

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 the PiCar-V

As an upgraded version, PiCar-V (Smart Video Car V2.0 for Raspberry Pi) is also suitable for the Raspberry Pi model B+, 2 model B 3 module B and 3 module B+, as the V1.0.

The car PiCar-V is equipped with a wide-angle USB webcam, three whole new circuit boards with less but more simple wiring, some acrylic plates with an adjusted structure, and new code suitable for almost all platforms to control the car.

In this book, we will show you how to build the PiCar-V via description, illustrations of physical components, in both hardware and software respects. You will enjoy learning how all this work. 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_PiCar-V.

You are welcome to pull requests and issue posts on our page on Github.

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.

Contents

Components List	1
Constructure Plates.....	1
Servo Accessories.....	1
Mechanical Fasteners.....	2
Wires	4
PCB	5
Other Components.....	6
Tools.....	8
Introduction.....	10
Visual Programming with Dragit	11
Building the Car	12
Fixing Rear Wheels.....	12
Upper Plate	13
Battery Holder.....	14
Rear Wheels (Screws)	15
PCB Assembly	16
Rear Wheels (Driving)	17
Circuits Building	18
Servo Configuration.....	20
A. Burn the image.....	20
B. Car Power Supply	21
C. Log into Raspberry Pi	21
D. Get Source Code	25
E. Go to the Code Directory	25
F. Install the Environment via the Script	25
G. Configure the Servo to 90 degrees	26
Continue to Assemble.....	27
Front Wheels.....	27
Steering Part.....	28
Front Half Chassis	30
Pan-and-Tilt.....	32
Other Components.....	35
Installing the Client (Operation on the PC)	36
Getting on the Road!	38
1. Run the Server(Operation on Raspberry Pi).....	38
2. Run the Client (Operation on PC)	39
Components.....	49
Robot HATS	49
PCA9865.....	50
TB6612	51
USB Webcam	52
SunFounder SF006C Servo	52
WiFi Adapter.....	53
DC Gear Motor	53

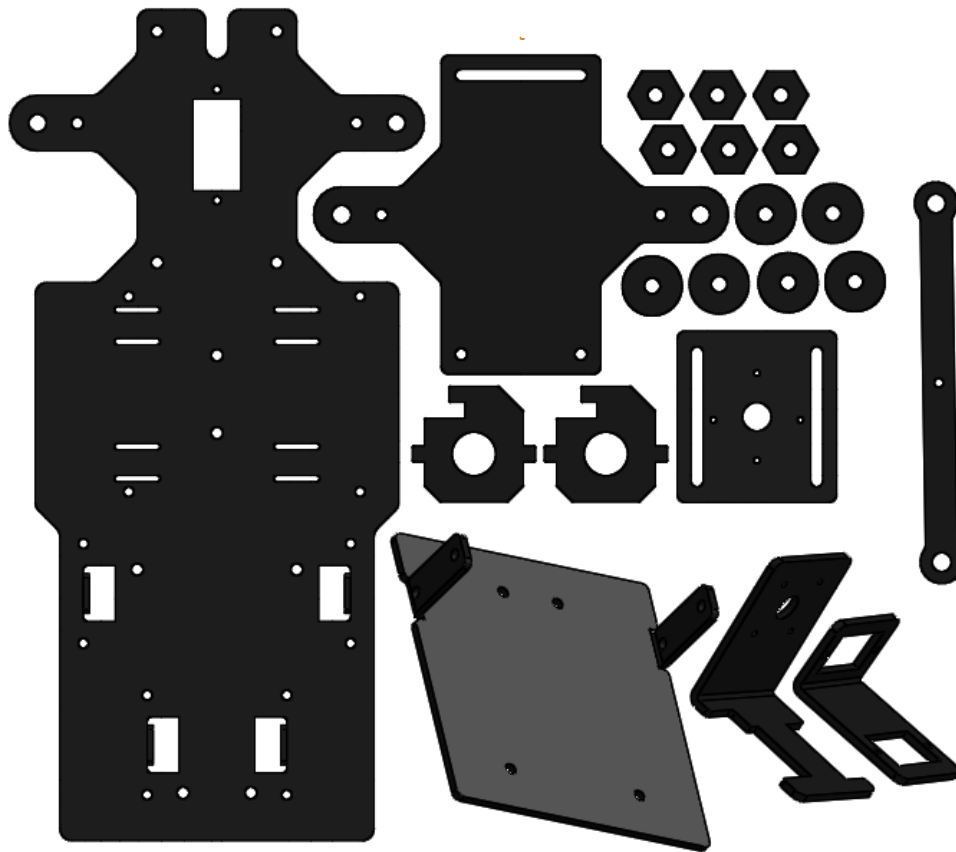
Code Explanation	54
Server Code	55
Client Code.....	57
1. Designing the Interface.....	58
2. Programing the Functions.....	61
Advanced.....	62
Appendix: Function of the Server Installation Scripts.....	63

Copyright Notice

All contents including but not limited to texts, images, and code in this manual are owned by the SunFounder Company. You should only use it for personal study, investigation, enjoyment, or other non-commercial or nonprofit purposes, under the related regulations and copyrights laws, without infringing the legal rights of the author and relevant right holders. For any individual or organization that uses these for commercial profit without permission, the Company reserves the right to take legal action.

Components List

Constructure Plates



1. Upper Plate x 1
2. Front Half Chassis x 1
3. Back Half Chassis x 1
4. Servo-Camera Base x 1
5. Servo Mount x 1
6. Camera Mount x 1
7. Steering Connector x 2
8. Bearing Shield x 6
9. Steering Linkage x 1
10. Hex Front Wheel Fixing Plate x 6





Servo Accessories

We will use the following four parts:

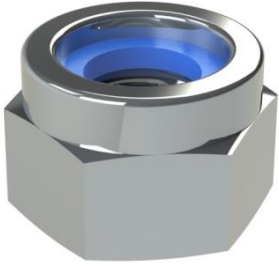



1. 1-arm Rocker Arm
2. 2-arm Rocker Arm
3. 4-arm Rocker Arm
4. Rocker Arm Screw
5. Rocker Arm Fixing Screw × 2




Mechanical Fasteners


Name	Component	Qty.
M1.6x4 Cross Self-tapping Screw		8
M2x8 Cross Screw		6
M2.5x6 Cross Screw		4
M2.5x12 Cross Screw		8

M3x8 Cross Screw		8
M3x8 Countersunk Cross Screw		2
M3x10 Cross Screw		7
M3x25 Cross Screw		4
M4x25 Cross Screw		2
M2 Nut		6
M2.5 Nut		12
M3 Nut		21

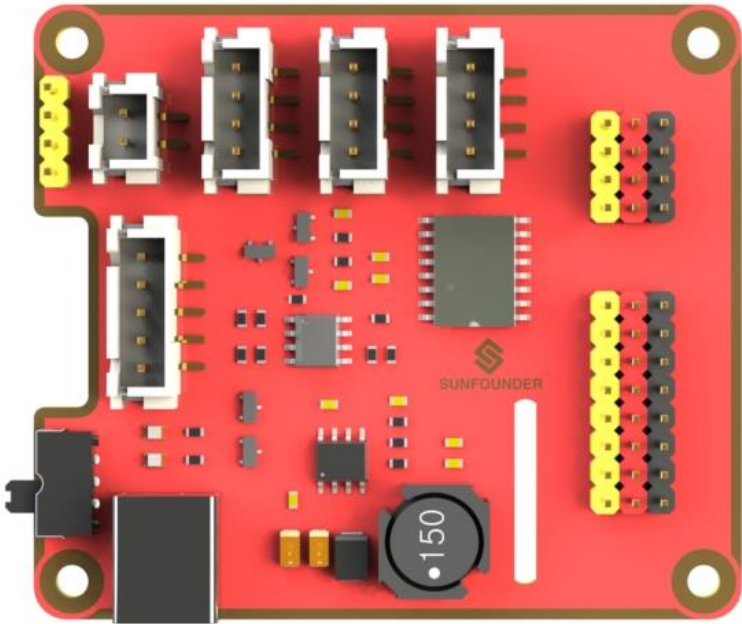
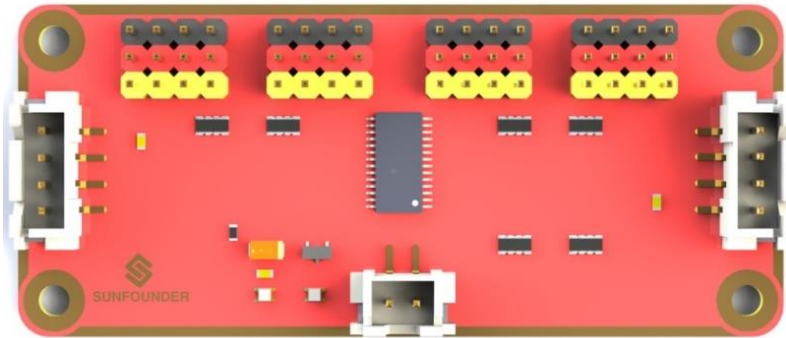
M4 Self-locking Nut		2
M2.5x8 Copper Standoff		8
M3x25 Copper Standoff		8
4x11x4 F694ZZ Flange Bearing		2

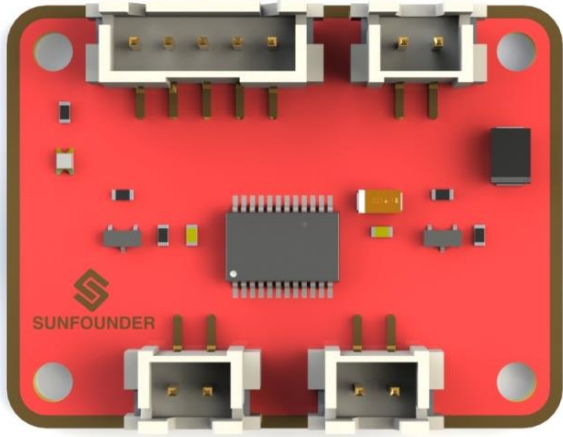
Wires

100mm HX2.54 5-Pin Jumper Wire		1
50mm HX-2.54 4-Pin Jumper Wire		1
50mm HX-2.54 2-Pin Jumper Wire		1

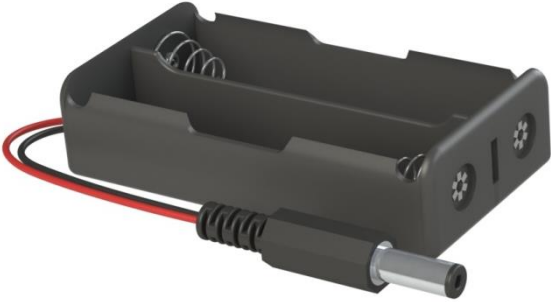
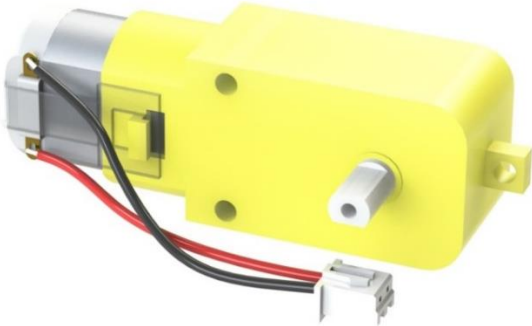
100mm HX-2.54 2-Pin Jumper Wire		1
------------------------------------	--	---




PCB




Robot HATS		1
PCA9685 PWM Driver		1

<p>TB6612 Motor Driver</p>		<p>1</p>
----------------------------	--	----------


Other Components




<p>2x18650 Battery Holder</p>		<p>1</p>
<p>DC Gear Motor</p>		<p>2</p>

<p>SunFounder SF006C Servo</p>		<p>3</p>
<p>120° Wide-angle USB Camera</p>		<p>1</p>
<p>Rear Wheel</p>		<p>2</p>

Front Wheel		2
USB Wi-Fi Adapter		1
Ribbon (30cm)		1

Tools

Cross Screwdriver		1
-------------------	--	---

Cross Socket Wrench		1
M2.5/M4 Small Wrench		1
M2/M3 Small Wrench		1

Introduction

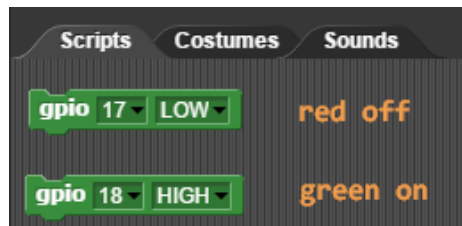
The **PiCar-V** is developed based on the previous Smart Video Car Kit. It's equipped with the new PCA9685 PWM driver, smaller and better-performing DC motor driver TB6612, a Robot Hat with power and interface integrated, a wider-angle camera with clearer vision, some better wheels and tires, and so on. Later we'll check more details.

The basic functions of this new version still stay the same, including controlling the PiCar-V remotely and streaming the video data it captures back. Then what's new? Let's unveil the mysteries!



Visual Programming with Dragit

Instead of coding the Sensor Kit directly in Python, you can also do it in a graphical/visual programming software - Dragit, developed based on Snap! (check <http://snap.berkeley.edu/> for more). What's good about Dragit is that, the programming process is visible with graphics. So you can just drag and drop the blocks (unit in the software) to make the code in each lesson, no need of the knowledge of any programming languages or syntaxes.



You can go to our official website www.sunfounder.com to download **How to use PiCar-V with the Dragit.zip** by clicking [LEARN](#) -> [Get Tutorials](#) -> [PiCar-V](#)

<https://www.sunfounder.com/learn/category/Smart-Video-Car-V2-0-for-Raspberry-Pi-PiCar-V.html>

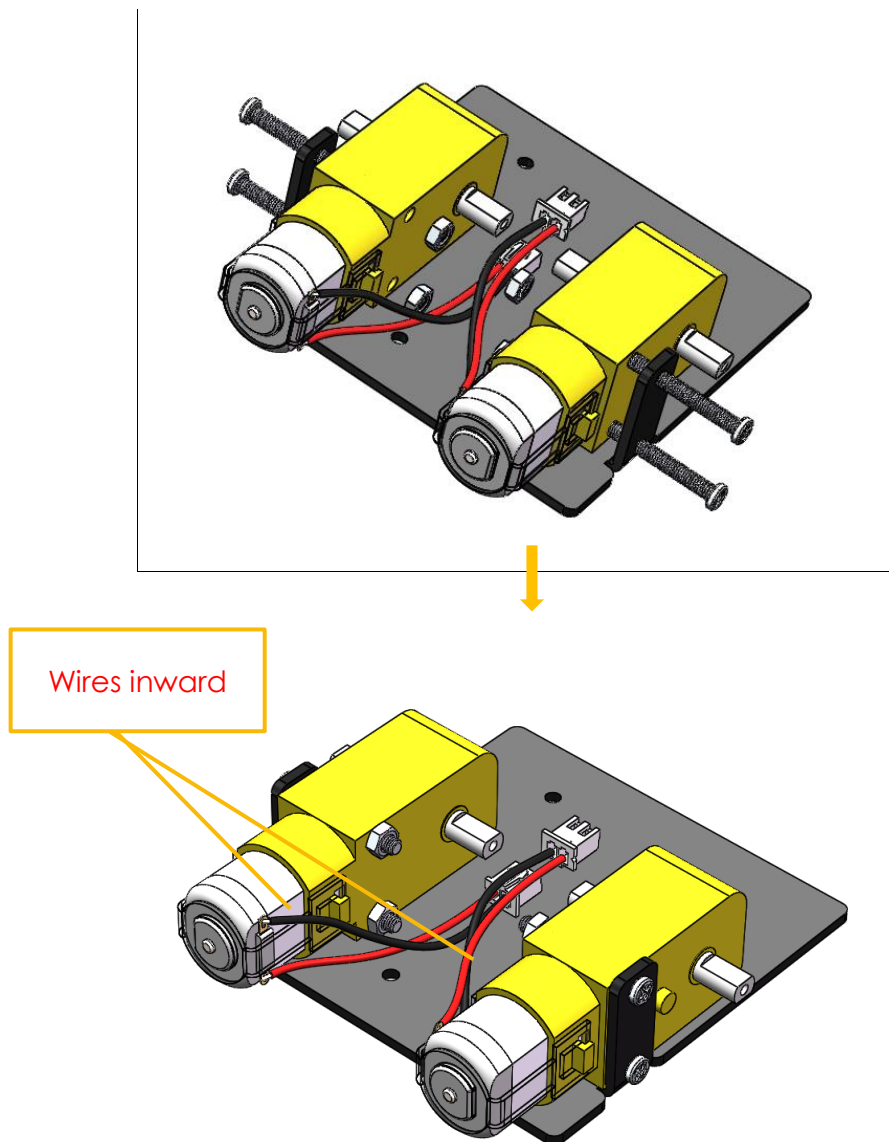
A screenshot of the SunFounder website. The header includes the SunFounder logo and navigation links: SHOP, LEARN, FORUM, BLOG, VIDEO, and WIKI. Below the header is a dark blue bar with 'All Categories' and a search bar. The main content area shows a breadcrumb trail: 'Get tutorials > Robot Kit > PiCar-V'. Below this, there is a section titled 'SunFounder:' with a message: 'If you have any questions, please send an email to support@sunfounder.com. For more tutorials, back to the Learn section. To share yc'. Under the 'Download:' label, there is a link with a zip file icon and the text 'How_to_Use_PiCar-V_with_the_Dragit.zip', which is highlighted with a pink box. Below this, under the 'Pdf:' label, there is a link with a pdf icon and the text 'Smart_Video_Car_V2.0_for_Raspberry_Pi_PiCar-V_.pdf'.

Building the Car

Are you excited when you open the box and see so many components? Keep your patience and take it easy. Please note that some details in the following steps need careful observation. You should double-check your work based on the figures in the manual after finishing each step. Don't worry! Kindly reminders will be given in some particular steps. Just follow the tutorials step by step. Okay, guys, let's start!

Fixing Rear Wheels

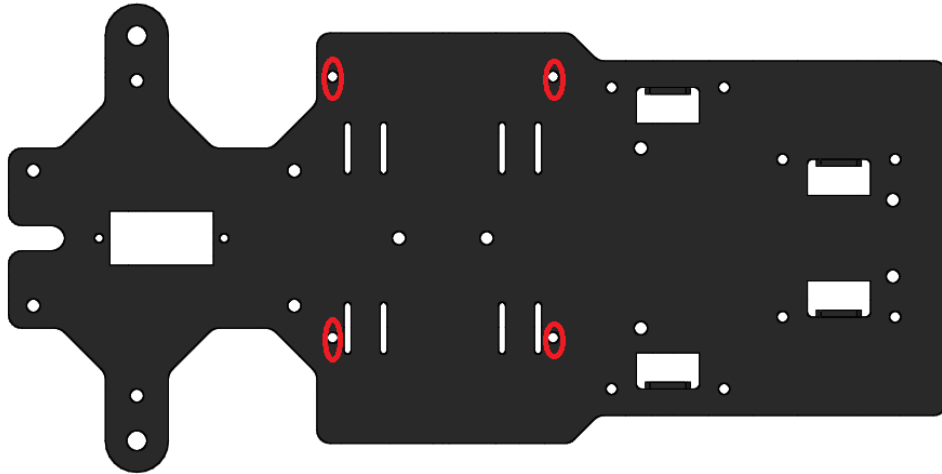
Assemble the two motors with four **M3x25 screws** and **M3 nuts**. Pay attention to place the motors with wires inward, providing convenience for connecting the circuit.



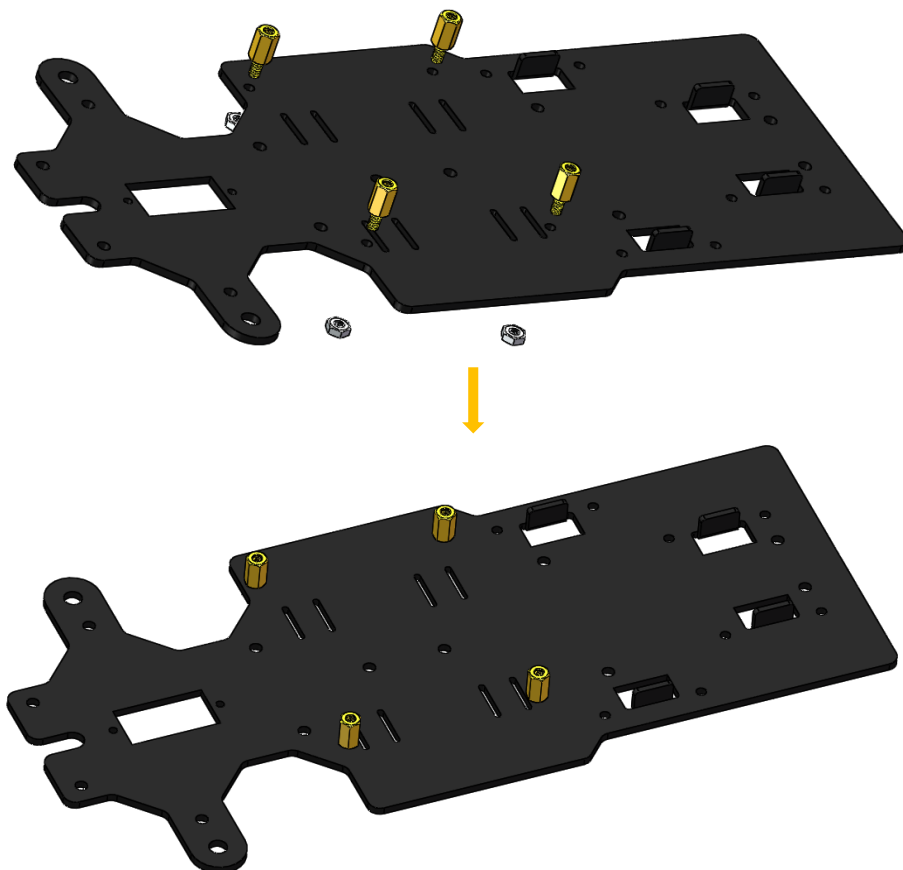
So now this part is completed. You can put it aside for now.

Upper Plate

Mount the **M2.5x8 copper standoffs** and **M2.5 nuts** into the **upper plate** first. Four copper standoffs are needed. So here 4 holes should be used, marked with cross as shown below:

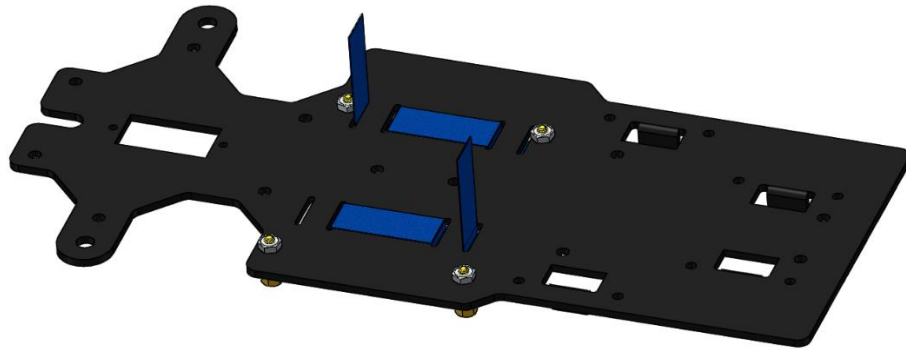


Assemble the **M2.5x8 copper standoffs** and **M2.5 nuts** as shown below. Pay attention that the side the protruding prop should face up.

