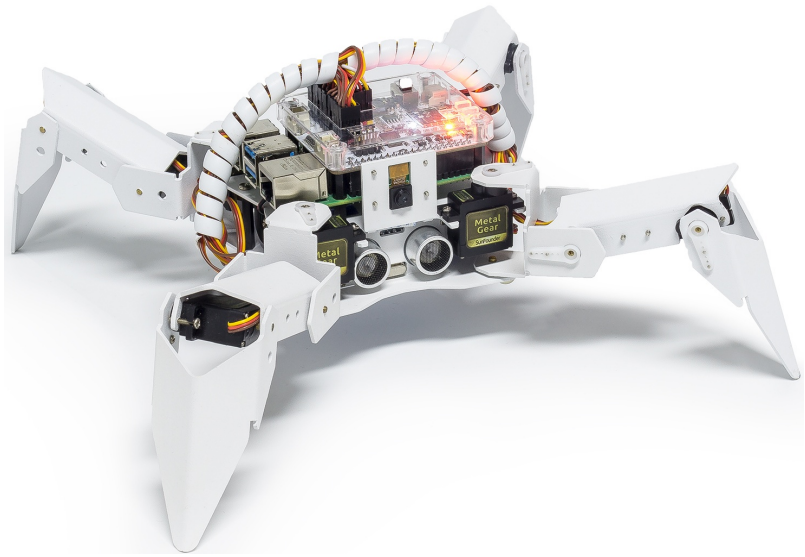
SunFounder PiCrawler Kit

www.sunfounder.com

Aug 02, 2022

CONTENTS

Thank you for choosing our PiCrawler.



PiCrawler is a Raspberry Pi quadruped robot with aluminum alloy structure. It is equipped with a camera module, which can perform color recognition, face detection and other items; 12 metal gear servos support it to walk, dance, and pose various postures; the ultrasonic module on the body allows it to quickly detect obstacles in front of it; the expansion board-robot HAT is equipped with a speaker, allowing it to express emotions such as happiness and excitement.

This document includes the list and assembly pdf, Robot HAT introduction and PiCrawler programming.

The programming part is divided into two chapters: *Play with Ezblock* & *Play with Python*, each chapter allows you to explain how to make PiCrawler work the way you want.

Ezblock Studio is a development platform developed by SunFounder for beginners, aiming to lower the barriers to entry for Raspberry Pi. It has two programming languages: Graphical and Python, which can be used on almost all different types of devices. With Bluetooth and Wi-Fi support, you can download codes on Ezblock Studio and remotely control Raspberry Pi.

More experienced makers can use the popular programming language-Python.

Content

CHAPTER

ONE

COMPONENT LIST AND ASSEMBLY INSTRUCTIONS

You need to check whether there are missing or damaged components according to the list first. If there are any problems, please contact us and we will solve them as soon as possible.

Please follow the steps on the PDF to assemble.

If the servo has been powered on, please do not turn the Servo shaft to avoid damage.

Note:

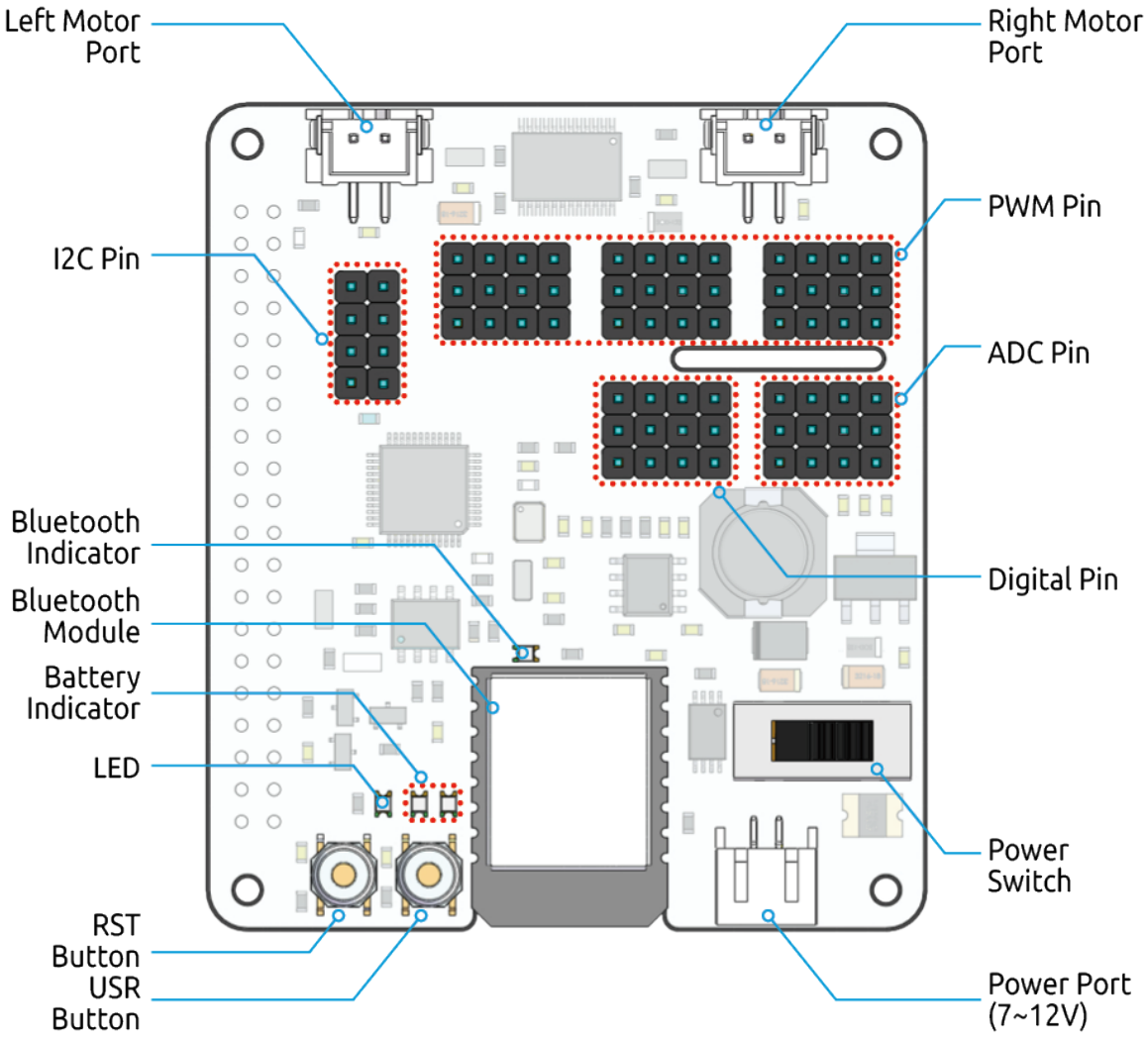
1. Before assembling, you need to buy 2 18650 batteries and fully charge them, refer to *About the Battery*.
2. Robot HAT cannot charge the battery, so you need to buy a battery charger at the same time.

- `Component List and Assembly Instructions.`

CHAPTER

TWO

ABOUT ROBOT HAT



Robot HAT is a multifunctional expansion board that allows Raspberry Pi to be quickly turned into a robot. An MCU is on board to extend the PWM output and ADC input for the Raspberry Pi, as well as a motor driver chip, Bluetooth module, I2S audio module and mono speaker. As well as the GPIOs that lead out of the Raspberry Pi itself.

It also comes with a Speaker, which can be used to play background music, sound effects and implement TTS functions to make your project more interesting.

Accepts 7-12V PH2.0 2pin power input with 2 power indicators. The board also has a user available LED and a button for you to quickly test some effects.

Note: You can see more details in the Robot HAT Documentation.

CHAPTER

THREE

PLAY WITH PYTHON

If you want to program in python, then you will need to learn some basic Python programming skills and basic knowledge of Raspberry Pi, please configure the Raspberry Pi first according to *Quick Guide on Python*.

3.1 Quick Guide on Python

This section is to teach you how to install Raspberry Pi OS, configure wifi to Raspberry Pi, remote access to Raspberry Pi to run the corresponding code.

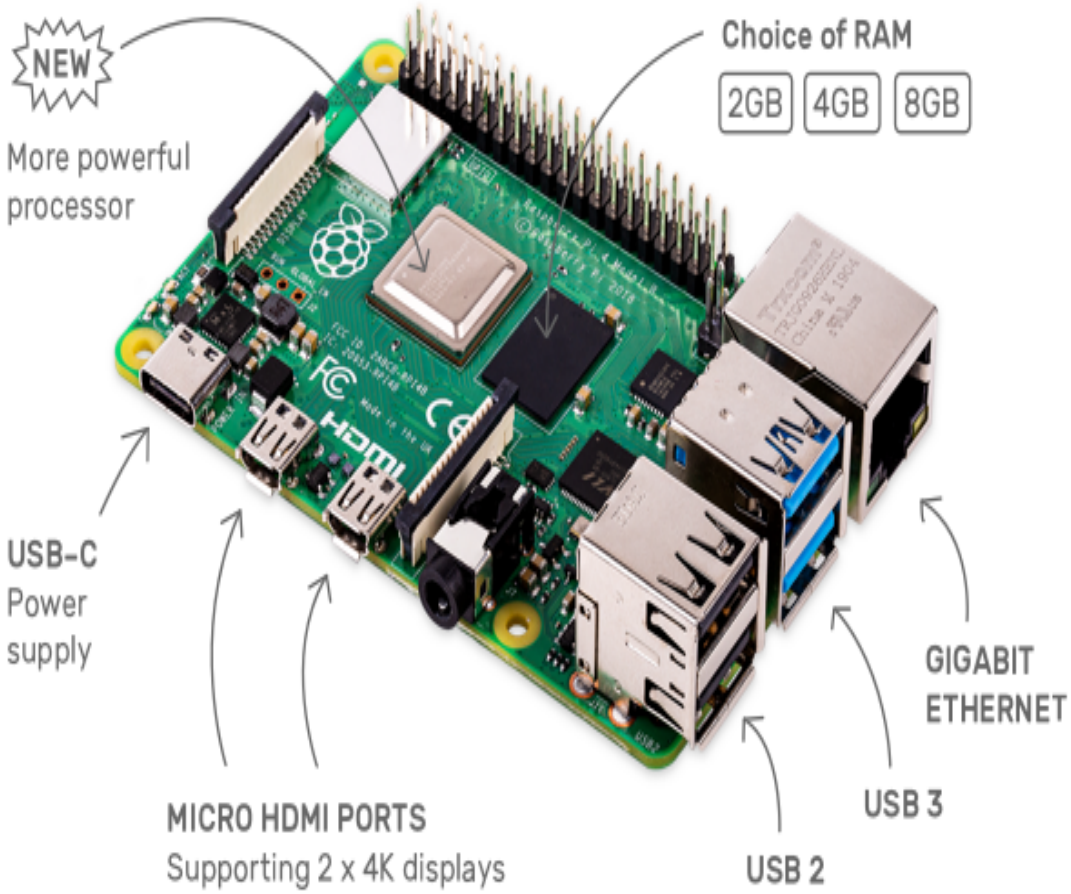
If you are familiar with Raspberry Pi and can open the command line successfully, then you can skip the first 3 parts and then complete the last part.

3.1.1 What Do We Need?

Required Components

Raspberry Pi

The Raspberry Pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. It is a capable little device that enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python.



Power Adapter

To connect to a power socket, the Raspberry Pi has a micro USB port (the same found on many mobile phones). You will need a power supply which provides at least 2.5 amps.

Micro SD Card

Your Raspberry Pi needs an Micro SD card to store all its files and the Raspberry Pi OS. You will need a micro SD card with a capacity of at least 8 GB

Optional Components

Screen

To view the desktop environment of Raspberry Pi, you need to use the screen that can be a TV screen or a computer monitor. If the screen has built-in speakers, the Pi plays sounds via them.

Mouse & Keyboard

When you use a screen , a USB keyboard and a USB mouse are also needed.

HDMI

The Raspberry Pi has a HDMI output port that is compatible with the HDMI ports of most modern TV and computer monitors. If your screen has only DVI or VGA ports, you will need to use the appropriate conversion line.

Case

You can put the Raspberry Pi in a case; by this means, you can protect your device.

Sound or Earphone

The Raspberry Pi is equipped with an audio port about 3.5 mm that can be used when your screen has no built-in speakers or when there is no screen operation.

3.1.2 Installing the OS

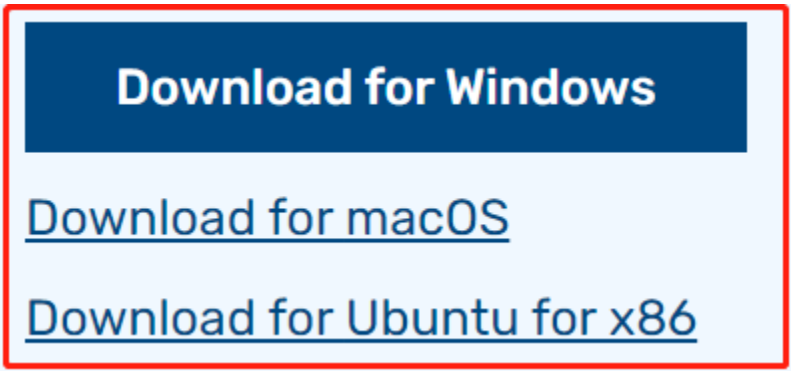
Required Components

Any Raspberry Pi	1 * Personal Computer
1 * Micro SD card	

Step 1

Raspberry Pi have developed a graphical SD card writing tool that works on Mac OS, Ubuntu 18.04 and Windows, and is the easiest option for most users as it will download the image and install it automatically to the SD card.

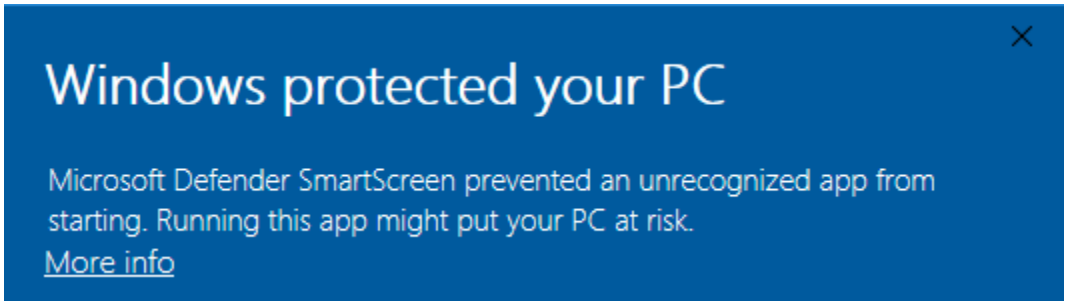
Visit the download page: https://www.raspberrypi.org/software/. Click on the link for the Raspberry Pi Imager that matches your operating system, when the download finishes, click it to launch the installer.



Step 2

When you launch the installer, your operating system may try to block you from running it. For example, on Windows I receive the following message:

If this pops up, click on **More info** and then **Run anyway**, then follow the instructions to install the Raspberry Pi Imager.



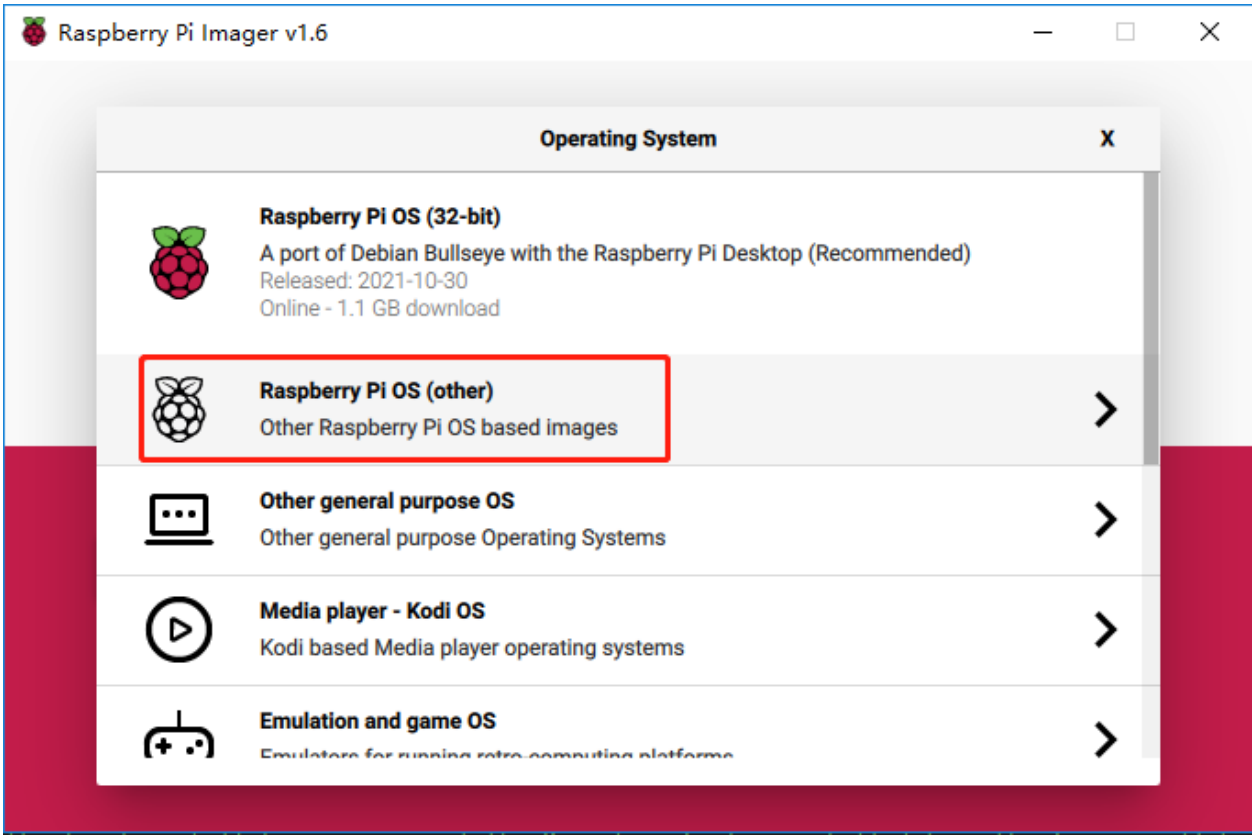
Step 3

Insert your SD card into the computer or laptop SD card slot.

