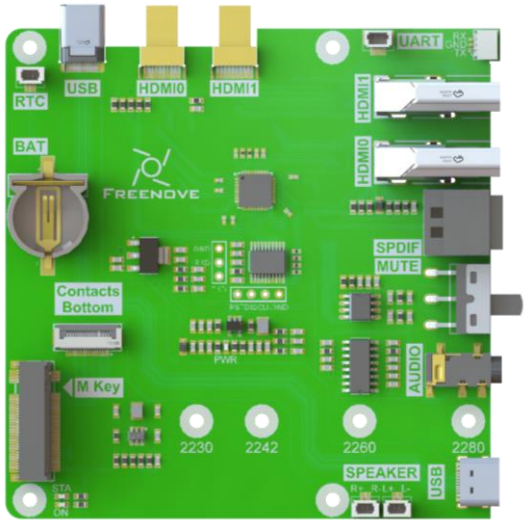
Raspberry Pi 5, Raspberry Pi 4B/3B+/3B integrates a Wi-Fi adaptor.If you did not connect Pi to WiFi. You can connect it to wirelessly control the robot.

3.2 Flashing OS to NVMe SSD

This step is to install the Raspberry Pi OS into the NVME SSD.

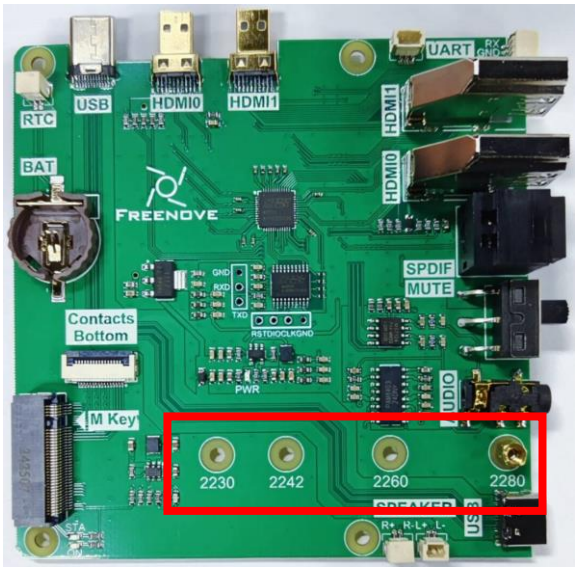
Component List

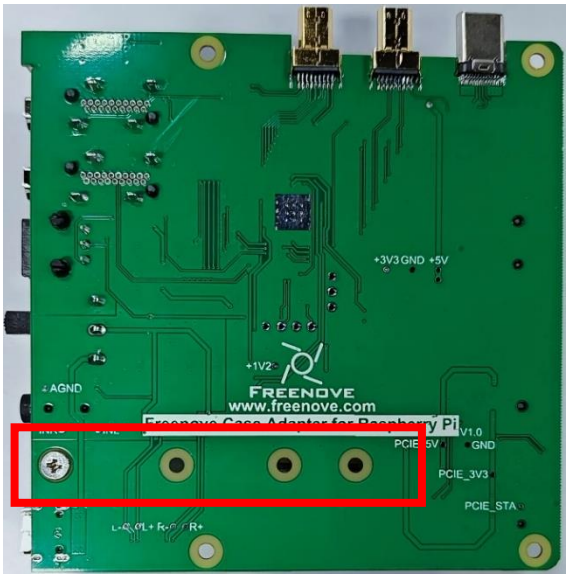
Required Components

<table>
<tr><td>Raspberry Pi 5 x 1
  </td><td>27W power adapter x1 (or a power adapter compatible with Raspberry Pi that can output 5.1V/5A officially)
  </td></tr>
<tr><td>Case Adapter Board x 1
  </td><td>NVMe SSD x 1
  </td></tr>
<tr><td>M2.5x3 Screws x 2
  </td><td>M2.5x5 Brass Standoff x 1
  </td></tr>
<tr><td colspan="2">FPC soft line-0.5-16P-15cm (same direction) x 1
  </td></tr>
</table>

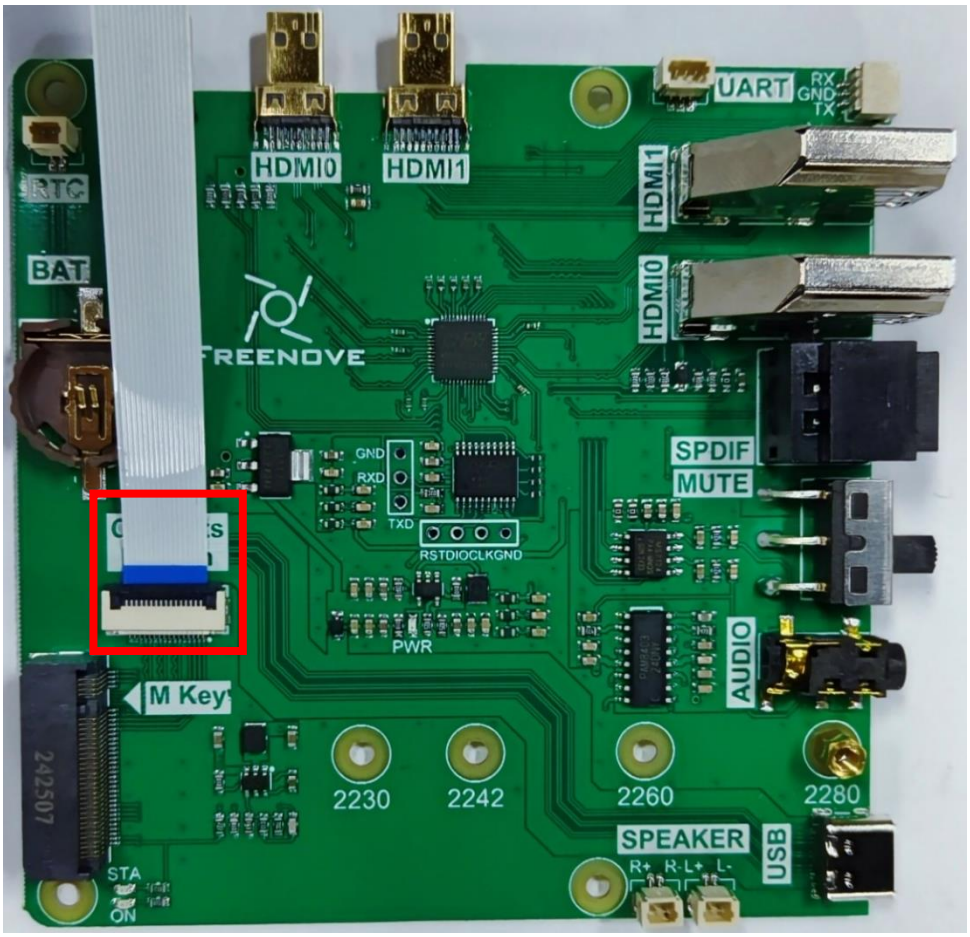
3.2.1 Assembly and Wiring

1. Install the mounting standoff for the NVMe SSD from the underside. Select and secure them in the suitable holes according to the length of your SSD.

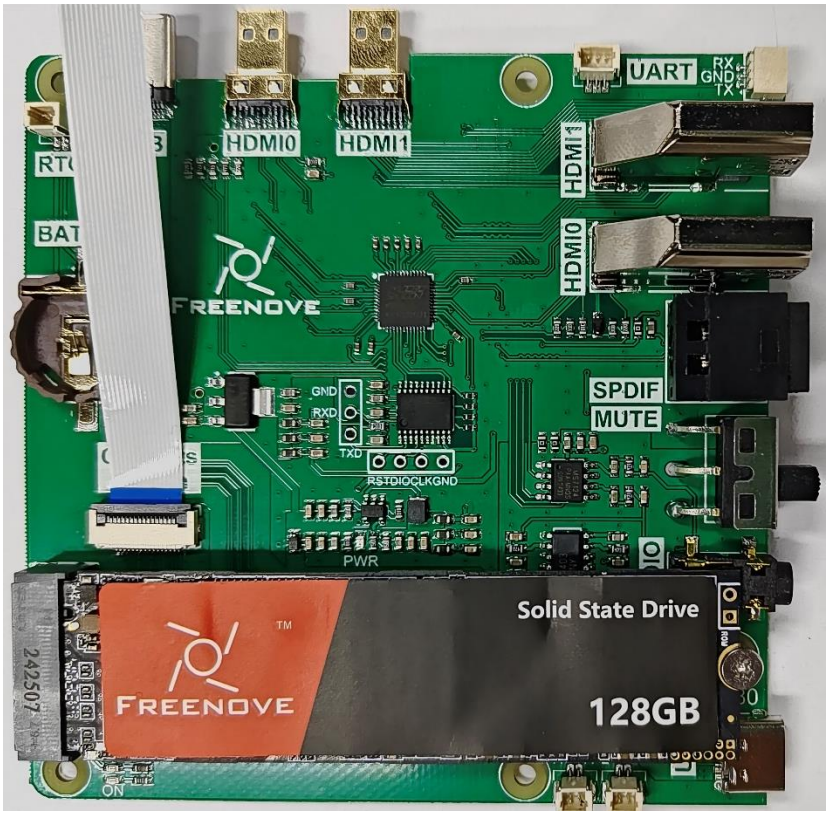




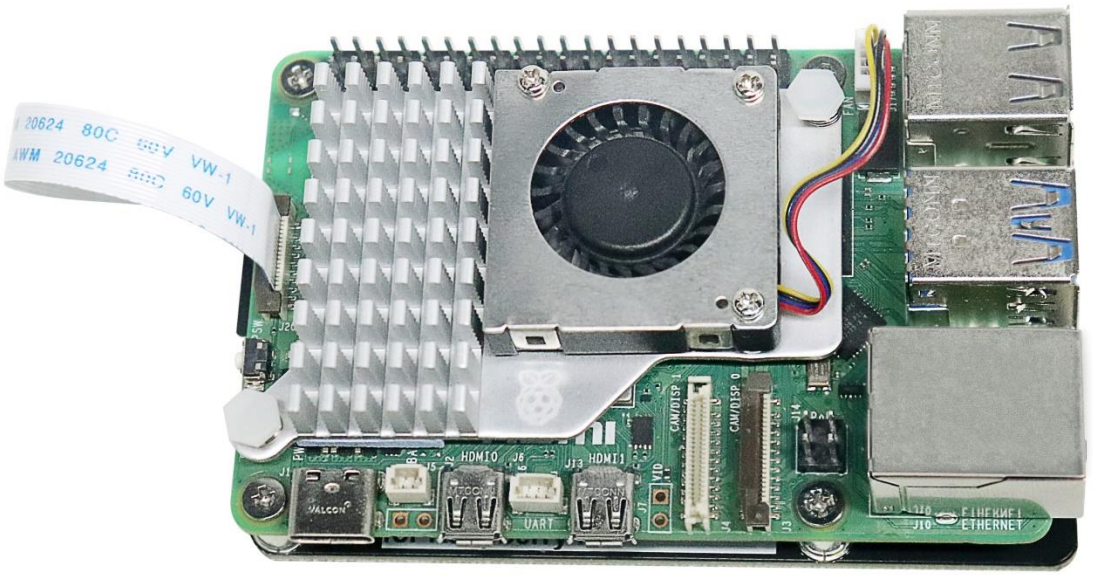
2. Connect the NVMe SSD cable to the FPC socket. Please note that the contacts of the NVMe SSD cable should face downward.



3. Tilt the NVMe SSD and insert it into the M.2 interface. Then, use M2.5x3 screws to secure the NVMe SSD to the standoff.



Connect the other end of the cable to Raspberry Pi 5.



3.2.2 SSD Detection

(Note: Not all SSDs are supported by Pi5.)

Run the following command in the Terminal to check whether SSD is detected.

Note that different SSDs display different content.

lspci

```
pi@raspberrypi:~ $ lspci
0000:00:00.0 PCI bridge: Broadcom Inc. and subsidiaries Device 2712 (rev 21)
0000:01:00.0 Non-Volatile memory controller: Silicon Motion, Inc. SM2263EN/SM2263XT SSD Controller (rev 03)
0001:00:00.0 PCI bridge: Broadcom Inc. and subsidiaries Device 2712 (rev 21)
0001:01:00.0 Ethernet controller: Device 1de4:0001
```

lsblk

```
pi@raspberrypi:~ $ lsblk
NAME        MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
mmcblk0     179:0    0  29.8G  0 disk
├─mmcblk0p1 179:1    0   512M  0 part /boot/firmware
└─mmcblk0p2 179:2    0  29.3G  0 part /
nvme0n1     259:0    0 476.9G  0 disk
pi@raspberrypi:~ $
```

As shown in the figure above, the device 'nvme0n1' with a capacity of 476.9GBytes shows up, indicating that the SSD has been correctly recognized. The detected capacity will depend on the size of your SSD. If your drive has been previously partitioned, you may also see some partition information displayed.

Please note: Installing the system will format the SSD, erasing all data. If necessary, please back up any data on your SSD before proceeding.

3.2.3 Enable PCIE3.0 (on OS written into SD Card)

If the SSD you received is with **Phison** controller, you may need to enable PCIE 3.0 .(This step is strongly recommended; without this step, the later process may fail.)

If it is not with Phison controller, you do not need to enable PCIE 3.0. You may skip this section.

Run the command lspci to check the controller.

```
pi@raspberrypi:~ $ lspci
0000:00:00.0 PCI bridge: Broadcom Inc. and subsidiaries BCM2712 PCIe Bridge (rev 21)
0000:01:00.0 Non-Volatile memory controller: Phison Electronics Corporation PS5021-E21
 PCIe4 NVMe Controller (DRAM-less) (rev 01)
0001:00:00.0 PCI bridge: Broadcom Inc. and subsidiaries BCM2712 PCIe Bridge (rev 21)
0001:01:00.0 Ethernet controller: Raspberry Pi Ltd RP1 PCIe 2.0 South Bridge
```

(The above screenshot is the result of an 128GB SSD with Phison as main controller.)

Enable PCIe Gen3.0

Add the line dtparam=pciex1_gen=3 to /boot/firmware/config.txt to enable PCIe Gen3.0.

As shown below, enter the command to open the file.

```
sudo nano /boot/firmware/config.txt
```

Add the line dtparam=pciex1_gen=3 to the end of the file, as shown below:

```
 GNU nano 7.2                  /boot/firmware/config.txt *
otg_mode=1

[all]

dtparam=pciex1_gen=3
```

Press Ctrl-O to save the file, Enter to confirm, and Ctrl-X to exit.

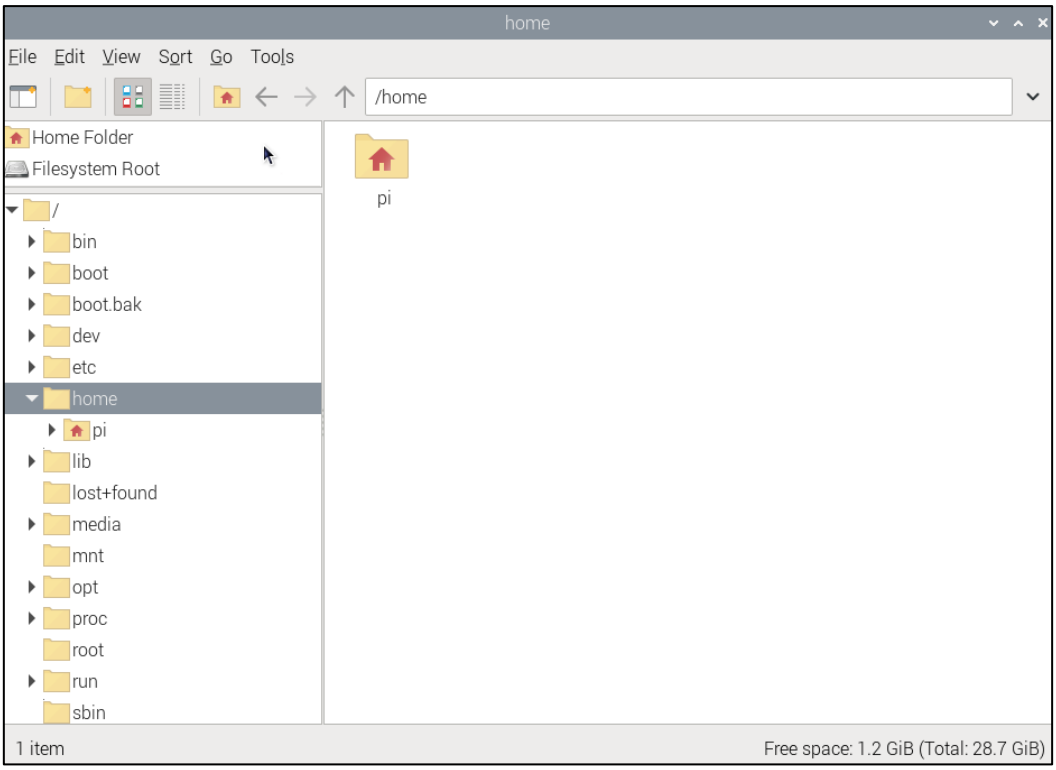
Reboot your Raspberry Pi.

```
sudo reboot
```

3.2.4 SSD Partitioning and Formatting

This step is not a must-do, but it can further test whether the SSD perform normally on Raspberry Pi to ensure smooth performance in later steps.

At this point, the hard drive cannot be seen in the file manager as the disk has not been partitioned yet.



Install a disk management tool with the following command:

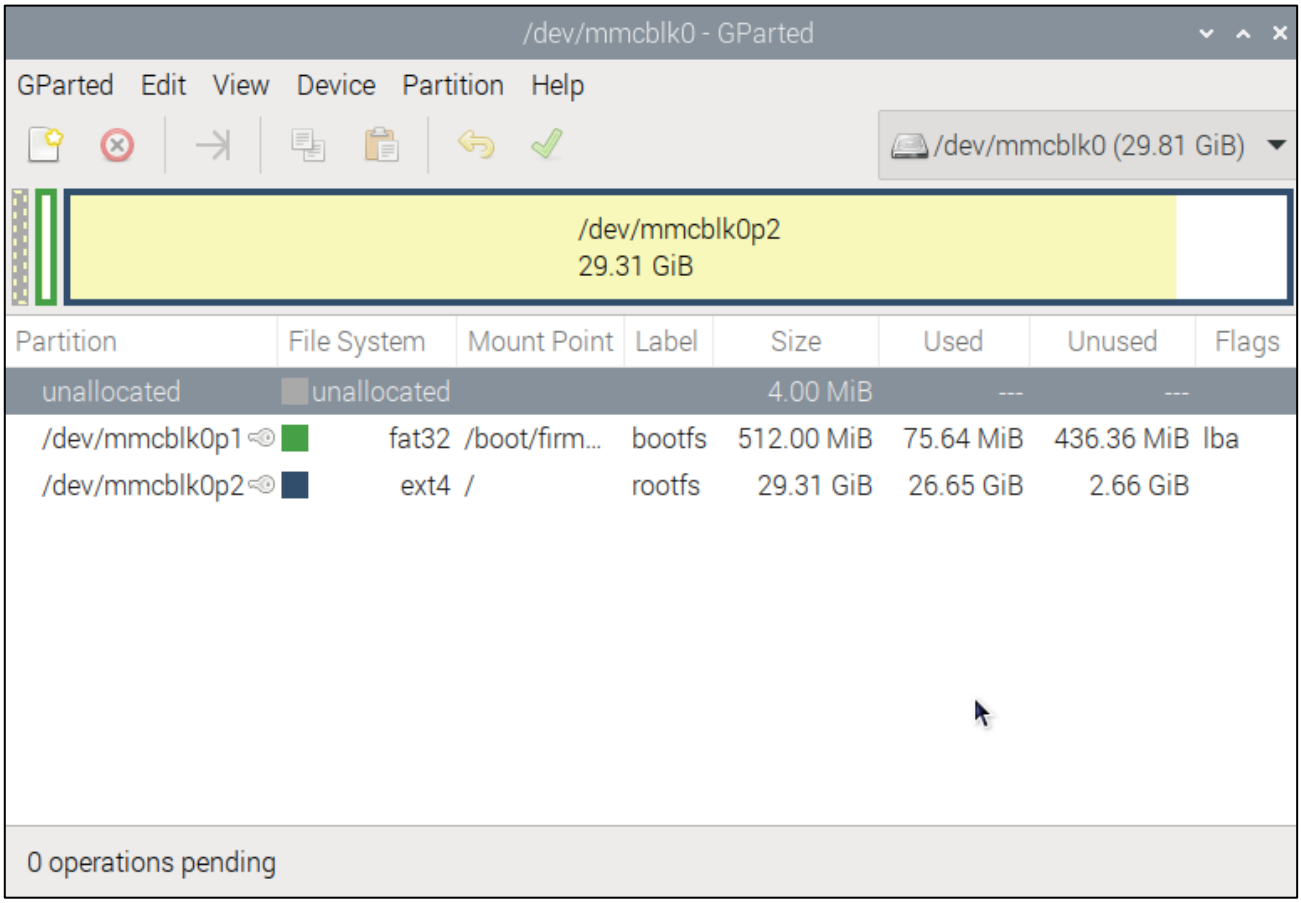
```
sudo apt-get install gparted
```

```
The following NEW packages will be installed:
  gparted gparted-common
0 upgraded, 2 newly installed, 0 to remove and 164 not upgraded.
Need to get 772 kB/2,483 kB of archives.
After this operation, 8,638 kB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://deb.debian.org/debian bookworm/main arm64 gparted arm64 1.3.1-1 [77
2 kB]
Fetched 772 kB in 0s (2,579 kB/s)
Selecting previously unselected package gparted-common.
(Reading database ... 222757 files and directories currently installed.)
Preparing to unpack .../gparted-common_1.3.1-1_all.deb ...
Unpacking gparted-common (1.3.1-1) ...
Selecting previously unselected package gparted.
Preparing to unpack .../gparted_1.3.1-1_arm64.deb ...
Unpacking gparted (1.3.1-1) ...
Setting up gparted-common (1.3.1-1) ...
Setting up gparted (1.3.1-1) ...
Processing triggers for mailcap (3.70+nmu1) ...
Processing triggers for desktop-file-utils (0.26-1) ...
Processing triggers for hicolor-icon-theme (0.17-2) ...
Processing triggers for gnome-menus (3.36.0-1.1) ...
Processing triggers for man-db (2.11.2-2) ...
pi@raspberrypi:~ $
```

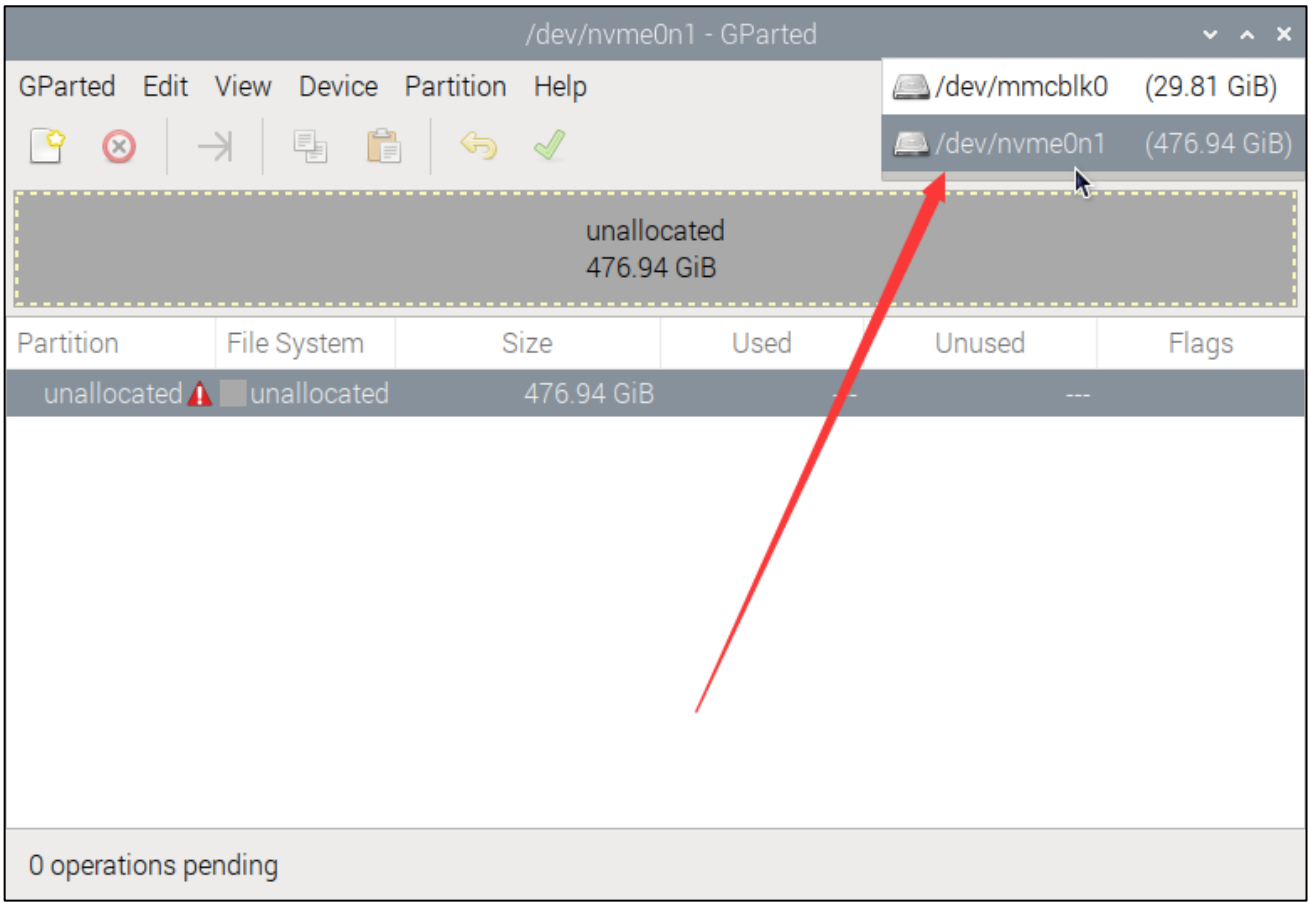
Open gparted with the command:

```
sudo gparted
```

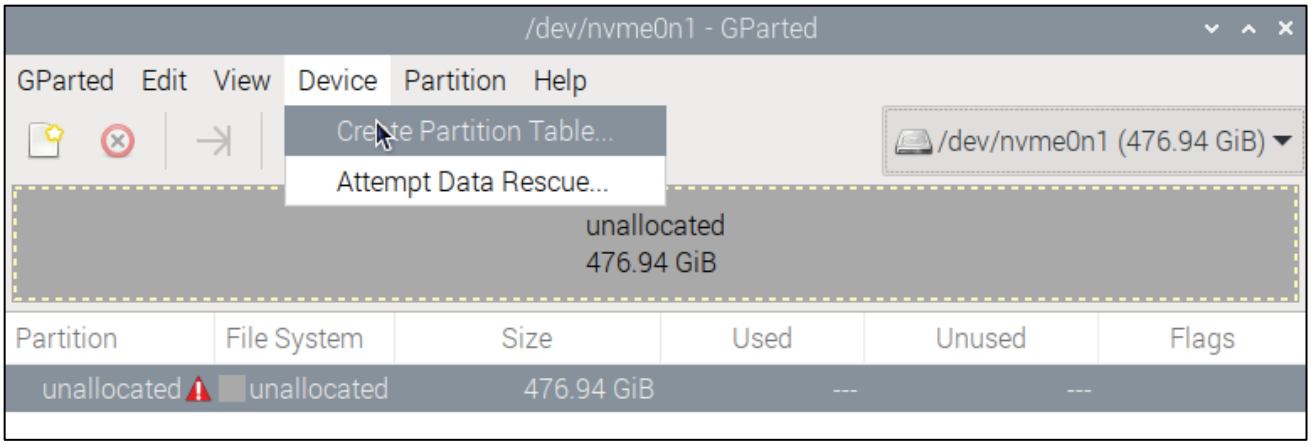
```
pi@raspberrypi:~ $ sudo gparted
error: XDG_RUNTIME_DIR is invalid or not set in the environment.
GParted 1.3.1
configuration --enable-libparted-dmraid --enable-online-resize
libparted 3.5
/dev/nvme0n1: unrecognised disk label
```



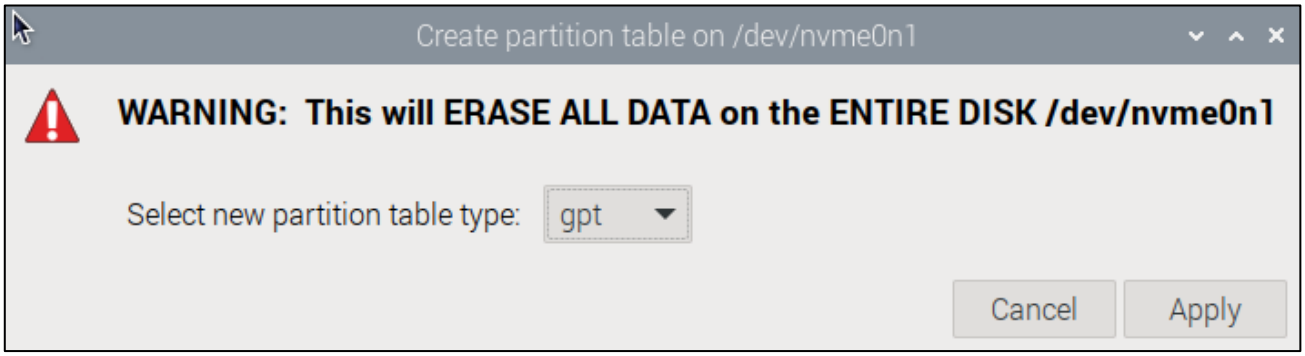
Click on the dropdown menu in the upper right corner and switch to NVME SSD.