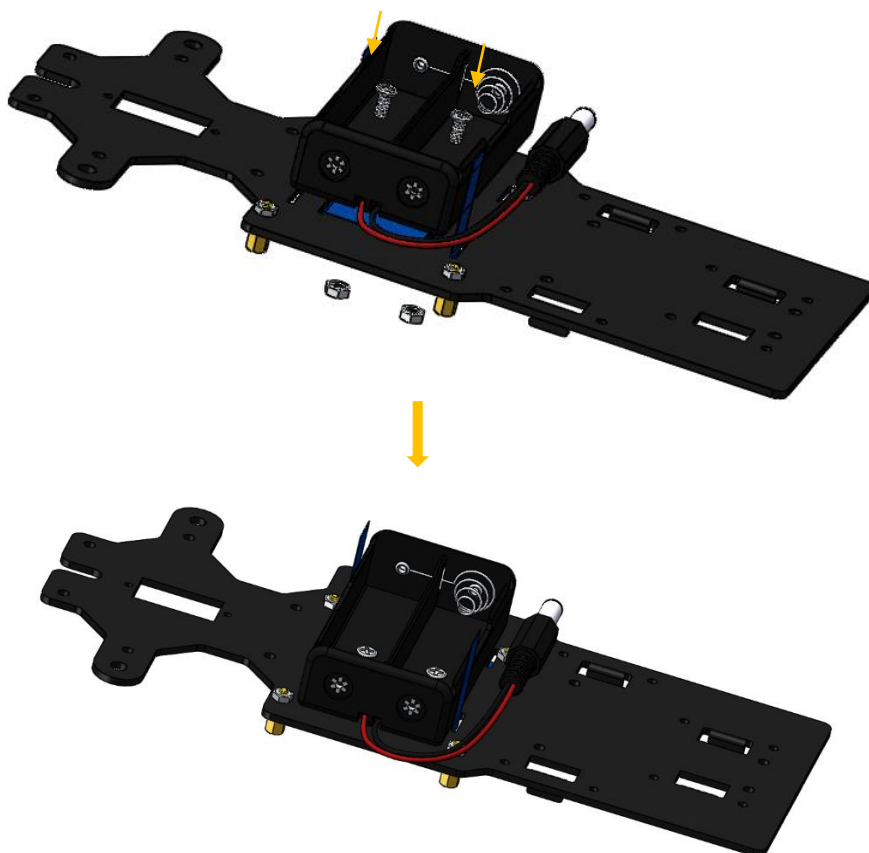


Battery Holder

Turn the Upper Plate upside down. Cut the **ribbon** into two halves. Thread them through the holes on the plate. Pay attention to the direction and leave one end longer out of the plate for each to remove the battery easily later.

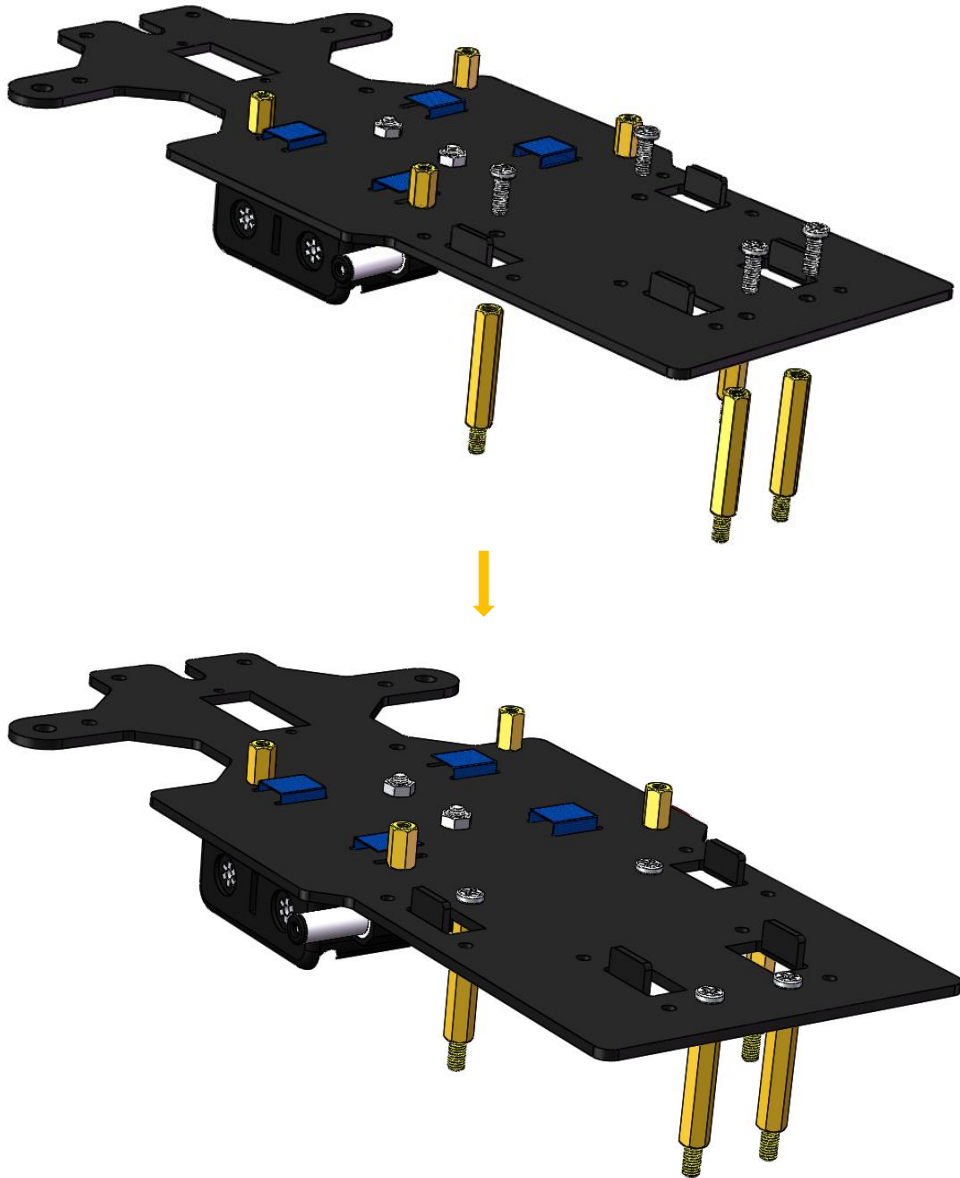


Fasten the battery holder with two **M3x8 countersunk screws** and **M3 nuts**: pay attention to the direction of battery holder's wire.



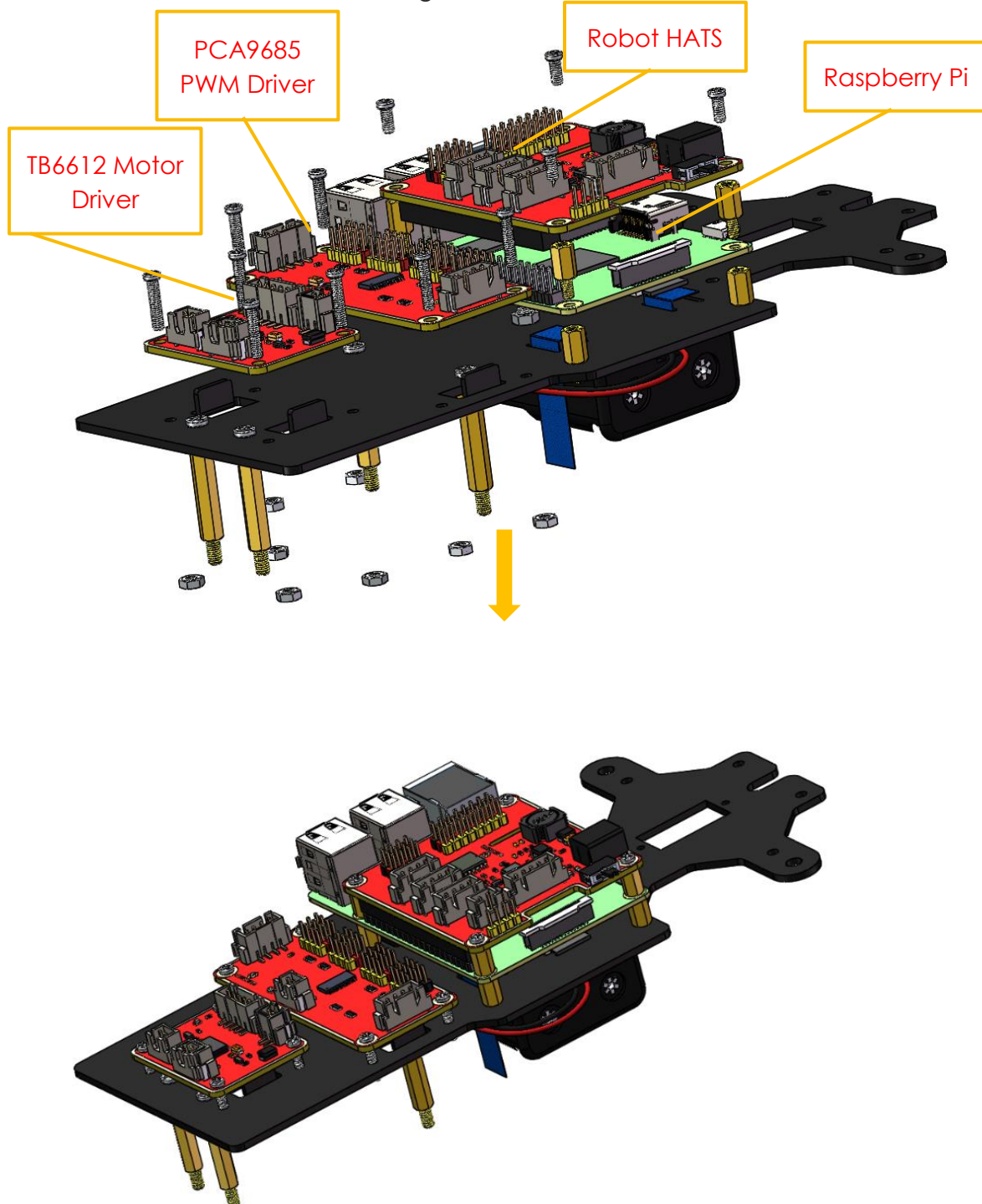
Rear Wheels (Screws)

Insert four **M3x8 screws** with four **M3x25 copper standoffs**:

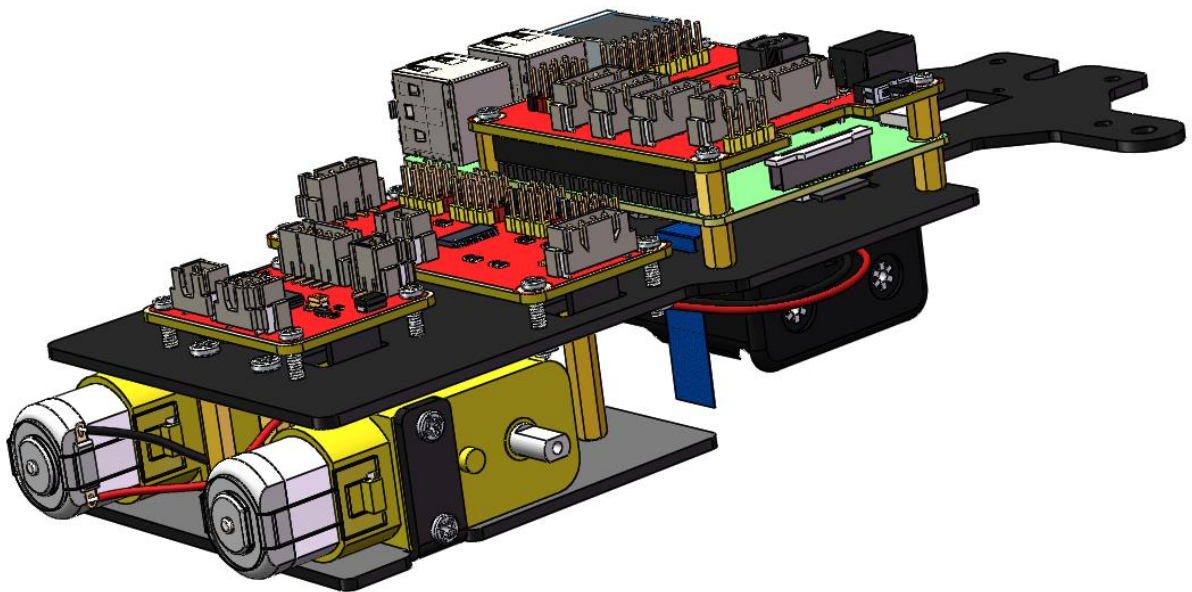
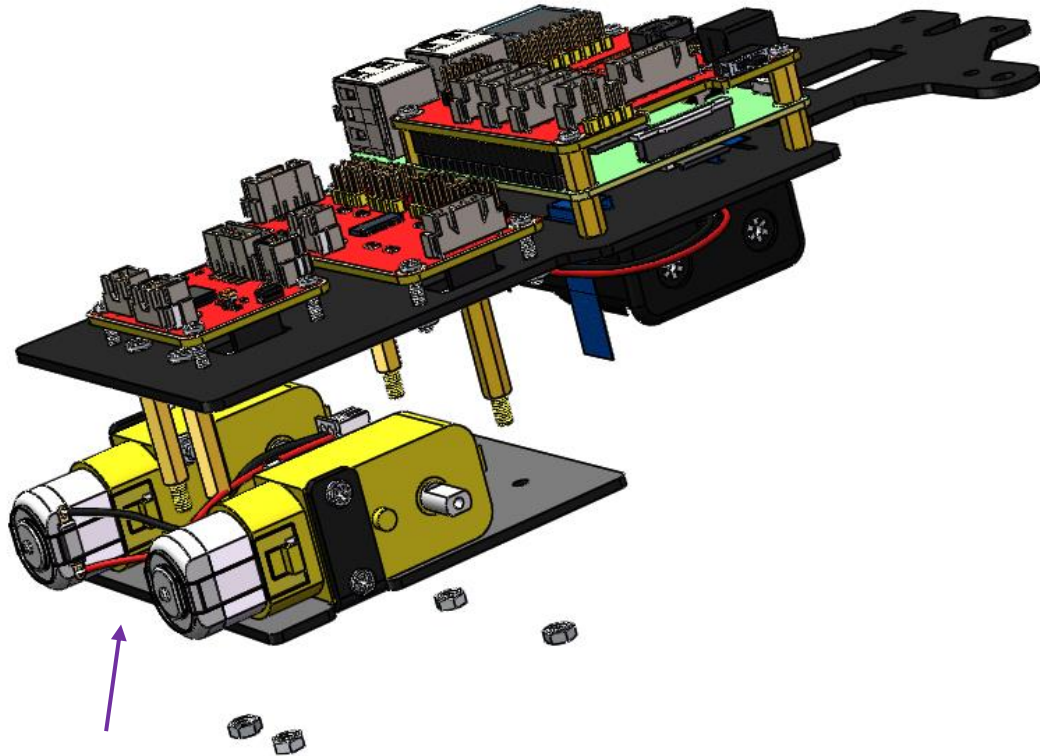


PCB Assembly

Assemble the **Raspberry Pi** (TF Card inserted) with eight **M2.5x8 copper standoffs**, then plug the **Robot HATS** onto it, and fix the **Robot HATS** with four **M2.5x6 screws**. Fix The **PCA9685 PWM Driver**, the **TB6612 Motor Driver** with eight **M2.5x12 screws** and **M2.5 nuts** into the **down plate**:

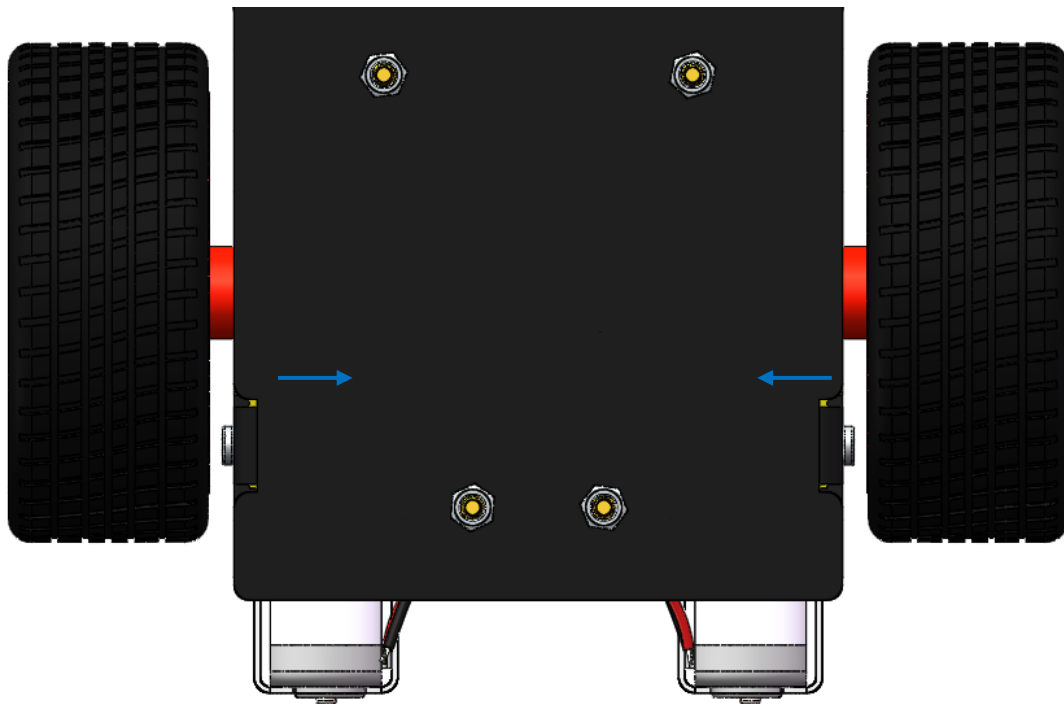


Rear Wheels (Driving)

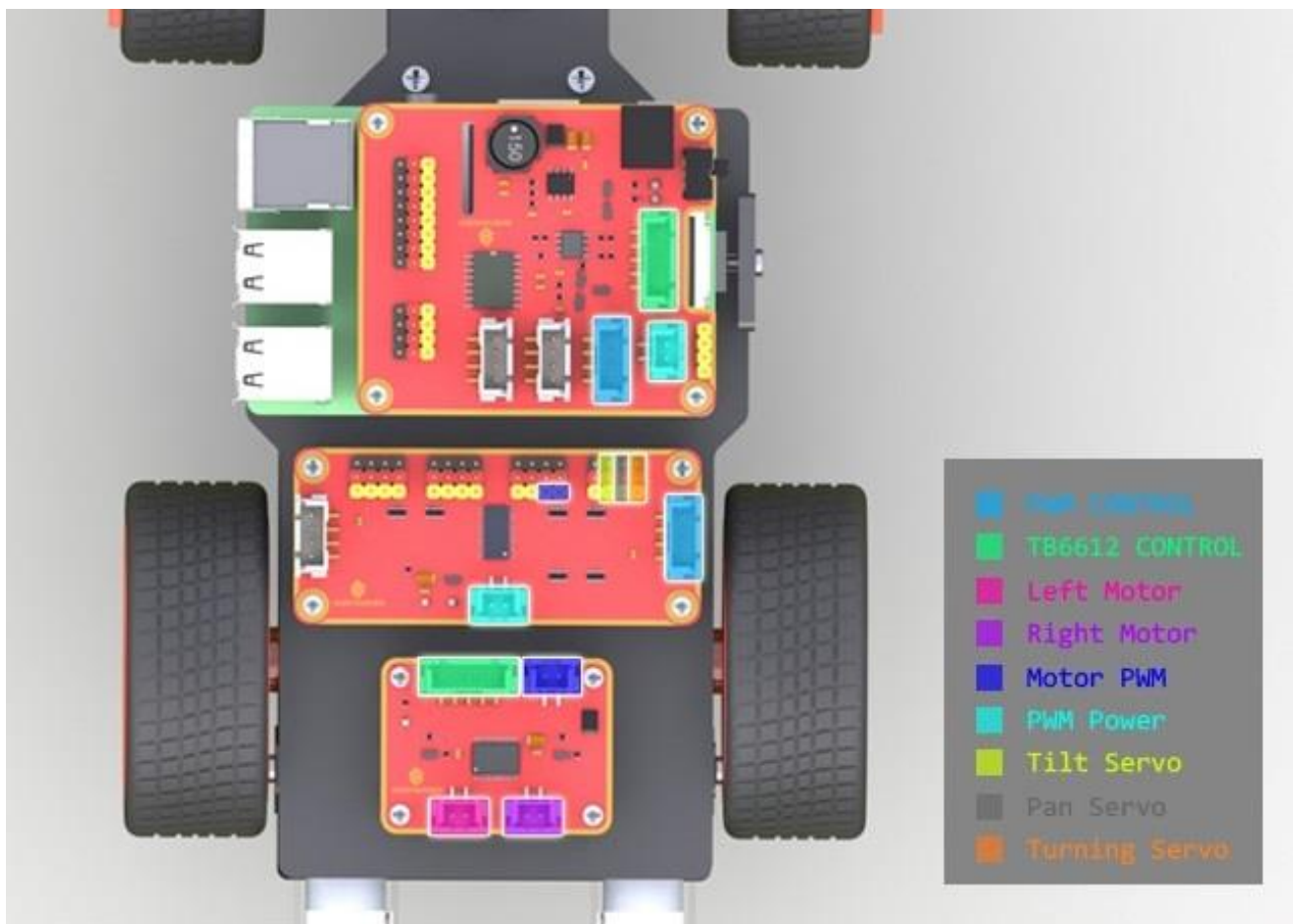


Assemble the **rear wheels**

Align the **rear wheels** with the motor shaft, and rotate to insert them gently.