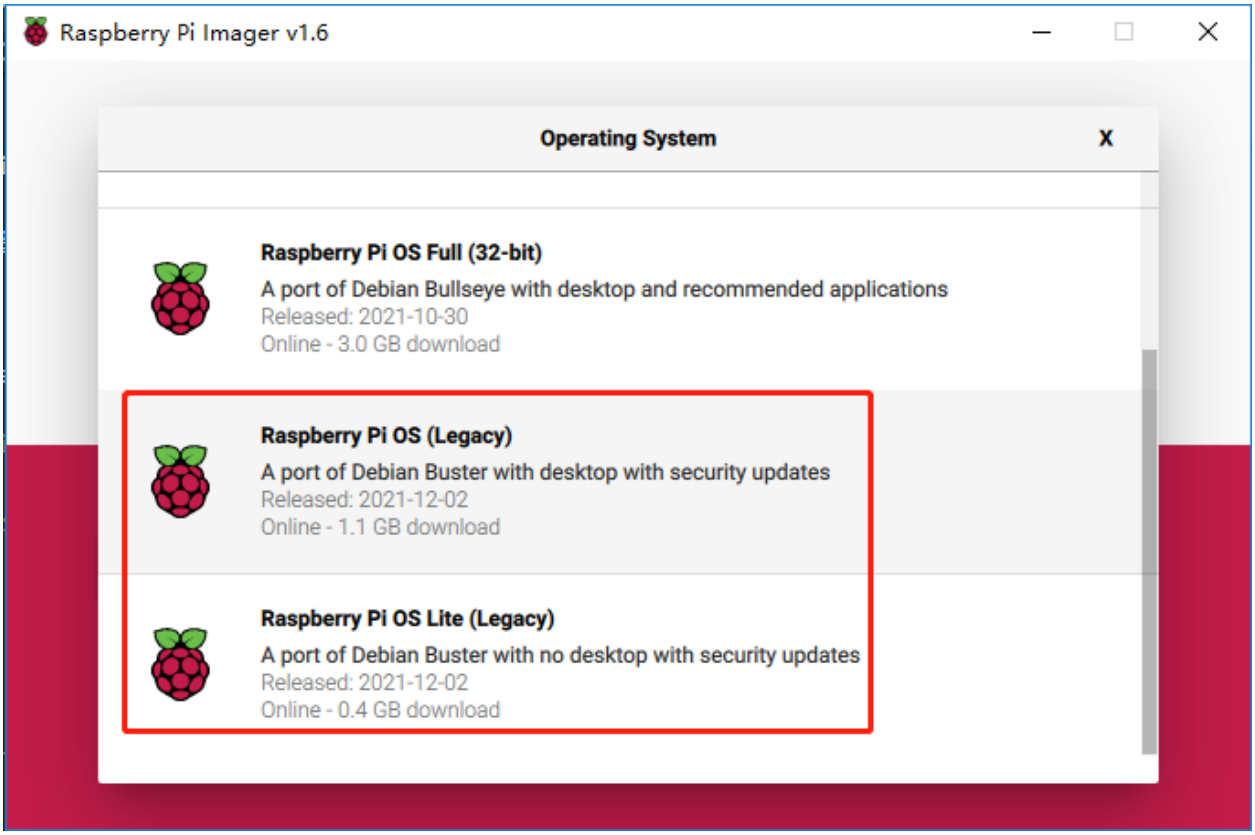
Step 4

Warning: Upgrading the Raspberry Pi OS to **Debian Bullseye** will cause some features to not work, so it is recommended to continue using the **Debian Buster** version.

In the Raspberry Pi Imager, click **CHOOSE OS** -> **Raspberry Pi OS(other)**.

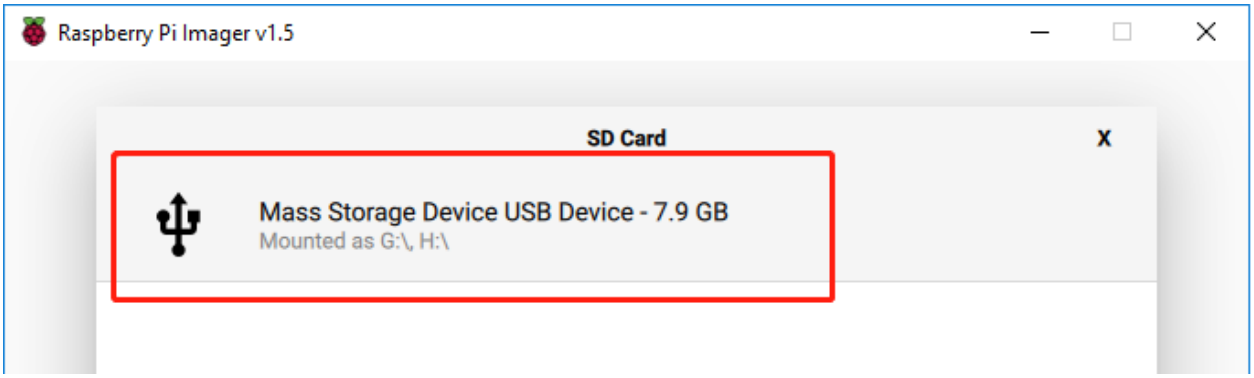


Scroll down to the end of the newly opened page and you will see **Raspberry Pi OS(Legacy)** and **Raspberry Pi OS Lite(Legacy)**, these are security updates for Debian Buster, the difference between them is with or without the desktop. It is recommended to install **Raspberry Pi OS(Legacy)**, the system with the desktop.



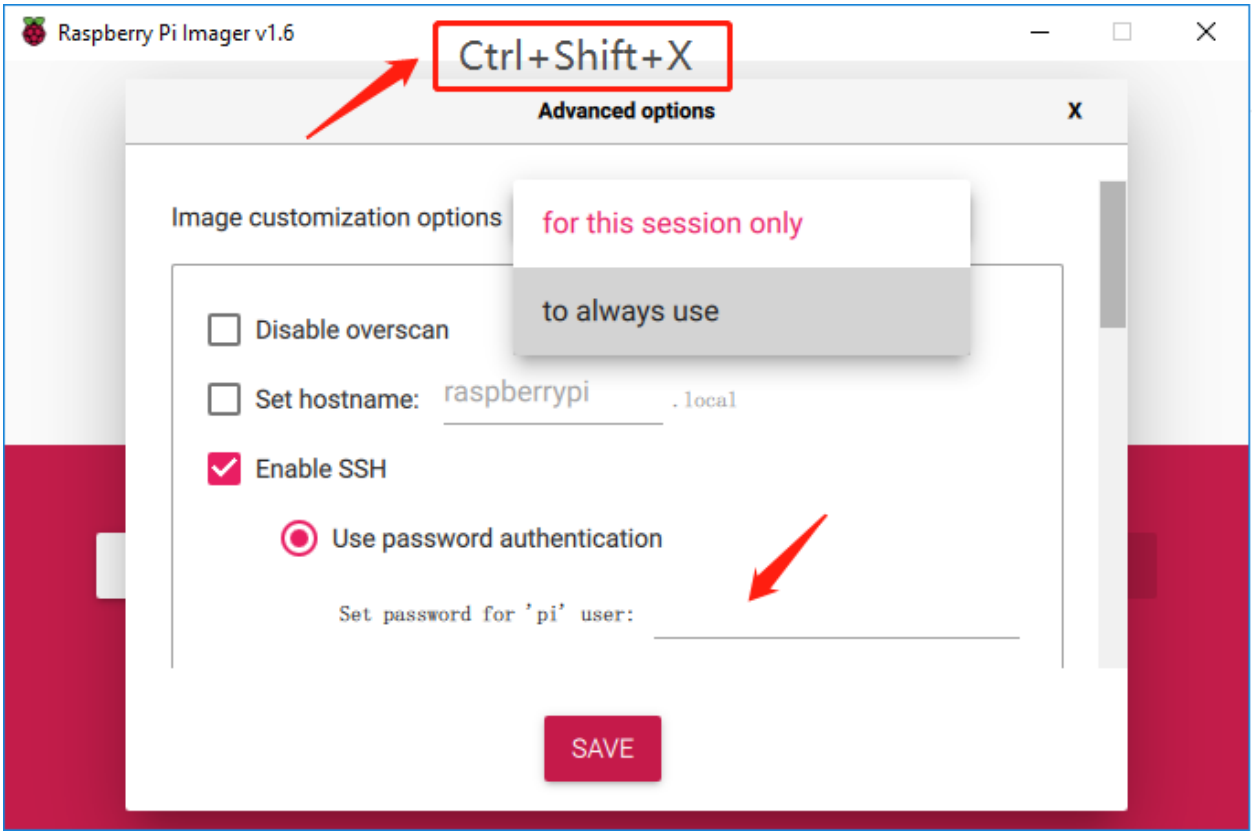
Step 5

Select the SD card you are using.



Step 6

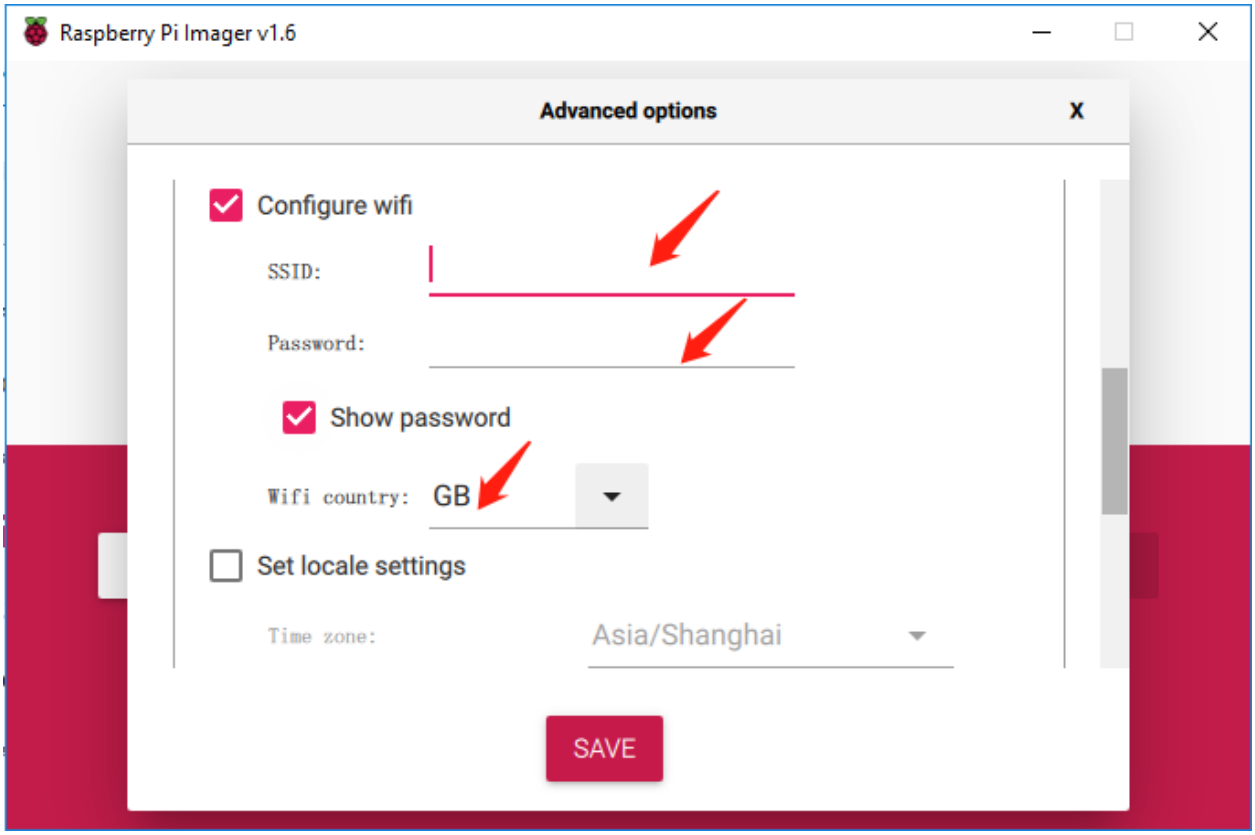
Press **Ctrl+Shift+X** to open the **Advanced options** page to enable SSH and configure wifi, these 2 items must be set, the others depend on your choice . You can choose to always use this image customization options.



Then scroll down to complete the wifi configuration and click **SAVE**.

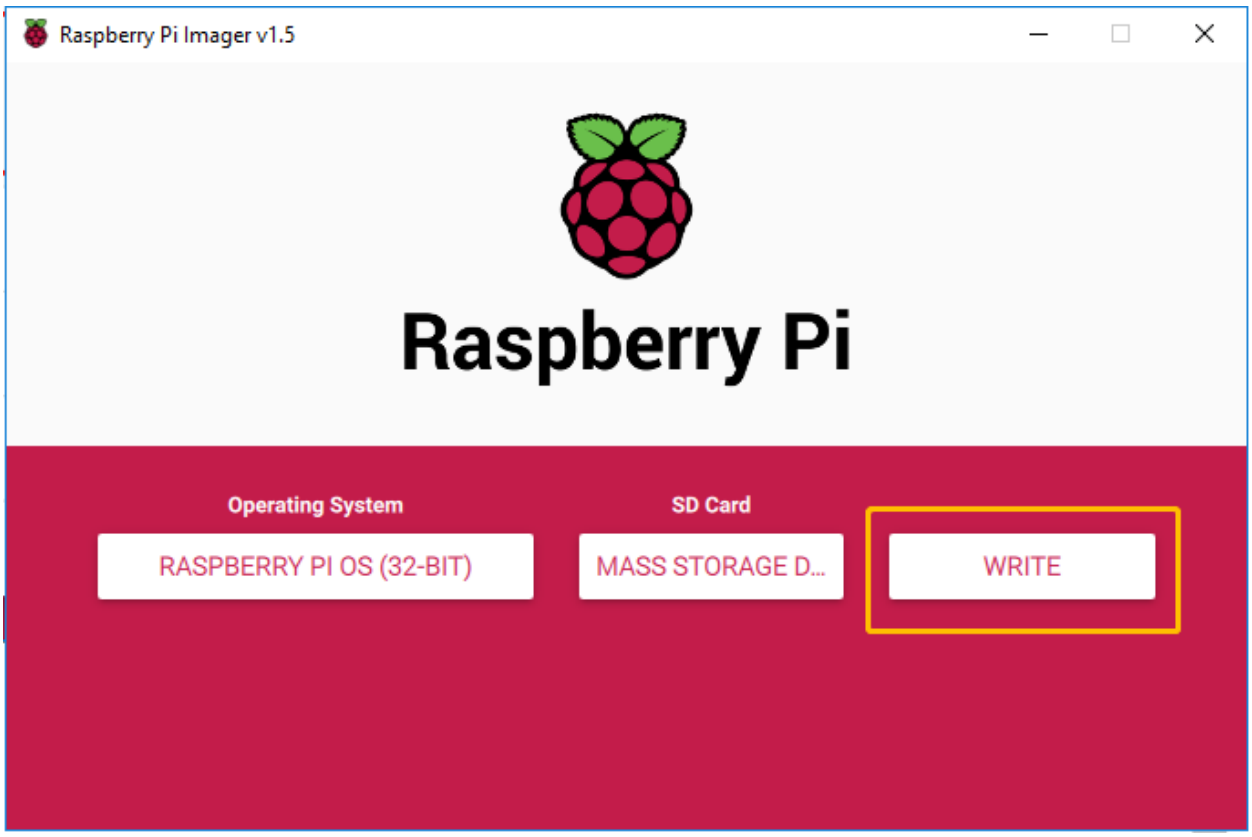
Note: **wifi country** should be set the two-letter ISO/IEC alpha2 code for the country in which you are using your Raspberry Pi, please refer to the following link: https://en.wikipedia.org/wiki/ISO_3166-1_alpha-2#Officially_assigned_code_elements

---



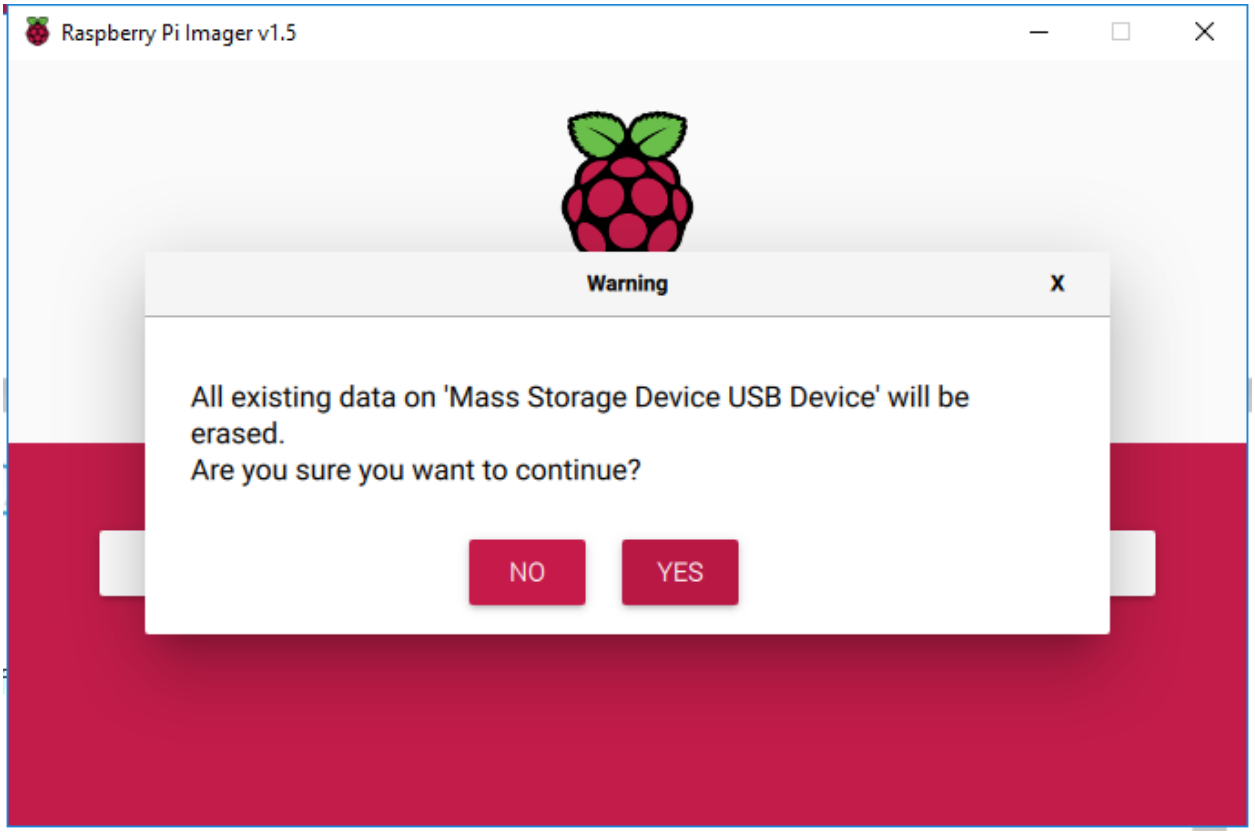
Step 7

Click the **WRITE** button.