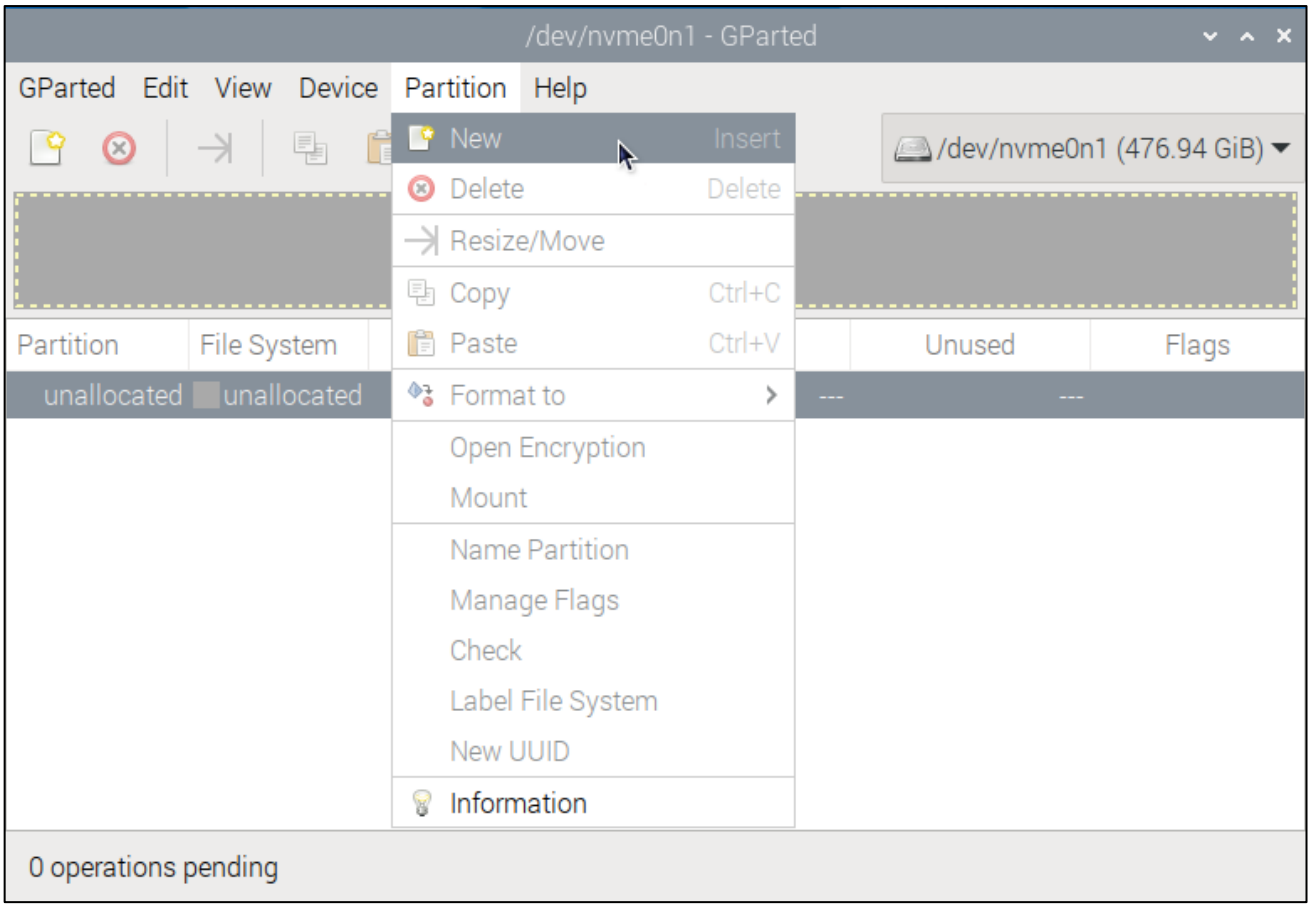
Click Device on the menu bar and select Create Partition Table.



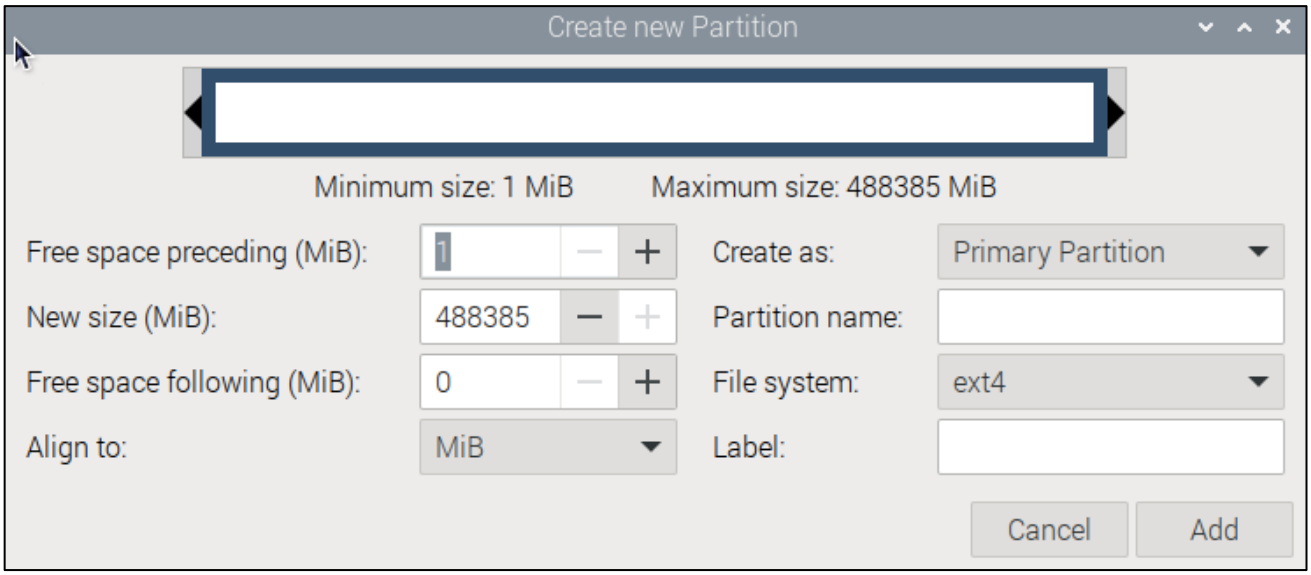
You will see the prompt that data will be erased. It is recommended to select gpt for partition table type. Click Apply.



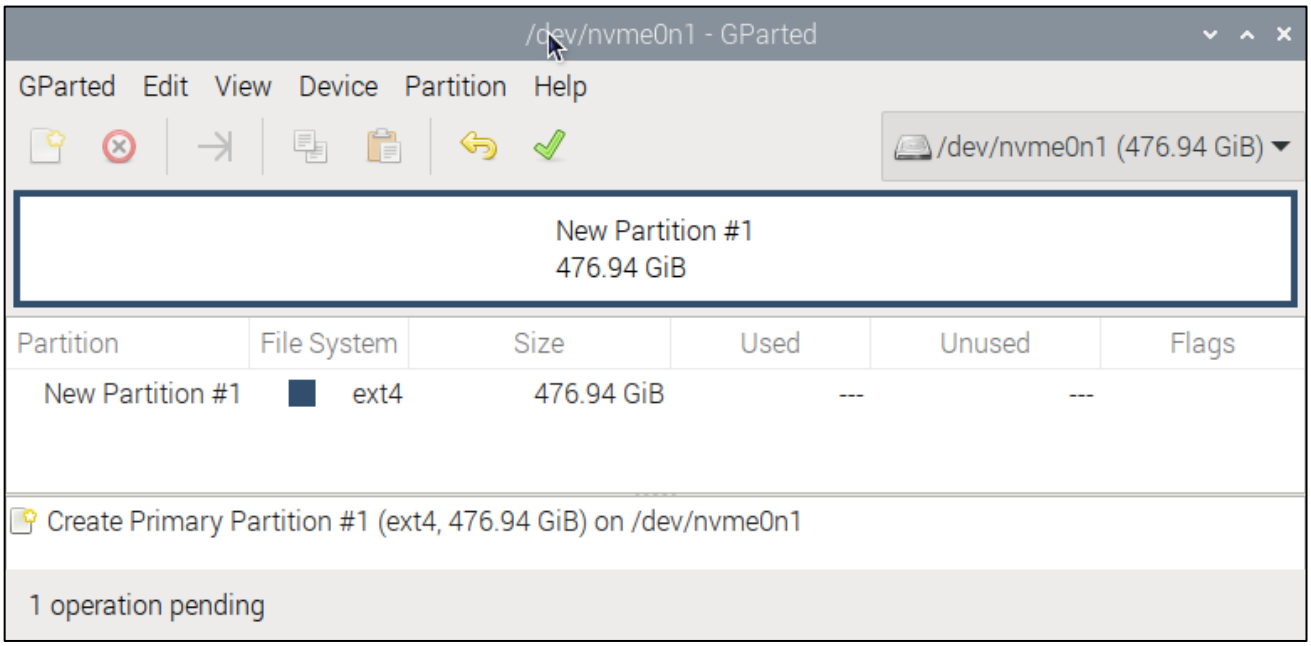
Click Partition on the menu bar, choose New.



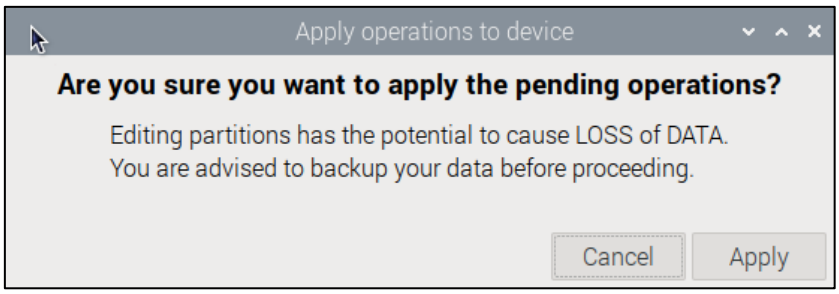
As shown in the figure below, the size of partition can be adjusted by dragging the mouse left and right, or by entering the size directly. The other options can be left as default setting. Here, we allocate all the capacity to a single partition. Click on Add.



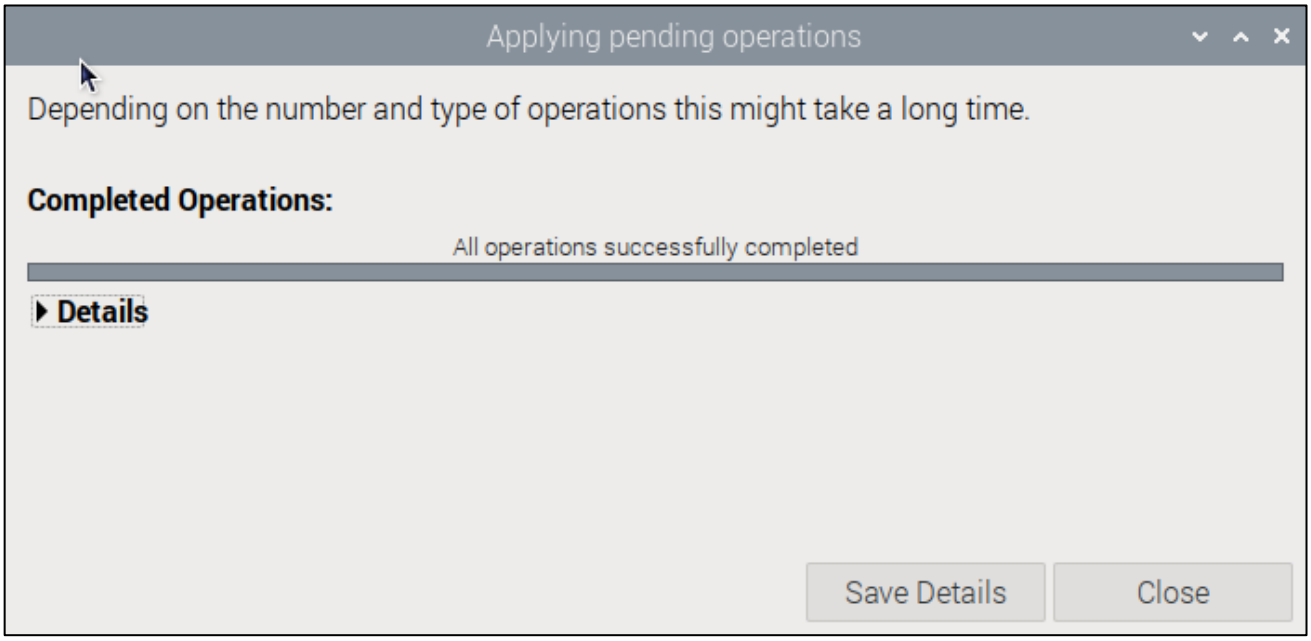
Click the check icon  to save the partition just built, as illustrated below.



Click on Apply.



Wait for it to complete and click on Close.

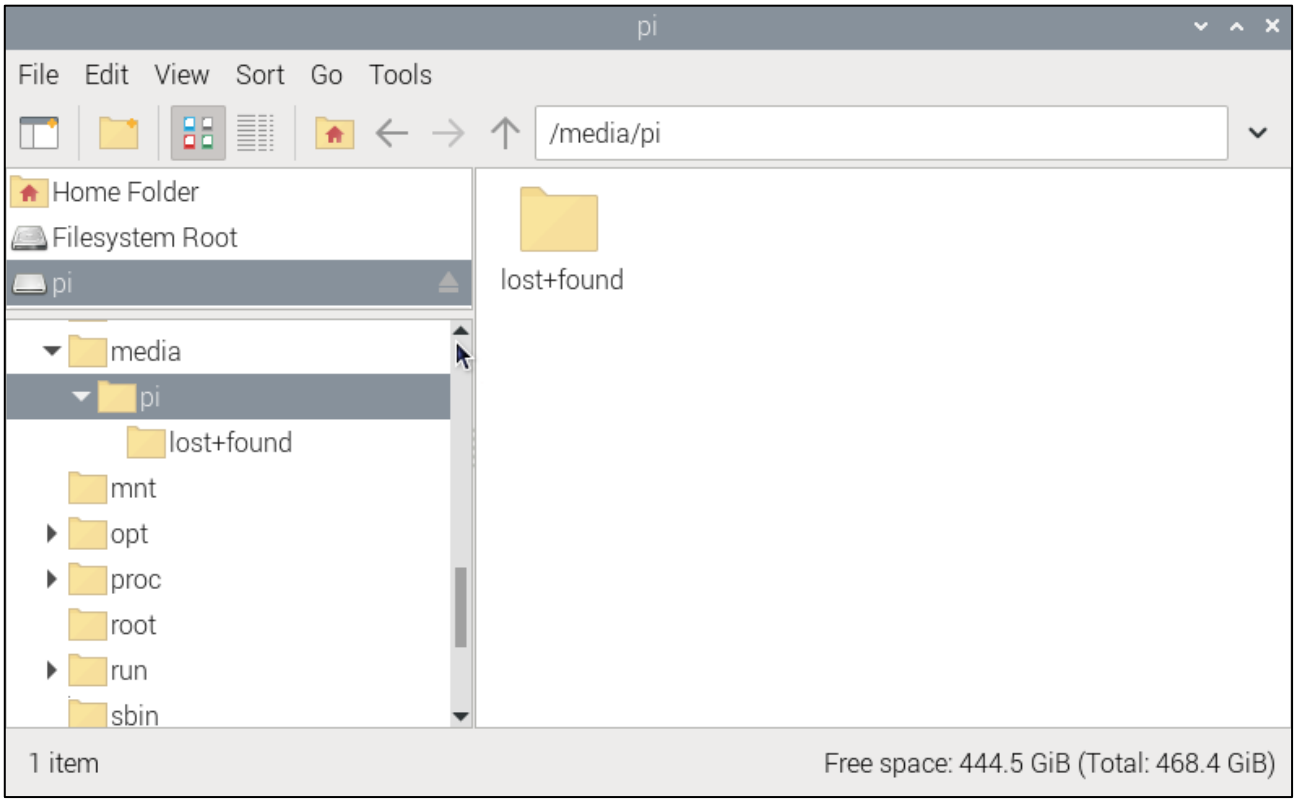


At this point, you can mount the disk using the mount command and then access the disk space through the file manager. Use the following command to mount the SSD:

```
mkdir pi
sudo mount /dev/nvme0n1p1 /media/pi
```

```
pi@raspberrypi:~ $ mkdir pi
pi@raspberrypi:~ $ sudo mount /dev/nvme0n1p1 /media/pi
```

Open the file manager, as shown below.



If you plan to use the SSD as a standard storage device, you can conclude the process here. However, if you want to further proceed with installing an operating system on the SSD, please read on.

3.2.5 Flash the OS

Install the OS to SSD with the method similar to that in the previous section on installing a system onto an SD card. This time, operate on the Raspberry Pi.

Install rpi-imager with the following command:

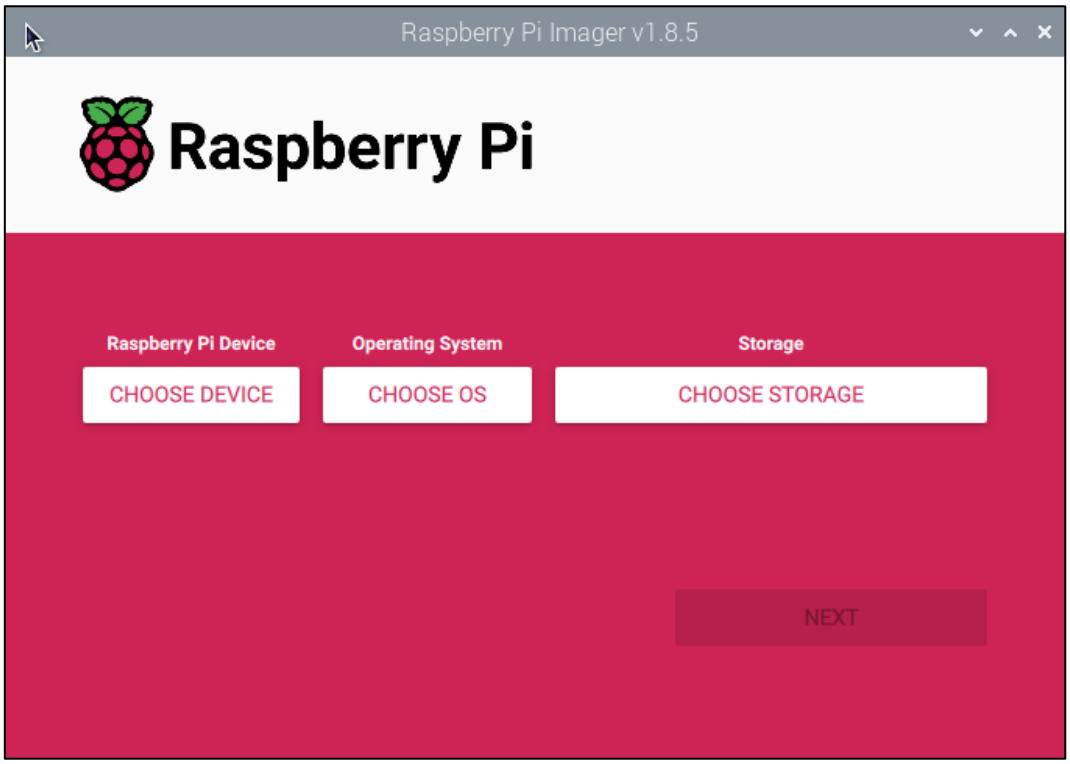
```
sudo apt install rpi-imager
```

```
pi@raspberrypi:~ $ sudo apt install rpi-imager
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
rpi-imager is already the newest version (1.8.5+rpt1).
0 upgraded, 0 newly installed, 0 to remove and 164 not upgraded.
pi@raspberrypi:~ $
```

Open rpi-imager:

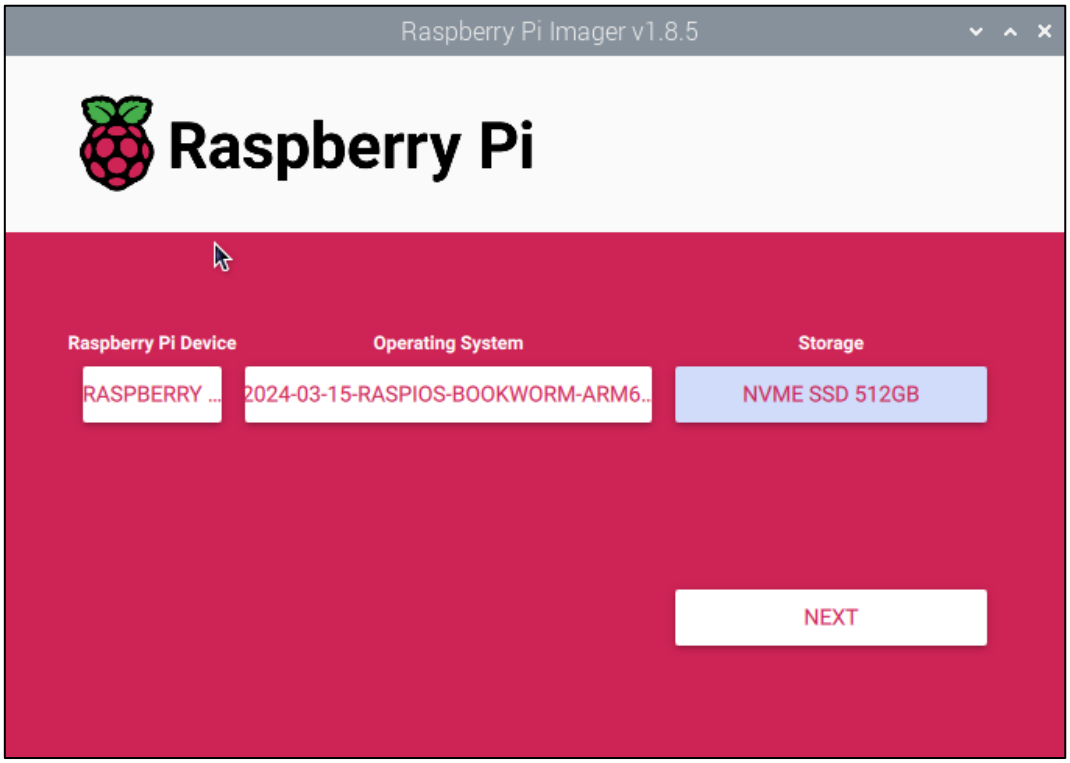
```
sudo rpi-imager
```

```
pi@raspberrypi:~ $ sudo rpi-imager
QStandardPaths: XDG_RUNTIME_DIR not set, defaulting to '/tmp/runtime-root'
error: XDG_RUNTIME_DIR is invalid or not set in the environment.
```

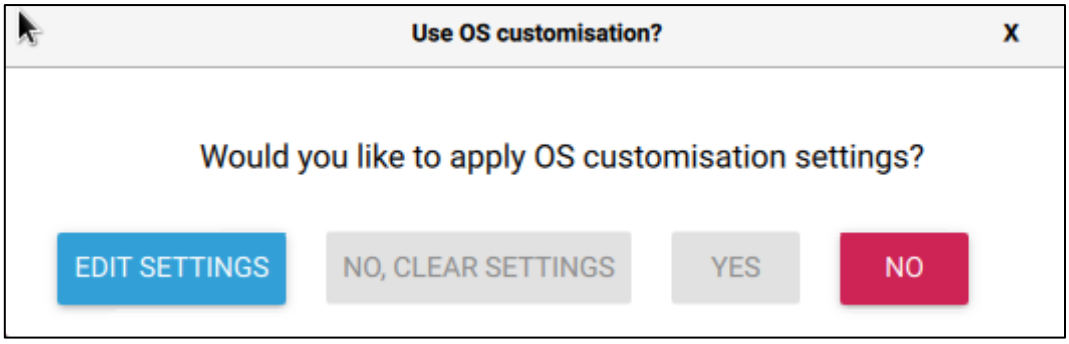


By this point, you should be quite familiar with the process.

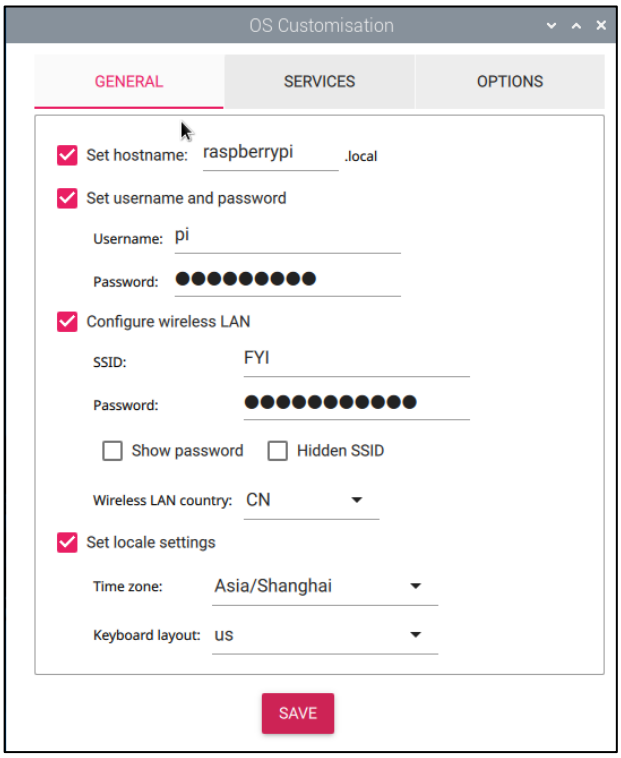
Select the Raspberry Pi 5 as your device and choose either an online download or an offline file for the operating system; in this case, an offline file is selected. (It is recommended to use a 64-bit Raspberry Pi system with recommended software). Choose your NVME SSD as the storage device. Click NEXT.