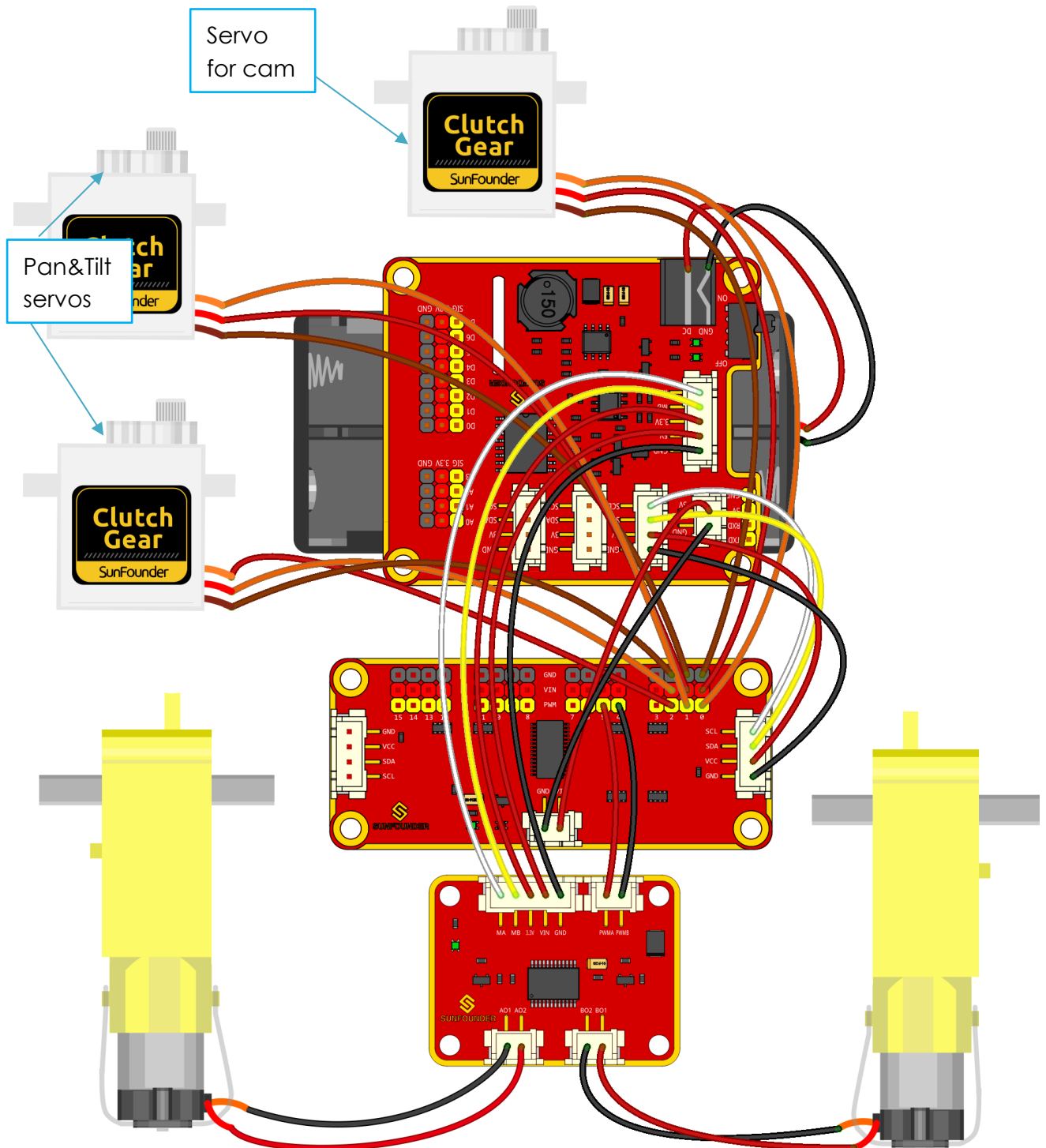
Circuits Building



Connect the **PWM CONTROL** of the Robot HATS with the PCA9685 PWM Driver, and the **TB6612 CONTROL** with the TB6612 Motor Driver.

Then the **Left Motor** and the **Right Motor** of the TB6612 to two motors, and the **Motor PWM** with the No. 4 and 5 PWM channel of the PCA9685.

Connect the **PWM Power** of the PCA9685 with the Robot HATS, and its channel 0, 1, and 2 of with the **Turning Servo**, the **Pan Servo** and the **Tilt Servo** respectively.



fritzing

Servo Configuration

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.

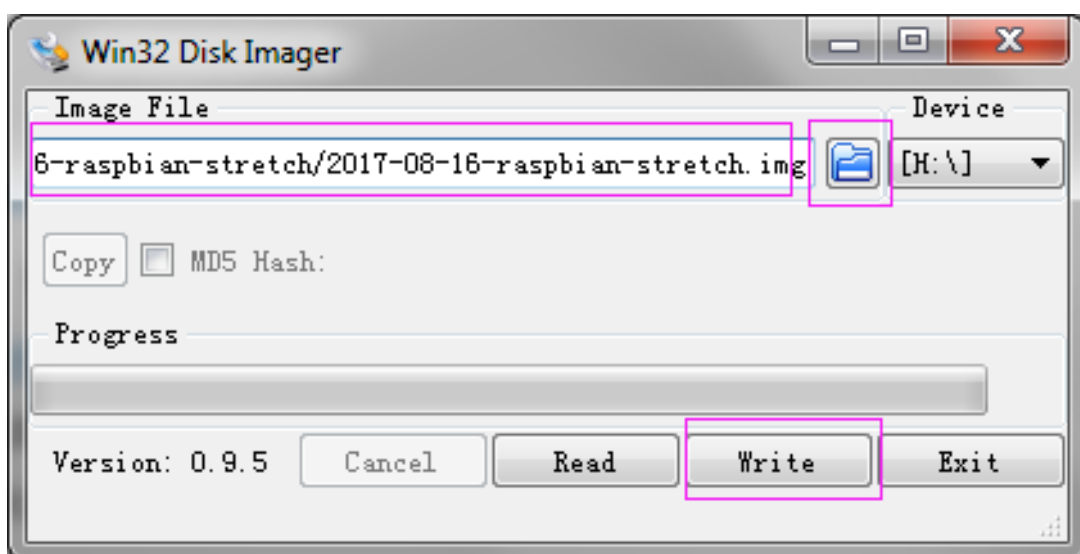
PLEASE DO FOLLOW THIS STEP BEFORE MOVING ON!

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.

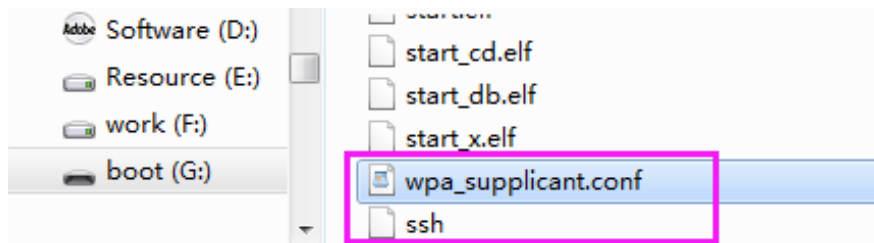
- 7) Create a WiFi configuration file **wpa_supplicant.conf** under **/boot** and add the following to the file:

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

network={
    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

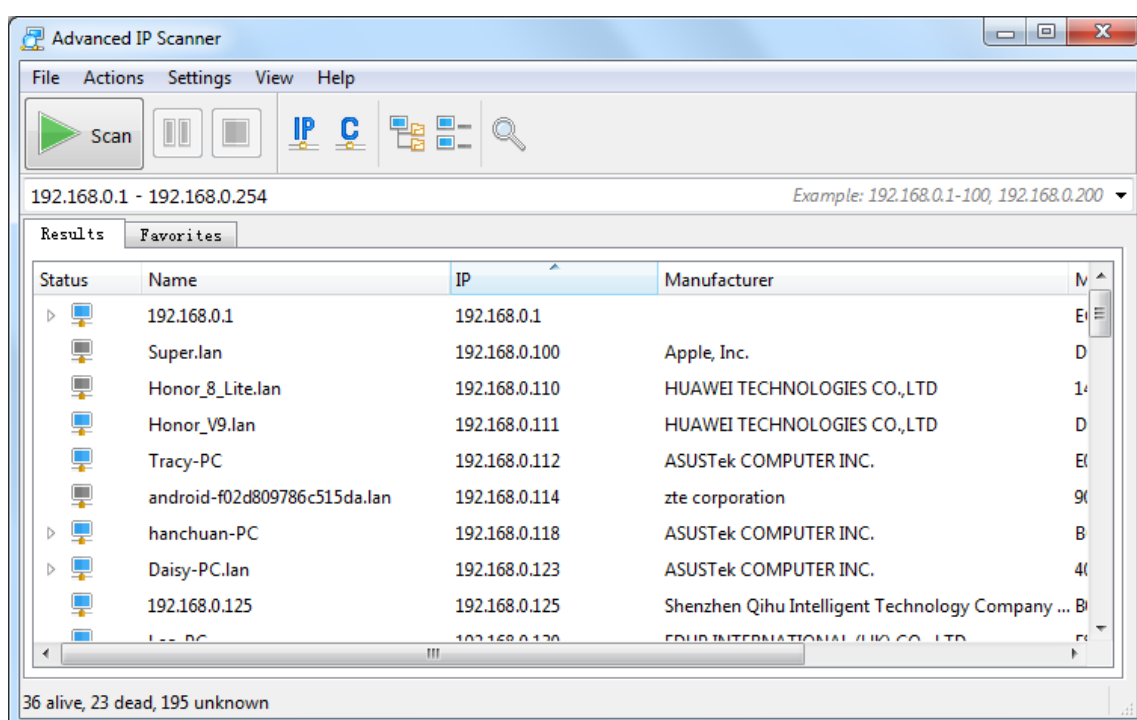
When using the Raspberry Pi, most of the time we do not connect it to piles of peripherals like keyboard and mouse; instead, we'd like to use the Raspberry Pi remotely in the PC with better functions and configurations for programing and development. In this way you won't need to prepare several displays and mice. You can use the serial port, ssh and rdp or VNC as long as you have already known the address of the Raspberry Pi.

IP Address

There are many network scanners available, and here we recommend the [Advanced IP scanner](#).

This software will detect the network segment of the computer and decide the scan range automatically. For example, when the IP is 192.168.1.101 of the computer, its scan range is 192.168.1.*

Click **Start** to start the scanning and you will see a computer list within the same network segment.



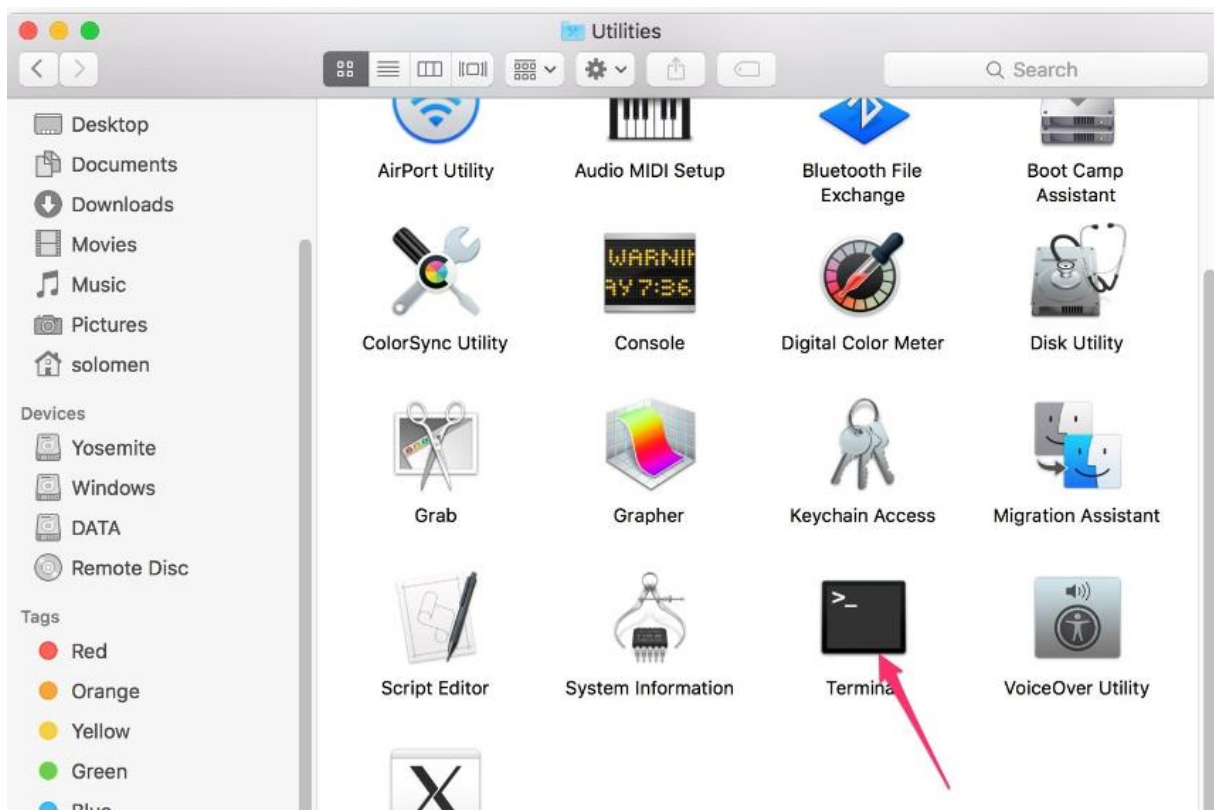
One advantage of this software is that it can find out the manufacturer of the network devices.

The device whose manufacturer is [Raspberry Pi Foundation](#) is the Raspberry Pi. Also, the IP address is helpful when you log into the Raspberry Pi remotely later.

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:

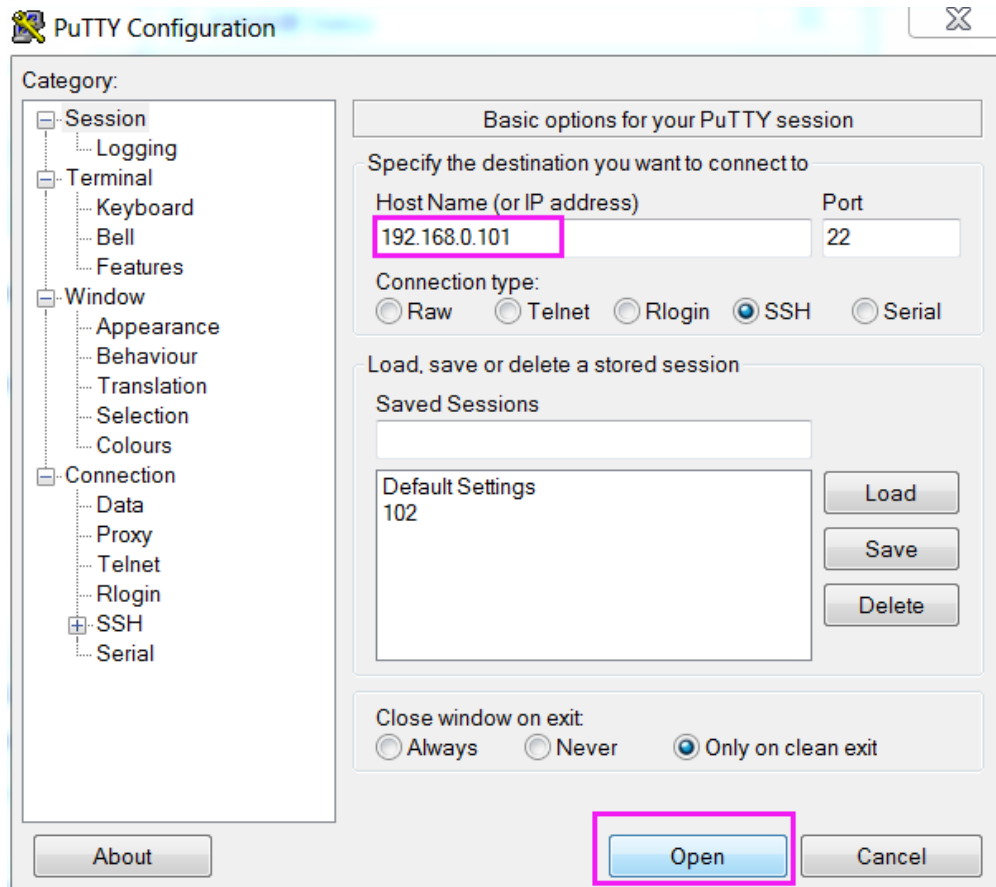
```
sudo apt-get install ssh
```

For other Linux platforms, please contact your supplier.

For Windows Users

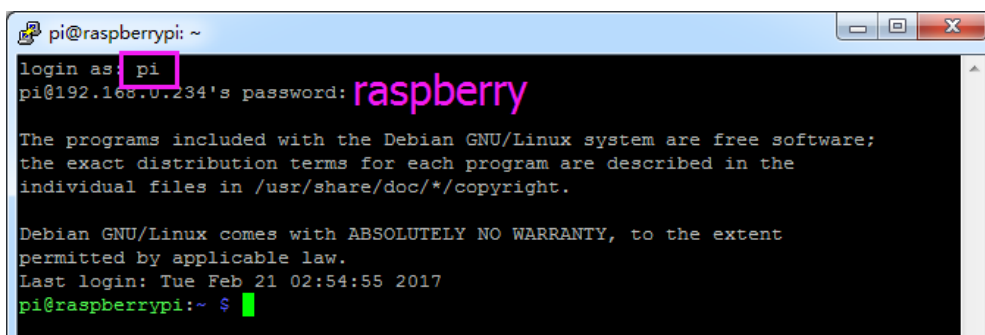
If your computer runs on Windows, you need to open the Bash shell with the help of some software. Here we recommend a tool PuTTY.

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).



6. 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.

Now, no matter what method you take, you can log into your Raspberry Pi.

D. Get Source Code

You can find the source code in our Github repositories. Download the source code by *git clone*:

```
cd ~/
git clone https://github.com/sunfounder/SunFounder_PiCar-V.git
```

Note: Please pay attention to your typing – if you get the prompt of entering your user name and password, you may have typed wrong. If unluckily you did so, press Ctrl + C to exit and try again.

Check by the **ls** command, then you can see the code directory *SunFounder_PiCar-V*:

```
pi@raspberrypi:~ $ cd ~/
pi@raspberrypi:~ $ git clone https://github.com/sunfounder/SunFounder_PiCar-V.git
Cloning into 'SunFounder_PiCar-V'...
remote: Counting objects: 747, done.
remote: Total 747 (delta 0), reused 0 (delta 0), pack-reused 747
Receiving objects: 100% (747/747), 9.08 MiB | 9.00 KiB/s, done.
Resolving deltas: 100% (293/293), done.
Checking connectivity... done.
pi@raspberrypi:~ $ ls
Desktop          Public
Documents        python_games
Downloads        SunFounder_PiCar-V
Music            Templates
Pictures         Videos
```

E. Go to the Code Directory

```
cd ~/SunFounder_PiCar-V
```

Enter the code directory and you can see the installation script:

```
pi@raspberrypi:~ $ cd SunFounder_PiCar-V/
pi@raspberrypi:~/SunFounder_PiCar-V $ ls
client      i2cHelper.py      LICENSE      README.md      show
datasheet  install_dependencies  mjpg-streamer  remote_control
```

F. Install the Environment via the Script

You can get all the required software and configuration done with the *install_dependencies* script. If you want to do step by step instead, please follow the instructions in the **Appendix**.

```
sudo ./install_dependencies
```

```
pi@raspberrypi:~/SunFounder_PiCar-V $ sudo ./install_dependencies
```

Notes:

1. The installation script will install the required components and configure for the running environment. Make sure your Raspberry is connected to the Internet during the installation, or it would fail.
2. The Raspberry Pi will prompt you to reboot after the installation. You're recommended to type in **yes** to reboot.

G. Configure the Servo to 90 degrees

After reboot, run the **picar** tool:

```
picar servo-install
```

```
pi@raspberrypi:~/SunFounder_PiCar-V $ picar servo-install
```

After the code is running, insert the rocker arm into the servo. You will see the rocker arm is rotate and then stop at a specific location. It means the servo is good. If the any of the conditions below happened to your servo, your servo is bad:

- 1) Noisy, hot
- 2) If unplug the servo line and rotate the rocker arm, it sounds like "ka" "ka" "ka" or there has no sounds of gear driving.
- 3) Rotate slowly but continuously.

If you find one of the conditions above, please send e-mail to service@sunfounder.com . We will change a new one to you. If it is broken in the process of using or assembling, you should go to the official website www.sunfounder.com to buy.

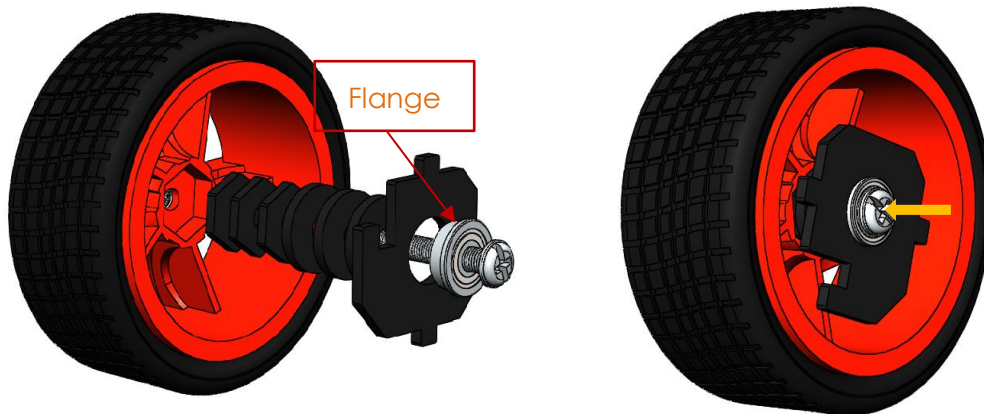
Continue to Assemble

Note: Please keep `servo-install` running in the whole process of assembly.

Note: Please pay attention to the direction of Steering Connector before assembling.

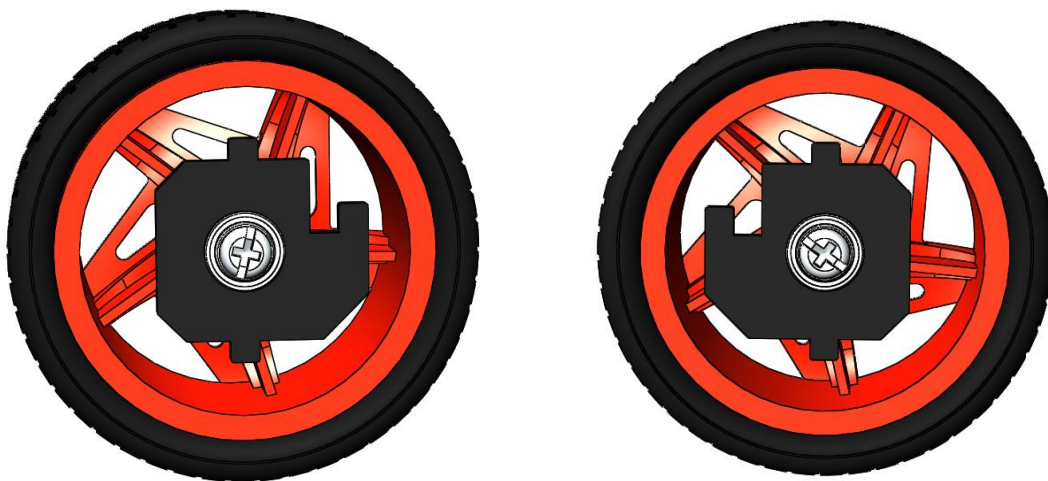
Front Wheels

Insert an **M4x25 screw** through a **Flange Bearing** (pay attention to the direction – the flange near the cap of the screw), a **Steering Connector**, 3 **Bearing Shields**, 3 **Hex Front Wheel Fixing Plates**, and a **front wheel**, into an **M4 Self-locking Nut** (note the direction) as shown below:



The Self-locking Nut should be screwed tight enough. It would be better to tighten the screw until the wheel and Steering Connector cannot move first, then loosen the screw a little, so that the Steering Plate can just move. Thus, the wheel can turn flexibly when the connection would not be too loose.

