Welcome

Thank you for choosing Freenove products!

About Battery

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

Get Support and Offer Input

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

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

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

support@freenove.com

Safety and Precautions

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

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

About Freenove

Freenove provides open source electronic products and services worldwide.

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

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

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

http://www.freenove.com

Copyright

FREENOVE®

FREE YOUR INNOVATION

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

Contents

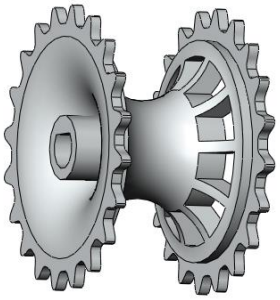
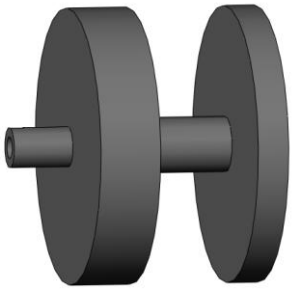
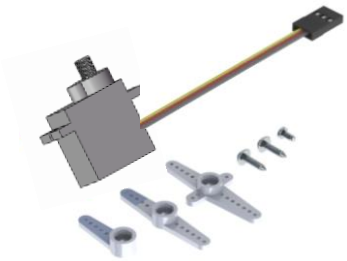
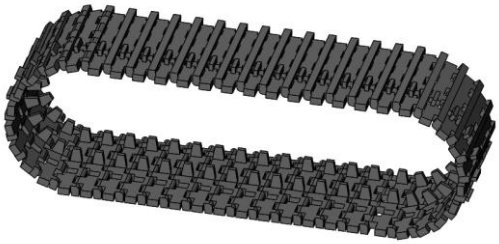
List

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

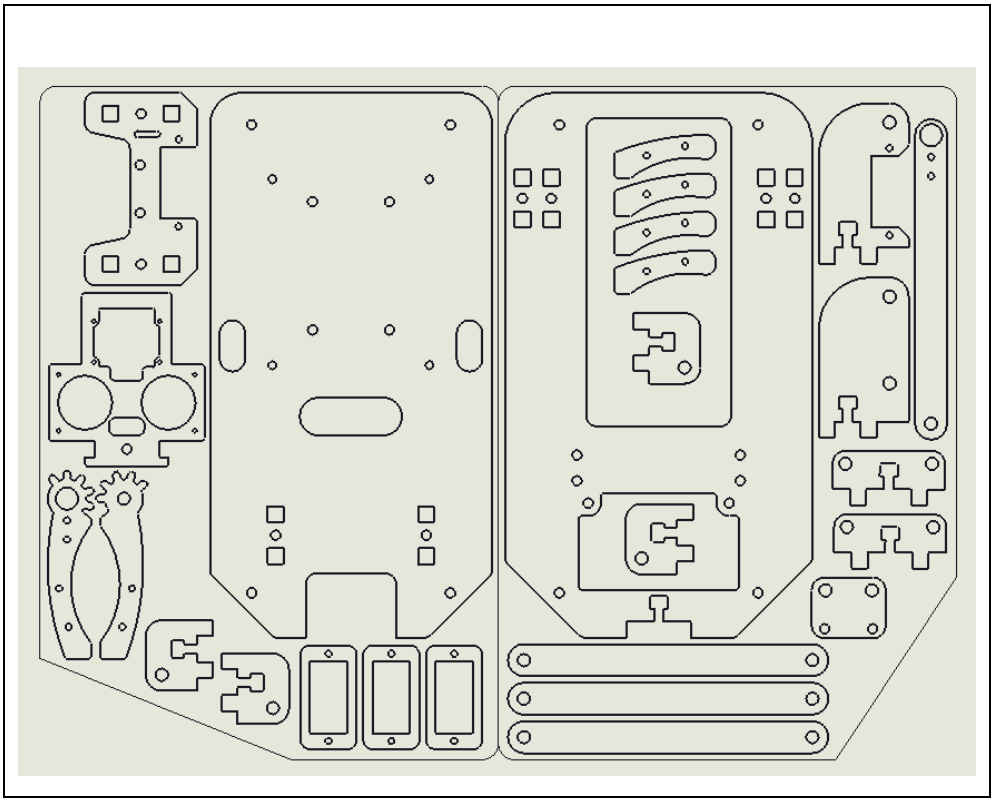
Machinery Parts

M3*24 Brass Standoff  x5 Freenove	M3*10 Brass Standoff  x3 Freenove	M2.5*11 Brass Standoff  x5 Freenove	M2.5*6+6 Screw  x5 Freenove	M4*10 Rivet  x9 Freenove
M3*12 Screw  x13 Freenove	M3*8 Screw  x18 Freenove	M2.5*7 Screw  x9 Freenove	M3*10 Countersunk Head Screw  x6 Freenove	M1.4*5 Screw  x12 Freenove
M2*20 Screw  x10 Freenove	M4*50 Half Tooth Screw  x2 Freenove	M4 Screw  x5 Freenove	M3 Nut  x19 Freenove	M2 Nut  x10 Freenove

Transmission Parts

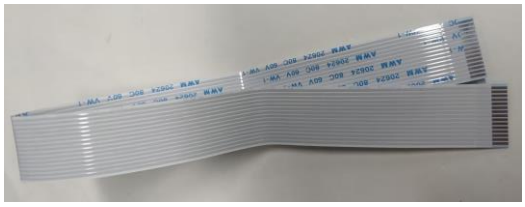
Track drive wheels x2 	Track follower wheels x2 
Servo package x2 	Track x2 
DC speed reduction motor x2 	Motor bracket package x2 

Acrylic Parts

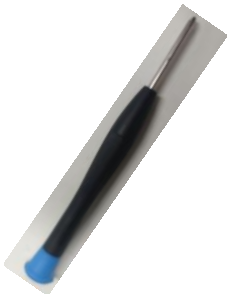


Electronic Parts

Line tracking module x1	Camera x1	HC-SR04 Ultrasonic Module x1
Jumper Wire F/F(4) x1		

XH-2.54-5Pin cable x1 
FPC soft line x1 

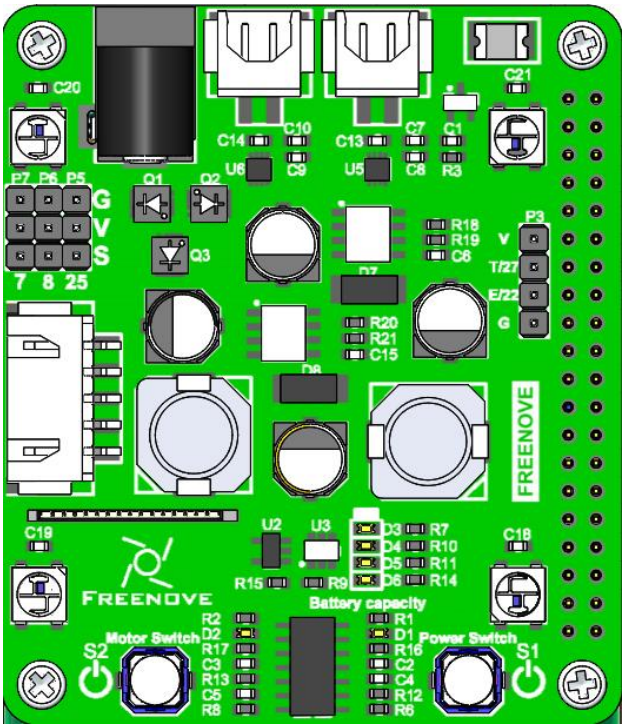
Tools

Cross screwdriver (3mm) x1 	Black tape x1 	Cable Tidy x20cm 	Red ball x1 
Cross screwdriver (2mm) x1 	Aluminum alloy coupling x2 	internal hexagonal wrench x1 	

Self-prepared Parts

2X 3.7V 18650 lithium **rechargeable** batteries with continuous discharge current >3a. **It is easier to find proper battery on eBay than Amazon. Search "18650 high drain" on eBay.** 
Raspberry Pi (Recommended model: Raspberry 4B / 3B+ / 3B) x1 

Tank Smart Car Board for Raspberry Pi



Preface

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

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

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

support@freenove.com

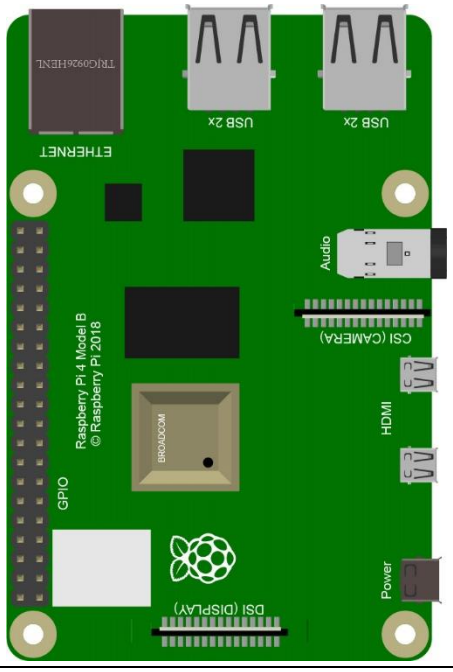
The contents in this book can help enthusiasts with little technical knowledge to make a smart car. If you are very interested in Raspberry Pi, and want to learn how to program and build the circuit, please visit our website www.freenove.com or contact us to buy the kits designed for beginners:
Freenove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

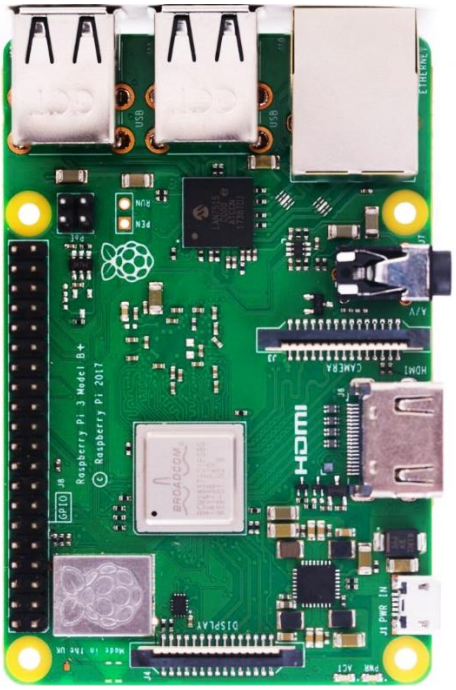
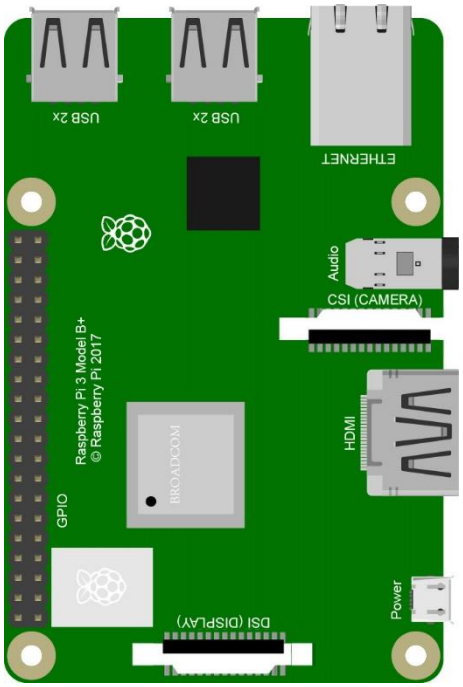
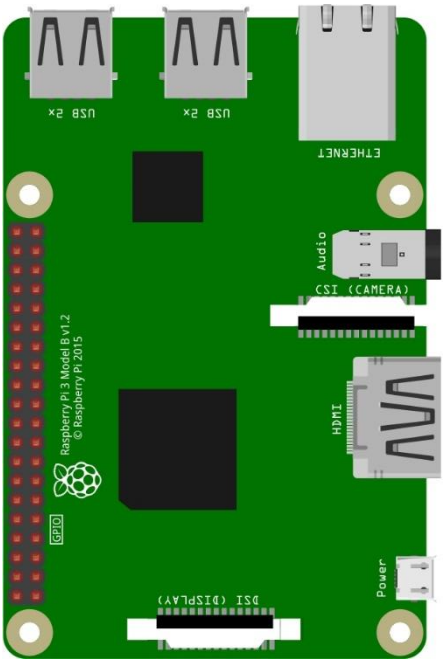
Raspberry Pi Introduction

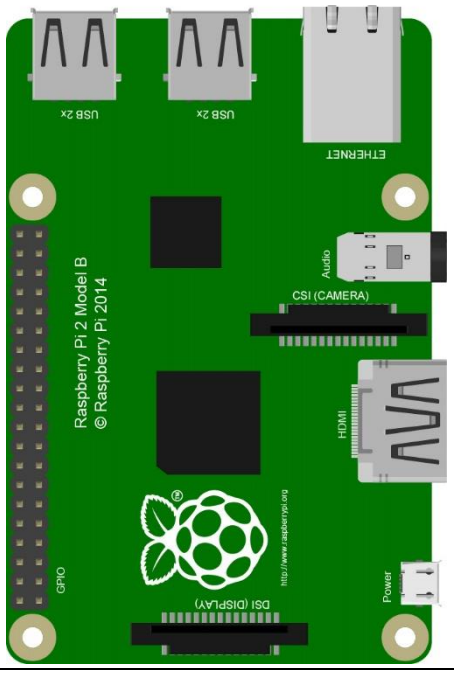
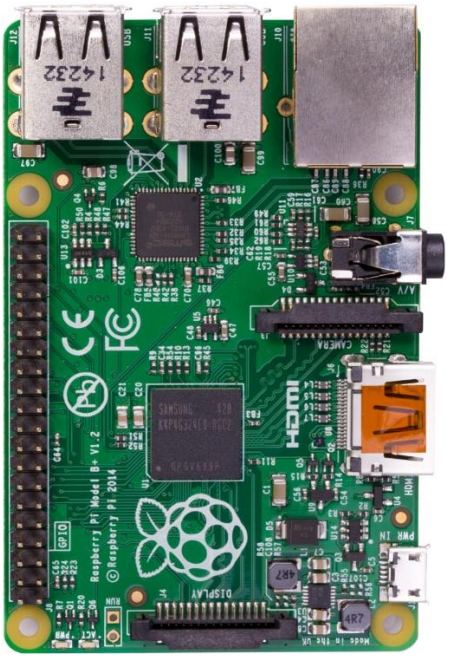
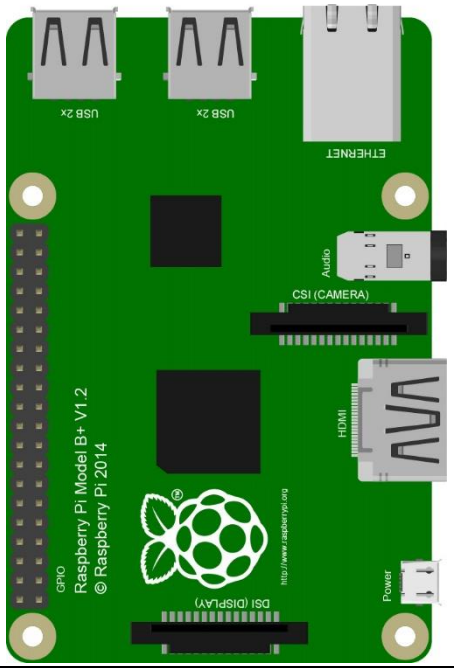
Raspberry Pi (called RPi, RPI, RasPi, the text these words will be used alternately later), a micro-computer with size of a card, quickly swept the world since its debut. It is widely used in desktop workstation, media center, smart home, robots, and even the servers, etc. It can do almost anything,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

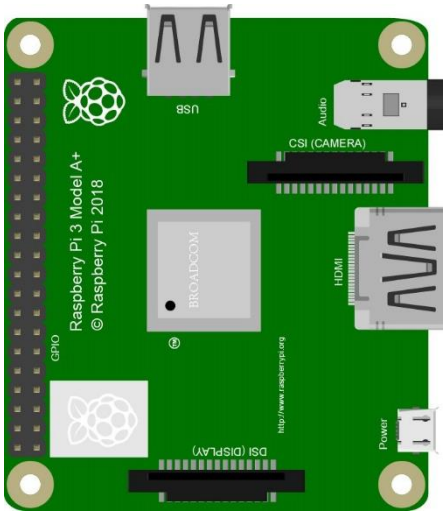
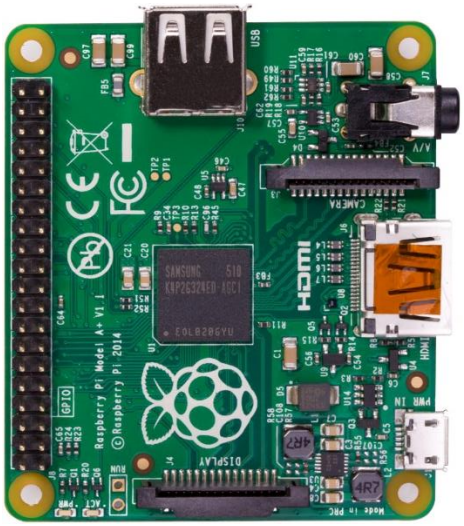
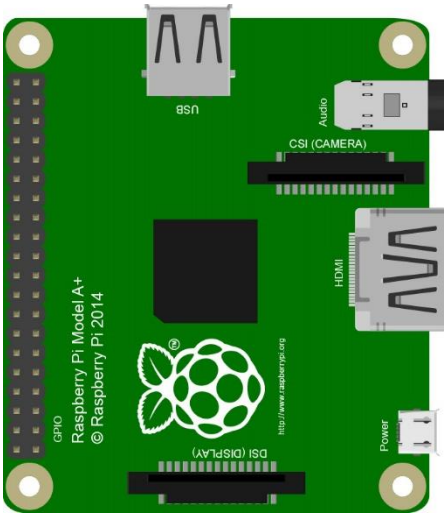
So far, Raspberry Pi has developed to the fourth generation. Changes in versions are accompanied by increase and upgrades in hardware. A type and B type, the first generation of products, have been stopped due to various reasons. Other versions are popular and active and the most important is that they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

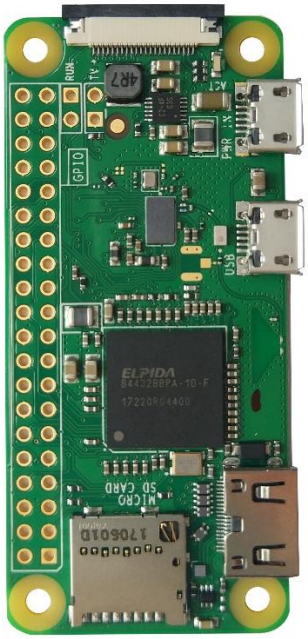
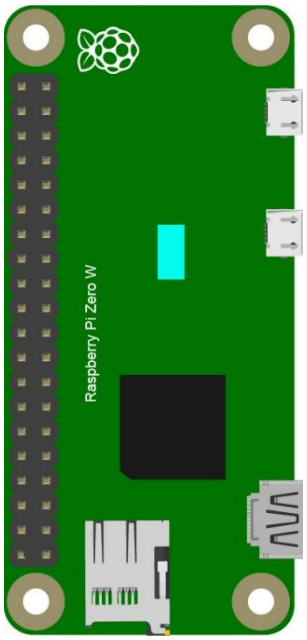
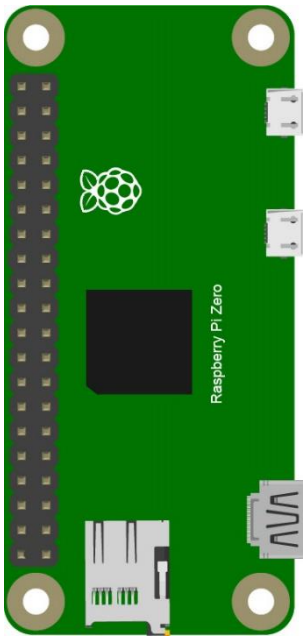
Below are the raspberry pi pictures and model pictures supported by this product.

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

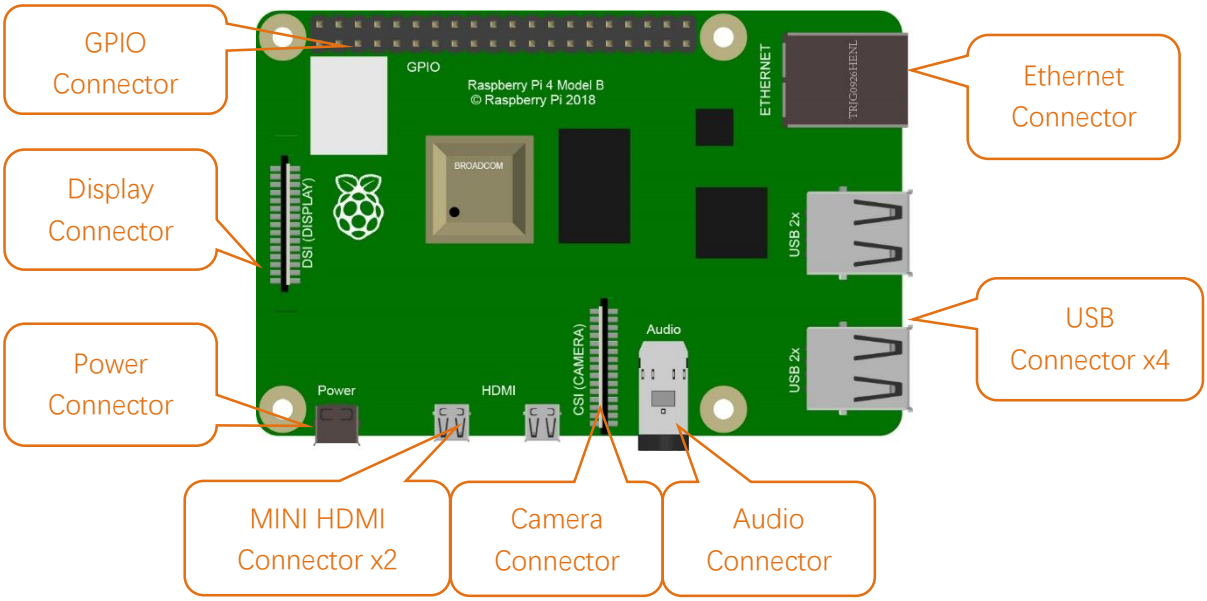
Practicality picture of Raspberry Pi 3 Model B+:	Model diagram of Raspberry Pi 3 Model B+:
	
Practicality picture of Raspberry Pi 3 Model B:	Model diagram of Raspberry Pi 3 Model B:
	

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

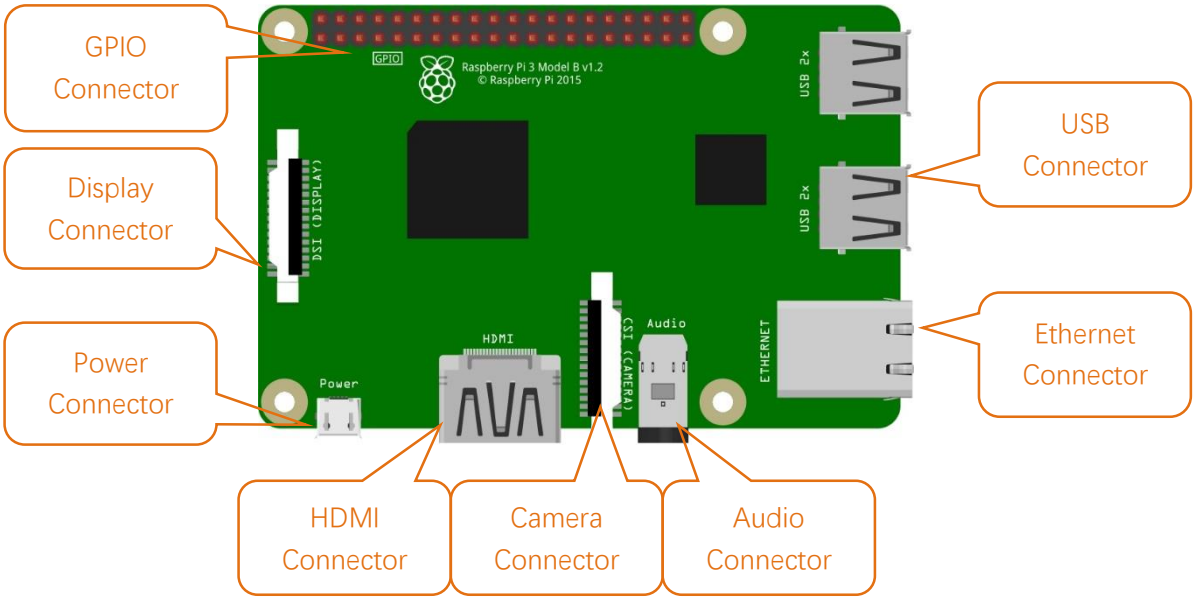
Practicality picture of Raspberry Pi 3 Model A+: 	Model diagram of Raspberry Pi 3 Model A+: 
Practicality picture of Raspberry Pi 1 Model A+: 	Model diagram of Raspberry Pi 1 Model A+: 

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

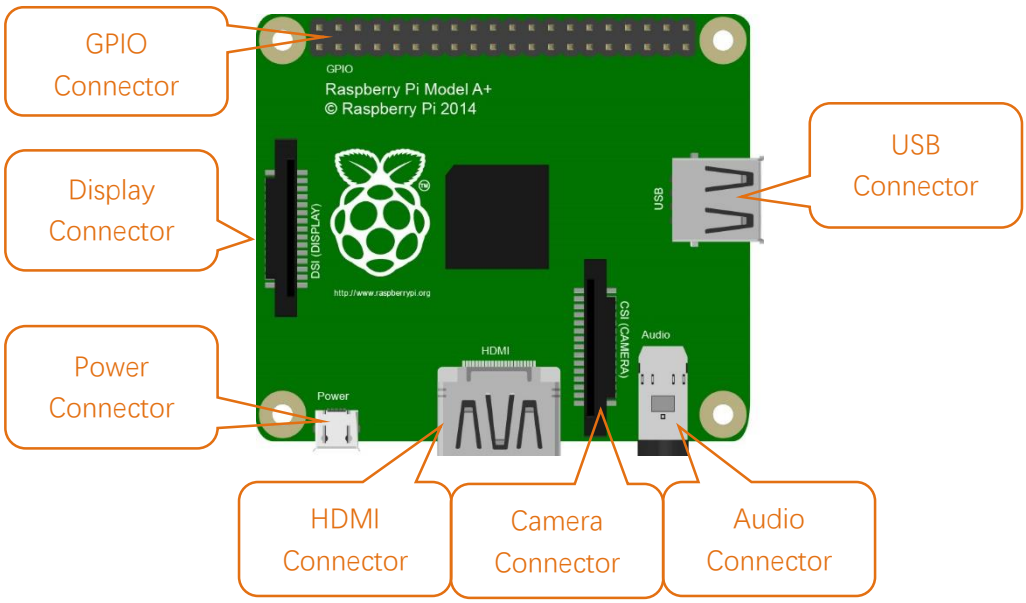
Hardware interface diagram of RPi 4B is shown below:



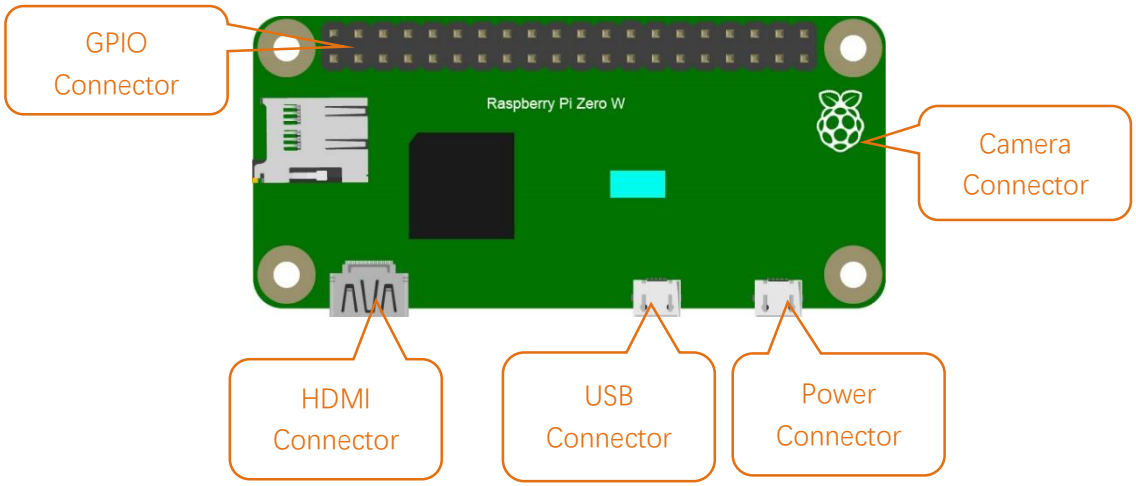
Hardware interface diagram of RPi 3B+/3B/2B/1B+ are shown below:



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



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

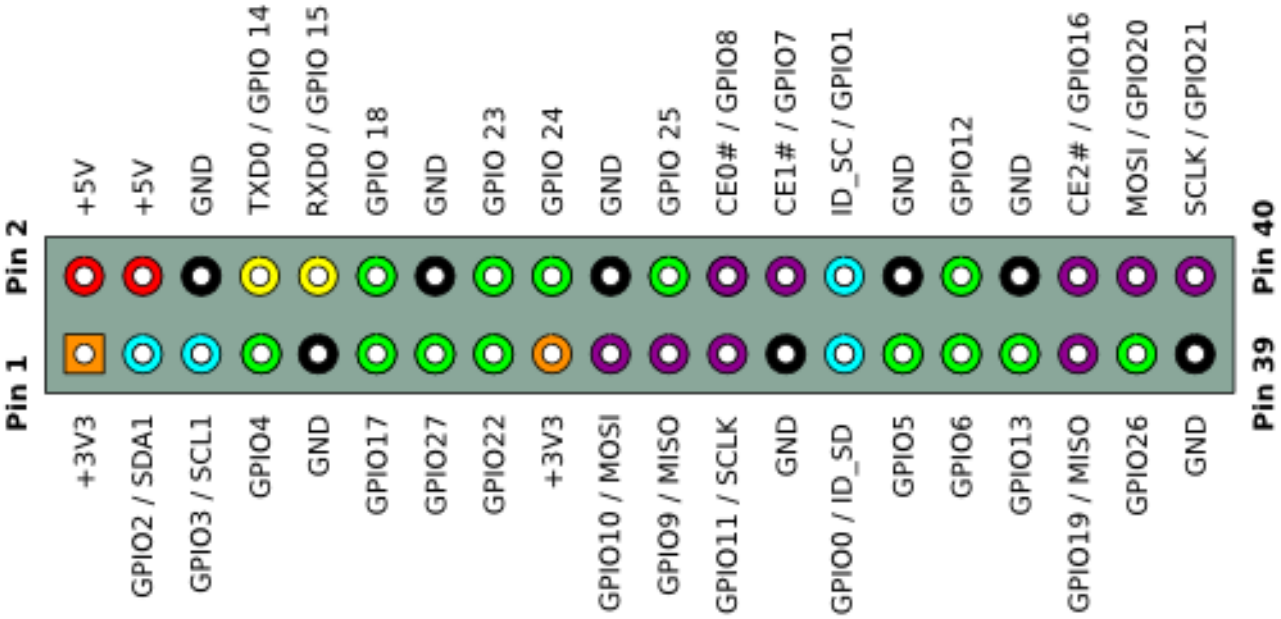


GPIO

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

BCM GPIO Numbering

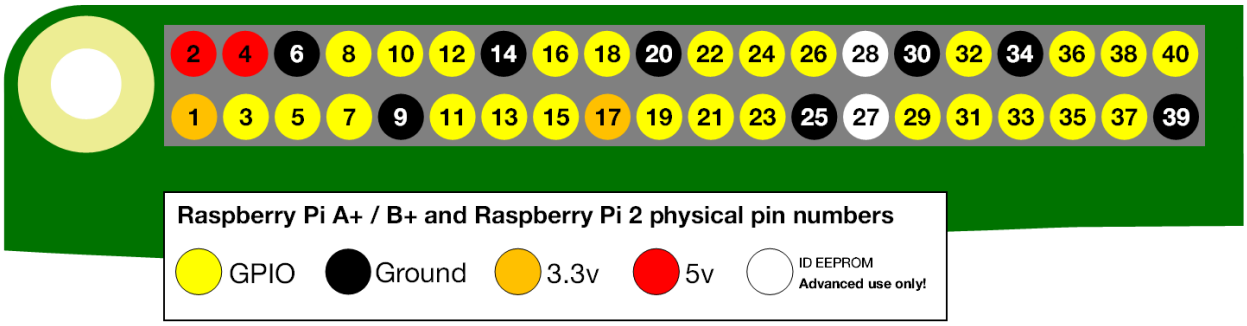
Raspberry Pi CPU use BCM2835/BCM2836/BCM2837of Broadcom. GPIO pin number is set by chip manufacturer. These are the GPIO pins as that computer recognizes. The numbers are unordered and don't make any sense to humans. You will need a printed reference or a reference board that fits over the pins.
Each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



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

PHYSICAL Numbering

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



WiringPi GPIO Numbering

Different from the previous mentioned two kinds of GPIO serial numbers, RPi GPIO serial number of the WiringPi was renumbered. Here we have three kinds of GPIO number mode: based on the number of BCM chip, based on the physical sequence number and based on wiringPi. The correspondence between these three GPIO numbers is shown below:

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

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

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

```
gpio readall
```

```
pi@raspberrypi:~ $ gpio readall
 +-----+-----+---------+------+---+---Pi 3---+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 |     |     |    3.3v |      |   |  1 || 2  |   |      | 5v      |     |     |
 |   2 |   8 |   SDA.1 | ALT0 | 1 |  3 || 4  |   |      | 5V      |     |     |
 |   3 |   9 |   SCL.1 | ALT0 | 1 |  5 || 6  |   |      | 0v      |     |     |
 |   4 |   7 | GPIO. 7 |   IN | 1 |  7 || 8  | 1 | ALT5 | TxD     | 15  | 14  |
 |     |     |      0v |      |   |  9 || 10 | 1 | ALT5 | RxD     | 16  | 15  |
 |  17 |   0 | GPIO. 0 |   IN | 0 | 11 || 12 | 0 | IN   | GPIO. 1 | 1   | 18  |
 |  27 |   2 | GPIO. 2 |   IN | 0 | 13 || 14 |   |      | 0v      |     |     |
 |  22 |   3 | GPIO. 3 |   IN | 0 | 15 || 16 | 0 | IN   | GPIO. 4 | 4   | 23  |
 |     |     |    3.3v |      |   | 17 || 18 | 0 | IN   | GPIO. 5 | 5   | 24  |
 |  10 |  12 |    MOSI | ALT0 | 0 | 19 || 20 |   |      | 0v      |     |     |
 |   9 |  13 |    MISO | ALT0 | 0 | 21 || 22 | 0 | IN   | GPIO. 6 | 6   | 25  |
 |  11 |  14 |    SCLK | ALT0 | 0 | 23 || 24 | 1 | OUT  | CE0     | 10  | 8   |
 |     |     |      0v |      |   | 25 || 26 | 1 | OUT  | CE1     | 11  | 7   |
 |   0 |  30 |   SDA.0 |   IN | 1 | 27 || 28 | 1 | IN   | SCL.0   | 31  | 1   |
 |   5 |  21 | GPIO.21 |   IN | 1 | 29 || 30 |   |      | 0v      |     |     |
 |   6 |  22 | GPIO.22 |   IN | 1 | 31 || 32 | 0 | IN   | GPIO.26 | 26  | 12  |
 |  13 |  23 | GPIO.23 |   IN | 0 | 33 || 34 |   |      | 0v      |     |     |
 |  19 |  24 | GPIO.24 |   IN | 0 | 35 || 36 | 0 | IN   | GPIO.27 | 27  | 16  |
 |  26 |  25 | GPIO.25 |   IN | 0 | 37 || 38 | 0 | IN   | GPIO.28 | 28  | 20  |
 |     |     |      0v |      |   | 39 || 40 | 0 | IN   | GPIO.29 | 29  | 21  |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+---Pi 3---+---+------+---------+-----+-----+
```

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

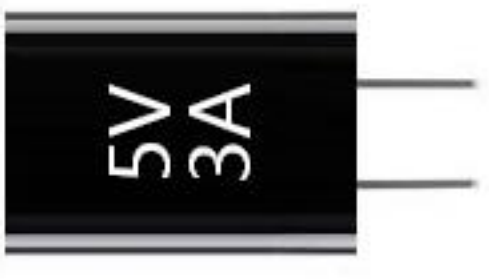
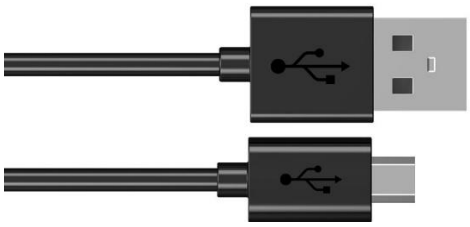
Chapter 0 Raspberry Pi Preparation

Install a System

Firstly, install a system for your RPi.

Component List

Required Components

Raspberry Pi 4B / 3B+/ 3B /3A+ (Recommended) 	5V/3A Power Adapter. Different versions of Raspberry Pi have different power requirements. 
Micro USB Cable x1 	Micro SD Card (TF Card) x1, Card Reader x1 

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

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

Power requirement of different versions of Raspberry Pi is shown in following table:

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

For more details, please refer to https://www.raspberrypi.org/help/faqs/#powerReqs

In addition, RPi also needs a network cable used to connect it to wide area network.

All of these components are necessary. Among them, the power supply is required at least 5V/2.5A, because lack of power supply will lead to many abnormal problems, even damage to your RPi. So power supply with 5V/2.5A is highly recommend. SD Card Micro (recommended capacity 16GB or more) is a hard drive for RPi, which is used to store the system and personal files. In later projects, the components list with a RPi will contains these required components, using only RPi as a representative rather than presenting details.

Optional Components

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

Required Accessories for Monitor

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

1.Display with HDMI interface

2.Mouse and Keyboard with USB interface

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

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

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

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

Required Accessories for Remote Desktop

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

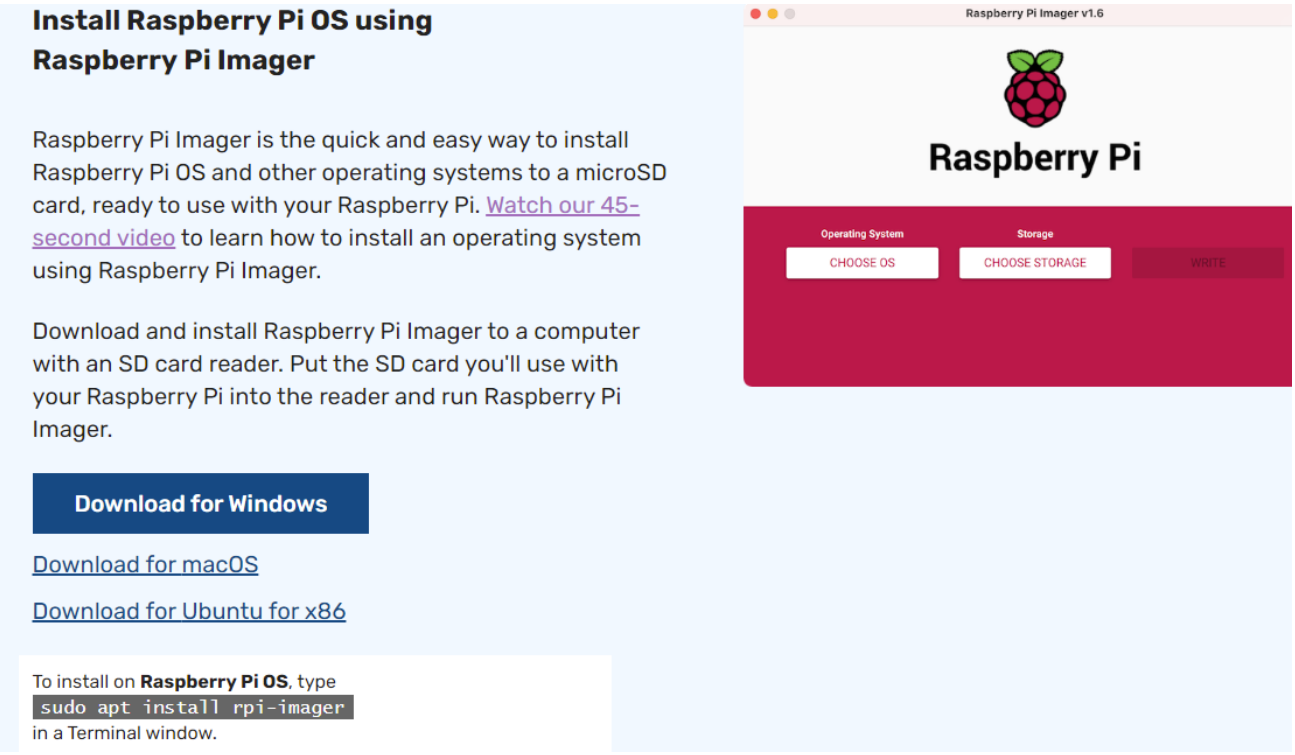
Item	Pi Zero	Pi Zero W	Pi A+	Pi 3A+	Pi B+/2B	Pi 3B/3B+/4B
Micro-USB to USB-A OTG cable	Yes	Yes	No	NO		
USB transferring to Ethernet interface	Yes	Yes	Yes			

Raspberry Pi OS

Install imager tool

Visit this website to install imager tool.

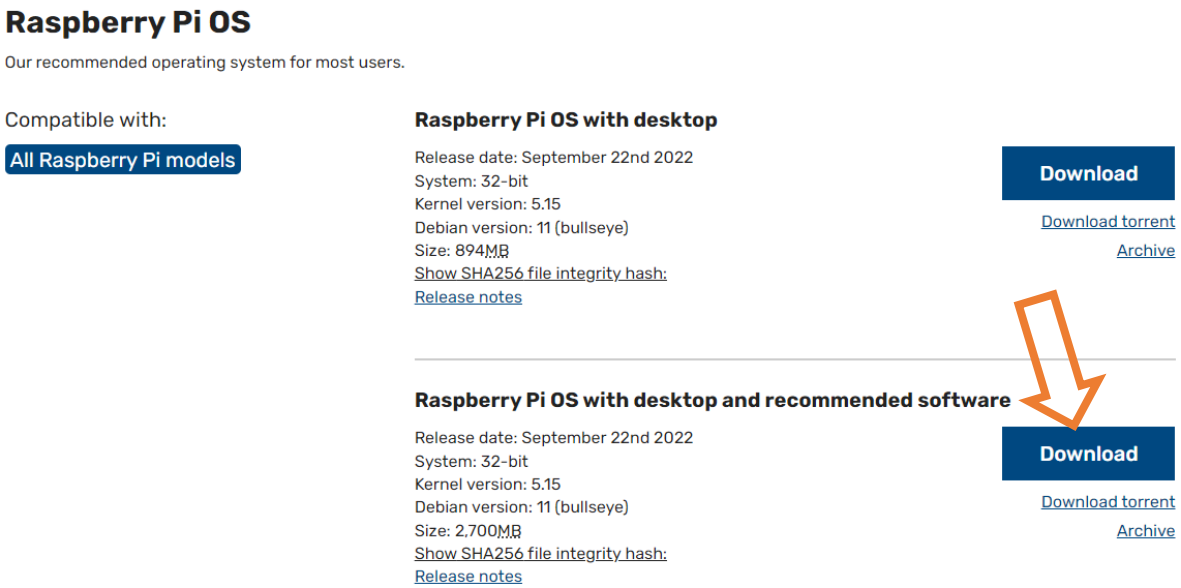
https://www.raspberrypi.com/software/



Download OS file

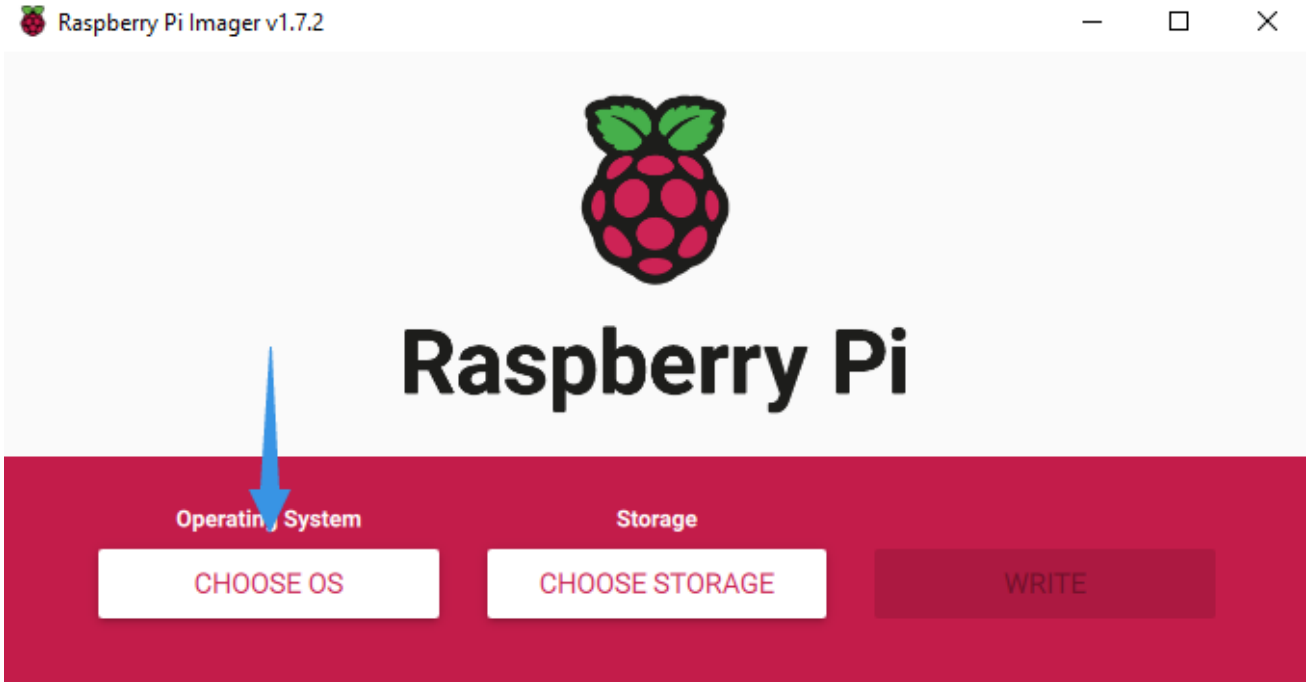
Visit following website to download the OS file.