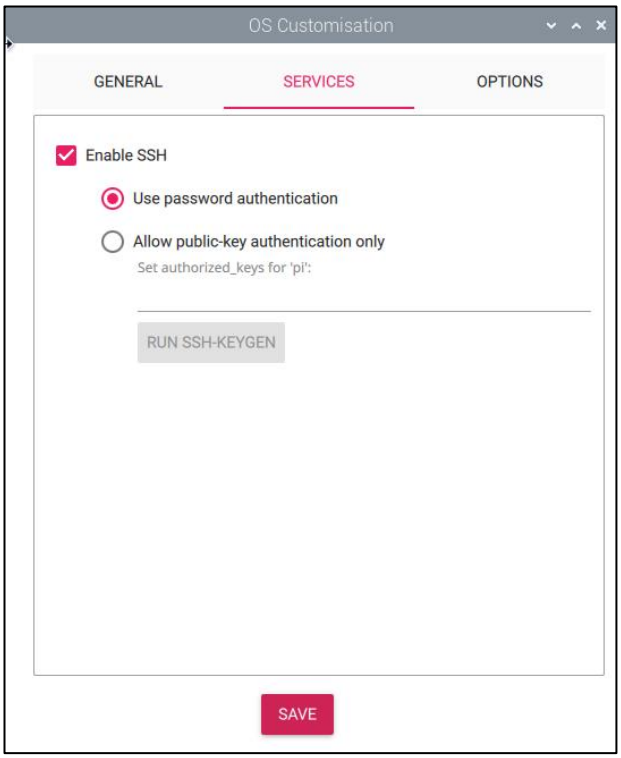


Click on EDIT SETTINGS.

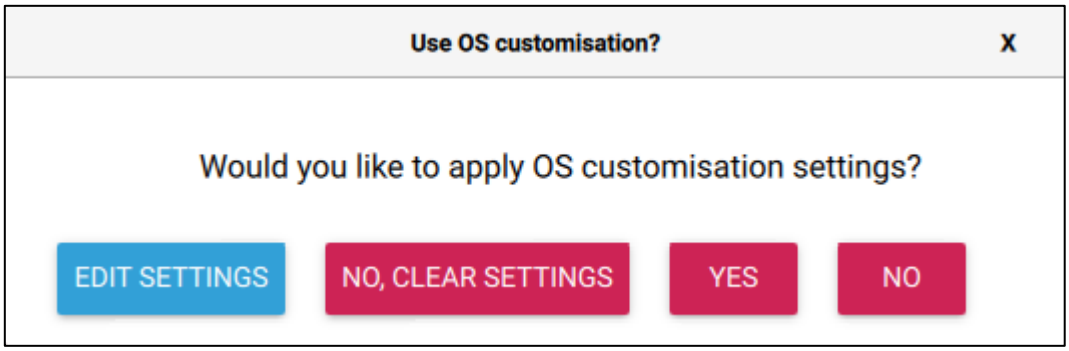


Wireless LAN Country must be correctly set; otherwise, it may fail to search the WiFi.
Enable SSH and click Save.

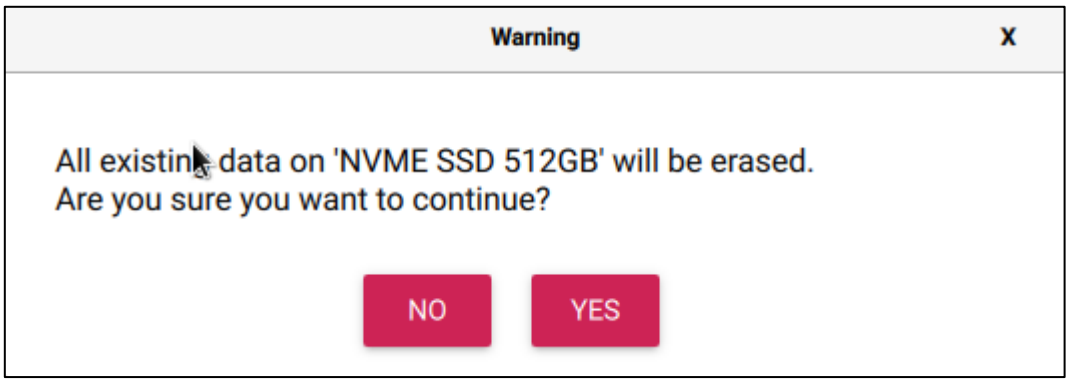




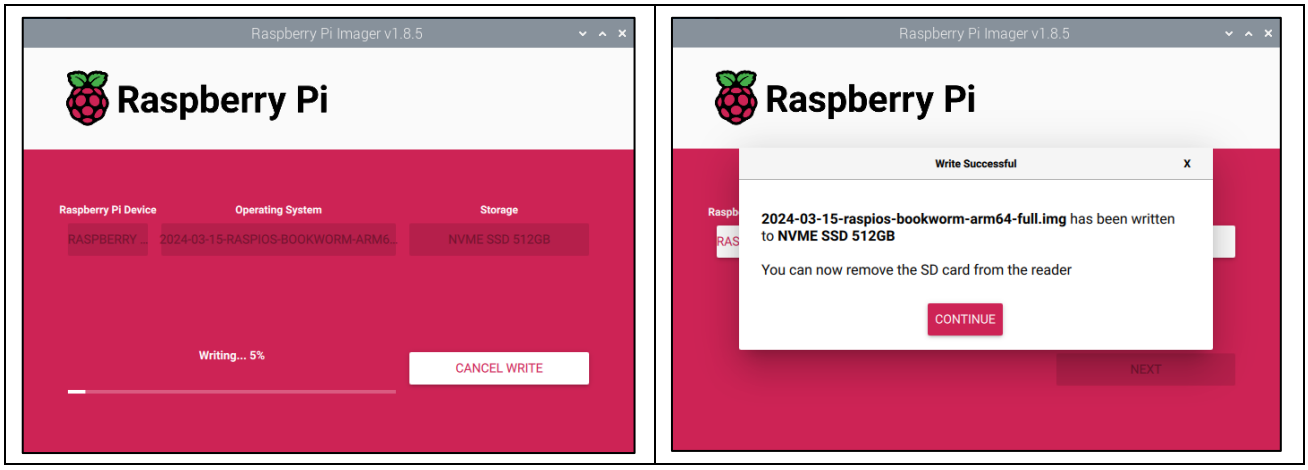
Click on YES.



Click on YES.



Wait for it to finish.



Congratulations! You have done the trickiest and the time-consuming part. Now that you have successfully installed the operating system onto the NVMe SSD, you are very close to achieving a triumph.

Next, boot into the system from SSD.

3.2.6 Enable PCIE3.0 (on system written into SSD)

If you have confirmed that SSD is with Phison controller in step 3, then you also need to enable PCIE3.0 on the system written into SSD.
If the controller of your SSD is not from Phison, **you can skip this section.**

The operation is as below:
Run the command lsblk to check the partitions of the SSD with Raspberry Pi OS written, as shown below:

```
pi@raspberrypi:~ $ lsblk
NAME        MAJ:MIN RM   SIZE RO TYPE MOUNTPOINTS
mmcblk0     179:0    0  29.3G  0 disk
├─mmcblk0p1 179:1    0   512M  0 part /boot/firmware
└─mmcblk0p2 179:2    0  28.8G  0 part /
nvme0n1     259:0    0 119.2G  0 disk
├─nvme0n1p1 259:1    0   512M  0 part
└─nvme0n1p2 259:2    0 118.7G  0 part
pi@raspberrypi:~ $
```

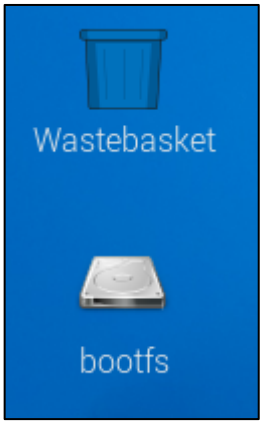
(The above screenshot is the result of an 128GB SSD with Phison as main controller.)

Run the following commands one by one to mount partition 1 of the SSD to the directory of /media/pi.

```
sudo mkdir /media/pi
sudo mount /dev/nvme0n1p1 /media/pi
```

```
pi@raspberrypi:~ $ sudo mkdir /media/pi
pi@raspberrypi:~ $ sudo mount /dev/nvme0n1p1 /media/pi
```

If it mounts successfully, you'll see the following disk icon on the desktop.



Open and modify the config.txt file with the following command.

```
sudo nano /media/pi/config.txt
```

```
pi@raspberrypi:~ $ sudo nano /media/pi/config.txt
```

Add the line dtparam=pciex1_gen=3 to the end of the file, as shown below:

```
# Disable compensation for displays with overscan
disable_overscan=1

# Run as fast as firmware / board allows
arm_boost=1

[cm4]
# Enable host mode on the 2711 built-in XHCI USB controller.
# This line should be removed if the legacy DWC2 controller is required
# (e.g. for USB device mode) or if USB support is not required.
otg_mode=1

[cm5]
dtoverlay=dwc2,dr_mode=host

[all]
dtparam=pciex1_gen=3
```

Press Ctrl-O to save the file, Enter to confirm, and Ctrl-X to exit.

3.2.7 Booting from SSD

After finishing flashing the OS to SSD, shutdown Raspberry Pi, remove the power supply, and remove the SD card. Then connect the power, the Raspberry Pi will boot from SSD.

The default boot order of Raspberry Pi is SD card → SSD → USB, Therefore, when the SD card is removed, the Raspberry Pi cannot detect the SD card, it will boot from SSD. By far, the Raspberry Pi can boot successfully from NVME SSD.

```
B1 SD Card Boot  Boot from SD Card before trying NVMe and then USB (RECOMMENDED)
B2 NVMe/USB Boot Boot from NVMe before trying USB and then SD Card
B3 Network Boot  Boot from Network unless override by SD Card
```

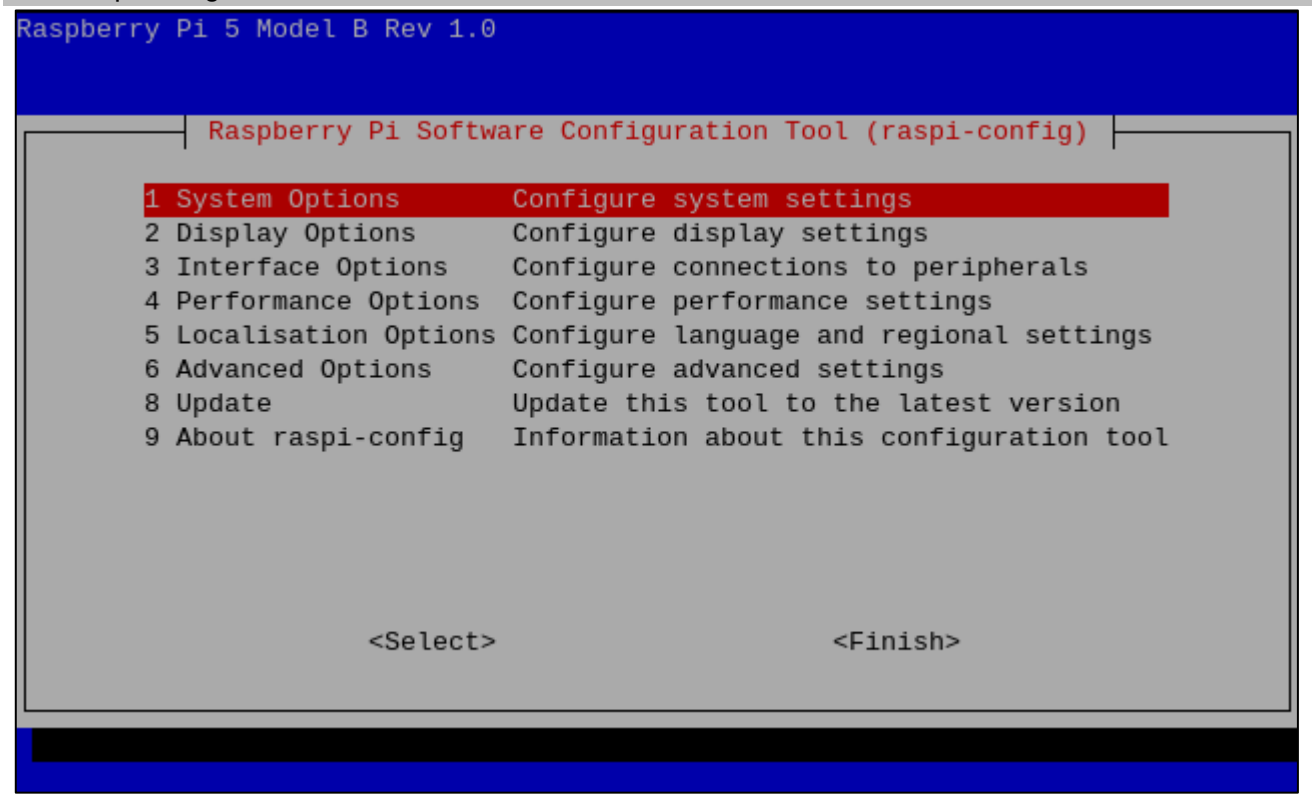
If you want the Raspberry Pi to boot from the SSD first, please continue with the following steps to modify the boot order. The boot order is saved in the Pi's EEPROM, so it does not matter whether you modify the boot order on SD card system or SSD system.

If you do not want to change the boot order, please skip this chapter.

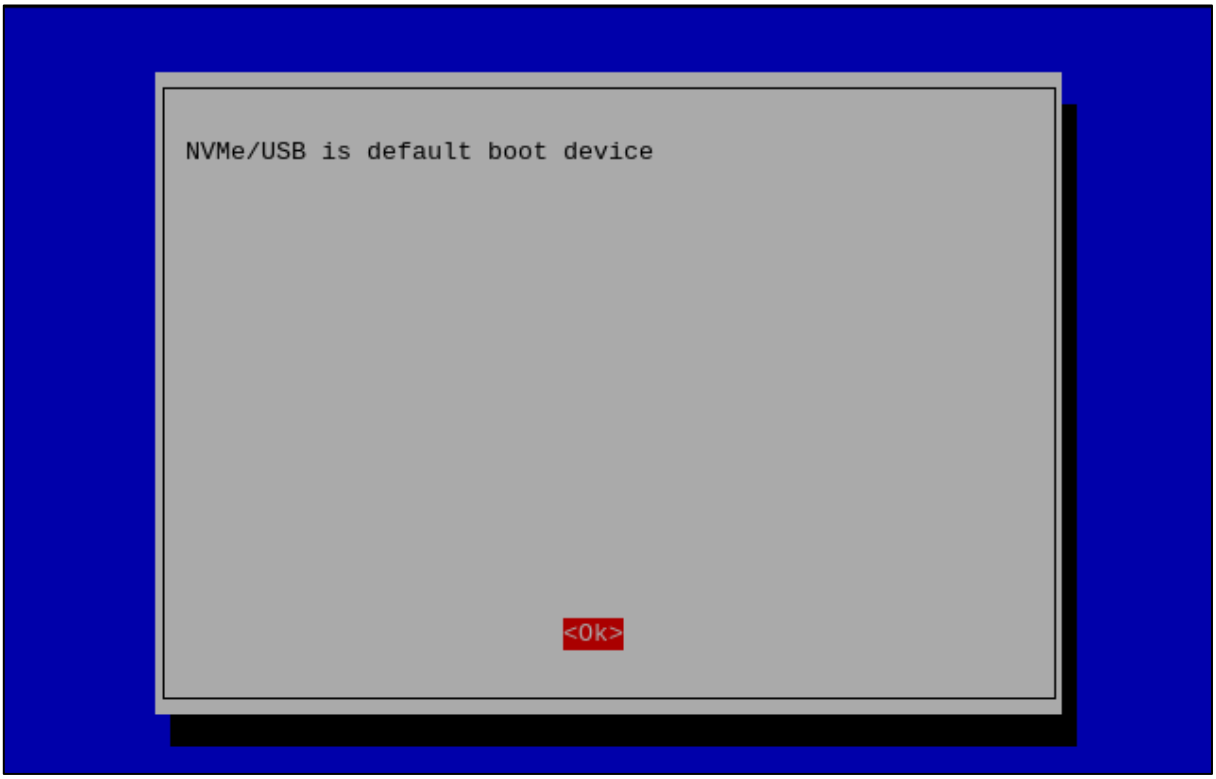
Configuring the Boot Order

Type the following command in the Terminal.

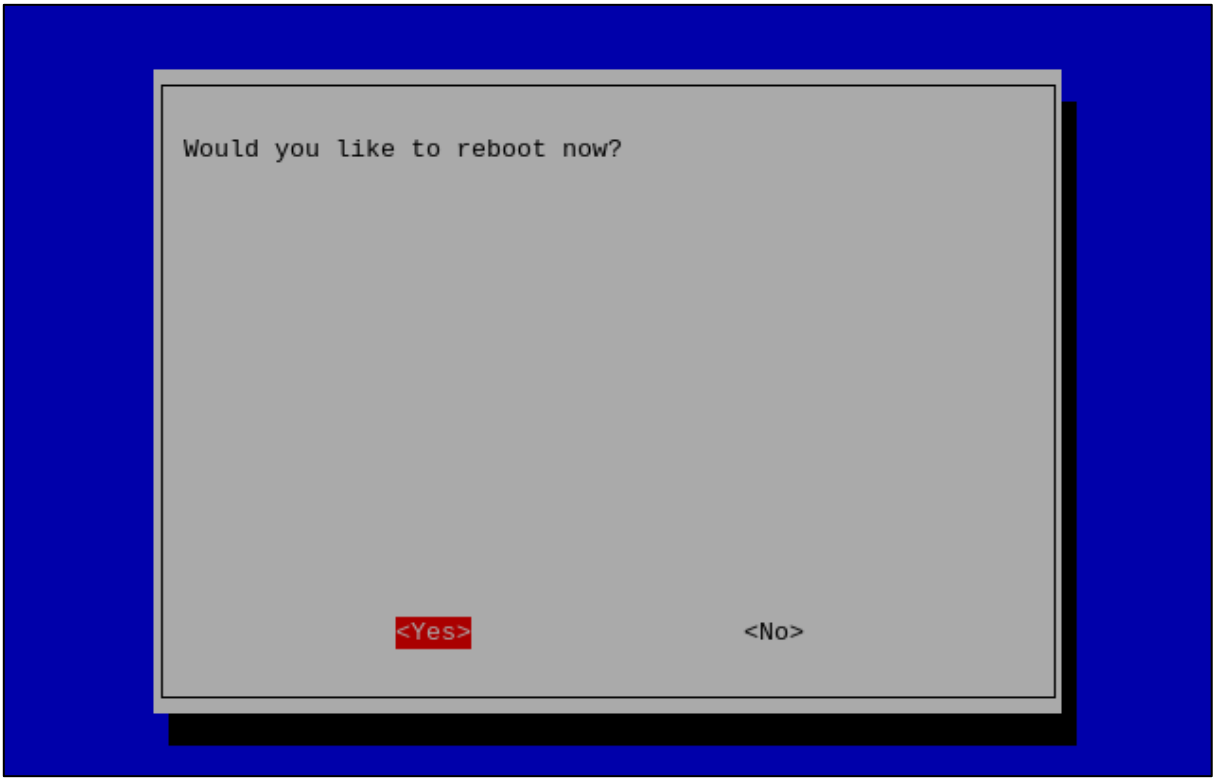
```
sudo raspi-config
```



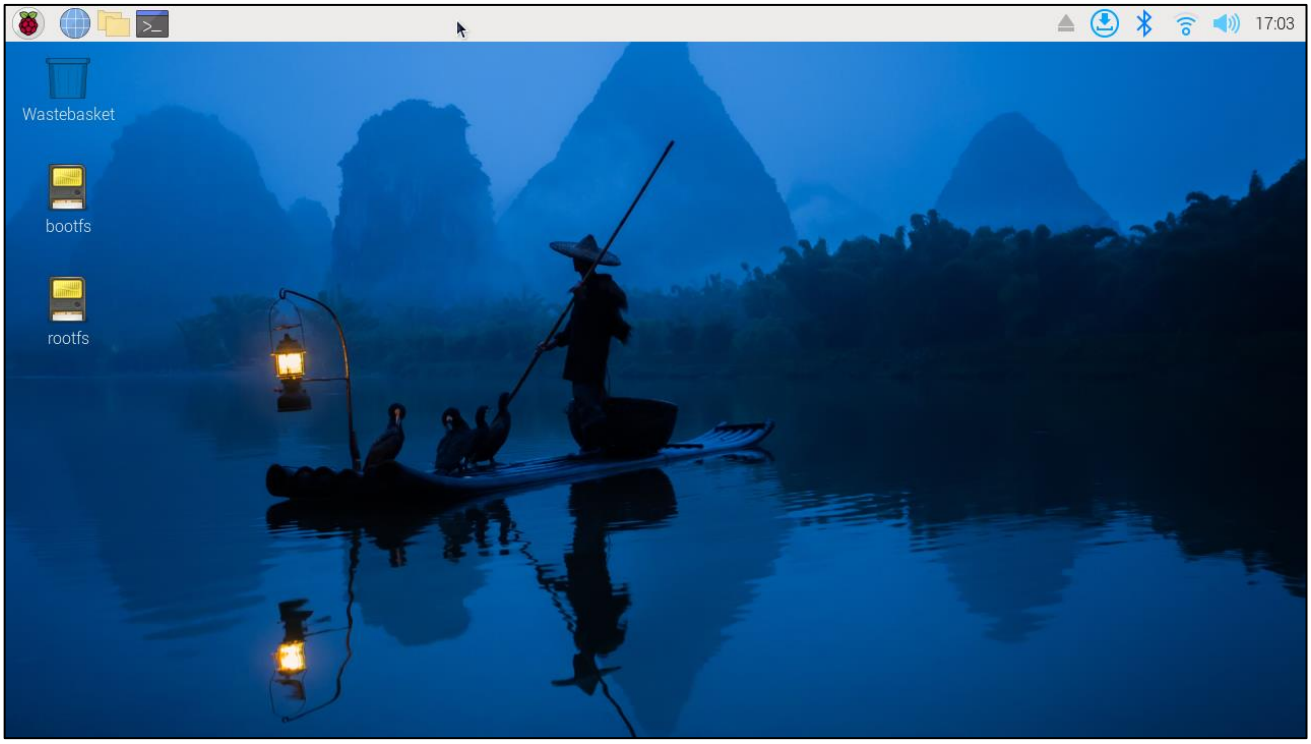
Using the keyboard's arrow keys and the Enter key, select the options in sequence.
"6 Advanced Options" →"A4 Boot Order"→"B2 NVME/USB Boot ···"



Select "OK"→"Finish"→"Yes", and reboot your Raspberry Pi.



At this point, upon restarting, the Raspberry Pi will boot from the NVME SSD first. If you are using an external monitor, you will see that the Raspberry Pi has booted up correctly. If your SD card is still inserted, you will also see an icon on the desktop as shown below.
With this, the process of booting the Raspberry Pi from the NVME SSD has been fully completed.



If you use VNC viewer, you will need to repeat the previous steps to activate the VNC service as it is not yet enabled in the new system on the SSD. Here, we take Windows as an example.
Run the following command:

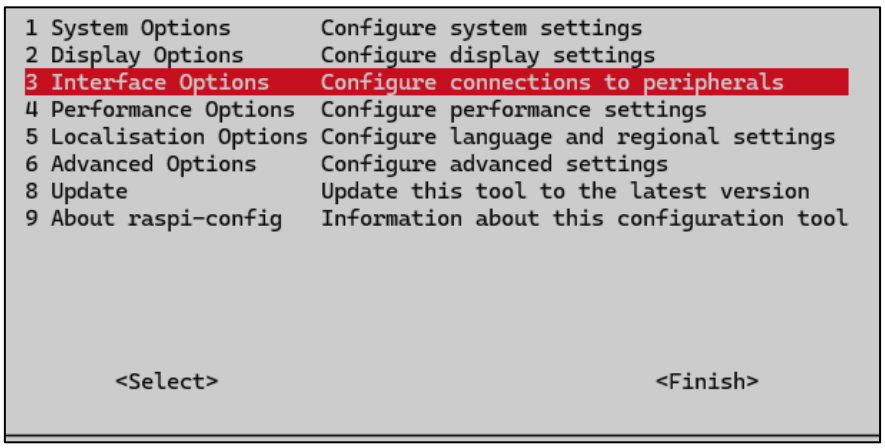
```
ssh pi@raspberrypi.local
```

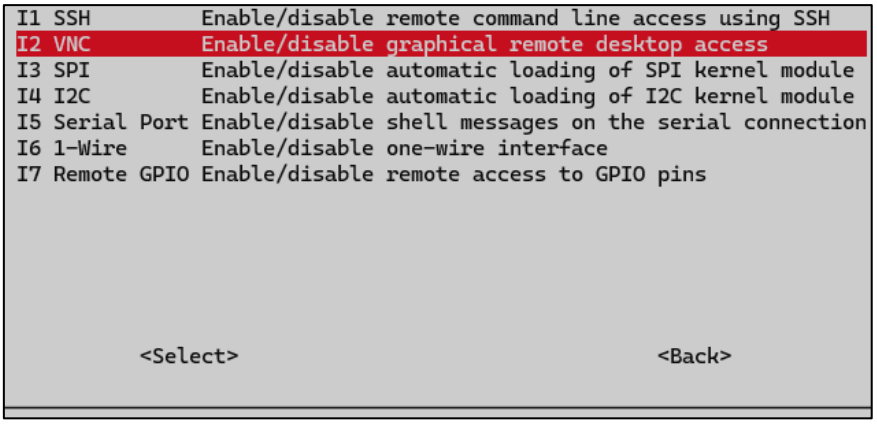
```
C:\Users\Administrator>ssh pi@raspberrypi.local
The authenticity of host 'raspberrypi.local (240e:3b4:3812:1fc0:954e:f55f:a772:fed5)' can't be established.
ECDSA key fingerprint is SHA256:hcx7u6H73nUsIc5WXA3HWa5GPSZEDroiz/mMbQx3ogc.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'raspberrypi.local,240e:3b4:3812:1fc0:954e:f55f:a772:fed5' (ECDSA) to the list of known hosts
.
pi@raspberrypi.local's password:
Linux raspberrypi 6.6.20+rpt-rpi-2712 #1 SMP PREEMPT Debian 1:6.6.20-1+rpt1 (2024-03-07) aarch64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Jun  3 16:50:25 2024

SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set a new password.

pi@raspberrypi:~ $
```

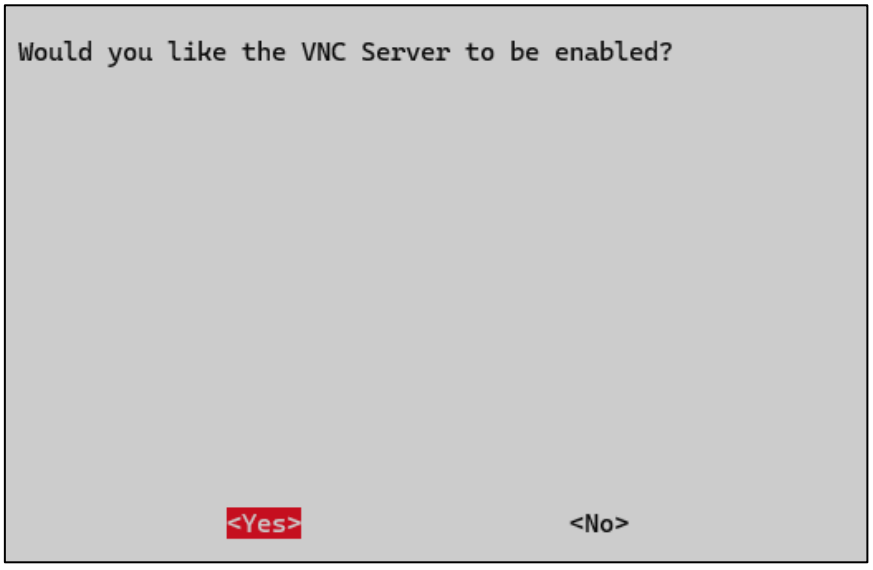
Once successfully ssh into Raspberry Pi, run the following command to open the configuration and enable VNC.

```
sudo raspi-config
```

Select “3 Interface Options”→”I2 VNC”→”Yes”→”Finish”.







