Preface

About SunFounder

SunFounder is a technology company focused on Raspberry Pi and Arduino open source community development. Committed to the promotion of open source culture, we strive to bring the fun of electronics making to people all around the world and enable everyone to be a maker. Our products include learning kits, development boards, robots, sensor modules and development tools. In addition to high quality products, SunFounder also offers video tutorials to help you make your own project. If you have interest in open source or making something cool, welcome to join us!

About This Kit

The kit is suitable for the Raspberry Pi model B+, 2 model B and 3 model B.

In this book, we will show you how to build the smart car via description, illustrations of physical components, and schematic diagrams of circuits, in both hardware and software respects. You may visit our website www.sunfounder.com to download the related code and view the user manual on **LEARN -> Get Tutorials** and watch related videos under **VIDEO**, or clone the code on our page of github.com at
https://github.com/sunfounder/Sunfounder_Smart_Video_Car_Kit_for_RaspberryPi

Free Support

If you have any **TECHNICAL questions**, add a topic under **FORUM** section on our website and we'll reply as soon as possible.

For **NON-TECH questions** like order and shipment issues, please **send an email to service@sunfounder.com**. You're also welcomed to share your projects on FORUM.

Reprint 4.0

Contents

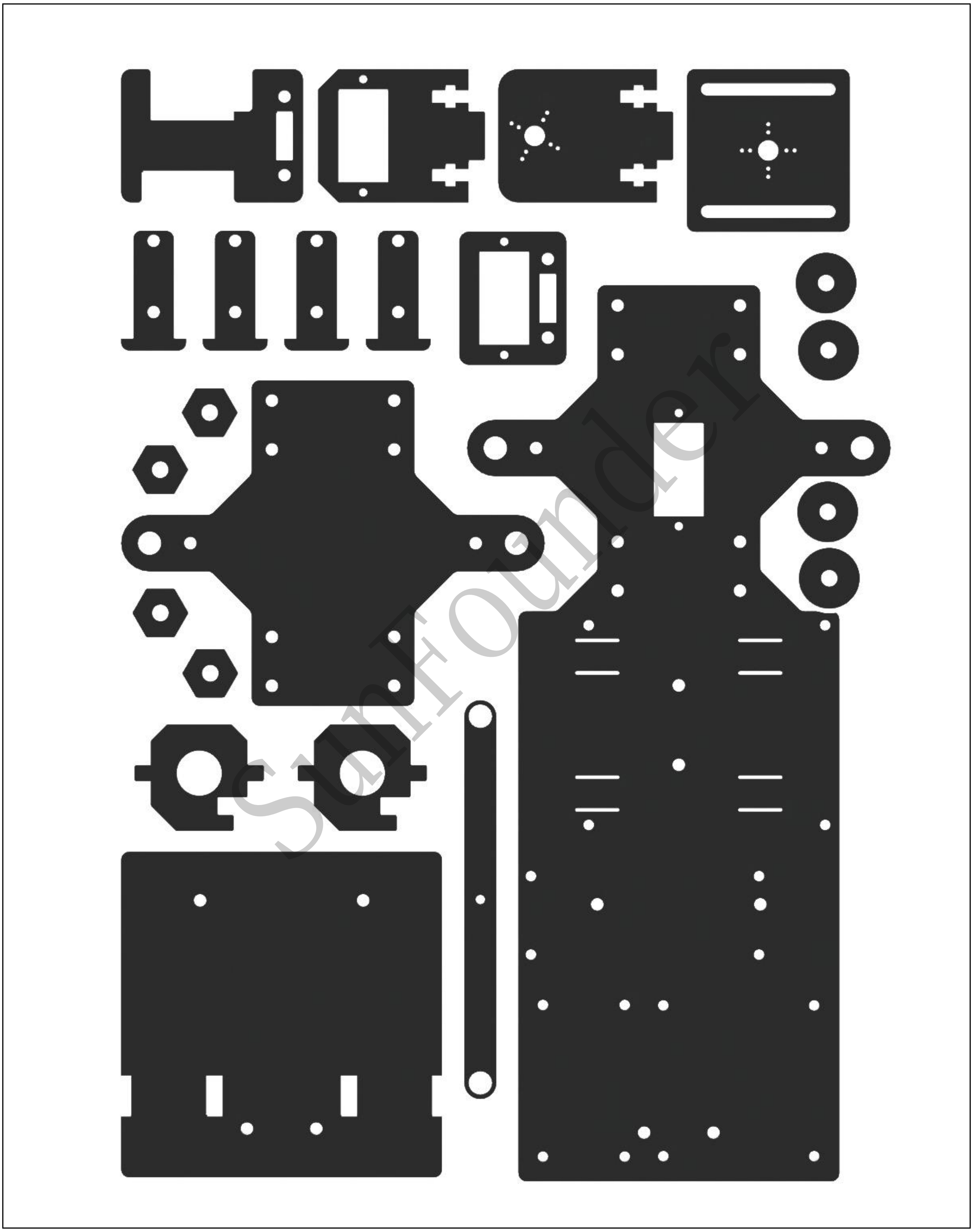
Introduction

The SunFounder Smart Video Car Kit for Raspberry Pi is composed of Raspberry Pi, step-down DC-DC converter module, USB camera, DC motor driver, and PCA9685-based servo controller. From the perspective of software, the smart car is of client/server (C/S) structure. The TCP server program is run on Raspberry Pi for direct control of the car. And the video data are acquired and delivered via the open source software MJPG-streamer in a real-time manner. The TCP client program is run on PC to send the control command. Both the client and server programs are developed in Python.

The smart car is developed based on the open source hardware Raspberry Pi and integrates the knowledge of mechanics, electronics and computer, thus having profound educational significance.

Components

A. Acrylic Plates



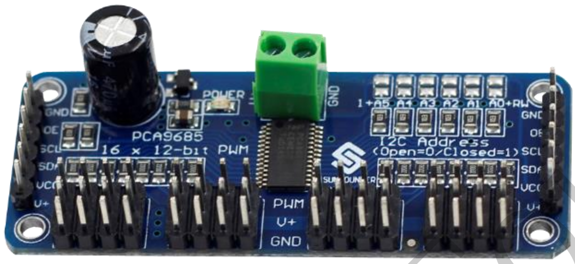
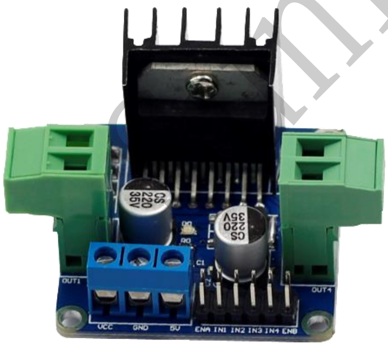
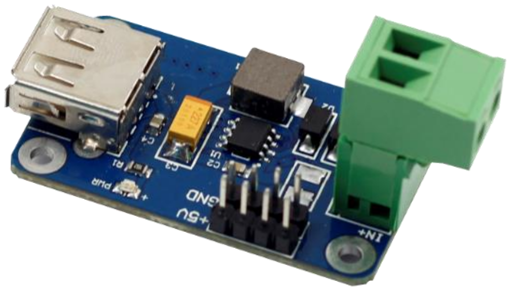
B. Mechanical Fasteners

Parts	Name	Qty.
	M1.2*4 self-tapping screw	8
	M2*8 screw	6
	M2.5*6 screw	16
	M3*10 countersunk screw	2
	M3*8 screw	8
	M3*10 screw	6
	M3*30 screw	4
	M4*25 screw	2
	M2.5*8 copper pillar	16
	M3*24 copper pillar	8
	M2 nut	6
	M2.5 nut	16
	M3 nut	20
	M4 self-locking nut	2
	F694ZZ flange bearing	2

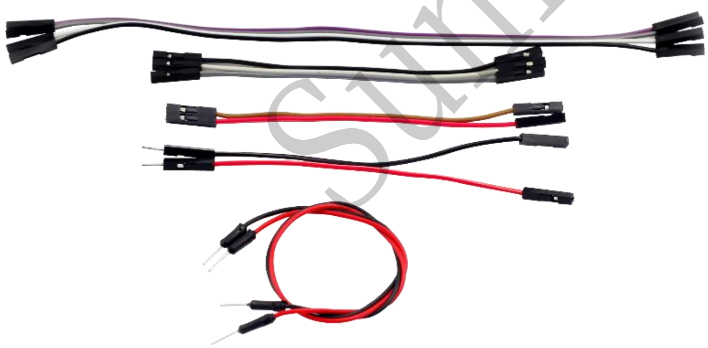
C. Drive Parts

Parts	Name	Qty.
	Tower Pro Micro Servo SG90	3
	Gear reducer	2
	Driven wheel	2
	Active wheel	2

D. Electrical Components

Parts	Name	Qty.
	Raspberry Pi Model B+	1
	16-Channel 12-bit PWM driver (servo controller)	1
	L298N DC motor driver	1
	Step-down DC-DC converter module	1

	USB Wi-Fi adapter	1
	USB camera	1
	18650*2 Battery holder	1
	Ribbon	1
	USB cable	1

	Cross socket wrench	1
	Cross screwdriver	1
	Cable Spiral Wrap	1
	20cm jumper wire (F to F)	4
	10cm jumper wire (F to F)	5
	10cm jumper wire (M to F)	2
	20cm jumper wire (M to M)	2

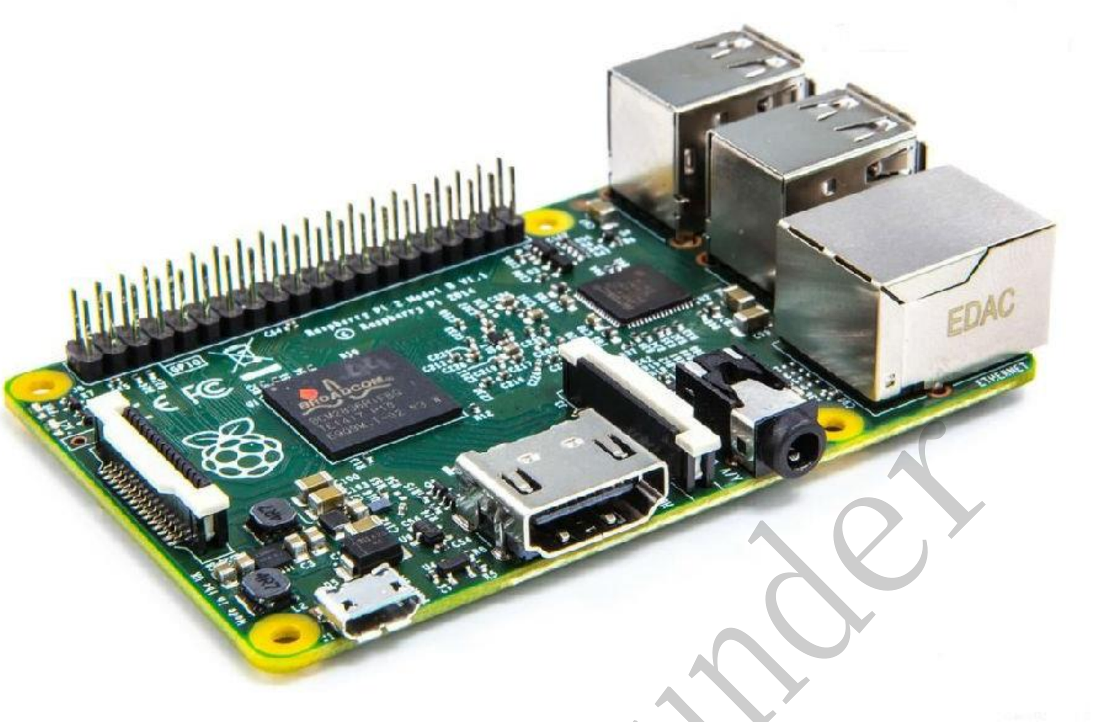
E. Self-provided Parts

The following parts are not included in the set.

Parts	Name	Qty. Needed
	18650 3.7V rechargeable Li-ion battery	2
	TF Card	1

Electrical Components Basics

A. Raspberry Pi

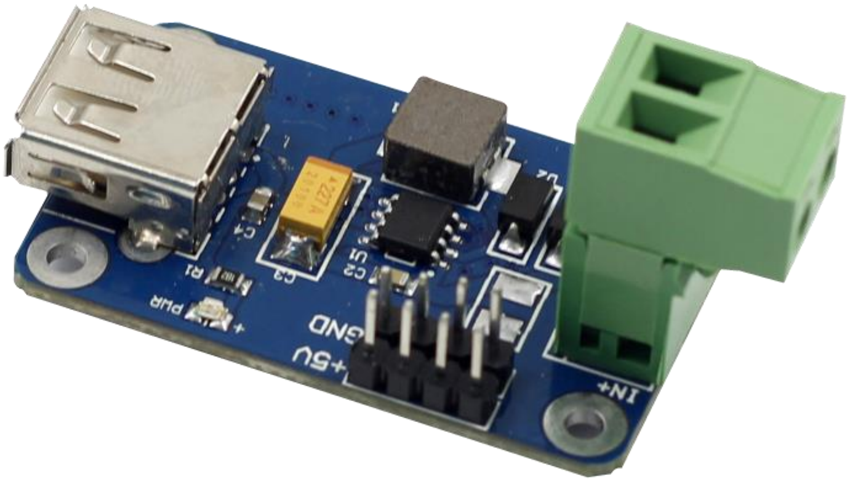


The Raspberry Pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is a capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python. It's capable of doing everything you'd expect a desktop computer to do, from browsing the internet and playing high-definition video, to making spreadsheets, doing word-processing, and playing games.

What's more, the Raspberry Pi has the ability to interact with the outside world, and has been used in a wide array of digital maker projects, from music machines and parent detectors to weather stations and tweeting birdhouses with infra-red cameras.

In this kit, we use Raspberry Pi as the core controller of driving equipment like DC motor and servo. With the device and a camera collaborated, video data can be acquired in a real-time manner and delivered by Wi-Fi network.

B. Step-down DC-DC Converter Module



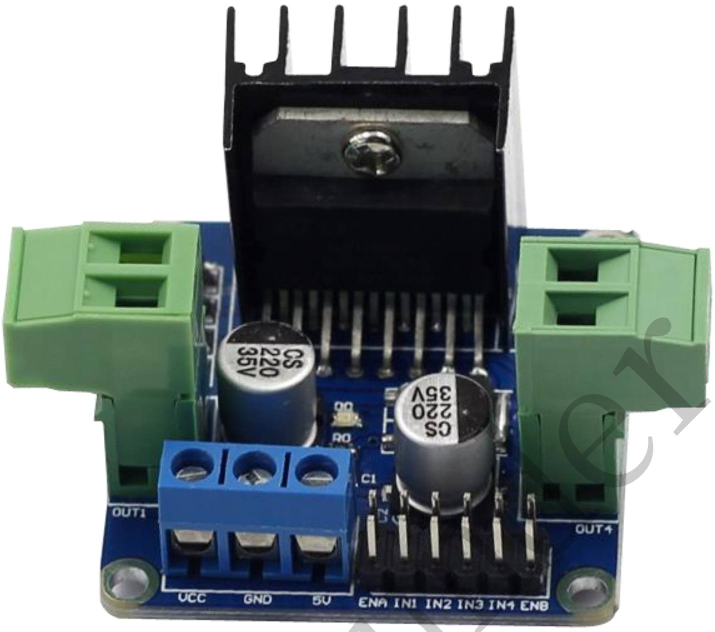
Built based on the chip XL1509, the module converts the battery output of 7.4V to 5V, so as to supply power to Raspberry Pi and the servo. As a DC to DC converter IC, the chip has an input voltage ranging from 4.5V to 40V and generates an output voltage of 5V with a current of as high as 2A. Please note: only when the input voltage is up to 6.5V, a 5V output can be supported.

C. Servo

In this smart car, one servo controls the direction of the car, and the other two, the movement of the camera between X axis and Y axis, thus defining the coverage of the camera. A servo is an automatic control system composed of DC motor, reduction gear set, sensor, and control circuit. It defines the rotation angle of the output shaft via delivering specific PWM signals.

Generally, a servo supports a maximum rotation angle of the shaft (like 180 degrees). It differs from a common DC motor in the rotation mode: a DC motor rotates by circle when a servo rotates in a certain degree and does not rotate in a round circle. Also, the former is used for power supply by its whole-circle rotation, while the latter is applied to controlling the rotation angle of an object (like joints of a robot).

D. DC Motor Driver

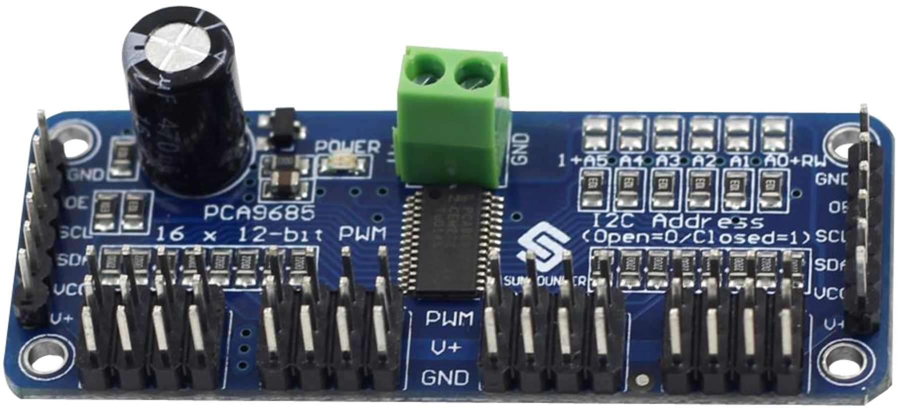


As the name suggests, the module is used to drive DC motors. The driver is built based on L298N. As a high-voltage and large-current chip for motor driving, encapsulated with 15 pins, the chip has a maximum operating voltage of 46V and an instant peak current of as high as 3A, with an operating current of 2A and rated power of 25W. Thus, it is completely capable of driving two low-power DC motors.

E. USB Wi-Fi Adaptor

The adapter helps Raspberry Pi connect to a Wi-Fi hot spot.

F. Servo Controller



The Servo Controller is built based on PCA9685. PCA9685 is a 16-channel LED controller with I2C bus interface. The resolution ratio of each channel is 12 bits (2^{12}=4096 levels). The controller works in a frequency between 40Hz and 1000Hz and its duty cycle can be adjusted in a range of 0 to 100%. It provides PWM signals for the servo and controls the rotation angle of the servo. Meanwhile, the module controls the duty cycle of the square waves output from channel 14 and 15 to regulate the rotational speed of the DC motor, so as to control the speed of the car.

G. 18650*2 Battery Holder

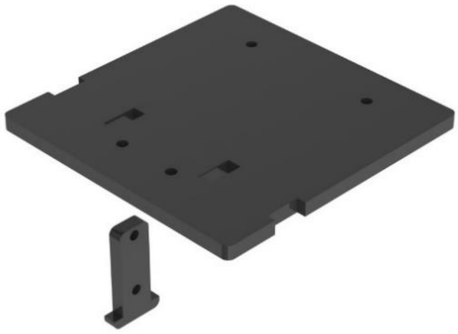
The battery is to use two 18650 batteries to power the modules and servos on the car.

Note: Please pay attention to the cathode (-) and anode (+) marks inside the holder (near each pole). Install the 18650 batteries accordingly: battery cathode to holder -, and battery anode to holder +.

