Adeept

RaspTank Pro

www.adeept.com

Resources Link

RobotName: AdeeptRaspTankPro

RobotURL: https://github.com/adeept/adeept_rasptankpro

RobotGit: https://github.com/adeept/adeept_rasptankpro.git

[Official Raspberry Pi website] https://www.raspberrypi.org/downloads/

[Official website] https://www.adeept.com/

[GitHub] https://github.com/adeept/adeept_rasptankpro/

[Image file and Documentation for structure assembly] https://www.adeept.com/learn/detail-49.html

Content

Premise

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.

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.

Introduction of RaspTank Pro Products

1. About RaspTank Pro products

RaspTank Pro is an open source intelligent robot product for artificial intelligence and robot enthusiasts and students. It is also an open Raspberry Pi-based robot development platform. It has the following features:

Easy to assemble: adopting structural modular design, open hardware list and detailed assembly tutorial.

Easy to learn: Complete and detailed development learning tutorials and sample codes are provided from algorithms to applications.

Rich functions: automatic obstacle avoidance, color recognition, color tracking, moving object detection, Web remote control, OLED display.

Aluminum alloy structure: strong and durable.

Extensible: the structure can be expanded and DIY.

Web remote control: Regardless of mobile phone, tablet, computer, Windows, Linux, Mac OS, as long as you can install Google Chrome browser to control the robot.

Support multiple versions of Raspberry Pi: support Raspberry Pi 3B, Raspberry Pi 3B+ and Raspberry Pi 4.

Supporting Python.

RaspTank Pro has the functions of grabbing objects and visual line tracking, color recognition, moving object detection, Web remote control, OLED display, etc.; RaspTank Pro uses a decelerated DC motor as a power unit, which has the advantage of fast speed; using crawler wheels makes RaspTank Pro has excellent off-road performance and can be applied to complex terrain; the program only needs to control the high and low levels of the corresponding GPIO ports to control the car to grab items, which is

convenient for novice makers to quickly learn and use the control methods of car products；

1. Raspberry Pi

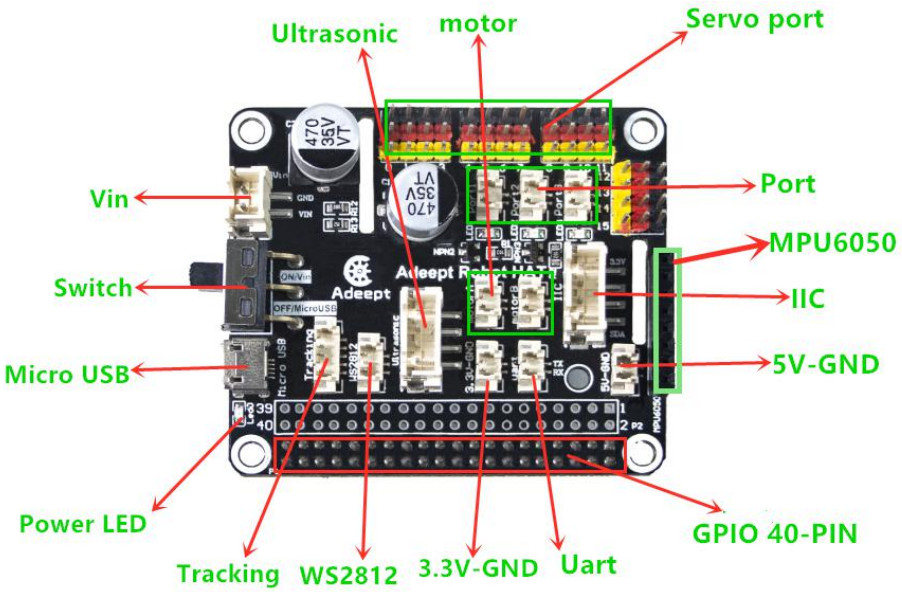
1.1 Introduction to Raspberry Pi

1.1.1 Raspberry Pi

Raspberry Pi (Raspberry Pi, RasPi/RPi) is developed by the British charity organization "Raspberry Pi Foundation", based on ARM microcomputer motherboard, only the size of a credit card, but has the basic functions of a personal computer. The original purpose of the Foundation's development of the Raspberry Pi was to improve the teaching level of the school's computer science and related disciplines, and cultivate the youth's computer programming interest and ability. Nowadays, most people use the Raspberry Pi for embedded development, which is mostly used in the Internet of Things, smart home and artificial intelligence.

1.1.2 Raspberry Pi motherboard

In our lessons, we will use the Raspberry Pi 4 motherboard. Let's take a look at the structure of the Raspberry Pi 4 motherboard. As shown in the following figure:



The following contents will briefly explain the main structure ports of the Raspberry Pi 4 motherboard:

(1) GPIO 40-PIN pin:

The General Purpose Input Output (GPIO) is designed as a slot with two rows of pins on the Raspberry Pi motherboard. GPIO can be used to connect various peripheral electronic devices and sensors to control or monitor these devices through input/output level signals. For example, you can use GPIO to control the speed of a DC motor, or read the measured distance of an ultrasonic sensor. These functional characteristics of GPIO make the Raspberry Pi different from ordinary computer motherboards because it gives developers the freedom to operate manually. We will further introduce GPIO in the subsequent chapters and use them extensively.

(2) Gigabit Ethernet port:

The Ethernet interface allows the Raspberry Pi to connect to the computer network in a wired manner, which allows us to easily access the Internet or log in to the Raspberry Pi remotely. The Raspberry Pi's Ethernet interface is implemented using a USB bus, and data is transferred through the USB bus. Most models of Raspberry Pi provide an Ethernet interface

(3) Micro HDMI port:

High-definition multimedia interface (High Definition Multimedia Interface, HDMI) is a fully digital video and sound transmission interface, used to transmit uncompressed audio and video signals. By connecting it to a display (or TV) equipped with an HDMI interface, the content of the Raspberry Pi can be displayed. The HDMI interface can transmit video and audio signals at the same time, so when we use it, we don't need to connect speakers to the audio interface of the Raspberry Pi. If we really need to play sound through the audio interface, we need to modify the operating system configuration accordingly.

(4) USB2.0/3.0 port:

The Universal Serial Bus (USB) interface is the most common interface on a computer. You can use it to connect devices such as keyboards, mice, USB flash drives, and wireless network cards. When the number of USB ports is not enough, we can also increase the number of USB ports through a USB hub.

(5) Audio port:

Audio interface (3.5mm headphone jack) When HDMI connection is not used, you can use the standard 3.5mm headphone jack speakers or headphones to play audio. At the same time, the interface also integrates a composite video interface with a composite audio and video output function, which is generally used to connect to old models of TVs, and is currently rarely used.

(6) MIPI CSI camera port:

The CSI interface can be used to connect the CSI camera to the Raspberry Pi via a ribbon cable for easy video recording and image capture. Compared with the USB camera, this camera module has better performance.

(7) USB-C 5V/3A power supply port:

The Micro USB power supply interface is one of the main power supply methods of the Raspberry Pi. The rated voltage is 5V. The standard current requirements of different versions of the Raspberry Pi are slightly different. For example: the 1B type only needs 700mA, and the 3B+ type requires 2.5A. The chargers of many Android mobile phones can provide the necessary voltage and current for the Raspberry Pi. The current demand of the Raspberry Pi is also related to the connected external device. It is recommended that it should be calculated in advance when using it. Choose a suitable current (power) power supply for the Raspberry Pi. When the external device has a large power, an independent power supply should be used Power supply for external devices.

(8) Micro SD card slot:

The SD card slot is located on the back of the Raspberry Pi motherboard. The SD/MicroSD card is an essential storage part of the Raspberry Pi. It is used to install the operating system and store data. The capacity of the SD card should be above 2GB. In order to have a better experience, it is recommended to equip your Raspberry Pi with a large-capacity (above 16G) high-speed (Class10 or above) SD card.

(9) Bluetooth port:

The Bluetooth function allows the Raspberry Pi to connect with Bluetooth-enabled devices (such as a mouse, keyboard, and handle).

(10) PoE HAT port:

Active Ethernet (Power Over Ethernet, PoE) refers to a technology that uses Ethernet for power transmission. On the basis of the original Micro USB and GPIO power supply, the Raspberry Pi 3B+ type adds a new power supply method over Ethernet. Users can use the network cable to supply power to the Raspberry Pi without the need to configure an additional power supply, which is convenient for certain application scenarios.

(11) MIPI DSI display port:

You can connect the LCD display to the Raspberry Pi, which is generally used for embedded product development. Under normal circumstances, the HDMI interface can already meet the demand.

1.1.3 Operating system

The Raspberry Pi supports a variety of operating systems, mainly based on Liunx and Windows, and most of them can be found on the official website of the Raspberry Pi Foundation (www.raspberrypi.org). The following briefly introduces two representative operating systems.

(1) Raspbian

Raspbian is the official operating system of the Raspberry Pi Foundation. It is customized based on Debian GNU/Linux and can run on all versions of the Raspberry Pi motherboard. According to the experience, Raspbian and Raspberry Pi combine the best, stable operation, powerful, easy to use, can basically meet various application needs, so it is strongly recommended to use Raspbian as the preferred operating system for Raspberry Pi. In the following chapters, we will further introduce the use of Raspbian in detail, and develop various applications on it.

(2) Windows 10 IoT Core

Windows 10 IoT Core is an operating system specifically created by Microsoft for the Internet of Things ecosystem. Windows 10 IoT Core is the core version of the Windows 10 IoT operating system. It has relatively simple functions and can run on the Raspberry Pi of type 2B or above. The installation and use of Windows 10 IoT Core will not be described in detail here. If you are interested, you can visit Microsoft's website for more information.

In addition to the two operating systems described above, there are several operating systems that support the Raspberry Pi, such as Ubuntu MATE, OSMC, LibreELEC, PiNet, RISC OS, etc. As for which one to choose, it depends on whether you want to use Raspberry What to do. If you want to use the Raspberry Pi as an ordinary computer or for electronic project development, then Raspbian is a very good choice. If you plan to use the Raspberry Pi as a media center, you can consider using OSMC or LibreELEC.

1.1.4 Programming language

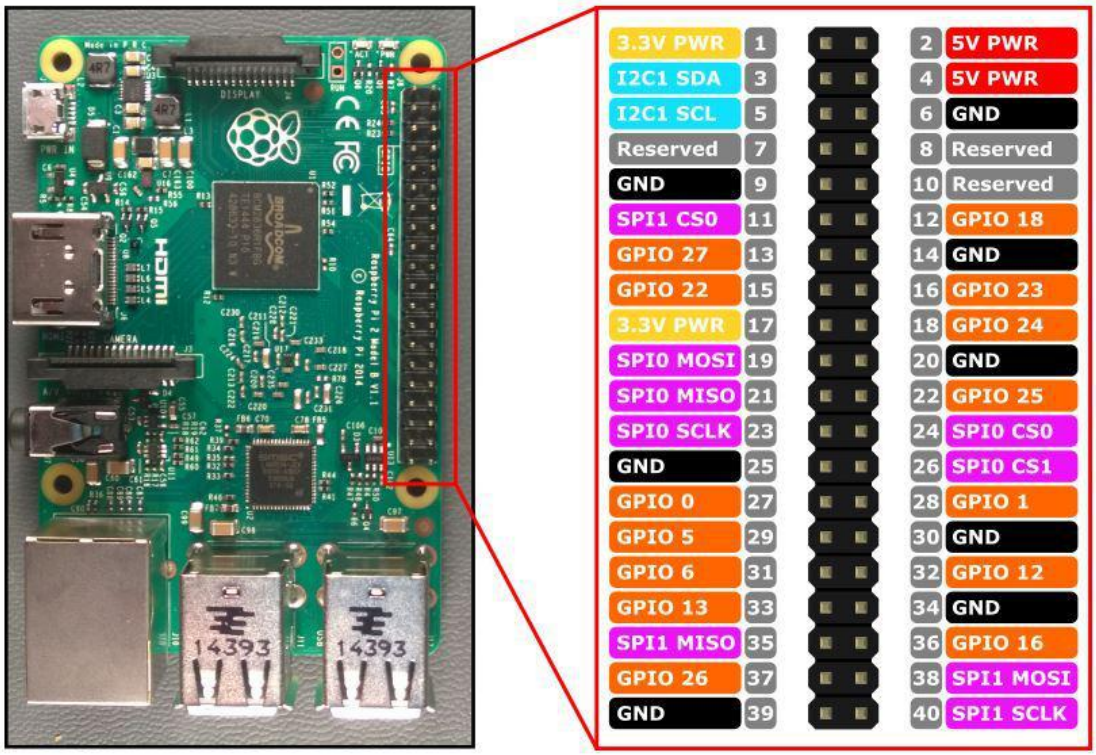
For the Raspberry Pi, there are many programming languages available. In fact, any language that can be compiled for the ARM architecture (such as the C language) can be used for the Raspberry Pi. The most popular language should be Python. In fact, the Pi in the name of the Raspberry Pi was inspired by the word Python. Python is an interpretive, object-oriented, and dynamic data type high-level programming language with powerful functions, good compatibility, and high reliability. Python programs are easy to write and

read. At present, there are two major versions of Python: Python 2 and Python 3. Both versions have been updated and maintained, but people still have disputes about which version to use. You can visit Python's official website (www.python.org) to understand more related content, in the future we will mainly use Python 3 for development introduction. In addition, because the compatibility of the Raspberry Pi is splendid, the program we wrote on the 3B+ model can be run on the Zero W model with little modification.

1.2 Introduction to GPIO

1.2.1 What is GPIO

GPIO (General Purpose I/O Ports) are general-purpose input/output ports. In layman's terms, they are some pins with two rows of pins. They can be used to output high and low levels or to read the state of the pins-whether it is high or low. Users can interact with the hardware through the GPIO port (such as UART), control the work of the hardware (such as LED, buzzer, etc.), read the working status signal of the hardware (such as interrupt signal), etc.



1.2.2 Introduction of GPIO pins

(1) GPIO pin comparison table

Raspberry Pi 40Pin Pin Comparison Table

wiringPi Encoding	BCM Encoding	Function Name	BOARD Encoding of Physical Pins		Function Name	BCM Encoding	wiringPi Encoding
		3.3V	1	2	5V		
8	2	SDA.1	3	4	5V		
9	3	SCL.1	5	6	GND		
7	4	GPIO.7	7	8	TXD	14	15
		GND	9	10	RXD	15	16
0	17	GPIO.0	11	12	GPIO.1	18	1
2	27	GPIO.2	13	14	GND		
3	22	GPIO.3	15	16	GPIO.4	23	4
		3.3V	17	18	GPIO.5	24	5
12	10	MOSI	19	20	GND		
13	9	MISO	21	22	GPIO.6	25	6
14	11	SCLK	23	24	CE0	8	10
		GND	25	26	CE1	7	11
30	0	SDA.0	27	28	SCL.0	1	31
21	5	GPIO.21	29	30	GND		
22	6	GPIO.22	31	32	GPIO.26	12	26
23	13	GPIO.23	33	34	GND		
24	19	GPIO.24	35	36	GPIO.27	16	27
25	26	GPIO.25	37	38	GPIO.28	20	28
		GND	39	40	GPIO.29	21	29

【Form description】:

(1) Three naming (coding) methods for Raspberry Pi pins

Three ways to name the Raspberry Pi pins:

The WiringPi number is the pin number of the functional wiring (such as TXD, PWM0, etc.); the BCM number is the Broadcom pin number, also known as GPIO; the

physical number is the number corresponding to the physical location of the pin on the Raspberry Pi motherboard (1 ~40).

(2) 3.3V/5V pin and GND pin

3.3V/5V pin and GND pin are commonly known as power and ground pins. The power and ground pins allow your Raspberry Pi to power some external components, such as LED lights. It should be noted that before using these pins to power any external modules or components, care should be taken. Excessive operating current or peak voltage may damage the Raspberry Pi. Do not use voltages greater than 5V!

(3) SDA and SCL pins

The SDA and SCL pins constitute the I2C interface. I2C is a simple, bidirectional two-wire synchronous serial bus developed by Philips. It only requires two wires to transfer information between devices connected to the bus. The Raspberry Pi can control multiple sensors and components through the I2C interface. Their communication is done through SDA (data pin) and SCL (clock speed pin). Each slave device has a unique address, allowing rapid communication with many devices. The ID_EEPROM pin is also an I2C protocol, which is used to communicate with HATs.

(4) SCLK, MOSI and MISO pins

SCLK, MOSI and MISO pins form the SPI interface. SPI is a serial peripheral interface, used to control components with a master-slave relationship, and works in a slave-in, master-out and master-in-slave manner. The SPI on the Raspberry Pi consists of SCLK, MOSI, and MISO interfaces, and SCLK is used for controlling data speed, MOSI sends data from the Raspberry Pi to the connected device, while MISO does the opposite.

(5) TXD and RXD pins

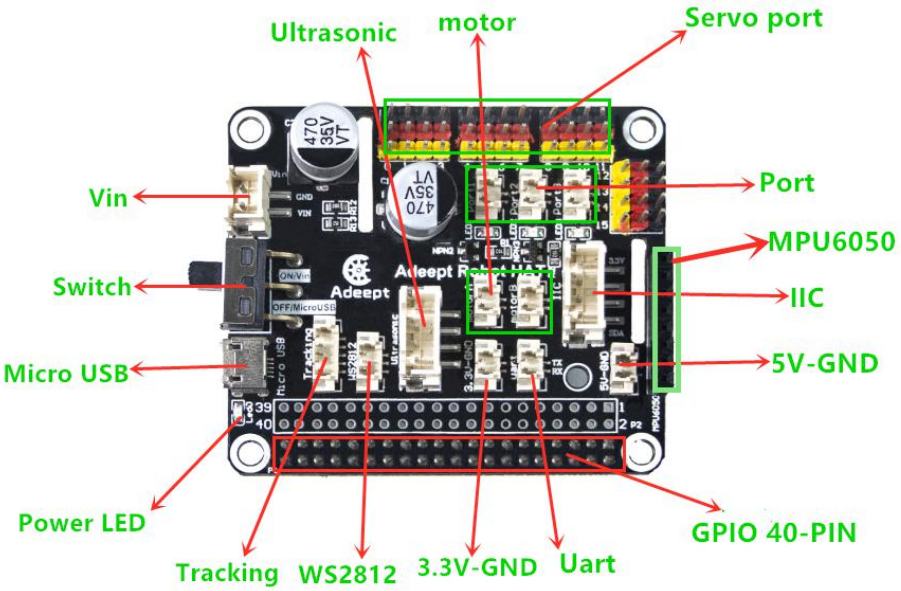
TXD and RXD form a UART interface. TXD is a pin to send data, and RXD is a pin to receive data. A friend who uses Arduino must have heard of UART or Serial. The Universal Asynchronous Receiver/Transmitter interface is used to connect the Arduino to

the computer for which it is programmed. It is also used for communication between other devices and the RX and TX pins. If the Raspberry Pi has a serial terminal enabled in raspi-config, you can use these pins to control the Raspberry Pi through a computer or directly to control the Arduino.

1.2 Introduction of Robot HAT Driver Board

1.2.1Introduction of Robot HAT driver board

When you get the robot product, you will see a board with its name printed on it called: Adeept Robot HAT, which is an important part of robot. There are many interfaces on the Robot HAT driver board. By these interfaces, you can connect some sensors and electronic hardware modules, so that you can achieve many extended functions. Our robot products need to be used in conjunction with the Raspberry Pi. Let's first get to know the Robot HAT driver board.



【Vin】： The vin interface is an interface for external power supply.

【Switch】： Switch is the switch of Robot HAT driver board, ON is to open, and OFF is to close.

【Micro USB】： The Micro USB interface can connect the Robot HAT driver board to a computer or other equipment, and can also supply power for the Robot HAT driver board.

【Power LED】 Power LED is used to indicate the power status of Robot HAT driver board. If the LED is on, it means that the Robot HAT driver board is powered on and can run; if the LED is off, it means that the Robot HAT driver board is not powered on.

【Tracking】 is the pin interface of Tracking Module.

【WS2812】 is the pin interface of WS2812 Module.

【3.3V-GND】 3.3V power supply interface.

【Uart】 Uart interface.

【GPIO 40-PIN】 General Purpose Input Output (GPIO) is designed as a slot with two rows of pins on the Robot HAT driver board. GPIO can be used to connect various peripheral electronic devices and sensors and control or monitor these devices with input/output level signals. In robot products, this GPIO interface is connected to the GPIO pins on the Raspberry Pi driver board.

【5V-GND】 5V power supply interface.

【IIC】IIC interface. It is also the interface of the OLED screen module.

【MPU6050】The interface of MPU6050 sensor.

【Port】is divided into Port1, Port2, and Port3 interfaces, which are commonly used to connect Small LED light.

【Servo port】Servo interface.

【motor】 is divided into motor1, motor2 interfaces.

【Ultrasonic】Ultrasonic interface.

1.2.2 Precautions for the use of Robot HAT driver board

When you are performing software installation, structural assembly or program debugging, you can use a USB cable to power the Raspberry Pi. If the Raspberry Pi is equipped with Robot HAT, you can connect the USB cable to the USB port on the Robot HAT. Robot HAT will power the Raspberry Pi by the GPIO interface.

Different Raspberry Pi have different current requirements. For example, the Raspberry Pi 3B needs at least 2A to boot up, and the Raspberry Pi 4 needs 3A to boot normally. When you use the power adapter to power the Raspberry Pi, you can check the specifications on your power adapter.

When the Robot HAT is connected to a load, such as a motor or multiple servos, you need to use a high-current power supply to connect to the Vin on the Robot HAT. You can use two 18650 batteries that support high-current to power the Robot HAT. For power supply, our product will provide a dual 18650 battery box with a 2pin interface. You can directly connect it to the Robot HAT.

When the USB interface on the Robot HAT is used for power supply, the switch of the Robot HAT does not control whether to supply power. The switch of the Robot HAT can only control the power supply of Vin.

Do not use the USB port on the Robot HAT and Vin to supply power at the same time. If you need to debug the program for a long time and don’t want to remove the battery, you can set the switch on the Robot HAT to OFF, so that when the USB cable is used to connect the Robot HAT, the Robot HAT is powered by USB.

If your robot restarts automatically after it is turned on, or after it is turned on normally, it is disconnected and restarted at the moment when the robot starts to move, it is likely that your power supply does not output enough current. The robot will automatically restart when it is turned on. Run the program to place all the servos in the neutral position. The voltage drop generated during this process causes the Raspberry Pi to restart.

We have tested that the peak current of the robot is around 3.75A when powered by 7.4V, so you need to use a battery that supports 4A output.

You can also use the power lithium battery to power the Robot HAT. Robot HAT supports power supply below 15V.

When assembling and installing the servo rocker arm, you can use a USB cable to power the Robot HAT. After the Raspberry Pi with the robot software is installed, it will control the Robot HAT to set all the servo ports to output neutral signals. You can connect the servo to any port. The gear of the servo will rotate to the neutral position, and then you can install the servo rocker arm according to the specified angle. After the rocker arm is installed, you can disconnect the servo from the Robot HAT , When you need to install the rocker arm of the second servo, connect the second servo to any servo port on the drive board.

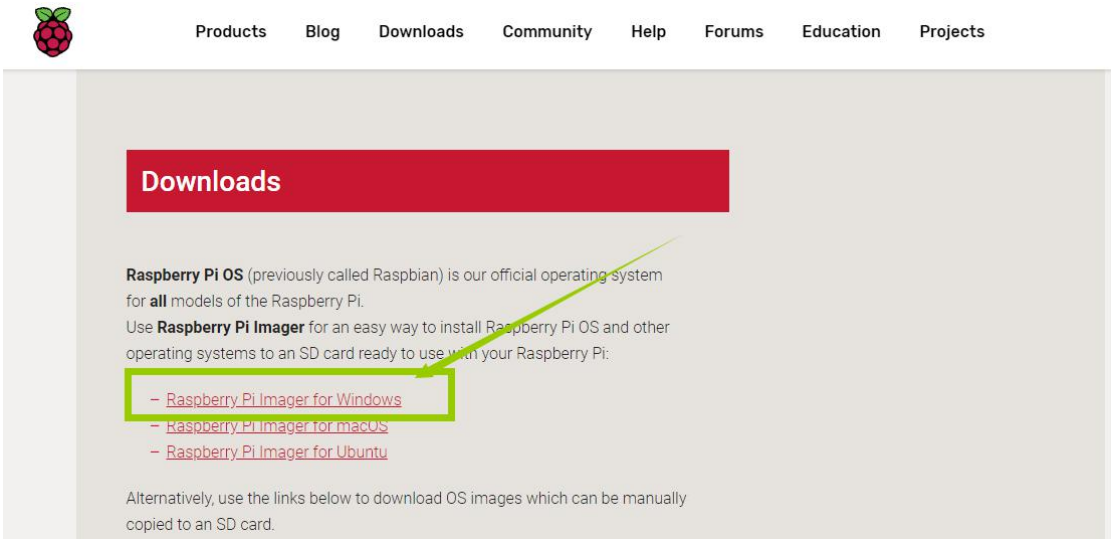
2. Installing and Configuring Raspberry Pi System

2.1 Downloading the installation software for the Raspberry Pi system

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

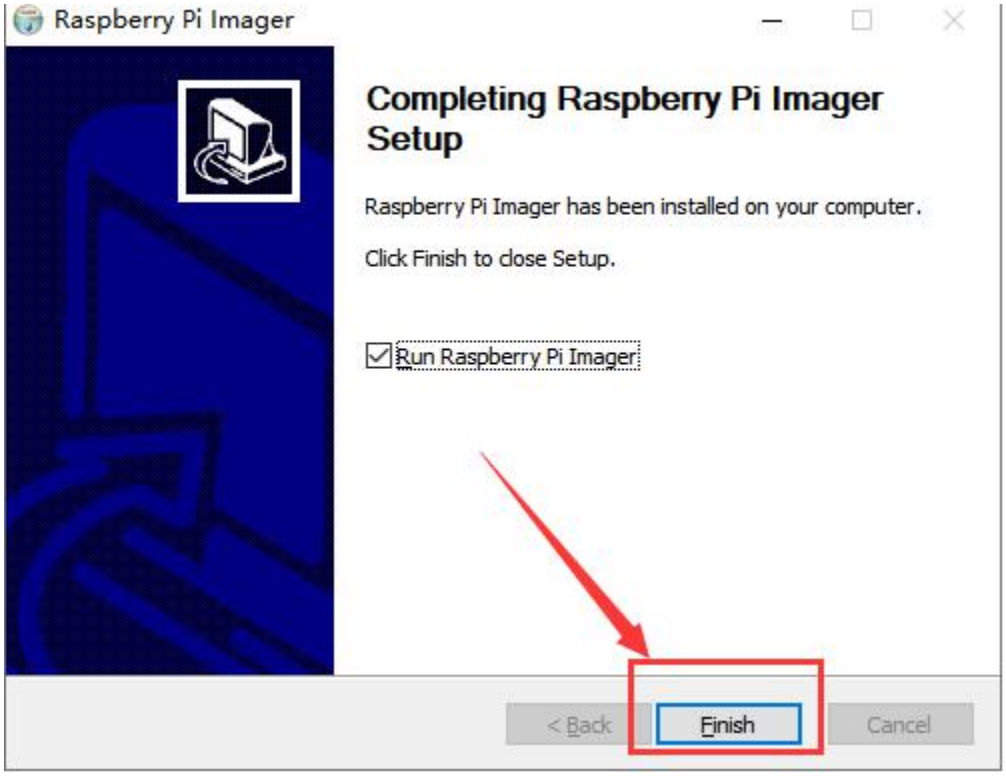
Step-by-Step Overview

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. After the download is complete, install the software and burn the Raspberry Pi system. Now take Windows as an example.



1. Open the file "imager.exe" after the download is complete, and click "Install".

2. Then click "Finish".



3. The software interface after opening is as follows:

2.2 Downloading the Raspberry Pi system Raspbian

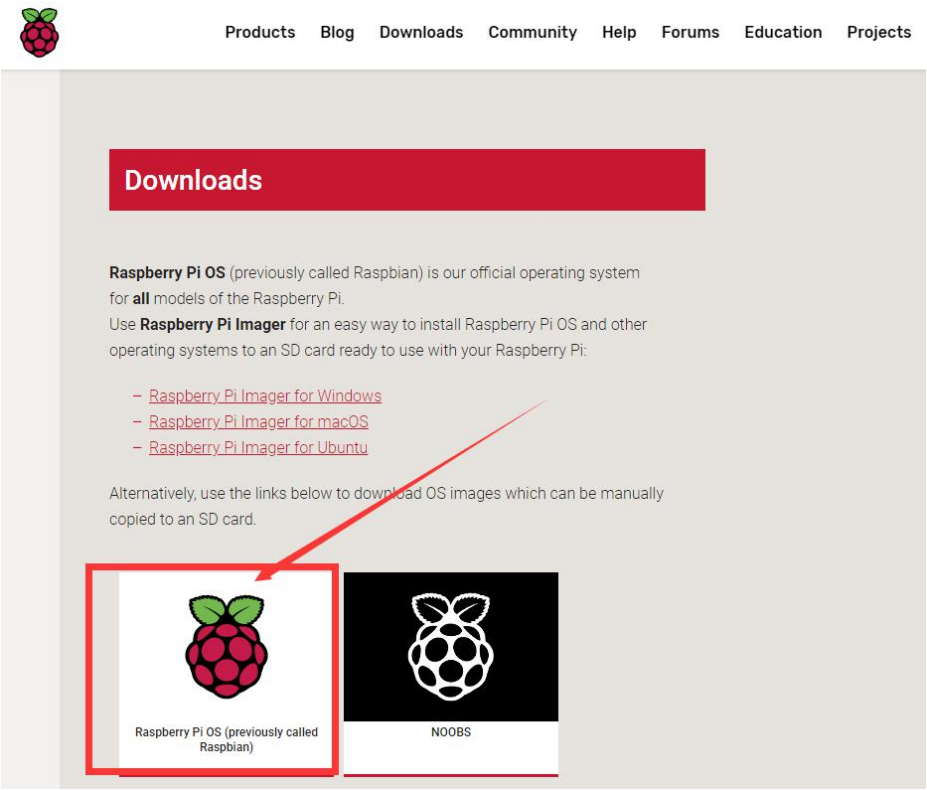
Raspbian is the official operating system of the Raspberry Pi Foundation. It is customized based on Debian GNU/Linux and can run on all versions of the Raspberry Pi motherboard. According to the experience, Raspbian combines Raspberry Pi the best. It is stable, powerful, and easy to use. It can basically meet the needs of various applications. This course uses Raspbian as the preferred operating system for the Raspberry Pi. Next, we will teach you how to download the Raspberry Pi system Raspbian. Now there are two ways to download Raspbian for Raspberry Pi system (we recommend method one first)

2.2.1 Method one :

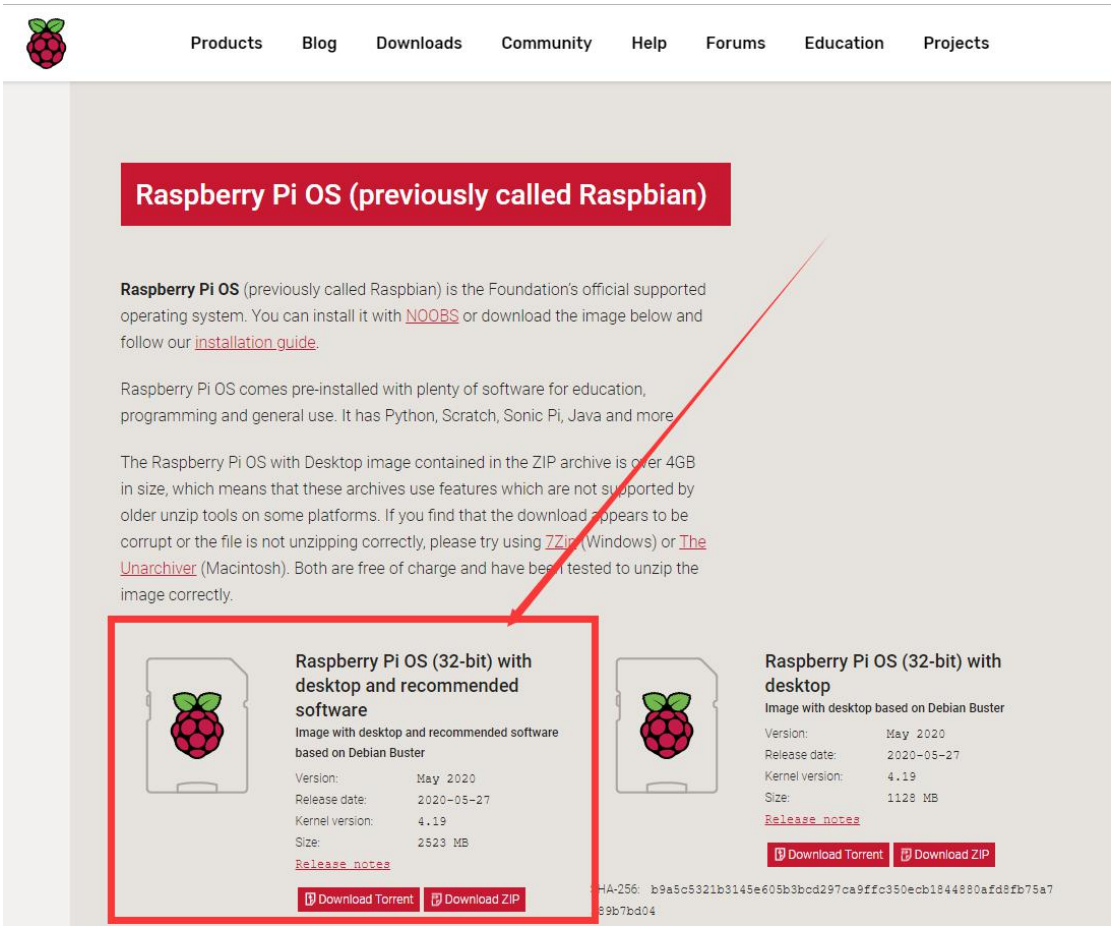
(1) visit the official website of the Raspberry Pi through a browser to download Raspbian:

https://www.raspberrypi.org/downloads/

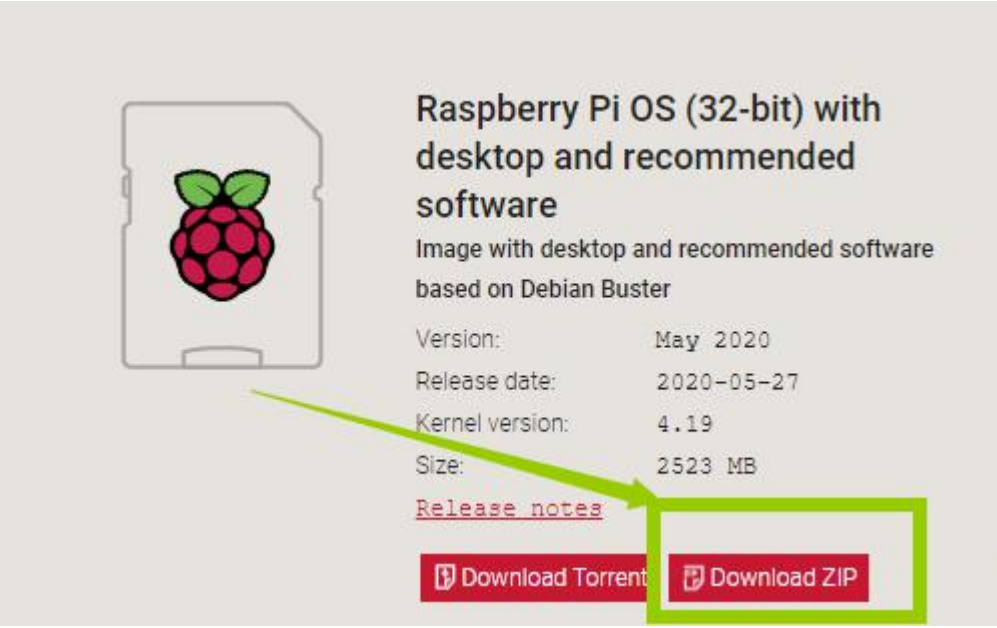
After logging in to the official website, click on the location as shown below:



(2) We need to find out the Raspberry Pi OS (32-bit) with desktop and recommended software. It contains a complete desktop system and recommended software packages.



(3) Choose to download the ".ZIP" file and wait for the download to complete:



(4) Find the ".ZIP" file you just downloaded, double-click to open it, and extract it. The uncompressed file format of the file is ".img". Pay attention, you must name the path of the uncompressed .img file all English letters without special characters.

•

2.2.2 Method two: Manually downloading the image file we provide and write it to the SD card (not recommended).

The image downloaded according to 2.1.1and 2.1.2 of method one is the latest official version of Raspbian and comes with some pre-installed software. At the same time, the normal operation of the robot product requires many other dependent libraries, although we provide a script to install these simple methods of relying on libraries (will be introduced in detail later), occasionally encountering dependent library updates may cause the installation of dependent libraries to fail, so we provide a Raspbian mirror file pre-installed with dependent libraries. The disadvantage of this method is that the mirror

files and related dependent libraries we provide cannot be kept updated at any time. Only when you encounter a very difficult problem, you can try this method to solve it.Download the Raspbian image file address we provide:

https://www.adeept.com/learn/detail-49.html

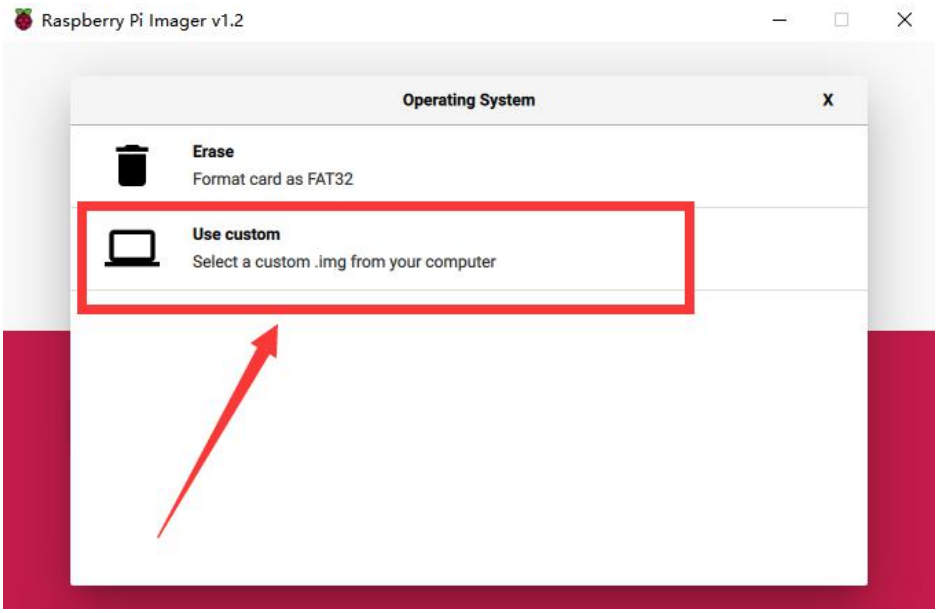
After the download is complete, decompress it. The path of the decompressed .img file must be all English letters and no special characters.

2.3 Burning the downloaded Raspberry Pi system to the SD card

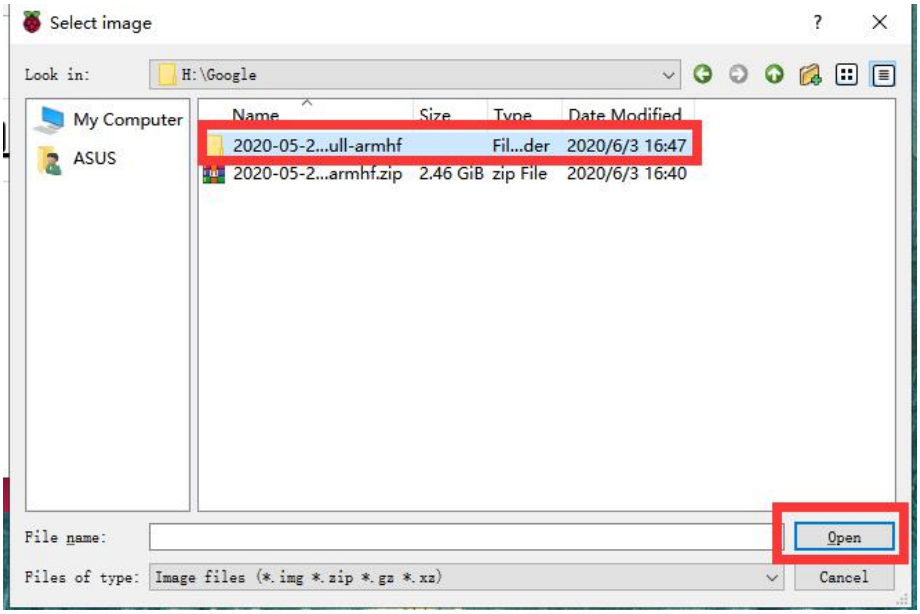
Use Raspberry Pi Imager to burn Raspberry Pi system to SD card.

(1) Click "CHOOSE OS" on the opened Raspberry Pi Imager software interface.

