

# Welcome

Thank you for choosing Freenove products!

## Getting Started

### Thank you for choosing Freenove products!

After you download the ZIP file we provide. Unzip it and you will get a folder contains several files and folders. There are two PDF files:

- **Tutorial.pdf**

It contains basic operations such as installing system for Raspberry Pi.

**The code in this PDF is in C and Python.**

## 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](mailto:support@freenove.com)**

## Safety and Precautions

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

- Keep this product out of the reach of children under 6 years old.
- This product should be used only when there is adult supervision present as young children lack necessary judgment regarding safety and the consequences of product misuse.
- This product contains small parts and parts, which are sharp. This product contains electrically conductive parts. Use caution with electrically conductive parts near or around power supplies, batteries and powered (live) circuits.
- When the product is turned ON, activated or tested, some parts will move or rotate. To avoid injuries to hands and fingers, keep them away from any moving parts!
- It is possible that an improperly connected or shorted circuit may cause overheating. Should this happen, immediately disconnect the power supply or remove the batteries and do not touch anything until it



cools down! When everything is safe and cool, review the product tutorial to identify the cause.

- Only operate the product in accordance with the instructions and guidelines of this tutorial, otherwise parts may be damaged or you could be injured.
- Store the product in a cool dry place and avoid exposing the product to direct sunlight.
- After use, always turn the power OFF and remove or unplug the batteries before storing.

## About Freenove

Freenove provides open source electronic products and services worldwide.

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

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

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

<http://www.freenove.com>

## Copyright

All the files, materials and instructional guides provided are released under [Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License](#). A copy of this license can be found in the folder containing the Tutorial and software files associated with this product.



This means you can use these resources in your own derived works, in part or completely, but **NOT for the intent or purpose of commercial use.**

Freenove brand and logo are copyright of Freenove Creative Technology Co., Ltd. and cannot be used without written permission.



®

Raspberry Pi® is a trademark of Raspberry Pi Foundation (<https://www.raspberrypi.org/>).

# Contents

Welcome .....	I
Getting Started .....	I
Safety and Precautions.....	I
About Freenove .....	II
Copyright .....	II
Contents .....	III
Preface .....	1
Raspberry Pi .....	2
Installing an Operating System .....	9
Component List .....	9
Optional Components.....	11
Raspberry Pi OS.....	13
Getting Started with Raspberry Pi .....	18
Chapter 0 Preparation .....	27
Linux Command .....	27
Install WiringPi .....	30
Obtain the Project Code.....	32
Python2 & Python3.....	33
Chapter 1 LCD1602.....	35
Project 1.1 I2C LCD1602 .....	35
Chapter 2 LCD2004.....	55
Project 2.1 I2C LCD2004.....	55
What's Next? .....	72



## Preface

Raspberry Pi is a low cost, **credit card sized computer** that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is an incredibly capable little device that enables people of all ages to explore computing, and to learn how to program in a variety of computer languages like Scratch and Python. It is capable of doing everything you would expect from a desktop computer, such as browsing the internet, playing high-definition video content, creating spreadsheets, performing word-processing, and playing video games. For more information, you can refer to Raspberry Pi official [website](#). For clarification, this tutorial will also reference Raspberry Pi as RPi, RPI and RasPi.

We provide both C and Python code for each project in this tutorial.

This kit does not contain [\*\*Raspberry and its accessories\*\*](#). You can also use the components and modules in this kit to create projects of your own design.

Additionally, if you encounter any issues or have questions about this tutorial or the contents of kit, you can always contact us for free technical support at:

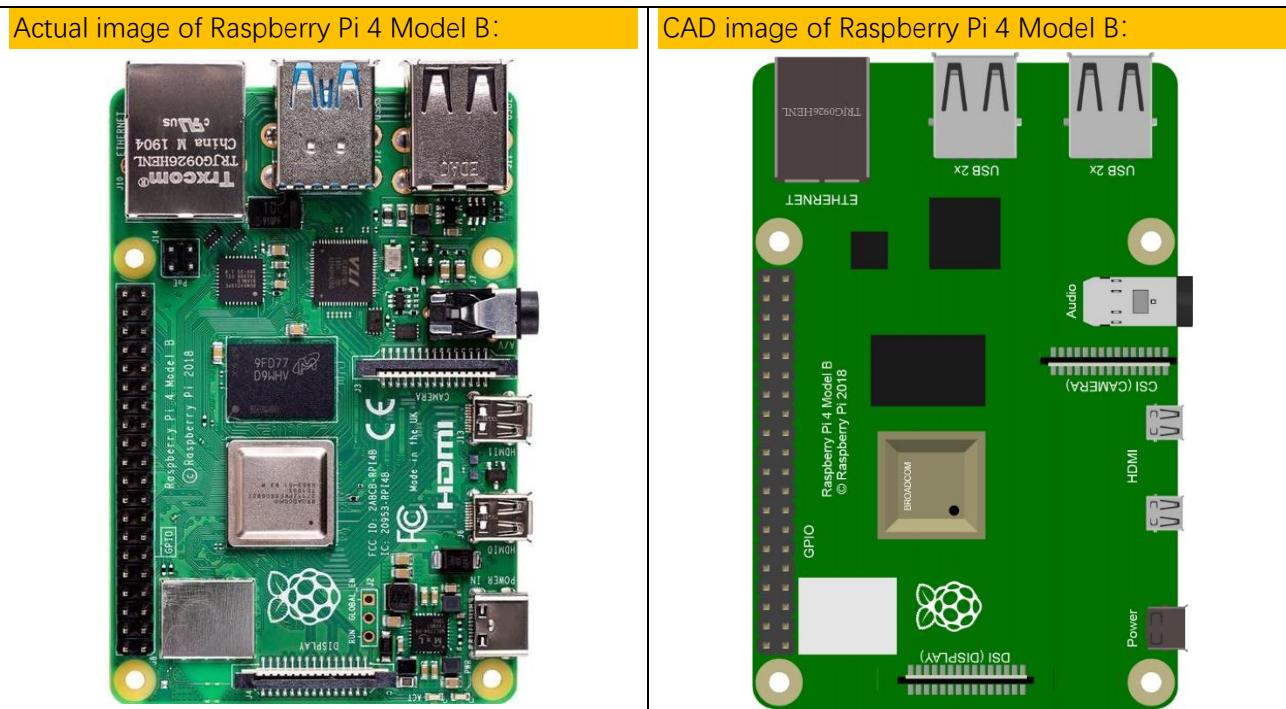
**[support@freenove.com](mailto:support@freenove.com)**

# Raspberry Pi

So far, at this writing, Raspberry Pi has advanced to its fourth generation product offering. Version changes are accompanied by increases in upgrades in hardware and capabilities.

The A type and B type versions of the first generation products have been discontinued due to various reasons. What is most important is that other popular and currently available versions are consistent in the order and number of pins and their assigned designation of function, making compatibility of peripheral devices greatly enhanced between versions.

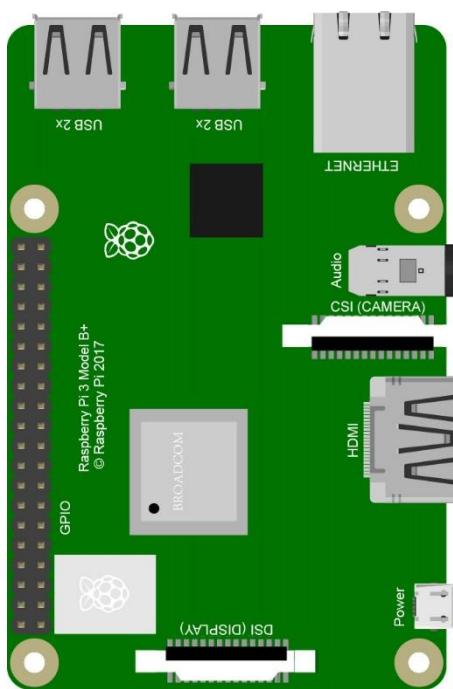
Below are the raspberry pi pictures and model pictures supported by this product. They have 40 pins.



Actual image of Raspberry Pi 3 Model B+:



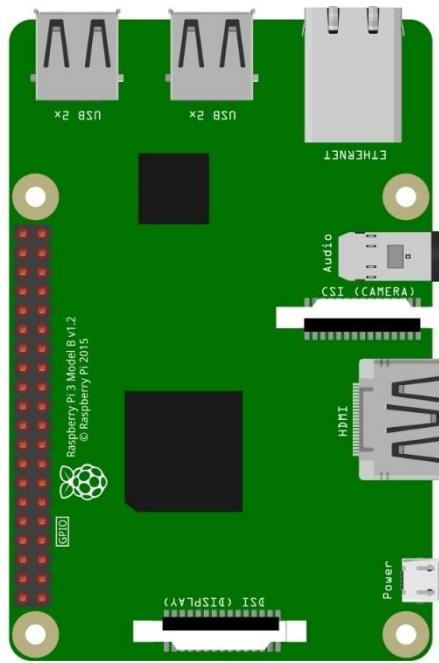
CAD image of Raspberry Pi 3 Model B+:



Actual image of Raspberry Pi 3 Model B:



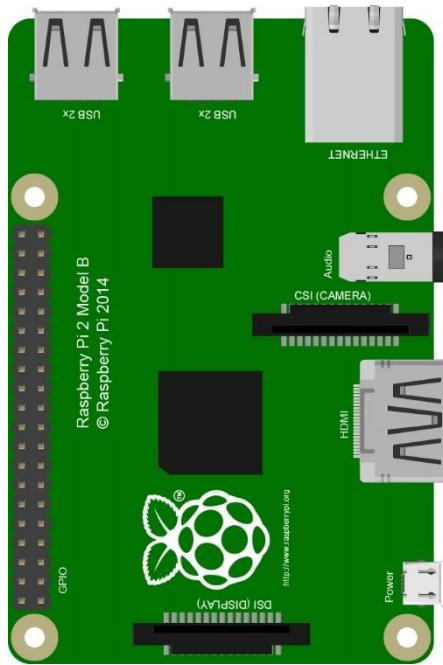
CAD image of Raspberry Pi 3 Model B:



Actual image of Raspberry Pi 2 Model B:



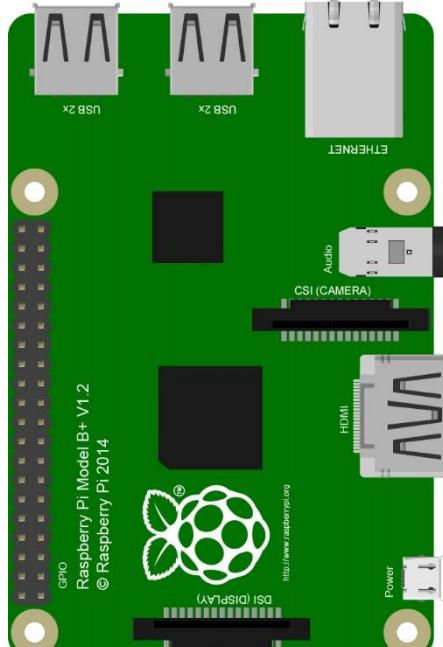
CAD image of Raspberry Pi 2 Model B:



Actual image of Raspberry Pi 1 Model B+:



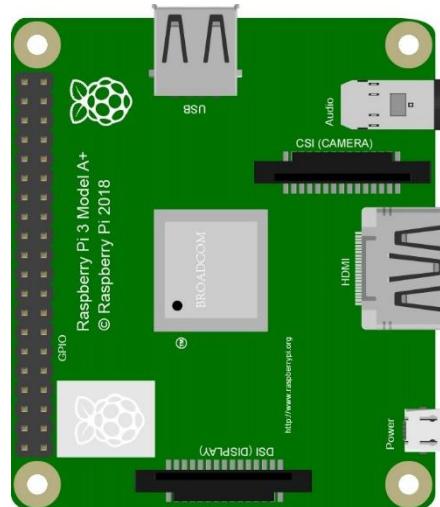
CAD image of Raspberry Pi 1 Model B+:



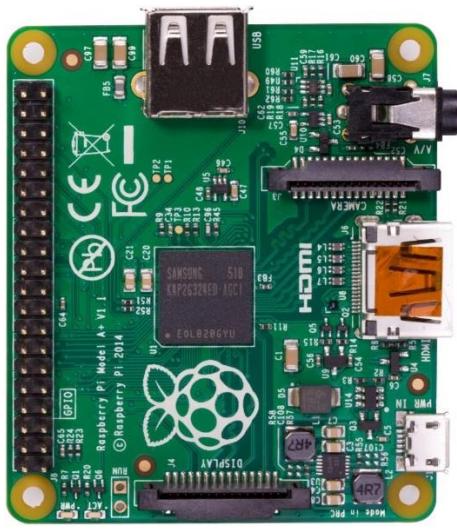
Actual image of Raspberry Pi 3 Model A+:



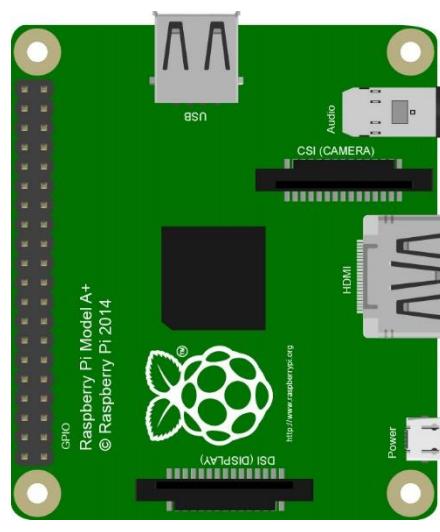
CAD image of Raspberry Pi 3 Model A+:



Actual image of Raspberry Pi 1 Model A+:



CAD image of Raspberry Pi 1 Model A+:

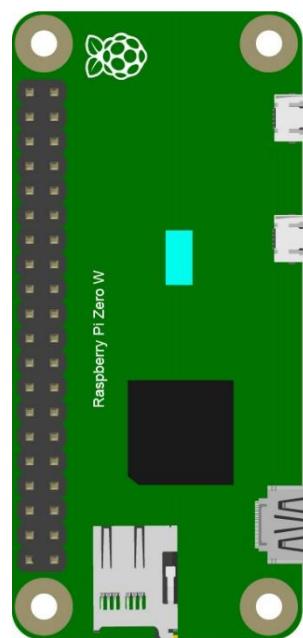




Actual image of Raspberry Pi Zero W:



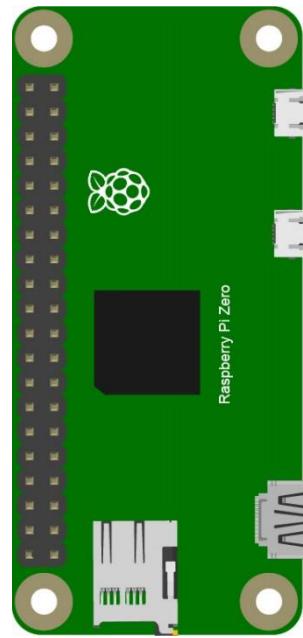
CAD image of Raspberry Pi Zero W:



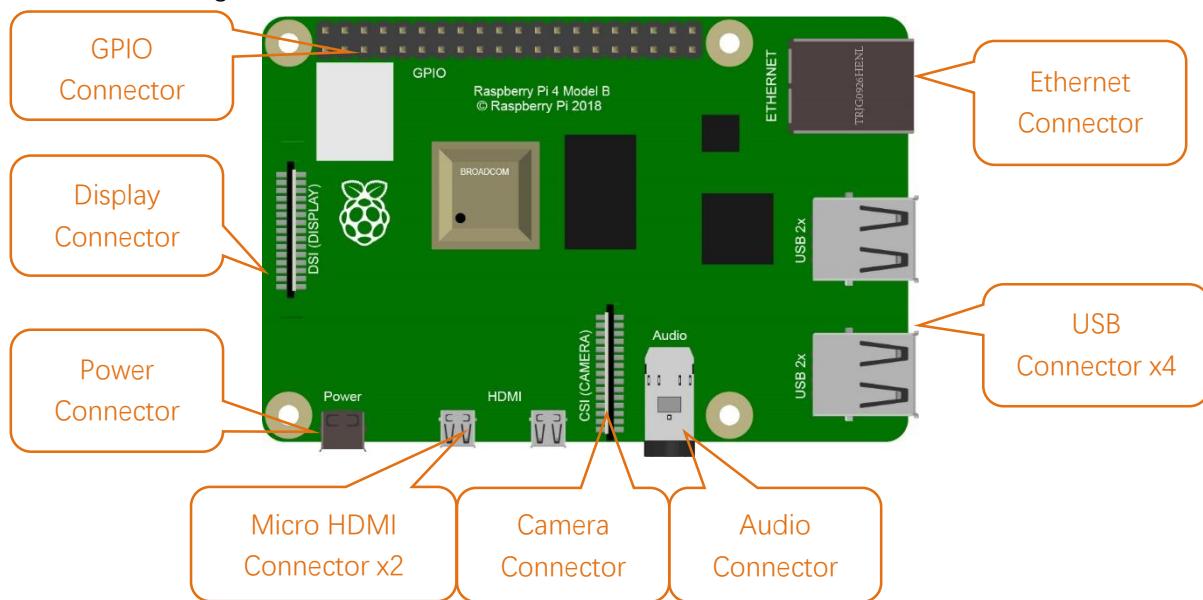
Actual image of Raspberry Pi Zero:



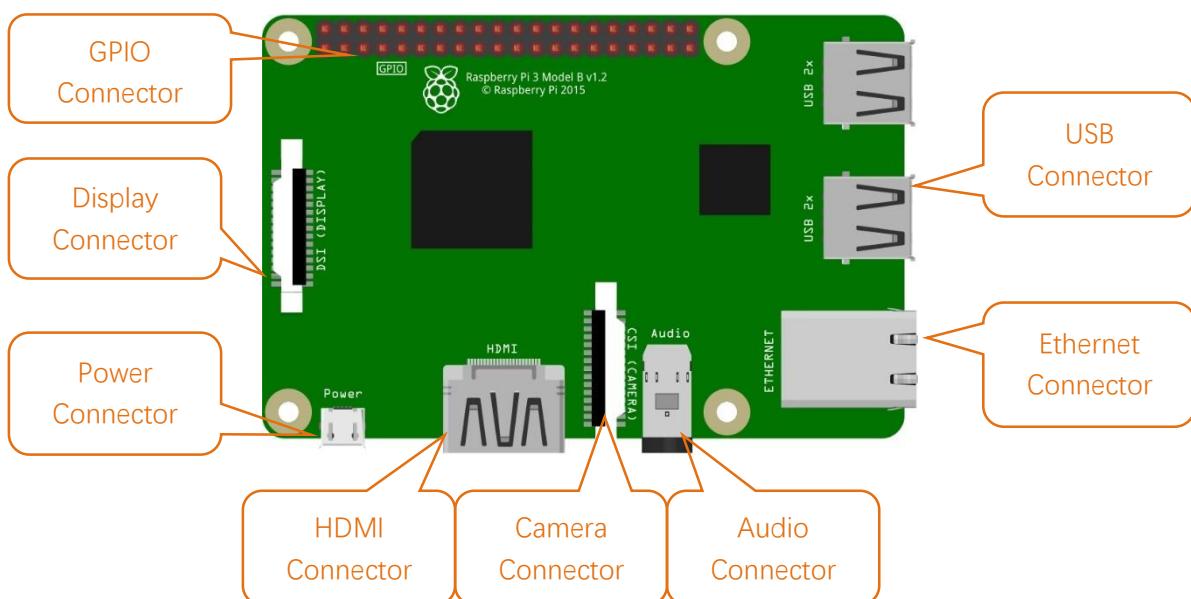
CAD image of Raspberry Pi Zero:



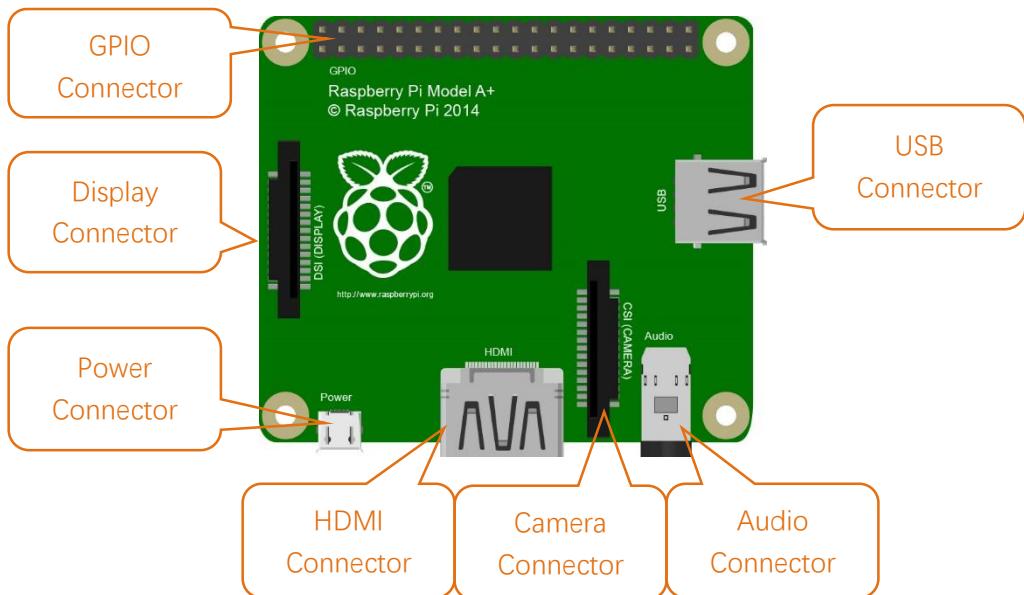
Hardware interface diagram of RPi 4B:



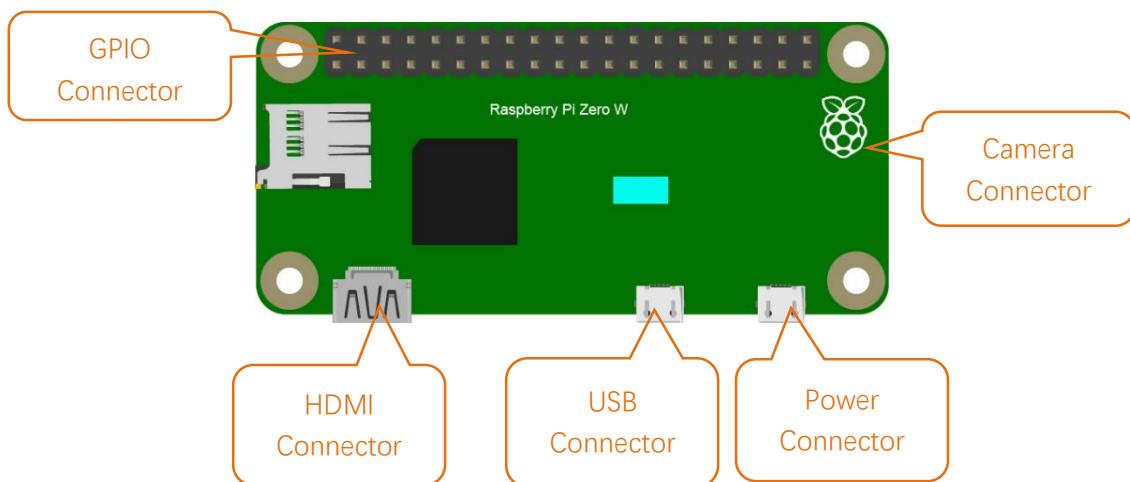
Hardware interface diagram of RPi 3B+/3B/2B/1B+:



Hardware interface diagram of RPi 3A+/A+:



Hardware interface diagram of RPi Zero/Zero W:



# Installing an Operating System

The first step is to install an operating system on your RPi so that it can be programmed and function. If you have installed a system in your RPi, you can start from Chapter 0 Preparation.