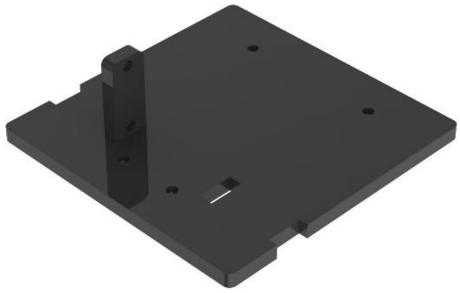
Assembly

A. Back Half Chassis + Rear Wheels

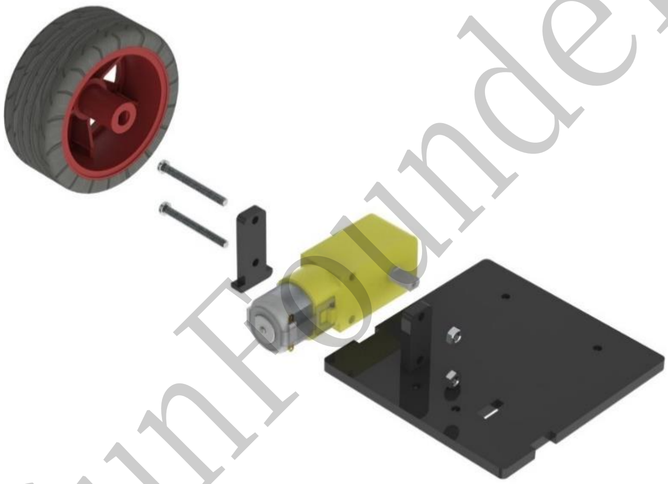
a) Assemble the following two acrylic plates



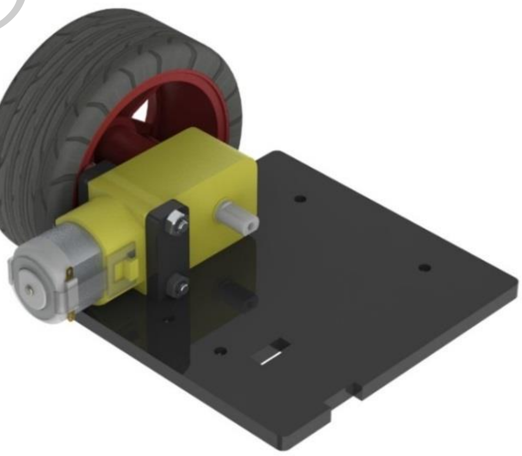
b) When completed, the assembly should look like the figure below.

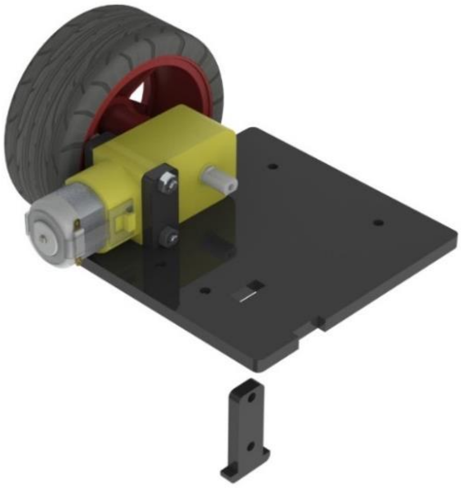
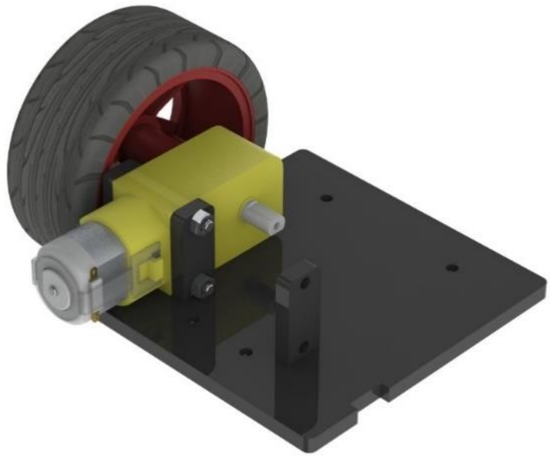


c) Assemble the gear reducer, the active wheel and following acrylic plates with two M3*30 screws and M3 nuts.

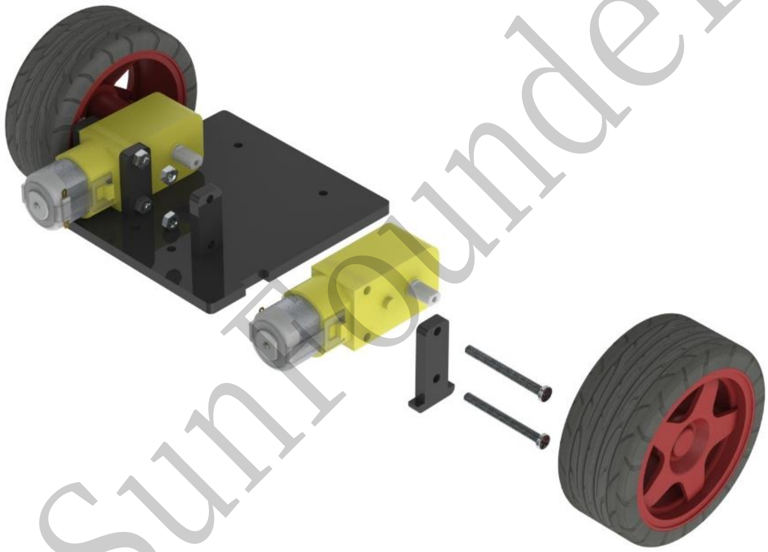


d) When completed, the assembly should look like the figure below.

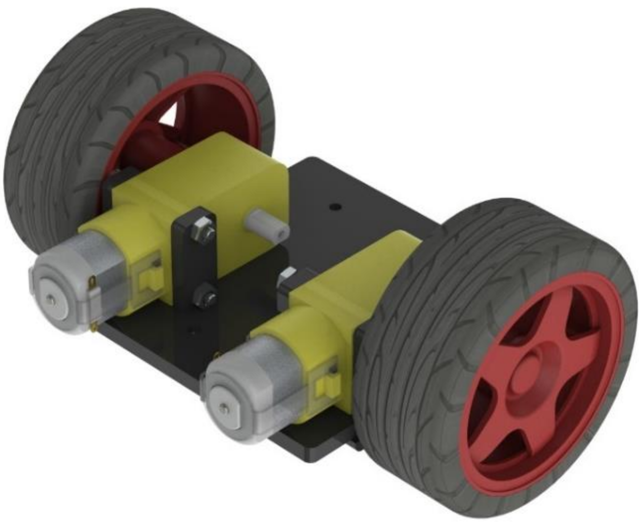


e) Assemble the following two acrylic plates 	f) When completed, the assembly should look like the figure below. 

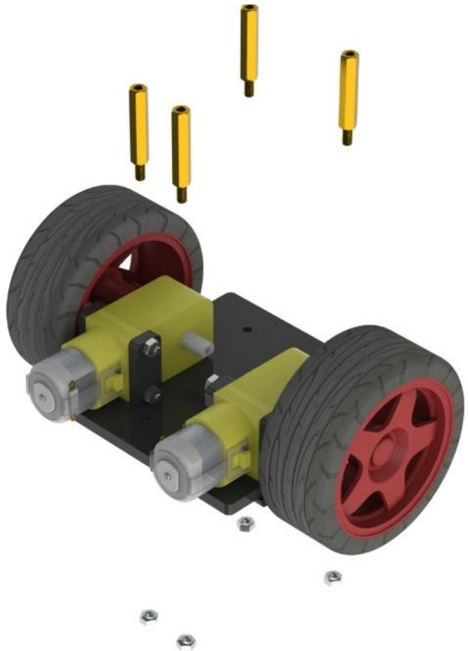
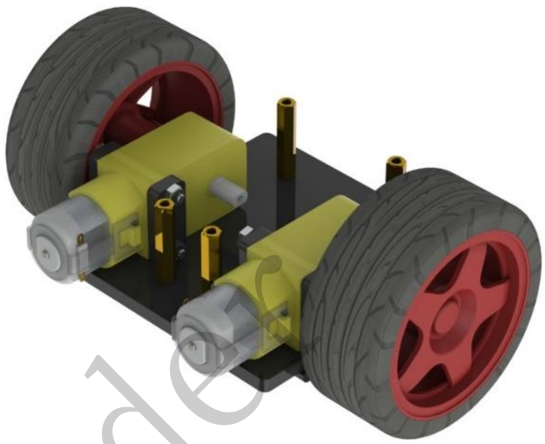
g) Assemble the gear reducer, the active wheel and the previously assembled part with two M3*30 screws and M3 nuts.



h) When completed, the assembly should look like the figure below.



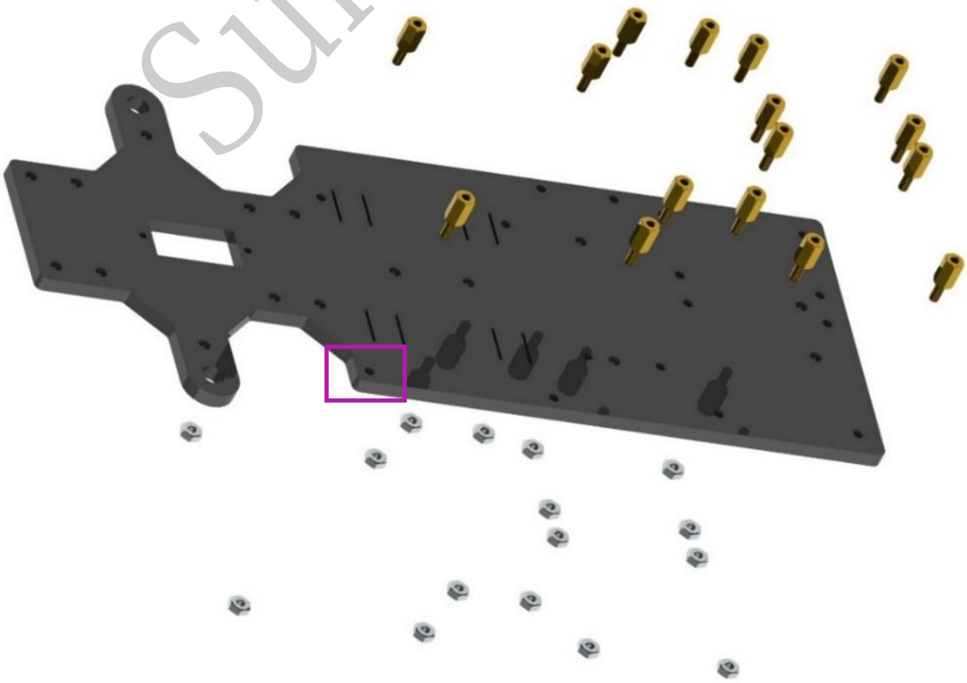
B. Back Half Chassis + Copper Standoffs

<table>
<tr>
<td>a) Assemble 4 M3*24 copper standoffs and 4 M3 nuts into the acrylic plate part as shown below.
</td>
<td>b) When completed, the assembly should look like the figure below.
</td>
</tr>
</table>

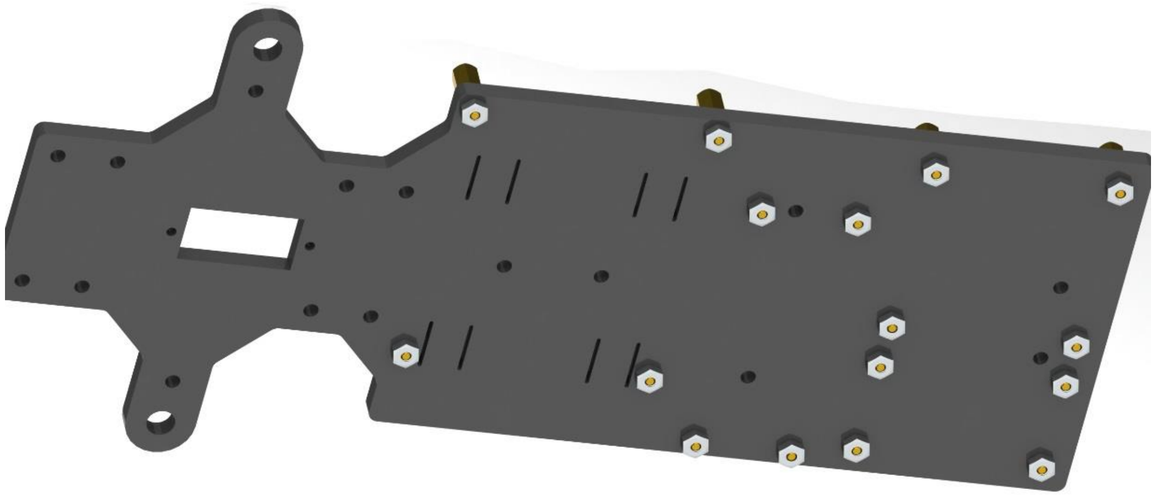
C. Upper Plate + Copper Standoffs

a) Assemble 16 M2.5*8 copper standoffs and 16 M2.5 nuts into the acrylic plate as shown below.

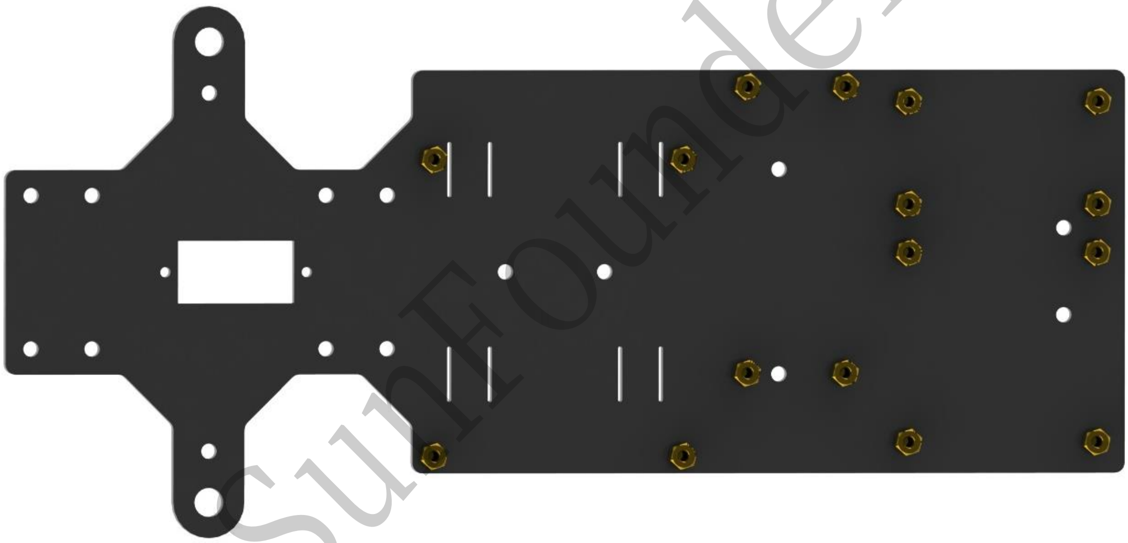
Pay attention to the face of the plate. Refer to the small hole in the plate as pointed by the arrow in the following figure.



b) When completed, the assembly should look like the figure below.



c) The view from the back of the plate:



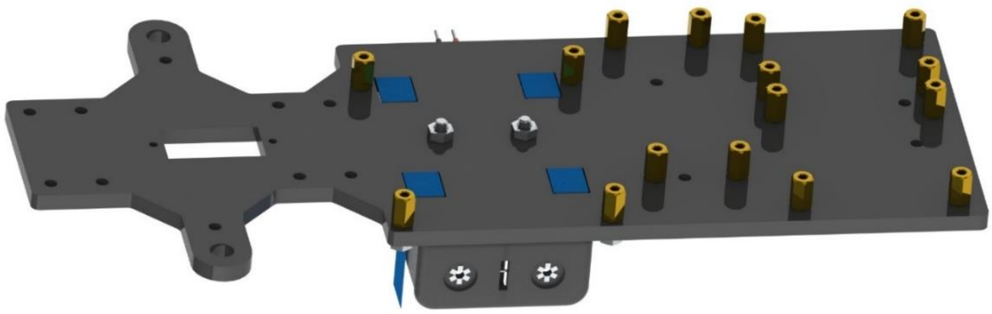
D. Battery Holder

a) Assemble the battery holder to the plate below with 2 M3*10 countersunk screws and 2 M3 nuts.

◆You can thread a ribbon through the plate below, so it will be easy to remove the battery, which is up to you.

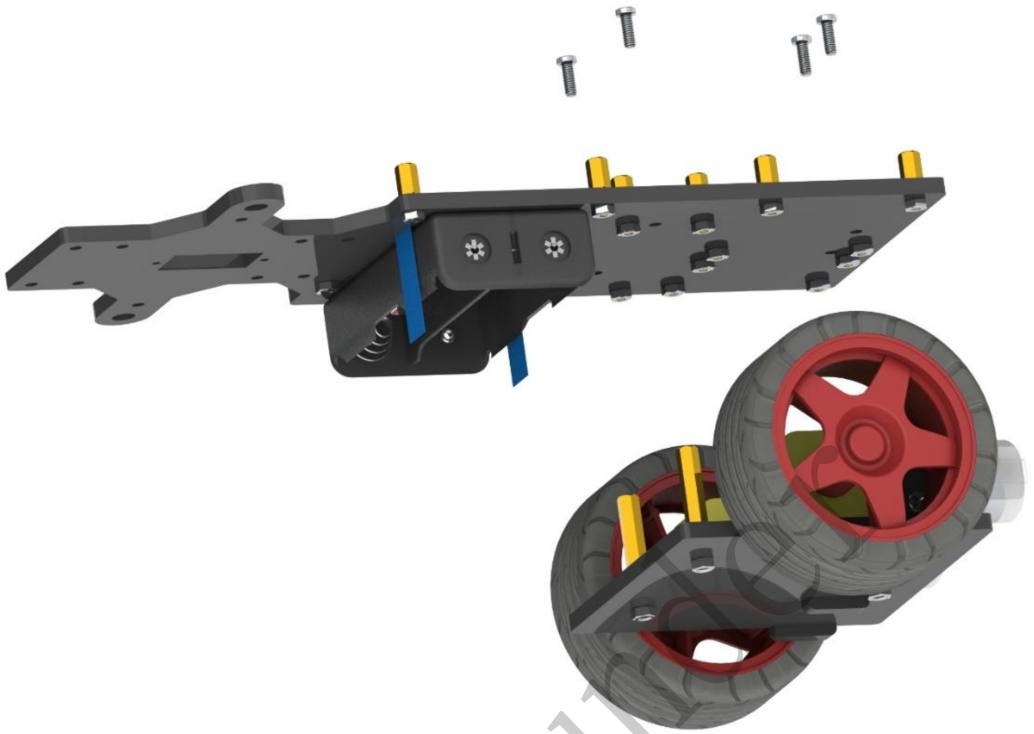
b) When completed, the assembly should look like the figure below.

c) The view from the top is as follows.

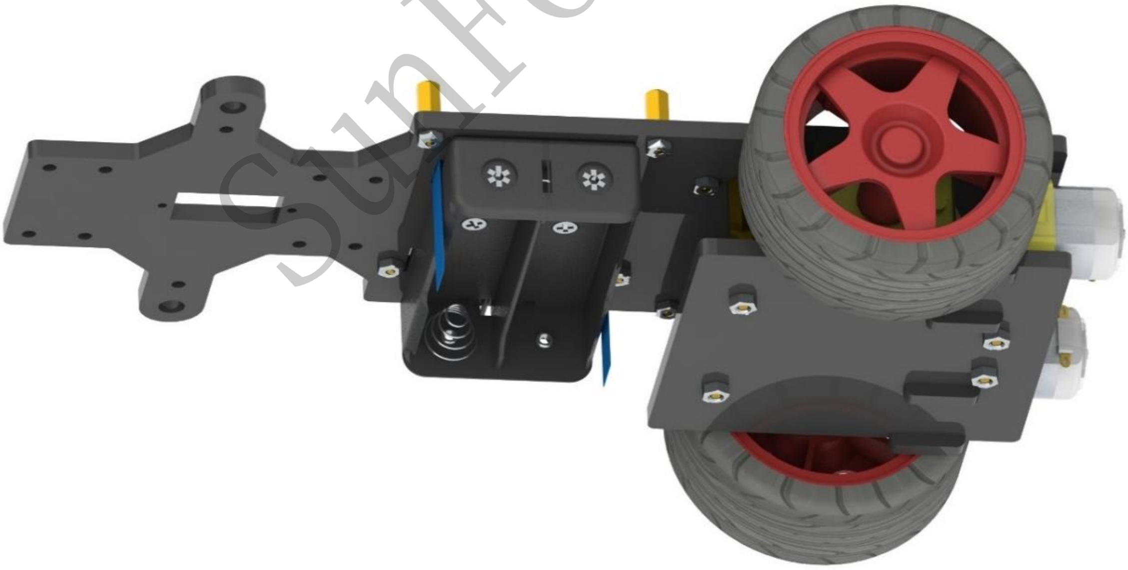


E. Back Chassis + Upper Plate

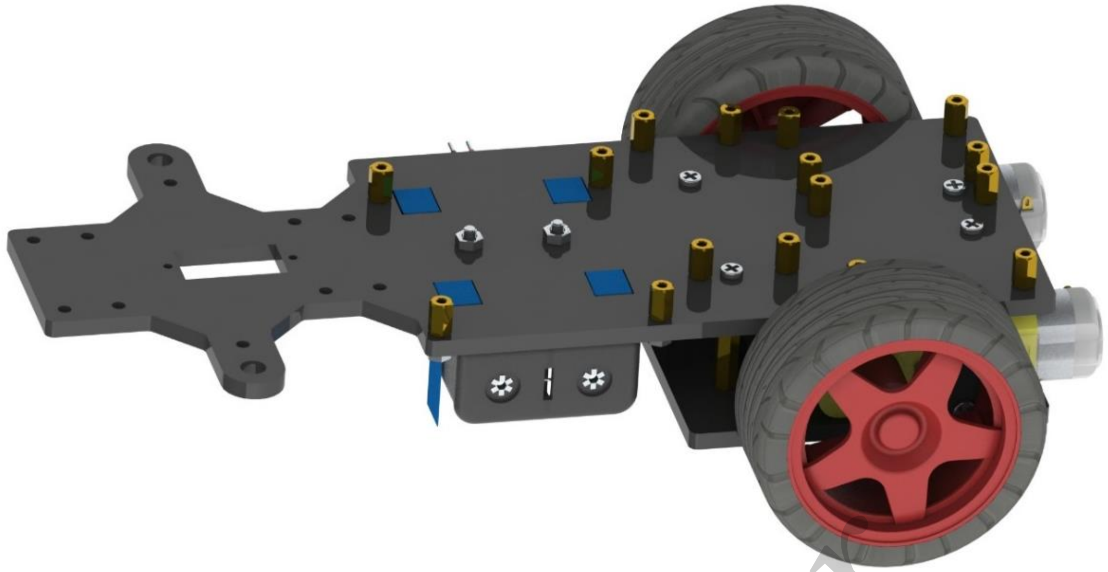
a) Connect the two assembled parts with 4 M3*8 screws.



b) When completed, the assembly should look like the figure below.