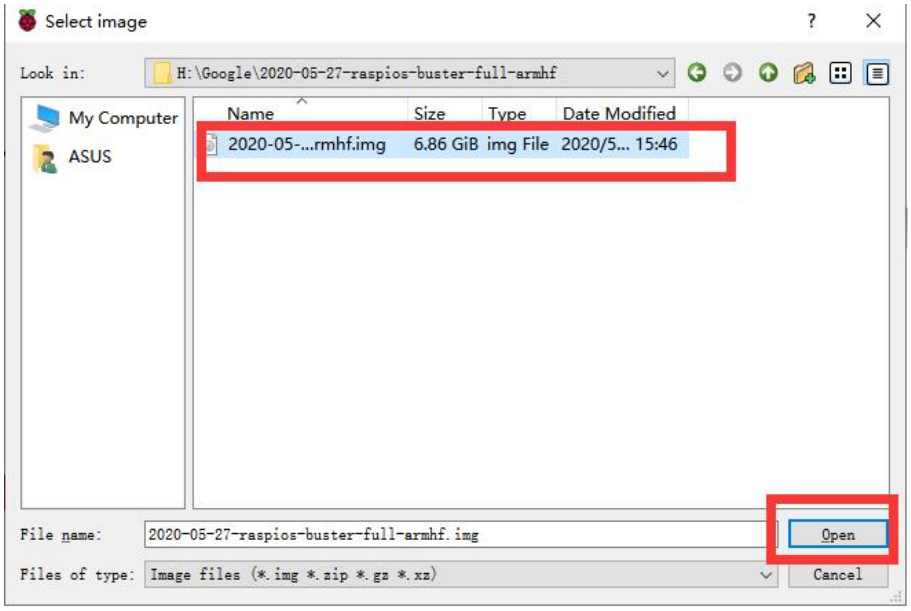
(2) Click "Use custom" and select a custom ".img" file from your computer, which is the ".img" file of the Raspberry Pi system that we downloaded and decompressed before.



(3) Select the ".img" file and click "Open".

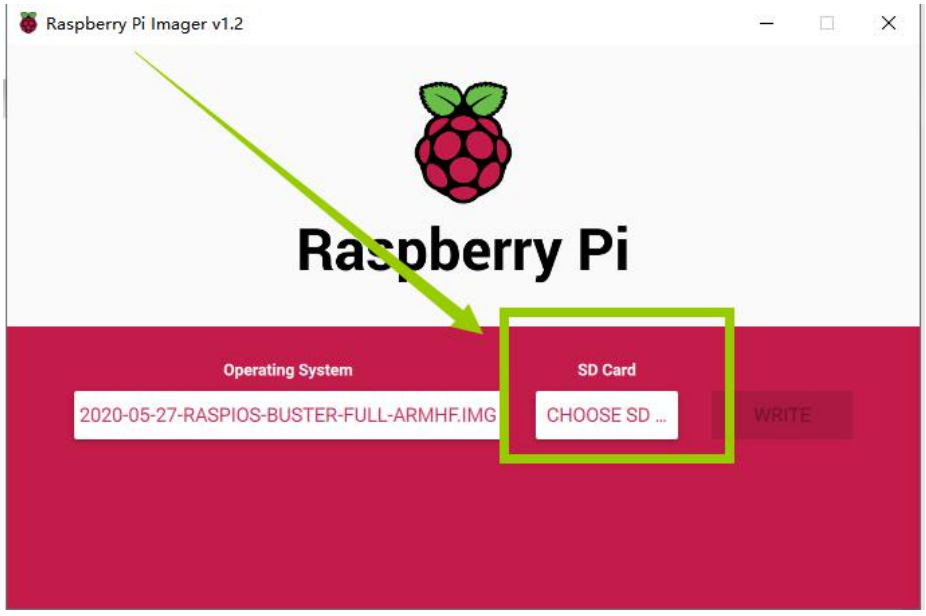


(4) Select the ".img" file and click "Open".

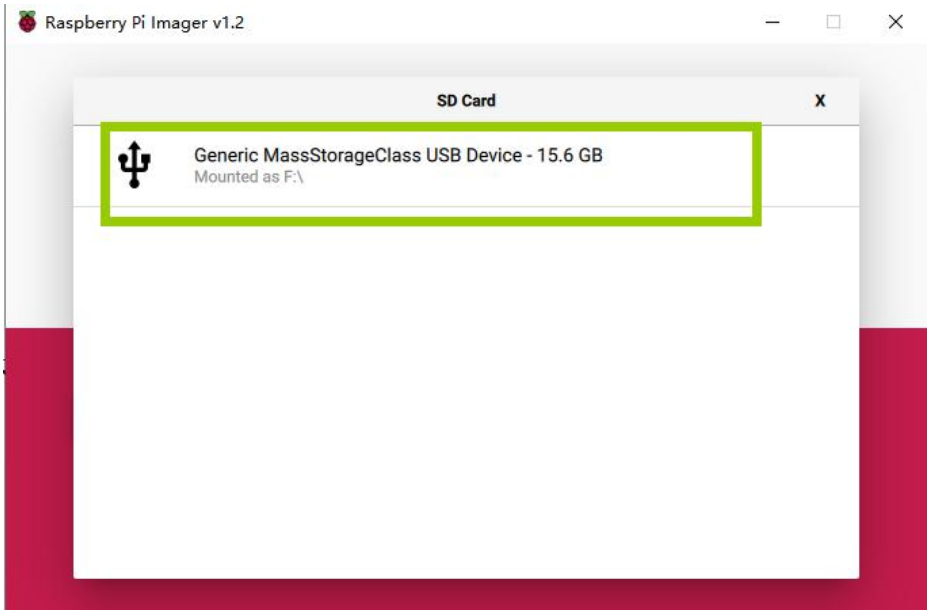


(5) Then on the interface of Raspberry Pi Imager, the ".img" file of our selected Raspberry Pi system will appear.

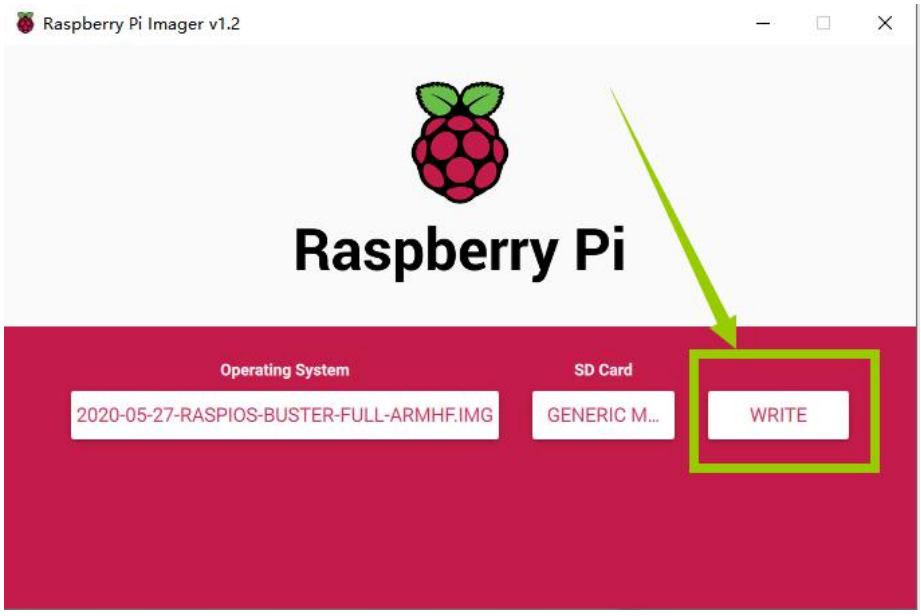
(6) Click "CHOOSE SD".



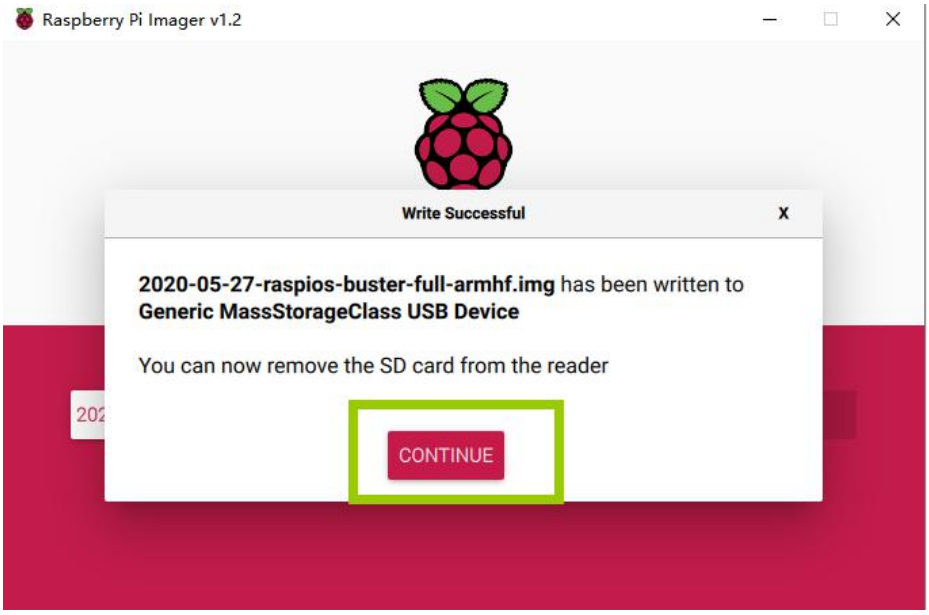
(7) Then select the SD card we need to burn.



(8) Click "WRITE" to write it to the SD card. Wait for the burn to complete.



(9) After the burning is completed, the following message will be prompted,indicating that the burning is finished, click "CONTINUE".



【Pay Attention】

Don't remove the SD card after burning! After the Raspberry Pi Imager is burned, the memory card will be ejected in the program. This will cause the subsequent copy

operation to prompt that the SD card has not been found. You can unplug the card reader from the computer and then plug it into the computer again.It is necessary to configure SSH and WIFI connection later. At this time, once the SD card is put into the Raspberry Pi to boot, it may cause subsequent headless WIFI configuration failure.

2.4 Starting the Raspberry Pi SSH service

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)

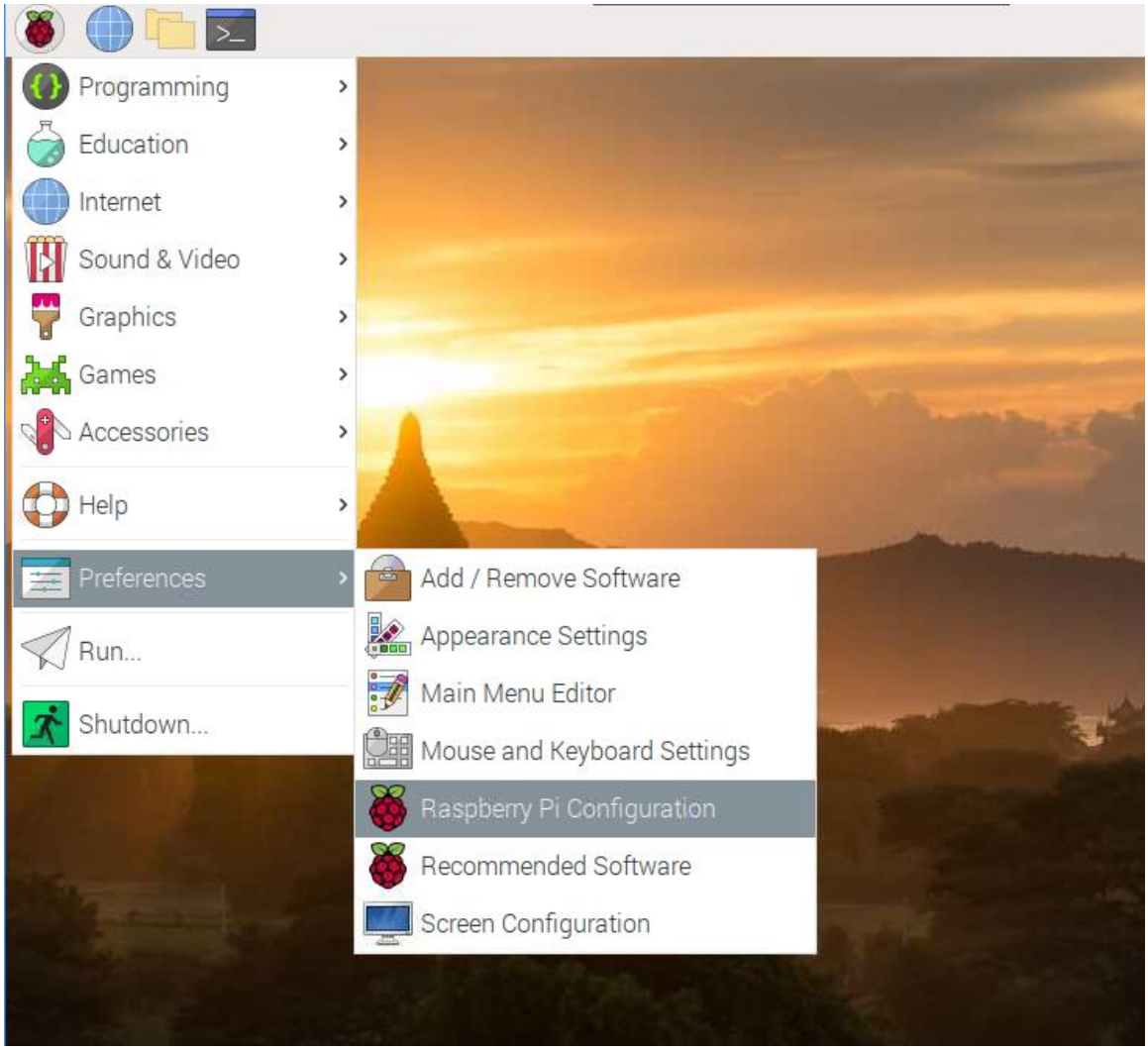
2.4.1 Method A: Enable SSH with peripherals

If you **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.

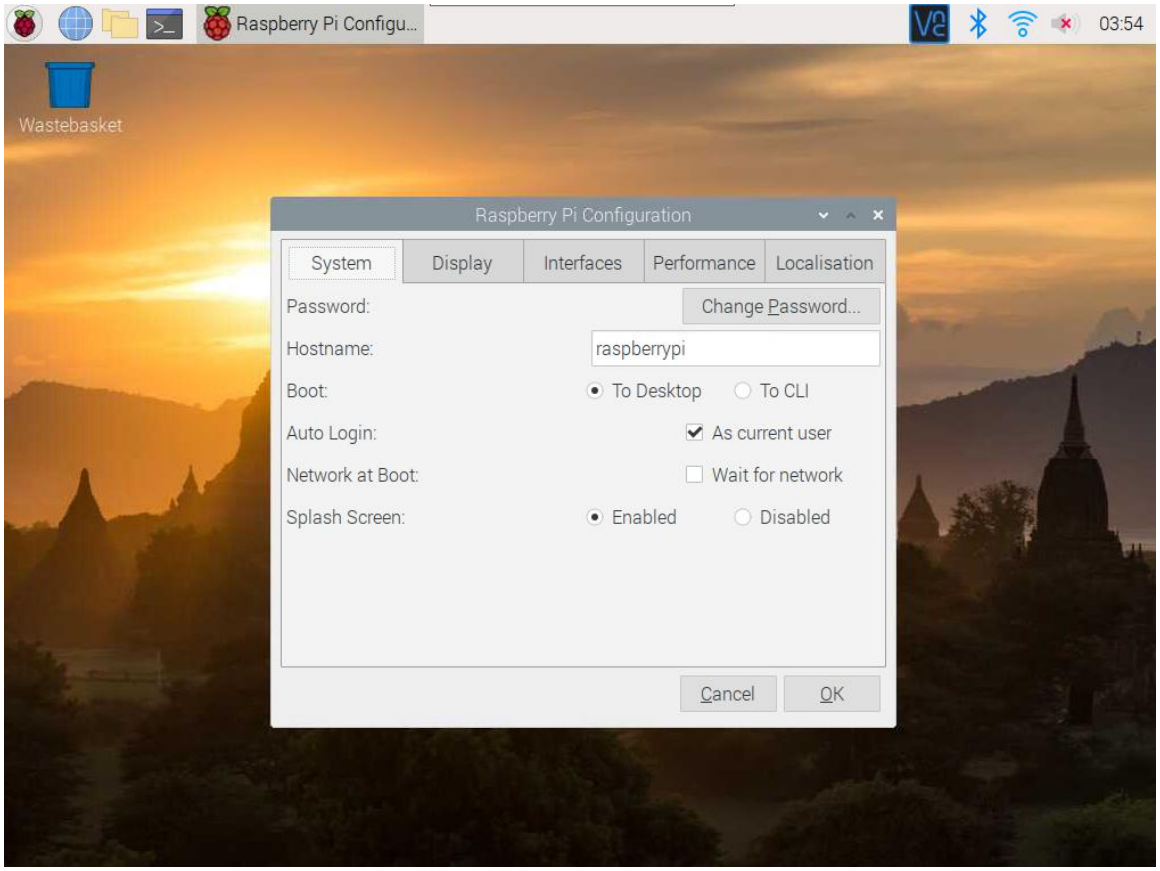
If you've connected a mouse, keyboard, or monitor to the Raspberry Pi, follow these steps to enable SSH.

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.

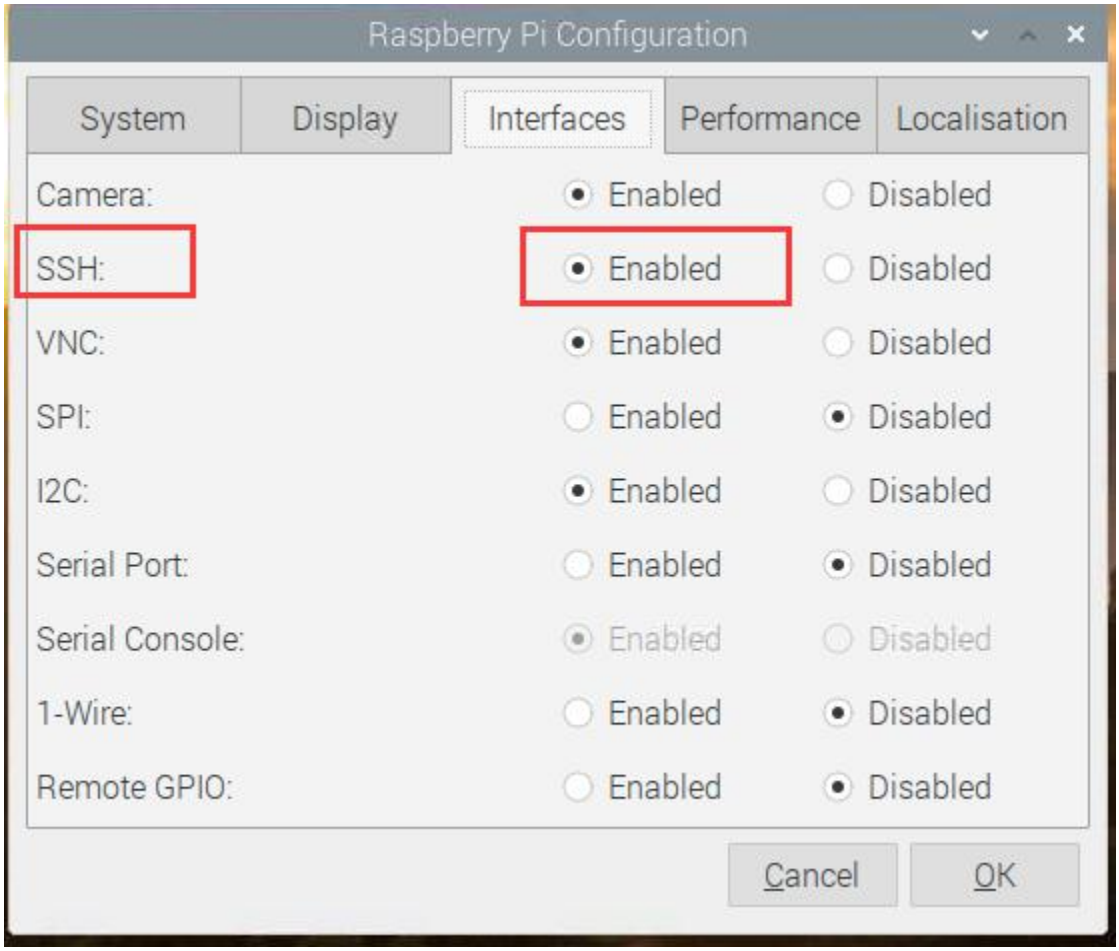
2.Go to Preferences menu, select Raspberry Pi Configuration.



3. Enter the Interfaces tab.



4. Select Enable next to SSH.



5.Click OK.

2.4.2 Method B: 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.

2.5 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]

2.5.1 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.

2.5.2 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.)

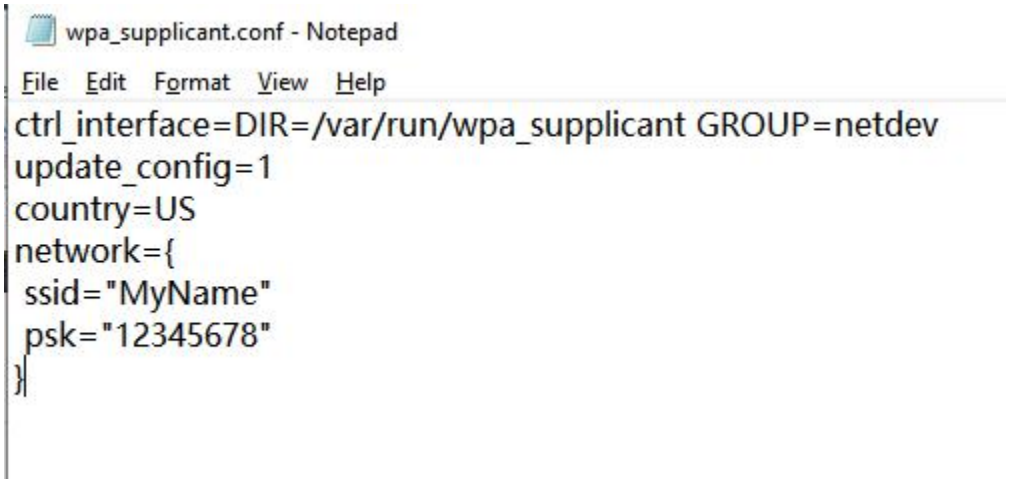
2. Create a file named `wpa_supplicant.conf` anywhere in your computer.

