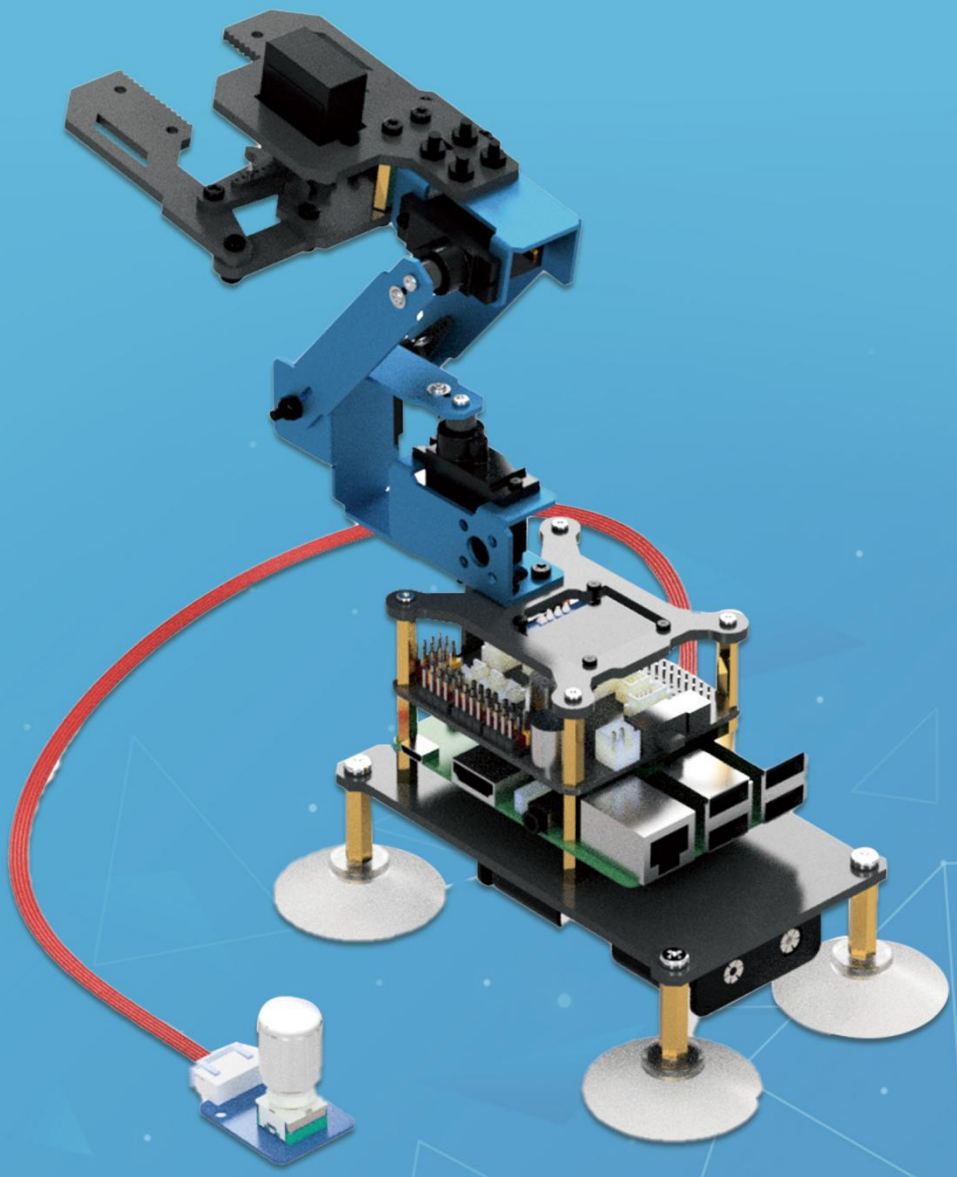
Adeept

RaspArm-S



www.adeept.com

Content

Introduction of RaspArm-S Robotic Arm

RaspArm-S is an open source robotic arm product which is based on the Raspberry Pi. It has a simple structure and a smooth learning curve. At the same time, this product provides 26 supporting courses (the course will be followed by higher-level function updates). From the beginning of getting the Raspberry Pi, everyone has to learn advanced knowledge and skills step by step, and understand how to use the Python language for rapid Maker hardware product development.

We provide a variety of control methods for RaspArm-S. You can not only use Web applications to control RaspArm-S on your mobile phone, but remotely control RaspArm-Son another computer. And because all the Raspberry Pi interfaces on the RaspArm-S are exposed, you can also connect the mouse, keyboard and monitor to the RaspArm-S to directly control it.

Regarding the control of RaspArm-S, we use the inverse connection method to control the servo, which is different from the control method that depends on inputting the angle of the servo. This control method only needs to input the position of the end point to control the execution of the robotic arm. For example, if we want to hold something, we only need to consider where we should put our hands. There is no need to consider that how many degrees each joint rotates.

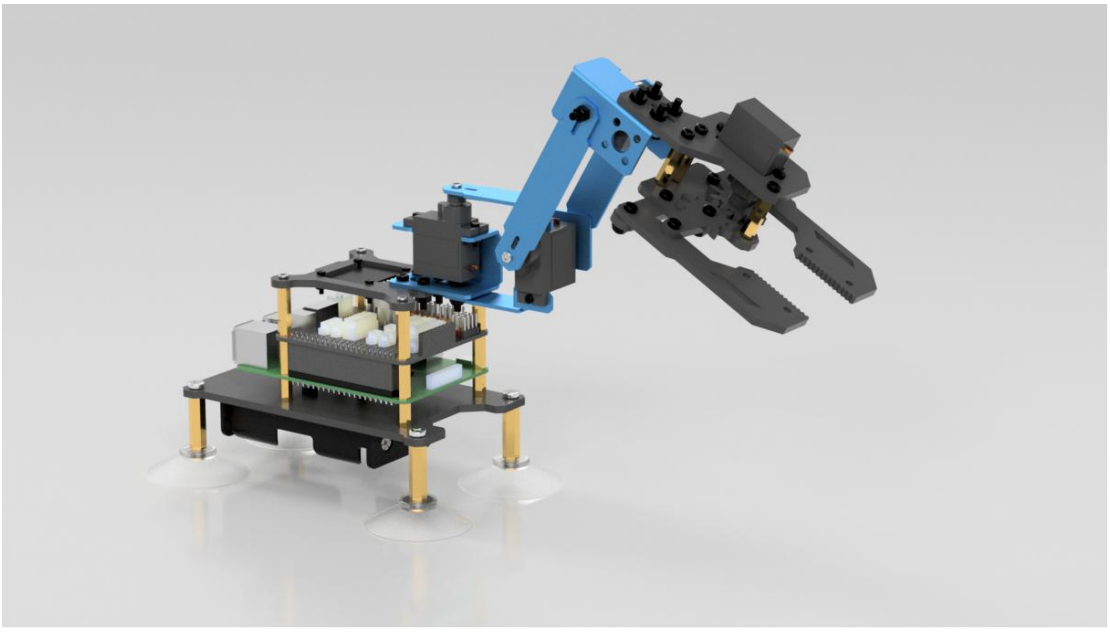
As an open source hardware product, we pay special attention to the secondary development potential of RaspArm-S. We also provide detailed API calling methods. If you want to build your own project, you can directly call the existing API to control RaspArm-S at the code level, for this can greatly improve development efficiency.

In order to obtain a reliable and stable structure, we have reduced the mechanical complexity of the product as much as possible, while maintaining the scalability of the product. For example, you can change the gripper to a pen, and then set or rewrite a variable in a program with the GUI. The function of the end servo will change from controlling the gripper to keeping the pen at a certain angle. The chuck is a relatively

complicated part of the structure. We use a linkage mechanism to increase the strength of the chuck.

There are drawings of the robotic arm in our document, including dimensions and assembly hole specifications, which is convenient for users to use the robotic arm in more occasions.The robotic arm itself is also a modular design, so users can flexibly assemble it into the form they need.

RaspArm-S is equipped with an OLED screen to display key information. When you are debugging or running the robotic arm on the desktop, you can use the micro USB interface on the driver board to power the RaspArm-S. No extra battery is required. This avoids the trouble of charging the product during the development process. The advantages of permanent power and simple structure enable you to focus on product development without any worries and realize your creative ideas more efficiently.



Raspberry Pi

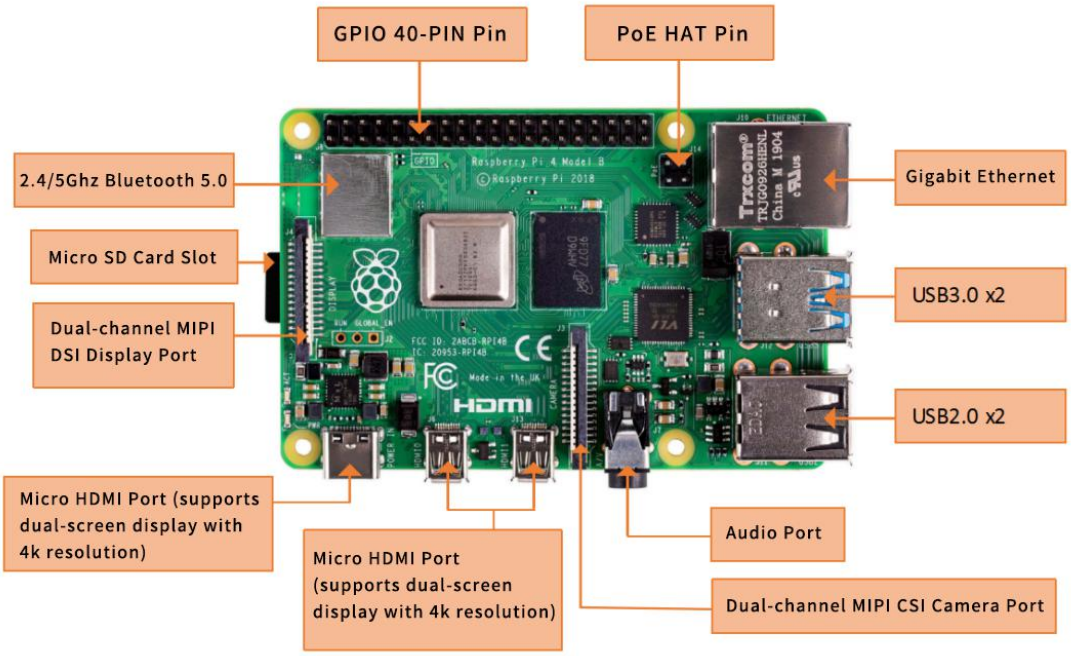
1. Introduction to Raspberry Pi

(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.

(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.

(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.

(4)Programming language

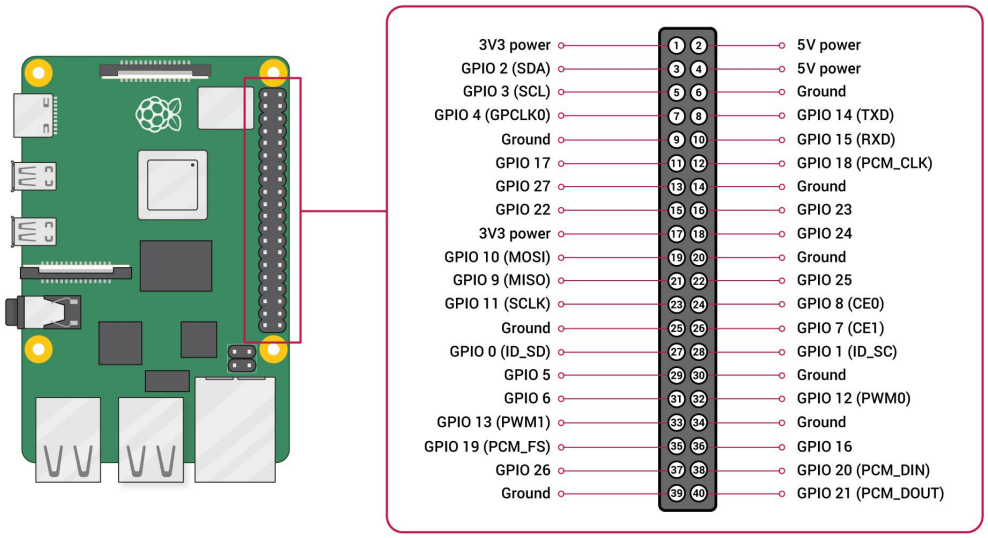
For the Raspberry Pi, there are many programming languages available. In fact, any language that can be compiled for the ARM architecture (such as the C language) can be used for the Raspberry Pi. The most popular language should be Python. In

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.

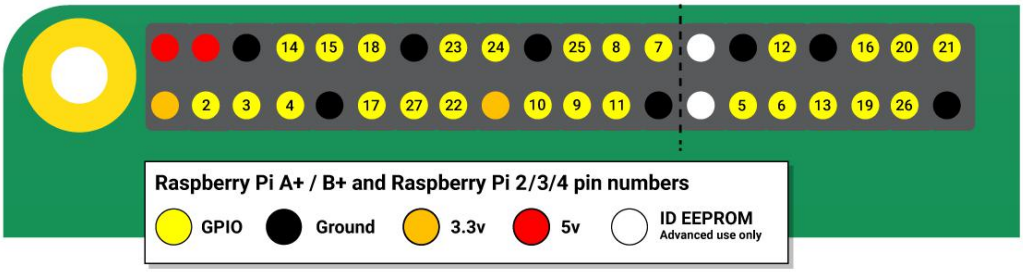
2. Introduction to GPIO

(1) What is GPIO

A powerful feature of the Raspberry Pi is the row of GPIO (general-purpose input/output) pins along the top edge of the board. A 40-pin GPIO header is found on all current Raspberry Pi boards (unpopulated on Pi Zero and Pi Zero W). Prior to the Pi 1 Model B+ (2014), boards comprised a shorter 26-pin header.



Any of the GPIO pins can be designated (in software) as an input or output pin and used for a wide range of purposes.



Note: the numbering of the GPIO pins is not in numerical order; GPIO pins 0 and 1 are present on the board (physical pins 27 and 28) but are reserved for advanced use (see below).

Voltages

Two 5V pins and two 3V3 pins are present on the board, as well as a number of ground pins (0V), which are unconfigurable. The remaining pins are all general purpose 3V3 pins, meaning outputs are set to 3V3 and inputs are 3V3-tolerant.

Outputs

A GPIO pin designated as an output pin can be set to high (3V3) or low (0V).

Inputs

A GPIO pin designated as an input pin can be read as high (3V3) or low (0V). This is made easier with the use of internal pull-up or pull-down resistors. Pins GPIO2 and GPIO3 have fixed pull-up resistors, but for other pins this can be configured in software.

More

As well as simple input and output devices, the GPIO pins can be used with a variety of alternative functions, some are available on all pins, others on specific pins.

-PWM (pulse-width modulation)

-Software PWM available on all pins

-Hardware PWM available on GPIO12, GPIO13, GPIO18, GPIO19

-SPI

-SPI0: MOSI (GPIO10); MISO (GPIO9); SCLK (GPIO11); CE0 (GPIO8), CE1 (GPIO7)

-SPI1: MOSI (GPIO20); MISO (GPIO19); SCLK (GPIO21); CE0 (GPIO18); CE1 (GPIO17); CE2 (GPIO16)

-I2C

-Data: (GPIO2); Clock (GPIO3)

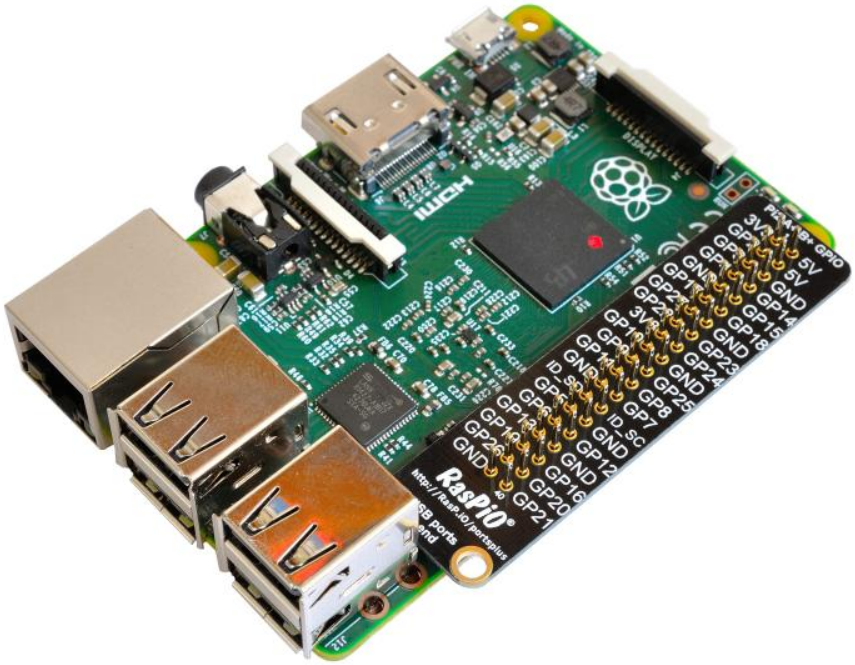
-EEPROM Data: (GPIO0); EEPROM Clock (GPIO1)

-Serial

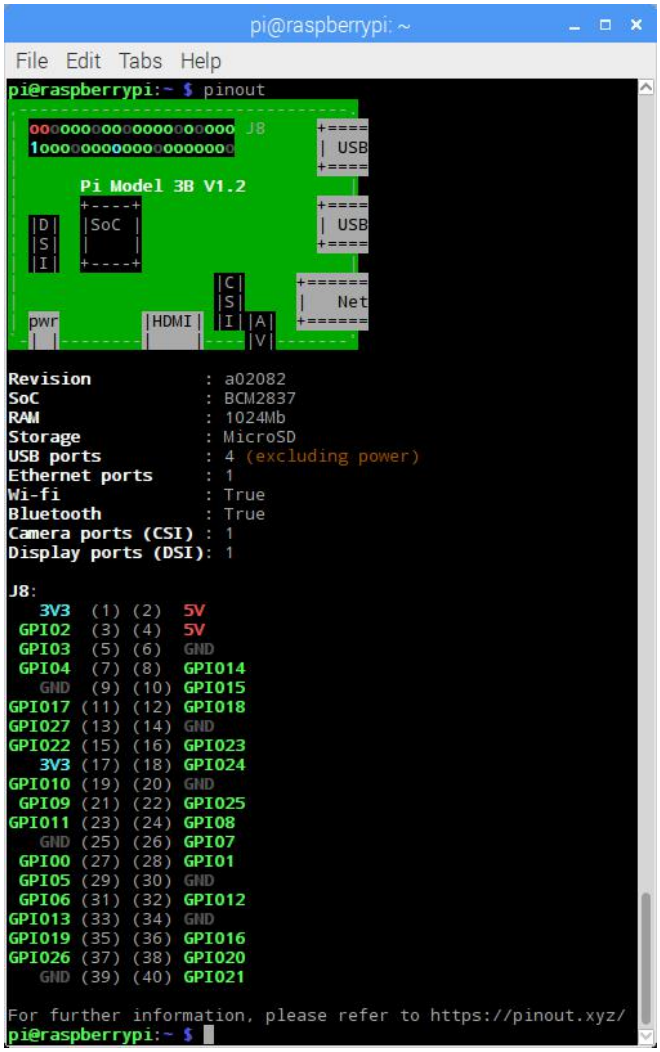
-TX (GPIO14); RX (GPIO15)

GPIO pinout

It's important to be aware of which pin is which. Some people use pin labels (like the RasPiO Portsplus PCB, or the printable Raspberry Leaf).



A handy reference can be accessed on the Raspberry Pi by opening a terminal window and running the command pinout. This tool is provided by the GPIO Zero Python library, which it is installed by default on the Raspberry Pi OS desktop image, but not on Raspberry Pi OS Lite.



For more details on the advanced capabilities of the GPIO pins see gadgetoid's interactive pinout diagram.

(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.

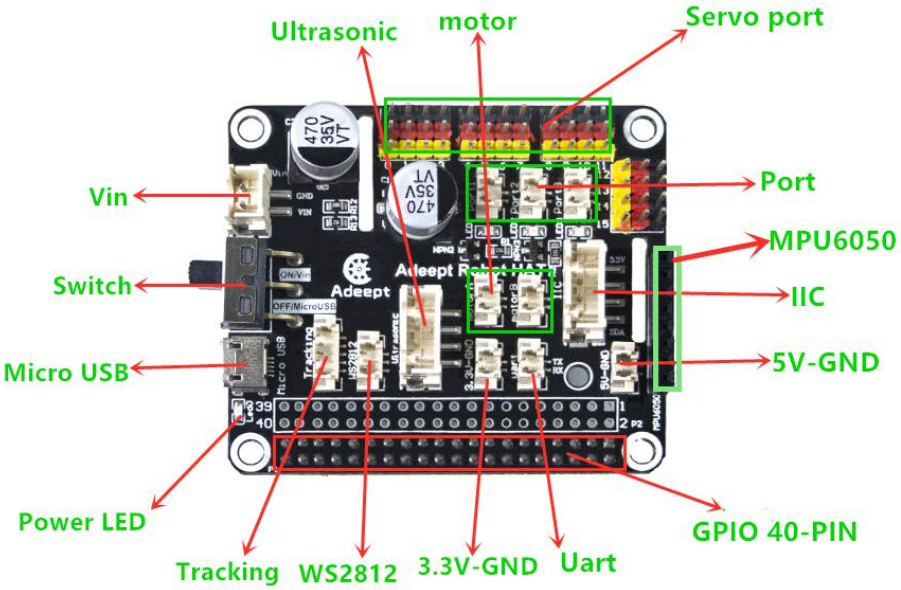
(5) TXD and RXD pins

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.

Introduction of Robot HAT Driver Board

1.Introduction of Robot HAT driver board

When you get the RaspArmS product, you will see a board with its name printed on it called: Adeept Robot HAT, which is an important part of RaspArmS. 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 RaspArmS 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 Alter 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.

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.

Lesson 1 Installing and Logging in to the Raspberry Pi System

In this lesson, we will learn how to install and remotely log in to the Raspberry Pi system under Windows. And we will download the code program to control the robot.

1.1 Preparation

(1) When studying this lesson, you need to prepare the following components first:

One SD card that has been formatted (we recommend using an SD card with memory above 16G), 1 card reader, Raspberry Pi development board.

(2) You need to insert the SD card into the card reader first, and then connect the card reader to the computer.

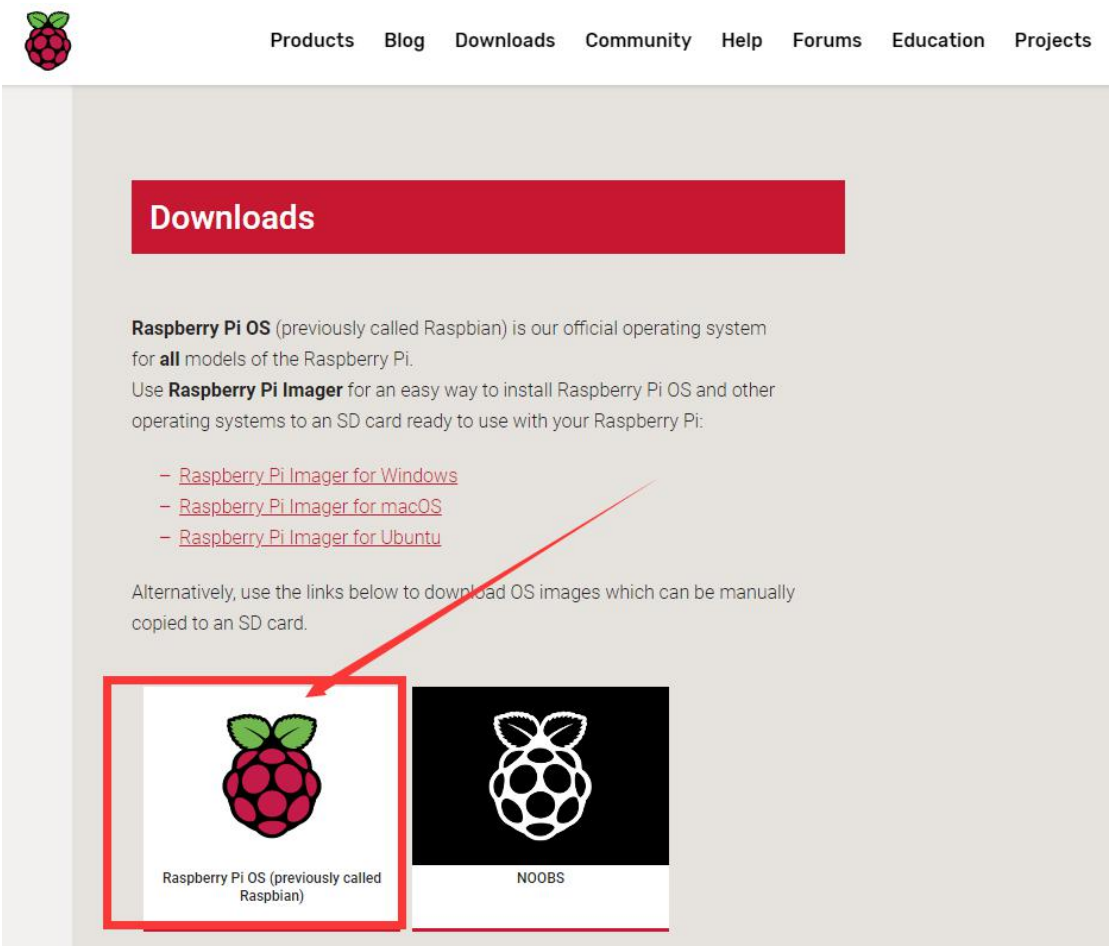
1.2 Downloading the Raspberry Pi system Raspbian

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.

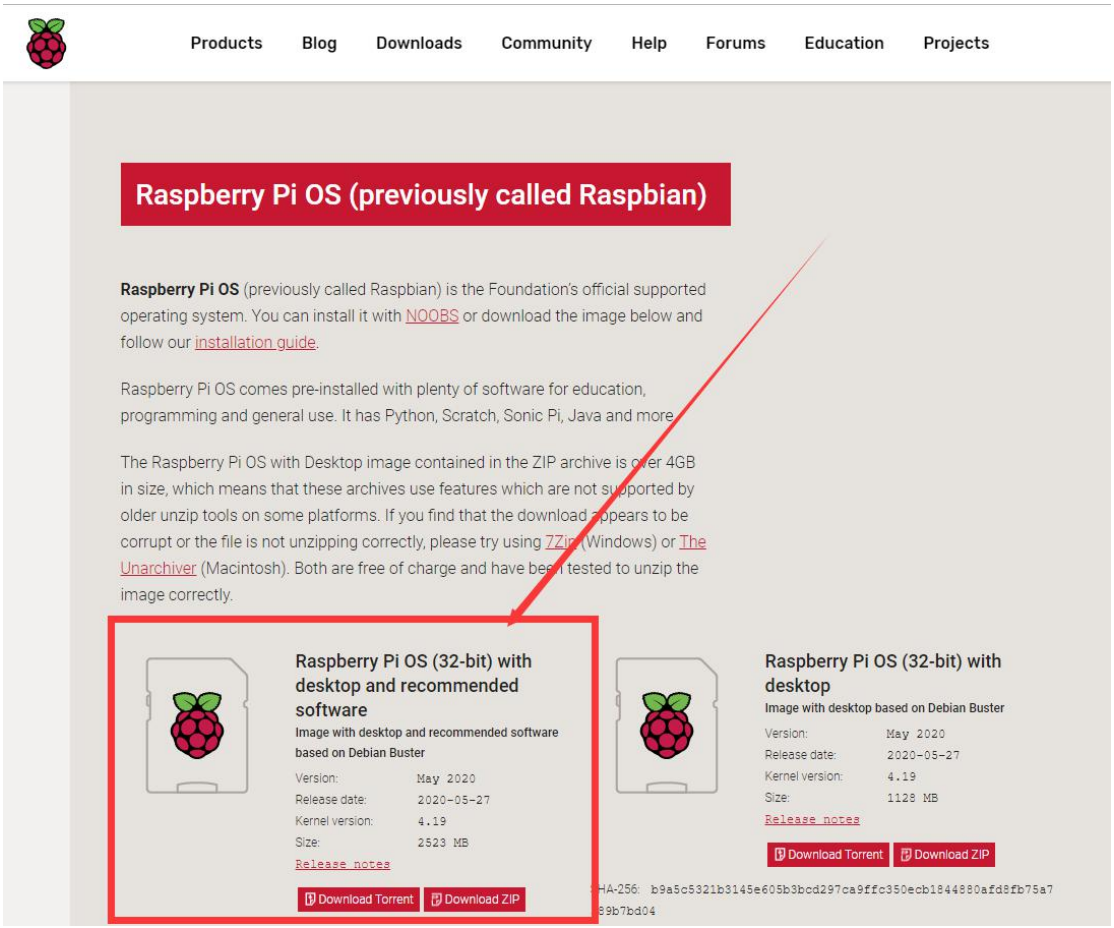
(1) First, visit the official website of the Raspberry Pi through a browser to download Raspbian:

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

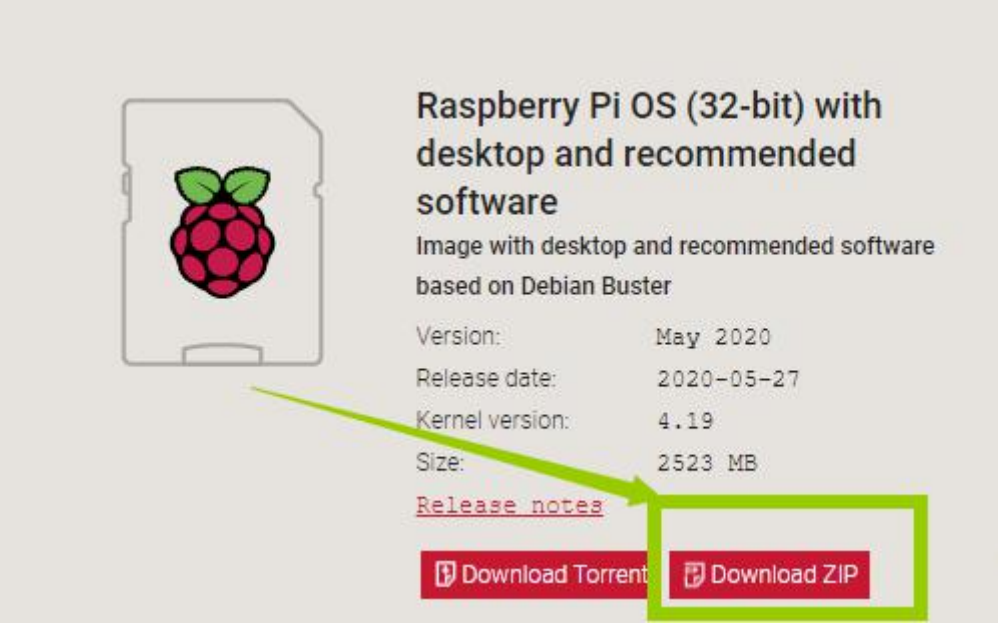
After logging in to the official website, click on the location shown below:



(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.

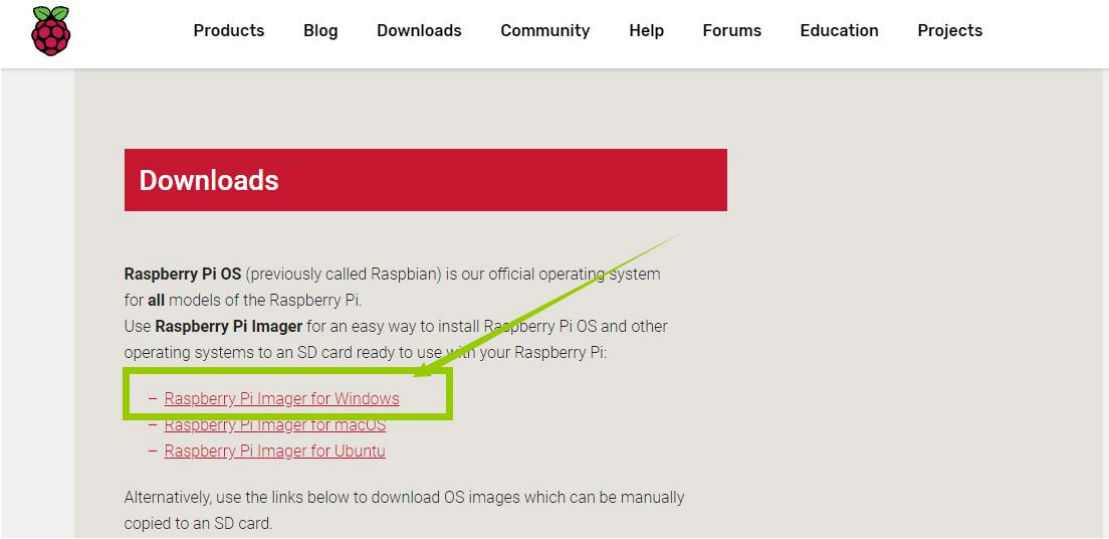
1.3 Burning the downloaded Raspberry Pi system to the SD card

We recommend using the Raspberry Pi Imager tool officially provided by the Raspberry Pi. Raspberry Pi Imager is a new image burning tool launched by the Raspberry Pi Foundation. Users can download and run this tool on Windows, macOS and Ubuntu to burn the system image for the Raspberry Pi. Its usage is similar to Etcher and win32diskimager.

1.3.1 Downloading Raspberry Pi Imager

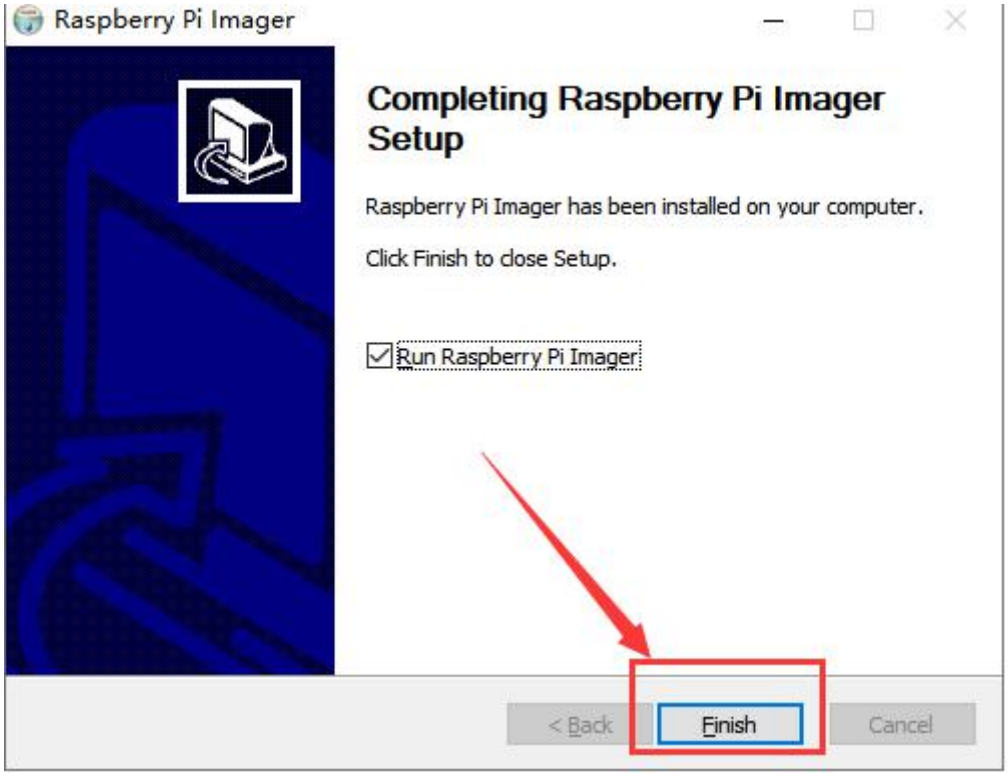
(1) Visit the official website of Raspberry Pi to download with a browser: https://www.raspberrypi.org/downloads/.

Click "Raspberry Pi Imager for Windows" to download. Wait for the download to complete.



(2) Open the downloaded file "imager.exe" and click "Install".

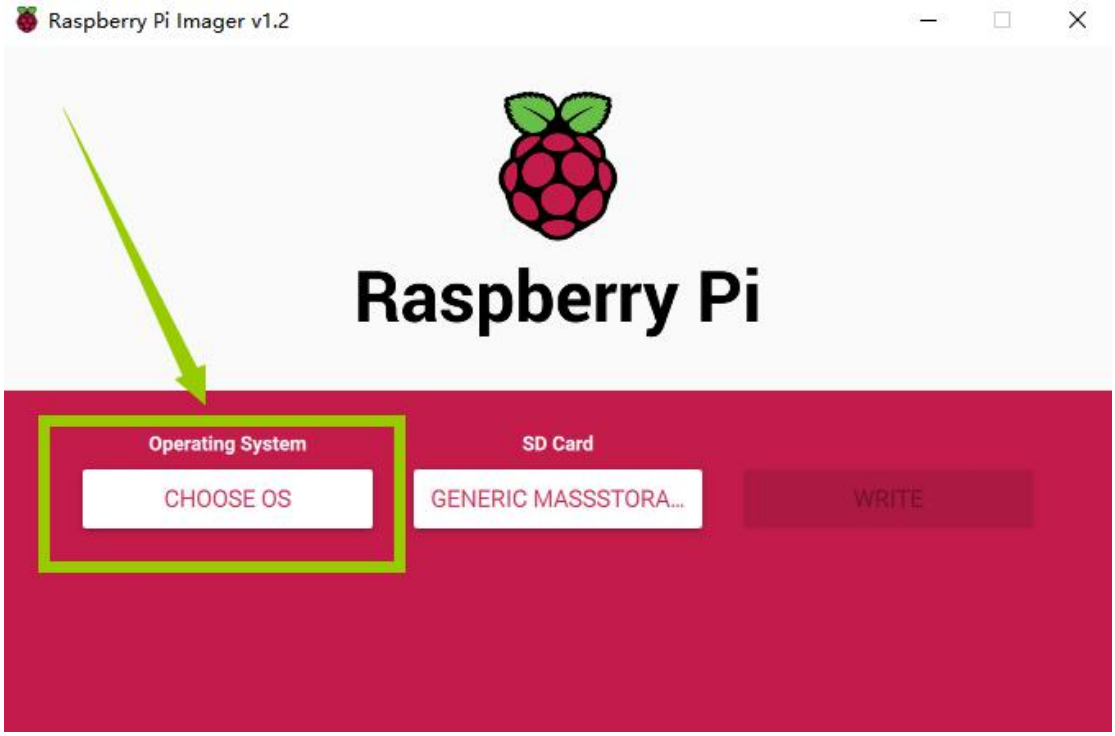
(3) Then click "Finish".



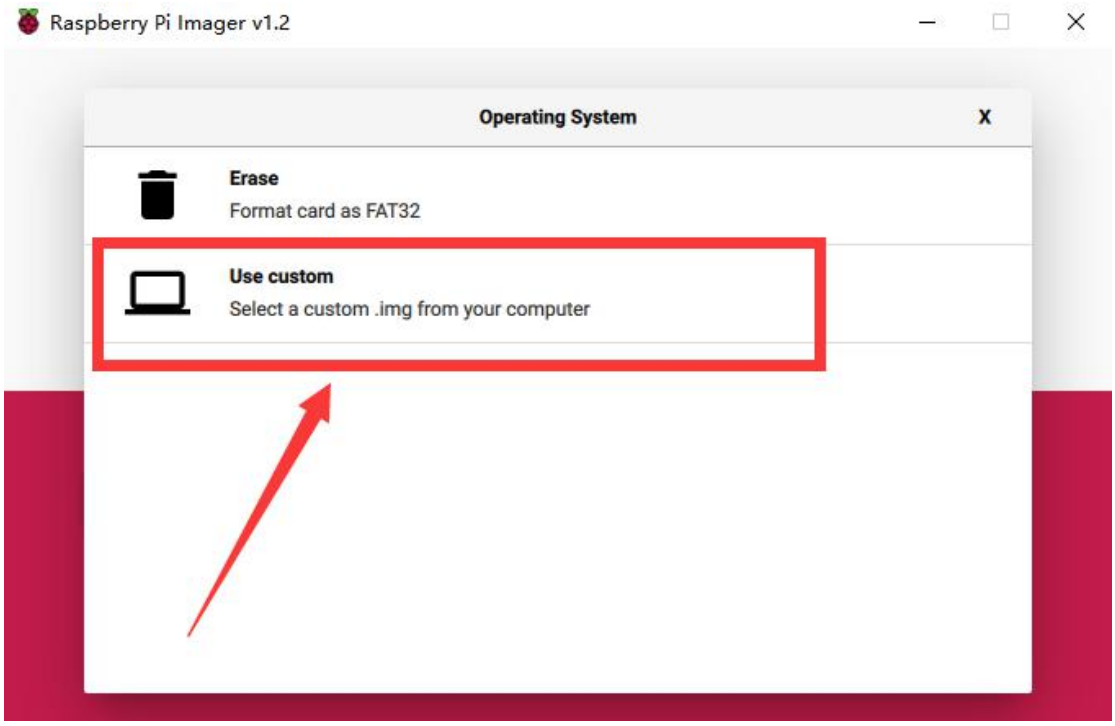
(4) The software interface after opening is as shown below:

1.3.2 Burning Raspberry Pi system to SD card with Raspberry Pi Imager

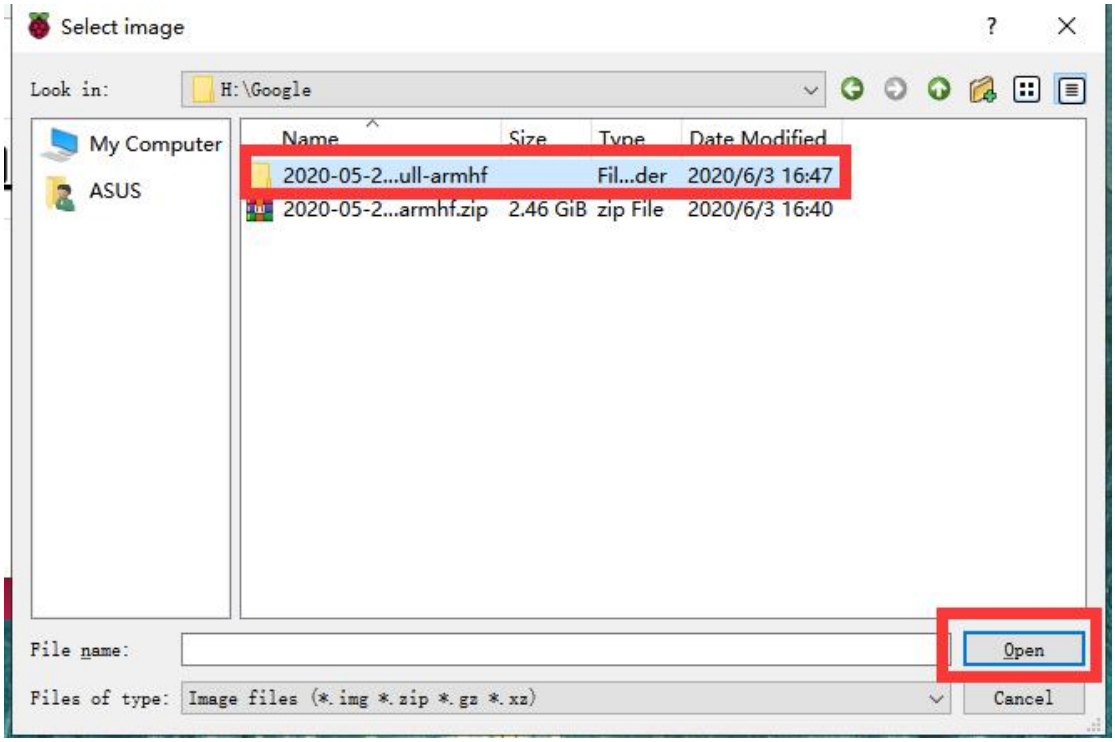
(1) Click "CHOOSE OS" on the opened Raspberry Pi Imager software interface.



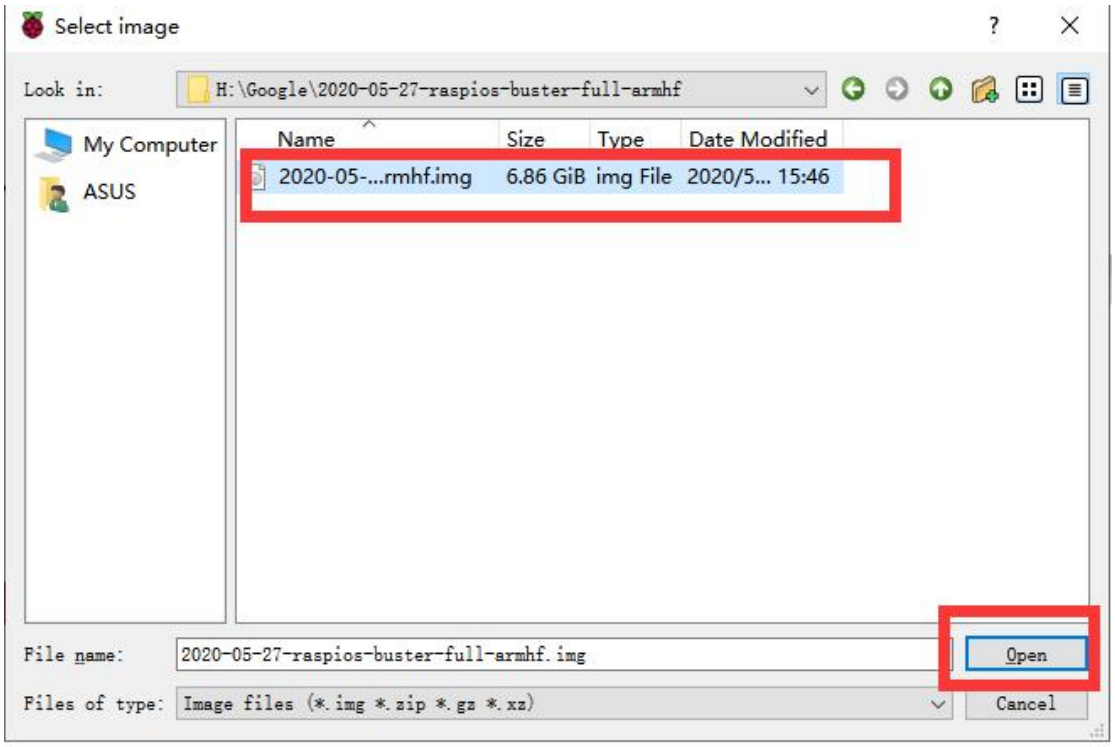
(2) Click "Use custom" and select a custom ".img" file from your computer, which is the ".img" file of the Raspberry Pi system that we downloaded and decompressed before.



(3) Find the ".img" file of the Raspberry Pi system that we downloaded and decompressed before. Click "Open".

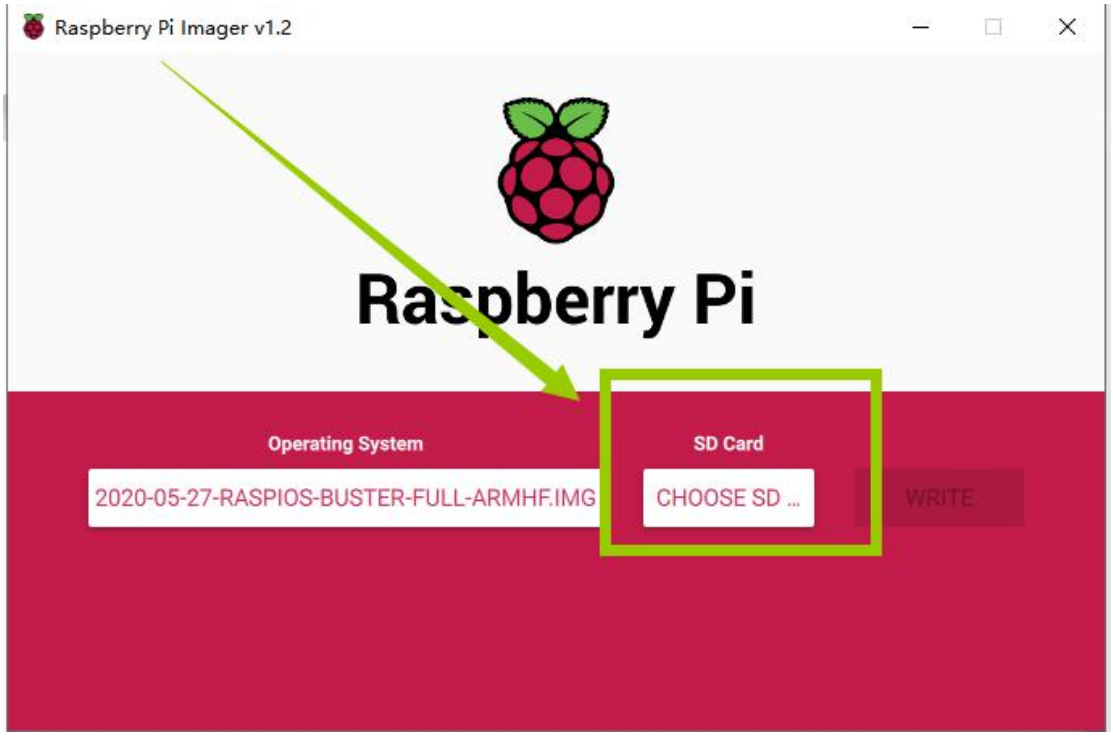


(4) Select the ".img" file and click "Open".

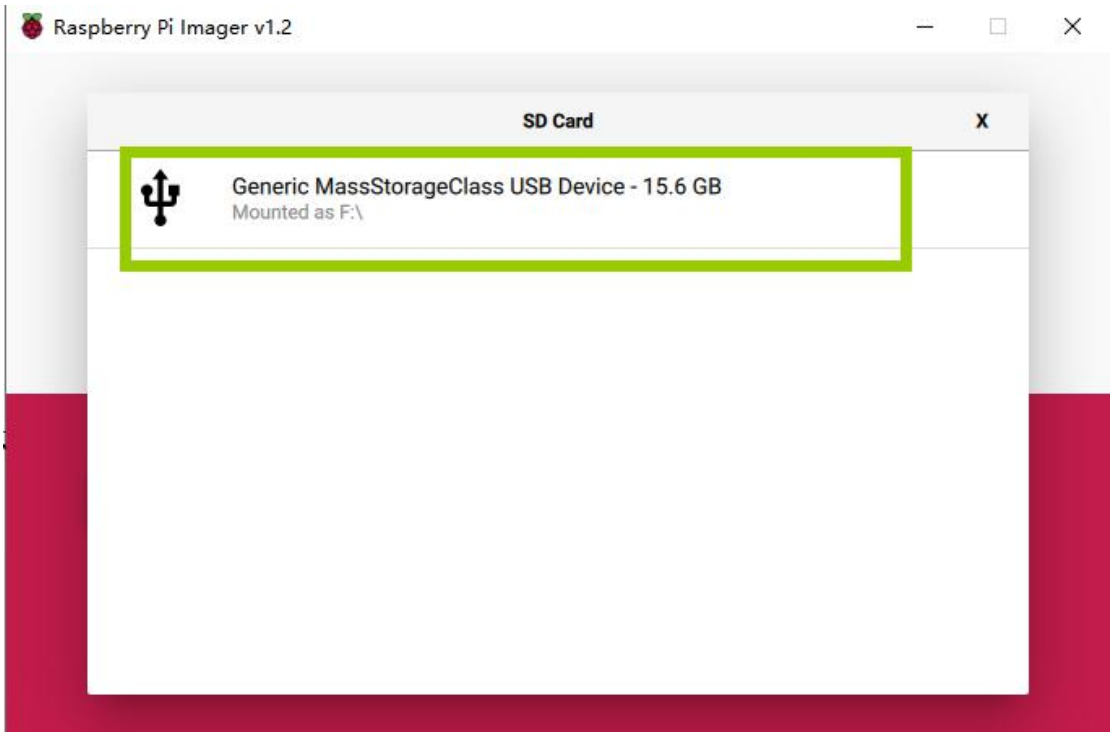


(5) Then on the interface of Raspberry Pi Imager, the ".img" file of our selected Raspberry Pi system will appear.

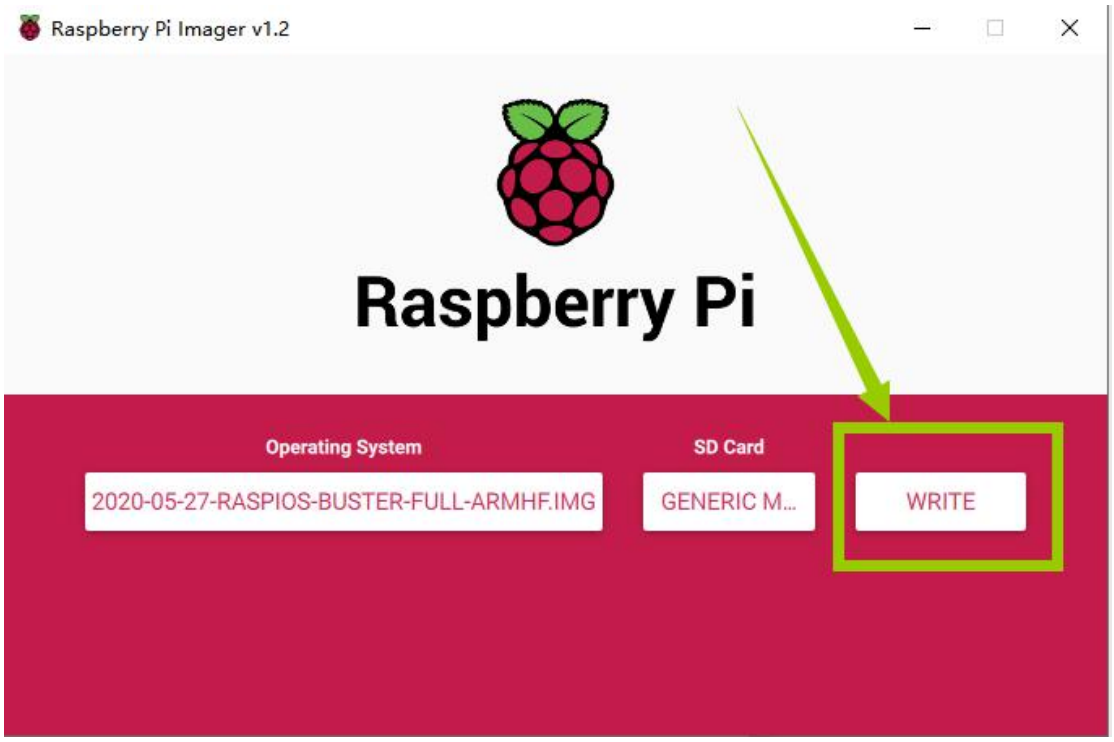
(6) Click "CHOOSE SD".



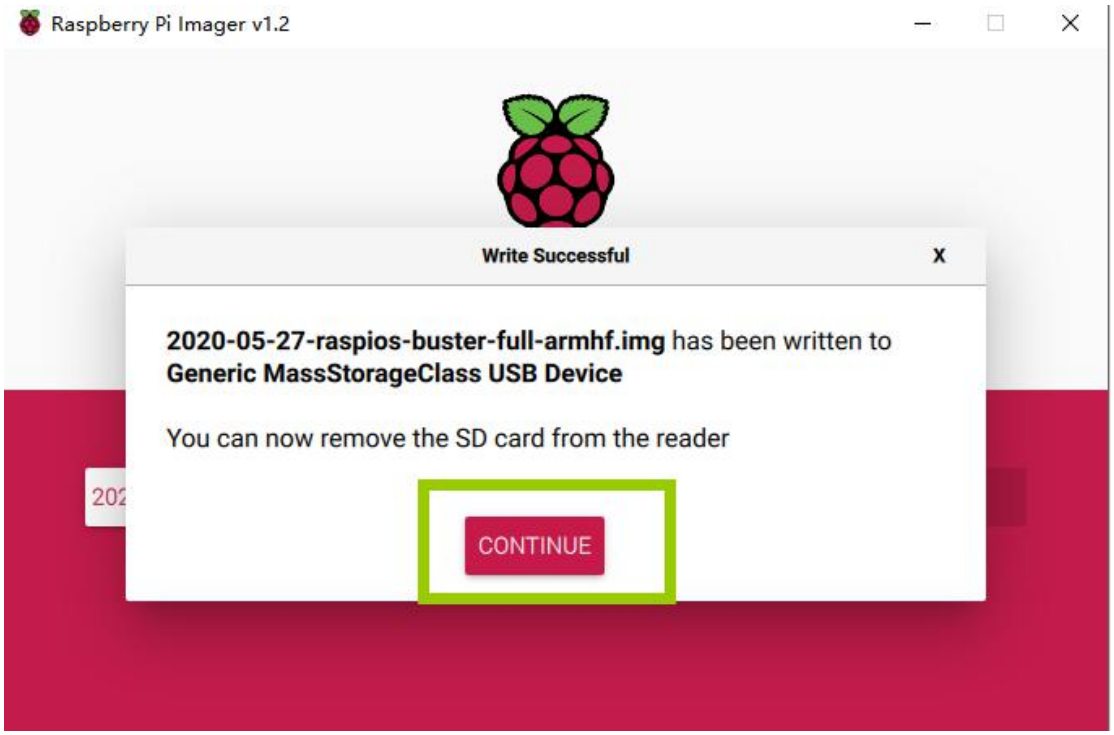
(7) Then select the SD card we need to burn.



(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】

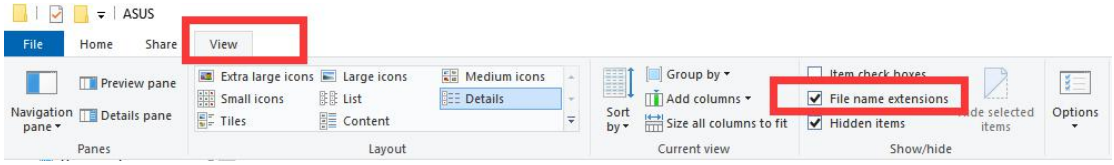
Don't remove the SD card after burning! After the Raspberry Pi Imager is burned, the memory card will be ejected in the program. This will cause the subsequent copy

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.

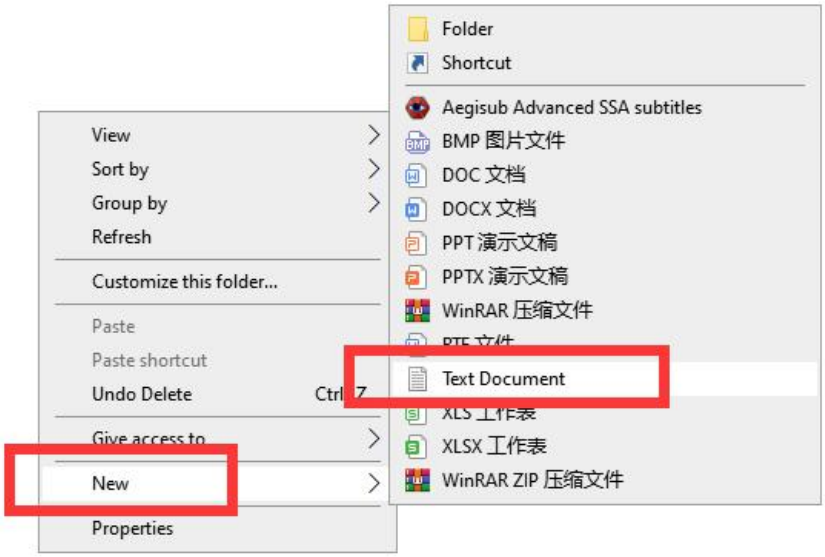
1.4 Starting the Raspberry Pi SSH service

SSH is a protocol designed to provide security for remote login sessions and other network services. Through the SSH service, you can remotely use the command line of the Raspberry Pi on another machine. In the subsequent operations and the process of using the Raspberry Pi, you can control the Raspberry Pi through another machine in the same local area network without connecting the mouse, keyboard and monitor to the Raspberry Pi. After 2016, Raspbian distributions disable the SSH service by default, so we need to manually enable it.

(1) We first enter the driver D of the computer, click "View" in the upper left corner, and select "File Extension", as shown below:



(2) Right-click on the blank space of the D drive, select "New", and select "Text Document".



