Welcome

Thank you for choosing Freenove products!

About Battery

First, read the document **About_Battery.pdf** in the unzipped folder.
If you did not download the zip file, please download it and unzip it via link below.
https://github.com/Freenove/Freenove_4WD_Smart_Car_Kit_for_Raspberry_Pi/archive/master.zip

Get Support and Offer Input

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

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

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

support@freenove.com

Safety and Precautions

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

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

About Freenove

Freenove provides open source electronic products and services worldwide.

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

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

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

http://www.freenove.com

Copyright

FREENOVE®

Free your innovation

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

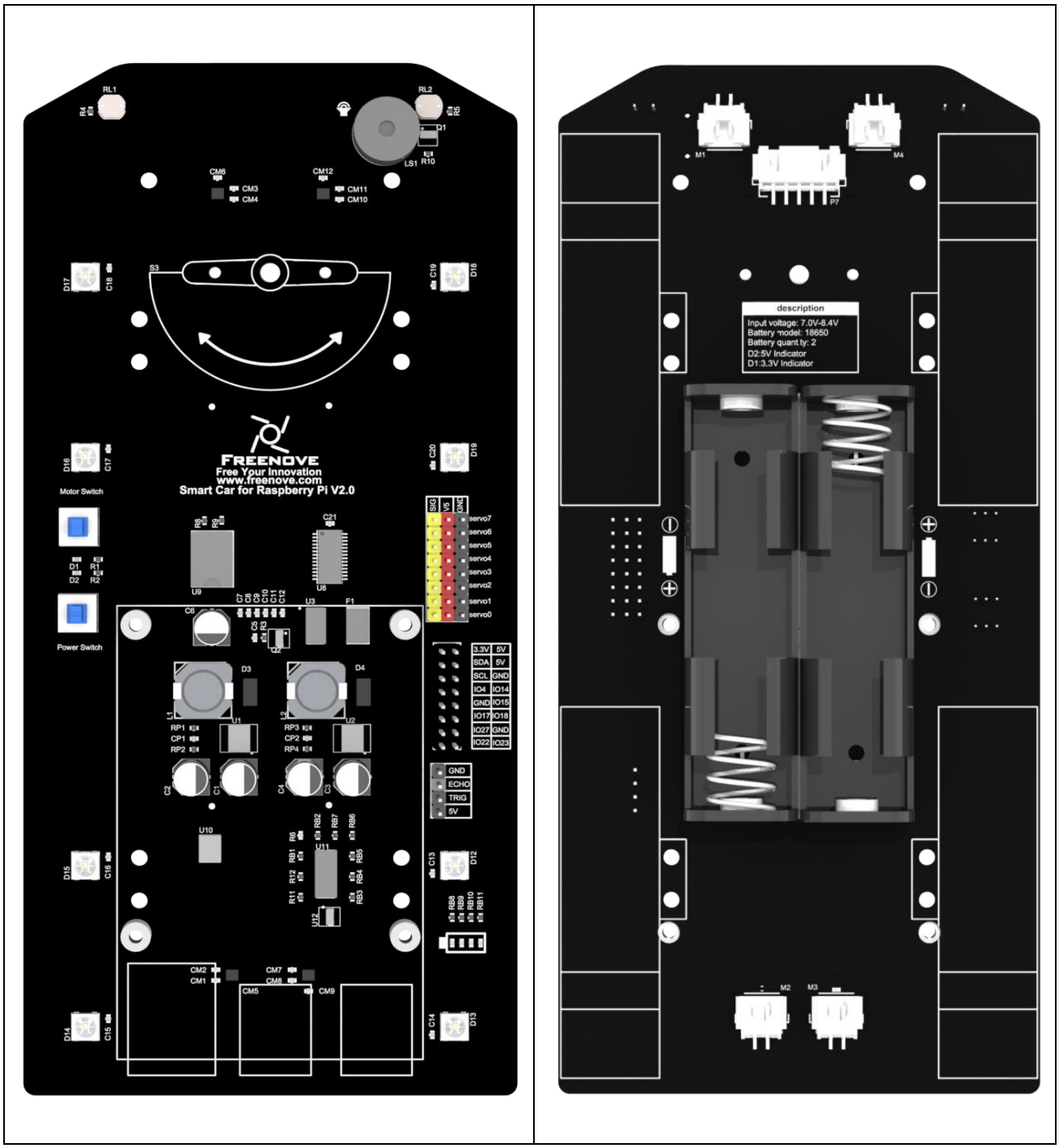
Need support? ✉ support.freenove.com

Contents

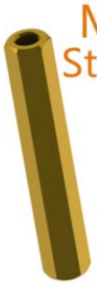
List

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

4WD Smart Car Board for Raspberry Pi

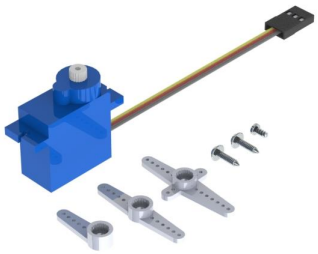
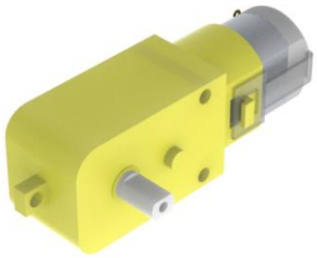


Machinery Parts

M1.4*4 self-tapping Screw  x12 Freenove	M2.5*4 Screw  x5 Freenove	M3*6 Screw  x5 Freenove	M2.5*8+6 Standoff  x5 Freenove	M3*30 Standoff  x3 Freenove
M2*10 Screw  x5 Freenove	M3*14 Screw  x4 Freenove	M2 Nut  x5 Freenove	M3 Nut  x4 Freenove	

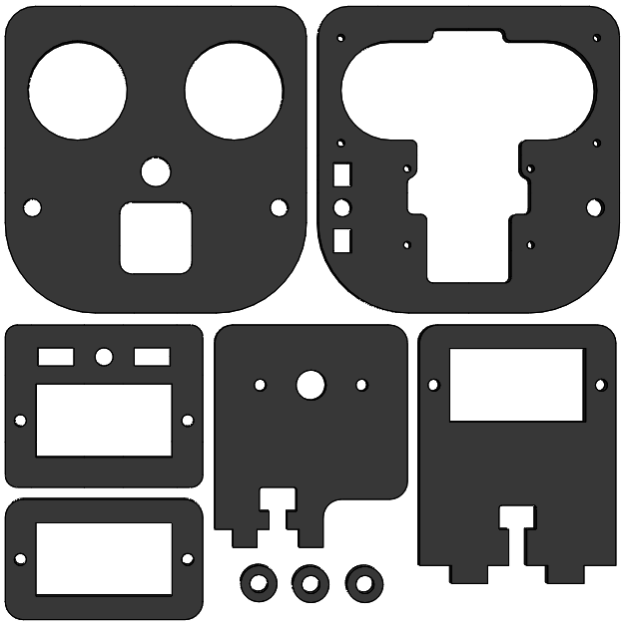
Note: You may receive M1.4*4 or M1.4*5. Can be used normally

Transmission Parts

Servo package x2 	Mecanum wheel x4 （2*2） 	
DC speed reduction motor x4 	Motor bracket package x4 	Coupling & screw for mecanum wheel x4 

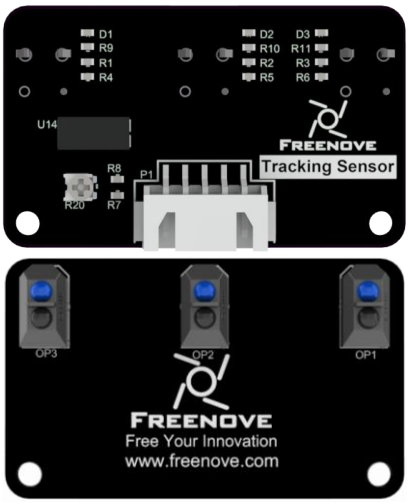
Acrylic Parts

For Pan Tilt

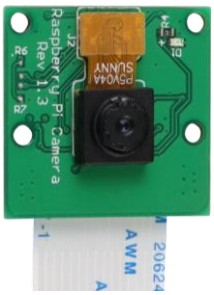


Electronic Parts

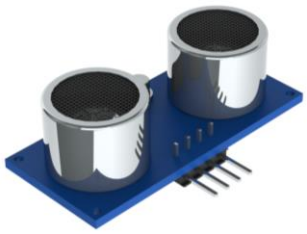
Line tracking module x1



Camera x1



HC-SR04 Ultrasonic Module x1



Connection board

Jumper Wire F/F(4) x1

XH-2.54-5Pin cable x1

Tools

Cross screwdriver (3mm) x1

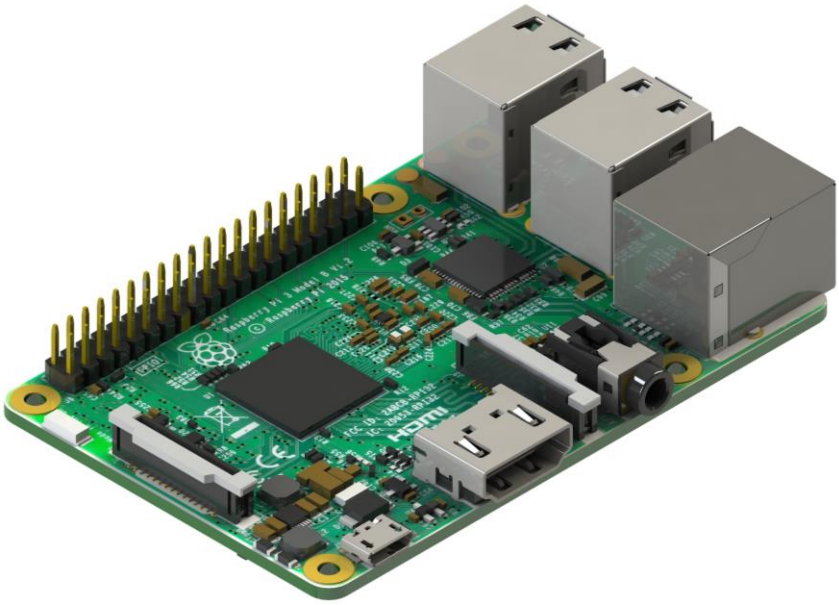
Black tape x1

Cable Tidy x25cm

Self-prepared Parts

Please refer to About_Battery.pdf in unzipped folder.

Raspberry Pi (Recommended model: Raspberry 4B / 3B+ / 3B) x1



Preface

Welcome to use Freenove 4WD Smart Car Kit for Raspberry Pi. Following this tutorial, you can make a very cool smart car with many functions.

This kit is based on Pi Raspberry, a popular control panel, so you can share and exchange your experience and design ideas with many enthusiasts all over the world. The parts in this kit include all electronic components, modules, and mechanical components required for making the smart car. And all of them are packaged individually. There are detailed assembly and commissioning instructions in this book.

And if you encounter any problems, please feel free to contact us for fast and free technical support.

support@freenove.com

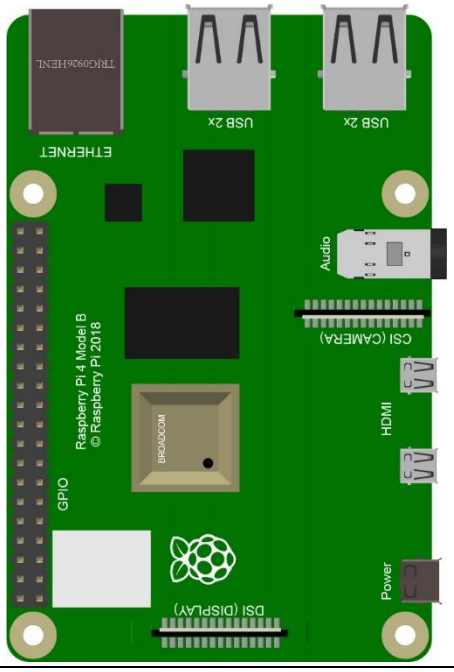
The contents in this book can help enthusiasts with little technical knowledge to make a smart car. If you are very interested in Raspberry Pi, and want to learn how to program and build the circuit, please visit our website www.freenove.com or contact us to buy the kits designed for beginners:
Freenove Basic\LCD1602\Super\Ultrasonic\RFID\Ultimate Starter Kit for Raspberry Pi

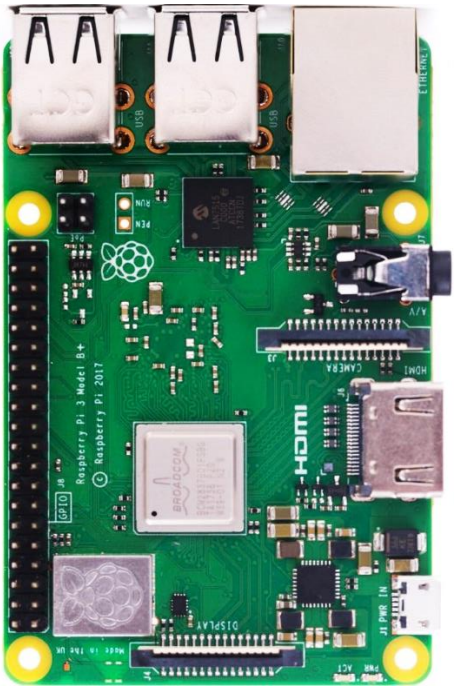
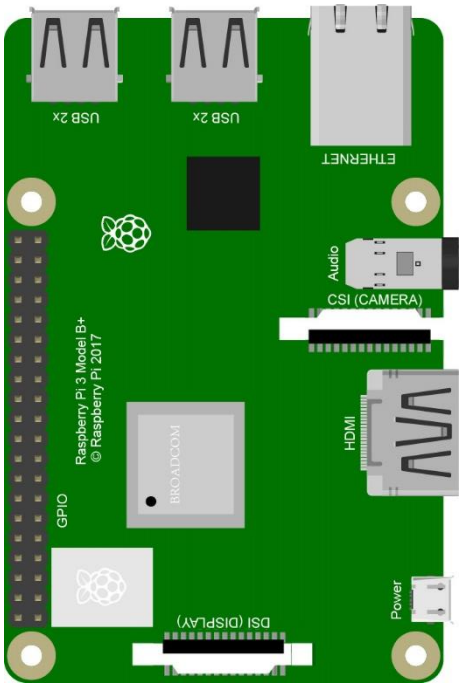
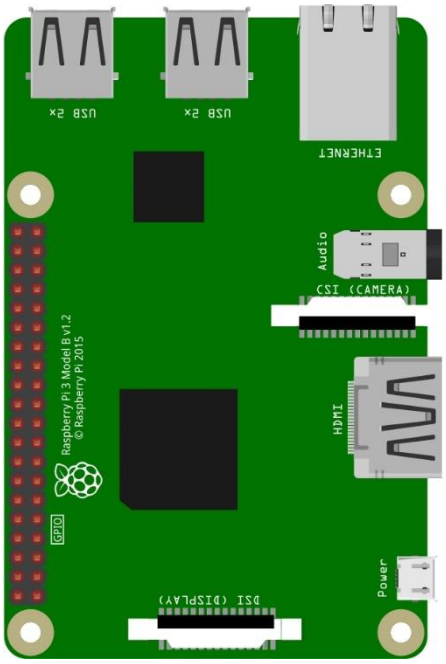
Raspberry Pi Introduction

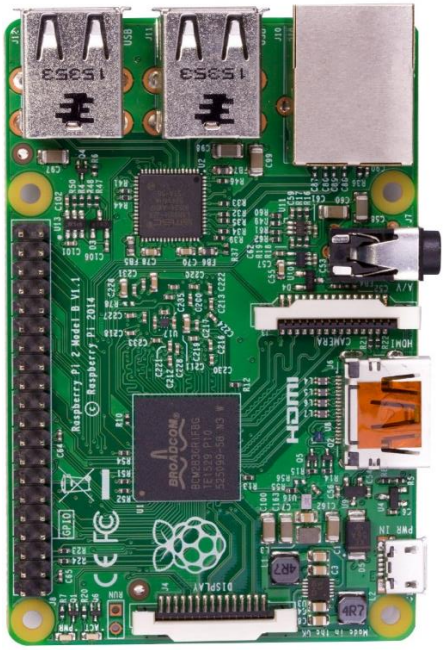
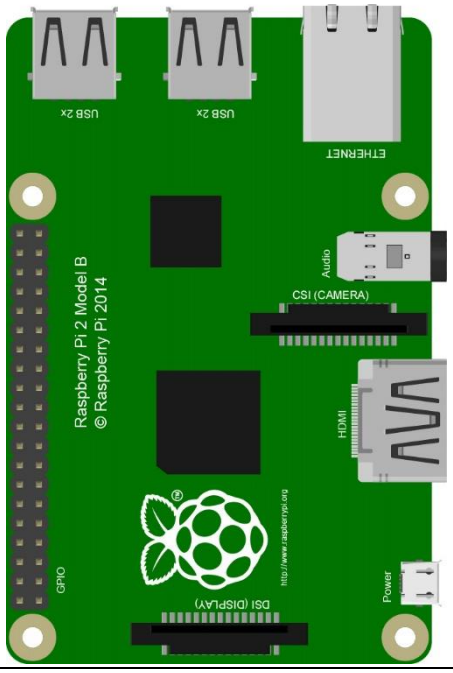
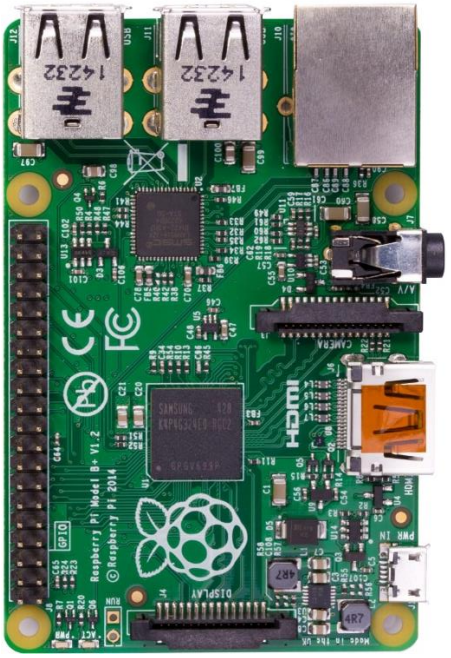
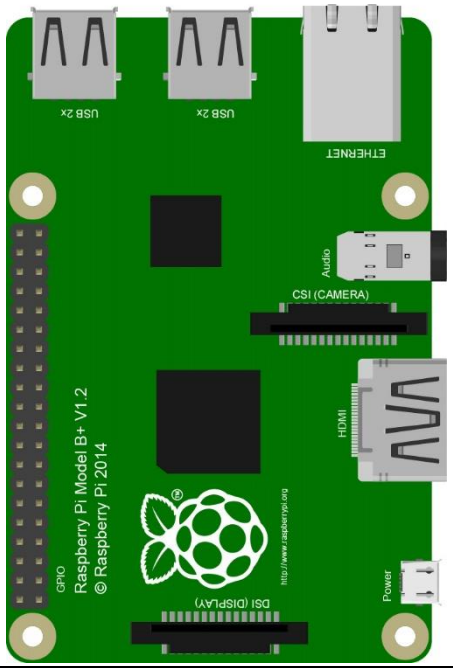
Raspberry Pi (called RPi, RPI, RasPi, the text these words will be used alternately later), a micro-computer with size of a card, quickly swept the world since its debut. It is widely used in desktop workstation, media center, smart home, robots, and even the servers, etc. It can do almost anything, which continues to attract fans to explore it. Raspberry Pi used to be running with Linux system and along with the release of windows 10 IoT. We can also run it with Windows. Raspberry Pi (with interfaces USB, network, HDMI, camera, audio, display and GPIO), as a microcomputer, can be running in command line mode and desktop system mode. Additionally, it is easy to operate just like Arduino, and you can even directly operate the GPIO of CPU.

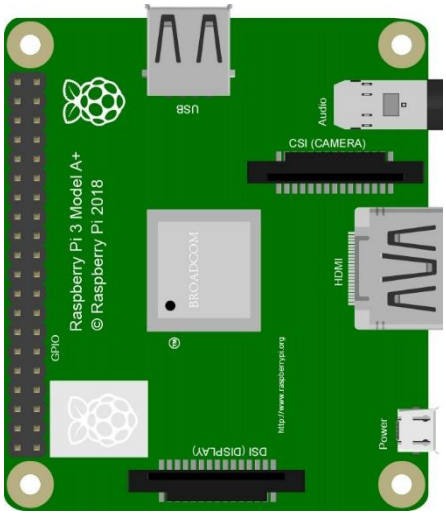
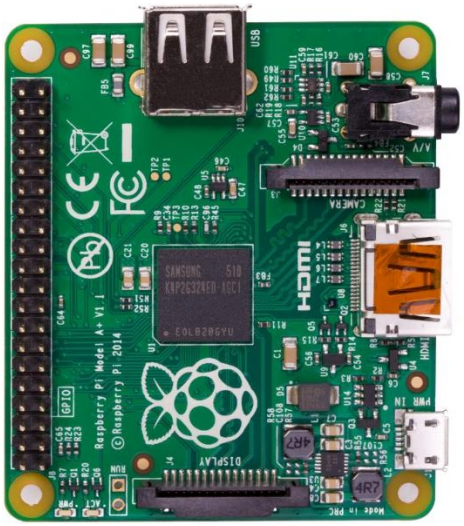
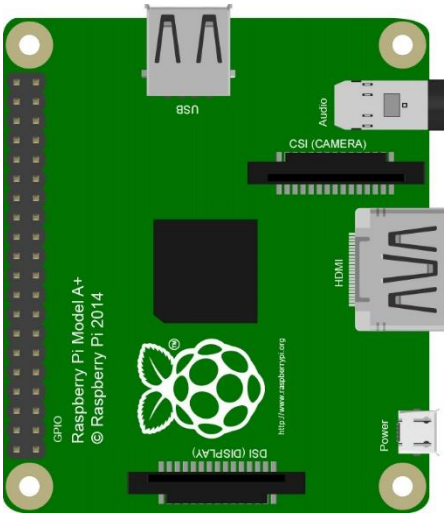
So far, Raspberry Pi has developed to the fourth generation. Changes in versions are accompanied by increase and upgrades in hardware. A type and B type, the first generation of products, have been stopped due to various reasons. Other versions are popular and active and the most important is that they are consistent in the order and number of pins, which makes the compatibility of peripheral devices greatly enhanced between different versions.

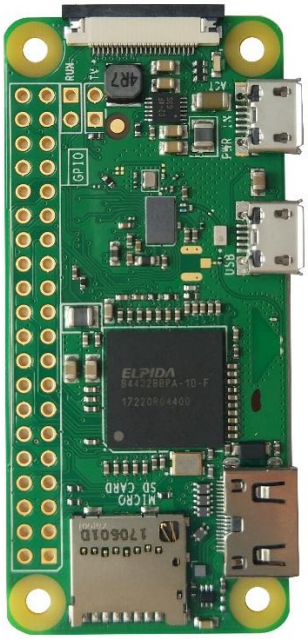
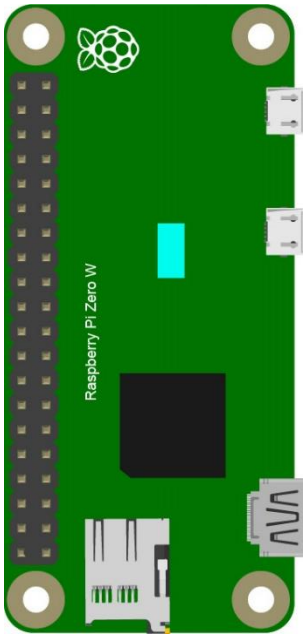
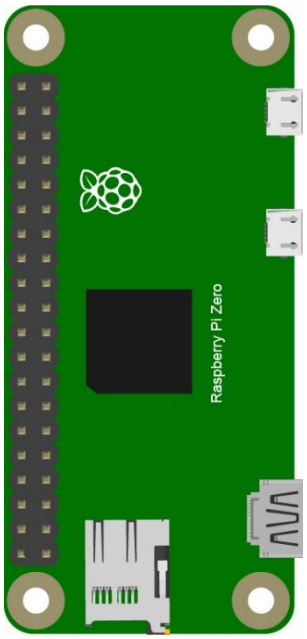
Below are the Raspberry Pi pictures and model pictures supported by this product.

Practicality picture of Raspberry Pi 4 Model B:	Model diagram of Raspberry Pi 4 Model B:
	

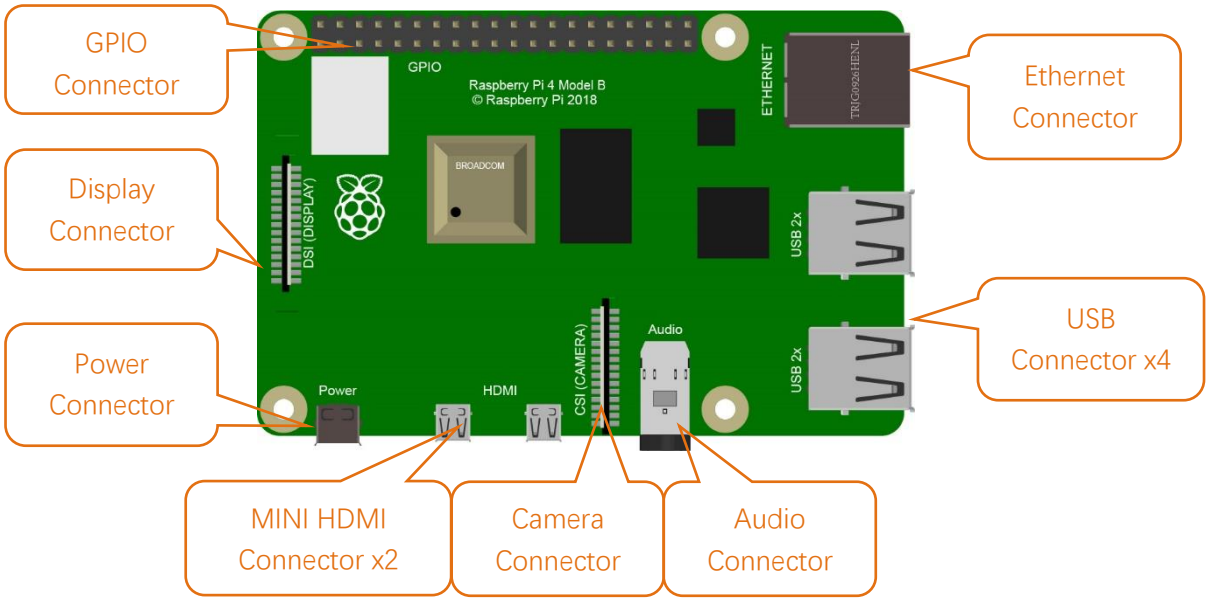
Practicality picture of Raspberry Pi 3 Model B+:	Model diagram of Raspberry Pi 3 Model B+:
	
Practicality picture of Raspberry Pi 3 Model B:	Model diagram of Raspberry Pi 3 Model B:
	

Practicality picture of Raspberry Pi 2 Model B: 	Model diagram of Raspberry Pi 2 Model B: 
Practicality picture of Raspberry Pi 1 Model B+: 	Model diagram of Raspberry Pi 1 Model B+: 

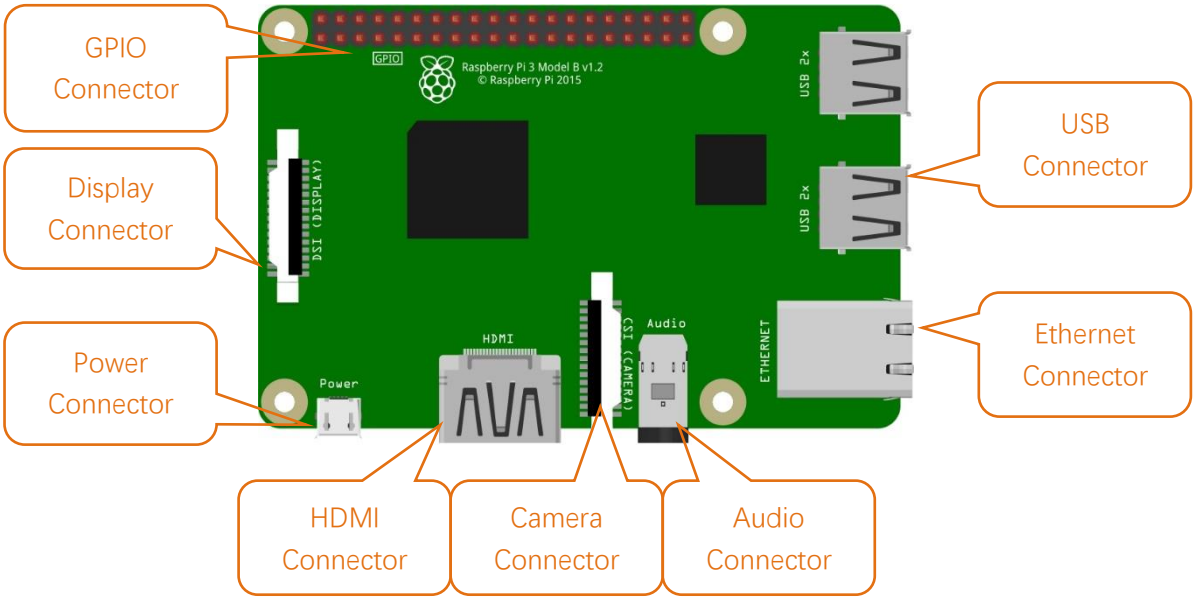
Practicality picture of Raspberry Pi 3 Model A+: 	Model diagram of Raspberry Pi 3 Model A+: 
Practicality picture of Raspberry Pi 1 Model A+: 	Model diagram of Raspberry Pi 1 Model A+: 

Practicality picture of Raspberry Pi Zero W:	Model diagram of Raspberry Pi Zero W:
	
Practicality picture of Raspberry Pi Zero:	Model diagram of Raspberry Pi Zero:
	

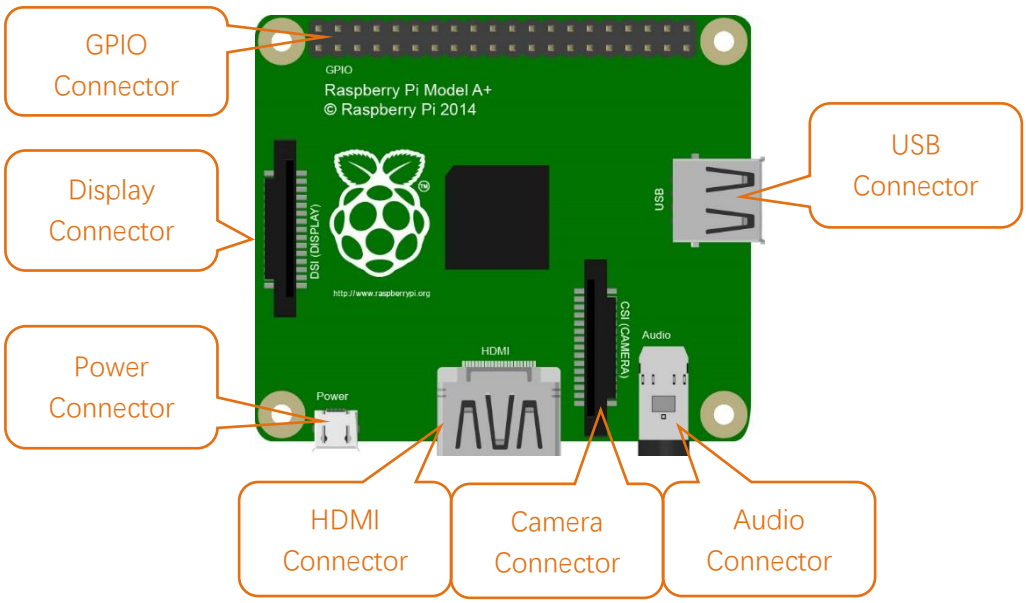
Hardware interface diagram of RPi 4B is shown below:



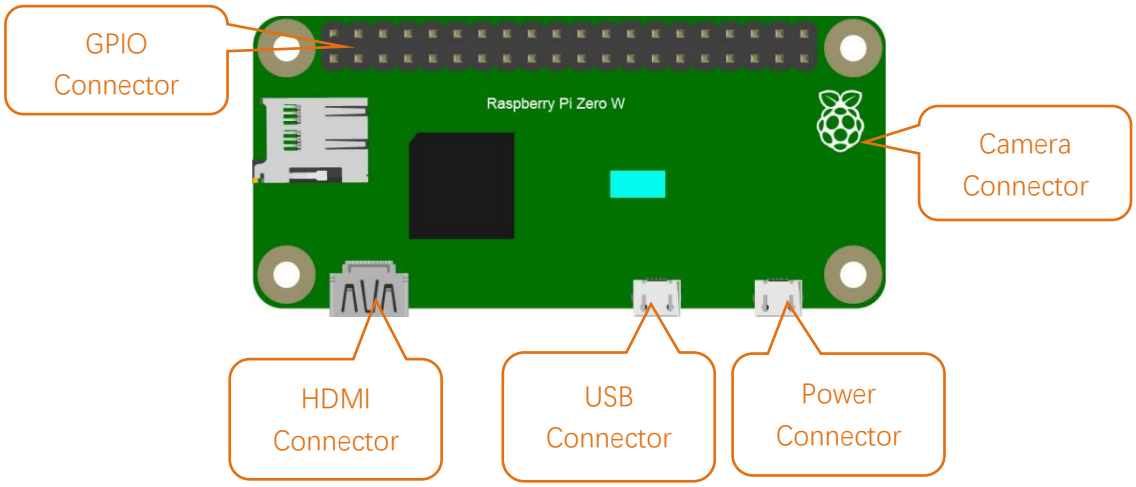
Hardware interface diagram of RPi 3B+/3B/2B/1B+ are shown below:



Hardware interface diagram of RPi 3A+/A+ is shown below:



Hardware interface diagram of RPi Zero/Zero W is shown below:

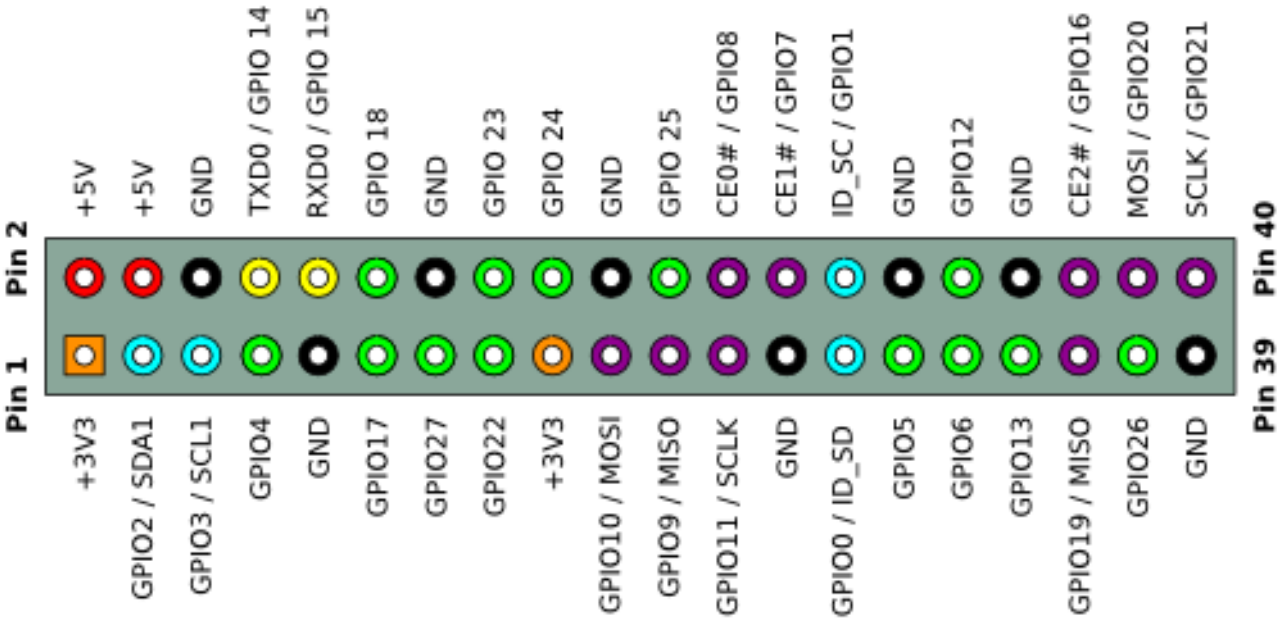


GPIO

GPIO: General purpose input/output. We will introduce the specific feature of the pins on the Raspberry Pi and what you can do with them. You can use them for all sorts of purposes. Most of them can be used as either inputs or outputs, depending on your program.
When programming the GPIO pins there are 3 different ways to refer to them: GPIO numbering, physical numbering, WiringPi GPIO Numbering.

BCM GPIO Numbering

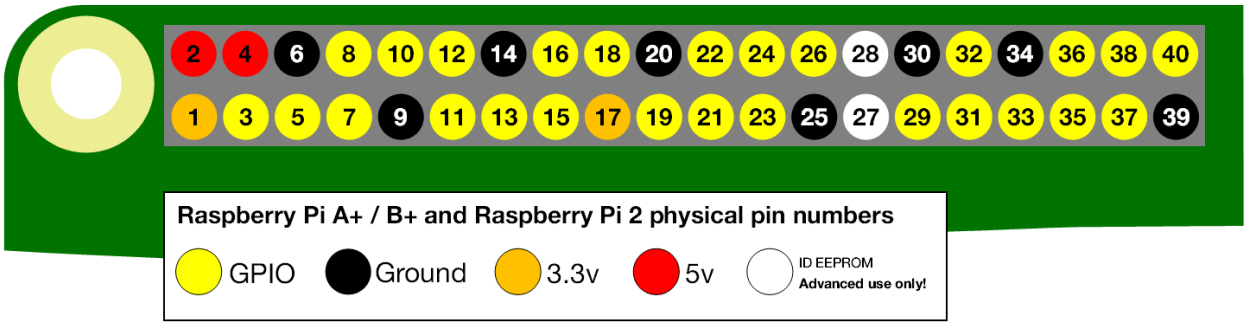
Raspberry Pi CPU use BCM2835/BCM2836/BCM2837of Broadcom. GPIO pin number is set by chip manufacturer. These are the GPIO pins as that computer recognizes. The numbers are unordered and don't make any sense to humans. You will need a printed reference or a reference board that fits over the pins.
Each pin is defined as below:



For more details about pin definition of GPIO, please refer to 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 mentioned two kinds of GPIO serial numbers, RPi GPIO serial number of the WiringPi was renumbered. Here we have three kinds of GPIO number mode: based on the number of BCM chip, based on the physical sequence number and based on wiringPi. The correspondence between these three GPIO numbers is shown below:

wiringPi Pin	BCM GPIO	Name	Header	Name	BCM GPIO	wiringPi Pin	
—	—	3.3v	1 | 2	5v	—	—	For Pi B
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	
wiringPi Pin	BCM GPIO	Name	Header	Name	BCM GPIO	wiringPi Pin	For A+, B+, 2B, 3B, 3B+, 4B, Zero

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

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

```
gpio readall
```

```
pi@raspberrypi:~ $ gpio readall
 +-----+-----+---------+------+---+---Pi 3---+---+------+---------+-----+-----+
 | 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  | 1 | ALT5 | TxD     | 15  | 14  |
 |     |     |      0v |      |   |  9 || 10 | 1 | ALT5 | 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 | ALT0 | 0 | 19 || 20 |   |      | 0v      |     |     |
 |   9 |  13 |    MISO | ALT0 | 0 | 21 || 22 | 0 | IN   | GPIO. 6 | 6   | 25  |
 |  11 |  14 |    SCLK | ALT0 | 0 | 23 || 24 | 1 | OUT  | CE0     | 10  | 8   |
 |     |     |      0v |      |   | 25 || 26 | 1 | OUT  | 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  |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+---Pi 3---+---+------+---------+-----+-----+
```

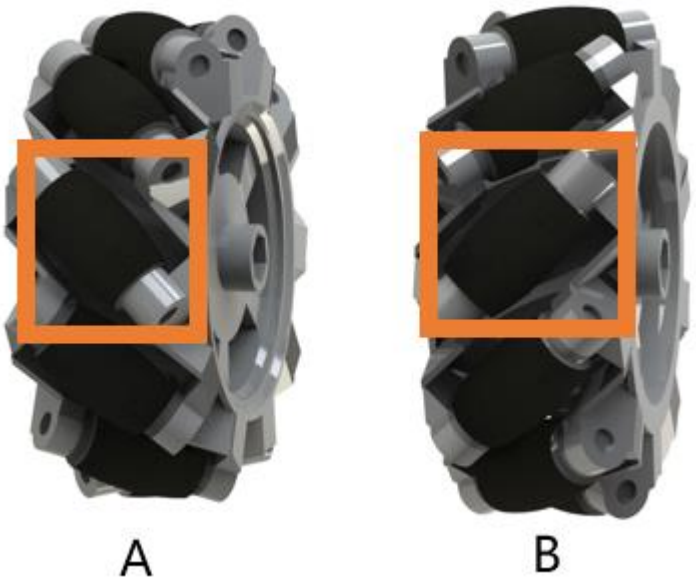
For more details about wiringPi, please refer to http://wiringpi.com/ .

Introduction to Mecanum wheel

Compared to regular wheels, mecanum wheels are a special type of wheel that can be considered as a composition of multiple small wheels. The rollers of mecanum wheels are arranged at a 45-degree angle, causing the direction of wheel speed to form a 45-degree angle with the horizontal axis. In other words, the movement of the mecanum wheels is not purely forward or backward like conventional wheels; instead, it has a component of motion at a 45-degree angle with respect to the ground, allowing the vehicle to move in diagonal directions as well as sideways, in addition to traditional forward and backward movements.

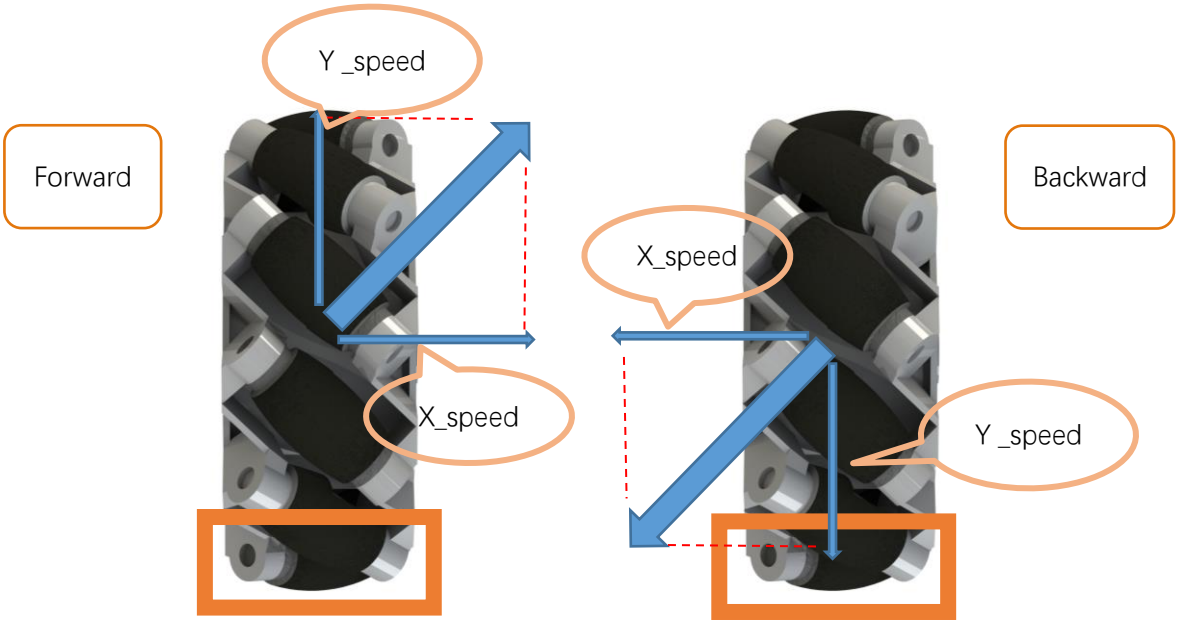
The capability to independently control each wheel allows us to break down vehicle's overall velocity into separate components for each wheel. This remarkable characteristic enables our vehicle to achieve true omnidirectional movement. Now, let's delve into its implementation.

First of all, it needs to be clear that the mecanum wheel has two different types, namely left-handed wheel (A) and right-handed wheel (B), as shown below.

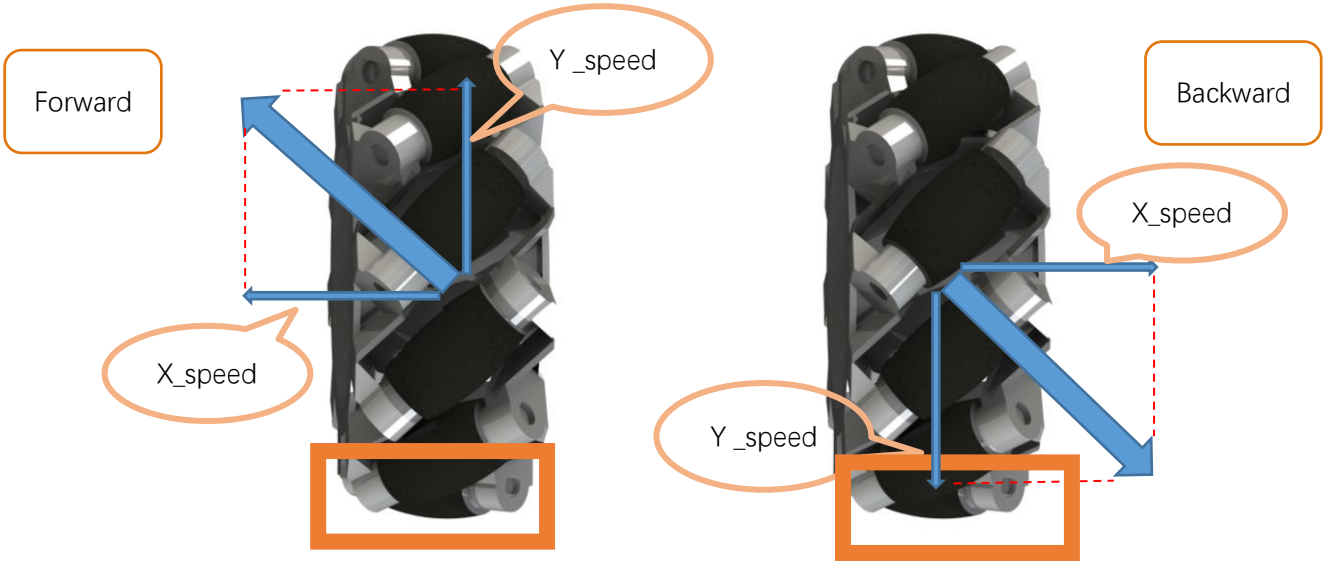


Let's analyze the A-type mecanum wheel in more detail. When this wheel rotates clockwise, its actual velocity behaves as shown in the diagram below. Unlike what we might expect based on the small wheels we see on top, the actual velocity aligns with the wheels at the bottom that are in contact with the ground.