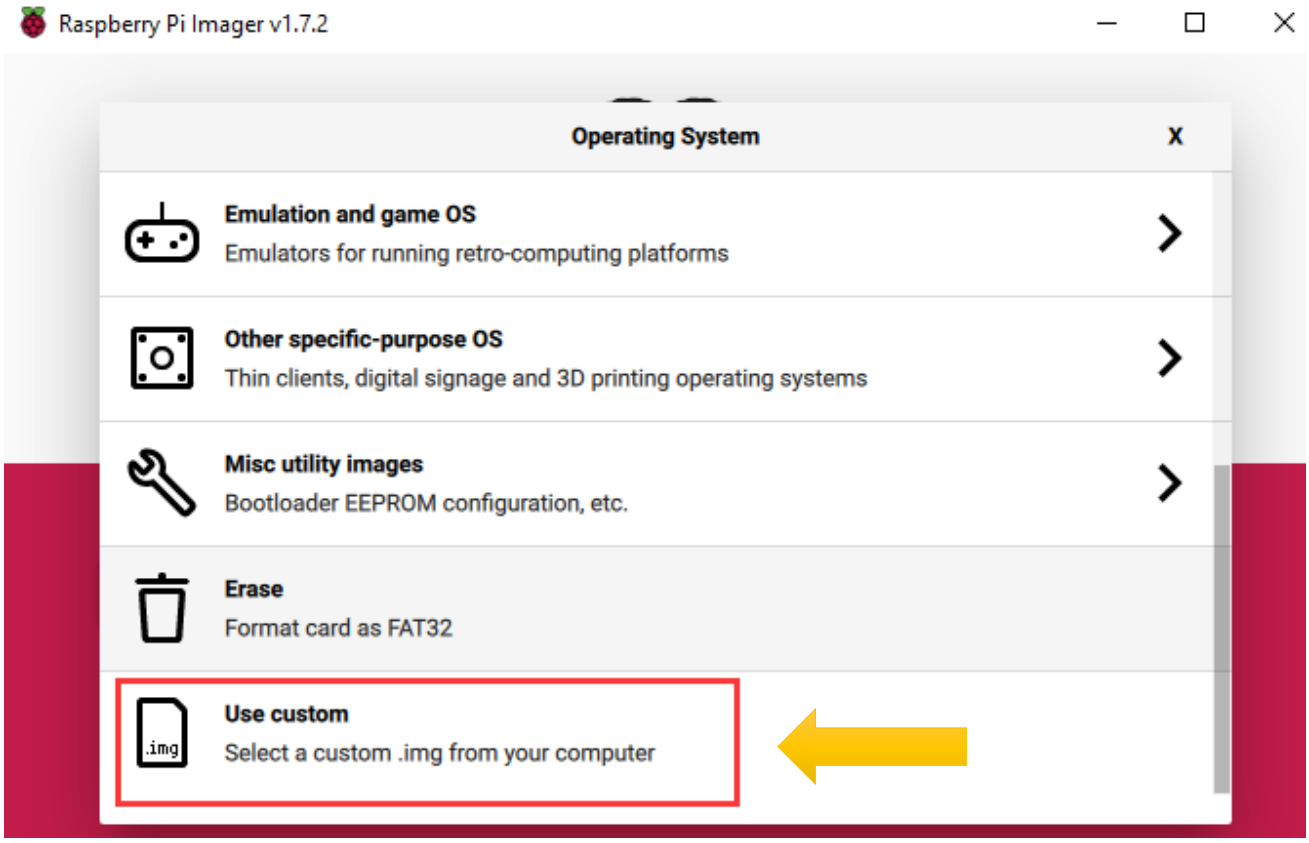
https://www.raspberrypi.com/software/operating-systems/



Write System to Micro SD Card

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





Choose the SD card.

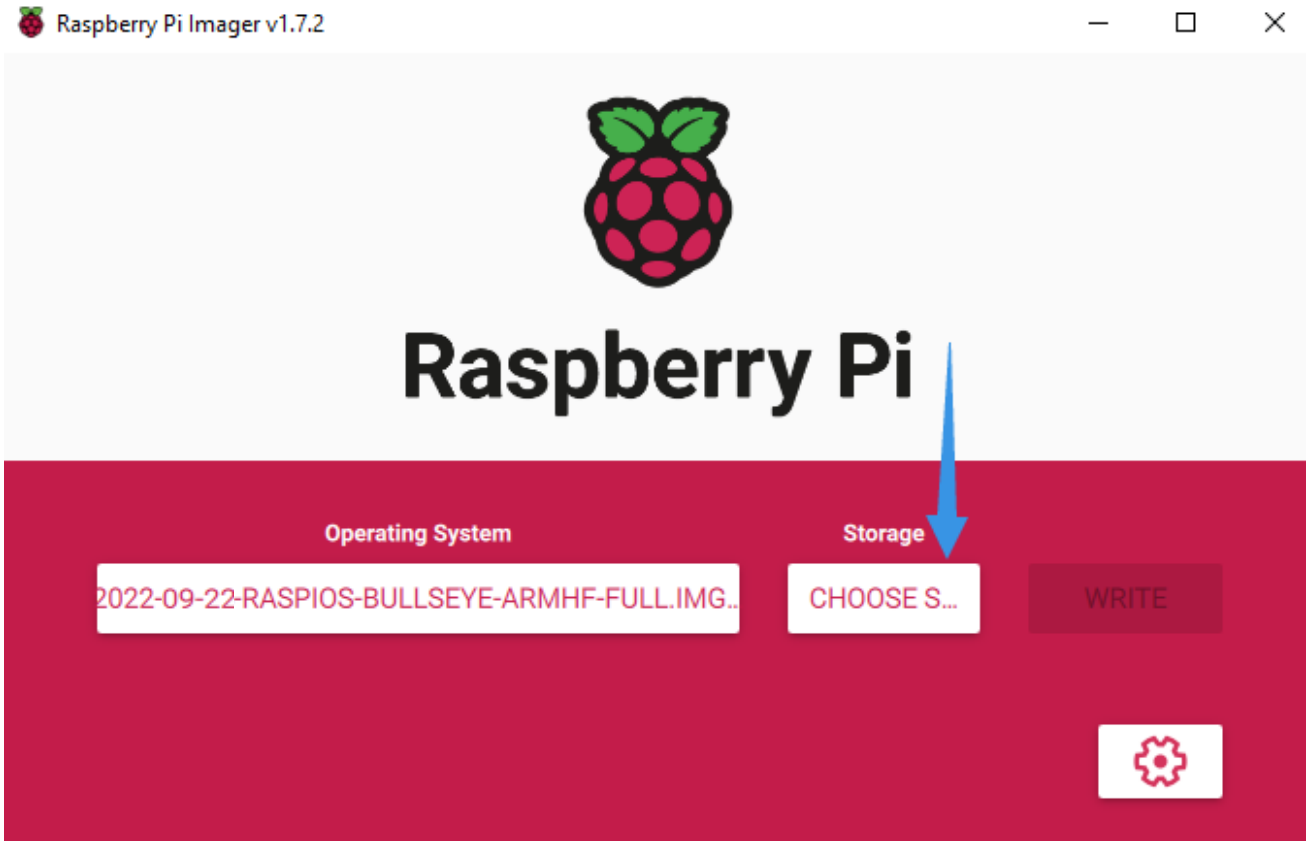
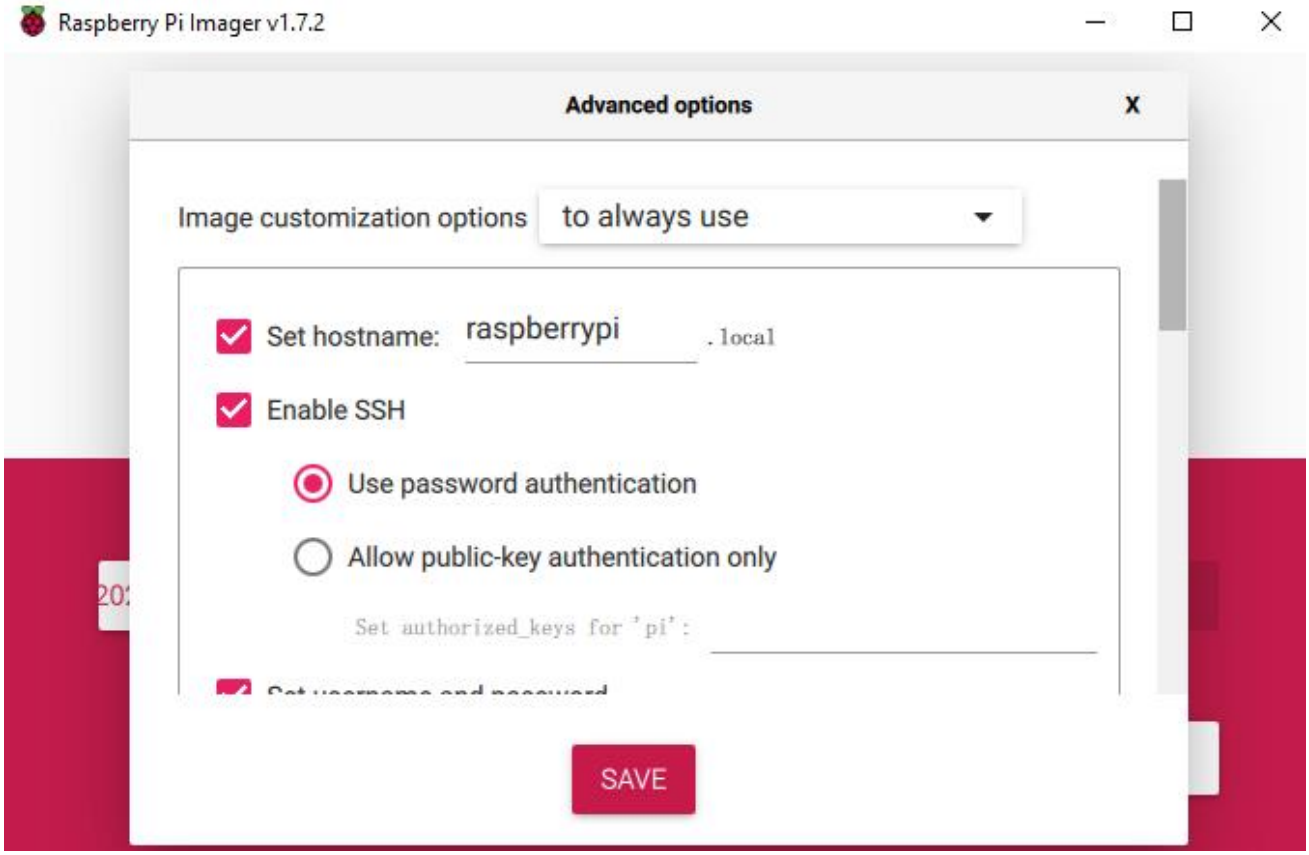
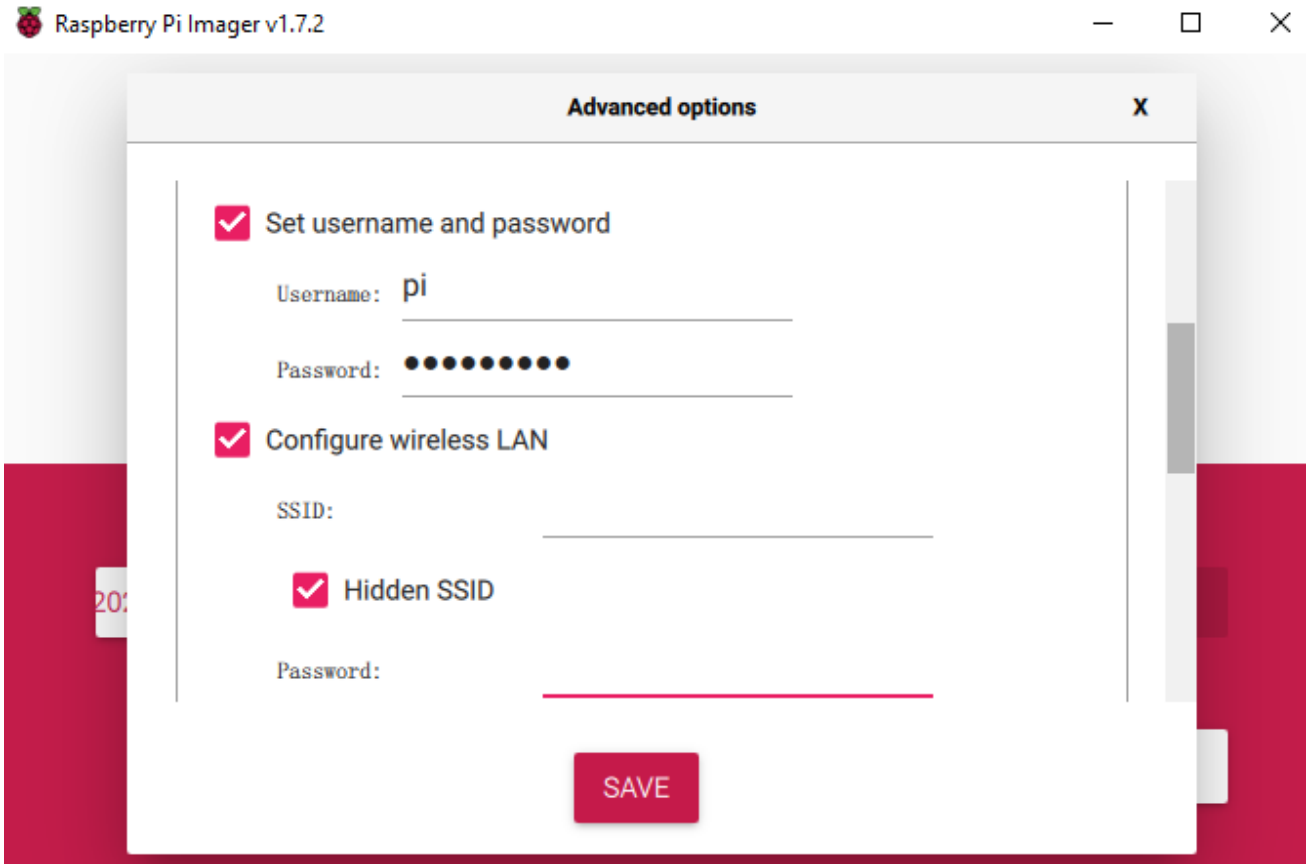


Image option.

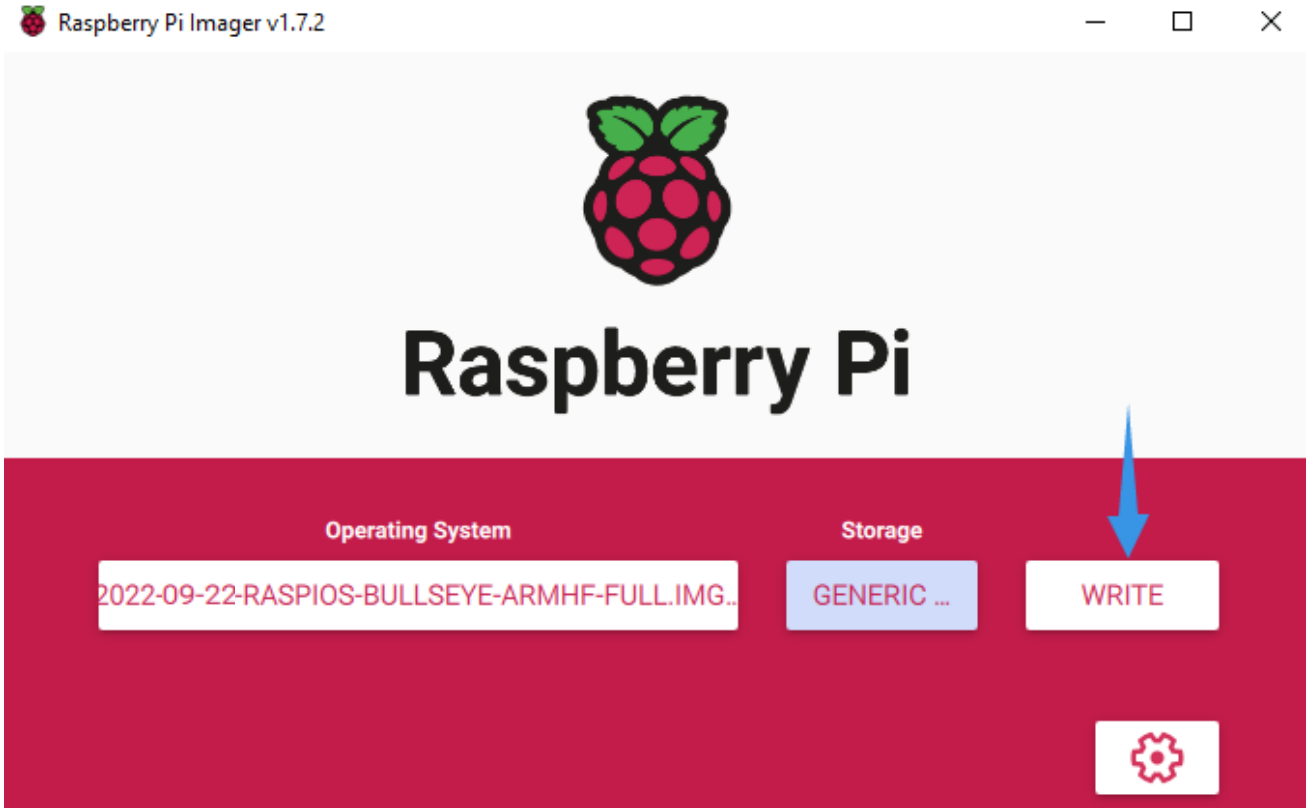
Enable SSH.



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



Finally WRITE.

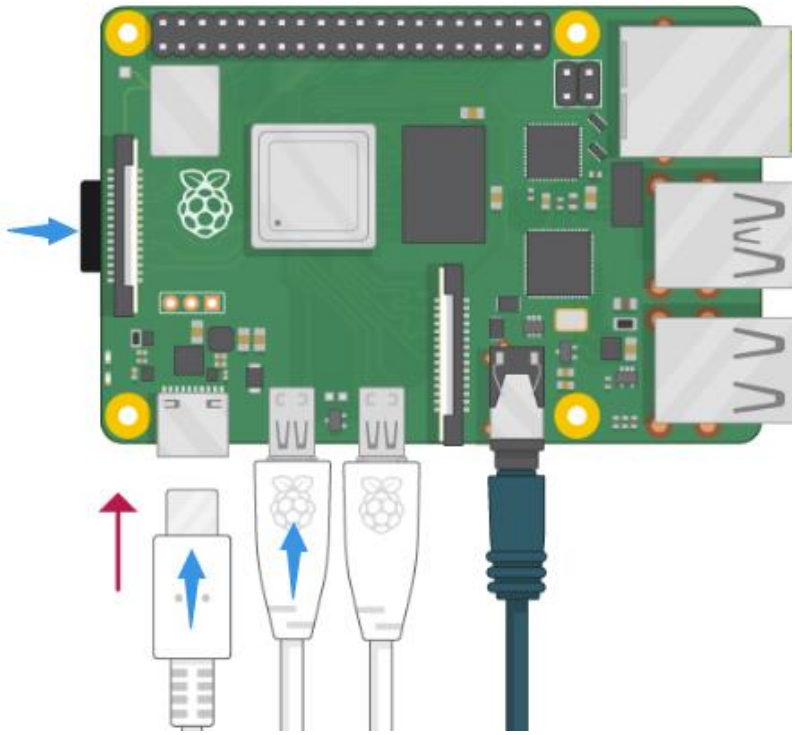


Start Raspberry Pi

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

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

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



Later, after setup, you will need to enter your user name and password to login. The default user name: pi; password: raspberry. After login, you should see the following screen.

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

Now you can skip to VNC Viewer.

Remote desktop & VNC

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

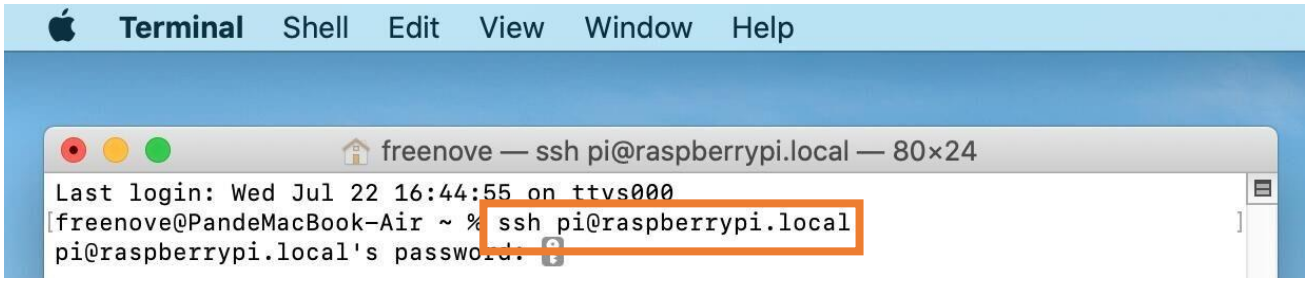
If you don't have a spare monitor, mouse and keyboard for your RPi, you can use a remote desktop to share a display, keyboard, and mouse with your PC. Below is how to use:
MAC OS remote desktop and Windows OS remote desktop.

MAC OS Remote Desktop

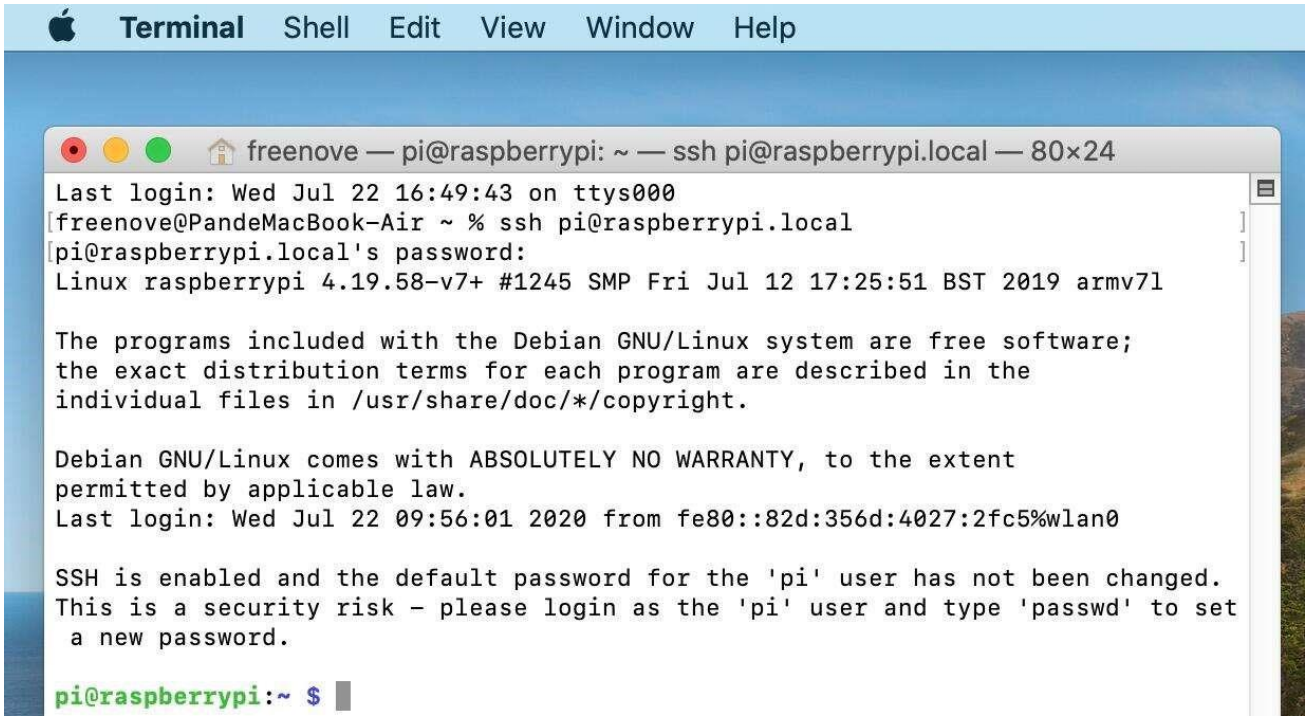
Open the terminal and type following command. **If this command doesn't work, please move to next page.**

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

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



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



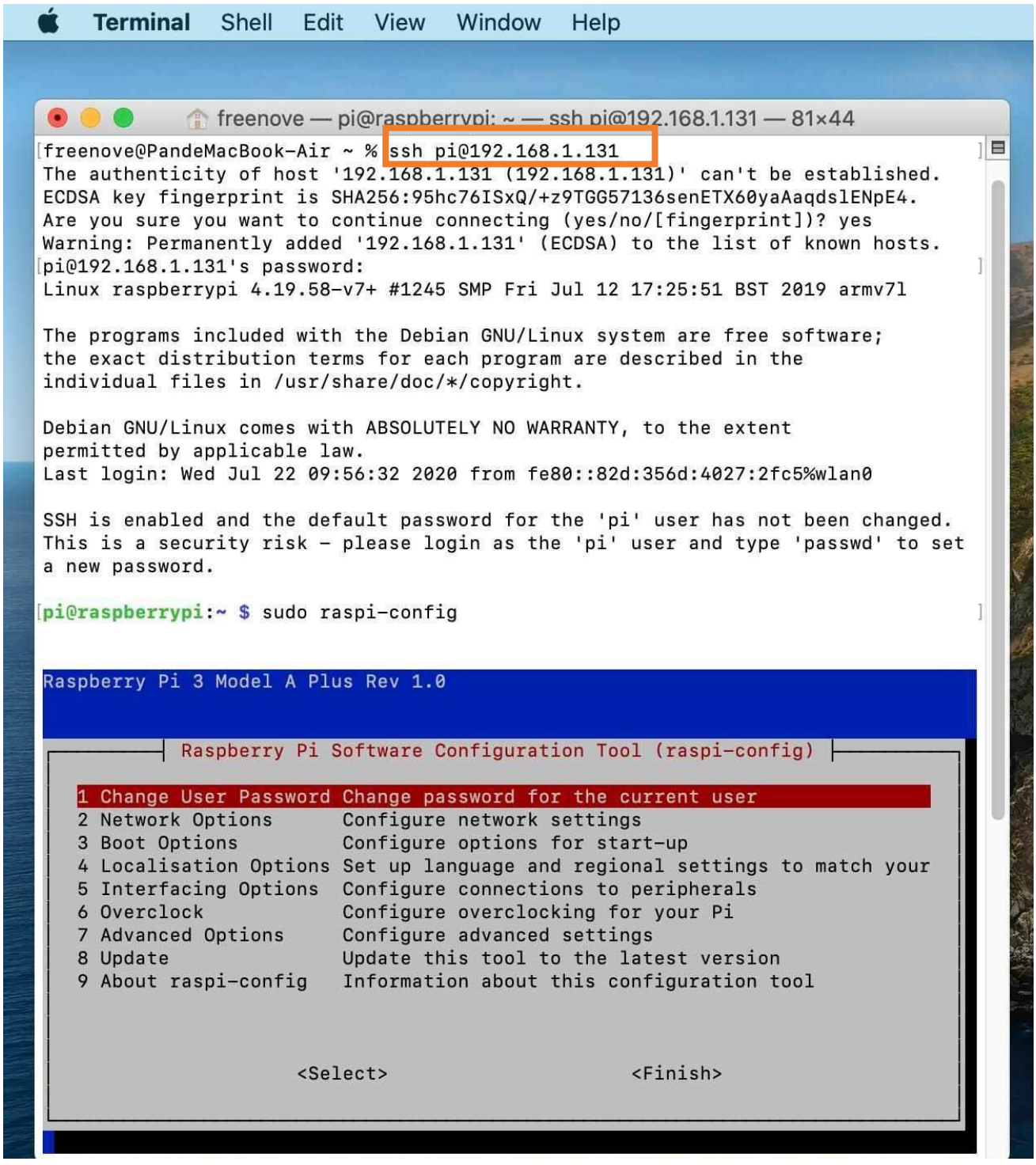
You can also use the IP address to log in Pi.

Enter **router** client to **inquiry IP address** named "raspberry pi". For example, I have inquired to **my RPi IP address, and it is "192.168.1.131".**

Open the terminal and type following command.

```
ssh pi@192.168.1.131
```

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



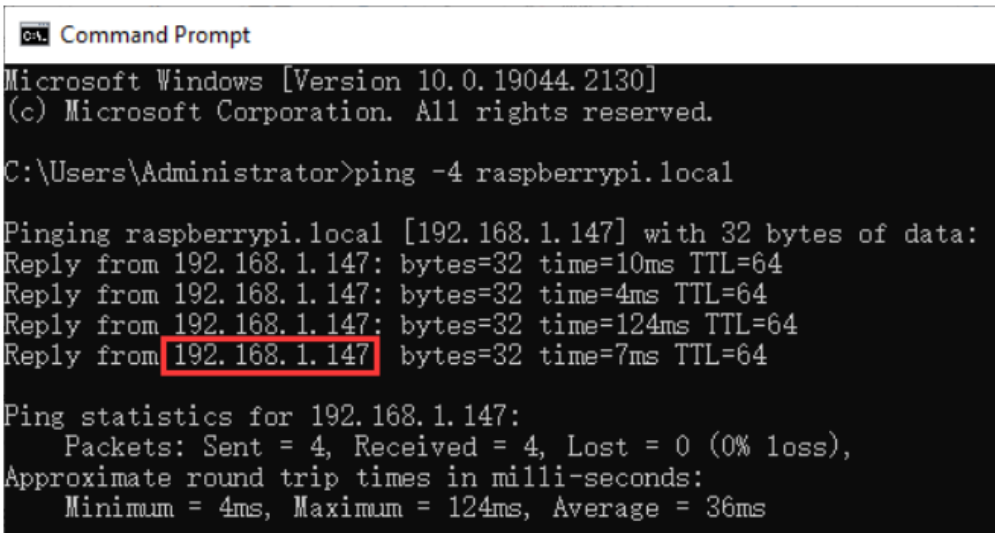
Then you can skip to VNC Viewer.

Windows OS Remote Desktop

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

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

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



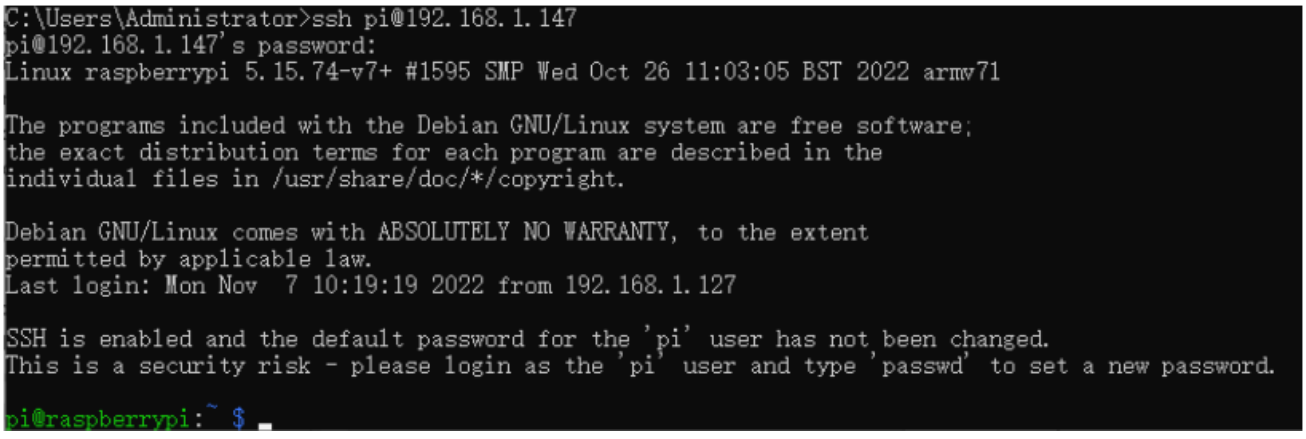
Then 192.168.1.147 is my Raspberry Pi IP.

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

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

Enter the following command:

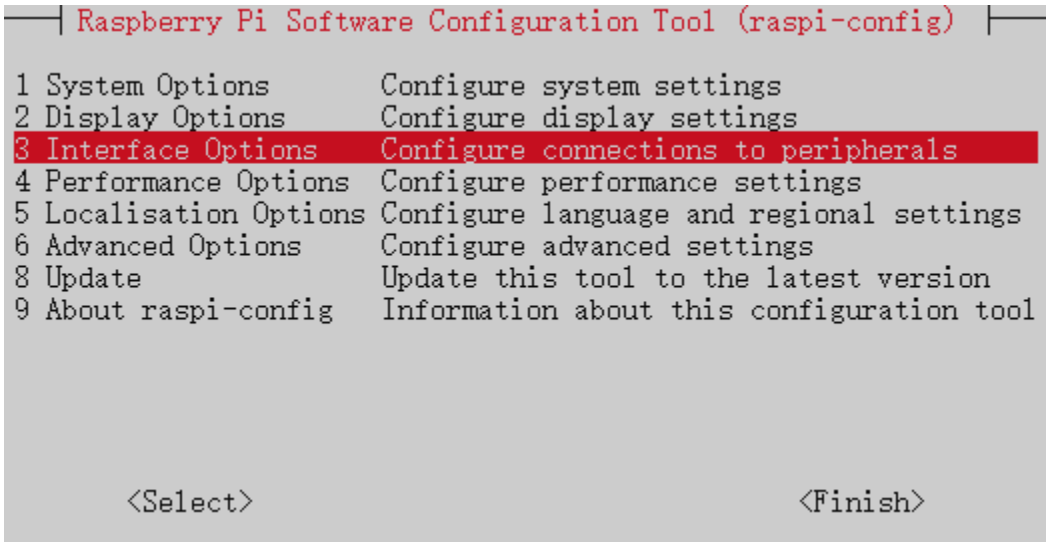
```
ssh pi@192.168.1.147
```

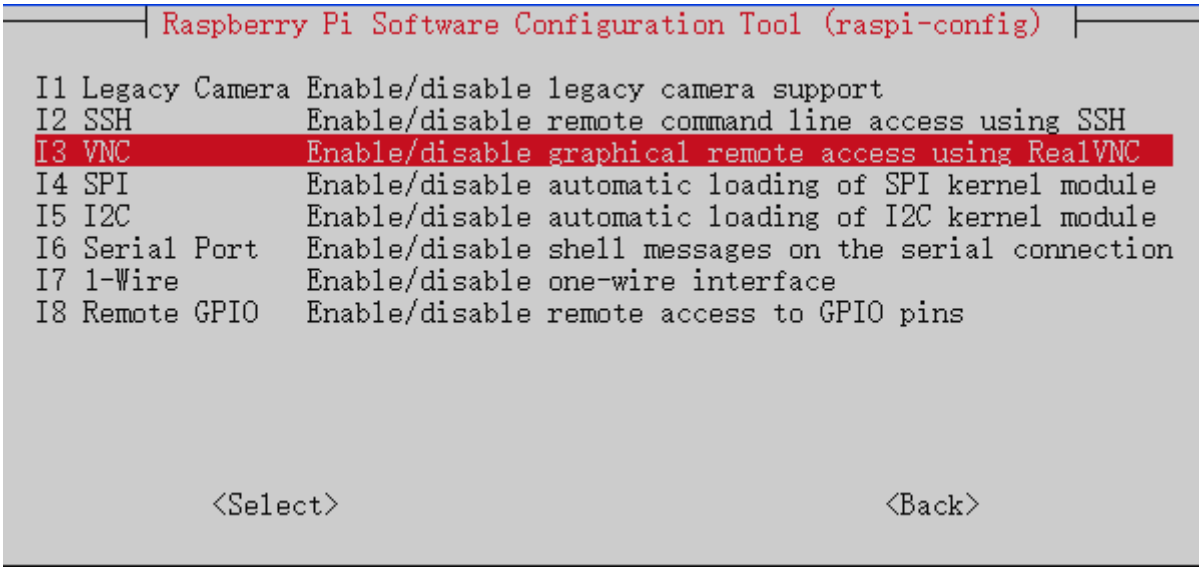


VNC Viewer & VNC

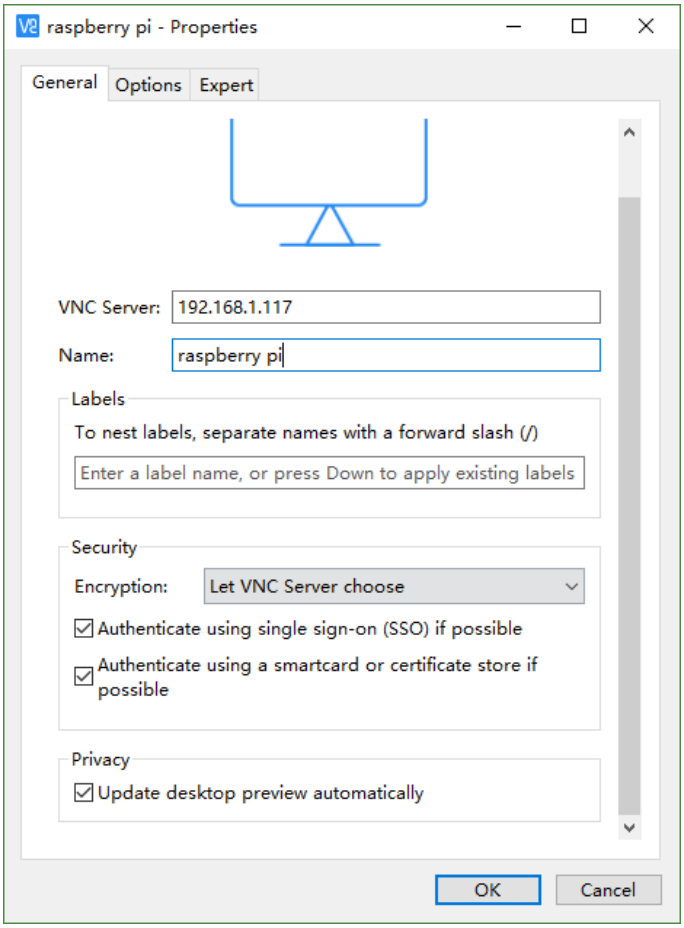
Type the following command. And select Interfacing **Options→ VNC →Yes→OK→Finish**. Here Raspberry Pi may need to be restarted, and choose ok. Then open VNC interface.

```
sudo raspi-config
```

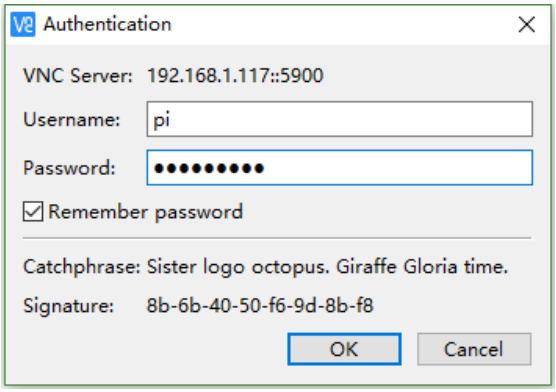




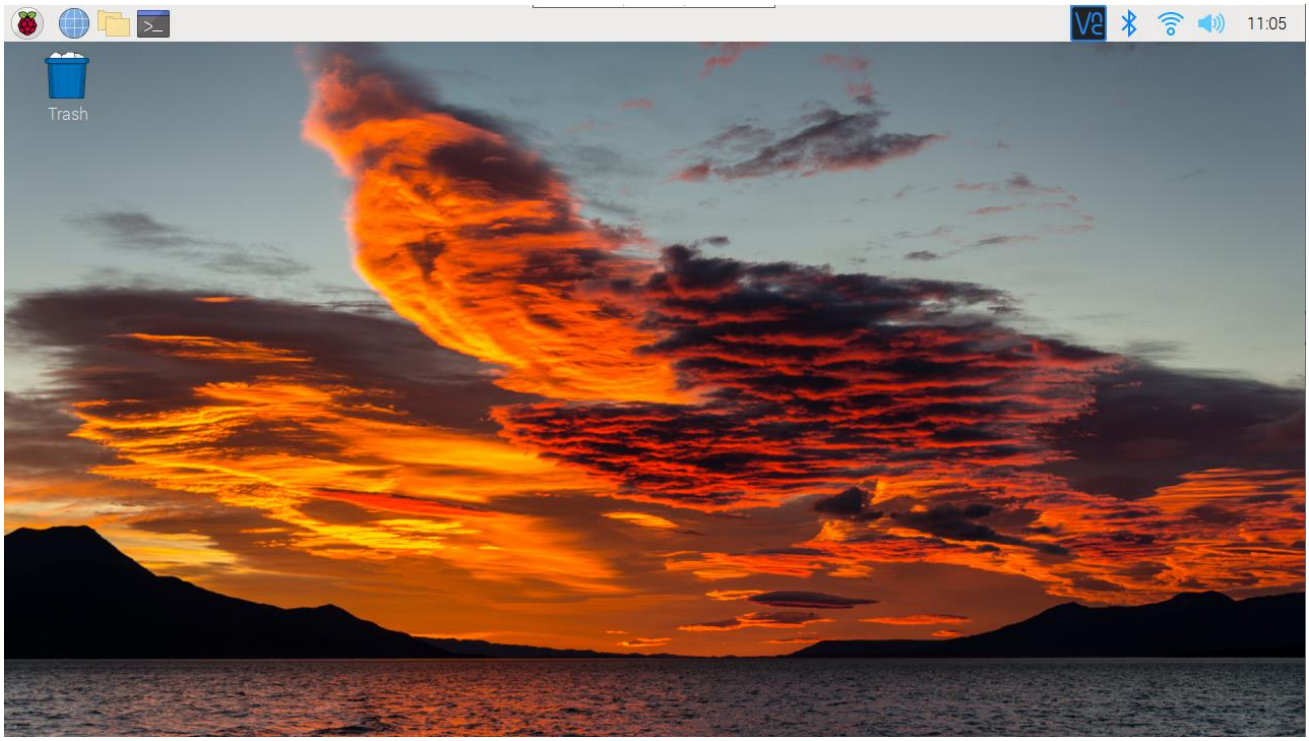
Then download and install VNC Viewer according to your computer system by clicking following link:
https://www.realvnc.com/en/connect/download/viewer/
After installation is completed, open VNC Viewer. And click File → New Connection. Then the interface is shown below.



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



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

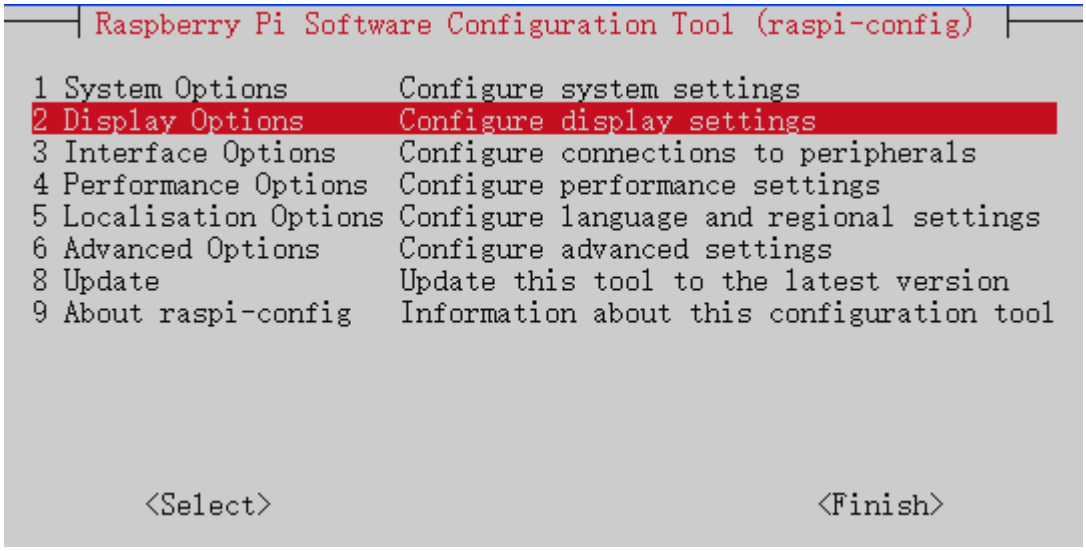


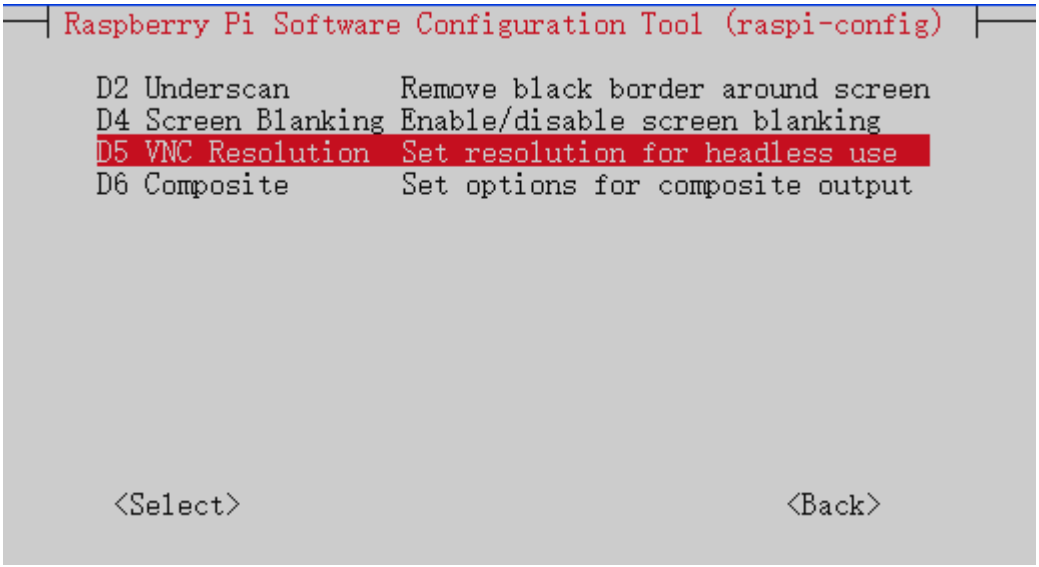
Here, you have logged in to Raspberry Pi successfully by using VNC Viewer

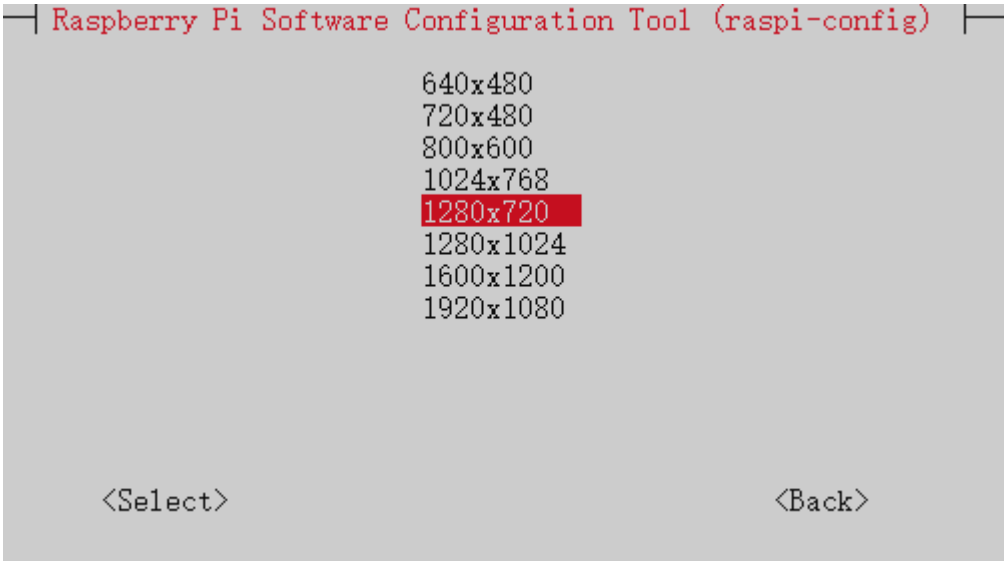
If the resolution ratio is not great or there is just a **little window**, you can set a proper resolution ratio via steps below.

```
sudo raspi-config
```

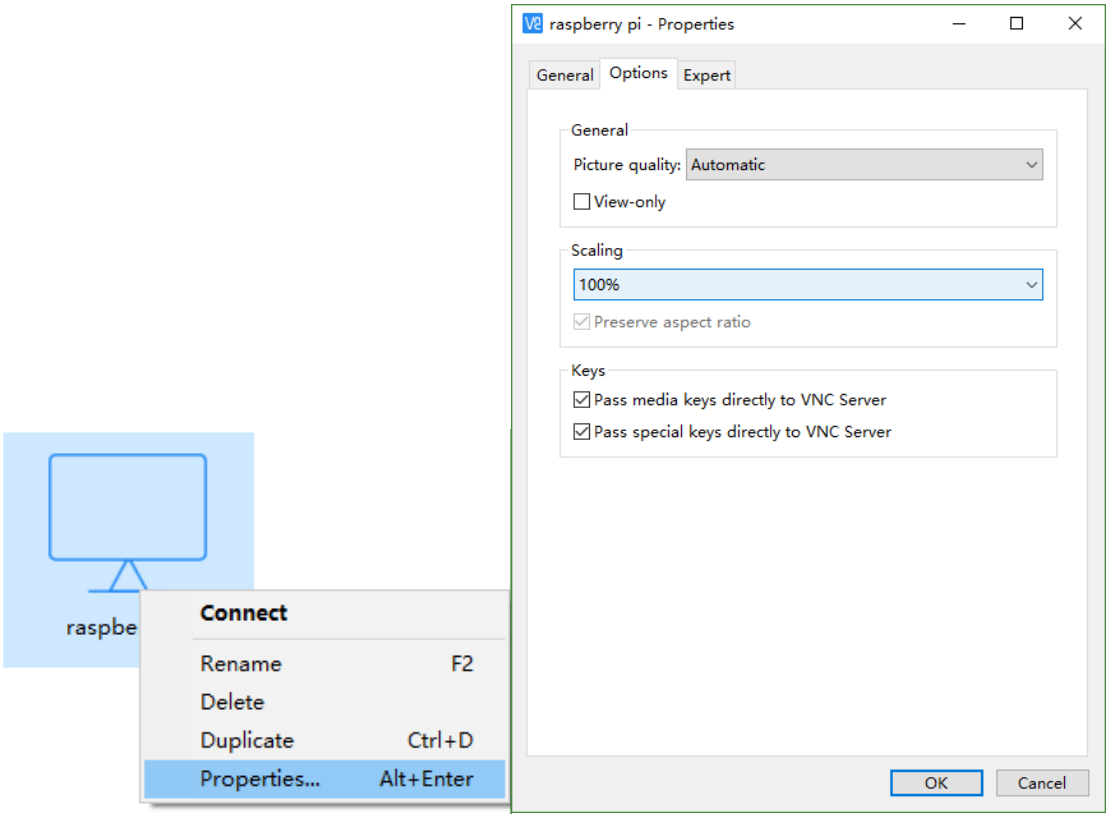
Select **Display Options→VNV Resolution→Proper resolution ratio (set by yourself)→OK→Finish→Yes**. And then reboot Raspberry Pi.