(3) Name the file "ssh", as shown below:

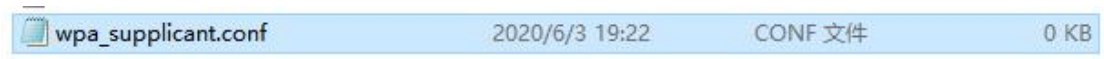
(4) Then delete the suffix ".txt". We will get an ssh file without any extension. As shown below:

(5) Copy this ssh file to the root directory of the SD card of the Raspberry Pi system. When the Raspberry Pi starts, it will automatically find this ssh file. If it is found, it will start SSH. This method only needs to be used once. After that, every time you start the Raspberry Pi, it will automatically start SSH without repeating the above operations. Copy the ssh file to the Raspberry Pi as shown below:

1.5 Setting up Raspberry Pi WIFI wireless connection

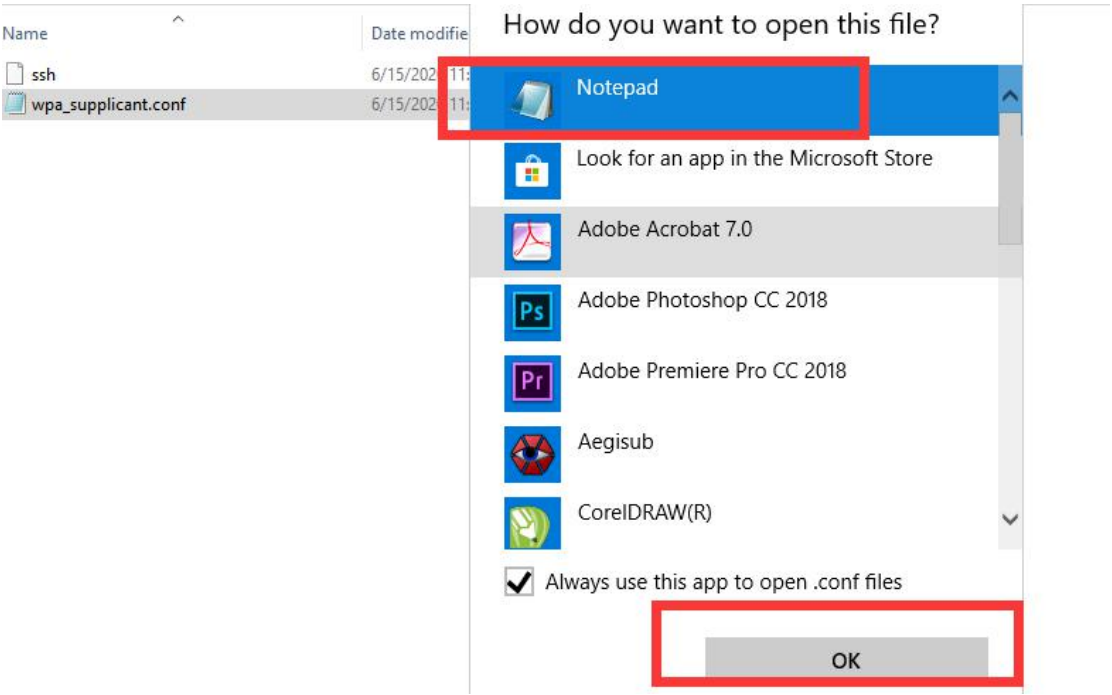
Next, we also need to set up a WIFI wireless connection for the Raspberry Pi.

(1) Create a new file named wpa_supplicant.conf in the root directory of the D driver of the computer.



(2) Click to select the wpa_supplicant.conf file, right-click the mouse, and select "Open Mode (H)".

(3) Select "Notepad" to open it.

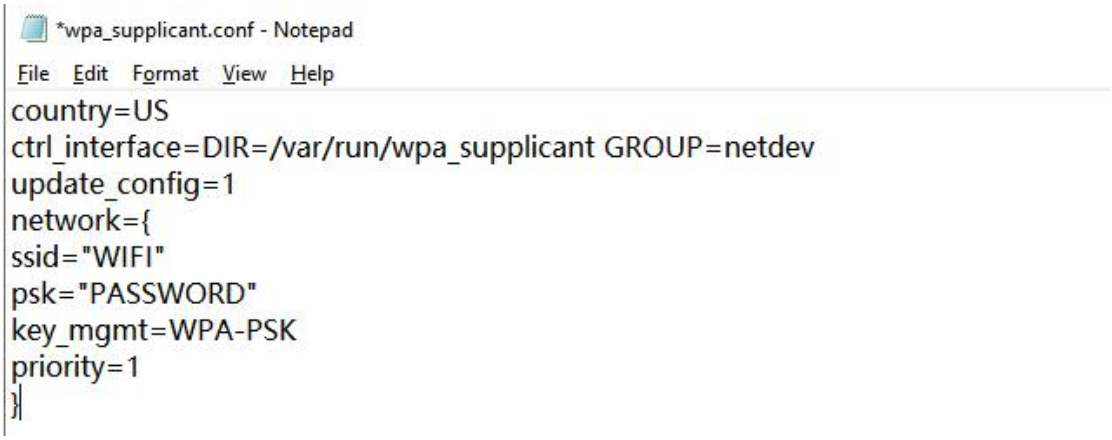


(4) Write the following contents:

country=US

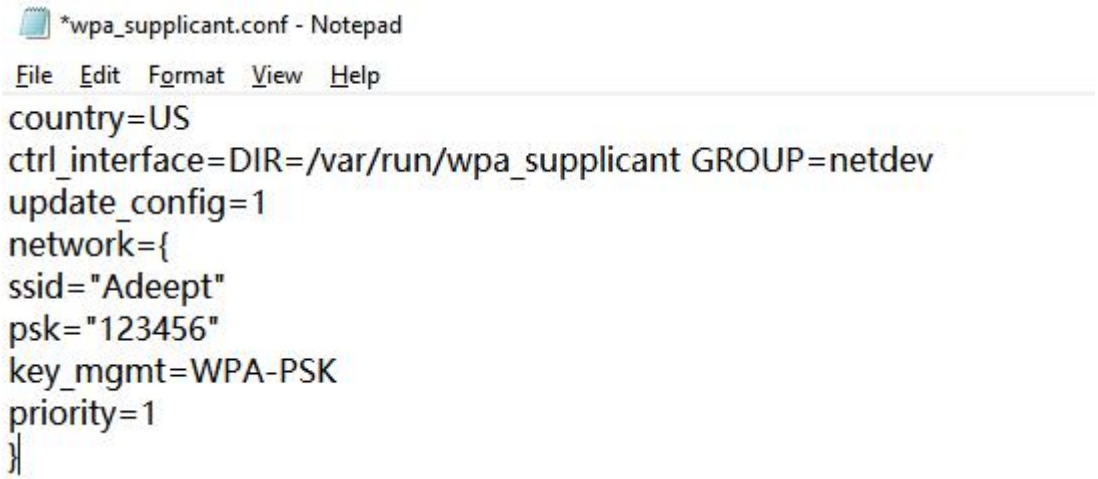
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev

```
update_config=1
network={
ssid="WIFI"
psk="PASSWORD"
key_mgmt=WPA-PSK
priority=1
 }
```



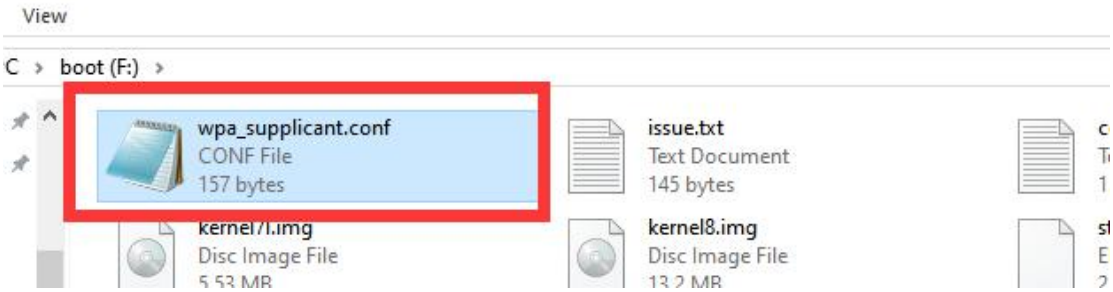
"Country" is your country code, do not modify it, the default is US; "ssid" needs to be changed to the name of the WIFI you want to connect; "psk" needs to be changed to the password of the WIFI you want to connect; other parts do not need to do any modifications.

For example, our company's WIFI name is Adeept, WIFI password is 123456, and the modified wpa_supplicant.conf file is as shown below:

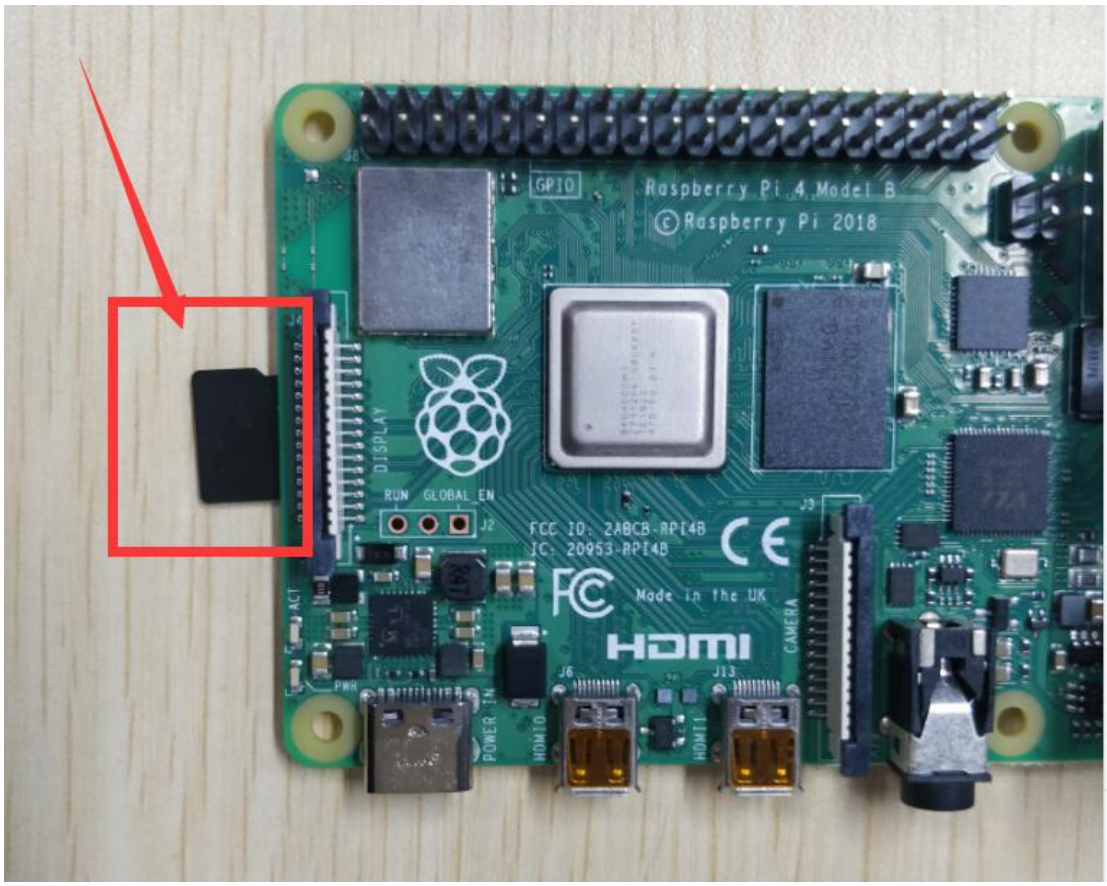


(5) Save the set wpa_supplicant.conf file, and then copy it to the root directory of

the SD card of the Raspberry Pi system. As shown below:



(6) Now we can take out the SD card and put it into the "MICRO SD CARD" card slot on the Raspberry Pi development board, and use the Type-C data cable to supply power to the Raspberry Pi. And then the Raspberry Pi will start up and run.



1.6 Remotely logging in to the Raspberry Pi system

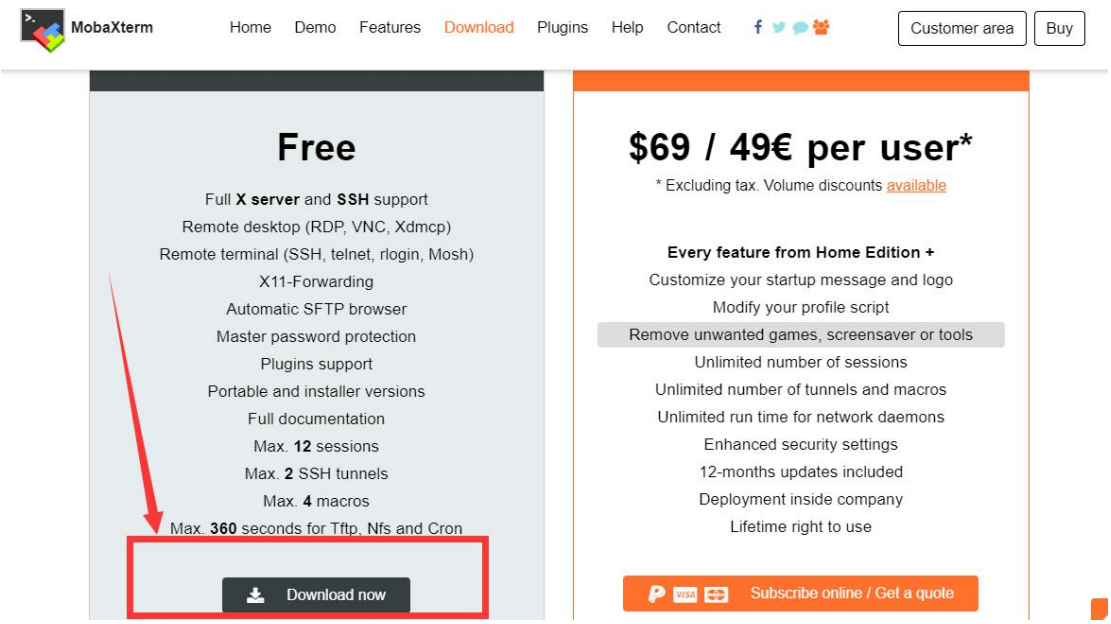
1.6.1 Download and install MobaXterm software

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

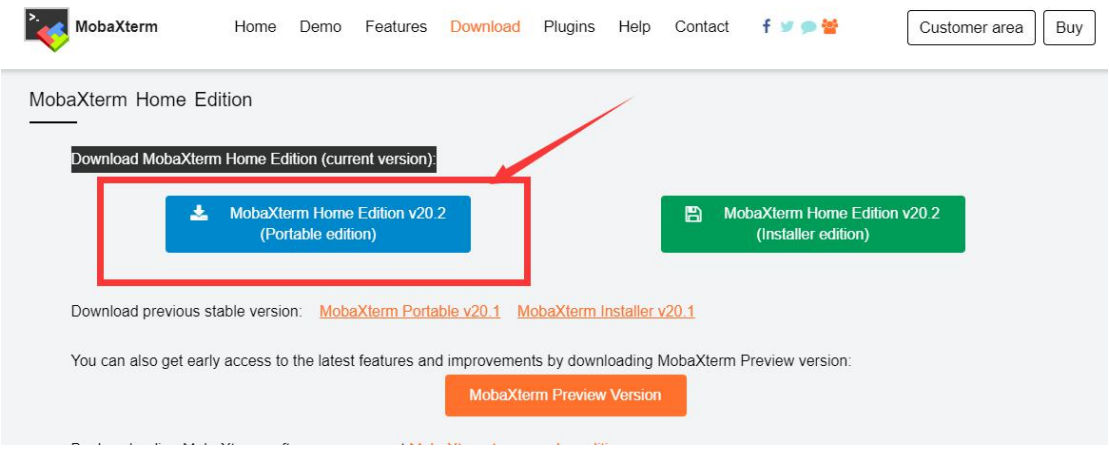
(1)Log in to the official website through a browser to download:

https://mobaxterm.mobatek.net/download.html.

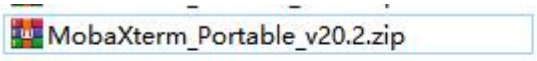
Select Free version to download.



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

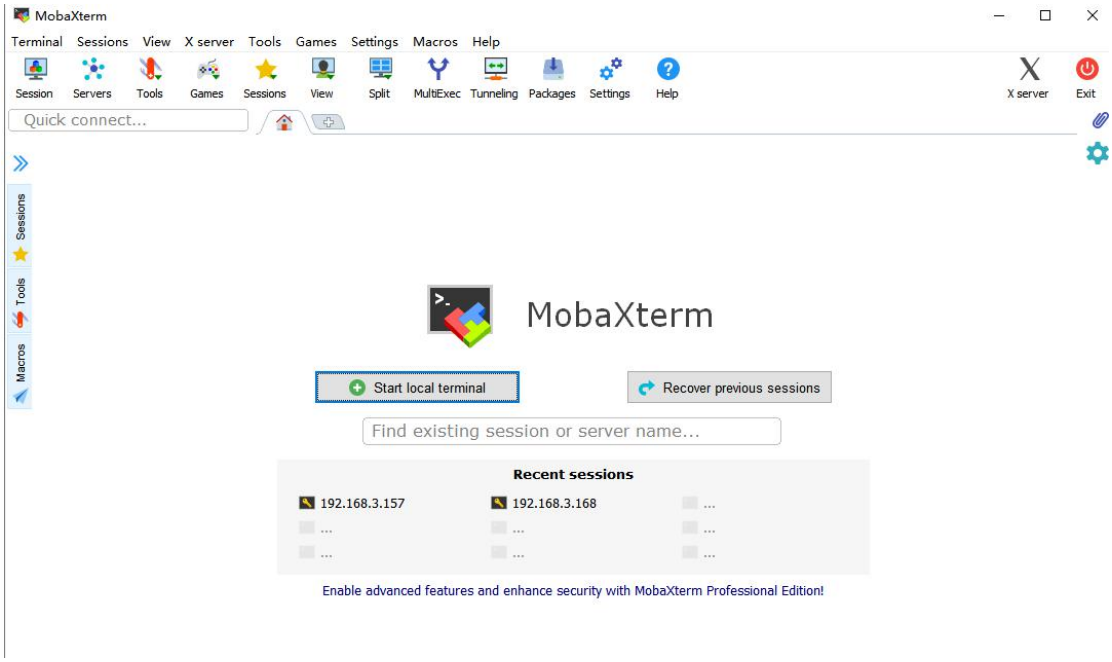


(3) Find the downloaded file MobaXterm_Portable_v20.2.zip, double-click to open it, extract it to get a new file



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

(5) Double-click to open MobaXterm_Personal_20.2.exe to directly open the MobaXterm software. The interface is as shown below:



1.6.2 Obtaining the IP address of the Raspberry Pi

1.6.2.1 Using the Pi with a display

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

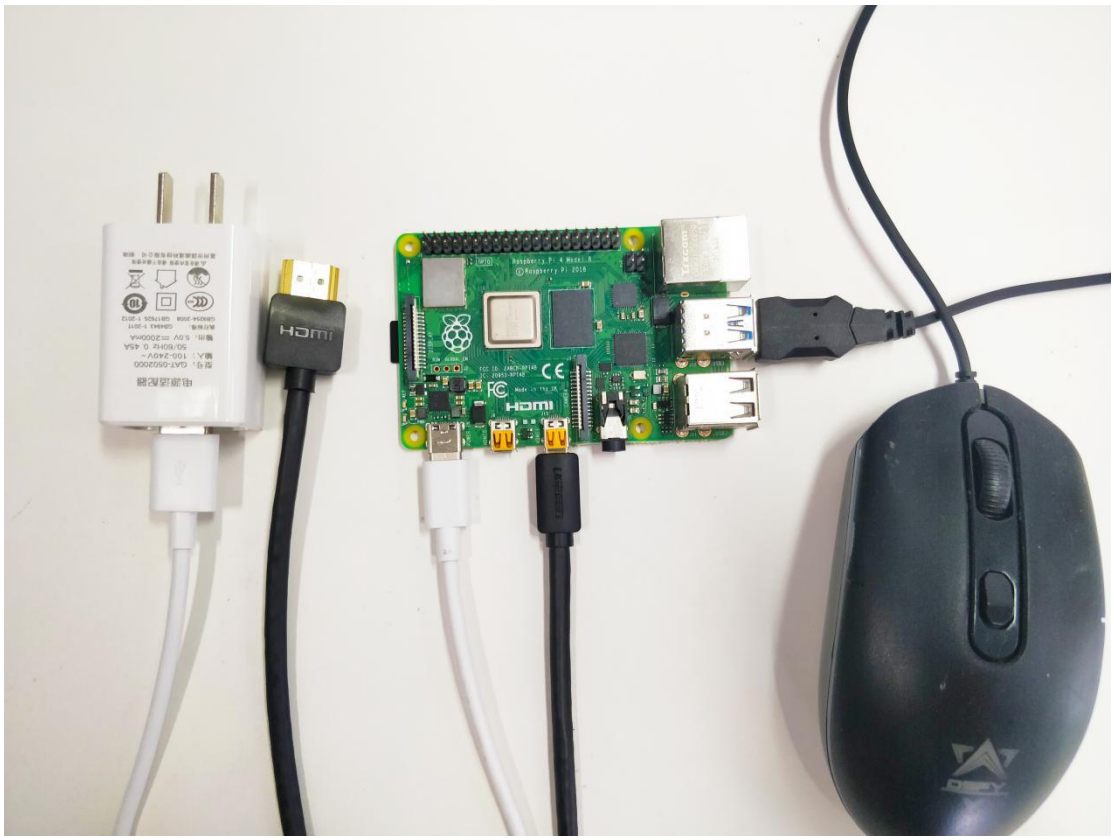
(1) One Type-C data cable: used to supply power to the Raspberry Pi.

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

(3) One mouse: used to operate.

(4) One monitor

(5) One Raspberry Pi



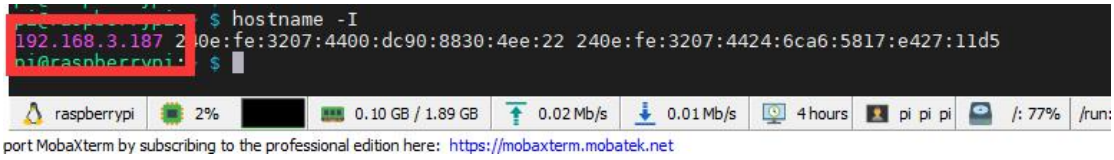
Connect the HDMI cable to the HDMI port of the monitor:

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.

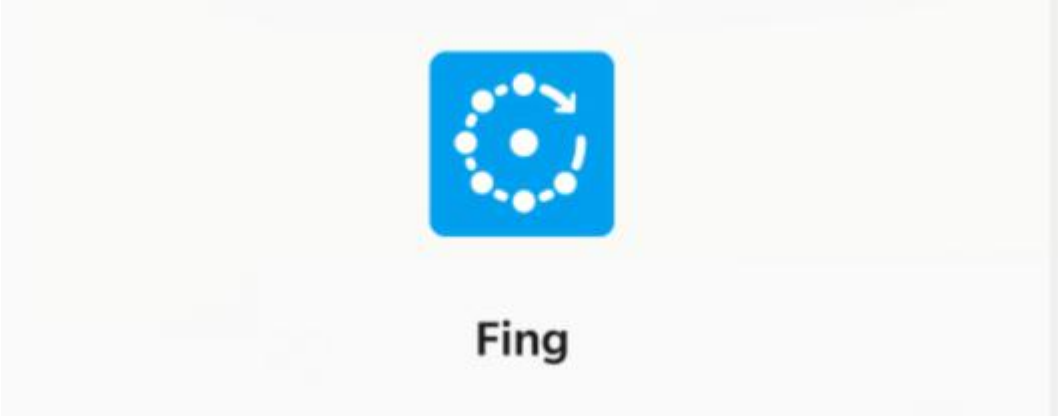
2. You can also open the command window of the Raspberry Pi and enter the following command to view the following IP address, you need to write it down:

hostname -I

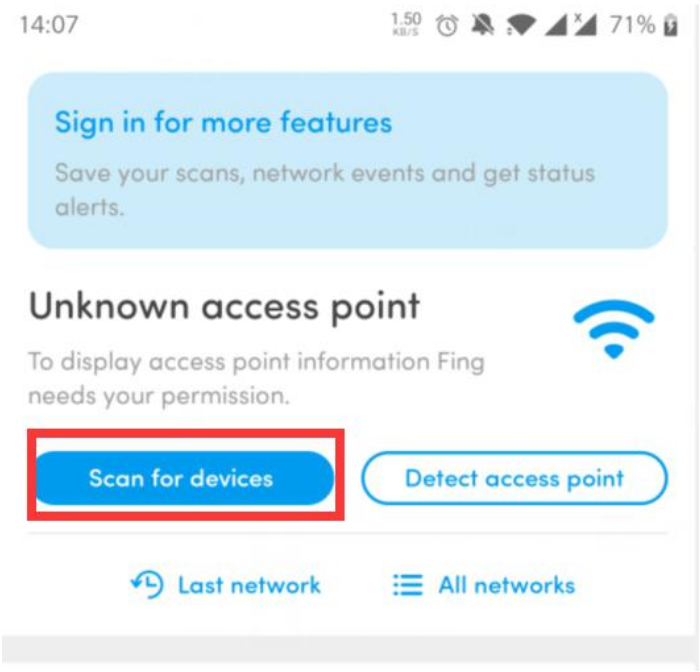


1.6.2.2 Getting the IP address of a Pi using your smartphone)

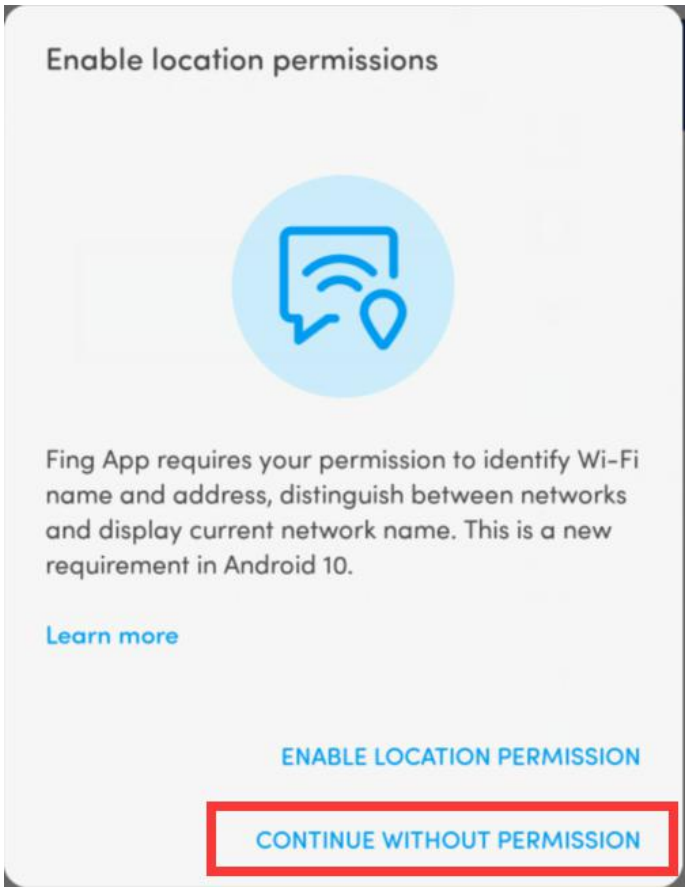
1. You need to download an APP called "Fing" on your phone, as shown below:



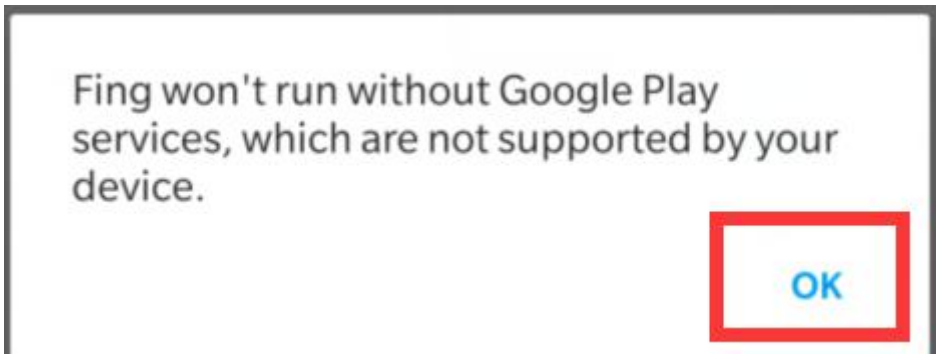
2. After the download is complete, your phone and Raspberry Pi need to be in the same local area network, that is, your phone and Raspberry Pi are connected to the same WIFI, then open "Fing" and click "Scan for devices":



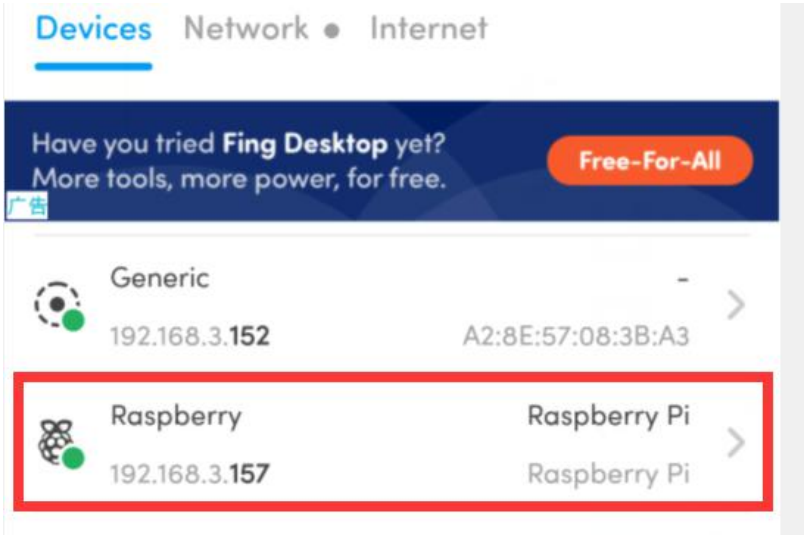
Click“CONTINUE WITHOUT PERMISSION”：



Click OK：



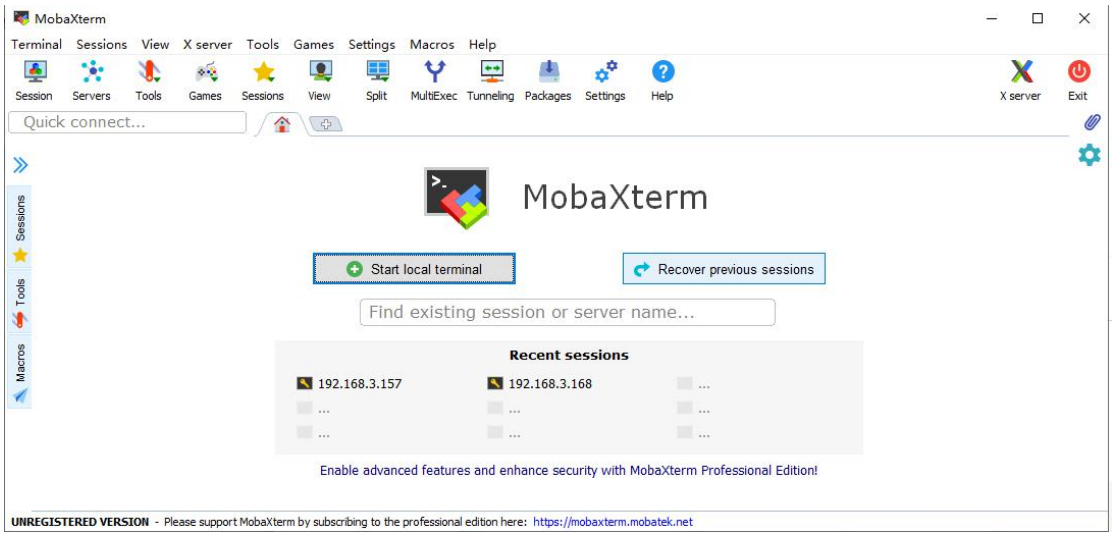
3. Wait for the scan to complete. In the list, you find a device named "Raspberry Pi". In the lower left corner, you will see the IP address of the Raspberry Pi: 192.168.3.157. You need to write down this IP address.



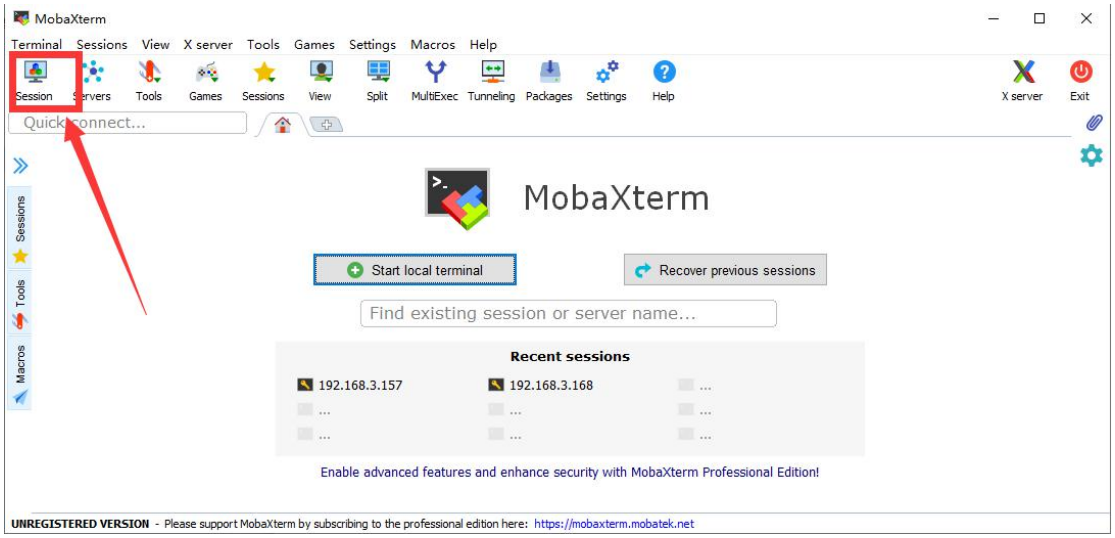
1.6.3 Remotely logging in to the Raspberry Pi system

(1) Open the software 

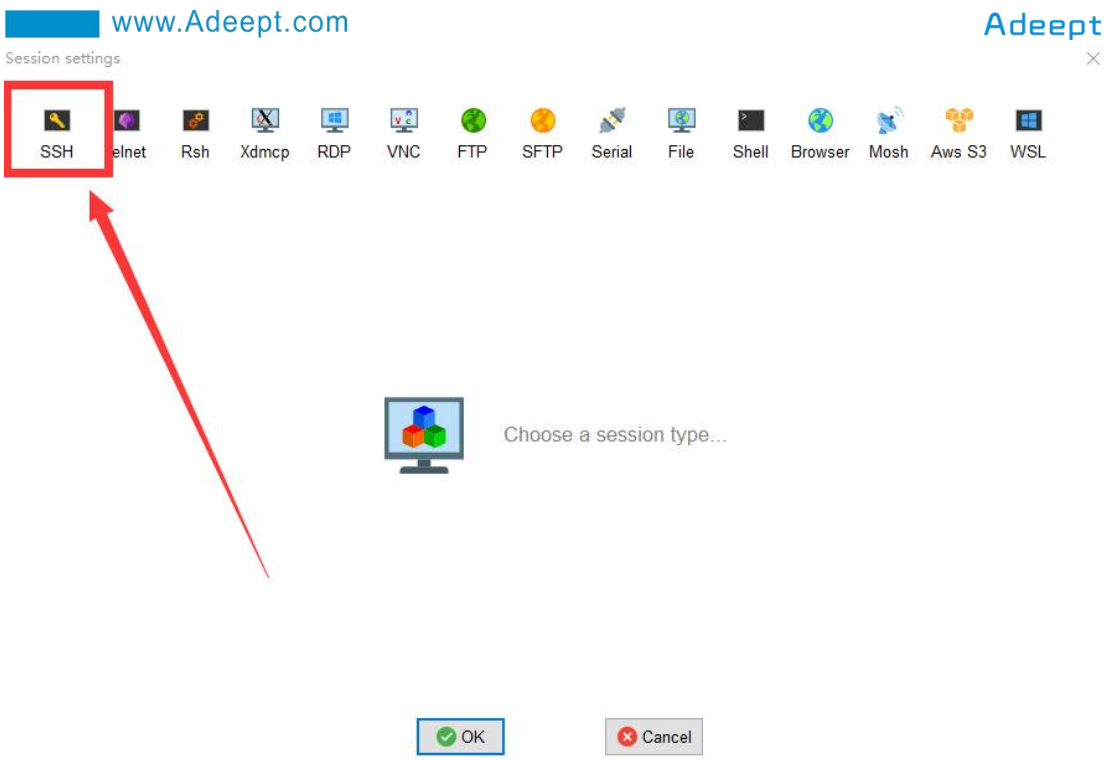
on the desktop, as shown below:



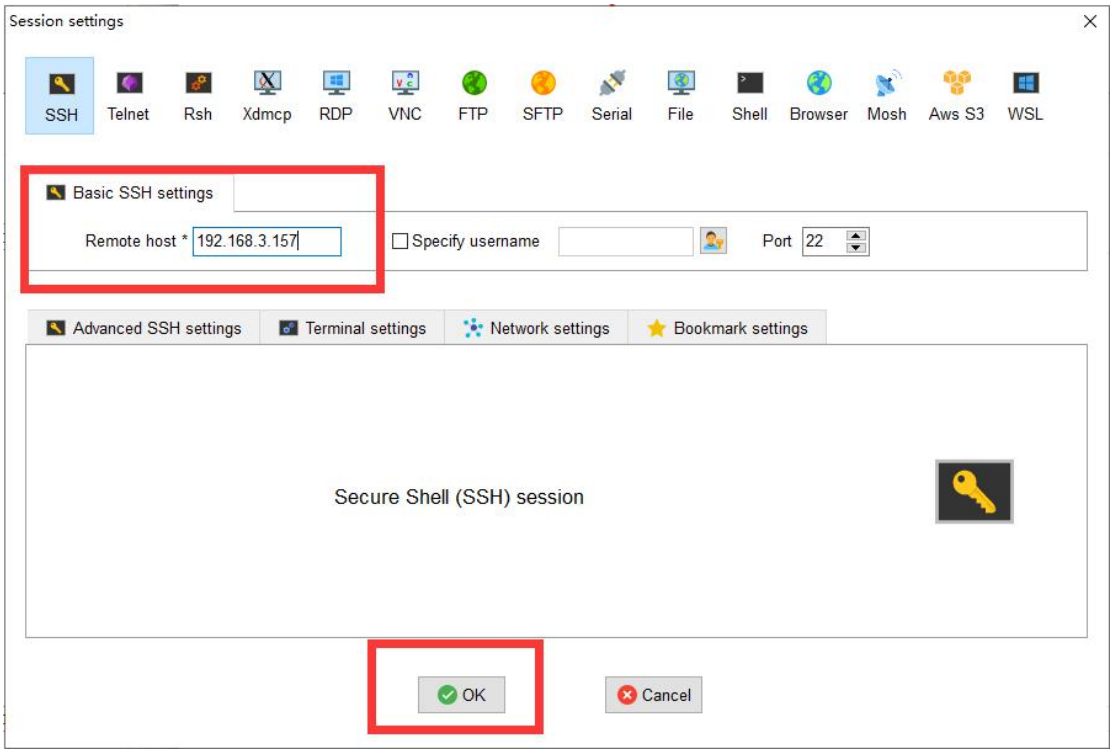
(2) Click "Session" in the upper left corner.



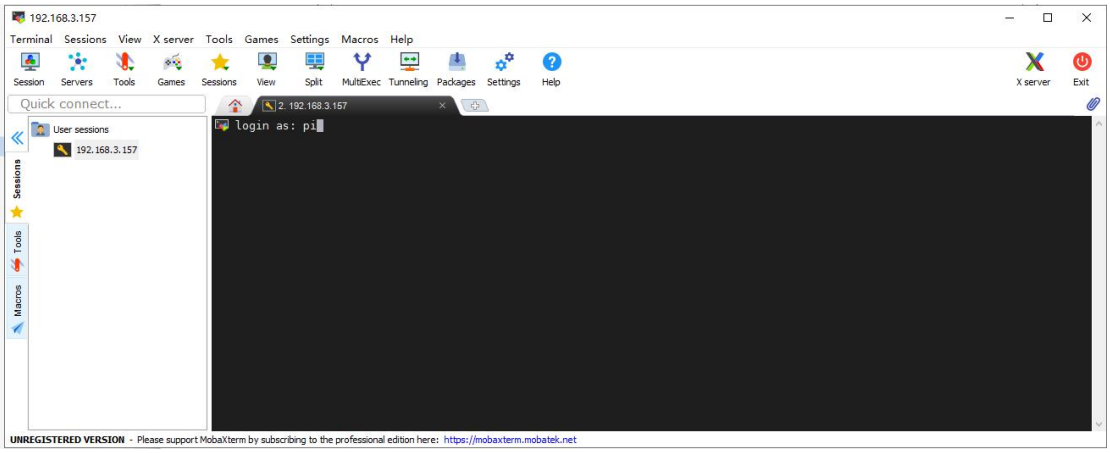
(3) Click "SSH".



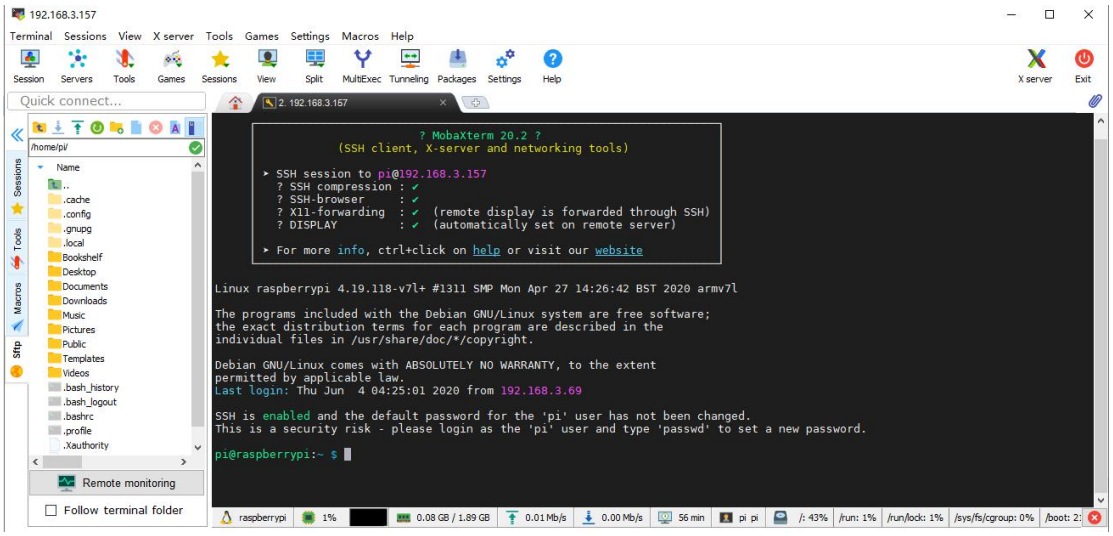
(4) Enter the IP address of the Raspberry Pi previously queried: 192.168.3.157, and confirm with "OK".



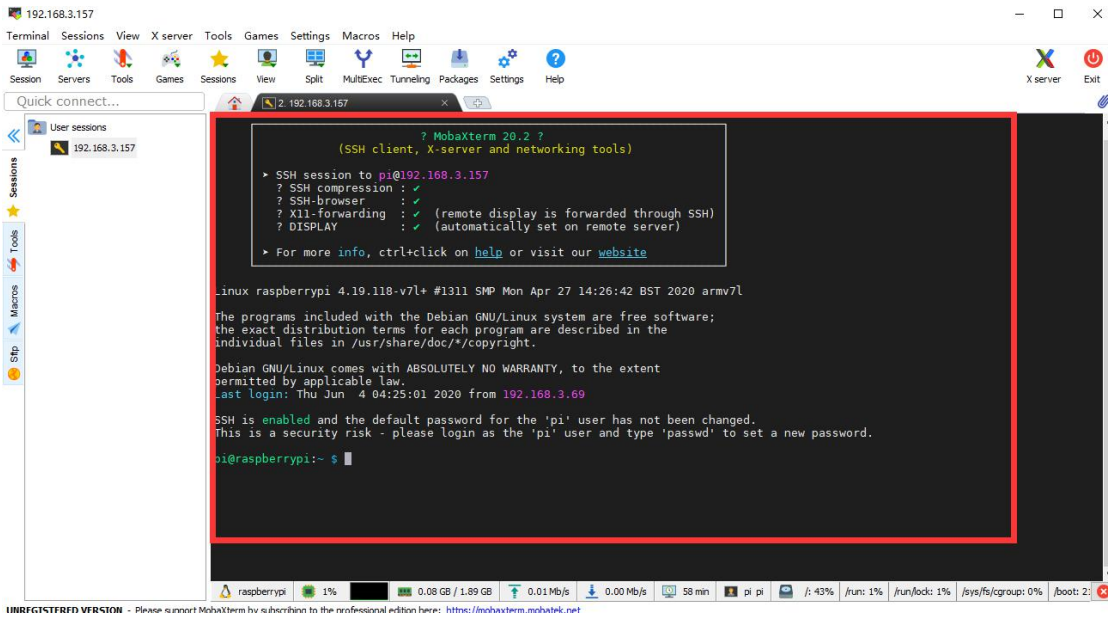
(5) Enter the default account of Raspberry Pi: pi, then press Enter, and then enter the default password of Raspberry Pi: raspberry. Press Enter to log in to the Raspberry Pi system.



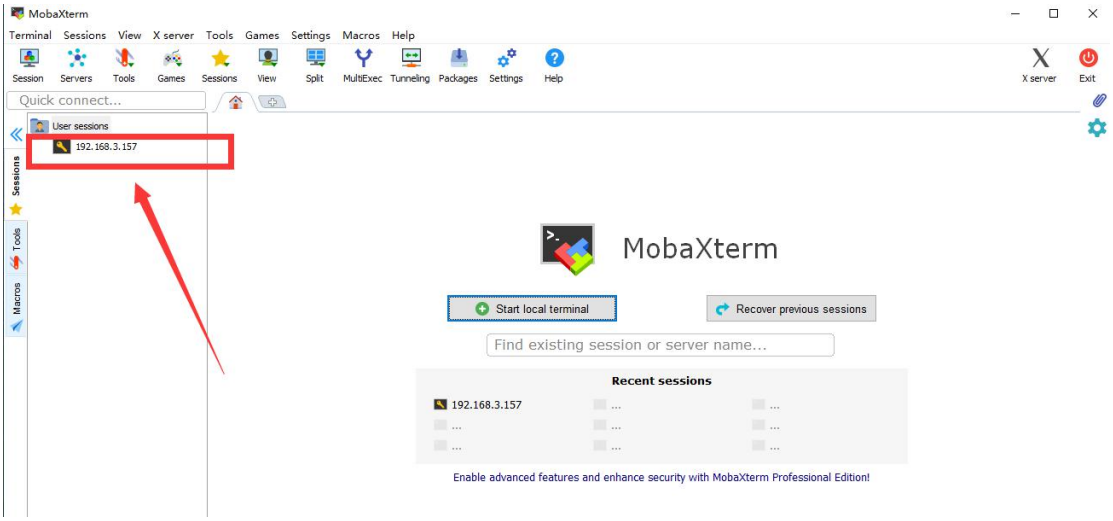
(6) After successfully logging into the Raspberry Pi system, the following interface will appear as shown below:



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



(8) When we close the MobaXterm software and then open it to connect to the Raspberry Pi, we can double-click the IP address under the "User sessions" on the left: 192.168.3.157, enter the account name: pi. Then the Raspberry Pi will be directly connected.



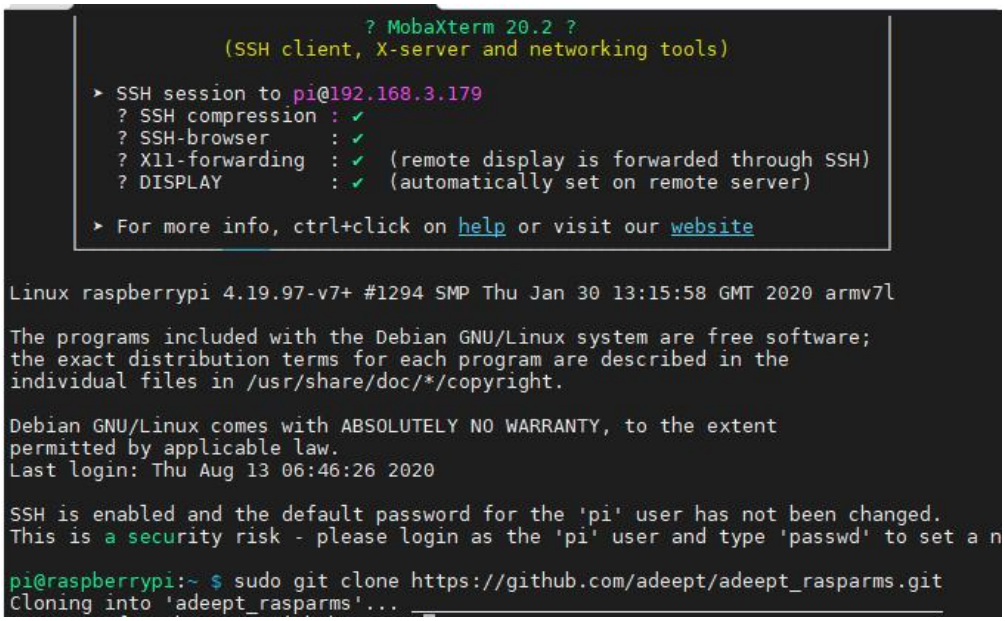
Lesson 2 Downloading and Installing the Relevant Code Program of the Robot

All the code programs of our robot products have been open sourced on GitHub. You need to download them to the Raspberry Pi and install the relevant dependencies before they can run normally.

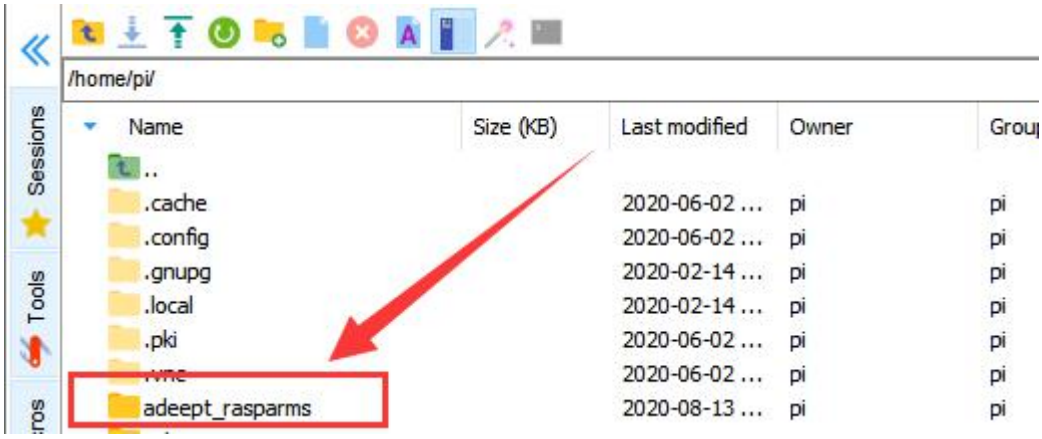
2.1 Downloading the code program to control the robot

After successfully logging in to the Raspberry Pi, we need to download the code program to control the robot, and enter the following command in the console:

sudo git clone https://github.com/adeept/adeept_rasparms.git



After successfully downloading, you will see a new folder adeept_rasparms in the file resource list on the left. This folder stores some very important program codes. We will teach you how to use it in detail in the following courses.



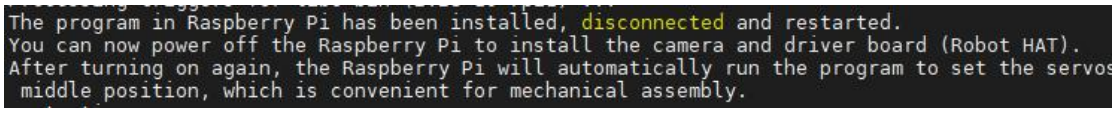
2.2 Installing the relevant dependency libraries of the robot

1. We have prepared a script program (setup.py) to install the dependent libraries needed by the robot and set up operations such as turning on the camera and automatically running at startup. This script program is in the directory of the adeept_rasparms folder you downloaded. You need to enter the following command in the command window of the Raspberry Pi to run it:

sudo python3 adeept_rasparms/setup.py

Press the "Enter" button and wait for the download and installation to complete.

2. After the download is complete, the following prompt will appear:



3. After the installation is complete, the Raspberry Pi will automatically disconnect the SSH connection and restart. At this time, if you are using the Raspberry Pi to connect with Putty, MobaXterm and other software, there will be an error message such as Network error: Software caused connection abort.Just close the

prompt.

2.3 Running the program of the Raspberry Pi robot

Raspberry Pi will automatically run the program of the robot product every time it restarts. Specifically, every time it restarts, it will run [RobotName]/server/webServer.py. Replace [RobotName] with the name of your robot product program folder, but if without connecting the Raspberry Pi camera and the driver board RobotHAT, webServer.py will not run successfully. This is a normal phenomenon, because the program of the robot product needs to use the camera and the PCA9685 chip. Our RobotHAT uses PCA9685 to control the servo.Raspberry Pi communicates with PCA9685 with I2C. If RobotHAT is not installed on Raspberry Pi, communication failure will occur when instantiating PCA9685's dependent library, which will cause program error.

We can turn off the Raspberry Pi, install the camera module and RobotHAT, and turn it on again, so that webServer.py can run successfully.

Under normal circumstances, there is no need for people to manually run webServer.py, because every time the Raspberry Pi is turned on, webServer.py will be run automatically.

You can open Google Chrome, enter the IP address of the Raspberry Pi and add: 5000 after it, press enter to jump to the Raspberry Pi web page, an example is shown below:

http://192.168.3.157:5000

If the page fails to open, you can log in to the Raspberry Pi by using SSH, and use the following command to end the robot program that automatically runs on startup to release resources. Otherwise, problems such as camera initialization failure or port occupation will occur. Enter the following command in the command window:

```
sudo killall python3
```

Use the following command to manually run webServer.py:

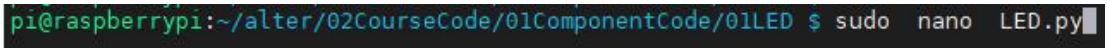
```
sudo python3 adeept_rasparms/server/webServer.py
```

2.4 Editing the code program in Raspberry Pi

To view and edit the code program in the Raspberry Pi on the MobaXterm terminal software, we provide two methods for reference.

The first is to use Linux commands. For example, if you need to view the code program LED.py in this section of the course, then you can enter the following commands in the console of the MobaXterm terminal:

sudo nano LED.py



1. In this way, you can view our code program. You can modify and edit the code with the direction keys on the keyboard to realize the control function you want. But you need to learn and search about the command operation method of Linux nano. Here we provide several common operation commands about nano:

1. To save the modified program, you need to press the shortcut key Ctrl+O on the keyboard

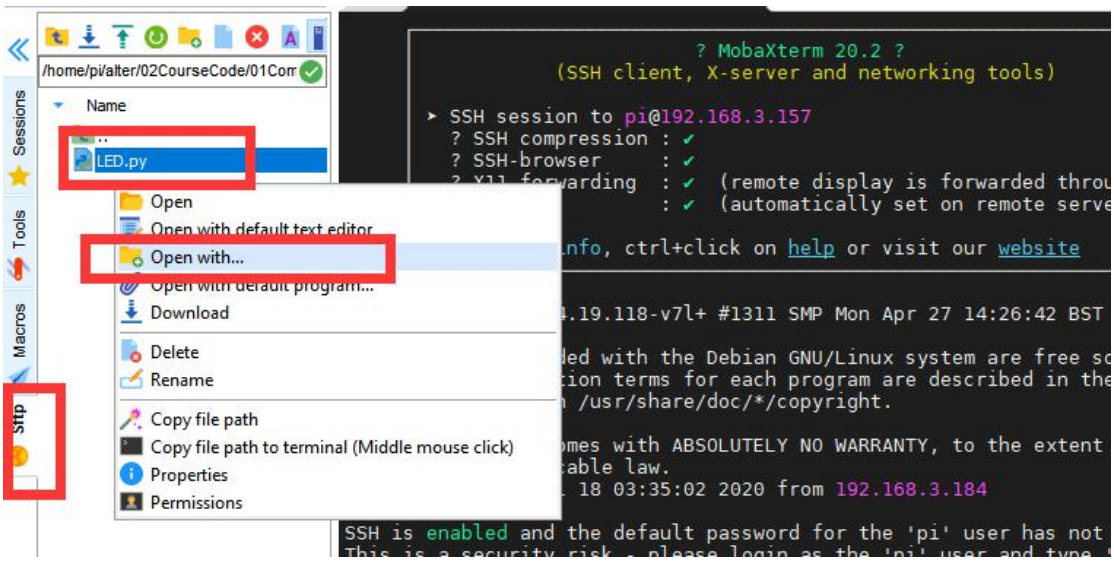
2. To exit the code editing interface, you need to press the shortcut key Ctrl+X on the keyboard

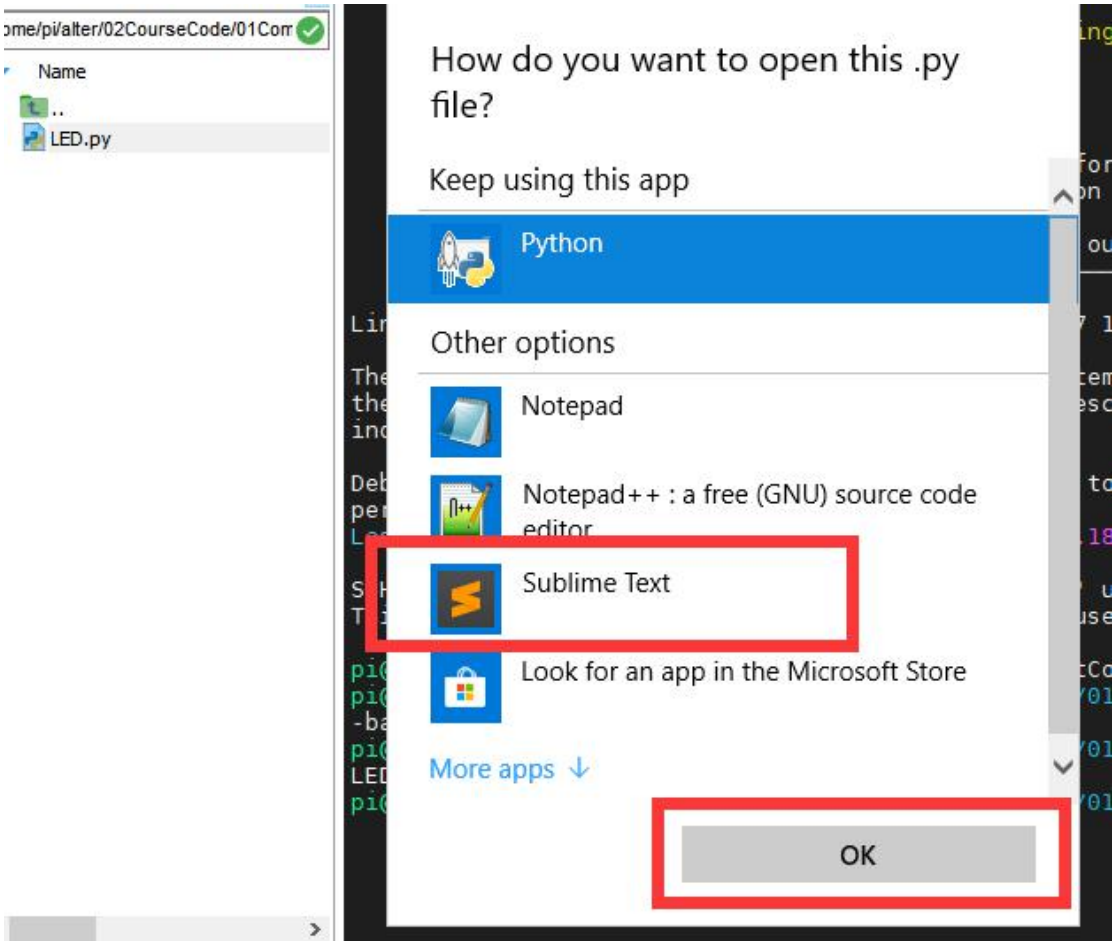
3. To cut a line of code, you can use Ctrl+K

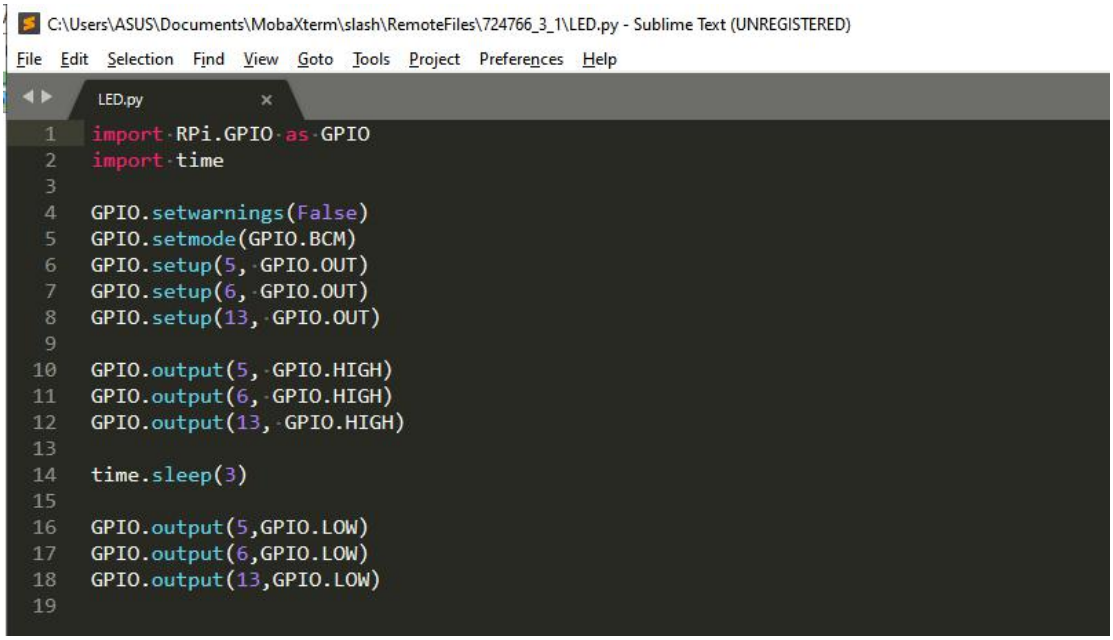
4. Paste the code, you can use Ctrl+U

```
 GNU nano 3.2                    LED.py                         Modified

import RPi.GPIO as GPIO
import time

GPIO.setwarnings(False)
GPIO.setmode(GPIO.BCM)
GPIO.setup(5, GPIO.OUT)
GPIO.setup(6, GPIO.OUT)
GPIO.setup(13, GPIO.OUT)

GPIO.output(5, GPIO.HIGH)
GPIO.output(6, GPIO.HIGH)
GPIO.output(13, GPIO.HIGH)

time.sleep(3)

GPIO.output(5,GPIO.LOW)
GPIO.output(6,GPIO.LOW)
GPIO.output(13,GPIO.LOW)

^G Get Help  ^O Write Out  ^W Where Is  ^K Cut Text  ^J Justify  ^C Cur Pos  M-U Undo  M-A Mark Text
^X Exit      ^R Read File  ^\ Replace   ^U Uncut Text ^T To Spell  ^_ Go To Line M-E Redo  M-6 Copy Text
```

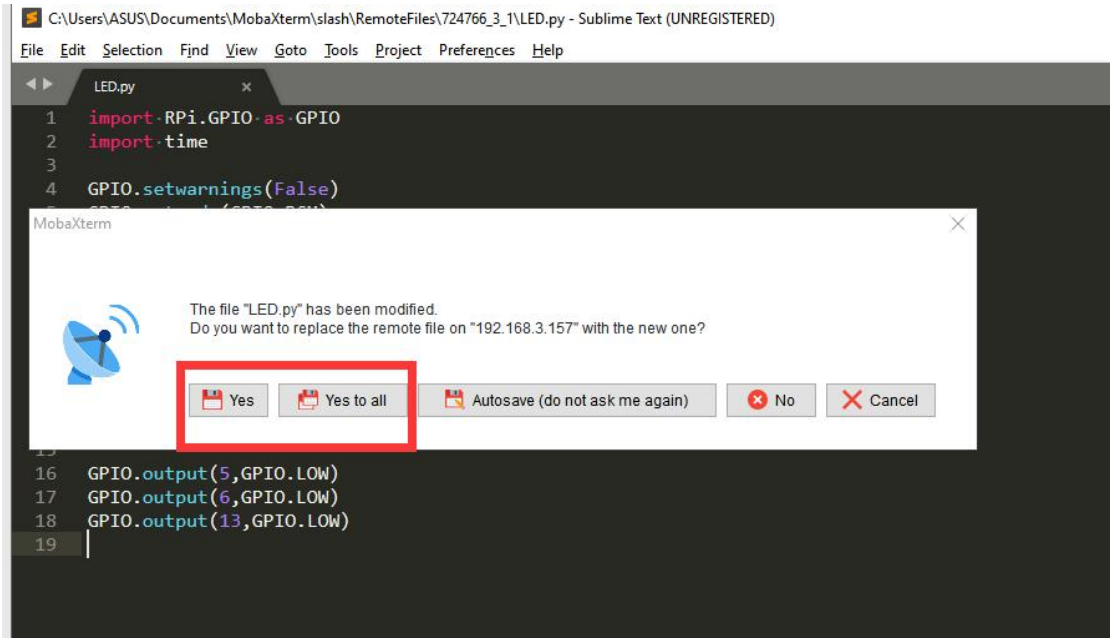
The second method is to use a third-party IDE tool to view and edit the code program of this lesson. You can find the file you want to edit in the file resource management system on the left side of MobaXterm, right-click the file, and click Open with , Choose your IDE, we recommend you to use Sublime Text IDE, you need to download it to edit files in the Raspberry Pi After editing: http://www.sublimetext.com/3. In this way, you can use your favorite IDE to edit the files in the Raspberry Pi. After editing, CTRL+S can save the files.