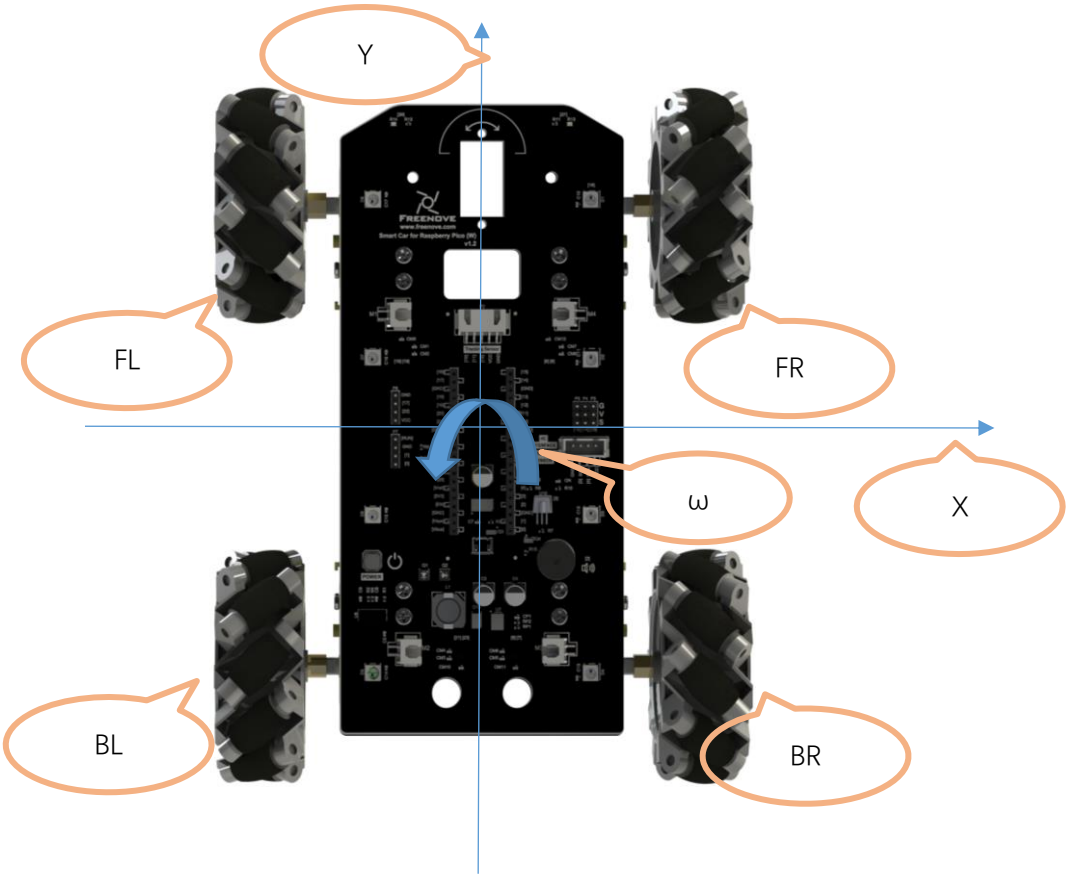
By considering the physics involved, we can decompose the velocity into components along the X-axis and Y-axis, allowing us to better comprehend its motion characteristics. Therefore, we can also consider that when the A-type mecanum wheel rotates forward, it generates a forward and rightward velocity. Conversely, when it rotates backward, it generates a backward and leftward velocity.



Next, analyze the speed of the B-type mecanum wheel in the same manner. It can be deducted that when the B-type mecanum wheel rotates forward, it generates both a forward and a leftward velocity. On the other hand, when it moves backward, it produces both a backward and a rightward velocity.



The following is the car chassis with A, B, A, B-type mecanum wheels. Now, try to analyze the relationship between the speed of each wheel and the motion of the car. Using a simple inverse kinematics calculation method, we can calculate the velocities of the four wheels when the mecanum chassis translates along the X-axis and along the Y-axis, and rotates around its geometric center. By simple addition, we can then compute the speed of the four wheels required for the composite motion of "translation + rotation" achieved by combining these three simple movements.



When the car moves along the Y axis, it can be seen that the speed of each wheel is equal to the translation speed, that is

$$\begin{cases} V(FR) = V(Y) \\ V(FL) = V(Y) \\ V(BL) = V(Y) \\ V(FR) = V(Y) \end{cases}$$

When the car moves along the X axis,

$$\begin{cases} V(FR) = -V(X) \\ V(FL) = +V(X) \\ V(BL) = -V(X) \\ V(FR) = +V(X) \end{cases}$$

When the cart moves counterclockwise with ω,

$$\begin{cases} V(FR) = +V(\omega) \\ V(FL) = -V(\omega) \\ V(BL) = -V(\omega) \\ V(FR) = +V(\omega) \end{cases}$$

Based on the above, the formula for solving the motion of the car chassis can be obtained as below.

$$\begin{cases} V(FR) = V(Y) - V(X) + V(\omega) \\ V(FL) = V(Y) + V(X) - V(\omega) \\ V(BL) = V(Y) - V(X) - V(\omega) \\ V(FR) = V(Y) + V(X) + V(\omega) \end{cases}$$

Note: When the car rotation is not considered, only the Y-axis velocity and X-axis velocity need to be retained, and the angular velocity ω can be removed.

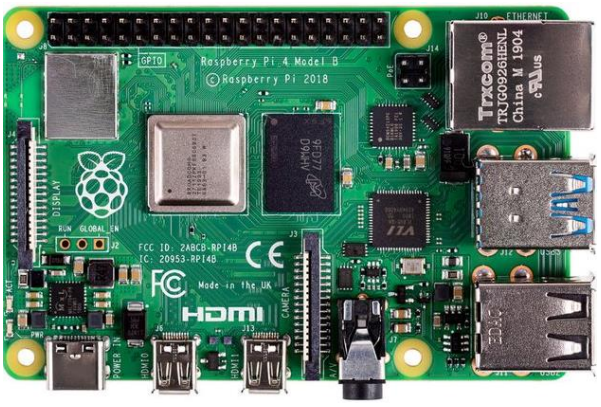
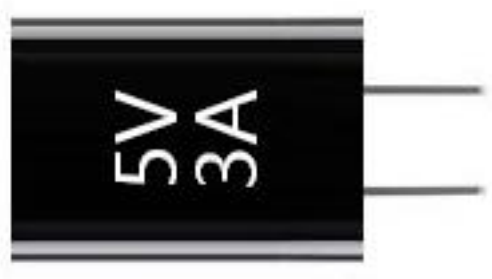
Chapter 0 Raspberry Pi Preparation

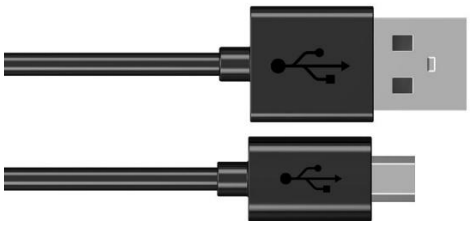
Install a System

Firstly, install a system for your RPi.

Component List

Required Components

Raspberry Pi 4B / 3B+/ 3B /3A+ (Recommended) 	5V/3A Power Adapter. Different versions of Raspberry Pi have different power requirements. 

Micro USB Cable x1	Micro SD Card (TF Card) x1, Card Reader x1
	

This robot also supports the following versions of the Raspberry Pi, but **additional accessories** need to be prepared by yourself.

Raspberry	Additional accessories
Raspberry Pi Zero W	Camera cable(>25cm) for zero w, 15 Pin 1.0mm Pitch to 22 Pin 0.5mm https://www.amazon.com/dp/B076Q595HJ/
Raspberry Pi Zero 1.3	wireless network adapter, Camera cable(>25cm) for zero w, 15 Pin 1.0mm Pitch to 22 Pin 0.5mm, OTG cable (USB Type micro B to USB Type A)
Raspberry Pi 2 Model B	wireless network adapter
Raspberry Pi 1 Model A+	wireless network adapter
Raspberry Pi 1 Model B+	wireless network adapter

Power requirement of different versions of Raspberry Pi is 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 a network cable used to connect it to wide area network.

All of these components are necessary. Among them, the power supply is required at least 5V/2.5A, because lack of power supply will lead to many abnormal problems, even damage to your RPi. So power supply with

5V/2.5A is highly recommend. SD Card Micro (recommended capacity 16GB or more) is a hard drive for RPi, which is used to store the system and personal files. In later projects, the components list with a RPi will contains these required components, using only RPi as a representative rather than presenting details.

Optional Components

Under normal circumstances, there are two ways to login to Raspberry Pi: using independent monitor, or remote desktop to share a monitor with your PC.

Required Accessories for Monitor

If you want to use independent monitor, mouse and keyboard, you also need the following accessories.

1. Display with HDMI interface
2. Mouse and Keyboard with USB interface

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

1. Micro-HDMI to HDMI converter wire.
2. Micro-USB to USB-A Receptacles converter wire (Micro USB OTG wire).
3. USB HUB.
4. USB transferring to Ethernet interface or USB Wi-Fi receiver.

For different Raspberry Pi, the optional items are slightly different. But all of their aims are to convert the special interface to standard interface of standard Raspberry Pi.

Item	Pi Zero	Pi Zero W	Pi A+	Pi 3A+	Pi B+/2B	Pi 3B/3B+/4B
Monitor	Yes	Yes	Yes	Yes	Yes	Yes
Mouse	Yes	Yes	Yes	Yes	Yes	Yes
Keyboard	Yes	Yes	Yes	Yes	Yes	Yes
Micro-HDMI to HDMI cable	Yes	Yes	No	No	No	No
Micro-USB to USB-A OTG cable	Yes	Yes	No	No	No	No
USB HUB	Yes	Yes	Yes	Yes	No	No
USB transferring to Ethernet interface	select one from two or select two from two	optional	select one from two or select two from two	optional	Internal Integration	Internal Integration
USB Wi-Fi receiver		Internal Integration		Internal Integration	optional	

Required Accessories for Remote Desktop

If you don't have an independent monitor, or you want to use a remote desktop, first you need to login to Raspberry Pi through SSH, then open the VNC or RDP service. So you need the following accessories.

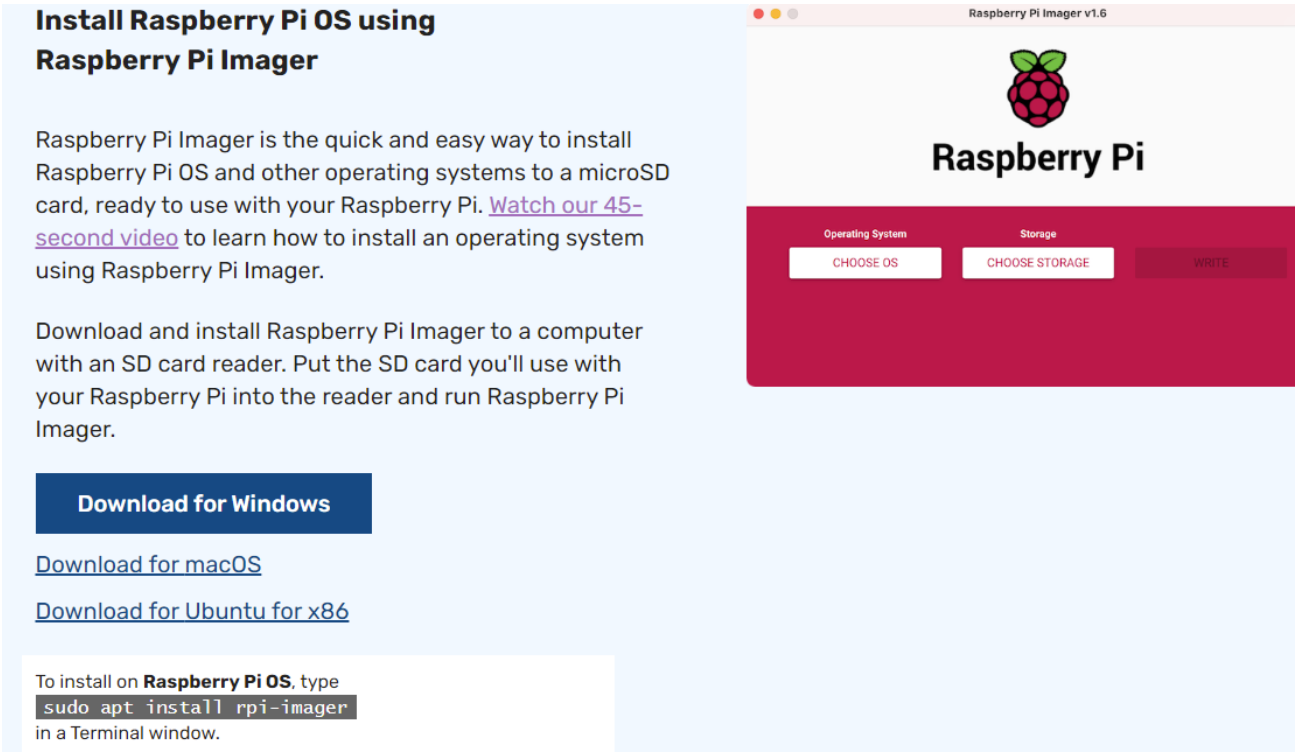
Item	Pi Zero	Pi Zero W	Pi A+	Pi 3A+	Pi B+/2B	Pi 3B/3B+/4B
Micro-USB to USB-A OTG cable	Yes	Yes	No	NO		
USB transferring to Ethernet interface	Yes	Yes	Yes			

Raspberry Pi OS

Install imager tool

Visit this website to install imager tool.

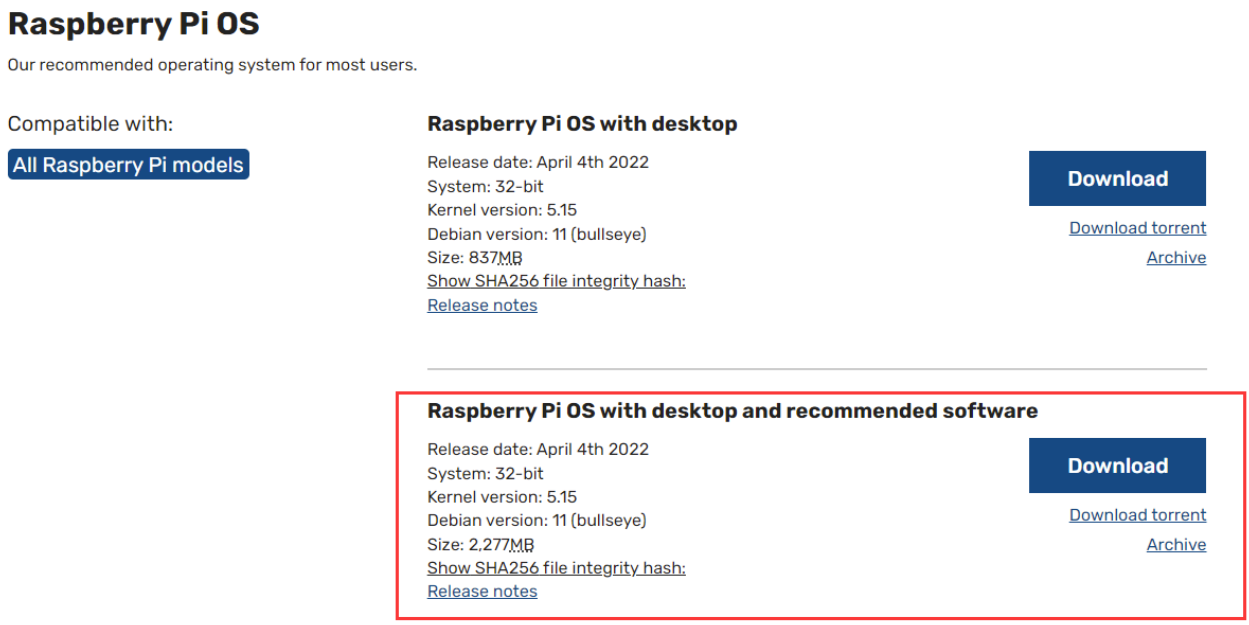
https://www.raspberrypi.com/software/



Download OS file

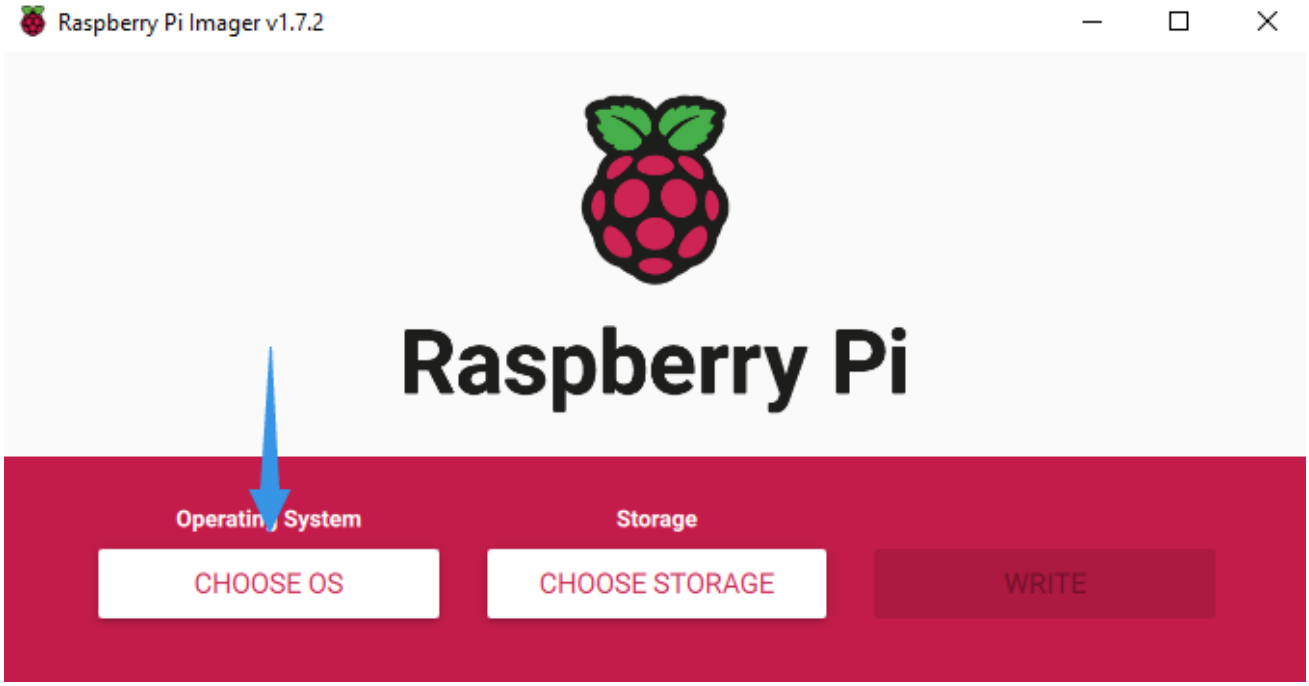
Visit following website to download the OS file.

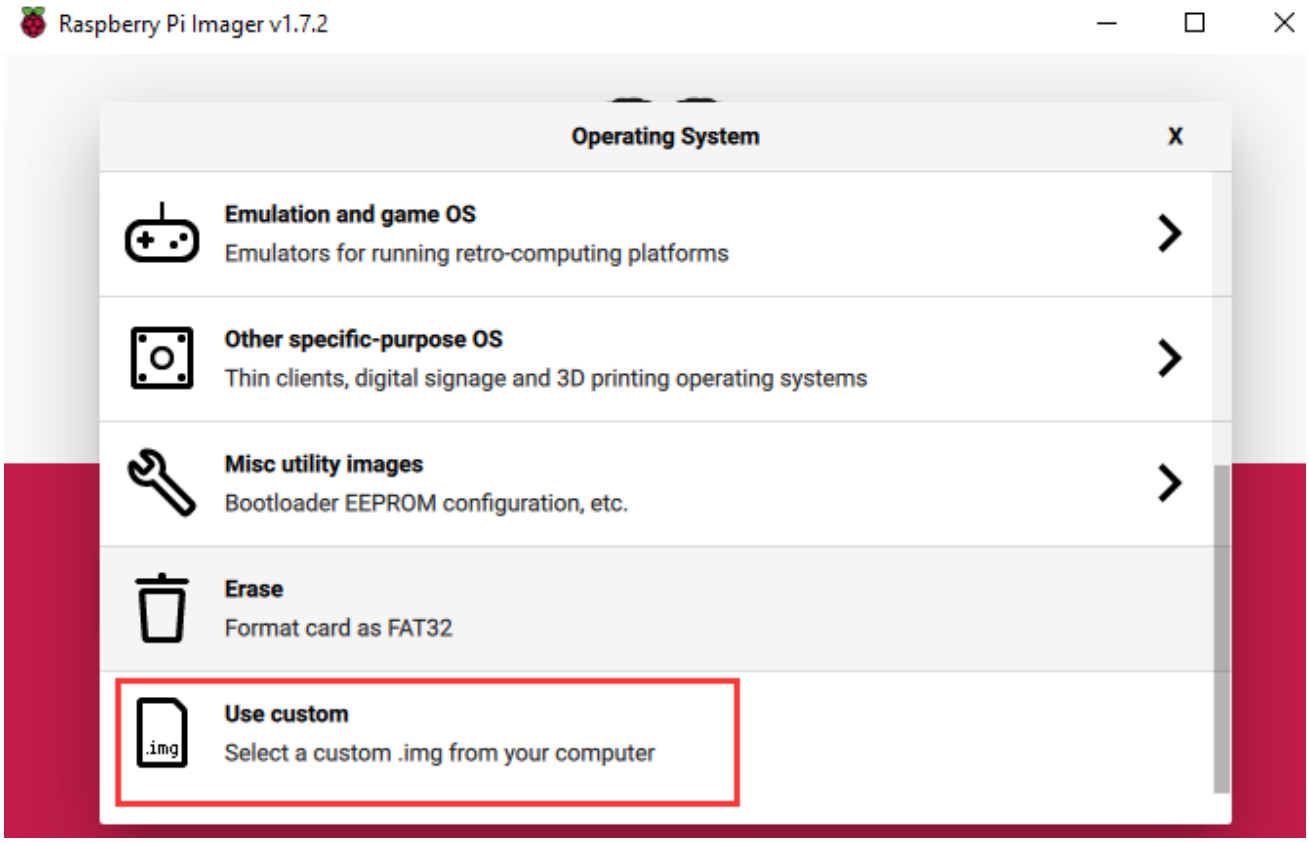
https://www.raspberrypi.com/software/operating-systems/



Write System to Micro SD Card

First, insert your Micro **SD card** into **card reader** and connect it to USB port of **PC**. Then open imager tool. Choose system that you just downloaded in Use custom.





Choose the SD card.

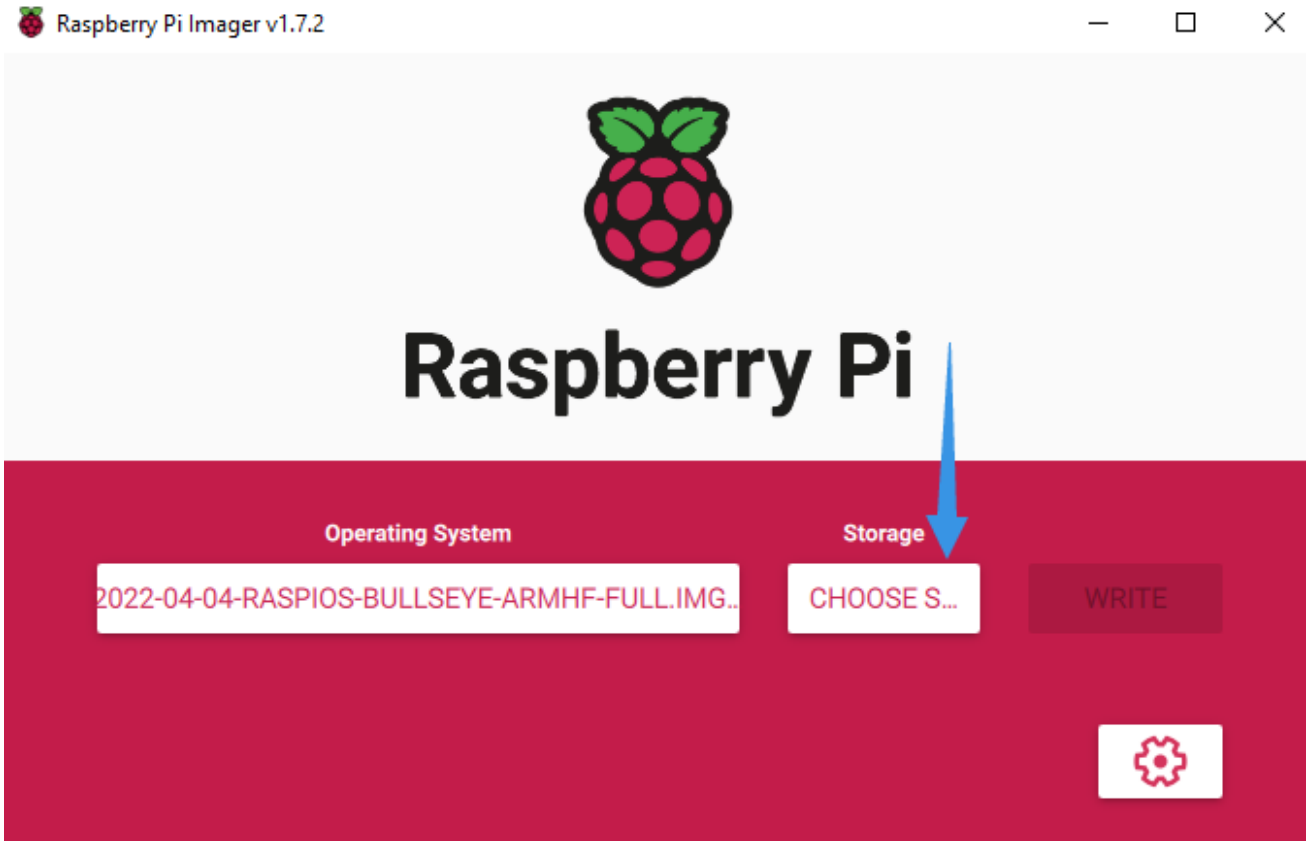
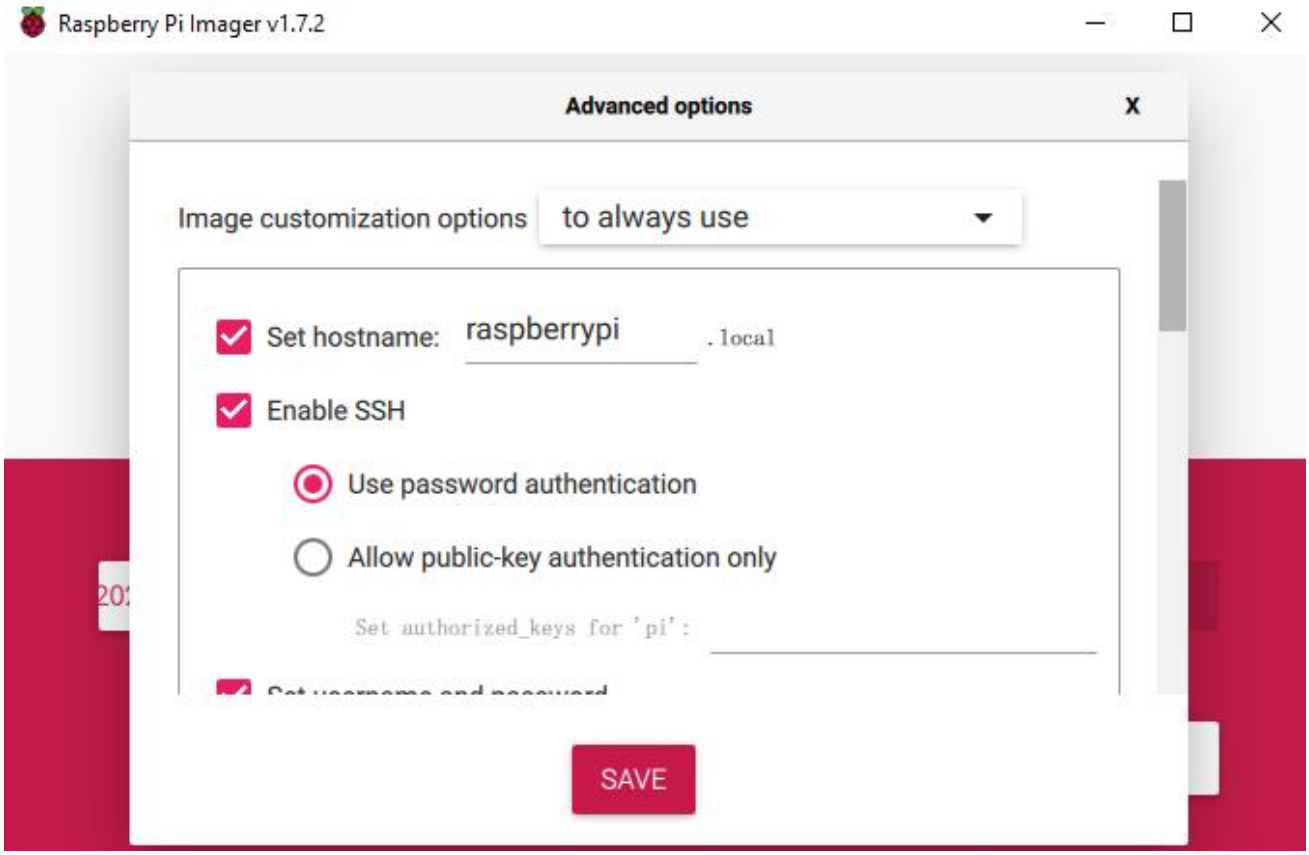
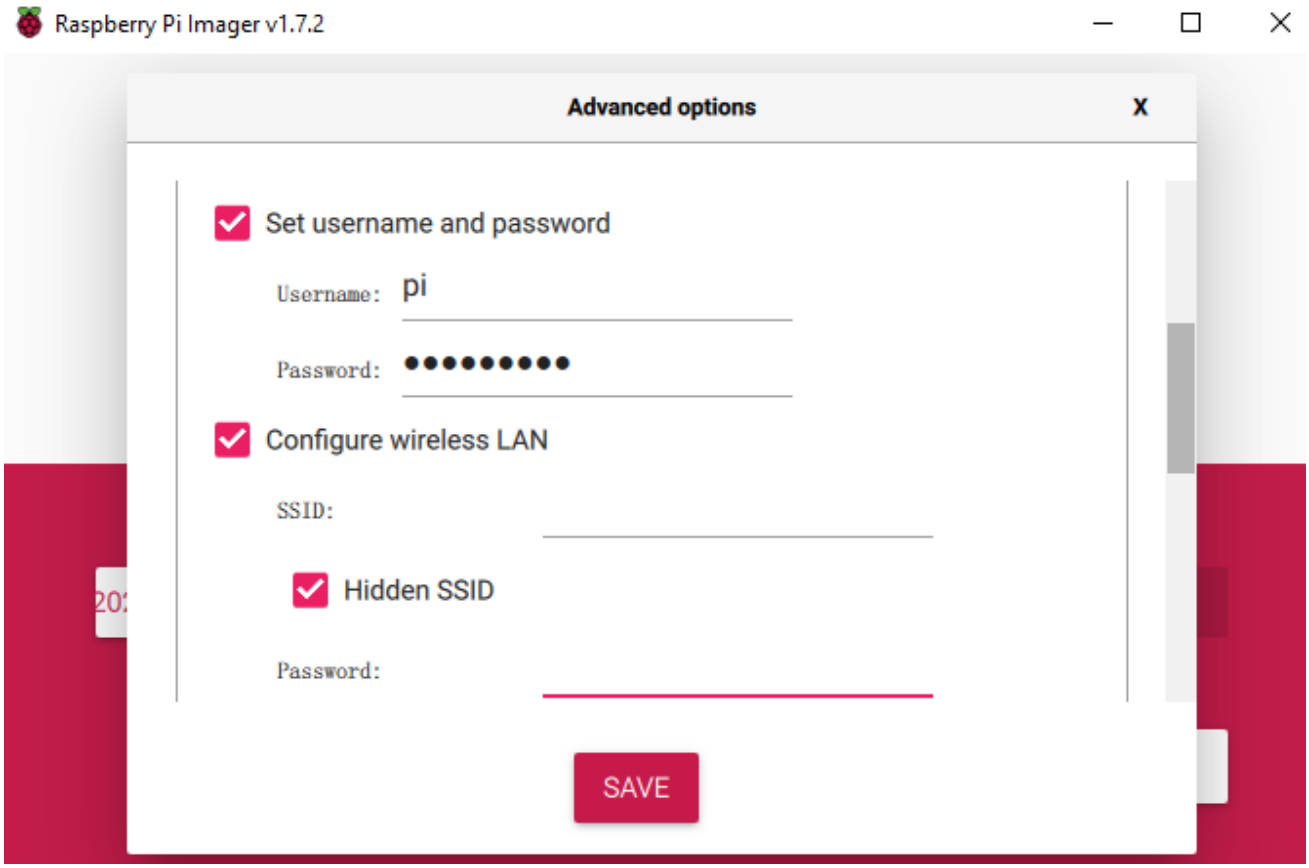


Image option.

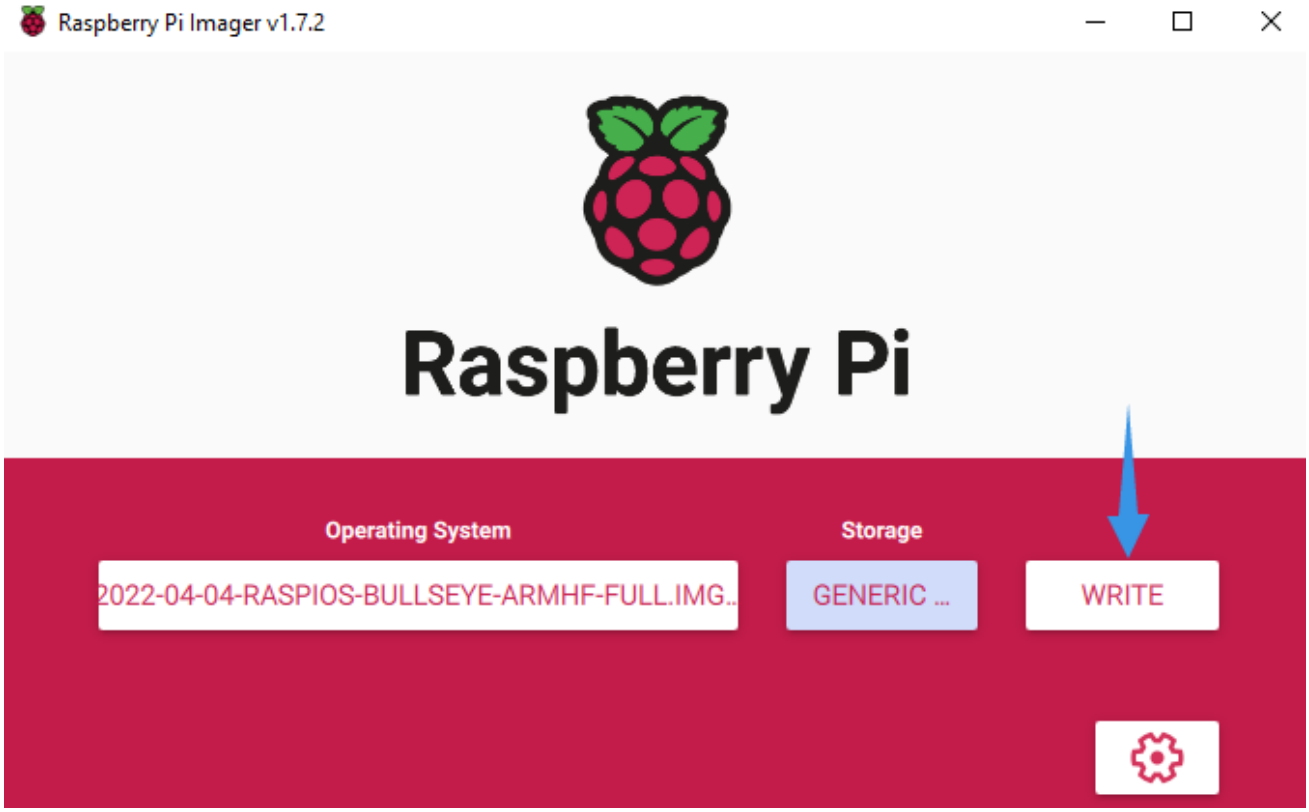
Enable SSH.



Configure WiFi and location. Here we set username as **pi**, password as **raspberry**



Finally WRITE.

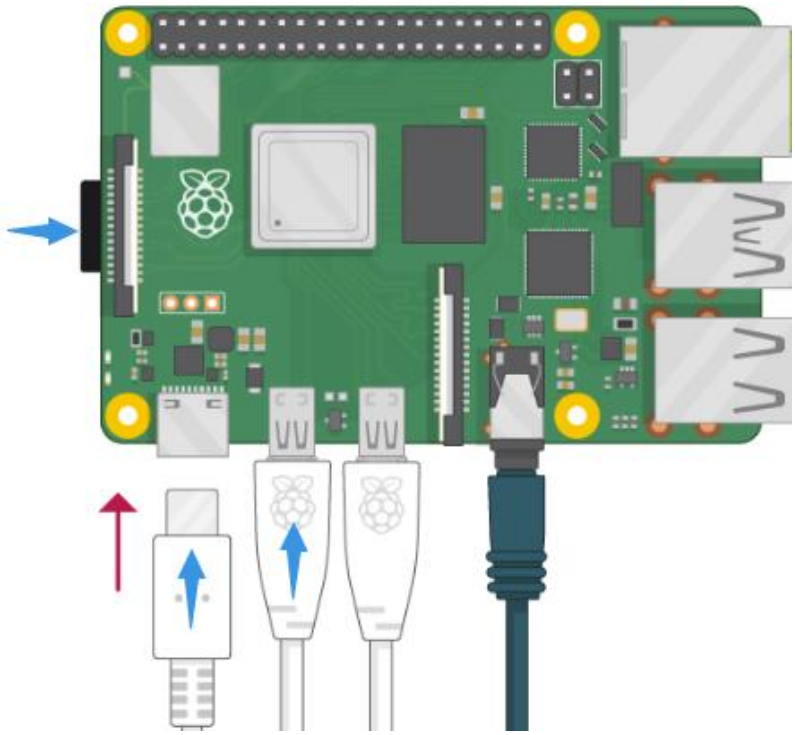


Start Raspberry Pi

If you don't have a spare monitor, please skip to next section.

If you have a spare monitor, please follow the steps in this section.

After the system is written successfully, put SD card into the SD card slot of RPi. Then connect your RPi to monitor through HDMI cable, attach your mouse and keyboard through the USB ports,



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.

You can connect WiFi on the right corner if WiFi is connected successfully.

Now you can skip to VNC Viewer.

Remote desktop & VNC

After you log in Raspberry Pi, please use VNC Viewer to connect Raspberry Pi for this robot. Other remote ways may not support GUI. **If you have logged in Raspberry Pi please skip to VNC Viewer.**

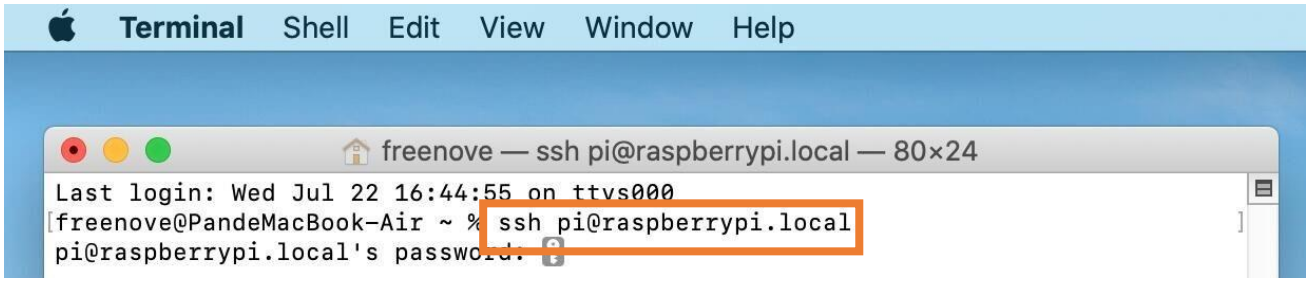
If you don't have a spare monitor, mouse and keyboard for your RPi, you can use a remote desktop to share a display, keyboard, and mouse with your PC. Below is how to use:
MAC OS remote desktop and Windows OS remote desktop.

MAC OS Remote Desktop

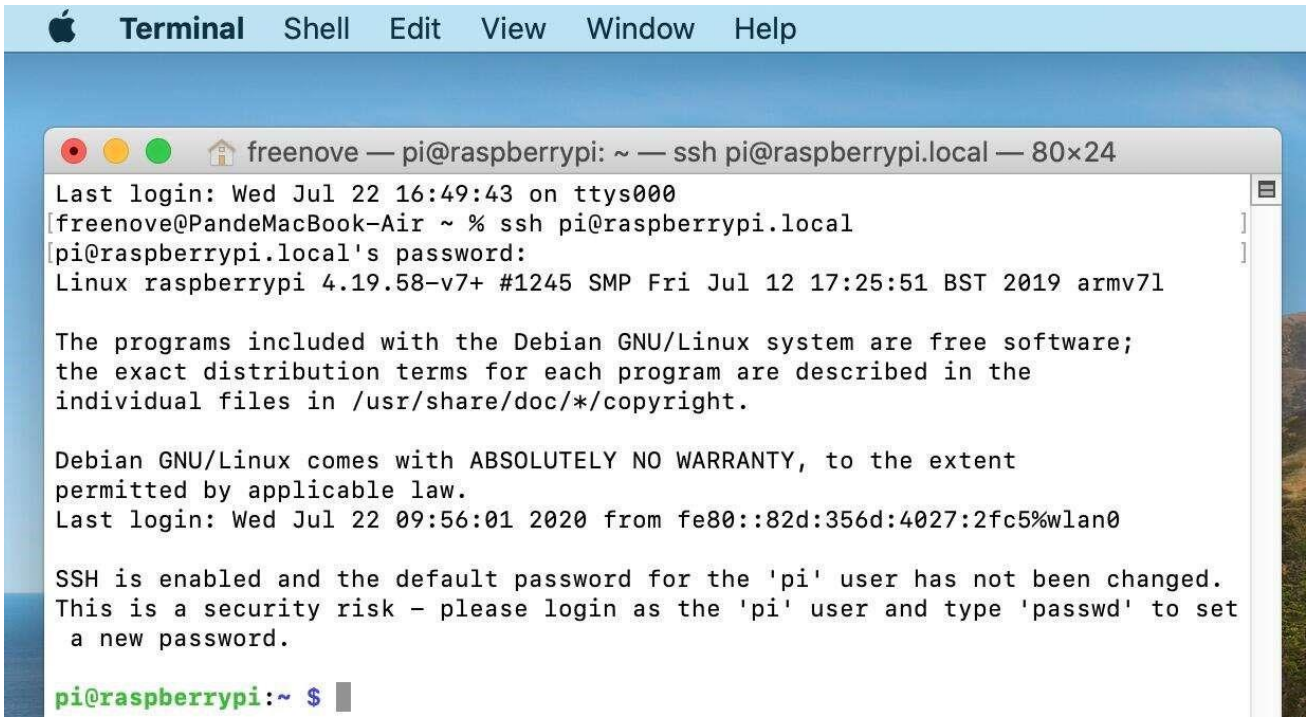
Open the terminal and type following command. **If this command doesn't work, please move to next page.**

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

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



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



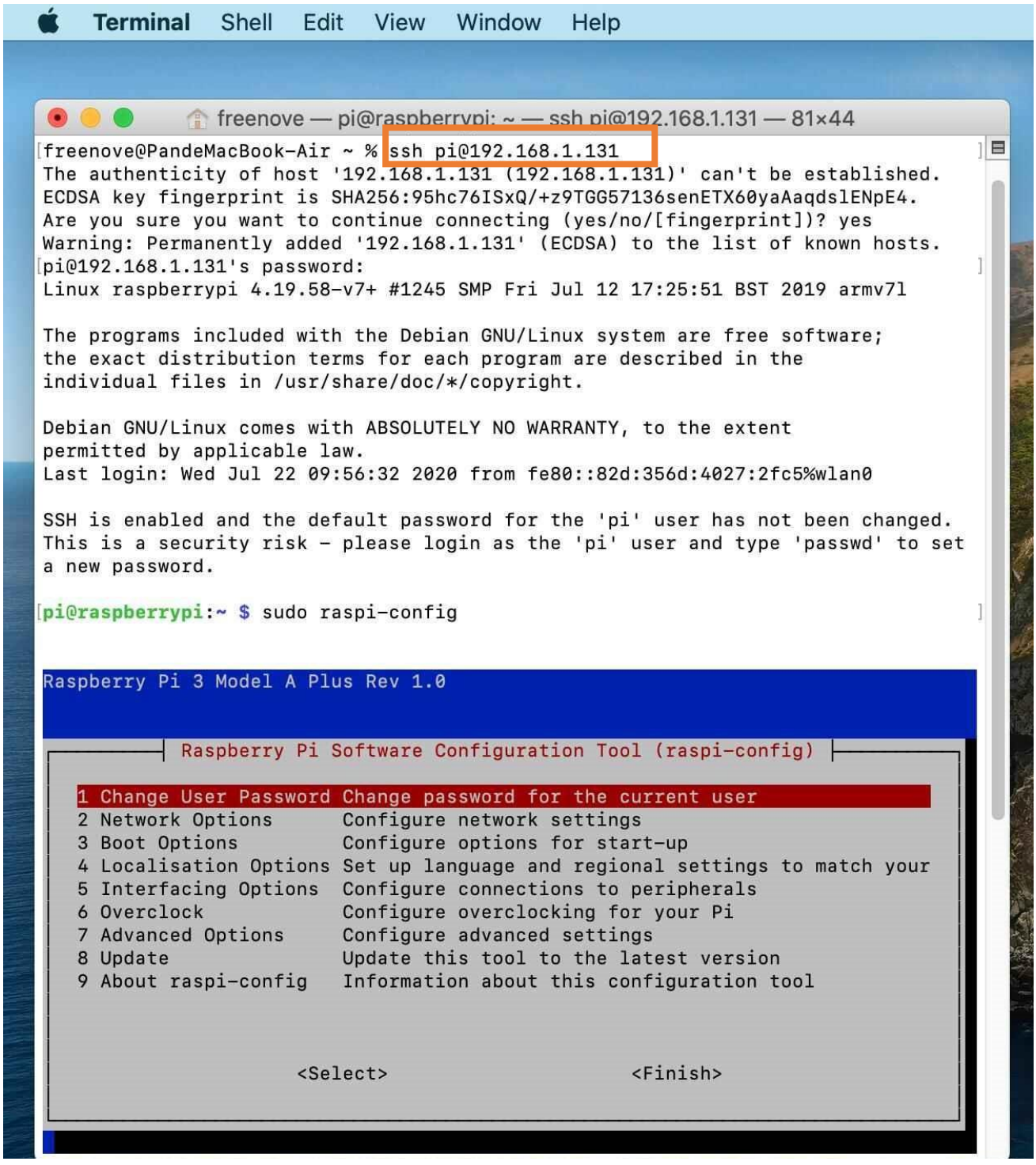
You can also use the IP address to log in Pi.

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.



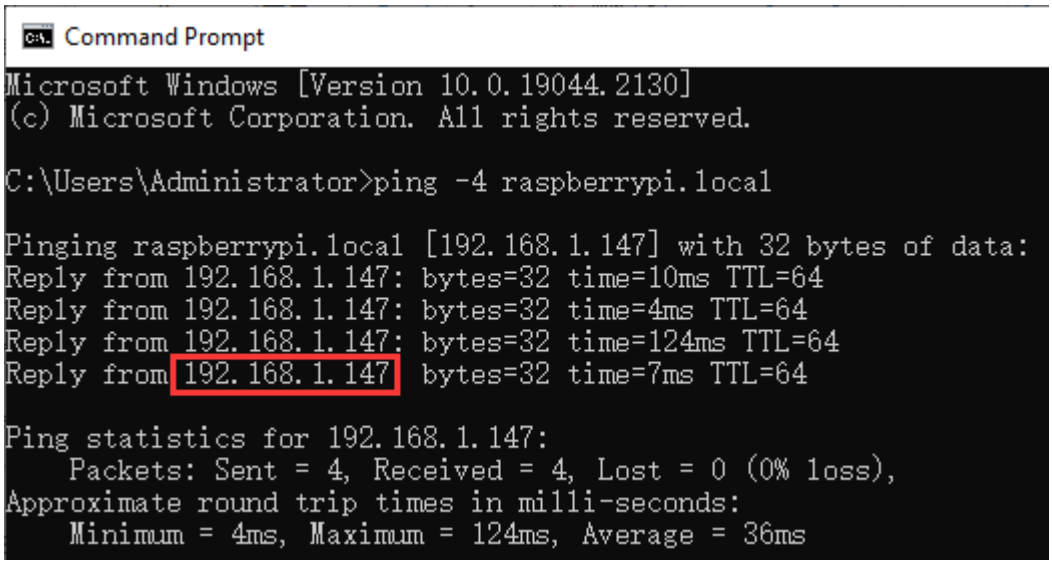
Then you can skip to VNC Viewer.