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



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

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

Chapter 1 Software installation and Test (necessary)

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

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

Note:

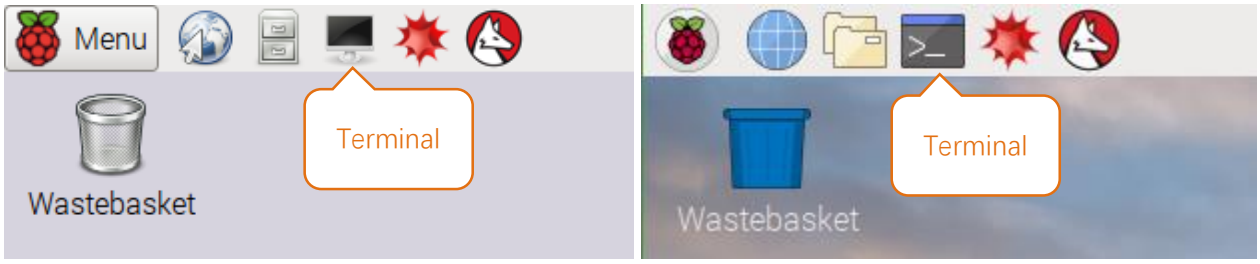
1, **Please use Raspberry Pi OS with Desktop**

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

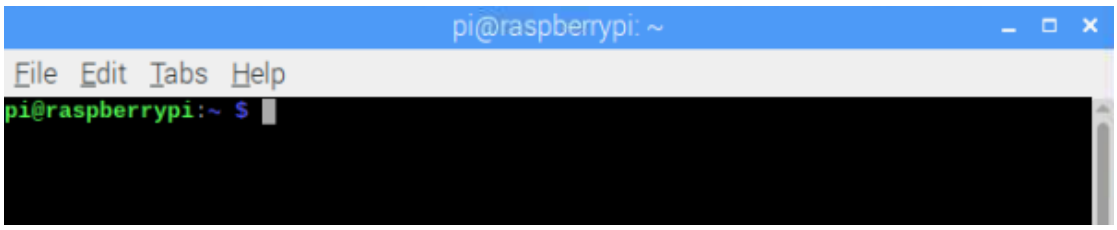
3, If you are using **remote desktop** to login Raspberry Pi, you need to use VNC viewer.

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 "CTRL + ALT + T" on the desktop.

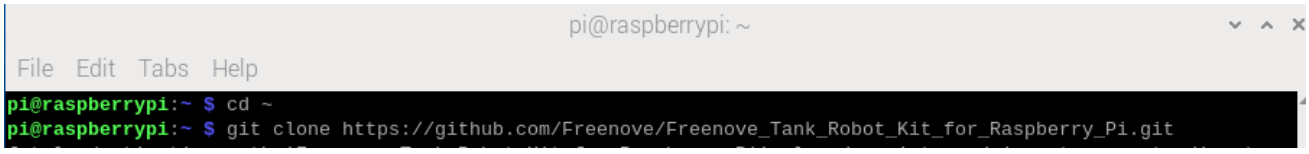


The terminal is shown below:



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

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



Downloading takes some time. Please wait with patience.

You can also find and download the code by visiting our official website (http://www.freenove.com) or our GitHub repository (https://github.com/freenove).
Please note that this tutorial is based on python3. Please check if the default pyton is python3 before using it.
If you have never learned python before, you can learn some basics through the following link:
https://python.swaroopch.com/basics.html

Set Python3 as default python (Necessary)

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

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

If it is python3, you can skip this section.

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

1. Enter directory /usr/bin

```
cd /usr/bin
```

2. Delete the originalpython link.

```
sudo rm python
```

3. Create new python links to python.

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

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

```
python
```

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

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

Shortcut Key

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

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

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

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

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

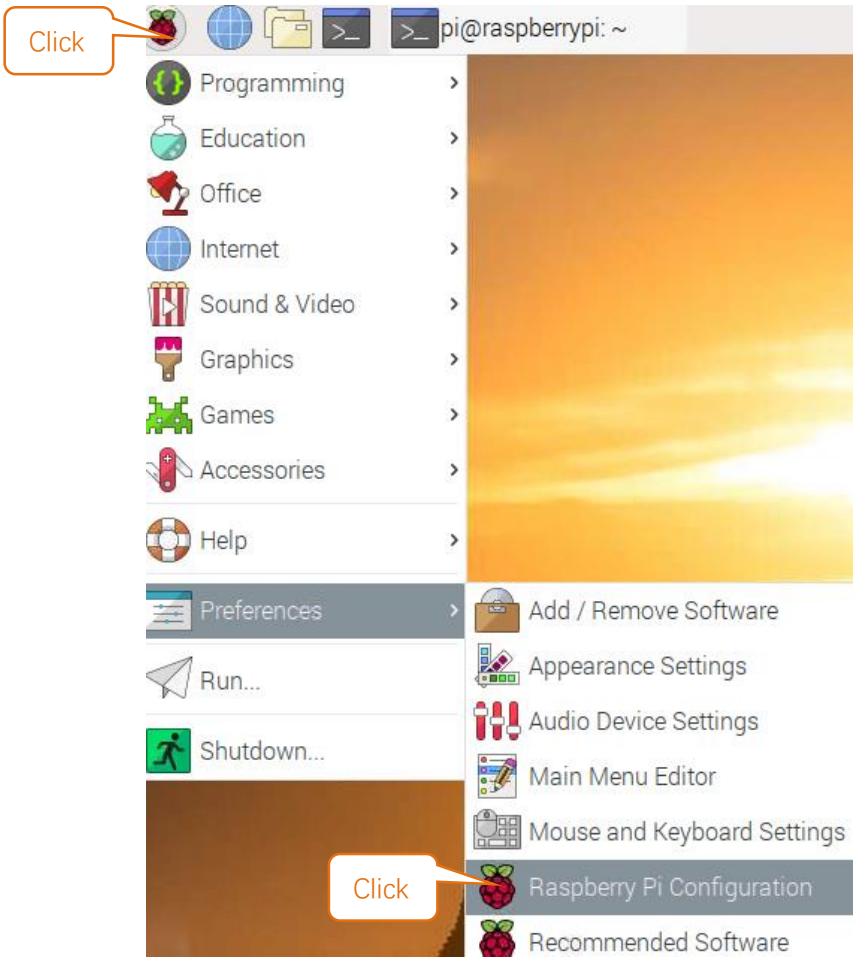
```
pi@raspberrypi:~ $ cd D
Desktop/   Documents/ Downloads/
pi@raspberrypi:~ $ cd Doc
```

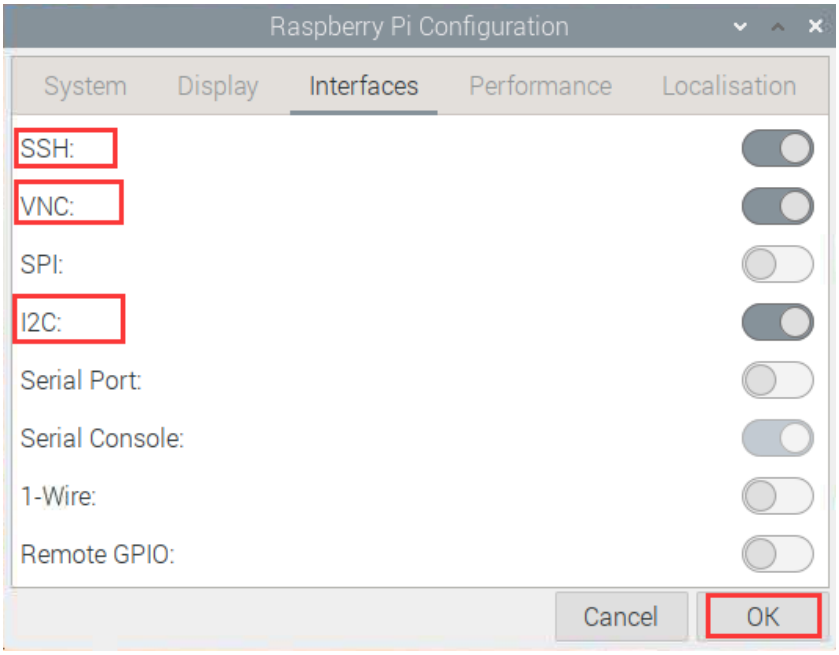
```
pi@raspberrypi:~ $ cd D
Desktop/   Documents/ Downloads/
pi@raspberrypi:~ $ cd Documents/
```

Step 2 Configuration

Enable VNC

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





Additional supplement

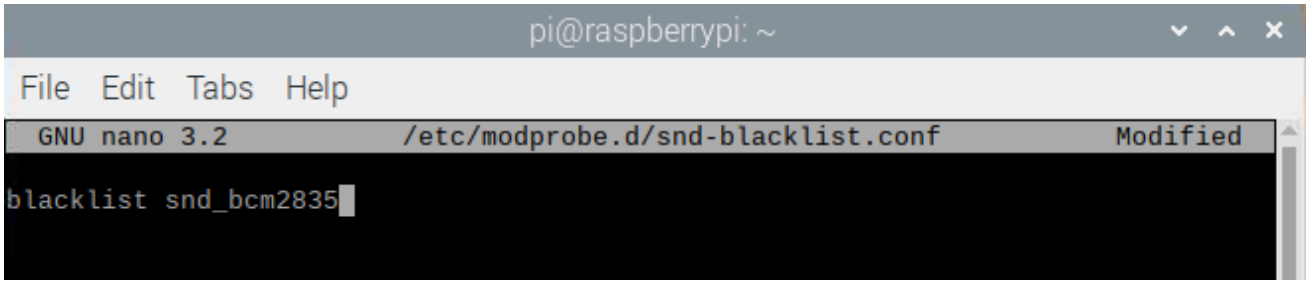
Raspbery Pi, other than 4B and 400, needs to disable the audio module, othewise the LED will not work properly.

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

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

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

```
blacklist snd_bcm2835
```



2. We also need to edit config file.

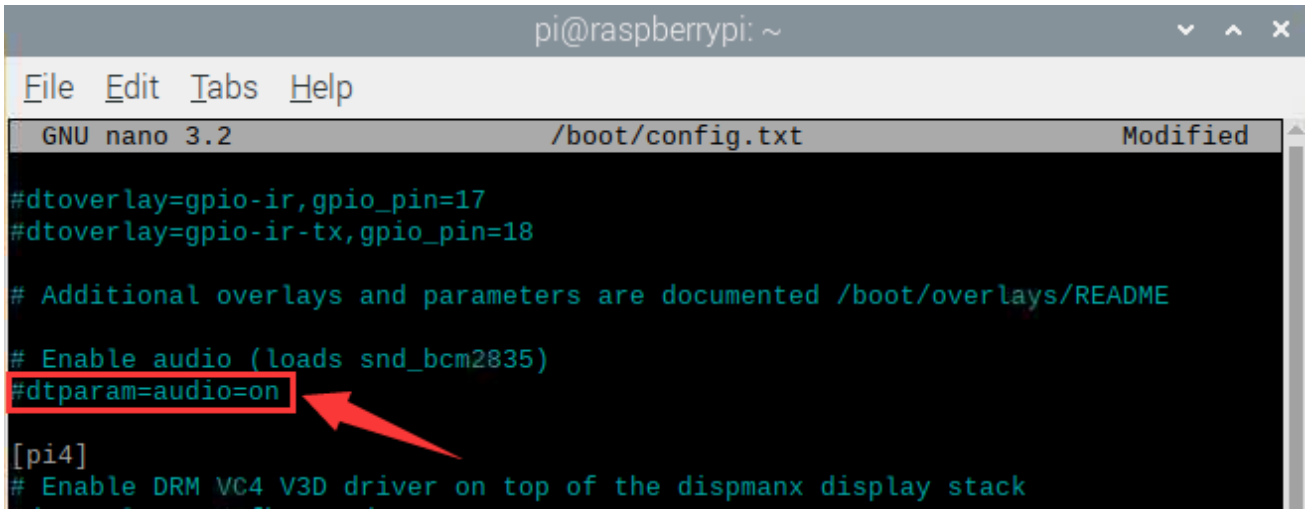
```
sudo nano /boot/config.txt
```

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

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

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

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



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

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

If your Raspberry PI cannot use the LED light or the LED light displays abnormally, please check whether the above configuration exists.

Step 3 Run the Libraries Installation Program

All the commands are based on python3. If the default python is python2, please refer to the Step1 to set python3 to default python.

1. Execute following commands to enter directory of "setup.py".

```
cd ~/Freenove_Tank_Robot_Kit_for_Raspberry_Pi/Code
```

2. Run setup.py

```
sudo python setup.py
```

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

```
Now the installation is successful.

Please reboot raspberry pi, 'sudo reboot'
```

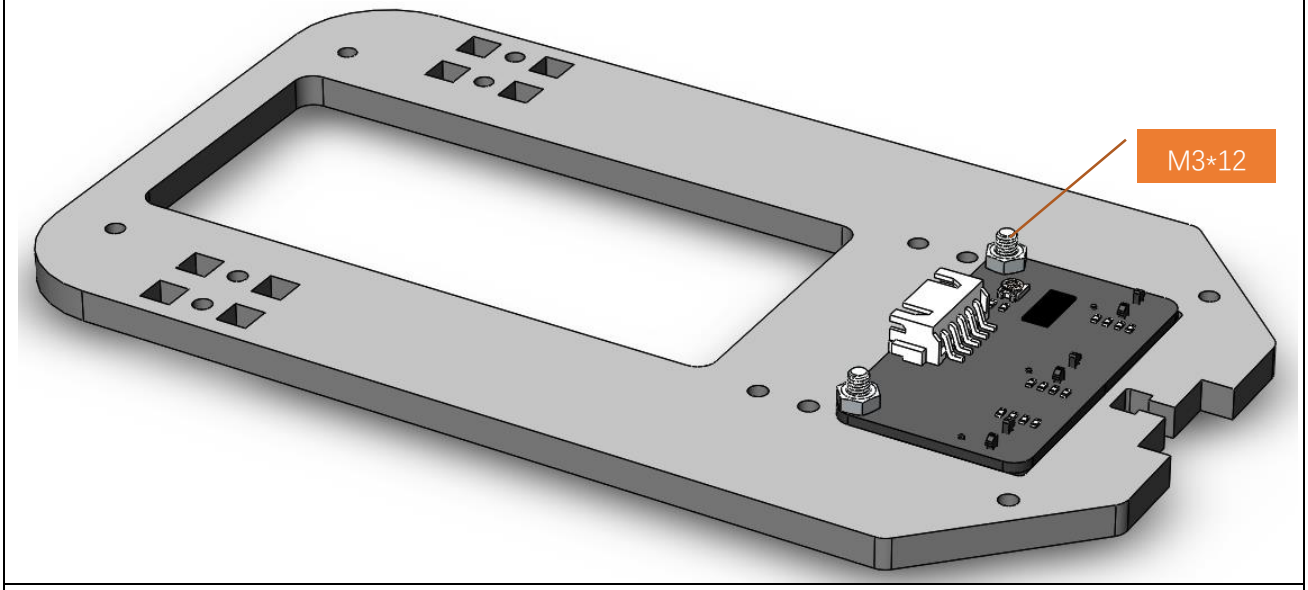
If the installation fails, please rerun setup.py. After the installation is completed, restart the Raspberry Pi. Most installation failures are caused by network reasons.

```
sudo python setup.py
```

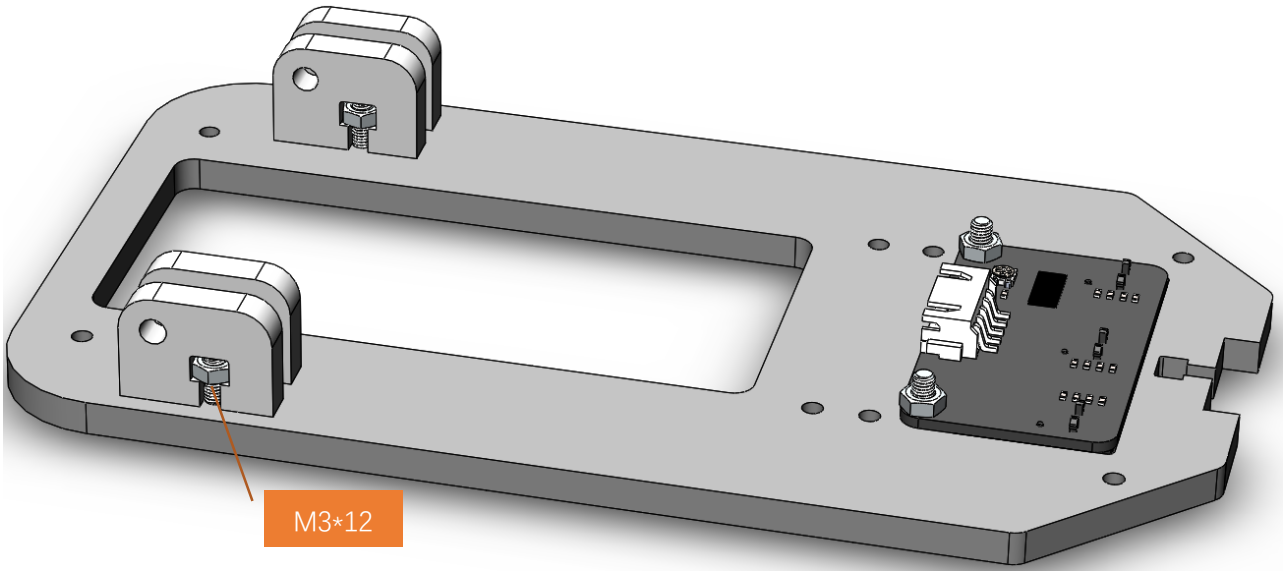
Chapter 2 Assemble Smart Car

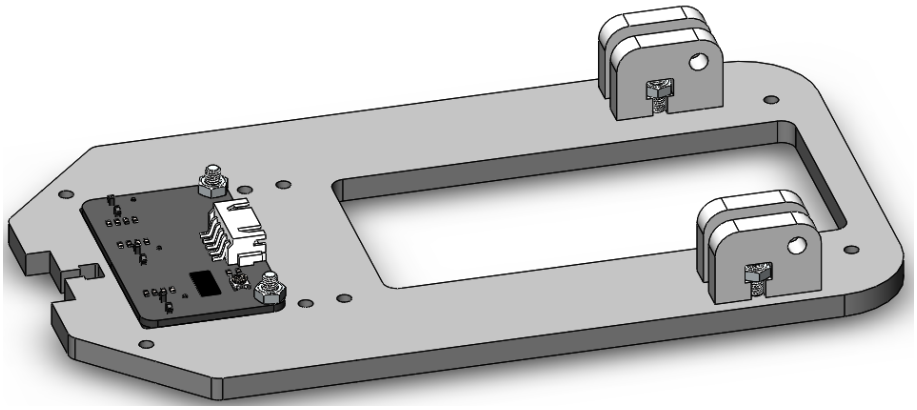
If you have any concerns, please feel free to contact us via support@freenove.com

Step 1 Install line tracking sensor.

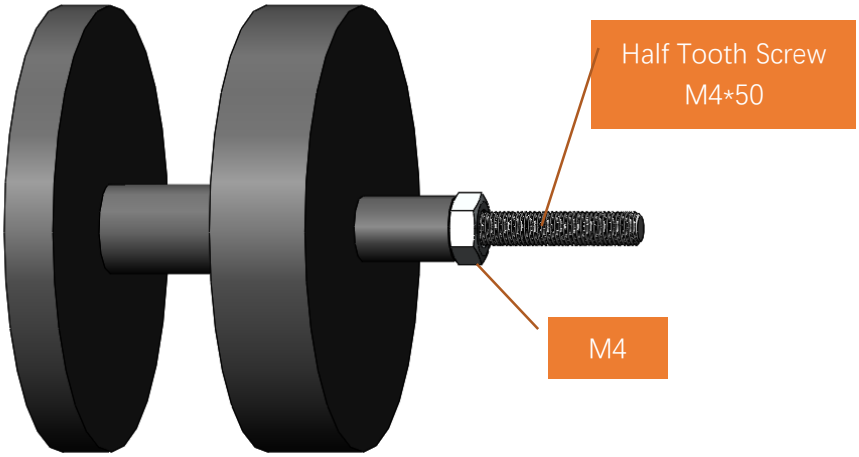


Step 2

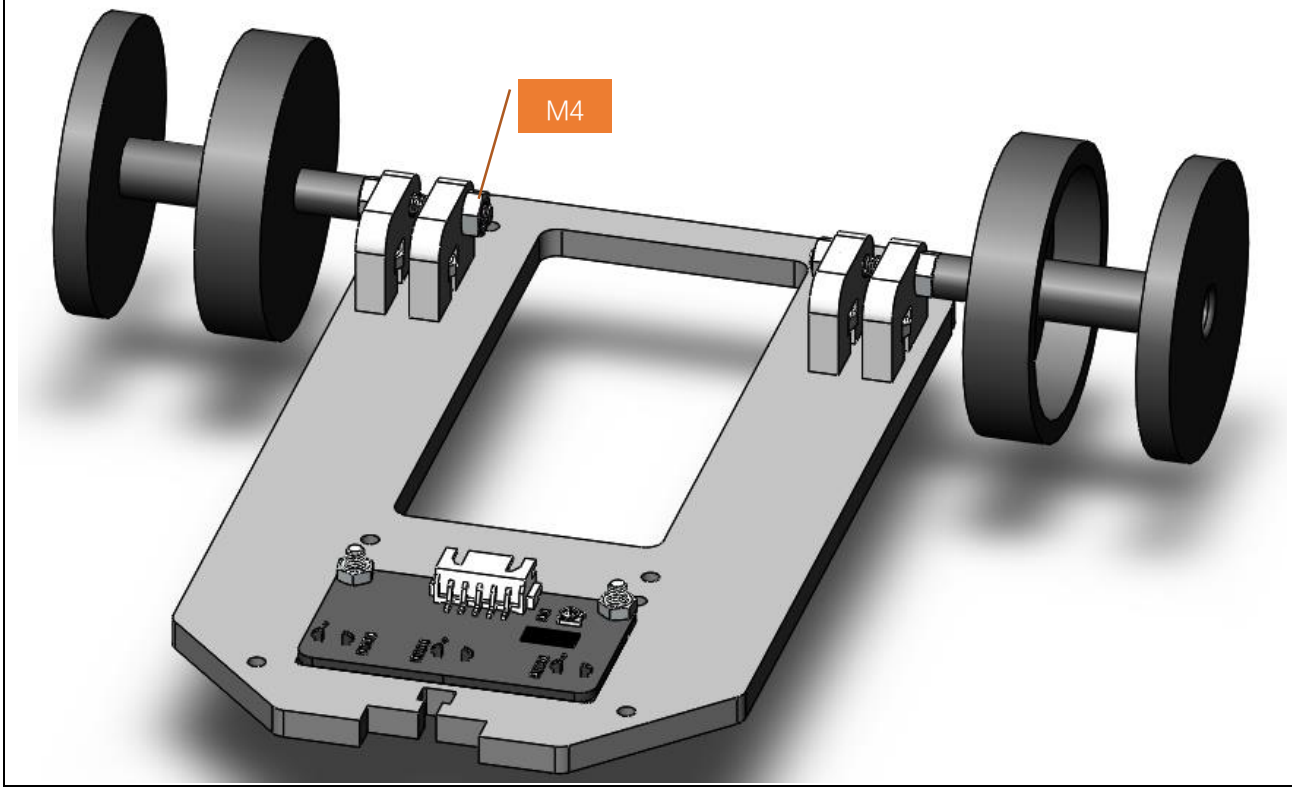




Step 3

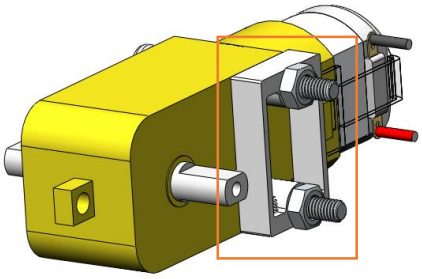


Step 4

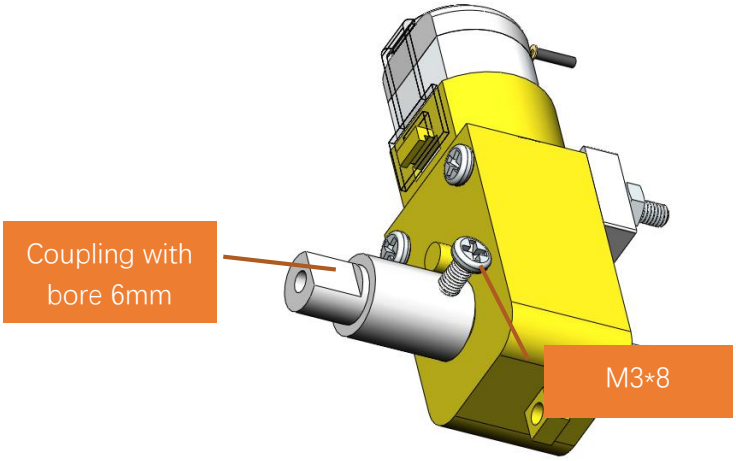


Step 5

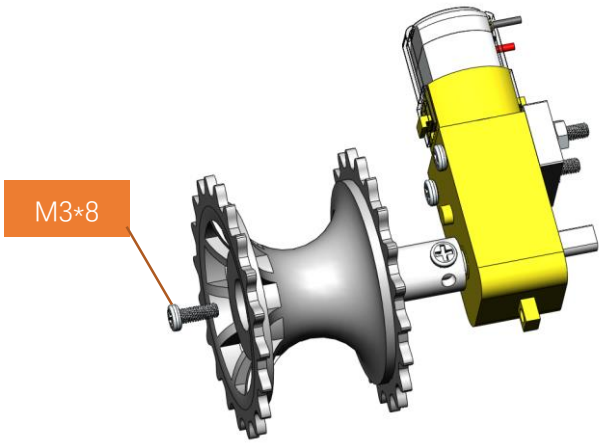
There is a special fixed bracket to fix motor, which contains an aluminum bracket, two M3*30 screws, two M3*8 screws, and two M3 nuts, as shown below:



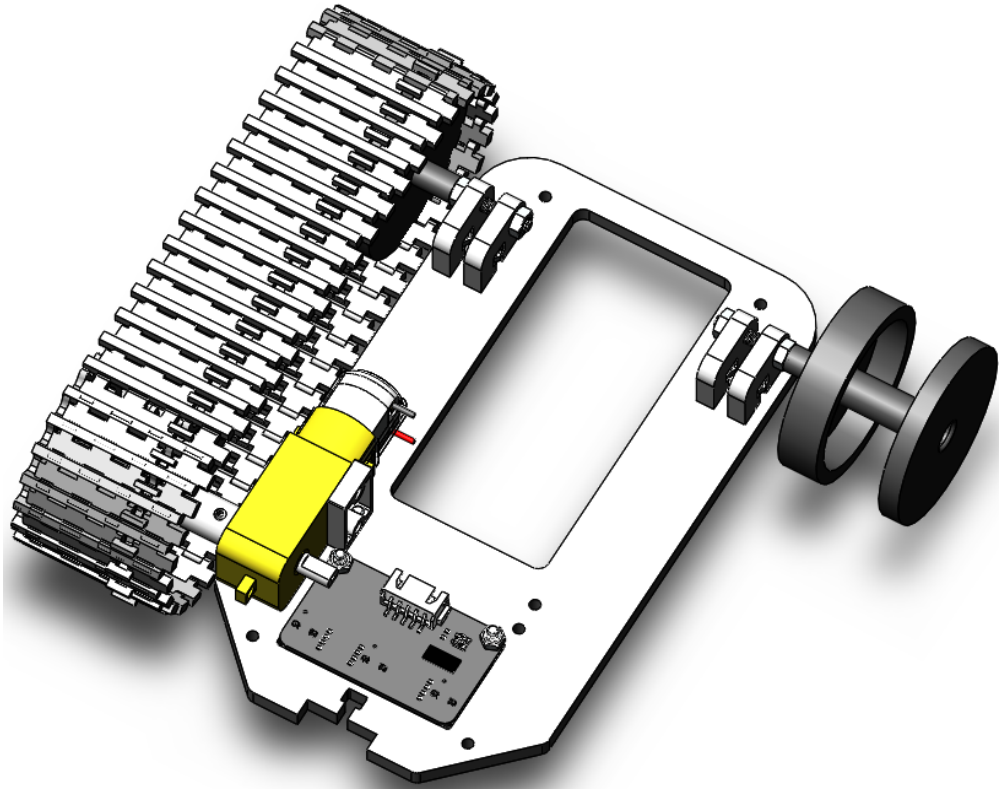
Step 6 Connect coupling with motor.

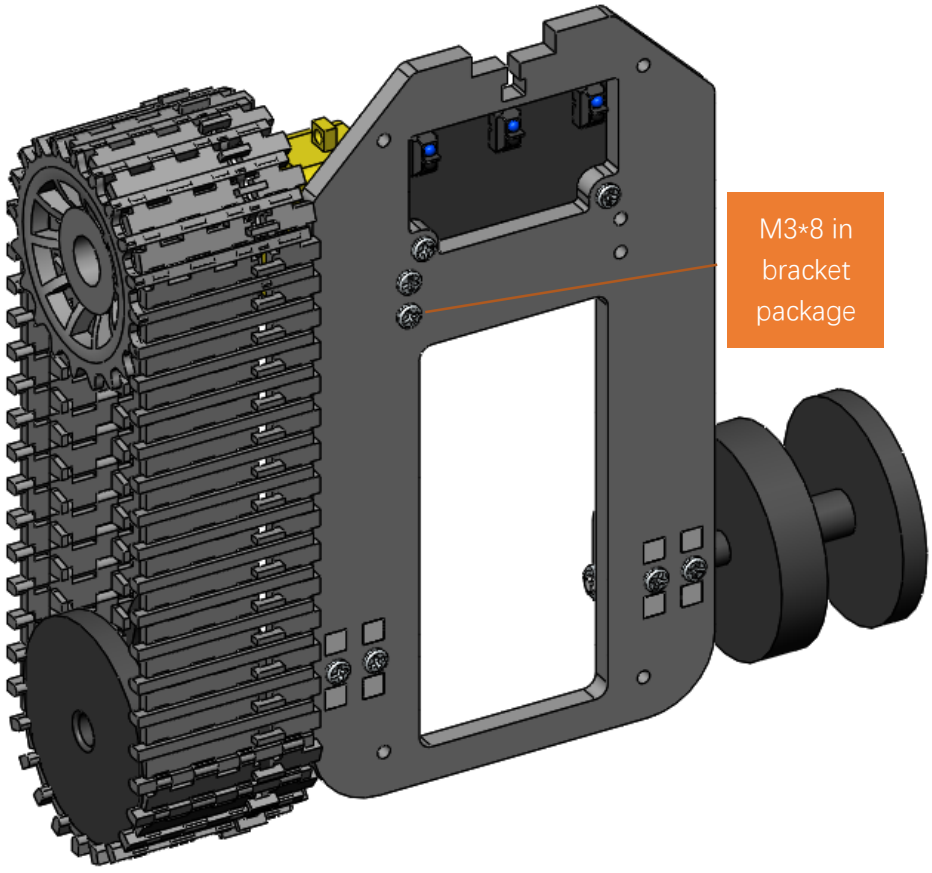


Step 7

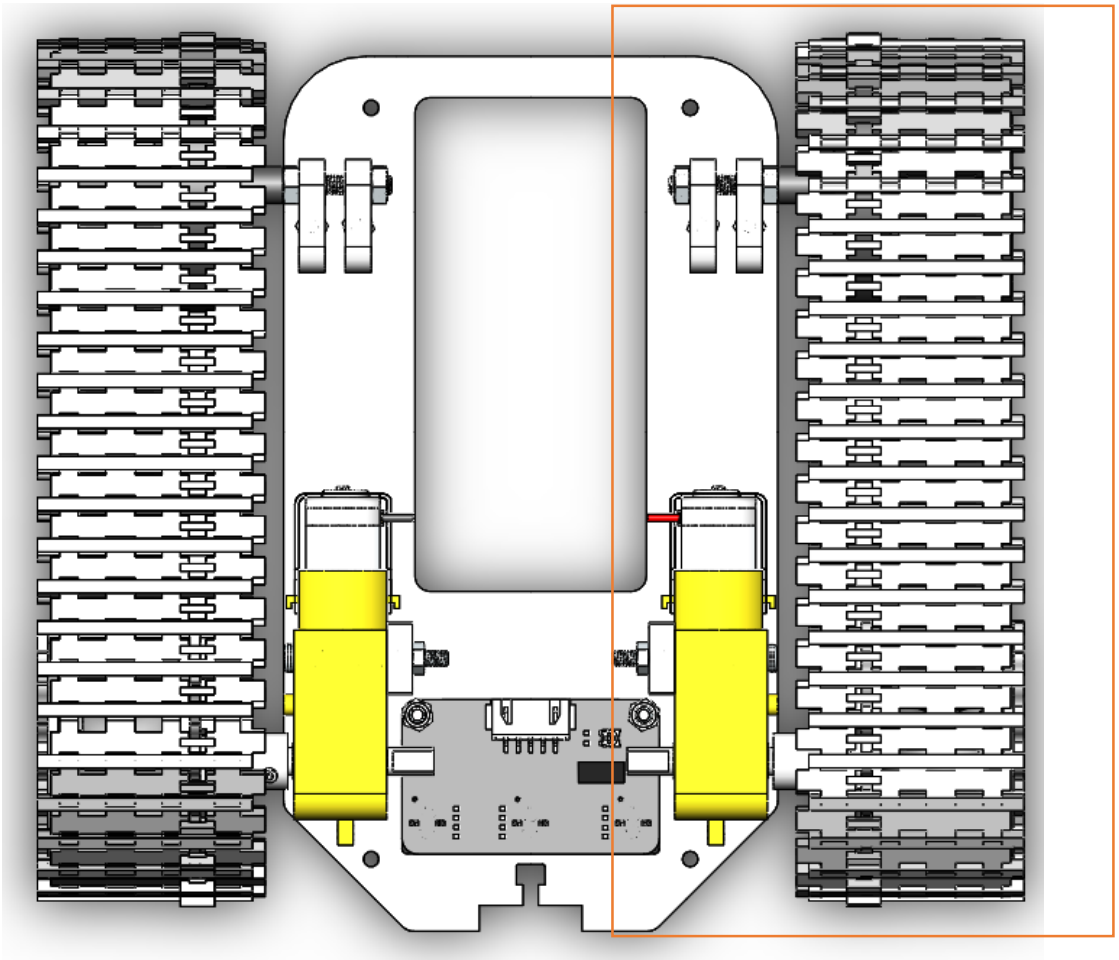


Step 8

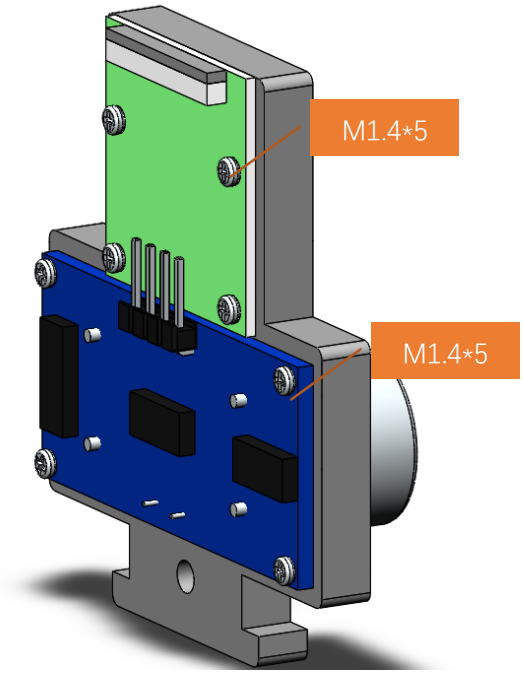




Step 9 Install another parts symmetrically.

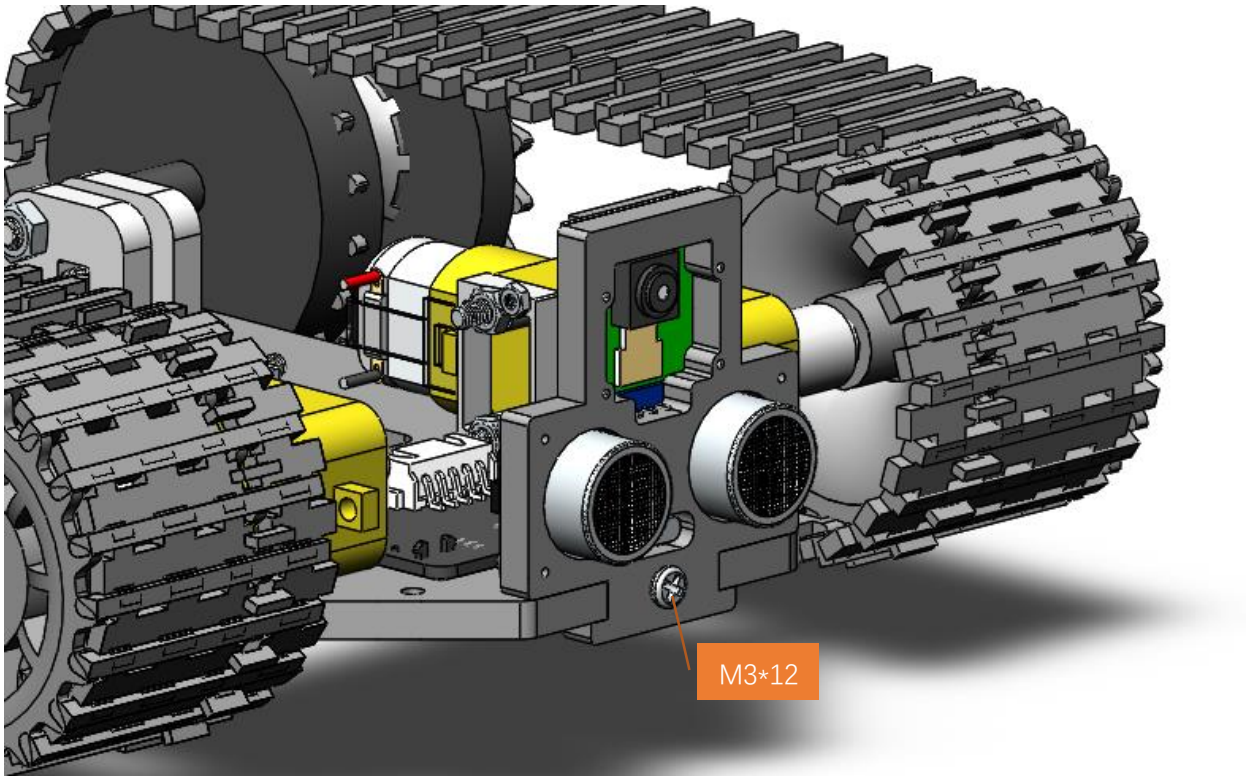


Step 10

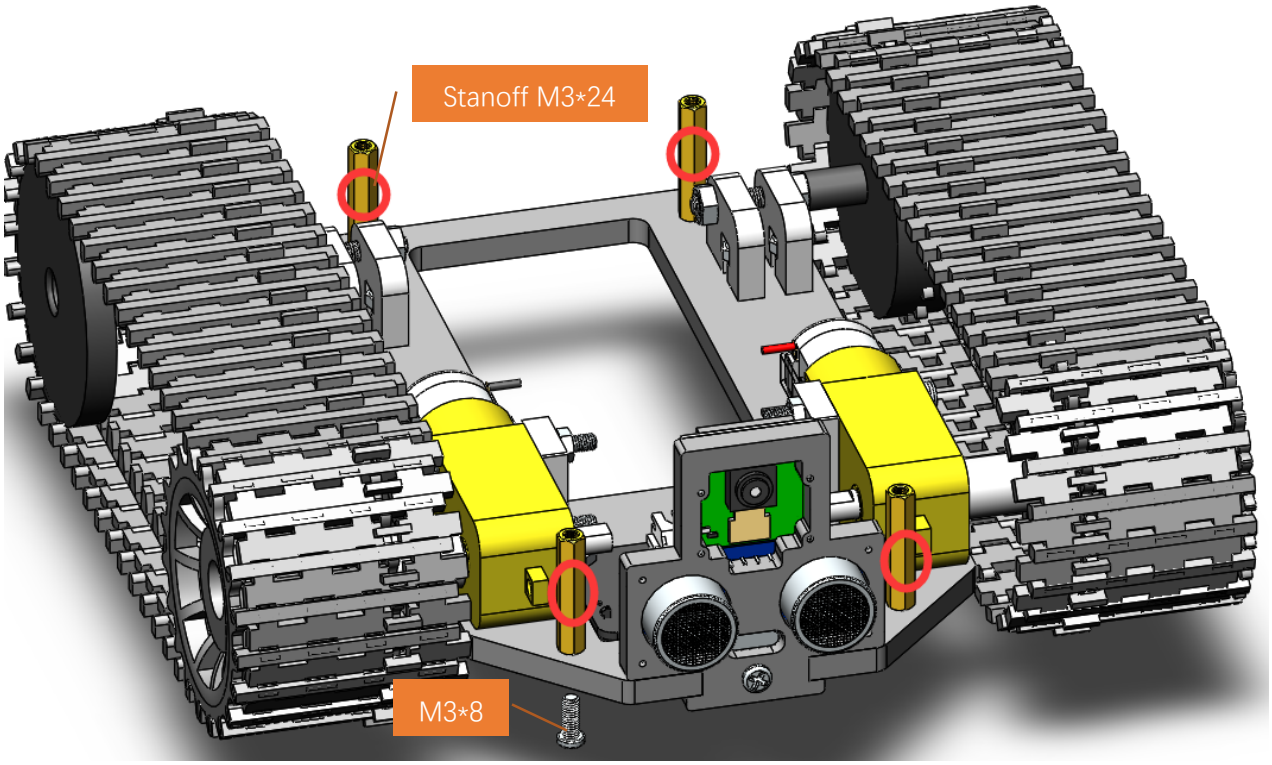


If they cannot be installed, please flip the acrylic board. The hole sizes are different on two sides.