## Component List

### Required Components

Any Raspberry Pi with 40 GPIO	5V/3A Power Adapter. Note: Different versions of Raspberry Pi have different power requirements (please check the power requirements for yours on the chart in the following page.)
Micro or Type-C USB Cable x1	Micro SD Card (TF Card) x1, Card Reader x1



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 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.org/help/faqs/#powerReqs>

In addition, RPi also needs an Ethernet network cable used to connect it to a WAN (Wide Area Network).

All these components are necessary for any of your projects to work. Among them, the power supply of at least 5V/2.5A, because a lack of a sufficient power supply may lead to many functional issues and even damage your RPi, we STRONGLY RECOMMEND a 5V/2.5A power supply. We also recommend using a SD Micro Card with a capacity of 16GB or more (which, functions as the RPi's "hard drive") and is used to store the operating system and necessary operational files.

## Optional Components

Under normal circumstances, there are two ways to login to Raspberry Pi: 1) Using a stand-alone monitor. 2) Using a remote desktop or laptop computer monitor “sharing” the PC monitor with your RPi.

### Required Accessories for Monitor

If you choose to use an independent monitor, mouse and keyboard, you also need the following accessories:

1. A display with a HDMI interface
2. A Mouse and a Keyboard with an USB interface

As to Pi Zero and Pi Zero W, you also need the following accessories:

1. A Mini-HDMI to HDMI Adapter and Cable.
2. A Micro-USB to USB-A Adapter and Cable (Micro USB OTG Cable).
3. A USB HUB.
4. USB to Ethernet Interface or USB Wi-Fi receiver.

For different Raspberry Pi Modules, the optional items may vary slightly but they all aim to convert the interfaces to Raspberry Pi standards.

	Pi Zero	Pi A+	Pi Zero W	Pi 3A+	Pi B+/2B	Pi 3B/3B+	Pi 4B
<b>Monitor</b>	Yes (All)						
<b>Mouse</b>	Yes (All)						
<b>Keyboard</b>	Yes (All)						
<b>Micro-HDMI to HDMI Adapter &amp; Cable</b>	Yes	No	Yes	No	No	No	No
<b>Micro-HDMI to HDMI Adapter &amp; Cable</b>	No					Yes	
<b>Micro-USB to USB-A Adapter &amp; Cable (Micro USB OTG Cable)</b>	Yes	No	Yes	No			
<b>USB HUB</b>	Yes	Yes	Yes	Yes	No	No	
<b>USB to Ethernet Interface</b>	select one from two or select two from two		optional		Internal Integration	Internal Integration	
<b>USB Wi-Fi Receiver</b>			Internal Integration		optional		



## Required Accessories for Remote Desktop

If you do not have an independent monitor, or if you want to use a remote desktop, you first need to login to Raspberry Pi through SSH, and then open the VNC or RDP service. This requires the following accessories.

	Pi Zero	Pi Zero W	Pi A+	Pi 3A+	Pi B+/2B	Pi 3B/3B+/4B
<b>Micro-USB to USB-A Adapter &amp; Cable (Micro USB OTG Cable)</b>	Yes	Yes	No			NO
<b>USB to Ethernet interface</b>	Yes	Yes	Yes			

## Raspberry Pi OS

Without Screen - Use Raspberry Pi - under Windows PC: <https://youtu.be/YND0RUuP-to>

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.org/downloads/>

### Manually install an operating system image

Browse a range of operating systems provided by Raspberry Pi and by other organisations, and download them to install manually.

[See all download options](#)



## Operating system images

Many operating systems are available for Raspberry Pi, including Raspberry Pi OS, our official supported operating system, and operating systems from other organisations.

[Raspberry Pi Imager](#) is the quick and easy way to install an operating system to a microSD card ready to use with your Raspberry Pi. Alternatively, choose from the operating systems below, available to download and install manually.

Download:  
[Raspberry Pi OS \(32-bit\)](#)  
[Raspberry Pi Desktop](#)  
[Third-Party operating systems](#)

### Raspberry Pi OS

Compatible with:

[All Raspberry Pi models](#)



### Raspberry Pi OS with desktop and recommended software

Release date: January 11th 2021  
 Kernel version: 5.4  
 Size: 2.863MB  
[Show SHA256 file integrity hash](#)  
[Release notes](#)

[Download](#)

[Download torrent](#)



And then the zip file is downloaded.



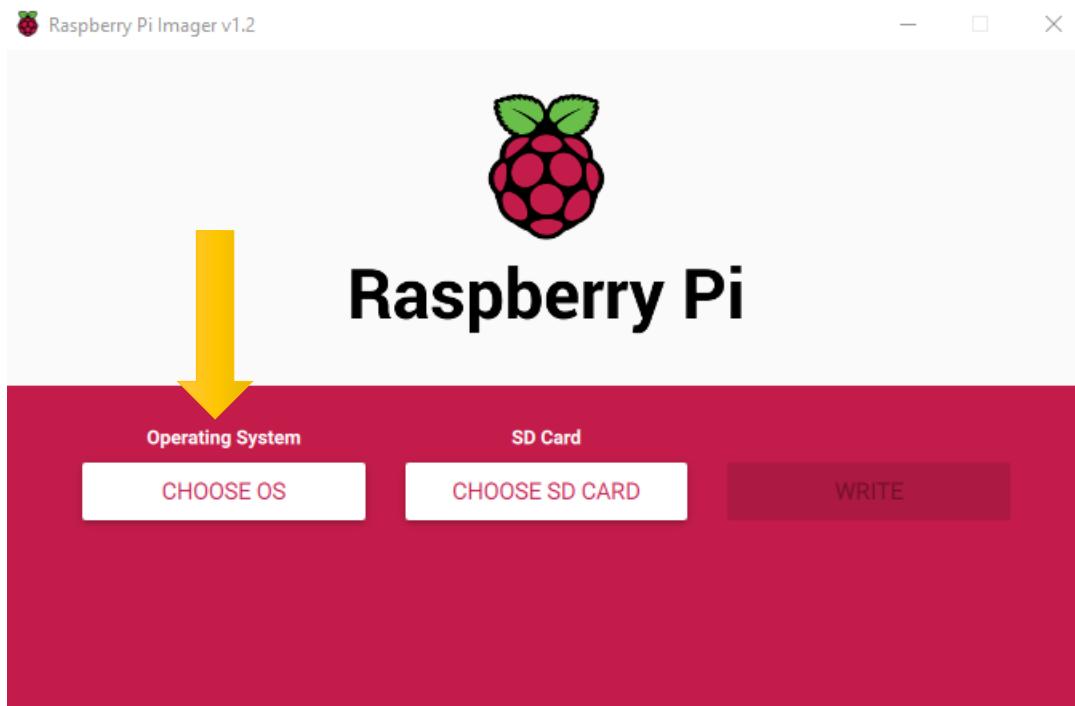


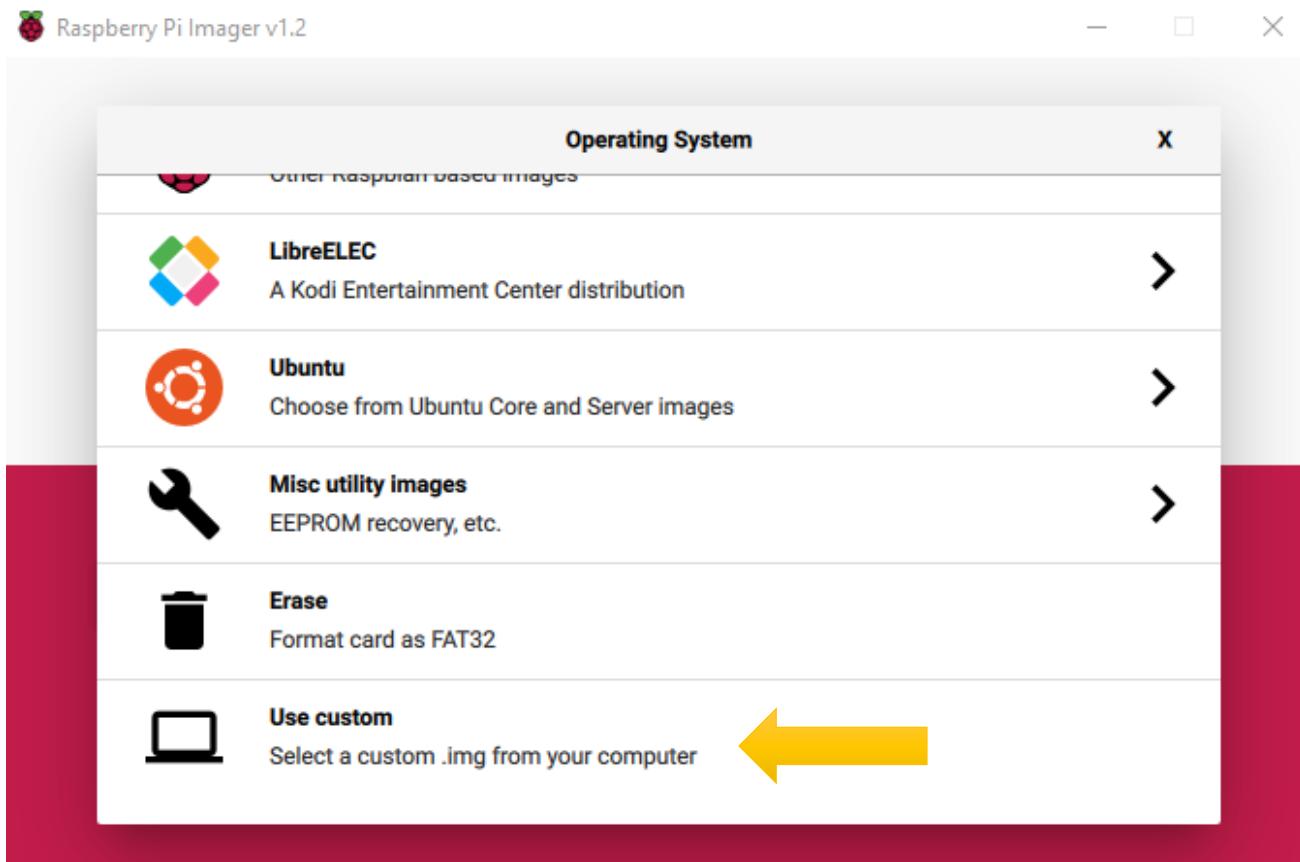
### Write System to Micro SD Card

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

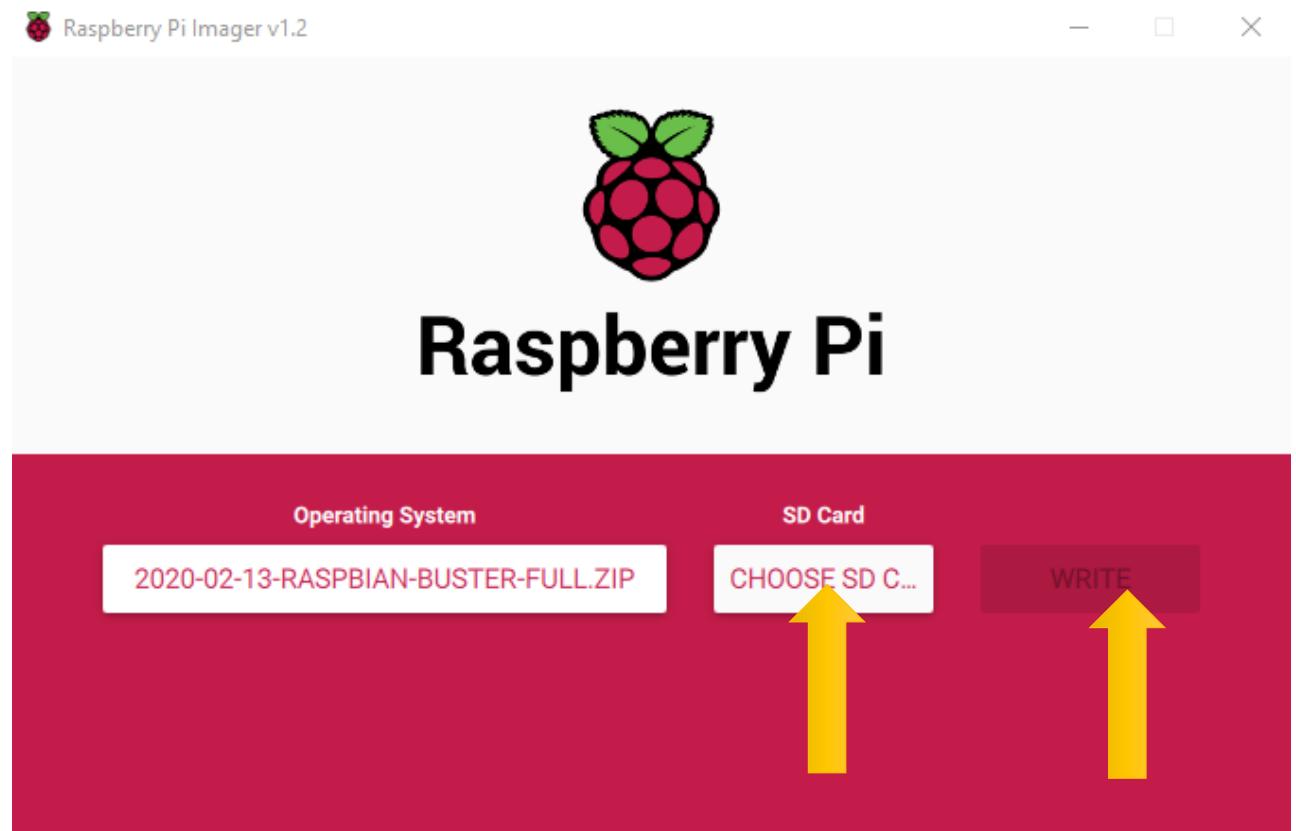


Then open imager toll. Choose system that you just downloaded in Use custom.





Choose the SD card. Then click "WRITE".

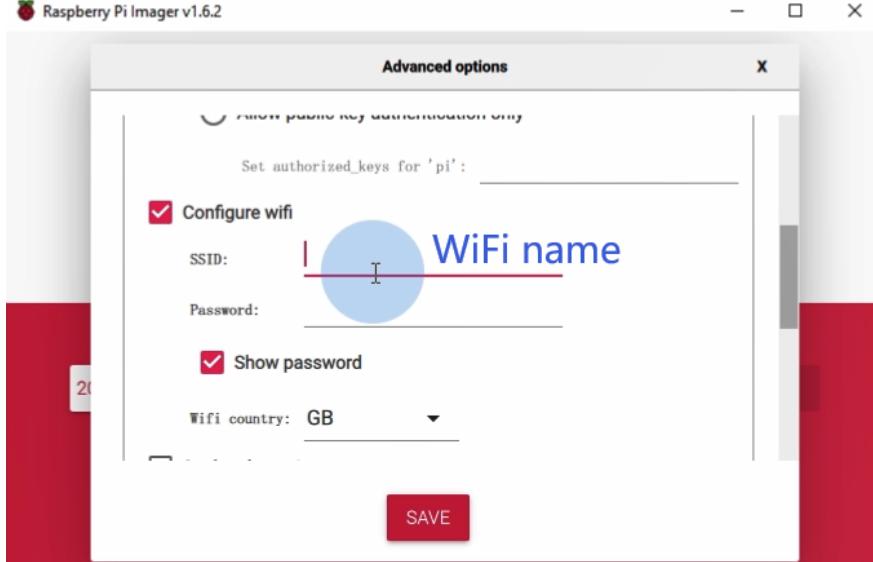
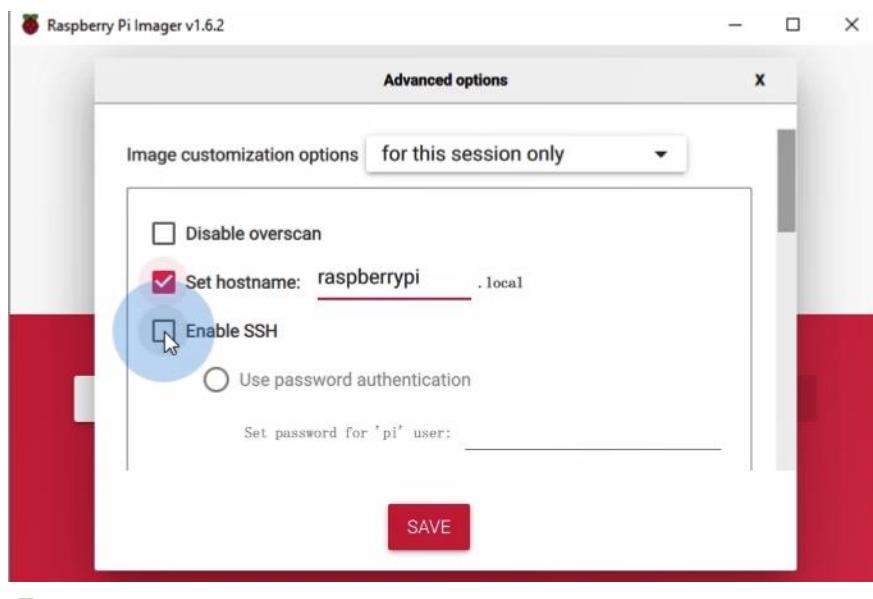


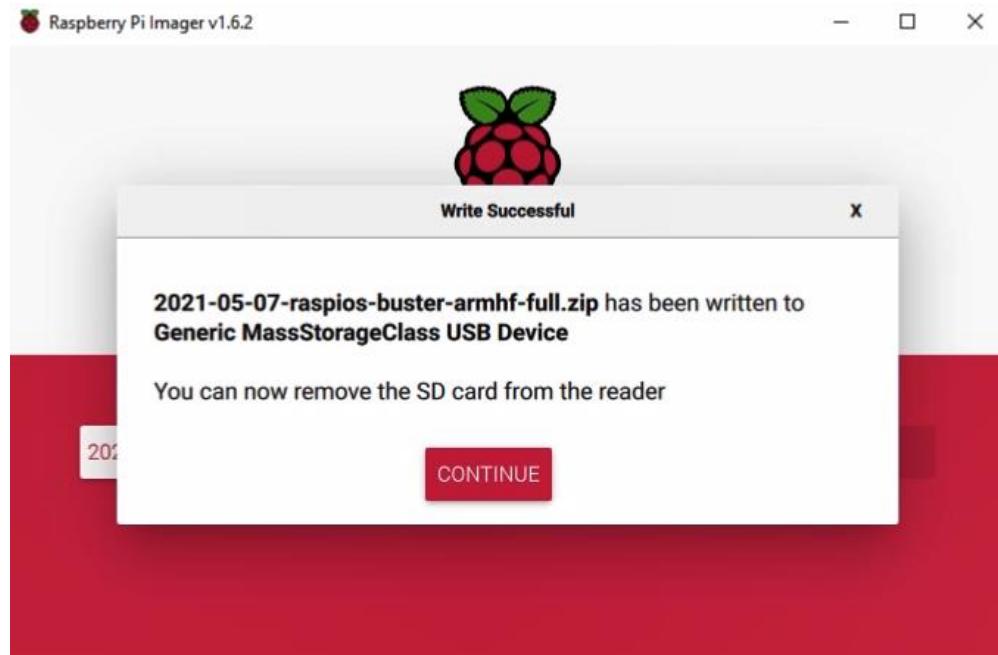


## Enable ssh and configure WiFi



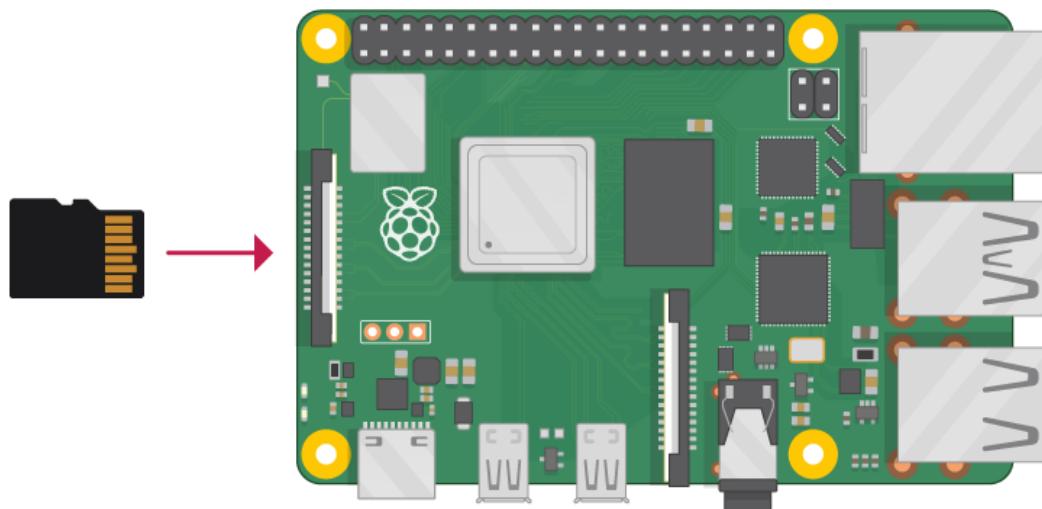
Press **Ctrl+Shift+x** to configure RPi.





## Insert SD card

Then remove SD card from card reader and insert it into Raspberry Pi.