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:

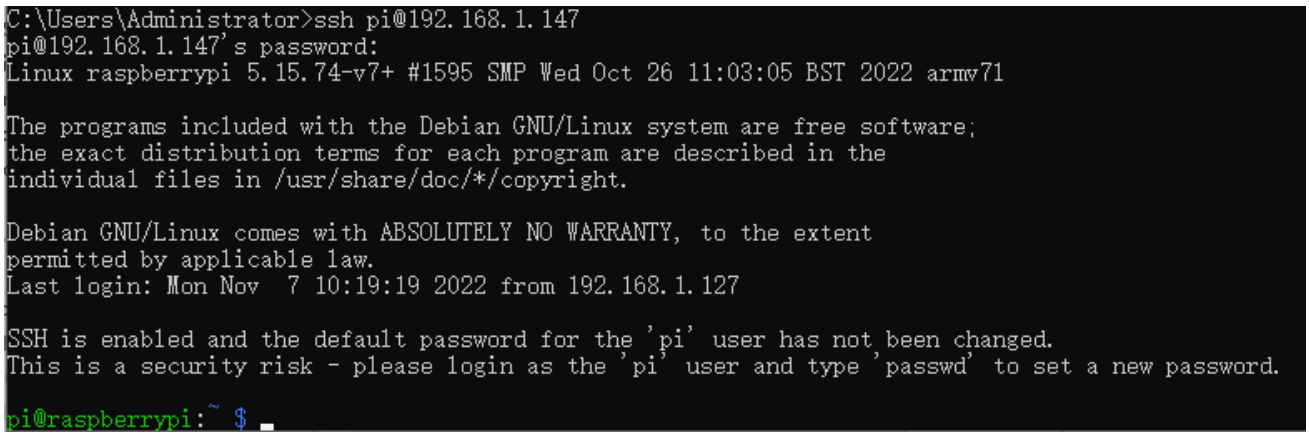
```
ping -4 raspberrypi.local
```



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".**

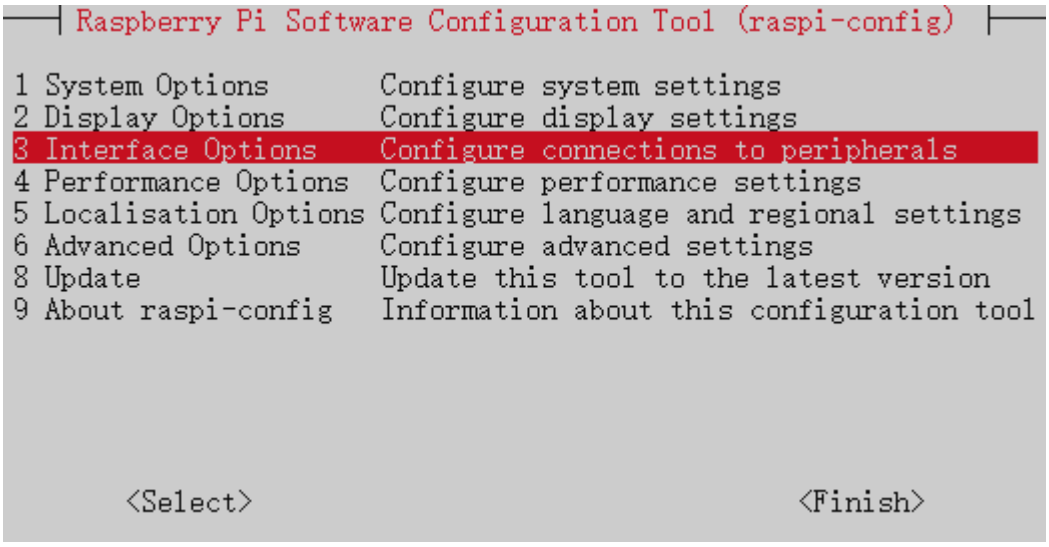
```
ssh pi@192.168.1.147
```

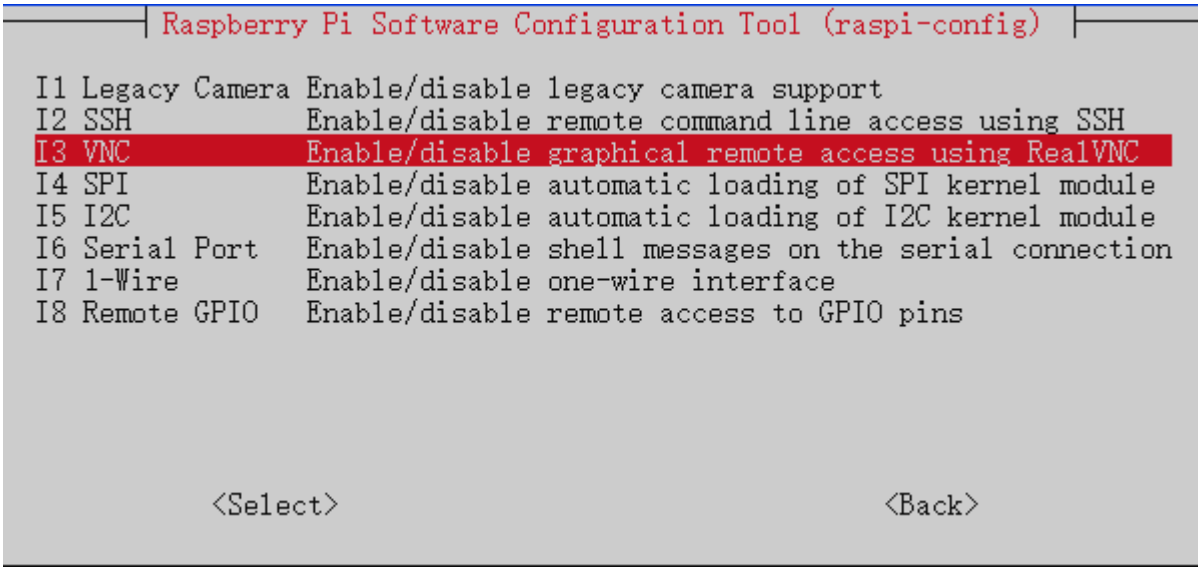


VNC Viewer & VNC

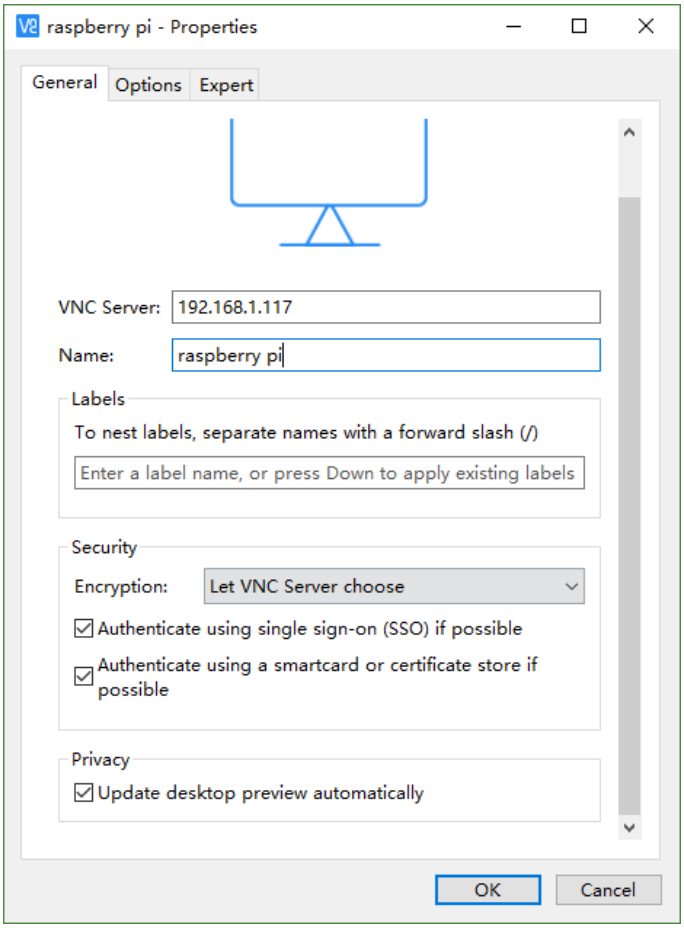
Type the following command. And select Interfacing **Options→ VNC →Yes→OK→Finish**. Here Raspberry Pi may need to be restarted, and choose ok. Then open VNC interface.

```
sudo raspi-config
```

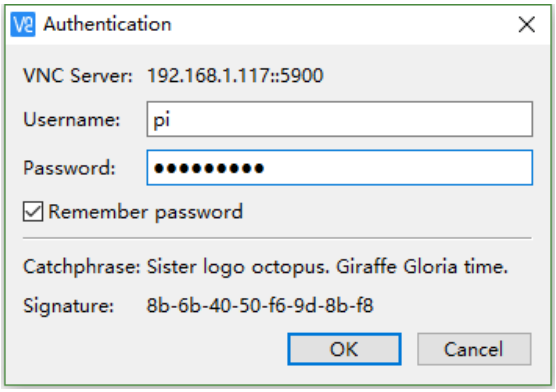




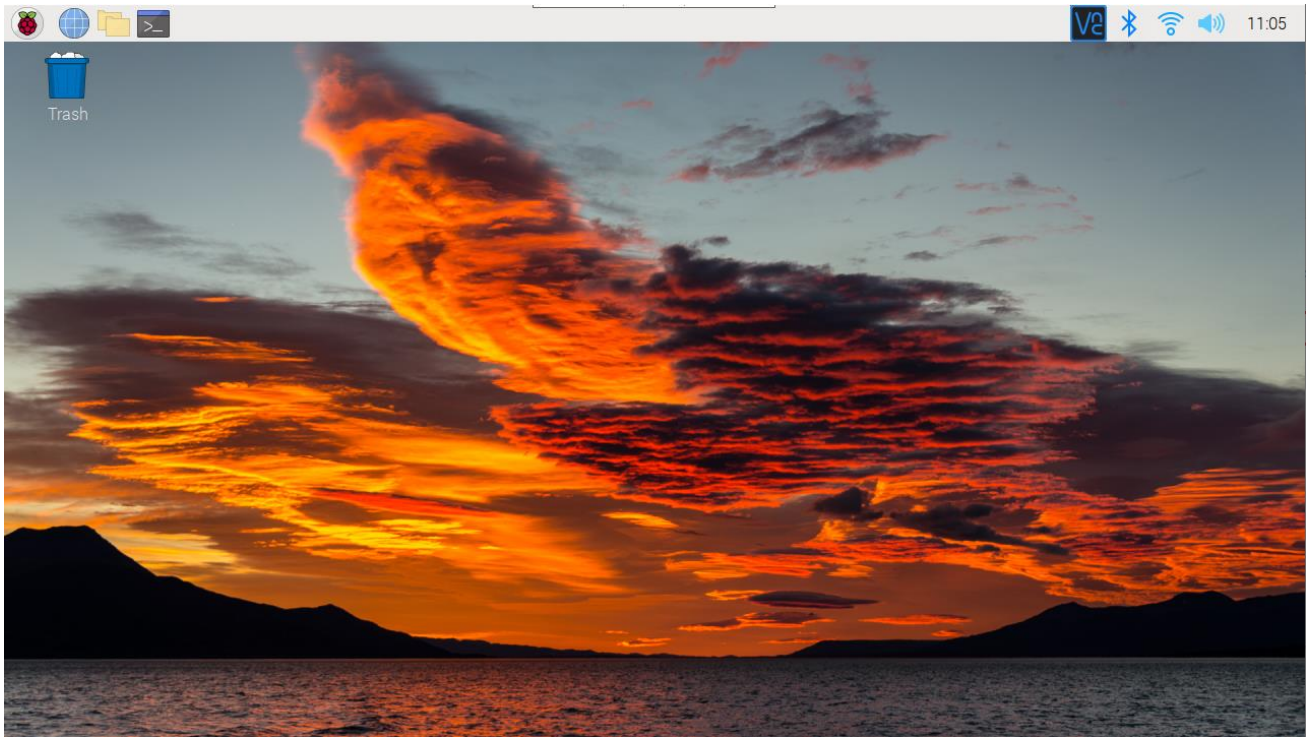
Then download and install VNC Viewer according to your computer system by clicking following link:
https://www.realvnc.com/en/connect/download/viewer/
After installation is completed, open VNC Viewer. And click File → New Connection. Then the interface is shown below.



Enter IP address of your Raspberry Pi and fill in a Name. And 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.

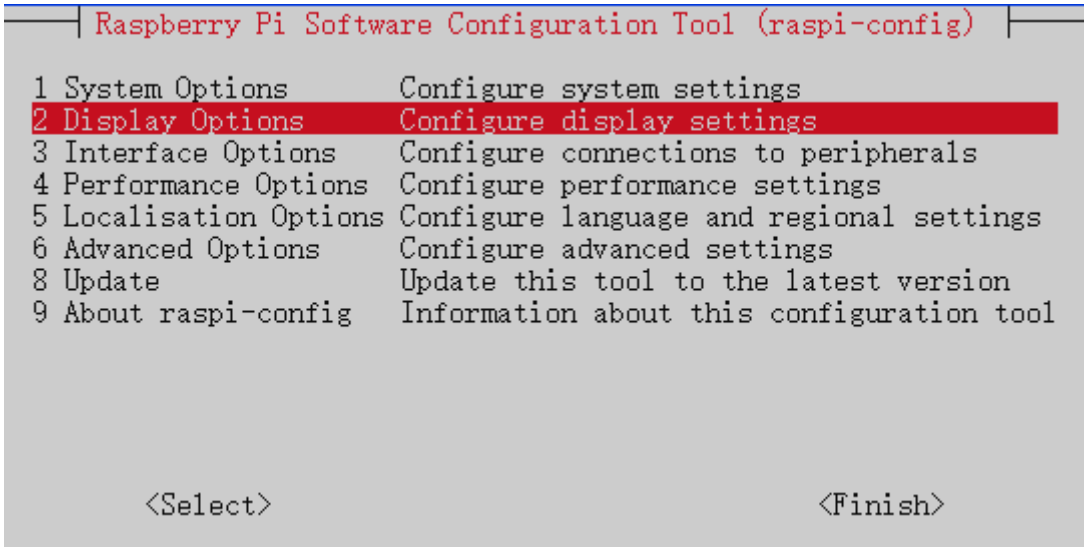


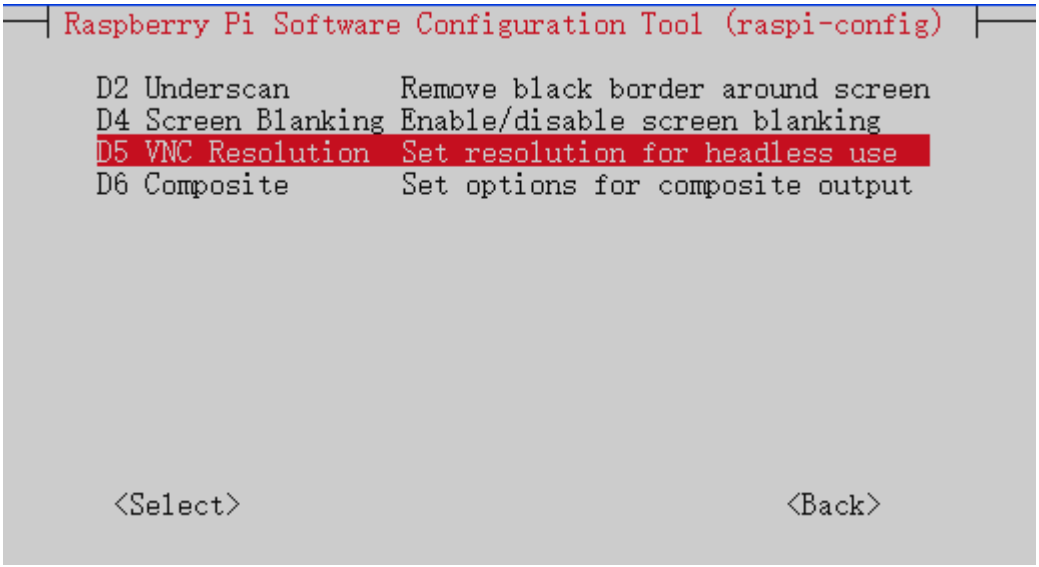
Here, you have logged in to Raspberry Pi successfully by using VNC Viewer

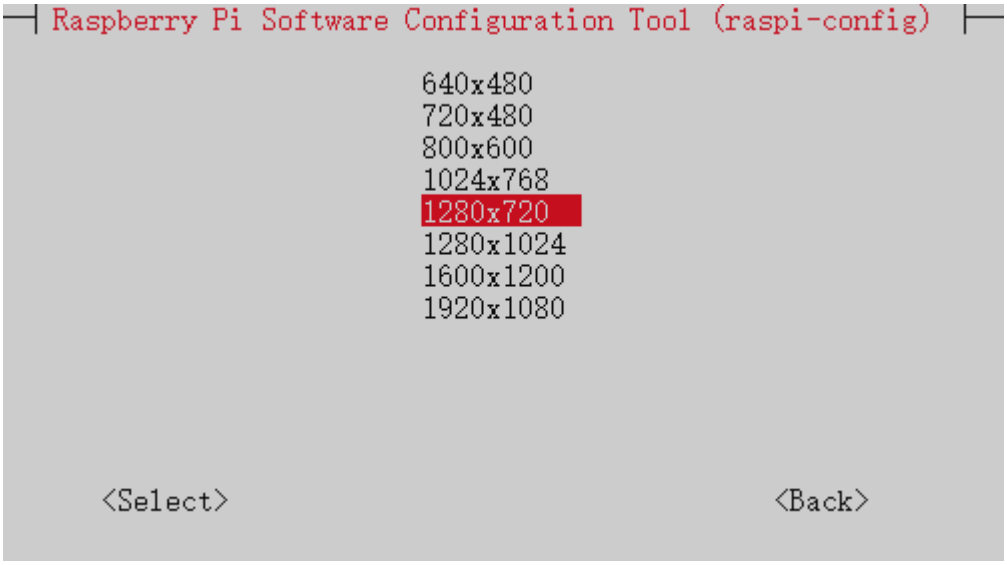
If the resolution ratio is not great or there is just a **little window**, you can set a proper resolution ratio via steps below.

```
sudo raspi-config
```

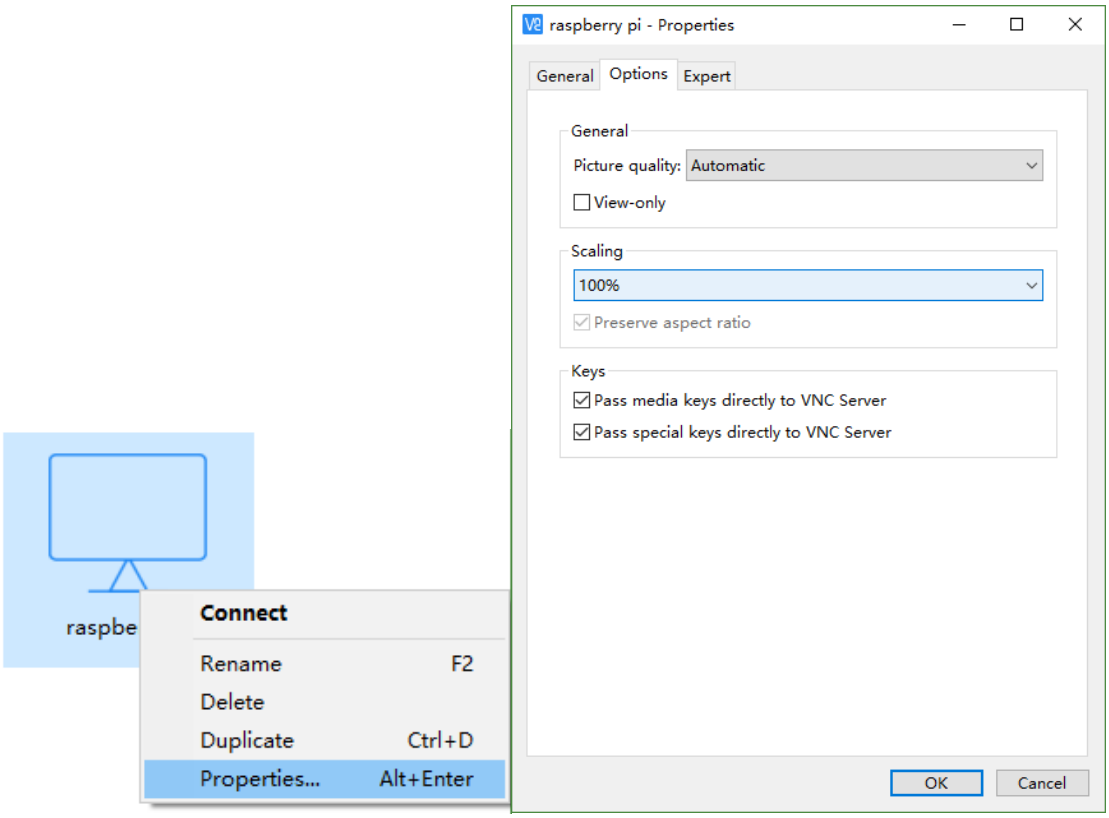
Select **Display Options→VNV Resolution→Proper resolution ratio (set by yourself)→OK→Finish→Yes**. And then reboot Raspberry Pi.







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 1 Software installation and Test (necessary)

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

In this chapter, we will make some necessary preparation: start your Pi Raspberry and install some necessary libraries. Then test some parts. Batteries are needed when driving peripherals such as motors, servos, LEDs, etc.

Note:

1, **Please use Raspberry Pi OS with Desktop**

2, The installation of libraries takes much time. **You can power Raspberry Pi with a power supply Cable**.

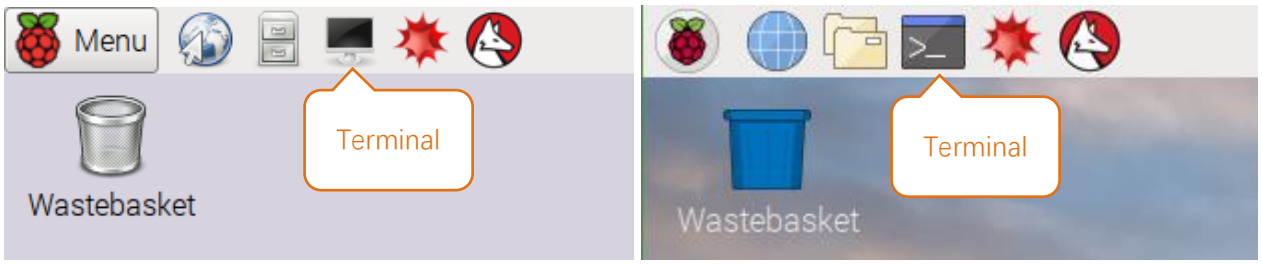
3, If you are using **remote desktop** to login Raspberry Pi, you need to use **VNC viewer**.

You can refer to this video.

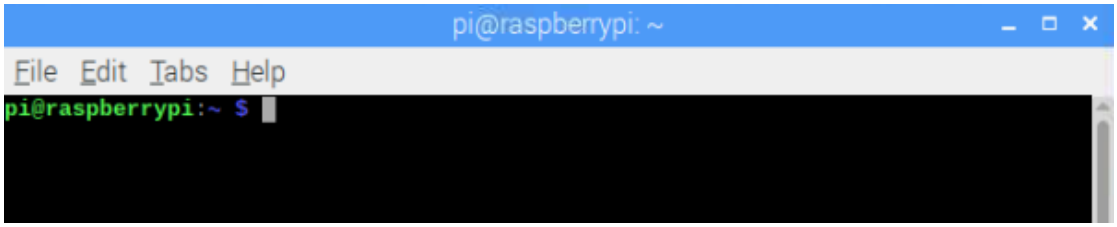
https://youtu.be/3a2OFKBMM-4

Step 1 Obtain the Code and Set python3 as Default

To download the code, you can power Raspberry Pi with a power supply cable **or** switch on S1 (Power Switch). Then open the Raspberry Pi and the terminal. You can open the terminal by clicking as shown below, or you can press “CTRL + ALT + T” on the desktop.

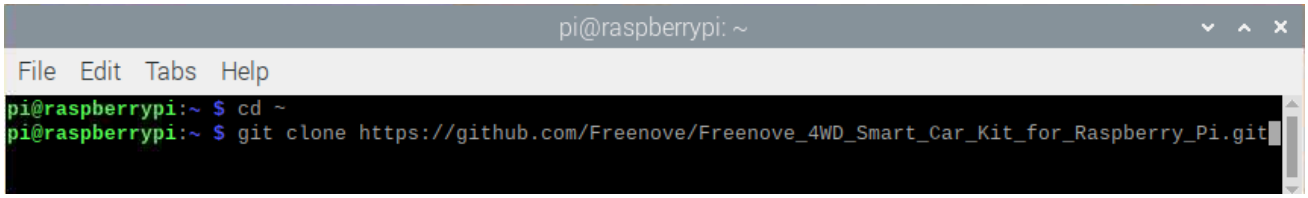


The terminal is shown below:



Open the terminal and type the following commands to obtain the car code. And the code will be placed in the directory "Pi". (Note: Here are two commands. Please execute them in order.)

```
cd
git clone https://github.com/Freenove/Freenove_4WD_Smart_Car_Kit_for_Raspberry_Pi
```



Downloading takes some time. Please wait with patience.

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

Please note that this tutorial is based on python3. If you want to use python2, please download another version of the tutorial.

Set Python3 as default python (Necessary)

First, execute python to check the default python on your Raspberry Pi. Press Ctrl-Z to exit.

```
pi@raspberrypi:~ $ python
```

If it is python3, you can skip this section.

If it is python2, you need execute the following commands to set default python to python3.

1. Enter directory /usr/bin

```
cd /usr/bin
```

2. Delete the original python link.

```
sudo rm python
```

3. Create new python links to python.

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

4. Check python. Press Ctrl-Z to exit.

```
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 set python2 as default python in **other projects**, just repeat the commands above and change python3 to python2.

Shortcut Key

Now, we will introduce several shortcuts that are very **useful** and **commonly used** in terminal.

1. **up and down arrow keys**. History commands can be quickly brought back by using up and down arrow keys, which are very useful when you need to reuse certain commands.

When you need to type commands, pressing "↑" will go backwards through the history of typed commands, and pressing "↓" will go forwards through the history of typed command.

2. **Tab key**. The Tab key can automatically complete the command/path you want to type. When there are multiple commands/paths conforming to the already typed letter, pressing Tab key once won't have any result. And pressing Tab key again will list all the eligible options. This command/path will be completely typed as soon as you press the Tab key when there is only one eligible option.

As shown below, under the '~'directory, enter the Documents directory with the "cd" command. After typing "cd D", press Tab key, then there is no response. Press Tab key again, then all the files/folders that begin with "D" is listed. Continue to type the character "oc", then press the Tab key, and then "Documents" is completely typed automatically.

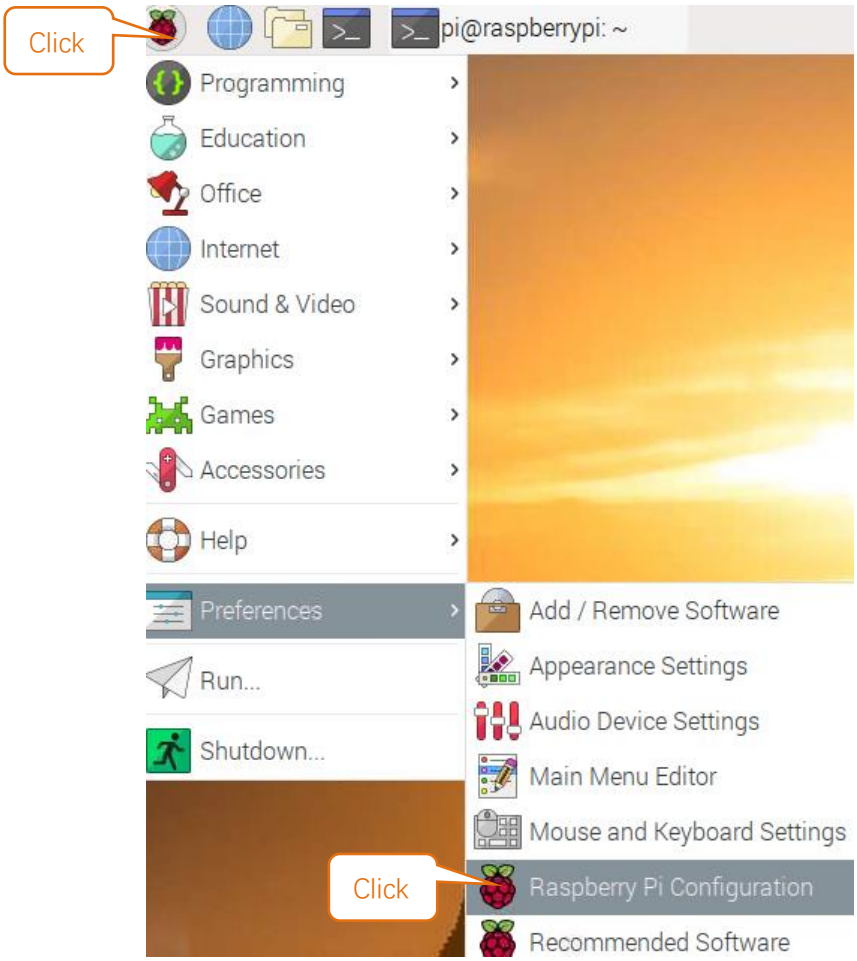
```
pi@raspberrypi:~ $ cd D
Desktop/   Documents/ Downloads/
pi@raspberrypi:~ $ cd Doc
```

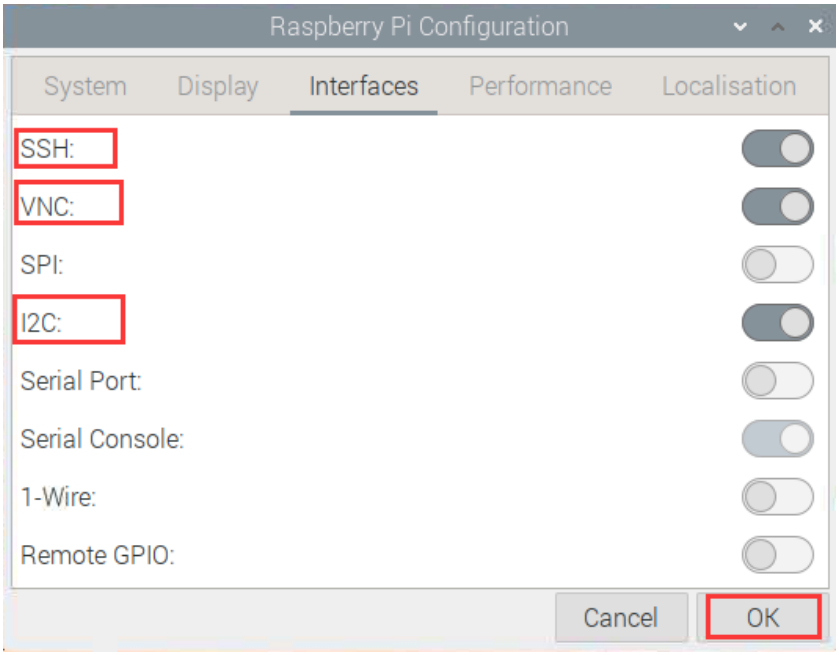
```
pi@raspberrypi:~ $ cd D
Desktop/   Documents/ Downloads/
pi@raspberrypi:~ $ cd Documents/
```

Step 2 Configuration

Enable I2C and VNC

The I2C interface Raspberry Pi is disabled by default. You need to open it manually.





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

```
lsmod | grep i2c
```

If I2C module has been enabled, the following content will show up (the numbers showing in your device may be different):

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

Install I2C-Tools

Type the command to install I2C-Tools.

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

Install python-smbus

Python-smbus is a module of the program Python, which contains some classes and methods to operate I2C.
Type the following command to install python-smbus:

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

Communication test

The smart car board has two chips, PCF8591 and PCA9685. Their I2C addresses are 0X48 and 0X40 respectively. Command “i2cdetect –y 1” can detect whether the board is successfully connected to Raspberry Pi.

```
i2cdetect -y 1
```

```
pi@raspberrypi:~ $ i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: 40 -- -- -- -- -- -- -- 48 -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

If an I2C device is connected to your RPI, its I2C address will be displayed here.

Additional supplement

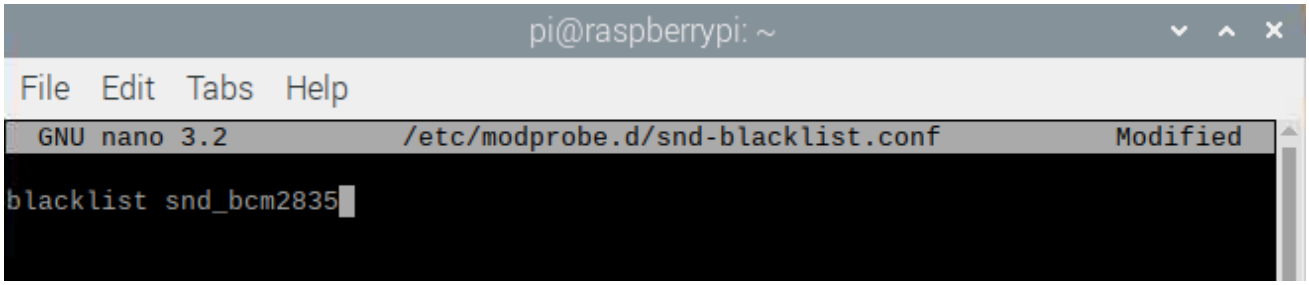
Raspbery Pi, other than 4B and 400, needs to disable the audio module, othewise the LED will not work properly.

1. Create a new snd-blacklist.conf and open it for editing

```
sudo nano /etc/modprobe.d/snd-blacklist.conf
```

Add following content: After adding the contents, you need to press Ctrl+O, Enter, Ctrl+Z.

```
blacklist snd_bcm2835
```



2. We also need to edit config file.

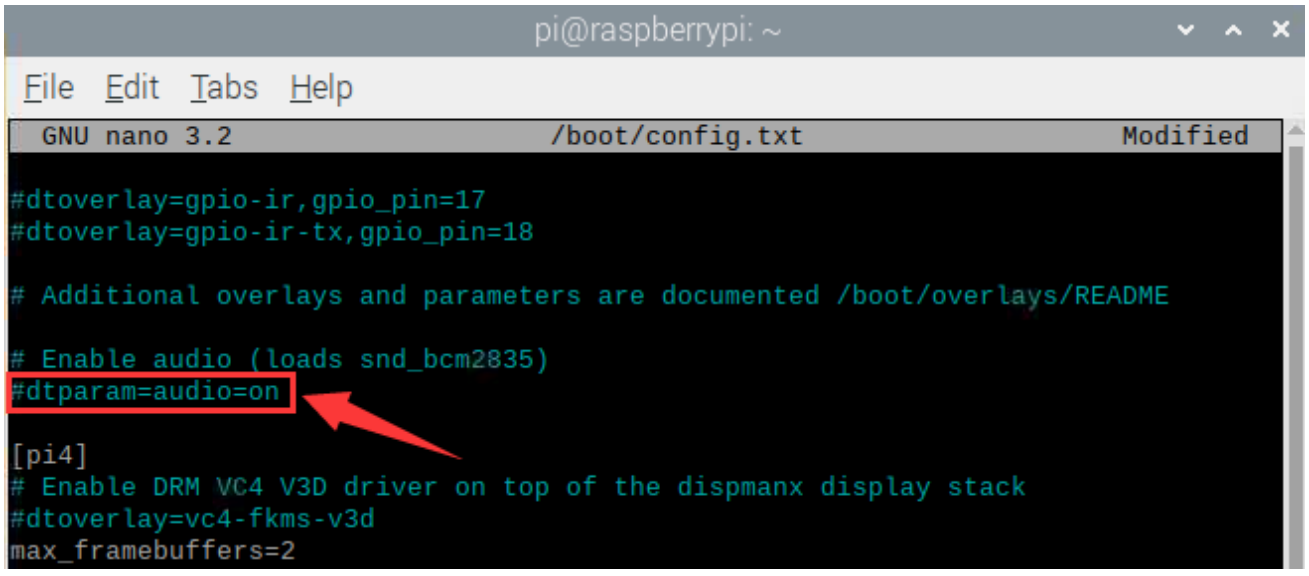
```
sudo nano /boot/config.txt
```

Find the contents of the following two lines (with Ctrl + W you can search):

```
# Enable audio (loads snd_bcm2835)
dtparam=audio=on
```

Add # to comment out the second line. Press Ctrl+O, Enter, Ctrl+X.

```
# Enable audio (loads snd_bcm2835)
# dtparam=audio=on
```



It will take effect after restarting, and you can restart after executing the next section.

If you want to restart the audio module, just restore the content modified in the above two steps.

Step 3 Run the Libraries Installation Program

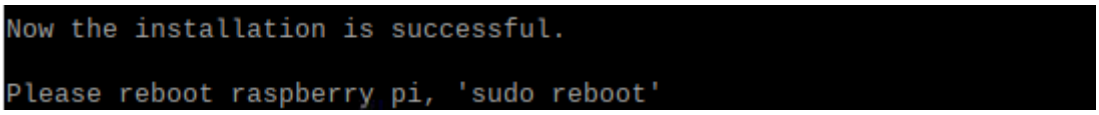
1. Execute following commands to enter directory of “setup.py”.

```
cd ~/Freenove_4WD_Smart_Car_Kit_for_Raspberry_Pi/Code
```

2. Run setup.py

```
sudo python setup.py
```

This program will automatically install the pca9685, rpi_ws281x, PyQt5 library, etc. Please **reboot** the Raspberry Pi after the installation is completed, as shown below.



If the installation fails, please rerun setup.py. After the installation is completed, restart the Raspberry Pi. Most installation failures are caused by network reasons.

```
sudo python setup.py
```

Chapter 2 Assemble Smart Car

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

You can refer to this video: https://youtu.be/G3Q8xNatXgM

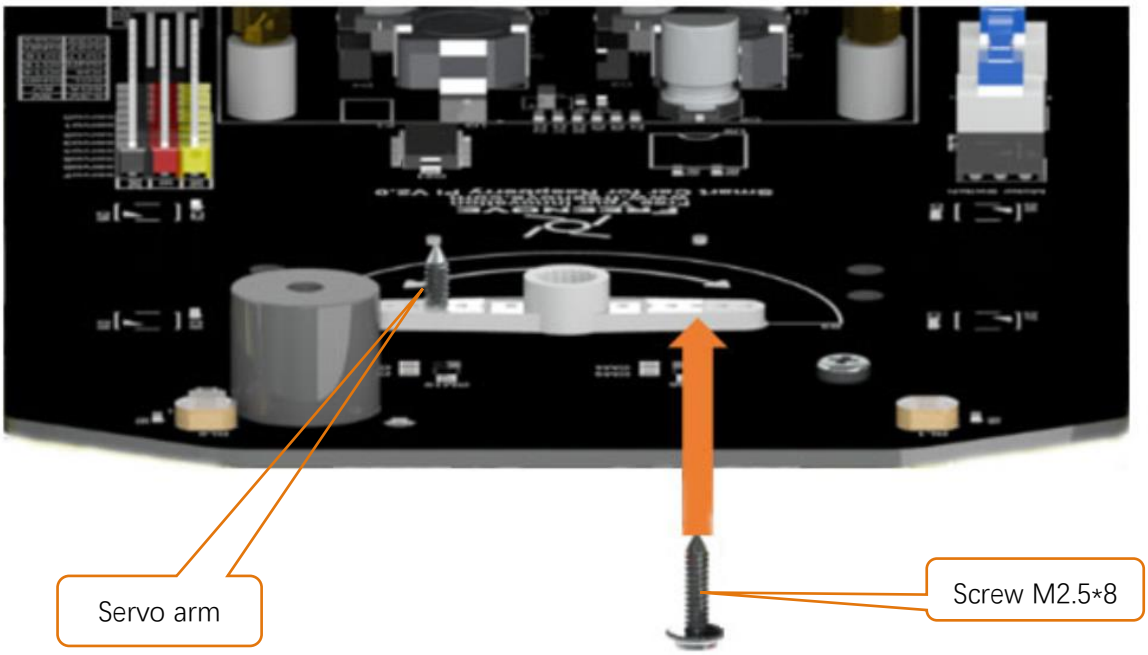
Note：The difference from the previous video is the installation of the wheels.

Please follow STEP 4 of the Wheel installation guide.

Motor, Wheel and Servo arm

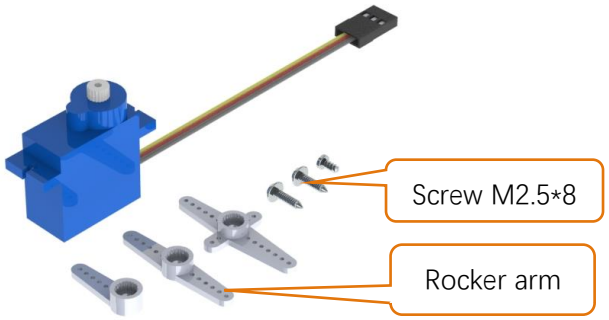
Installation steps:

Step 0



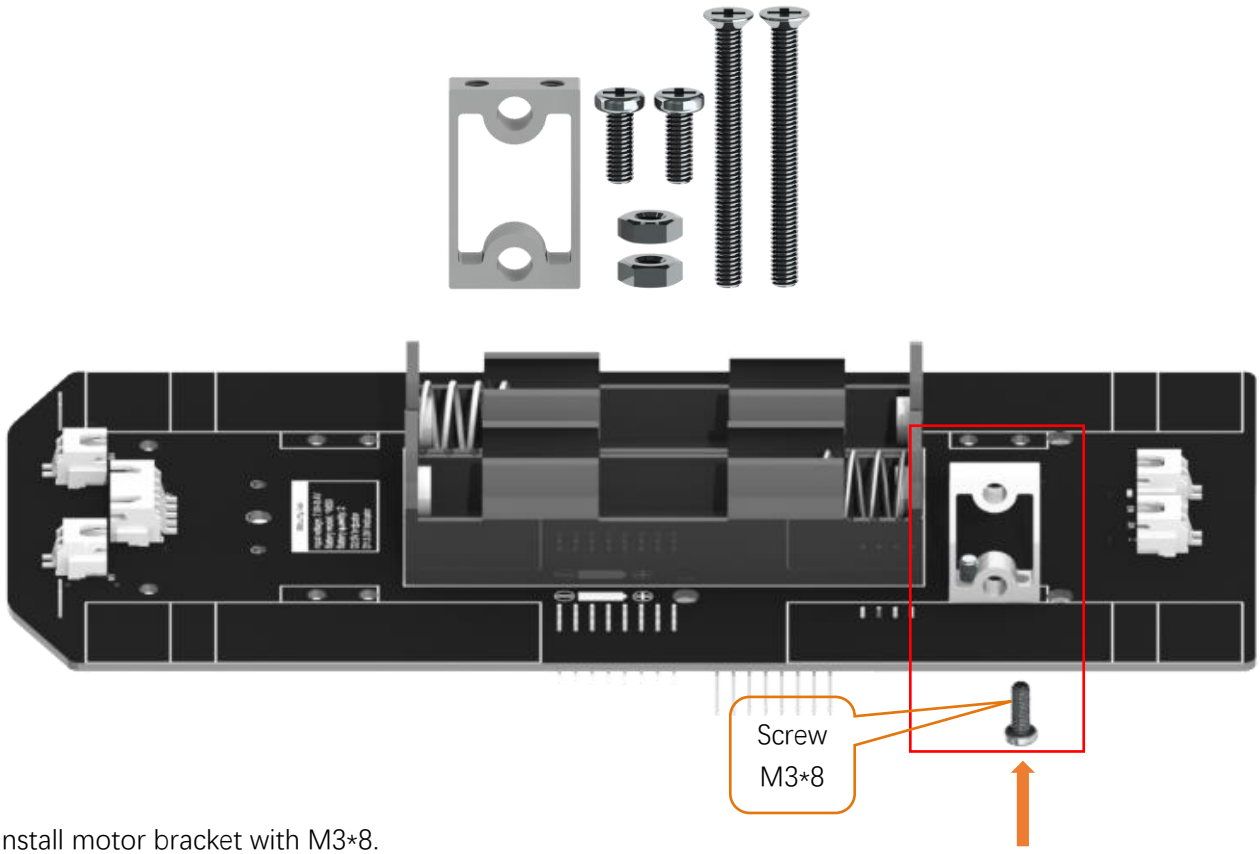
Place the Rocker arm on the smart car board in accordance with its silkscreen printing. Use two M2.5*8 screws to install it to smart car board.

There are two servo packages. Each package contains one servo, three rocker arms, one M2*4 screw and two M2.5*8 screws, as shown below:



Step 1

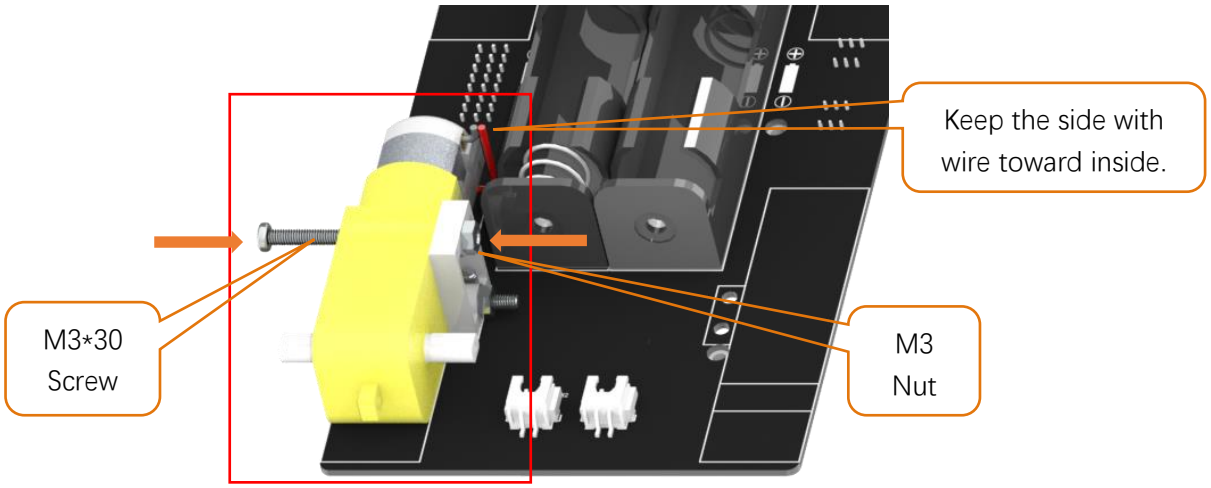
There are 4 bracket packages to fix motors, each containg an aluminum bracket, two M3*30 screws, two M3*8 screws and two M3 nuts, as shown below:



Install motor bracket with M3*8.

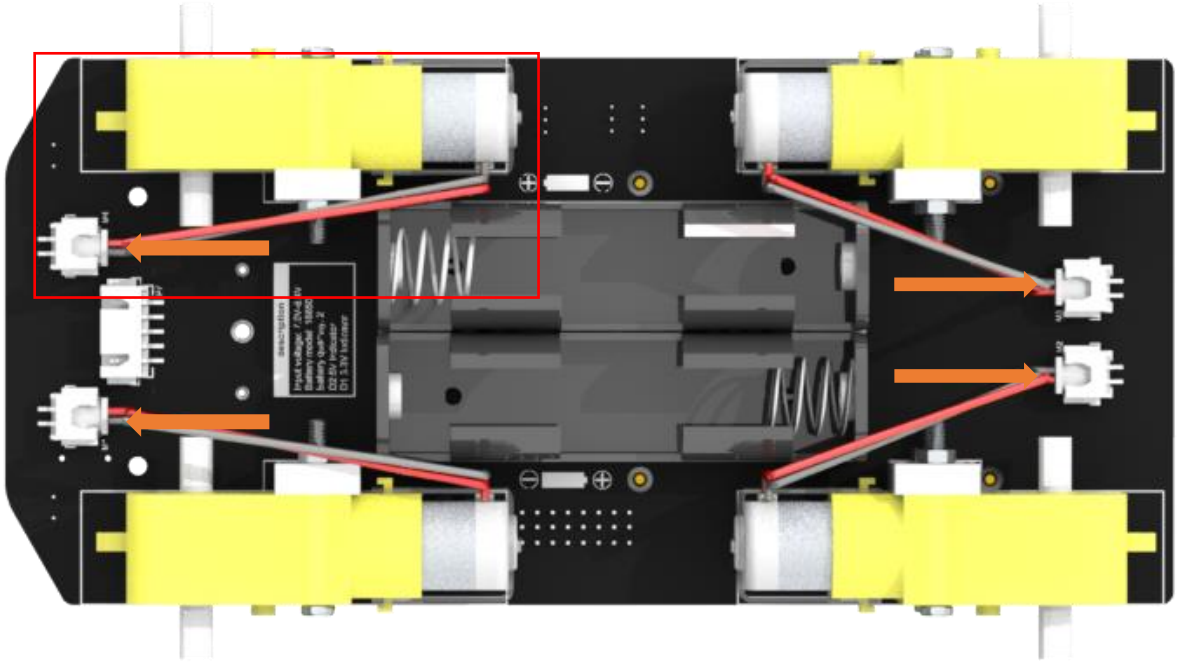
You can also attach this bracket to motor first.

