Free your innovation

Freenove is an open-source electronics platform.
www.freenove.com

Warning

When you purchase or use this product, please note the following:

- This product contains small parts. Swallowing or improper operation them can cause serious infections and death. Seek immediate medical attention when the accident happened.
- Do not allow children under 3 years old to play with or near this product. Please place this product in where children under 3 years of age cannot reach.
- Do not allow children lack of ability of safe to use this product alone without parental care.
- Never use this product and its parts near any AC electrical outlet or other circuits to avoid the potential risk of electric shock.
- Never use this product near any liquid and fire.
- Keep conductive materials away from this product.
- Never store or use this product in any extreme environments such as extreme hot or cold, high humidity and etc.
- Remember to turn off circuits when not in use this product or when left.
- Do not touch any moving and rotating parts of this product while they are operating.
- Some parts of this product may become warm to touch when used in certain circuit designs. This is normal. Improper operation may cause excessively overheating.
- Using this product not in accordance with the specification may cause damage to the product.

About

Freenove is an open-source electronics platform. Freenove is committed to helping customer quickly realize the creative idea and product prototypes, making it easy to get started for enthusiasts of programing and electronics and launching innovative open source products. Our services include:

- Electronic components and modules
- Learning kits for Arduino
- Learning kits for Raspberry Pi
- Learning kits for Technology
- Robot kits
- Auxiliary tools for creations

Our code and circuit are open source. You can obtain the details and the latest information through visiting the following web sites:
http://www.freenove.com
https://github.com/freenove

Your comments and suggestions are warmly welcomed, please send them to the following email address:
support@freenove.com

References

You can download the sketches and references used in this product in the following websites:
http://www.freenove.com
https://github.com/freenove
If you have any difficulties, you can send email to technical support for help.

The references for this product is named Freenove Four-wheeled Smart Car Kit for Raspberry Pi, which includes the following folders and files:

- Code Project code
- Datasheet Datasheet
- Readme.md Instructions
- Tutorial.pdf User Manual
- Picture Picture resource
- InstallationPackage Installation packages required for the project

Support

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

- Quality problems of products
- Problems in using products
- Questions for learning and technology
- Opinions and suggestions
- Ideas and thoughts

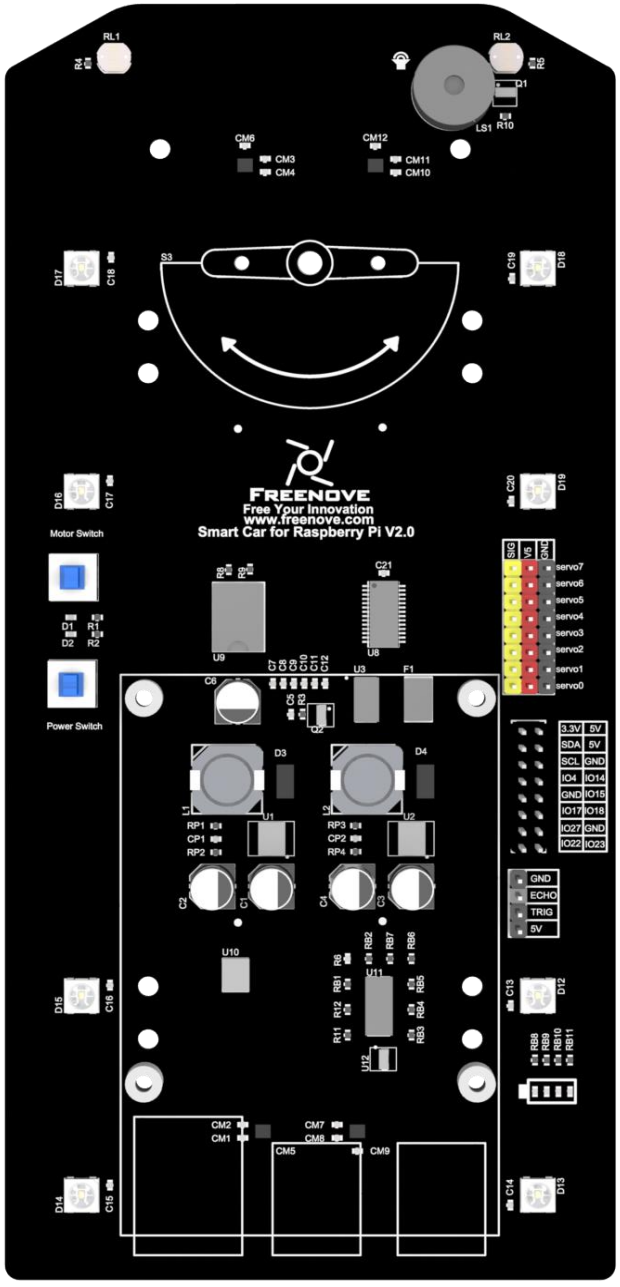
Please send email to:
support@freenove.com
On working day, we usually reply to you within 24 hours.

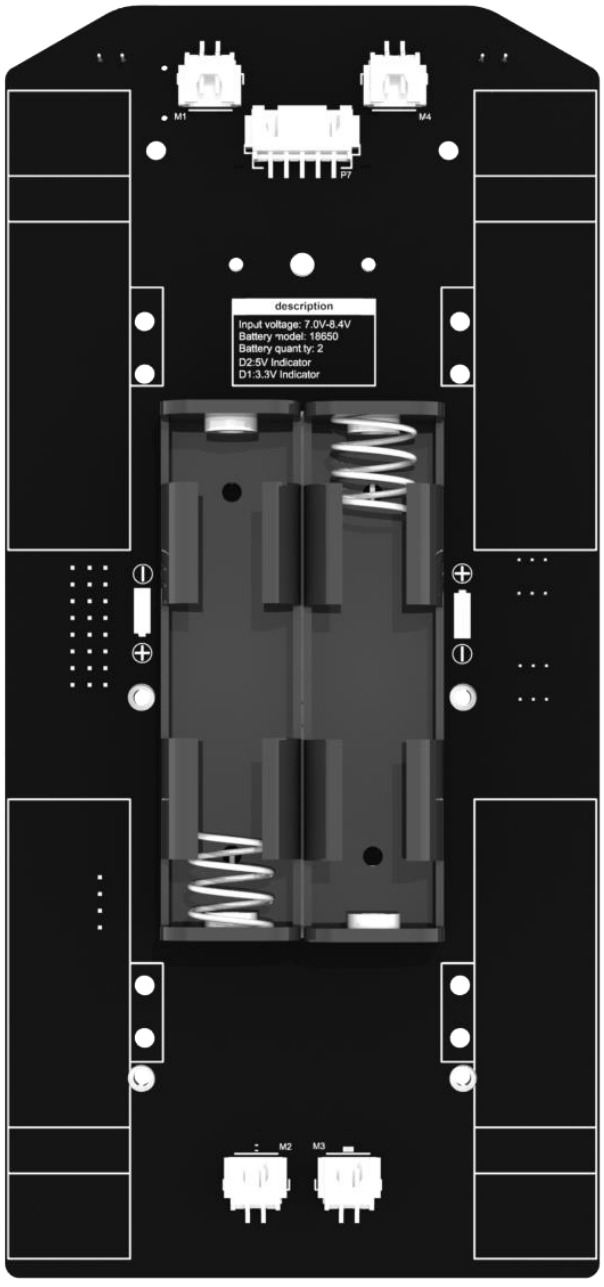
Copyright

Contents

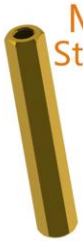
List

4WD Smart Car Board for Raspberry Pi

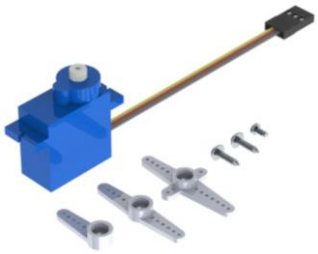




Machinery Parts

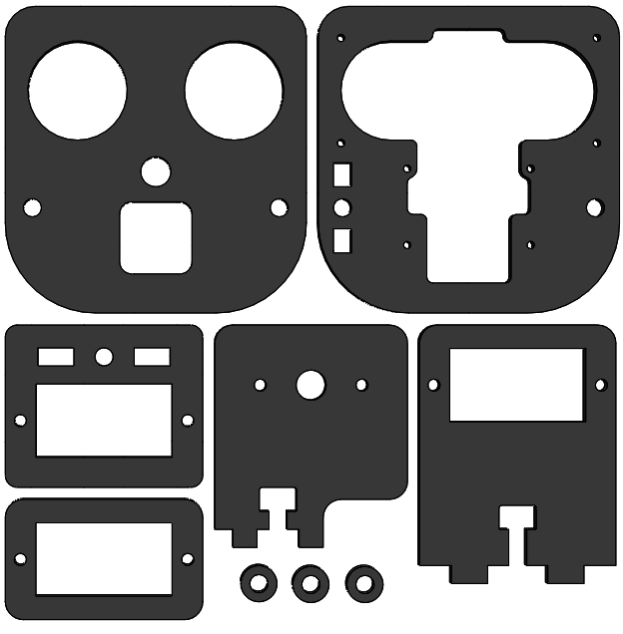
M1.4*4 self-tapping Screw  x12 Freenove	M2.5*4 Screw  x5 Freenove	M3*6 Screw  x5 Freenove	M2.5*8+6 Standoff  x5 Freenove	M3*30 Standoff  x3 Freenove	M2*10 Screw  x5 Freenove
M3*12 Screw  x3 Freenove	M3*14 Screw  x2 Freenove	M2 Nut  x5 Freenove	M3 Nut  x4 Freenove		

Transmission Parts

Servo package x2 	Driven wheel x4 
DC speed reduction motor x4 	Motor bracket package x4 

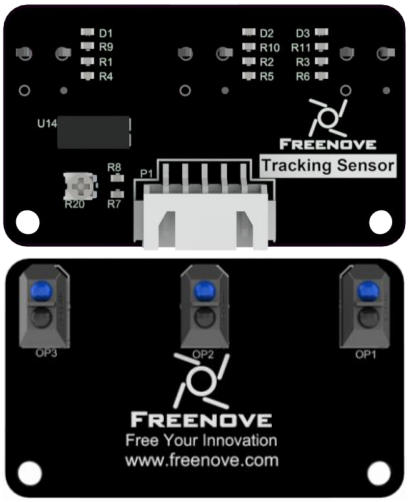
Acrylic Parts

For Pan Tilt

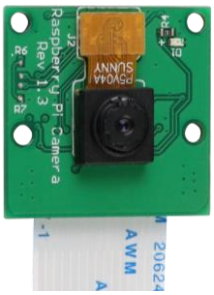


Electronic Parts

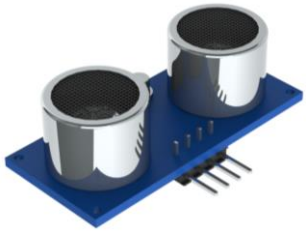
Line tracking module x1



Camera x1



HC-SR04 Ultrasonic Module x1



Connection board

Jumper Wire F/F(4) x1

XH-2.54-5Pin cable x1

Tools

Cross screwdriver x1

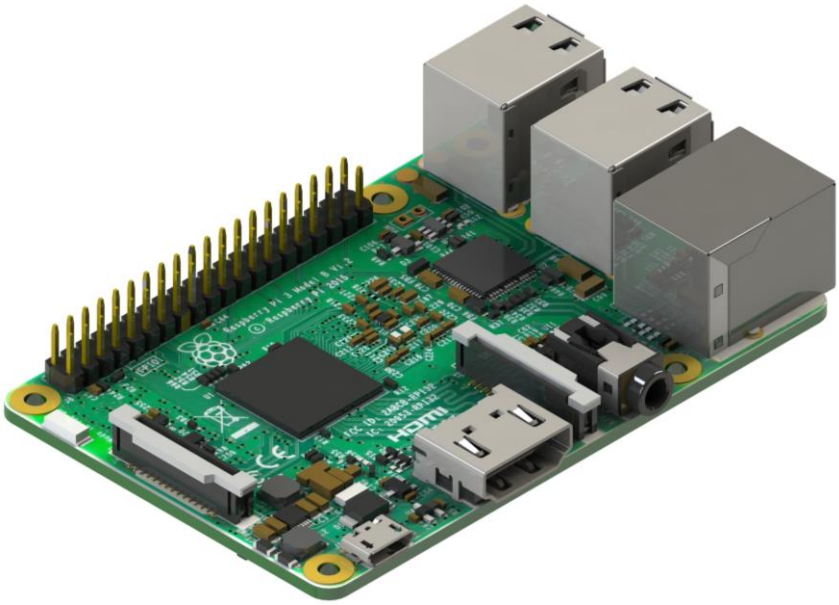
Slotted screwdriver x1

Multifunctional Spanner x1

Self-prepared Parts

Two 18650 lithium batteries without protection board.

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



Preface

Welcome to use Freenove Four-wheeled Smart Car Kit for Raspberry Pi. By using this tutorial, you can make a very cool smart car with many functions.

This kit is based on the popular control panel Pi Raspberry, 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

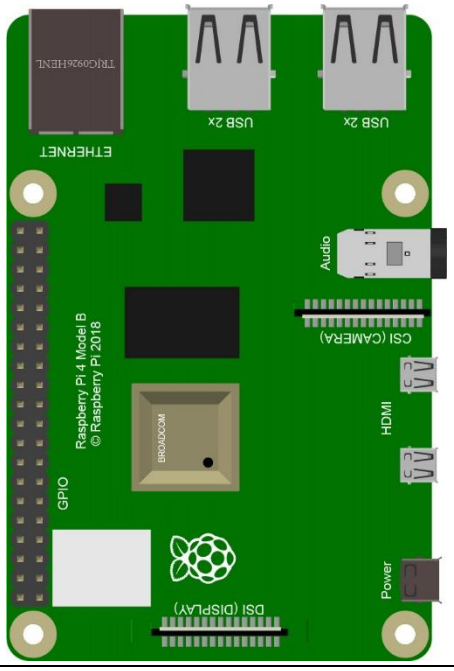
The contents in this book can ensure enthusiastic with little technical knowledge to make the smart car. If you are very interested in Raspberry Pi, and want to learn how to program and build the circuit, please visit our website www.freenove.com or contact us to buy the kits designed for beginners: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

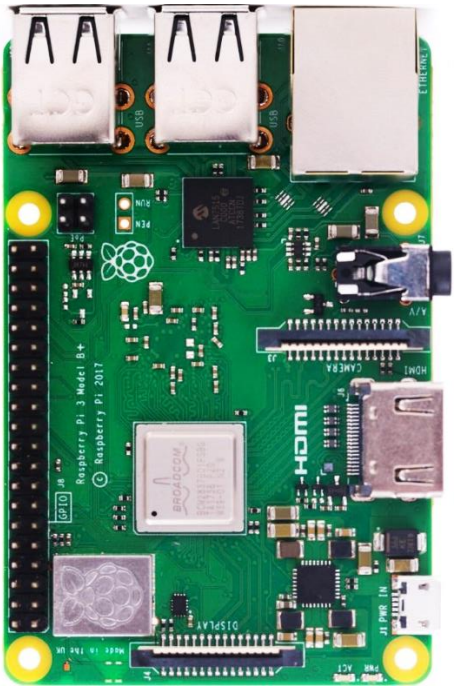
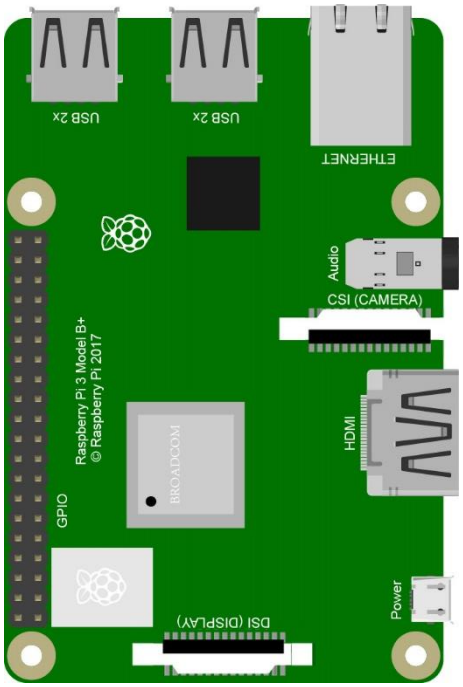
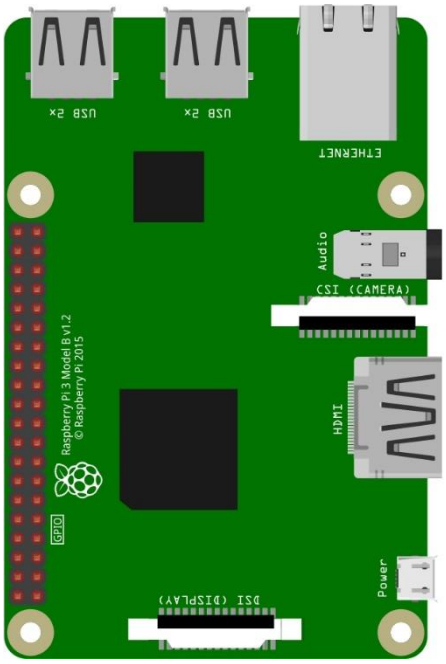
Raspberry Pi

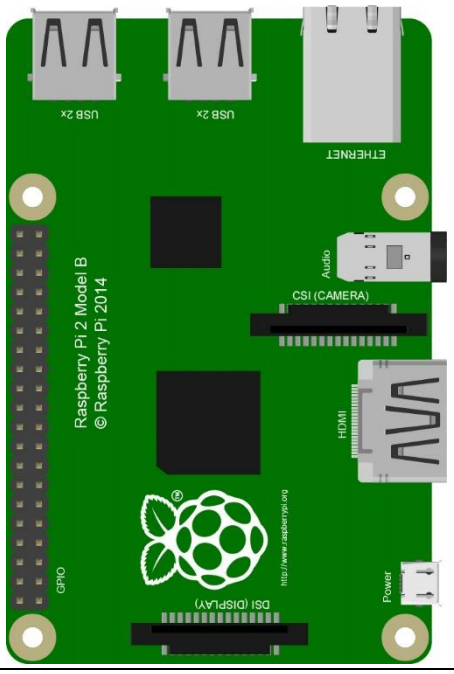
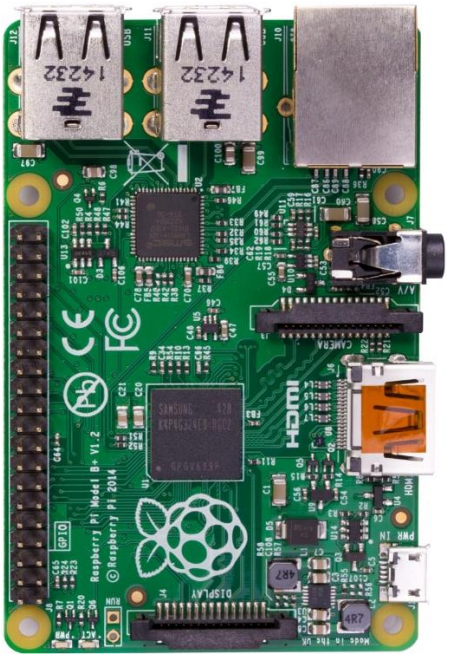
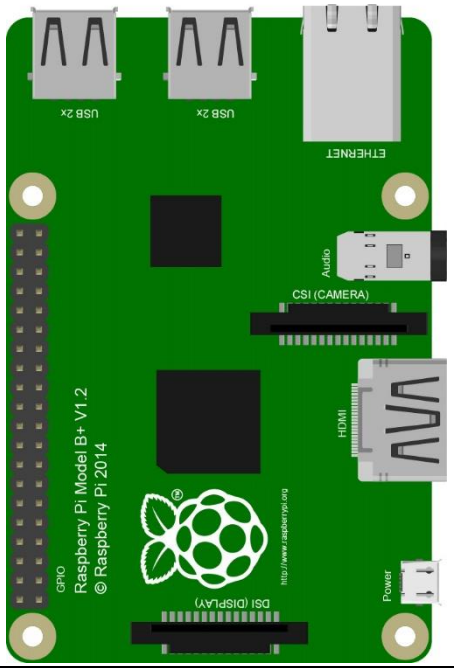
Raspberry Pi (called RPi, RPI, RasPi, the text these words will be used alternately later), a micro-computer with size of a card, quickly swept the world since its debut. It is widely used in desktop workstation, media center, smart home, robots, and even the servers, etc. It can do almost anything, 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.

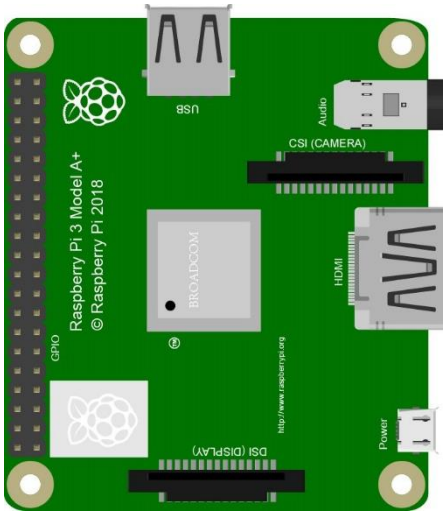
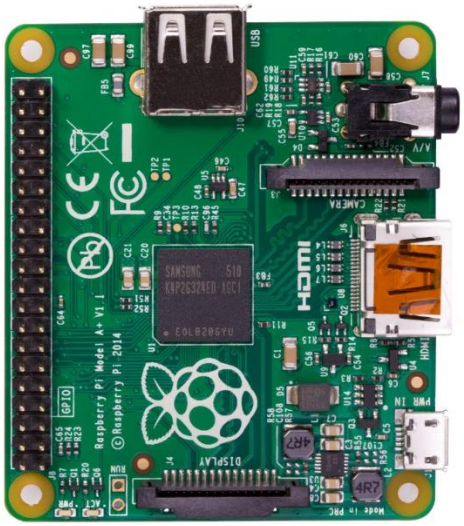
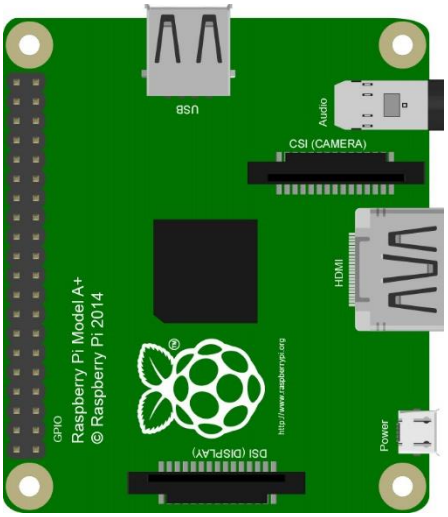
So far, Raspberry Pi has developed to the fourth generation. Changes in versions are accompanied by increase and upgrades in hardware. A type and B type, the first generation of products, have been stopped due to various reasons. Other versions are popular and active and the most important is that they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

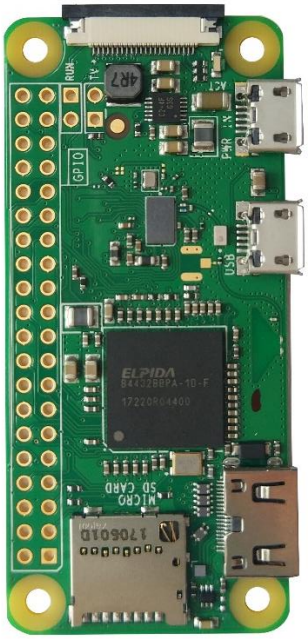
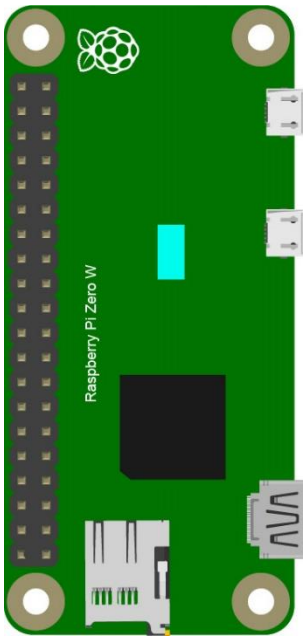
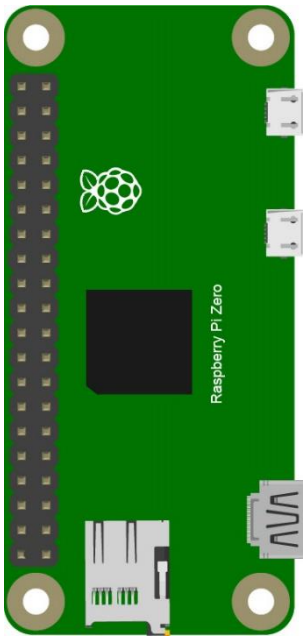
Below are the raspberry pi pictures and model pictures supported by this product.

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

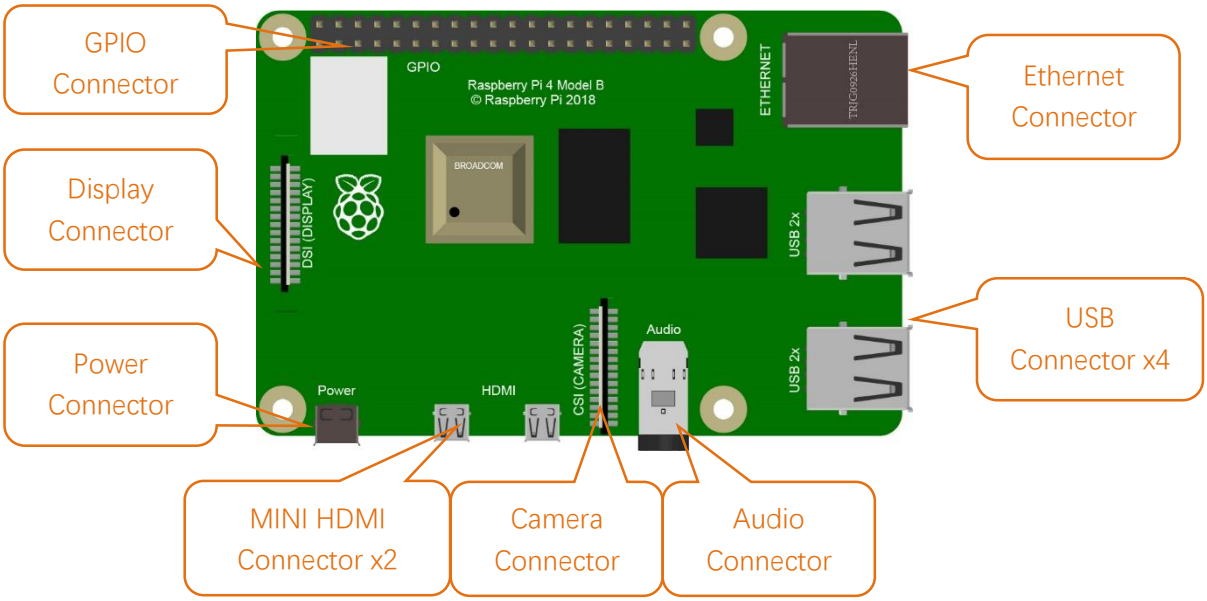
Practicality picture of Raspberry Pi 3 Model B+:	Model diagram of Raspberry Pi 3 Model B+:
	
Practicality picture of Raspberry Pi 3 Model B:	Model diagram of Raspberry Pi 3 Model B:
	

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+: 

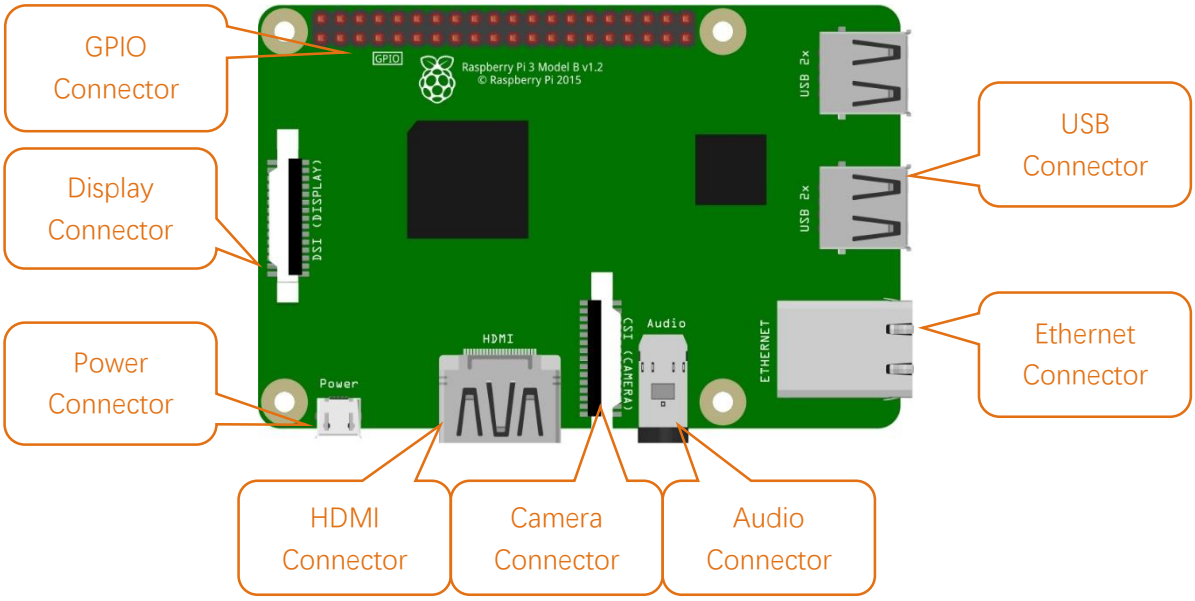
Practicality picture of Raspberry Pi 3 Model A+: 	Model diagram of Raspberry Pi 3 Model A+: 
Practicality picture of Raspberry Pi 1 Model A+: 	Model diagram of Raspberry Pi 1 Model A+: 

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: 

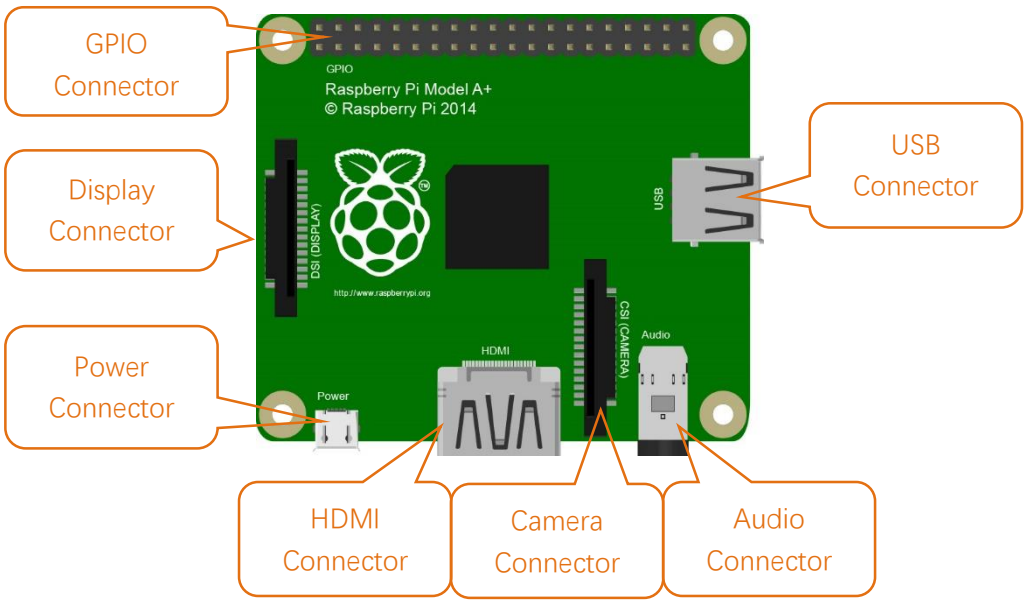
Hardware interface diagram of RPi 4B is shown below:



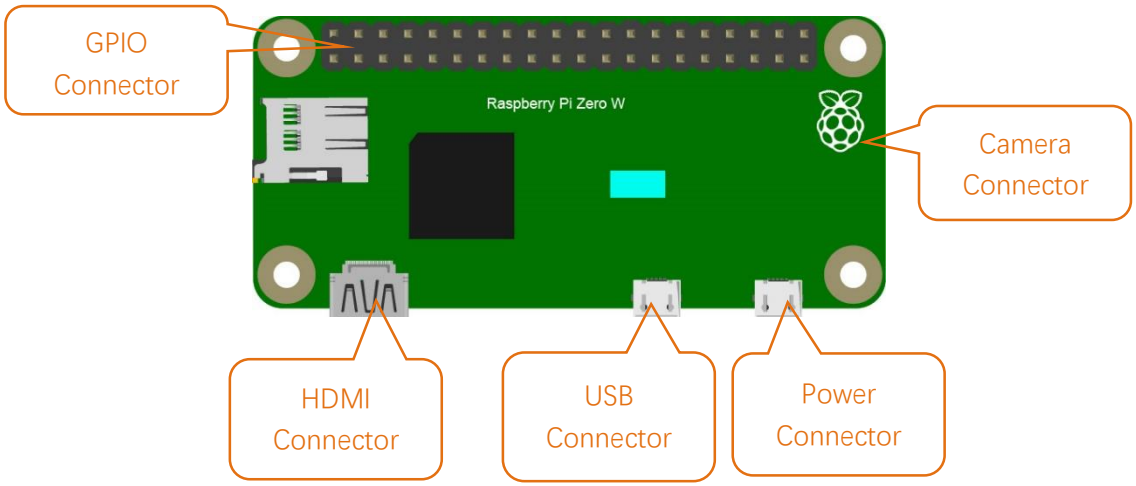
Hardware interface diagram of RPi 3B+/3B/2B/1B+ are shown below:



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



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

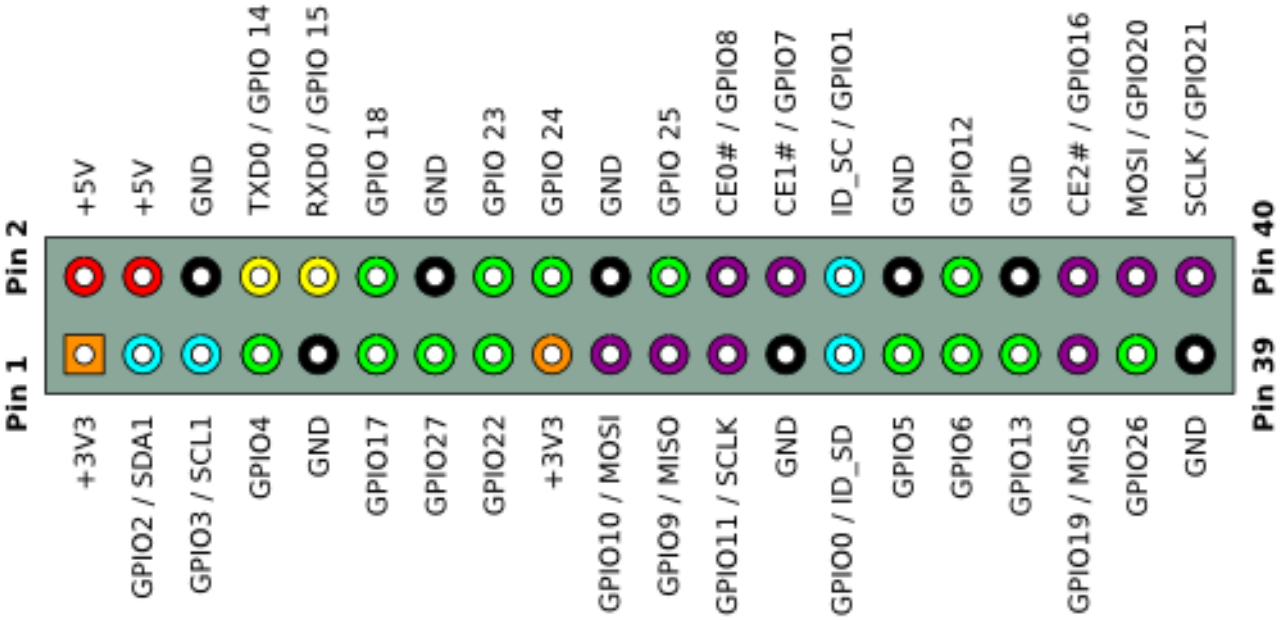


GPIO

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

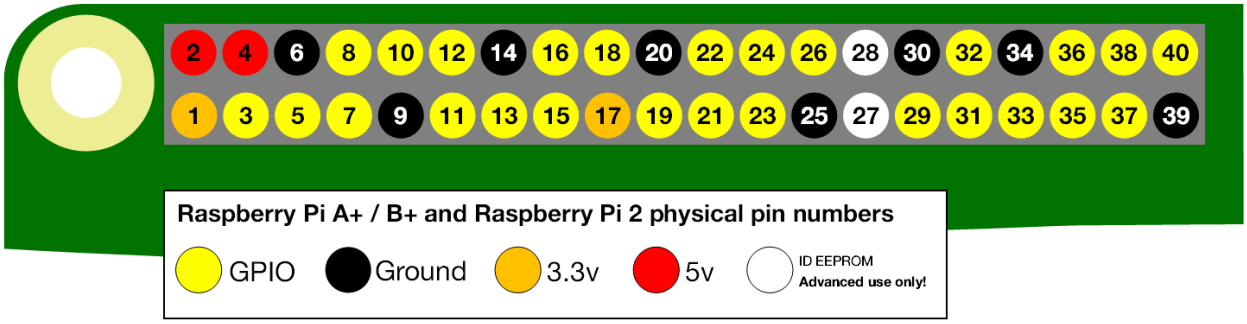
Raspberry Pi CPU use BCM2835/BCM2836/BCM2837of Broadcom. GPIO pin number is set by chip manufacturer. These are the GPIO pins as that computer recognizes. The numbers are unordered and don't make any sense to humans. You will need a printed reference or a reference board that fits over the pins.
Each 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 A+, B+, 2B, 3B, 3B+, 4B, Zero
8	R1:0/R2:2	SDA	3 | 4	5v	—	—	
9	R1:1/R2:3	SCL	5 | 6	0v	—	—	
7	4	GPIO7	7 | 8	TxD	14	15	
—	—	0v	9 | 10	RxD	15	16	
0	17	GPIO0	11 | 12	GPIO1	18	1	
2	R1:21/R2:27	GPIO2	13 | 14	0v	—	—	
3	22	GPIO3	15 | 16	GPIO4	23	4	
—	—	3.3v	17 | 18	GPIO5	24	5	
12	10	MOSI	19 | 20	0v	—	—	
13	9	MISO	21 | 22	GPIO6	25	6	
14	11	SCLK	23 | 24	CE0	8	10	
—	—	0v	25 | 26	CE1	7	11	
30	0	SDA.0	27 | 28	SCL.0	1	31	
21	5	GPIO.21	29 | 30	0V			
22	6	GPIO.22	31 | 32	GPIO.26	12	26	
23	13	GPIO.23	33 | 34	0V			
24	19	GPIO.24	35 | 36	GPIO.27	16	27	
25	26	GPIO.25	37 | 38	GPIO.28	20	28	
		0V	39 | 40	GPIO.29	21	29	
wiringPi Pin	BCM GPIO	Name	Header	Name	BCM GPIO	wiringPi Pin	

For Pi B

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

You can also use the following command to view their 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---+---+------+---------+-----+-----+
```

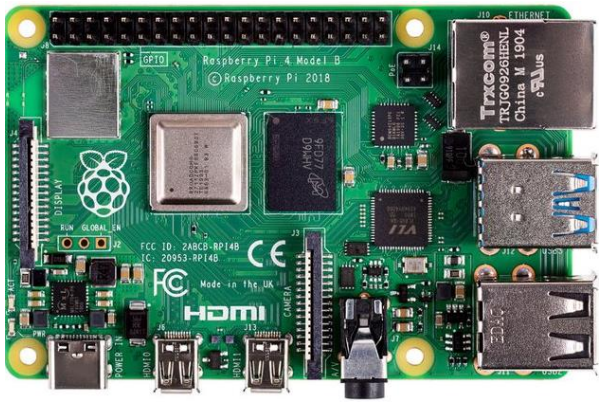
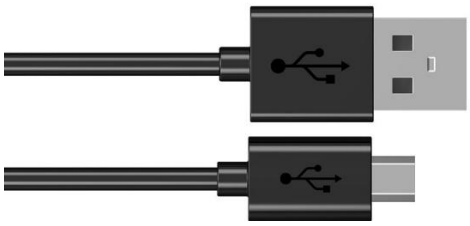
For more details about wiringPi, please refer to http://wiringpi.com/ .

Install the System

Firstly, install a system for your RPi.

Component List

Required Components

Any Raspberry Pi 	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 

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. Mini-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.

<table>
<tr><th></th><th>Pi Zero</th><th>Pi Zero W</th><th>Pi A+</th><th>Pi 3A+</th><th>Pi B+/2B</th><th>Pi 3B/3B+/4B</th></tr>
<tr><td>Monitor</td><td>Yes</td><td>Yes</td><td>Yes</td><td>Yes</td><td>Yes</td><td>Yes</td></tr>
<tr><td>Mouse</td><td>Yes</td><td>Yes</td><td>Yes</td><td>Yes</td><td>Yes</td><td>Yes</td></tr>
<tr><td>Keyboard</td><td>Yes</td><td>Yes</td><td>Yes</td><td>Yes</td><td>Yes</td><td>Yes</td></tr>
<tr><td>Mini-HDMI to HDMI converter wire</td><td>Yes</td><td>Yes</td><td>No</td><td>No</td><td>No</td><td>No</td></tr>
<tr><td>Micro-USB to USB-A Receptacles converter wire (Micro USB OTG wire)</td><td>Yes</td><td>Yes</td><td>No</td><td>No</td><td>No</td><td>No</td></tr>
<tr><td>USB HUB</td><td>Yes</td><td>Yes</td><td>Yes</td><td>Yes</td><td>No</td><td>No</td></tr>
<tr><td>USB transferring to Ethernet interface</td><td rowspan="2">select one from two or select two from two</td><td>optional</td><td rowspan="2">select one from two or select two from two</td><td>optional</td><td>Internal Integration</td><td rowspan="2">Internal Integration</td></tr>
<tr><td>USB Wi-Fi receiver</td><td>Internal Integration</td><td>Internal Integration</td><td>optional</td></tr>
</table>

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.

	Pi Zero	Pi Zero W	Pi A+	Pi 3A+	Pi B+/2B	Pi 3B/3B+/4B
Micro-USB to USB-A Receptacles converter wire (Micro USB OTG wire)	Yes	Yes	No	NO		
USB transferring to Ethernet interface	Yes	Yes	Yes			

Raspbian System

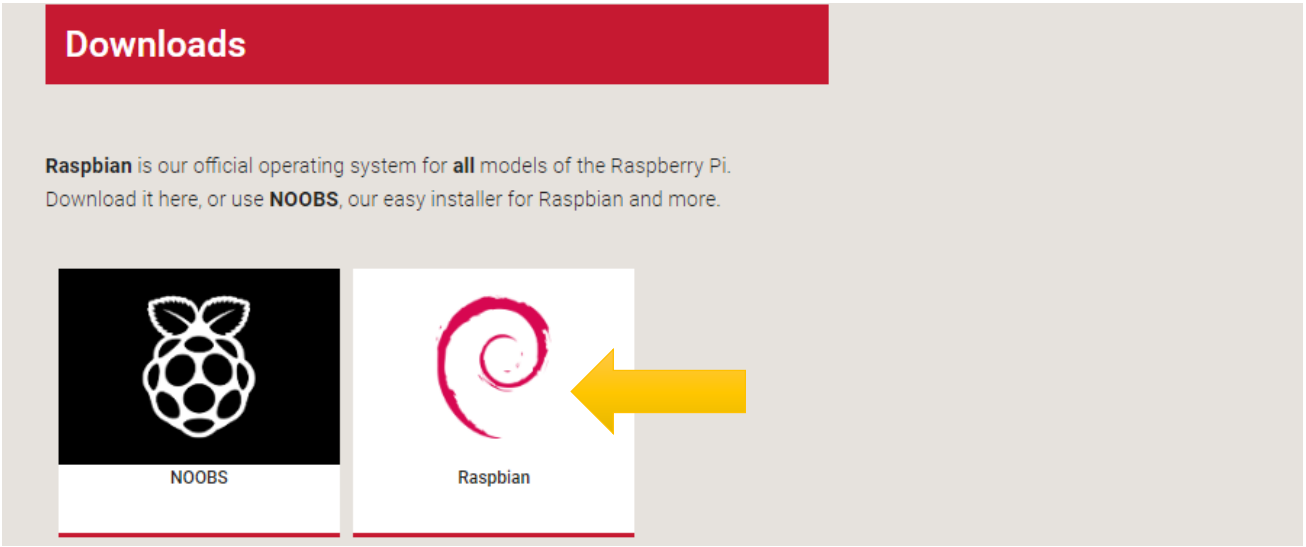
Tool and System image

Software Tool

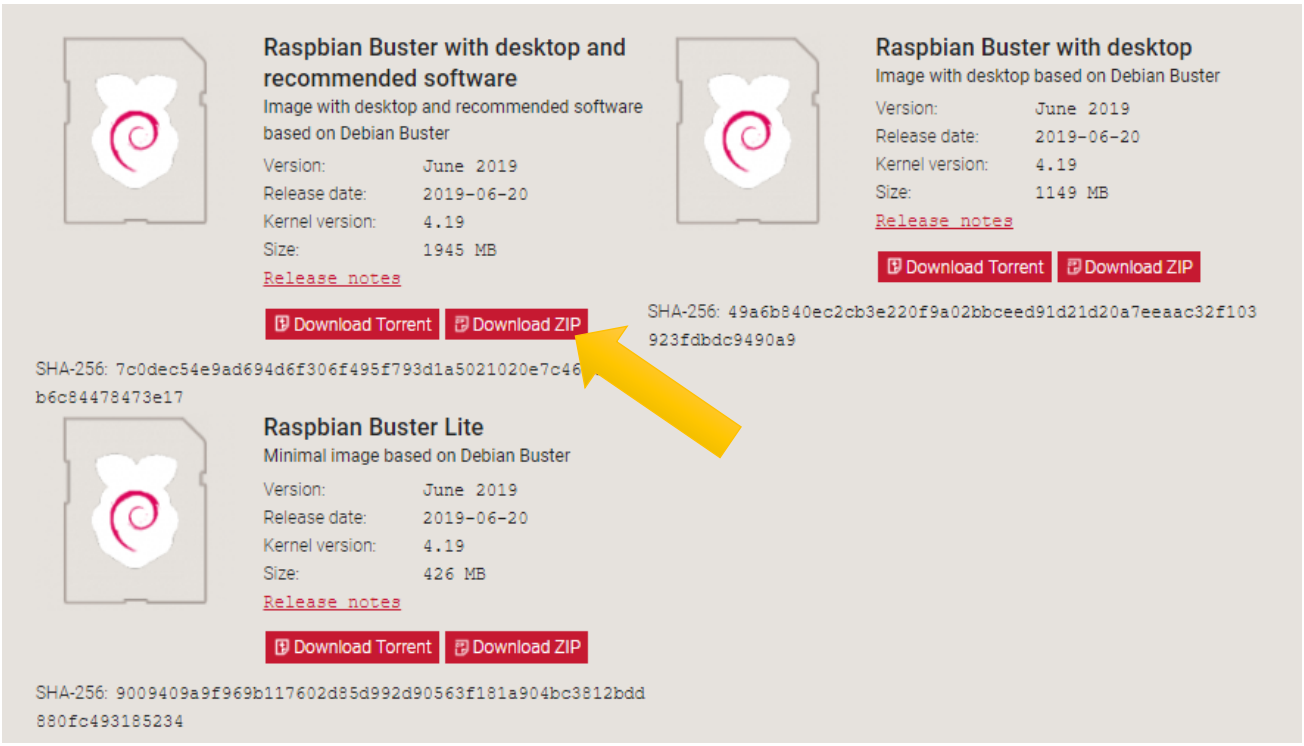
A tool Disk Imager Win32 is required to write system. You can download and install it through visiting the web site: https://sourceforge.net/projects/win32diskimager/

Selecting System

Visit RPi official website (https://www.RaspberryPi.org/), click "Downloads" and choose to download "RASPBIAN". RASPBIAN supported by RPI is an operating system based on Linux, which contains a number of contents required for RPi. We recommended RASPBIAN system to beginners. All projects in this tutorial are operated under the RASPBIAN system.



https://www.raspberrypi.org/downloads/raspbian/

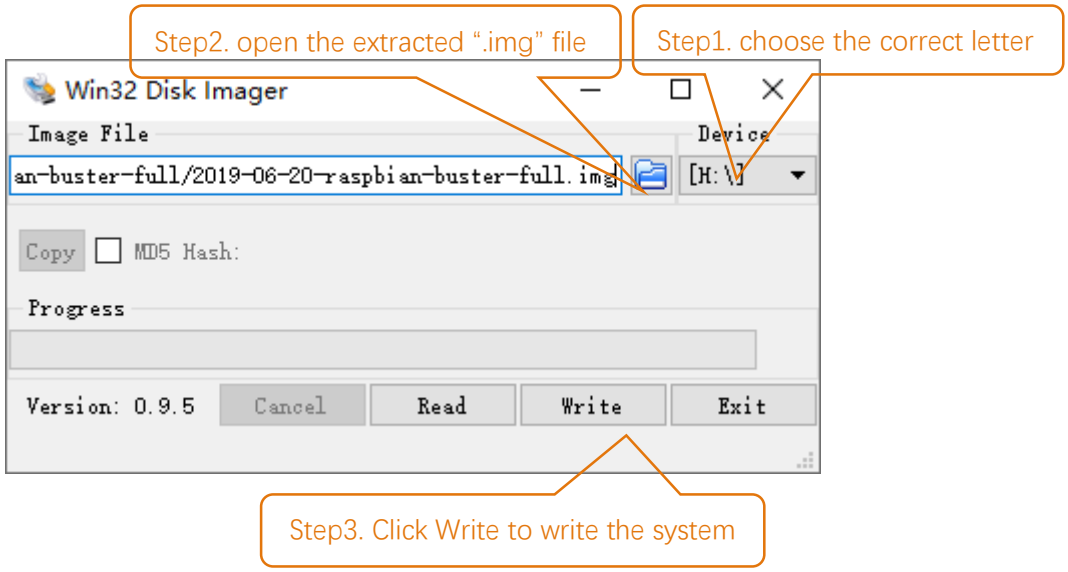


After download, extract file with suffix (.img). Preparation is ready to start making the system.

Write System to Micro SD Card

First, put your Micro SD card into card reader and connect it to USB port of PC. Then open Win32 disk imager, choose the correct letter of your Micro SD Card (here is “H”), open the extracted “.img” file and then click the "Write".

Note: If there is already a system in your card, the computer will prompt you to format it. Just ignore the prompt and write it directly.



Start Raspberry Pi

If you don't have a spare monitor, please jumper to next section. If you have a spare monitor, please follow steps in this section.

After the system is written successfully, take out Micro SD Card and put it into the card slot of RPi. Then connect RPi to screen through the HDMI, to mouse and keyboard through the USB port, to network cable through the network card interface and to the power supply. Then your RPi starts initially. Later, you need to enter the user name and password to login. The default user name: pi; password: raspberry. Enter and login. After login, you can enter the following interface.

Now, you have successfully installed the RASPBIAN operating system for your RPi.

Remote desktop & VNC

If you don't have a spare display, mouse and keyboard for your RPi, you can use a remote desktop to share a display, keyboard, and mouse with your PC. Below is how to use remote desktop under the Windows operating system to control RPi.

Under windows, Raspberry Pi can be generally accessed remotely through two applications. The first one is the windows built-in application remote desktop, which corresponds to the Raspberry Pi xrdp service. The second one is the free application VNC Viewer, which corresponds to the VNC interface of Raspberry Pi. Each way has its own advantages. You can choose either one or two.

Windows	Raspberry Pi
Remote Desktop Connection	Xrdp
VNC Viewer	VNC

VNC Viewer can not only run under Windows, but also under system MAC, Linux, IOS, Android and so on.

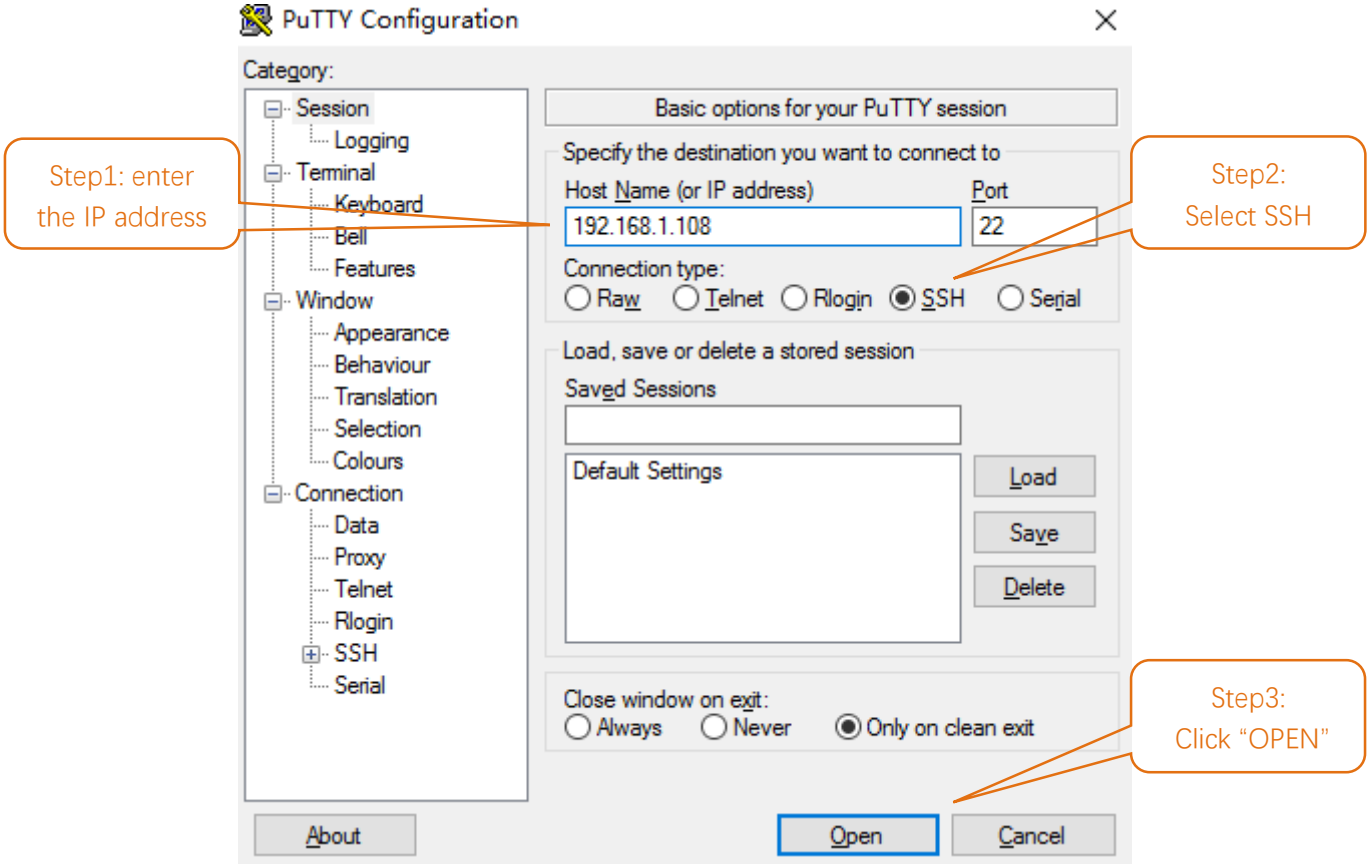
SSH

Under previous Raspbian system, SSH is opened by default. Under the latest version of Raspbian system, it is closed by default. So you need to open it first.

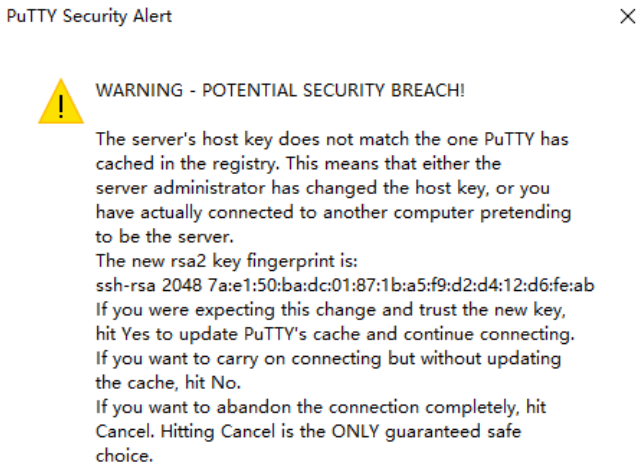
Method: after the system is written. Create a folder named “ssh” under generated boot disk, then the SSH connection will be opened.

And then, download the tool software Putty. Its official address: http://www.putty.org/
Or download it here: http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

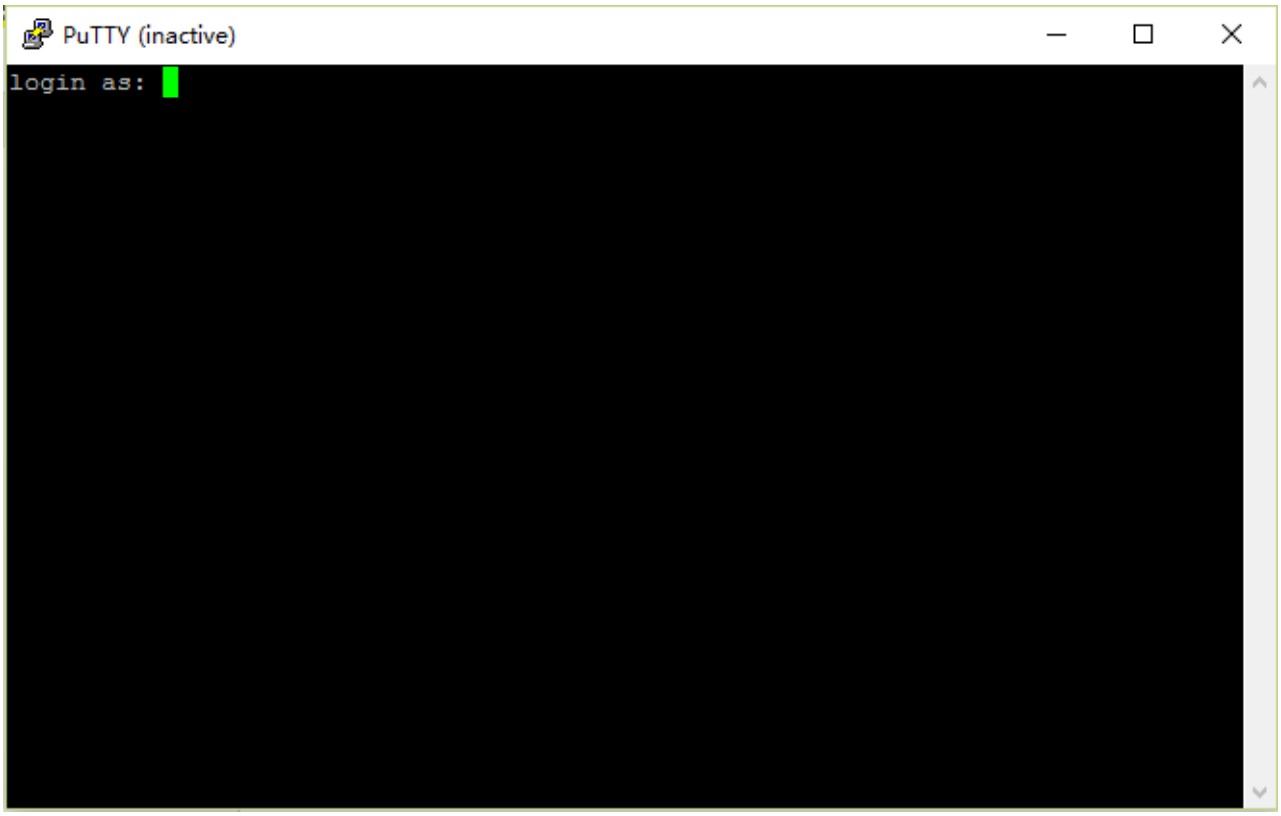
Then use cable to connect your RPi to the routers of your PC LAN, to ensure your PC and your RPi in the same LAN. Then put the system Micro SD Card prepared before into the slot of the RPi and turn on the power supply waiting for starting RPi. Later, enter control terminal of the router to inquiry IP address named “raspberry pi”. For example, I have inquired to my RPi IP address, and it is “192.168.1.108". Then open Putty, enter the address, select SSH, and then click "OPEN", as shown below:



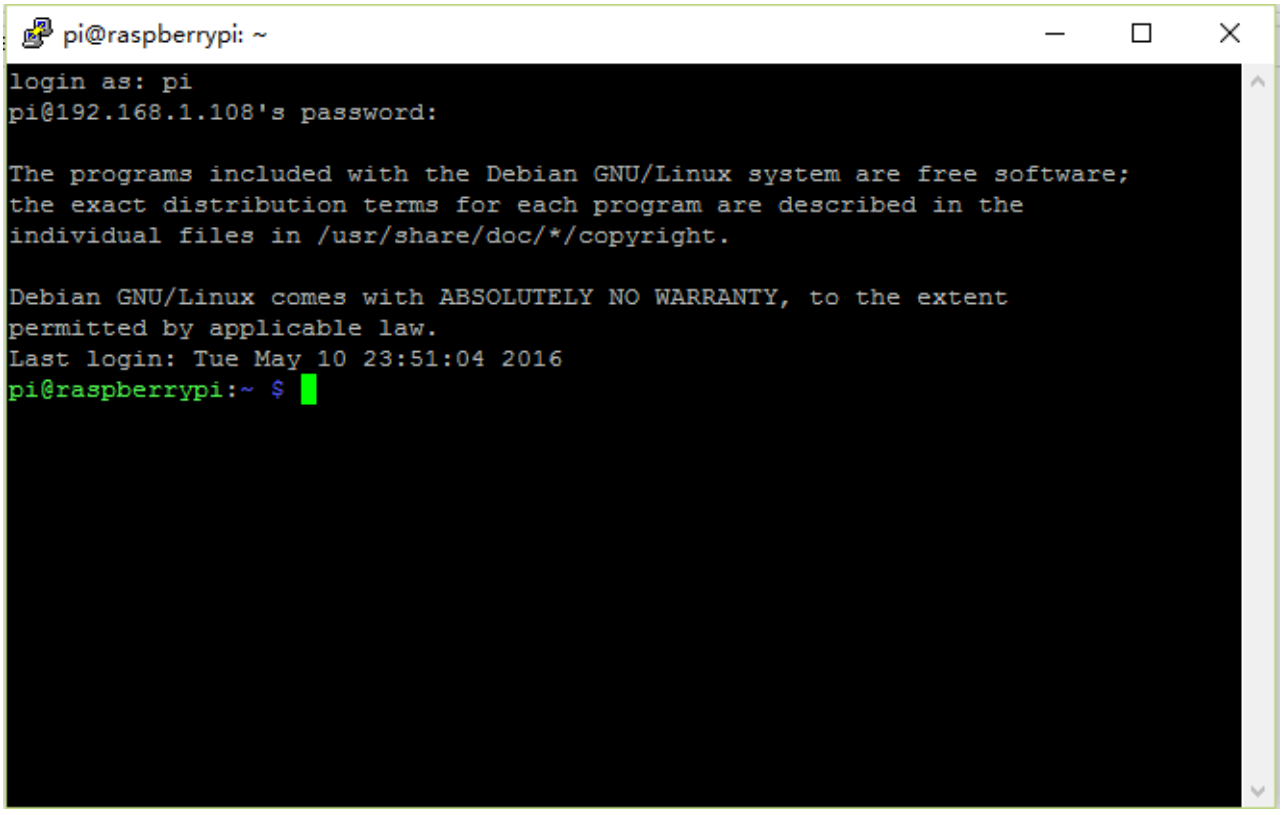
There will appear a security warning at first login. Just click “YES”.



Then there will be a login interface (RPi default user name: **pi**; the password: **raspberry**). When you enter the password, there will be **no display** on the screen. This is normal. After the correct input, press "Enter" to confirm.



Then enter the command line of RPi, which means that you have successfully login to RPi command line mode.



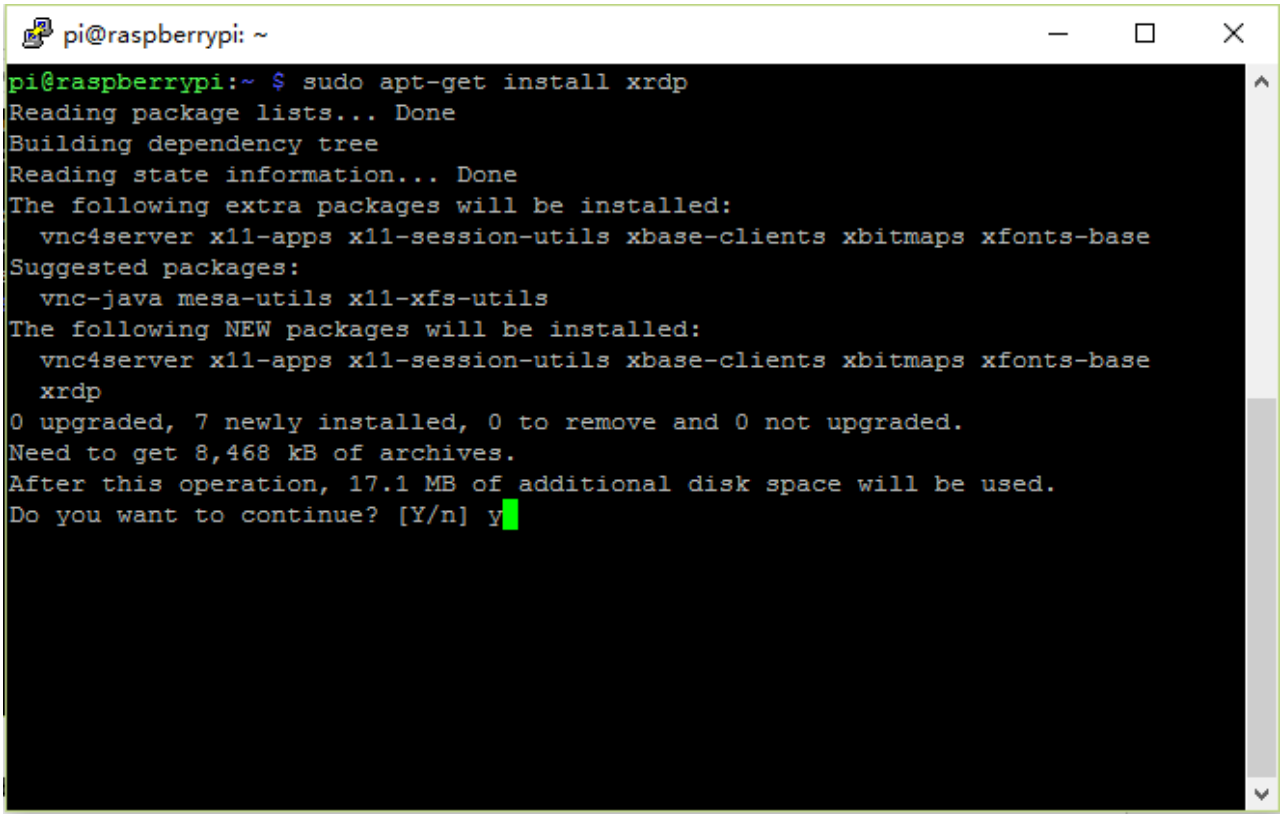
Remote Desktop Connection & xrdp

If you want to use built-in Remote Desktop Connection under Windows, you need install xrdp service on Raspberry Pi.

Next, install a xrdp service, an open source remote desktop protocol(xrdp) server, for RPi. Type the following command, then press enter to confirm:

```
sudo apt-get install xrdp
```

Later, the installation starts.

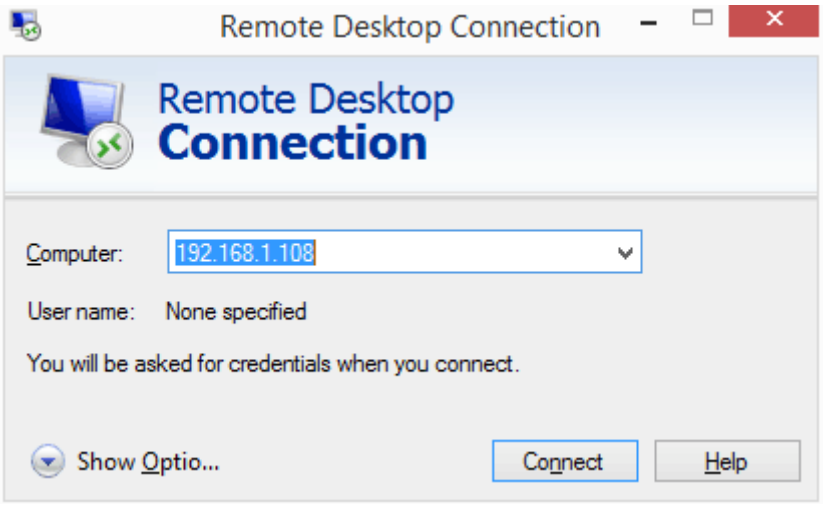


Enter "Y", press key “Enter” to confirm.

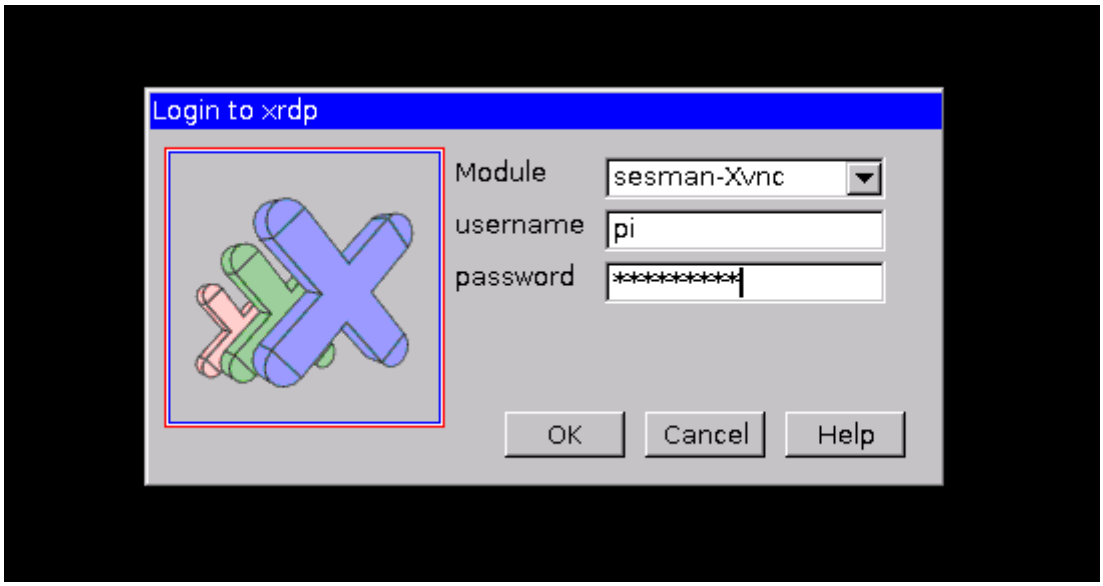
After the installation is completed, you can use Windows remote desktop applications to login to your RPi.

Login to Windows remote desktop

Use "WIN+R" or search function, open the remote desktop application "mstsc.exe" under Windows, enter the IP address of RPi and then click “Connect”.



Later, there will be xrdp login screen. Enter the user name and password of RPi (RPi default user name: pi; password: raspberry) and click “OK”.



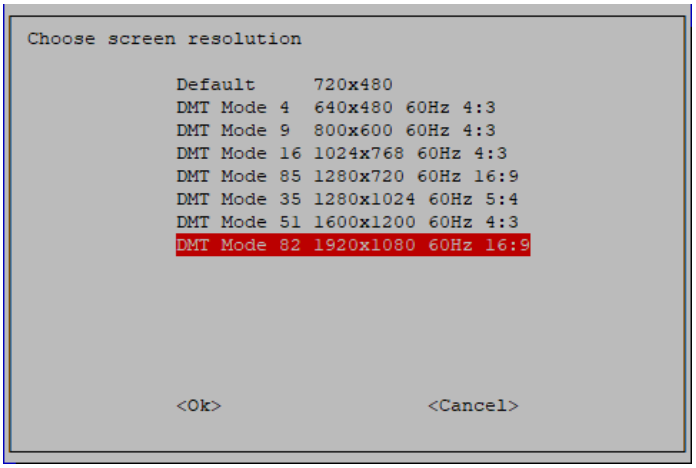
Later, you can enter the RPi desktop system.

Here, you have successfully used the remote desktop login to RPi.

If you think resolution ratio is not OK, you can set a proper resolution ratio on set interface of Raspberry Pi.