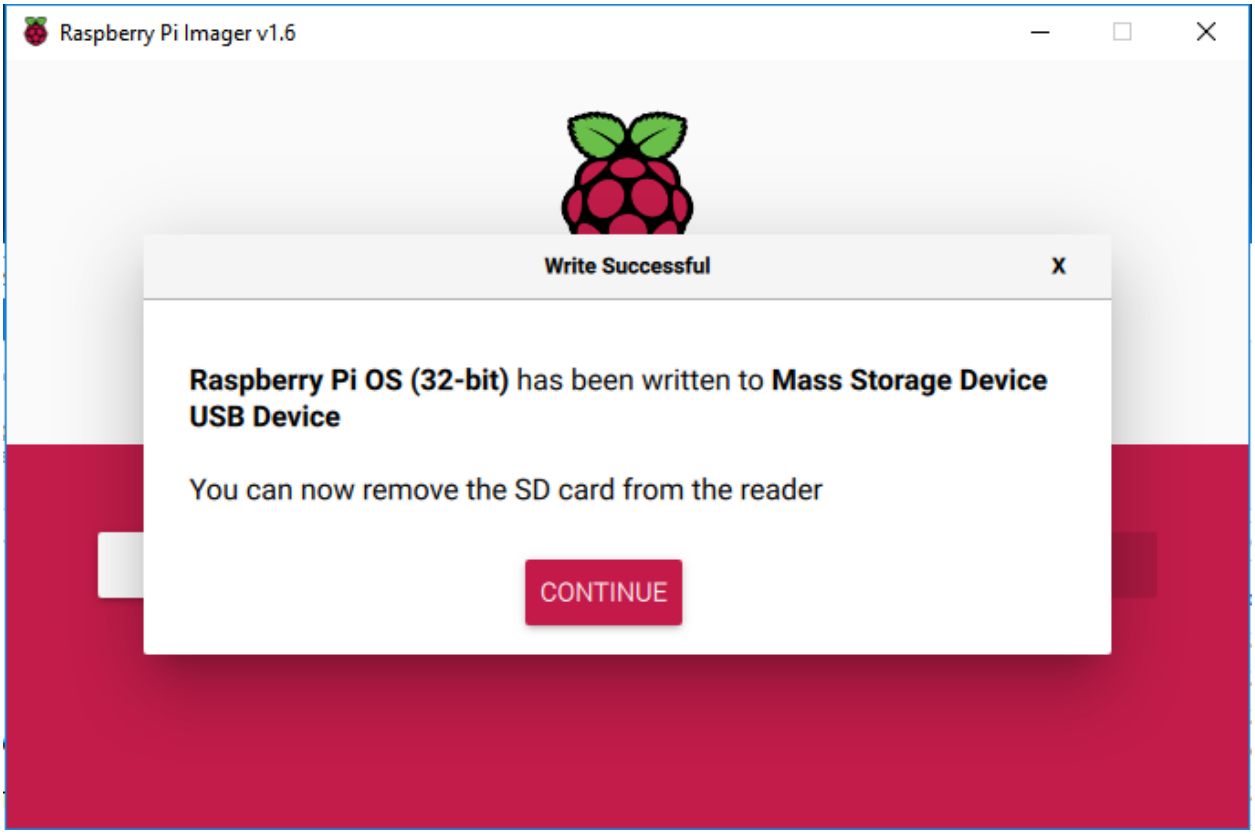
Step 8

If your SD card currently has any files on it, you may wish to back up these files first to prevent you from permanently losing them. If there is no file to be backed up, click **Yes**.



Step 9

After waiting for a period of time, the following window will appear to represent the completion of writing.



3.1.3 Set up Your Raspberry Pi

If You Have a Screen

If you have a screen, it will be easy for you to operate on the Raspberry Pi.

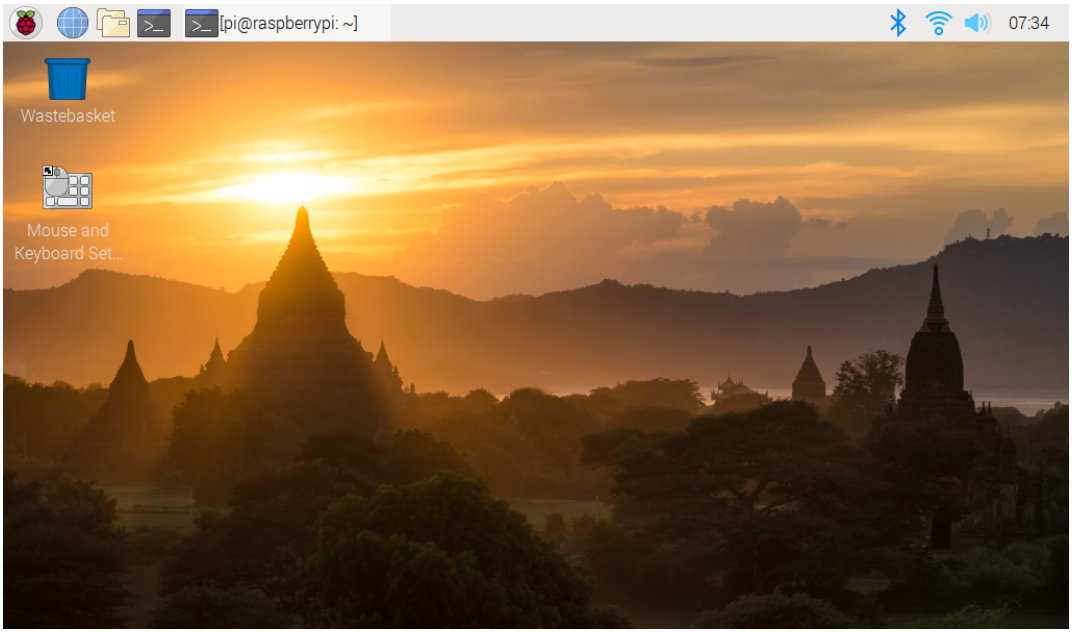
Required Components

Any Raspberry Pi	1 * Power Adapter
1 * Micro SD card	1 * Screen Power Adapter
1 * HDMI cable	1 * Screen
1 * Mouse	1 * Keyboard

1. Insert the SD card you've set up with Raspberry Pi OS into the micro SD card slot on the underside of your Raspberry Pi.
2. Plug in the Mouse and Keyboard.
3. Connect the screen to Raspberry Pi's HDMI port and make sure your screen is plugged into a wall socket and switched on.

   Note: If you use a Raspberry Pi 4, you need to connect the screen to the HDMI0 (nearest the power in port).

4. Use the power adapter to power the Raspberry Pi. After a few seconds, the Raspberry Pi OS desktop will be displayed.



If You Have No Screen

If you don't have a display, you can log in to the Raspberry Pi remotely, but before that, you need to get the IP of the Raspberry Pi.

Get the IP Address

After the Raspberry Pi is connected to WIFI, we need to get the IP address of it. There are many ways to know the IP address, and two of them are listed as follows.

1. Checking via the router

If you have permission to log in the router(such as a home network), you can check the addresses assigned to Raspberry Pi on the admin interface of router.

The default hostname of the Raspberry Pi OS is **raspberrypi**, and you need to find it. (If you are using ArchLinuxARM system, please find alarmpi.)

2. Network Segment Scanning

You can also use network scanning to look up the IP address of Raspberry Pi. You can apply the software, **Advanced IP scanner** and so on.

Scan the IP range set, and the name of all connected devices will be displayed. Similarly, the default hostname of the Raspberry Pi OS is **raspberrypi**, if you haven't modified it.

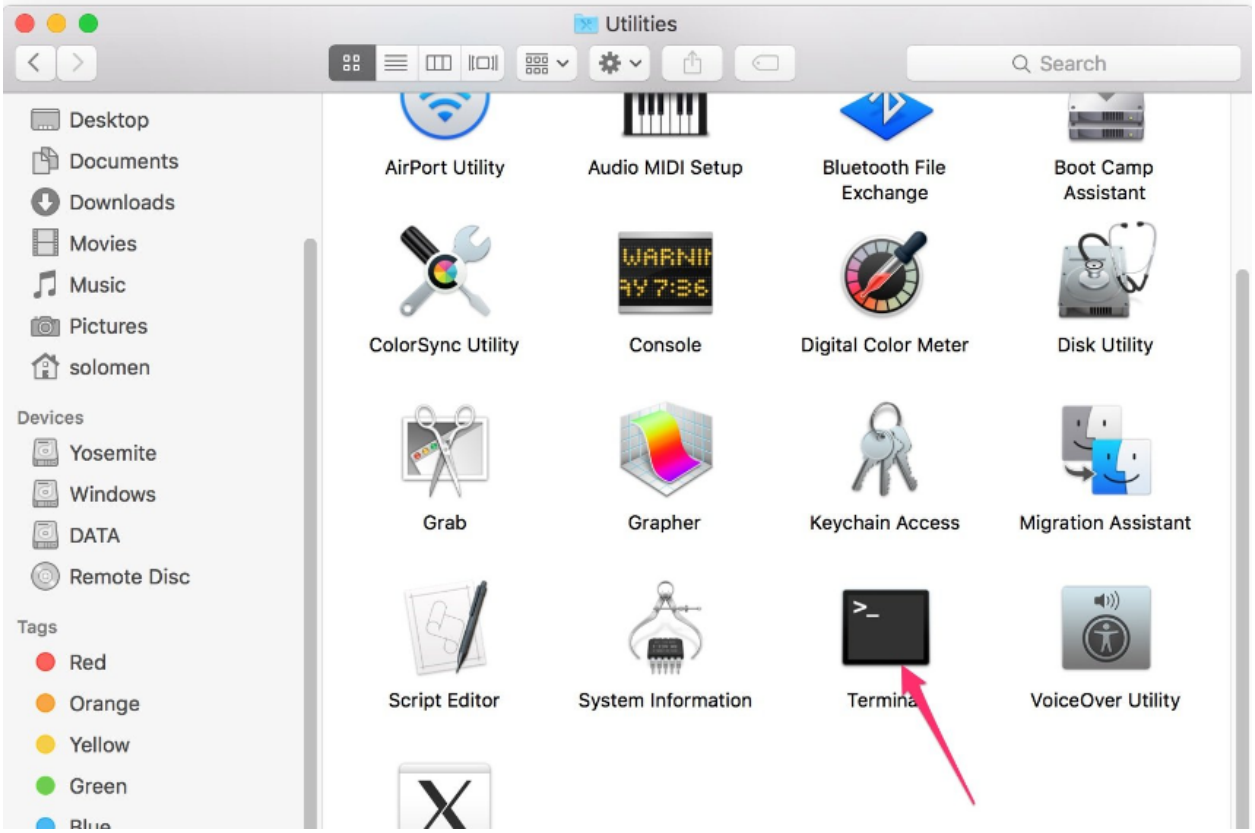
Use the SSH Remote Control

We can open the Bash Shell of Raspberry Pi by applying SSH. Bash is the standard default shell of Linux. The Shell itself is a program written in C that is the bridge linking the customers and Unix/Linux. Moreover, it can help to complete most of the work needed.

For Linux or/Mac OS X Users

Step 1

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



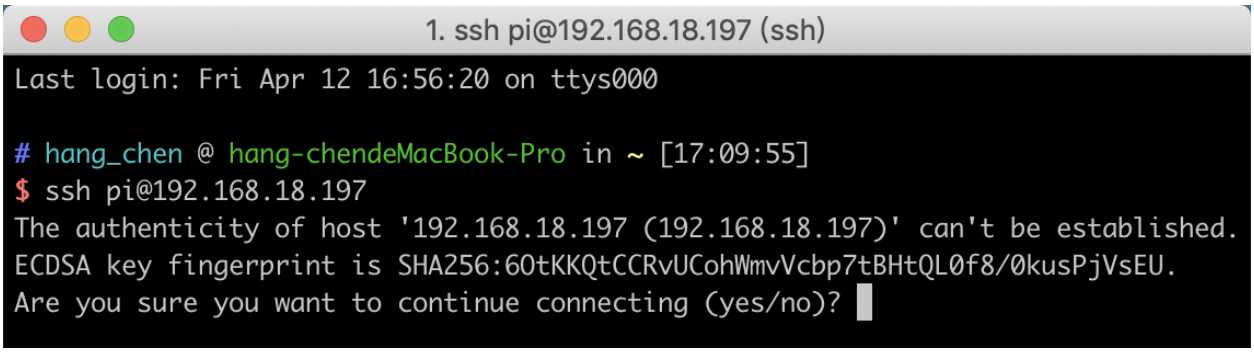
Step 2

Type in **ssh pi@ip_address** . “pi”is your username and “ip_address” is your IP address. For example:

```
ssh pi@192.168.18.197
```

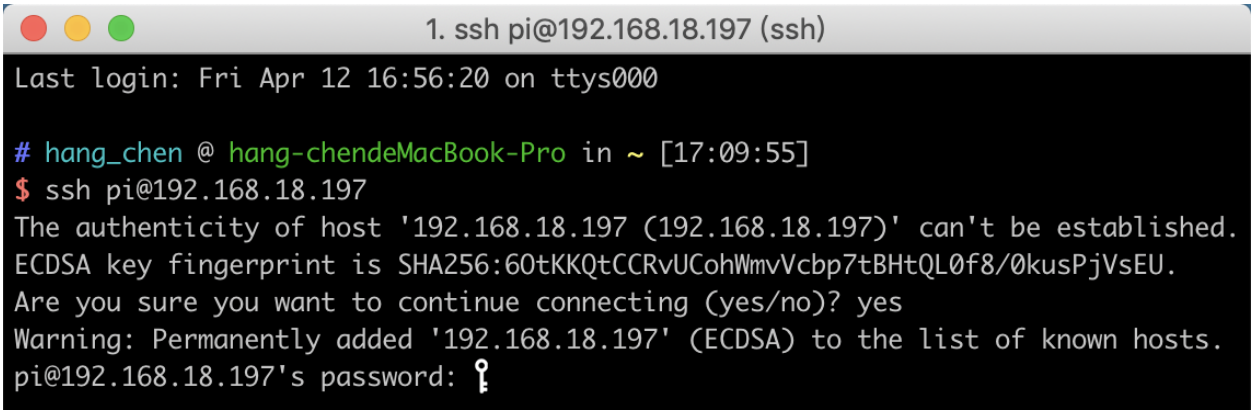
Step 3

Input”yes”.



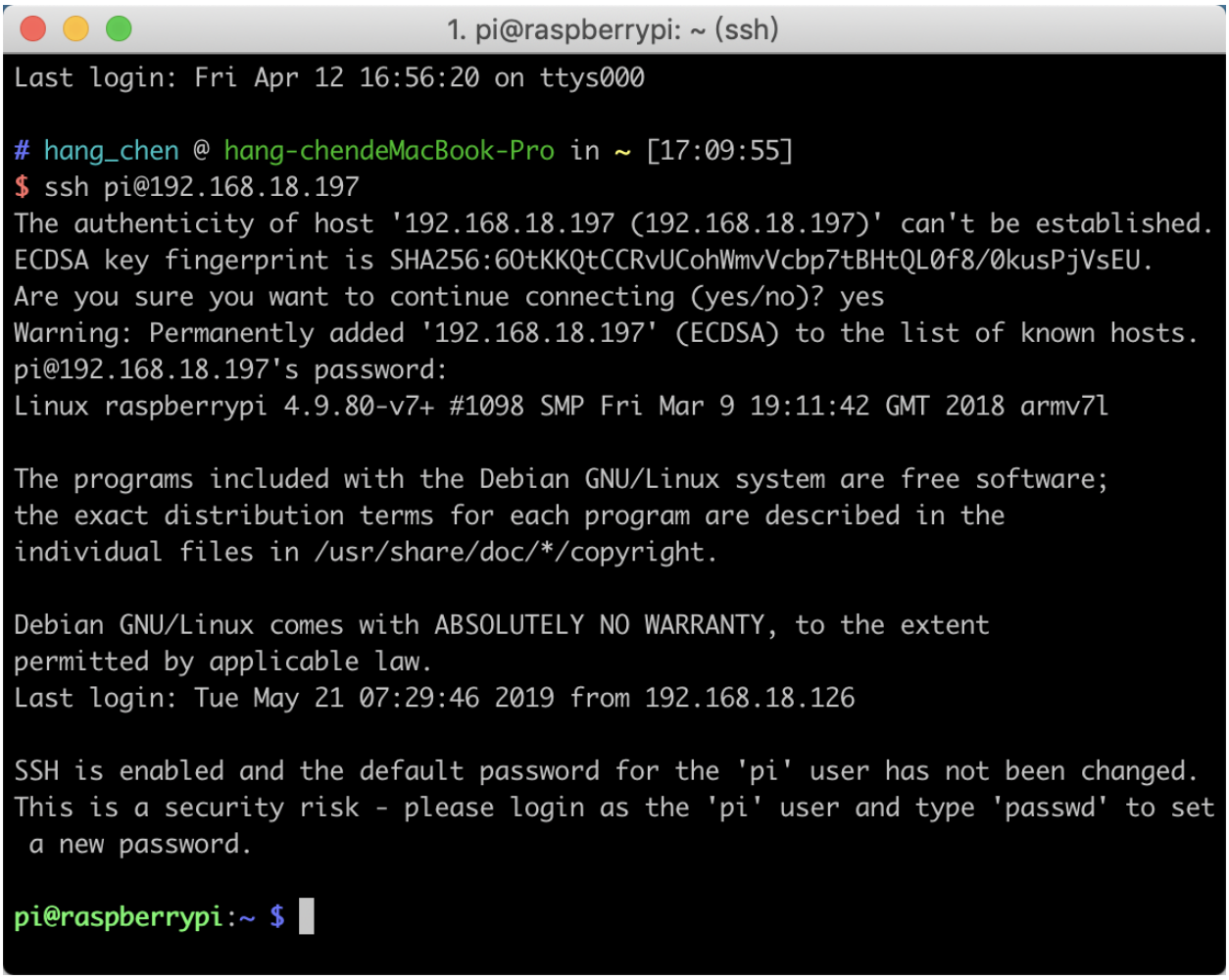
Step 4

Input the passcode and the default password is **raspberry**.