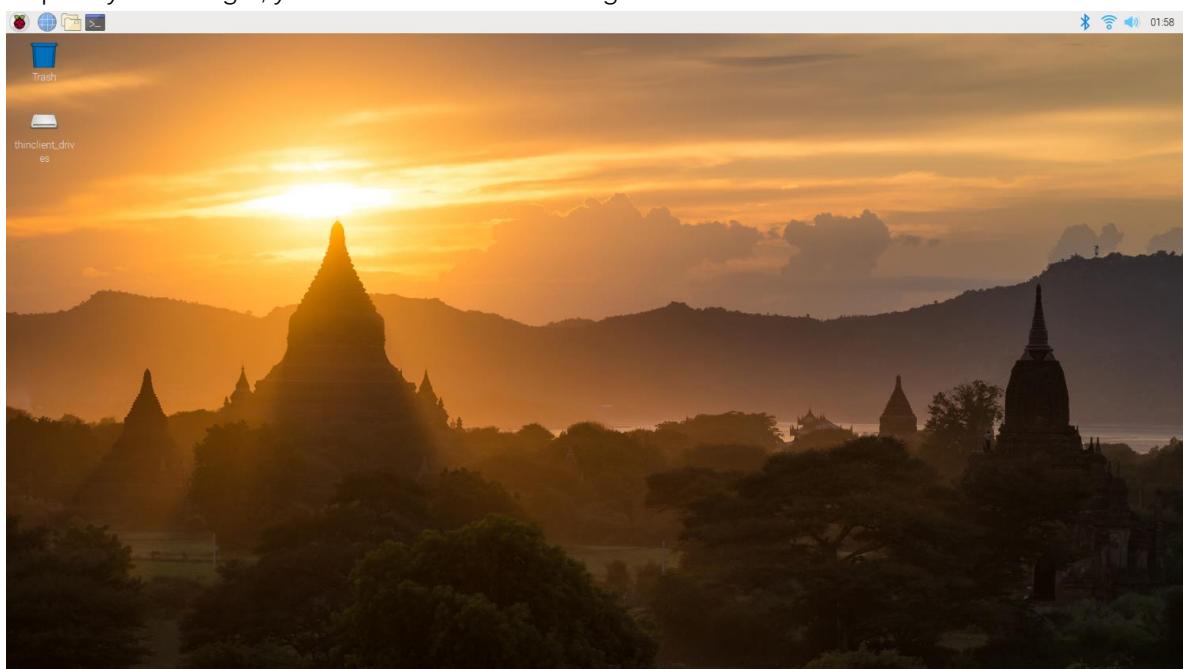


## Getting Started with Raspberry Pi

### 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 4B, 3B+/3B 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.



## 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:

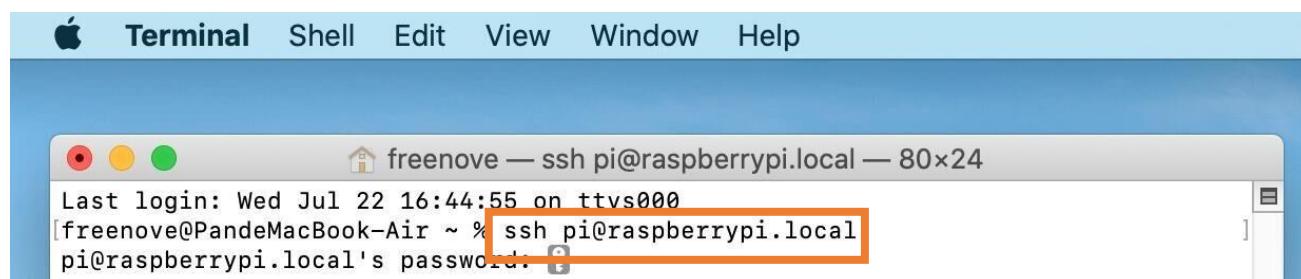
[MAC OS remote desktop](#) and [Windows OS remote desktop](#).

### MAC OS Remote Desktop

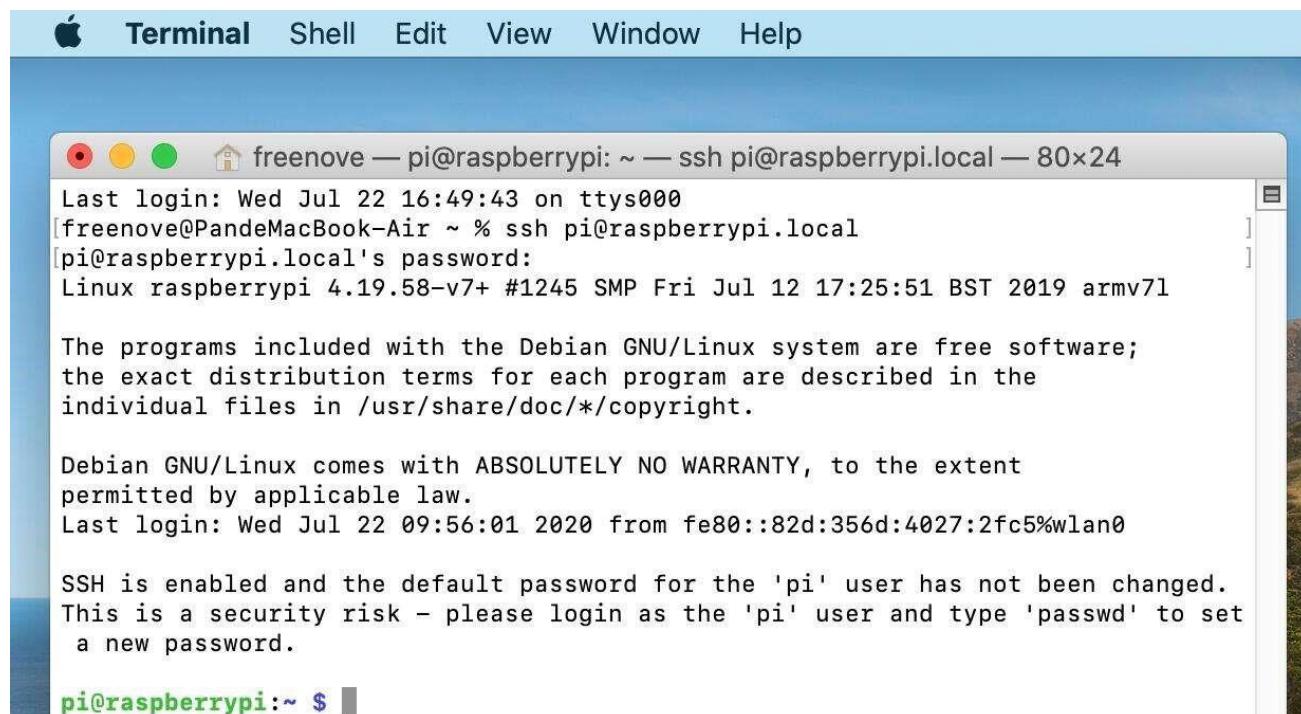
Open the terminal and type following command. **If this command doesn't work, please move to next page.**

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

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



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



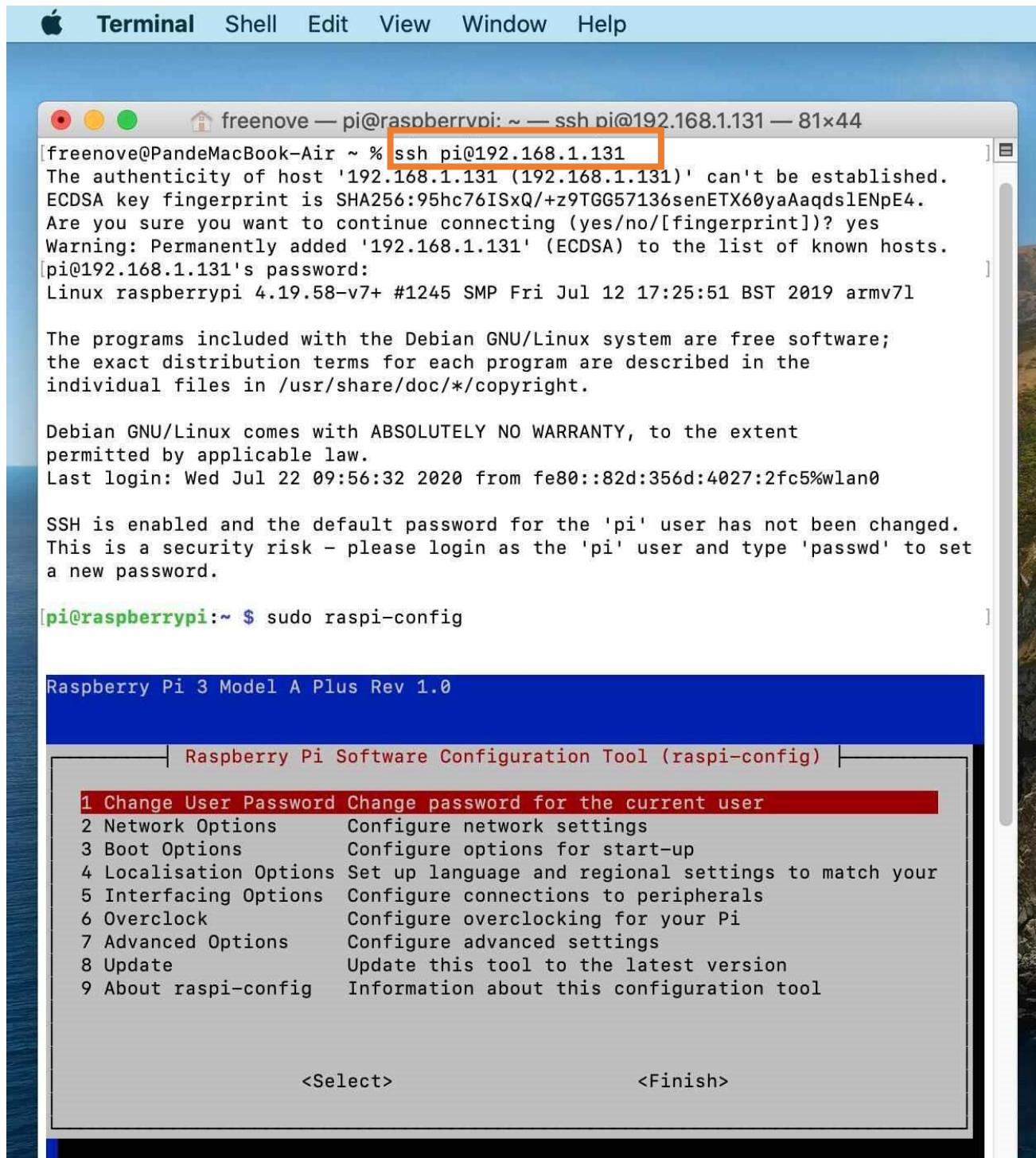
You can also use the IP address to log in Pi.

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 RaspberryPi without desktop.

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

```
ping raspberrypi.local
```



```
Microsoft Windows [Version 10.0.18363.1556]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Administrator>ping raspberrypi.local

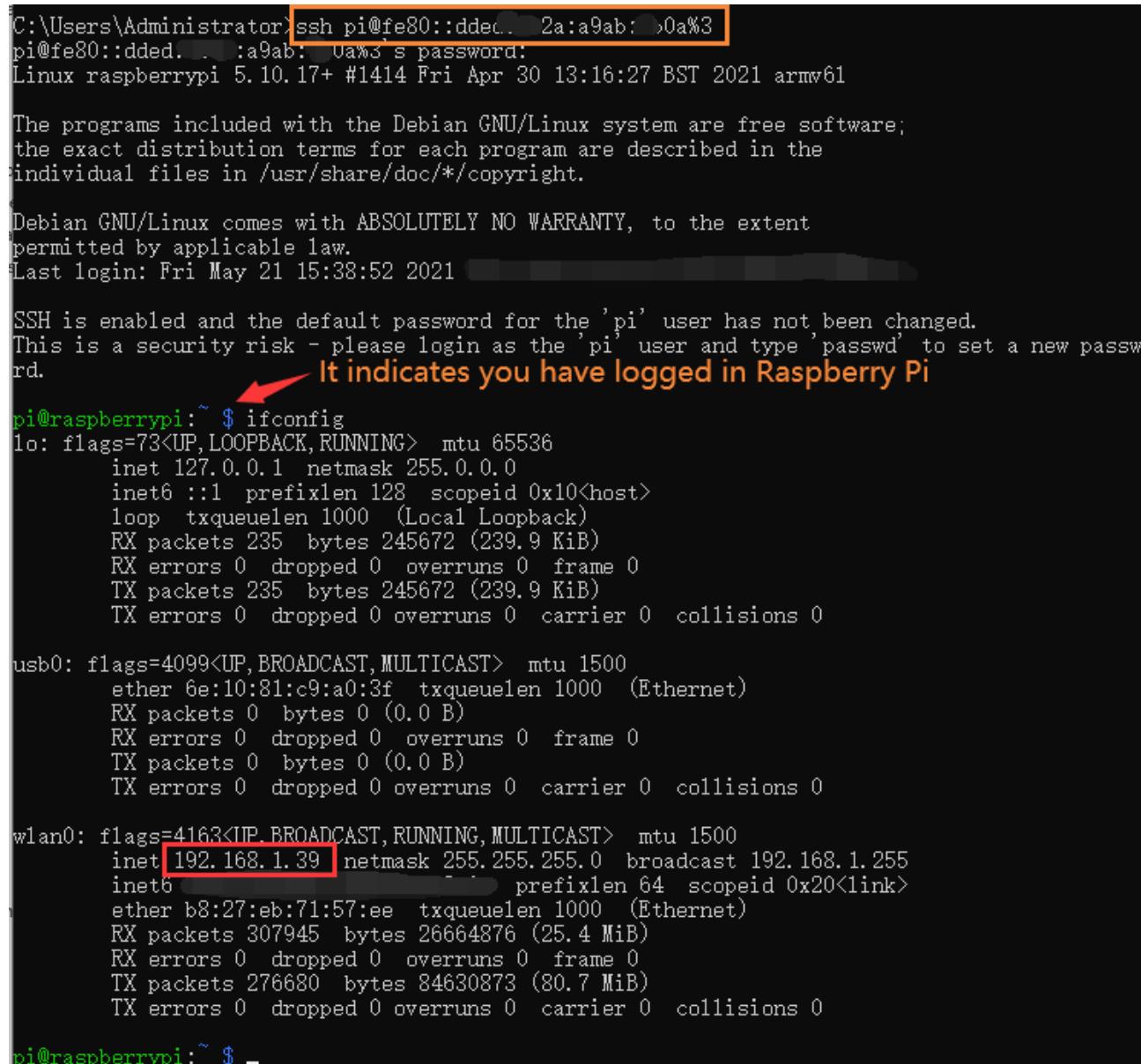
Pinging raspberrypi.local [fe80::2a9ab:10a%3] with 32 bytes of data:
```

The one before raspberrypi.local is the IPV6 address of RaspberryPi

Use following command to login Raspberry Pi.

```
ssh pi@xxxxxxxxxxxxx(IPV6 address)
```

Enter yes not y if needed.



```
C:\Users\Administrator>ssh pi@fe80::2a9ab:10a%3
pi@fe80::2a9ab:10a%3's password:
Linux raspberrypi 5.10.17+ #1414 Fri Apr 30 13:16:27 BST 2021 armv6l

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: Fri May 21 15:38:52 2021

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: ~ $ ifconfig
lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
        inet6 ::1 prefixlen 128 scopeid 0x10<host>
            loop txqueuelen 1000 (Local Loopback)
            RX packets 235 bytes 245672 (239.9 KiB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 235 bytes 245672 (239.9 KiB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

usb0: flags=4099<UP,BROADCAST,MULTICAST> mtu 1500
    ether 6e:10:81:c9:a0:3f txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 192.168.1.39 netmask 255.255.255.0 broadcast 192.168.1.255
        inet6 fe80::b827:ebff:fe71:57ee prefixlen 64 scopeid 0x20<link>
            ether b8:27:eb:71:57:ee txqueuelen 1000 (Ethernet)
            RX packets 307945 bytes 26664876 (25.4 MiB)
            RX errors 0 dropped 0 overruns 0 frame 0
            TX packets 276680 bytes 84630873 (80.7 MiB)
            TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

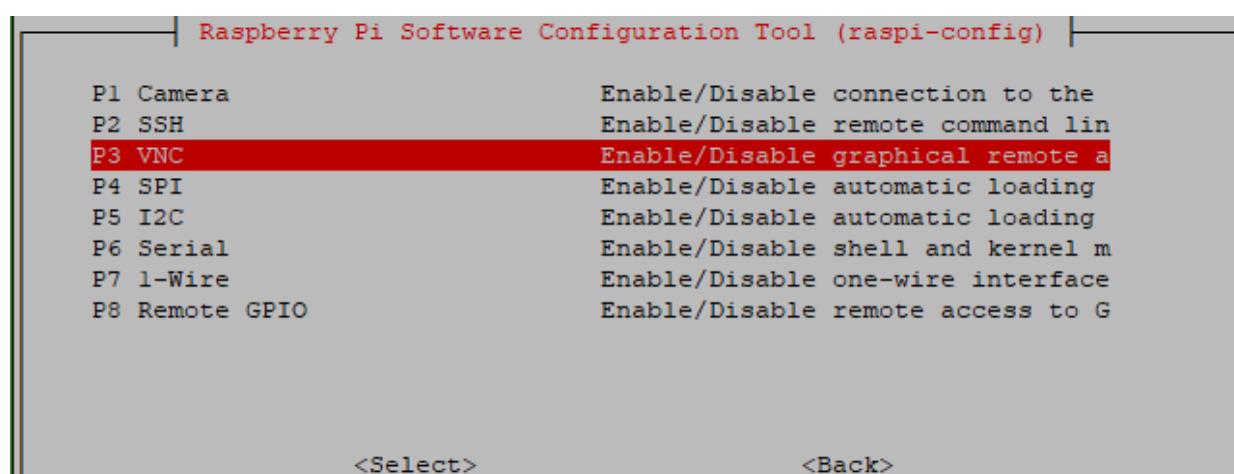
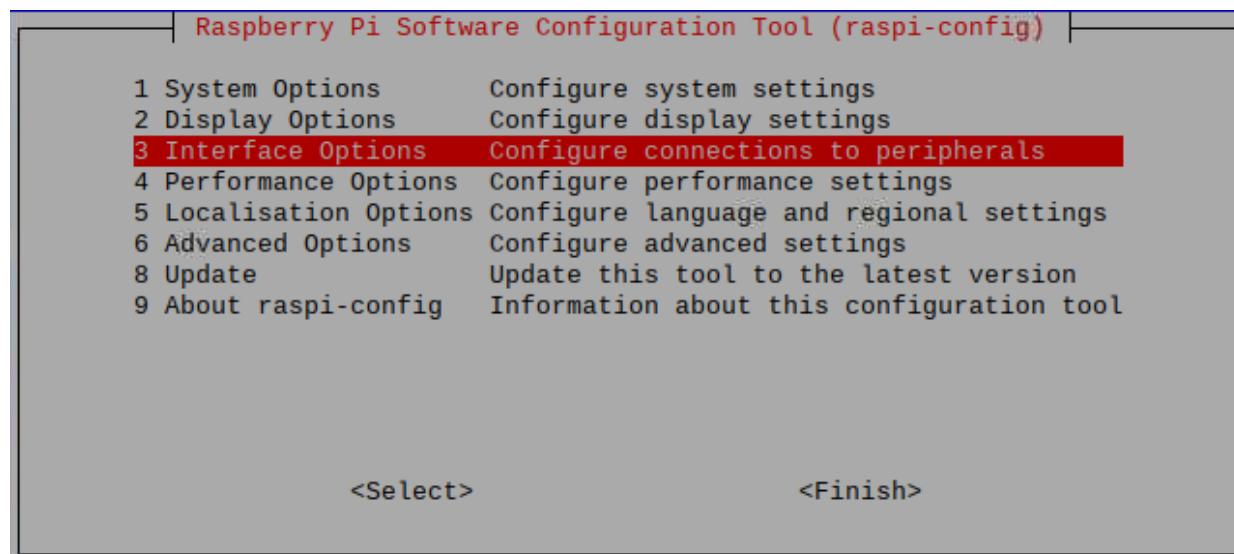
pi@raspberrypi: ~ $
```

## VNC Viewer & VNC

### Enable VNC

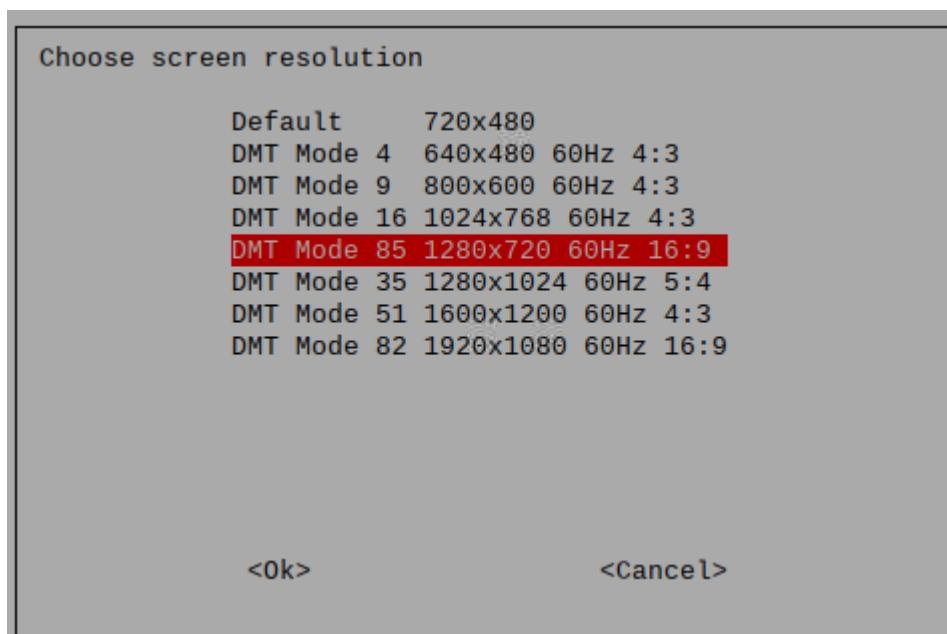
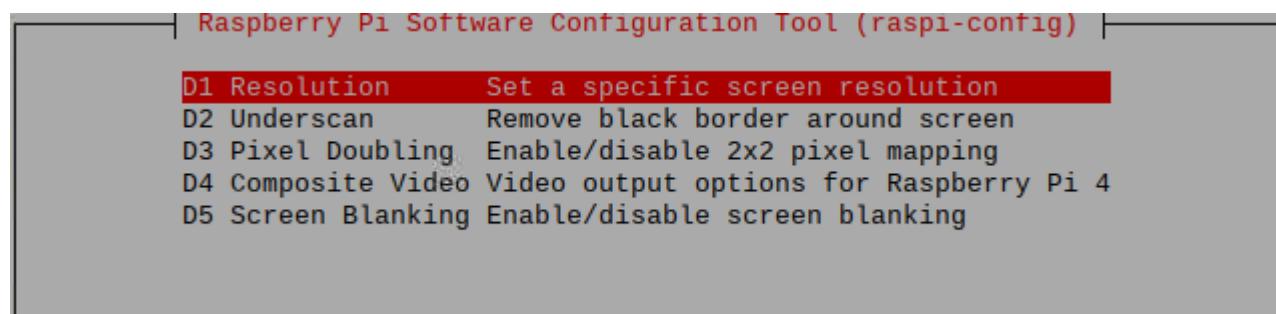
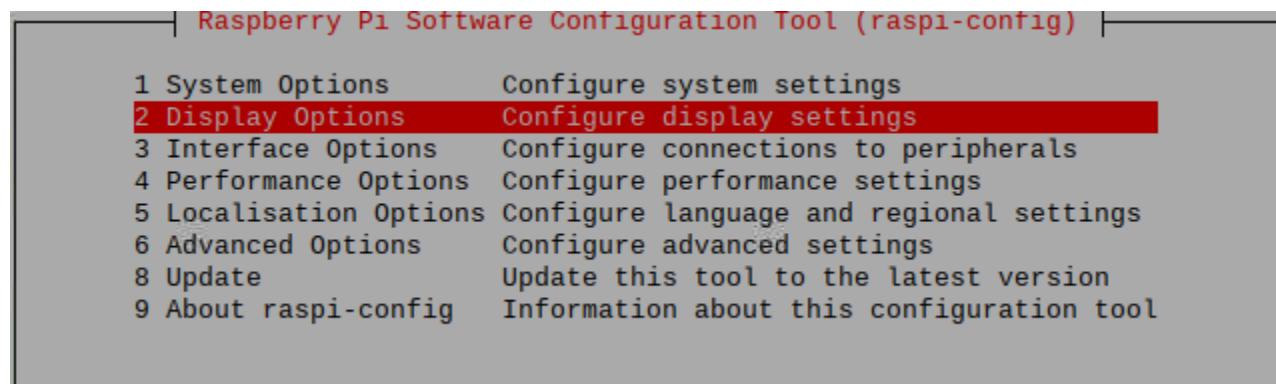
Type the following command. And select Interface Options→P3 VNC → Enter→Yes→OK. Here Raspberry Pi may need be restarted, and choose ok. Then open VNC interface.

```
sudo raspi-config
```



