Free your innovation

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

Warning

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

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

About

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

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

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

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

References

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

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

- Client Code of Client
- mjpg-streamer Code of mjpg-streamer
- Server Code of Server
- Readme.txt Instructions

Support

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

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

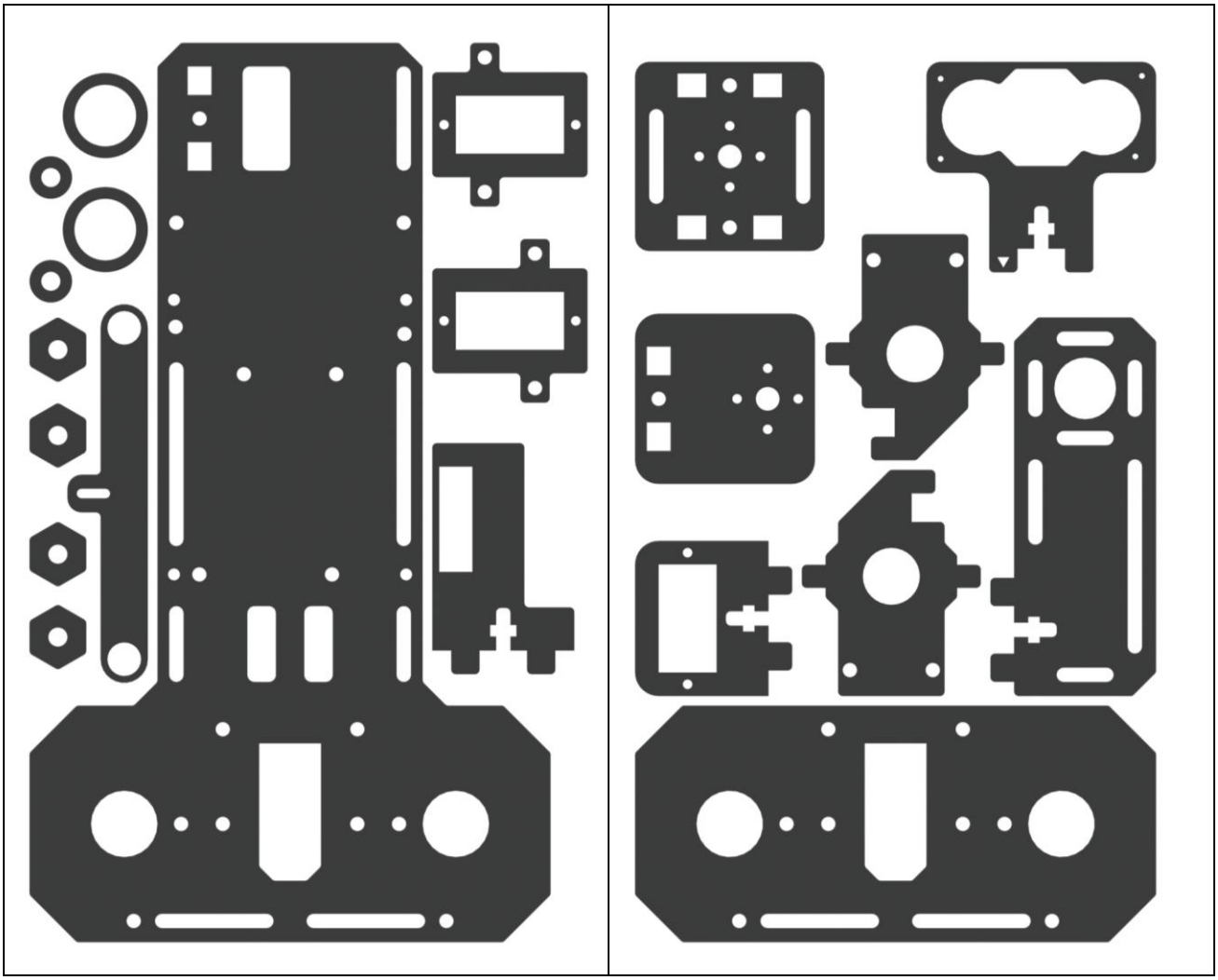
Please send email to:
support@freenove.com
On working day, we usually reply to you within 24 hours.

Copyright

Contents

List

Acrylic Sheet



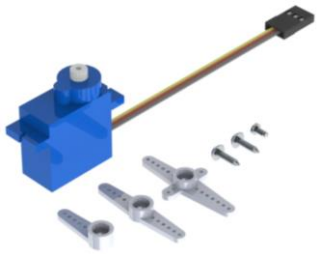
Surface of the acrylic sheet is covered with a layer of protective film. You need to remove it first before using. Some holes of the acrylic sheets may have residues, so you may need to clean them before using.

Machinery Parts

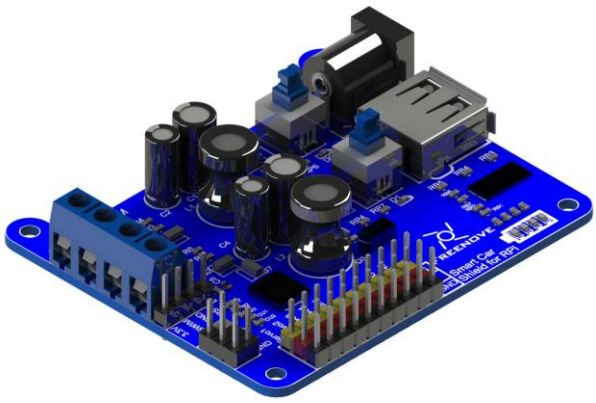
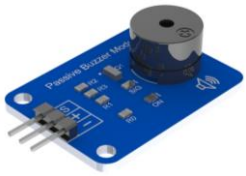
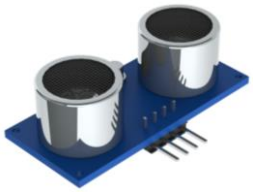
In addition to bearing F624ZZ and F687ZZ, the number of each following part provided is more than required.

M4 Washer  x4 Freenove	M2 Nut  x8 Freenove	M3 Nut  x11 Freenove	M4 Nut  x2 Freenove	M2*10 Screw  x8 Freenove	M3*4 Screw  x6 Freenove
M3*8 Screw  x22 Freenove	M3*12 Screw  x5 Freenove	M3*30 Screw  x6 Freenove	M4*40 Screw  x2 Freenove	M3*10 Countersunk Head Screw  x4 Freenove	M1.4*6 Self-tapping Screw  x6 Freenove
F624ZZ Bearing  x2 Freenove	F687ZZ Bearing  x4 Freenove	M3*6 Copper Standoff  x6 Freenove	M3*12 Copper Standoff  x6 Freenove	M3*30 Copper Standoff  x6 Freenove	M4 Spring Washer  x4 Freenove
M2.5*8 Scew  x10 Freenove	M2.5*6 M-F Copper Standoff  x6 Freenove	M2.5*14 Copper Standoff  x6 Freenove			

Transmission Parts

Servo x3 	Driven wheel x1 
DC speed reduction motor x2 	Driving wheel x2 

Electronic Parts

Freenove Smart Car Shield for RPi x1 	Freenove Passive Buzzer Module x1 	Freenove RGB LED Module x1 
18650x2 Battery Holder x1 	USB Camera x1 	HC-SR04 Ultrasonic Module x1 
Micro USB Cable x1 		
Jumper Wire M/M x10 		

Tools

Cross screwdriver x1 	Slotted screwdriver x1 	Multifunctional Spanner x1 

Self-prepared Parts

Two 18650 lithium batteries without protection board. 
Raspberry Pi (Recommended model: Raspberry 3 Model B) x1 

Preface

Welcome to use Freenove Three-wheeled Smart Car Kit for Raspberry Pi. Whether you are a senior maker, or have little technical knowledge, by using this tutorial, you can make a very cool smart car with video surveillance and ultrasonic radar.

This kit is based on the popular control panel Pi Raspberry, so you can share and exchange your experience and design ideas with many enthusiasts all over the world. The parts in this kit include all electronic components, modules, and mechanical components required for making the smart car. And all of them are packaged individually. There are detailed assembly and commissioning instructions in this book. And if you encounter any problem, you can always look for fast and free technical support to us to ensure that your smart car is successfully assembled and run.

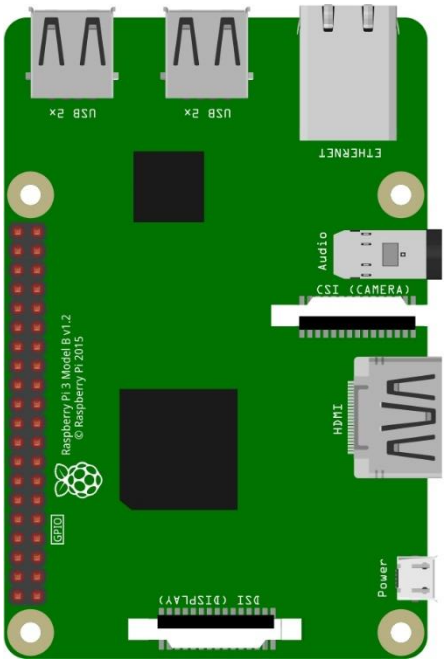
The contents in this book can ensure enthusiastic with little technical knowledge to make the smart car. If you are very interested in Raspberry Pi, and want to learn how to program and build the circuit, please visit our website or contact us to buy the kits designed for beginners:

Freenove Basic\LCD1602\Super\Ultrasonic\RFID\Ultimate Starter Kit for Raspberry Pi

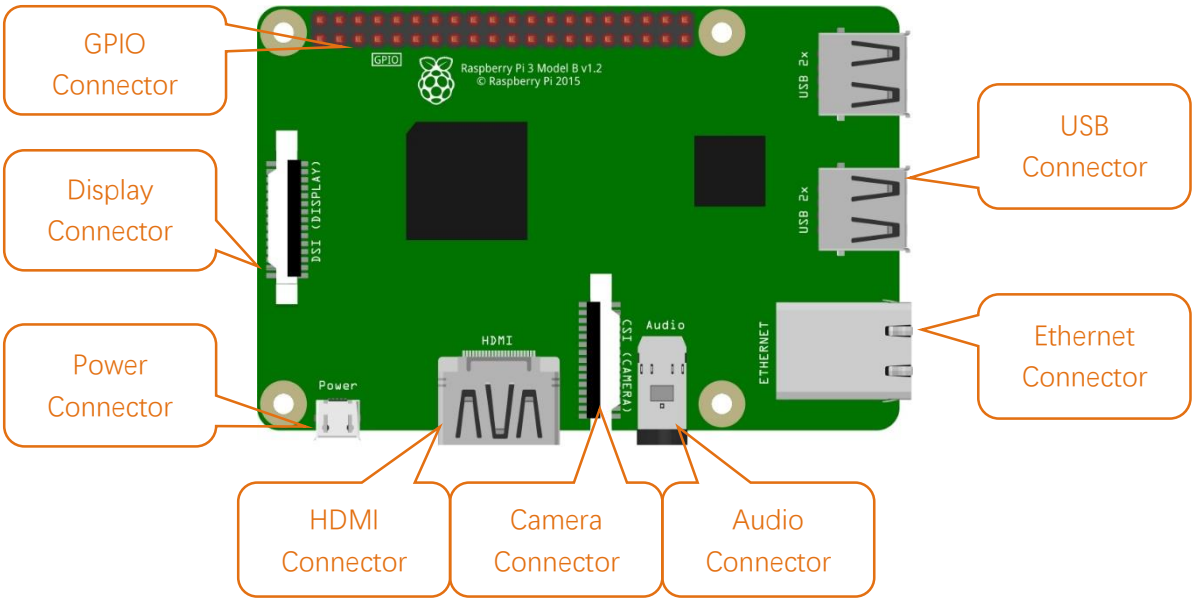
Raspberry Pi

Raspberry Pi (called RPi, RPI, RasPi, the text these words will be used alternately behind), 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 in Linux system and along with the release of windows 10 IoT, we can also run it in Windows. Raspberry Pi (with interfaces for USB, network, HDMI, camera, audio, display and GPIO), as a microcomputer, can be running in command line mode and desktop system mode. Additionly, it is easy to operate just like Arduino, and you can even directly operate the GPIO of CPU. So far, Pi Raspberry has 7 versions: A type, A+ type, B type, B+ type, second-generation B type, third-generation B type and Zero version, respectively. 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. Orther 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. The projects in this tutorial, with no special note, are using Raspberry Pi 3 Model B (RPi3B), which is compatible with Raspberry Pi A+, B+, 2B and Zero.

Schematic diagram of RPi3B is shown below:

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

Hardware interface diagram of RPi3B is shown below:



GPIO

General purpose input/output; in this specific case the pins on the Raspberry Pi and what you can do with them. So called because you can use them for all sorts of purposes; most can be used as either inputs or outputs, depending on your program.

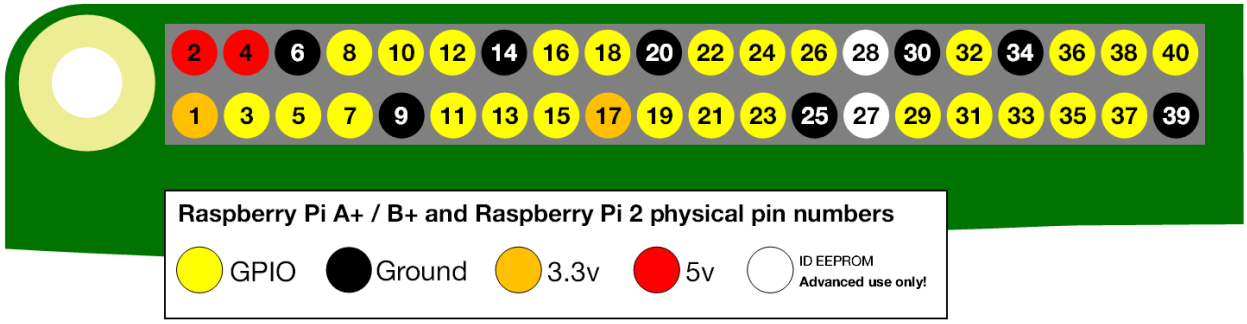
When programming the GPIO pins there are two different ways to refer to them: GPIO numbering and physical numbering.

GPIO NUMBERING

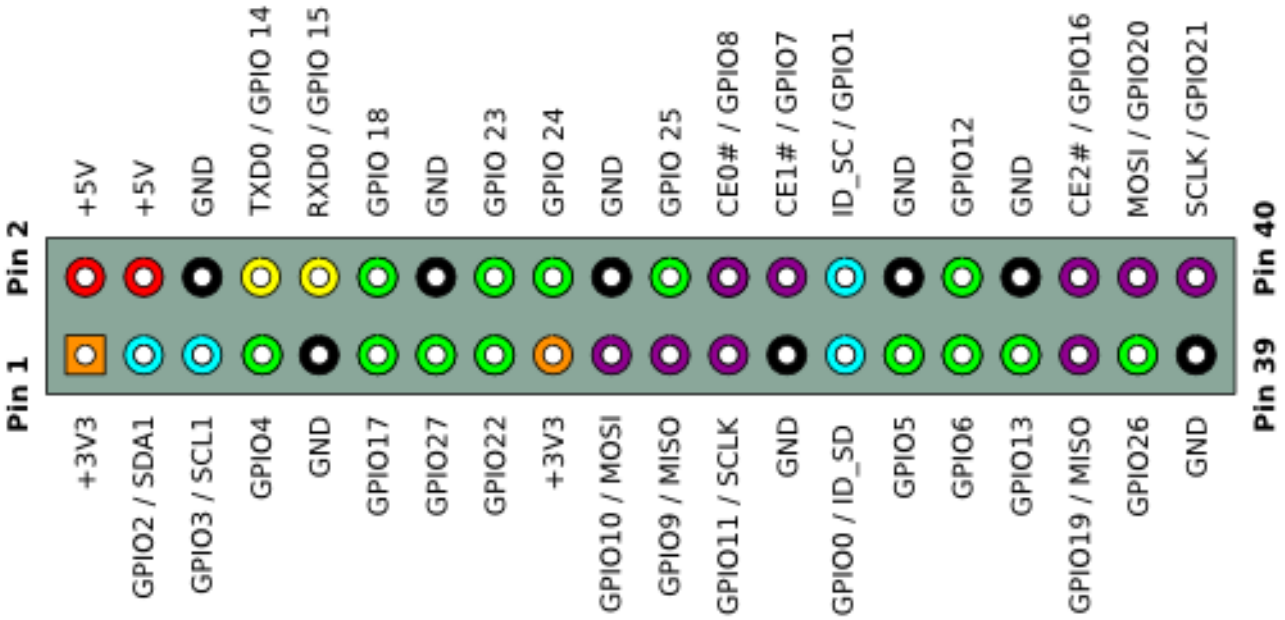
These are the GPIO pins as the computer sees them. The numbers don't make any sense to humans, they jump about all over the place, so there is no easy way to remember them. You will need a printed reference or a reference board that fits over the pins.

PHYSICAL NUMBERING

The other 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' and it looks like this:



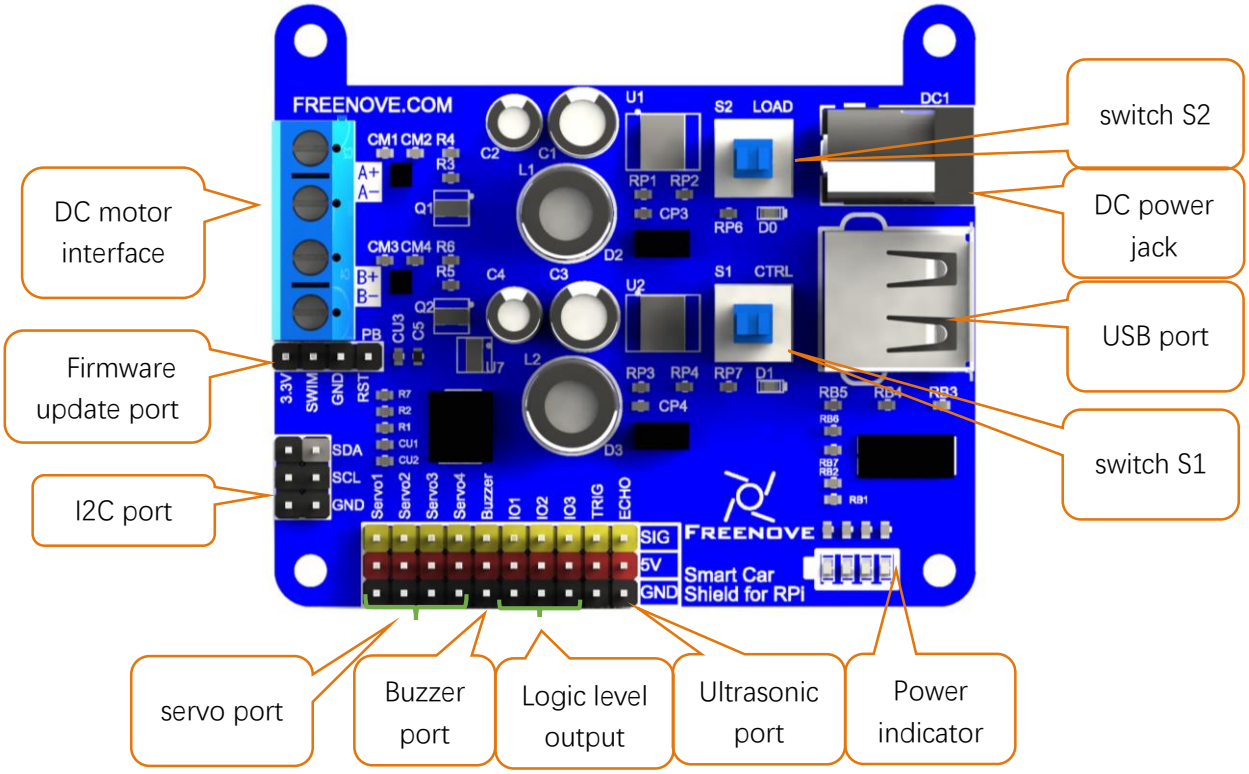
Each pin is defined as below:



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

Smart Car Shield for RPi

Smart Car Shield for RPi (hereinafter called Shield) is designed to extend smart car control board, which is suitable for control board with host computer communication ability of I2C. The size and position of the location holes on the shield is suitable for RPi. Below is the schematic diagram and function description of the Smart Car Shield for RPi:



- DC power jack: this shield uses 5.6~11V DC power supply. If the power supply is beyond this voltage range, it may cause damage to Shield.
- There are two Voltage Regulation System for 5V/3A, which are controled by switch S1 and S2 respectively.
  a) First 5V/3A power supply is output by the USB port on the right side of Shield, which is used to provide power for RPi. In addition, the motor driver, controller, battery voltage detector and other systems logic power on the Shield are powered by this one. It is controlled by the switch S1:CTRL for on or off. This means that if you want the Shield to work, you have to turn on the switch S1:CTRL.
  b) Second 5V/3A power is output by the red pins on Shield (marked as 5V). Motor, servo, buzzer, LED, ultrasonic and other load are powered by this one. It is controled by the switch S2:LOAD for on or off. This means that if you want the load connected to the Shield to work, you have to turn on the switch S2:LOAD.
- Power indicator. Four LEDs are used to indicate the power. With the power consumption, LEDs will be turned off one by one.
- I2C communication port. Shield using I2C communication protocol. The default device address for I2C is 0x18, which can be changed to any address through specific command. And the data can be preserved with power off.
- Firmware update port. If there is a new released firmware, the firmware can be updated through this port.
- DC motor interface. This shield can control speed and steering of two motors. Maximum current of each

motor is 1.8A, and voltage is the input voltage of Shield.

- Servo port. This shield provides four servo ports. The servo control accuracy is 1us, which is 0.09 degrees.
- Buzzer port: This shield provides 1 buzzer port. This port can produce PWM with frequency 0-65535Hz, and duty cycle 50%.
- The logic level output: This shield provides 3 common ports which can output logic level (0V/3.3V).
- Ultrasonic port: This shield provides a SR04 ultrasonic port.
- USB power port: This shield provides a USB power port to supply power for RPi.

The communication and command about this shield will be explained later.

Chapter 0 Preparation

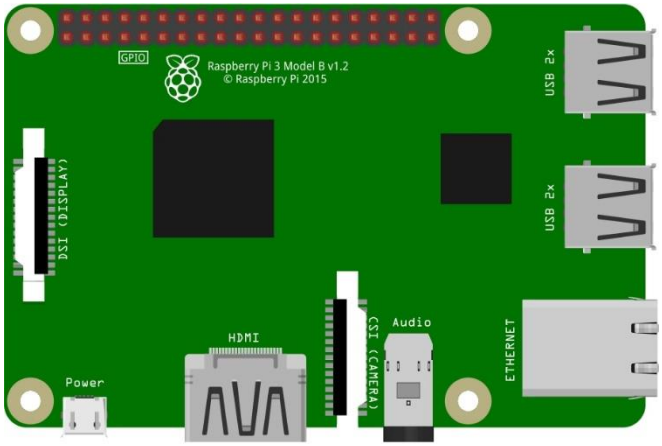
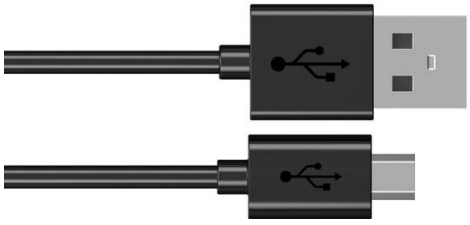
In this chapter, we will do some necessary preparation work: start your Pi Raspberry and install some necessary libraries.

Step 0.1 Install the System

Firstly, install the system for your RPi. The RPi system we choose is official recommended RASPBIAN operating system. All the projects and the code in this book work in this system. If you have installed RPi RASPBIAN system, and it can be used normally, you can skip this step.

Component List

Required Components

Raspberry Pi 3B x1 	5V/2A Power Adapter 
Micro USB Cable x1 	Micro SD Card (TF Card) x1; Card Reader x1 

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/2A, because lack of power supply will lead to many abnormal problems, even damage to your RPI. So use of power supply with 5V/2A is highly recommend. SD Card Micro (recommended capacity 8GB or more) is a hard drive for RPi, which is used to store the system and personal files.

Optional Components

1.Display with HDMI interface

2.Mouse and Keyboard with USB interface

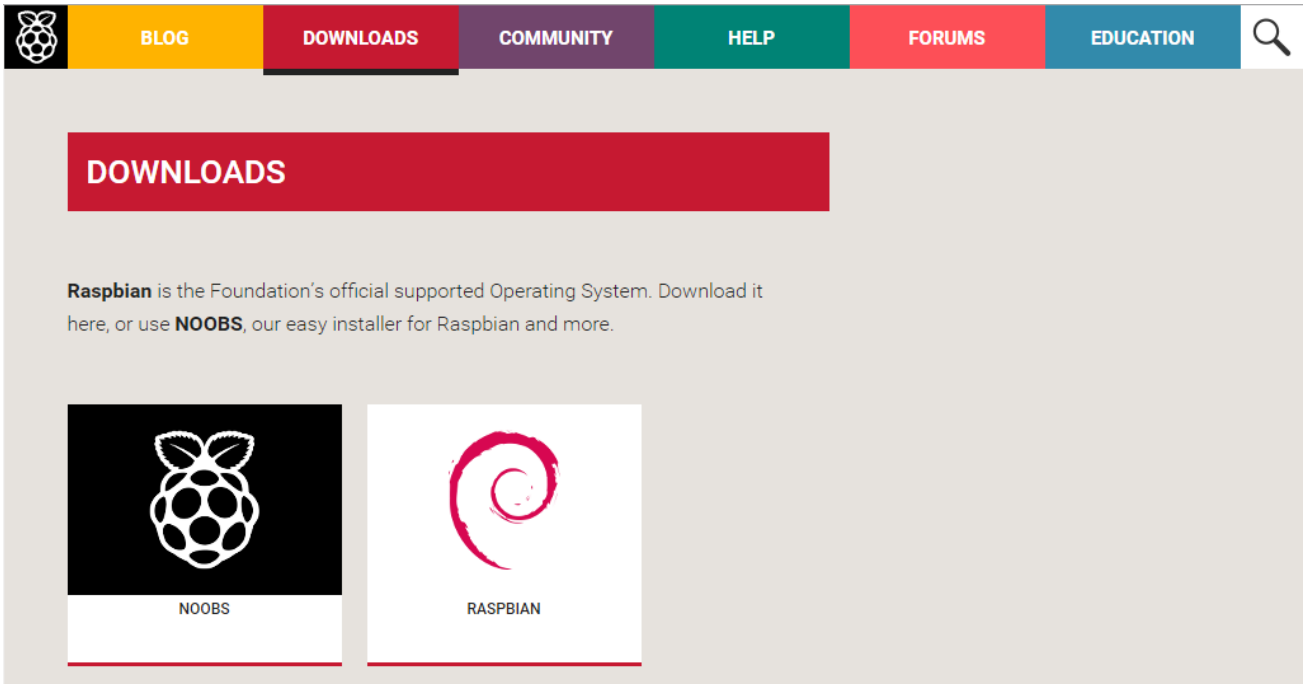
Among these optional components, the Display as a screen for RPi. If no, it does not matter, you can use a remote desktop of your personal PC to control your RPi. Mouse and keyboard are the same, if no, use remote desktop and share a set of keyboard and mouse personal with PC.