```
sudo raspi-config
```

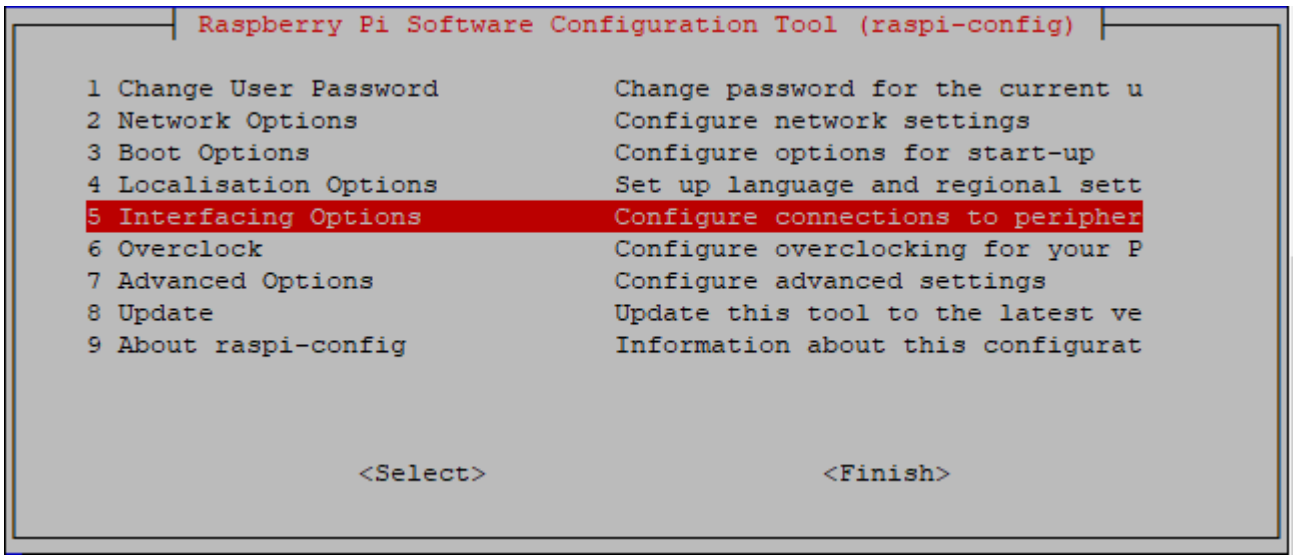
Select 7 Advanced Options→A5 Resolution→proper resolution ratio(set by yourself)→OK. If it needs restart, just restart.

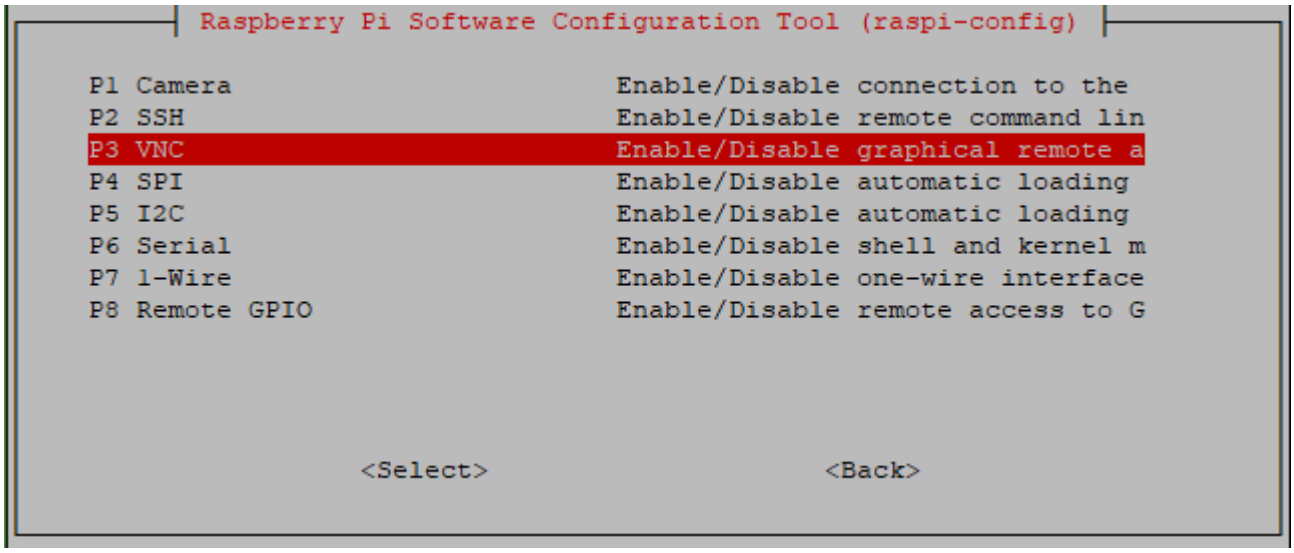


VNC Viewer & VNC

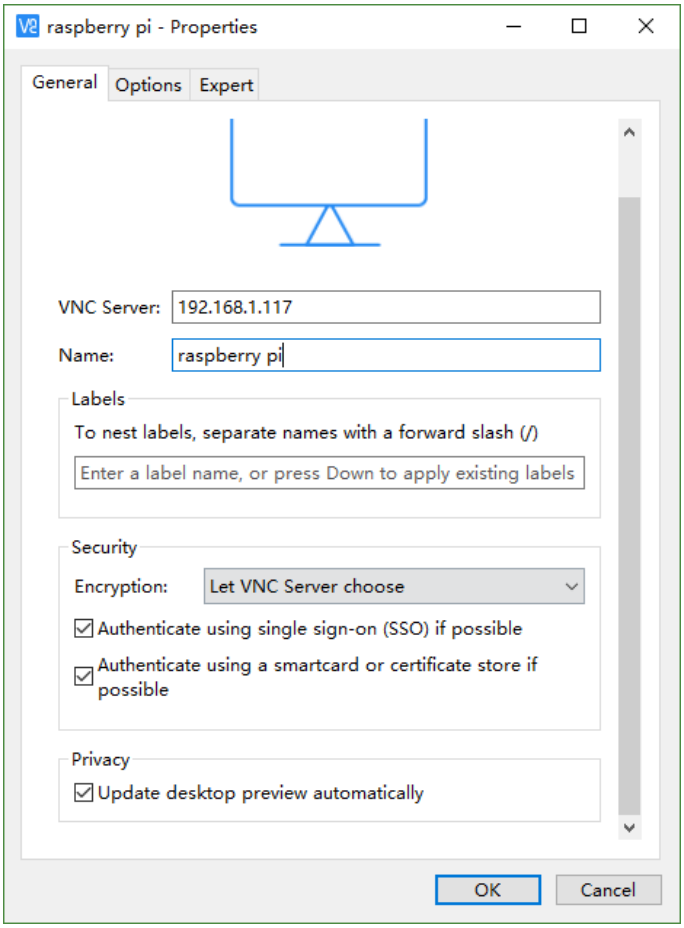
Type the following command. And select 5 Interfacing Options→P3 VNC →Yes→OK→Finish. Here Raspberry Pi may need be restarted, and choose ok. Then open VNC interface.

```
sudo raspi-config
```

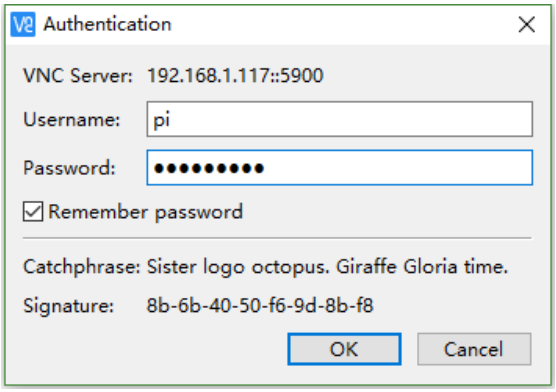




Then download and install VNC Viewer by click following link:
https://www.realvnc.com/en/connect/download/viewer/windows/
After installation is completed, open VNC Viewer. And click File → New Connection. Then the interface is shown below.



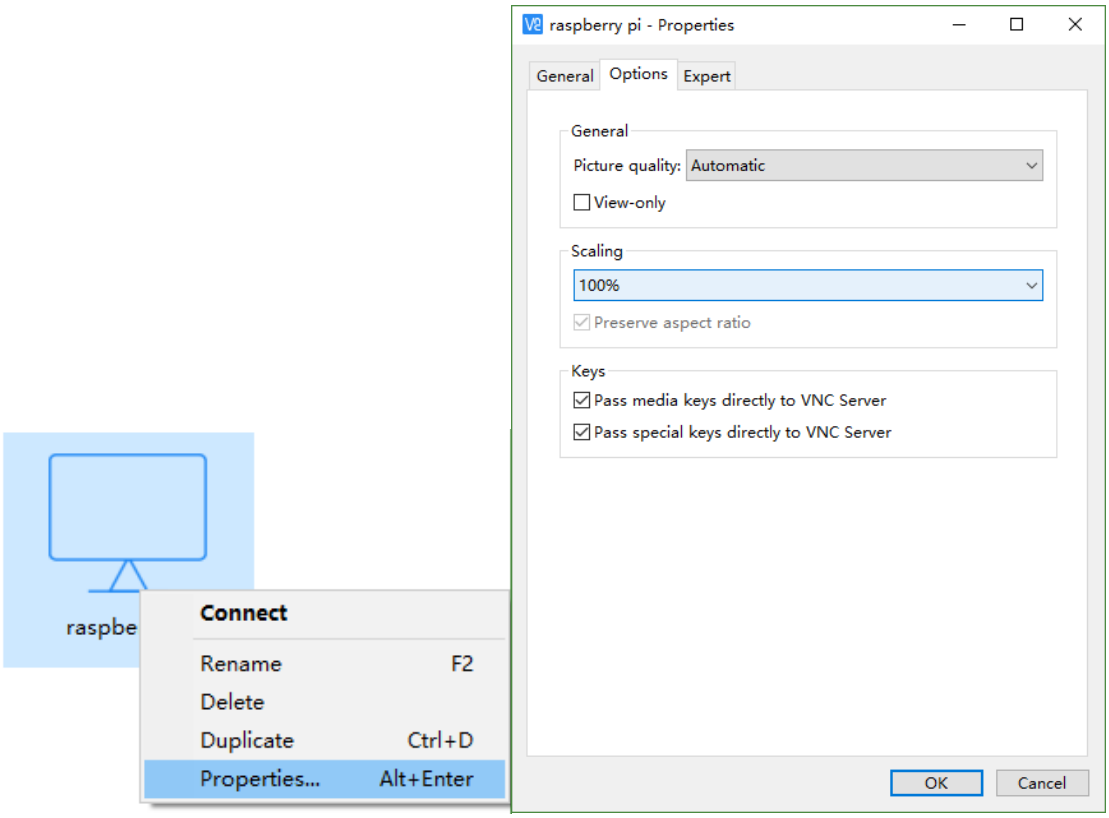
Enter ip address of your Raspberry Pi and fill in a Name. And click OK.
Then on the VNC Viewer panel, double-click new connection you just created, and the following dialog box pops up.



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

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

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.

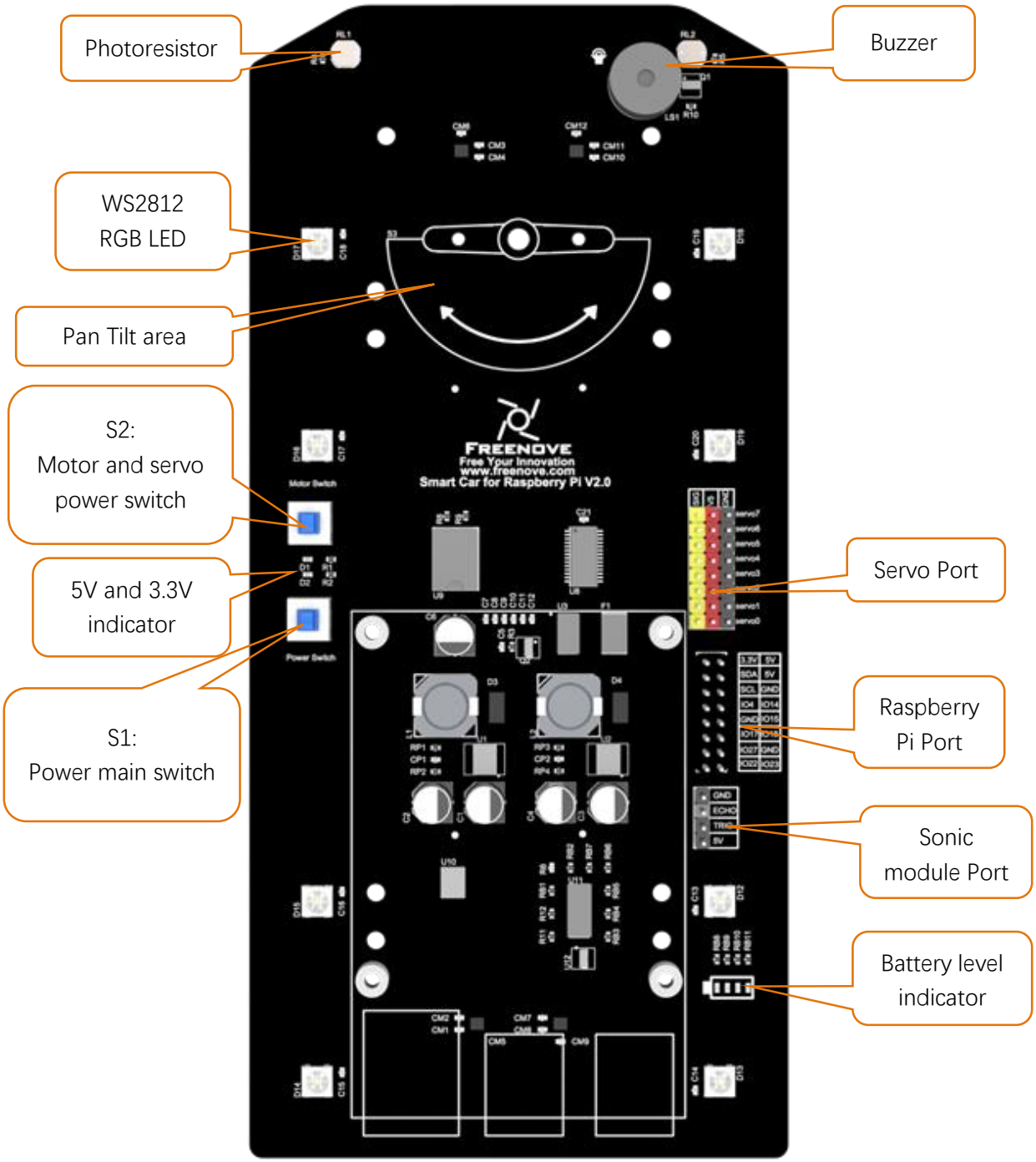
Then continue to do some preparation work: install a GPIO library wiringPi for your RPi.

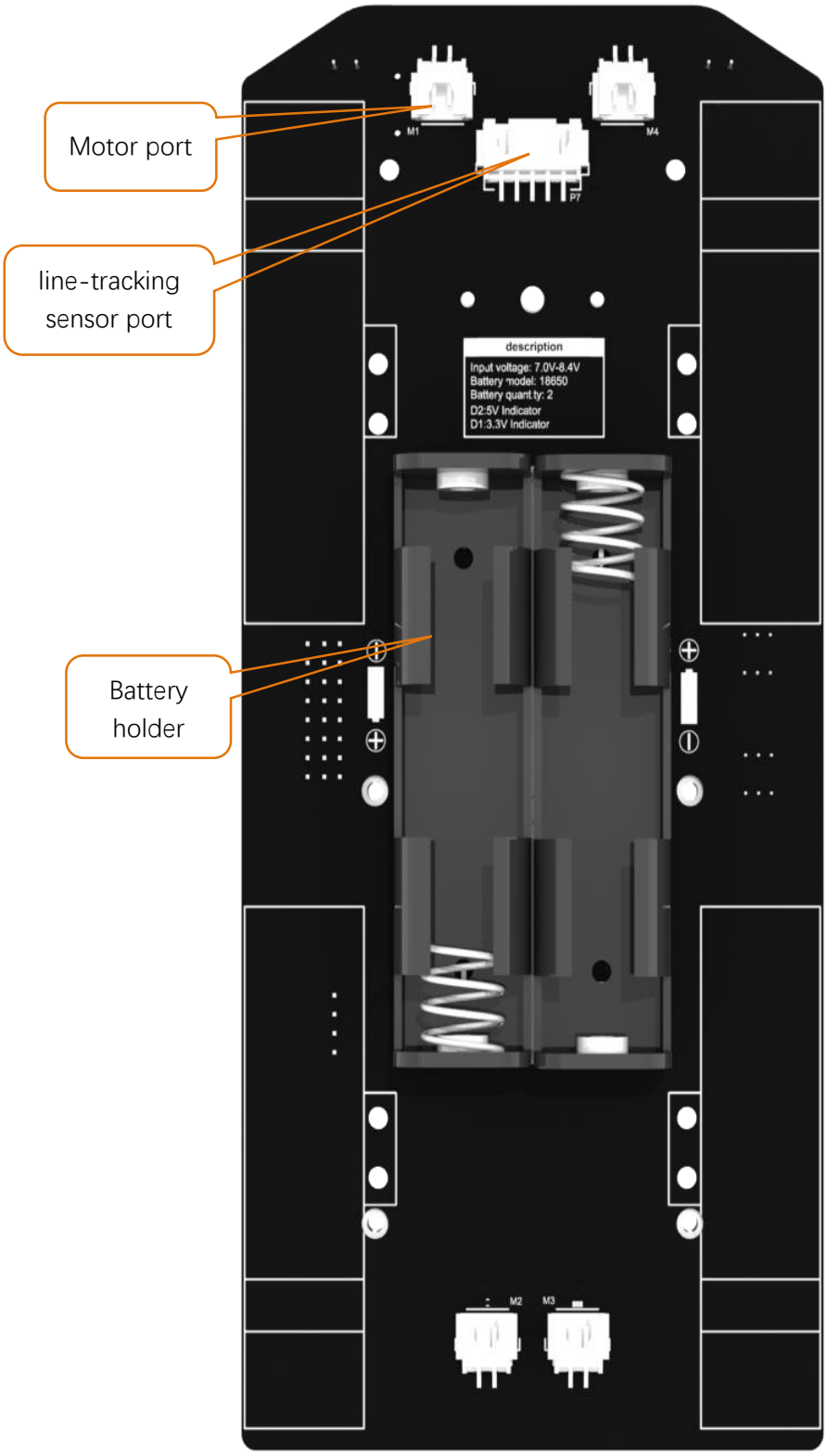
Wi-Fi

Raspberry Pi 4B/3B+/3B integrates a Wi-Fi adaptor. You can use it to connect to your Wi-Fi. Then you can use the wireless remote desktop to control your RPi. This will be helpful for the following work. Raspberry Pi of other models can use wireless remote desktop through accessing an external USB wireless card.

Smart Car Shield for RPi

The smart car board for the Raspberry Pi can communicate with Raspberry Pi through I2C. The positioning holes are applicable to the Raspberry Pi. The following are the features and functions of the smart car board.




Motor port
line-tracking sensor port
Battery holder
M1
M4
P7
description
Input voltage: 7.0V-8.4V
Battery model: 18650
Battery quantity: 2
D2:5V Indicator
D1:3.3V Indicator
M2
M3

- **Smart car board**: It is powered by two 18650 batteries with 7V-8.4V. There are two switches, S1 is main power switch, and S2 is motor and servo power switch. The power switch mainly controls the power supply to the chip on the PCB board, except the motor and servo. When pressed, the 5V and 3.3V indicator lights will be on, while the motor switch controls the motor and the servo.
- **Servo port**: There are 8 servo ports with control accuracy of 0.09 degrees.
- **Raspberry Pi port**: It is used to connect smart car board and Raspberry Pi with connection board, so that the Raspberry Pi can directly control the smart car board.
- **Sonic module port**: The interface of HC-SR04 Ultrasound Module is provided.
- **Battery level indicator**: It consists of four LEDs. When the battery power decreases gradually, the LEDs will be gradually turned off.
- **RGB LED**: There are 8 ws2812 LEDs on the board, which can be controlled separately by raspberry pi.
- **Buzzer**: It is directly controlled by raspberry pi, which is turned on at high level and off at low level.
- **Photoresistor**: It is used to achieve light tracing function of the car.
- **Motor port**: There are four ports, which are used to control the forward and reverse of the motor.
- **Line tracking sensor port**: It is used to connected with Line tracking sensor (Infrared Module) of Car.

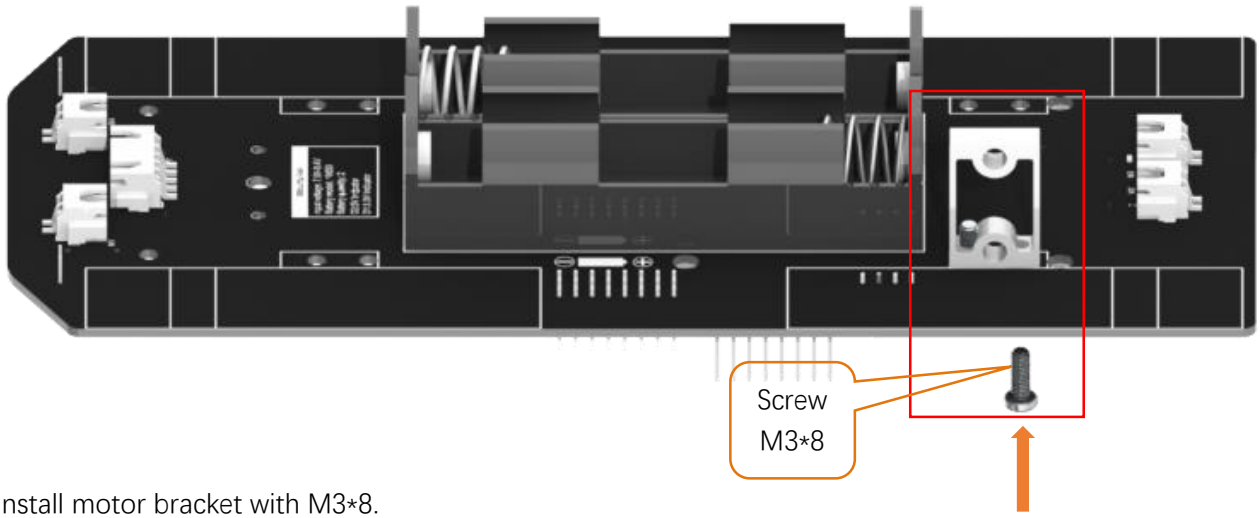
Chapter 1 Assemble Smart Car

Motor and wheel

There is a special fixed bracket bracket to fix motor, which contains an aluminum bracket, two M3*30 screws, two M3*8 screws, and two M3 nuts, as shown below:

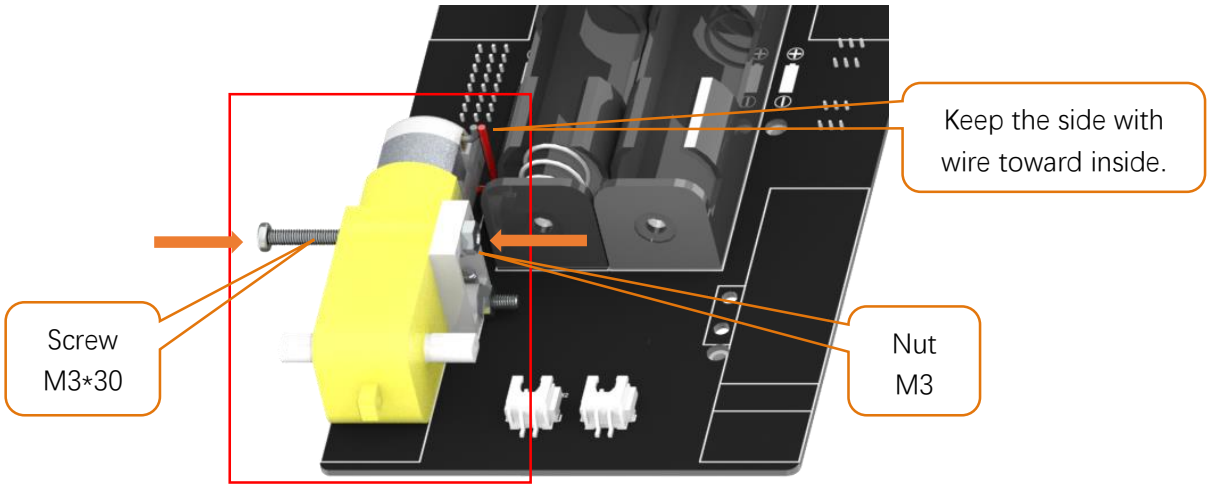
Installation steps:

Step 1



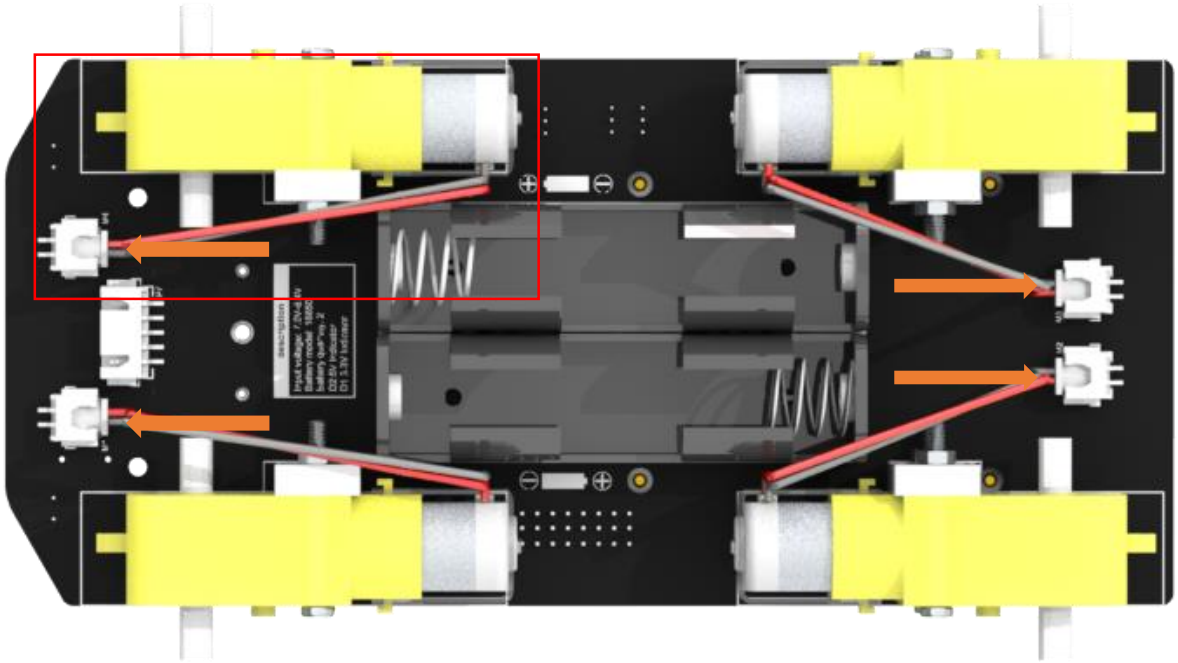
Install motor bracket with M3*8.

Step 2



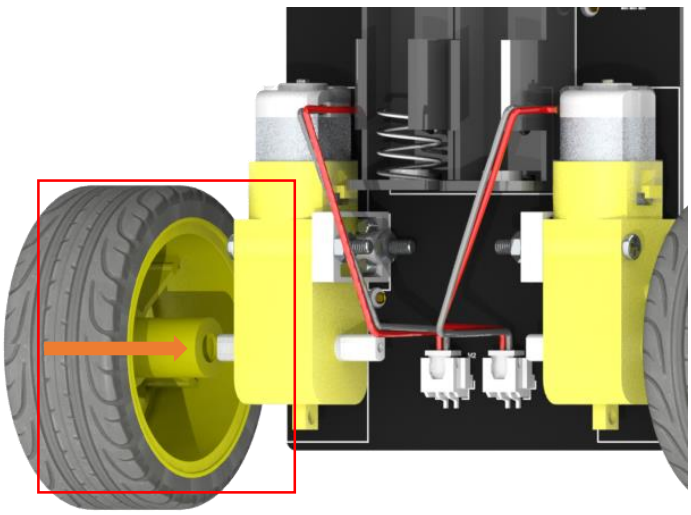
Install motor to motor bracket with screw M3*30 and Nut M3.

Step 3

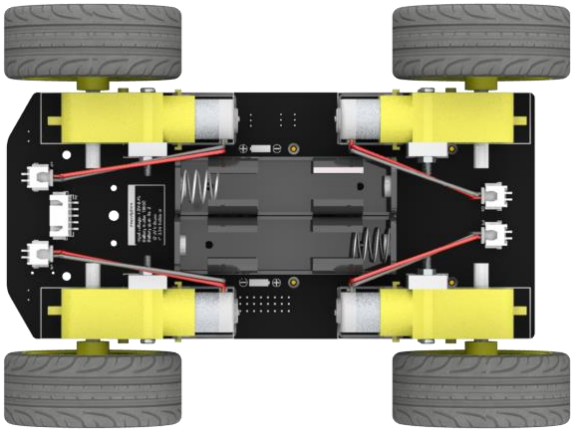


The installation of rest 3 sets of motor is the same. Then connect motor wire to motor port. If you think the wires are too long, you can tie a knot.

Step 4



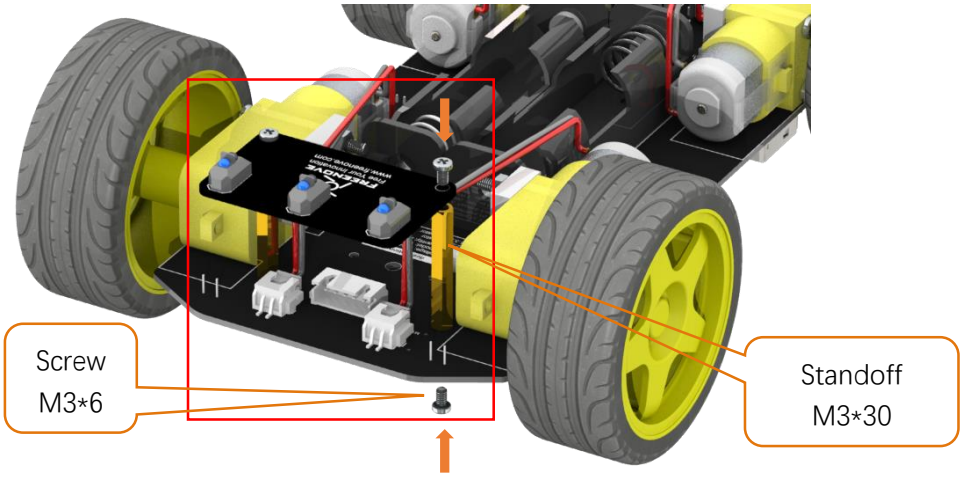
Install 4 wheels to motor. The mounting hole is not a round hole



Now, all motors and wheels are installed successful.

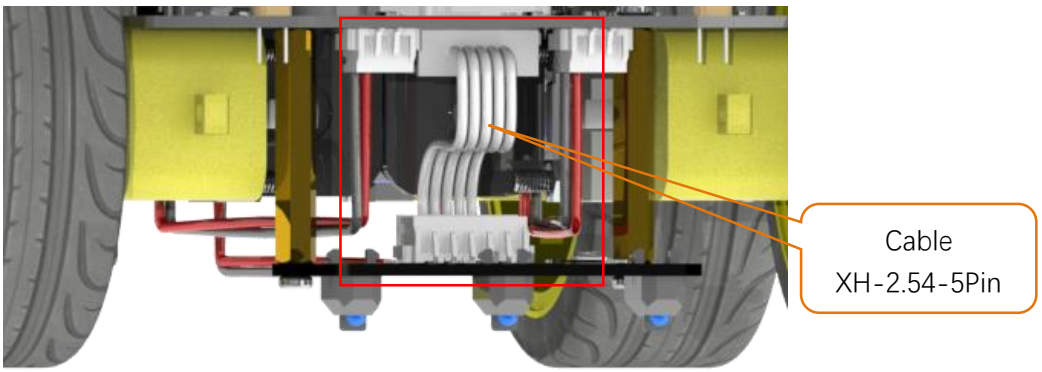
Infrared line tracking module

Step 1



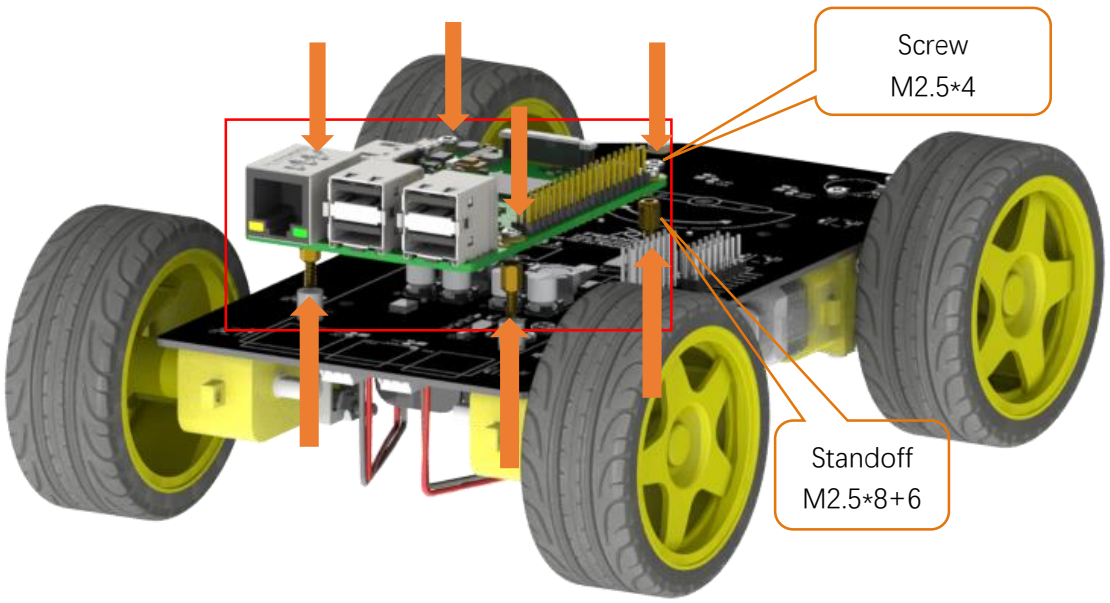
First install two standoff M3*30 on smart car board with screw M3*6. Then install Line tracking module on standoff with screw M3*6.

Step 2



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

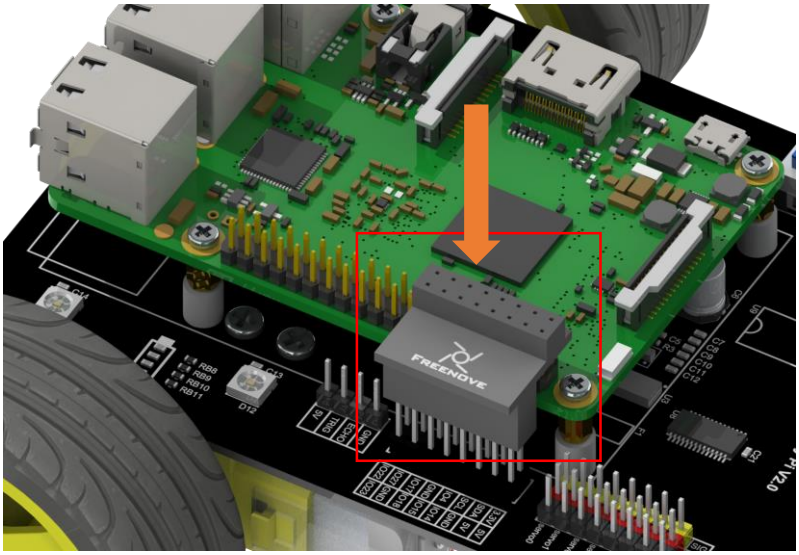
Raspberry Pi



Keep Raspberry Pi consistent with the screen printing on smart car board. First install four standoff M2.5*8+6 to the nuts on smart car board. Then install Raspberry Pi to standoff with four screw M2.5*4.

Connection board

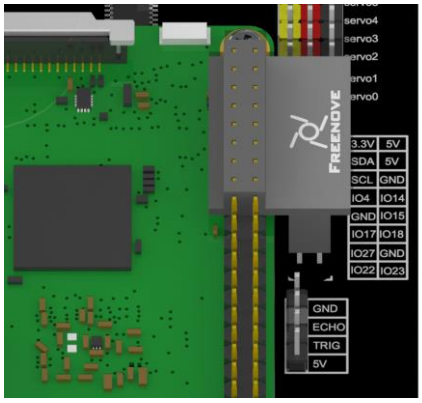
Step 1



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

Note, please don't reverse it. Or the Raspberry Pi will be damaged.

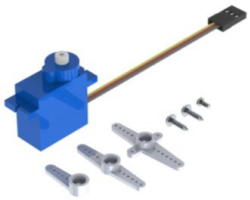
Step 2

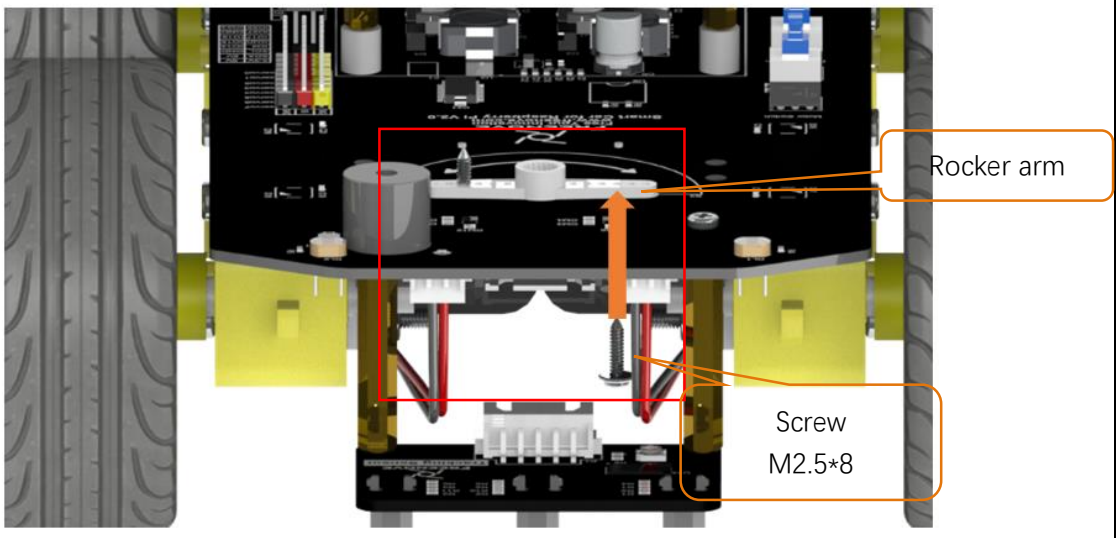


Press connection board to make connection stable.

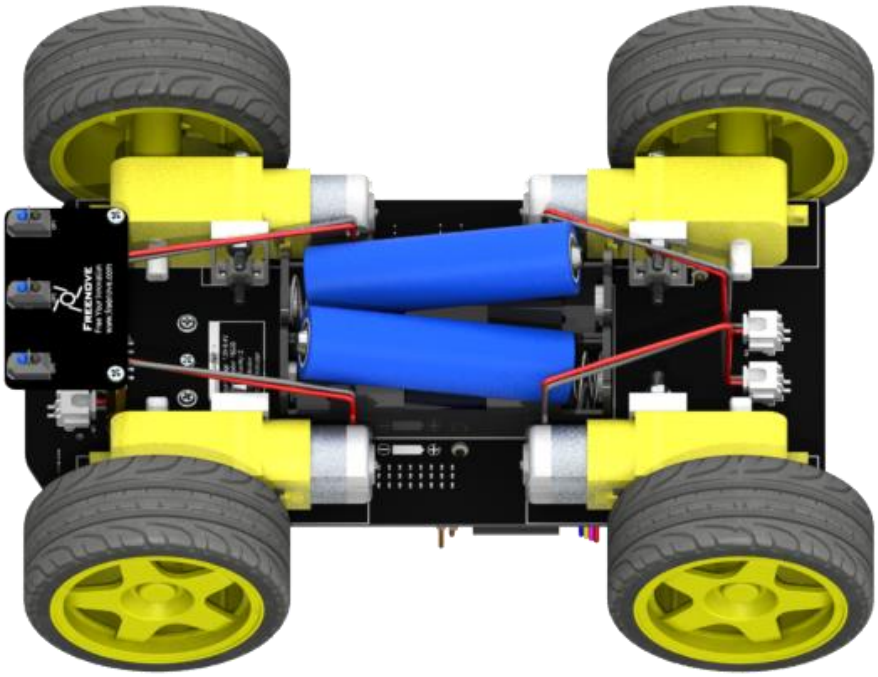
Pan Tilt

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:





Keep the Rocker arm consistent with screen printing on smart car board. Use two screw M2.5*8 to install it with smart car board.



Finally, install two 18650 batteries.

Chapter 2 Software installation and Test (necessary)

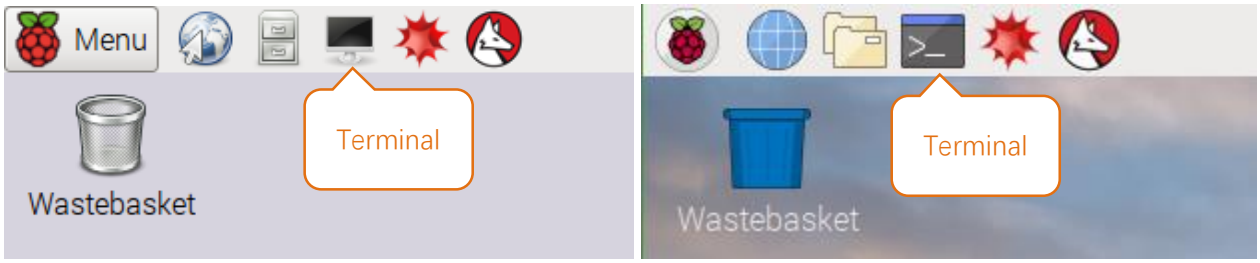
In this chapter, we will do some necessary preparation work: start your Pi Raspberry and install some necessary libraries. Then test some parts.

Note:

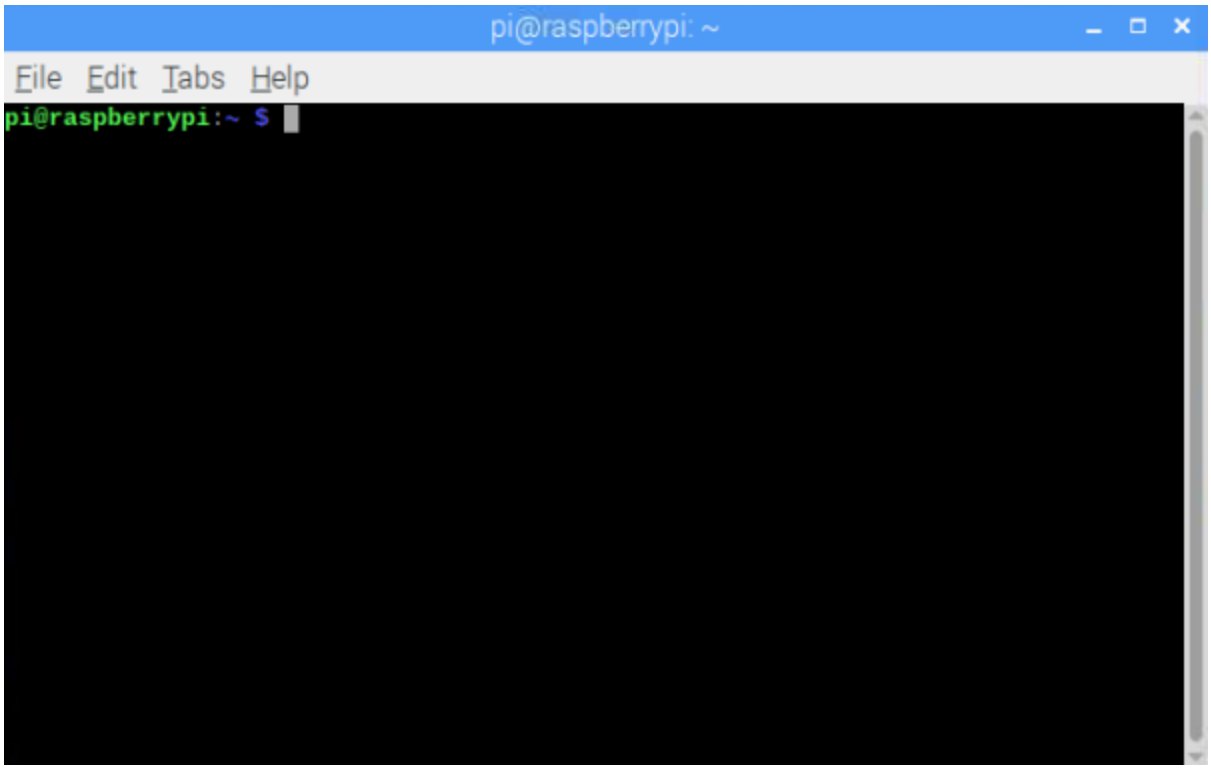
The installation of libraries needs much time. You can power Raspberry Pi with a power supply cable to save battery power of the car. Batteries are needed when driving peripherals such as motors, servos, LEDs, etc.

Step 1 Obtain the Code

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 "CTAL + ALT + T" on the desktop.



The terminal is shown below:



Type the following command in the terminal to obtain the code for the smart car. And place the code in the user directory "Pi". (Note: Here are two commands. Please excite commands in order.)