## Set Resolution

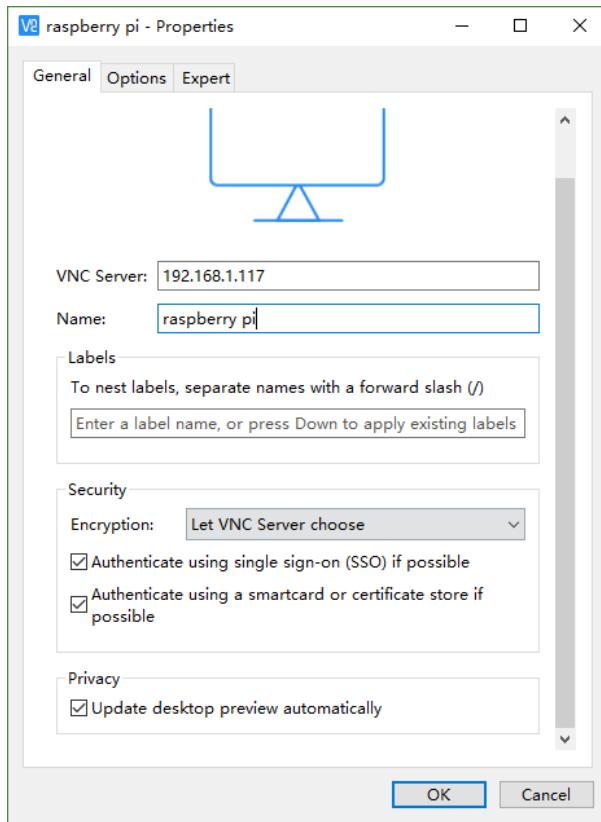
You can also set other resolutions. If you don't know what to set, you can set it as 1280x720 first.



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.

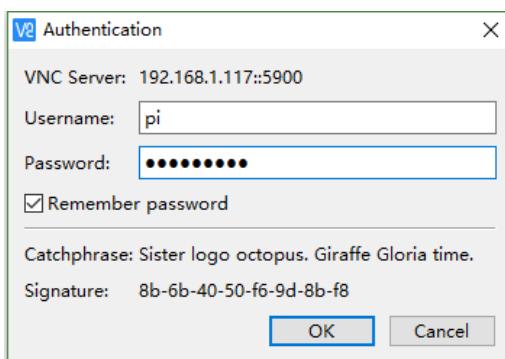


Enter ip address of your Raspberry Pi and fill in a name. Then click OK.

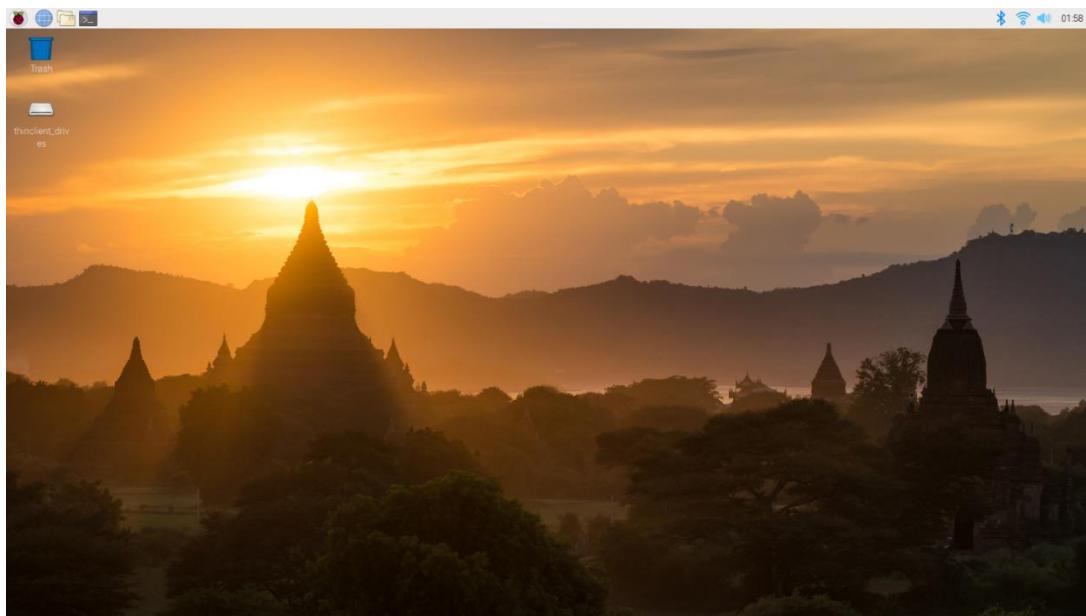
Then on the VNC Viewer panel, double-click new connection you just created,



and the following dialog box pops up.

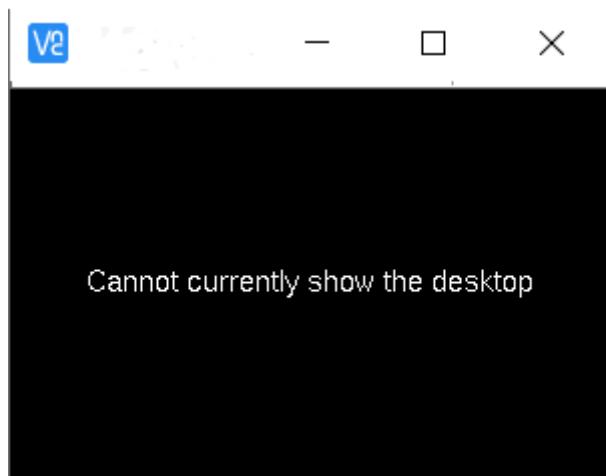


Enter username: **pi** and Password: **raspberry**. And click OK.

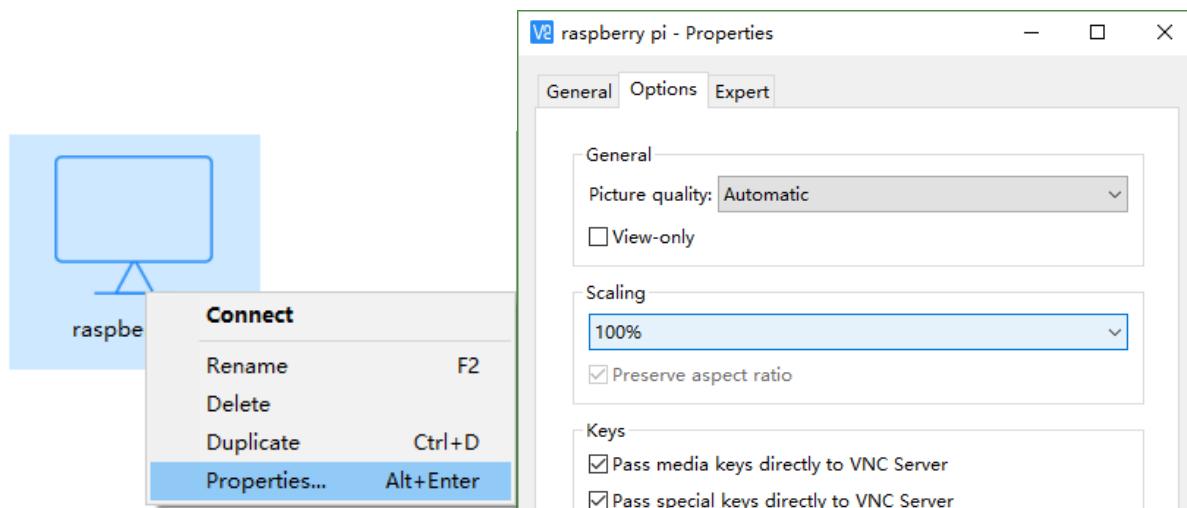


Here, you have logged in to Raspberry Pi successfully by using VNC Viewer

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 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.



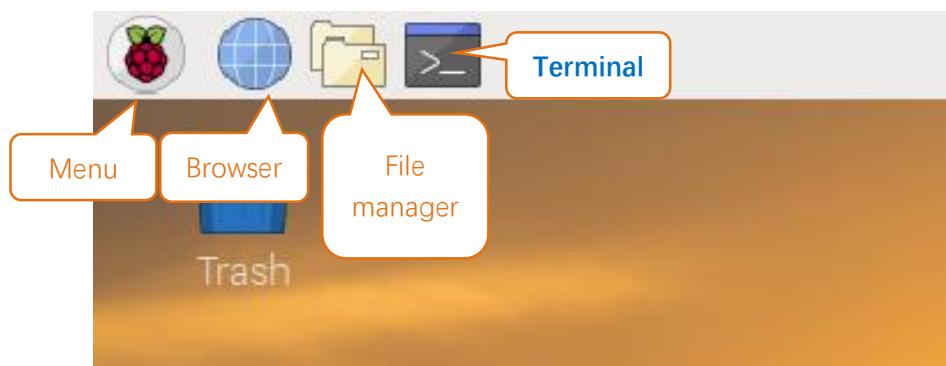
# Chapter 0 Preparation

Why "Chapter 0"? Because in program code the first number is 0. We choose to follow this rule. In this chapter, we will do some necessary foundational preparation work: Start your Raspberry Pi and install some necessary libraries.

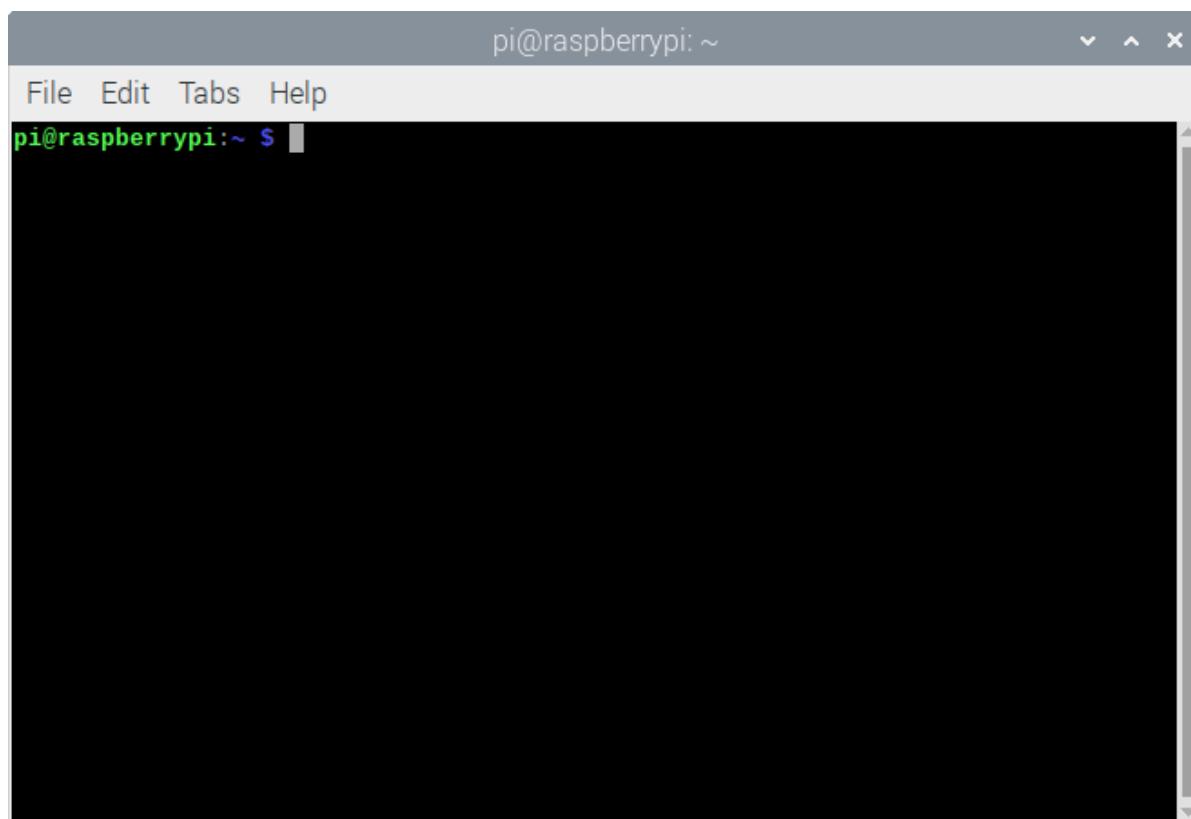
## Linux Command

Raspberry Pi OS is based on the Linux Operation System. Now we will introduce you to some frequently used Linux commands and rules.

First, open the Terminal. All commands are executed in Terminal.



When you click the Terminal icon, following interface appears.



**Note: The Linux is case sensitive.**

First, type “ls” into the Terminal and press the “Enter” key. The result is shown below:



```
pi@raspberrypi:~ $ ls
Desktop
Documents
Downloads
Freenove_Three-wheeled_Smart_Car_Kit_for_Raspberry_Pi
Freenove_Ultimate_Starter_Kit_for_Raspberry_Pi
MagPi
mu_code
```

The “ls” command lists information about the files (the current directory by default).

Content between “\$” and “pi@raspberrypi:” is the current working path. “~” represents the user directory, which refers to “/home/pi” here.

```
pi@raspberrypi:~ $ pwd
/home/pi
```

“cd” is used to change directory. “/” represents the root directory.

```
pi@raspberrypi:~ $ cd /usr
pi@raspberrypi:/usr $ ls
bin  games  include  lib  local  man  sbin  share  src
pi@raspberrypi:/usr $ cd ~
pi@raspberrypi:~ $
```

Later in this Tutorial, we will often change the working path. Typing commands under the wrong directory may cause errors and break the execution of further commands.

Many frequently used commands and instructions can be found in the following reference table.

Command	instruction
<b>ls</b>	Lists information about the FILEs (the current directory by default) and entries alphabetically.
<b>cd</b>	Changes directory
<b>sudo + cmd</b>	Executes cmd under root authority
<b>./</b>	Under current directory
<b>gcc</b>	GNU Compiler Collection
<b>git clone URL</b>	Use git tool to clone the contents of specified repository, and URL in the repository address.

There are many commands, which will come later. For more details about commands. You can refer to:

<http://www.linux-commands-examples.com>

## Shortcut Key

Now, we will introduce several commonly used shortcuts that are very useful in Terminal.

1. **Up and Down Arrow Keys:** Pressing “↑” (the Up key) will go backwards through the command history and pressing “↓” (the Down Key) will go forwards through the command history.

2. **Tab Key:** The Tab key can automatically complete the command/path you want to type. When there is only one eligible option, the command/path will be completely typed as soon as you press the Tab key even you only type one character of the command/path.

As shown below, under the '~' directory, you enter the Documents directory with the "cd" command. After typing "cd D", pressing the Tab key (there is no response), pressing the Tab key again then all the files/folders that begin with "D" will be listed. Continue to type the letters "oc" and then pressing the Tab key, the "Documents" is typed automatically.

```
pi@raspberrypi:~ $ cd D
Desktop/  Documents/ Downloads/
pi@raspberrypi:~ $ cd Doc█
```

```
pi@raspberrypi:~ $ cd D
Desktop/  Documents/ Downloads/
pi@raspberrypi:~ $ cd Documents/
```

## Install WiringPi

WiringPi is a GPIO access library written in C language for the used in the Raspberry Pi.

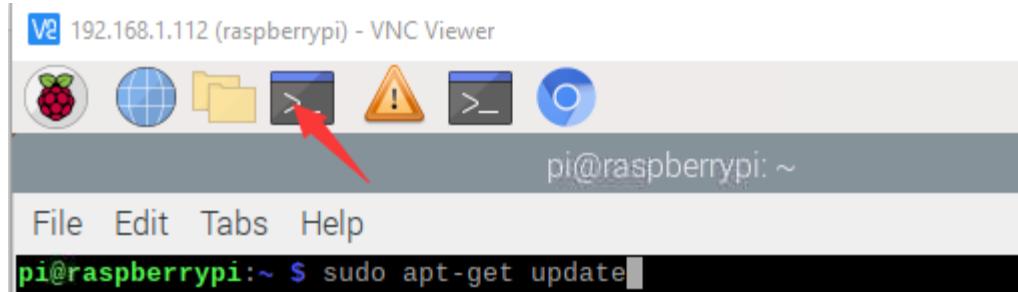
### WiringPi Installation Steps

To install the WiringPi library, please open the Terminal and then follow the steps and commands below.

Note: For a command containing many lines, execute them one line at a time.

Enter the following commands **one by one** in the **terminal** to install WiringPi:

```
sudo apt-get update
git clone https://github.com/WiringPi/WiringPi
cd WiringPi
./build
```



```
pi@raspberrypi:~ $ sudo apt-get update
Cloning into 'WiringPi'...
remote: Enumerating objects: 41, done.
remote: Counting objects: 100% (41/41), done.
remote: Compressing objects: 100% (38/38), done.
remote: Total 1483 (delta 15), reused 13 (delta 1), pack-reused 1442
Receiving objects: 100% (1483/1483), 793.91 KiB | 924.00 KiB/s, done.
Resolving deltas: 100% (918/918), done.
```

```
pi@raspberrypi:~ $ git clone https://github.com/WiringPi/WiringPi
pi@raspberrypi:~/WiringPi $ ./build
wiringPi Build script
=====
```

All Done.

NOTE: To compile programs with wiringPi, you need to add:  
 -lwiringPi  
 to your compile line(s) To use the Gertboard, MaxDetect, etc.  
 code (the devLib), you need to also add:  
 -lwiringPiDev  
 to your compile line(s).

Run the gpio command to check the installation:

```
gpio -v
```

That should give you some confidence that the installation was a success.

```
gpio version: 2.60
Copyright (c) 2012-2018 Gordon Henderson
This is free software with ABSOLUTELY NO WARRANTY.
For details type: gpio -warranty

Raspberry Pi Details:
```

## Obtain the Project Code

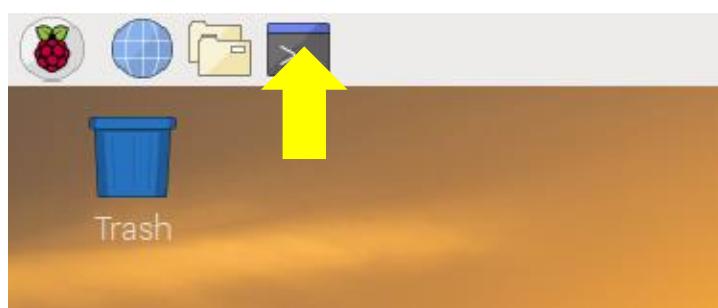
After the above installation is completed, you can visit our official website (<http://www.freenove.com>) or our GitHub resources at (<https://github.com/freenove>) to download the latest available project code. We provide both **C** language and **Python** language code for each project to allow ease of use for those who are skilled in either language.

This is the method for obtaining the code:

In the pi directory of the RPi terminal, enter the following command.

```
cd  
git clone --depth 1 https://github.com/freenove/Freenove_LCD_Module
```

(There is no need for a password. If you get some errors, please check your commands.)

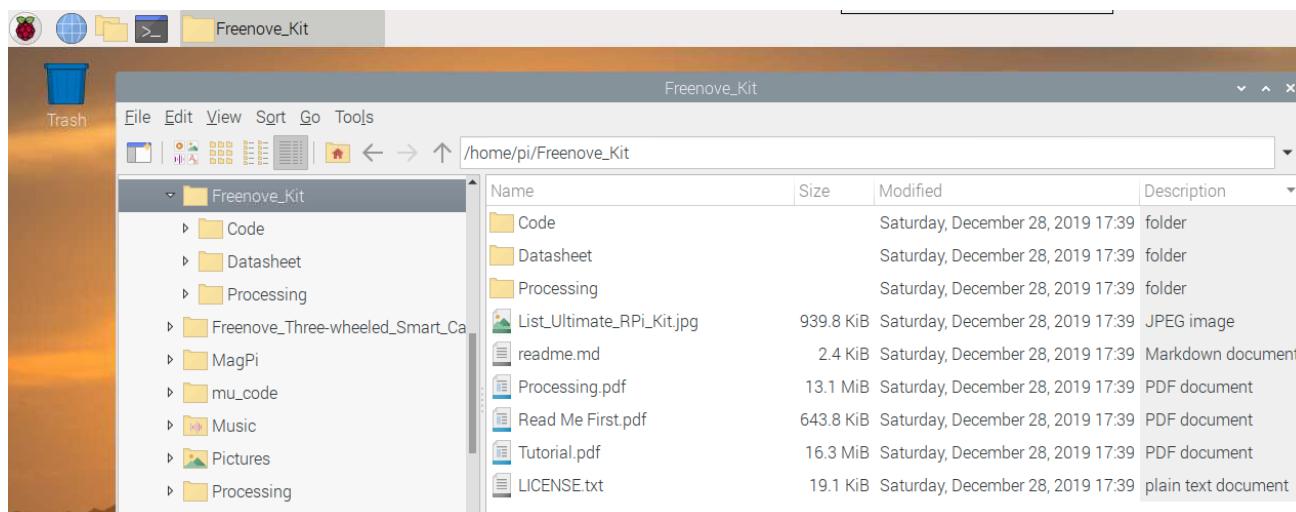


After the download is completed, a new folder " Freenove\_LCD\_Module" is generated, which contains all of the tutorials and required code.