Software Tool

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

Selecting System

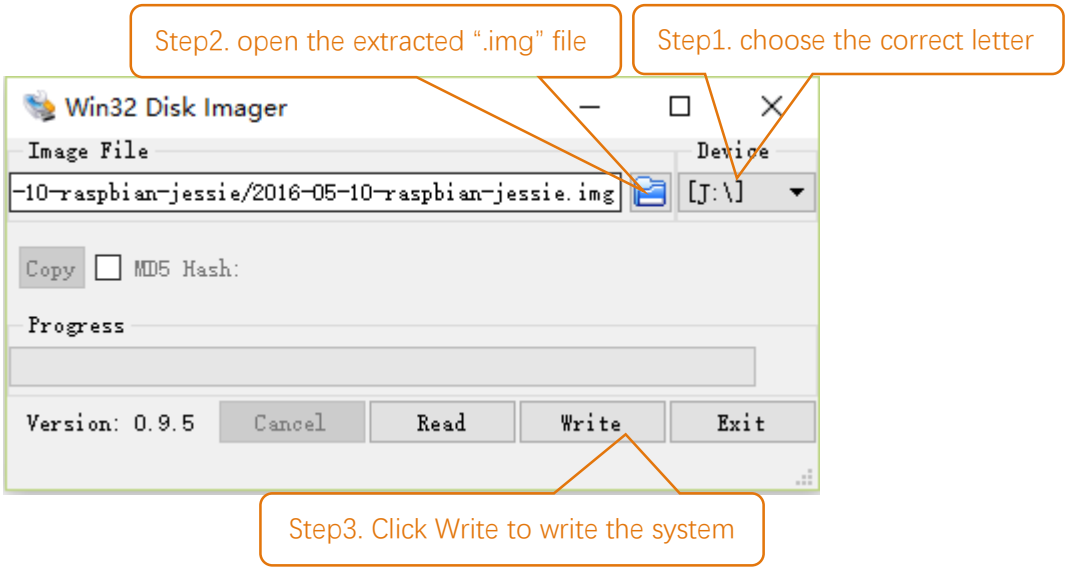
Visit RPi official website (https://www.Raspberry Pi.org/), click "Downloads" and choose to download "RASPBIAN". RASPBIAN supported by RPI is an operating system based on Linux, which contains a number of contents required for RPi. We recommended RASPBIAN system to beginners.



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

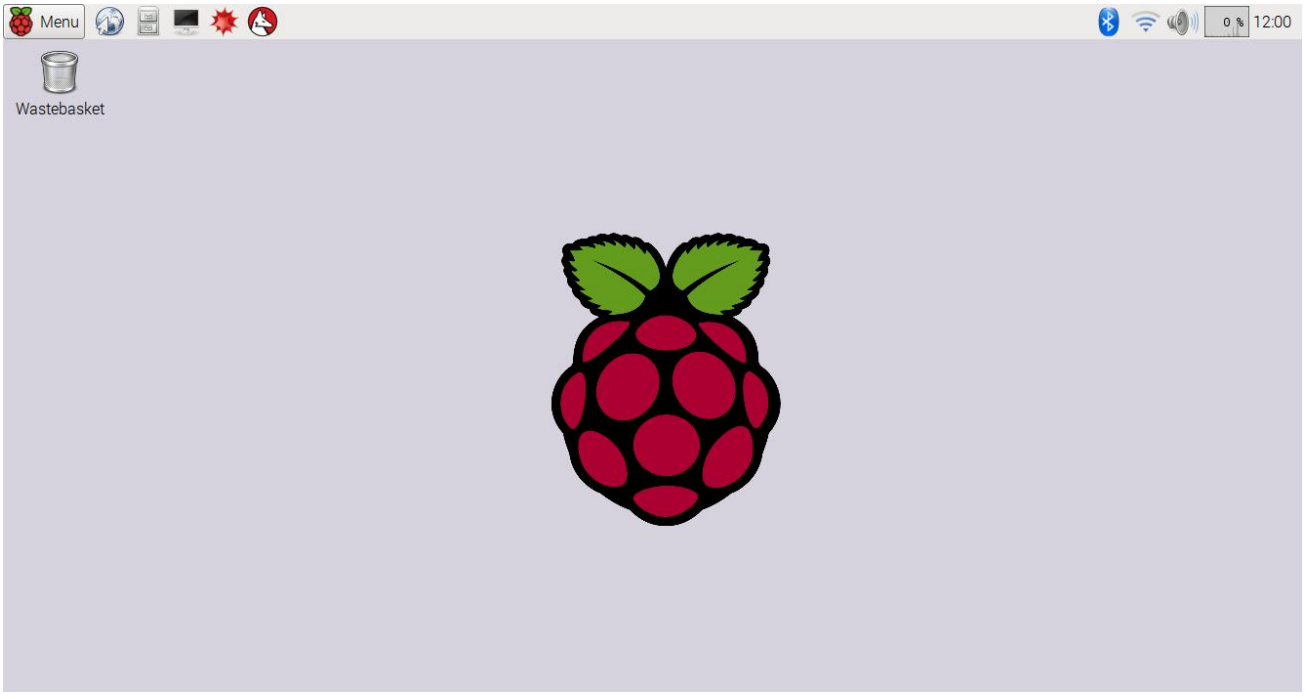
Write System to Micro SD Card

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



Start Raspberry Pi

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



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

Remote desktop

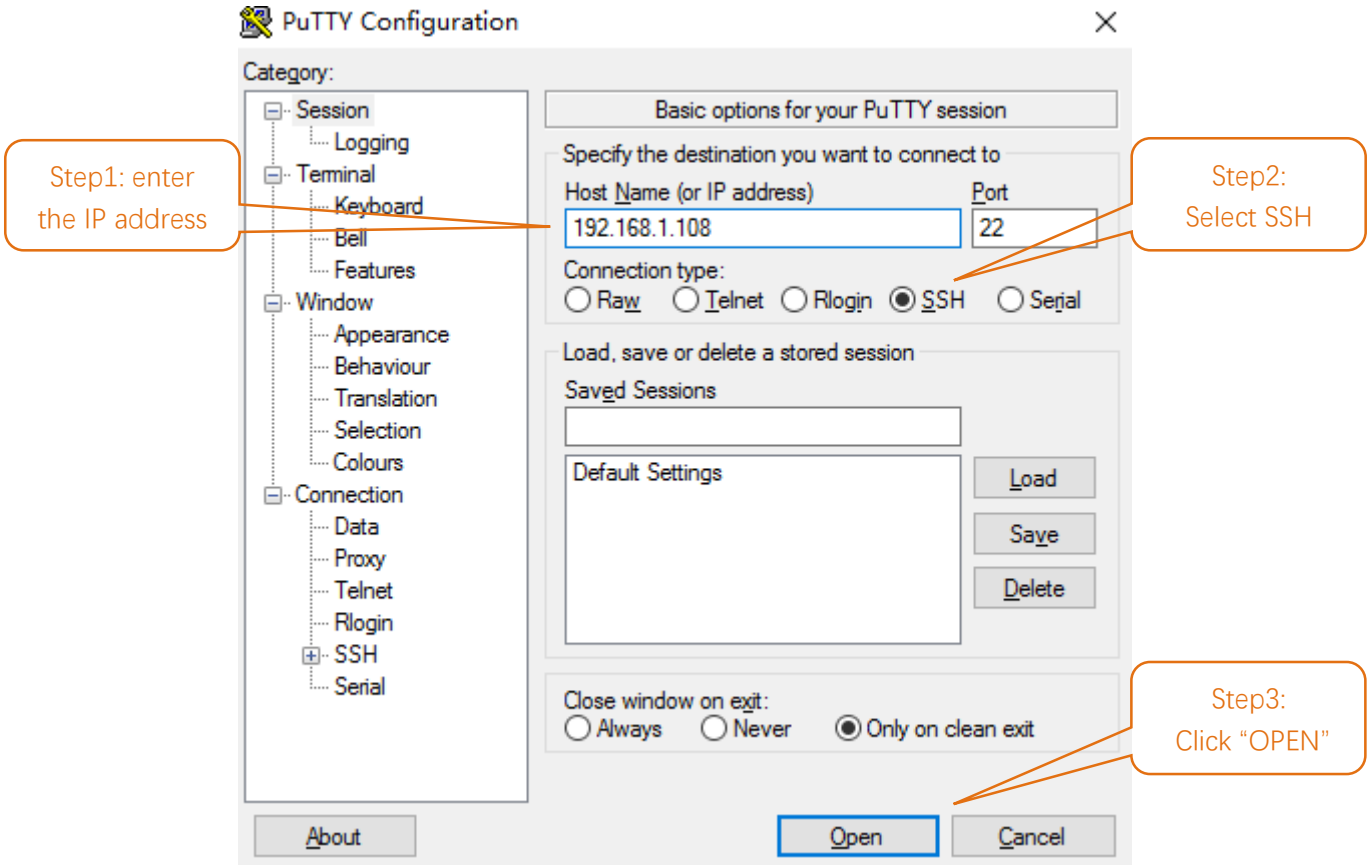
If you don't have a spare display, mouse and keyboard for your RPi, you can use a remote desktop to share a display, keyboard, and mouse with your PC. Below is how to use remote desktop to control RPi under the Windows operating system.

Install Xrdp Services

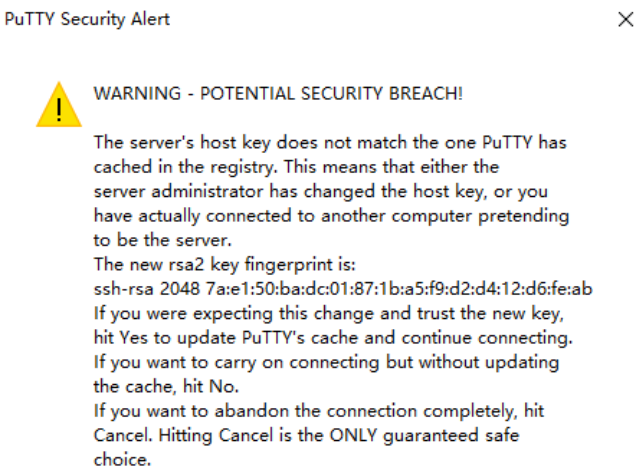
First, download the tool software Putty. Its official address: http://www.putty.org/

Or download it here: http://www.chiark.greenend.org.uk/~sgtatham/putty/download.html

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

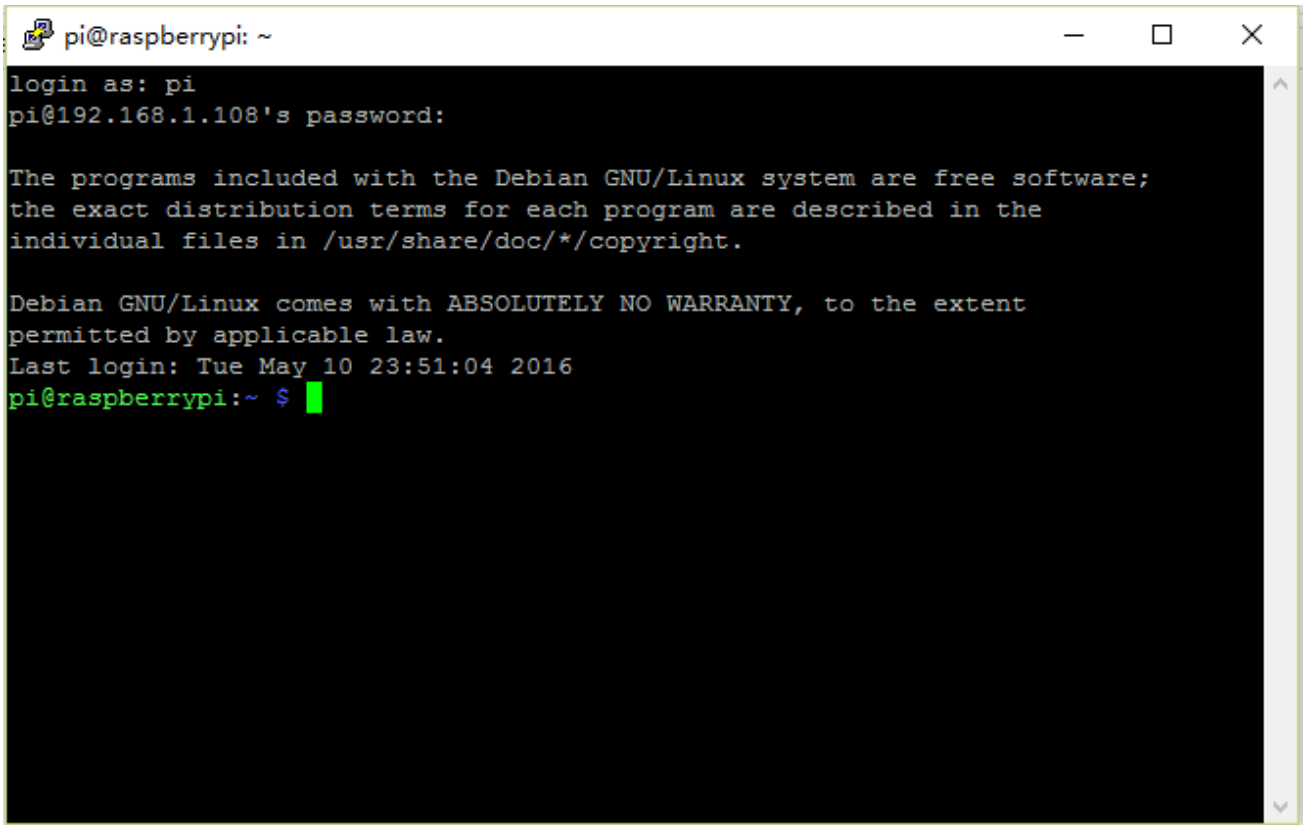


There will appear a security warning at first logining. Just click “YES”.



Then there will be a login interface (RPi default user name: pi; the password: raspberry). When you enter the password, there will be no display on the screen, but this does not mean that you didn’t enter. After the correct output, press “Enter” to confirm.

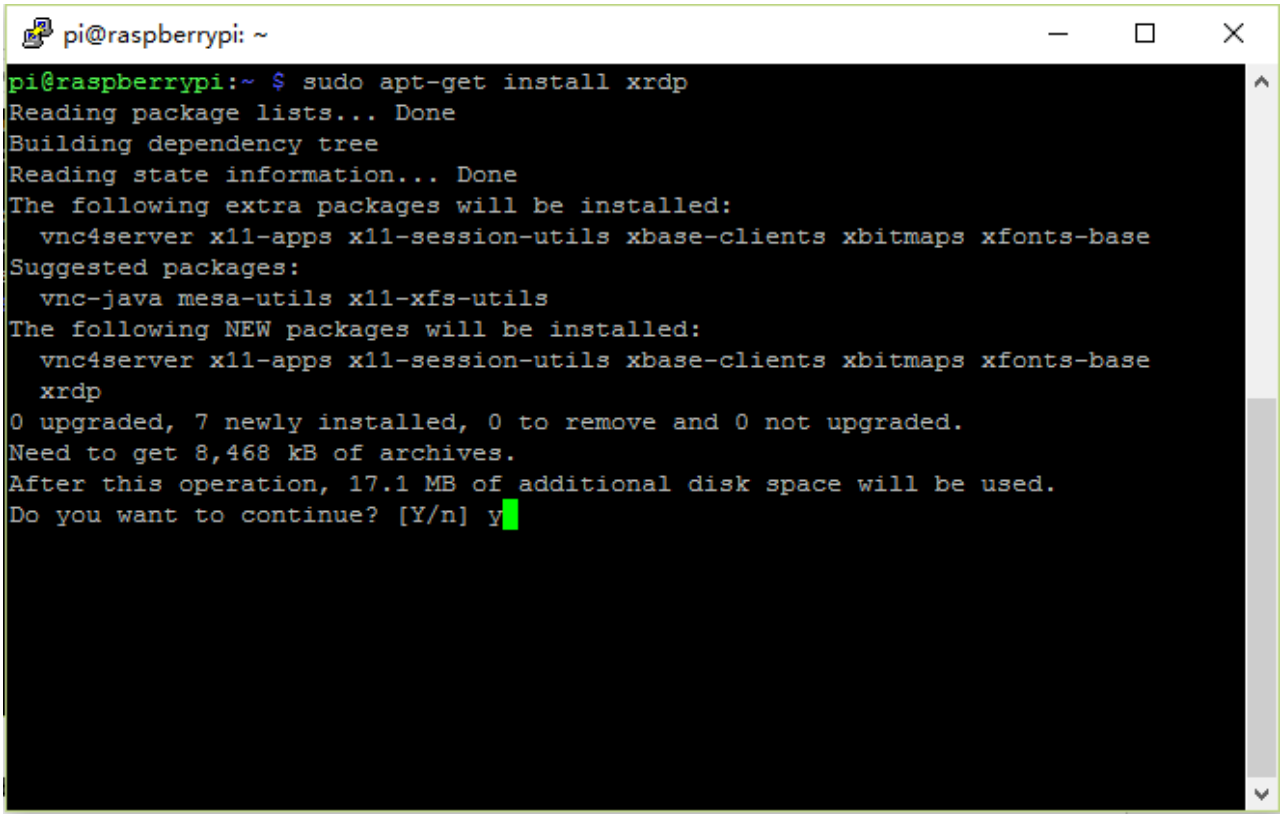
Then enter the command line of RPi, which means that you have successfully login to RPi command line mode.



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

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

Latter, the installation starts.

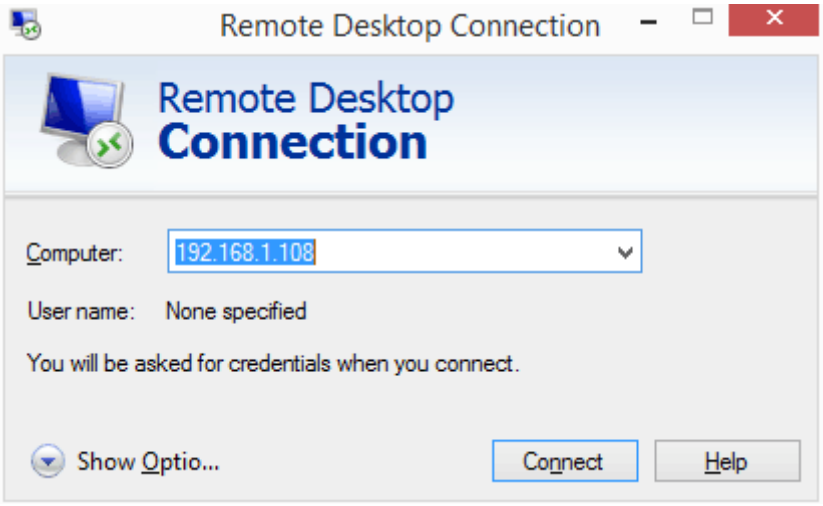


Enter "Y", press key “Enter” to confirm.

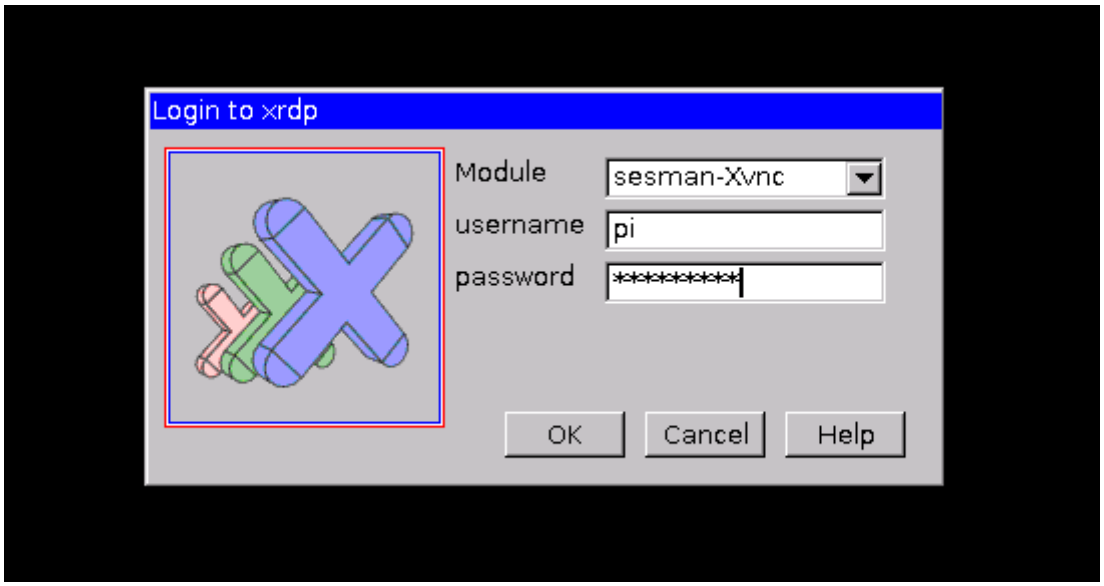
After the installation is completed, you can use Windows remote desktop applications to login to your RPi.

Login to Windows remote desktop

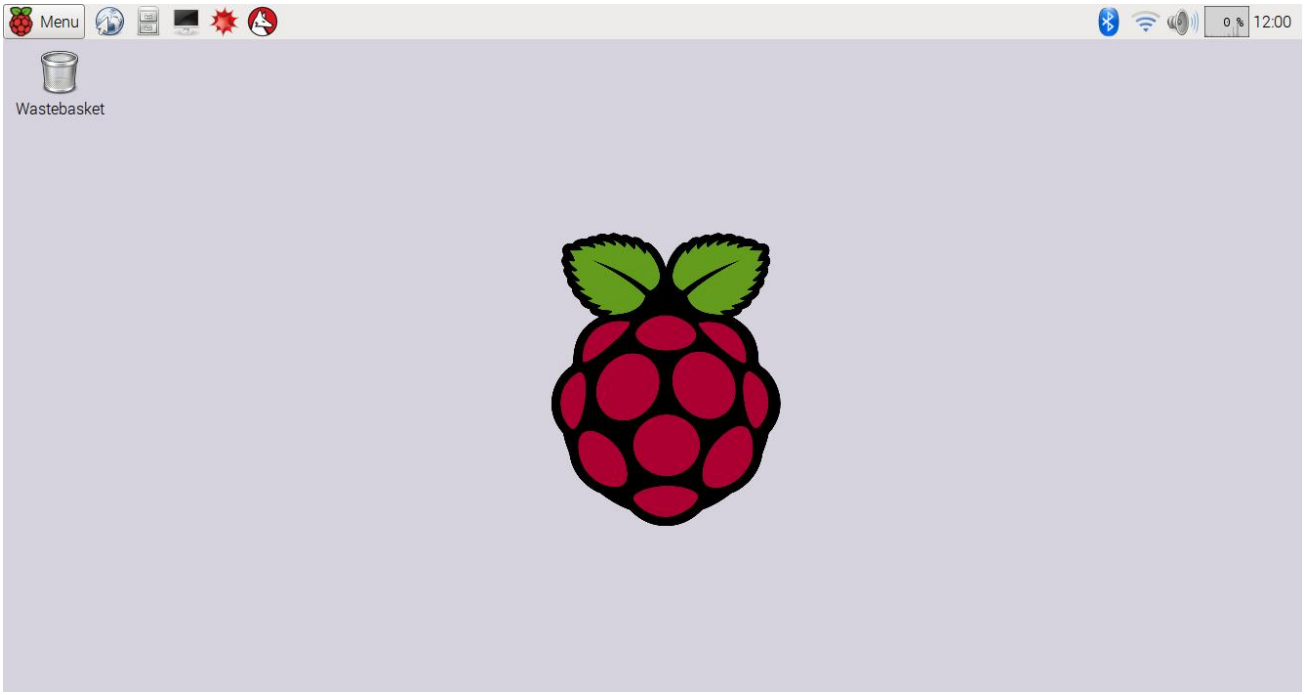
Use "WIN+R" or search function, open the remote desktop application "mstsc.exe" under Windows, enter the IP address of RPi and then click “Connect”.



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



Later, you can enter the RPi desktop system.



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

WiFi

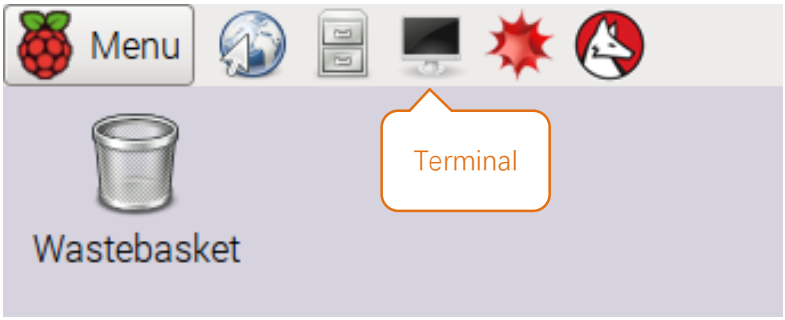
Raspberry Pi 3 Model B integrates a WLAN card. You can use it to connect to your WiFi. Then you can use the wireless remote desktop to control your RPi. This will be helpful for the following work. Raspberry Pi of other models can use wireless remote desktop through accessing an external USB wireless card.

Step 0.2 Configure I2C

Enable I2C

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

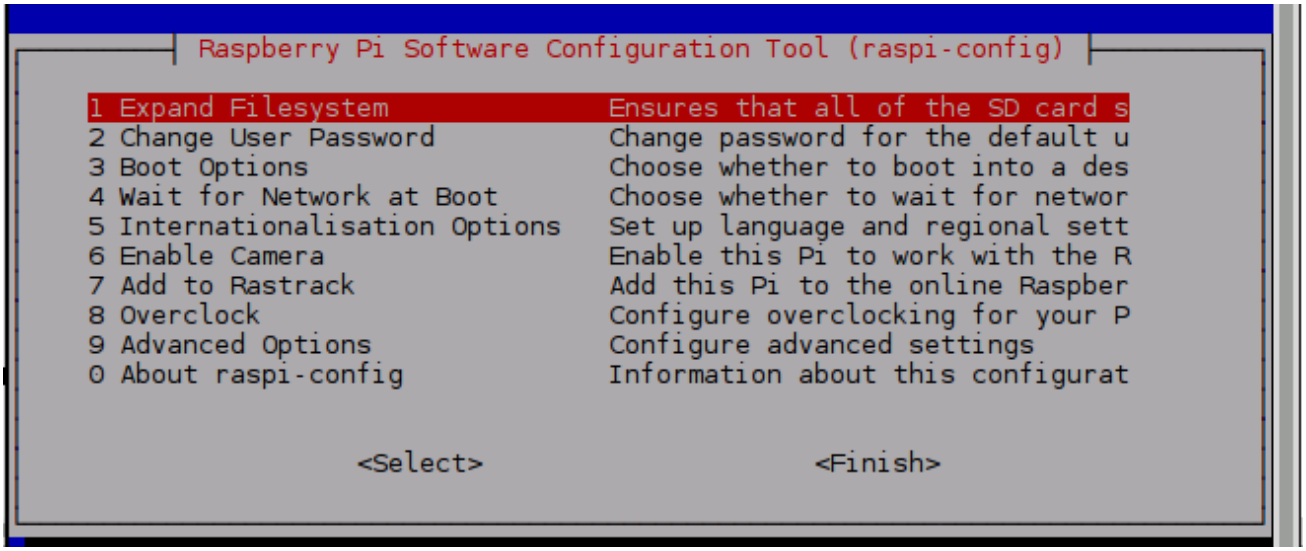
open the terminal:



Type command in the terminal:

```
sudo raspi-config
```

Then open the following dialog box:



Choose "9 Advanced Options" "A6 I2C"→"Yes"→"Finish" in order and restart your RPi later. Then the I2C module is started.

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

```
lsmod | grep i2c
```

If the I2C module has been started, the following content will be shown:

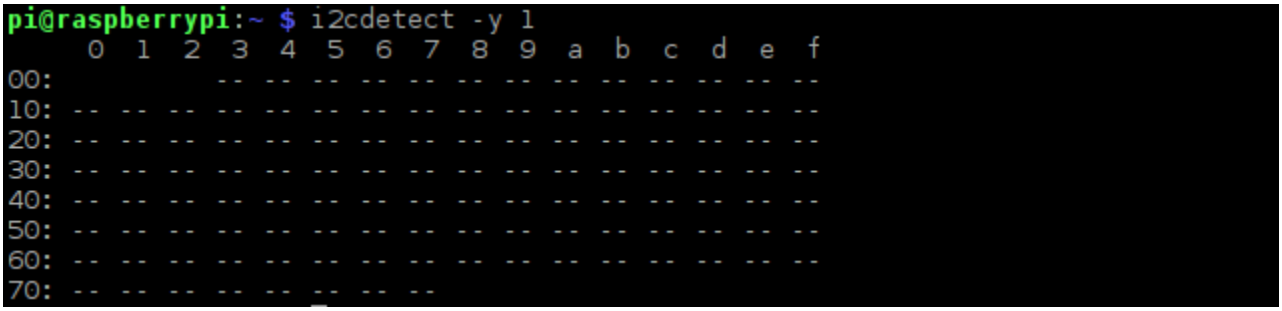
Install I2C-Tools

Type the command to install I2C-Tools.