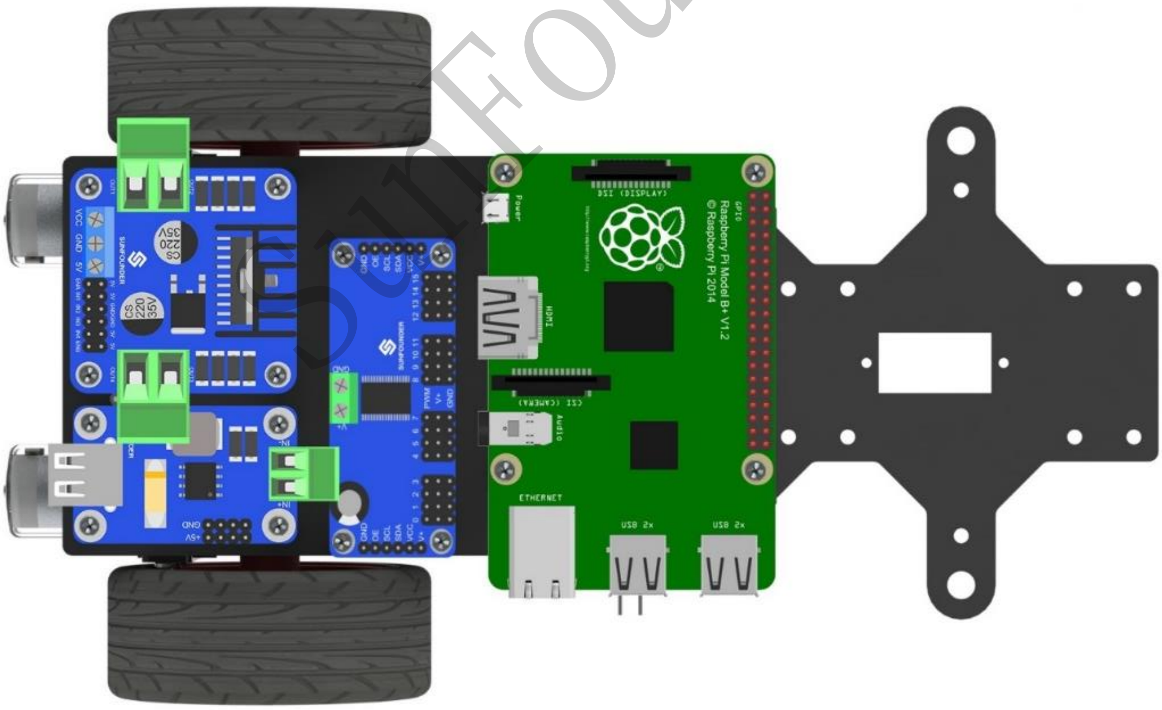
c) The view from the back of the plate:



F. Electrical Module Assembly

Assemble the electrical components to the car with M2.5*6 screws.

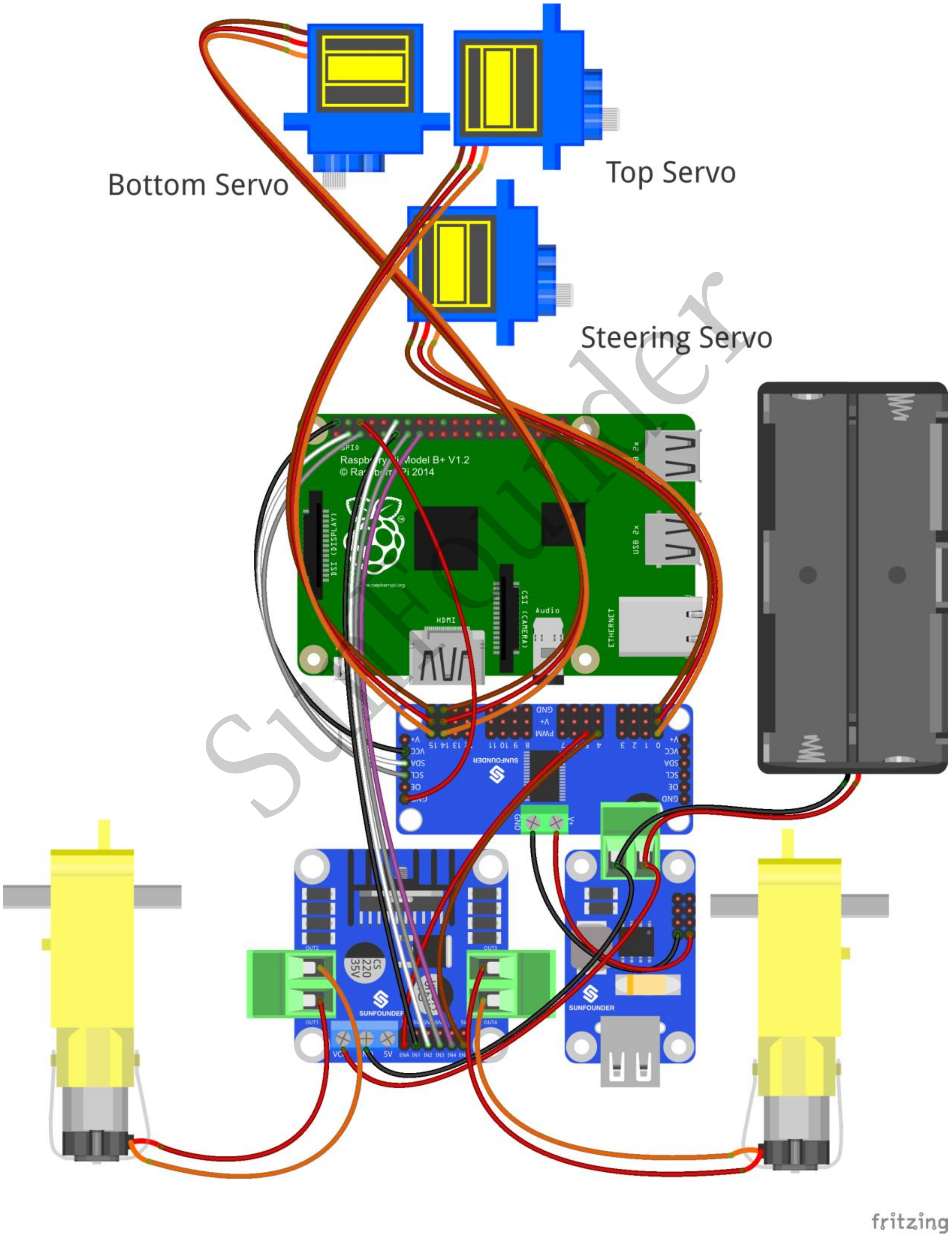
See the figure below.



Circuit Connecting

Preview:

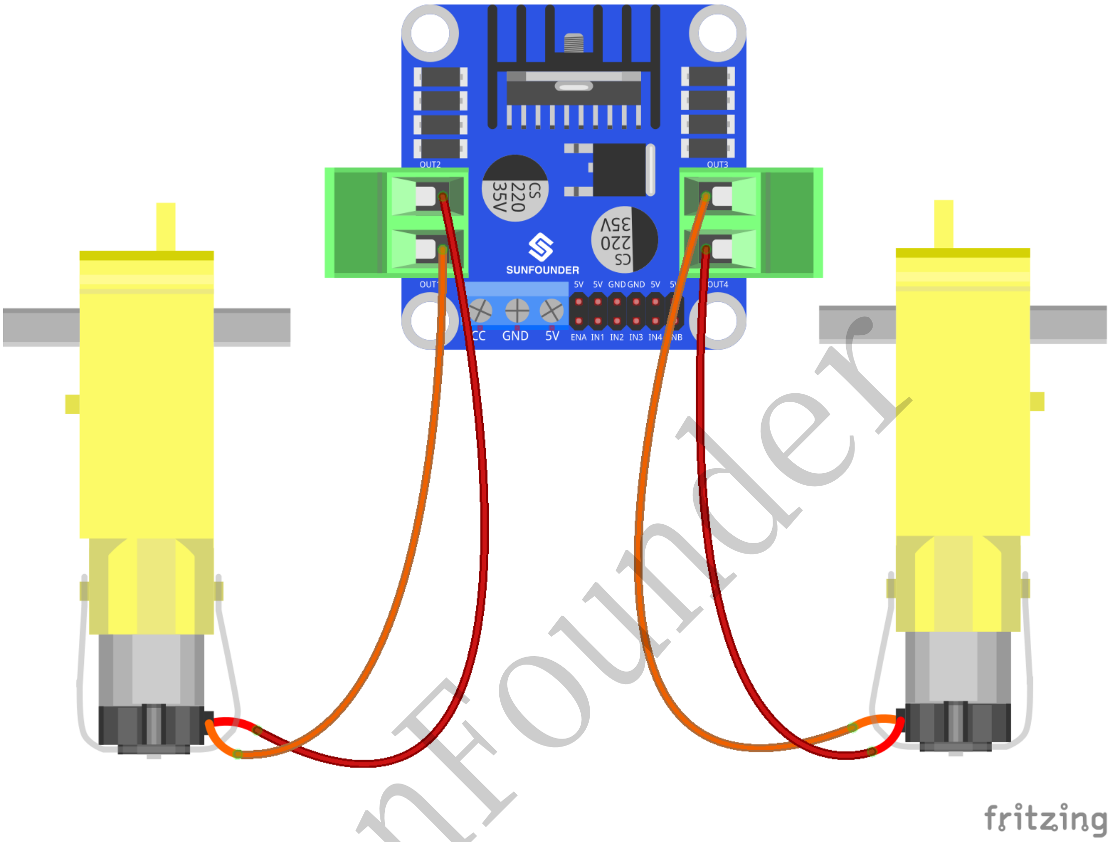
Looks complicated but don't worry! The detailed procedures will be given below, step by step.



Step 1: Connect the two DC motors with the motor driver.

You may remove the L-shaped PCB connector and plug it back after wiring.

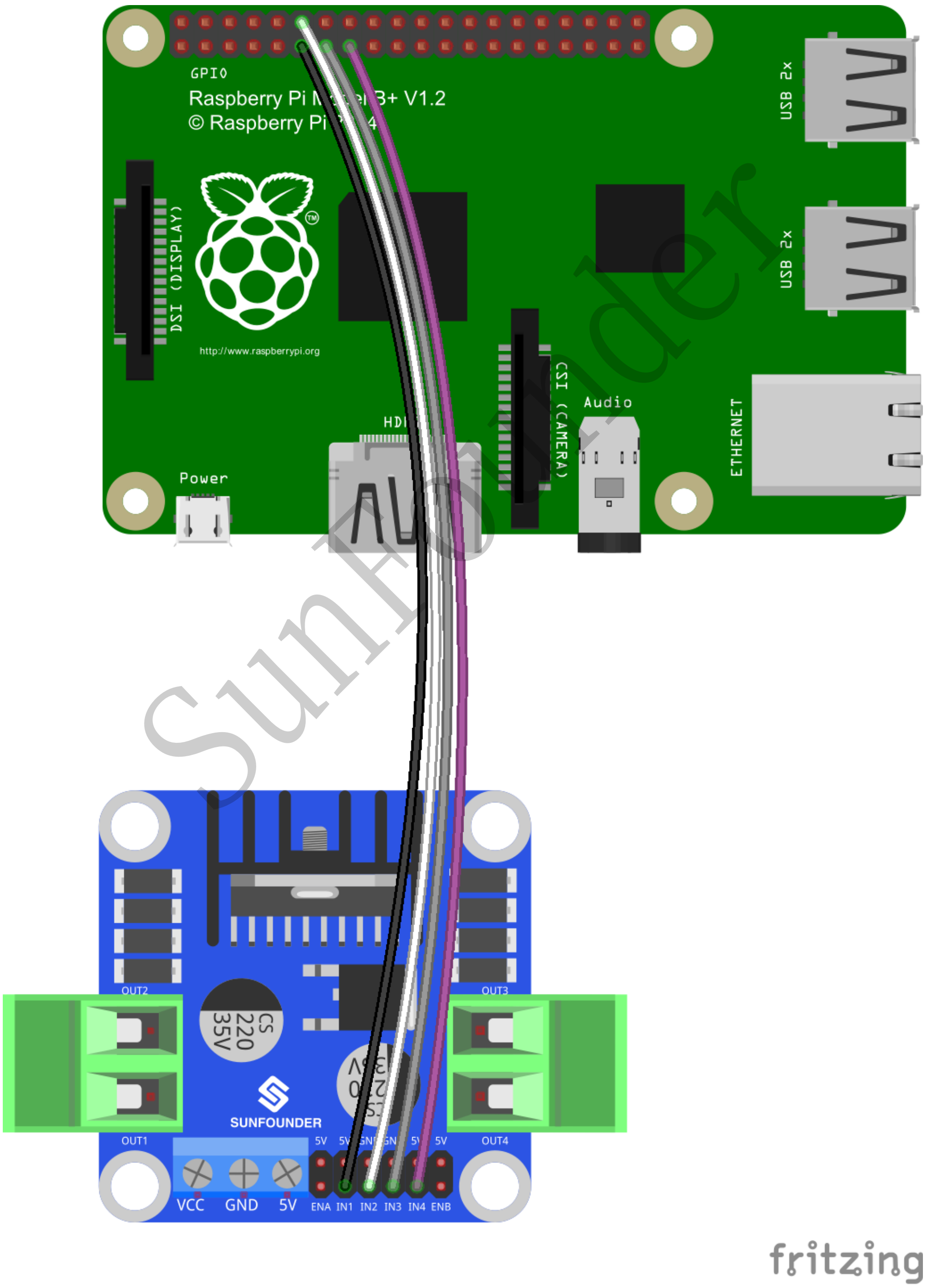
Note: It doesn't matter how to wire the motors. After all the assembly is done, if the car moves in an opposite direction of what you control, just swap the wiring of the two motors and it will work normally.



After wiring, it should be like this:

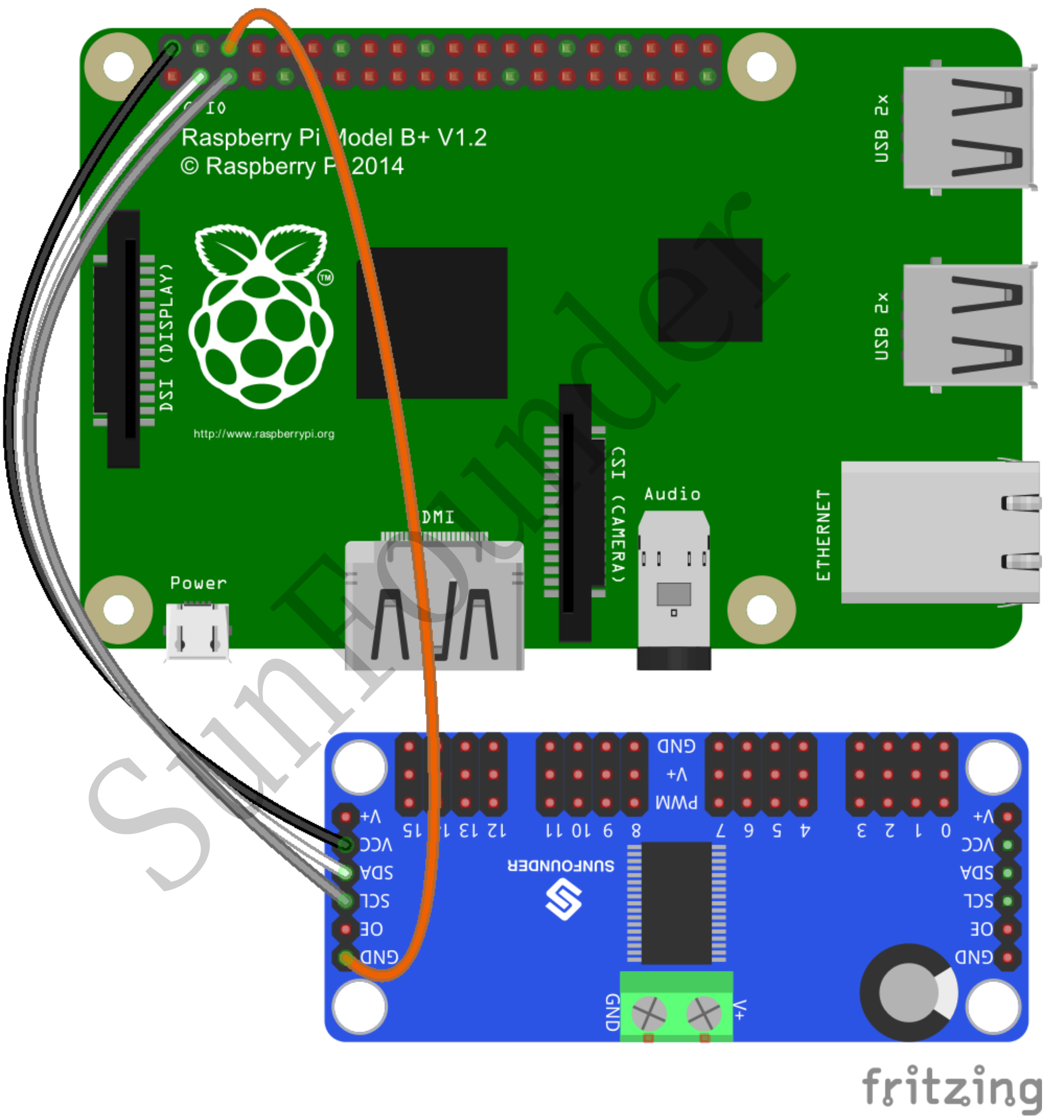
Step 2: Connect the motor driver with the Raspberry Pi GPIO port based on the following table.