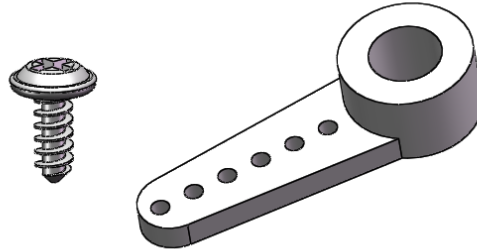
Assemble the other front wheel in the same way, but bear in mind the Steering Connector plate on the wheel should be symmetric with the previous one:



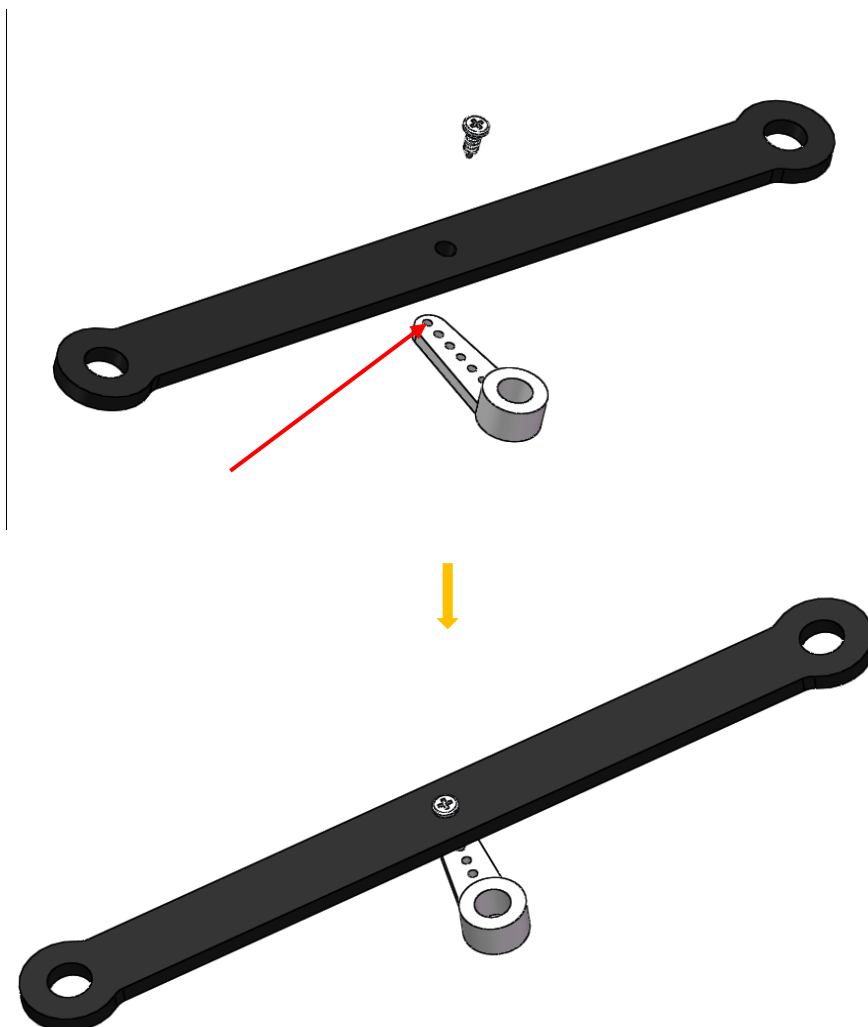
Now two front wheels have finished assembly.

Steering Part

Take out the **1-arm Rocker Arm** and the **Rocker Arm Screw** (a longer one in the servo pack):

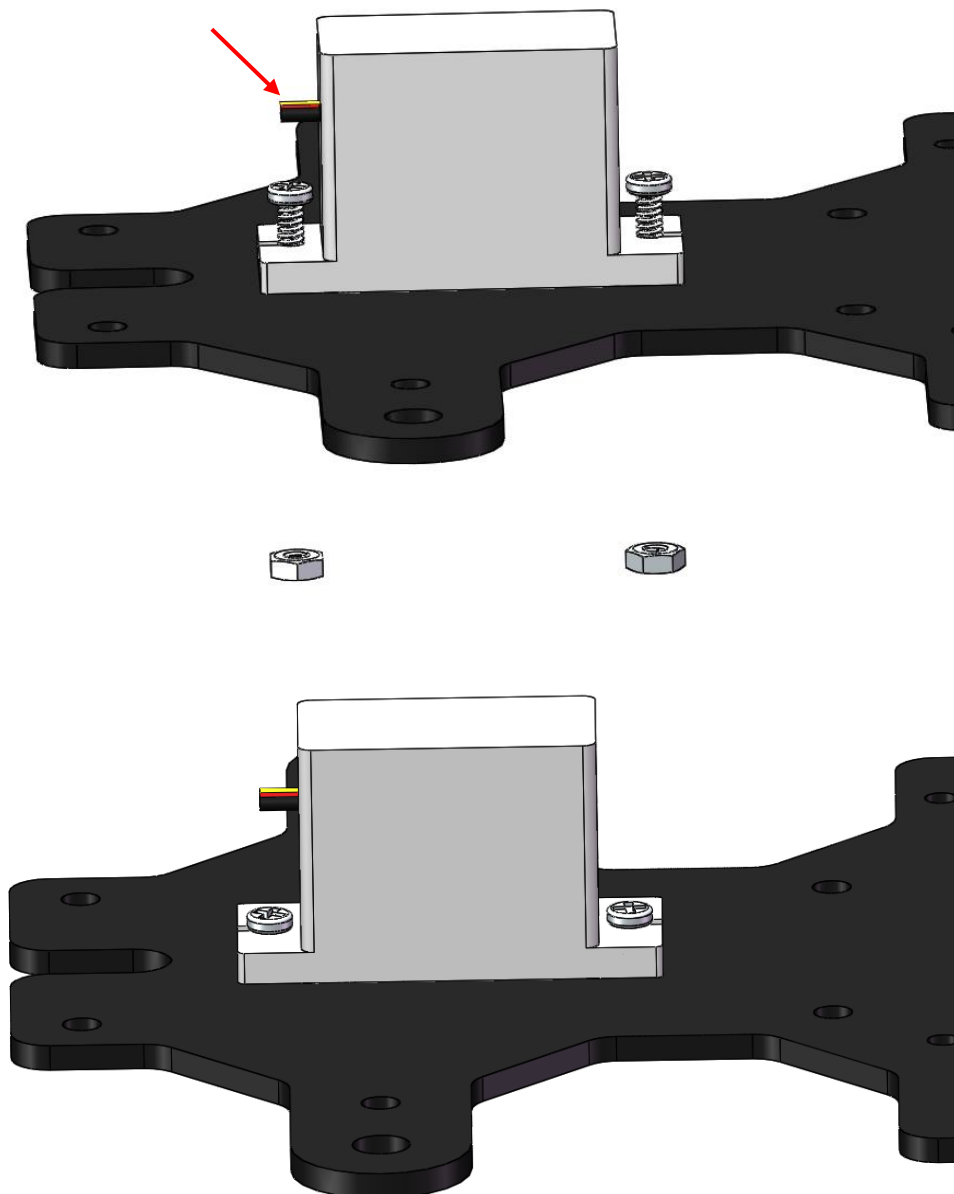


Connect the **Steering Linkage** and the **rocker arm** with the screw. **Note:** Insert it into the **FIRST** hole of the arm (as indicated by the **arrow** below) which is the farthest from the gears. Since the screw is larger than the hole, you should try to screw it hardly so as tight to the arm. Don't worry of the arm which is soft.



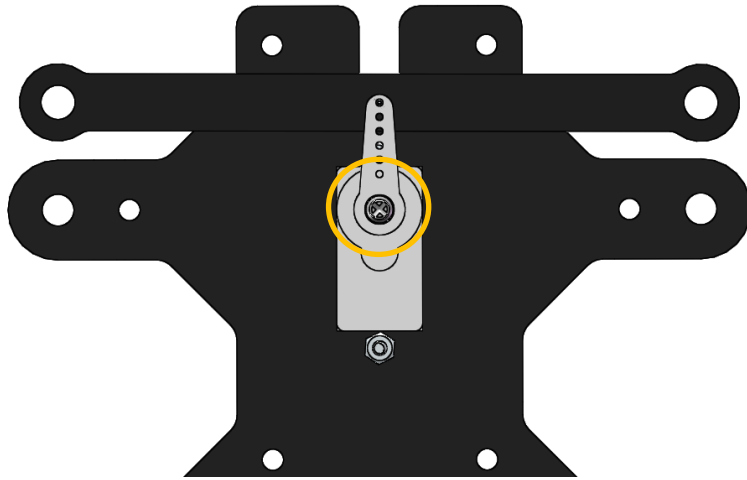
And also fasten them as tightly as possible, and **then loosen the screw a little so the Steering Linkage can move flexibly.**

Mount the servo that connects to PWM Channel 0, the steering servo, to the Upper Plate with two **M2x8 Screws** and **M2 nuts** (pay attention to the direction of the servo **wires**):



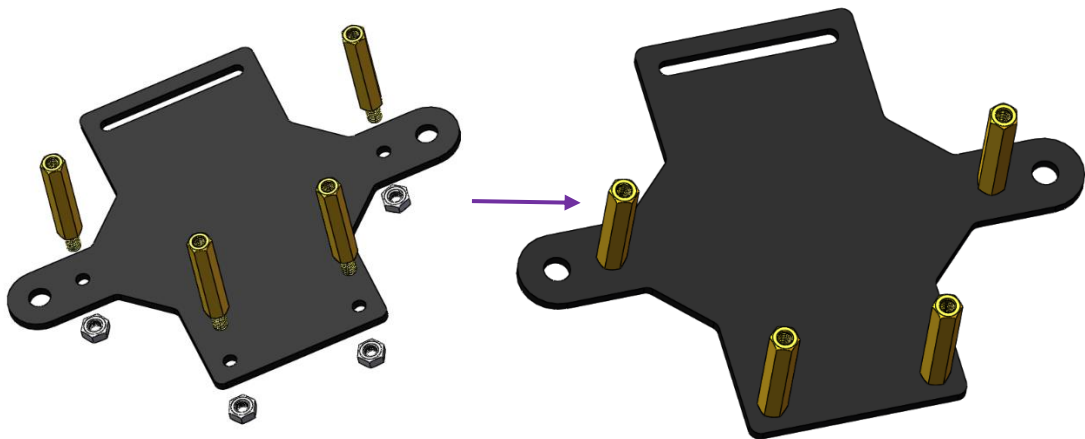
Connect the Steering Linkage and the rocker arm. **Make sure the rocker arm is in an angle as shown below.** Try to spin the arm on the shaft, **NOT** on the linkage). If it's unmovable, it means the servo is adjusted to 90 degrees. But if it spins, it indicates that the servo is not adjusted successfully; check whether the wiring is correct and run the program at **Step 5** before. Besides, **the screw is quite sharp at the end, so be careful to assemble in case of getting hurt.**

If everything is OK, take out the **Rocker Arm Fixing Screw** (the shortest) to connect them as shown below.



Front Half Chassis

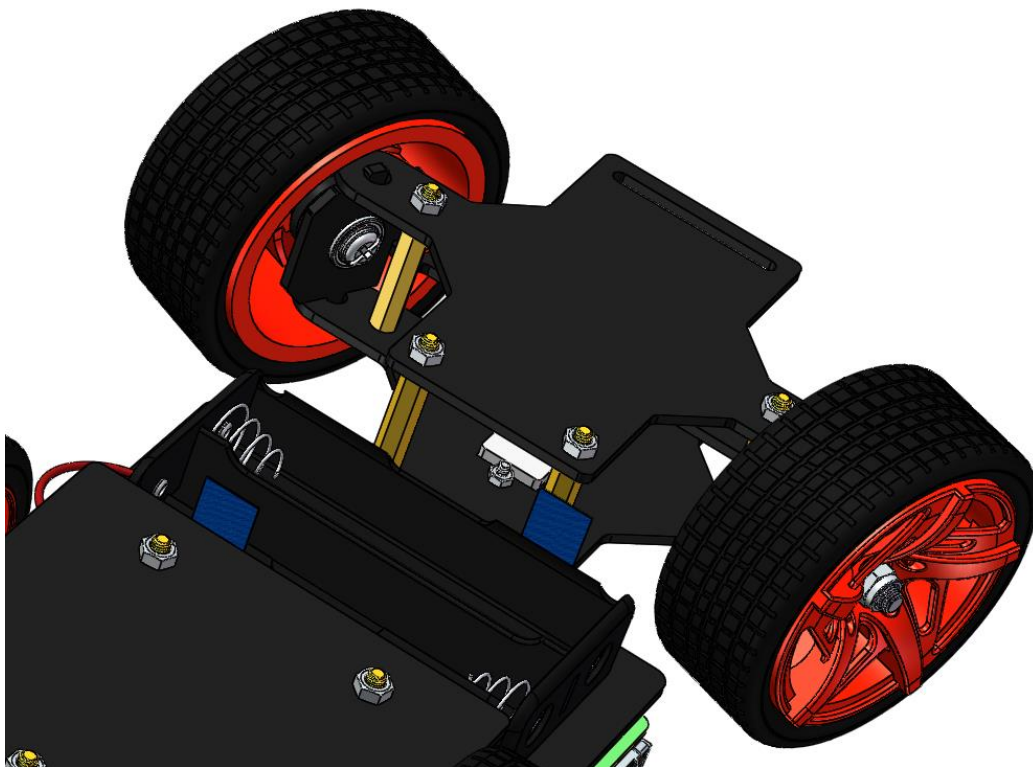
Assemble the **Front Half Chassis** with four **M3x25 copper standoffs** and **M3 nuts** as shown below:



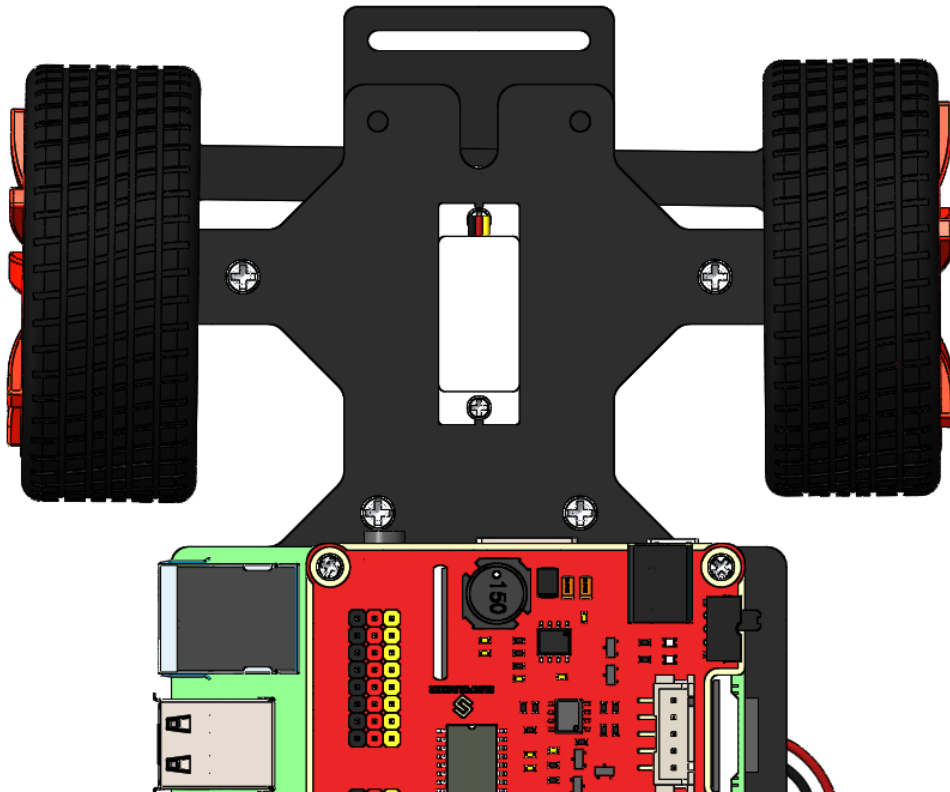
Take out the assembled front wheels and the Upper Plate, and mount the wheels onto the Upper Plate carefully: insert one **bulge** of the Steering Connector at one wheel into the **hole** on the Steering Linkage plate at the corresponding end, and the other **bulge** into the Upper Plate (pay attention to the height level of two bulges), and similar with the other wheel.



Then put the assembled Front Half Chassis onto the Upper Plate with standoffs aligned with the holes.



Hold them carefully, turn upside down, and fasten the standoffs and Upper Plate with four **M3x8 screws**:



Pan-and-Tilt

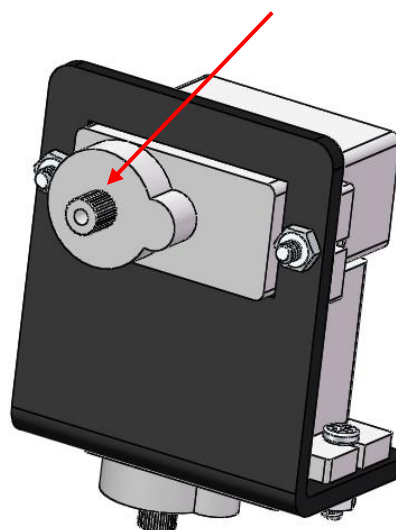
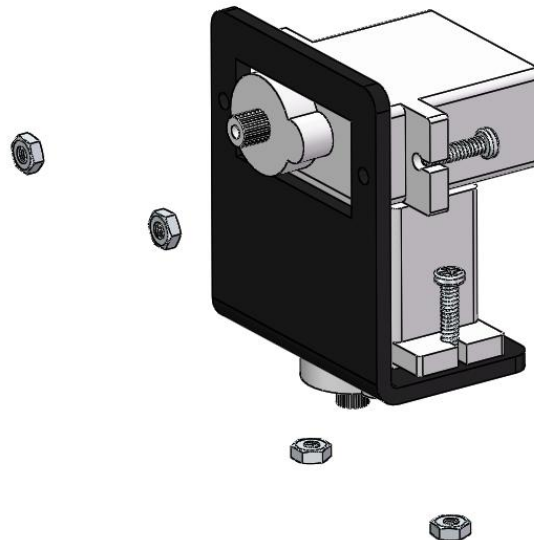
Take out the **Cross rocker arm** and mount it onto the **Pan-and-tilt Base** with four **M1.6x4 screws** (into the hole like this). Pay attention to the holes on the round rocker arm to be fastened; though the pointed end of the screwdriver is a little bigger than the screws, you can still tighten them – just insert them and screw slowly:



Do the same operation to the Tilt Arm.

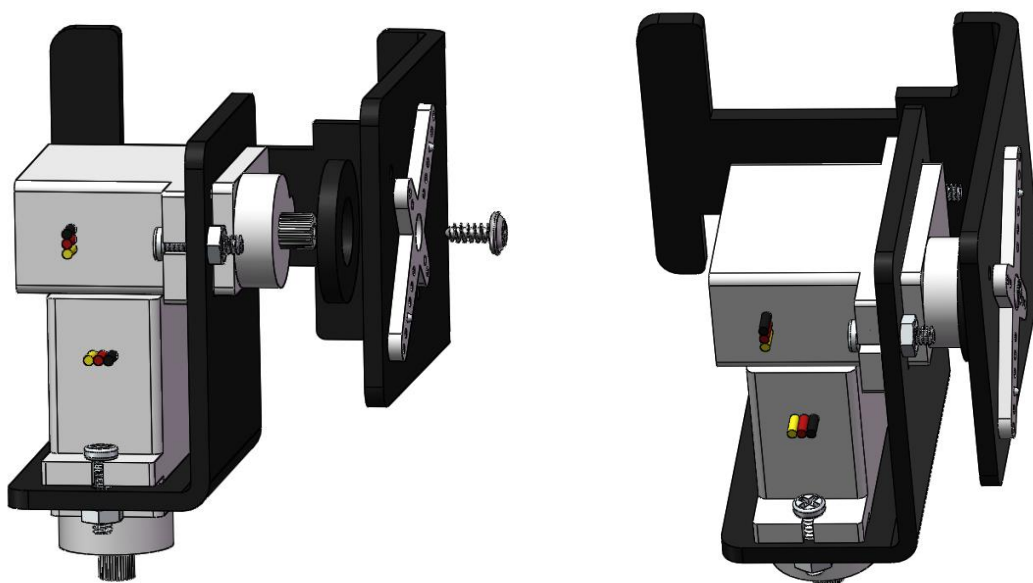


Assemble the two servos connected to PWM Channel i.e. the **Pan Servo**, to the **Pan Servo Mount** with four **M2x8 screws** and the **M2 nuts** (pay attention to the relative position of the servo shaft to the slot on the plate):

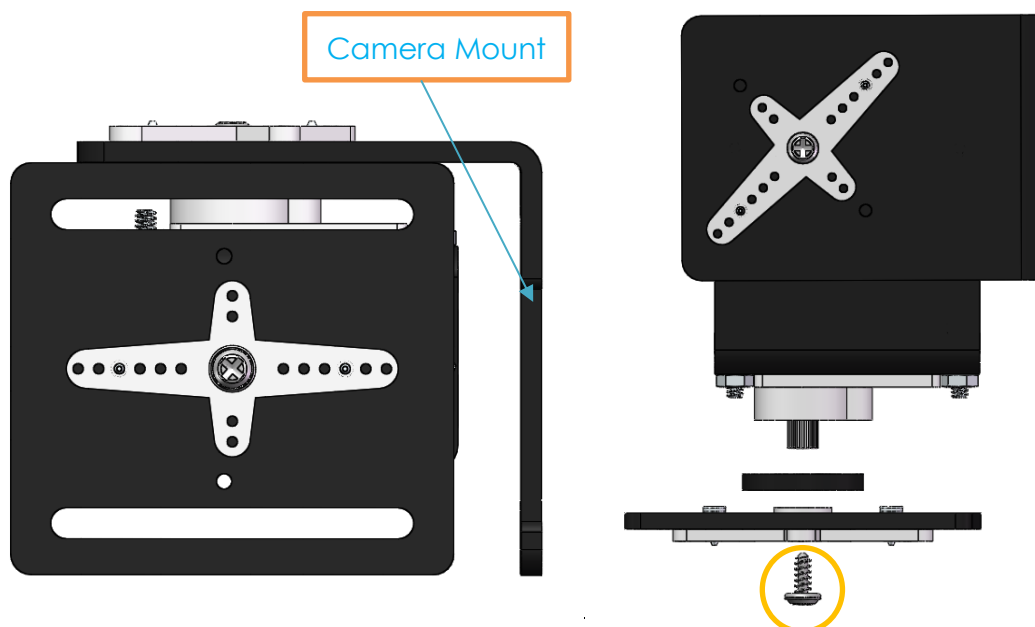


Make sure the servo shaft has spun to 90 degrees. Then assemble the **Tilt Arm** and **The Gasket** with the **Rocker Arm Fixing Screw** in a fixed angle as shown below:

Note: Do not rotate the servo by hands in case of breaking the servo down.