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

I2C device address detection:

```
i2cdetect –y 1
```



If there are I2C devices connected to your RPi, here will display their I2C device address.

Install python-smbus

Python-smbus is a module of the program Python, which contains some classes and methods to operate I2C.

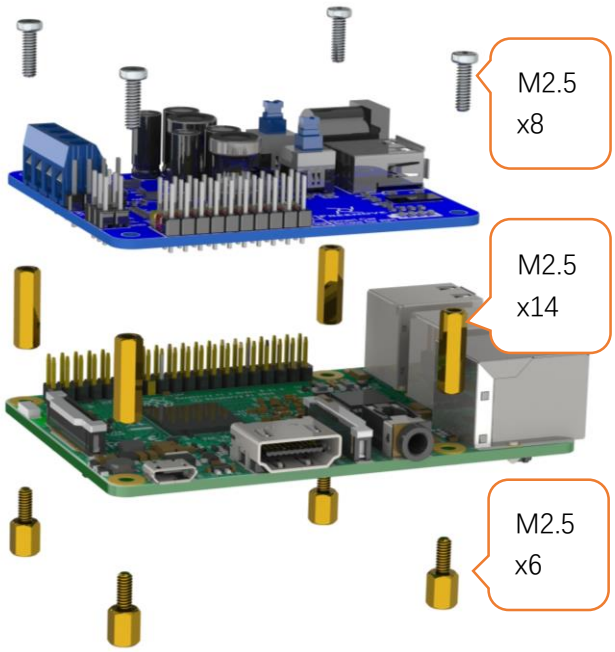
Type the following command to install python-smbus:

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

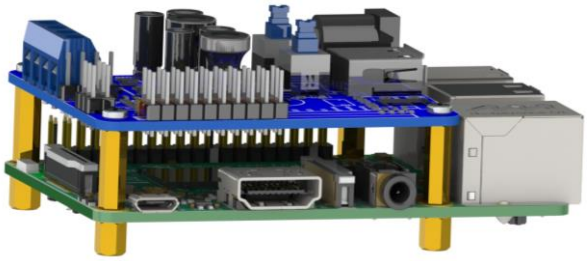
Communication test

Follow the steps below to connect the Shield with the RPi.

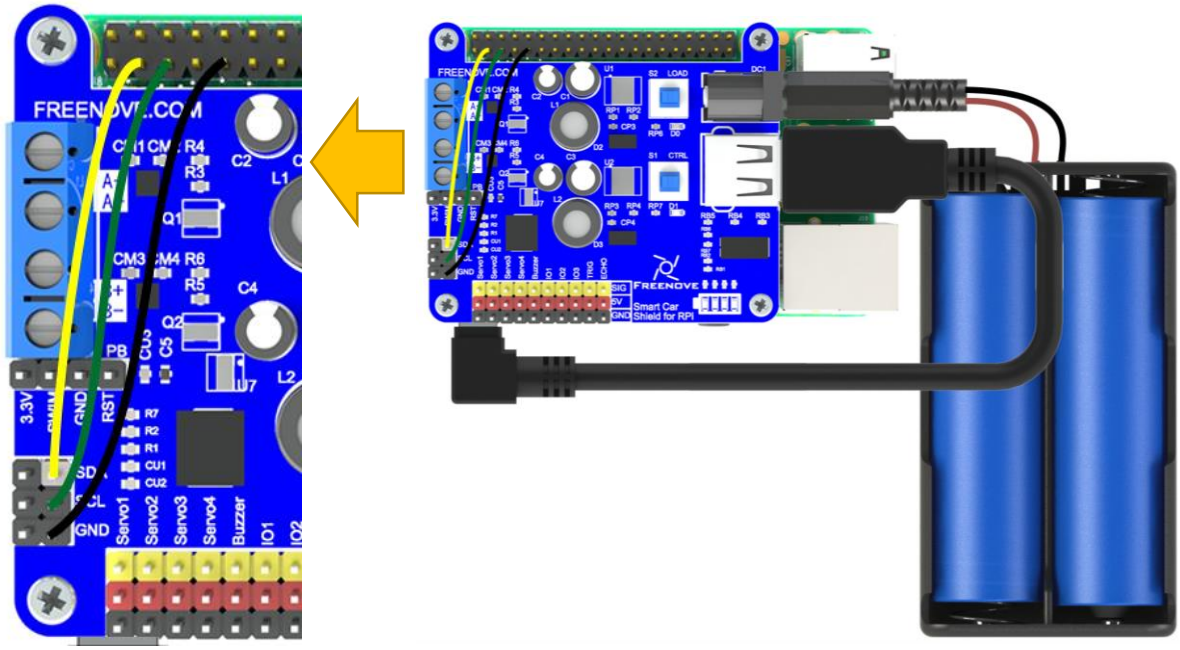
1.Prepare the following boards and parts.



2. Assembly



3.Use Jumper Wire F-F to connect the Shield with I2C port of RPi. Use the battery box to supply power for the Shield, and open the switch S1. RPi can be powered by USB power port, or external power supply adapter.



Default I2C address of the Shield is 0x18. Execute command i2cdetect-y1 again to detect whether the shield is connected to RPi successfully.

```
i2cdetect –y 1
```

```
pi@raspberrypi:~ $ i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:          -- -- -- -- -- -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- 18 -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

Step 0.3 Obtain the Code

Type the following command in the terminal to obtain the code for the smart car. And place the code in the user directory "Pi".

```
cd ~
git clone http://github.com/freenove/Freenove_Three-wheeled_Smart_Car_for_Raspberry_Pi.git
```

```
pi@raspberrypi:~ $ git clone http://github.com/freenove/Freenove_Three-wheeled_S
mart_Car_Kit_for_Raspberry_Pi.git
Cloning into 'Freenove_Three-wheeled_Smart_Car_Kit_for_Raspberry_Pi'...
remote: Counting objects: 163, done.
remote: Compressing objects: 100% (124/124), done.
remote: Total 163 (delta 34), reused 162 (delta 33), pack-reused 0
Receiving objects: 100% (163/163), 674.63 KiB | 5.00 KiB/s, done.
Resolving deltas: 100% (34/34), done.
Checking connectivity... done.
```

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).

Step 0.4 Install mjpg-streamer

Camera is drived by mjpg-streamer. So you need to install mjpg-streamer.

Install

Open the terminal and execute the following command to install.

1. Install the relay for mjpg-streamer:

```
sudo apt-get install libv4l-dev libjpeg8-dev imagemagick
```

2. Generate executable file mjpg-streamer:

```
cd ~/Freenove_Three_wheeled_Smart_Car_for_Raspberry_Pi/mjpg-streamer
make USE_LIBV4L2=true clean all
```

Test mjpg-streamer

Connect the camera to any one of the USB ports on the RPi. And execute the following command to verify that the camera is successfully connected to RPi.

```
ls /etc/video*
```

If the results list the video0, the camera is connected successfully.

```
pi@raspberrypi:~ $ ls /dev/video*
/dev/video0
```

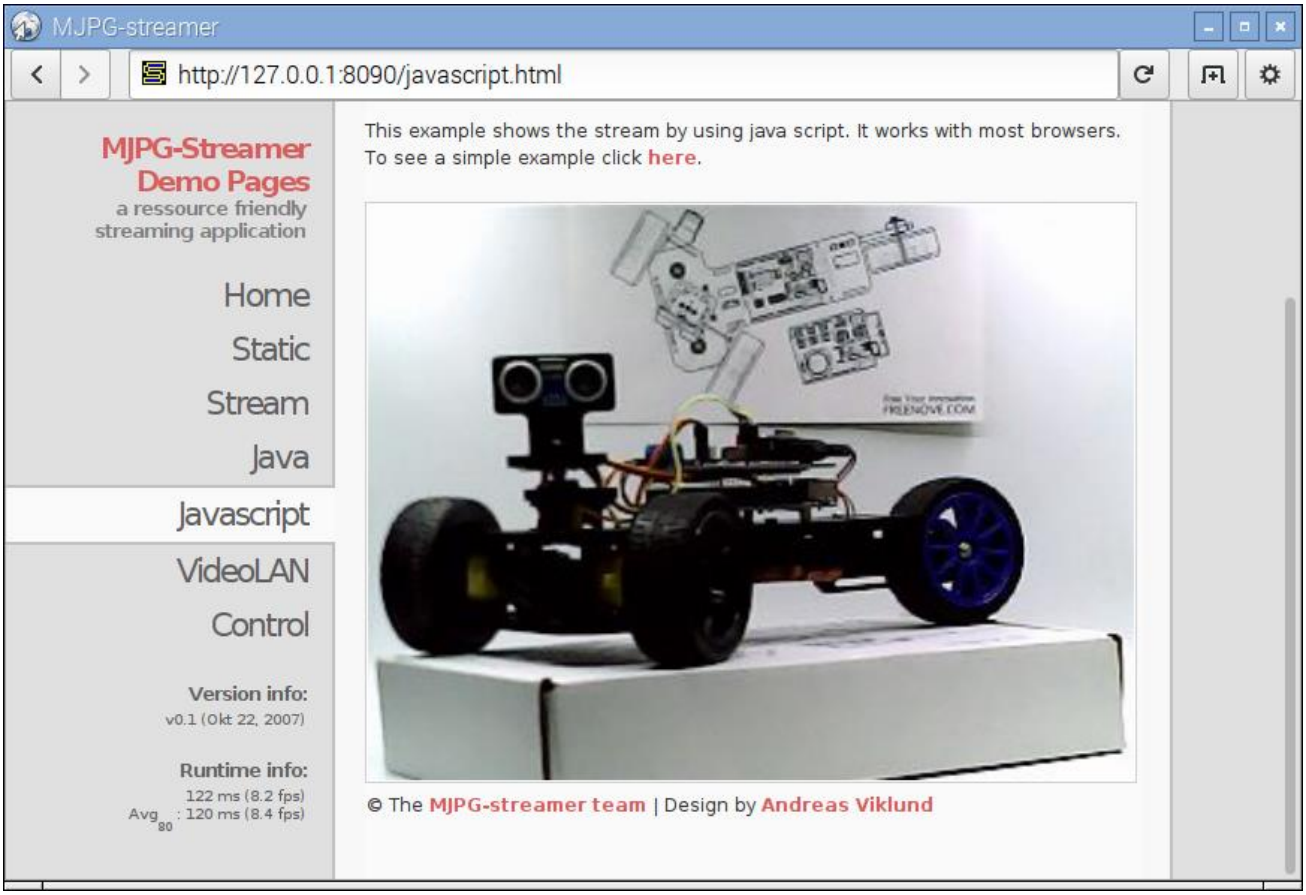
Under the mjpg-streamer directory, execute the following command to open the mjpg-streamer service.

```
sh Start_mjpg_Streamer.sh
```

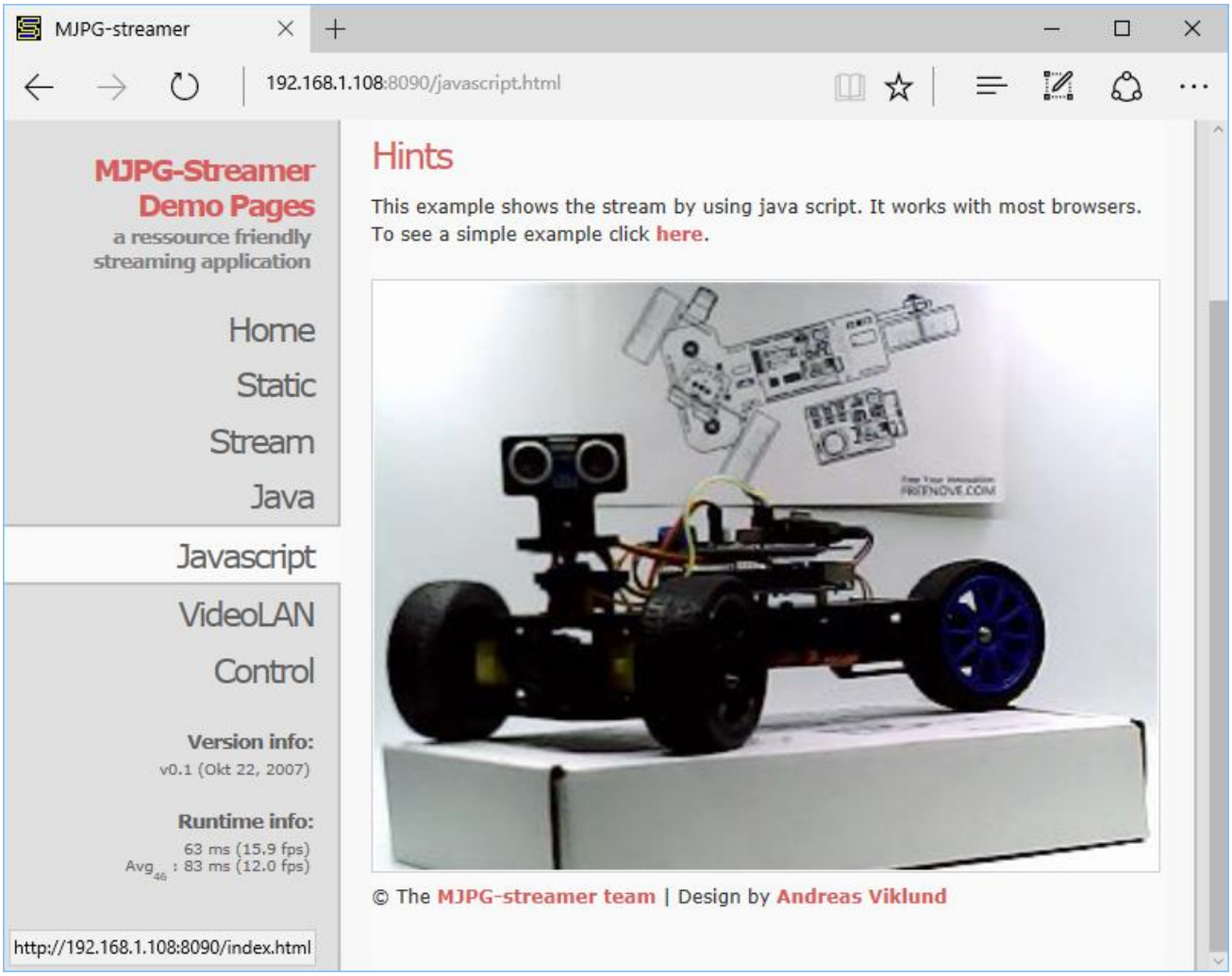
```
pi@raspberrypi:~/Freenove_Three-wheeled_Smart_Car_Kit_for_Raspberry_Pi/mjpg-stre
amer $ sh Start_mjpg_Streamer.sh
MJPG Streamer Version: svn rev: Unversioned directory
 i: Using V4L2 device.: /dev/video0
 i: Desired Resolution: 320 x 240
 i: Frames Per Second.: 30
 i: Format............: YUV
 i: JPEG Quality......: 80
 o: www-folder-path...: ./www/
 o: HTTP TCP port.....: 8090
 o: username:password.: disabled
 o: commands..........: enabled
```

Open Web browser of RPi, access to http://127.0.0.1:8090/ or http://localhost:8090/. Then the following picture appears.

Click the "Javascript" tab on the left of navigation bar, then you will be see the real-time picture of the camera. If the picture is not clear enough, you can rotate the lens to adjust the focus.



You can access this page through accessing http://xxx.xxx.xxx.xxx:8090/ with your Web browser of your PC or mobile phone. This requires your PC or mobile phone to be in the same local area network with your RPi, where xxx.xxx.xxx.xxx is IP address of RPi. For example, my RPi IP address is 192.168.1.108. In the Windows 10, access to http://192.168.1.108:8090/ through the browser, as is shown below.



Step 0.5 Install PyQt4

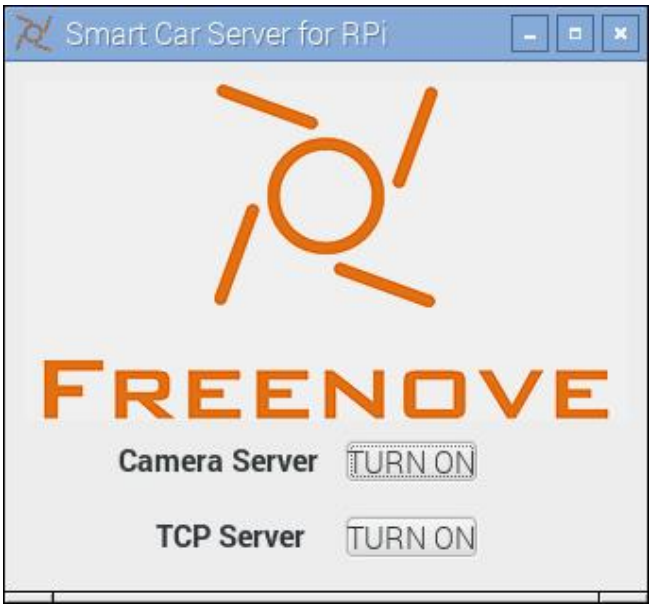
The project code is based on PyQt4. So operation of the program requires the support of PyQt4.
Open the terminal and execute the following command to install PyQt4.

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

After the installation completed, type the following command to test whether PyQt4 is installed successfully.

```
cd ~/Freenove_Three_wheeled_Smart_Car_for_Raspberry_Pi/Server
python main.py
```

If you can run it successfully, and the following picture appears, it means that PyQt4 has been successfully installed. Then click on the top right corner to close the program.



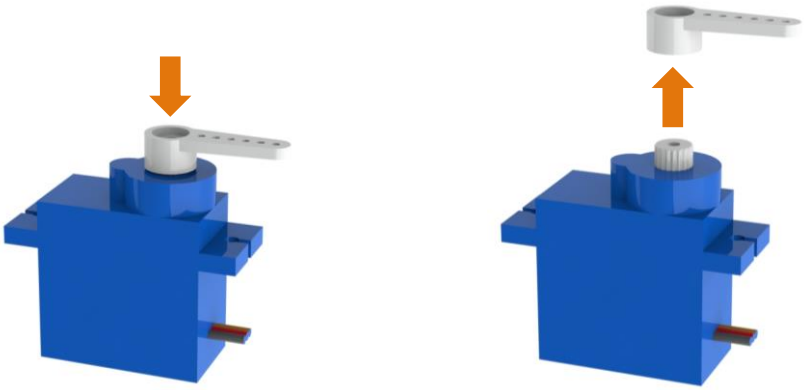
Step 0.6 Test

Next, test the servo, motor, buzzer, RGBLED module and so on.

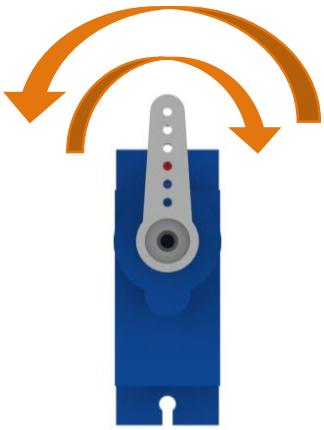
Servo

The servo can be connected with rocker arm to drive other parts to move. There are 3 kinds of rocker arm, and 3 screws for the servo. The smaller screw is used to fix the rocker arm onto servo.

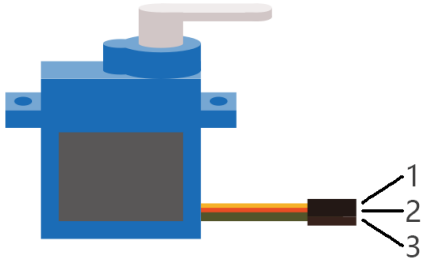
You can install or remove the rocker arm as below:

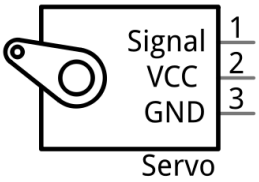


You can turn the rocker arm to rotate in the range from 0 to 180 degrees with hand:

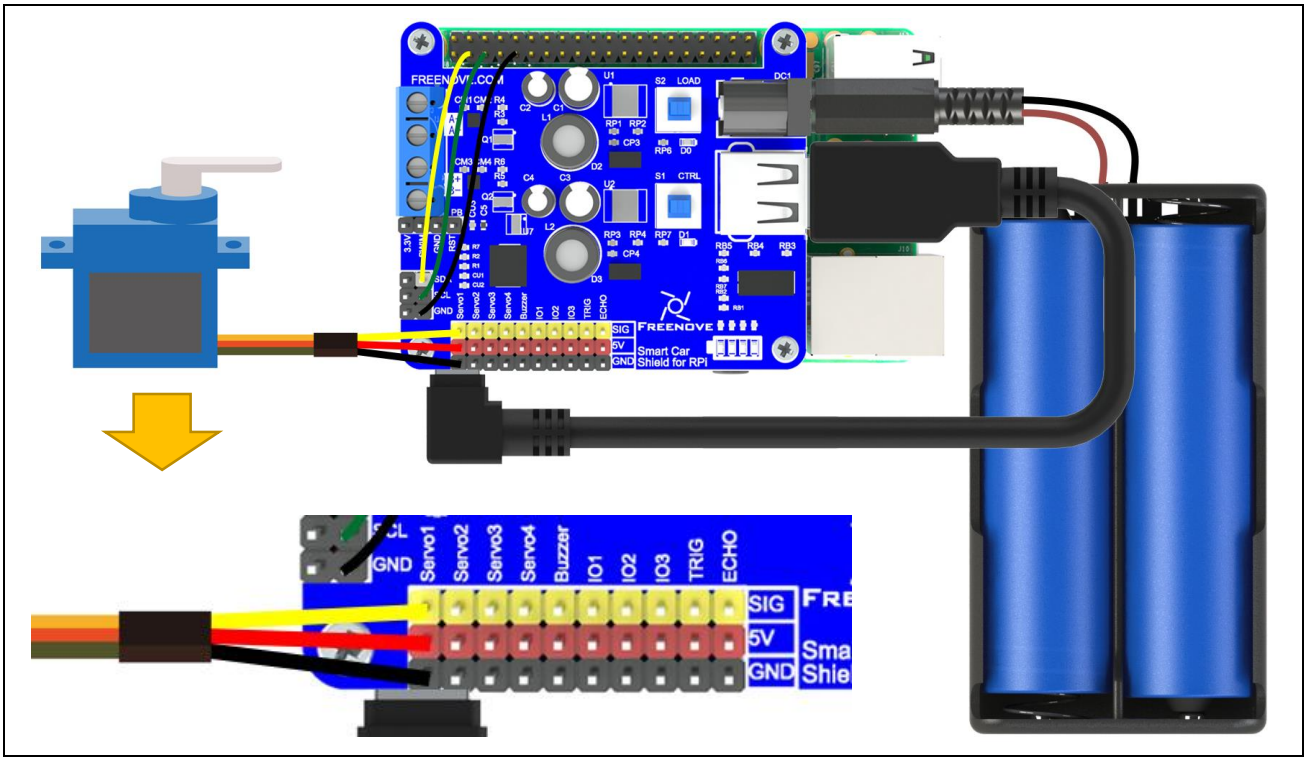


Servo has three lines for controlling. The yellow one is for the signal line, the red line for the power VCC, the black one for the power GND.





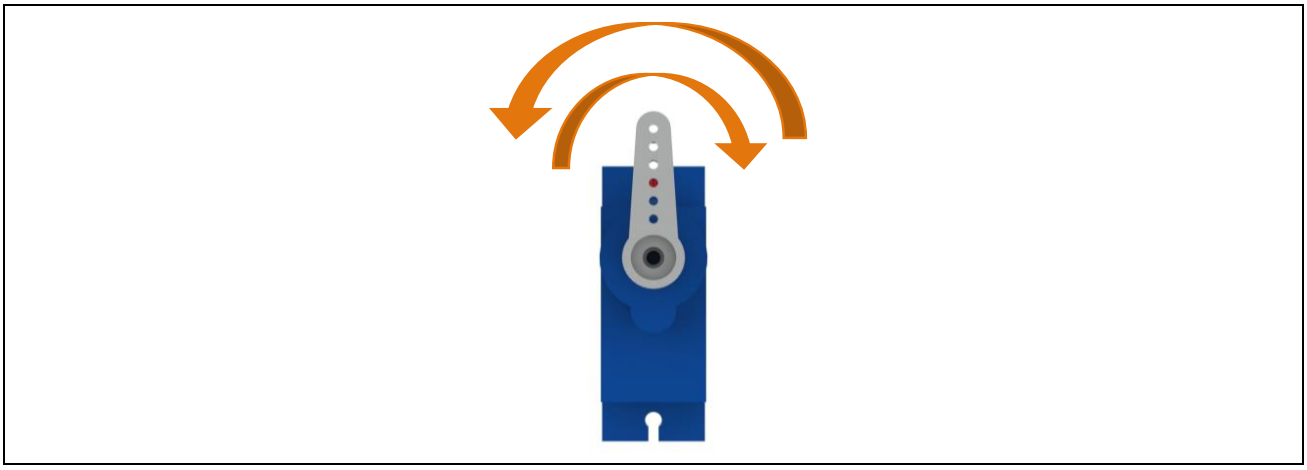
According to the following steps, connect any one of servos to the Servo1 port of the Shield. Open the switch S2, then the servo will rotate to 90 degrees automatically.



Type the following command in the terminal to test the servo:

```
cd ~/Freenove_Three_wheeled_Smart_Car_for_Raspberry_Pi/Server
python mDev.py servo
```

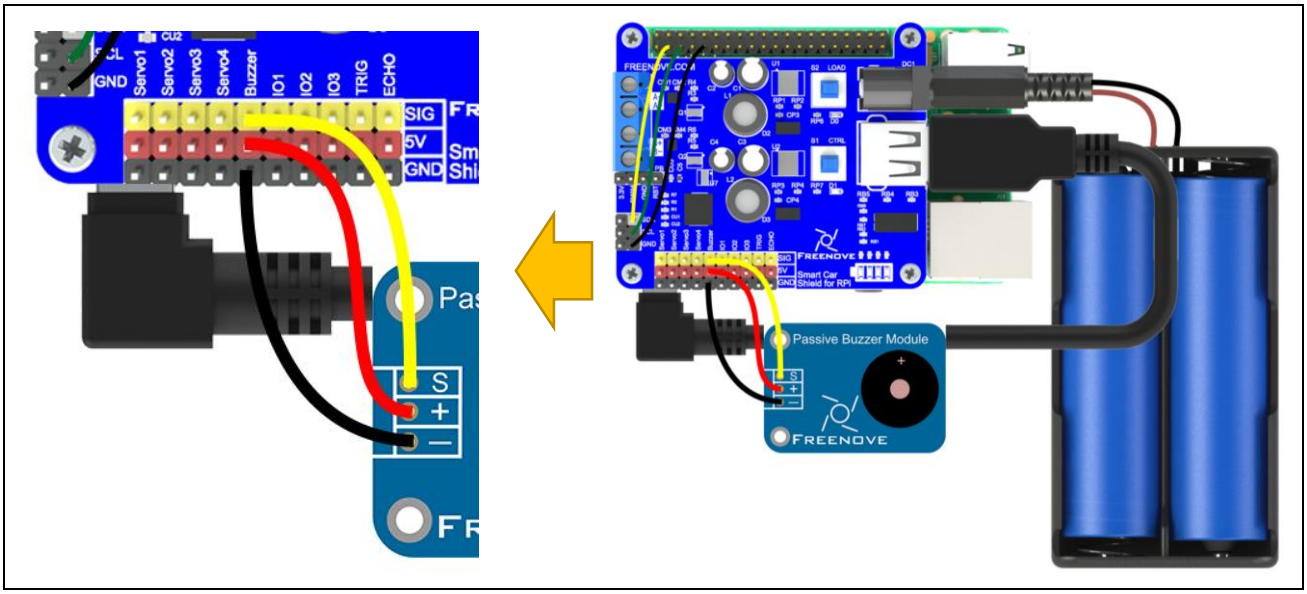
Then, the servo will rotate back and forth in a certain range. Press Ctrl+C to terminate the program.



After the test is completed, close switch S2 then reopen it. Wait the servo rotating to 90 degrees to stop, then pull off the servo line. After that, do not rotate the servo manually not to affect the following installation. Make all other servos to rotated to 90 degrees according to this method.

Buzzer

Connect the buzzer module to the buzzer port in the Shield.

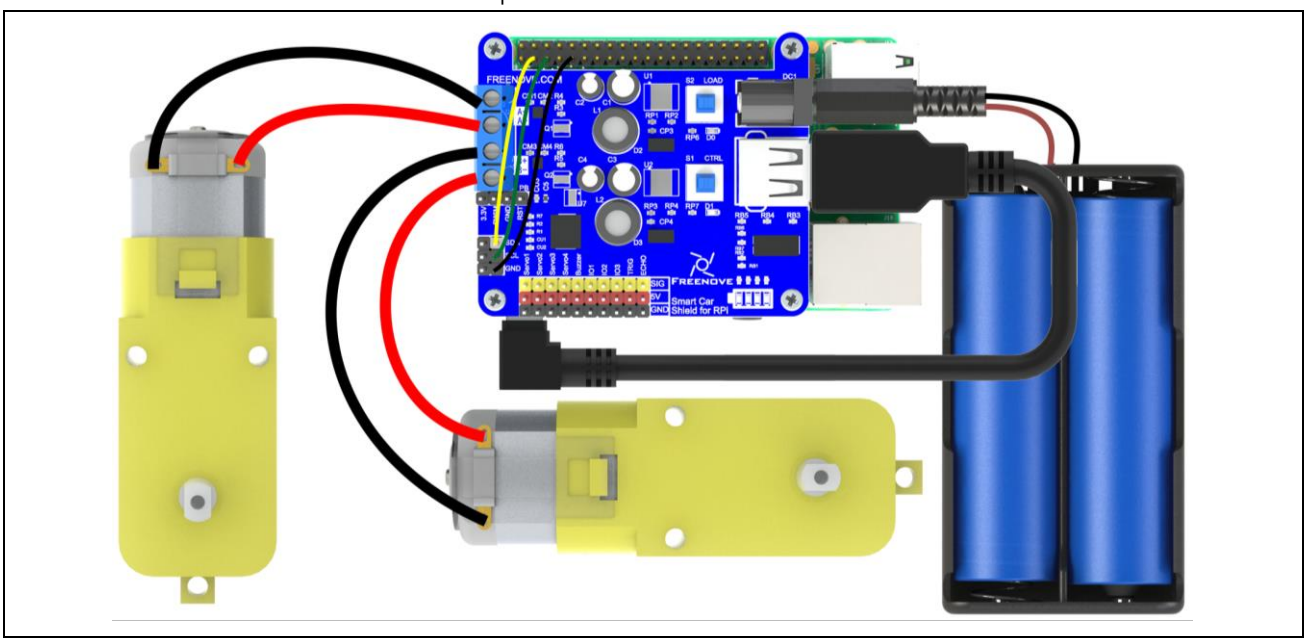


Type the following command. Then the buzzer will start to sound. The program will exit 3 seconds later, then the buzzer stop sounding.