Raspberry Pi GPIO Port	DC Motor Driver
Pin11	IN1
Pin12	IN2
Pin13	IN3
Pin15	IN4

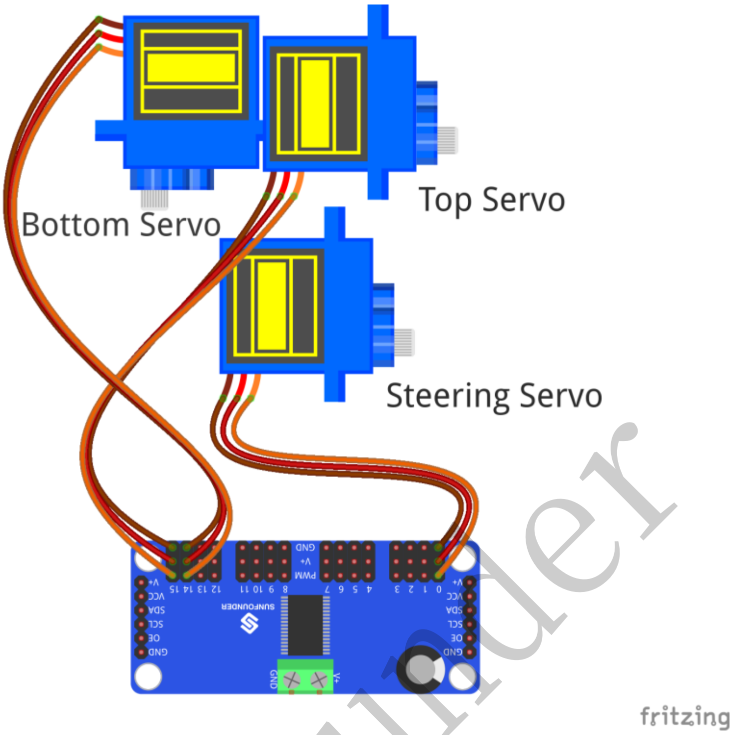


Step 3: Connect the servo controller with the Raspberry Pi GPIO port as follows:

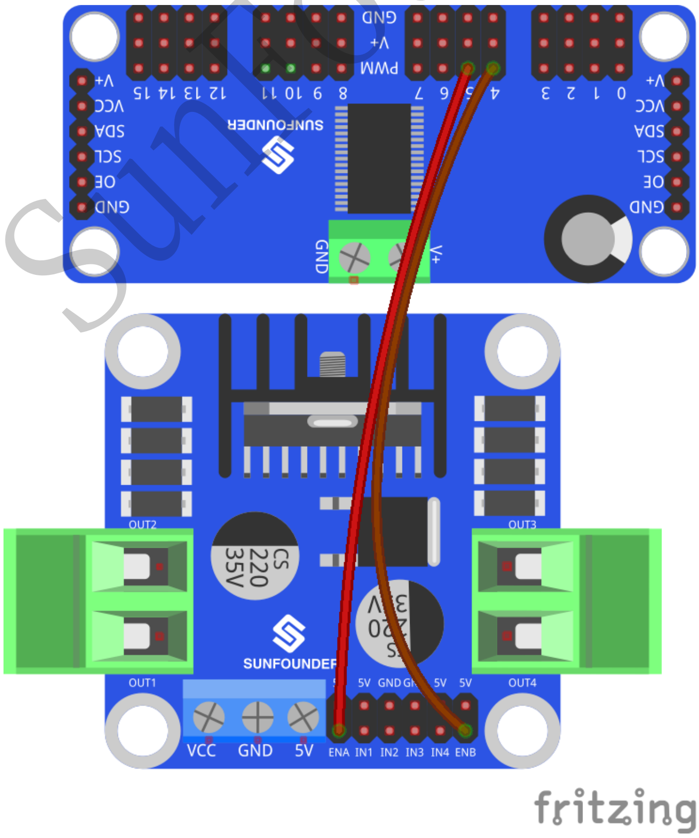
Raspberry Pi GPIO Port	Servo Controller
Pin 2 (5V)	VCC
Pin 3 (SDA)	SDA
Pin 5 (SCL)	SCL
GND	GND



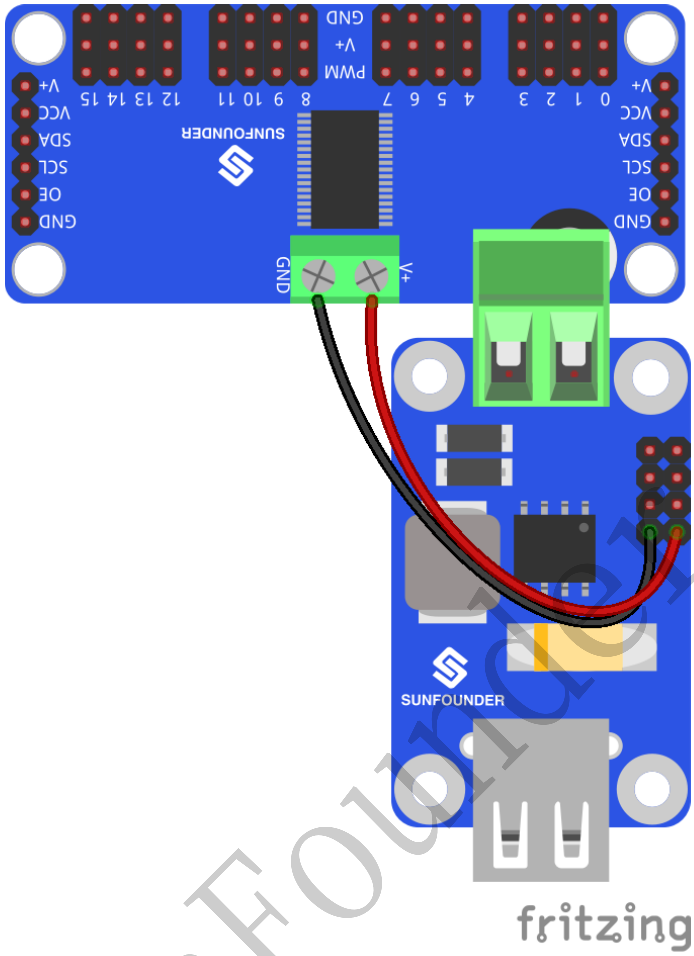
Step 4: Hook up the servo that controls the car's direction to CH0 of the servo controller, and the two servos that control the view of the camera to CH14 and CH15 respectively, as shown below:



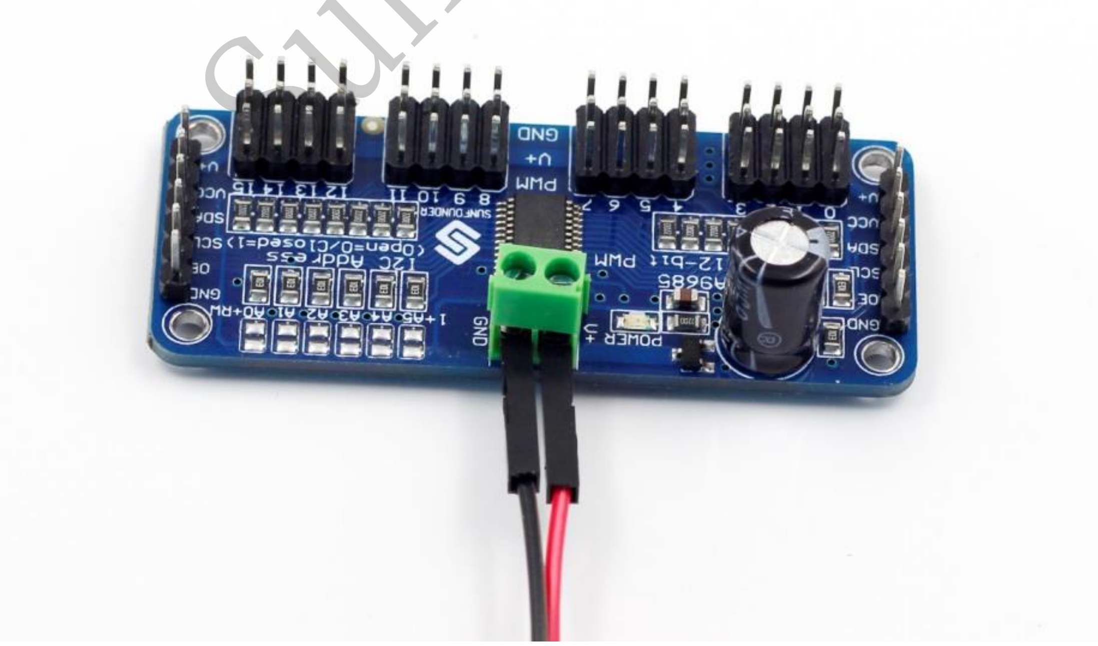
Step 5: Connect the motor driver with the servo controller.



Step 6: Connect the servo controller with the step-down DC-DC converter module. For the connector, loosen the screws, insert the wire, and then tighten the screws with a screwdriver.

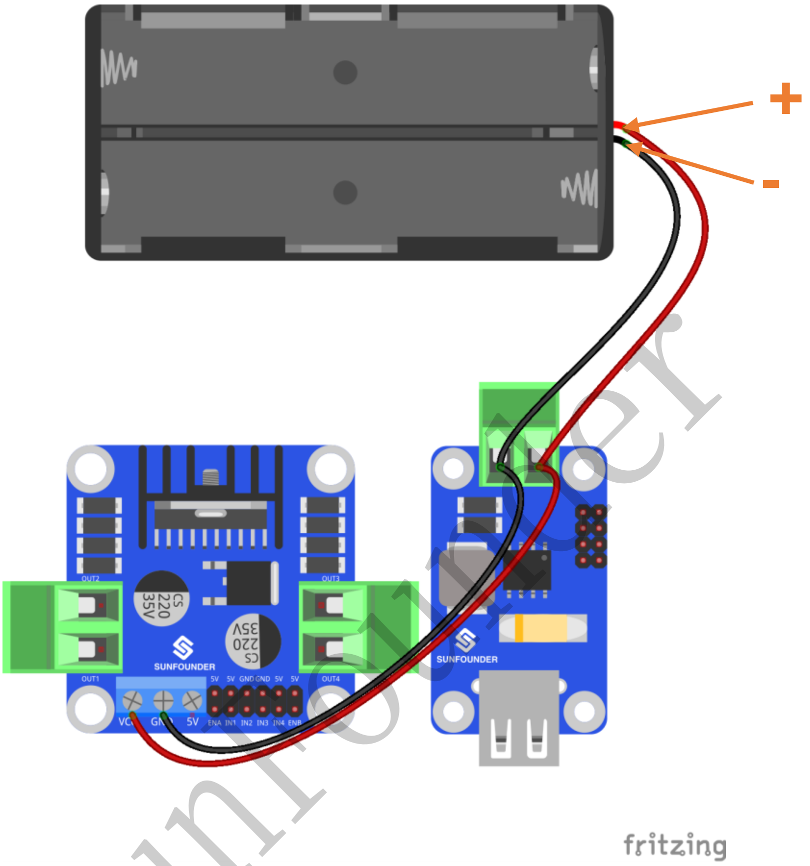


The connection should be like:

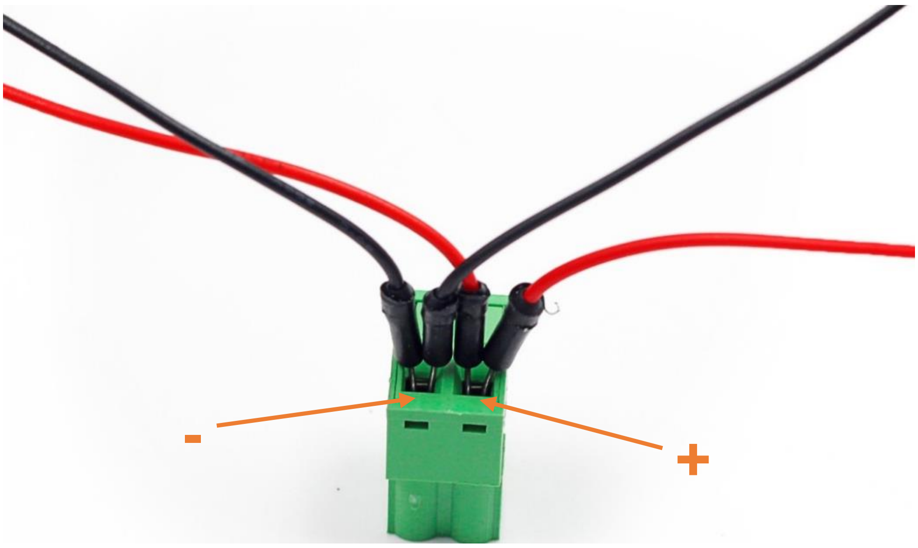


Step 7: Connect the battery holder with the step-down DC-DC converter module and the DC motor driver.

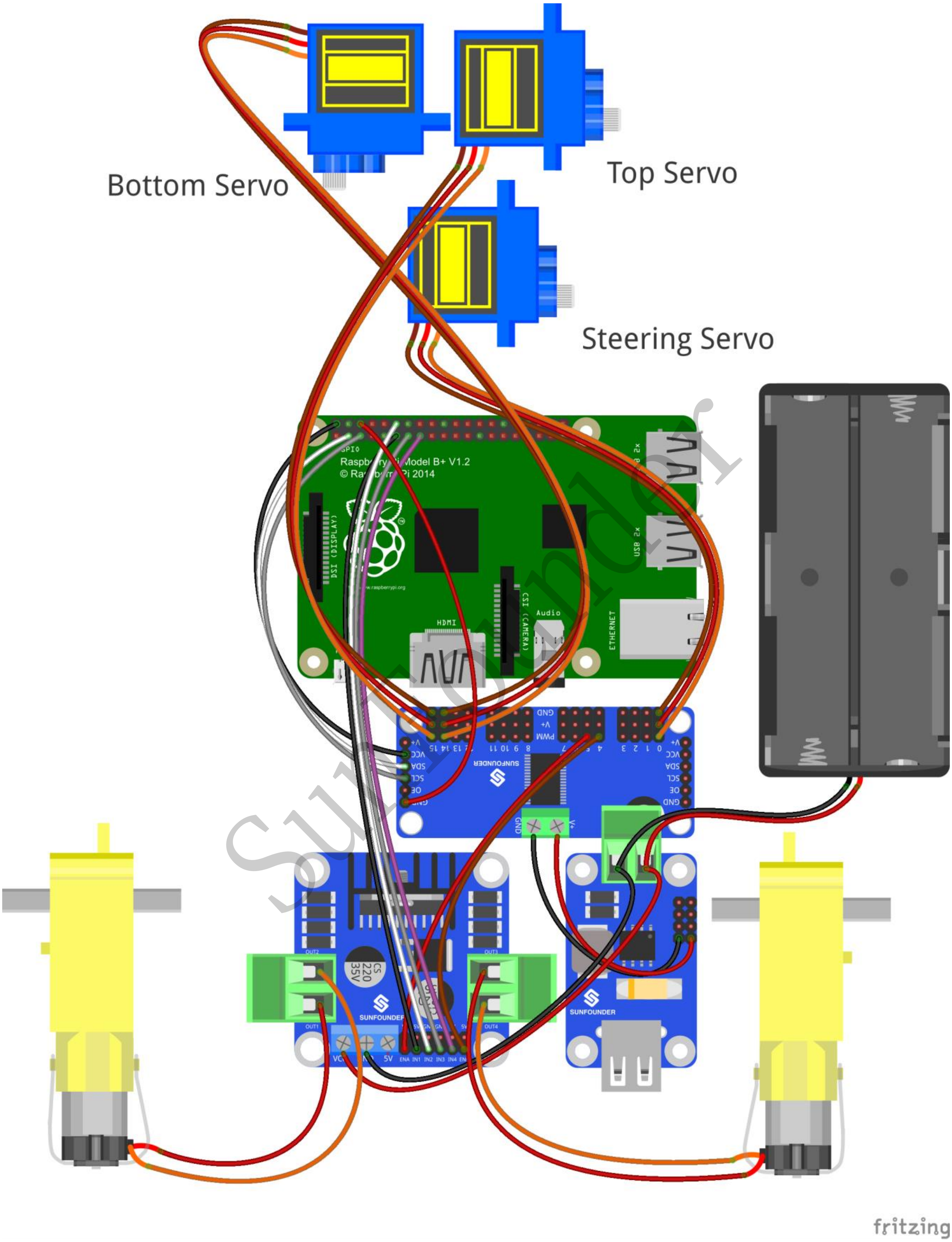
Note: Please DO NOT install the batteries at this step! Do it later when you've completed all the wiring. And take care that the red wire is the power and should be connected to the anode of the holder, when the black wire, i.e. the ground, to cathode.



Insert the four wires into two jacks of the connector, two in each. Pay attention: the red wires to the anode of the step-down module, when the black ones to the cathode. Also DO NOT plug a red wire with a black one into the same jack, or it'll cause a short circuit!



The whole picture of wiring should be like this:



Step 8: Connect the Raspberry Pi with the step-down DC-DC converter module, the USB Wi-Fi adapter and the USB camera.

Servo Calibration (Operation on Raspberry Pi)

In the subsequent part of Operation on Raspberry Pi, you need to log into the Raspberry Pi REMOTELY instead of direct operation on a screen connected to the RPi – which is also impossible because the HDMI port on the smart car is blocked and cannot connect to one such display.

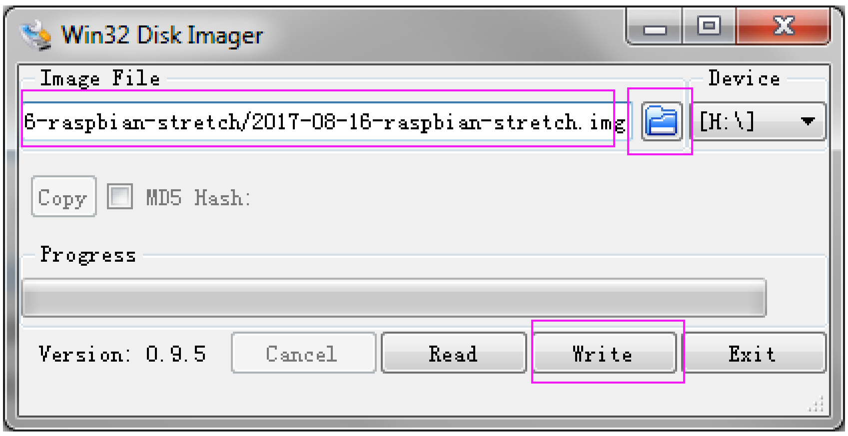
So the back part of the car is completed. Next we'll move on to the front part. But before assembly, since servos are used in this part, they need some configuration for protection. We need to make the servo rotate to the 90 degrees when it's mounted, so the rotating range can match with the application in the car. Otherwise, damages may be caused to servos. **PLEASE DO FOLLOW THIS STEP BEFORE MOVING ON!**

And since the servos used in this kit are adjusted by software and there's no such physical sticking point as other servos, here we need to configure the servo via software. First you need to finish some software installation before the configuration.

A. Burn the image

Take the following steps before inserting the TF card into the Raspberry Pi:

1) Prepare the tool of image burning. Such as **win32DiskImager**
2) Download the complete image on the official website at this link: https://www.raspberrypi.org/downloads/raspbian/. Both the RASPBIAN STRETCH WITH DESKTOP and RASPBIAN STRETCH LITE are available, but RASPBIAN STRETCH WITH DESKTOP would be a better choice if you have no special requirements.
3) Unzip the downloaded package and you will see the *xxxx-xx-xx-raspbian- stretch.img* file inside. **Note**: This file is **NOT** extractable.
4) Open the win32DiskImager and insert the SD card into the computer with a card reader. Select the image file and the drive (the card) in the tool. Click **Write** to write the system to the SD card. Wait for a while until the system is written to the card.



5) When the progress bar comes to the end and a prompt "**write successful**" appears, click **OK**.

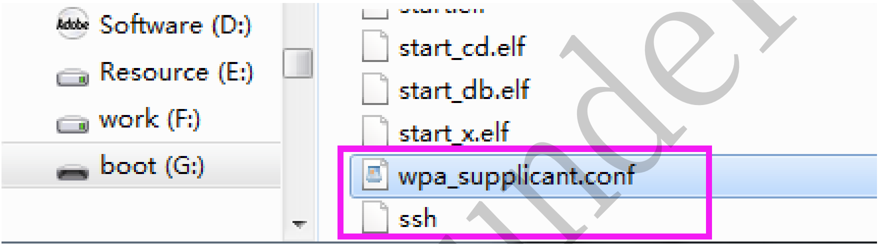
6) Create a blank file *ssh* under the */boot* directory to enable remote login and delete the suffix in the file name.