When you need to save the modified file, you can press the shortcut key CTRL+S, and then select Yes or Yes to all.


C:\Users\ASUS\Documents\MobaXterm\slash\RemoteFiles\724766_3_1\LED.py - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
LED.py
1 import RPi.GPIO as GPIO
2 import time
3
4 GPIO.setwarnings(False)
MobaXterm
The file "LED.py" has been modified.
Do you want to replace the remote file on "192.168.3.157" with the new one?
Yes
Yes to all
Autosave (do not ask me again)
No
Cancel
16 GPIO.output(5,GPIO.LOW)
17 GPIO.output(6,GPIO.LOW)
18 GPIO.output(13,GPIO.LOW)
19

Lesson 3 Creating a WiFi Hotspot on Raspberry Pi

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

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

1.Install related dependent libraries:

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

2.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.

3.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 1234. Once the connection is successful, you can log in to 192.168 .12.1: 5000 using a browser to open the WEB application to control the robot.

4.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.

5.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:

```
sudo create_ap wlan0 eth0 Adeept 1234
```

Adeept is the name of the WIFI hotspot, 1234 is the password of the WIFI hotspot.

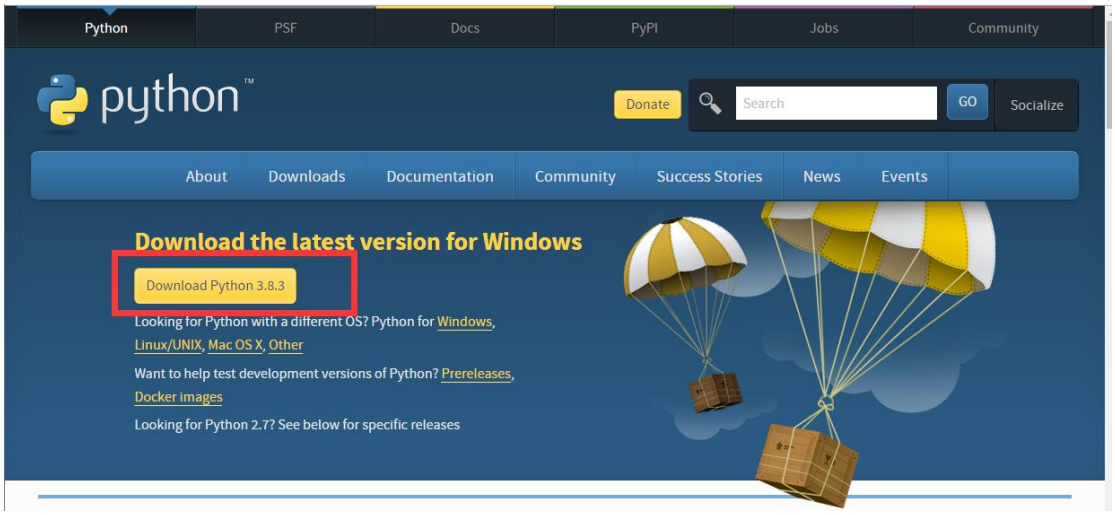
Lesson 4 Downloading and Installing Python

The course sample code programs of the RaspArm-S robot are all based on Python. Some courses need to use the Python development environment, so you need to download Python before learning the course. Below is a tutorial for downloading and installing Python on Windows 10.

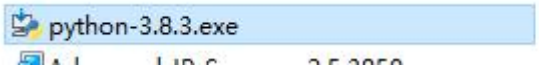
4.1 Downloading and installing Python

(1) Log in to the official website by browser: **https://www.python.org/downloads/**

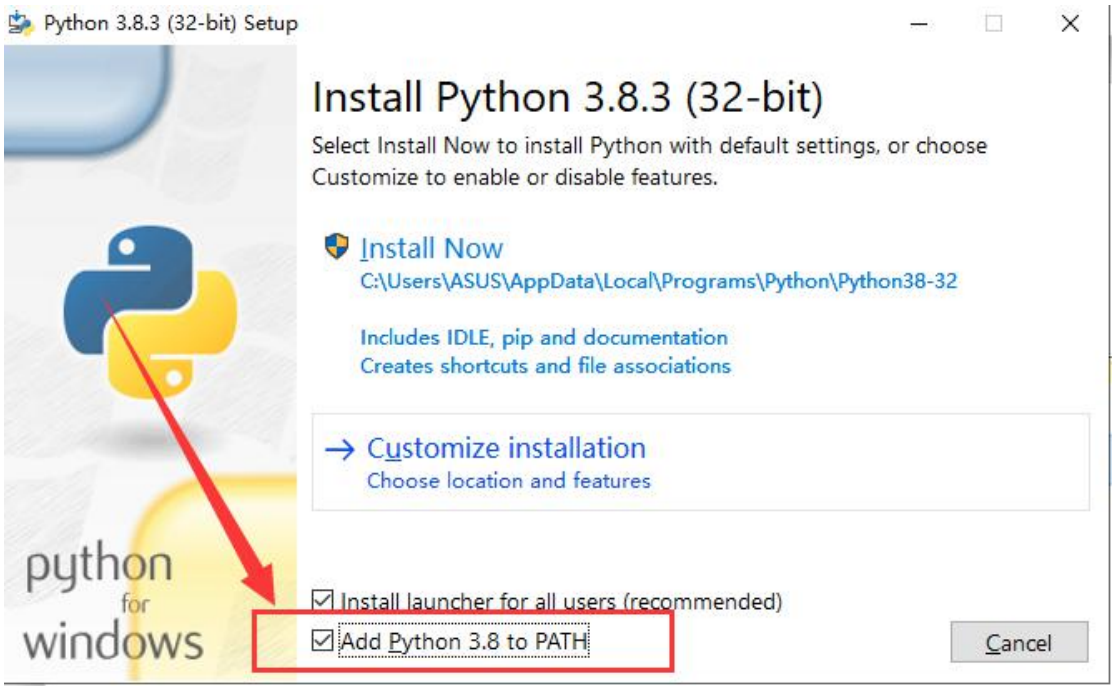
(2) Click the "Download Python 3.8.3" button to download and wait for the download to complete:



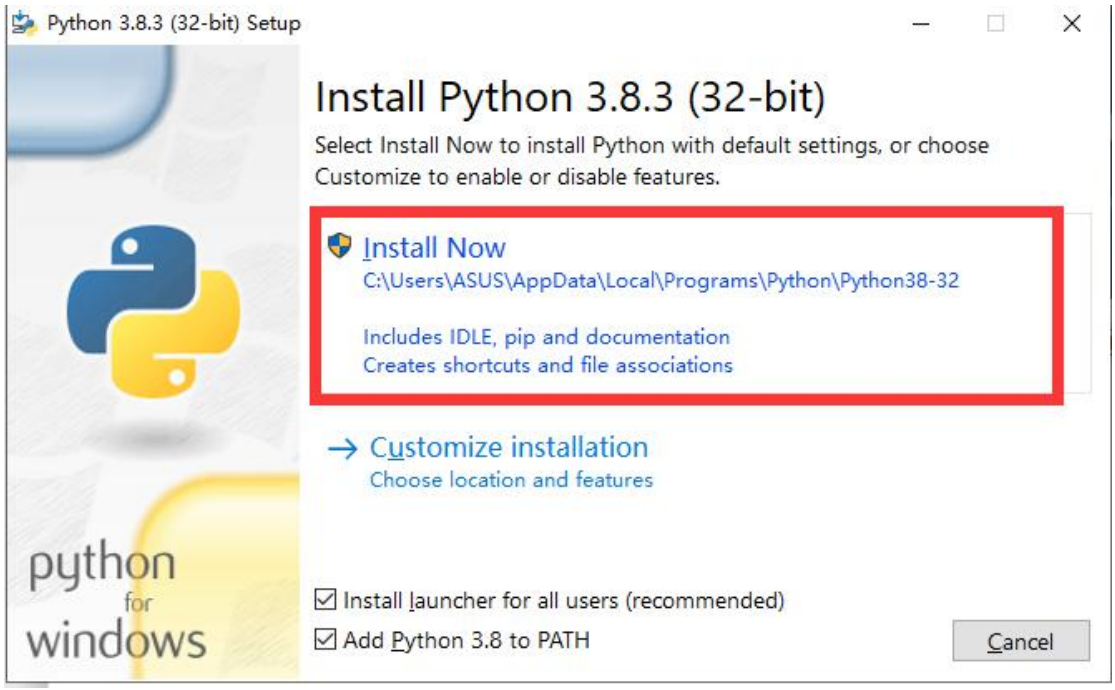
(3) Open the downloaded file, double-click to open it to install:



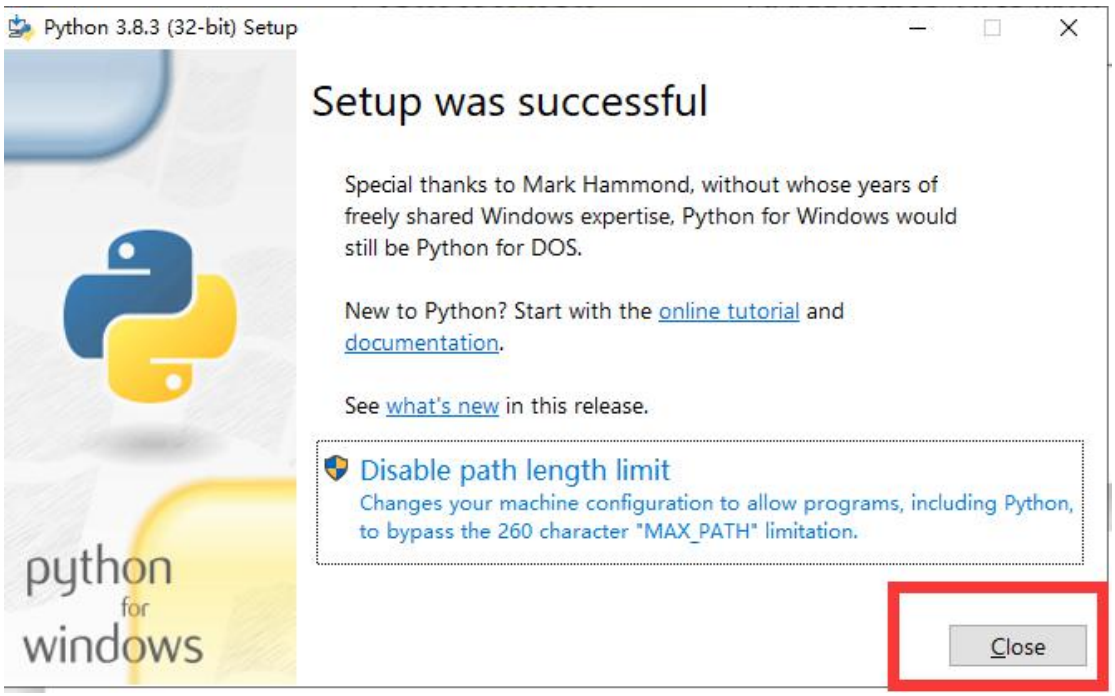
(4) Select the "Add Python 3.8 to PATH" option:



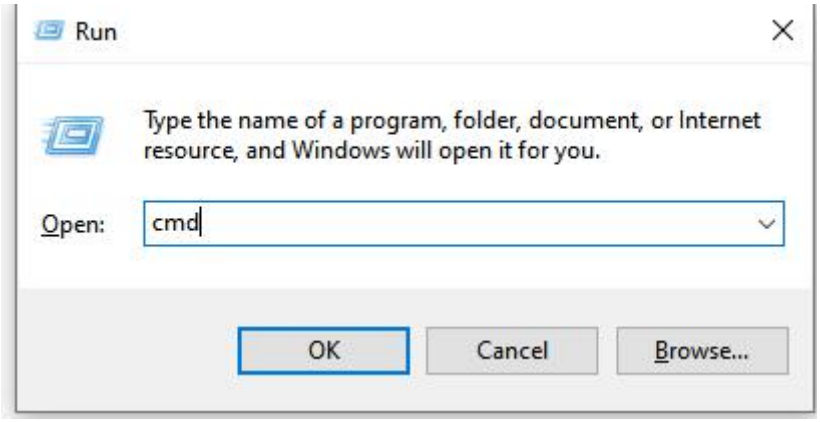
(5) Then click "Install Now" to install.



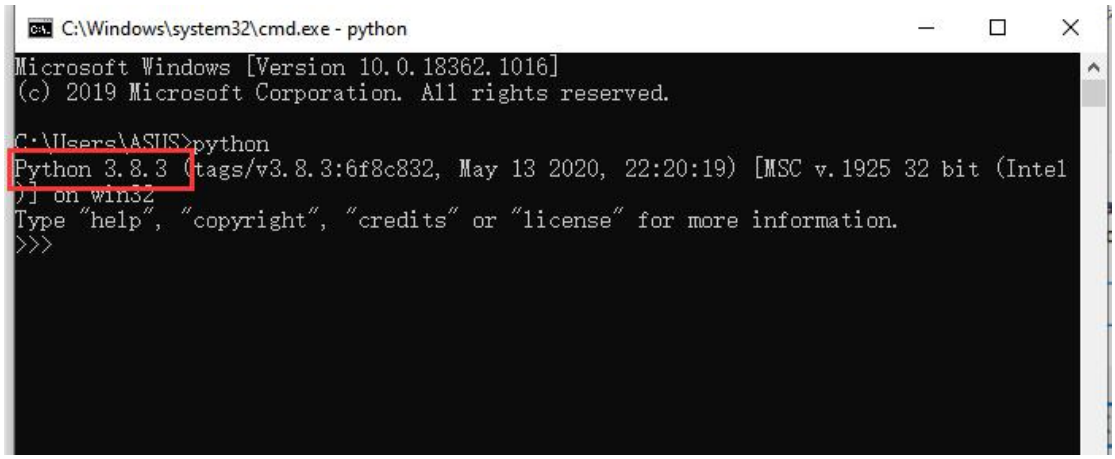
(6) Wait for the Python installation to complete and click "Close" to close.



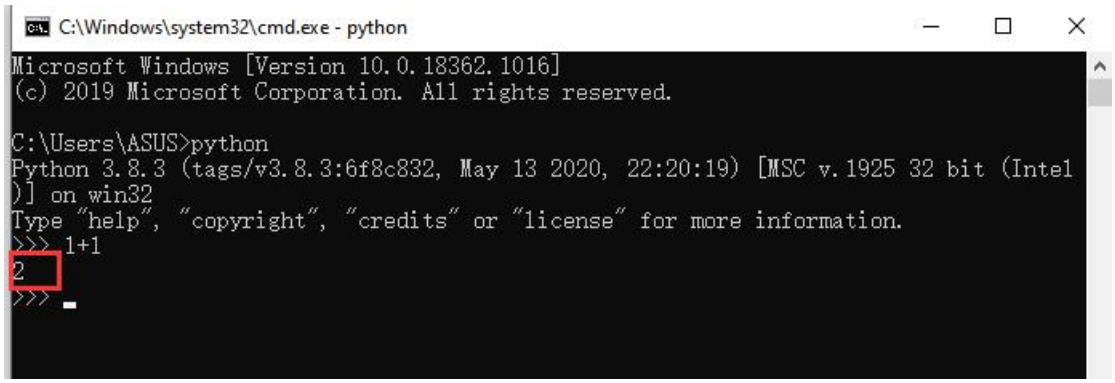
(7) Check whether Python is installed successfully. Press the shortcut key Win+R, then enter cmd in the run bar, click OK to open the command window:



(8) In the command line window, enter python, then press Enter, the version number checked is consistent with the version number downloaded and installed.



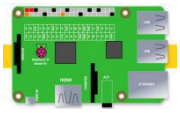
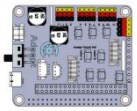
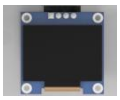
(9) Enter 1+1 to see if it can run to calculate the correct result.



Lesson 5 Displaying Text on the OLED Screen

In this lesson, we will learn how to display text on the OLED screen.

5.1 Components used in this course

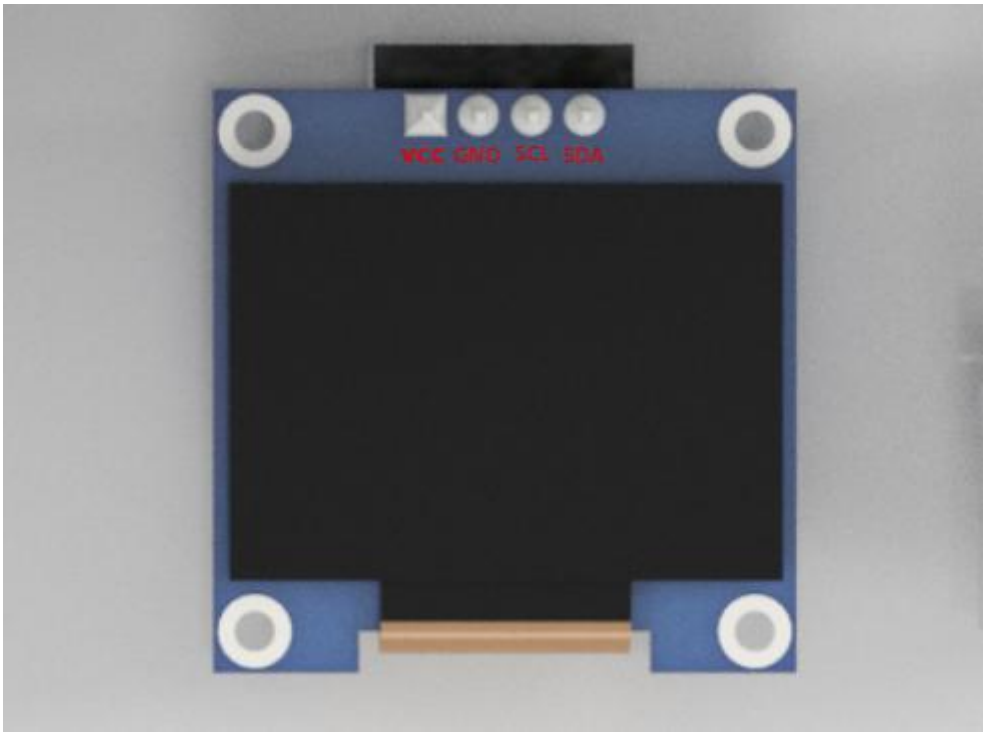
Components	Quantity	Picture
Raspberry Pi	1	
Robot HAT	1	
4 pin wire	1	
OLED screen	1	

5.2 Introduction of OLED Screen

OLED (Organic Light-Emitting Diode), also known as organic electric laser display, organic light emitting semiconductor (Organic Electroluminesence Display, OLED). OLED is a kind of current-type organic light-emitting device, which produces light by the injection and recombination of carriers, and the luminous intensity is proportional to the injected current. The RaspArm-S robot uses an OLED screen to display the expressions or some parameters of the robot. OLED Screen is a commonly used module on robot products. Due to the black non-luminous feature of OLED Screen, this type of screen has extremely high contrast. Even if the ambient light is strong, you can see the information on the OLED Screen clearly, and the power consumption is relatively low. We only need to connect the power supply and the GPIO port to control it.

If you want to use OLED Screen, you need to use a 4-pin cable with anti-reverse

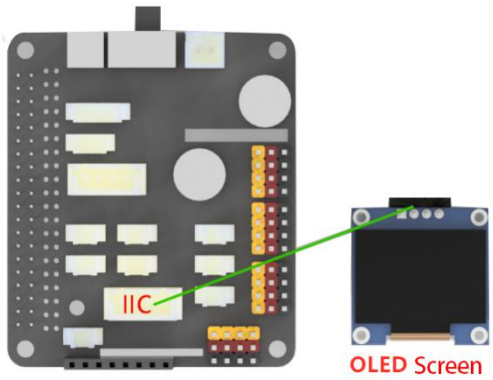
interface to connect the OLED screen to the IIC interface on the Robot HAT.



If you do not use Robot HAT driver board to connect with Raspberry Pi driver board, then you need to connect Vin of OLED screen to 5V or 3.3V of Raspberry Pi, and connect GND of OLED screen to GND of Raspberry Pi. Connect SCL of Robot HAT to SCL of OLED, and SCA of Robot HAT to SCA of Raspberry Pi. Please refer to the pin definition diagram of Raspberry Pi for specific pins.

5.3 Wiring diagram (Circuit diagram)

If you want to use the OLED Screen module, you need to connect the IIC interface on the Robot HAT driver board, as shown in the figure below:

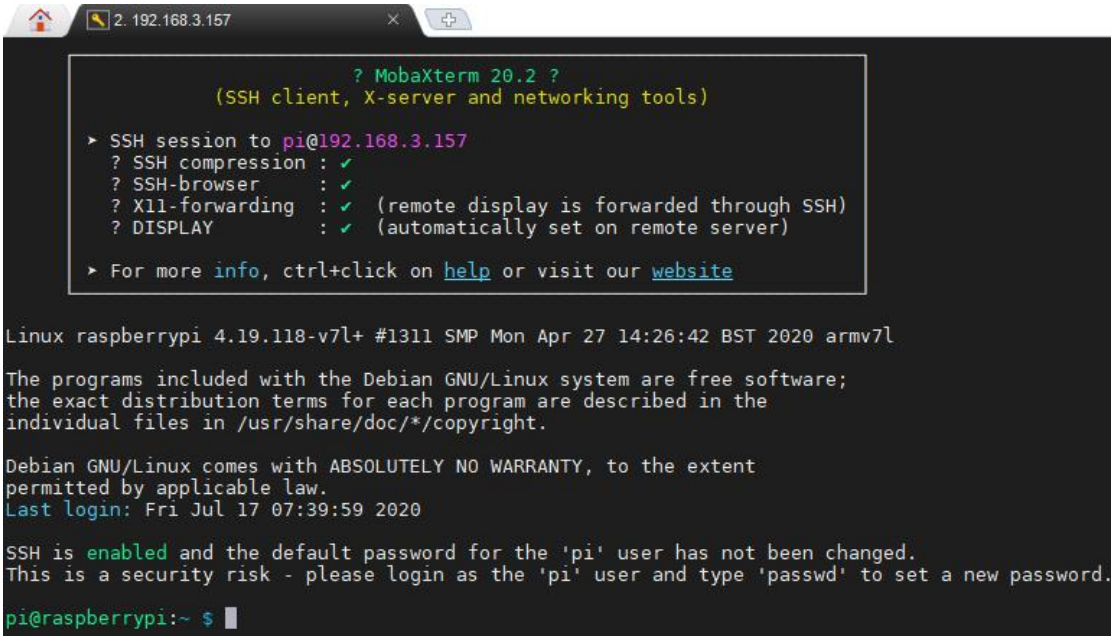


5.4 How to display text on the OLED screen

5.4.1Run the program of this course

1. Open the terminal software MobaXterm:

2. Log in to your Raspberry Pi (the way to log in to Raspberry Pi has been introduced in Lesson 1):

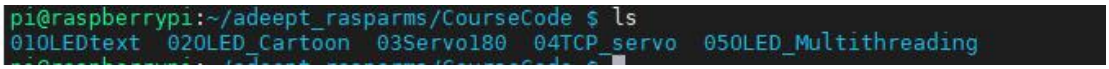


3. The relevant code programs of the RaspArm-S robot are stored in the adeept_rasparms folder, which has been explained in "2.1 Downloading the code program for controlling the robot" in Lesson 2. First, you need to enter a command with the command window of the Raspberry Pi to enter the folder where the course code is stored: CourseCode, this folder stores the sample code program for each course, enter the following command:

cd adeept_rasparms/CourseCode

4. Enter the command to display the contents of the current directory:

ls



5. The 01OLEDtext folder contains the sample code of this lesson. Enter the command to enter this folder:

cd 01OLEDtext

6. Enter the command to display the contents of the current directory:

ls

```
pi@raspberrypi:~/adeept_rasparms/CourseCode/01OLEDtext $ ls
OLEDtext.py
```

7. OLEDtext.py is a python program. When using the OLED Screen module, we need to install the Python dependent library needed to control the OLED screen, called luma.oled library, and enter the following commands in the console of the command window:

sudo pip3 install luma.oled

```
pi@raspberrypi:~/rasparms/CourseCode/01OLEDtext $ sudo pip3 install luma.oled
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting luma.oled
```

If a download timeout warning appears, you can download the luma.oled library by adding a image source, and enter the following command:

sudo pip3 install luma.oled -i https://pypi.tuna.tsinghua.edu.cn/simple/

```
pi@raspberrypi:~/rasparms/CourseCode/01OLEDtext $ sudo pip3 install luma.oled -i https://pypi.tuna.tsinghua.edu.cn/simple/
Looking in indexes: https://pypi.tuna.tsinghua.edu.cn/simple/, https://www.piwheels.org/simple
Collecting luma.oled
  Downloading https://pypi.tuna.tsinghua.edu.cn/packages/35/94/b0febec7ddf50f3a3bea88ca6ee8e0bb7e2695a8086b2cd9fccc92e1e1fd
ed-3.5.0-py2.py3-none-any.whl (18 kB)
Collecting luma.core>=1.14.0
  Downloading https://pypi.tuna.tsinghua.edu.cn/packages/b7/85/245afbecadc566a696b4e8f863755a30564390ca302c7c516396b160296f
re-1.15.0-py2.py3-none-any.whl (55 kB)
     |████████████████████████████████| 55 kB 114 kB/s
Requirement already satisfied: pillow>=4.0.0 in /usr/lib/python3/dist-packages (from luma.core>=1.14.0->luma.oled) (5.4.1)
Requirement already satisfied: spidev<=3.4; platform_system == "Linux" in /usr/lib/python3/dist-packages (from luma.core>=1
uma.oled) (3.4)
Requirement already satisfied: pyftdi in /usr/local/lib/python3.7/dist-packages (from luma.core>=1.14.0->luma.oled) (0.51.2
Requirement already satisfied: smbus2 in /usr/local/lib/python3.7/dist-packages (from luma.core>=1.14.0->luma.oled) (0.3.0)
Collecting RPI.GPIO; platform_system == "Linux"
  Downloading https://www.piwheels.org/simple/rpi-gpio/RPi.GPIO-0.7.0-cp37-cp37m-linux_armv7l.whl (69 kB)
     |████████████████████████████████| 69 kB 8.0 kB/s
Requirement already satisfied: pyserial>=3.0 in /usr/lib/python3/dist-packages (from pyftdi->luma.core>=1.14.0->luma.oled)
Requirement already satisfied: pyusb>=1.0.0 in /usr/local/lib/python3.7/dist-packages (from pyftdi->luma.core>=1.14.0->luma
1.0.2)
Installing collected packages: RPI.GPIO, luma.core, luma.oled
Successfully installed RPI.GPIO-0.7.0 luma.core-1.15.0 luma.oled-3.5.0
```

8. OLEDtext.py is a python program.You can run this program on the Raspberry Pi by directly typing the following commands:

sudo python3 OLEDtext.py

```
pi@raspberrypi:~/adeept_rasparms/CourseCode/01OLEDtext $ sudo  python3  OLEDtext.py
```

9. After successfully running the program, you will observe that "ROBOT" will be displayed on the OLED screen.

10. If you want to terminate the running program, you can press the shortcut key Ctrl+C on the keyboard.

4.4.2 The main code program of this lesson

After the above hands-on practice, you already know how to use and run our course sample code program. You must be curious about how our code program is programmed to display text on OLED screen on the Raspberry Pi. Let's get to know the main code program.Here we use Subline IDE to view and edit the code program of this lesson. For the specific method, please see the content of Lesson 2: "2.4 Editing the code program in Raspberry Pi". In the file manager of the MobaXterm terminal, find adeept_rasparms/CourseCode /01OLEDtext, open the code of this lesson: OLEDtext.py.

1. First import the library used to control the OLED screen.

```
from luma.core.interface.serial import i2c
from luma.core.render import canvas
from luma.oled.device import ssd1306
```

2. Then import the library used to control the delay time.

```
import time
```

3. Instantiate the object used to control the OLED.

```
serial = i2c(port=1, address=0x3C)
device = ssd1306(serial, rotate=0)
```

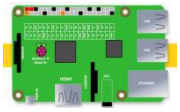
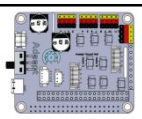
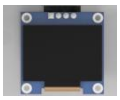
4. Display the text "ROBOT" at the position (0, 20) on the OLED screen.

```
with canvas(device) as draw:
    draw.text((0, 20), "ROBOT", fill="white")

while True:
    time.sleep(10)
```

Lesson 6 Playing Animation on the OLED Screen

In this lesson, we will learn how to play animation on the OLED screen..

6.1 Components used in this course

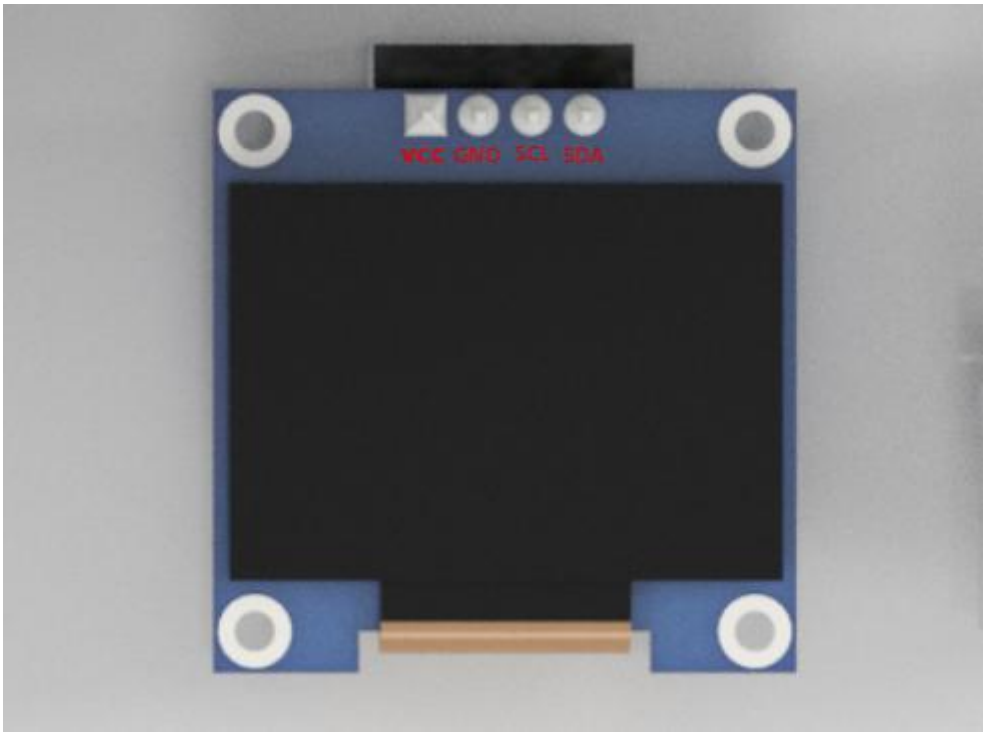
Components	Quantity	Picture
Raspberry Pi	1	
Robot HAT	1	
OLED screen	1	
4 pin wire	1	

6.2 Introduction of OLED Screen

OLED (Organic Light-Emitting Diode), also known as organic electric laser display, organic light emitting semiconductor (Organic Electroluminesence Display, OLED). OLED is a kind of current-type organic light-emitting device, which produces light by the injection and recombination of carriers, and the luminous intensity is proportional to the injected current. The RaspArm-S robot uses an OLED screen to display the expressions or some parameters of the robot. OLED Screen is a commonly used module on robot products. Due to the black non-luminous feature of OLED Screen, this type of screen has extremely high contrast. Even if the ambient light is strong, you can see the information on the OLED Screen clearly, and the power consumption is relatively low. We only need to connect the power supply and the GPIO port to control it.

If you want to use OLED Screen, you need to use a 4-pin cable with anti-reverse

interface to connect the OLED screen to the IIC interface on the Robot HAT.



If you do not use Robot HAT driver board to connect with Raspberry Pi driver board, then you need to connect Vin of OLED screen to 5V or 3.3V of Raspberry Pi, and connect GND of OLED screen to GND of Raspberry Pi. Connect SCL of Robot HAT to SCL of OLED, and SCA of Robot HAT to SCA of Raspberry Pi. Please refer to the pin definition diagram of Raspberry Pi for specific pins.

6.3 How to play animation on OLED screen

6.3.1 How to play animation on OLED screen

1. First use Python to generate a PPM sequence string.
2. Then play the PPM sequence string.

6.3.2 PPM format file generation

1. The OLED screen can display images, and the image format needs to be PPM format.

2. The principle of animation is to play multiple images per second.

3. First, draw a few pictures in jpg format (the ones you want to display) in the PC and put them in the same folder (jpg). Note that these pictures need a pure black background.

4. Then create a new folder named ppm.

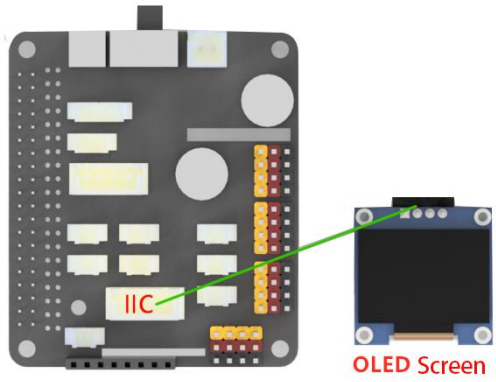
5. Finally, create a new ppmGenout.py outside the folder, this program is responsible for generating PPM sequence strings.

6. It is also necessary to create another oledPlay.py, which is responsible for playing the PPM sequence string that has been generated.

7. This program can be executed on Raspberry Pi or other computers, and Python3 and pillow library need to be installed.

6.4 Wiring diagram (Circuit diagram)

When the OLED Screen module is in use, it needs to be connected to the IIC interface on the Robot HAT driver board, as shown below:

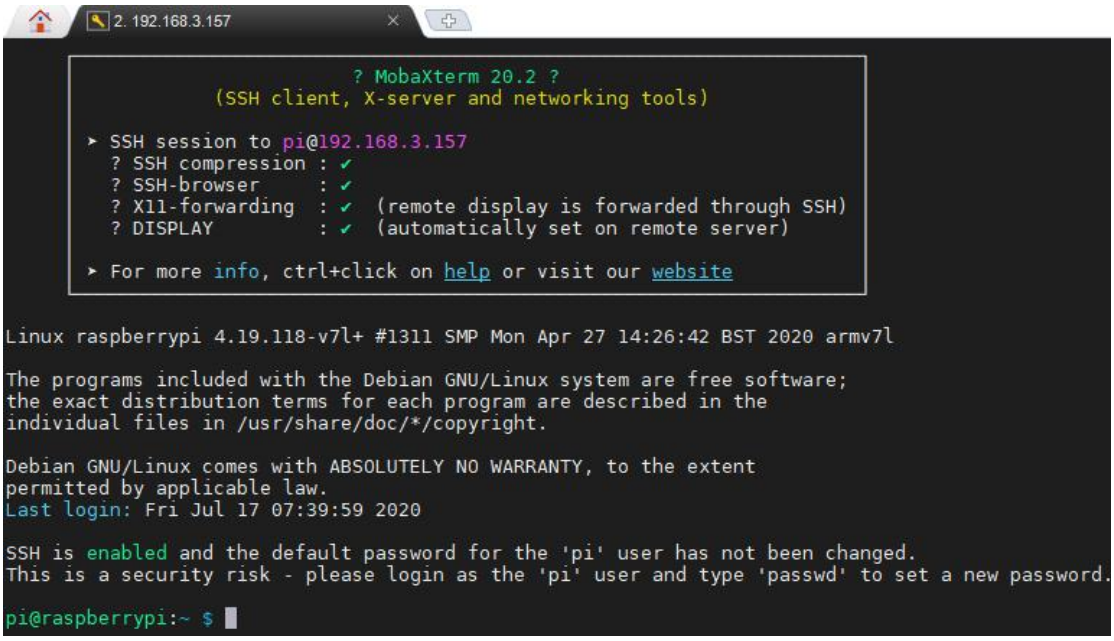


6.5 How to play animation on OLED screen

6.5.1 Run the program of this course

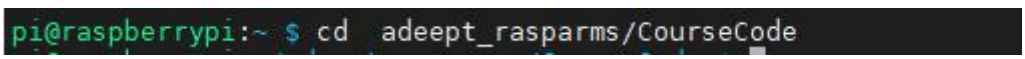
1. Open the terminal software MobaXterm:

2. Log in to your Raspberry Pi (the way to log in to Raspberry Pi has been introduced in Lesson 1):



3. The relevant code programs of the RaspArm-S robot are stored in the adeept_rasparms folder, which has been explained in "2.1 Downloading the code program for controlling the robot" in Lesson 2. First, you need to enter a command with the command window of the Raspberry Pi to enter the folder where the course code is stored: CourseCode, this folder stores the sample code program for each course, enter the following command:

cd adeept_rasparms/CourseCode



4. Enter the command to display the contents of the current directory:

ls

```
pi@raspberrypi:~/adeept_rasparms/CourseCode $ ls
01OLEDtext  02OLED_Cartoon  03Servo180  04TCP_servo  05OLED_Multithreading
```

5. The 02OLED_Cartoon folder contains the sample code of this lesson. Enter the command to enter this folder:

cd 02OLED_Cartoon

```
pi@raspberrypi:~/adeept_rasparms/CourseCode $ cd 02OLED_Cartoon
```

6. Enter the command to display the contents of the current directory:

ls

```
pi@raspberrypi:~/adeept_rasparms/CourseCode/02OLED_Cartoon $ ls
jpg  oledPlay.py  ppm  ppmGenout.py
```

7. It can be found that there are four files.These are the reference cases we provide.If you want to realize your own creativity, you can also imitate this method. The "jpg" folder is to store pictures in jpg format that have been drawn (if you want to play other pictures, then you can put the pictures in this folder). Note that these pictures need a pure black background; "ppm" is a folder that needs to be established, named ppm, and the inside is empty; "ppmGenout.py" is a python program that generates PPM sequence strings from pictures in jpg format. After running it, a sequence string in ppm format will be generated in the "ppm" folder. "OledPlay.py" is a code program for playing PPM sequence strings.

6.5.1.1 First generate PPM sequence string

1. Before running the program, you need to install the pillow library. Enter the following command:

pip install pillow

```
pi@raspberrypi:~/rasparms/CourseCode/02OLED_Cartoon $ pip install pillow
Defaulting to user installation because normal site-packages is not writeable
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
```

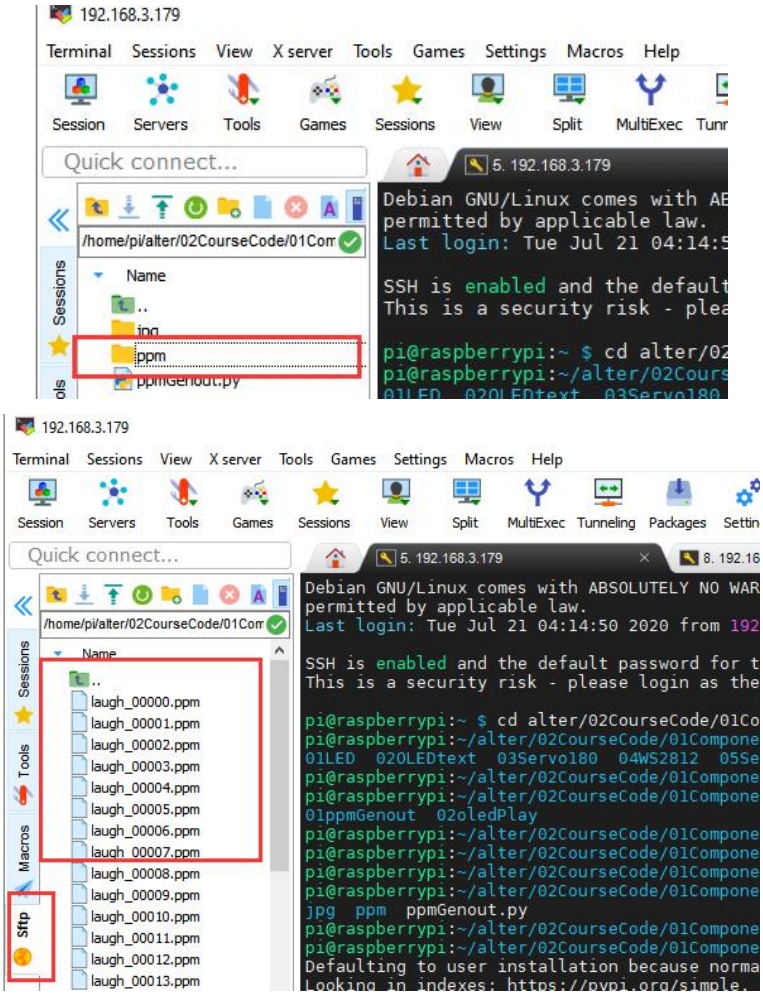
2. First, you need to run ppmGenout.py to generate the PPM sequence string. Enter the following command to run the ppmGenout.py program on the Raspberry Pi:

sudo python3 ppmGenout.py

```
pi@raspberrypi:~/adeept_rasparms/CourseCode/02OLED_Cartoon $ sudo  python3 ppmGenout.py
```

3. Then in the file explorer on the left side of MobaXterm, open the folder "ppm"

of this lesson in the adeept_rasparms/CourseCode/02OLED_Cartoon directory. This folder was empty before. After you successfully run the ppmGenout.py program After that, many ppm sequence strings will be generated inside, as shown below:



6.5.1.2 Play the generated PPM sequence string

1. When using the OLED Screen module, we need to install the Python dependency library required to control the OLED screen, called the luma.oled library, and enter the following commands in the console of the command window:

sudo pip3 install luma.oled

```
pi@raspberrypi:~/rasparms/CourseCode/01OLEDtext $ sudo pip3 install luma.oled
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting luma.oled
```

2. Then continue to download the driver library of the OLED Screen module: Adafruit_SSD1306, enter the following command:

sudo pip3 install Adafruit_SSD1306

```
pi@raspberrypi:~/rasparms/CourseCode/02OLED_Cartoon $ sudo pip3 install Adafruit_SSD1306
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting Adafruit_SSD1306
  Downloading https://www.piwheels.org/simple/adafruit-ssd1306/Adafruit_SSD1306-1.6.2-py3
Requirement already satisfied: Adafruit-GPIO>=0.6.5 in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: spidev in /usr/lib/python3/dist-packages (from Adafruit-GP
Requirement already satisfied: adafruit-pureio in /usr/local/lib/python3.7/dist-packages
D1306) (1.1.5)
Installing collected packages: Adafruit-SSD1306
Successfully installed Adafruit-SSD1306-1.6.2
```

3. After successfully generating the PPM sequence string, you need to run the oledPlay.py program again, so that the animation will be displayed on the OLED screen:

sudo python3 oledPlay.py

```
pi@raspberrypi:~/rasparms/CourseCode/02OLED_Cartoon $ sudo  python3  oledPlay.py
```

4. After running the program successfully, you will observe that an animation will be played on the OLED screen.

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

6.5.2 The main code program of this lesson

After the above hands-on practice, you already know how to use and run our course sample code program. You must be curious about how our code program is programmed to play animation on OLED screen on the Raspberry Pi. Let’s get to know the main code program.We use Sublime IDE to view and edit the code program of this lesson. For details, please refer to the content of Lesson 2: "2.4 Editing the Code Program in Raspberry Pi". In the file manager of the MobaXterm terminal, find adeept_rasparms/CourseCode/02OLED_Cartoon, and open the code of this lesson: ppmGenout.py and oledPlay

6.5.2.1.Let's first learn the program ppmGenout.py that generates PPM sequence strings

First import the location of jpg and ppm images

```
jpgPath = 'jpg/'
ppmPath = 'ppm/'
```

Get the file names of all jpg images in the jpg folder.

```
jpgNames = os.listdir(jpgPath)
```

Convert the pictures in the "jpg" folder one by one. First open a jpg image.

```
for frameName in jpgNames:
    image = Image.open(jpgPath+frameName)
```

Delete the suffix .jpg from the original name, and re-modify it to the suffix .ppm, so that it becomes a file of ppm sequence.

```
    newName = frameName[:-4]+'.ppm'
    image.save((ppmPath+newName), 'ppm')
```

6.5.2.2. Learn to play the program of PPM sequence string oledPlay.py

Import OLED screen related libraries.

```
import time
import Adafruit_GPIO.SPI as SPI
import Adafruit_SSD1306
```

Import the library for image processing.

```
from PIL import Image
from PIL import ImageDraw
from PIL import ImageFont

import os
```

Obtain the absolute path of this file.

```
curpath = os.path.realpath(__file__)
thisPath = "/" + os.path.dirname(curpath) + "/"
```

Initialize the OLED screen.

```
RST = 24
disp = Adafruit_SSD1306.SSD1306_128_64(rst=RST)
disp.begin()
```

Set the playback frame rate.

```
FPS = 30
```

Calculate the delay time per frame according to the playback frame rate.

```
LaughImage = []
```

Clear the screen.

```
disp.clear()
disp.display()
```

All images will be stored in this array.

```
LaughImage = []
```

tell the program where you put the ppm sequence

```
ppmPath = 'ppm/'
```

Get the names of all frames in this folder.

```
ppmNames = os.listdir(ppmPath)
```

Sort these files by name.

```
ppmNames.sort()
```

Import these frames, then open the file in PPM format and binarize it with convert('1'); store the converted image into LaughImage.

```
for frameName in ppmNames:
    image = Image.open(thisPath+ppmPath+frameName).convert('1')
    LaughImage.append(image)
```

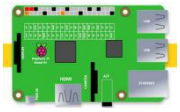
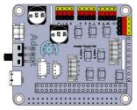
Displaying images frame by frame is equivalent to playing a cartoon.

```
for i in range(0, len(LaughImage)):
    disp.image(LaughImage[i])
    disp.display()
    time.sleep(timeDelay)
```

Lesson 7 How to Control 180° Servo

In this lesson, we will learn how to control 180° Servo.

7.1 Components used in this course

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

7.2 Introduction of 180° Servo

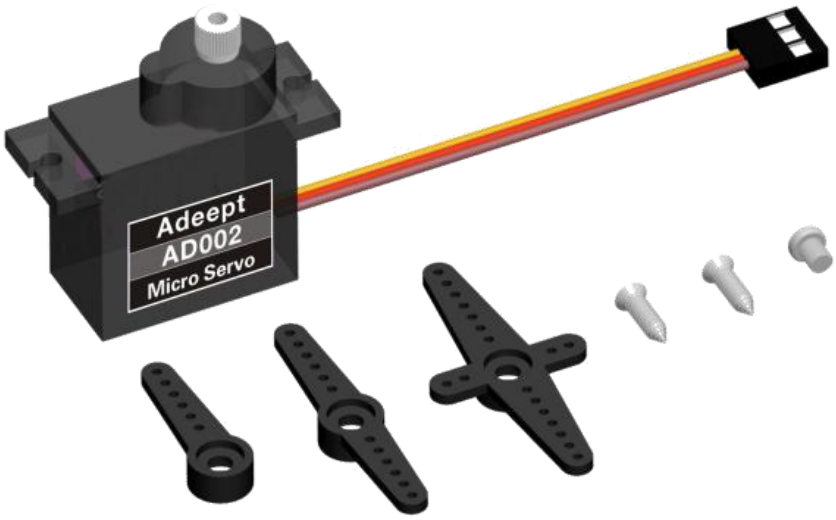
What is a servo?

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°. It is used in the RaspArm-S robot products. Since the 180° servo can use the PWM signal to control the rotation angle of a certain mechanism, it is a more commonly used module in robot products. The walking robot, robotic arm and pan/tilt are all driven by it.

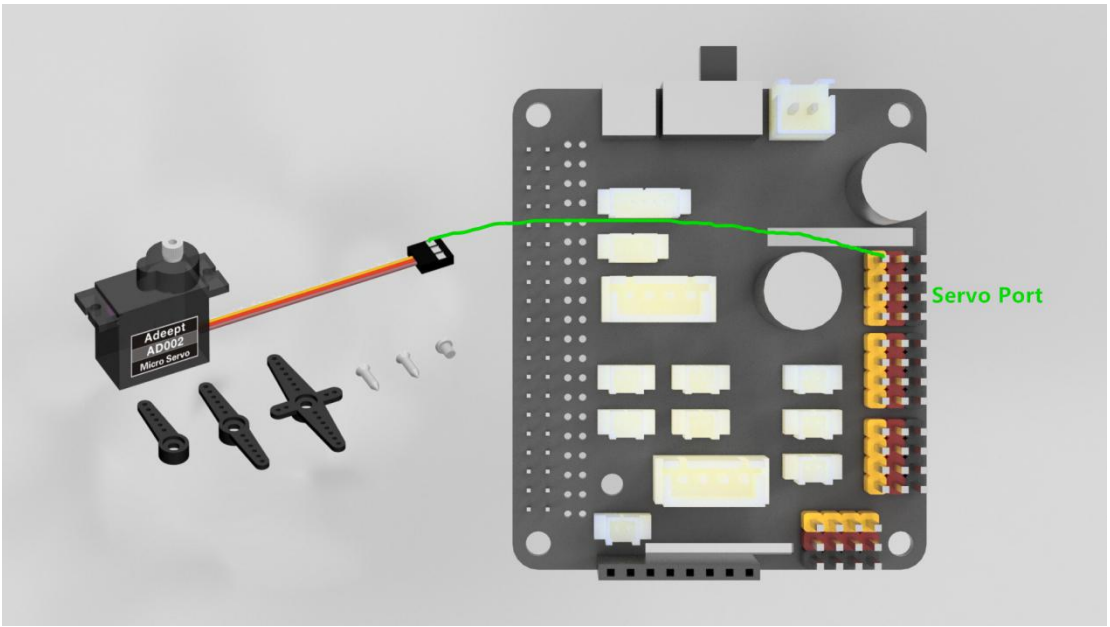
On the Raspberry Pi driver board Robot HAT, there is a PCA9685 chip specially used to control the servo. The Raspberry Pi uses I2C to communicate with the PCA9685. It controls the servo by sending pulse signals from the microcontroller. These pulses tell the servo mechanism of the servo where to move. The picture of the

180° servo is as follows:



7.3 Wiring diagram (Circuit diagram)

When the 180°Servo module is in use, it needs to be connected to the servo interface on the RobotHAT 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 black pin,as shown below:

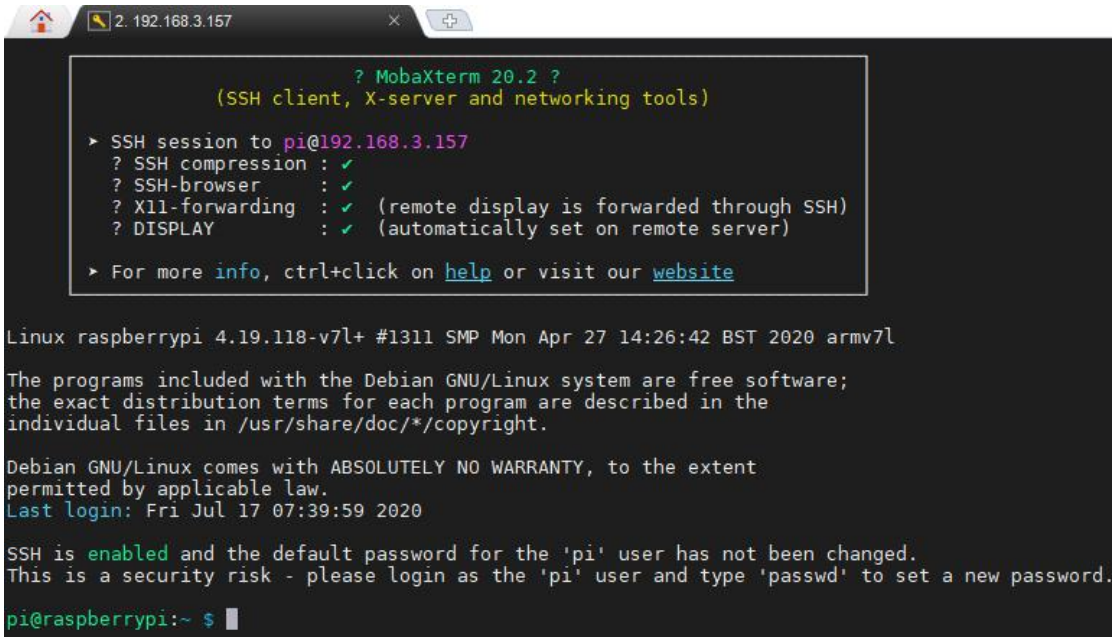


7.4 How to control 180°Servo

7.4.1 Run the program of this course

1. Open the terminal software MobaXterm:

2. Log in to your Raspberry Pi (the way to log in to Raspberry Pi has been introduced in Lesson 1):



3. The relevant code programs of the RaspArm-S robot are stored in the adeept_rasparms folder, which has been explained in "2.1 Downloading the code program for controlling the robot" in Lesson 2. First, you need to enter a command with the command window of the Raspberry Pi to enter the folder where the course code is stored: CourseCode, this folder stores the sample code program for each course, enter the following command:

cd adeept_rasparms/CourseCode

```
pi@raspberrypi:~ $ cd  adeept_rasparms/CourseCode
```

4. Enter the command to display the contents of the current directory:

ls

```
pi@raspberrypi:~/adeept_rasparms/CourseCode $ ls
01OLEDtext  02OLED_Cartoon  03Servo180  04TCP_servo  05OLED_Multithreading
```

5.The 03Servo180 folder stores the sample code of this course. Enter the command to enter this folder:

cd 03Servo180

```
pi@raspberrypi:~/adeept_rasparms/CourseCode $ cd 03Servo180
```

6. Enter the command to display the contents of the current directory:

ls

```
pi@raspberrypi:~/adeept_rasparms/CourseCode/03Servo180 $ ls
180Servo_01.py  180Servo_02.py
```

7. 180Servo_01.py is to control the servo to reciprocate; 180Servo_02.py is to control the servo to reciprocate slowly. When using the 180°Servo module, we need to install the Python dependency library which is needed to control the 180°Servo: Adafruit_PCA9685, and enter the following command in the console of the command window:

sudo pip3 install adafruit-pca9685

```
pi@raspberrypi:~/rasparms/CourseCode/03Servo180 $ sudo pip3 install adafruit-pca9685
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
```

8.180Servo_01.py is a python program. You can run this program on the Raspberry Pi by directly typing the following commands:

sudo python3 180Servo_01.py

```
pi@raspberrypi:~/adeept_rasparms/CourseCode/03Servo180 $ sudo  python3  180Servo_01.py
```

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

10. If you want to terminate the running program, you can press the shortcut key Ctrl+C on the keyboard.

7.4.2 The main code program of this lesson

After the above hands-on practice, you already know how to use and run our course sample code program. Then you must be curious to know how our code program is programmed on the Raspberry Pi to control the 180° servo. Let's learn about the main code program together.Here we use Subline IDE to view and edit the code program of this lesson. For the specific method, please see the content of Lesson 2: "2.4 Editing the code program in the Raspberry Pi". In the file manager of the MobaXterm terminal, find adeept_rasparms/CourseCode /03Servo180, open the code for this lesson: 180Servo_01.py and 180Servo_02.py.

```
1  import Adafruit_PCA9685
2  import time
3
4  pwm = Adafruit_PCA9685.PCA9685()
5  pwm.set_pwm_freq(50)
6
7  while 1:
8      pwm.set_pwm(3, 0, 300)
9      time.sleep(1)
10     pwm.set_pwm(3, 0, 400)
11     time.sleep(1)
```

The code in the figure above is for the program 180Servo_01.py.In the above code, set_pwm_freq(50) is used to set the frequency of PWM 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 other servos, we need to refer to the specific servo documentation to set this value.

pwm.set_pwm(3, 0, 300) is used to control the rotation of a servo to a certain position. 3 is the port number of the servo, which corresponds to the number marked on the RobotHAT drive board.Pay attention to the occasion that do not insert the ground wire, VCC and signal wire in the reverse direction when connecting the servo with the driver board. Brown to black, red to red, and yellow to yellow; 0 is the deviation value for controlling the rotation of the servo, but our program does not use this function to correct the deviation (the reason for the error of the servo can be referred to 4.2 precautions for structural assembly); 300 is the PWM duty cycle value to be set. Depending on the servo, the servo angle represented by this value is also

different. 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°.