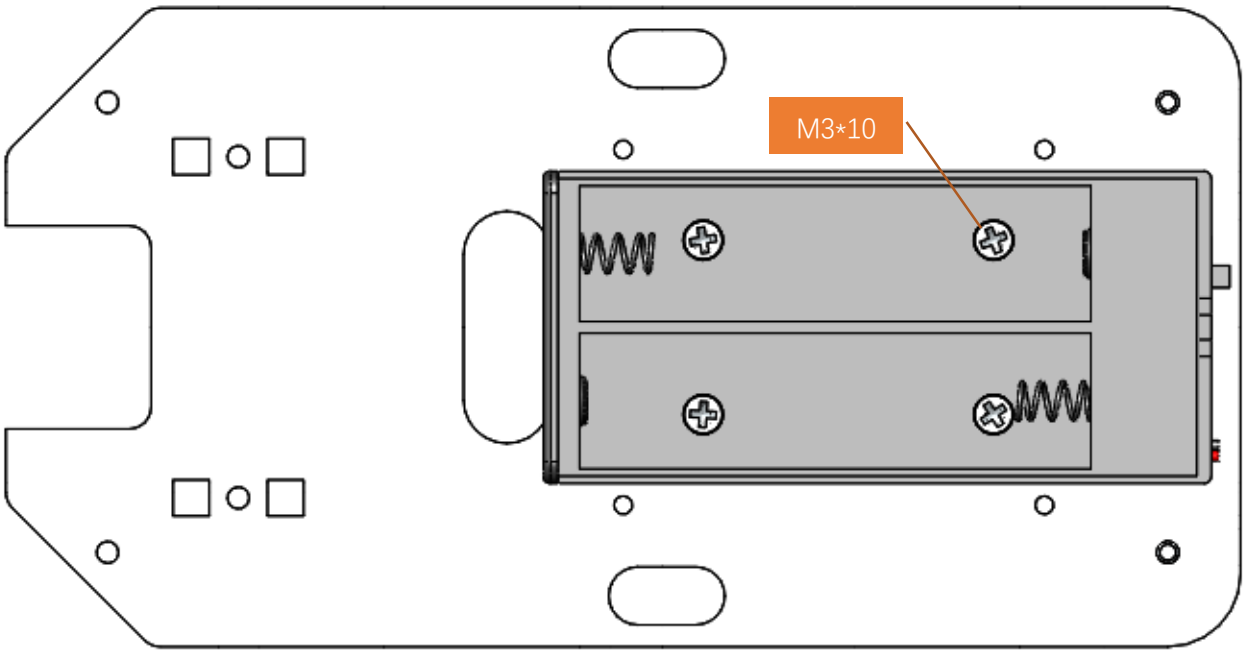
Step 11

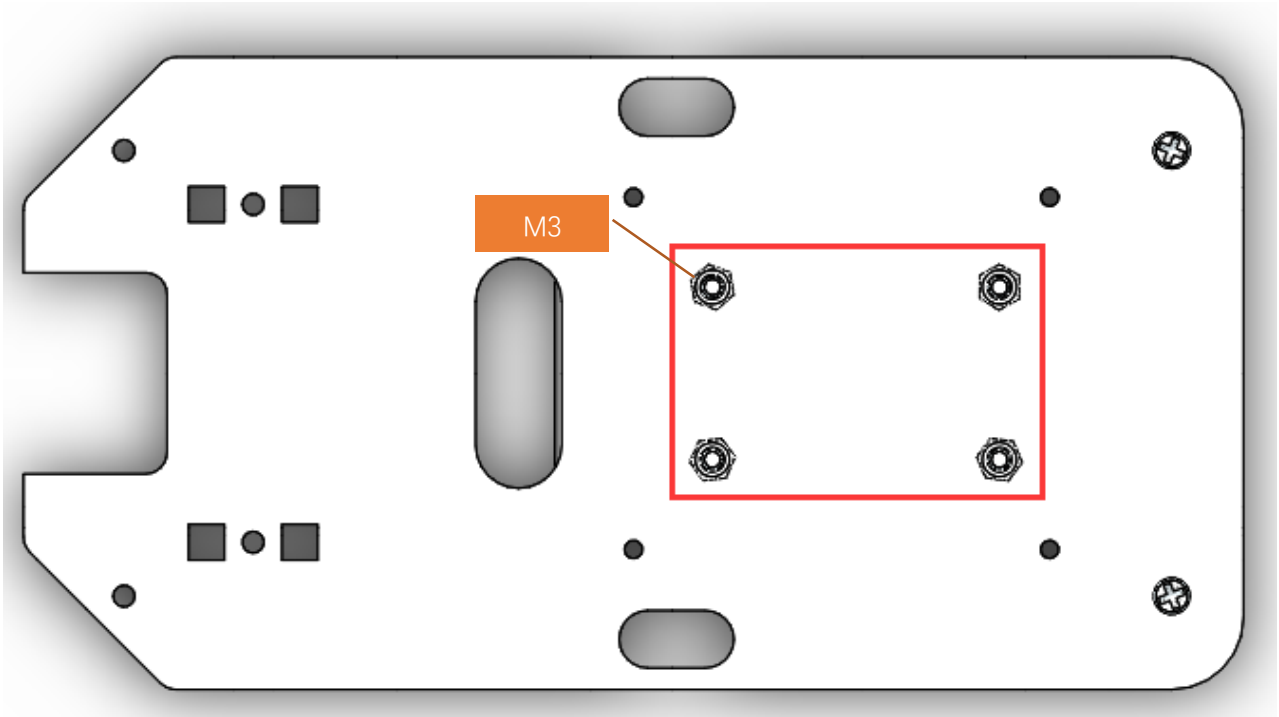


Step 12

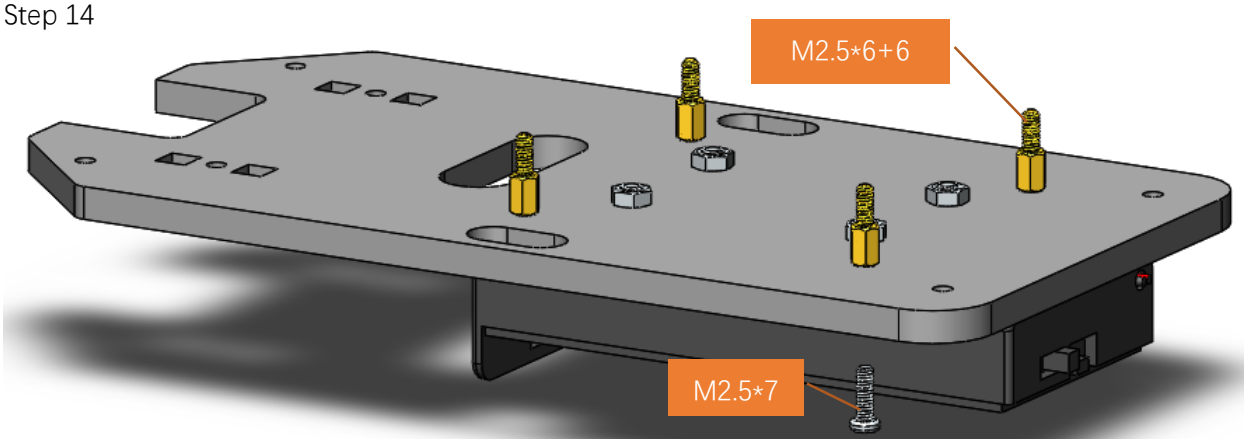


Step 13

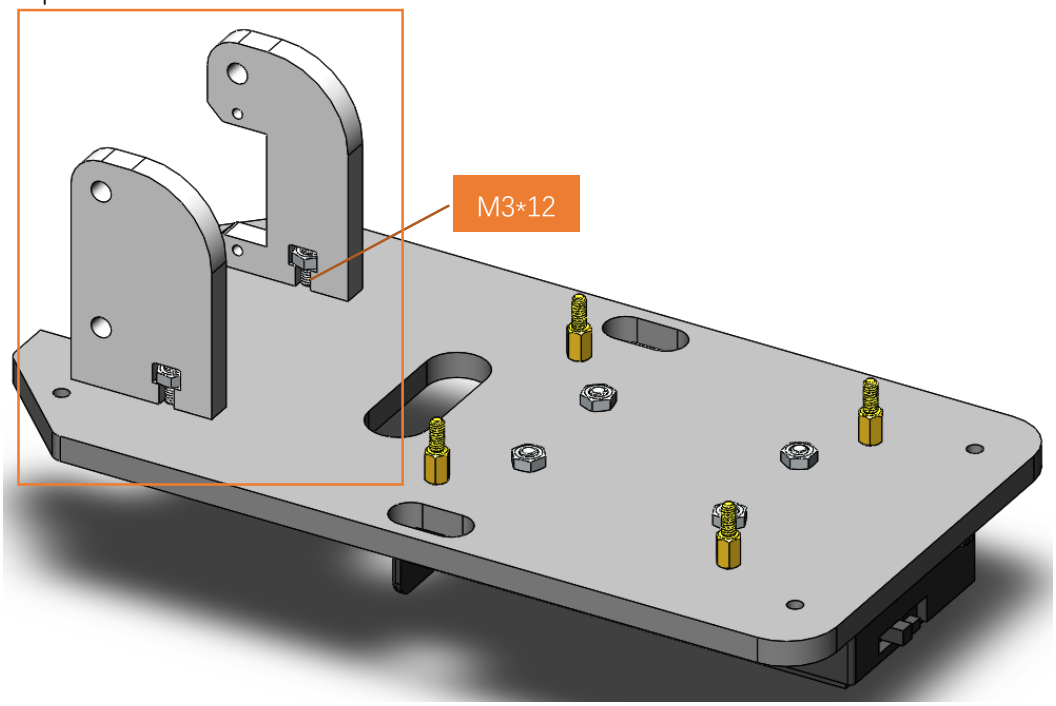




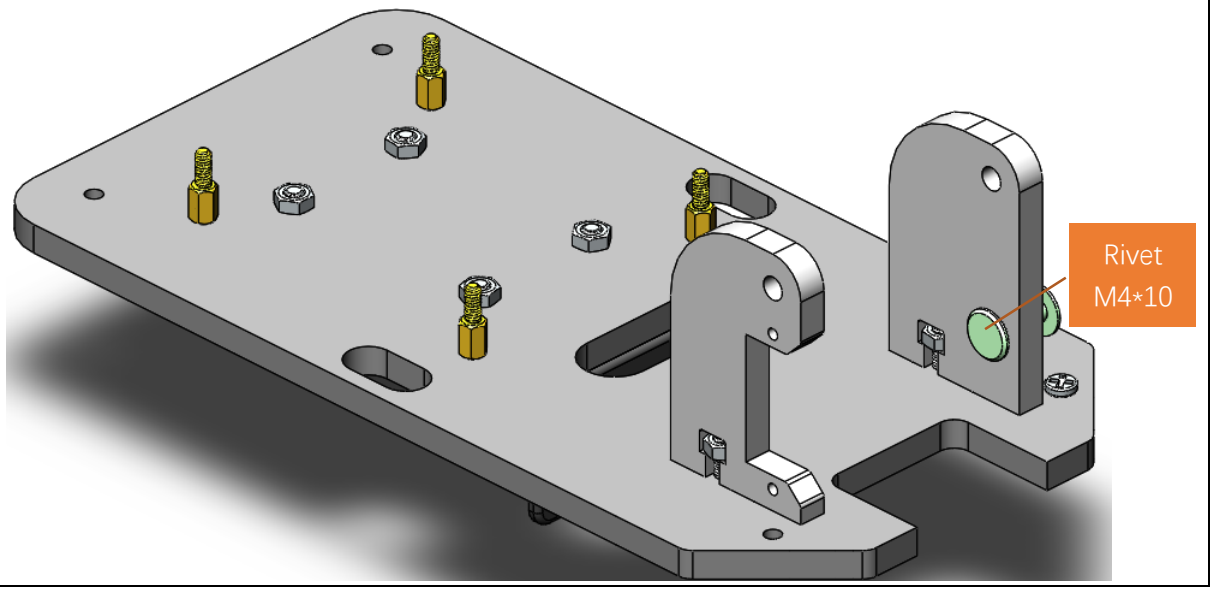
Step 14



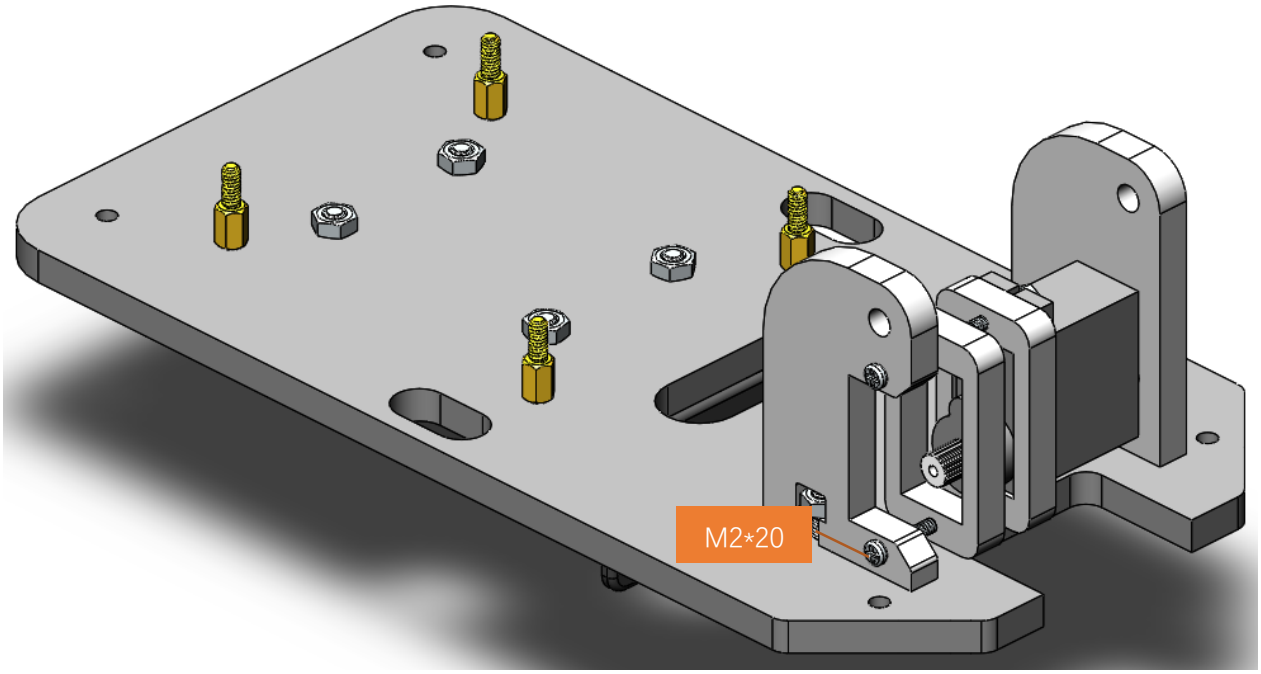
Step 15

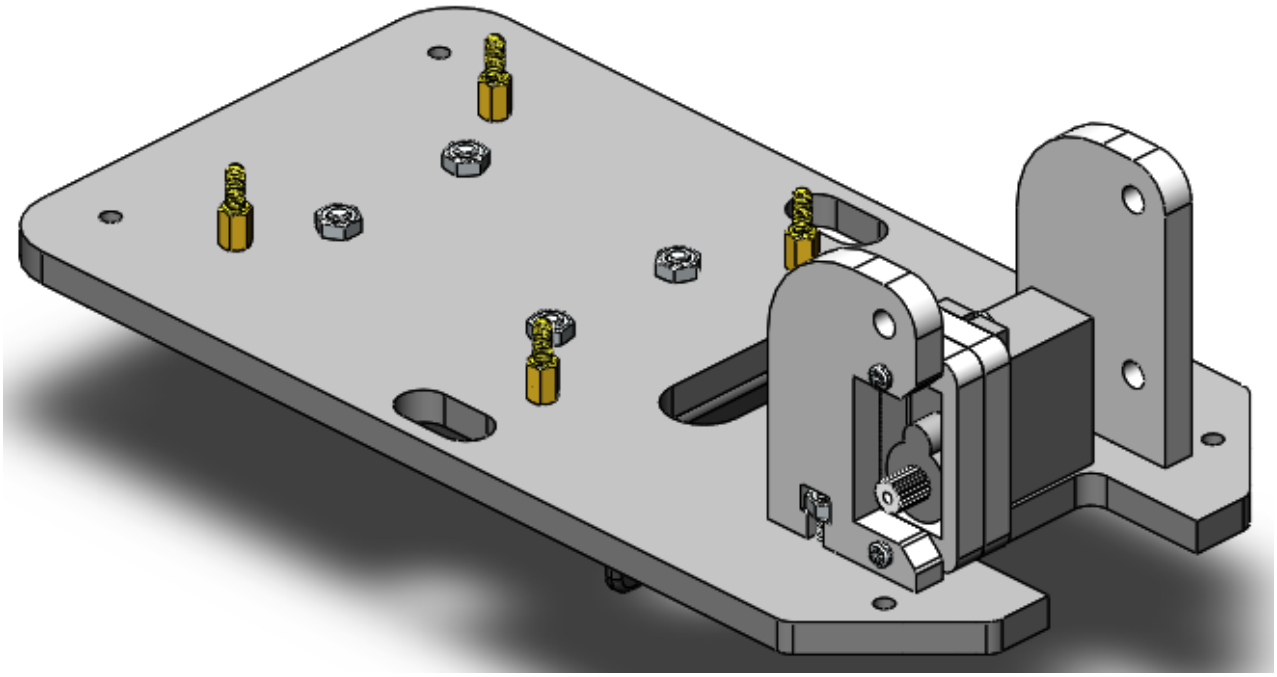


Step 16

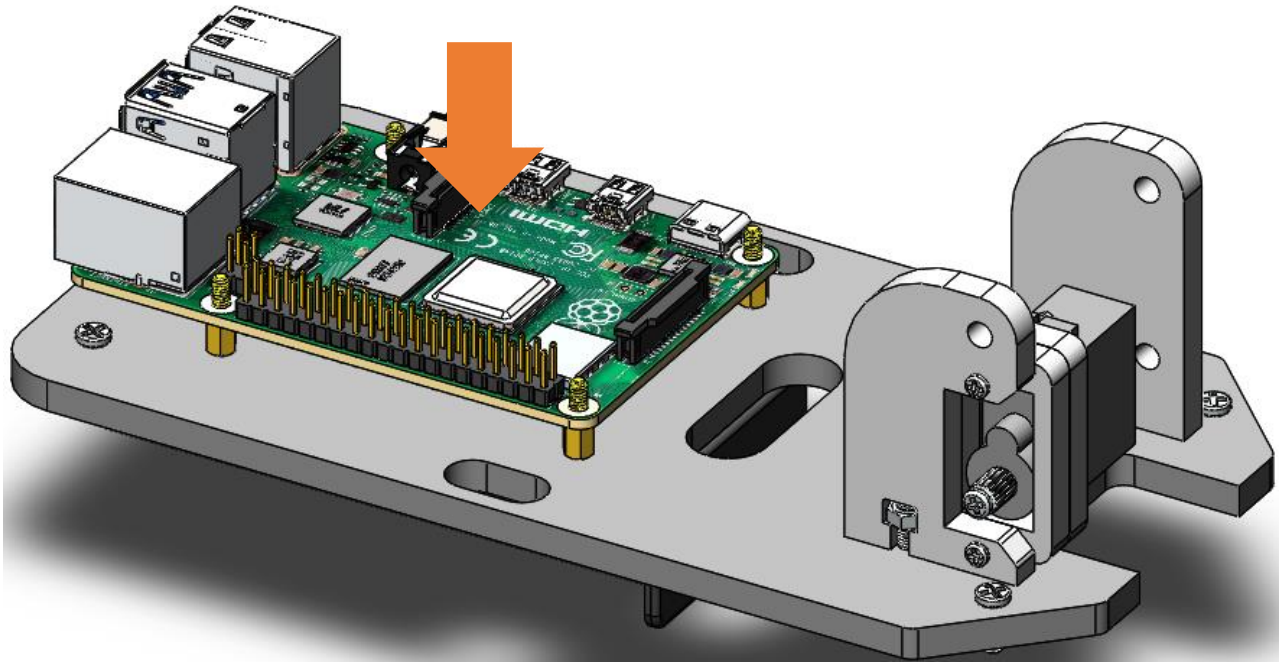


Step 17

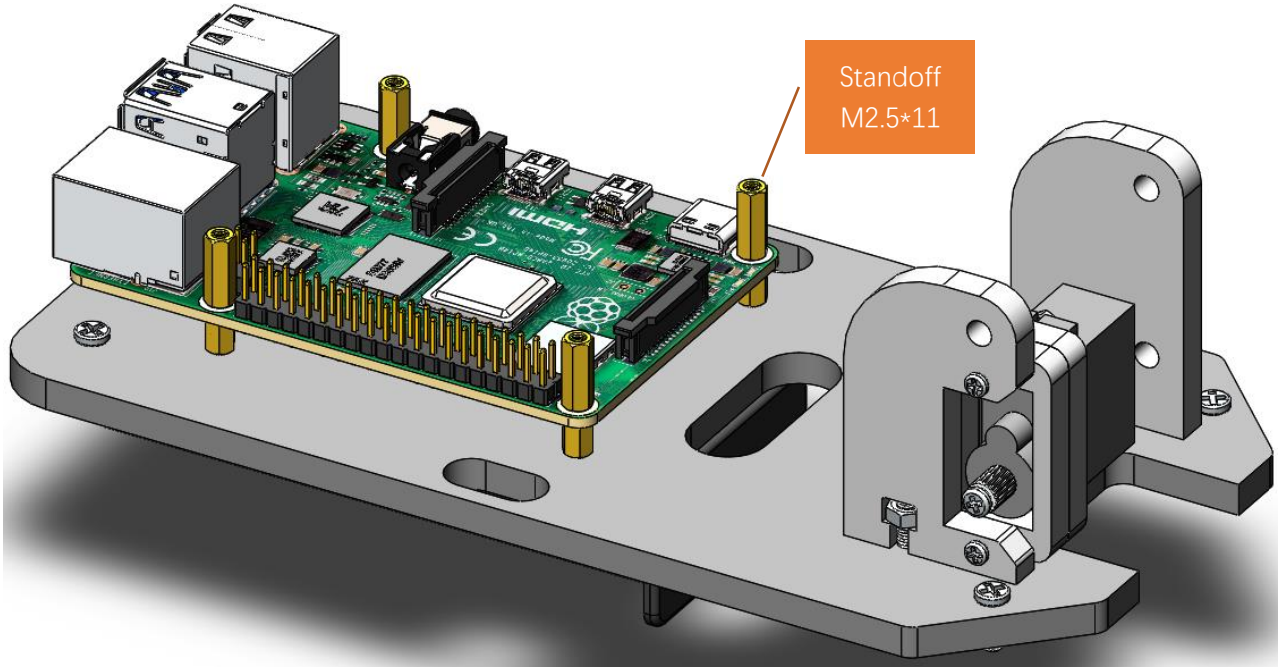




Step 18

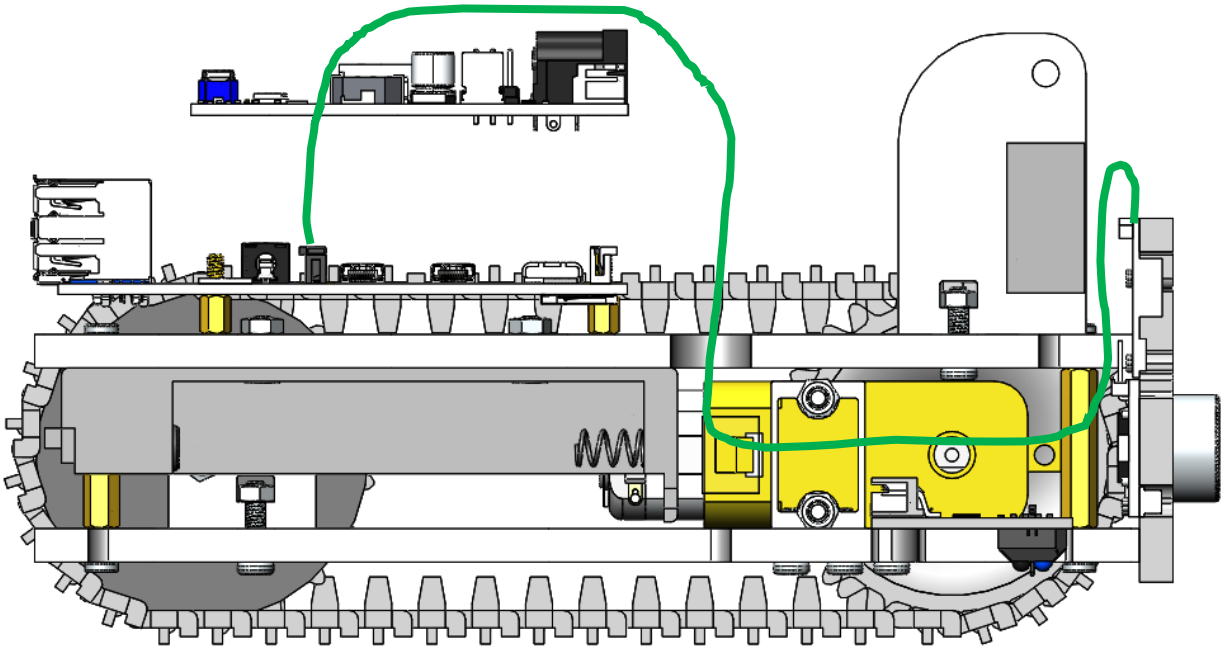


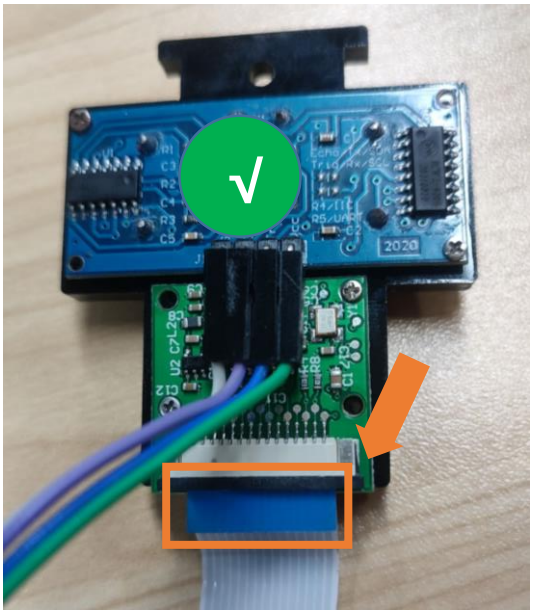
Step 19

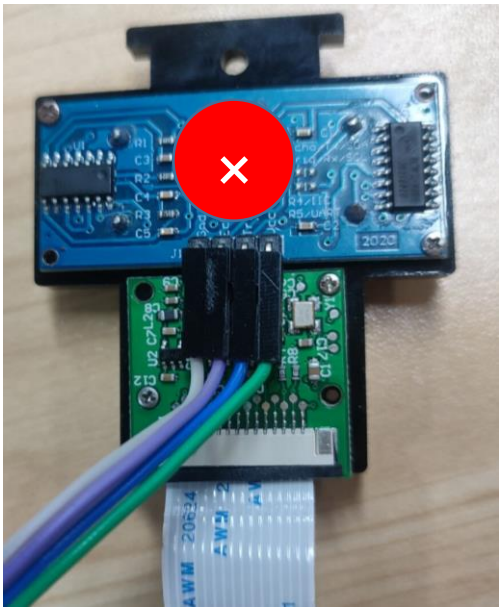


Step 20 Conect camera.

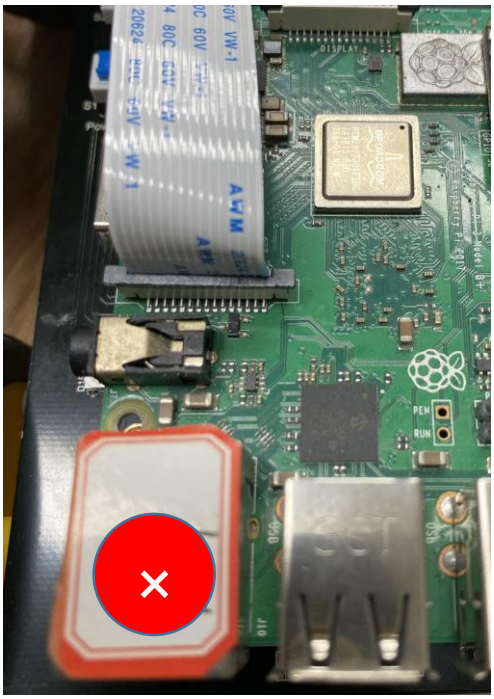
The CSI camera must be connected or disconnected under no power and when Raspberry Pi is shut down, or the camera may be burned.





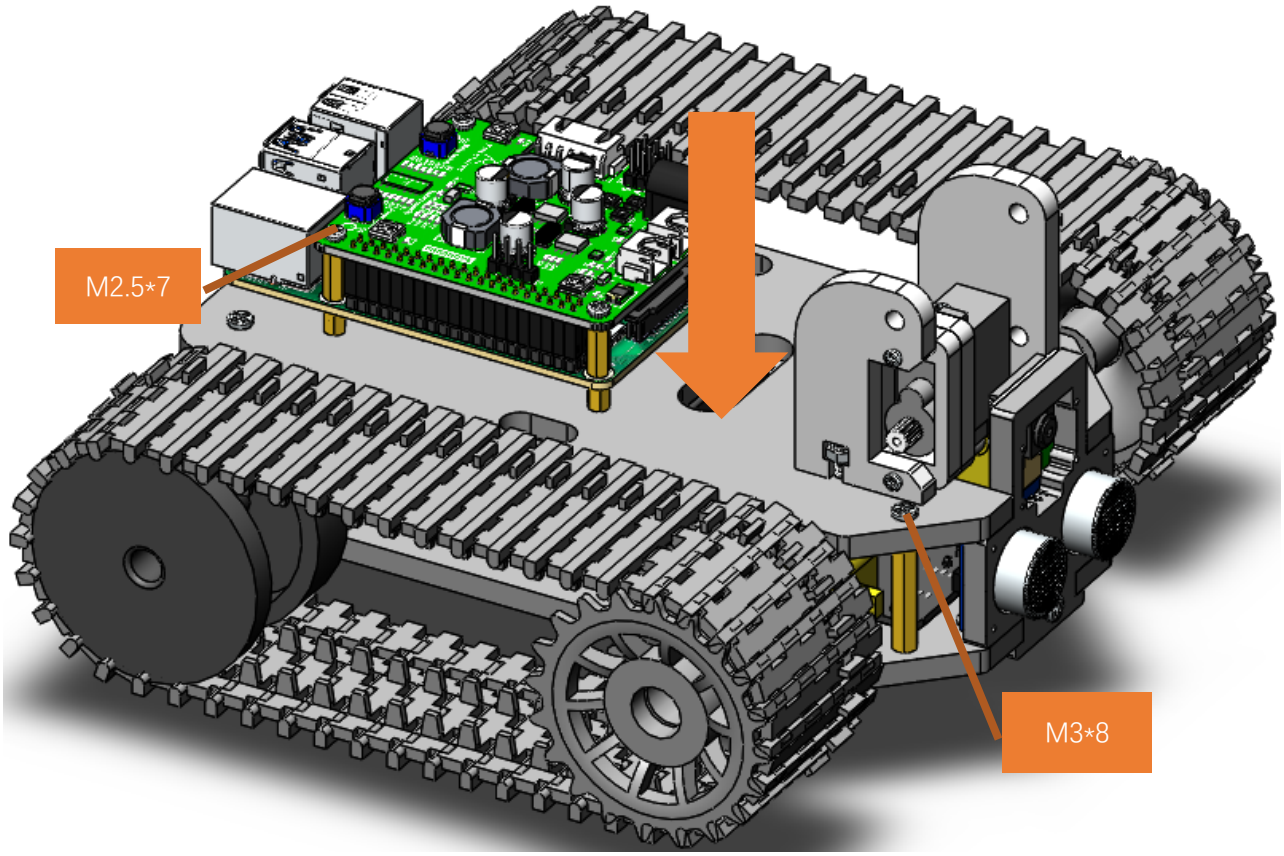


Pay attention to **the Blue bar of cable.**

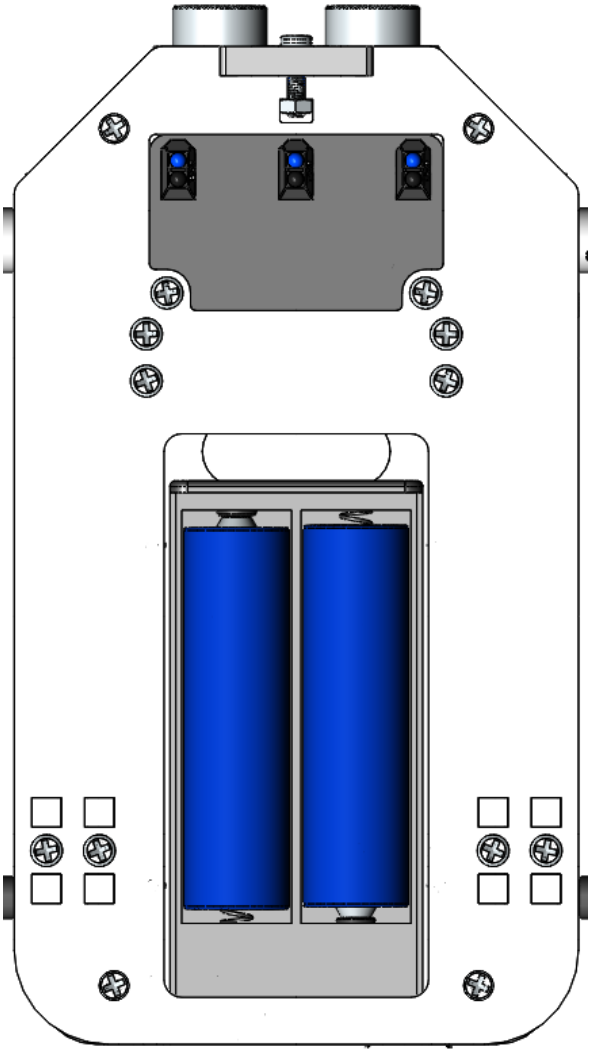


Pay attention to the Blue bar of cable.

Step 21

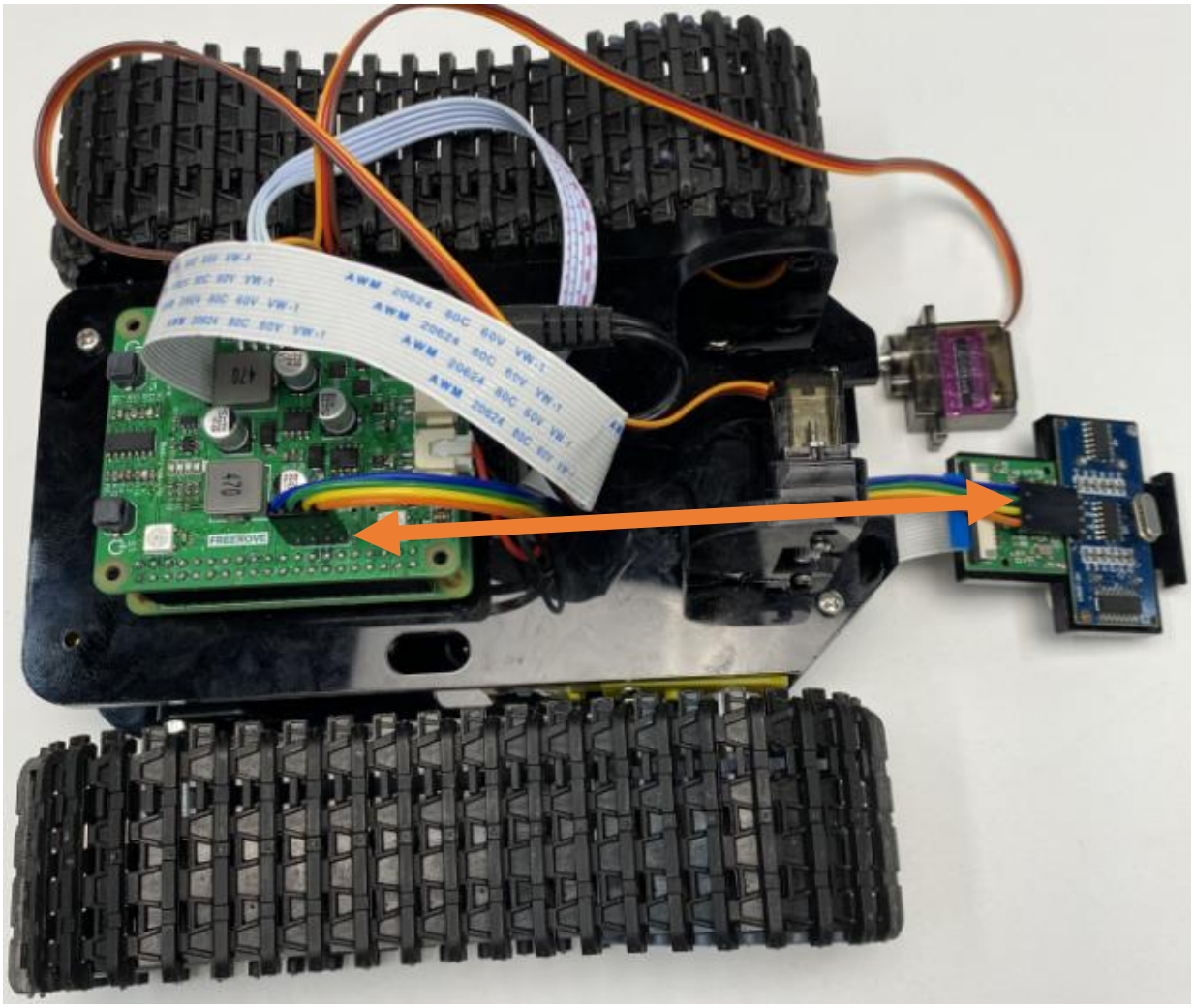


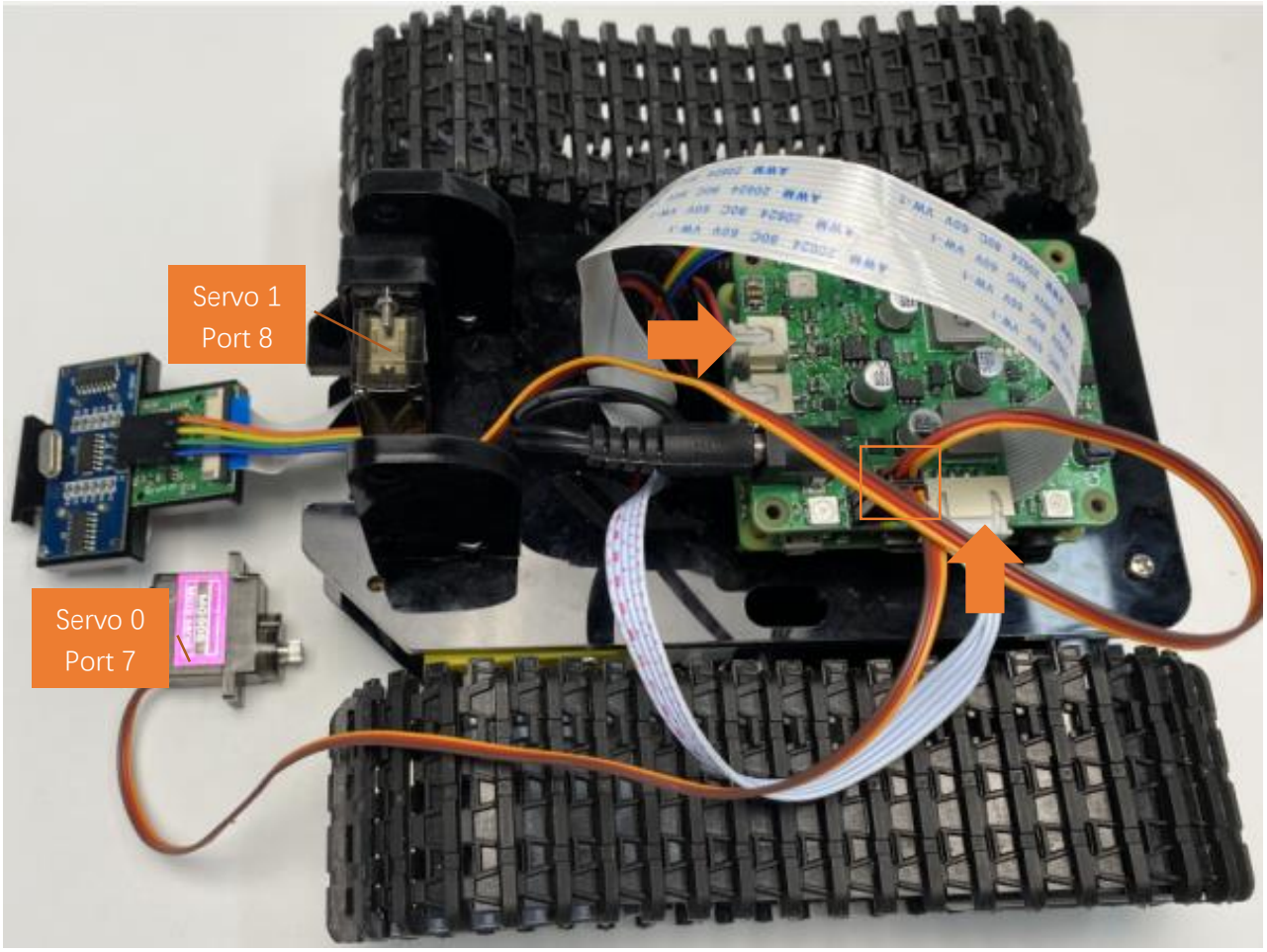
Step 22 Install batteries.



Turn on the switch of battery holder.

Step 23 Complete other wiring.





Step 24 Make servo rotate to 90°.

Turn on the two switches.

Execute following commands in terminal one by one.

```
sudo pigpiod
cd ~/Freenove_Tank_Robot_Kit_for_Raspberry_Pi/Code/Server
sudo python servo.py
```

If prompted to install the library, you can follow the steps below to install it.

In the latest Raspberry Pi OS, "pigpio" library has been installed. You only need to run the command to enable it.

```
sudo pigpiod
```

```
pi@raspberrypi:~ $ sudo pigpiod
pi@raspberrypi:~ $
```

If the "pigpio" library has not yet been installed, please follow the steps to install it.

Run the command to install "pigpio" library.

```
sudo apt-get install pigpio
```

```
pi@raspberrypi: ~
File  Edit  Tabs  Help
pi@raspberrypi:~ $ sudo apt-get install pigpio
```

As the "pigpio" library is used to control the motors and servos, the following command should be run before running the code.

```
sudo pigpiod
```

This is to open library runtime threads.

If it is not opened, such error will be reported when the code runs.

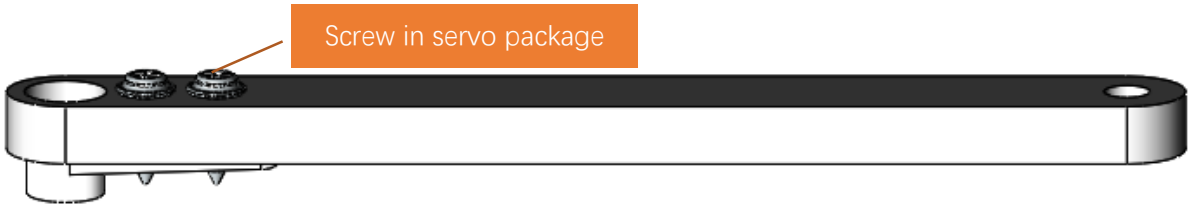
```
pi@raspberrypi: ~/Freenove_Tank_Robot_Kit_for_Raspberry_Pi/Code/Server
File  Edit  Tabs  Help
pi@raspberrypi:~ $ cd ~/Freenove_Tank_Robot_Kit_for_Raspberry_Pi/Code/Server
pi@raspberrypi:~/Freenove_Tank_Robot_Kit_for_Raspberry_Pi/Code/Server $ sudo python servo.py
Now servo 0 will be rotated to 150° and servos 1 will be rotated to 90°.
If they were already at 150° and 90°, nothing would be observed.
Please keep the program running when installing the servos.
After that, you can press ctrl-C to end the program.
Can't connect to pigpio at localhost(8888)

Did you start the pigpio daemon? E.g. sudo pigpiod

Did you specify the correct Pi host/port in the environment
variables PIGPIO_ADDR/PIGPIO_PORT?
E.g. export PIGPIO_ADDR=soft, export PIGPIO_PORT=8888

Did you specify the correct Pi host/port in the
pigpio.pi() function? E.g. pigpio.pi('soft', 8888)
Traceback (most recent call last):
  File "/home/pi/Freenove_Tank_Robot_Kit_for_Raspberry_Pi/Code/Server/servo.py", line 61, in <module>
    servo=Servo()
  File "/home/pi/Freenove_Tank_Robot_Kit_for_Raspberry_Pi/Code/Server/servo.py", line 11, in __init__
    self.PwmServo.set_mode(self.channel1,pigpio.OUTPUT)
  File "/usr/lib/python3/dist-packages/pigpio.py", line 1376, in set_mode
    return _u2i(_pigpio_command(self.sl, _PI_CMD_MODES, gpio, mode))
  File "/usr/lib/python3/dist-packages/pigpio.py", line 1025, in _pigpio_command
    sl.s.send(struct.pack('IIII', cmd, p1, p2, 0))
AttributeError: 'NoneType' object has no attribute 'send'
pi@raspberrypi:~/Freenove_Tank_Robot_Kit_for_Raspberry_Pi/Code/Server $
```

Please note that when the Raspberry Pi shuts down, the "pigpio" library will also be disabled automatically. You need to enable the "pigpio" library again when you boot up the Raspberry Pi next time.

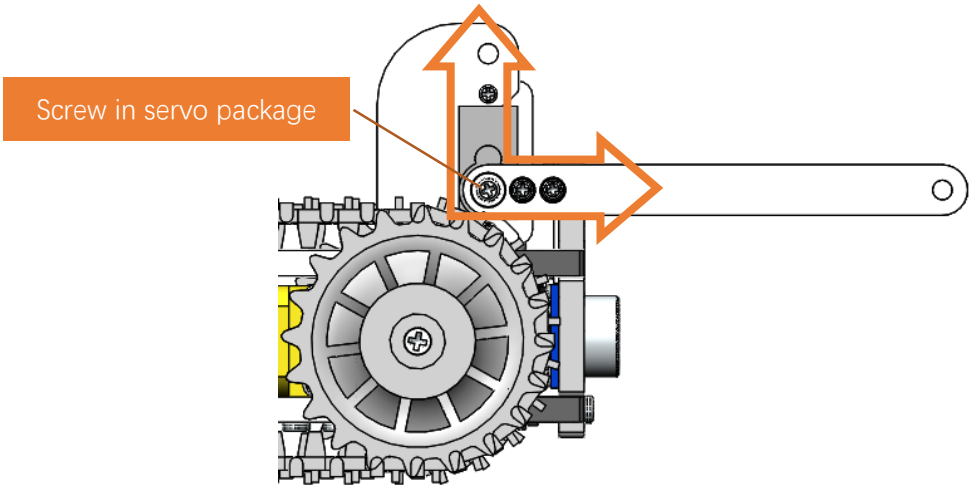
```
sudo pigpiod
```

```
pi@raspberrypi:~ $ sudo pigpiod
pi@raspberrypi:~ $
```