Now you should be able to access Raspberry Pi via VNC.

Chapter 4 Functionality Tests

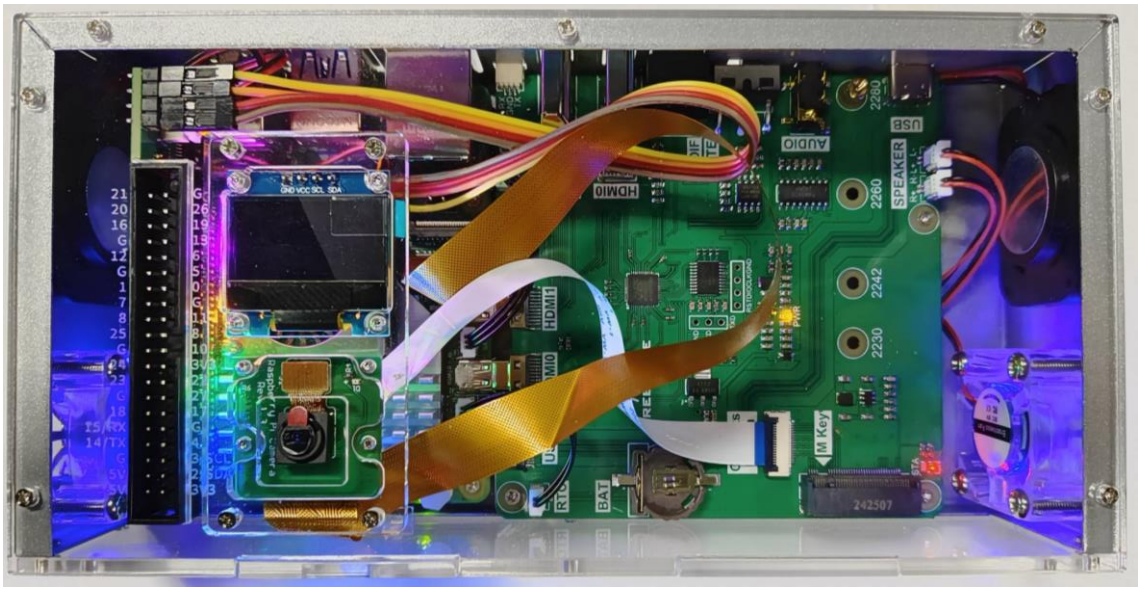
Before powering on the Freenove Computer Case for Raspberry Pi, please make sure that all cable connections are correct. Due to its multiple functions, this case requires an adequate power supply. We highly recommend using the official Raspberry Pi 5.1V / 5A power adapter (https://www.raspberrypi.com/products/27w-power-supply). Failure to do so may result in the Freenove Computer Case for Raspberry Pi being unusable or causing damage to components.

4.1 Phenomena of a Successful Boot

If the computer case boots up successfully, you should see the following phenomena:

- On the Raspberry Pi 5, the STAT indicator light remains steadily lit in green.
- On the Case Adapter Board:
  - Both the PWR power indicator and the ON indicator are constantly illuminated.
  - The status indicator of the GPIO Board blinks intermittently,
  - The LED lights display a rainbow pattern.
  - The fans operate in adaptive mode, with their rotational speed adjusting automatically in response to temperature changes within the computer case.

If you have any questions of the above, please contact us at **support@freenove.com**

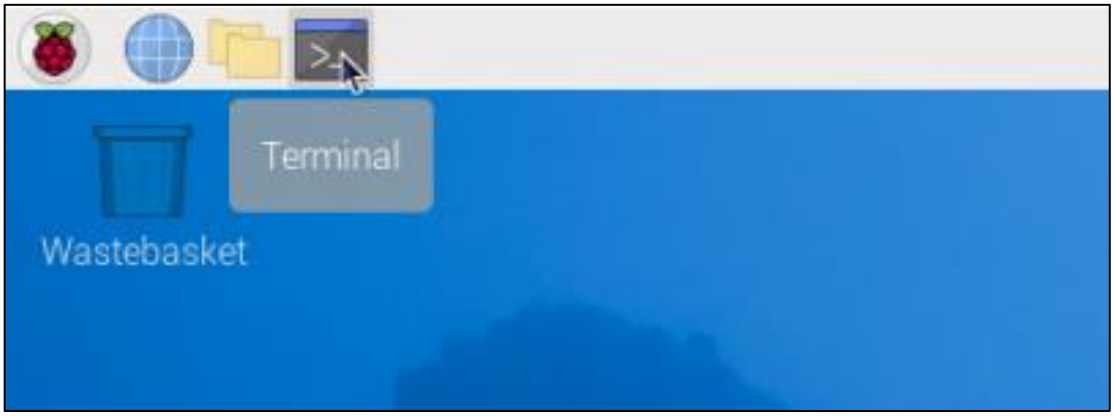


If you see the above results, then your computer case is correctly assembled and functioning well. At the point, you can connect a screen to your Raspberry Pi or access it via VNC viewer.

4.2 Component Tests

Before we start the tests, we need to download the test code and install some necessary libraries.

Click Terminal

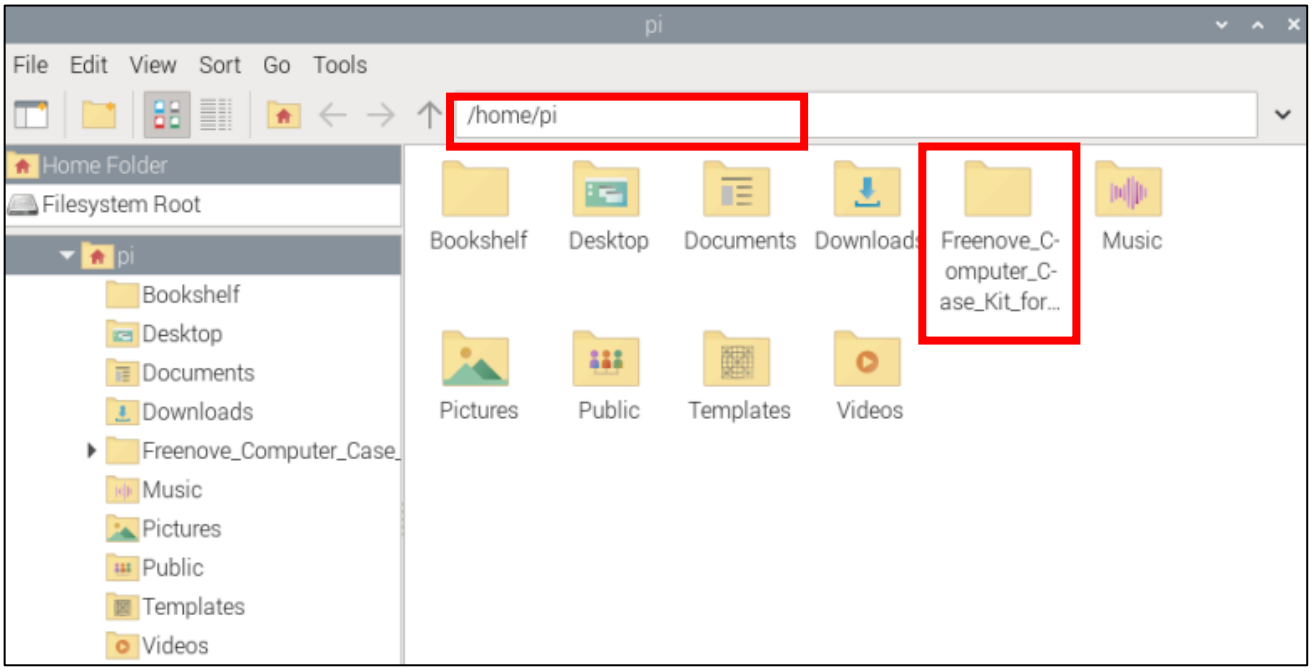


Run the following commands to obtain to code to your Raspberry Pi. After that, you can find the folder under the /home/pi directory.

You can also find and download the code **in Raspberry Pi** by visiting our official website (http://www.freenove.com) or our GitHub repository (https://github.com/freenove)

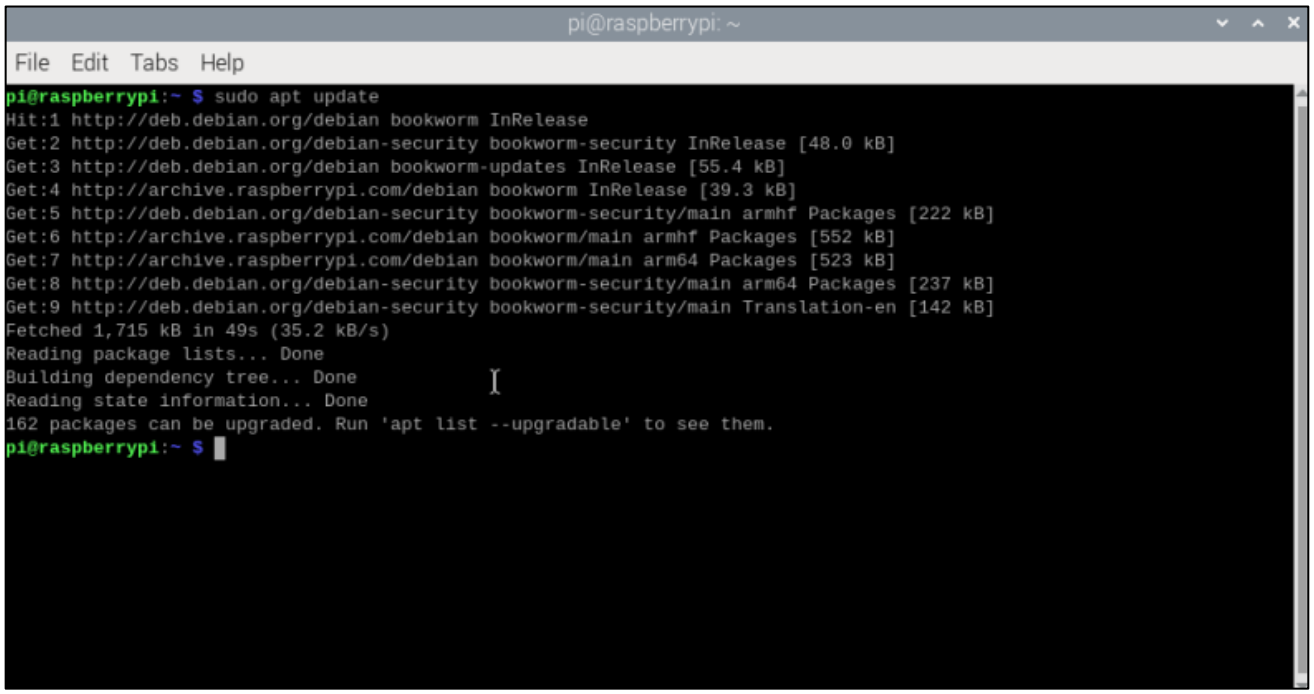
```
cd ~

git clone https://github.com/Freenove/Freenove_Computer_Case_Kit_for_Raspberry_Pi.git
```

```
pi@raspberrypi:~ $ git clone https://github.com/Freenove/Freenove_Computer_Case_
Kit_for_Raspberry_Pi.git
Cloning into 'Freenove_Computer_Case_Kit_for_Raspberry_Pi'...
```

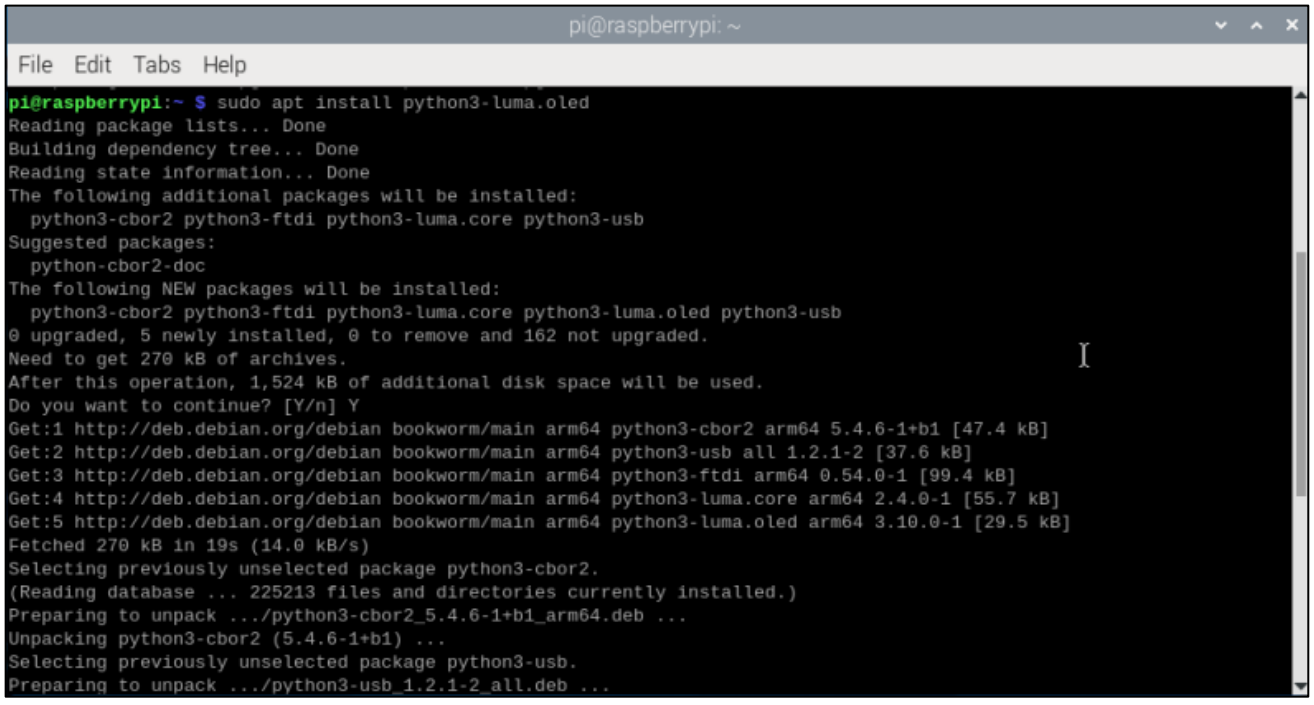


Run the following commands one by one to install the OLED library. Without it, the OLED screen may not work as expected.

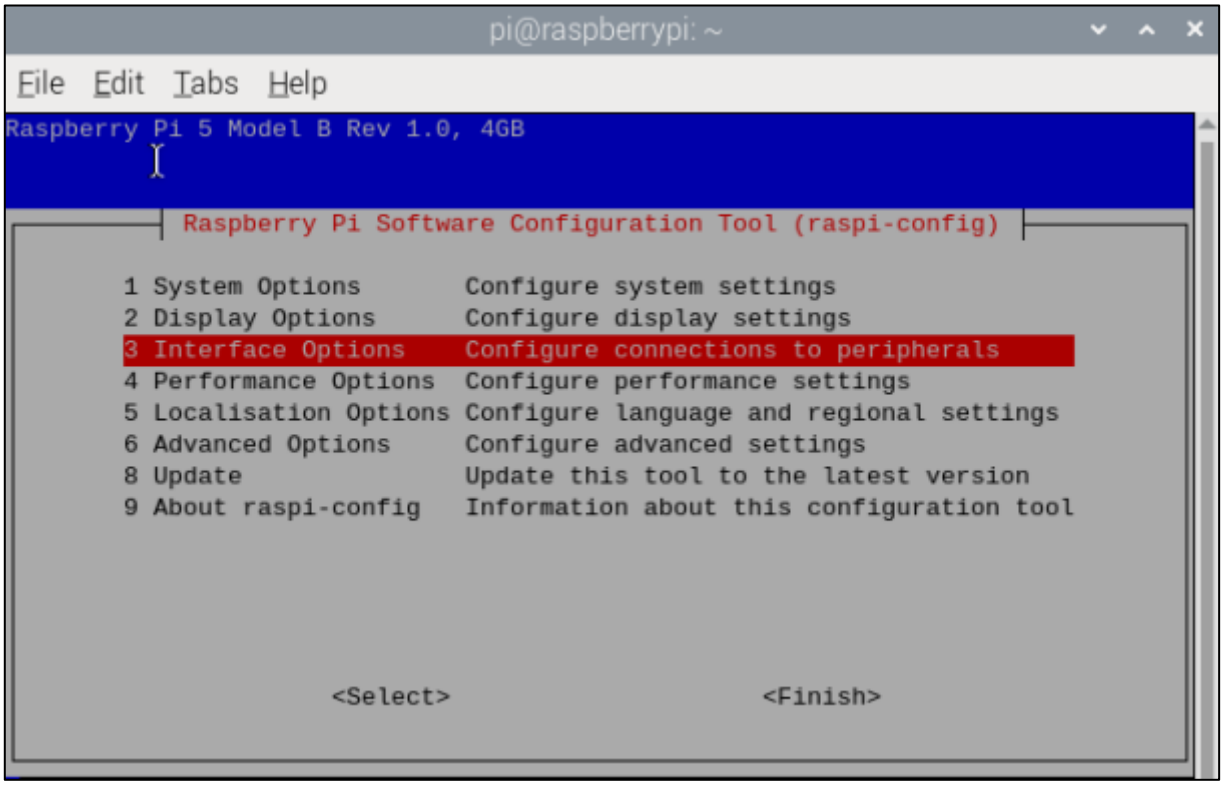
```
sudo apt update
```

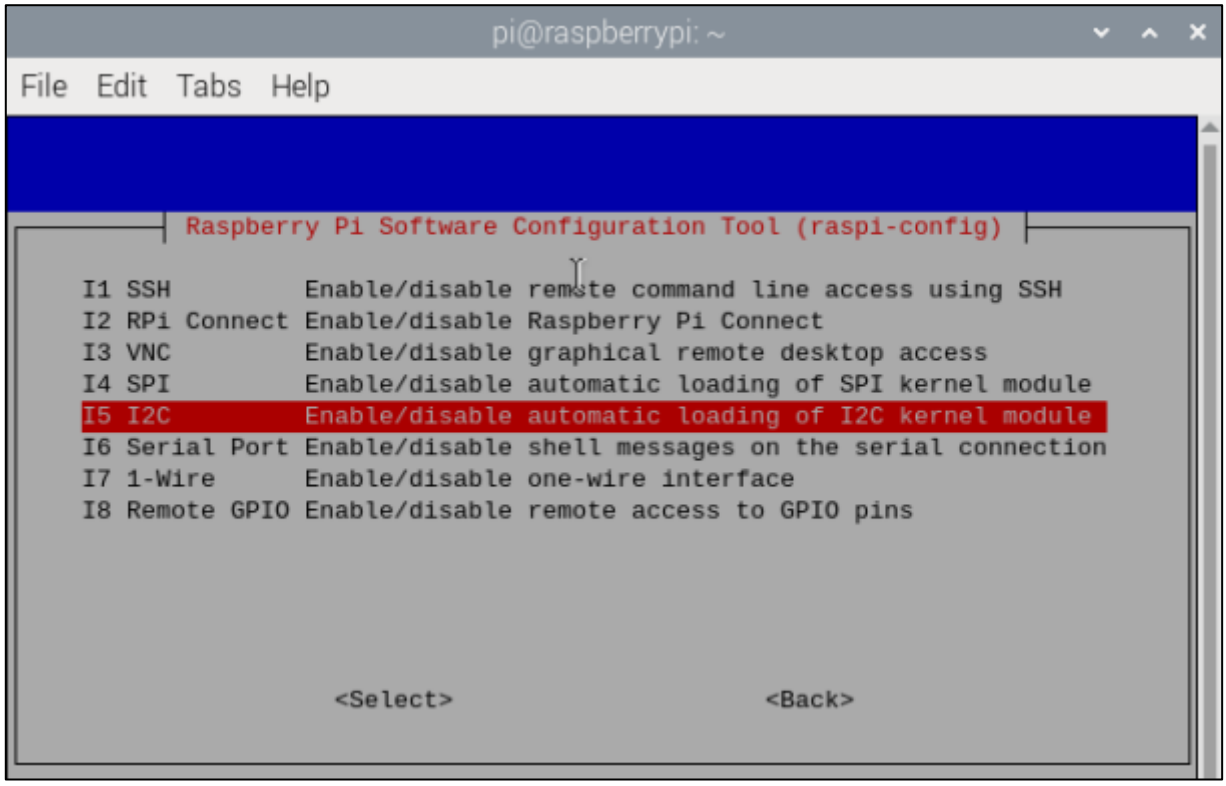



```
sudo apt install python3-luma.oled
```



As the GPIO Board and OLED screen are controlled via the RPi 5's IIC, we need to enable I2C on RPi 5. Run sudo raspi-config, select Interface Options -> I5 I2C -> Enter -> Yes -> OK.

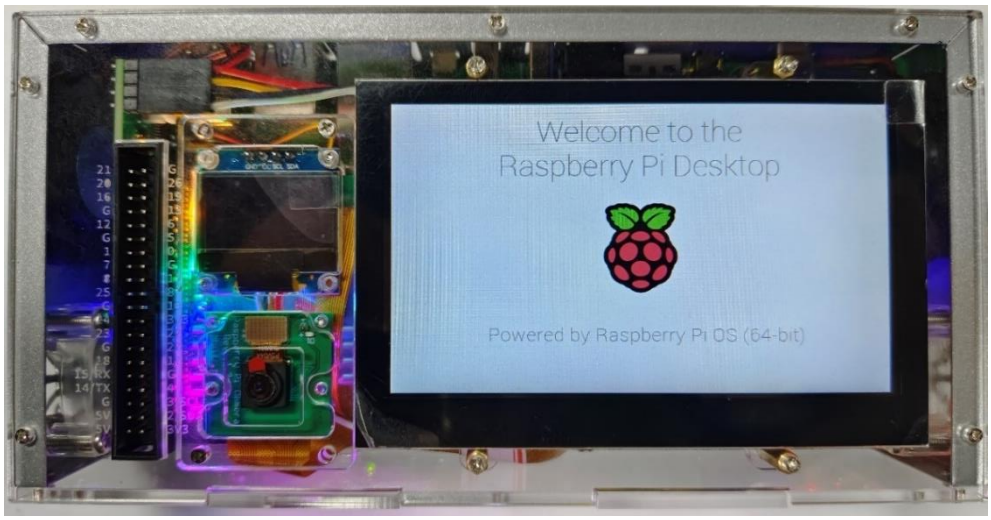




4.2.1 LCD Screen Test

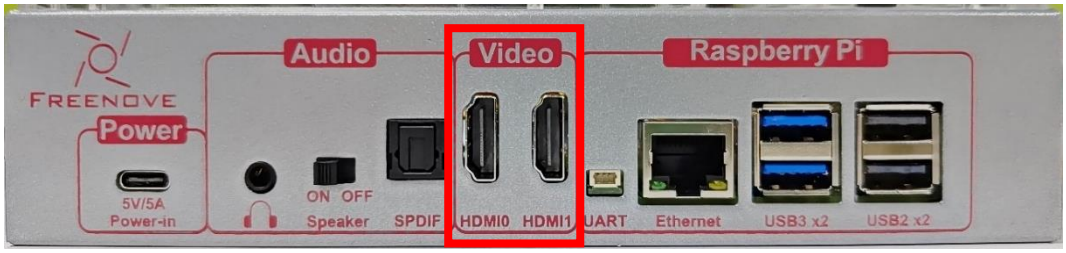
If your purchase is not an FNK0100K, you can skip to the next test.

When the Raspberry Pi 5 starts up, you can see the boot screen of the Raspberry Pi 5 on the monitor. If there is no display, please check whether the cables are connected correctly.
If you would like to learn more information about the display, you may click
https://github.com/Freenove/Freenove_Touchscreen_Monitor_for_Raspberry_Pi



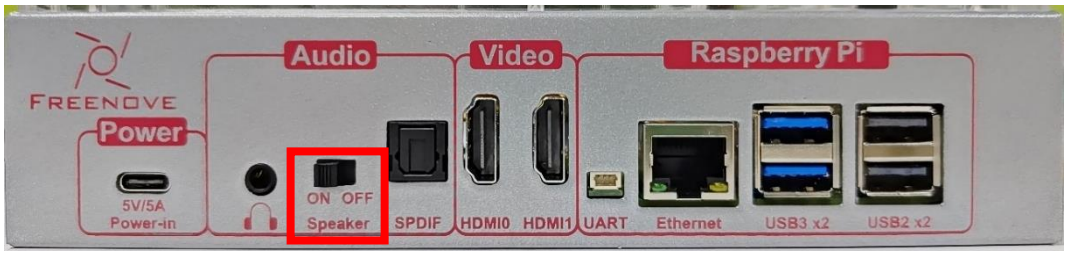
4.2.2 HDMI Test

If you have an HDMI screen yourself, you can connect it to the case with an HDMI cable.

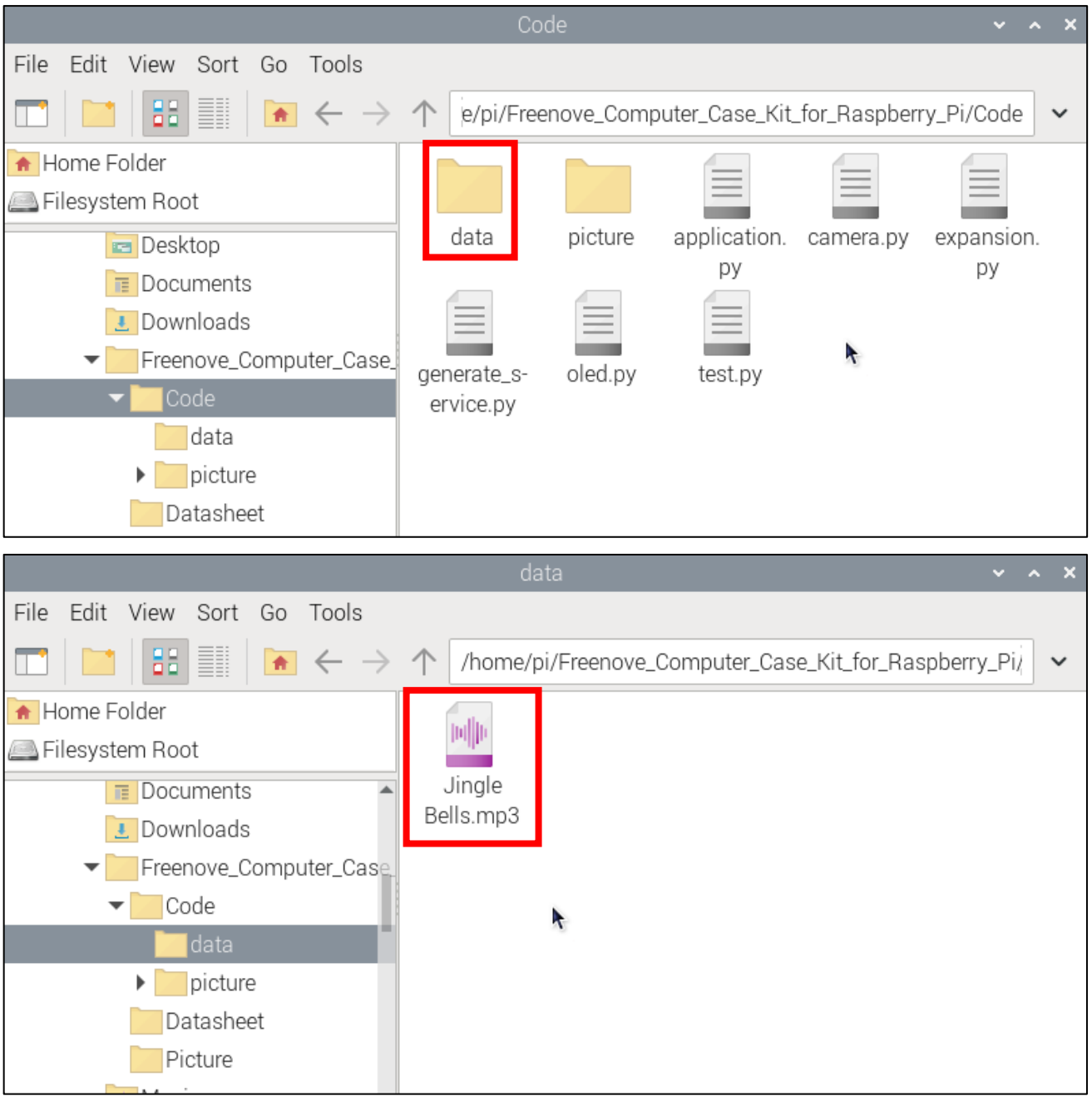


4.2.3 Speaker Test

Before playing music, please check whether the speak switch is ON.