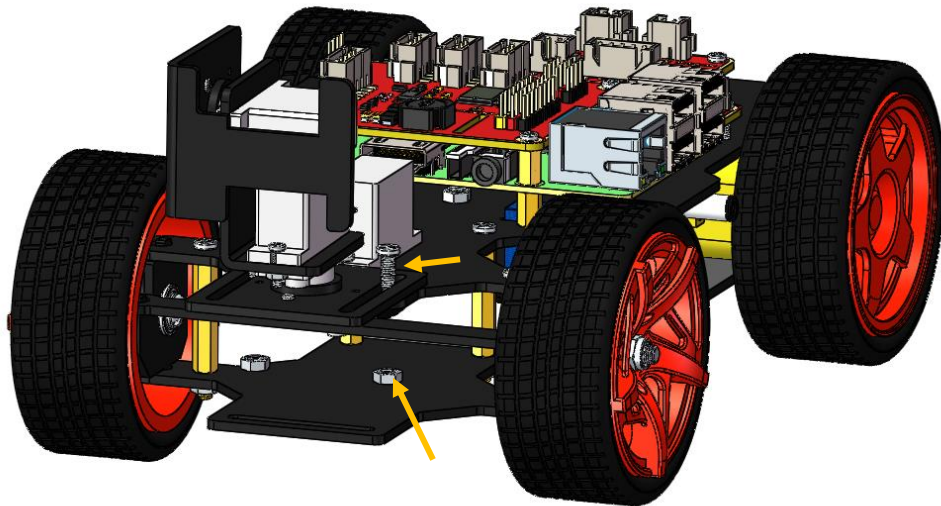


Similarly check that the servo shaft has spun to 90 degrees. Then assemble the **Pan-and-tilt Base** with the **Rocker Arm Fixing Screw** and **The Gasket** in the angle as shown below (the shaft of the pan servo should be closer to the left of the base plate):



Assemble the Pan-and-tilt to the car with two **M3x10 screws** and the **M3 nuts** if everything is OK.

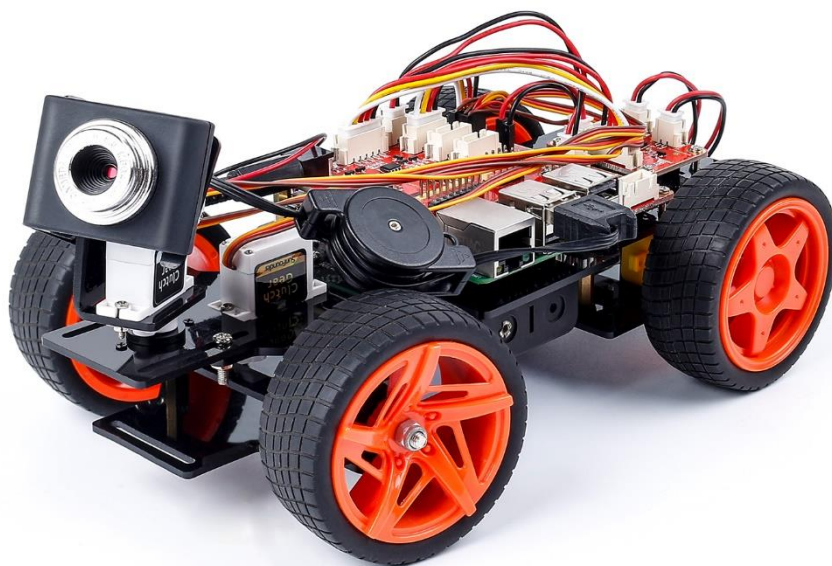
Note: In order to see clear the process of assembling, the servo in the picture is not in 90 degree. Do not rotate the servo grudgingly in case of breaking the servo down.



Other Components

Take out the camera and nip it to the Camera Mount. Connect its USB cable to the USB port on the Raspberry Pi.

So now, the whole assembly is DONE! Congratulations!



Installing the Client (Operation on the PC)

Since you've finished the car building, now it's time to configure the environment for the client. Or you may skip this part and go to the subsequent Web Client for controlling on the webpage.

Environment Configuration

The client is written by Python 3 + PyQt5, so it's necessary to install the two in your PC.

Python 3 website: <https://www.python.org/downloads/>

Select the proper version of Python 3 for your PC, and follow the instructions to install.

PyQt5 website: <https://www.riverbankcomputing.com/software/pyqt/download5/>

The installation method is provided: <http://pyqt.sourceforge.net/Docs/PyQt5/installation.html>, and the preferred way is installing by pip:

```
pip3 install pyqt5
```

For Windows users who are inclined to click **Next** all the way during the installation, you can try to search out the .exe file: <https://sourceforge.net/projects/pyqt/>. Then you can complete the installation in the same way as other software in Windows.

Notes:

1. You may need to install python3-pip. If you get the prompt of "pip3 not found", first


```
apt-get install python3-pip
```

Then,

```
pip3 install pyqt5
```

2. The method of `pip3 install pyqt5` only supports the 32-bit and 64-bit Windows, 64-bit OS X and 64-bit Linux. If your PC is running on the 32-bit Linux or other systems, you will be prompted that the corresponding version file cannot be found. In this case, please take the second method **Building and Installing from Source** provided on the PyQt5 download page.

Check whether the installation is successful. Open the Python 3 command line/Python 3 shell after installation, and type in `import PyQt5`. If there is no error message, then everything is OK:

On **Windows**, double click on the Python 3.5  , type in `import PyQt5`, and run:

```
Python 3.5 (32-bit)
Python 3.5.2 (v3.5.2:4def2a2901a5, Jun 25 2016, 22:01:18) [MSC v.1900 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import PyQt5
```

On **Linux** and **Mac OS X**, type in command `python3` in the terminal, and `import PyQt5` and run

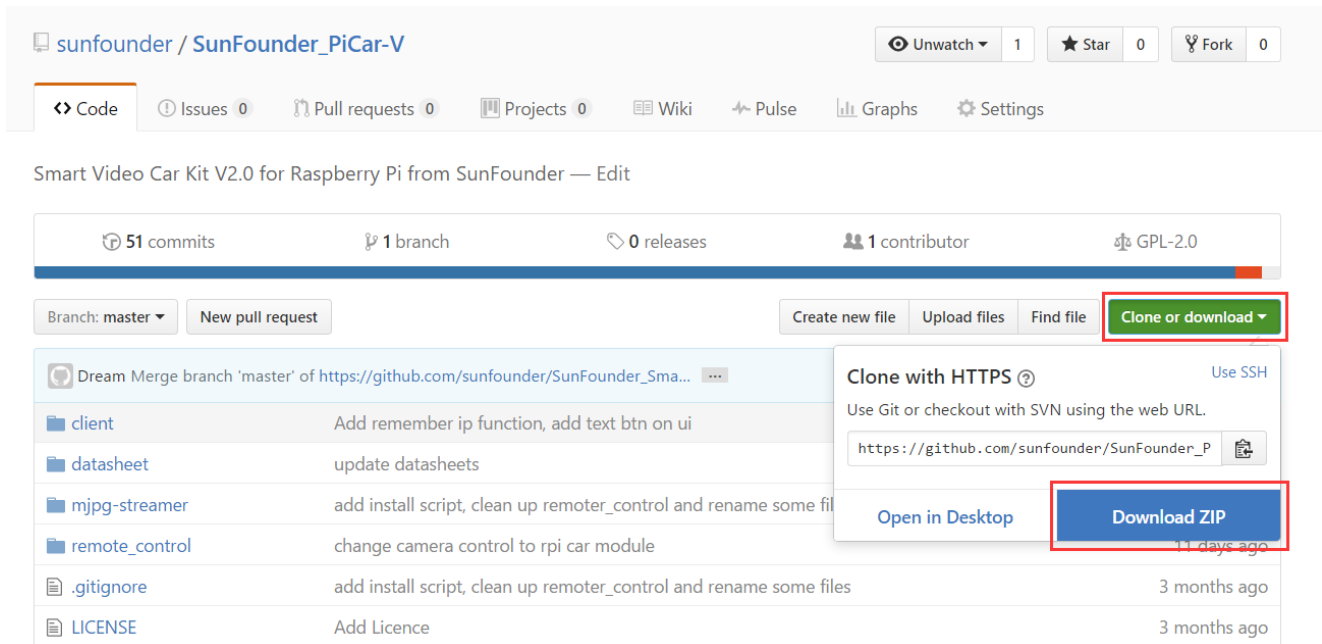
```
pi@pi-virtual-machine: ~
pi@pi-virtual-machine:~$ python3
Python 3.5.2 (default, Jul 5 2016, 12:43:10)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import PyQt5
```

Code Package Download

Open a web browser and go to the Github page of the PiCar-V:

https://github.com/sunfounder/SunFounder_PiCar-V

Click **Clone or download** on the page, and click **Download ZIP** as shown below.



After download, unzip the file.

Or in **Linux**, you can download the code package by git clone (**recommended**):

Install git with software manager:

Ubuntu/Debian:

```
sudo apt-get install git
```

Fedora/Red Hat:

```
sudo yum install git
```

Then clone the code:

```
git clone https://github.com/sunfounder/SunFounder_PiCar-V.git
```

Note: Please pay attention to your typing – if you get the prompt of entering your user name and password, you may have typed wrong. If unluckily you did so, press Ctrl + C to exit and try again.

Getting on the Road!

Make sure all the procedures have been finished, and check there is no problem about the mechanical assembly and software installation. Then you can start the car now. Install two batteries, and turn on the power switch.

Tips: If you keep the Raspberry Pi power on from the beginning, now you do not need to turn it off. DO NOT unplug the Micro USB cable until you've installed the batteries and switched it on. So you do not need to turn it on or off again. If later the batteries run out, or half does, and you want to do the debugging, you can also just plug in the USB cable for power without shutting down. The Robot HATS integrates a backflow prevention diode to protect it from being damaged by the USB power, so does the Raspberry Pi. So take it easy for using the two power sources at the same time!

Here is what we're going to do:

Use the Raspberry Pi as the **server**. Run a web server with an API for controlling the car and transmitting images captured by the camera.

Then take a PC, cell phone, or tablet as the **client**, and acquire the images from the camera and control the car by calling the API of the web server.

1. Run the Server(Operation on Raspberry Pi)

Remotely log into the Raspberry Pi. You can find a startup script *start* under the *remote_control* directory. Run the script to start the web service.

```
cd ~/SunFounder_PiCar-V/remote_control
sudo ./start
```

The script will enable the service and the corresponding data will appear. The hardware is initialized at the same time, so the servos connected to the front wheels and the pan-and-tilt will turn, indicating the hardware initialization is done.

```
pi@raspberrypi:~ $ cd ~/SunFounder_PiCar-V/remote_control/
pi@raspberrypi:~/SunFounder_PiCar-V/remote_control $ sudo ./start
Server running
Performing system checks...

DEBUG "front_wheels.py": Set debug off
DEBUG "front_wheels.py": Set wheel debug off
DEBUG "Servo.py": Set debug off
DEBUG "back_wheels.py": Set debug off
DEBUG "TB6612.py": Set debug off
DEBUG "TB6612.py": Set debug off
DEBUG "PCA9685.py": Set debug off
DEBUG "camera.py": Set debug off
```

```
DEBUG "camera.py": Set pan servo and tilt servo debug off
DEBUG "Servo.py": Set debug off
DEBUG "Servo.py": Set debug off
None
System check identified no issues (0 silenced).
November 09, 2016 - 06:28:41
Django version 1.10.2, using settings 'remote_control.settings'
Starting development server at http://0.0.0.0:8000/
Quit the server with CONTROL-C.
```

If you get the result similar as shown above, the server is ready. Now move on to start the client.

2. Run the Client (Operation on PC)

There are **TWO** methods to run the client. On one hand, you can open *Client* in the directory of the SunFounder_PiCar-V in PC, run *client.py* by Python 3. Or, if you happen to fail to install PyQt 5 or Python 3, or just want to control it on your phone or tablet, we have also prepared a web client via web browser.

- Method A

Open *Client* in the directory of the SunFounder_PiCar-V in PC, run *client.py* by Python 3.

If yours PC runs on **Linux**, you can run python 3 client.py in the terminal.

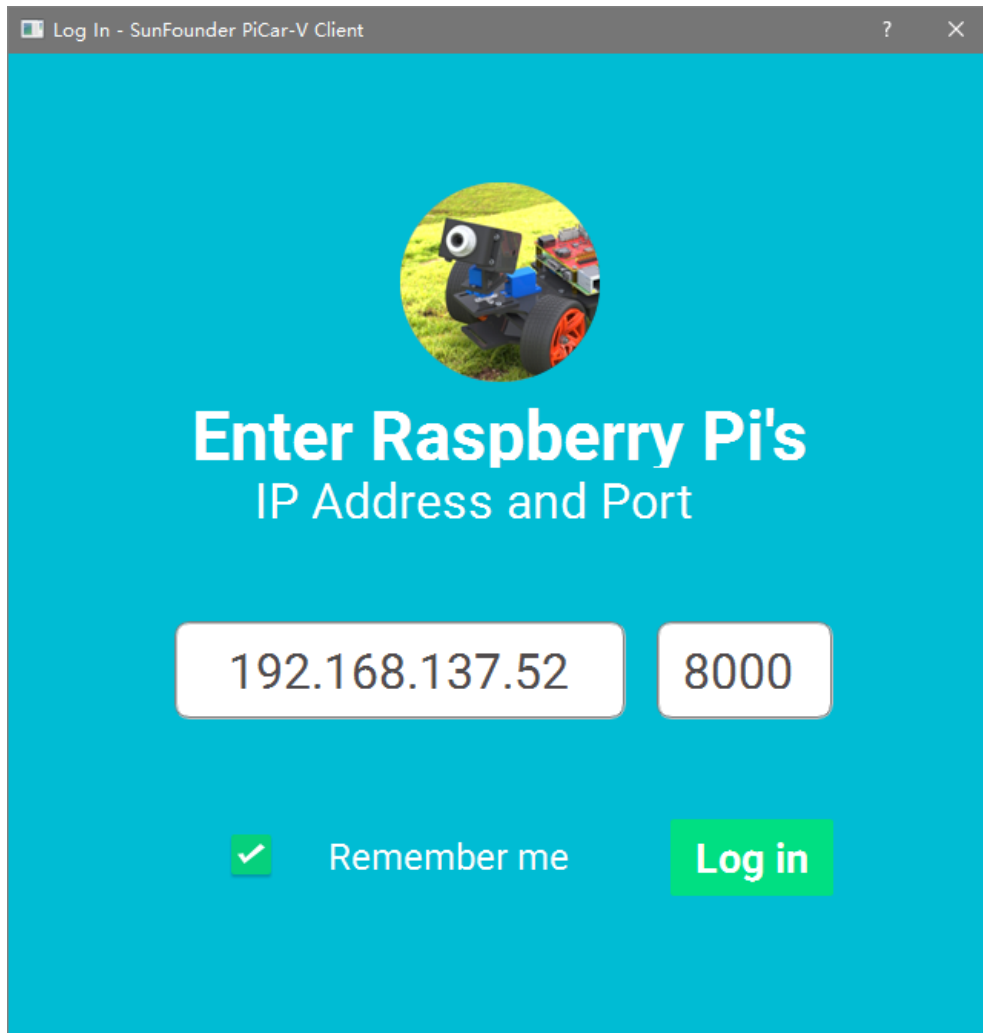
```
python client.py # If Python 3 is the default or only python
python3 client.py # Or you have both Python 2 and Python 3, and the default is Python 2
```

If it runs on **Windows**, since Python 3 has been installed previously, you can just double-click to run *client.py*. Make sure it runs under Python 3 not Python 2, if both are installed in your PC.

Notes:

In **Linux**, if only Python 3 is installed, python = python 3 in the terminal; if there're both Python 2 and Python 3 (quite a lot of systems have those two pre-installed), the python command will start Python 2 by default. To use Python 3, you need to use python 3 commands. Or you can configure the Python environment, to set the Python 3 as default.

Run the client and you will see:



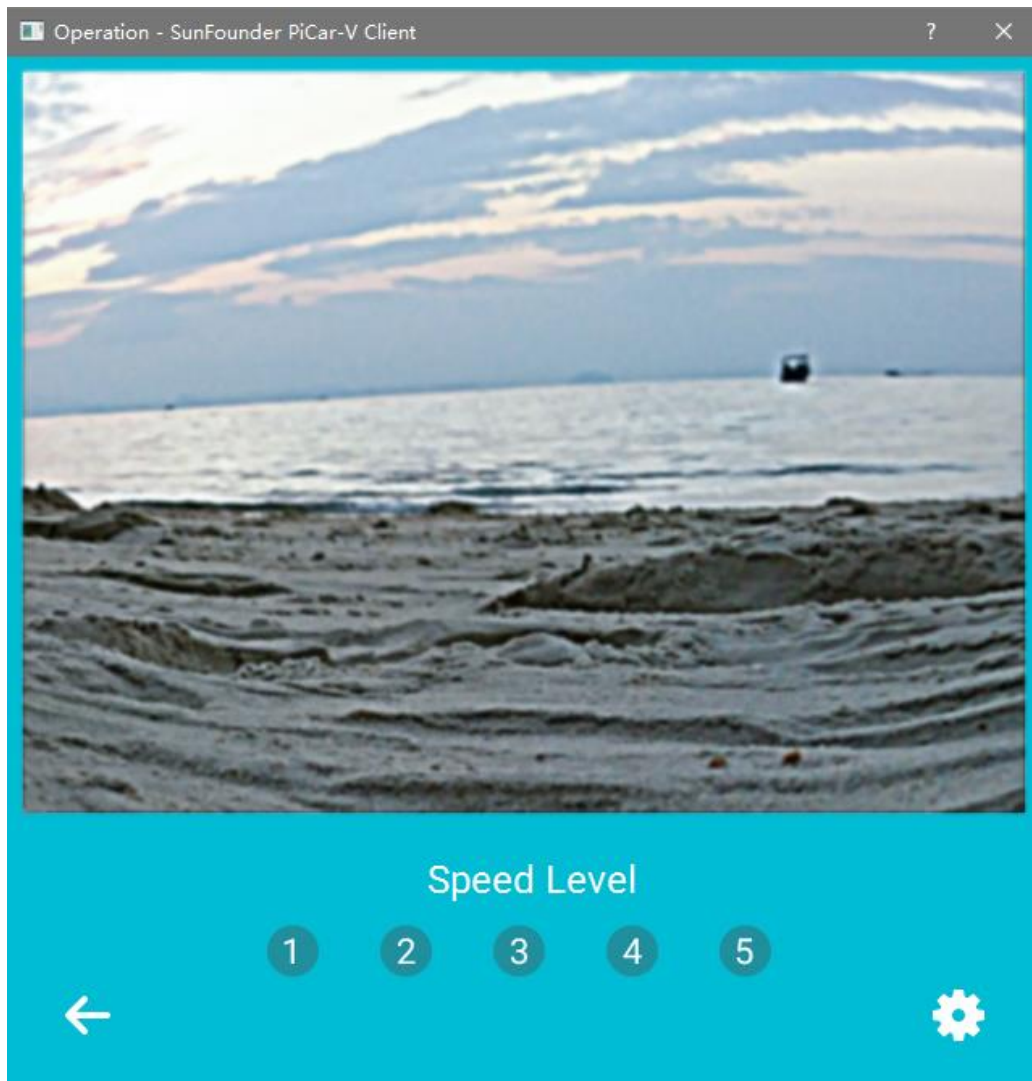
Log In - SunFounder PiCar-V Client

Enter Raspberry Pi's
IP Address and Port

192.168.137.52 8000

☒ Remember me **Log in**

This is the login interface of the client. Enter your Raspberry Pi's IP address and the port, 8000 by default, and then click **Log in** to connect to the Raspberry Pi. It may take a while to establish connection, sometimes may not respond at all, so just wait several seconds. If it's successful, you'll be on the operation interface. If not, double check your Raspberry Pi's IP address and make sure the server script is run without error prompts.



The interface will show the view captured by the camera in a real-time manner.

There are some number buttons at the bottom to adjust the car's speed, ascending from level 1 to 5. You can also press the key 1-5 on the keyboard. The rest control movements of the car are implemented by keyboard.