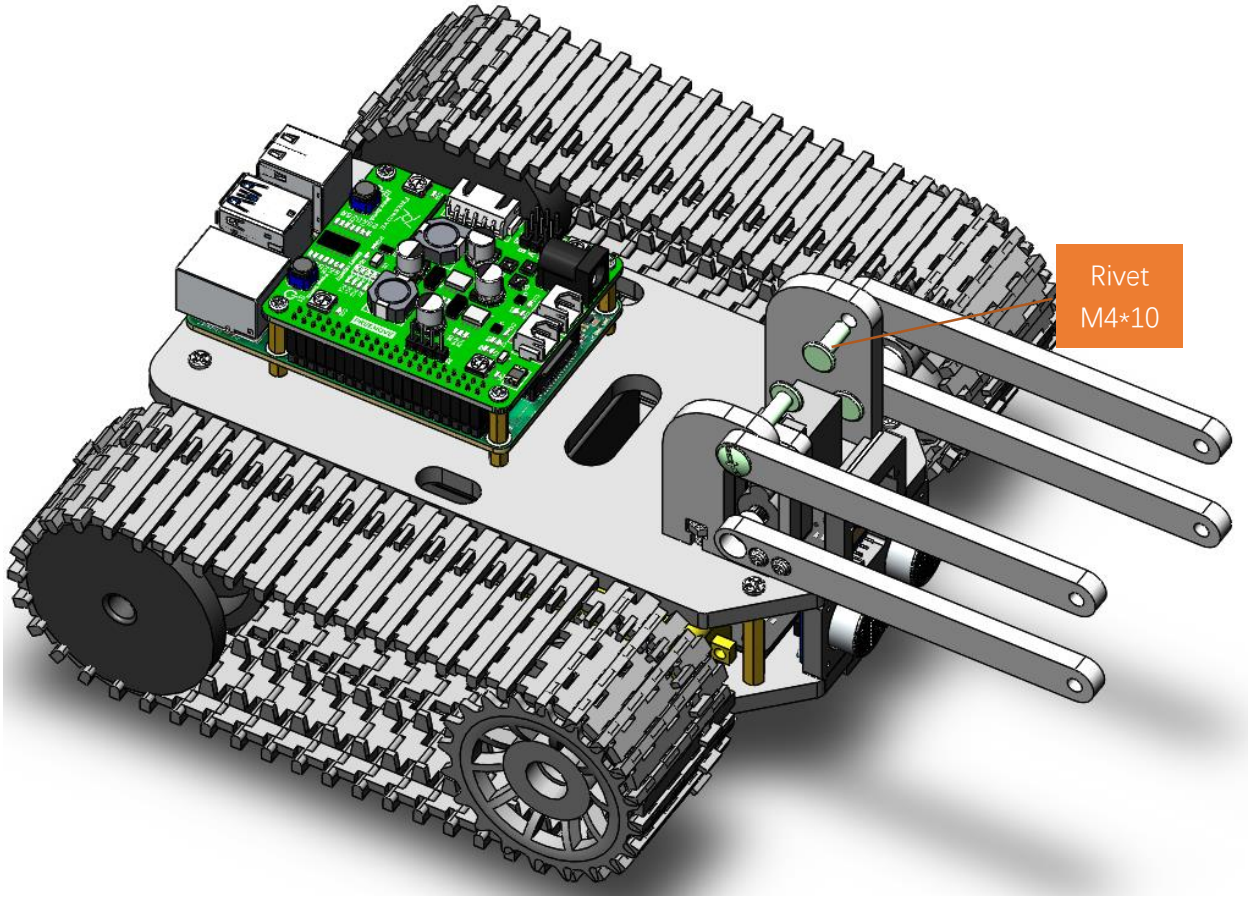
Step 25



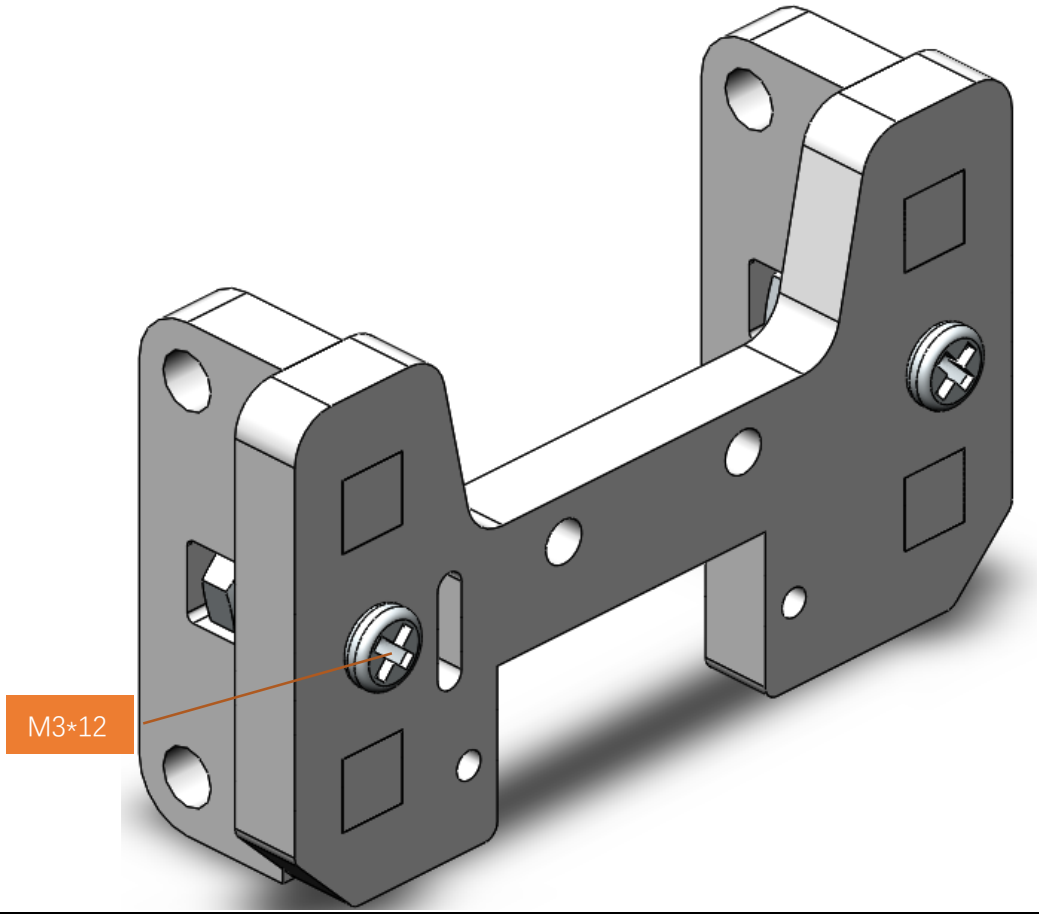
Step 26



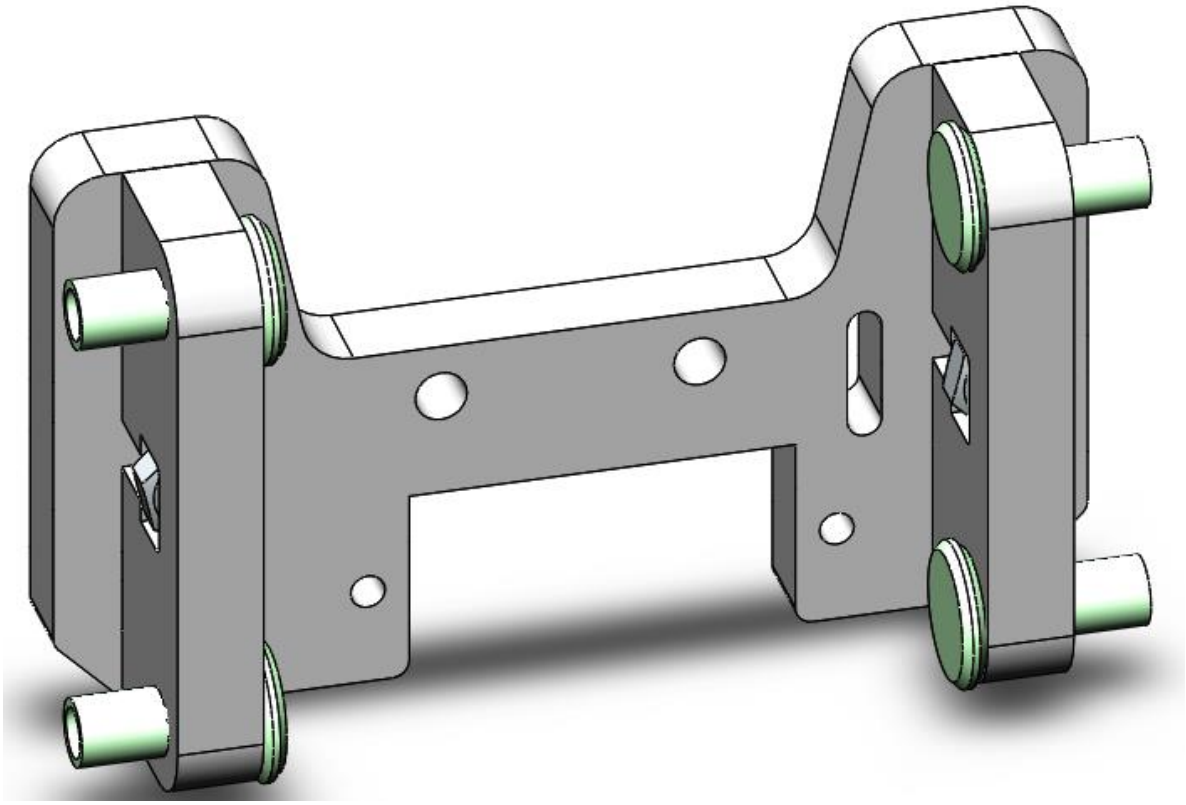
Step 27



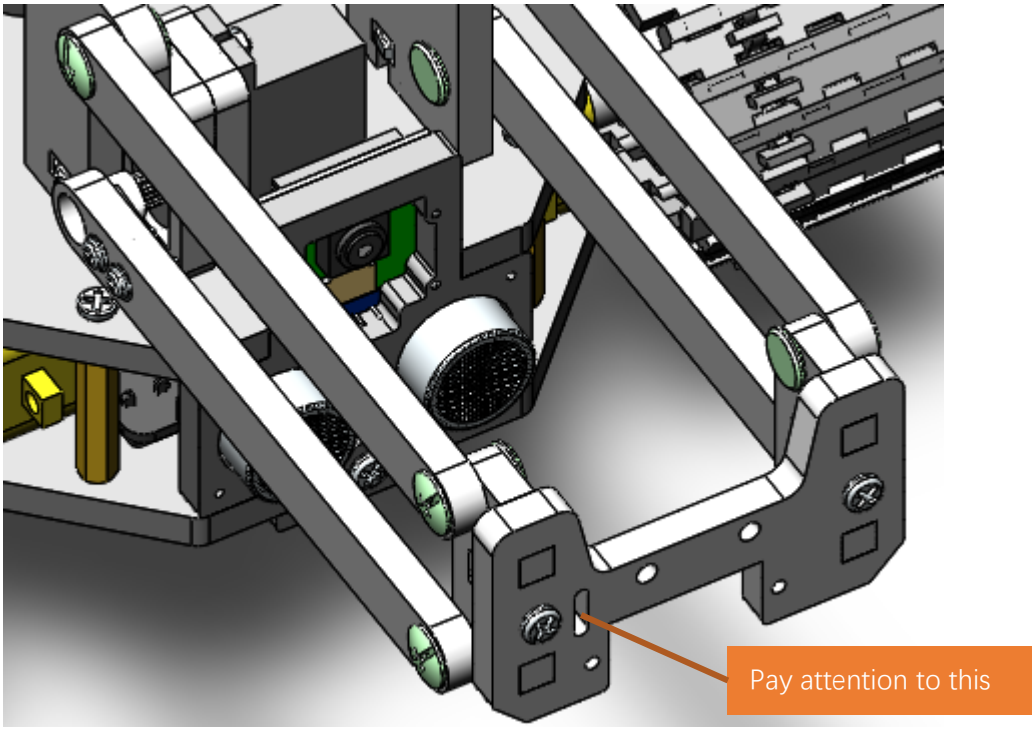
Step 28



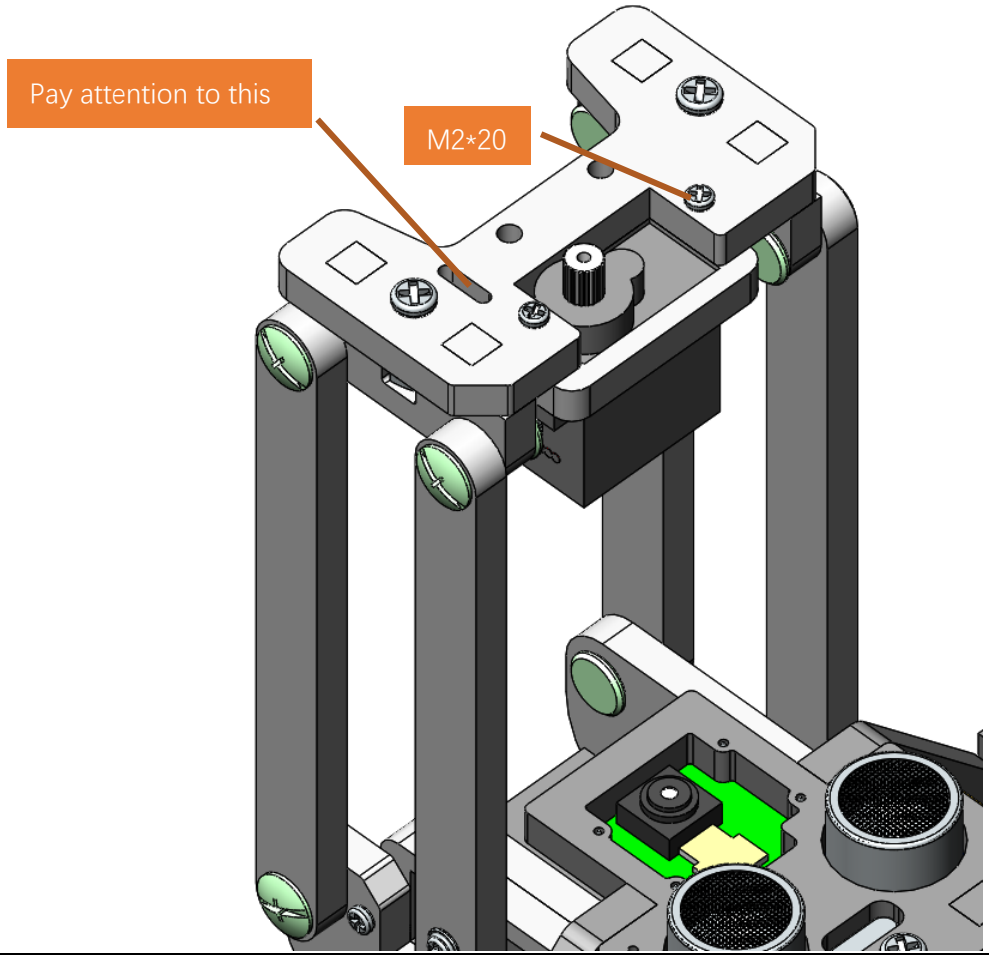
Step 29



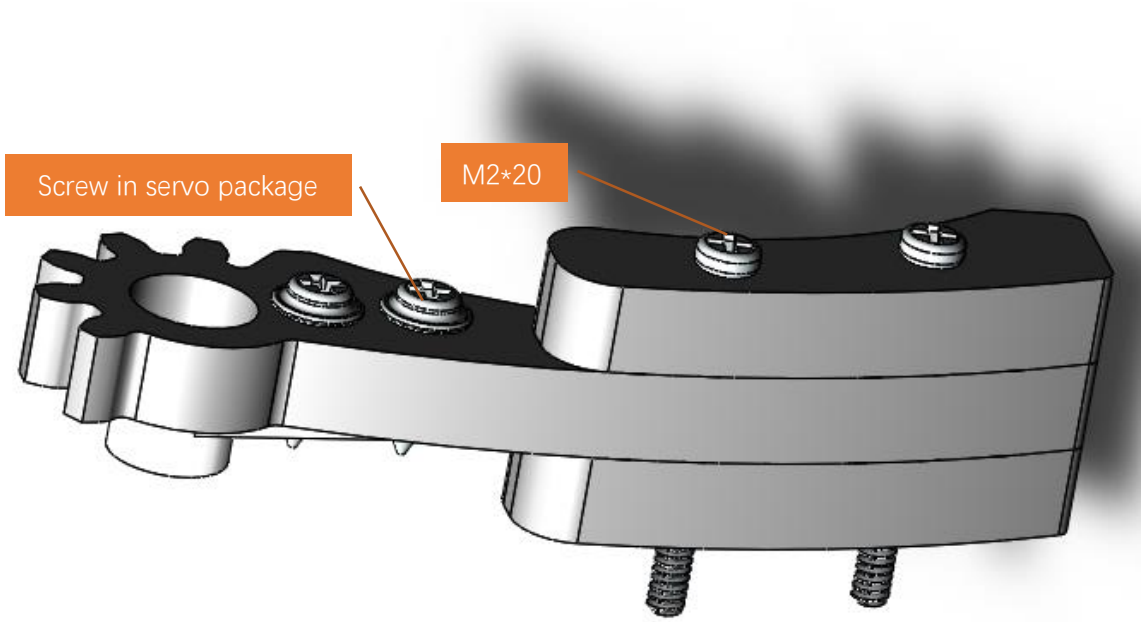
Step 30



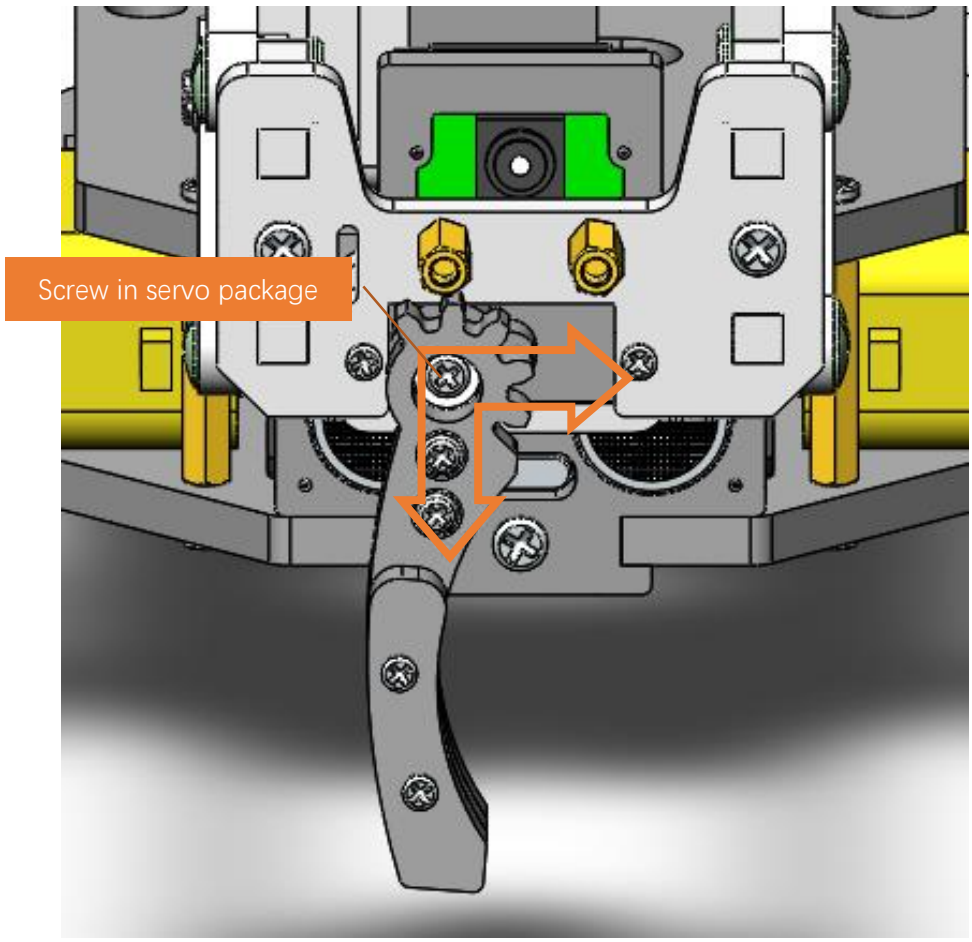
Step 31



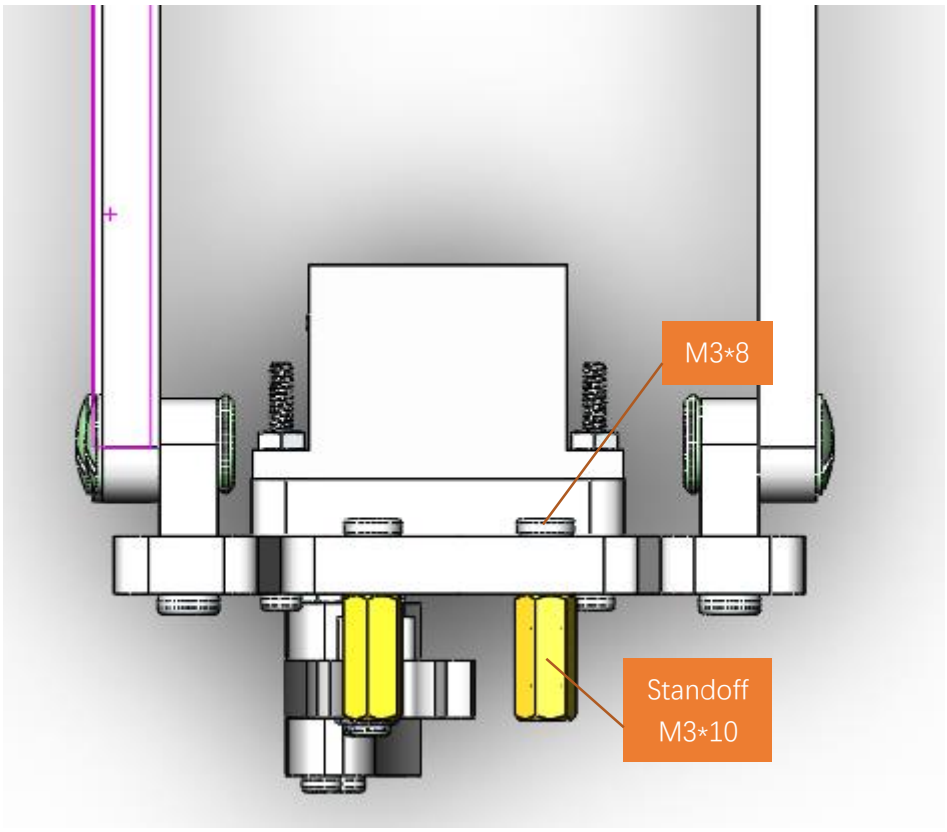
Step 32



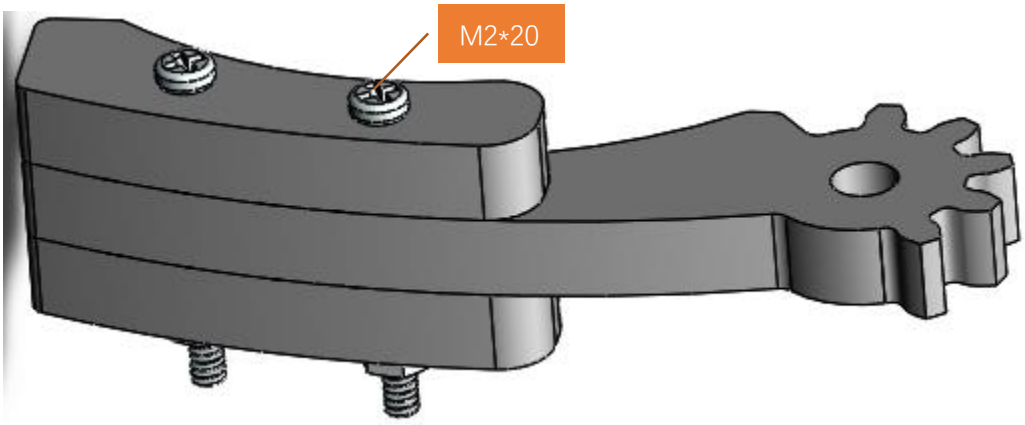
Step 33



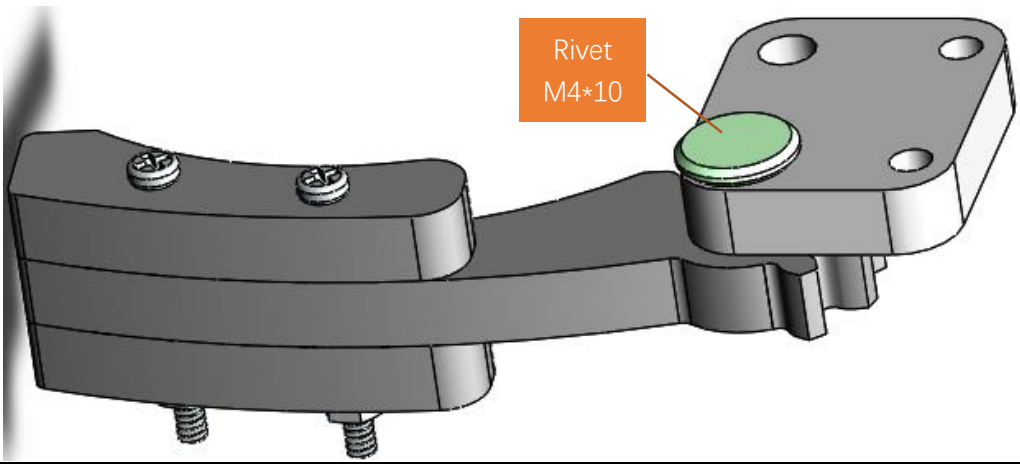
Step 34



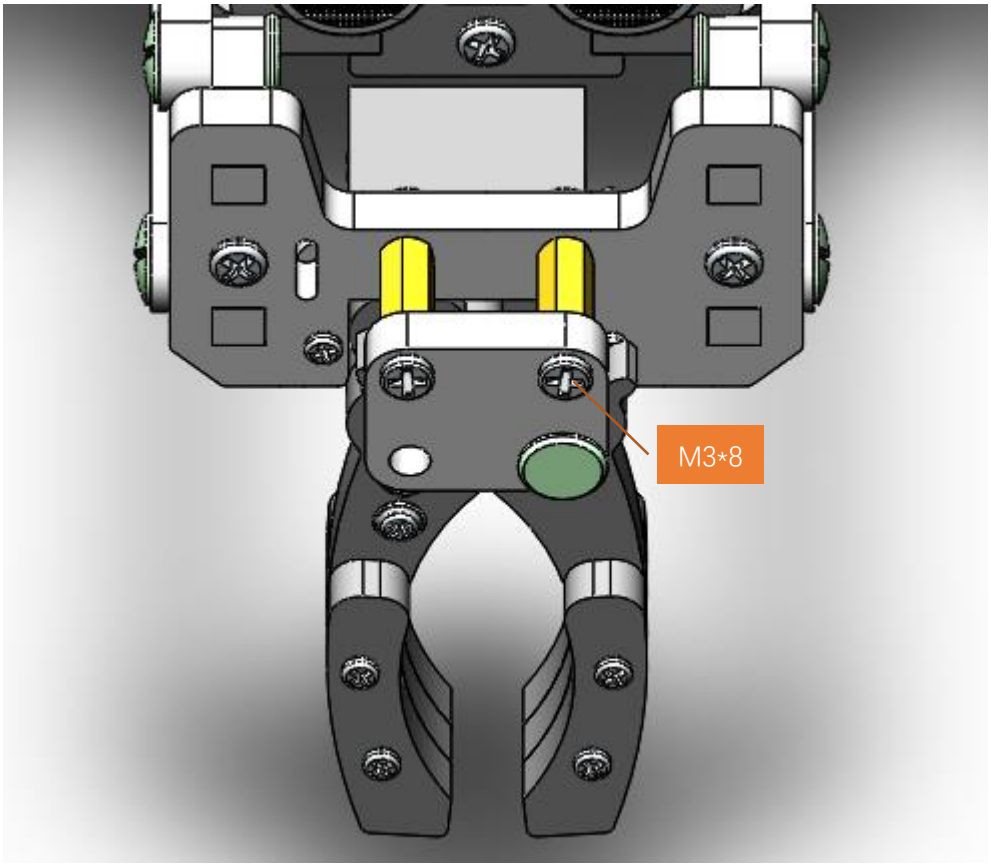
Step 35



Step 36



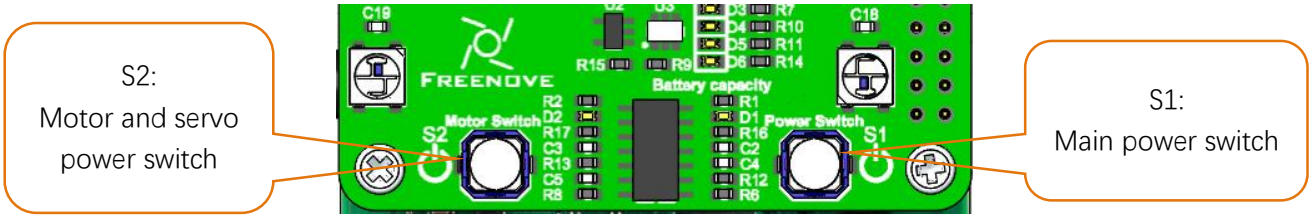
Step 37



Chapter 3 Module test (necessary)

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

This section requires that the car must be equipped with batteries, and S1 power switch and S2 motor switch should be pressed until the corresponding power indicator lights up.



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.

Button S1 is the Raspberry PI power supply switch. When you only press S1, you can perform other tests except motor and servo.

Button S2 is the motor and servo power switch. When you press S1 and S2, you can test the motor and servo.

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

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

Motor

Install pigpio library

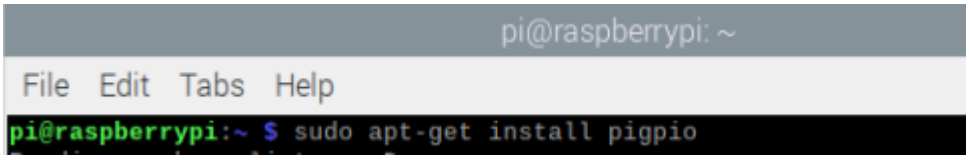
On the latest system, the library is already installed, you just need to run the following instructions to start the library.

```
sudo pigpiod
```

If you are prompted to install the library, you can do so by following these steps. If it is already installed, skip the following steps.

Type the following command to install “pigpio” library.

```
sudo apt-get install pigpio
```



Due to the "pigpio" library used for motor and servo control, the following commands need to be executed before you can run the written code.

```
sudo pigpiod
```

This is done to open a thread for the runtime of the library, and if not, the error will be prompted when running the code.

If you finish running the program and want to close the library, use the following command.

```
sudo killall pigpiod
```

If you want to know more pigpio library, can learn through the link below: https://abyz.me.uk/rpi/pigpio/

Run program

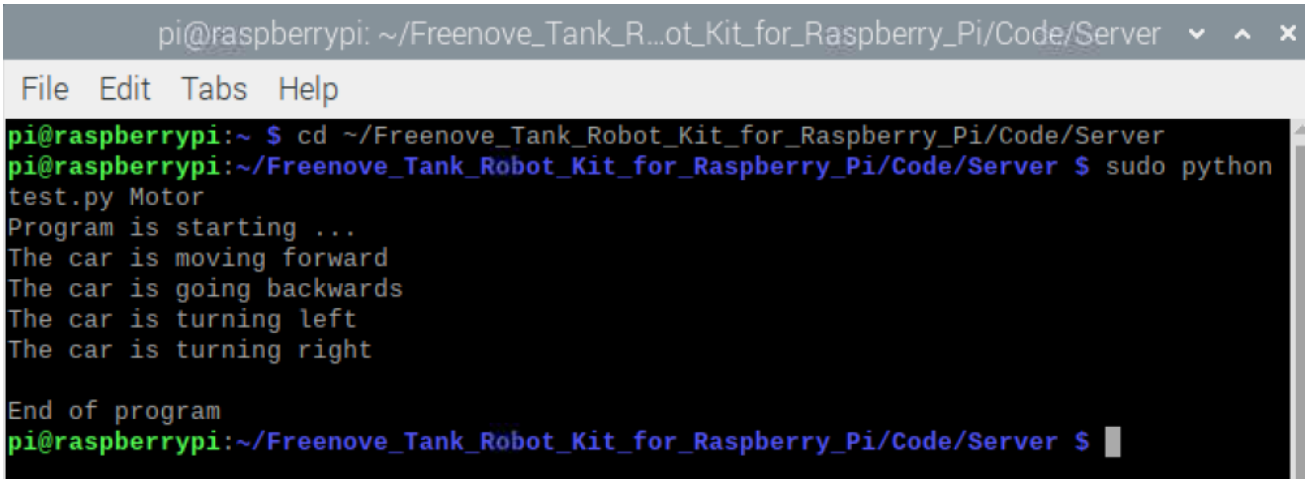
Open the terminal of Raspberry Pi. Enter the following commands to test the motor.

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

```
cd ~/Freenove_Tank_Robot_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. **If the car doesn't work normally, please check if both switches are pressed.**

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

1. Find Motor.py in the following path in your Raspberry Pi:
Freenove_Tank_Robot_Kit_for_Raspberry_Pi/Code/Server/Motor.py
Open Motor.py and add a "–" before duty1,2 like below. If one of the motors has a steering problem, only one of them needs to be changed, with duty1 representing the left motor and duty2 representing the right motor.

```
1  def setMotorModel(self,duty1,duty2):
2          duty1,duty2 =self.duty_range(duty1,duty2)
3          self.left_Wheel(-duty1)
4          self.right_Wheel(-duty2)
```

Then save the modification and try again.

It is also worth noting that the "pigpio" library will automatically close when you power off or restart your Raspberry PI. When you restart your Raspberry PI, you will need to re-enter the command to open the "pigpio" library, otherwise an error will be displayed when running the code.

"Pigpio Library" Auto Start

For ease of use, you can set up the "pigpio" library to start automatically on startup, so that you can run the code directly after startup. The specific Settings are as follows:

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

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

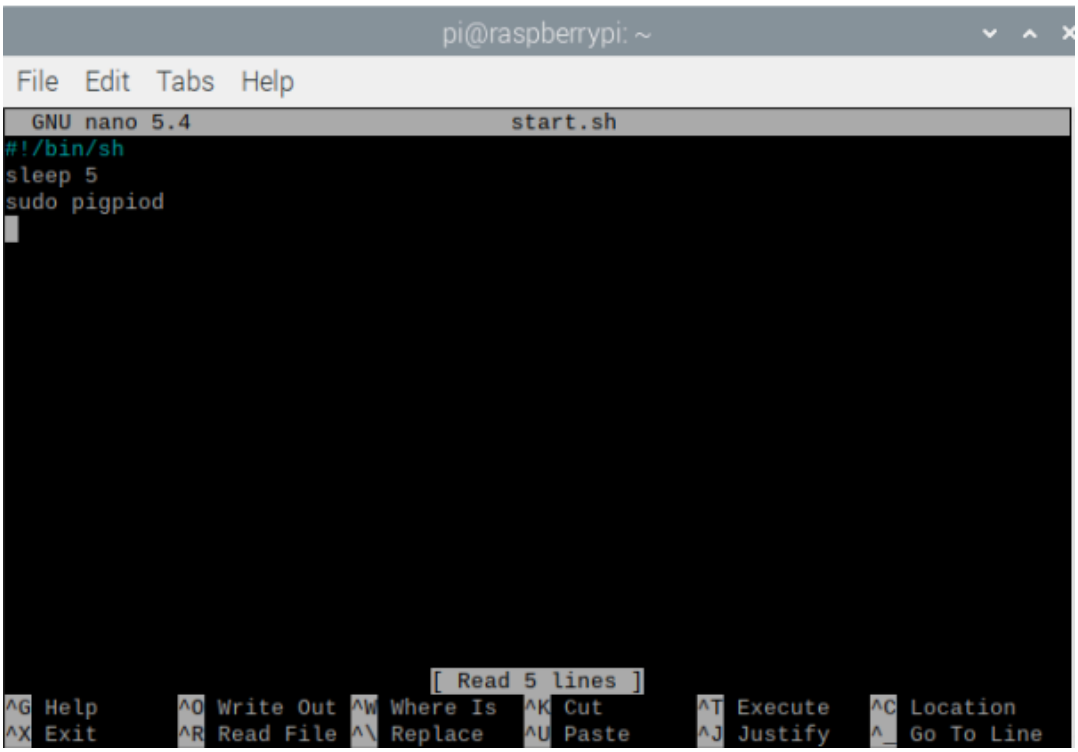
2 Open "start.sh".

```
sudo nano start.sh
```

3 Add the following contents to "start.sh" file.

```
#!/bin/sh
sleep 5
sudo pigpiod
```

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



4 Modify permissions.

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

5 Enter the following command to create a directory.

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

6 create and open “start.desktop” file

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

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

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

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

8 Modify permissions.

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

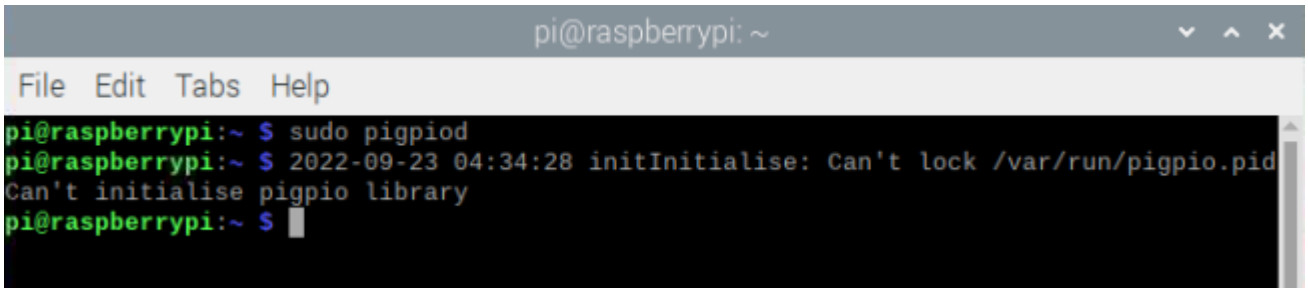
9 Finally enter the following content to reboot Raspberry Pi.

```
sudo reboot
```

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

When you restart, the pigpio library has been started, and you can enter the following command in the command:

```
sudo pigpiod
```



This indicates that the "pigpio" library has been started.

The code is as below:

```
from Motor import *
PWM=Motor()
def test_Motor():
    try:
        PWM.setMotorModel(2000,2000)       #Forward
        print ("The car is moving forward")
        time.sleep(1)
        PWM.setMotorModel(-2000,-2000)      #Back
        print ("The car is going backwards")
        time.sleep(1)
        PWM.setMotorModel(-2000,2000)       #Left
        print ("The car is turning left")
        time.sleep(1)
        PWM.setMotorModel(2000,-2000)       #Right
        print ("The car is turning right")
        time.sleep(1)
        PWM.setMotorModel(0,0)              #Stop
        print ("\nEnd of program")
    except KeyboardInterrupt:
        PWM.setMotorModel(0,0)
        print ("\nEnd of program")
```

Reference

setMotorModel(data1,data2)

This function has two input parameters, which control the left motor and the right motor respectively. When the input parameter is in the range of 0~4095, the motor rotates forward. In the range of -4095~0, the motor rotates in reverse. The larger the absolute value, the faster the motor. When the input is 0, the motor will stop. If the function input is as follows :setMotorModel(2000,2000), the two motors will rotate forward and the car will move forward.

Infrared Line tracking module

Run program

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

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

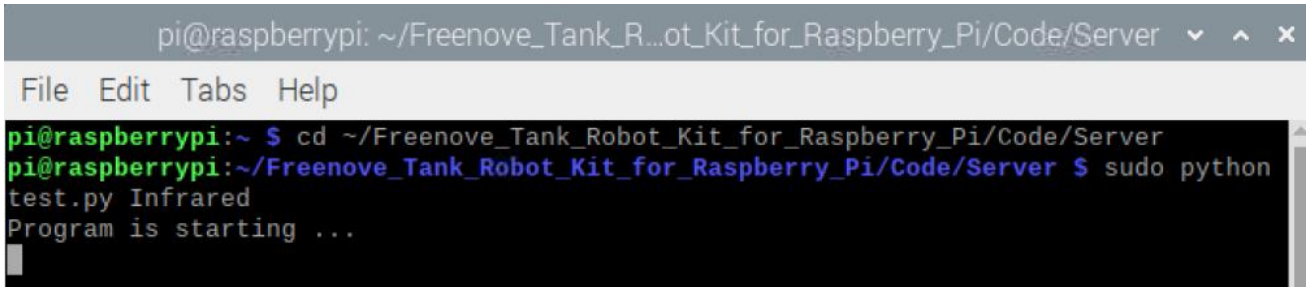
```
pi@raspberrypi:~/Freenove_Tank_Robot_Kit_for_Raspberry_Pi/Code/Server $
```

1. If not, execute the cd command:

```
cd ~/Freenove_Tank_Robot_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:

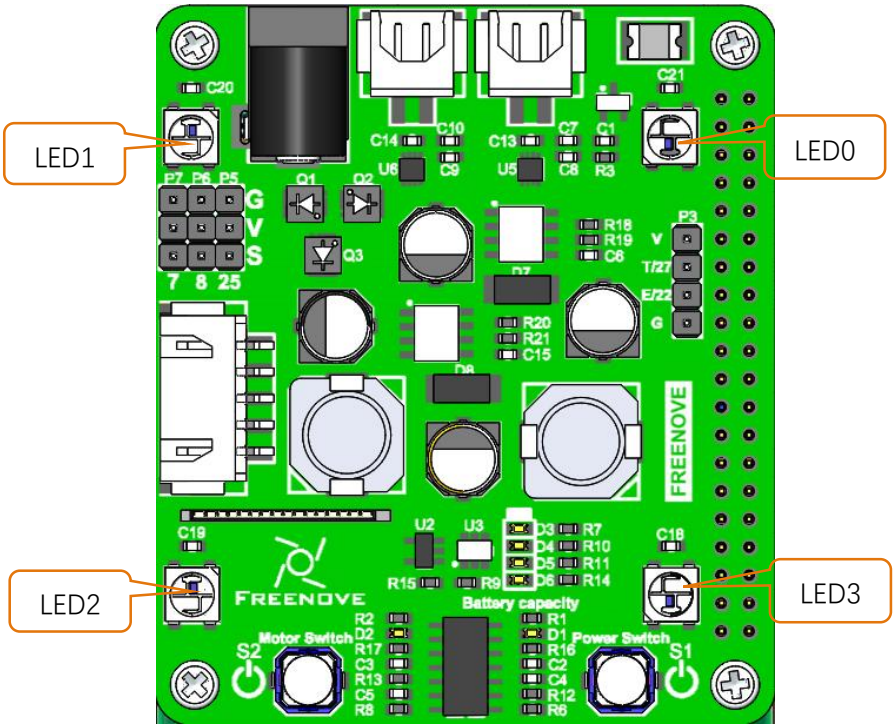
```
from Line_Tracking import *
line=Line_Tracking()
def test_Infrared():
    try:
        while True:
            if GPIO.input(IR01)!=True and GPIO.input(IR02)==True and GPIO.input(IR03)!=True:
                print ('Middle')
            elif GPIO.input(IR01)!=True and GPIO.input(IR02)!=True and GPIO.input(IR03)==True:
                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 4 RGB LEDs on the smart car board, as shownbelow. You can control them separately.



Run program

Enter the following commands to test LEDs.

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

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

1. If not, execute the cd command:

```
cd ~/Freenove_Tank_Robot_Kit_for_Raspberry_Pi/Code/Server
```

2. Execute test.py command:

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

```
pi@raspberrypi: ~/Freenove_Tank_R...ot_Kit_for_Raspberry_Pi/Code/Server
File  Edit  Tabs  Help
pi@raspberrypi:~ $ cd ~/Freenove_Tank_Robot_Kit_for_Raspberry_Pi/Code/Server
pi@raspberrypi:~/Freenove_Tank_Robot_Kit_for_Raspberry_Pi/Code/Server $ sudo python
test.py Led
Program is starting ...
The LED has been lit, the color is red green blue white

End of program
pi@raspberrypi:~/Freenove_Tank_Robot_Kit_for_Raspberry_Pi/Code/Server $
```

Result:

All LEDs will be turned on for 3 seconds, and colors from LED0 to LED3 are: red, green, blue and white. You can end the program ahead of time by pressing "ctrl+c".

If the LED color display order is not correct, open the "**Led.py**" file in the current directory and modify the value of the "self.ORDER" variable on line 16.

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

The code of test.py is as below:

```
1  import time
2  from Led import *
3  led=Led()
4  def test_Led():
5          try:
6              led.ledIndex(0x01,255,0,0)       #Red
7              led.ledIndex(0x02,0,255,0)       #green
8              led.ledIndex(0x04,0,0,255)       #blue
9              led.ledIndex(0x08,255,255,255)   #white
10
11             print ("The LED has been lit, the color is red green blue white")
12             time.sleep(3)                    #wait 3s
13             led.colorWipe(led.strip, Color(0,0,0))  #turn off the light
14             print ("\nEnd of program")
15         except KeyboardInterrupt:
16             led.colorWipe(led.strip, Color(0,0,0))  #turn off the light
17             print ("\nEnd of program")
```

Reference

ledIndex(Index, R, G, B)
This function has 4 parameters. The first one is the index of the LED that you want to control. Its value is hexadecimal. There are LED0~3. The rest 3 parameters are R G B value of color respectively. For example, ledeindex(0x01,255,0,0) makes LED 0 light to red; ledeindex(0x08,255,255,255) makes LED 3 light white.
colorWipe(strip, color, wait_ms)
This function erases the color of one pixel at a time. It has three input parameters: strip represents the Neopixel object, color represents the color to be erased, and wait_ms represents the erasure interval. The default is 50ms. For example, colorWipe(strip, Color(255,0,0),20) means that the LED0 is red first, wait for 20ms, and then the LED1 is also red, until all four LEDs are lit and red.

LED Show

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

Run Program

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

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

1.If not, execute the cd command:

```
cd ~/Freenove_Tank_Robot_Kit_for_Raspberry_Pi/Code/Server
```

2.Run Led.py:

```
sudo python Led.py
```

```
pi@raspberrypi: ~/Freenove_Tank_R...ot_Kit_for_Raspberry_Pi/Code/Server
File  Edit  Tabs  Help
pi@raspberrypi:~ $ cd ~/Freenove_Tank_Robot_Kit_for_Raspberry_Pi/Code/Server
pi@raspberrypi:~/Freenove_Tank_Robot_Kit_for_Raspberry_Pi/Code/Server $ sudo python
Led.py
Program is starting ...
Chaser animation
Rainbow animation
^Cpi@raspberrypi:~/Freenove_Tank_Robot_Kit_for_Raspberry_Pi/Code/Server $
```

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      = 4      # Number of LED pixels.
```

```
LED_PIN        = 18      # GPIO pin connected to the pixels (18 uses PWM!).
LED_FREQ_HZ    = 800000  # LED signal frequency in hertz (usually 800khz)
LED_DMA        = 10      # DMA channel to use for generating signal (try 10)
LED_BRIGHTNESS = 255     # Set to 0 for darkest and 255 for brightest
LED_INVERT     = False   # True to invert the signal (when using NPN transistor level shift)
LED_CHANNEL    = 0       # set to '1' for GPIOs 13, 19, 41, 45 or 53
# Define functions which animate LEDs in various ways.
class Led:
    def __init__(self):
        self.ORDER = "GRB"  #Control the sending order of color data
        # 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()
    def LED_TYPR(self,order,R_G_B):
        B=R_G_B & 255
        G=R_G_B >> 8 & 255
        R=R_G_B >> 16 & 255
        Led_type=["GRB","GBR","RGB", "RBG","BRG","BGR"]
        color =
[Color(G,R,B),Color(G,B,R),Color(R,G,B),Color(R,B,G),Color(B,R,G),Color(B,G,R)]
        if order in Led_type:
            return color[Led_type.index(order)]
    def colorWipe(self,strip, color, wait_ms=50):
        """Wipe color across display a pixel at a time."""
        color=self.LED_TYPR(self.ORDER,color)
        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."""
        color=self.LED_TYPR(self.ORDER,color)
        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."""
```

```
48          if pos<0 or pos >255:
49              r=g=b=0
50          elif pos < 85:
51              r=pos * 3
52              g=255 - pos * 3
53              b=0
54          elif pos < 170:
55              pos -= 85
56              r=255 - pos * 3
57              g=0
58              b=pos * 3
59          else:
60              pos -= 170
61              r=0
62              g=pos * 3
63              b=255 - pos * 3
64          return self.LED_TYPR(self.ORDER,Color(r,g,b))
65      def rainbow(self,strip, wait_ms=20, iterations=1):
66          """Draw rainbow that fades across all pixels at once."""
67          for j in range(256*iterations):
68              for i in range(self.strip.numPixels()):
69                   self.strip.setPixelColor(i, self.wheel((i+j) & 255))
70              self.strip.show()
71              time.sleep(wait_ms/1000.0)
72      def rainbowCycle(self,strip, wait_ms=20, iterations=5):
73          """Draw rainbow that uniformly distributes itself across all pixels."""
74          for j in range(256*iterations):
75              for i in range(self.strip.numPixels()):
76                  self.strip.setPixelColor(i, self.wheel((int(i * 256 / self.strip.numPixels())
    + j) & 255))
77              self.strip.show()
78              time.sleep(wait_ms/1000.0)
79      def theaterChaseRainbow(self,strip, wait_ms=50):
80          """Rainbow movie theater light style chaser animation."""
81          for j in range(256):
82              for q in range(3):
83                  for i in range(0, self.strip.numPixels(), 3):
84                      self.strip.setPixelColor(i+q, self.wheel((i+j) % 255))
85                  self.strip.show()
86                  time.sleep(wait_ms/1000.0)
87                  for i in range(0, strip.numPixels(), 3):
88                      strip.setPixelColor(i+q, 0)
89  led=Led()
90  # Main program logic follows:
```

```
91  if __name__ == '__main__':
92      print ('Program is starting ... ')
93      try:
94          while True:
95              print ("Chaser animation")
96              led.colorWipe(led.strip, Color(255,0, 0))  # Red wipe
97              led.colorWipe(led.strip, Color(0, 255, 0))  # Green wipe
98              led.colorWipe(led.strip, Color(0, 0, 255))  # Blue wipe
99              led.theaterChaseRainbow(led.strip)
100             print ("Rainbow animation")
101             led.rainbow(led.strip)
102             led.rainbowCycle(led.strip)
103             led.colorWipe(led.strip, Color(0,0,0),10)
104     except KeyboardInterrupt:  # When 'Ctrl+C' is pressed, the child program destroy() will be
105 executed.
106         led.colorWipe(led.strip, Color(0,0,0),10)
```

Reference

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

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

strip.show()

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

wheel(pos)

Generate rainbow colors in range of 0-255.

LED_TYPR(self,order,R_G_B)

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

theaterChaseRainbow(strip, wait_ms)

The function is used to make 4 LEDs show one color at the same time, and change to various colors to make a **blink**. The blinking interval is wait_ms, and its default value is 50ms.

rainbow(strip, wait_ms,)

This function achieves the effect of rainbow **breathing**. It makes 4 LEDs display **same** color simultaneously, and then change them all into various colors like breathing. The interval is wait_ms. The default value is 20ms.

rainbowCycle(strip, wait_ms)

This function also achieves the effect of rainbow **breathing**. but unlike rainbow(), it makes 4 LEDs to display **different** colors at the same time, and then change them into various color separately. The interval is wait_ms. The default value is 20ms.

Result analysis

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

Chasing animation: first let the 4 LEDs light red one by one in turn, then green and blue. Interval is 50ms between two LED, so the LED will display a round of red, then a round of green, and the last round of blue, like chasing. Then let the LEDs blink with different colors with an interval of 50ms, rendering a tense atmosphere, thus completing the chase animation.

Rainbow animation: The effect of the rainbow is different from the effect of blinking. The blinking is to make the LED on, off, on, and off. And the rainbow is to make LED on all the time, and switch between different colors, and the interval is shorter than the blinking. First, make the 4 LEDs display one color at the same time and then change the color with intervals of 20ms. And then make the 4 LEDs display different colors at the same time, and then change the color to produce another rainbow effect.

Servo

Run program

Enter the following commands in the terminal to test servos.

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

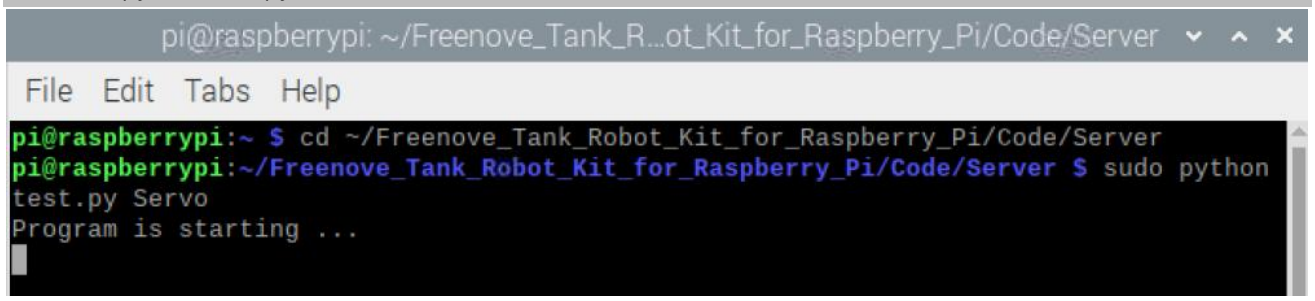
```
pi@raspberrypi:~/Freenove_Tank_Robot_Kit_for_Raspberry_Pi/Code/Server $
```

1.If not, execute the cd command:

```
cd ~/Freenove_Tank_Robot_Kit_for_Raspberry_Pi/Code/Server
```

2.Execute Servo.py command:

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



The servo will repeat the following: servo 0 is closed gradually, servo 1 from top to bottom, and then from bottom to top, servo 0 is opened gradually. You can press Ctrl + C to end the program.