Open the data folder under the Freenove_Computer_Case_Kit_for_Raspberry_Pi/Code directory, and click the mp3 file to display.



4.2.4 Camera Test

Enter the command `cd Freenove_Computer_Case_Kit_for_Raspberry_Pi/Code` to enter the directory where the test files are located.

```
pi@raspberrypi:~ $ cd Freenove_Computer_Case_Kit_for_Raspberry_Pi/Code
pi@raspberrypi:~/Freenove_Computer_Case_Kit_for_Raspberry_Pi/Code $
```

Run `python test.py -h` to view the test command.

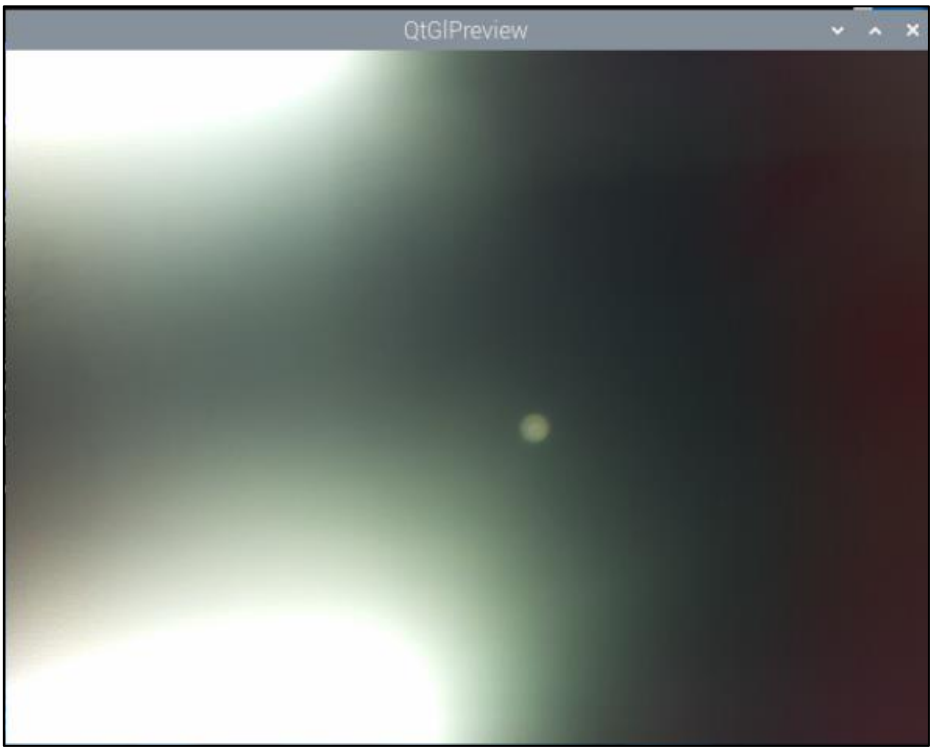
```
pi@raspberrypi:~/Freenove_Computer_Case_Kit_for_Raspberry_Pi/Code $ python test.py -h
Usage: test.py --camera | --oled | --fan | --led <mode:1-4>
```

Run the following command to test the camera.

```
python test.py --camera
```

```
pi@raspberrypi:~/Freenove_Computer_Case_Kit_for_Raspberry_Pi/Code $ python test.py --camera
[0:42:32.275987636] [3481]  INFO Camera camera_manager.cpp:325 libcamera v0.3.2+27-7330f29b
[0:42:32.283309431] [3491]  INFO RPI pisp.cpp:695 libpisp version v1.0.7 28196ed6edcf 29-08-2024 (
16:33:32)
[0:42:32.292570250] [3491]  INFO RPI pisp.cpp:1154 Registered camera /base/axi/pcie@120000/rp1/i2c
@80000/ov5647@36 to CFE device /dev/media2 and ISP device /dev/media0 using PiSP variant BCM2712_C
0
[0:42:32.295613343] [3481]  INFO Camera camera.cpp:1197 configuring streams: (0) 640x480-XBGR8888
(1) 640x480-GRBG_PISP_COMP1
[0:42:32.295716330] [3491]  INFO RPI pisp.cpp:1450 Sensor: /base/axi/pcie@120000/rp1/i2c@80000/ov5
647@36 - Selected sensor format: 640x480-SGRBG10_1X10 - Selected CFE format: 640x480-PC1G
view image...
QStandardPaths: wrong permissions on runtime directory /run/user/1000, 0770 instead of 0700
Use Ctrl+C to exit...
```

Press CTRL+C to exit the program.



4.2.5 OLED Test

Run the following code to test the OLED.

```
python test.py --oled
```

```
pi@raspberrypi:~/Freenove_Computer_Case_Kit_for_Raspberry_Pi/Code $ python test.py --oled
Use Ctrl+C to exit...
```

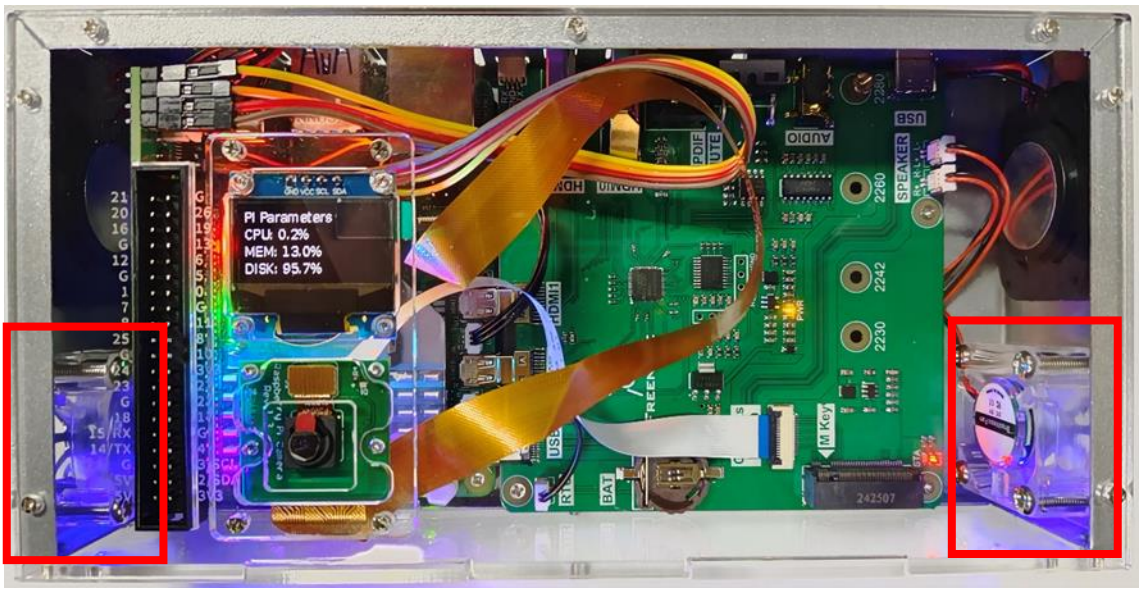
You should see the text displayed on it, as shown below. Press CTRL+C to exit the program.

4.2.6 Fan Test

Enter the following command to test the fans. The PWM duty cycles of the fans on both sides will gradually increase until they reach their maximum values, after which they will gradually decrease. Press Ctrl + C to exit once the testing is complete.

```
python test.py --fan
```

```
pi@raspberrypi:~/Freenove_Computer_Case_Kit_for_Raspberry_Pi/Code $ python test.py --fan
Use Ctrl+C to exit...
```



4.2.7 LED Test

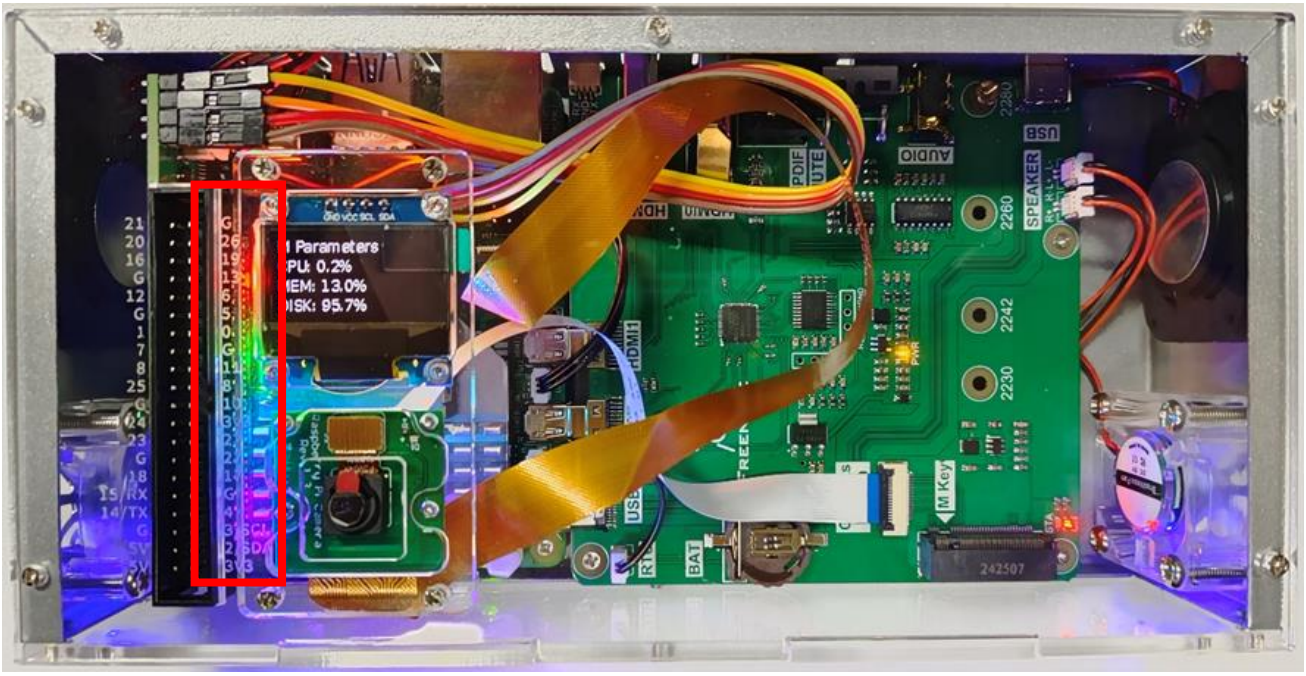
Enter the following code to test the LED lights. You can choose from different modes:

1. Static Mode: The LED lights display a single color, cycling through red - green - blue.
2. Following Mode: The LED lights will be lit one after another in sequence.
3. Breathing Mode: The brightness of the LED lights gradually increases and then gradually decreases.
4. Rainbow Mode: The LED lights display colors like a rainbow.

```
python test.py --led <mode:1-4>
```

```
pi@raspberrypi:~/Freenove_Computer_Case_Kit_for_Raspberry_Pi/Code $ python test.py --led 1
Use Ctrl+C to exit...
^CKeyboardInterrupt
pi@raspberrypi:~/Freenove_Computer_Case_Kit_for_Raspberry_Pi/Code $ python test.py --led 2
Use Ctrl+C to exit...
^CKeyboardInterrupt
pi@raspberrypi:~/Freenove_Computer_Case_Kit_for_Raspberry_Pi/Code $ python test.py --led 3
Use Ctrl+C to exit...
^CKeyboardInterrupt
pi@raspberrypi:~/Freenove_Computer_Case_Kit_for_Raspberry_Pi/Code $ python test.py --led 4
Use Ctrl+C to exit...
```

The LED lights will work in the mode you choose. Please check whether they work as expected.



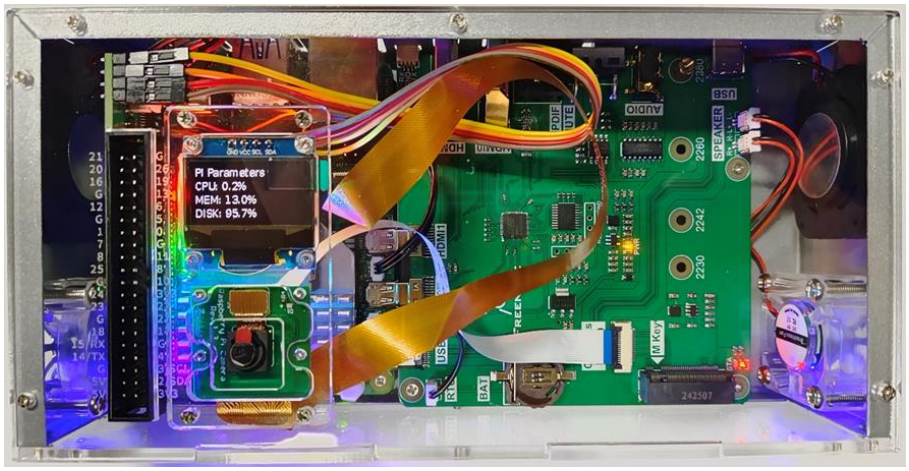
4.3 Overall Test

If all components pass their tests, it indicates that they are all in good condition.
If you encounter any issues with them, please feel free to contact us at: support@freenove.com

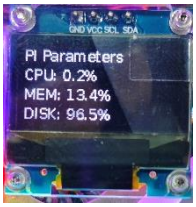
You can test the overall functions of the computer case using the following command:

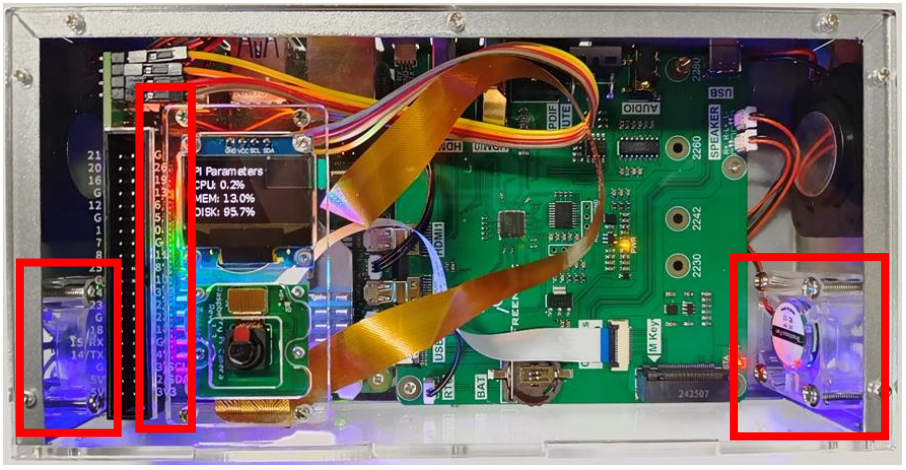
```
python application.py
```

```
pi@raspberrypi:~/Freenove_Computer_Case_Kit_for_Raspberry_Pi/Code $ python application.py
```



All components will enter operational mode. Here's what you can expect:
OLED Display: Automatically switches between different display contents.
LED Lights: Illuminate in rainbow mode.
Fans: Retrieves the PWM duty cycle from the Pi 5 CPU heatsink and automatically adjust their own PWM duty cycle based on the retrieved value.
Note: If the fans do not start rotating, it may be due to the Pi 5 CPU active cooler not being activated. Please ensure that the active cooler is functioning correctly and is properly connected.





Chapter 5 Auto Start Setting

By now, you can start using your Raspberry Pi case. If you don't want to change to our configuration, you can skip to the next chapter.

You can set it to start up automatically by entering the following commands. After the settings are completed, when your Raspberry Pi 5 restarts, the OLED display and the case fans will run automatically.

5.1 Commands

Run the following commands to perform a one-click configuration for your Raspberry Pi 5. When the Raspberry Pi 5 boots up, the OLED display will automatically show some basic information about the Raspberry Pi 5. The GPIO Board will retrieve the PWM duty cycle of the Raspberry Pi 5's CPU active cooler and use it to set the PWM duty cycle of the case fans.

```
sudo python generate_service.py
```

```
pi@raspberrypi:~/Freenove_Computer_Case_Kit_for_Raspberry_Pi/Code $ sudo python generate_service.py
Current Directory: /home/pi/Freenove_Computer_Case_Kit_for_Raspberry_Pi/Code
Current Username: pi
Created symlink /etc/systemd/system/multi-user.target.wants/my_app_running.service → /etc/systemd/system/my_app_
running.service.
```

Execute the following command, and the OLED display will automatically show some basic information about the Raspberry Pi 5. Meanwhile, the GPIO Board will retrieve the PWM duty cycle of the Raspberry Pi 5's CPU active cooler and use it to set the PWM duty cycle of the case fans.

```
sudo systemctl start my_app_running.service
```

```
pi@raspberrypi:~/Freenove_Computer_Case_Kit_for_Raspberry_Pi/Code $ sudo systemctl start my_app_running.service
```

However, this won't configure your Raspberry Pi 5 to start these devices automatically when it boots up. As a result, when the Raspberry Pi 5 restarts, you'll still need to run the command again to start the OLED display and the case fans.

By running the following command, you can stop the OLED display and the case fan. If you have configured your Raspberry Pi 5 to start these devices automatically upon boot, the OLED screen and the case fans will start on their own when the Raspberry Pi restarts.

```
sudo systemctl stop my_app_running.service
```

```
pi@raspberrypi:~/Freenove_Computer_Case_Kit_for_Raspberry_Pi/Code $ sudo systemctl stop my_app_running.service
```

Run the following command to configure your Raspberry Pi 5 to start the OLED display and the case fans automatically upon boot (use with caution).

```
sudo systemctl enable my_app_running.service
```

```
pi@raspberrypi:~/Freenove_Computer_Case_Kit_for_Raspberry_Pi/Code $ sudo systemctl enable my_app_running.service
```

Run the following commands will disable the auto start of the OLED display and the case fans upon the Raspberry Pi 5 boots up (use with caution).

```
sudo systemctl disable my_app_running.service
```

```
pi@raspberrypi:~/Freenove_Computer_Case_Kit_for_Raspberry_Pi/Code $ sudo systemctl disable my_app_running.service
```

5.2 Code Introduction

By invoking different commands, you can set up one-click startup for your OLED display and case fans. Below are the codes:

```
import os
import time
import sys
import shutil

DEBUG = False

def check_application_py(filenanme="application.py"):
    if not os.path.exists(filenanme):
        print("Error: {filenanme} does not exist in the current directory.")
        sys.exit(1)

def get_current_directory():
    if DEBUG:
        print("Getting current directory...")
    return os.getcwd()

def get_current_username_from_directory(directory):
    if DEBUG:
        print("Extracting username from directory path...")
    try:
        parts = directory.split('/')
        if len(parts) > 2 and parts[1] == 'home':
            return parts[2]
        else:
            print("Error: Unable to extract username from directory path.")
            sys.exit(1)
    except Exception as e:
        print(f"Error extracting username from directory path: {e}")
        sys.exit(1)

def create_my_app_running_service(directory, username):
    service_content = f"""[Unit]
Description=My Python Script Service

[Service]
ExecStart=/usr/bin/python3 {directory}/application.py
WorkingDirectory={directory}
StandardOutput=inherit
```

```
StandardError=inherit
Restart=always
User={username}

[Install]
WantedBy=multi-user.target
"""
    service_file_path = os.path.join('/etc/systemd/system/', 'my_app_running.service')
    if os.path.exists(service_file_path):
        os.remove(service_file_path)
        if DEBUG:
            print(f"Existing my_app_running.service file removed: {service_file_path}")
    with open(service_file_path, 'w') as service_file:
        service_file.write(service_content)
    if DEBUG:
        print(f"my_app_running.service created at {service_file_path}")

def run_system_command(command):
    if DEBUG:
        print(f"Running command: {command}")
    try:
        result = os.system(command)
        if result != 0:
            print(f"Error executing command: {command}")
            sys.exit(1)
    except Exception as e:
        print(f"Error executing command: {command}")
        print(e)
        sys.exit(1)

def remove_pycache_folder(directory):
    pycache_path = os.path.join(directory, '__pycache__')
    if os.path.exists(pycache_path):
        try:
            shutil.rmtree(pycache_path)
            if DEBUG:
                print(f"__pycache__ folder removed: {pycache_path}")
        except Exception as e:
            print(f"Error removing __pycache__ folder: {e}")
            sys.exit(1)

if __name__ == "__main__":
    # Step 1: Check if application.py exists
    check_application_py()
```

```
84
85      # Step 2: Get current directory
86      current_directory = get_current_directory()
87
88      # Step 3: Get current username from directory
89      current_username = get_current_username_from_directory(current_directory)
90
91      print(f"Current Directory: {current_directory}")
92      print(f"Current Username: {current_username}")
93
94      # Step 4: Create my_app_running.service file
95      create_my_app_running_service(current_directory, current_username)
96
97      # Step 5: Run systemctl daemon-reload
98      run_system_command("sudo systemctl daemon-reload")
99      time.sleep(1)
100
101     # Step 6: Enable my_app_running.service
102     run_system_command("sudo systemctl enable my_app_running.service")
103     time.sleep(1)
104
105     '''
106     # Disable my_app_running.service
107     run_system_command("sudo systemctl disable my_app_running.service")
108     time.sleep(1)
109
110     # Stop my_app_running.service
111     run_system_command("sudo systemctl stop my_app_running.service")
112     time.sleep(1)
113     '''
114
115     # Step 7: Start my_app_running.service
116     run_system_command("sudo systemctl start my_app_running.service")
117     time.sleep(1)
118
119     remove_pycache_folder(current_directory)
```

Import the feature modules and set whether to enable the DEBUG function.

```
1   import os
2   import time
3   import sys
4   import shutil
5
6   DEBUG = False
```

Step 1: Check if application.py exists

```
81  if __name__ == "__main__":
82      # Step 1: Check if application.py exists
83      check_application_py()
```

Step 2: Get the current directory.

```
85      # Step 2: Get current directory
86      current_directory = get_current_directory()
```

Step 3: Pass the current path as a parameter into the function, retrieve the current username under that path, and print both the current path and the username.

```
88      # Step 3: Get current username from directory
89      current_username = get_current_username_from_directory(current_directory)
90
91      print(f"Current Directory: {current_directory}")
92      print(f"Current Username: {current_username}")
```

Step 4: Create an app running service using the current path parameter and the username parameter.

```
94      # Step 4: Create my_app_running.service file
95      create_my_app_running_service(current_directory, current_username)
```

Step 5: Run system commands.

```
97      # Step 5: Run systemctl daemon-reload
98      run_system_command("sudo systemctl daemon-reload")
99      time.sleep(1)
```

Step 6: Enable the app running service and set your device to start automatically on boot.

```
101     # Step 6: Enable my_app_running.service
102     run_system_command("sudo systemctl enable my_app_running.service")
103     time.sleep(1)
```

Step 7: Start your device.

```
115     # Step 7: Start my_app_running.service
116     run_system_command("sudo systemctl start my_app_running.service")
117     time.sleep(1)
```

Finally, delete the __pycache__ folder.

```
119     remove_pycache_folder(current_directory)
```

Reference

def check_application_py(filenanme="application.py"):

Check if application.py exists

Parameters

filename: name of the file, in string type, which is "application.py" by default

def get_current_directory():

Step 2: Get the current directory.

def get_current_username_from_directory(directory):

Get username under the current directory

Parameters

directory: retrieve from the path
`def create_my_app_running_service(directory, username):` Create an app running service using the current path. **Parameters** **directory:** under the current directory **username:** username
`def run_system_command(command):` Run system commands **Parameters** **command:** system command, in string type
`def remove_pycache_folder(directory):` Remove the _pycache_ folder。 **Parameters** **directory:** the _pycache_ folder under the current directory

Chapter 6 Changing Resolution

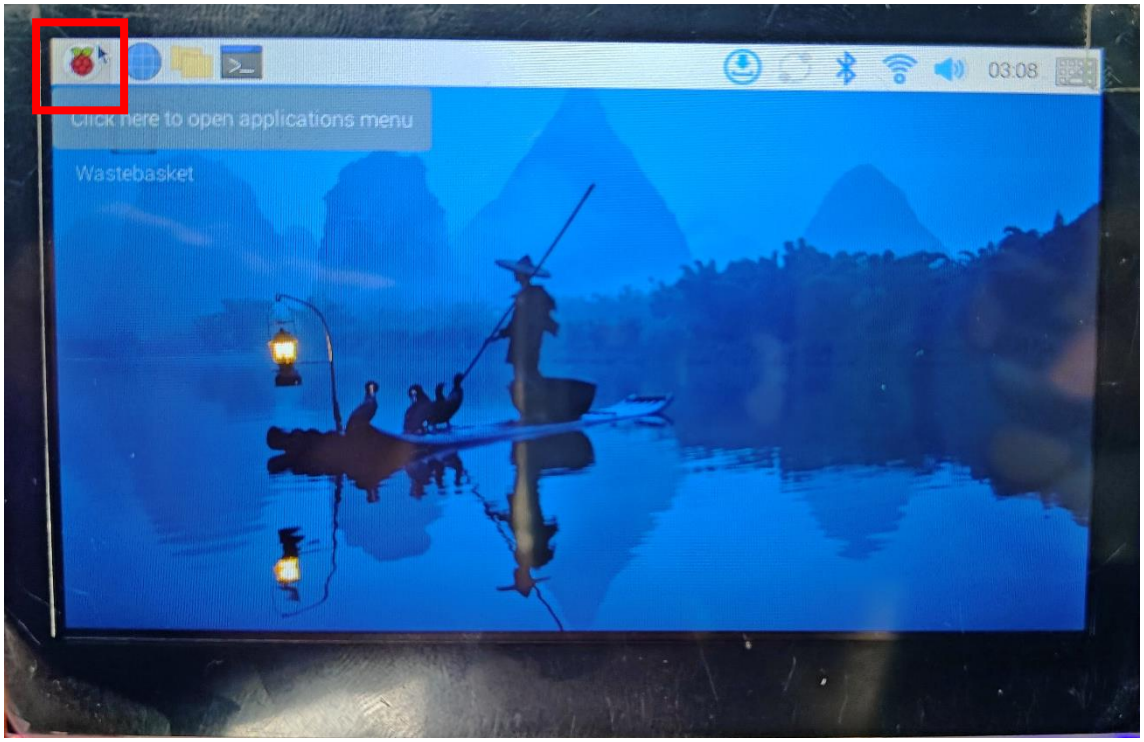
If your purchase is FNK0100B, you can skip to the next chapter.