7) Create a WiFi configuration file *wpa_supplicant.conf* under */boot* and add the following to the file:

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=GB

networ={
        ssid="juhao"
        psk="sunfounder"
        key_mgmt=WPA-PSK

}
```

ssid= "(name of the Wi-Fi)"

psk= "(your Wi-Fi password)"



B. Car Power Supply

Now you can unplug the TF card from the PC and insert it into the Raspberry Pi. For power supply of the smart car, please note:

1) Supply the Raspberry Pi independently with a 5V/2A power – the software installation and calibration needs a long time and the battery won't be able to support.

2) Also keep the power on by battery – the servos and motors are powered by the battery, so ensure the power is on for them when the RPi is supplied separately.

Plug in the USB Wi-Fi adapter (skip this if you use a Raspberry Pi 3 with built-in WiFi device) and complete the setting in the way you're comfortable with.

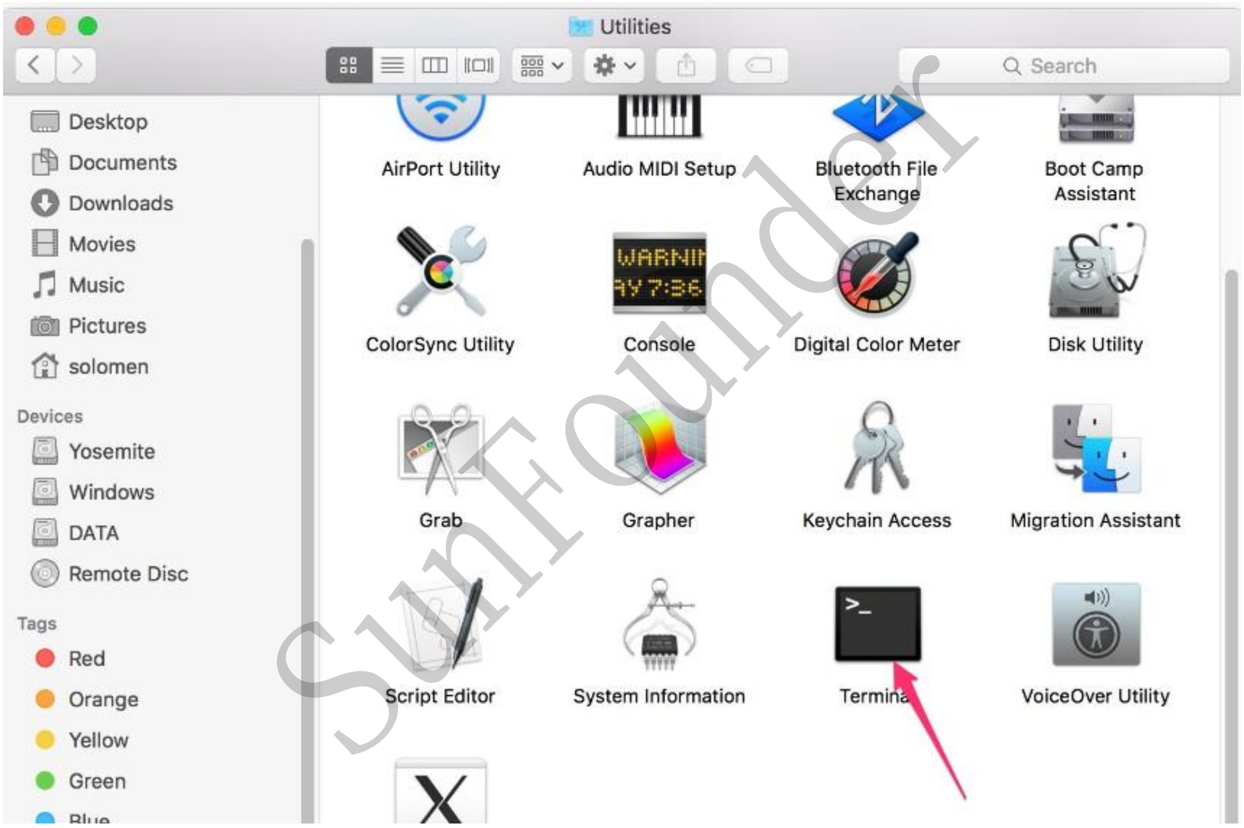
After the blinking indicator LED on the RPi turns into steady light on, you can check the IP address for the router. Now log into the RPi.

C. Log into Raspberry Pi

For Linux or Mac OS X Users

For Linux and Mac OS X users, you can only open the Bash shell via ssh.

Go to **Applications**->**Utilities**, find the **Terminal**, and open it.



Type in `ssh pi@ip_address` – ssh is the tool for remote login, `pi`, the user name, and `ip_address`, as the name suggests, your Pi's IP address. For example:

```
ssh pi@192.168.0.101
```

Press **Enter** to confirm. If you get a prompt that no ssh is found, you need to install an ssh tool like Ubuntu and Debian by yourself:

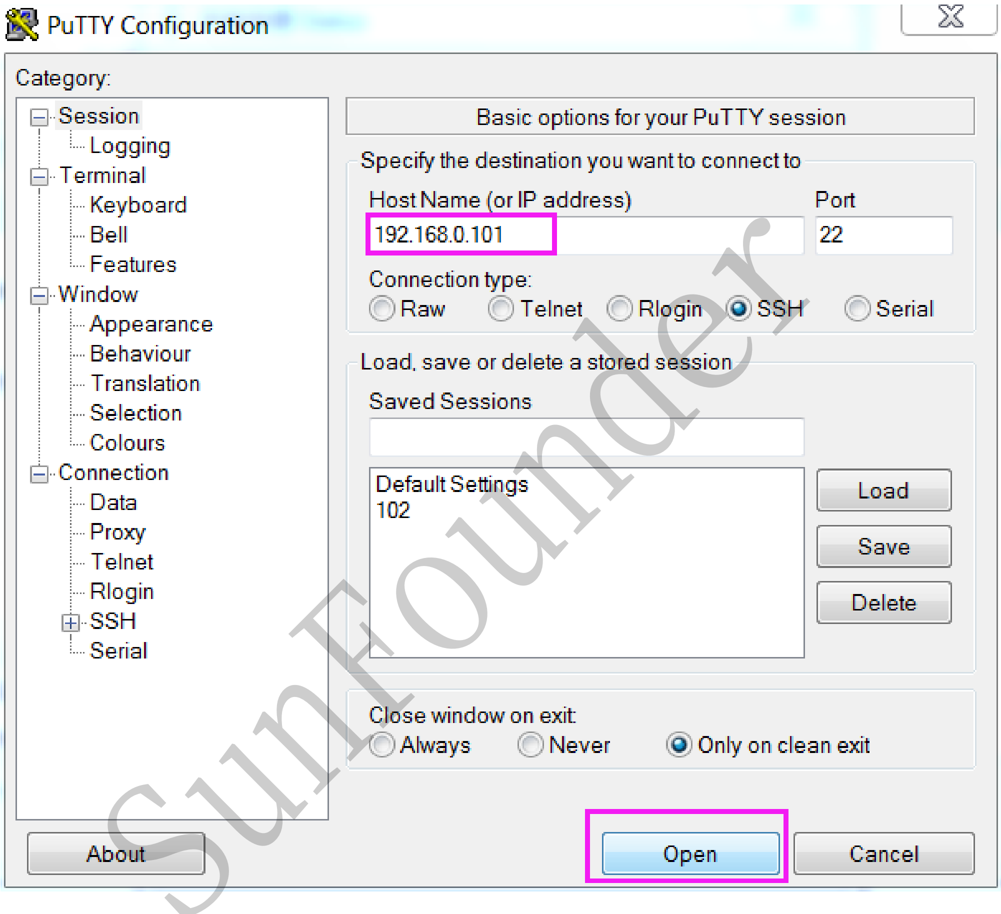
```
sudo apt-get install ssh
```

For other Linux platforms, please contact your supplier.

For Windows Users

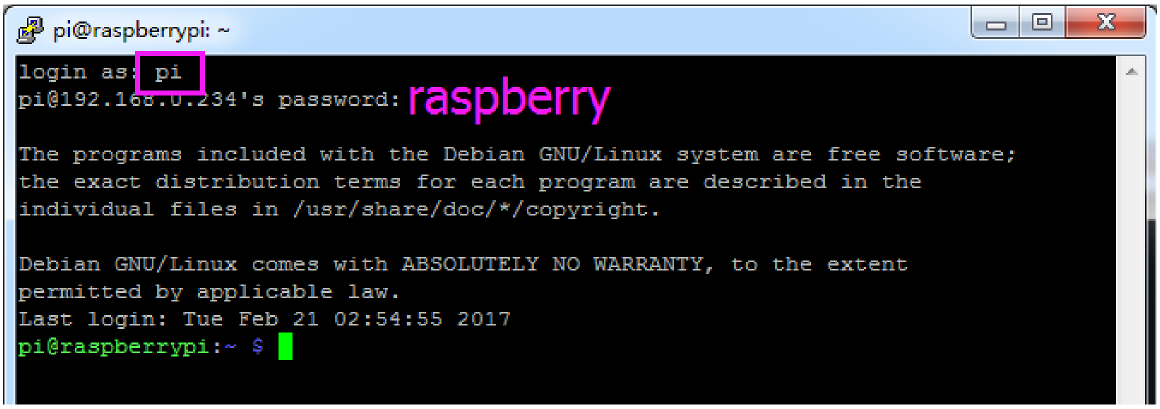
If your computer runs on Windows, you need to pen the Bash shell with the help of some software. Here we recommend a tool PuTTY.

1. Download PuTTY
2. Open PuTTY and click **Session** on the left tree-alike structure (generally it's collapsed upon PuTTY startup).Enter the IP address of the RPi you just got in the textbox under Host Name (or IP address) and 22 under Port (by default it is 22)



3. Click **Open**. Note that when you first log in to the Raspberry Pi with the IP address, you'll be prompted with a security reminder. Just click **Yes**. When the PuTTY window prompts login as: type in the user name of the RPi: pi, and password: raspberry (the default one, if you haven't changed it).

Note: When you're typing the password in, the window shows nothing just null, but you're in fact is typing things in. So just focus on typing it right and press Enter. After you log in the RPi successfully, the window will display as follows.



This window is just like the Command Line window in Linux.

D. Get Source Code

Download the source code directly from Github to your Raspberry Pi.

```
cd ~
git clone https://github.com/sunfounder/Sunfounder_Smart_Video_Car_Kit_for_RaspberryPi.git
```

E. Install python-dev, python-smbus

Install `python-dev` and `python-smbus`:

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install python-dev
sudo apt-get install python-smbus
```

F. Setup I2C Port

Run the command to open **Raspberry Pi Software Configuration Tool (raspi-config)**

```
sudo raspi-config
```

Enable I2C:

Select **Advanced Options** => **I2C** => **<Yes>** => **<Ok>** => **<Yes>**

Select **<Finish>**. Close the window.

If a message of rebooting appears, click **<No>**. Before reboot, we still need to complete some configurations.

G. Start calibration

Run the calibration server on the Raspberry Pi, and wait for client to connect.

Make sure that the circuit is connected properly. Power the smart car, open a terminal in Linux. Connect with your Raspberry Pi via the *ssh*. Go to the directory *Sunfounder_Smart_Video_Car_Kit_for_RaspberryPi/server*, run the server *cali_server.py*: if the servo gets stuck with an abnormal sound, unplug the wires at once, and redo servo adjustment for calibration in case of further damages.

```
cd ~/Sunfounder_Smart_Video_Car_Kit_for_RaspberryPi/server
sudo python cali_server.py
```

```
pi@raspberrypi ~ $ cd Sunfounder_Smart_Video_Car_Kit_for_RaspberryPi/server/
pi@raspberrypi ~/Sunfounder_Smart_Video_Car_Kit_for_RaspberryPi/server $ sudo python cali_server.py
offset_x = 0
offset_y = 0
offset = 0
turning0 = True
turning1 = True
Waiting for connection...
```

Then the contents of *config* are printed and the last line: **Waiting for connection…**

At this time, the servo might move a bit (this is the original position set).

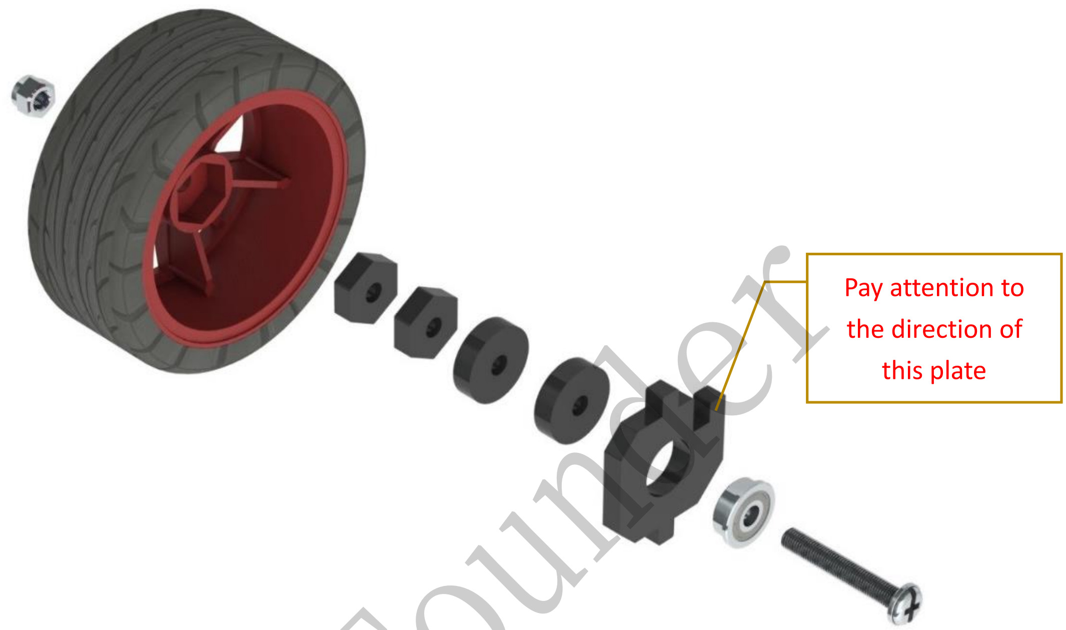
Note: During the subsequent assembly, cali_server.py should keep running.

Continue to Assemble

Note: Please keep cali_server.py running in the whole process of assembly.

A. Front Wheels

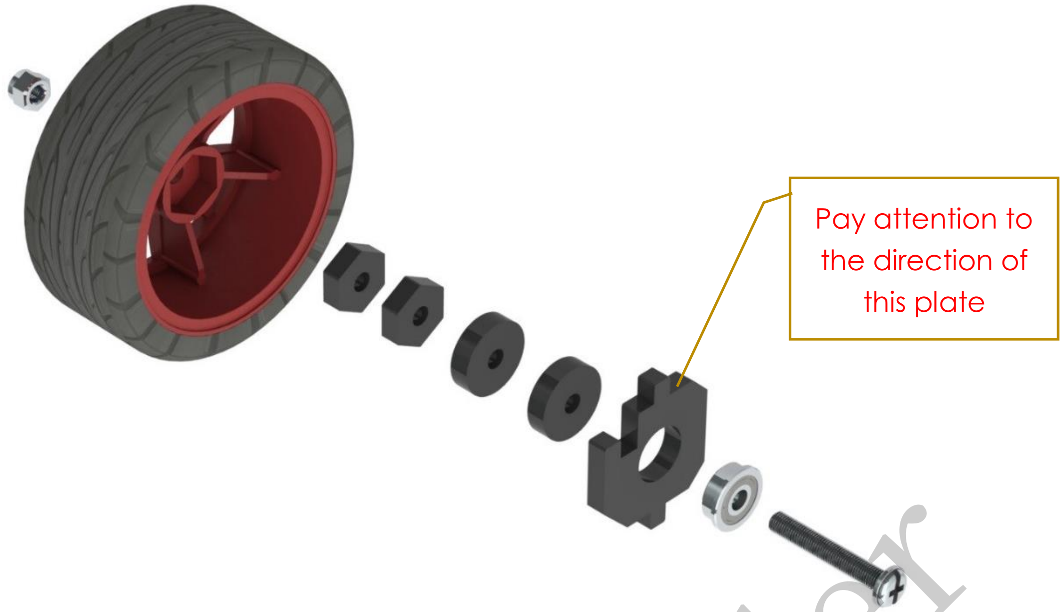
a) Fasten the F694ZZ flange bearing, driven wheel and following acrylic plates with an M4*25 screw and an M4 self-locking nut in the way as shown below. Use the cross socket wrench to fasten the M4 self-locking nut.



b) When completed, the assembly should look like the figure below.

c) Bear in mind that DO NOT over-tighten the nut or else the wheel cannot turn flexibly. Neither too loose, in case the gap between the parts is too large.

d) Fasten the F694ZZ flange bearing, driven wheel and following acrylic plates with an M4*25 screw and an M4 self-locking nut in the way shown as below.



e) When completed, the assembly should look like the figure below.

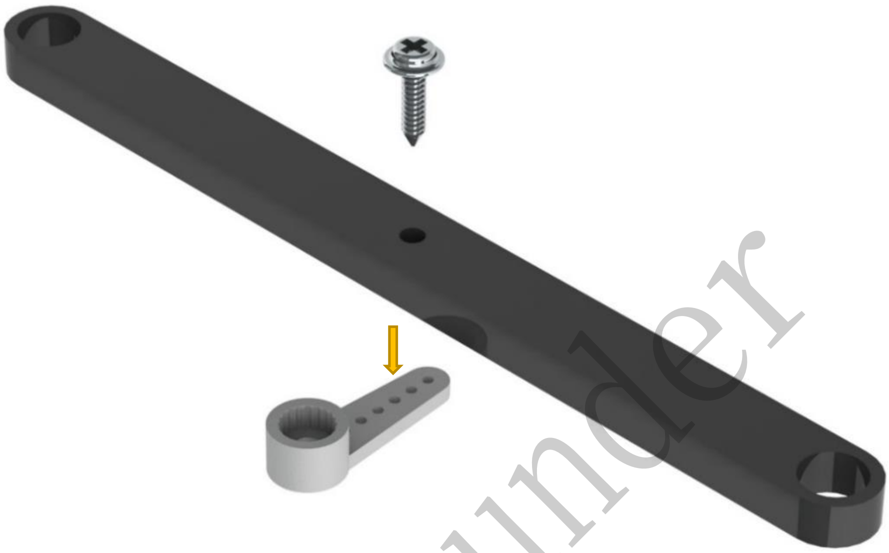
f) Bear in mind that DO NOT over tighten the nut or else the wheel cannot turn flexibly. Neither too loose, in case the gap between the parts is too large.

B. Steering Linkage + Servo Rocker Arm

a) Connect the following acrylic plate to the second hole of the rocker arm (see the figure below) with an M2*8 self-tapping screw.

◆ The M2*8 self-tapping screw is contained in the package of the servo; it is one of the two longest screws.

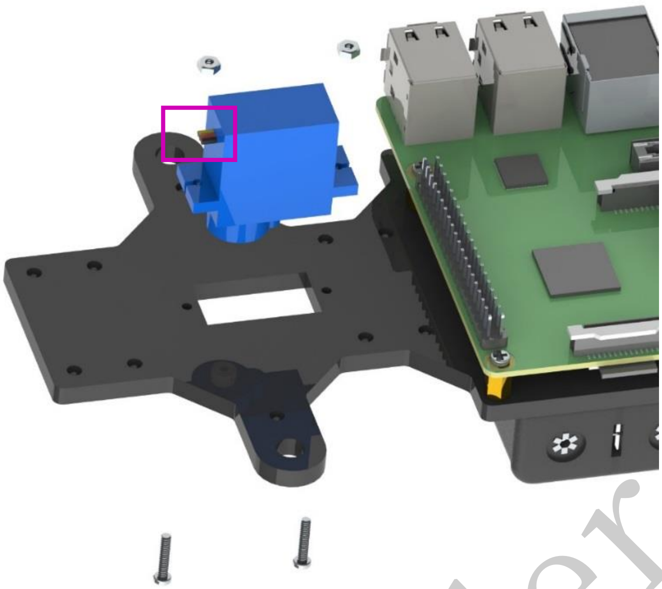
◆ Also the rocker arm is packaged together with the servo. *Note*: Be careful with the screw of pricking your fingers.