```
cd ~/Freenove_Three_wheeled_Smart_Car_for_Raspberry_Pi/Server
python mDev.py buzzer
```

Motor

Connect the motor module to the motor port in the Shield.

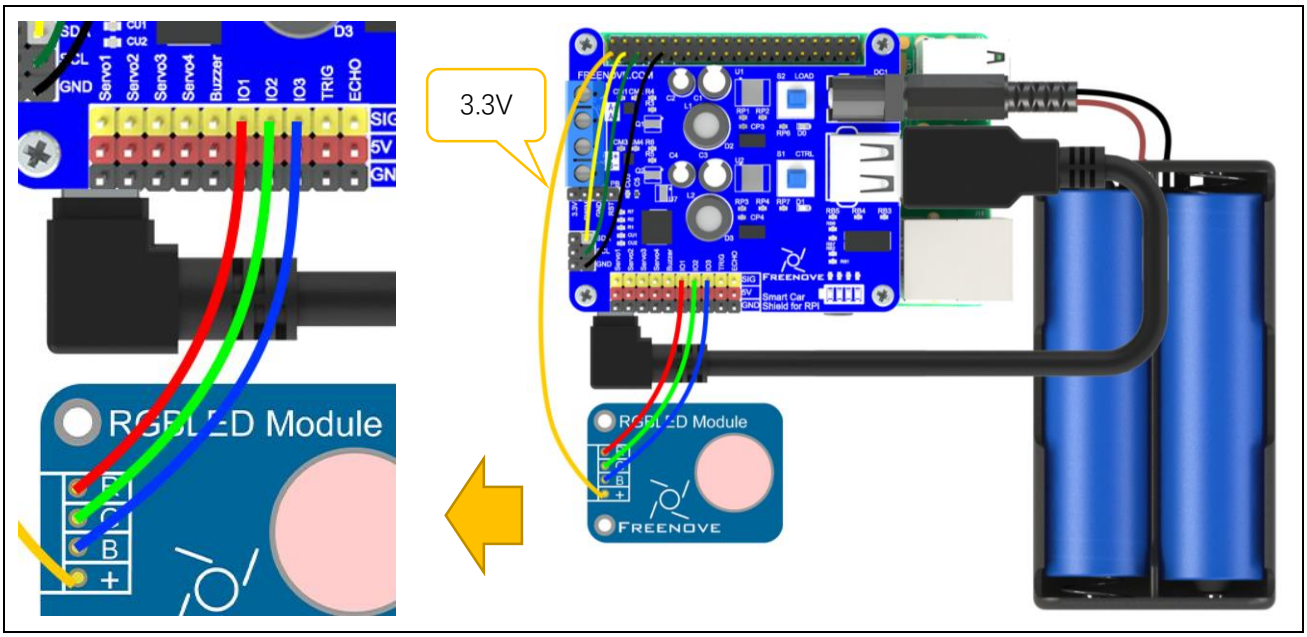


Type the following command, then the servo will rotate back and forth in a certain range. Press Ctrl+C to terminate the program.

```
cd ~/Freenove_Three_wheeled_Smart_Car_for_Raspberry_Pi/Server
python mDev.py motor
```

RGBLED Module

Connect pin R, G, B of RGBLEG Module to port IO1, IO2, IO3 of the Shield respectively. Connect "+" to 3.3V of RPi.

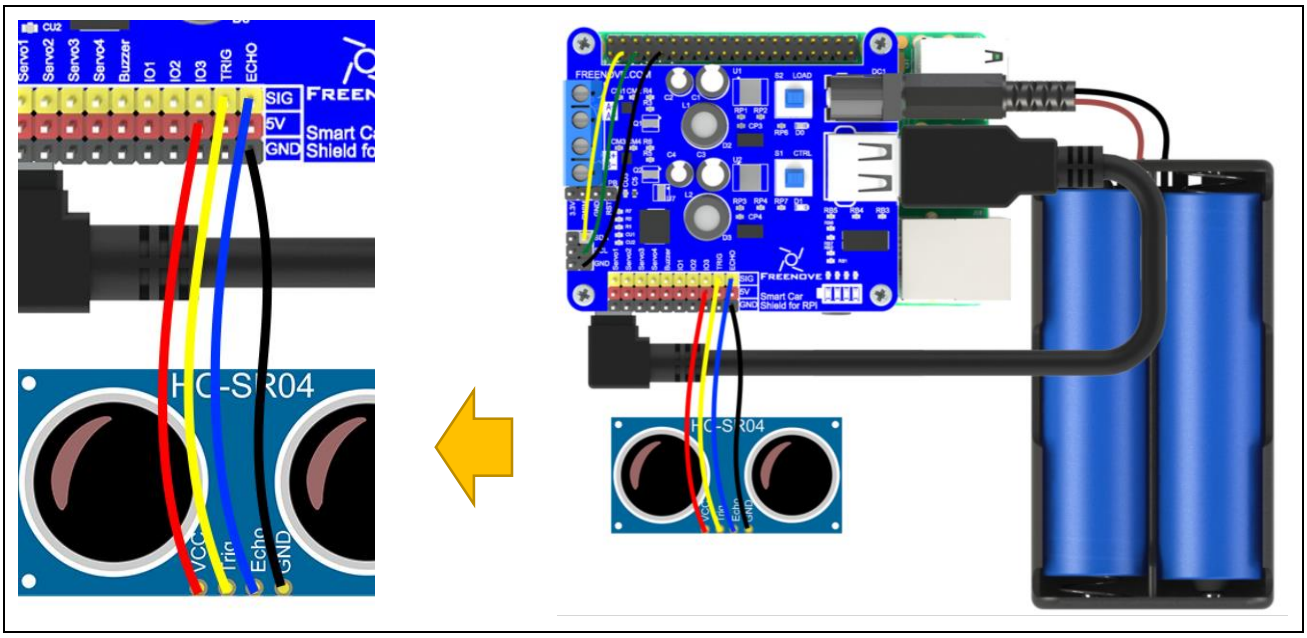


Type the following command, RGBLED will emit red, green, and blue light several times circularly. Then the program exits, the RGBLED is turned off.

```
cd ~/Freenove_Three_wheeled_Smart_Car_for_Raspberry_Pi/Server
python mDev.py RGBLED
```

Ultrasonic Module

Connect pin VCC, GND, TRIG, ECHO of Ultrasonic Module to port 5V, GND, TRIG, ECHO of the Shield respectively:



Type the following command to print out the ultrasonic measurement data in the terminal.