On keyboard:

W, S: to control the rear wheels to move forward/backward;

A, D: to control the front wheels to turn left/right;

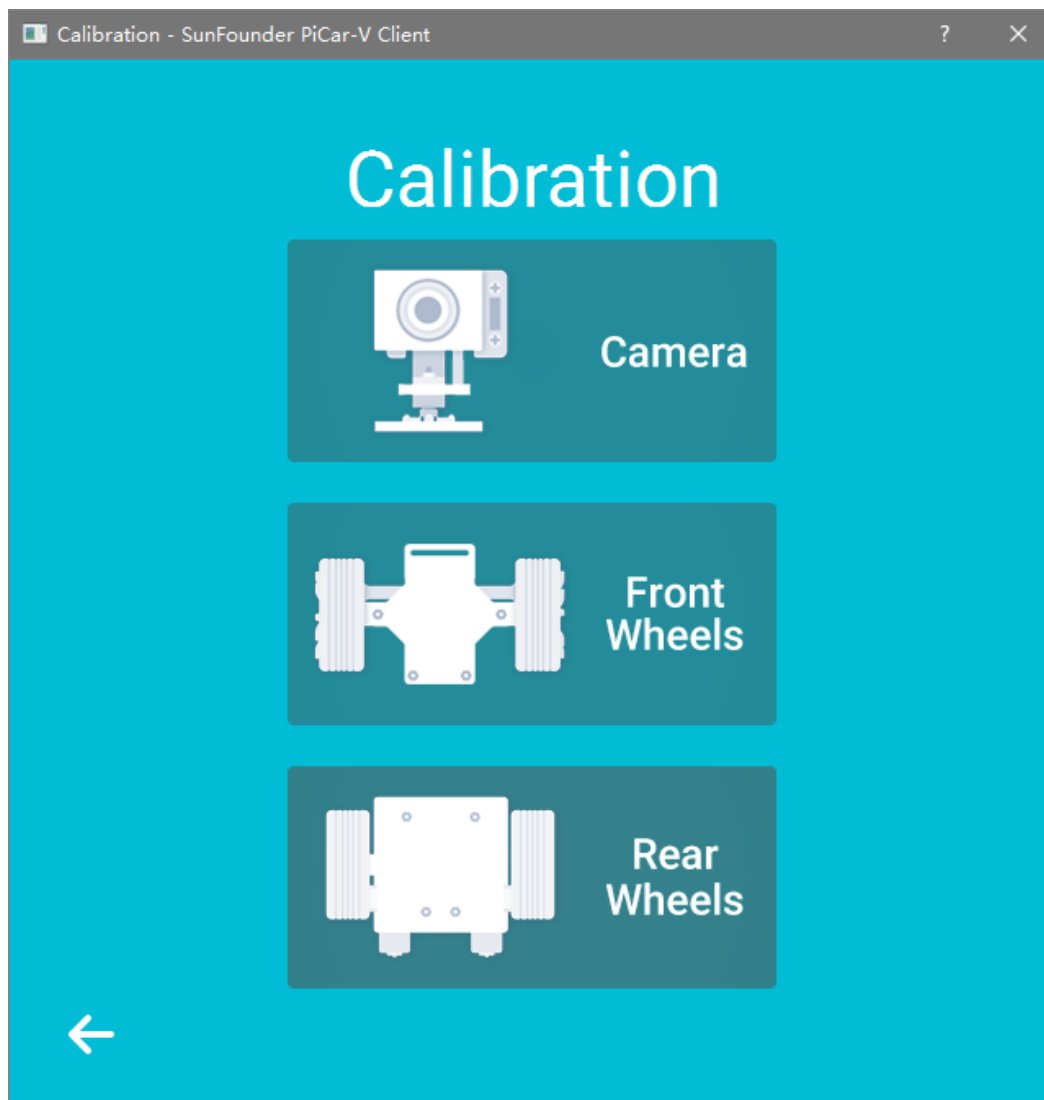
↑, ↓: to control the camera's rotating up/down;

←, →: to control the camera's rotating left/right.

Click the left arrow button at the bottom left or press **Alt + ←** to return to the previous page, that is, the login.

For better application, you're recommended to calibrate the assembled car first.

Click the setting button at the bottom right or press **Alt + S**, and you will see the **Calibration** page as below (click the three larger icons for the corresponding calibration):



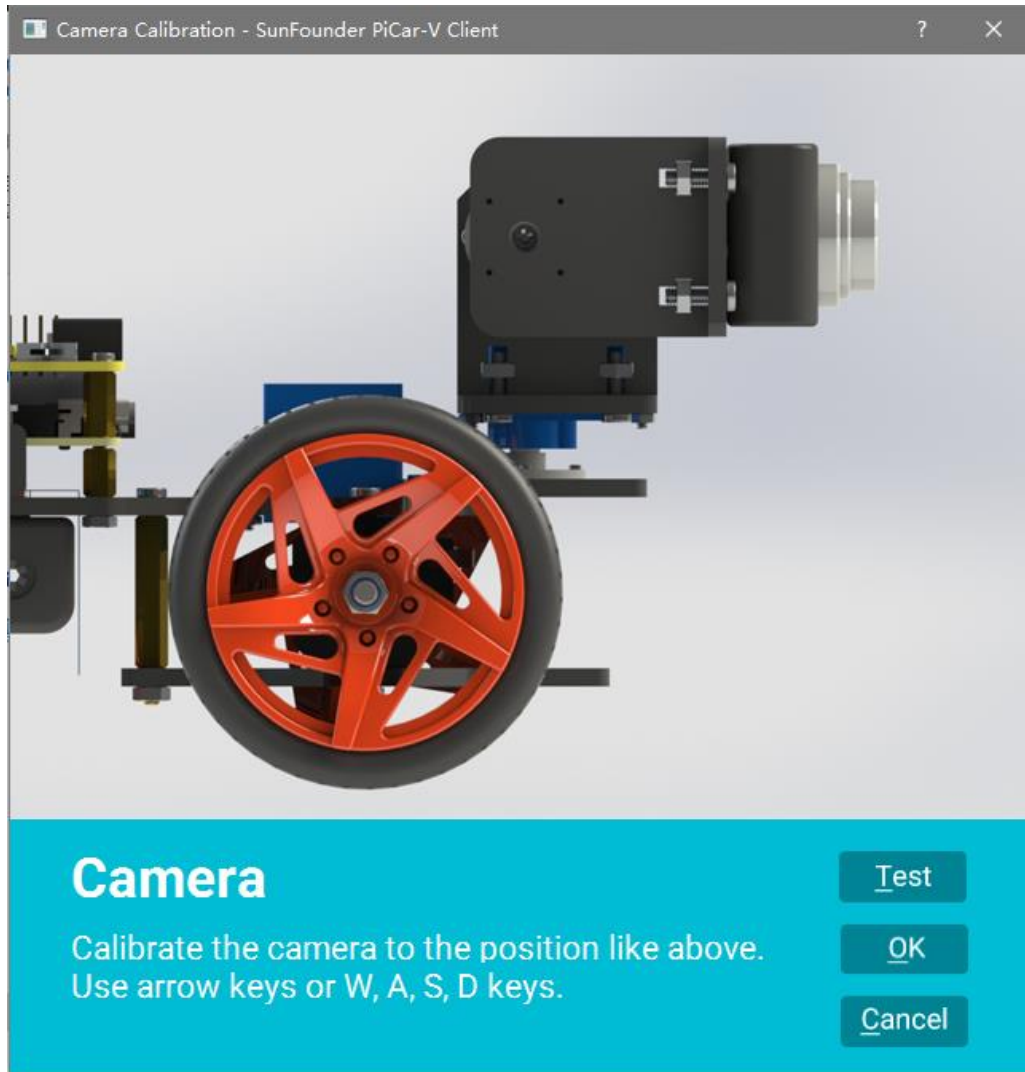
Camera – to calibrate the camera movement. Short cut: **Alt + C**.

Front Wheels – to calibrate the front wheels. Short cut: **Alt + F**.

Rear Wheels – to calibrate the rear wheels. Short cut: **Alt + R**.

Click the arrow return button or **Alt + ←** and you will go back to the operation interface.

Calibration: Camera



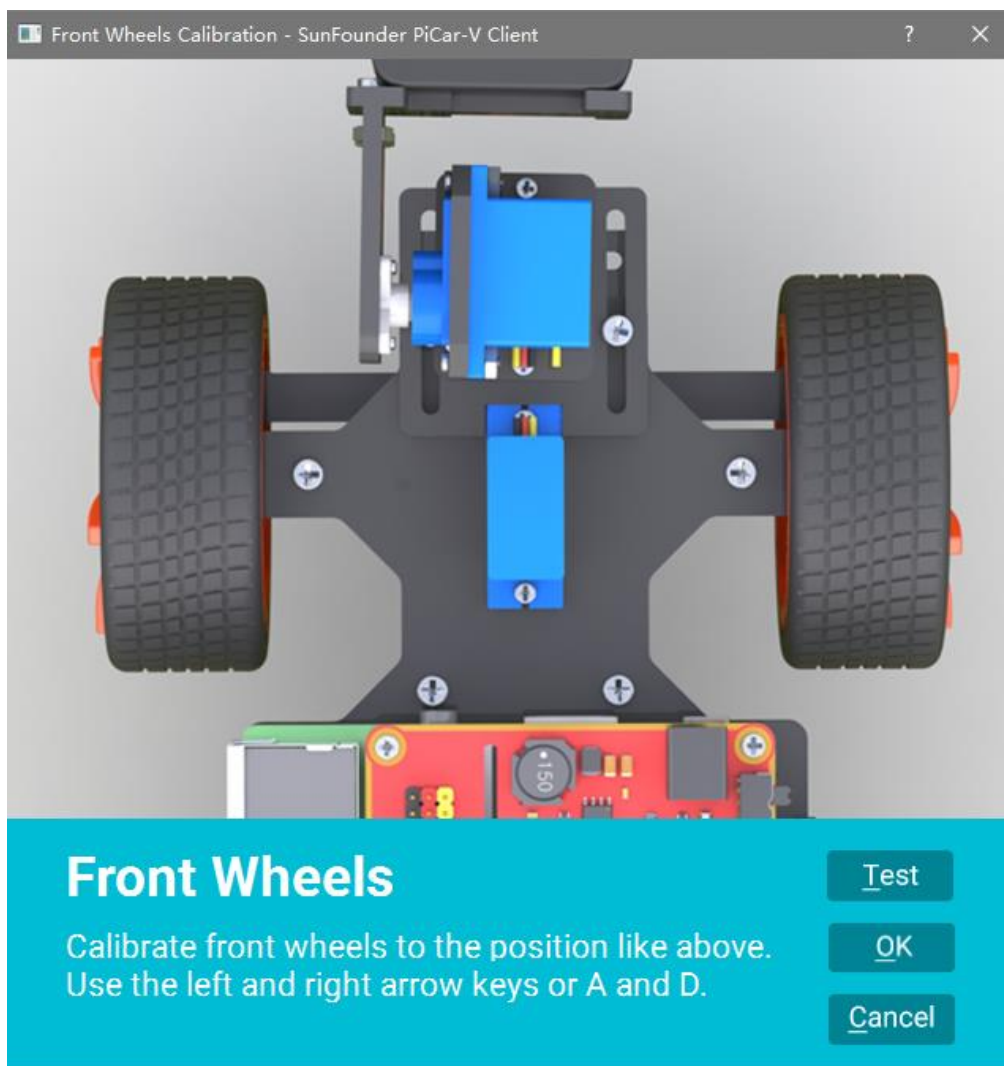
When completed, the camera should look like the figure above. The camera should be in the center line of the car and face exactly front. You can use the keyboard to adjust, by the arrow keys \uparrow , \leftarrow , \downarrow , and \rightarrow , or **W**, **A**, **S**, and **D**. The servo turning angle is very small, so it's quite suitable for fine tuning; press and hold it to coarsely adjust quickly. Click **OK** or press **Alt + O** to save the calibration result; click **Cancel** or press **Alt + C** to cancel the calibration and exit.

Notes:

During the calibration, the servo may get stuck with an abnormal sound, and heated abnormally after a while if the servo shaft is not adjusted to 90° in the previous mechanical assembly. You should at once disconnect the wires of the servo and remove the rocker arm to readjust the shaft to 90° (refer to the previous **Servo Configuration** part).

After all the adjustments are done, click **OK**, and the server will write the data into the database files. Then the servo will respond to the database initialization with a little movement. Only after the initialization is done can you start the next step, or the server will reject the calibration request.

Calibration: Front Wheels

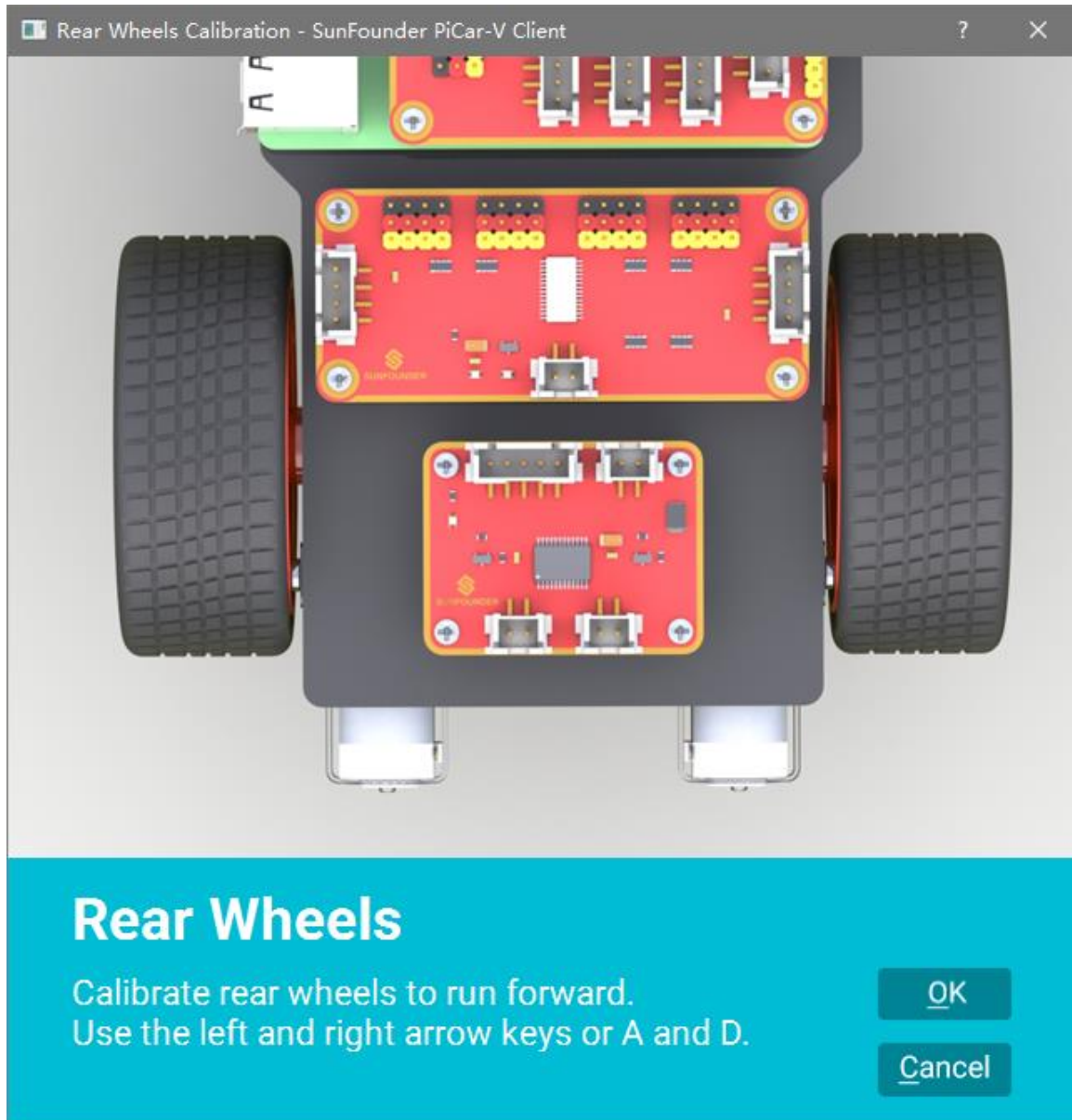


Adjust the front wheels to the appropriate position according to the above figure. You can use the keyboard to adjust, that is, \leftarrow , \rightarrow , **A**, and **D**.

Click **OK** or press **Alt + O** to save the calibration result; click **Cancel** or press **Alt + C** to cancel the calibration and exit.

If the servo gets stuck, remove the rocker arm and readjust again (refer to the **Servo Configuration**).

Calibration: Rear Wheels

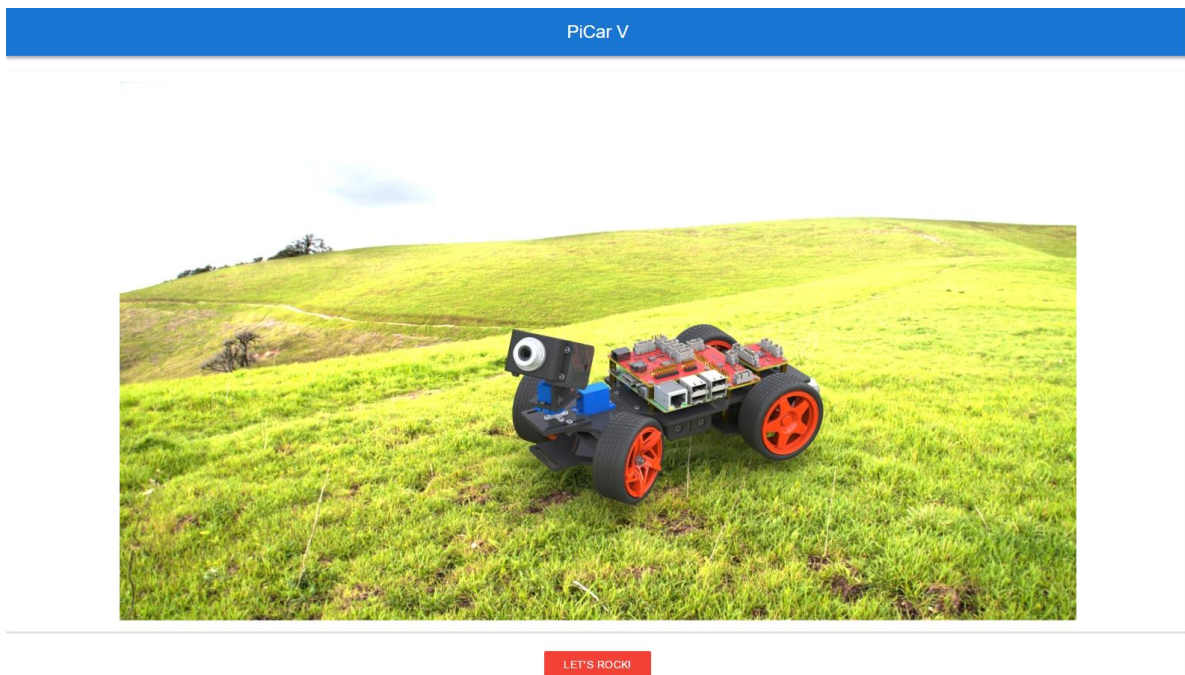


Adjust the rotating direction of the rear wheels, to make the car go forward. Adjust the wheels by the keys ← and → or **A** and **D**. The left and right is to control the left and right wheels respectively and each clicking will reverse the rotation of the wheel.

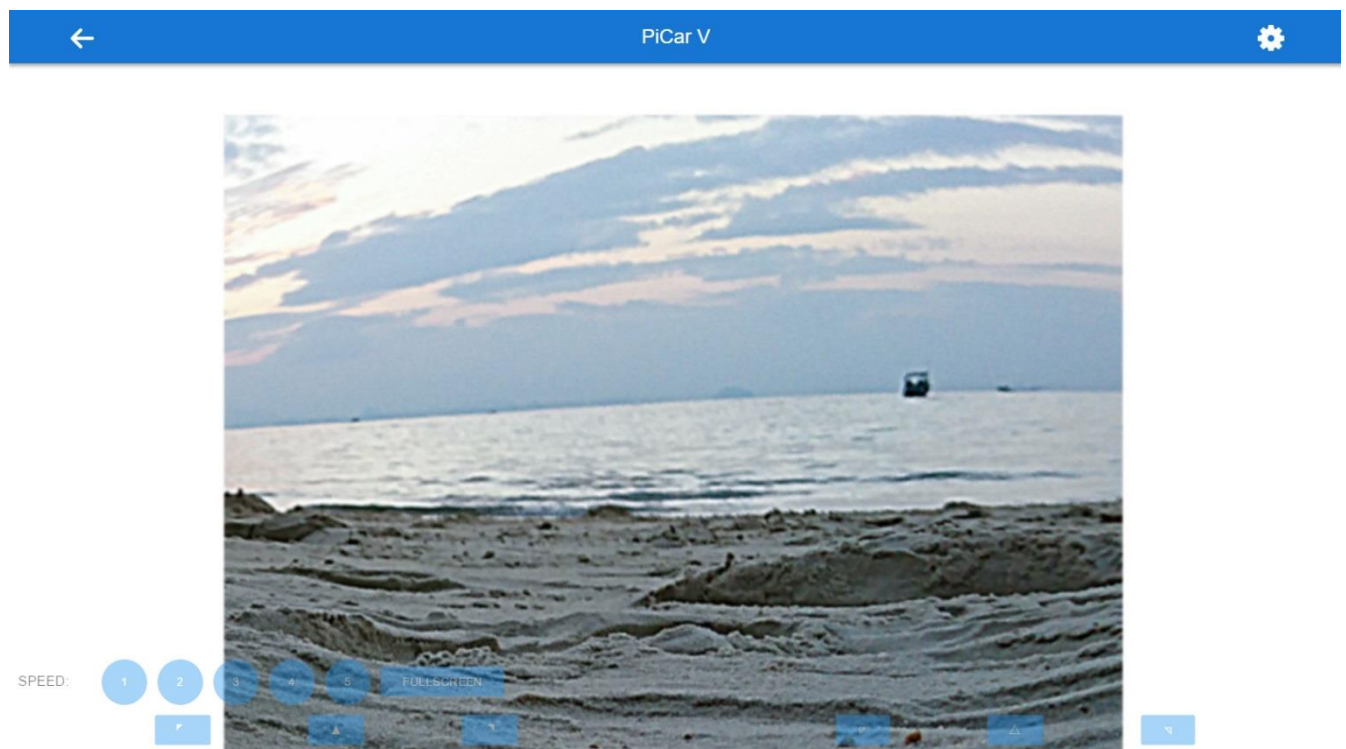
After all the calibration is done, you can return to the operation interface. Make sure both LED indicators on the Robot HATS board light up. Now you can unplug the Micro USB cable and put the car on the ground. Let's start the journey!

- **Method B**

Visit the server of the car at http://<RPi_IP_address>:8000/. You will see a welcome page:



Click **LET'S ROCK** to go to the operation interface:



On this page, you can press the keys **W**, **A**, **S**, and **D** on the keyboard to control the car to move **forward**, **backward**, **turn left**, and **turn right**, press the arrow keys to control the camera's movement, and number **1~5** to change the speed level.

Also for mobile phones, tap the **FULLSCREEN** button to have a better view and performance. Then, tap the buttons of 5 speed levels on the page to control the speed, and the arrow buttons to control the direction of the car and the pan-and-tilt. But you can only tab **one** touchpoint at one time.



