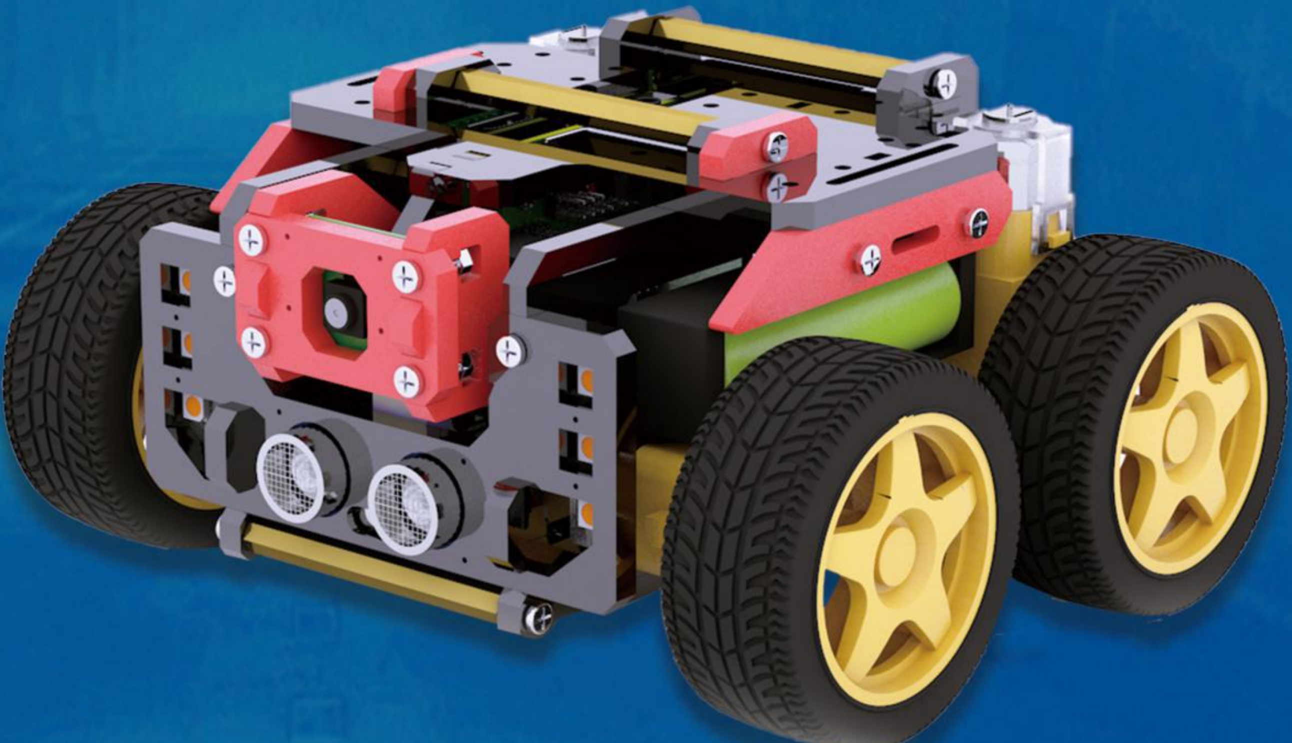
Adeept®

AWR

Adeept Wheeled Robot

Adeept Smart Car Robot Kit for Raspberry Pi



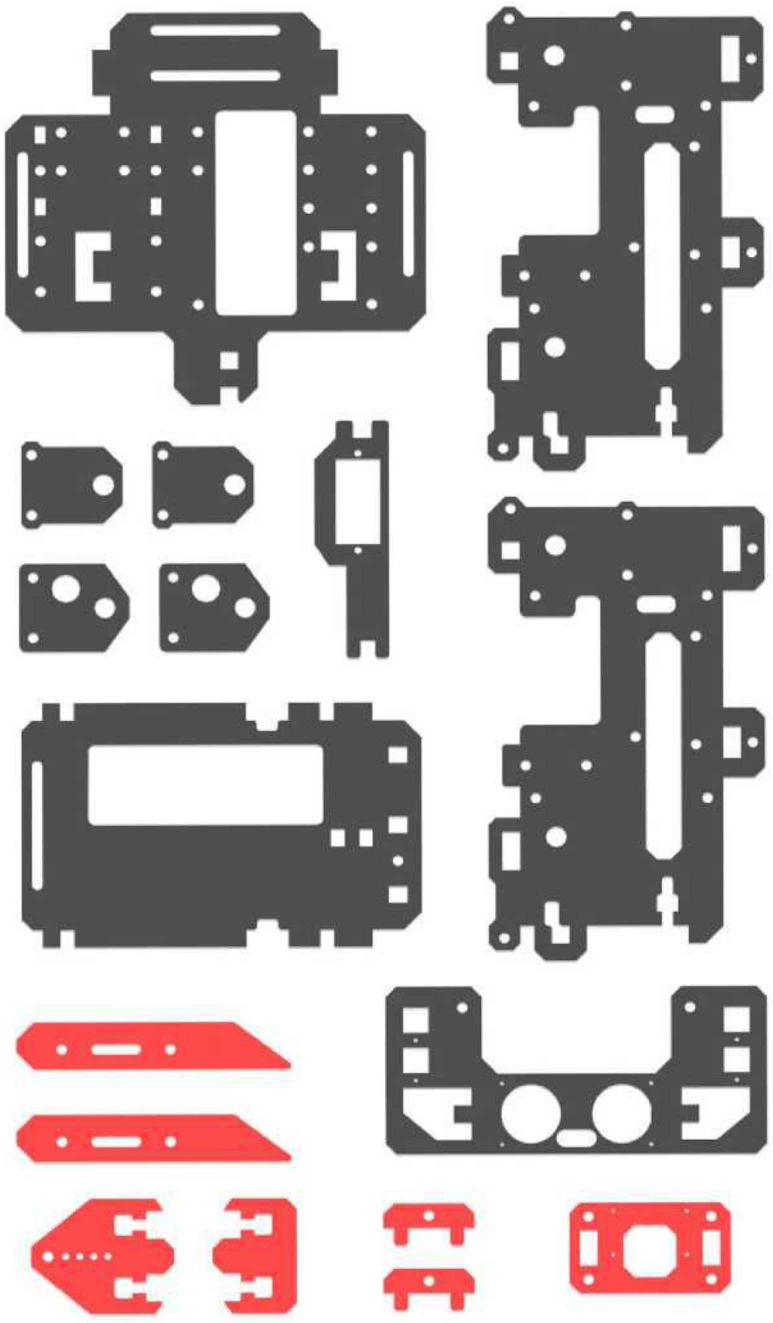
www.adeept.com

❖ Resources Links

RobotName: Adeept_AWR
RobotURL: https://github.com/adeept/Adeept_AWR
RobotGit: https://github.com/adeept/Adeept_AWR.git
[Official Raspberry Pi website] https://www.raspberrypi.org/downloads/
[Official website] https://www.adeept.com/
[GitHub] https://github.com/adeept/Adeept_AWR
[Image file and Documentation for structure assembly] https://www.adeept.com/learn/detail-35.html

Components List

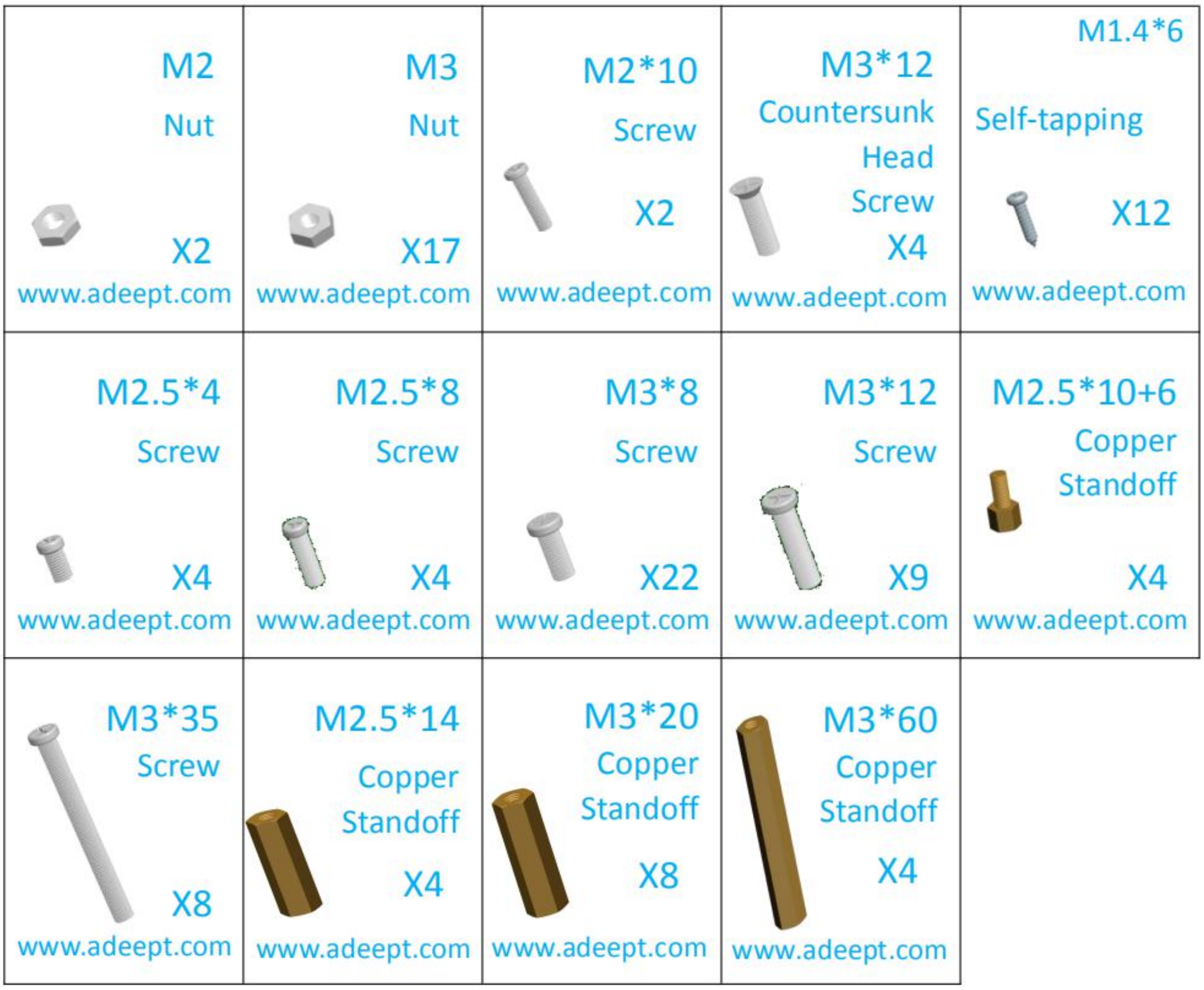
Acrylic Plates



The acrylic plates are fragile, so please be careful when assembling them in case of breaking. The acrylic plate is covered with a layer of protective film. You need to remove it first. Some holes in the acrylic may have residues, so you need to clean them before the use.

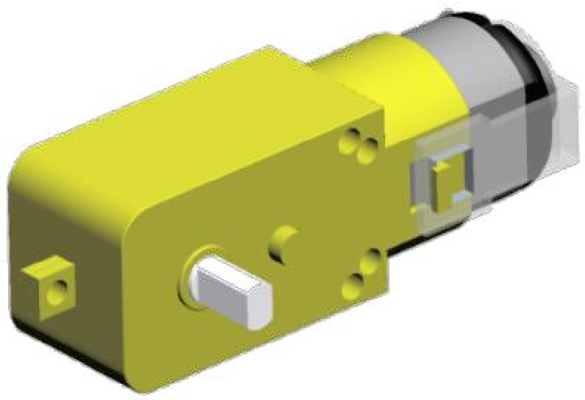
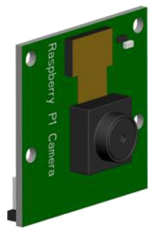
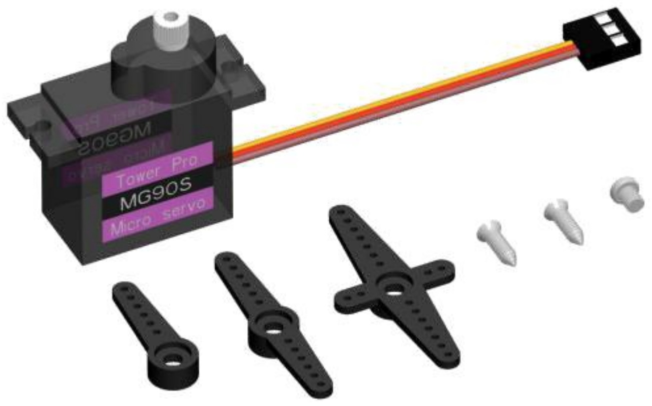
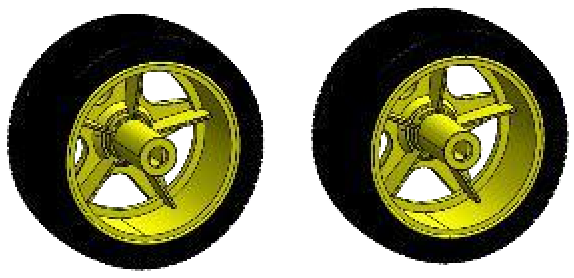
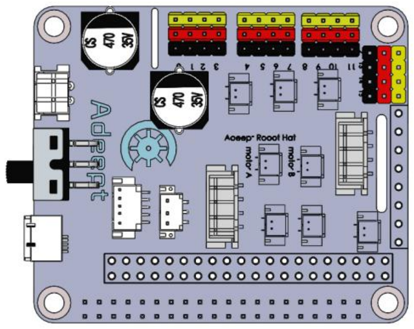
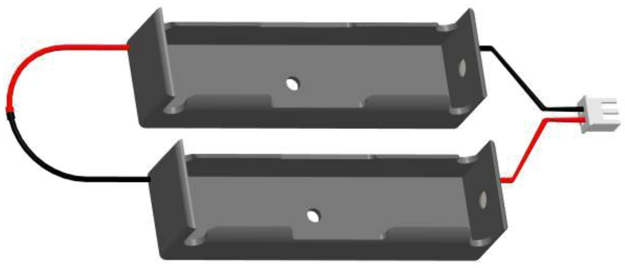
Adeept

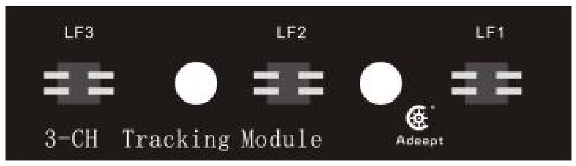
Machinery Parts



M2 Nut X2 www.adeept.com	M3 Nut X17 www.adeept.com	M2*10 Screw X2 www.adeept.com	M3*12 Countersunk Head Screw X4 www.adeept.com	M1.4*6 Self-tapping X12 www.adeept.com
M2.5*4 Screw X4 www.adeept.com	M2.5*8 Screw X4 www.adeept.com	M3*8 Screw X22 www.adeept.com	M3*12 Screw X9 www.adeept.com	M2.5*10+6 Copper Standoff X4 www.adeept.com
M3*35 Screw X8 www.adeept.com	M2.5*14 Copper Standoff X4 www.adeept.com	M3*20 Copper Standoff X8 www.adeept.com	M3*60 Copper Standoff X4 www.adeept.com	

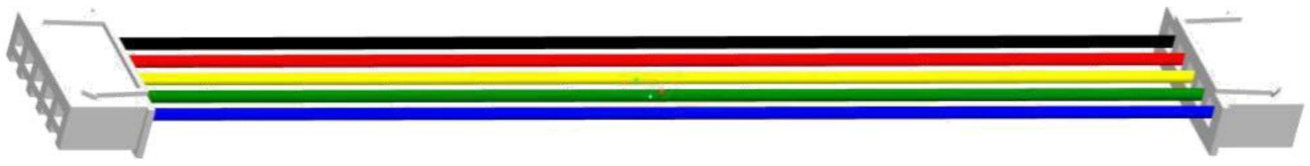
Electronic Parts

Motor X4 	Raspberry Pi Camera X1 
Servo x1 	Wheel x4 
ROBOT HAT X1 	18650 Battery Holder Set X1 
CAR LIGHT X2 www.adeept.com Sharing Perfects Innovation	3-Pin Wire X2

Adeept Ultrasonic Module X1	3 Tracking Module X1
HC-SR04 Vcc Trig Echo Gnd	

4 PIN WIRE X1

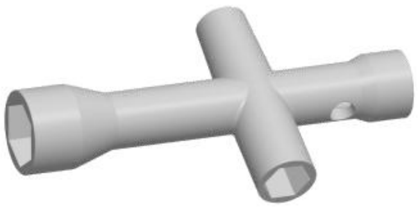
5-Pin Wire X1

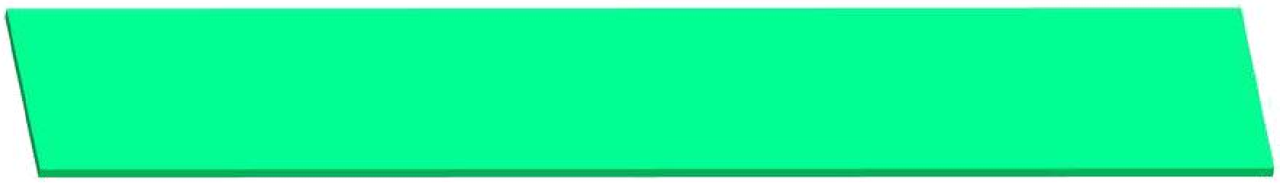


3-Pin Wire X2

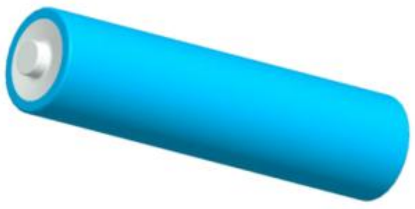
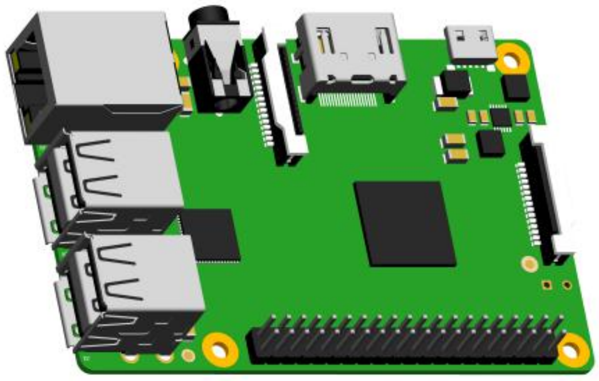
Raspberry P1 Camera Ribbon X1

Tools

Hex Wrench-2.0mm X1	Cross Screwdriver X1	Cross Socket Wrench X1
		

Large Cross-head Screwdriver X1
Winding Pipe X1
Ribbon X1 

Self-prepared Parts

18650 Battery X2	Raspberry Pi X1
	

Requirements for 18650 lithium battery: 18650 lithium battery is required for normal operation of the robot, and the current output is above 4A.

Content

❖ 1. Premise

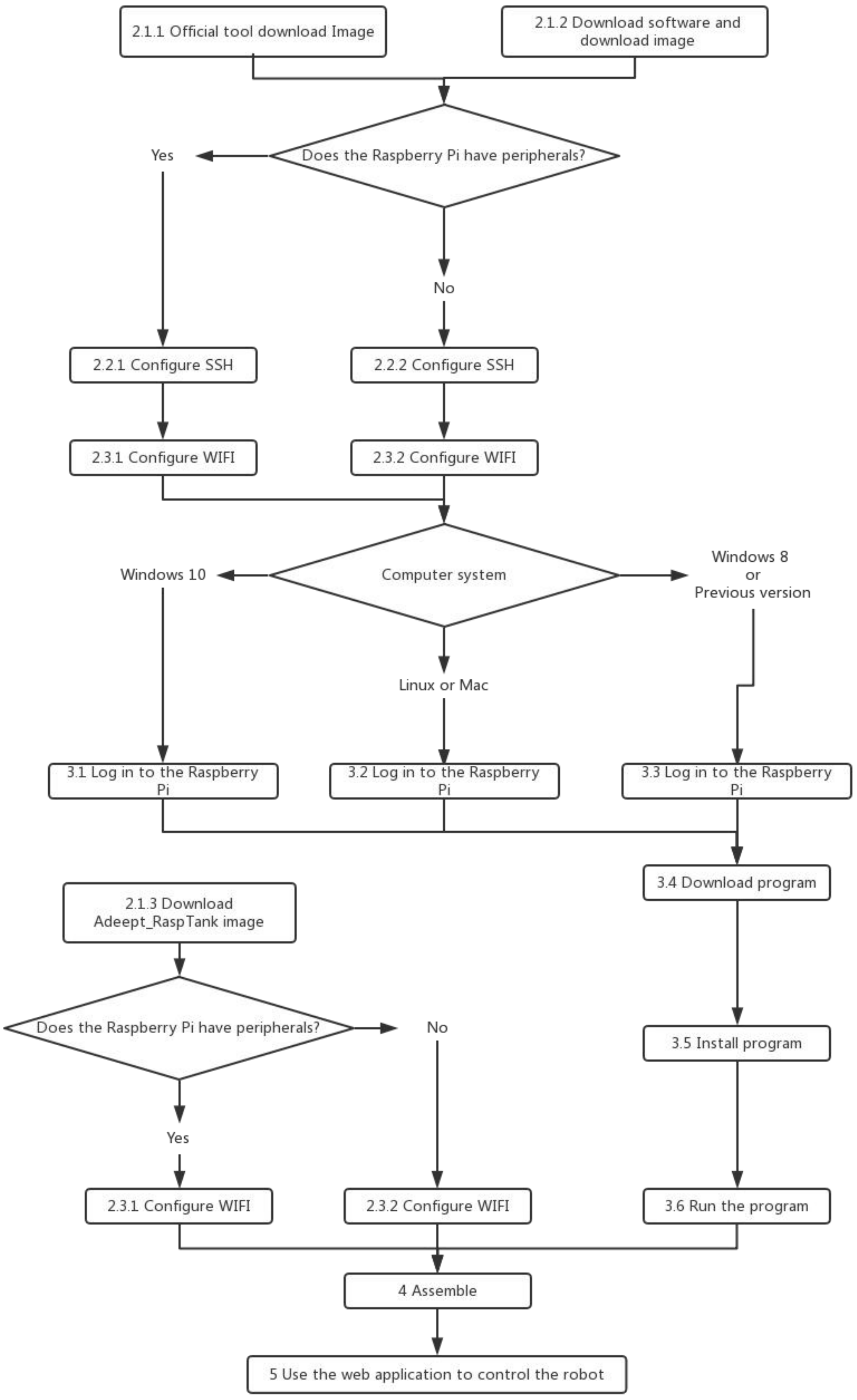
1.1 STEAM and Raspberry Pi

STEAM stands for Science, Technology, Engineering, Arts and Mathematics. It's a type of trans disciplinary education idea focused on practice. As a board designed for computer programming education, Raspberry Pi has lots of advantages over other robot development boards. Therefore, Raspberry Pi is used for function control of the robot.

1.2 About The Documentation

This documentation is for software installation and operation guide for the Python robot product. It describes every detail of the whole process of fulfilling the robot project by Python and Raspberry Pi from scratch as well as some precautions. Hope you can get started with the Raspberry Pi robot on Python and make more creations with this documentation.

According to the different situations of different users, there will be some changes in the process of this document, you can refer to the following process:


2.1.1 Official tool download Image
2.1.2 Download software and download image
Does the Raspberry Pi have peripherals?
Yes
No
2.2.1 Configure SSH
2.2.2 Configure SSH
2.3.1 Configure WIFI
2.3.2 Configure WIFI
Computer system
Windows 10
Windows 8 or Previous version
Linux or Mac
3.1 Log in to the Raspberry Pi
3.2 Log in to the Raspberry Pi
3.3 Log in to the Raspberry Pi
3.4 Download program
2.1.3 Download Adeept_RaspTank image
Does the Raspberry Pi have peripherals?
No
Yes
3.5 Install program
2.3.1 Configure WIFI
2.3.2 Configure WIFI
3.6 Run the program
4 Assemble
5 Use the web application to control the robot

❖ 2. Raspberry Pi System Installation and Development Environment Establishment

2.1 Install An Operating System for The Raspberry Pi

2.1.1 Method A: Write 'Raspbian' to The SD Card by Raspberry Pi Imager

Raspberry Pi Imager is an image writing tool to SD card developed by the Raspberry Pi Organization. It comes with many versions working on different systems and it's quite easy to use; all you need is choose the operating system and SD card, Raspberry Pi Imager will download the corresponding image file for the system and install it to the SD card.

Step-by-Step Overview

1. Prepare an SD card (16G or larger) and an SD card reader

2. Download the `Raspberry Pi Imager` on the official website

- [Raspberry Pi Imager for Windows]

- [Raspberry Pi Imager for macOS]

- [Raspberry Pi Imager for Ubuntu]

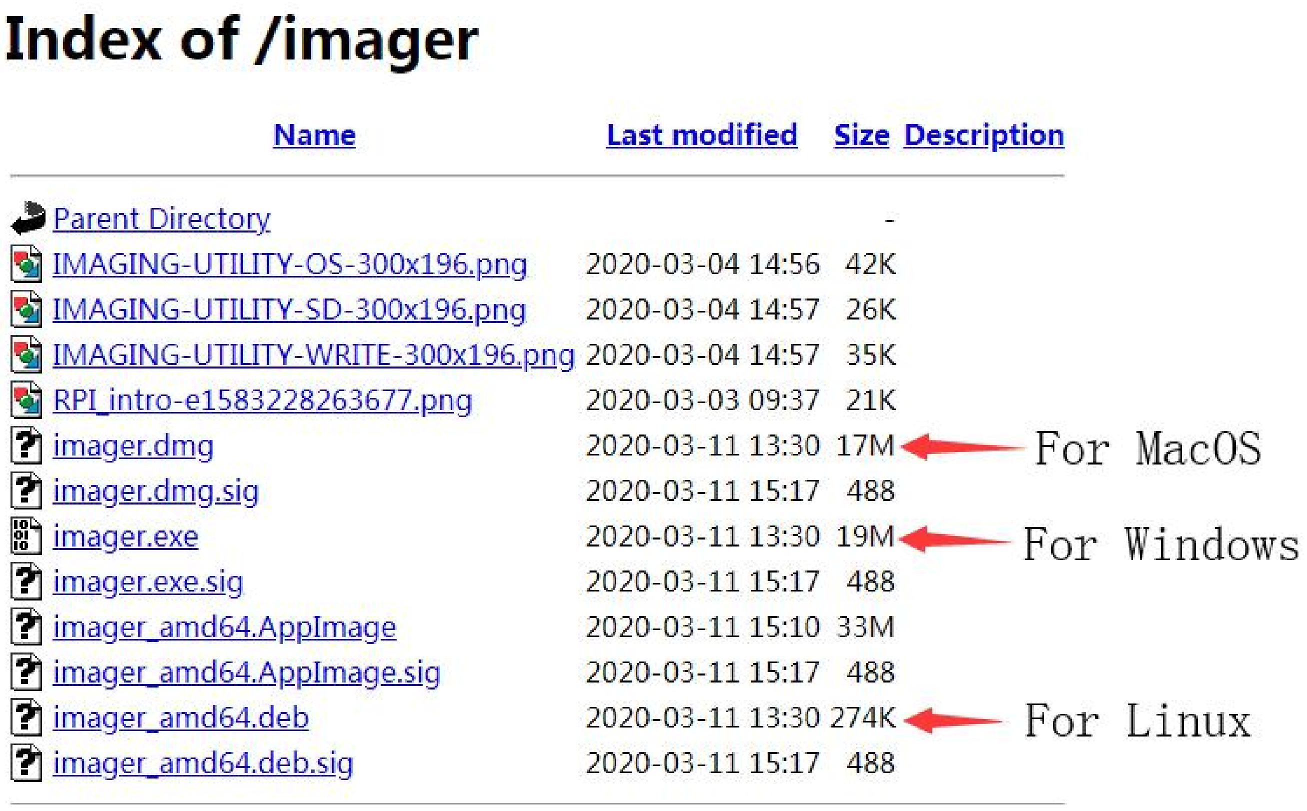
3. Install the `Raspberry Pi Imager`

4. Write the operating system for Raspberry Pi to the SD card with `Raspberry Pi Imager` `Raspbian Full - A port of Debian with desktop and recommended application`

5. Leave the SD card connected after writing is completed, we'll use for configuring SSH and WiFi connection later.

Detailed Steps:

●Open a web browser on your computer, go to the Raspberry Pi website [Official Raspberry Pi website], find and download the Raspberry Pi Imager for your computer OS, or click on the links above for the corresponding system to directly download and install.



Index of /imager

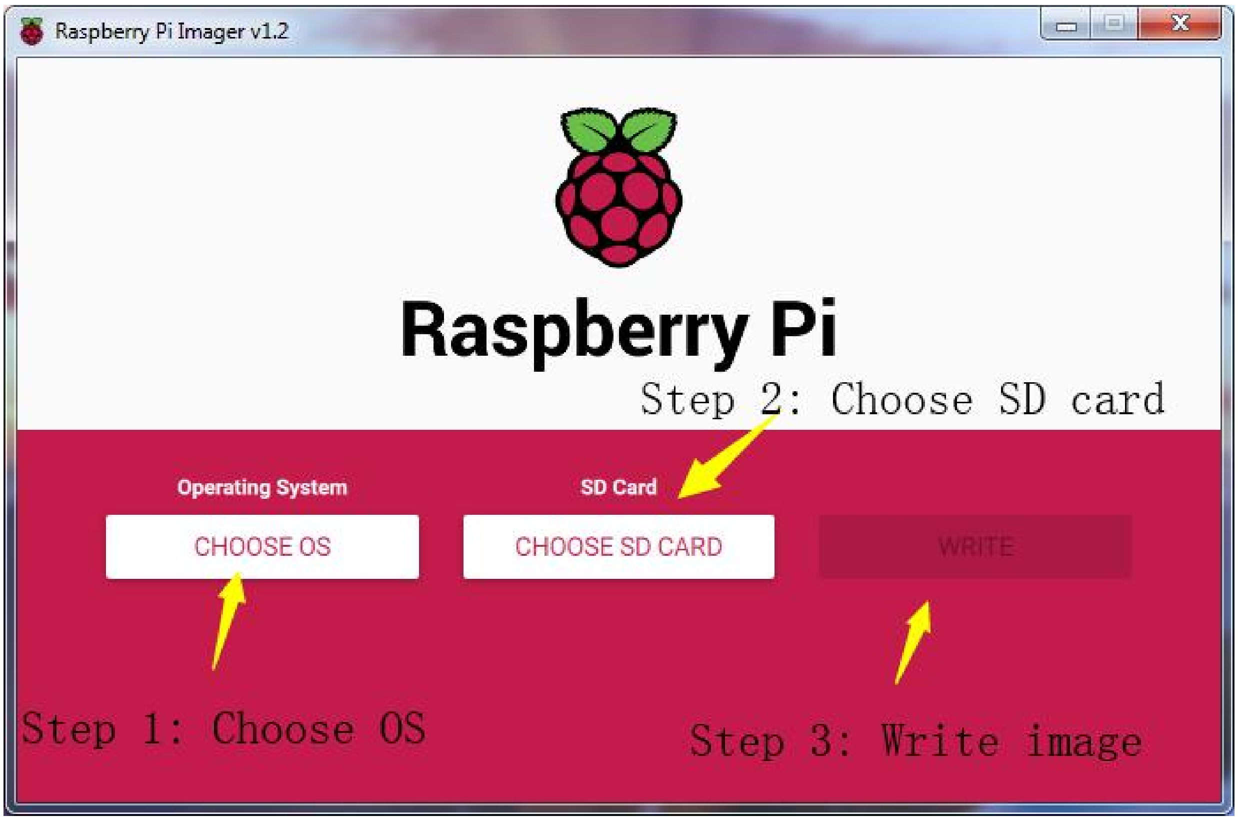
Name	Last modified	Size	Description
Parent Directory		-	
IMAGING-UTILITY-OS-300x196.png	2020-03-04 14:56	42K	
IMAGING-UTILITY-SD-300x196.png	2020-03-04 14:57	26K	
IMAGING-UTILITY-WRITE-300x196.png	2020-03-04 14:57	35K	
RPI_intro-e1583228263677.png	2020-03-03 09:37	21K	
imager.dmg	2020-03-11 13:30	17M	
imager.dmg.sig	2020-03-11 15:17	488	
imager.exe	2020-03-11 13:30	19M	
imager.exe.sig	2020-03-11 15:17	488	
imager_amd64.AppImage	2020-03-11 15:10	33M	
imager_amd64.AppImage.sig	2020-03-11 15:17	488	
imager_amd64.deb	2020-03-11 13:30	274K	
imager_amd64.deb.sig	2020-03-11 15:17	488	

●Insert the SD card into the card reader, connect the card reader with your computer.

●Run the Raspberry Pi Imager, select CHOOSE OS -> Raspbian(other) -> Raspbian Full - A port of Debian with desktop and recommended applications.

●Click on CHOOSE SD CARD for the SD card to write the Raspbian Full, please be noted that the image writing will automatically delete all files on the SD card if any.

●Click on WRITE, wait for the writing. The Raspberry Pi Imager needs to download the Raspbian image file during the process. You can download the file following the step in 2.1.2.



●Do not remove the SD card connected after writing is completed, we'll use for configuring SSH and WiFi connection later. Otherwise, if you remove the card, insert it into the Raspberry Pi and boot, WiFi configuration without any peripherals may fail in the following process.

2.1.2 Method B: Download The Image File Raspbian and Write It to The SD Card Manually

●Since the image file is downloaded with Raspberry Pi Imager in 2.1.1, it can take a long time due to a slow network in some places. You may then manually download the image file Raspbian and write it to the SD card with th Raspberry Pi Imager.

Step-by-Step Overview

1. Prepare an SD card (16G or larger) and an SD card reader

2. Download the `Raspberry Pi Imager` on the official website [Official Raspberry Pi Website]

- [Raspberry Pi Imager for Windows]

- [Raspberry Pi Imager for macOS]

- [Raspberry Pi Imager for Ubuntu]

3. Install the `Raspberry Pi Imager`

4. Download the image file `Raspbian`

- Torrent file:

[Raspbian-Raspbian Buster with desktop and recommended software]

-Zip file: [Raspbian - Raspbian Buster with desktop and recommended software]

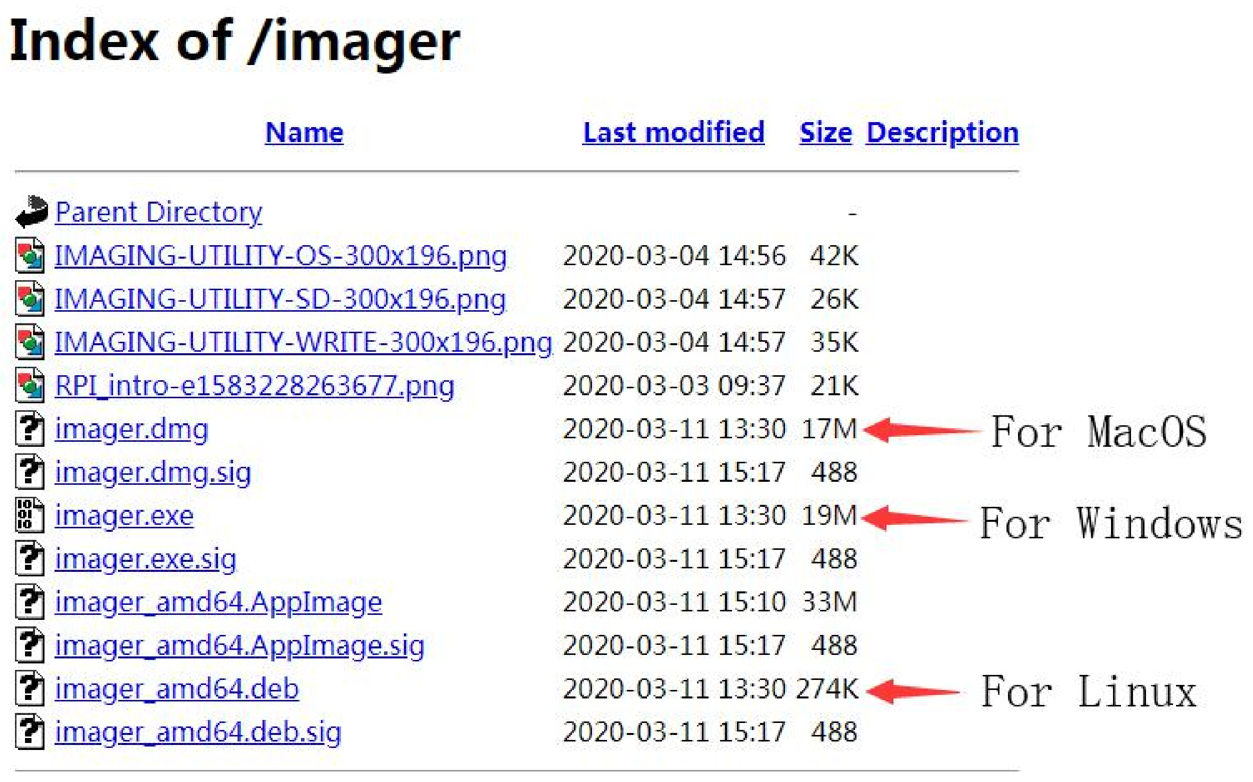
5. Unzip the file, be noted that the path should be in English for the `.img` file extracted, no special characters allowed.

6. Write the image file `Raspbian` downloaded to the SD card with `Raspberry Pi Imager`

7. Leave the SD card connected after writing is completed, we'll use for configuring SSH and WiFi connection later.

Detailed Steps:

●Open a web browser on your computer, go to the Raspberry Pi website[Official Raspberry Pi website], find and download the Raspberry Pi Imager for your computer OS, or click on the links above for the corresponding system to directly download and install.



Index of /imager

Name	Last modified	Size	Description
Parent Directory		-	
IMAGING-UTILITY-OS-300x196.png	2020-03-04 14:56	42K	
IMAGING-UTILITY-SD-300x196.png	2020-03-04 14:57	26K	
IMAGING-UTILITY-WRITE-300x196.png	2020-03-04 14:57	35K	
RPI_intro-e1583228263677.png	2020-03-03 09:37	21K	
imager.dmg	2020-03-11 13:30	17M	
imager.dmg.sig	2020-03-11 15:17	488	
imager.exe	2020-03-11 13:30	19M	
imager.exe.sig	2020-03-11 15:17	488	
imager_amd64.AppImage	2020-03-11 15:10	33M	
imager_amd64.AppImage.sig	2020-03-11 15:17	488	
imager_amd64.deb	2020-03-11 13:30	274K	
imager_amd64.deb.sig	2020-03-11 15:17	488	

●On the Raspberry Pi website [Official Raspberry Pi website], select through Downloads -> Raspbian -> Raspbian Buster with desktop and recommended software, and click on the torrent or zip file to download. Unzip the file after download, be noted that the path should be in English for the .img file extracted, no special characters allowed; otherwise Raspberry Pi Imager may not open the .img file. It's recommended to save the .img file to the root directory of the C:\ or D:\ disk, but do not save .img on the SD card.