When testing servo steering gear, you need to observe whether servo 0 is closed to the minimum, whether servo 1 hits the ground during the descent,

When servo 0 is just closed to the minimum and servo 1 drops close to the ground without touching the ground, it means that your servo 0 and servo 1 are properly installed.

When other conditions occur, please check that the installation steps of the relevant servo are correct.

For details, see Step 24 in Chapter 2.

The code is as below:

```
1  from servo import *
2  servo=Servo()
3  def test_Servo():
4      try:
5          while True:
6              for i in range(90,145,1):
7                  servo.setServoPwm('0',i)
8                  time.sleep(0.01)
9              for i in range(140,95,-1):
10                 servo.setServoPwm('1',i)
11                 time.sleep(0.01)
12             for i in range(95,140,1):
```

```
13                  servo.setServoPwm('1',i)
14                  time.sleep(0.01)
15              for i in range(145,90,-1):
16                  servo.setServoPwm('0',i)
17                  time.sleep(0.01)
18      except KeyboardInterrupt:
19          servo.setServoPwm('0',90)
20          servo.setServoPwm('1',140)
21          print ("\nEnd of program")
```

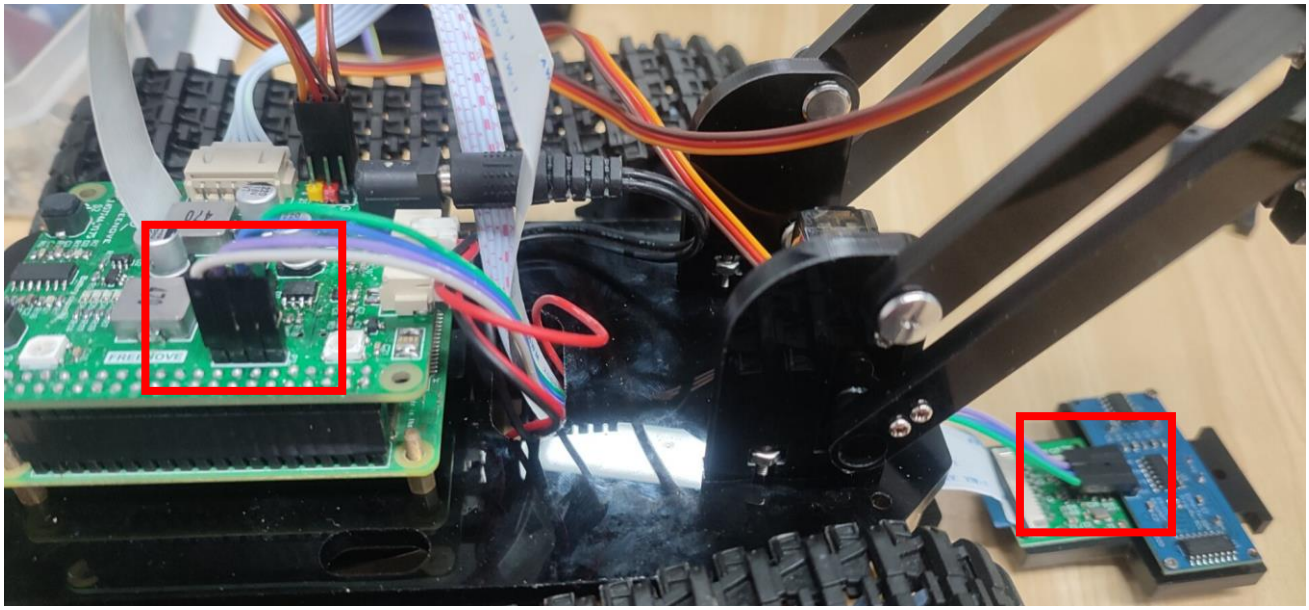
Reference

setServoPwm(Servo,angle)

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

Ultrasonic module

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



When connecting the ultrasonic module, you should keep the silk screen of the ultrasonic module and the smart car board consistent. Vcc should be connected to 5V, Trig to TRIG, Echo to ECHO, and Gnd to GND. If the connection is wrong, for example, if Vcc is connected to GND, and Gnd is connected to 5V, it will cause the damage to ultrasonic module. After the wiring is completed, you can start testing.

Run program

Enter following command in the terminal:

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

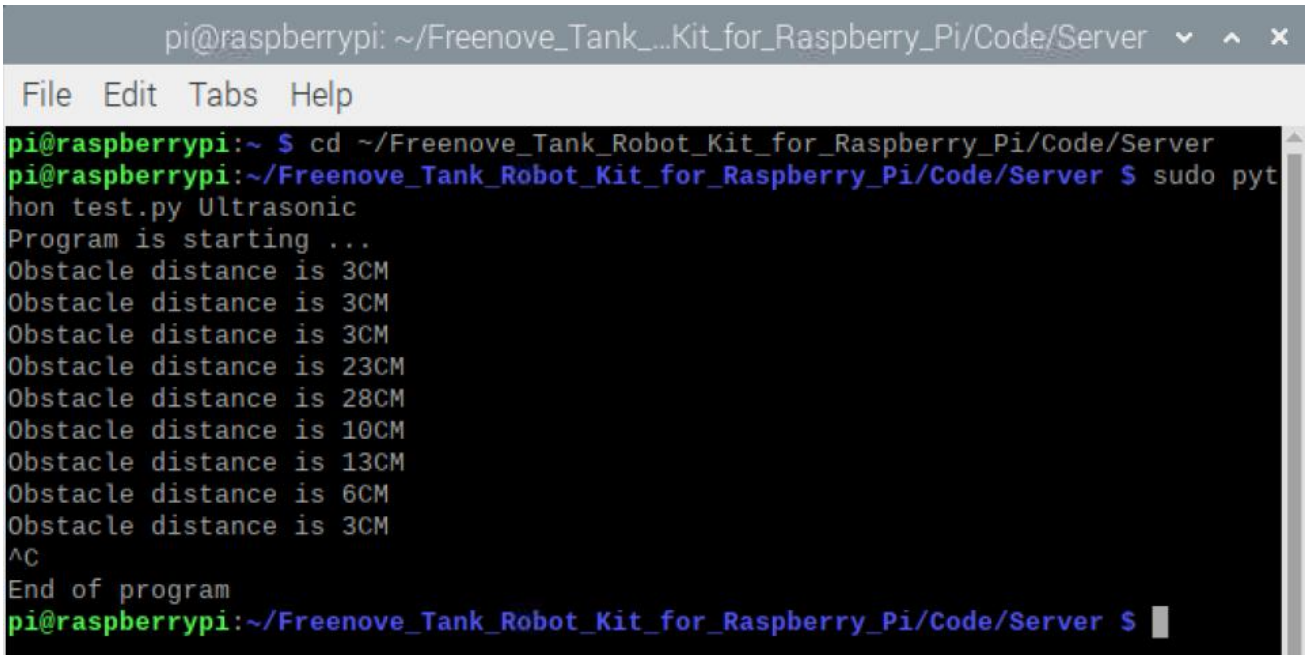
```
pi@raspberrypi:~/Freenove_Tank_Robot_Kit_for_Raspberry_Pi/Code/Server $
```

1. If not, execute the cd command:

```
cd ~/Freenove_Tank_Robot_Kit_for_Raspberry_Pi/Code/Server
```

2. Execute test.py command:

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



Result:

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

The code is as below:

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

Reference

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

Camera

Next let us connect the camera to smart car board.

First **turn off S1** (Power Switch) and **S2**, **shut down Raspberry Pi** and disconnect power cable. If the data cable is used to power the Raspberry Pi, disconnect the data cable and install the CSI camera to the Raspberry Pi camera interface when the Raspberry Pi is powered off.

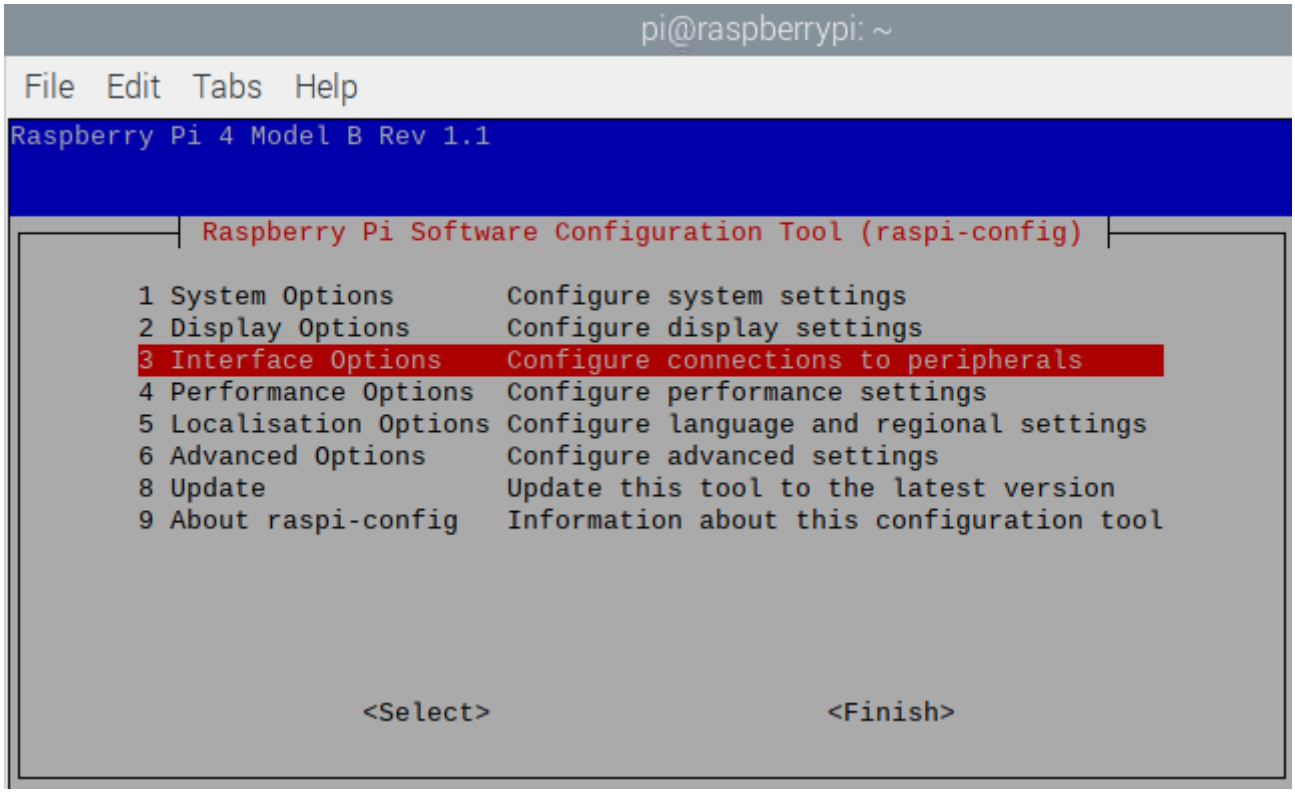
The CSI camera must be connected or disconnected under no power and when Raspberry Pi is shut down, or the camera may be burned.

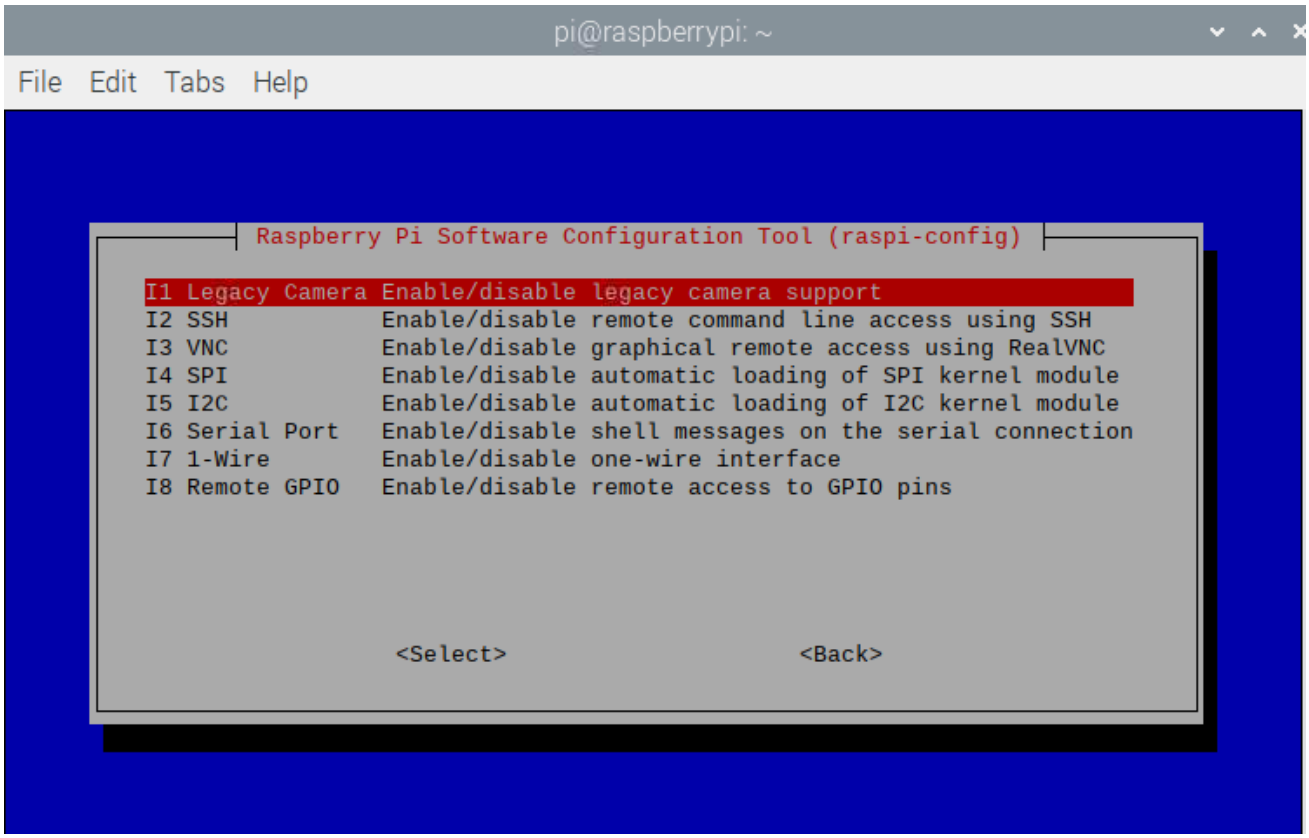
To use the camera, you need to disable legacy camera, which is disabled by default on the latest Raspberry Pi OS. If it is not disabled, please do it as below.

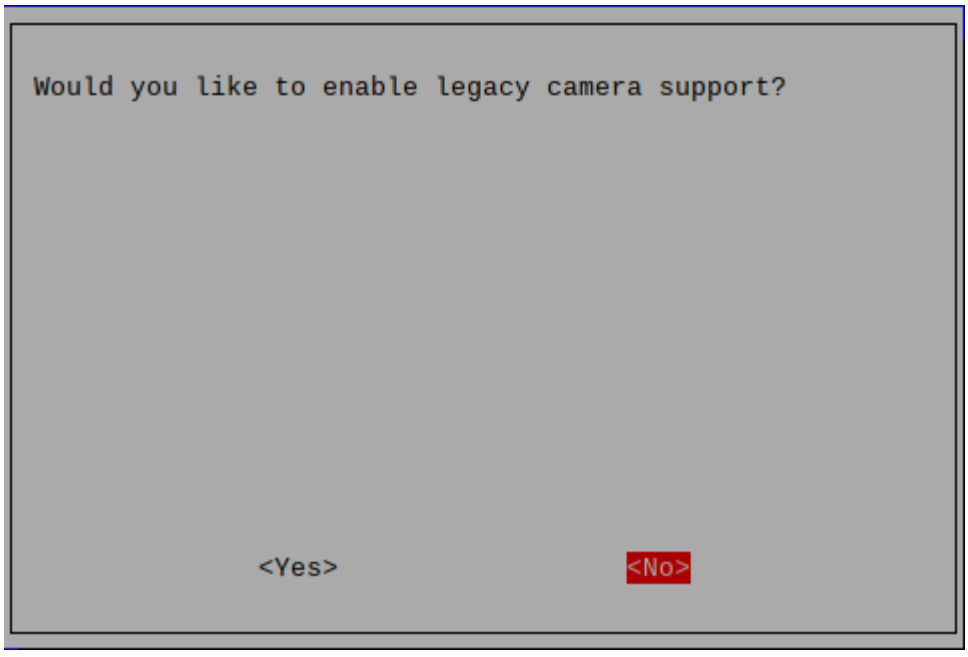
Enter the following command. Choose **Interface Options→ Legacy Camera →No→OK→Finish,**and then restart the Raspberry Pi.

```
sudo raspi-config
```

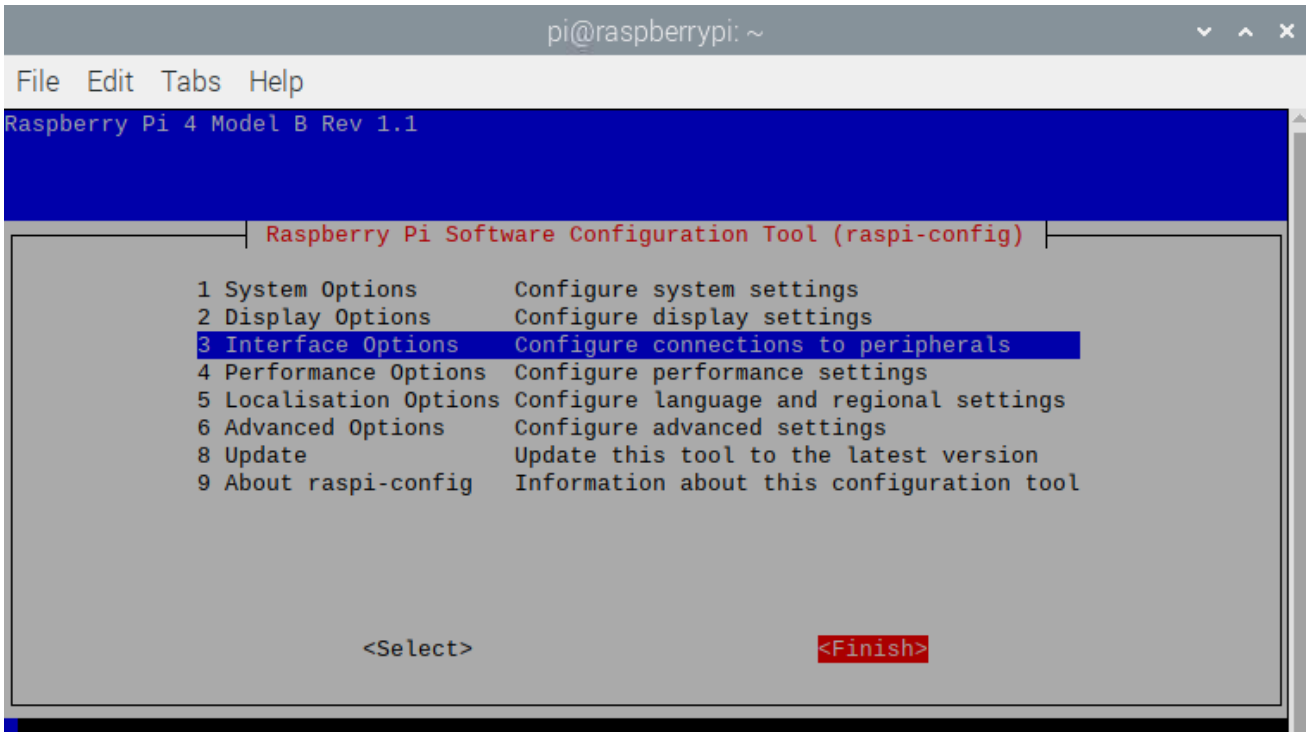
```
pi@raspberrypi:~ $ sudo raspi-config
```

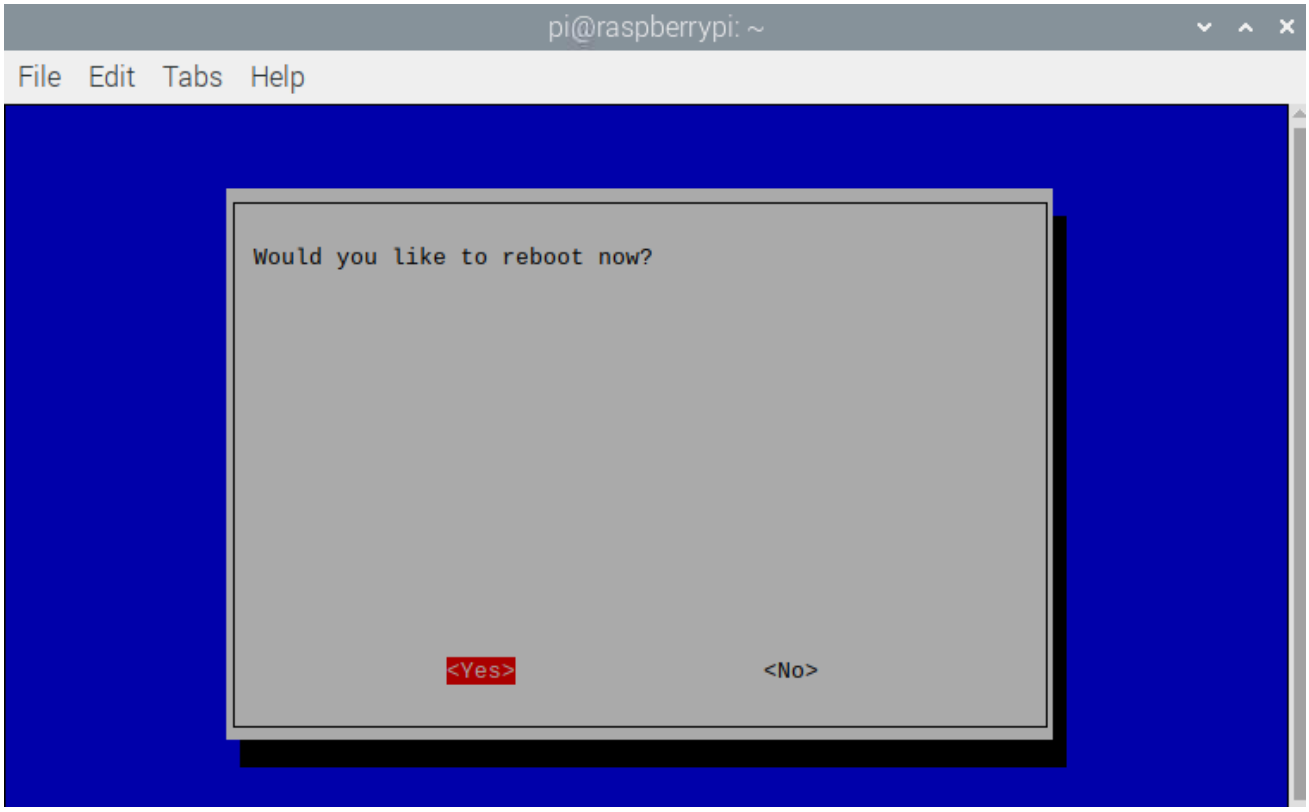



pi@raspberrypi: ~
File Edit Tabs Help
Raspberry Pi Software Configuration Tool (raspi-config)
I1 Legacy Camera Enable/disable legacy camera support
I2 SSH Enable/disable remote command line access using SSH
I3 VNC Enable/disable graphical remote access using RealVNC
I4 SPI Enable/disable automatic loading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


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


Legacy camera support is disabled.
<Ok>

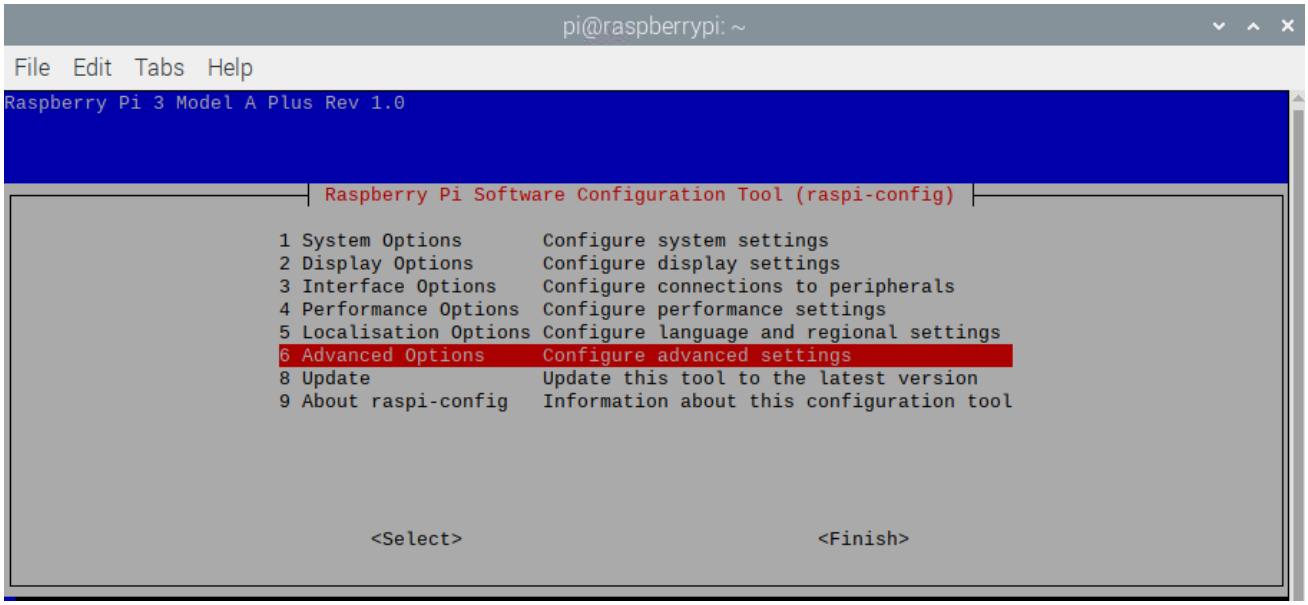

pi@raspberrypi: ~
File Edit Tabs Help
Raspberry Pi 4 Model B Rev 1.1
Raspberry Pi Software Configuration Tool (raspi-config)
1 System Options Configure system settings
2 Display Options Configure display settings
3 Interface Options Configure connections to peripherals
4 Performance Options Configure performance settings
5 Localisation Options Configure language and regional settings
6 Advanced Options Configure advanced settings
8 Update Update this tool to the latest version
9 About raspi-config Information about this configuration tool
<Select>
<Finish>

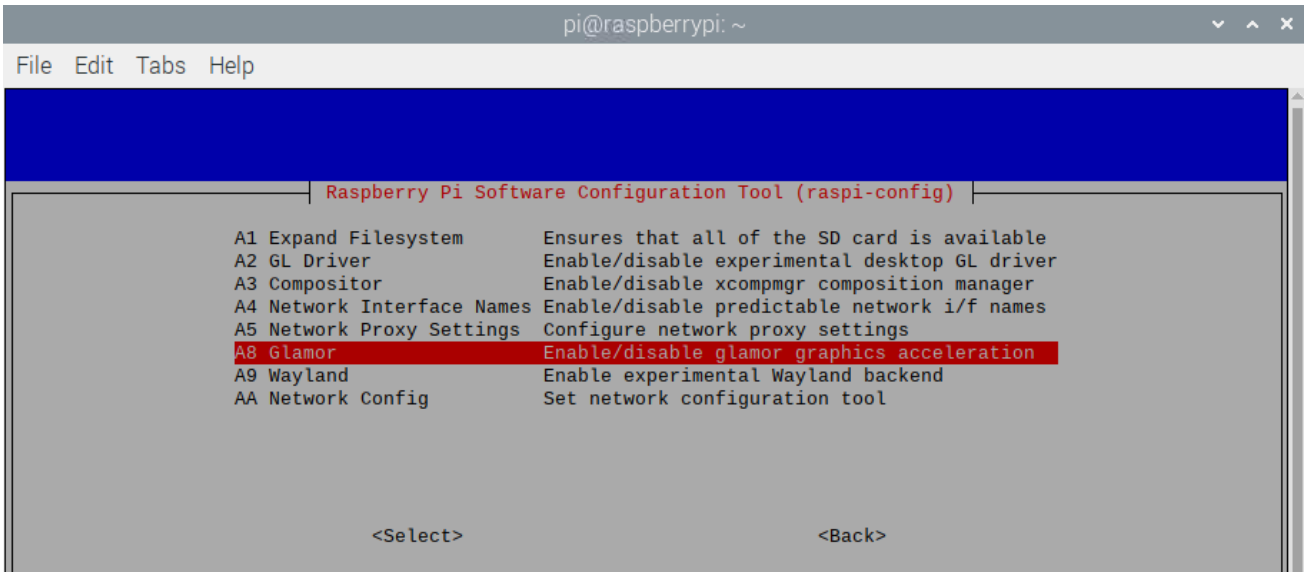


libcamera-apps does not work properly on Pi 0 to 3 devices when running the latest Bullseye images. A workaround is to open a terminal, run "**sudo raspi-config**", navigate to "**Advanced Options**" and enable "**Glamor**" graphic acceleration. Then reboot your Pi.

```
sudo raspi-config
```

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





reboot your Pi.

Run Program

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

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

1.If not, execute the cd command:

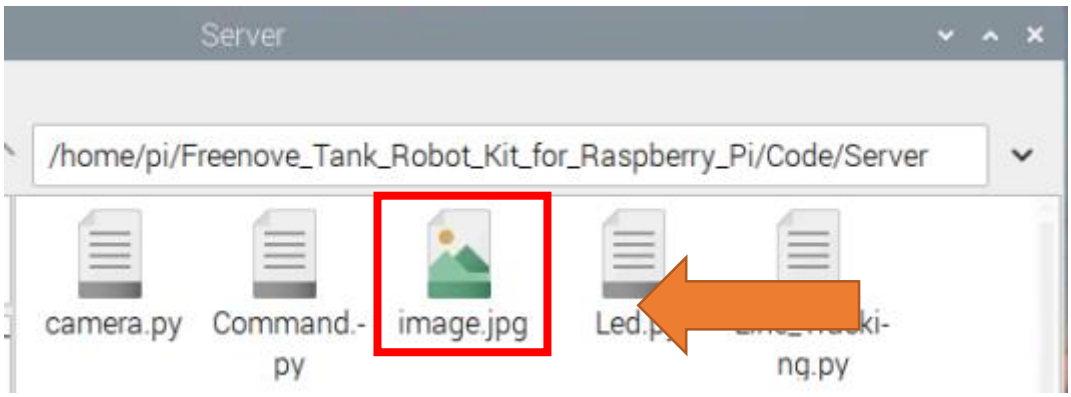
```
cd ~/Freenove_Tank_Robot_Kit_for_Raspberry_Pi/Code/Server
```

2.Run Led.py:

```
python camera.py
```

```
pi@raspberrypi: ~/Freenove_Tank_Robot_Kit_for_Raspberry_Pi/Code/Server
File  Edit  Tabs  Help
pi@raspberrypi:~ $ cd ~/Freenove_Tank_Robot_Kit_for_Raspberry_Pi/Code/Server
pi@raspberrypi:~/Freenove_Tank_Robot_Kit_for_Raspberry_Pi/Code/Server $ python camera.py
[1:06:58.425973939] [2099]  INFO Camera camera_manager.cpp:293 libcamera v0.0.1+21-7c855784
[1:06:58.551763560] [2101]  INFO RPI raspberrypi.cpp:1414 Registered camera /base/soc/i2c0mux/i2c@1/ov5647@3
6 to Unicam device /dev/media3 and ISP device /dev/media0
[1:06:58.563427284] [2099]  INFO Camera camera.cpp:1026 configuring streams: (0) 640x480-XBGR8888
[1:06:58.564453318] [2101]  INFO RPI raspberrypi.cpp:800 Sensor: /base/soc/i2c0mux/i2c@1/ov5647@36 - Selecte
d sensor format: 640x480-SGRBG10_1X10 - Selected unicam format: 640x480-pgAA
qt.qpa.xcb: QXcbConnection: XCB error: 148 (Unknown), sequence: 191, resource id: 0, major code: 140 (Unknow
n), minor code: 20
{'SensorTimestamp': 4040637190000, 'ScalerCrop': (16, 0, 2560, 1920), 'DigitalGain': 1.5220524072647095, 'Co
lourGains': (1.667592167854309, 1.2345681190490723), 'SensorBlackLevels': (1024, 1024, 1024, 1024), 'AeLocke
d': False, 'Lux': 396.4407043457031, 'FrameDuration': 30103, 'ColourCorrectionMatrix': (1.9956737756729126,
-0.6335589289665222, -0.3621244430541992, -0.34845301508903503, 1.5599479675292969, -0.21148532629013062, 0.
0021042288281023502, -0.7127978801727295, 1.7106940746307373), 'AnalogueGain': 3.375, 'ColourTemperature': 5
820, 'ExposureTime': 29968}
pi@raspberrypi:~/Freenove_Tank_Robot_Kit_for_Raspberry_Pi/Code/Server $
```

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



Part of code is as below:

```
1  import time
2  from libcamera import Transform
3  from picamera2 import Picamera2, Preview
4  picam2 = Picamera2()
5  preview_config = picam2.create_preview_configuration(main={"size": (640, 480)},
   transform=Transform(hflip=1, vflip=1))  #  Rotate the camera at an Angle of 180
6  picam2.configure(preview_config)
7  picam2.start_preview(Preview.QTGL)
8  picam2.start()
9  time.sleep(2)
10 metadata = picam2.capture_file('image.jpg')
11 print(metadata)
12 picam2.close()
```

Chapter 4 Ultrasonic Obstacle Avoidance Car

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

Description

The obstacle avoidance function of the vehicle mainly uses HC-SR04 ultrasonic module. The ultrasonic module will detect the distance between obstacles and the robot in real time, and then control the tank robot to move according to different distances.

Run program

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

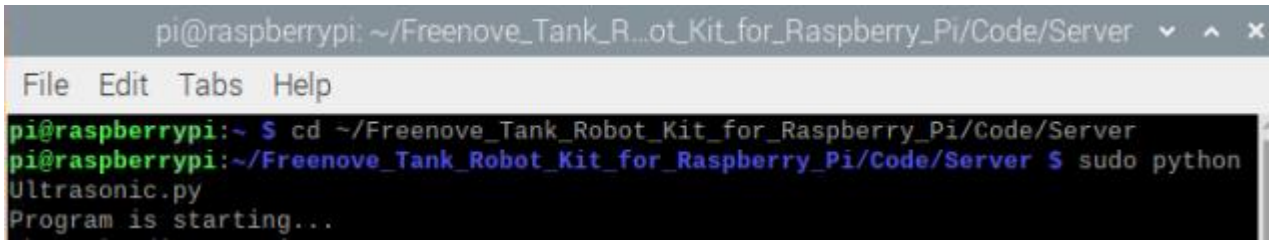
```
pi@raspberrypi:~/Freenove_Tank_Robot_Kit_for_Raspberry_Pi/Code/Server $
```

1. If not, execute the cd command:

```
cd ~/Freenove_Tank_Robot_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_motor(self,distance):
2         if(distance!=0):
3             if distance < 45 :
4                 self.PWM.setMotorModel(-1500,-1500) #Back
5                 time.sleep(0.4)
6                 self.PWM.setMotorModel(-1500,1500)  #Left
7                 time.sleep(0.2)
8             else :
9                 self.PWM.setMotorModel(1500,1500)   #Forward
10    def run(self):
11        self.PWM=Motor()
12        while True:
13            distance = self.get_distance()
```

```
            time.sleep(0.2)
            #print ("The distance is "+str(distance)+"CM")
            self.run_motor(distance)
ultrasonic=Ultrasonic()
if __name__ == '__main__':
    print ('Program is starting ... ')
    servo=Servo()
    servo.setServoPwm('0',90)
    servo.setServoPwm('1',140)
    try:
        ultrasonic.run()
    except KeyboardInterrupt:  # When 'Ctrl+C' is pressed, the child program destroy() will be executed.
        PWM.setMotorModel(0,0)
        servo.setServoPwm ('0',90)
        servo.setServoPwm ('1',140)
        print ("\nEnd of program")
```

Result analysis

When the tank robot detects that the distance between the obstacle and the car is less than 45cm, it will successively retreat for a period of time and then turn left for a period of time. When the robot detects that the distance between the obstacle and the robot is greater than or equal to 45cm, it will move forward.

In addition, you need to pay attention to the following:

In obstacle avoidance mode, only a single ultrasonic module is used for recognition, so in this mode, you need to observe the movement of the car. When the car is misidentified, please stop the car as soon as possible or pick up the car and place it in an open space. Otherwise, the car's manipulator servo could be damaged due to a collision, and may even burn your Raspberry PI expansion board or your raspberry PI.

Chapter 5 Infrared tracking automatic wrecker

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

Description

The circuit tracking function of the tank robot mainly uses infrared module. When the sensor detects a black line, the corresponding LED lights up to control the tank robot's movement according to the values of the three sensors. The tank robot detects whether there are obstacles on the road through the ultrasonic sensor while finding the line. When there are obstacles, the tank robot starts the obstacle clearing function. After the obstacle is cleared, the tank robot continues to find the line.

Run program

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

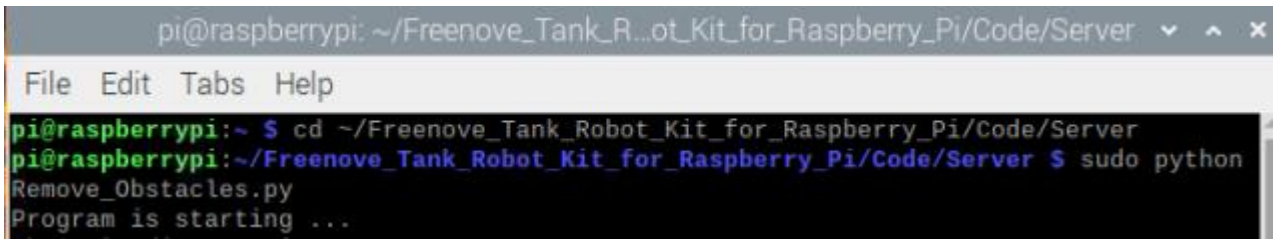
```
pi@raspberrypi:~/Freenove_Tank_Robot_Kit_for_Raspberry_Pi/Code/Server $
```

1. If not, execute the cd command:

```
cd ~/Freenove_Tank_Robot_Kit_for_Raspberry_Pi/Code/Server
```

2. Run Line_Tracking.py:

```
sudo python Remove_Obstacles.py
```



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

Before running the project, you should use black tape to make a suitable closed track, place the car on the track after the completion of the production, and then run the relevant code. After 1.5S, the car will start the function of automatically removing obstacles.

When the car is in the task, you need to observe the movement of the car, when the car runs out of the designated track, should stop the car or put the car on the original designated track.

The code is as below:

```
1 import RPi.GPIO as GPIO
2 import time
3 from Motor import *
4 from servo import *
5 from Ultrasonic import *
```

```
from Action import *
from Line_Tracking import *
class Remove_Obstacles:
    def __init__(self):
        self.IR01 = 16
        self.IR02 = 20
        self.IR03 = 21
        self.servo=Servo()
        self.PWM=Motor()
        self.distance=Ultrasonic()
        self.infrared=Line_Tracking()
        self.action=ServoMode()
        GPIO.setmode(GPIO.BCM)
        GPIO.setup(self.IR01,GPIO.IN)
        GPIO.setup(self.IR02,GPIO.IN)
        GPIO.setup(self.IR03,GPIO.IN)
    def run_Move(self):
        self.action.ServoMode('1')
        self.PWM.setMotorModel(-1500,1500)          #Left
        time.sleep(1.5)
        self.PWM.setMotorModel(0,0)                 #Stop
        self.action.ServoMode('2')
        self.PWM.setMotorModel(1500,-1500)          #Right
        time.sleep(1.4)

    def run_Line(self):
        self.LMR=0x00
        if GPIO.input(self.IR01)==True:
            self.LMR=(self.LMR | 4)
        if GPIO.input(self.IR02)==True:
            self.LMR=(self.LMR | 2)
        if GPIO.input(self.IR03)==True:
            self.LMR=(self.LMR | 1)
        if self.LMR==2:
            self.PWM.setMotorModel(1200,1200)     #Forward
        elif self.LMR==4:
            self.PWM.setMotorModel(-1500,2500)    #Left
        elif self.LMR==6:
            self.PWM.setMotorModel(-2000,4000)    #Left
        elif self.LMR==1:
            self.PWM.setMotorModel(2500,-1500)    #Right
        elif self.LMR==3:
            self.PWM.setMotorModel(4000,-2000)    #Right
        elif self.LMR==7:
```

```
            pass

    def run_Action(self):
        distance=self.distance.get_distance()
        #print ("Obstacle distance is "+str(distance)+"CM")
        self.run_Line()
        if(distance > 5.0 and distance <= 12.0):
            self.PWM.setMotorModel(0,0)          #Stop
            time.sleep(0.01)
            self.run_Move()
        elif(distance > 0.0 and distance <= 5.0):
            self.PWM.setMotorModel(-1200,-1200)  #Back

    def run(self):
        time.sleep(1.5)
        while True:
            self.run_Action()

auto_Clear=Remove_Obstacles()
# Main program logic follows:
if __name__ == '__main__':
    print ('Program is starting ... ')
    try:
        auto_Clear.run()
    except KeyboardInterrupt: # When 'Ctrl+C' is pressed,the child program will be executed.
        auto_Clear.PWM.setMotorModel(0,0) #Stop
        auto_Clear.servo.setServoPwm('0',90)
        auto_Clear.servo.setServoPwm('1',140)
        print ("\nEnd of program")
```

Result analysis

The car has the functions of line inspection and automatic obstacle clearance. The specific realization of the car's line inspection function is as follows:

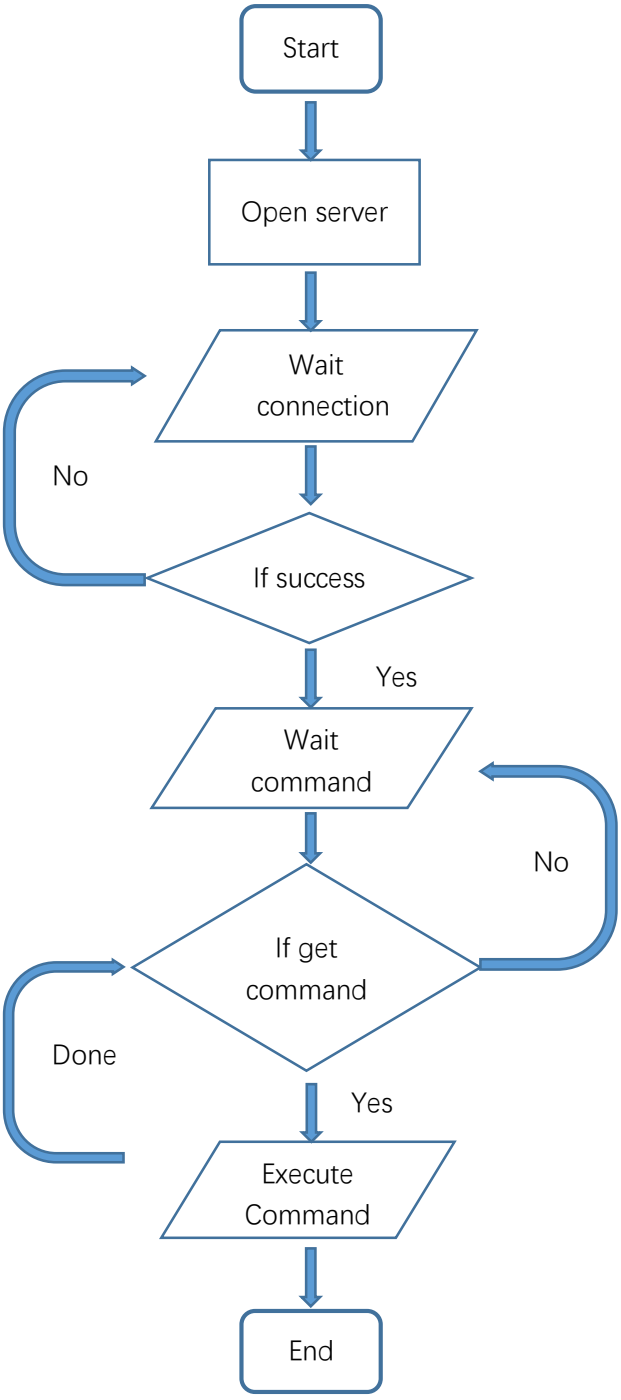
There are three sensors on the left, center and right. When the black line is detected by the sensor, it will display a high level, or it is low. When the sensor left: high, middle: low, right: low, the car gently turns left. When the sensor is left: high, middle: high, right: low, the car turns left. When the sensor left: low, medium: high, right: low, the car goes straight. When the sensor left: low, medium: low, right: high, the car gently turns right. When the sensor is left: low, middle: high, right: high, the car turns right.

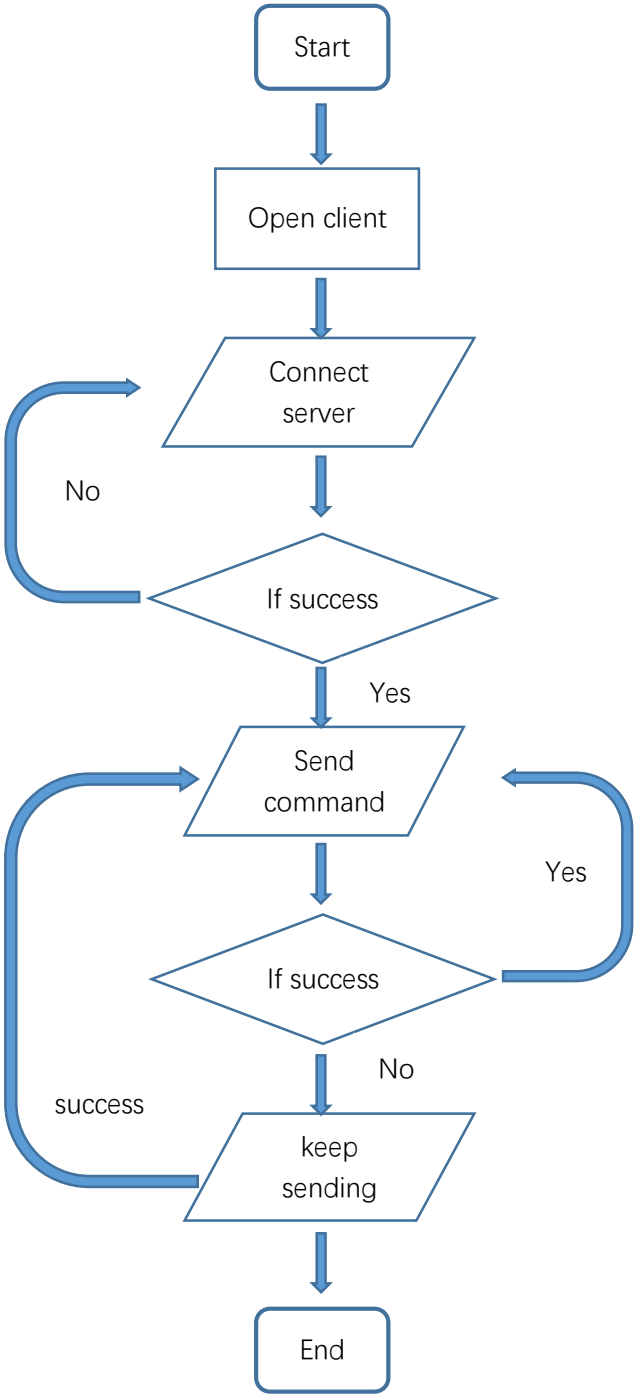
The car's automatic obstacle clearance function is as follows: under the condition of line inspection, the car detects whether there are obstacles in front of the car in real time through the ultrasonic module. When the car detects obstacles in front of the car, the car stops and removes the obstacles through the steering gear. After clearing, the car continues to perform the function of line inspection.

Chapter 6 Smart video car

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

The smart video car integrates the previous functions of obstacle avoidance, line tracking and obstacle clearing, video transmission, ball tracking, LED and so on. It consists of a server and a client, so it can be controlled remotely.