The above code cannot control the rotation speed of the servo. If we want a servo to slowly reciprocate between two positions, we need to use the method of increasing or decreasing the variable to control it.

```
import Adafruit_PCA9685
import time

pwm = Adafruit_PCA9685.PCA9685()
pwm.set_pwm_freq(50)

while 1:
    for i in range(0,100):
        pwm.set_pwm(3, 0, (300+i))
        time.sleep(0.05)
    for i in range(0,100):
        pwm.set_pwm(3, 0, (400-i))
        time.sleep(0.05)
```

Using the above code can make the servo rotate slowly back and forth between 300 and 400, but this method of controlling the servo also has great drawbacks. When the program is executed to the slow motion part of the servo, it will be blocked, which will seriously affect the program. Therefore, a multi-threaded solution is provided in our robot product program to solve this problem.

Lesson 8 Simple TCP Communication

In this lesson, we use remote control of the servo as a TCP communication example. This example helps beginners understand how the desktop GUI program communicates with the Raspberry Pi.

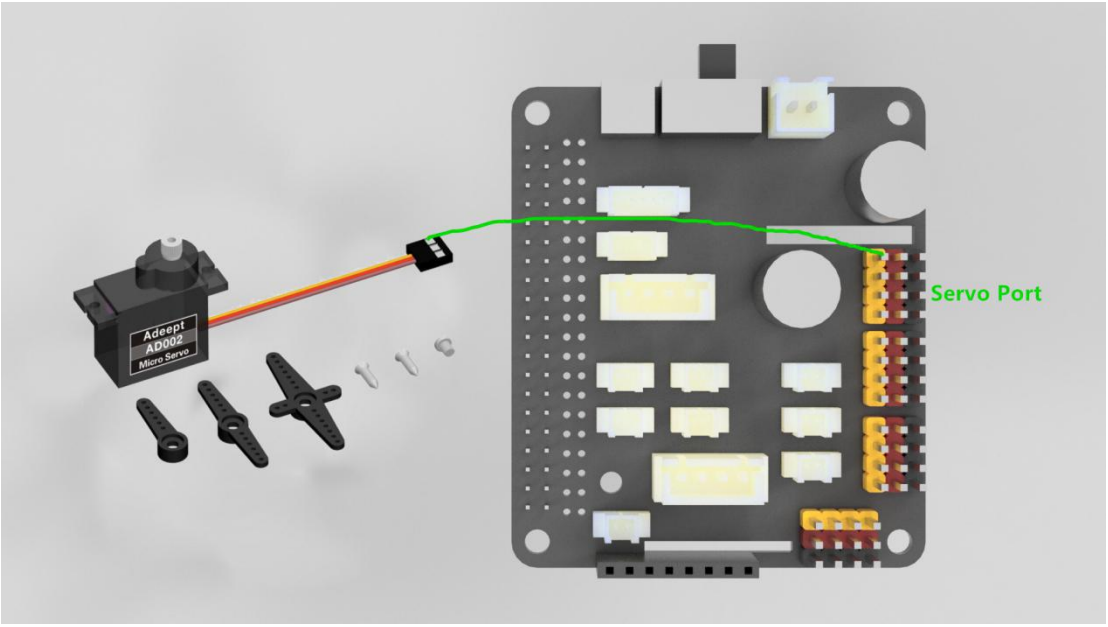
You can use the program with a graphical interface written by yourself to communicate with the Raspberry Pi on other devices to achieve the purpose of controlling the Raspberry Pi.

8.1 About Tkinter and Socket

The GUI programming method introduced in this chapter is completely done by Python language, specifically, the Tkinter library is used.Tkinter is Python's standard GUI library. Python uses Tkinter to quickly create GUI applications. Because Tkinter is built into the Python installation package, as long as Python is installed, you can import the Tkinter library, and IDLE is also written in Tkinter. For simple graphical interface Tkinter can still cope with it.

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

8.2 Wiring diagram (Circuit diagram)



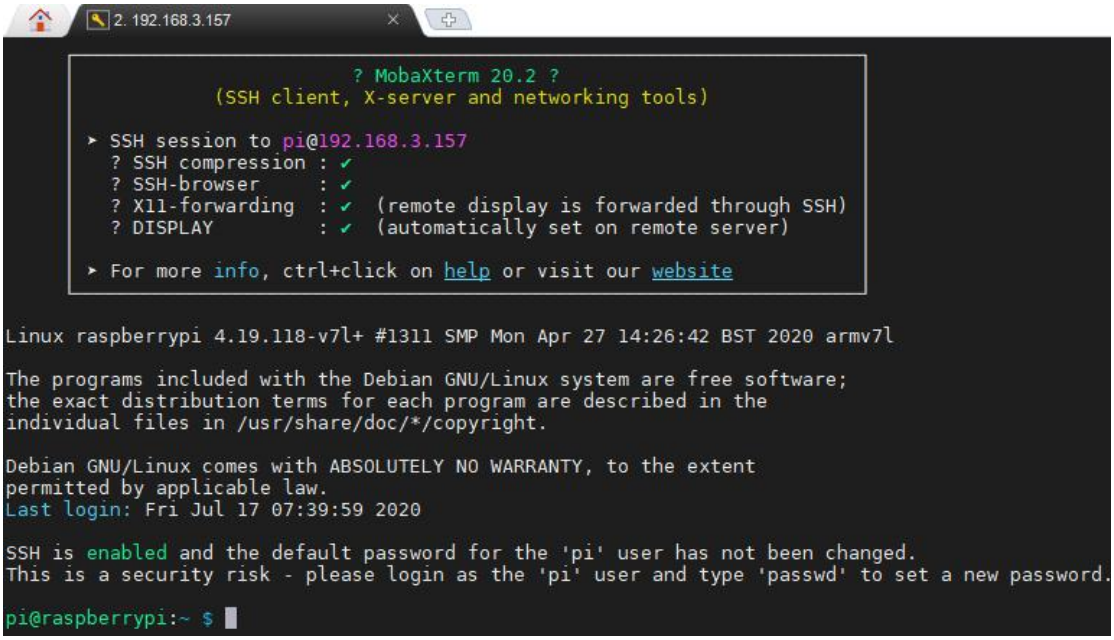
8.3 How to remotely control Servo

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

8.3.1Run the tcpPI.py program in the Raspberry Pi first

1. Open the terminal software MobaXterm:

2. Log in to your Raspberry Pi (the way to log in to Raspberry Pi has been introduced in Lesson 1):

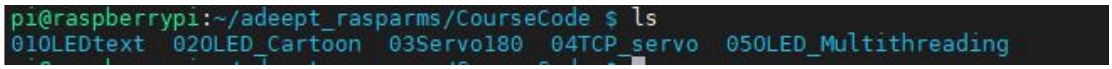


3. The relevant code programs of the RaspArm-S robot are stored in the adeept_rasparms folder, which has been explained in "2.1 Downloading the code program for controlling the robot" in Lesson 2. First, you need to enter a command with the command window of the Raspberry Pi to enter the folder where the course code is stored: CourseCode, this folder stores the sample code program for each course, enter the following command:

cd adeept_rasparms/CourseCode

4. Enter the command to display the contents of the current directory:

ls



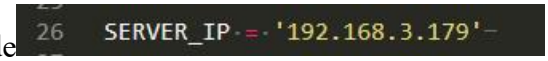
5. The 04TCP_servo folder contains the sample code of this lesson. Enter the command to enter this folder:

cd 04TCP_servo

6. Enter the command to display the contents of the current directory:

ls

```
pi@raspberrypi:~/adeept_rasparms/CourseCode/04TCP_servo $ ls
tcpPC.py  tcpPI.py
```

7. tcpPC.py needs to be run on the PC, you need to modify the IP inside 

to the IP of your Raspberry Pi, so that the connection can be established; tcpPI.py runs on the Raspberry Pi.When using the 180°Servo module, we need to install the Python dependency library which is needed to control the 180°Servo: Adafruit_PCA9685, and enter the following command in the console of the command window:

sudo pip3 install adafruit-pca9685

```
pi@raspberrypi:~/rasparms/CourseCode/04TCP_servo $ sudo pip3 install adafruit-pca9685
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
```

8. Then run the tcpPI.py program on the Raspberry Pi, and directly enter the following commands to run this program on the Raspberry Pi:

sudo python3 tcpPI.py

```
pi@raspberrypi:~/alter/02CourseCode/01ComponentCode/12TCP_LED $ sudo python3 tcpPI.py
```

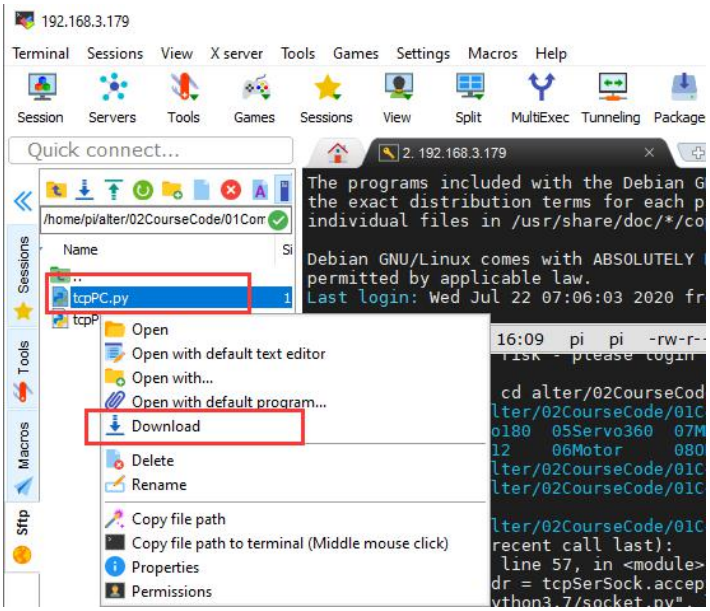
9. After successfully running the program, you also need to run the tcpPC.py program on the PC. You need to change the IP inside to the IP of your Raspberry Pi, so that the connection can be established.

8.3.2Run the tcpPC.py program on the PC

1. In the directory:

adeept_asparms/CourseCode/04TCP_servo

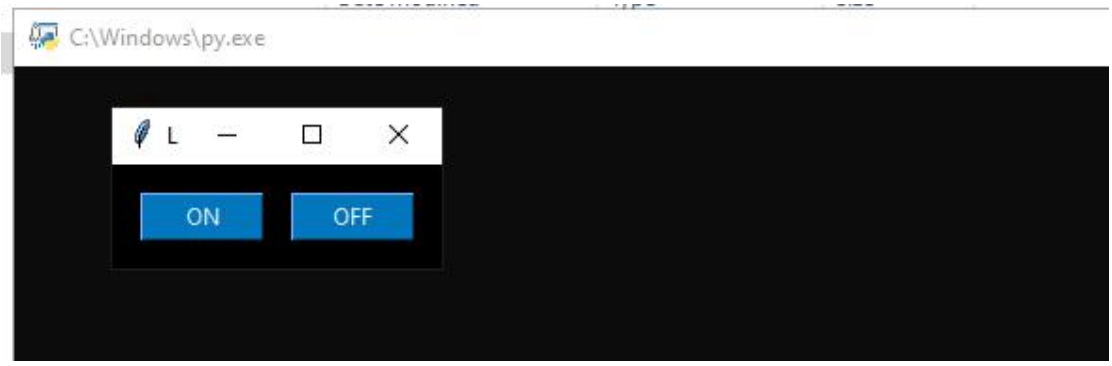
Download the file tcpPC.py to the PC and remember the path of this file.



2. Use MobaXterm or other IDE to open the tcpPC.py file on your computer. You need to modify the IP address in the SERVER_IP line of code and change it to the IP of your Raspberry Pi (please check the IP address of the Raspberry Pi in lesson one), so that the PC can establish a connection with the Raspberry Pi.

```
25
26  SERVER_IP = '192.168.3.179'
27
```

3. After the modification is completed, you can directly double-click to open the tcpPC.py file. After opening, as shown below, there are two buttons "ON" and "OFF". When you click the "ON" button, the servo will rotate; when you click the "OFF" button, the servo will stop rotating.



8.4 Studying the program code of tcpPI.py and tcpPC.py

8.4.1 Learning tcpPI.py program code

Import the socket library which is used for TCP communication.

```
10  import socket
11
```

Define a flag fun_flag to control the servo; then use multi-thread to control the servo.

```
17  #定义标志位
18  fun_flag = 0
19  """
20  定义多线程的类
21  """
22  class Functions(threading.Thread):
23      def __init__(self, *args, **kwargs):
24          super(Functions, self).__init__(*args, **kwargs)
25          self.__flag = threading.Event()
26          self.__flag.clear()
```

Next is the configuration related to TCP communication. PORT is the defined port number. You can freely choose numbers from 0-65535. It is recommended to choose the numbers after 1023, which needs to be consistent with the port number defined by the client in the PC.

```
44  HOST = ''
45  PORT = 10223
46  BUFSIZ = 1024
47  ADDR = (HOST, PORT)
48
49  tcpSerSock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
50  tcpSerSock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR,1)
51  tcpSerSock.bind(ADDR)
52  tcpSerSock.listen(5)
53
```

Start monitoring the client connection, and start receiving the information sent from the client after the client connection is successful.

```
57  tcpCliSock, addr = tcpSerSock.accept()
58
```

Receive the information from the client, if the information content is on, the servo rotates, if the information content is off, the servo stops rotating.

```
93      elif 'on' == data:
94          fun_flag = 1
95          threadX.resume()
96      
97      
98      elif 'off' == data:
99          fun_flag = 0
00          time.sleep(1)
01          threadX.pause()
```

Finally, print out the received data, and continue to monitor the next message sent by the client.

```
86      print(data)
```

8.4.2 Learning tcpPC.py program code

Import the socket library used for TCP communication.

```
4   from socket import *
```

Python uses Tkinter to quickly create GUI applications and instantiate them while importing them.

```
9   import tkinter as tk
```

Call this method to send the command'on' to control the rotation of the servo.

```
15      tcpClicSock.send(('on').encode())
```

Call this method to send the command'off' to stop the servo.

```
21      tcpClicSock.send(('off').encode())
```

Enter the IP address of the Raspberry Pi here (you need to change it to your Raspberry Pi IP address when you use it).

```
26  SERVER_IP = '192.168.3.179'
27
```

Next is the configuration related to TCP communication. PORT is the defined port number. You can freely choose numbers from 0-65535. It is recommended to choose a 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)
```

Define a GUI window.

```
root = tk.Tk()
```

Set the title of the window.

```
root.title('Lights')
```

The size of the window, the middle x is the English letter x.

```
root.geometry('175x55')
```

Define the background color of the window.

```
root.config(bg='#000000')
```

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, it will 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, start the message loop.

```
root.mainloop()
```

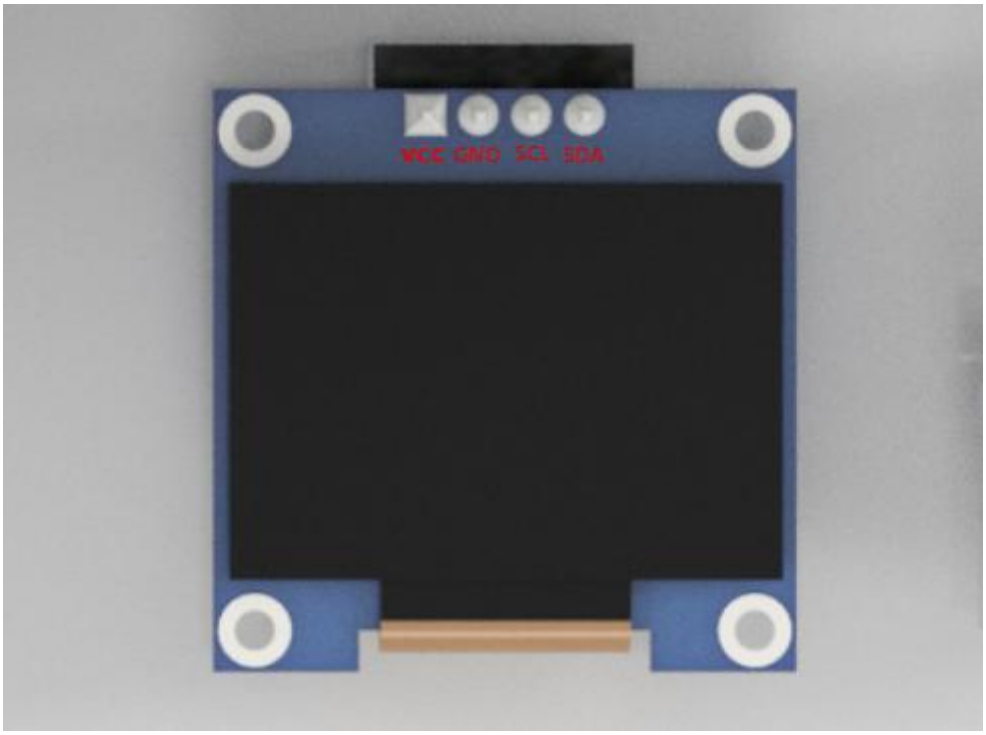
Lesson 9 Multithreading OLED Screen Control

In this lesson, we will use multithreading to display text and other information on the OLED screen.

9.1 Introduction of OLED screen

OLED (Organic Light-Emitting Diode), also known as organic electric laser display, organic light emitting semiconductor (Organic Electroluminesence Display, OLED). OLED is a kind of current-type organic light-emitting device, which produces light by the injection and recombination of carriers, and the luminous intensity is proportional to the injected current. The Alter robot uses an OLED screen to display the expressions or some parameters of the robot. OLED Screen is a commonly used module on robot products. Due to the black non-luminous feature of OLED Screen, this type of screen has extremely high contrast. Even if the ambient light is strong, you can see the information on the OLED Screen clearly, and the power consumption is relatively low. We only need to connect the power supply and the GPIO port to control it.

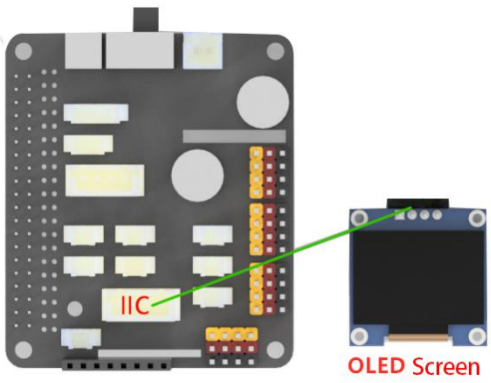
If you want to use OLED Screen, you need to use a 4-pin cable with anti-reverse interface to connect the OLED screen to the IIC interface on the Robot HAT.



If you do not use Robot HAT driver board to connect with Raspberry Pi driver board, then you need to connect Vin of OLED screen to 5V or 3.3V of Raspberry Pi, and connect GND of OLED screen to GND of Raspberry Pi. Connect SCL of Robot HAT to SCL of OLED, and SCA of Robot HAT to SCA of Raspberry Pi. Please refer to the pin definition diagram of Raspberry Pi for specific pins.

9.2 Preparation

You need to prepare a robot in the RaspArm-S that has been assembled, and make sure that the robot has been connected to the OLED Screen module.



9.3 Using multithreading to display text on OLED Screen

9.3.1 Learning the code program of this lesson

We use multi-threading to display text on the OLED Screen. Here, we use Subline IDE to view and edit the code program of this lesson. For the specific method, please see "2.4 Editing the Code Program in the Raspberry Pi" in Lesson 2. The specific code and comments are as follows:

In the file manager of the MobaXterm terminal, find adeept_rasparms/CourseCode/05OLED_Multithreading, and open the code of this lesson: OLED.py.

Import related dependent libraries.

```
from luma.core.interface.serial import i2c
from luma.core.render import canvas
from luma.oled.device import ssd1306, ssd1325, ssd1331, sh1106
import time
import threading
```

Set to display the initial content of each line on the OLED.

```
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'
```

ID used to pause the thread.

```
        self.__flag = threading.Event()
```

Set to True.

```
        self.__flag.set()
```

ID used to stop the thread.

```
        self.__running = threading.Event()
```

Set running to True.

```
37          self.__running.set()
```

Return immediately when it is True, block when it is False, and return when the internal flag is True.

```
41          self.__flag.wait()
```

Set to False to block the thread.

```
55      def pause(self):
56          self.__flag.clear()
```

Set to True to stop blocking with thread.

```
59          self.__flag.set()
```

Resume the thread from the suspended state, if it has been suspended.

```
60      def stop(self):
61          self.__flag.set()
```

Set to False.

```
62          self.__running.clear()
```

Call this function to control the OLED screen. Position is the number of the line where you want to change the content, can be 1-6, text is the content.

```
64      def screen_show(self, position, text):
65          global text_1, text_2, text_3, text_4, text_5, text_6
66          if position == 1:
67              text_1 = text
68          elif position == 2:
69              text_2 = text
70          elif position == 3:
71              text_3 = text
72          elif position == 4:
73              text_4 = text
74          elif position == 5:
75              text_5 = text
76          elif position == 6:
77              text_6 = text
78          self.resume()
```

Instantiate the OLED screen object.

```
84      screen = OLED_ctrl()
85
```

Start the thread.

```
89      screen.start()
```

Set the content of the first line to 12345678.

```
94      screen.screen_show(1, '123345678')
```

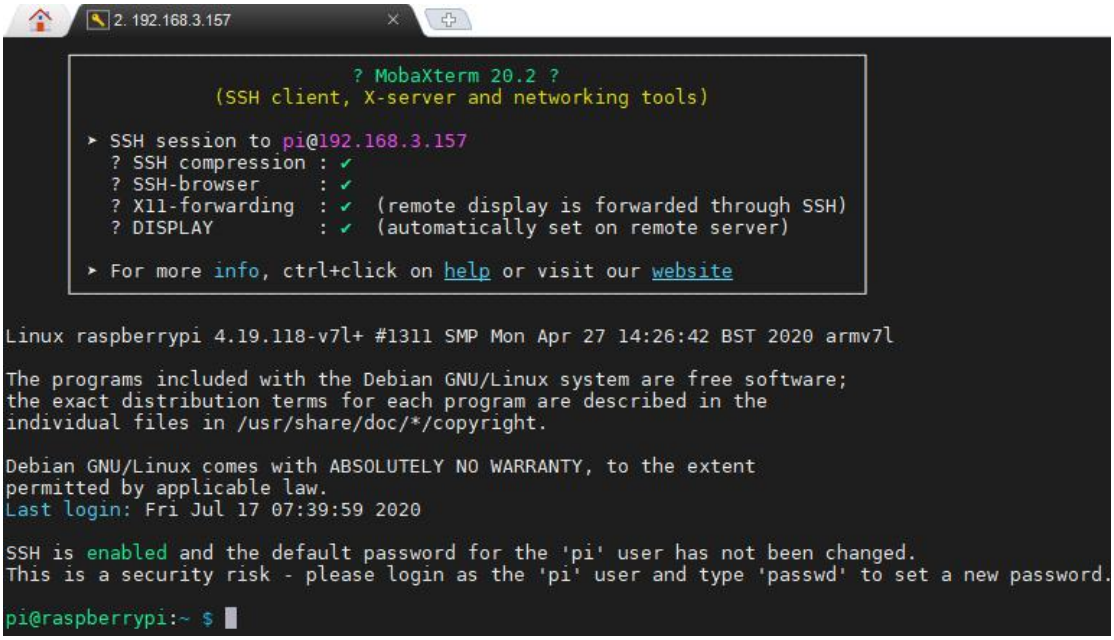
Since the above operation will not block the thread, we need to loop here to avoid the program exit after the execution is complete.

```
99      while 1:
100         time.sleep(10)
101         pass
```

9.3.2 Run the program of this course

1. Open the terminal software MobaXterm:

2. Log in to your Raspberry Pi (the way to log in to Raspberry Pi has been introduced in Lesson 1):

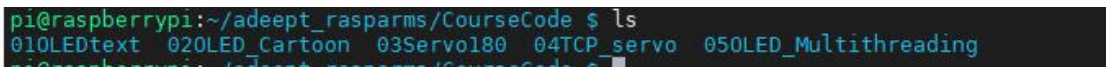


3. The relevant code programs of the RaspArm-S robot are stored in the adeept_rasparms folder, which has been explained in "2.1 Downloading the code program for controlling the robot" in Lesson 2. First, you need to enter a command with the command window of the Raspberry Pi to enter the folder where the course code is stored: CourseCode, this folder stores the sample code program for each course, enter the following command:

cd adeept_rasparms/CourseCode

4. Enter the command to display the contents of the current directory:

ls



5. The 05OLED_Multithreading folder contains the sample code of this lesson. Enter the command to enter this folder:

cd 05OLED_Multithreading

6. Enter the command to display the contents of the current directory:

ls

```
pi@raspberrypi:~/adeept_rasparms/CourseCode/050LED_Multithreading $ ls
OLED.py
```

7. When using the OLED Screen module, we need to install the Python dependent library needed to control the OLED screen, called luma.oled library, and enter the following commands in the console of the command window:

sudo pip3 install luma.oled

```
pi@raspberrypi:~/rasparms/CourseCode/050LED_Multithreading $ sudo pip3 install luma.oled
Looking in indexes: https://pypi.org/simple, https://www.piwheels.org/simple
Collecting luma.oled
  Downloading https://files.pythonhosted.org/packages/35/94/b0febec7ddf50f3a3bea88ca6ee8ed9fccc92e1e1fd/luma.oled-3.5.0-py2.py3-none-any.whl
Collecting luma.core>=1.14.0 (from luma.oled)
  Downloading https://files.pythonhosted.org/packages/b7/85/245afbecadc566a696b4e8f863755516396b160296f/luma.core-1.15.0-py2.py3-none-any.whl (55kB)
    100% |                                | 61kB 3.1kB/s
Requirement already satisfied: pillow>=4.0.0 in /usr/lib/python3/dist-packages (from luma
```

8. OLED.py is the sample code for this lesson, enter the command to run this program:

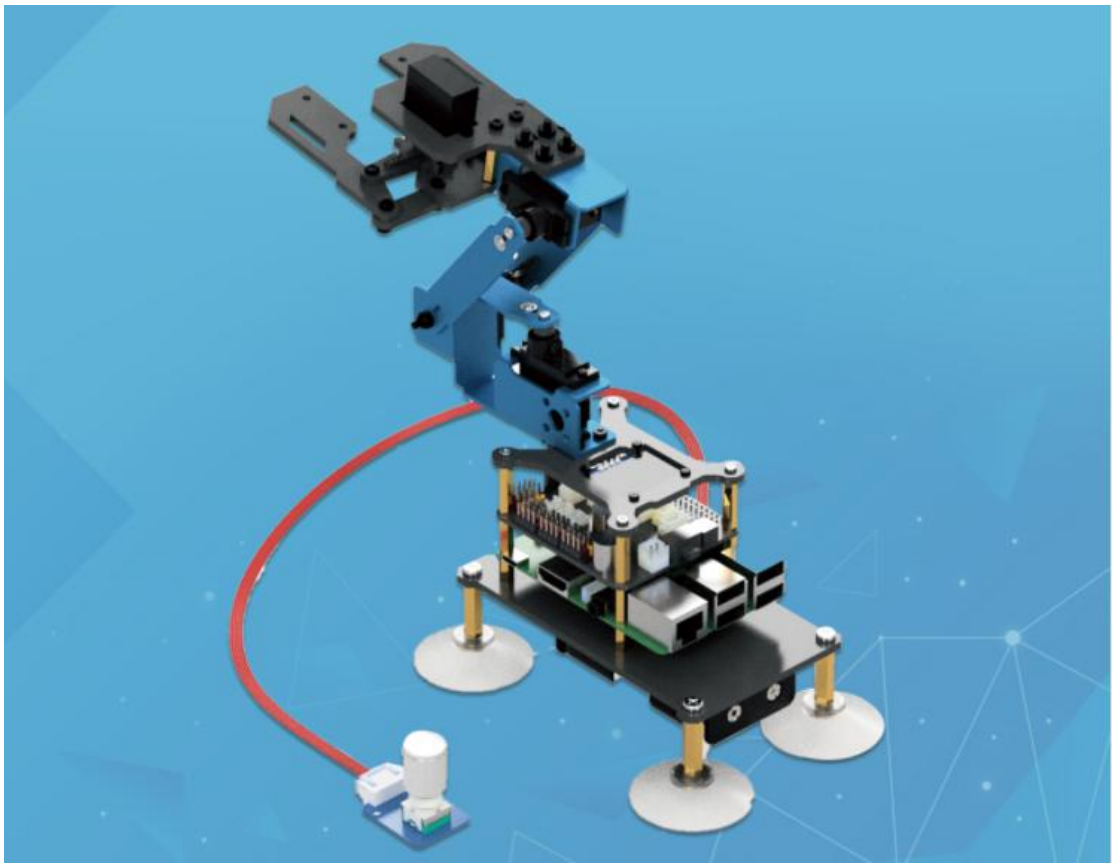
sudo python3 OLED.py