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"
}
```



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

2. If you've alreay copied the file `ssh` to the SD card as instructed in **2.4**, 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]

2.6 Remotely logging in to the Raspberry Pi system

Before using SSH to connect to the Raspberry Pi, you need to know the IP address of the Raspberry Pi and the software that supports SSH.

The remote login to the Raspberry Pi is realized through the SSH protocol, and the Raspberry Pi can be remotely logged in through the software with the SSH protocol. For example: putty, MobaXterm, etc.

Windows 10, LInux and Mac OS come with SSH function, and you can also log in to the Raspberry Pi remotely through the terminal.

2.6.1 Obtaining the IP address of the Raspberry Pi

2.6.1.1 Obtaining an IP address with an external display

We provide a simple and fast way to get the Raspberry Pi IP address. You need to prepare the following components:

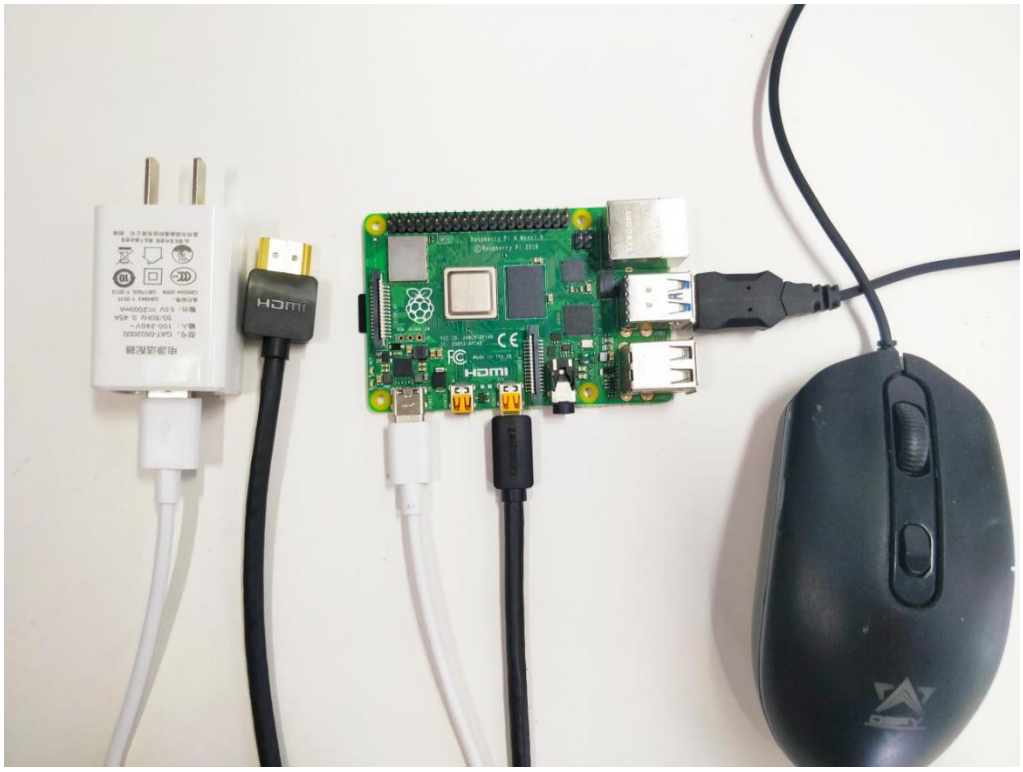
(1) One Type-C data cable: used to supply power to the Raspberry Pi.

(2) One HDMI cable: used to connect the monitor.

(3) One mouse: used to operate.

(4) One monitor

(5) One Raspberry Pi



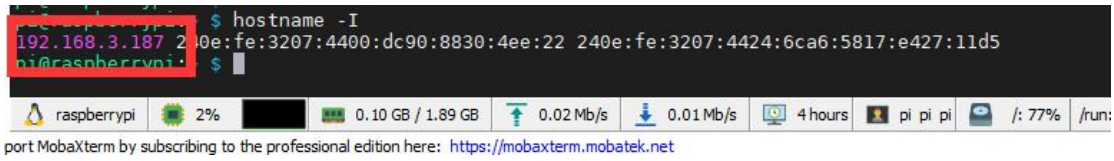
Connect the HDMI cable to the HDMI port of the monitor:

1.Turn on the monitor switch, and connect the mouse to the USB port of the Raspberry Pi, supply power to the Raspberry Pi with the Type-C data cable, then the Raspberry Pi starts. After entering the system interface, we move the mouse cursor to the " 

" in the upper right corner, then it will display the IP address of the Raspberry Pi: 192.168.3.157 (the IP address of each Raspberry Pi is different). It is necessary for you to record this IP address for it is needed to log in to the Raspberry Pi system later.

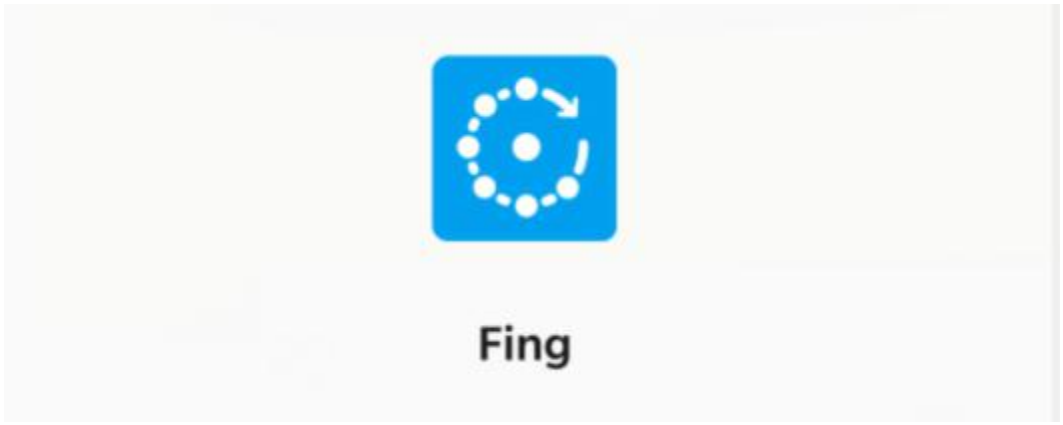
2. You can also check the following IP address by opening the command window of the Raspberry Pi and entering the following command, you need to write it down:

hostname -I

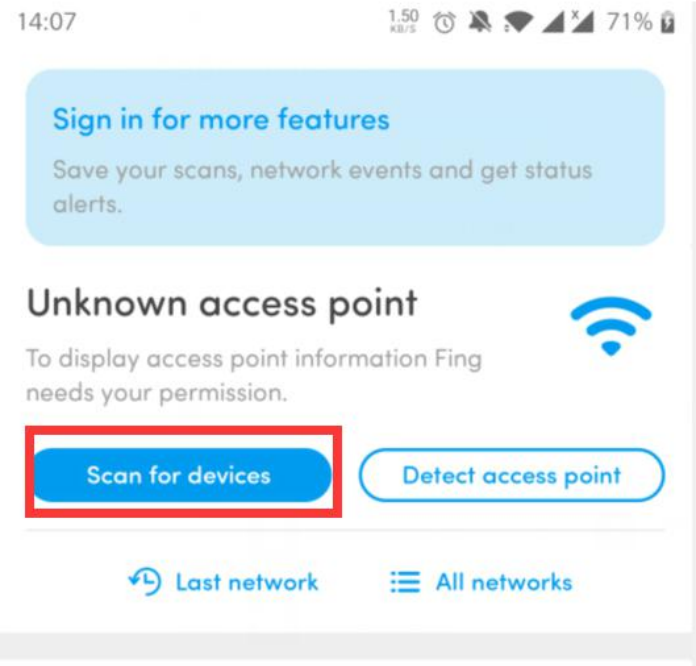


2.6.1.2 Obtaining an IP address with a mobile phone

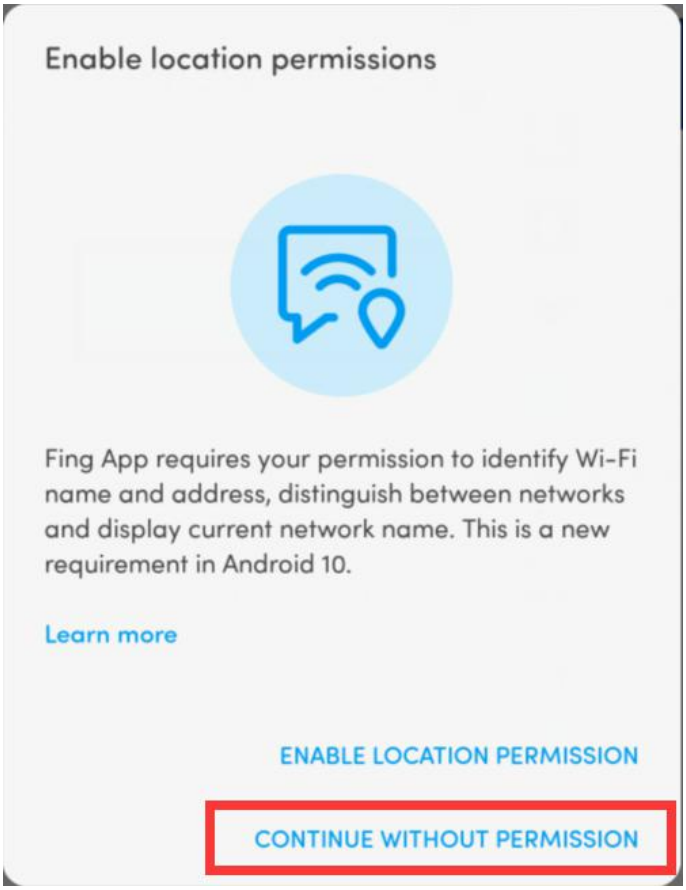
1. You need to download an APP called "Fing" on your phone, as shown below:



2. After the download is complete, your phone and Raspberry Pi need to be in the same local area network, that is, your phone and Raspberry Pi are connected to the same WIFI, then open "Fing" and click "Scan for devices":

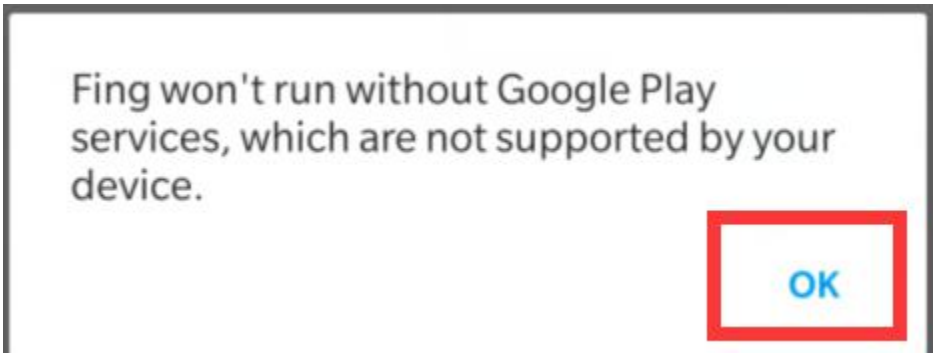


Click“CONTINUE WITHOUT PERMISSION”:

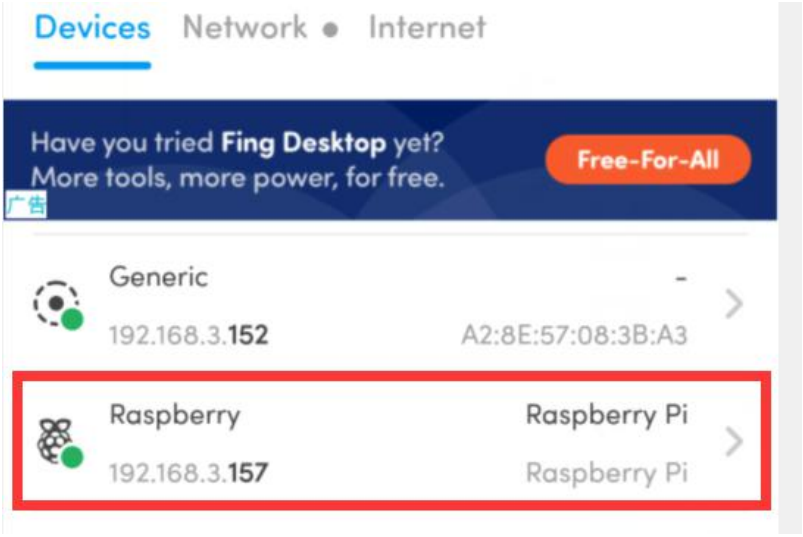

Enable location permissions
Fing App requires your permission to identify Wi-Fi name and address, distinguish between networks and display current network name. This is a new requirement in Android 10.
Learn more
ENABLE LOCATION PERMISSION
CONTINUE WITHOUT PERMISSION

0

Click OK：


Fing won't run without Google Play services, which are not supported by your device.
OK

3. Wait for the scan to complete. In the list, you find a device named "Raspberry Pi". In the lower left corner, you will see the IP address of the Raspberry Pi: 192.168.3.157. You need to write down this IP address.



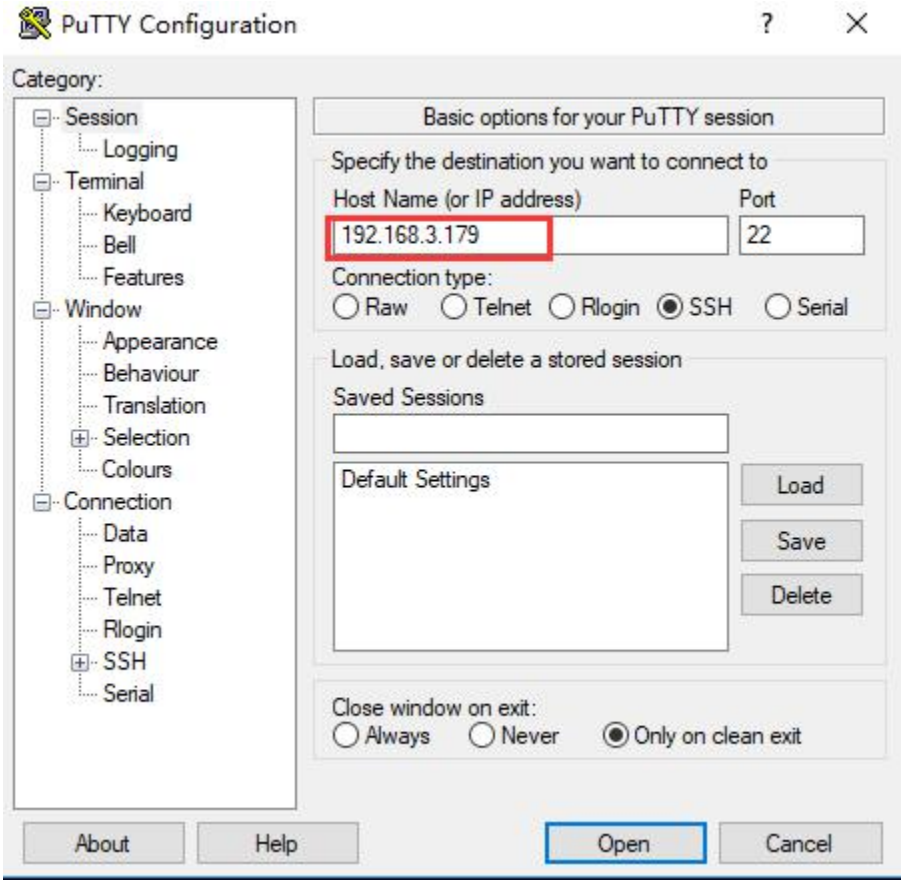
2.6.2 Remotely logging in to the Raspberry Pi system

This course recommends two kinds of software for SSH login to Raspberry Pi. In actual use, you only need to download one. LInux or Mac OS comes with SSH function, you can log in to the Raspberry Pi remotely with the terminal without downloading software.

2.6.2.1 Putty

You need to download and install PuTTY corresponding to your computer system version, and use it to log in to the Raspberry Pi. PuTTY download address: https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html

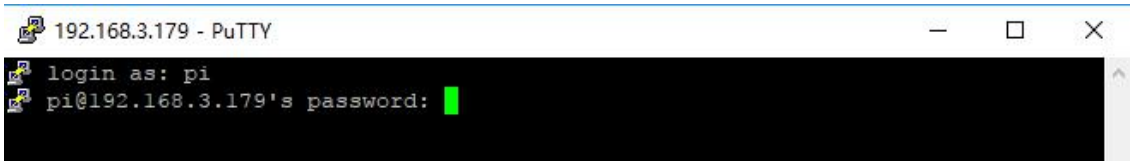
Run PuTTY, enter the IP address of the Raspberry Pi into the Host Name, and click Open.



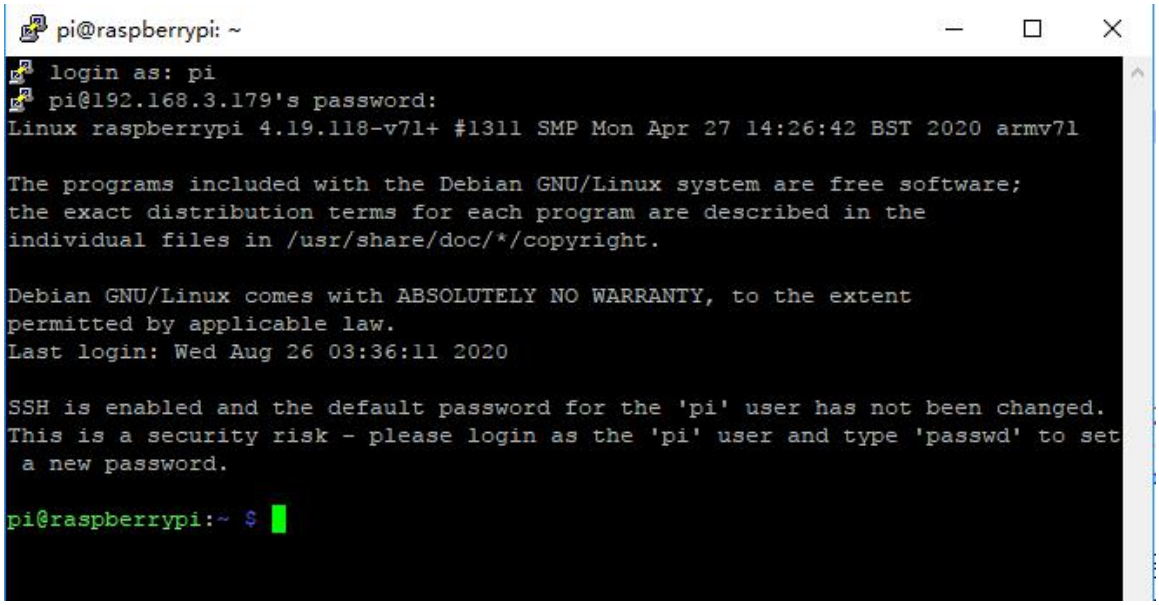
• If it prompts Network error: Connection timed out, it means you probably entered the wrong IP address.

• When the connection is normal, you will see a security warning. You can safely ignore it and click the "Yes" button. You will see this warning when PuTTY connects to a Raspberry Pi that has not been connected before.

• You will now see the usual login prompt. Log in with the same username and password as the Pi itself. The default login name of Raspbian is pi and the password is raspberry. When entering the password, the screen will not display the entered password. After entering raspberry, press Enter to confirm.



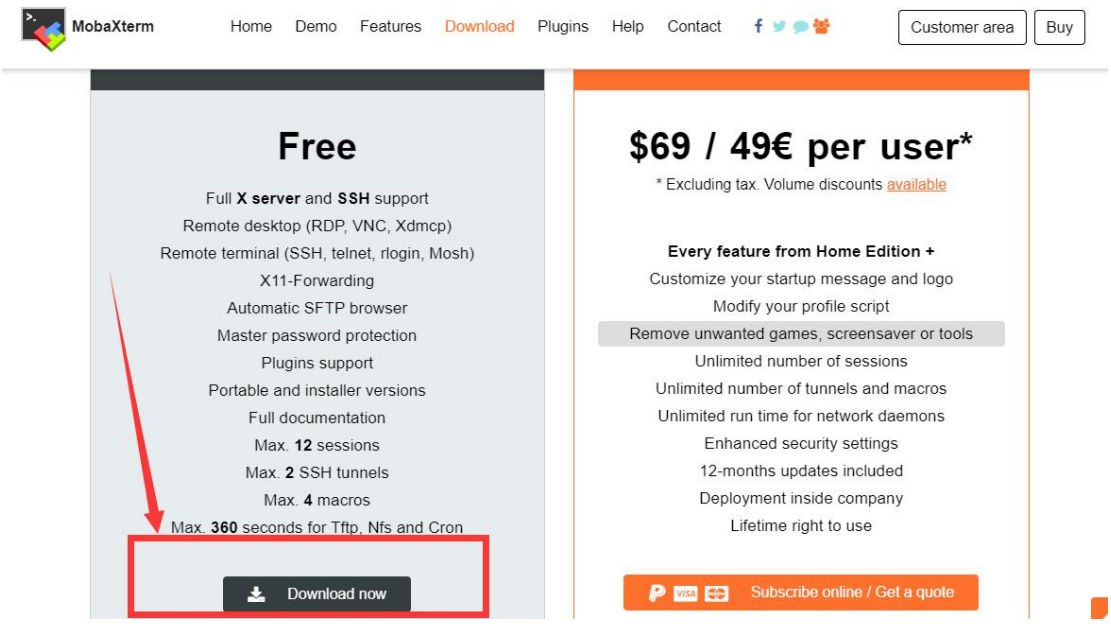
• You should now have the Raspberry Pi prompt, which will be the same as the prompt on the Raspberry Pi itself.



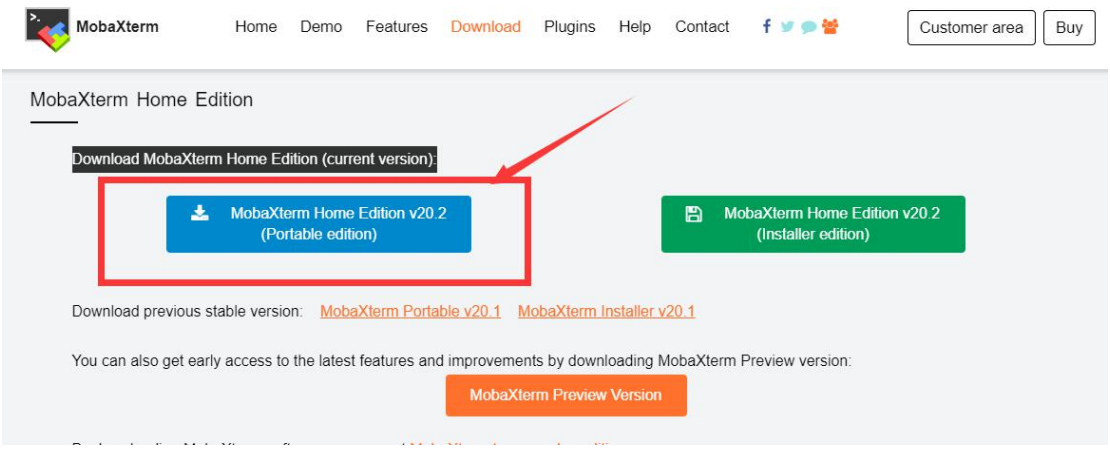
2.6.2.2 MobaXterm

MobaXterm is a terminal tool software that can be used to remotely control the Raspberry Pi.

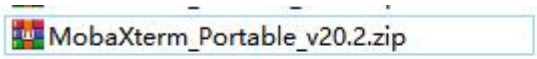
(1) Log in to the official website with a browser to download: https://mobaxterm.mobatek.net/download.html. Choose the Free version to download.

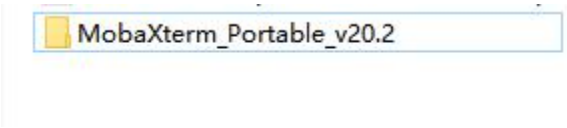


(2) Download the Portable edition of MobaXterm Home Edition (current version):



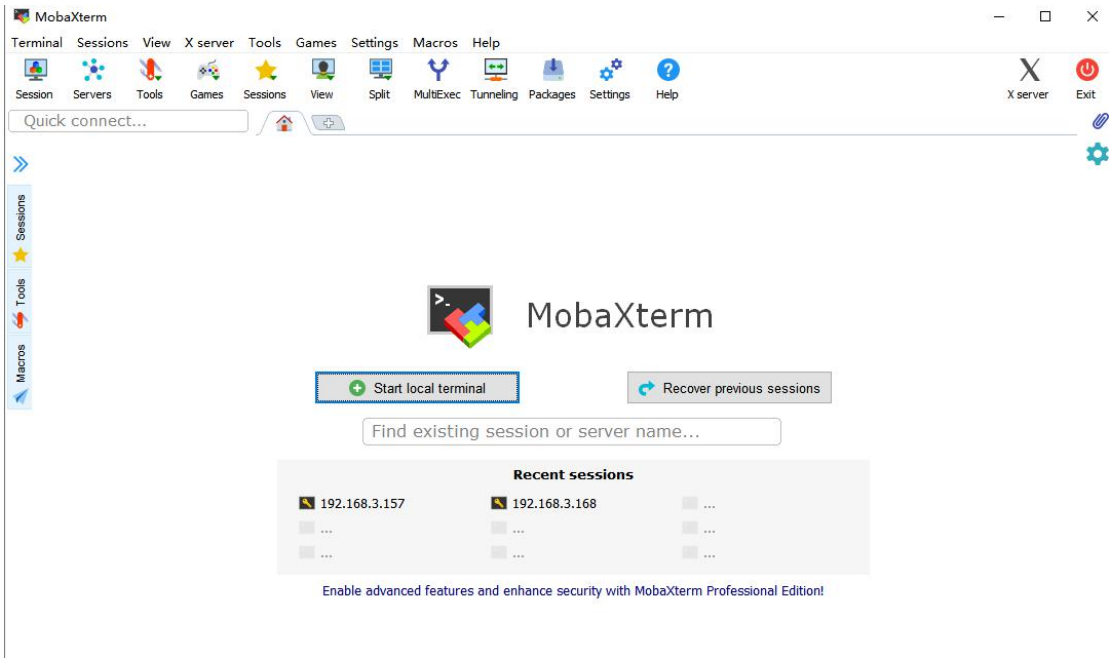
(3) Find the downloaded file MobaXterm_Portable_v20.2.zip, double-click to open it, unzip it to get a new file.



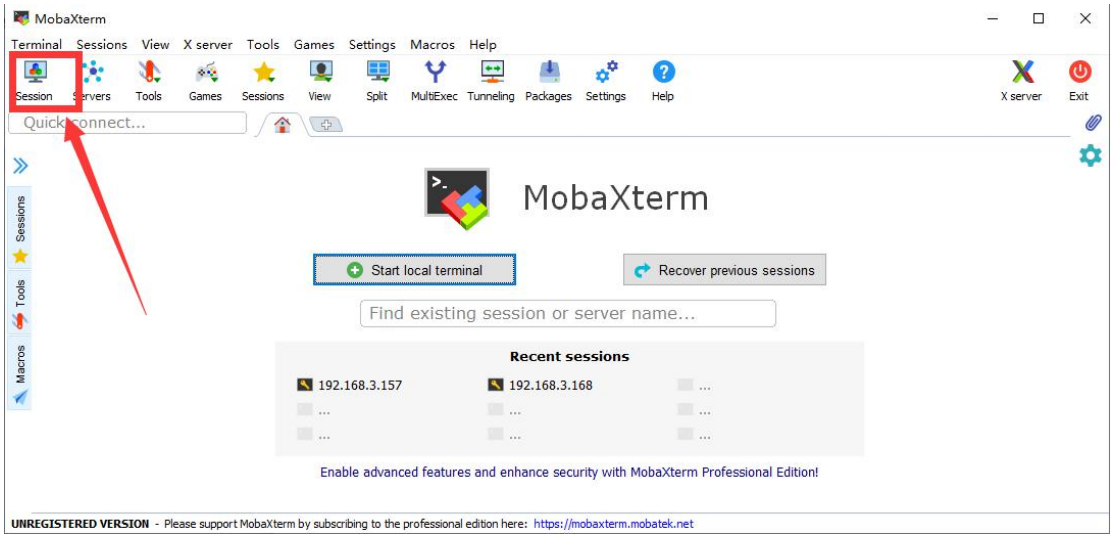


(4) Open the unzipped folder, there is a file MobaXterm_Personal_20.2.exe inside.

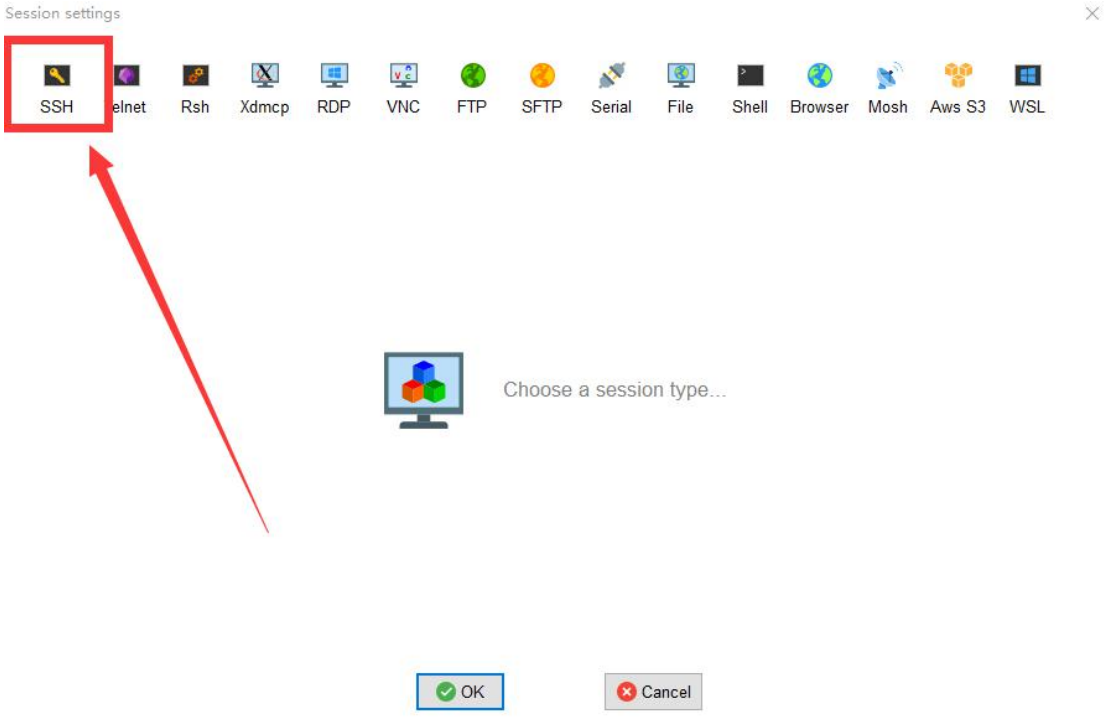
(5) Double-click to open MobaXterm_Personal_20.2.exe, and then directly open the MobaXterm software. The interface is as follows:



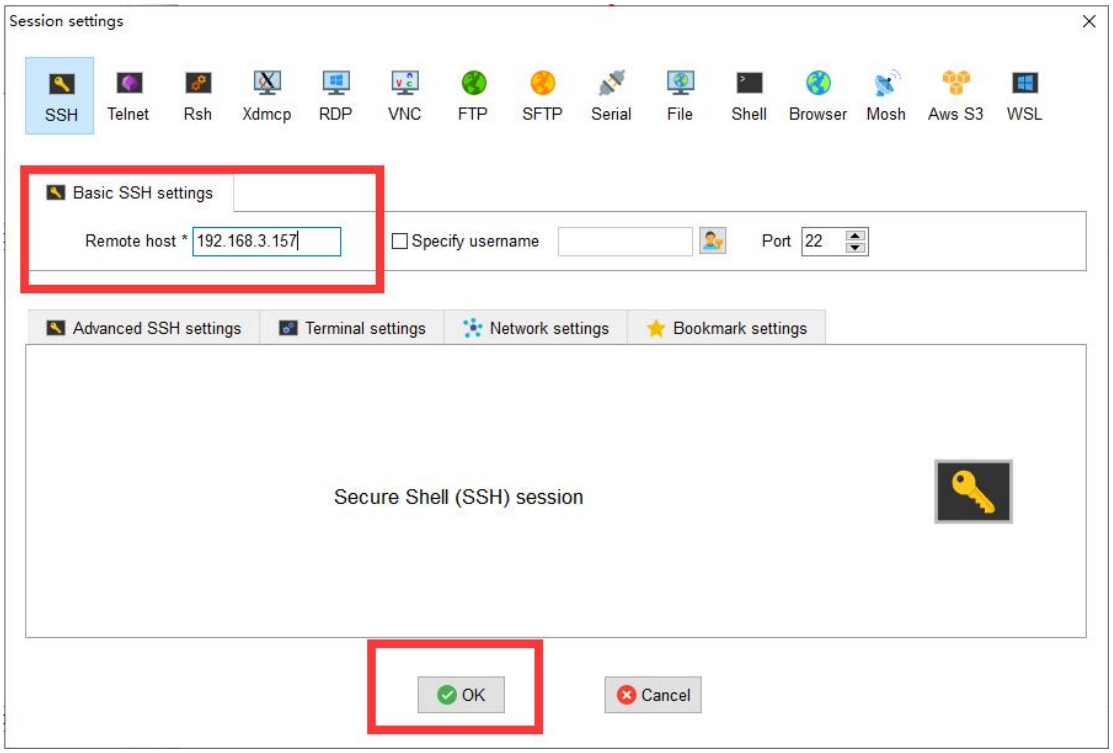
(6) Click "Session" in the upper left corner.



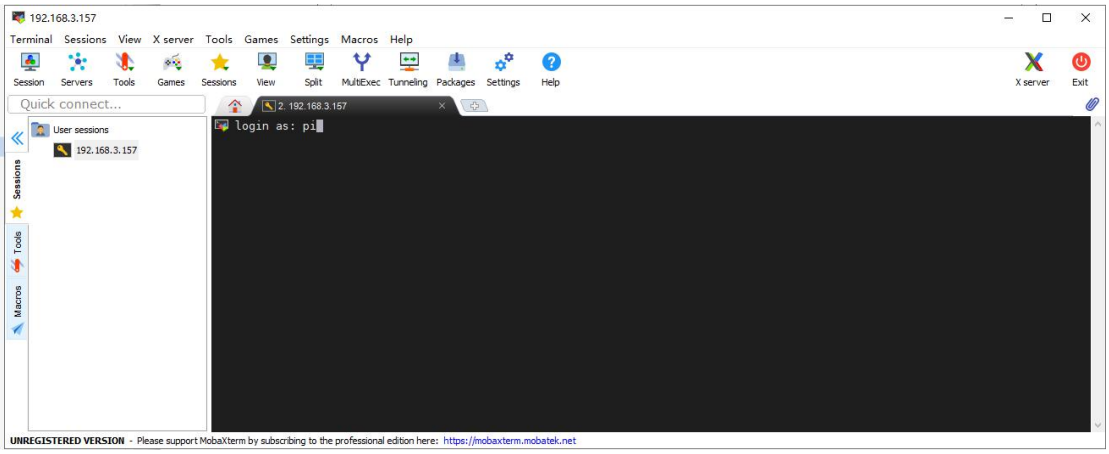
(7) Click "SSH".



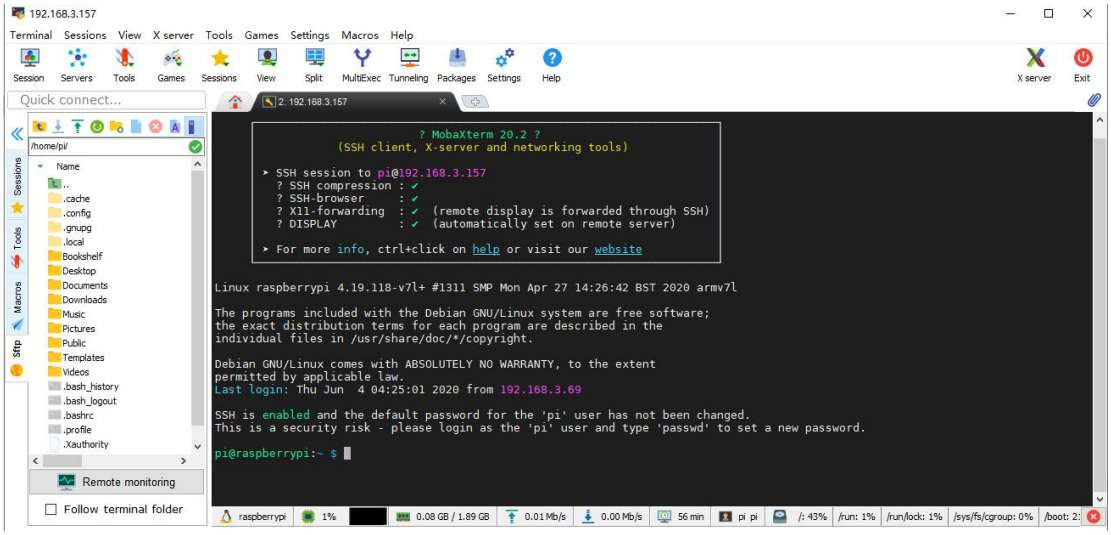
(8) Enter the IP address of the Raspberry Pi queried before: 192.168.3.157, and click "OK" to confirm.



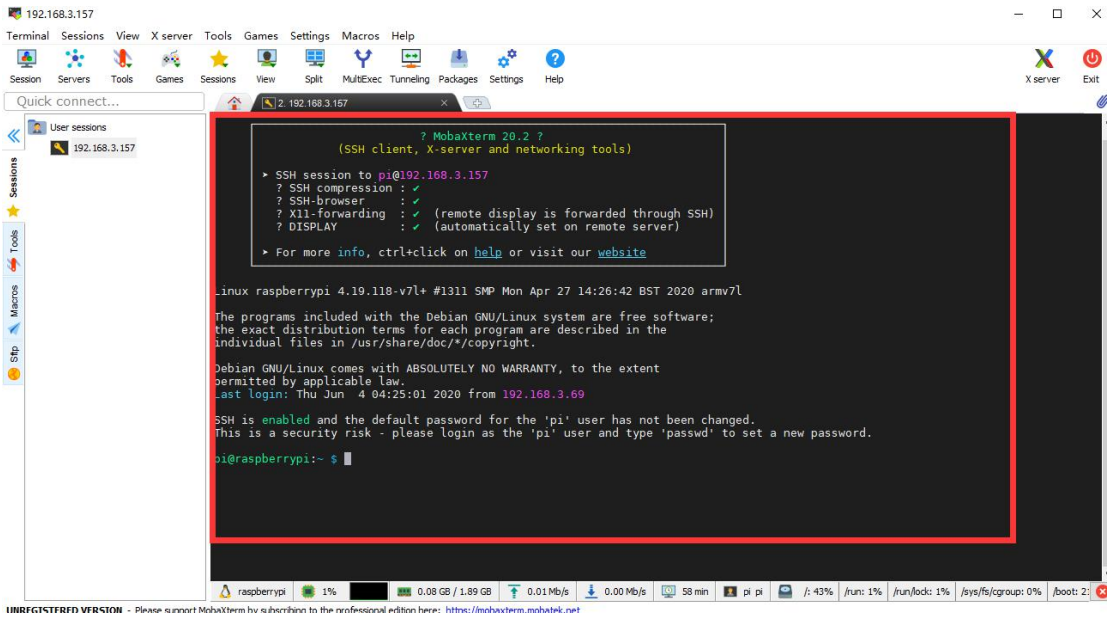
(9) Enter the Raspberry Pi default account: pi, then press the Enter key, and then enter the Raspberry Pi default password: raspberry. Press Enter to log in to the Raspberry Pi system.



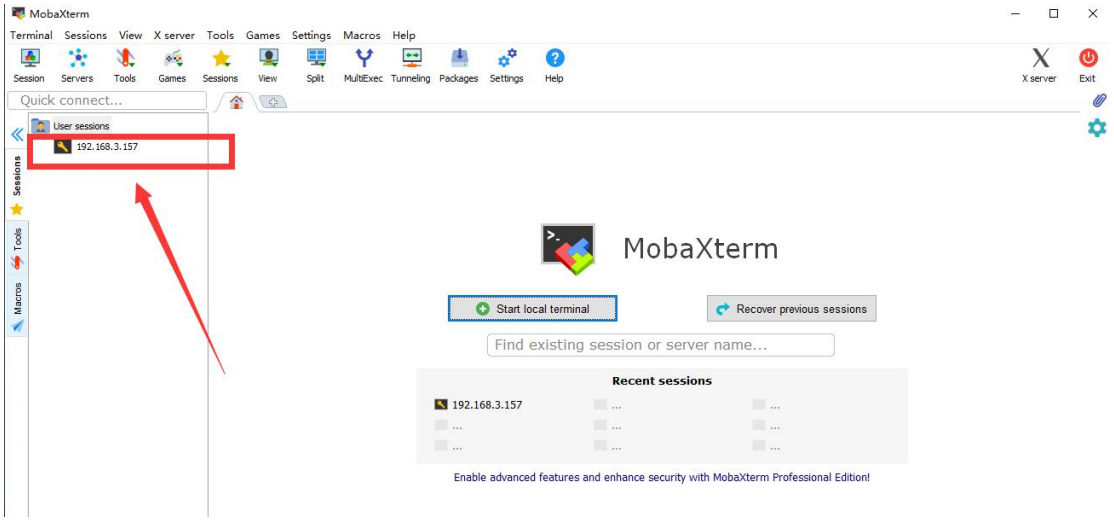
(10) After successfully logging in to the Raspberry Pi system, the following interface will appear:



(11) The red box in the figure below is the command window, where you can control the Raspberry Pi by entering commands.



(12) When we close the MobaXterm software and open MobaXterm again to connect to the Raspberry Pi, we can double-click the IP address under "User sessions" on the left: 192.168.3.157, enter the account name: pi, and you can directly connect to the Raspberry Pi.



2.6.2.3 Windows10, LInux and Mac OS comes with SSH function

Steps to connect to Raspberry Pi via SSH:

1. Open a console terminal window.

2. The initial user name of the Raspberry Pi is pi and the initial password is raspberry.

3. Enter ssh pi@<IP> in the command line and replace <IP> with your Raspberry Pi IP address, as shown in the following example:

ssh pi@192.168.3.157

4. Press Enter, and the prompt Are you sure you want to continue connecting (yes/no)?

5. Enter yes, press Enter, pi@192.168.3.157's password: appears, fill in the initial password raspberry of the Raspberry Pi, pay attention to the case, there will be no changes on the screen during the password input, but it does not Indicates that the input was not successful, press enter after the input is complete.

6. Now you have logged in to the Raspberry Pi.

2.7 Downloading the Raspberry Pi robot product program

• For the power supply of Raspberry Pi, please refer to this official document Power supply.

• Our Raspberry Pi robot driver board Robot HAT can directly supply power to the Raspberry Pi through the GPIO pins. However, because the software installation time in the Raspberry Pi is relatively long, it is not recommended to use battery power when installing the Raspberry Pi. The Raspberry Pi robot driver board Robot HAT or camera need not be installed when installing the software in the Raspberry Pi. This does not affect the software installation, but when you run the installed program, you must connect the driver board and the Raspberry Pi camera, otherwise Will cause the program to report an error.

• If you manually download the image file provided by us, you only need to load the SD card into the Raspberry Pi to boot, and the program of the robot product will run automatically. You can skip content 2.7.

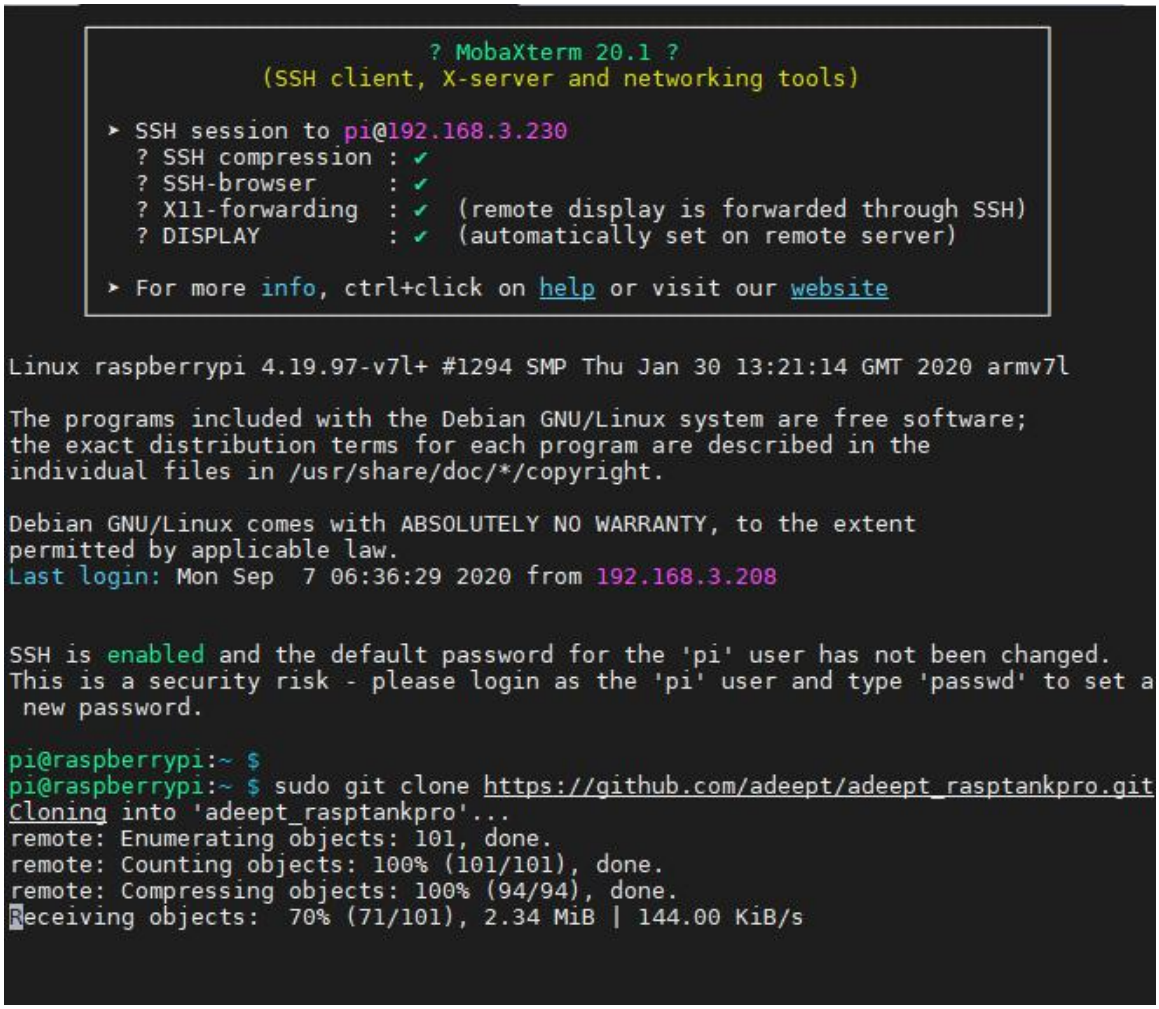
2.7.1 Downloading the Raspberry Pi robot product program

• All the code of our product has been open sourced on GitHub, you need to download it to the Raspberry Pi and install the relevant dependencies before it can run normally. github address: https://github.com/adeept/adeept_rasptankpro

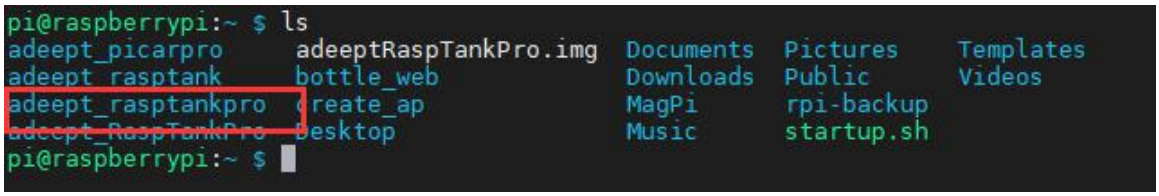
1. After the operations in the previous section, we have logged in to the Raspberry Pi, and enter the following commands in the console:

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

2. After the input is complete, press enter to start downloading the robot program from GitHub. This process will continue for a period of time. Wait patiently for the download to complete.



3. After the download is complete, a new folder, adept_rasptankpro, will appear, in which the product code is stored. Check through the Linux command "ls".



2.7.2 Installing the dependency library of the robot program

• If you used 2.2.1 to manually download the Raspbian image file and write it to the SD card to write the image file to the SD card, you can refer to this section to install the dependent libraries.

•We have prepared a script to install all the dependent libraries that need to be used and set up operations such as turning on the camera and automatically running on startup.

•Enter the following code in the console and run the script setup.py to install the required dependent libraries.

sudo python3 adeept_rasptankpro/setup.py

```
pi@raspberrypi:~ $ sudo python3 adeept_rasptankpro/setup.py
```

• Press enter, the next operation is automatically completed by the script program, this process may last for tens of minutes or several hours depending on the network environment, just wait patiently.

• After the installation is complete, the console will display text:

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 (Robot 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.

```
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 (R
obot HAT).
After turning on again, the Raspberry Pi will automatically run the program to s
et the servos port signal to turn the servos to the middle position, which is co
nvenient for mechanical assembly.
restarting...
```

•After the installation is complete, the Raspberry Pi will automatically disconnect the SSH connection and restart. At this time, if you are using a Raspberry Pi connected by software such as Putty, there will be an error message such as Network error: Software caused connection abort, which is normal, just close it.

3. Running the Program and WEB Control Interface

●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.

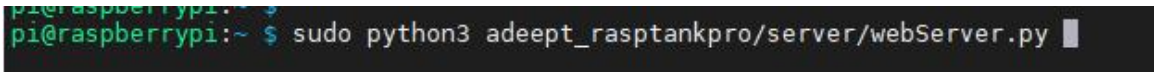
●How to tell whether the robot has run the `webServer.py` or not: If the WS2812-LED lights up with the breathing effect, it means the robot has booted and runs the program automatically.

●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

●Use the following command to run webServer.py

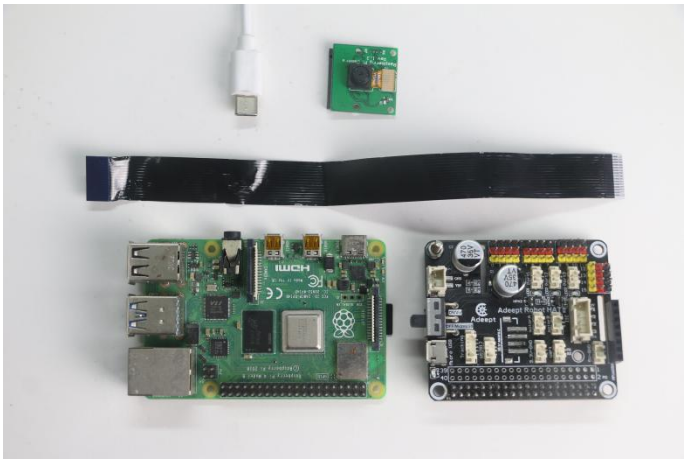
sudo python3 adeept_rasptankpro/server/webServer.py



3.1 Running the Raspberry Pi robot program

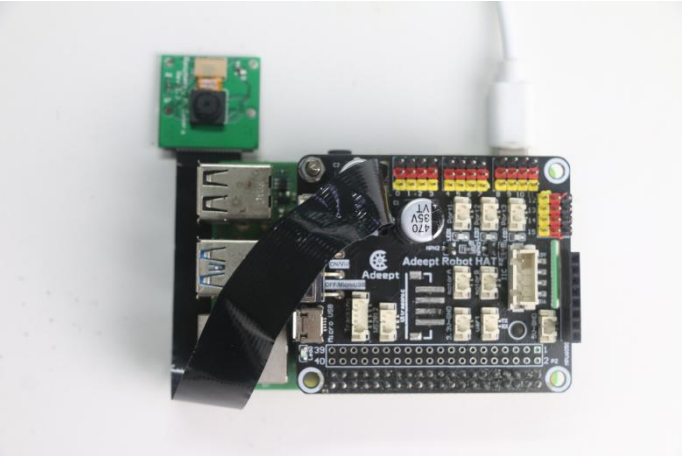
Only by successfully running the webServer.py program on the Raspberry Pi, can you access the Raspberry Pi via IP on the browser (after installing the dependent libraries, the Raspberry Pi will automatically run webServer.py).

Prepare the components required for installation, and turn off the power of the Raspberry Pi during installation.



Install the camera cable, contact the metal surface of the cable with the metal surface of the Raspberry Pi (the same is true for installing the camera).

Install Robot HAT and camera, and connect the Raspberry Pi power supply.



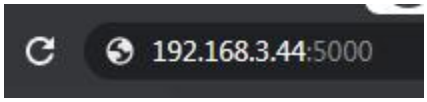
After the Raspberry Pi is turned on (about 30-50s), you can access the Raspberry Pi through a browser.

3.2 Introduction to web control interface functions

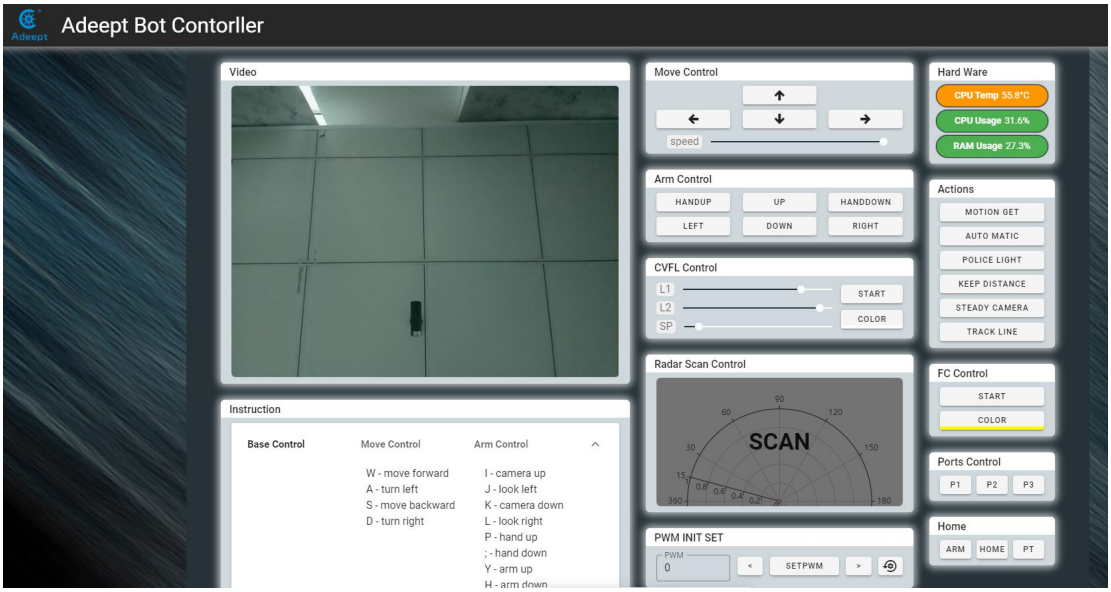
1. Make sure your device is in the same local area network as the Raspberry Pi.

2. Obtain the IP address of the Raspberry Pi (refer to the software installation section).

3. Open the browser on the device (chrome browser is recommended to avoid possible browser compatibility issues), enter the IP address of your Raspberry Pi in the address bar, and visit port 5000, for example: 192.168.3.44 :5000



The web controller will then be loaded into the browser.



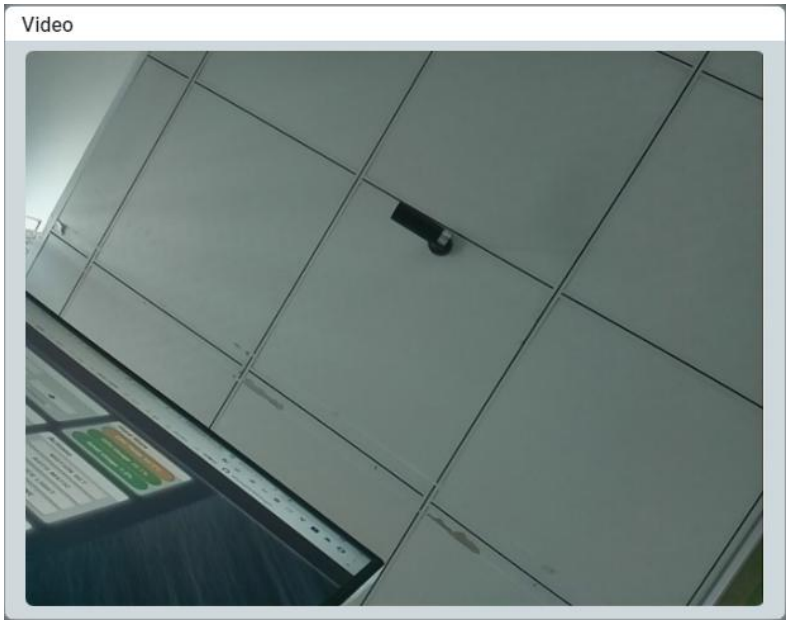
Depending on the product, the modules on the web controller are also different. Below is a list of most of the modules and their usage. You can compare the modules displayed in your browser to understand their functions and usage.

3.2.1 Basic module

Basic modules can be found in almost all products. These modules are used to control the core functions of robot products. At this time, only the camera can work, and other functions can only be realized after RaspTankPro is installed.

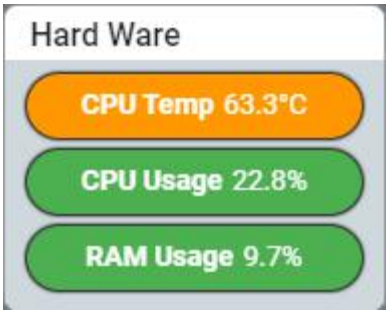
1. Vedio module

Show the screen 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.



2. Hard wave module

Display the hardware information of the robot product.



CPU Temp: Display the Raspberry Pi CPU temperature;

CPU Usage: Display the CPU occupancy rate of the Raspberry Pi;

RAM Usage: Display the RAM usage of Raspberry Pi

3. Move Control Module

Control the movement of the robot product back and forth.

speed: Use the slider to control the speed when the robot moves.

4. Arm Control Module

Control PTZ and robotic arm. Depending on the product, the number of buttons and manipulation methods are also different.

UP: The camera angle of view moves upward

DOWN: the camera angle of view moves downward

GRAB: clamping chuck

LOOSE: Loosen the chuck

LEFT: The robotic arm wants to turn left

RIGHT: The robotic arm turns to the right

HANDDOWN: The robotic arm rotates downward

HANDUP: The robotic arm rotates upward

3.2.2 Advanced Function Module

Advanced function modules refer to those modules used to perform advanced functions of robotic products. Function button group. Used to switch the function of the robot.

1. Actions Module

MOTION GET: switch watchdog mode. In this mode, the robot product stops moving and reacts to the moving objects detected in the camera. The moving objects are framed in the video of the Vedio module.

AUTO MATIC: Switch automatic obstacle avoidance mode. In this mode, the robot product will automatically advance and use the ultrasonic module to detect obstacles. When encountering obstacles, try to find other ways.

POLICE LIGHT: switch the police light mode. In this mode, the LED lights of the robot product will flash like a warning light.

STEADY CAMERA: Switch camera stabilization mode. In this mode, the robot product will try to maintain the vertical stability of the camera.

TRACK LINE: switch hunting mode. In this mode, the robot product will try to move along the white lines on the black ground.

2. FC Control Module

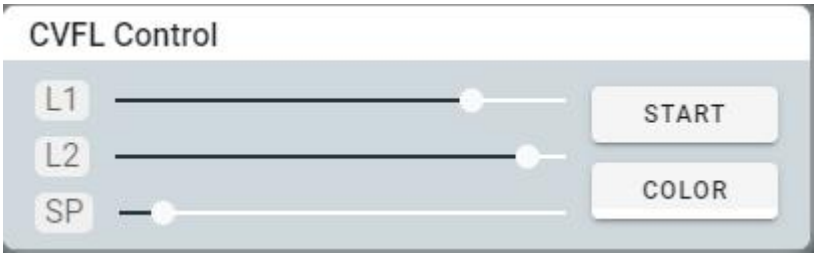
Control the switch of the color tracking function and the setting of the color to be tracked.

COLROR: Select the color to be traced.

START: Switch color tracking mode.

3. CVFL Control Module

The switch that controls the visual patrol function.



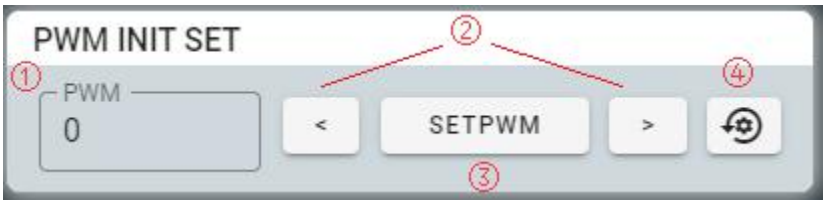
L1, L2: The two parameters of L1 and L2 are used to set the height of the two auxiliary lines. Robot products will only recognize the lines in the screen surrounded by two auxiliary lines.

SP: Deflection tolerance. The larger the value set, the more the robot tends to go straight.

COLOR: Switch to search for white lines on black or black lines on white.

START: Switch the visual tracking function.

4. PWM INIT SET module



It is used to fine-tune the default angle of the servo, which can correct the angle error generated when the servo is installed. When returning to the center, the rudder will return to this default angle.

①Enter the number of the PWM port connected to the servo you want to fine-tune in the PWM input box

②Click these two buttons to make the servo rotate slightly clockwise or counterclockwise

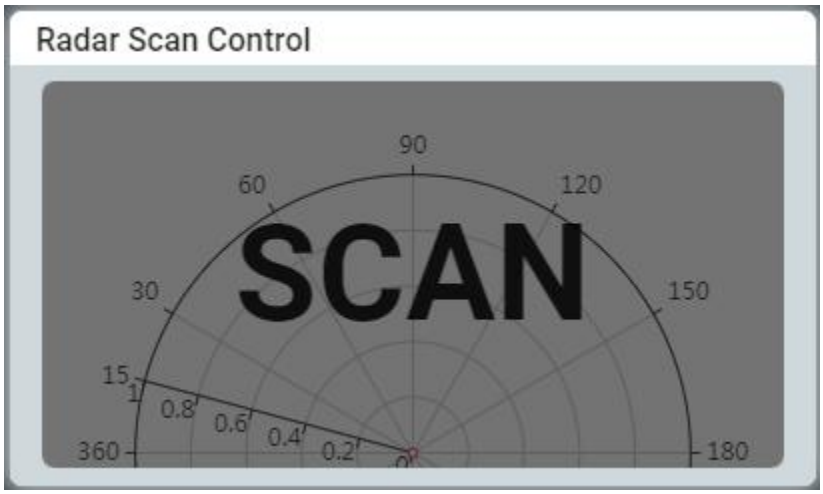
③Click this button to save the current servo angle as the default angle

④ Click this button to initialize the default angle of all servos to factory settings

5. Home Module

HOME: Return all servos to their initial positions.

6. Radar Scan Control Module



Used to perform the ultrasound scan function and display the scan results.

RaspTankPro does not have this function.

7. Ports Control Module

Control the switch of Port1, Port2, Port3 on the development board

4. Setting the Program to Run Automatically After Startup

4.1 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 2.7.2, 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.

4.2 Change the program that starts automatically

●After step 5.**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.

5.How to Edit the Code Program in Raspberry Pi

●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 2.6.2 **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,see 2.6.2.2 for installation tutorial.

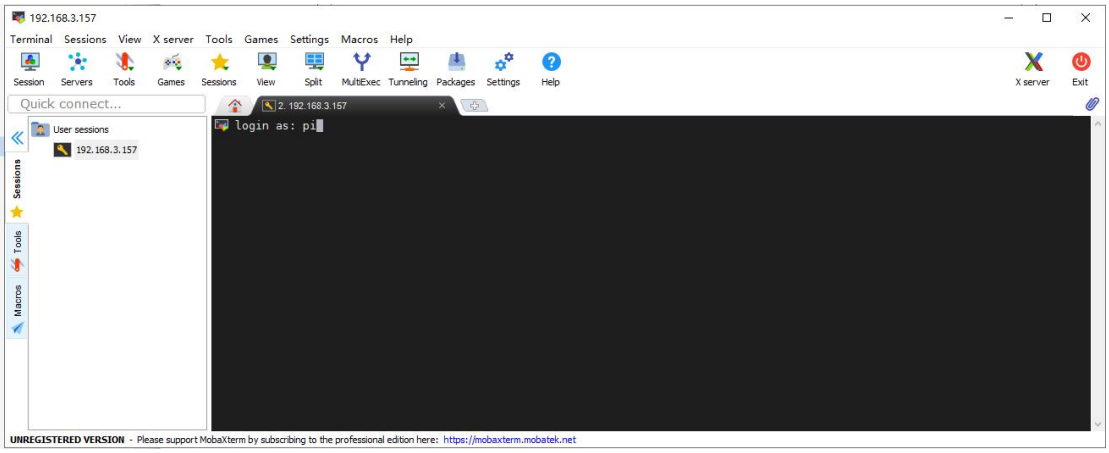
●MobaXterm is a terminal tool software that can be used to remotely control the Raspberry Pi and remote control is available when SSH is on. For Raspberry Pi's method of enabling SSH and automatically connecting to WIFI, please refer to steps **2.2** and **2.3**.

●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.

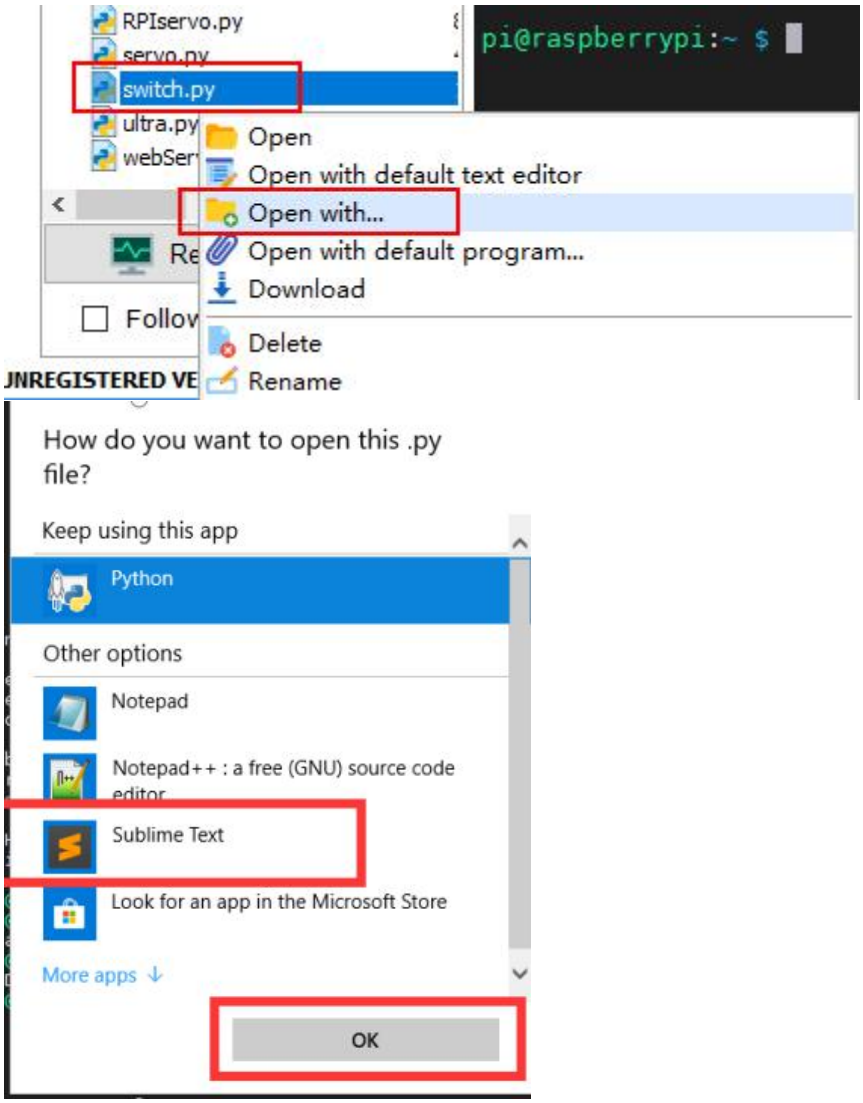
●After the success of the login, MobaXterm will automatically save the conversation, when connected to the raspberry pie again next time only need to double click on the left side of the IP address can be connected to the Raspberry Pi again, if there is no save username and password will need to input 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, press "CTRL+S" to save the file and it will be automatically synchronized to the Raspberry Pi.



●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]

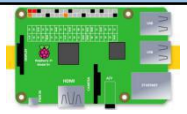
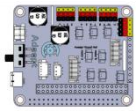
●You can learn more about Linux permissions through maketecheasier article from the article link

Understanding File Permissions.

6. Controlling WS2812 LED to Change Color

In this lesson, we will learn how to control WS2812 LED.

6.1 Components needed for this lesson

Components	Quantity	Picture
Raspberry Pi	1	
Robot HAT	1	
3 pin wire	1	
WS2812 RGB LED	1	

6.2 Introduction of WS2812 RGB LED

The WS2812 RGB module is a low-power RGB tri-color lamp integrated with a current control chip. Its appearance is the same as a 5050LED lamp bead, and each element is a pixel. The pixel contains an intelligent digital interface data latch signal shaping and amplifying drive circuit, as well as a high-precision internal oscillator and a 12V high-voltage programmable constant current control part, which effectively ensures the color of the pixel point light is highly consistent.

WS2812 LED light is a very commonly used module on our robot products. There are three WS2812 lights on each module. Please pay attention to the signal line when connecting. The signal line needs to be connected to WS2812 after being led out from the Raspberry Pi. At the "IN" end of the LED light module, when the next WS2812 LED module needs to be connected, the signal line is drawn 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 to install the driver board RobotHAT, the WS2812 LED module can be connected to the WS2812 interface on the RobotHAT using a 3pin cable.

We use a third-party library [rpi_ws281x] to control WS2812 LED lights, you can learn about it on https://github.com/richardghirst/rpi_ws281x.

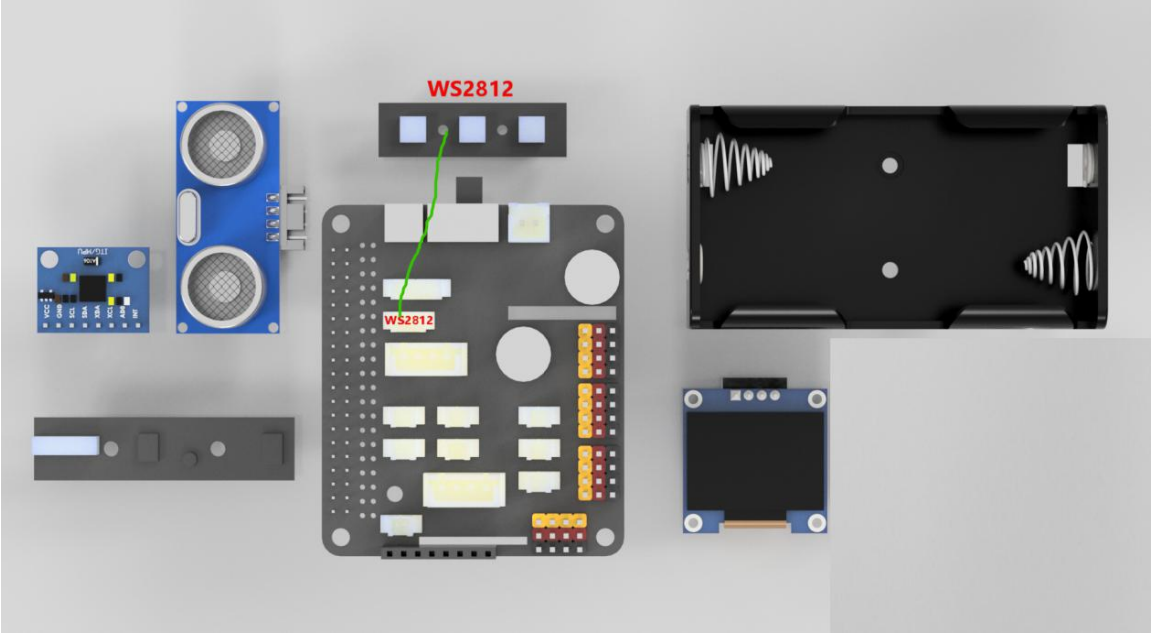
If you connect the WS2812 LED module to the WS2812 interface of RobotHAT, the signal line is equivalent to the GPIO 12 of the Raspberry Pi.

• The documentation has required to install all the dependent libraries needed by the robot. If you do not install the dependent libraries, you can use the following command to install rpi_ws281x for the Raspberry Pi. Since the Raspberry Pi has two built-in versions of Python, we use Python3 here Take the code as an example, so use pip3 to install the library.