```
cd ~
git clone https://github.com/Freenove/Freenove_Four_wheeled_Smart_Car_Kit_for_Raspberry_Pi.git
```

Downloading need much 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 notice that All code for this smart car is written by using Python2. If you execute it under python 3, there will be error message.

Set Python2 as default python

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

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

If it is python2, you can skip this section.

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

1. Enter directory /usr/bin

```
cd /usr/bin
```

2. Delete the old python link.

```
sudo rm python
```

3. Create new python links to python.

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

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

```
python
```

```
pi@raspberrypi:~ $ cd /usr/bin
pi@raspberrypi:/usr/bin $ sudo rm python
pi@raspberrypi:/usr/bin $ sudo ln -s python2 python
pi@raspberrypi:/usr/bin $ python
Python 2.7.16 (default, Apr  6 2019, 01:42:57)
[GCC 8.2.0] on linux2
Type "help", "copyright", "credits" or "license" for more
>>>
```

If you want to set python3 as default python in **other projects**.

Just repeat command above and change python2 to python3.

```
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.
>>>
```

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 command, pressing "↑" will bring back the previous command, and pressing "↓" will bring back the latter 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 directly completed 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 completed automatically.

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

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

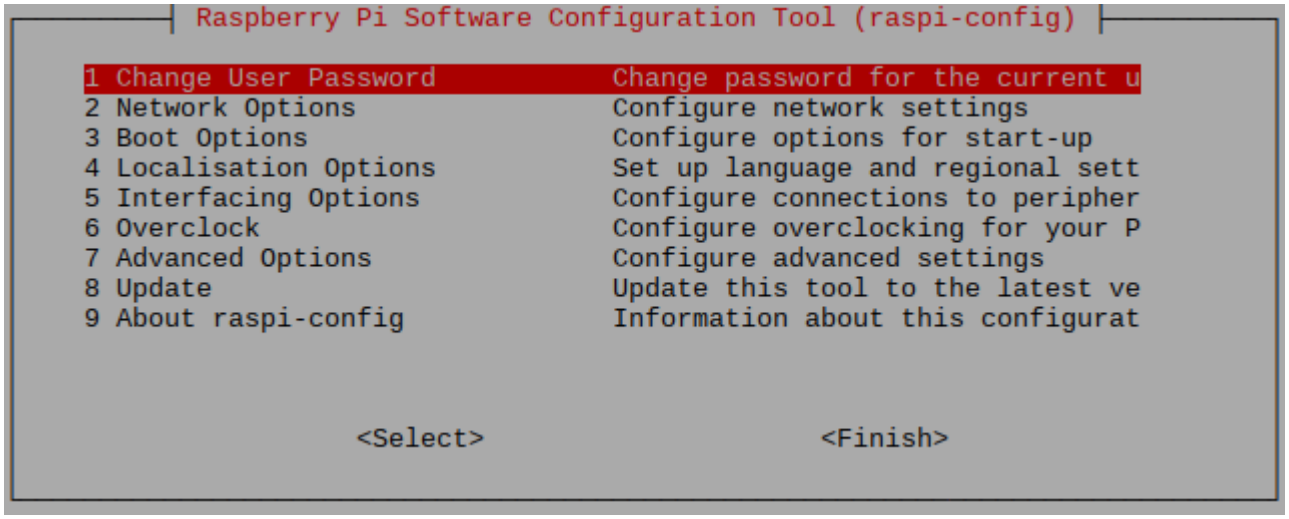
Step 2 Configure I2C

Enable I2C

The I2C interface raspberry pi is closed in default. You need to open it manually. You can enable the I2C interface in the following way. Open the terminal. Type command in the terminal:

```
sudo raspi-config
```

Then open the following dialog box:



Choose “5 Interfacing Options”→“P5 I2C”→“Yes”→“Finish” in order and restart your RPi later. Then the I2C module is started.

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

```
lsmod | grep i2c
```

If I2C module has been started, following content will be shown (the number of your device may be different):

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

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

Step 3 Install PyQt4

The project code is based on PyQt4. So operation of the program requires the support of PyQt4.

Open the terminal and execute the following command to install PyQt4. **(Note: Here are three commands. Please excite commands in order.)**

```
sudo apt-get update
sudo apt-get  install python-qt4
sudo apt-get  install python-dev
```

Installation need much time. Please wait with patience. Successful installation is shown below:

```
pi@raspberrypi:~ $ sudo apt-get  install python-dev
Reading package lists... Done
Building dependency tree
Reading state information... Done
python-dev is already the newest version (2.7.13-2).
python-dev set to manually installed.
0 upgraded, 0 newly installed, 0 to remove and 116 not upgraded.
```

Step 4 Install WS281X library

Install WS281X library in Raspberry Pi, which is used for WS281X RGB LED on the car.

1. Enter the following commands in the terminal: (Note that there are two commands to execute separately)

```
sudo apt-get update
sudo apt-get install build-essential python-dev python-pip unzip wget scons swig
```

2. Execute following command to install WS281X library. Following is only **one command.** Don't divide it.

```
cd ~/Freenove_Four_wheeled_Smart_Car_Kit_for_Raspberry_Pi/InstallationPackage && unzip rpi_ws281x.zip && cd rpi_ws281x && sudo scons && sudo pip install rpi_ws281x
```

```
pi@raspberrypi:~ $ cd ~/Freenove_Four_wheeled_Smart_Car_Kit_for_Raspberry_Pi/InstallationPackage &&
unzip rpi_ws281x.zip && cd rpi_ws281x && sudo scons && sudo pip install rpi_ws281x
```

Because WS281X LED needs high clock accuracy, audio core module should be disabled on the Raspberry Pi.

1.Edit the file:

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

Add a following line here:

```
blacklist snd_bcm2835
```

Then the file is saved by pressing “CTRL + O”, then click “ENTER”, then “CTRL + X” to closes the editor.

2. Edit the configuration file:

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

Below are lines with the following content (with Ctrl + W you can search):

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

This bottom line is commented out with a hashtag # at the beginning of the line:

```
#dtparam=audio=on
```

3.We restart the system

```
sudo reboot
```

Step 5 Install Opencv library

Execute following commands in the terminal to install Opencv library:

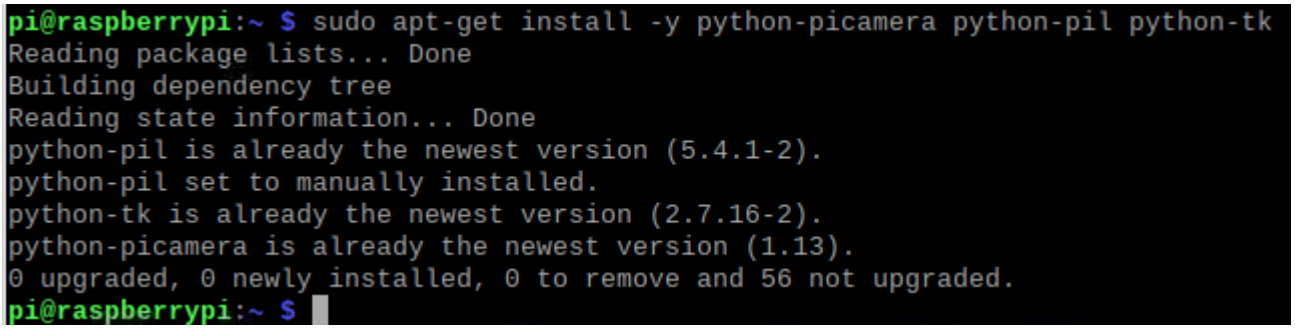
1.nstall opencv development environment:

```
sudo apt-get install -y libopencv-dev python-opencv libopencv-contrib-dev
```

2.Install some tools:

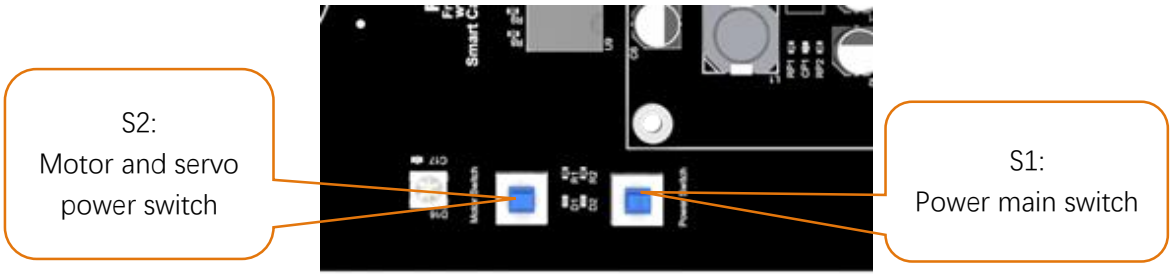
```
sudo apt-get install -y python-picamera python-pil python-tk
```

Successful installation is shown below:



Step 6 Module test(necessary)

In this section, the car must be equipped with **batteries**, and **S1** power switch and **S2** motor switch need be turned on. 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.

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

Motor

Run program

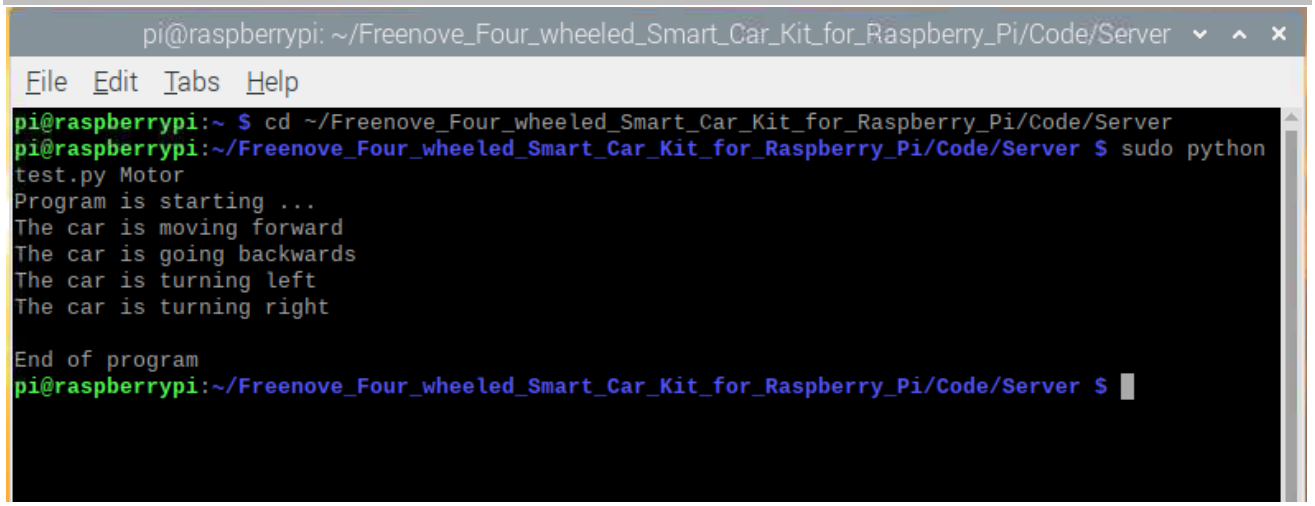
Open the terminal of Raspberry Pi. Enter the following commands to test the motor.

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

```
cd ~/Freenove_Four_wheeled_Smart_Car_Kit_for_Raspberry_Pi/Code/Server
```

2.Execute test.py command:

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



Result:

The car moves forward for 1 seconds, then moves back for 1 seconds, then turns left for 1 seconds, turns right for 1 seconds, then stops. You can press "Ctrl + C" to end the program ahead of time.

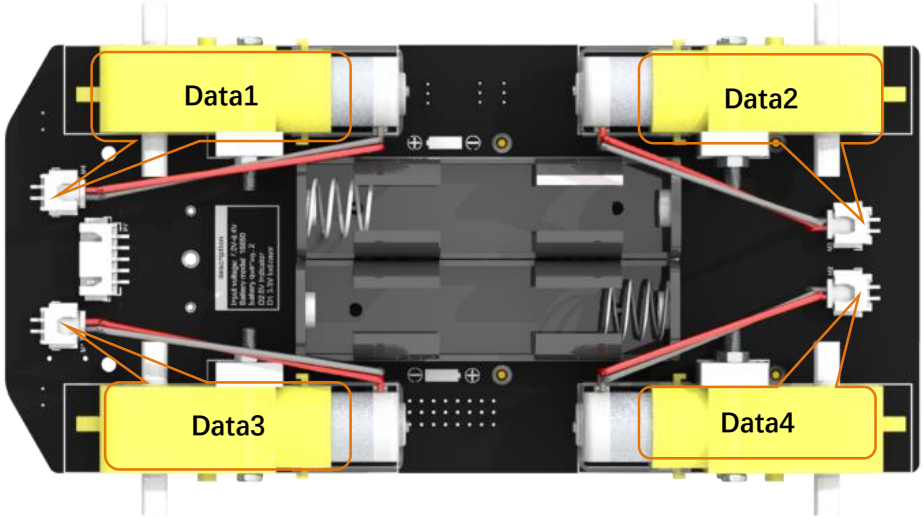
The code is as below:

```
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)         #Left
        print "The car is turning left"
        time.sleep(1)
        PWM.setMotorModel(2000,2000,-1500,-1500)         #Right
        print "The car is turning right"
        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 reverse. The larger the absolute value is, the larger the motor speed is. When the input is 0, the motor will stop. If the function has following input: setMotorModel(2000,2000, 2000, 2000), four motors will rotate forward and the car will move forward.



ADC Module

Run program

Enter 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.