Step 5

We now get the Raspberry Pi connected and are ready to go to the next step.



Note: When you input the password, the characters do not display on window accordingly, which is normal. What you need is to input the correct password.

For Windows Users

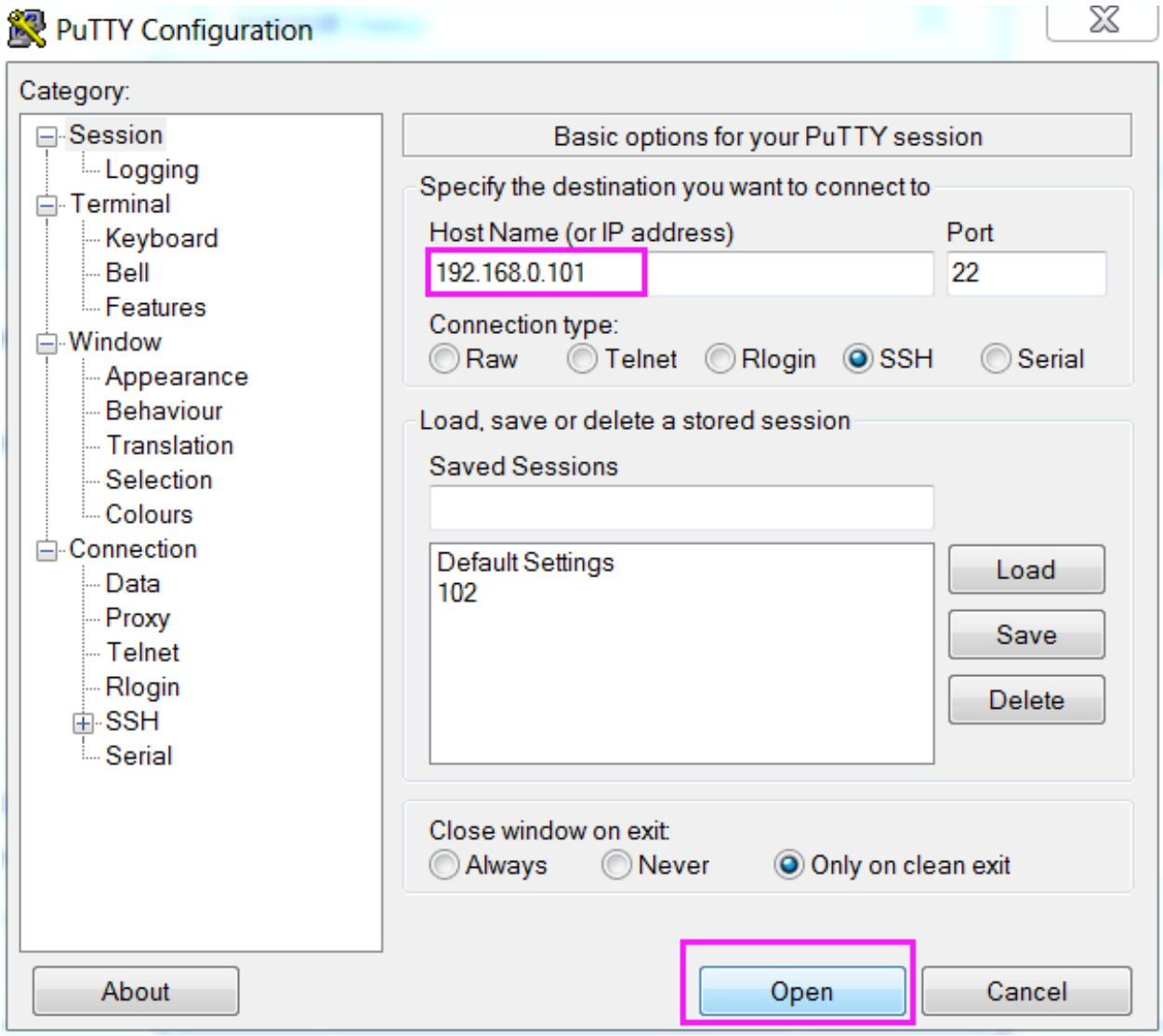
If you're a Windows user, you can use SSH with the application of some software. Here, we recommend **PuTTY**.

Step 1

Download PuTTY.

Step 2

Open PuTTY and click **Session** on the left tree-alike structure. Enter the IP address of the RPi in the text box under **Host Name (or IP address)** and **22** under **Port** (by default it is 22).



Step 3

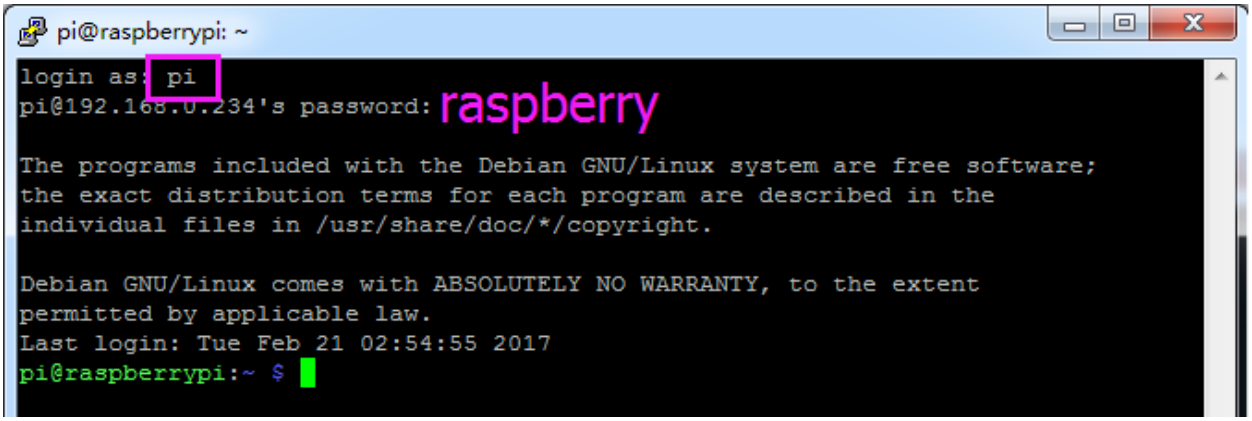
Click **Open**. Note that when you first log in to the Raspberry Pi with the IP address, there prompts a security reminder. Just click **Yes**.

Step 4

When the PuTTY window prompts "**login as:"**, type in "**pi"**(the user name of the RPi), and **password: "**raspberry" (the default one, if you haven't changed it).

Note: When you input the password, the characters do not display on window accordingly, which is normal. What you need is to input the correct password.

If inactive appears next to PuTTY, it means that the connection has been broken and needs to be reconnected.



Step 5

Here, we get the Raspberry Pi connected and it is time to conduct the next steps.

Note: If you are not satisfied with using the command window to control the Raspberry Pi, you can also use the remote desktop function, which can help us manage the files in the Raspberry Pi easily.

For details on how to do this, please refer to *Remote Desktop*.

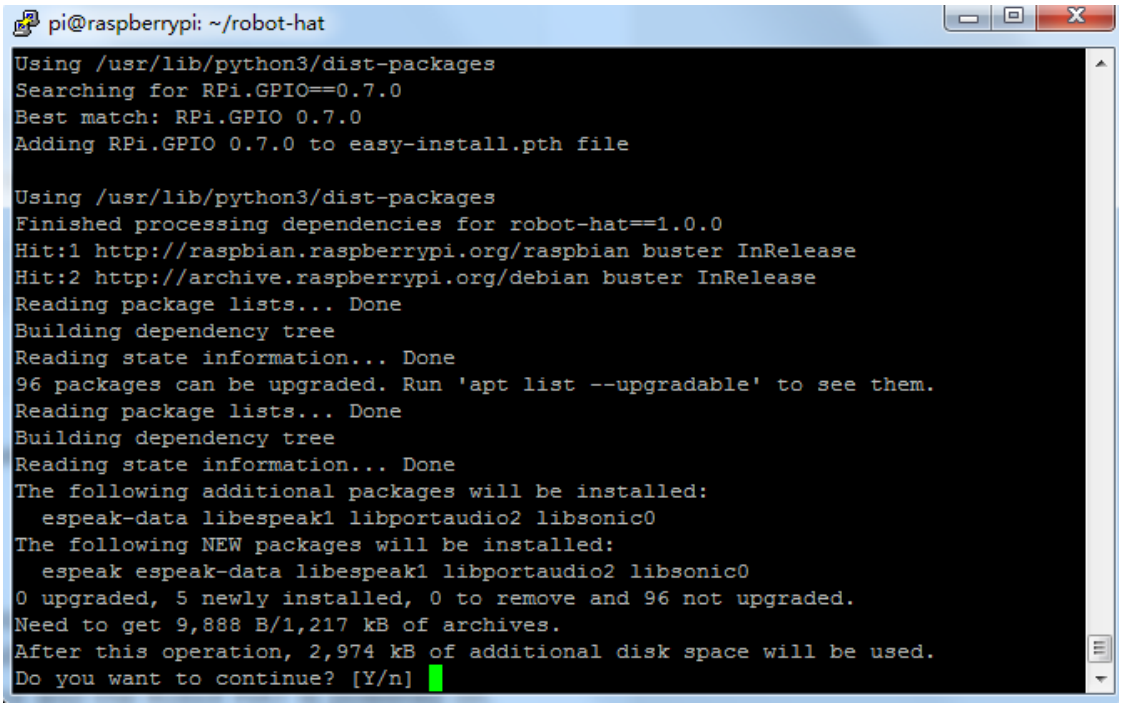
3.1.4 Download and Run the Code

We can download the files by using `git clone` in the command line.

Install `robot-hat` module first.

```
cd /home/pi/
git clone https://github.com/sunfounder/robot-hat.git
cd robot-hat
sudo python3 setup.py install
```

Note: Running `setup.py` will download some necessary components. You may fail to download due to network problems. You may need to download again at this time. In the following cases, enter `Y` and press Enter.



Then download the code and install `vilib` module.

```
cd /home/pi/
git clone https://github.com/sunfounder/vilib.git
cd vilib
sudo python3 install.py
```

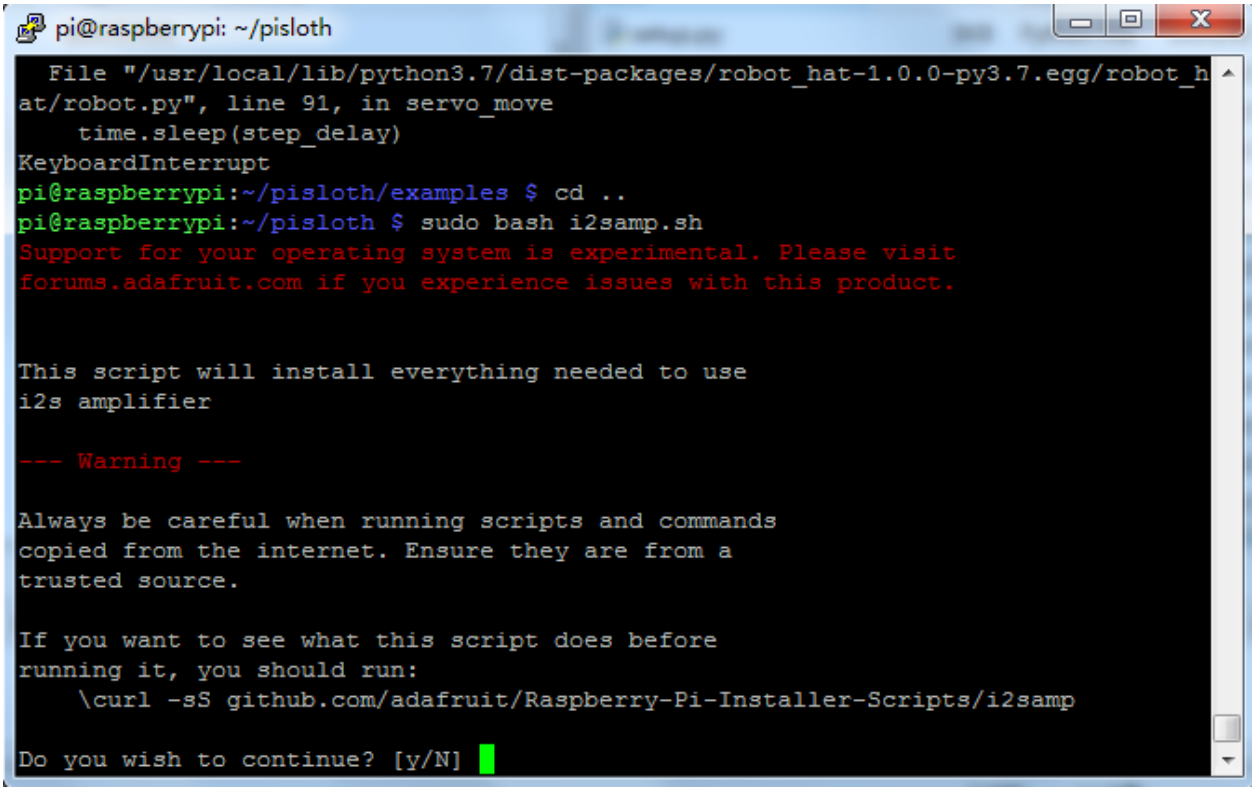
Then download the code and install `picrawler` module.

```
cd /home/pi/
git clone -b v2.0 https://github.com/sunfounder/picrawler.git
cd picrawler
sudo python3 setup.py install
```

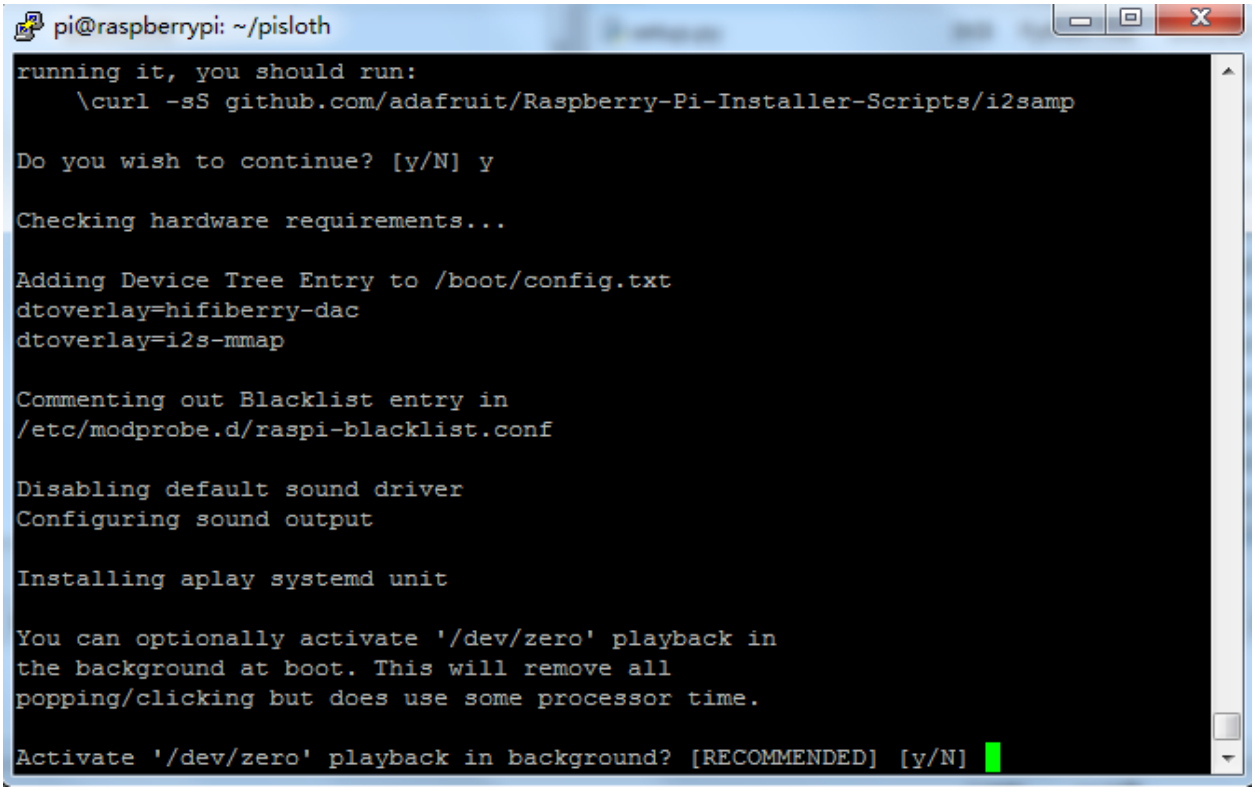
This step will take a little time, so please be patient.

Finally, you need to run the script `i2samp.sh` to install the components required by the i2s amplifier, otherwise the pislot will have no sound.