```
pip3 install rpi-ws281x
```

6.3 Circuit diagram (wiring diagram)

When the WS2812 LED module is in use, the IN port needs to be connected to the WS2812 port on the RobotHAT driver board, as shown in the figure below:



6.4 How to control WS2812 LED

6.4.1 Running the code

1. Remotely log in to the Raspberry Pi terminal.

```
Linux raspberrypi 4.19.118-v7l+ #1311 SMP Mon Apr 27 14:26:42 BST 2020 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: Sat Aug 29 08:17:49 2020 from 192.168.3.208

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:~ $
```

2. Enter the command and press Enter to enter the folder where the program is located:

cd adeept_rasptankpro/server/

```
pi@raspberrypi:~ $ cd adeept_rasptankpro/server/
pi@raspberrypi:~/adeept_rasptankpro/server $
```

3. View the contents of the current directory file:

ls

```
pi@raspberrypi:~/adeept_rasptankpro/server $ ls
app.py            findline.py   instruction.txt   OLED.py        server.py
appserver.py      FPV.py        Kalman_filter.py  PID.py         servo.py
base_camera.py    FPVtest.py    LEDapp.py         __pycache__    switch.py
camera_opencv.py  functions.py  LED.py            robotLight.py  ultra.py
dist              info.py       move.py           RPIservo.py    webServer.py
pi@raspberrypi:~/adeept_rasptankpro/server $
```

4. Enter the command and press Enter to run the program:

sudo python3 LED.py

```
pi@raspberrypi:~/adeept_rasptankpro/server $ sudo python3 LED.py
```

5. After running the program successfully, you will observe that the WS2812 light turns red.

6.5 Main code program

After updating the code, the actual code may be different. For the complete code, refer to the file adept_rasptankpro/server/LED.py

Next, explain the program. This program is written in and executed on the Raspberry Pi. For the specific method, you can refer to Lesson 5 How to Edit the Code Program of the Raspberry Pi.

Import dependency

```
1. import time
2. from rpi_ws281x import *
```

构建 LED 控制类

```
1.  class LED:
2.      def __init__(self):
3.          self.LED_COUNT      = 16 #  Set to the total number of LED lights on the robot product, which can be more than the total number of LED lights connected to the Raspberry Pi
4.          self.LED_PIN        = 12 # Set as the input pin number of the LED lamp group
5.          self.LED_FREQ_HZ    = 800000
6.          self.LED_DMA        = 10
7.          self.LED_BRIGHTNESS = 255
8.          self.LED_INVERT     = False
9.          self.LED_CHANNEL    = 0
10.
11.     # Use the configuration items above to create a strip
12.         self.strip = Adafruit_NeoPixel(
13.         self.LED_COUNT,
14.         self.LED_PIN,
15.         self.LED_FREQ_HZ,
16.         self.LED_DMA,
17.         self.LED_INVERT,
18.         self.LED_BRIGHTNESS,
19.         self.LED_CHANNEL
20.         )
21.     self.strip.begin()
22.
23.     def colorWipe(self, R, G, B): # This function is used to change the color of the LED
24.         color = Color(R, G, B)
25.         for i in range(self.strip.numPixels()): # Only one LED light color can be set at a time, so a cycle is required
26.             self.strip.setPixelColor(i, color)
27.             self.strip.show() # After calling the show method, the color will really change
```

Instantiate the object and execute the method function. The function colorWipe() needs to pass in three parameters, which are R, G, and B, corresponding to the brightness of the three primary colors of light, 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 will be emitted. The specific example is as follows:

```
1.  if __name__ == '__main__':
2.      LED = LED()
```

```
try:
    while 1:
        LED.colorWipe(255, 0, 0)  #All 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 the WS2812 lights to switch among the three colors, press CTRL+C to exit the program.

• If you want to control the color of a single light, you can use the following code to achieve, where i is the serial number of the light, the serial number of the first light connected to the signal line from the driver board is 0, and the serial number of the second light is 1. , And so on, R, G, B are the brightness corresponding to the three color channels:

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

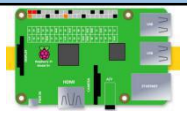
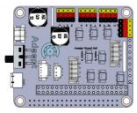
LED.strip.show()

• Note: You must use the Color() method to pack the RGB value, and then pass it to setPixelColor().

7. Controlling the Servo

In this lesson, we will learn how to control the servo.

7.1 Components needed for this course

Components	Quantity	Picture
Raspberry Pi	1	
Robot HAT	1	
180° Servo	1	

7.2 Introduction of servo

What is a servo?

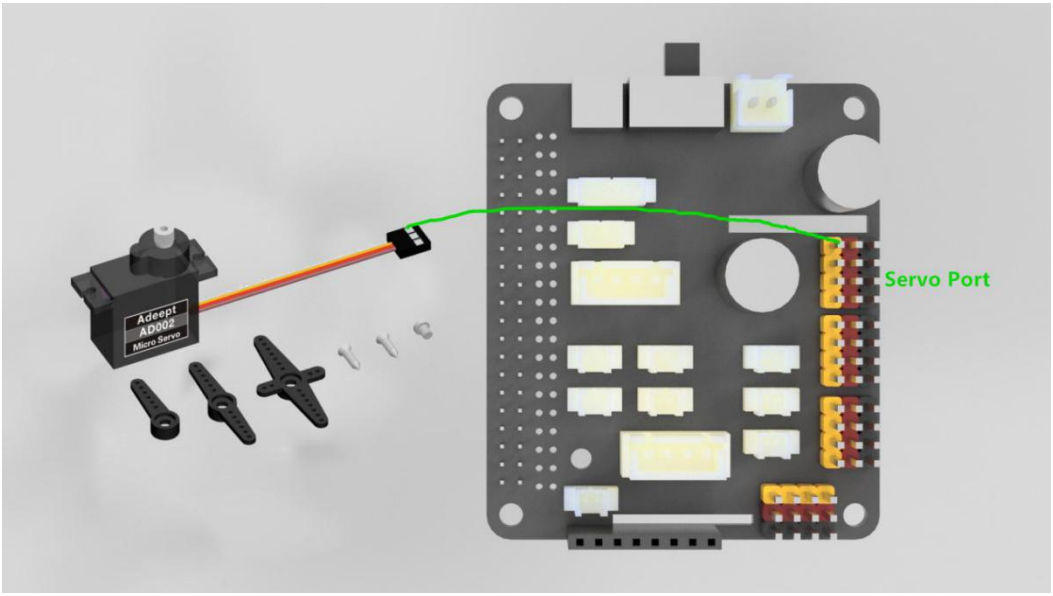
The servo is a position (angle) servo driver, which is suitable for those control systems that require constant angle changes and can be maintained. It has been widely used in high-end remote control toys, such as airplanes, submarine models, and remote control robots.

We use a 180° servo in this lesson, which can move between 0° and 180°. Since the 180° servo can use the PWM signal to control the rotation angle of a certain mechanism, it is commonly used in robot products. Module.

On our Raspberry Pi driver board Robot HAT, there is a dedicated PCA9685 chip used to control the servo. Raspberry Pi uses I2C to communicate with PCA9685. The servo is controlled by sending pulse signals from the microcontroller. These pulses tell the servo mechanism of the servo where to move. The 180° servo is as follows:

7.3 Circuit diagram (wiring diagram)

When the 180° Servo module is in use, it needs to be connected to the servo interface on the Robot HAT driver board. The yellow wire is connected to the yellow pin, the red wire is connected to the red pin, and the brown wire is connected to the On the black pins, as shown below:



7.4.1 Running the code

1. Remotely log in to the Raspberry Pi terminal.

```
Linux raspberrypi 4.19.118-v7l+ #1311 SMP Mon Apr 27 14:26:42 BST 2020 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: Sat Aug 29 08:17:49 2020 from 192.168.3.208

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:~ $
```

2. Enter the command and press Enter to enter the folder where the program is located:

cd adeept_rasptankpro/server/

```
pi@raspberrypi:~ $ cd adeept_rasptankpro/server/
pi@raspberrypi:~/adeept_rasptankpro/server $
```

3. View the contents of the current directory file:

ls

```
pi@raspberrypi:~/adeept_rasptankpro/server $ ls
app.py            findline.py   instruction.txt  OLED.py        server.py
appserver.py      FPV.py        Kalman_filter.py PID.py         servo.py
base_camera.py    FPVtest.py    LEDapp.py        __pycache__    switch.py
camera_opencv.py  functions.py  LED.py           robotLight.py  ultra.py
dist              info.py       move.py          RPIservo.py    webServer.py
pi@raspberrypi:~/adeept_rasptankpro/server $
```

4. Enter the command and press Enter to run the program:

sudo python3 servo.py

```
pi@raspberrypi:~/adeept_rasptankpro/server $ sudo python3 servo.py
```

5. After running the program successfully, you will observe that the servo will rotate regularly.

6. When you want to terminate the running program, you can press the shortcut key "Ctrl + C" on the keyboard.

7.4.2 Main code program

Complete code refer to servo.py / RPIservo.py

Control the servo to rotate to a certain angle

```
1.  import Adafruit_PCA9685 # Import the library used to communicate with PCA9685
2.  import time
3.
4.  pwm = Adafruit_PCA9685.PCA9685()  # Instantiate the object used to control the PWM
5.  pwm.set_pwm_freq(50)  # Set the frequency of the PWM signal
6.
7.  while 1:  # Make the servo connected to the No. 3 servo port on the RobotHAT drive board reciprocate
8.      pwm.set_pwm(3, 0, 300)
9.      time.sleep(1)
10.     pwm.set_pwm(3, 0, 400)
11.     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 products needs to be controlled by a 50Hz PWM signal. If you use For other servos, this value 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 port number of the servo, which corresponds to the number marked on the RobotHAT driver board, but pay attention When the steering gear is connected to the drive board, do not insert the ground wire, VCC and signal wire in the reverse direction, brown to black, red to red, and yellow to yellow; 0 is the deviation value for controlling the rotation of the steering gear, which is not used in our program This function is used to correct the deviation (the cause of the error of the steering gear can be referred to the precautions for structural assembly); 300 is the value of the PWM

duty cycle you want to set. Depending on the steering gear, this value represents a different steering gear angle. The PWM duty cycle range of the servo we use is about 100 to 560, which corresponds to a rotation range of about 0° to 180°.

•Using the above code to control the steering gear does not control the rotation speed of the steering gear. If we want a steering gear to slowly swing back and forth between two positions, we need to use the method of increasing or decreasing variables to control the steering gear.

Control the servo to move slowly

```
1.  import Adafruit_PCA9685 # Import the library used to communicate with PCA9685
2.  import time
3.
4.  pwm = Adafruit_PCA9685.PCA9685()   # Instantiate the object used to control the PWM
5.  pwm.set_pwm_freq(50)   # Set the frequency of the PWM signal
6.
7.  while 1:
8.      for i in range(0,100):  # Make the steering gear move slowly from 300 to 400
9.          pwm.set_pwm(3, 0, (300+i))
10.         time.sleep(0.05)
11.     for i in range(0,100):  #Make the steering gear move slowly from 400 to 300
12.         pwm.set_pwm(3, 0, (400-i))
13.         time.sleep(0.05)
```

•Using 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 great drawbacks. When the program is executed to the slow motion part of the steering gear, it will be blocked, which will seriously affect The performance effect of the program, so a multi-threaded solution is provided in our robot product program to solve this problem.

Non-blocking control method

• 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.

```
1.  import RPIservo #  Import a library that uses multiple threads to control the steering gear
2.  import time
3.
4.  sc = RPIservo.ServoCtrl()   # Instantiate the object that controls the servo
5.  sc.start()  # Start this thread, when the servo does not move, the thread is suspended
6.
7.  while 1:
8.      sc.singleServo(3, -1, 2)
9.      time.sleep(1)
10.     sc.stopWiggle()
11.
12.     sc.singleServo(3, 1, 2)
13.     time.sleep(1)
14.     sc.stopWiggle()
```

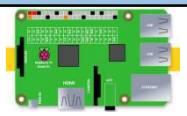
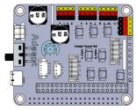
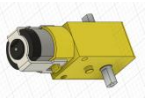
• Use the above code to control the servo to reciprocate, except for time.sleep(), it will not block the running of the context program.

• Call singleServo() to start motion, call stopWiggle() to stop motion, singleServo() requires three parameters, which are the port number of the servo to be controlled (3 is to control the No. 3 servo connected to RobotHAT) , The direction of the steering gear (1 or -1), and the speed of the steering gear (the larger the value, the faster the movement).

8. Controlling Motor to Rotate

In this lesson, we will learn how to control the motor.

8.1 Components needed for this lesson

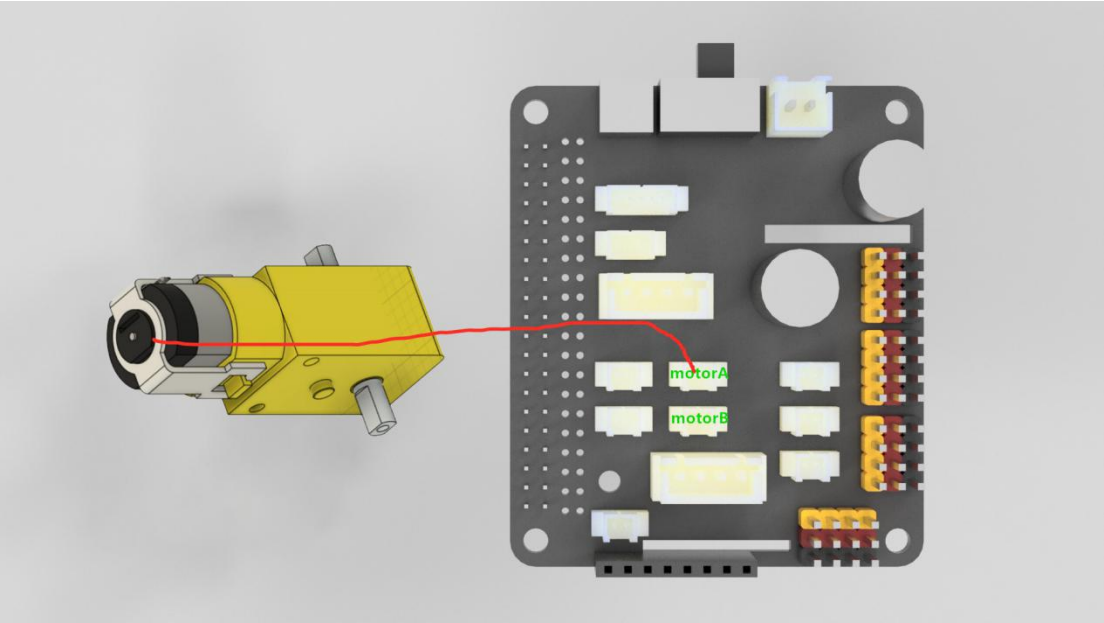
Components	Quantity	Picture
Raspberry Pi	1	
Robot HAT	1	
DC Motor	1	

8.2 Introduction of DC Motor

RaspTank Pro robot products use a DC motor as a power device. DC motor is a device that converts DC electrical energy into mechanical energy. It is widely used to drive various equipment, such as electric fans, remote control cars, and power windows. It is very suitable as The walking mechanism of the robot.

8.3 Circuit diagram (wiring diagram)

When the DC Motor module is in use, it needs to be connected to the motorA or motorB interface on the Robot HAT drive board. The yellow wire is connected to the yellow pin, the red wire is connected to the red pin, and the brown wire is connected On the black pins, as shown below:



8.4 How to control Motor

8.4.1 Running the code

1. Remotely log in to the Raspberry Pi terminal.

```
Linux raspberrypi 4.19.118-v7l+ #1311 SMP Mon Apr 27 14:26:42 BST 2020 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: Sat Aug 29 08:17:49 2020 from 192.168.3.208

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:~ $
```

2. Enter the command and press Enter to enter the folder where the program is located:

cd adeept_rasptankpro/server/

```
pi@raspberrypi:~ $ cd adeept_rasptankpro/server/
pi@raspberrypi:~/adeept_rasptankpro/server $
```

3. View the contents of the current directory file:

ls

```
pi@raspberrypi:~/adeept_rasptankpro/server $ ls
app.py            findline.py   instruction.txt  OLED.py        server.py
appserver.py      FPV.py        Kalman_filter.py PID.py         servo.py
base_camera.py    FPVtest.py    LEDapp.py        __pycache__    switch.py
camera_opencv.py  functions.py  LED.py           robotLight.py  ultra.py
dist              info.py       move.py          RPIservo.py    webServer.py
pi@raspberrypi:~/adeept_rasptankpro/server $
```

4. Enter the command and press Enter to run the program:

sudo python3 move.py

```
pi@raspberrypi:~/adeept_rasptankpro/server $ sudo python3 move.py
```

After running the program successfully, you will observe that Motor will rotate for about 1 second and then stop, and the program will also stop. If you need the motor to rotate again, you need to run the program again.

8.4.2 Main code program

The complete code reference file move.py.

```
1.  import time
2.  import RPi.GPIO as GPIO #Import the library used to control GPIO
3.
4.  GPIO.cleanup()  #  Reset the high and low levels of the GPIO port
5.  GPIO.setwarnings(False) #  Ignore some irrelevant errors
6.  GPIO.setmode(GPIO.BCM)  # There are three coding methods for the GPIO port of the Raspberry Pi, we
    choose BCM coding to define the GPIO port
7.
8.  """
9.  The following code definition is used to control the GPIO of the L298N chip. This definition is different for
    different Raspberry Pi driver boards. To
10. """
11. Motor_A_EN    = 4
12. Motor_B_EN    = 17
```

```
13.
14. Motor_A_Pin1  = 14
15. Motor_A_Pin2  = 15
16. Motor_B_Pin1  = 27
17. Motor_B_Pin2  = 18
18.
19. def motorStop():    # Stop motor rotation
20.     GPIO.output(Motor_A_Pin1, GPIO.LOW)
21.     GPIO.output(Motor_A_Pin2, GPIO.LOW)
22.     GPIO.output(Motor_B_Pin1, GPIO.LOW)
23.     GPIO.output(Motor_B_Pin2, GPIO.LOW)
24.     GPIO.output(Motor_A_EN, GPIO.LOW)
25.     GPIO.output(Motor_B_EN, GPIO.LOW)
26.
27. def setup():    # GPIO initialization, GPIO motor cannot be controlled without initialization
28.     global pwm_A, pwm_B
29.     GPIO.setwarnings(False)
30.     GPIO.setmode(GPIO.BCM)
31.     GPIO.setup(Motor_A_EN, GPIO.OUT)
32.     GPIO.setup(Motor_B_EN, GPIO.OUT)
33.     GPIO.setup(Motor_A_Pin1, GPIO.OUT)
34.     GPIO.setup(Motor_A_Pin2, GPIO.OUT)
35.     GPIO.setup(Motor_B_Pin1, GPIO.OUT)
36.     GPIO.setup(Motor_B_Pin2, GPIO.OUT)
37.
38.     motorStop() # Avoid the motor starting to rotate automatically after the initialization is completed
39.     try:        # Try here to avoid errors caused by repeated PWM settings
40.         pwm_A = GPIO.PWM(Motor_A_EN, 1000)
41.         pwm_B = GPIO.PWM(Motor_B_EN, 1000)
42.     except:
43.         pass
44.
45. def motor_A(direction, speed): # The function used to control the A port motor
46.     if direction == 1:
47.         GPIO.output(Motor_A_Pin1, GPIO.HIGH)
48.         GPIO.output(Motor_A_Pin2, GPIO.LOW)
49.         pwm_A.start(100)
50.         pwm_A.ChangeDutyCycle(speed)
51.     if direction == -1:
52.         GPIO.output(Motor_A_Pin1, GPIO.LOW)
53.         GPIO.output(Motor_A_Pin2, GPIO.HIGH)
54.         pwm_A.start(100)
55.         pwm_A.ChangeDutyCycle(speed)
56.
57. def motor_B(direction, speed):#  The function used to control the B port motor
58.     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 the motors of port A and B to rotate at full speed for 3 seconds'''
motor_A(1, 100)
motor_B(1, 100)
time.sleep(3)

"""
Control the motors of port A and B 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 of port A and B from rotating'''
motorStop()
```

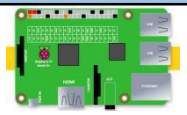
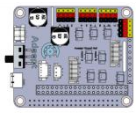
• The above code can be used to control the movement of the motor. The structure of the two functions motor_A and motor_B are the same, but the steering gear port to be controlled is different. This function requires two parameters, one is the direction and the other is the speed. The direction parameter can be 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. When the voltage is too low The motor may not rotate, so the speed value should not be too low.

•According to the actual situation, changing this speed value for a robot driven by a motor will only slow down the starting speed of the robot, and has little effect on the maximum speed, and when the given speed is too low, it will cause the motor to lock up.

9. Reading the Data of the Ultrasonic Ranging Module

In this lesson, we will learn how to read the data of the ultrasonic ranging module.

9.1 Components needed for this lesson

Components	Quantity	Picture
Raspberry Pi	1	
Robot HAT	1	
ultrasonic module	1	
4 pin wire	1	

9.2 Introduction of Ultrasonic Ranging Module

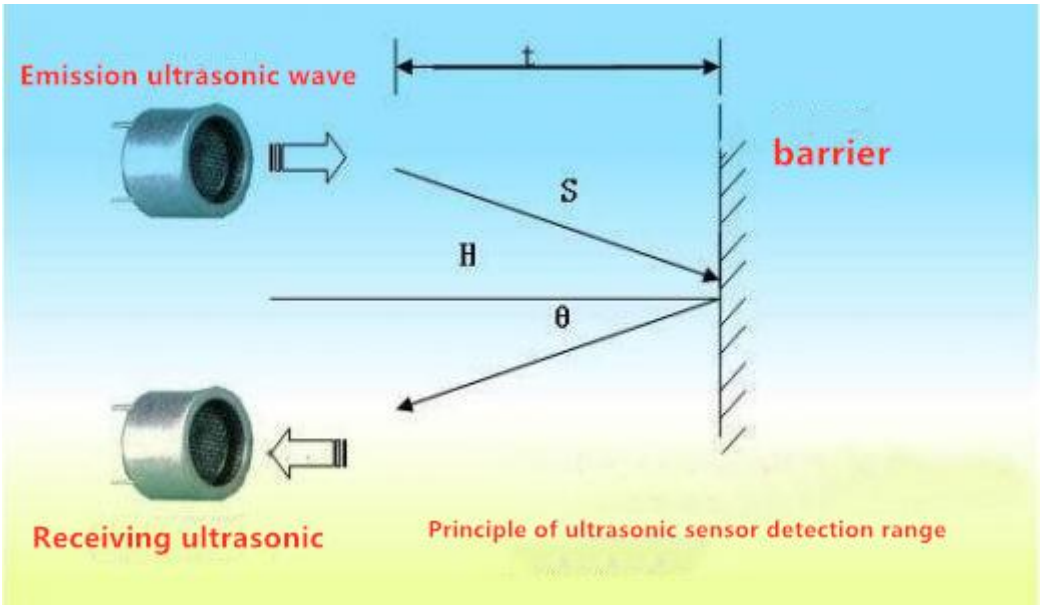
The ultrasonic ranging module used in our product has four pins, namely VCC, GND, Echo and Trig. HC-SR04 can provide non-contact distance sensing function of 2cm-400cm, and the ranging accuracy can reach 3mm ; Module includes ultrasonic transmitter, receiver and control circuit. The basic working principle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

Use the IO port TRIG to trigger the ranging and give a high level signal of at least 10us.

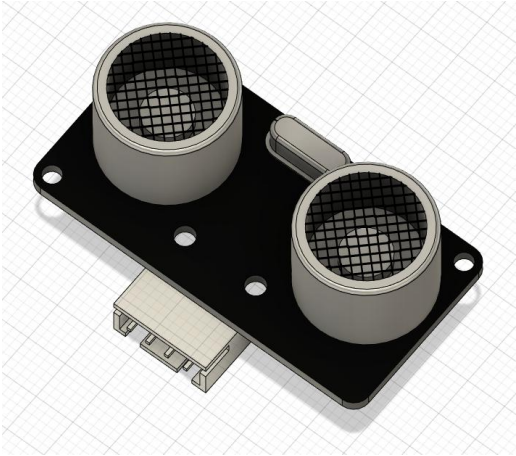
The module automatically sends 8 40khz square waves, and automatically detects whether there is a signal return.

There is a signal return, and a high level is output through the IO port ECHO. The duration of the high level is the time from the ultrasonic wave to the return.

The principle of distance detection by ultrasonic ranging sensor, the method of detecting distance by ultrasonic is called echo detection method, that is, the ultrasonic transmitter emits ultrasonic waves in a certain direction, and the timer starts timing at the same time as the launch time. The ultrasonic waves propagate in the air and encounter obstacles on the way. When the object surface (object) is blocked, it will be reflected back immediately, and the ultrasonic receiver will immediately stop timing when the reflected ultrasonic wave is received. The propagation speed of ultrasonic waves in the air is 340m/s. According to the time t recorded by the timer, the distance s from the launch point to the obstacle surface can be calculated, namely: s=340t/2. Using this principle of ultrasound, the ultrasonic ranging module is widely used in practical applications, such as car reversing radar, drones, and smart cars.

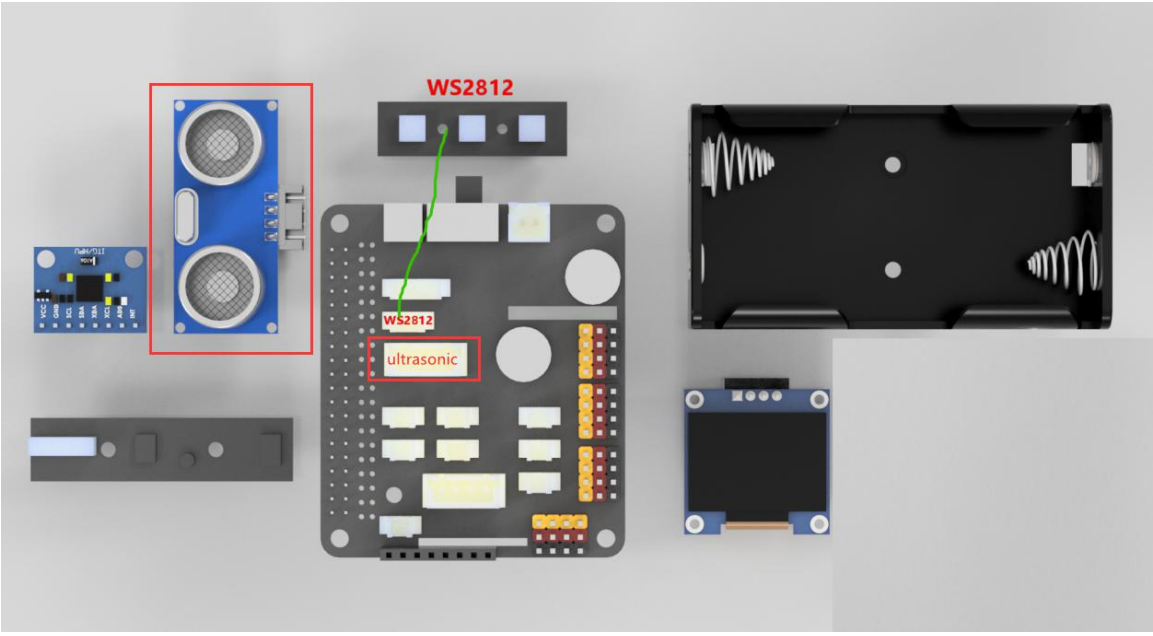


When using Robot HAT driver board, the ultrasonic sensor needs to be connected to the Ultrasonic interface on the driver board, and must not be connected to the IIC port to avoid burning the ultrasonic module. (IIC is an interface used to connect I2C devices, and the pin positions of VCC and GND are different from Ultrasonic).



9.3 Circuit diagram (wiring diagram)

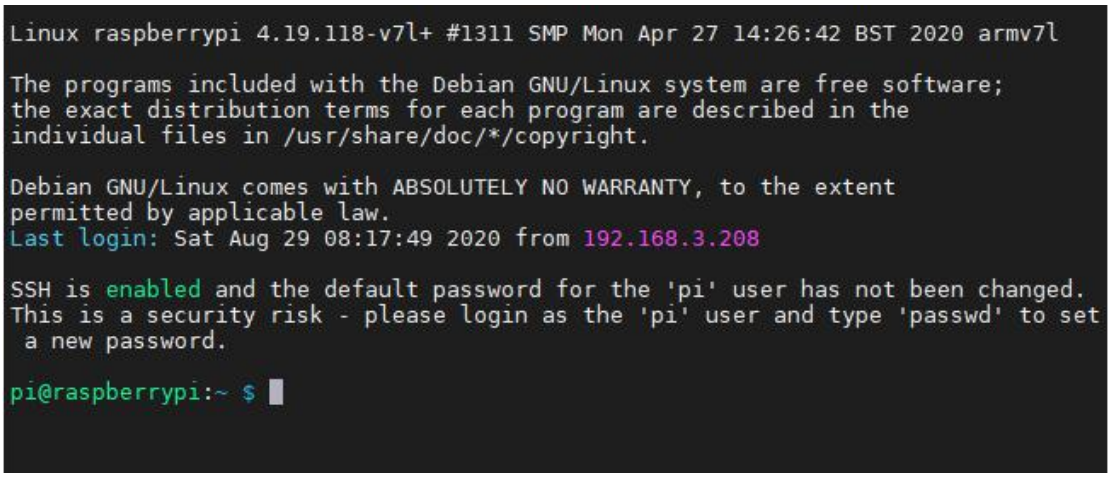
When using the Robot HAT driver board, you need to connect the ultrasonic sensor to the Ultrasonic interface on the driver board. Do not connect to the IIC port to avoid burning the ultrasonic module. (IIC is an interface used to connect I2C devices, and the pin positions of VCC and GND are different from Ultrasonic).



9.4 Obtaining the data of the ultrasonic sensor

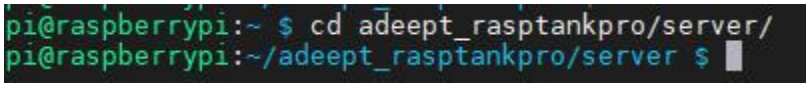
9.4.1 Running the code

1. Remotely log in to the Raspberry Pi terminal.



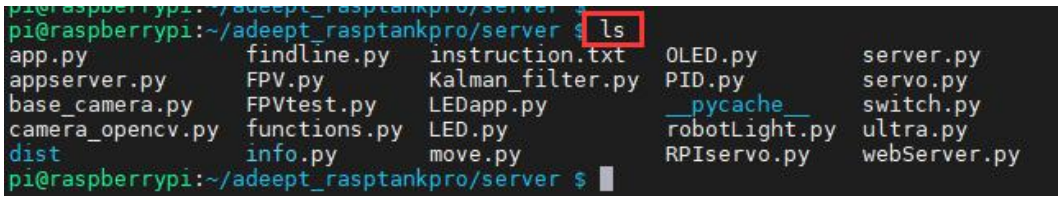
2. Enter the command and press Enter to enter the folder where the program is located:

cd adeept_rasptankpro/server/



3. View the contents of the current directory file:

ls



4. Enter the command and press Enter to run the program:

sudo python3 ultra.py

```
pi@raspberrypi:~/adeept_rasptankpro/server $ sudo python3 ultra.py
```

5. After successfully running the program, the command window will display the distance data of the obstacle detected by the ultrasonic sensor. When you use an object to approach it directly in front of the ultrasonic sensor, the detected distance data will change.

6. When you want to terminate the running program, you can press the shortcut key "Ctrl + C" on the keyboard.

9.4.2 Main code program

The complete code reference file ultra.py.