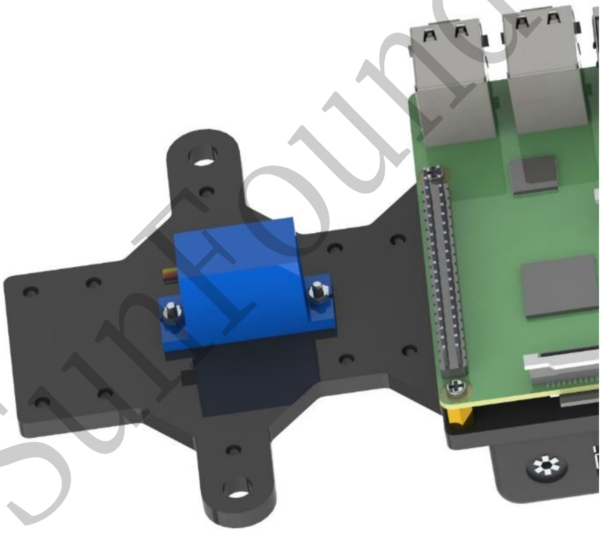
b) Tighten the screw first, then loosen it a little to ensure the Steering Linkage's movement, the assembly should look like the figure below.

C. Steering Servo + Upper Plate

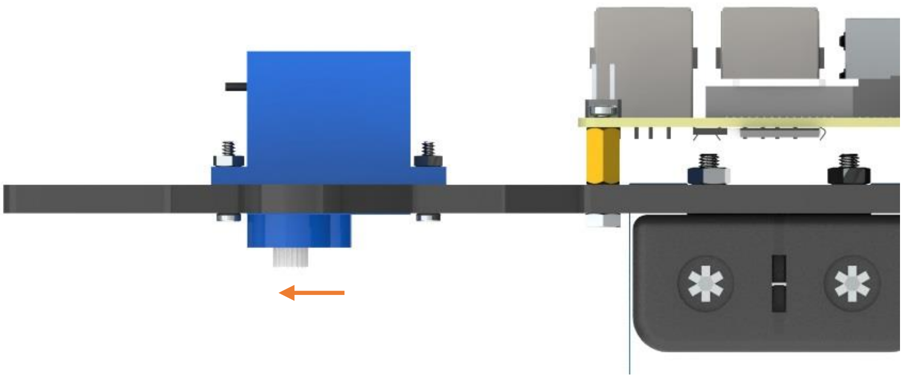
a) Connect the servo to the acrylic plate below with two M2*8 screws and M2 nuts.



b) When completed, the assembly should look like the figure below.



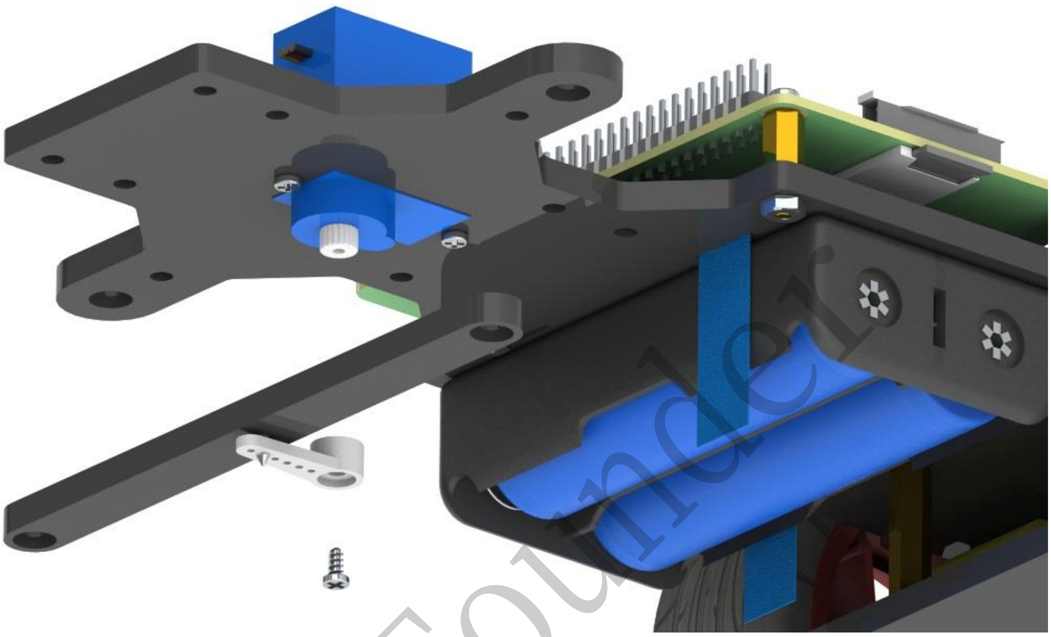
◆ **Note:** The servo should be placed with its output shaft toward the front of the plate (see the following figure).



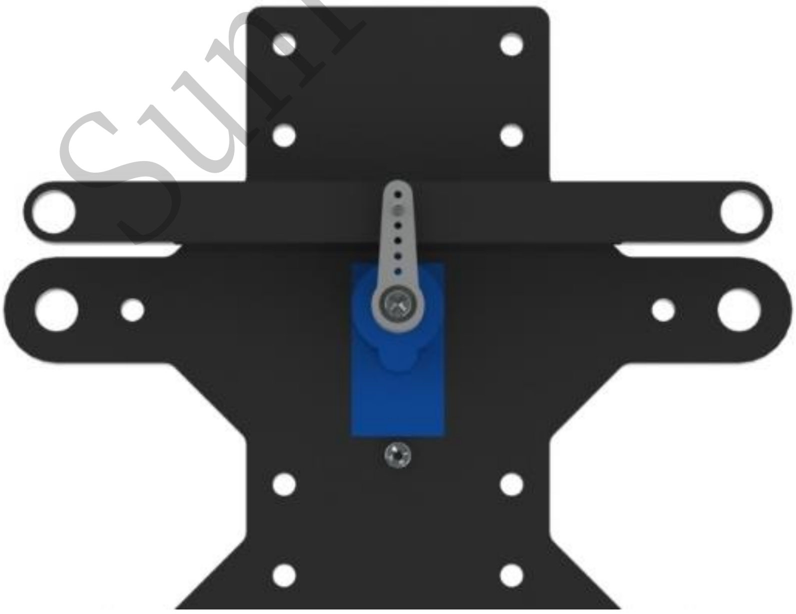
D. Steering Servo + Steering Linkage

a) Connect the following parts with an M2*4 screw.

- The M2*4 screw is contained in the package of the servo; it is the shortest of the screws in the package.
- Keep the Steering Linkage perpendicular with the middle of the car; otherwise, remove it and mount again. DO NOT directly turn it in case of damaging the servo.
- Minor errors are allowed, which can corrected later by programming.



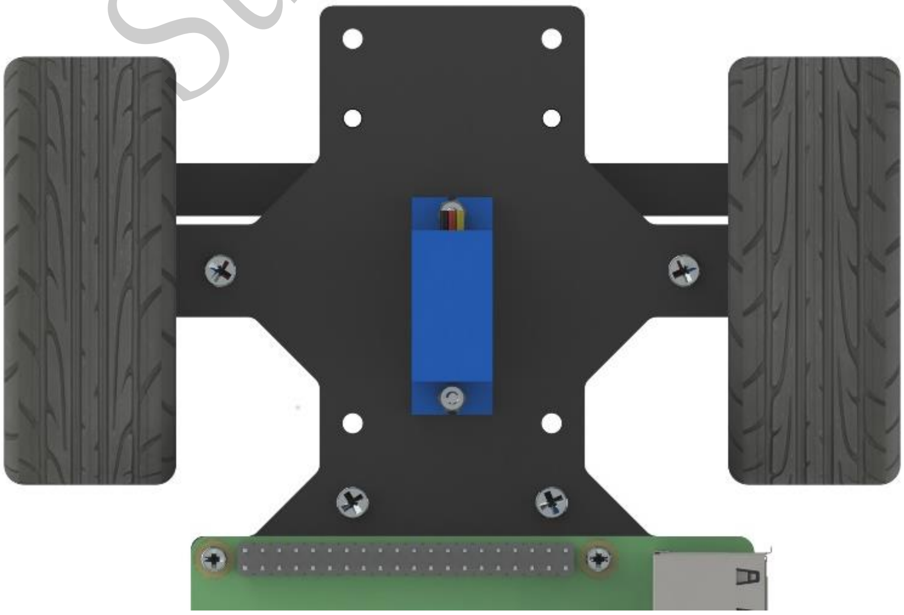
b) When completed, the assembly should look like the figure below.



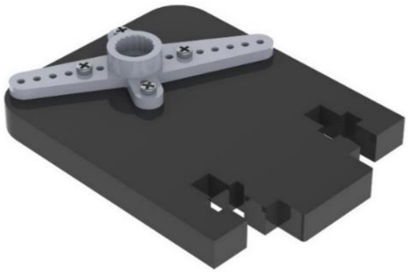
E. Front Chassis + Upper Plate

a) Connect the following parts and wheels with M3*8 screws, M3*24 copper standoffs and M3 nuts, 4 for each.

b) When completed, the assembly should look like the figure below.

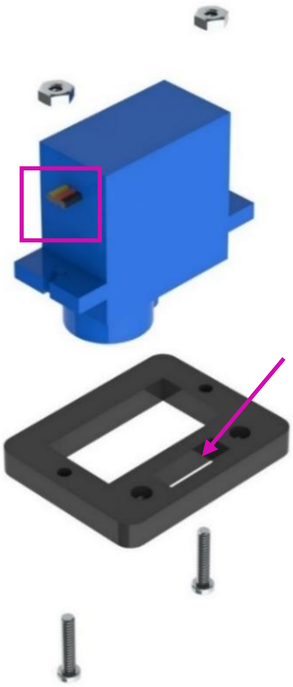


F. Plates + Servo Rocker Arms

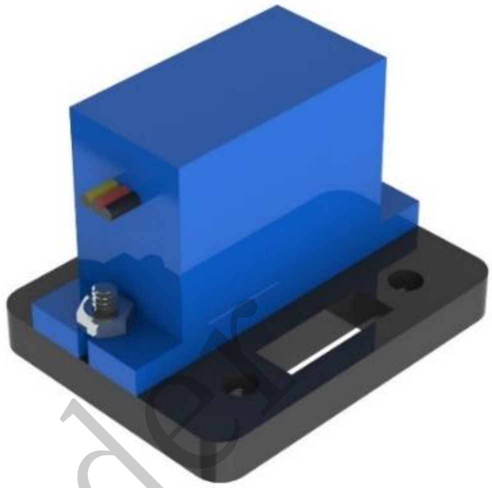
<table>
<tr>
<td>a) Connect the rocker arm of the servo to the acrylic plate below with 4 M1.2*4 screws.
◆The rocker arm is packaged together with the servo.
</td>
<td>b) When completed, the assembly should be like the figure below.
</td>
</tr>
<tr>
<td>c) Connect the rocker arm of the servo to the acrylic plate below with four M1.2*4 screws.
◆The rocker arm is packaged together with the servo.
</td>
<td>d) When completed, the assembly should be like the figure below.
</td>
</tr>
</table>

G. Pan/Tilt Servo + Plate

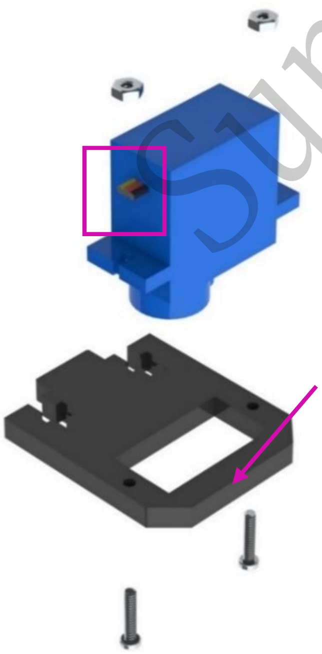
a) Connect the servo to the acrylic plate below with two M2*8 screws and M2 nuts, and we name it "**pan servo**".



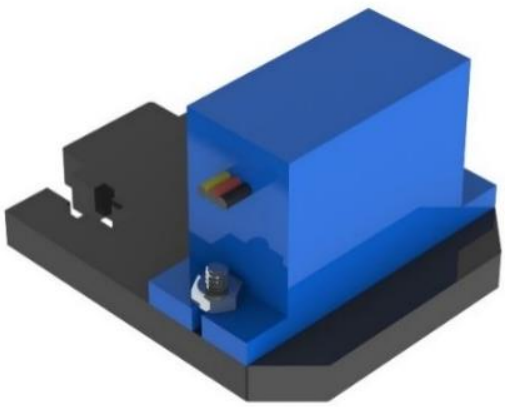
b) When completed, the assembly should look like the figure below.



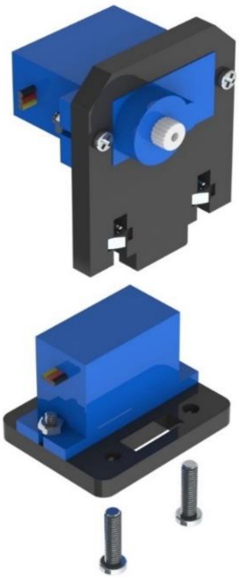
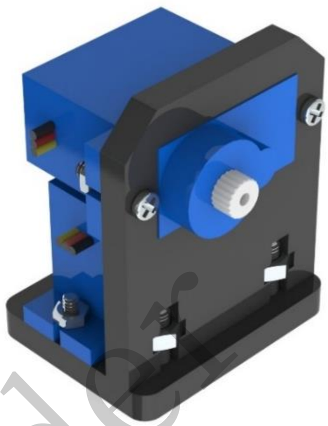
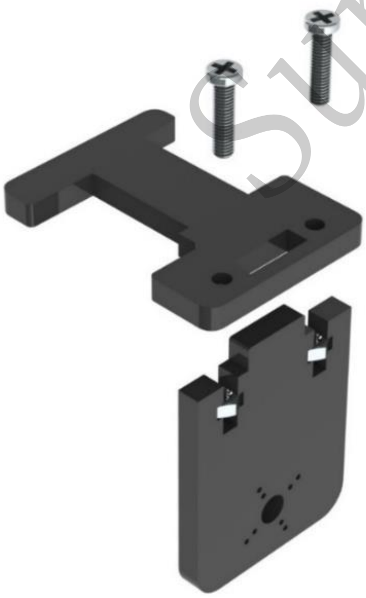
c) Connect the servo to the acrylic plate below with two M2*8 screws and M2 nuts and we name it "**tilt servo**".



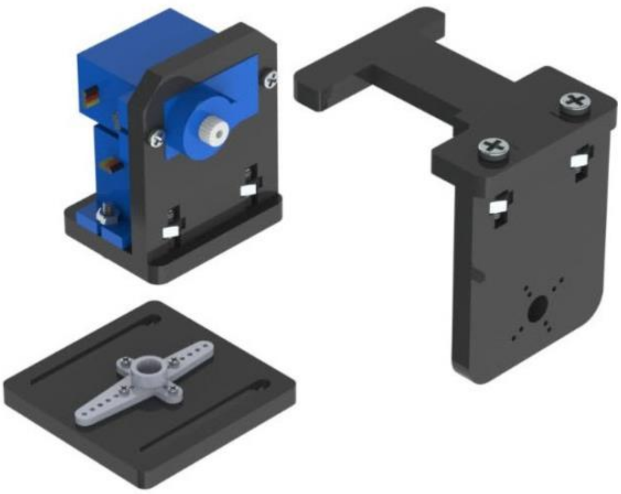
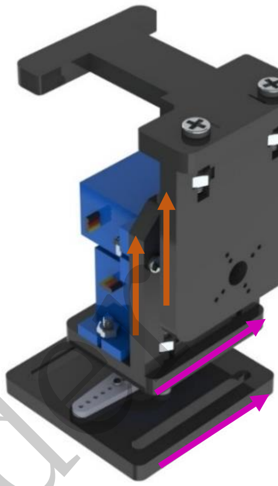
d) When completed, the assembly should look like the figure below.



H. Pan Servo Plate + Tilt Servo Plate

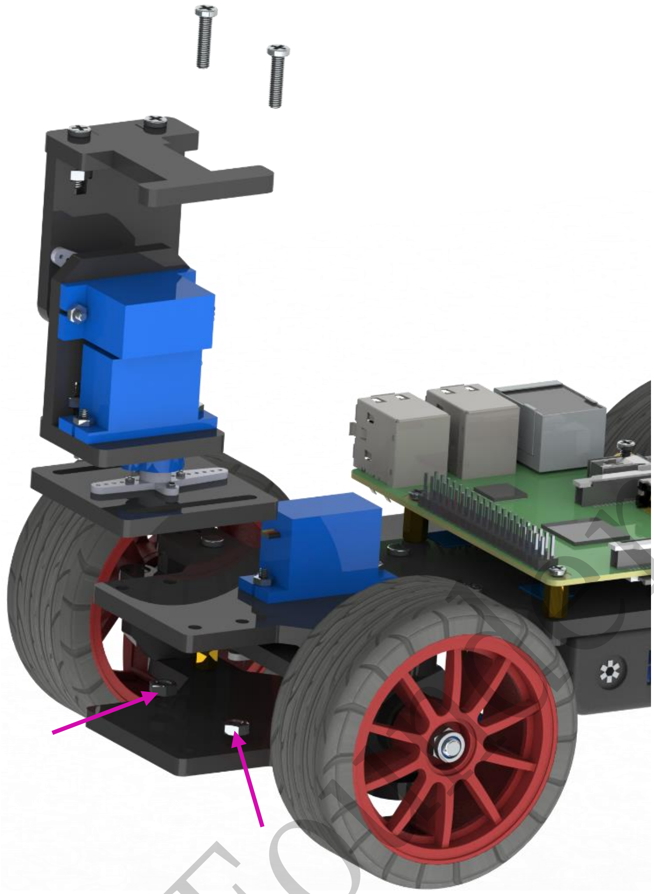
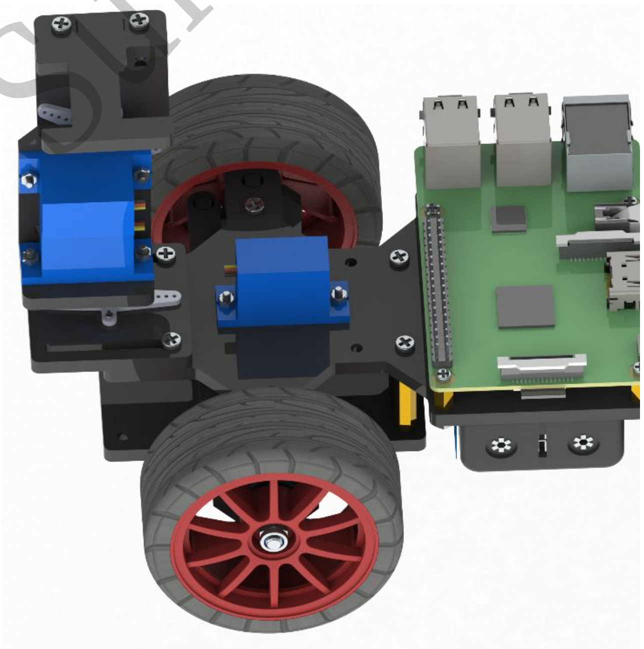
a) Connect the two plate parts together with two M3*10 screws and M3 nuts. The two plates should be perpendicular to each other. 	b) When completed, the assembly should look like the figure below. 
c) Connect the two parts at a right angle with two M3*10 screws and M3 nuts. 	d) When completed, the assembly should look like the figure below. 