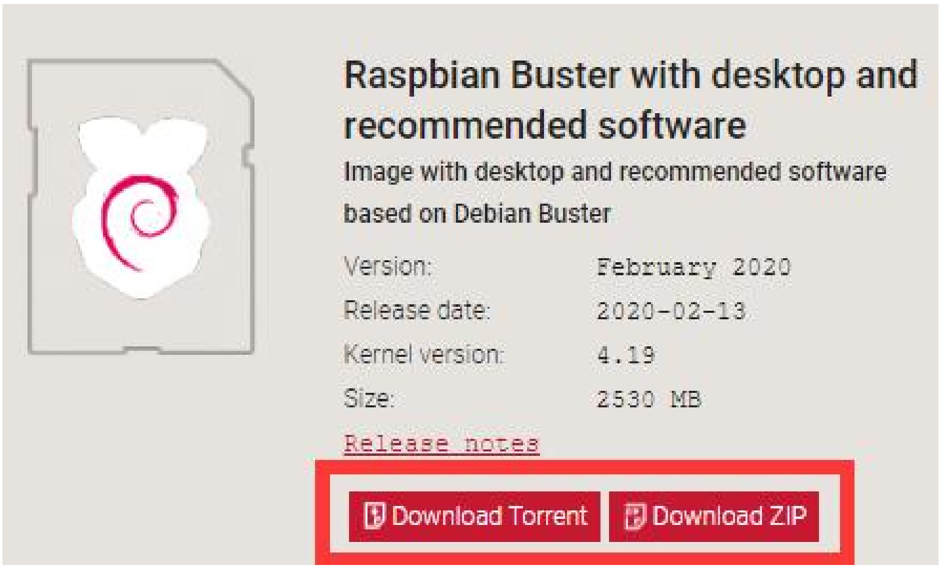

Raspbian Buster with desktop and recommended software

Image with desktop and recommended software based on Debian Buster

Version: February 2020

Release date: 2020-02-13

Kernel version: 4.19

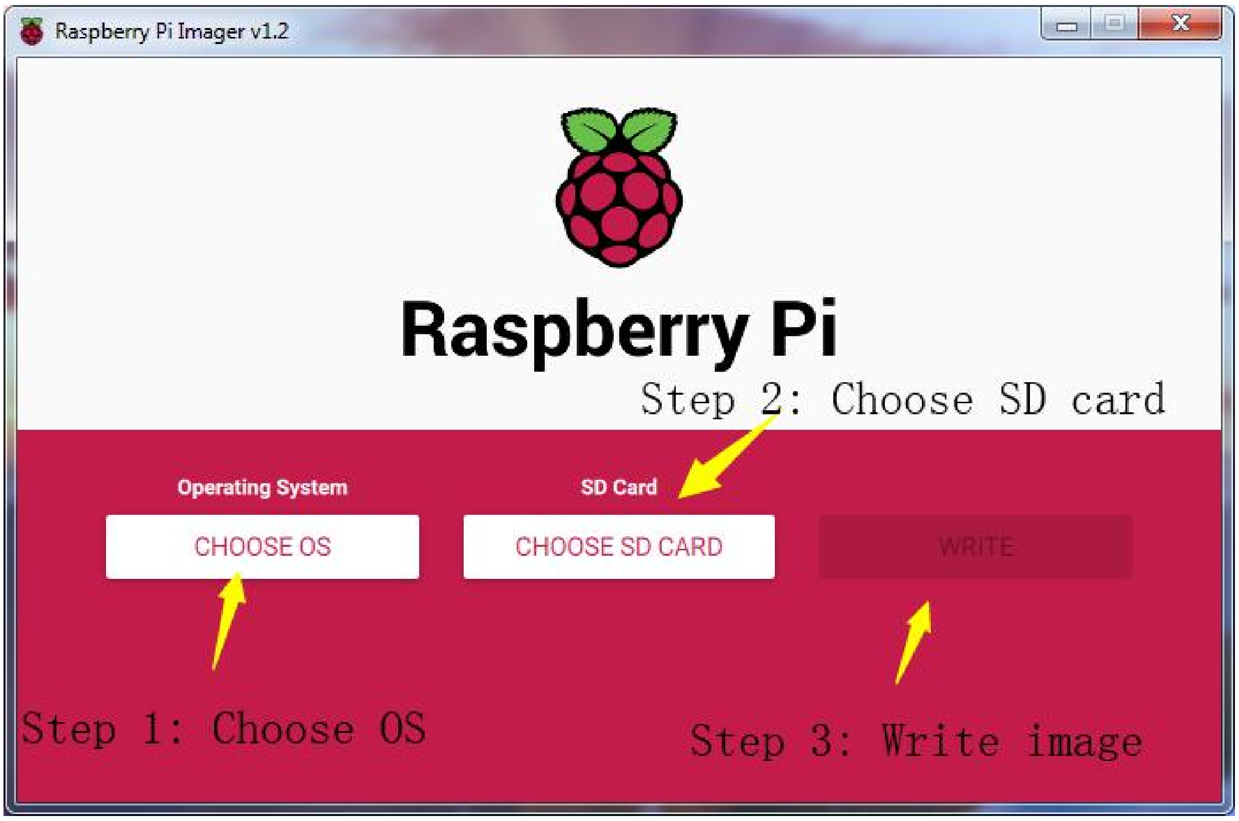
Size: 2530 MB

●Insert the SD card into the card reader, connect the card reader and your computer.

●Run the Raspberry Pi Imager, select CHOOSE OS, and then Use custom to find the .img extracted, click Open.

●Select CHOOSE SD CARD for the SD card to write the Raspbian, please be noted that the image writing will automatically delete all files on the SD card if any.

●Click on WRITE, wait for the writing.



●Do not remove the SD card connected after writing is completed, we'll use for configuring SSH and WiFi connection later. Otherwise, if you remove the card, insert it into the Raspberry Pi and boot it up, WiFi configuration without any peripherals may fail in the following process.

2.1.3 Method C: Manually Download The Image File Provided by Us and Write It to The SD Card (Not Recommended)

● The Raspbian image file downloaded in **2.1.1** and **2.1.2** is the official source with some preinstalled software. To operate the robot, you may need many dependent libraries. Though we provide the simple script to install them (see details later), failure can happen during installation if the library is not the latest version. Therefore, despite we provide the downloading of the Raspbian image file, it may happen that our image file and the dependent libraries are not most updated versions. Please only use when you encounter the most troublesome situation.

●Step-by-Step Overview

1. Prepare an SD card (16G or larger) and an SD card reader
2. Download the `Raspberry Pi Imager` from the official website [Official Raspberry Pi website]

- [Raspberry Pi Imager for Windows]
- [Raspberry Pi Imager for macOS]
- [Raspberry Pi Imager for Ubuntu]

3. Install the `Raspberry Pi Imager`

4. Download the image file `Adeept_AWR`

- [Image file for the Adeept_AWR Robot]

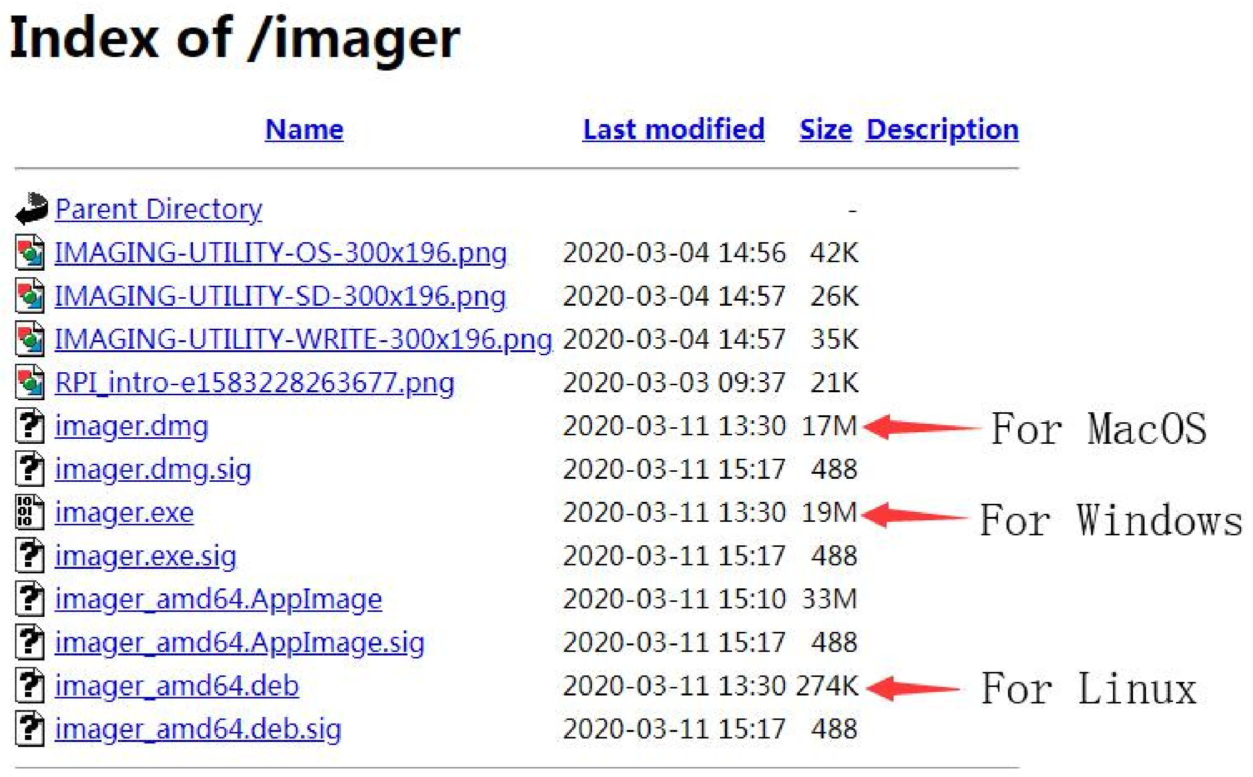
5. Unzip the file, be noted that the path should be in English for the `.img` file extracted, no special characters allowed.

6. Write the image file `Raspbian` downloaded to the SD card with `Raspberry Pi Imager`

7. Leave the SD card connected after writing is completed, we'll use for configuring SSH and WiFi connection later.

Detailed Steps:

●Open a web browser on your computer, go to the Raspberry Pi website [Official Raspberry Pi website], find and download the Raspberry Pi Imager for your computer OS, or click on the links above for the corresponding system to directly download and install.



Index of /imager

Name	Last modified	Size	Description
Parent Directory		-	
IMAGING-UTILITY-OS-300x196.png	2020-03-04 14:56	42K	
IMAGING-UTILITY-SD-300x196.png	2020-03-04 14:57	26K	
IMAGING-UTILITY-WRITE-300x196.png	2020-03-04 14:57	35K	
RPI_intro-e1583228263677.png	2020-03-03 09:37	21K	
imager.dmg	2020-03-11 13:30	17M	
imager.dmg.sig	2020-03-11 15:17	488	
imager.exe	2020-03-11 13:30	19M	
imager.exe.sig	2020-03-11 15:17	488	
imager_amd64.AppImage	2020-03-11 15:10	33M	
imager_amd64.AppImage.sig	2020-03-11 15:17	488	
imager_amd64.deb	2020-03-11 13:30	274K	
imager_amd64.deb.sig	2020-03-11 15:17	488	

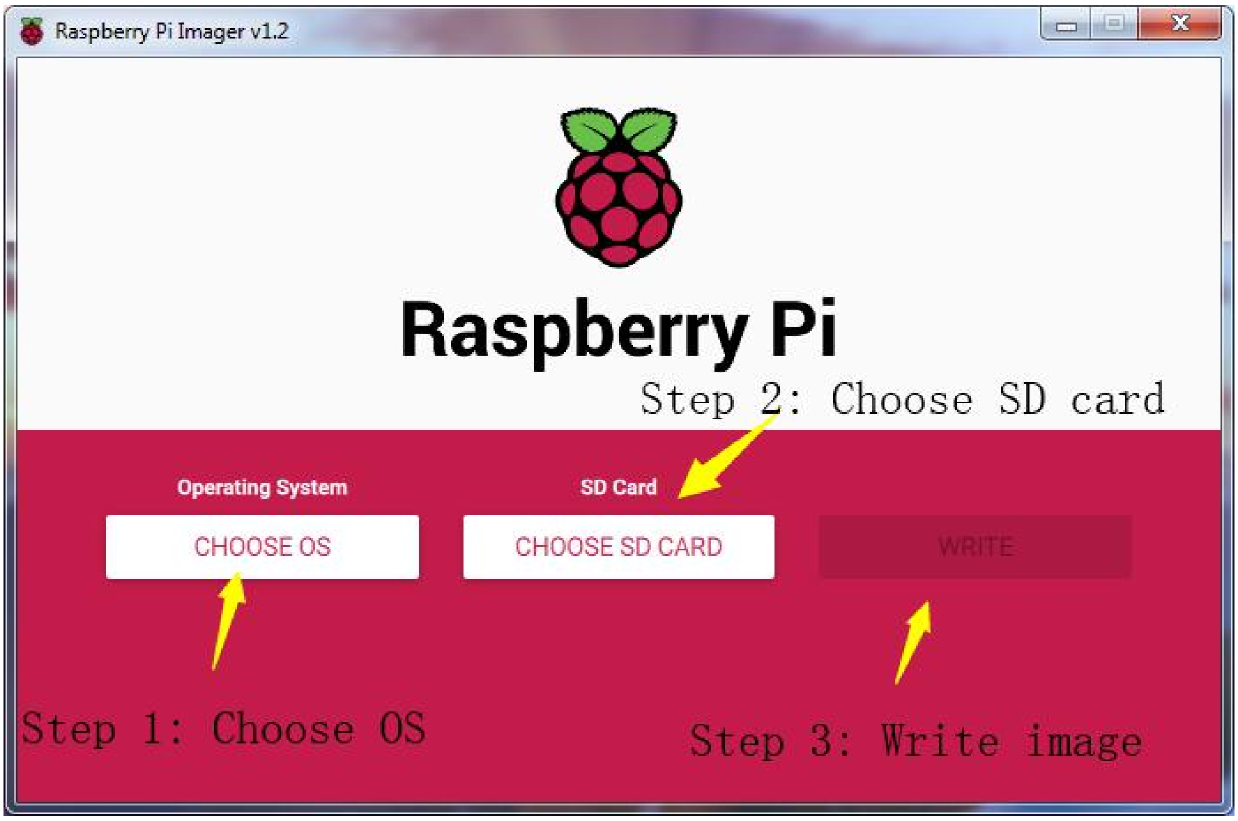
●Go to our [official website], find and download the image file [Image file for the Adeept_AWR Robot]. Unzip the file, be noted that the path should be in English for the .img file extracted, no special characters allowed. otherwise Raspberry Pi Imager may not open the .img file. It's recommended to save the .img file to the root directory of the C:\ or D:\ disk, but do not save .img on the SD card.

●Insert the SD card into the card reader, connect the card reader and your computer.

●Run the Raspberry Pi Imager, select CHOOSE OS, and then Use custom to find the .img extracted, click Open.

●Select CHOOSE SD CARD for the SD card to write the Raspbian, please be noted that the image writing will automatically delete all files on the SD card if any.

●Click on WRITE, wait for the writing.



●Do not remove the SD card connected after writing is completed, we'll use for configuring WiFi connection later. Otherwise, if you remove the card, insert it into the Raspberry Pi and boot it up, WiFi configuration without any peripherals may fail in the following process.

2.2 Enable SSH Server of Raspberry Pi

●By SSH (Secure Shell) server, you can use the command line of Raspberry Pi remotely on another device. In the subsequent operation and when using the Raspberry Pi, you don't have to connect a mouse, keyboard, or monitor to it, but simply control it on a computer in the same LAN.

●As of the November 2016 release, Raspbian has the SSH server disabled by default. You will have to enable it manually.

●The method to enable the SSH in this documentation can be referred to the Raspberry Pi official website SSH(Secure Shell)

2.2.1 Method A: Enable SSH with Peripherals

●If you use (**2.1.3 to manually download the image file we provide and write it to the SD card**) to write the operating system of the Raspberry Pi to the SD card, you do not need to refer to this section to open SSH, because The SSH service in the image is already enabled.

●If you've connected a mouse, keyboard, or monitor to the Raspberry Pi, follow these steps to enable SSH.

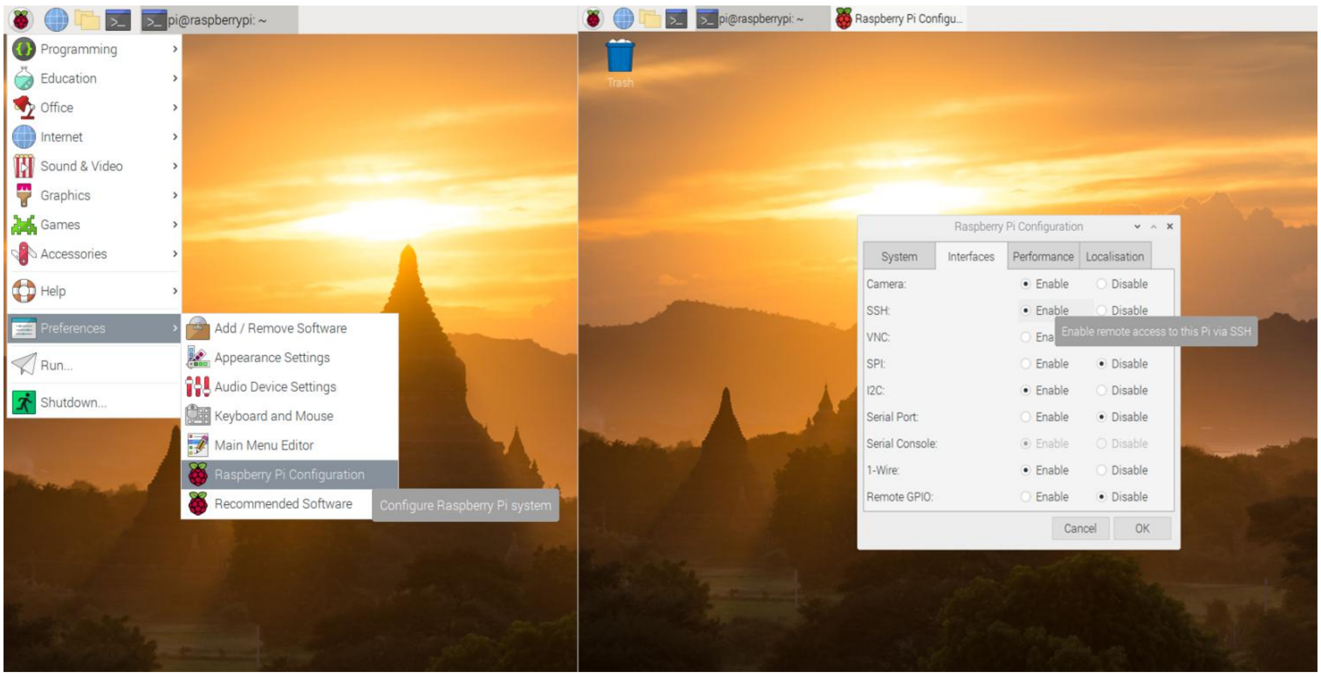
1.Remove the SD card from the computer, insert it to the Raspberry Pi, connect a mouse, keyboard, and monitor to the Raspberry Pi, boot it up.

2.Go to Preferences menu, select Raspberry Pi Configuration.

3.Go to Interfaces option.

4.Select Enable next to SSH.

5.Click on OK.



Camera: [✓ Enable] Disable
SSH: [✓ Enable] Disable
VNC: Ena
SPI: Enable [✓ Disable]
I2C: [✓ Enable] Disable
Serial Port: Enable [✓ Disable]
Serial Console: [✓ Enable] Disable
1-Wire: [✓ Enable] Disable
Remote GPIO: Enable [✓ Disable]

2.2.2 Method A: Enable SSH without Peripherals

●If you use (**2.1.3 to manually download the image file we provide and write it to the SD card**) to write the operating system of the Raspberry Pi to the SD card, you do not need to refer to this section to open SSH, because The SSH service in the image is already enabled.

●If you haven't connected any monitor to the Raspberry Pi, follow these steps to enable SSH.

1. Do not remove the SD card after `Raspberry Pi Imager` writes the image file.

2. Create a file named `ssh` under any directory, without any extension name. You may create a `ssh.txt` and delete the `.txt` (make sure under Folder Options the box of Hide extensions for known file types is unchecked. Then you have an `ssh` file without extension name.

3. Copy the `ssh` file and paste to the root directory of the SD card. The Raspberry Pi will auto search for the `ssh` file when booting, and enable SSH if the file is found. You only need to copy for one time because the Raspberry Pi then will automatically enable SSH at every boot.

4. Do not remove the SD card if you need to configure WiFi.

2.3 Configure WiFi on Raspberry Pi

●There are many ways to connect WiFi for Raspberry Pi. Two methods are provided in this documentation; you may visit the official Raspberry Pi website for more: [Wireless connectivity]

2.3.1 Method A: WiFi Connection with Peripherals

● If you've connected a mouse, keyboard, or monitor to the Raspberry Pi, follow these steps to configure WiFi.

1.Remove the SD card from the computer, insert it to the Raspberry Pi, connect a mouse, keyboard, and monitor to the Raspberry Pi, boot it up.

2. Select the WiFi icon at the top right corner on the monitor, find the WiFi to connect and select.

3. Type in the password for the WiFi, connect.

4. After it's connected successfully, the WiFi will be saved and the Raspberry Pi will auto connect for next boot, so you don't need to connect peripherals every time.

2.3.2 Method A: WiFi Connection without Peripherals

●If you haven't connected any monitor to the Raspberry Pi, follow these steps to configure WiFi.

●This method is based on the [official documentation]

1. Do not remove the SD card after `Raspberry Pi Imager` has written the image file. (This method works for the situation that the Raspbian image file has just been written to the SD card; if you've already plugged the SD card into the Raspberry Pi and got it rebooted after the image file being written, the configuration may fail.)

2. Create a file named `wpa_supplicant.conf` anywhere in your computer.

3. Open the file `wpa_supplicant.conf` created with Textbook, enter the following code:

```
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1
country=Insert country code here
network={
ssid="Name of your WiFi"
psk="Password for your WiFi"
}
```

4. Type in your own information for `Insert country code here`, `Name of your WiFi`, and `Password for your WiFi`. Pay attention to the capitalization. Refer to the example below:

```
    ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
    update_config=1
    country=US
    network={
    ssid="MyName"
    psk="12345678"
}
```

5. Save and exit. Copy the `wpa_supplicant.conf` to the root directory of the SD card.

6. If you've already copied the file `ssh` to the SD card as instructed in **2.2**, then both the WiFi and SSH settings without peripherals are done. You may remove the SD card, insert it into the Raspberry Pi, and boot it up.

7. For more about the file `wpa_supplicant.conf`, refer to the official documentation [WIRELESS-CLI]

❖ 3 Log In to The Raspberry Pi and Install The App

●If you followed the steps in **2.2.1** and **2.3.1** for SSH and WiFi configuration, you may remove the peripherals now and use SSH to remotely control the Raspberry Pi later on.

●If you followed the steps in **2.2.2** and **2.3.2**, you may now insert the SD card into the Raspberry Pi and boot it up. The Raspberry Pi will auto boot and connect WiFi when powered on, with no need of peripherals.

●If you use the operation steps of **2.1.3** to write to the SD card, you only need to refer to **2.3.1** or **2.3.2** to configure the WIFI, you can install the SD card into the Raspberry Pi, and the robot product program will Automatic operation, you can skip some content, refer to **5 Use WEB application to control the robot** after the structure is assembled.

●Some steps mentioned below are based on the official Raspberry Pi documentation SSH.

●For power supply of the Raspberry Pi, refer to the official documentation Power supply.

●The Motor HAT board of the Adeept Raspberry Pi Robot can supply power for the Raspberry Pi via GPIO port. However, since it may take a long time to install software on the Raspberry Pi, it's not recommended to supply with the batteries during this process. You may skip the installation of the Motor HAT board or camera during software installation; though you need to make sure the driver board and camera for the Raspberry Pi when it's ready to run the software installed, or a program error will occur.

3.1 Log into Raspberry Pi (Windows 10)

●For Windows 10, SSH is built in the versions after October 2018, so you don't need any third-party software.

●For lower versions of Windows OS, SSH is not built in, and you may log into the Raspberry Pi by referring to the official documentation [SSH using Windows].

●Before connecting the Raspberry Pi via SSH, you need to know the IP address of the Raspberry Pi. Check the Management interface for your router, or download the app `Network Scanner` -> search for a device named `RASPBERRY` or `Raspberry Pi Foundation` to get the IP address.

●For other methods of obtaining the IP address of Raspberry Pi, refer to the official documentation [IP Address]

●Press the keys 'win'+'R', type in `cmd`, and press 'enter'.

●The default user is `pi`, and the password is `raspberry`.

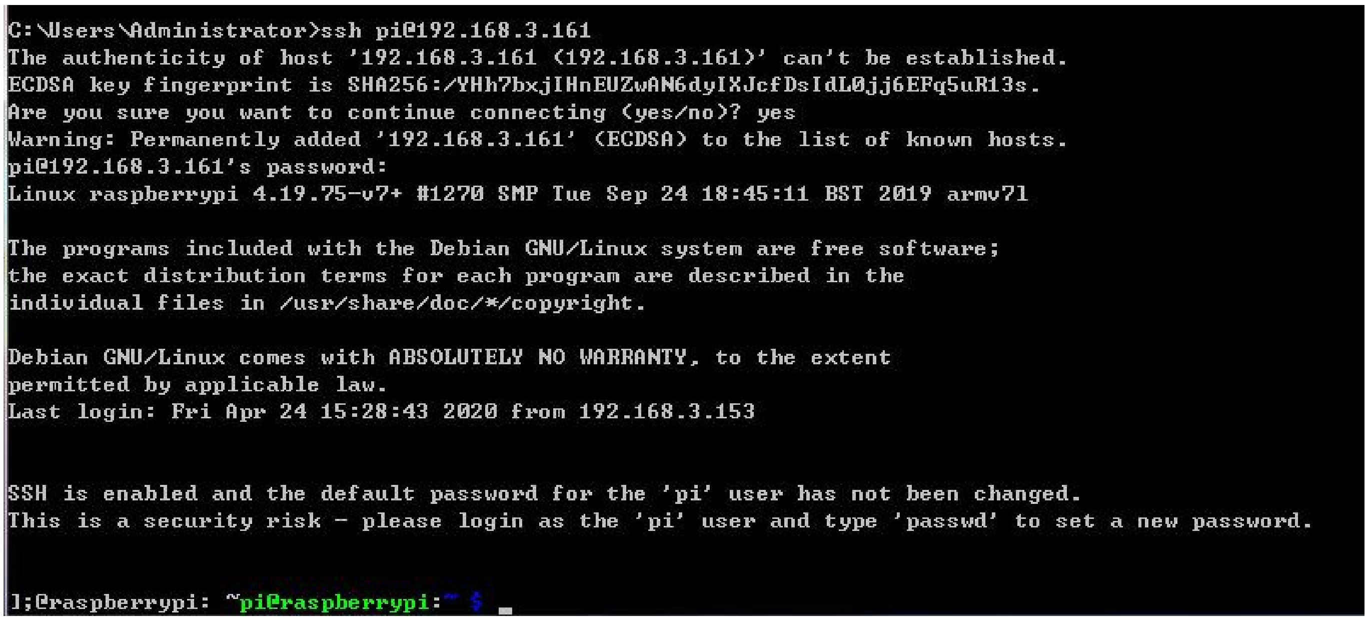
● Type in `ssh pi@<IP>` in the command line, replace the `<IP>` with the IP address of your Raspberry Pi, as shown below:

```
ssh pi@192.168.3.161
```

●Press Enter key and a prompt will appear: `Are you sure you want to continue connecting (yes/no)?`

●Type in `yes`, press Enter and it'll show `pi@192.168.3.161's password:`, type in the initial password of the Raspberry Pi, `raspberry` (pay attention to capitalization). There's no change on the screen when you're typing in, but it doesn't mean you're not entering the information. Press ‘enter’ after you finish typing in.

●So now you've logged into the Raspberry Pi.



```
C:\Users\Administrator>ssh pi@192.168.3.161
The authenticity of host '192.168.3.161 (192.168.3.161)' can't be established.
ECDSA key fingerprint is SHA256:/YHh7bxjIHnEUZwAN6dyIXJcfDsIdL0jj6EFq5uR13s.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.3.161' (ECDSA) to the list of known hosts.
pi@192.168.3.161's password:
Linux raspberrypi 4.19.75-v7+ #1270 SMP Tue Sep 24 18:45:11 BST 2019 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri Apr 24 15:28:43 2020 from 192.168.3.153


SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set a new password.


];@raspberrypi: ~pi@raspberrypi:~ $ _
```

3.2 Log into Raspberry Pi (Linux or Mac OS)

●Before connecting the Raspberry Pi via SSH, you need to know the IP address of the Raspberry Pi. Check the Management interface for your router, or download the app `Network Scanner` -> search for a device named `RASPBERRY` or `Raspberry Pi Foundation` to get the IP address.

●For other methods of obtaining the IP address of Raspberry Pi, refer to the official documentation [IP Address]

●Open the terminal window (or command line)

●The default user is `pi`, and the password is `raspberry`.

●Type in `ssh pi@<IP>` in the command line, replace `<IP>` with the IP address of your Raspberry Pi as shown below:

ssh pi@192.168.3.161

●Press Enter key and a prompt will appear: `Are you sure you want to continue connecting (yes/no)?`

●Type in `yes`, press Enter and it'll show `pi@192.168.3.161's password:`, type in the initial password of the Raspberry Pi, `raspberry` (pay attention to capitalization). There's no change on the screen when you're typing in, but it doesn't mean you're not entering the information. Press ‘enter’ after you finish typing in.

●So now you've logged into the Raspberry Pi.

3.3 Log into Raspberry Pi (Windows 8 or Previous Version)

●For lower versions of Windows OS, SSH is not built in, and you may log into the Raspberry Pi by referring to the official documentation Raspberry Pi[SSH using Windows].

●Before connecting the Raspberry Pi via SSH, you need to know the IP address of the Raspberry Pi. Check the Management interface for your router, or download the app `Network Scanner` -> search for a device named `RASPBERRY` or `Raspberry Pi Foundation` to get the IP address.

●For other methods of obtaining the IP address of Raspberry Pi, refer to the official documentation [IP Address]

●You may need to download the `PuTTY` version for your OS and log into Raspberry Pi with the tool. [Click here to download PuTTY]

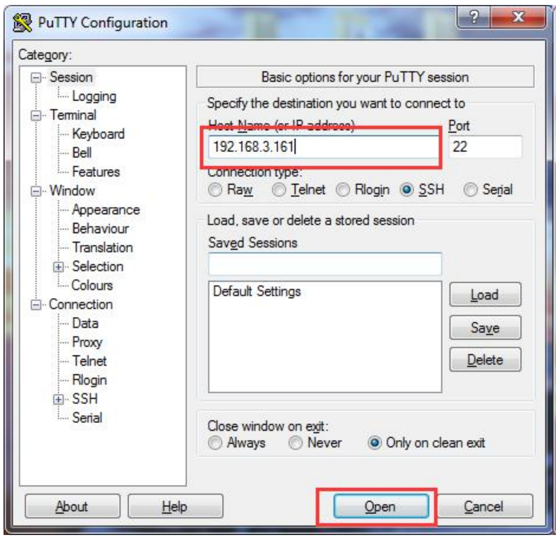
●Run `PuTTY`, type in the IP address of Raspberry Pi for `Host Name`, and click 'Open'.

●If a prompt of `Network error: Connection timed out` appears, possibly you've entered an incorrect IP address.

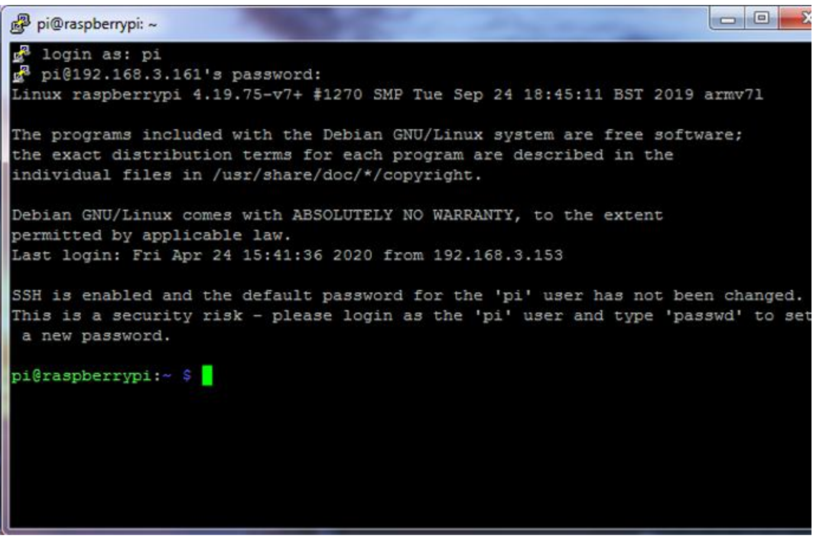
●When the connection works you will see the security warning shown below. You can safely ignore it, and click the 'Yes' button. You will only see this warning the first time PuTTY connects to a Raspberry Pi that it has not seen before.

●You will now see the usual login prompt. Log in with the same username and password you would use on the Pi itself. The default login for Raspbian is `pi` with the password `raspberry`.

●You should now have the Raspberry Pi prompt which will be identical to the one found on the Raspberry Pi itself.



Specify the destination you want to connect to
Host Name (or IP address): 192.168.3.161
Port: 22
Connection type: Raw Telnet Rlogin [✓ SSH] Serial
Close window on exit: Always Never [✓ Only on clean exit]



```
login as: pi
pi@192.168.3.161's password:
Linux raspberrypi 4.19.75-v7+ #1270 SMP Tue Sep 24 18:45:11 BST 2019 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri Apr 24 15:41:36 2020 from 192.168.3.153

SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set
 a new password.

pi@raspberrypi:~ $
```

3.4 Download Program of The Raspberry Pi Robot

●The code for the robot product has been uploaded to [GitHub], you may need to download to your Raspberry Pi and install the corresponding dependent libraries to run the program.

●In the previous section you've logged into the Raspberry Pi, and here type in the follow command in the terminal window:

```
sudo git clone https://github.com/adeept/Adeept_AWR.git
```

●Press '**enter**' to start downloading the program of the robot from GitHub. It may take some time, please wait until it's done.

3.5 Install Corresponding Dependent Libraries

● Follow the steps below to install the libraries if you wrote the image file to the SD card based on 2.1.1 Write 'Raspbian' to the SD card by `Raspberry Pi Imager` and 2.1.2 Download the image file `Raspbian` and write it to the SD card manually.

● Here a script is provided for installing all dependent libraries needed and configuration of starting the camera and other auto start programs.

● Type in the code below in the terminal window to run the dependent libraries for the script `setup.py`:

```
sudo python3 Adeept_AWR/setup.py
```

● Press ‘**enter**’ and the script will auto run. This may take minutes or hours, depending on the network status. Please wait until it's done.

● After installation is completed, the following prompts will appear:

```
The program in Raspberry Pi has been installed, disconnected and restarted.
You can now power off the Raspberry Pi to install the camera and driver board (Motor HAT). After turning on again, the Raspberry Pi will automatically run the program to set the servos port signal to turn the servos to the middle position, which is convenient for mechanical assembly.
```

● When installation is completed, the Raspberry Pi will automatically disconnect SSH and reboot. If you used puTTy to connect the Raspberry Pi, there can be an error prompt like `Network error:Software caused connection abort`. You can just ignore and close it.

3.6 Run the Raspberry Pi Robot's Program

● Raspberry Pi auto runs the program for the robot when rebooting every time, which is the part `[RobotName]/server/webServer.py` (replace `[RobotName]` for the name of the folder for your robot product's program). However, if the Raspberry Pi camera or Motor HAT is not connected, the `webServer.py` can't run well. It makes sense because the robot's program needs the camera and the chipset PCA9685. Motor HAT controls servo with PCA9685; the Raspberry Pi communicates with PCA9685 via I2C; so if Motor HAT is not connected to Raspberry Pi, a program error will occur when instantiating dependent libraries for PCA9685 due to communication failure.

● Power off the Raspberry Pi, connect camera module and Motor HAT, reboot it up and now `webServer.py` can run.

● Generally you don't need to manually run `webServer.py` since it's auto run by Raspberry Pi at every boot.

● Open a web browser (Google Chrome for example), in the address bar type in the IP address of the Raspberry Pi, add `**:5000**` to the end as shown below, and press ‘enter’ to redirect to the web page of the Raspberry Pi:

```
http://192.168.3.157:5000/
```

● If it fails to enter the page, log into the Raspberry Pi via SSH, type in the command below to end the program auto run at boot to release resources, or else issues like camera initialization failure or occupied ports.

```
sudo killall python3
```

●Type in the command below to run `**webServer.py**`:

```
sudo python3 Adeept_AWR/server/webServer.py
```

●Check whether there's any error and solve them based on instructions in the **Q&A** section below.

❖ 4 Assembly and Precautions

4.1 Structure Assembly

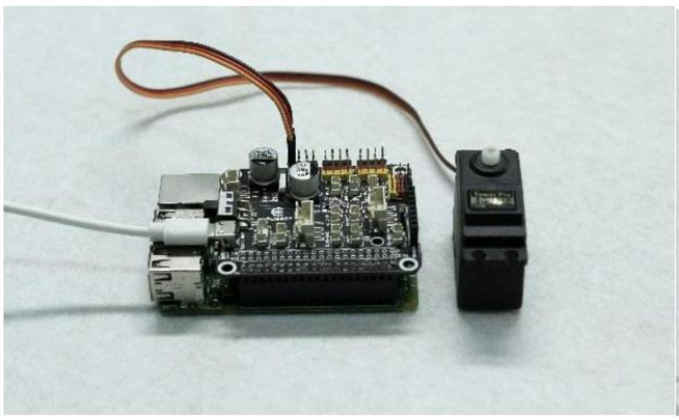
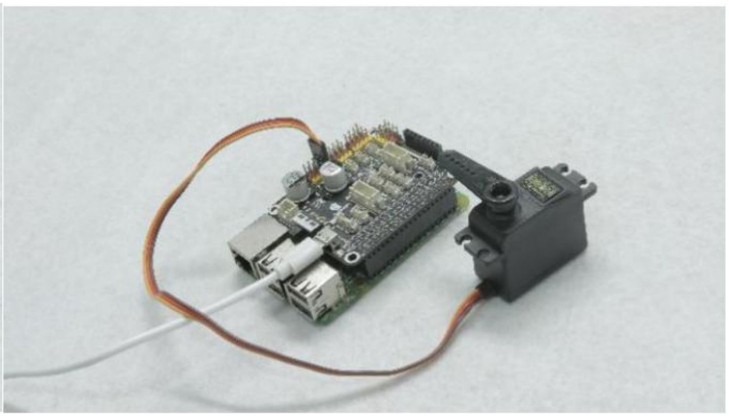
Before assembling

Before installing the robot, **you need to install the software (refer to second half of the document) to control the robot on the Raspberry Pi**. Because the servo needs to be returned to the original position during the robot assembly process, which requires the assistance of a Raspberry Pi with software that runs normally.

How to make a servo return to original position

Connect the servo to the Raspberry Pi that is **turned on**. When the connection is completed, the servo will immediately run, and it will automatically return to the original position in a very short time. Then you can install the rocker arm on the servo at a specified angle.

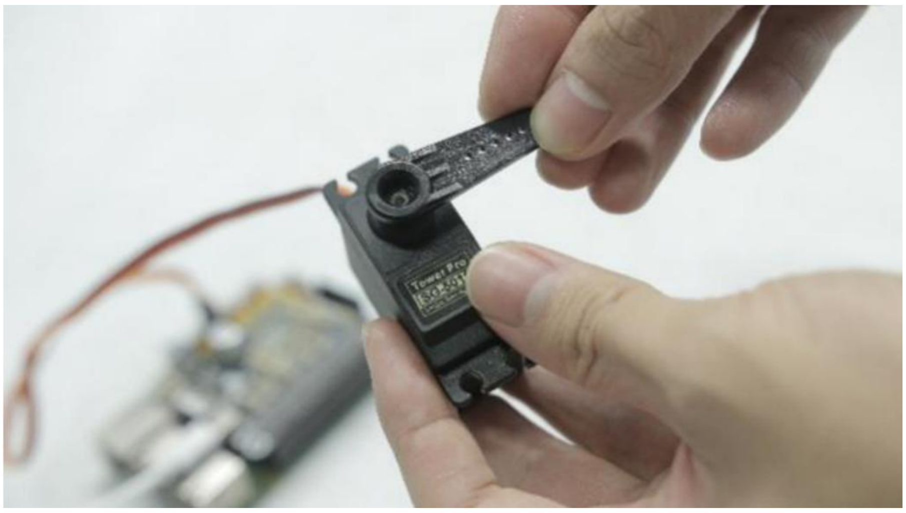
automatically return ➔ install rocker arm

(The type of servo and the installation angle of the rocker arm is for reference only. Please refer to the actual product and assembly part.)

Whether the servo has returned to original position

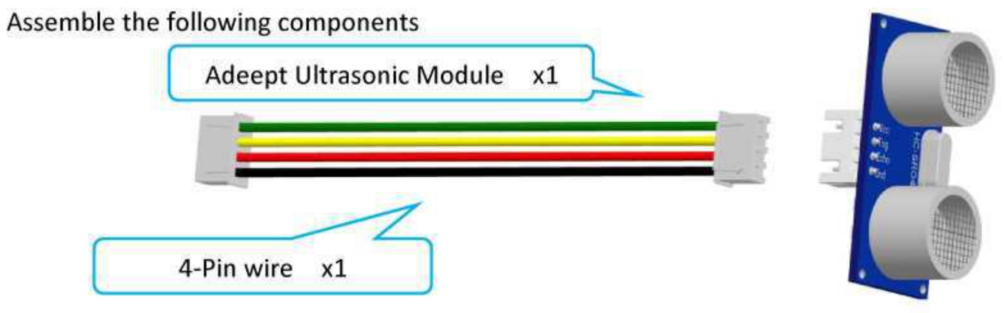
You can test whether the servo has returned to original position by pulling the rocker arm (don't try too hard to prevent damage to the servo). The servo that has automatically returned cannot be pulled.

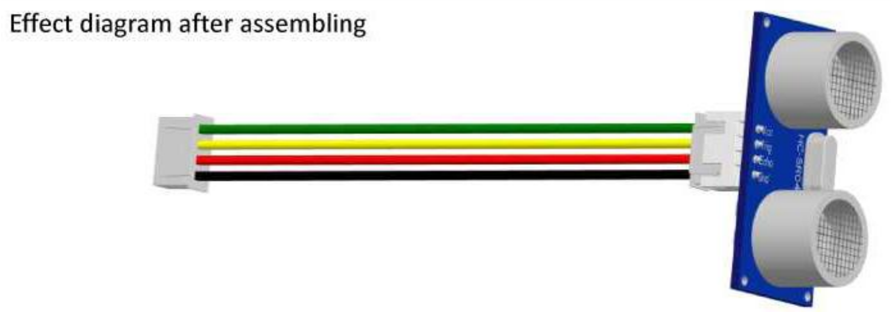


If your servo does not return to the original position automatically, you can manually run the server.py file and then try to connect the servo.

●Preparations before Assembly

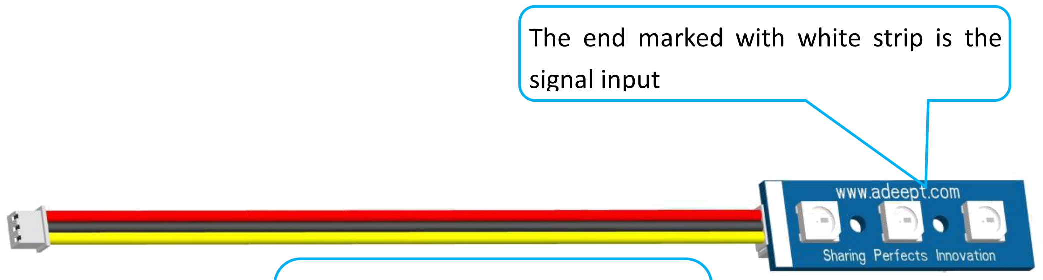
Connect the Adeept Ultrasonic Module with 4-Pin wire.