```
1.  import RPi.GPIO as GPIO
2.  import time
3.
4.  Tr = 11 # Pin number of the input end of the ultrasonic module
5.  Ec = 8 # Pin number of the output end of the ultrasonic module
6.
7.  GPIO.setmode(GPIO.BCM)
8.  GPIO.setup(Tr, GPIO.OUT,initial=GPIO.LOW)
9.  GPIO.setup(Ec, GPIO.IN)
10.
11. def checkdist():
12.     GPIO.output(Tr, GPIO.HIGH) # Set the input terminal of the module to high level, and send out an initial
    sound wave
13.     time.sleep(0.000015)
14.     GPIO.output(Tr, GPIO.LOW)
15.
16.     while not GPIO.input(Ec): # When the module no longer receives the initial sound wave
17.         pass
18.     t1 = time.time() # Write down the time when the initial sound wave was emitted
19.     while GPIO.input(Ec): # When the module receives the return sound wave
20.         pass
21.     t2 = time.time() # Write down the time when the return sound wave was captured
22.
23.     return round((t2-t1)*340/2,2) # Calculate the distance
24.
25. """
26. Output the ultrasonic ranging result once every second, output ten times
27. '''
28. for i in range(10):
```

```
29.     print(checkdist())
30.     time.sleep(1)
```

• Regarding the ultrasound module, since this is a commonly used module and is often used in many primary projects, in order to reduce the complexity of the program, although there are some blocking parts in this code, we did not use multithreading to solve If your project has a high demand for product expression, you can refer to the reconstruction of multi-threaded ultrasound.

• When your project needs to use the ultrasonic function, you do not need to rewrite the above code, just copy the ultra.py in the server folder of the robot program to the same folder as your own project. Then use the following code to get the ultrasonic ranging information:

```
1.  import ultra
2.  distance = ultra.checkdist()
```

10.RaspTank Pro Assembly Tutorial and Precautions

10.1 Documentation for structure assembly

Parts List

Copper pillar

M2*6 x4

M2.5*6+6 x8

M2.5*14 x4

M3*20 x8

M3*30 x4

M3*60 x4

Nylon column

M3*20 x4

M3*40 x7

Round head screw

M2*8 x14

M2*14 x4

M2.5*4 x8

M2.5*8 x2

M2.5*12 x1

M3x4 x44

M3x10 x40

M4x45 x2

Flat head screw

M3x6 x2

M3x16 11

Self-tapping screws

M1.4x6 x4

M1.7x6x6 x10

Nuts

M2 x22

M3 x12

Locknut

M3LOCK x13

M4LOCK x2

Mechanical Parts

F624ZZ bearing x2

481 metal gasket x4

Track (including coupling and wheels) x2

51105 Thrust Bearing x1

Electrical parts

0.96 OLED screen	x1
18650 battery holder	x1
RobotHAT driver board	x1
MPU6050 (pin)	x1
130 rpm DC motor (with cable)	x2
WS2812 light bar	x2
Ultrasonic (horizontal anti-reverse connection)	x1
Tracking module	x1
MG946R servo	x2
AD002 servo	x3
Camera Module	x1
Camera long cable	x1
3pin wire (2812)	x2
4pin wire	x2
5pin wire	x1
Servo extension wire	x2

Tool

Winding Pipe: 1

Large Cross-head Screwdriver: 1

Small Cross-head Screwdriver: 1

Cross Socket Wrench: 1

Hexagon screwdriver suitable for the hexagon socket screws of the track coupling: 1

Bring your own parts

Raspberry Pi x1

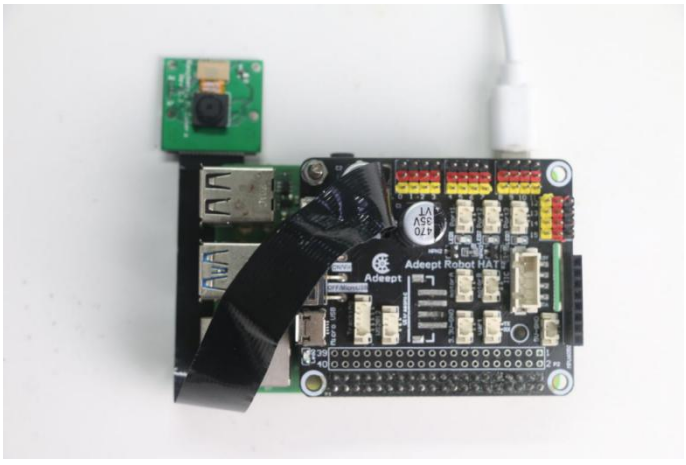
18650 battery x2 (**Choose an 18650 battery with "high rate discharge", or choose an 18650 battery that supports a maximum output current of at least 4A**)

10.2 Tips for structural assemblage

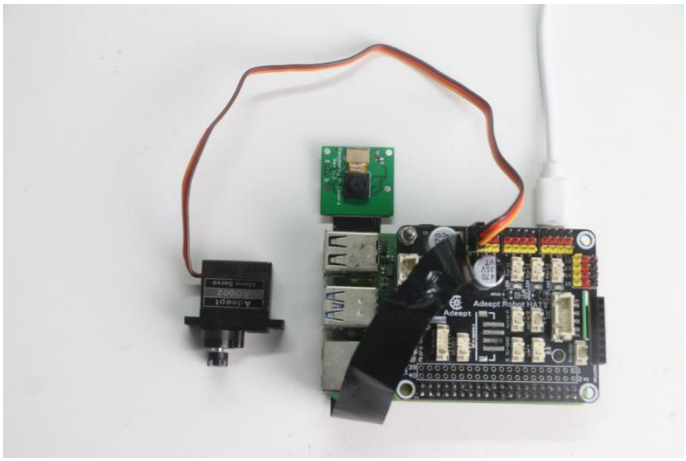
Due to the large number of servo used in this product, the installation of the servo has a greater impact on the performance of the product, so you need to power on the servo and control the servo to rotate to the middle position before installing the servo rocker arm, in this way, the rocker arm of the servo installed at the specified angle can be in the middle position of it.

Steps to power on the servo:

1.Boot the Raspberry Pi.



2.Install the servo. Pay attention to the direction of the interface during installation. The yellow wire is connected to the yellow pin, the red wire is connected to the red pin, and the brown wire is connected to the black pin.



3.After the Raspberry Pi is powered on, it will automatically run webServer.py, and after webServer.py is running, it will control all the servo ports to send signals to move to the middle position. When you install the servo rocker arm, you can connect the servo to any port at any time. When the servo is connected to the port, the servo will rotate to the middle position. Install the rocker arm according to the specified angle then you can disconnect the servo from the port and plug in a new servo that has not been installed with a rocker arm. This new servo without a rocker arm will also rotate to the middle position.

4. It takes a while to boot the Raspberry Pi to control the PCA9685 to set all the servo ports to the signal to rotate to the middle position. The boot process of the Raspberry Pi is about 30-50s.

5. All the servo rocker arm installation angles shown in the document are the middle position of the servo rotation. When the servo is turned on to the middle position, install the servo rocker arm at the angle shown in the document.

6. Note: all lock nuts involved in this document should not be tightened.

10.3 Precautions for power supply during assembly

1. When you are performing software installation, structural assembly or program debugging, you can use a USB cable to power the Raspberry Pi. If the Raspberry Pi is installed with RobotHAT, you can connect the USB cable to the USB port on the RobotHAT. RobotHAT will power the Raspberry Pi with the GPIO interface.

2. Different raspberry parties have different current requirements. For example, the Raspberry Pi 3B needs at least 2A to boot up, and the Raspberry Pi 4 needs 3A to boot normally. You can check before you use the power adapter to power the Raspberry Pi. The specifications on your power adapter.

3. When RobotHAT is connected to a load, such as a motor or multiple servos, a high-current power supply is required to connect to the Vin on the RobotHAT. You can use two 18650 batteries that support high-current to power the RobotHAT.Our product will provide a 18650 battery holder with 2pin interface, you can directly connect it with RobotHAT.

4. When using the USB interface on RobotHAT to supply power, the switch of RobotHAT does not control whether to supply power, the switch of RobotHAT can only control the power supply of Vin.

5. Do not use the USB port and Vin on the RobotHAT 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 turn the switch on the RobotHAT to OFF, so that when you use the USB cable to connect to the RobotHAT , RobotHAT is powered by USB.

6. If your robot restarts automatically after booting, or disconnects and restarts at the moment the robot starts moving after normal booting, it is most likely because your power supply does not output enough current. The robot will automatically restart when it is turned on. Run the program to place all the servos in the neutral position. The voltage drop generated during this process causes the Raspberry Pi to restart.

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

8. You can also use power lithium battery to power RobotHAT, RobotHAT supports power supply below 15V.

9. When installing the servo rocker arm in the structure assembly, you can use the USB cable to power the RobotHAT. After the Raspberry Pi with the robot software installed, it will control the RobotHAT to set all the servo ports to output neutral signals. You can connect the servo to any servo port, the gear of the servo will turn to the neutral position, and then you can install the servo rocker arm according to the specified angle. After the rocker arm is installed, you can disconnect the servo and RobotHAT. When you need to install the rocker arm of the second servo, connect the second servo to any servo port on the drive board.

10.4 Assembly

10.4.1 Screw color description

•In order to make the structural assembly process more intuitive, we dye the screws used in the product. During the assembly process, you can refer to the fastener color in the tutorial to determine which type of screw and nut to use.

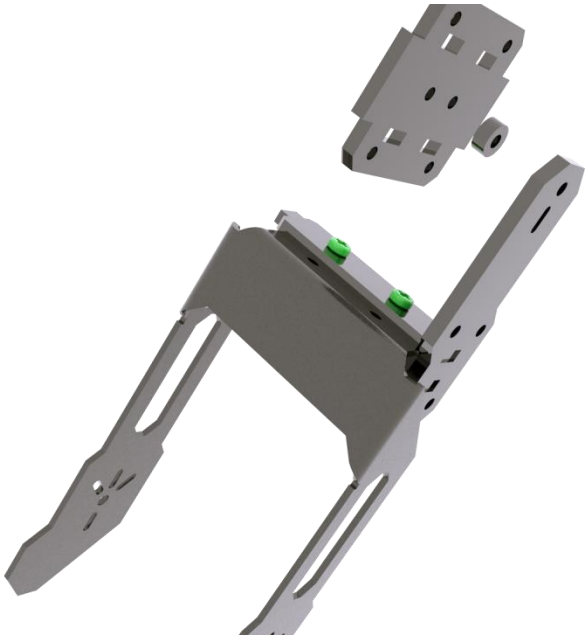
•The actual color of the product is subject to the product, and the actual screws are not colored.

• The screws of M1.4x6 and M4x45 are very different, so we use silver in the tutorial and they are not dyed.

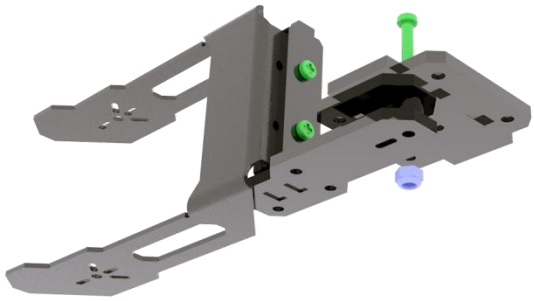
10.4.2 Robotic arm assembly

Use two M3X16 screws and two M3 nuts to fix the three acrylic panels shown above on the aluminum alloy base of the robotic arm.

Prepare the acrylic panel shown on the right side of the above picture. There is no need to fix it on the assembled component. Since the acrylic panel on the right has directions, the rendering diagram will fix it on the component to indicate the subsequent assembly position.

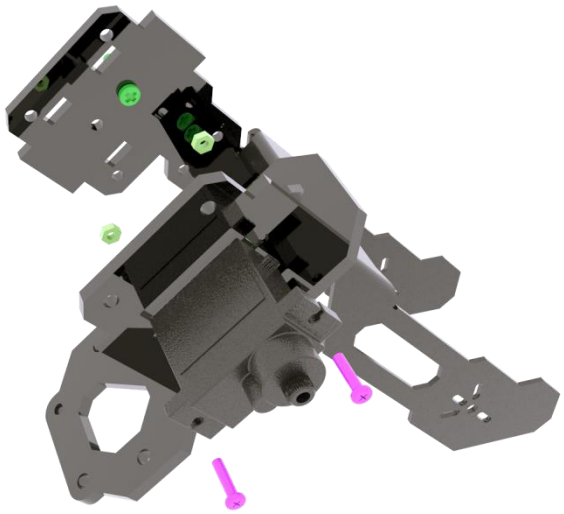


Prepare the acrylic panel and acrylic gasket as shown in the picture above.

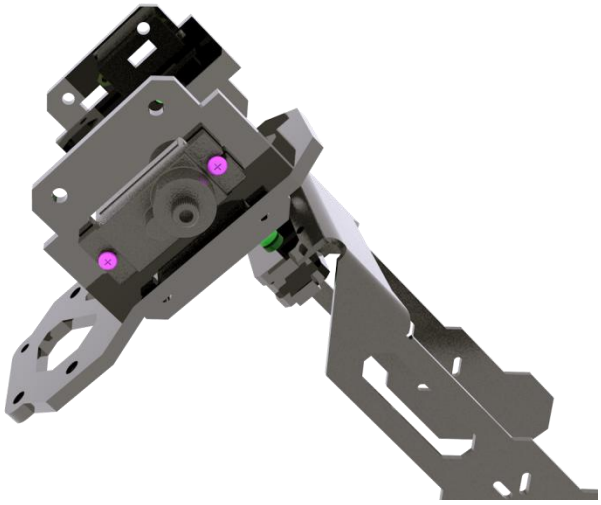


Use M3X16 screws and M3-LOCK lock nuts to fix the acrylic panel and acrylic washer as shown in the figure on the side panel. It should be noted that the lock nut should not be tightened too tightly, because this is the rotating pair of the movable part. After tightening, it will not be able to rotate and the servo will be blocked.

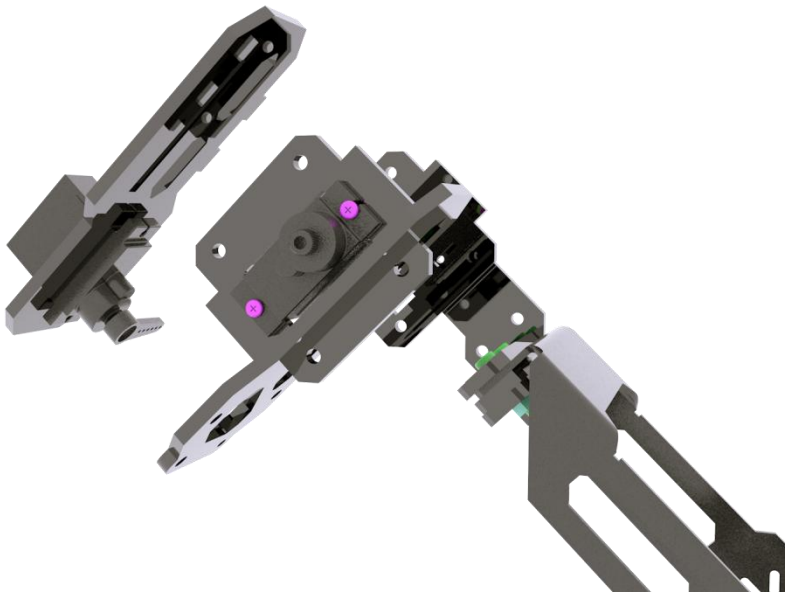
- • Note: All lock nuts involved in this document should not be tightened.



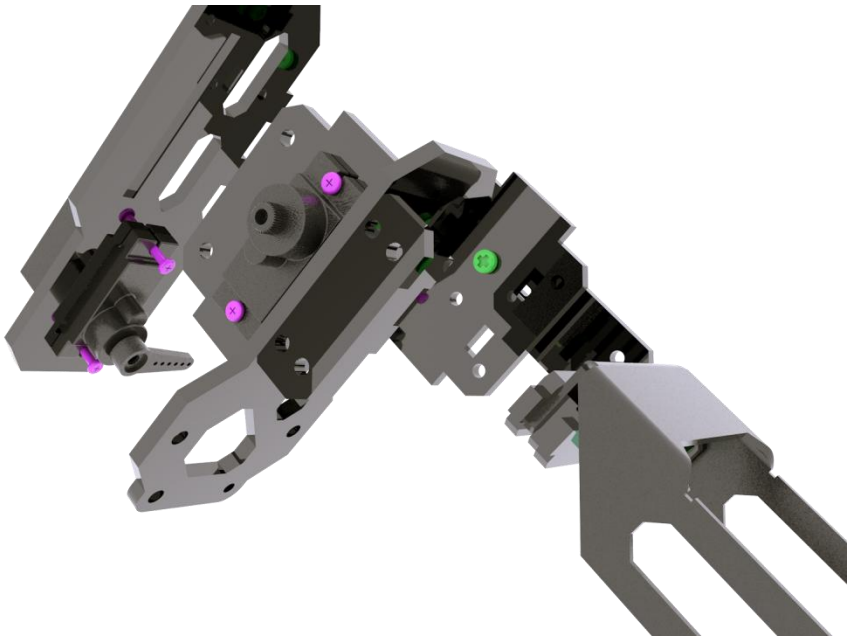
Use M2X8 screws and M2 nuts to fix the AD002 servo on the acrylic panel shown in the figure above. The acrylic panel here is similar in shape to the acrylic panel just installed, except that there are more mounting holes for fixing the
servo.



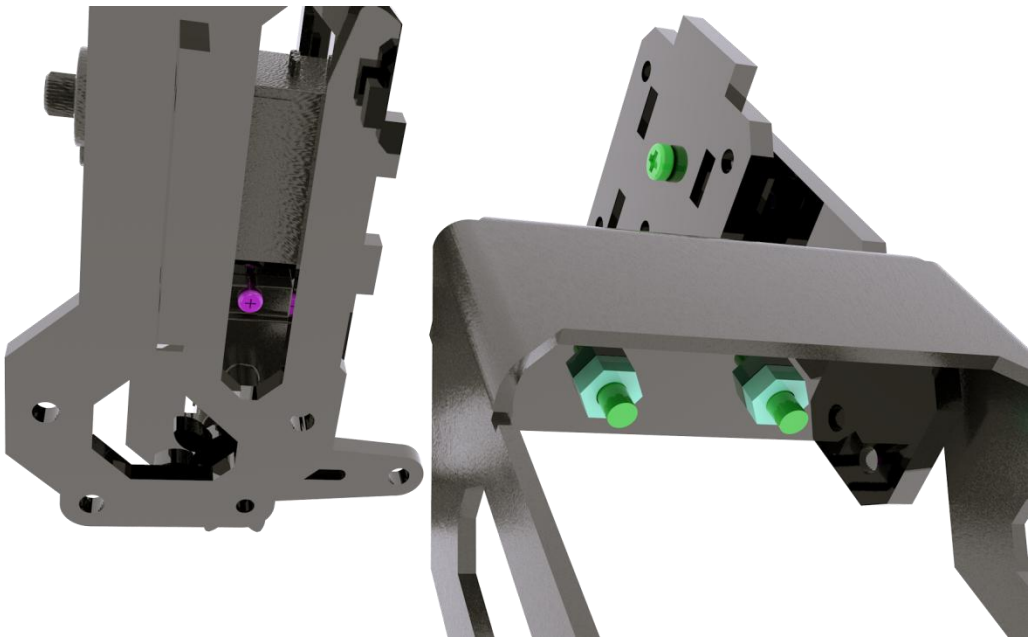
It looks like the picture above after installation.



Prepare the acrylic panel (top panel of GRIPPER) and AD002 servo as shown in the picture above.

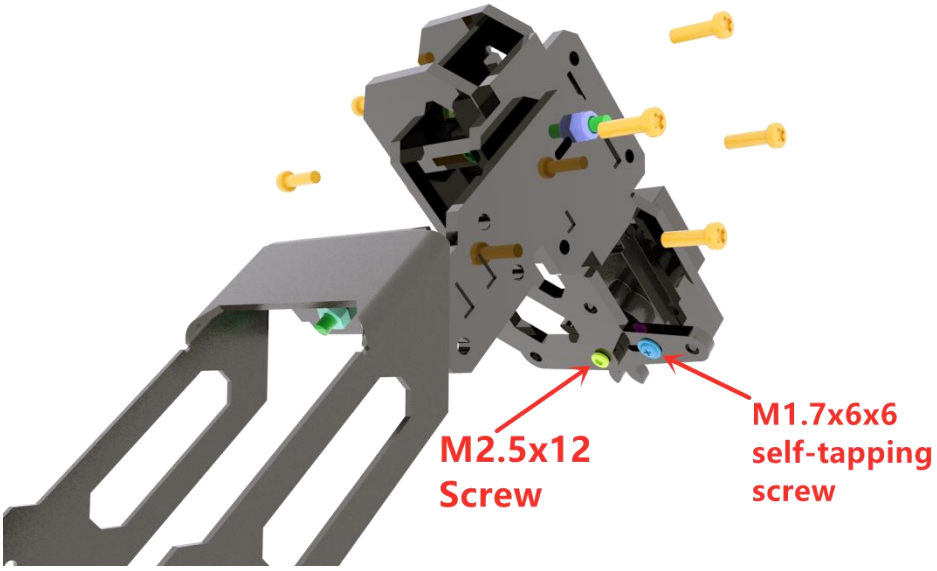


Use M2X8 screws and M2 nuts to fix AD002 servo to the upper panel of the arm chuck.

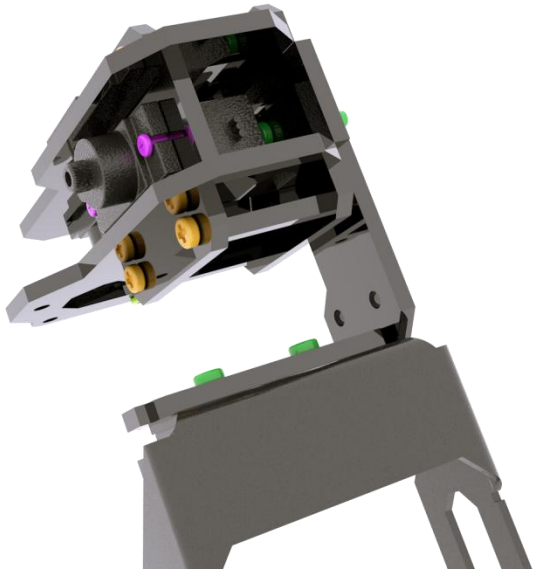


Assemble the upper panel and lower panel of the chuck. Here you can load the gear shown above between the servo arm and the lower panel. For the time being, you don't need to fix it. You only need to know that the gear is installed here. The gear is clamped in this position to avoid the gear cannot be installed due to too small gap after assembly.

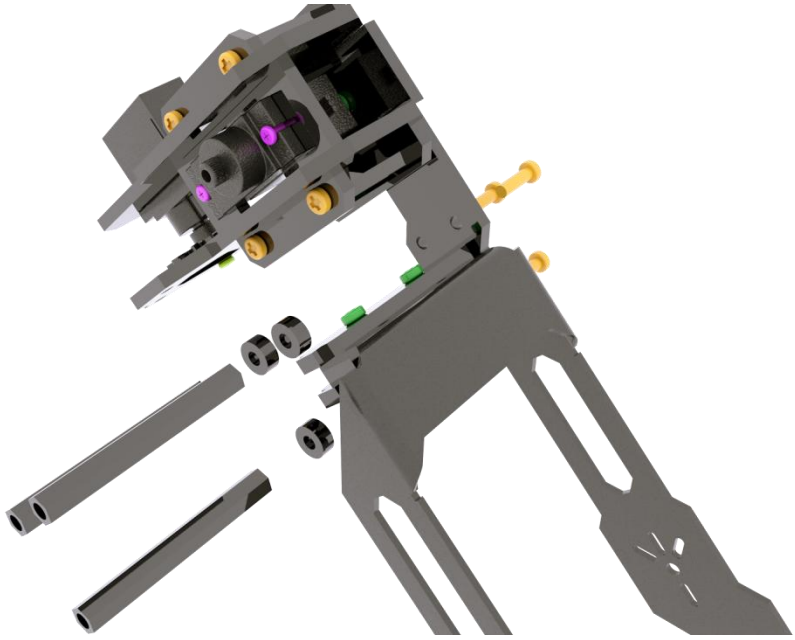
Use four M3X20 nylon posts and eight M3X10 screws to fix the left and right panels of the chuck together.



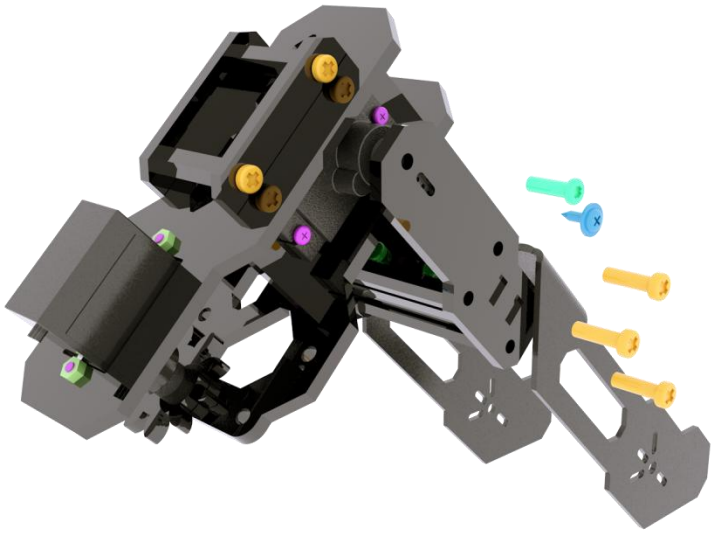
As shown above, tighten the M3X10 screws. Use 1 M2.5x12 Screw and 1 M1.7x6x6 self-tapping screw to fix.



The assembled appearance is shown in the figure.

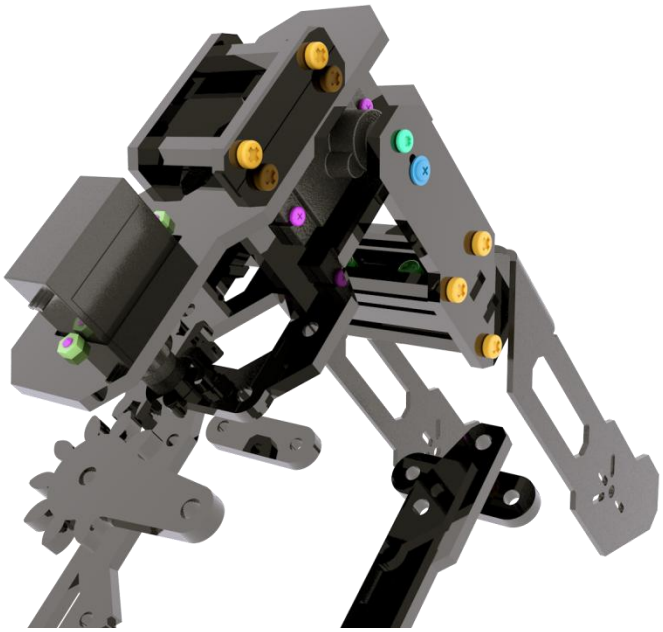


Use three M3X10 screws to fix the acrylic washer and M3X40 nylon column to the side panel of the robot arm.



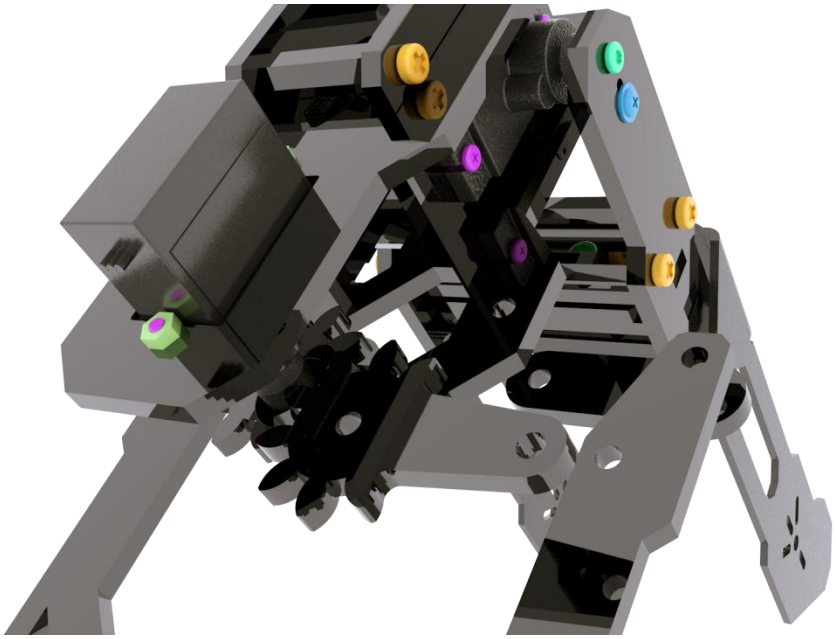
Use three M3X10 screws to fix the panel on the other side.

Use M1.7X6X6 self-tapping screws to fix the side panel and the servo arm, and use M2.5X8 screws to fix the servo to the side panel.

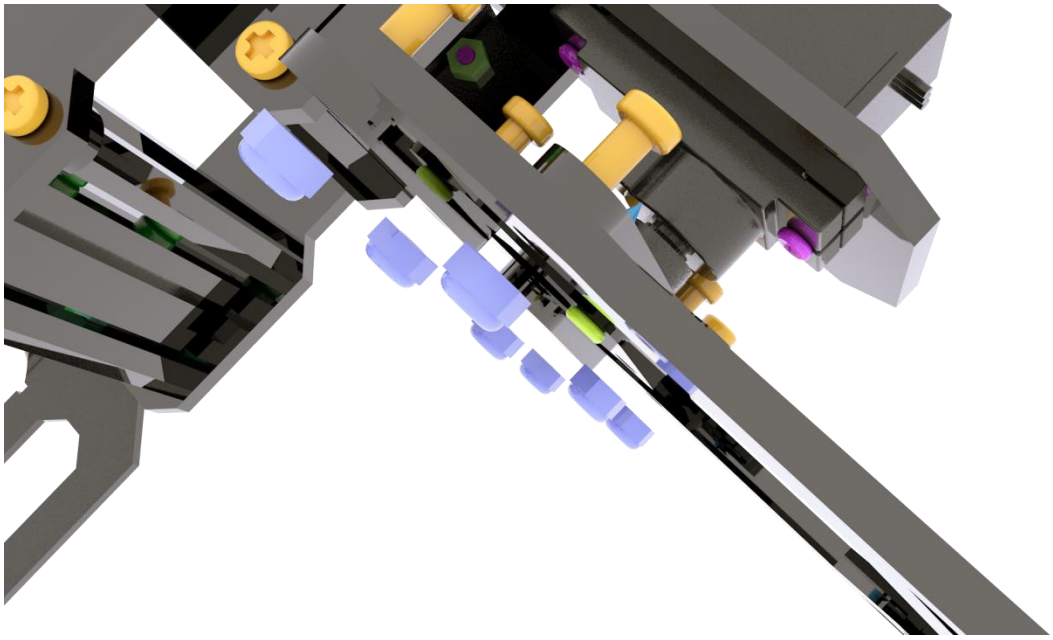


The assembled look is shown in the figure above.

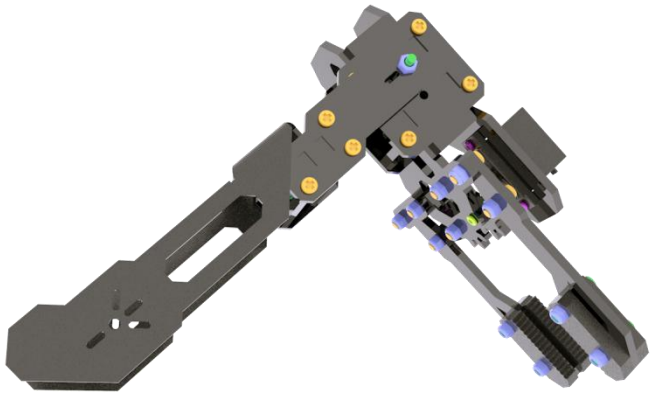
Prepare the gear on the other side and the connecting rod parts for the chuck.



The installation method is shown in the figure above.

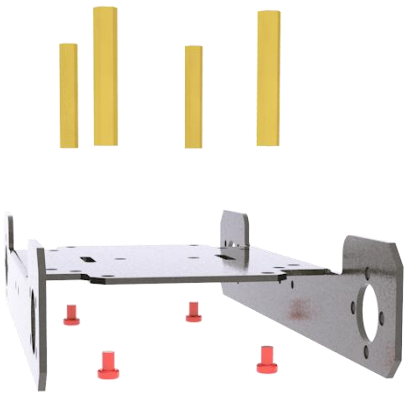


Use M3X10 screws and M3-LOCK locknuts to assemble the connecting rods.

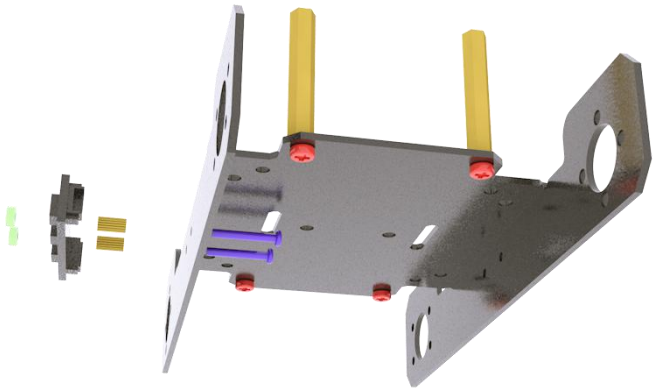


The robot arm after assembly is as shown in the figure above.

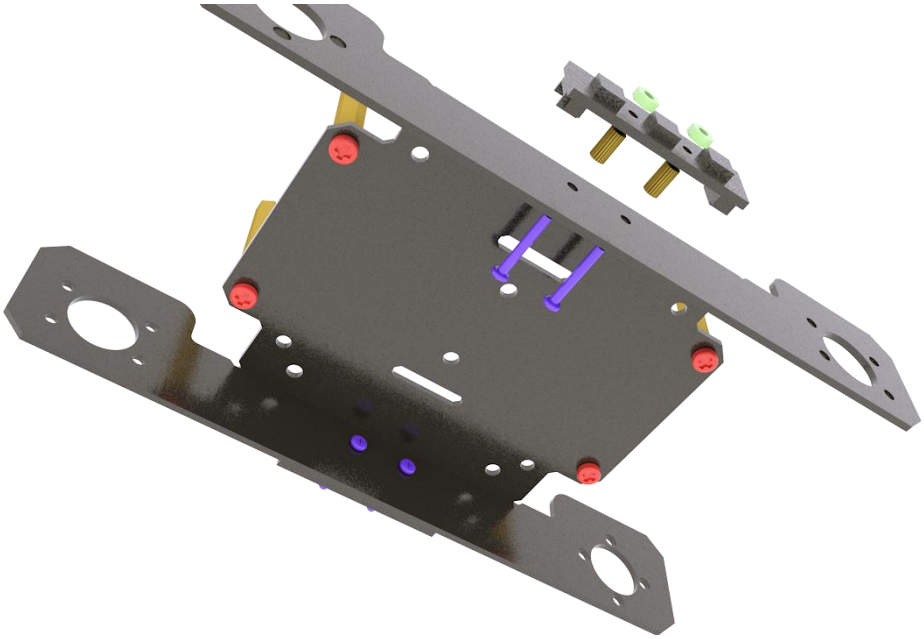
10.4.3 Body assembly



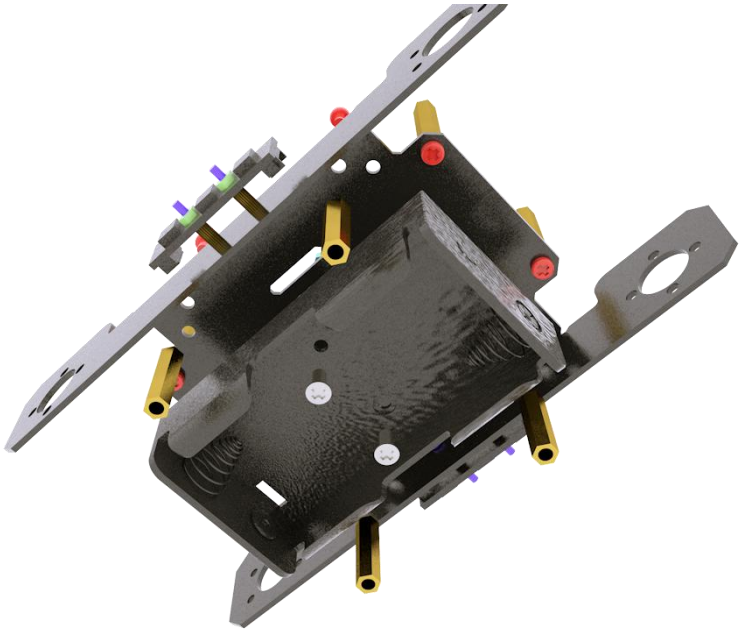
Use M3X4 screws to fix the M3X30 on the aluminum alloy base. The specific installation location is shown in the figure below.



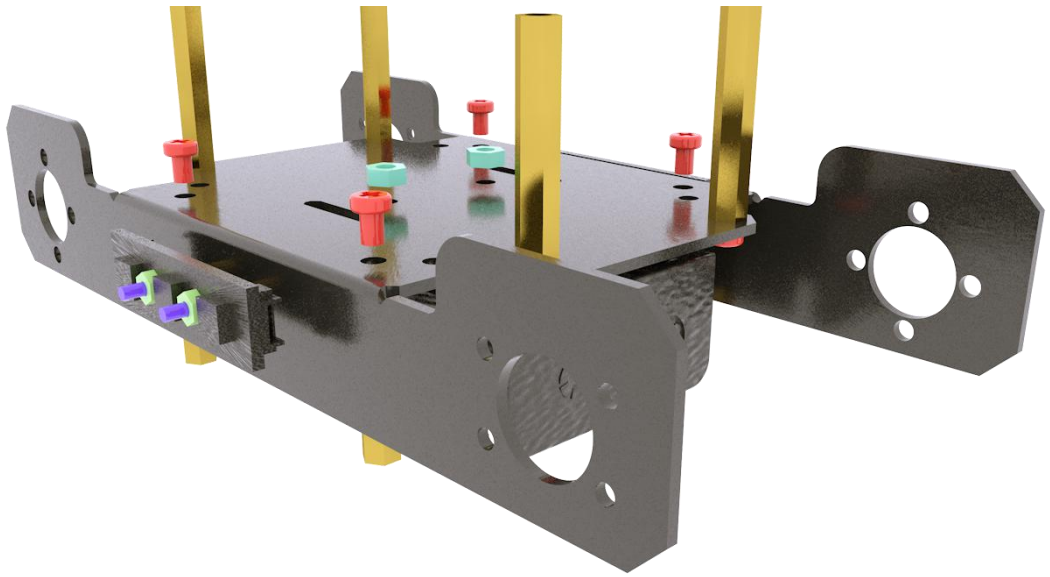
Use M2X14 screws to pierce through the side panel of the aluminum alloy base,then pierce through M2X6 copper posts and WS2812 light bar, and use M2 nuts to fix.



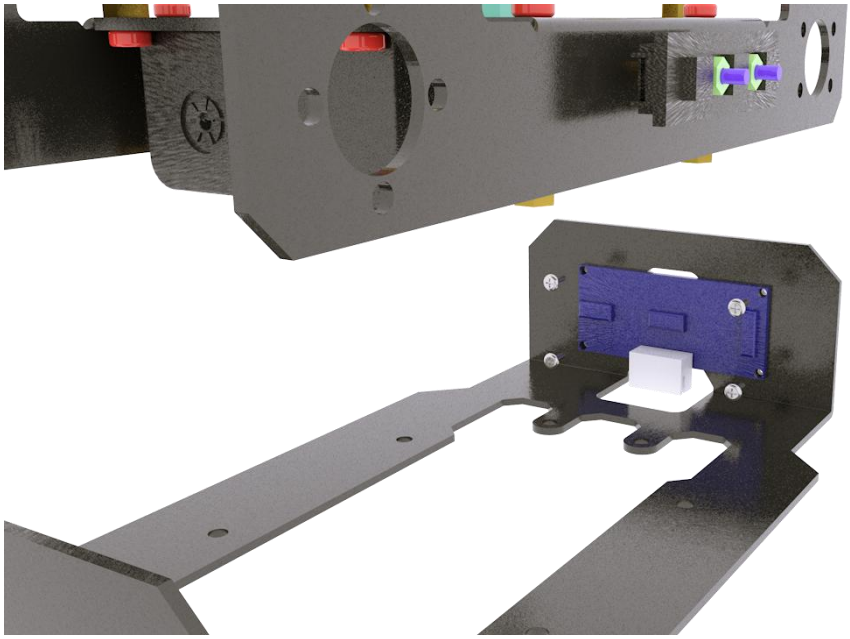
The light bar on the other side is installed as above.



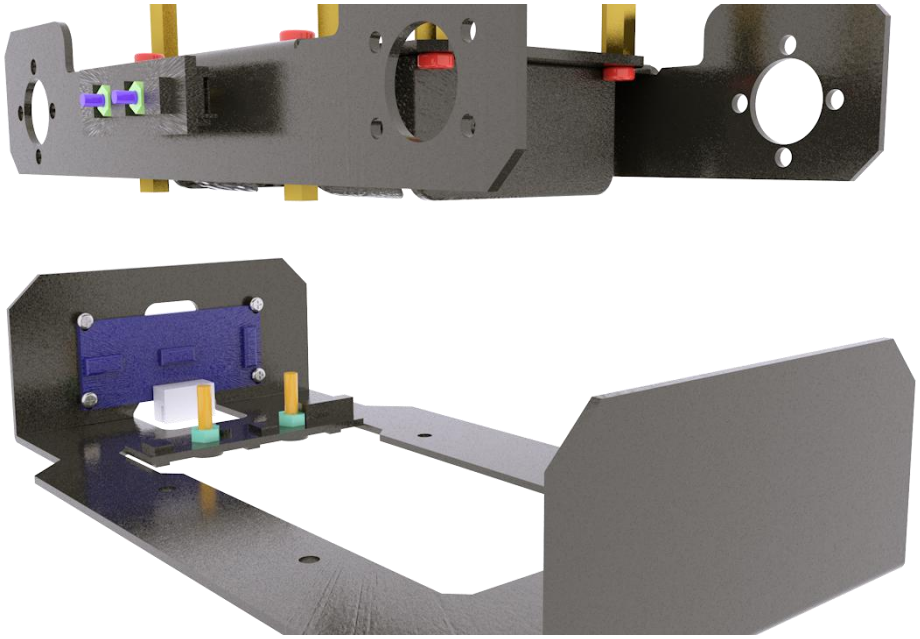
Use M3X4 screws to fix the M3X20 copper column on the aluminum alloy base, and use M3X6 countersunk head screws and M3 nuts to fix the battery holder on the aluminum alloy base. The fixed position is shown in the figure below.



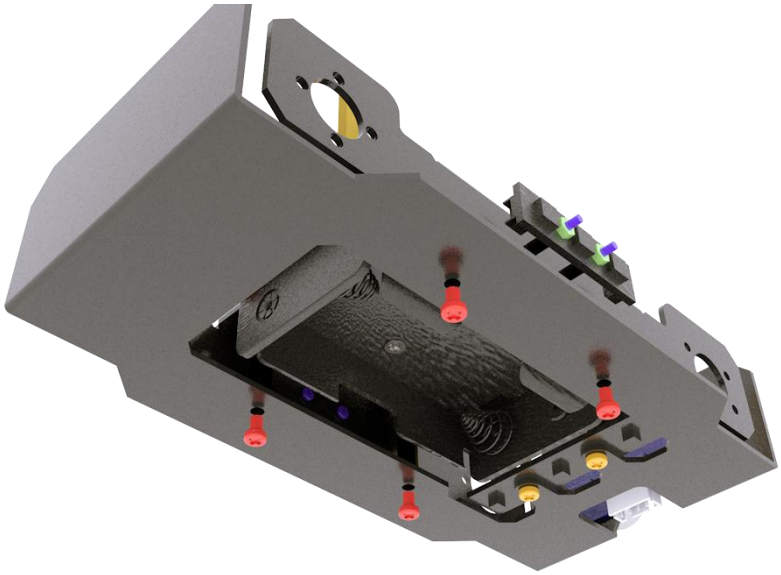
The picture above shows the fixed position of the M3X20 copper column and the battery holder.



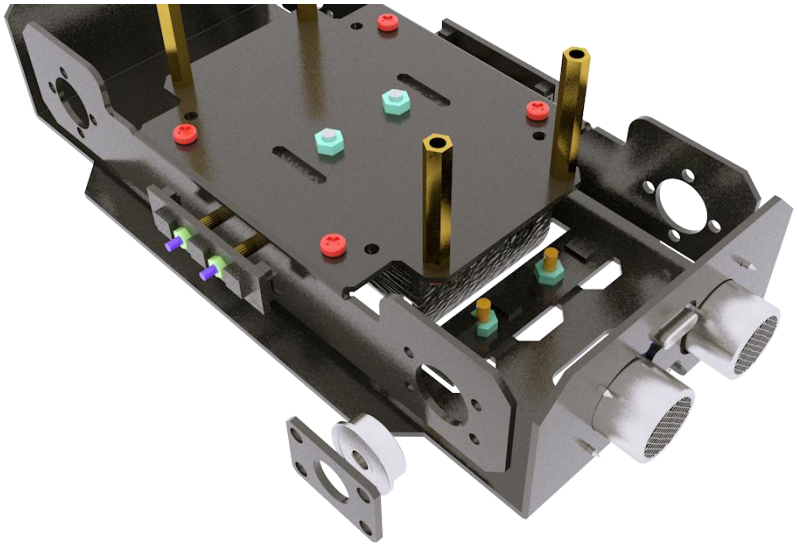
Use M1.4X6 self-tapping screws to fix the ultrasonic module on the lower plate.



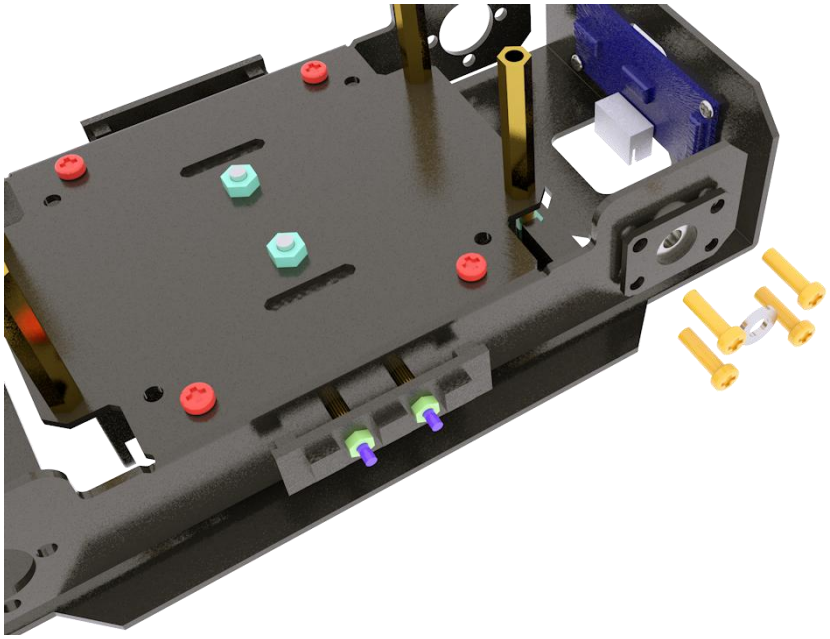
Use M3X10 screws and M3 nuts to fix the tracking module on the lower plate. Note that the arrow on the tracking module needs to point to the front of the robot to install.



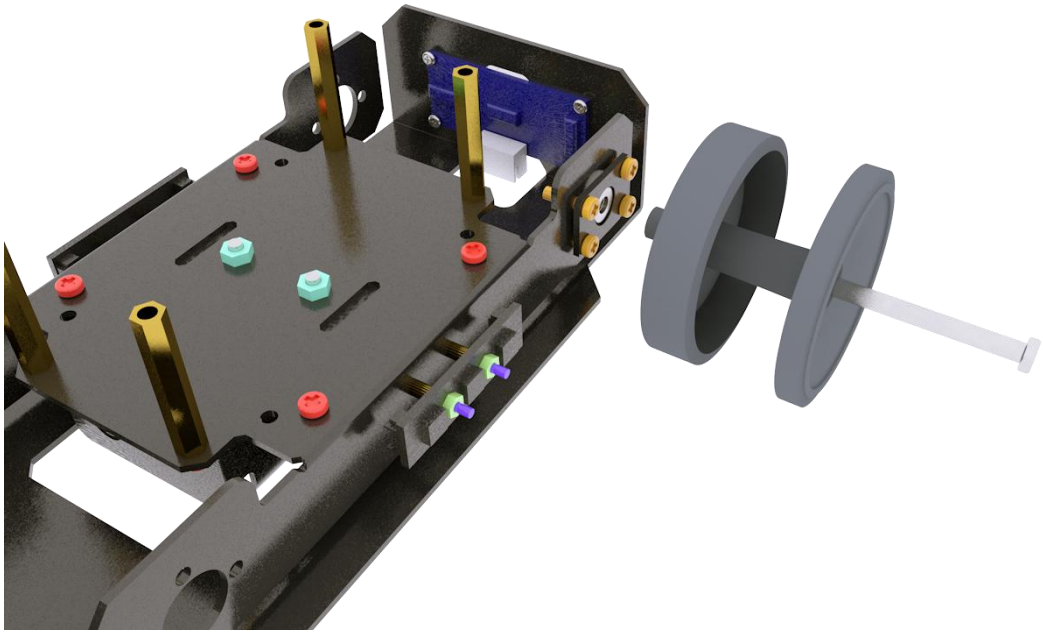
Use M3X4 screws to fix the lower plate to the base.



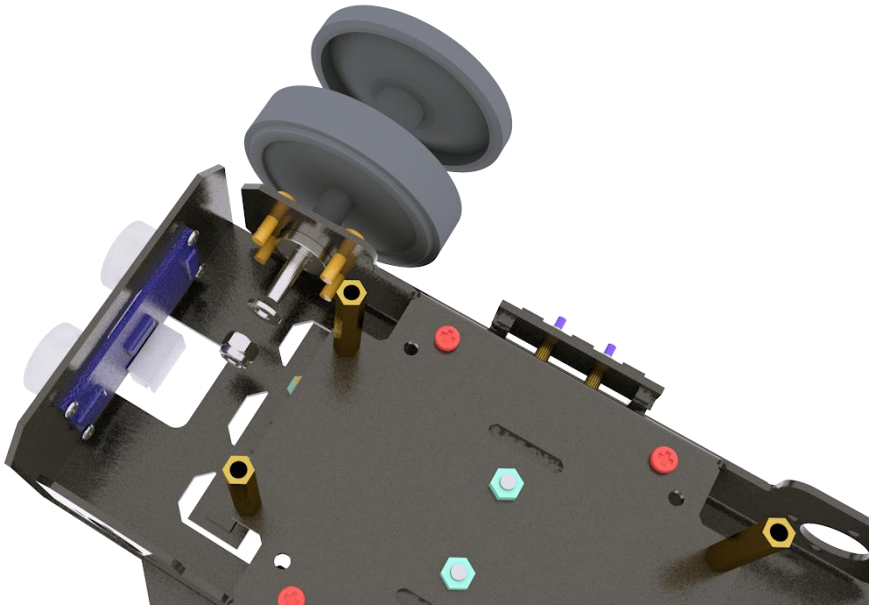
Install the F624ZZ bearing to the base.



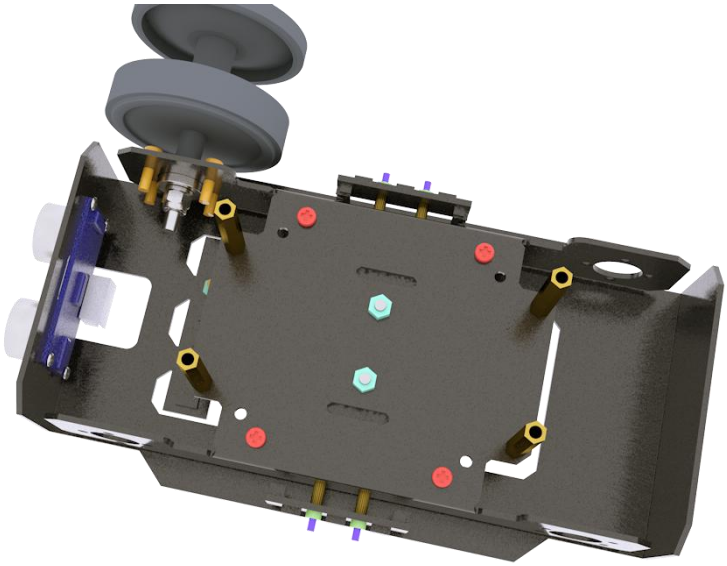
Use M3X10 screws to fix. Prepare the metal gasket as shown in the picture, and install it between the bearing and the wheel.



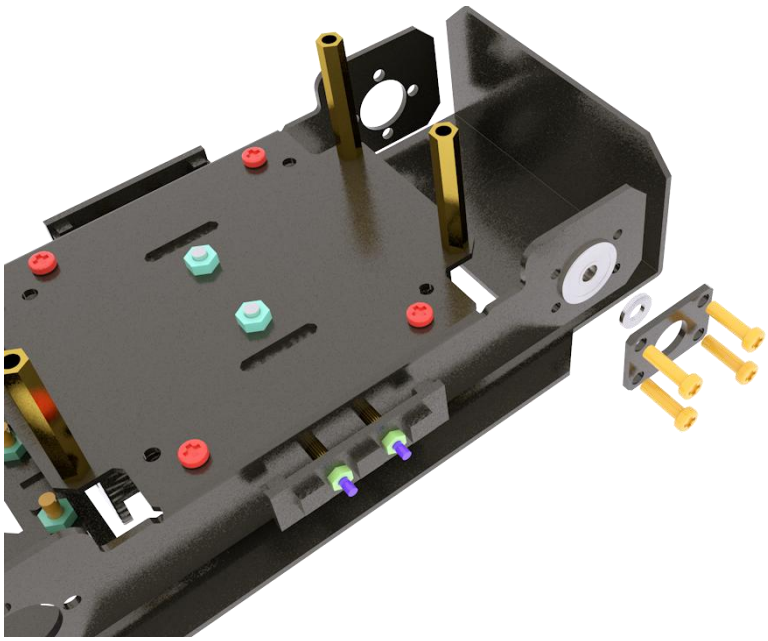
Use M4X45 screws to install the wheels.



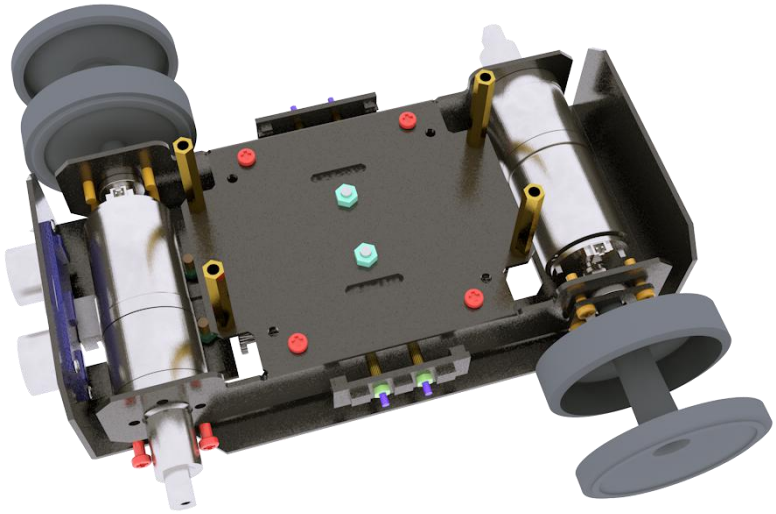
Use M4-LOCK lock nuts to fix the wheels. Note that the spacers as shown in the figure also need to be installed between the lock nuts and the bearings.



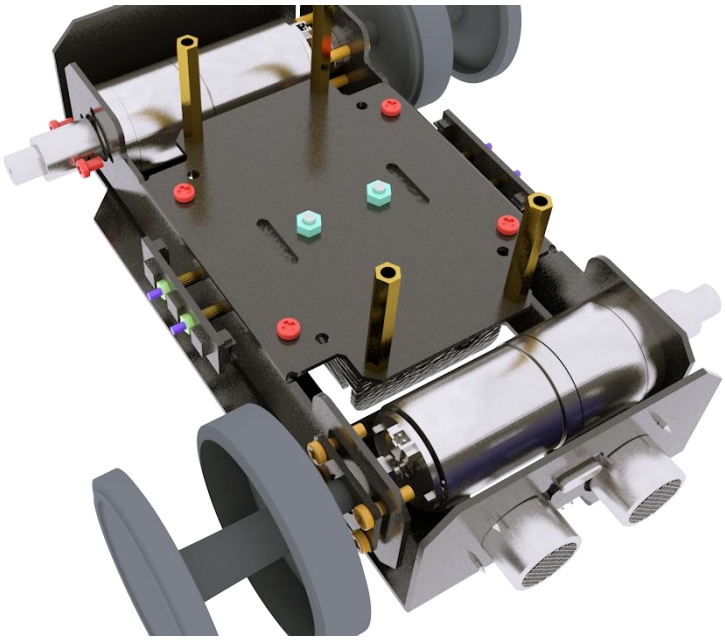
It looks like the picture after installation.



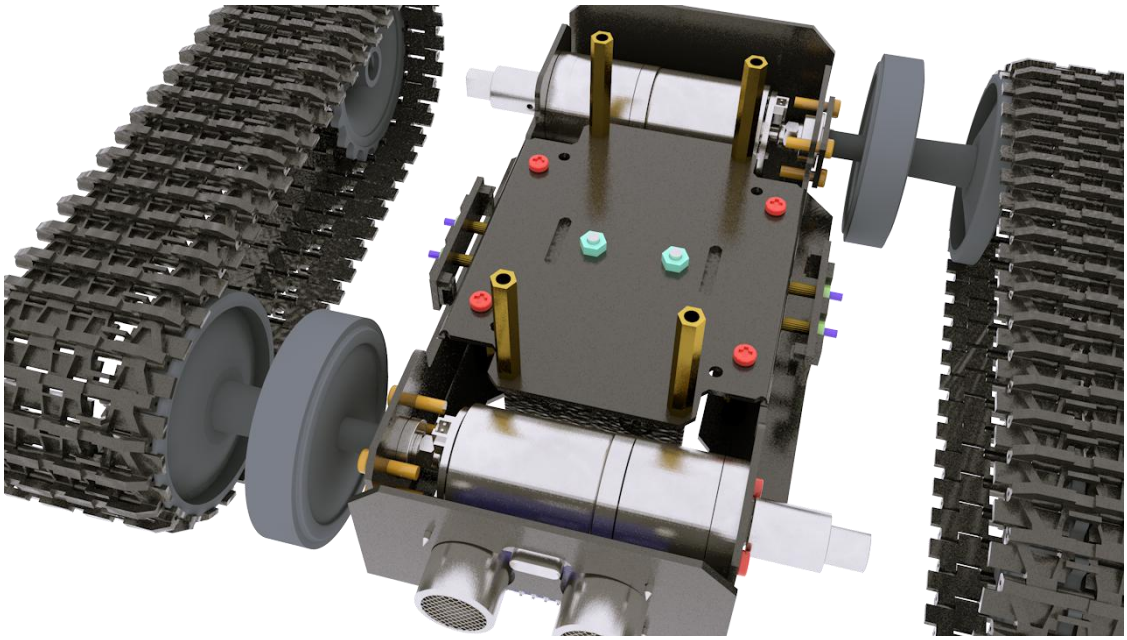
Use the same method to install the bearings and wheels on the other side.



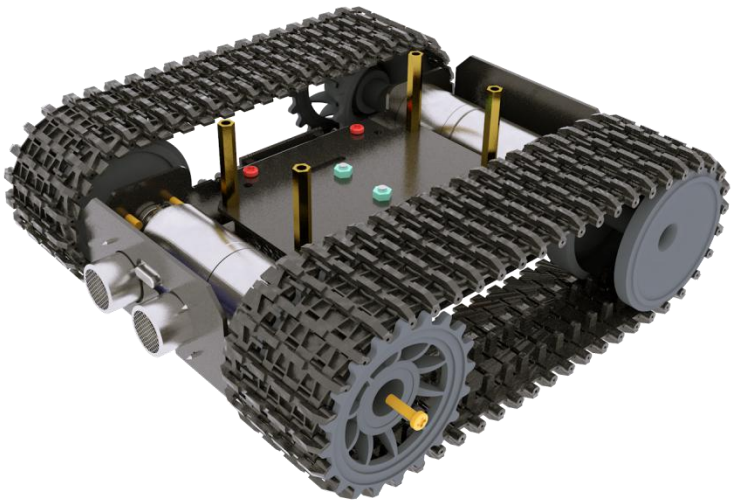
Install the motor and the coupling. The coupling screws need to be tightened. Use M3X4 screws to fix the motor to the base.



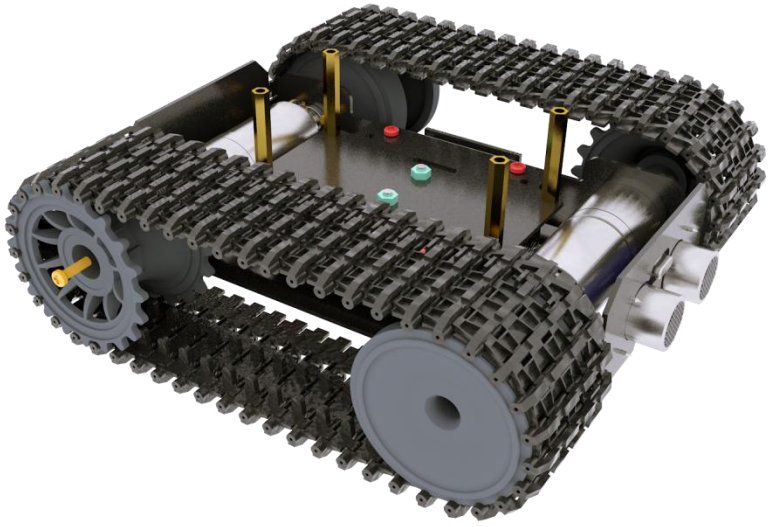
Use the same method to install the motor on the other side.



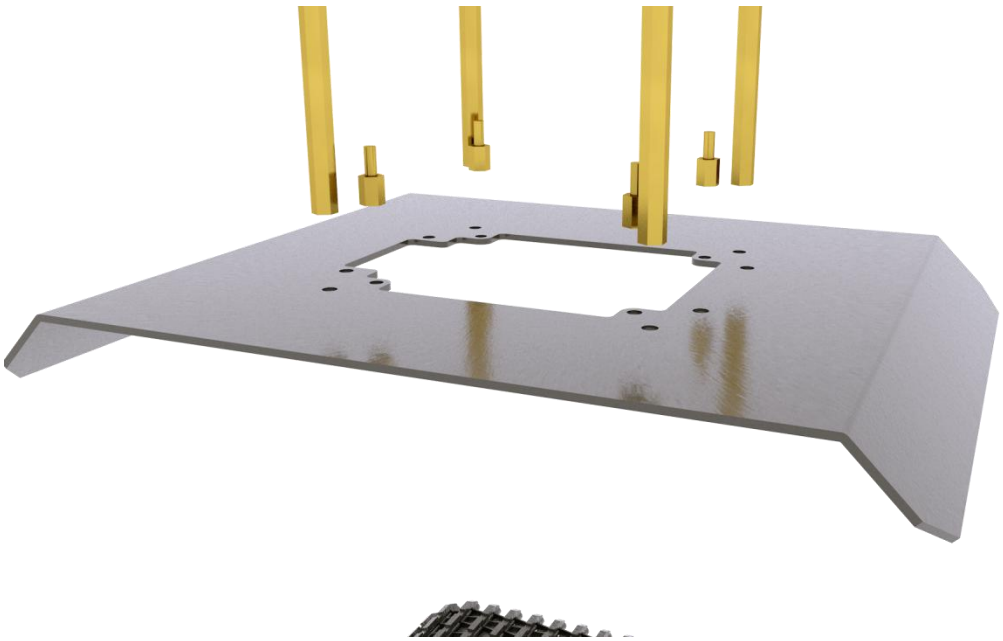
Install tracks on both sides.



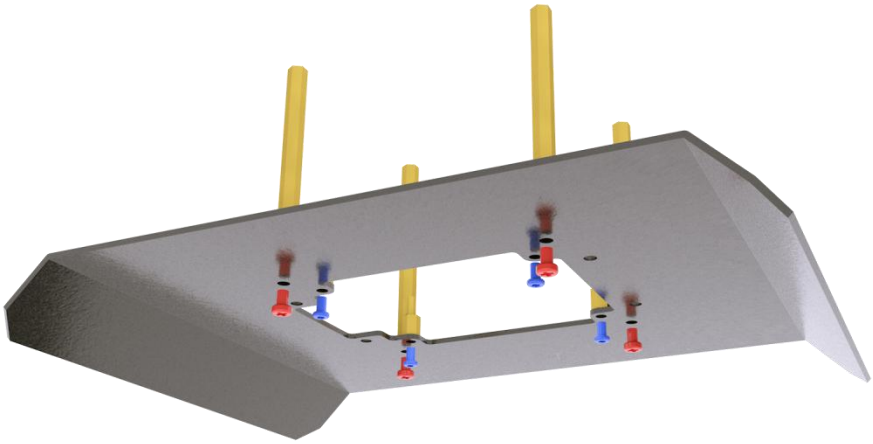
Use M3X10 screws to fix the wheel to the coupling.



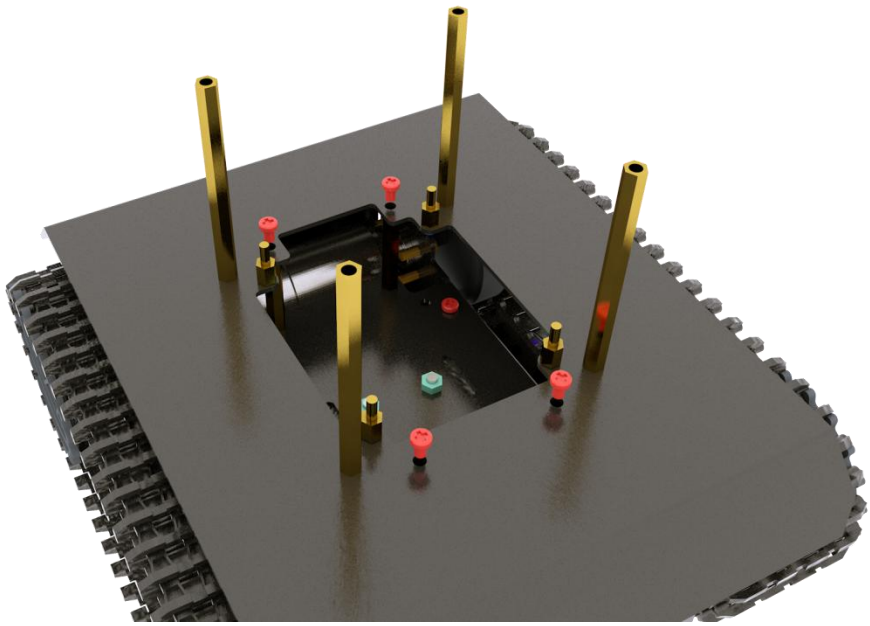
The installation method is the same on the other side.



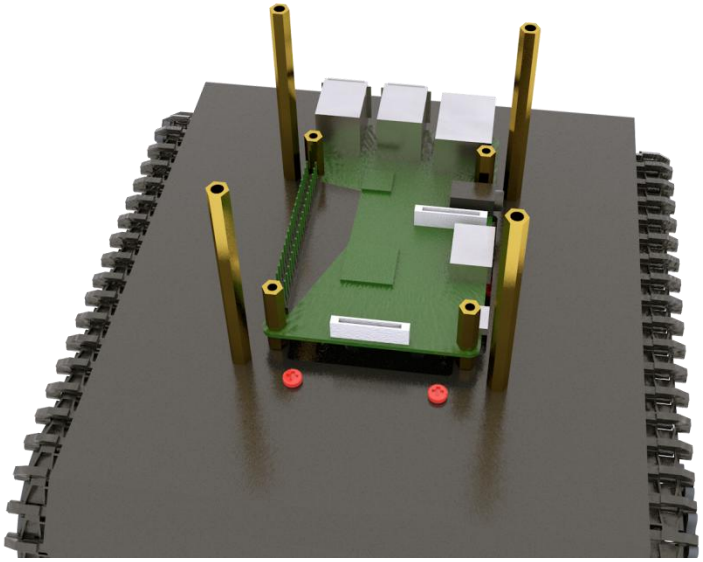
Use M3X4 screws and M2.5X4 screws to fix the M3X60 copper posts and M2.5X6+6 copper posts on the upper section. The specific fixing position is shown in the figure below.



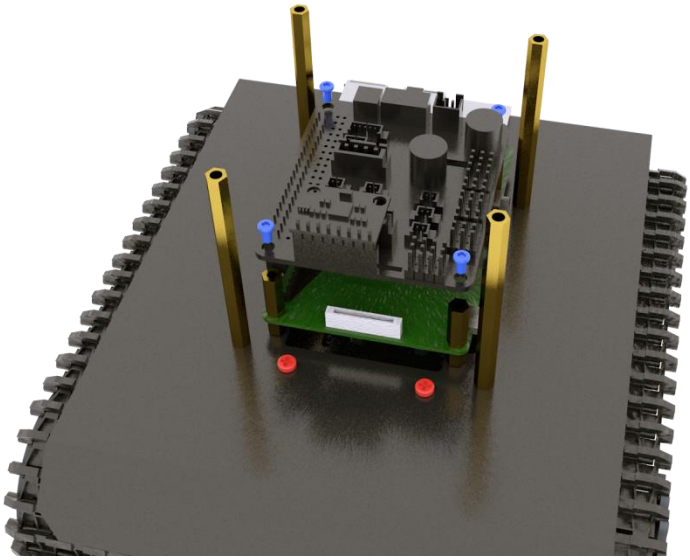
The installation location is shown in the figure.



Use M3X4 screws to fix the upper plate to the base.

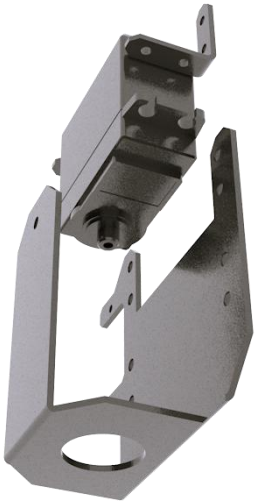


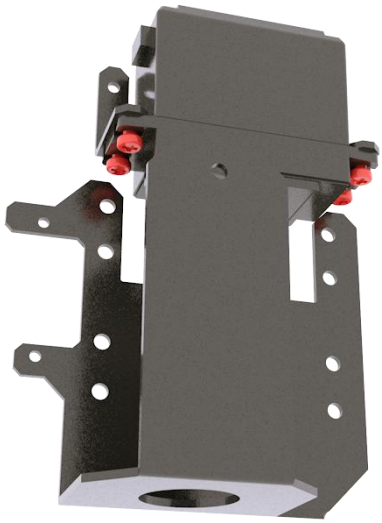
Use M2.5X14 copper posts to fix the Raspberry Pi to the base. At this time, you need to connect the camera cable to the Raspberry Pi.



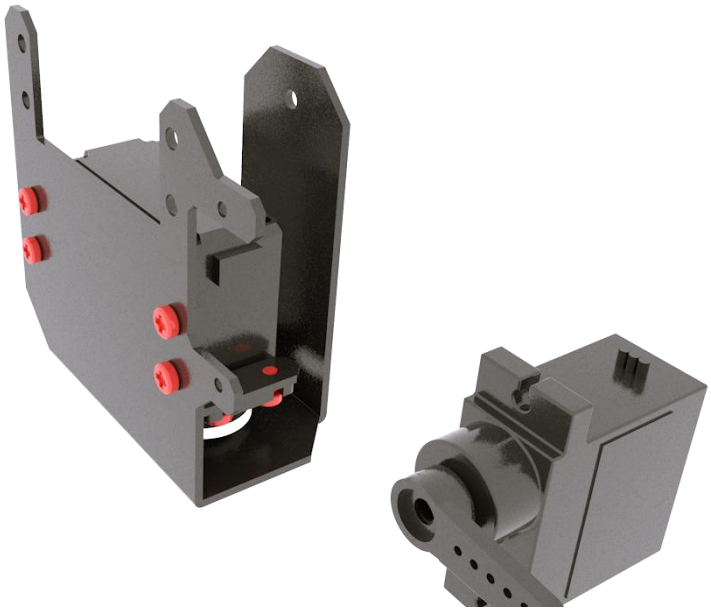
Use M2.5X4 screws to fix the Robot HAT driver board to the Raspberry Pi.

10.4.4 PTZ assembly

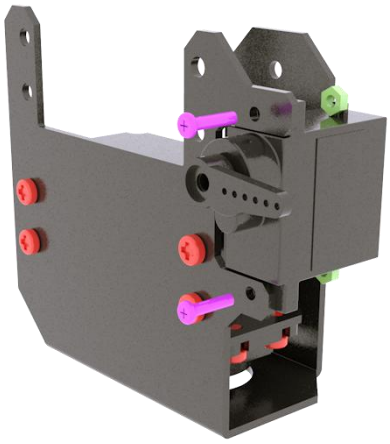




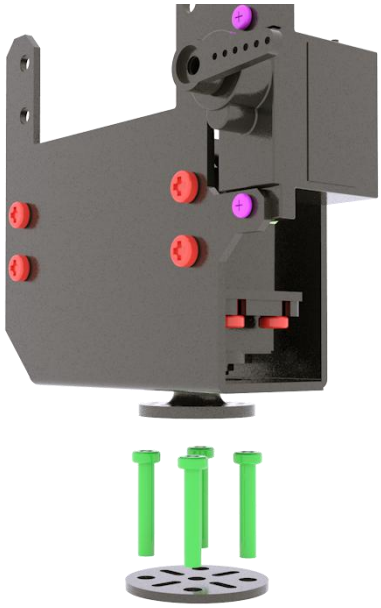
Use M3X4 screws to fix the servo to the servo bracket as shown in the two pictures above. The servo bracket has tapped threads, so no nuts are needed for installation here.



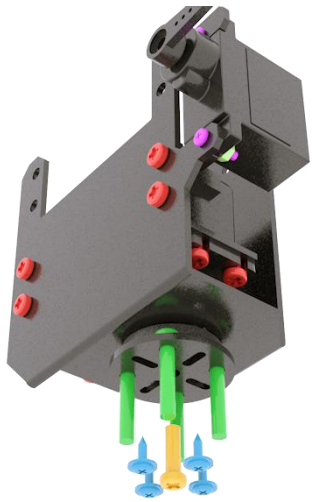
Use M3X4 screws to fix the servo bracket to the PTZ base.



Use M2X8 screws and M2 nuts to fix the AD002 servo to the base of the PTZ.



Install the round rocker arm of the large metal servo, and pierce the M3X16 screws through the round aluminum alloy parts shown below.



Use M1.7X6X6 self-tapping screws and M3X10 screws to fix the round aluminum alloy parts to the servo rocker arm and servo.

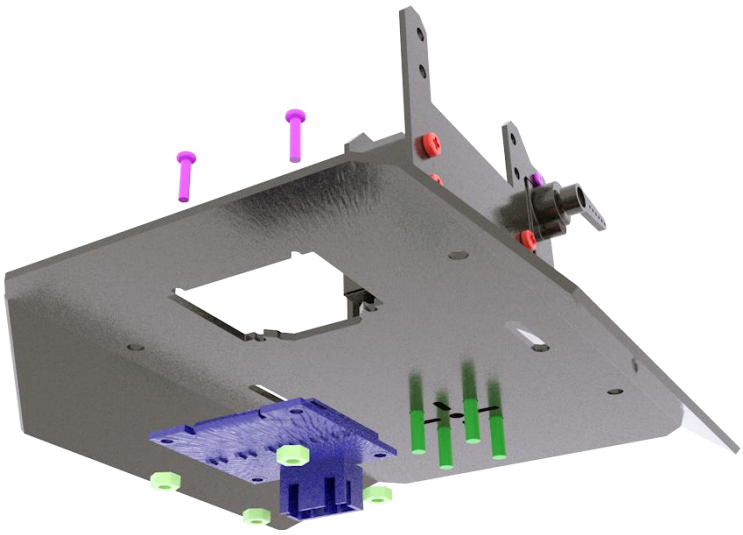
【Attention】:

(1) When installing the M3X10 screw, the M3X10 screw must be aligned with the center point of the round aluminum alloy part and be relatively vertical, so that the M3X10 screw can be rotated smoothly.

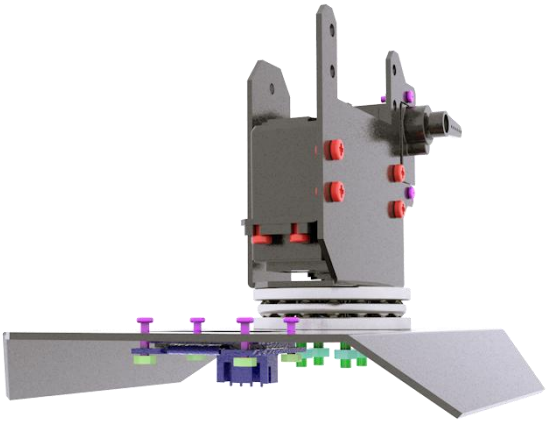
(2) If the M3X10 screw cannot be rotated in, it may be that you did not align the center point of the vertical round aluminum alloy part when installing the M3X10 screw, which caused the M3X10 screw to have an inclined angle when installed, and it would not be able to rotate in. For slipping, we recommend that you use a flat-blade screwdriver to remove the M3X10 screw, clean the thread of the M3X10 screw, and then reinstall the M3X10 screw. The M3X10 screw must be aligned with the center point of the round aluminum alloy part and relatively vertical.

Then install a round aluminum alloy part, and then install 51105 thrust bearing.

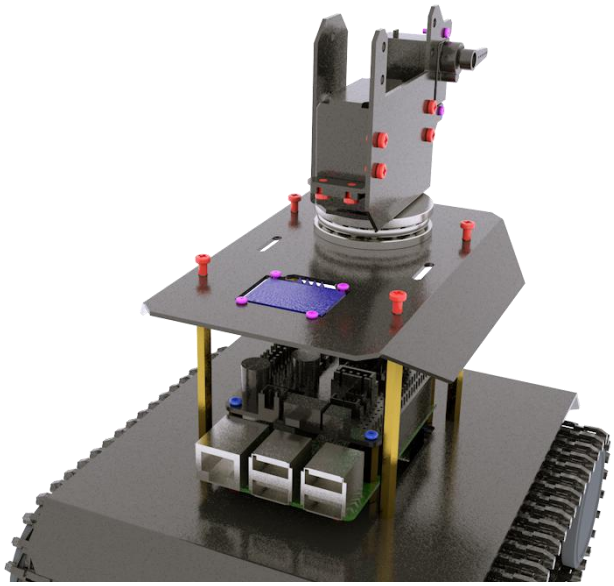
Install the 51105 thrust bearing. Note here that the thrust bearing consists of three separate parts. Subsequent assembly will compress the thrust bearing.



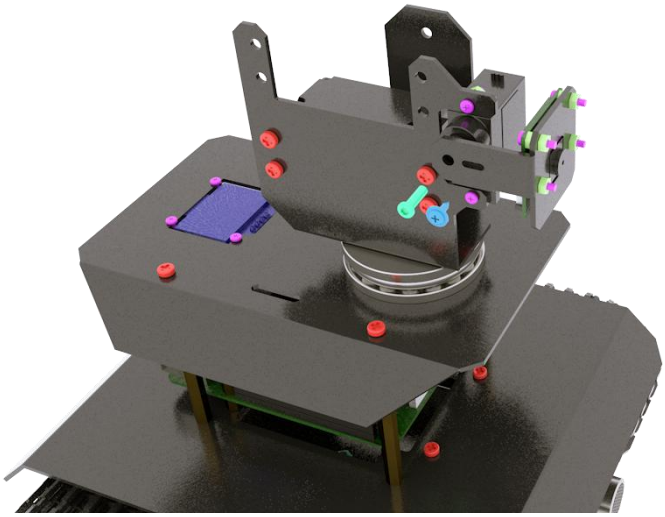
Install the PTZ base assembly to the section below the PTZ. Use M2X8 screws and M2 nuts to install the OLED screen.



Use the M3 nut to fix the PTZ base assembly to the lower section. The side view is shown in the figure above. After tightening the nut, the thrust bearings are fixed together and the PTZ base assembly can rotate normally.

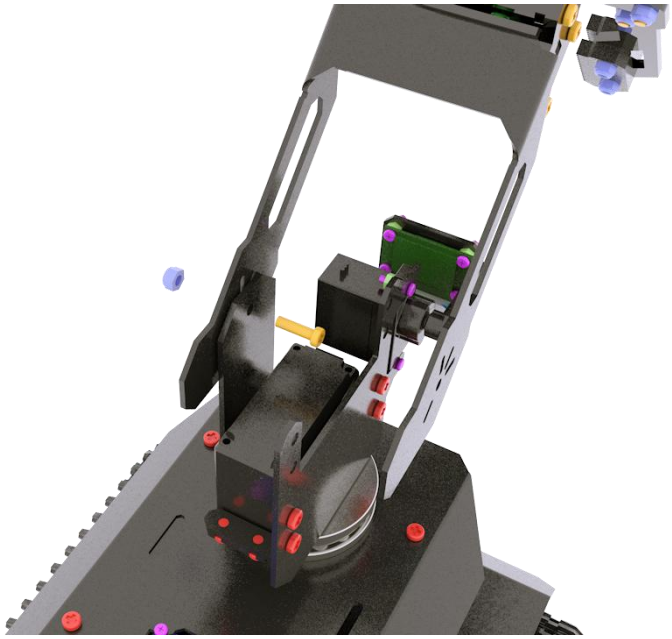


Use M3X4 screws to fix the upper plate assembly to the body assembly. Before performing this step, you need to connect the cable. Refer to the wiring method chapter for details.



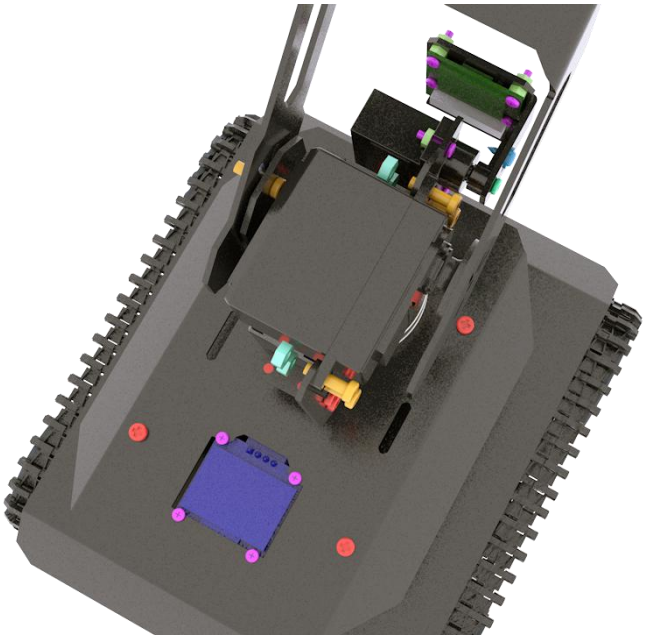
Use 4 M2X8 screws and 8 M2 nuts to install the Raspberry Pi camera to the aluminum alloy camera bracket.

Use M2.5X8 screws and M1.7X6X6 self-tapping screws to fix the aluminum alloy camera bracket to the servo rocker arm and servo.



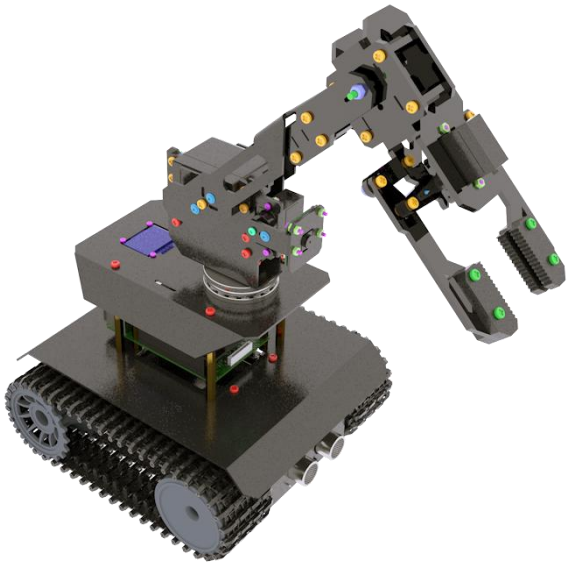
Use M3X10 screws and M3-LOCK lock nuts to fix the robot arm to the base of the PTZ. Note here that the lock nuts cannot be tightened too much, otherwise it will affect the rotation of the robot arm assembly.

After the installation is complete, as shown above.



Use M3X10 screws and M3 nuts to fix the servo to the base of the PTZ.

Use M3X10 screws and M1.7X6X6 self-tapping screws to connect the robot arm to the PTZ base.



The assembly is complete.

10.4.5 Wiring method

●If you don't know how to connect the camera, you can refer to the official documentation of the Raspberry Pi camera.

●The motor on the left is connected to the MOTOR-B port; the motor on the right is connected to the MOTOR-A port.

●The bottom servo that controls the pan/tilt component to swing left and right is connected to the Port 0 servo port of the driver board.

●There are two servos that control the mechanical arm to swing up and down. One is large and one is small. The large servo fixed on the PTZ assembly is connected to the Port 1 servo port, and the small servo fixed in the middle of the mechanical arm is connected to Port 2 Servo port.

●The servo that controls the gripping action of the chuck is connected to the Port 3 servo port.

●The servo that controls the tilting motion of the camera is connected to the Port 4 servo port.

●The ultrasound module is connected to the Ultrasonic port using a 4pin cable. Never connect the ultrasound to the IIC port, as the ultrasound module will be permanently damaged.

●The WS2812 light bar is connected to the WS2812 port using a 3pin cable. Note here that the signal to control the light is sent by the Raspberry Pi and sent to the WS2812 light bar by the 3PIN line by the driver board. A section with white stripes drawn from the light bar needs to be connected (The text of the interface on the back of this end is marked with IN). Take it out from the other end of the light bar (the interface is marked OUT) and connect it to the input end of the next light bar by the 3PIN line (the white stripe is drawn and the interface is marked with IN).

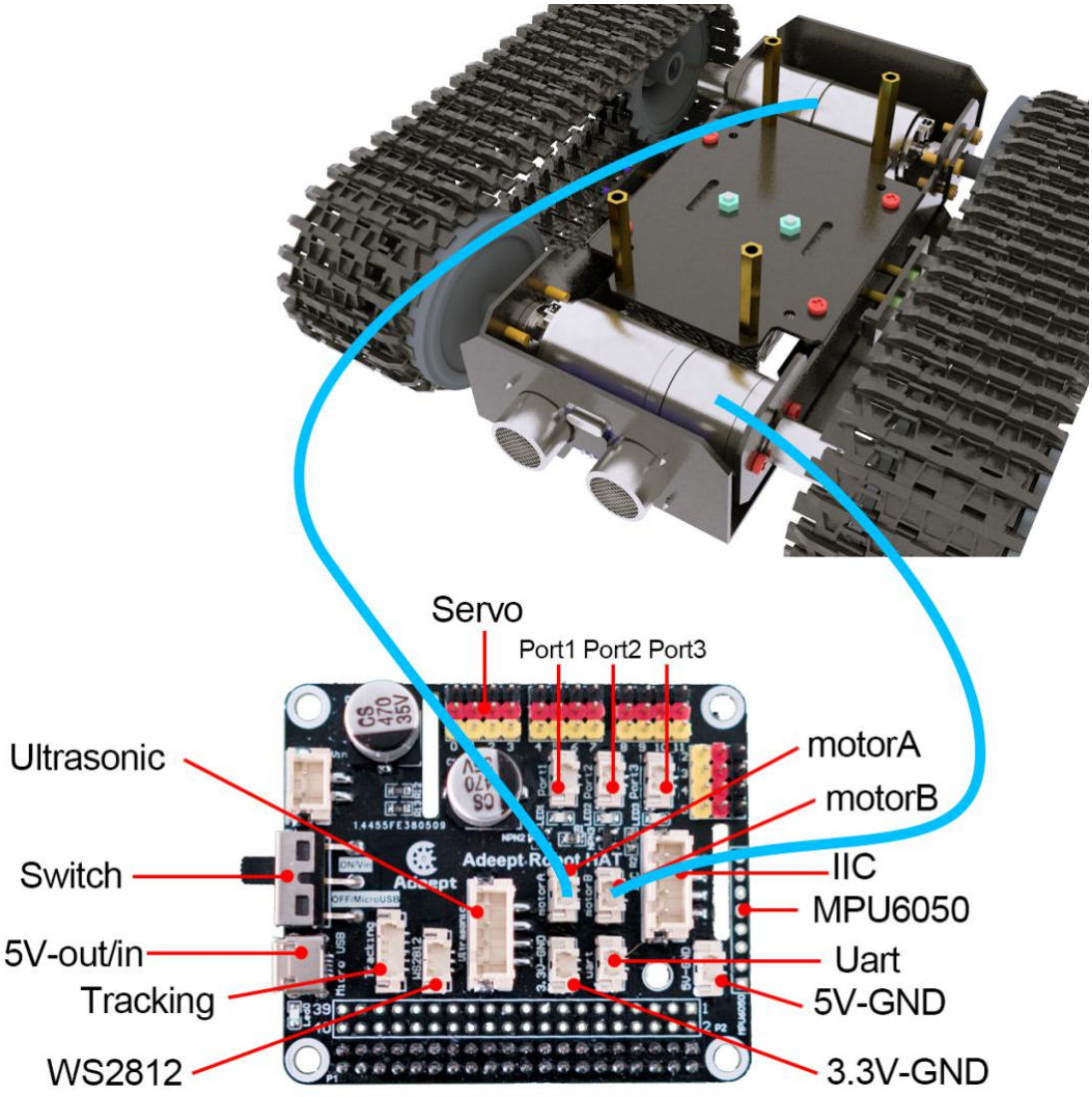
●The OLED screen is connected to the IIC port of the driver board via a 4PIN cable.

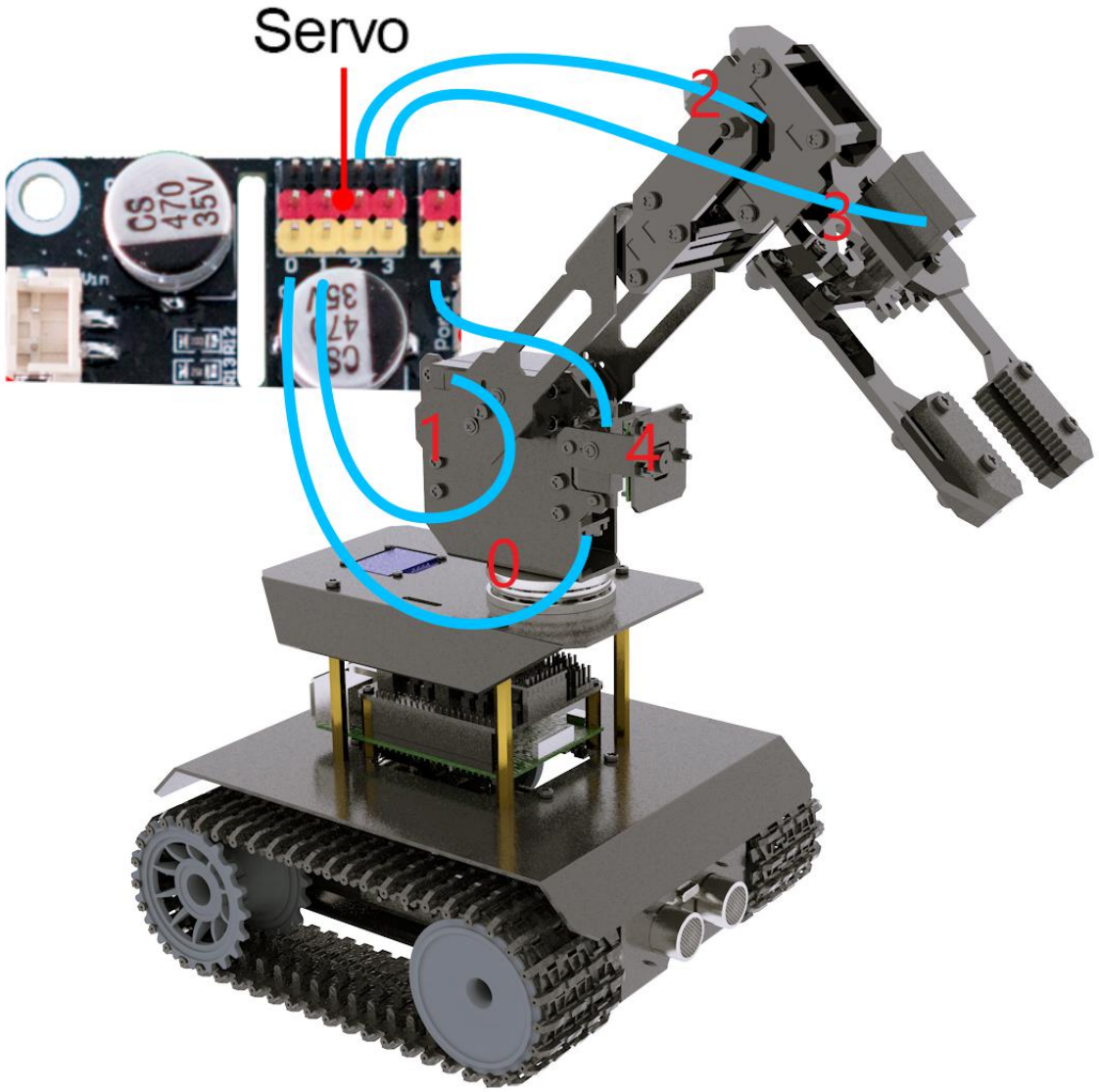
●MPU6050 is inserted into the 8PIN Dupont port on the front of the driver board.

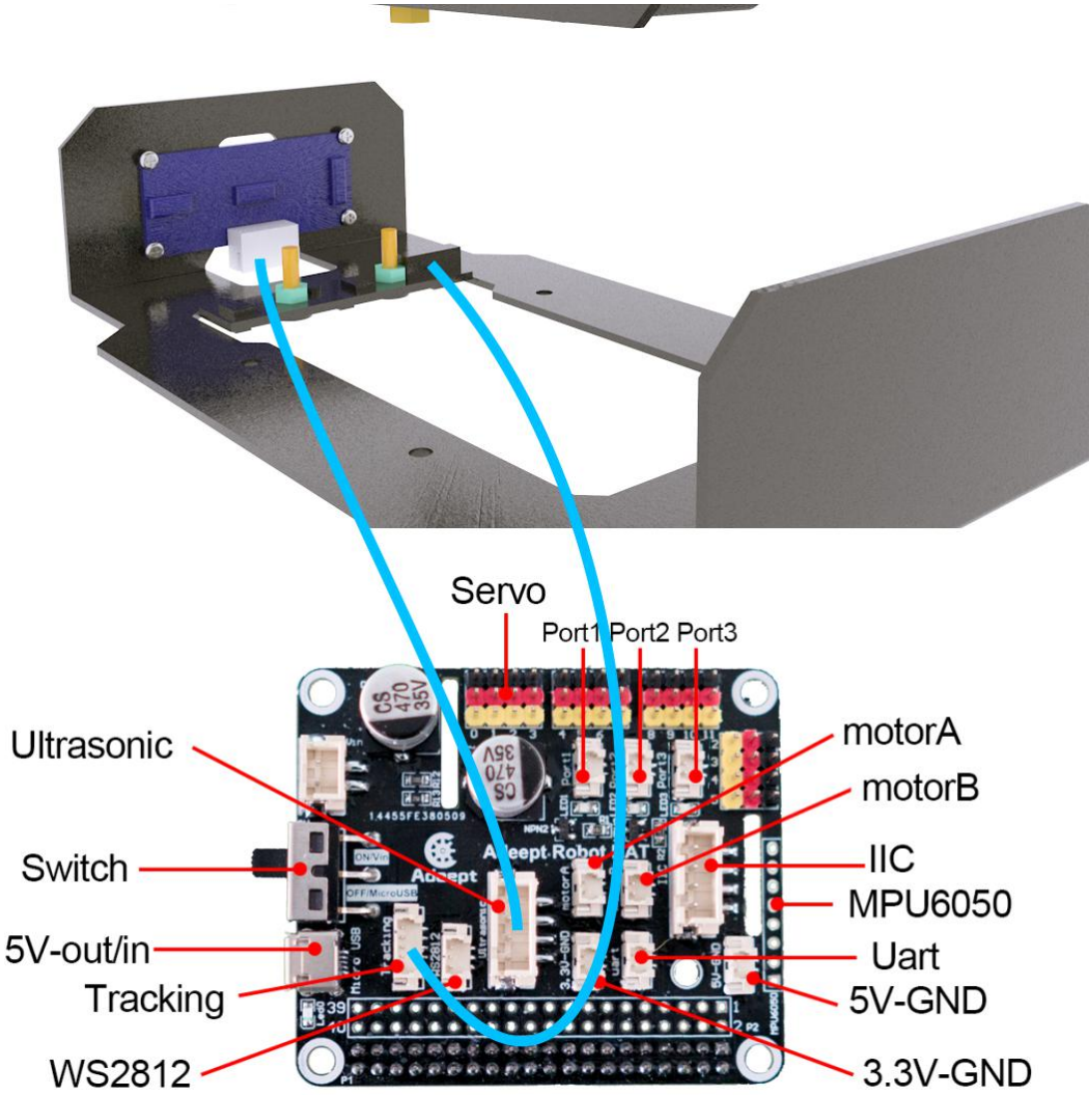
●The power supply is connected to the VIN port.

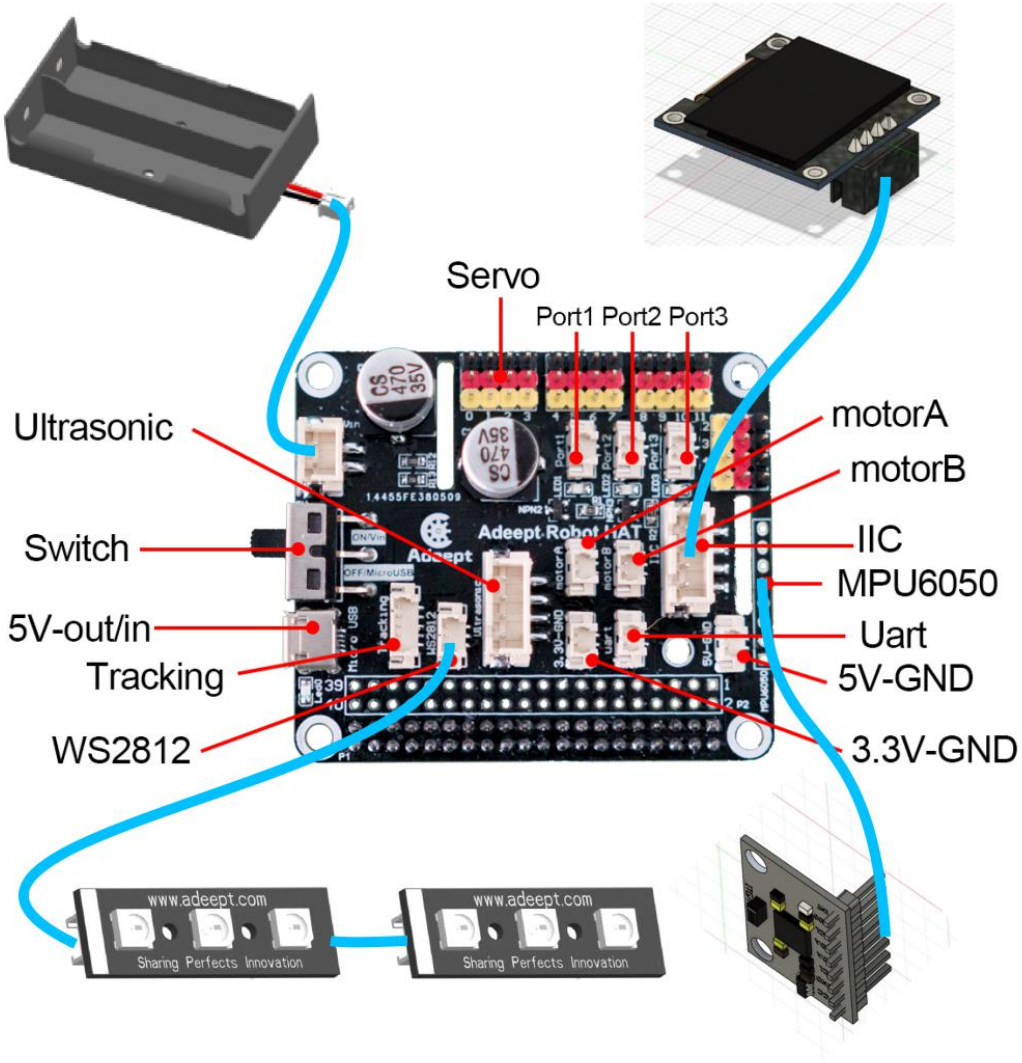
●The tracking module is connected to the Tracking port via a 5PIN wire.

•


Servo
Port1 Port2 Port3
Ultrasonic
Switch
5V-out/in
Tracking
WS2812
motorA
motorB
IIC
MPU6050
Uart
5V-GND
3.3V-GND


Servo
2
3
1
4
0


Servo
Port1 Port2 Port3
Ultrasonic
Switch
5V-out/in
Tracking
WS2812
motorA
motorB
IIC
MPU6050
Uart
5V-GND
3.3V-GND


Servo
Port1 Port2 Port3
Ultrasonic
Switch
5V-out/in
Tracking
WS2812
motorA
motorB
IIC
MPU6050
Uart
5V-GND
3.3V-GND
www.adeept.com
Sharing Perfects Innovation
www.adeept.com
Sharing Perfects Innovation

11.Controlling RaspTank Pro for Infrared Line Tracking

11. 1 Infrared line tracking module

●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.

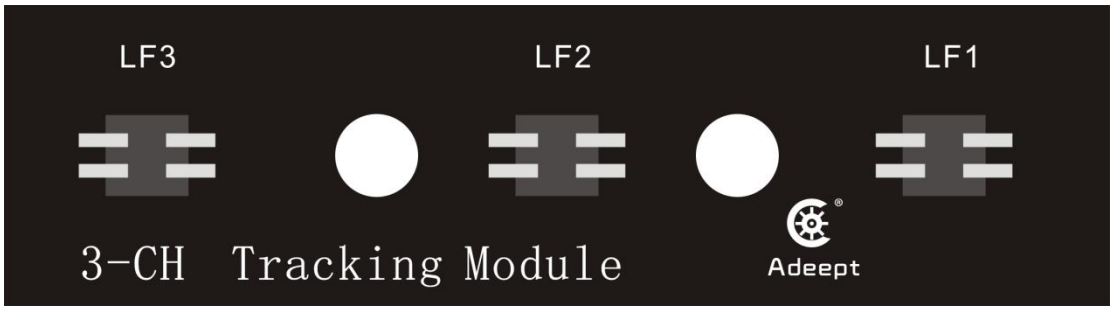
●Our program defaults to finding black lines on a white background (reflecting infrared light) (not reflecting infrared light).

●Before using the three-channel infrared line patrol module, you need to connect it to the Tracking interface on Robot 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.

On the white paper "road" with black lines drawn, because the black lines and white paper have different reflection coefficients of light, the "road"-the black line can be judged according to the intensity of the received reflected light. In the Tracking module, a more common detection method-infrared detection method is used.

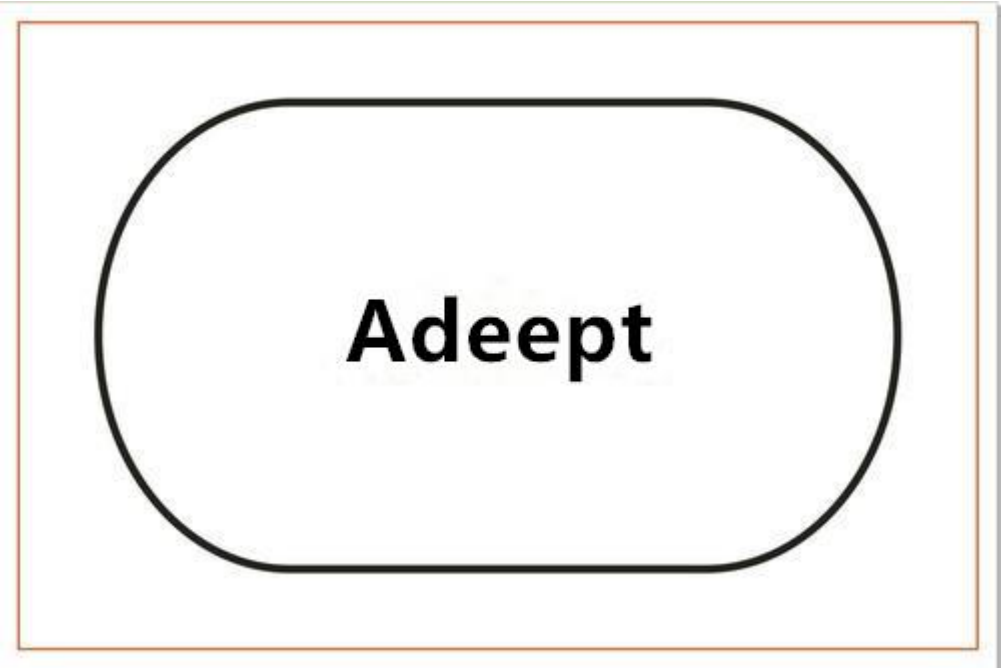
Infrared detection method uses infrared rays to have different reflection properties on different colored physical surfaces. When the car is running, the infrared light is continuously emitted to the ground. When the infrared light meets the white ground, the diffuse emission occurs, and the reflected light is received by the receiving tube installed on the car; if it encounters a black line, the infrared light is absorbed, and the car The receiver tube on the receiver cannot receive the signal.



11.2 Preparation

1. The assembled RaspTank Pro.

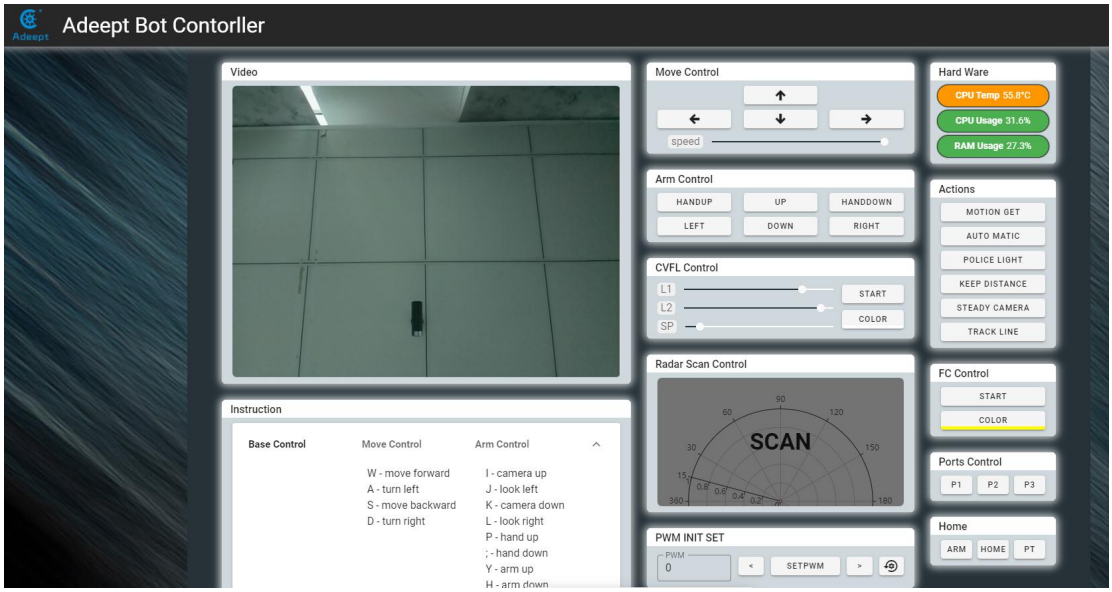
2. Make a patrol track.



11.3 Running line tracking program

1. Turn on RasTankPro, the boot time is about 1 minute (webServer.py will run automatically when the machine is turned on).

2. After RasTankPro is turned on, enter the IP address of your Raspberry Pi through the Google browser of your mobile phone or computer, and access port 5000, for example: 192.168.3.44:5000. The web controller will then be displayed on the browser.



1. Put the car on the ready-made patrol track.

2. Click "TRACK LINE", RasTank Pro starts to drive along the black line.

3. When you want to terminate the line tracking function, you can click "TRACK LINE" again.

4. The height of the line patrol module of RasTank Pro from the ground is about 14mm. When the line-following module cannot be used normally, please refer to: Line-following module adjustment tutorial. Tutorial link: https://www.adeept.com/learn/detail-49.html

11.4 Main code program

The complete code reference file functions.py.