This folder name seems a little too long. We can simply rename it by using the following command.

```
mv Freenove_LCD_Module/ Freenove_Kit/
```

"Freenove\_Kit" is now the new and much shorter folder name.



If you have no experience with Python, we suggest that you refer to this website for basic information and knowledge.

<https://python.swaroopch.com/basics.html>

## Python2 & Python3

If you only use C/C++, you can skip this section.

Python code, used in our kits, can now run on Python2 and Python3. **Python3 is recommend**. If you want to use Python2, please make sure your Python version is 2.7 or above. Python2 and Python3 are not fully compatible. However, Python2.6 and Python2.7 are transitional versions to python3, therefore you can also use Python2.6 and 2.7 to execute some Python3 code.

You can type “python2” or “python3” respectively into Terminal to check if python has been installed. Press Ctrl-Z to exit.

```
pi@raspberrypi:~ $ python2
Python 2.7.13 (default, Nov 24 2017, 17:33:09)
[GCC 6.3.0 20170516] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>>
[2]+ Stopped                  python2
pi@raspberrypi:~ $ python3
Python 3.5.3 (default, Jan 19 2017, 14:11:04)
[GCC 6.3.0 20170124] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> █
```

Type “python”, and Terminal shows that it links to python2.

```
pi@raspberrypi:~ $ python
Python 2.7.13 (default, Nov 24 2017, 17:33:09)
[GCC 6.3.0 20170516] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> █
```

If you want to use Python3 in Raspberry Pi, it is recommended to set python3 as default Python by following the steps below.

1. Enter directory /usr/bin

```
cd /usr/bin
```

2. Delete the old python link.

```
sudo rm python
```

3. Create new python links to python3.

```
sudo ln -s python3 python
```

4. Execute python to check whether the link succeeds.

```
python
```

```
pi@raspberrypi:/usr/bin $ sudo rm python
pi@raspberrypi:/usr/bin $ sudo ln -s python3 python
pi@raspberrypi:/usr/bin $ python
Python 3.5.3 (default, Jan 19 2017, 14:11:04)
[GCC 6.3.0 20170124] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> █
```

If you want to use Python2, repeat the steps above and just change the third command to the following:

```
sudo ln -s python2 python
```

```
pi@raspberrypi:/usr/bin $ sudo rm python
pi@raspberrypi:/usr/bin $ sudo ln -s python2 python
pi@raspberrypi:/usr/bin $ python
Python 2.7.13 (default, Nov 24 2017, 17:33:09)
[GCC 6.3.0 20170516] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```

We will only use the term “Python” without reference to Python2 or Python3. You can choose to use either. Finally, all the necessary preparations have been completed! Next, we will combine the RPi and electronic components to build a series of projects from easy to the more challenging and difficult as we focus on learning the associated knowledge of each electronic circuit.

# Chapter 1 LCD1602

In this chapter, we will learn about the LCD1602 Display Screen,

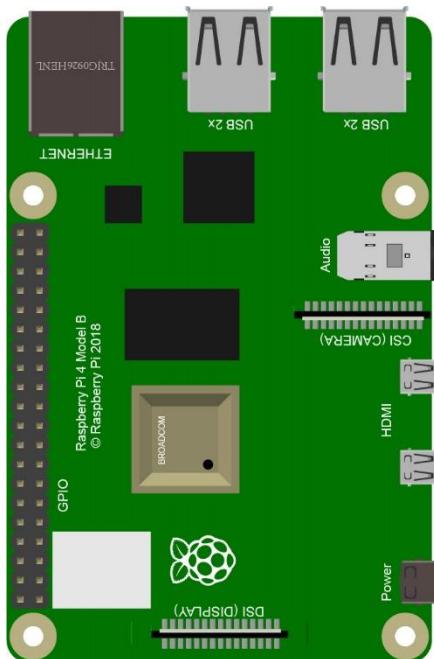
## Project 1.1 I2C LCD1602

In this section we learn how to use lcd1602 to display something.

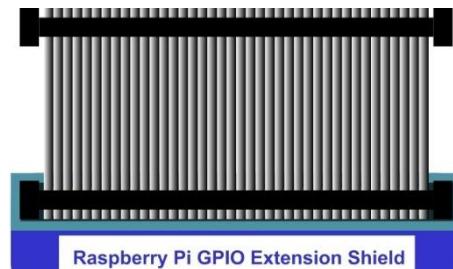
### Component List

#### Raspberry Pi

(Recommended: Raspberry Pi 4B / 3B+ / 3B  
Compatible: 3A+ / 2B / 1B+ / 1A+ / Zero W / Zero)



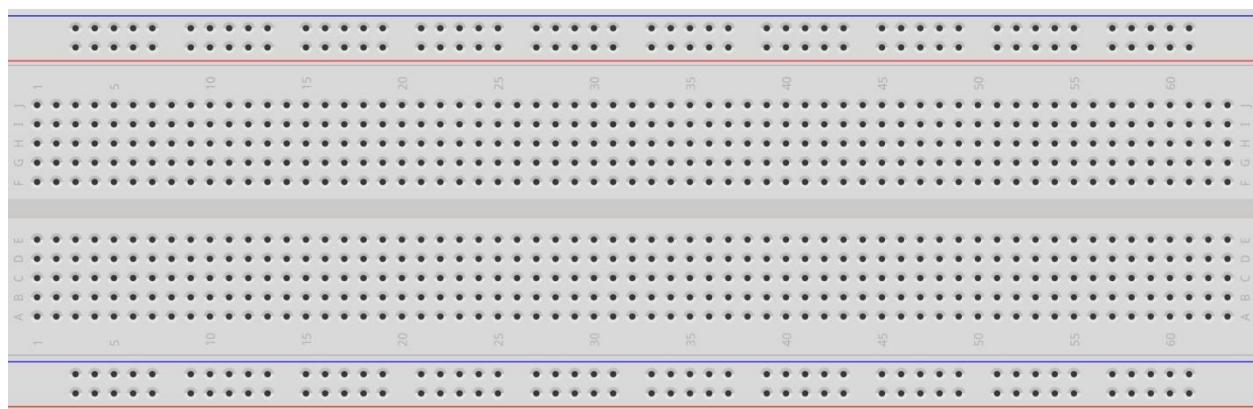
#### GPIO Extension Board & Ribbon Cable

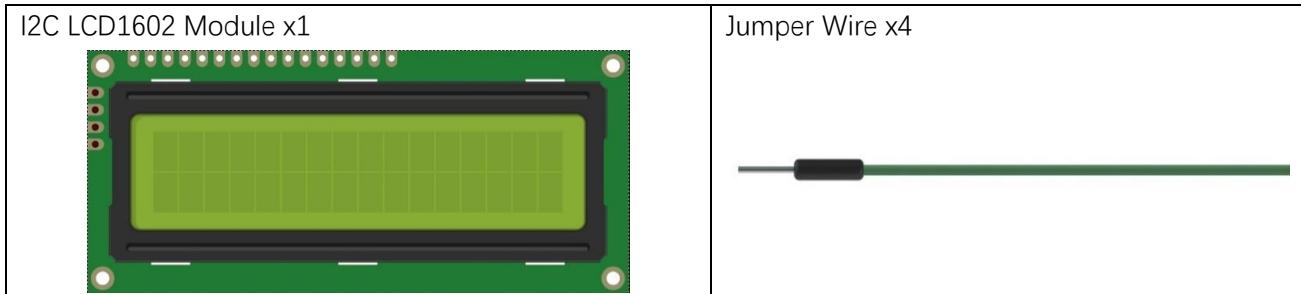


Raspberry Pi GPIO Extension Shield

3V3	5V
SDA1	5V
SCL1	GND
GPIO4	TXDO
GND	RXD0
GPIO17	GPIO18
GPIO27	GND
GPIO22	GPIO23
3V3	GPIO24
MOSI	GND
MISO	GPIO25
SCK	CE0
GND	CE1
SDA0	SCL0
GPIO5	GND
GPIO6	GPIO12
GPIO13	GND
GPIO19	GPIO16
GPIO26	GPIO20
GND	GPIO21

#### Breadboard x1





## GPIO

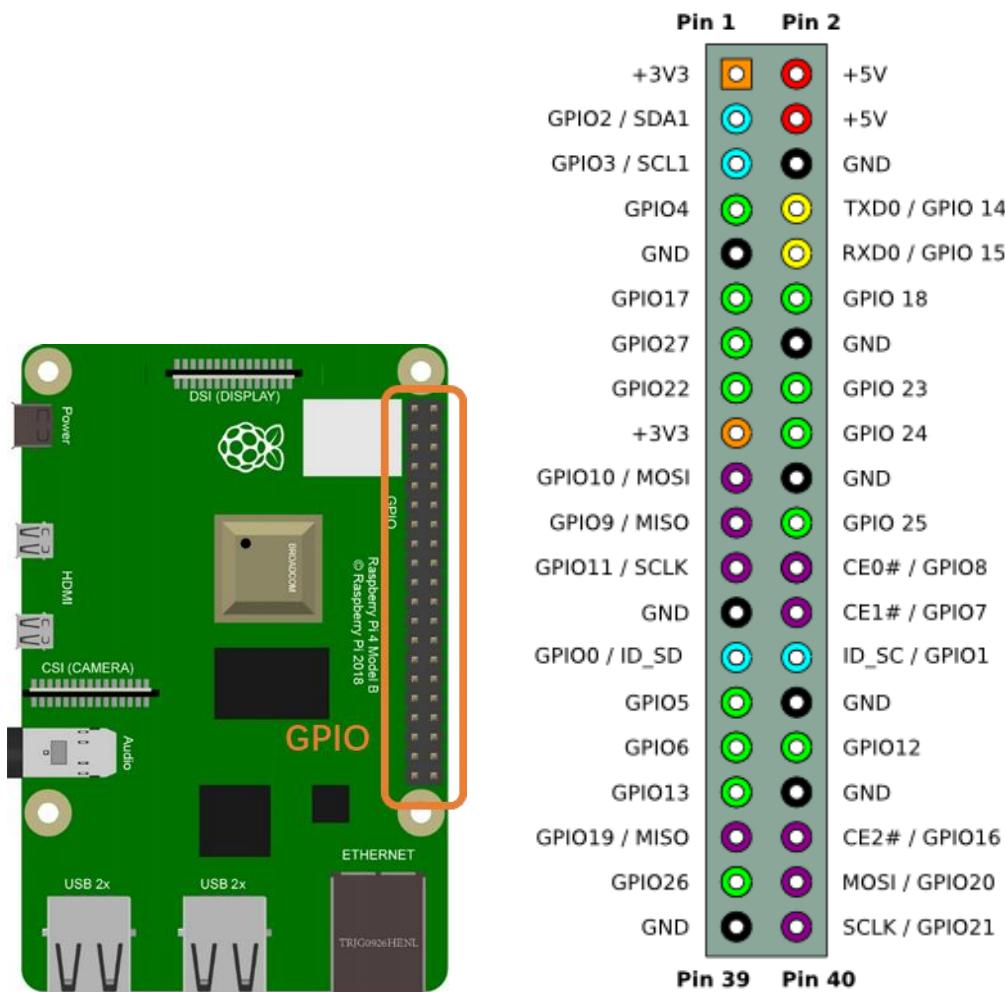
GPIO: General Purpose Input/Output. Here we will introduce the specific function of the pins on the Raspberry Pi and how you can utilize them in all sorts of ways in your projects. Most RPi Module pins can be used as either an input or output, depending on your program and its functions.

When programming GPIO pins there are 3 different ways to reference them: GPIO Numbering, Physical Numbering and 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.

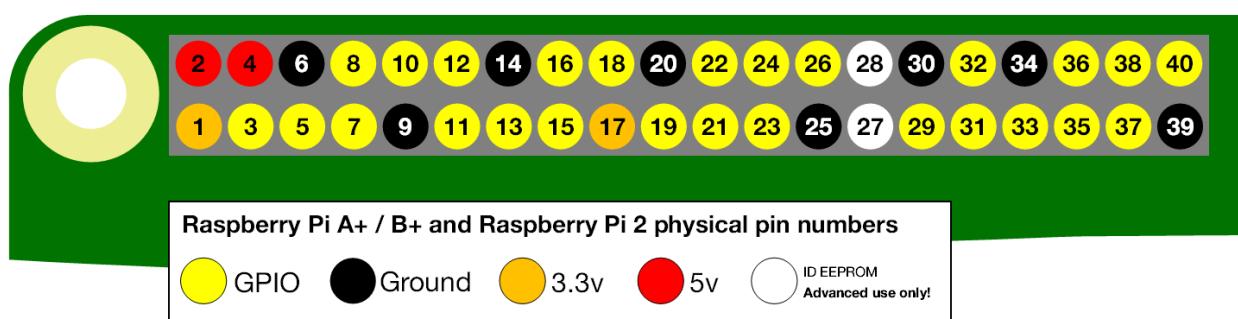
Each pin's functional assignment is defined in the image below:



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

## PHYSICAL Numbering

Another way to refer to the pins is by simply counting across and down from pin 1 at the top left (nearest to the SD card). This is 'Physical Numbering', as shown below:



## WiringPi GPIO Numbering

Different from the previous two types of GPIO serial numbers, RPi GPIO serial number of the WiringPi are numbered according to the BCM chip use in RPi.

wiringPi Pin	BCM GPIO	Name	Header	Name	BCM GPIO	wiringPi Pin	
wiringPi Pin	BCM GPIO	Name	Header	Name	BCM GPIO	wiringPi Pin	
—	—	3.3v	1   2	5v	—	—	For A+, B+, 2B, 3B, 3B+, 4B, Zero
8	R1:0/R2:2	SDA	3   4	5v	—	—	
9	R1:1/R2:3	SCL	5   6	0v	—	—	
7	4	GPIO7	7   8	TxD	14	15	
—	—	0v	9   10	RxD	15	16	
0	17	GPIO0	11   12	GPIO1	18	1	
2	R1:21/R2:27	GPIO2	13   14	0v	—	—	
3	22	GPIO3	15   16	GPIO4	23	4	
—	—	3.3v	17   18	GPIO5	24	5	
12	10	MOSI	19   20	0v	—	—	
13	9	MISO	21   22	GPIO6	25	6	
14	11	SCLK	23   24	CE0	8	10	
—	—	0v	25   26	CE1	7	11	
30	0	SDA.0	27   28	SCL.0	1	31	
21	5	GPIO.21	29   30	0V	—	—	
22	6	GPIO.22	31   32	GPIO.26	12	26	
23	13	GPIO.23	33   34	0V	—	—	
24	19	GPIO.24	35   36	GPIO.27	16	27	
25	26	GPIO.25	37   38	GPIO.28	20	28	
		0V	39   40	GPIO.29	21	29	

For Pi B

(For more details, please refer to <https://projects.drogon.net/raspberry-pi/wiringpi/pins/> )

You can also use the following command to view their correlation.

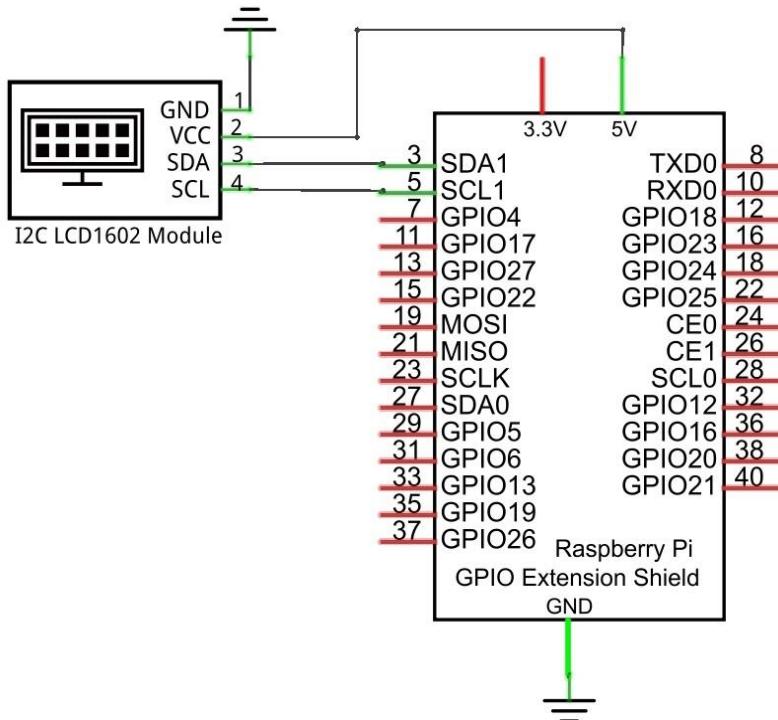
```
gpio readall
```

Pi 4B											
BCM	wPi	Name	Mode	V	Physical	V	Mode	Name	wPi	BCM	
		3.3v			1	2		5v			
2	8	SDA.1	ALT0	1	3	4		5v			
3	9	SCL.1	ALT0	1	5	6		0v			
4	7	GPIO. 7	IN	1	7	8	0	IN	TxD	15	14
		0v			9	10	1	IN	RxD	16	15
17	0	GPIO. 0	IN	0	11	12	0	IN	GPIO. 1	1	18
27	2	GPIO. 2	IN	0	13	14		0v			
22	3	GPIO. 3	IN	0	15	16	0	IN	GPIO. 4	4	23
		3.3v			17	18	0	IN	GPIO. 5	5	24
10	12	MOSI	IN	0	19	20		0v			
9	13	MISO	IN	0	21	22	0	IN	GPIO. 6	6	25
11	14	SCLK	IN	0	23	24	1	IN	CE0	10	8
		0v			25	26	1	IN	CE1	11	7
0	30	SDA.0	IN	1	27	28	1	IN	SCL.0	31	1
5	21	GPIO.21	IN	1	29	30		0v			
6	22	GPIO.22	IN	1	31	32	0	IN	GPIO.26	26	12
13	23	GPIO.23	IN	0	33	34		0v			
19	24	GPIO.24	IN	0	35	36	0	IN	GPIO.27	27	16
26	25	GPIO.25	IN	0	37	38	0	IN	GPIO.28	28	20
		0v			39	40	0	IN	GPIO.29	29	21

## Circuit

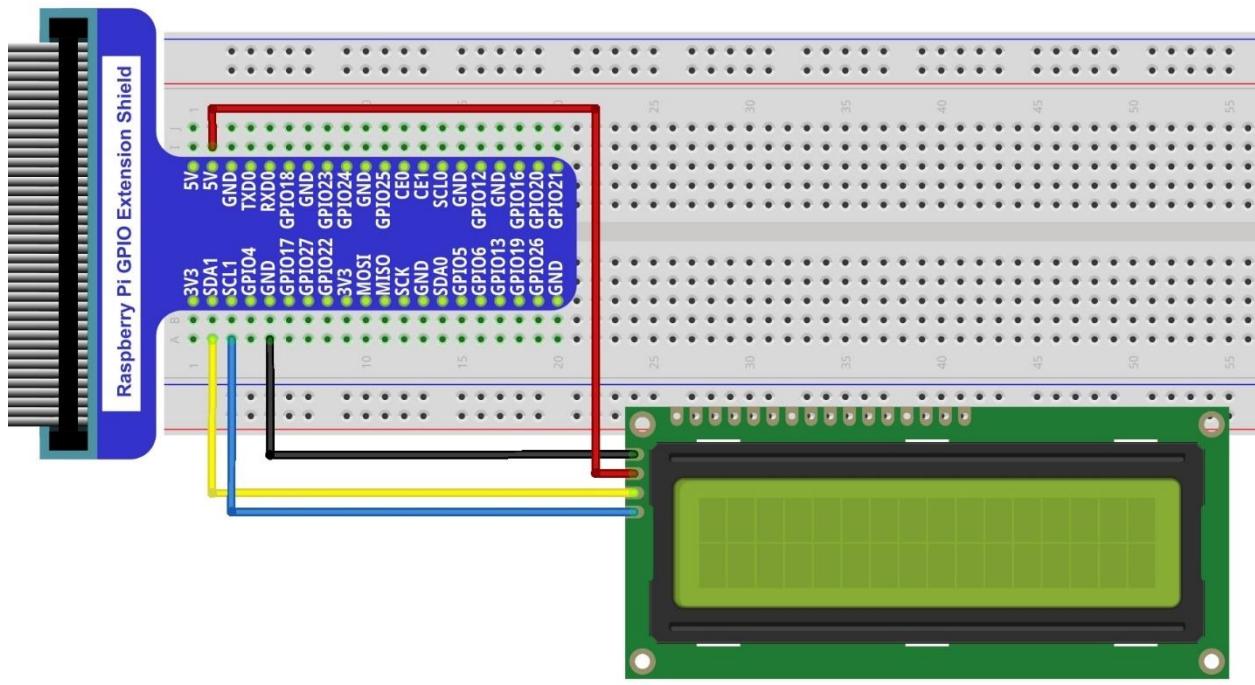
Note that the power supply for I2C LCD1602 in this circuit is 5V.

Schematic diagram



Hardware connection. If you need any support, please feel free to contact us via: [support@freenove.com](mailto:support@freenove.com)

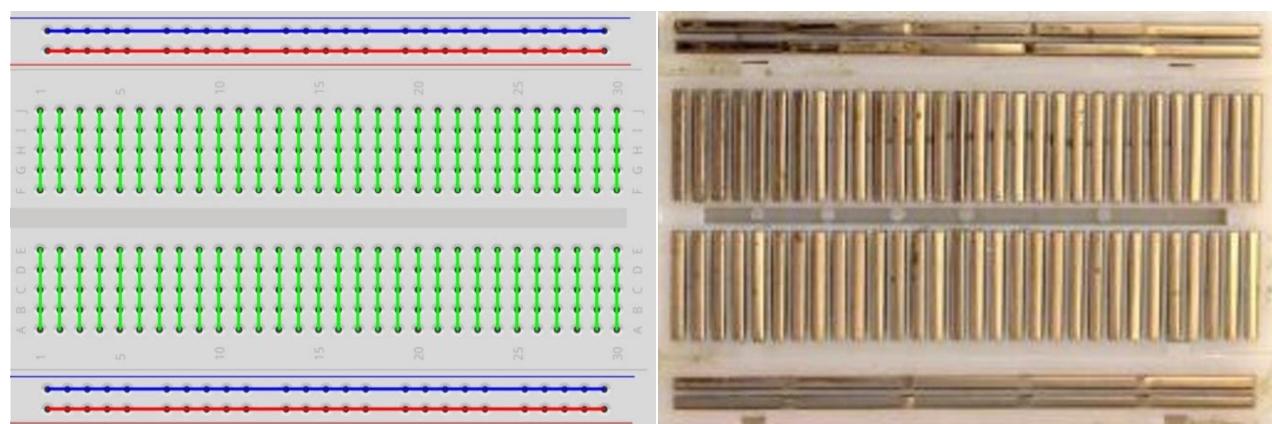
**NOTE: It is necessary to configure 12C and install Smbus first.**



## Component knowledge

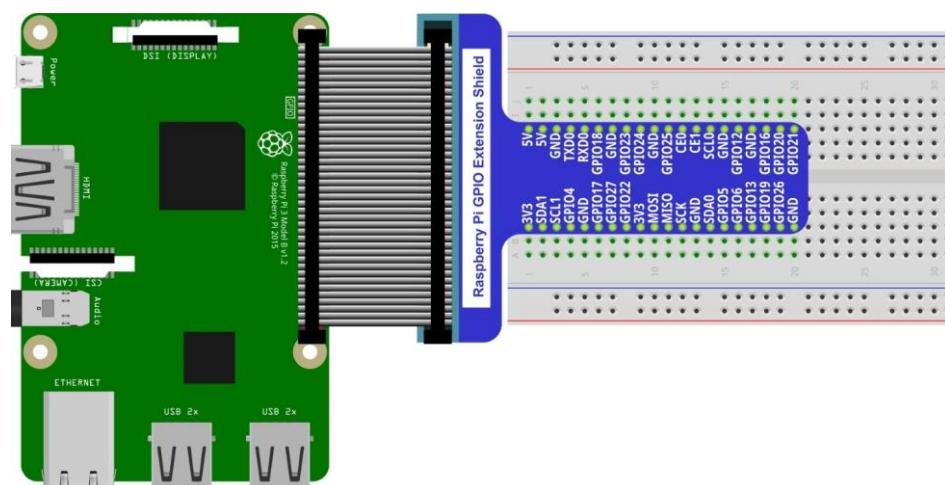
### Breadboard

Here we have a small breadboard as an example of how the rows of holes (sockets) are electrically attached. The left picture shows the ways the pins have shared electrical connection and the right picture shows the actual internal metal, which connect these rows electrically.



### GPIO Extension Board

GPIO board is a convenient way to connect the RPi I/O ports to the breadboard directly. The GPIO pin sequence on Extension Board is identical to the GPIO pin sequence of RPi.



### I2C communication

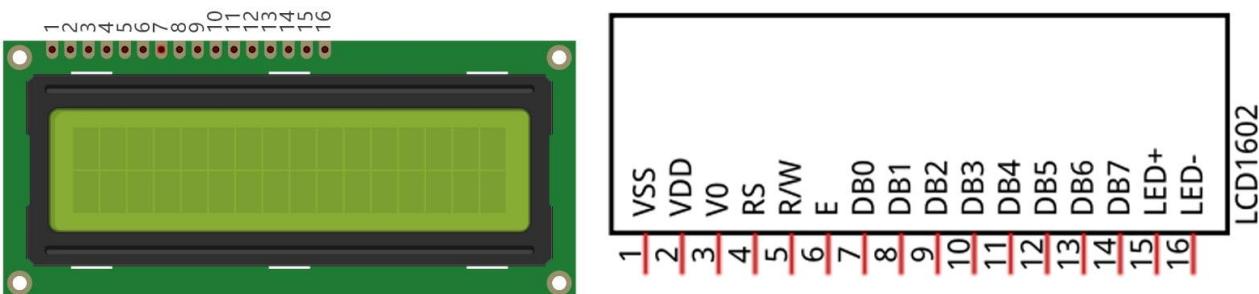
I2C (Inter-Integrated Circuit) has a two-wire serial communication mode, which can be used to connect a micro-controller and its peripheral equipment. Devices using I2C communications must be connected to the serial data line (SDA), and serial clock line (SCL) (called I2C bus). Each device has a unique address which can be used as a transmitter or receiver to communicate with devices connected via the bus.

### LCD1602 communication

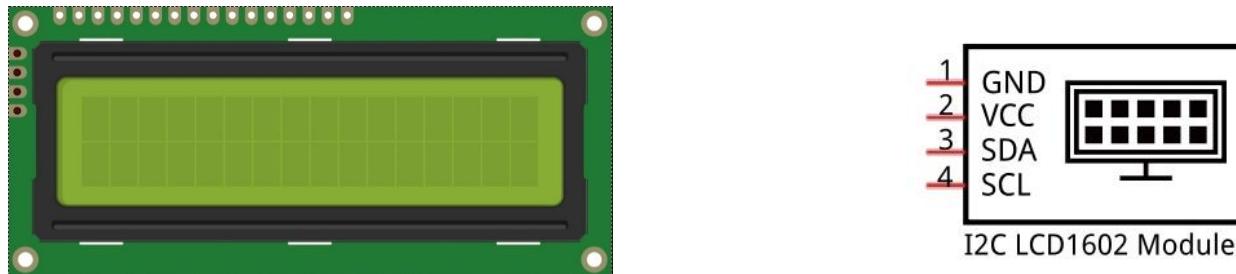
There are LCD1602 display screen and the I2C LCD. We will introduce both of them in this chapter. But what we use in this project is an I2C LCD1602 display screen. The LCD1602 Display Screen can display 2 lines of



characters in 16 columns. It is capable of displaying numbers, letters, symbols, ASCII code and so on. As shown below is a monochrome LCD1602 Display Screen along with its circuit pin diagram

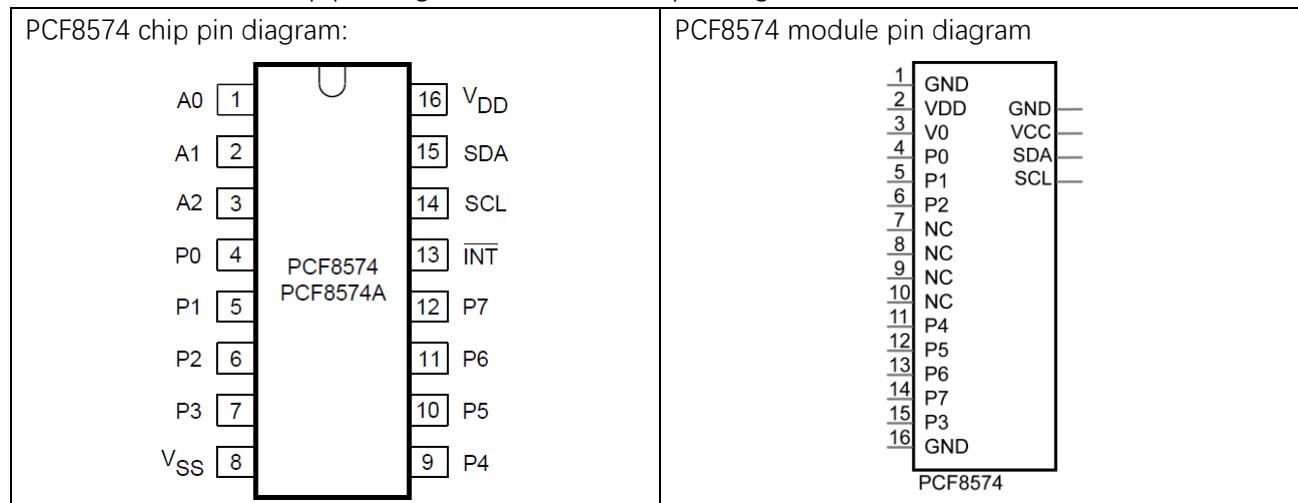


I2C LCD1602 Display Screen integrates a I2C interface, which connects the serial-input & parallel-output module to the LCD1602 Display Screen. This allows us to only use 4 lines to operate the LCD1602.

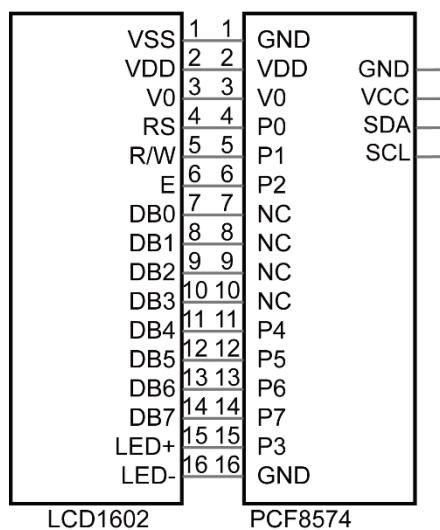


The serial-to-parallel IC chip used in this module is PCF8574T (PCF8574AT), and its default I2C address is 0x27(0x3F). You can also view the RPI bus on your I2C device address through command "i2cdetect -y 1" (refer to the "configuration I2C" section below).

Below is the PCF8574 chip pin diagram and its module pin diagram:



PCF8574 module pins and LCD1602 pins correspond to each other and connected to each other:



Because of this, as stated earlier, we only need 4 pins to control the 16 pins of the LCD1602 Display Screen through the I2C interface.

In this project, we will use the I2C LCD1602 to display some static characters and dynamic variables.

## Configure I2C and Install Smbus

If you have already configured I2C and installed Smbus, skip this section. If you have not, proceed with the following configuration.

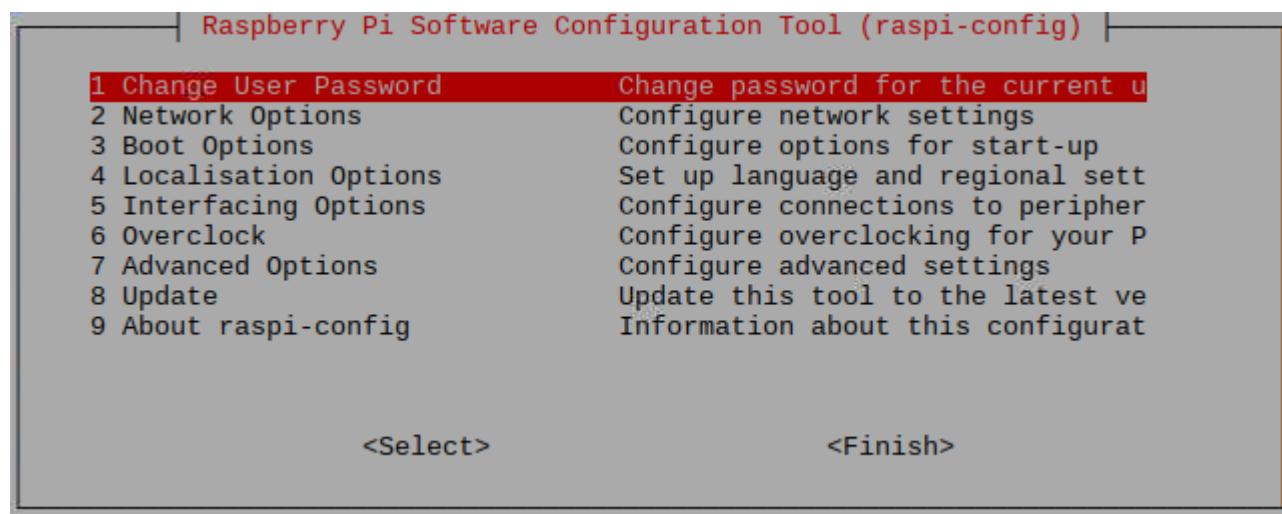
### Enable I2C

The I2C interface in Raspberry Pi is disabled by default. You will need to open it manually and enable the I2C interface as follows:

Type command in the Terminal:

```
sudo raspi-config
```

Then open the following dialog box:



Choose “5 Interfacing Options” then “P5 I2C” then “Yes” and then “Finish” in this order and restart your RPi. The I2C module will then be started.

Type a command to check whether the I2C module is started:

```
lsmod | grep i2c
```

If the I2C module has been started, the following content will be shown. “bcm2708” refers to the CPU model. Different models of Raspberry Pi display different contents depending on the CPU installed:

```
pi@raspberrypi:~ $ lsmod | grep i2c
i2c_bcm2708          4770  0
i2c_dev              5859  0
pi@raspberrypi:~ $
```

## Install I2C-Tools

Next, type the command to install I2C-Tools. It is available with the Raspberry Pi OS by default.

```
sudo apt-get install i2c-tools
```

I2C device address detection:

```
i2cdetect -y 1
```

When you use the serial-parallel IC chip PCF8574T, its I2C default address is 0x27. When the serial-parallel IC chip you use is PCF8574AT, its I2C default address is 0x3F.

When you use the serial-parallel IC chip PCF8574T, the result should look like this:

```
pi@raspberrypi:~ $ i2cdetect -y 1
      0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00: --
10: --
20: -- 27 -- -- -- -- -- --
30: --
40: --
50: --
60: --
70: --
```

Here, 27 (HEX) is the I2C address of LCD2004 Module (PCF8574T).

When you are using PCF8574AT, and its default I2C address is 0x3F.

## Install Smbus Module

```
sudo apt-get install python-smbus
sudo apt-get install python3-smbus
```

## Code

This code will have your RPi's CPU temperature and System Time Displayed on the LCD1602.

### C Code 1.1 I2CLCD1602

If you did not [configure I2C and install Smbus](#), please complete the configuration and installation. If you did, please continue.

First, observe the project result, and then learn about the code in detail.

**If you have any concerns, please contact us via: support@freenove.com**

1. Use cd command to enter 1.1\_I2CLCD1602 directory of C code.

```
cd ~/Freenove_Kit/Freenove_LCD_Module_for_Raspberry_Pi/Code/C_Code/1.1_I2CLCD1602
```

2. Use following command to compile "I2CLCD1602.c" and generate executable file "I2CLCD1602".

```
gcc I2CLCD1602.c -o I2CLCD1602 -lwiringPi -lwiringPiDev
```

3. Then run the generated file "I2CLCD1602".

```
sudo ./I2CLCD1602
```

After the program is executed, the LCD1602 Screen will display your RPi's CPU Temperature and System Time.



**NOTE: After the program is executed, if you cannot see anything on the display or the display is not clear, try rotating the white knob on back of LCD1602 slowly, which adjusts the contrast, until the screen can display the Time and Temperature clearly.**



The following is the program code:

1	#include <stdlib.h>
2	#include <stdio.h>

```
3 #include <wiringPi.h>
4 #include <wiringPiI2C.h>
5 #include <pcf8574.h>
6 #include <lcd.h>
7 #include <time.h>
8
9 int pcf8574_address = 0x27;           // PCF8574T:0x27, PCF8574AT:0x3F
10 #define BASE 64          // BASE any number above 64
11 //Define the output pins of the PCF8574, which are directly connected to the LCD1602 pin.
12 #define RS     BASE+0
13 #define RW     BASE+1
14 #define EN     BASE+2
15 #define LED    BASE+3
16 #define D4     BASE+4
17 #define D5     BASE+5
18 #define D6     BASE+6
19 #define D7     BASE+7
20
21 int lcdhd;// used to handle LCD
22 void printCPUtemperature() { // sub function used to print CPU temperature
23     FILE *fp;
24     char str_temp[15];
25     float CPU_temp;
26     // CPU temperature data is stored in this directory.
27     fp=fopen("/sys/class/thermal/thermal_zone0/temp","r");
28     fgets(str_temp, 15, fp);      // read file temp
29     CPU_temp = atof(str_temp)/1000.0; // convert to Celsius degrees
30     printf("CPU's temperature : %.2f \n",CPU_temp);
31     lcdPosition(lcdhd,0,0);      // set the LCD cursor position to (0,0)
32     lcdPrintf(lcdhd,"CPU:%.2fC",CPU_temp); // Display CPU temperature on LCD
33     fclose(fp);
34 }
35 void printDataTime() { //used to print system time
36     time_t rawtime;
37     struct tm *timeinfo;
38     time(&rawtime); // get system time
39     timeinfo = localtime(&rawtime); //convert to local time
40     printf("%s \n",asctime(timeinfo));
41     lcdPosition(lcdhd,0,1); // set the LCD cursor position to (0,1)
42
43     lcdPrintf(lcdhd,"Time:%02d:%02d:%02d",timeinfo->tm_hour,timeinfo->tm_min,timeinfo->tm_sec);
44     //Display system time on LCD
45 }
46 int detectI2C(int addr){ //Used to detect i2c address of LCD
```

```
47     int _fd = wiringPiI2CSetup (addr);
48     if (_fd < 0) {
49         printf("Error address : 0x%x \n",addr);
50         return 0 ;
51     }
52     else{
53         if(wiringPiI2CWrite(_fd,0) < 0) {
54             printf("Not found device in address 0x%x \n",addr);
55             return 0;
56         }
57         else{
58             printf("Found device in address 0x%x \n",addr);
59             return 1 ;
60         }
61     }
62 }
63 int main(void) {
64     int i;
65     printf("Program is starting ... \n");
66     wiringPiSetup();
67     if(detectI2C(0x27)) {
68         pcf8574_address = 0x27;
69     }else if(detectI2C(0x3F)) {
70         pcf8574_address = 0x3F;
71     }else{
72         printf("No correct I2C address found, \n"
73             "Please use command 'i2cdetect -y 1' to check the I2C address! \n"
74             "Program Exit. \n");
75         return -1;
76     }
77     pcf8574Setup(BASE,pcf8574_address);//initialize PCF8574
78     for(i=0;i<8;i++) {
79         pinMode(BASE+i,OUTPUT);      //set PCF8574 port to output mode
80     }
81     digitalWrite(LED,HIGH);      //turn on LCD backlight
82     digitalWrite(RW,LOW);        //allow writing to LCD
83     lcdhd = lcdInit(2,16,4,RS,EN,D4,D5,D6,D7,0,0,0,0); // initialize LCD and return "handle"
used to handle LCD
84     if(lcdhd == -1){
85         printf("lcdInit failed !");
86         return 1;
87     }
88     while(1){
89         printCPUtemperature(); //print CPU temperature
```

```

90     printDateTime();      // print system time
91     delay(1000);
92 }
93 return 0;
94 }
```

First, define the I2C address of the PCF8574 and the Extension of the GPIO pin, which is connected to the GPIO pin of the LCD1602. LCD1602 has two different i2c addresses. Set 0x27 as default.

```

int pcf8574_address = 0x27;          // PCF8574T:0x27, PCF8574AT:0x3F
#define BASE 64                  // BASE any number above 64
//Define the output pins of the PCF8574, which are directly connected to the LCD1602 pin.
#define RS      BASE+0
#define RW      BASE+1
#define EN      BASE+2
#define LED     BASE+3
#define D4      BASE+4
#define D5      BASE+5
#define D6      BASE+6
#define D7      BASE+7
```

Then, in main function, initialize the PCF8574, set all the pins to output mode, and turn ON the LCD1602 backlight (without the backlight the Display is difficult to read).

```

pcf8574Setup(BASE, pcf8574_address); // initialize PCF8574
for(i=0; i<8; i++) {
    pinMode(BASE+i, OUTPUT);      // set PCF8574 port to output mode
}
digitalWrite(LED, HIGH);           // turn on LCD backlight
```

Then use `lcdInit()` to initialize LCD1602 and set the `RW` pin of LCD1602 to 0 (can be written) according to requirements of this function. The return value of the function called "Handle" is used to handle LCD1602".

```

lcdhd = lcdInit(2, 16, 4, RS, EN, D4, D5, D6, D7, 0, 0, 0, 0); // initialize LCD and return
"handle" used to handle LCD
```

Details about `lcdInit()`:

```

int lcdInit (int rows, int cols, int bits, int rs, int strb,
           int d0, int d1, int d2, int d3, int d4, int d5, int d6, int d7);
```

This is the main initialization function and must be executed first before you use any other LCD functions.

**Rows** and **cols** are the rows and columns of the Display (e.g. 2, 16 or 4, 20). **Bits** is the number of how wide the number of bits is on the interface (4 or 8). The **rs** and **strb** represent the pin numbers of the Display's RS pin and Strobe (E) pin. The parameters **d0** through **d7** are the pin numbers of the 8 data pins connected from the RPi to the display. Only the first 4 are used if you are running the display in 4-bit mode.

The return value is the 'handle' to be used for all subsequent calls to the lcd library when dealing with that LCD, or -1 to indicate a fault (usually incorrect parameter)

For more details about LCD Library, please refer to: <https://projects.drogon.net/raspberry-pi/wiringpi/lcd-library/>

In the next "while", two subfunctions are called to display the RPi's CPU Temperature and the SystemTime.

First look at subfunction printCPUtemperature(). The CPU temperature data is stored in the "/sys/class/thermal/thermal\_zone0/temp" file. We need to read the contents of this file, which converts it to temperature value stored in variable CPU\_temp and uses lcdPrintf() to display it on LCD.

```
void printCPUtemperature() { //subfunction used to print CPU temperature

    FILE *fp;
    char str_temp[15];
    float CPU_temp;
    // CPU temperature data is stored in this directory.
    fp=fopen("/sys/class/thermal/thermal_zone0/temp", "r");
    fgets(str_temp, 15, fp);      // read file temp
    CPU_temp = atof(str_temp)/1000.0; // convert to Celsius degrees
    printf("CPU's temperature : %.2f \n", CPU_temp);
    lcdPosition(lcdhd, 0, 0);     // set the LCD cursor position to (0,0)
    lcdPrintf(lcdhd, "CPU:%.2fC", CPU_temp); // Display CPU temperature on LCD
    fclose(fp);
}
```

Details about lcdPosition() and lcdPrintf():

#### **lcdPosition (int handle, int x, int y);**

Set the position of the cursor for subsequent text entry.

#### **lcdPutchar (int handle, uint8\_t data)**

#### **lcdPuts (int handle, char \*string)**

#### **lcdPrintf (int handle, char \*message, ...)**

These output a single ASCII character, a string or a formatted string using the usual print formatting commands to display individual characters (it is how you are able to see characters on your computer monitor).

Next is subfunction printDataTime() used to display System Time. First, it gets the Standard Time and stores it into variable Rawtime, and then converts it to the Local Time and stores it into timeinfo, and finally displays the Time information on the LCD1602 Display.