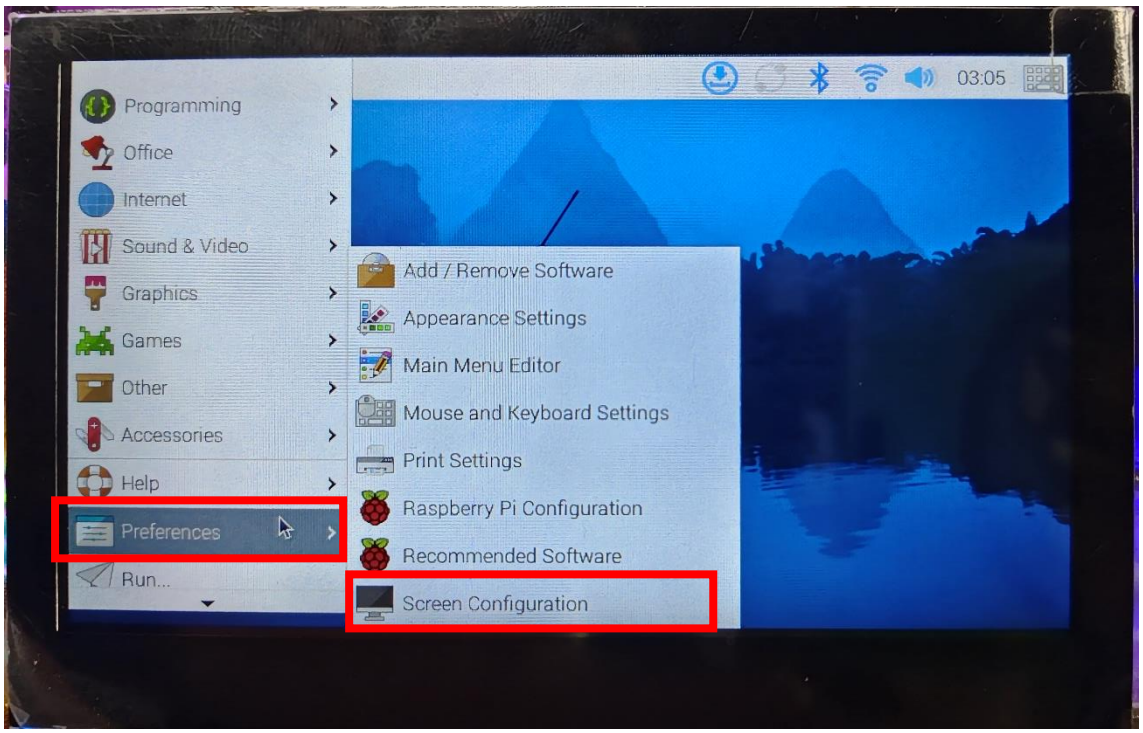
The FNK0100K features a display with a resolution of 800×480. When the Raspberry Pi 5 boots up, it creates two display areas: DSI-1 and HDMI-A-1, which have different resolutions. This mismatch will cause two issues:
1. VNC Connection: You will see no content in the VNC window because VNC mirrors the HDMI-A-1 area, which may not match the display configuration.
2. Touch Control: You cannot control the HDMI-A-1 interface via touch on the computer case's display.

Solution: Overlap the two display areas and adjust the resolution of HDMI-A-1 to match the screen (800×480).

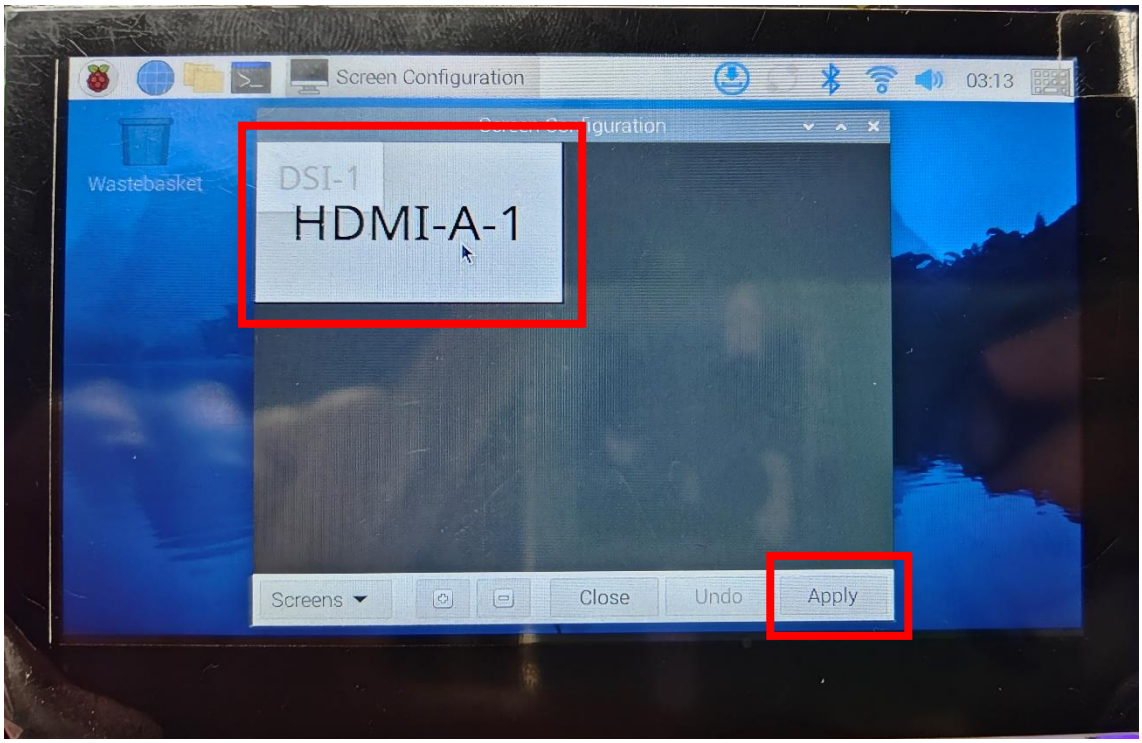
6.1 Overlapping Displaying Area

Do as following to overlap the display areas of DSI-1 and HDMI-A-1.
Click on the screen: applications menu ->Preferences -> Screen Configuration.



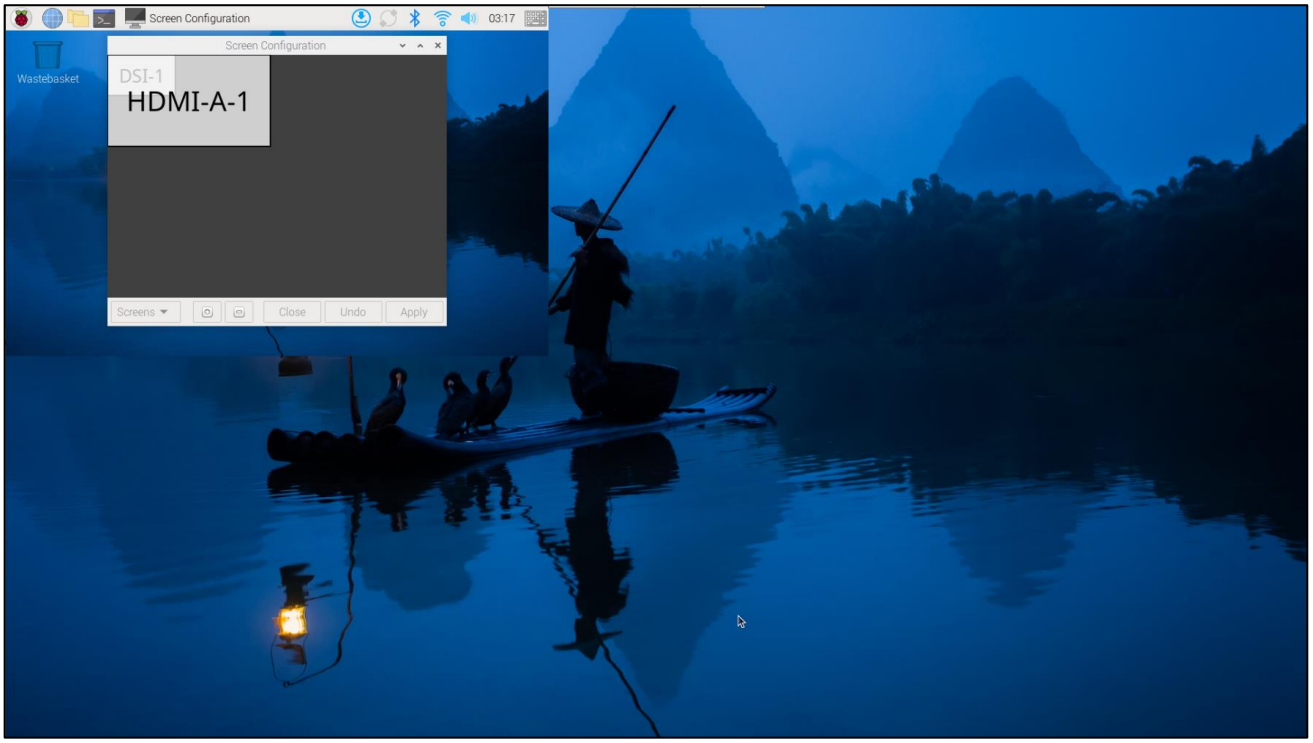


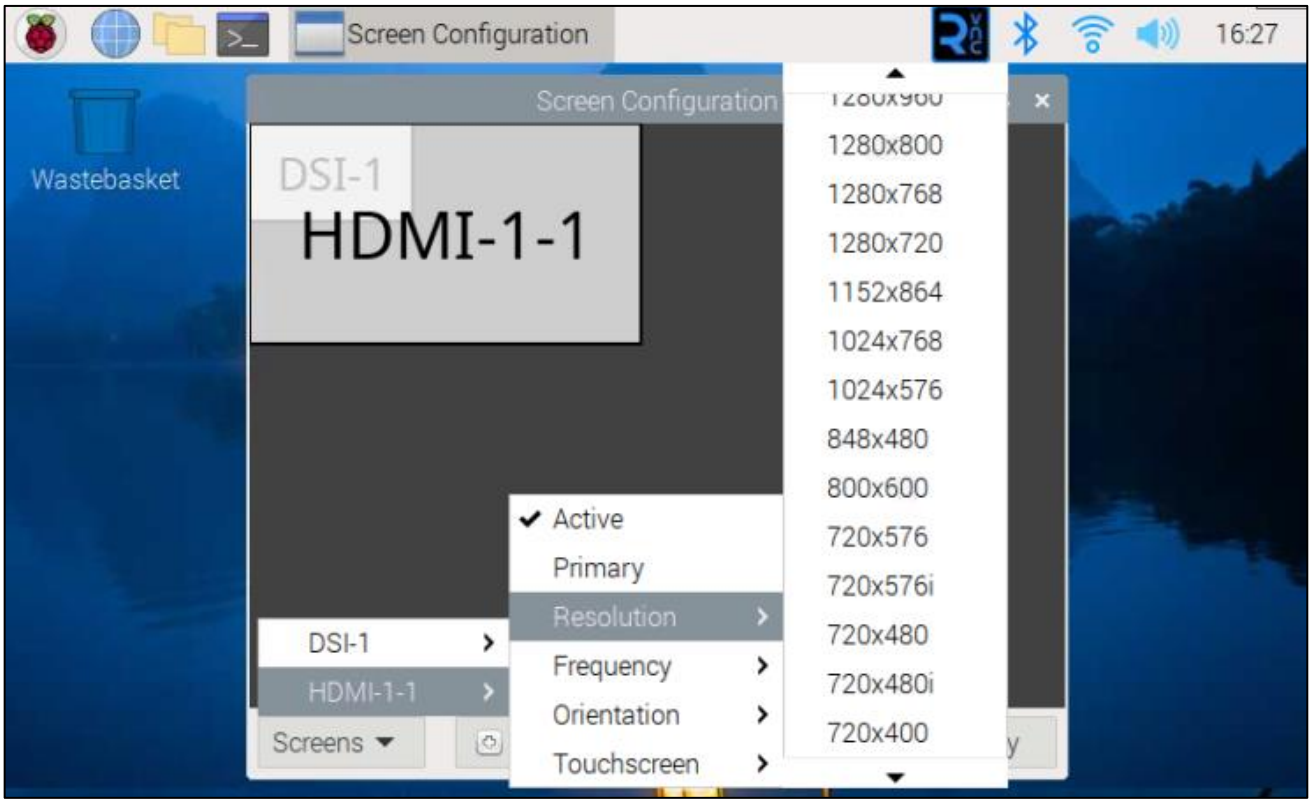
Overlap the two areas and click Apply ->OK.



6.2 Adding Resolution Option

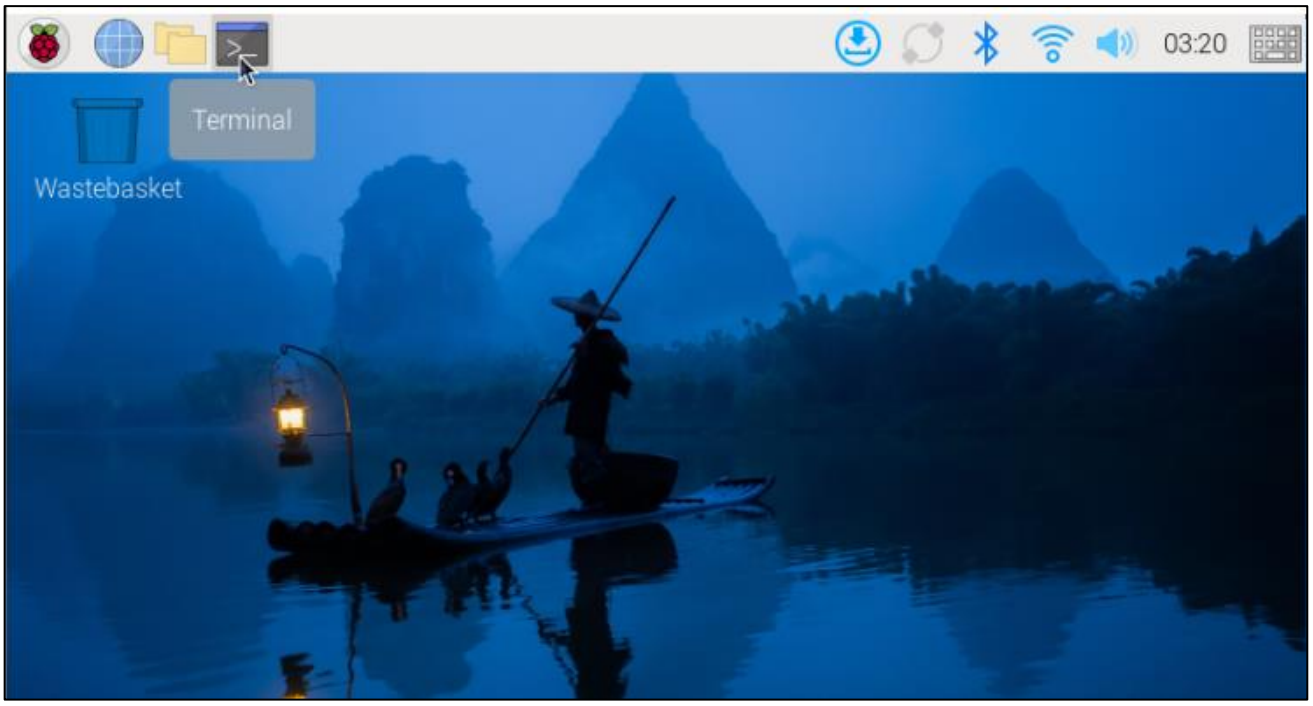
After overlapping, due to their different resolutions, the HDMI-A-1display area will be larger, causing the VNC display to show uneven or mismatched regions. Additionally, there is no 800×480 resolution option available for HDMI-A-1.





To configure the HDMI-A-1 with the same resolution (800x480) as that of DSI-1, we need to add a resolution option for HDMI-A-1 with the following steps:

Open Terminal:



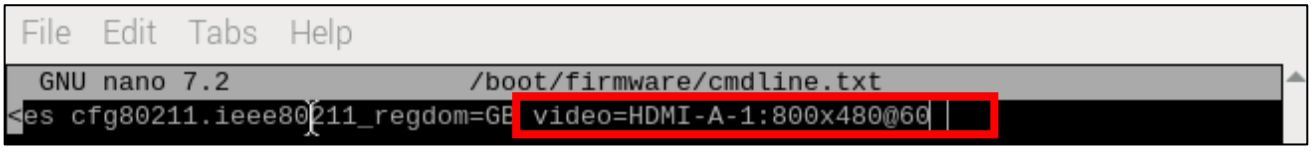
Run the following command:

```
sudo nano /boot/firmware/cmdline.txt
```

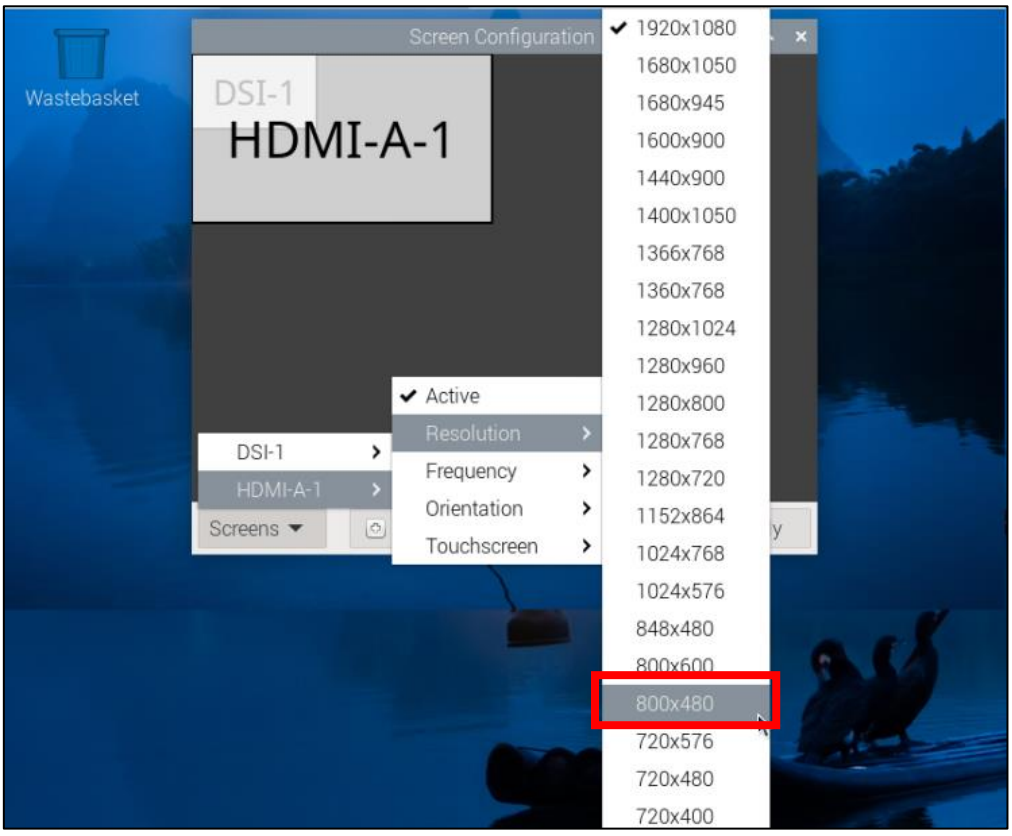
Put the following parameter (video=HDMI-A-1:800x480@60) on the same line with other parameters, separated with a space. Do not use newlines. Press Ctrl+S to save and Ctrl+X to exit. Then, restart your system (a restart is mandatory for the changes to take effect). Alternatively, you can manually adjust the resolution as needed.

https://www.raspberrypi.com/documentation/computers/configuration.html#kernel-command-line-cmdline-txt

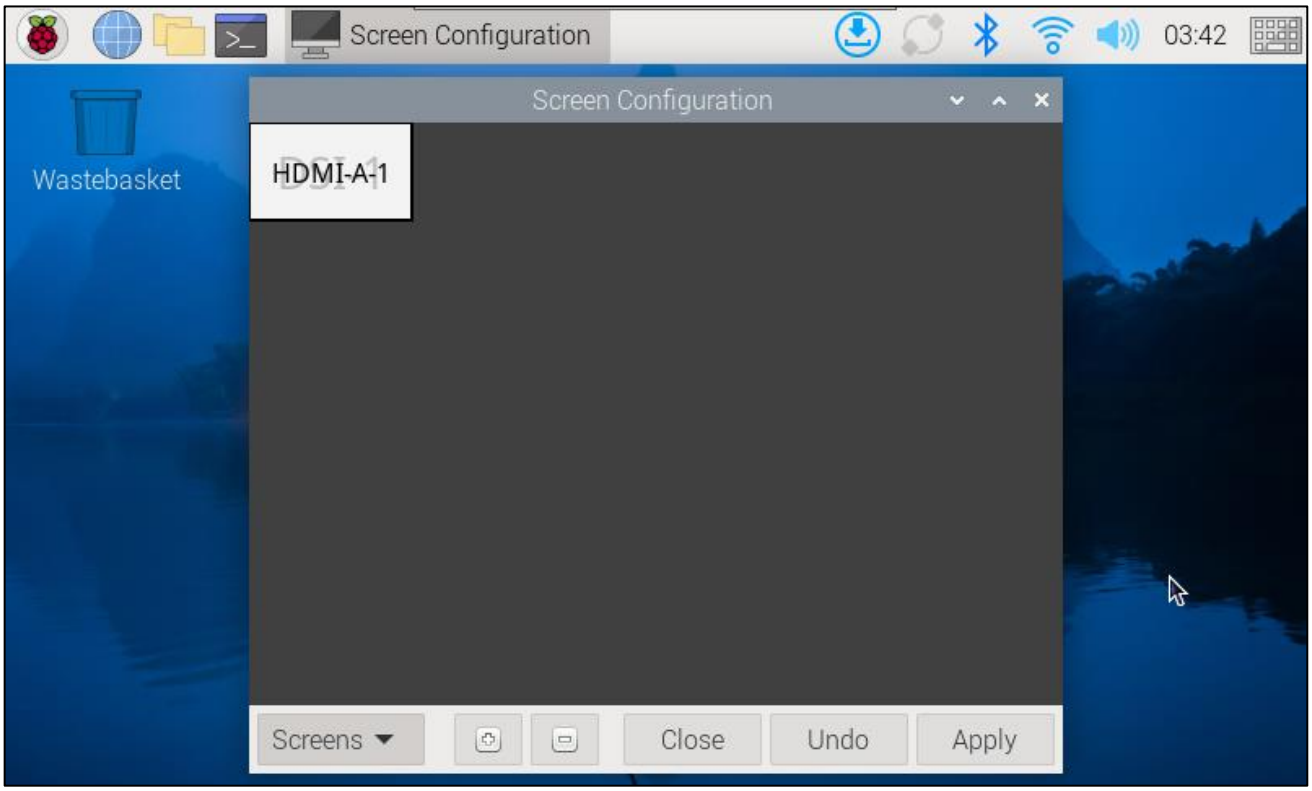
```
video=HDMI-A-1:800x480@60
```



After rebooting, configure the resolution for HDMI-A-1. Select Screens -> HDMI-A-1 -> Resolution -> 800x480 -> Apply -> OK.



After the configuration, you can see that the display area of HDMI-A-1 is the same as that of DSI-1.



6.3 Installing Full Version of VNC Server

The built-in VNC server on the Raspberry Pi is limited in functionality, and you may experience connection issues after completing the previous steps. To access advanced features such as bidirectional file transfer, clipboard sharing, and more, it is strongly recommended to install the full version of VNC Server. Please enter the following command to install it:

```
sudo apt-get install realvnc-vnc-server
```

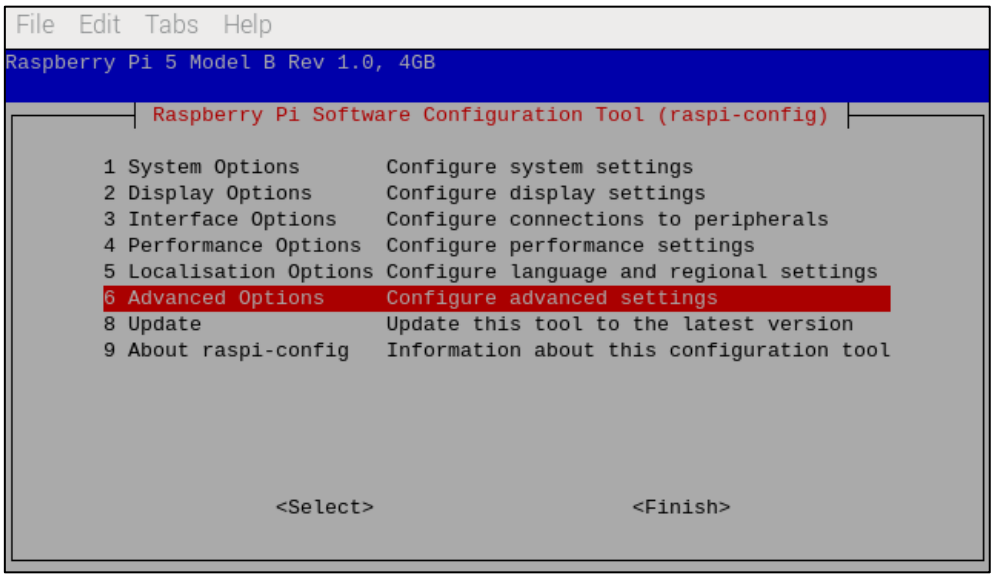
```
pi@raspberrypi:~ $ sudo apt-get install realvnc-vnc-server
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
realvnc-vnc-server is already the newest version (7.11.0.18).
0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
```

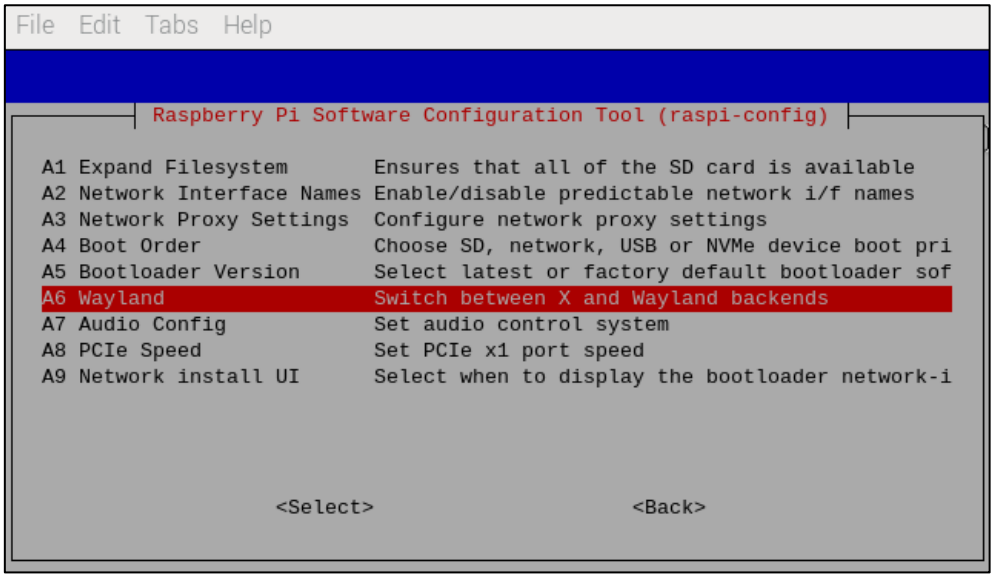
Run the following command to restart VNC service.