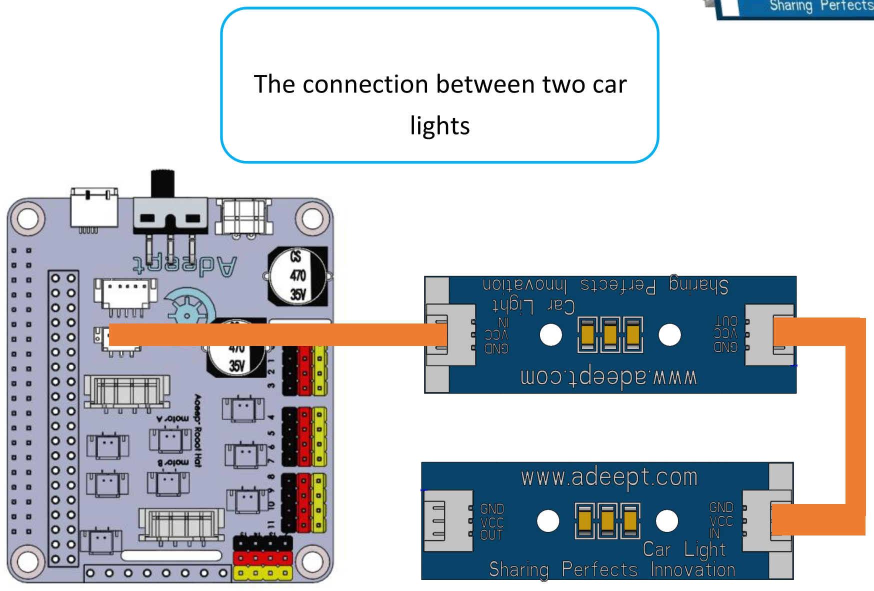

Assemble the following components
Adeept Ultrasonic Module x1
4-Pin wire x1



Connect the car light.

Please note that the end marked with white strip is the signal input, and the end without white strip is the signal output and connect the WS2812 to the Adeept Robot HAT.

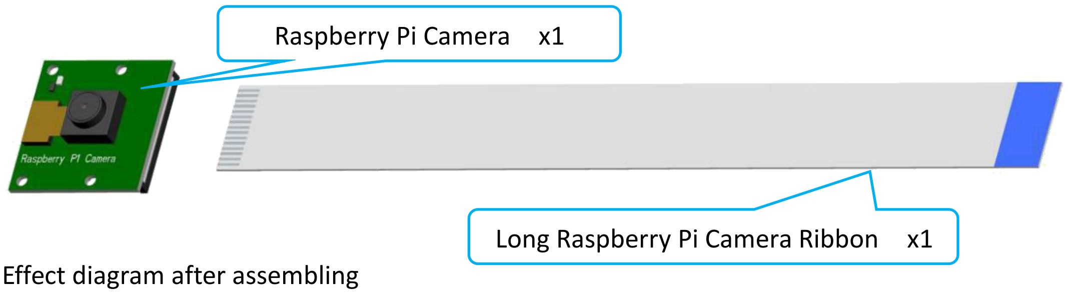

The end marked with white strip is the signal input
www.adeept.com
Sharing Perfects Innovation


The connection between two car lights
www.adeept.com
GND
VCC
OUT
GND
VCC
IN
Car Light
Sharing Perfects Innovation

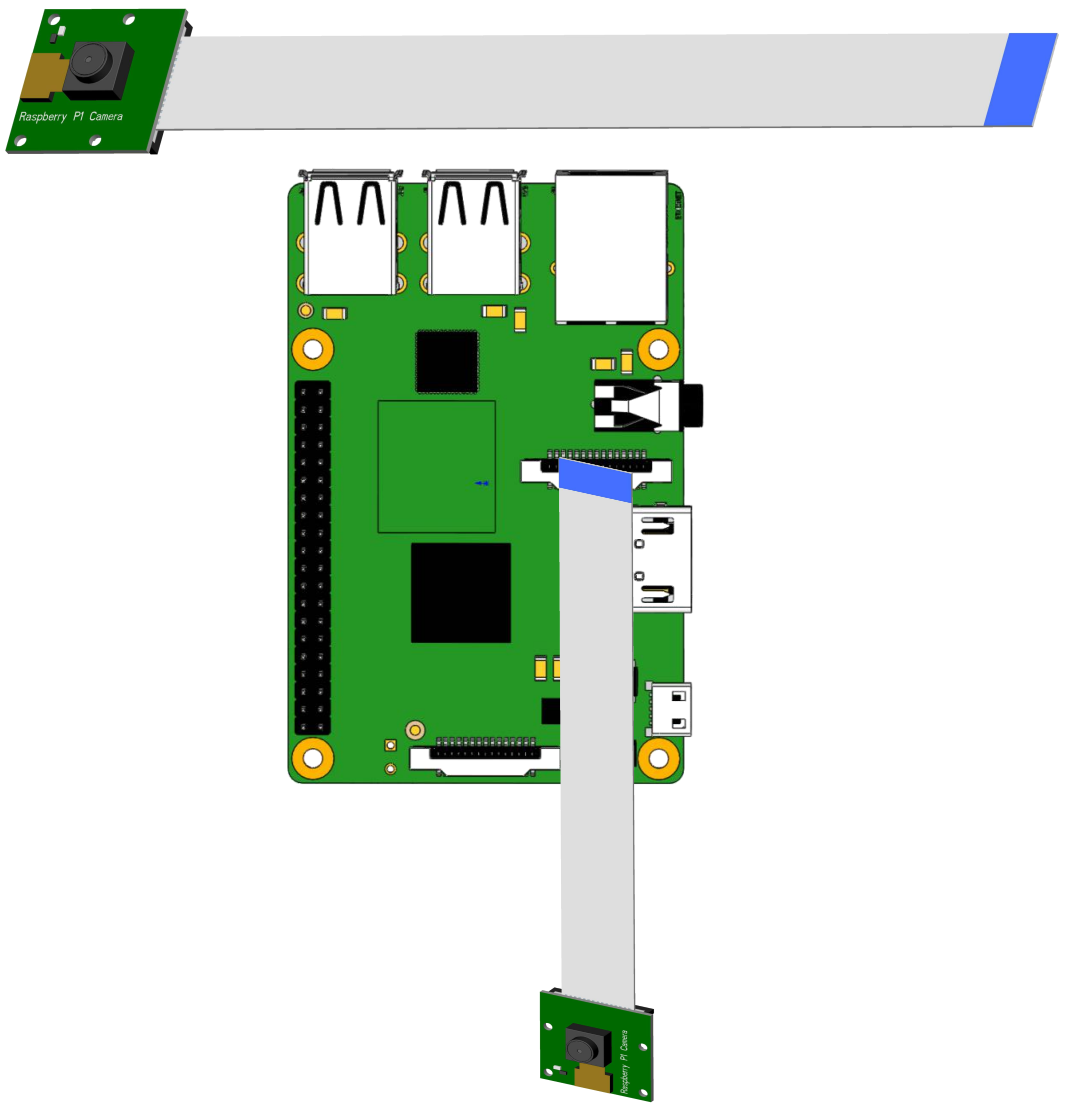
Connect the Raspberry Pi Camera and the ribbon.

Note: That in the next operation, the Pi Camera of the Raspberry Pi should always be connected to th e Raspberry Pi, and do not reverse the wires of the Raspberry Pi.

Assemble the following components



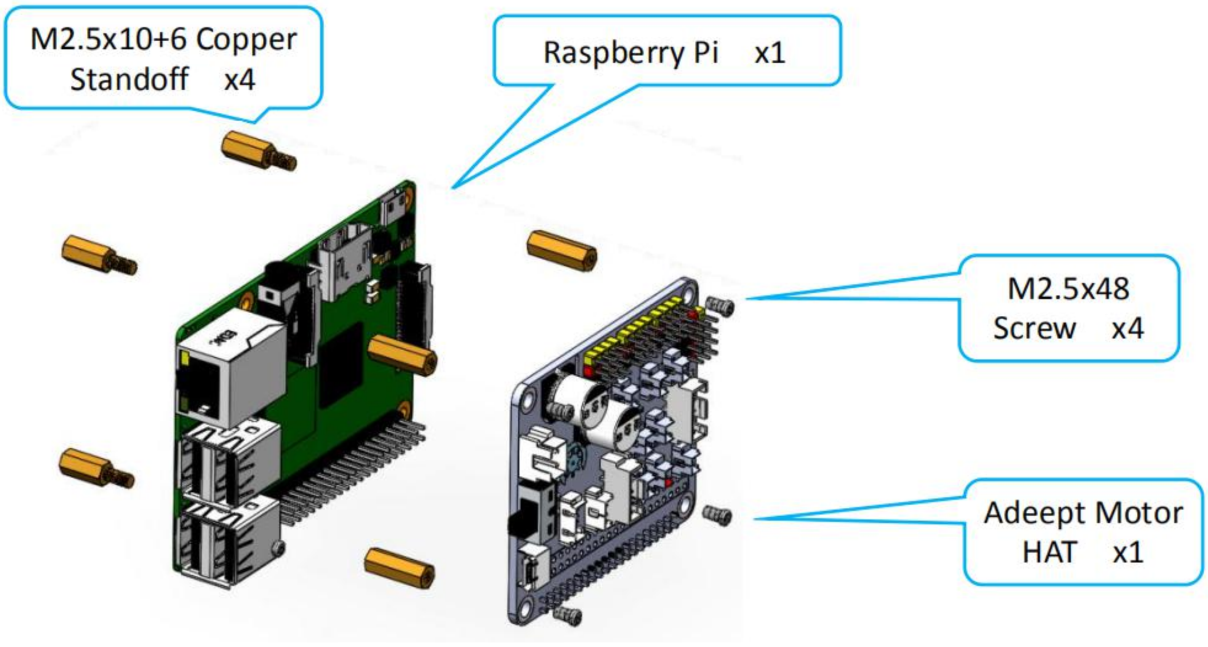
Effect diagram after assembling



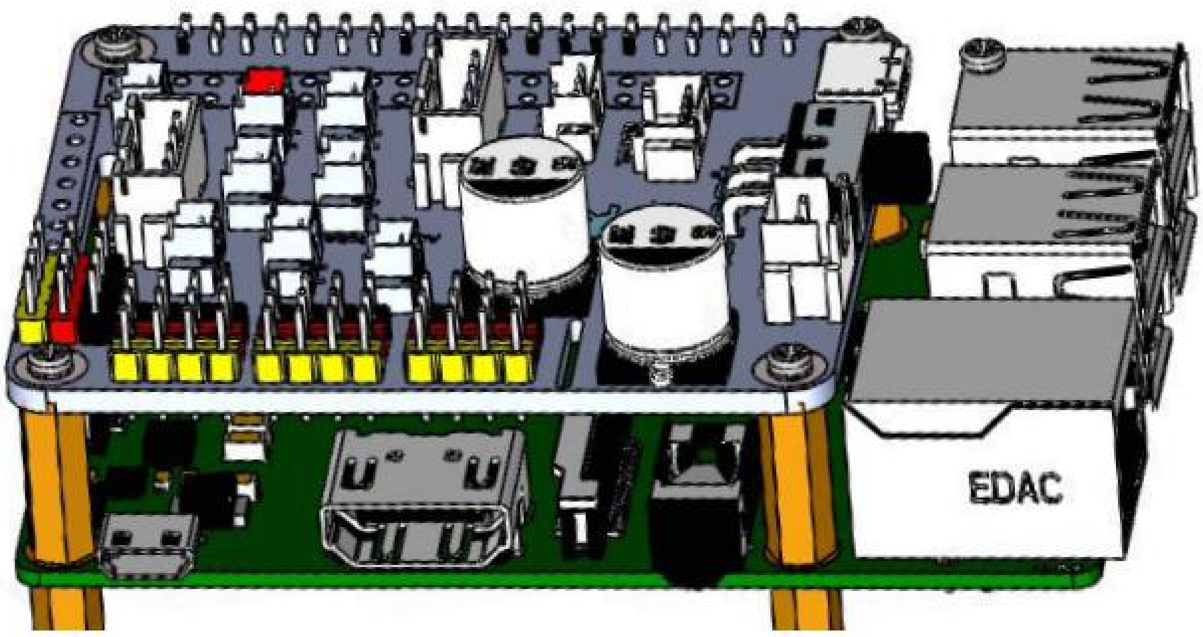
- **Fix four M2.5x10+6 Copper Standoffs on Raspberry Pi.**

and insert the Adeept Motor HAT into Raspberry Pi.

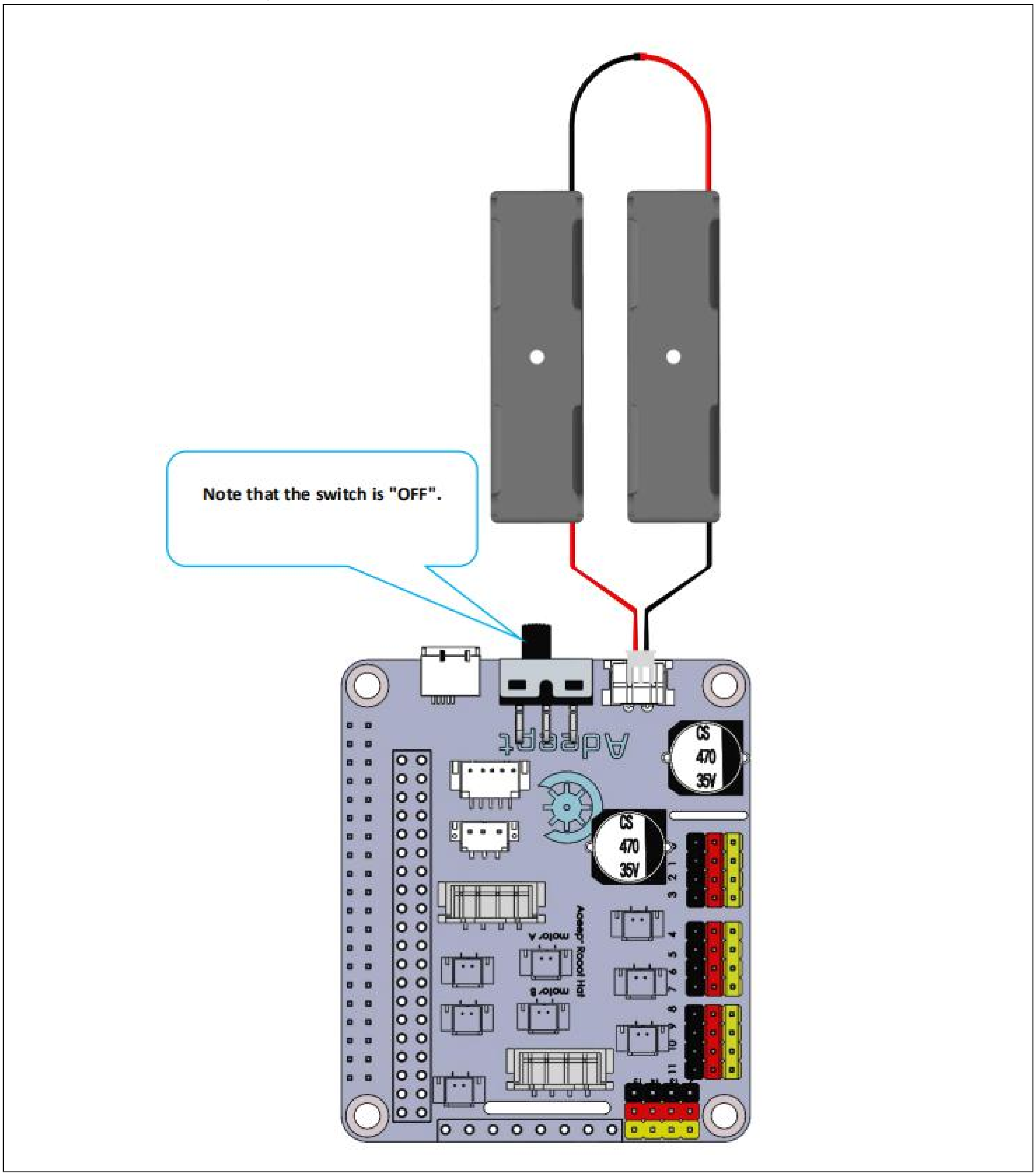
Assemble the following components



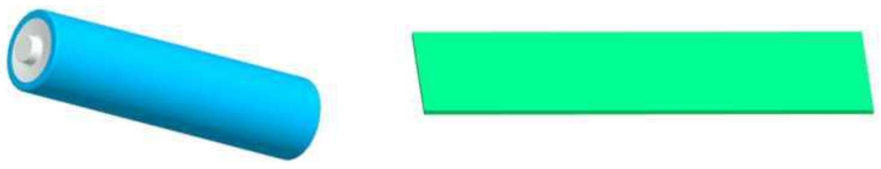
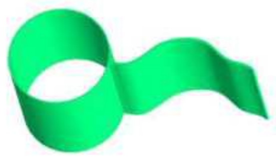
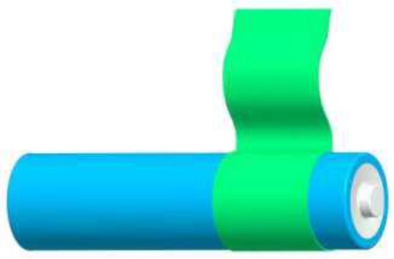
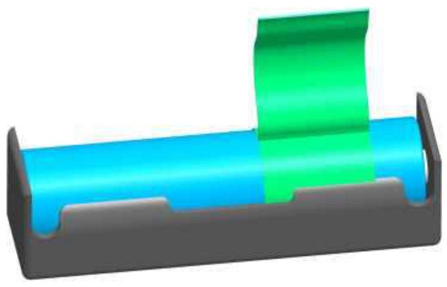
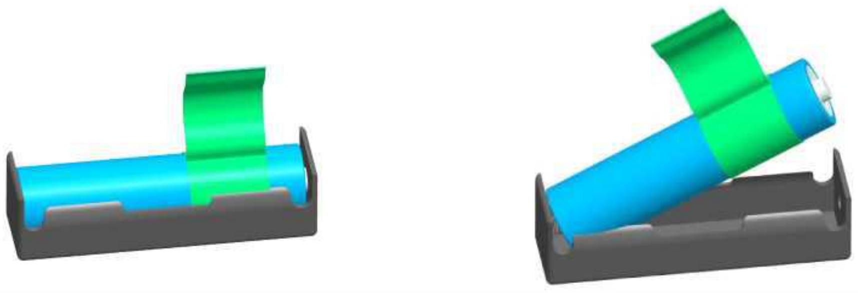
Effect diagram after assembling



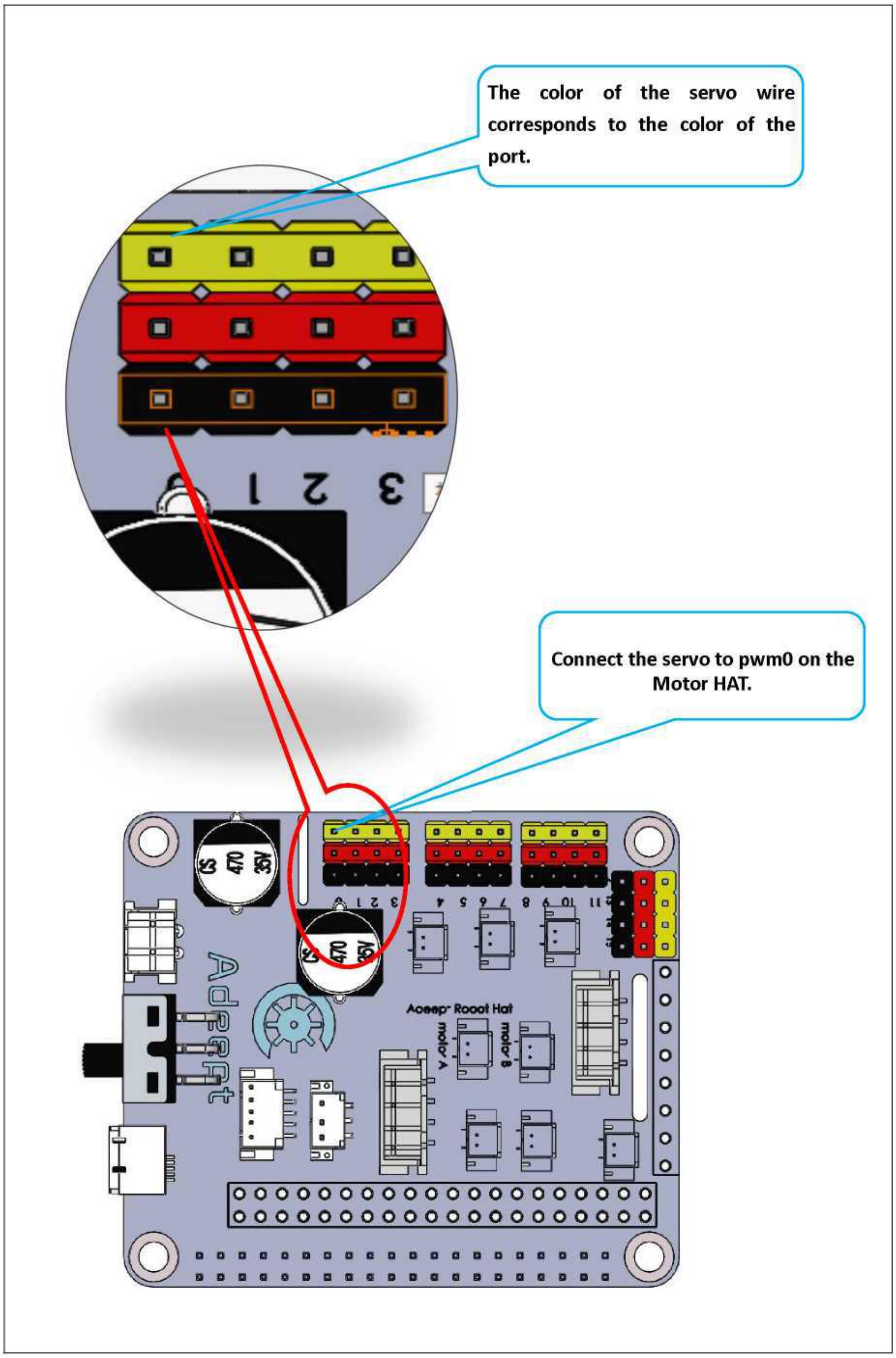
1. Connect the 18650 Battery Holder Set to the Adeept Motor HAT.



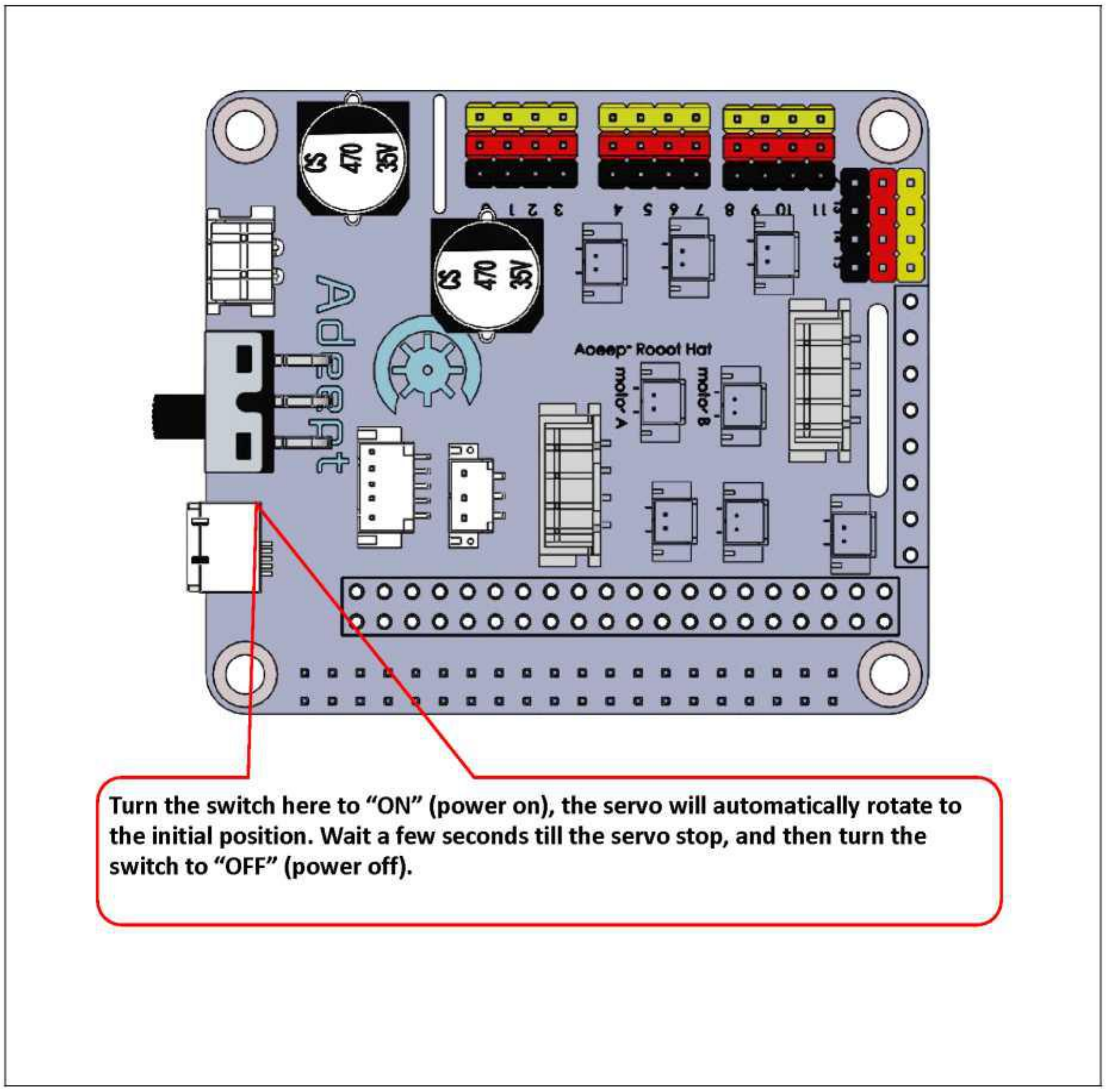
2. Put two 18650 batteries in 18650 Battery Holder Set according to the following method.

Take out 1 ribbons and 1 batteries. 
Roll one end of the ribbon to let through a battery and fix. 
Insert the batteries into the rings - ribbon closer to the anode. 
Install the batteries into the holder based on the pole. 
To remove the batteries, just pull the ribbon and take them out. 

3. Connect servos to Adeept Motor HAT.


The color of the servo wire corresponds to the color of the port.
Connect the servo to pwm0 on the Motor HAT.
Adeept
Adeept Robot Hat
motor A
motor B

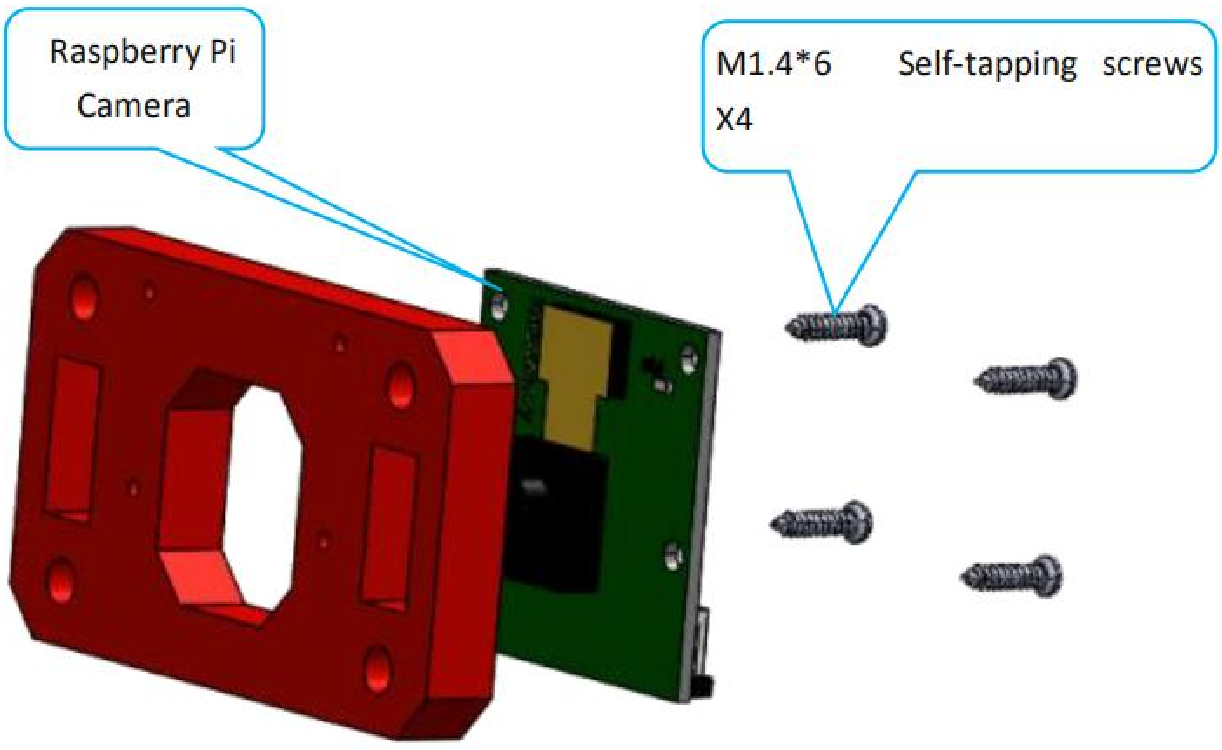
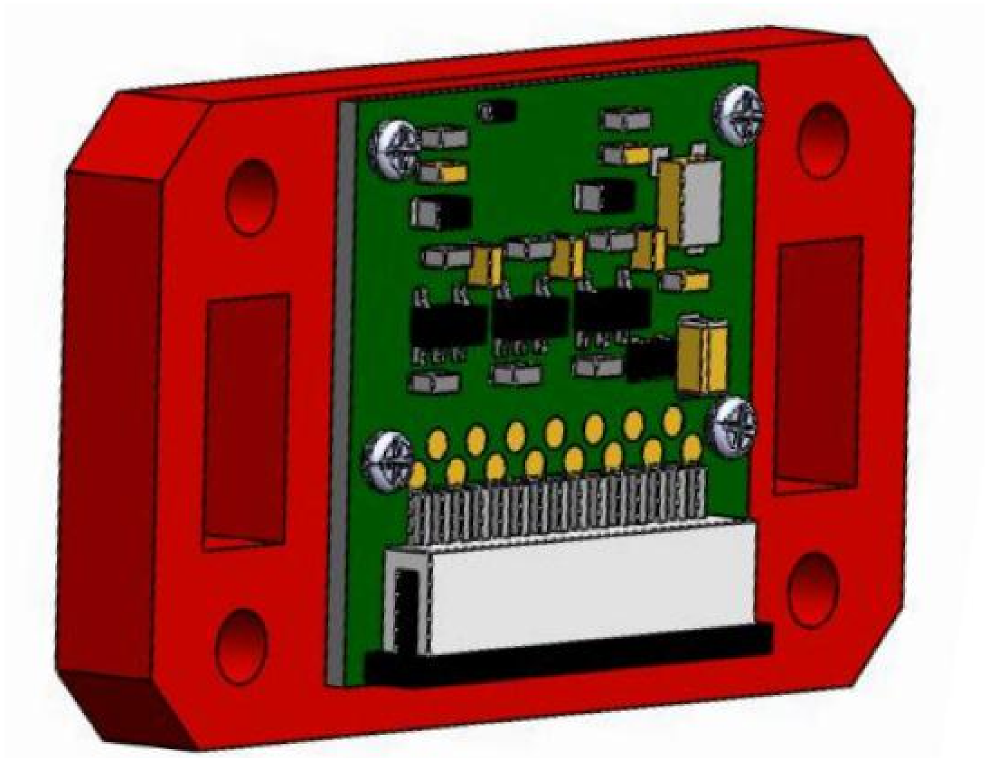
4. Before switching on, you need to insert the configured SD card into the Raspberry Pi. For details, please refer to the third chapter of the document. Otherwise, the servo will not rotate to the middle position after booting. If SD card is not inserted, the servo needs to be rotated to the middle position manually.


Adeept Robot Hat
motor A
motor B
Turn the switch here to “ON” (power on), the servo will automatically rotate to the initial position. Wait a few seconds till the servo stop, and then turn the switch to “OFF” (power off).

After debugging, remove the servo and battery holder, and take the 18650 batteries out of the Holder Set. Do not rotate the rotation axis before the servo fixed to the rocker arm. Otherwise, you need to re-debug the servo.