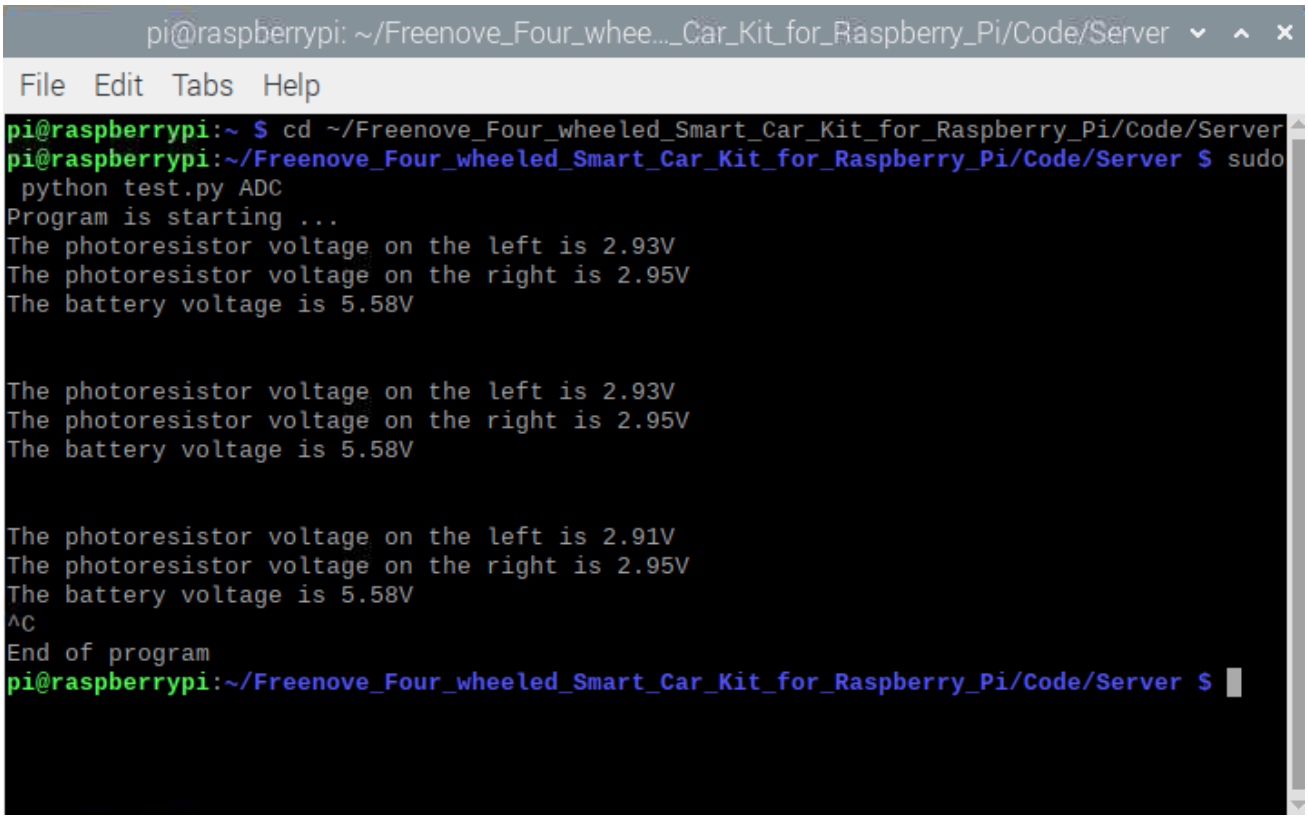
```
pi@raspberrypi:~/Freenove_Four_wheeled_Smart_Car_Kit_for_Raspberry_Pi/Code/Server $
```

1.If not, execute the cd command:

```
cd ~/Freenove_Four_wheeled_Smart_Car_Kit_for_Raspberry_Pi/Code/Server
```

2.Execute test.py command:

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



Result:

Every 1s, the voltage value of the two photoresistors and the battery voltage value 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:

```
1  from ADC import *
2  adc=Adc()
3  def test_Adc():
4      try:
5          while True:
6              Left_IDR=adc.recvADC(0)
7              print ("The photoresistor voltage on the left is "+str(Left_IDR)+"V")
```

```
8               Right_IDR=adc.recvADC(1)
9               print ("The photoresistor voltage on the right is "+str(Right_IDR)+"V")
10              Power=adc.recvADC(2)
11              print ("The battery voltage is "+str(Power*3)+"V")
12              time.sleep(1)
13              print '\n'
14      except KeyboardInterrupt:
15          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 the **battery after divided.** After multiplying by 3, it is the actual battery voltage value

Infrared Line tracking module

Run program

Enter 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.

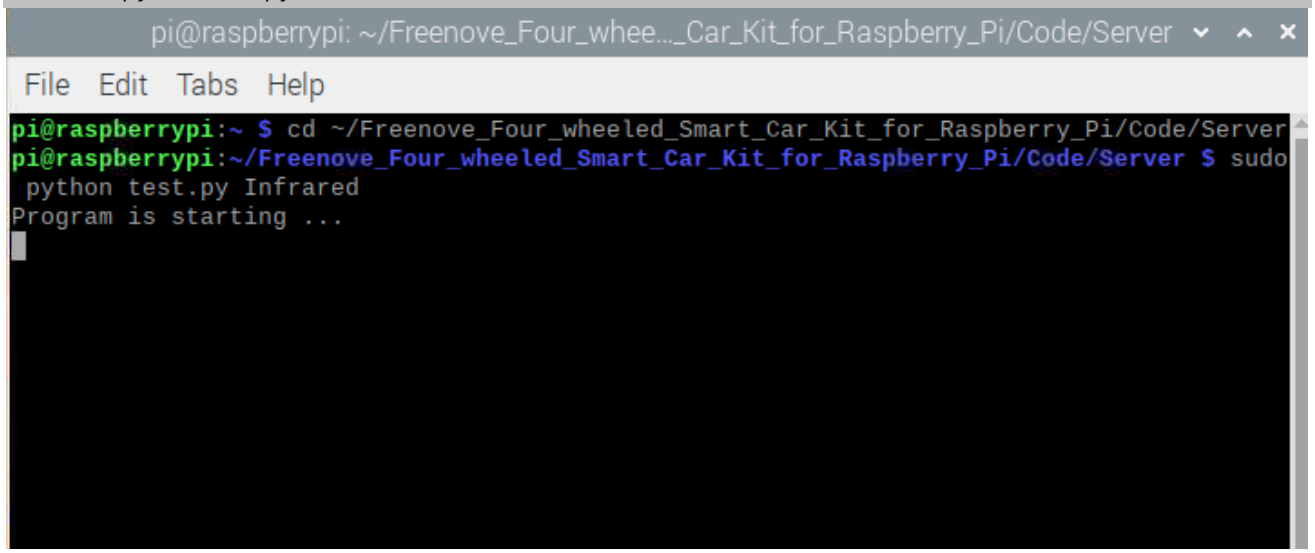
```
pi@raspberrypi:~/Freenove_Four_wheeled_Smart_Car_Kit_for_Raspberry_Pi/Code/Server $
```

1. If not, execute the cd command:

```
cd ~/Freenove_Four_wheeled_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.
The code is as below:

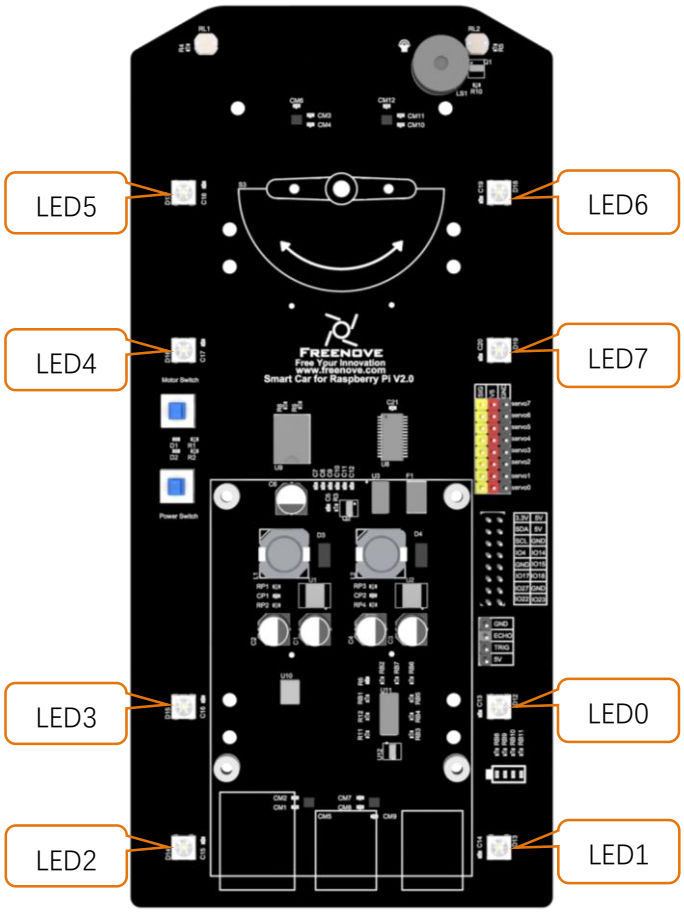
```
from Infrared_Obstacle_Avoidance import *
def test_Infrared():
    try:
        while True:
            if GPIO.input(IR01)!=True and GPIO.input(IR02)==True and GPIO.input(IR03)!=True:
                print 'Middle'
            elif GPIO.input(IR01)!=True and GPIO.input(IR02)!=True and GPIO.input(IR03)==True:
                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 is below. You can control them separately.



Run program

Enter the following commands to test LED.

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

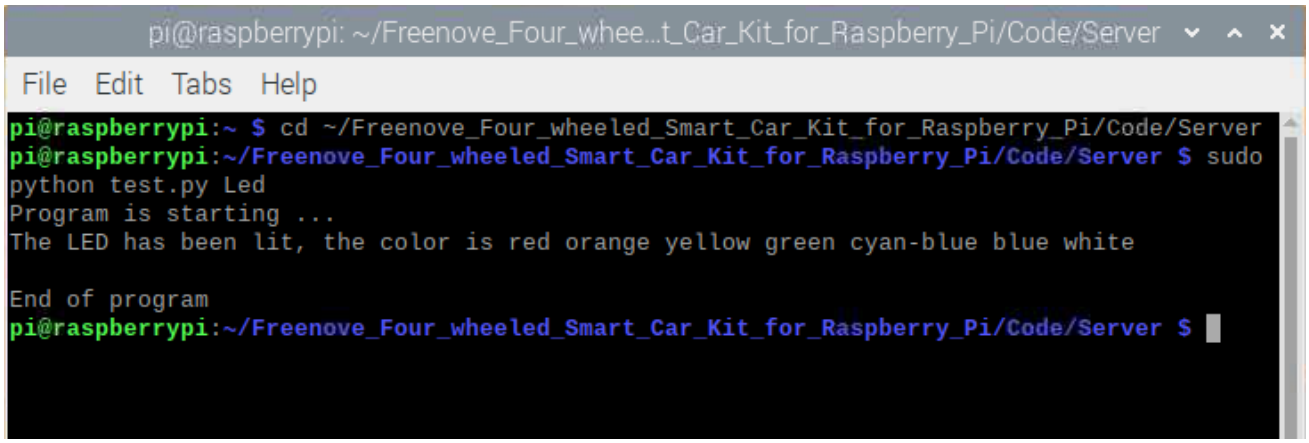
```
pi@raspberrypi:~/Freenove_Four_wheeled_Smart_Car_Kit_for_Raspberry_Pi/Code/Server $
```

1.If not, execute the cd command:

```
cd ~/Freenove_Four_wheeled_Smart_Car_Kit_for_Raspberry_Pi/Code/Server
```

2.Execute test.py command:

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



Result:

All the 8 LEDs light up for 3s, and the colors are: red, orange, yellow, green, blue, cyan-blue, purple and white. You can press "Ctrl + C" to end the program ahead of time.

The code is as below:

```
1  import time
2  try:
3      from Led import *
4      led=Led()
5      def test_Led():
6          try:
7              led.ledIndex(0,255,0,0)       #Red
8              led.ledIndex(1,255,125,0)     #orange
9              led.ledIndex(2,255,255,0)     #yellow
10             led.ledIndex(3,0,255,0)       #green
11             led.ledIndex(4,0,255,255)     #cyan-blue
12             led.ledIndex(5,0,0,255)       #blue
13             led.ledIndex(6,128,0,128)     #purple
14             led.ledIndex(7,255,255,255)   #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)
The function has four input parameters, the first one is number of the LED needed to be controlled, and the other three are the colors of the R, G, and B channels, For example: ledIndex(0,255,0,0), light LED0, color It is red.

colorWipe(strip, color, wait_ms)
This function erases the color of one pixel at a time, 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 color is red first, wait for 20ms, and then the LED1 color is also red, so that all eight LEDs are lit and red.

Buzzer

Run the program

Enter 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.

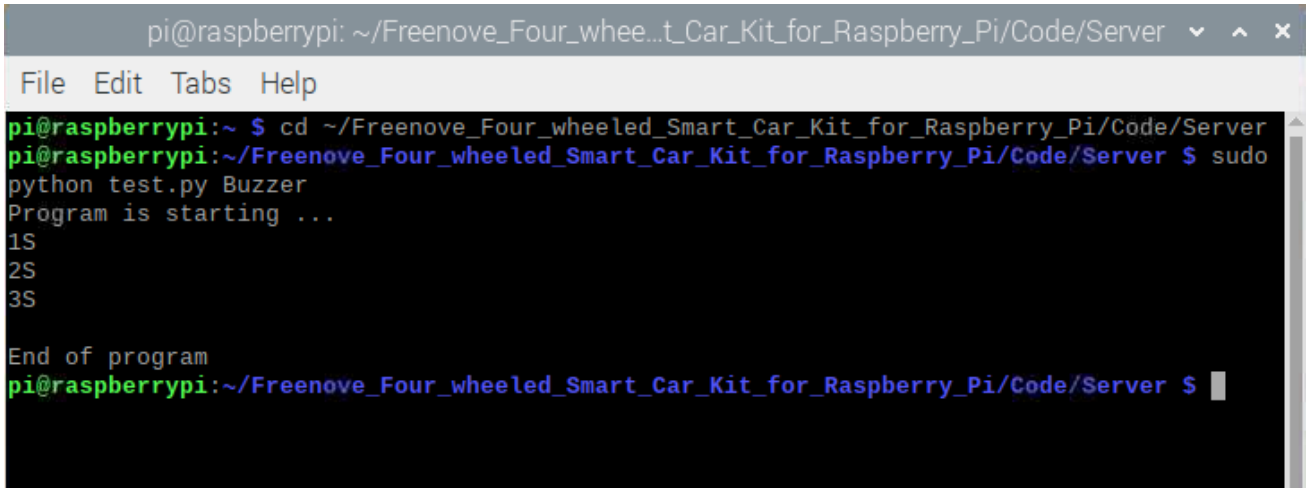
```
pi@raspberrypi:~/Freenove_Four_wheeled_Smart_Car_Kit_for_Raspberry_Pi/Code/Server $
```

1.If not, execute the cd command:

```
cd ~/Freenove_Four_wheeled_Smart_Car_Kit_for_Raspberry_Pi/Code/Server
```

2.Execute test.py command:

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



Result:

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

The code is as below:

```
from Buzzer import *
buzzer=Buzzer()
def test_Buzzer():
    try:
        buzzer.run(cmd.CMD_START)
        time.sleep(1)
        print "1S"
        time.sleep(1)
        print "2S"
        time.sleep(1)
        print "3S"
        buzzer.run(cmd.CMD_STOP)
        print "\nEnd of program"
    except KeyboardInterrupt:
        buzzer.run(cmd.CMD_STOP)
        print "\nEnd of program"
```

Reference

buzzer.run(cmd)

This function has one input parameter. If the input is cmd.CMD_START, the buzzer will be turned on. If the input is cmd.CMD_STOP, the buzzer will be turned off. cmd.CMD_START and cmd.CMD_STOP are custom command prompts, and there are many similar command prompts. You can open Command.py to view.

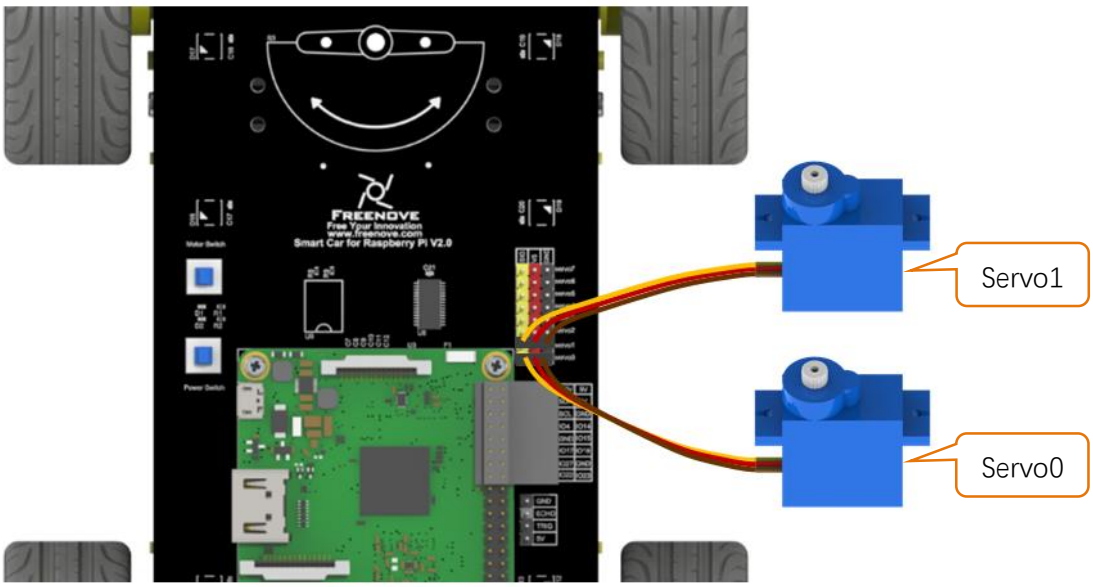
Servo

Run program

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

Next let's 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 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_Four_wheeled_Smart_Car_Kit_for_Raspberry_Pi/Code/Server $
```

1.If not, execute the cd command:

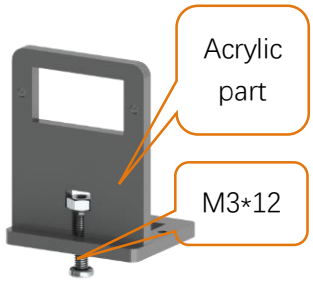
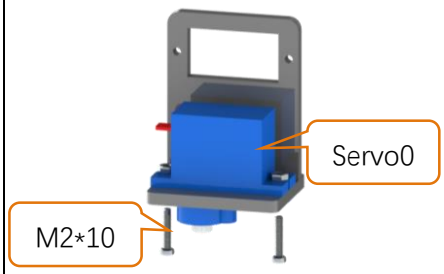
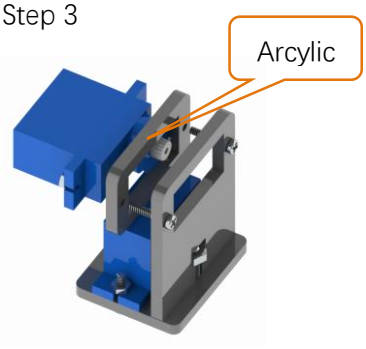
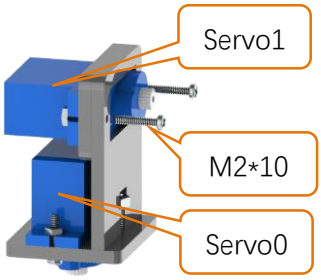
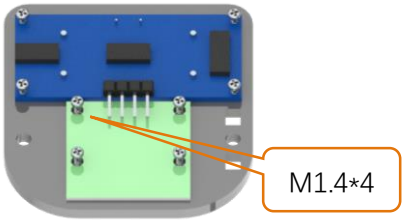
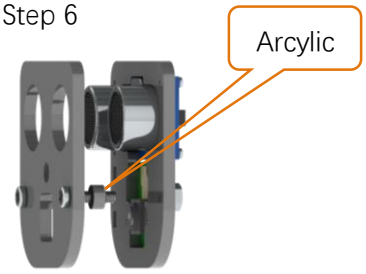
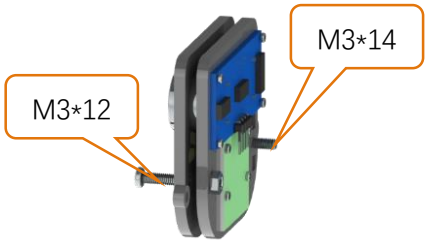
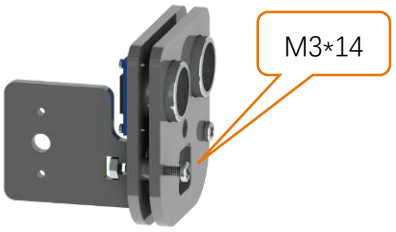
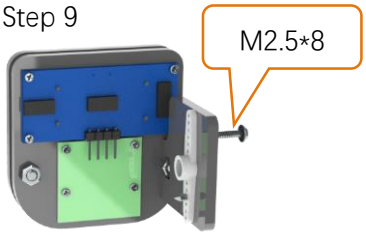
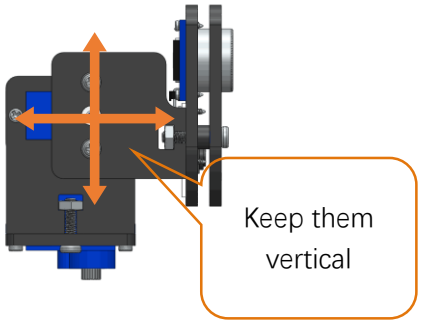
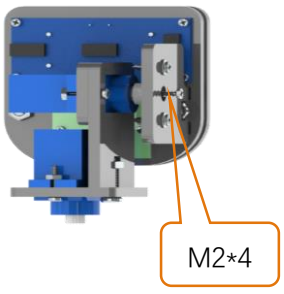
```
cd ~/Freenove_Four_wheeled_Smart_Car_Kit_for_Raspberry_Pi/Code/Server
```

2.Execute Servo.py command:

```
sudo python servo.py
```

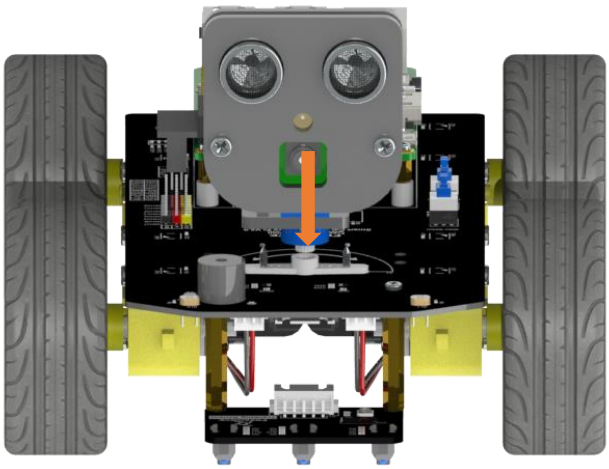
Then servos rotate to the proper angle. Please keep the connection between the servos and the smart car board.

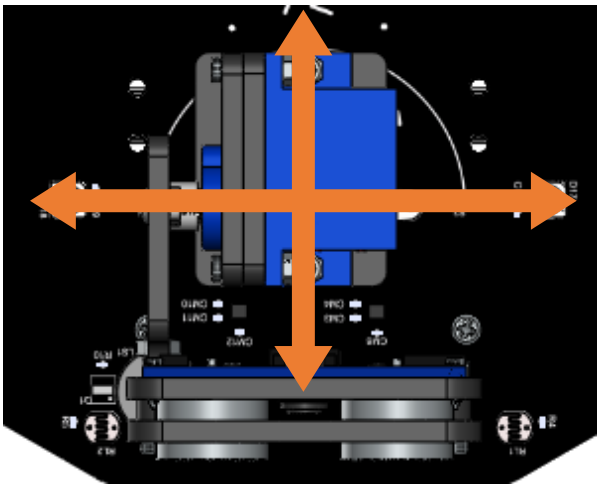
Installation steps: (**Note: Do not disorder Servo0 and Servo1 during the installation.**)

<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>Step 10
 
</td>
<td>Step 11
 
</td>
<td>After finished
  </td>
</tr>
</table>

Install Pan Tilt on smart car board.

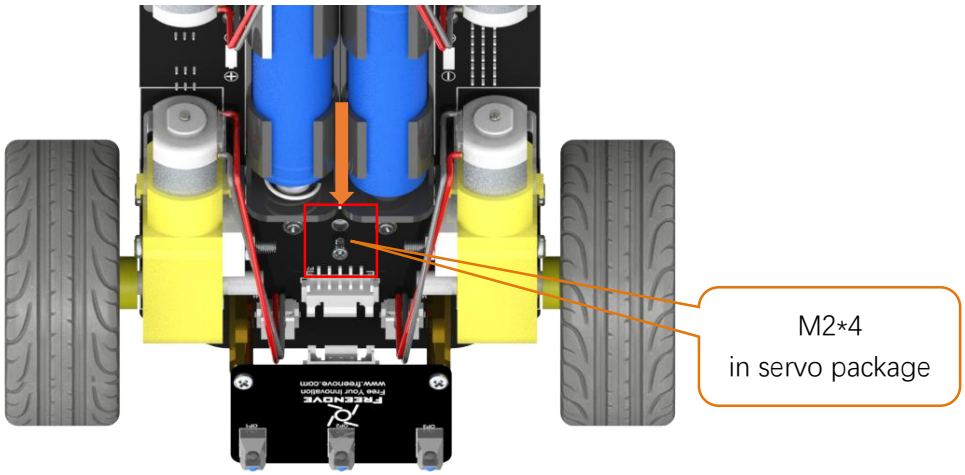
Step 1





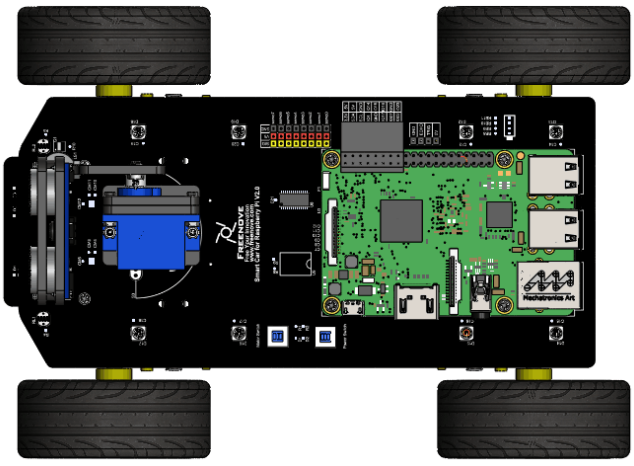
Keep the pan tilt as right picture and install servo0 with rocker arm.

Step 2



Then use slotted screwdriver to support the M2*4 screw to fix servo 0.

Step 3



Now Pan Tilt is install successfully. (note: wiring about the ultrasonic module and camera module of pan tilt will be introduced later.)

Enter 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.

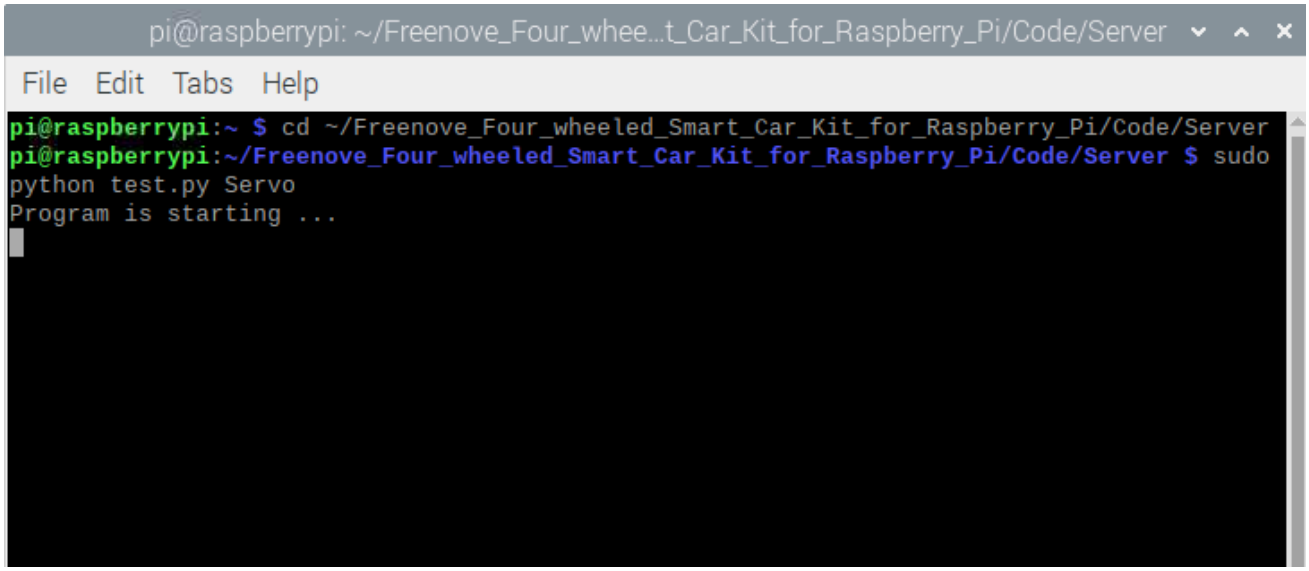
```
pi@raspberrypi:~/Freenove_Four_wheeled_Smart_Car_Kit_for_Raspberry_Pi/Server $
```

1. If not, execute the cd command:

```
cd ~/Freenove_Four_wheeled_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:

```
from servo import *
pwm=Servo()
def test_Servo():
    try:
        while True:
            for i in range(50,110,1):
                pwm.setServoPwm(1,0,i,0)
                time.sleep(0.01)
            for i in range(110,50,-1):
                pwm.setServoPwm(1,0,i,0)
                time.sleep(0.01)
            for i in range(20,80,1):
                pwm.setServoPwm(0,1,0,i)
                time.sleep(0.01)
            for i in range(80,20,-1):
                pwm.setServoPwm(0,1,0,i)
                time.sleep(0.01)
    except KeyboardInterrupt:
        pwm.setServoPwm(1,1,90,20)
        print "\nEnd of program"
```

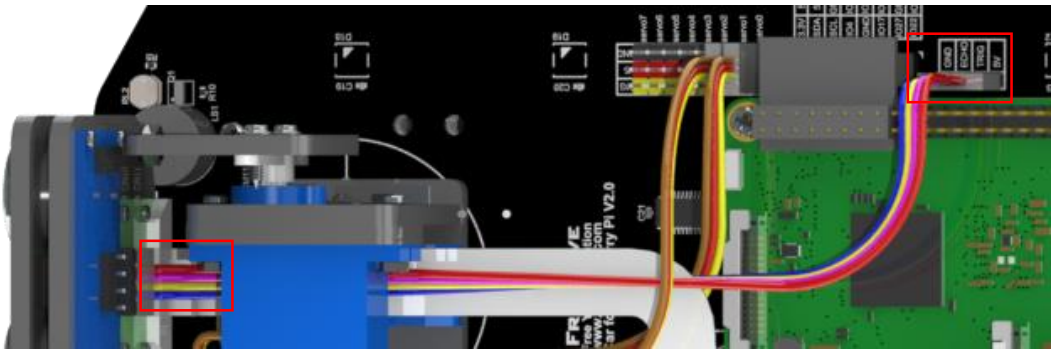
Reference

setServoPwm(Servo1,Servo2,angle1,angle2)

The function has four input parameters. The first two input is related to two servos. If the input 1, it means the servo is selected. If the input is 0, it means the servo is not selected. The latter two input represent the rotation angle of servo 0 and servo 1, example: **setServoPwm (1,0,90,30)** means: only the servo0 is selected, the servo0 will rotate to 90 degrees, the servo 1 is not selected, and it will not rotate to 30 degrees. The maximum rotation angle of the servo is only 180 degrees.

Ultrasonic module

Next, use jumper 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 be freely rotated, and after the wiring is completed, connect servo cable again. When wiring, you should keep the silk screen of the ultrasonic module and the smart car board consistent. Vcc should be is connected to 5V, Trig is connected to TRIG, Echo is connected to ECHO, and Gnd is connected to GND.

If it 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 test.

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_Four_wheeled_Smart_Car_Kit_for_Raspberry_Pi/Code/Server $
```

1. If not, execute the cd command:

```
cd ~/Freenove_Four_wheeled_Smart_Car_Kit_for_Raspberry_Pi/Code/Server
```

2. Execute test.py command:

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

```
pi@raspberrypi: ~/Freenove_Four_whee...t_Car_Kit_for_Raspberry_Pi/Code/Server
File  Edit  Tabs  Help
pi@raspberrypi:~ $ cd ~/Freenove_Four_wheeled_Smart_Car_Kit_for_Raspberry_Pi/Code/Server
pi@raspberrypi:~/Freenove_Four_wheeled_Smart_Car_Kit_for_Raspberry_Pi/Code/Server $ sudo
python test.py Ultrasonic
Program is starting ...
Obstacle distance is 30CM
Obstacle distance is 667CM
Obstacle distance is 8CM
Obstacle distance is 8CM
Obstacle distance is 11CM
Obstacle distance is 22CM
Obstacle distance is 10CM
Obstacle distance is 0CM
Obstacle distance is 13CM
Obstacle distance is 34CM
Obstacle distance is 664CM
Obstacle distance is 10CM
Obstacle distance is 9CM
Obstacle distance is 16CM
^C
End of program
pi@raspberrypi:~/Freenove_Four_wheeled_Smart_Car_Kit_for_Raspberry_Pi/Code/Server $
```

Result:

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

The code is as below:

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

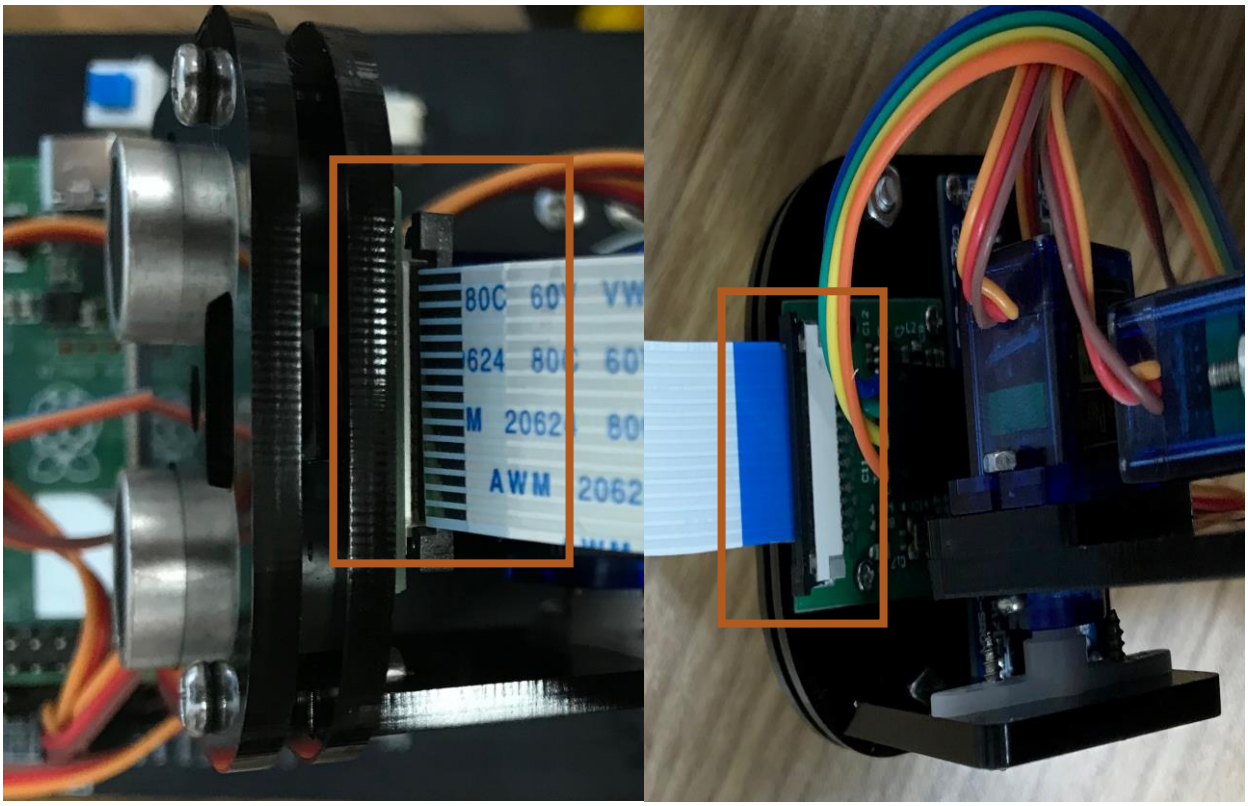
Reference

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

Camera

Next let's 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 Raspberry Pi is shut down, or the camera may be burned.)**

Step 1



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

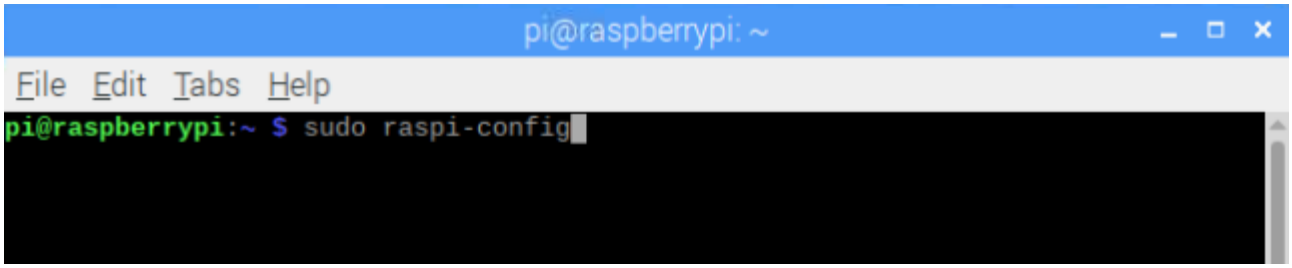
Step 2

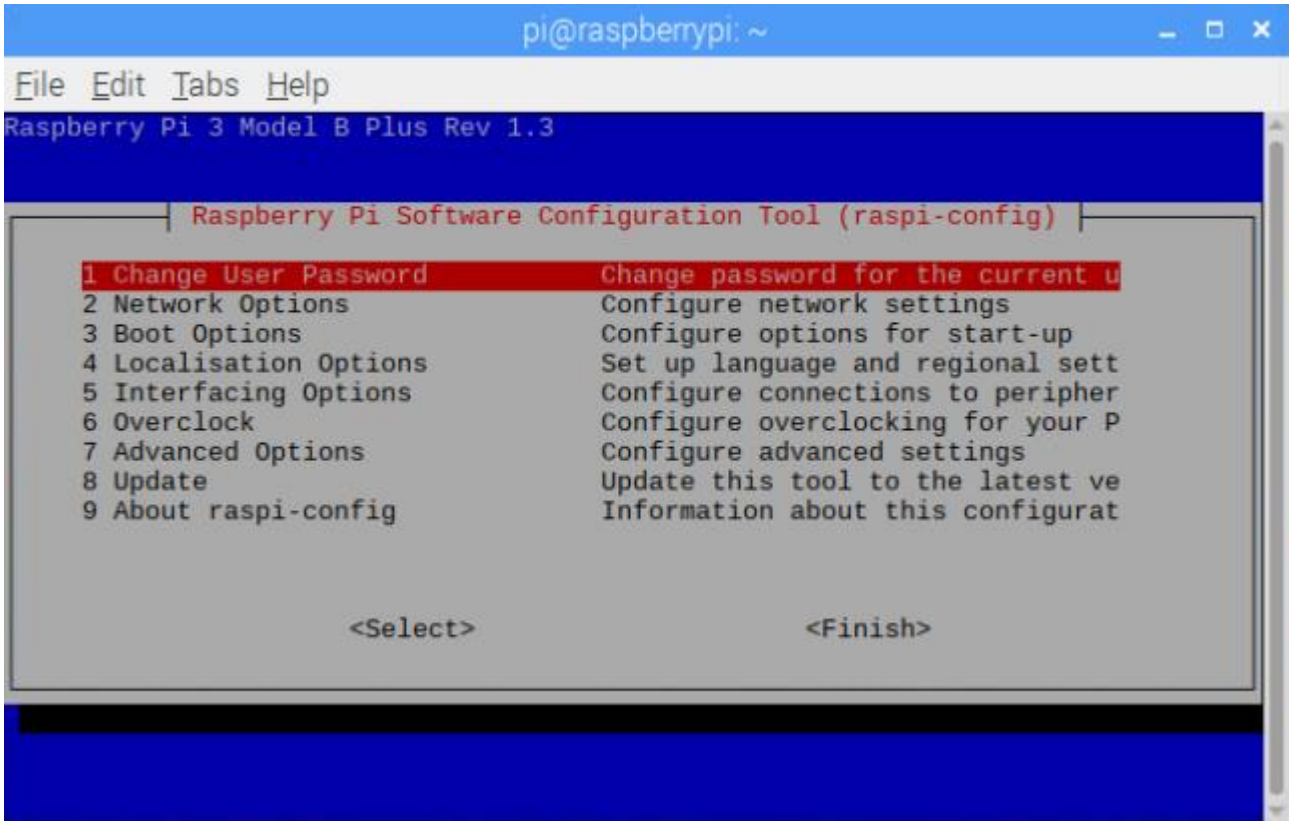
Connect another end of cable to raspberry pi. Please note front and back of the cable.

Enable camera

Turn on S1 or use cable to power Raspberry Pi, and start it. Then enter following commands in the terminal to enable camera:

```
sudo raspi-config
```

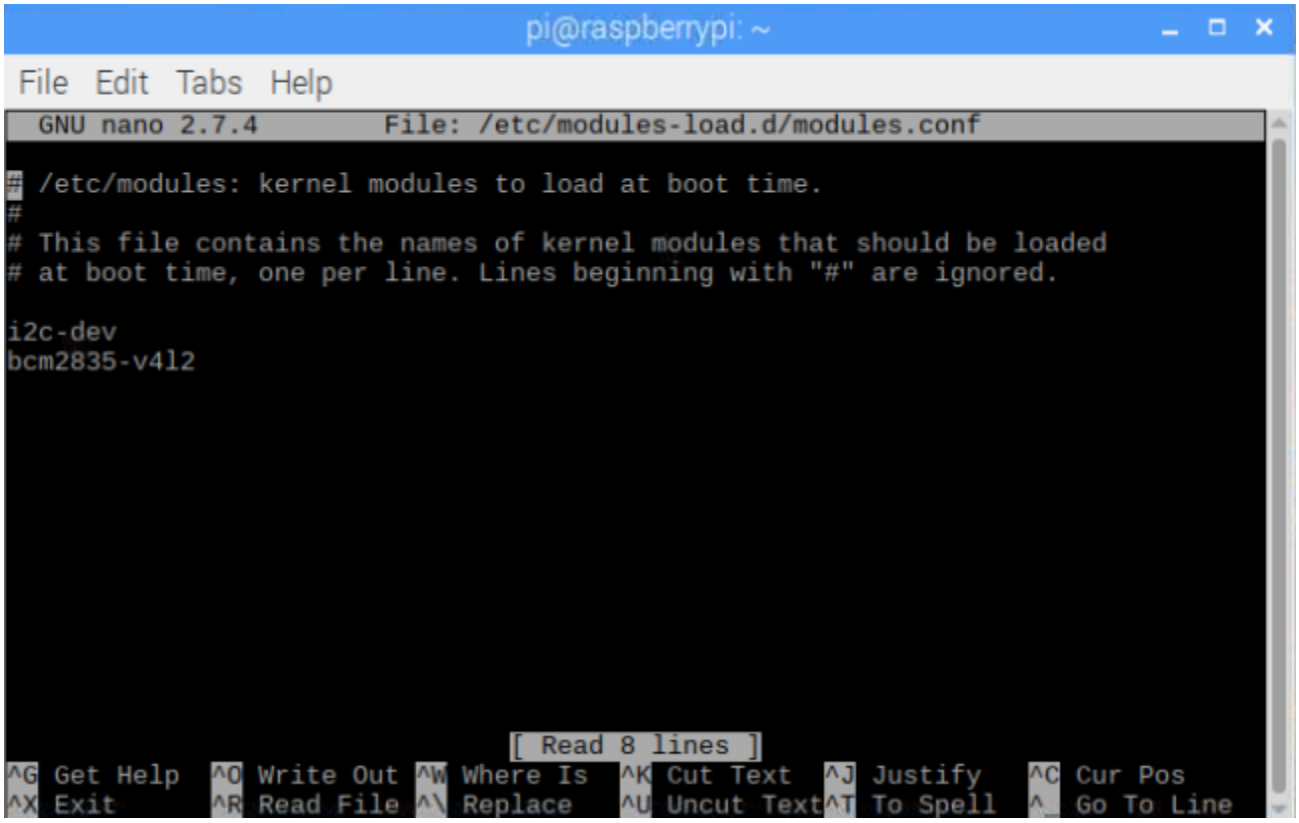




Select 5 Interfacing Options --> P1 Camera --> Yes-->Finish，reboot Raspberry Pi.

3, Then enter following command to modify the config file to make raspberry Pi generate /dev/video0.

```
sudo nano /etc/modules-load.d/modules.conf
```



Add following line:

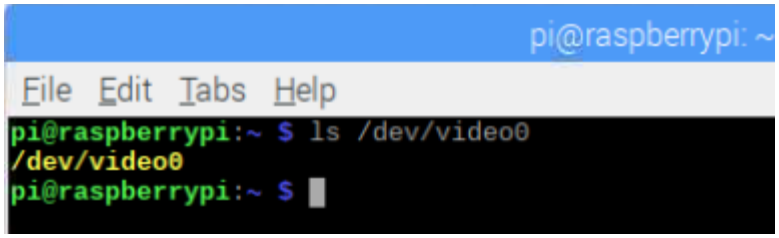
```
bcm2835-v4l2
```

Press “CTRL + O”, then ENTER, then “CTRL + X” to save and close.

Then reboot Raspberry Pi and enter following commands:

```
ls /dev/video0
```

Then the device node will be shown below:



Run program

Enter following commands in the terminal to test camera.

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

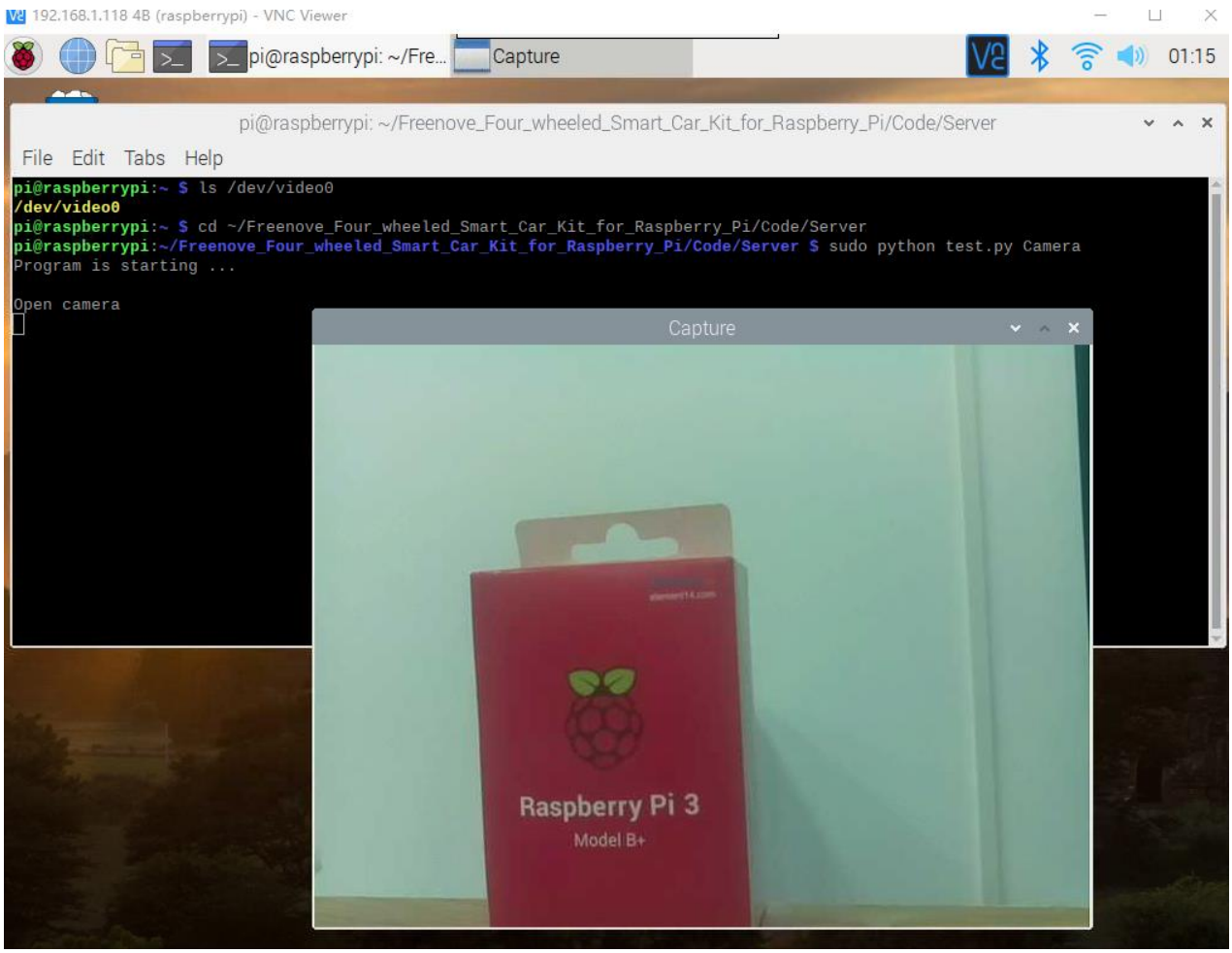
```
pi@raspberrypi:~/Freenove_Four_wheeled_Smart_Car_Kit_for_Raspberry_Pi/Code/Server $
```

1. If not, execute the cd command:

```
cd ~/Freenove_Four_wheeled_Smart_Car_Kit_for_Raspberry_Pi/Code/Server
```

2. Execute test.py command:

```
sudo python test.py Camera
```



Result:

A window will pop up and display the picture shot by the camera. Press "Ctrl + "C" in the **terminal** window to end the program. (Note: pressing "Ctrl + C" in the **camera** window doesn't work.)

The coed is below:

```
1  import cv2
2  def test_Camera():
3          try:
4              print "\nOpen camera"
5              capturing_Flag = True
6              cap = cv2.VideoCapture(0)
7              while(capturing_Flag):
8                  ret, frame = cap.read()
9                  cv2.imshow("Capture", frame)
10                 cv2.waitKey(5)
11             cv2.destroyAllWindows()
12         except KeyboardInterrupt:
13                 print "\nClose camera"
14                 capturing_Flag = False
```

Charpter 3 LED Show

Description

In the test of Chapter 2, we have controlled 8 LEDs to display different colors in turn. On this basis, we add some algorithm 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 your LED style you want.

Run Program

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

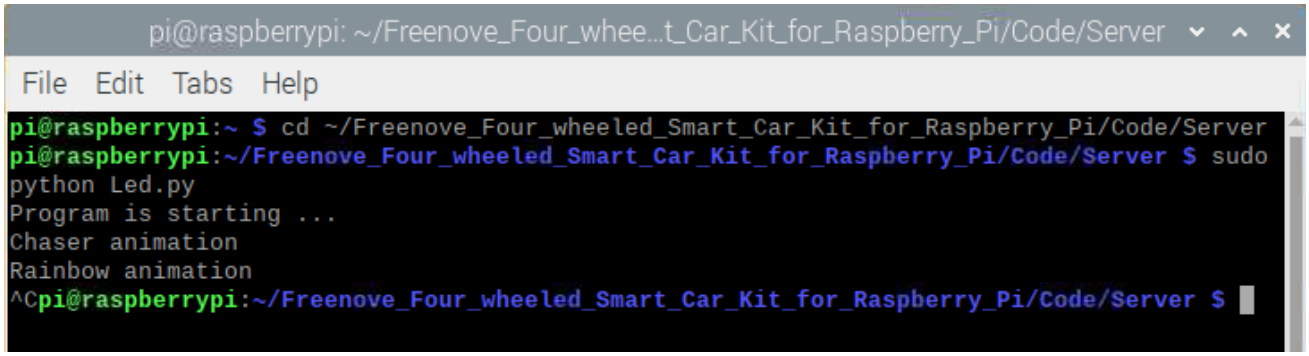
```
pi@raspberrypi:~/Freenove_Four_wheeled_Smart_Car_Kit_for_Raspberry_Pi/Code/Server $
```

1.If not, execute the cd command:

```
cd ~/Freenove_Four_wheeled_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:
```

```
    def __init__(self):
        # Create NeoPixel object with appropriate configuration.
        self.strip = Adafruit_NeoPixel(LED_COUNT, LED_PIN, LED_FREQ_HZ, LED_DMA, LED_INVERT,
LED_BRIGHTNESS, LED_CHANNEL)
        # Intialize the library (must be called once before other functions).
        self.strip.begin()
        self.number='0'
    def colorWipe(self,strip, color, wait_ms=50):
        """Wipe color across display a pixel at a time."""
        for i in range(self.strip.numPixels()):
            self.strip.setPixelColor(i, color)
            self.strip.show()
            time.sleep(wait_ms/1000.0)

    def theaterChase(self,strip, color, wait_ms=50, iterations=10):
        """Movie theater light style chaser animation."""
        for j in range(iterations):
            for q in range(3):
                for i in range(0,self.strip.numPixels(), 3):
                    self.strip.setPixelColor(i+q, color)
                self.strip.show()
                time.sleep(wait_ms/1000.0)
                for i in range(0, self.strip.numPixels(), 3):
                    self.strip.setPixelColor(i+q, 0)

    def wheel(self,pos):
        """Generate rainbow colors across 0-255 positions."""
        if pos < 85:
            return Color(pos * 3, 255 - pos * 3, 0)
        elif pos < 170:
            pos -= 85
            return Color(255 - pos * 3, 0, pos * 3)
        else:
            pos -= 170
            return Color(0, pos * 3, 255 - pos * 3)

    def rainbow(self,strip, wait_ms=20, iterations=1):
        """Draw rainbow that fades across all pixels at once."""
        for j in range(256*iterations):
            for i in range(self.strip.numPixels()):
                 self.strip.setPixelColor(i, self.wheel((i+j) & 255))
            self.strip.show()
            time.sleep(wait_ms/1000.0)

```

```
    def rainbowCycle(self,strip, wait_ms=20, iterations=5):
        """Draw rainbow that uniformly distributes itself across all pixels."""
        for j in range(256*iterations):
            for i in range(self.strip.numPixels()):
                self.strip.setPixelColor(i, self.wheel((int(i * 256 / self.strip.numPixels())
+ j) & 255))
            self.strip.show()
            time.sleep(wait_ms/1000.0)

    def theaterChaseRainbow(self,strip, wait_ms=50):
        """Rainbow movie theater light style chaser animation."""
        for j in range(256):
            for q in range(3):
                for i in range(0, self.strip.numPixels(), 3):
                    self.strip.setPixelColor(i+q, self.wheel((i+j) % 255))
                self.strip.show()
                time.sleep(wait_ms/1000.0)
                for i in range(0, strip.numPixels(), 3):
                    strip.setPixelColor(i+q, 0)
led=Led()
# Main program logic follows:
if __name__ == '__main__':
    print ('Program is starting ... ')
    try:
        while True:
            print "Chaser animation"
            led.colorWipe(led.strip, Color(255, 0, 0))  # Red wipe
            led.colorWipe(led.strip, Color(0, 255, 0))  # Green wipe
            led.colorWipe(led.strip, Color(0, 0, 255))  # Blue wipe'''
            led.theaterChaseRainbow(led.strip)
            print "Rainbow animation"
            led.rainbow(led.strip)
            led.rainbowCycle(led.strip)
            led.colorWipe(led.strip, Color(0,0,0),10)
    except KeyboardInterrupt:  # When 'Ctrl+C' is pressed, the child program destroy() will be
executed.
        led.colorWipe(led.strip, Color(0,0,0),10)
```

Reference

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

This function is a function of WS2812 library. It is same as the previously customized ledIndex() function. It is used to lights up one LED and 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 show red color light.

strip.show()
This function is of WS2812 library. When the LED color is set, this function needs to be executed, then the LED will show the corresponding color. After the color is set; LED will still has not change if this function is not executed.

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

rainbow(strip, wait_ms,)
This function achieves the effect of rainbow **breathing**. It makes 8 Leds display **same** color at the same time, and then change all 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 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 another round of green, the last round of blue Color, like chasing. And then let the LEDs blink with an interval of 50ms, with different colors, render a tense atmosphere, thus complete the chase animation.

Rainbow animation: The effect of the rainbow is different from the effect 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.

Chapter 4 Light tracing Car

Description

The light-seeking function of the car is mainly use a photoresistor. The car has two photoresistors located at both ends of the front to detect light.

The 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.

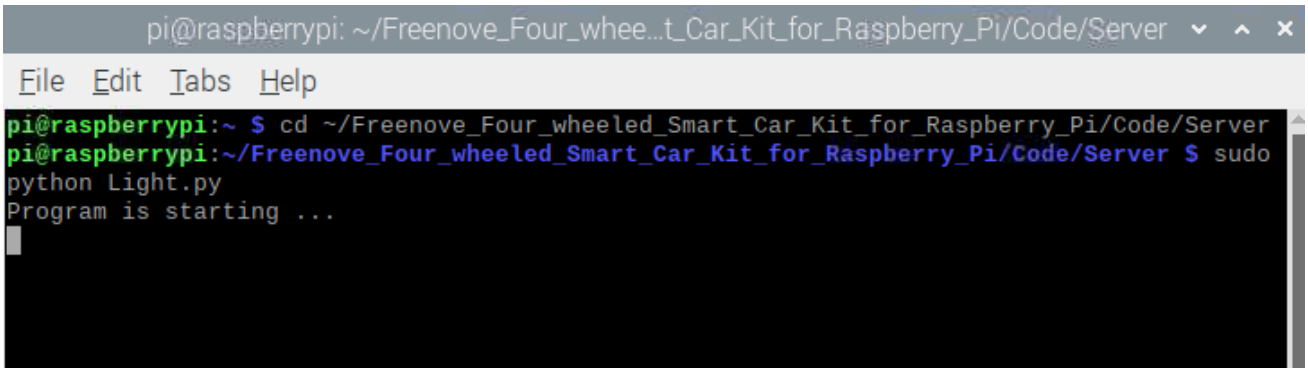
```
pi@raspberrypi:~/Freenove_Four_wheeled_Smart_Car_Kit_for_Raspberry_Pi/Code/Server $
```

1. If not, execute the cd command:

```
cd ~/Freenove_Four_wheeled_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:

```
1  import time
2  from Motor import *
3  from ADC import *
4  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 left and right photoresistor are less than 2.99, the car move forward straight. and
When one of the voltages is greater 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

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. The ultrasonic module measures the obstacle distance on the left, middle and right directions. Then control 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.

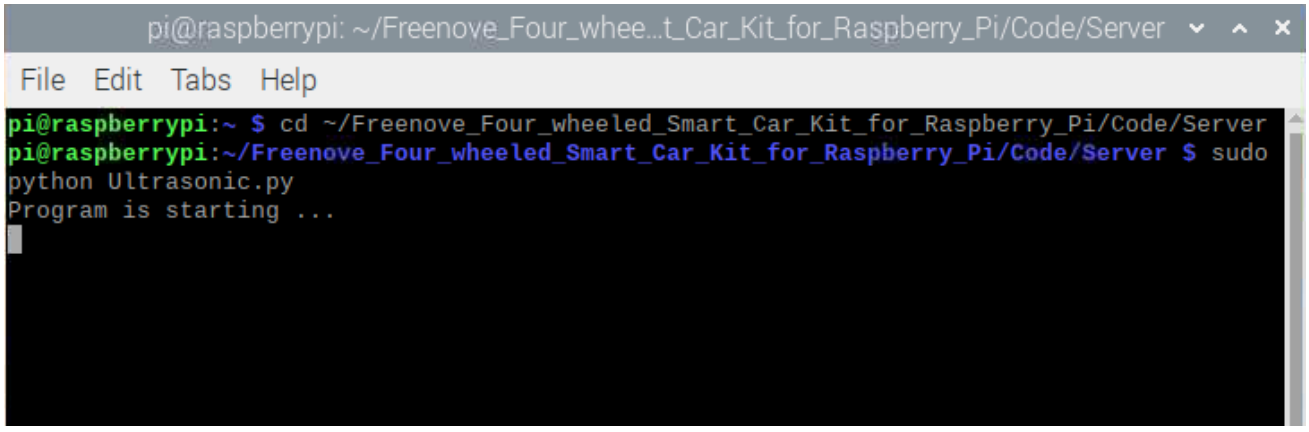
```
pi@raspberrypi:~/Freenove_Four_wheeled_Smart_Car_Kit_for_Raspberry_Pi/Code/Server $
```

1. If not, execute the cd command:

```
cd ~/Freenove_Four_wheeled_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
5         self.pwm_S.setServoPwm(1,0,30,0)
6         R = self.get_distance()
7         time.sleep(0.2)
8
9         self.pwm_S.setServoPwm(1,0,90,0)
10        L = self.get_distance()
11        time.sleep(0.2)
```

```
12
13          self.pwm_S.setServoPwm(1,0,150,0)
14          M = self.get_distance()
15          time.sleep(0.2)
16
17          while True:
18              self.pwm_S.setServoPwm(1,0,30,0)
19              R = self.get_distance()
20              self.run_motor(L,M,R)
21              time.sleep(0.2)
22
23              self.pwm_S.setServoPwm(1,0,90,0)
24              L = self.get_distance()
25              self.run_motor(L,M,R)
26              time.sleep(0.2)
27
28              self.pwm_S.setServoPwm(1,0,150,0)
29              M = self.get_distance()
30              self.run_motor(L,M,R)
31              time.sleep(0.2)
32              self.run_motor(L,M,R)
33
34  ultrasonic=Ultrasonic()
35  # Main program logic follows:
36  if __name__ == '__main__':
37      print ('Program is starting ... ')
38      try:
39          ultrasonic.run()
40      except KeyboardInterrupt:  # When 'Ctrl+C' is pressed, the child program destroy() will be
41  executed.
42          PWM.setMotorModel(0,0,0,0)
43          ultrasonic.pwm_S.setServoPwm(1,0,90,20)
```

Result analysis

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

When the distance between the left>30cm, middle >30cm, right>30cm. It means that there is no obstacle within 30cm. Then make the car move forward.

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

When the distance between the left<30cm, middle <30cm, right>30cm. It means that there is an obstacle on the left side of the car, then make the car turn right.

When the distance between the left>30cm, middle <30cm, right<30cm. It means that there is an obstacle on the right side of the car, then make the car turn left.

Chapter 6 Infrared line tracking Car

Description

The line tracing function of the car mainly uses the infrared module. When the sensor detects black line the corresponding led will light up. Control the car move according to the value of three sensors.

Run program

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

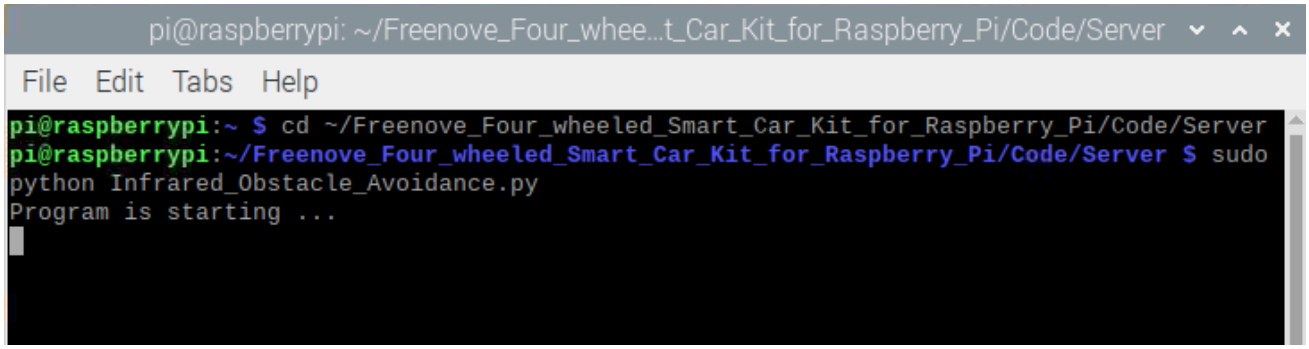
```
pi@raspberrypi:~/Freenove_Four_wheeled_Smart_Car_Kit_for_Raspberry_Pi/Code/Server $
```

1. If not, execute the cd command:

```
cd ~/Freenove_Four_wheeled_Smart_Car_Kit_for_Raspberry_Pi/Code/Server
```

2. Run Infrared_Obstacle_Avoidance.py:

```
sudo python Infrared_Obstacle_Avoidance.py
```



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

```
1  import time
2  from Motor import *
3  import RPi.GPIO as GPIO
4  IR01 = 14
5  IR02 = 15
6  IR03 = 23
7  GPIO.setmode(GPIO.BCM)
8  GPIO.setup(IR01,GPIO.IN)
9  GPIO.setup(IR02,GPIO.IN)
10 GPIO.setup(IR03,GPIO.IN)
11 class Infrared_Obstacle_Avoidance:
12     def run(self):
13
14         while True:
15             self.LMR=0x00
16             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(-1400,-1400,1400,1400)
            elif self.LMR==6:
                PWM.setMotorModel(-800,-800,1200,1200)
            elif self.LMR==1:
                PWM.setMotorModel(1400,1400,-1400,-1400)
            elif self.LMR==3:
                PWM.setMotorModel(1200,1200,-800,-800)
            elif self.LMR==7:
                PWM.setMotorModel(0,0,0,0)

Infrared=Infrared_Obstacle_Avoidance()
# 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. Make the car turns left.

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

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

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

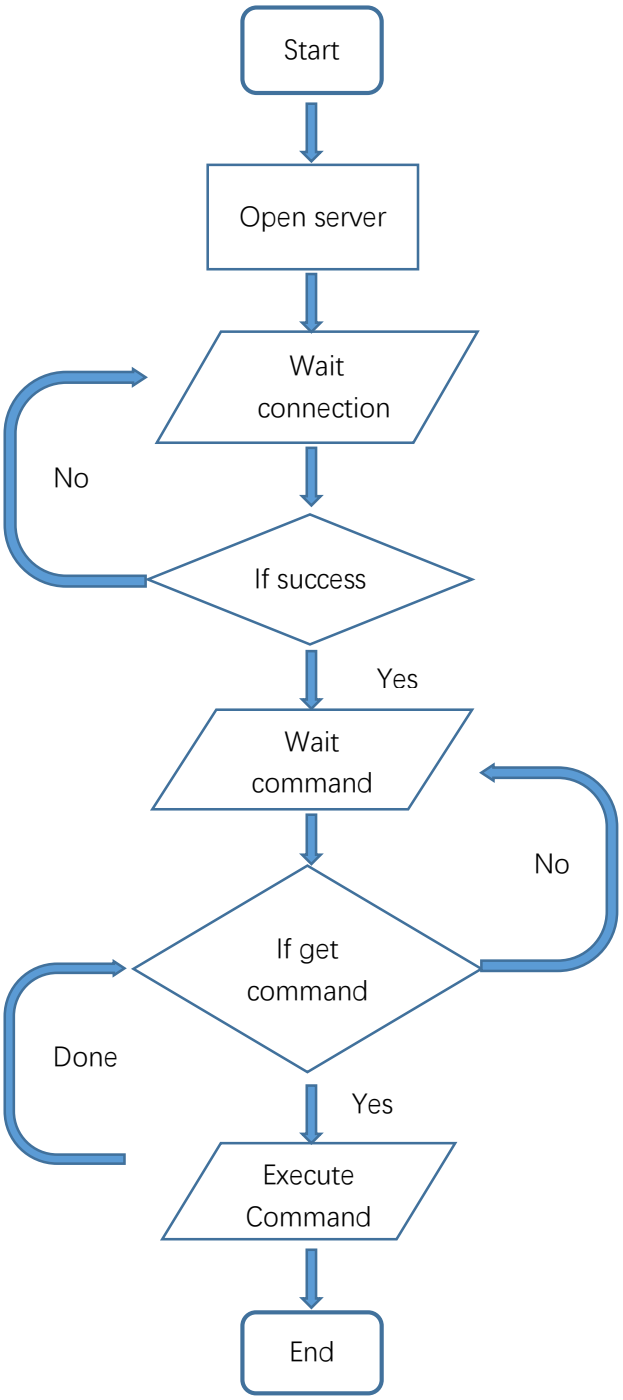
When the sensor on left: low, middle: high, right: high. Make the car turn right lightly.

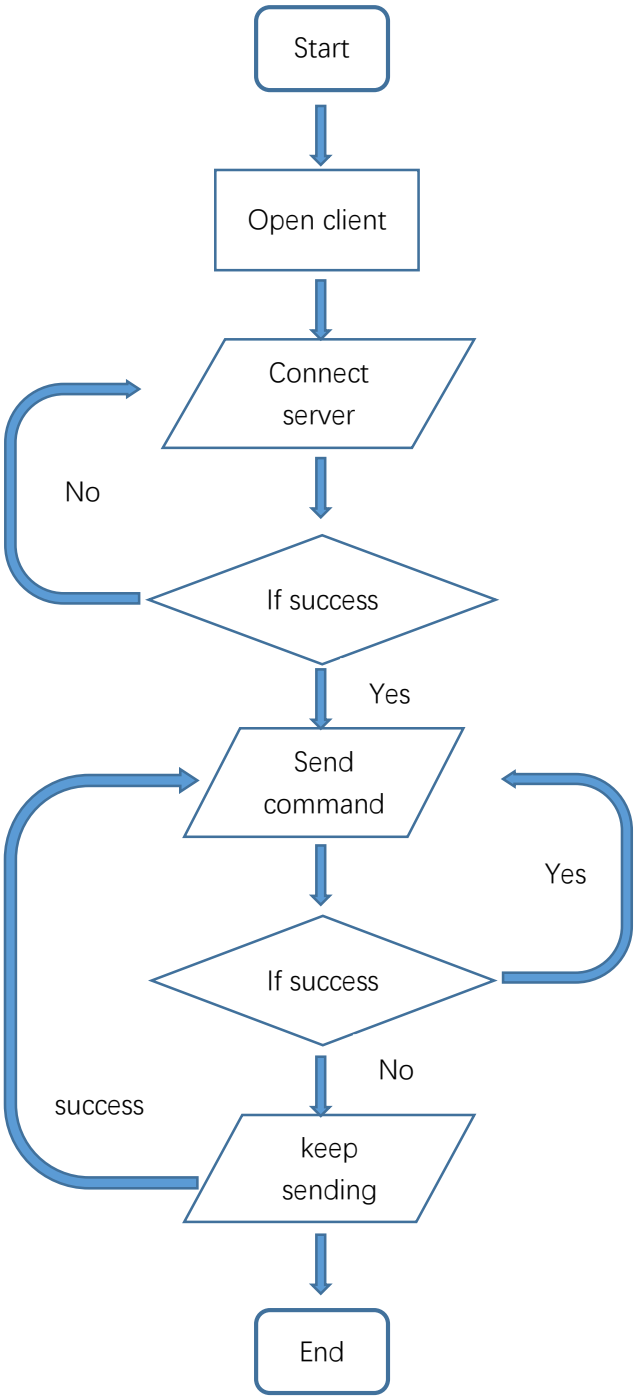
When the car is picked up, the three LEDs of the infrared module light up, and the car stops moving.

.

Chapter 7 Smart video car

The smart video car combines the functions of light tracing, obstacle avoidance and line tracing, transmit video, face detection, LED and other functions. And the server and the client are created. The car can be controlled remotely.