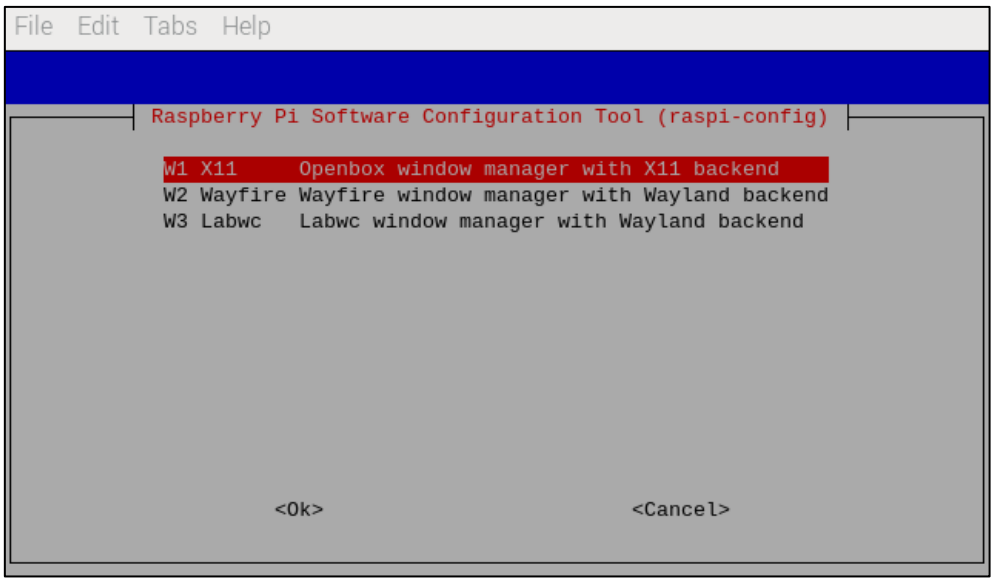
```
sudo systemctl restart vncserver-x11-serviced
```

```
pi@raspberrypi:~ $ sudo systemctl restart vncserver-x11-serviced
```

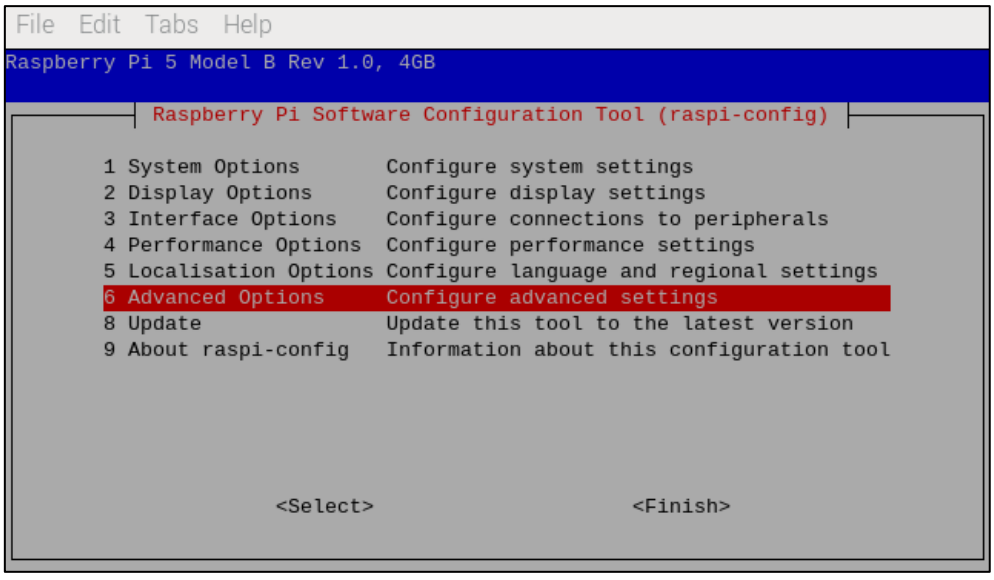
Run `sudo raspi-config`, select Advanced Options -> A6 Wayland-> W1 X11-> Enter -> OK.

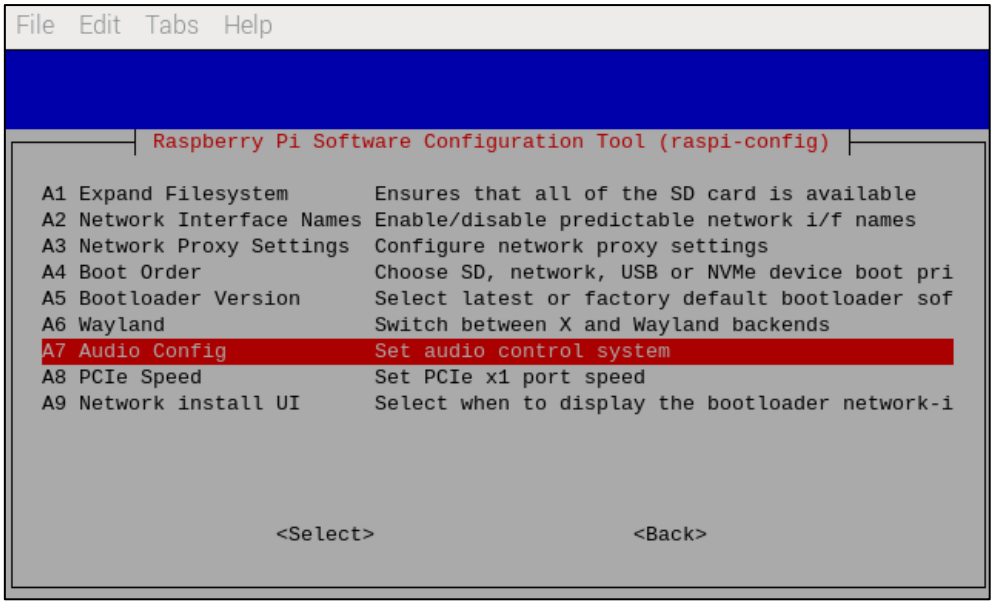


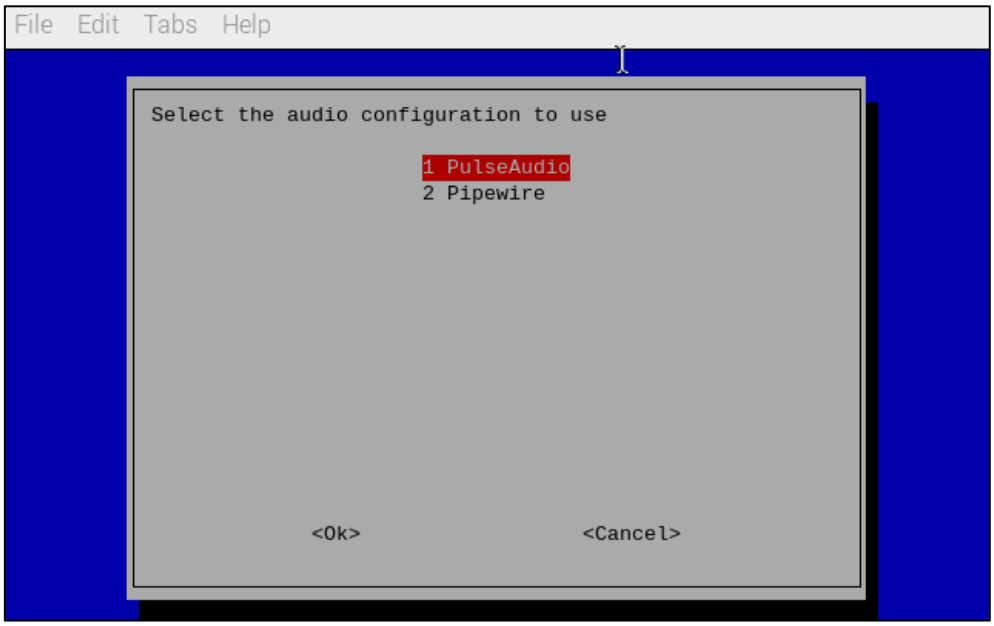




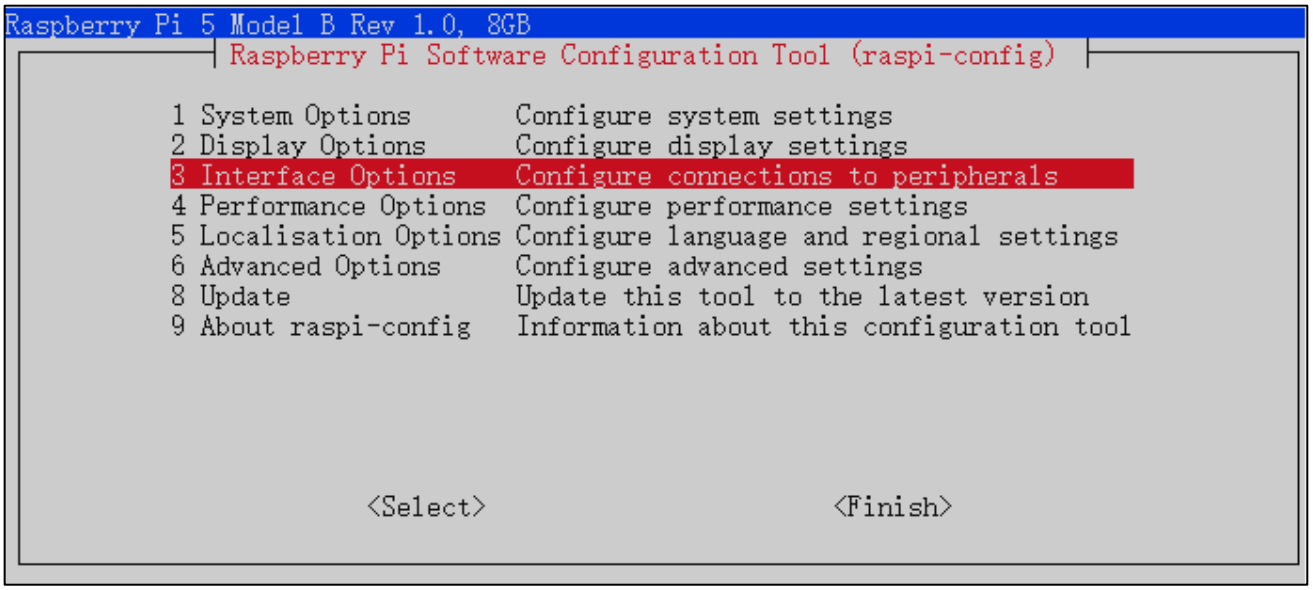
Select Advanced Options -> A7 Audio Config -> 1 PulseAudio -> Enter -> OK.

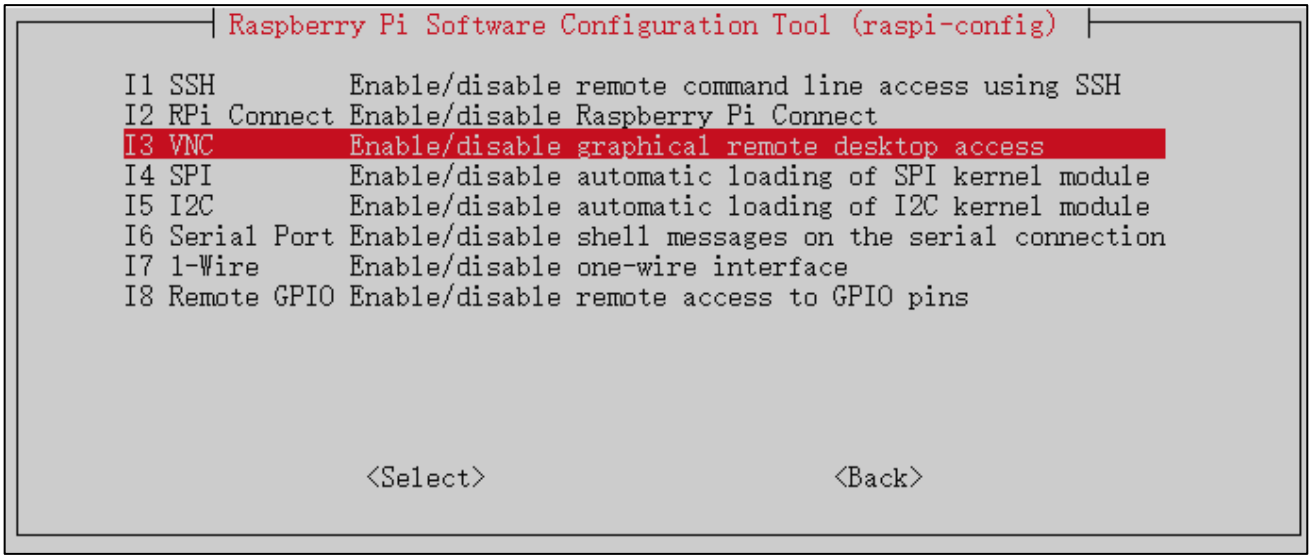



File Edit Tabs Help
Raspberry Pi Software Configuration Tool (raspi-config)
A1 Expand Filesystem Ensures that all of the SD card is available
A2 Network Interface Names Enable/disable predictable network i/f names
A3 Network Proxy Settings Configure network proxy settings
A4 Boot Order Choose SD, network, USB or NVMe device boot pri
A5 Bootloader Version Select latest or factory default bootloader sof
A6 Wayland Switch between X and Wayland backends
A7 Audio Config Set audio control system
A8 PCIe Speed Set PCIe x1 port speed
A9 Network install UI Select when to display the bootloader network-i
<Select>
<Back>


File Edit Tabs Help
Select the audio configuration to use
1 PulseAudio
2 Pipewire
<Ok>
<Cancel>

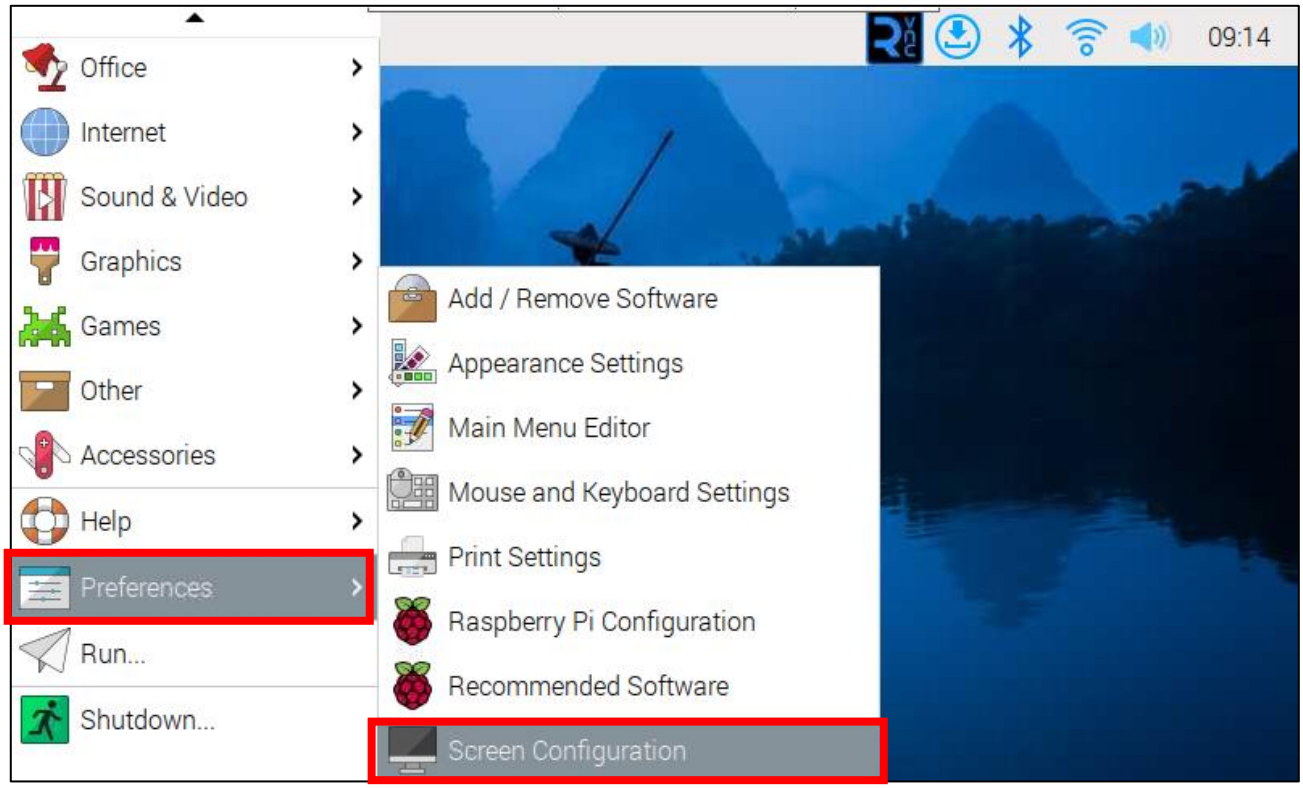
Select Interface Options -> I3 VNC -> Enter -> OK. Reboot your Raspberry Pi and open VNC viewer.





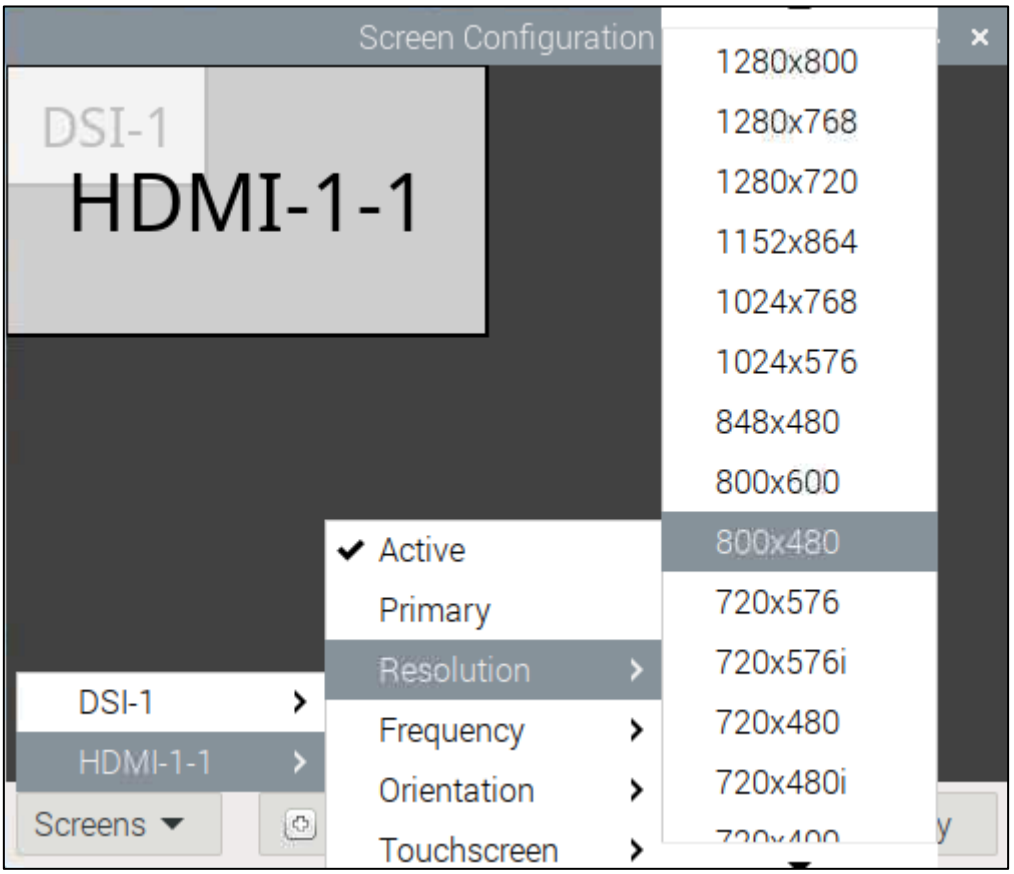
6.4 Restart HDMI

Activate HDMI again. Click Preferences -> Screen Configuration.



Select Screen -> HDMI-1-1 -> Active.

Select Screen -> HDMI-1-1 -> Resolution -> 800x480.



Chapter 7 Communication Protocols and Controls

7.1 OLED Control

Code

Below is the code for controlling OLED display.

```
from luma.core.interface.serial import i2c
from luma.oled.device import ssd1306
from PIL import Image, ImageDraw, ImageFont, ImageSequence
import time
import os
import shutil

class OLED:
    def __init__(self, bus_number=1, i2c_address=0x3C):
        # Initialize I2C interface and OLED display
        self.bus_number = bus_number
        self.i2c_address = i2c_address
        self.serial = i2c(port=self.bus_number, address=self.i2c_address)
        self.device = ssd1306(self.serial)
        self.buffer = Image.new('1', (self.device.width, self.device.height))
        self.draw = ImageDraw.Draw(self.buffer)

        self.default_font_path = "/usr/share/fonts/truetype/dejavu/DejaVuSans.ttf"
        self.default_font_size = 16
        self.font = ImageFont.load_default()

    def clear(self):
        # Clear the content in the buffer
        self.buffer = Image.new('1', (self.device.width, self.device.height))
        self.draw = ImageDraw.Draw(self.buffer)

    def show(self):
        # Display the content in the buffer on the OLED screen
        self.device.display(self.buffer)

    def close(self):
        # Close the I2C bus
        pass  # The luma.oled library does not require explicitly closing the I2C bus

```

```
    def draw_point(self, xy, fill=None):
        # Draw a point in the buffer
        self.draw.point(xy, fill=fill)

    def draw_line(self, xy, fill=None):
        # Draw a line in the buffer
        self.draw.line(xy, fill=fill)

    def draw_rectangle(self, xy, outline=None, fill=None):
        # Draw a rectangle in the buffer
        self.draw.rectangle(xy, outline=outline, fill=fill)

    def draw_ellipse(self, xy, outline=None, fill=None):
        # Draw an ellipse in the buffer
        self.draw.ellipse(xy, outline=outline, fill=fill)

    def draw_circle(self, xy, radius, outline=None, fill=None):
        # Draw a circle in the buffer
        self.draw.ellipse((xy[0] - radius, xy[1] - radius, xy[0] + radius, xy[1] + radius),
outline=outline, fill=fill)

    def draw_arc(self, xy, start, end, fill=None, width=1):
        # Draw an arc in the buffer
        self.draw.arc(xy, start, end, fill=fill, width=width)

    def draw_polygon(self, xy, outline=None, fill=None):
        # Draw a polygon in the buffer
        self.draw.polygon(xy, outline=outline, fill=fill)

    def draw_text(self, text, position=(0, 0), font_size=None):
        # Display text in the buffer
        if font_size is None:
            font = self.font
        else:
            font = ImageFont.truetype(self.default_font_path, font_size)
        self.draw.text(position, text, font=font, fill="white")

    def draw_image(self, image_path, position=(0, 0), resize=None):
        # Display an image in the buffer
        try:
            image = Image.open(image_path).convert('1')
            if resize is not None:
                image = image.resize(resize, Image.LANCZOS)
            else:
```

```
79                  image = image.resize((self.device.width, self.device.height), Image.LANCZOS)
80              self.buffer.paste(image, position)
81          except FileNotFoundError:
82              print(f"Error: File not found - {image_path}")
83          except Exception as e:
84              print(f"Error displaying image: {e}")
85
86      def draw_gif(self, gif_path, position=(0, 0), resize=None):
87          # Display a GIF animation
88          temp_folder = "temp"
89          if not os.path.exists(temp_folder):
90              os.makedirs(temp_folder)
91          try:
92              gif = Image.open(gif_path)
93              frames = []
94              frame_delays = []
95              for frame in ImageSequence.Iterator(gif):
96                  delay = frame.info.get('duration', 100) / 1000.0
97                  frame_delays.append(delay)
98                  width, height = frame.size
99                  target_height = height
100                 target_width = height * 2
101                 if width < target_width:
102                     new_image = Image.new('L', (target_width, target_height), 0)
103                     x_offset = (target_width - width) // 2
104                     new_image.paste(frame, (x_offset, 0))
105                 else:
106                     new_image = Image.new('L', (width, target_height), 0)
107                     y_offset = (target_height - height) // 2
108                     new_image.paste(frame, (0, y_offset))
109                     target_width = width
110                 new_image = new_image.convert('1')
111                 if resize is not None:
112                     new_image = new_image.resize(resize, Image.LANCZOS)
113                 else:
114                     new_image = new_image.resize((self.device.width, self.device.height),
115 Image.LANCZOS)
116                 new_image = new_image.resize((self.device.width, self.device.height),
117 Image.LANCZOS)
118                 frame_path = os.path.join(temp_folder, f"frame_{len(frames)}.png")
119                 new_image.save(frame_path)
120                 frames.append(frame_path)
121             for frame_path, delay in zip(frames, frame_delays):
122                 self.draw_image(frame_path, position = position, resize = resize)
```

```
123                 self.show()
124                 if delay > 0.17:
125                     time.sleep(delay - 0.17)
126         except FileNotFoundError:
127             print(f"Error: File not found - {gif_path}")
128         except Exception as e:
129             print(f"Error displaying GIF: {e}")
130         finally:
131             if os.path.exists(temp_folder):
132                 shutil.rmtree(temp_folder)
133
134     def save_buffer_to_image(self, image_path="saved_image.png"):
135         # Save the content in the buffer as an image file
136         self.buffer.save(image_path)
137
138
139 if __name__ == "__main__":
140     print("Starting OLED display example...")
141
142     oled = OLED()
143     try:
144         # Display text
145         print("Step 1: Displaying text 'Hello, World!'")
146         oled.draw_text("Hello, World!", position=(0, 0))
147         oled.show()
148         time.sleep(0.5)
149
150         # Draw a point
151         print("Step 2: Drawing point (64, 32)")
152         oled.draw_point((64, 32), fill="white")
153         oled.show()
154         time.sleep(0.5)
155
156         # Draw lines
157         print("Step 3: Drawing line ((0, 0), (127, 63)), ((0, 63), (127, 0))")
158         oled.draw_line(((0, 0), (127, 63)), fill="white")
159         oled.draw_line(((0, 63), (127, 0)), fill="white")
160         oled.show()
161         time.sleep(0.5)
162
163         # Draw a rectangle
164         print("Step 4: Drawing rectangle ((44, 12), (84, 52))")
165         oled.draw_rectangle(((44, 12), (84, 52)), outline="white", fill=None)
166         oled.show()
```

```
167         time.sleep(0.5)
168
169         # Draw an ellipse
170         print("Step 5: Drawing ellipse ((20, 20), (100, 60))")
171         oled.draw_ellipse(((20, 20), (100, 60)), outline="white", fill=None)
172         oled.show()
173         time.sleep(0.5)
174
175         # Draw a circle
176         print("Step 6: Drawing circle (64, 32) with radius 20")
177         oled.draw_circle((64, 32), 20, outline="white", fill=None)
178         oled.show()
179         time.sleep(0.5)
180
181         # Draw an arc
182         print("Step 7: Drawing arc ((10, 30), (110, 50)) from 0 to 180 degrees")
183         oled.draw_arc(((10, 30), (110, 50)), 0, 180, fill="white", width=1)
184         oled.draw_arc(((10, 30), (110, 50)), 180, 360, fill="white", width=1)
185         oled.show()
186         time.sleep(0.5)
187
188         # Draw a polygon
189         print("Step 8: Drawing polygon ((20, 20), (40, 40), (60, 20), (40, 0))")
190         oled.draw_polygon(((20, 20), (40, 40), (60, 20), (40, 0)), outline="white", fill=None)
191
192         oled.show()
193         time.sleep(0.5)
194
195         # Save the buffer content to an image file
196         # Display an image
197         print("Step 9: Displaying image './picture/1.bmp'")
198         oled.draw_image("./picture/1.bmp")
199         oled.show()
200         time.sleep(0.5)
201         oled.draw_image("./picture/2.png")
202         oled.show()
203         time.sleep(0.5)
204         oled.draw_image("./picture/3.jpg")
205         oled.show()
206         time.sleep(0.5)
207         oled.clear()
208
209         # Display a GIF animation
210         oled.draw_gif("./picture/1.gif")
```

```
211             time.sleep(0.5)
212
213             # Save the buffer content to an image file
214             #oled.save_buffer_to_image("1.bmp")
215             #oled.save_buffer_to_image("1.png")
216             #oled.save_buffer_to_image("1.jpg")
217         except Exception as e:
218             print(f"An error occurred: {e}")
219         except KeyboardInterrupt:
220             print("Keyboard interrupt detected. Exiting...")
221         finally:
222             # Clear the display
223             print("Step 10: Clearing display")
224             oled.clear()
225
226             # Close the display
227             print("Step 11: Closing OLED display")
228             oled.close()
```