Server

The server works on the Raspberry Pi and can transmit camera data, ultrasonic data, etc. to the client, and it can also receive commands from the client.
In the Server folder, there is a server.py file which contains main server code.
get_interface_ip() is used to get IP address of the native Raspberry Pi wlan0, without manually modifying the code to set IP parameters.
StartTcpServer() is used to start the TCP service. The channel of port 5003 is mainly used to send and receive commands between the client and the server. The channel of port 8003 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:

```
def get_interface_ip(self):
        s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
        return socket.inet_ntoa(fcntl.ioctl(s.fileno(), 0x8915, struct.pack('256s',
                                "wlan0"[:15]))[20:24])
def StartTcpServer(self):
        HOST=str(self.get_interface_ip())
        self.server_socket1 = socket.socket()
        self.server_socket1.setsockopt(socket.SOL_SOCKET,socket.SO_REUSEPORT,1)
        self.server_socket1.bind((HOST, 5003))
        self.server_socket1.listen(1)
        self.server_socket = socket.socket()
        self.server_socket.setsockopt(socket.SOL_SOCKET,socket.SO_REUSEPORT,1)
        self.server_socket.bind((HOST, 8003))
        self.server_socket.listen(1)
        print('Server address: '+HOST)

def StopTcpServer(self):
        try:
            self.connection.close()
            self.connection1.close()
        except Exception ,  e:
            print( "No client connection")

def sendvideo(self):
        try:
            self.connection,self.client_address = self.server_socket.accept()
            self.connection=self.connection.makefile('rb')
        except:
            pass
```

```
30          self.server_socket.close()
31          print ("socket video connected ... ")
32          camera = Picamera2()
33          camera.configure(camera.create_video_configuration(main={"size": (400,
    300)},transform=Transform(hflip=1,vflip=1)))
34          output = StreamingOutput()
35          encoder = JpegEncoder(q=90)
36          camera.start_recording(encoder, FileOutput(output),quality=Quality.VERY_HIGH)
37          while True:
38              with output.condition:
39                  output.condition.wait()
40                  frame = output.frame
41              try:
42                  lenFrame = len(output.frame)
43                  #print("output .length:",lenFrame)
44                  lengthBin = struct.pack('<I', lenFrame)
45                  self.connection.write(lengthBin)
46                  self.connection.write(frame)
47              except Exception as e:
48                  camera.stop_recording()
49                  camera.close()
50                  print ("End transmit ... " )
51                  self.resetVideoThread()
52                  break
```

Open Server

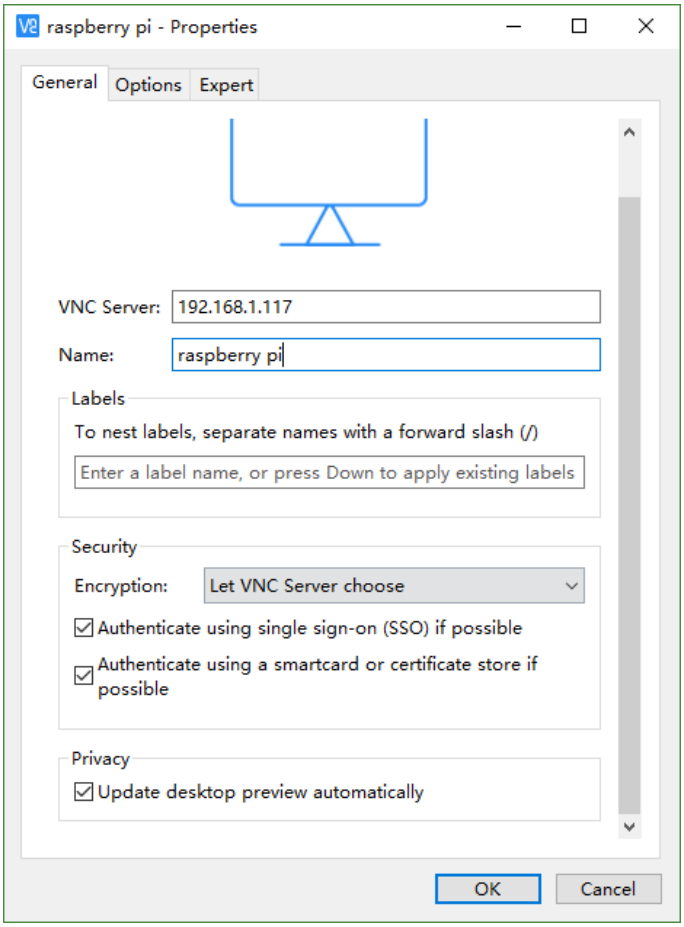
Step 1 Login Raspberry Pi via VNC viewer

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

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

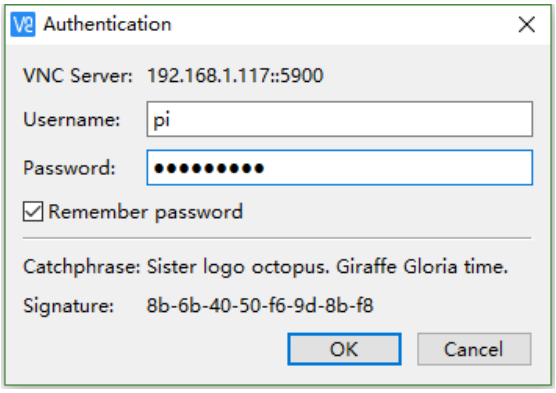
https://www.realvnc.com/en/connect/download/viewer/

After installation is completed, open VNC Viewer. And click File → New Connection. Then the interface is shown below.



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

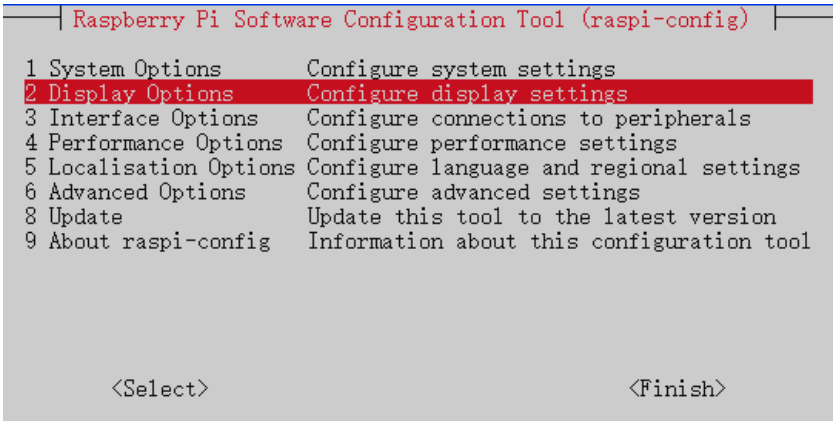
Then on the VNC Viewer panel, double-click new connection you just created, and the following dialog box pops up. Enter username: **pi** and Password: **raspberry**. And click OK.

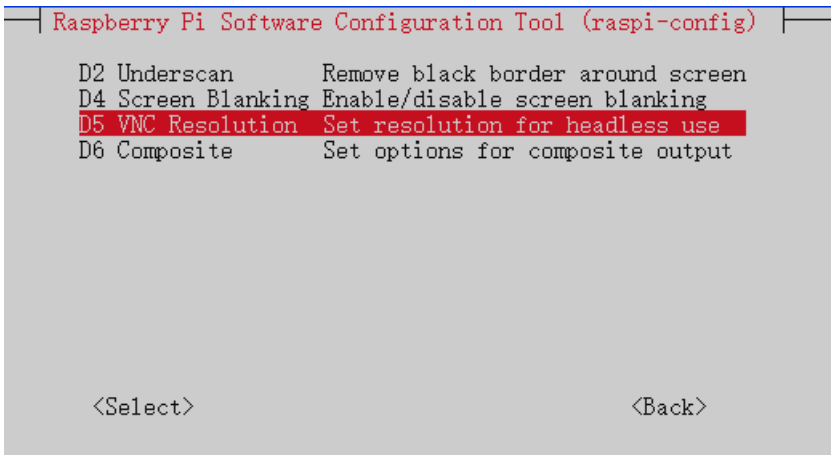


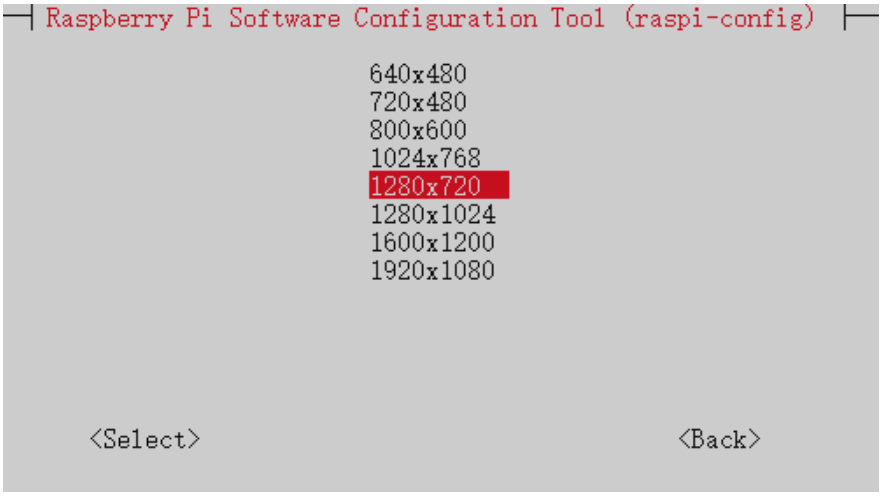
If the resolution ratio is not great or there is just a **little window**, you can set a proper resolution ratio via steps below.

```
sudo raspi-config
```

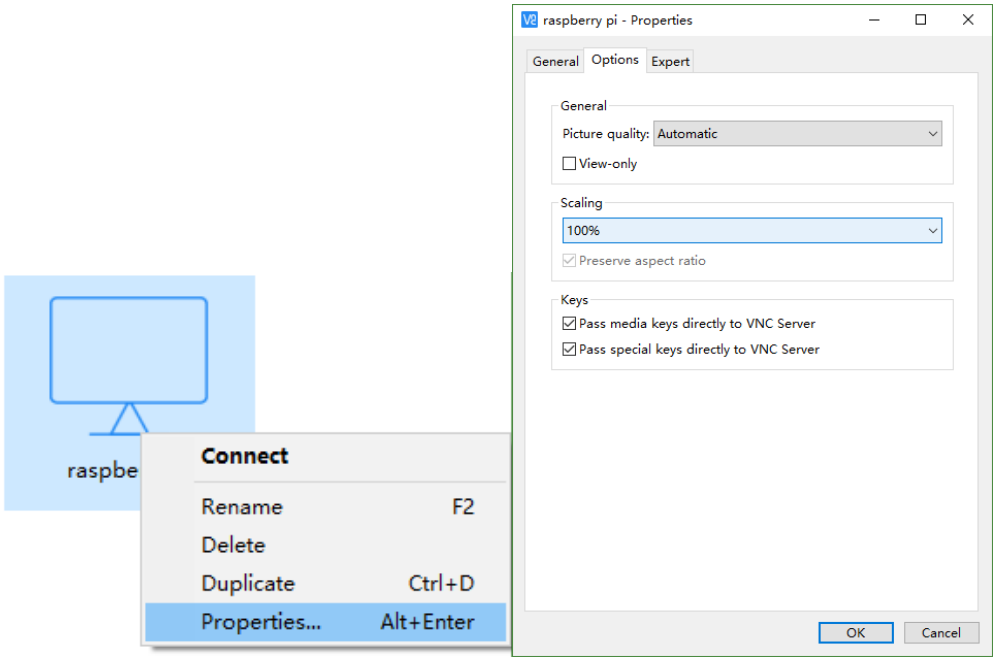
Select **Display Options→VNV Resolution→Proper resolution ratio (set by yourself)→OK→Finish→Yes**. And then reboot Raspberry Pi.







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



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

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

Step 2 Run commands

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

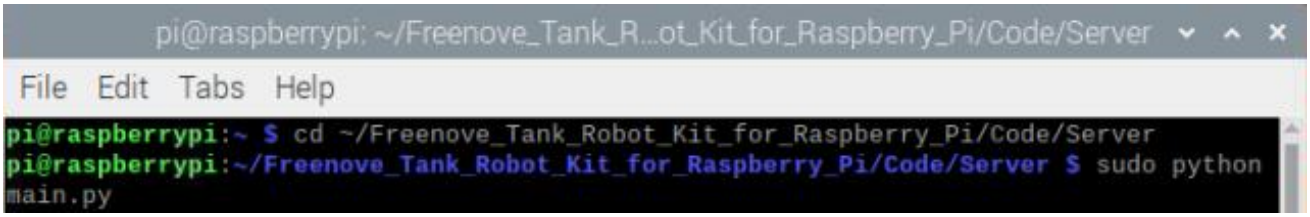
Enter the following command in the terminal.

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

```
cd ~/Freenove_Tank_Robot_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 the interface, you can also enter the commands to open the server. It is more convenient.

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

```
cd ~/Freenove_Tank_Robot_Kit_for_Raspberry_Pi/Code/Server
```

2. Run main.py:

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

or Run main,py with following command:

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

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

Sever Auto Start

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

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

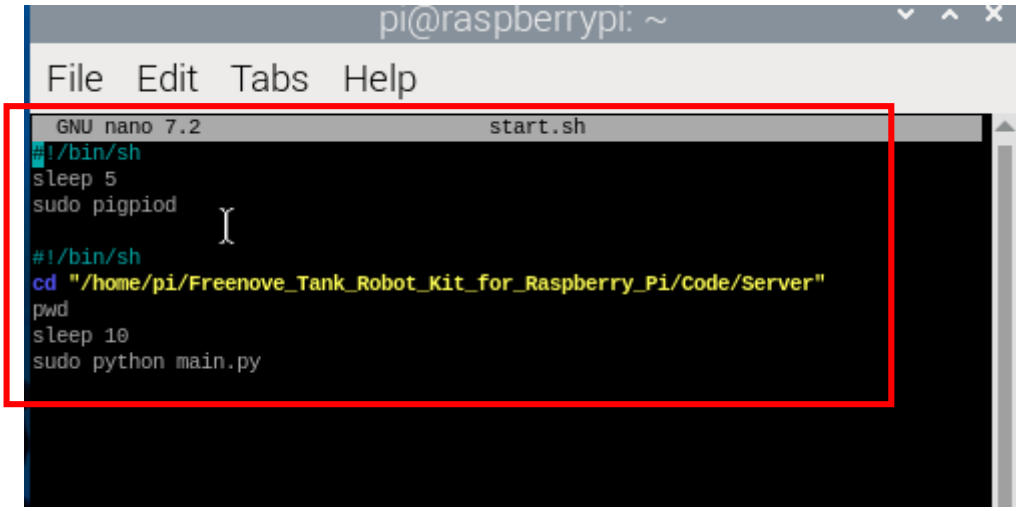
11 Open “start.sh”.

```
sudo nano start.sh
```

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

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

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



13 Modify permissions.

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

14 Enter the following command to create a directory.

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

15 create and open “start.desktop” file

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

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

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

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

17 Modify permissions.

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

18 Finally enter the following content to reboot Raspberry Pi.

```
sudo reboot
```

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

If you have already set up the "Pigpio Library" Auto Start, Or the above Settings already exist,you can skip the above steps.

Client

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

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

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

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

    def streaming(self,ip):
        stream_bytes = b' '
        try:
            self.client_socket.connect((ip, 8003))
```

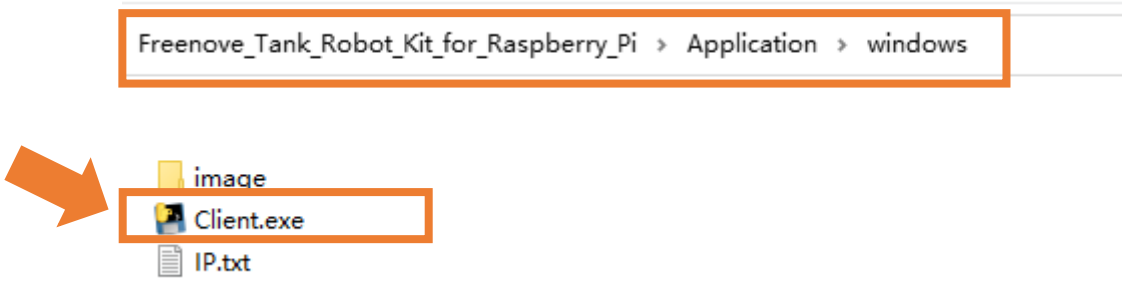
```
35              self.connection = self.client_socket.makefile('rb')
36          except:
37              #print ("command port connect failed")
38              pass
39          while True:
40              try:
41                  stream_bytes= self.connection.read(4)
42                  leng=struct.unpack('<L', stream_bytes[:4])
43                  jpg=self.connection.read(leng[0])
44                  if self.IsValidImage4Bytes(jpg):
45                      if self.video_Flag:
46                          self.image = cv2.imdecode(np.frombuffer(jpg, dtype=np.uint8),
    cv2.IMREAD_COLOR)
47                          self.video_Flag=False
48              except Exception as e:
49                  print (e)
50                  break
```

Run client on windows system

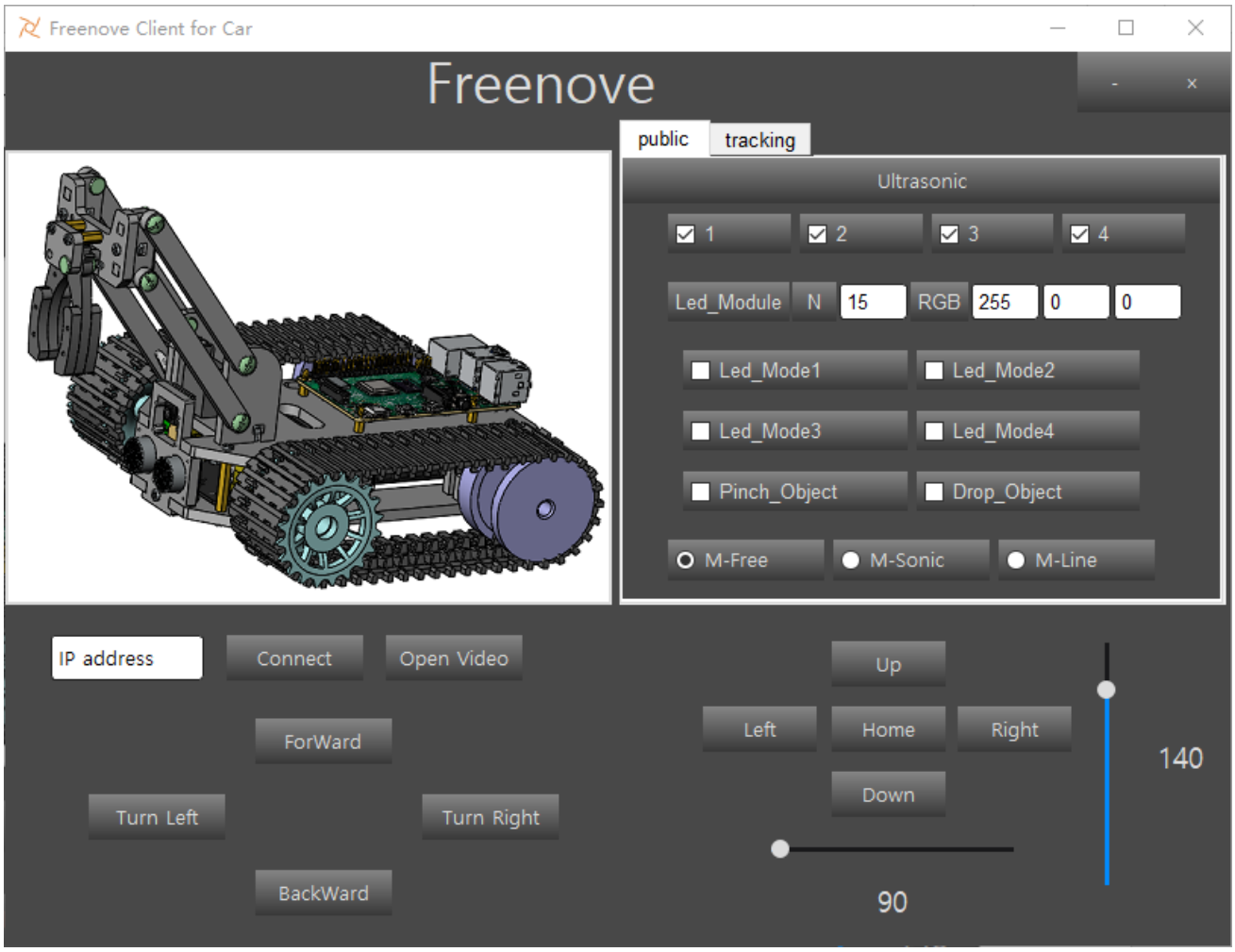
There are two ways to run Client on Windows.

Option 1 Running executable file directly

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



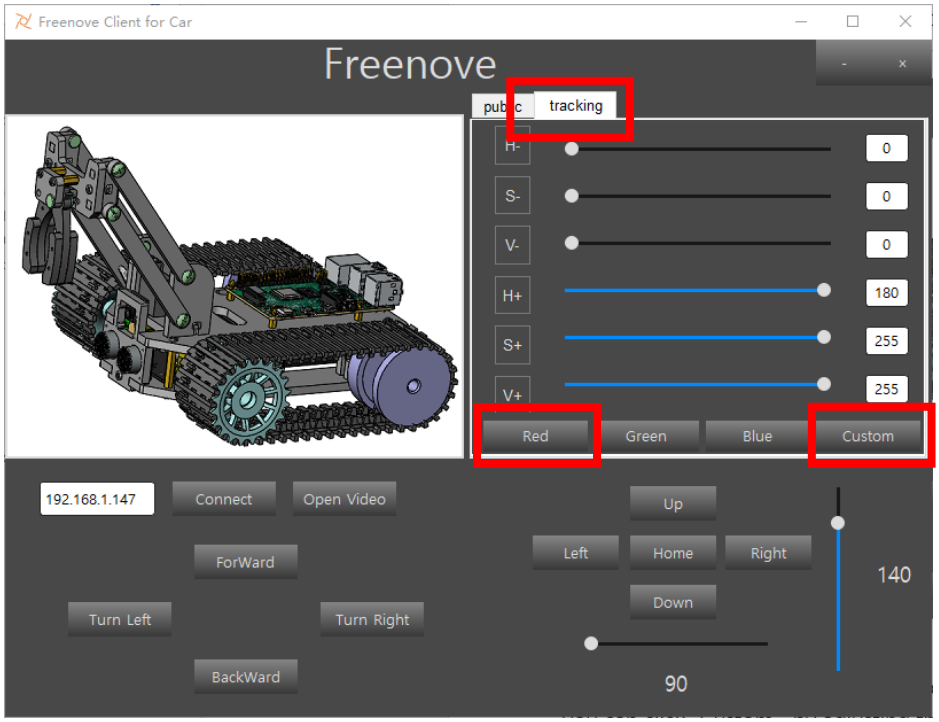
The client interface is shown as below:



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

Note: when Raspberry Pi is shut down, server will be closed. You need open server again the next time.
If pressing forward but the car moves backward, please refer to page 66 to modify the code.

The control interface of tank robot tracking ball is as follows:



Choose the color of the ball in your hand, take the Red ball as an example, when you click the button "red", into the tank robot tracking red ball control program, when your environment is poor recognition of red, you can click "Custom", by adjusting the HSV parameters to adjust the accuracy of recognition,
This recognition program is for color recognition, can track red objects not limited to red ball.

When the car motion control is abnormal, you can click the color button again to make the car stop moving.

After the color threshold is displayed, the recognition effect can be observed and the appropriate data can be selected according to the following HSV color range table to improve the recognition accuracy.
The HSV color range is listed below:

	black	gray	white	red		orange	yellow	green	cyan	blue	purple
H-	0	0	0	0	156	11	26	35	78	100	125
H+	180	180	180	10	180	25	34	77	99	124	155
S-	0	0	0	43		43	43	43	43	43	43
S+	255	43	30	255		255	255	255	255	255	255
V-	0	46	221	46		46	46	46	46	46	46
V+	46	220	255	255		255	255	255	255	255	255

Using red as an example, set HSV to the following:
[H-, S-, V-, H+, S+, V+] [0, 118, 31, 6, 255, 255]
You can adjust the value range to make the tracking better.

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

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

Install python3

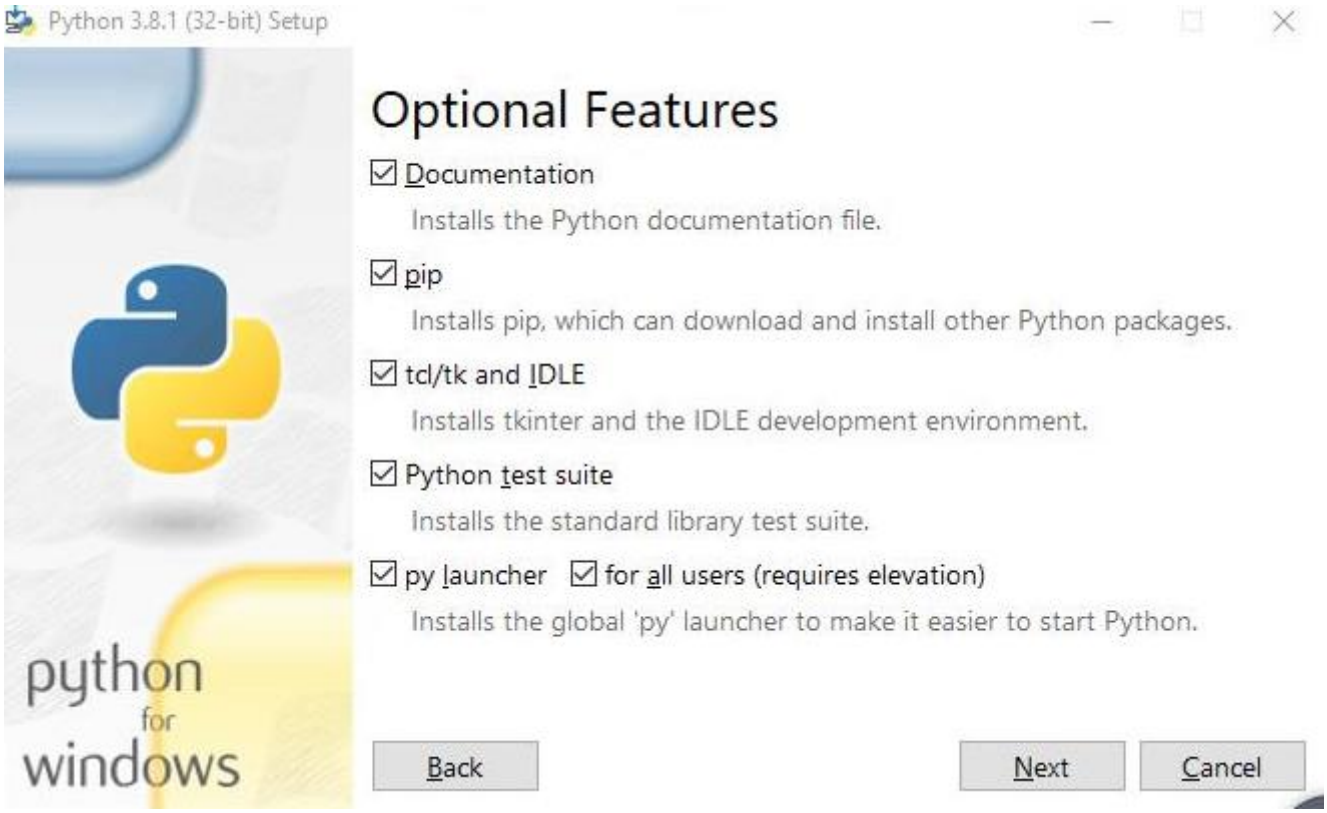
Download the installation file:
https://www.python.org/downloads/windows/

Click Latest Python 3 Release - Python 3.8.1

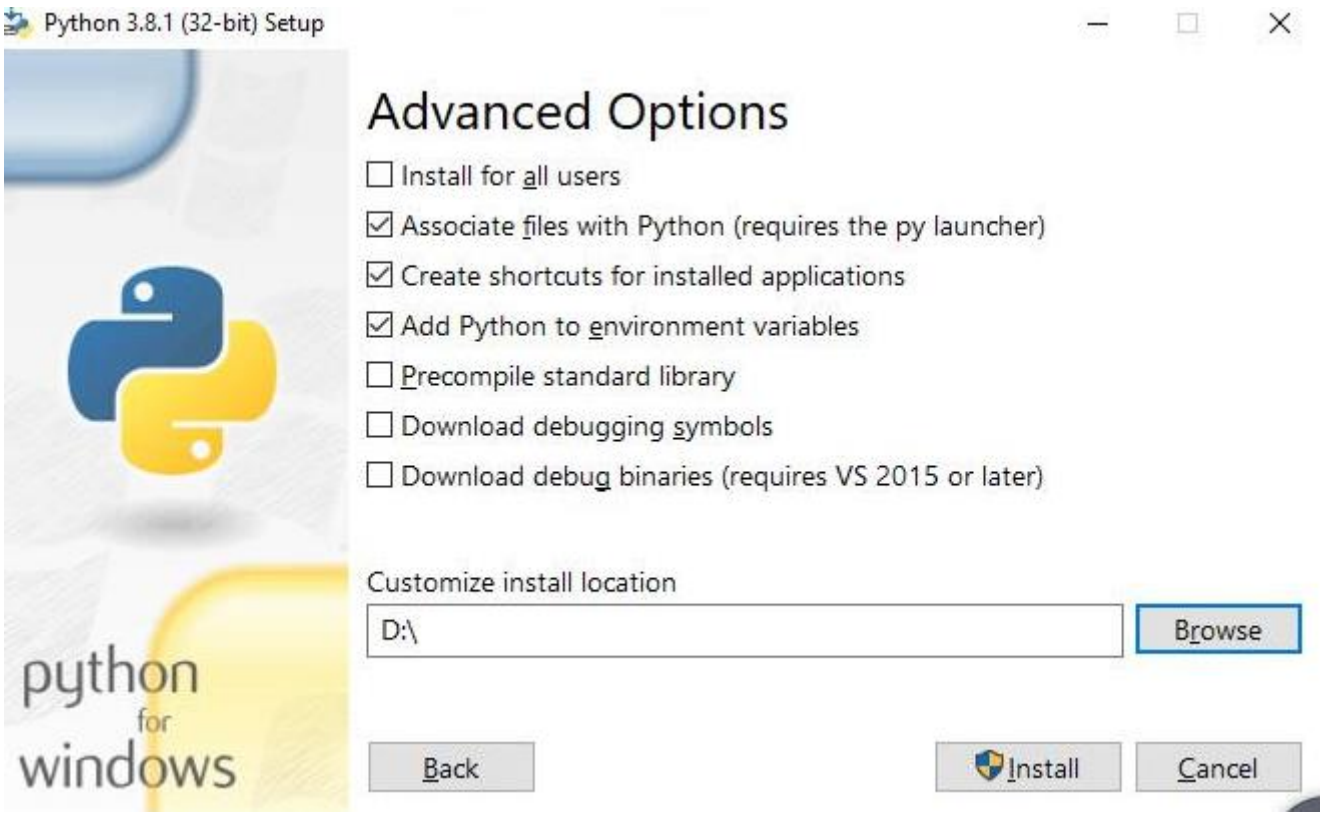
Version	Operating System	Description
Gzipped source tarball	Source release	
XZ compressed source tarball	Source release	
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later
Windows help file	Windows	
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64
Windows x86-64 executable installer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

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

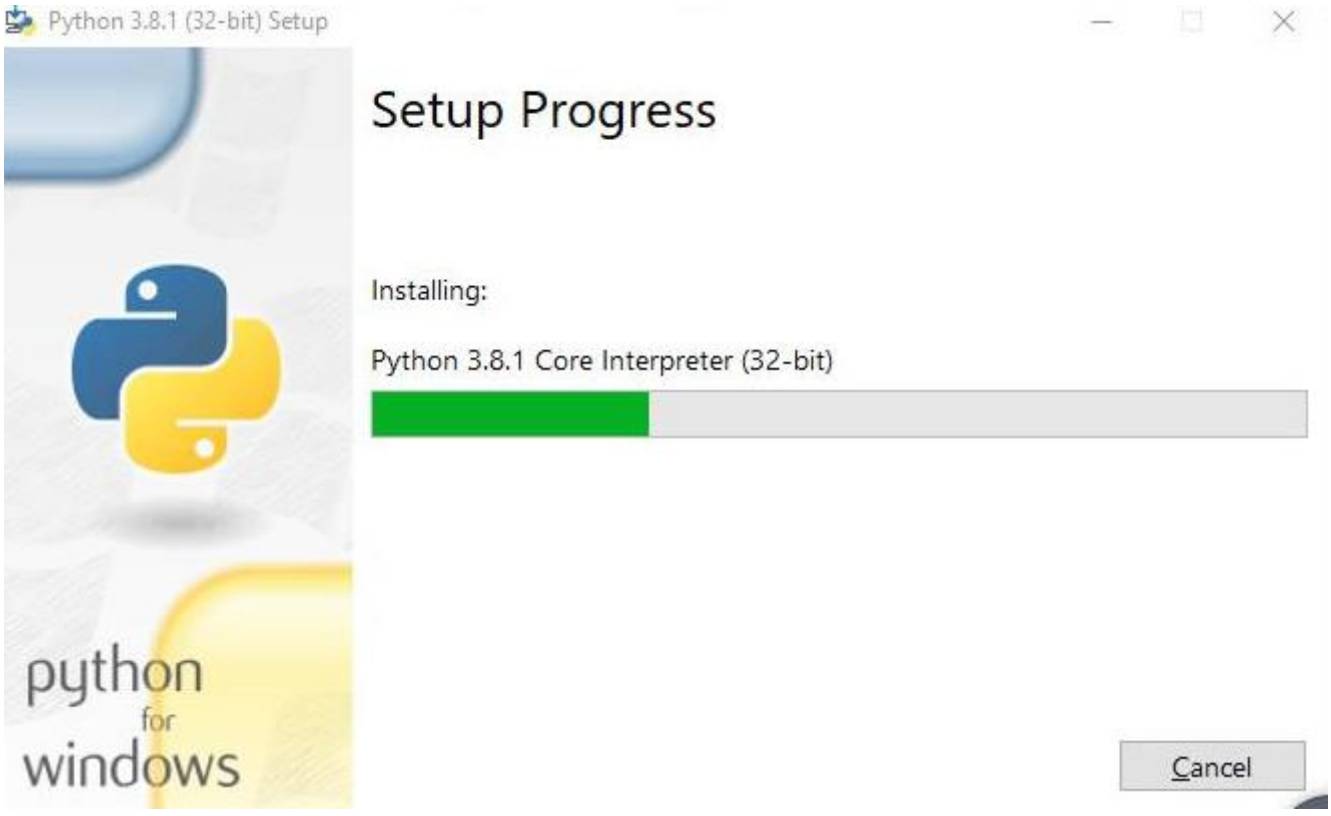
Select "**Add Python 3.8 to PATH**". You can choose other installation features.



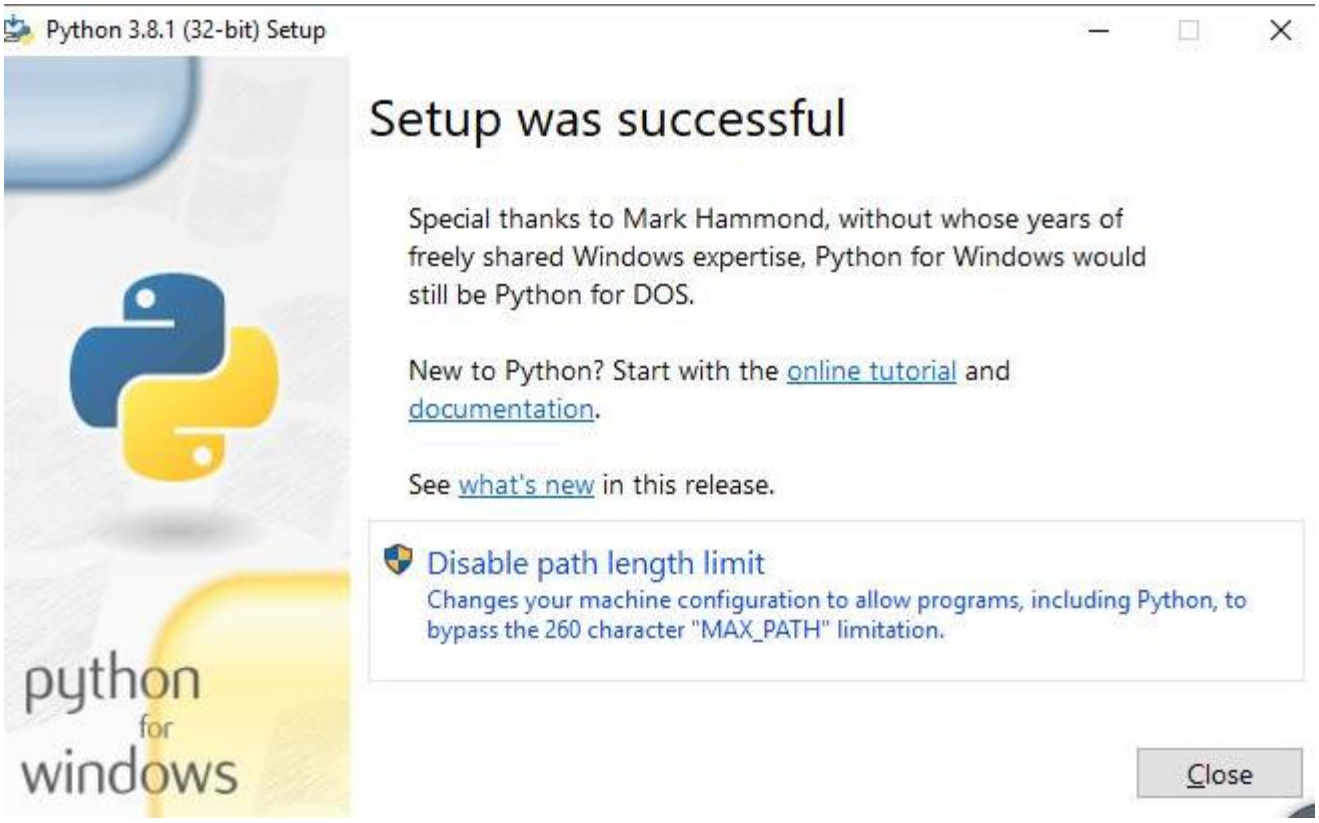
Select all options and click "Next".



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



Wait installing.



Now, installation is completed.

Install PyQt5、opencv、numpy and other libraries.

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

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

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

Then put it into D disk for example.

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

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

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

```
D:
```

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

```
cd D:\Freenove_Tank_Robot_Kit_for_Raspberry_Pi\Code
```

3. Run setup_windows.py:

```
python3 setup_windows.py
```

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

D:\>cd D:\Freenove_Tank_Robot_Kit_for_Raspberry_Pi\Code

D:\Freenove_Tank_Robot_Kit_for_Raspberry_Pi\Code>python3 setup_windows.py
```

Or enter the unzipped directory Freenove_Tank_Robot_Kit_for_Raspberry_Pi\Code\Client.
If your python3 fails to execute, you can try using python with the following command:

```
python setup_windows.py
```

And double-click **setup_client.py** or open it with python3.
Installation will take some time. Just wait patiently. For successful installation, it will prompt "All libraries installed successfully":

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

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

Open client

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

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

```
D:
```

2. Enter directory where Main.py is located:

```
cd D:\Freenove_Tank_Robot_Kit_for_Raspberry_Pi\Code\Client
```

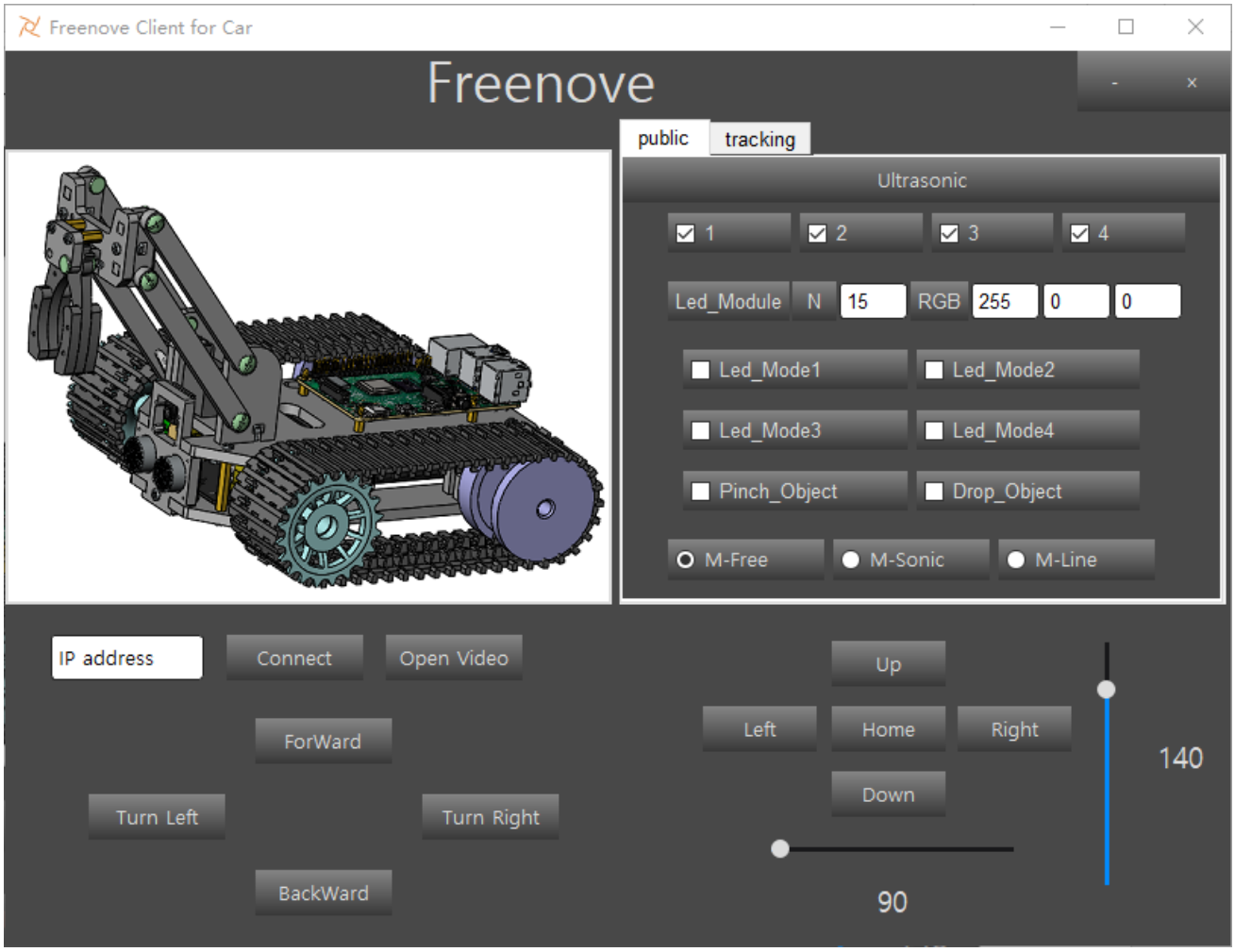
3. Run Main.py:

```
python Main.py
```

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

D:\>cd D:\Freenove_Tank_Robot_Kit_for_Raspberry_Pi\Code\Client

D:\Freenove_Tank_Robot_Kit_for_Raspberry_Pi\Code\Client>python Main.py
```

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

The client interface is shown as below:



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

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

If pressing forward but the car moves backward, please refer to page 63 to modify the code.

Control

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

The car has three work modes:

Mode	Function
M-Free (Mode1)	Free control mode
M-Sonic (Mode2)	Ultrasonic obstacle avoidance mode
M-Line (Mode3)	Infrared tracking automatically clears obstacles 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
Led_Mode 1,2,3,4	L	Switch Led Mode
The car work modes	Q	Switch the car work modes
Pinch_Object	O	Pinch Object
Drop_Object	P	Drop Object

The function of SliderBar is below:

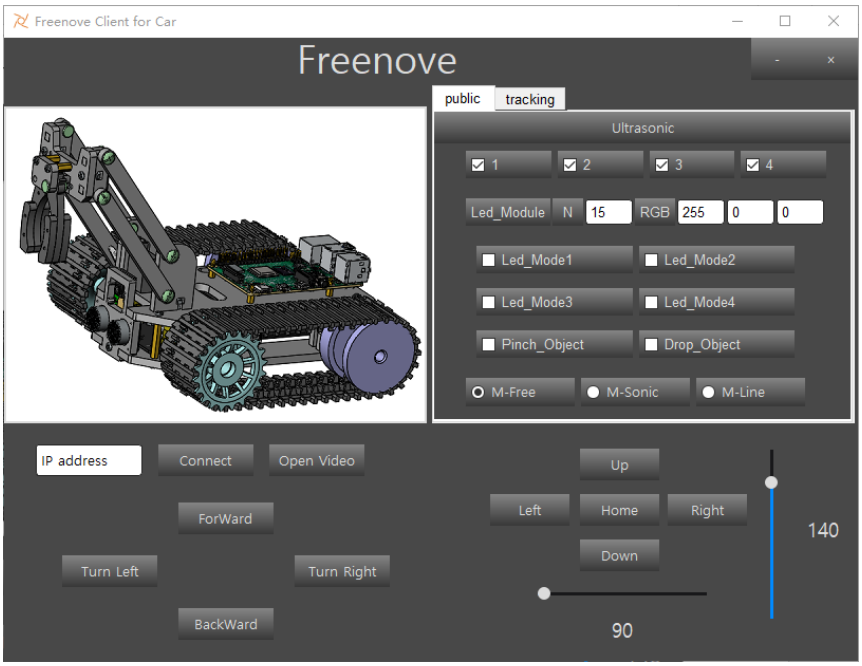
SliderBar	Function
Servo 1,2,	SliderBar Servo 1, 2 are used to slightly adjust the angle. you can slightly tune it via the SliderBar.

Other control information:

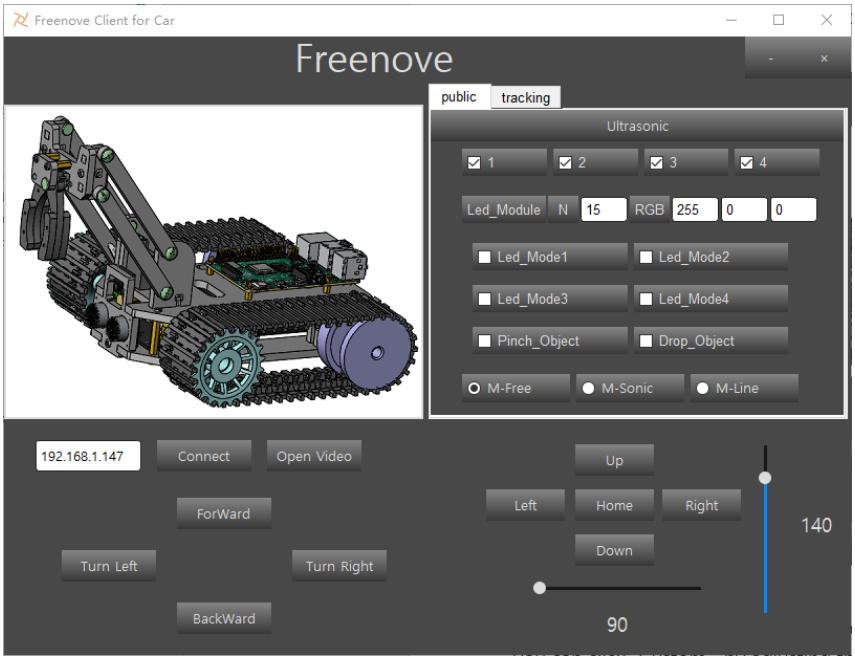
Control	Function
IP address Edit box	Enter IP address of Raspberry Pi
R,G,B Edit box	Control the color of LED selected.
Button “Ultrasonic“	Show the distance from obstacle.

When you enter the IP address for the first time, the program saves the IP address. When you close the program and open the program again, the program automatically fills in the last IP address. The IP address is saved in the “IP.txt” file.