Step 2



Install motor to bracket with M3*30 screw and M3 Nut.

Step 3



The installation of the rest 3 sets of motor is the same. Then connect motor wires to motor ports. If you think the wires are too long, you can tie a knot.

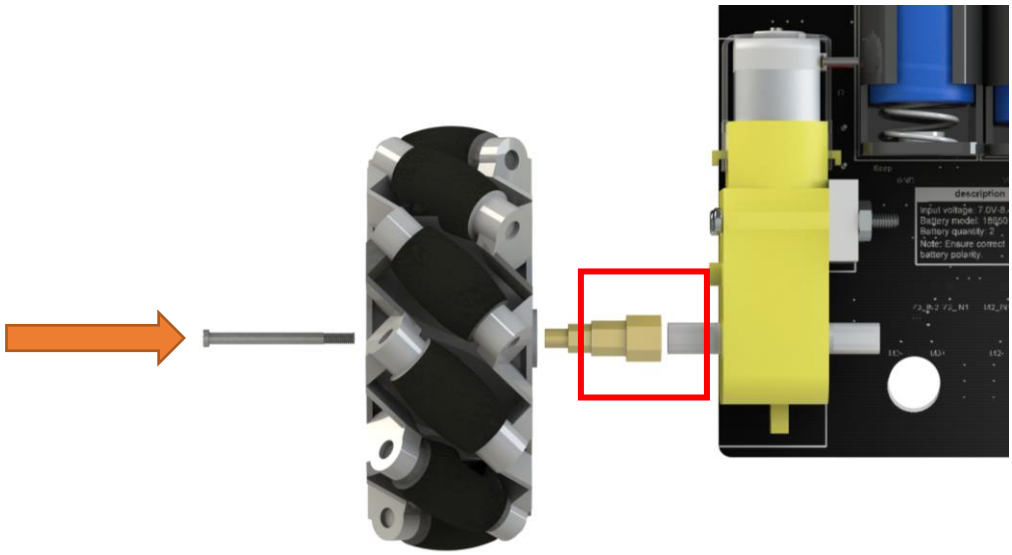
Step 4 Install the mecanum wheels

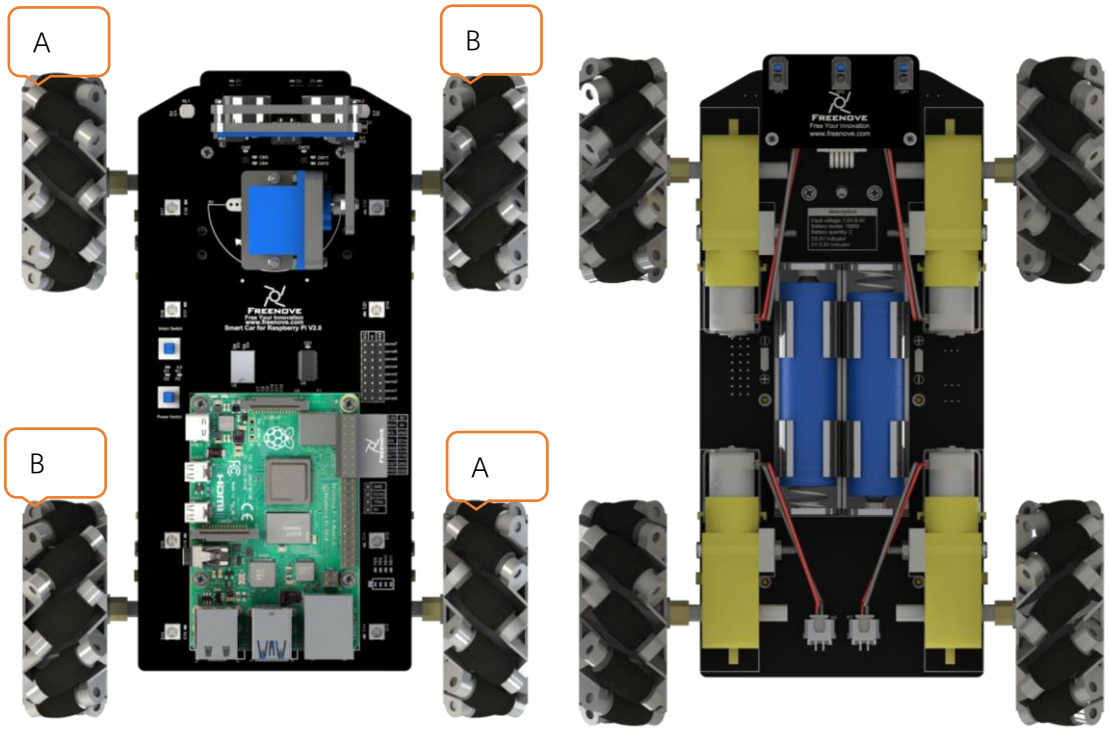
Based on the introduction to the mecanum wheels, assemble them following the sequence of A-B-A-B from M1 to M4.

Slide the mecanum wheel coupling onto the motor's transmission rod.

Attach the wheel to the coupling.

Finally, use the provided screws to tighten the connection securely.

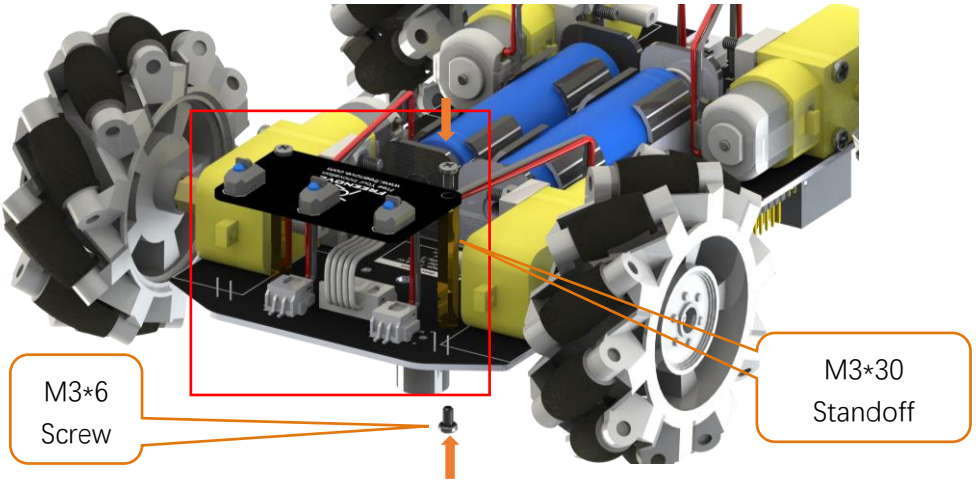




Note: Improper installation of the four wheels can lead to issues that prevent the car from functioning correctly. It is essential to strictly adhere to the installation method shown in the diagram to ensure proper functioning and smooth operation of the mecanum wheel-based car.

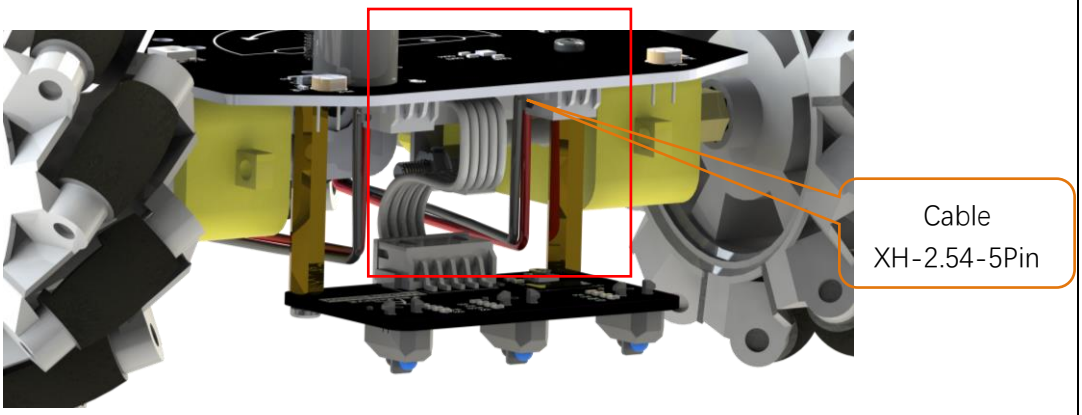
Infrared line tracking module

Step 1



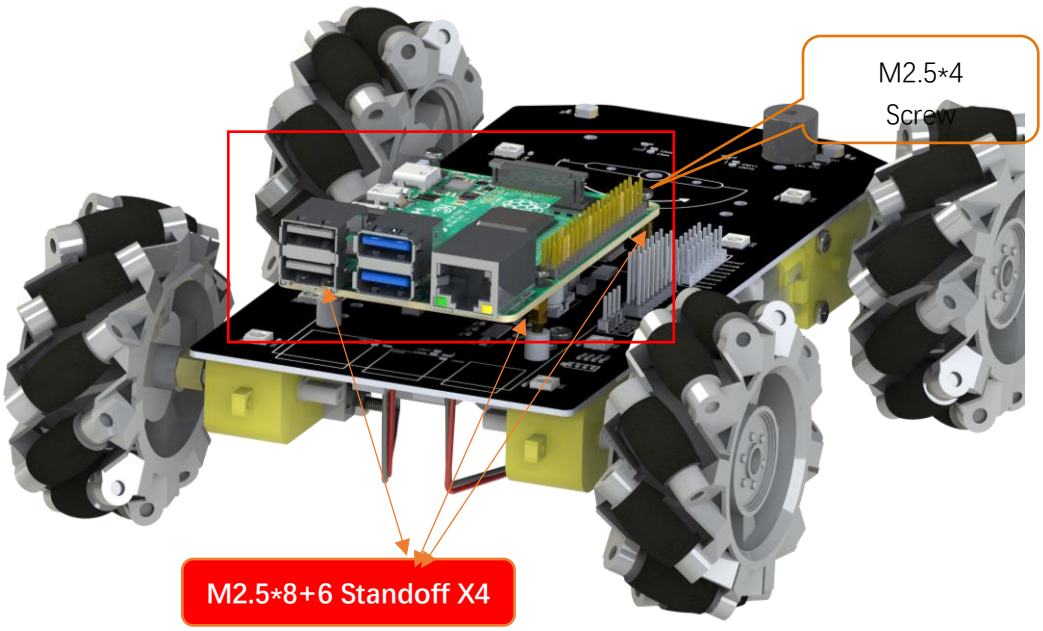
First install two M3*30 standoffs on smart car board with M3*6 screws. Then install line tracking module on standoffs with M3*6 screws.

Step 2



Connect line tracking module to smart car board with XH-2.54-5Pin cable.

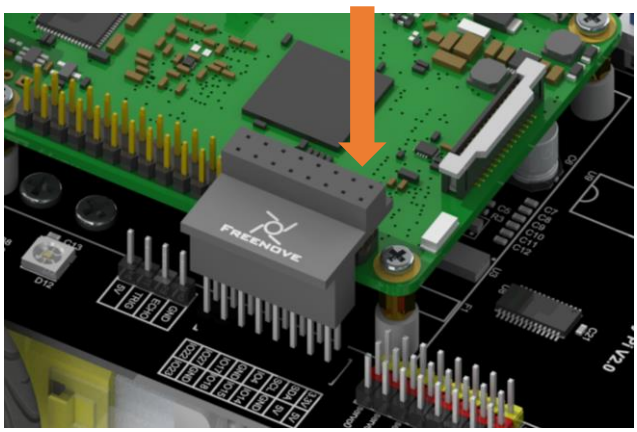
Raspberry Pi



First install four M2.5*8+6 standoffs to the nuts on smart car board. And then place the Rapberry Pi on the standoff in accordance with its silkscreen printing, and use four M2.5*4 screws to install it.

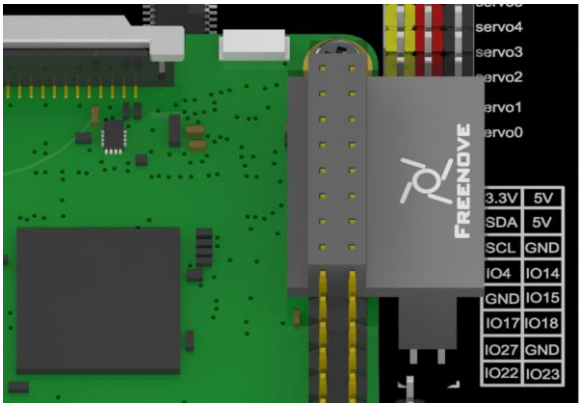
Connection board

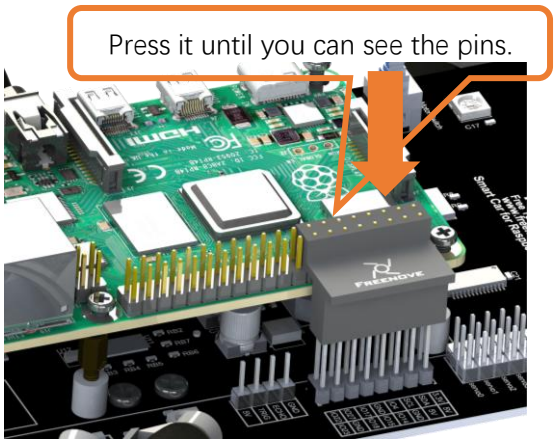
Step 1

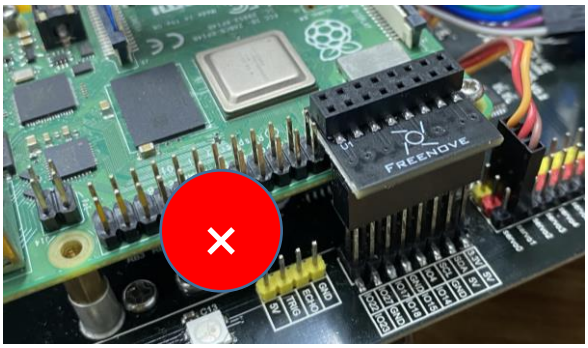


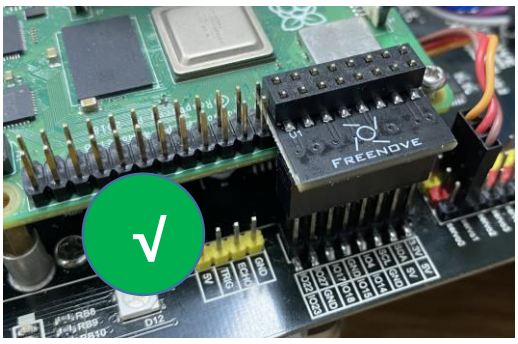
Install the connection board as shown in the figure above. Long female header connector should be connected to smart car board and the short one should be connected to Raspberry Pi.

Step 2



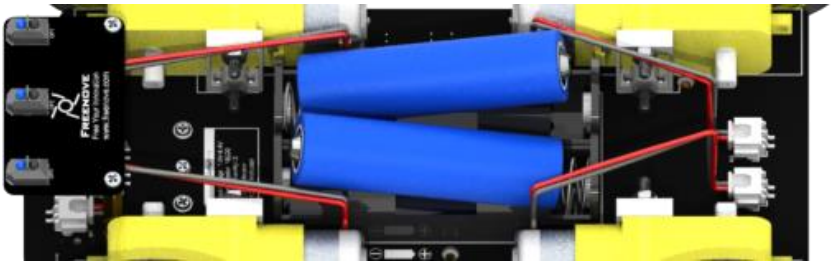






If you have any concerns, please feel free to contact us via support@freenove.com
We will offer you satisfied solution.

Pan Tilt



Finally, install two 18650 batteries. **Please refer to About_Battery.pdf in the unzipped folder.**

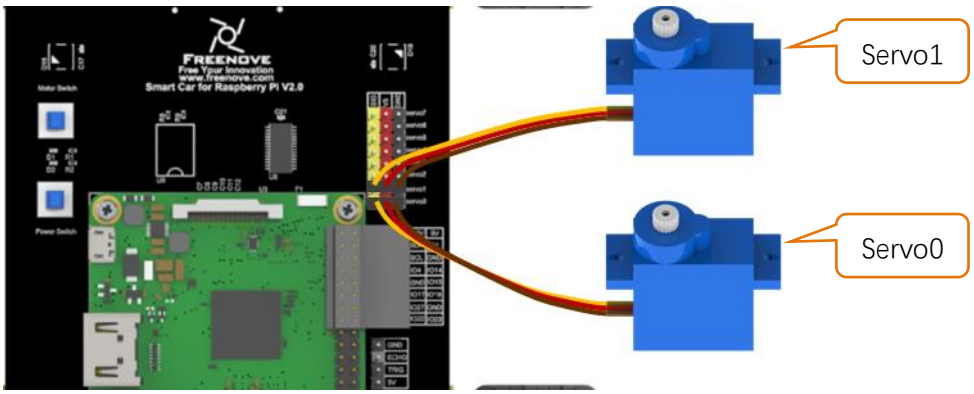
Please push the battery to + ends of battery holder to make the connection good enough.

Run program

In the first chapter, we did not install the Pan-Tilt. Because we need to run programs for the installation of the servos to ensure that the servos rotate to the correct angle.

Next let us install the Pan-Tilt.

Connect two servos to port Servo0 and port Servo1 on the smart car board. And please remember the numbers of the servos.



Enter the following command in the terminal:

If the terminal displays the directory as below (where test.py is located). You can **directly** execute the servo.py command.

```
pi@raspberrypi:~/Freenove_4WD_Smart_Car_Kit_for_Raspberry_Pi/Code/Server $
```

1.If not, execute the cd command:

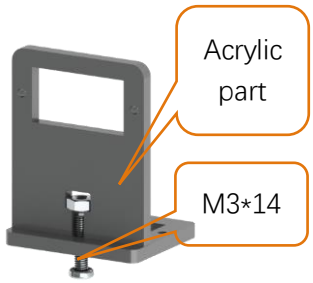
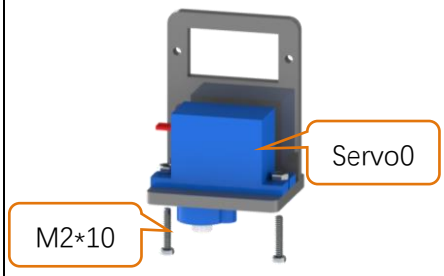
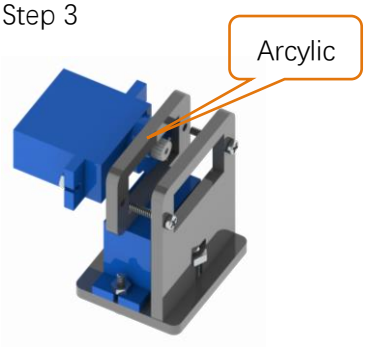
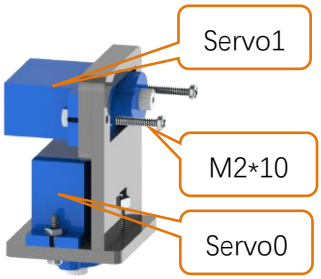
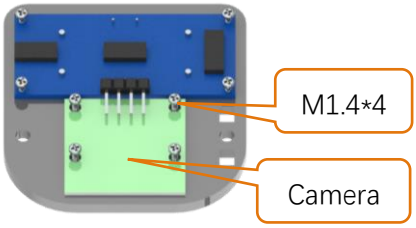
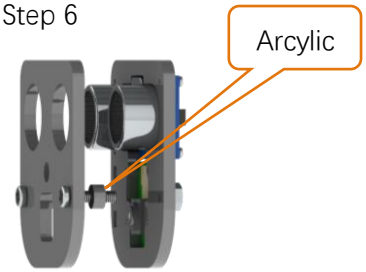
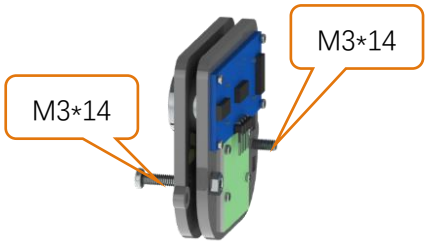
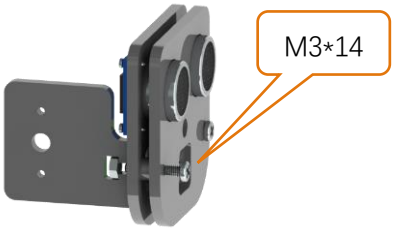
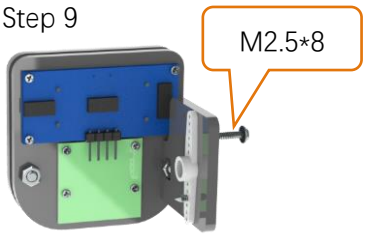
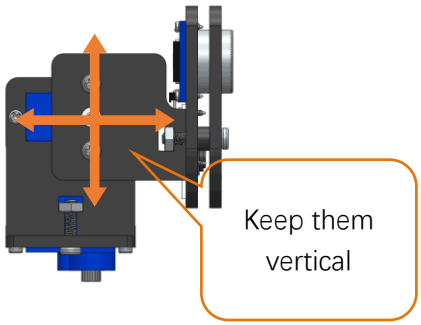
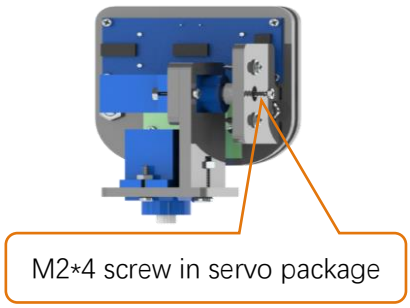
```
cd ~/Freenove_4WD_Smart_Car_Kit_for_Raspberry_Pi/Code/Server
```

2.Execute Servo.py command:

```
sudo python servo.py
```

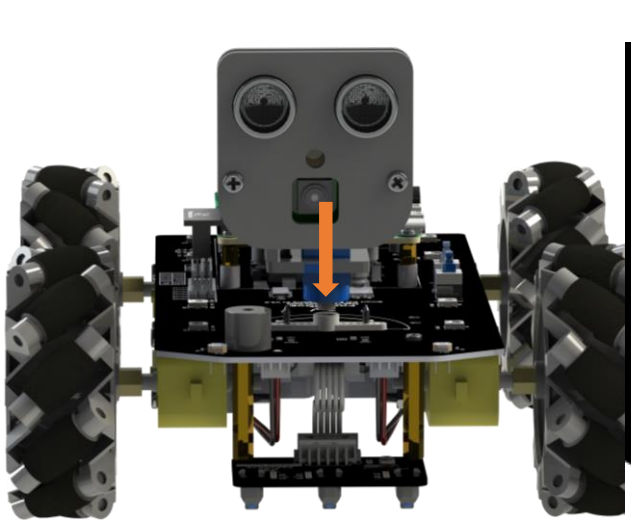
Then servos rotate to a proper angle. Please keep the connection between the servos and the smart car board.

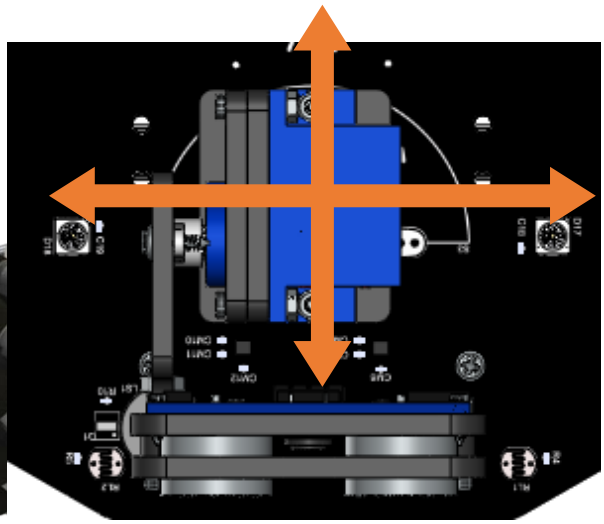
Installation steps: (**Note: Do not disorder Servo0 and Servo1 during the installation.**)

<table>
<tr>
<td>Step 1
 
</td>
<td>Step 2
 
</td>
<td>Step 3
 
</td>
</tr>
<tr>
<td>Step 4
 
</td>
<td>Step 5
 
</td>
<td>Step 6
 
</td>
</tr>
<tr>
<td>Step 7
 
</td>
<td>Step 8:
 
</td>
<td>Step 9
 
</td>
</tr>
<tr>
<td colspan="3">Now please refer to Wiring Section to wring ultrasonic sensor and camera frist.</td>
</tr>
<tr>
<td>Step 10
 
</td>
<td>Step 11
 
</td>
<td>After finished
  </td>
</tr>
</table>

Install Pan Tilt on smart car board.

Step 1

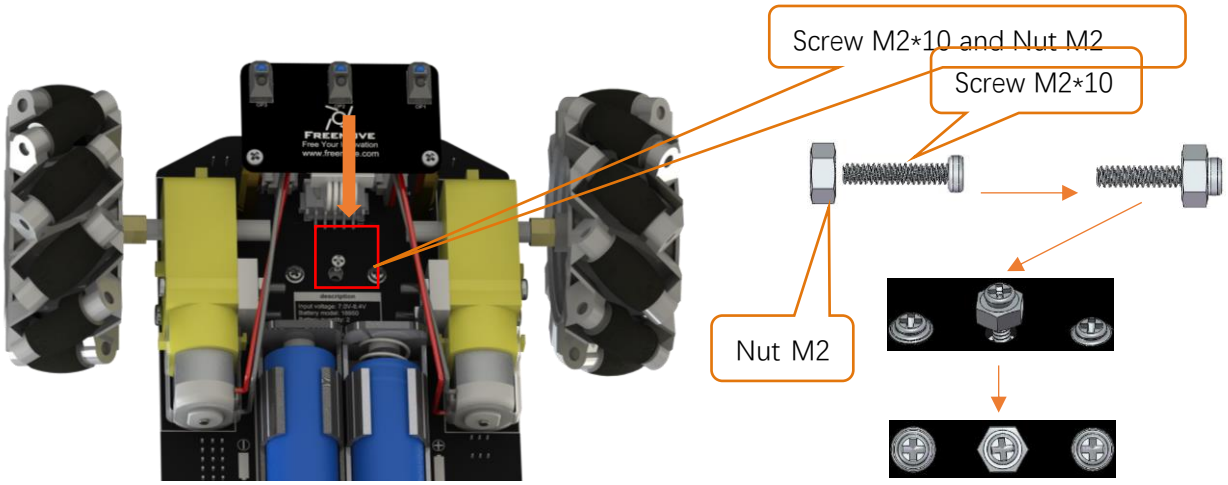




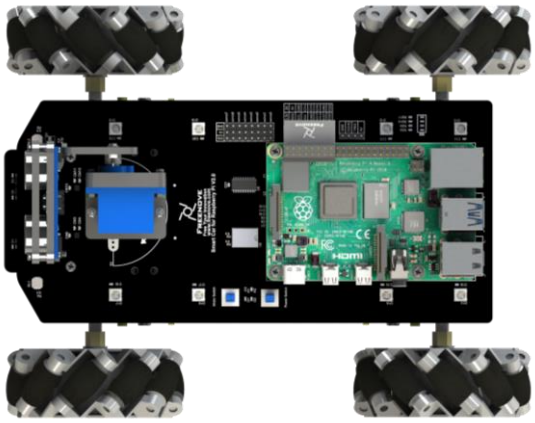
Keep the pan tilt as shown in the right picture and install servo0 with rocker arm.

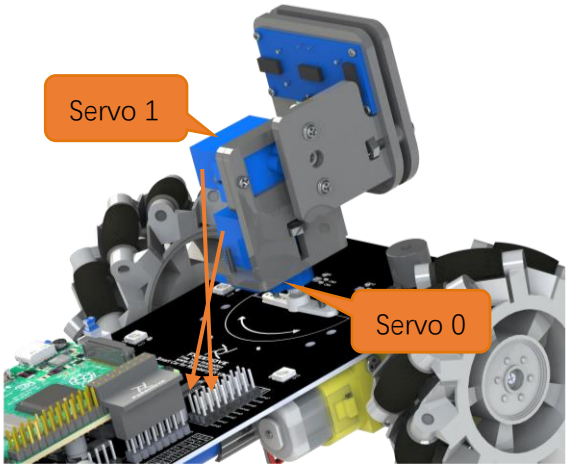
Step 2

Use a Cross screwdrive to support M2 * 10 screws and M2 nuts to fix the servo 0.



Step 3





Pay attention to servo wiring.

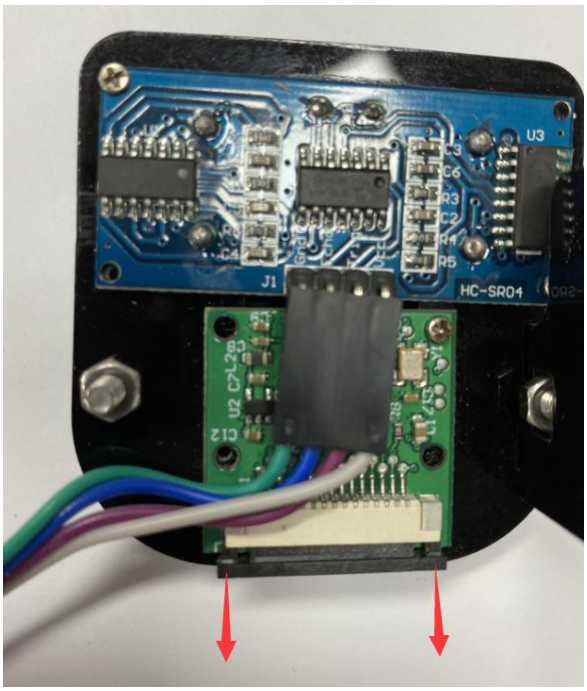
(Note: wiring about the ultrasonic and camera module will be introduced later.)

Wiring

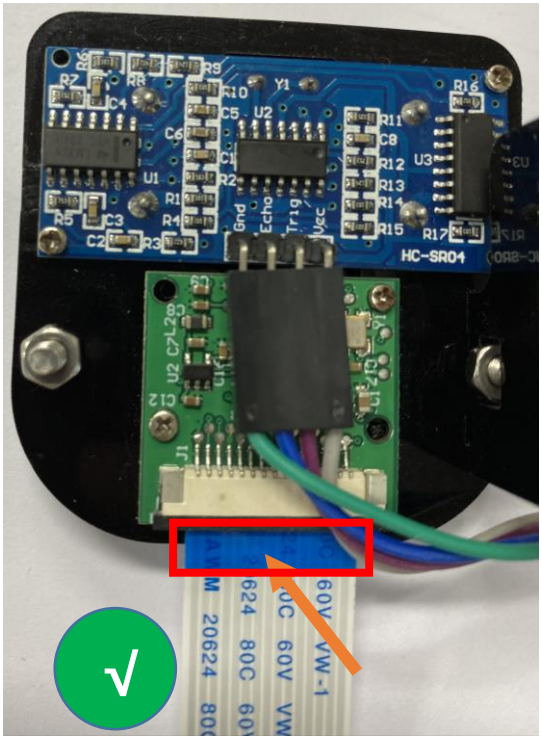
Camera

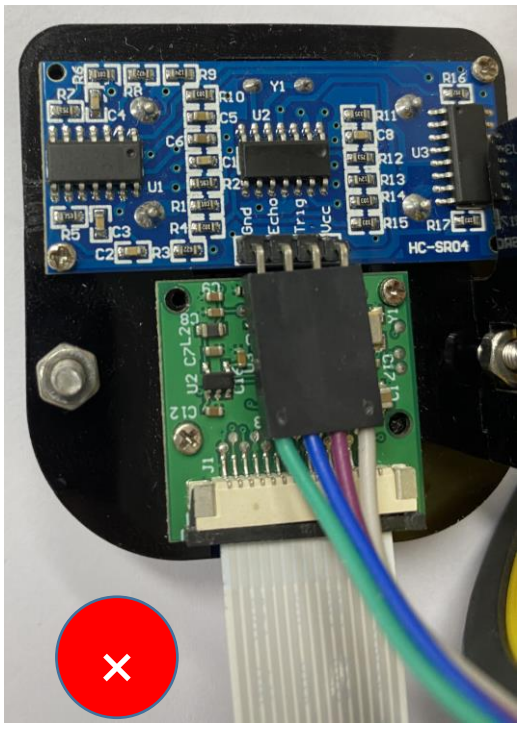
You need shut down Raspberry Pi when wire camera.

Step 1



Step 2



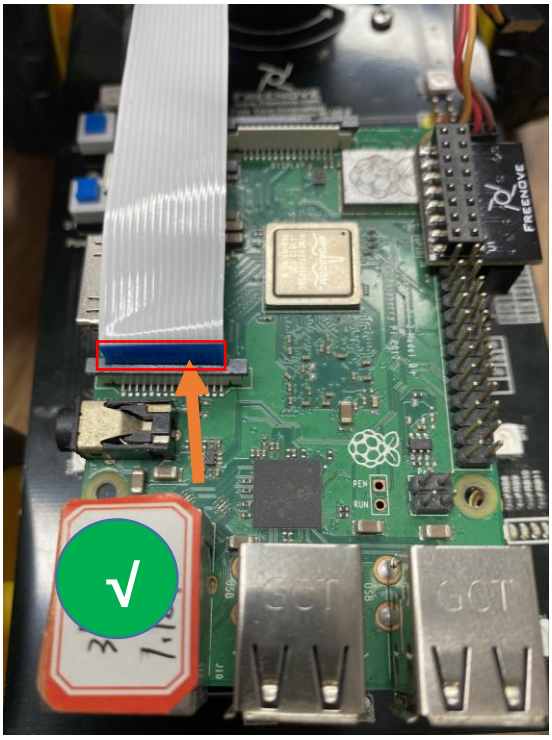


The Blue side of cable should be toward to Servo.

Connect one end of cable to camera. Please note the front and back of the cable.

Step 3

Step 4



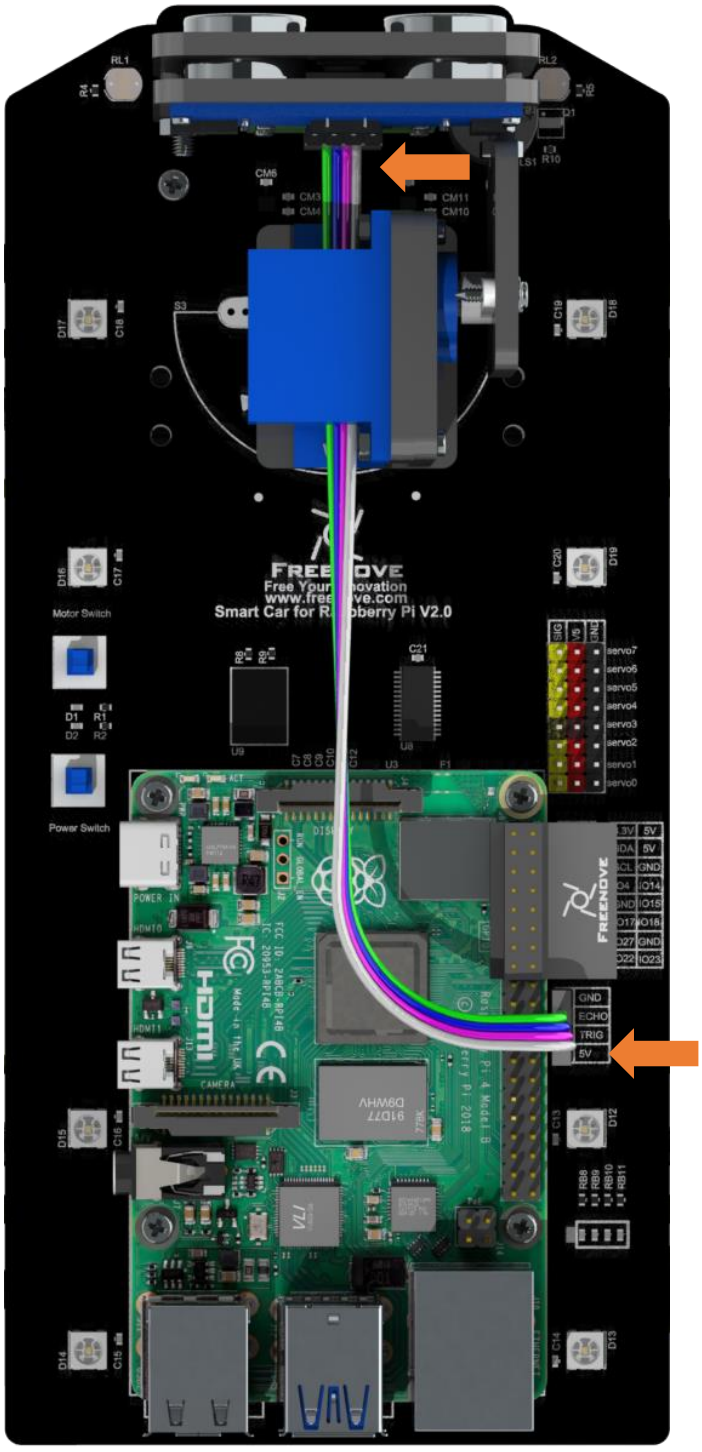
The Blue side of cable should be toward to RPi USB port.

Connect another end of cable to Raspberry Pi. Please note the front and back of the cable.

Ultrasonic

Use jumper wires F/F to connect ultrasonic module with pins on smart car board.

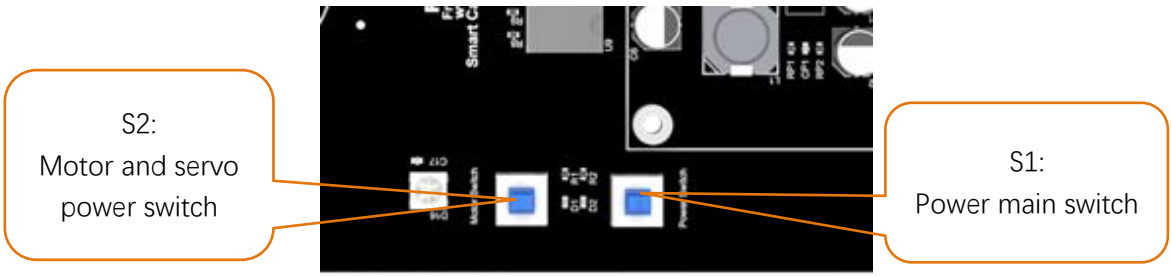
GND-GND, VCC-5V, ECHO-ECHO, TRIG-TRIG



Chapter 3 Module test (necessary)

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

In this section, the car must be equipped with **batteries**, and **Both S1** power switch and **S2** motor switch need to be **preessed**. Then 5V, 3.3V, battery power indicators will be turned on.



During the test, the motor will work. So you can disconnect the wheels or put it on the ground to avoid that it falls down and is damaged. Next, test RGB LED, motor, ultrasonic module, servo, etc.

You can still power Raspberry Pi with a power supply Cable when switches are pressed.

If you have never learned python before, you can learn some basic knowledge via the link below: https://python.swaroopch.com/basics.html

Motor

Basic Movements

Run program

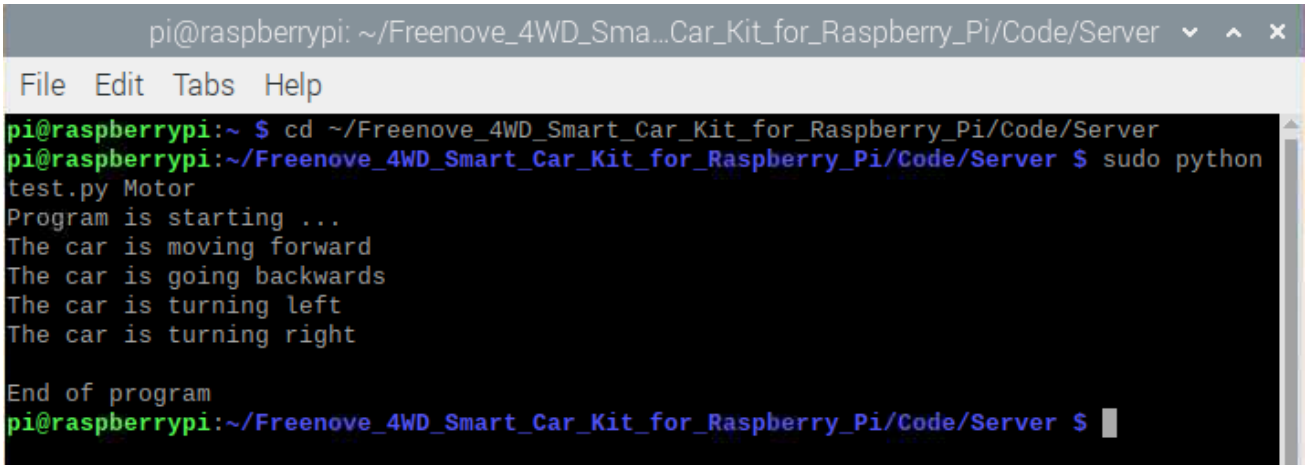
Open the terminal of Raspberry Pi. Enter the following commands to test the motor.

1. Use the cd command to enter the directory where test.py is located.

```
cd ~/Freenove_4WD_Smart_Car_Kit_for_Raspberry_Pi/Code/Server
```

2. Execute test.py command:

```
sudo python test.py Motor
```



Result:

The car advances for 1 second, then reverses for 1 second, followed by a left turn for 1 second, and subsequently a right turn for 1 second. Subsequently, it undergoes a 2-second translation in both left and right directions, as well as in all four diagonal directions. Ultimately, it comes to a stop. You can press "Ctrl + C" to end the program ahead of time. **If the car doesn't work as expected, please check if both switches are pressed.**

If the direction is reversed, it moves back then move forward, please follow steps below.

1. Find Motor.py in the following path in your Raspberry Pi:
Freenove_4WD_Smart_Car_Kit_for_Raspberry_Pi/Code/Server/Motor.py
Open Motor.py and add a "–" before duty1,2,3,4 like below.

```
1  def setMotorModel(self,duty1,duty2,duty3,duty4):
2          duty1,duty2,duty3,duty4=self.duty_range(duty1,duty2,duty3,duty4)
3          self.left_Upper_Wheel(-duty1)
4          self.left_Lower_Wheel(-duty2)
5          self.right_Upper_Wheel(-duty3)
6          self.right_Lower_Wheel(-duty4)
```

Then save the modification and try again.

The code is as below:

```
from Motor import *
PWM=Motor()
def test_Motor():
    try:
        PWM.setMotorModel(1000,1000,1000,1000)        #Forward
        print ("The car is moving forward")
        time.sleep(1)
        PWM.setMotorModel(-1000,-1000,-1000,-1000)     #Back
        print ("The car is going backwards")
        time.sleep(1)
        PWM.setMotorModel(-1500,-1500,2000,2000)        #Turn left
        print ("The car is turning left")
        time.sleep(1)
        PWM.setMotorModel(2000,2000,-1500,-1500)        #Turn right
        print ("The car is turning right")
        time.sleep(1)
        PWM.setMotorModel(-2000,2000,2000,-2000)        #Move left
        print ("The car is moving left")
        time.sleep(1)
        PWM.setMotorModel(2000,-2000,-2000,2000)        #Move right
        print ("The car is moving right")
        time.sleep(1)
        PWM.setMotorModel(0,2000,2000,0)         #Move diagonally to the left and forward
        print ("The car is moving diagonally to the left and forward")
        time.sleep(1)
        PWM.setMotorModel(0,-2000,-2000,0)        #Move diagonally to the right and backward
        print ("The car is moving diagonally to the right and backward")
        time.sleep(1)
        PWM.setMotorModel(2000,0,2000,00)        #Move diagonally to the right and forward
        print ("The car is moving diagonally to the right and forward")
        time.sleep(1)
        PWM.setMotorModel(-2000,0,-2000,0)        #Move diagonally to the left and backward
        print ("The car is moving diagonally to the left and backward")
        time.sleep(1)
        PWM.setMotorModel(0,0,0,0)                #Stop
        print ("\nEnd of program")
    except KeyboardInterrupt:
        PWM.setMotorModel(0,0,0,0)
        print ("\nEnd of program")
```

Reference

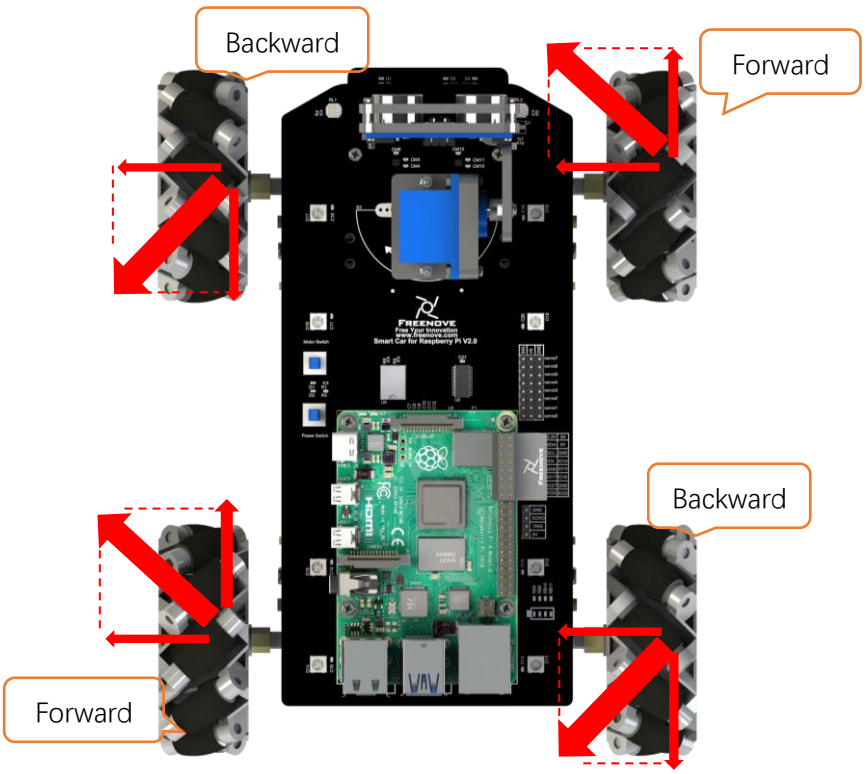
setMotorModel(data1,data2,data3,data4)

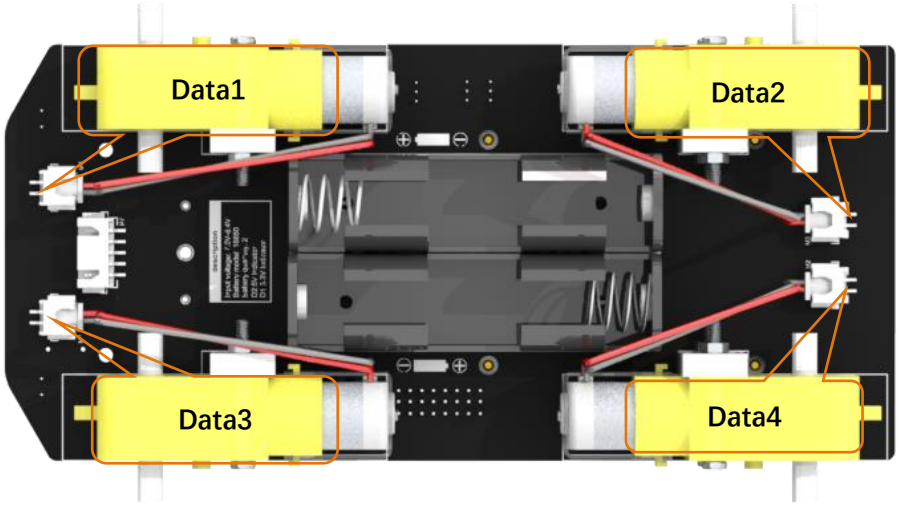
This function has four input parameters that control the left front motor, the left rear motor, the right front motor, and the right rear motor. When the input parameter is within 0~4096, the motor will rotate forward. If it is within -4096~0, the motor will rotate reversely. The larger the absolute value is, the faster the motor is. When the input is 0, the motor will stop. If the function is input as follows: setMotorModel(2000,2000, 2000, 2000), four motors will rotate forward and the car will move forward.