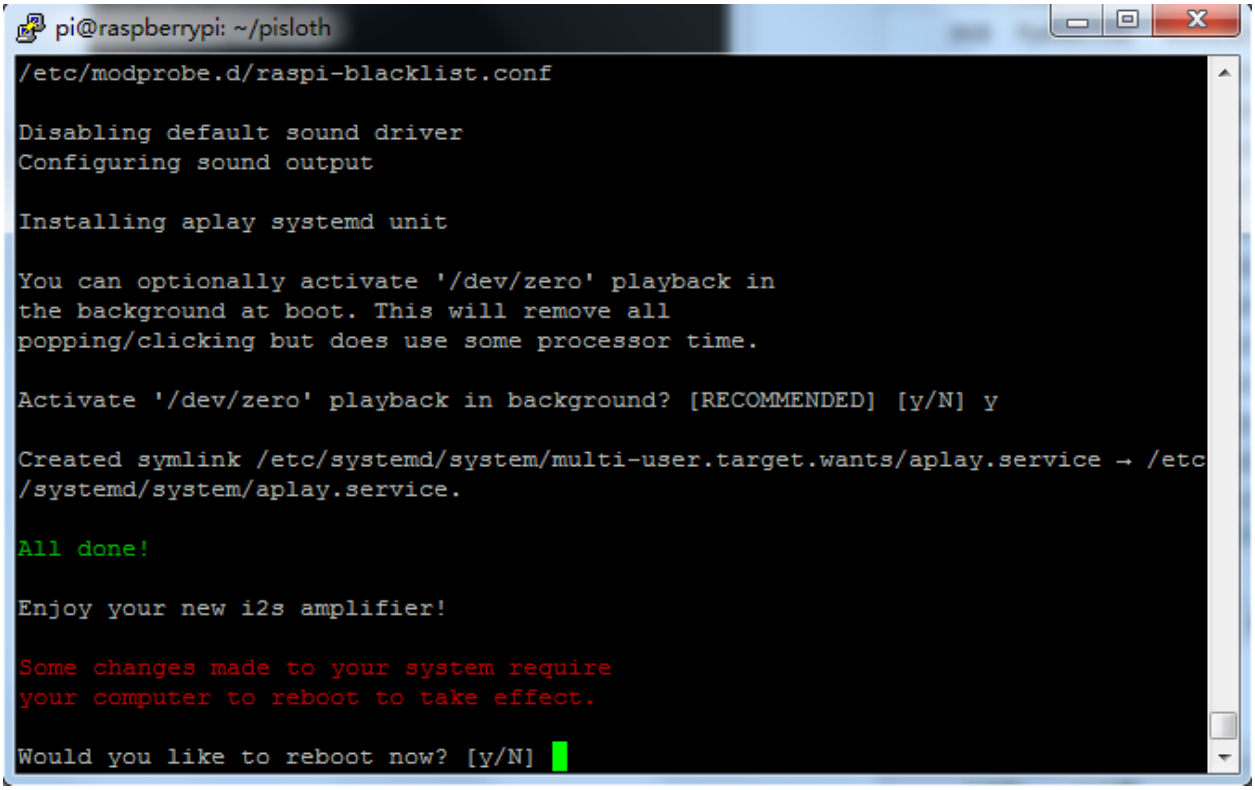
```
cd /home/pi/picrawler
sudo bash i2samp.sh
```



Type `y` and press `Enter` to continue running the script.



Type `y` and press `Enter` to run `/dev/zero` in the background.

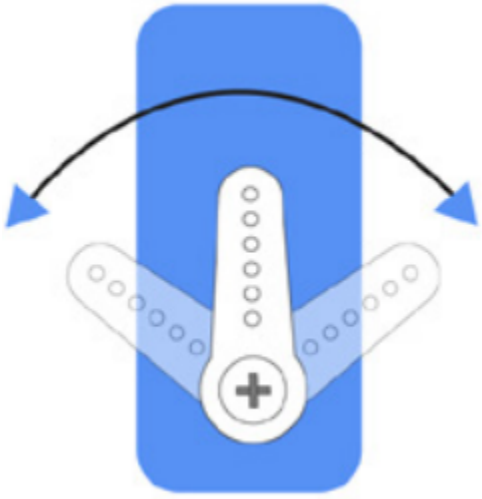


Type `y` and press `Enter` to restart the machine.

Note: If there is no sound after restarting, you may need to run the `i2samp.sh` script multiple times.

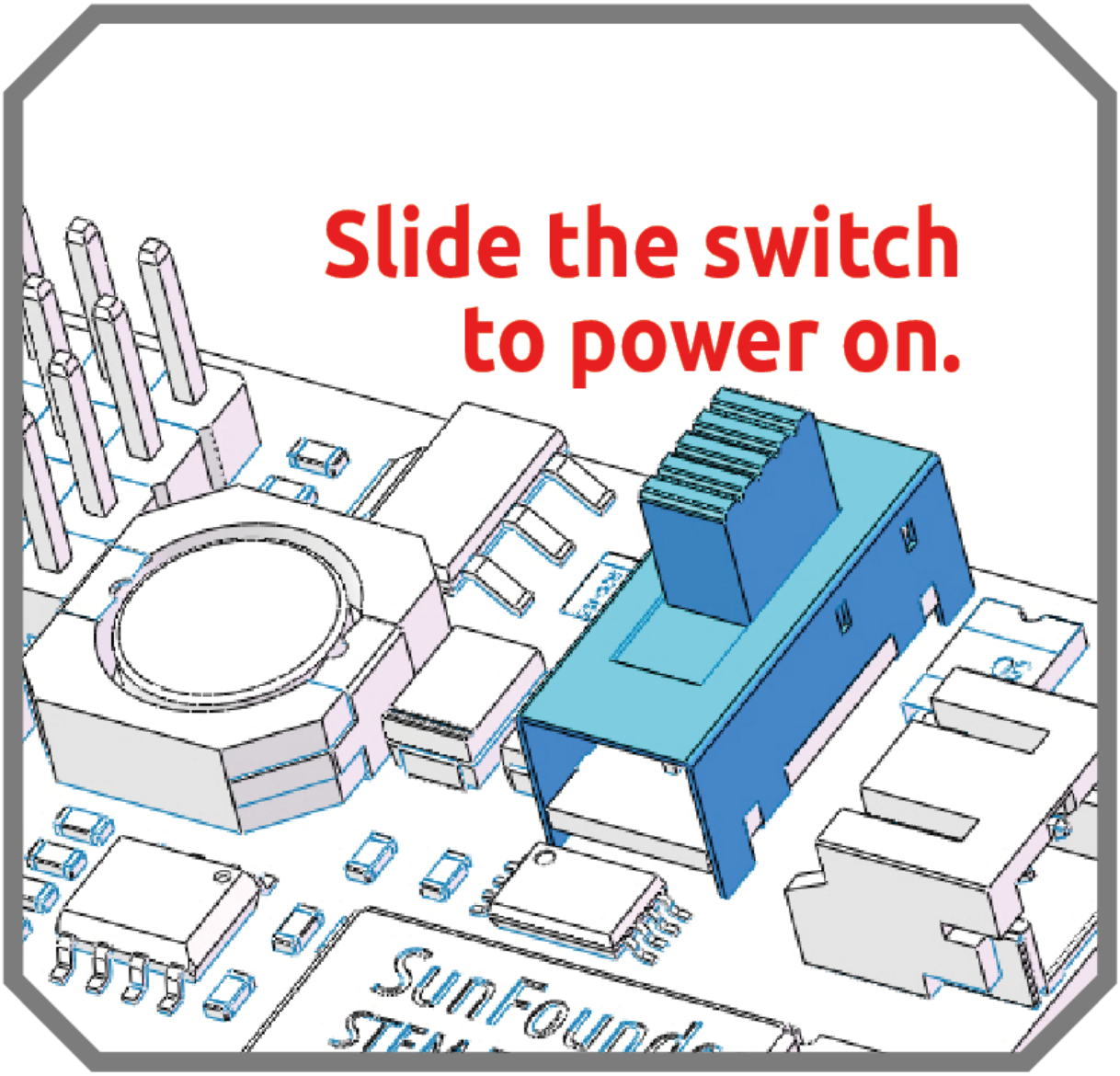
3.1.5 Servo Adjust

To ensure that the servo has been properly set to 0°, first insert the rocker arm into the servo shaft and then gently rotate the rocker arm to a different angle.



Follow the instructions on the assembly foldout, insert the battery holder cable and turn the power switch to the ON.

Wait for 1-2 minutes, there will be a sound to indicate that the Raspberry Pi boots successfully.

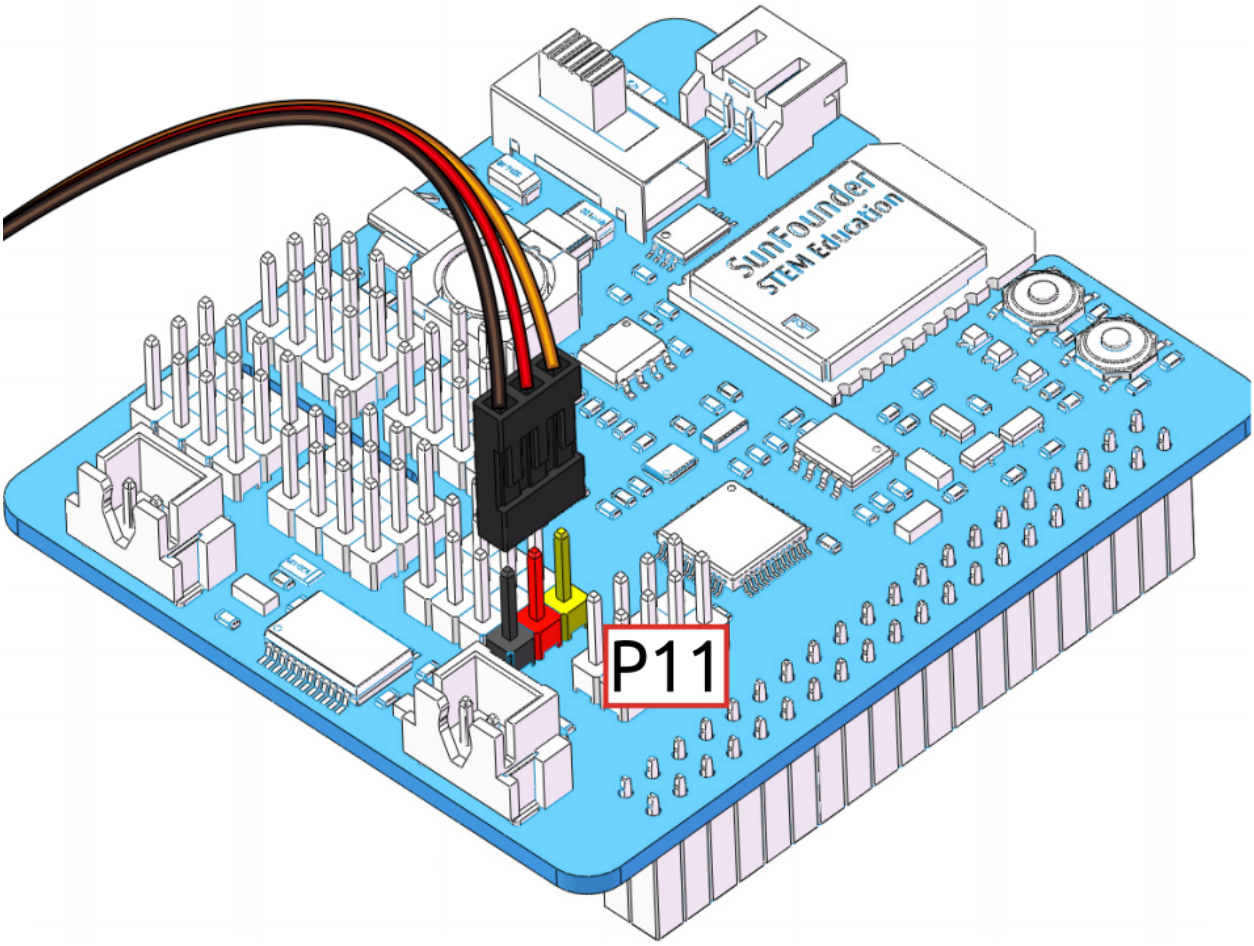


Now, run `servo_zeroing.py` in the `examples/` folder.

```
cd /home/pi/picrawler/examples
sudo python3 servo_zeroing.py
```

Note: If you get an error, try re-enabling the Raspberry Pi's I2C port, see: *I2C Configuration*.

Next, plug the servo cable into the P11 port as follows.



At this point you will see the servo arm rotate to a specific position (0°). If the servo arm does not return to 0°, press the RST button to restart the Robot HAT.

Now you can continue the installation as instructed on the assembly foldout.

Note:

- Do not unplug this servo cable before fixing it with the servo screw, you can unplug it after fixing it.
- Do not rotate the servo while it is powered on to avoid damage; if the servo shaft is not inserted at the right angle, pull the servo out and reinsert it.
- Before assembling each servo, you need to plug the servo cable into P11 and turn on the power to set its angle to 0°.

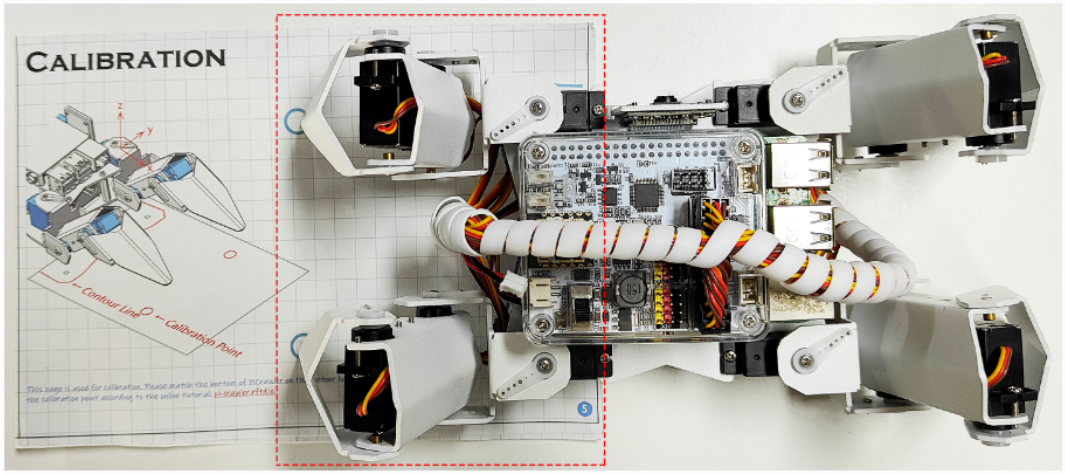
3.2 Calibrate the PiCrawler

Due to possible deviations during PiCrawler installation or limitations of the servos themselves, some servo angles may be slightly tilted, so you can calibrate them.

Of course you can skip this chapter if you think the assembly is perfect and doesn't require calibration.

The specific steps are as follows:

1. Take out the assembly leaflet, turn it to the last page, and lay it flat on the table. Then place the PiCrawler as shown below, aligning its bottom with the outline on the calibration chart.

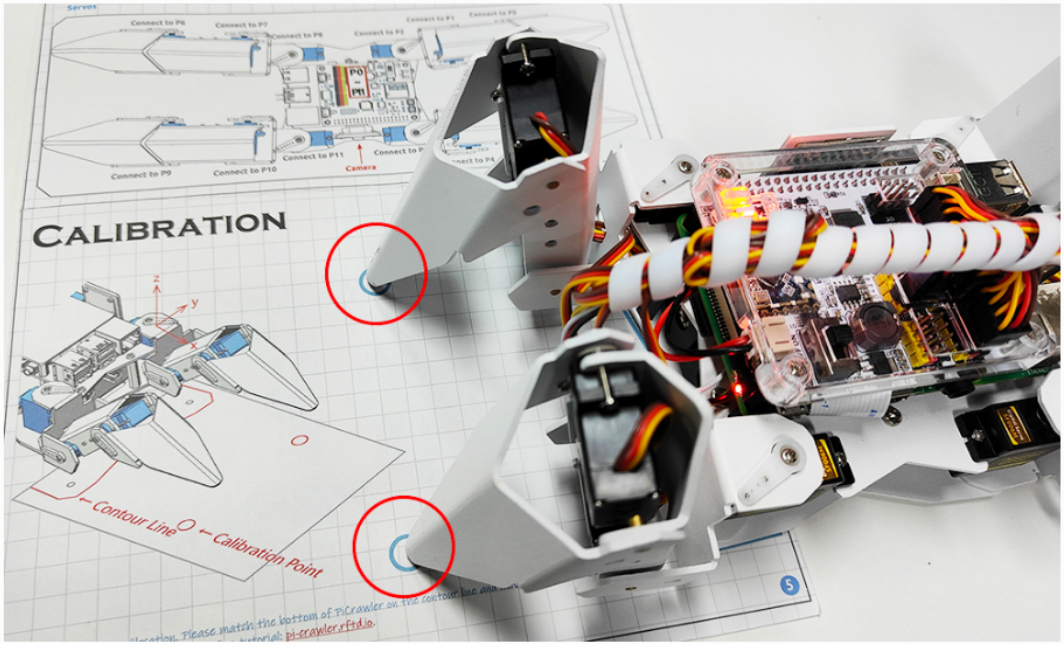


2. Run the `calibration.py`.

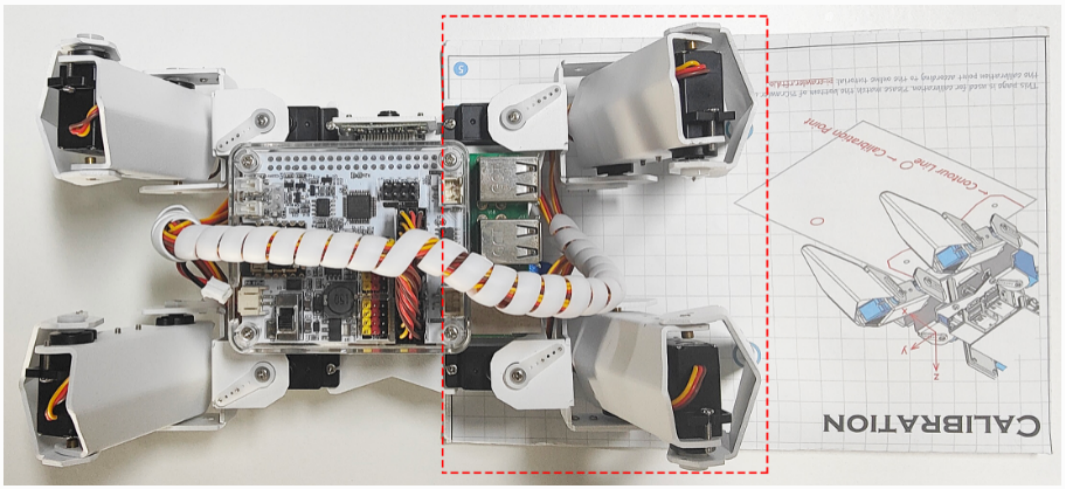
```
cd /home/pi/picrawler/examples/calibration
sudo python3 calibration.py
```

After running the above code, you will see the following interface displayed in the terminal.