I. Servos + Rocker Arm Plates

<table>
<tr>
<td>a) Connect the following parts without any screws.
</td>
<td>b) When completed, the assembly should look like the figure below.
The two plates as indicated with arrows below should be assembled in parallel; otherwise, remove and mount again. DO NOT directly rotate them in case of damages to the servo.
</td>
</tr>
<tr>
<td>◆ Top view:

◆ Front view:
</td>
<td>c) Then fasten them with an M2*4 screw.
The M2*4 screw is contained in the package of the servo and is the shortest of the screws.
</td>
</tr>
</table>

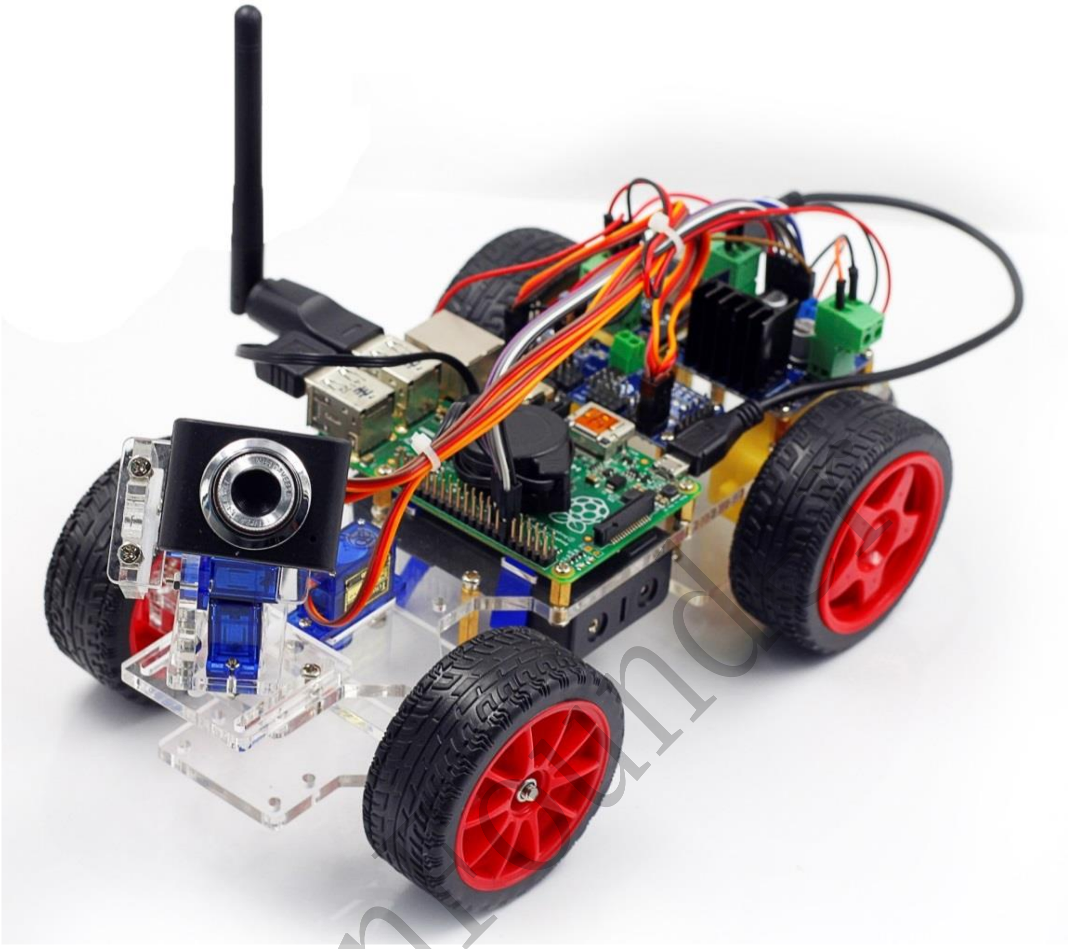
J. Mount + Car

Assemble the mount part and the car with two M3*10 screws and M3 nuts. 
When completed, the assembly should look like the figure below. 

----------------------- Now the car is completed. Congratulations! ---------------------------

The car should be assembled successfully as shown below:

To organize the wires, you can use the cable spiral wrap.



Bottom view:

Car Calibration (Operation on PC)

A. Get the source code

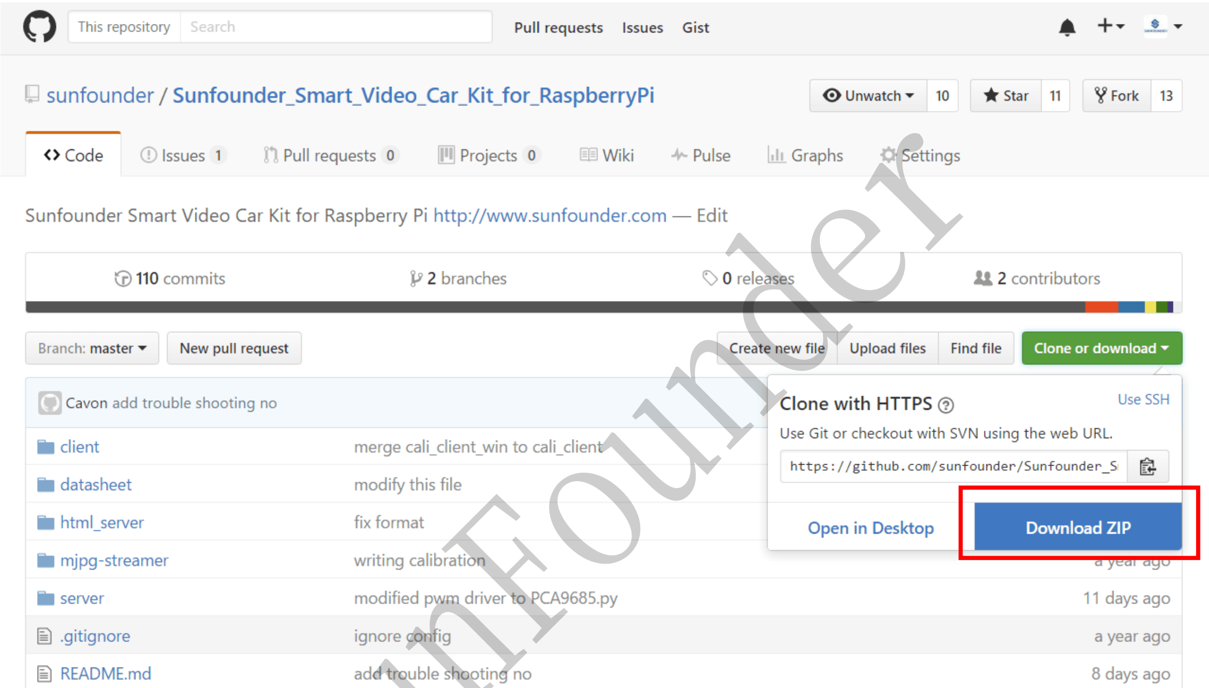
Download the source code directly from Github to your PC at:

https://github.com/sunfounder/Sunfounder_Smart_Video_Car_Kit_for_RaspberryPi,

Or search for Sunfounder in Github and find the repository:

Sunfounder_Smart_Video_Car_Kit_for_RaspberryPi.

Click **Download ZIP** on the sidebar of the page, as shown below.



After download, unzip the file. Find the file whose name containing "–master" and delete its suffix.

The file name then should be: *Sunfounder_Smart_Video_Car_Kit_for_RaspberryPi*.

B. Install Python software

For Linux users:

Install `python-tk` for a remote interface, open a Terminal, and type in the command:

```
sudo install apt-get install python-tk
```

For Windows users:

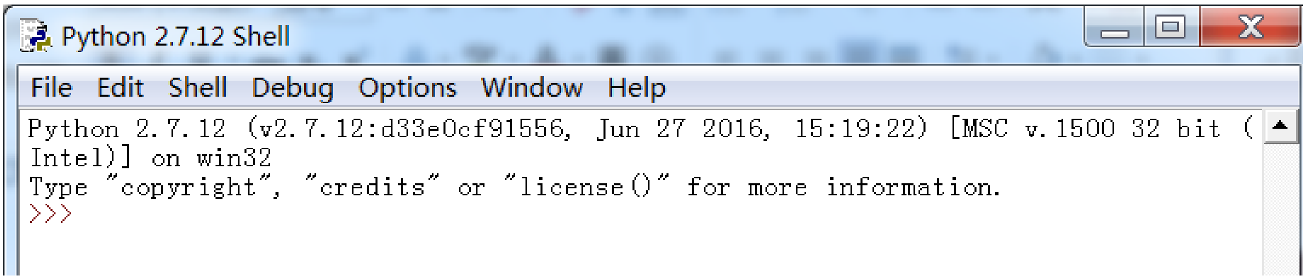
Go to the Python website www.python.org, find the latest Python 2 and install. After installation, RESTART the computer.

C.Run cali_client

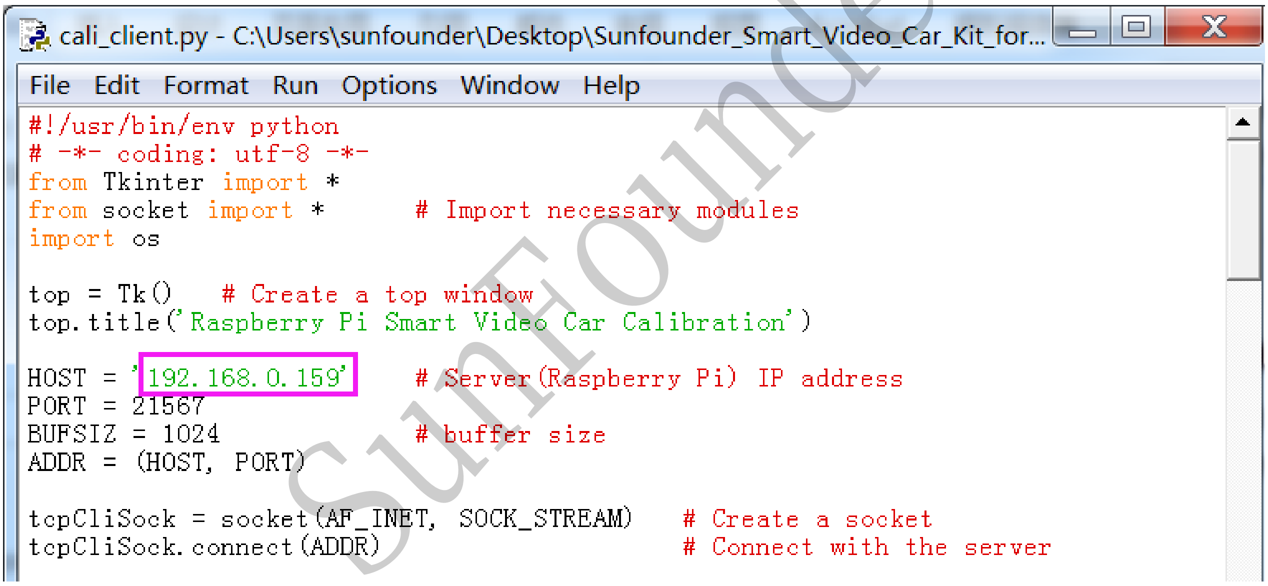
Run the calibration client on PC to connect the server on Raspberry Pi.

For Windows users:

Click the **Start** button on your computer, and type in python in the search bar, and you can find the IDLE (Python GUI). Click it and then a window will pop up.



Click **File** -> **Open** -> **Sunfounder_Smart_Video_Car_Kit_for_RaspberryPi-master** -> **client** -> **cali_client.py** to open this file, modify the value of **HOST** for the IP address of the Raspberry Pi.

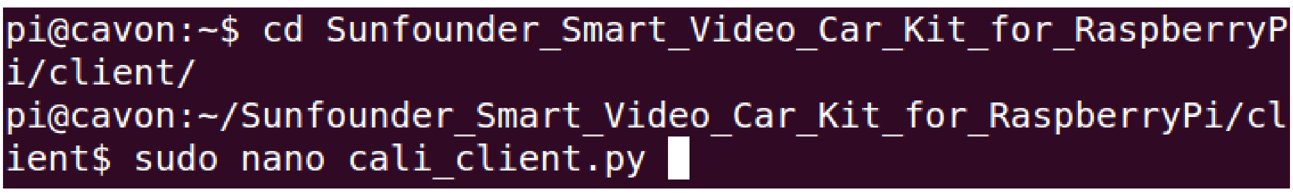


After modification, save the file and click the **Run** menu, and select **Run Module**.

For Linux users:

Open another terminal in Linux (not via ssh on your Pi). Find the sketch downloaded and edit *client/cali_client.py:*

```
cd client/
sudo nano cali_client.py
```



Find the variable of *HOST*:

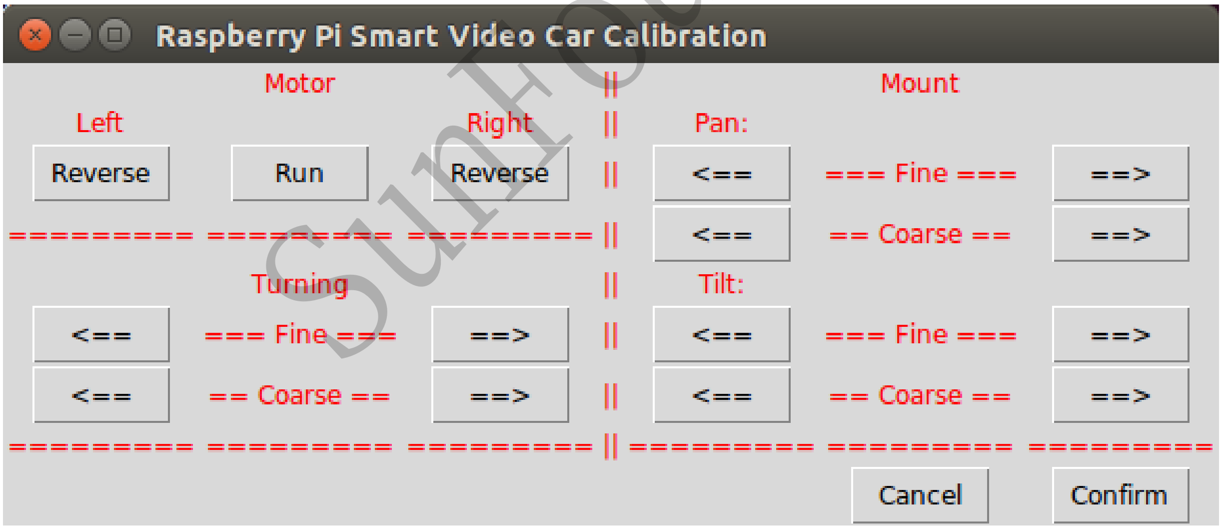
```
  GNU nano 2.2.6        File: cali_client.py

#!/usr/bin/env python
# -*- coding: utf-8 -*-
from Tkinter import *
from socket import *      # Import necessary modules
import os

top = Tk()   # Create a top window
top.title('Raspberry Pi Smart Video Car Calibration')

HOST = '192.168.0.133'    # Server(Raspberry Pi) IP address
PORT = 21567
BUFSIZ = 1024             # buffer size
ADDR = (HOST, PORT)
```

Enter your own address of the Raspberry Pi there. Press **Ctrl+O** to save and **Ctrl+X** exit.

Run *cali_client.py*:

```
sudo python cali_client.py
```

No matter what system your computer is running on, Linux or Windows, when you run *cali_client.py*, a window **Raspberry Pi Smart Video Car Calibration** will pop up:



In the terminal remotely connected with the Raspberry Pi, the IP address of the PC will be printed.

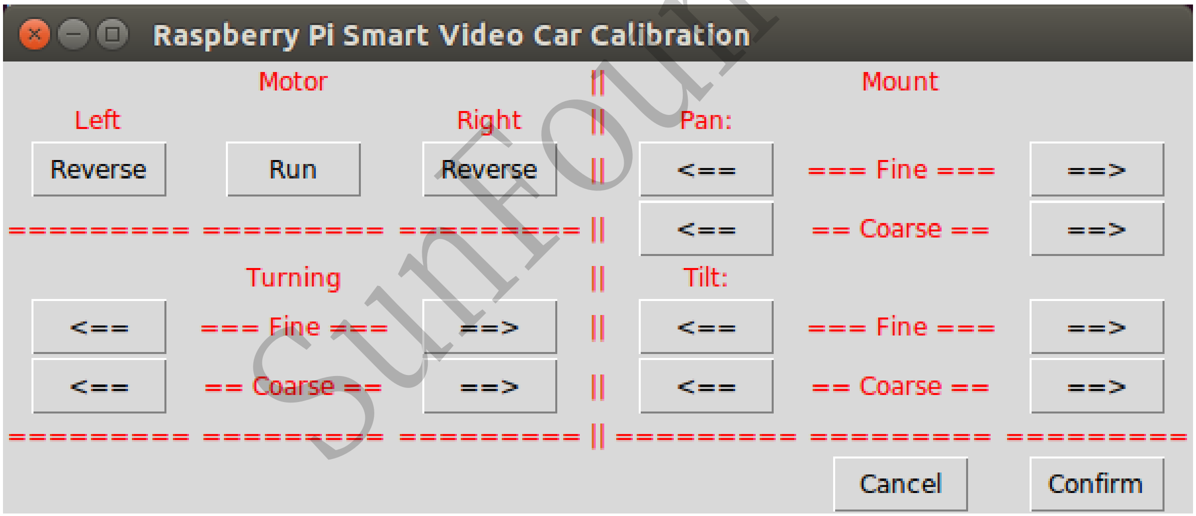
```
pi@raspberrypi ~/Sunfounder_Smart_Video_Car_Kit_for_RaspberryPi/server $ sudo python cali_server.py
offset_x = 0
offset_y = 0
offset = 0
turning0 = True
turning1 = True
Waiting for connection...
...connected from : ('192.168.0.124', 54220)
```

Then you can start calibrating. Before that, take out your package box and place it vertically with the side face to the table. Put the car onside the box and keep it balanced. The purpose is to keep the wheels of the car off the table.

By default, the front wheels should be directly pointed towards the front; the camera on the tilt servo should be face up no matter what directions the pan servo is pointed at.

D. Start Calibration

On the calibration UI there are three sections: **Motor**, **Turning**, and **Mount**.

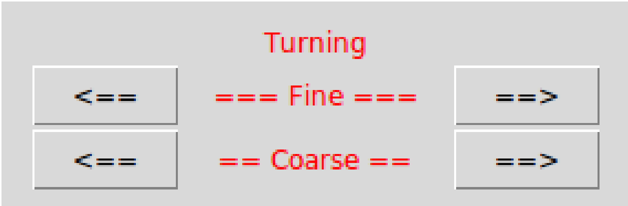


Motor Adjustment

Click **Run.** The car will walk forward. Check whether both the back wheels move forward together. If either fails to do so, your wiring may be wrong. But don't worry! You don't need to rewire; just click the corresponding **Reverse** in Motor section of the Calibration window shown above. After clicking, observe whether the wheel you just adjusted is turning forward.

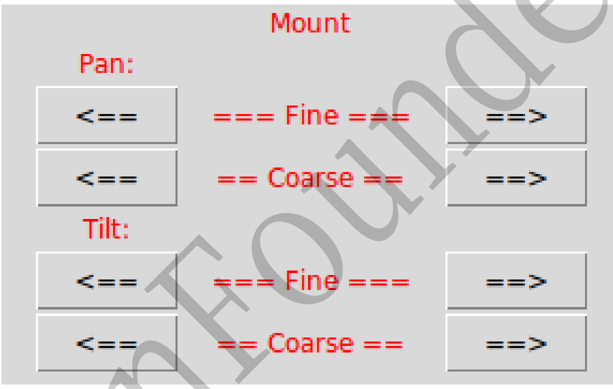
If the reversing works normally, click **Run** again to stop the wheels from spinning.

Turning Adjustment



Currently the front wheels should be pointed at the exact front direction. But if it is not, you need to make some adjustments. In the **Turning** section in the **Calibration** window, click the left (<==) and right (==>) arrow buttons in the upper line to make fine adjustments, and those in the lower to coarsely adjust the turning direction. Keep adjusting until the wheels is oriented to the front exactly. Then you may place the car on the table and click **Run** to verify whether it runs in a straight line. If not, perform the adjustment again till it does.

Mount Adjustment



Now the pan servo on the car should be pointed at the exact front direction and the camera face up. But if not, you may need to adjust them similarly. In the Mount section, there are the adjustments for the pan servo and tilt one. Also you have two kinds of adjustments, fine and coarse. Keep adjusting until they are pointed at the right direction.

After all the adjustments are done, click **Confirm**.

And the program will exit.

Then the Raspberry Pi returns to the status: **Waiting for connection...**

```
Waiting for connection...
```

Press Ctrl + C to exit.

MJPG-streamer (Operation on Raspberry Pi)

A. MJPG-streamer Installation

Introduction

The acquisition and transmission of video data by the SunFounder Smart Video Car is fulfilled based on MJPG-streamer.

MJPG-streamer is a command line application that copies JPG-frame from a single input plugin to multiple output plugins. It can be used to stream JPEG files over an IP-based network from the webcam to a viewer like Firefox, Cambozola and Videolanclient or even to a Windows mobile device running the TCPMP-Player.

It was written for embedded devices with very limited resources in terms of RAM and CPU. Its origin, the "uvc_streamer" was written, because Linux-UVC compatible cameras directly produce JPEG-data, allowing fast and performant M-JPEG streams even from an embedded device running OpenWRT. The input module "input_uvc.so" captures such JPG frames from a connected webcam.