Import the functional modules.

```
1   from luma.core.interface.serial import i2c
2   from luma.oled.device import ssd1306
3   from PIL import Image, ImageDraw, ImageFont, ImageSequence
4   import time
5   import os
6   import shutil
```

Create an OLDE class.

```
8   class OLED:
```

In the main function, instantiate an object by calling this class.

```
139 if __name__ == "__main__":
140     print("Starting OLED display example...")
141
142     oled = OLED()
```

Step 1: Display texts on the OLED screen.

```
143     try:
144         # Display text
145         print("Step 1: Displaying text 'Hello, World!'")
146         oled.draw_text("Hello, World!", position=(0, 0))
147         oled.show()
148         time.sleep(0.5)
```

Step 2: Draw a point on it.

```
150         # Draw a point
151         print("Step 2: Drawing point (64, 32)")
152         oled.draw_point((64, 32), fill="white")
153         oled.show()
154         time.sleep(0.5)
```

Step 3: Draw a line on the OLED display.

```
156         # Draw lines
157         print("Step 3: Drawing line ((0, 0), (127, 63)), ((0, 63), (127, 0))")
158         oled.draw_line(((0, 0), (127, 63)), fill="white")
159         oled.draw_line(((0, 63), (127, 0)), fill="white")
160         oled.show()
161         time.sleep(0.5)
```

Step 4: Draw a rectangle on the OLED display.

```
163         # Draw a rectangle
164         print("Step 4: Drawing rectangle ((44, 12), (84, 52))")
165         oled.draw_rectangle(((44, 12), (84, 52)), outline="white", fill=None)
166         oled.show()
167         time.sleep(0.5)
```

Step 5: Draw an ellipse on the OLED display.

```
169         # Draw an ellipse
170         print("Step 5: Drawing ellipse ((20, 20), (100, 60))")
171         oled.draw_ellipse(((20, 20), (100, 60)), outline="white", fill=None)
172         oled.show()
173         time.sleep(0.5)
```

Step 6: Draw a circle on the OLED display.

```
175         # Draw a circle
176         print("Step 6: Drawing circle (64, 32) with radius 20")
177         oled.draw_circle((64, 32), 20, outline="white", fill=None)
178         oled.show()
179         time.sleep(0.5)
```

Step 7: Draw an arc on the OLED display.

```
181         # Draw an arc
182         print("Step 7: Drawing arc ((10, 30), (110, 50)) from 0 to 180 degrees")
183         oled.draw_arc(((10, 30), (110, 50)), 0, 180, fill="white", width=1)
184         oled.draw_arc(((10, 30), (110, 50)), 180, 360, fill="white", width=1)
185         oled.show()
186         time.sleep(0.5)
```

Step 8: Draw a polygon on the OLED display.

```
        # Draw a polygon
        print("Step 8: Drawing polygon ((20, 20), (40, 40), (60, 20), (40, 0))")
        oled.draw_polygon(((20, 20), (40, 40), (60, 20), (40, 0)), outline="white", fill=None)

        oled.show()
        time.sleep(0.5)
```

Step 9: Display images and GIF animations on the OLED Screen

If an exception occurs during program execution, print the error message. If a keyboard interrupt is detected, exit the running program.

```
        # Save the buffer content to an image file
        # Display an image
        print("Step 9: Displaying image './picture/1.bmp'")
        oled.draw_image("./picture/1.bmp")
        oled.show()
        time.sleep(0.5)
        oled.draw_image("./picture/2.png")
        oled.show()
        time.sleep(0.5)
        oled.draw_image("./picture/3.jpg")
        oled.show()
        time.sleep(0.5)
        oled.clear()

        # Display a GIF animation
        oled.draw_gif("./picture/1.gif")
        time.sleep(0.5)

        # Save the buffer content to an image file
        #oled.save_buffer_to_image("1.bmp")
        #oled.save_buffer_to_image("1.png")
        #oled.save_buffer_to_image("1.jpg")
    except Exception as e:
        print(f"An error occurred: {e}")
    except KeyboardInterrupt:
        print("Keyboard interrupt detected. Exiting...")
```

Step 10: Clear the dislay.

```
    finally:
        # Clear the display
        print("Step 10: Clearing display")
        oled.clear()
```

Step 10: Turn OFF the OLDE display.

```
226         # Close the display
227         print("Step 11: Closing OLED display")
228         oled.close()
```

Reference

```
def __init__(self, bus_number=1, i2c_address=0x3C):
```

Initialize the I2C interface and the OLED display.

Parameters

bus_number: I2C bus, default is I2C1

i2c_address: I2C address, default is 0x3C.

```
def clear(self):
```

Clear all contents displayed on the screen.

```
def show(self):
```

Show contents on the OLED display.

```
def close(self):
```

Turn OFF the OLED display.

```
def draw_point(self, xy, fill=None):
```

Draw a point on the OLED screen.

Parameters

xy: Specifies the coordinates where the point should be drawn on the screen.

fill: The fill color for the point.

```
def draw_line(self, xy, fill=None):
```

Draw a line on the OLED screen.

Parameters

xy: Specifies the coordinates where the line should be drawn on the screen.

fill: The fill color for the line.

```
def draw_rectangle(self, xy, outline=None, fill=None):
```

Draw a rectangle on the OLED screen.

Parameters

xy: Specifies the coordinates where the rectangle should be drawn on the screen.

outline: The color of the rectangle's border.

fill: The color used to fill the inside of the rectangle.

```
def draw_ellipse(self, xy, outline=None, fill=None):
```

Draw an ellipse on the OLED screen.

Parameters

xy: Specifies the coordinates where the ellipse should be drawn on the screen.

outline: The color of the ellipse 's border.

fill: The color used to fill the inside of the ellipse.

```
def draw_circle(self, xy, radius, outline=None, fill=None):
```

Draw a circle on the OLED screen.

Parameters

xy: Specifies the coordinates where the circle should be drawn on the screen.

radius: Radius of the circle.
outline: The color of the circle 's border.
fill: The color used to fill the inside of the circle.

```
def draw_arc(self, xy, start, end, fill=None, width=1):
```

Draw an arc on the OLED screen.
Parameters
xy: Specifies the coordinates where the arc should be drawn on the screen.
start: The starting angle of the arc.
end: The ending angle of the arc.
fill: The color used to fill the inside of the arc.
width: The width of the arc's border.

```
def draw_polygon(self, xy, outline=None, fill=None):
```

Draw a polygon on the OLED screen.
Parameters
xy: Specifies the coordinates where the polygon should be drawn on the screen.
outline: The color of the polygon 's border.
fill: The color used to fill the inside of the polygon.

```
def draw_image(self, image_path, position=(0, 0), resize=None):
```

Display an image on the OLED screen.
Parameters
image_path: The path of the image files.
position(0,0): Starting pixel coordinates of the image.
resize: Adjust the resolution size of the image.

```
def draw_gif(self, gif_path, position=(0, 0), resize=None):
```

Display GIF animation on the OLED screen.
Parameters
gif_path: The path of the GIF animation file.
position(0,0): The starting pixel coordinates for the GIF animation on the OLED screen.
resize: Adjust the resolution size of the GIF animation.

```
def save_buffer_to_image(self, image_path="saved_image.png"):
```

Save the current image in the buffer.
Parameters
image_path: The name of the image to be saved in the current directory. Default is saved_image.png.

7.2 GPIO Board Control

7.2.1 GPIO Board Registers

The GPIO Board is controlled via the I2C interface on the Raspberry Pi 5. Below are the register addresses for the GPIO Board:

Register	Function	Register	KEY Value
0x00	Set I2C Address	0xF5	Get all LED colors
0x01	Set specified LED color	0xF6	Get LED mode
0x02	Set all LEDs' colors	0xF7	Get fan mode
0x03	Set LED mode	0xF8	Get fan frequency
0x04	Set fan mode	0xF9	Get Fan0 duty cycle
0x05	Set fan frequency	0xFA	Get Fan1 duty cycle
0x06	Set fan duty cycle	0xFB	Get temperature threshold
0x07	Set fan mode temperature threshold	0xFC	Get chassis temperature
0x08	Set power-on self-test feature	0xFD	Get brand name
0xF3	Get I2C Address	0xFE	Get firmware version
0xF4	Get specified LED color	0xFF	Save current data

We have flashed firmware onto the GPIO Board. The factory default I2C address of the GPIO Board is 0x21. Additionally, we have included some basic configurations within the firmware to control the LED lights and case fans. You can use the Raspberry Pi 5 to read from and write to the GPIO Board's registers in order to control the LED lights and fans.

LED lights

Regarding the LED lights, we have configured five modes:

0. **Off Mode**: In this mode, all LED lights are turned off.
1. **Static Display Mode**: The LED lights display a fixed color. You can set them to any desired color.
2. **Follow Display Mode**: The LED lights illuminate sequentially. You can set them to any desired color.
3. **Breathing Mode**: The brightness of the LED lights gradually increases and then decreases. You can set them to any desired color.
4. **Rainbow Mode**: The LED lights cycle through colors like a rainbow. The color cannot be customized in this mode.

Case Fans

Regarding the case fans, we have configured three modes:

0. **Off Mode**: In this mode, the fans remain inactive and do not start.
1. **Manual Mode**: You can manually set the fan speed by entering any duty cycle value between 0 and 255. This allows you to control the rotational speed of the fans precisely.

2. **Automatic Mode**: Upon reaching the minimum activation temperature, the fan will run at full speed, with the default activation temperature set to 30°C.

Cooling Down Behavior:

-The fans cease operation entirely when the internal temperature falls below 27°C.

Default Temperature Thresholds: The default temperature monitoring range is set between 30°C and 45°C. If needed, you can adjust this range by modifying the appropriate register settings.

Other Functions

1. Power-ON Check Function

To initiate the power-on check feature, change the value of register 0x07 to 1.

During power-up, the GPIO Board will be controlled to make the LED lights display in rainbow mode and start the case fans for 3 seconds.

2. Saving Current Configurations

If you want to save the current configurations such as the LED light colors, running modes, case fan duty cycles, and their running modes, modify the value of register 0xFF to 1.

The saving process usually takes around 20 milliseconds. Make sure to wait for more than 20 milliseconds before sending new data; otherwise, the new data won't be saved.

Important Note: The saving operation is a one -time action.If you need to save a new set of configurations, you must modify the register value again.

7.2.2 Code

```
# -*- coding: utf-8 -*-
import smbus
import time

class Expansion:
    IIC_ADDRESS = 0x21
    REG_I2C_ADDRESS = 0x00        # Set I2C address
    REG_LED_SPECIFIED = 0x01      # Set specified LED color
    REG_LED_ALL = 0x02            # Set all LEDs color
    REG_LED_MODE = 0x03           # Set LED running mode
    REG_FAN_MODE = 0x04           # Set fan running mode
    REG_FAN_FREQUENCY = 0x05      # Set fan frequency
    REG_FAN_DUTY = 0x06           # Set fan duty cycle
    REG_FAN_THRESHOLD = 0x07      # Set fan temperature threshold
    REG_POWER_ON_CHECK = 0x08     # Set power-on check
```

```
    REG_SAVE_FLASH = 0xff          # Save to flash

    REG_I2C_ADDRESS_READ = 0xf3    # Read I2C address
    REG_LED_SPECIFIED_READ = 0xf4 # Read specified LED color
    REG_LED_ALL_READ = 0xf5        # Read all LEDs color
    REG_LED_MODE_READ = 0xf6       # Read LED mode
    REG_FAN_MODE_READ = 0xf7       # Read fan mode
    REG_FAN_FREQUENCY_READ = 0xf8 # Read fan frequency
    REG_FAN0_DUTY_READ = 0xf9      # Read fan duty cycle 1 value
    REG_FAN1_DUTY_READ = 0xfa      # Read fan duty cycle 2 value
    REG_FAN_THRESHOLD_READ = 0xfb # Read fan temperature threshold
    REG_TEMP_READ = 0xfc           # Read temperature value
    REG_BRAND = 0xfd               # Read brand
    REG_VERSION = 0xfe             # Read version

    def __init__(self, bus_number=1, address=IIC_ADDRESS):
        # Initialize I2C bus and address
        self.bus_number = bus_number
        self.bus = smbus.SMBus(self.bus_number)
        self.address = address

    def write(self, reg, values):
        # Write data to I2C register
        try:
            if isinstance(values, list):
                self.bus.write_i2c_block_data(self.address, reg, values)
            else:
                self.bus.write_byte_data(self.address, reg, values)
        except IOError as e:
            print("Error writing to I2C bus:", e)

    def read(self, reg, length=1):
        # Read data from I2C register
        if length == 1:
            return self.bus.read_byte_data(self.address, reg)
        else:
            return self.bus.read_i2c_block_data(self.address, reg, length)

    def end(self):
        # Close I2C bus
        self.bus.close()

    def set_i2c_addr(self, addr):
        # Set I2C address
```

```
60          self.address = addr
61          self.write(self.REG_I2C_ADDRESS, addr)
62
63      def set_led_color(self, led_id, r, g, b):
64          # Set color for specified LED
65          cmd = [led_id, r, g, b]
66          self.write(self.REG_LED_SPECIFIED, cmd)
67
68      def set_all_led_color(self, r, g, b):
69          # Set color for all LEDs
70          cmd = [r, g, b]
71          self.write(self.REG_LED_ALL, cmd)
72
73      def set_led_mode(self, mode):
74          # Set LED running mode
75          self.write(self.REG_LED_MODE, mode)
76
77      def set_fan_mode(self, mode):
78          # Set fan running mode
79          self.write(self.REG_FAN_MODE, mode)
80
81      def set_fan_frequency(self, freq):
82          # Set fan frequency
83          frequency = [
84              (freq >> 24) & 0xFF,
85              (freq >> 16) & 0xFF,
86              (freq >> 8) & 0xFF,
87              freq & 0xFF
88          ]
89          self.write(self.REG_FAN_FREQUENCY, frequency)
90
91      def set_fan_duty(self, duty0, duty1):
92          # Set fan duty cycle
93          duty = [duty0, duty1]
94          self.write(self.REG_FAN_DUTY, duty)
95
96      def set_fan_threshold(self, low_threshold, high_threshold):
97          # Set fan temperature threshold
98          threshold = [low_threshold, high_threshold]
99          self.write(self.REG_FAN_THRESHOLD, threshold)
100
101     def set_power_on_check(self, state):
102         # Set power-on check state
103         self.write(self.REG_POWER_ON_CHECK, state)
```

```
104
105     def set_save_flash(self, state):
106         # Save configuration to flash
107         self.write(self.REG_SAVE_FLASH, state)
108
109     def get_iic_addr(self):
110         # Get I2C address
111         return self.read(self.REG_I2C_ADDRESS_READ)
112
113     def get_led_color(self, led_id):
114         # Get color for specified LED
115         cmd = [led_id]
116         self.write(self.REG_LED_SPECIFIED, cmd)
117         return self.read(self.REG_LED_SPECIFIED_READ, 3)
118
119     def get_all_led_color(self):
120         # Get color for all LEDs
121         return self.read(self.REG_LED_ALL_READ, 12)
122
123     def get_led_mode(self):
124         # Get LED running mode
125         return self.read(self.REG_LED_MODE_READ)
126
127     def get_fan_mode(self):
128         # Get fan running mode
129         return self.read(self.REG_FAN_MODE_READ)
130
131     def get_fan_frequency(self):
132         # Get fan frequency
133         arr = self.read(self.REG_FAN_FREQUENCY_READ, 4)
134         freq = (arr[0] << 24) | (arr[1] << 16) | (arr[2] << 8) | arr[3];
135         return freq
136
137     def get_fan0_duty(self):
138         # Get fan duty cycle 1 value
139         return self.read(self.REG_FAN0_DUTY_READ)
140
141     def get_fan1_duty(self):
142         # Get fan duty cycle 2 value
143         return self.read(self.REG_FAN1_DUTY_READ)
144
145     def get_fan_threshold(self):
146         # Get fan temperature threshold
147         return self.read(self.REG_FAN_THRESHOLD_READ, 2)
```

```
148
149     def get_temp(self):
150         # Get temperature value
151         return self.read(self.REG_TEMP_READ)
152
153     def get_brand(self):
154         # Get brand information
155         brand_bytes = self.read(self.REG_BRAND, 9)
156         return ''.join(chr(b) for b in brand_bytes).rstrip('\x00')
157
158     def get_version(self):
159         # Get version information
160         version_bytes = self.read(self.REG_VERSION, 14)
161         return ''.join(chr(b) for b in version_bytes).rstrip('\x00')
162
163 if __name__ == '__main__':
164     expansion_board = Expansion()
165     try:
166         '''
167         # Below is the default configuration for extension, which you can comment out if you
168 are not using it.
169         # As long as expansion_board.set_save_flash(1) is not used,
170         # all configurations will be temporary,
171         # and will revert to default Settings when expansion_board.set_save_flash(1) is
172 powered off.
173         '''
174         ''''
175         print("Config expansion board ...")
176         expansion_board.set_i2c_addr(expansion_board.IIC_ADDRESS)
177         expansion_board.set_all_led_color(255,255,255) # set all led color: r,g,b. 0~255
178         expansion_board.set_led_mode(4)                # led mode: 4# 1: RGB, 2: Following, 3:
179 Breathing, 4: Rainbow
180         expansion_board.set_fan_mode(2)                # fan mode: 1: Manual Mode, 2: Auto
181 Mode
182         expansion_board.set_fan_duty(0, 0)             # Set the fan 1 and fan 2 duty cycle,
183 0~255
184         expansion_board.set_fan_threshold(30, 45)      # Set the temperature threshold, (low
185 temperature, high temperature)
186         expansion_board.set_power_on_check(1)          # Set power-on check state, 1: Enable,
187 0: Disable
188         expansion_board.set_save_flash(1)              # Save configuration to flash, 1:
189 Enable, 0: Disable
190         time.sleep(0.5)
191         print("Config expansion board done!")
```

```
192         '''
193
194         count = 0
195         expansion_board.set_all_led_color(0, 0, 255)
196         expansion_board.set_fan_duty(0, 0)
197         expansion_board.set_led_mode(2)
198         expansion_board.set_fan_mode(1)
199         expansion_board.set_fan_frequency(50)
200         expansion_board.set_fan_duty(0, 0)
201         time.sleep(3)
202         while True:
203             count += 1
204             if count % 5 == 0:
205                 print("get iic addr: 0x{:02X}".format(expansion_board.get_iic_addr()))
206                 print("get all led color:", expansion_board.get_all_led_color())
207                 print("get led mode:", expansion_board.get_led_mode())
208                 print("get fan mode:", expansion_board.get_fan_mode())
209                 print("get fan frequency:", expansion_board.get_fan_frequency())
210                 print("get fan0 duty:", expansion_board.get_fan0_duty())
211                 print("get fan1 duty:", expansion_board.get_fan1_duty())
212                 print("get fan threshold:", expansion_board.get_fan_threshold())
213                 print("get temp:", expansion_board.get_temp())
214                 print("get brand:", expansion_board.get_brand())
215                 print("get version:", expansion_board.get_version())
216                 print("")
217             time.sleep(0.03)
218
219     except Exception as e:
220         print("Exception:", e)
221     except KeyboardInterrupt:
222         print("KeyboardInterrupt")
223     finally:
224         expansion_board.set_all_led_color(0, 0, 0)
225         expansion_board.end()
```

Import functional modules.

```
2   import smbus
3   import time
```

Create an Expansion class and define the I2C address and register address.

```
5   class Expansion:
6       IIC_ADDRESS = 0x21
7       REG_I2C_ADDRESS = 0x00        # Set I2C address
8       REG_LED_SPECIFIED = 0x01      # Set specified LED color
```

```
    REG_LED_ALL = 0x02            # Set all LEDs color
    REG_LED_MODE = 0x03           # Set LED running mode
    REG_FAN_MODE = 0x04           # Set fan running mode
    REG_FAN_FREQUENCY = 0x05      # Set fan frequency
    REG_FAN_DUTY = 0x06           # Set fan duty cycle
    REG_FAN_THRESHOLD = 0x07      # Set fan temperature threshold
    REG_POWER_ON_CHECK = 0x08     # Set power-on check
    REG_SAVE_FLASH = 0xff          # Save to flash

    REG_I2C_ADDRESS_READ = 0xf3   # Read I2C address
    REG_LED_SPECIFIED_READ = 0xf4 # Read specified LED color
    REG_LED_ALL_READ = 0xf5       # Read all LEDs color
    REG_LED_MODE_READ = 0xf6      # Read LED mode
    REG_FAN_MODE_READ = 0xf7      # Read fan mode
    REG_FAN_FREQUENCY_READ = 0xf8 # Read fan frequency
    REG_FAN0_DUTY_READ = 0xf9     # Read fan duty cycle 1 value
    REG_FAN1_DUTY_READ = 0xfa     # Read fan duty cycle 2 value
    REG_FAN_THRESHOLD_READ = 0xfb # Read fan temperature threshold
    REG_TEMP_READ = 0xfc          # Read temperature value
    REG_BRAND = 0xfd              # Read brand
    REG_VERSION = 0xfe            # Read version
```

Instantiate an object by calling this class.

```
if __name__ == '__main__':
    expansion_board = Expansion()
```

Set the color of all LED lights, set the case fans' duty cycle, set the LED light mode, set the fan mode.

```
        count = 0
        expansion_board.set_all_led_color(0, 0, 255)
        expansion_board.set_fan_duty(0, 0)
        expansion_board.set_led_mode(2)
        expansion_board.set_fan_mode(1)
        expansion_board.set_fan_frequency(50)
        expansion_board.set_fan_duty(0, 0)
        time.sleep(3)
```

Every 0.03 seconds, increment the duty cycle of the case fans by 1. Every 0.15 seconds, retrieve the following data: I2C address, colors of all LED lights, LED light mode, case fan mode, duty cycles of both case fans, temperature thresholds, temperature readings, developer information, and version number. Then, print out this information.

```
        while True:
            count += 1
            if count % 5 == 0:
                print("get iic addr: 0x{:02X}".format(expansion_board.get_iic_addr()))
```

```
206                 print("get all led color:", expansion_board.get_all_led_color())
207                 print("get led mode:", expansion_board.get_led_mode())
208                 print("get fan mode:", expansion_board.get_fan_mode())
209                 print("get fan frequency:", expansion_board.get_fan_frequency())
210                 print("get fan0 duty:", expansion_board.get_fan0_duty())
211                 print("get fan1 duty:", expansion_board.get_fan1_duty())
212                 print("get fan threshold:", expansion_board.get_fan_threshold())
213                 print("get temp:", expansion_board.get_temp())
214                 print("get brand:", expansion_board.get_brand())
215                 print("get version:", expansion_board.get_version())
216                 print("")
217             time.sleep(0.03)
```

If an exception occurs during program execution, print the error value; if a keyboard input interruption is detected, exit the running program; finally, set the color of all LED lights to 0 and close the I2C communication.

```
219     except Exception as e:
220         print("Exception:", e)
221     except KeyboardInterrupt:
222         print("KeyboardInterrupt")
223     finally:
224         expansion_board.set_all_led_color(0, 0, 0)
225         expansion_board.end()
```

Reference

```
def __init__(self, bus_number=1, address=IIC_ADDRESS):
```

Initialize I2C interface and set the I2C address.

Parameters

bus_number: I2C bus, default is I2C1

address: I2C address

```
def write(self, reg, values):
```

Write a value to the specified register.

Parameters

reg: the specified register

value: the value to be sent

```
def read(self, reg, length=1):
```

Retrieve a value from the specified register.

Parameters

reg: the specified register

length: the length of the value to be retrieved

```
def end(self):
```

Close the I2C communication.

```
def set_i2c_addr(self, addr):
```

Set the I2C address.

Parameters

addr: The I2C address to be set, with a range of 0 - 255.

```
def set_led_color(self, led_id, r, g, b):
```

Set the color of the specified LED light.

Parameters

led_id: number of the specified LED light, with a range of 0 - 3.

r: The red component value for setting the color of all LED lights. The range is from 0 to 255.

g: The green component value for setting the color of all LED lights. The range is from 0 to 255.

b: The blue component value for setting the color of all LED lights. The range is from 0 to 255.

```
def set_all_led_color(self, r, g, b):
```

Set the color of all LED lights.

Parameters

r: The red component value for setting the color of all LED lights. The range is from 0 to 255.

g: The green component value for setting the color of all LED lights. The range is from 0 to 255.

b: The blue component value for setting the color of all LED lights. The range is from 0 to 255.

```
def set_led_mode(self, mode):
```

Set the mode of the LED lights.

Parameters

mode: The mode of the LED lights. The range of the parameter is from 0 to 4.

```
def set_fan_mode(self, mode):
```

Set the mode of the case fans.

Parameters

mode: The mode of the case fans. The range of the parameter is from 0 to 2.

```
def set_fan_frequency(self, freq):
```

Set the frequency of the case fans.

Parameters

freq: the frequency value of the case fan.

```
def set_fan_duty(self, duty0, duty1):
```

Set the duty cycle values of the case fans.

Parameters

duty0: the duty cycle value of case fan FAN0

duty1: the duty cycle value of chassis fan FAN1

```
def set_fan_threshold(self, low_threshold, high_threshold):
```

Set the temperature threshold range for the automatic mode of the case fans.

Parameters

low_threshold: the lower limit of the temperature threshold

high_threshold: the upper limit of the temperature threshold

```
def set_power_on_check(self, state):
```

Set the state of power-on check.

Parameters

state: The state of power-on check, with 0 as disabled and 1 as enabled.

```
def set_save_flash(self, state):
```

Save the current configuration to flash. **Parameters** **state:** The state of configuration saving, with 0 meaning not saved, and 1 referring to saved.
`def get_iic_addr(self):`
Get the I2C address.
`def get_led_color(self, led_id):`
Get the color data of the specified colored light. **Parameters** **led_id:** The number of the specified LED light, with a range from 0 to 3.
`def get_all_led_color(self):`
Get the color data of all LED lights.
`def get_led_mode(self):`
Get the mode of the LED lights.
`def get_fan_mode(self):`
Get the mode of the case fans.
`def get_fan_frequency(self):`
Get the frequency value of the case fan.
`def get_fan0_duty(self):`
Get the duty cycle value of case fan FAN0.
`def get_fan1_duty(self):`
Get the duty cycle value of case fan FAN1.
`def get_fan_threshold(self):`
Get the temperature threshold data of the case fans.
`def get_temp(self):`
Get the internal temperature of the computer case.
`def get_brand(self):`
Get the brand name.
`def get_version(self):`
Get the current firmware version number.

What's next?

Thank you again for choosing Freenove products.

THANK YOU for participating in this learning experience!

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

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

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

Thank you again for choosing Freenove products.