3. Press `2` and `3` keys respectively to choose left 2 legsthen press `w`, `a`, `s`, `d`, `r`, and `f` keys to move them to the calibration point.



4. Now, change the calibration paper to the right and press the `1` and `4` keys to choose right 2 legs, then press `w`, `a`, `s`, `d`, `r`, and `f` keys to move them to the calibration point.

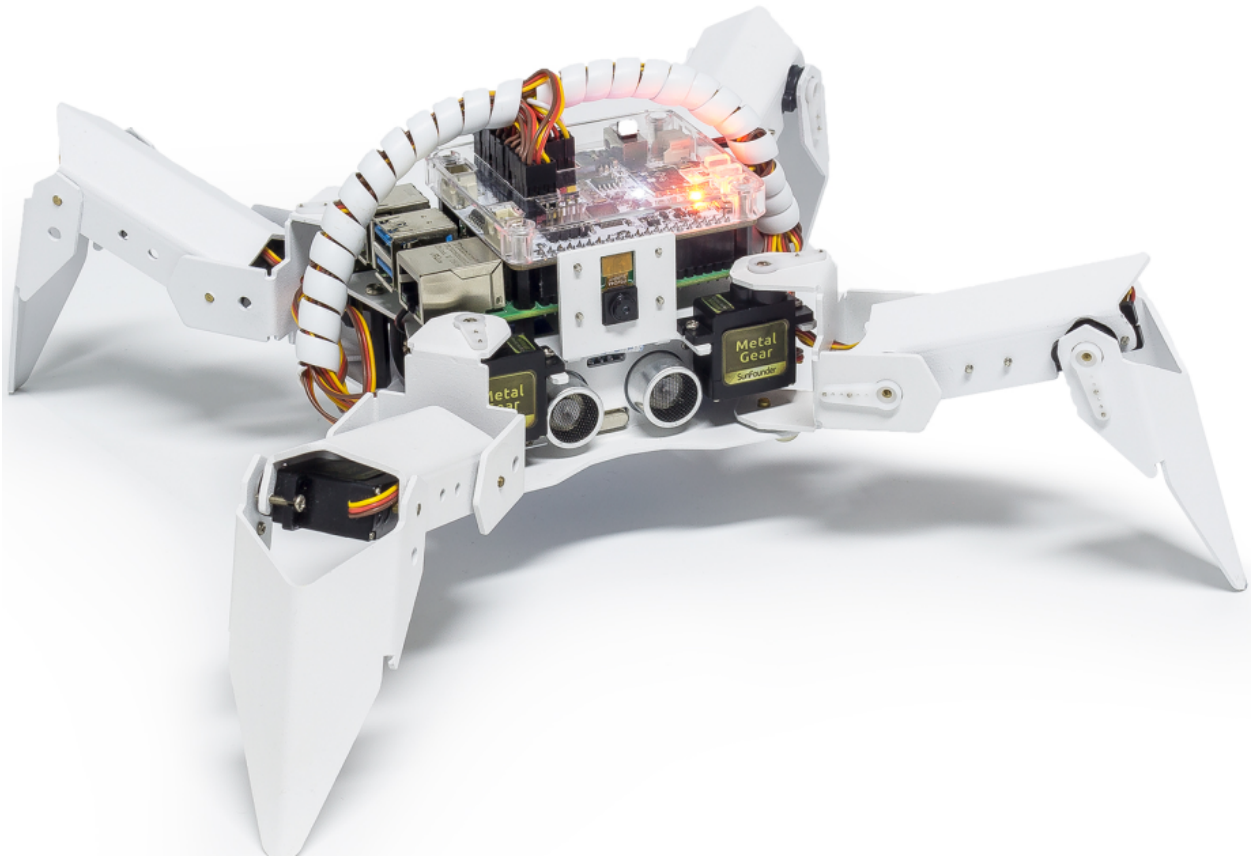


5. After the calibration is completed, press the `space` key to save, you will be prompted to enter `Y` to confirm, and then `ctrl+c` to exit the program to complete the calibration.

After the assembly is complete, you can try to run the projects below.

3.3 Move

This is PiCrawler's first project. Perform its most basic function - move.



Run the Code

```
cd /home/pi/picrawler/examples
sudo python3 move.py
```

After the code is executed, PiCrawler will perform the following actions in sequence: move forward, move backward, turn left, turn right, stand.

Code

Note: You can **Modify/Reset/Copy/Run/Stop** the code below. But before that, you need to go to source code path like `pisloth\examples`. After modifying the code, you can run it directly to see the effect.

```
from picrawler import Picrawler
from time import sleep

crawler = Picrawler([10,11,12,4,5,6,1,2,3,7,8,9])
#crawler.set_offset([0,0,0,0,0,0,0,0,0,0,0,0])

def main():

    speed = 100

    while True:

        crawler.do_action('forward',2,speed)
```

(continues on next page)

(continued from previous page)

```
        sleep(0.05)
        crawler.do_action('backward',2,speed)
        sleep(0.05)
        crawler.do_action('turn left',2,speed)
        sleep(0.05)
        crawler.do_action('turn right',2,speed)
        sleep(0.05)
        crawler.do_action('turn left angle',2,speed)
        sleep(0.05)
        crawler.do_action('turn right angle',2,speed)
        sleep(0.05)
        crawler.do_step('stand',speed)
        sleep(1)

if __name__ == "__main__":
    main()
```

How it works?

First, import the `Picrawler` class from the `picrawler` library you have installed, which contains all of PiCrawler's actions and the functions that implement them.

```
from picrawler import Picrawler
```

Then instantiate the `crawler` class.

```
crawler = Picrawler([10,11,12,4,5,6,1,2,3,7,8,9])
```

Finally use the `crawler.do_action()` function to make Pisloth move.

```
crawler.do_action('forward',2,speed)
crawler.do_action('backward',2,speed)
crawler.do_action('turn left',2,speed)
crawler.do_action('turn right',2,speed)
crawler.do_action('turn left angle',2,speed)
crawler.do_action('turn right angle',2,speed)
```

In general, all movement of PiCrawler can be implemented with the `do_action()` function. It has 3 parameters:

- `motion_name` is the name of specific actions, including: `forward`, `turn right`, `turn left`, `backward`, `turn left angle`, `turn right angle`.
- `step` represents the number of each action is done, the default is 1.
- `speed` indicates the speed of the action, the default is 50 and the range is 0~100.

In addition, `crawler.do_step('stand',speed)` is also used here to make PiCrawler stand. The usage of this function will be explained in the following example.

3.4 Keyboard Control

In this project, we will learn how to use the keyboard to remotely control the PiCrawler. You can control the PiCrawler to move forward, backward, left, and right.

Run the Code

```
cd /home/pi/picrawler/examples
sudo python3 keyboard_control.py
```

Press keys on keyboard to control PiCrawler! * w: Forward * a: Turn left * s: Backward * d: Turn right * esc: Quit

Code

```
from picrawler import Picrawler
from time import sleep
import readchar

crawler = Picrawler([10,11,12,4,5,6,1,2,3,7,8,9])
speed = 90

manual = '''
Press keys on keyboard to control PiCrawler!
    w: Forward
    a: Turn left
    s: Backward
    d: Turn right
    esc: Quit
'''

def show_info():
    print("\033[H\033[J",end='')  # clear terminal windows
    print(manual)


def main():
    show_info()
    while True:
        key = readchar.readkey()
        key = key.lower()
        if key in('wsad'):
            if 'w' == key:
                crawler.do_action('forward',1,speed)
            elif 's' == key:
                crawler.do_action('backward',1,speed)
            elif 'a' == key:
                crawler.do_action('turn left',1,speed)
            elif 'd' == key:
                crawler.do_action('turn right',1,speed)
            sleep(0.05)
            show_info()
        elif key == readchar.key.CTRL_C or key in readchar.key.ESCAPE_SEQUENCES:
            print("\n Quit")
            break

        sleep(0.02)
```

(continues on next page)

(continued from previous page)

```
if __name__ == "__main__":
    main()
```

How it works?

PiCrawler should take appropriate action based on the keyboard characters read. The `lower()` function converts upper case characters into lower case characters, so that the letter remains valid regardless of case.

```
while True:
    key = readchar.readkey()
    key = key.lower()
    if key in('wsad'):
    if 'w' == key:
        crawler.do_action('forward',1,speed)
    elif 's' == key:
        crawler.do_action('backward',1,speed)
    elif 'a' == key:
        crawler.do_action('turn left',1,speed)
    elif 'd' == key:
        crawler.do_action('turn right',1,speed)
    sleep(0.05)
    show_info()
    elif key == readchar.key.CTRL_C or key in readchar.key.ESCAPE_SEQUENCES:
    print("\n Quit")
        break
```

3.5 Sound Effect

In this example, we use PiCrawler's (to be precise, Robot HAT's) sound effects. It consists of three parts, namely **Muisc**, **Sound**, **Text to Speech**.

Install i2samp

Before using that functions, first activate the speaker so that it will be enabled and can make sounds.

Run `i2samp.sh` in, and this script will install everything needed to use i2s amplifier.

```
cd /home/pi/picrawler/
sudo bash i2samp.sh
```

There will be several prompts asking to confirm the request. Respond to all prompts with a **Y**. After the changes have been made to the Raspberry Pi system, the computer will need to reboot for these changes to take effect.

After rebooting, run the `i2samp.sh` script again to test the amplifier. If a sound successfully plays from the speaker, the configuration is complete.

Run the Code

```
cd /home/pi/picrawler/examples
sudo python3 sound_effect.py
```

After the code runs, please operate according to the prompt that printed on the terminal.

Input key to call the function! * q: Play background music * 1: Play sound effect * 2: Play sound effect with threads * t: Text to speak

Code

```
from time import sleep
from robot_hat import Music,TTS
import readchar

music = Music()
tts = TTS()

manual = '''
Input key to call the function!
    q: Play background music
    1: Play sound effect
    2: Play sound effect with threads
    t: Text to speak
'''

def main():
    print(manual)

    flag_bgm = False
    music.music_set_volume(20)
    tts.lang("en-US")


    while True:
        key = readchar.readkey()
        key = key.lower()
        if key == "q":
            flag_bgm = not flag_bgm
            if flag_bgm is True:
                music.background_music('./musics/sports-Ahjay_Stelino.mp3')
            else:
                music.music_stop()

        elif key == "1":
            music.sound_effect_play('./sounds/talk1.wav')
            sleep(0.05)
            music.sound_effect_play('./sounds/talk3.wav')
            sleep(0.05)
            music.sound_effect_play('./sounds/sign.wav')
            sleep(0.5)

        elif key =="2":
            music.sound_effect_threading('./sounds/talk1.wav')
            sleep(0.05)
            music.sound_effect_threading('./sounds/talk3.wav')
            sleep(0.05)
            music.sound_effect_threading('./sounds/sign.wav')
            sleep(0.5)

        elif key == "t":
            words = "Hello"
            tts.say(words)

if __name__ == "__main__":
    main()
```

How it works?

Functions related to background music include these:

- `music = Music()` : Declare the object.
- `music.music_set_volume(20)` : Set the volume, the range is 0~100.
- `music.background_music(./musics/sports-Ahjay_Stelino.mp3)` : Play music files, here is the **sports-Ahjay_Stelino.mp3** file under the `./musics` path.
- `music.music_stop()` : Stop playing background music.

Note: You can add different sound effects or music to `musics` or `sounds` folder via *Filezilla Software*.

Functions related to sound effects include these:

- `music = Music()`
- `music.sound_effect_play('./sounds/talk1.wav')` : Play the sound effect file, here is the **talk1.wav** file under the `./musics` path.
- `muisc.sound_effect_threading('./sounds/talk1.wav')` : Play the sound effect file in a new thread mode without suspending the main thread.

Functions related to Text to Speech include these:

- `tts = TTS()`
- `tts.say(words)` : Text audio.
- `tts.lang("en-US")` : Set the language.

Note: Set the language by setting the parameters of `lang("")` with the following characters.

Table 1: Language

zh-CN	Mandarin (Chinese)
en-US	English-United States
en-GB	English-United Kingdom
de-DE	Germany-Deutsch
es-ES	España-Español
fr-FR	France-Le français
it-IT	Italia-lingua italiana

3.6 Obstacle Avoidance

In this project, picrawler will use an ultrasonic module to detect obstacles in front. When PiCrawler detects an obstacle, it will send a signal and look for another direction to move forward.

Run the Code

```
cd /home/pi/picrawler/examples
sudo python3 avoid.py
```

After the code runs, PiCrawler will walk forward. If it detects that the distance of the obstacle ahead is less than 10cm, it will stop and sound a warning, then turn left and stop. If there is no obstacle in the direction after turning left or the obstacle distance is greater than 10, it will continue to move forward.

Code

Note: You can **Modify/Reset/Copy/Run/Stop** the code below. But before that, you need to go to source code path like `picrawler\examples`. After modifying the code, you can run it directly to see the effect.

```
from picrawler import Picrawler
from robot_hat import TTS, Music
from robot_hat import Ultrasonic
from robot_hat import Pin
import time
import os

tts = TTS()
music = Music()
```

(continues on next page)

(continued from previous page)

```
crawler = Picrawler([10,11,12,4,5,6,1,2,3,7,8,9])
#crawler.set_offset([0,0,0,0,0,0,0,0,0,0,0,0])
sonar = Ultrasonic(Pin("D2") ,Pin("D3"))

alert_distance = 15

speed = 100

def main():
    distance = sonar.read()
    print(distance)
    if distance < 0:
        pass
    elif distance <= alert_distance:
        try:
            music.sound_effect_threading('./sounds/sign.wav')
        except Exception as e:
            print(e)
        crawler.do_action('turn left angle',3,speed)
        time.sleep(0.2)
    else :
        crawler.do_action('forward', 1,speed)
        time.sleep(0.2)


if __name__ == "__main__":
    while True:
        main()
```

How it works?

You can get the distance by importing the `Ultrasonic` class.

```
from robot_hat import Ultrasonic
```

Then initialize the ultrasonic pins.

```
sonar = Ultrasonic(Pin("D2") ,Pin("D3"))
```

Here is the main program.

- Read the `distance` detected by ultrasonic module and filter out the values less than 0 (When the ultrasonic module is too far from the obstacle or cannot read the data correctly, `distance<0` will appear).
- When the `distance` is less than or equal to `alert_distance` (the threshold value set earlier, which is 10), play the sound effect `sign.wav`. PiCrawler does `turn left angle`.
- When the `distance` is greater than `alert_distance`, PiCrawler will move `forward`.

```
distance = sonar.read()
print(distance)
if distance < 0:
    pass
elif distance <= alert_distance:
    try:
        music.sound_effect_threading('./sounds/sign.wav')
    except Exception as e:
```

(continues on next page)

(continued from previous page)

```
        print(e)
    crawler.do_action('turn left angle',3,speed)
    time.sleep(0.2)
else :
    crawler.do_action('forward', 1,speed)
    time.sleep(0.2)
```

Note: You can add different sound effects or music to `musics` or `sounds` folder via *Filezilla Software*.

3.7 Computer Vision

This project will officially enter the field of computer vision!

Run the Code

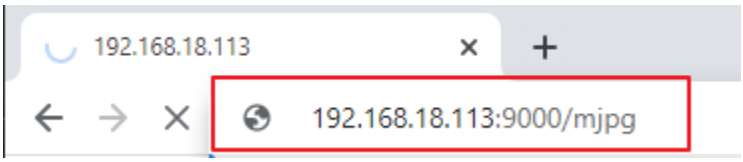
```
cd /home/pi/picrawler/examples
sudo python3 display.py
```

View the Image

After the code runs, the terminal will display the following prompt:

```
No desktop !
* Serving Flask app "vilib.vilib" (lazy loading)
* Environment: production
WARNING: Do not use the development server in a production environment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:9000/ (Press CTRL+C to quit)
```

Then you can enter `http://<your IP>:9000/mjpg` in the browser to view the video screen. such as: `https://192.168.18.113:9000/mjpg`



After the program runs, you will see the following information in the final:

- Input key to call the function!
- q: Take photo
- 1: Color detect : red
- 2: Color detect : orange
- 3: Color detect : yellow
- 4: Color detect : green
- 5: Color detect : blue
- 6: Color detect : purple

- 0: Switch off Color detect
- rScan the QR code
- f: Switch ON/OFF face detect
- s: Display detected object information

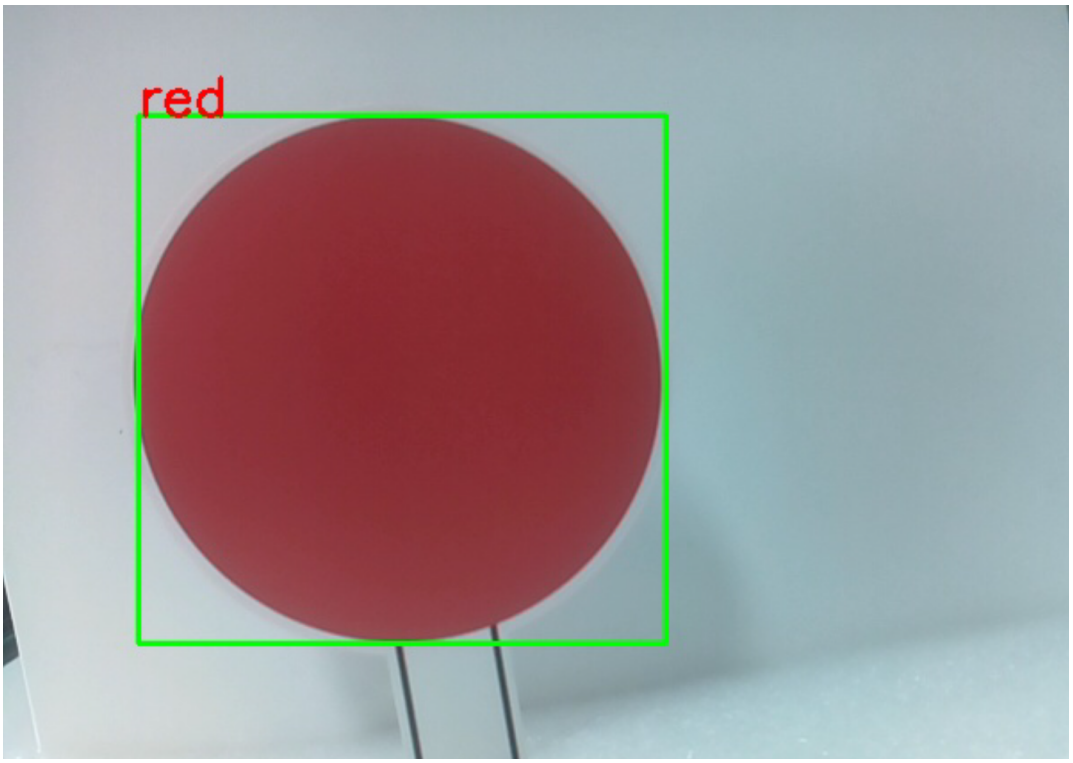
Please follow the prompts to activate the corresponding functions.