Server

The server works on the Raspberry Pi and can transmit camera data, ultrasonic data, etc. to the client, and 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
12          self.server_socket = socket.socket()
13          self.server_socket.setsockopt(socket.SOL_SOCKET,socket.SO_REUSEPORT,1)
14          self.server_socket.bind((HOST, 8000))
15          self.server_socket.listen(1)
16          print('Server address: '+HOST)
17      def StopTcpServer(self):
18          try:
19              self.connection.close()
20              self.connection1.close()
21          except Exception ,  e:
22              print "No client connection"
23  def sendvideo(self):
24          try:
25              self.connection,self.client_address = self.server_socket.accept()
26              self.connection=self.connection.makefile('rb')
27          except:
28              pass
29          try:
```

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

Open Server

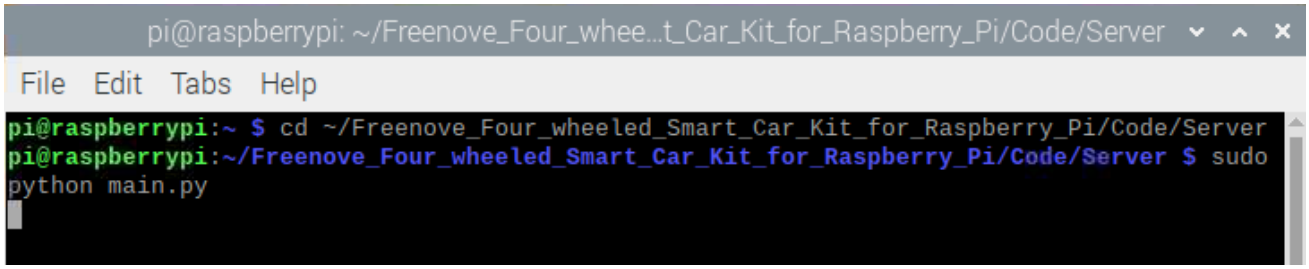
Enter following command in the terminal.

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

```
cd ~/Freenove_Four_wheeled_Smart_Car_Kit_for_Raspberry_Pi/Code/Server
```

2. Run main.py:

```
sudo python main.py
```



The interface is as below:

Click “On” to open the server.

If you don’t like 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_Four_wheeled_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.

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:

```
1  class VideoStreaming:
2      def __init__(self):
3          self.face_cascade = cv2.CascadeClassifier(r'haarcascade_frontalface_default.xml')
4          self.video_Flag=True
5          self.connect_Flag=False #new
6      def StartTcpClient(self,IP):
7          self.client_socket1 = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
8          self.client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
9          print 'StartTcpClient'
10     def StopTcpcClient(self):
```

```
        try:
            self.connection.close()
            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)
            faces = self.face_cascade.detectMultiScale(gray,1.3,5)
            if len(faces)>0 :
                for (x,y,w,h) in faces:
                    img = cv2.circle(img, (x+w/2,y+h/2), (w+h)/4, (0, 255, 0), 2)
        else:
            pass
        cv2.imwrite('video.jpg', img)

    def streaming(self,ip):
        try:
            stream_bytes = b' '
            self.client_socket.connect((ip, 8000))
            self.connection = self.client_socket.makefile('wb')
            while True:
                try:
                    stream_bytes += self.connection.read(1024)
                    first = stream_bytes.find(b'\xff\xd8')
                    last = stream_bytes.find(b'\xff\xd9')
                    if first != -1 and last != -1 :
                        jpg = stream_bytes[first:last + 2]
                        stream_bytes = stream_bytes[last + 2:]
```

```
55                          if self.IsValidImage4Bytes(jpg):
56                              image = cv2.imdecode(np.frombuffer(jpg, dtype=np.uint8),
57  cv2.IMREAD_COLOR)
58                              if self.video_Flag:
59                                  self.face_detect(image)
60                  except:
61                      break
62          except:
63              pass
```

Run client on windows system

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 and 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 python2.7

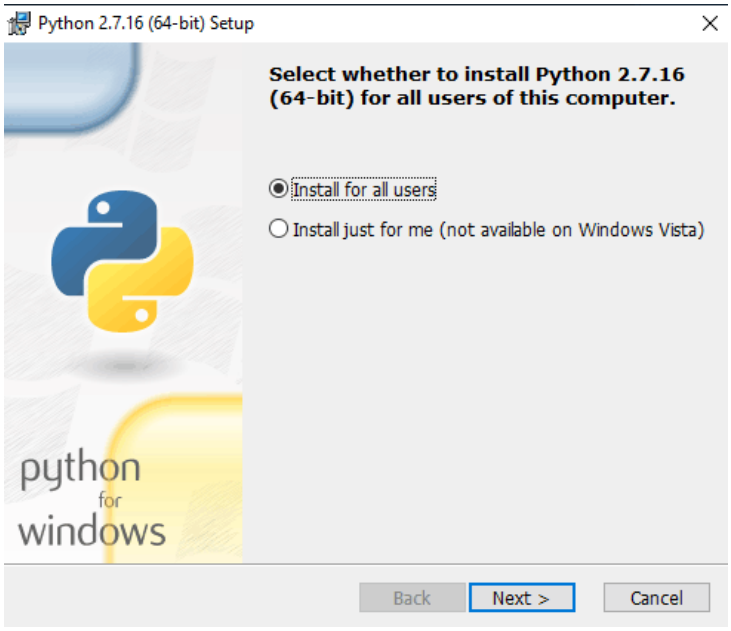
Download the installation file via below:
https://www.python.org/downloads/windows/

Click Latest Python 2 Release - Python 2.7.16.

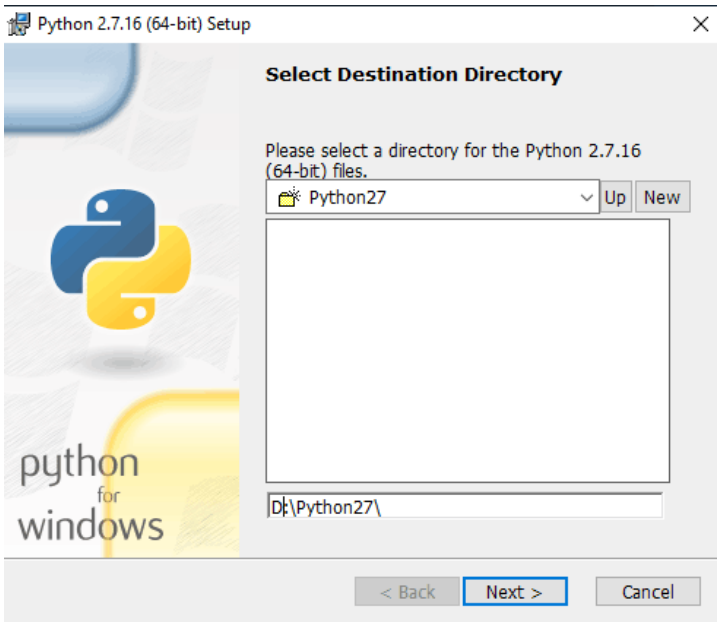
Windows x86-64 MSI installer	Windows
Windows x86 MSI installer	Windows

Select and install according to your system.

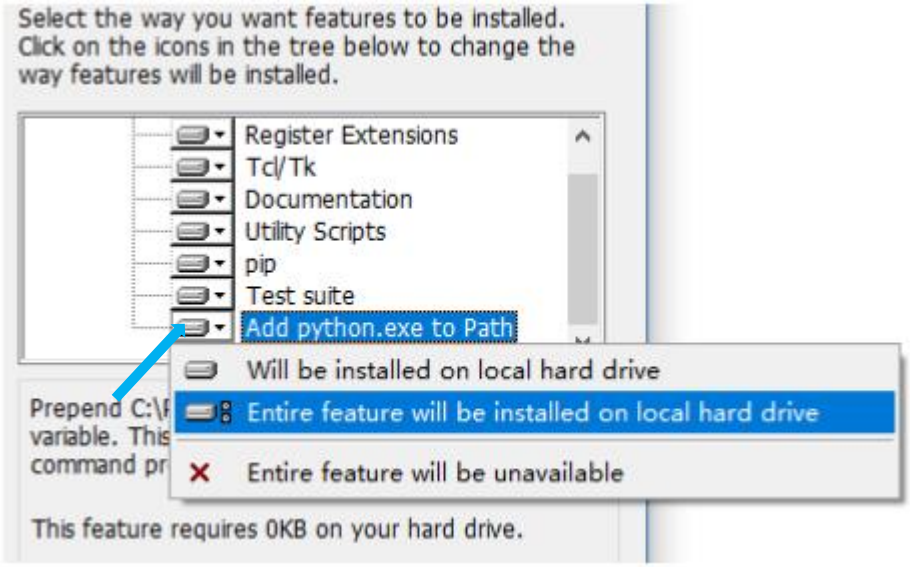
Click Next.



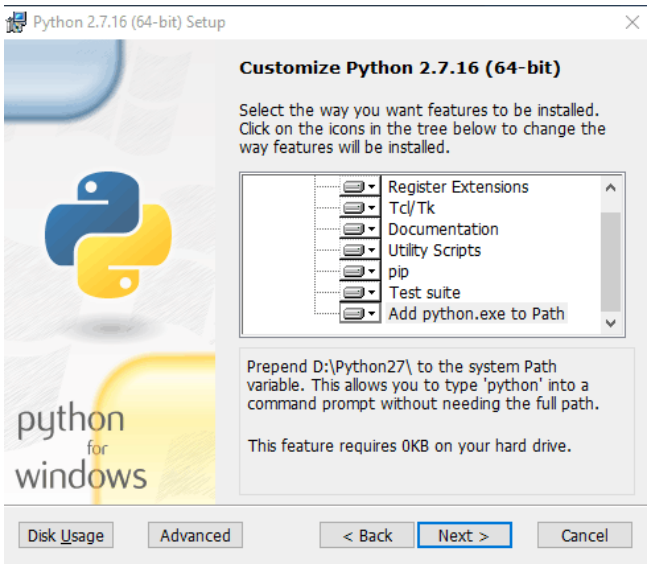
Click Next.



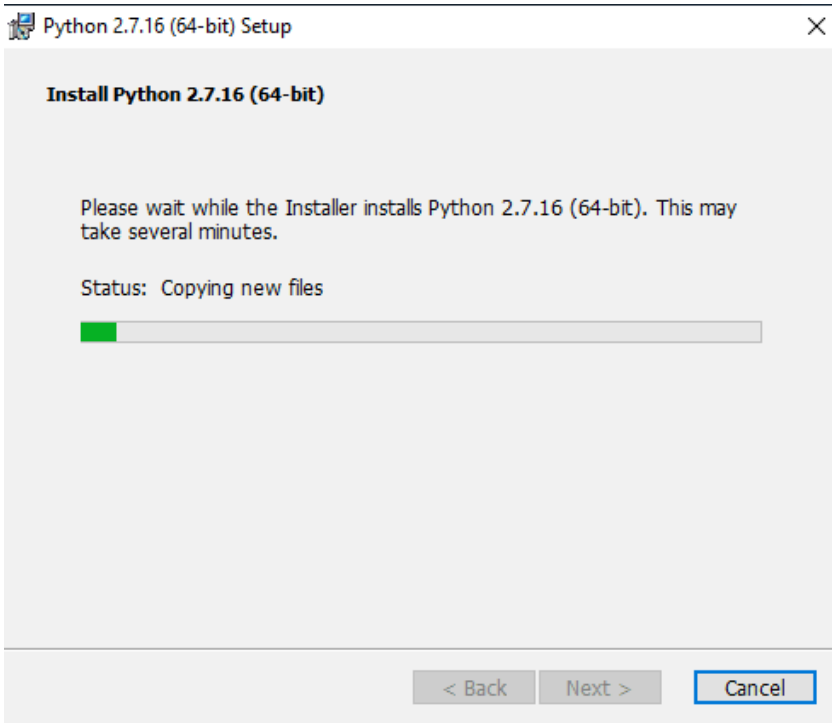
Choose a directory. Then Click Next



During the installation process, you need to Add python.exe to Path.
Select "Entire feature will be installed on local hard drive".



Click Next.



Wait installation with patience.

Click Finish.

Install pyqt4

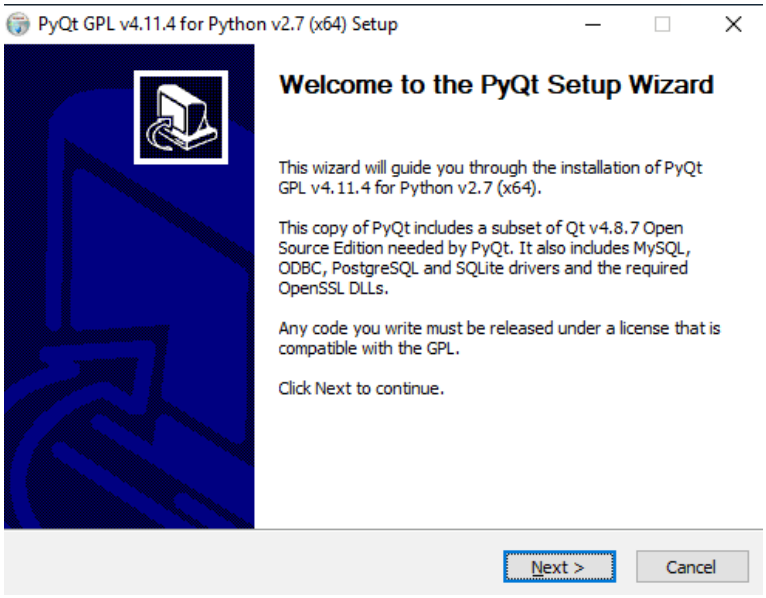
Download installation file via below:

https://sourceforge.net/projects/pyqt/files/PyQt4/PyQt-4.11.4/

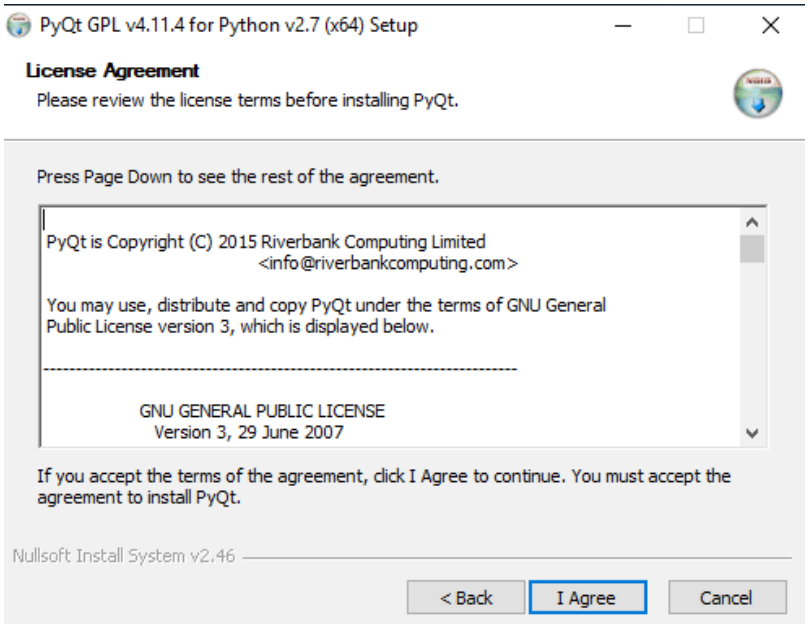
Select corresponding package according to your python version.

PyQt4-4.11.4-gpl-Py2.7-Qt4.8.7-x64.exe	2015-06-11	33.1 MB
PyQt4-4.11.4-gpl-Py2.7-Qt4.8.7-x32.exe	2015-06-11	29.5 MB

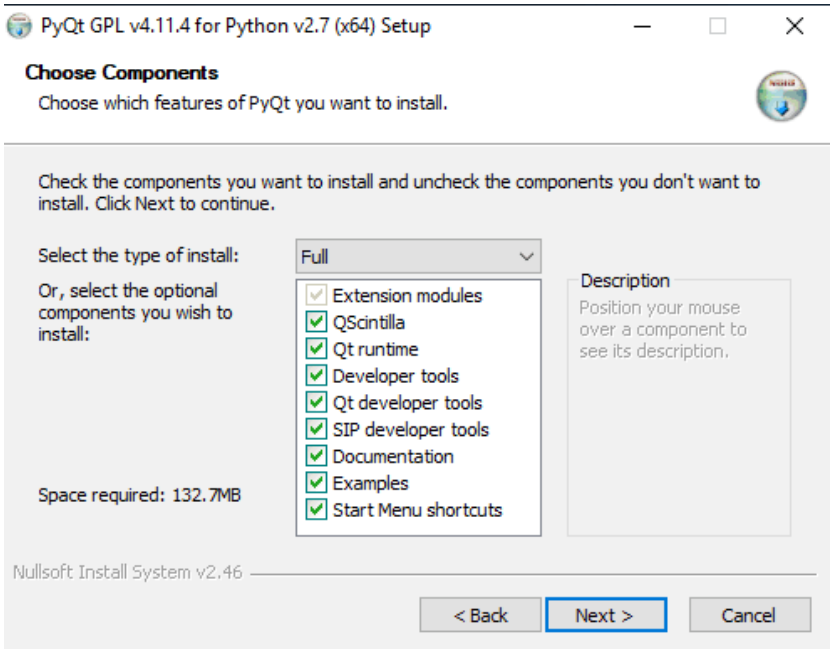
Start installation.



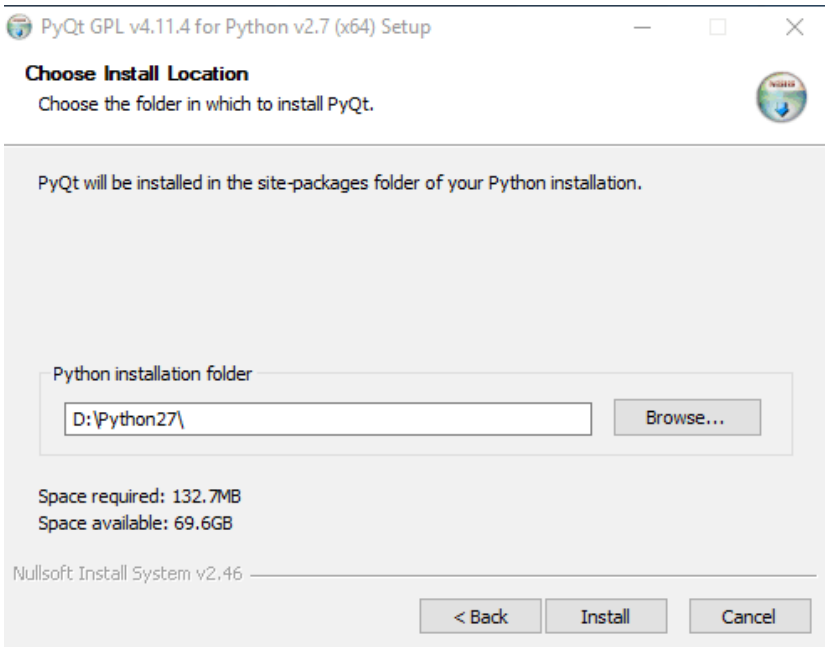
Click Next.



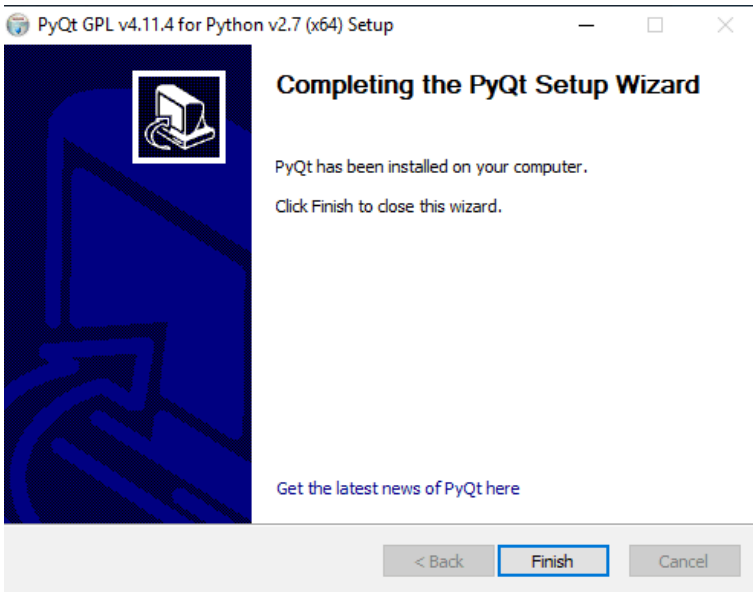
Click I Agree.



Click Next.



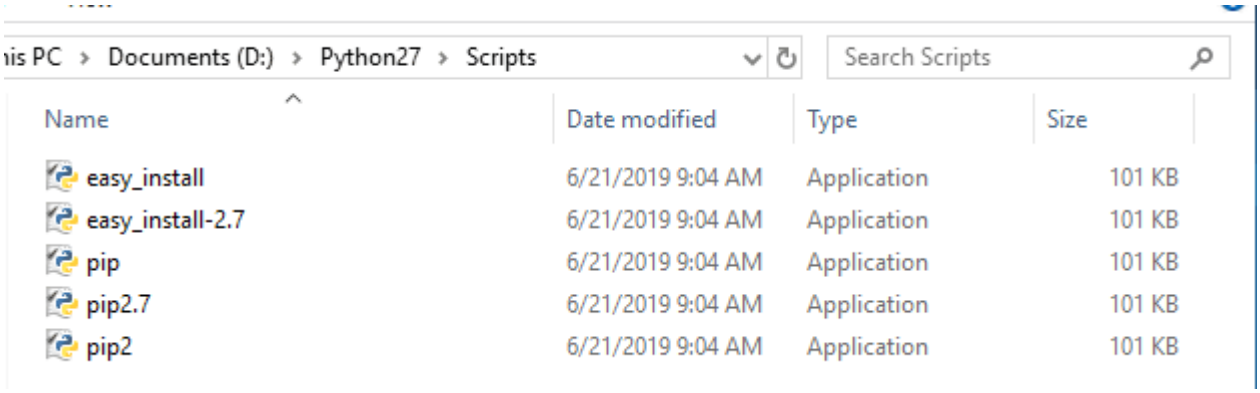
The previous python2 installation path will be selected automatically. (Note: keep the same path as python2), then click Install.



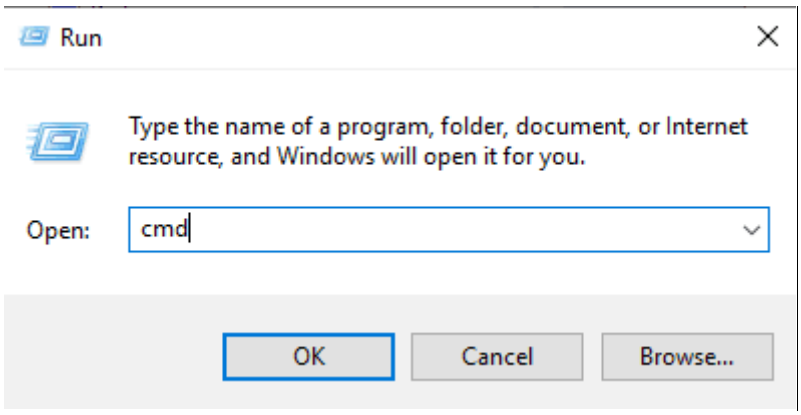
Click Finish.

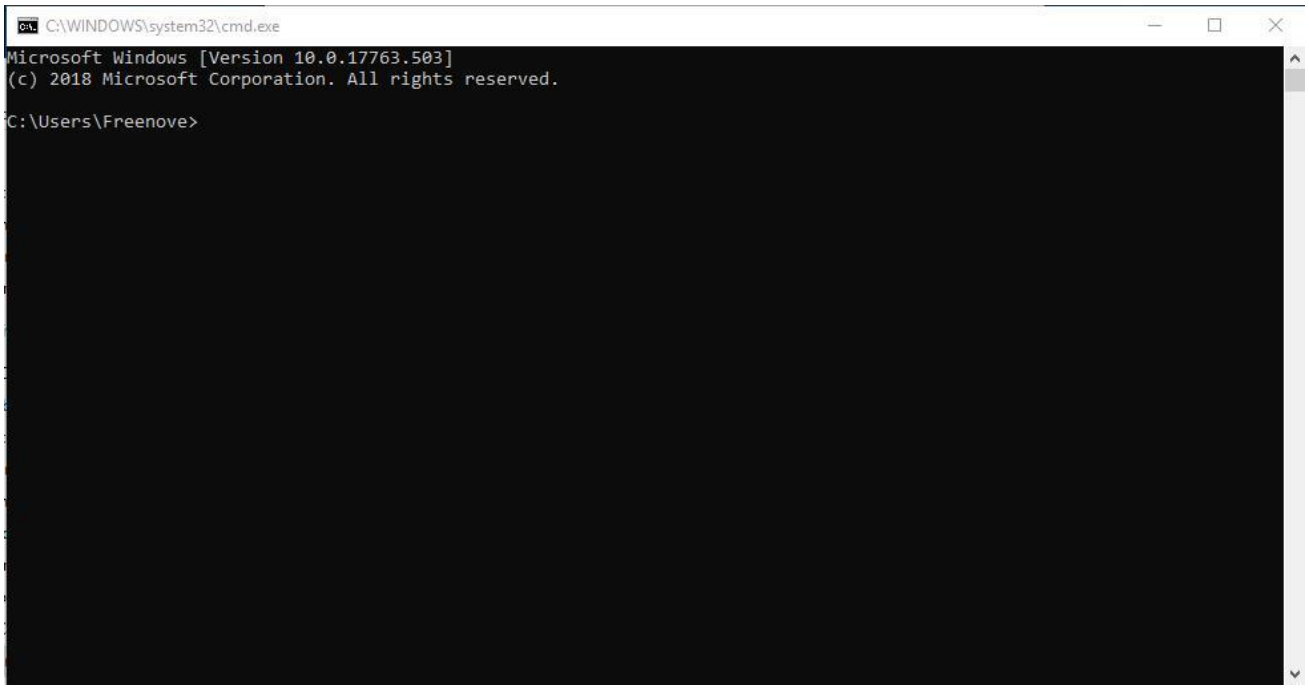
Upgrade pip

Go to the directory Script under python 2.7 installation directory, find the pip file, and confirm that the pip file exists.



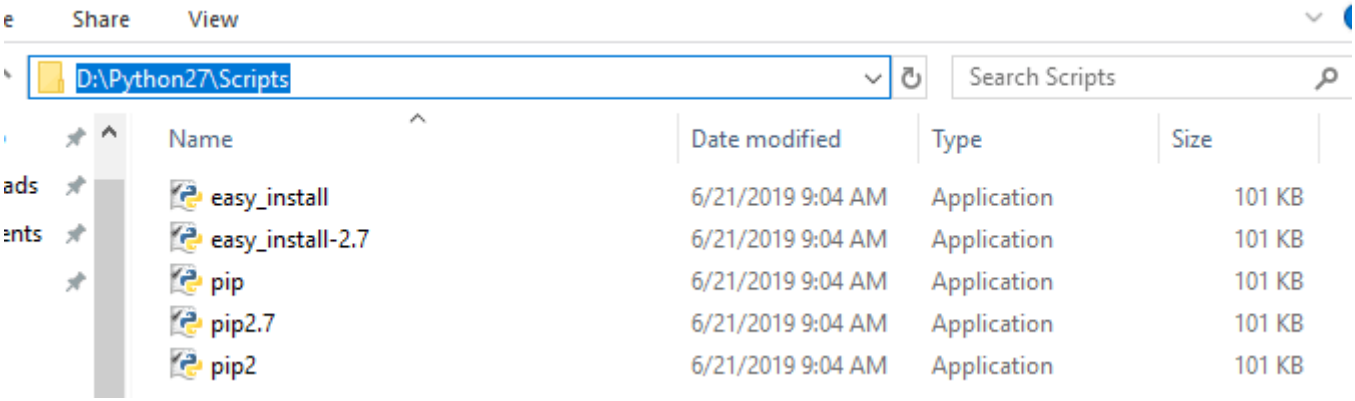
Then press" win+R" and enter "cmd" and click ok.





Enter path of the pip file.

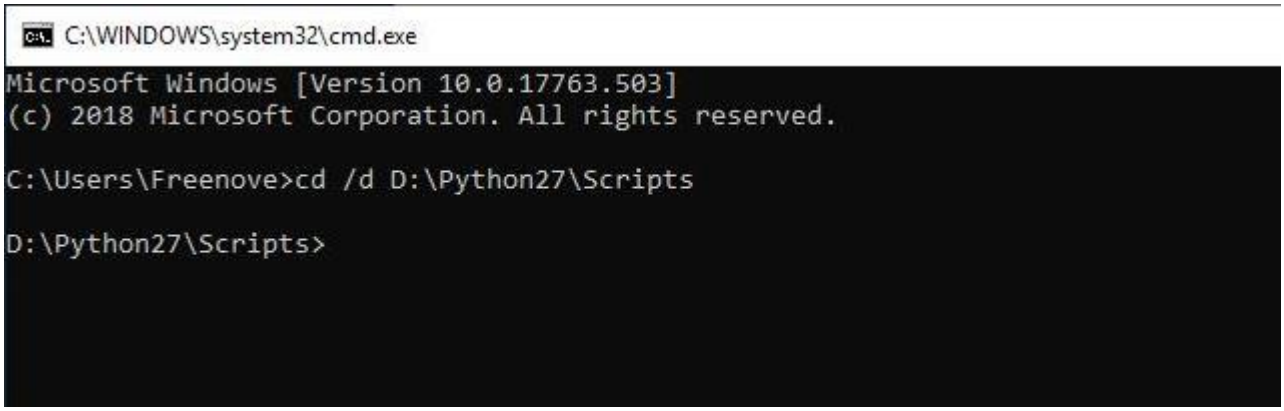
My pip file path is: **D**:\Python27\Scripts



So I should enter following command below to enter directory where pip file is located:

```
cd /d D:\Python27\Scripts
```

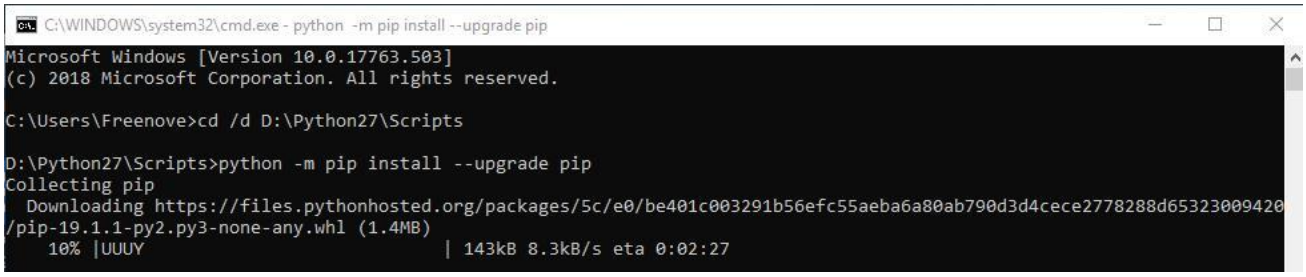
If your pip path is **E**:\Python27\Scripts, you need enter following command: cd /**E** **E**:\Python27\Scripts



Enter following command to upgrade pip:

```
python -m pip install --upgrade pip
```

Wait installation with patience.



If the installation has errors, just enter the command again:

```
python -m pip install --upgrade pip
```

```
C:\WINDOWS\system32\cmd.exe - python -m pip install --upgrade pip
    progress_bar)
  File "D:\Python27\lib\site-packages\pip\_internal\download.py", line 897, in _download_http_url
    _download_url(resp, link, content_file, hashes, progress_bar)
  File "D:\Python27\lib\site-packages\pip\_internal\download.py", line 617, in _download_url
    hashes.check_against_chunks(downloaded_chunks)
  File "D:\Python27\lib\site-packages\pip\_internal\utils\hashes.py", line 48, in check_against_chunks
    for chunk in chunks:
  File "D:\Python27\lib\site-packages\pip\_internal\download.py", line 585, in written_chunks
    for chunk in chunks:
  File "D:\Python27\lib\site-packages\pip\_internal\utils\ui.py", line 159, in iter
    for x in it:
  File "D:\Python27\lib\site-packages\pip\_internal\download.py", line 574, in resp_read
    decode_content=False):
  File "D:\Python27\lib\site-packages\pip\_vendor\urllib3\response.py", line 465, in stream
    data = self.read(amt=amt, decode_content=decode_content)
  File "D:\Python27\lib\site-packages\pip\_vendor\urllib3\response.py", line 430, in read
    raise IncompleteRead(self._fp_bytes_read, self.length_remaining)
  File "D:\Python27\lib\contextlib.py", line 35, in __exit__
    self.gen.throw(type, value, traceback)
  File "D:\Python27\lib\site-packages\pip\_vendor\urllib3\response.py", line 345, in _error_catcher
    raise ReadTimeoutError(self._pool, None, 'Read timed out.')
ReadTimeoutError: HTTPSConnectionPool(host='files.pythonhosted.org', port=443): Read timed out.
You are using pip version 18.1, however version 19.1.1 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.

D:\Python27\Scripts>python -m pip install --upgrade pip
Collecting pip
  Downloading https://files.pythonhosted.org/packages/5c/e0/be401c003291b56efc55aeba6a80ab790d3d4cece2778288d65323009420
/pip-19.1.1-py2.py3-none-any.whl (1.4MB)
    66% |UUUUUUUUUUUUUUUUUUUUUY          | 911kB 20kB/s eta 0:00:23
```

Successful installation is shown as below:

```
D:\Python27\Scripts>python -m pip install --upgrade pip
Collecting pip
  Downloading https://files.pythonhosted.org/packages/5c/e0/be401c003291b56efc55ae
/pip-19.1.1-py2.py3-none-any.whl (1.4MB)
    100% |UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU| 1.4MB 21kB/s
Installing collected packages: pip
  Found existing installation: pip 18.1
    Uninstalling pip-18.1:
      Successfully uninstalled pip-18.1
Successfully installed pip-19.1.1

D:\Python27\Scripts>_
```

Install numpy library

Enter following command to install numpy:

```
pip install numpy
```

Wait with patience. Successful installation is shown as below:

```
D:\Python27\Scripts>pip install numpy
DEPRECATION: Python 2.7 will reach the end of its life on January 1st, 2020. Please upgrade your Python as Python 2.7 wo
n't be maintained after that date. A future version of pip will drop support for Python 2.7.
Collecting numpy
  WARNING: Retrying (Retry(total=4, connect=None, read=None, redirect=None, status=None)) after connection broken by 'Re
adTimeoutError("HTTPSConnectionPool(host='pypi.org', port=443): Read timed out. (read timeout=15)",)': /simple/numpy/
  Downloading https://files.pythonhosted.org/packages/a6/db/18770d6b8419188d56b8ddd9794cb34c2d9f1d272ed8b40fa1ee38a3ca06
/numpy-1.16.4-cp27-cp27m-win_amd64.whl (11.9MB)
    |UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU| 11.9MB 939kB/s
Installing collected packages: numpy
Successfully installed numpy-1.16.4

D:\Python27\Scripts>_
```

Install opencv library

Enter following command to install opencv.