Now, let's explore how the car accomplishes omnidirectional movement. Drawing from the earlier discussion on mecanum wheel installation, we understand that these wheels can generate two sets of diagonal 45-degree velocities. Each velocity can further be decomposed into two components: one along the X-axis and the other along the Y-axis.

Let's take a closer look at the function " PWM.setMotorModel(-2000,2000,2000,-2000) " mentioned in line seventeen, which enables the car to move to the left. In this function, the parameters represent the speed and direction of each of the four wheels.

Since the speed of all four wheels is equal at 2000, the X-axis and Y-axis components of the velocity are also equal. On one side of the car, two wheels generate a forward and a backward velocity, effectively canceling each other out. However, this results in a solitary leftward velocity component. The leftward velocities from both sides combine, thus moving the car laterally to the left.





Data1	Data2	Date3	Data4	Movements of the Car
X	X	X	X	Forward
-X	-X	-X	-X	Backward
-X	-X	X	X	Turn left
X	X	-X	-X	Turn right
-X	X	X	-X	Move left
X	-X	-X	X	Move right
0	X	X	0	Left and forward
0	-X	-X	0	Right and backward
X	0	0	X	Right and forward
-X	0	0	-X	Left and backward
0	0	0	0	Stop

The above describes the common actions performed by the Raspberry Pi with fixed and equally-paced movements of mecanum wheels. In this context, X represents a positive integer, with values ranging from 0 to 4096.In this context, X represents a positive integer, with values ranging from 0 to 4096.

Advanced Movements

Run program

Open the terminal of Raspberry Pi. Enter the following commands to test the motor.

1. Use the cd command to enter the directory where test.py is located.

```
cd ~/Freenove_4WD_Smart_Car_Kit_for_Raspberry_Pi/Code/Server
```

2. Execute test.py command:

```
sudo python test.py Rotate
```

```
pi@raspberrypi:~/Freenove_4WD_Smart_Car_Kit_for_Raspberry_Pi/Code/Server $ sudo python test.py Rotate
Program is starting ...
rotating
rotating
rotating
rotating
rotating
rotating
rotating
rotating
^C
End of program
pi@raspberrypi:~/Freenove_4WD_Smart_Car_Kit_for_Raspberry_Pi/Code/Server $
```

Result:

The car rotates while moving straight ahead, and the console continuously prints "rotating". If you find that the car isn't rotating as expected, access the "Motor.py" file and adjust the value of "time_propotion". Here are the detailed steps:

1. Switch the path to the program folder

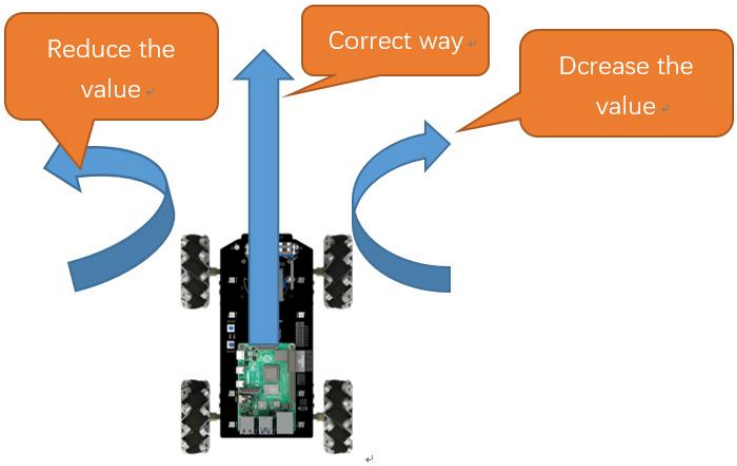
```
cd ~/Freenove_4WD_Smart_Car_Kit_for_Raspberry_Pi/Code/Server
```

2. Open “Motor.py” with nano editing tool.

```
sudo nano Motor.py
```

3. Modify the value of “time_propotion”, do not make significant increases or decreases at once; keep the changes within a range of tenths place (one decimal point).

```
class Motor:
    def __init__(self):
        self.pwm = PCA9685(0x40, debug=True)
        self.pwm.setPWMFreq(50)
        self.time_proportion = 3      #Depend on your own car,If you want to get the best out of the
```



4. Press Ctrl + O to save the change and Enter to confirm. Exit the editor with Ctrl+x.

Note: Changing to an appropriate value is crucial as it determines whether the car can successfully execute both the rotation and the straight movement.

Reference

```
def Rotate(self,n):
    angle = n
    bat_compensate =7.5/(self.adc.recvADC(2)*3)
    while True:
        W = 2000

        VY = int(2000 * math.cos(math.radians(angle)))
        VX = -int(2000 * math.sin(math.radians(angle)))

        FR = VY - VX + W
        FL = VY + VX - W
        BL = VY - VX - W
        BR = VY + VX + W

        PWM.setMotorModel(FL, BL, FR, BR)
        print("rotating")
        time.sleep(5*self.time_proportion*bat_compensate/1000)
        angle -= 5
```

In the preceding basic movements, we employed a fixed and consistent speed, ensuring that all four Mecanum wheels maintained equal velocity. If you are not familiar with the movement of mecanum wheels, please refer to the introduction of mecanum wheels in the preface for more information.

Next, understand how the mecanum wheel car accomplishes both forward movement and rotation. In the program, VY represents the speed mapped to the Y-axis during rotational motion, while VX represents the speed mapped to the X-axis. The angular velocity of the car's rotation is denoted by W, with counterclockwise direction considered positive.

Lines ten to thirteen involve calculating the distribution of speeds to each wheel based on the motion equation of the mecanum wheel chassis. We set an appropriate speed for the car to spin counterclockwise, and use the delay function to rotate the car by 5°. To ensure the direction of forward movement aligns with the rotation, we adjust the corresponding angle by subtracting 5°.

Additionally, the delay function includes a simple voltage linear compensation. You have the option to modify the proportion value "time_proportion" to achieve the best forward rotation effect, tailoring it to your specific requirements.

ADC Module

Run program

Enter the following commands to test ADC module.

If the terminal displays the directory as below (where test.py is located). You can **directly** execute the test.py command.

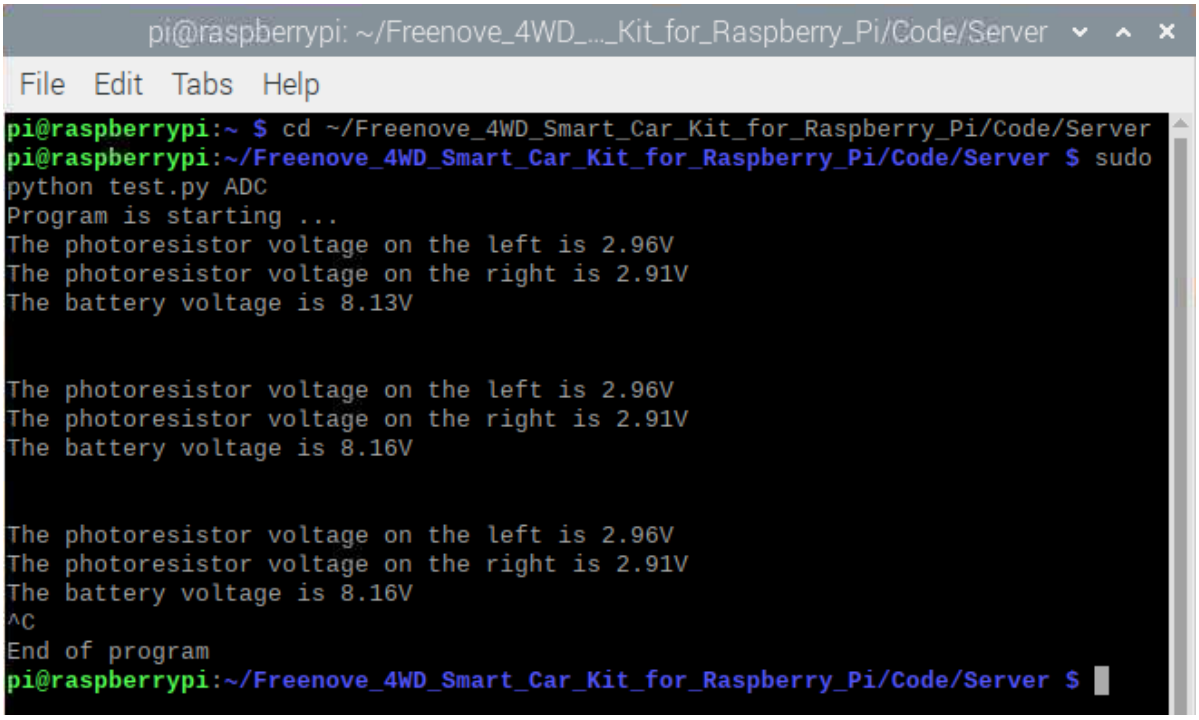
```
pi@raspberrypi:~/Freenove_4WD_Smart_Car_Kit_for_Raspberry_Pi/Code/Server $
```

1. If not, execute the cd command:

```
cd ~/Freenove_4WD_Smart_Car_Kit_for_Raspberry_Pi/Code/Server
```

2. Execute test.py command:

```
sudo python test.py ADC
```



Result:

Every 1s, the voltage values of the two photoresistors and the battery are output. The value read for the first time is not stable and inaccurate when the chip just starts. It will be stable later. You can press "Ctrl + C" to end program.

The code is as below:

```
from ADC import *
adc=Adc()
def test_Adc():
    try:
        while True:
            Left_IDR=adc.recvADC(0)
            print ("The photoresistor voltage on the left is "+str(Left_IDR)+"V")
            Right_IDR=adc.recvADC(1)
            print ("The photoresistor voltage on the right is "+str(Right_IDR)+"V")
            Power=adc.recvADC(2)
            print ("The battery voltage is "+str(Power*3)+"V")
            time.sleep(1)
            print '\n'
    except KeyboardInterrupt:
        print "\nEnd of program"
```

Reference

recvADC(channel)

This function has only one input parameter, which can be 0, 1 or 2.

When the input is **0**, the value of this function is the voltage value of the **left** photoresistor.

When the input is **1**, the value of this function is the voltage value of the **right** photoresistor.

When the input is **2**, the value of this function is the voltage value of **each battery.** After multiplying by 3, it is the actual battery voltage value

Infrared Line tracking module

Run program

Enter the following command in the terminal to test line tracking module.

If the terminal displays the directory as below (where test.py is located), you can **directly** execute the test.py command.

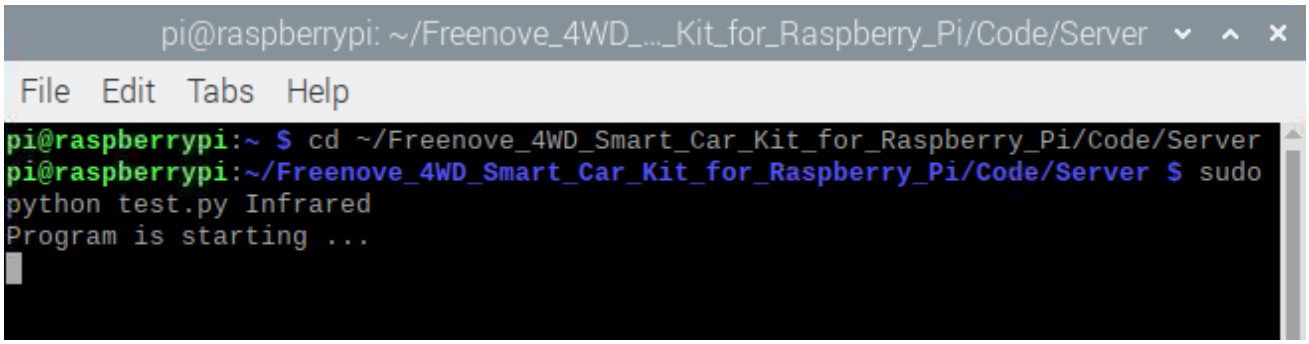
```
pi@raspberrypi:~/Freenove_4WD_Smart_Car_Kit_for_Raspberry_Pi/Code/Server $
```

1. If not, execute the cd command:

```
cd ~/Freenove_4WD_Smart_Car_Kit_for_Raspberry_Pi/Code/Server
```

2. Execute test.py command:

```
sudo python test.py Infrared
```



Result:

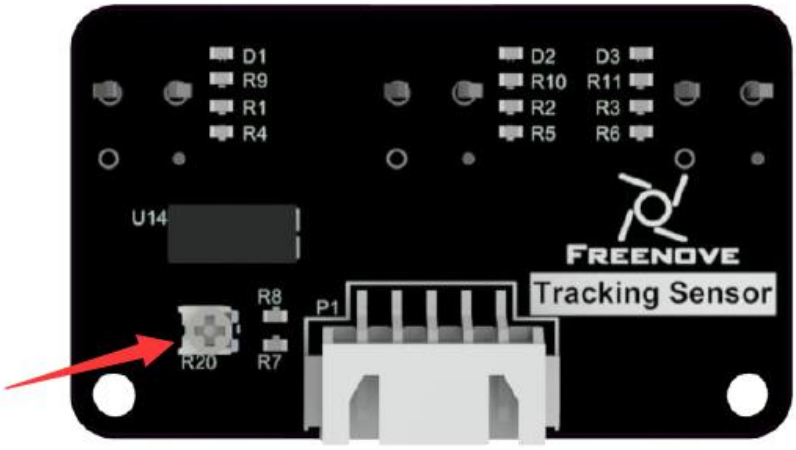
When the black line is on the left side of the module, the left LED will light up and the terminal will print “Left”;

When the black line is in the middle of the module, the middle LED will light up and the terminal will print “Middle”.

When the black line is on the right side of the module, right The LED will light up, the terminal will print “Right”,

You can press “Ctrl + C” to end the program.

If there are issues with the test, the infrared device may need to be adjusted. The adjustment screw is on top of the sensor.



The code is as below:

```
from Infrared_Obstacle_Avoidance import *
def test_Infrared():
    try:
        while True:
            if GPIO.input(IR01)!=True and GPIO.input(IR02)==True and GPIO.input(IR03)!=True:
                print 'Middle'
            elif GPIO.input(IR01)!=True and GPIO.input(IR02)!=True and GPIO.input(IR03)==True:
                print 'Right'
            elif GPIO.input(IR01)==True and GPIO.input(IR02)!=True and GPIO.input(IR03)!=True:
                print 'Left'
    except KeyboardInterrupt:
        print "\nEnd of program"
```

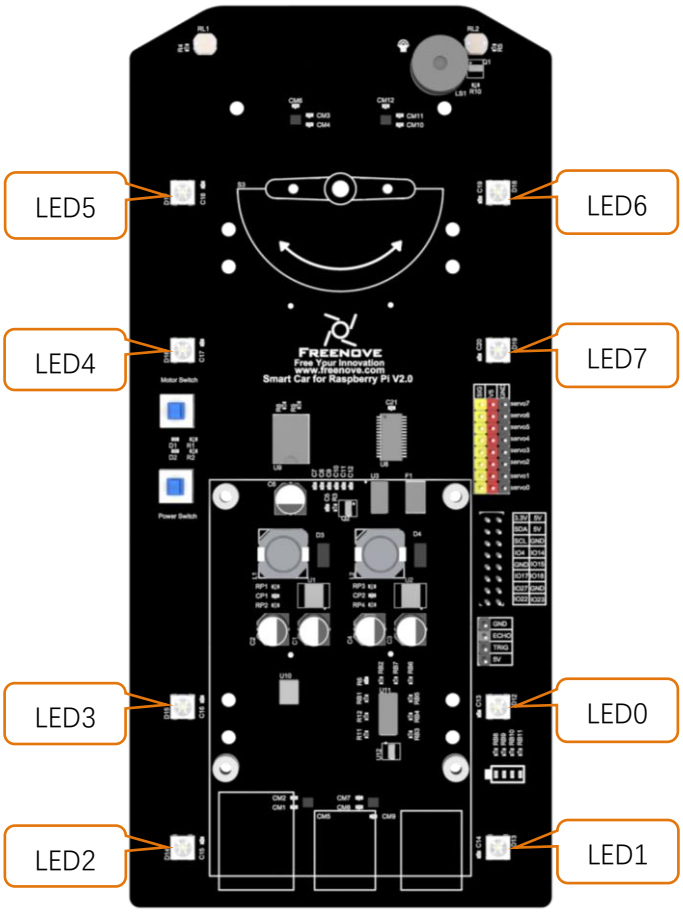
Reference

GPIO.input(IO)

This function has an input parameter. If the IO input is high level, GPIO.input(IO) returns True. If the IO input is low level, GPIO.input(IO) returns False.

LED

There are 8 RGB LEDs on the smart car board, as shownbelow. You can control them separately.



Run program

Enter the following commands to test LEDs.

If the terminal displays the directory as below (where test.py is located), you can **directly** execute the test.py command.

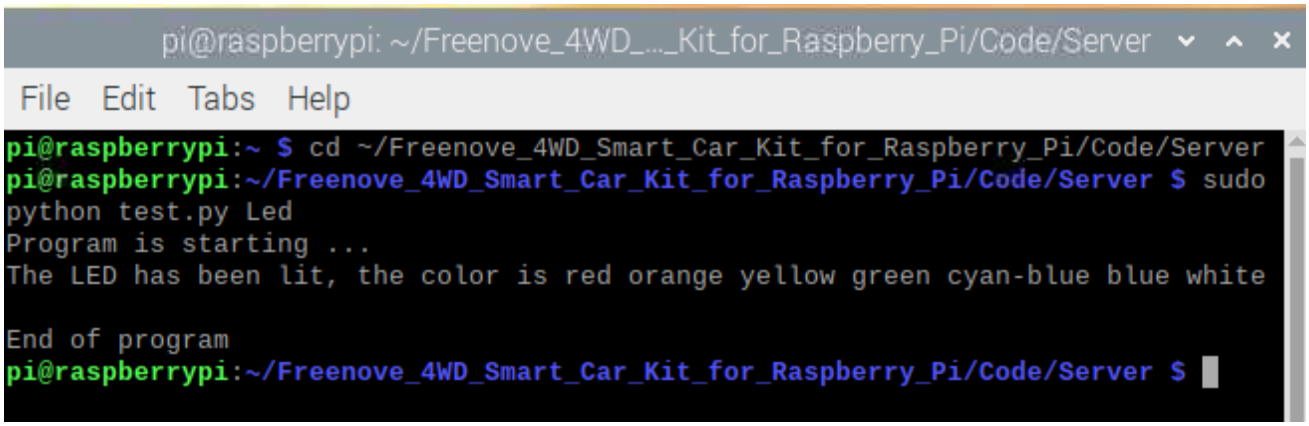
```
pi@raspberrypi:~/Freenove_4WD_Smart_Car_Kit_for_Raspberry_Pi/Code/Server $
```

1.If not, execute the cd command:

```
cd ~/Freenove_4WD_Smart_Car_Kit_for_Raspberry_Pi/Code/Server
```

2.Execute test.py command:

```
sudo python test.py Led
```



Result:

All LEDs will be turned on for 3 seconds, and colors from LED0 to LED7 are: red, orange, yellow, green, cyan, blue, purple, and white. You can end the program ahead of time by pressing "ctrl+c".

If the LED color display order is not correct, open the "**Led.py**" file in the current directory and modify the value of the "self.ORDER" variable on line 15.

The code of test.py is as below:

```
1  import time
2  try:
3      from Led import *
4      led=Led()
5      def test_Led():
6          try:
7              led.ledIndex(0x01,255,0,0)       #Red
8              led.ledIndex(0x02,255,125,0)     #orange
9              led.ledIndex(0x04,255,255,0)     #yellow
10             led.ledIndex(0x08,0,255,0)       #green
11             led.ledIndex(0x10,0,255,255)     #cyan-blue
12             led.ledIndex(0x20,0,0,255)       #blue
13             led.ledIndex(0x40,128,0,128)     #purple
14             led.ledIndex(0x80,255,255,255)   #white
15             print "The LED has been lit, the color is red orange yellow green cyan-blue blue
16 white"
17             time.sleep(3)                #wait 3s
18             led.colorWipe(led.strip, Color(0,0,0))  #turn off the light
19             print "\nEnd of program"
20         except KeyboardInterrupt:
21             led.colorWipe(led.strip, Color(0,0,0))  #turn off the light
22             print "\nEnd of program"
23 except:
24     pass
```

Reference

ledIndex(Index, R, G, B)
This function has 4 parameters. The first one is the index of the LED that you want to control. Its value is hexadecimal. There are LED0~7. The rest 3 parameters are R G B value of color respectively. For example, ledindex(0x01,255,0,0) makes LED 0 light to red; ledeindex(0x40,0,255,0) makes LED 6 light green.

colorWipe(strip, color, wait_ms)
This function erases the color of one pixel at a time. It has three input parameters: strip represents the Neopixel object, color represents the color to be erased, and wait_ms represents the erasure interval. The default is 50ms. For example, colorWipe(strip, Color(255,0,0),20) means that the LED0 is red first, wait for 20ms, and then the LED1 is also red, until all eight LEDs are lit and red.

LED Show

Now we add some algorithms in this chapter to make the LED display more styles. You can take this as a reference, then you can use your imagination to write your own algorithm to achieve the LED styles you want.

Run Program

If the terminal displays the directory as below, you can **directly** run the Led.py.

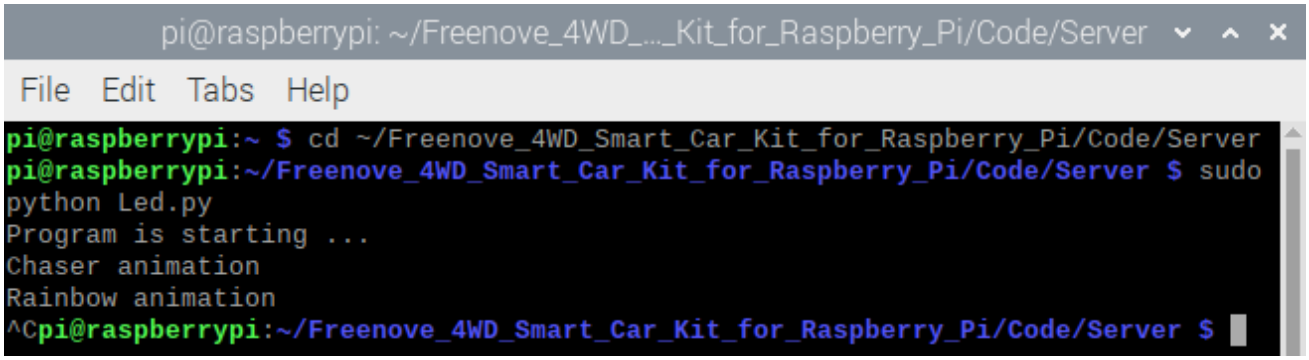
```
pi@raspberrypi:~/Freenove_4WD_Smart_Car_Kit_for_Raspberry_Pi/Code/Server $
```

1.If not, execute the cd command:

```
cd ~/Freenove_4WD_Smart_Car_Kit_for_Raspberry_Pi/Code/Server
```

2.Run Led.py:

```
sudo python Led.py
```



You can press "Ctrl + C" to end the program.

Part of code is as below:

```
1  # -*-coding: utf-8 -*-
2  import time
3  from rpi_ws281x import *
4  # LED strip configuration:
5  LED_COUNT      = 8      # Number of LED pixels.
6  LED_PIN        = 18      # GPIO pin connected to the pixels (18 uses PWM!).
7  LED_FREQ_HZ    = 800000  # LED signal frequency in hertz (usually 800khz)
8  LED_DMA        = 10      # DMA channel to use for generating signal (try 10)
9  LED_BRIGHTNESS = 255     # Set to 0 for darkest and 255 for brightest
10 LED_INVERT     = False   # True to invert the signal (when using NPN transistor level shift)
11 LED_CHANNEL    = 0       # set to '1' for GPIOs 13, 19, 41, 45 or 53
12 # Define functions which animate LEDs in various ways.
13 class Led:
14     def __init__(self):
15         self.ORDER = "GRB"  #Control the sending order of color data
16         # Create NeoPixel object with appropriate configuration.
17         self.strip = Adafruit_NeoPixel(LED_COUNT, LED_PIN, LED_FREQ_HZ, LED_DMA, LED_INVERT,
18 LED_BRIGHTNESS, LED_CHANNEL)
19         # Intialize the library (must be called once before other functions).
```

```
20          self.strip.begin()
21      def LED_TYPR(self,order,R_G_B):
22          B=R_G_B & 255
23          G=R_G_B >> 8 & 255
24          R=R_G_B >> 16 & 255
25          Led_type=["GRB","GBR","RGB", "RBG","BRG","BGR"]
26          color =
27  [Color(G,R,B),Color(G,B,R),Color(R,G,B),Color(R,B,G),Color(B,R,G),Color(B,G,R)]
28          if order in Led_type:
29              return color[Led_type.index(order)]
30      def colorWipe(self,strip, color, wait_ms=50):
31          """Wipe color across display a pixel at a time."""
32          color=self.LED_TYPR(self.ORDER,color)
33          for i in range(self.strip.numPixels()):
34              self.strip.setPixelColor(i, color)
35              self.strip.show()
36              time.sleep(wait_ms/1000.0)
37      def wheel(self,pos):
38          """Generate rainbow colors across 0-255 positions."""
39          if pos<0 or pos >255:
40              r=g=b=0
41          elif pos < 85:
42              r=pos * 3
43              g=255 - pos * 3
44              b=0
45          elif pos < 170:
46              pos -= 85
47              r=255 - pos * 3
48              g=0
49              b=pos * 3
50          else:
51              pos -= 170
52              r=0
53              g=pos * 3
54              b=255 - pos * 3
55          return self.LED_TYPR(self.ORDER,Color(r,g,b))
56      def rainbow(self,strip, wait_ms=20, iterations=1):
57          """Draw rainbow that fades across all pixels at once."""
58          for j in range(256*iterations):
59              for i in range(self.strip.numPixels()):
60                  self.strip.setPixelColor(i, self.wheel((i+j) & 255))
61              self.strip.show()
62              time.sleep(wait_ms/1000.0)
63      def rainbowCycle(self,strip, wait_ms=20, iterations=5):
```

```
64          """Draw rainbow that uniformly distributes itself across all pixels."""
65          for j in range(256*iterations):
66              for i in range(self.strip.numPixels()):
67                  self.strip.setPixelColor(i, self.wheel((int(i * 256 / self.strip.numPixels())
68  + j) & 255))
69              self.strip.show()
70              time.sleep(wait_ms/1000.0)
71      def theaterChaseRainbow(self,strip, wait_ms=50):
72          """Rainbow movie theater light style chaser animation."""
73          for j in range(256):
74              for q in range(3):
75                  for i in range(0, self.strip.numPixels(), 3):
76                      self.strip.setPixelColor(i+q, self.wheel((i+j) % 255))
77                  self.strip.show()
78                  time.sleep(wait_ms/1000.0)
79                  for i in range(0, strip.numPixels(), 3):
80                      strip.setPixelColor(i+q, 0)
81  led=Led()
82  # Main program logic follows:
83  if __name__ == '__main__':
84      print ('Program is starting ... ')
85      try:
86          while True:
87              print "Chaser animation"
88              led.colorWipe(led.strip, Color(255,0, 0))  # Red wipe
89              led.colorWipe(led.strip, Color(0, 255, 0))  # Green wipe
90              led.colorWipe(led.strip, Color(0, 0, 255))  # Blue wipe
91              led.theaterChaseRainbow(led.strip)
92              print "Rainbow animation"
93              led.rainbow(led.strip)
94              led.rainbowCycle(led.strip)
95              led.colorWipe(led.strip, Color(0,0,0),10)
96      except KeyboardInterrupt:  # When 'Ctrl+C' is pressed, the child program destroy() will be
97  executed.
98          led.colorWipe(led.strip, Color(0,0,0),10)
```

Reference

strip.setPixelColor(Index,color(R,G,B))

This is a function of WS2812 library. It is the same as the previously customized ledIndex() function. It is used to light up one LED and it has two input parameters. The first one is the LED number, the second one is used to set the color of the LED. For example, strip.setPixelColor(1,Color(255, 0, 0)), and write strip.show() in the next line, then LED1 will light red.

strip.show()

This function is of WS2812 library. When the LED color is set with the previous fuction, this function needs to be executed to make the LED show the corresponding color. If the color is set, but this function is not executed LED will not change color.

wheel(pos)
Generate rainbow colors in range of 0-255.

LED_TYPR(self,order,R_G_B)
Change the order in which the LED color data is transmitted. When the value of the order parameter is "RGB", the order of data transmission should be: R-G-B; when the value of the order parameter is "GBR", and the order of data transmission should be: G-B-R

theaterChaseRainbow(strip, wait_ms)
The function is used to make 8 LEDs show one color at the same time, and change to various colors to make a **blink**. The blinking interval is wait_ms, and its default value is 50ms.

rainbow(strip, wait_ms,)
This function achieves the effect of rainbow **breathing**. It makes 8 LEDs display **same** color simultaneously, and then change them all into various colors like breathing. The interval is wait_ms. The default value is 20ms.

rainbowCycle(strip, wait_ms)
This function also achieves the effect of rainbow **breathing**. but unlike rainbow(), it makes eight LEDs to display **different** colors at the same time, and then change them into various color separately. The interval is wait_ms. The default value is 20ms.

Result analysis

This code mainly achieves two LED effects, chasing animation and rainbow animation.

Chasing animation: first let the 8 LEDs light red one by one in turn, then green and blue. Interval is 50ms between two LED, so the LED will display a round of red, then a round of green, and the last round of blue, like chasing. Then let the LEDs blink with different colors with an interval of 50ms, rendering a tense atmosphere, thus completing the chase animation.

Rainbow animation: The effect of the rainbow is different from the effect of blinking. The blinking is to make the LED on, off, on, and off. And the rainbow is to make LED on all the time, and switch between different colors, and the interval is shorter than the blinking. First, make the eight LEDs display one color at the same time and then change the color with intervals of 20ms. And then make the eight LEDs display different colors at the same time, and then change the color to produce another rainbow effect.

Buzzer

Run the program

Enter the following command in the terminal to test buzzer.

If the terminal displays the directory as below (where test.py is located). You can **directly** execute the test.py command.

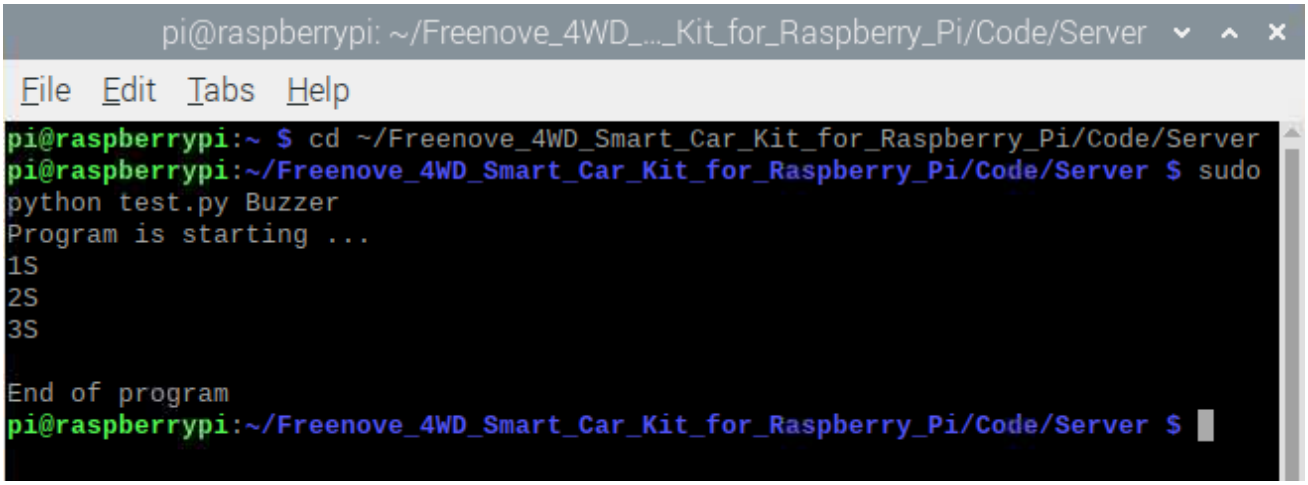
```
pi@raspberrypi:~/Freenove_4WD_Smart_Car_Kit_for_Raspberry_Pi/Code/Server $
```

1 If not, execute the cd command:

```
cd ~/Freenove_4WD_Smart_Car_Kit_for_Raspberry_Pi/Code/Server
```

2 Execute test.py command:

```
sudo python test.py Buzzer
```



Result:

The buzzer will be turned on and last for 3s. Then the program will automatically end or you can press "Ctrl + C" to end the program.

The code is as below:

```
1  from Buzzer import *
2  buzzer=Buzzer()
3  def test_Buzzer():
4      try:
5          buzzer.run(cmd.CMD_START)
6          time.sleep(1)
7          print "1S"
8          time.sleep(1)
9          print "2S"
10         time.sleep(1)
11         print "3S"
12         buzzer.run(cmd.CMD_STOP)
13         print "\nEnd of program"
```

```
14      except KeyboardInterrupt:
15          buzzer.run(cmd.CMD_STOP)
16          print "\nEnd of program"
```

Reference

buzzer.run(cmd)

This function has one input parameter. If the input is '1', the buzzer will be turned on. If the input is '0', the buzzer will be turned off.

Servo

Enter the following commands in the terminal to test servos.

If the terminal displays the directory as below (where test.py is located), you can **directly** execute the test.py command.

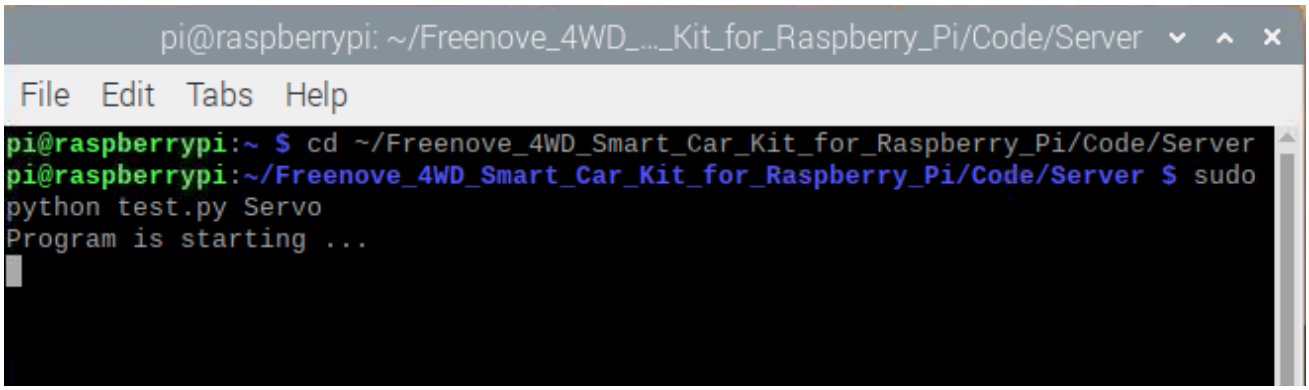
```
pi@raspberrypi:~/Freenove_4WD_Smart_Car_Kit_for_Raspberry_Pi/Code/Server $
```

1. If not, execute the cd command:

```
cd ~/Freenove_4WD_Smart_Car_Kit_for_Raspberry_Pi/Code/Server
```

2. Execute test.py command:

```
sudo python test.py Servo
```



Result:

The servo 0 repeats rotating from left to right and then from right to left. The servo 1 repeats rotating from bottom to top and then from top to bottom. You can press "Ctrl + C" to end the program.

The code is as below:

```
1  from servo import *
2  pwm=Servo()
3  def test_Servo():
4      try:
5          while True:
6              for i in range(50,110,1):
7                  pwm.setServoPwm('0',i)
8                  time.sleep(0.01)
9              for i in range(110,50,-1):
10                 pwm.setServoPwm('0',i)
11                 time.sleep(0.01)
12             for i in range(80,150,1):
13                 pwm.setServoPwm('1',i)
14                 time.sleep(0.01)
15             for i in range(150,80,-1):
16                 pwm.setServoPwm('1',i)
17                 time.sleep(0.01)
18     except KeyboardInterrupt:
```

```
19          pwm.setServoPwm('0',90)
20          pwm.setServoPwm('1',90)
21          print "\nEnd of program"
```

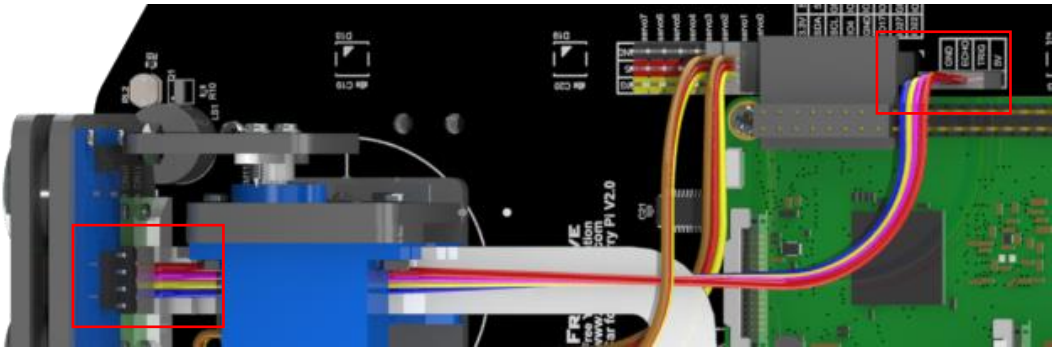
Reference

setServoPwm(Servo,angle)

There are 2 parameters.
The first one is related to servo index.
The second one is related to the angle of servos.
For example,
setServoPwm('0',20) makes servo0 rotate to 20°.
setServoPwm('1',90) makes servo1 rotate to 90°.

Ultrasonic module

Next, use jumper wires F/F to connect ultrasonic module with pins on smart car board.



When connecting the ultrasonic module, you need **disconnect** the **Servo1** cable, so that the servo can rotate freely, and after the wiring is completed, connect the servo cable again. When wiring, you should keep the silk screen of the ultrasonic module and the smart car board consistent. Vcc should be connected to 5V, Trig to TRIG, Echo to ECHO, and Gnd to GND.
If the connection is wrong, for example, if Vcc is connected to GND, and Gnd is connected to 5V, it will cause the damage to ultrasonic module. After the wiring is completed, you can start testing.

Run program

Enter following command in the terminal:
If the terminal displays the directory as below (where test.py is located). You can **directly** execute the test.py command.

```
pi@raspberrypi:~/Freenove_4WD_Smart_Car_Kit_for_Raspberry_Pi/Code/Server $
```

1. If not, execute the cd command:

```
cd ~/Freenove_4WD_Smart_Car_Kit_for_Raspberry_Pi/Code/Server
```

2. Execute test.py command:

```
sudo python test.py Ultrasonic
```

```
pi@raspberrypi: ~/Freenove_4WD_..._Kit_for_Raspberry_Pi/Code/Server
File  Edit  Tabs  Help
pi@raspberrypi:~ $ cd ~/Freenove_4WD_Smart_Car_Kit_for_Raspberry_Pi/Code/Server
pi@raspberrypi:~/Freenove_4WD_Smart_Car_Kit_for_Raspberry_Pi/Code/Server $ sudo
python test.py Ultrasonic
Program is starting ...
Obstacle distance is 21CM
Obstacle distance is 596CM
Obstacle distance is 660CM
Obstacle distance is 7CM
Obstacle distance is 3CM
Obstacle distance is 285CM
Obstacle distance is 7CM
Obstacle distance is 8CM
Obstacle distance is 7CM
Obstacle distance is 110CM
Obstacle distance is 363CM
Obstacle distance is 353CM
^C
End of program
pi@raspberrypi:~/Freenove_4WD_Smart_Car_Kit_for_Raspberry_Pi/Code/Server $
```

Result:

Every 1s, the distance between the obstacle and the ultrasonic module will be printed out, and you can press "Ctrl + C" to end the program.

The code is as below:

```
1  from Ultrasonic import *
2  ultrasonic=Ultrasonic()
3  def test_Ultrasonic():
4      try:
5          while True:
6              data=ultrasonic.get_distance()   #Get the value
7              print ("Obstacle distance is "+str(data)+"CM")
8              time.sleep(1)
9      except KeyboardInterrupt:
10         print "\nEnd of program"
```

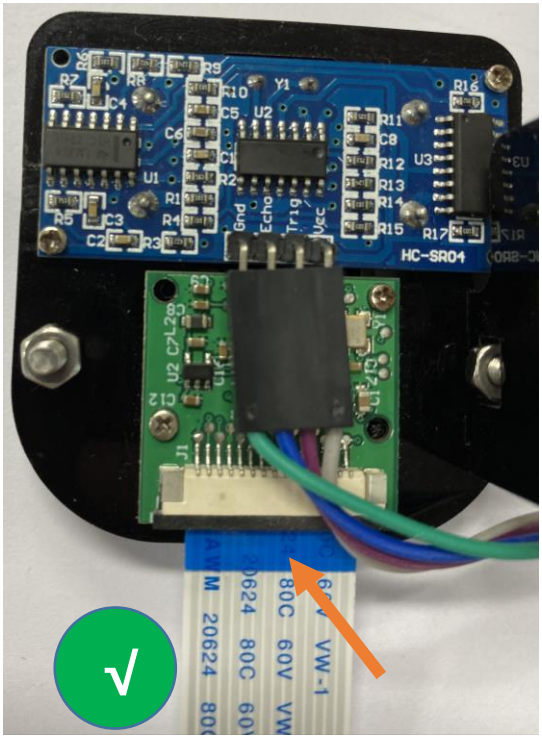
Reference

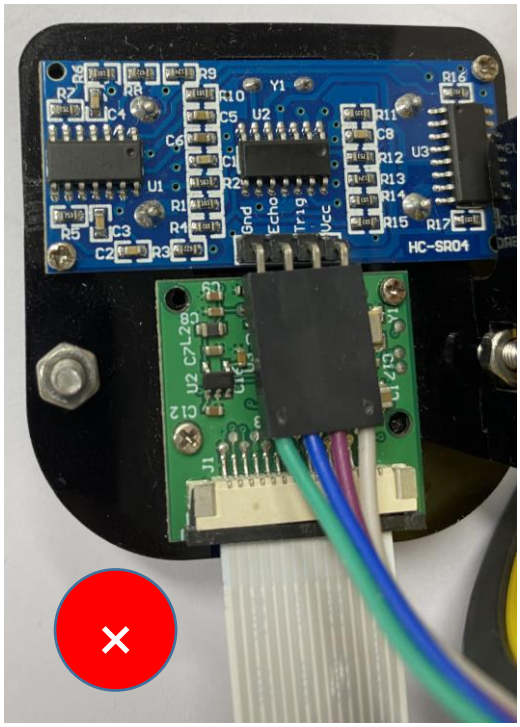
get_distance()
This function is used to obtain the distance between ultrasonic module and obstacles in front of it, with unit CM.

Camera

Next let us connect the camera to smart car board. First **turn off S1** (Power Switch), **shut down Raspberry Pi** and disconnect power cable. If the data cable is used to power the Raspberry Pi, disconnect the data cable and install the CSI camera to the Raspberry Pi camera interface when the Raspberry Pi is powered off. (**The CSI camera must be connected or disconnected under no power and when Raspberry Pi is shut down, or the camera may be burned.)**

Step 1

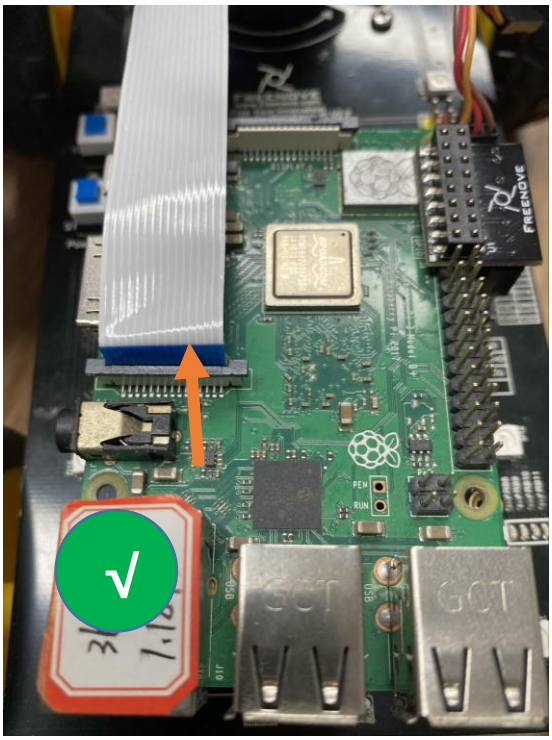




The Blue side of cable should be toward to Servo.

Connect one end of cable to camera. Please note the front and back of the cable.

Step 2



The Blue side of cable should be toward to RPi USB port.

Connect another end of cable to Raspberry Pi. Please note the front and back of the cable.

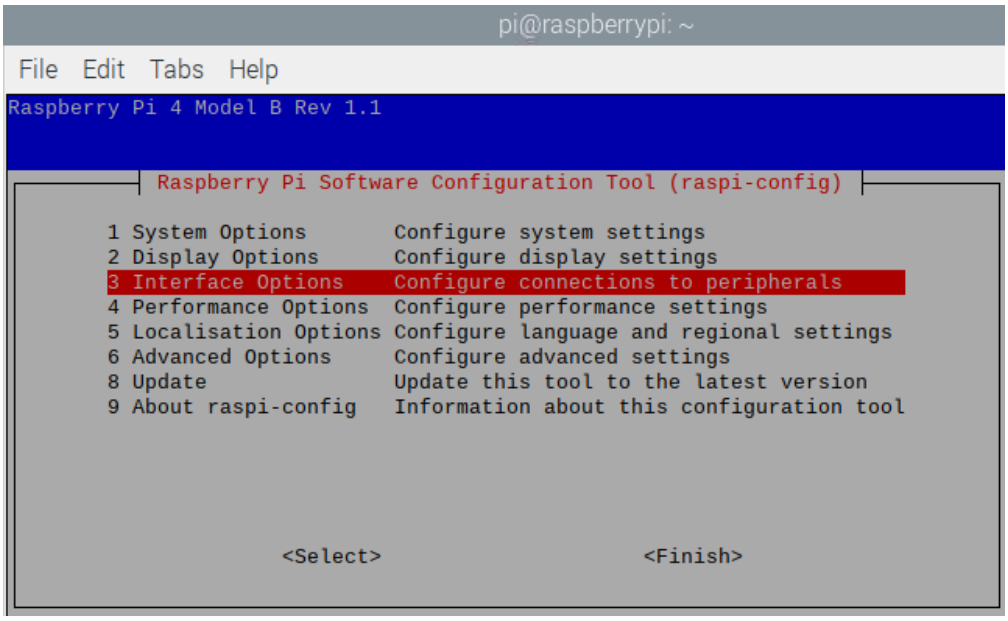
Run program

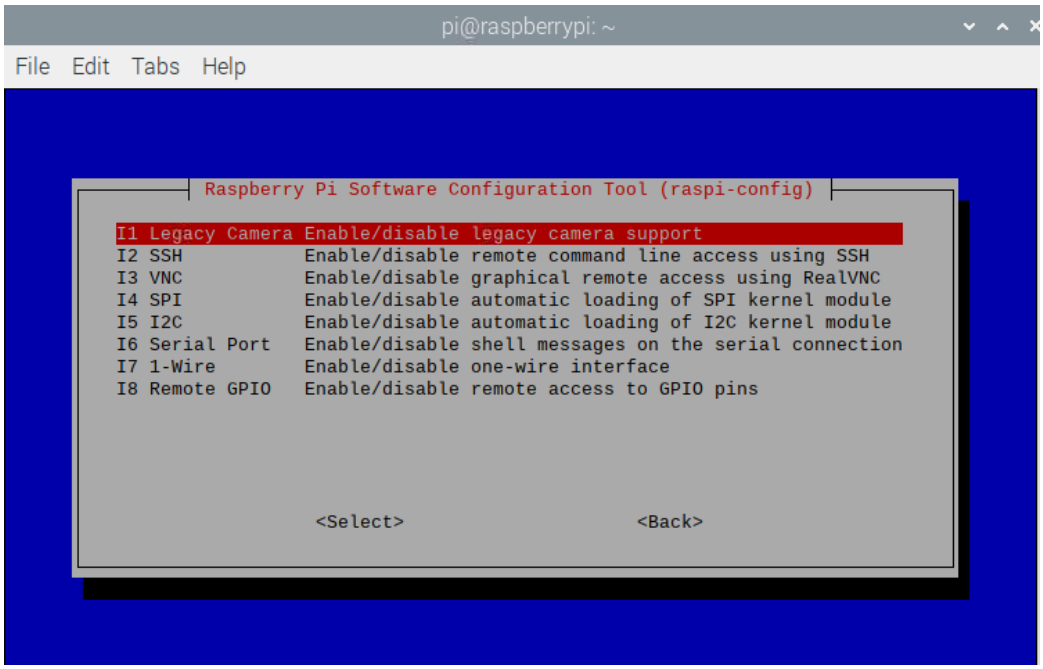
To use the camera, you need to disable legacy camera, which is disabled by default on the latest Raspberry Pi OS. If it is not disabled, please do it as below.