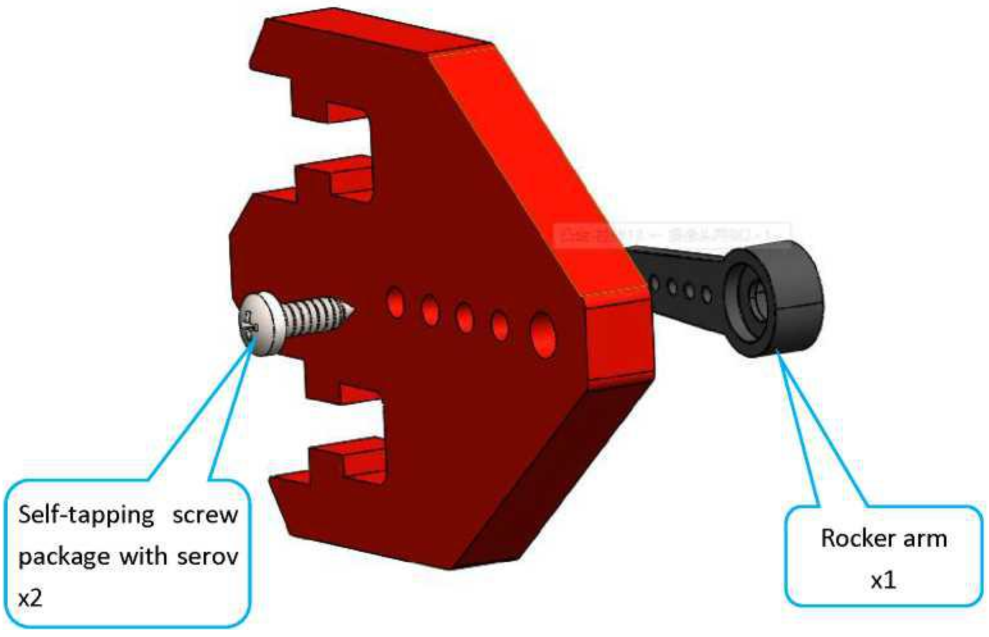
●**Body parts.**

1. Fix Raspberry Pi Camera on Acrylic Plates

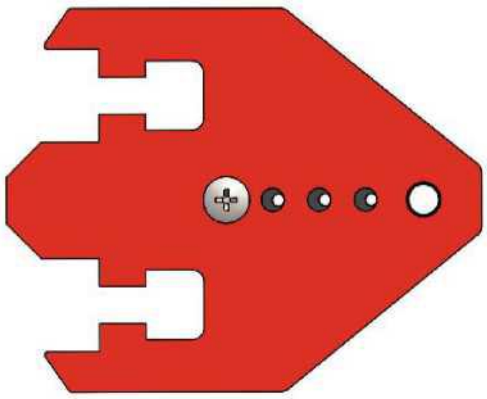
Assemble the following components

Effect diagram after assembling


2. Fix a rocker arm to the acrylic plate

Assemble the following components

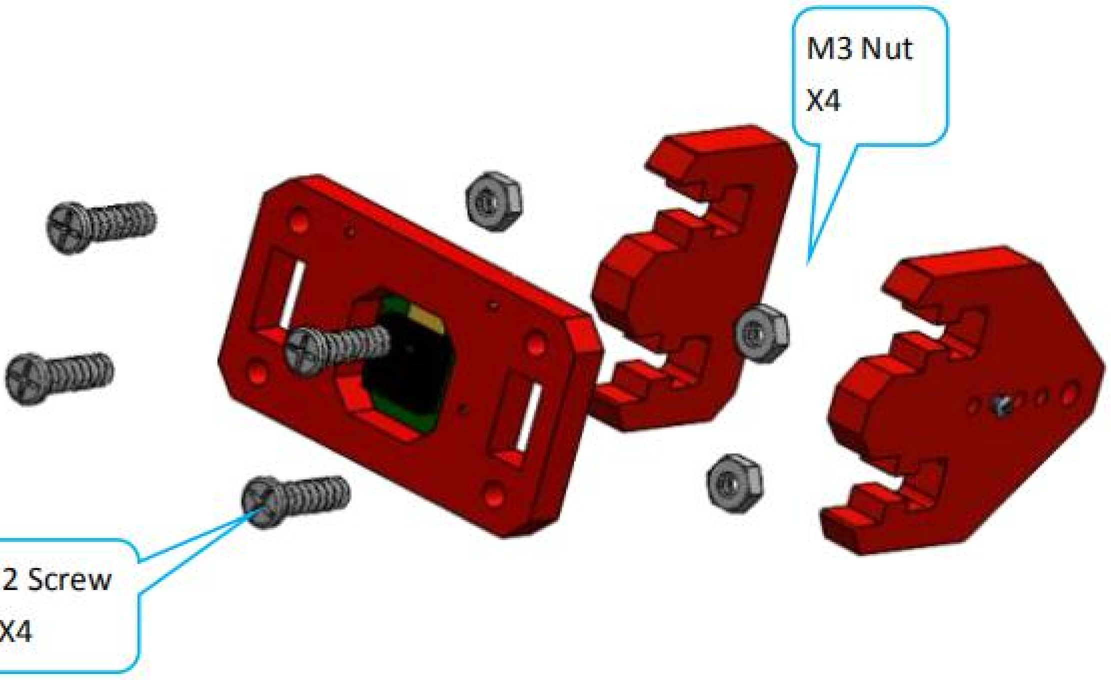


Effect diagram after assembling

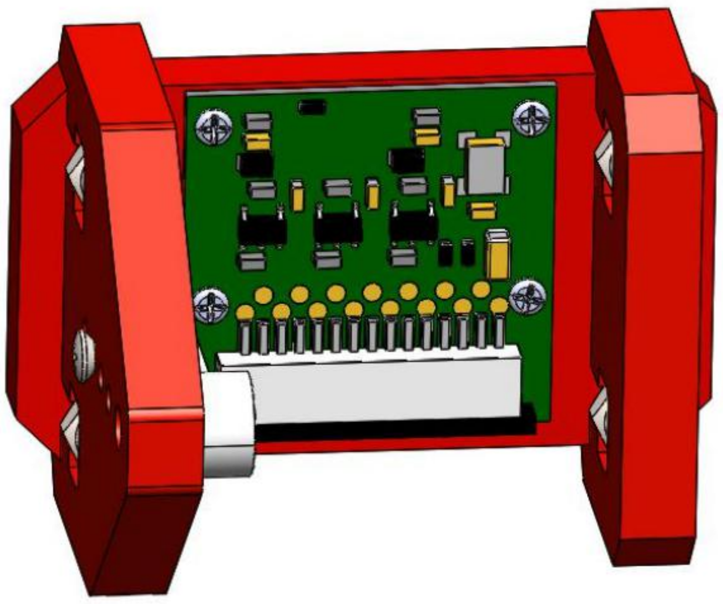


3. Assemble the camera.

Assemble the following components

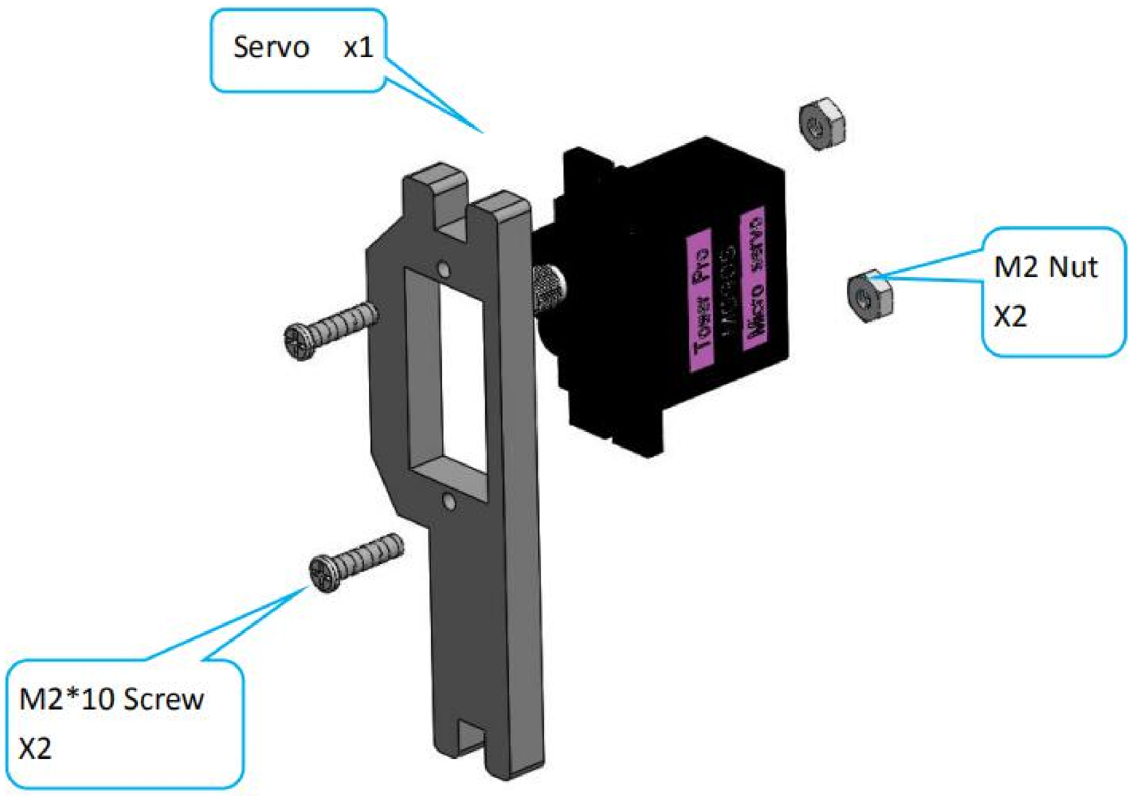


Effect diagram after assembling

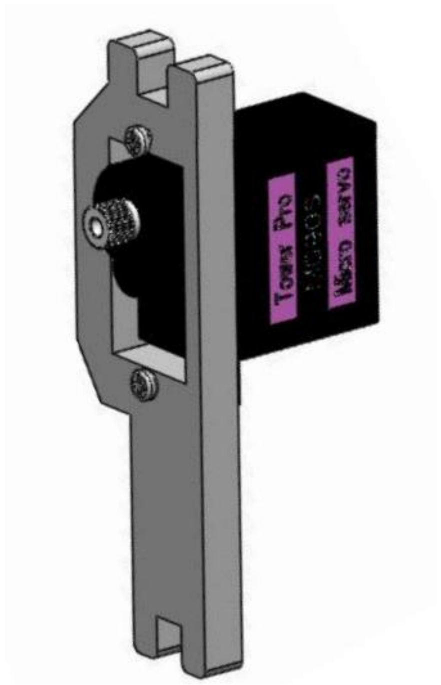


4. Fix a debugged servo to the acrylic plate.

Assemble the following components

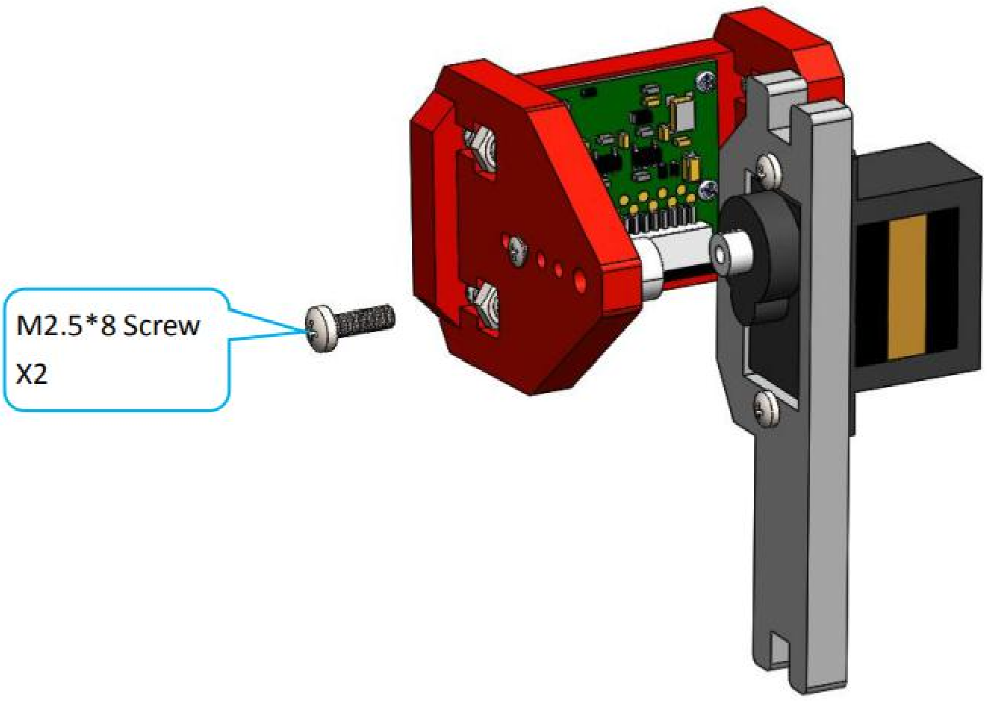


Effect diagram after assembling

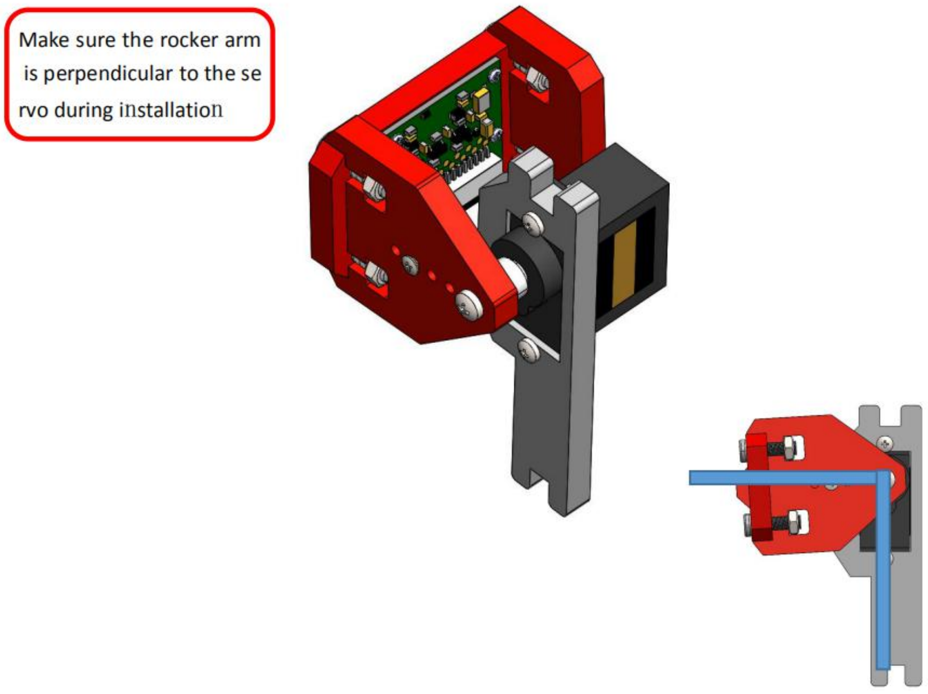


5. Fix the rocker arm on acrylic to the servo on acrylic.

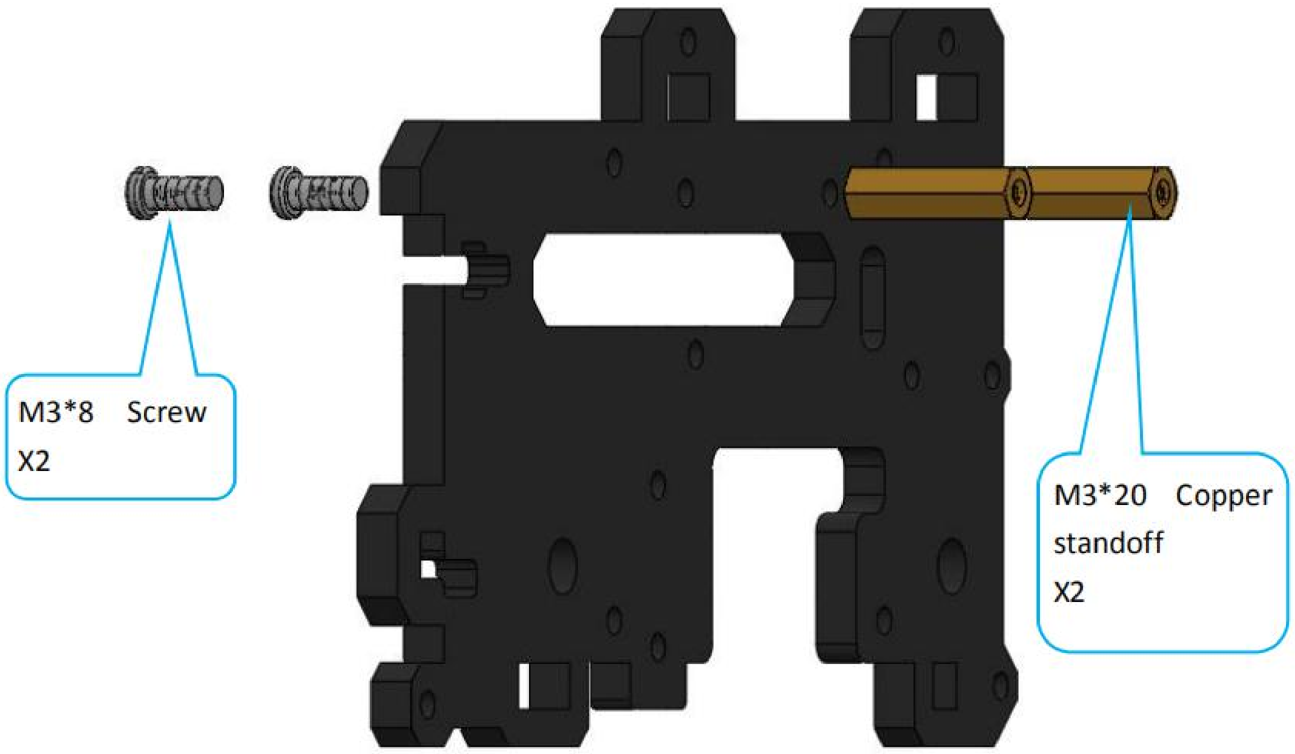
Assemble the following components



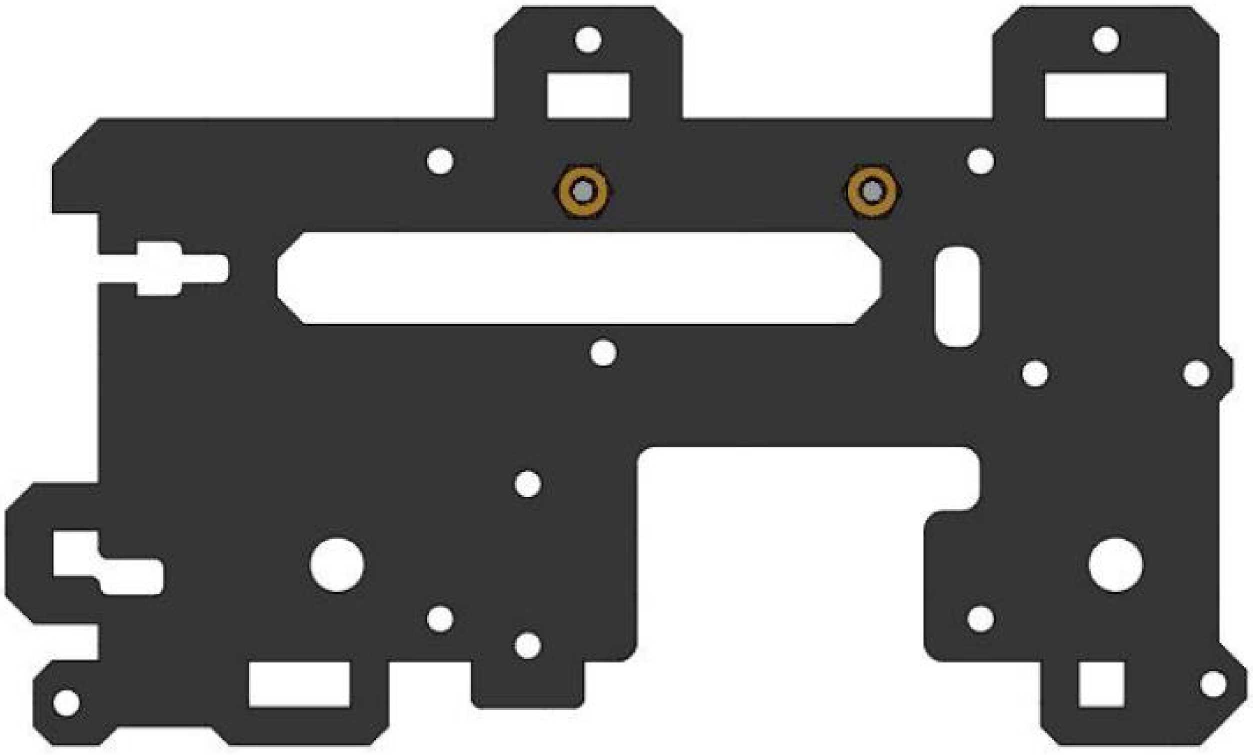
Effect diagram after assembling


Make sure the rocker arm is perpendicular to the se rvo during installation

Assemble the following components

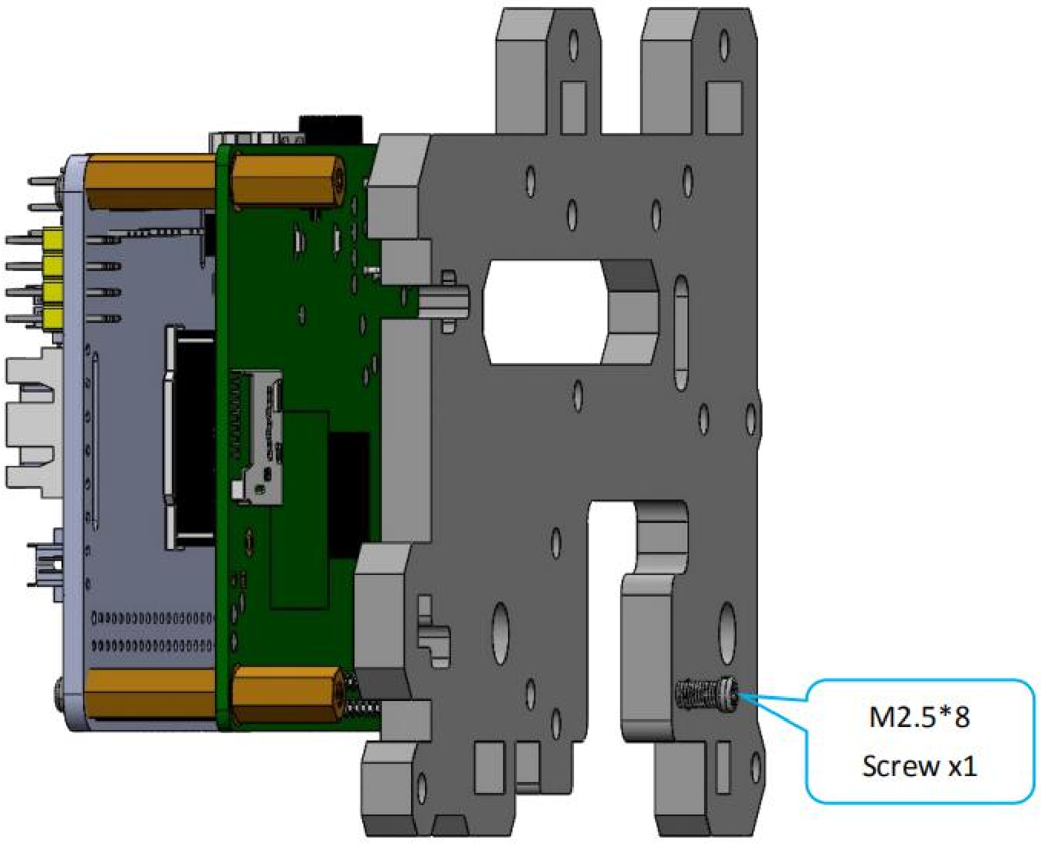


Effect diagram after assembling

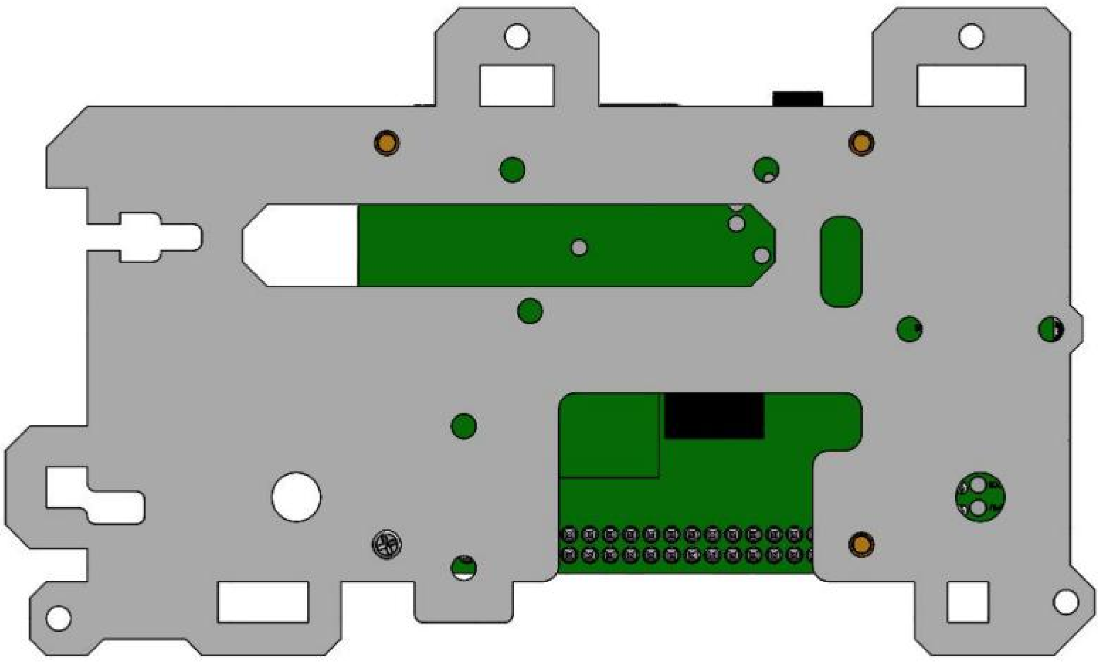


6. Fix motors to the acrylic

Assemble the following components

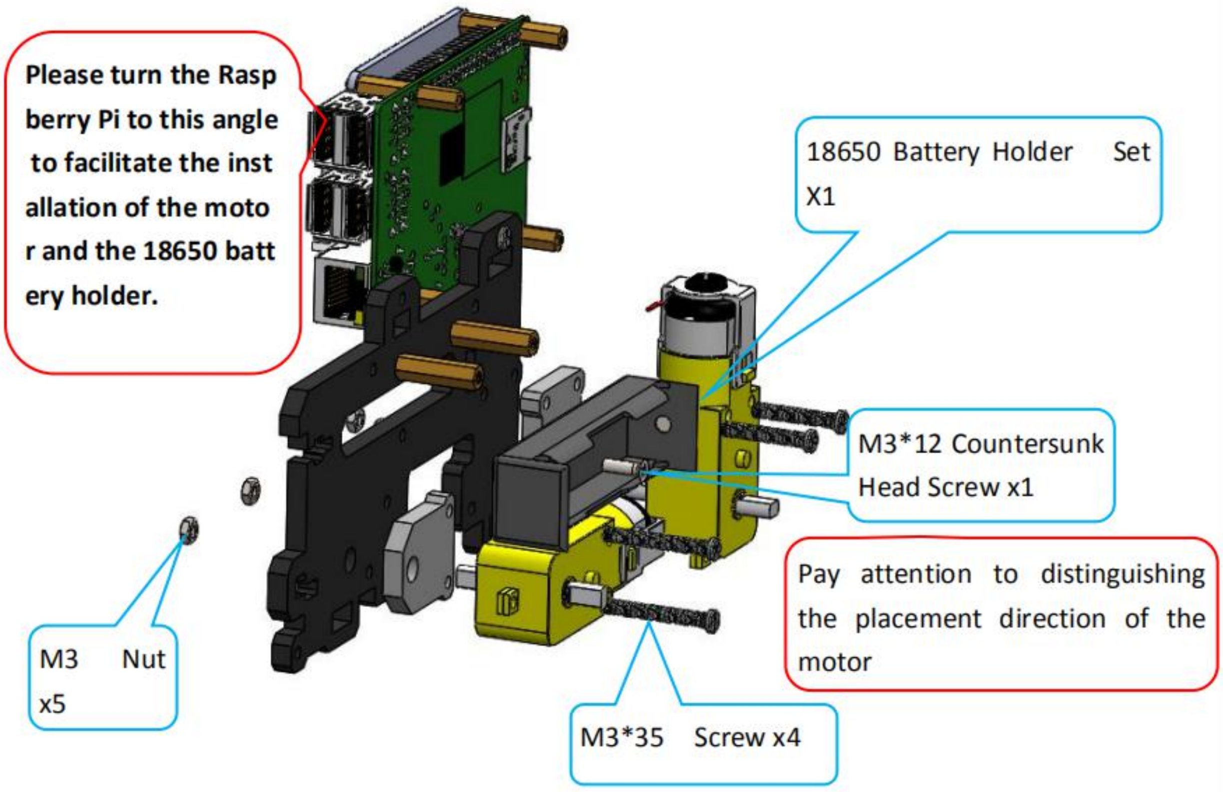


Effect diagram after assembling

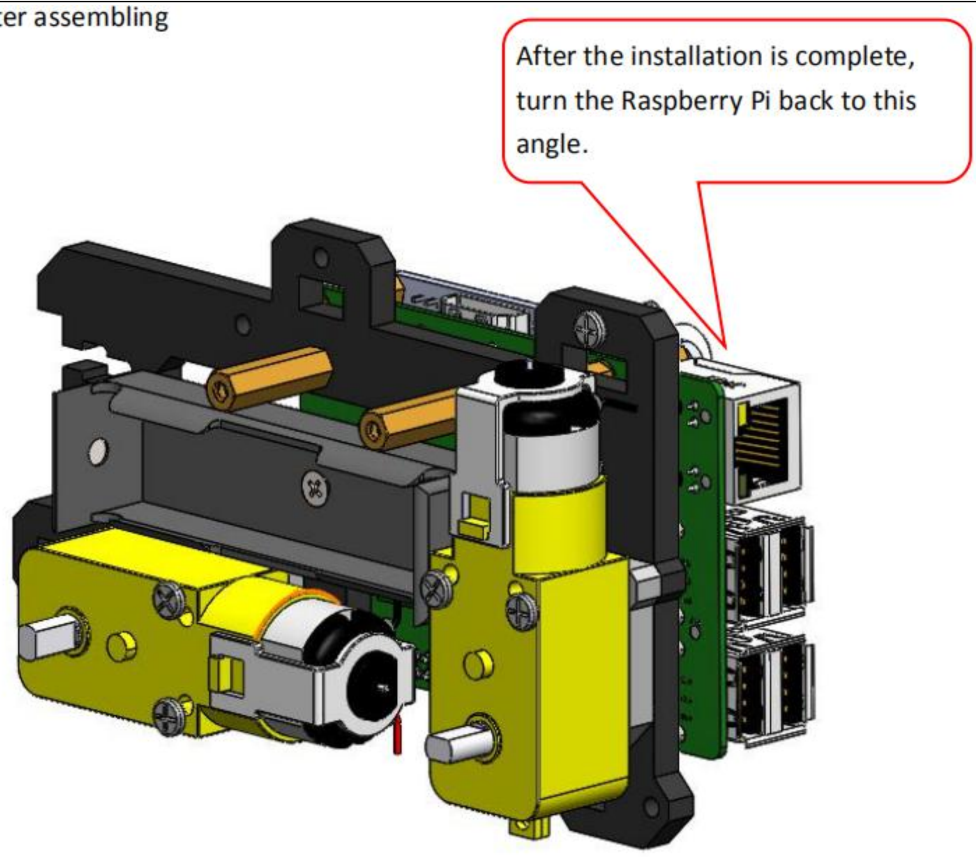


7. Fix one section of the 18650 Battery Holder Set to acrylic

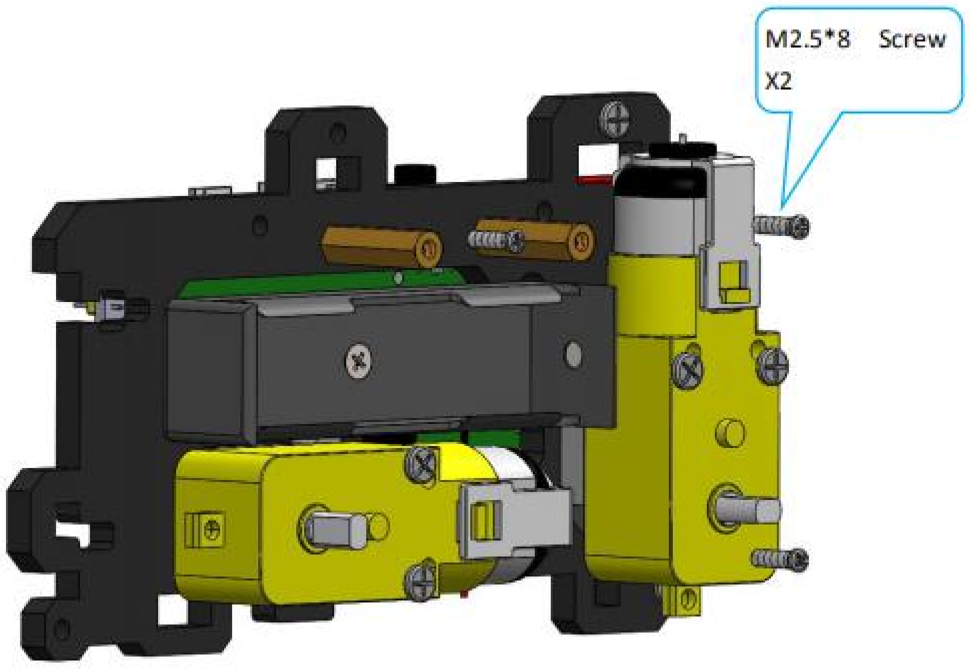
Assemble the following components


Please turn the Raspberry Pi to this angle to facilitate the installation of the motor and the 18650 battery holder.
18650 Battery Holder Set X1
M3*12 Countersunk Head Screw x1
Pay attention to distinguishing the placement direction of the motor
M3*35 Screw x4
M3 Nut x5

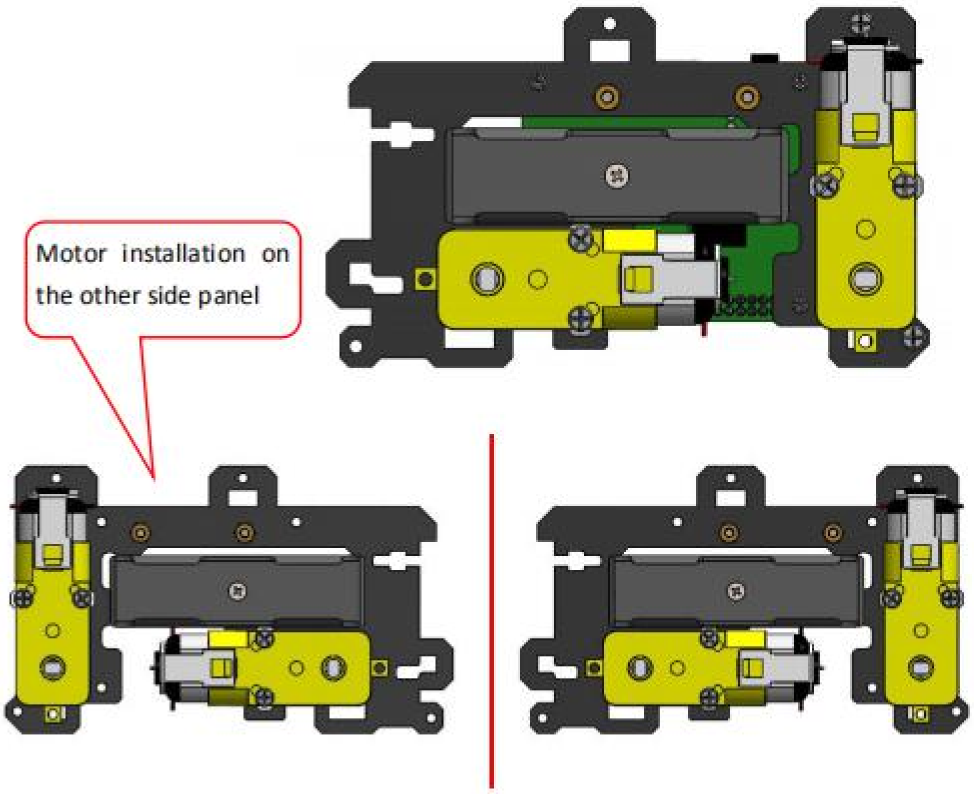
Effect diagram after assembling


After the installation is complete, turn the Raspberry Pi back to this angle.

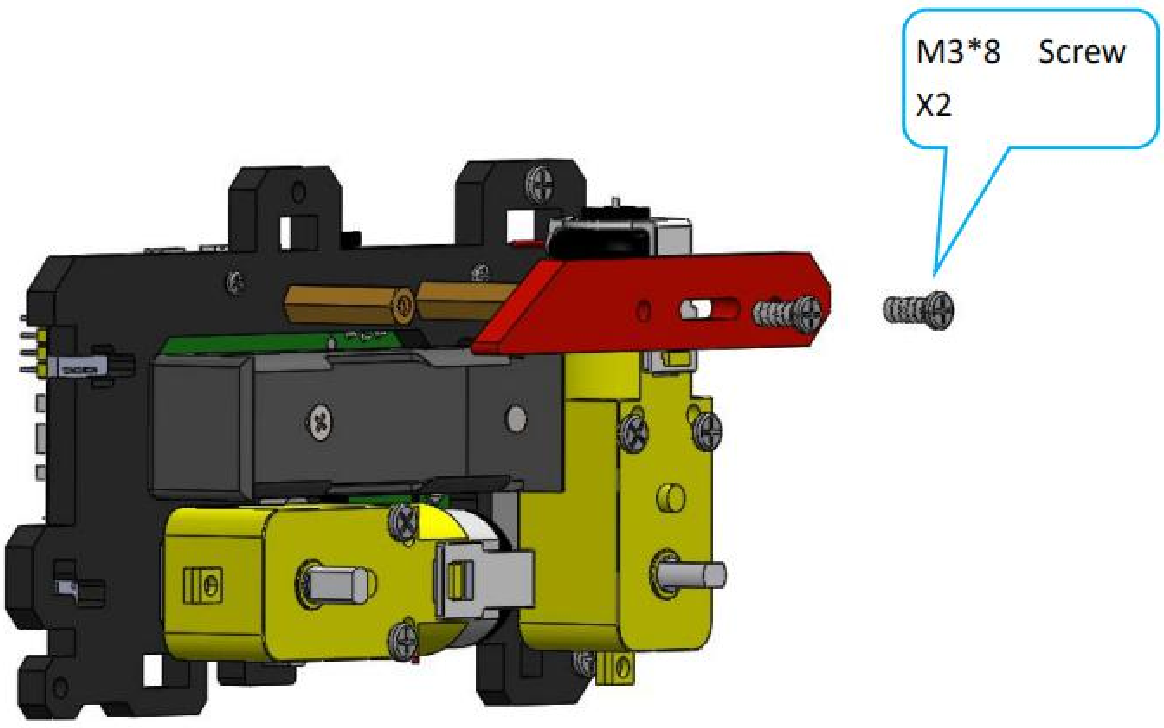
Assemble the following components



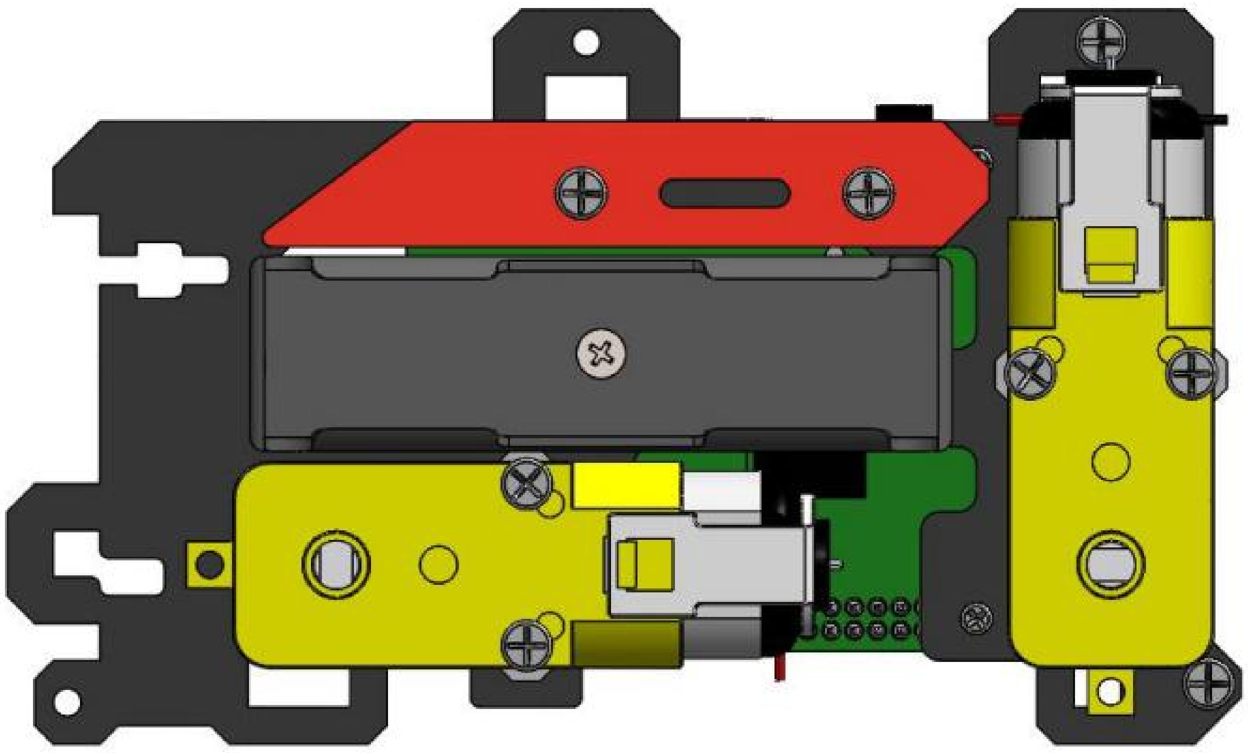
Effect diagram after assembling



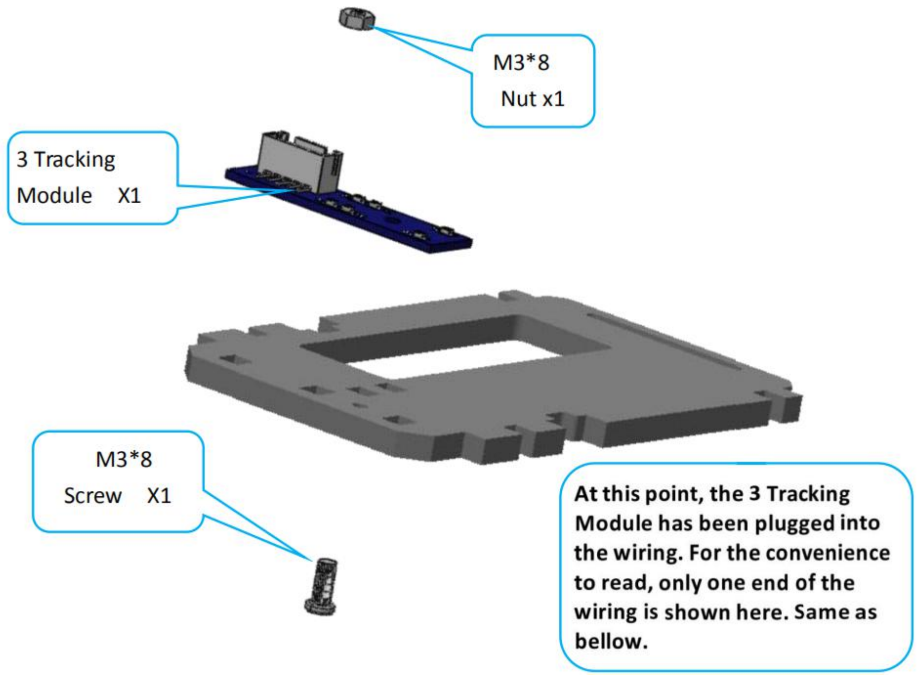
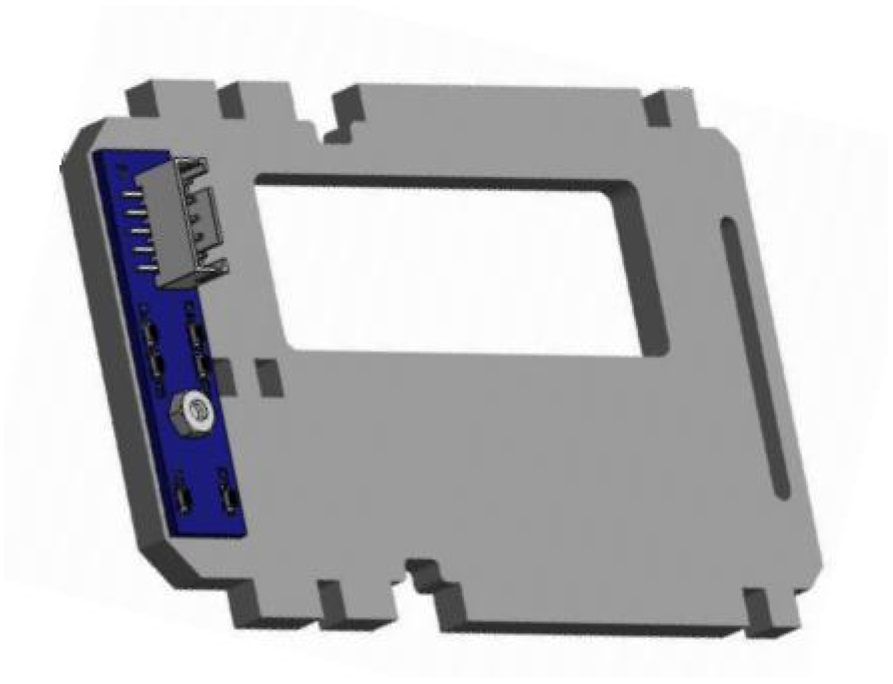
Assemble the following components



Effect diagram after assembling

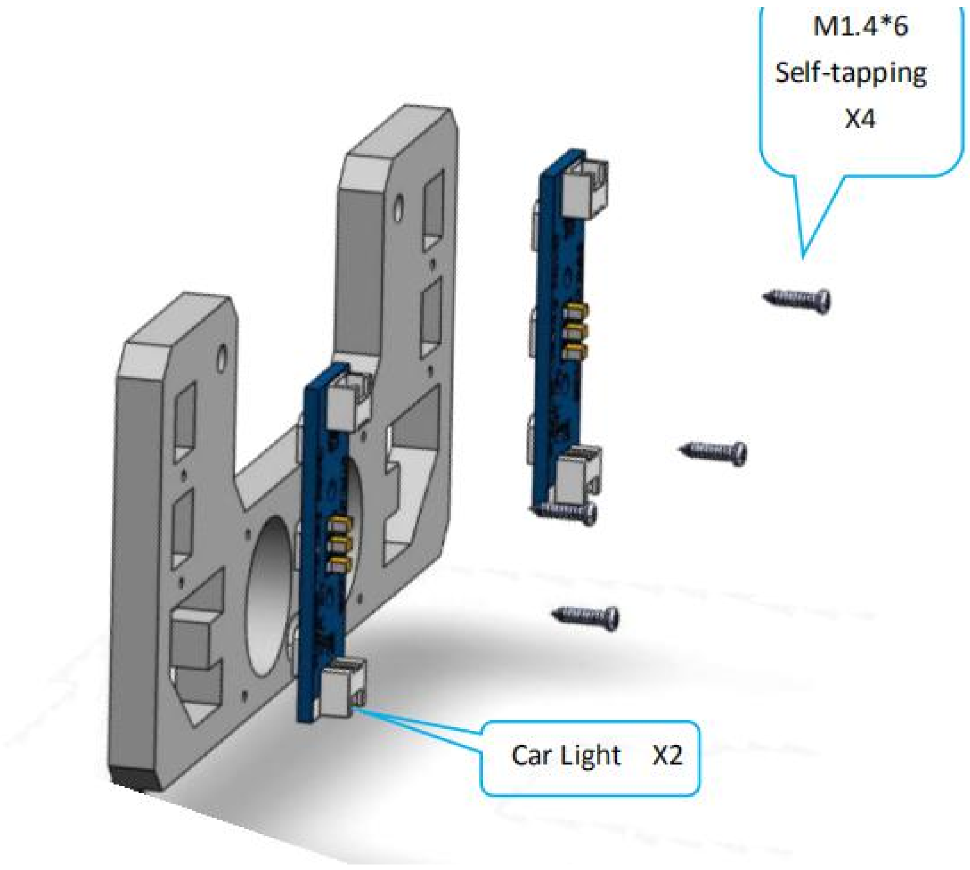


8. Fix 3 Tracking Module on acrylic

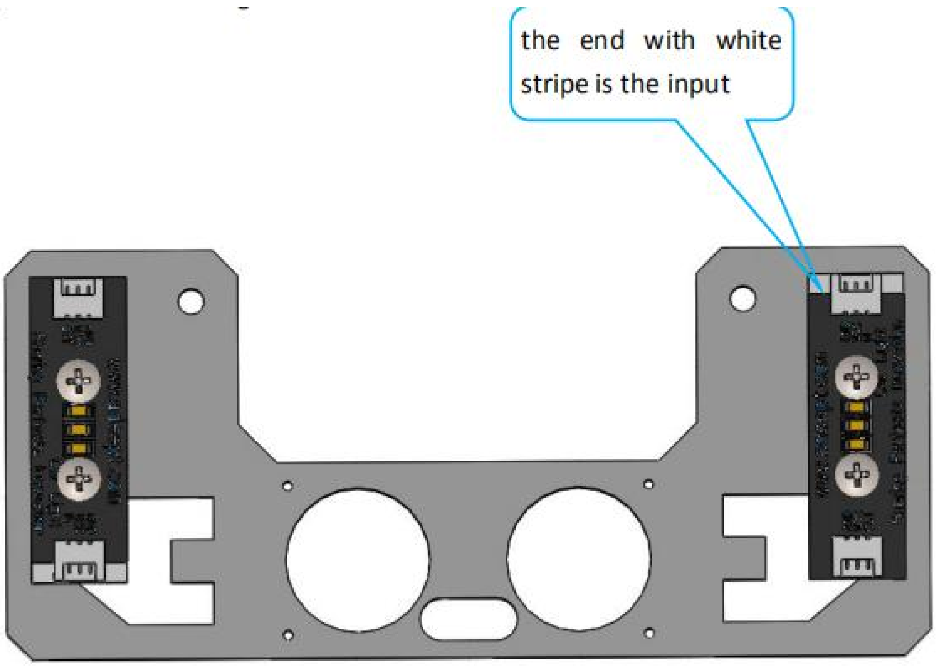
Assemble the following components  M3*8 Nut x1 3 Tracking Module X1 M3*8 Screw X1 At this point, the 3 Tracking Module has been plugged into the wiring. For the convenience to read, only one end of the wiring is shown here. Same as bellow.
Effect diagram after assembling 