- **Take Photo**

  Type `q` in the terminal and press Enter. The picture currently seen by the camera will be saved (if the color detection function is turned on, the mark box will also appear in the saved picture). You can see these photos from the `/home/pi/Pictures/PiCrawler/` directory of the Raspberry Pi. You can use tools such as *Filezilla Software* to transfer photos to your PC.

- **Color Detect**

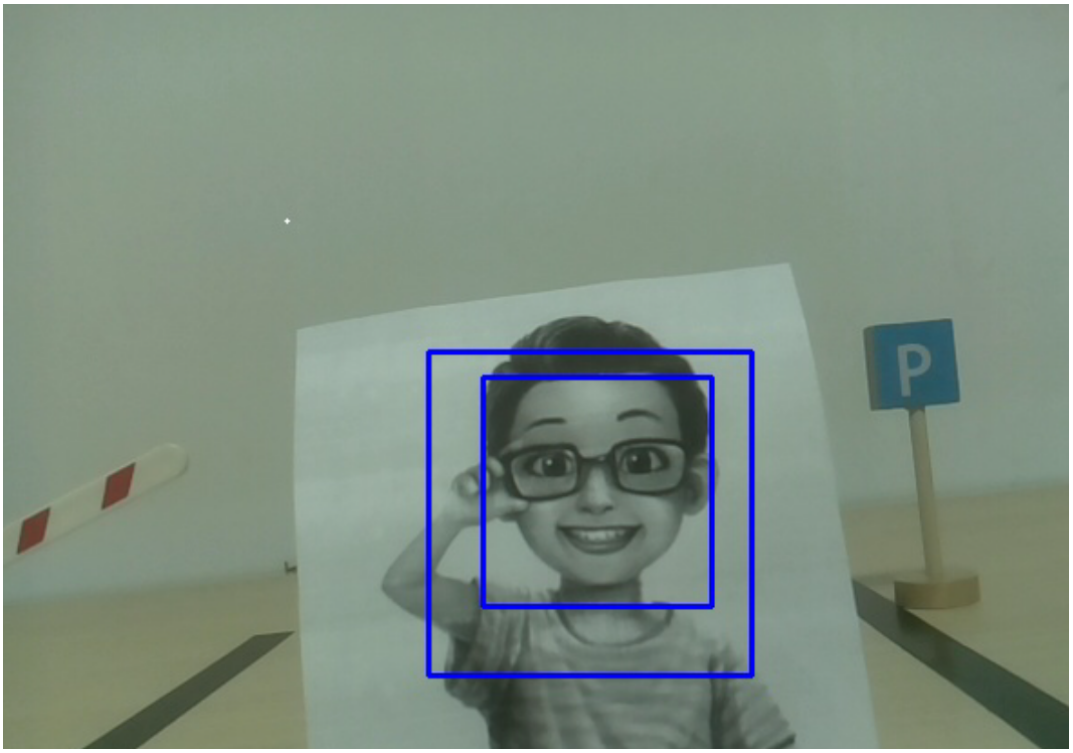
  Entering a number between `1~6` will detect one of the colors in "red, orange, yellow, green, blue, purple". Enter `0` to turn off color detection.

  Note: You can download and print the `PDF Color Cards` for color detection.

- **Face Detect**

  Type `f` to turn on face detection.



- **QR Code Detect**

  Enter `r` to open the QR code recognition. No other operations can be performed before the QR code is recognized. The decoding information of the QR code will be printed in the terminal.

- **Display Information**

Entering `s` will print the information of the face detection (and color detection) target in the terminal. Including the center coordinates (X, Y) and size (Weight, height) of the measured object.

Code

```
from pydoc import text
from vilib import Vilib
from time import sleep, time, strftime, localtime
import threading
import readchar


flag_face = False
flag_color = False
qr_code_flag = False

manual = '''
Input key to call the function!
    q: Take photo
    1: Color detect : red
    2: Color detect : orange
    3: Color detect : yellow
    4: Color detect : green
    5: Color detect : blue
    6: Color detect : purple
    0: Switch off Color detect
    rScan the QR code
    f: Switch ON/OFF face detect
    s: Display detected object information
'''

color_list = ['close', 'red', 'orange', 'yellow',
        'green', 'blue', 'purple',
]

def face_detect(flag):
    print("Face Detect:" + str(flag))
    Vilib.face_detect_switch(flag)


def qrcode_detect():
    global qr_code_flag
    if qr_code_flag == True:
        Vilib.qrcode_detect_switch(True)
        print("Waitting for QR code")

    text = None
    while True:
        temp = Vilib.detect_obj_parameter['qr_data']
        if temp != "None" and temp != text:
            text = temp
            print('QR code:%s'%text)
        if qr_code_flag == False:
            break
        sleep(0.5)
    Vilib.qrcode_detect_switch(False)
```

(continues on next page)

(continued from previous page)

```
def take_photo():
    _time = strftime('%Y-%m-%d-%H-%M-%S',localtime(time()))
    name = 'photo_%s'%_time
    path = "/home/pi/Pictures/PiCrawler/"
    Vilib.take_photo(name, path)
    print('photo save as %s%s.jpg'%(path,name))


def object_show():
    global flag_color, flag_face

    if flag_color is True:
        if Vilib.detect_obj_parameter['color_n'] == 0:
            print('Color Detect: None')
        else:
            color_coodinate = (Vilib.detect_obj_parameter['color_x'],Vilib.detect_obj_
↪parameter['color_y'])
            color_size = (Vilib.detect_obj_parameter['color_w'],Vilib.detect_obj_
↪parameter['color_h'])
            print("[Color Detect] ","Coordinate:",color_coodinate,"Size",color_size)

    if flag_face is True:
        if Vilib.detect_obj_parameter['human_n'] == 0:
            print('Face Detect: None')
        else:
            human_coodinate = (Vilib.detect_obj_parameter['human_x'],Vilib.detect_obj_
↪parameter['human_y'])
            human_size = (Vilib.detect_obj_parameter['human_w'],Vilib.detect_obj_
↪parameter['human_h'])
            print("[Face Detect] ","Coordinate:",human_coodinate,"Size",human_size)


def main():
    global flag_face, flag_color, qr_code_flag
    qrcode_thread = None

    Vilib.camera_start(vflip=False,hflip=False)
    Vilib.display(local=True,web=True)
    print(manual)

    while True:
        # readkey
        key = readchar.readkey()
        key = key.lower()
        # take photo
        if key == 'q':
            take_photo()
        # color detect
        elif key != '' and key in ('0123456'):  # '' in ('0123') -> True
            index = int(key)
            if index == 0:
                flag_color = False
                Vilib.color_detect('close')
            else:
                flag_color = True
                Vilib.color_detect(color_list[index]) # color_detect(color:str ->␣
↪color_name/close)
```

(continues on next page)

(continued from previous page)

```
            print('Color detect : %s'%color_list[index])
        # face detection
        elif key =="f":
            flag_face = not flag_face
            face_detect(flag_face)
        # qrcode detection
        elif key =="r":
            qr_code_flag = not qr_code_flag
            if qr_code_flag == True:
                if qrcode_thread == None or not qrcode_thread.is_alive():
                    qrcode_thread = threading.Thread(target=qrcode_detect)
                    qrcode_thread.setDaemon(True)
                    qrcode_thread.start()
            else:
                if qrcode_thread != None and qrcode_thread.is_alive():
                # wait for thread to end
                    qrcode_thread.join()
                    print('QRcode Detect: close')
        # show detected object information
        elif key == "s":
            object_show()

        sleep(0.5)


if __name__ == "__main__":
    main()
```

How it works?

The first thing you need to pay attention to here is the following function. These two functions allow you to start the camera.

```
Vilib.camera_start()
Vilib.display()
```

Functions related to "object detection":

- `Vilib.face_detect_switch(True)` : Switch ON/OFF face detection
- `Vilib.color_detect(color)` : For color detection, only one color detection can be performed at the same time. The parameters that can be input are: `"red"`, `"orange"`, `"yellow"`, `"green"`, `"blue"`, `"purple"`
- `Vilib.color_detect_switch(False)` : Switch OFF color detection
- `Vilib.qrcode_detect_switch(False)` : Switch ON/OFF QR code detection, Returns the decoded data of the QR code.
- `Vilib.gesture_detect_switch(False)` : Switch ON/OFF gesture detection
- `Vilib.traffic_sign_detect_switch(False)` : Switch ON/OFF traffic sign detection

The information detected by the target will be stored in the `detect_obj_parameter = Manager().dict()` dictionary.

In the main program, you can use it like this:

```
Vilib.detect_obj_parameter['color_x']
```

The keys of the dictionary and their uses are shown in the following list:

- `color_x`: the x value of the center coordinate of the detected color block, the range is 0~320
- `color_y`: the y value of the center coordinate of the detected color block, the range is 0~240
- `color_w`: the width of the detected color block, the range is 0~320
- `color_h`: the height of the detected color block, the range is 0~240
- `color_n`: the number of detected color patches
- `human_x`: the x value of the center coordinate of the detected human face, the range is 0~320
- `human_y`: the y value of the center coordinate of the detected face, the range is 0~240
- `human_w`: the width of the detected human face, the range is 0~320
- `human_h`: the height of the detected face, the range is 0~240
- `human_n`: the number of detected faces
- `traffic_sign_x`: the center coordinate x value of the detected traffic sign, the range is 0~320
- `traffic_sign_y`: the center coordinate y value of the detected traffic sign, the range is 0~240
- `traffic_sign_w`: the width of the detected traffic sign, the range is 0~320
- `traffic_sign_h`: the height of the detected traffic sign, the range is 0~240
- `traffic_sign_t`: the content of the detected traffic sign, the value list is *['stop','right','left','forward']*
- `gesture_x`: The center coordinate x value of the detected gesture, the range is 0~320
- `gesture_y`: The center coordinate y value of the detected gesture, the range is 0~240
- `gesture_w`: The width of the detected gesture, the range is 0~320
- `gesture_h`: The height of the detected gesture, the range is 0~240
- `gesture_t`: The content of the detected gesture, the value list is *["paper","scissor","rock"]*
- `qr_date`: the content of the QR code being detected
- `qr_x`: the center coordinate x value of the QR code to be detected, the range is 0~320
- `qr_y`: the center coordinate y value of the QR code to be detected, the range is 0~240
- `qr_w`: the width of the QR code to be detected, the range is 0~320
- `qr_h`: the height of the QR code to be detected, the range is 0~320

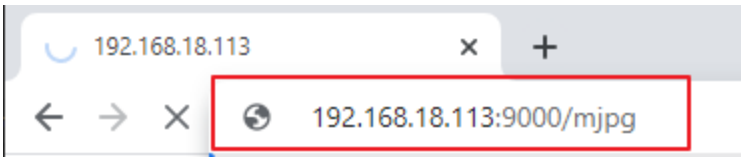
3.8 Record Video

This example will guide you how to use the recording function.

Run the Code

```
cd /home/pi/picrawler/examples
sudo python3 record_video.py
```

After the code runs, you can enter `http://<your IP>:9000/mjpg` in the browser to view the video screen. such as: `http://192.168.18.113:9000/mjpg`



Recording can be stopped or started by pressing the keys on the keyboard.

- Press `q` to begin recording or pause/continue, `e` to stop recording or save.
- If you want to exit the program, press `esc`.

Code

```
from time import sleep,strftime,localtime
from vilib import Vilib
import readchar

manual = '''
Press keys on keyboard to control recording:
    Q: record/pause/continue
    E: stop
    ESC: Quit
'''

def print_overwrite(msg,  end='', flush=True):
    print('\r\033[2K', end='',flush=True)
    print(msg, end=end, flush=True)

def main():
    rec_flag = 'stop' # start,pause,stop
    vname = None
    Vilib.rec_video_set["path"] = "/home/pi/Videos/" # set path

    Vilib.camera_start(vflip=False,hflip=False)
    Vilib.display(local=True,web=True)
    sleep(0.8)  # wait for startup

    print(manual)
    while True:
        # read keyboard
        key = readchar.readkey()
        key = key.lower()
        # start,pause
        if key == 'q':
            key = None
            if rec_flag == 'stop':
                rec_flag = 'start'
                # set name
                vname = strftime("%Y-%m-%d-%H.%M.%S", localtime())
                Vilib.rec_video_set["name"] = vname
                # start record
                Vilib.rec_video_run()
                Vilib.rec_video_start()
                print_overwrite('rec start ...')
            elif rec_flag == 'start':
                rec_flag = 'pause'
                Vilib.rec_video_pause()
                print_overwrite('pause')
```

(continues on next page)

(continued from previous page)

```
            elif rec_flag == 'pause':
                rec_flag = 'start'
                Vilib.rec_video_start()
                print_overwrite('continue')
        # stop
        elif key == 'e' and rec_flag != 'stop':
            key = None
            rec_flag = 'stop'
            Vilib.rec_video_stop()
            print_overwrite("The video saved as %s%s.avi"%(Vilib.rec_video_set["path
↪"],vname),end='\n')
        # quit
        elif key == readchar.key.CTRL_C or key in readchar.key.ESCAPE_SEQUENCES:
            Vilib.camera_close()
            print('\nquit')
            break

        sleep(0.1)

if __name__ == "__main__":
    main()
```

How it works?

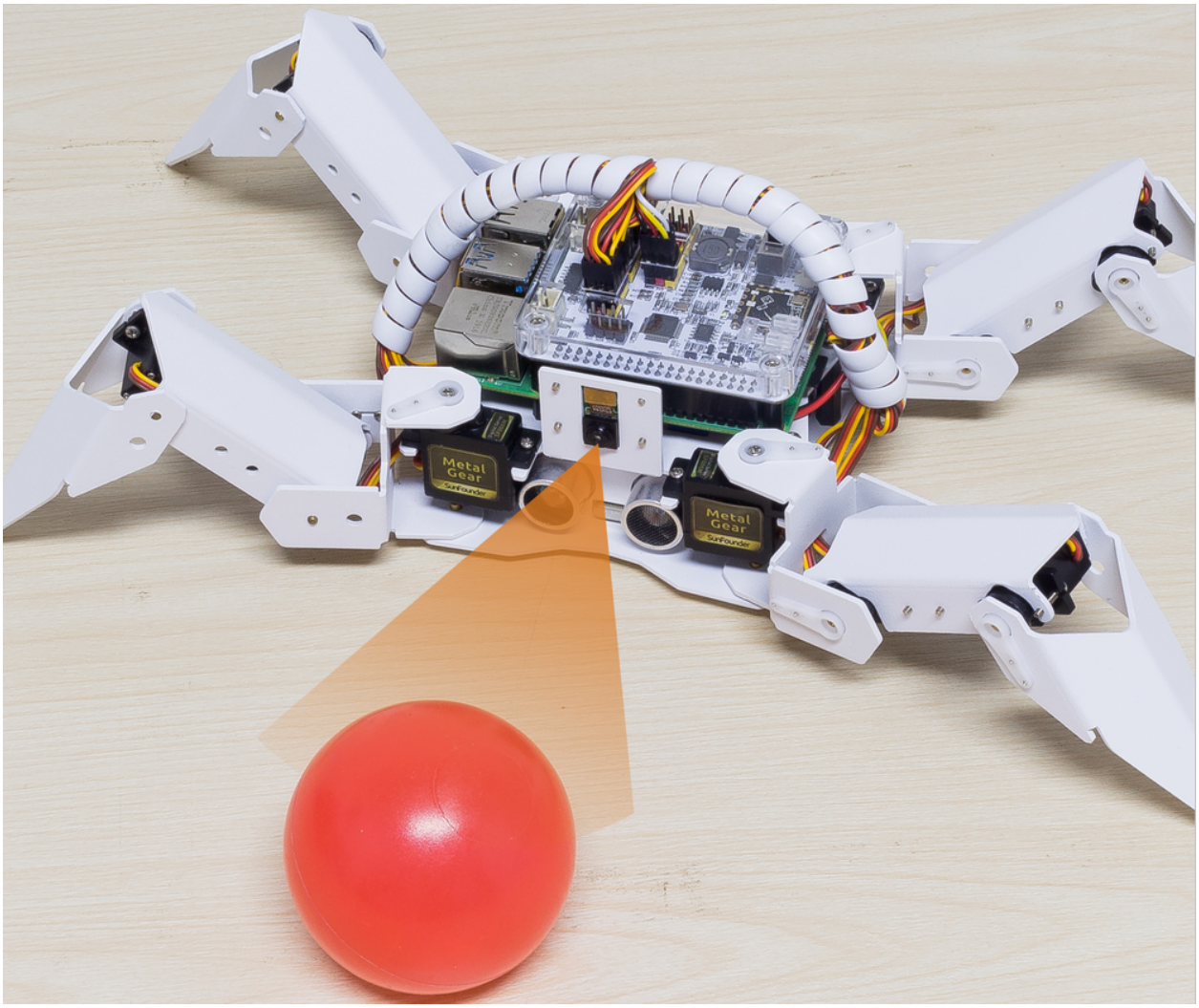
Functions related to recording include the following:

- `Vilib.rec_video_run(video_name)` : Started the thread to record the video. `video_name` is the name of the video file, it should be a string.
- `Vilib.rec_video_start()` : Start or continue video recording.
- `Vilib.rec_video_pause()` : Pause recording.
- `Vilib.rec_video_stop()` : Stop recording.