```
1.  import RPi.GPIO as GPIO
2.  import time
1.  import move
3.
4.
5.  # The output pins of the hunting module
6.  line_pin_right = 19
7.  line_pin_middle = 16
```

```
line_pin_left = 20

Dir_forward   = 0
Dir_backward  = 1

left_forward  = 1
left_backward = 0

right_forward = 0
right_backward= 1

mark = 0
"""
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)

class Functions(threading.Thread):
    ...

    def trackLineProcessing(self):
        status_right = GPIO.input(line_pin_right)
        status_middle = GPIO.input(line_pin_middle)
        status_left = GPIO.input(line_pin_left)
        global mark
        if status_left ==0 and status_middle == 1 and status_right ==0:# (0 1 0)
            move.motor_left(1, left_forward, 80)    # move.motor_left(status, left_forward, speed)   status:1 means action, 0 means end. left_forward:Left motor forward. speed: motor speed.
            move.motor_right(1, right_forward, 80)# right_forward: Right motor forward
            mark = 1

        elif status_left ==1 and status_middle == 1 and status_right ==0:# (1 1 0 )
            if mark !=2:
                move.motor_left(1, left_backward, 80)   # left_backward: Left motor backward
                move.motor_right(1, right_backward, 80) # right_backward: Right motor backward
                time.sleep(0.03)
            move.motor_left(1, left_forward, 70)
            move.motor_right(1, right_forward, 100)
            mark = 2

        elif status_left ==1 and status_middle == 0 and status_right ==0:#(1 0 0)
```

```
        if mark !=3:
            move.motor_left(1, left_backward, 80)
            move.motor_right(1, right_backward, 80)
            time.sleep(0.03)
        move.motor_left(1, left_forward, 0)
        move.motor_right(1, right_forward, 100)
        time.sleep(0.02)
        mark = 3

    elif  status_left ==0 and status_middle == 1 and status_right ==1:# (0 1 1)
        if mark !=4:
            move.motor_left(1, left_backward, 80)
            move.motor_right(1, right_backward, 80)
            time.sleep(0.03)
        move.motor_left(1, left_forward, 100)
        move.motor_right(1, right_forward, 70)
        mark = 4

    elif status_left ==0 and status_middle == 0 and status_right ==1:# (0 0 1)
        if mark !=5:
            move.motor_left(1, left_backward, 80)
            move.motor_right(1, right_backward, 80)
            time.sleep(0.03)
        move.motor_left(1, left_forward, 100)
        move.motor_right(1, right_forward, 0)
        time.sleep(0.02)
        mark = 5

    else:
        if mark ==0 :
            move.motor_left(1, left_forward, 80)
            move.motor_right(1, right_forward, 80)
        elif mark == 1:

            move.motor_left(1, left_forward, 80)
            move.motor_right(1, right_forward, 80)
        elif mark == 2 or mark == 3:              # (1 0 0)

            move.motor_left(1, left_forward, 0)
            move.motor_right(1, right_forward, 100)
            time.sleep(0.03)
        elif mark == 4 or mark == 5:

            move.motor_left(1, left_forward, 100)
            move.motor_right(1, right_forward, 0)
            time.sleep(0.03)
```

```
70.
71.         time.sleep(0.1)
22.
23. if __name__ == '__main__':
24.     setup()
25.     while 1:
26.         Functions().trackLineProcessing()
```

• When your project needs to use the line tracking function, you don’t need to rewrite the above code, just copy functions.py and move.py in the server folder of the robot program to the same as your own project In the folder of, then use the following code to use the line tracking function:

```
1.  import functions
2.
3.  functions.setup()
4.
5.  while 1:
6.      functions.trackLineProcessing()
```

•The reason why you need to import move.py is because functions.py needs to use t he methods in move.py to control the movement of the robot. If you use other methods to control the movement of the robot, then you only need to rewrite the relevant code in fun ctions.py That's it.

12. Using Multithreading to Make Police Lights and Breathing Lights

12.1 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 non-blocking control method in the steering gear control, which is the control method of the steering gear encapsulated by multi-threading

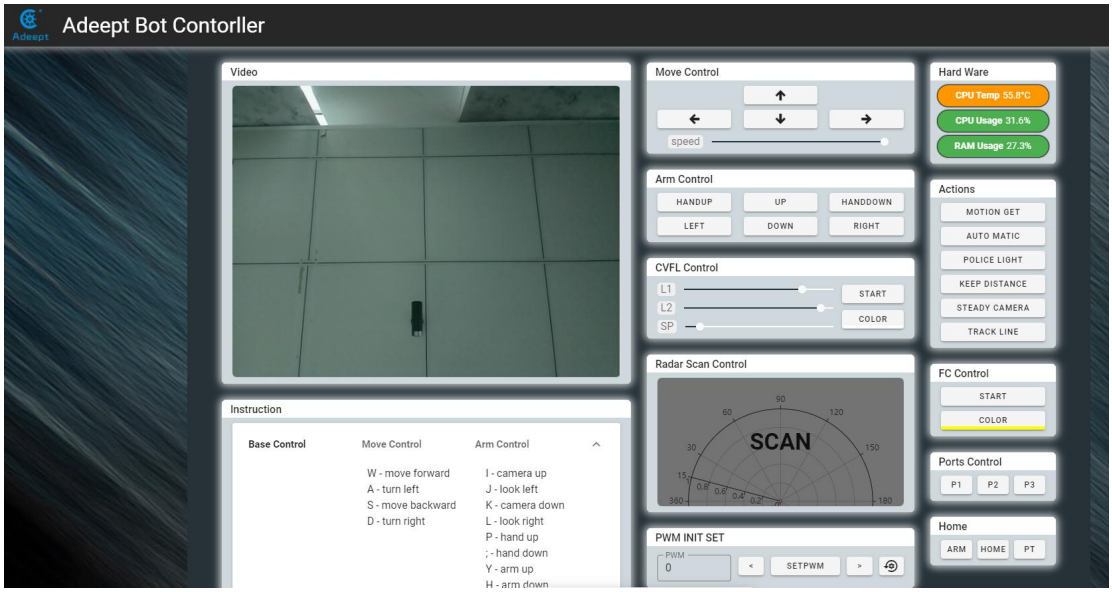
●We use Python's threading library to provide thread-related work. Threads are the smallest unit of work in an application. The current version of Python does not provide multi-thread priority, thread group, and thread cannot be stopped, suspended, resumed and interrupted.

12.2 Realizing the WS2812 LED lighting effect with multithreading

12.2.1 Running the line tracking program

1. Turn on RasTank Pro, the boot time is about 1 minute.

2. After RasTank Pro is turned on, enter the IP address of your Raspberry Pi through the Google browser of your mobile phone or computer, and access port 5000, for example: 192.168.3.44:5000. The web controller will then be displayed on the browser.



3. Click "POLICE LIGHT", RaspTank Pro starts to flash lights of different colors.

4. When you want to stop the warning light function, click "POLICE LIGHT" again.

12.3 Main code program

The complete code reference file robotLight.py.

12.3.1 Realization of warning 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 to avoid wasting CPU resources.

• Here we use the wait() method to block the thread. From the realization of the requirement of controlling the thread, the code and comments are as follows:

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


"""
Use the Threading module to create a thread, inherit directly from threading.Thread, and then rewrite the __init__ method and run method'''
class RobotLight(threading.Thread):
    def __init__(self, *args, **kwargs):
        """
        Some LED lamp settings are initialized here
        '''
        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 color channels of RGB, there is 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

        """
        Mode variable,'none' will cause the thread to block and hang, and the light will not change;
        'police' is the police light mode, red and blue flashing alternately;
        The'breath' breathing light can be set to a 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)
```

```
45.         self.__flag = threading.Event()
46.         self.__flag.clear()
47.
48.     # Define functions which animate LEDs in various ways.
49.     def setColor(self, R, G, B):
50.         """
51.         Set the color of all lights
52.         '''
53.         color = Color(int(R),int(G),int(B))
54.         for i in range(self.strip.numPixels()):
55.             self.strip.setPixelColor(i, color)
56.             self.strip.show()
57.
58.
59.     def setSomeColor(self, R, G, B, ID):
60.         """
61.         Set the color of a few lights, and the ID is an array of the serial numbers of the lights
62.         '''
63.         color = Color(int(R),int(G),int(B))
64.         #print(int(R),'  ',int(G),'  ',int(B))
65.         for i in ID:
66.             self.strip.setPixelColor(i, color)
67.             self.strip.show()
68.
69.
70.     def pause(self):
71.         """
72.         Call this function, set __flag to False, and block the thread
73.         '''
74.         self.lightMode = 'none'
75.         self.setColor(0,0,0)
76.         self.__flag.clear()
77.
78.
79.     def resume(self):
80.         """
81.        Call this function, set __flag to True, and start the thread
82.         '''
83.         self.__flag.set()
84.
85.
86.     def police(self):
87.         """
88.         Call this function to turn on the warning light mode
89.         '''
90.         self.lightMode = 'police'
```

```
        self.resume()


    def policeProcessing(self):
        """
        The concrete realization of the warning 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 flashing 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):
                """
                    To call this function to turn on the breathing light mode, you need to enter three parameters,
                which are the brightness of the three color channels of RGB, as the color when the brightness of the breathing
                light is maximum
                '''
                self.lightMode = 'breath'
                self.colorBreathR = R_input
                self.colorBreathG = G_input
                self.colorBreathB = B_input
                self.resume()


            def breathProcessing(self):
                """
                Specific implementation method of breathing light
```

```
135.            '''
136.            while self.lightMode == 'breath':
137.                """
138.                All lights gradually brighten
139.                '''
140.                for i in range(0,self.breathSteps):
141.                    if self.lightMode != 'breath':
142.                        break
143.                    self.setColor(self.colorBreathR*i/self.breathSteps,
144.                                  self.colorBreathG*i/self.breathSteps,
145.                                  self.colorBreathB*i/self.breathSteps)
146.                    time.sleep(0.03)
147.                """
148.                 All lights gradually dim
149.                '''
150.                for i in range(0,self.breathSteps):
151.                    if self.lightMode != 'breath':
152.                        break
153.                    self.setColor(self.colorBreathR-(self.colorBreathR*i/self.breathSteps),
154.                                  self.colorBreathG-(self.colorBreathG*i/self.breathSteps),
155.                                  self.colorBreathB-(self.colorBreathB*i/self.breathSteps))
156.                    time.sleep(0.03)
157.
158.
159.        def lightChange(self):
160.            """
161.            This function is used to select tasks to execute
162.            '''
163.            if self.lightMode == 'none':
164.                self.pause()
165.            elif self.lightMode == 'police':
166.                self.policeProcessing()
167.            elif self.lightMode == 'breath':
168.                self.breathProcessing()
169.
170.
171.        def run(self):
172.            """
173.            Functions for multi-threaded tasks
174.            '''
175.            while 1:
176.                self.__flag.wait()
177.                self.lightChange()
178.                pass
179.
180.
```

```
if __name__ == '__main__':
    RL=RobotLight() # 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 warning light mode and stop after 15 seconds
    '''
    RL.police()
    time.sleep(15)
    RL.pause()
```

12.3.2 Using warning lights or breathing lights in other projects

• When your project needs to use LED warning lights or breathing lights, you don't need to rewrite the above code, just copy robotLight.py in the server folder of the robot program to the same as your own project 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 the 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 warning light mode and stop after 15 seconds
'''
RL.police()
time.sleep(15)
RL.pause()
```

13. Controlling RaspTank Pro to Automatically Avoid Obstacles

13.1 Introduction to Automatic Obstacle Avoidance

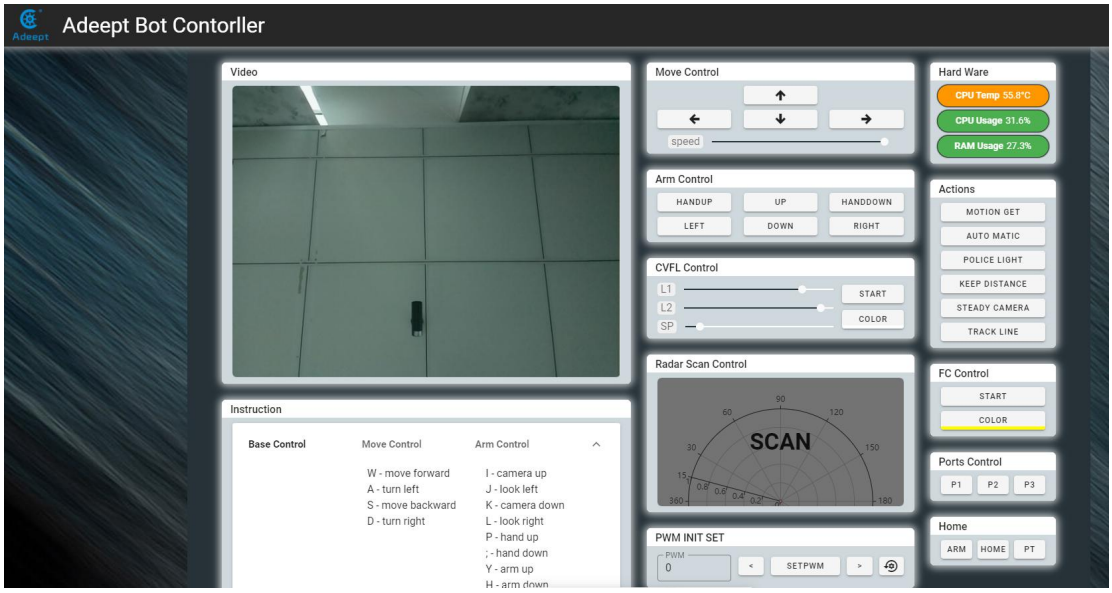
Since the ultrasonic module of this product can only move up and down with the camera, and the left and right movement can only move with the body left and right, and cannot move left and right relative to the body, so the obstacle avoidance function of this robot is relatively simple, as long as there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

13.2 Turning on automatic obstacle avoidance function

13.2.1 Running automatic obstacle avoidance program

1. Turn on RaspTank Pro, the boot time is about 1 minute.

2. After RaspTankPro is turned on, enter the IP address of your Raspberry Pi with the Google browser of your mobile phone or computer, and access port 5000, for example: 192.168.3.44:5000 (see lesson 2 for detailed steps). The web controller will then be displayed on the browser.



3. Place the car on the prepared patrol track.

4. After clicking "AUTO MATIC", the robot will automatically avoid obstacles when encountering obstacles. .

5. When you want to terminate the automatic obstacle avoidance function, you can click "AUTO MATIC" again.

13.3 Main code program

The complete code reference file functions.py.

• The automatic obstacle avoidance function is implemented based on the ultrasonic distance measurement module, so before writing your project, first copy ultra.py and move.py to the same file directory as your project, and then use the following code To use the automatic obstacle avoidance function.完整代码参考文件 functions.py。