9. Fix two Car lights acrylic plate

Assemble the following components

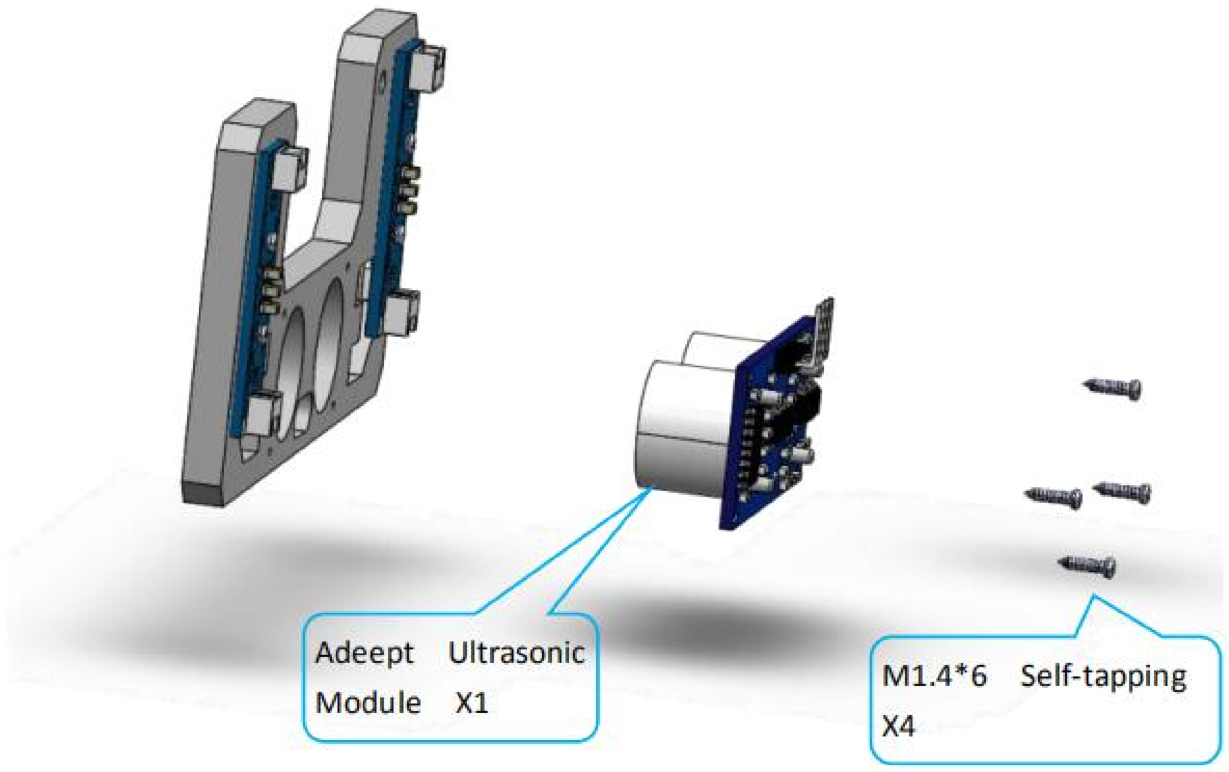


Effect diagram after assembling

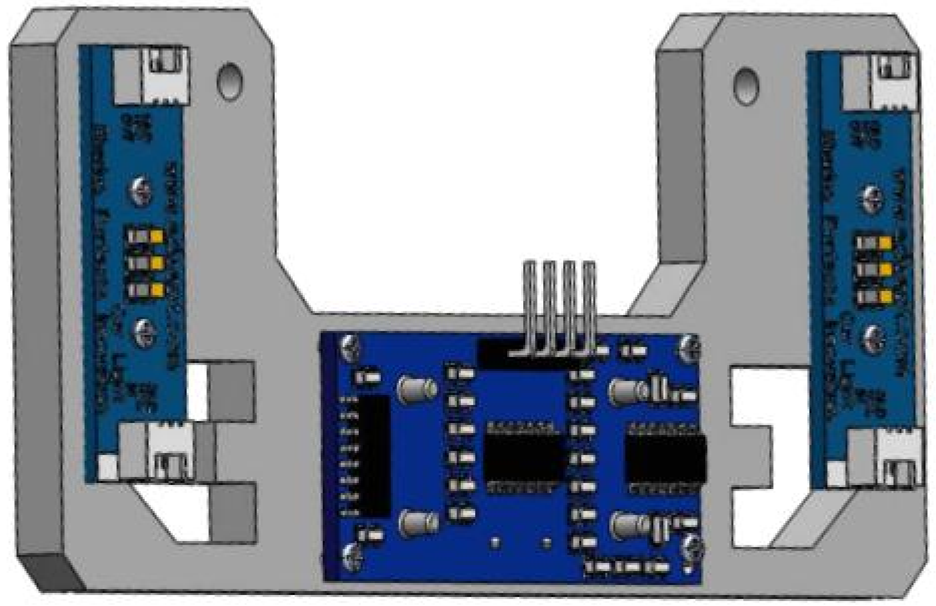


10. Fix Adeept Ultrasonic Module on the acrylic plate.

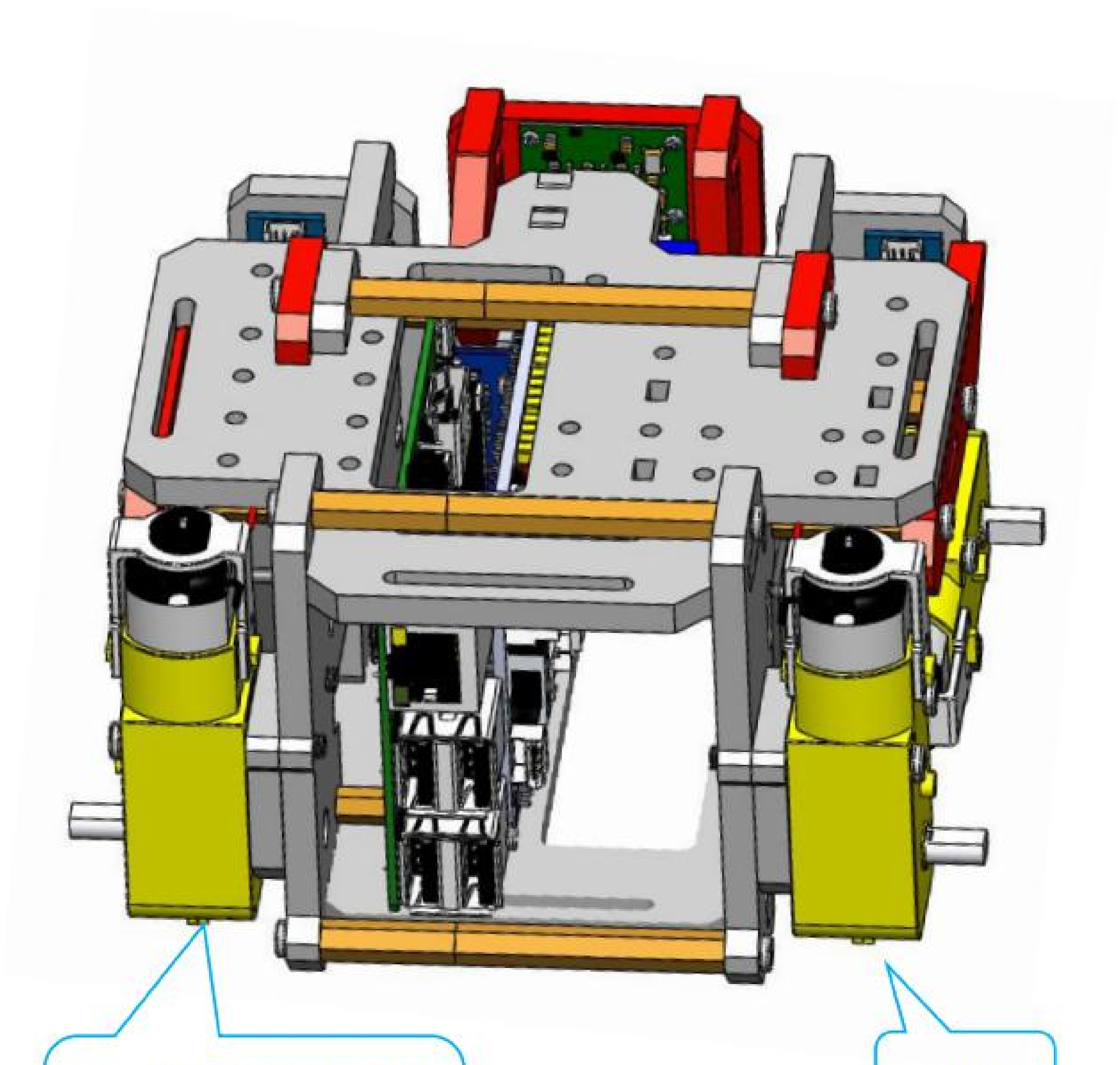
Assemble the following components



Effect diagram after assembling



Pay attention to the distinction between motor A and B, and pay attention to the direction of the motor placement.

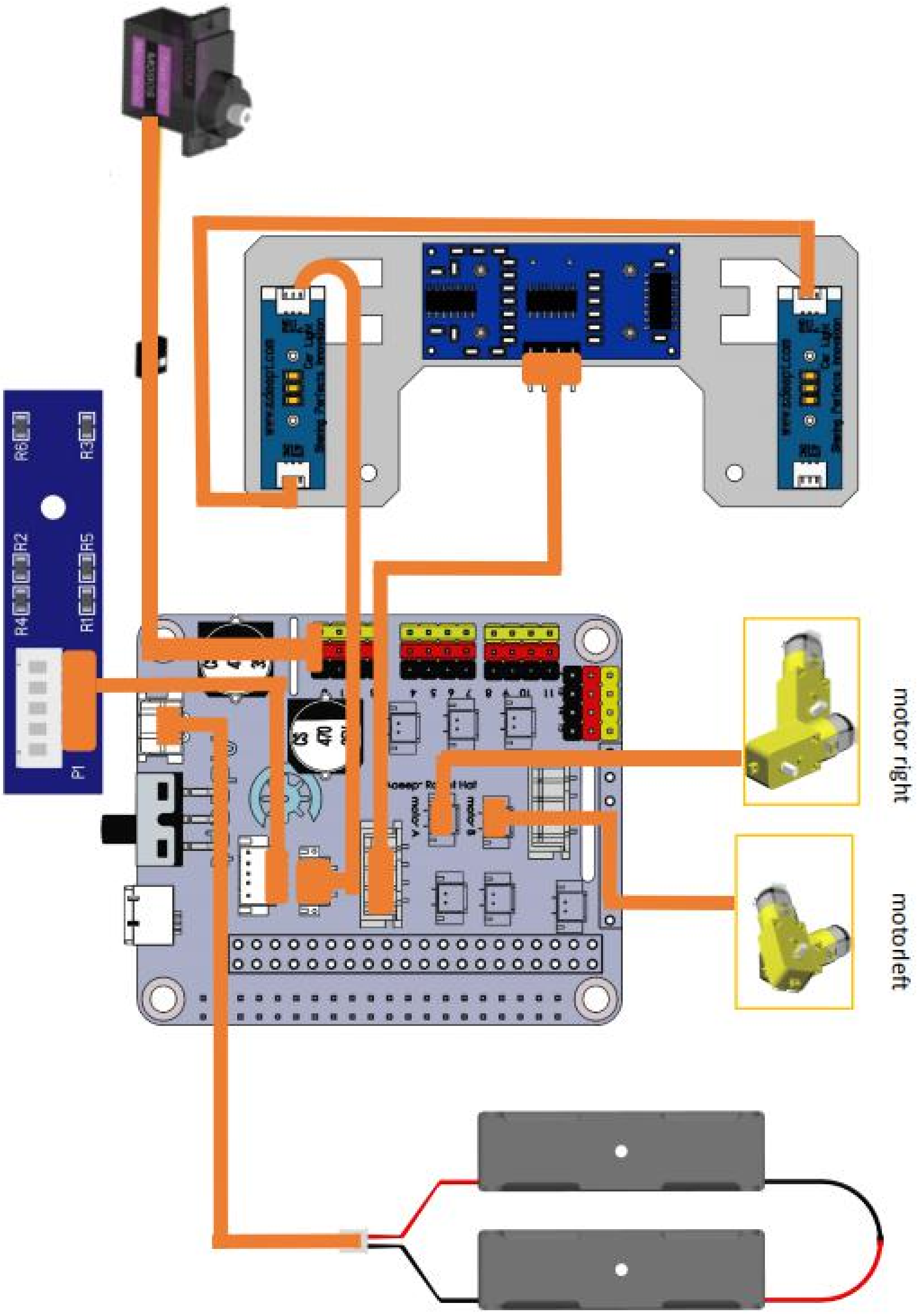


The motor on this side should be connected to Motor-B on Adeept Motor Hat

Motor A

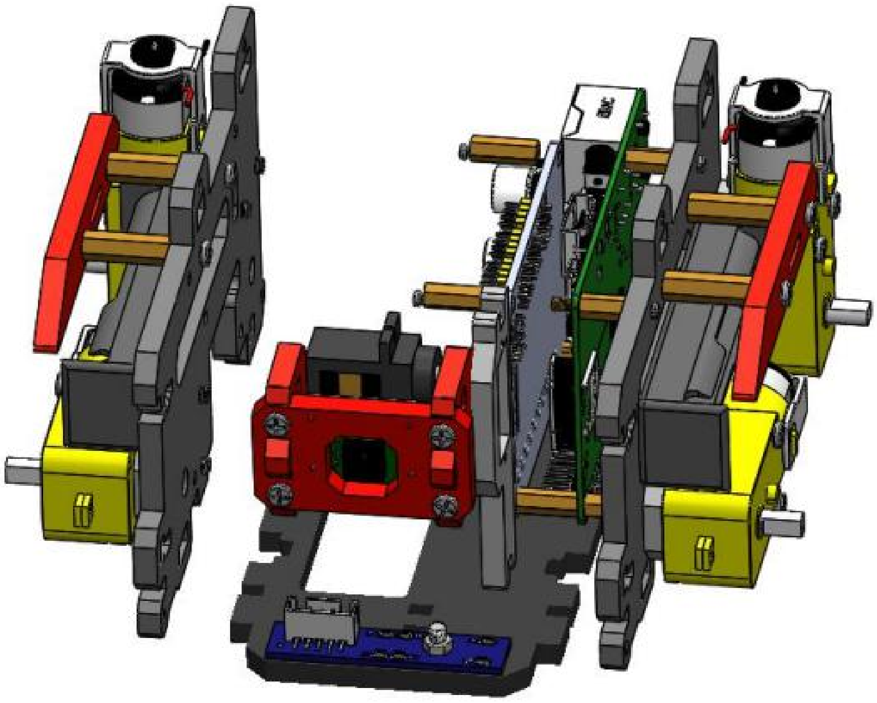
11. Connect the Adeept Ultrasonic Module, Car Light, 3 Tracking Module Set and motor as shown below before assembling the body part.

Assemble the following components

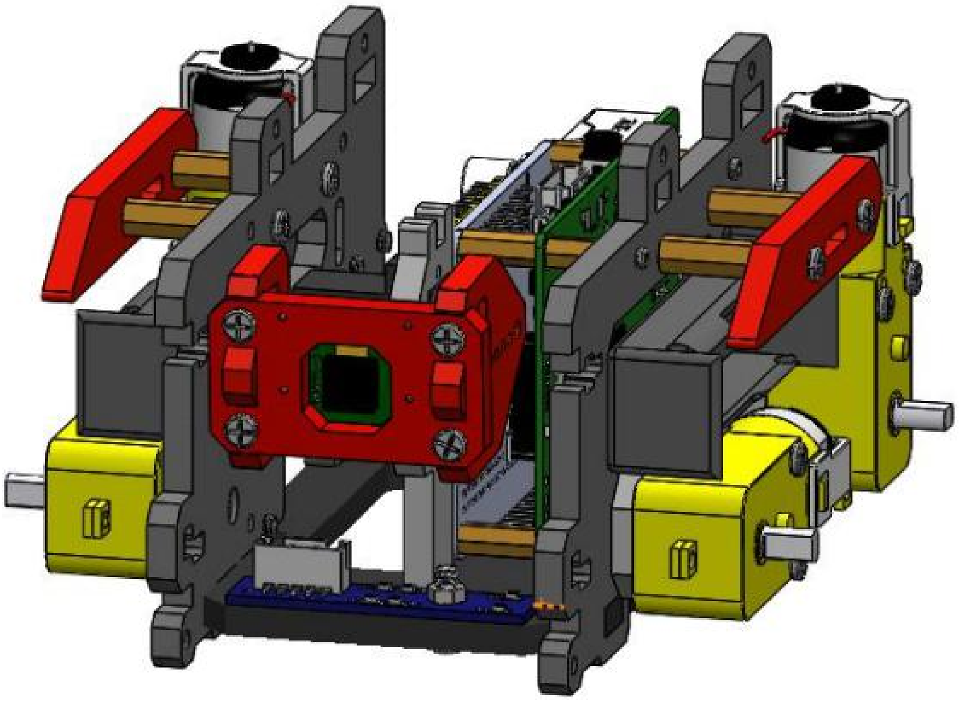


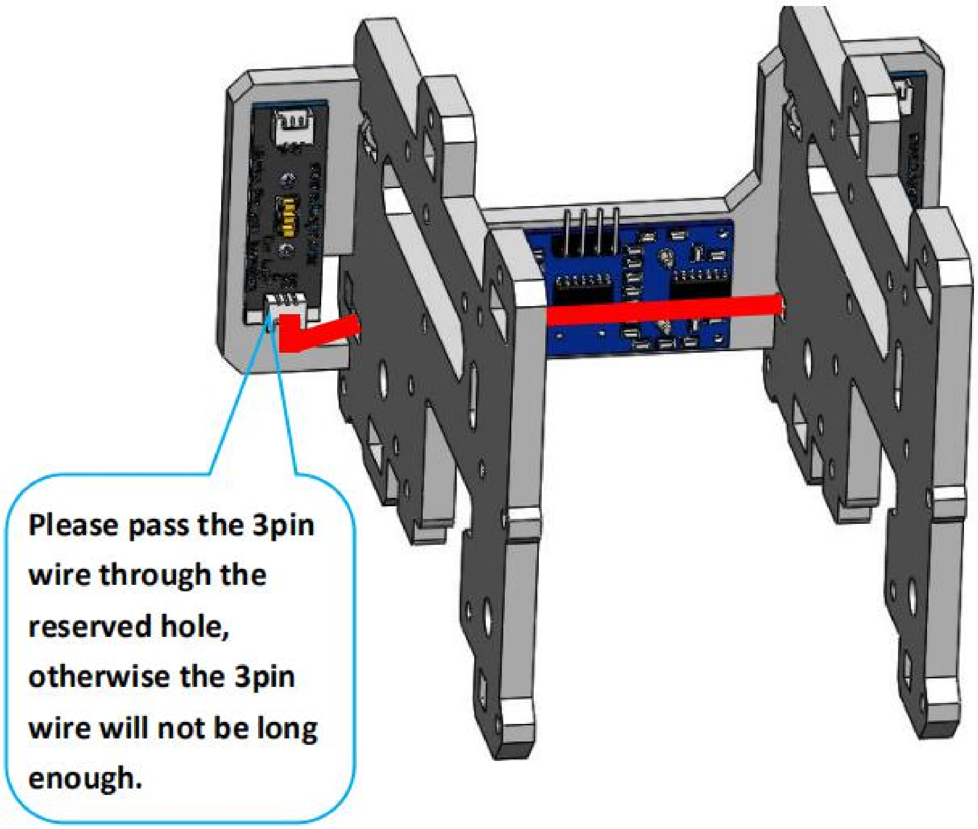
12. Assemble the body part.

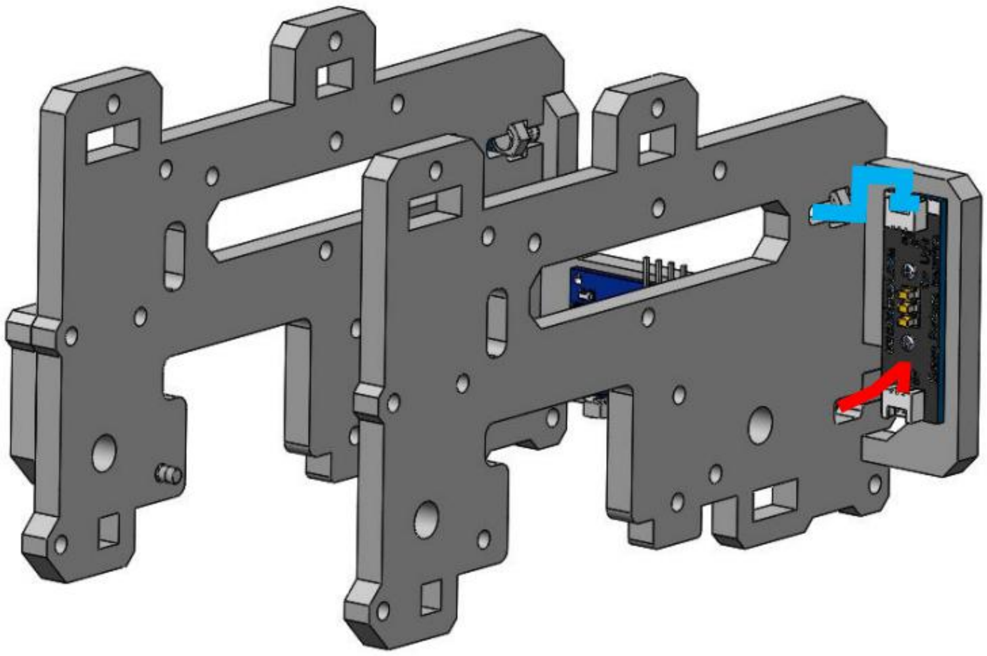
Assemble the following components



Effect diagram after assembling

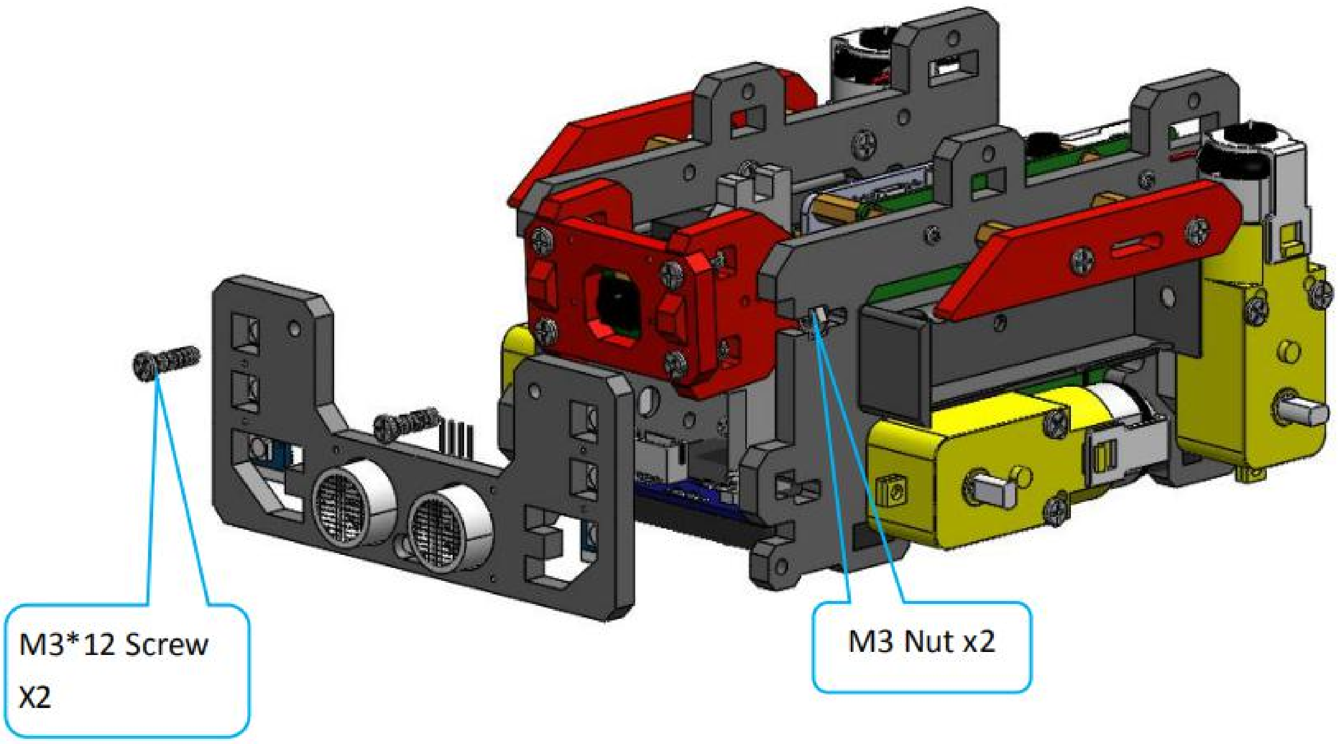



Please pass the 3pin wire through the reserved hole, otherwise the 3pin wire will not be long enough.

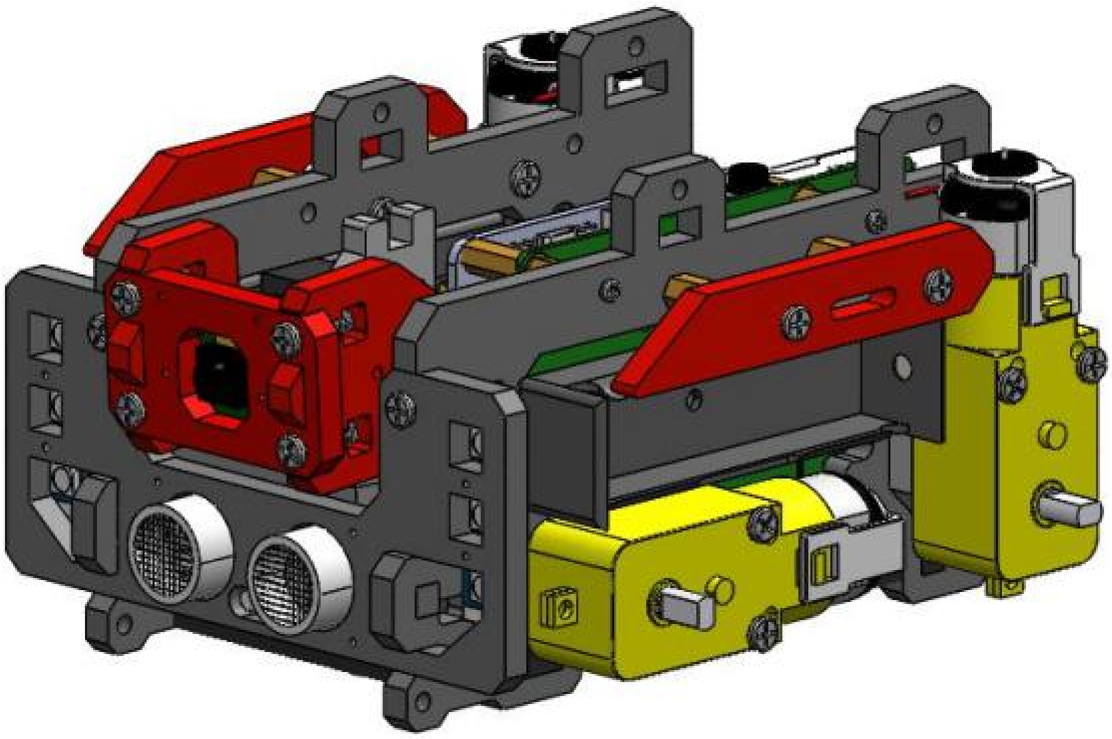


13. Connect the front part and the body part.

Assemble the following components



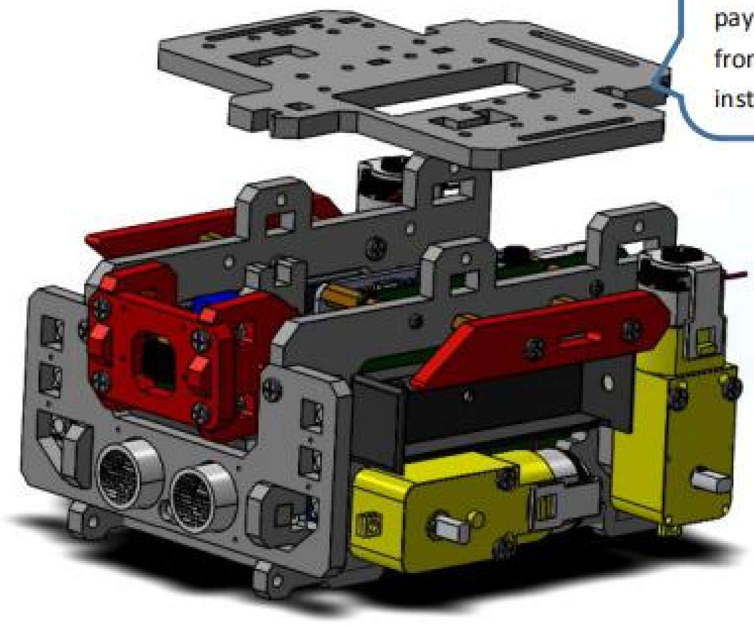
Effect diagram after assembling



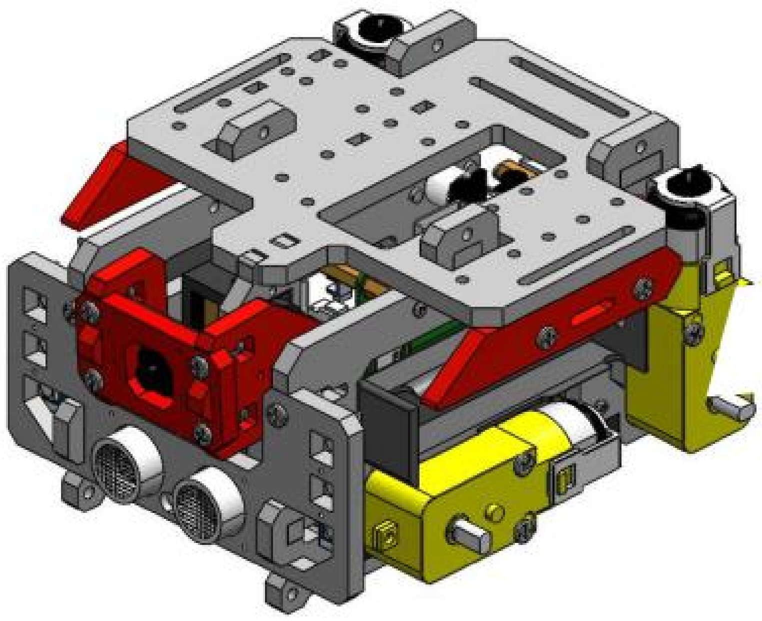
Assemble the expansion board on top.

Assemble the following components

This acrylic plate is asymmetrical, please pay attention to the front and back when installing

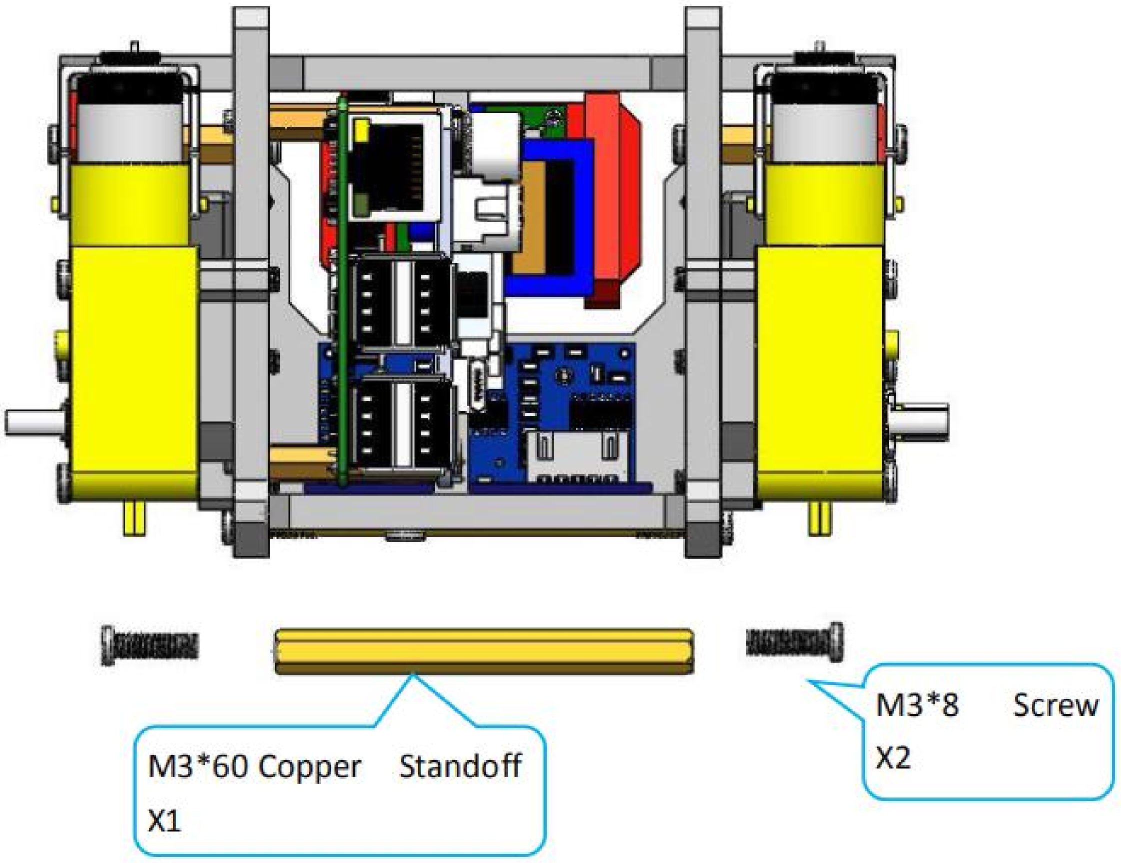


Effect diagram after assembling

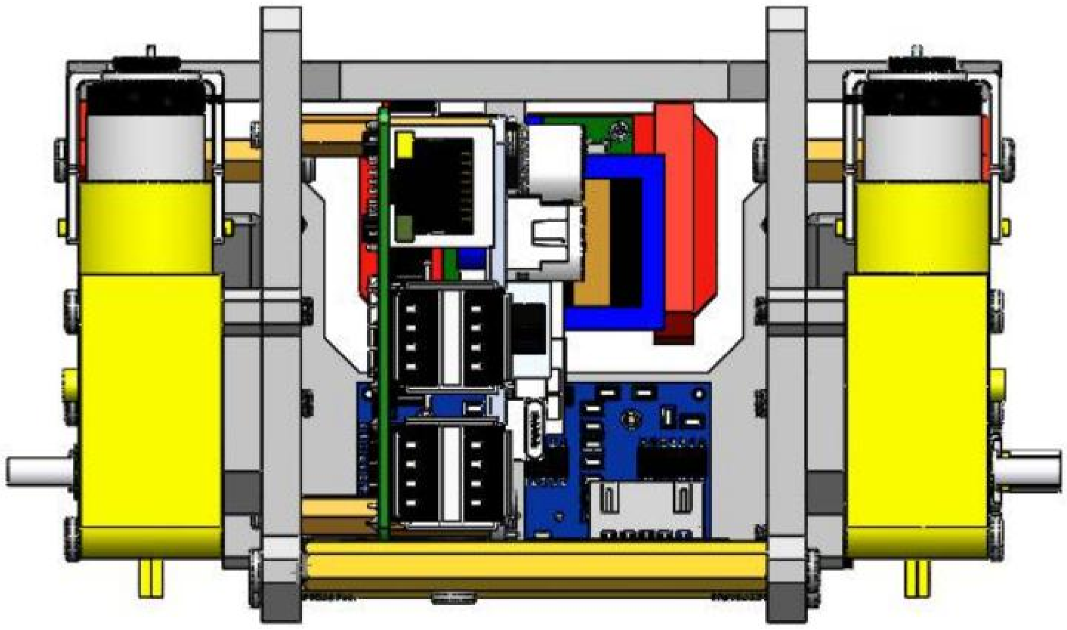


14. Strengthen the body part.

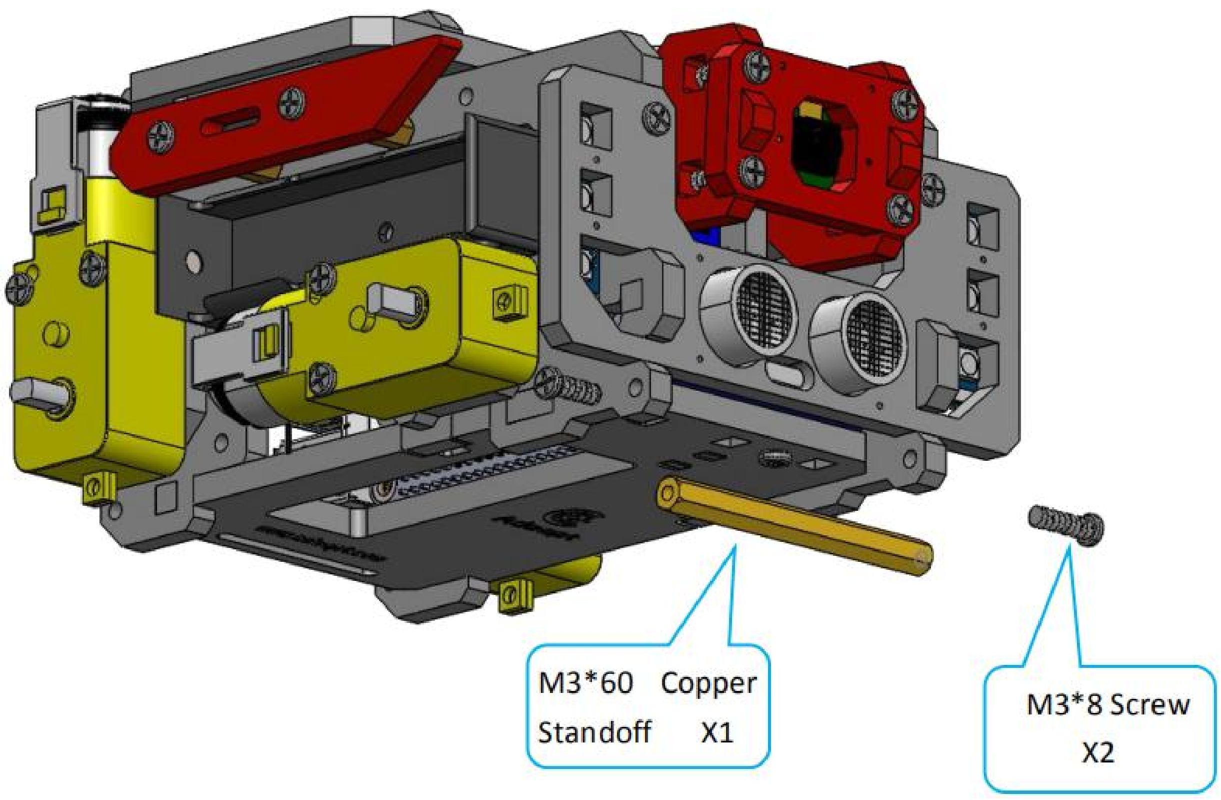
Assemble the following components



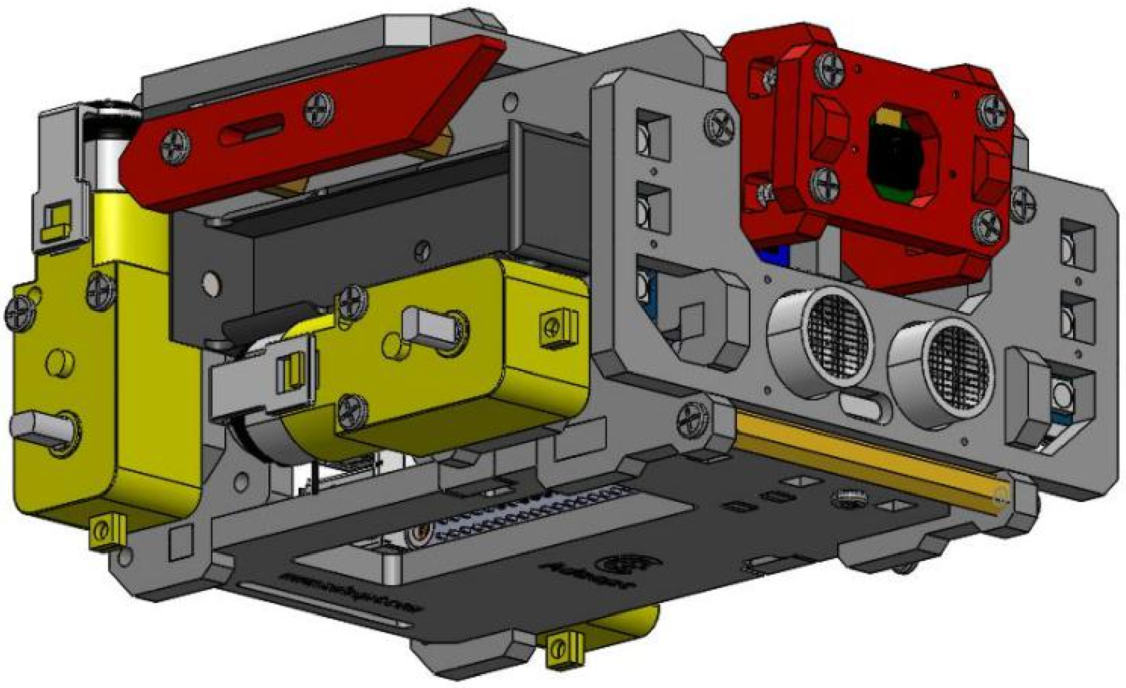
Effect diagram after assembling

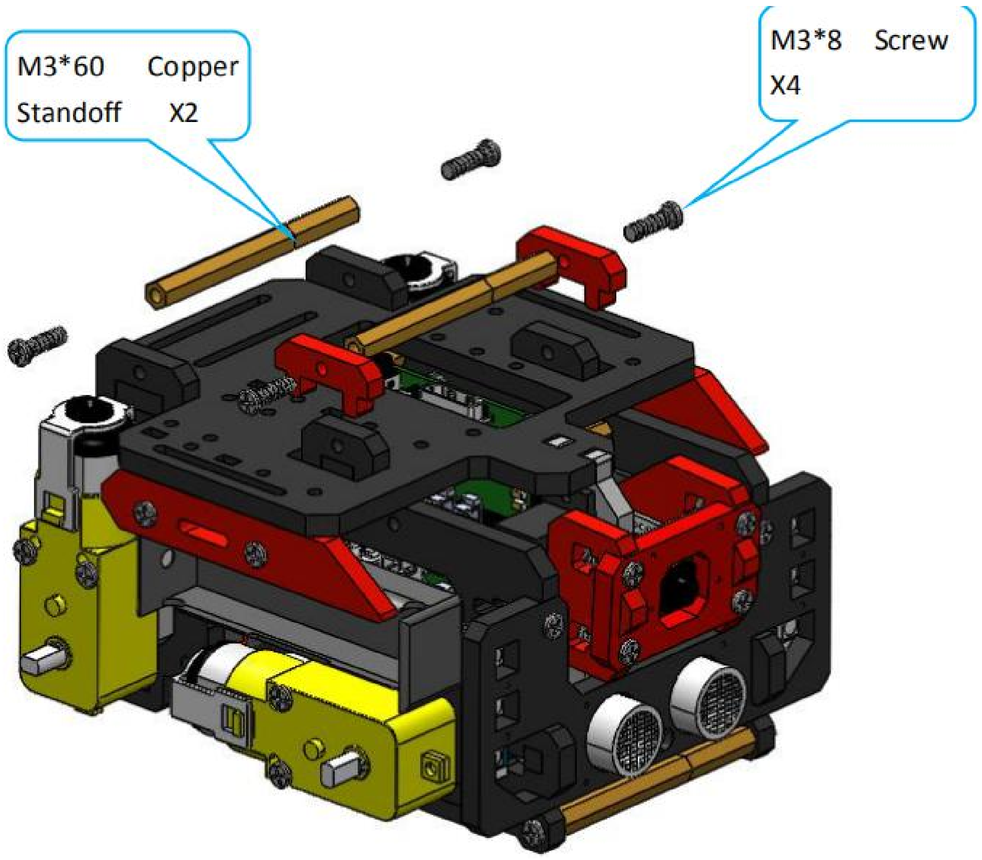
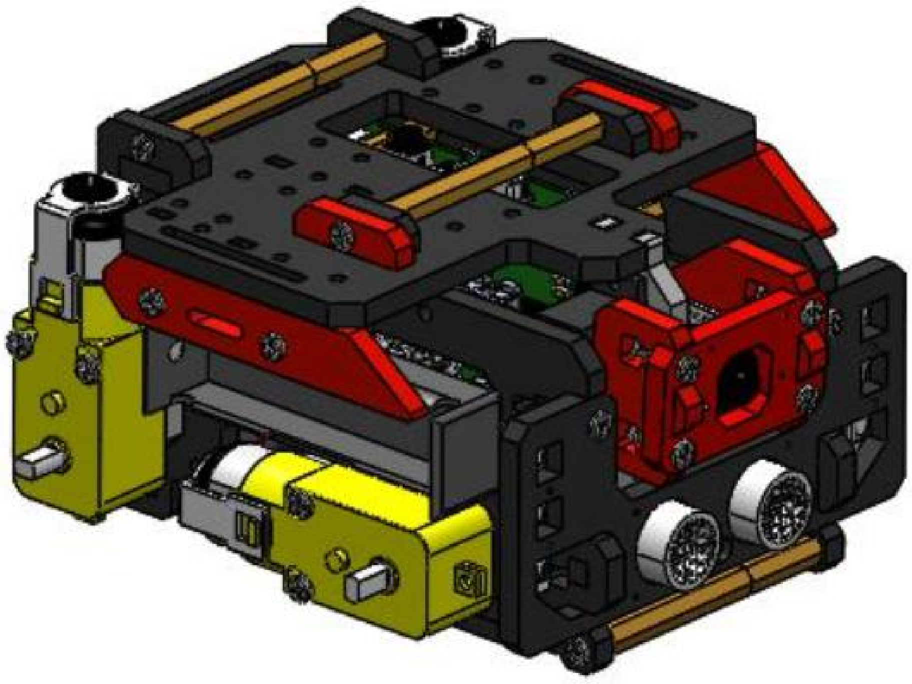