```
cd ~/Freenove_Three_wheeled_Smart_Car_for_Raspberry_Pi/Server
python mDev.py Ultrasonic
```

Step 0.7 Server and Client

The entire project code contains two parts: the server and the client. The server runs on RPi. The client runs on other devices, and of course, it can also run on the same RPi with the server. The device that the client runs on requires the installation of Python and PyQt4. If you need to run the client on Linux system, you need install PyQt according to Step 0.4. If you want to run the client on windows, you need install the program and service according to the following steps.

Run the cilent on Windows OS

Install Python2.7

Access https://www.python.org/downloads/windows/, select the latest version of Python2.7 and download. As shown below, the current (2016-11-16) latest version of Python2.7 is Python2.7.12. Click to enter the download page.

Then enter Python2.7.12 download list page. Select the corresponding file according to your operating system.

Files

Version	Operating System
Gzipped source tarball	Source release
XZ compressed source tarball	Source release
Mac OS X 32-bit i386/PPC installer	Mac OS X
Mac OS X 64-bit/32-bit installer	Mac OS X
Windows debug information files	Windows
Windows debug information files for 64-bit binaries	Windows
Windows help file	Windows
Windows x86-64 MSI installer	Windows
Windows x86 MSI installer	Windows

After download is completed, double-click to install.

Install PyQt4

Visit https://riverbankcomputing.com/software/pyqt/download,
or visit https://sourceforge.net/projects/pyqt/files/PyQt4/PyQt-4.11.4/,
choose the PyQt4-4.11.4-gpl-Py2.7-Qt4.8.7-xxx.exe to download. Select x32 or x64 according to your operating system.

PyQt4-4.11.4-gpl-Py2.7-Qt4.8.7-x64.exe	Windows 64-bit installer
PyQt4-4.11.4-gpl-Py2.7-Qt4.8.7-x32.exe	Windows 32-bit installer

After download is completed, double-click to install, and the installation path will be selected to Python installation directory automatically.

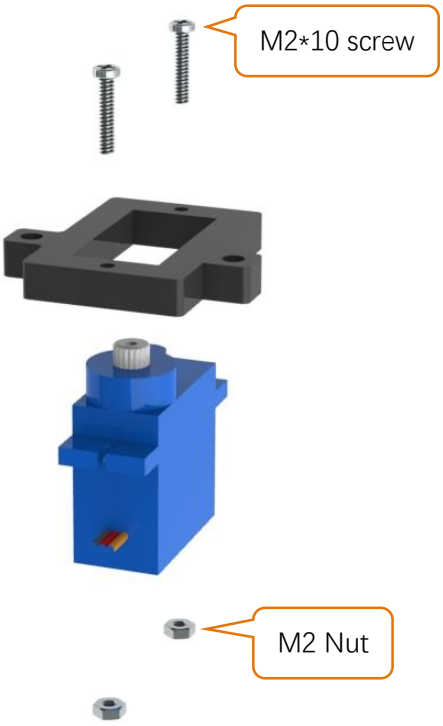
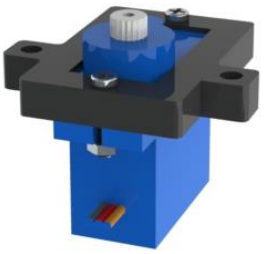
Chapter 1 Smart video car

First, let's make a smart video car.

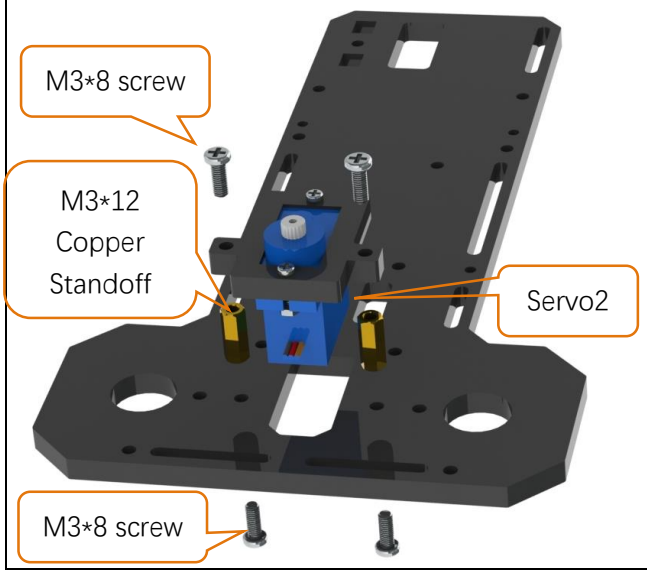
Assembly

Pan-tilt

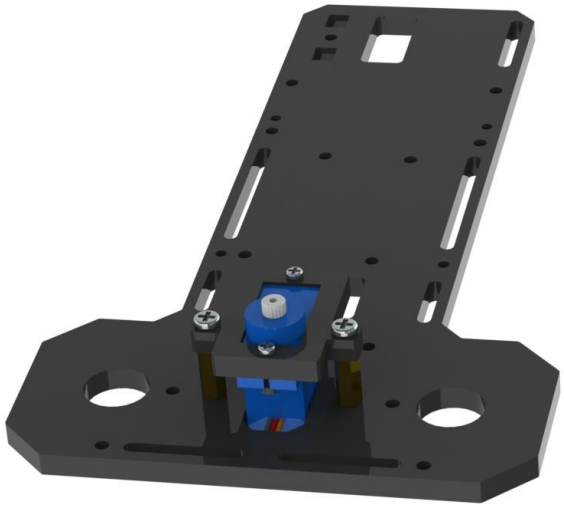
Assemble servo2

1.Assemble the following components	After assembling
	

2.Assemble the following components

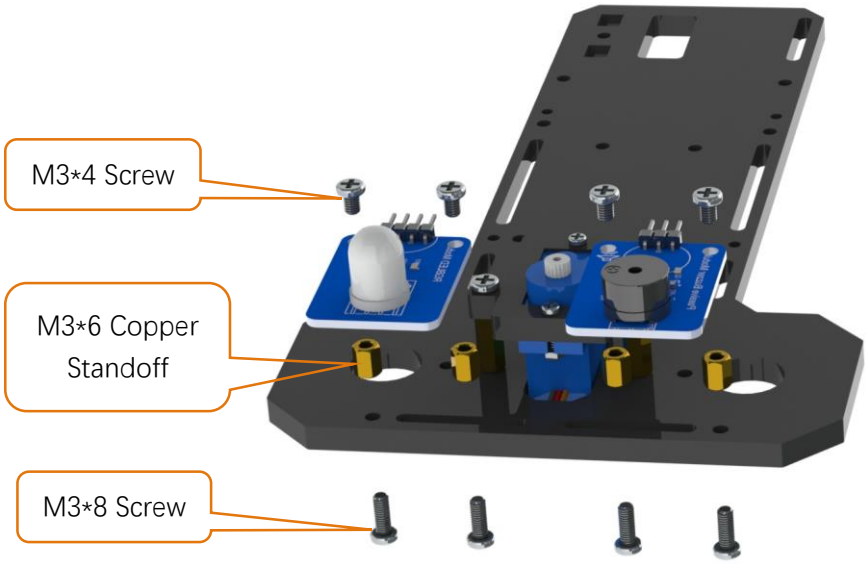


After assembling

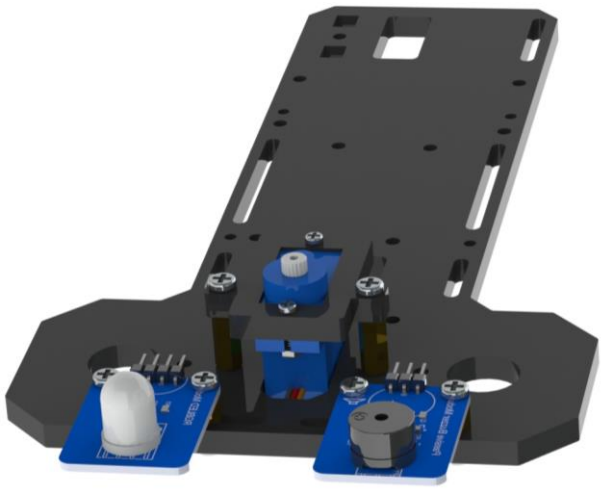


Assemble RGBLED Module and Passive Buzzer Module

Assemble the following components

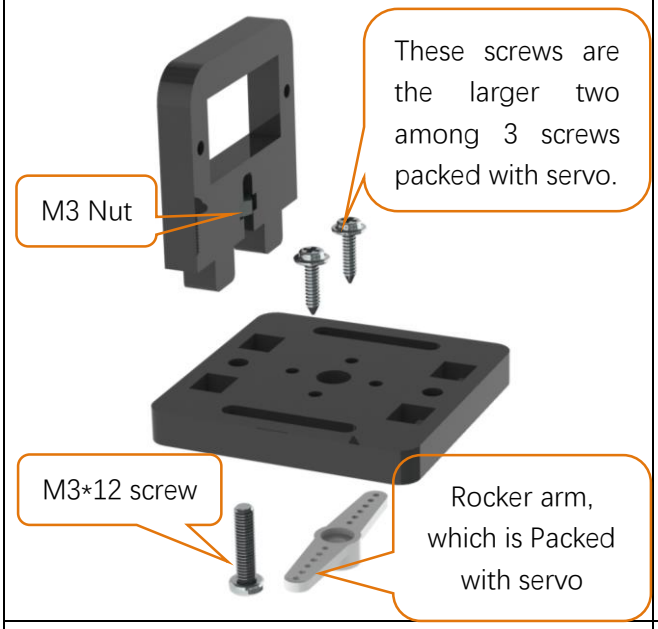


After assembling



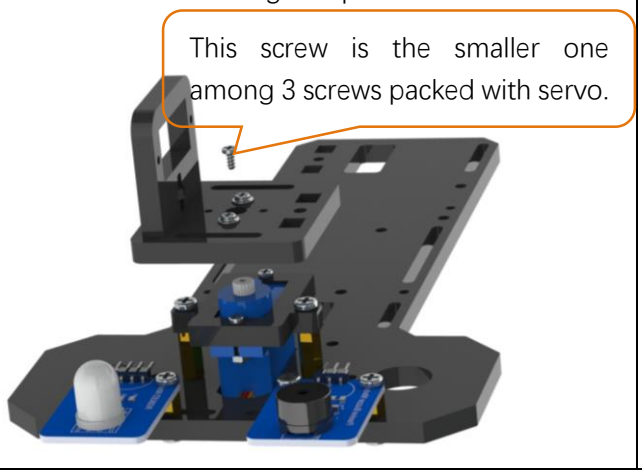
Assemble servo3 support

Assemble the following components

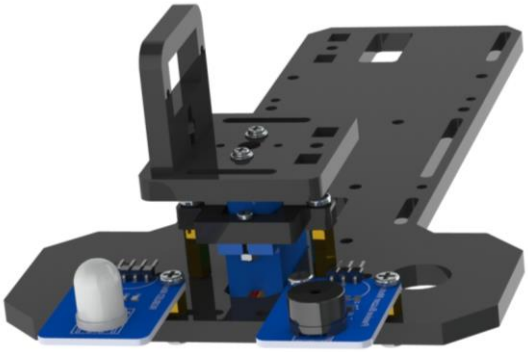


After assembling

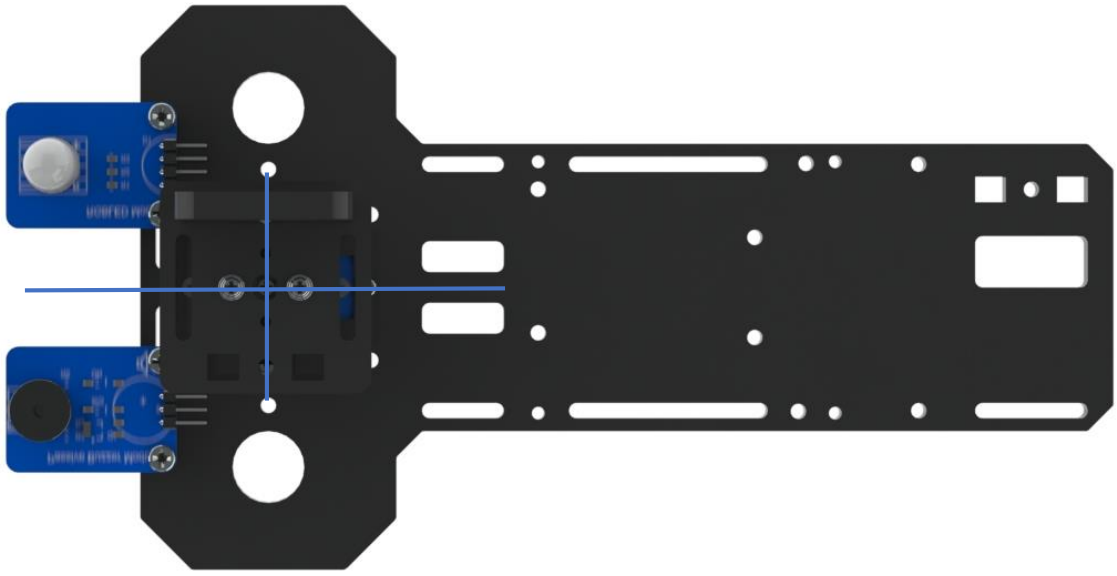
Assemble the following components



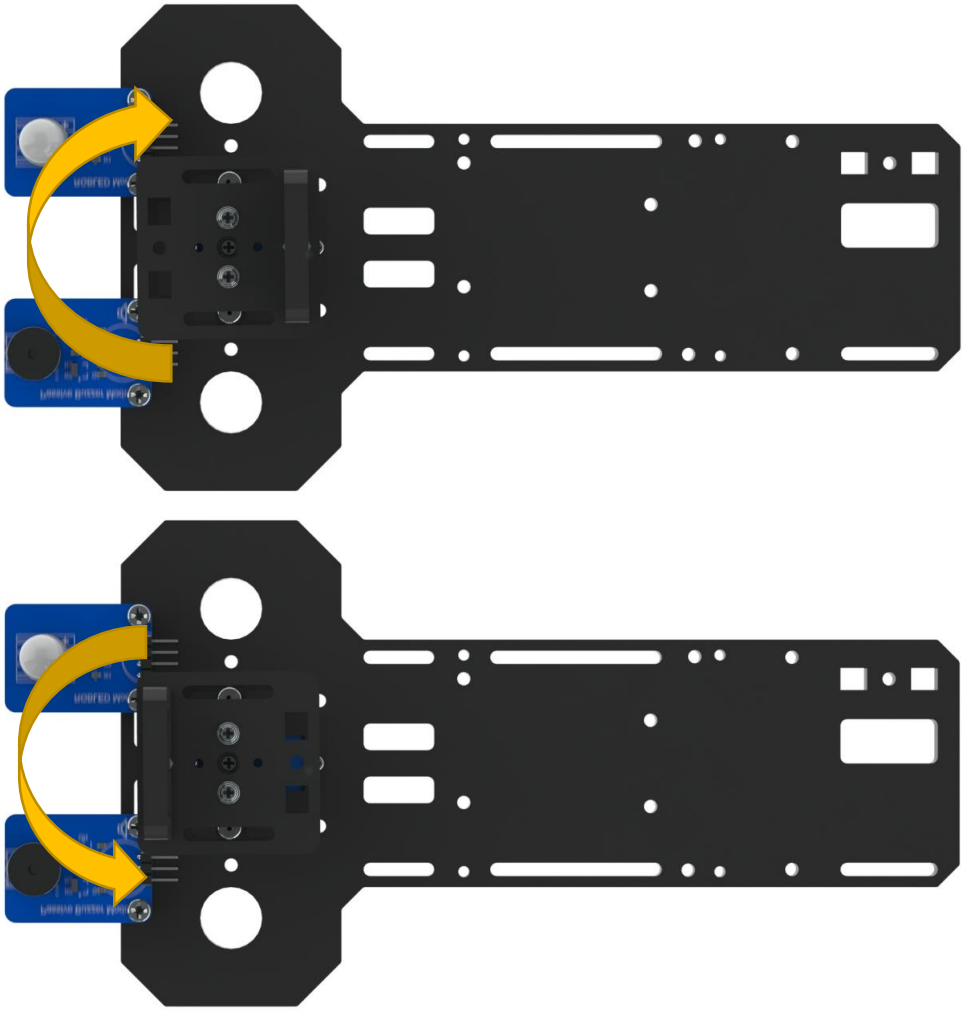
After assembling



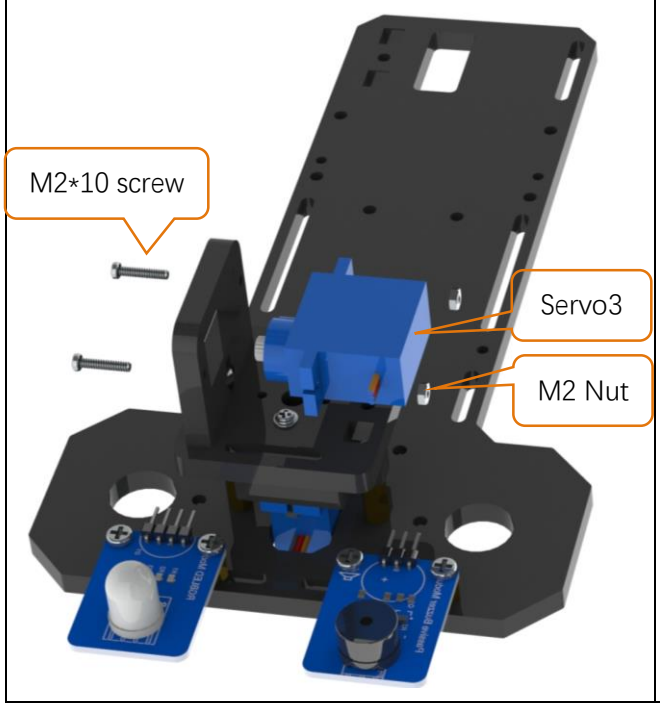
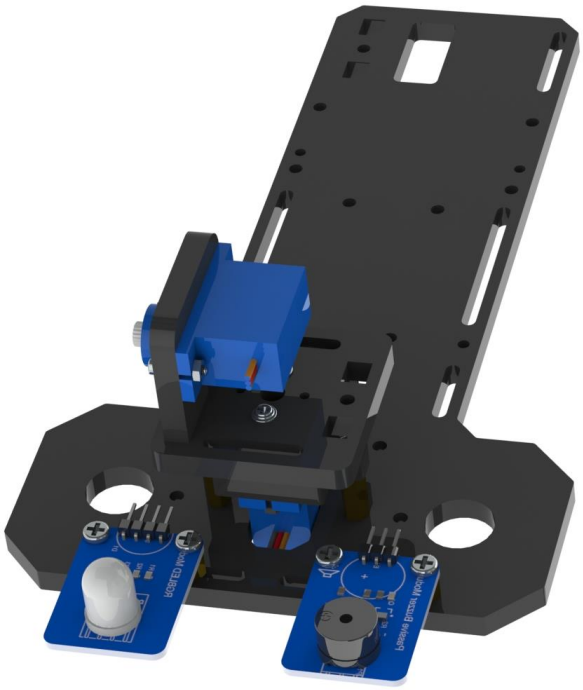
Keep the servo2 rotating to 90 degrees. If the drection is changed before installation, please make it rotate to the 90 degrees with the previous method. Complete the installation according to the direction below.



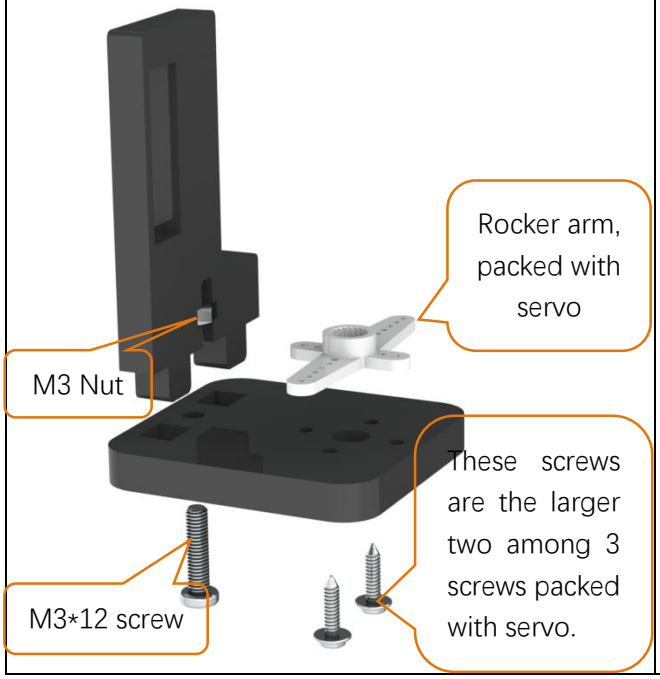
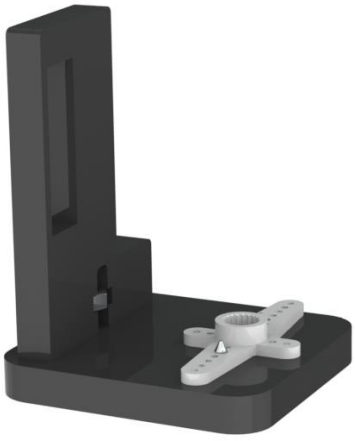
After correct assembly, the bracket can rotate 90 degrees to left or right.



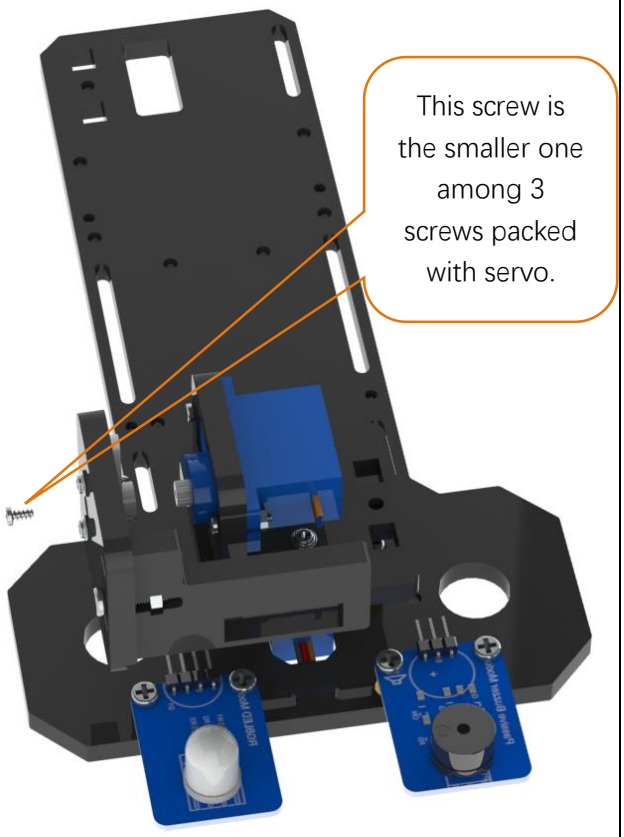
Assemble servo3

Assemble the following components	After assembling
	

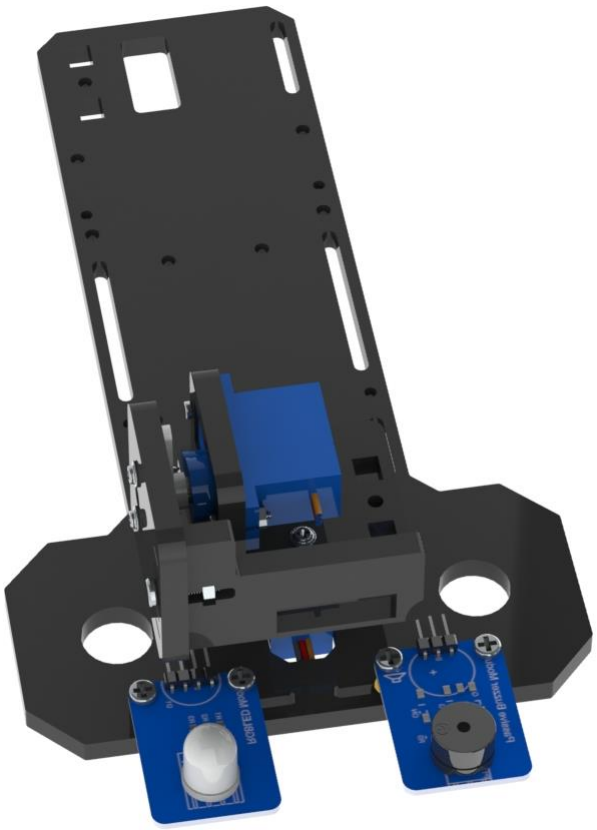
Assemble camera support

Assemble the following components	After assembling
	

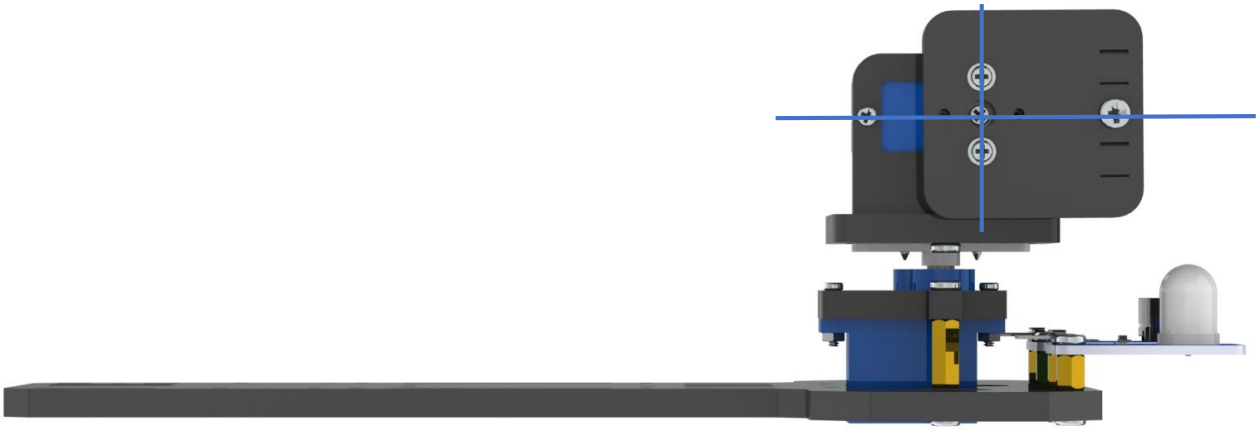
Assemble the following components



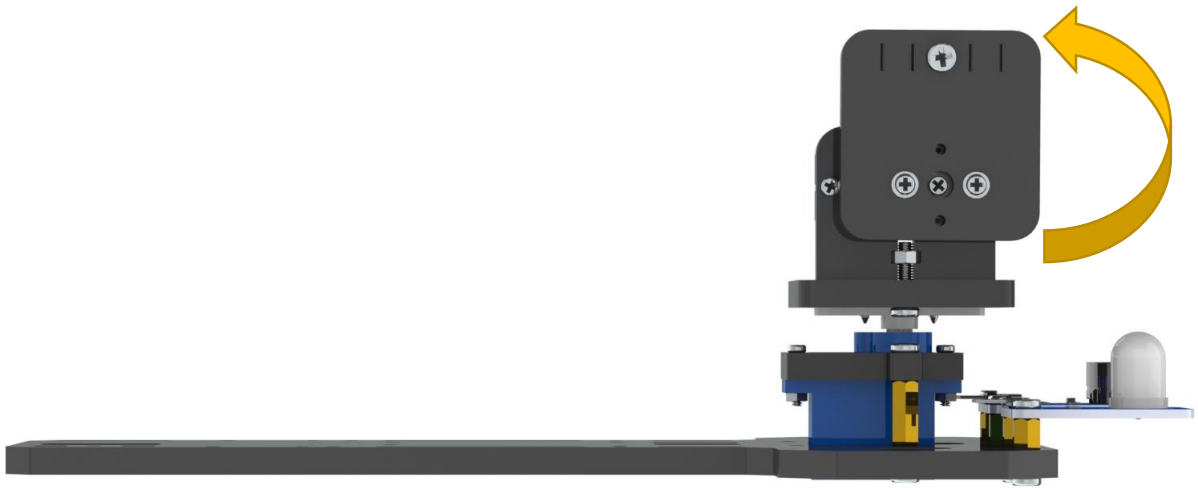
After assembling



Keep the servo3 rotating to 90 degrees. If the drection is changed before installation, please make it rotate to the 90 degrees with the previous method. Complete the installation according to the direction below.



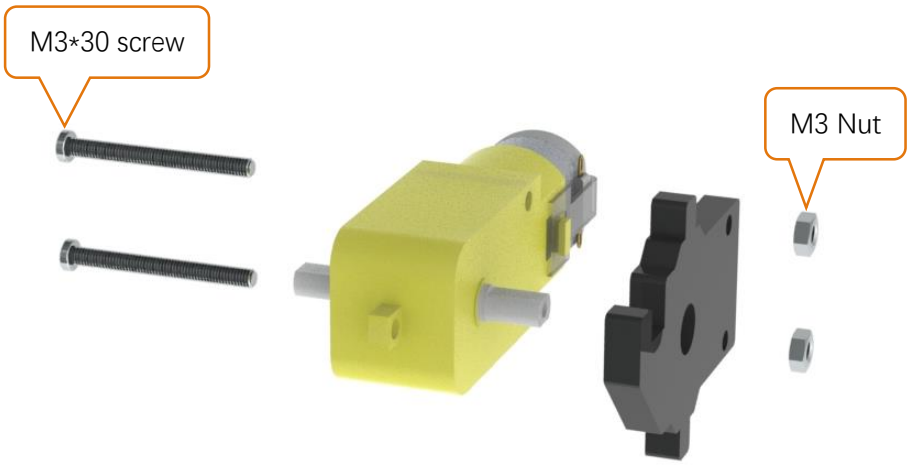
After the correct assembly, the support can rotate up to 90 degrees, as shown below.



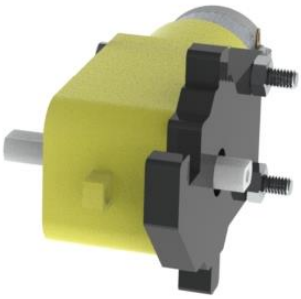
Car body

Driving wheel and motor

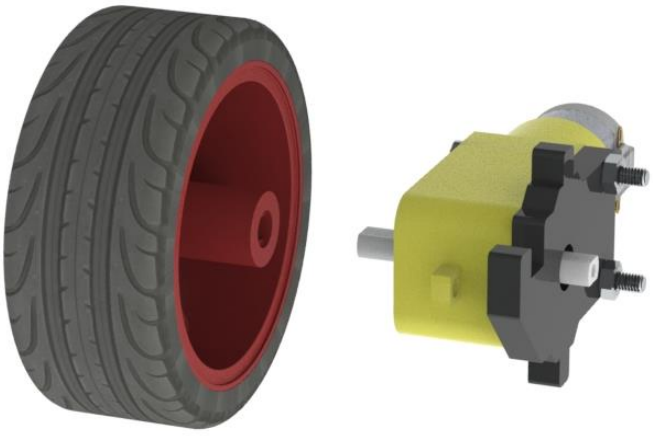
1.Assemble the following components



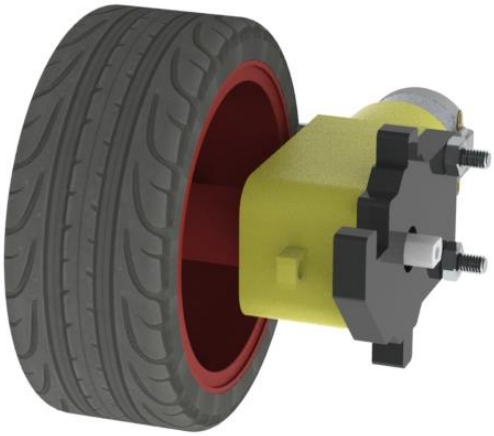
After assembling



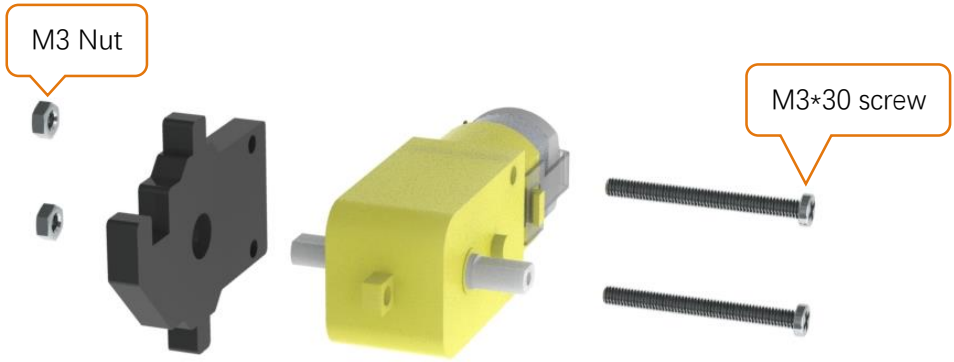
2.Install the wheel(Left)



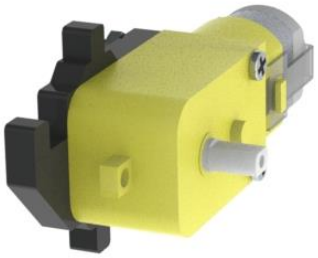
After assembling



3.Assemble the following components



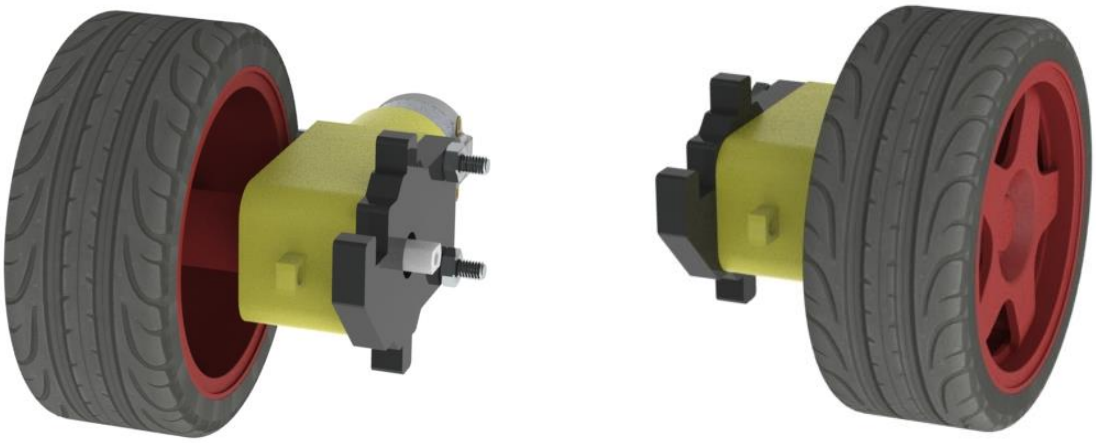
After assembling



4.Install the wheel(Right)

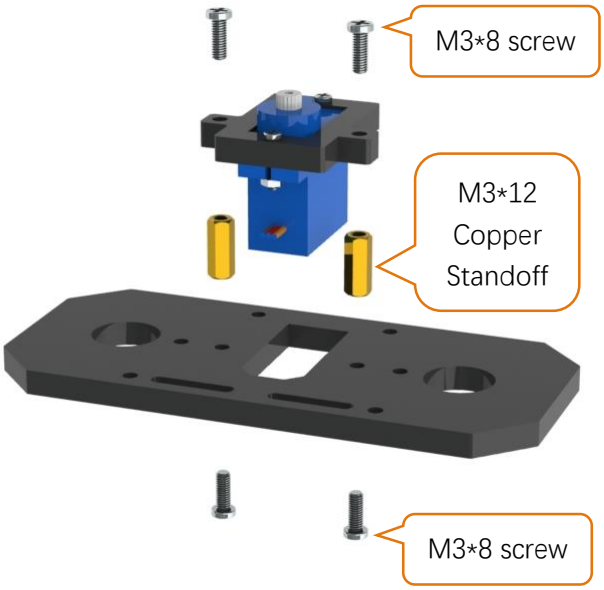
After assembling

After finishing the previous installing, you'll get two parts as follows:

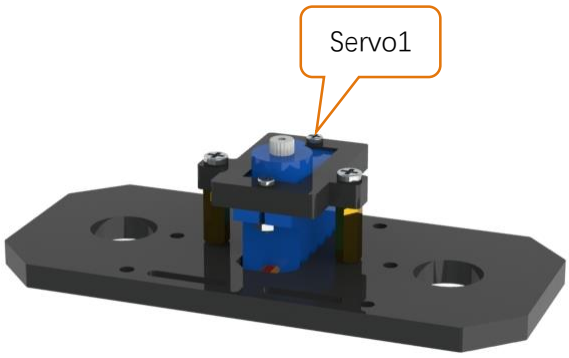


Assemble servo1

Assemble the following components

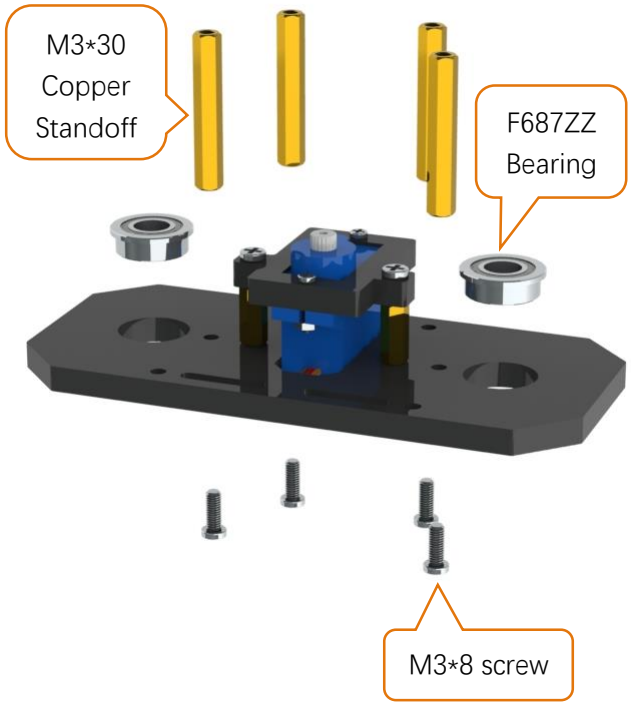


After assembling

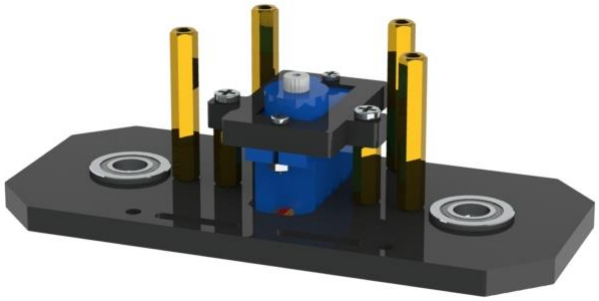


Assemble Copper Standoff

Assemble the following components

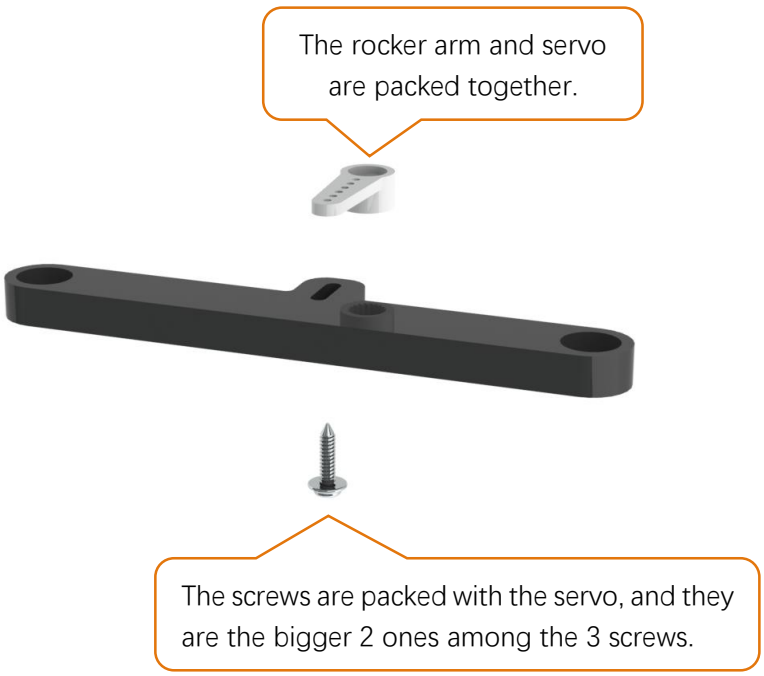


After assembling

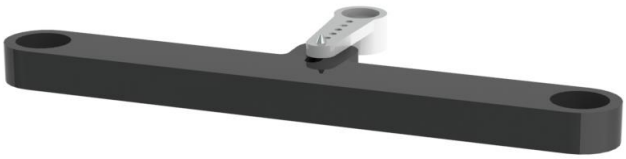


Assemble Direction stick

Assemble the following components



After assembling

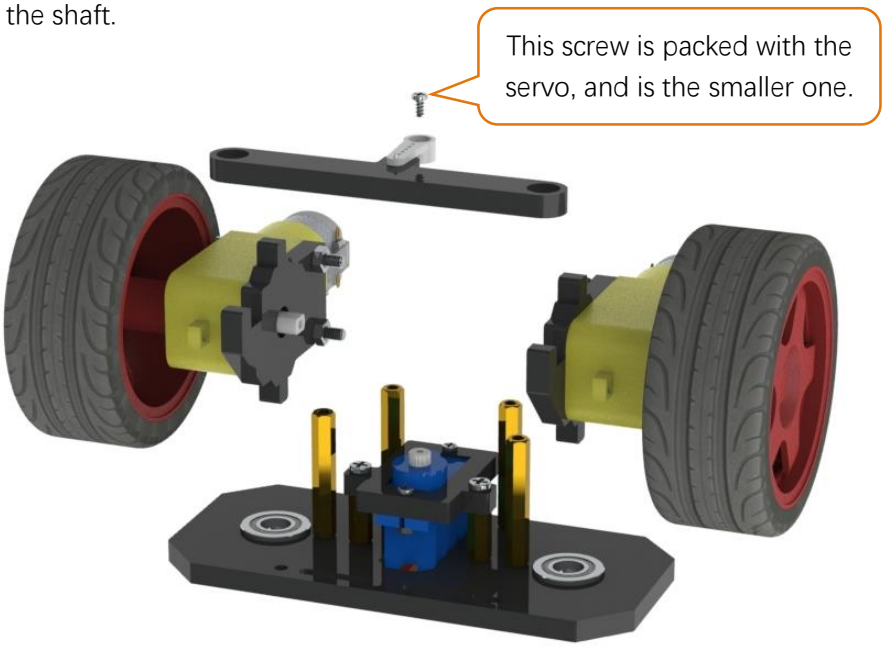


Pay attention not to tighten the screw, otherwise the acrylic plate can't move freely.

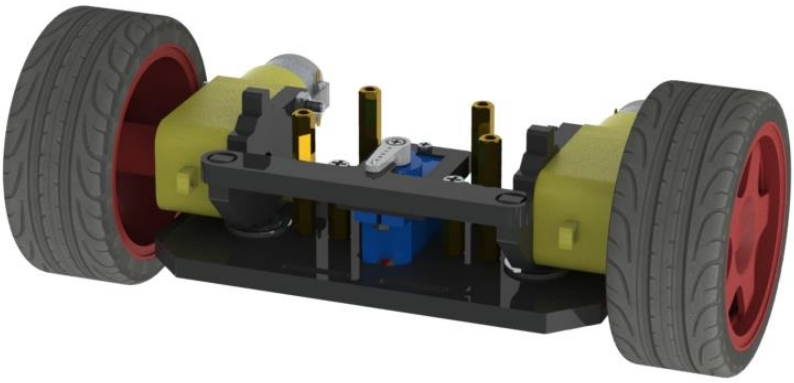
Assembly car head

Assemble the following components

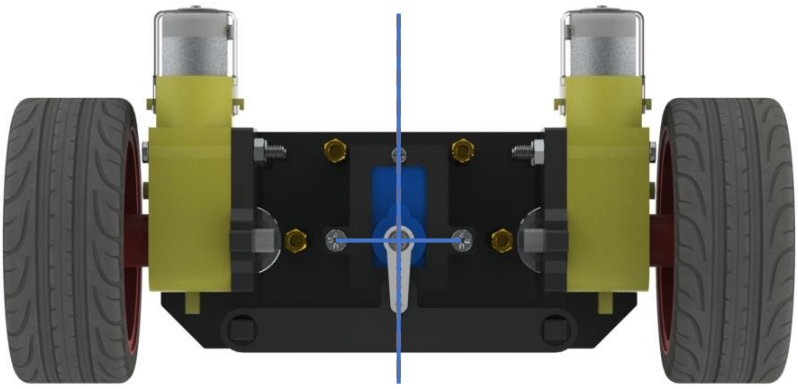
The rocker arm should be installed in the middle position between its rotation range. A little deviation is acceptable. If it is not installed in the middle position, you should remove the rocker arm and install it again instead of turning the shaft.



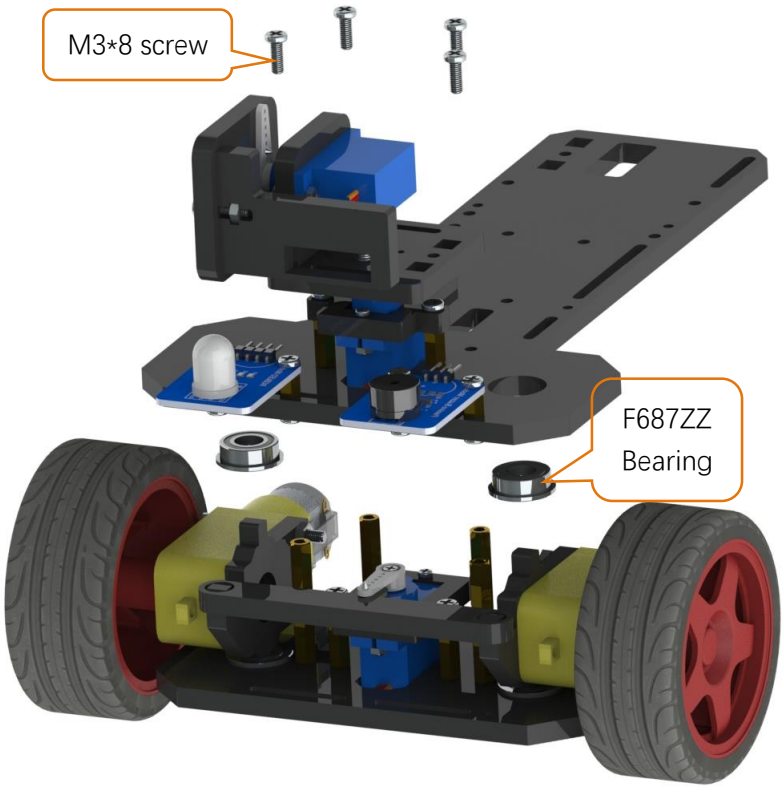
After assembling



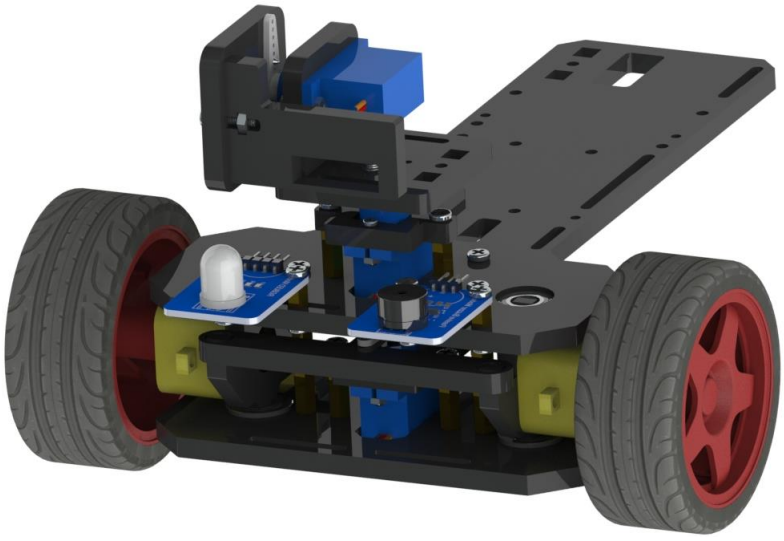
Keep the servo1 rotating to 90 degrees. If the drection is changed before installation, please make it rotate to the 90 degrees with the previous method. Complete the installation according to the direction below.



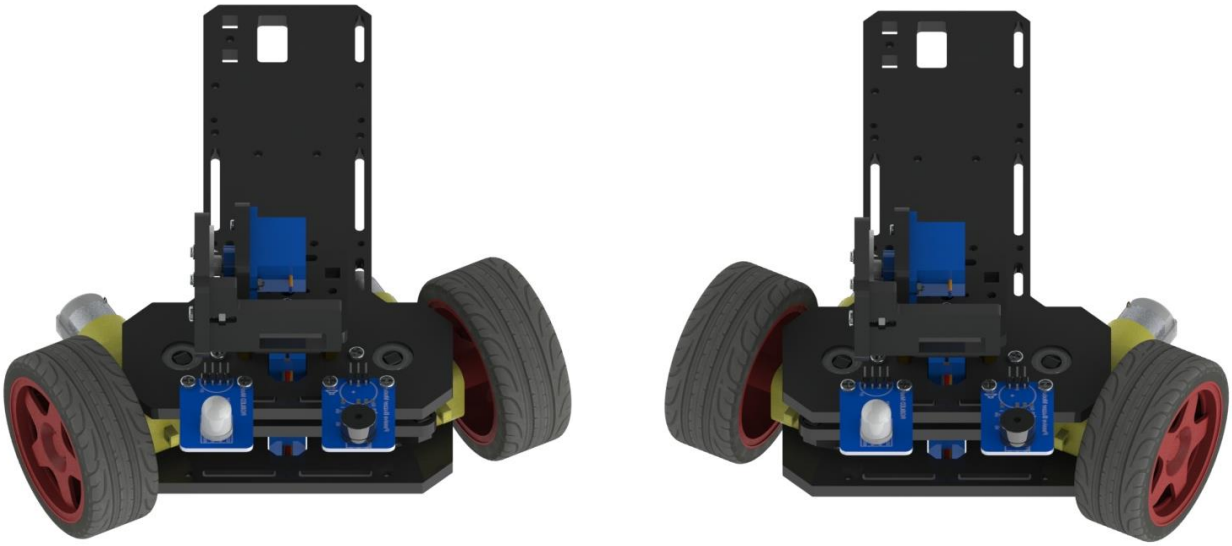
Assemble the following components



After assembling

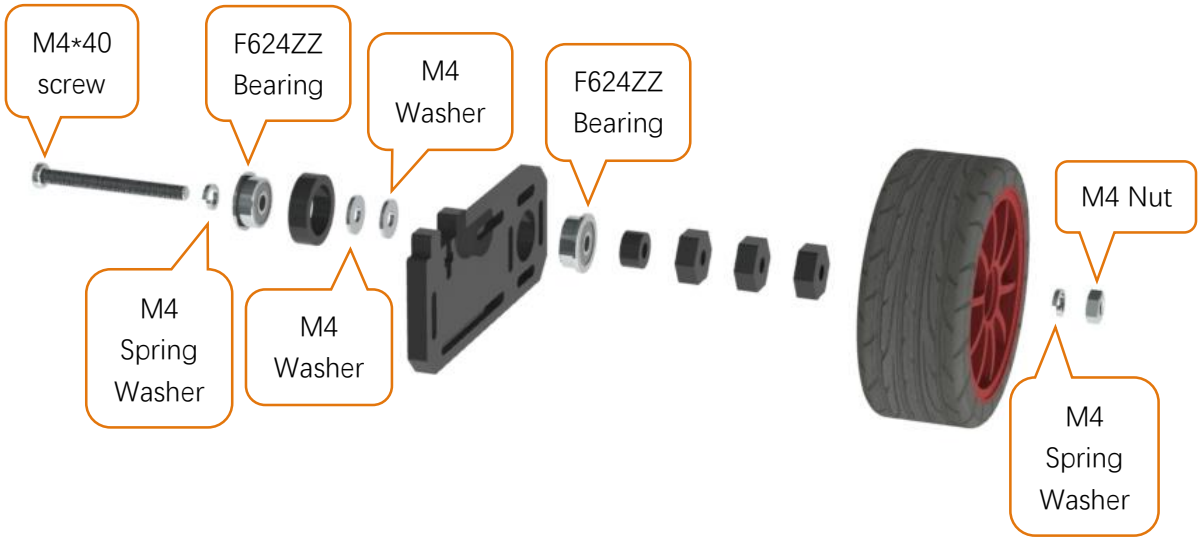


The two wheels can turn left and right



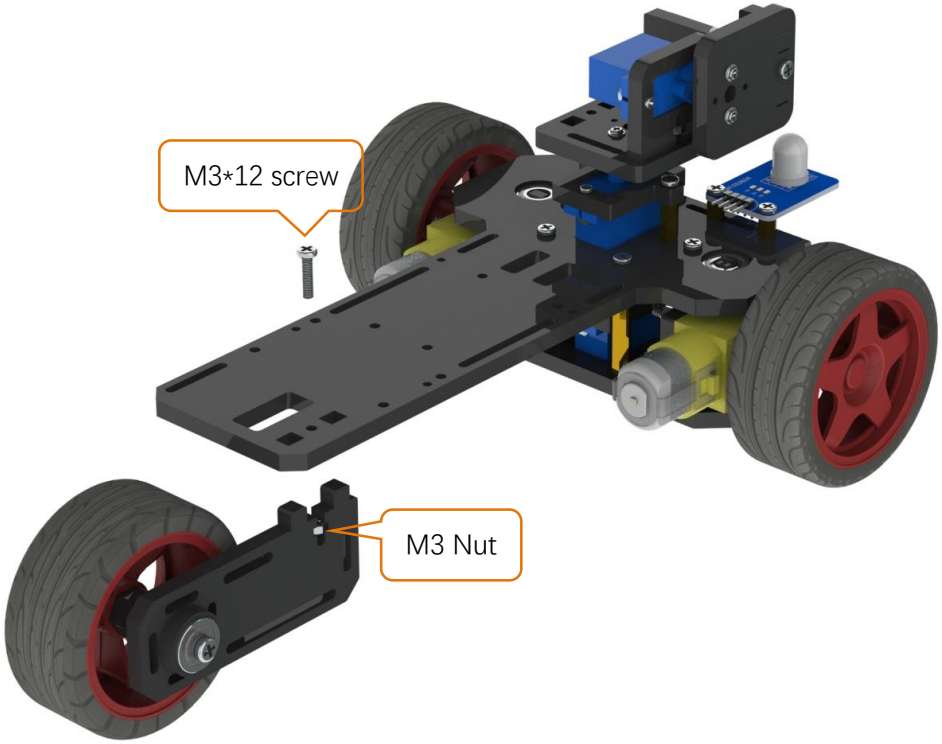
Assemble driven wheel

Assemble the following components

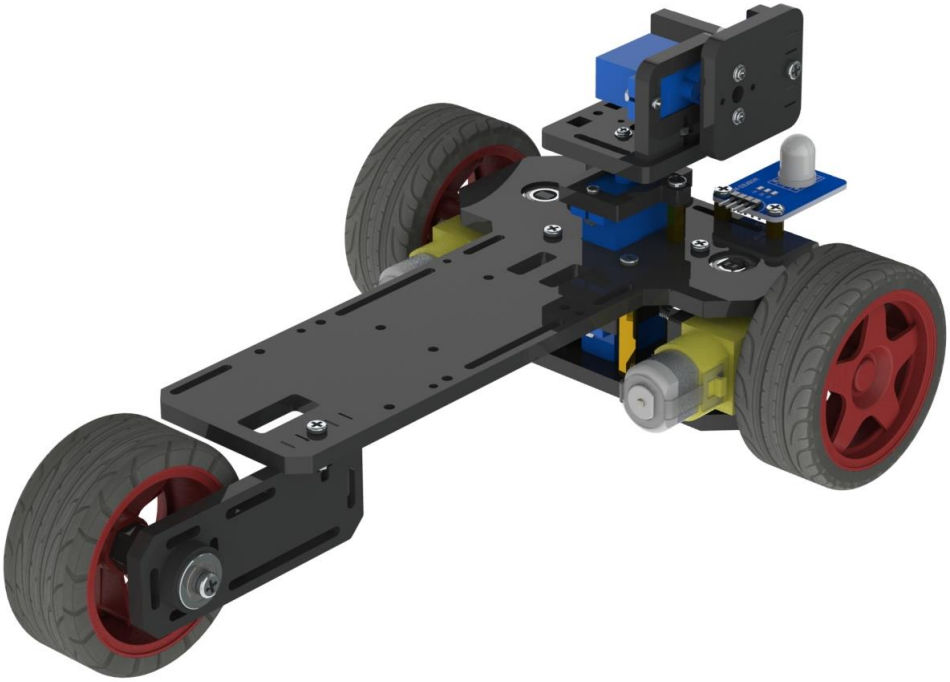


After assembling

Assemble the following components



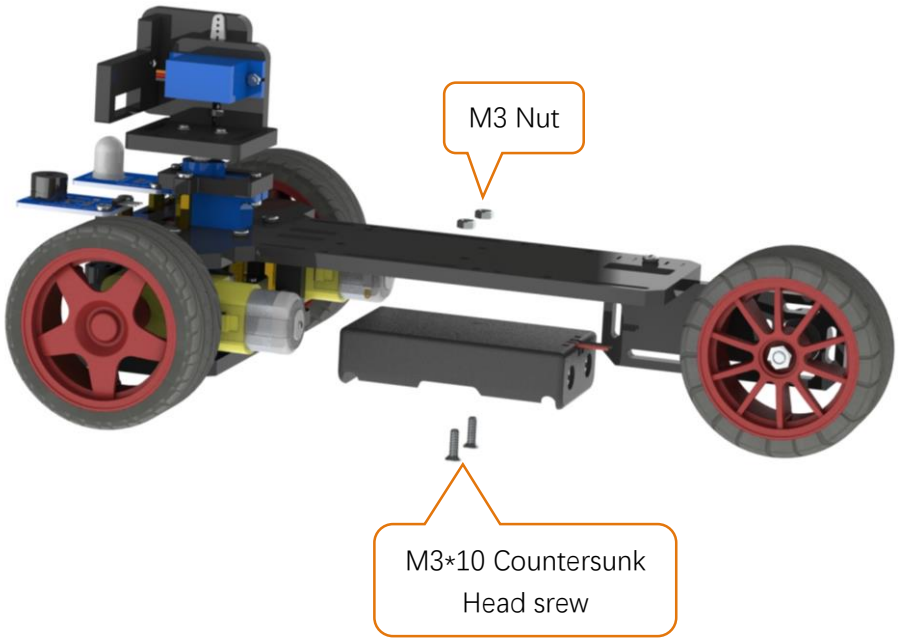
After assembling



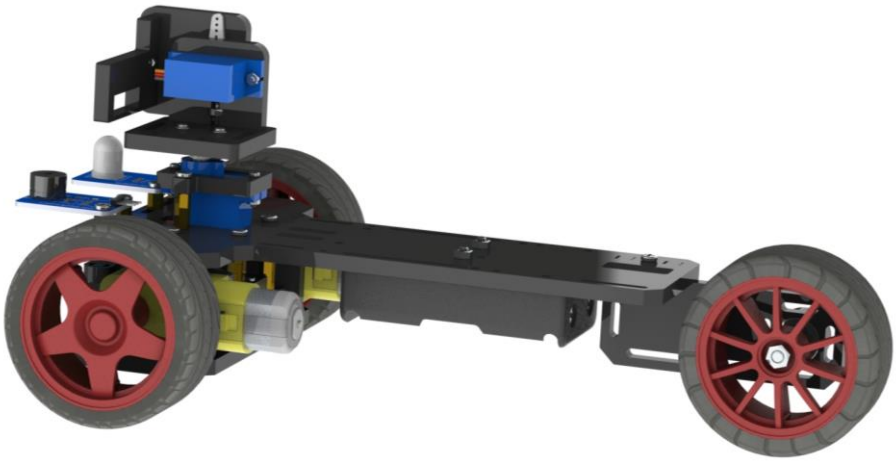
Electronic device

Assemble battery box

Assemble the following components

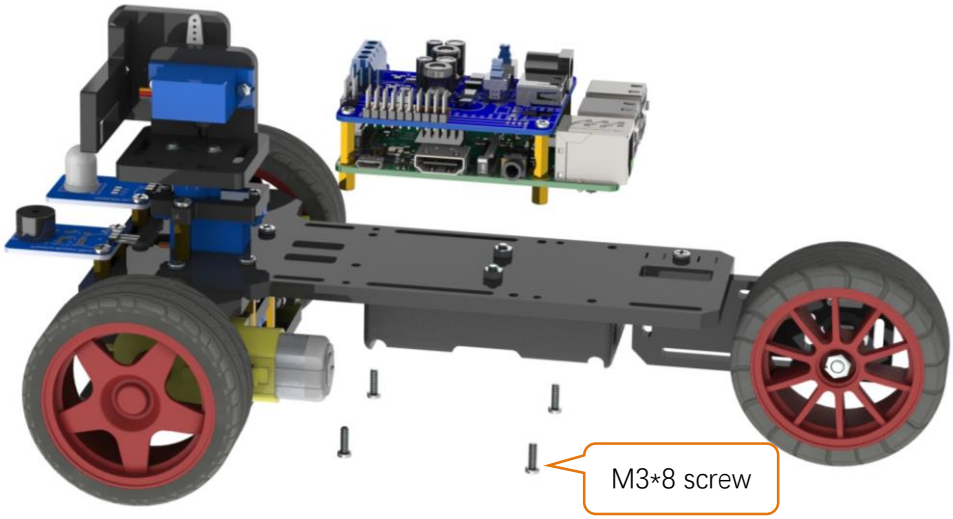


After assembling

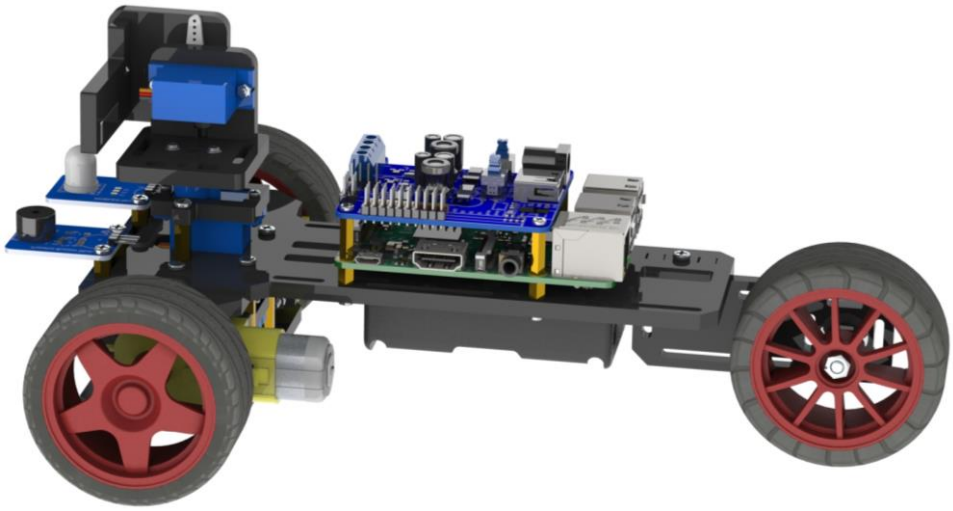


RPi and camera

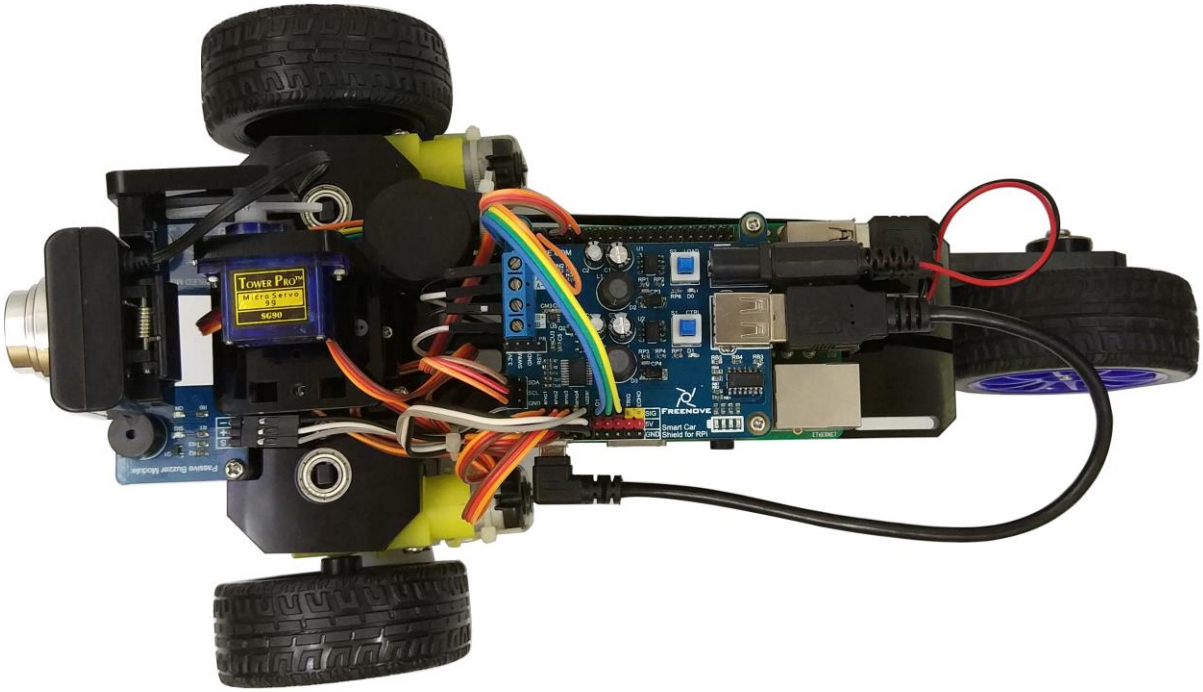
Assemble the following components

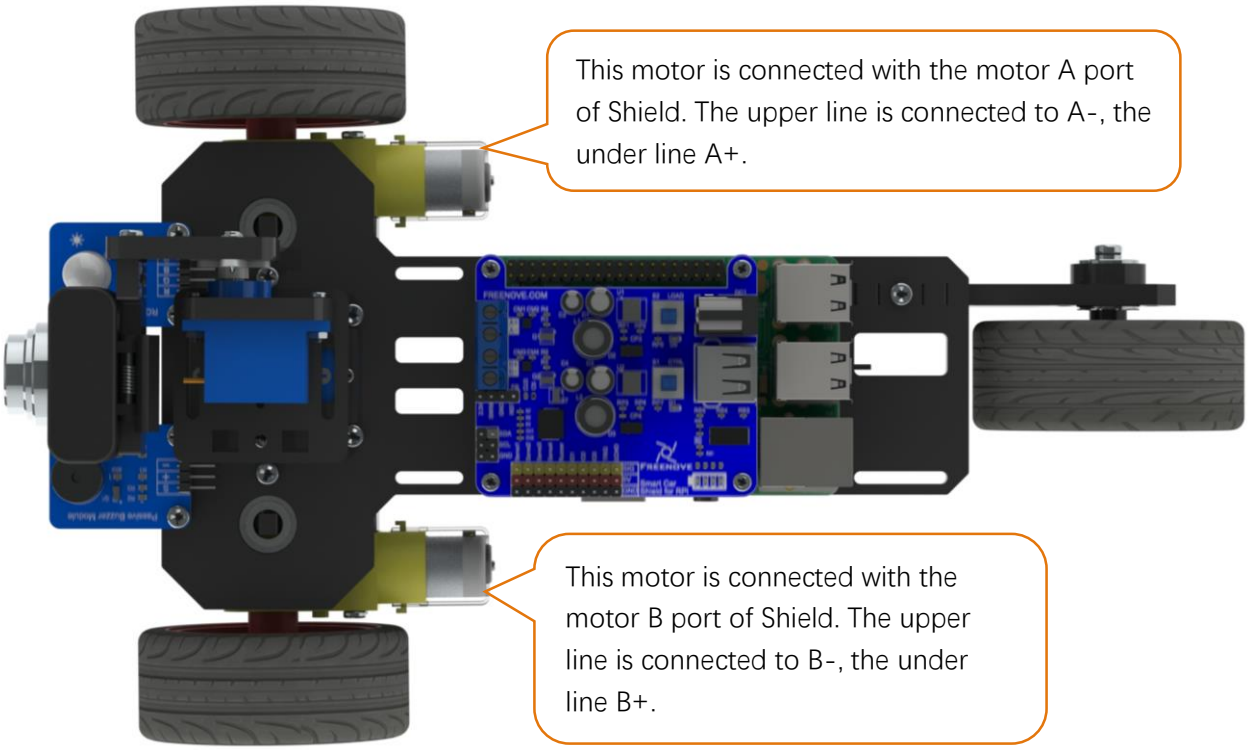


After assembly



Connection: Servo1, Servo2, Servo3 are connected to Servo1, Servo2, Servo3 port of the Shield respectively. RGBLED Module, Buzzer Module, and other loads are connected to the Shield in the same connection mode with front "test" section. The motor is connected as below, in which, if we find motor steering error, exchange connection of two lines to Shield. The camera is connected to any USB port on the RPi. Using the matching Micro USB Cable to connect Shield USB port with RPi power supply port. Assemble two 18650 batteries in the battery box, and connect the interface of battery box to DC power jack of the Shield.


TOWER PRO
SG90


This motor is connected with the motor A port of Shield. The upper line is connected to A-, the under line A+.
This motor is connected with the motor B port of Shield. The upper line is connected to B-, the under line B+.

Run the code

After the assembly is completed, run the code according to the following way. Then you can let the video car run up.

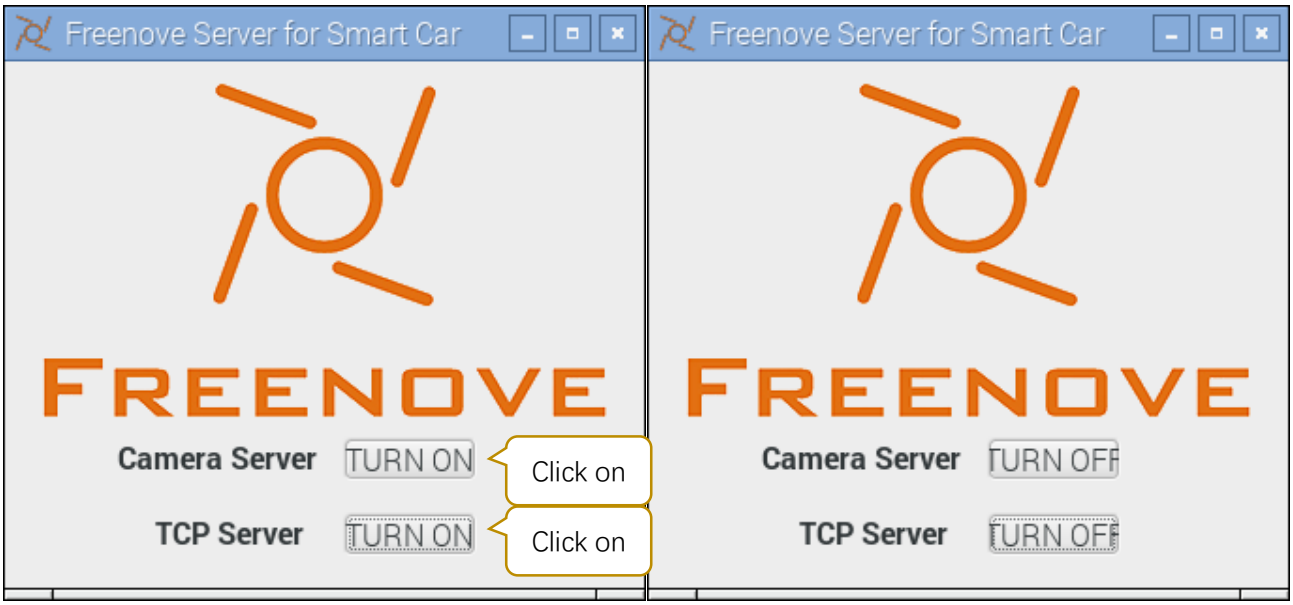
Open the server

Windowed server

Open the switch S1 and S2 on the Shield. After the RPi starts, use the remote desktop to connect RPi. Then open the terminal, and execute the following command to open the server.