```
pi@raspberrypi:~/adeept_rasparms/CourseCode/050LED_Multithreading $ sudo  python3  OLED.py
```

9. After running the program successfully, you will see some text messages on the OLED screen of the robot.

Lesson 10 RaspArm-S Assembly Tutorial

In this lesson, we will learn how to assemble the RaspArm-S robotic arm. It has two forms of structure. The first form is a robotic arm that can grip objects. For the assembly tutorial, please refer to "10.2 RaspArm-S Robotic Arm Assembly Tutorial" in this lesson. "; The second form is a robotic arm that can be transformed into a "writing pen". For the assembly tutorial, please refer to "10.3 RaspArm-S Writing Pen Robotic Arm Assembly Tutorial" in this section of the course.



Before assembling, you need to prepare the parts used for assembling the RaspArm-S robot arm according to "10.1 Parts List".

10.1 Parts list

Electrical parts

Name	Quantity

AD002 servo	4
RobotHAT driver board	1
0.96 OLED screen	1
18650 battery holder	1
Rotary coding switch	1

Copper pillar

Name	Quantity
M2.5x10+6	12
M3x18	8
M4x22	4
M2.5x14	4

Screw

Name	Quantity
M4x6	4
M2.5x8	11
M2.5x10	1
M2x8	20
M3x10	24
M3x8 Countersunk head screw	2

Nut

Name	Quantity
M2	20
M3	10
M3_LOCK lock nut	10

Wire

Name	Quantity
4pin wire	1
Servo extension wire	3
Long 5pinwire (for rotary coding switch)	1

Mechanical Parts

Name	Quantity
Sucker	4

Tool

Name	Quantity
Cross Socket Wrench	1
Large Cross-head Screwdriver	1
Winding Pip	1

Own parts

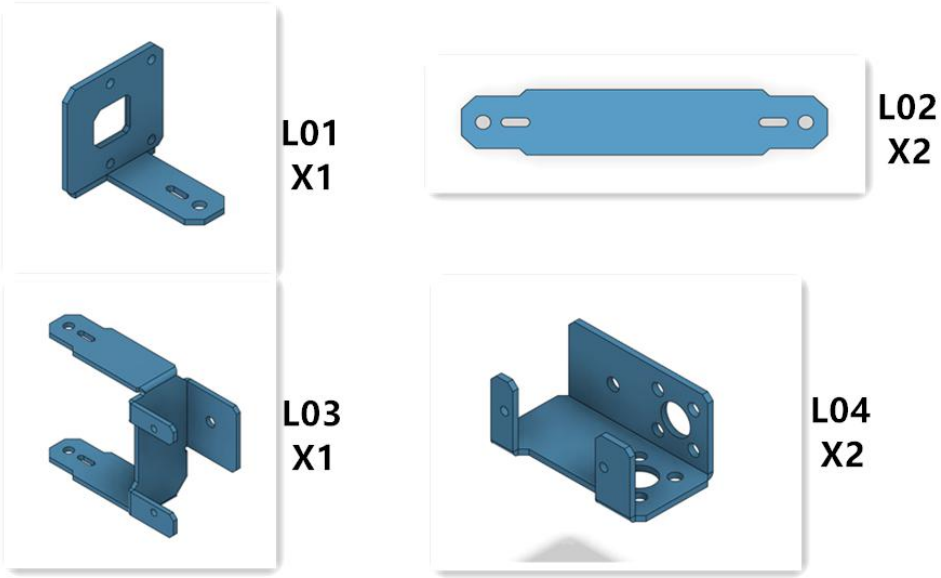
Name	Quantity
18650 battery	2
Raspberry Pi 3B/Raspberry Pi 4	1

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 color of the fastener in the tutorial to determine which type of screw and nut to use.

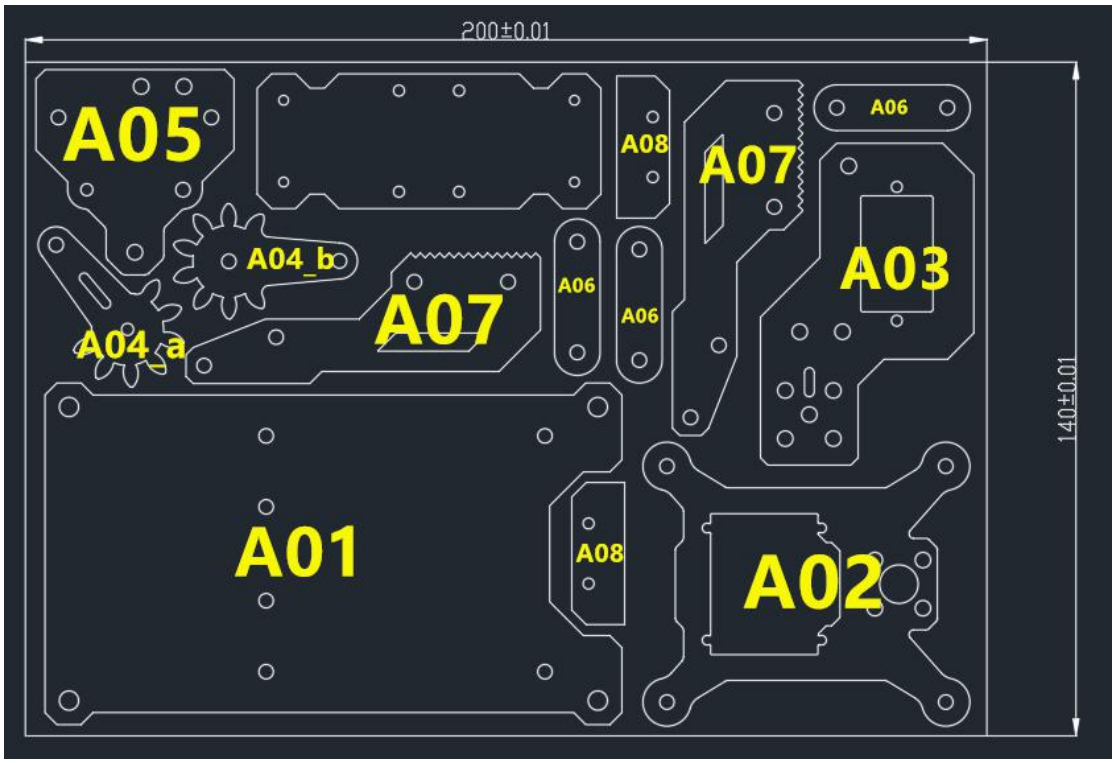
The actual color of the product is subject to the product, and the actual screws are not colored.

Names of aluminum alloy parts (the numbers in the figure are marked for the convenience of explaining the assembly)



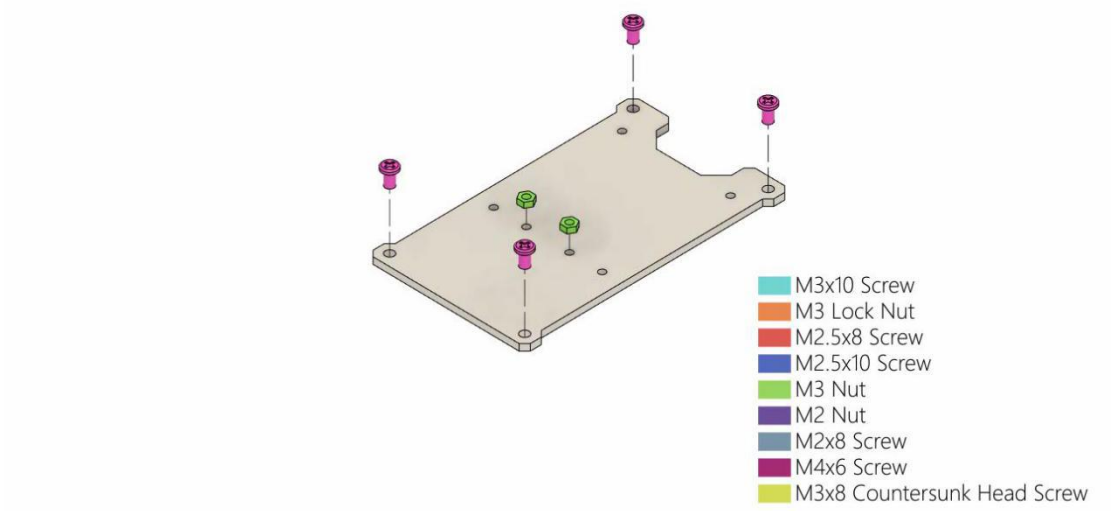
Acrylic part name (the number in the figure is for the convenience

of explaining the assembly and marking)



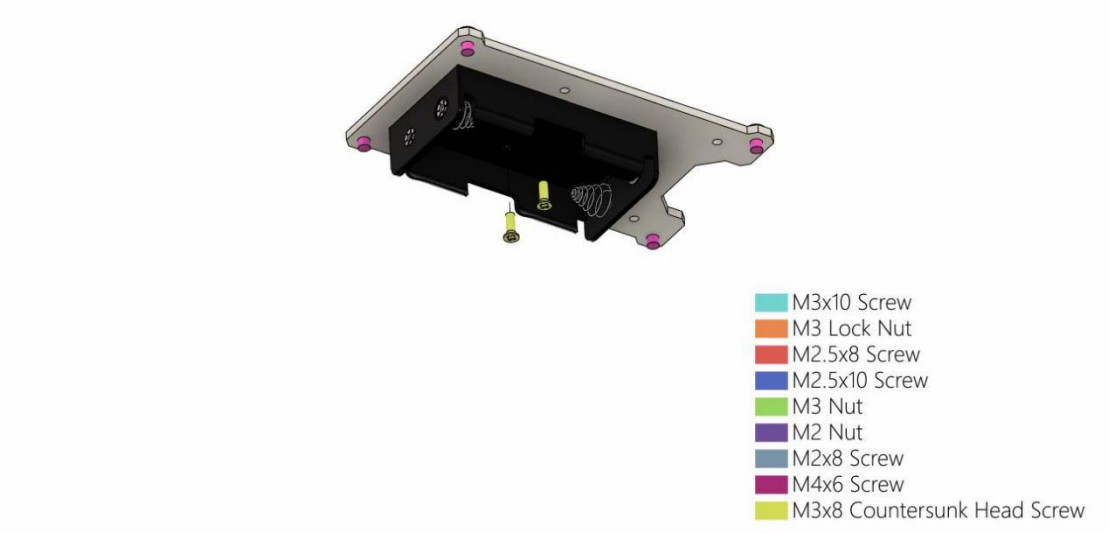
10.2 RaspArms Robotic Arm Assembly Tutorial

(1) Use 4 M4x6 Screws and 2 M3 Nuts to install on the acrylic part A01 as shown in the figure below.



(2) Use 2 M3x8 Countersunk head screws to install the 18650 battery holder on

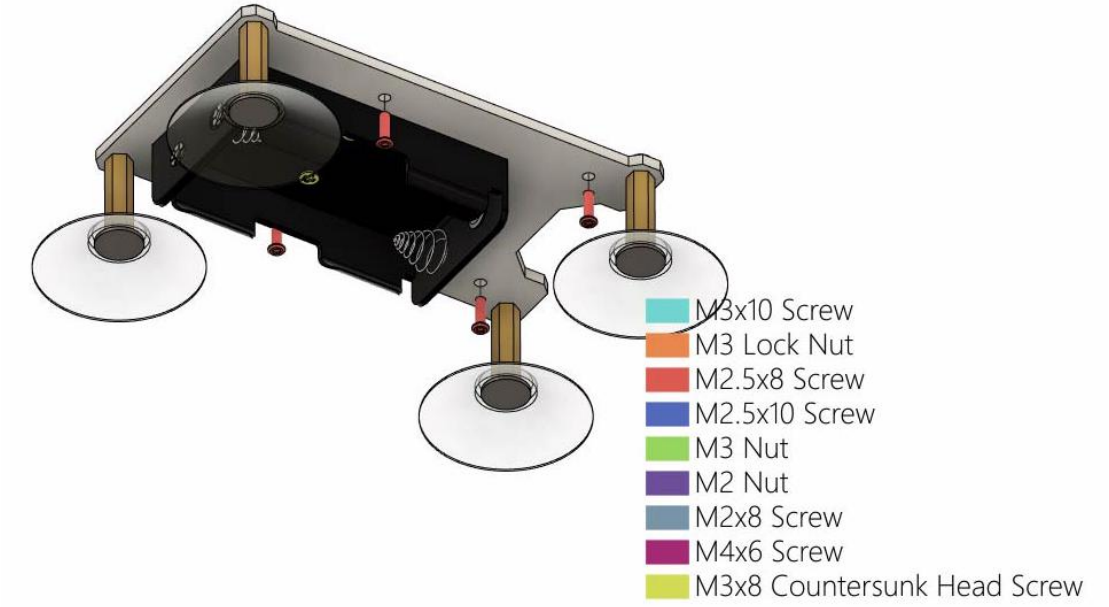
the acrylic part A01.



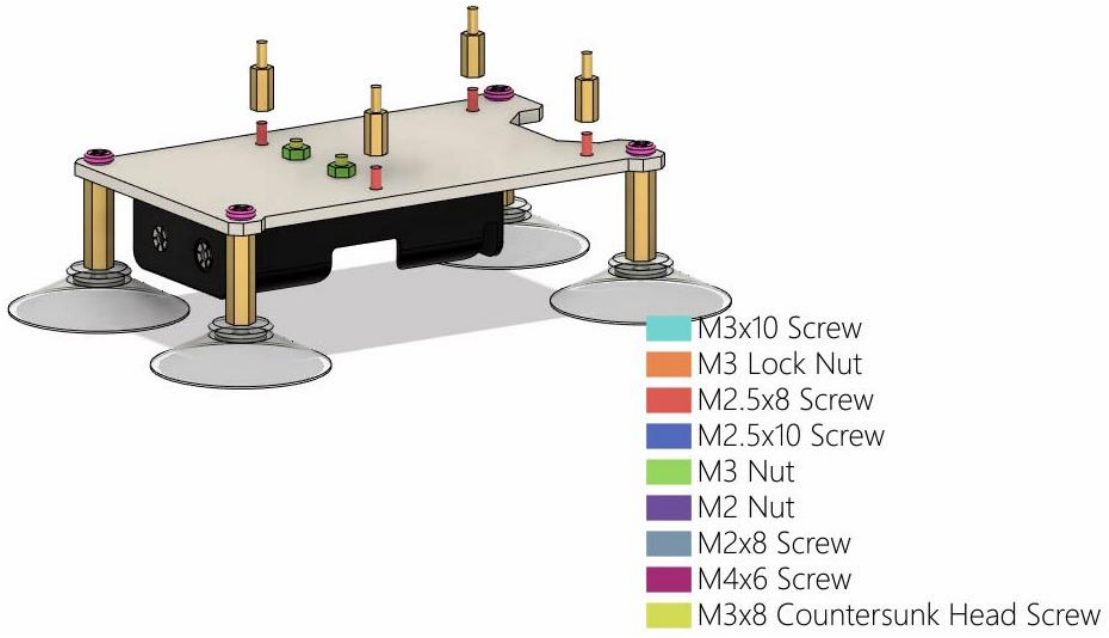
(3) Install 4 M4x22 Copper standoffs under the acrylic part A01.

Then install 4 sucker on the M4x22 Copper standoff.

(4) Install 4 M2.5x8 Screws on the acrylic part A01.

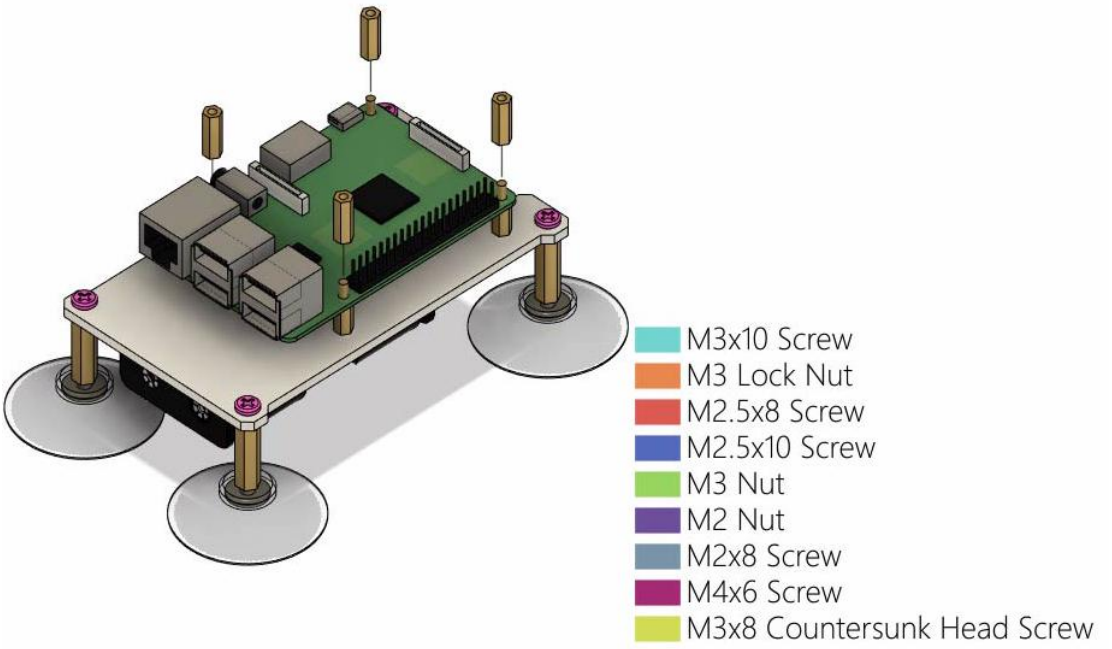


Then install 4 M2.5x10+6 Copper standoffs on the M2.5x8 Screw.

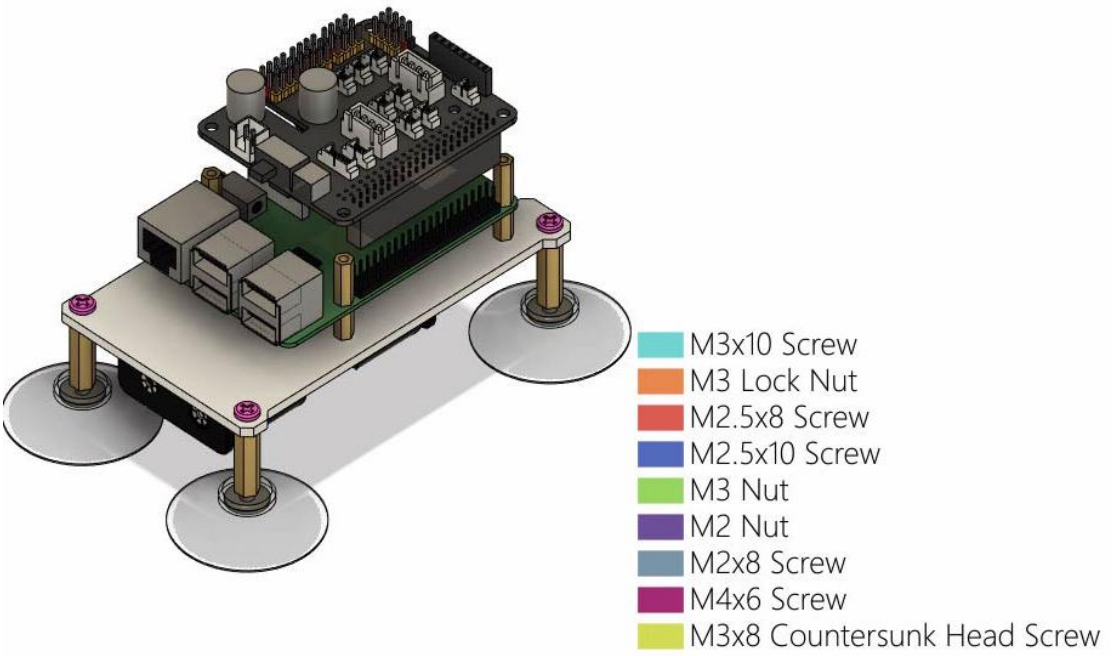


(5) Install the Raspberry Pi on the M2.5x10+6 Copper standoff;

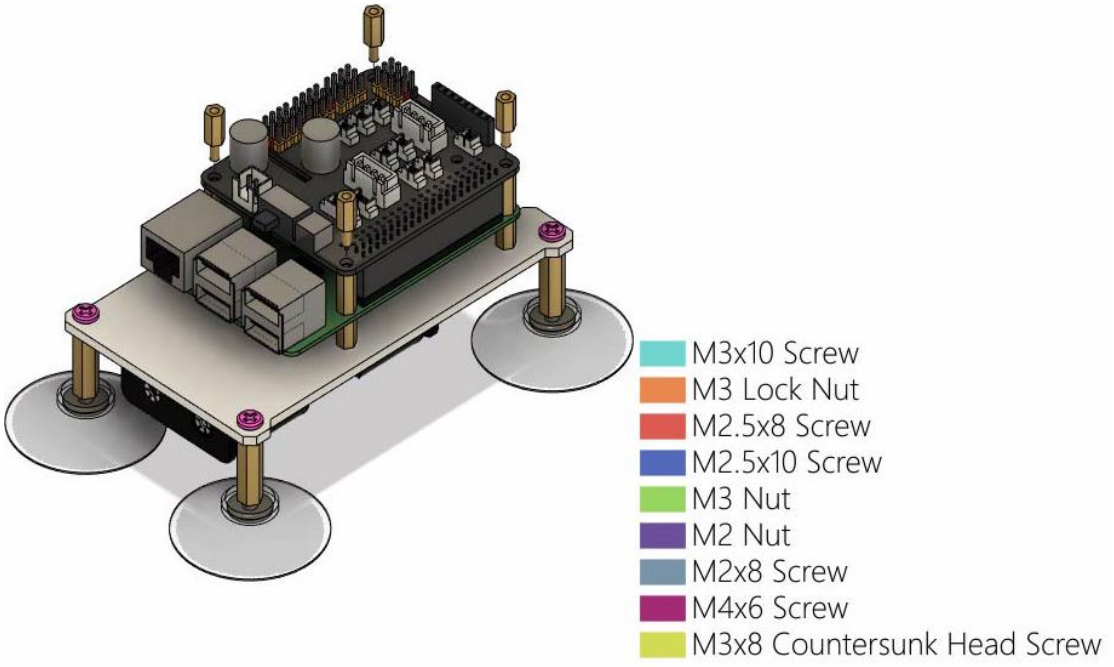
Use 4 M2.5x14 Copper standoffs to fix on the Raspberry Pi.



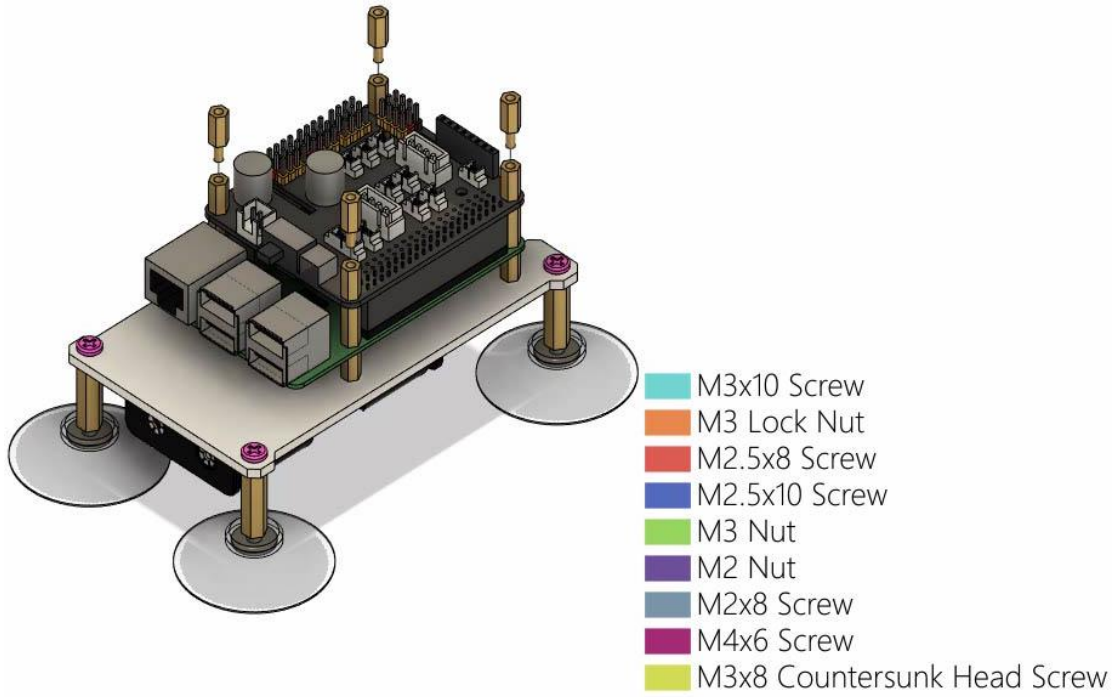
(6) Install the RobotHAT driver board on the Raspberry Pi, and pay attention to the alignment of the GPIO pin ports.



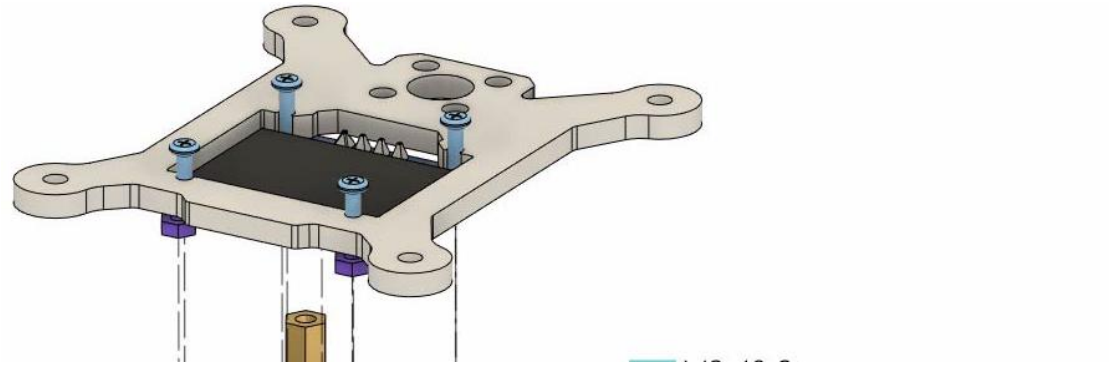
(7) Use 4 M2.5x10+6 Copper standoffs to fix RobotHAT driver board.



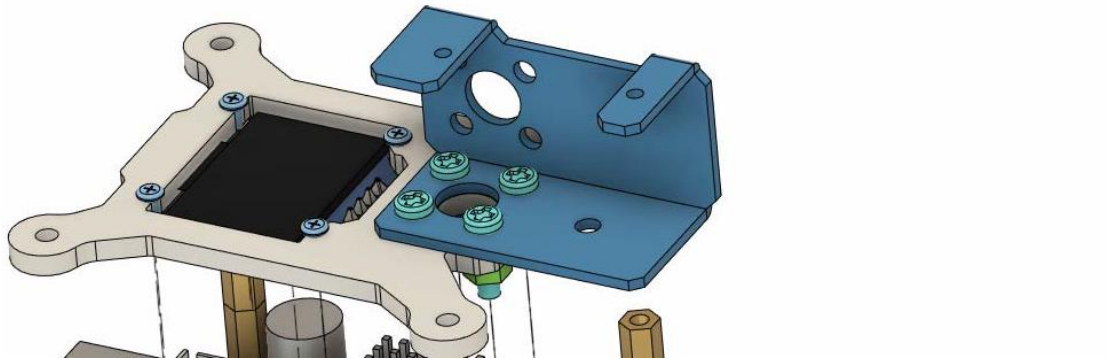
Then use 4 M2.5x10+6 Copper standoffs to connect and fix.



(8) First use 4 M2x8 Screws and 4 M2 nuts to fix the 0.96 OLED screen on the acrylic part A02. Note the orientation of the 0.96 OLED screen.



(9) Place the aluminum alloy part L04 on the acrylic part A02, the holes need to be aligned, and then use 4 M3x10 Screws and 4 M3 Nuts to fix L04.



Wiring method

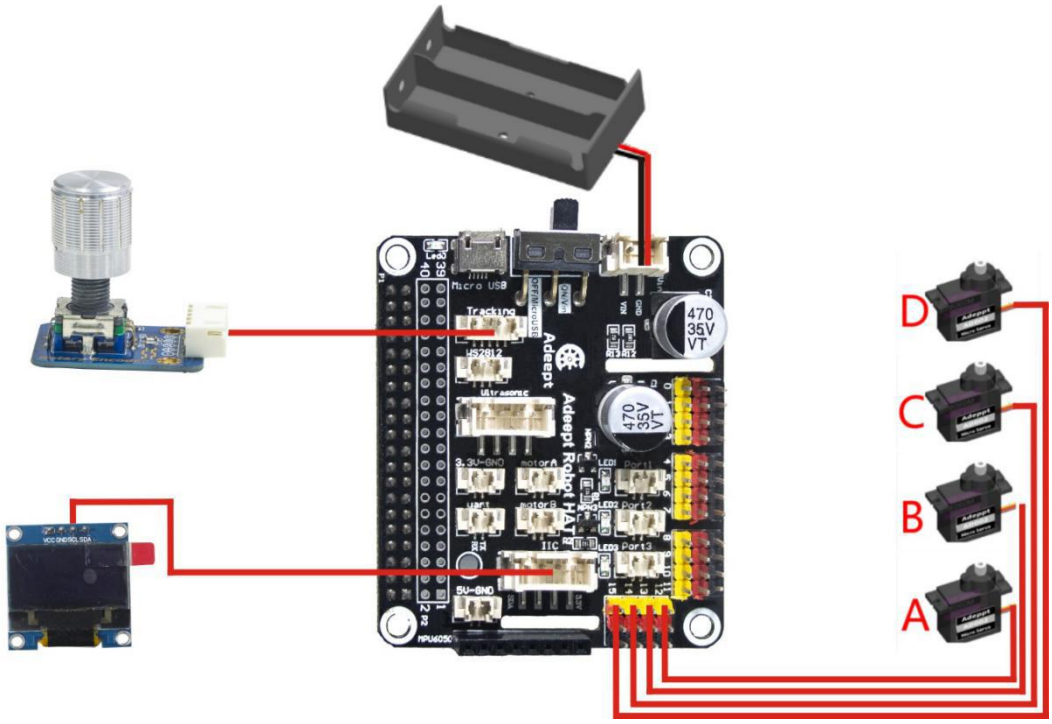
When assembling to this step, you need to consider how to wire in advance.

First, you need to connect the external power cord to the Vin interface of the RobotHAT driver board;

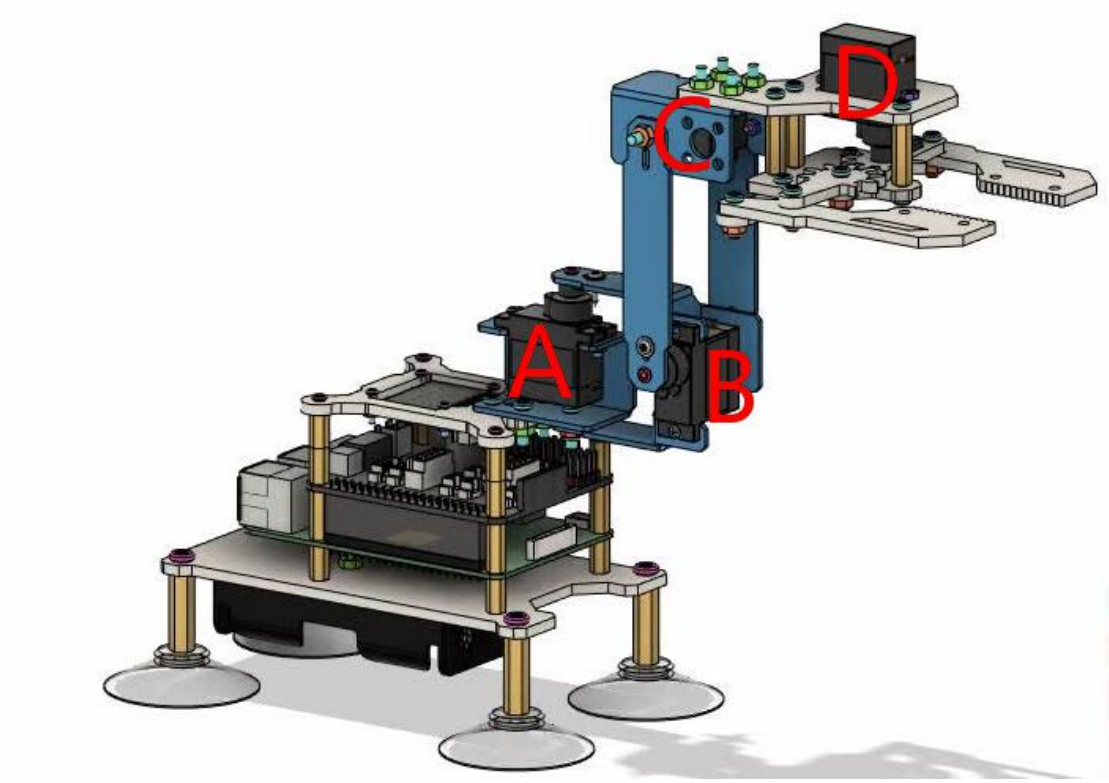
Use the 5pin wire to connect the rotary encoder switch to the Track interface of the RobotHAT drive board;

Use a 4pin wire to connect the 0.96 OLED screen to the IIC interface of the RobotHAT driver board;

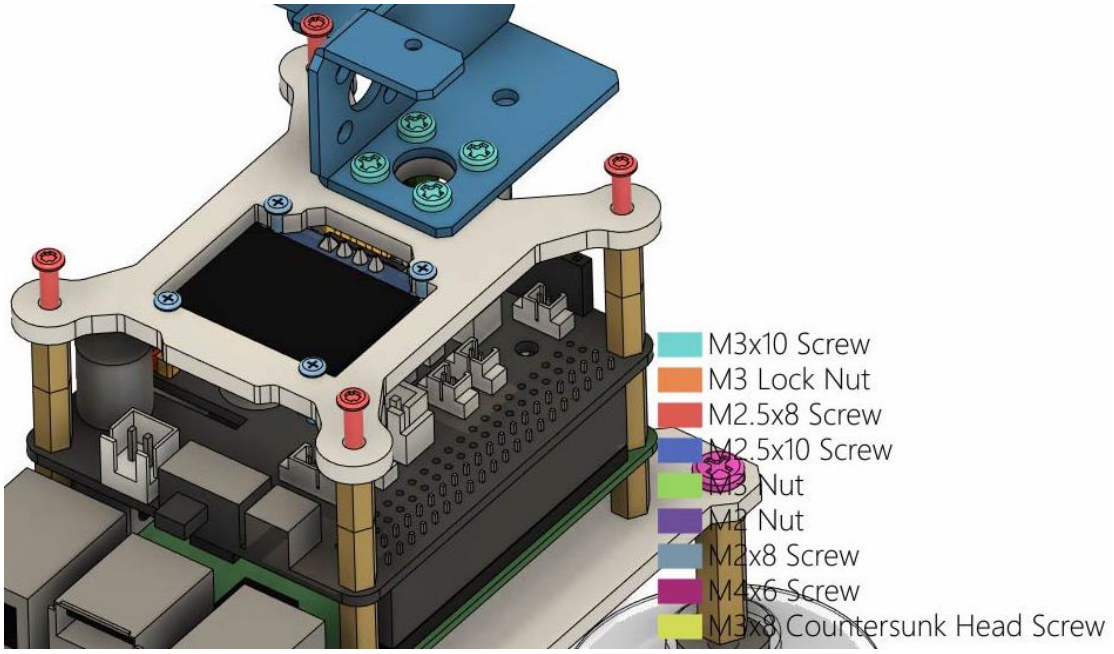
Connect the AD002 servo to the Servo interface. In our course example, the 4 servos are divided into A, B, C, D from bottom to top, and the 4 servos A, B, C, D are connected to Servo respectively on pins 12, 13, 14, and 15 of the interface, pay attention to the color alignment when wiring, otherwise the wiring will be reversed.



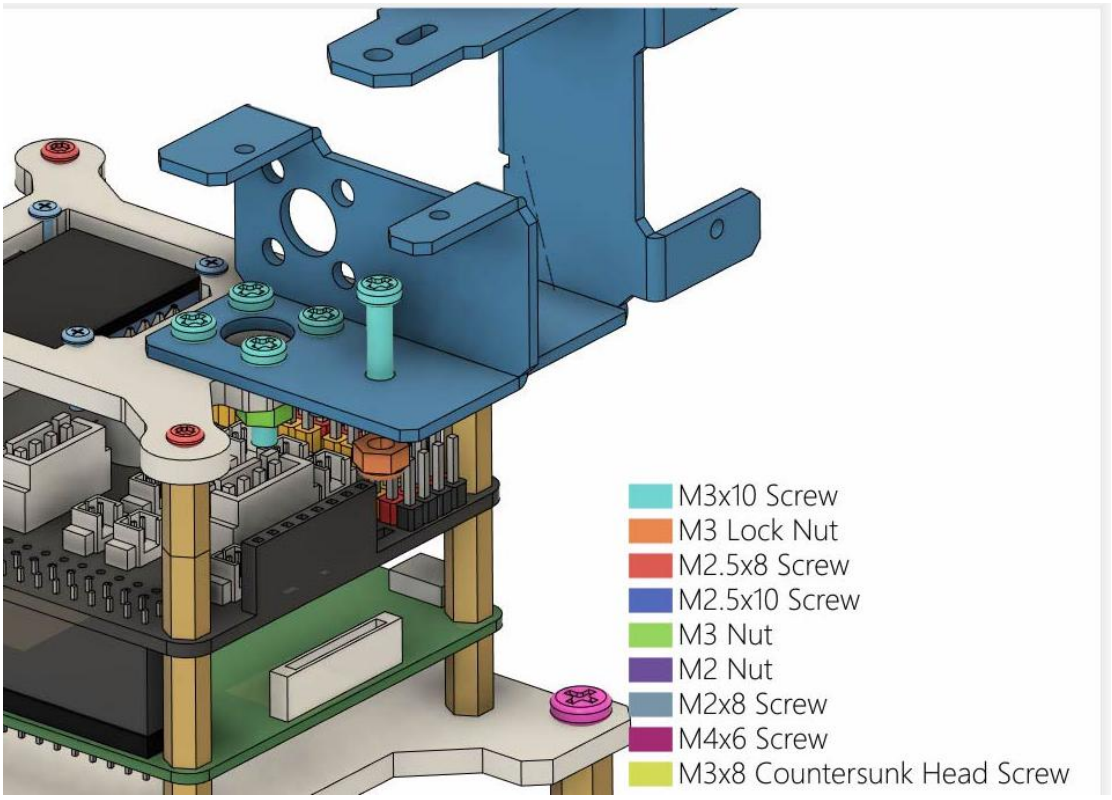
The figure below is the structure diagram of the servo of the robotic arm.



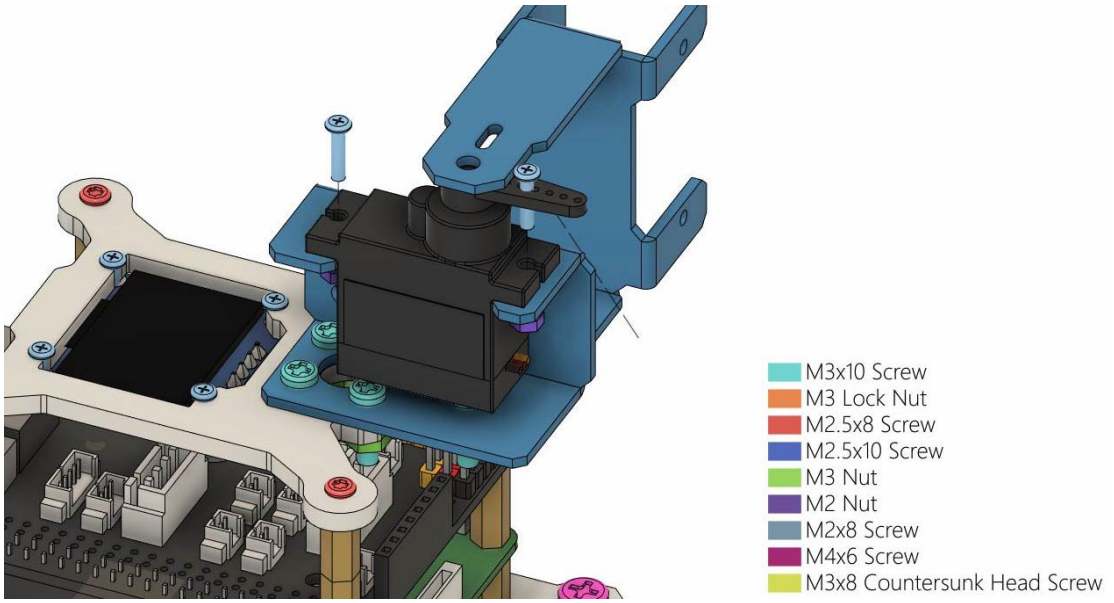
(10) Use 4 M2.5x8 Screws to install the assembled acrylic part A02 above the RobotHAT drive board.



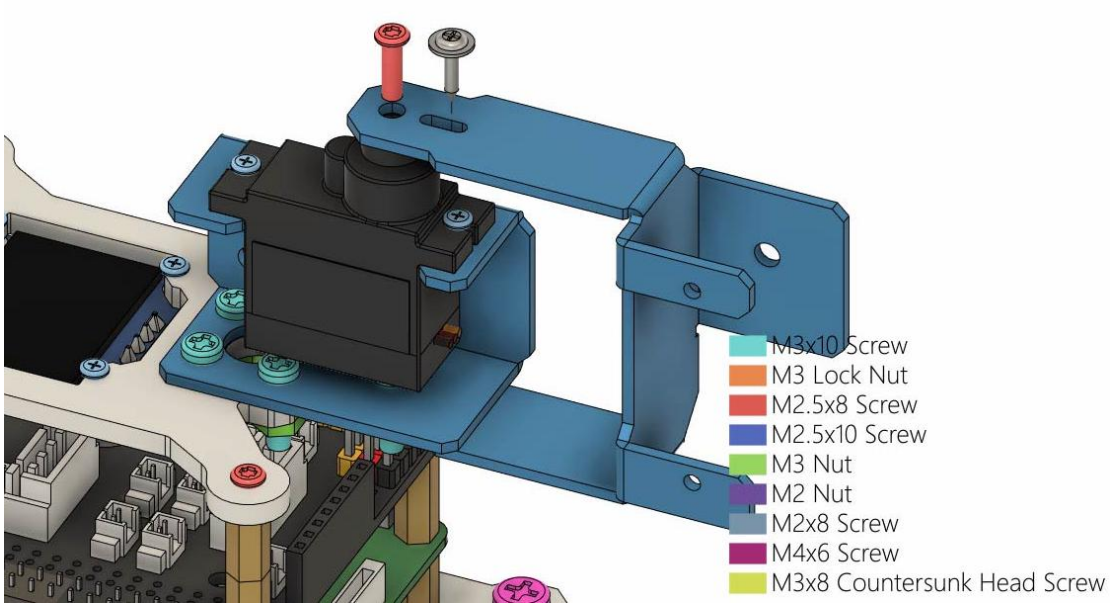
(11) Install the aluminum alloy part L03 on the aluminum alloy part L04, and fix it with 1 M3x10 Screws and 1 M3 Lock.



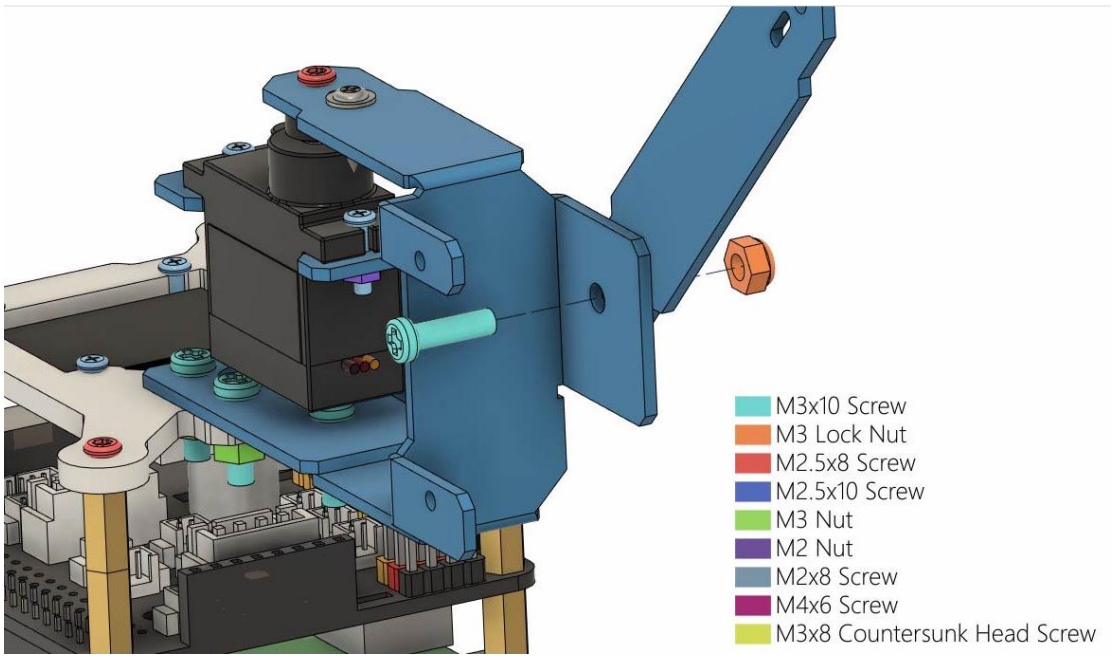
(12) Install AD002 servo on L03. The servo needs to be equipped with its own rocker arm. Use 2 M2x8 Screws and 2 M2 Nuts to fix the servo. The servo number of this part is A.



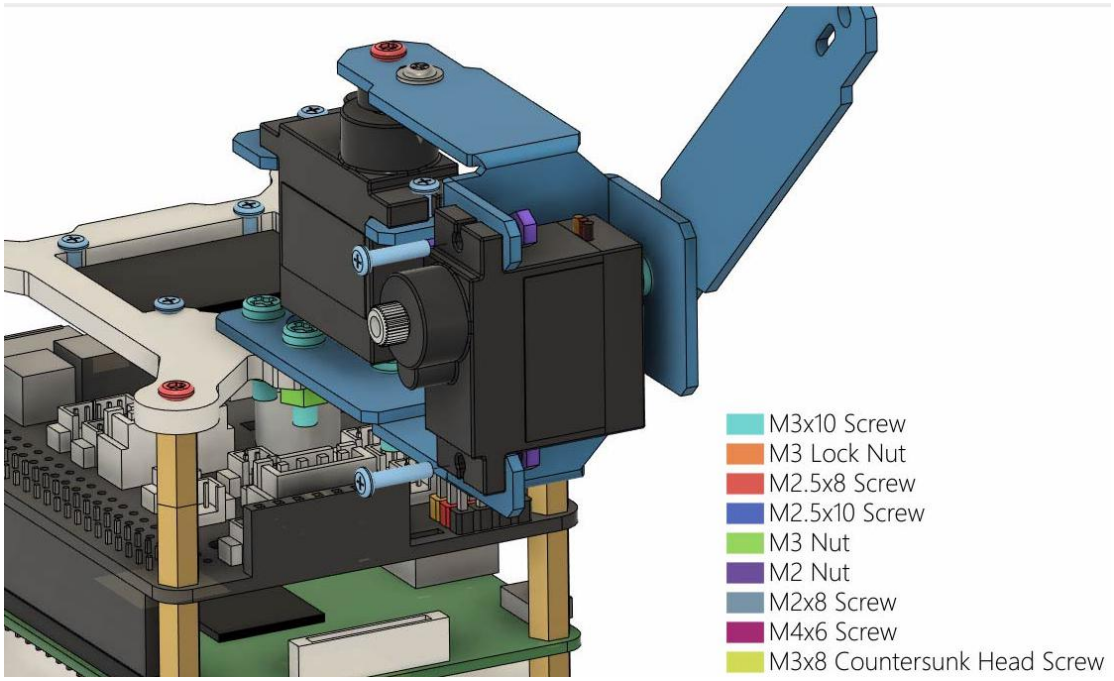
(13) Rotate L03 to the position as shown in the figure below, and use 1 M2.5x8 Screw and 1 M1.4x6 self-tapping screw that comes with the servo to fix L03.



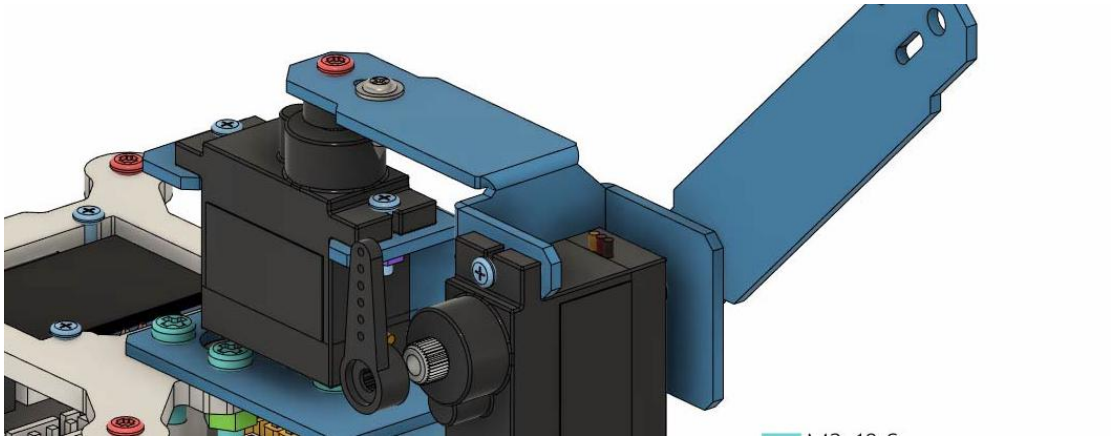
(14) Install the aluminum alloy part L02 on the L03, and use 1 M3x10 Screw and 1 M3 Lock Nut to fix L02.



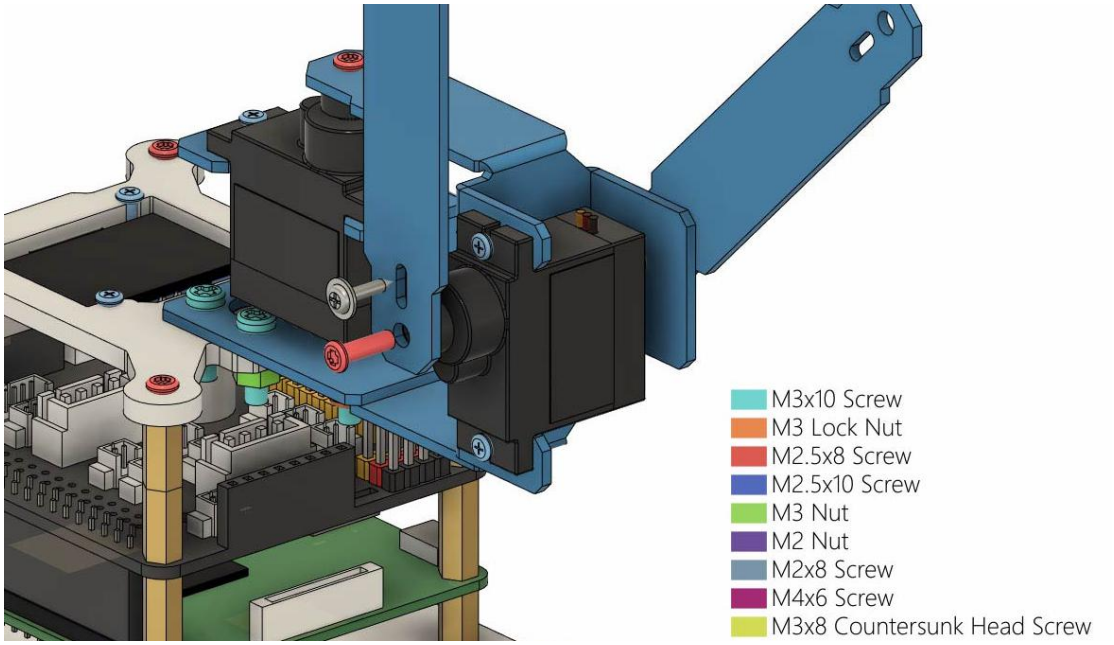
(15) Install AD002 stervo on L03 as shown in the figure below, and fix AD002 servo with 2 M2x8 Screws and 2 M2 Nuts. The servo number for this part is B.



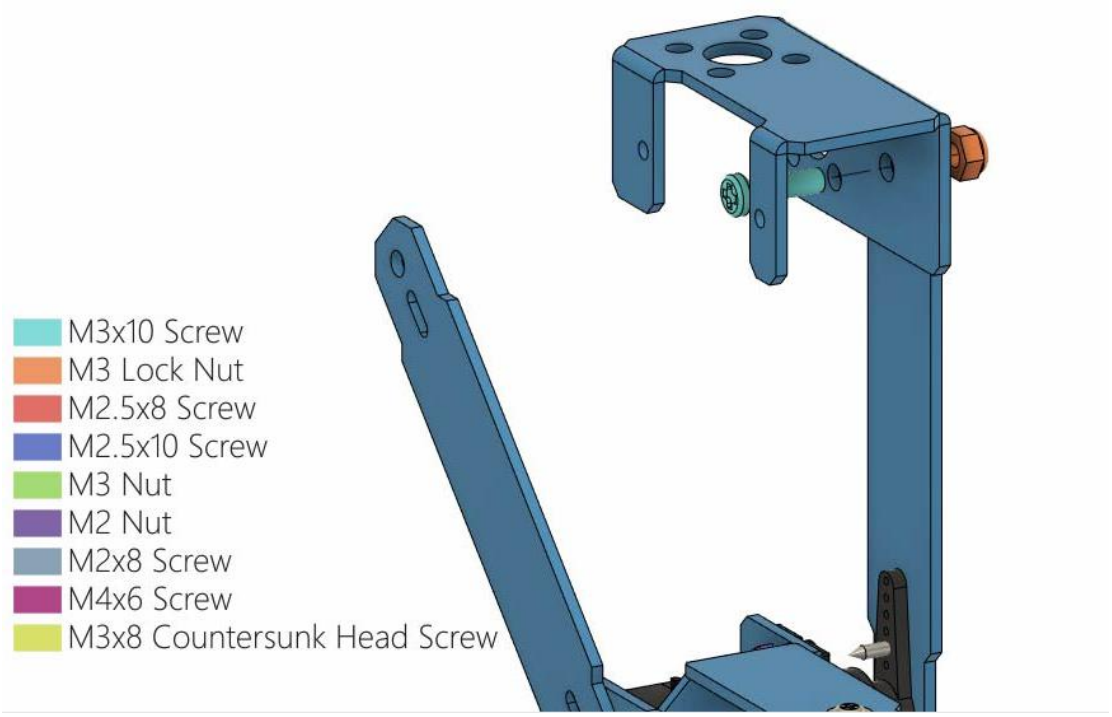
Then install the rocker arm that comes with the servo to the servo.



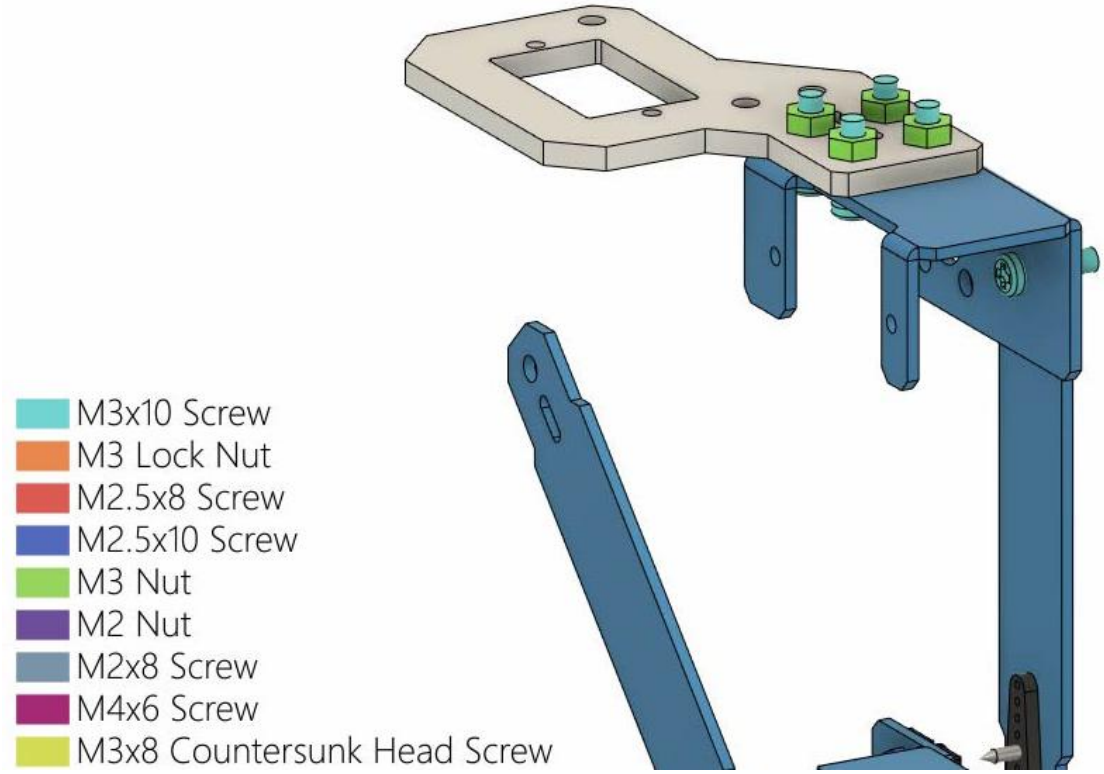
(16) Install another L02 on the position of the servo as shown in the figure below, and use 1 M2.5x8 Screw and 1 M1.4x6 self-tapping screw that comes with the servo to fix L02.



(17) Install another aluminum alloy part L04 on L02 as shown below, and fix L04 with 1 M3x10 Screw and 1 M3 Lock Nut.

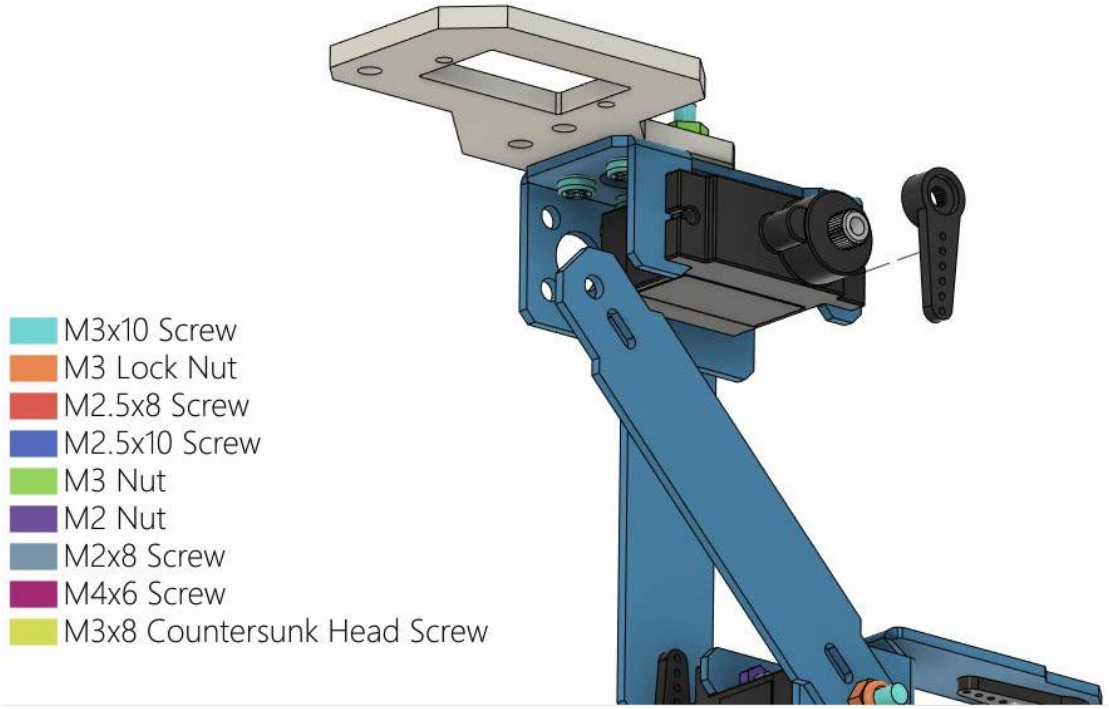


(18) Install the acrylic part A03 on L04 as shown in the figure below, and use 4 M3x10 Screws and 4 M3 Nuts to fix A03.

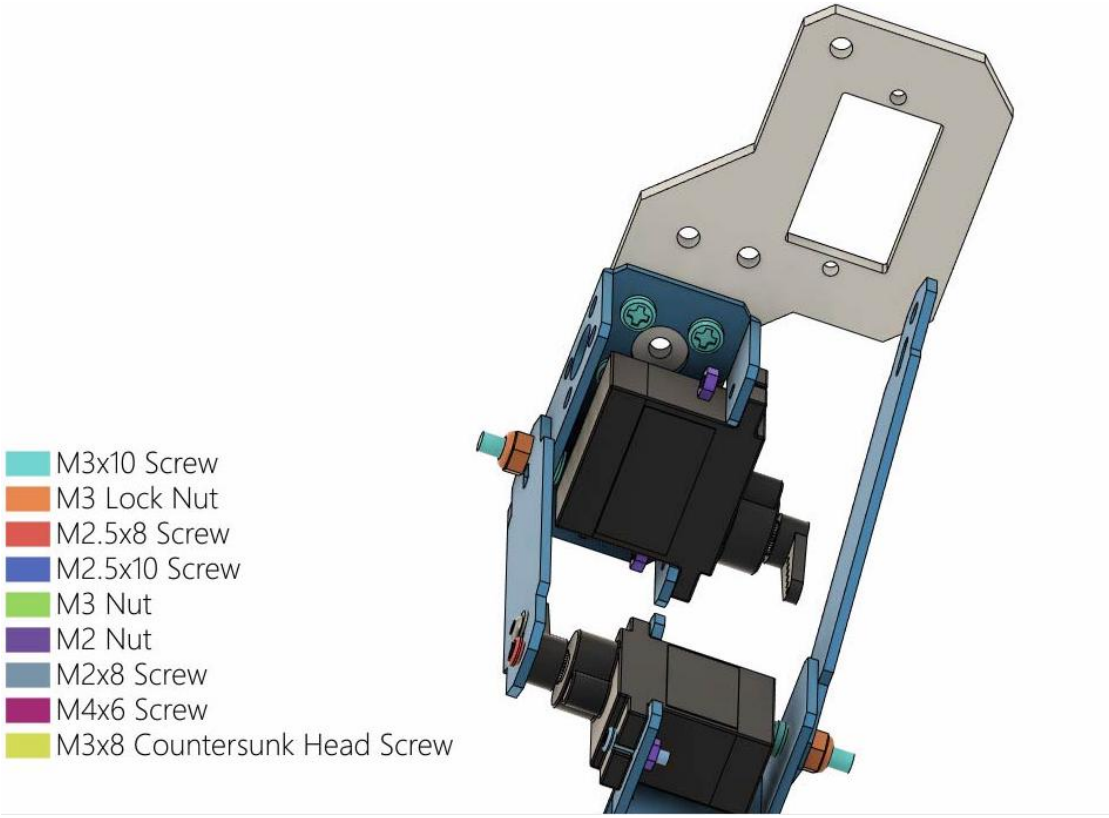


(19) Install AD002 servo on L04 as shown in the figure below, and install the servo rocker arm. The number of this part of the servo in the course is D3, which

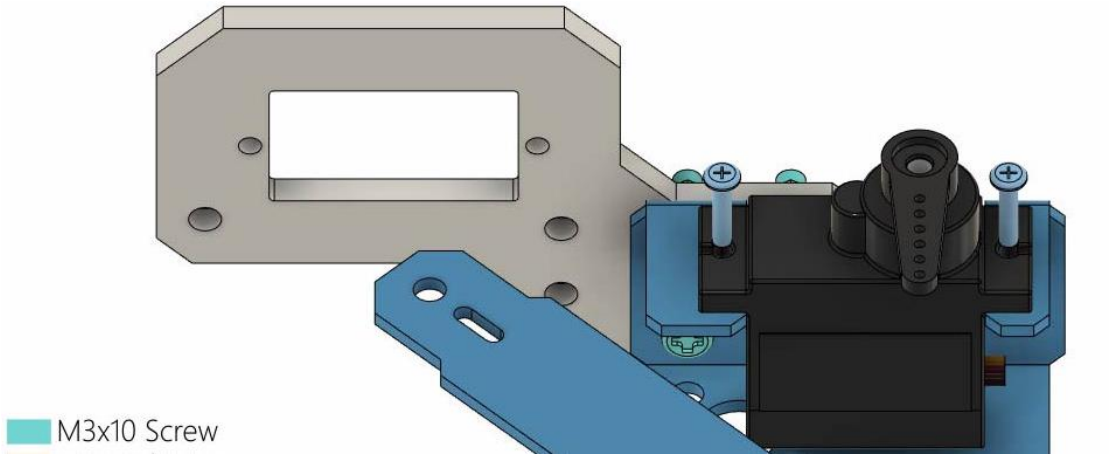
means the third servo of the robot arm from bottom to top. The servo number for this part is C.



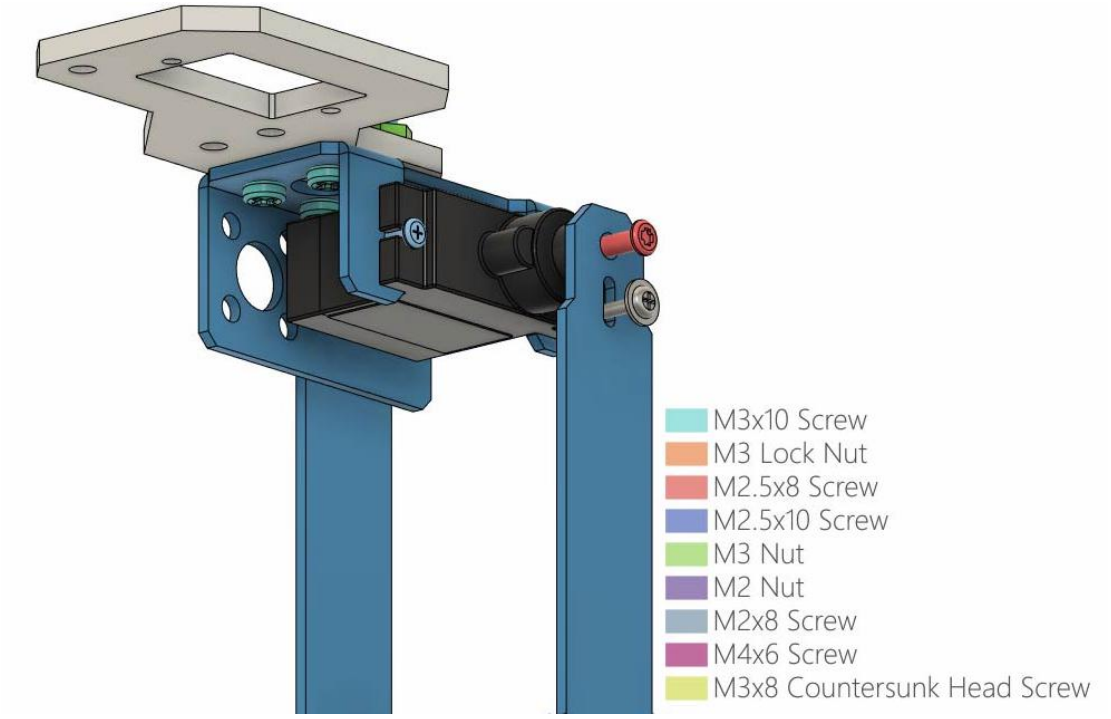
Use 2 M2 Nuts to fix the servo.



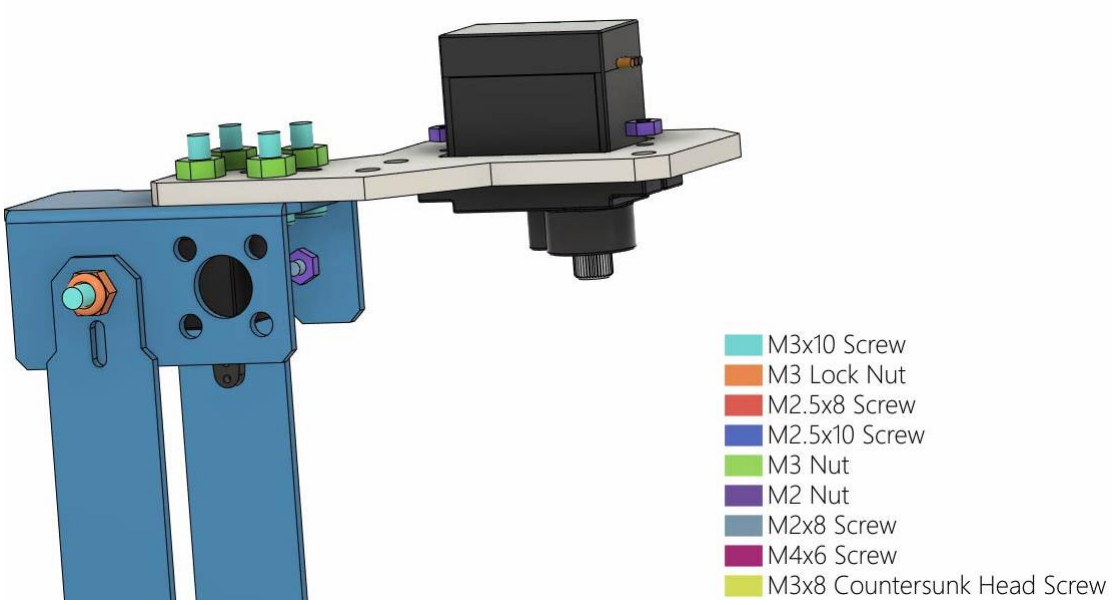
Then use 2 M2x8 Screws to fix the servo.



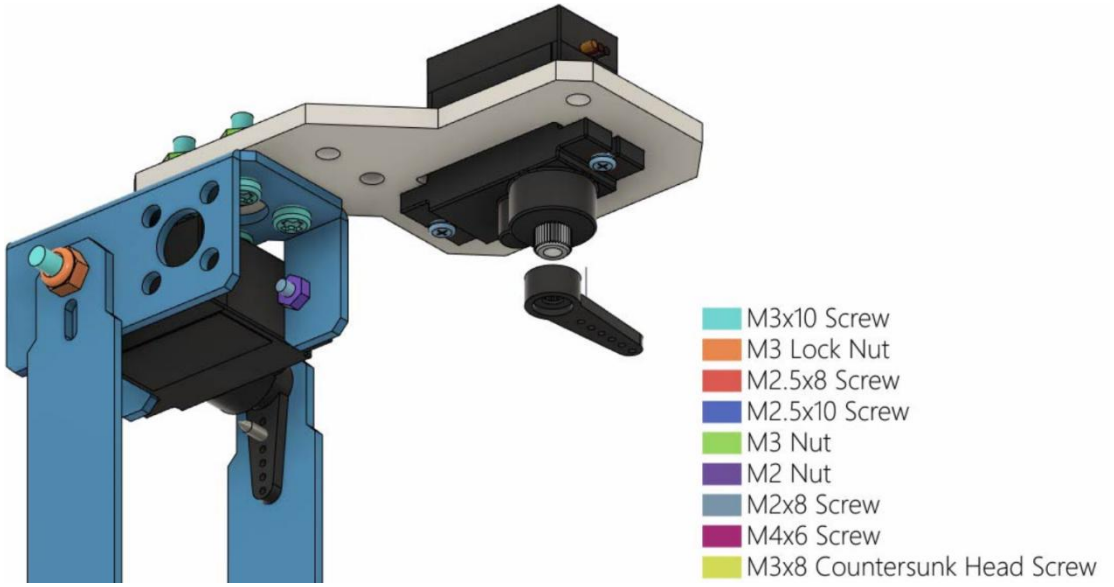
(20) Use 1 M2.5x8 Screw and 1 M1.4x6 self-tapping screw that comes with the servo to fix L02 on the servo.



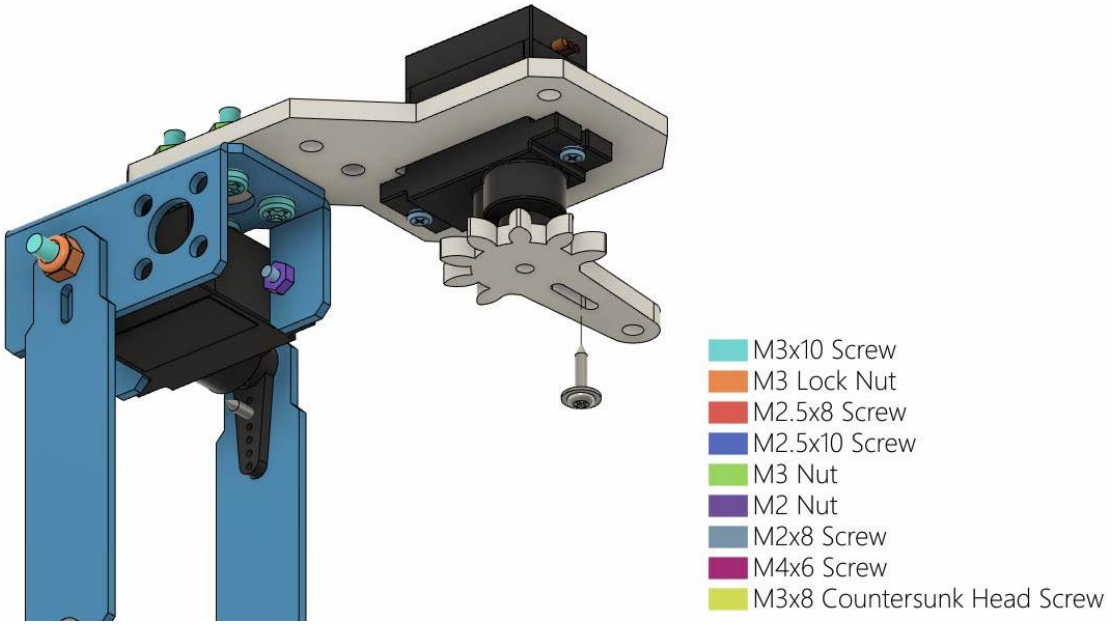
(21) Install AD002 servo on A03 as shown in the figure below, pay attention to the direction of the servo. You need to find 2 M2 Nuts to fix the servos. The servo number of this part is D.



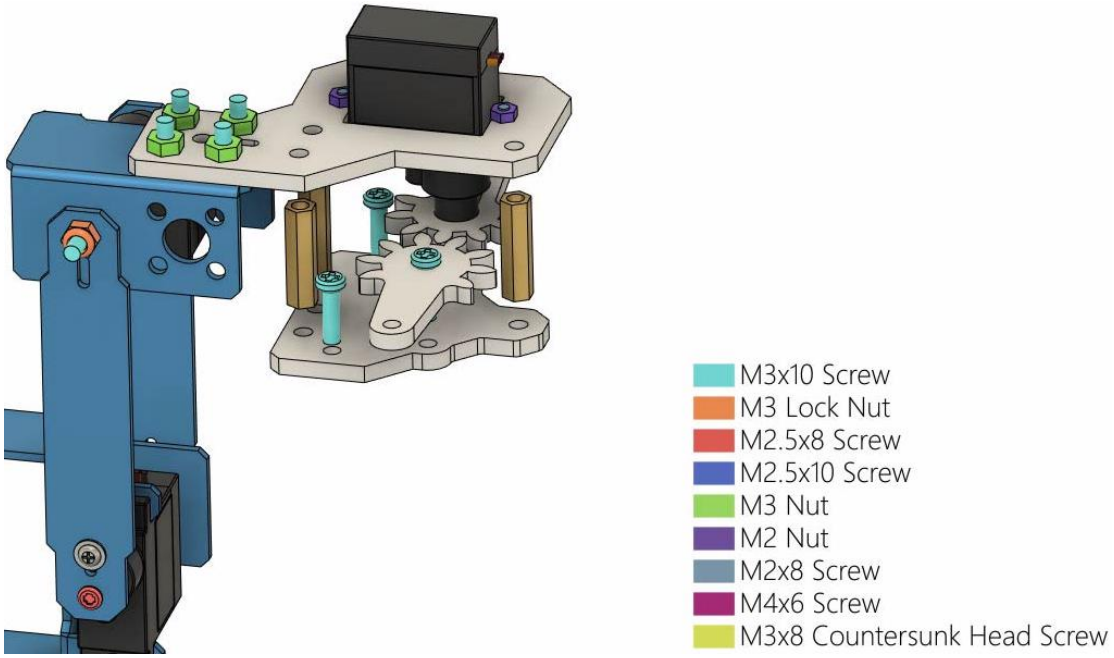
Use 2 M2x8 Screws to fix the servo, and then install the rocker arm that comes with the servo.

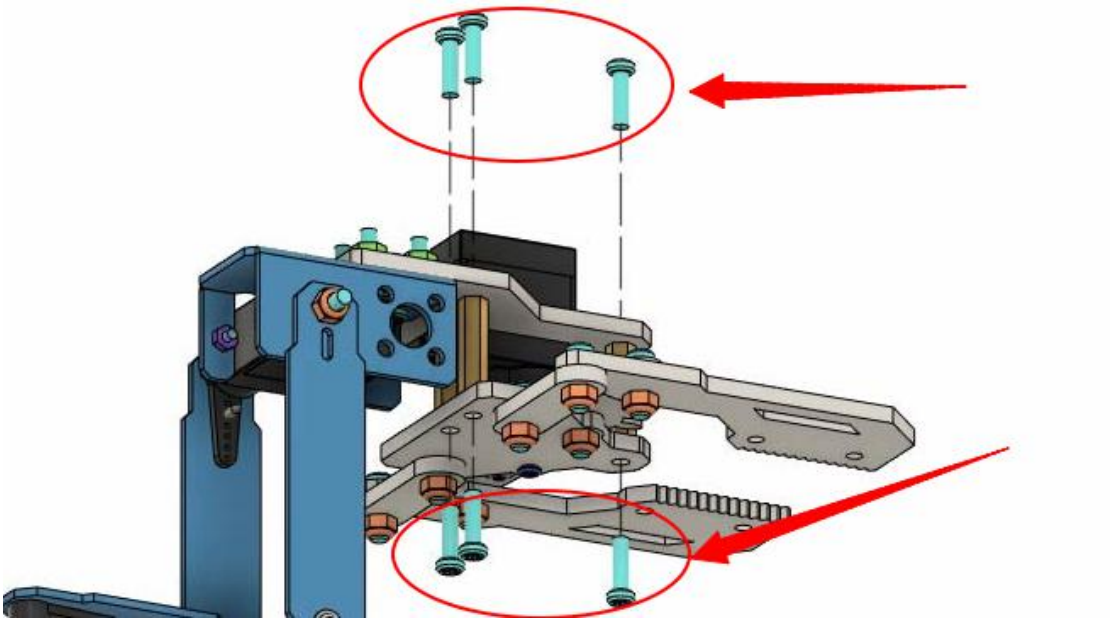


(22) Install the acrylic part A04_a on the rocker arm of the servo and fix it with a M1.4x6 self-tapping screw that comes with the servo.

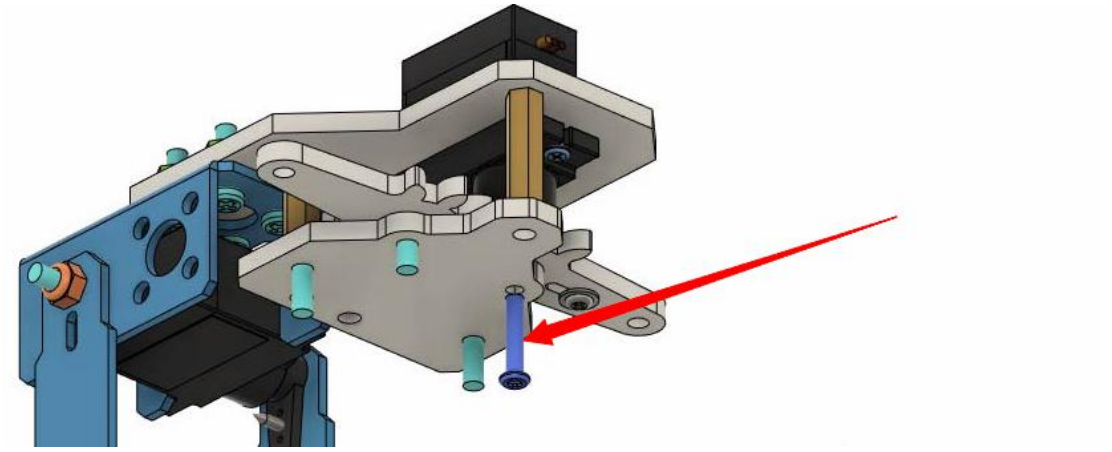


(23) Use 1 M3x10 Screw to pierce another acrylic part A04_b, and place this A04_b and the A04_a connected to the servo on the same horizontal line; find 2 M3x10 Screws, 3 M3x18 Copper standoffs, and 1 Acrylic part A05. Place them as shown in the figure below. When assembling, first install and fix 3 M3x18 Copper standoffs on the 3 holes on the A03 of the servo, and use 3 M3x10 Screws to fix the M3x18 Copper standoff. Then install the A05 below on the M3x18 Copper standoff and fix it with 3 M3x10 Screws. Finally, put two M3x10 Screws into the corresponding holes, and then install A04_b on the same horizontal line as A04_a.

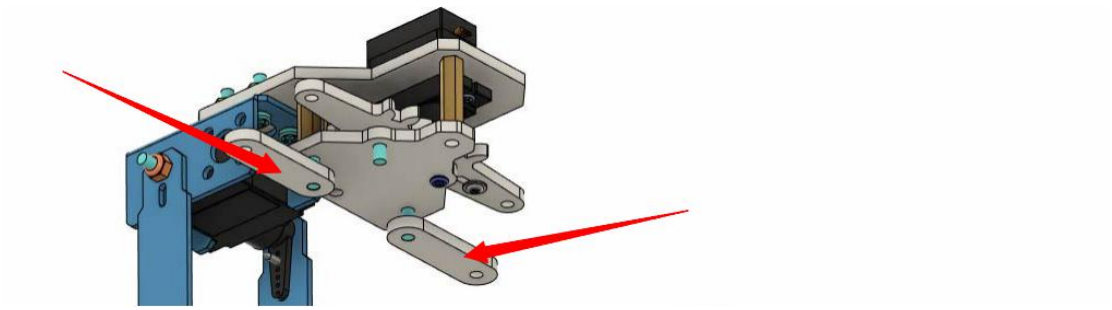




(24) Below A05, you need to use 1 M2.5x10 Screw to insert the hole connected with the servo, this screw needs to go through A04_a.



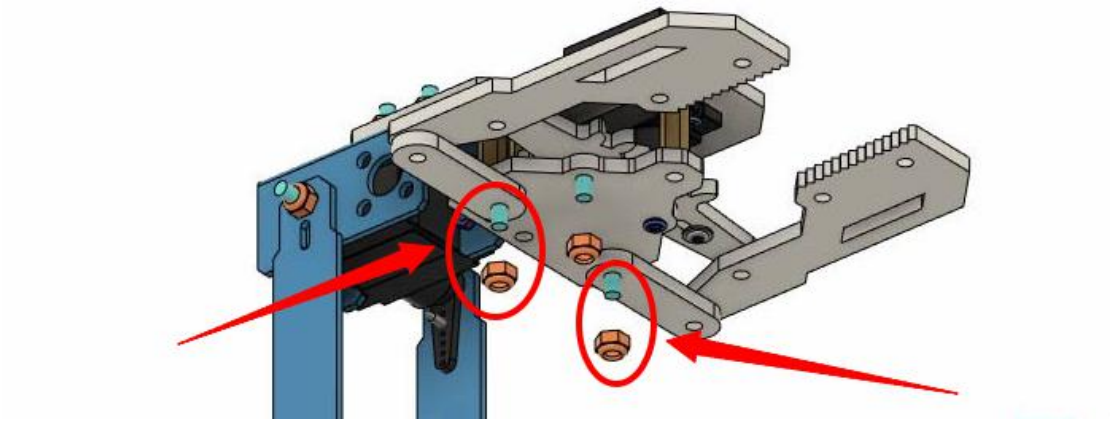
(25) Install two A06s below A05.



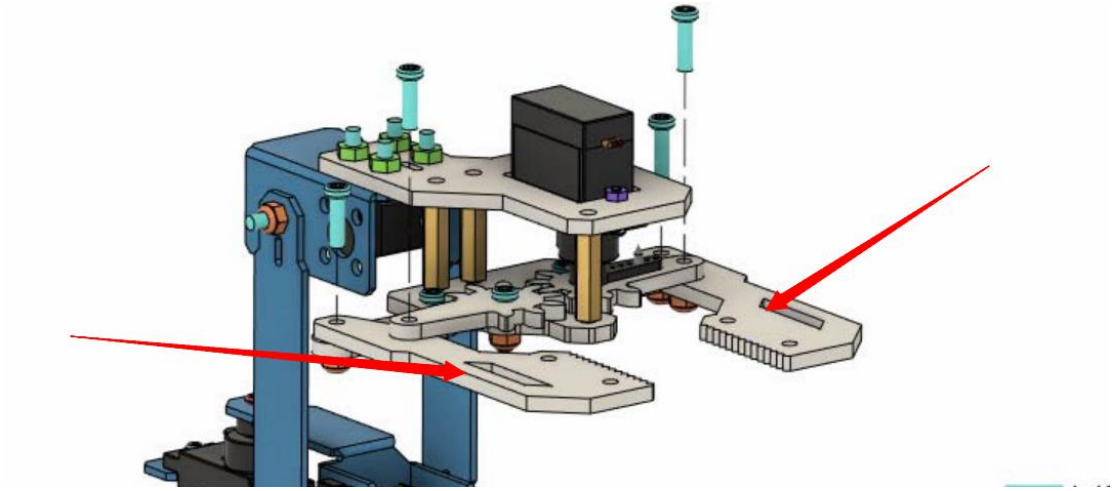
Use 3 M3 Lock Nuts to fix.

【Pay attention】

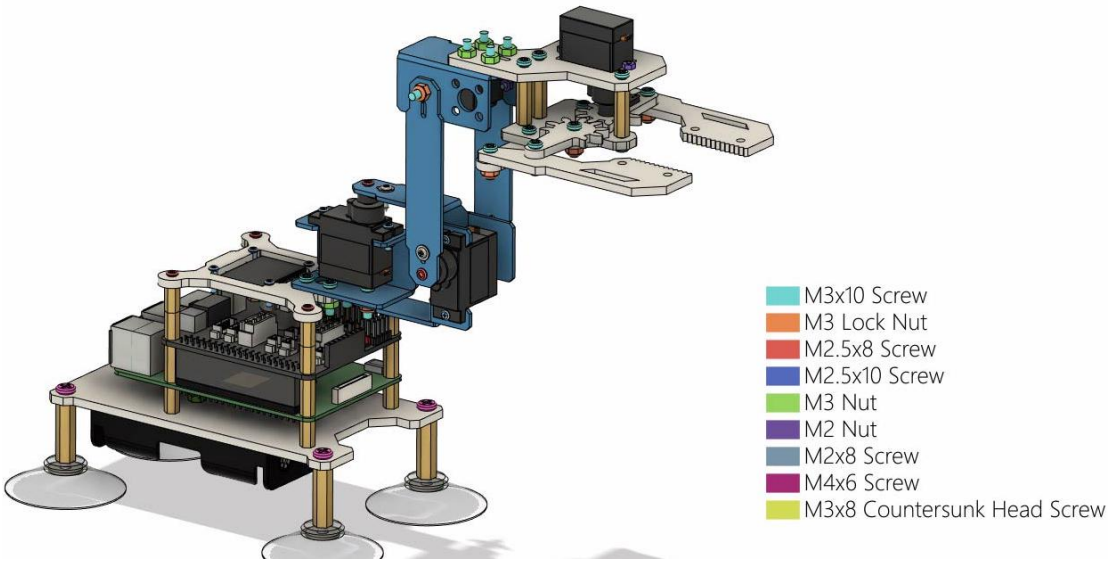
The two M3 Lock Nuts used to fix A06 should not be installed too tightly, loosen them slightly. If they are too tight, the robot arm will not be able to grip things.



(26) Install two A07s on A06 and fix them with four M3x10 Screws and four M3 Lock Nuts.



(27) The final assembly is completed.



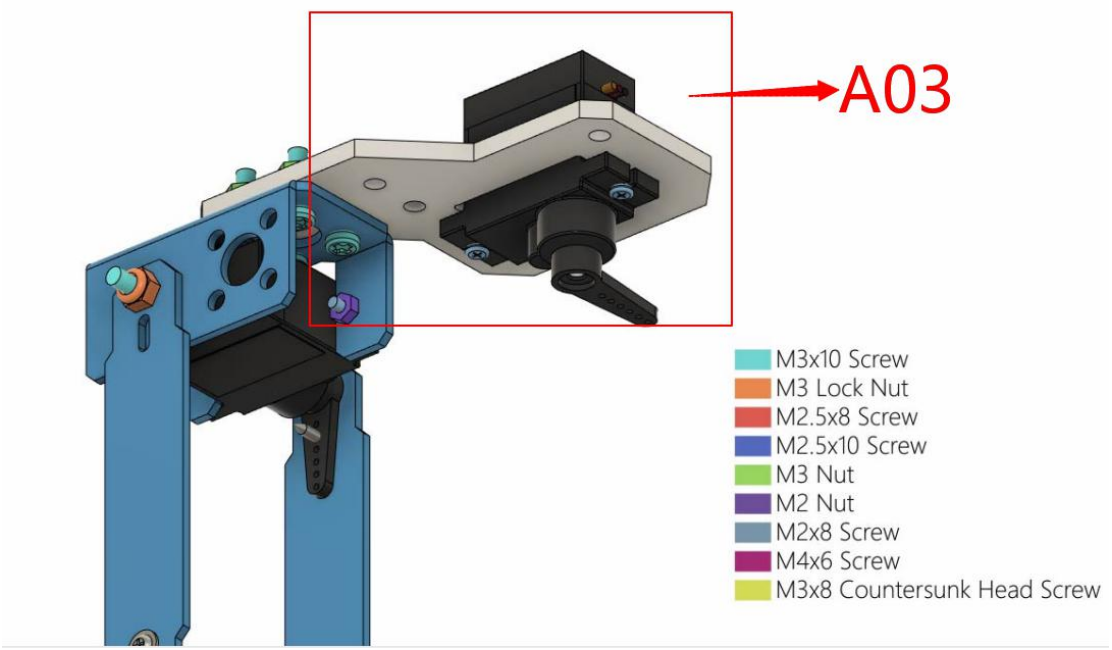
10.3 RaspArm-S writing pen robot arm assembly tutorial

If the shape and structure of the RaspArm-S robotic arm cannot satisfy your more creativity and ideas, then you can follow the assembly tutorial below to transform it into a "writing pen" shape robotic arm.

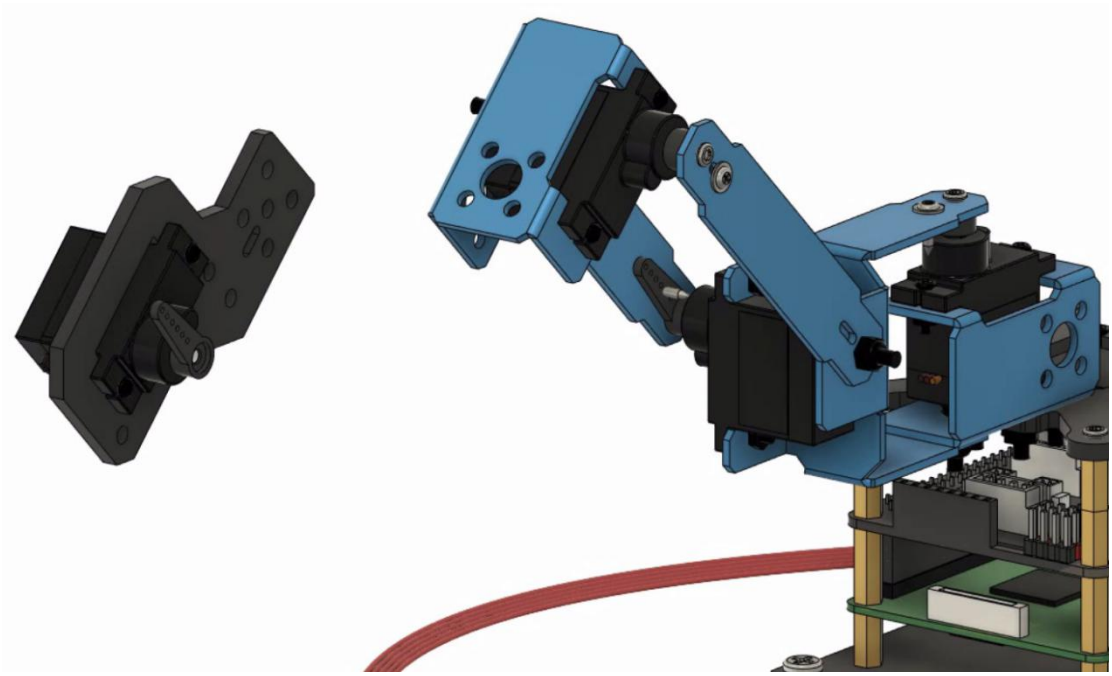
The RaspArm-S writing pen shape manipulator is modified from the shape and structure of the RaspArm-S manipulator arm, and its chuck is replaced with a structure that can be equipped with writing pens.

10.3.1 Assembling the robotic arm for the first time

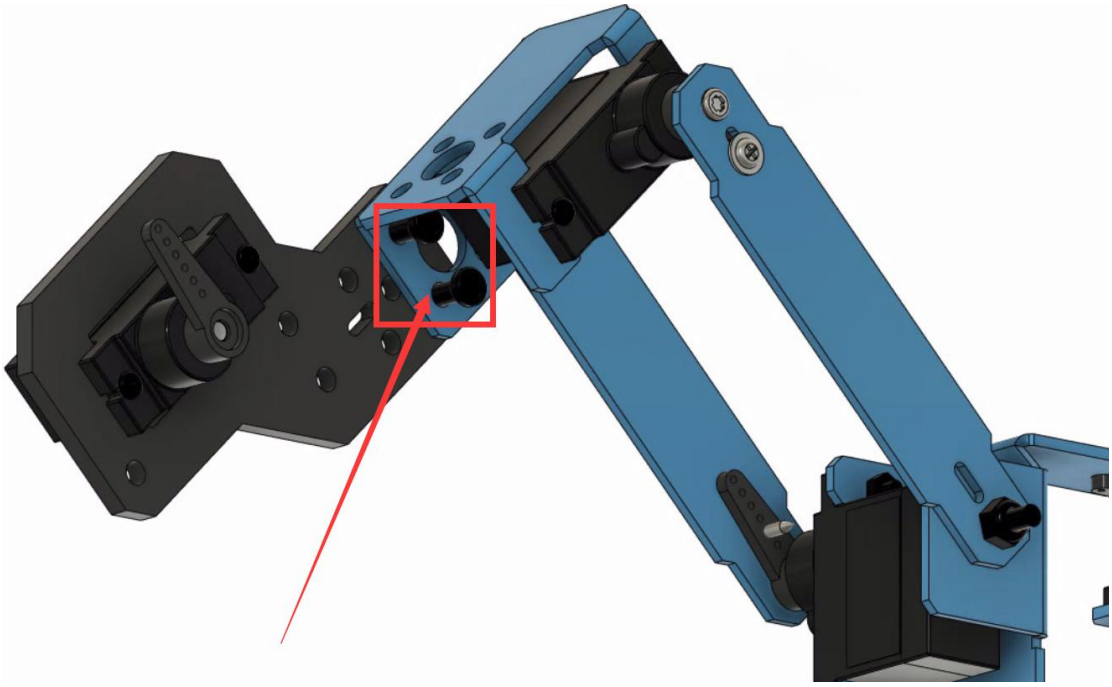
If you did not perform the "10.2" steps to complete the robotic arm assembly, then you are assembling this robotic arm for the first time. For the first part of the assembly tutorial, please refer to (1)—(21) Steps in this lesson "10.2 RaspArm-S Robotic Arm Assembly Tutorial". At step (20), the robotic arm is assembled into the following form:



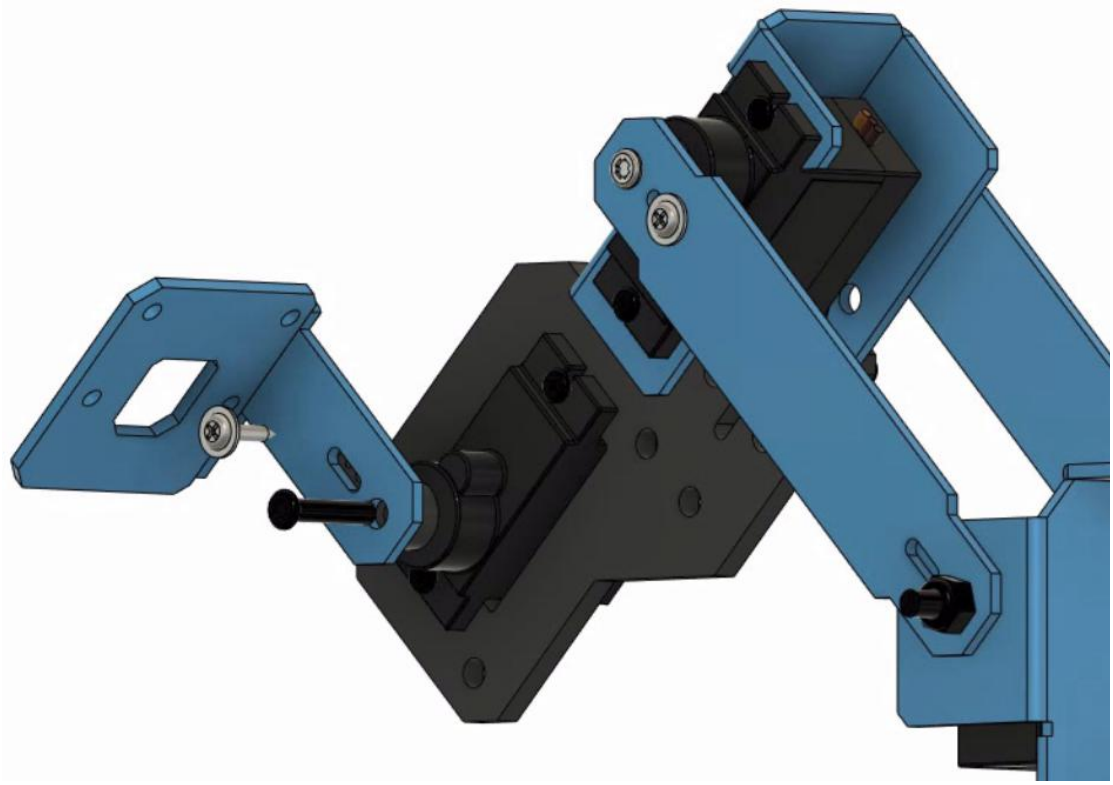
(22) You need to disassemble the A03 parts marked in the picture, as shown below:



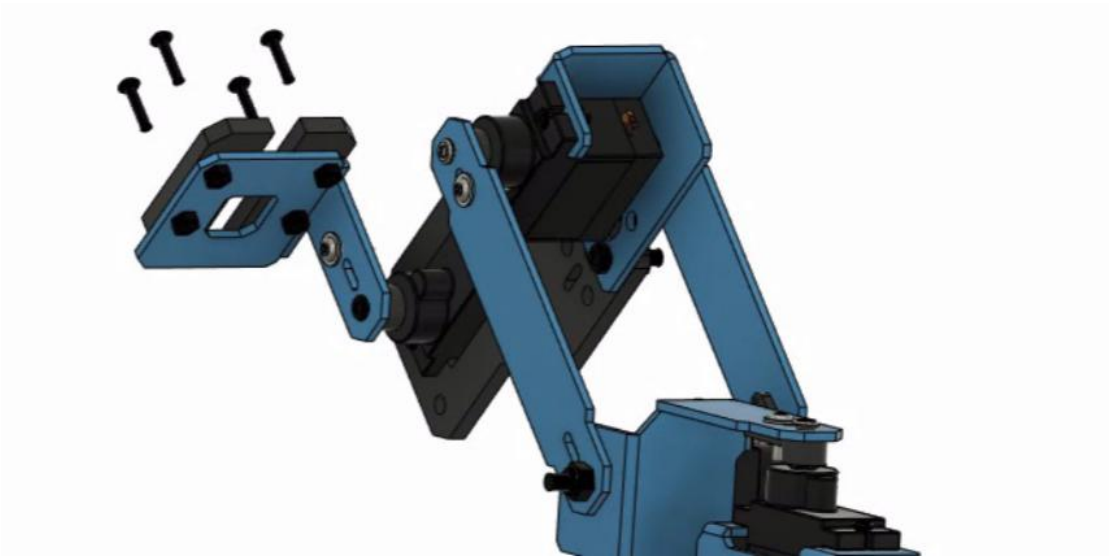
(22) Then use 2 M3x10 Screws and 2 M3 Nuts to install the disassembled A03 (including the servo) to the side of L04, as shown below:



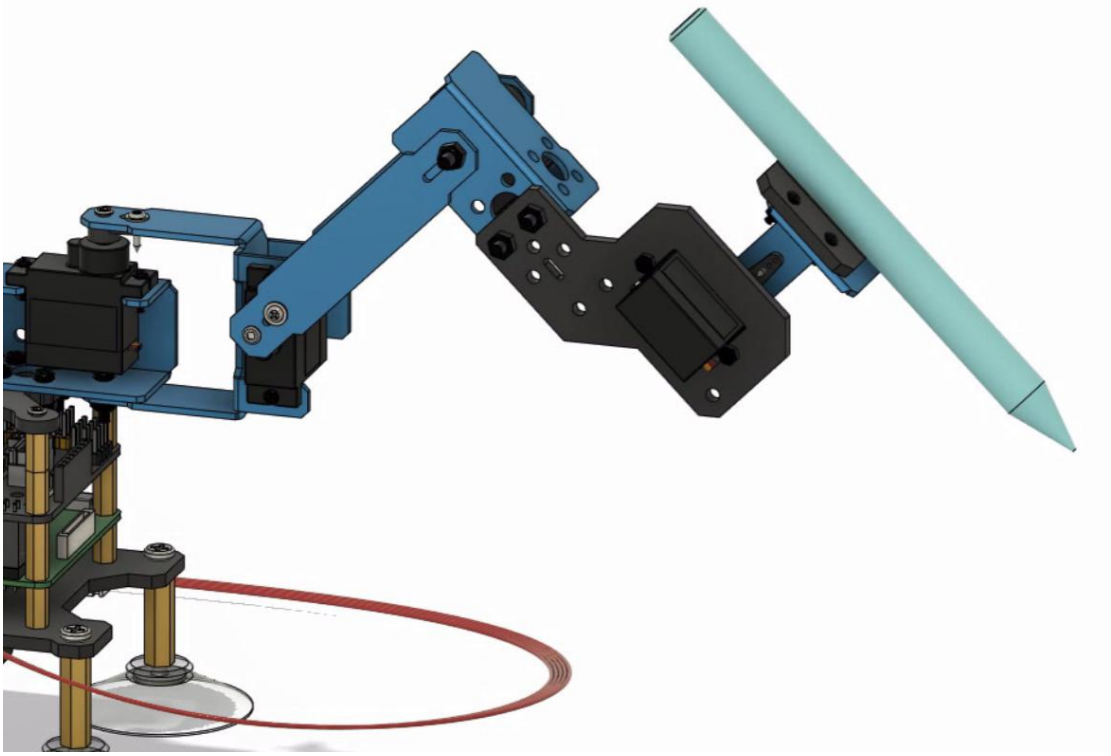
(23) Install the aluminum alloy part L01 on the D servo, and use 1 M2.5x8 Screw and 1 M1.4x6 self-tapping screw that comes with the servo to fix L01.



(24) Install 2 acrylic parts A08 on L01, pay attention to their placement direction, and fix them with 4 M2x8 Screws and 4 M2 nuts.



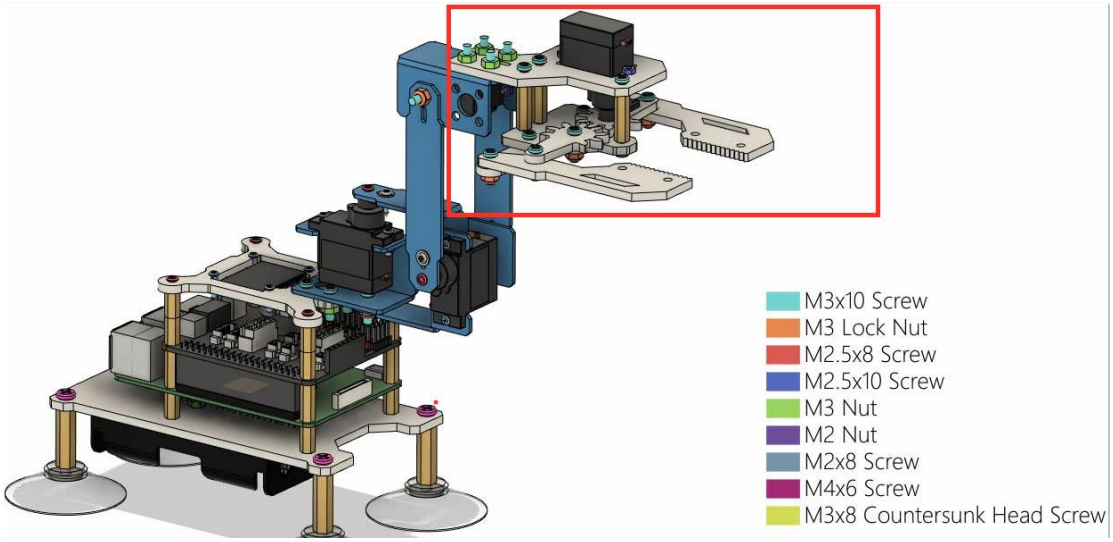
(25) After the final assembly is completed, you can use a rubber ring or rope to fix the pen. Then operate the writing pen robotic arm according to the following lesson case.



10.3.2 Basing on the robot arm to refit into a writing pen robot arm

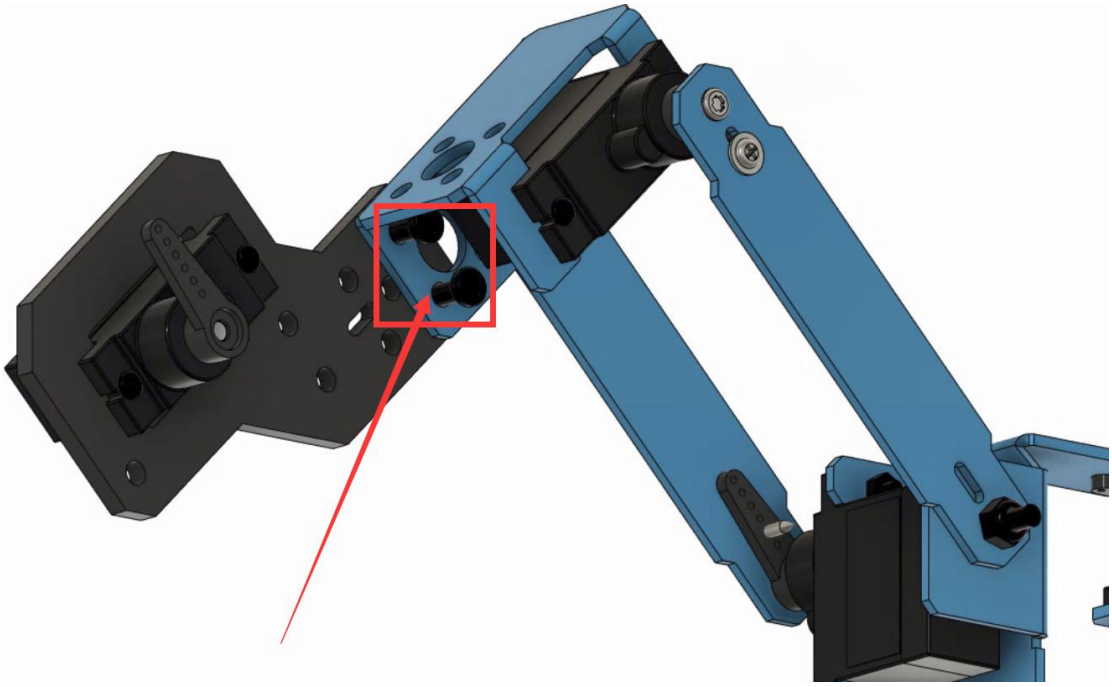
If you have installed the robotic arm according to the steps in "10.2" of this lesson, how do you convert it into a pen robotic arm? Let's modify it together.

(1) First, the chuck part of the robotic arm needs to be disassembled, as the part in the red circle in the picture below, when disassembling, you need to keep the A03 part (do not disassemble the servo).

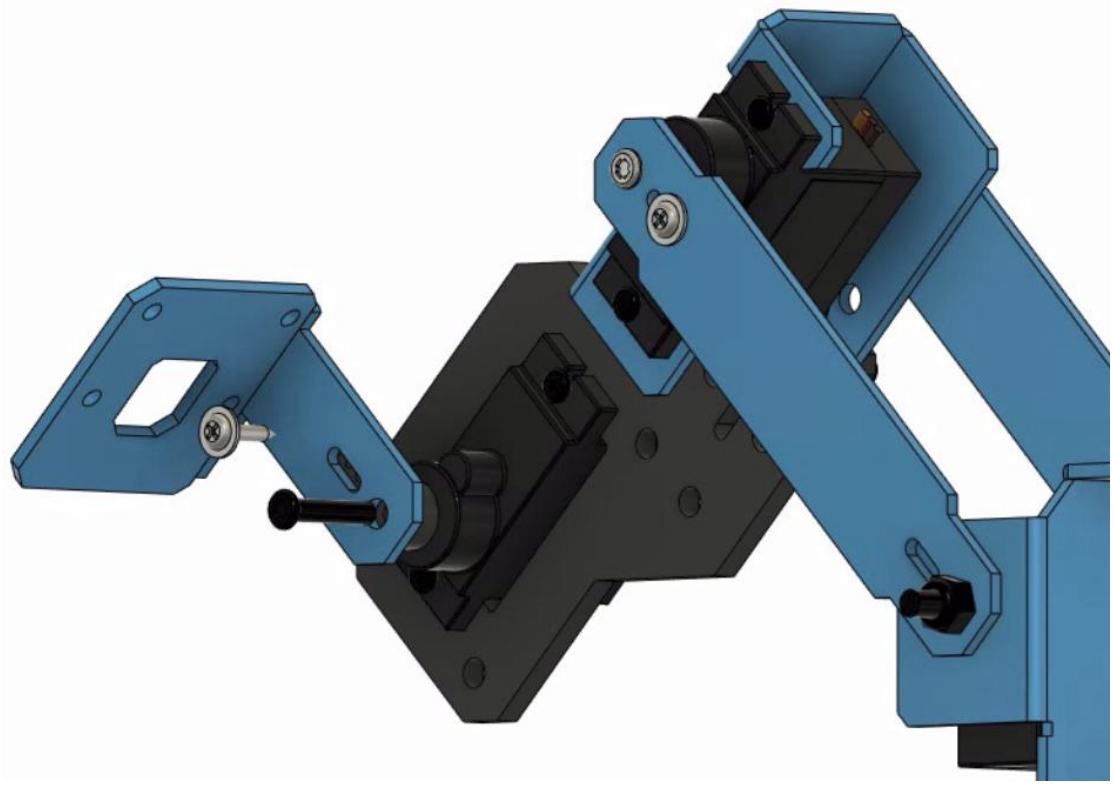


The structure part of A03 that has been disassembled is retained as shown below:

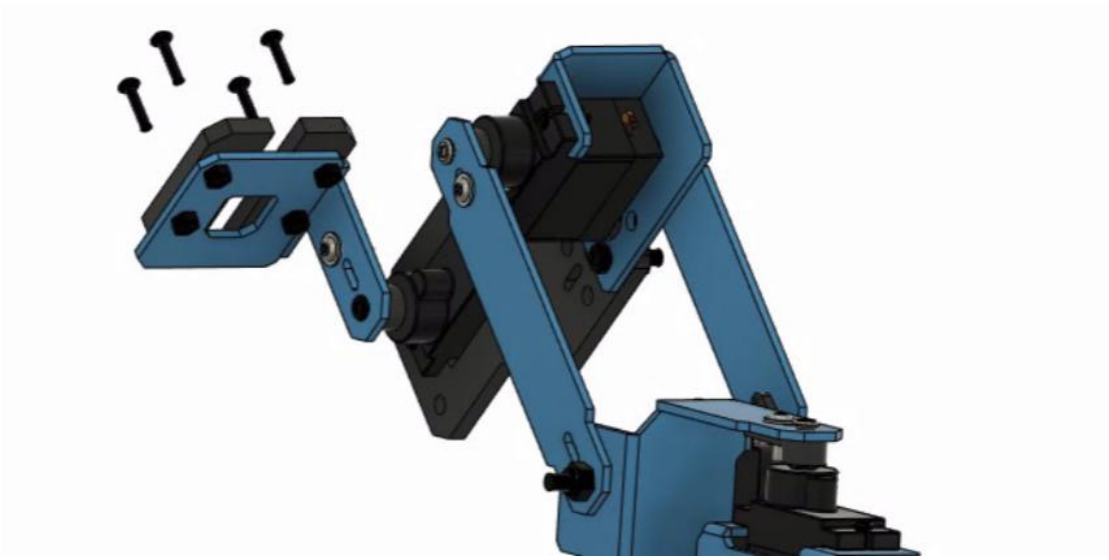
(2) Then use 2 M3x10 Screws and 2 M3 Nuts to install the disassembled A03 part (including the servo) to the side of L04, as shown in the following figure:



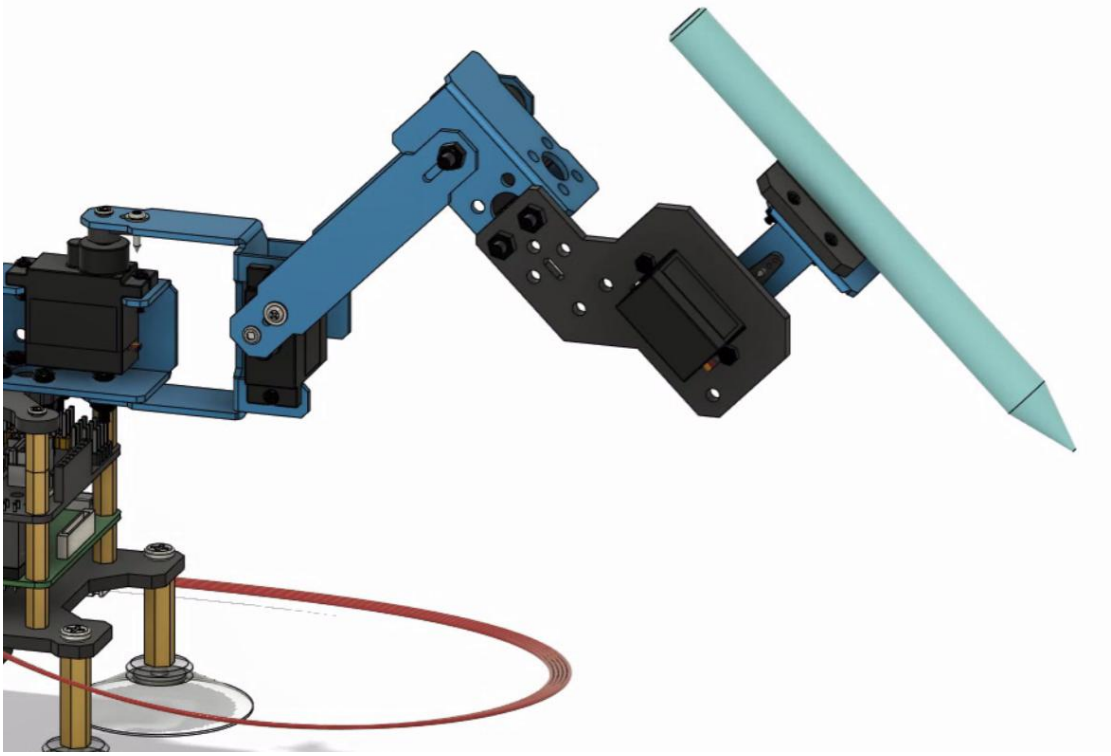
(3) Install the aluminum alloy part L01 on the D servo, and use 1 M2.5x8 Screw and 1 M1.4x6 self-tapping screw that comes with the servo to fix L01.



(4) Install the 2 acrylic parts A08 on L01, pay attention to their placement direction, and fix them with 4 M2x8 Screws and 4 M2 nuts.



(5) After the final assembly is completed, you can use a rubber ring or rope to fix the pen. Then operate the writing pen robotic arm according to the following lesson case.



(6) For the use case of the morphological structure of the robotic arm writing pen, please refer to the content of lesson 14 below.

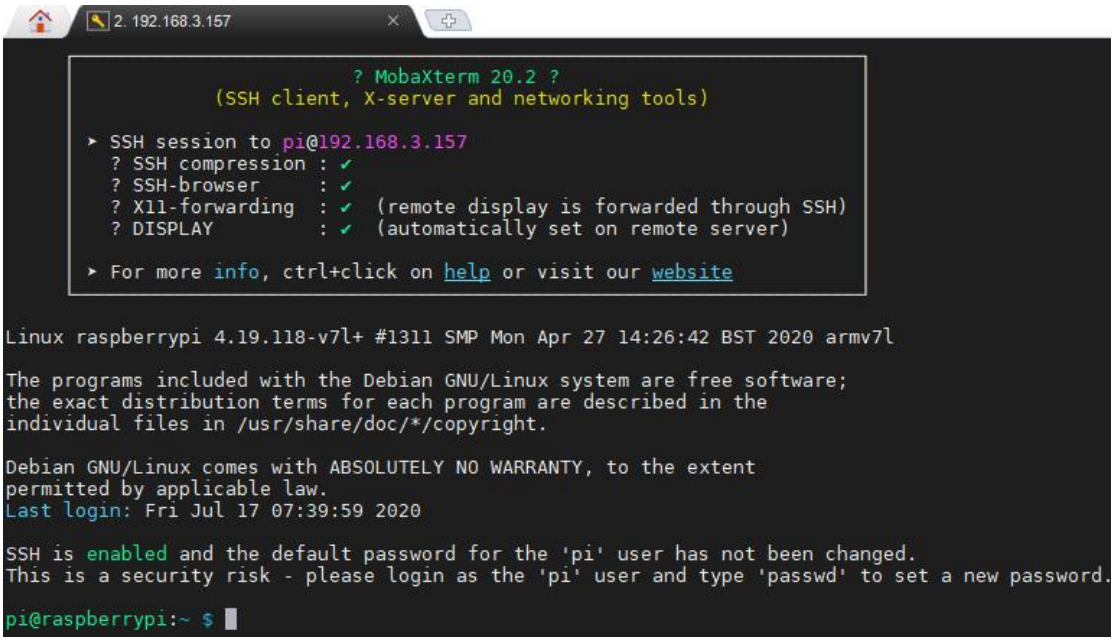
Lesson 11 Remotely Control of Robotic Arm with GUI

In this lesson, we will learn how to remotely control the robotic arm with the GUI.

11.1 How to open the GUI control interface

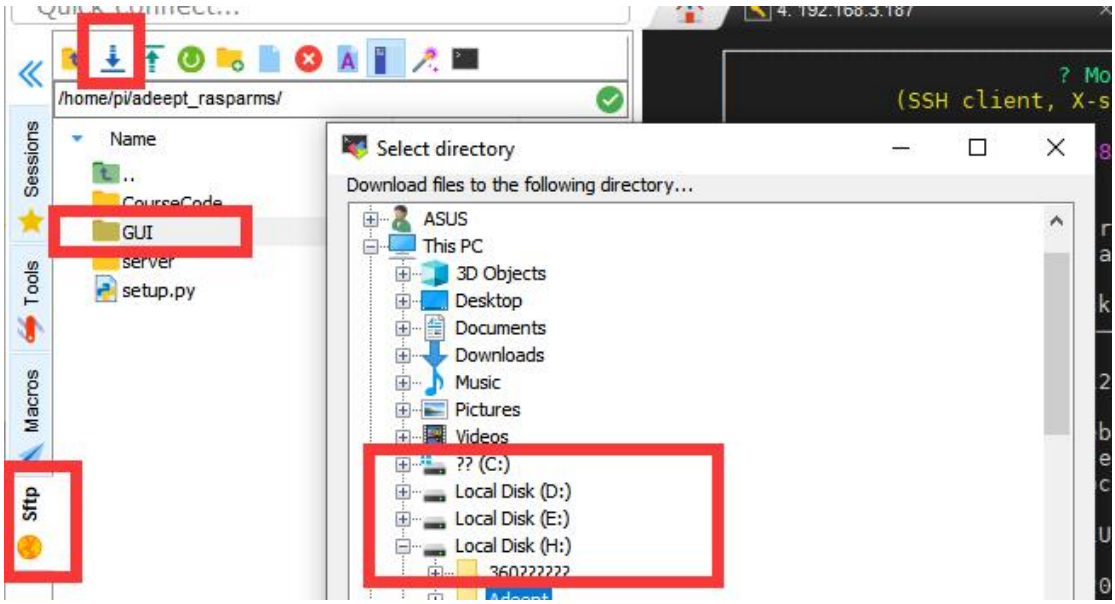
1. Open the terminal software MobaXterm:

2. Log in to your Raspberry Pi (the way to log in to Raspberry Pi has been introduced in Lesson 1):



3. The relevant code programs of the RaspArm-S robot are stored in the folder of

adeept_rasparms, which has been explained in "2.1 Downloading the Code Program for Controlling the Robot" in Lesson 2, you need to download the GUI folder to your computer under the directory of adeept_rasparms, click the GUI with the mouse, and then click the button to download the code program to the PC, and save it in the path of English letters, as shown in the figure below:



4. Then double-click on the PC to open the Python program GUI for Remote.py, you must make sure that the python environment has been installed on your PC. The GUI interface after opening is as follows:

【Pay attention】

If you cannot connect, I suggest you turn off the Raspberry Pi again, and then turn on the Raspberry Pi again.

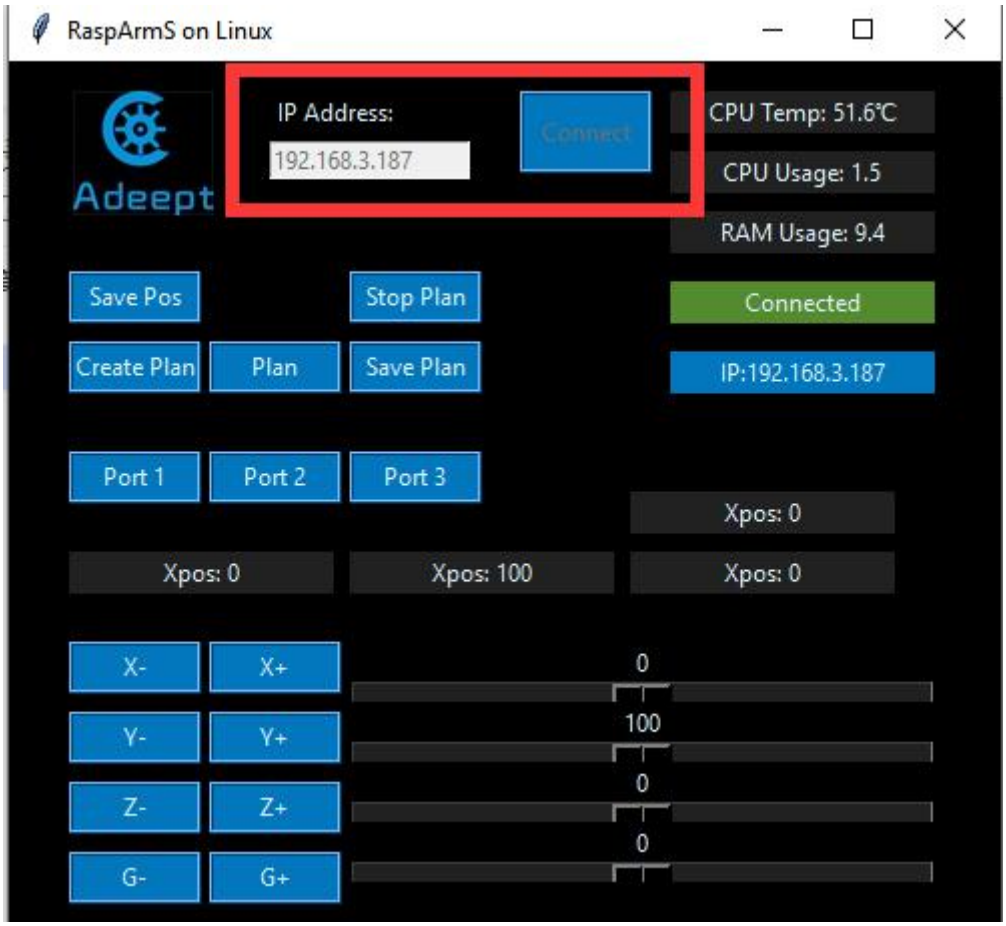
11.2 GUI function introduction

When studying this section, you only need to read and understand, you don't need to do any click operations. We will provide an operation case in section 11.3.

(1) 

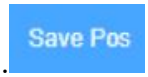
:

After opening the GUI control interface, you need to enter the IP address of the Raspberry Pi in the IP Address input field. Then click the Connect button to connect to the Raspberry Pi, after successful connection, as shown below:

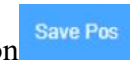


(2) 

:

1. 

:

Click the button 

, and the robot arm will record the coordinate position of the X, Y, Z, G axis of the current end point of the robot arm.

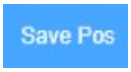
2. 

:

Click the button 

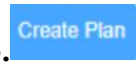
and the end point of the robotic arm will repeatedly move between the positions of the different coordinate points recorded in the last save.

When the end point of the robotic arm is at different coordinate points (more than

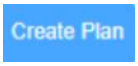
2 coordinate points), click the button 

, and the robotic arm will record the position information of the different coordinate points in the form of an array [X, Y, Z, G]. If you click the button 

, the end point of the robotic arm will repeatedly move between the recorded positions of different coordinate points.

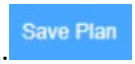
3. 

:

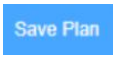
When you click the button 

, the previously saved coordinate point position information will be deleted, and you will enter the setting mode of re-recording the position of the end point of the robotic arm. You only need to click once. At this time, you can slide the X, Y, Z, and G sliders. To adjust the position of the end point of the robotic arm, each adjustment, combined with the button 

can save the coordinates of the end point of the robotic arm.

4. 

:

Click the button 

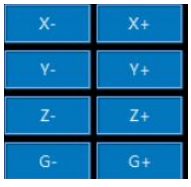
, the system will record the coordinate point position information of the end point of the robotic arm in the form of an array [X, Y, Z, G], and save it as a ".json" file, which will be saved in the course routine as a "plan.json" file.

5. 

:

Click the button 

and the robot arm will stop moving.

(3) 

1. [X+ and X-]: Controlling the end point of the robotic arm to move left and right

(1) X+:

If the x value becomes larger, the end point of the robotic arm will shift to the right. It

can be adjusted by the slider on the right.

(2) X-:

If the x value becomes smaller, the end point of the robotic arm is shifted to the left.

2. [Y+ and Y-]: Controlling the end point of the robotic arm to move back and forth

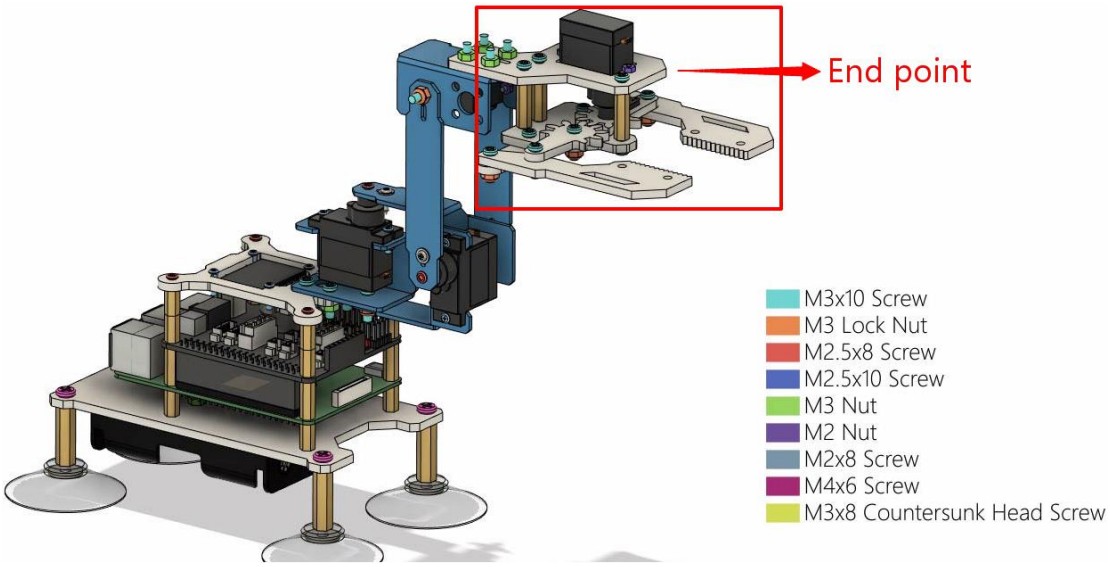
(1) Y+:

If the Y value becomes larger, the end point of the robotic arm moves forward. It can be adjusted by the slider on the right.

(2) Y-:

If the Y value becomes smaller, the end point of the robotic arm moves backward.

The area about the end point of the robotic arm is as follows:



3. [Z+ and Z-]: controlling the end point of the robotic arm to move up and down

(1) Z+:

If the Z value becomes larger, the end point of the robotic arm moves upward. It can be adjusted by the slider on the right.

(2) Z-:

If the Z value becomes smaller, the end point of the robotic arm moves downward.

4. [G+ and G-]: controlling the opening and closing of the mechanical arm clip

(1) G+:

If the G value becomes larger, the mechanical arm clamp is closed. It can be adjusted

by the slider on the right.

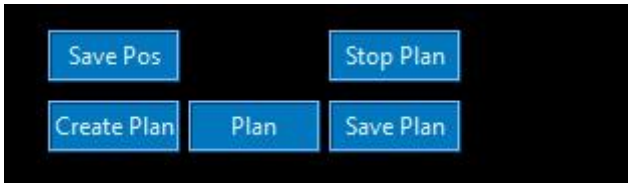
(2) G-:

If the G value becomes smaller, the mechanical arm clip opens.

11.3 Controlling the end point of the robotic arm to move between the new position points

When you open the GUI control application for the first time, click the button 

, and the robot arm will repeat the movement between the last recorded and saved position points (must be 2 different positions), then how to change the end point of the robotic arm to move at different points? Let's learn together.



First click the button 

to stop the moving robot arm, and then click the button 

to start recording a new position point. At this time, you need to combine the X, Y, Z, and G sliding buttons in the figure below to locate the end point of the robot arm to make adjustments, such as sliding the X slide bar to the far right, and turning the end point of the robotic arm to the right. At this time, we need to save the position of this coordinate point. By clicking the button 

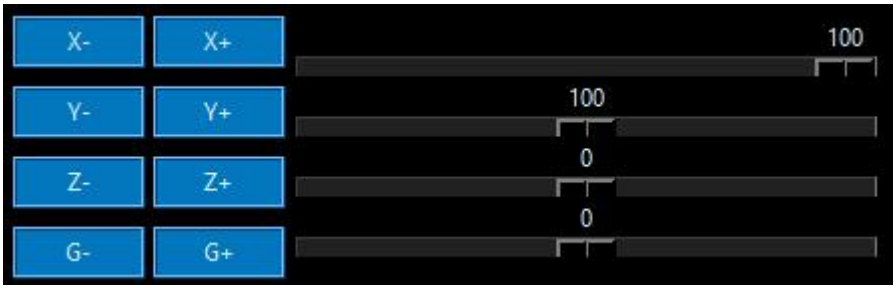
, we can save the current position; then we also need to record the new position again, continue to slide the X slide bar to the middle position, click the button 

at this time to save the current position; continue to slide the X slide bar to the leftmost position, click the button 

at this time, save the current position; now we have recorded three different position points, now click the button 

, the end point of the robotic arm will repeat the movement between the three different position points just recorded. If you want to continue running the location you just recorded

and saved when you turn on the Raspberry Pi next time, then you can record the location information by clicking the button Save Plan, and that's it.



Lesson 12 Controlling the Robotic Arm with Raspberry Pi Peripherals

In this lesson, we will learn how to control the movement of the robotic arm with the GUI when the Raspberry Pi is connected to an external monitor.

12.1 How to connect an external monitor to the Raspberry Pi

1. We provide a way to connect the Raspberry Pi to the display, you need to prepare the following components:

(1) 1 Type-C data cable: power supply for Raspberry Pi.

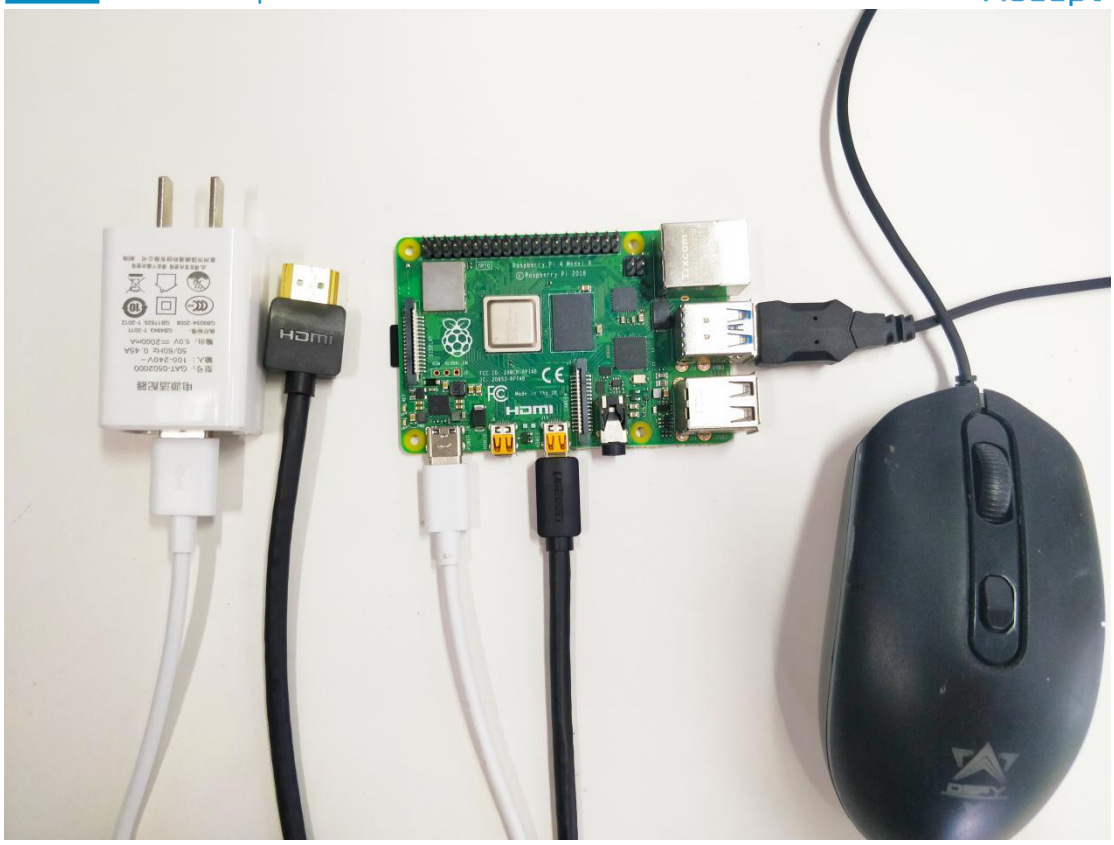
(2) 1 HDMI cable: used to connect the monitor.

(3) 1 mouse: used for operation.

(4) 1 display

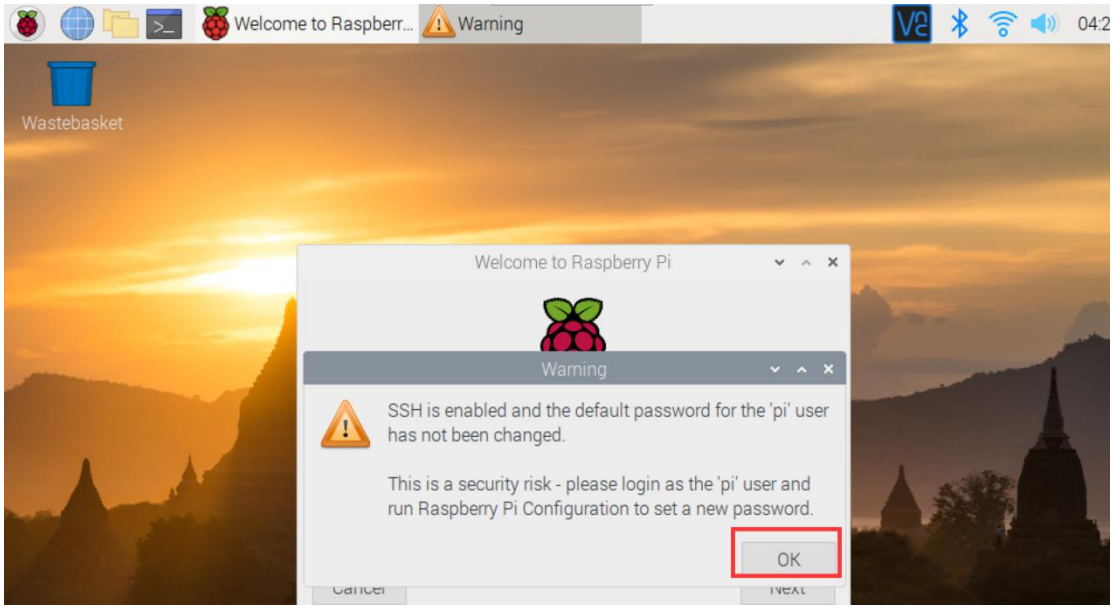
(5) 1 Raspberry Pi (The code file for controlling the robot must be downloaded in the Raspberry Pi according to the 2.1 course)

(6) 1 keyboard

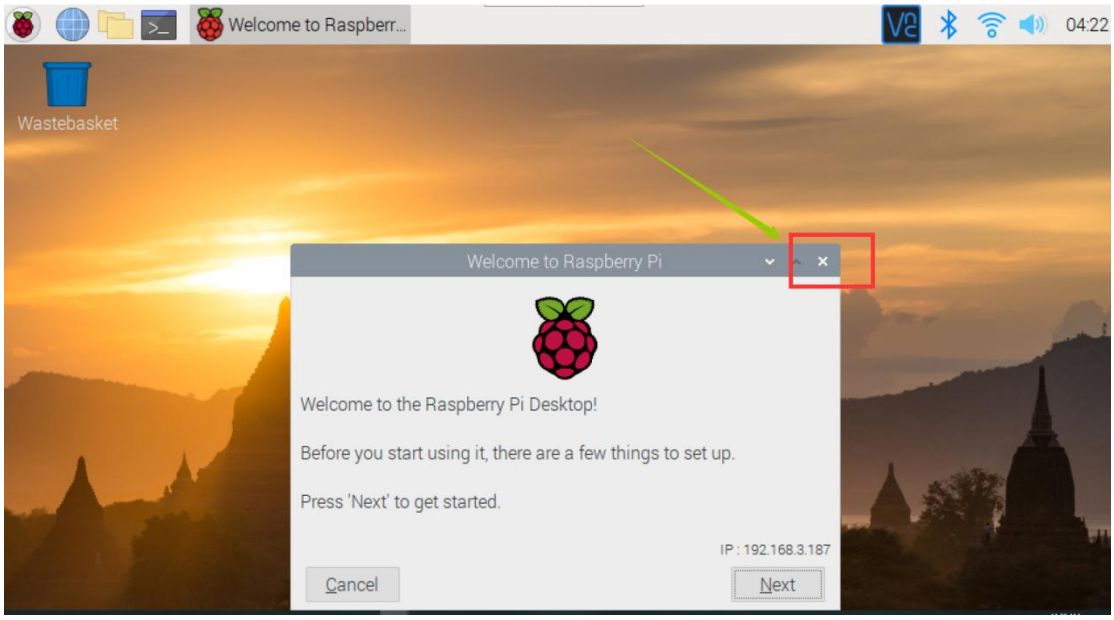


2. Connect the HDMI cable connected to the Raspberry Pi to the HDMI port of the display:

3. Turn on the display switch and supply power to the Raspberry Pi. You will see the startup interface of the Raspberry Pi, and wait for the Raspberry Pi to start successfully. Then click OK.



Click the X button in the upper right corner.



12.2 Controlling the robotic arm with GUI

1. After successfully connecting to the monitor and successfully starting the Raspberry Pi, you need to click the Raspberry Pi Terminal window on the upper left. As shown below:

2. Enter the following command in the opened Terminal window:

cd adeept_rasparms/server

ls

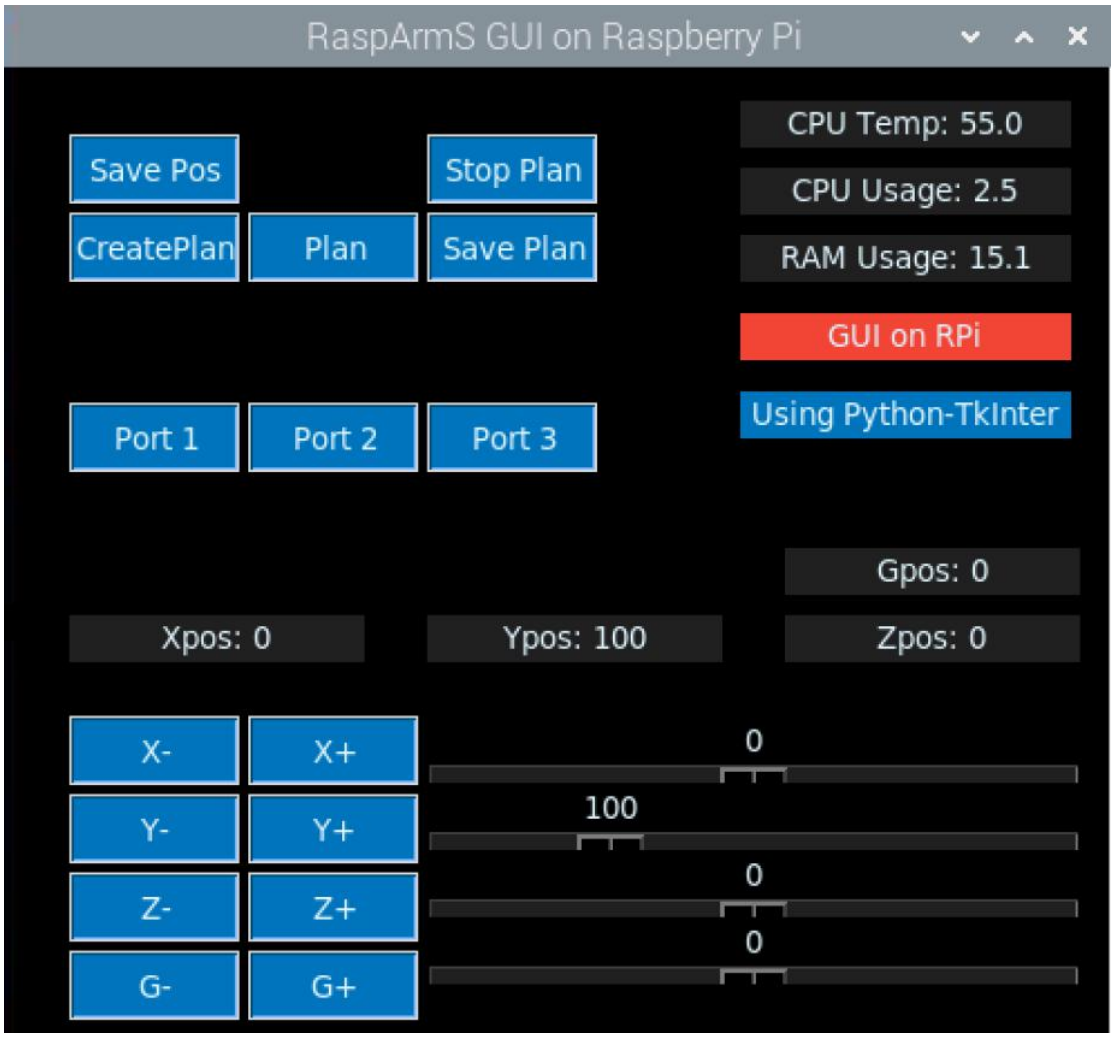
```
pi@raspberrypi:~ $ cd adeept_rasparms/server
pi@raspberrypi:~/adeept_rasparms/server $ ls
 bottle.py                index_py.bk   OLED.py          server.py
 bottle.pyc               info.py       plan.json        simpleCtrl.py
 config.json              info.pyc      __pycache__      smooth.py
'GUI on Raspberry Pi.py'  __init__.py   raspArmS.py      switch.py
 index.html               mutiTest.py   raspArmS.pyc     webServer.py
```

3. Enter the command to run the GUI application:

sudo python3 'GUI on Raspberry Pi.py'