Assemble the following components



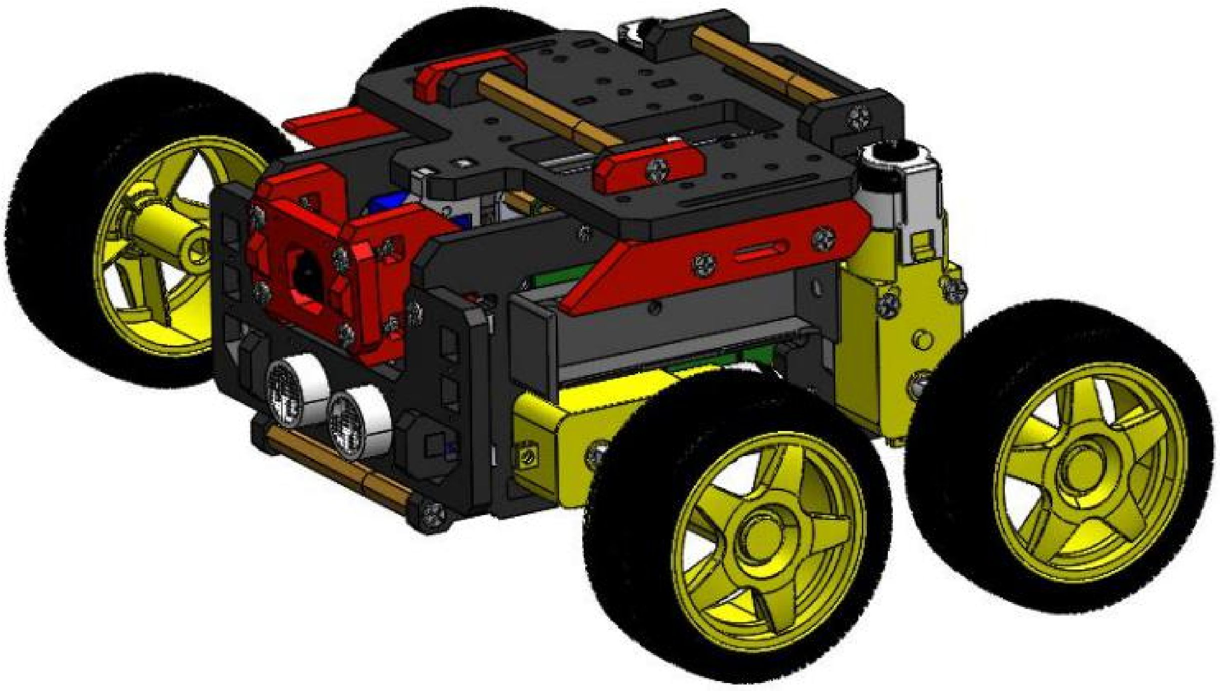
Effect diagram after assembling



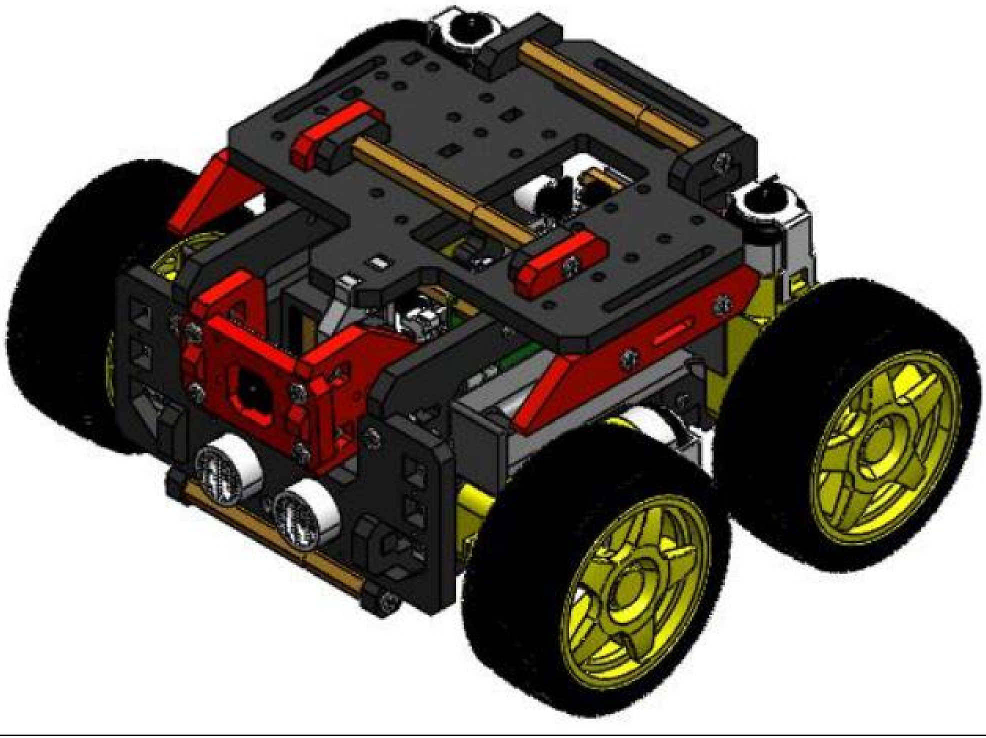
Assemble the following components 
Effect diagram after assembling 

15. Install the wheels on the car.

Assemble the following components



Effect diagram after assembling



4.2 Tips for Structural Assemblage

●Since many servos are used in the product, the servo installation is critical for the robot. Before installing the rocker arm to the servo, you need to connect the servo to power and make the servo shaft rotate to the central position, so the rocker arm installed at the designated degree will be in the central position.

●Generally Raspberry Pi will auto run `webServer.py` when booting, when `webServer.py` will control all the ports connected to servos to send a signal of rotating to the central position. When assembling the servo, you can connect it to any servo port anytime. After connecting the servo to the port, the gears will rotate to the central position; assemble the rocker arm to the servo, disconnect the servo from the port, and insert more servos to repeat rocker arm assembly (all servos will be in the central position).

●When the servo is connected to power, try moving the rocker arm. If it can't be moved, it indicates the program for the servo works; otherwise there's error for the servo program. Run the line `[RobotName]/initPosServos.py` (replace `[RobotName]` with the folder name of your robot's program) to make the servo rotate to the central position.

●When booting (it may take 30-50s), it takes a while for the Raspberry Pi to control PCA9685 to set signal of all servo ports for central position rotating.

4.3 Tips for Power Provision

●When you install the software, assemble the structure and debug the program, you can use a USB cable to power the raspberry pi. If the raspberry pi is installed with Motor HAT, you can connect the USB cable to the USB interface on the Motor HAT. Motor HAT will supply power to the raspberry pi through the GPIO interface.

●The demand of different raspberry pi for electric current is different. For example, the raspberry pi 3B needs at least 2A of current to start up, and the raspberry pi 4 needs 3A to start up normally. You can check the specifications on your power adapter before you use the power adapter to power the raspberry pi.

●When Motor HAT is connected to a load (such as connecting to a motor or multiple servos), you need to use a power supply that supports high current to connect to **Vin** on Motor HAT. You can use two 18650 batteries that support high current to power Motor HAT. Our product will provide a dual 18650 battery box with 2pin interface, you can directly connect it to Motor HAT.

●When you use the USB interface on Motor HAT to supply power, Motor HAT's switch does not control whether to supply power, Motor HAT's switch can only control the power supply of **Vin**.

●Do not use the USB interface and Vin on the Motor HAT to supply power at the same time. If you need to debug the program for a long time and do not want to remove the battery, you can set the switch on the Motor HAT to OFF. So, when using a USB cable to connect Motor HAT, Motor HAT is powered by USB.

●If your robot restarts automatically after starting up or the robot disconnects and restarts suddenly when it starts moving after been started up normally, it is most likely because your power supply does not output enough current. When the robot is starting up, it will automatically run the program to put all the servos in the neutral position. The voltage drop caused by this process causes the raspberry pi to restart.

●We have tested that when using 7.4V power supply, the peak current of the robot is about 3.75A, so you need to use a battery that supports 4A output.

●You can also use a power lithium battery to power Motor HAT. Motor HAT supports the power supply that is below 15V.

●You can use a USB cable to supply power to Motor HAT when the installing the rocker arm of the servo during structural assembly. After the robot software is installed, the raspberry pi will control Motor HAT to set all servo ports to output the neutral signal after it is started up. At this time, you can connect the servo to any servo port, the servo gear will turn to the neutral position, and then you can install the rocker arm of the servo according to the specified angle. After the rocker arm is installed, the servo can be disconnected from Motor HAT. When you need to install the rocker arm of the second servo, you only need to connect the second servo to any servo port on the drive board.

❖ 5 Controlling Robot via WEB App

● The WEB app is developed for common users to control the robot in an easier way. It's convenient to use WEB app; you may use it to wirelessly control the robot on any device with a web browser (Google Chrome was used for testing).

● Generally Raspberry Pi will auto run `webServer.py` when booting and establish a web server in the LAN. You may then use any other computer, mobile or tablet in the same LAN to visit the web page and control the robot.

● How to tell whether the robot has run the `webServer.py` or not: If the WS2812-LED lights up with the breathing effect, it means the robot has booted and runs the program automatically.

● If the program is not run when the robot is booted, try to connect Raspberry Pi via SSH, manually run `webServer.py` with code and check the errors. Refer to the **Q&A** below or email us for help (before manually running `webServer.py`, you need to end the program possibly auto run in the back end to release resources.

```
sudo killall python3
sudo python3 [RobotName]/server/webServer.py
```

● If the `webServer.py` is auto run successfully, open a web browser (here Google Chrome), type in the IP address of the Raspberry Pi, with `:5000` added to the end, and go to the next step, as shown below:

```
192.168.3.157:5000
```

● If no image is displayed, try manual running `webServer.py` as described in the step above.

● If image is shown, you can control the robot to move now. You may check the description for keyboard shortcuts `**Instruction**` at the bottom and control the robot based on its general functions with the keyboard.

● `**Video**` window shows the image captured by the robot's camera in real time.

● `**Move Control**` window is to control the basic movements of the robot.

● `**Arm Control**` control the rotation of the robot Platform Servo.

• `**UP**`: control the Platform Servo to turn upward.

• `**DOWN**`: control the Platform Servo to turn down.

● `CVFL Control` window is to control the visual line following function of the robot. Here only an overview for the function is described; more details will be provided in the OpenCV section:

·`**START**`: Enable or disable the visual line following function.

·`**COLOR**`: Switch between white and black line following. By default the robot follows white lines; click the button to switch to black line following.

·The line following function analyzes two pixels in parallel and utilizes the information detected; the positions of these two pixels are `**L1**` and `**L2**`.

·`**SP**` is the threshold of the turning command based on the visual analysis results. A bigger `SP` value means a big deviation; though a particularly small `SP` value may stop the robot from moving as it can't aim the target and find the direction.

● When the visual line following function is enabled, the video screen will automatically become binarized results, making the visual analysis clearer.

● `**Hard Ware**` window displays CPU temperature, CPU occupancy rate, and memory usage of the Raspberry Pi.

● `**Actions**`window control unique functions of the robot:

• `**MOTION GET**`: Motion detection function based on OpenCV. When objects move in the view of the camera, the program will circle the part in the `Video` window, and the LED light on the robot will show respective changes.

• `**AUTO MATIC**`: Obstacle avoidance function based on ultrasonic. When the ultrasonic module on the robot detects an obstacle, it will automatically turn left, and take a step backward before turning if it's too close to the obstacle.

• `**SPEECH**` Based on the control function of the multi-threaded WS2812-LED, the color of the robot's WS2812-LED lamp can alternately blink.

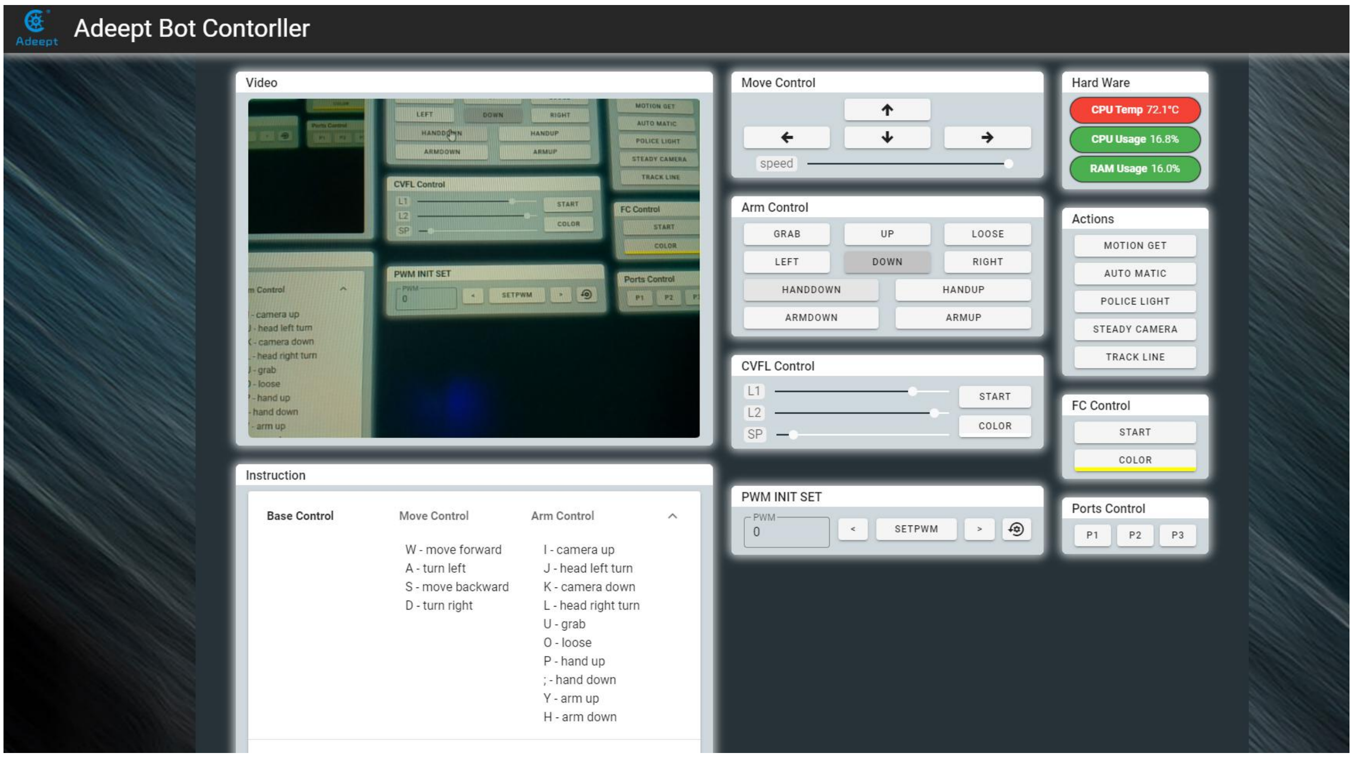
●`**FC Control**` window control`s the color lock function of the robot:

·`**START**`: Enable or disable color searching and tracking function.

·`**COLOR**`: Select the color to track.

·When the function is on, the robot will automatically lock one particular color in the camera view. By default it tracks bright yellow objects. You can change the color as you want. When an object is locked, the LED on the robot will turn orange. The camera of the robot can not only conduct pitching motion, but also can lock objects of a certain color horizontally.

●`**PWM INIT SET**`: Used to adjust the initial angle of each servo port of the robot. If you find that the angle of a certain servo of your robot is not right, you can use this function to fine-tune it to the correct angle.



Base Control | Move Control | Arm Control

W - move forward
A - turn left
S - move backward
D - turn right

I - camera up
J - head left turn
K - camera down
L - head right turn
U - grab
O - loose
P - hand up
; - hand down
Y - arm up
H - arm down

❖ 6 Common Problems and Solutions(Q&A)

● Where to find the IP address of the Raspberry Pi?

Before connecting the Raspberry Pi via SSH, you need to know the IP address of the Raspberry Pi. Check the Management interface for your router, or download the app `Network Scanner` -> search for a device named `RASPBERRY` or `Raspberry Pi Foundation` to get the IP address.

For other methods of obtaining the IP address of Raspberry Pi, refer to the official documentation [IP Address]

● Errors occur with `permission denied` prompt when I manually run `server.py` or `webServer.py`.

The Raspberry Pi needs the root permission to run the dependent libraries for WS2812 LED lights control.

You need to add `sudo` to the beginning of `server.py` or `webServer.py` to run the program.

```
sudo python3 [PATH]/server.py
sudo python3 [PATH]/webServer.py
```

● I can't create the hots pot for the robot.

You need to use the open source project create_ap to setup the robot's hotspot. Prior to use, disconnect WiFi network but DO NOT turn the WiFi module off, or the create_ap will show an error of hardware being blocked.

● The servo rotates to an abnormal degree.

Before assembling the rocker arm and servo, you need to make the servo gears rotate to the central position of its rotating range. Then assemble the rocker arm based on the angle instructed in the documentation. There can be a deviation of less than 9° due to the structure of the servo (number of teeth is 20 for the servo gears). For better performance, you may refer to the servo control documentation for initial degree adjustment by code.

● The servo is shaking.

Probably the servo reducing gear is broken.

● Raspberry Pi can't boot.

Remove all parts on the driver board. Only connect the board to Raspberry Pi and power supply, reboot.

● "Remote side unexpectedly closed network connection" shows on a popup window.

There can be error prompts during installation because the Raspberry Pi will auto reboot after the installation, which will disconnect the board.

● Program crashes after double clicking on client.py or GUI.py.

Run the script by `python client.py` or `python GUI.py` in cmd and check the error reports.

● How to initialize the servo's angle?

If you've finished software installation on the Raspberry Pi, just boot it up and the servo will be initialized.

● I can connect to the Raspberry Pi terminal via SSH \ Raspberry Pi failed to connect a WiFi.

The power supply methods will not influence control by SSH. Check whether you've created the file `wpa_supplicant.conf` for multiple times. If yes, that's problem causing SSH errors.

● Can I supply the Motor HAT and Raspberry Pi via USB?

A 2A output is required for a Raspberry Pi 3B, when at least 3A is needed for a Raspberry Pi 4. You can use the USB power for software installation and testing, but it's not suitable for high power module like servo or motor adjustment as it may result in low voltage. It's recommended to use battery for power here.

● After installation, the robot shows no response when booting.

The `server.py` or `webServer.py` may not run due to some reasons. Try to manually run `server.py` or `webServer.py` and check whether there's any error prompt.

●The servo doesn't return to the central position when connected to the driver board.

In general, the Raspberry Pi will auto run `webServer.py` when booting, and `webServer.py` will run and control the servo ports to send a signal of rotating to the central position. When assembling the servo, you can connect it to any servo port anytime. After connecting the servo to the port, the gears will rotate to the central position; assemble the rocker arm to the servo, disconnect the servo from the port, and insert more servos to repeat rocker arm assembly (all servos will be in the central position).

When the servo is powered on, try moving the rocker arm. If it can't be moved, it indicates the program for the servo works; otherwise there's error for the servo program. Run the line `[RobotName]/initPosServos.py` (replace `[RobotName]` with the folder name of your robot's program) to make the servo rotate to the central position.

When booting (it may take 30-50s), it takes a while for the Raspberry Pi to control PCA9685 to set signal of all servo ports for central position rotating.

●no cv2 error occurs when I manually run `server.py` or `webServer.py`.

OpenCV is not installed correctly. Type in the command sudo pip3 install opencv-contrib-python in the Raspberry Pi to manually install OpenCV.

●When using a computer to copy ssh and wpa_supplicant.conf to the SD card, it prompts that there is no SD card

If this happens, unplug the card reader and connect it to the computer.

●SSH can't connect, error WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!

Enter the following in the command line and press Enter

```
ssh-keygen -R Add the Raspberry Pi's IP address
```

For example:

```
ssh-keygen -R 192.168.3.157
```

Then you can SSH to the Raspberry Pi again

●Raspberry Pi automatically restarts after booting / restart the robot once it starts to move

If your robot automatically restarts after powering on, or disconnects and restarts when the robot starts to move after normal power on, it is likely because your power supply does not output enough current, and the robot will automatically run the program when it starts Put all the servos in the neutral position, the voltage drop caused by this process causes the Raspberry Pi to restart.

We have tested that when using 7.4V power supply, the peak current of the robot is about 3.75A, so you need to use support 4A output battery.

●The direction of servo movement is incorrect

Due to the different batches of servos, when the same angle change trend is given to the servos, the actual direction of motion of the servos may be opposite. We have set an interface to adjust the direction of the servos in the program. You need to open RPIservo.py, Find the array sc_direction in ServoCtrl. If the direction of the servo of port 3 is reversed, change the fourth 1 to -1.

(The serial number of the array starts from zero, so port 3 corresponds to the fourth 1).

If the servo direction of port 3 is not correct:

Before modification:

```
self.sc_direction = [1,1,1,1, 1,1,1,1, 1,1,1,1, 1,1,1,1]
```

After modification (the serial number of the array starts from zero, so port 3 corresponds to the fourth 1):

```
self.sc_direction = [1,1,1,-1, 1,1,1,1, 1,1,1,1, 1,1,1,1]
```

●Motor movement direction is incorrect

Due to the different batches of motors, when the same signal is given, the direction of rotation of the motor may be different. We have set an interface to adjust the direction of rotation of the motor in the program. You need to open move.py. In the program part, you can see To the following variable definitions:

```
Dir_forward     =    0
Dir_backward =  1
left_forward    =    0
left_backward = 1
right_forward   =    0
right_backward  =    0
```

If all your motor actions are reversed, just change Dir_forward = 0 to Dir_forward = 1,Just change Dir_backward = 1 to Dir_backward = 0.

If you only have one motor reversed, you only need to change the corresponding set of variables.

●After running the server, I get an error and can't find config.txt

This is because the installation script did not copy con fi g.txt to the specified location due to permissions problems during installation. The new version of webServer will not use this file, only the old version of the server will use it. Copy the server folder of the Raspberry Pi to / etc / of the Raspberry Pi, use the following command

```
sudo cp -f //home/pi/adeept_PiCar-B/server/config.txt //etc/config.txt
```

Just replace adeept_PiCar-B above with your product name, we will take PiCar-B as an example here.

●Running GUI.py reports ip.txt NotFound error

The correct mode of operation is to execute GUI.py directly in the GUI directory. If you execute GUI.py in another directory, it will look for ip.txt in your current directory.

❖ 7 Set The Program to Start Automatically

7.1 Set The Specified Program to Run Automatically at Boot

●This section only introduces the auto-run method used by our products. If you need more information about the Raspberry Pi auto-run program, you can refer to this document from itechfythe document Auto-Run.

●If you have used the operation steps of **3.5** or **3.6**, then the script program has been configured to automatically run the program at startup. In this chapter, we explain how to set a program to start automatically at startup from scratch.

●First we use the following code to create a new startup.sh:

```
sudo touch //home/pi/startup.sh
```

●Edit startup.sh

```
sudo nano startup.sh
```

●Write the following content in startup.sh, where python3 is followed by the program you want to run automatically. Note that you must use an absolute path here. Let's take webServer.py as an example.

```
#!/bin/sh
sudo python3 [RobotName]/server/webServer.py
```

●After **Ctrl** + **X** To exit editing. Press Y Save. **Enter** Confirm and exit editing.

●Give startup.sh permissions, where *** is the Linux permission code, we do not recommend the use of permissions such as 777, but for novices 777 can avoid many account and permissions problems, of course, you can also set it to 700 , So that only the owner can read, write and execute startup.sh, you can learn more about Linux permissions through this article from maketecheasier the article link Understanding File Permissions.

```
sudo chmod 777 //home/pi/startup.sh
```

●Edit rc.local to configure the script to run automatically

```
sudo nano /etc/rc.local
```

●Add the following content under **fi** in the original document, save and exit:

```
//home/pi/startup.sh start
```

● Of course, you can also replace the above script file path with other scripts you want to run automatically.

7.2 Change The Program That Starts Automatically

●After step 7.1, you can already set the program to run automatically at boot. If you want to change the program to run automatically at boot, just edit startup.sh:

```
sudo nano //home/pi/startup.sh
```

●For example, if we want to replace webServer.py with server.py, we only need to edit the following:

Replace

```
sudo python3 [RobotName]/server/webServer.py
```

with

```
sudo python3 [RobotName]/server/server.py
```

●Save and exit so that the robot will automatically run server.py instead of webServer.py the next time the robot is turned on.

●server.py is a socket server used when using pythonGUI. We do not recommend it to novices here, because you need to manually install a lot of dependent libraries in the computer that controls it to allow the GUI to communicate with it normally. It is recommended to use the WEB application to control the Raspberry Pi robot.

❖ 8 Remote Operation of Raspberry Pi Via MobaXterm

●To make daily use of the Raspberry Pi more convenient, we usually do not connect peripherals such as mouse, keyboard, and monitor to the Raspberry Pi. Since our Raspberry Pi is installed inside the robot, often with peripherals to control the Raspberry Pi, the efficiency of programming and testing will be seriously affected. Therefore, we introduce a method of programming in the Raspberry Pi.

●There are many ways to program in the Raspberry Pi. For example, you can use **3.x to log in to the Raspberry Pi** without using a third-party tool. You can also create files in the Raspberry Pi. Almost all operations can use SSH to connect to the Raspberry Pi in the terminal, but for many people, it will be a disappointing experience when a lot of codes are written in the terminal. This chapter introduces a method that can facilitate the transfer of files to the Raspberry Pi. This method can directly edit programs in the Raspberry Pi.

●This method requires the third-party software MobaXterm,Website address

●MobaXterm is a terminal tool software that can be used to remotely control the Raspberry Pi and remote control is available when SSH is on. For Raspberry Pi's method of enabling SSH and automatically connecting to WIFI, please refer to steps **2.2** and **2.3**.

●Download and install MobaXterm.

●To obtain the IP address of the Raspberry Pi, you can refer to the method of **3.x and log into the Raspberry Pi** in this document to obtain the IP address of the Raspberry Pi.

●To run MobaXterm, firstly, create a new session, click Session in the upper left corner, click SSH in the pop-up window, fill in the IP address of the Raspberry Pi behind Remote host, and finally click OK, the default account name of the Raspberry Pi is pi , The default password is raspberry. Just the password doesn't appear on the screen when you enter it and the * number doesn't mean nothing **Enter** successfully, press after login to log in to the Raspberry Pi, MobaXterm will remind you to save the password.You need to choose.

●If the user name and password are correct, you can change the user name and password according to the prompt in the terminal, which is more secure.

●After the success of the login, MobaXterm will automatically save the conversation, when connected to the raspberry pie again next time only need to double click on the left side of the IP address can be connected to the Raspberry Pi again, if there is no save username and password will need to input the user name and password, if the IP address of the Raspberry Pi changed, you need to start a new dialogue.

●After a successful login, the left column is replaced with a file transfer system, which allows you to interact with the system inside the Raspberry Pi. If you want to return to session selection, just click Sessions.

●Programs you write on other devices can be transferred to the Raspberry Pi by simple drag and drop, and then the Raspberry Pi can be controlled in the terminal to execute the program, or the files in the raspberry Pi can be dragged to other devices.

●If you want to use another IDE to edit files in Raspberry Pi, you can find the file you want to edit in the file transfer system on the left side of the MobaXterm. Right-click on this file and select your IDE so you can use your favorite on other devices IDE to edit the Raspberry Pi file, after editing is completed **CTRL+S** save the file.

●However, it should be noted that when you use MobaXterm's file transfer system to edit files in the Raspberry Pi, you need to pay attention to the permissions problem, because the file transfer system does not have root permissions, so if you are prompted to save after editing the file The permission denied error causes the file cannot be saved after editing. You need to use the following command to give the file you want to edit permission to be edited by MobaXterm:

```
sudo chmod 776 [FileName]
```

●You can learn more about Linux permissions through maketecheasier article from the article link Understanding File Permissions。

❖ 9 How to Control WS2812 RGB LED

●WS2812 LED light is a commonly used module on our robot products. There are three WS2812 lights on each module. Please pay attention when connecting. The signal line is different in direction, which needs to be connected to WS2812 after being led out from the Raspberry Pi. The IN end of the LED lamp module, when the next WS2812 LED module needs to be connected, the signal line is led out from the OUT end of the previous WS2812 module and connected to the IN end of the next WS2812 LED.

●When using the Raspberry Pi with the driver board Motor HAT installed, the WS2812 LED module can be connected to the WS2812 interface on the Motor HAT using a 3pin cable.

●We use a third-party library rpi_ws281x to control the WS2812 LED light, you can learn more about this project on GitHub.

●If you connect the WS2812 LED module to Motor HAT's WS2812 interface, the signal line is equivalent to the Raspberry Pi On GPIO 12, information about the pin number of the Raspberry Pi can refer to this official document GPIO

●Use the following command to install rpi_ws281x for the Raspberry Pi. Since the Raspberry Pi has two versions of Python built in, the Python3 code is used as an example, so pip3 is adopted to install the library.

```
pip3 install rpi-ws281x
```

●Next, we will explain the program. This program is written in the Raspberry Pi and executed in the Raspberry Pi. For the specific method, you can refer to **8 Programming in the Raspberry Pi.**

●Import dependencies

```
import time

from rpi_ws281x import *
```

●Construction of LED control class

```
class LED:
    def __init__(self):
        self.LED_COUNT          = 16 # Set to the total number of LED lights on the robot product.There are more LED lights on the
Raspberry Pi
        self.LED_PIN            = 12 # Set to the input pin number of the LED group
        self.LED_FREQ_HZ        = 800000
        self.LED_DMA            = 10
        self.LED_BRIGHTNESS = 255
        self.LED_INVERT         = False
        self.LED_CHANNEL        = 0

        # Use the configuration item above to create a strip
        self.strip = Adafruit_NeoPixel(
            self.LED_COUNT,
            self.LED_PIN,
            self.LED_FREQ_HZ,
            self.LED_DMA,
            self.LED_INVERT,
            self.LED_BRIGHTNESS,
            self.LED_CHANNEL
            )
        self.strip.begin()

    def colorWipe(self, R, G, B): # This function is used to change the color of the LED light
        color = Color(R, G, B)
        for i in range(self.strip.numPixels()): # Only one LED light color can be set at a time, so we need to do a loop
            self.strip.setPixelColor(i, color)
            self.strip.show() # The color will only change after calling the show method
```

●Instantiate the object and execute the method function. The function colorWipe () needs to pass in three parameters, namely R, G, and B, which correspond to the brightness of the three primary colors of red, green, and blue. The value range is 0- 255, the larger the value, the higher the brightness of the corresponding color channel. If the values of the three color channels are the same, white light is emitted. Specific examples are as follows:

```
if __name__ == '__main__':
    LED = LED()
    try:
        while 1:
            LED.colorWipe(255, 0, 0)   # All the lights turn red
            time.sleep(1)
            LED.colorWipe(0, 255, 0)   # All lights turn green
            time.sleep(1)
            LED.colorWipe(0, 0, 255)   # All lights turn blue
            time.sleep(1)
    except:
        LED.colorWipe(Color(0,0,0))   # Turn off all lights
```

●The above code will control all WS2812 lights to cycle through the three colors, press **CTRL+C** to exit the program.

●If you want to control the color of a single lamp, you can use the following code, where i is the serial number of the lamp, the serial number of the first lamp connected from the driver board is 0, and the second lamp is 1. By analogy, R, G, B are the brightness of the corresponding three color channels:

```
LED.strip.setPixelColor(i, Color(R, G, B))
LED.strip.show()
```

●Note: You must use the Color () method to pack the RGB values and then pass them to setPixelColor ()

❖ 10 How to Control The Servo

10.1 Control The Steering Gear to Rotate to A Certain Angle

●Since the servo can use the PWM signal to control the rotation angle of a mechanism, it is a more commonly used module on robot products. Walking robots, robotic arms and gimbals are all driven by the servo. In our Raspberry Pi The driver board Motor HAT has a dedicated PCA9685 chip for controlling the servo. The Raspberry Pi uses I2C to communicate with the PCA9685. You only need to install the Raspberry Pi driver board Motor HAT on the Raspberry Pi, and the Raspberry Pi will be connected to the PCA9685. No other wires are required for connection.

●The Raspberry Pi uses Python code to control the steering gear, and requires third-party libraries Adafruit_PCA9685, Adafruit-PCA9685Project address, if you run the installation script of the robot software, you do not have to manually install it again, if you do not have the security of running the robotTo install the script, use the following command to install Adafruit_PCA9685 for Python3 in the Raspberry Pi:

```
sudo pip3 install adafruit-pca9685
```

●After the installation, you can use the Python3 code in the Raspberry Pi to control the servo:

```
import Adafruit_PCA9685      # Import the library used to communicate with PCA9685
import time

pwm = Adafruit_PCA9685.PCA9685()    # Instantiate the object used to control the PWM
pwm.set_pwm_freq(50) # Set the frequency of the PWM signal

while 1：  # Make the servo connected to the No. 3 servo port on the Motor HAT drive board reciprocate
    pwm.set_pwm(3, 0, 300)
    time.sleep(1)
    pwm.set_pwm(3, 0, 400)
    time.sleep(1)
```

●In the above code, set_pwm_freq (50) is used to set the PWM frequency to 50Hz. This setting depends on the model of the servo. The servo used by our robot product needs to be controlled by a 50Hz PWM signal. If you use other The value of the servo needs to be set by referring to the specific servo documentation.

●pwm.set_pwm (3, 0, 300) This method is used to control the rotation of a servo to a certain position, where 3 is the servo port number, which corresponds to the number identified on the Motor HAT driver board, but pay attention to the rudder When the machine is connected to the drive board, do not insert the reverse direction of the ground wire, VCC and signal wire, brown to black, red to red, yellow to yellow; 0 is the deviation of controlling the rotation of the servo Our program does not use this function to correct the deviation (the reason for the error of the steering gear can refer to **4.2 Structural Assembly Note**); 300 is the PWM duty cycle value you want to set. According to the different servos, this value represents different servo angles.The PWM duty cycle range of the servos we use is approximately 100 to 560, which corresponds to a rotation range of approximately 0 ° to 180 °.

●The above code to control the steering gear does not control the rotation speed of the steering gear. If we want to make a certain steering gear swing back and forth slowly between two positions, we need to use an increasing or decreasing variable method to control the steering gear.

10.2 Control The Slow Motion of The Steering Gear

```
import Adafruit_PCA9685  # Import the library used to communicate with PCA9685
import time

pwm = Adafruit_PCA9685.PCA9685()# Instantiate the object used to control the PWM
pwm.set_pwm_freq(50)     # Set the frequency of the PWM signal

while 1:
        for i in range(0,100):   # Slowly move the servo from 300 to 400
                pwm.set_pwm(3, 0, (300+i))
                time.sleep(0.05)
        for i in range(0,100):   # Slowly move the servo from 400 to 300
                pwm.set_pwm(3, 0, (400-i))
                time.sleep(0.05)
```

●The above code can make the steering gear rotate slowly back and forth between 300 and 400, but this method of controlling the steering gear also has a lot of drawbacks. When the program is executed until the slow movement of the steering gear will block, this will seriously affect the program Performance, so we provide a multi-threaded solution in our robot product program to solve this problem.

10.3 Non-blocking Control

●You can find the RPIservo.py file in the server folder of the robot product, copy it to the same folder as the program you want to run, and then you can use this method in your program.

```
import RPIservo     # Import a library that uses multiple threads to control the steering gear
import time

sc = RPIservo.ServoCtrl()# Instantiate the object that controls the steering gear
sc.start()  # Start this thread, when the servo does not move, the thread is suspended

while 1:
        sc.singleServo(3, -1, 2)
        time.sleep(1)
        sc.stopWiggle()

        sc.singleServo(3, 1, 2)
        time.sleep(1)
        sc.stopWiggle()
```

●The above code can be used to control the servo to reciprocate, except for time.sleep (), it will not block the running of the context program.

●Call singleServo () to start the movement of the servo, call stopWiggle () to stop the movement of the servo, where singleServo ()Three parameters are required, which are the port number of the servo to be controlled (3 is to control the No. 3 servo connected to Motor HAT), the movement direction of the servo (1 or -1), and the movement speed of the servo (the greater the value, the greater The sooner).