```
cd ~/Freenove_Three_wheeled_Smart_Car_for_Raspberry_Pi/Server
python Main.py
```

Later, the following window interface appears. Click on the two buttons in the window to open the camera service and TCP communication service, respectively.



If the terminal shows information below, it indicates that the camera service and TCP communication service have been opened.

```
pi@raspberrypi:~/Freenove_Three-wheeled_Smart_Car_Kit_for_Raspberry_Pi/Server $
python Main.py
btn_CameraServer Clicked!
.............Camera server starting ......
MJPG Streamer Version: svn rev: Unversioned directory
 i: Using V4L2 device.: /dev/video0
 i: Desired Resolution: 320 x 240
 i: Frames Per Second.: 30
 i: Format............: YUV
 i: JPEG Quality......: 80
 o: www-folder-path...: ./www/
 o: HTTP TCP port.....: 8090
 o: username:password.: disabled
 o: commands..........: enabled
TCP Server Thread Starting ...
Wating for connect ...
```

When you want to close them, first click on two TURN OFF button to close the services, and then click on the close button on the top right corner of the window to terminate the program.

Server in command line mode

If you do not like the windowed server, you can open the camera and TCP communication services directly through the commands. Open the switch S1 and S2 on the Shield. After the RPi starts, use the remote desktop connect RPi. Then open the terminal and execute the following command.