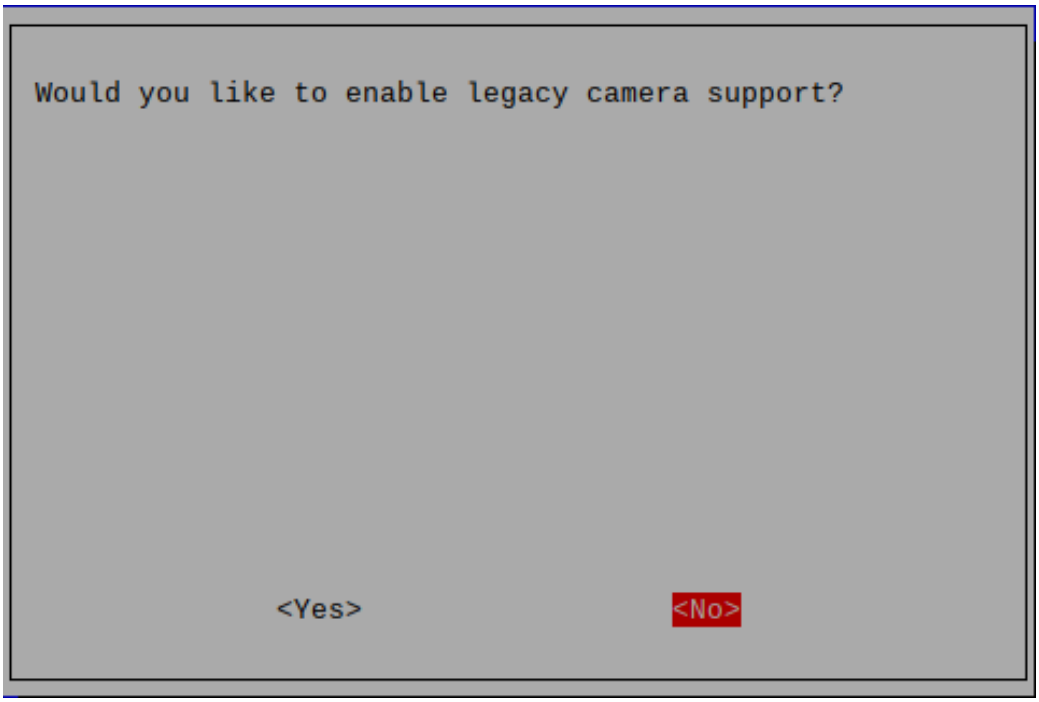
Enter the following command. Choose **Interface Options→ Legacy Camera →No→OK→Finish,**and then restart the Raspberry Pi.

```
sudo raspi-config
```

```
pi@raspberrypi:~ $ sudo raspi-config
```



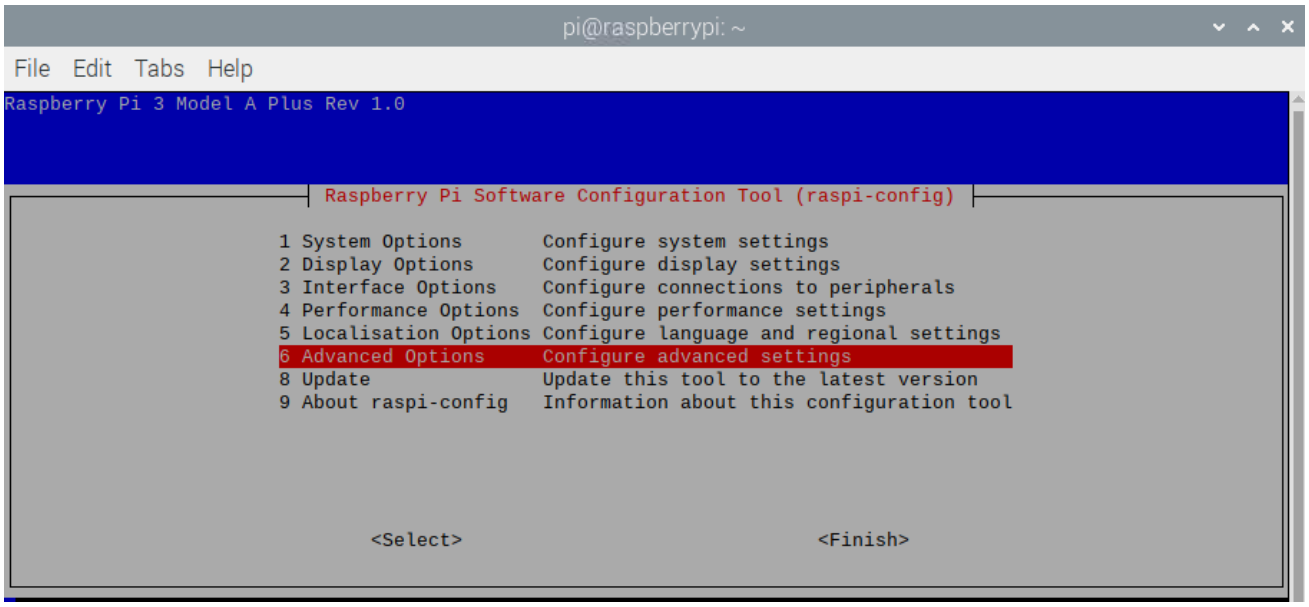

pi@raspberrypi: ~
File Edit Tabs Help
Raspberry Pi Software Configuration Tool (raspi-config)
I1 Legacy Camera Enable/disable legacy camera support
I2 SSH Enable/disable remote command line access using SSH
I3 VNC Enable/disable graphical remote access using RealVNC
I4 SPI Enable/disable automatic loading of SPI kernel module
I5 I2C Enable/disable automatic loading of I2C kernel module
I6 Serial Port Enable/disable shell messages on the serial connection
I7 1-Wire Enable/disable one-wire interface
I8 Remote GPIO Enable/disable remote access to GPIO pins
<Select>
<Back>

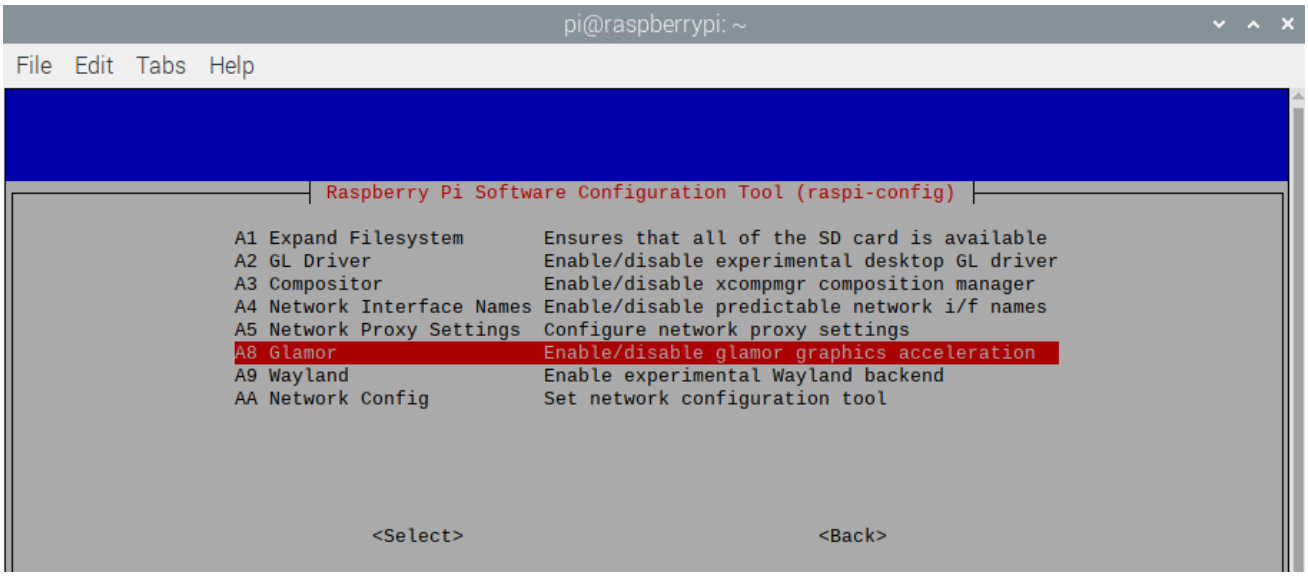

Would you like to enable legacy camera support?
<Yes>
<No>

libcamera-apps does not work properly on Pi 0 to 3 devices when running the latest Bullseye images. A workaround is to open a terminal, run "**sudo raspi-config**", navigate to "**Advanced Options**" and enable "**Glamor**" graphic acceleration. Then reboot your Pi.

```
sudo raspi-config
```

```
pi@raspberrypi:~ $ sudo raspi-config
```





Then reboot your Pi.

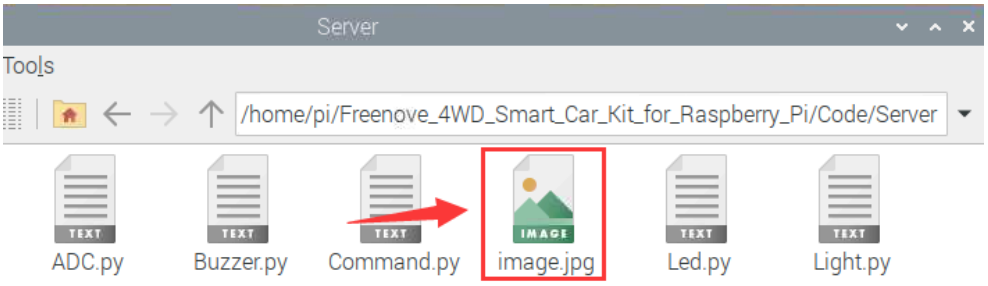
1. execute the cd command:

```
cd ~/Freenove_4WD_Smart_Car_Kit_for_Raspberry_Pi/Code/Server
```

2. Execute the following command:

```
python camera.py
```

Then please open and check the generated image.jpg under /Freenove_4WD_Smart_Car_Kit_for_Raspberry_Pi/Code/Server.



Chapter 4 Light tracing Car

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

Description

The light-tracing function of the car mainly uses a photoresistor. The car has two photoresistors located on the left and right sides at the front to detect light

A photoresistor is a resistor based on the photoelectric effect of the semiconductor. The resistance changes with the intensity of the incident light. With the incident light intensity increasing, the resistance decreases. With the incident light intensity decreasing, the resistance increases.

And the change of the resistance value also causes voltage applied to the photoresistor changes. According to the change of voltage, the position of the light to the car will be detected, and then make the car move corresponding action to trace light.

Put your car in a darker environment.

Run program

If the terminal displays the directory as below, you can **directly** execute the Light.py command.

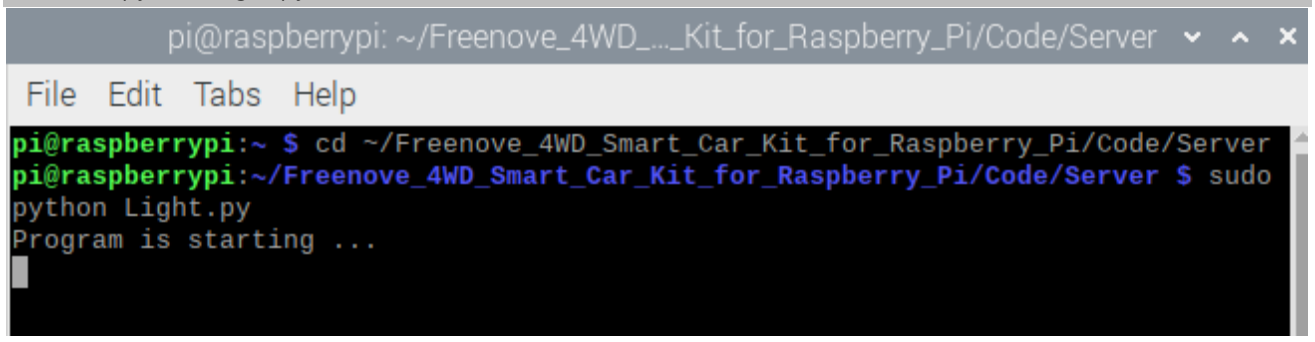
```
pi@raspberrypi:~/Freenove_4WD_Smart_Car_Kit_for_Raspberry_Pi/Code/Server $
```

1. If not, execute the cd command:

```
cd ~/Freenove_4WD_Smart_Car_Kit_for_Raspberry_Pi/Code/Server
```

2. Run Light.py:

```
sudo python Light.py
```



You can press "Ctrl + C" to end the program.

The code is below:

```
import time
from Motor import *
from ADC import *
class Light:
    def run(self):
        try:
            self.adc=Adc()
            self.PWM=Motor()
            self.PWM.setMotorModel(0,0,0,0)
            while True:
                L = self.adc.recvADC(0)
                R = self.adc.recvADC(1)
                if L < 2.99 and R < 2.99 :
                    self.PWM.setMotorModel(600,600,600,600)

                elif abs(L-R)<0.15:
                    self.PWM.setMotorModel(0,0,0,0)

                elif L > 3 or R > 3:
                    if L > R :
                        self.PWM.setMotorModel(-1200,-1200,1400,1400)

                    elif R > L :
                        self.PWM.setMotorModel(1400,1400,-1200,-1200)

        except KeyboardInterrupt:
            led_Car.PWM.setMotorModel(0,0,0,0)

if __name__=='__main__':
    print ('Program is starting ... ')
    led_Car=Light()
    led_Car.run()
```

Result analysis

When the voltages of left and right photoresistor are less than 2.99V, the car move forward straightly. And when one of the voltages is greater than 3V:

If the left voltage is greater than the right, the car turns left.

If the right voltage is greater than the left, the car turns right.

You can change the judgment of the program to achieve the result you want, according to the light intensity of the environment.

Chapter 5 Ultrasonic Obstacle Avoidance Car

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

Description

The obstacle avoidance function of the car mainly uses the HC-SR04 ultrasonic module. The ultrasonic module is controlled by the servo. The servo rotates to the left, middle and right repeatedly, so that the ultrasonic module measures the distance of obstacles on the left, middle and right directions. And then it controls the car to move according to different distances.

Run program

If the terminal displays the directory as below, you can **directly** run the Ultrasonic.py.

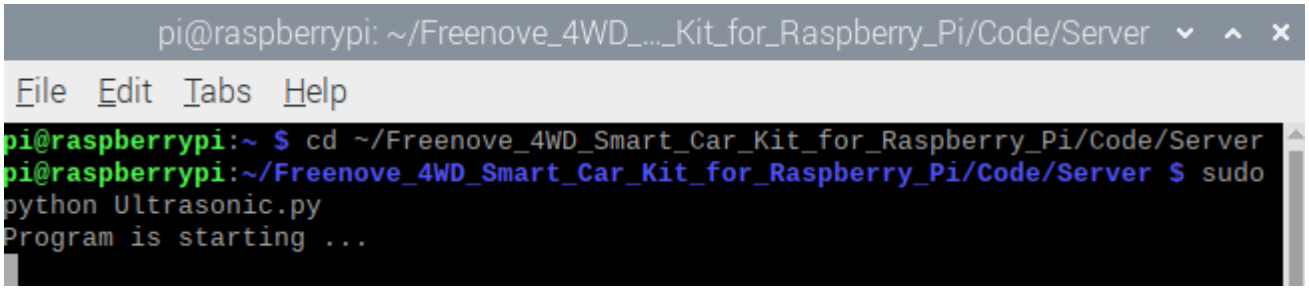
```
pi@raspberrypi:~/Freenove_4WD_Smart_Car_Kit_for_Raspberry_Pi/Code/Server $
```

1. If not, execute the cd command:

```
cd ~/Freenove_4WD_Smart_Car_Kit_for_Raspberry_Pi/Code/Server
```

2. Run Ultrasonic.py:

```
sudo python Ultrasonic.py
```



You can press "Ctrl + C" to end the program.

Part of code is as below:

```
1     def run(self):
2         self.PWM=Motor()
3         self.pwm_S=Servo()
4         for i in range(30,151,60):
5                 self.pwm_S.setServoPwm('0',i)
6                 time.sleep(0.2)
7                 if i==30:
8                     L = self.get_distance()
9                 elif i==90:
10                    M = self.get_distance()
11                else:
12                    R = self.get_distance()
13        while True:
```

```
            for i in range(90,30,-60):
                self.pwm_S.setServoPwm('0',i)
                time.sleep(0.2)
                if i==30:
                    L = self.get_distance()
                elif i==90:
                    M = self.get_distance()
                else:
                    R = self.get_distance()
                self.run_motor(L,M,R)
            for i in range(30,151,60):
                self.pwm_S.setServoPwm('0',i)
                time.sleep(0.2)
                if i==30:
                    L = self.get_distance()
                elif i==90:
                    M = self.get_distance()
                else:
                    R = self.get_distance()
                self.run_motor(L,M,R)
ultrasonic=Ultrasonic()
# Main program logic follows:
if __name__ == '__main__':
    print ('Program is starting ... ')
    try:
        ultrasonic.run()
    except KeyboardInterrupt:  # When 'Ctrl+C' is pressed, the child program destroy() will be
executed.
        PWM.setMotorModel(0,0,0,0)
        ultrasonic.pwm_S.setServoPwm('0',90)
```

Result analysis

Let servo0 rotate back and forth to 30 degrees, 90 degrees and 150 degrees respectively. And the ultrasonic module also follows the movement to measure the obstacle distance of these three angles.

When distances detected on the left>30cm, middle >30cm, right>30cm. It means that there is no obstacle within 30cm. So the car move forward.

When distances detected on the left<30cm, middle <30cm, right<30cm, it means that the car enters a dead end, so the car move back and turned back.

When distances detected on the left<30cm, middle <30cm, right>30cm. It means that there is an obstacle on the left side of the car, so the car turn right.

When distances detected on the left>30cm, middle <30cm, right<30cm. It means that there is an obstacle on the right side of the car, so the car turn left.

Chapter 6 Infrared Line Tracking Car

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

Description

The line tracing function of the car mainly uses an infrared module. When the sensor detects black line, the corresponding LED will light up, which controls the car to move according to the value of three sensors.

Run program

If the terminal displays the directory as below, you can **directly** run the program.

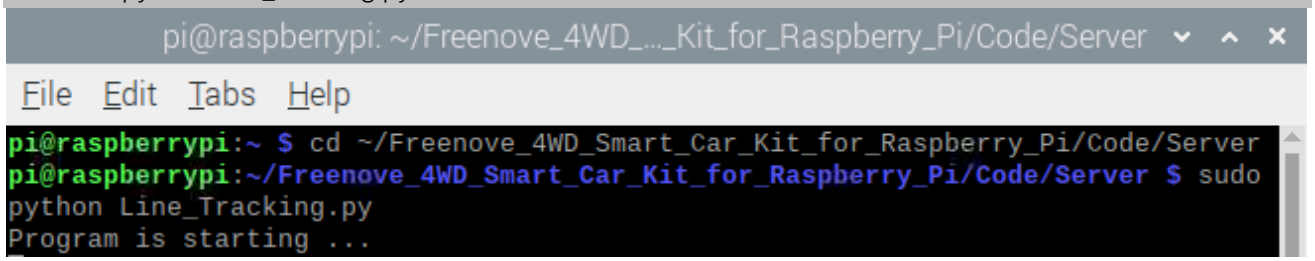
```
pi@raspberrypi:~/Freenove_4WD_Smart_Car_Kit_for_Raspberry_Pi/Code/Server $
```

1. If not, execute the cd command:

```
cd ~/Freenove_4WD_Smart_Car_Kit_for_Raspberry_Pi/Code/Server
```

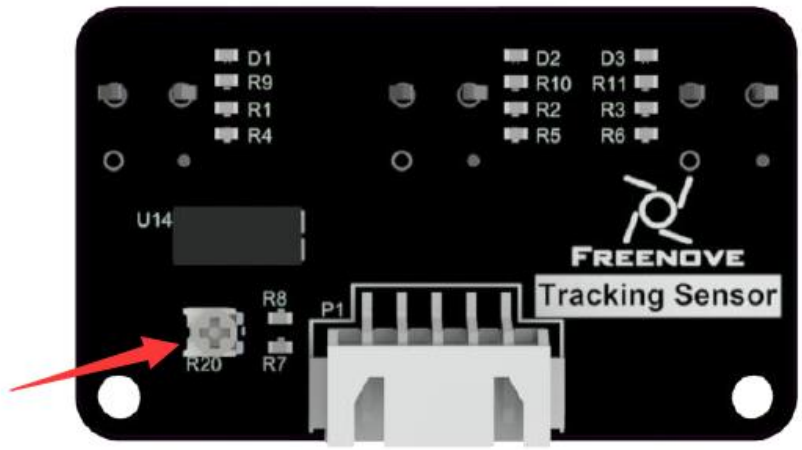
2. Run Line_Tracking.py:

```
sudo python Line_Tracking.py
```



You can press "Ctrl + C" to end the program.

If there are issues with the test, the infrared device may need to be adjusted. The adjustment screw is on top of the sensor.



The code is as below:

```
import time
from Motor import *
import RPi.GPIO as GPIO
IR01 = 14
IR02 = 15
IR03 = 23
GPIO.setmode(GPIO.BCM)
GPIO.setup(IR01,GPIO.IN)
GPIO.setup(IR02,GPIO.IN)
GPIO.setup(IR03,GPIO.IN)
class Line_Tracking:
    def run(self):
        while True:
            self.LMR=0x00
            if GPIO.input(IR01)==True:
                self.LMR=(self.LMR | 4)
            if GPIO.input(IR02)==True:
                self.LMR=(self.LMR | 2)
            if GPIO.input(IR03)==True:
                self.LMR=(self.LMR | 1)
            if self.LMR==2:
                PWM.setMotorModel(800,800,800,800)
            elif self.LMR==4:
                PWM.setMotorModel(-1500,-1500,2500,2500)
            elif self.LMR==6:
                PWM.setMotorModel(-2000,-2000,4000,4000)
            elif self.LMR==1:
                PWM.setMotorModel(2500,2500,-1500,-1500)
            elif self.LMR==3:
                PWM.setMotorModel(4000,4000,-2000,-2000)
            elif self.LMR==7:
                pass

infrared=Line_Tracking()
# Main program logic follows:
if __name__ == '__main__':
    print ('Program is starting ... ')
    try:
        infrared.run()
    except KeyboardInterrupt:  # When 'Ctrl+C' is pressed, the child program destroy() will be
executed.
        PWM.setMotorModel(0,0,0,0)
```

Result analysis

There are 3 sensors on the left, middle and right. When the black line is detected by a sensor, it will show high level, or it is low.

When the sensor on left: high, middle: low, right: low, the car turns left lightly.

When the sensor on left: high, middle: high, right: low, the car turns left.

When the sensor on left: low, middle: high, right: low, the car moves forward straight.

When the sensor on left: low, middle: low, right: high, the car turns right lightly.

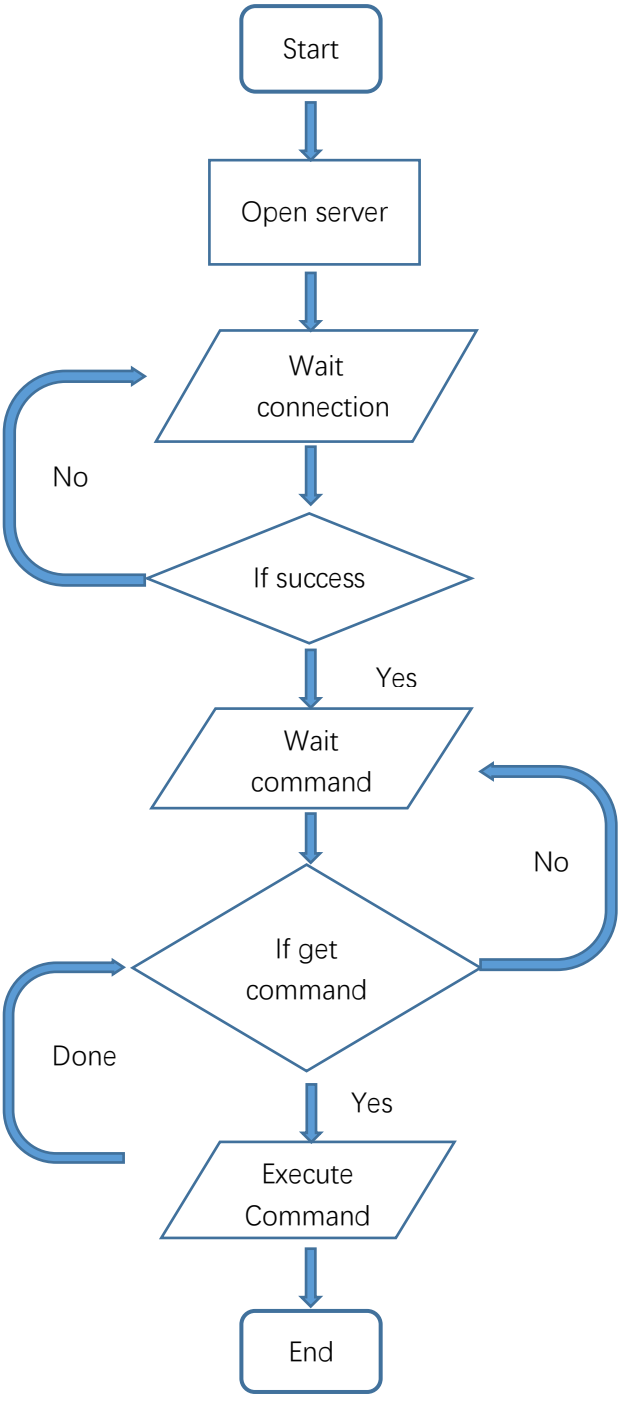
When the sensor on left: low, middle: high, right: high, the car turns right.

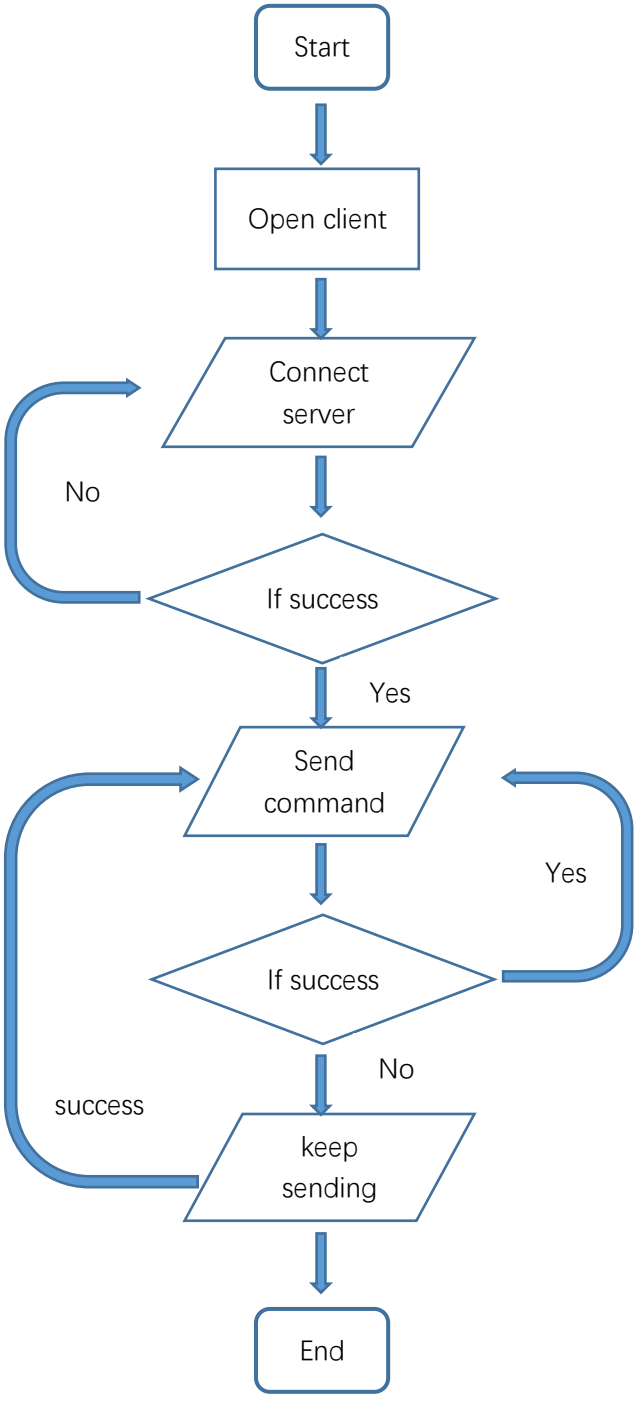
Chapter 7 Smart video car

You can refer to this video: https://youtu.be/3WH4QYPWN-I

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

The smart video car integrates the previous functions of light tracing, obstacle avoidance, line tracing, video transmission, face detection, LED and so on. And it is built with a server and a client, so it can be controlled remotely.





Server

The server works on the Raspberry Pi and can transmit camera data, ultrasonic data, etc. to the client, and it can also receive commands from the client.
In the Server folder, there is a server.py file which contains main server code.
get_interface_ip() is used to get IP address of the native Raspberry Pi wlan0, without manually modifying the code to set IP parameters.
StartTcpServer() is used to start the TCP service. The channel of port 5000 is mainly used to send and receive commands between the client and the server. The channel of port 8000 is used for the server to transmit the collected camera data to the client.
StopTcpServer() is used to stop the TCP service.
sendvideo() is used to sends the camera data.

Part of server code is as follows:

```
1  def get_interface_ip(self):
2          s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
3          return socket.inet_ntoa(fcntl.ioctl(s.fileno(), 0x8915, struct.pack('256s',
4                                  "wlan0"[:15]))[20:24])
5  def StartTcpServer(self):
6          HOST=str(self.get_interface_ip())
7          self.server_socket1 = socket.socket()
8          self.server_socket1.setsockopt(socket.SOL_SOCKET,socket.SO_REUSEPORT,1)
9          self.server_socket1.bind((HOST, 5000))
10         self.server_socket1.listen(1)
11         self.server_socket = socket.socket()
12         self.server_socket.setsockopt(socket.SOL_SOCKET,socket.SO_REUSEPORT,1)
13         self.server_socket.bind((HOST, 8000))
14         self.server_socket.listen(1)
15         print('Server address: '+HOST)
16
17 def StopTcpServer(self):
18         try:
19             self.connection.close()
20             self.connection1.close()
21         except Exception ,  e:
22             print "No client connection"
23
24 def sendvideo(self):
25         try:
26             self.connection,self.client_address = self.server_socket.accept()
27             self.connection=self.connection.makefile('rb')
28         except:
29             pass
```

```
30          self.server_socket.close()
31          try:
32              with picamera.PiCamera() as camera:
33                  camera.resolution = (400,300)       # pi camera resolution
34                  camera.framerate = 30               # 15 frames/sec
35                  time.sleep(2)                       # give 2 secs for camera to initilize
36                  start = time.time()
37                  stream = io.BytesIO()
38                  # send jpeg format video stream
39                  print "Start transmit ... "
40                  for foo in camera.capture_continuous(stream, 'jpeg', use_video_port = True):
41                      try:
42                          self.connection.flush()
43                          stream.seek(0)
44                          b = stream.read()
45                          lengthBin = struct.pack('L', len(b))
46                          self.connection.write(lengthBin)
47                          self.connection.write(b)
48                          if time.time() - start > 600:
49                              break
50                          stream.seek(0)
51                          stream.truncate()
52                      except :
53                          print "End transmit ... "
54                          break
55          except:
56              print "Camera unintall"
```

Open Server

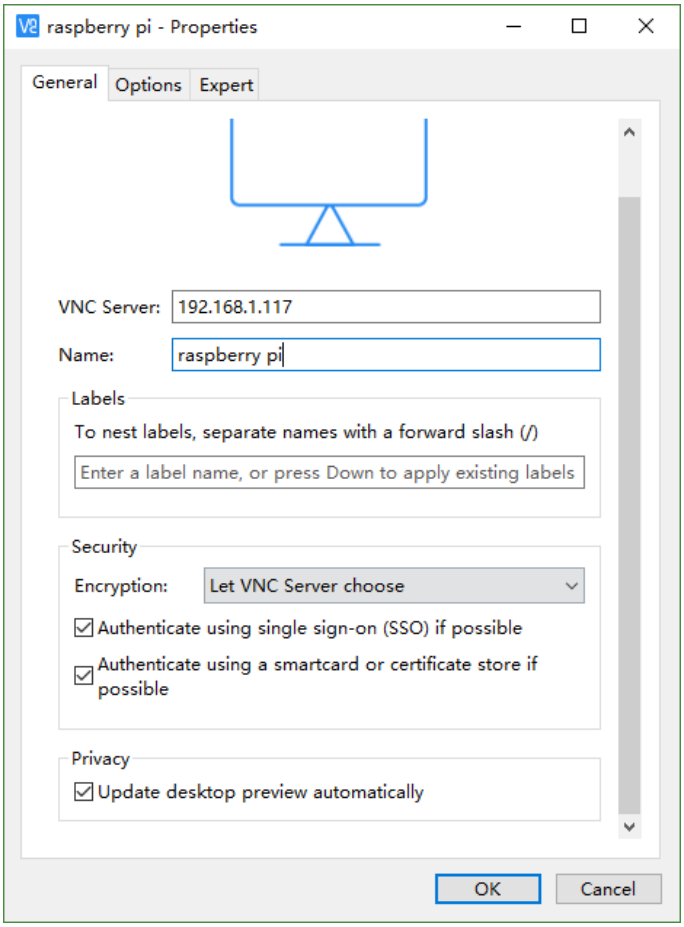
Step 1 Login Raspberry Pi via VNC viewer

Because server and client use GUI. You need use VNC viewer as remote desktop way.

Download and install VNC Viewer according to your computer system by clicking following link:

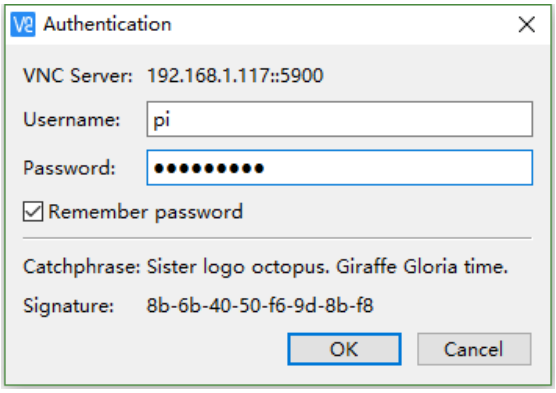
https://www.realvnc.com/en/connect/download/viewer/

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. And click OK.

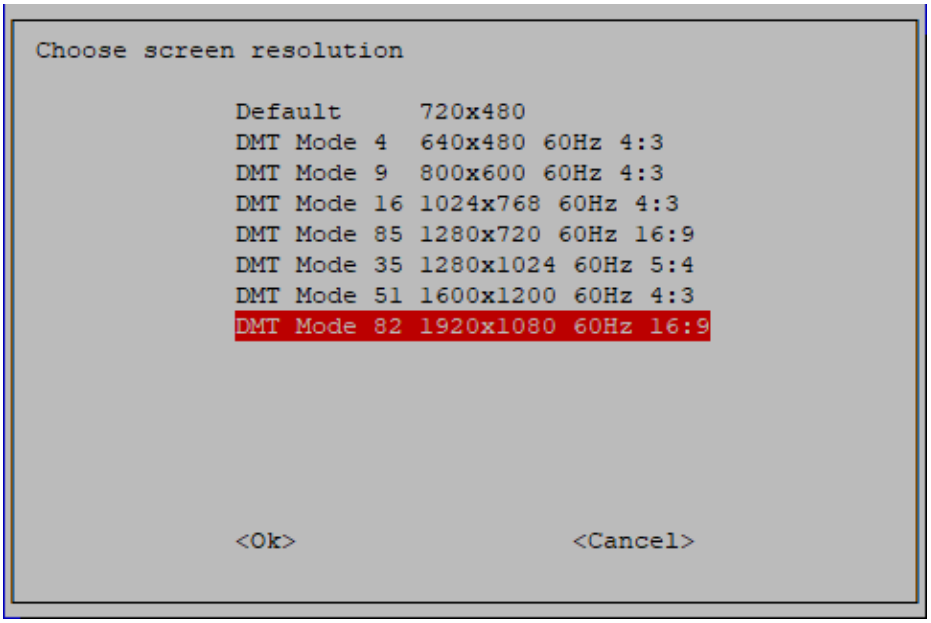
Then on the VNC Viewer panel, double-click new connection you just created, and the following dialog box pops up. Enter username: **pi** and Password: **raspberry**. And click OK.



If the resolution ratio is not great or there is just a **little window**, you can set a proper resolution ratio via steps below.

```
sudo raspi-config
```

Select 7 Advanced Options→A5 Resolution→proper resolution ratio (set by yourself)→OK→Finish. And then reboot Raspberry Pi.



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.

Step 2 Run commands

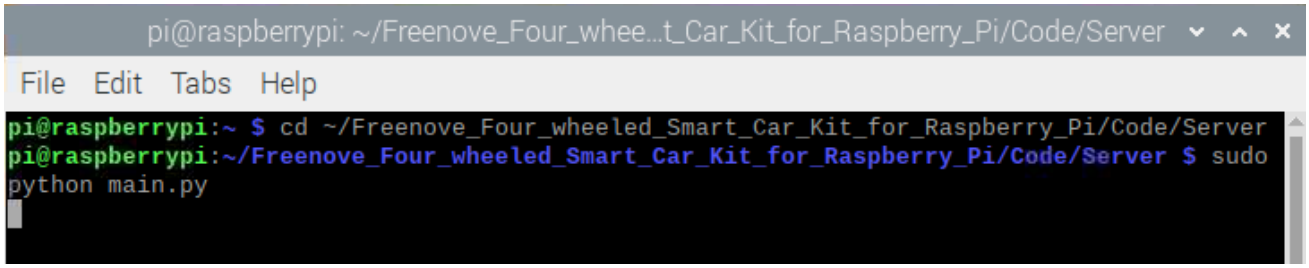
If you are using **remote desktop mode** to login Raspberry Pi, you need use VNC viewer.
Enter the following command in the terminal.

1. Use cd command to enter directory where main.py is located:

```
cd ~/Freenove_4WD_Smart_Car_Kit_for_Raspberry_Pi/Code/Server
```

2. Run main.py:

```
sudo python main.py
```



The interface is as below:

If you don't like the interface, you can also enter the commands to open the server. It is more convenient.

1. Use cd command to enter directory where main.py is located:

```
cd ~/Freenove_4WD_Smart_Car_Kit_for_Raspberry_Pi/Code/Server
```

2. Run main.py:

```
sudo python main.py -t -n
```

or Run main,py with following command:

```
sudo python main.py -tn
```

"-t" means open TCP communication. "-n" means don't show interface.

Sever Auto Start

1 Open the terminal and execute the following two commands respectively to create a "start.sh" file.

```
cd ~
sudo touch start.sh
```

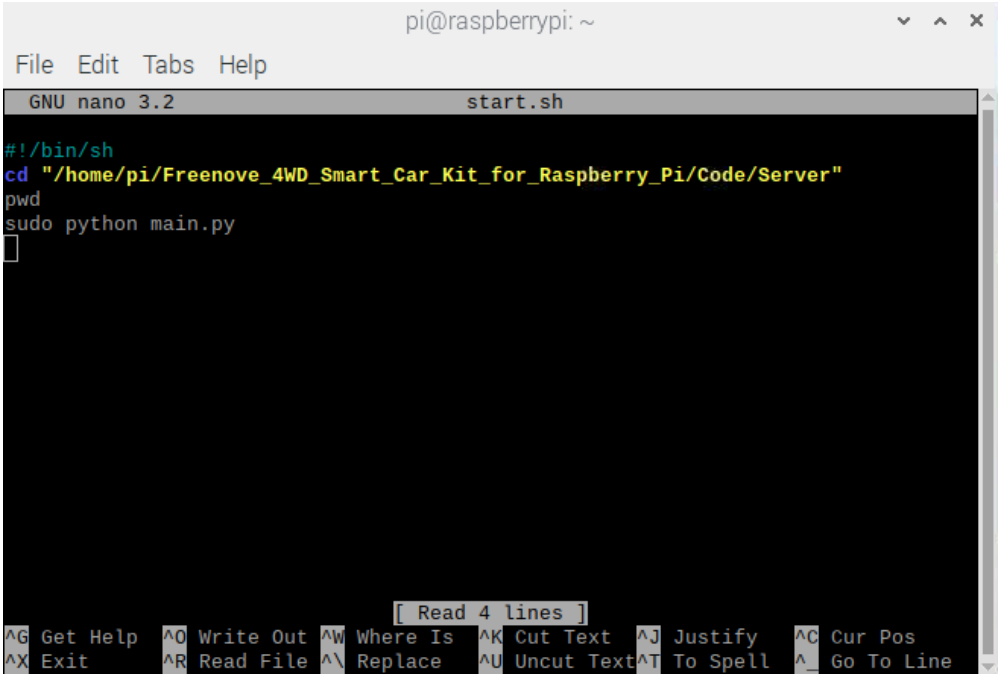
2 Open "start.sh".

```
sudo nano start.sh
```

3 Add the following contents to “start.sh” file.

```
#!/bin/sh
cd "/home/pi/Freenove_4WD_Smart_Car_Kit_for_Raspberry_Pi/Code/Server"
pwd
sleep 10
sudo python main.py
```

Press Ctrl + O and then press Enter to save it. Press Ctrl+X to exit.



4 Modify permissions.

```
sudo chmod 777 start.sh
```

5 Enter the following command to create a directory.

```
mkdir ~/.config/autostart/
```

6 create and open “start.desktop” file

```
sudo nano .config/autostart/start.desktop
```

7 Add the following content to “start.desktop” file.

```
[Desktop Entry]
Type=Application
Name=start
NoDisplay=true
Exec=/home/pi/start.sh
```

Press Ctrl + O and then press Enter to save it. Press Ctrl+X to exit.

8 Modify permissions.

```
sudo chmod +x .config/autostart/start.desktop
```

9 Finally enter the following content to reboot Raspberry Pi.

```
sudo reboot
```

Note: To cancel auto start, please delete the files “start.sh” and “start.desktop” created above.

Client

The client connects to the server through TCP, which receives the video stream from the server, and other commands. And it also sends commands to the server to control the car.
Clients can run on different systems, such as windows, Linux, and so on. However, you need to install related software and libraries.

The related program is mainly in the Video.py file under the Client folder.
Part of client code is as below:

```
class VideoStreaming:
    def __init__(self):
        self.face_cascade = cv2.CascadeClassifier(r'haarcascade_frontalface_default.xml')
        self.video_Flag=True
        self.connect_Flag=False
        self.face_x=0
        self.face_y=0
    def StartTcpClient(self,IP):
        self.client_socket1 = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        self.client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    def StopTcpcClient(self):
        try:
            self.client_socket.shutdown(2)
            self.client_socket1.shutdown(2)
            self.client_socket.close()
            self.client_socket1.close()
        except:
            pass

    def IsValidImage4Bytes(self,buf):
        bValid = True
        if buf[6:10] in (b'JFIF', b'Exif'):
            if not buf.rstrip(b'\0\r\n').endswith(b'\xff\xd9'):
                bValid = False
        else:
            try:
                Image.open(io.BytesIO(buf)).verify()
            except:
                bValid = False
        return bValid

    def face_detect(self,img):
        if sys.platform.startswith('win'):
            gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
```

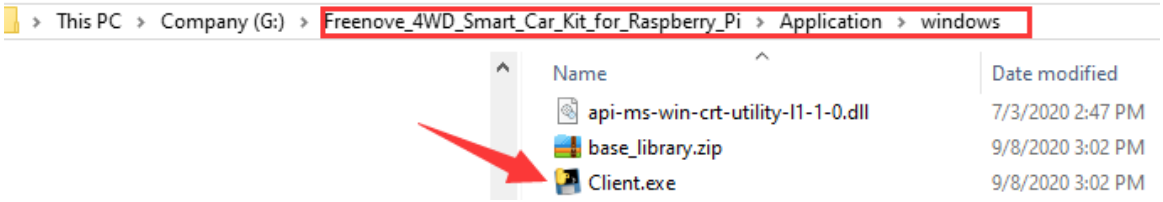
```
35              faces = self.face_cascade.detectMultiScale(gray,1.3,5)
36              if len(faces)>0 :
37                  for (x,y,w,h) in faces:
38                      self.face_x=float(x+w/2.0)
39                      self.face_y=float(y+h/2.0)
40                      img = cv2.circle(img, (x+w/2,y+h/2), (w+h)/4, (0, 255, 0), 2)
41              else:
42                  self.face_x=0
43                  self.face_y=0
44          cv2.imwrite('video.jpg', img)
45
46      def streaming(self,ip):
47          stream_bytes = b' '
48          try:
49              self.client_socket.connect((ip, 8000))
50              self.connection = self.client_socket.makefile('rb')
51          except:
52              #print "command port connect failed"
53              pass
54          while True:
55              try:
56                  stream_bytes= self.connection.read(4)
57                  leng=struct.unpack('L', stream_bytes[:4])
58                  jpg=self.connection.read(leng[0])
59                  if self.IsValidImage4Bytes(jpg):
60                                  image = cv2.imdecode(np.frombuffer(jpg, dtype=np.uint8),
61  cv2.IMREAD_COLOR)
62                                  if self.video_Flag:
63                                      self.face_detect(image)
64                                      self.video_Flag=False
65              except:
66                 break
```

Run client on windows system

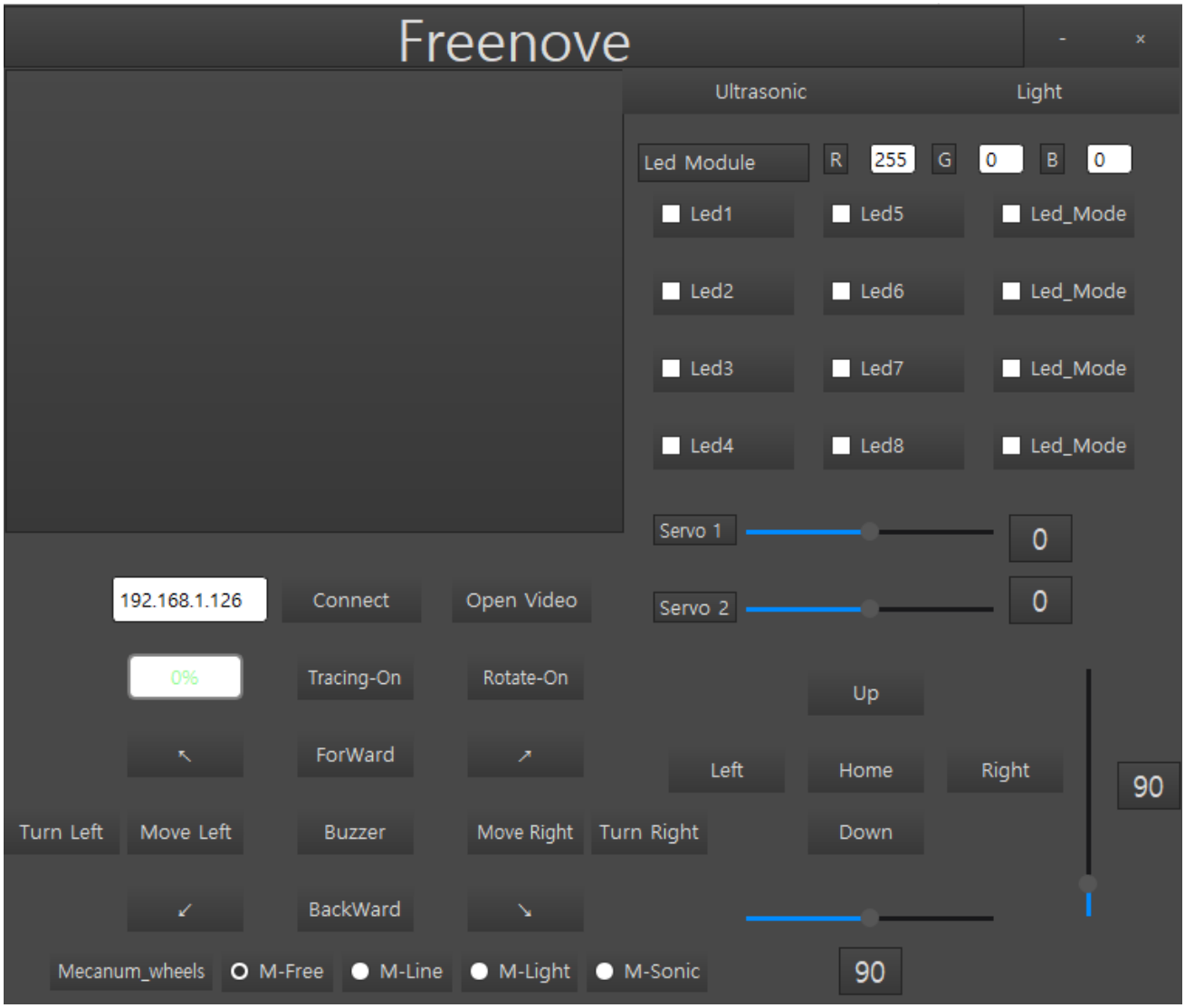
There are two ways to run Client on Windows.