❖ 11 How to Control DC Motor

●If the Raspbian image version you installed is Raspbian Full provided by the official website, you do not need to install other dependent libraries. We only need to control the GPIO port of the Raspberry Pi for simple high and low levels and PWM to control the L298N chip on Motor HAT, thus controlling the direction and speed of the motor.

●When you use the Motor HAT driver board to connect the motor, because the current required by the motor is relatively large, try not to use a USB cable to power the Raspberry Pi and the driver board. There is a 2pin Vin interface on the driver board, which can be used

```
import time
import RPi.GPIO as GPIO# Import the library used to control GPIO

GPIO.cleanup()       # Reset the high and low levels of the GPIO port
GPIO.setwarnings(False) # Ignore some insignificant errors
GPIO.setmode(GPIO.BCM)    # There are three encoding methods for the GPIO port of the Raspberry Pi, we choose BCM encoding to define the GPIO port

'''
The following code defines the GPIO used to control the L298N chip. This definition is different for different Raspberry Pi driver boards.
'''
Motor_A_EN      = 4
Motor_B_EN      = 17

Motor_A_Pin1    = 14
Motor_A_Pin2    = 15
Motor_B_Pin1    = 27
Motor_B_Pin2    = 18

def motorStop():     # Stop motor rotation
	GPIO.output(Motor_A_Pin1, GPIO.LOW)
	GPIO.output(Motor_A_Pin2, GPIO.LOW)
	GPIO.output(Motor_B_Pin1, GPIO.LOW)
	GPIO.output(Motor_B_Pin2, GPIO.LOW)
	GPIO.output(Motor_A_EN, GPIO.LOW)
	GPIO.output(Motor_B_EN, GPIO.LOW)
```

```
def setup():     # GPIO initialization, GPIO motor cannot be controlled without initialization
    global pwm_A, pwm_B
    GPIO.setwarnings(False)
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(Motor_A_EN, GPIO.OUT)
    GPIO.setup(Motor_B_EN, GPIO.OUT)
    GPIO.setup(Motor_A_Pin1, GPIO.OUT)
    GPIO.setup(Motor_A_Pin2, GPIO.OUT)
    GPIO.setup(Motor_B_Pin1, GPIO.OUT)
    GPIO.setup(Motor_B_Pin2, GPIO.OUT)

    motorStop()    # Avoid motor starting to rotate automatically after initialization
    try:        # Try is used here to avoid errors due to repeated PWM settings
        pwm_A = GPIO.PWM(Motor_A_EN, 1000)
        pwm_B = GPIO.PWM(Motor_B_EN, 1000)
    except:
        pass

 def motor_A(direction, speed):    # The function used to control the motor of port A
    if direction == 1:
        GPIO.output(Motor_A_Pin1, GPIO.HIGH)
        GPIO.output(Motor_A_Pin2, GPIO.LOW)
        pwm_A.start(100)
        pwm_A.ChangeDutyCycle(speed)
     if direction == -1:
        GPIO.output(Motor_A_Pin1, GPIO.LOW)
        GPIO.output(Motor_A_Pin2, GPIO.HIGH)
        pwm_A.start(100)
        pwm_A.ChangeDutyCycle(speed)

 def motor_B(direction, speed):    # The function used to control the motor of port B
    if direction == 1:
        GPIO.output(Motor_B_Pin1, GPIO.HIGH)
        GPIO.output(Motor_B_Pin2, GPIO.LOW)
        pwm_B.start(100)
        pwm_B.ChangeDutyCycle(speed)
     if direction == -1:
        GPIO.output(Motor_B_Pin1, GPIO.LOW)
        GPIO.output(Motor_B_Pin2, GPIO.HIGH)
        pwm_B.start(100)
        pwm_B.ChangeDutyCycle(speed)

'''
```

```
Control A and B motors to rotate at full speed for 3 seconds
'''
motor_A(1, 100)
motor_B(1, 100)
time.sleep(3)

'''
Control A and B motors to rotate in opposite directions at full speed for 3 seconds
'''
motor_A(-1, 100)
motor_B(-1, 100)
time.sleep(3)

'''
Stop the motor rotation of A and B ports
'''
motorStop()
```

●The above codes can be used to control the motor movement. The structure of the two functions motor_A and motor_B is the same, but the control servo port is different. This function requires two parameters, one is the direction and the other is the speed. 1 or -1. The minimum speed is 0 and the maximum value is 100. Since the speed adjustment is adjusted by PWM, it is actually equivalent to adjusting the voltage value of the motor port. The motor has a deceleration mechanism as a load. There will be no rotation, so the speed value should not be too low.

●According to the actual situation, changing the speed value of the robot driven by the motor will only make the robots starting speed slower, which has little effect on the maximum speed, and when the given speed is too low, it will cause the motor to stall.

❖ 12 Ultrasonic Module

● The camera used by our Raspberry Pi robot is monocular, which cannot collect depth information. Therefore, many of our robot products use ultrasonic ranging modules to obtain depth information and detect whether there is an obstacle in a certain direction to obtain the distance of the obstacle.

●The principle of ultrasonic ranging is shown in the following formula:

$$\mathbf{S} = (\mathbf{T}_2 - \mathbf{T}_1) \times \mathbf{V}_s / 2$$

● S is the distance of the obstacle, T2 is the time when the echo is received, T1 is the time when the sound wave is emitted, and Vs is the speed of sound propagation in the air. We can use the above formula to find the distance of the obstacle.

● The model of the ultrasonic ranging module used in our robot products is HC-SR04. The HC-SR04 module has four pins, namely VCC, GND, Echo and Trig. Distance sensing function 2cm-400cm, ranging accuracy can reach 3mm; module includes ultrasonic transmitter, receiver and control circuit. The basic working principle is as follows:

・Trigger distance measurement using TRIG of IO port, automatically send 8 40khz square waves to a high-level signal

・module of at least 10us, and automatically detect whether a signal returns

・When a signal returns, a high level is output through the IO port ECHO. The duration of the high level is the time from the transmission of the ultrasonic wave to the return.

●When the Motor HAT driver board is used, you need to connect the HC-SR04 to the Ultrasonic interface on the driver board. You must not connect it to the IIC port to avoid burning the ultrasonic module. (IIC is an interface used to connect I2C devices, and its VCC and GND pin positions are different from Ultrasonic)

●The method of using Python3 to obtain ultrasonic ranging results is as follows:

```
import RPi.GPIO as GPIO
import time

Tr = 11 # Pin number of input terminal of ultrasonic module
Ec = 8 # Pin number of output terminal of ultrasonic module
```

```
GPIO.setmode(GPIO.BCM)
GPIO.setup(Tr, GPIO.OUT,initial=GPIO.LOW)
GPIO.setup(Ec, GPIO.IN)

def checkdist():
        GPIO.output(Tr, GPIO.HIGH) # Set the input end of the module to high level and emit an initial sound wave
        time.sleep(0.000015)
        GPIO.output(Tr, GPIO.LOW)

        while not GPIO.input(Ec): # When the module no longer receives the initial sound wave
                pass
        t1 = time.time() # Note the time when the initial sound wave is emitted
        while GPIO.input(Ec): # When the module receives the return sound wave
                pass
        t2 = time.time() # Note the time when the return sound wave is captured

        return round((t2-t1)*340/2,2) # Calculate distance

'''
Output ultrasonic ranging results once per second and output ten times
'''
for i in range(10):
        print(checkdist())
        time.sleep(1)
```

●As regard the ultrasonic module, this is a commonly used module in many preliminary projects in order to reduce the complexity of the program, although there is a blocking part in this code, we did not use multi-threading to solve it If your project has a high demand for product performance, you can refer to the content of **14** to reconstruct multi-threaded ultrasound.

●When your project needs to use the ultrasonic function, you don't need to rewrite the above code, just copy ultra.py in the robot program server folder to the same folder as your own project Use the following code to obtain ultrasonic ranging information:

```
import ultra

distance = ultra.checkdist()
```

❖ 13 Line Tracking

● Some of our robot products are equipped with a three-channel infrared line patrol module. The line patrol module is converted to the robot's line patrol function design. The three-channel infrared line patrol module contains 3 groups of sensors, where each group of sensors consists of an infrared emitting LED and an infrared sensor photoelectric Transistor composition, the robot determines whether there is a line detected by detecting the infrared light intensity detected by the infrared sensor phototransistor. It can detect the white line (reflected infrared light) on a black background (non-reflected infrared light), and can also detect a white background The black line on (reflects infrared light) (does not reflect infrared light).

● Since the Raspberry Pi can only read digital signals, the three-channel infrared tracking module is equipped with a potentiometer. You can use a cross screwdriver to adjust the potentiometer on the infrared tracking module to adjust the sensitivity of the infrared sensor phototransistor.

● Our program defaults to finding black lines on a white background (reflecting infrared light) (not reflecting infrared light).

● Before using the three-channel infrared line patrol module, you need to connect it to the Tracking interface on Motor HAT using a 5-pin cable.

●The three-way infrared line patrol module has an arrow pattern on the back of the sensor. The direction of the arrow is the direction of the robot.

●The codes of the three-channel infrared line patrol module are as follows:

```
import RPi.GPIO as GPIO
import time

# Hunt module output pin
line_pin_right = 19
line_pin_middle = 16
line_pin_left = 20
'''
Initialize your GPIO port related to the line patrol module
'''
def setup():
        GPIO.setwarnings(False)
        GPIO.setmode(GPIO.BCM)
        GPIO.setup(line_pin_right,GPIO.IN)
```

```
    GPIO.setup(line_pin_middle,GPIO.IN)
    GPIO.setup(line_pin_left,GPIO.IN)

def run():
    '''
    Read the values of three infrared sensor phototransistors (0 is no line detected, 1 is line detected)
    This routine takes the black line on white as an example
    '''
    status_right = GPIO.input(line_pin_right)
    status_middle = GPIO.input(line_pin_middle)
    status_left = GPIO.input(line_pin_left)

    # Detect if the line hunting module senses the line
    if status_middle == 1:
        '''
        Control the robot forward
        '''
        print('forward')

    elif status_left == 1:
        '''
        Control the robot to turn left
        '''
        print('left')

    elif status_right == 1:
        '''
        Control the robot to turn right
        '''
        print('right')

    else:
        '''
        If the line is not detected, the robot stops, you can also make the robot go backwards
        '''
        print('stop')

if __name__ == '__main__':
    setup()
    while 1:
        run()
```

●When your project needs to use the line patrol function, you don't need to rewrite the above code, just copy findline.py and move.py in the robot program server folder to the same as your own project In the folder, then use the following code to use the line patrol function:

```
import findline

findline.setup()

while 1:
        findline.run()
```

● The reason why you need to import move.py is findline.py needs to use the method in move.py to control the robot movement. If you use other methods, then you only need to rewrite the relevant code in findline.py.

❖ 14 Make A Police Light or Breathing Light

14.1 Multi-threading Introduction

●This chapter introduces the use of multi-threading to achieve some effects related to WS2812 LED lights. Multi-threading is a commonly used operation in robot projects. Because robots have high requirements for real-time response, when performing a certain task, try not to block main thread communication.

● Multi-threading is similar to executing multiple different programs or tasks simultaneously. Multi-threading has the following advantages:

・Using threads can put long-running tasks in the background for processing.

・To improve the operating efficiency of the program, the subsequent real-time video and OpenCV processing of video frames use multi-threading to greatly increase the frame rate.

・The encapsulated multi-threaded task is more convenient to call, similar to the **10.3 non-blocking control method** is to use multi-threaded encapsulated control method

●We use Python's threading library to provide thread-related work. Threads are the smallest unit of work in an application. The current version of Python does not provide multi-thread priority, thread group, and thread cannot be stopped, suspended, resumed and interrupted.

14.2 Realization of Police Lights / Breathing Lights

●We use the following code to achieve multi-threaded control of LED lights, and when the LED does not change, keep the thread blocked, so as not to waste CPU resources.

●The wait () method is used here to block the thread, from the realization of the need to control the thread.

```
import time
import sys
from rpi_ws281x import *
import threading
```

```
'''
Use the Threading module to create threads, inherit directly from threading.Thread, and
then override the __init__ method and the run method
'''
class RobotLight(threading.Thread):
    def __init__(self, *args, **kwargs):
        '''
        Here initialize some settings about LED lights
        '''
        self.LED_COUNT        = 16      # Number of LED pixels.
        self.LED_PIN        = 12      # GPIO pin connected to the pixels (18 uses PWM!).
        self.LED_FREQ_HZ  = 800000  # LED signal frequency in hertz (usually 800khz)
        self.LED_DMA        = 10      # DMA channel to use for generating signal (try 10)
        self.LED_BRIGHTNESS = 255     # Set to 0 for darkest and 255 for brightest
        self.LED_INVERT     = False   # True to invert the signal (when using NPN transistor
level shift)
        self.LED_CHANNEL  = 0    # set to '1' for GPIOs 13, 19, 41, 45 or 53

        '''
        Set the brightness of the three RGB color channels, no need to change here, these
values will be automatically set after the subsequent call of the breathing light function
        '''
        self.colorBreathR = 0
        self.colorBreathG = 0
        self.colorBreathB = 0
        self.breathSteps = 10

        '''
        The mode variable, 'none' will make the thread block and hang, the light will not
change;
          'police' is a police light mode, red and blue flash alternately;
          'breath' breathing light, you can set the specified color.
        '''
        self.lightMode = 'none'      #'none' 'police' 'breath'

        # Create NeoPixel object with appropriate configuration.
        self.strip = Adafruit_NeoPixel(self.LED_COUNT, self.LED_PIN, self.LED_FREQ_HZ,

                                  self.LED_DMA, self.LED_INVERT, self.LED_BRIGHTNESS,
                                  self.LED_CHANNEL)
        # Intialize the library (must be called once before other functions).
        self.strip.begin()
        super(RobotLight, self).__init__(*args, **kwargs)
        self.__flag = threading.Event()
        self.__flag.clear()

    # Define functions which animate LEDs in various ways.
    def setColor(self, R, G, B):
        '''
        Set the color of all lights
        '''
        color = Color(int(R),int(G),int(B))
        for i in range(self.strip.numPixels()):
            self.strip.setPixelColor(i, color)
            self.strip.show()


    def setSomeColor(self, R, G, B, ID):
        '''
        Set the color of some lamps, the ID is the array of the serial number of this lamp
        '''
        color = Color(int(R),int(G),int(B))
        #print(int(R),'  ',int(G),'  ',int(B))
        for i in ID:
            self.strip.setPixelColor(i, color)
            self.strip.show()


    def pause(self):
        '''
        Call this function, set __flag to False, block the thread
        '''
        self.lightMode = 'none'
        self.setColor(0,0,0)
        self.__flag.clear()


    def resume(self):
        '''
        Call this function, set __flag to True to start the thread
        '''
        self.__flag.set()


    def police(self):
        '''
        Call this function to turn on the police light mode
        '''
        self.lightMode = 'police'
        self.resume()


    def policeProcessing(self):
        '''
        The specific realization of the police light mode
        '''
        while self.lightMode == 'police':
            '''
            Blue flashes 3 times
            '''
            for i in range(0,3):
                self.setSomeColor(0,0,255,[0,1,2,3,4,5,6,7,8,9,10,11])
                time.sleep(0.05)
                self.setSomeColor(0,0,0,[0,1,2,3,4,5,6,7,8,9,10,11])
                time.sleep(0.05)
            if self.lightMode != 'police':
                break
            time.sleep(0.1)
            '''
            Red flashes 3 times
            '''
            for i in range(0,3):
                self.setSomeColor(255,0,0,[0,1,2,3,4,5,6,7,8,9,10,11])
                time.sleep(0.05)
                self.setSomeColor(0,0,0,[0,1,2,3,4,5,6,7,8,9,10,11])
                time.sleep(0.05)
            time.sleep(0.1)


    def breath(self, R_input, G_input, B_input):
        '''
        Call this function to turn on the breathing light mode, you need to enter three
parameters, namely the brightness of the RGB three color channels, as the color when the
brightness of the breathing lamp is maximum
        '''
        self.lightMode = 'breath'
        self.colorBreathR = R_input
        self.colorBreathG = G_input
        self.colorBreathB = B_input
        self.resume()


    def breathProcessing(self):
        '''
        Specific realization method of breathing lamp
        '''
        while self.lightMode == 'breath':
            '''
            All lights gradually brighten
            '''
            for i in range(0,self.breathSteps):
                if self.lightMode != 'breath':
                    break
                self.setColor(self.colorBreathR*i/self.breathSteps,
                              self.colorBreathG*i/self.breathSteps,
                              self.colorBreathB*i/self.breathSteps)
                time.sleep(0.03)
            '''
            All lights are getting darker
            '''
            for i in range(0,self.breathSteps):
                if self.lightMode != 'breath':
                    break
                self.setColor(self.colorBreathR-(self.colorBreathR*i/self.breathSteps),
                              self.colorBreathG-(self.colorBreathG*i/self.breathSteps),
                              self.colorBreathB-(self.colorBreathB*i/self.breathSteps))
                time.sleep(0.03)


    def lightChange(self):
        '''
        This function is used to select the task to perform
        '''
        if self.lightMode == 'none':
            self.pause()
        elif self.lightMode == 'police':
            self.policeProcessing()
        elif self.lightMode == 'breath':
            self.breathProcessing()
    def run(self):
        '''
        Functions for multi-threaded tasks
        '''
        while 1:
            self.__flag.wait()
            self.lightChange()
            pass


if __name__ == '__main__':
    RL=RobotLight()   # Instantiate the object that controls the LED light
    RL.start()     # Start thread

    '''
    Start breathing light mode and stop after 15 seconds
    '''
    RL.breath(70,70,255)
    time.sleep(15)
    RL.pause()

    '''
    Pause for 2 seconds
    '''
    time.sleep(2)

    '''
    Start the police light mode and stop after 15 seconds
    '''
    RL.police()
    time.sleep(15)

    RL.pause()
```

14.3 Warning Lights or Breathing Lights in Other Projects

● When your project needs to use LED lights for warning lights or breathing lights, you don't need to rewrite the above code, just copy robotLight.py in the robot program server folder to the same In the folder, then use the following code to use the warning light or breathing light:

```
import robotLight
```

```
RL=robotLight.RobotLight()    # Instantiate the object that controls the LED light
RL.start()        # Start thread

'''
Start breathing light mode and stop after 15 seconds
'''
RL.breath(70,70,255)
time.sleep(15)
RL.pause()

'''
Pause for 2 seconds
'''
time.sleep(2)

'''
Start the police light mode and stop after 15 seconds
'''
RL.police()
time.sleep(15)
RL.pause()
```

❖ 15 Real-Time Video Transmission

● Real-time video and OpenCV function are the advantages of the Raspberry Pi robot. This chapter introduces the method of real-time video. In fact, there are many ways to transfer the images collected by the Raspberry Pi camera to other devices through the network The robot uses the open source project [flask-video-streaming] from Github the MIT open source agreement, you can click the link to view the source code of the project.

●The reason for the selection is flask-video-streaming. This solution is the most convenient and the most efficient of the many solutions we have tried. The part related to OpenCV also has a good interface to rewrite it as multi-threaded processing.

● Since this project requires the use of Flask and related dependent libraries, our robot software installation script contains the content of installing these dependent libraries. If your Raspberry Pi has not run the robot software installation script, you need to use the following command to install .

```
sudo pip3 install flask
sudo pip3 install flask_cors
```

● This chapter does not introduce the OpenCV part first, only introduces how to see the real-time picture of the Raspberry Pi camera on other devices.

● First download flask-video-streaming this project in the Raspberry Pi. You can download it from Clone on GitHub or download it on your computer and then pass it to the Raspberry Pi. The download command using the Raspberry Pi console is as follows:

```
sudo git clone https://github.com/miguelgrinberg/flask-video-streaming.git
```

● After downloading or transmitting flask-video-streaming in the Raspberry Pi, run the app.py in flask-video-streaming:

```
cd flask-video-streaming
sudo python3 app.py
```

● Not to use sudo python3 flask-video-streaming / app.py to run, there will be an error that * .jpeg is not found.

● Open the browser on the device on the same local area network as the Raspberry Pi (we use Google Chrome to test), and enter the IP address of the Raspberry Pi plus the video streaming port number: 5000 in the address bar, as shown in the following example:

```
192.168.3.157:5000
```

● Now you can see the page created by the Raspberry Pi on the browser of your computer or mobile phone. Note that the default screen is not from the screen of the Raspberry Pi camera, but three digital pictures cyclically playing 1, 2, 3

● If your page can log in and is playing a picture of 1 \ 2 \ 3 numbers in a loop, it means that the flask-related programs are running normally. Next, you can make some modifications to app.py so that it can display the Raspberry Pi on the page in real time. Camera screen.

```
sudo nano app.py
```

● Here we use nano that comes with Raspbian to open app.py for editing in the console. Since it is just some operations for commenting and deleting comments, there is no need to use other IDEs for editing.

● After opening the IDE, we comment out the code:

```
if os.environ.get('CAMERA'):
    Camera = import_module('camera_' + os.environ['CAMERA']).Camera
else:
    from camera import Camera
```

● You can comment out these lines of code by filling in # at the beginning of the code line, or you can write a ''' at the beginning and end of the entire code to comment out a certain code. The relevant code after the change is as follows:

```
# if os.environ.get('CAMERA'):
#       Camera = import_module('camera_' + os.environ['CAMERA']).Camera
# else:
#       from camera import Camera
```

or

```
'''
if os.environ.get('CAMERA'):
      Camera = import_module('camera_' + os.environ['CAMERA']).Camera
else:
      from camera import Camera
```

```
'''
```

●Finally, uncomment the code that imports Camera from camera_pi,

```
    # from camera_pi import Camera
```

delete in front of #, note that there is a space after the # here, and also delete, the changed code is as follows:

```
    from camera_pi import Camera
```

●The following is the complete code of the modified app.py:

```
#!/usr/bin/env python
from importlib import import_module
import os
from flask import Flask, render_template, Response

# import camera driver
'''
if os.environ.get('CAMERA'):
    Camera = import_module('camera_' + os.environ['CAMERA']).Camera
else:
    from camera import Camera
'''

# Raspberry Pi camera module (requires picamera package)
from camera_pi import Camera

app = Flask(__name__)

@app.route('/')
def index():
    """Video streaming home page."""
    return render_template('index.html')

def gen(camera):
    """Video streaming generator function."""
    while True:
        frame = camera.get_frame()
        yield (b'--frame\r\n'
               b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')


@app.route('/video_feed')
def video_feed():
    """Video streaming route. Put this in the src attribute of an img tag."""
    return Response(gen(Camera()),
                              mimetype='multipart/x-mixed-replace; boundary=frame')


if __name__ == '__main__':
    app.run(host='0.0.0.0', threaded=True)
```

●After editing, press **CTRL+X** to launch the editing, and prompt whether to save the changes, press **Y** and **Entry** after saving the changes.

●Then you can run app.py:

```
sudo app.py
```

●Open the browser on the device on the same local area network as the Raspberry Pi (we use Google Chrome to test), and enter the IP address of the Raspberry Pi plus the video streaming port number: 5000 in the address bar, as shown in the following example:

```
192.168.3.157:5000
```

●Now you can see the page created by the Raspberry Pi on the browser of your computer or mobile phone. After loading successfully, the page will display the real-time image of the Raspberry Pi camera.

●This feature uses projects from GitHub flask-video-streaming.

❖ 16 Automatic Obstacle Avoidance

●The ultrasonic module of this product can only move up and down with the camera, and the left and right movement can only rotate with the body of the body, and can not move left and right relative to the body, so the obstacle avoidance function of this robot is relatively simple, as long as there is an obstacle in front Turn left and retreat if the obstacle is too close, and move forward if the obstacle is far away or there is no obstacle.

●The automatic obstacle avoidance function is based on the ultrasonic ranging module, so before writing your project, you must first put **ultra.py RPIservo.py** and copy **move.py** to the same file directory as your project, and then use the following code to use the automatic obstacle avoidance function.

```
import ultra
import move
import time
import Adafruit_PCA9685
import RPIservo

# Get the initial data of pwm
def num_import_int(initial):
        global r
        with open(thisPath+"/RPIservo.py") as f:
                for line in f.readlines():
                        if(line.find(initial) == 0):
                                r=line
        begin=len(list(initial))
        snum=r[begin:]
        n=int(snum)
        return n

scGear = RPIservo.ServoCtrl() # servo control
scanNum = 3                    # The car will scan the three parts in front of it, namely left, middle and right, that is, 1 represents left, 2 represents middle, and 3 represents right
scanPos = 1                    # The next position to be scanned by the car, the value can be 1, 2, and 3 corresponding to left, middle and right
scanDir = 1                    # The direction of the car scanning, 1 is the leftmost, -1 is the rightmost
scanList = []                  # Record the distance of obstacles in three directions that are scanned
scanServo = 1                  # Pan-tit servo number
scanRange = 100                # The range of the angle scanned by servo
rangeKeep = 0.7                # Threshold, if it is less than it, turns; if it is greater than it, go back
```

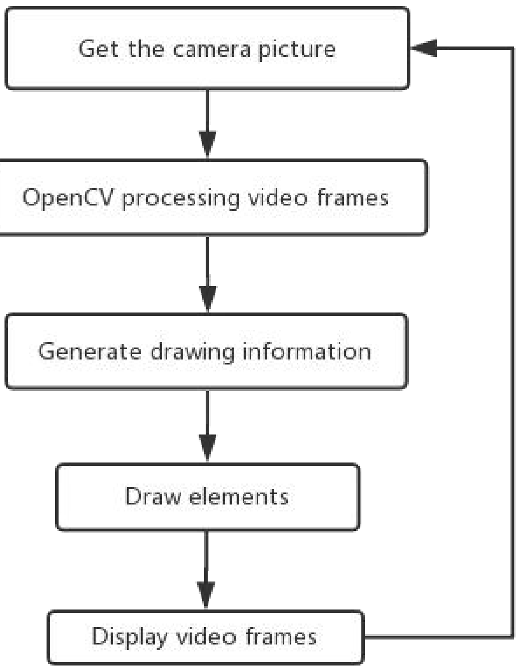
```
# Initialize servo angle
pwm = Adafruit_PCA9685.PCA9685()
pwm.set_pwm_freq(50)

pwm0_direction = 1
pwm0_init = num_import_int('init_pwm0 = ')
pwm0_max   = 520
pwm0_min   = 100
pwm0_pos   = pwm0_init

pwm1_direction = 1
pwm1_init = num_import_int('init_pwm1 = ')
pwm1_max   = 520
pwm1_min   = 100
pwm1_pos   = pwm1_init

pwm2_direction = 1
pwm2_init = num_import_int('init_pwm2 = ')
pwm2_max   = 520
pwm2_min   = 100
pwm2_pos   = pwm2_init

while True:
    # The pan-tit servo rotates to the left, middle and right directions and uses ultrasonic to scan and record the distance
of obstacles
if scanPos == 1:
        pwm.set_pwm(scanServo, 0, pwm1_init + scanRange)
        time.sleep(0.3)
        scanList[0] = ultra.checkdist()
    elif scanPos == 2:
        pwm.set_pwm(scanServo, 0, pwm1_init)
        time.sleep(0.3)
        scanList[1] = ultra.checkdist()
    elif scanPos == 3:
        pwm.set_pwm(scanServo, 0, pwm1_init - scanRange)
        time.sleep(0.3)
        scanList[2] = ultra.checkdist()

    scanPos += scanDir

    # If the scanning angle is not within the set range
    if scanPos > scanNum or scanPos < 1:
        # Change the scanned direction
        if scanDir == 1: scanDir = -1
        elif scanDir == -1: scanDir = 1
        # Restore scanned location
        scanPos += scanDir*2
    print(scanList)

    # If the distance of the nearest obstacle in front is less than the threshold
    if min(scanList) < rangeKeep:
        # If the closest obstacle is on the left
        if scanList.index(min(scanList)) == 0:
            # Then, turn right
            scGear.moveAngle(2, -30)
        # If the closest obstacle is right ahead
        elif scanList.index(min(scanList)) == 1:
            # See the value on the left and right to determine which is larger
            if scanList[0] < scanList[2]:
                # Full rudder to the right and move
                scGear.moveAngle(2, -45)
            else:
                # Otherwise, full rudder to the left and move
                scGear.moveAngle(2, 45)
        # If the closest obstacle is on the right
        elif scanList.index(min(scanList)) == 2:
            # Then, turn left
            scGear.moveAngle(2, 30)
        if max(scanList) < rangeKeep or min(scanList) < rangeKeep/3:
            move.motor_lef(1, 1, 80)
            move.motor_right(1, 1, 80)
    # If the distance of the nearest obstacle in front is greater than the threshold, the car moves backward
    else:
        move.motor_left(1, 0, 80)
        move.motor_right(1, 0, 80)
```

❖ 17 Why OpenCV Uses Multi-threading to Process Video Frames

●The OpenCV function is based on the GitHub project flask-video-streaming, we changed the camera_opencv.py to perform OpenCV related operations.

17.1 Single Thread Processing of Video Frames

●First, we introduce the process of single-thread processing of video frames. Let 's start with a simple one, so that you will understand why OpenCV uses multiple threads to process video frames. The process of single-thread processing of video frames is as follows:

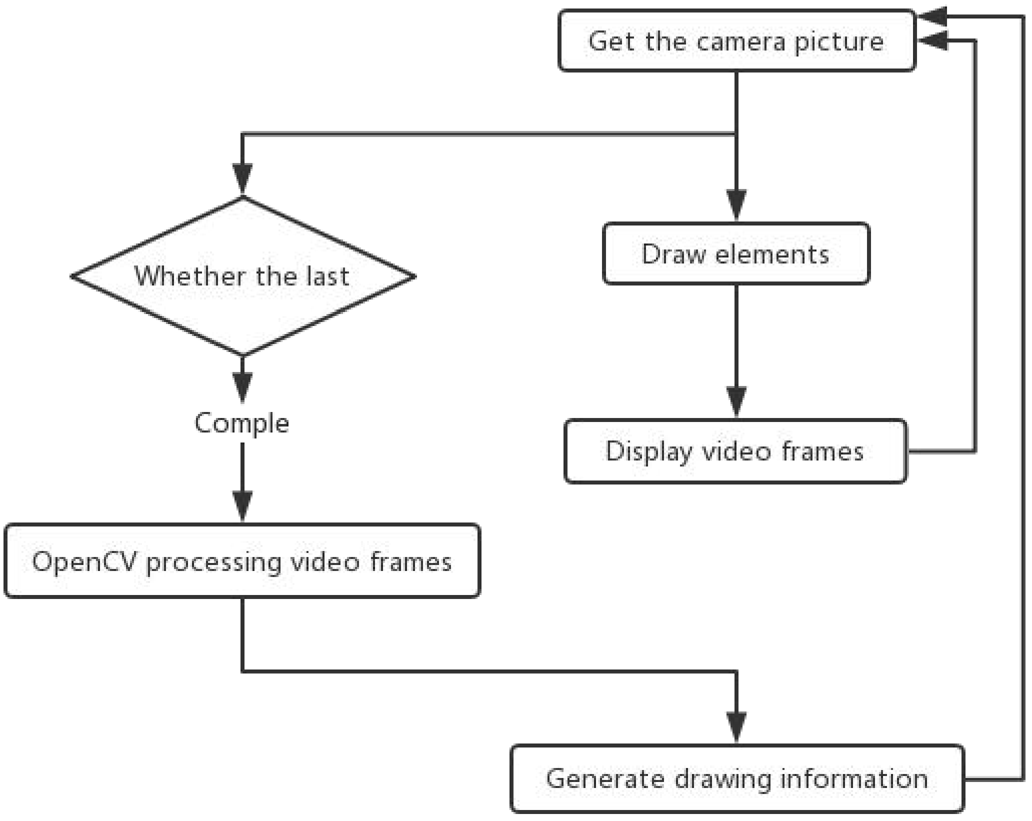


● Process explanation: First obtain a frame from the camera, and then use OpenCV to analyze the content of this frame. After the analysis is completed, the information to be drawn is generated, such as the position of the center point of the target object, the text lamp information that needs to be generated on the screen, then Draw those elements on the screen according to the generated drawing information, and finally display the processed and drawn frame on the page.

●Such a processing flow will cause each frame to be collected to wait for the OpenCV related process to be processed. After this frame is displayed, the second frame can be collected for processing and analysis to make it abnormally stuck.

17.2 Multi-thread Processing of Video Frames

●Next, the process of multi-thread processing of video frames is introduced:



●Process explanation: In order to improve the frame rate, we separate the analysis task of the video frame from the process of acquisition and display, and place it in a background thread to execute and generate drawing information.

●We change the complete code of camera_opencv.py with too many threads as follows: (This code is only for reference of multi-threading principle, and the OpenCV function is intuitively deleted for the sake of demonstration)

```
import os
import cv2
from base_camera import BaseCamera
import numpy as np
import datetime
import time
import threading
import imutils

class CVThread(threading.Thread):
    '''
    This class is used to process OpenCV analysis of video frames in the background. For more basic usage principles of
the multi-threaded class, please refer to 14.2
    '''
    def __init__(self, *args, **kwargs):
        self.CVThreading = 0

        super(CVThread, self).__init__(*args, **kwargs)
        self.__flag = threading.Event()
        self.__flag.clear()


    def mode(self, imgInput):
        '''
        This method is used for incoming video frames that need to be processed
        '''
        self.imgCV = imgInput
        self.resume()


    def elementDraw(self,imgInput):
        '''
        Draw elements in the picture
        '''
        return imgInput


    def doOpenCV(self, frame_image):
        '''
       Add the content to be processed by OpenCV here
        '''
        self.pause()


    def pause(self):
        '''
        Block the thread and wait for the next frame
        '''
        self.__flag.clear()
        self.CVThreading = 0

    def resume(self):
        '''
       Resume the thread
        '''
        self.__flag.set()

    def run(self):
        '''
        Process video frames in the background thread
        '''
        while 1:
            self.__flag.wait()
            self.CVThreading = 1
            self.doOpenCV(self.imgCV)


class Camera(BaseCamera):
    video_source = 0

    def __init__(self):
        if os.environ.get('OPENCV_CAMERA_SOURCE'):
            Camera.set_video_source(int(os.environ['OPENCV_CAMERA_SOURCE']))
        super(Camera, self).__init__()

    @staticmethod
    def set_video_source(source):
        Camera.video_source = source

    @staticmethod
    def frames():
        camera = cv2.VideoCapture(Camera.video_source)
        if not camera.isOpened():
            raise RuntimeError('Could not start camera.')
        '''
        Instantiation CVThread()
        '''
        cvt = CVThread()
        cvt.start()

        while True:
            # read current frame
```

```
img = camera.read()

if cvt.CVThreading:
        '''
        If OpenCV is processing video frames, skip
        '''
        pass
else:
        '''
        If OpenCV is not processing video frames, give the video frame processing thread a new video frame
and resume the processing thread
        '''
        cvt.mode(img)
        cvt.resume()
'''
Draw elements on the screen
'''
img = cvt.elementDraw(img)

# encode as a jpeg image and return it
yield cv2.imencode('.jpg', img)[1].tobytes()
```

●The above is the code principle of using multi-threading to process OpenCV. The following introduction of specific functions of OpenCV will skip the explanation of multi-threading and directly introduce OpenCV's method of processing video frames.

❖ 18 OpenCV Learn to Use OpenCV

●The real-time video transmission function comes from the open source project of Github the MIT open source agreement flask-video-streaming.

●First, prepare two .py files in the same folder in the Raspberry Pi. The code is as follows:

• app.py

```
#!/usr/bin/env python3

from importlib import import_module
import os
from flask import Flask, render_template, Response

from camera_opencv import Camera

app = Flask(__name__)

def gen(camera):
        while True:
                frame = camera.get_frame()
                yield (b'--frame\r\n'
                              b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')


@app.route('/')
def video_feed():
        return Response(gen(Camera()),
                                        mimetype='multipart/x-mixed-replace; boundary=frame')


if __name__ == '__main__':
        app.run(host='0.0.0.0', threaded=True)
```

• base_camera.py

```
import time
import threading
try:
        from greenlet import getcurrent as get_ident
except ImportError:
        try:
```

```
        from thread import get_ident
    except ImportError:
        from _thread import get_ident


class CameraEvent(object):
    """An Event-like class that signals all active clients when a new frame is
    available.
    """
    def __init__(self):
        self.events = {}

    def wait(self):
        """Invoked from each client's thread to wait for the next frame."""
        ident = get_ident()
        if ident not in self.events:
            # this is a new client
            # add an entry for it in the self.events dict
            # each entry has two elements, a threading.Event() and a timestamp
            self.events[ident] = [threading.Event(), time.time()]
        return self.events[ident][0].wait()

    def set(self):
        """Invoked by the camera thread when a new frame is available."""
        now = time.time()
        remove = None
        for ident, event in self.events.items():
            if not event[0].isSet():
                # if this client's event is not set, then set it
                # also update the last set timestamp to now
                event[0].set()
                event[1] = now
            else:
                # if the client's event is already set, it means the client
                # did not process a previous frame
                # if the event stays set for more than 5 seconds, then assume
                # the client is gone and remove it
                if now - event[1] > 5:
                    remove = ident
        if remove:
            del self.events[remove]

    def clear(self):
        """Invoked from each client's thread after a frame was processed."""
        self.events[get_ident()][0].clear()


class BaseCamera(object):
    thread = None    # background thread that reads frames from camera
    frame = None    # current frame is stored here by background thread
    last_access = 0    # time of last client access to the camera
    event = CameraEvent()

    def __init__(self):
        """Start the background camera thread if it isn't running yet."""
        if BaseCamera.thread is None:
            BaseCamera.last_access = time.time()

            # start background frame thread
            BaseCamera.thread = threading.Thread(target=self._thread)
            BaseCamera.thread.start()

            # wait until frames are available
            while self.get_frame() is None:
                time.sleep(0)

    def get_frame(self):
        """Return the current camera frame."""
        BaseCamera.last_access = time.time()

        # wait for a signal from the camera thread
        BaseCamera.event.wait()
        BaseCamera.event.clear()

        return BaseCamera.frame

    @staticmethod
    def frames():
        """"Generator that returns frames from the camera."""
        raise RuntimeError('Must be implemented by subclasses.')

    @classmethod
    def _thread(cls):
        """Camera background thread."""
        print('Starting camera thread.')
        frames_iterator = cls.frames()
        for frame in frames_iterator:
            BaseCamera.frame = frame
            BaseCamera.event.set()    # send signal to clients
            time.sleep(0)

            # if there hasn't been any clients asking for frames in
            # the last 10 seconds then stop the thread
            if time.time() - BaseCamera.last_access > 10:
                frames_iterator.close()
                print('Stopping camera thread due to inactivity.')
                break
        BaseCamera.thread = None
```

●When you use the follow-up tutorial to develop a certain OpenCV related function, you only need to put the corresponding camera_opencv.py in the same folder as this app.py and base_camera.py, and then run it in the Raspberry Pi console just app.py.

● Open the browser with a device on the same local area network as the Raspberry Pi, enter the IP address of the Raspberry Pi in the address bar, and access port 5000. The address is shown in the following example:

192.168.3.161:5000

❖ 19 Using OpenCV to Realize Color Recognition and Tracking

19.1 Color Recognition and Color Space

●For the development preparation and operation of OpenCV function, please refer to **18**.

●Create camera_opencv.py in the folder where app.py and base_camera.py in 18. The code related to the OpenCV color tracking function to be introduced in this chapter is written in camera_opencv.py

●For safety reasons, this routine does not control the motor or servo motion, and only outputs OpenCV calculation results.

●We use OpenCV for color recognition using the HSV color space. Before introducing the code, we first need to understand the color space and why we use the HSV color space instead of the more common RGB color space for color recognition.

●**Color space**:

•Color space is how colors are organized. With the help of color space and tests for physical devices, fixed analog and digital representations of colors can be obtained. The color space can be defined by just picking some colors at random, for example, the Pantone system just takes a specific set of colors as samples, and then defines the name and code for each color; it can also be based on a strict mathematical definition, such as Adobe RGB , SRGB.

●**RGB color space**:

• RGB uses an additive color mixing method because it describes the ratio of various "lights" to produce colors. The light is continuously superimposed from dark to produce color. RGB describes the values of red, green and blue light. RGBA is to add alpha channel to RGB to achieve transparency effect.

Common color spaces based on RGB mode are sRGB, Adobe RGB, and Adobe Wide Gamut RGB.