```
pi@raspberrypi:~/adeept_rasparms/server $ sudo python3 'GUI on Raspberry Pi.py'
```

4.Then you will see the control interface of the GUI application. The introduction to the use of GUI applications has been introduced in "Lesson 11 Remotely Controlling Robot Arms by GUI". You can control the robotic arm with the corresponding buttons to realize your creativity.



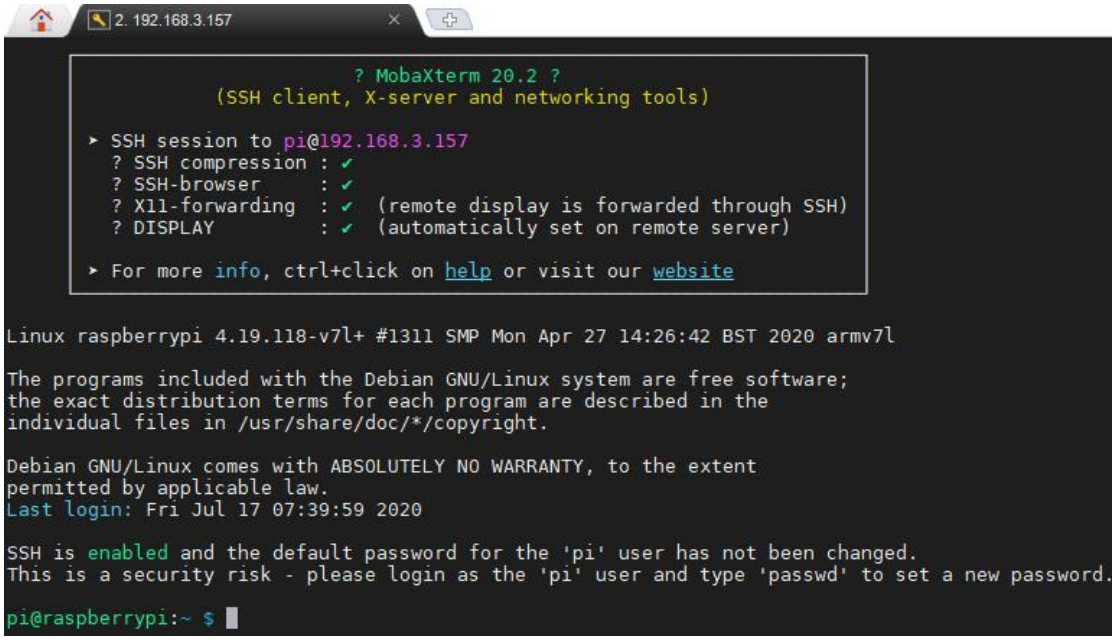
Lesson 13 Controlling the Robotic Arm with a Web Application

We have developed a WEB application to control the robotic arm to lower the barriers to use of the robotic arm. The WEB application is convenient to use. You can remotely control the robotic arm on a device with a browser (Google browser is recommended).

13.1 How to open the web application

1. Open the terminal software MobaXterm:

2. Log in to your Raspberry Pi (the way to log in to Raspberry Pi has been introduced in Lesson 1):

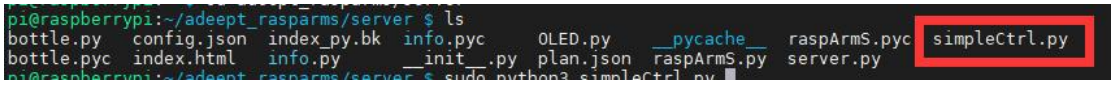


3. The relevant code programs of the RaspArm-S robot are stored in the folder of adeept_rasparms, which have been explained in "2.1 Downloading the code program to control the robot" in Lesson 2. First, you need to enter the server directory with the command window of the Raspberry Pi and enter the following command:

cd adeept_rasparms/server

4. Enter the command to display the contents of the current directory:

ls



5. Enter the command to run simpleCtrl.py:

sudo killall python3

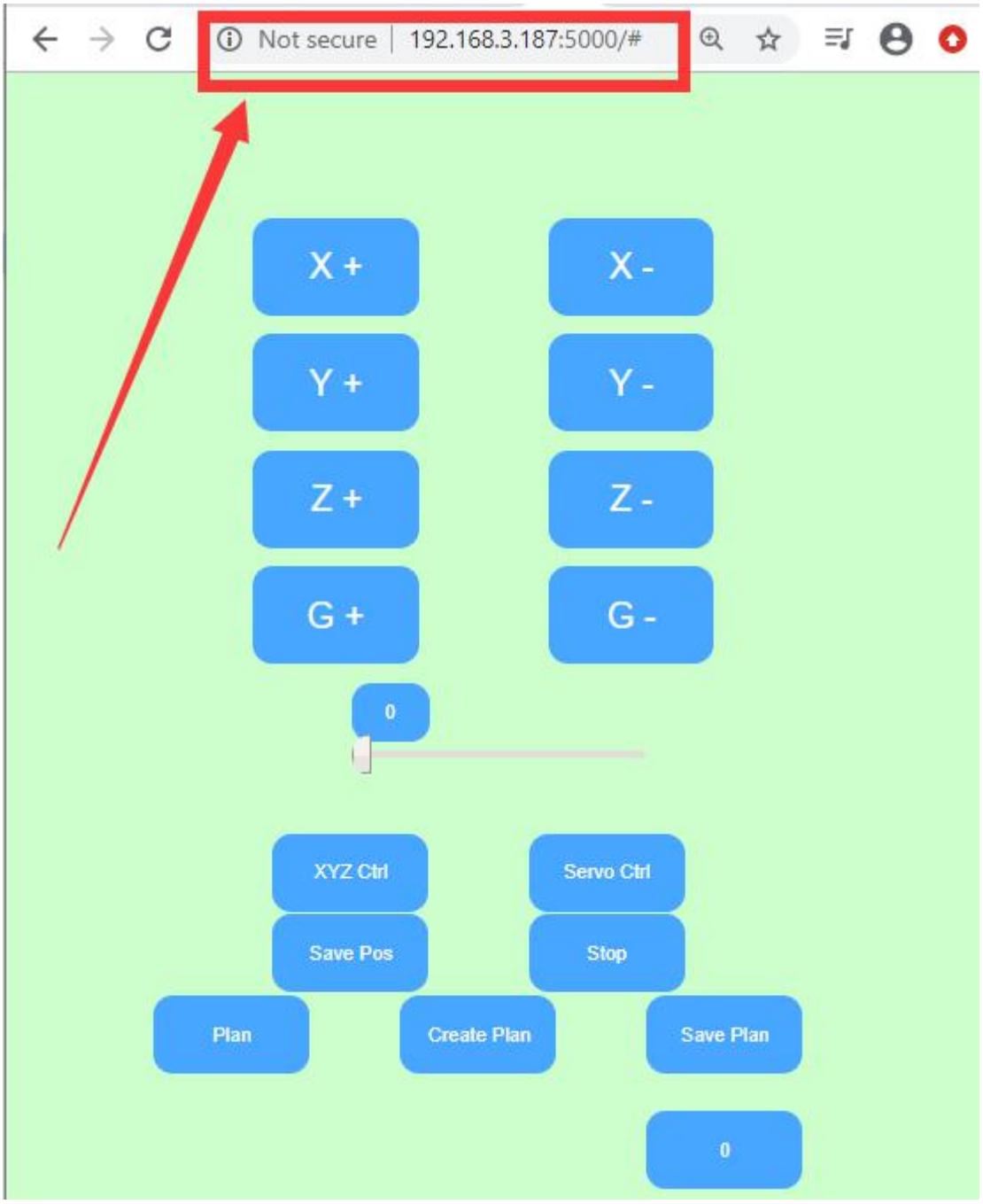
sudo python3 simpleCtrl.py