```
pip install opencv-python
```

Wait with patience. Successful installation is shown as below:

```
D:\Python27\Scripts>pip install opencv-python
DEPRECATION: Python 2.7 will reach the end of its life on January 1st, 2020. Please upgrade your Python as Python 2.7 wo
n't be maintained after that date. A future version of pip will drop support for Python 2.7.
Collecting opencv-python
  Downloading https://files.pythonhosted.org/packages/bb/a5/c647e02f0a33e7e4002e935fe6f4e3f68ed89cb688f109337733c76a6285
/opencv_python-4.1.0.25-cp27-cp27m-win_amd64.whl (37.4MB)
     |UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU| 37.4MB 1.7MB/s
Requirement already satisfied: numpy>=1.11.1 in d:\python27\lib\site-packages (from opencv-python) (1.16.4)
Installing collected packages: opencv-python
Successfully installed opencv-python-4.1.0.25
```

Install PIL library

Enter following command to install PIL.

```
pip install Pillow
```

Wait with patience. Successful installation is shown as below:

```
D:\Python27\Scripts>pip install Pillow
DEPRECATION: Python 2.7 will reach the end of its life on January 1st, 2020. Please upgrade your Python as Python 2.7 wo
n't be maintained after that date. A future version of pip will drop support for Python 2.7.
Collecting Pillow
  Downloading https://files.pythonhosted.org/packages/2a/d4/2bd7d1e67aaa42666cc083ee8a5212f239753396c823de85c9a1ca20f57e
/Pillow-6.0.0-cp27-cp27m-win_amd64.whl (1.8MB)
     |UUUUUUUUUUUUUUUUUUUUUUUUUUUUUUUU| 1.8MB 731kB/s
Installing collected packages: Pillow
Successfully installed Pillow-6.0.0

D:\Python27\Scripts>
```

Here, installation of all library and sofreware is completed.

Open client

If have not download the zip file, download it via below:

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

Then unzip it and delete "-master" to rename it to "Freenove_Four_wheeled_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 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 Main.py is located:

```
cd D:\Freenove_Four_wheeled_Smart_Car_Kit_for_Raspberry_Pi\Code\Client
```

3. Run Main.py:

```
python Main.py
```

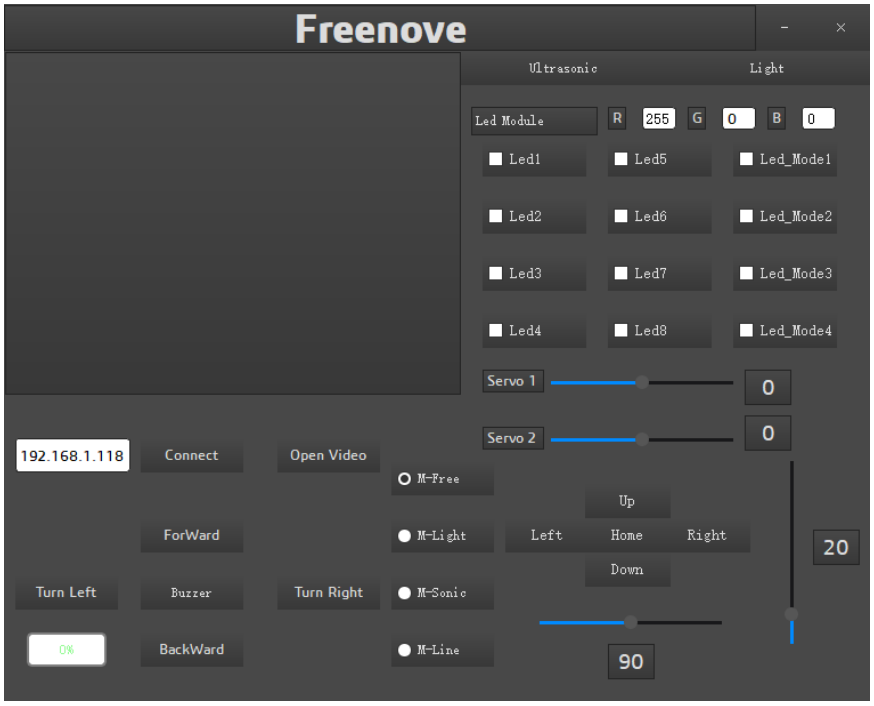
```
C:\Users\Freenove>D:

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_Four_wheeled_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:

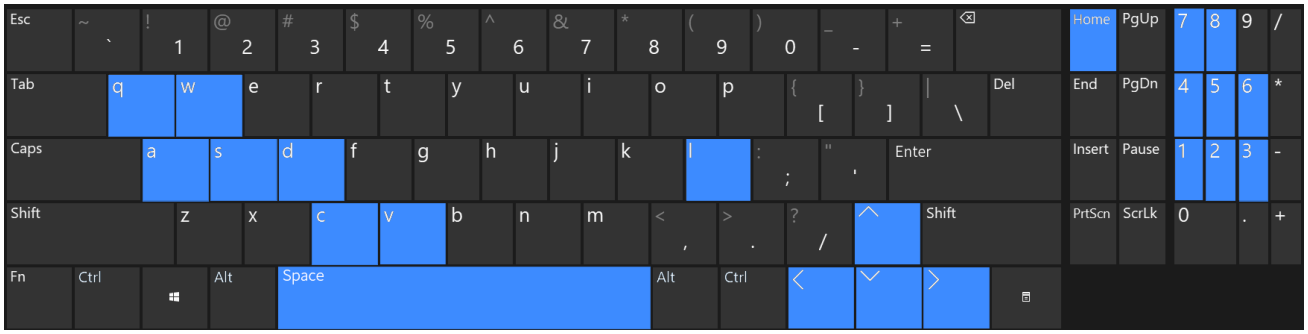


After the client opens successfully, you need open the Raspberry Pi and **open server first**, then enter the IP address of the Raspberry Pi in the white IP edit box, and then click "Connect" to connect smart car to Raspberry Pi. After the connection is successful, you can click on the controls on the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

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

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	Turn left
Turn Right	D	Turn right
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	Q	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 for angle fine tuning. If the servo is not fully centered during installation, you can fine 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

Run client in Raspberry Pi (Linux system)

Run client

Enter the following commands at the terminal.

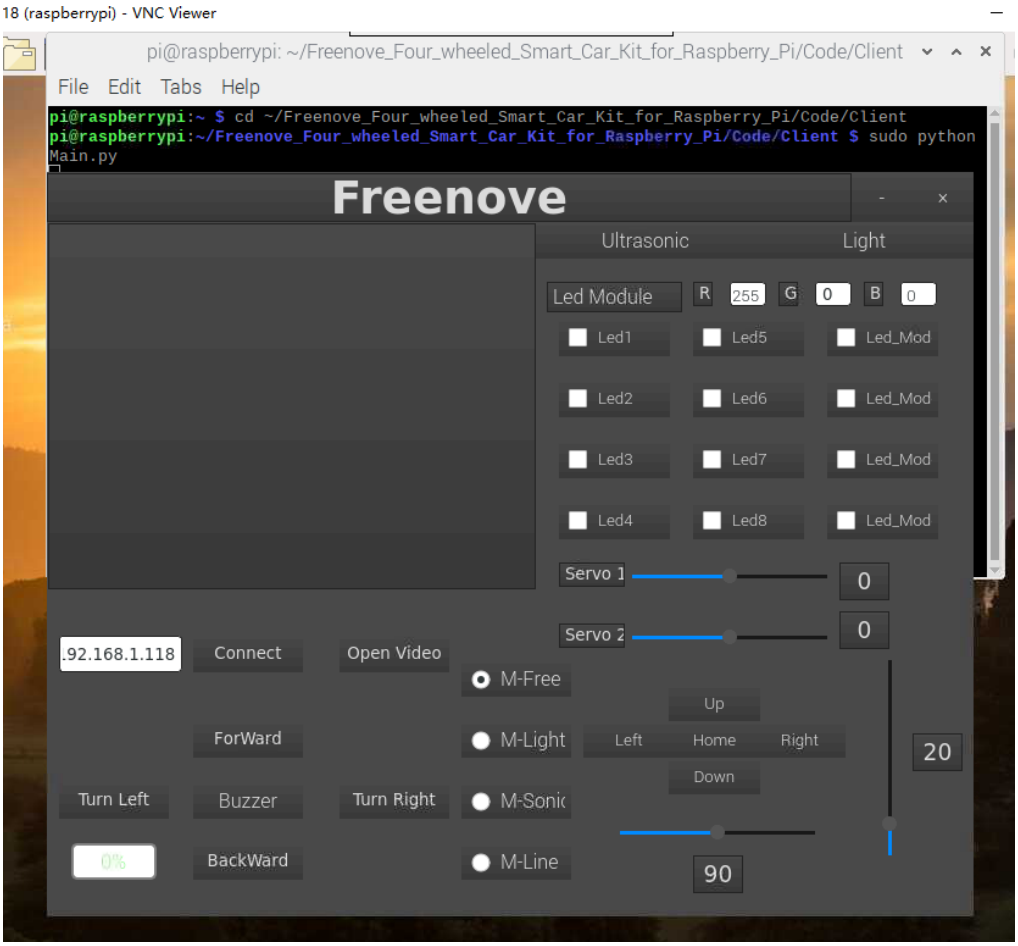
1. Use the cd command to go to the directory where Main.py is located.

```
cd ~/Freenove_Four_wheeled_Smart_Car_Kit_for_Raspberry_Pi/Code/Client
```

2. Run Main.py:

```
sudo python Main.py
```

The interface is shown below:



The control mode of client on Linux is the same as that on Windows, but it does not have the function of face detection.

改如下

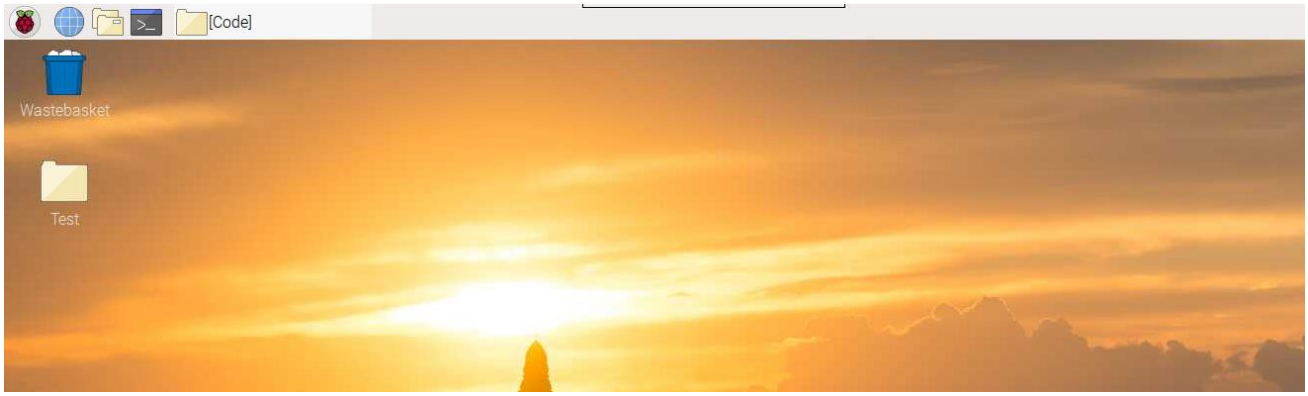
The control mode of client on Linux is the same as that of Windows, but it does not have the function of face detection and tracking.

Free innovation

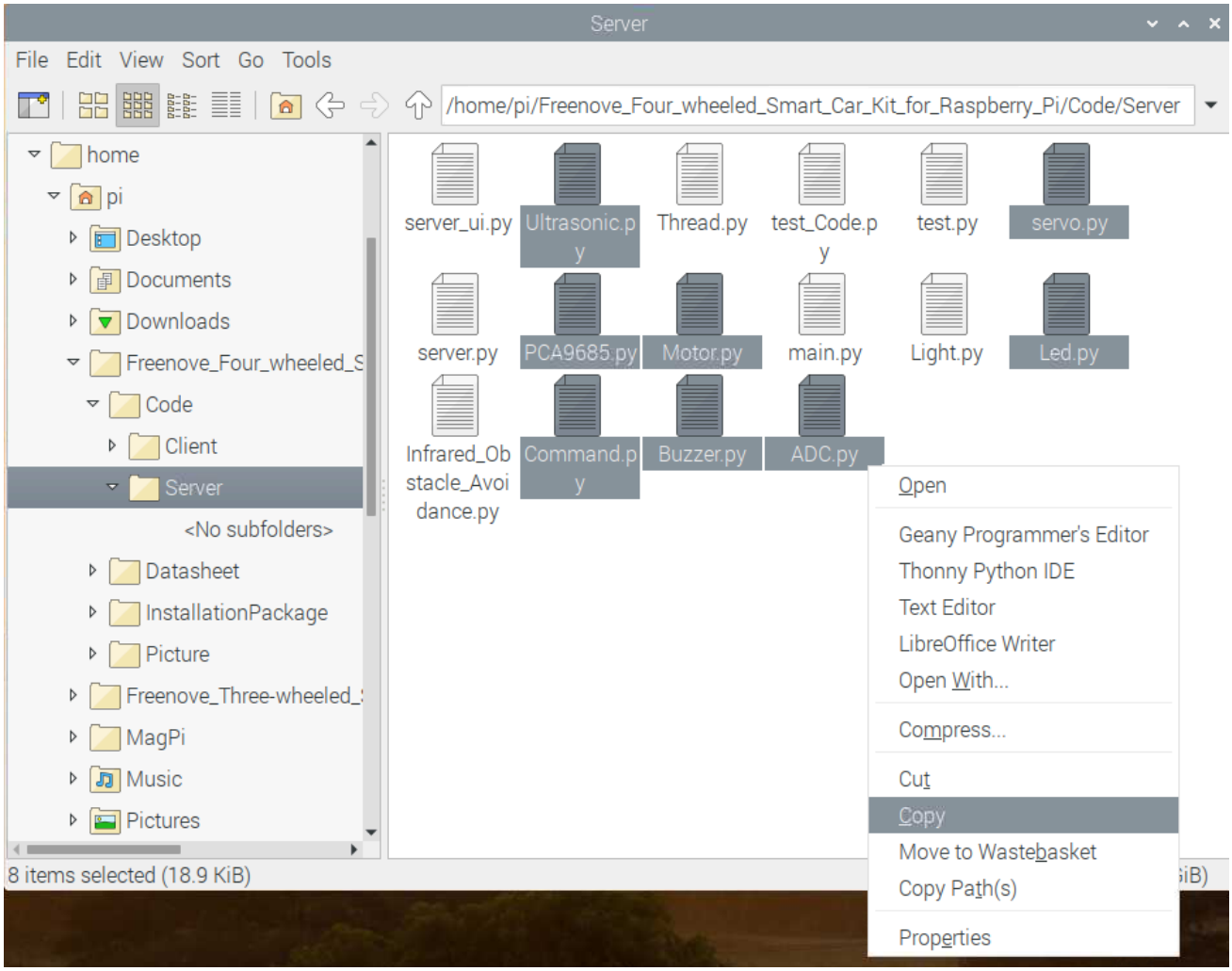
If you want to write your own program to control the car, just follow this section. We will teach you how to program this car.

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

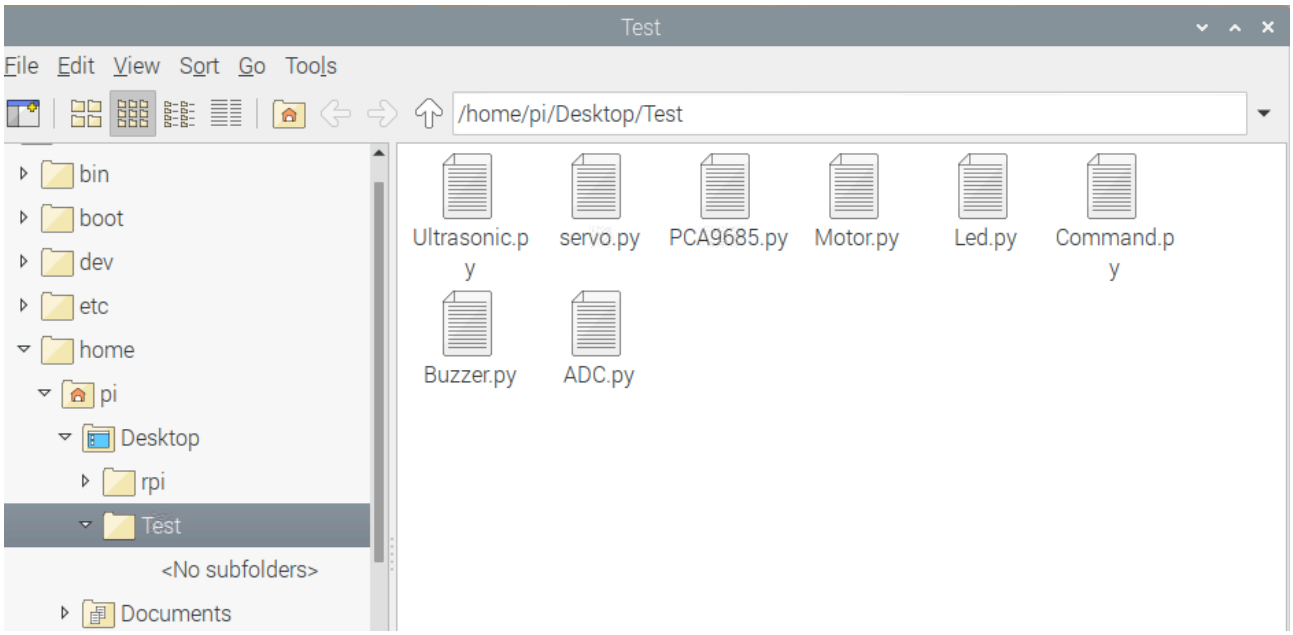
First, turned on S1 and S2. Then open Raspberry Pi, right click and create a new folder on the desktop: Test



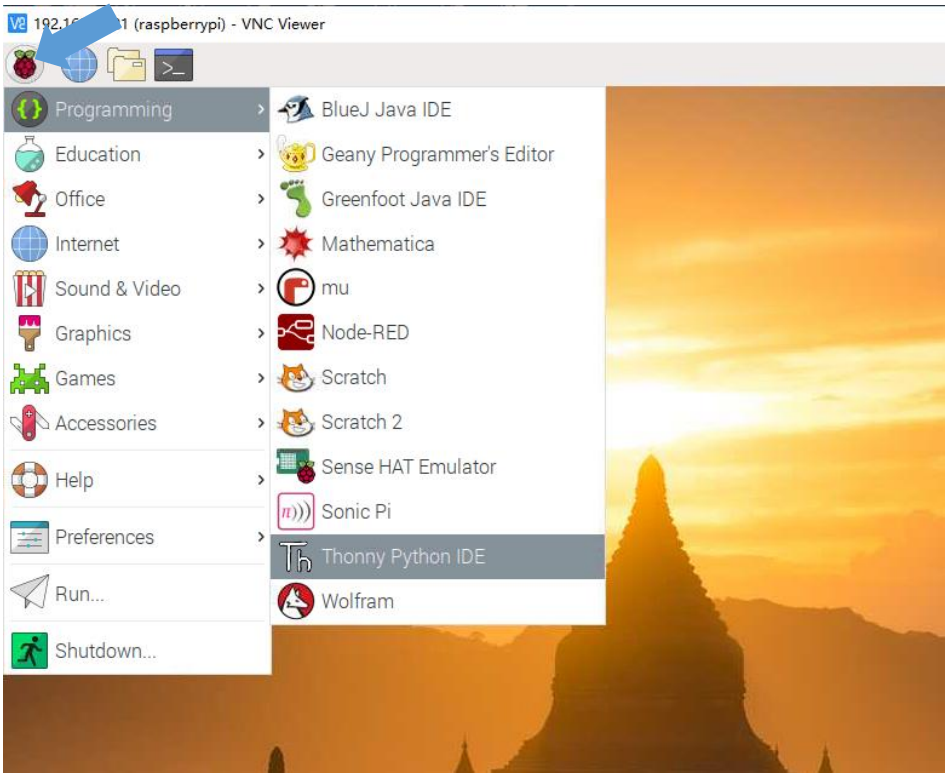
Open Freenove_Four_wheeled_Smart_Car_Kit_for_Raspberry_Pi/Code/Server in your Raspberry Pi and cope following **8 files** into the Test folder we created.

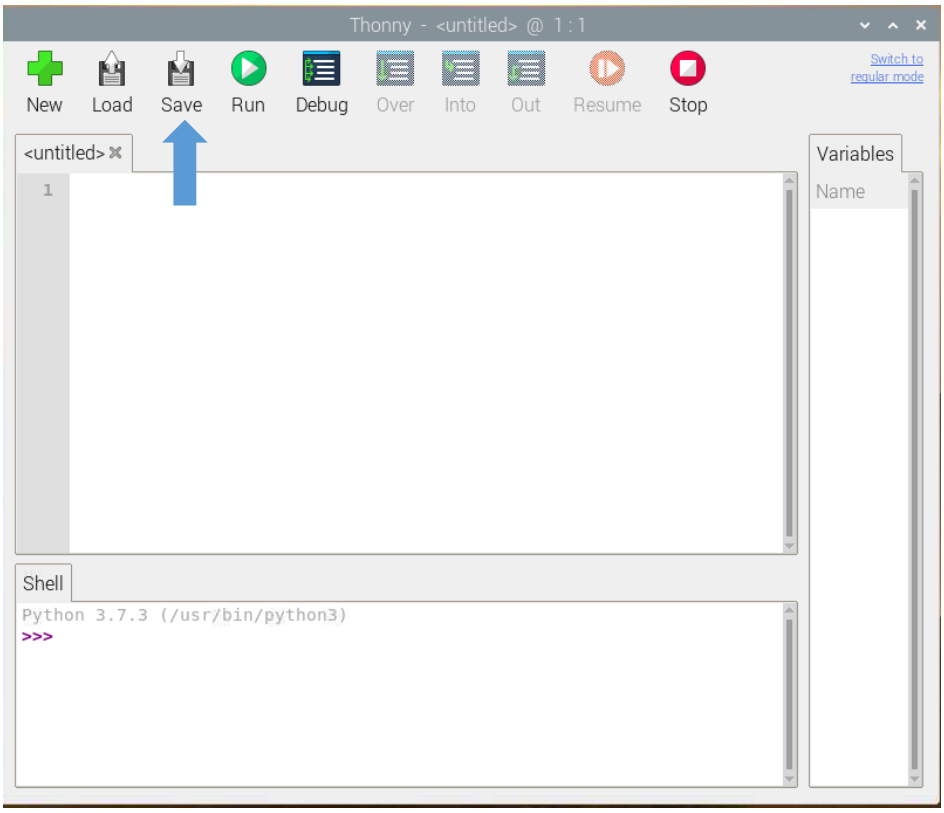


Paste them in Test folder.

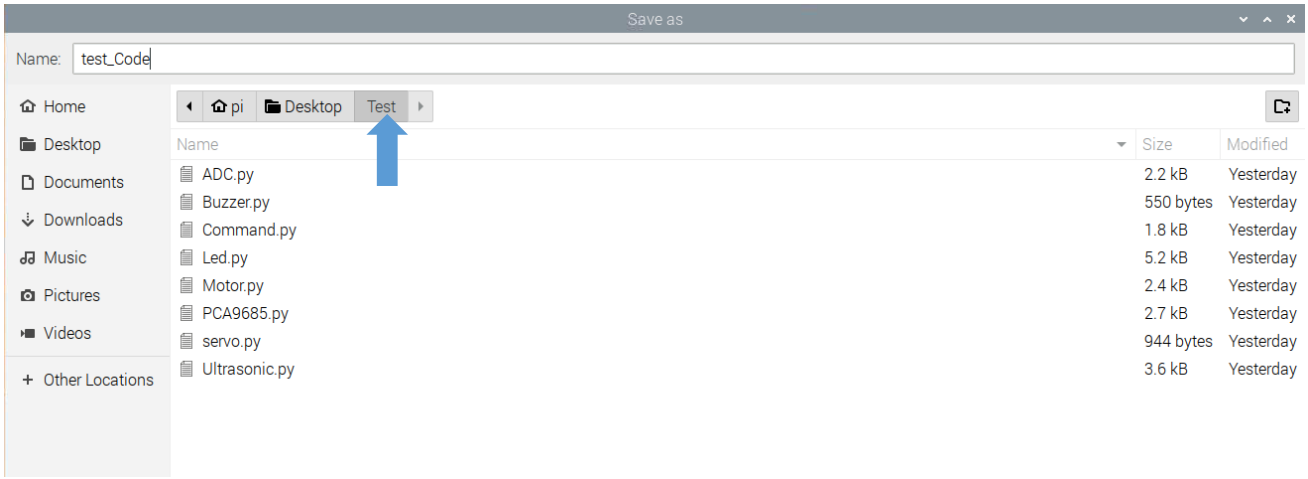


Run Thonny Python IDE

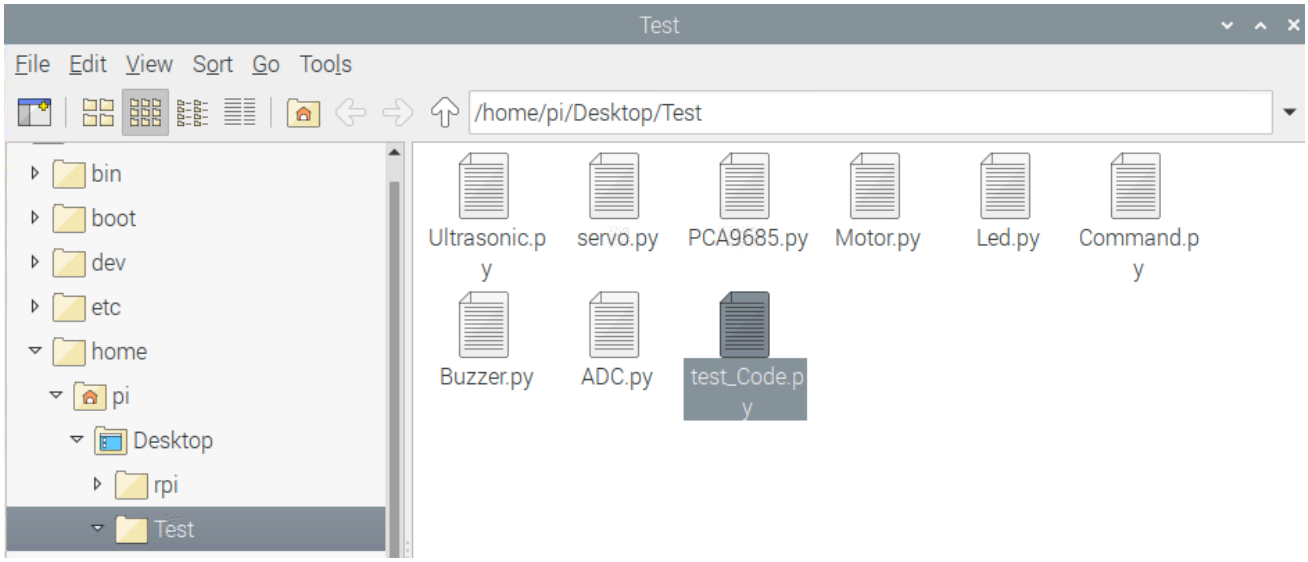




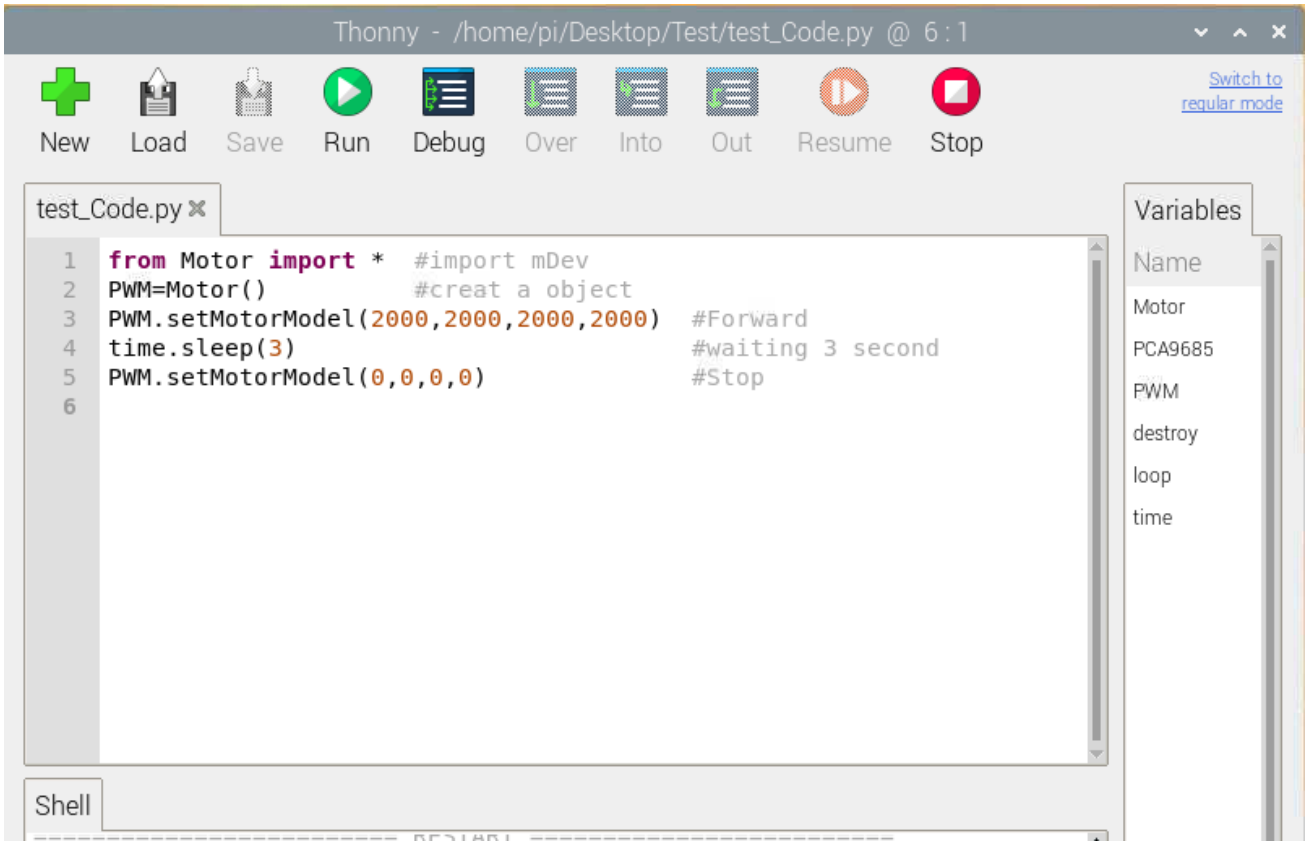
Click Save and save it into the Test folder, with name: test_Code.



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



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



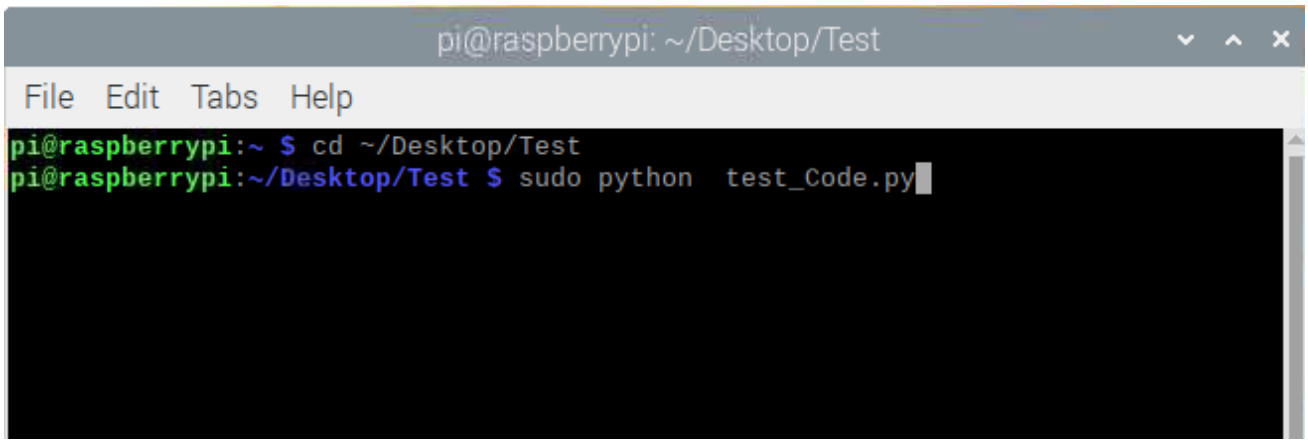
Note: the code and library are written by **Python 2.** You need execute the code with **python 2**.

Open the terminal and use 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(4,255,255,0)                  #yellow
led.ledIndex(7,0,255,0)                    #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(cmd.CMD_START)           #Start
time.sleep(3)                       #wait 3s
buzzer.run(cmd.CMD_STOP)            #Stop
```

Servo. For more details, please refer to Servo.

```
from servo import *   #import Led
pwm=Servo()           #create an object
#Servo rotates from 0 degrees to 180 degrees
for i in range(0,180,1):
    pwm.setServoPwm(1,0,i,0)
    time.sleep(0.01)
#Servo rotates from 180 degrees to 0 degrees
for i in range(180,0,-1):
    pwm.setServoPwm(1,0,i,0)
    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 code can be integrated to 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.