```
cd ~/Freenove_Three_wheeled_Smart_Car_for_Raspberry_Pi/Server
python Main.py –mnt
```

or

```
python Main.py –m –t -n
```

Parameter "-m" is mjpg-streamer, which means to open the camera service. "-t" means to open the tcp service. "-n" means not to use the visual window interface.

Later, if the following contents appears, it indicates that the camera and tcp services have been opened.

```
pi@raspberrypi:~/Freenove_Three-wheeled_Smart_Car_Kit_for_Raspberry_Pi/Server $
python Main.py -mnt
.............Camera server starting ......
TCP Server Thread Starting ...
Wating for connect ...
MJPG Streamer Version: svn rev: Unversioned directory
 i: Using V4L2 device.: /dev/video0
 i: Desired Resolution: 320 x 240
 i: Frames Per Second.: 30
 i: Format............: YUV
 i: JPEG Quality......: 80
 o: www-folder-path...: ./www/
 o: HTTP TCP port.....: 8090
 o: username:password.: disabled
 o: commands..........: enabled
```

Press twice Ctrl-C or Ctrl-\ to terminate the program.

Open the client

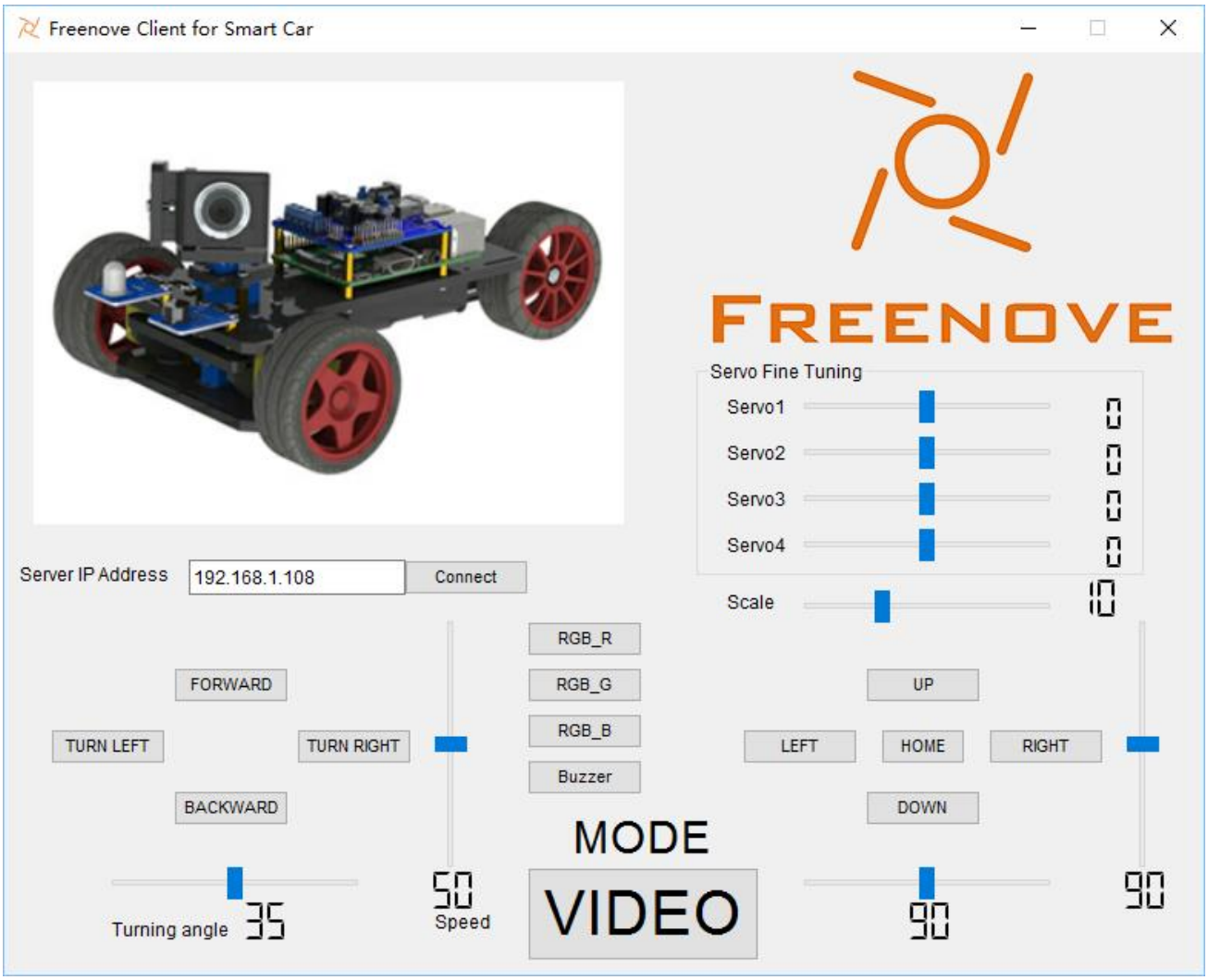
The client can run under any operating system in which Python and PyQt4 is installed. For example, Windows OS, Linux OS.

Client under Windows OS

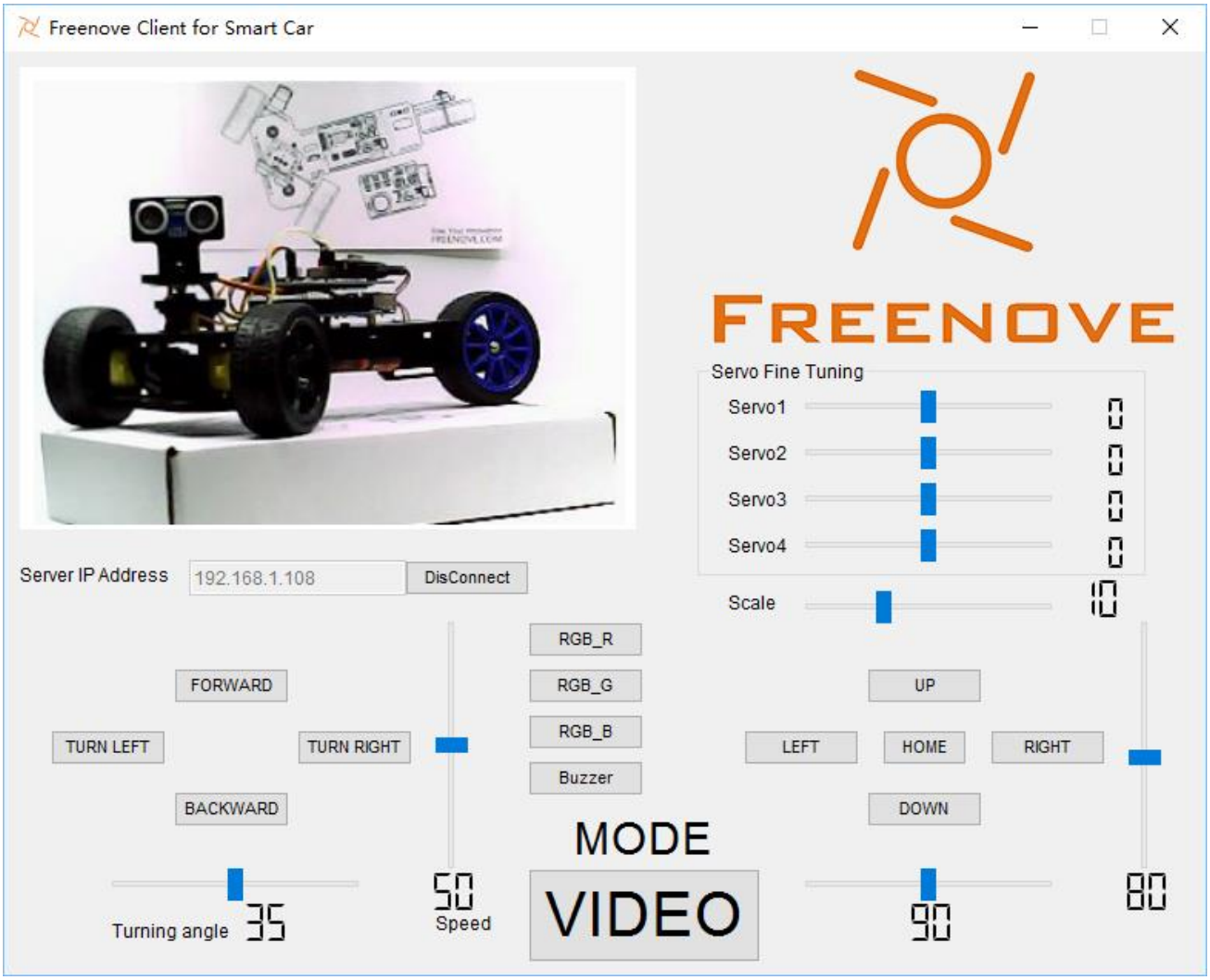
Press WIN+R, and type cmd to open the command line window. Then type the following command to open the client.

```
D:
cd \Freenove_Three_wheeled_Smart_Car_for_Raspberry_Pi\Client\
python Main.py
```

Or enter path "D:\Freenove_Three_wheeled_Smart_Car_for_Raspberry_P\Client\" and double-click Main.py with open way of Python2.7.exe. Then the following window interface appears.



In the edit box Server IP Address, input IP address of video car RPi and click Connect. Make sure that server of your RPi have been opened already, and the switch S1 and S2 on the Shield have also been opened.



After the connection is successful, you can control the video car.

Control mode

Besides using the mouse to click on the button in the window to control the video car, you can also use the highlight keys of keyboard to control the video car, as shown below.

The following is the corresponding action of Button/Key.

Button	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	H	Turn camera back Home
RGB_R	R	On/off red LED of RGBLED
RGB_G	G	On/off green LED of RGBLED
RGB_B	B	On/off blue LED of RGBLED
Buzzer	V	On/off Buzzer

The function of the SiliderBar with name in the window is shown below.

SliderBar	Function
Speed	Control the forward / backward speed of the car.
Turning angle	Control turning angle.
Scale	The minimum angle of each rotation of the camera.
Servo Fine Turning 1,2,3,4	Servo1,2,3,4 is for angle fine tuning settings. If the servo is not completely centered in installation, you can make a fine tuning by the SliderBar

Orther control:

Control	Function
Edit box Server IP Address	IP address of Server
Button “Connect/DisConnect”	Connect or DisConnect Server
Mode Button	Switch to Video/Radar Mode

Client under Windows OS

Similarly, using python to execute code Client.py and type the following command to open the client.

```
cd ~/Freenove_Video_Car_for_Raspberry_Pi/Client
python Main.py
```

Later, the same client window interface appears. and the control mode and operation mode are also the same with Windows OS.

Other operating systems

Under operating system in which Python and PyQt4 is stalled, you can run the client application through Python. Because their code is same, and with same control method.

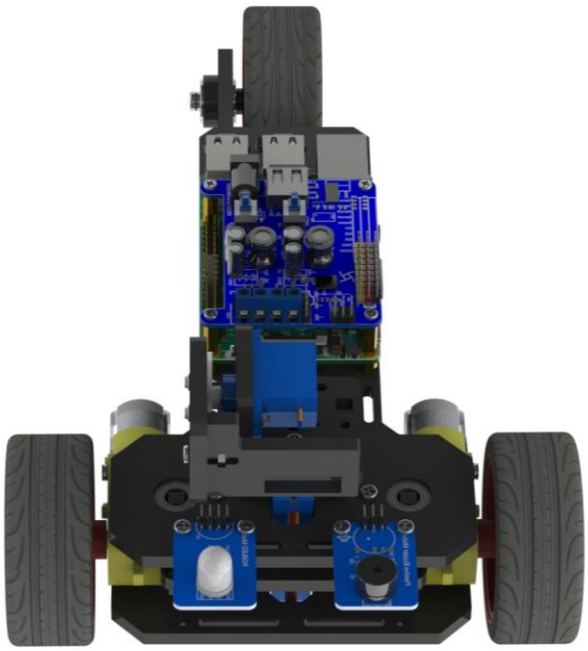
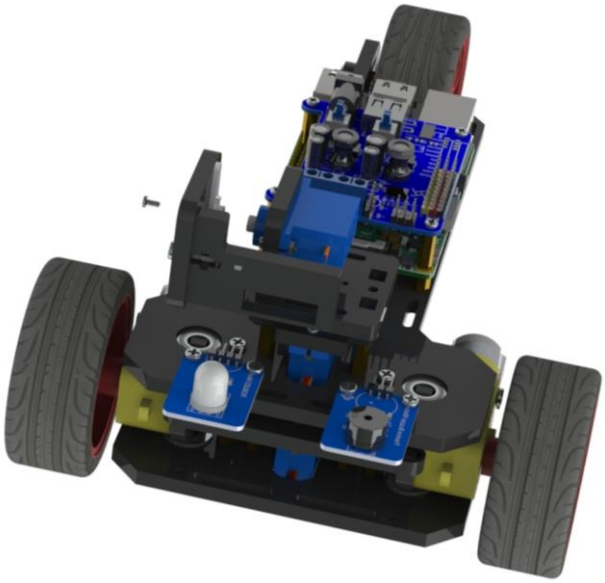
Chapter 2 Ultrasonic RadarCar

Assembly

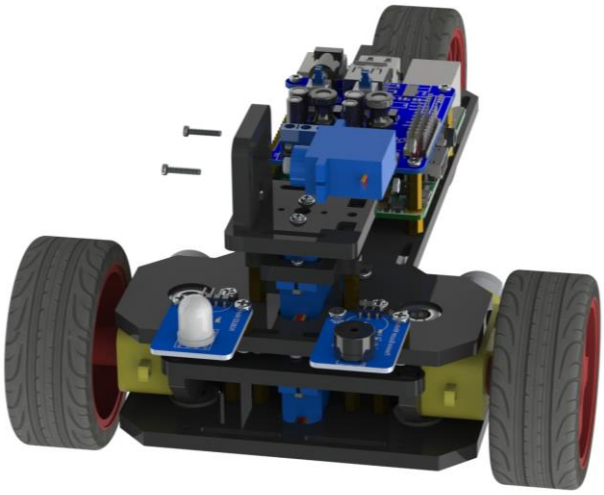
On the basis of the front video car, remove the camera, camera support, servo3, and servo3 support. Then assemble the support for Ultrasonic Module.

replace pan-tilt

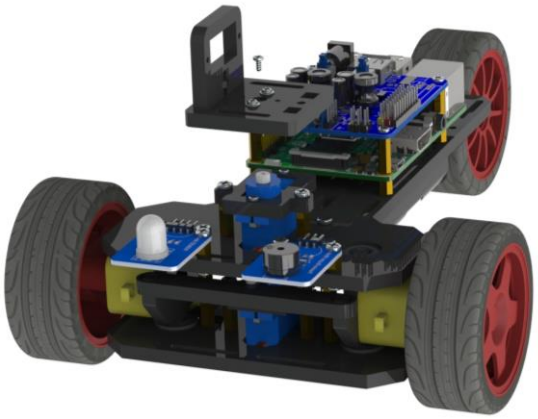
Remove camera part

Remove camera	Remove camera support
	

Remove servo3



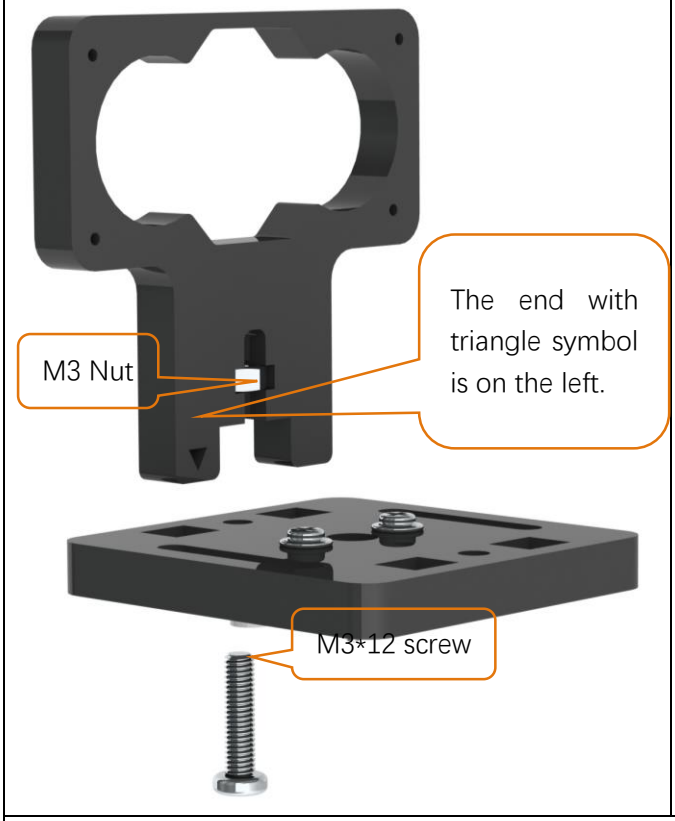
Remove servo3 support



Separate Servo3 support

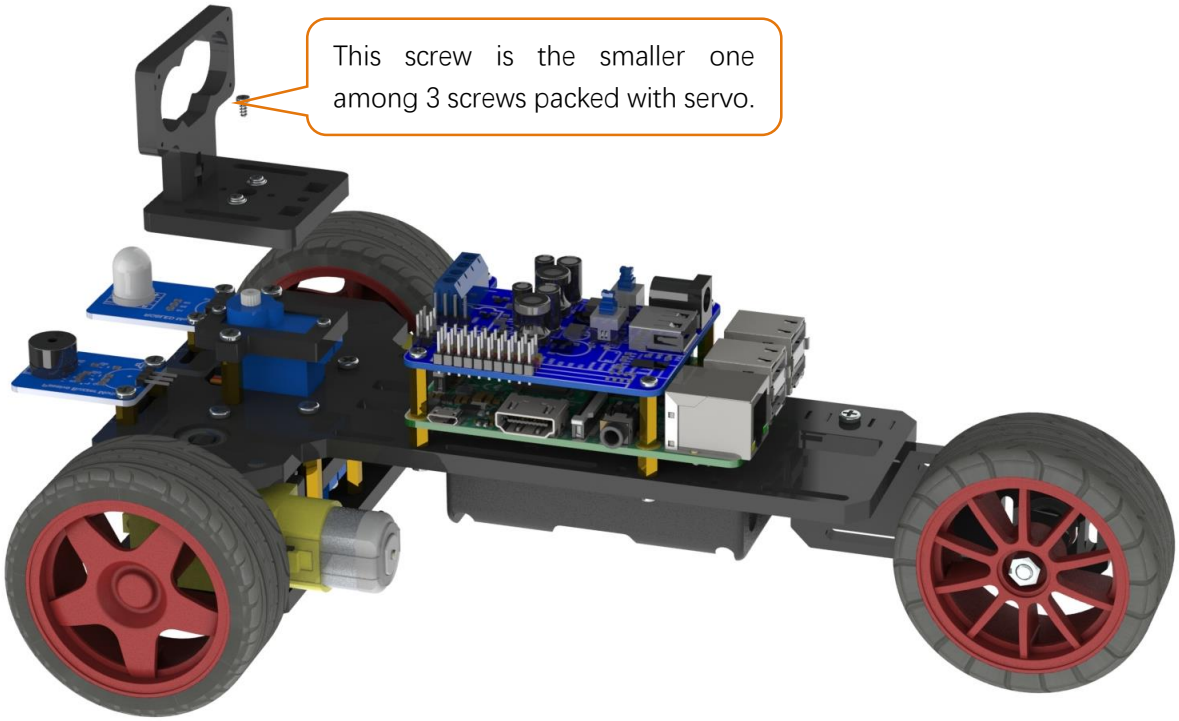
Assembly ultrasonic module support

Assemble the following components

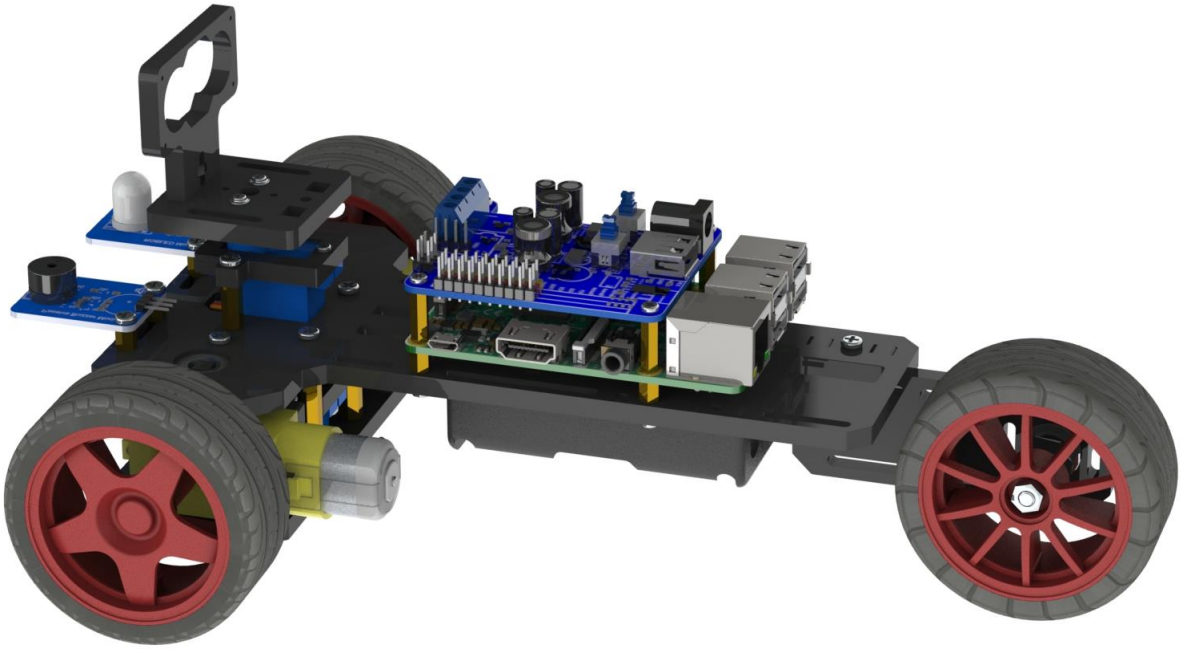


After assembling

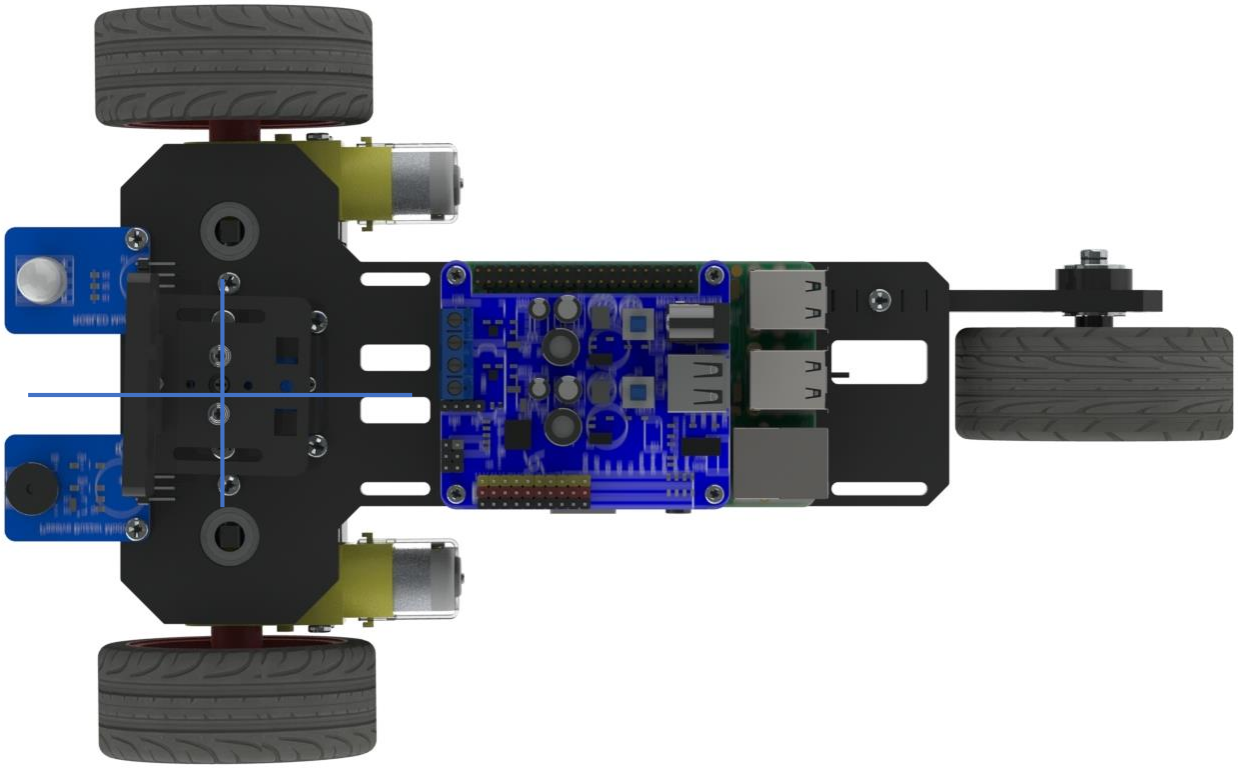
Assemble the following components



After assembling

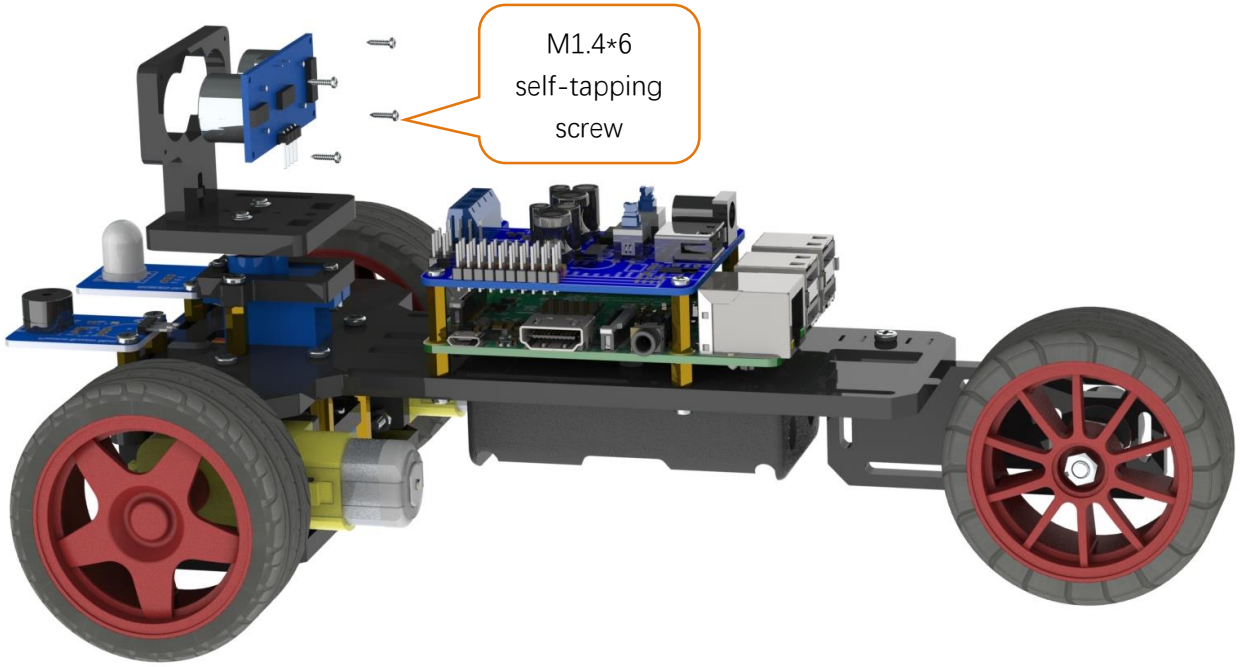


Keep the servo2 rotating to 90 degrees. If the drection is changed before installation, please make it rotate to the 90 degrees with the previous method. Complete the installation according to the direction below.

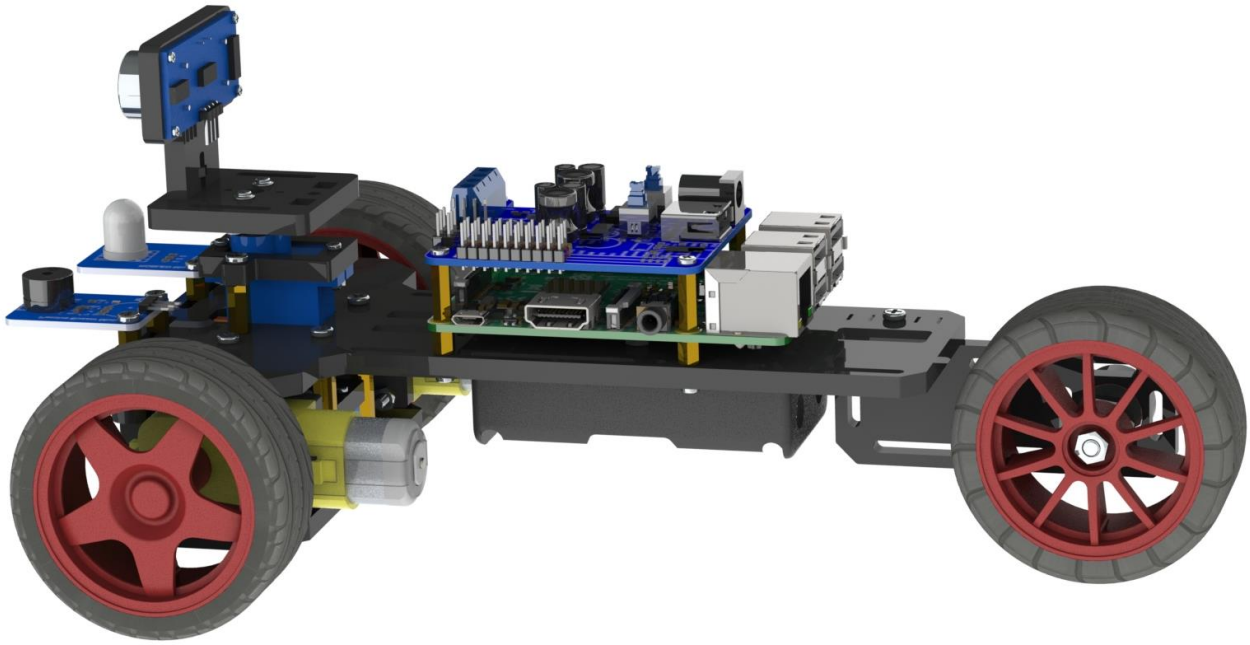


Ultrasonic Module

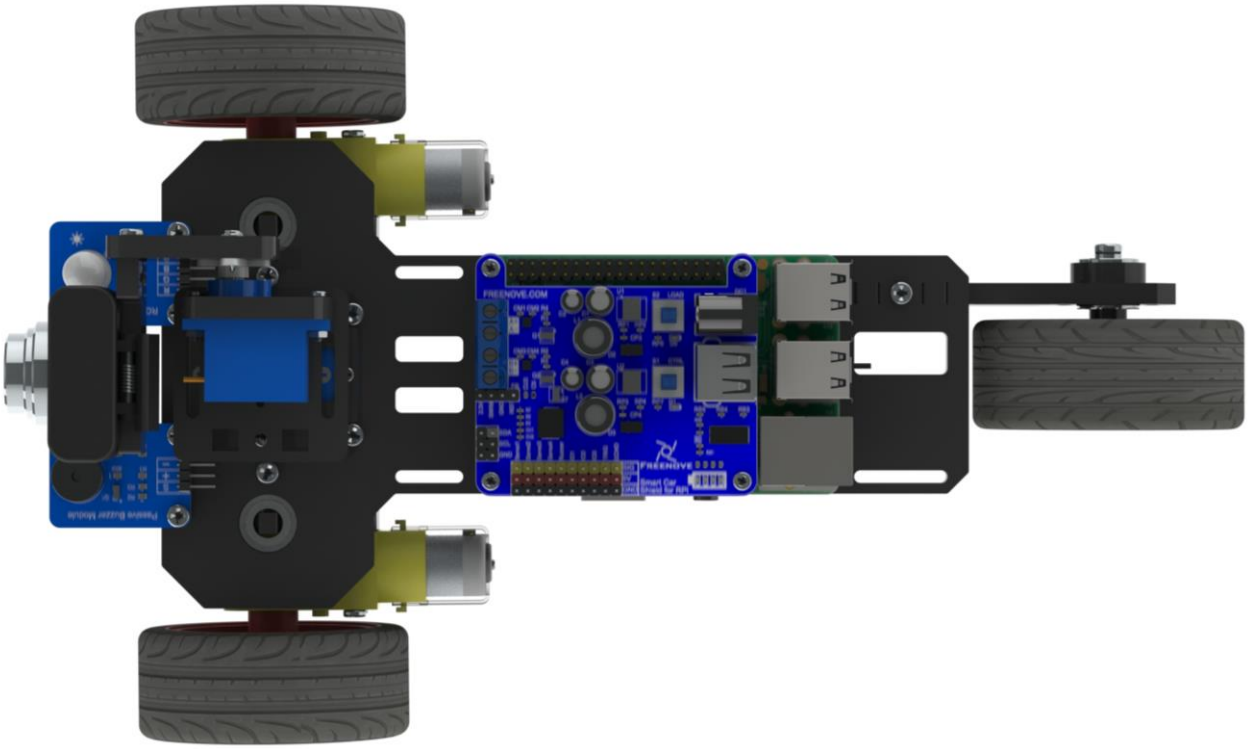
Use M-M Jumper to connect the Ultrasonic Module with the Shield. And then use TWO M1.4 * 6 tapping screws to assemble the Ultrasonic Module. As follows:



After assembling



Connection: connection mode between Shield and Ultrasonic Module, is the same as front "Test" section. Connection mode between other devices and Shield is the same with front “Smart video car”.



Run the Code

Open the server

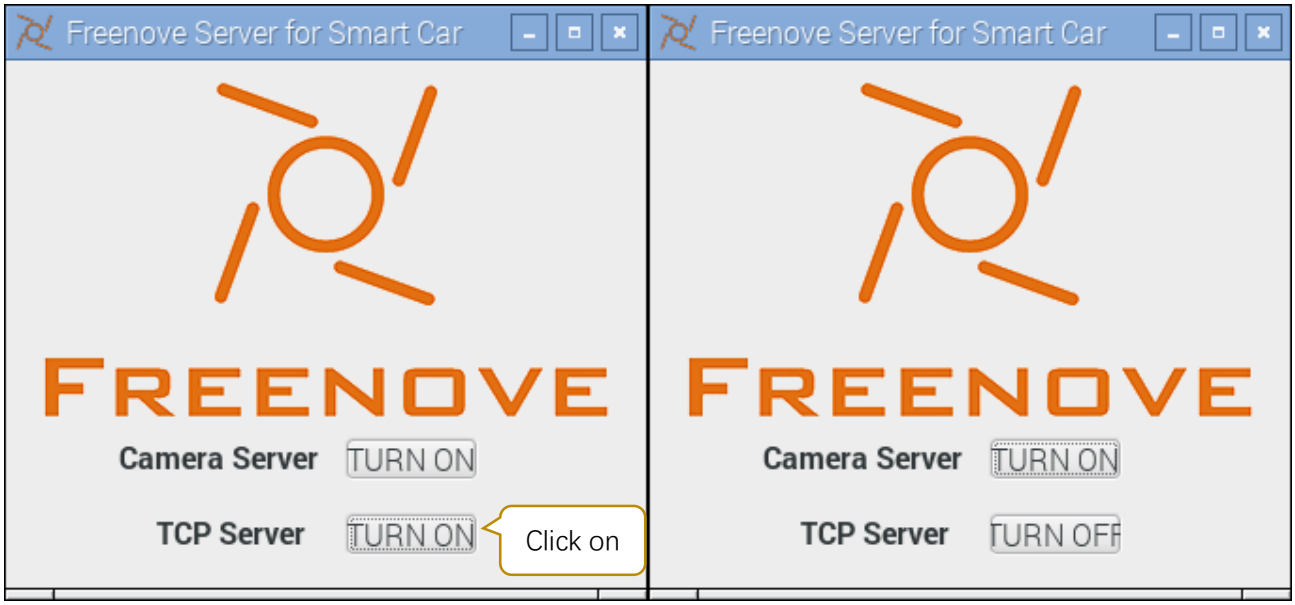
The method of opening the server is the same as the video car. Since there is no Camera device now, please do not open the Camera Server in the server window or use the "-m" command option.

Windowed server

Open the switch S1 and S2 on the Shield. After the RPi starts, use the remote desktop to connect RPi. Then open the terminal, and execute the following command to open the server.