Option 1 Running executable file directly

Find the "Client.exe" file in the specified directory, double click it and the Client is opened.



The client interface is shown as below:



After the client opens successfully, you need open the Raspberry Pi and **open server first**, then enter the IP address of the Raspberry Pi in the white IP edit box, and then click "Connect" to connect smart car to Raspberry Pi. After the connection is successful, you can click on the controls on the interface to operate the car.

Note: when Raspberry Pi is shut down, server will be closed. You need open server again the next time.

If pressing forward but the car moves backward, please refer to page 60 to modify the code.

Option 2 Install python3 and some related python libraries to run the client

If you want to modify the client, please follow this section.
This section will be completed in your **computer with windows system, not Raspberry Pi.**
There are many relevant software and libraries needed to be installed in Windows system, which takes a long time. At this time, it does not need to run Server or use Raspberry Pi. You can shut down Raspberry Pi first. After the installation is completed, you need to open Raspberry Pi and server again.

Install python3

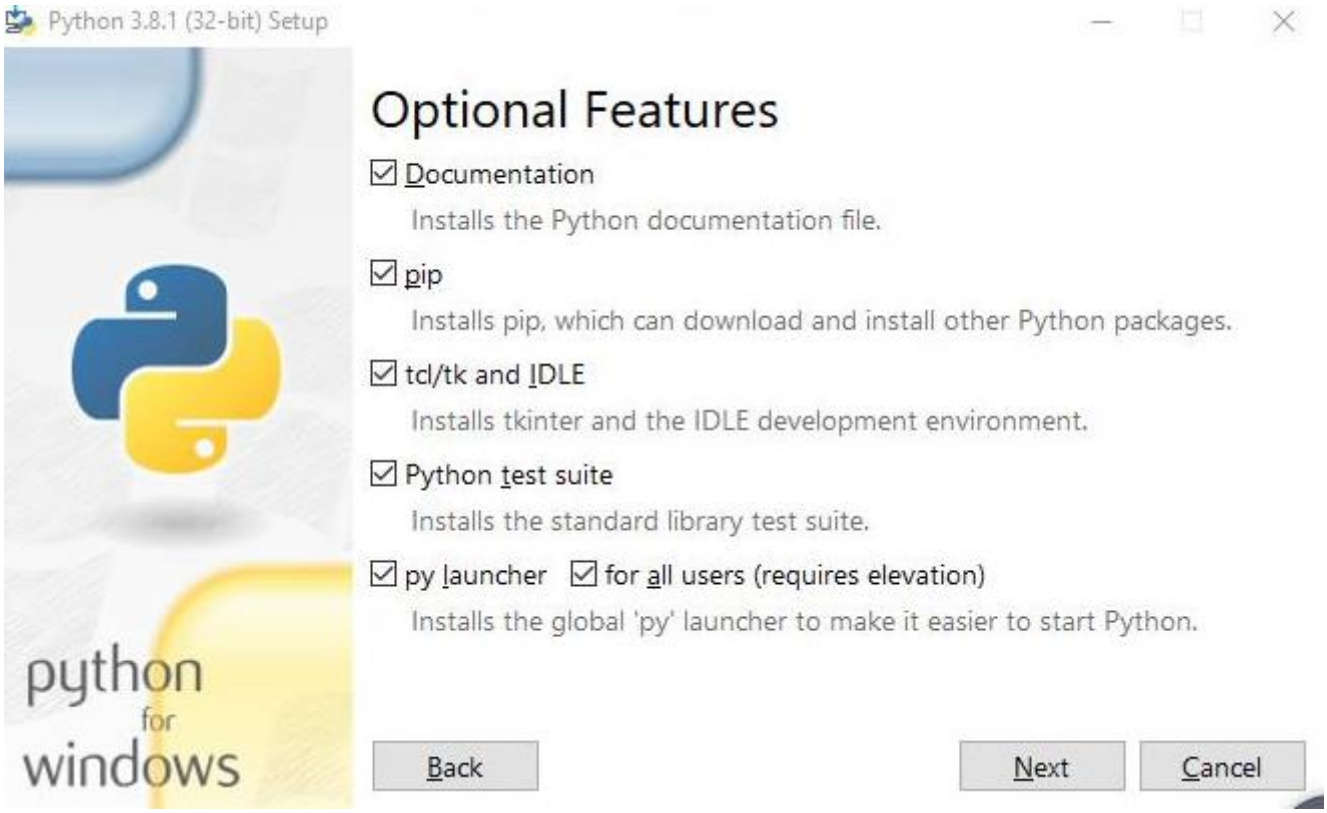
Download the installation file:
https://www.python.org/downloads/windows/

Click Latest Python 3 Release - Python 3.8.1

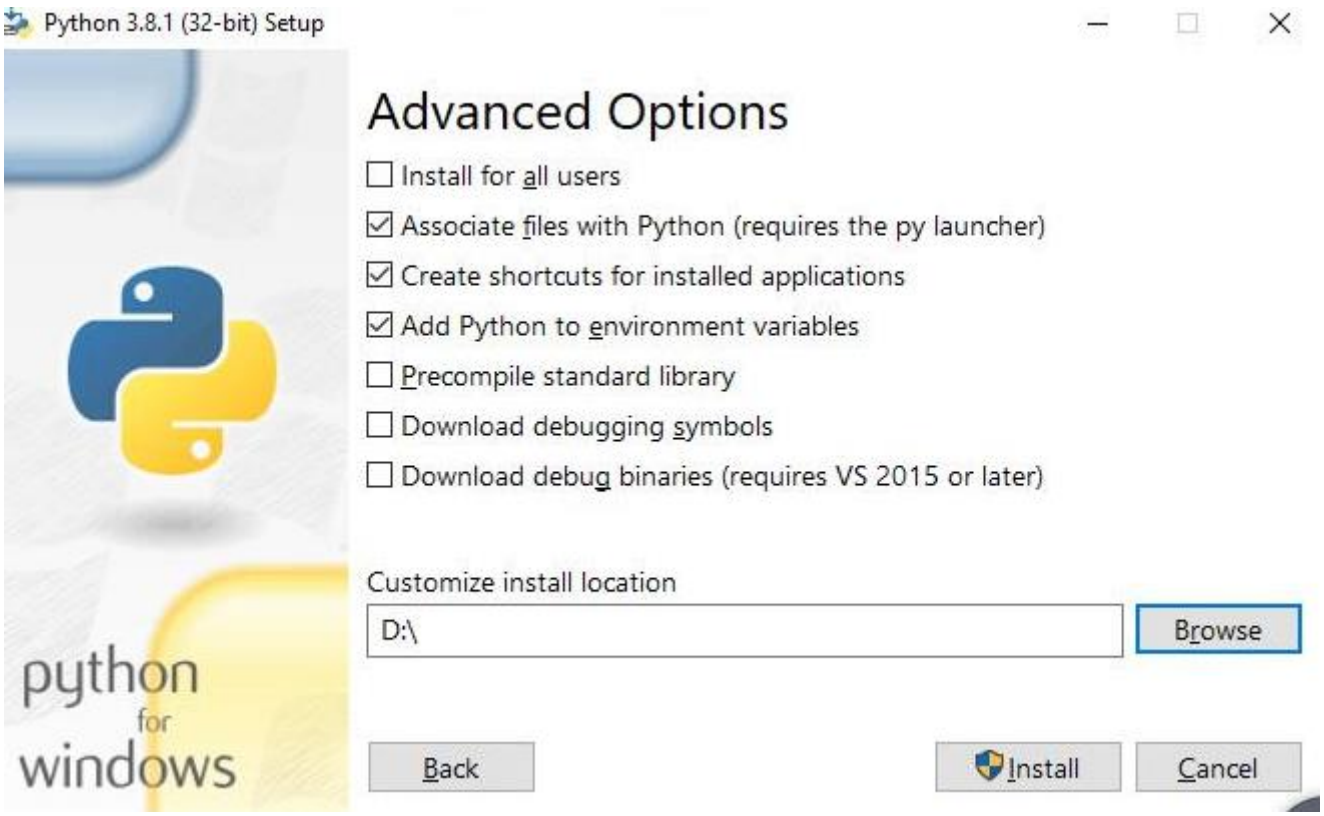
Version	Operating System	Description
Gzipped source tarball	Source release	
XZ compressed source tarball	Source release	
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later
Windows help file	Windows	
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64
Windows x86 embeddable zip file	Windows	
Windows x86 executable installer	Windows	
Windows x86 web-based installer	Windows	

Choose and download Windows x86 executable installer. After downloading successfully, install it.

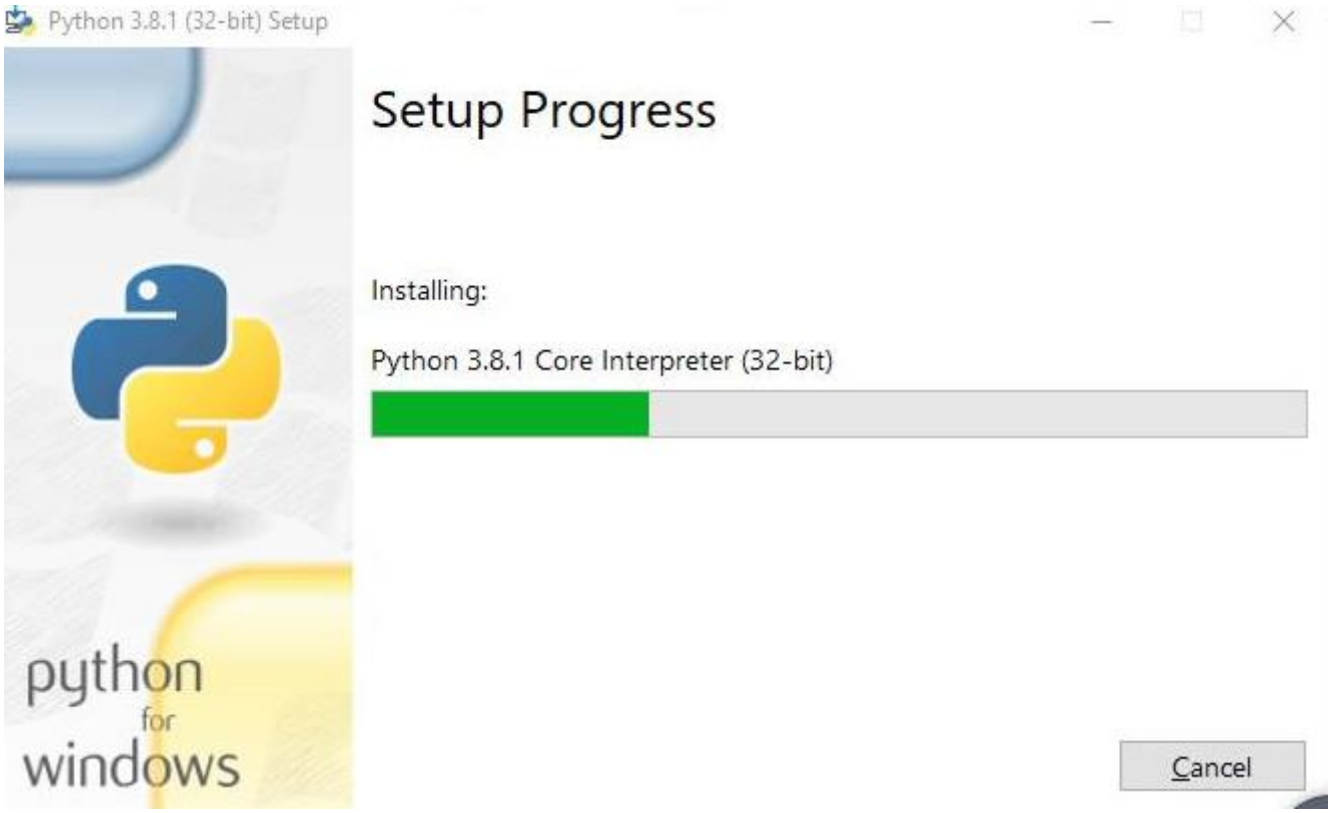
Select "**Add Python 3.8 to PATH**". You can choose other installation features.



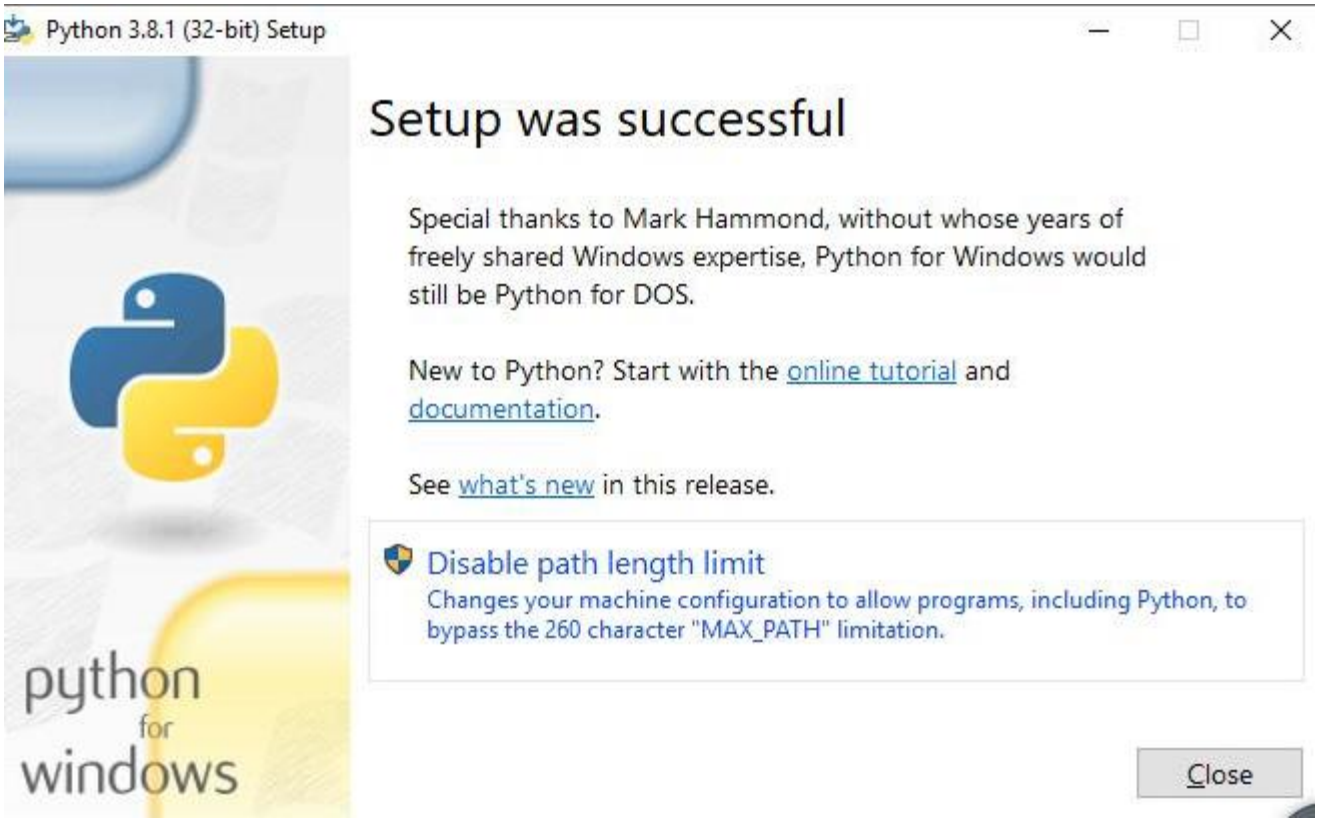
Select all options and click "Next".



Here, my install location is D. You can also choose other location. Then click "Install".



Wait installing.



Now, installation is completed.

Install PyQt5、opencv、numpy and other libraries.

If have not download the zip file, do so via:

https://github.com/Freenove/Freenove_4WD_Smart_Car_Kit_for_Raspberry_Pi/archive/master.zip

Then unzip it and delete "-master" to rename it to "Freenove_4WD_Smart_Car_Kit_for_Raspberry_Pi".

Then put it into D disk for example.

You can also place it into other disks (like E), but the path in the following command should be modified accordingly (replace D: by E:).

Press "win + R" and enter cmd, and click ok. Then enter following commands.

1. Enter D disk. (If you put it into E, it should be E:)

```
D:
```

2. Enter directory where setup_windows.py is located: (If you put it into E, it should be E:)

```
cd D:\Freenove_4WD_Smart_Car_Kit_for_Raspberry_Pi\Code
```

3. Run setup_windows.py:

```
Python3 setup_windows.py
```

```
C:\Users\Freenove>D:

D:\>cd D:\Freenove_4WD_Smart_Car_Kit_for_Raspberry_Pi\Code

D:\Freenove_4WD_Smart_Car_Kit_for_Raspberry_Pi\Code>Python3 setup_windows.py
```

Or enter the unzipped directory Freenove_4WD_Smart_Car_Kit_for_Raspberry_Pi\Code\Client.

And double-click **setup_client.py** or open it with python3.

Installation will take some time. Just wait patiently. For successful installation, it will prompt "All libraries installed successfully":

```
Package        Version
-------------- -------------
Click          7.0
numpy          1.18.1
opencv-python  4.1.2.30
Pillow         7.0.0
pip            19.2.3
PyQt5          5.13.2
PyQt5-sip      12.7.0
pyqt5-tools    5.13.2.1.6rc1
python-dotenv  0.10.3
setuptools     41.2.0
```

If not all installations are successful, it will prompt "Some libraries have not been installed yet. Please run ' Python3 setup_windows.py ' again", then you need to execute the Python3 setup_windows.py command again. Most of the installation failures are caused by poor networks. You can check your network before installing.

Open client

Press “win + R” and enter cmd, and click ok. Then enter following commands.

1. Enter D disk. If you put it into E, it should be E:

```
D:
```

2. Enter directory where Main.py is located:

```
cd D:\Freenove_4WD_Smart_Car_Kit_for_Raspberry_Pi\Code\Client
```

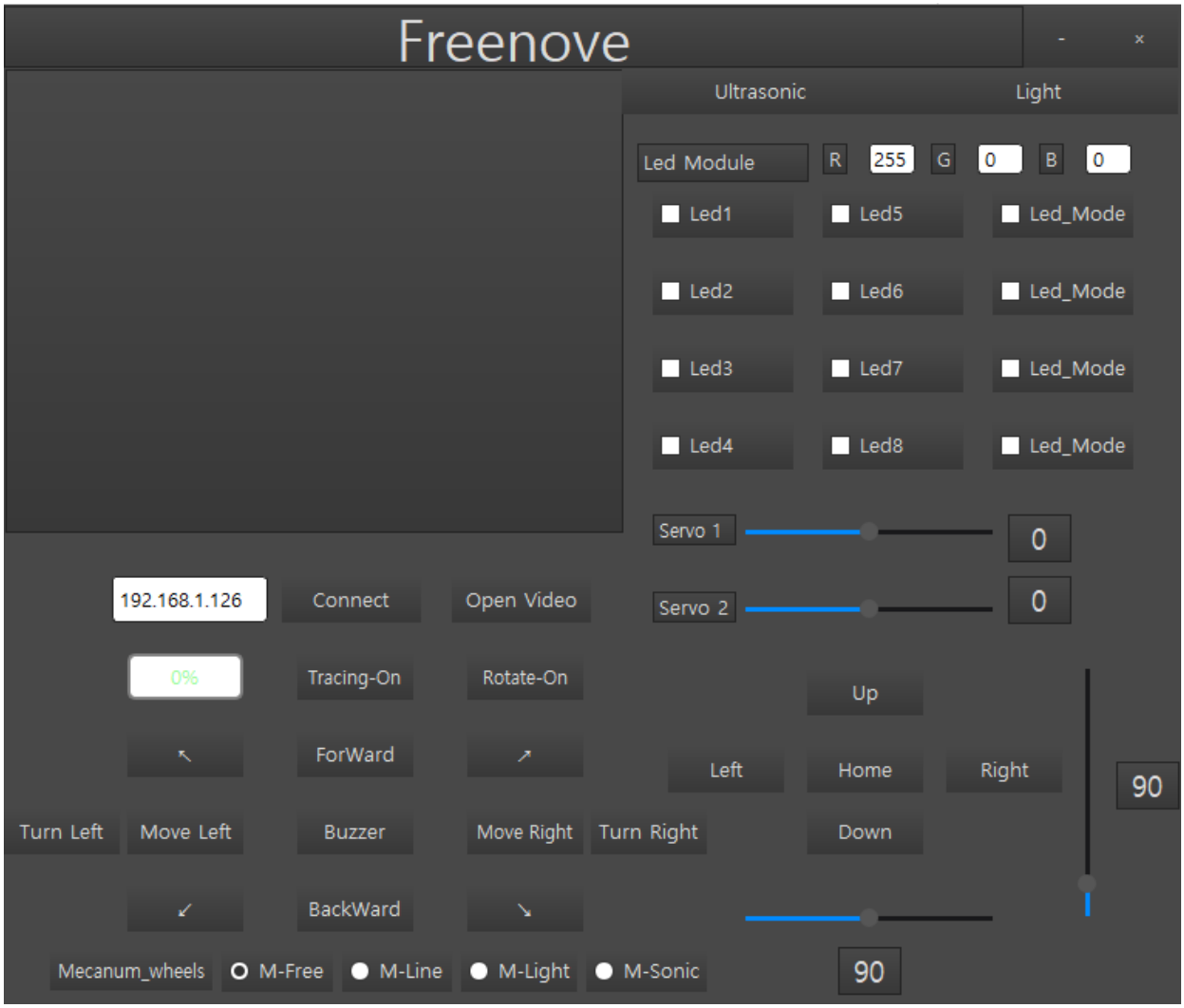
3. Run Main.py:

```
python Main.py
```

```
C:\Users\Freenove>D:

D:\>cd D:\Freenove_Four_wheeled_Smart_Car_Kit_for_Raspberry_Pi\Client

D:\Freenove_Four_wheeled_Smart_Car_Kit_for_Raspberry_Pi\Client>python Main.py
```

Or enter the unzipped directory and enter following directory: Freenove_4WD_Smart_Car_Kit_for_Raspberry_Pi\Code\Client. And double-click **Main.py** or open it with python to open the client.

The client interface is shown as below:



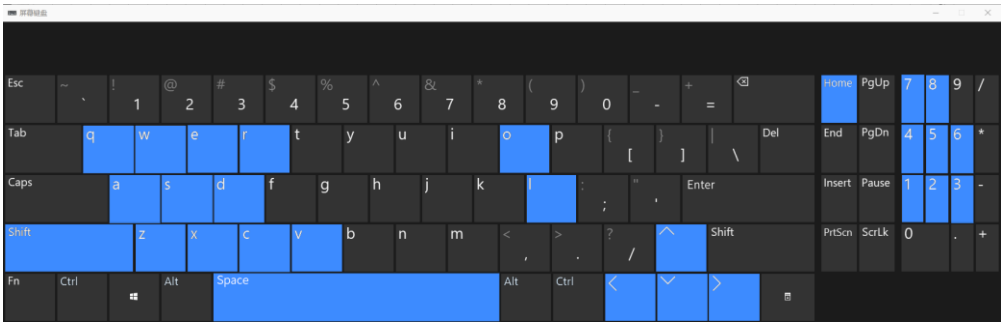
After the client opens successfully, you need open the Raspberry Pi and **open server first**, then enter the IP address of the Raspberry Pi in the white IP edit box, and then click "Connect" to connect smart car to Raspberry Pi. After the connection is successful, you can click on the controls on the interface to operate the car.

Note: when Raspberry Pi is shut down, server will be closed. You need open server again the next time.

If pressing forward but the car moves backward, please refer to page 51 to modify the code.

Control

And you can also control the car with following blue keys.



The car has four work modes:

Mode	Function
M-Free (Mode1)	Free control mode
M-Light (Mode2)	Light tracing mode
M-Sonic (Mode3)	Ultrasonic obstacle avoidance mode
M-Line (Mode4)	Infrared line tracking mode

The following is the corresponding operation of the buttons and keys.

Button on Client	Key	Action
ForWard	W	Move
BackWard	S	Back off
Turn Left	A	Move left
Turn Right	D	Move right
Move Left	Shift+A	Turn left
Move Right	Shift+D	Turn right
↖	Q	Left-forward diagonal
↗	E	Right-forward diagonal
↙	Z	Left-backward diagonal
↘	X	Right-backward diagonal
Rotate On/Off	O	On/off Rotation
Left	left arrow	Turn camera left
Right	right arrow	Turn camera right
Up	up arrow	Turn camera up
Down	down arrow	Turn camera down
Home	Home	Turn camera back Home
Connect/ Disconnect	C	On/off Connection
Open Video/ Close Video	V	On/off Video
Mode 1,2,3,4	R	Switch Mode
Buzzer	Space	On/off Buzzer
Led 1,2,3,4,5,6,7,8	1,2,3,4,5,6,7,8	On/off Led 1,2,3,4,5,6,7,8
Led_Mode 1,2,3,4	L	Switch Led Mode

The function of SliderBar is below:

SliderBar	Function
Servo 1,2,	SliderBar Servo 1, 2 are used to slightly adjust the angle. If the servo is not fully centered during installation, you can slightly tune it via the SliderBar.

Other control information:

Control	Function
IP address Edit box	Enter IP address of Raspberry Pi
Power box	Show power level
R,G,B Edit box	Control the color of LED selected.

Button “Ultrasonic“	Show the distance from obstacle.
Button “Light “	Show voltage of two photoresistors.
Button “Tracing-On/Off “	Open and close face tracking

If you don’t want to enter IP address after open the client, you can make some modification as below:

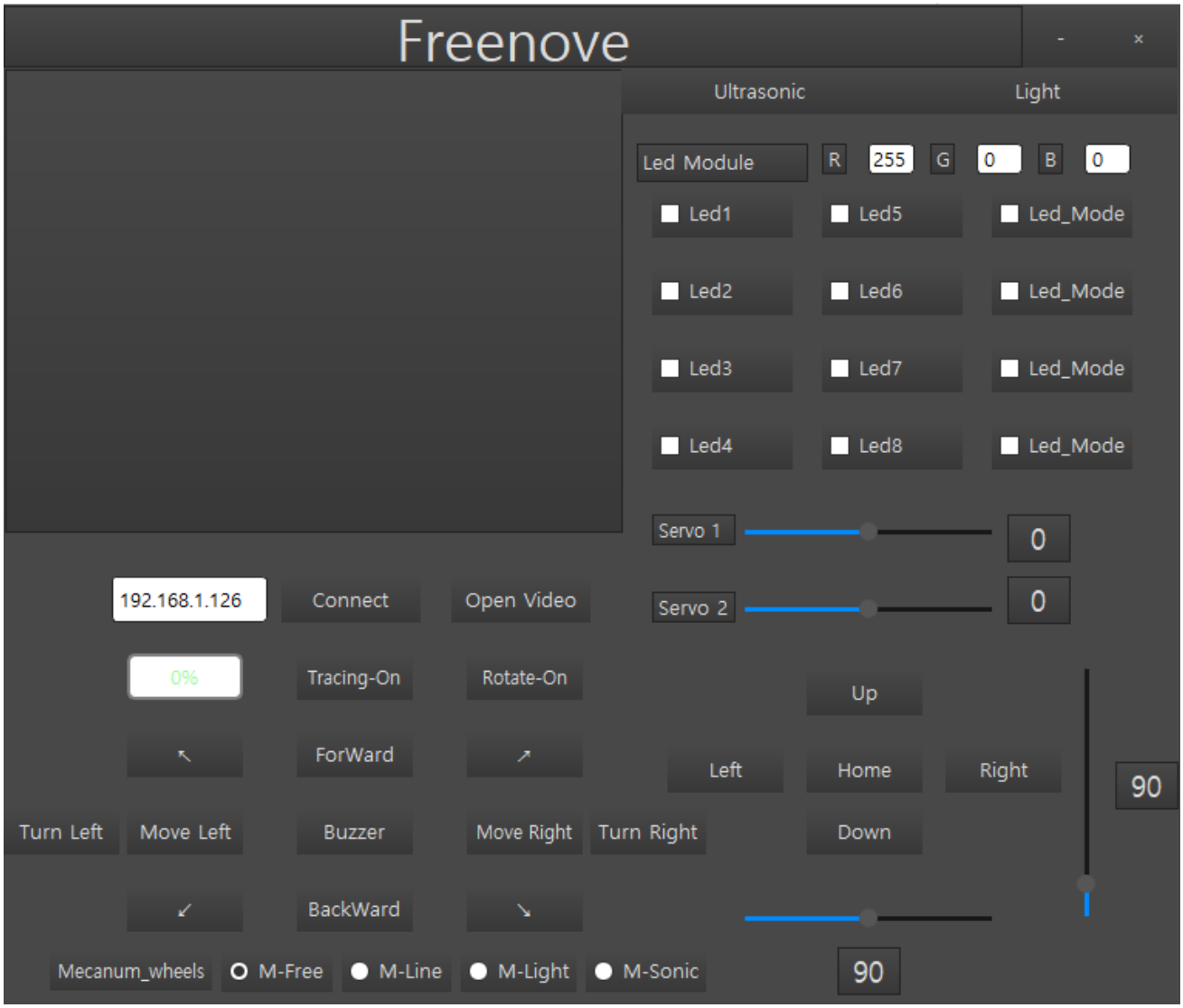
1. Open “Client_Ui.py” under directory “Client”, then find code in the thirty-sixth line from the bottom.

```
self.IP.setText(_translate("Client", "IP address", None))
```

2. Modify IP address to IP address of your Raspberry Pi. For example, my rpi IP is 192.168.1.126. After modification, it should be as below:

```
self.IP.setText(_translate("Client", "192.168.1.126", None))
```

Then save and close. And then restart your client. You can see it is modified successfully.



Run client on macOS system

Here take MacOS 10.13 as an example. To run the client on MacOS, you need to install some software and libraries. At this time, it does not need to run the server or use the Raspberry Pi. So you can turn off the Raspberry Pi first. After the installation is complete, turn on the Raspberry Pi and run the server. MacOS 10.13 comes with python2, but no python3. However, the programs in this project need run under python3, so you need to install it first.

Install python3

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

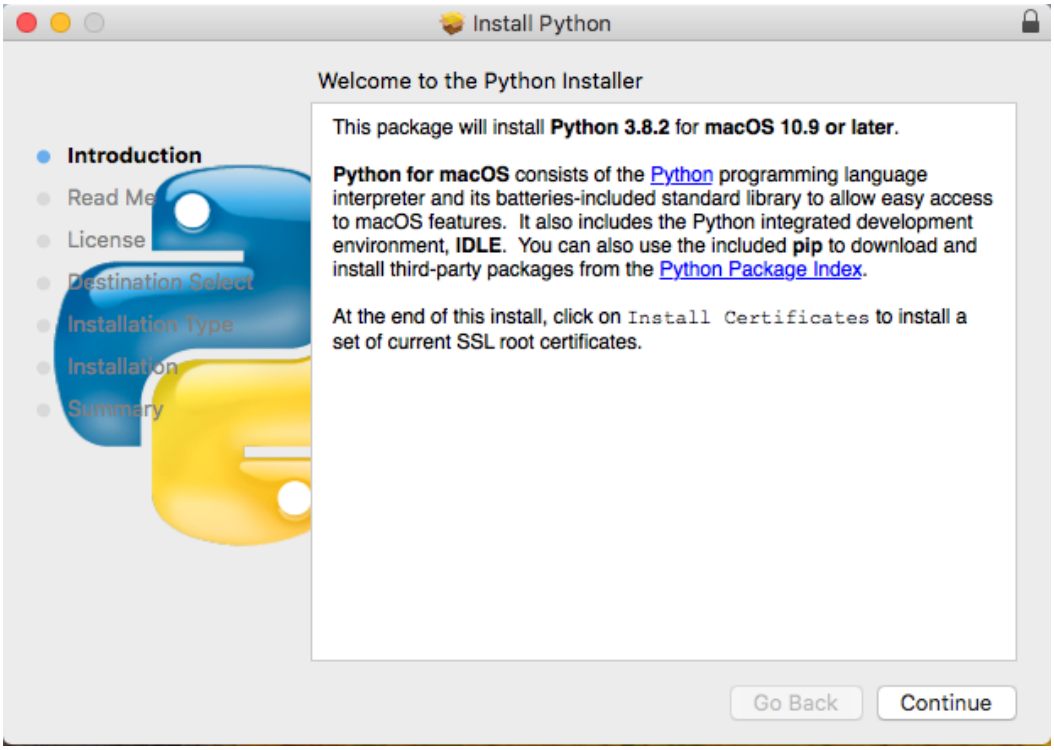
Python 3.8.1	Dec. 18, 2019	Download
Python 3.7.6	Dec. 18, 2019	Download

If your macOS is 11. Like 11.0, please install python 3.9.

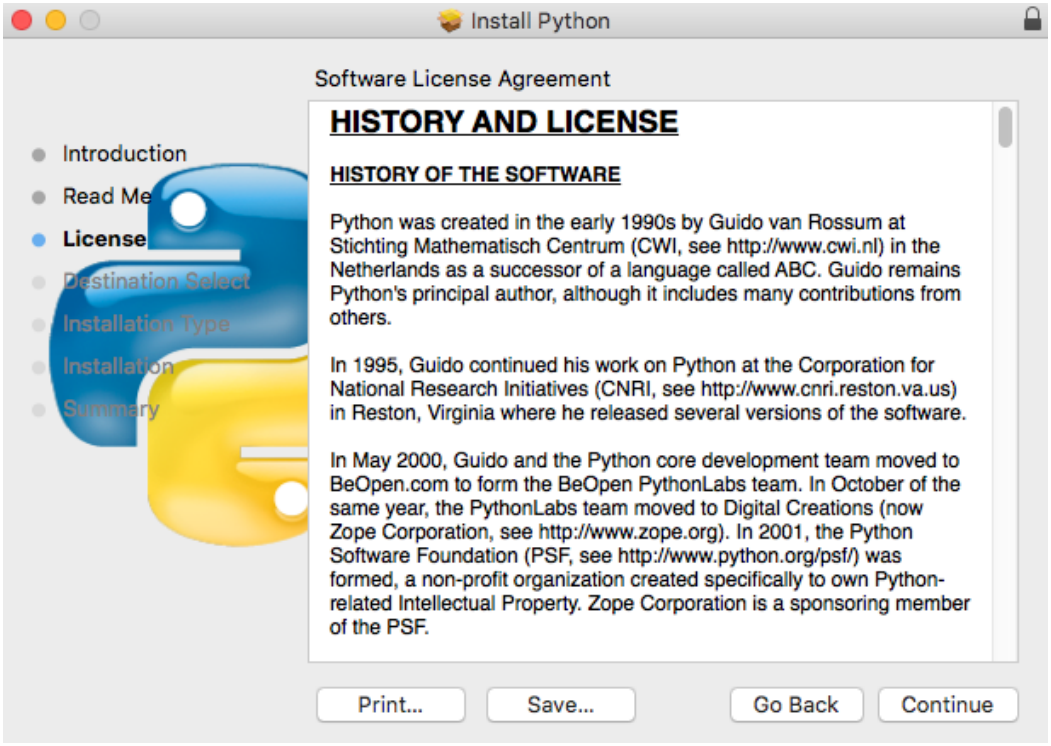
If your macOS is NOT 11, like 10.15, please install python 3.8. If you have installed python 3.9. You need uninstall it first.

Version	Operating System	Description
Gzipped source tarball	Source release	
XZ compressed source tarball	Source release	
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later
Windows help file	Windows	
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64
Windows x86 embeddable zip file	Windows	
Windows x86 executable installer	Windows	
Windows x86 web-based installer	Windows	

At bottom of the page, click macOS 64-bit installer and download installation package.

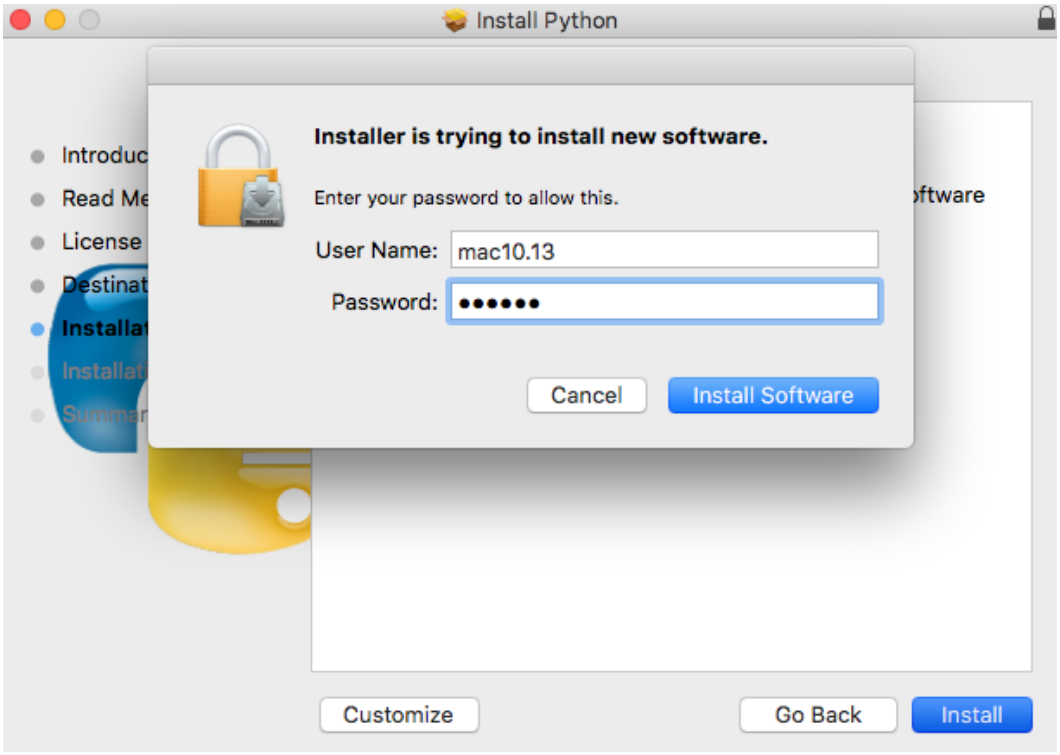


Click Continue.

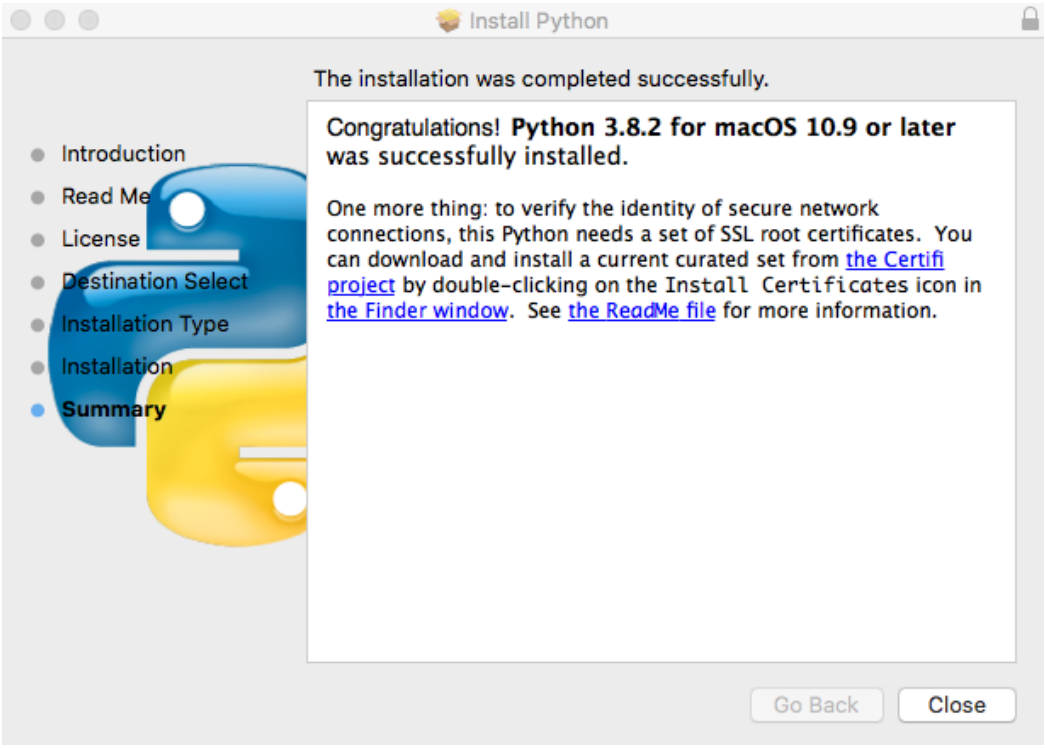


Click Continue

Click Agree.



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

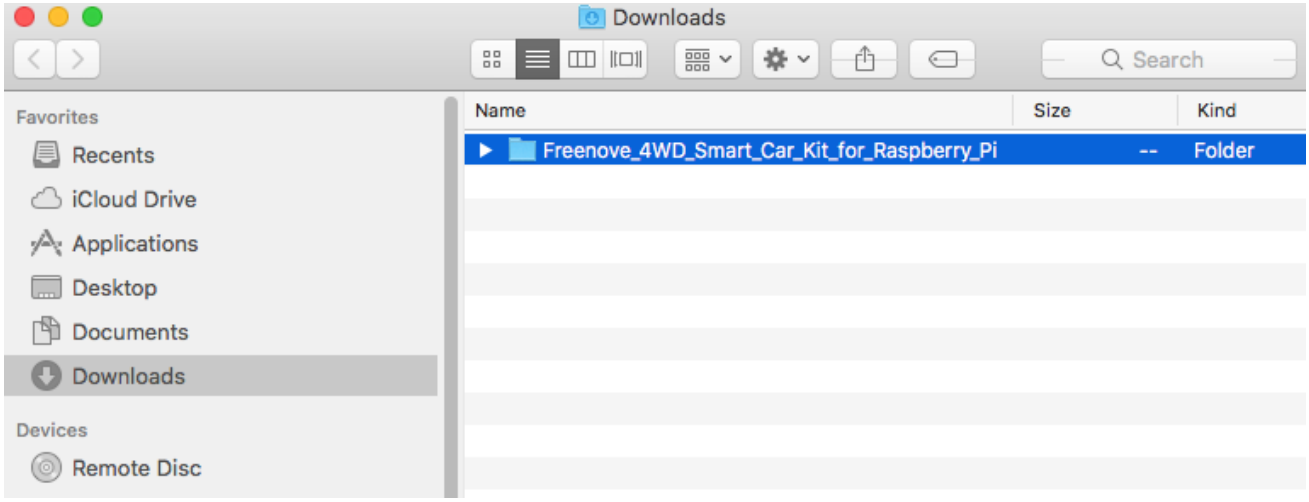


Now the installation succeeds.

You can find it in Aapplications.

Install PyQt5、opencv、numpy and other libraries

If there is no code for this car in your macOS system device, you can download it via the link below:
https://github.com/Freenove/Freenove_4WD_Smart_Car_Kit_for_Raspberry_Pi/archive/master.zip
After downloaded successfully, you can find it under Downloads.



Open the Terminal.

Type following commands in Terminal.

1 Enter "Downloads", (Where the Car code is located. If your location for it is different, please enter the location in your device.)

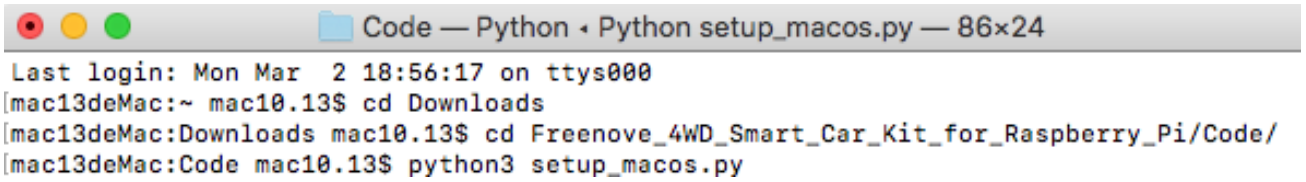
```
cd Downloads
```

2 Enter directory where setup_macos.py is located:

```
cd Freenove_4WD_Smart_Car_Kit_for_Raspberry_Pi/Code/
```

3 Run setup_macos.py:

```
python3 setup_macos.py
```



Installation will take some time. Just wait patiently. For successful installation, it will prompt "All libraries installed successfully":

```
Package                Version
---------------------- --------
numpy                  1.18.1
opencv-python-headless 4.2.0.32
Pillow                 7.0.0
pip                    20.0.2
PyQt5                  5.14.1
PyQt5-sip              12.7.1
setuptools             41.2.0

All libraries installed successfully
mac13deMac:Code mac10.13$
```

If not all installations are successful, it will prompt "Some libraries have not been installed yet. Please run 'python3 setup_macos.py' again", then you need to execute the python3 setup_macos.py command again. Most of the installation failures are caused by poor networks. You can check your network before installing.

If you are using **macOS under 11.0, like 10.15.** **Just skip to "Open client".**

If you are using **macOS 11.0 or later version**. Please run commands below:

```
pip3 uninstall PyQt5
pip3 install PyQt5
```

Open client

Following the previous step, after the installation is completed, you are now in the directory where setup_macos.py is located.