```
pi@raspberrypi:~/adeept_rasparms/server $ sudo python3 simpleCtrl.py
//home/pi/adeept_rasparms/server/plan.json
//home/pi/adeept_rasparms/server/config.json
[{'initPosA': 319, 'initPosB': 374, 'initPosC': 288, 'initPosD': 189}, {'servoN
servoNumD': 15}]
Import config [initPos]:
initPosA:319
initPosB:374
initPosC:288
initPosD:189
--- --- ---
Import config [servoPort]:
servoNumA:12
servoNumB:13
servoNumC:14
servoNumD:15
--- --- ---
//home/pi/adeept_rasparms/server/plan.json
192.168.3.187
waiting for connection...
Bottle v0.13-dev server starting up (using WSGIRefServer())...
Listening on http://0.0.0.0:5000/
Hit Ctrl-C to quit.
```

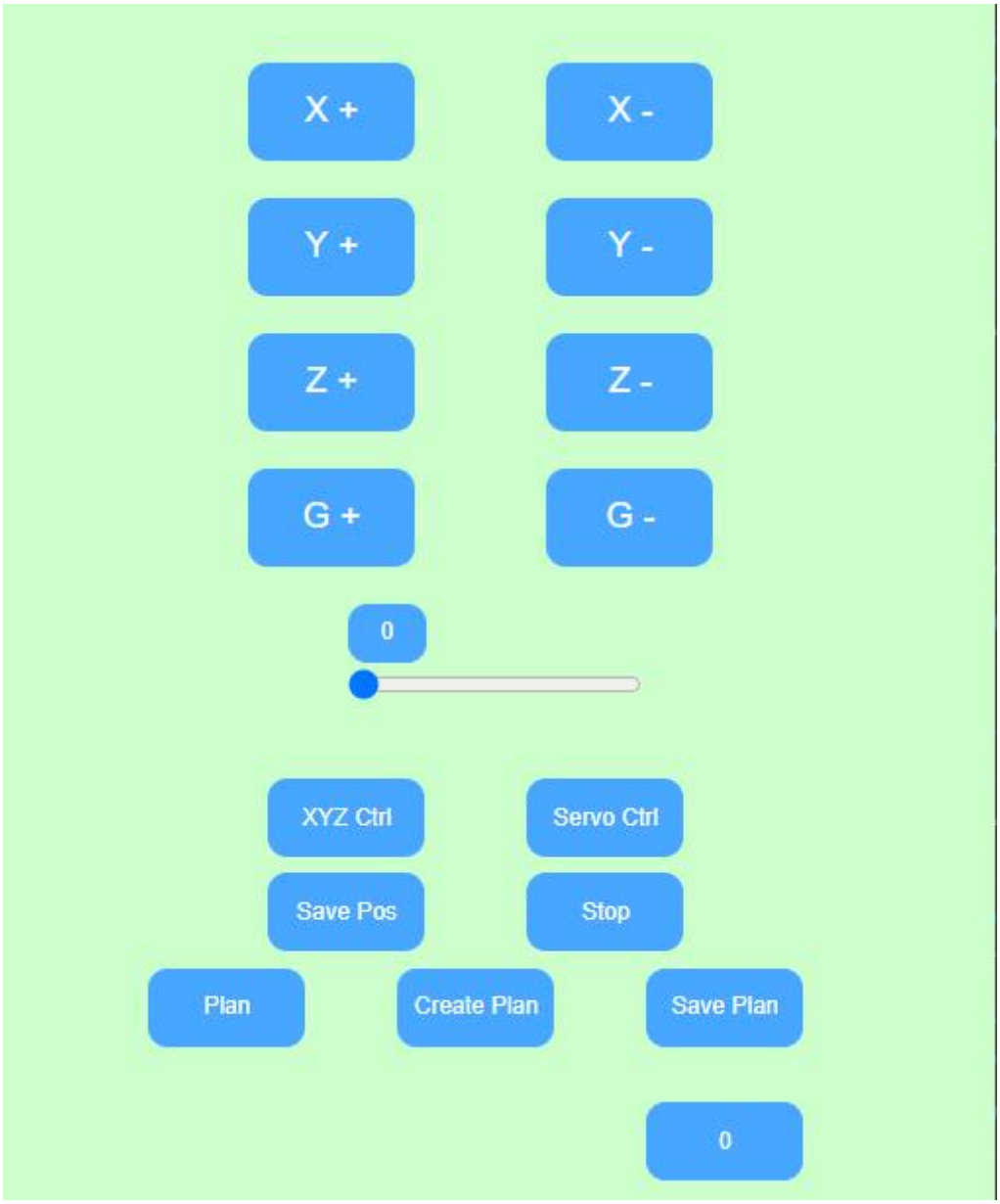
At this point, you need to open the Google browser (on your phone or computer) and enter the URL in the following format in the address bar: Raspberry Pi IP: 5000

For example, the IP address of my Raspberry Pi is 192.168.3.187, then the URL format in the address bar of the browser is: 192.168.3.187:5000

The interface after logging in to the web application is shown below:



13.2 Introduction to Web Application Functions



When learning the content of section 13.2, you only need to read and understand, without any click operations. We will provide an operation case in section 13.3.

The following is an introduction to the functions of each button.

13.2.1 [X+ and X-]: Controlling the end point of the robotic arm to move left and right

(1) X+:

If you click the X+ button, the x value becomes larger, and the end point of the

robotic arm shifts to the right.

(2) X-:

If you click the X- button, the x value becomes smaller and the end point of the robotic arm is shifted to the left.

13.2.2 [Y+ and Y-]: Controlling the end point of the robotic arm to move back and forth

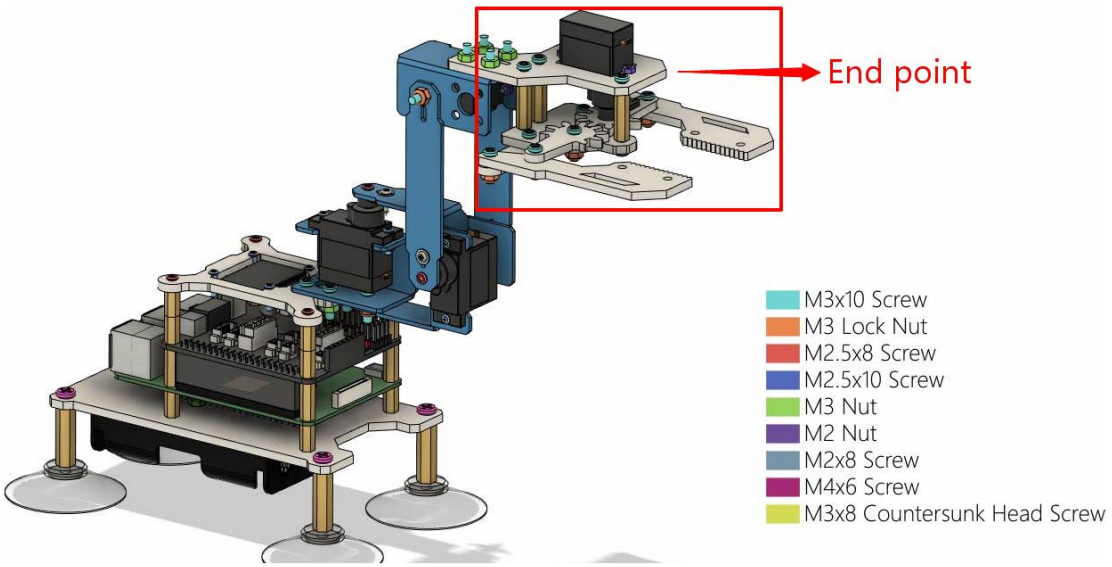
(1) Y+:

If you click the Y+ button, the Y value becomes larger and the end point of the robotic arm moves forward.

(2) Y-:

If you click the Y-button, the Y value becomes smaller and the end point of the robotic arm moves backward.

The position of the end point of the robotic arm is as follows:



13.2.3 [Z+ and Z-]: Controlling the end point of the robotic arm to move up and down

(1) Z+:

If you click the Z+ button, the Z value becomes larger, and the end point of the

robotic arm moves upward.

(2) Z-:

If you click the Z- button, the Z value becomes smaller and the end point of the robotic arm moves downward.

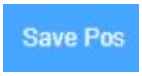
13.2.4 [G+ and G-]: Controlling the opening and closing of the mechanical arm clip

(1) G+:

If you click the G+ button, the G value becomes larger and the mechanical arm clip opens.

(2) G-:

If you click the G- button, the G value becomes smaller, and the mechanical arm clamp closes.

13.2.5 

Click the button 

, and the robot arm will record the coordinate position of the X, Y, Z, G axis of the current end point of the robot arm.

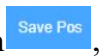
13.2.6 

Click the button 

, the end point of the robotic arm will repeatedly move between the recorded positions of different coordinate points.

Click the button 

directly, and the robotic arm will move back and forth repeatedly between the positions where the coordinate points are saved in the "plan.json" file by default.

When the end points of the robotic arm are at different coordinate points (more than 2 coordinate points), click the button 

, and the robotic arm will record the position information of the different coordinate points in the form of an array [X, Y, Z, G]. If you click the button 

, the end point of the robotic arm will repeatedly

move between the recorded positions of different coordinate points.

13.2.7 Create Plan

When you click the button Create Plan, the previously saved coordinate point position information will be deleted. At this time, you can combine the button Save Pos to recreate and record the new coordinate position information.

13.2.8 Save Plan

When you click the button Create Plan, the previously saved coordinate point position information will be deleted, and you will enter the setting mode of re-recording the position of the end point of the robotic arm. You only need to click it once. At this time, you can click the X, Y, Z, and G buttons. Adjust the position of the end point of the robotic arm. Each time you adjust, combine the button Save Pos to save the coordinates of the end point of the robotic arm.

13.2.9 Stop

Click the button Stop and the robot arm will stop moving.

13.3 Controlling the end point of the robotic arm to move between the new position points

When you open the web control application for the first time, click the button Plan, and the robot arm will repeat the movement between the last recorded and saved position points (must be 2 different positions), then how to change the end point of the robotic arm to move at different points? Let's learn together.

First click the button 

to stop the moving robot arm, and then click the button 

to start recording a new position point. At this time, you need to combine the X, Y, Z, and G buttons to adjust the position of the end point of the robot arm, such as by clicking the button 

, the end point of the robotic arm rotates to the right to a position to stop. At this point, we need to save the position of this coordinate point. By clicking the button 

, we can save the current position; then we need to record the new position again, continue to slide the X slider to the middle position, and click the button 

to save the current position to record; now we have recorded 2 different position points, now click the button 

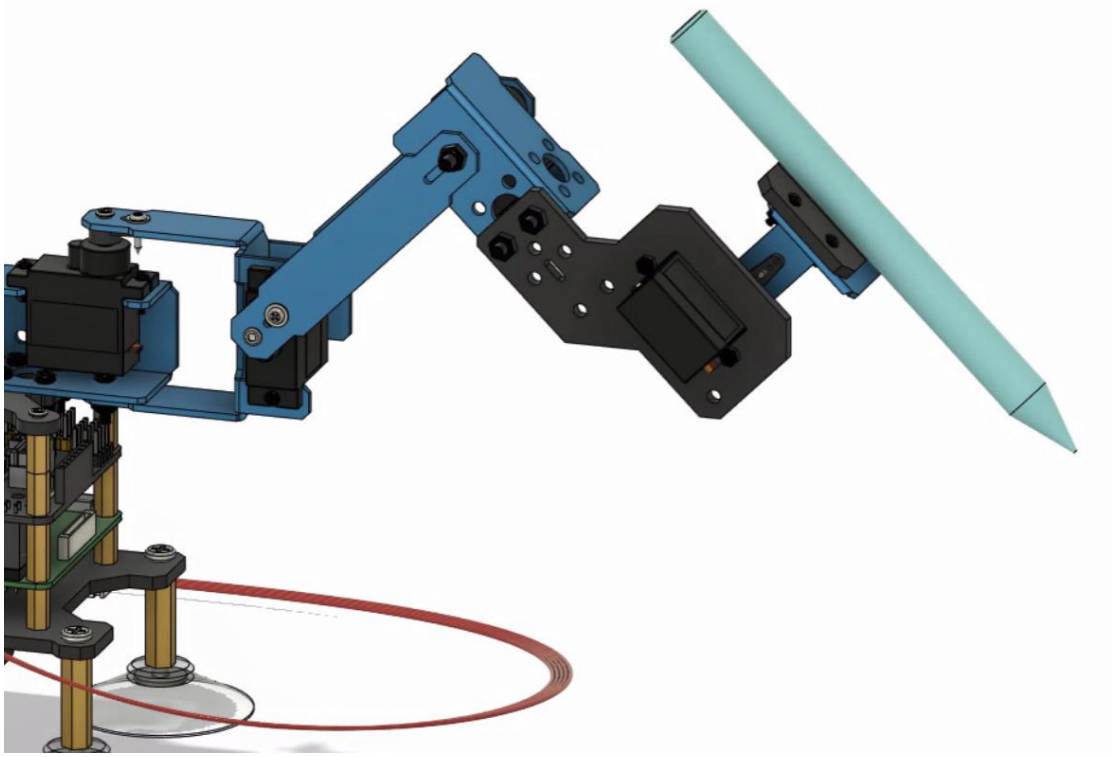
, the end point of the robotic arm will repeat movement between the 2 different positions just recorded. If you want to continue running the location you just recorded and saved when you turn on the Raspberry Pi next time, you can record the location information by clicking the button 

, and that's it.

Lesson 14 Replacing the Arm Clip with a Writing Pen

In this lesson, we will learn another form of the RaspArm-S robotic arm, which is to replace the clip of the robotic arm with a writing pen. With this new structure, you can use it for some creations, such as drawing. The specific assembly steps are in "10.3 RaspArm-S Writing Pen Robotic Arm Assembly Tutorial" in Lesson 10.

The assembled structure of RaspArm-S in writing pen form is as follows:

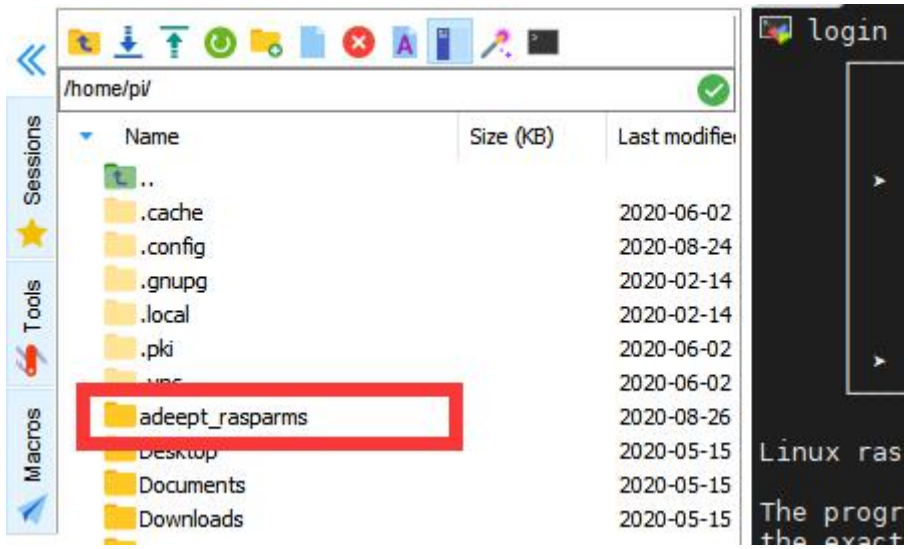


14.1 Modifying raspArmS.py to pen mode

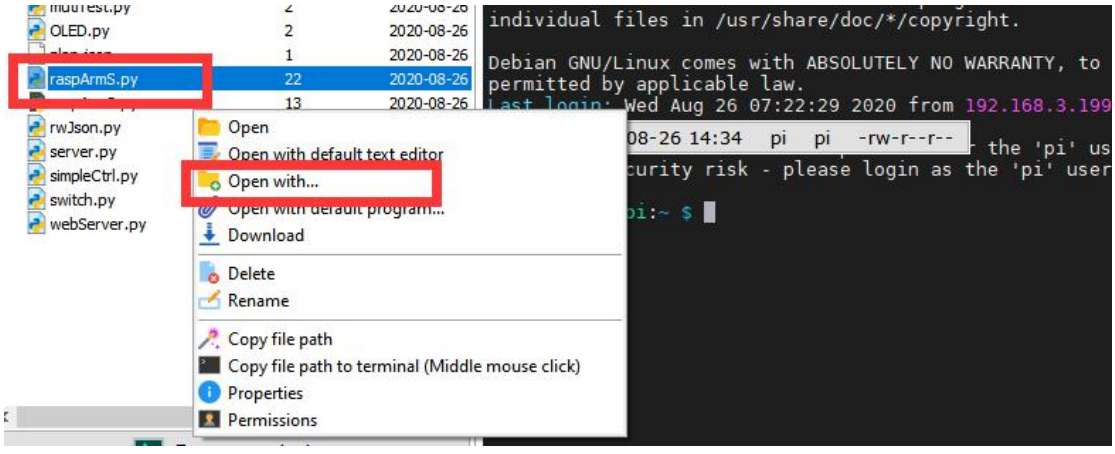
You need to modify the raspArmS.py program when using the pen shape structure.

1. Open the terminal software MobaXterm:

2. Log in to your Raspberry Pi (the way to log in to the Raspberry Pi has been introduced in Lesson 1), in the file explorer on the left, go to the directory of adeept_rasparms/server:



3. Select the program raspArmS.py, right-click the mouse, and use the third-party tool Subline IDE to view and edit the code program. For the specific method, please see the content of Lesson 2: "2.4 Editing the code program in the Raspberry Pi".



4. Find line 257, modify gripper to pen, and save the modified file. This

completes the setting.

14.2 How to control a robotic arm in the form of a writing pen

Most of the functions related to robotic arm control are encapsulated in raspArmS.py, which is an open API. First, you need to create a new .py file in the same folder as raspArmS.py, then you can write code programs to realize your own creativity in this newly created .py folder. The usage instructions of the API function raspArmS.py are explained in detail in Lesson 15.

If you just want to simply experience the functions of this form, you can also operate it by opening the GUI application. You can refer to Lesson 11.

Lesson 15 API Instructions

Most of the functions related to robotic arm control are encapsulated in raspArmS.py. This lesson introduces the calling methods of these API, which is convenient for users to carry out secondary development.

First, you need to create a new .py file and place it in the same folder as raspArmS.py. The code described below is written in the newly created .py file.

15.1 How to call API functions in the project

Import raspArms,the object that instantiates and controls the robotic arm is ras, call the start() function in raspArmS.py with ras, you don't need to input any parameters, call the start() function to control the motion of the robotic arm.

```
import raspArmS
import time

ras = raspArmS.RaspArmS()
ras.start()
```

15.2 xyzInput([x, y, z])

【effect】:

xyzInput([x, y, z]) can control the end point of the robotic arm to move to a certain point (the end point of the robotic arm is roughly the point where the chuck clamp is located).

【Call example】:

```
ras = raspArmS.RaspArmS()
```

```
ras.xyzInput([x, y, z])
```

【Parameter Description】:

1. [x, y, z] is the coordinate point of the Cartesian coordinate system where the end point is located.

2. When the x value is 0, the end point of the robotic arm is the point directly in front of it, and the x value becomes larger, and the end point shifts to the right; the x value becomes smaller, the end point shifts to the left.

3. The y value is the position of the end point relative to the front and back of the robot arm: if the y value becomes larger, the end point position will shift forward; if the y value becomes smaller, the end point position will shift backward

4. The z value is the position of the end point relative to the upper and lower positions of the robotic arm: if the z value becomes larger, the end point position will shift upward; if the z value becomes smaller, the end point position will shift downward.

15.3 servoAngInput(angleInput)

【effect】:

Controlling each servo of the robotic arm to rotate a certain angle.

【Call example】:

```
ras = raspArmS.RaspArmS()
```

```
ras.servoAngInput(angleInput)
```

【Parameter Description】:

The angleInput parameter is an array of four servo angles.

For example, when the parameter angleInput is [10, 20, 30, 40]:

It means to control A servo to rotate 10°, B servo to rotate 20°, C servo to rotate 30°, and D servo to rotate 40° (all the above degrees are relative to the initial position of the servo).

【pay attention】

If the parameter angleInput is [0, 0, 0, 0], it means to control all the servos to

rotate to the initial position.

15.4 servoInitSet()

【effect】:

Setting the neutral position of the servo.

【Call example】:

```
ras = raspArmS.RaspArmS()
```

```
ras.servoInitSet()
```

【Parameter Description】:

There is no need to write parameters when calling servoInitSet(), but you need to enter commands in the console to adjust the position of the servo. Refer to the table below:

Input the command	Operating	Description
w	Enter	Increase the PWM of the corresponding servo by 1
s	Enter	The PWM of the corresponding servo is reduced by 1
q	Enter	Select the next servo
a	Enter	Select the last servo
x	Enter	Confirm and save settings
c	Enter	Cancel settings and exit (will not save)

15.5 changeMode(' ')

【effect】:

Changing the working mode.

【Call example】:

```
ras = raspArmS.RaspArmS()
```

```
ras = raspArmS.RaspArmS()
```

```
ras.changeMode('gripper') #Set to chuck mode
ras.changeMode('pen')     #Set to pen mode
```

【Parameter Description】:

Special note: The working mode selection needs to cooperate with the corresponding mechanical structure assembly form to operate normally. There are two structural forms of the robotic arm, one is a "mechanical arm" that can grip objects, and the other is a "writing pen robotic arm" that can write and draw. For details, please refer to "Lesson 10 RaspArm-S Assembly Tutorial".

1. In the writing pen mode, Rudder D will try its best to automatically maintain the aluminum alloy parts that connects to it and maintain an angle with the ground.

2. In the chuck mode, D servo is used to control the opening and closing of the chuck.

15.6 changeSpeed(speedInput)

【effect】:

Changing the speed of the servo.

【Call example】:

```
ras = raspArmS.RaspArmS()
```

```
ras.changeSpeed(speedInput)
```

【Parameter Description】:

1. The range of the parameter speedInput: 0 <speedInput <= 100.

2. The default value is 100.

3. This setting affects the running speed of the moveXYZ (newPos) function of 15.7

15.7 moveXYZ(newPos)

【effect】:

Controlling the robotic arm to slowly move to a new position.

【Call example】:

```
ras = raspArmS.RaspArmS()
```

```
ras.moveXYZ(newPos)
```

【Parameter Description】:

1. The parameter newPos is an array of new positions, such as [0, 0, 0, 0].

2. This function will cause the program to block, and it will jump out when ras.globalCommand =='stop'.

15.8 gripper(command)

【effect】:

Controlling the chuck of the robotic arm.

【Call example】:

```
ras = raspArmS.RaspArmS()
```

```
ras.gripper(command)
```

【Parameter Description】:

1. When the parameter command is "catch", the mechanical arm chuck is clamped.

2. When the parameter command is "loose", the mechanical arm chuck is released.

3. When the parameter command value is between 0-90, the chuck of the mechanical arm forms a certain angle, 90 is clamping, and 0 is loosening.

15.9 planGoes(planList)

【effect】:

The robotic arm automatically repeats the scheduled task.

【Call example】:

```
ras = raspArmS.RaspArmS()
```

```
ras.planGoes(planList)
```

【Parameter Description】:

1. The parameter planList is an array composed of each target point position, for example: [[90, 140, 0, 90], [-90, 140, 0, 0], [0, 140, 90, 0]]

2. This function will cause the program to block, and it will jump out when ras.globalCommand =='stop'.

15.10 simpleMoveStart(axis, direction)

【Effect】:

Controlling the simple movement of the robotic arm. (If the robotic arm is converted into a PTZ, you can use this function to control the movement of the PTZ in real time)

【Call example】:

```
ras = raspArmS.RaspArmS()
```

```
ras.simpleMoveStart(axis, direction)
```

【Parameter Description】:

1. The parameter axis is used to select the desired motion axis, which can be: 'X', 'Y', and 'Z'.

2. The parameter direction is used to select the control direction, which can be: '+' or '-'.

3. This function will not block the program, and the robot arm will start to move after the call is completed.

【Pay attention】:

```
ras.simpleMoveStart(axis, 'stop')
```

1. The parameter axis is used to select the motion axis that you want to stop. It

can be: ‘X’, ‘Y’, and ‘Z’.

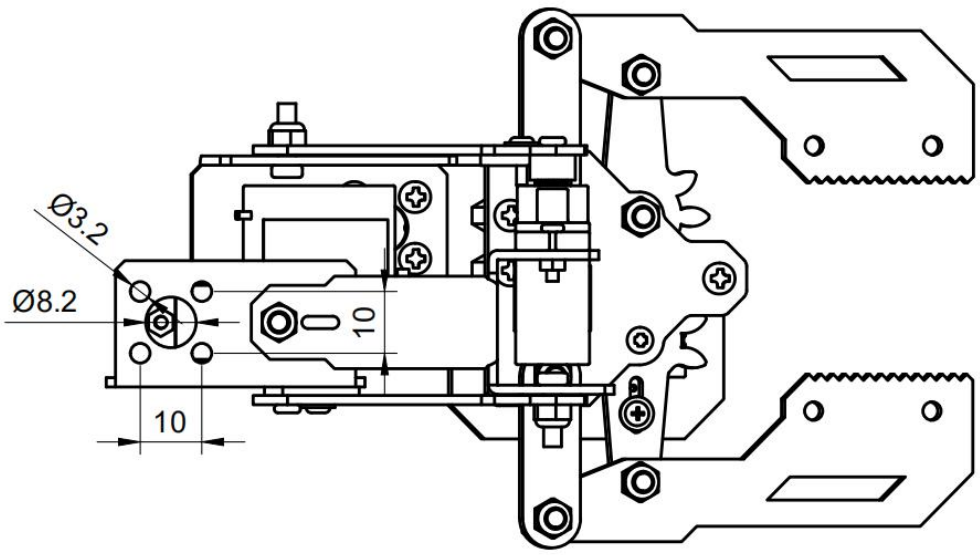
2. After an axis starts moving, this function needs to be called to stop the axis.

Lesson 16 Mounting the RaspArm-S Robotic Arm to Other Robots

You can use the RaspArm-S manipulator as an extension module of other robots and install it on the robot. For example, you can install it on the Alter Dog product developed and designed by us:

We open sourced the key assembly dimension parameters of this robotic arm, as shown in "Dimensional Drawing.pdf".

This assembly size is also a commonly used size for the steering wheel of a digital servo. You can expand it appropriately according to your needs.



The robot arm can be fixed directly with M3x10 screws and M3 nuts, or it can be fixed with copper pillars.

After the robotic arm is installed, please refer to the content of lesson 15 for the related API call method of controlling the robotic arm. If you change the servo connection port, you need to modify the course sample code config.json provided by us. The method of modifying the code program in the Raspberry Pi is as follows:to view and edit our code program on the MobaXterm terminal software, we provide two methods for reference.

(1) The first is to use Linux commands. For example, if you need to view and modify the code program config.json of this lesson, then you can log in to the Raspberry Pi with the MobaXterm terminal and enter the following command in the command window (Raspberry Pi console) :

sudo nano adeept_rasparms/server/config.json

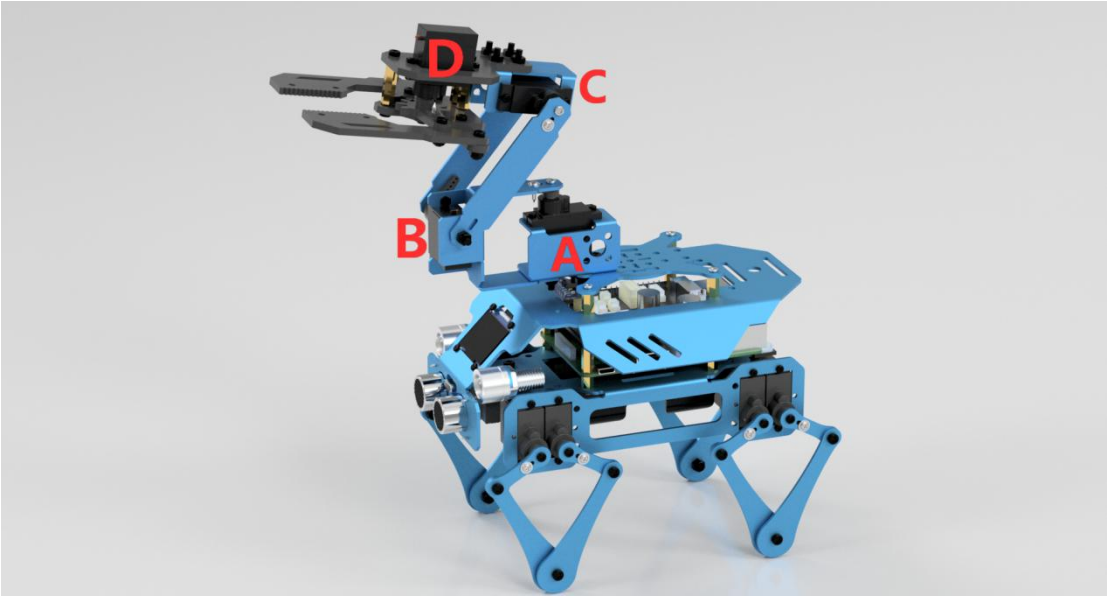
```
pi@raspberrypi:~ $ sudo nano adeept_rasparms/server/config.json
```

After opening the code file, you need to find this line of code:

```
{"servoNumA": 12, "servoNumB": 13, "servoNumC": 14, "servoNumD": 15}
```

Among them, 12, 13, 14, and 15 are the default servo interface numbers of our course examples. You need to modify them according to your actual situation. The

structure diagram of the servo of the robotic arm from bottom to top is as follows:



In this way, you can view and modify our code program. You can modify and edit the code with the direction keys on the keyboard to realize the control function you want. But you need to learn and search about the command operation method of Linux nano.

[Nano commonly used operation commands]:

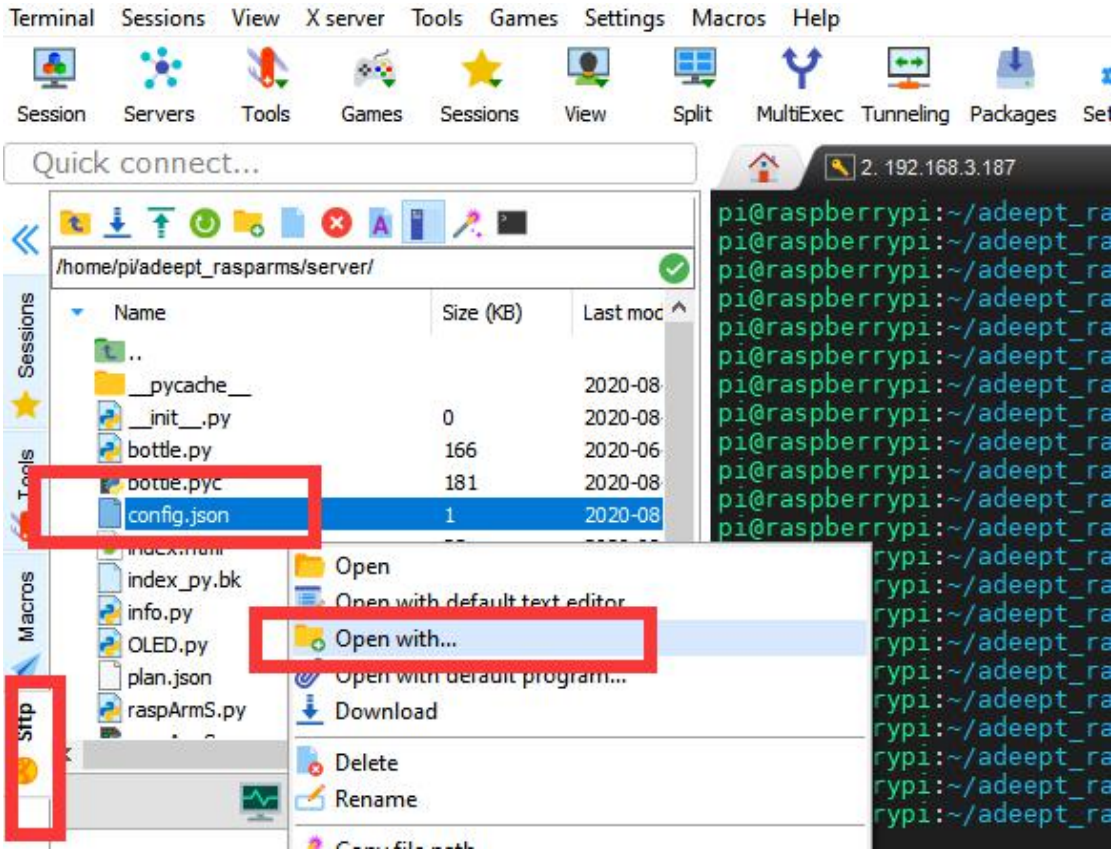
1. To save the modified program, you need to press the shortcut key Ctrl+O on the keyboard

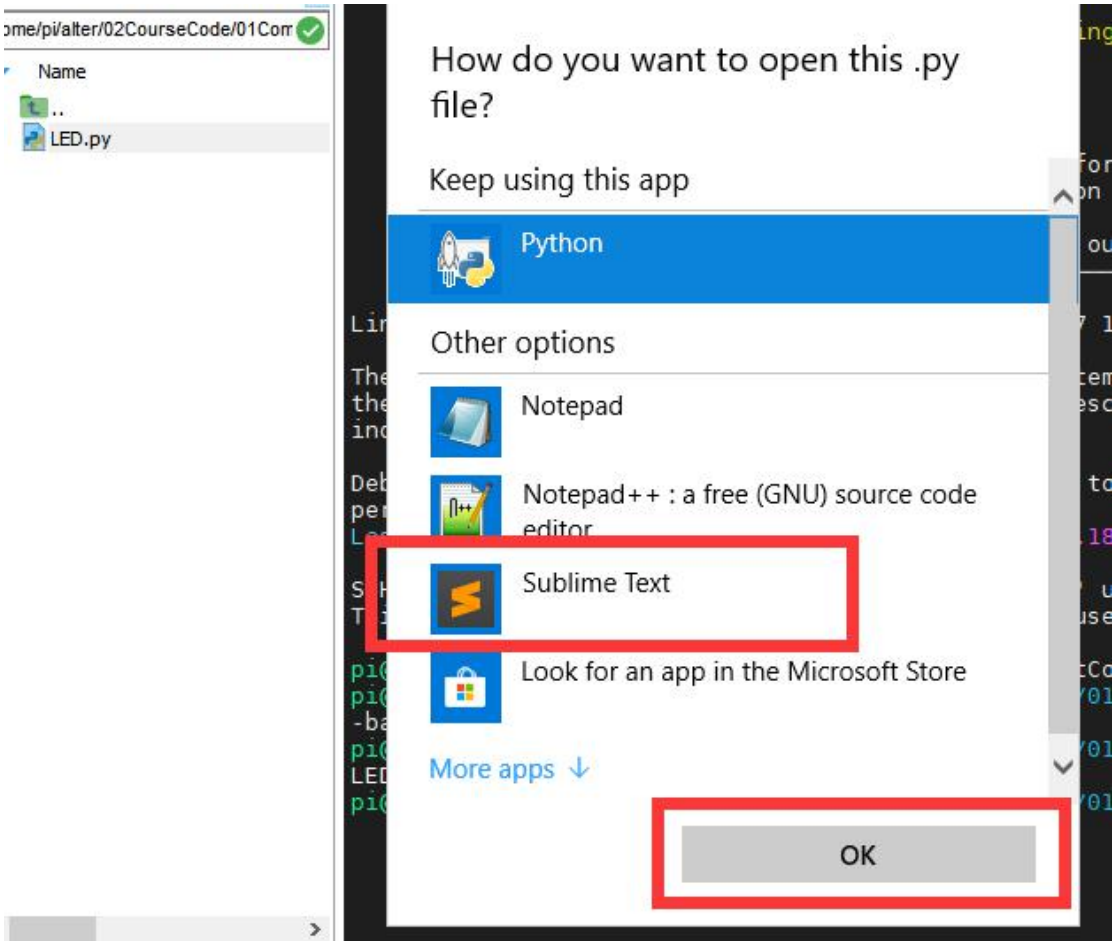
2. To exit the program code interface, you need to press the shortcut key Ctrl+X on the keyboard

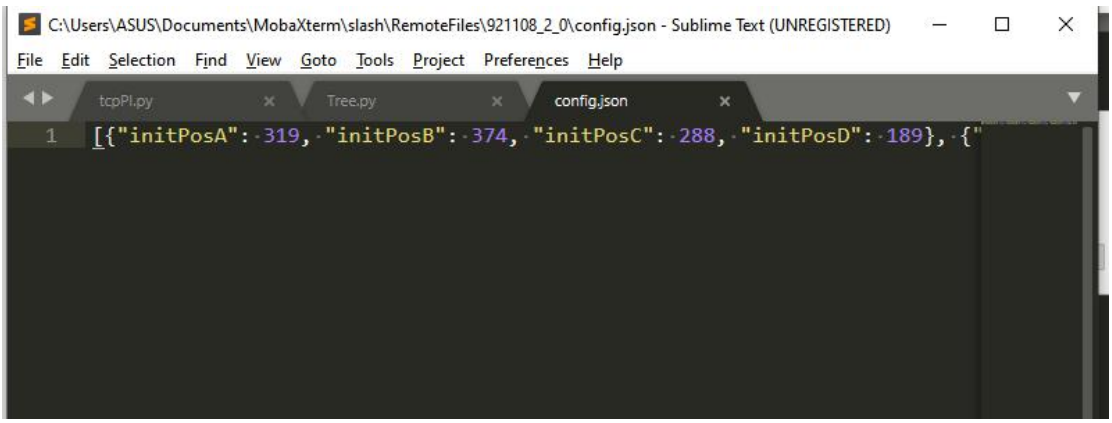
3. To cut a line of code, you can use Ctrl+K

4.To paste the code, you can use Ctrl+U

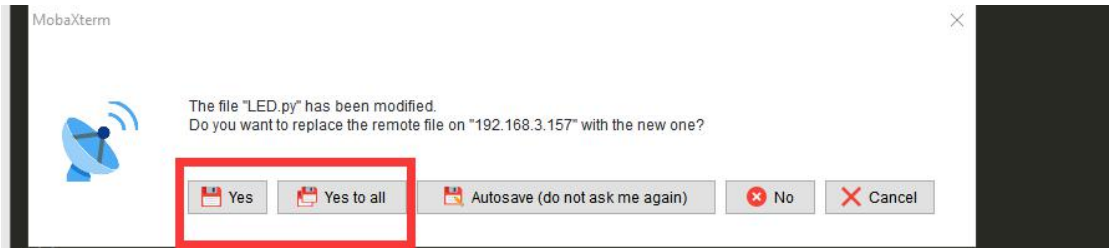
(2) The second method is to use a third-party IDE tool to view and edit the code program of this lesson. You can find the file you want to edit in the file resource management system on the left of MobaXterm, and click the right mouse button on this file. Click Open with, select your IDE, we recommend you to use Sublime Text IDE, you need to download it before you can use: http://www.sublimetext.com/3, so that you can use your favorite IDE to edit the files in the Raspberry Pi. After editing, CTRL+S can save the file.







When you need to save the modified file, you can press the shortcut key CTRL+S, and then select Yes or Yes to all.



Lesson 17 Using Multithreading to Control the Servo of RaspArm-S

In this lesson, we will learn how to use multi-threading to control RaspArms servo.

17.1 Introduction to Multithreading

This lesson introduces the use of multi-threading to realize the control of 180° servo. Multi-threading is a very common operation that we use in robot projects, because robots have high requirements for real-time response. When performing a certain task, it is necessary to try not to block the main thread communication.

Multithreading is similar to executing multiple different programs or tasks at the same time. Multithreading has the following advantages:

1. Using threads to put time-consuming tasks in the background for processing.

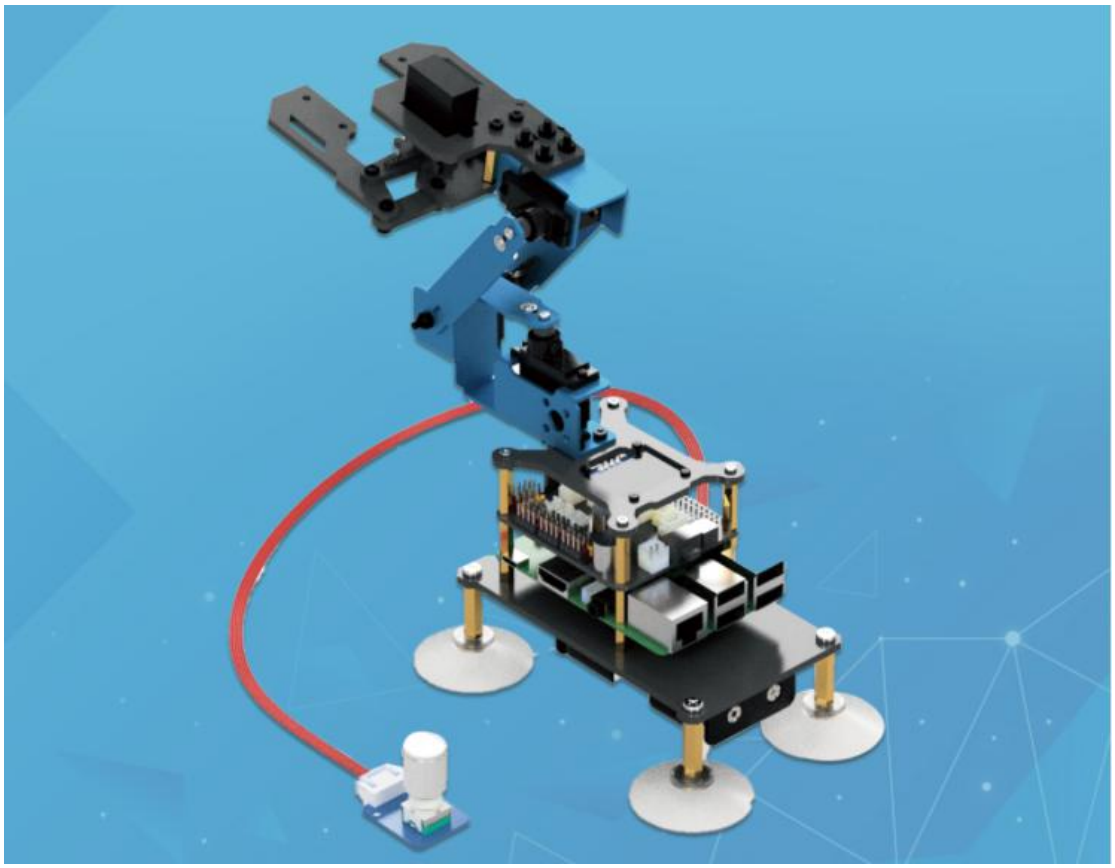
2. Improving the operating efficiency of the program.

3. The encapsulated multi-threaded task is more convenient to call, similar to the non-blocking control method, that is, the control method of the servo is encapsulated by multi-threading

We use Python's threading library to implement multi-threaded control of the robot. The thread is the smallest unit of work in the application. The current Python version does not yet provide multi-thread priority, thread group, thread can not be stopped, suspended, resumed, and interrupted.

17.2 Preparation

You need to prepare the assembled RaspArm-S robotic arm. Make sure that the robotic arm has been connected to the circuit according to the assembled circuit diagram. Refer to "10.2 Robotic Arm Assembly Tutorial" for details.



17.3 Controlling the servo with multiple threads

17.3.1 Learning the code program of multi-thread control servo

We use the following code to implement multi-threaded control of the servo. Here, we use Subline IDE to view and edit the code program of this lesson. Please see section 2.4 in Lesson 2 for specific methods. The specific code and comments are as follows:

In the file manager of the MobaXterm terminal, find the adeept_rasparms/server directory and open the code of this lesson: mutiTest.py.

Import the library socket for TCP communication, import the library threading for enabling multi-threading and the library raspArms for controlling the servo.

```
import socket

import threading

import raspArmS
import time
```

Instantiate the servo control object and start the thread that controls the servo.

```
ras = raspArmS.RaspArmS()
ras.start()
```

This method is used to receive information from the TCP client and perform corresponding actions.

```
def runTcp():
    speed_set = 100
    posBuffer = [0,50,0,0]

    while True:
```

This method is used to establish a TCP server. After the server is established, it starts to monitor the client connection. After the connection is successful, runTcp() is called to receive instructions from the client.

```
def tcpConnection():
```

Define the information related to TCP communication, the port number needs to be consistent with the client.

```
        HOST = ''
        PORT = 10223                            #Define port serial
        BUFSIZ = 1024                           #Define buffer size
        ADDR = (HOST, PORT)
```

Set up a TCP server.

```
        tcpSerSock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        tcpSerSock.setsockopt(socket.SOL_SOCKET,socket.SO_REUSEADDR,1)
        tcpSerSock.bind(ADDR)
```

Start listening for client connections.

```
        tcpSerSock.listen(5)                    #Start server,waiting for client
        print('waiting for connection...')
```

The client connects successfully.

```
        tcpCliSock, addr = tcpSerSock.accept()
        print('...connected from :', addr)
```

Start receiving instructions from the client.

```
120         try:
121             runTcp()
122         except Exception as e:
123             print(e)
124
125         time.sleep(1)
```

Instantiate the thread object.

```
131 rotaryInputThreading=threading.Thread(target=tcpConnection)
```

Start the thread guard so that when the main thread ends, this thread will also end.

```
136 rotaryInputThreading.setDaemon(True)
```

Start the thread.

```
141 rotaryInputThreading.start()
```

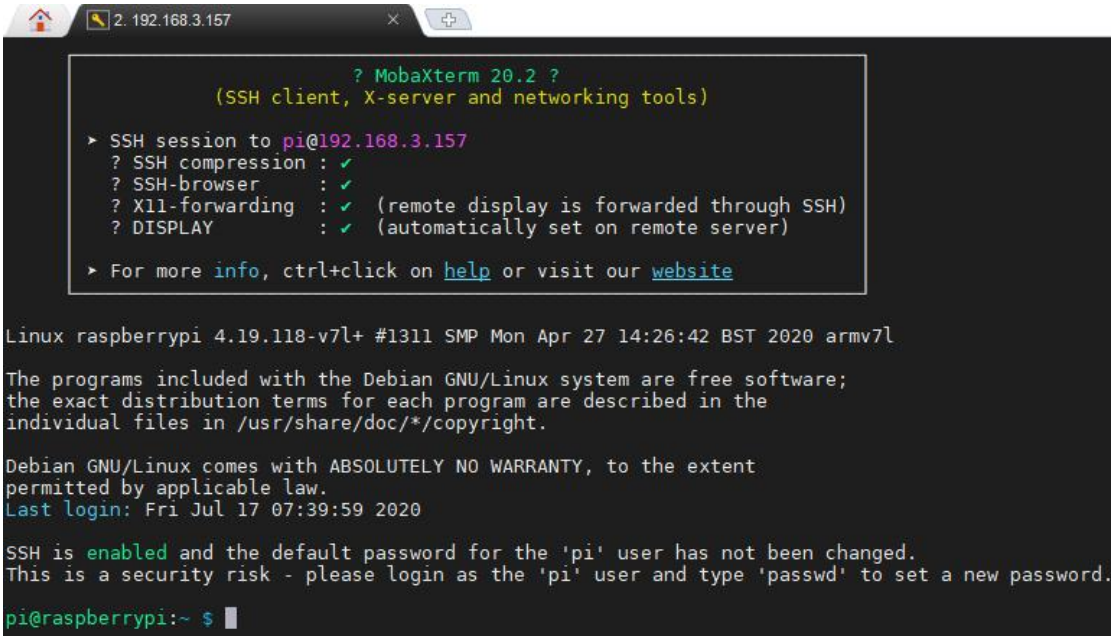
This loop is used to keep the main thread from exiting.

```
150     while 1:
151         time.sleep(100)
152         pass
```

17.3.2 Running the code program of multi-thread control servo

1. Open the terminal software MobaXterm:

2. Log in to your Raspberry Pi (the way to log in to Raspberry Pi has been introduced in Lesson 1):

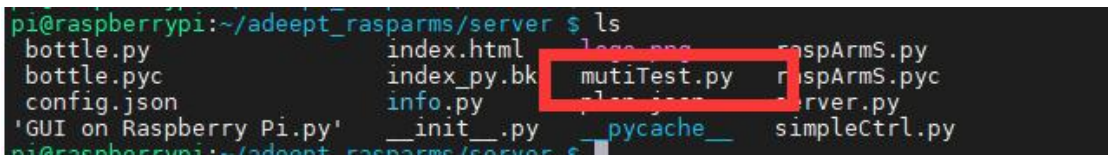


3. Enter the Course Code folder, this folder stores the sample code program to control the robot, enter the following command:

cd adeept_rasparms/server

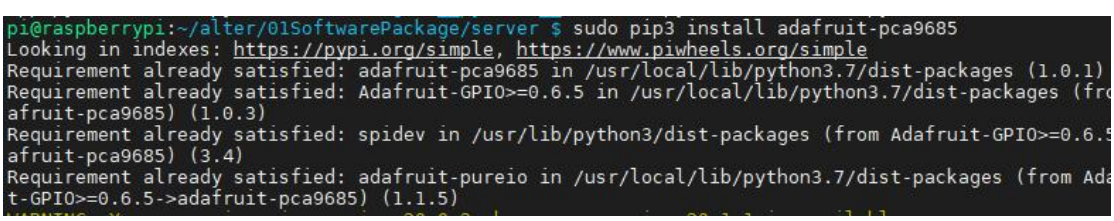
4. Enter the command to display the contents of the current directory:

ls



5. When using the Servo module, we need to install the Python dependency library to control Servo: Adafruit_PCA9685, enter the following command in the console of the command window:

sudo pip3 install adafruit-pca9685



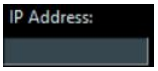
6. mutiTest.py is the sample code for this lesson, enter the command to run this program:

sudo python3 mutiTest.py

After successful operation, you will see the following interface, prompting you to establish a connection with the Raspberry Pi. We follow the next 7th step to open the GUI application control program on the PC.

```
pi@raspberrypi:~/adeept_rasparms/server $ sudo  python3  mutiTest.py
//home/pi/adeept_rasparms/server/plan.json
//home/pi/adeept_rasparms/server/config.json
[{'initPosA': 319, 'initPosB': 374, 'initPosC': 288, 'initPosD': 189},
rvoNumB': 13, 'servoNumC': 14, 'servoNumD': 15}]
Import config [initPos]:
initPosA:319
initPosB:374
initPosC:288
initPosD:189
--- --- ---
Import config [servoPort]:
servoNumA:12
servoNumB:13
servoNumC:14
servoNumD:15
--- --- ---
//home/pi/adeept_rasparms/server/plan.json
you can add something else here.
waiting for connection...
```

7. Open the GUI application on your computer: GUI for Remote.py (there are related operation methods in Lesson 11, find GUI in the directory of adeept_rasparms, click the button  to download the GUI folder to the PC), in the PC double-click to open the GUI for Remote.py in the GUI folder (you must have Python installed on your computer), and the opened interface is as follows:

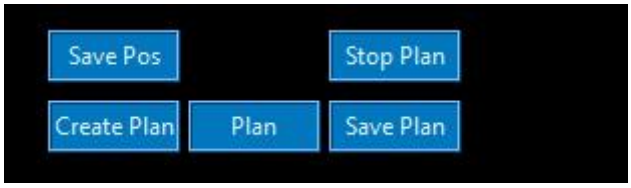
8. You need to enter your Raspberry Pi IP address in 

, and then click the Connect button. After the connection is successful, the button 

on the right will turn green: with the GUI control interface, you can operate the robotic arm.

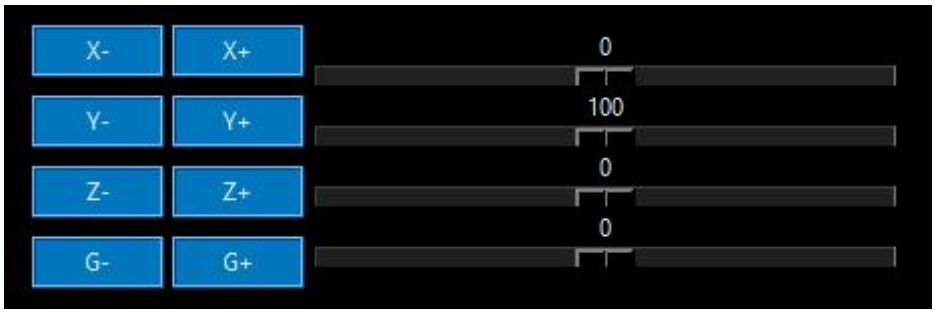
17.4 Controlling the end point of the robotic arm to move between the new position points

When you open the GUI control application for the first time, click the button Plan, and the robot arm will repeat the movement between the last recorded and saved position points (must be 2 different positions), then how to change the end point of the robotic arm to move at different points? Let's learn together.



First click the button Stop Plan to stop the moving robot arm, and then click the button Create Plan to start recording a new position point. At this time, you need to

combine the X, Y, Z, and G sliding buttons in the figure below to locate the end point of the robot arm to make adjustments, such as sliding the X slide bar to the far right, and turning the end point of the robotic arm to the right. At this time, we need to save the position of this coordinate point. By clicking the button Save Pos, we can save the current position; then we also need to record the new position again, continue to slide the X slide bar to the middle position, click the button Save Pos at this time to save the current position; continue to slide the X slide bar to the leftmost position, click the button Save Pos at this time, Save the current position; now we have recorded three different position points, now click the button Plan, the end point of the robotic arm will repeat the movement between the three different position points just recorded. If you want to continue running the location you just recorded and saved when you turn on the Raspberry Pi next time, then you can record the location information by clicking the button Save Plan, and that's it.



When you click the X, Y, Z, G buttons to adjust, the coordinate position of the end point of the robotic arm will be printed in the command window of the Raspberry Pi.

```
Waiting for connection...
...connected from : ('192.168.3.199', 64013)
[0, 100, 0, 0]
[6, 100, 0, 0]
[7, 100, 0, 0]
[18, 100, 0, 0]
[19, 100, 0, 0]
[19, 91, 0, 0]
[19, 90, 0, 0]
[19, 90, 8, 0]
[19, 90, 9, 0]
[19, 90, 9, 12]
[19, 90, 9, 12]
[19, 90, 9, 22]
```

Lesson 18 Introduction to the Soft Motion Method of the Servo

In this lesson, we will learn the soft motion method of the servo.

18.1 Learning the smooth motion method smooth.py

Here we use Subline IDE to view and edit the code program of this course. For the specific method, please see "2.4 Editing the Code Program in Raspberry Pi" in Lesson 2. The specific code and comments are as follows:

In the file manager of the MobaXterm terminal, find adeept_rasparms/CourseCode/06Smooth, and open the code of this lesson: smooth.py.

Import the library that is used to control the servo.

```
4   import Adafruit_PCA9685
5   import time
```

Instantiate the servo control object.

```
pwm = Adafruit_PCA9685.PCA9685()
pwm.set_pwm_freq(50)
```

Define three positions and let the servo move slowly to these positions in sequence.

```
16  servoPos_1 = 150
17  servoPos_2 = 400
18  servoPos_3 = 300
```

Put these positions into the array.

```
23  posList = [servoPos_1, servoPos_2, servoPos_3]
```

Define the port number of the servo you want to control.

```
28  servoNum = 12
```

This is the initial position and the variable which is used to store the end point of the last movement of the servo.

```
33    lastPos = 300
```

Enter a new position in the parameter of this function, the rudder will move smoothly from lastPos to newPosInput.

```
38  def smoothServo(newPosInput):
```

Calculate the difference between lastPos and newPosInput.

```
    errorPos = newPosInput - lastPos
```

Control the servo to move from lastPos to newPosInput little by little.

```
    for i in range(0, abs(errorPos)):
        nowPos = int(lastPos + errorPos*i/abs(errorPos))
        pwm.set_pwm(servoNum, 0, nowPos)
        time.sleep(0.01)
```

Update lastPos as the starting point of the next exercise.

```
60      lastPos = newPosInput
61      pwm.set_pwm(servoNum, 0, lastPos)
```

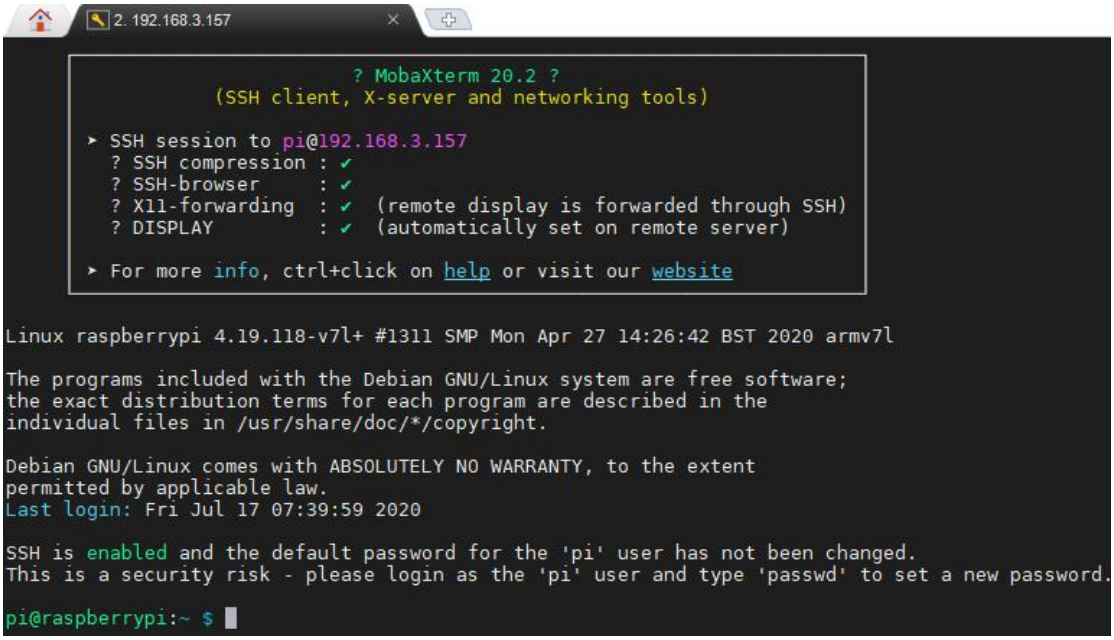
Main function, which is used to control the servo to move slowly to the three points in posList.

```
def main():
    for j in posList:
        smoothServo(j)
```

18.2 Running smooth.py on Raspberry Pi

1. Open the terminal software MobaXterm:

2. Log in to your Raspberry Pi (the way to log in to Raspberry Pi has been introduced in Lesson 1):

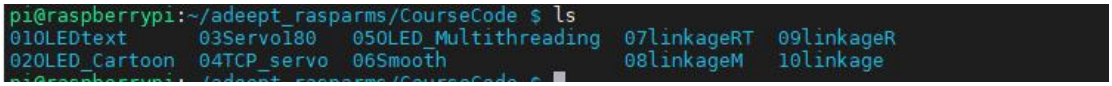


3. The relevant code programs of the RaspArm-S robot are stored in the folder of adeept_rasparms, which has been explained in "2.1 Downloading the Code Program for Controlling the Robot" in Lesson 2. First, you need to enter a command with the command window of the Raspberry Pi to enter the folder where the course code is stored: CourseCode, this folder stores the sample code program for each course, enter the following command:

cd adeept_rasparms/CourseCode

4. Enter the command to view the contents of the current directory:

ls



5. The 06Smooth folder stores the sample code of this lesson. Enter the command to enter this folder:

cd 06Smooth

6. Enter the command to view the contents of the current directory:

ls

```
pi@raspberrypi:~/adeept_rasparms/CourseCode/06Smooth $ ls
smooth.py
```

7.smooth.py is the sample code for this lesson, enter the command to run this program:

sudo python3 smooth.py

```
pi@raspberrypi:~/adeept_rasparms/CourseCode/06Smooth $ sudo  python3  smooth.py
```

8. After running the program successfully, you will observe that the robotic arm will move between the three positions set in the program.

Lesson 19 Inverse Solution Function of Plane Link

In this lesson, we will learn the inverse solution function of the plane link.

19.1 Brief description

The plane linkage mechanism is a plane mechanism that composed of several components connected by a low pair (rotating pair, moving pair). The simplest plane linkage mechanism consists of four components, called plane four-bar mechanism. The plane linkage mechanism is widely used in various machinery, instruments and various electromechanical products.

The main advantages of the plane linkage mechanism are:

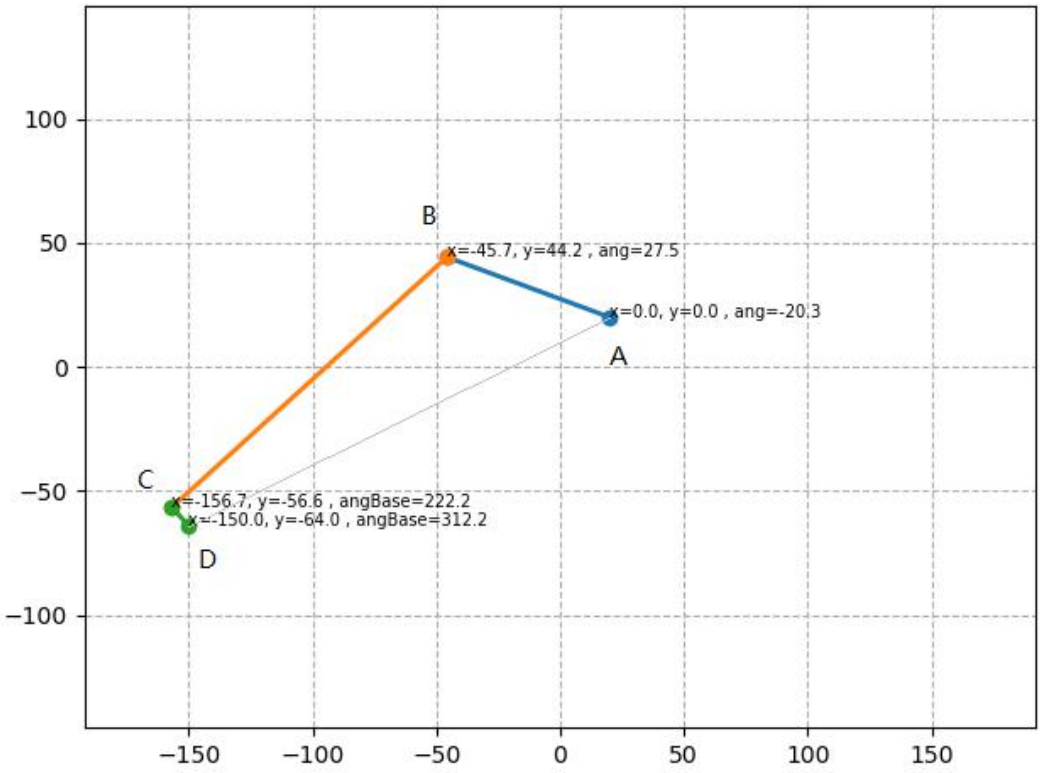
1) Since the kinematic pairs in the linkage mechanism are all low pairs of surface contact, they bear low pressure, which is convenient for lubrication, with lighter wear and high load-bearing capacity;

2) The shape of the component is simple, and it is convenient to add T. The contact between the components is maintained by the geometric constraints of the component itself, so the component works reliably;

3) It can realize the transformation between multiple sports forms:

4) The use of connecting rods can achieve a variety of motion trajectory requirements.

In our robotic arm, the plane connecting rod structure is also used. As shown in the figure below, ABCD constitutes a simple connecting rod, where the rotation of point A and point B is driven by the servo, and the connecting rod CD and the connecting rod BC are fixed right angle, the connecting rod CD is used to compensate the error of the end point of the connecting rod.



19.2 Learning the code program of the inverse function of the plane Link

Here we use Subline IDE to view and edit the code program of this course. For the specific method, please see "2.4 Editing the Code Program in Raspberry Pi" in Lesson 2.

In the file manager of the MobaXterm terminal, find adeept_rasparms/CourseCode, download the folder 07linkageRT to your PC, select this folder, click the button  to download the file to the PC, save it in the path of English letters, and use Subline IDE to open the linkageRT.py in the 07linkageRT folder. The main code and comments are as follows:

Import the mathematical function library numpy.