```
cd ~/Freenove_Three_wheeled_Smart_Car_for_Raspberry_Pi/Server
python Main.py
```

Later, the following window interface appears. Click on the corresponding button of TCP Server to open TCP communication Server.



If the terminal shows information below, it indicates that the camera service and TCP communication service have been opened.

```
pi@raspberrypi:~/Freenove_Three-wheeled_Smart_Car_Kit_for_Raspberry_Pi/Server $
python Main.py
TCP Server Thread Starting ...
Wating for connect ...
```

When you want to close them, first click on two TURN OFF button to close the services, and then click on the close button on the top right corner of the window to terminate the program.

Server in command line mode

Type the following command to open the TCP service.

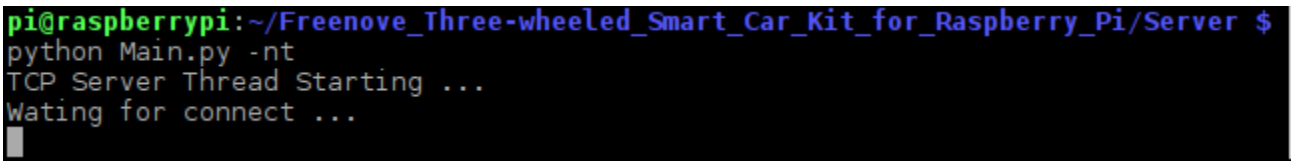
```
cd ~/Freenove_Three_wheeled_Smart_Car_for_Raspberry_Pi/Server
python Main.py –nt
```

or

```
python Main.py  –t -n
```

Parameter "-t" is means to open the tcp service. "-n" means not to use the visual window interface.

Later, if the following contents appears, it indicates that the tcp service has been opened.



Press Ctrl-C or Ctrl-\ to terminate the program.

Open the client

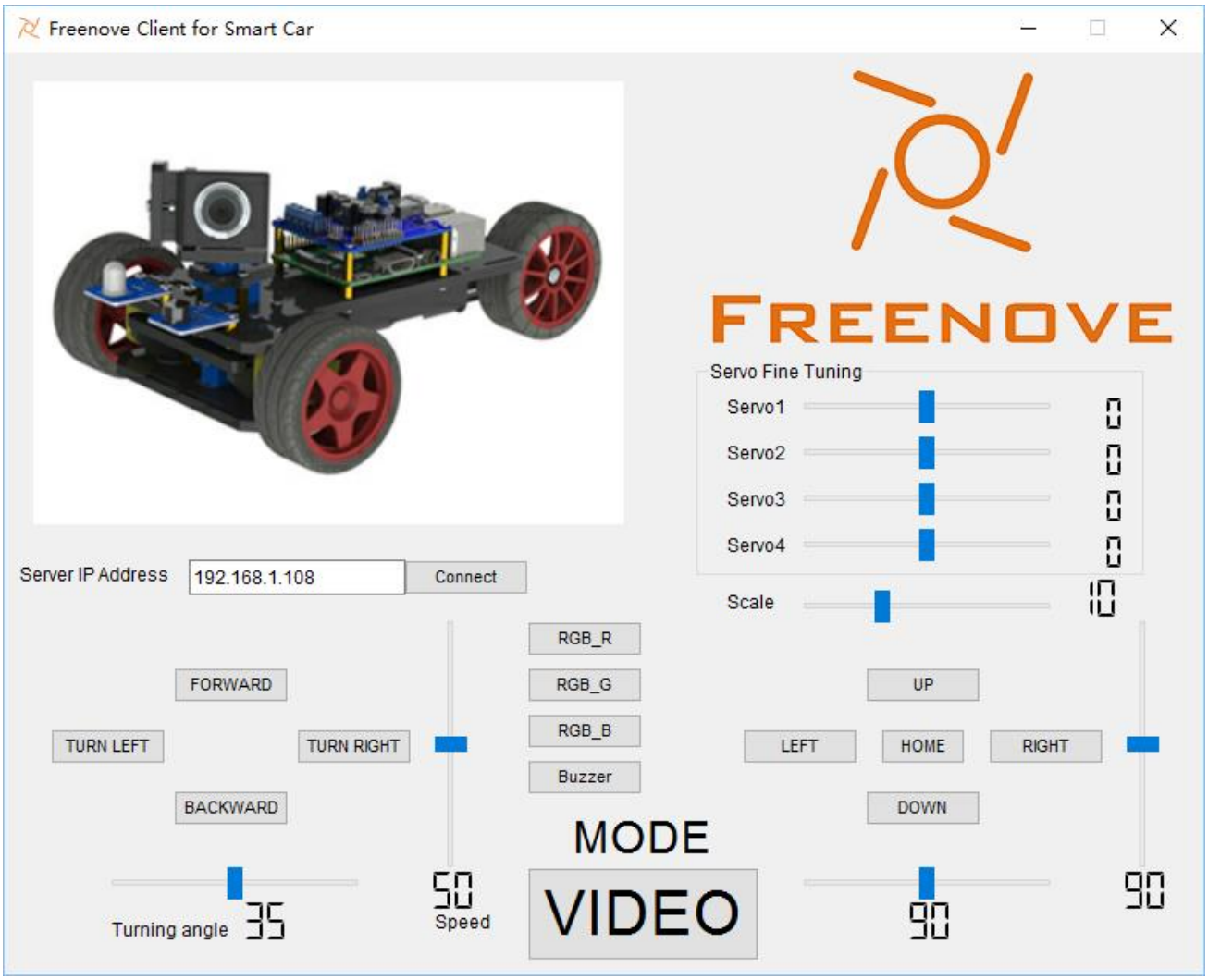
The way to open the client is the same as “Smart video car”

Client under Windows OS

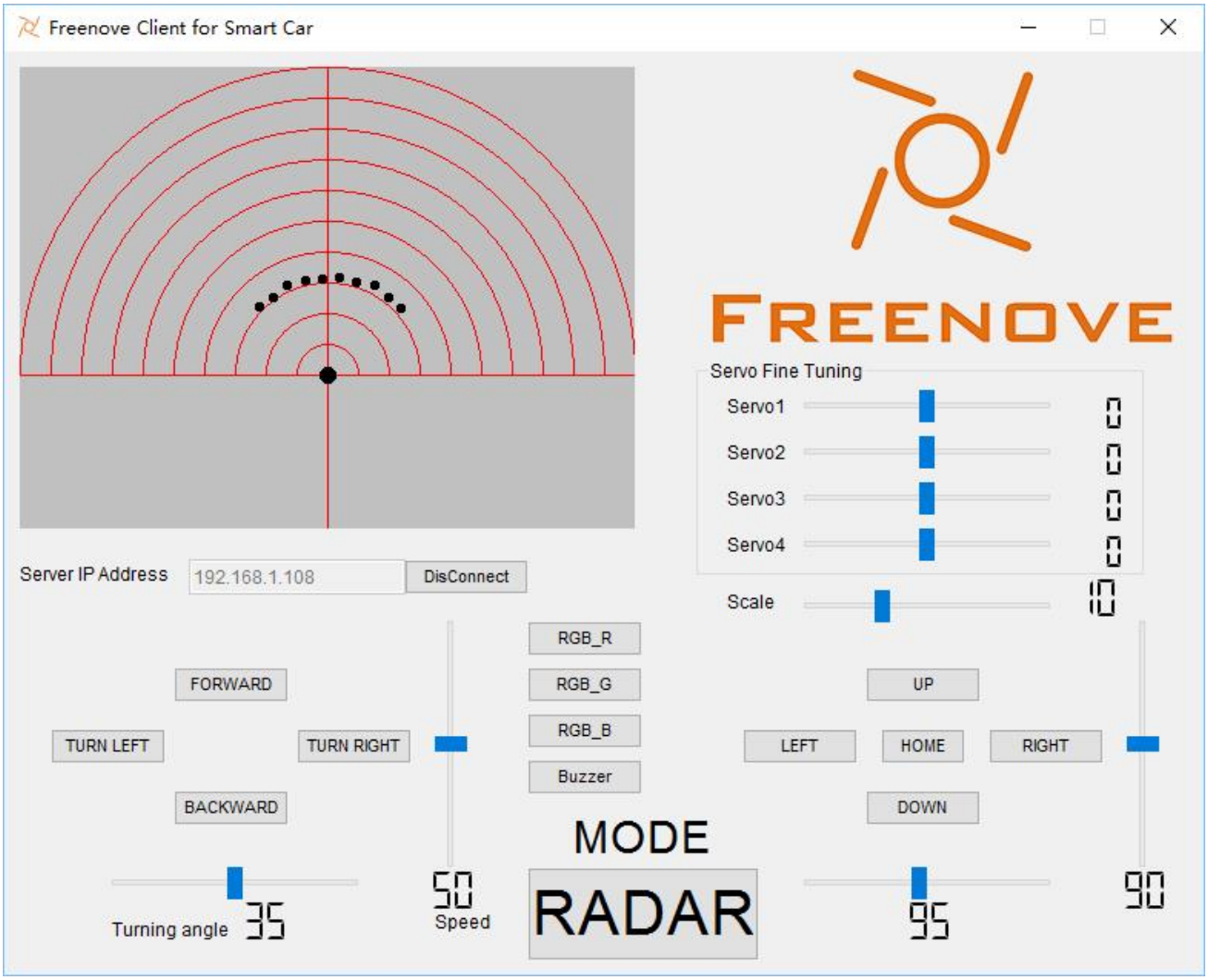
Press WIN+R, and type cmd to open the command line window. Then type the following command to open the client.

```
D:
cd \Freenove_Three_wheeled_Smart_Car_for_Raspberry_Pi\Client\
python Main.py
```

Or enter path “D:\Freenove_Three_wheeled_Smart_Car_for_Raspberry_P\Client\” and double-click Main.py with open way of Python2.7.exe. Then the following window interface appears.



In the edit box Server IP Address, input IP address of your RPi and click Connect. Make sure that server of your RPi have been opened already, and the switch S1 and Shield S2 have also been opened.
Then click on the MODE Button, the model will be switched to RADAR. Then the car enters ultrasonic radar mode, and constantly scan the distance, and the results will be displayed in the window. In this process, you can still control the car to move, and control the RGBLED module or Passive Buzzer module on the car.



Chapter 3 Part of the Code

In this chapter, we will explain the working principle and the key code of the whole project.

Server and Client

Server

The server is working on the RPi, and provides mjpg-streamer service and TCP service. And mjpg-streamer is for the transmission of camera data, TCP service for the transfer of the command.

mjpg-streamer

In the Camera_Server.py under "Server" folder, define a class Camera_Server, which inherits from the class threading.Thread. So when method .start () is called, Shell script file "Start_mjpg_Streamer.sh" is called, then the mjpg-streamer service is opened in a new thread. After mjpg-streamer starts, when the end signal (Ctrl-C, Ctrl-\, etc.) is received, it will exit, the thread will end.

Class Stop_Camera_Server is used to close the mjpg-streamer service. The class calls the Shell script file "Stop_mjpg_Streamer.sh" in a new thread, obtains the pid of the mjpg-streamer process, and then sends the end signal to end the service. And then the two threads will end.

Camera_Server.py

```
import threading
import os

class Camera_Server(threading.Thread):
    def camera_Http_Server(self):
        os.system("sudo /home/pi/Program/mjpg-streamer/Start_CMD_Suhayl.sh")

    def run(self):
        print ".............Camera server starting ......"
        self.camera_Http_Server()
        print ".............Camera server stop.........."
    def stop(self):
        os.system("sudo ./Stop_mjpg_Streamer.sh")

class Stop_Camera_Server(threading.Thread):
    def run(self):
        os.system("sudo ./Stop_mjpg_Streamer.sh")
```

About the command for opening “mjpg-streamer”, you can open the file “Start_mjpg_Streamer.sh” to view.

```
#!/bin/sh
curr_file=$0
cd "$(dirname "$curr_file")"
./mjpg_streamer -i "./input_uvc.so -y -d /dev/video0 -n -r 320*240 -f 30" -o
"./output_http.so -p 8090 -w ./www"
```

The meaning of the option parameters is as follows:

"-i", means the video input method, where the "input_uvc.so" is the input mode of USB Video Class (uvc).

The parameters can be specified as below:

```
 [-d | --device ].......: video device to open (your camera)
 [-r | --resolution ]...: the resolution of the video device,
                          can be one of the following strings:
                          QSIF QCIF CGA QVGA CIF VGA
                          SVGA XGA SXGA
                          or a custom value like the following
                          example: 640x480
 [-f | --fps ].........: frames per second
 [-y | --yuv ].........: enable YUYV format and disable MJPEG mode
 [-q | --quality ].....: JPEG compression quality in percent
                          (activates YUYV format, disables MJPEG)
 [-m | --minimum_size ].: drop frames smaller then this limit, useful
                          if the webcam produces small-sized garbage frames
                          may happen under low light conditions
 [-n | --no_dynctrl ]...: do not initalize dynctrls of Linux-UVC driver
 [-l | --led ].........: switch the LED "on", "off", let it "blink" or leave
                          it up to the driver using the value "auto"
```

"-o" means the video output method, where the "output_http.so" is used. The parameters can be specified as below:

```
 [-w | --www ].........: folder that contains webpages in
                          flat hierarchy (no subfolders)
 [-p | --port ].........: TCP port for this HTTP server
 [-c | --credentials ]...: ask for "username:password" on connect
 [-n | --nocommands ]....: disable execution of commands
```

The following is the file "Stop_mjpg_Streamer.sh", in which the pid of "mjpg_streamer" process is obtained, and then the kill command is used to end the process.

```
#!/bin/sh
pid="$(pgrep mjpg_streamer)"
echo "mjpg_streamer pid: "$pid
kill -9 $pid
```

TCP

In the mTCPServer.py under folder "Server", define a class mTCPServer, which is used to create a TCP monitor port, and, to forward the commands to Shield after receiving request from the client. Because the monitor port function is also blocked, it will be created in a new thread. You can open file mTCPServer.py to view the details.

mDEV

In the mDev.py under folder "Server", define class mDEV, which contains the control commands and methods for Shield. The following is the content of class mDEV.

```
import smbus
import time

def numMap(value,fromLow,fromHigh,toLow,toHigh):
    return (toHigh-toLow)*(value-fromLow) / (fromHigh-fromLow) + toLow

class mDEV:
    CMD_SERVO1      =   0
    CMD_SERVO2      =   1
    CMD_SERVO3      =   2
    CMD_SERVO4      =   3
    CMD_PWM1        =   4
    CMD_PWM2        =   5
    CMD_DIR1        =   6
    CMD_DIR2        =   7
    CMD_BUZZER      =   8
    CMD_IO1         =   9
    CMD_IO2         =   10
    CMD_IO3         =   11
    CMD_SONIC       =   12
    SERVO_MAX_PULSE_WIDTH = 2500
    SERVO_MIN_PULSE_WIDTH = 500
    Is_IO1_State_True = False
    Is_IO2_State_True = False
    Is_IO3_State_True = False
    Is_Buzzer_State_True = False
    def __init__(self,addr=0x18):
        self.address = addr #default address of mDEV
        self.bus=smbus.SMBus(1)
        self.bus.open(1)
    def i2cRead(self,reg):
        self.bus.read_byte_data(self.address,reg)

    def i2cWrite1(self,cmd,value):
        self.bus.write_byte_data(self.address,cmd,value)
```

```
    def i2cWrite2(self,value):
        self.bus.write_byte(self.address,value)

    def writeReg(self,cmd,value):
        try:
            self.bus.write_i2c_block_data(self.address,cmd,[value>>8,value&0xff])
        except Exception,e:
            print Exception,"I2C Error :",e

    def readReg(self,cmd):
        [a,b]=self.bus.read_i2c_block_data(self.address,cmd,2)
        return a<<8 | b

    def getSonicEchoTime():
        SonicEchoTime = mdev.readReg(mdev.CMD_SONIC)
        return SonicEchoTime

    def getSonic(self):
        SonicEchoTime = mdev.readReg(mdev.CMD_SONIC)
        distance = SonicEchoTime * 17.0 / 1000.0
        return distance
    def setShieldI2cAddress(self,addr): #addr: 7bit I2C Device Address
        if (addr<0x03) or (addr > 0x77) :
            return
        else :
            mdev.writeReg(0xaa,(0xbb<<8)|(addr<<1))
```

The following table is corresponding Shield interface of the command. You can operate them through the member function writeReg and readReg.

Command	Interface	Read/Write	Valid value
CMD_SERVO1	Servo1	W	0-20000
CMD_SERVO2	Servo2	W	0-20000
CMD_SERVO3	Servo3	W	0-20000
CMD_SERVO4	Servo4	W	0-20000
CMD_PWM1	Motor1 Speed	W	0-1000
CMD_PWM2	Motor2 Speed	W	0-1000
CMD_DIR1	Motor1 Steering	W	0 or non-0
CMD_DIR2	Motor2 Steering	W	0 or non-0
CMD_BUZZER	Buzzer	W	0-65535
CMD_IO1	IO1	W	0 or non-0
CMD_IO2	IO2	W	0 or non-0
CMD_IO3	IO3	W	0 or non-0
CMD_SONIC	TRIG/ECHO	R	

Default I2C address of the Shield is 0x18 (7bit). Member function setShieldI2cAddress (self, addr) is used to modify the I2C address of Shield. Don't change its address unless it conflicts with the I2C address of your other device. Because detection range of the I2CTool is 0x03-0x77, so the range of parameter "addr" is limited to 0x03-0x77. It is invalid when beyond this range. When the I2C address is modified, the Shield need to be restarted. when there is a need, change the parameter addr of the class constructor to the new I2C address to facilitate use later. Please use this function with caution.

```
    def __init__(self,addr=0x18):
    ......
    def setShieldI2cAddress(self,addr): #addr: 7bit I2C Device Address
        if (addr<0x03) or (addr > 0x77) :
            return
        else :
            mdev.writeReg(0xaa, (0xbb<<8)|(addr<<1))
```

Client

Connect the Client to the server through the TCP port. Then define class TCPClient in the TCPClient.py under "Client" folder, which contains the method for connecting to server and sending data.

```
from socket import *

class TCPClient:
    #HOST = '127.0.0.1'
    HOST = '192.168.1.108'
    PORT = 12345
    BUFSIZ = 1024
    ADDR = (HOST, PORT)

    def __init__(self):
        #self.client = socket(AF_INET, SOCK_STREAM)
        pass

    def connectToServer(self, address = ADDR):
        self.client = socket(AF_INET, SOCK_STREAM)
        self.client.connect(address)

    def disConnect(self):
        try:
            self.client.close()
        except Exception, e:
            print Exception, "Disconnect error:", e

    def sendData(self, data):
```

```
        try:
            self.client.send(data)
        except Exception, e:
            print Exception, "Send TCP Data error:", e

    def recvData(self):
        try:
            return self.client.recv(self.BUFSIZ)
        except Exception, e:
            print Exception, "Recv TCP Data error:", e
```

In “main.py”, monitor the action of the control and keyboard, and send a command to the server.
View “main.py” for detailed information.
In “Config.txt”, preserve the IP address of successful connection to the server last time, and load it the next time start the client. This enables you to connect to the server quickly without entering the IP address of the server in the edit box each time.

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 Freenovee lectronic 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.

Thank you again for choosing Freenove products.