When you run the program for the first time, the contents of the "IP.txt" file are: "IP address", then the program page is opened as follows:



After you enter the address, for example, 192.168.1.147, close the program and open the program page again as follows:



Run client on macOS system

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

Install python3

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

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

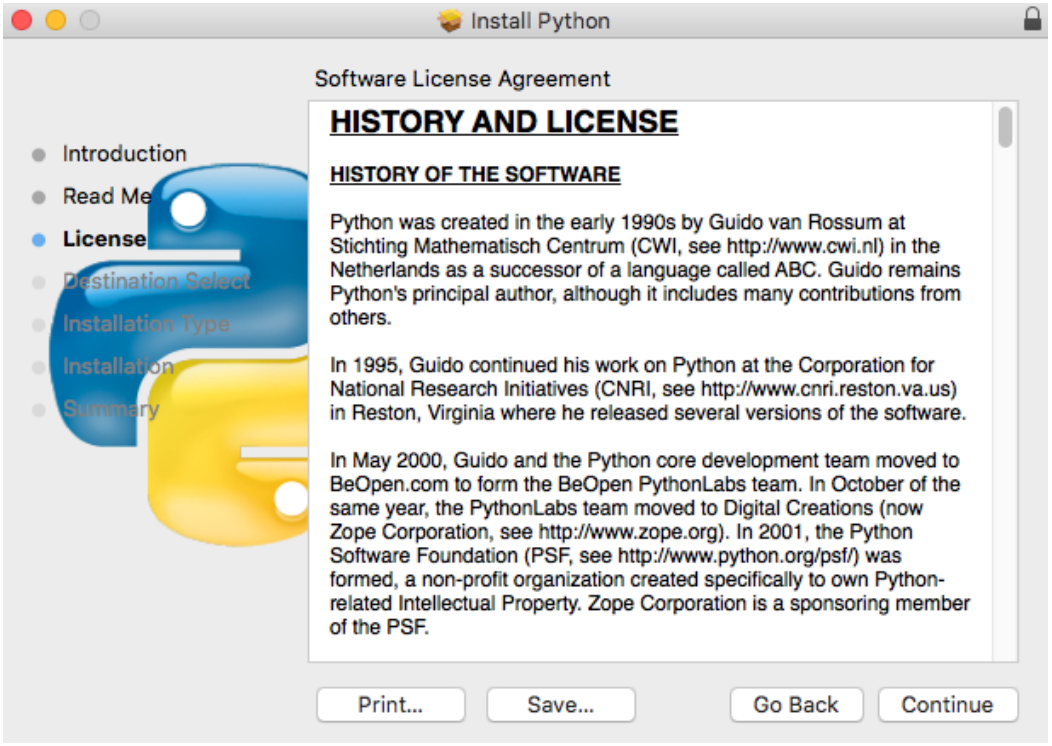
If your macOS is 11. Like 11.0, please install python 3.9.

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

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

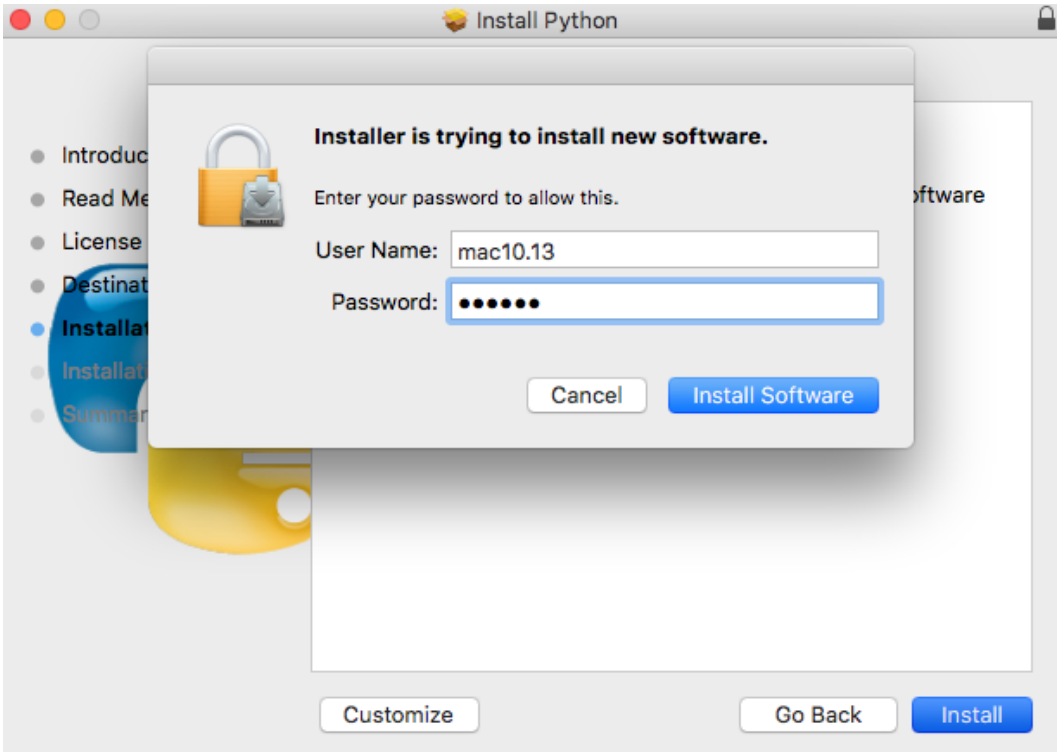
At bottom of the page, click macOS 64-bit installer and download installation package.

Click Continue.

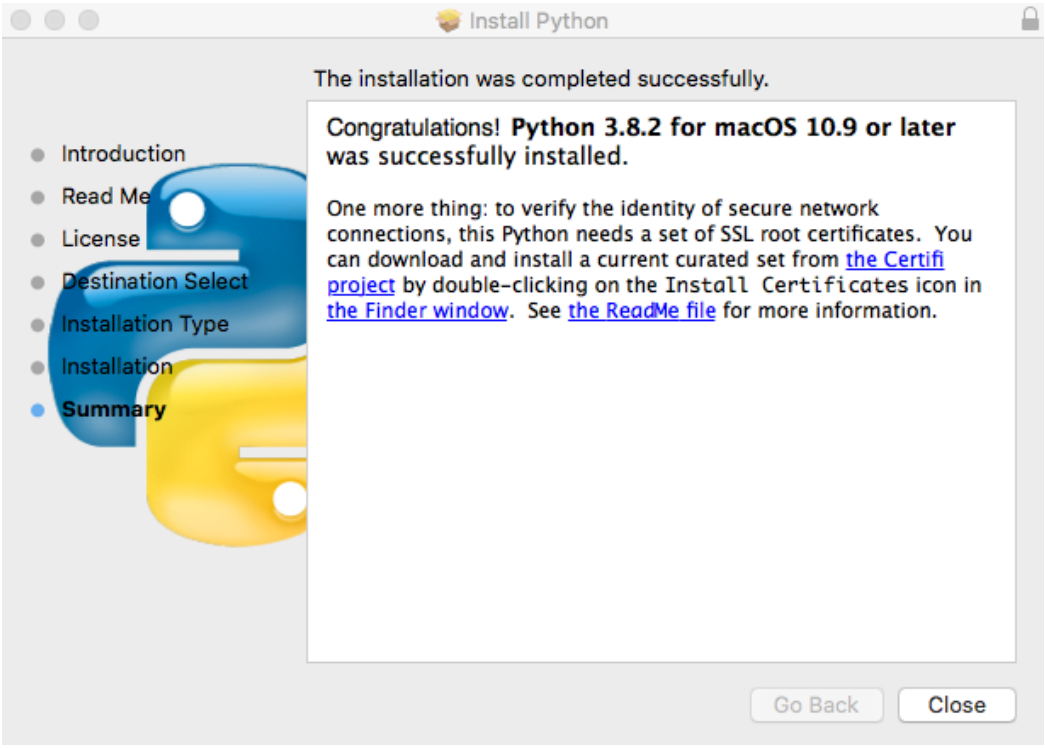


Click Continue

Click Agree.



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

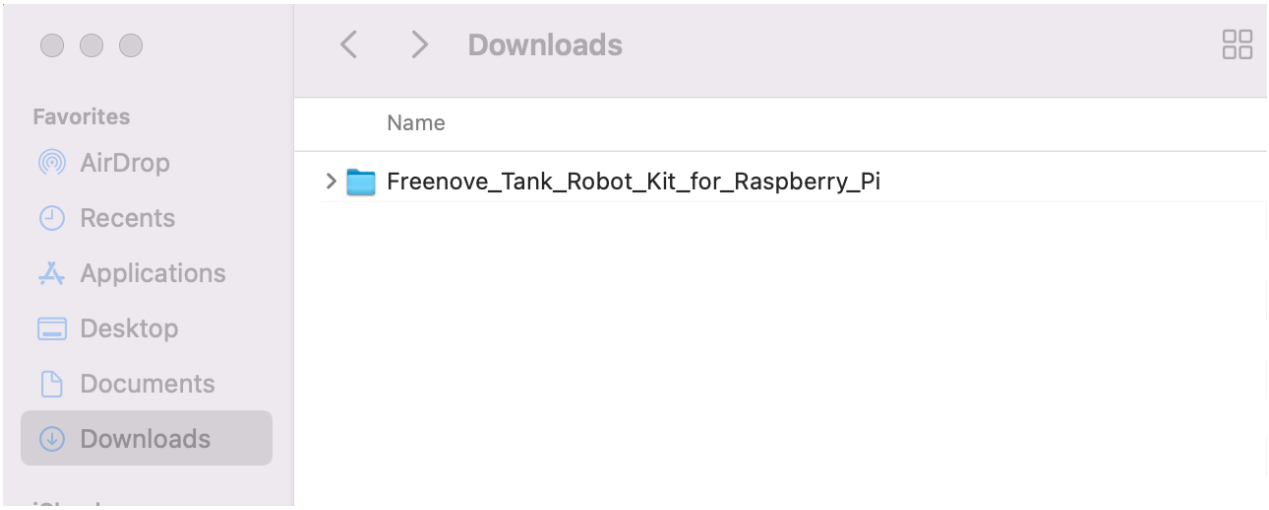


Now the installation succeeds.

You can find it in Aapplications.

Install PyQt5、opencv、numpy and other libraries

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



Open the Terminal.

Type following commands in Terminal.

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

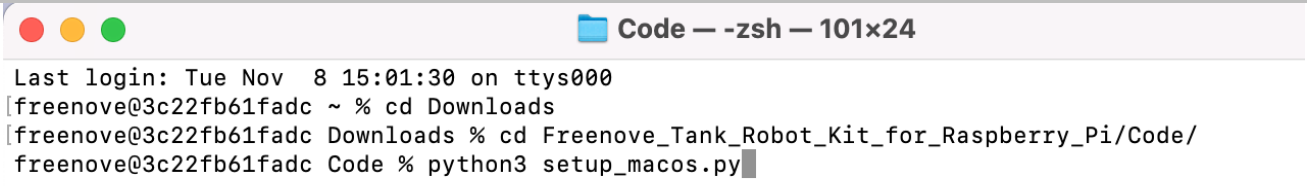
```
cd Downloads
```

2.Enter directory where setup_macos.py is located:

```
cd Freenove_Tank_Robot_Kit_for_Raspberry_Pi/Code/
```

3.Run setup_macos.py:

```
python3 setup_macos.py
```



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

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

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

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

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

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

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

Open client

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

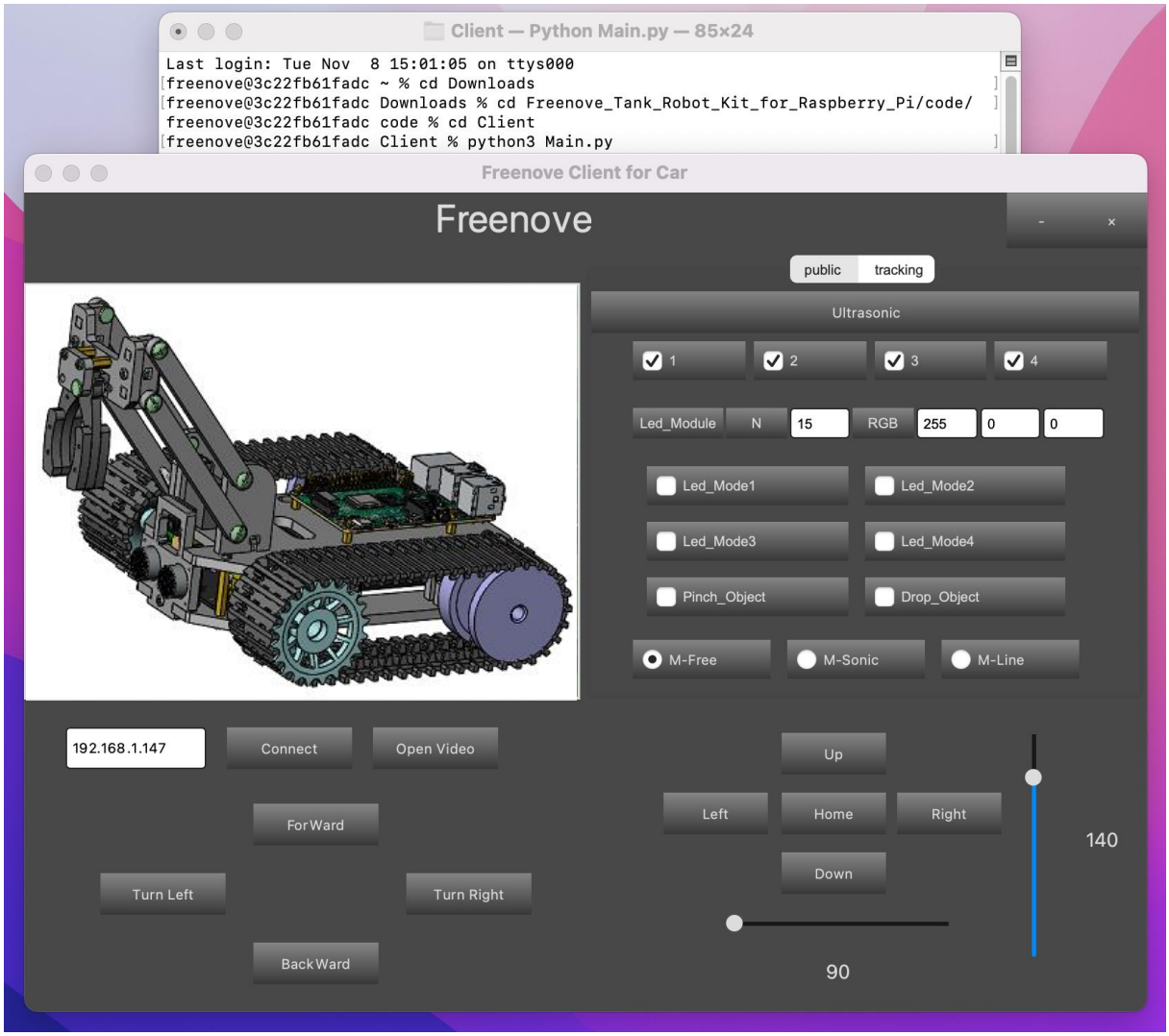
```
Package                 Version
----------------------- --------
numpy                   1.18.1
opencv-python-headless  4.2.0.32
Pillow                  7.0.0
pip                     20.0.2
PyQt5                   5.14.1
PyQt5-sip               12.7.1
setuptools              41.2.0

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

1.Type following command to enter Client folder.

```
cd Client/
```

2.Type following command to run Main.py.

```
python3 Main.py
```



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

Run client in Raspberry Pi (Linux system)

Install Opencv library

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

1. Install opencv development environment:

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

2. Install some tools:

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

Run client

Enter the following commands at the terminal.

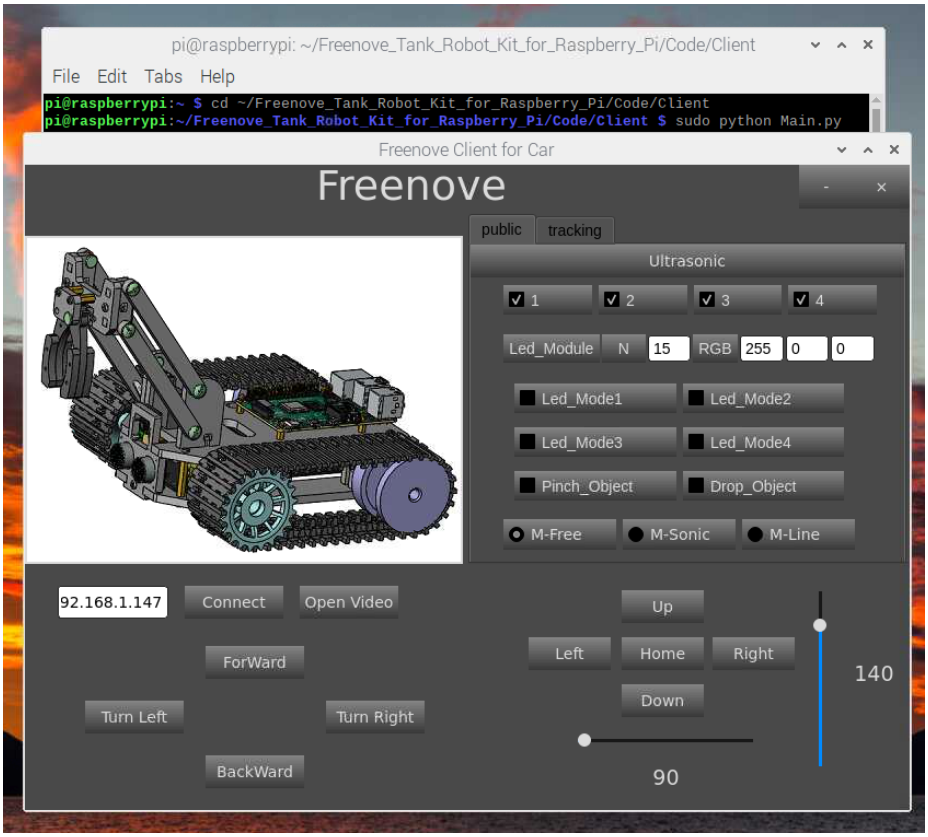
1. Use the cd command to go to the directory where Main.py is located.

```
cd ~/Freenove_Tank_Robot_Kit_for_Raspberry_Pi/Code/Client
```

2. Run Main.py:

```
sudo python Main.py
```

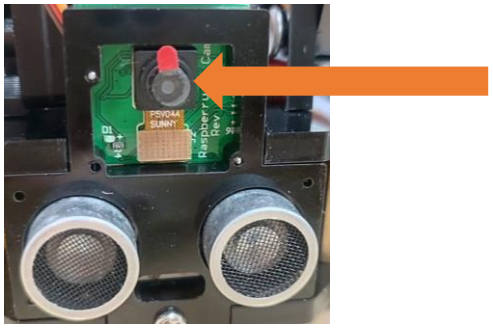
The interface is shown below:



The control mode of client on Linux is the same as that of Windows.

If the image is not clear, please check whether the camera protective film is torn off.

Trouble shooting



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

You need check batteries power indicator or recharge batteries.

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

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

If the raspberry pi system is stuck for a long time, you need reboot raspberry pi.

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

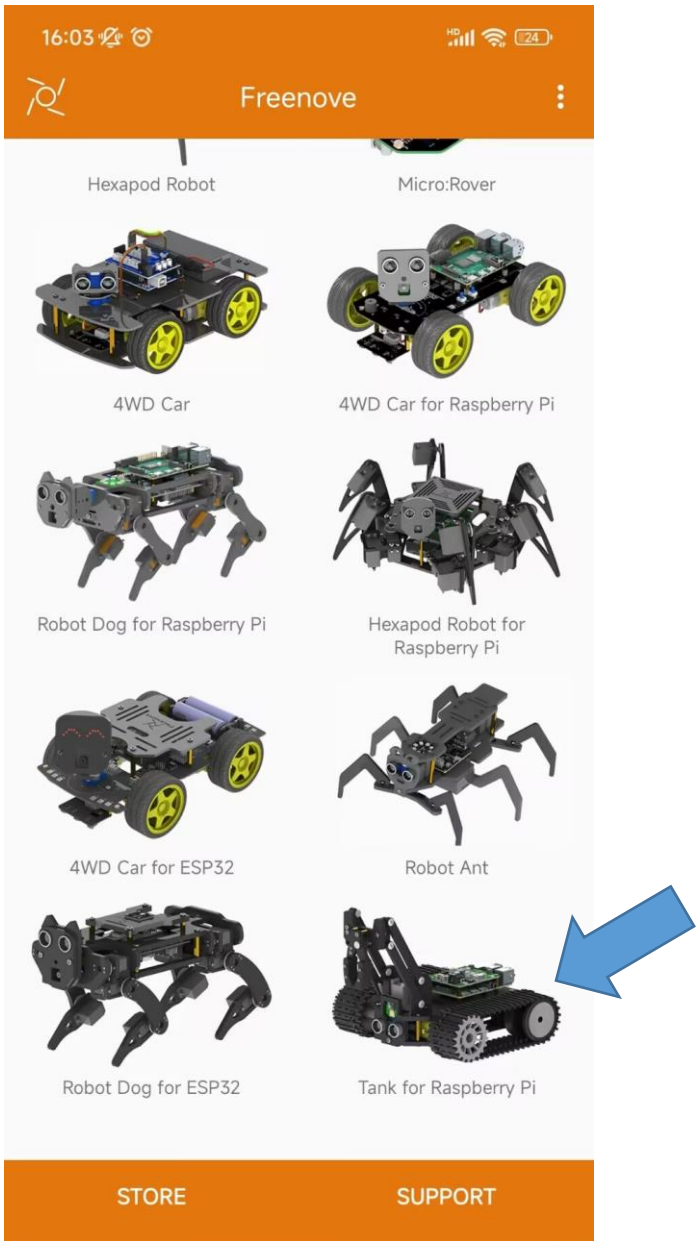
support@freenove.com

Android and iOS app

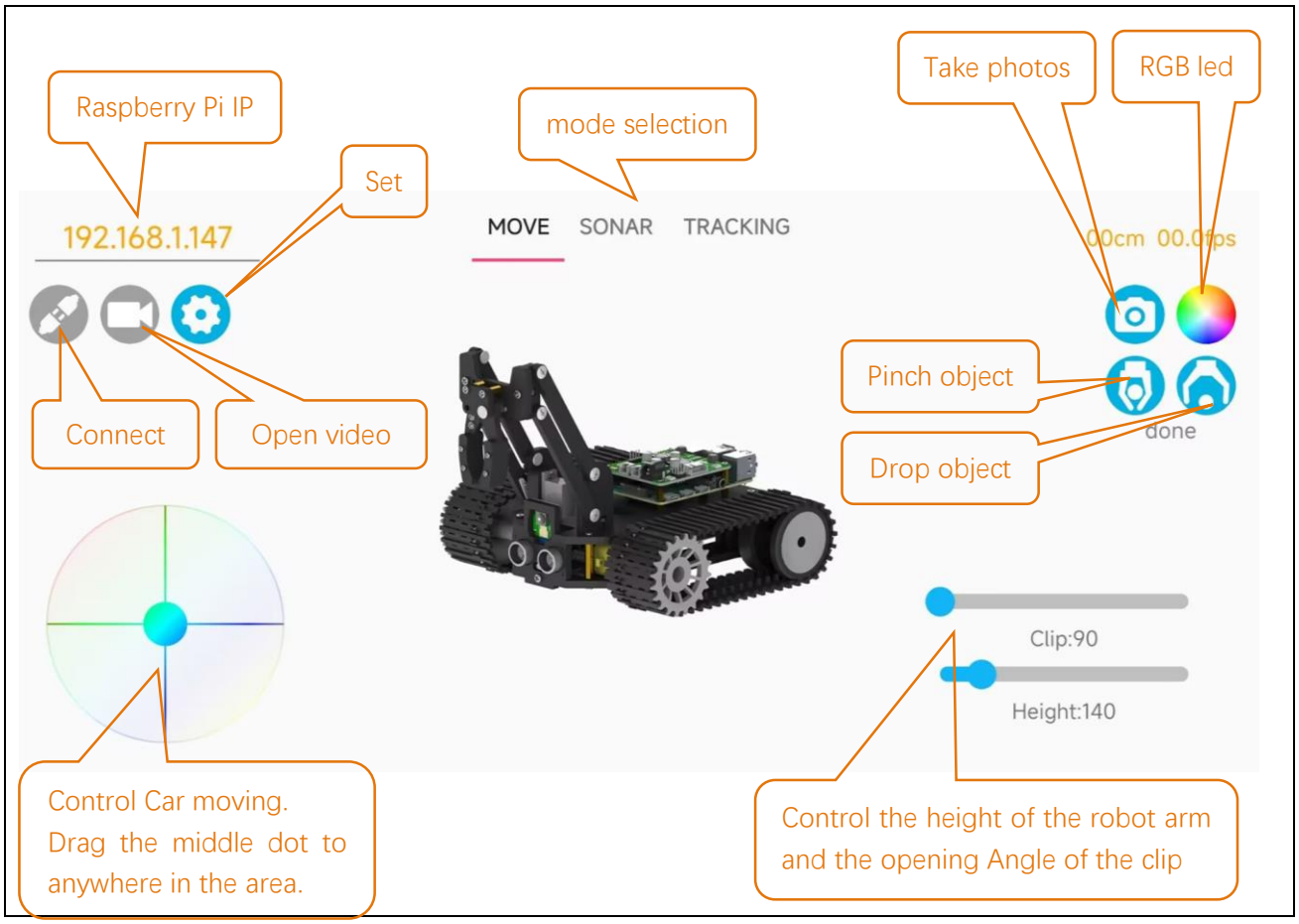
You can download and install the Freenove Android app from below:
On Google play:
https://play.google.com/store/apps/details?id=com.freenove.suhayl.Freenove
On GitHub:
https://github.com/Freenove/Freenove_App_for_Android
In this github repository, you can find the App instruction (Tutorial.pdf).

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

Open the app and select the car.



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



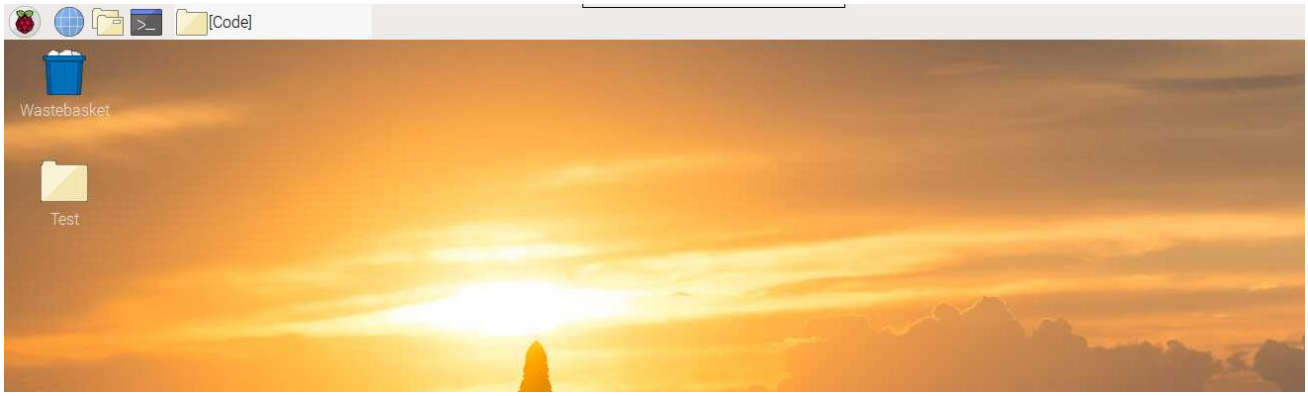
Free innovation

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

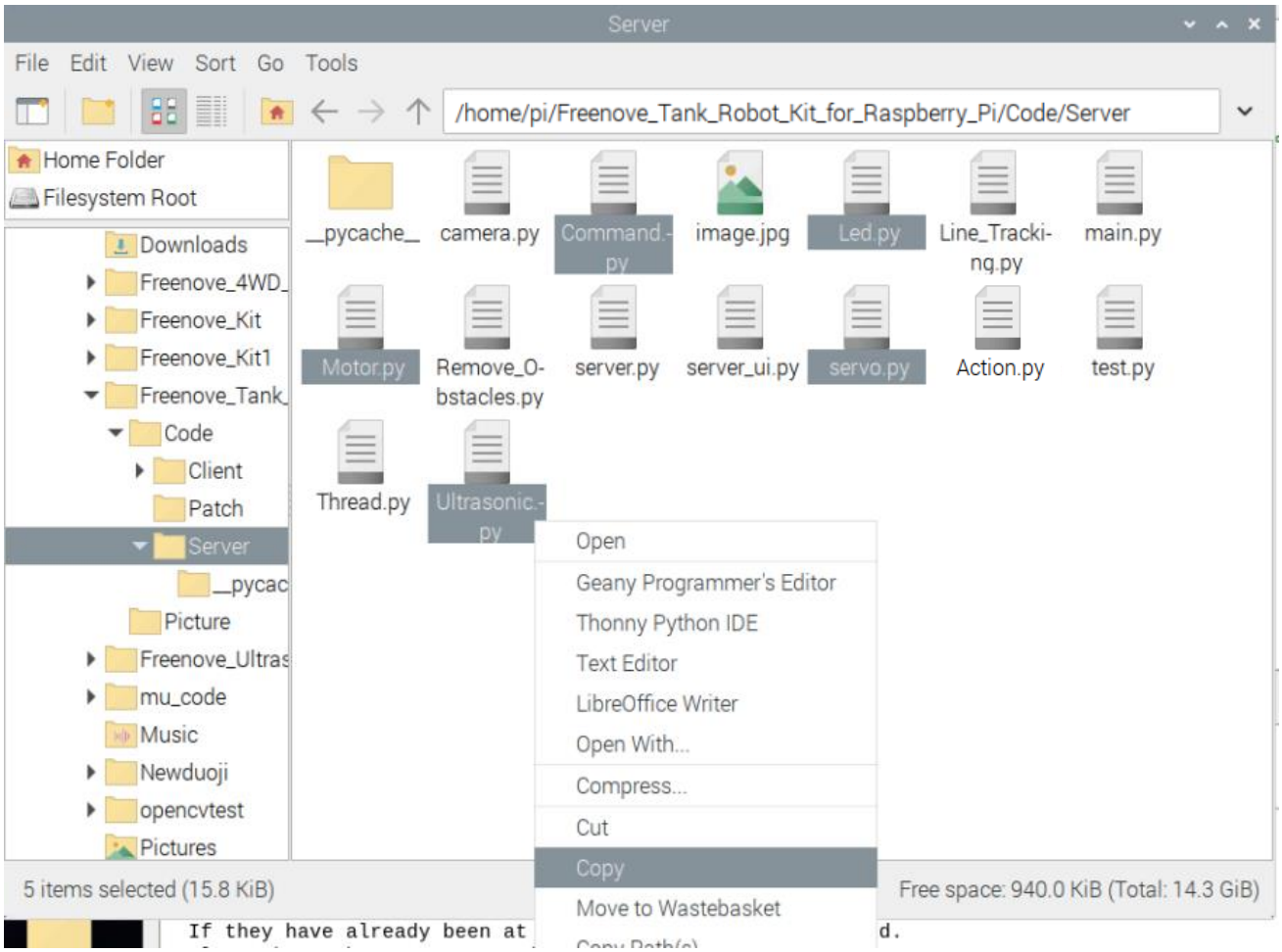
If you want to write your own program to control the car, just follow this section. We will teach you how to program this car.

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

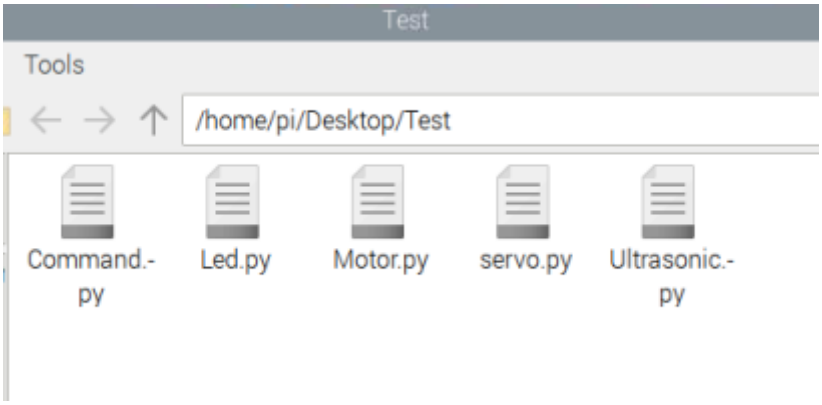
First, turned on S1 and S2. Then open Raspberry Pi, right click and create a new folder on the desktop: Test



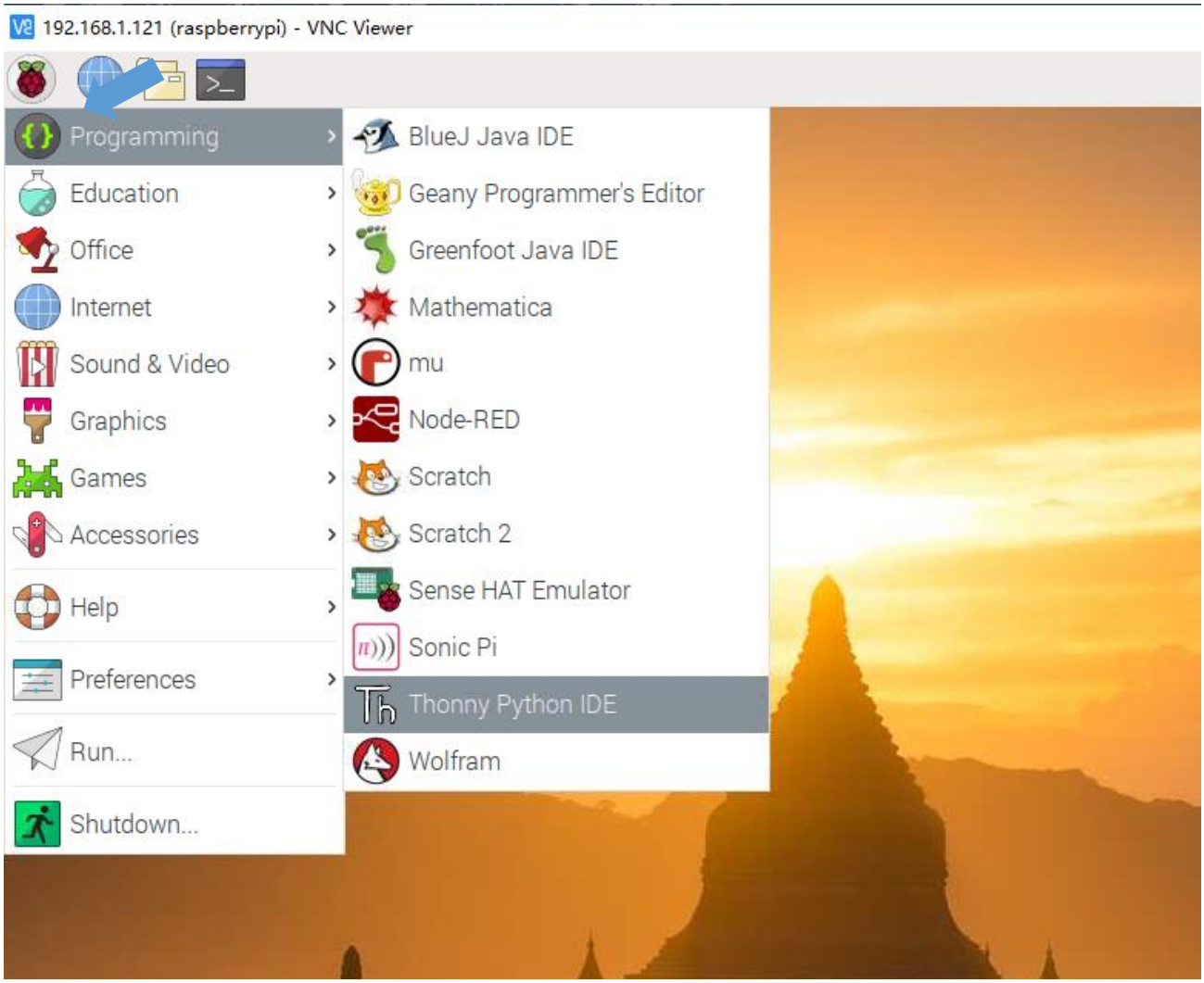
Open Freenove_Tank_Robot_Kit_for_Raspberry_Pi/Code/Server in your Raspberry Pi and copy the following **5 files** into the Test folder we created.

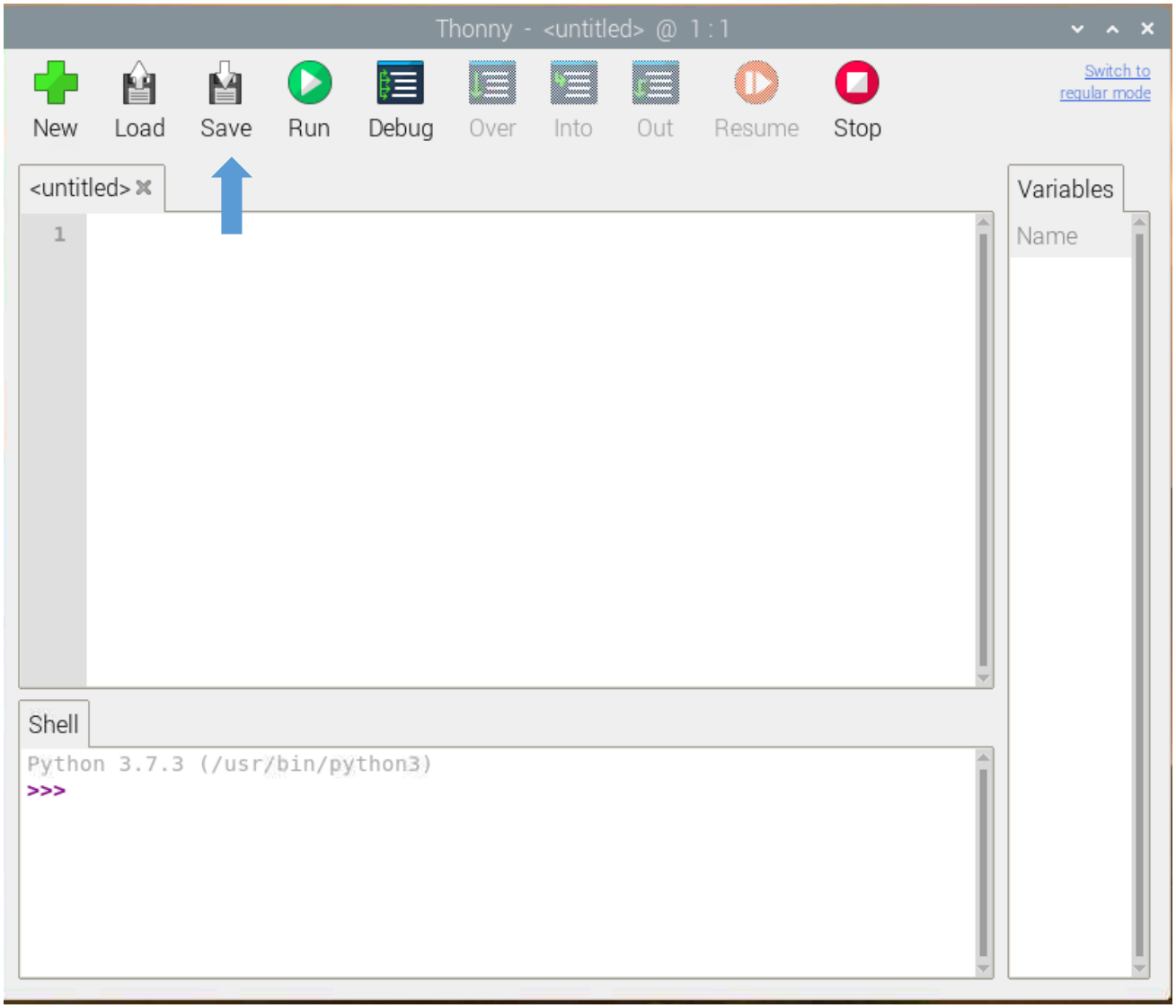


Paste them in Test folder.

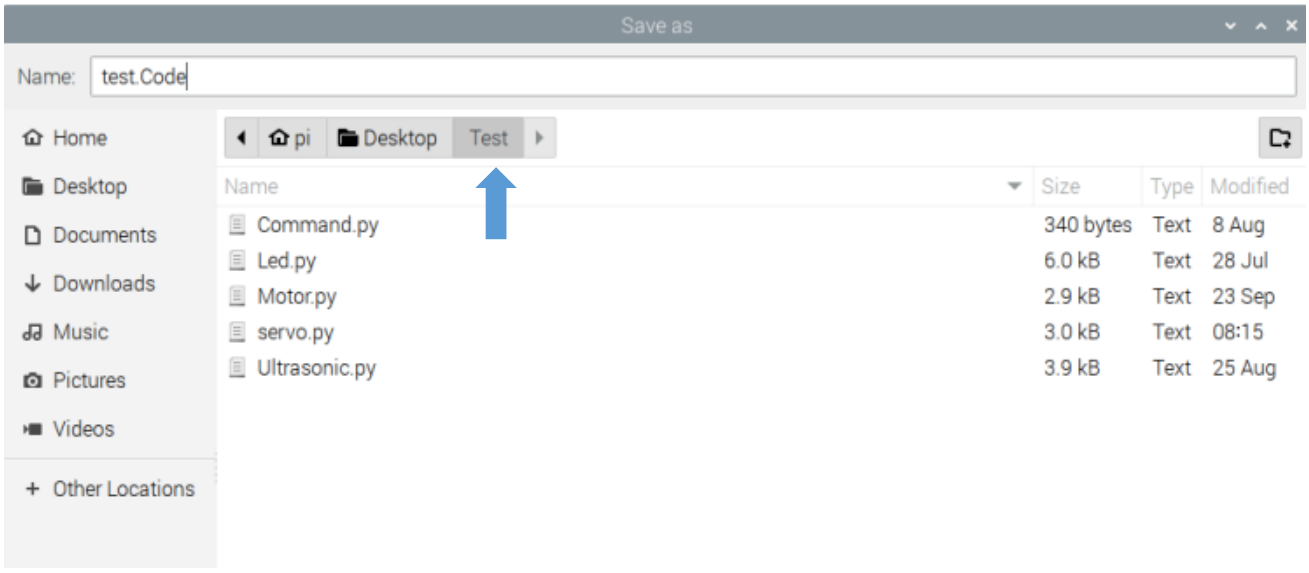


Run Thonny Python IDE

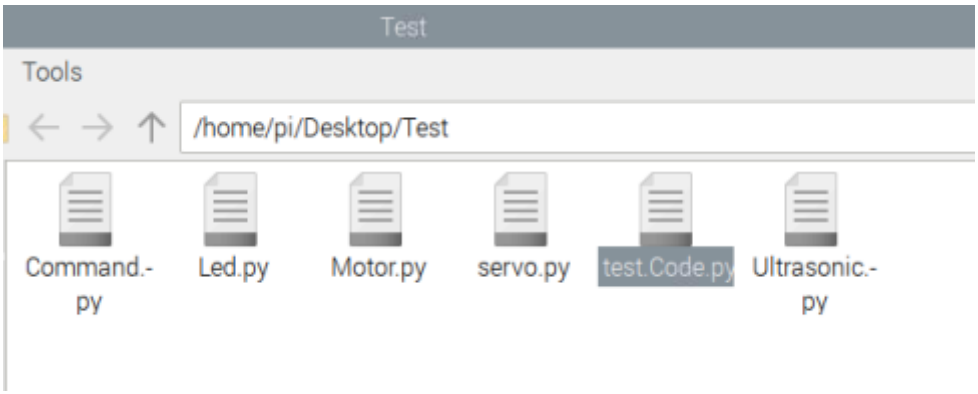

192.168.1.121 (raspberrypi) - VNC Viewer
Programming
Education
Office
Internet
Sound & Video
Graphics
Games
Accessories
Help
Preferences
Run...
Shutdown...
BlueJ Java IDE
Geany Programmer's Editor
Greenfoot Java IDE
Mathematica
mu
Node-RED
Scratch
Scratch 2
Sense HAT Emulator
Sonic Pi
Thonny Python IDE
Wolfram



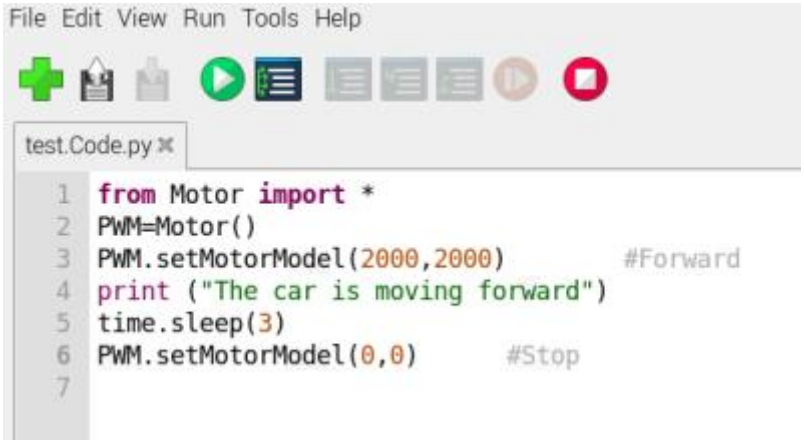
Click Save and save it into the Test folder, with name: test_Code.



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



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



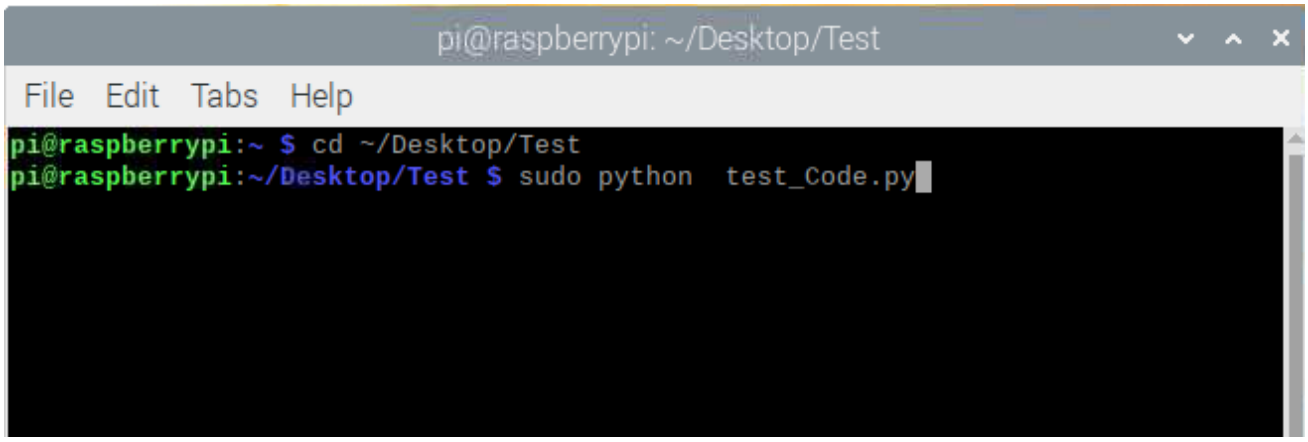
Note: the code and library are written by **Python 3.** You need execute the code with **python 3**.

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

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

Run test_Code.py:

```
sudo python  test_Code.py
```



Code example

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

For more details, please refer to Motor.

```
1  from Motor import *                          #import Motor
2  PWM=Motor()                                  #create an object
3  PWM.setMotorModel(2000,2000)     #Forward
4  print("The car is moving forward")
5  time.sleep(3)                                #waiting 3 second
   PWM.setMotorModel(0,0)                 #Stop
```

LED. For more details, please refer to LED.

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

Servo. For more details, please refer to Servo.

```
from servo import *   #import Led
pwm = Servo()            #create an object
#Servo rotates from 90 degrees to 150 degrees
for i in range(90, 150, 1) :
    pwm.setServoPwm('0', i)
    time.sleep(0.01)
#Servo rotates from 140 degrees to 90 degrees
for i in range(145, 90, -1) :
    pwm.setServoPwm('0', i)
    time.sleep(0.01)
```

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

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

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

What's next?

Thanks for your reading.

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

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

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

Thank you again for choosing Freenove products.