Click **FULLSCREEN** again to bring back the title bar. And then, tap the setting button at the top-right corner of the page to go to the calibration page:

←CalibrationOK

Camera Calibration:

▲

◀▶

▼

Front Wheels Calibration:

◀▶

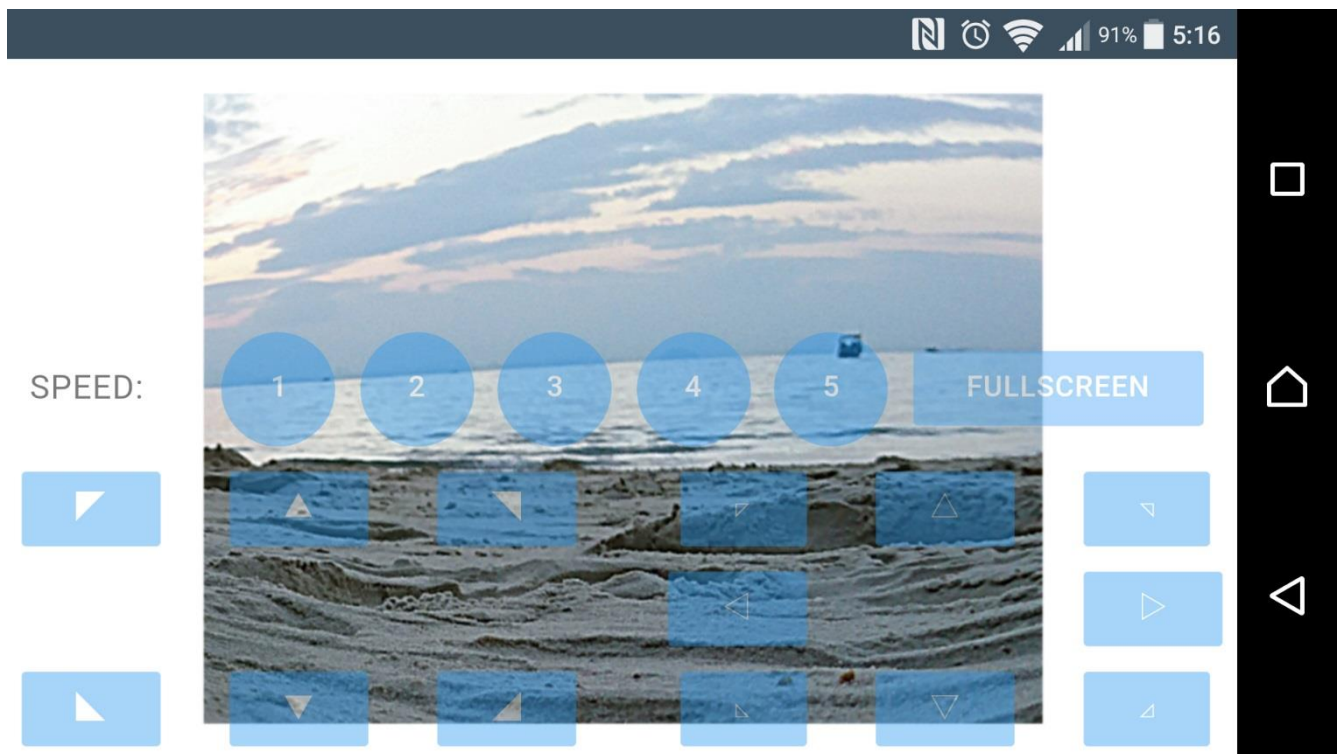
Rear Wheels Calibration:

LEFTRIGHT

There are three calibration parts: **Camera Calibration**, **Front Wheels Calibration** and **Rear Wheels Calibration**.

Do the calibration for the servos and motors just as done in the PyQt client. Click **OK** to save the result after all the calibration is done.

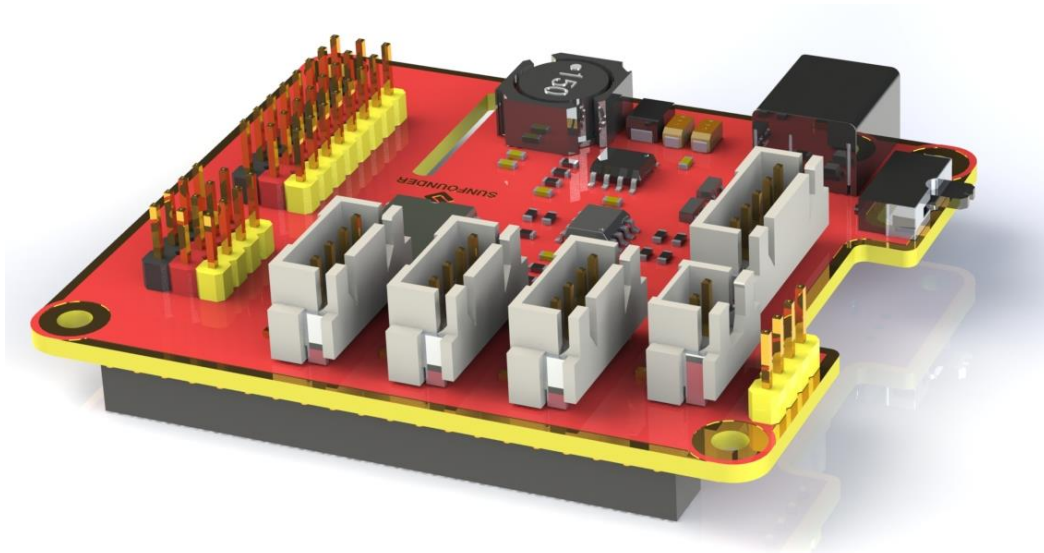
Take this screenshot from an Android phone:



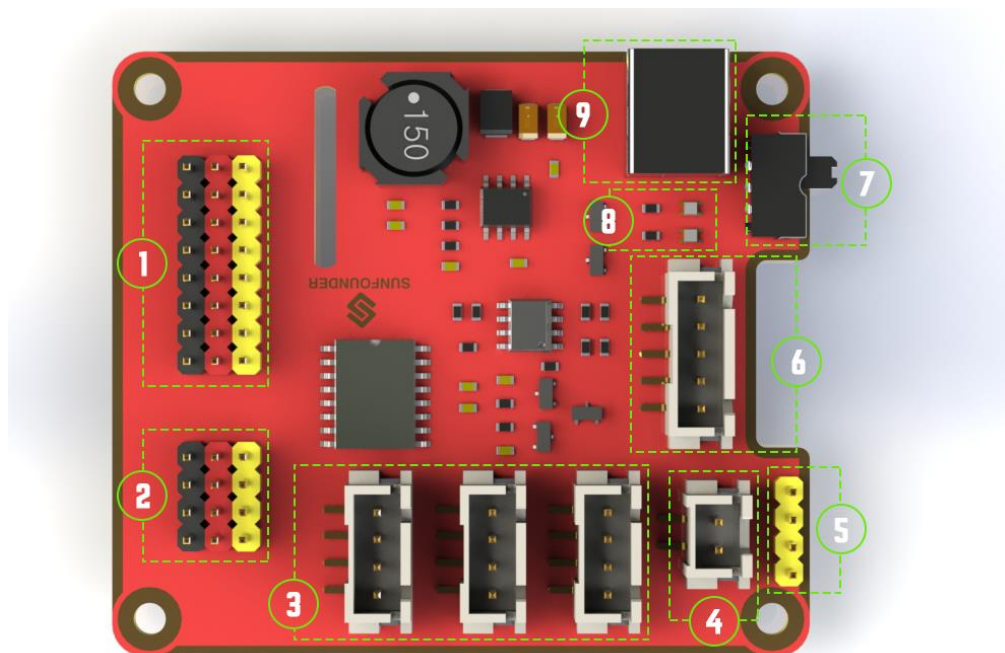
Though the appearance may not be as good as on the PC. Later updates may be released irregularly on [Github](#). You are welcome to fork our repository and submit a Pull request with your changes. If there is no problem after testing, we are more than pleased to merge your request.

Components

Robot HATS



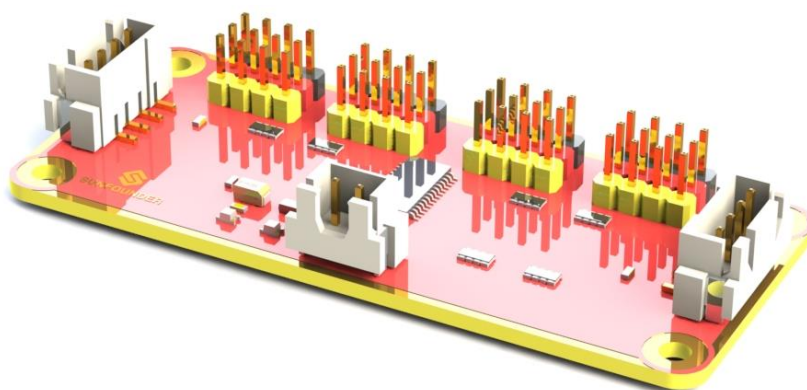
Robot HATS is a specially-designed HAT for a 40-pin Raspberry Pi and can work with Raspberry Pi model B+, 2 model B, and 3 model B. It supplies power to the Raspberry Pi from the GPIO ports. Thanks to the design of the ideal diode based on the rules of HATS, it can supply the Raspberry Pi via both the USB cable and the DC port thus protecting it from damaging the TF card caused by batteries running out of power. The PCF8591 is used as the ADC chip, with I2C communication, and the address 0x48.



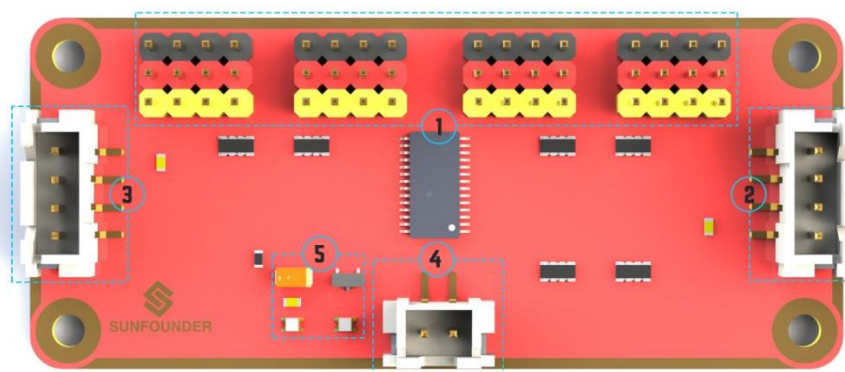
1. **Digital ports**: 3-wire digital sensor ports, signal voltage: 3.3V, VCC voltage: 3.3V.

2. **Analog ports:** 3-wire 4-channel 8-bit ADC sensor port, reference voltage: 3.3V, VCC voltage: 3.3V.
3. **I2C ports:** 3.3V I2C bus ports
4. **5V power output:** 5V power output to PWM driver.
5. **UART port:** 4-wire UART port, 5V VCC, perfectly working with SunFounder FTDI Serial to USB.
6. **TB6612 motor control ports:** includes 3.3V for the TB6612 chip, 5V for motors, and direction control of motors MA and MB; working with SunFounder TB6612 Motor driver.
7. **Switch:** power switch
8. **Power indicators:** indicating the voltage – 2 indicators on: >7.9V; 1 indicator on: 7.9V~7.4V; no indicator on: <7.4V. To protect the batteries, you're recommended to take them out for charge when there is no indicator on. The power indicators depend on the voltage measured by the simple comparator circuit; the detected voltage may be lower than normal depending on loads, so it is just for reference.
9. **Power port:** 5.5/2.1mm standard DC port, input voltage: 8.4~7.4V (limited operating voltage: 12V~6V).

PCA9865

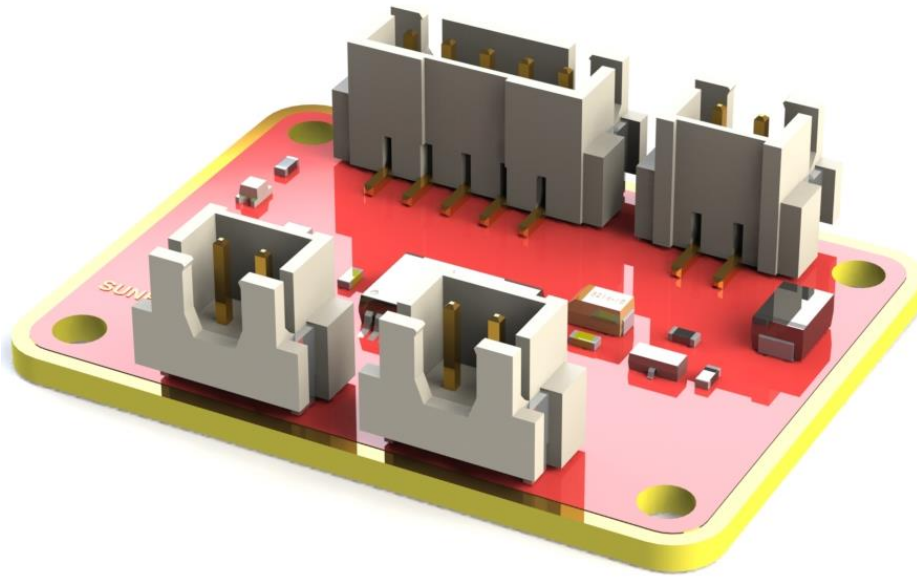


PCA9865 16-channel 12-bit I2C Bus PWM driver. It supports independent PWM output power and is easy to use 4-wire I2C port for connection in parallel, distinguished 3-color ports for PWM output.

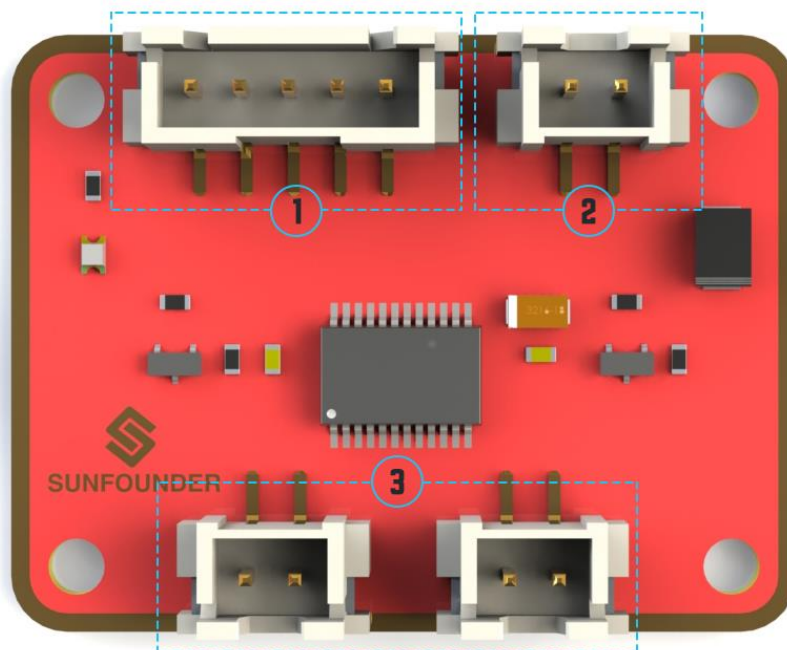


1. **PWM output ports:** 3-color ports, independent power PWM output port, connect to the servo directly.
- 2 & 3. **I2C port:** 4-wire I2C port, can be used in parallel. Compatible with 3.3V/5.5V
4. **PWM power input:** 12V max.
5. **LED:** power indicator for the chip and for the PWM power input.

TB6612



The TB6612 Motor Driver module is a low heat generation one and small packaged motor drive.



1. **Power and motor control port:** includes pins for supplying the chip and the motors and controlling the motors' direction

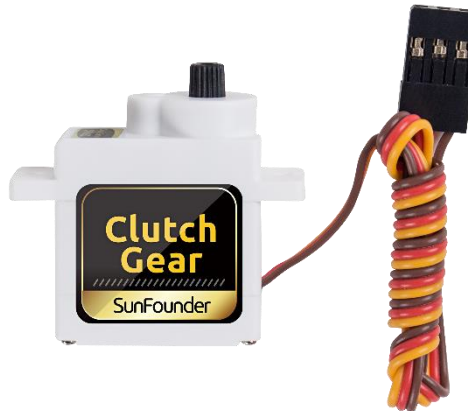
2. **PWM input for the motors:** PWM signal input for adjusting the speed of the two motors
3. **Motor output port:** output port for two motors

USB Webcam



This camera supports a wide angle of 120°, which provides a wide and clear vision, thus giving better experience when you're using it on the PiCar-V.

SunFounder SF006C Servo



Clutch gear digital servo with a DC core motor inside, After a certain load, the steering gear reducer will automatically clutch and protect the product from damage and normal load.

Function of the Performance:

Item	V = 4.8V	V = 6.0V
Consumption Current* (No Load)	$\leq 50\text{mA}$	$\leq 60\text{mA}$
Stall Current	$\leq 550\text{mA}$	$\leq 650\text{mA}$
Rated Torque	$\geq 0.6 \text{ kgf} \cdot \text{cm}$	$\geq 0.7 \text{ kgf} \cdot \text{cm}$
Max. Torque	$\geq 1.4 \text{ kgf.cm}$	$\geq 1.6 \text{ kgf.cm}$
No Load Speed	$\leq 0.14\text{sec}/60^\circ$	$\leq 0.12\text{sec}/60^\circ$

WiFi Adapter



The Ralink RT5370 Wireless Adapter is used in the car. Below are the features:

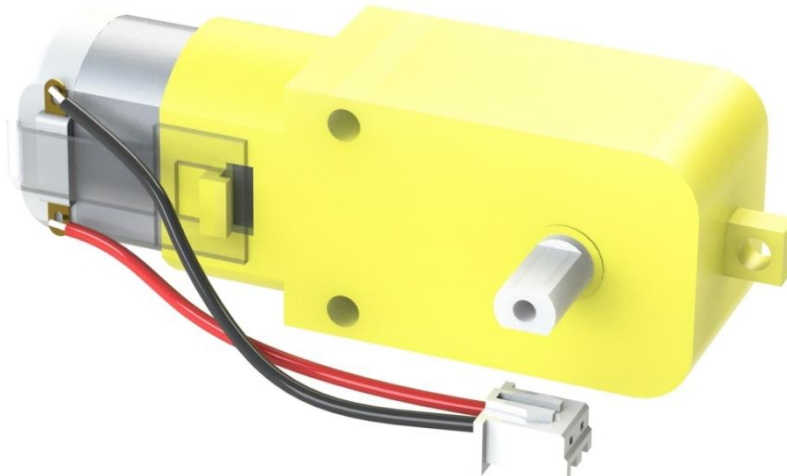
USB: XC1291

Network Connectivity: Wireless-Wi-Fi 802.11b, Wireless-Wi-Fi 802.11g, Wireless-Wi-Fi 802.11n

Applicable Network Type: Ethernet

Transmission Rate: 150Mbps

DC Gear Motor



It's a DC motor with a speed reducing gear train. See the parameters below:












Motor	Model	F130SA-11200-38V
	Rated Voltage	4.5V-6V
	No-load Current	≤80mA
	No-load Speed	10000±10%
Gear Reducer	Gear Ratio	1:48
	Speed (no-load)	≈200rpm (≈180rpm in test)
	Current	≤120mA

Code Explanation

There are two folders under `/home/pi`: *SunFounder_PiCar* for controlling motors and steering and *SunFounder_PiCar-V* for pan/tilt and wireless control. Here look into the code of PiCar-V.

Since too many contents and instructions are involved in the code, we will not cover every detail. For more knowledge about Python 2, Python 3, PyQt, Django, HTML, and CSS in the programs, you can visit related websites or buy books to learn by yourself. In the following part, we will go through the overall structure and the process in brief.

First, let's check the files in the code folder:

	<code>.git</code>	11/7/2016 2:02 PM	File folder	
	<code>client</code>	11/3/2016 4:32 PM	File folder	
	<code>datasheet</code>	9/13/2016 6:20 PM	File folder	
	<code>mjpg-streamer</code>	9/8/2016 2:14 PM	File folder	
	<code>remote_control</code>	11/3/2016 4:32 PM	File folder	
	<code>.gitignore</code>	9/8/2016 2:14 PM	Visual Studio Code	1 KB
	<code>i2cHelper.py</code>	8/25/2016 6:25 PM	Python File	4 KB
	<code>install_dependencies</code>	11/7/2016 10:49 AM	File	3 KB
	<code>LICENSE</code>	1/4/2016 3:34 PM	File	16 KB
	<code>README.md</code>	11/7/2016 1:59 PM	MD File	4 KB
	<code>show</code>	1/4/2016 3:34 PM	File	2 KB

- `.git` is a hidden directory to store all the information of a Git repository; it's generated automatically after you create the repository.
- `client` is to store the code of the PC client.
- `datasheet` stores the manual of some chips used in the hardware modules.
- `mjpg-streamer` is an open source streaming media library, through which the data of the camera is transferred.
- `Remote_control` is to store the code of the web server; it controls the motor and servo of the car based on the API request.
- `.gitignore` records the requests of file types to be ignored when the Github repository is synchronized.
- `i2cHelper.py` is a Python script written by Python 2 to configure and detect the I2C connection.
- `install_dependencies`, an executable bash script for simple installation and environment configuration.
- `LICENSE`, as the name suggests, is a text file of GNU V2 license.
- `README.md` and `show` record some information normally for statement and prompts.

Server Code

The server code is based on Django 1.10 (adapt to the latest release if needed). If you are interested in this, you can visit the Django website <https://www.djangoproject.com/>, or read *The Django Book* to learn more at <http://gsl.mit.edu/media/programs/south-africa-summer-2015/materials/djangobook.pdf>. Here we will just learn how the web server works. **Note:** The code may be updated irregularly on Github to fix bugs and release some functions update. So the code here is the initial version. You can view the updates in the Github repository at: https://github.com/sunfounder/SunFounder_PiCar-V.

Open the code directory and check the file by `ls`:

```
pi@raspberrypi:~ $ cd SunFounder_PiCar-V/
pi@raspberrypi:~/SunFounder_PiCar-V $ ls
datasheet      install_dependencies  mjpg-streamer  remote_control
client         i2cHelper.py         LICENSE        README.md       show
```

`remote_control` is the main code directory of the web server. Open the code directory by `cd remote_control`, and check the file by `ls`:

```
pi@raspberrypi:~/SunFounder_PiCar-V $ cd remote_control
pi@raspberrypi:~/SunFounder_PiCar-V/remote_control $ ls
db.sqlite3  manage.py  remote_control  start  static
```

- `db.sqlite3` is generated when you create a Django project.
- `manage.py` is the main program of the Django project and is generated when the project is created. It normally does not need to be modified.
- `remote_control` includes the main code files.
- `start` is a small script written to run `sudo python manage.py runserver 0.0.0.0:8000`, and servo installation with attribute `install` just for convenience.
- `static` is to store some static pictures on the web.

The Django web server normally runs `sudo python manage.py runserver` to start. The address `0.0.0.0:8000` means the listening address covers all the addresses on the LAN, and the port number is 8000. Here we will just focus on the code in `remote_control` folder. Go to the directory via `cd remote_control`:

```
pi@raspberrypi:~/SunFounder_PiCar-V/remote_control $ cd remote_control
pi@raspberrypi:~/SunFounder_PiCar-V/remote_control/remote_control $ ls
driver  __init__.py  settings.py  templates  urls.py  views.py  wsgi.py
```

- `driver` stores the drivers of the car.
- `__init__.py` is automatically generated when you create a Django project which is a standard necessary document of python package, just leave it there.
- `settings.py` is automatically generated and stores the related settings.

- `templates` is a Django app for storing the webs in the html format.
- `urls.py` is generated automatically to configure the URL to associate with the code.
- `views.py` is the code for page control which is associated by the URL. It calls the templates to show the page and the driver to control the car.
- `wsgi.py` is generated automatically and does not need changes. For more, please visit the official website of Django.