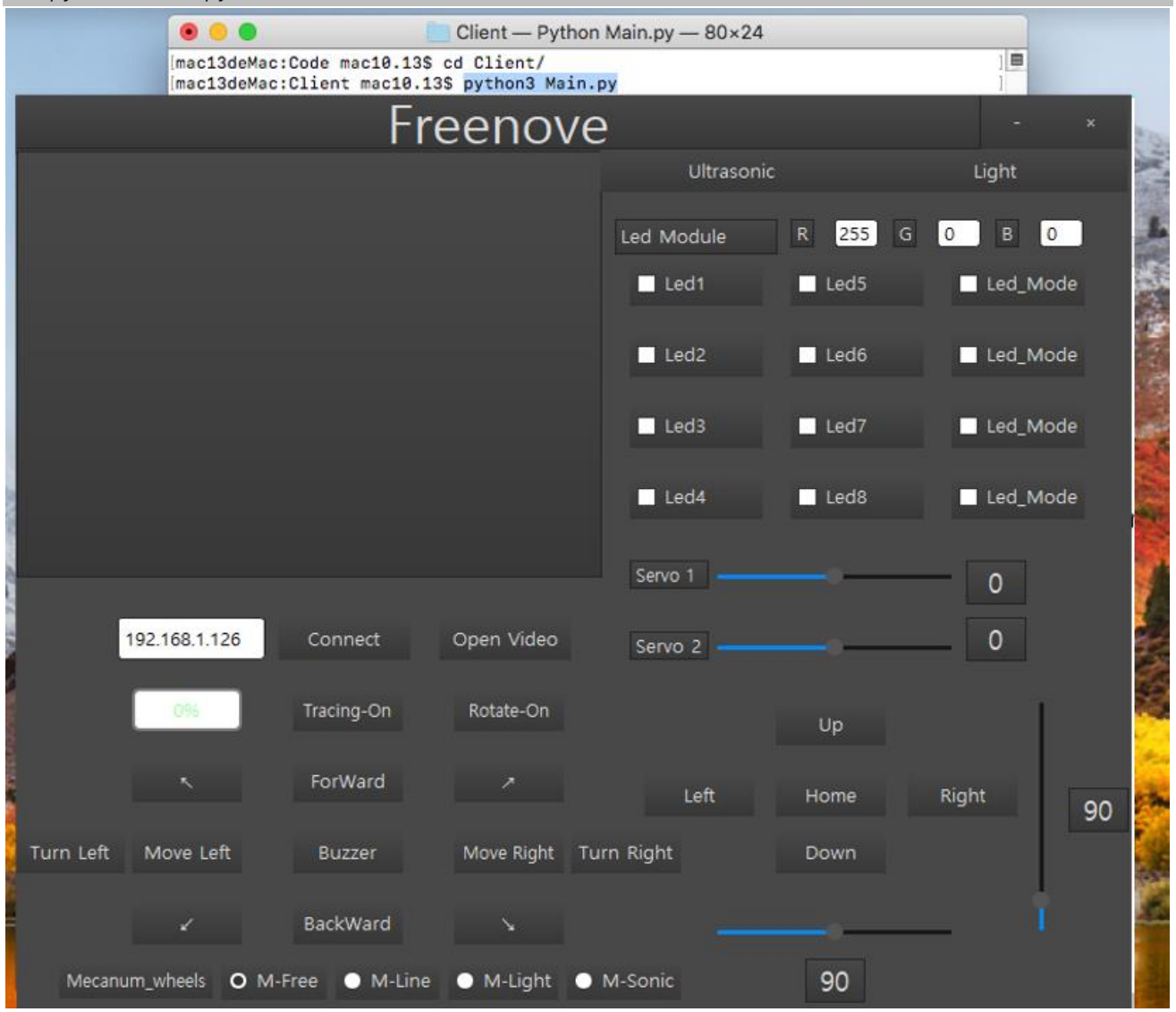
```
Package                 Version
----------------------- --------
numpy                   1.18.1
opencv-python-headless  4.2.0.32
Pillow                  7.0.0
pip                     20.0.2
PyQt5                   5.14.1
PyQt5-sip               12.7.1
setuptools              41.2.0

All libraries installed successfully
mac13deMac:Code mac10.13$
```

1.Type following command to enter Client folder.

```
cd Client/
```

2.Type following command to run Main.py.

```
python3 Main.py
```



The control way of Raspberry Pi macOS System client is same with Windows (Control).

Run client in Raspberry Pi (Linux system)

Install Opencv library

Execute the following commands in the terminal to install Opencv library:

1. Install opencv development environment:

```
sudo apt-get install -y libopencv-dev python3-opencv
```

2. Install some tools:

```
sudo apt-get install -y  python3-pil python3-tk
```

Run client

Enter the following commands at the terminal.

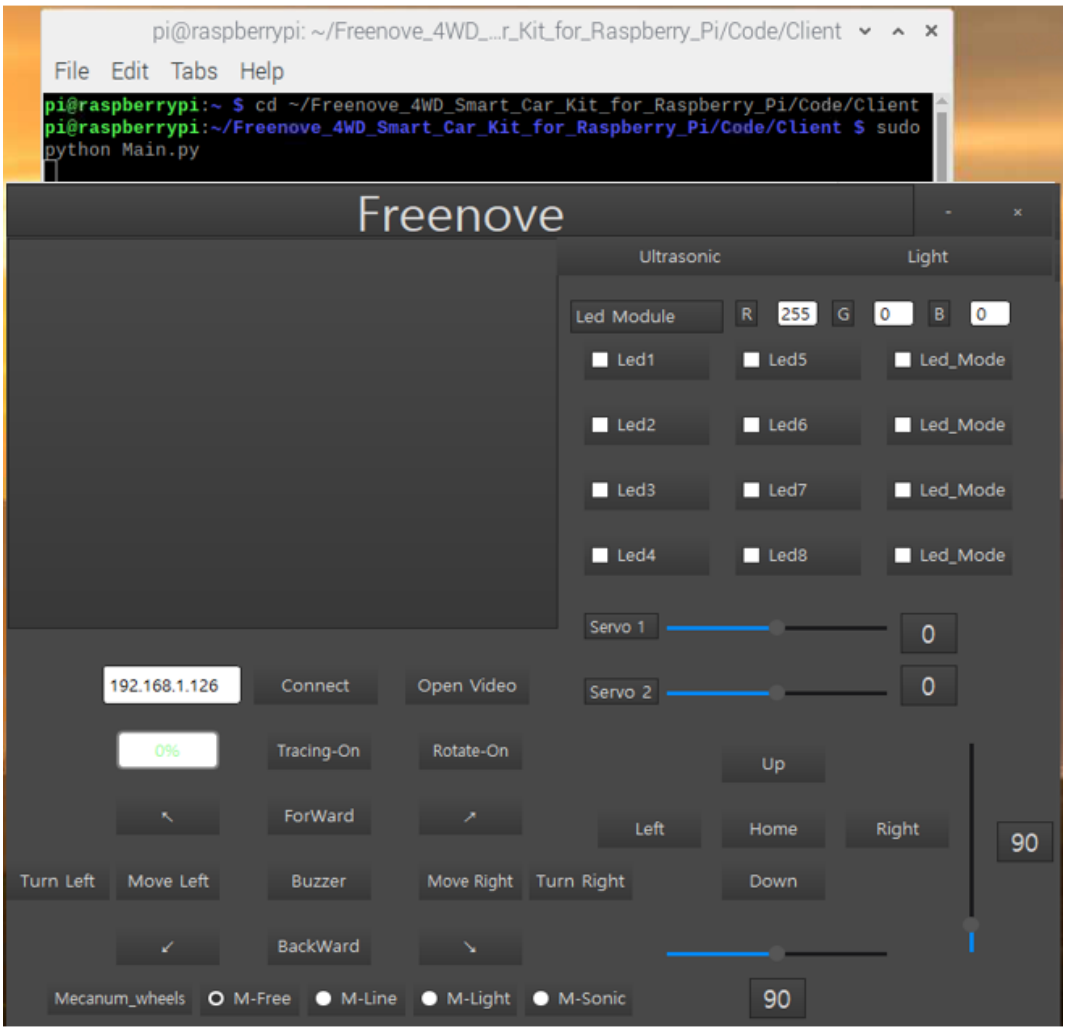
1. Use the cd command to go to the directory where Main.py is located.

```
cd ~/Freenove_4WD_Smart_Car_Kit_for_Raspberry_Pi/Code/Client
```

2. Run Main.py:

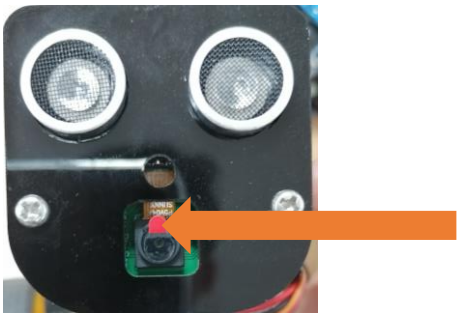
```
sudo python Main.py
```

The interface is shown below:



Please check whether the camera protective film is torn off.

Troubleshooting



If the car works abnormally, it may be caused by following reasons: Raspberry Pi system is stuck or batteries have no power.

You need check batteries power indicator or recharge batteries. Make sure batteries have enough power. When the batteries voltage is less than 7V, the buzzer will make regular sound.

If the batteries are OK, Raspberry Pi system is stuck. You need wait some time to check if the client works. Or reopen the server and client.

The latest Raspberry Pi official system is not stable. It occasionally is stuck. The old version is more stable.

If the Raspberry Pi system is stuck for a long time, you need reboot Raspberry Pi.

If you have any concerns, please feel free to contact us with pictures:

support@freenove.com

Communication Command

Communication Command Format

1. Each command is composed of four parts: command string, delimiter, parameter, and terminator.
A. The first string of each command serves to differentiate the primary category of the command, such as "CMD_MODE."
B. The "#" character acts as a delimiter between the command string and the parameter, used to separate the two elements.
C. Each command concludes with "\n," which serves to terminate the command. It is utilized to distinguish between individual commands. For instance: "CMD_MODE #0\n."

2. Parse of Commands
When parsing commands, first separate the commands with "\n", and then separate the command word and parameters of each command with"#". The characters after "\n" are divided to the next command for parsing.

Command Words

We have defined the following command strings.

```
class COMMAND:
    CMD_MOTOR = "CMD_MOTOR"
    CMD_M_MOTOR = "CMD_M_MOTOR"
    CMD_CAR_ROTATE = "CMD_CAR_ROTATE"
    CMD_LED = "CMD_LED"
    CMD_LED_MOD = "CMD_LED_MOD"
    CMD_SERVO = "CMD_SERVO"
    CMD_BUZZER = "CMD_BUZZER"
    CMD_MODE ="CMD_MODE"
```

Explanation of Communication Commands

CMD_MOTOR

is used to control the motor speeds of the standard version of the car. The four parameters correspond to the speeds of the four motors: left-front, left-rear, right-front, and right-rear. The speed range for each motor is from -4056 to 4056.

App command	Action
CMD_MOTOR#1500#1500#1500#1500\n	Forward
CMD_MOTOR#-1500#-1500#-1500#-1500\n	Backward
CMD_MOTOR#-1500#-1500#1500#1500\n	Turn Left
CMD_MOTOR#1500#1500#-1500#-1500\n	Turn Right
CMD_MOTOR#0#0#0#0\n	Stop

CMD_ M_MOTOR

is used to regulate the motor speeds of the standard version of the car. The command incorporates four parameters, each with a distinct role:

1. The first parameter signifies the angle, with respect to the Y-axis, of joystick one on the mobile app.
2. The second parameter denotes the displacement of joystick one.
3. The third parameter represents the angle, again relative to the Y-axis, of joystick two.
4. The fourth parameter indicates the displacement of joystick two.

In this context:

- The angle ranges from -180 to 180 degrees.
- The displacement falls within the interval of 0 to 4056.

App command	Action
CMD_M_MOTOR#0#1500#0#0\n	Forward
CMD_M _MOTOR#180#1500#0#0\n	Reverse
CMD_M_MOTOR#0#0#90#1500\n	Turn-left
CMD_M_MOTOR#0#0#-90#1500\n	Turn-right
CMD_M_MOTOR#0#0#0#0\n	Stop
CMD_M_MOTOR#90#1500#0#0\n	Left-translation
CMD_M_MOTOR#90#1500#0#0\n	Right-translation
CMD_M_MOTOR#45#1500#0#0\n	Front-left, diagonal translation
CMD_M_MOTOR#45#1500#0#0\n	Front-right, diagonal translation
CMD_M_MOTOR#135#1500#0#0\n	Rear-left, diagonal translation
CMD_M_MOTOR#-135#1500#0#0\n	Rear-right, diagonal translation

CMD_CAR_ROTATE

is employed to manage the directional control of the car's rotation. The command integrates four parameters, each with a distinct function:

1. The first parameter signifies the angle, relative to the Y-axis, of joystick one on the mobile app.
2. The second parameter represents the displacement of joystick one.
3. The third parameter denotes the angle, with respect to the Y-axis, of joystick two.
4. The fourth parameter indicates the displacement of joystick two.

In this context:

- The angle ranges from -180 to 180 degrees.
- The displacement falls within the interval of 0 to 4056.

The extent of rotation is contingent upon the angle specified in the third parameter, offering precise control over the direction of rotation for the car.

App command	Action
CMD_CAR_ROTATE#0#0#0#0\n	Rotate forward
CMD_CAR_ROTATE #0#0#0#0\n	Rotate backward
CMD_CAR_ROTATE #0#0#0#0\n	Stop
CMD_CAR_ROTATE #90#1500#0#0\n	Rotate to the left

CMD_CAR_ROTATE #90#1500#0#0\n	Rotate to the right

CMD_LED

is used to control the color of the light in RGB mode. There are four parameters, representing the intensity values of the red, green, and blue color channels, as well as transparency. The range for each channel is from 0 to 255.

App command	Action
CMD_LED#255#0#0#0	RGB Red
CMD_LED#0#255#0#0	RGB Green
CMD_LED#0#255#0#0	RGB Bliue

CMD_LED_MOD

is used to control the RGB LED to different modes, with one parameter.

App command	Action
CMD_LED_MOD #0	Turn OFF RGB LEDs
CMD_LED_MOD #1	Common RGB Mode
CMD_LED_MOD #2	Following Mode
CMD_LED_MOD #3	Blink Mode
CMD_LED_MOD #4	Breathing Mode
CMD_LED_MOD #5	Rainbow Mode

CMD_SERVO

is used to control the angle of the servo motor. It has two parameters, representing the rotation angles of servo 0 and servo 1.

CMD_SERVO

is used to control the buzzer. It has one parameter: when the parameter is set to 0, the buzzer is turned off; when set to 1, it is turned on.

CMD_MODE

is used to control the auto modes of the car with one parameter.

App command	Action
CMD_MODE #0	Control mode
CMD_MODE #1	Light-tracing mode
CMD_MODE #2	Line-tracking mode
CMD_MODE #3	Obstacle avoidance mode

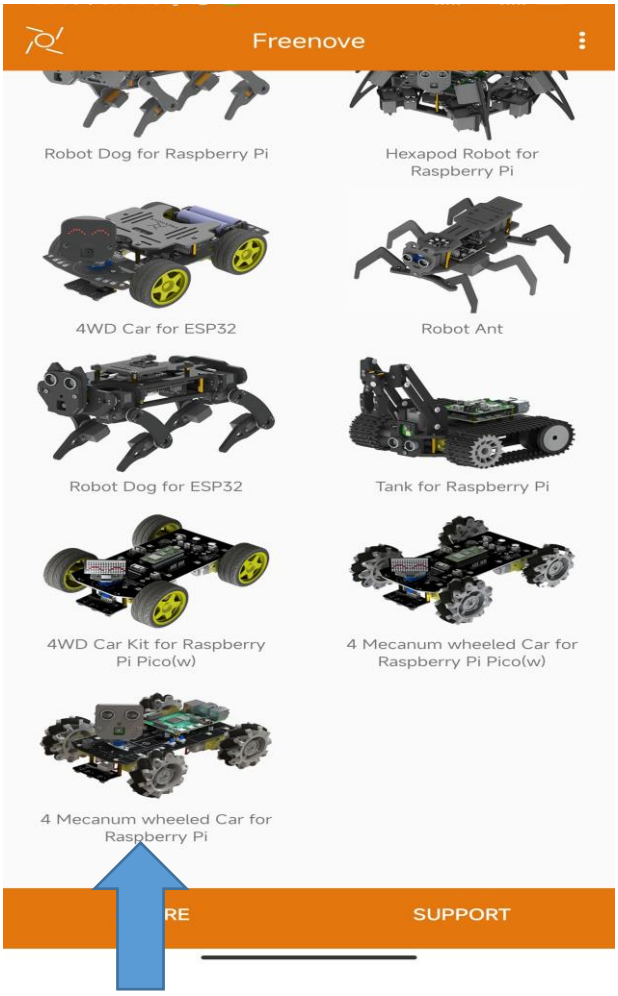
Android and iOS app

You can download and install the Freenove Android app from below:
On Google play:
https://play.google.com/store/apps/details?id=com.freenove.suhayl.Freenovez

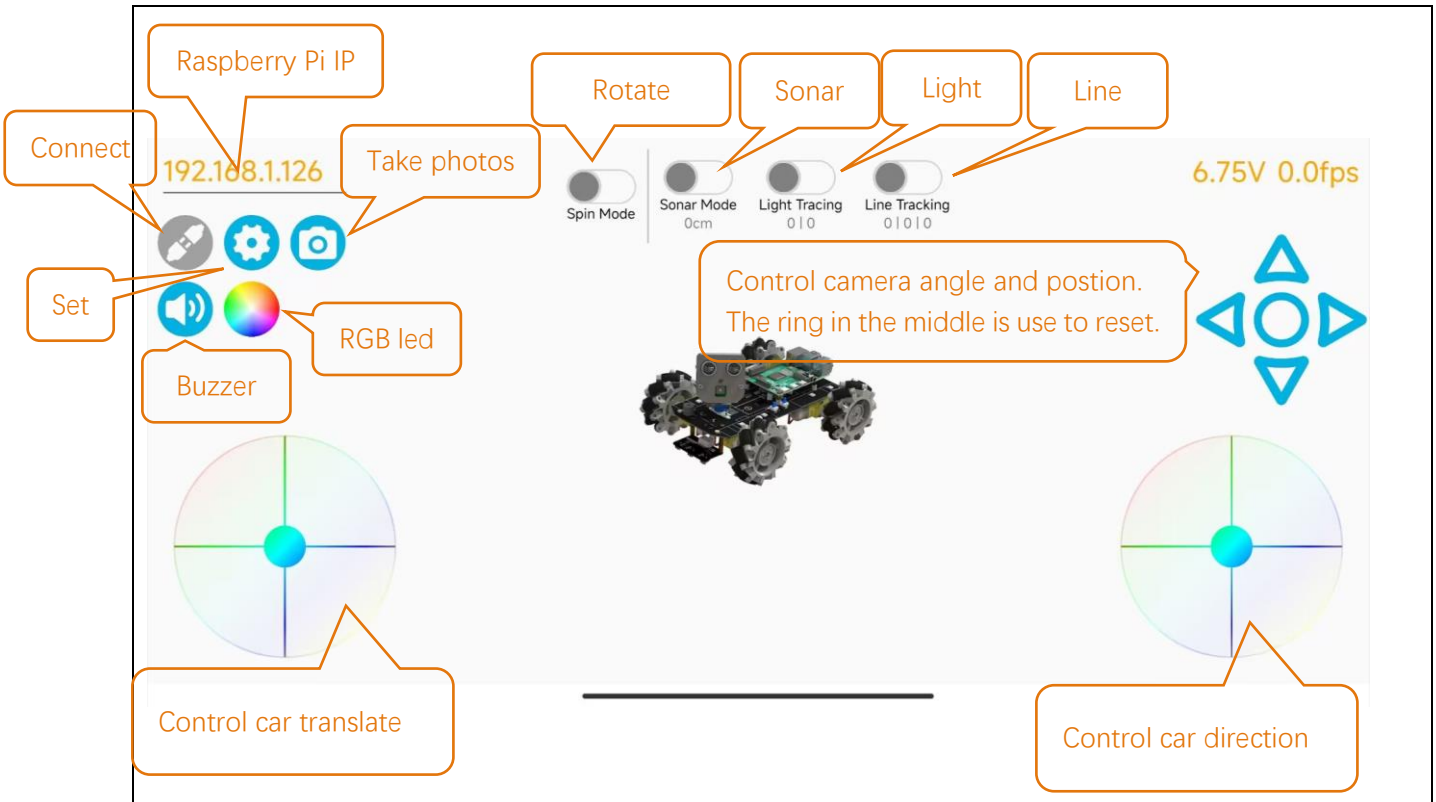
On GitHub:
https://github.com/Freenove/Freenove_App_for_Android
In this github repository, you can find the App instruction (Tutorial.pdf).

You can download and install the Freenove **iPhone ios app** by searching **freenove** in app store.

Open the app and select the car.



Open the server in Raspberry Pi car first. And enter your Pi IP.



Before proceeding with mobile app control, ensure the following:

1. Connect both your phone and Raspberry Pi to the same network environment.
2. Adjust the rotation time ratio as per the Advanced Movements section in Chapter 3.

Upon entering the mobile app, you will see two control joysticks. The first one is for controlling the car's omnidirectional movement. The second joystick has two functions. When the "Spin Mode" above it is not activated, it controls the direction of the car's movement. When the "Spin Mode" is activated, it controls the angle at which the car rotates while moving forward. In spin mode, keep the joystick pressed; releasing it will stop the car.

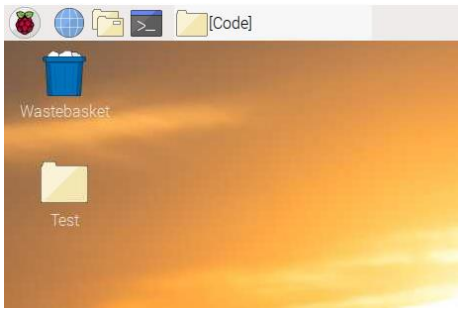
Free innovation

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

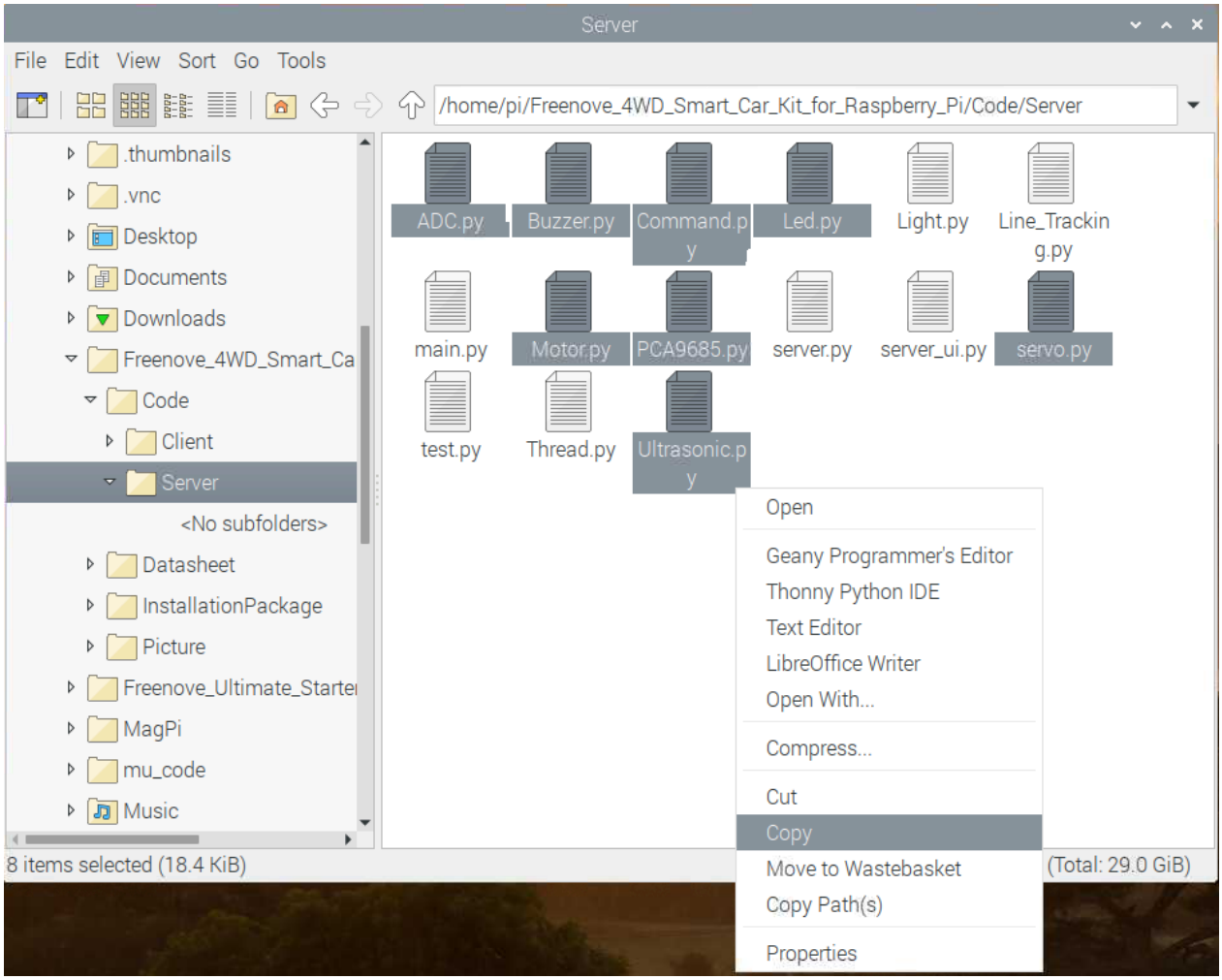
If you want to write your own program to control the car, just follow this section. We will teach you how to program this car.

If you have never learned python before, you can learn some basic knowledge via the link below: https://python.swaroopch.com/basics.html

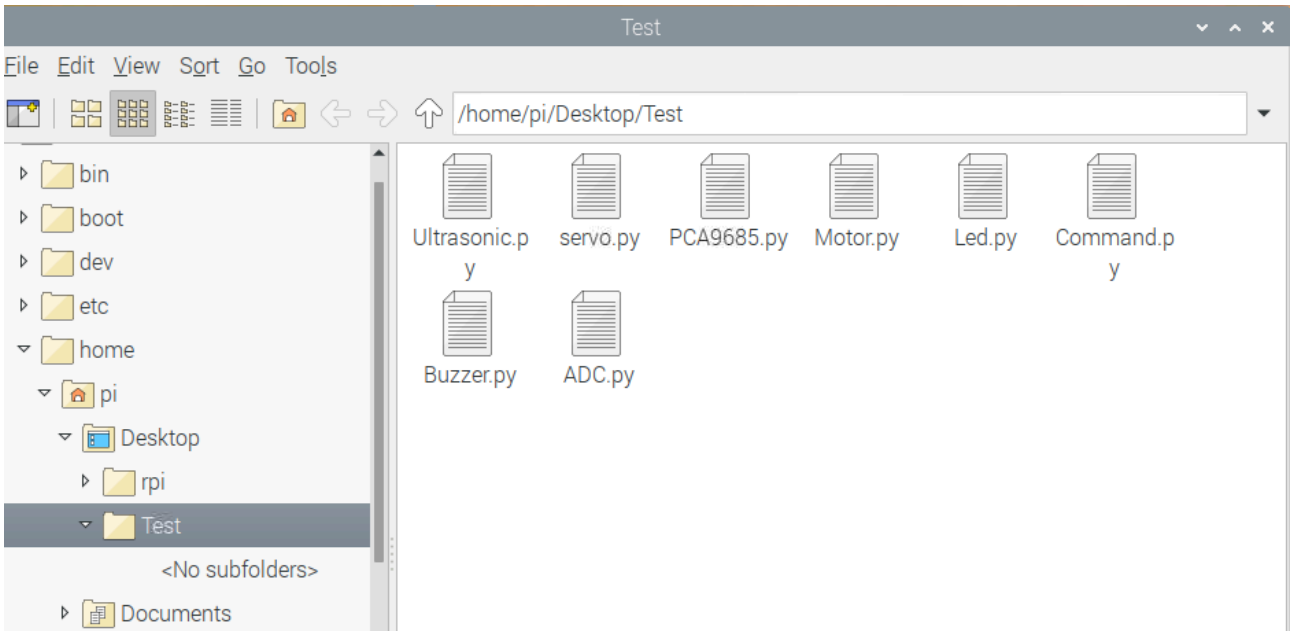
First, turned on S1 and S2. Then open Raspberry Pi, right click and create a new folder on the desktop: Test



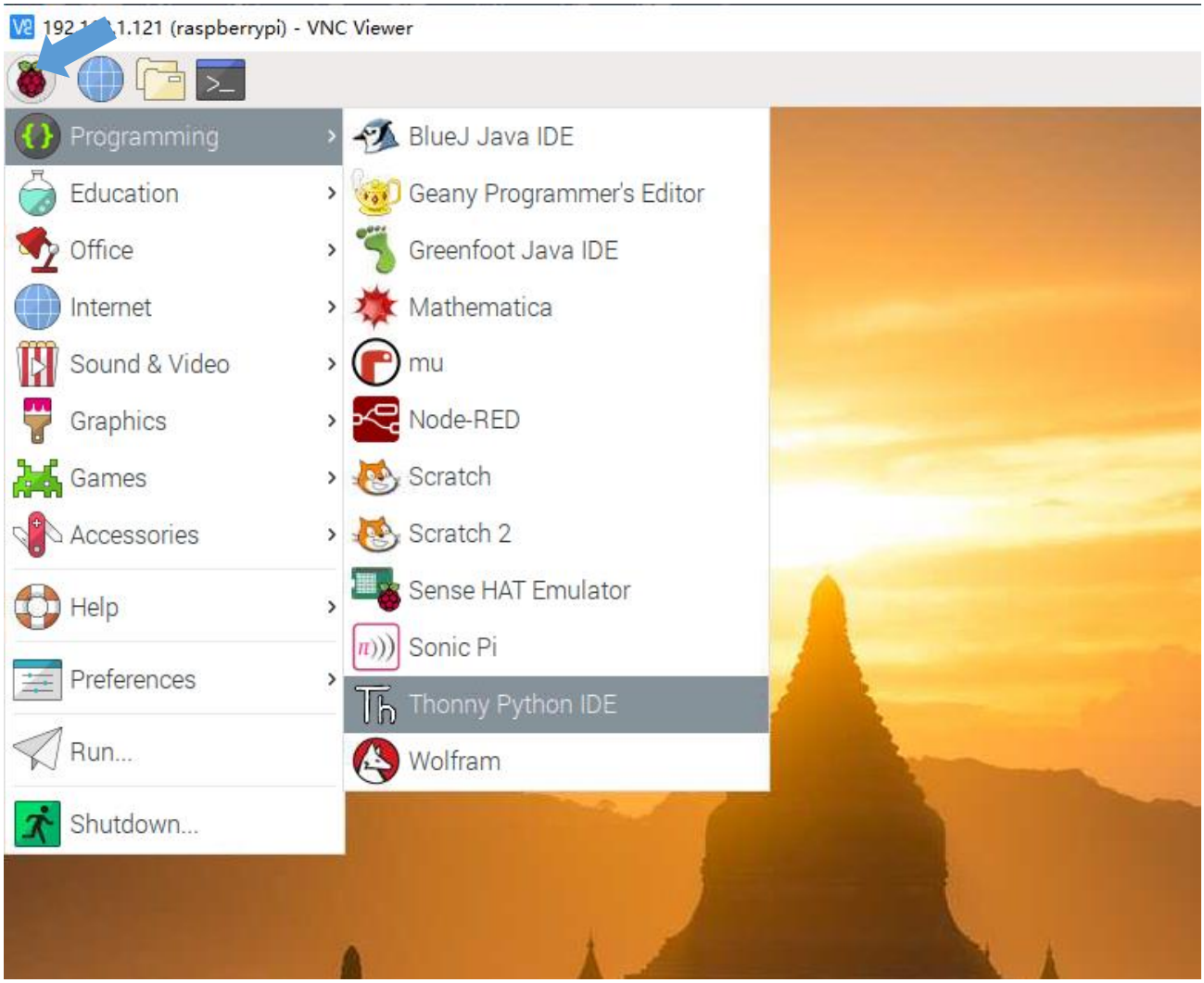
Open Freenove_4WD_Smart_Car_Kit_for_Raspberry_Pi/Code/Server in your Raspberry Pi and copy the following **8 files** into the Test folder we created.

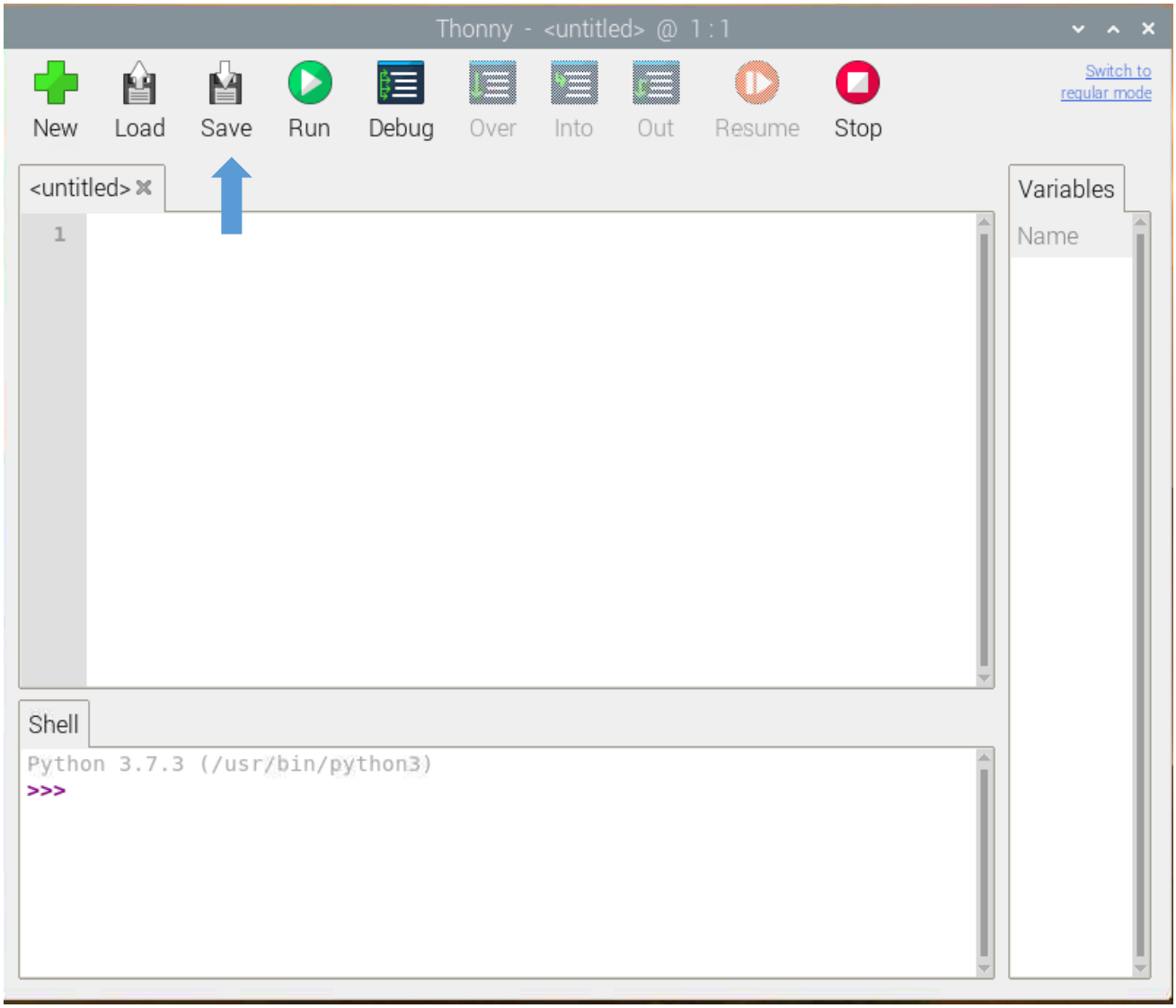


Paste them in Test folder.

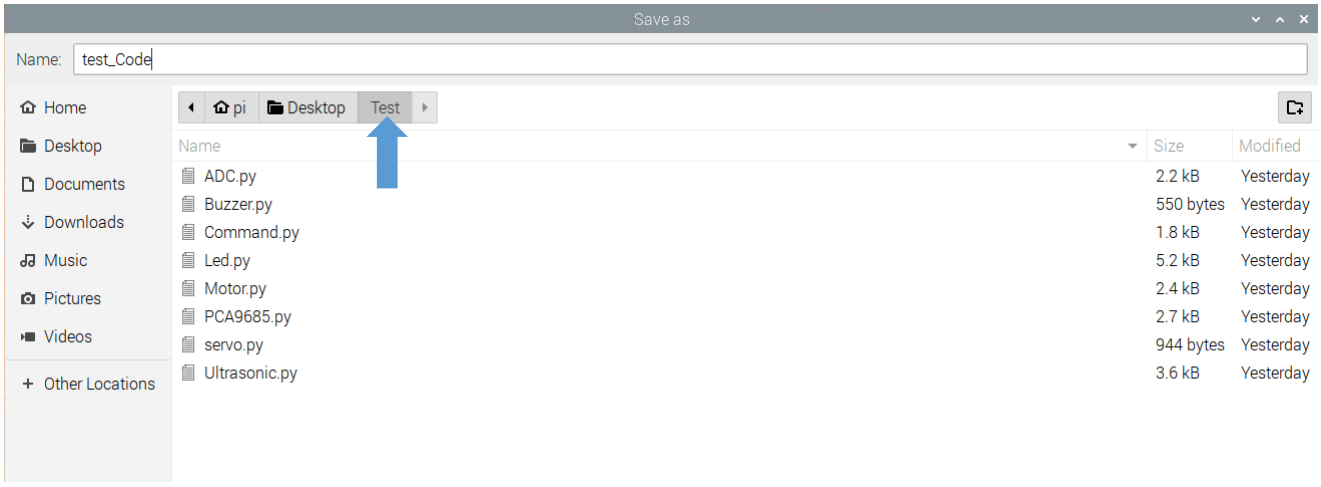


Run Thonny Python IDE

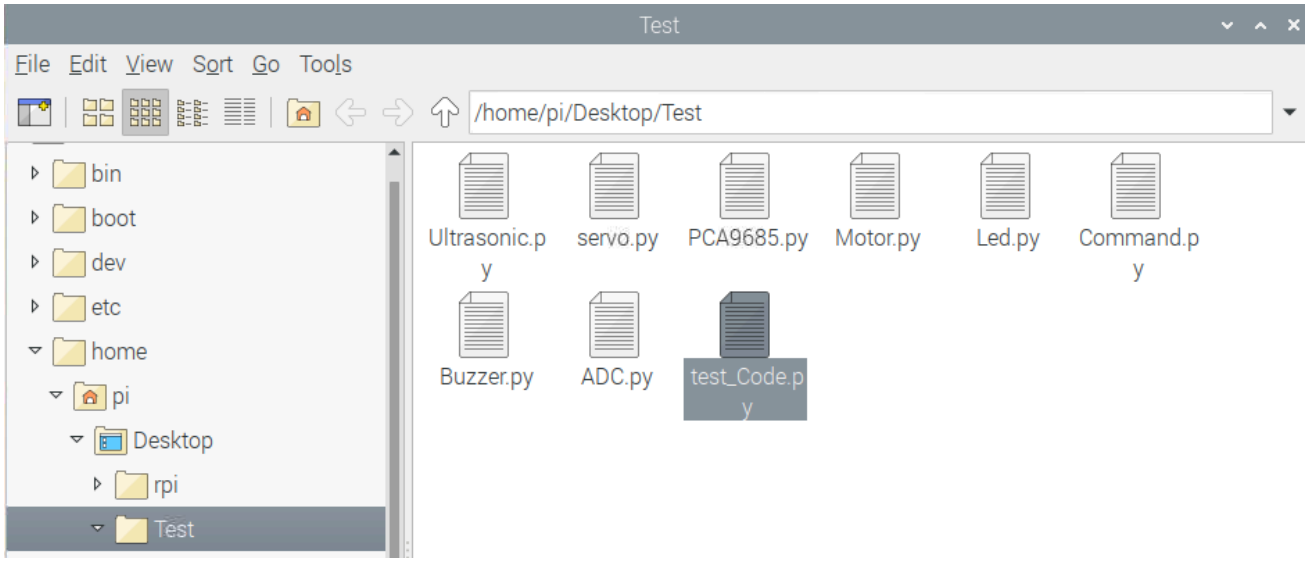




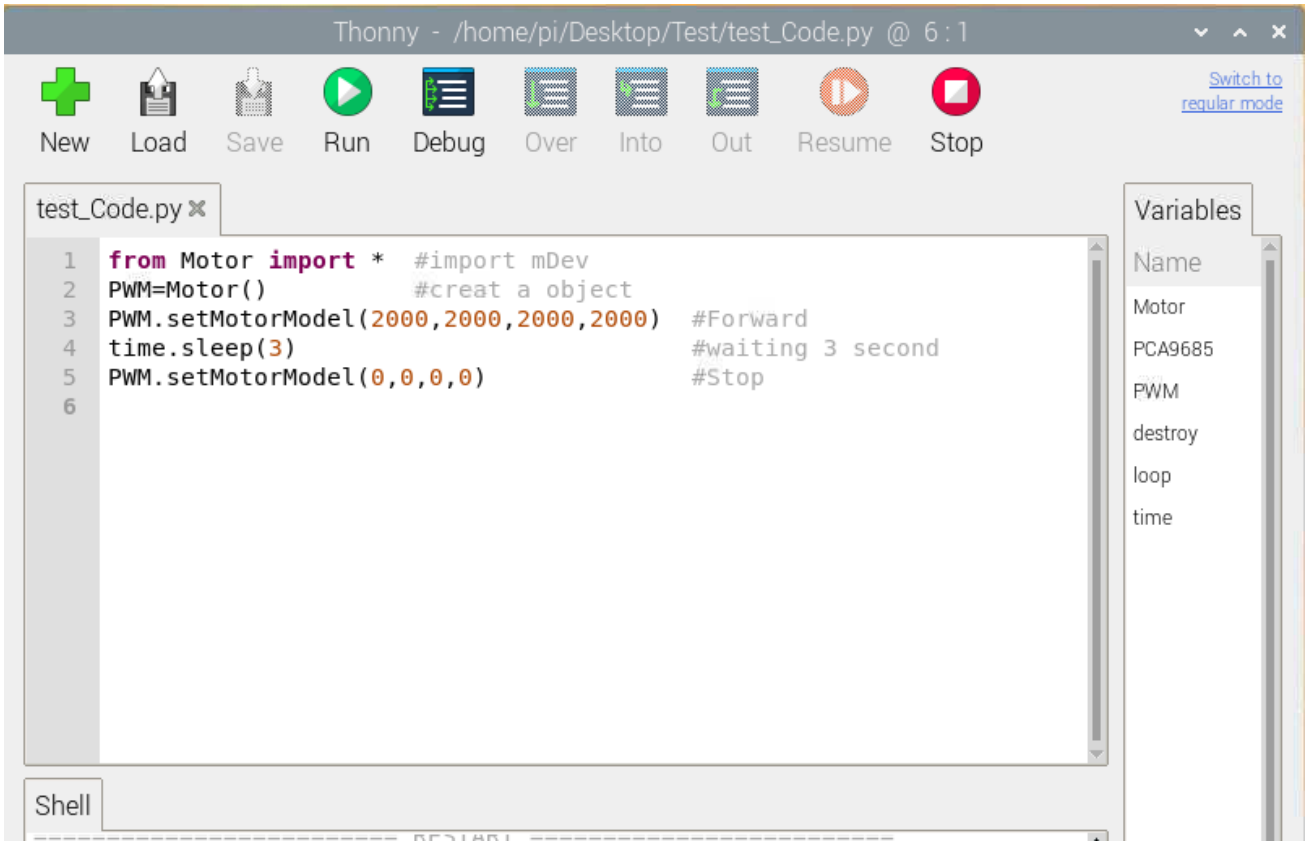
Click Save and save it into the Test folder, with name: test_Code.



Now you can see the file test_Code.py we created.



Then write code in test_Code.py, then click save.



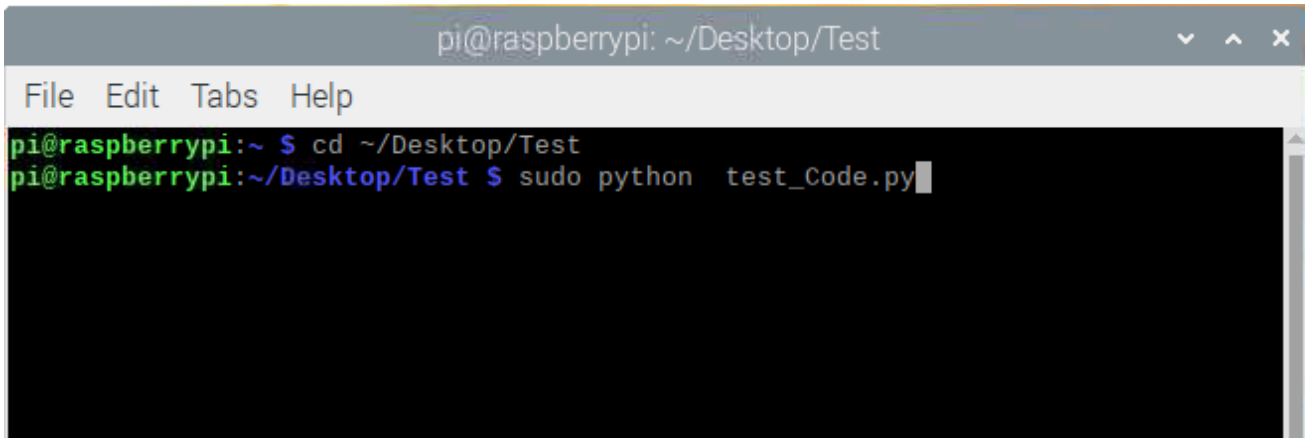
Note: the code and library are written by **Python 3.** You need execute the code with **python 3**.

Open the terminal and use the following command to enter the directory where test_Code.py is located:

```
cd ~/Desktop/Test
```

Run test_Code.py:

```
sudo python  test_Code.py
```



Code example

Following are code example for the parts. For more detail, please refer to Module test section.

For more details, please refer to Motor.

```
from Motor import *                          #import Motor
PWM=Motor()                                  #create an object
PWM.setMotorModel(2000,2000,2000,2000)       #Forward
time.sleep(3)                                #waiting 3 second
PWM.setMotorModel(0,0,0,0)                   #Stop
```

ADC. For more details, please refer to ADC.

```
from ADC import *              #import ADC
adc=Adc()                      #create an object
Left_IDR=adc.recvADC(0)        #get value
print ("The photoresistor voltage on the left is "+str(Left_IDR)+"V")
```

LED. For more details, please refer to LED.

```
from Led import *                          #import Led
led=Led()                                  #create an object
led.ledIndex(0x04,255,255,0)                  #yellow
led.ledIndex(0x80,0,255,0)                    #green
time.sleep(5)                              #wait 5s
led.colorWipe(led.strip, Color(0,0,0))     #turn off
```

Buzzer. For more details, please refer to Buzzer.

```
from Buzzer import *                #import Led
from Command import COMMAND as cmd  #import Led
buzzer=Buzzer()                     #create an object
buzzer.run('1')           #Start
time.sleep(3)                       #wait 3s
buzzer.run('0')            #Stop
```

Servo. For more details, please refer to Servo.

```
from servo import *   #import Led
pwm = Servo()            #create an object
#Servo rotates from 30 degrees to 150 degrees
for i in range(30, 150, 1) :
    pwm.setServoPwm('0', i)
    time.sleep(0.01)
#Servo rotates from 150 degrees to 0 degrees
for i in range(150, 30, -1) :
    pwm.setServoPwm('0', i)
    time.sleep(0.01)
```

Ultrasonic module. For more details, please refer to Ultrasonic module.

```
from Ultrasonic import *         #import Led
ultrasonic=Ultrasonic()          #create an object
data=ultrasonic.get_distance()   #Get the value
print ("Obstacle distance is "+str(data)+"CM")
```

These codes can be integrated into one code to achieve your requirement.

What's next?

Thanks for your reading.

This book is all over here. If you find any mistakes, missions or you have other ideas and questions about contents of this book or the kit and ect., please feel free to contact us, and we will check and correct it as soon as possible.

After completing the contents in this book, you can try to reform this smart car, such as purchasing and installing other Freenove electronic modules, or improving the code to achieve different functions. We will also try our best to add more new functions and update the code on our github (https://github.com/freenove).

If you want to learn more about Arduino, Raspberry Pi, smart cars, robots and orther interesting products in science and technology, please continue to focus on our website. We will continue to launch cost-effective, innovative and exciting products.
www.freenove.com

Thank you again for choosing Freenove products.