```
void printDataTime() { //used to print system time

    time_t rawtime;
    struct tm *timeinfo;
    time(&rawtime); // get system time
    timeinfo = localtime(&rawtime); // convert to local time
    printf("%s \n", asctime(timeinfo));
    lcdPosition(lcdhd, 0, 1); // set the LCD cursor position to (0,1)
    lcdPrintf(lcdhd, "Time:%d:%d:%d", timeinfo->tm_hour, timeinfo->tm_min, timeinfo->tm_sec);
    //Display system time on LCD
}
```

## Python Code 1.1 I2CLCD1602

If you did not [configure I2C and install Smbus](#), please complete the configuration and installation. If you did, please continue.

First, observe the project result, and then learn about the code in detail.

**If you have any concerns, please contact us via: support@freenove.com**

1. Use cd command to enter 1.1\_I2CLCD1602 directory of Python code.

```
cd ~/Freenove_Kit/Freenove_LCD_Module_for_Raspberry_Pi/Code/Python_Code/1.1_I2CLCD1602
```

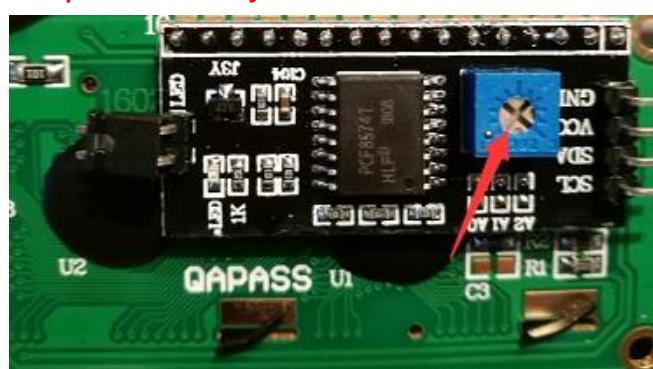
2. Use Python command to execute Python code "I2CLCD1602.py".

```
python I2CLCD1602.py
```

After the program is executed, the LCD1602 Screen will display your RPi's CPU Temperature and System Time.



**NOTE:** After the program is executed, if you cannot see anything on the display or the display is not clear, try rotating the white knob on back of LCD1602 slowly, which adjusts the contrast, until the screen can display the Time and Temperature clearly.



The following is the program code:

```

1  from PCF8574 import PCF8574_GPIO
2  from Adafruit_LCD1602 import Adafruit_CharLCD
3
4  from time import sleep, strftime
5  from datetime import datetime
6
7

```

```
8 def get_cpu_temp():      # get CPU temperature and store it into file
9     "/sys/class/thermal/thermal_zone0/temp"
10    tmp = open('/sys/class/thermal/thermal_zone0/temp')
11    cpu = tmp.read()
12    tmp.close()
13    return '{:.2f}'.format(float(cpu)/1000) + ' C'
14
15 def get_time_now():      # get system time
16     return datetime.now().strftime('%H:%M:%S')
17
18 def loop():
19     mcp.output(3, 1)      # turn on LCD backlight
20     lcd.begin(16, 2)      # set number of LCD lines and columns
21     while(True):
22         #lcd.clear()
23         lcd.setCursor(0, 0) # set cursor position
24         lcd.message('CPU: ' + get_cpu_temp()+'\n')# display CPU temperature
25         lcd.message(get_time_now())    # display the time
26         sleep(1)
27
28 def destroy():
29     lcd.clear()
30
31 PCF8574_address = 0x27 # I2C address of the PCF8574 chip.
32 PCF8574A_address = 0x3F # I2C address of the PCF8574A chip.
33 # Create PCF8574 GPIO adapter.
34 try:
35     mcp = PCF8574_GPIO(PCF8574_address)
36 except:
37     try:
38         mcp = PCF8574_GPIO(PCF8574A_address)
39     except:
40         print('I2C Address Error !')
41         exit(1)
42 # Create LCD, passing in MCP GPIO adapter.
43 lcd = Adafruit_CharLCD(pin_rs=0, pin_e=2, pins_db=[4, 5, 6, 7], GPIO=mcp)
44
45 if __name__ == '__main__':
46     print('Program is starting ...')
47     try:
48         loop()
49     except KeyboardInterrupt:
50         destroy()
```

Two modules are used in the code, PCF8574.py and Adafruit\_LCD1602.py. These two documents and the code files are stored in the same directory, and neither of them is dispensable. Please DO NOT DELETE THEM! PCF8574.py is used to provide I2C communication mode and operation method of some of the ports for the RPi and PCF8574 IC Chip. Adafruit module Adafruit\_LCD1602.py is used to provide some functional operation method for the LCD1602 Display.

In the code, first get the object used to operate the PCF8574's port, then get the object used to operate the LCD1602.

```
address = 0x27 # I2C address of the PCF8574 chip.  
# Create PCF8574 GPIO adapter.  
mcp = PCF8574_GPIO(address)  
# Create LCD, passing in MCP GPIO adapter.  
lcd = Adafruit_CharLCD(pin_rs=0, pin_e=2, pins_db=[4, 5, 6, 7], GPIO=mcp)
```

According to the circuit connection, port 3 of PCF8574 is connected to the positive pole of the LCD1602 Display's backlight. Then in the loop () function, use of mcp.output (3,1) to turn the LCD1602 Display's backlight ON and then set the number of LCD lines and columns.

```
def loop():  
    mcp.output(3,1)      # turn on the LCD backlight  
    lcd.begin(16,2)      # set number of LCD lines and columns
```

In the next while loop, set the cursor position, and display the CPU temperature and time.

```
while(True):  
    lcd.clear()  
    lcd.setCursor(0,0) # set cursor position  
    lcd.message('CPU: ' + get_cpu_temp()+'\n')# display CPU temperature  
    lcd.message(get_time_now()) # display the time  
    sleep(1)
```

CPU temperature is stored in file "/sys/class/thermal/thermal\_zone0/temp". Open the file and read content of the file, and then convert it to Celsius degrees and return. Subfunction used to get CPU temperature is shown below:

```
def get_cpu_temp():      # get CPU temperature and store it into file  
    "/sys/class/thermal/thermal_zone0/temp"  
    tmp = open('/sys/class/thermal/thermal_zone0/temp')  
    cpu = tmp.read()  
    tmp.close()  
    return '{:.2f}'.format(float(cpu)/1000) + ' C'
```

Subfunction used to get time:

```
def get_time_now():      # get the time  
    return datetime.now().strftime('%H:%M:%S')
```

Details about PCF8574.py and Adafruit\_LCD1602.py:

### Module PCF8574

This module provides two classes **PCF8574\_I2C** and **PCF8574\_GPIO**.

Class **PCF8574\_I2C**: provides reading and writing method for PCF8574.

Class **PCF8574\_GPIO**: provides a standardized set of GPIO functions.

More information can be viewed through opening PCF8574.py.

Adafruit\_LCD1602 Module

**Module Adafruit\_LCD1602**

This module provides the basic operation method of LCD1602, including class Adafruit\_CharLCD. Some member functions are described as follows:

**def begin(self, cols, lines):** set the number of lines and columns of the screen.

**def clear(self):** clear the screen

**def setCursor(self, col, row):** set the cursor position

**def message(self, text):** display contents

More information can be viewed through opening Adafruit\_CharLCD.py.

# Chapter 2 LCD2004

In the previous chapter, we studied the LCD1602 display. In order to display more content, in this chapter, we will learn about the LCD2004 Display Screen.

## Project 2.1 I<sup>2</sup>C LCD2004

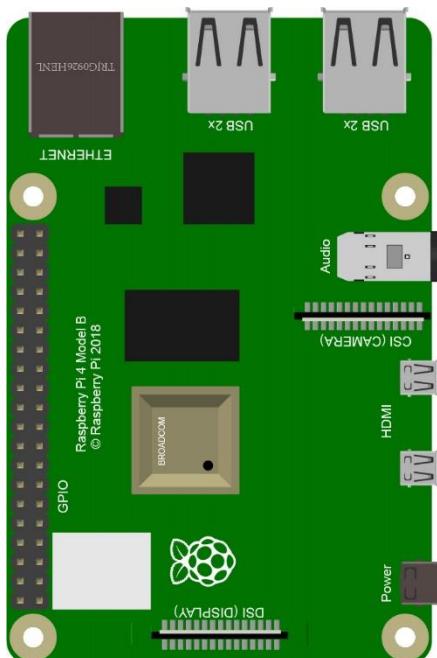
In this section we learn how to use LCD2004 to display something.

### Component List

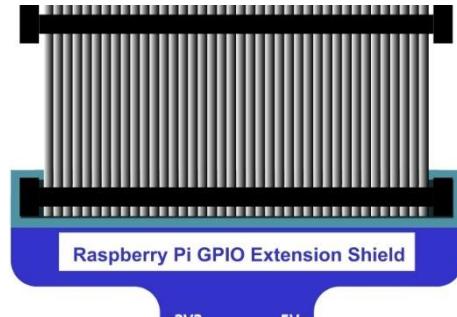
#### Raspberry Pi

(Recommended: Raspberry Pi 4B / 3B+ / 3B)

Compatible: 3A+ / 2B / 1B+ / 1A+ / Zero W / Zero)

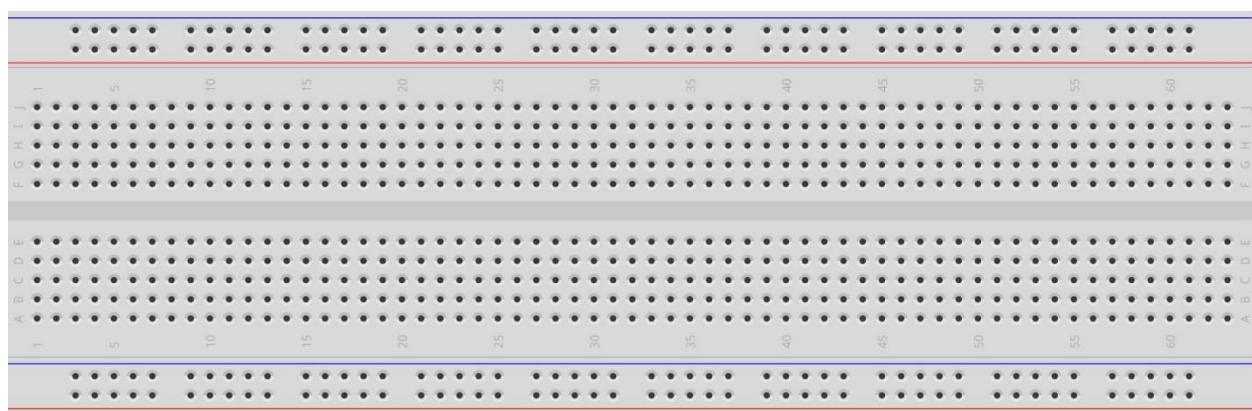


#### GPIO Extension Board & Ribbon Cable



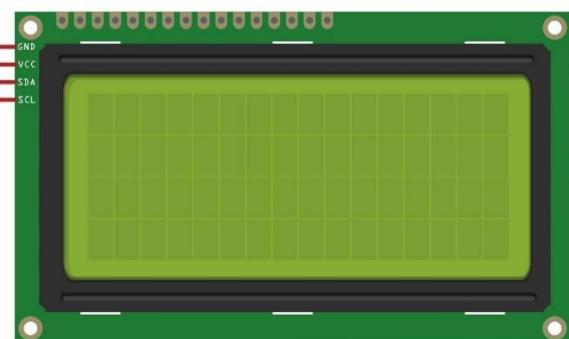
3V3	5V
SDA1	5V
SCL1	GND
GPIO4	TXDO
GND	RXD0
GPIO17	GPIO18
GPIO27	GND
GPIO22	GPIO23
3V3	GPIO24
MOSI	GND
MISO	GPIO25
SCK	CE0
GND	CE1
SDA0	SCL0
GPIO5	GND
GPIO6	GPIO12
GPIO13	GND
GPIO19	GPIO16
GPIO26	GPIO20
GND	GPIO21

#### Breadboard x1





I2C LCD2004 Module x1



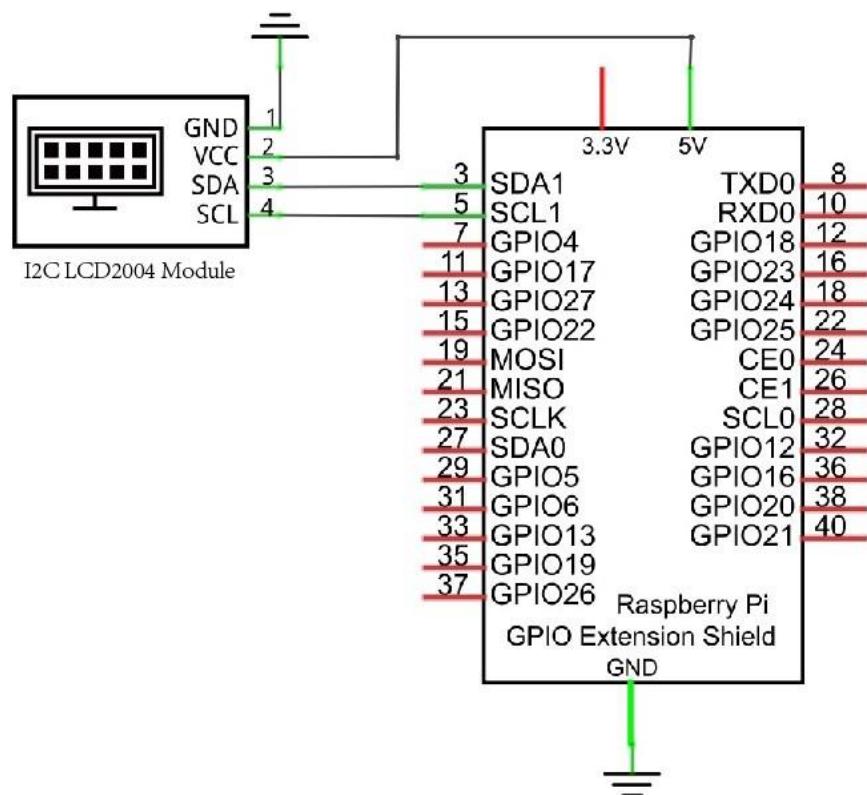
Jumper Wire x4



## Circuit

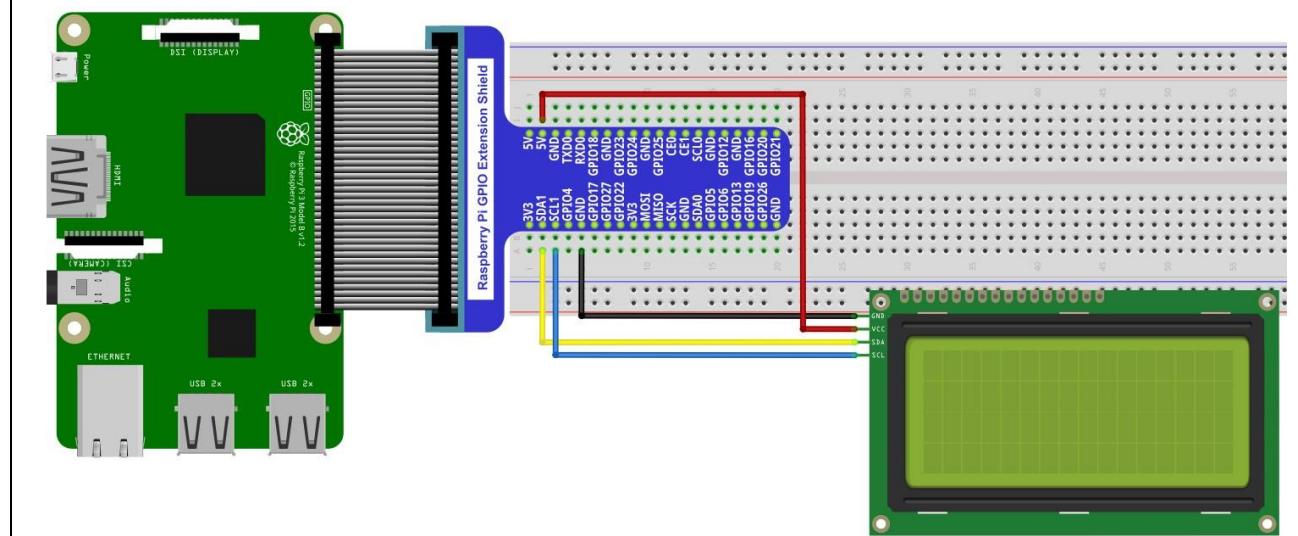
Note that the power supply for I2C LCD2004 in this circuit is 5V.

Schematic diagram



Hardware connection. If you need any support, please feel free to contact us via: [support@freenove.com](mailto:support@freenove.com)

**NOTE: It is necessary to configure 12C and install Smbus first.**



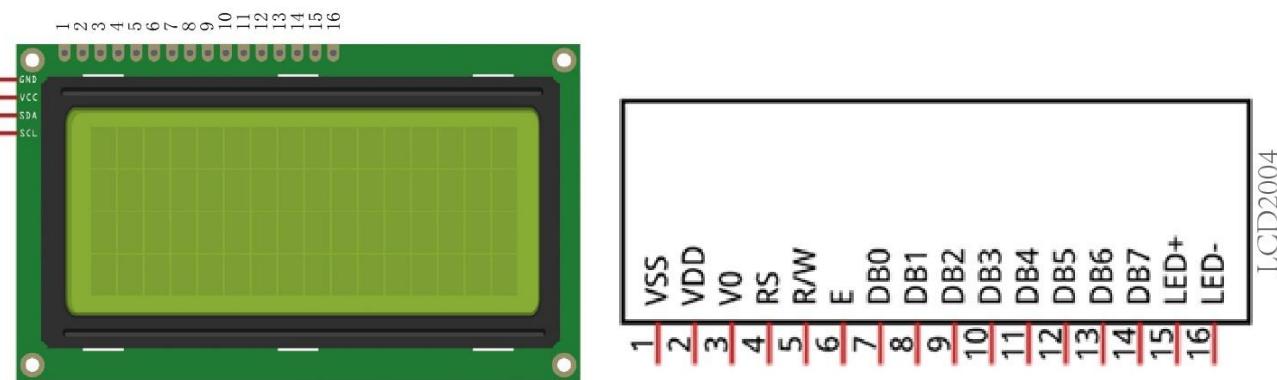
## Component knowledge

### I2C communication

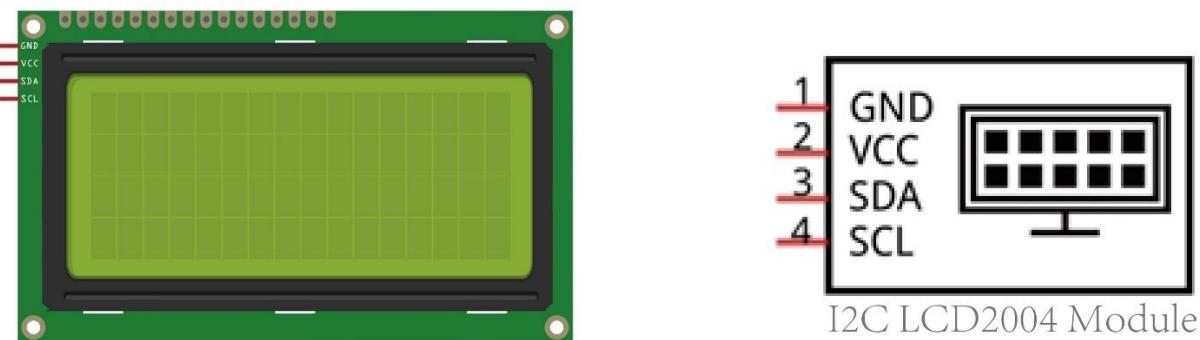
I2C (Inter-Integrated Circuit) has a two-wire serial communication mode, which can be used to connect a micro-controller and its peripheral equipment. Devices using I2C communications must be connected to the serial data line (SDA), and serial clock line (SCL) (called I2C bus). Each device has a unique address which can be used as a transmitter or receiver to communicate with devices connected via the bus.

### LCD2004 communication

There are LCD2004 display screen and the I2C LCD. We will introduce both of them in this chapter. But what we use in this project is an I2C LCD2004 display screen. The LCD2004 Display Screen can display 4 lines of characters in 20 columns. It is capable of displaying numbers, letters, symbols, ASCII code and so on. As shown below is a monochrome LCD2004 Display Screen along with its circuit pin diagram.

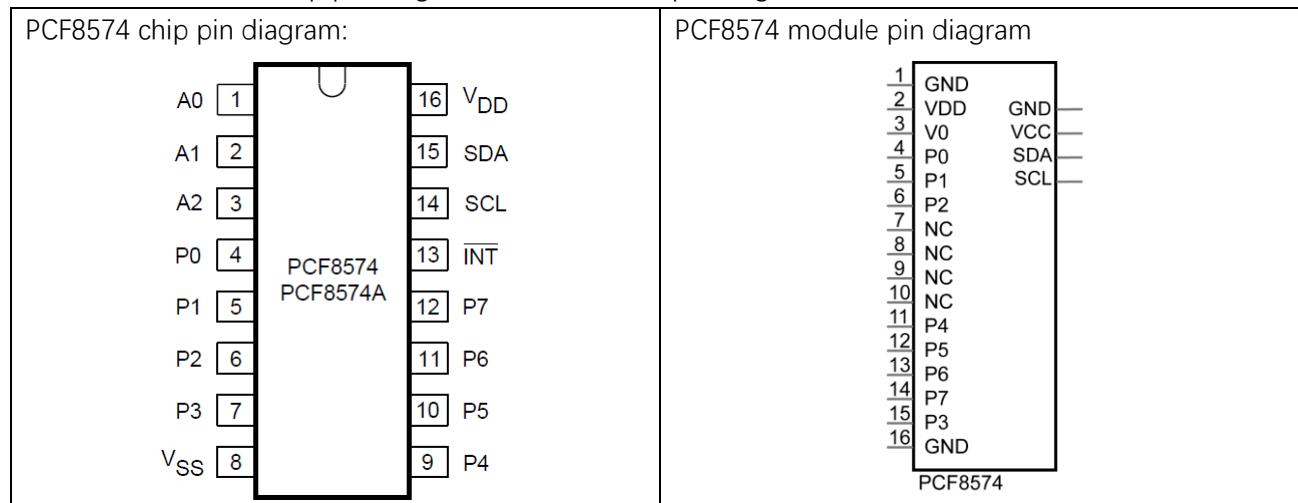


I2C LCD2004 display screen integrates a I2C interface, which connects the serial-input & parallel-output module to the LCD2004 display screen. This allows us to only use 4 lines to the operate the LCD2004.

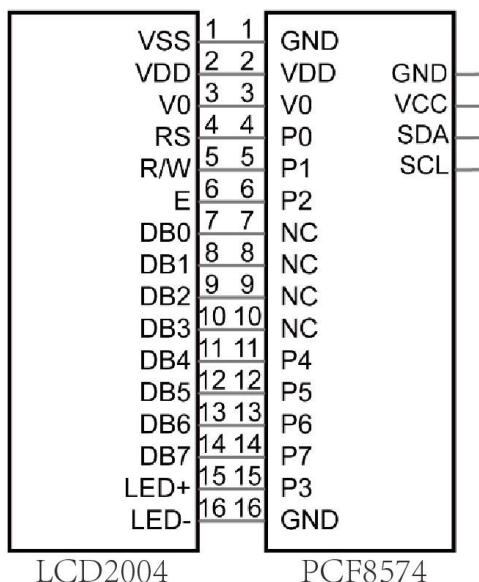


The serial-to-parallel IC chip used in this module is PCF8574T (PCF8574AT), and its default I2C address is 0x27(0x3F). You can also view the RPI bus on your I2C device address through command "i2cdetect -y 1".

Below is the PCF8574 chip pin diagram and its module pin diagram:



PCF8574 module pins and LCD1602 pins correspond to each other and connected to each other:



Because of this, as stated earlier, we only need 4 pins to control the 16 pins of the LCD2004 Display Screen through the I<sup>2</sup>C interface.

In this project, we will use the I2C LCD2004 to display some static characters and dynamic variables.

## Configure I2C and Install Smbus

If you did not configure I2C and install Smbus, please complete the configuration and installation. If you have, skip this section.

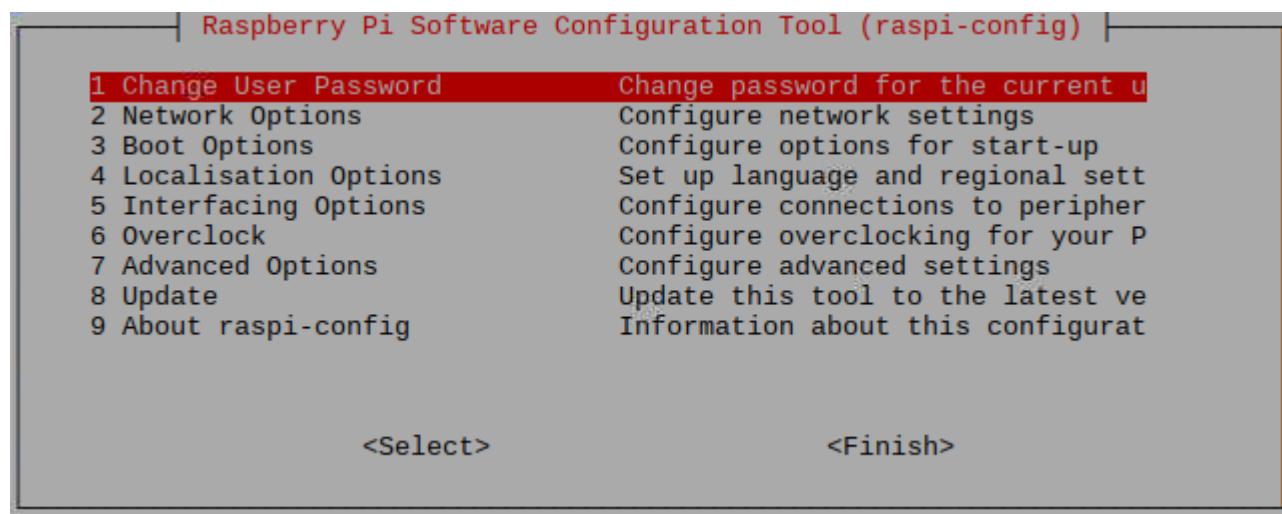
### Enable I2C

The I2C interface in Raspberry Pi is disabled by default. You will need to open it manually and enable the I2C interface as follows:

Type command in the Terminal:

```
sudo raspi-config
```

Then open the following dialog box:



Choose “5 Interfacing Options” then “P5 I2C” then “Yes” and then “Finish” in this order and restart your RPi. The I2C module will then be started.

Type a command to check whether the I2C module is started:

```
lsmod | grep i2c
```

If the I2C module has been started, the following content will be shown. “bcm2708” refers to the CPU model. Different models of Raspberry Pi display different contents depending on the CPU installed:

```
pi@raspberrypi:~ $ lsmod | grep i2c
i2c_bcm2708          4770  0
i2c_dev              5859  0
pi@raspberrypi:~ $
```

## Install I2C-Tools

Next, type the command to install I2C-Tools. It is available with the Raspberry Pi OS by default.

```
sudo apt-get install i2c-tools
```

I2C device address detection:

```
i2cdetect -y 1
```

When you use the serial-parallel IC chip PCF8574T, its I2C default address is 0x27. When the serial-parallel IC chip you use is PCF8574AT, its I2C default address is 0x3F.

When you use the serial-parallel IC chip PCF8574T, the result should look like this:

```
pi@raspberrypi:~ $ i2cdetect -y 1
      0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00: --
10: --
20: -- 27 -- -- -- -- -- -- -- -- -- -- -- --
30: --
40: --
50: --
60: --
70: --
```

Here, 27 (HEX) is the I2C address of LCD2004 Module (PCF8574T).

When you are using PCF8574AT, and its default I2C address is 0x3F.

## Install Smbus Module

```
sudo apt-get install python-smbus
sudo apt-get install python3-smbus
```

## Code

This code will have your RPi's CPU temperature and System Time Displayed on the LCD2004.

### C Code 2.1 I2CLCD2004

If you did not [configure I2C and install Smbus](#), please complete the configuration and installation. If you did, please continue.

First, observe the project result, and then learn about the code in detail.

**If you have any concerns, please contact us via: [support@freenove.com](mailto:support@freenove.com)**

4. Use cd command to enter 1.1\_I2CLCD2004 directory of C code.

```
cd ~/Freenove_Kit/Freenove_LCD_Module_for_Raspberry_Pi/Code/C_Code/1.1_I2CLCD2004
```

5. Use following command to compile "I2CLCD2004.c" and generate executable file "I2CLCD2004".

```
gcc I2CLCD2004.c -o I2CLCD2004 -lwiringPi -lwiringPiDev
```

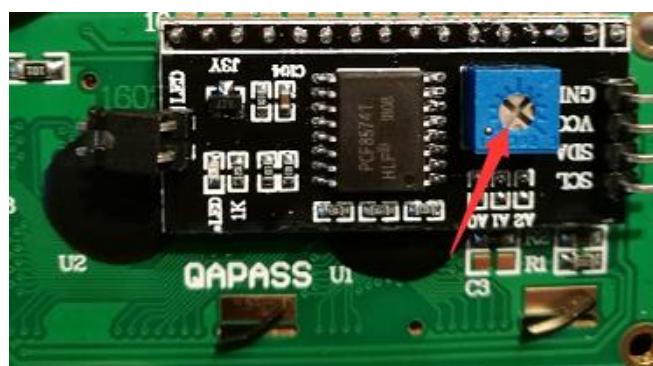
6. Then run the generated file "I2CLCD2004".

```
sudo ./I2CLCD2004
```

After the program is executed, the LCD2004 Screen will display your RPi's CPU Temperature and System Time.



**NOTE:** After the program is executed, if you cannot see anything on the display or the display is not clear, try rotating the white knob on back of LCD2004 slowly, which adjusts the contrast, until the screen can display the Time and Temperature clearly.



The following is the program code:

```
1 #include <stdlib.h>
2 #include <stdio.h>
3 #include <wiringPi.h>
4 #include <wiringPiI2C.h>
5 #include <pcf8574.h>
6 #include <lcd.h>
7 #include <time.h>
8
9 int pcf8574_address = 0x27;           // PCF8574T:0x27, PCF8574AT:0x3F
10 #define BASE 64          // BASE any number above 64
11 //Define the output pins of the PCF8574, which are directly connected to the LCD2004 pin.
12 #define RS      BASE+0
13 #define RW      BASE+1
14 #define EN      BASE+2
15 #define LED     BASE+3
16 #define D4      BASE+4
17 #define D5      BASE+5
18 #define D6      BASE+6
19 #define D7      BASE+7
20
21 int lcdhd;// used to handle LCD
22 void printCPUtemperature() { // sub function used to print CPU temperature
23     FILE *fp;
24     char str_temp[15];
25     float CPU_temp;
26     // CPU temperature data is stored in this directory.
27     fp=fopen("/sys/class/thermal/thermal_zone0/temp", "r");
28     fgets(str_temp, 15, fp);      // read file temp
29     CPU_temp = atof(str_temp)/1000.0;    // convert to Celsius degrees
30     printf("CPU's temperature : %.2f \n", CPU_temp);
31     lcdPosition(lcdhd, 0, 2);      // set the LCD cursor position to (0, 2)
32     lcdPrintf(lcdhd, "CPU:%.2fC", CPU_temp); // Display CPU temperature on LCD
33     fclose(fp);
34 }
35 void printDataTime() { //used to print system time
36     time_t rawtime;
37     struct tm *timeinfo;
38     time(&rawtime); // get system time
39     timeinfo = localtime(&rawtime); //convert to local time
40     printf("%s \n", asctime(timeinfo));
41     lcdPosition(lcdhd, 0, 3); // set the LCD cursor position to (0, 3)
42
43     lcdPrintf(lcdhd, "Time:%02d:%02d:%02d", timeinfo->tm_hour, timeinfo->tm_min, timeinfo->tm_sec);
```

```
44 //Display system time on LCD
45 }
46 int detectI2C(int addr){ //Used to detect i2c address of LCD
47     int _fd = wiringPiI2CSetup (addr);
48     if (_fd < 0){
49         printf("Error address : 0x%x \n",addr);
50         return 0 ;
51     }
52     else{
53         if(wiringPiI2CWrite(_fd,0) < 0){
54             printf("Not found device in address 0x%x \n",addr);
55             return 0;
56         }
57         else{
58             printf("Found device in address 0x%x \n",addr);
59             return 1 ;
60         }
61     }
62 }
63 int main(void){
64     int i;
65     printf("Program is starting ... \n");
66     wiringPiSetup();
67     if(detectI2C(0x27)){
68         pcf8574_address = 0x27;
69     }else if(detectI2C(0x3F)){
70         pcf8574_address = 0x3F;
71     }else{
72         printf("No correct I2C address found, \n"
73             "Please use command 'i2cdetect -y 1' to check the I2C address! \n"
74             "Program Exit. \n");
75         return -1;
76     }
77     pcf8574Setup(BASE,pcf8574_address);//initialize PCF8574
78     for(i=0;i<8;i++){
79         pinMode(BASE+i,OUTPUT); //set PCF8574 port to output mode
80     }
81     digitalWrite(LED,HIGH); //turn on LCD backlight
82     digitalWrite(RW,LOW); //allow writing to LCD
83     lcdhd = lcdInit(4,20,4,RS,EN,D4,D5,D6,D7,0,0,0,0); // initialize LCD and return "handle"
84     used to handle LCD
85     if(lcdhd == -1){
86         printf("lcdInit failed !");
87         return 1;
```

```

87 }
88 lcdPosition(lcdhd, 0, 0); // set the LCD cursor position to (0,0)
89 lcdPrintf(lcdhd, "FREENOVE");
90 lcdPosition(lcdhd, 0, 1); // set the LCD cursor position to (0,1)
91 lcdPrintf(lcdhd, "www. freenove. com");
92 while(1){
93     printCPUtemperature(); //print CPU temperature
94     printDataTime();      // print system time
95     delay(1000);
96 }
97 return 0;
98 }
```

First, define the I2C address of the PCF8574 and the Extension of the GPIO pin, which is connected to the GPIO pin of the LCD2004. LCD2004 has two different i2c addresses. Set 0x27 as default.

```

int pcf8574_address = 0x27;           // PCF8574T:0x27, PCF8574AT:0x3F
#define BASE 64                      // BASE any number above 64
//Define the output pins of the PCF8574, which are directly connected to the LCD2004 pin.
#define RS    BASE+0
#define RW    BASE+1
#define EN    BASE+2
#define LED   BASE+3
#define D4    BASE+4
#define D5    BASE+5
#define D6    BASE+6
#define D7    BASE+7
```

Then, in main function, initialize the PCF8574, set all the pins to output mode, and turn ON the LCD2004 backlight (without the backlight the Display is difficult to read).

```

pcf8574Setup(BASE, pcf8574_address); // initialize PCF8574
for(i=0;i<8;i++){
    pinMode(BASE+i, OUTPUT); // set PCF8574 port to output mode
}
digitalWrite(LED, HIGH); // turn on LCD backlight
```

Then use `lcdInit()` to initialize LCD2004 and set the `RW` pin of LCD2004 to 0 (can be written) according to requirements of this function. The return value of the function called "Handle" is used to handle LCD2004".

```

lcdhd = lcdInit(4, 20, 4, RS, EN, D4, D5, D6, D7, 0, 0, 0, 0); // initialize LCD and return
"handle" used to handle LCD
```

Details about `lcdInit()`:

```

int lcdInit (int rows, int cols, int bits, int rs, int strb,
            int d0, int d1, int d2, int d3, int d4, int d5, int d6, int d7);
```

This is the main initialization function and must be executed first before you use any other LCD functions.

**Rows** and **cols** are the rows and columns of the Display (e.g. 2, 16 or 4, 20). **Bits** is the number of how wide the number of bits is on the interface (4 or 8). The **rs** and **strb** represent the pin numbers of the Display's

RS pin and Strobe (E) pin. The parameters **d0** through **d7** are the pin numbers of the 8 data pins connected from the RPi to the display. Only the first 4 are used if you are running the display in 4-bit mode.

The return value is the ‘handle’ to be used for all subsequent calls to the lcd library when dealing with that LCD, or -1 to indicate a fault (usually incorrect parameter)

For more details about LCD Library, please refer to: <https://projects.drogon.net/raspberry-pi/wiringpi/lcd-library/>

In the next “while”, two subfunctions are called to display the RPi’s CPU Temperature and the SystemTime. First look at subfunction printCPUtemperature(). The CPU temperature data is stored in the “/sys/class/thermal/thermal\_zone0/temp” file. We need to read the contents of this file, which converts it to temperature value stored in variable CPU\_temp and uses lcdprintf() to display it on LCD.

```
void printCPUtemperature() { //subfunction used to print CPU temperature

    FILE *fp;
    char str_temp[15];
    float CPU_temp;
    // CPU temperature data is stored in this directory.
    fp=fopen("/sys/class/thermal/thermal_zone0/temp", "r");
    fgets(str_temp, 15, fp);      // read file temp
    CPU_temp = atof(str_temp)/1000.0; // convert to Celsius degrees
    printf("CPU's temperature : %.2f \n", CPU_temp);
    lcdPosition(lcdhd, 0, 2);     // set the LCD cursor position to (0, 2)
    lcdPrintf(lcdhd, "CPU:%.2fC", CPU_temp); // Display CPU temperature on LCD
    fclose(fp);
}
```

Details about lcdPosition() and lcdprintf():

**lcdPosition (int handle, int x, int y);**

Set the position of the cursor for subsequent text entry.

**lcdPutchar (int handle, uint8\_t data)**

**lcdPuts (int handle, char \*string)**

**lcdPrintf (int handle, char \*message, …)**

These output a single ASCII character, a string or a formatted string using the usual print formatting commands to display individual characters (it is how you are able to see characters on your computer monitor).

Next is subfunction printDataTime() used to display System Time. First, it gets the Standard Time and stores it into variable Rawtime, and then converts it to the Local Time and stores it into timeinfo, and finally displays the Time information on the LCD2004 Display.

```
void printDataTime() { //used to print system time

    time_t rawtime;
    struct tm *timeinfo;
    time(&rawtime); // get system time
    timeinfo = localtime(&rawtime); // convert to local time
    printf("%s \n", asctime(timeinfo));
    lcdPosition(lcdhd, 0, 3); // set the LCD cursor position to (0, 3)
    lcdPrintf(lcdhd, "Time:%d:%d:%d", timeinfo->tm_hour, timeinfo->tm_min, timeinfo->tm_sec);
```

```
//Display system time on LCD  
}
```

## Python Code 2.1 I2CLCD2004

If you did not [configure I2C and install Smbus](#), please complete the configuration and installation. If you did, please continue.

First, observe the project result, and then learn about the code in detail.

**If you have any concerns, please contact us via: [support@freenove.com](mailto:support@freenove.com)**

3. Use cd command to enter 2.1\_I2CLCD2004 directory of Python code.

```
cd ~/Freenove_Kit/Freenove_LCD_Module_for_Raspberry_Pi/Code/Python_Code/2.1_I2CLCD2004
```

4. Use Python command to execute Python code "I2CLCD2004.py".

```
python I2CLCD2004.py
```

After the program is executed, the LCD2004 Screen will display your RPi's CPU Temperature and System Time.



**NOTE:** After the program is executed, if you cannot see anything on the display or the display is not clear, try rotating the white knob on back of LCD2004 slowly, which adjusts the contrast, until the screen can display the Time and Temperature clearly.



The following is the program code:

```
1 from PCF8574 import PCF8574_GPIO
2 from Adafruit_LCD2004 import Adafruit_CharLCD
3
4 from time import sleep, strftime
```

```
5  from datetime import datetime
6
7  def get_cpu_temp():      # get CPU temperature and store it into file
8      "/sys/class/thermal/thermal_zone0/temp"
9      tmp = open('/sys/class/thermal/thermal_zone0/temp')
10     cpu = tmp.read()
11     tmp.close()
12     return '{:.2f}'.format(float(cpu)/1000) + ' C'
13
14 def get_time_now():      # get system time
15     return datetime.now().strftime('%H:%M:%S')
16
17 def loop():
18     mcp.output(3,1)      # turn on LCD backlight
19     lcd.begin(20,4)      # set number of LCD lines and columns
20     lcd.setCursor(0,0)    # set cursor position
21     lcd.message('FREENOVE')
22     lcd.setCursor(0,1)    # set cursor position
23     lcd.message('www. freenove. com')
24     while(True):
25         #lcd.clear()
26         lcd.setCursor(0,2)  # set cursor position
27         lcd.message('CPU: ' + get_cpu_temp()+'\n')# display CPU temperature
28         lcd.setCursor(0,3)  # set cursor position
29         lcd.message(get_time_now())  # display the time
30         sleep(1)
31
32 def destroy():
33     lcd.clear()
34
35 PCF8574_address = 0x27 # I2C address of the PCF8574 chip.
36 PCF8574A_address = 0x3F # I2C address of the PCF8574A chip.
37 # Create PCF8574 GPIO adapter.
38 try:
39     mcp = PCF8574_GPIO(PCF8574_address)
40 except:
41     try:
42         mcp = PCF8574_GPIO(PCF8574A_address)
43     except:
44         print('I2C Address Error !')
45         exit(1)
46 # Create LCD, passing in MCP GPIO adapter.
47 lcd = Adafruit_CharLCD(pin_rs=0, pin_e=2, pins_db=[4,5,6,7], GPIO=mcp)
```

```

49 if __name__ == '__main__':
50     print ('Program is starting ... ')
51     try:
52         loop()
53     except KeyboardInterrupt:
54         destroy()

```

Two modules are used in the code, PCF8574.py and Adafruit\_LCD2004.py. These two documents and the code files are stored in the same directory, and neither of them is dispensable. Please DO NOT DELETE THEM! PCF8574.py is used to provide I2C communication mode and operation method of some of the ports for the RPi and PCF8574 IC Chip. Adafruit module Adafruit\_LCD2004.py is used to provide some functional operation method for the LCD2004 Display.

In the code, first get the object used to operate the PCF8574's port, then get the object used to operate the LCD2004.

```

address = 0x27 # I2C address of the PCF8574 chip.
# Create PCF8574 GPIO adapter.
mcp = PCF8574_GPIO(address)
# Create LCD, passing in MCP GPIO adapter.
lcd = Adafruit_CharLCD(pin_rs=0, pin_e=2, pins_db=[4,5,6,7], GPIO=mcp)

```

According to the circuit connection, port 3 of PCF8574 is connected to the positive pole of the LCD2004 Display's backlight. Then in the loop () function, use of mcp.output (3,1) to turn the LCD2004 Display's backlight ON and then set the number of LCD lines and columns.

```

def loop():
    mcp.output(3,1)      # turn on the LCD backlight
    lcd.begin(20,4)      # set number of LCD lines and columns

```

In the next while loop, set the cursor position, and display the CPU temperature and time.

```

while(True):
    #lcd.clear()
    lcd.setCursor(0,2) # set cursor position
    lcd.message('CPU: ' + get_cpu_temp()+'\n')# display CPU temperature
    lcd.setCursor(0,3) # set cursor position
    lcd.message(get_time_now()) # display the time
    sleep(1)

```

CPU temperature is stored in file "/sys/class/thermal/thermal\_zone0/temp". Open the file and read content of the file, and then convert it to Celsius degrees and return. Subfunction used to get CPU temperature is shown below:

```

def get_cpu_temp():      # get CPU temperature and store it into file
    "/sys/class/thermal/thermal_zone0/temp"
    tmp = open('/sys/class/thermal/thermal_zone0/temp')
    cpu = tmp.read()
    tmp.close()
    return '{:.2f}'.format( float(cpu)/1000 ) + ' C'

```

Subfunction used to get time:

```

def get_time_now():      # get the time

```

```
return datetime.now().strftime('%H:%M:%S')
```

Details about PCF8574.py and Adafruit\_LCD2004.py:

### Module PCF8574

This module provides two classes **PCF8574\_I2C** and **PCF8574\_GPIO**.

Class **PCF8574\_I2C**: provides reading and writing method for PCF8574.

Class **PCF8574\_GPIO**: provides a standardized set of GPIO functions.

More information can be viewed through opening PCF8574.py.

Adafruit\_LCD2004 Module

### Module Adafruit\_LCD2004

This module provides the basic operation method of LCD2004, including class Adafruit\_CharLCD. Some member functions are described as follows:

**def begin(self, cols, lines)**: set the number of lines and columns of the screen.

**def clear(self)**: clear the screen

**def setCursor(self, col, row)**: set the cursor position

**def message(self, text)**: display contents

More information can be viewed through opening Adafruit\_CharLCD.py.



## What's Next?

THANK YOU for participating in this learning experience! If you have completed all of the projects successfully you can consider yourself a Raspberry Pi Master.

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](mailto: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 are interesting in processing, you can study the Processing.pdf in the unzipped folder.

If you want to learn more about Arduino, 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.