Installation

Plug the USB camera into Raspberry Pi, and run the command lsusb. The GEMBIRD represents the USB camera; since it is printed on the screen, it indicates the system has recognized the camera.

```
lsusb
```

You'll see:

```
pi@raspberrypi:~ $ lsusb
Bus 001 Device 004: ID 1908:2310 GEMBIRD
Bus 001 Device 003: ID 0424:ec00 Standard Microsystems Corp. SMSC9512/9514 Fast Ethernet Adapter
Bus 001 Device 002: ID 0424:9514 Standard Microsystems Corp.
Bus 001 Device 001: ID 1d6b:0002 Linux Foundation 2.0 root hub
```

Check whether the driver for the camera works normally:

```
ls /dev/vid*
```

You'll see:

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

If */dev/video0* is printed, the driver is in the normal state.

Then, install the following software needed:

```
sudo apt-get install subversion
sudo apt-get install libv4l-dev
```

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

Compile the source code of MJPG-streamer:

```
cd /home/pi/Sunfounder_Smart_Video_Car_Kit_for_RaspberryPi/mjpg-streamer/mjpg-streamer
sudo make USE_LIBV4L2=true clean all
```

Install:

```
sudo make DESTDIR=/usr install
```

B. Testing

Operation on Raspberry Pi

Keep Run the mjpg-streamer

Run the program:

```
sudo sh start.sh
```

Then the video data acquisition will start, like this:

```
UVCIOC_CTRL_MAP - Error: Inappropriate ioctl for device
mapping control for Pan Reset
UVCIOC_CTRL_MAP - Error: Inappropriate ioctl for device
mapping control for Tilt Reset
UVCIOC_CTRL_MAP - Error: Inappropriate ioctl for device
mapping control for Pan/tilt Reset
UVCIOC_CTRL_MAP - Error: Inappropriate ioctl for device
mapping control for Focus (absolute)
UVCIOC_CTRL_MAP - Error: Inappropriate ioctl for device
mapping control for LED1 Mode
UVCIOC_CTRL_MAP - Error: Inappropriate ioctl for device
mapping control for LED1 Frequency
UVCIOC_CTRL_MAP - Error: Inappropriate ioctl for device
mapping control for Disable video processing
UVCIOC_CTRL_MAP - Error: Inappropriate ioctl for device
mapping control for Raw bits per pixel
UVCIOC_CTRL_MAP - Error: Inappropriate ioctl for device
 o: www-folder-path...: ./www/
 o: HTTP TCP port.....: 8080
 o: username:password.: disabled
 o: commands..........: enabled
```

Type in the following address (replace *192.168.0.126* with your Raspberry Pi IP address) at the address bar of your browser (Firefox is recommended): (operation on PC)

http://192.168.0.126:8080/stream.html

Press **Enter** and you will see the view captured by the camera displayed on the screen in a real-time manner.


192.168.0.126:8080/stream.html
MJPG-Streamer Demo Pages
a ressource friendly streaming application
Home
Static
Stream
Java
Javascript
VideoLAN
Control
Version info:
v0.1 (Okt 22, 2007)
Display the stream
Hints
This example shows a stream. It works with a few browsers like Firefox for example. To see a simple example click here. You may have to reload this page by pressing F5 one or more times.
Source snippet

© The MJPG-streamer team | Design by Andreas Viklund

Get on the Road!

A. Run TCP server (Operation on Raspberry Pi)

Now a terminal's already open for remote login to the Raspberry Pi and run the *mjpg-streamer*, and you need to keep it RUNNING. Open one more to log into the Raspberry Pi to run *tcp_server*.

Go to the directory with *cd*.

```
cd ~/Sunfounder_Smart_Video_Car_Kit_for_RaspberryPi/server
```

Then run tcp_server.py:

```
sudo python tcp_server.py
```

The server program on the Raspberry Pi will be running and waiting for the client to connect to the Raspberry Pi.

```
pi@raspberrypi ~/Sunfounder_Smart_Video_Car_Kit_for_RaspberryPi/server $ sudo python tcp_server.py
Waiting for connection...
```

B. Run the Client (operation on PC)

Run the client on PC to connect the server on the Raspberry Pi.

For Windows users:

Open the IDLE:

```
Python 2.7.12 Shell
File  Edit  Shell  Debug  Options  Window  Help
Python 2.7.12 (v2.7.12:d33e0cf91556, Jun 27 2016, 15:19:22) [MSC v.1500 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>>
```

Click **File** -> **Open** -> **Sunfounder_Smart_Video_Car_Kit_for_RaspberryPi** -> **client** -> **client App.py** to open this file, modify the value of the HOST for the IP address of the Raspberry Pi.

```
client_App.py - C:\Users\sunfounder\Desktop\Sunfounder_Smart_Video_Car_Kit_fo...
File  Edit  Format  Run  Options  Window  Help
#!/usr/bin/env python
# -*- coding: utf-8 -*-
from Tkinter import *
from socket import *      # Import necessary modules

ctrl_cmd = ['forward', 'backward', 'left', 'right', 'stop', 'read cpu_temp', 'ho

top = Tk()   # Create a top window
top.title('Sunfounder Raspberry Pi Smart Video Car')

HOST = '192.168.0.147'    # Server(Raspberry Pi) IP address
PORT = 21567
BUFSIZ = 1024             # buffer size
ADDR = (HOST, PORT)

tcpCliSock = socket(AF_INET, SOCK_STREAM)   # Create a socket
tcpCliSock.connect(ADDR)                    # Connect with the server
```

After modification, save the file and select **Run** -> **Run Module**.

For Linux users:

Open another terminal again. But DO NOT log into the Raspberry Pi remotely in this terminal. Go to the client with *cd* and edit the file *client.py*:

```
pi@cavon:~$ cd Sunfounder_Smart_Video_Car_Kit_for_RaspberryPi
/client/
pi@cavon:~/Sunfounder_Smart_Video_Car_Kit_for_RaspberryPi/cli
ent$ sudo nano client_App.py
```

Similar to changes in calibration, change the IP address in client program in your PC:

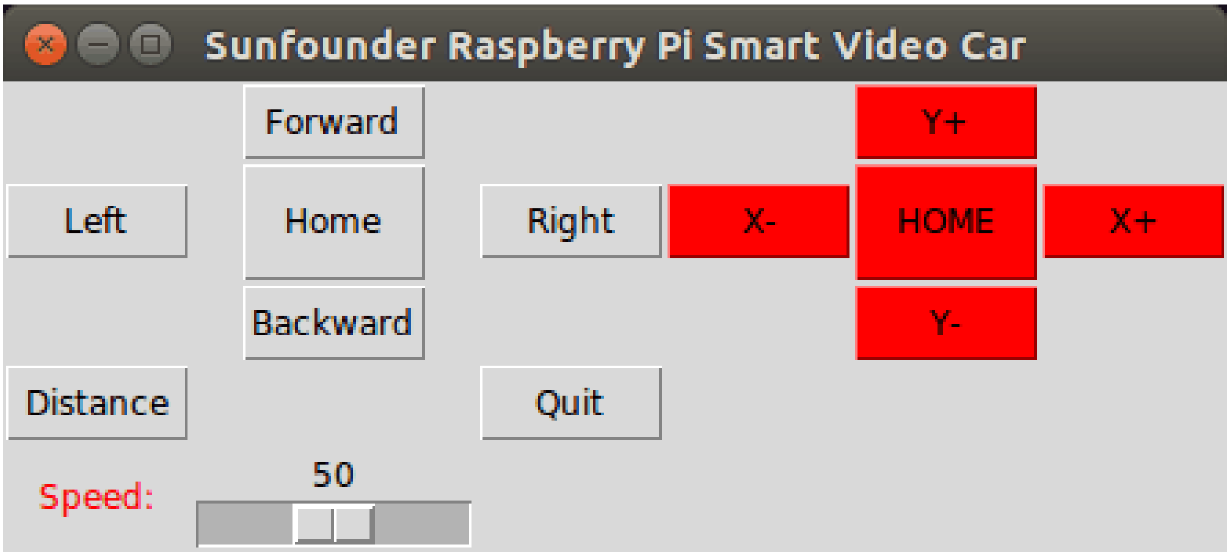
```
  GNU nano 2.2.6        File: client_App.py

#!/usr/bin/env python
# -*- coding: utf-8 -*-
from Tkinter import *
from socket import *       # Import necessary modules

ctrl_cmd = ['forward', 'backward', 'left', 'right', 'stop', $

top = Tk()   # Create a top window
top.title('Sunfounder Raspberry Pi Smart Video Car')

HOST = '192.168.0.133'    # Server(Raspberry Pi) IP address
PORT = 21567
BUFSIZ = 1024             # buffer size
ADDR = (HOST, PORT)
```

After the alteration, press **Ctrl** + **O** and save and **Ctrl** + **X** to exit.

Then run the client program:

```
sudo python client_App.py
```

No matter what system your computer is running on, Linux or Windows, when you run the *client_App.py*, the following window will appear on your screen:



You can click buttons such as **Forward** and **Backward** to control the car moving remotely. Or click **X+**, **X-**, **Y+**, and **Y-** to control the coverage of the camera.

Note:

The server program must be run ***before*** *you run the client program. Some settings must be completed for the server before the service is done. A communication endpoint needs to be created for the server to "listen" to requests from the client. Take the server as a receptionist or an operator of the bus phone in a company. Once the phone and device installation is completed and the receptionist or operator is in place, the service begins.*

Program Analysis and Explanation

Abstract

From the perspective of software, the smart car is of C/S structure. The TCP server program is run on Raspberry Pi to listen to the command from the client and control the car accordingly. The client program is run on the PC and connected with the server through the TCP, which provides the user with a graphical user interface (GUI) to conveniently control the Raspberry Pi remotely. Both the client and server programs are written in Python.

Make sure that the circuit is connected properly. Power the smart car, log in the Raspberry Pi remotely, go to the directory *Sunfounder_Smart_Video_Car_Kit_for_RaspberryPi* and check the files under it.

```
cd ~/Sunfounder_Smart_Video_Car_Kit_for_RaspberryPi
ls
```

```
pi@raspberrypi ~/Sunfounder_Smart_Video_Car_Kit_for_RaspberryPi $ ls
client  datasheet  html_server  i2cHelper.py  mjpg-streamer  README.md  server
```

You can see seven files under the directory: client, datasheet, html_server, i2cHelper.py, mjpg-streamer, a file README.md, and server.

Wherein,

client, the client run on your PC,
datasheet contains some PDF files about the chip (you need to view them on PC),
html_server, the web server run on the Raspberry Pi for Android app client,
i2cHelper, a simple script to help you set up i2c on the Raspberry Pi,
mjpg-streamer, the camera driver to acquire and upload images,
README.md, an introduction file with update information,
server, the server run on the Raspberry Pi for the client on your PC.

Introduction of Socket

The C/S-structure program of the SunFounder Raspberry Pi-based Smart Car is written based on the socket module of the Python language. Socket wraps and applies the TCP/IP and is used to describe IP address and port. Also it is a network data structure for computer. The socket module should be created before the communication of network applications. If the socket can be said to be the plug of a telephone, which is the lowest layer of communication, then the combination of IP address and ports can be said to be that of area code and phone numbers. Only having the hardware for a phone call making is not enough. You still need to know whom and where to call. An Internet address is composed of the essential IP address and port for network communication.

1. Server

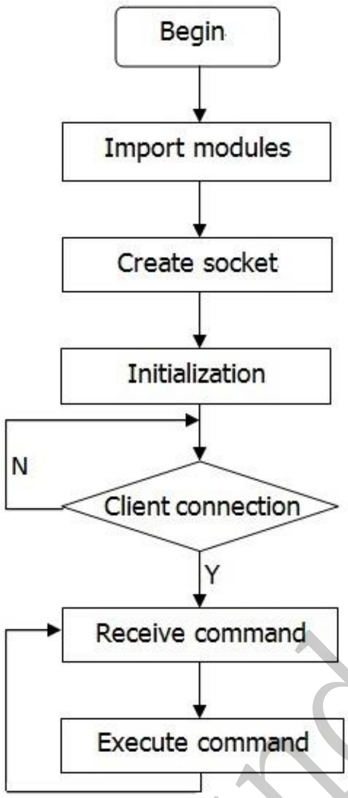
Here we provide a pseudocode which creates a universal TCP server for explanation. Note that this is just one of the methods for server design. After you have a good knowledge about it, you can alter the pseudocode as you want:

```
s = socket( )                 # Create a socket for the server.
s.bind( )                    # Bind the address to the socket.
s.listen( )                   # Listen to the connection.
inf_loop:                    # Indefinite loop of the server.
    c = s.accept( )           # Accept the connection from the client.
comm_loop:                 # Communication loop.
    c.recv( )/c.send( )        # Dialog (receiving or sending data)
c.close( )                   # Close the socket of the client.
s.close( )                   # Close the socket of the server (optional).
```

All kinds of socket can be created via the function *socket.socket( )* and then bound with IP address and port by the function *bind( )*. Since TCP is a connection-oriented communication system, some settings need to be completed before the TCP server starts operation. The TCP server must "listen" to connections from the client.

After the settings are done, the server will enter an indefinite loop. A simple, like single-thread, server will call the function *accept( )* to wait for the coming connection. By default, the function *accept( )* is a blocking one, which means it is suspended before the connection comes. Once a connection is received, the function *accept( )* returns a separate client socket for the subsequent communication. After the temporary socket is created, communication begins. Both the server and client use the new socket for data sending and receiving. The communication does not end until either end closes the connection or sends a null character string.

Process Diagram of Server Program



2. Client

It is easier to create a TCP client than to do a server. Take the following pseudocode:

```
c = socket( )                  # Create a client socket.
c.connect( )                  # Try to connect a server.
comm_loop:                 # Communication loop.
    c.send( )/c.recv( )       # Dialog (sending out and receiving data)
c.close( )                   # Close the client socket.
```

As mentioned above, all sockets are created via the function *socket.socket( )*. Then, the function *connect( )* can be called to connect the server. After the connection is built, the dialog between the client and the server is enabled. When the dialog ends, the client can close the socket and the connection.

Introduction of Tkinter

Developed based on Tkinter, our client program carries graphical interfaces. Tkinter is a GUI widget set for Python. We can develop application programs with graphical interfaces fast by Python language based on it. It is quite easy to use Tkinter. All you have to do is to import the module into Python.

To create and run a GUI program, take the following steps:

a) Import the Tkinter module (by *import Tkinter* or *from Tkinter import* *).

b) Create a top window object to contain the whole GUI program.

c) Create the GUI module needed on the object and enable the functions.

d) Connect the GUI modules with the code at the system back-end.

e) Enter the main event loop.

Take a simple GUI program:

Create a file *Tk_test.py* under the path */home*:

```
touch Tk_test.py
```

Add executable privilege to the file:

```
chmod +x Tk_test.py
```

Open the file:

```
vim Tk_test.py
```

Type in the following code:

```
#!/usr/bin/env python
from Tkinter import *

top = Tk()        # Create a top window
top.title('Sunfounder.com')

label = Label(top, text='Hello Geeks !', fg='blue')  # Create a label and set its foreground color as blue
label.pack()     # layout

top.mainloop()  # main loop
```

Save the code and exit.

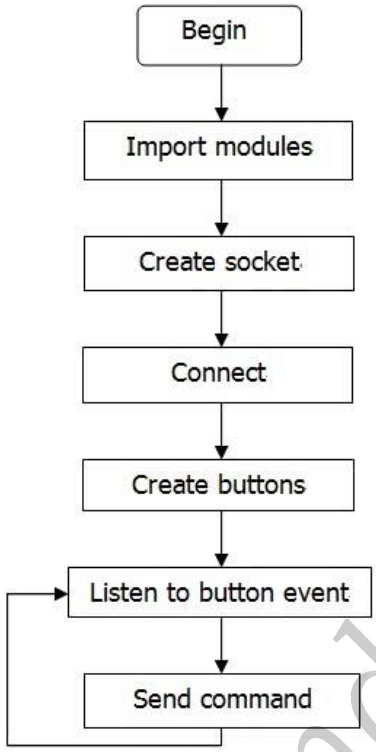
Run

```
./Tk_test.py
```

Then the following picture will appear on your screen:

Click to close the program.

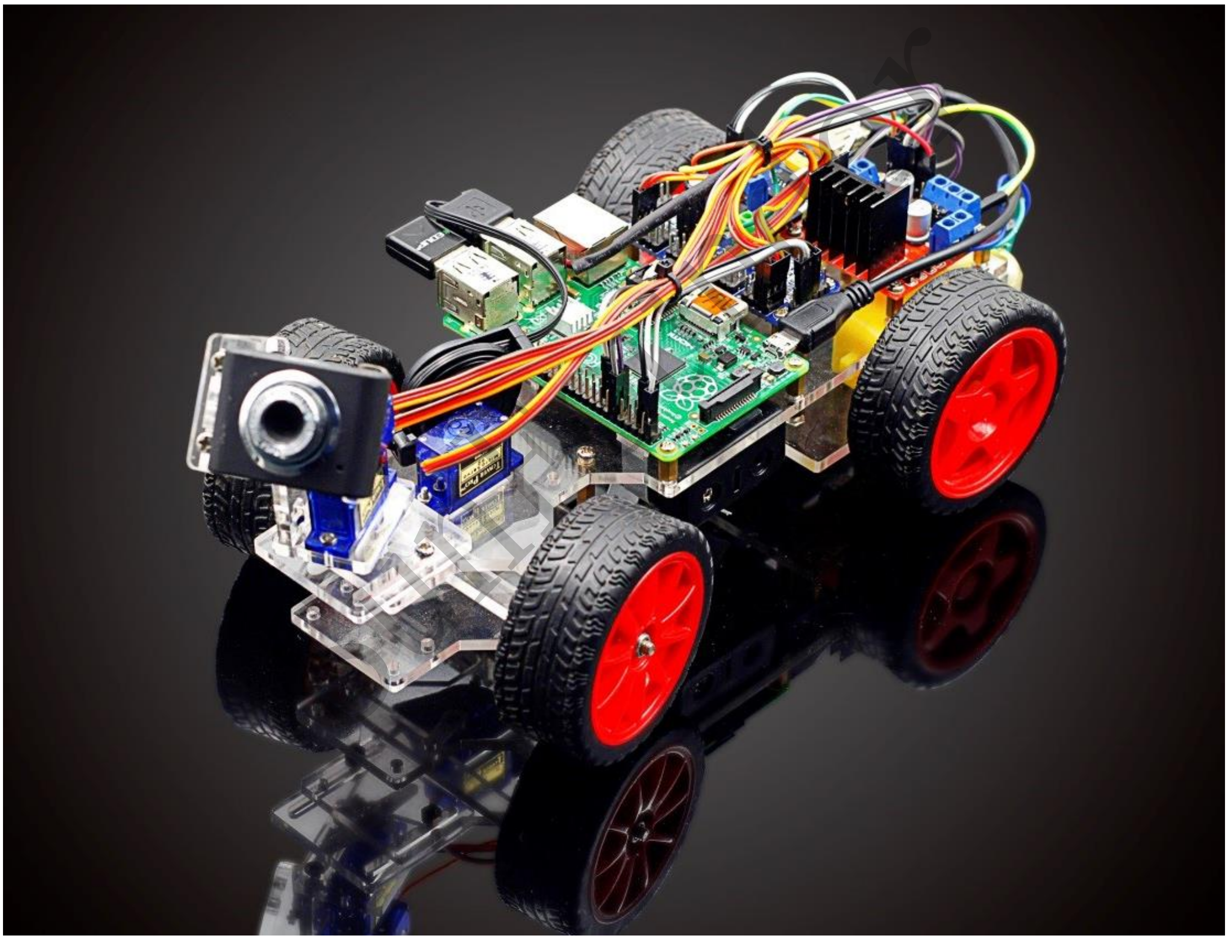
Process Diagram of Client Program



Summary

In this manual, having learned the related components for building the car kit, you've gone through the assembly of the mechanical parts and electrical modules with the knowledge of Raspberry Pi as well as a brief introduction of the key parts like servo, Wi-Fi adapter, etc. Also you've got a lot of software and coding, which lays a solid foundation for your future journey of exploring open-source field.

The **SunFounder Smart Video Car for Raspberry Pi** is not only a toy, but more a meaningful development kit for Raspberry Pi. After all the study and hands-on practice of the kit, you should have a better understanding of Raspberry Pi. Now, get started to make better work!



Copyright Notice