```
1.  import ultra
2.  import move
3.  import time
4.
5.  """
6.  Initialize the motor and set the obstacle avoidance distance
7.  '''
8.  move.setup()
9.  rangeKeep = 0.5
10.
11. def automaticProcessing():
12.     if rangeKeep/3 > ultra.checkdist():
13.         """
14.         If the distance detected by the ultrasound is less than 1/3 of the obstacle avoidance distance, go backwards
15.         '''
16.         move.move(100, 'backward', 'no', 0.5)
17.
18.     elif rangeKeep > ultra.checkdist():
19.         """
20.         If the distance detected by the ultrasound is less than the obstacle avoidance distance, turn left, or turn right
21.         '''
22.         move.move(100, 'no', 'left', 0.5)
23.
```

```
    else:
        """
        If the distance detected by the ultrasound is greater than the obstacle avoidance distance, move forward
        '''
        move.move(100, 'forward', 'no', 0.5)
        time.sleep(0.1)

while 1:
    automaticProcessing()
```

• The above code calls the move() function in move.py to control the movement of the robot, move(speed, direction, turn, radius), where speed is used to set the speed of the robot, the maximum value is 100, and the minimum value is 0. This speed regulation method actually changes the voltage of the motor port through PWM. In the actual test, rewriting this value has little effect on the limit speed, but has a greater effect on the acceleration. When the value is too small, the motor will lose energy and load due to the deceleration mechanism. It may cause it to not produce enough torque to rotate.

•Direction is used to set the moving direction of the robot, which can be set to'forward','backward' or'no':

-When set to forward, the robot moves forward

-When set to backward, the robot moves backward

-When set to no, the robot will not move back and forth. At this time, if the turn variable is also no, the robot will stop moving

•Turn is used to set the rotation movement of the robot, which can be set to left, right or no:

-When set to left, if direction is no, the robot will turn left in place

-When set to left, if the direction is forward or backward, the robot will turn with a turning radius. For details, refer to the introduction of the radius parameter below.

-When set to right, if direction is no, the robot will turn right in place

-When set to right, if the direction is forward or backward, the robot will turn with a turning radius. For details, refer to the introduction of the radius parameter below.

-When set to no, if the direction is also no, the robot stops moving

•Radius is used to set the turning radius of the robot, which does not prevent the motor from stalling due to insufficient torque. The radius parameter in our product program does not play a practical role. We provide this interface. If your ground is very smooth, you can rewrite it. The program uses the radius parameter to increase the expressiveness of the robot:

The value of radius can be set to 0-1. It actually changes the speed difference between the fast and slow motors of the steering movement with the turning radius. The speed of the slower motor will be multiplied by the coefficient of radius to slow it down. Principle The voltage of the motor is adjusted by PWM, so if this value is set incorrectly, the motor with a lower speed may be blocked when the robot is turning.

14. How to Open the Real-time Video Screen of RaspTank Pro

●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

Video Streaming Demonstration

●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

```
'''
f os.environ.get('CAMERA'):
    Camera = import_module('camera_' + os.environ['CAMERA']).Camera
lse:
    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')
```

```
24.
25.
26. def gen(camera):
27.     """Video streaming generator function."""
28.     while True:
29.         frame = camera.get_frame()
30.         yield (b'--frame\r\n'
31.                b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')
32.
33.
34. @app.route('/video_feed')
35. def video_feed():
36.     """Video streaming route. Put this in the src attribute of an img tag."""
37.     return Response(gen(Camera()),
38.                     mimetype='multipart/x-mixed-replace; boundary=frame')
39.
40.
41. if __name__ == '__main__':
42.     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:

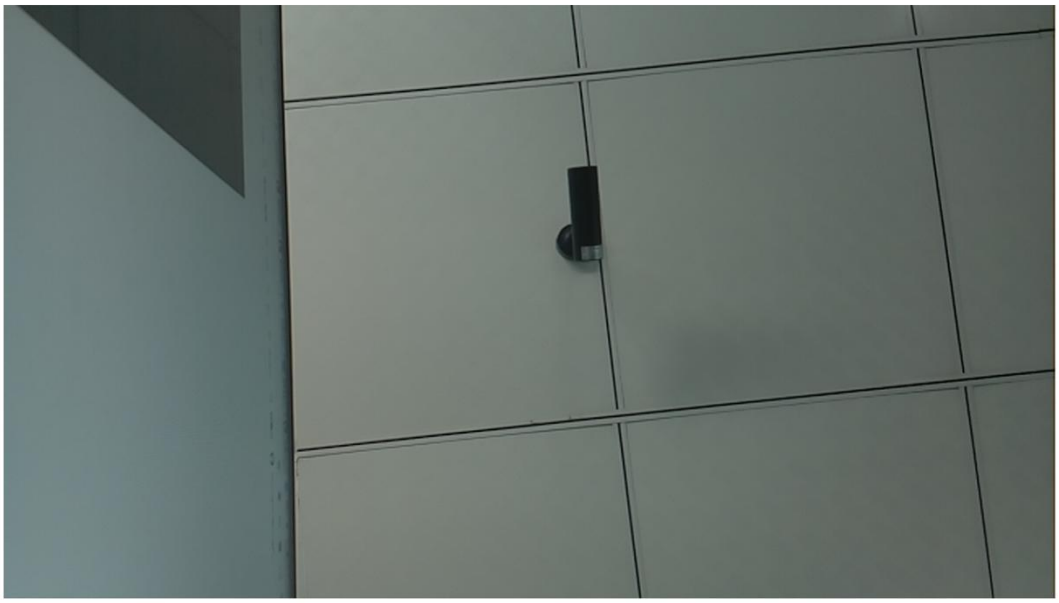
```
sudo app.py
```

●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.

Video Streaming Demonstration



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

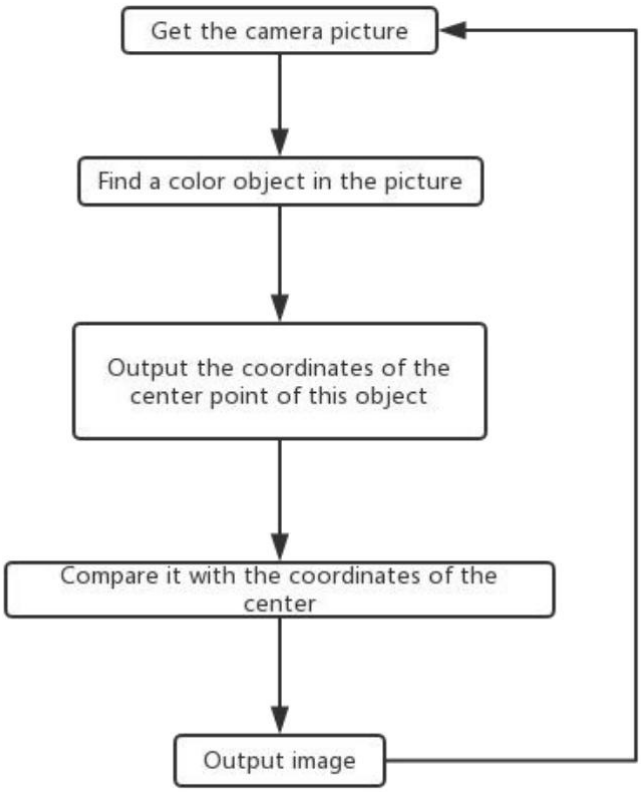
15. OpenCV Function

15.1 The principle of using multithreading to process video frames

• The OpenCV function is based on the GitHub project flask-video-streaming. We changed the camera_opencv.py in this project to perform OpenCV related operations.

15.1.1 Single-threaded processing of video frames

•First, we introduce the process of single-threaded processing of video frames, starting from the simple, so that you will better understand why OpenCV processes video frames using multiple threads to operate, the process of single-threaded processing of video frames is as follows:

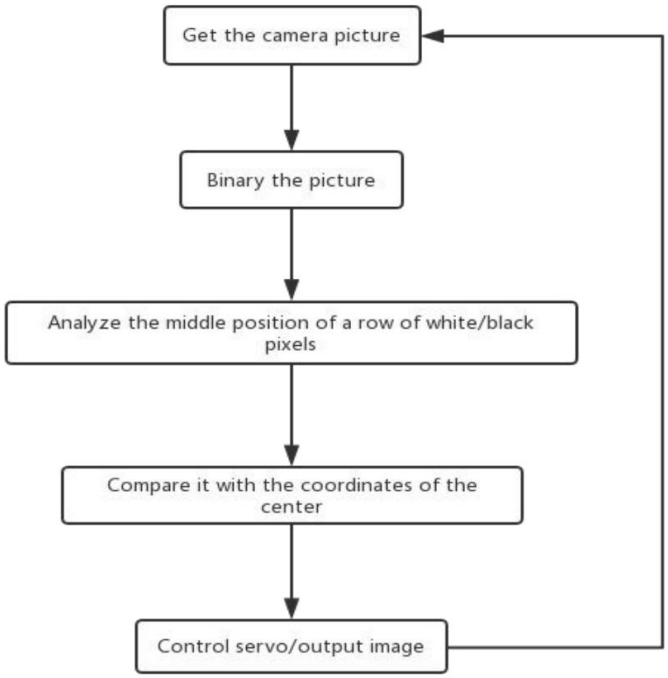


• Process explanation: First obtain a frame from the camera, and then use OpenCV to analyze the content of this frame. After the analysis is completed, the information that needs to be drawn is generated, such as the center point of the target object, and the text lamp information that needs to be generated 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.

•Using such a processing flow will result in the need to wait for the OpenCV-related flow to process each frame of the captured frame. After this frame is displayed, the second frame can be collected and then processed and analyzed. The drawback is that it will seriously affect the frame rate of the video. , It becomes abnormally stuck.

15.1.2 Multi-threaded processing of video frames

• Next, introduce the process of multi-threaded processing of video frames:



• Process explanation: In order to improve the frame rate, we separate the analysis task of the video frame from the acquisition-display process, put it in a background thread for execution and generate drawing information.

• The complete code of camera_opencv.py that we changed to multiple threads is as follows: (The code here is only for reference on the principle of multi-threading, and the OpenCV function is deleted for visualization).

```
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's task of analyzing video frames in the background
    '''
```

```
14.  def __init__(self, *args, **kwargs):
15.      self.CVThreading = 0
16.
17.      super(CVThread, self).__init__(*args, **kwargs)
18.      self.__flag = threading.Event()
19.      self.__flag.clear()
20.
21.
22.  def mode(self, imgInput):
23.      """
24.      This method is used to pass in video frames that need to be processed
25.      '''
26.      self.imgCV = imgInput
27.      self.resume()
28.
29.
30.  def elementDraw(self,imgInput):
31.      """
32.      Draw elements on the screen
33.      '''
34.      return imgInput
35.
36.
37.  def doOpenCV(self, frame_image):
38.      """
39.      Add content to be processed by OpenCV here
40.      '''
41.      self.pause()
42.
43.
44.  def pause(self):
45.      """
46.      Block the thread and wait for the next frame to be processed
47.      '''
48.      self.__flag.clear()
49.      self.CVThreading = 0
50.
51.  def resume(self):
52.      """
53.      Resuming the thread
54.      '''
55.      self.__flag.set()
56.
57.  def run(self):
58.      """
59.      Processing video frames in a background thread
```

```
60.         '''
61.         while 1:
62.             self.__flag.wait()
63.             self.CVThreading = 1
64.             self.doOpenCV(self.imgCV)
65.
66.
67.     class Camera(BaseCamera):
68.         video_source = 0
69.
70.         def __init__(self):
71.             if os.environ.get('OPENCV_CAMERA_SOURCE'):
72.                 Camera.set_video_source(int(os.environ['OPENCV_CAMERA_SOURCE']))
73.             super(Camera, self).__init__()
74.
75.         @staticmethod
76.         def set_video_source(source):
77.             Camera.video_source = source
78.
79.         @staticmethod
80.         def frames():
81.             camera = cv2.VideoCapture(Camera.video_source)
82.             if not camera.isOpened():
83.                 raise RuntimeError('Could not start camera.')
84.             """
85.             Instantiate CVThread()
86.             '''
87.             cvt = CVThread()
88.             cvt.start()
89.
90.             while True:
91.                 # read current frame
92.                 _, img = camera.read()
93.
94.                 if cvt.CVThreading:
95.                     """
96.                     If OpenCV is processing video frames, skip
97.                     '''
98.                     pass
99.                 else:
100.                         """
101.                         If OpenCV is not processing video frames, give the thread that processes the video frame a new video frame and resume the processing thread
102.                         '''
103.                         cvt.mode(img)
104.                         cvt.resume()
```

```
105.            """
106.            Draw elements on the screen
107.            '''
108.            img = cvt.elementDraw(img)
109.
110.            # encode as a jpeg image and return it
111.            yield cv2.imencode('.jpg', img)[1].tobytes()
```

The above is the code principle of using multi-threading to process OpenCV. The following OpenCV specific function introduction will skip the part of explaining multi-threading and directly introduce the method of OpenCV processing video frames.

15.2 Preparation for OpenCV function development

• The real-time video transmission function comes from the open source project [flask-video-streaming] of Github's MIT open source agreement.

• First, prepare two .py files in the same folder on 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
```

```
5.  except ImportError:
6.      try:
7.          from thread import get_ident
8.      except ImportError:
9.          from _thread import get_ident
10.
11.
12. class CameraEvent(object):
13.     """An Event-like class that signals all active clients when a new frame is
14.     available.
15.     """
16.     def __init__(self):
17.         self.events = {}
18.
19.     def wait(self):
20.         """Invoked from each client's thread to wait for the next frame."""
21.         ident = get_ident()
22.         if ident not in self.events:
23.             # this is a new client
24.             # add an entry for it in the self.events dict
25.             # each entry has two elements, a threading.Event() and a timestamp
26.             self.events[ident] = [threading.Event(), time.time()]
27.         return self.events[ident][0].wait()
28.
29.     def set(self):
30.         """Invoked by the camera thread when a new frame is available."""
31.         now = time.time()
32.         remove = None
33.         for ident, event in self.events.items():
34.             if not event[0].isSet():
35.                 # if this client's event is not set, then set it
36.                 # also update the last set timestamp to now
37.                 event[0].set()
38.                 event[1] = now
39.             else:
40.                 # if the client's event is already set, it means the client
41.                 # did not process a previous frame
42.                 # if the event stays set for more than 5 seconds, then assume
43.                 # the client is gone and remove it
44.                 if now - event[1] > 5:
45.                     remove = ident
46.         if remove:
47.             del self.events[remove]
48.
49.     def clear(self):
50.         """Invoked from each client's thread after a frame was processed."""
```

```
51.         self.events[get_ident()][0].clear()
52.
53.
54. class BaseCamera(object):
55.     thread = None  # background thread that reads frames from camera
56.     frame = None  # current frame is stored here by background thread
57.     last_access = 0  # time of last client access to the camera
58.     event = CameraEvent()
59.
60.     def __init__(self):
61.         """Start the background camera thread if it isn't running yet."""
62.         if BaseCamera.thread is None:
63.             BaseCamera.last_access = time.time()
64.
65.             # start background frame thread
66.             BaseCamera.thread = threading.Thread(target=self._thread)
67.             BaseCamera.thread.start()
68.
69.             # wait until frames are available
70.             while self.get_frame() is None:
71.                 time.sleep(0)
72.
73.     def get_frame(self):
74.         """Return the current camera frame."""
75.         BaseCamera.last_access = time.time()
76.
77.         # wait for a signal from the camera thread
78.         BaseCamera.event.wait()
79.         BaseCamera.event.clear()
80.
81.         return BaseCamera.frame
82.
83.     @staticmethod
84.     def frames():
85.         """Generator that returns frames from the camera."""
86.         raise RuntimeError('Must be implemented by subclasses.')
87.
88.     @classmethod
89.     def _thread(cls):
90.         """Camera background thread."""
91.         print('Starting camera thread.')
92.         frames_iterator = cls.frames()
93.         for frame in frames_iterator:
94.             BaseCamera.frame = frame
95.             BaseCamera.event.set()  # send signal to clients
96.             time.sleep(0)
```

```
97.
98.         # if there hasn't been any clients asking for frames in
99.         # the last 10 seconds then stop the thread
100.                 if time.time() - BaseCamera.last_access > 10:
101.                     frames_iterator.close()
102.                     print('Stopping camera thread due to inactivity.')
103.                     break
104.             BaseCamera.thread = None
```

• When you use subsequent tutorials to develop a certain OpenCV-related function, you only need to put the corresponding camera_opencv.py in the same folder as the app.py and base_camera.py, and then in the Raspberry Pi console Just run app.py.

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

192.168.3.157:5000

15.3 Using OpenCV for color tracking

15.3.1 Color recognition and color space

•For the development preparation and operation of OpenCV functions, please refer to 15.2.

• Create camera_opencv.py in the folder where app.py and base_camera.py in 15.2 are located. The code related to the OpenCV color tracking function introduced in this chapter is written in camera_opencv.py.

• For safety, this routine does not control the movement of the motor or the steering gear, but only outputs the result of OpenCV.

• 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 the organization of colors. With the help of color space and physical device testing, fixed analog and digital representations of colors can be obtained. The color space can be defined by just picking some colors at will. For example, the Pantone system just takes a set of specific colors as samples, and then defines the name and code for each color; it can also be based on rigorous mathematical definitions, such as Adobe RGB , SRGB.

• RGB color space:

-RGB adopts additive color mixing method, because it describes the ratio of various "lights" to produce colors. Light starts from dark and continues to overlap to produce colors. RGB describes the value of red, green and blue light. RGBA adds an alpha channel to RGB to achieve transparency.

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

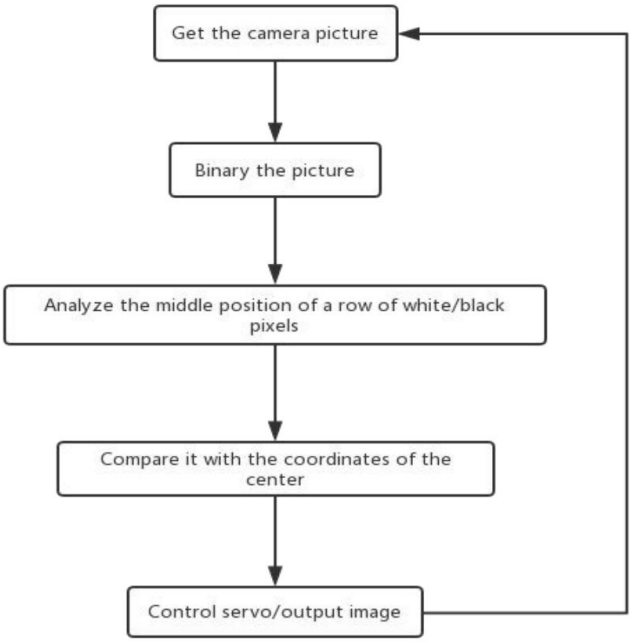
• HSV color space:

-HSV (Hue: Hue, Saturation: Saturation, Lightness; Value), also known as HSB (B means Brightness) is commonly used by artists, because compared with the terms of additive and subtractive color mixing, the concepts of hue, saturation, etc. are used to describe colors More natural and intuitive. HSV is a variant of RGB color space, and its content and color scale are closely related to its source-RGB color space

– Using the HSV color space in the OpenCV 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 the color recognition is not a certain color, but a certain color. A range of colors, so the HSV color space that is more in line with human eye habits should be used for color recognition.

15.3.2 Color recognition and tracking process

• 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.



15.3.3 Specific code

- **camera_opencv.py**

```
1.  import os
2.  import cv2
3.  from base_camera import BaseCamera
4.  import numpy as np
5.
6.  '''
7.  Set the target color, HSV color space
8.  '''
9.  colorUpper = np.array([44, 255, 255])
10. colorLower = np.array([24, 100, 100])
11.
12. font = cv2.FONT_HERSHEY_SIMPLEX
13.
14. class Camera(BaseCamera):
15.     video_source = 0
16.
17.     def __init__(self):
18.         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 images captured by the camera

            hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)  #Convert the captured image to HSV color space
            mask = cv2.inRange(hsv, colorLower, colorUpper) #Traverse the colors in the target color range in the HSV color space screen, and turn these color blocks into masks
            mask = cv2.erode(mask, None, iterations=2)  #Corrupt the small blocks of the mask (noise) in the picture (the small blocks of color or noise disappear)
            mask = cv2.dilate(mask, None, iterations=2) #Expand, change 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 screen
            center = None
            if len(cnts) > 0:    #If the number of entire masks in the screen is greater than one
                '''
                Find the center point coordinates of the object of the target color and the size of the object in the screen
                '''
                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)
                '''
                Obtain the center point coordinates of the target color object and output it
                '''
                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()
```

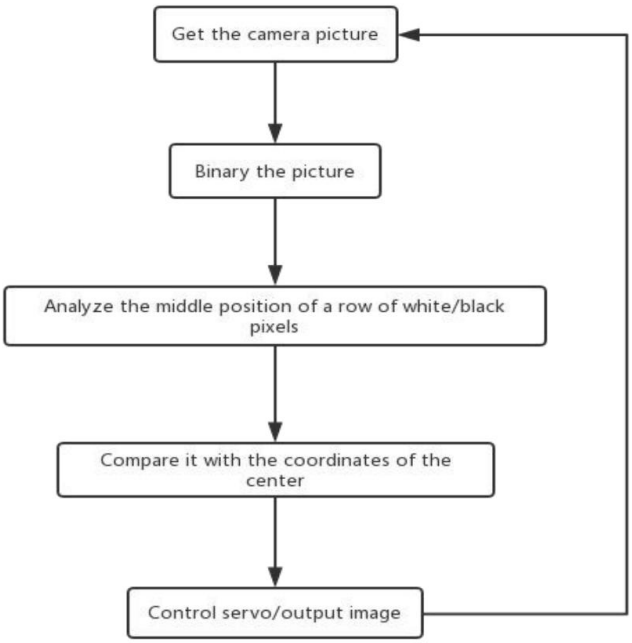
•You can set the color you want to recognize by changing colorUpper and colorLower. It should be noted that the H value (hue) of the normal HSV color space is 0-360, but in OpenCV, the H value range is 0- 180.

15.3.4 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

15.4 Using OpenCV for visual line inspection

15.4.1 Visual inspection process



•For the development preparation and operation of OpenCV functions, please refer to 15.2.

•Create camera_opencv.py in the folder where app.py and base_camera.py in 15.2 are located. The code related to the OpenCV visual line tracking function introduced in this chapter is written in camera_opencv.py.

- • For safety, this routine does not control the movement of the motor or the steering gear, but only outputs the result of OpenCV.

15.4.2 Specific code

```
1. import os
2. import cv2
3. 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 reference horizontal position, the larger the value, the lower, but it cannot be 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 image captured by the camera

            """
            Convert the screen to black and white, and then binarize it (the value of each pixel in the screen is 255 except for 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 to denoise
            colorPos = img[linePos]                 #Get the array of pixel values       of this line of linePos
            try:
                lineColorCount_Pos = np.sum(colorPos == lineColorSet)   #Get the number of pixels of the line color (line width)
```

```
            lineIndex_Pos = np.where(colorPos == lineColorSet)      #Get the horizontal position of the endpoint of the line on the line of linePos
            """
            Use the endpoint position and line width to calculate the position of the line center point
            '''
            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 the line is not detected, the line width above is 0 as the denominator will cause an error, so you know that no line is 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()
```

16. GUI Control Function

• Our old version of the robot program provides a desktop GUI program to control the robot. The GUI program is written in Python, but this method has a high threshold and difficulty, and it is not recommended for novices.

• This GUI program is currently only compatible with Windows systems, and is included in the client directory of the robot software package, generally called GUI.py.

16.1 Installing GUI dependency libraries

• Install Python3:

–We need to install Python on the computer to run our PC-side programs. 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 with Python, and OpenCV needs to use its related functions.

-Press Win+R keys, enter cmd, and click OK to open CMD.

-Enter the following to install numpy:

pip3 install numpy

– – After the input is complete, press Entry to start downloading and installing numpy.

• Install OpenCV:

-The same method as installing numpy.

– After opening CMD, enter the following:

pip3 install opencv-contrib-python

-After the input is complete, press Entry to start downloading and installing opencv.

• Install zmq and pybase64:

--Zmq and pybase64 are libraries for real-time video

– After opening CMD, enter the following:

pip3 install zmq pybase64

– -After the input is complete, press Entry to start downloading and installing zmq and pybase64.

16.2 Introduction to the functions of the GUI control interface

•Since the web terminal and GUI are not connected, if you want to use GUI to control robot products, you need to manually run server.py. (It is consistent with the method of manually running webserver.py, except that the object is replaced by server.py).

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

•Python running window: This window will appear accompanied by each GUI when it is opened. Any runtime exceptions will be displayed here. If you close this window, the GUI will be exited.

•Camera video stream window: display the screen 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, and click Connect to connect the GUI to the Raspberry Pi.

• Raspberry Pi status and connection status display bar: Display some hardware information and current connection status of the Raspberry Pi.

•Mobile control:

–Forward: Control the robot to move forward, shortcut key W.

–Backward: Control the robot to move backward, the shortcut key S.

–Left: Control the robot to turn left, shortcut key A.

-Right: Control the robot to turn right, shortcut key D.

• Robotic arm and camera control (due to the different layout, the shortcut keys here are different from those of the WEB application):

-Up: Control the camera to move up, the shortcut key I.

-Down: Control the camera downward movement, shortcut key K.

–\: Control the rotation of the mechanical arm chuck, the shortcut key U.

–/: To control the rotation of the mechanical arm chuck, the shortcut key O.

--Grab: Control the rotation of the mechanical arm chuck, shortcut key J.

-Loose: Control the rotation of the mechanical arm chuck, the shortcut key L.

–<--: Control the robot arm forward, shortcut key Q.

---->: To control the robot arm backward, the shortcut key E.

-Home: Control all servos to return to the neutral position, shortcut key H.

16.3 Controlling LED lights via TCP communication

•You can communicate with the Raspberry Pi using the program with a graphical interface you write on other devices to achieve the purpose of controlling the Raspberry Pi.

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

-Tkinter is the standard GUI library for Python. Python uses Tkinter to quickly create GUI applications.

Since Tkinter is built into the python installation package, you can import the Tkinter library as long as you install Python, and IDLE is also written in Tkinter, so Tkinter can handle the simple graphical interface easily.

• We use the Socket library to communicate between devices. Socket is also called "socket". Applications usually send requests to the network or respond to network requests through "sockets", so that between hosts or processes on a computer Can communicate.

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

• Ready to burn Raspbian Raspberry Pi, related dependent libraries and connection methods, you can refer to the WS2812 LED light tutorial to operate. If you don't use RobotHAT, you can 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 lights. If you have not installed it yet and have not run the installation script of the robot, you can use the following command in the Raspberry Pi console to install:

sudo pip3 install rpi_ws281x

• We 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 the socket library 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 freely choose from 0-65535. It is recommended to choose 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 monitoring the client connection, and start receiving the information sent from the client after the client connection is successful
'''
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 continue to monitor the next message sent by the client
    '''
    print(data)
```

The client program on the PC is as follows:

```
'''
Import the socket library 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 the defined port number. You can freely choose 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 part of the GUI
'''
root = tk.Tk()  # Define a window
root.title('Lights')    #  Title of the window
root.geometry('175x55') # The size of the window, the x in the middle 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 on the button is'ON', the text color of the button is #E1F5FE, and the background color of the button is #0277BD. When the button is pressed, call lights_on() 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)

'''
Define another button in the same way. The difference is that the text on the button is changed to'OFF'. When the button 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 out in the terminal of Raspberry Pi, indicating that the program runs successfully.

17. How to Use OpenCV to Open Real-time Video Screen

●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
```

```
42.
43.     '''
44.     We encode the frame into stream data and save it in the memory buffer
45.     '''
46.     encoded, buffer = cv2.imencode('.jpg', frame_image)
47.     jpg_as_text = base64.b64encode(buffer)
48.
49.     '''
50.     Here we send the stream data in the buffer through base64 encoding to the video receiving end
51.     '''
52.     footage_socket.send(jpg_as_text)
53.
54.     '''
55.     Clear the stream in preparation for the next frame
56.     '''
57.     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 :

```
1.  '''
58. First import the required libraries
2.  '''
3.  import cv2
4.  import zmq
5.  import base64
6.  import numpy as np
7.
8.  '''
9.  Here we instantiate the zmq object used to receive the frame
10. Note that the port number needs to be consistent with the sender's
11. '''
12. context = zmq.Context()
13. footage_socket = context.socket(zmq.PAIR)
14. footage_socket.bind('tcp://*:5555')
15.
16. while True:
17.     '''
18.     Received video frame data
19.     '''
20.     frame = footage_socket.recv_string()
21.
22.     """
23.     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.

18. How to Use OpenV to Process Video Frames

●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 **17** to send video frames to the host computer, or refer to the content of **14** 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 **17**. 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)
```

```
27.
28.     '''
29.     Interpret a buffer as a 1-dimensional array
30.     '''
31.     npimg = np.frombuffer(img, dtype=np.uint8)
32.
33.     '''
34.     Decode a one-dimensional array into an image
35.     '''
36.     source = cv2.imdecode(npimg, 1)
37.
38.     '''
39.     Display image
40.     '''
41.     cv2.imshow("Stream", source)
42.
43.     '''
44.     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
45.     '''
46.     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:

```
1.  '''
2.  First import the required libraries
3.  '''
4.  import cv2
5.  import zmq
6.  import base64
7.  import numpy as np
8.
9.  '''
10. Here we instantiate the zmq object used to receive the frame
11. Note that the port number needs to be consistent with the sender's
12. '''
13. context = zmq.Context()
14. footage_socket = context.socket(zmq.PAIR)
15. footage_socket.bind('tcp://*:5555')
16.
17. while True:
18.     '''
19.     Received video frame data
20.     '''
```

```
    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)
```

19. How to Turn on the UART of Raspberry Pi

●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).

19.1 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.

19.2 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.

19.3 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.

19.4 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
...
```

19.5 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.

20. How to Display Information on the OLED Screen

This product is equipped with an OLED screen to display some information, you can customize the content displayed on the screen by the code.

●The code related to the OLED screen is in [robot_name]/server/OLED.py.

```
'''
Import the libraries needed to control the OLED screen
'''
from luma.core.interface.serial import i2c
from luma.core.render import canvas
from luma.oled.device import ssd1306
import time

'''
Import multi-threaded library
'''
import threading

'''
Connect to OLED screen, the default drive address is 0X3C
'''
try:
    serial = i2c(port=1, address=0x3C)
    device = ssd1306(serial, rotate=0)
except:
    print('OLED disconnected\nOLED 没有连接')

'''
OLED screen can display 6 lines of text, here set the initial content of each line of text
'''
text_1 = 'HELLO WORLD'
text_2 = 'IP:CONNECTING'
text_3 = '<ARM> OR <PT> MODE'
text_4 = 'MPU6050 DETECTING'
text_5 = 'FUNCTION OFF'
text_6 = 'Message:None'


'''
Multithreading is used here to control the OLED screen, so that the OLED screen will not be blocked in other programs.
'''
class OLED_ctrl(threading.Thread):
```

```
    def __init__(self, *args, **kwargs):
        super(OLED_ctrl, self).__init__(*args, **kwargs)
        self.__flag = threading.Event()  #The flag used to pause the thread
        self.__flag.set()     # Set to True
        self.__running = threading.Event()    # The flag used to stop the thread
        self.__running.set()     # Set running to True

    def run(self):
        '''
         This is the content the thread specifically executes
        '''
        while self.__running.isSet():
            self.__flag.wait()    # Return immediately when it is True, block when it is False until the internal flag is True
            with canvas(device) as draw:
                '''
                Use the draw function to write text on the screen, which requires three parameters, the first is the position parameter, the second is the text content, and the third is the text color
                The OLED we use is a monochrome OLED screen, so the third parameter is'white'
                '''
                draw.text((0, 0), text_1, fill="white")
                draw.text((0, 10), text_2, fill="white")
                draw.text((0, 20), text_3, fill="white")
                draw.text((0, 30), text_4, fill="white")
                draw.text((0, 40), text_5, fill="white")
                draw.text((0, 50), text_6, fill="white")
            print('loop')
            '''
            The thread hangs after changing the text content, waiting for the next command
            '''
            self.pause()

    def pause(self):
        self.__flag.clear()  # Set to False to let the thread block

    def resume(self):
        self.__flag.set()  # Set to True to stop the thread from blocking

    def stop(self):
        self.__flag.set()     # Resume the thread from the suspended state, if it has been suspended
        self.__running.clear()     #  Set to False

    def screen_show(self, position, text):
        '''
         Change the content of a line by calling this function externally
        '''
```

```
77.         global text_1, text_2, text_3, text_4, text_5, text_6
78.         if position == 1:
79.             text_1 = text
80.         elif position == 2:
81.             text_2 = text
82.         elif position == 3:
83.             text_3 = text
84.         elif position == 4:
85.             text_4 = text
86.         elif position == 5:
87.             text_5 = text
88.         elif position == 6:
89.             text_6 = text
90.         self.resume()
91.
92. if __name__ == '__main__':
93.     screen = OLED_ctrl()
94.     screen.start()
95.     '''
96.      Replace the text on the first line with'ROBOT'
97.     '''
98.     screen.screen_show(1, 'ROBOT')
99.     while 1:
100.                 '''
101.                 It should be noted that the program cannot be exited, because the exit will cause the screen to
        go out.
102.                 '''
103.                 time.sleep(10)
104.                 pass
```

21.How to Control RaspTank Pro 140 via Mobile APP

●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 Lesson 3

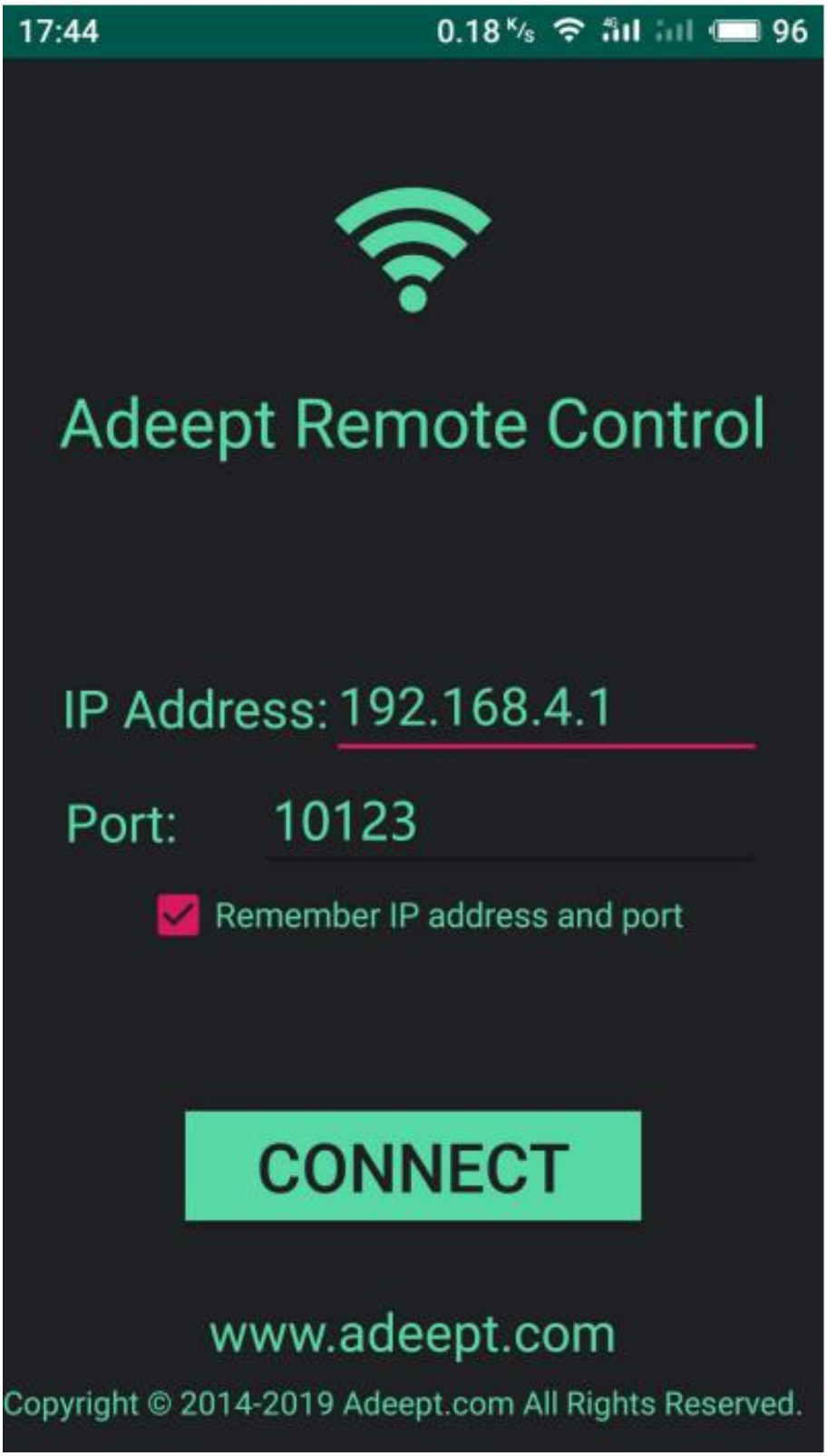
●Our old version of the robot program is equipped with the method of using the mobile phone APP to control, but the mobile phone APP only supports Android phones and cannot use the functions related to the Raspberry Pi camera. The program corresponding to this function in the Raspberry Pi is **appserver.py**

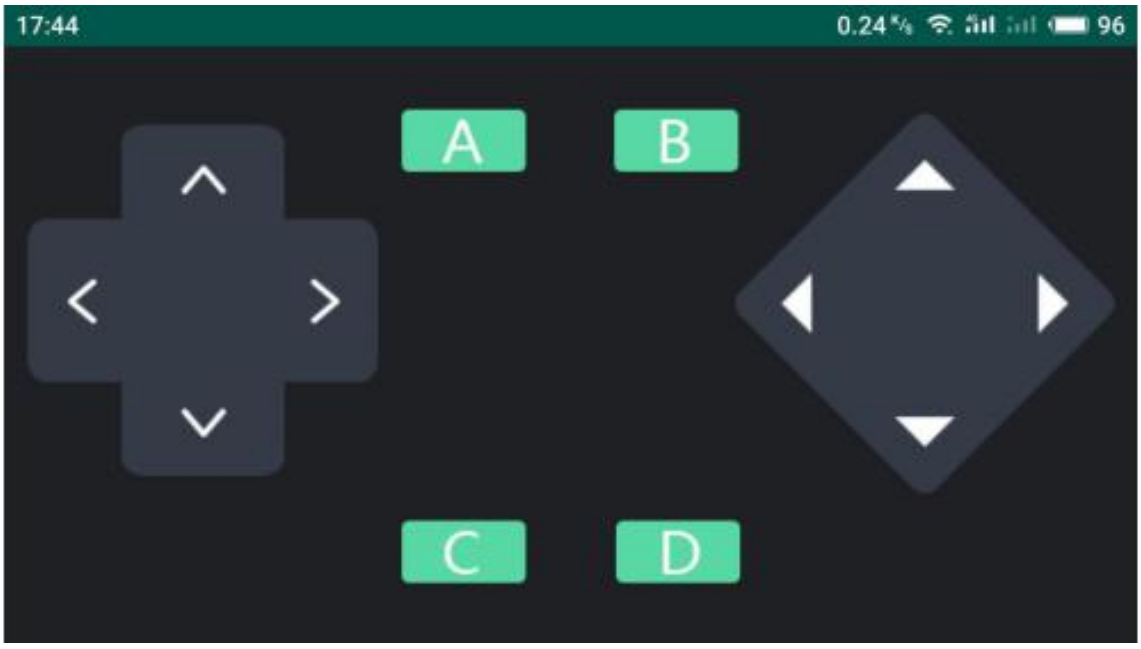
●You can find the download address of the mobile APP in our official website. The installation method is the same as that of the ordinary mobile phone APP.

●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**.


17:44
0.18 K/s
96
Adeept Remote Control
IP Address: 192.168.4.1
Port: 10123
Remember IP address and port
CONNECT
www.adeept.com
Copyright © 2014-2019 Adeept.com All Rights Reserved.


17:44
0.24
96
A
B
C
D

22. How to Turn on the Raspberry Pi Hotspot

●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 using a browser .12.1: 5000 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.

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 Robot 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_rasptank/server/config.txt //etc/config.txt

Just replace adeept_rasptank above with your product name, we will take Rasptank as an example here.

Adeept

RaspTank Pro

www.adeept.com