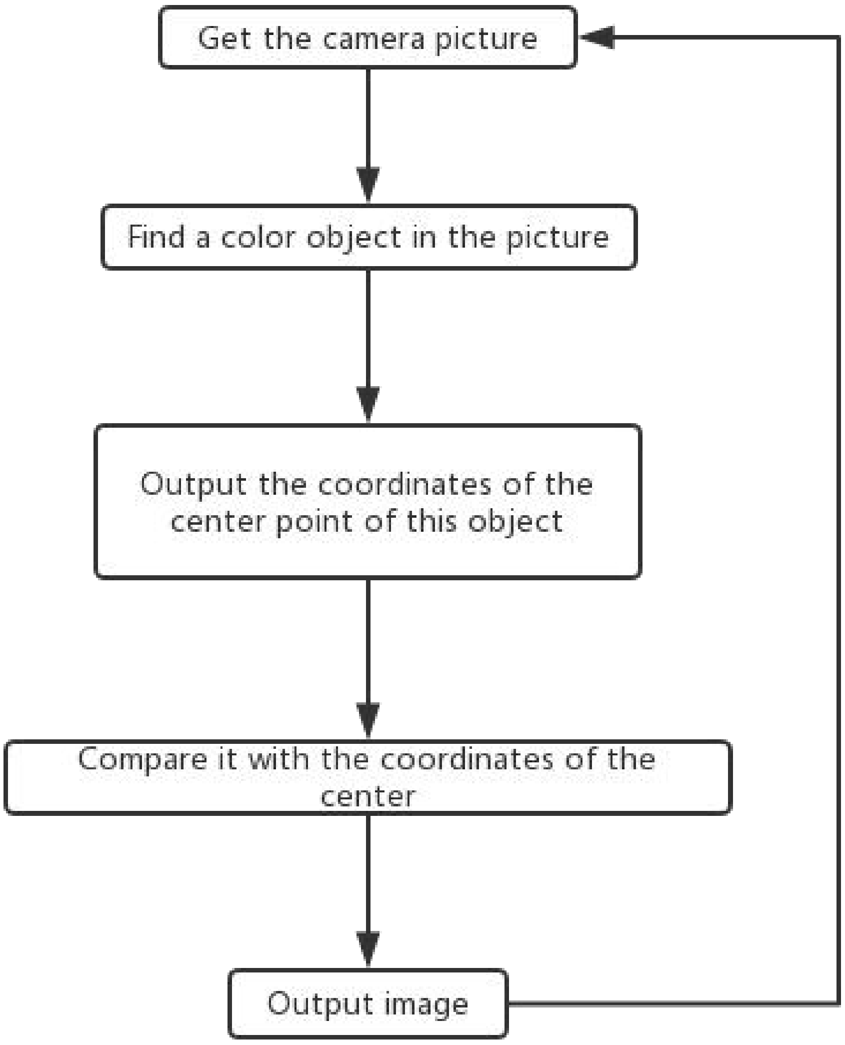
●**HSV color space**:

•HSV (Hue: Hue, Saturation: Saturation, Brightness; Value), also known as HSB (B refers to Brightness) is commonly used by artists, because compared with the term of addition and subtraction, the use of hue, saturation and other concepts to describe color is more natural and intuitive. HSV is a variant of the RGB color space, its content and color scale are closely related to its source-RGB color space

• Using the HSV color space in OpenCV's color recognition function can make the recognition result more accurate, less affected by ambient light, and it is very convenient to define the color range, because what you are looking for during color recognition is not a certain color, but a certain one Range of colors, so use the HSV color space that is more in line with human eye habits for color recognition.

19.2 Process of Color Identifying and Tracking

●We can use this function to control the servo to make the camera aim at a certain color object, the general process is as follows


Get the camera picture
Find a color object in the picture
Output the coordinates of the center point of this object
Compare it with the coordinates of the center
Output image

19.3 Specific Code

●camera_opencv.py

```
import os
import cv2
from base_camera import BaseCamera
import numpy as np

'''
```

```
Set target color, HSV color space
'''
colorUpper = np.array([44, 255, 255])
colorLower = np.array([24, 100, 100])

font = cv2.FONT_HERSHEY_SIMPLEX

class Camera(BaseCamera):
        video_source = 0

        def __init__(self):
                if os.environ.get('OPENCV_CAMERA_SOURCE'):
                        Camera.set_video_source(int(os.environ['OPENCV_CAMERA_SOURCE']))
                super(Camera, self).__init__()

        @staticmethod
        def set_video_source(source):
                Camera.video_source = source

        @staticmethod
        def frames():
                camera = cv2.VideoCapture(Camera.video_source)
                if not camera.isOpened():
                        raise RuntimeError('Could not start camera.')

                while True:
                        # read current frame
                        img = camera.read() #Get the picture captured by the camera

                        hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)      #Convert captured images to HSV color space
                        mask = cv2.inRange(hsv, colorLower, colorUpper) #Traverse the colors in the target color range in the HSV
color space, and turn these color blocks into masks
                        mask = cv2.erode(mask, None, iterations=2) #Corrosion of small pieces of mask (noise) in the picture
becomes small (small pieces of color or noise disappear)
                        mask = cv2.dilate(mask, None, iterations=2) #Inflate, and resize the large mask that was reduced in the
previous step to its original size
                        cnts = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL,
                                cv2.CHAIN_APPROX_SIMPLE)[-2]                  #Find a few masks in the picture
                        center = None
                        if len(cnts) > 0:       #If the number of whole masks in the picture is greater than one
                                '''
                                Find the coordinates of the center point of the object of the target color and the size of the object in
the picture
```

```
        '''
        c = max(cnts, key=cv2.contourArea)
        ((box_x, box_y), radius) = cv2.minEnclosingCircle(c)
        M = cv2.moments(c)
        center = (int(M["m10"] / M["m00"]), int(M["m01"] / M["m00"]))
        X = int(box_x)
        Y = int(box_y)
        '''
        Get the center point coordinates of the target color object and output
        '''
        print('Target color object detected')
        print('X:%d'%X)
        print('Y:%d'%Y)
        print('-------')

        '''
        Write text on the screen：  Target Detected
        '''
        cv2.putText(img,'Target Detected',(40,60), font, 0.5,(255,255,255),1,cv2.LINE_AA)
        '''
        Draw a frame around the target color object
        '''
        cv2.rectangle(img,(int(box_x-radius),int(box_y+radius)),
                      (int(box_x+radius),int(box_y-radius)),(255,255,255),1)
    else:
        cv2.putText(img,'Target Detecting',(40,60), font, 0.5,(255,255,255),1,cv2.LINE_AA)
        print('No target color object detected')

    # encode as a jpeg image and return it
    yield cv2.imencode('.jpg', img)[1].tobytes()
```

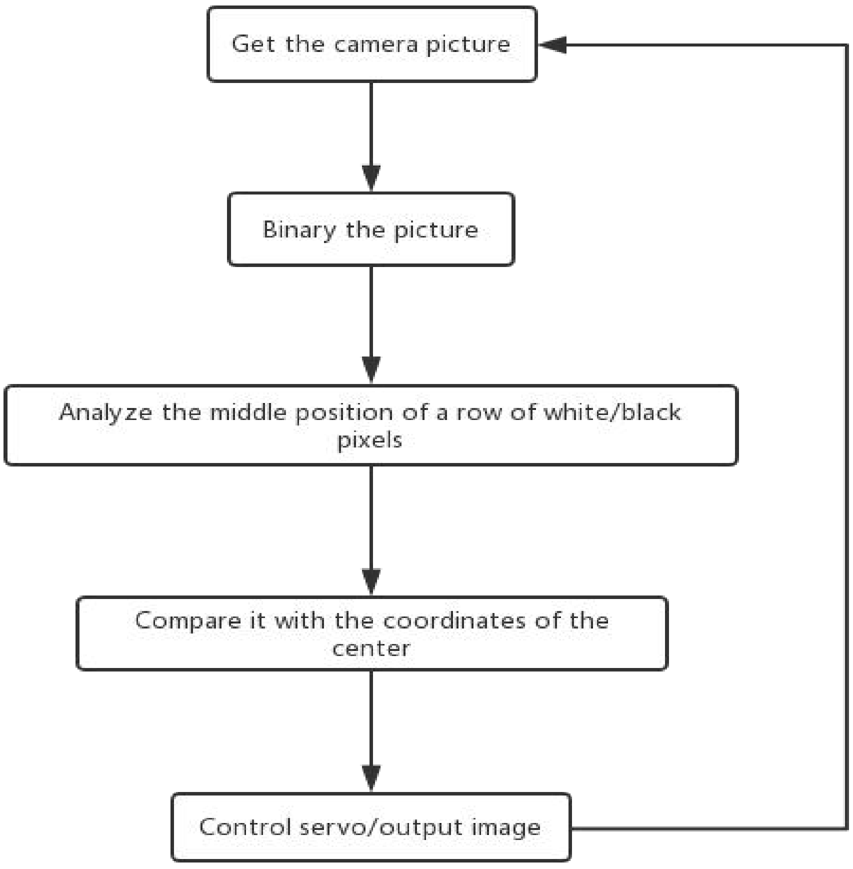
●You can set the color you want to recognize by changing colorUpper and colorLower. The thing to note here is that normal The H value (hue) of the HSV color space is 0-360, but in OpenCV, the H value ranges from 0-180.

19.4 HSV Color Component Range in OpenCV

HSV\color	Black	Grey	White	Red	Orange	Yellow	Green	Cyan	Blue	Purple
H_min	0	0	0	0|156	11	26	35	78	100	125
H_max	180	180	180	10|180	25	34	77	99	124	155
S_min	0	0	0	43	43	43	43	43	43	43
S_max	255	43	30	255	255	255	255	255	255	255
V_min	0	46	221	46	46	46	46	46	46	46
V_max	46	220	255	255	255	255	255	255	255	255

❖ 20 Machine Line Tracking Based on OpenCV

20.1 Visual Line Inspection Process


Get the camera picture
Binary the picture
Analyze the middle position of a row of white/black pixels
Compare it with the coordinates of the center
Control servo/output image

●For the development preparation and operation of OpenCV function, please refer to **18**.

●Create camera_opencv.py in the folder where app.py and base_camera.py in **18**, the code related to the OpenCV visual line tracking function to be introduced in this chapter is written in camera_opencv.py.

●For safety reasons, this routine does not control the motor or servo motion, and only outputs OpenCV calculation results.

20.2 Specific Code

```
import os
import cv2
from base_camera import BaseCamera
import numpy as np
import time
import threading
import imutils

'''
Set the color of the line, 255 is the white line, 0 is the black line
'''
lineColorSet = 255
'''
Set the horizontal position of the reference, the larger the value, the lower, but not greater than the vertical resolution
of the video (default 480)
'''
linePos = 380

class Camera(BaseCamera):
	video_source = 0
	def __init__(self):
		if os.environ.get('OPENCV_CAMERA_SOURCE'):
			Camera.set_video_source(int(os.environ['OPENCV_CAMERA_SOURCE']))
		super(Camera, self).__init__()

	@staticmethod
	def set_video_source(source):
		Camera.video_source = source

	@staticmethod
	def frames():
		camera = cv2.VideoCapture(Camera.video_source)
		if not camera.isOpened():
			raise RuntimeError('Could not start camera.')

		while True:
			img = camera.read() #Get the picture captured by the camera

			'''
```

```
        Convert the picture to black and white, and then binarize (the value of each pixel in the picture is 255 except 0)
        '''
        img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        retval, img =   cv2.threshold(img, 0, 255, cv2.THRESH_OTSU)
        img = cv2.erode(img, None, iterations=6)     #Use Corrosion Denoising
        colorPos = img[linePos]                             #Get an array of pixel values for linePos
        try:
            lineColorCount_Pos = np.sum(colorPos == lineColorSet)     #Get the number of pixels of line color (line width)
            lineIndex_Pos = np.where(colorPos == lineColorSet)        #Get the horizontal position of the end point of the line in the linePos line
            '''
            Use the endpoint position and line width to calculate the position of the center point of the line
            '''
            left_Pos = lineIndex_Pos[0][lineColorCount_Pos-1]
            right_Pos = lineIndex_Pos[0][0]
            center_Pos = int((left_Pos+right_Pos)/2)

            print('The position of the center point of the line is： %d'%center_Pos)
        except:
            '''
            If no line is detected, the line width above 0 will cause an error, so that you know that no line has been detected
            '''
            center_Pos = 0
            print('No line detected')

        '''
        Draw a horizontal reference line
        '''
        cv2.line(img,(0,linePos),(640,linePos),(255,255,64),1)
        if center_Pos:
            '''
            If a line is detected, draw the center point of the line
            '''
            cv2.line(img,(center_Pos,linePos+300),(center_Pos,linePos-300),(255,255,64),1)

        # encode as a jpeg image and return it
        yield cv2.imencode('.jpg', img)[1].tobytes()
```

❖ 21 Create A WiFi Hotspot on The Raspberry Pi

●The method of turning on the WIFI hotspot in our robot product uses a project from GitHub create_ap. Our installation script will automatically install this program and related dependent libraries. If you have not run our installation script, you can use the following command to install create_ap:

```
sudo git clone https://github.com/oblique/create_ap.git cd
create_ap
sudo make install
```

●Install related dependent libraries:

```
sudo apt-get install -y util-linux procps hostapd iproute2 iw haveged dnsmasq
```

●Before turning on the hotspot, your Raspberry Pi cannot be connected to WIFI, and the WIFI module cannot be turned off, so when you test the hotspot function, you need to connect the necessary peripherals for the Raspberry Pi.

●Under normal circumstances, if the robot program is not connected to the WIFI when it is turned on, it will automatically turn on the hotspot. You can use your phone or computer to search for the WIF named Adeept. The default password is 12345678. Once the connection is successful, you can log in to 192.168 .12.1: 5000 using a browser to open the WEB application to control the robot.

●If your Raspberry Pi is connected to peripherals, and you want to test the Raspberry Pi ’ s ability to turn on hotspots, you can click the WIFI icon in the upper right corner of the Raspberry Pi ’ s desktop, click the name of the connected WIFI, click forget, and never turn Off WIFI, if it is already in the off state, you need to turn it on.

●When the WIFI module of the Raspberry Pi is turned on and is not connected to any known network, you can enter the following command on the console to turn on the WIFI:

```
sudo create_ap wlan0 eth0 Adeept 12345678
```

●Adeept is the name of the WIFI hotspot, 12345678 is the password of the WIFI hotspot.

❖ 22 Install GUI Dependent Item under Window

●Our old version of robot programs all provide a desktop GUI program to control the robot. The GUI program is written in Python language, but this method uses a higher threshold and difficulty, and is not recommended for novices.

●This GUI program is only suitable for the Windows system for the time being. It is included in the client directory of the robot software package and is generally called GUI.py.

22.1 Install GUI Dependencies

●Install Python3:

• We need to install Python in the computer to run our PC-side program. The code of this product is developed and tested using Python3, so we need to download Python3.7 or above to prevent possible compatibility errors.

• Python3 download address

• After downloading Python, double-click the installation package to install it.

• When installing Python, be sure to select Add Python to PATH.

●Install Numpy:

• NumPy is a basic software package for scientific computing in Python, and OpenCV needs to use its related functions.

• Press the **win**+**R** key, enter cmd, and click OK to open the CMD.

• Enter the following to install numpy:

```
pip3 install numpy
```

• After input, Entry press to Start downloading and installing numpy.

●Install OpenCV:

• The method of installing numpy is the same.

• After opening CMD, enter the following:

```
pip3 install opencv-contrib-python
```

• After input,**Entry** Start downloading and installing opencv.

●Press to install zmq and pybase64:

• zmq and pybase64 are libraries for real-time video

• After opening CMD, enter the following:

```
pip3 install zmq pybase64
```

• After entering, press **Entry**. Start downloading and installing zmq and pybase64.

❖ 23 How to Use GUI

●Because the web and the GUI are not connected, if you want to use the GUI to control the robot product, you need to manually run server.py. (The method is the same as manually running webserver.py, except that the object is changed to server.py).

●When the server.py in the Raspberry Pi runs successfully, enter the IP address of the Raspberry Pi on the GUI control terminal on the PC, and then click Connect to control the robot.

● Python running window: This window will accompany each GUI when it is opened. Any runtime exceptions will be displayed here. If this window is closed, the GUI will be exited.

● Camera video streaming window: Display the picture captured by the camera. Depending on the product type, the window rendering method may be different, and some products can also interact with this window.

●IP address input box: Enter the IP address of the Raspberry Pi here. Clicking Connect will allow the GUI to connect to the Raspberry Pi. Raspberry Pi status and connection status display bar: Shows some hardware information of the Raspberry Pi and the current connection status.

●Mobile control:

• Forward: Control the robot to move forward; the shortcut key is W.

• Backward: Control the robot to move backwards; the shortcut key is S.

• Left: Control the robot to turn left; the shortcut key is A.

• Right: Control the robot to turn right; the shortcut key is D.

● Robot camera control (due to different layouts, the shortcut keys here are different from WEB applications):

• Up: Control the camera to move upwards; the shortcut key is I.

• Down: Control the camera to move downwards; the shortcut key is K.

• Home: Control all servos to return to the neutral position; the shortcut key is H.

●FindColor:By default, the Robot finds the biggest yellow object in its view and follows it. When it gets close enough, it would stop, and if it gets too close to the yellow object, it would go back.

●WatchDog:If the camera on the robot detects an object moving or changing, the LEDs on the robot will turn red. This feature is developed based on Adrian Rosebrock's OpenCV code on pyimagesearch.com. You can also learn more about the OpenCV to gain more fun to play with, such as syncing the captured image to the dropbox after detecting the motion of the object. The example program we provide just makes the LEDs display red however. For other functions, you can install the corresponding packages according to your needs, just by changing the code in FPV.py

●FindLine:The robot can track lines and follow them, proceeding along a preset path that can be altered by moving the lines, and this part of Python program is easy to understand. You can open findline.py and learn to write it yourself.

●Add More Functions:Function 5 and Function 6 buttons are placeholders for other functions you want to add. This robot is based on raspberry pi so there are a lot more functions you can play with, but some other libraries are required.

We intend to simplify the installation steps as much as possible to lower the barriers for more people. Hence, for example, voice recognition, which requires a large number of libraries t o be installed, will not be provided in the standard program. If you are interested in this, you can try to expand more. We will offer the installation and application methods of other functions in the follow-up tutorials. Please subscribe our Youtube channel for more.

●Change LED Color:You can control the colors of the LEDs on the robot in real time by dragging these three sliders. These three sliders correspond to the brightness of the three channels of RGB. In theory, you can create 16,777,216 (256^ 3) kinds of colors through these three sliders.

❖ 24 Control The WS2812 LED via GUI

● You can use the program with a graphical interface written by yourself to communicate with the Raspberry Pi on other devices to achieve the purpose of controlling the Raspberry Pi.

● The GUI programming method introduced in this chapter is completely done by Python language, specifically, the Tkinter library is used.

• Tkinter is Python's standard GUI library. Python uses Tkinter to quickly create GUI applications. Because Tkinter is built into the Python installation package, as long as Python is installed, you can import the Tkinter library, and IDLE is also written in Tkinter. For simple graphical interface Tkinter can still cope with it.

●We use the Socket library to communicate between devices. Socket is also called "socket". Applications usually send requests to the network or answer network requests through the "socket", so that the process between the host or a computer can communicate.

●In this chapter, we take the remote control of the LED lights as an example, because almost all of our robots are equipped with WS 2812 LED modules. This simpler example also helps novices to understand how the desktop GUI program works communicate with the Raspberry Pi.

●Ready to burn the Raspbian Raspberry Pi, you can refer to **the 9 module-WS2812 LED light** for related dependent libraries and connection methods. If you don't use Motor HAT, just connect the signal port (IN) of WS2812 LED to GPIO12 (BCM 18) of Raspberry Pi.

●For the detailed definition of Raspberry Pi pins, you can see this link to understand:Raspberry Pi Pinout

●Install the Python library used to control the WS2812 LED light. If it has not been installed or the robot installation script has not been run, you can use the following command to install it in the Raspberry Pi console:

```
sudo pip3 install rpi_ws281x
```

●We will use the Raspberry Pi as the server and the PC as the client.

●The program of the server in the Raspberry Pi is as follows:

```
'''
These two libraries are used to control WS2812 LED lights
'''
from rpi_ws281x import *
import argparse

'''
Import socket library to be used for TCP communication
'''
```

```
import socket

'''
Some settings related to LED lights come from the WS281X routine
Source Code:https://github.com/rpi-ws281x/rpi-ws281x-python/
'''
LED_COUNT       = 24
LED_PIN         = 18
LED_FREQ_HZ     = 800000
LED_DMA             = 10
LED_BRIGHTNESS  = 255
LED_INVERT      = False
LED_CHANNEL     = 0

'''
Process arguments
'''
parser = argparse.ArgumentParser()
parser.add_argument('-c', '--clear', action='store_true', help='clear the display on exit')
args = parser.parse_args()

'''
Create NeoPixel object with appropriate configuration.
'''
strip = Adafruit_NeoPixel(LED_COUNT, LED_PIN, LED_FPEQ_HZ, LED_DMA, LED_INVERT, LED_BRIGHTNESS,
LED_CHANNEL)

'''
Intialize the library
'''
strip.begin()

'''
Next is the configuration related to TCP communication, where PORT is the defined port number, you can choose freely
from 0-65535, it is recommended to select the number after 1023, which needs to be consistent with the port number
defined by the client in the PC
'''
HOST = ''
PORT = 10223
BUFSIZ = 1024
ADDR = (HOST, PORT)

tcpSerSock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

```
tcpSerSock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR,1)
tcpSerSock.bind(ADDR)
tcpSerSock.listen(5)

'''
Start listening to the client connection, after the client connection is successful, start to receive the information sent from the client
'''
tcpCliSock, addr = tcpSerSock.accept()

while True:
	data = ''

	'''
	Receive information from the client
	'''
	data = str(tcpCliSock.recv(BUFSIZ).decode())
	if not data:
		continue

	'''
	Turn on the light if the information content is on
	If the information content is off, turn off the light
	'''
	elif 'on' == data:
		for i in range(strip.numPixels()):
			strip.setPixelColor(i, Color(255, 0, 255))
			strip.show()
	elif 'off' == data:
		for i in range(strip.numPixels()):
			strip.setPixelColor(i, Color(0, 0, 0))
			strip.show()

	'''
    Finally print out the received data and start to continue listening to the next message from the client
	'''
	print(data)
```

●The program of the client in the PC is as follows:

```
'''
Import socket library to be used for TCP communication
'''
from socket import *
```

```
'''
Python uses Tkinter to quickly create GUI applications and instantiate them while importing
'''
import tkinter as tk

def lights_on():
	'''
	Call this method to send the light-on command 'on'
	'''
	tcpClicSock.send(('on').encode())

def lights_off():
	'''
	Call this method to send the light off command 'off'
	'''
	tcpClicSock.send(('off').encode())

'''
Enter the IP address of the Raspberry Pi here
'''
SERVER_IP = '192.168.3.35'

'''
Next is the configuration related to TCP communication, where PORT is a defined port number, you can choose freely
from 0-65535, it is recommended to choose the number after 1023, which needs to be consistent with the port number
defined by the server in the Raspberry Pi
'''
SERVER_PORT = 10223
BUFSIZ = 1024
ADDR = (SERVER_IP, SERVER_PORT)
tcpClicSock = socket(AF_INET, SOCK_STREAM)

tcpClicSock.connect(ADDR)

'''
The following is the GUI part
'''
root = tk.Tk()   # Define a window
root.title('Lights')   # Window title
root.geometry('175x55')# The size of the window, the middle x is the English letter x
root.config(bg='#000000')      # Define the background color of the window
```

```
'''
Use Tkinter's Button method to define a button. The button is on the root window. The name of the button is 'ON'. The text color of the button is # E1F5FE. The background color of the button is # 0277BD. )function
'''
btn_on = tk.Button(root, width=8, text='ON', fg='#E1F5FE', bg='#0277BD', command=lights_on)

'''
Choose a location to place this button
'''
btn_on.place(x=15, y=15)

'''
The same method defines another key, the difference is that the text above the key is changed to 'OFF'. When the key is pressed, the lights_off () function is called
'''
btn_off = tk.Button(root, width=8, text='OFF', fg='#E1F5FE', bg='#0277BD', command=lights_off)

btn_off.place(x=95, y=15)

'''
Finally open the message loop
'''
root.mainloop()
```

●We first run the program in the Raspberry Pi, and then open the program on the PC (first run the server and then the client).

● Click 'ON', the light is on, and 'on' is printed in the terminal of the Raspberry Pi, indicating that the program runs successfully.

❖ 25 Real-time Video Transmission Based on OpenCV

● This chapter introduces real-time video transmission, which can transmit the images collected by the camera to other places in real time for displaying images or handing it to the host computer for machine vision processing.

● The software functions of this tutorial are based on opencv, numpy, zmq (read Zero MQ) and base64 libraries. Before writing the code, you need to install these libraries.

```
pip3 install opencv-contrib-python numpy zmq pybase64
```

● In this tutorial, the hardware mainly uses a PC and a Raspberry Pi with a camera installed, because it can introduce the installation methods of related libraries on the Windows platform and Linux platform at the same time.

● OpenCV is an open source computer vision library. In the Linux system, you can enter in the terminal:

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

● In the windows system, you can install it by downloading the .whl file of opencv, or you can use the following command in the terminal to install OpenCV:

```
pip3 install opencv-contrib-python
```

● NumPy is a basic software package for scientific calculations using Python. In Linux, install numpy by typing **sudo pip3 install numpy** in the terminal.

● In Windows, install numpy by typing **pip3 install numpy** on the command line (cmd) (need to install python3.x in advance).

● zmq and base64 are used for frame transmission and frame encoding and decoding respectively in this project. In linux, enter **sudo pip3 install zmq pybase64** to install, and in windows, enter **pip3 install zmq pybase64** to install.

● After installing the relevant libraries, let's explain the program of the video sending end. The RPiCam.py python program is used to collect the pictures from the camera, and encode the collected pictures to the receiving end of the video. So we put RPiCam.py into the Raspberry Pi and run it.

● RPiCam.py:

```
'''
First import the required libraries, the above has a specific introduction to these libraries
'''
import cv2
import zmq
import base64
import picamera
from picamera.array import PiRGBArray

'''
Here we need to fill in the IP address of the video receiver (the IP address of the PC)
```

```
'''
IP = '192.168.3.11'

'''
Then initialize the camera, you can change these parameters according to your needs
'''
camera = picamera.PiCamera()
camera.resolution = (640, 480)
camera.framerate = 20
rawCapture = PiRGBArray(camera, size=(640, 480))

'''
Here we instantiate the zmq object used to send the frame, using the tcp communication protocol, where 5555 is the port number
The port number can be customized, as long as the port number of the sending end and the receiving end are the same
'''
context = zmq.Context()
footage_socket = context.socket(zmq.PAIR)
footage_socket.connect('tcp://%s:5555'%IP)
print(IP)

'''
Next, loop to collect images from the camera, because we are using a Raspberry Pi camera, so use_video_port is True
'''
for frame in camera.capture_continuous(rawCapture, format="bgr", use_video_port=True):

    '''
    Since imencode () function needs to pass in numpy array or scalar to encode the image
Here we convert the collected frame to numpy array
    '''
    frame_image = frame.array

    '''
    We encode the frame into stream data and save it in the memory buffer
    '''
    encoded, buffer = cv2.imencode('.jpg', frame_image)
    jpg_as_text = base64.b64encode(buffer)

    '''
    Here we send the stream data in the buffer through base64 encoding to the video receiving end
    '''
    footage_socket.send(jpg_as_text)
```

```
        '''
        Clear the stream in preparation for the next frame
        '''
        rawCapture.truncate(0)
```

● In the following, we explain the program on the receiving end. Since the libraries used here are cross-platform, PC.py can be run on a Windows computer or another Linux computer.

●PC.py :

```
'''
First import the required libraries
'''
import cv2
import zmq
import base64
import numpy as np

'''
Here we instantiate the zmq object used to receive the frame
Note that the port number needs to be consistent with the sender's
'''
context = zmq.Context()
footage_socket = context.socket(zmq.PAIR)
footage_socket.bind('tcp://*:5555')

while True:
        '''
        Received video frame data
        '''
        frame = footage_socket.recv_string()

        '''
        Decode and save it to the cache
        '''
        img = base64.b64decode(frame)

        '''
        Interpret a buffer as a 1-dimensional array
        '''
        npimg = np.frombuffer(img, dtype=np.uint8)

        '''
        Decode a one-dimensional array into an image
        '''
        source = cv2.imdecode(npimg, 1)
```

```
    '''
    Display image
    '''
    cv2.imshow("Stream", source)

    '''
    Generally, waitKey () should be used after imshow () to leave time for image drawing, otherwise the window will
appear unresponsive and the image cannot be displayed
    '''
    cv2.waitKey(1)
```

●When running the program, we first run RPiCam.py in the Raspberry Pi and PC.py in the PC to see the real-time picture of the Raspberry Pi in the PC.

❖ 26 Use OpenCV to Process Video Frames on The PC

●Due to the limited computing power of the Raspberry Pi, our OpenCV in the Raspberry Pi can only guarantee a relatively high frame rate when implementing simple functions such as color recognition and visual line inspection. If we need more complex machine vision functions , We need to send the video frames that need to be analyzed to the device equipped with advanced GPU to process, and finally send the processed results to the robot where the Raspberry Pi is located to perform the corresponding operation, and the machine vision of the Raspberry Pi robot The ability is stronger, thus achieving more advanced functions.

●We can refer to the content of **26** to send video frames to the host computer, or refer to the content of **15** to let the Raspberry Pi put the video stream on a page, and the host computer obtains the video stream from the page to analyze the video frame.

●The content of this chapter is based on **26**. First, we open PC.py as follows:

```
'''
First import the required libraries
'''
import cv2
import zmq
import base64
import numpy as np

'''
Here we instantiate the zmq object used to receive the frame
Note that the port number needs to be consistent with the sender's
'''
context = zmq.Context()
```

```
footage_socket = context.socket(zmq.PAIR)
footage_socket.bind('tcp://*:5555')

while True:
    '''
    Received video frame data
    '''
    frame = footage_socket.recv_string()

    '''
    Decode and save it to the cache
    '''
    img = base64.b64decode(frame)

    '''
    Interpret a buffer as a 1-dimensional array
    '''
    npimg = np.frombuffer(img, dtype=np.uint8)

    '''
    Decode a one-dimensional array into an image
    '''
    source = cv2.imdecode(npimg, 1)

    '''
    Display image
    '''
    cv2.imshow("Stream", source)

    '''
    Generally, waitKey () should be used after imshow () to leave time for image drawing, otherwise the window will
appear unresponsive and the image cannot be displayed
    '''
    cv2.waitKey(1)
```

● After source = cv2.imdecode (npimg, 1), you can use OpenCV to process the source, as shown below is the routine for binarizing the real-time video image from the Raspberry Pi using the host computer:

```
'''
First import the required libraries
'''
import cv2
import zmq
import base64
import numpy as np

'''
Here we instantiate the zmq object used to receive the frame
Note that the port number needs to be consistent with the sender's
'''
context = zmq.Context()
footage_socket = context.socket(zmq.PAIR)
footage_socket.bind('tcp://*:5555')

while True:
    '''
    Received video frame data
    '''
    frame = footage_socket.recv_string()

    '''
    Decode and save it to the cache
    '''
    img = base64.b64decode(frame)

    '''
    Interpret a buffer as a 1-dimensional array
    '''
    npimg = np.frombuffer(img, dtype=np.uint8)

    '''
    Decode a one-dimensional array into an image
    '''
    source = cv2.imdecode(npimg, 1)

    '''
    Convert image to grayscale
    '''
    source = cv2.cvtColor(source, cv2.COLOR_BGR2GRAY)

    '''
    Binary image
    '''
    retval, source =    cv2.threshold(source, 0, 255, cv2.THRESH_OTSU)

    '''
Remove small noise in the image
```

```
        '''
        source = cv2.erode(source, None, iterations=6)

        '''
        Display image
        '''
        cv2.imshow("Stream", source)

        '''
        Generally, waitKey () should be used after imshow () to leave time for image drawing, otherwise the window will
appear unresponsive and the image cannot be displayed
        '''
        cv2.waitKey(1)
```

❖ 27 Enable UART

● UART is a more commonly used communication protocol between devices. Using UART, you can allow MCUs such as Arduino, STM32, or ESP32 to communicate with the Raspberry Pi, which can make your robot more powerful.

● However, for some Raspberry Pis, the UART that is enabled by default is not a full-featured UART, so you need to refer to the following steps to enable the full-featured UART. The following parts are from the official documentation of the Raspberry Pi The Raspberry Pi UARTs.

● The SoCs used on the Raspberry Pis have two built-in UARTs, a PL011 and a mini UART. They are implemented using different hardware blocks, so they have slightly different characteristics. However, both are 3.3V devices, which means extra care must be taken when connecting up to an RS232 or other system that utilises different voltage levels. An adapter must be used to convert the voltage levels between the two protocols. Alternatively, 3.3V USB UART adapters can be purchased for very low prices.

● By default, on Raspberry Pis equipped with the wireless/Bluetooth module (Raspberry Pi 3 and Raspberry Pi Zero W), the PL011 UART is connected to the Bluetooth module, while the mini UART is used as the primary UART and will have a Linux console on it. On all other models, the PL011 is used as the primary UART.

● In Linux device terms, by default, /dev/ttyS0 refers to the mini UART, and /dev/ttyAMA0 refers to the PL011. The primary UART is the one assigned to the Linux console, which depends on the Raspberry Pi model as described above. There are also symlinks:

/dev/serial0, which always refers to the primary UART (if enabled), and /dev/serial1, which similarly always refers to the secondary UART (if enabled).

27.1 Mini UART and CPU Core Frequency

● The baud rate of the mini UART is linked to the core frequency of the VPU on the VC4 GPU. This means that, as the VPU frequency governor varies the core frequency, the baud rate of the mini UART also changes. This makes the mini UART of limited use in the default state. By default, if the mini UART is selected for use as the primary UART, it will be disabled. To enable it, add enable_uart=1 to config.txt. This will also fix the core frequency to 250MHz (unless force_turbo is set, when it will be fixed to the VPU turbo frequency). When the mini UART is not the primary UART, for example you are using it to connect to the Bluetooth controller, you must add core_freq=250 to config.txt, otherwise the mini UART will not work.

● The default value of the enable_uart flag depends on the actual roles of the UARTs, so that if ttyAMA0 is assigned to the Bluetooth module, enable_uart defaults to 0. If the mini UART is assigned to the Bluetooth module, then enable_uart defaults to 1. Note that if the UARTs are reassigned using a Device Tree Overlay (see below), enable_uart defaults will still obey this rule.

27.2 Disabling Linux's Use of Console UART

● In a default install of Raspbian, the primary UART (serial0) is assigned to the Linux console. Using the serial port for other purposes requires this default behaviour to be changed. On startup, systemd checks the Linux kernel command line for any console entries, and will use the console defined therein. To stop this behaviour, the serial console setting needs to be removed from command line.

●This can be done by using the raspi-config utility, or manually.

sudo raspi-config

●Select option 5, Interfacing options, then option P6, Serial, and select No. Exit raspi-config.

●To manually change the settings, edit the kernel command line with sudo nano

/boot/cmdline.txt . Find the console entry that refers to the serial0 device, and remove it, including the baud rate setting. It will look something like console=serial0,115200 . Make sure the rest of the line remains the same, as errors in this configuration can stop the Raspberry Pi from booting.

●Reboot the Raspberry Pi for the change to take effect.

27.3 UART Output on GPIO Pins

●By default, the UART transmit and receive pins are on GPIO 14 and GPIO 15 respectively, which are pins 8 and 10 on the GPIO header.

27.4 UARTs and Device Tree

● Various UART Device Tree Overlay definitions can be found in the kernel github tree. The two most useful overlays are disable-bt and miniuart-bt .

●disable-bt disables the Bluetooth device and restores UART0/ttyAMA0 to GPIOs 14 and 15. It is also necessary to disable the system service that initialises the modem so it doesn't use the UART: sudo systemctl disable hciuart .

● miniuart-bt switches the Raspberry Pi 3 and Raspberry Pi Zero W Bluetooth function to use the mini UART (ttyS0), and restores UART0/ttyAMA0 to GPIOs 14 and 15. Note that this may reduce the maximum usable baudrate (see mini UART limitations below). It is also necessary to edit /lib/systemd/system/hciuart.service and replace ttyAMA0 with ttyS0, unless you have a system with udev rules that create /dev/serial0 and /dev/serial1. In this case, use /dev/serial1 instead because it will always be correct. If cmdline.txt uses the alias serial0 to refer to the user-accessible port, the firmware will replace it with the appropriate port whether or not this overlay is used.

●There are other UART-specific overlays in the folder. Refer to /boot/overlays/README for details on Device Tree Overlays, or run dtoverlay -h overlay-name for descriptions and usage information.

●For full instructions on how to use Device Tree Overlays see this page. In brief, add a line to the config.txt file to enable Device Tree Overlays. Note that the -overlay.dts part of the filename is removed.

```
...
dtoverlay=disable-bt
...
```

27.5 Relevant Differences Between PL011 and Mini UART

●The mini UART has smaller FIFOs. Combined with the lack of flow control, this makes it more prone to losing characters at higher baudrates. It is also generally less capable than the PL011, mainly due to its baud rate link to the VPU clock speed.

●The particular deficiencies of the mini UART compared to the PL011 are :

- No break detection
- No framing errors detection
- No parity bit
- No receive timeout interrupt
- No DCD, DSR, DTR or RI signals

Further documentation on the mini UART can be found in the SoC peripherals document here.

❖ 28 Control Your AWR with An Android Device

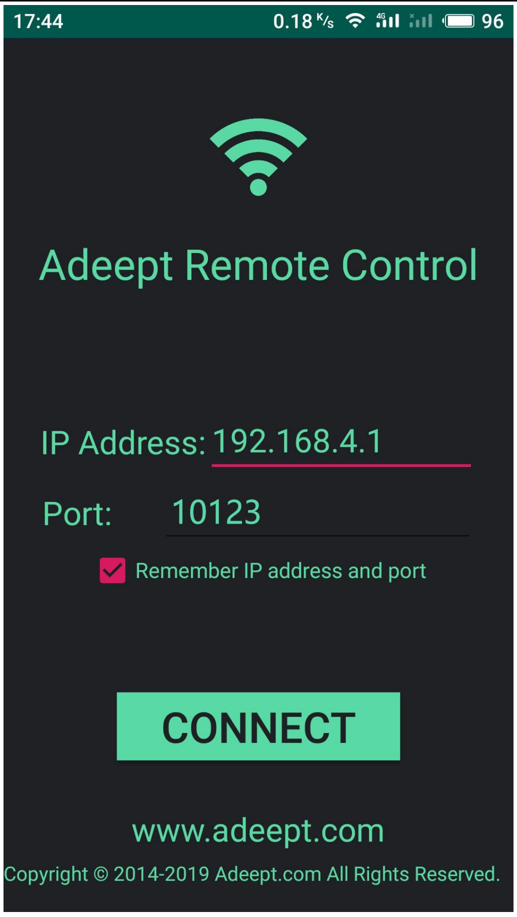
●If you want to use a mobile phone or tablet to control the robot, we first recommend that you use the WEB application to control the robot, because the WEB application has more functions, maintenance and updates are more frequent, and most importantly, the WEB application can be cross-platform No matter whether you use Android system or iOS system, as long as Google Chrome is installed, you can use the WEB application to control the robot.

●For the usage method of WEB application, please refer to **5 Using WEB Application to Control Robot**.

●Our old version of the robot program is equipped with the method of using the mobile phone APP to control, but the mobile phone APP only supports Android phones and cannot use the functions related to the Raspberry Pi camera. The program corresponding to this function in the Raspberry Pi is **appserver.py**

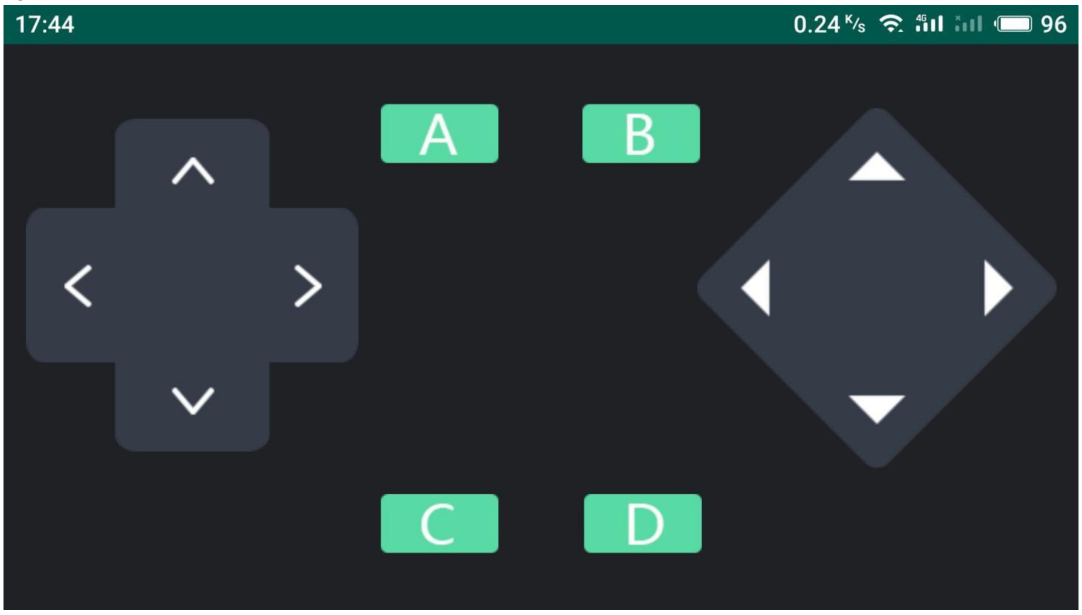
●You can find the download address of the mobile APP in our official website. The installation method is the same as that of the ordinary mobile phone APP.

●Open the mobile app, enter the IP address of the Raspberry Pi in the IP address field of the mobile app, and enter 10123 in the port number. Click.**Connect**



●It should be noted that the port number when using the WEB application is 5000, the port number when using the GUI program is 10223, and the port number when using the mobile APP is 10123.

●The controller on the left can control the robot to move back and forth, left and right, and the controller on the right can control other movements of the robot. You can change the specific operation by editing **appserver.py**.



Conclusion

Through the above operations on DarkPaw, you should learn how to use python language programming to control DarkPaw work on the Raspberry Pi, And also learned how to assemble a DarkPaw.

If you have any questions about this product, please contact us via email or forum, we will reply to your questions within one working day:

support@adeept.com

https://www.adeept.com/forum/

If you want to try our other products, you can visit our website:

www.adeept.com

For more product information, please visit:

https://www.adeept.com/learn/

For more product latest video updates, please visit:

https://www.adeept.com/video/

Thank you for using Adeept products.

Adeept

STEM Education Products and Service Provider

Shenzhen Adeept Technology Co., Ltd.

: www.adeept.com
: support@adeept.com
: Rm.1315, Shihong Building, No.2095 Bixin Rd.,
Longgang Dist., Shenzhen CHINA