So here is how the code works: Run the main program `manage.py` which will be associated with `urls.py` automatically, to respond to the URL. When you run the web browser like the Chrome to visit `http://<rpi_ip_address>:<port>` or visit the configured API via the client, the `manage.py` will turn to `views.py` due to the association of the `urls.py`. Then the `views.py` processes this and returns the templates to the browser. In addition, it will call the `driver` package on the basis of the parameters set in the browser to control the car.

Now open the folder `driver` and check:

```
pi@raspberrypi:~/SunFounder_PiCar-V/remote_control/remote_control $ cd driver/
pi@raspberrypi:~/SunFounder_PiCar-V/remote_control/remote_control/driver $ ls
camera.py  config  __init__.py  stream.py
```

The `driver` folder mainly includes the driver modules for controlling the pan and tilt and camera streamer.

`camera.py` for controlling the pan-and-tilt.

`config` stores the calibration data.

`__init__.py` is the essential file of the package and you can just leave it alone.

`stream.py` is a video streaming service based on the MJPG-streamer.

Exit and open the folder `templates` to view:

```
pi@raspberrypi:~/SunFounder_PiCar-V/remote_control/remote_control/driver $ cd ../
pi@raspberrypi:~/SunFounder_PiCar-V/remote_control/remote_control $ cd templates/
admin.py  __init__.py  models.py  tests.py
apps.py  migrations  templates  views.py
```

This folder is created by the `manage.py startapp` just for calling the templates conveniently. Therefore, the files have not been changed except for the `templates`. Open the `templates` again:

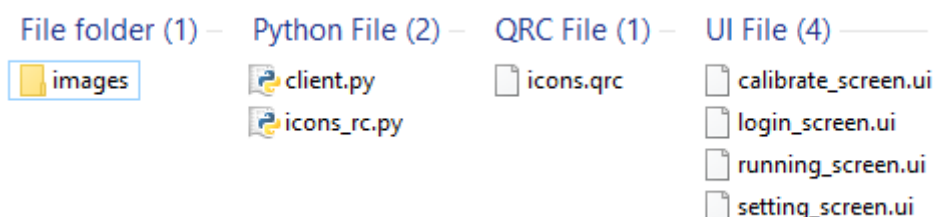
```
pi@raspberrypi:~/SunFounder_PiCar-V/remote_control/remote_control/templates $ cd
templates
pi@raspberrypi:~/SunFounder_PiCar-V/remote_control/remote_control/templates/templates $ ls
base.html  cali.html  run.html
```

There are three HTML files that also consist of layers. The low level `base.html` stores contents contained on each page such as the `<head>` of HTML, the overall layout, and contents of the home page by default. The surface layer: `cali.html` for calibration and `run.html` to control the car.

So that's the code of the server. Next we will come to the code of the client written by PyQt.

Client Code

In **Windows**, you can check the contents of client directory by grouping files by file type:



- `client.py`, is the main part of the code of the client.
- `icons_rc.py` is the python code converted from the UI resource file of Qt, which is the `icons.qrc` file beside, by `pyrcc5 -o icons_rc.py icons.qrc`.
- `icons.qrc` is the Qt resource file and restores the location of the resource pictures.
- `.ui` is the ui file, or the graphical user interface (GUI) file designed by the Qt tool. Open it by a text editor and you will see the code is written similar to an xml file.
- `images` stores the picture resources used in designing the UI (user interface). Do not rename or move the folder and the files in it, or else the corresponding content on the UI will be missing because it fails to find the graphic resource.
- `_pycache_` not shown in the figure above, is a cache folder automatically generated when you run a `.py` file of the client. If you've just downloaded it and not run yet, it's not generated.

The code of the client is written in Python 3, and the GUI is designed by the Qt tool. The open source library PyQt is used in the `client.py` to load the `.ui` file written by Qt into the python code.

For more information about PyQt, please visit the website:

<https://riverbankcomputing.com/software/pyqt/intro>

The interfaces go one by one like this:

1. The **login** page: enter the Raspberry Pi's IP address and click **Log in** go to the operation interface.
2. The **operation** interface will show the view captured by the camera. You can control the movement of the car on this page. You can click the return button, to go back to login, and the settings button to go to calibration interface.
3. The **calibration** interface: you can control and adjust the car by keyboard. Click **OK** to save the calibration data, and **Cancel** to return to the upper level interface.

Now after going through the code, you may wonder actually how to design the graphical interface and use PyQt. So let's take a closer look!

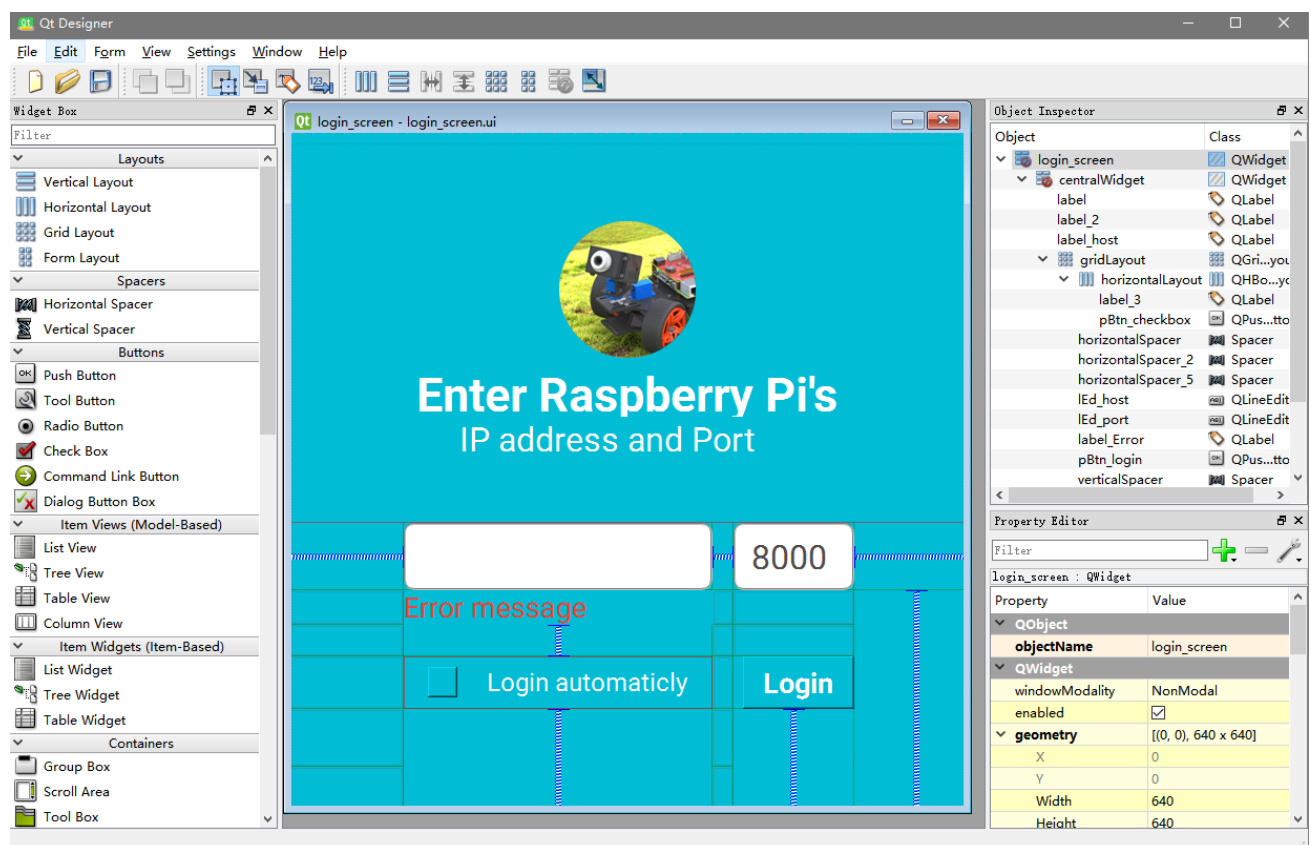
1. Designing the Interface

Let's start with the design interface.

In **Windows**, if you install the PyQt by the .exe package, select the option to install all, so the GUI design tools like the Qt Designer will be installed automatically. Or you can install the entire kit of Qt by yourself. Download the package (may need some setting and registration):

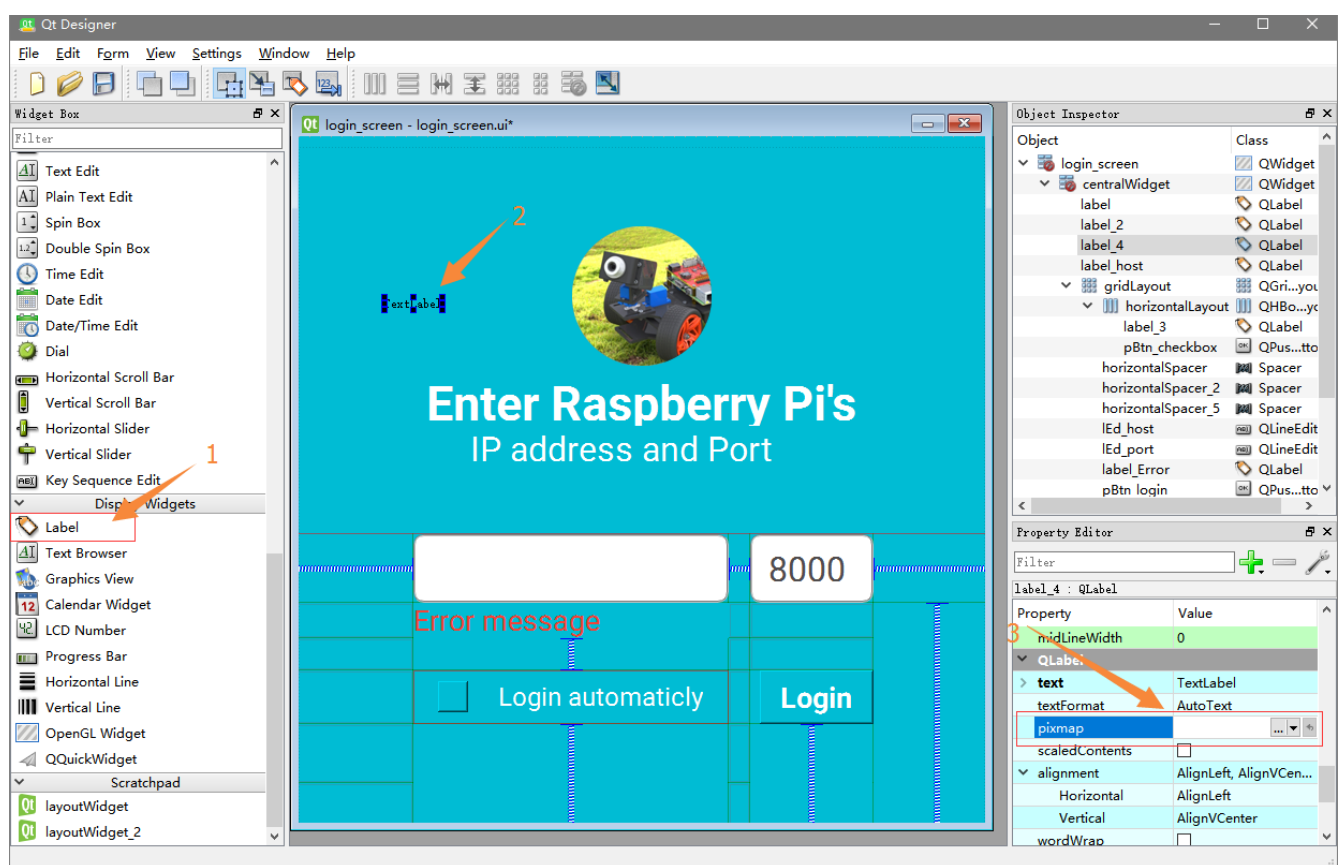
<https://www.qt.io/developers/>

Open the .ui file with a Qt tool after the installation is completed. The Qt designer in Qt tools is a GUI design tool.

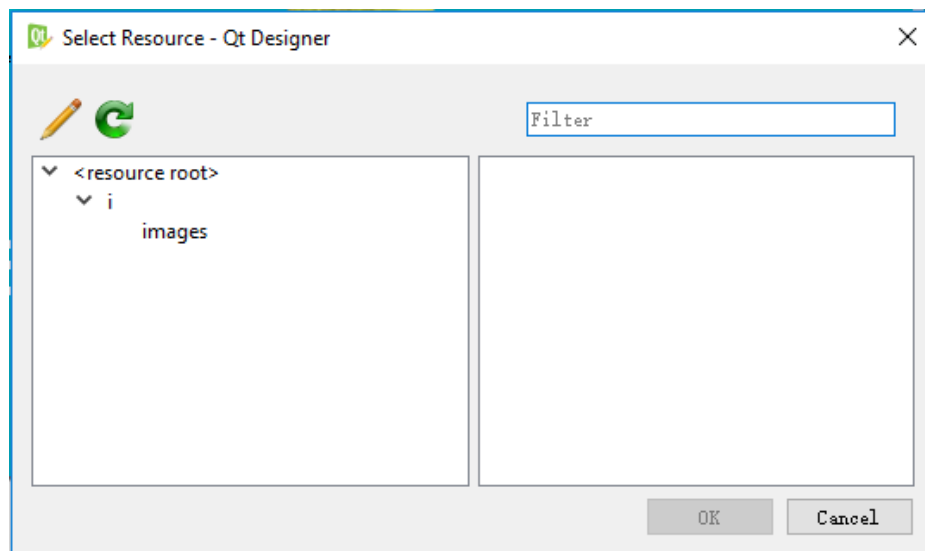
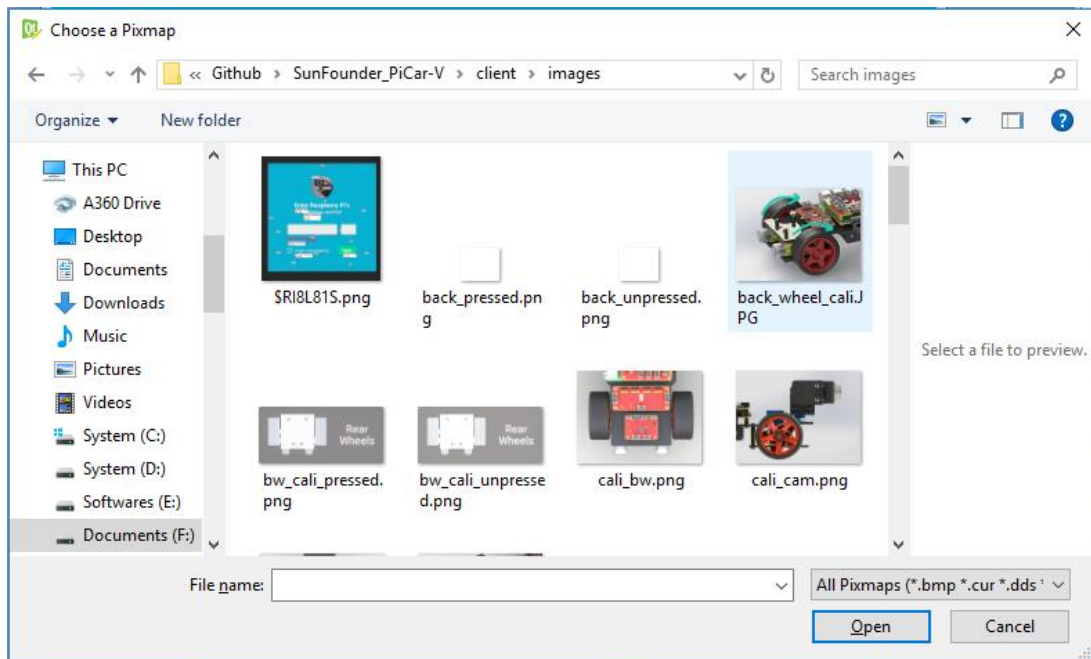


Design the first interface ([login](#)), with the GUI design tool. In the left column is the **Widget Box** section. You can just drag the components available here to the edit window in the center to see the effect directly. In the right column, on top is **Object Inspector**, where you can check the association and category of the added objects. Under this section, the property of the object is shown. You can set the attributes in this part, to differentiate all the widget objects.

Let's take the picture of the login interface as an example. Drag a **Label** from **Widget Box** to the picture, then click on the label and there is a square around, by which you can change its size and position. Then in the property part, select **pixmap** under **text** and click the box next to it, and on the pop-up drop-down list, select to read the picture from resource or from file.



If you choose to read from file, find the picture to be used, select it and click **OK**; to read from resource, create the resource file first. It is very simple – by the pencil icon (edit button) in the window of creating from resource. Click the icon, and select to create a resource and load it.



The setting of other components such as button, line edit and on, is just similar, though the properties are different. The name of the objects whose property has been changed will be marked bold automatically for convenience in next check. You can open those .ui files to see the property details that can be changed, or the widget type by objects in **Object Inspector**. After editing, you can select **Window -> Preview** in the Qt Designer to see the effect, or by the shortcut **Ctrl + R**.

Don't need to worry if you are not familiar with Qt because great details are given in the Qt document. If you don't know which property is for what use, you can check **Help** or the Qt assistant. And those sample sketches of Qt will help a lot for you to learn about Qt.

2. Programing the Functions

Now the interface design is finished. However, the interface files are just some static images and cannot be put into use. So the next step, write the client.py to implement all the interfaces with the corresponding functions as well as redirections. The PyQt is like this:

```
1
2 import sys
3 from PyQt5 import QtCore, QtWidgets, uic
4
5 qtCreatorFile = "" # Enter file here.
6
7 Ui_MainWindow, QtBaseClass = uic.loadUiType(qtCreatorFile)
8
9 class MyApp(QtGui.QMainWindow, Ui_MainWindow):
10     def __init__(self):
11         QtGui.QMainWindow.__init__(self)
12         Ui_MainWindow.__init__(self)
13         self.setupUi(self)
14
15 if __name__ == "__main__":
16     app = QtGui.QApplication(sys.argv)
17     window = MyApp()
18     window.show()
19     sys.exit(app.exec_())
```

In Line 5, enter the name of the ui file you designed in the quotation marks.

In Lin 7e, "uic" is the contents in the previously imported PyQt; the built-in function loads the ui files.

In Line 9-13, the class defined is just your GUI window. Define (def) the corresponding functions in the class and write the functions in them. The first `def __init__ (self)` is the initialization function which is a constructor in C++. It means when the instance is created, the program will run the code and initialize the instance.

Line 15-19 is the entry of the `__main__` function, where it starts to execute the whole program.

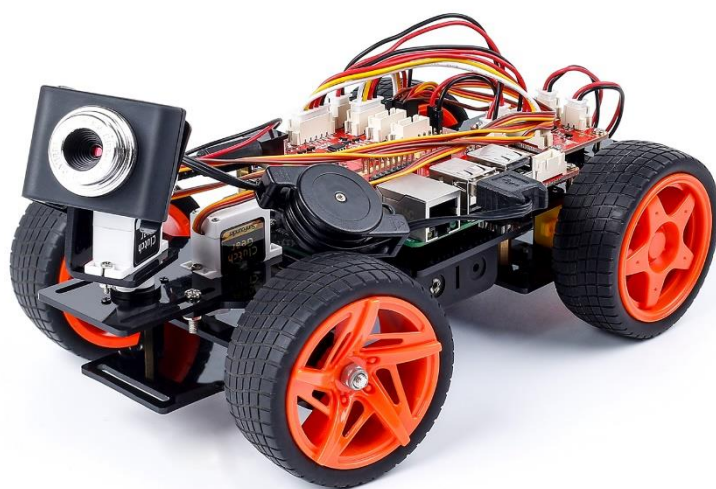
So here's how the program goes:

1. Run the `__main__` function and create an instance `app` of `QApplication`, which is the GUI application.
2. Create an instance `window` of the `MyApp` class type. After the creation is done, run the `__init__` initialization function of the class.
3. The window instance is displayed.
4. Wait for a while. If the instance `app` of the GUI application sends an exit signal `app.exec_()`, then enter `sys.exit()` to exit Python.

So that's all for the code. Here we've only looked at the entry-level usage. For more detailed code explanation, go to the *client.py* file. There you can see very detailed comments.

Advanced

Now the car is running! You may try control it or view the image captured in Windows, Linux, or Mac OS X.



So what to do next?

Yes, sensors! There are 8 digital and 4 analog channels on the Robot HATS, as well as two I2C ports. Plus the 3mm slot on the acrylic plate in the car head, it enables the connection of various sensors, making the car smarter and more intriguing as you want!

And, coding! If you know programming and are capable of compiling code, you may change the code provided for the kit to make the car more fabulous. For instance, you can find the code that controls the front and back wheels in the folder *SunFounder_PiCar*, downloaded to the directory */home/pi* if you've installed the program by *install-dependencies*.

You may check what's in the file:

```
cd ~/SunFounder_PiCar
ls
```

After you've changed the code, you can go to the page of the Github repository for the car, and fork it to help improve it! Also you're quite welcome to post issues and request a pull on our Github page:

PiCar-V:

https://github.com/sunfounder/SunFouner_PiCar-V

PiCar driver:

https://github.com/sunfounder/SunFounder_PiCar

And of course, if you have any questions or ideas, please feel free to email us at service@sunfounder.com.

Appendix: Function of the Server Installation Scripts

You may have a question: What do the installation scripts do when we install the server on the Raspberry Pi by them? So here let's check the detailed steps.

1. Install pip

```
apt-get install python-pip
```

2. Use pip to install django

```
sudo pip install django
```

3. Install i2c-tools and python-smbus

```
sudo apt-get install i2c-tools python-smbus -y
```

4. Install PiCar drivers

```
cd ~/
```

```
git clone --recursive https://github.com/sunfounder/SunFounder\_PiCar.git
```

```
cd SunFounder_PiCar
```

```
sudo python setup.py install
```

5. Download source code

```
cd ~/
```

```
git clone https://github.com/sunfounder/SunFounder_PiCar-V.git
```

6. Copy the MJPG-Streamer file to system directory

```
cd ~/SunFounder_PiCar-V
```

```
sudo cp mjpg-streamer/mjpg_streamer /usr/local/bin
```

```
sudo cp mjpg-streamer/output_http.so /usr/local/lib/
```

```
sudo cp mjpg-streamer/input_file.so /usr/local/lib/
```

```
sudo cp mjpg-streamer/input_uvc.so /usr/local/lib/
```

```
sudo cp -R mjpg-streamer/www /usr/local/www
```

7. Export paths

```
export LD_LIBRARY_PATH=/usr/local/lib/ >> ~/.bashrc
```

```
export LD_LIBRARY_PATH=/usr/local/lib/ >> ~/.profile
```

```
source ~/.bashrc
```

8. Enable I2C1

Edit the file /boot/config.txt:

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

Add the line in the end:

```
dtoverlay=i2c_arm=on
```

9. Reboot

```
sudo rebootss
```