```
6  import numpy as np
7
8  import time
```

Set the length of AB to 70, the length of BC to 150.0, and the length of CD to 10.

```
linkageLenA = 70
linkageLenB = 150.0

linkageLenDebug = 10
```

servoNumCtrl = [0,1] is the number of the AB servos; servoDirection = [1,-1] corresponds to the movement direction of the AB servos, set by 1 and -1.

```
servoNumCtrl = [0,1]
servoDirection = [1,-1]
```

The array linkageLenCtrl which is composed of the lengths of links AB and BC.

```
linkageLenCtrl = [linkageLenA, linkageLenB]
```

The function planeLinkageReverse() is used to input the length of the link and the error parameters, the end position of the link and the number of the servos, and the rotation angle of the AB servos of this connecting rod mechanism can be obtained.

```
def planeLinkageReverse(linkageLen, linkageEnDe, servoNum, debugPos, goalPos):
```

In the planeLinkageReverse() function, first calculate the incremental error, and use debugPos to correct the initial position error. Generally, debugPos is [0,0].

```
    goalPos[0] = goalPos[0] + debugPos[0]
    goalPos[1] = goalPos[1] + debugPos[1]

```

Used to calculate the angle error between the end point of the link and the rotation axis of the servo.

```
    AngleEnD = np.arctan(linkageEnDe/linkageLen[1])*180/np.pi
```

The input links AB, BC, and CD are abstracted into AB and BD to facilitate subsequent calculations.

```
    linkageLenREAL = np.sqrt(((linkageLen[1]*linkageLen[1])+(linkageEnDe*linkageEnDe)))

```

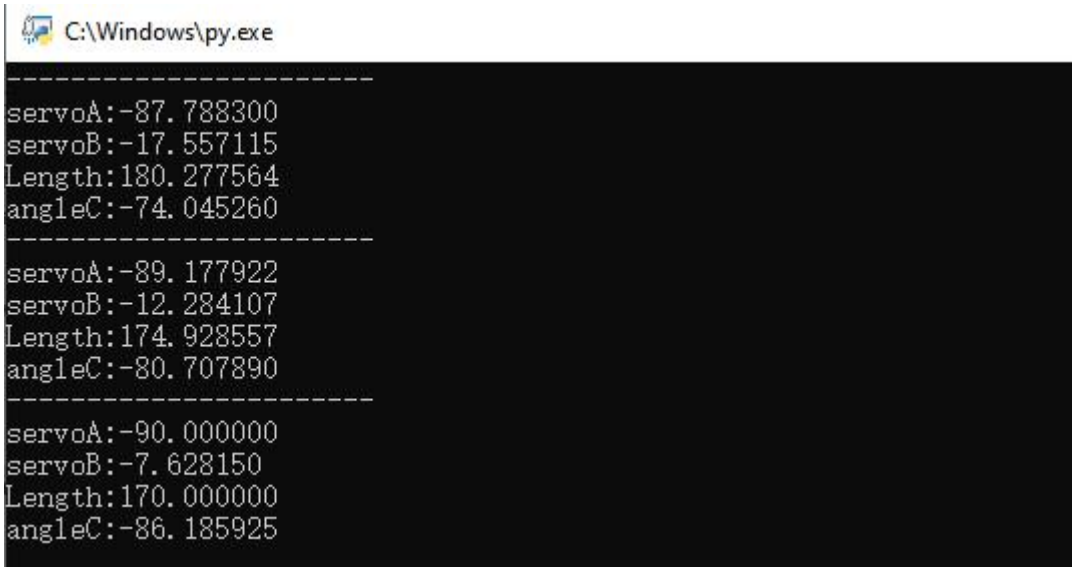
Perform trigonometric calculations.

```
    if goalPos[0] < 0:
        goalPos[0] = -goalPos[0]
        mGenOut = linkageLenREAL*linkageLenREAL-linkageLen[0]*linkageLen[0]
        nGenOut = mGenOut/(2*linkageLen[0])

        angleGenA = np.arctan(goalPos[1]/goalPos[0])+np.arcsin(nGenOut/np.s
        angleGenB = np.arcsin((goalPos[1]-linkageLen[0]*np.cos(angleGenA)),
```

19.3 Running linkageRT.py on the PC

Find the folder 07linkageRT that you downloaded in step 19.2 on your PC, double-click to open the linkageRT.py program inside, you must make sure you have downloaded and installed Python according to "Lesson 4".

After double-clicking to open, as shown in the figure below, in the running command window, the printed data are the rotation angle of servo A; the rotation angle of servo B; the length of AD; the length of AD; the angle of AD.



Lesson 20 Drawing Lines with Matplotlib

In this lesson, we will learn how to use matplotlib to draw simple lines.

20.1 Brief description

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+. There is also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged.[3] SciPy makes use of Matplotlib.

Matplotlib was originally written by John D. Hunter, since then it has an active development community,[4] and is distributed under a BSD-style license. Michael Droettboom was nominated as matplotlib's lead developer shortly before John Hunter's death in August 2012,[5] and further joined by Thomas Caswell.

20.2 Learning the code program of linkageM.py

Here we use Subline IDE to view and edit the code program of this course. For the specific method, please see "2.4 Editing the Code Program in Raspberry Pi" in Lesson 2.

In the file manager of the MobaXterm terminal, find adeept_rasparms/CourseCode, download the 08linkageM folder to your PC, select this folder, click the button  to download the file to the PC, save it in the path of English letters, and use Subline IDE opens linkageM.py in the 08linkageM folder. The main code and comments are as follows:

Import numpy.

```
1  import numpy as np
2  '''
```

Import matplotlib.

```
from matplotlib import pyplot as plt
```

The function drawLine() is used to input the coordinates pos1 of the initial point of the line segment and the coordinates pos2 of the end point.

```
def drawLine(pos1, pos2):
```

In the drawLine() function, X represents an array of X points, and Y represents an array of Y points. Use plt.plot(x, y) to draw this line segment.

```
    x = [pos1[0], pos2[0]]
    y = [pos1[1], pos2[1]]
    plt.plot(x, y)
```

Call the drawLine() function to draw a line segment starting at (0,0) and ending at (2,2).

```
drawLine([0,0], [2,2])
```

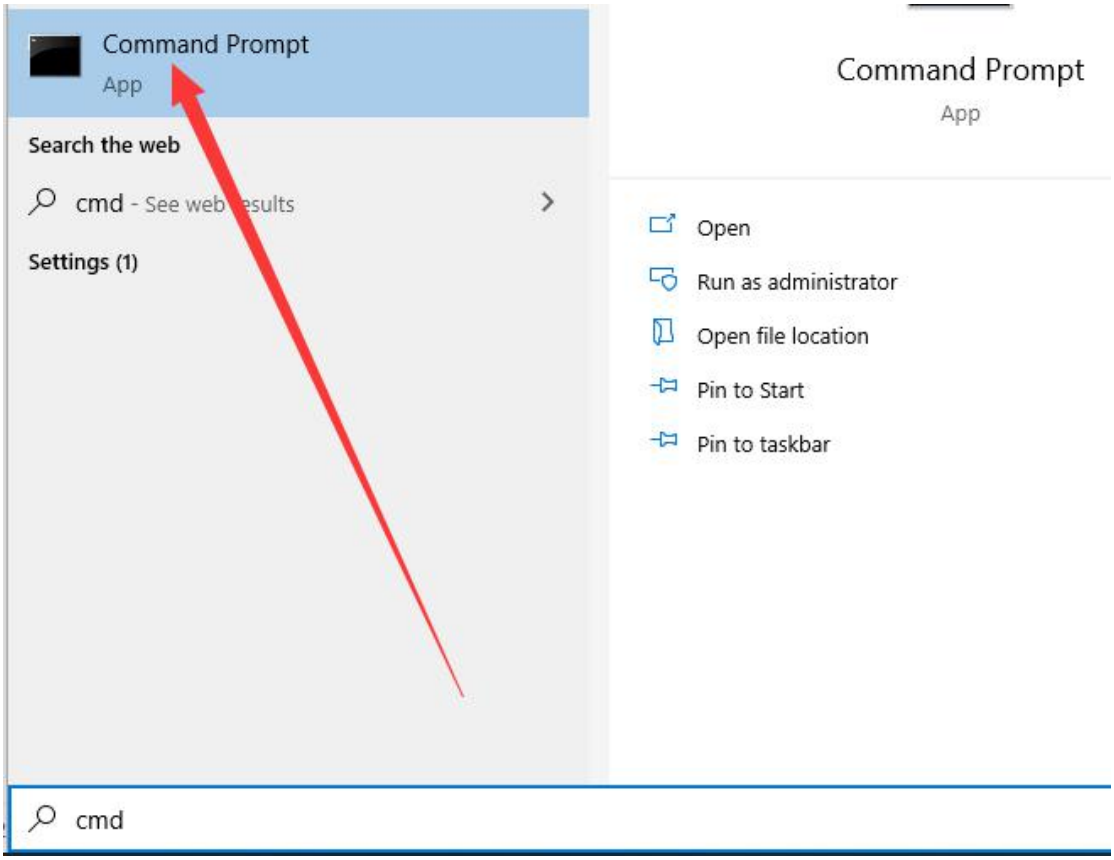
Show the drawn line.

```
plt.show()
```

20.3 Running the linkageM.py program on the PC

Find the folder 08linkageM you downloaded in step 20.2 on your PC, copy the linkageM.py program inside to the C drive directory, and double-click to open the linkageM.py program inside. Make sure you have followed Lesson 4 to download and install Python.

1. First open the cmd command window on the PC:



2. In the opened command window, enter the command to install matplotlib:

pip install matplotlib

3. After the installation is complete, you can use the mouse to double-click to open the linkageM.py, and then you will see the following picture, the middle is the line segment drawn by the program.

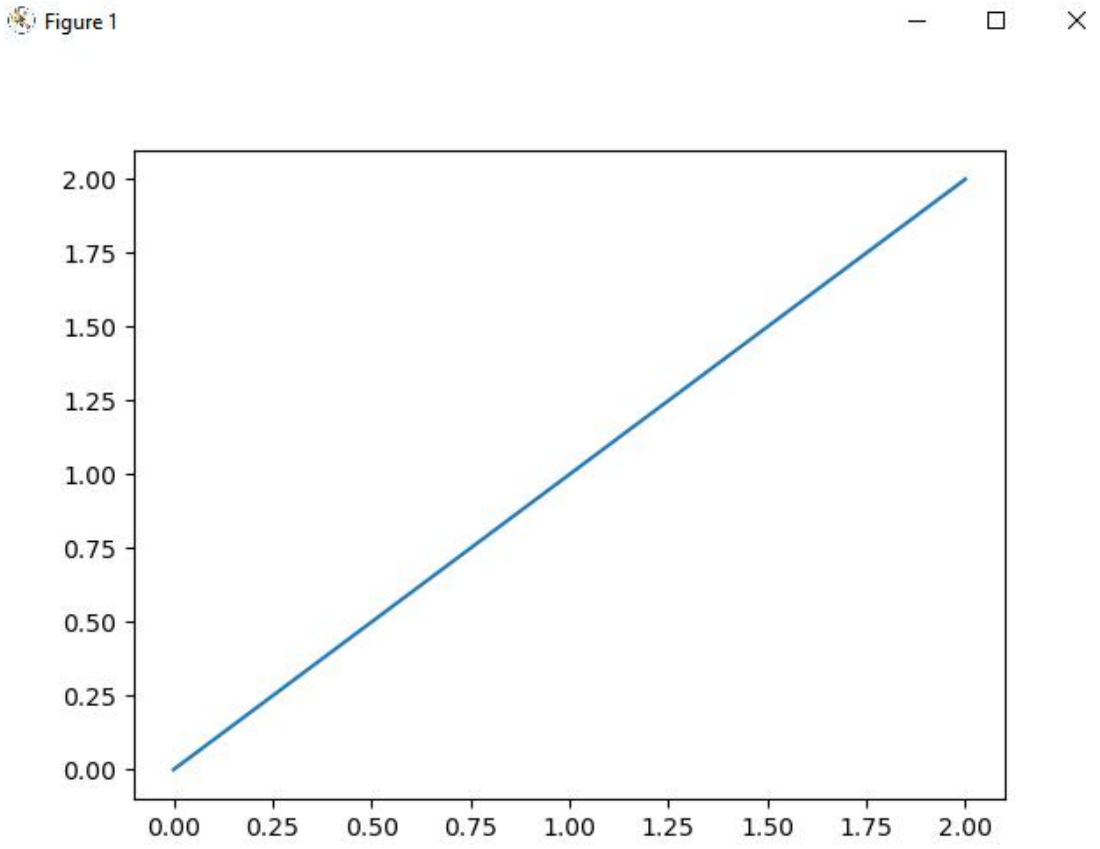


Figure 1
2.00
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

Lesson 21 Simulating a Plane Link with Matplotlib

In this lesson, we will learn how to use matplotlib to simulate a plane link.

21.1 Brief description

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+. There is also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged. SciPy makes use of Matplotlib.

Matplotlib was originally written by John D. Hunter, since then it has an active development community, and is distributed under a BSD-style license. Michael Droettboom was nominated as matplotlib's lead developer shortly before John Hunter's death in August 2012, and further joined by Thomas Caswell.

21.2 Learning the code program of linkageR.py

Here we use Subline IDE to view and edit the code program of this course. For the specific method, please see "2.4 Editing the Code Program in Raspberry Pi" in Lesson 2.

In the file manager of the MobaXterm terminal, find the adeept_rasparms/CourseCode, download the folder 09linkageR to your PC, select this folder, click the button  to download the file to the PC, save it in the path of English letters, and use The Subline IDE opens the linkageR.py in the 09linkageR folder. The main code and comments are as follows:

Import numpy.

```
1  import numpy as np
2  '''
```

Import matplotlib.

```
from matplotlib import pyplot as plt
```

Import the library for animation.

```
import matplotlib.animation as animation
```

Set the length of AB to 70, the length of BC to 150.0, and the length of CD to 10.

```
linkageLenA = 70
linkageLenB = 150.0

linkageLenDebug = 10
```

Define the initial position deviation of the robotic arm.

```
debugPosA = 0.0
debugPosB = 0.0
```

Define the text size displayed here.

```
sizeFont = 7
```

servoNumCtrl = [0,1] is the number of the AB servos; servoDirection = [1,-1] corresponds to the movement direction of the AB servos, set by 1 and -1.

```
servoNumCtrl = [0,1]
servoDirection = [1,-1]
```

The array linkageLenCtrl composed of the lengths of links AB and BC.

```
linkageLenCtrl = [linkageLenA, linkageLenB]
```

The function planeLinkageReverse() is used to input the length of the connecting rod and the error parameters, the end position of the link and the number of the servos, and the rotation angle of the AB servos of this link mechanism can be obtained.

```
def planeLinkageReverse(linkageLen, linkageEnDe, servoNum, debugPos, goalPos):
```

In the planeLinkageReverse() function, first calculate the incremental error, and use debugPos to correct the initial position error. Generally, debugPos is [0,0].

```
    goalPos[0] = goalPos[0] + debugPos[0]
    goalPos[1] = goalPos[1] + debugPos[1]
```

Used to calculate the angle error between the end point of the link and the

rotation axis of the servo B.

```
    AngleEnD = np.arctan(linkageEnDe/linkageLen[1])*180/np.pi
```

The input links AB, BC, and CD are abstracted into AB and BD to facilitate subsequent calculations.

```
45      linkageLenREAL = np.sqrt(((linkageLen[1]*linkageLen[1])+(linkageEnDe*linkageEnDe)))
46
```

Perform trigonometric calculations.

```
    if goalPos[0] < 0:
        goalPos[0] = - goalPos[0]
        mGenOut = linkageLenREAL*linkageLenREAL-linkageLen[0]*linkageLen[0]
        nGenOut = mGenOut/(2*linkageLen[0])

        angleGenA = np.arctan(goalPos[1]/goalPos[0])+np.arcsin(nGenOut/np.s
        angleGenB = np.arcsin((goalPos[1]-linkageLen[0]*np.cos(angleGenA))
```

The animateLine() function is used to draw a line segment. The input parameters are the initial coordinate point X and Y values of the line segment, the length of the line segment, the angle of the line segment, and the angle deviation.

```
113 def animateLine(xInput, yInput, lineLen, angleInput, debugInput):
114     angleOri = angleInput
115     debugOri = debugInput
116
117     aOut = angleInput + debugInput
```

Initialize the link segment to be moved.

```
168 fig = plt.figure(tight_layout=True)
169 line_1, = plt.plot([],[],'o-',lw=2)
170 line_2, = plt.plot([],[],'o-',lw=2)
171 line_3, = plt.plot([],[],'o-',lw=2)
172
```

Initialize the text to be displayed.

```
176 textPoint_A = plt.text(4, 0.8, '', fontsize=sizeFont)
177 textPoint_B = plt.text(4, 0.8, '', fontsize=sizeFont)
178 textPoint_C = plt.text(4, 0.8, '', fontsize=sizeFont)
179 textPoint_D = plt.text(4, 0.8, '', fontsize=sizeFont)
180
```

Animate() is the animation function.

```
184 def animate(i):
```

In the Animate() function, call the planeLinkageReverse() function to get the rotation angle of the A and B servos. Note that there is a variable i, which is used to

make animations, which will be explained later.

```
    a = planeLinkageReverse(linkageLenCtrl, linkageLenDebug, servoNumCtrl, [20,20], [150,-100+i])
```

Call the animateLine() function to draw the line segment.

```
    [x1,y1,a1] = animateLine(20, 20, linkageLenA, a[0], 180)
    [x2,y2,a2] = animateLine(x1[1],y1[1],linkageLenB,a[1],a1+90)
    [x3,y3,a3] = animateLine(x2[1],y2[1],linkageLenDebug,a2,90)
```

Apply animation to linkage.

```
    line_1.set_data(x1,y1)
    line_2.set_data(x2,y2)
    line_3.set_data(x3,y3)
```

Show some key information on the link.

```
    textPoint_A.set_text("x=%.1f, y=%.1f , ang=%.1f"%(debugPosA, debugPosB, a[0]*servoDirection[servoNumCtrl[0]]))
    textPoint_B.set_text("x=%.1f, y=%.1f , ang=%.1f"%(x1[1], y1[1], a[1]*servoDirection[servoNumCtrl[1]]))
    textPoint_C.set_text("x=%.1f, y=%.1f , angBase=%.1f"%(x2[1], y2[1], a2))
    textPoint_D.set_text("x=%.1f, y=%.1f , angBase=%.1f"%(x3[1], y3[1], a3))
```

Text application animation.

```
    textPoint_A.set_position((20, 20))
    textPoint_B.set_position((x1[1], y1[1]))
    textPoint_C.set_position((x2[1], y2[1]))
    textPoint_D.set_position((x3[1], y3[1]))
```

Note here that there must be a',' after the last return value, do not delete.

```
    return line_1,line_2,line_3,textPoint_A,textPoint_B,textPoint_C,textPoint_D,
```

Create a new artboard for animation drawing.

```
plt.axis("equal")
plt.grid()
plt.grid(ls="--")
plt.xlim(-200,200)
plt.ylim(-200,200)
```

Use animation to define range() to determine the range of animation variables.

```
ani = animation.FuncAnimation(fig, animate, range(0, 200),
                              interval=10, blit=True, init_func=initAnimate)
```

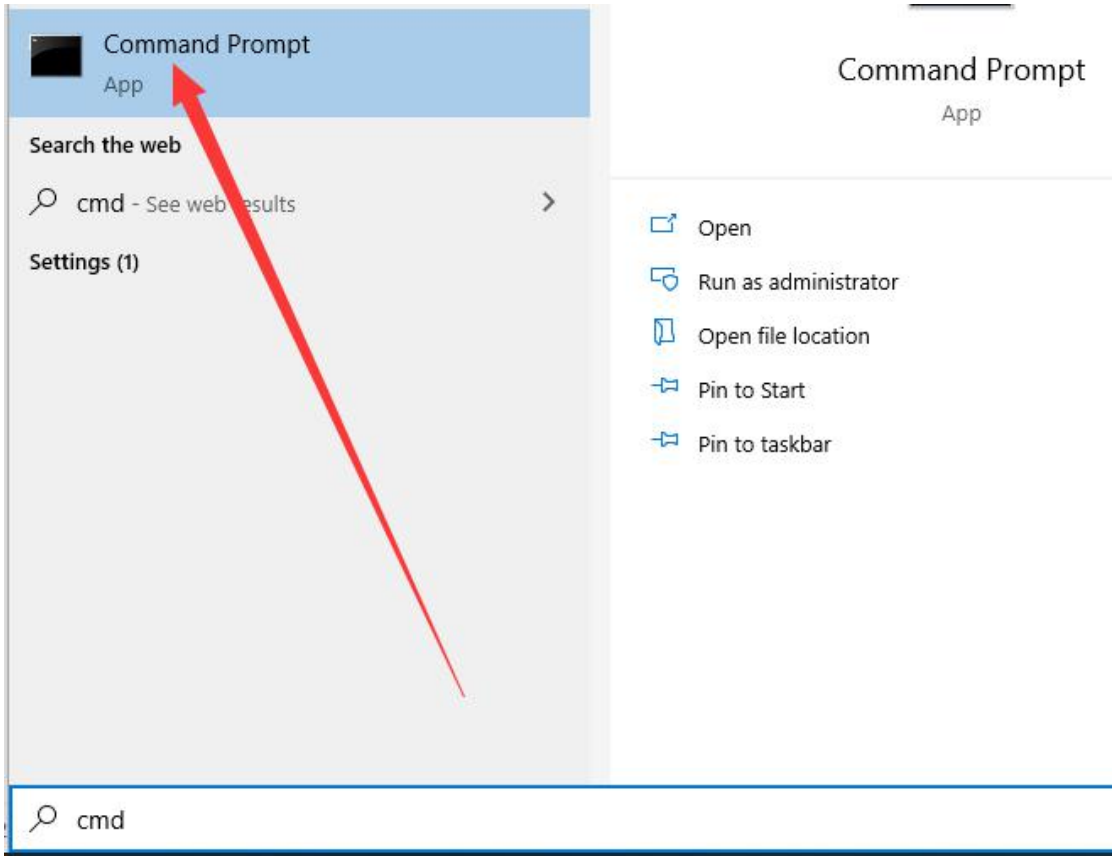
Display simulation motion model.

```
plt.show()
```

21.3 Running the linkageR.py program on the PC

On your PC, find the folder 09linkageR that you downloaded in step 21.2, copy the linkageR.py program inside to the C drive directory, and double-click to open the linkageR.py program inside. Make sure you have downloaded in accordance with “Lesson 4 Downloading and Installing Python".

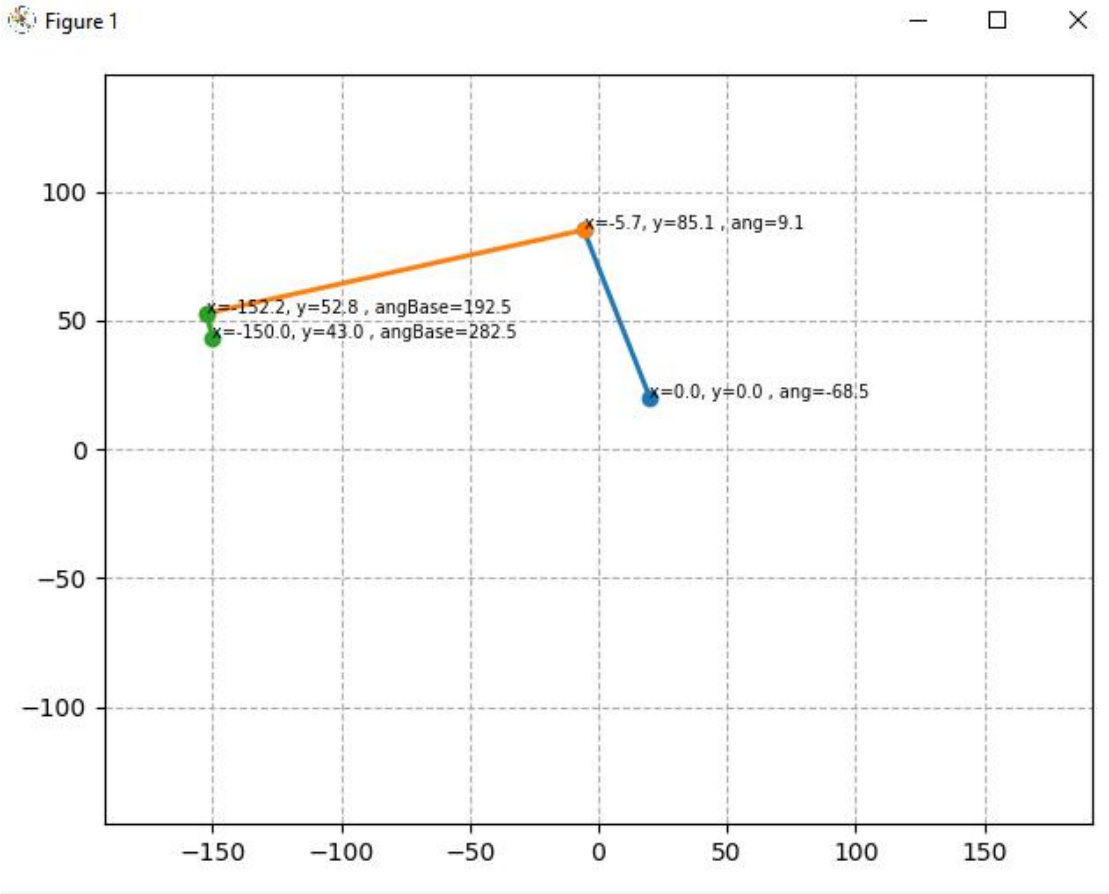
1. First open the cmd command window on the PC:



2. In the opened command window, enter the command to install matplotlib:

pip install matplotlib

3. After the installation is complete, you can use the mouse to double-click to run linkageR.py, and then you will see the following figure, which is the plane link simulation of matplotlib.



Lesson 22 Matplotlib Dual-view Linkage Simulation

In this lesson, we will learn how to use matplotlib for dual-view linkage simulation.

21.1 Brief description

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK+. There is also a procedural "pylab" interface based on a state machine (like OpenGL), designed to closely resemble that of MATLAB, though its use is discouraged. SciPy makes use of Matplotlib.

Matplotlib was originally written by John D. Hunter, since then it has an active development community and is distributed under a BSD-style license. Michael Droettboom was nominated as matplotlib's lead developer shortly before John Hunter's death in August 2012, and further joined by Thomas Caswell.

22.2 Learning the code program of linkage.py

Here we use Subline IDE to view and edit the code program of this course. For the specific method, please see "2.4 Editing the Code Program in Raspberry Pi" in Lesson 2.

In the file manager of the MobaXterm terminal, find the adeept_rasparms/CourseCode, download the folder 10linkage to your PC, select this folder, click the button  to download the file to the PC, save it in the path of English letters, and use The Subline IDE opens the linkage.py in the 10linkage folder. The main code and comments are as follows:

Import numpy.

```
import numpy as np
'''
```

Import matplotlib.

```
from matplotlib import pyplot as plt
```

Import the library for animation.

```
import matplotlib.animation as animation
```

Here defines the distance between the A-axis of the servo and the servo axis responsible for the left and right swings. This distance is determined by the structure and cannot be changed.

```
linkageLenC = 40
```

Define the position of the oscillating servo here to correct the initial position error, generally 0 is enough.

```
linkageLenD = 0
linkageLenE = 0
```

Define the text size.

```
sizeFont = 7
```

servoNumCtrl = [0,1] is the number of the AB servos; servoDirection = [1,-1] corresponds to the movement direction of the AB servos, set by 1 and -1.

```
servoNumCtrl = [0,1]
servoDirection = [1,-1]
```

This function is used to input the three-dimensional coordinates of the end point of the link, and return the rotation angle of the left and right swing servo and the length of the yellow line segment.

[Important Reminder]: The length of the yellow line segment is the X value of the link inverse solution function of the previous course, and the Z value here is the Y value of the link inverse solution function of the previous course.

```
def bodyCoordinatePointCtrl(x, y, z):
    x = x - linkageLenE/2
    y = y + linkageLenD/2
    y = -y
```

The function used to draw the line segment is the same as in the previous lesson.

```
def animateLine(xInput, yInput, lineLen, angleInput, debugInput):
    angleOri = angleInput
    debugOri = debugInput
```

The Animate() function is responsible for generating animation, the i value inside the function is the input variable.

```
def animate(i):
```

Return the rotation angle of the left and right servo and the length of the yellow line segment.

```
    a = bodyCoordinatePointCtrl(100-i, -130, 60)
```

Call the animateLine() function to draw the line segment.

```
    [x1,y1,a1] = animateLine(linkageLenE/2, -linkageLenD/2, linkageLenC, a[0], -90)
    [x2,y2,a2] = animateLine(x1[1],y1[1],a[1],0,a1)
```

Apply animation to linkage.

```
    line_1.set_data(x1,y1)
    line_2.set_data(x2,y2)
```

Show some key information on the link.

```
    textPoint_A.set_text("x=%.1f, y=%.1f, ang=%.1f"%(linkageLenE/2, linkageLenD/2, a[0]*servoDirection[servoNumCtrl[0]]))
    textPoint_B.set_text("x=%.1f, y=%.1f, ang=%.1f"%(x1[1], y1[1], a[1]*servoDirection[servoNumCtrl[1]]))
    textPoint_C.set_text("x=%.1f, y=%.1f, angBase=%.1f"%(x2[1], y2[1], a2))
```

Note here that there must be a',' after the last return value, do not delete.

```
    return line_1,line_2,textPoint_A,textPoint_B,textPoint_C,

```

Create a new artboard for animation drawing.

```
plt.axis("equal")
plt.grid()
plt.grid(ls="--")
plt.xlim(-200,200)
plt.ylim(-200,200)
```

Use animation to define range() to determine the range of animation variables.

```
ani = animation.FuncAnimation(fig, animate, range(0, 200),
                              interval=10, blit=True, init_func=initAnimate)
```

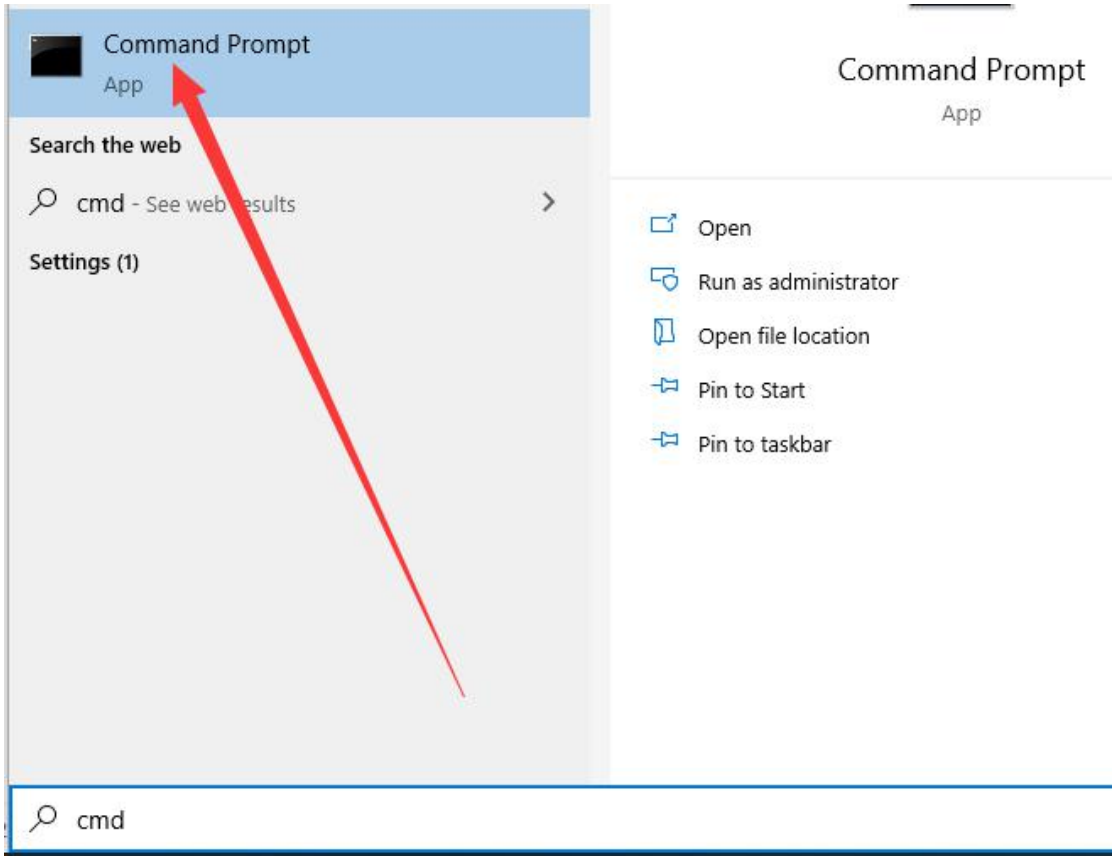
Display simulation motion model.

```
plt.show()
```

22.3 Running the linkage.py program on the PC

On your PC, find the folder 10linkage that you downloaded in step 22.2, copy the linkage.py program inside to the C drive directory, and double-click to open the linkage.py program inside. Make sure you have downloaded in accordance with “Lesson 4 Downloading and Installing Python".

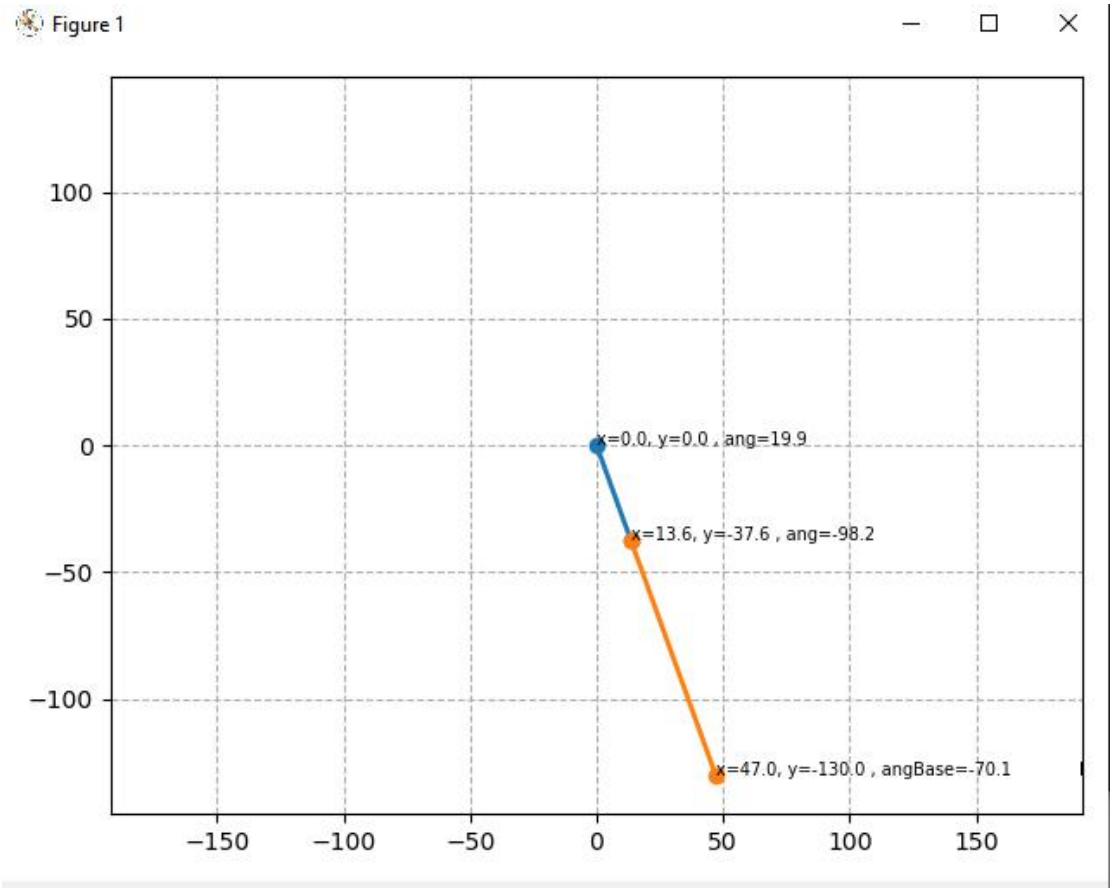
1. First open the cmd command window on the PC:



2. In the opened command window, enter the command to install matplotlib:

pip install matplotlib

3. After the installation is complete, you can use the mouse to double-click to run linkage.py, and then you will see the following figure, which is the matplotlib's dual-view linkage simulation.



Lesson 23 Reading and Saving json Files

In this lesson, we will learn how to read and save json files.

23.1 Learning the code program of rwJson.py

Here we use Subline IDE to view and edit the code program of this course. For the specific method, please see "2.4 Editing the Code Program in Raspberry Pi" in Lesson 2.

In the file manager of the MobaXterm terminal, find adeept_rasparms/server, and open the code of this lesson: rwJson.py. The specific code and comments are as follows:

Import related dependent libraries.

```
4  import json
5  import os
6
```

Get the absolute path of the .py file to find plan.json.

```
10  curpath = os.path.realpath(__file__)
11  thisPath = "/" + os.path.dirname(curpath) + "/"
```

Open plan.json.

```
16  planJsonFileHere = open(thisPath + 'plan.json', 'r')
```

Read the content of plan.json.

```
21  contentPlanGose = planJsonFileHere.read()
```

Decode the read content into the original format.

```
26  planGoseList = json.loads(contentPlanGose)
27
```

At this point you can use planGoseList like a normal array.

```
31  print(planGoseList)
32
```

Convert the array to be saved into a json string.

```
37  content2write = json.dumps(planGoseList)
```

You can change the file name of newPlans.json and create a new file.

```
42  file2write = open('newPlan.json', 'w')
```

Write json string in the newly created file.

```
47  file2write.write(content2write)
```

Save and close the newly created file.

You can find a file named newPlan.json appeared in the folder.

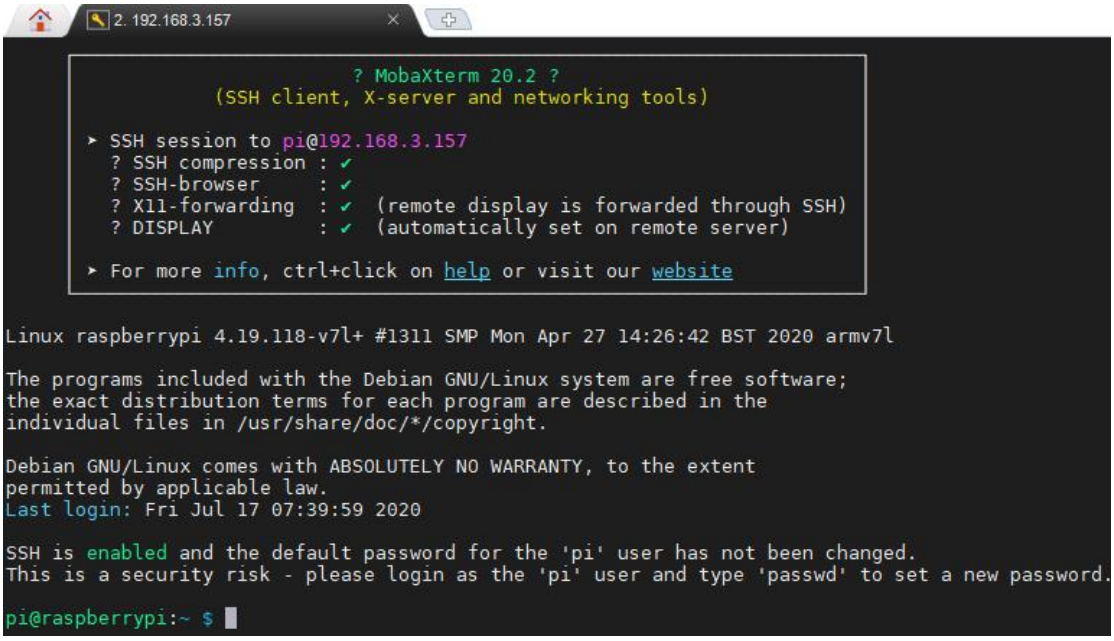
You can use the above method to import this file.

```
54  file2write.close()
```

23.2 Running rwJson.py program on Raspberry Pi

1. Open the terminal software MobaXterm:

2. Log in to your Raspberry Pi (the way to log in to the Raspberry Pi has been introduced in Lesson 1):

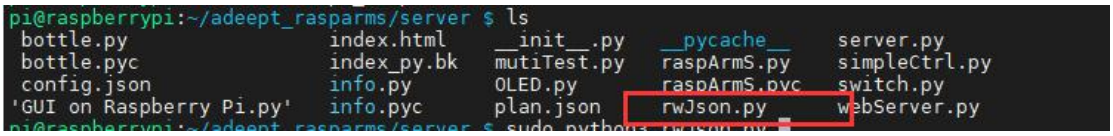


3. The relevant code programs of the RaspArm-S robot are stored in the folder of adeept_rasparms, which have been explained in "2.1 Downloading the code program to control the robot" in Lesson 2.First, you need to enter the server directory with the command window of the Raspberry Pi and enter the following command:

cd adeept_rasparms/server

4. Enter the command to view the contents of the current directory:

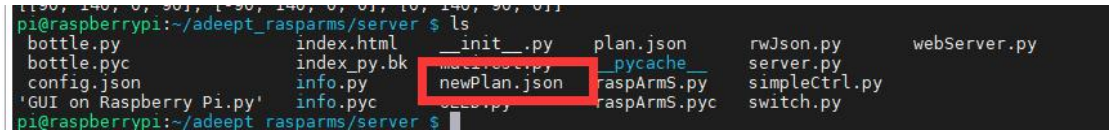
ls



5.Enter the command to run the program:

sudo python3 rwJson.py

6.You can find a file named newPlan.json appears in the folder.



Lesson 24 Using json Data to Record and Repeat Actions

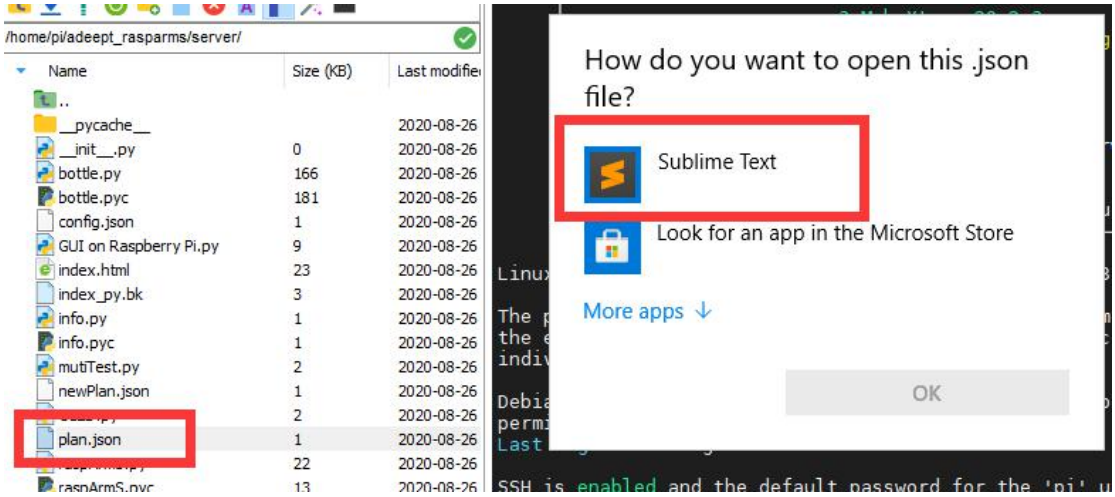
In this lesson, we will learn how to use json data to record and repeat actions.

24.1 Editting a new motion path

In the adeept_rasparms/server directory of the Raspberry Pi, there is a plan.json file, edit this file, you can edit and save the motion path information of the robotic arm, save it after editing, and the saved motion path information can be automatically run next time you run the robotic arm program. Here is how to modify it.

1. Open the terminal software MobaXterm:

2. Log in to your Raspberry Pi (the way to log in to the Raspberry Pi has been introduced in Lesson 1), in the file manager of the MobaXterm terminal, find the adeept_rasparms/server directory, right-click the file plan.json, open the plan.json file by selecting the third-party tool Sublime Text by "Open with" in the options:



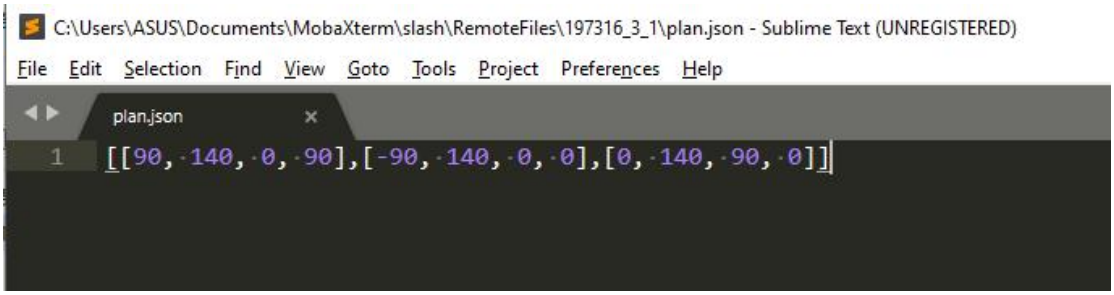
【Note】

If you don't have the third-party tool Sublime Text, you can use the nano command of Linux to operate, and the specific method can be viewed in 2.4.

3. After opening, as shown below, the data of these two-dimensional arrays is to save and record the coordinate data of the robot arm at a certain position. Each element in the array represents the coordinate information of a position, such as [90,140,0,90] ,it means the coordinate information of a position point. The first data is the X-axis coordinate position, the second data is the Y-axis coordinate position, the third data is the Z-axis coordinate position, and the fourth data is the rotation of the chuck. angle.

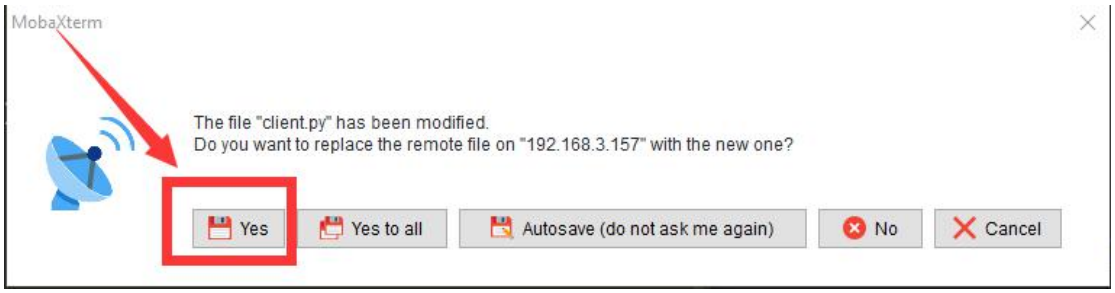
The three elements in the array are three different locations:[[90, 140, 0, 90],[-90, 140, 0, 0],[0, 140, 90, 0]], which are connected when the robot arm enters the automatic operation mode (in GUI or Web application interface, click the button 

, the robot arm enters the automatic motion mode), the path of its motion is saved in the plan.json file location point. If you want to run the new path automatically, you can manually modify the data at the corresponding location in the plan.json file and save it.



【Note】

When you click Save, you need to select "Yes".



Lesson 25 json Data Communication

In this lesson, we will learn json communication. In the example of this lesson, two sample programs are provided: server.py and client.py, which are on different devices in the same LAN. It can also be run on the same device. Run server.py first and then client.py. Server.py can send two arrays to client.py in a loop, and client.py can be used like a normal array. It is possible to send the dictionary without sending the array, and the method is the same.

25.1 Learning the code program of server.py

Here we use Subline IDE to view and edit the code program of this course. For the specific method, please see "2.4 Editing the Code Program in Raspberry Pi" in Lesson 2.

In the file manager of the MobaXterm terminal, find adeept_rasparms/CourseCode/11Json, and open the code of this lesson: server.py. The specific code and comments are as follows:

Import libraries related to TCP communication.

```
import socket
import time
import json

```

Initialize the TCP server.

```
HOST = ''
PORT = 10223                            #Define port serial
BUFSIZ = 1024                           #Define buffer size
ADDR = (HOST, PORT)
```

Establish a TCP server and monitor client connections.

```
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 server,waiting for client
print('waiting for connection...')
```

Client connection is successful.

```
tcpCliSock, addr = tcpSerSock.accept()
print('...connected from :', addr)
```

Define two arrays.

```
list_1 = [10, 20, 30, 40]
list_2 = [50, 60, 70, 80]
```

Encode array into json string.

```
    s = json.dumps(list_1)
```

Send this string.

```
    tcpCliSock.send(str(s).encode())
    time.sleep(1)

    s = json.dumps(list_2)
    tcpCliSock.send(str(s).encode())
    time.sleep(1)
```

25.2 Learning the code program of client.py

Here we use Subline IDE to view and edit the code program of this course. For the specific method, please see "2.4 Editing the Code Program in Raspberry Pi" in Lesson 2.

In the file manager of the MobaXterm terminal, find adeept_rasparms/CourseCode/11Json, and open the code of this lesson: client.py. The specific code and comments are as follows:

The IP address here needs to be changed to the IP address of the server, which is the IP address of the device where you run the client.py program. For example, if you run client.py on your computer, then you need to modify SERVER_IP to your computer's IP.

The IP address here needs to be changed to the IP address of the server.

```
SERVER_IP = "192.168.3.187"
SERVER_PORT = 10223   #Define port serial
BUFSIZ = 1024         #Define buffer size
ADDR = (SERVER_IP, SERVER_PORT)
tcpClicSock = socket(AF_INET, SOCK_STREAM) #Set connection value for socket
tcpClicSock.connect(ADDR)
```

Receive the Json string sent from the server.

```
18      data = (tcpClicSock.recv(BUFSIZ)).decode()
19
```

Decode the Json string into the corresponding format.

```
23      output = json.loads(data)
24
```

Print output, actually output can now be used as an array.

```
28      print(output)
```

25.3 How to realize json communication

1. In the file manager of the MobaXterm terminal, find adeept_rasparms/CourseCode/11Json, download the client.py file to your PC, select this folder, click the button  to download the file to the PC, and save it in English letters Under the path.

2. In the MobaXterm terminal, log in to your Raspberry Pi and use the command to enter under the adeept_rasparms/CourseCode/11Json:

cd adeept_rasparms/CourseCode/11Json

```
pi@raspberrypi:~ $ cd  adeept_rasparms/CourseCode/11Json
```

3.Enter the command to view the contents of the current directory:

ls

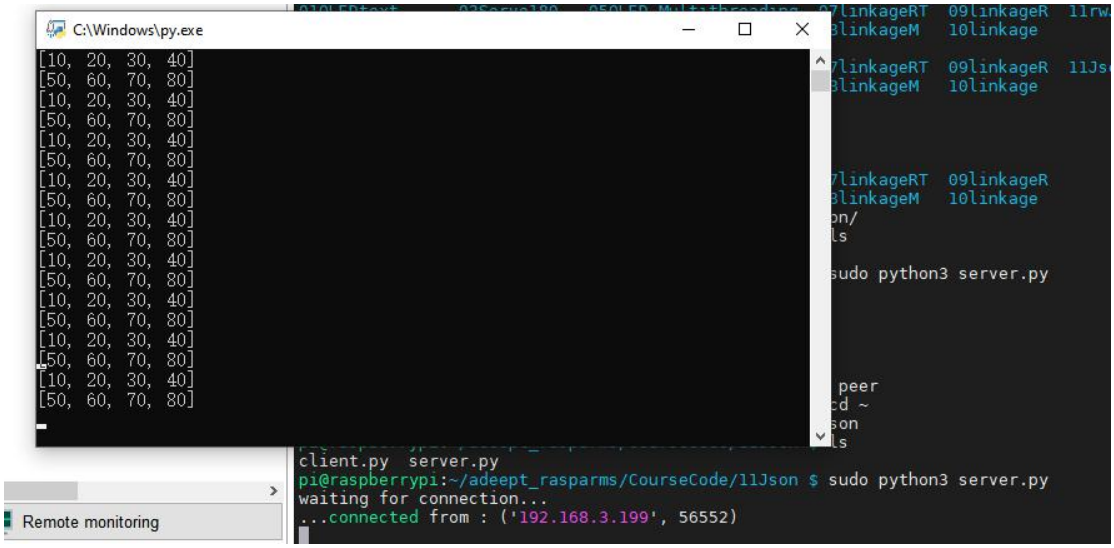
```
pi@raspberrypi:~/adeept_rasparms/CourseCode/11Json $ ls
client.py  server.py
```

4. Enter the command to run the server.py program:

sudo python3 server.py

```
pi@raspberrypi:~/adeept_rasparms/CourseCode/11Json $ sudo python3 server.py
waiting for connection...
```

5.Open the client.py program you downloaded in step 1 on the PC, and then the following data will appear.


C:\Windows\py.exe
[10, 20, 30, 40]
[50, 60, 70, 80]
[10, 20, 30, 40]
[50, 60, 70, 80]
[10, 20, 30, 40]
[50, 60, 70, 80]
[10, 20, 30, 40]
[50, 60, 70, 80]
[10, 20, 30, 40]
[50, 60, 70, 80]
[10, 20, 30, 40]
[50, 60, 70, 80]
[10, 20, 30, 40]
[50, 60, 70, 80]
[10, 20, 30, 40]
[50, 60, 70, 80]
Remote monitoring
client.py server.py
pi@raspberrypi:~/adeept_rasparms/CourseCode/11Json $ sudo python3 server.py
waiting for connection...
...connected from : ('192.168.3.199', 56552)

Lesson 26 Using a Rotary Encoder to Control a Robotic Arm

In this lesson, we will introduce how to use a rotary encoder to control a robotic arm.

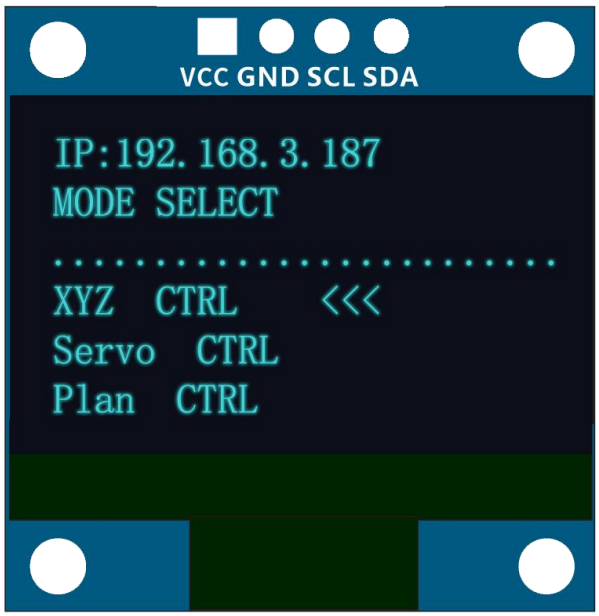
26.1 About the rotary encoder

Rotary encoder is a device used to measure the speed and cooperate with PWM technology to achieve rapid speed adjustment. By photoelectric conversion, the photoelectric rotary encoder can convert the angular displacement and angular velocity of the output shaft into corresponding electrical pulses for digital output (REP). In this lesson, the rotary encoder we use mainly acts as a switch button. With this feature of the rotary encoder, you can use it to control the robotic arm.

26.2 How to use a rotary encoder to control a robotic arm

1. First, you need to supply power to the robotic arm, and the OLED screen is lit (if the screen is not lit, then you need to ensure that you have successfully installed the dependent libraries according to the content of 2.2), you will see the first level

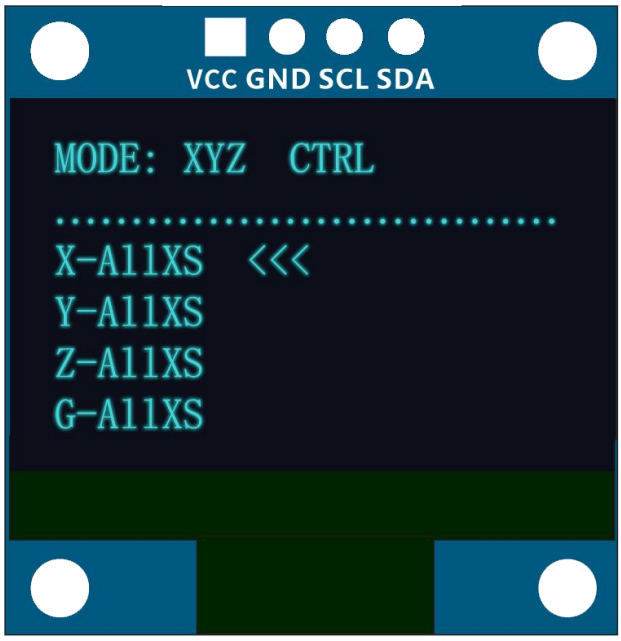
directory:



2. In the first-level directory, you will see three options. When you want to enter the first option, you can rotate it by operating the rotary encoder.

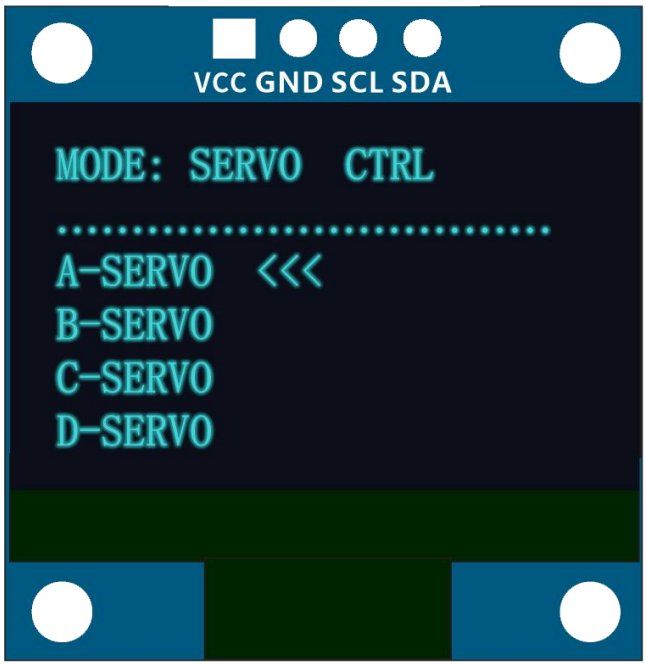
3. How to enter "XYZ CTRL", when "<<<" is marked in the first option, you can press the rotary encoder to enter the first option, as shown below:

When "<<<" is marked in the first item: X-ALLXS, press the rotary encoder at this time to turn on the mode of controlling the end point of the robotic arm to move in the X-axis direction. Then you can control the robotic arm by the rotary encoder.



4. How to return to the previous or first level main catalog, you can return to the previous or first main catalog by long pressing the rotary encoder.

5. After returning to the main menu, you can use the rotary encoder. When "<<<" is marked in the second option, you can press the rotary encoder to enter the second option: SERVO CTRL, this mode controls the A, B, C, D servos on the robotic arm structure respectively. You can select A-SERVO by pressing, and the rotary encoder controls the movement of the A servo, as shown in the following figure:



Adeept

STEM Education Products and Service Provider

Shenzhen Adeept Technology Co.,Ltd.

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