`Vilib.rec_video_set["path"] = "/home/pi/video/test/"` sets the storage location of video files.

3.9 Bull Fight

Make PiCrawler an angry bull! Use its camera to track and rush the red cloth!



Run the Code

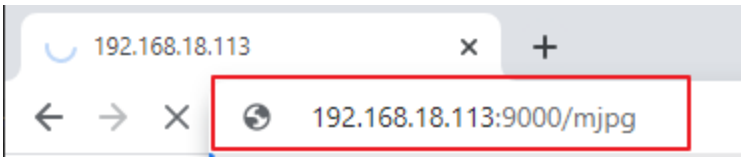
```
cd /home/pi/picrawler/examples
sudo python3 bull_fight.py
```

View the Image

After the code runs, the terminal will display the following prompt:

```
No desktop !
* Serving Flask app "vilib.vilib" (lazy loading)
* Environment: production
WARNING: Do not use the development server in a production environment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:9000/ (Press CTRL+C to quit)
```

Then you can enter `http://<your IP>:9000/mjpg` in the browser to view the video screen. such as: `https://192.168.18.113:9000/mjpg`



Code

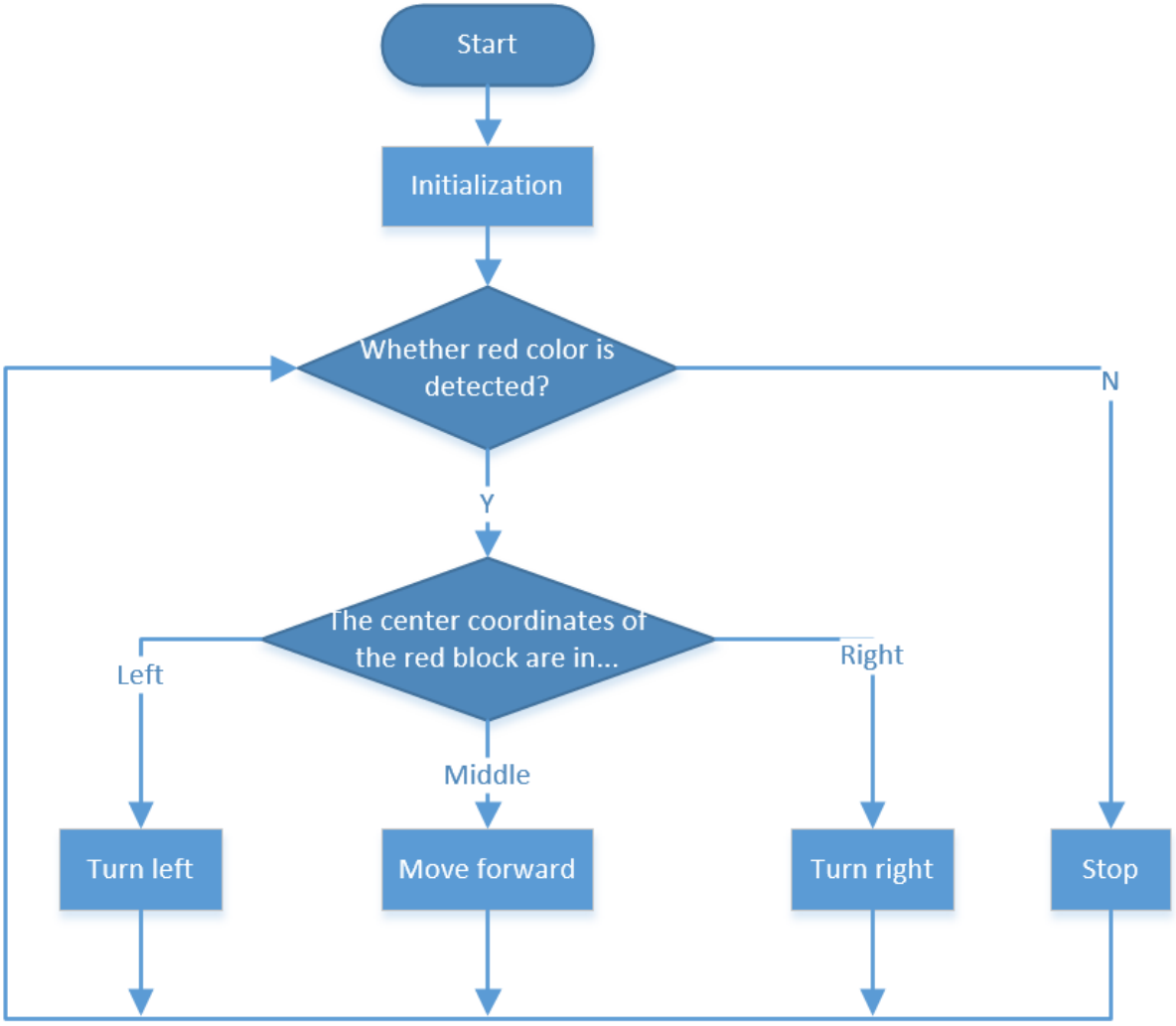
Note: You can **Modify/Reset/Copy/Run/Stop** the code below. But before that, you need to go to source code path like `picrawler\examples`. After modifying the code, you can run it directly to see the effect.

```
from picrawler import Picrawler
from time import sleep
from robot_hat import Music
from vilib import Vilib


crawler = Picrawler([10,11,12,4,5,6,1,2,3,7,8,9])
#crawler.set_offset([0,0,0,0,0,0,0,0,0,0,0,0])

music = Music()

def main():
    Vilib.camera_start()
    Vilib.display()
    Vilib.color_detect("red")
    speed = 100

    while True:
        if Vilib.detect_obj_parameter['color_n']!=0:
            coordinate_x = Vilib.detect_obj_parameter['color_x']
            music.sound_effect_threading('./sounds/talk1.wav')

            if coordinate_x < 100:
                crawler.do_action('turn left',1,speed)
                sleep(0.05)
            elif coordinate_x > 220:
                crawler.do_action('turn right',1,speed)
                sleep(0.05)
            else :
                crawler.do_action('forward',2,speed)
                sleep(0.05)
        else :
            crawler.do_step('stand',speed)
            sleep(0.05)


if __name__ == "__main__":
    main()
```

How it works?

In general, this project combines the knowledge points of *Move*, *Computer Vision* and *Sound Effect*.

Its flow is shown in the figure below:



3.10 Treasure Hunt

Arrange a maze in your room and place six different color cards in six corners. Then control PiCrawler to search for these color cards one by one!

Note: You can download and print the `PDF Color Cards` for color detection.

Run the Code

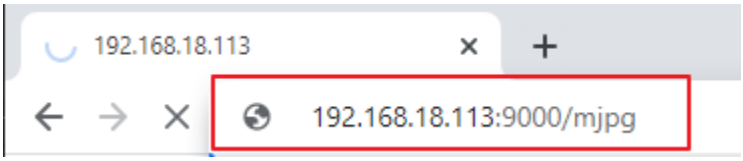
```
cd /home/pi/picrawler/examples
sudo python3 treasure_hunt.py
```

View the Image

After the code runs, the terminal will display the following prompt:

```
No desktop !
* Serving Flask app "vilib.vilib" (lazy loading)
* Environment: production
WARNING: Do not use the development server in a production environment.
Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:9000/ (Press CTRL+C to quit)
```

Then you can enter `http://<your IP>:9000/mjpg` in the browser to view the video screen. such as: `http://192.168.18.113:9000/mjpg`



Code

```
from picrawler import Picrawler
from time import sleep
from robot_hat import Music,TTS
from vilib import Vilib
import readchar
import random
import threading

crawler = Picrawler([10,11,12,4,5,6,1,2,3,7,8,9])
#crawler.set_offset([0,0,0,0,0,0,0,0,0,0,0,0])

music = Music()
tts = TTS()

manual = '''
Press keys on keyboard to control Picrawler!
        w: Forward
        a: Turn left
        s: Backward
        d: Turn right
        space: Say the target again
        esc: Quit
'''

color = "red"
color_list=["red","orange","yellow","green","blue","purple"]
key_dict = {
        'w': 'forward',
        's': 'backward',
        'a': 'turn_left',
        'd': 'turn_right',
}
def renew_color_detect():
        global color
        color = random.choice(color_list)
        Vilib.color_detect(color)
        tts.say("Look for " + color)

key = None
```

(continues on next page)

(continued from previous page)

```
lock = threading.Lock()
def key_scan_thread():
        global key
        while True:
                key_temp = readchar.readkey()
                print('\r',end='')
                with lock:
                        key = key_temp.lower()
                        if key == readchar.key.SPACE:
                                key = 'space'
                        elif key == readchar.key.CTRL_C or key in readchar.key.ESCAPE_
↪SEQUENCES:
                                key = 'quit'
                                break
                sleep(0.01)

def main():
        global key
        action = None
        Vilib.camera_start(vflip=False,hflip=False)
        Vilib.display(local=False,web=True)
        sleep(0.8)
        speed = 100
        print(manual)

        sleep(1)
        _key_t = threading.Thread(target=key_scan_thread)
        _key_t.setDaemon(True)
        _key_t.start()

        tts.say("game start")
        sleep(0.05)
        renew_color_detect()
        while True:

                if Vilib.detect_obj_parameter['color_n']!=0 and Vilib.detect_obj_
↪parameter['color_w']>100:
                        tts.say("will done")
                        sleep(0.05)
                        renew_color_detect()

                with lock:
                        if key != None and key in ('wsad'):
                                action = key_dict[str(key)]
                                key =  None
                        elif key == 'space':
                                tts.say("Look for " + color)
                                key =  None
                        elif key == 'quit':
                                _key_t.join()
                                Vilib.camera_close()
                                print("\n\rQuit")
                                break

                if action != None:
                        crawler.do_action(action,1,speed)
                        action = None
```

(continues on next page)

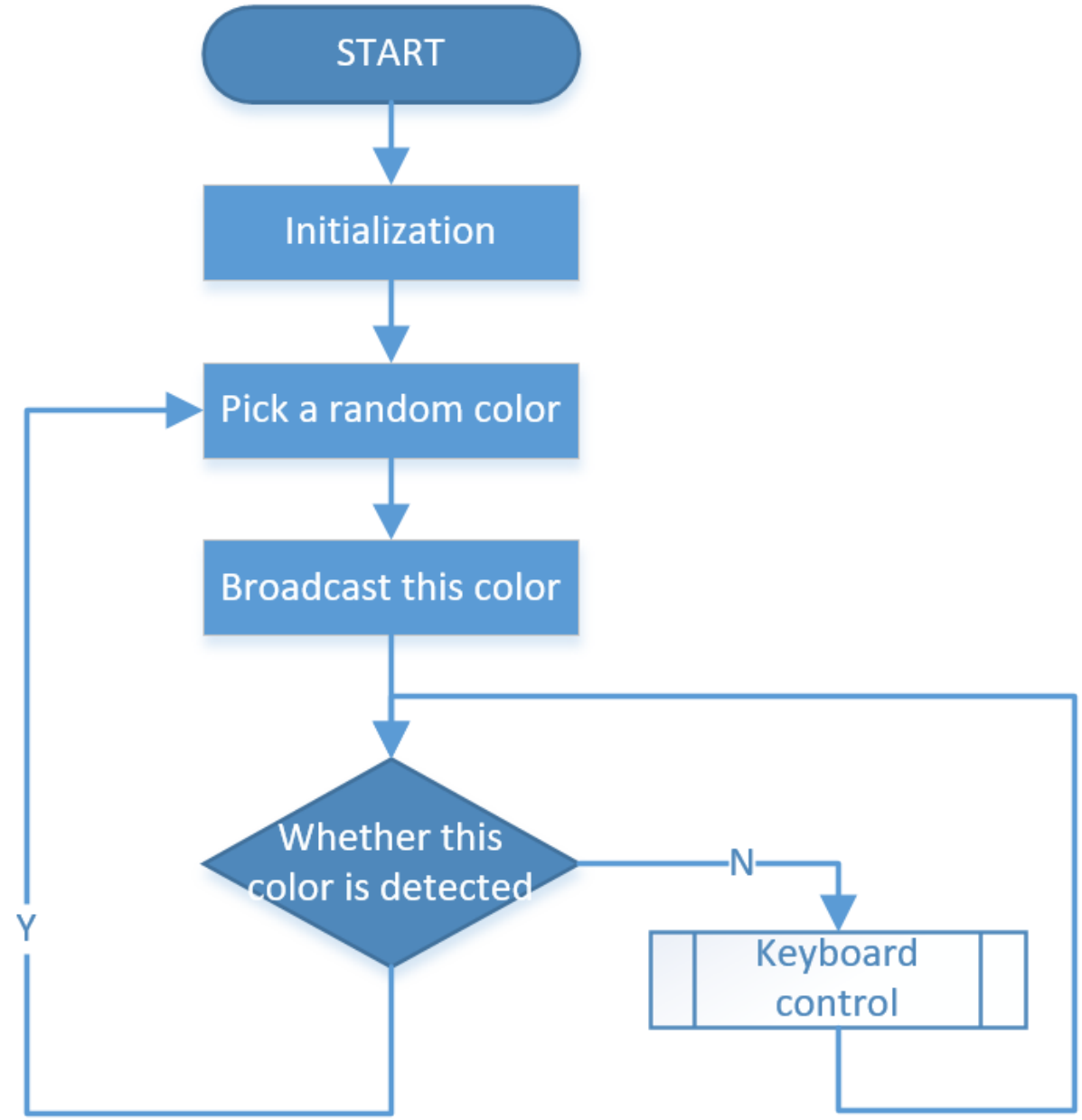
(continued from previous page)

```
                sleep(0.05)


if __name__ == "__main__":
        main()
```

How it works?

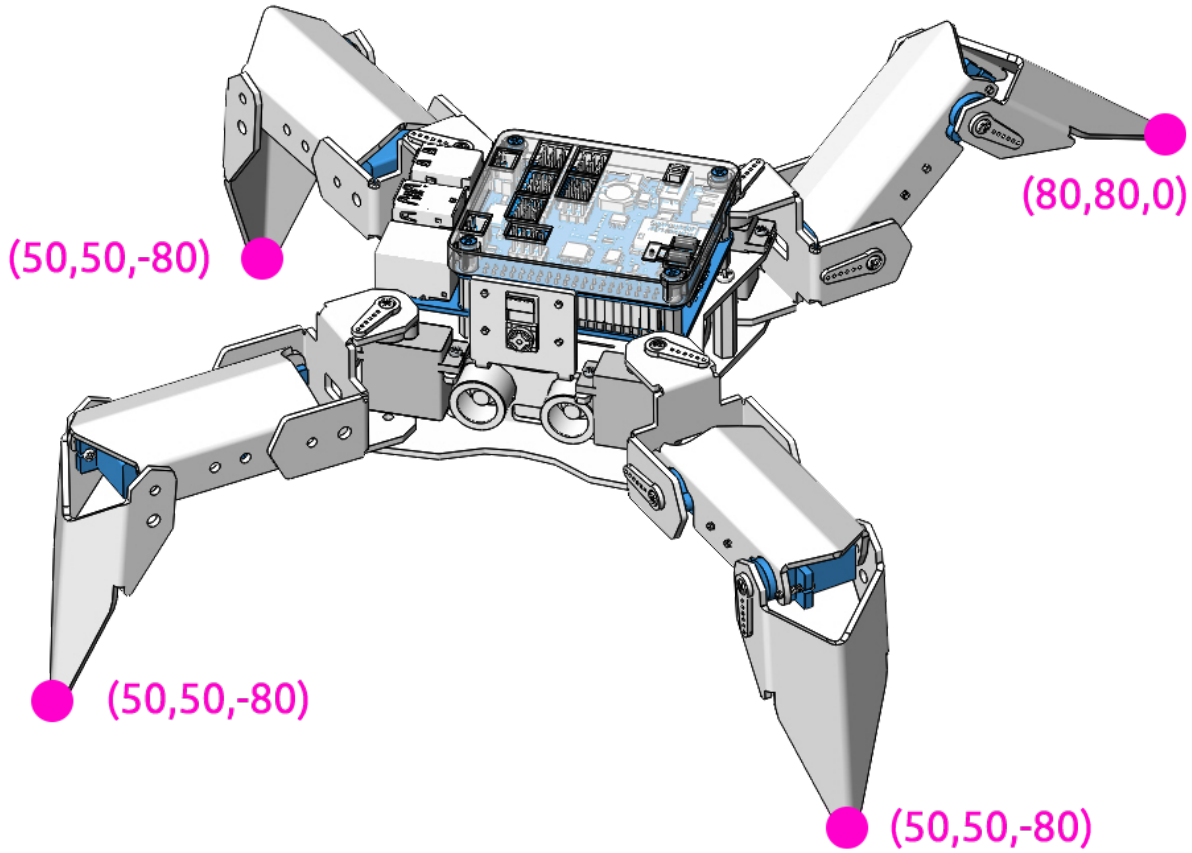
In general, this project combines the knowledge points of *Keyboard Control*, *Computer Vision* and *Sound Effect*.

Its flow is shown in the figure below:



3.11 Pose

PiCrawler can assume a specific posture by writing a coordinate array. Here it assumes a raised right rear foot posture.



Run the Code

```
cd /home/pi/picrawler/examples
sudo python3 do_step.py
```

Code

```
from picrawler import Picrawler
from time import sleep

crawler = Picrawler([10,11,12,4,5,6,1,2,3,7,8,9])
#crawler.set_offset([0,0,0,0,0,0,0,0,0,0,0,0])

## [right front],[left front],[left rear],[right rear]
new_step=[[50, 50, -80], [50, 50, -80],[80, 80, 0], [50, 50, -80]]

def main():

    speed = 100

    while True:
```

(continues on next page)

(continued from previous page)

```
        crawler.do_step('stand',speed)
        print(crawler.step_list.get('stand'))
        sleep(3)
        crawler.do_step(new_step,speed)
        print(new_step)
        sleep(3)

if __name__ == "__main__":
    main()
```

How it works?

In this code, the code you need to pay attention to is this `crawler.do_step()`.

Similar to `do_action()`, `do_step()` can also manipulate PiCrawler's behavior. The difference is that the former can perform the continuous behavior of `move forward`, while the latter can be used to make separate gestures of `stand` and `sit`.

It has two uses:

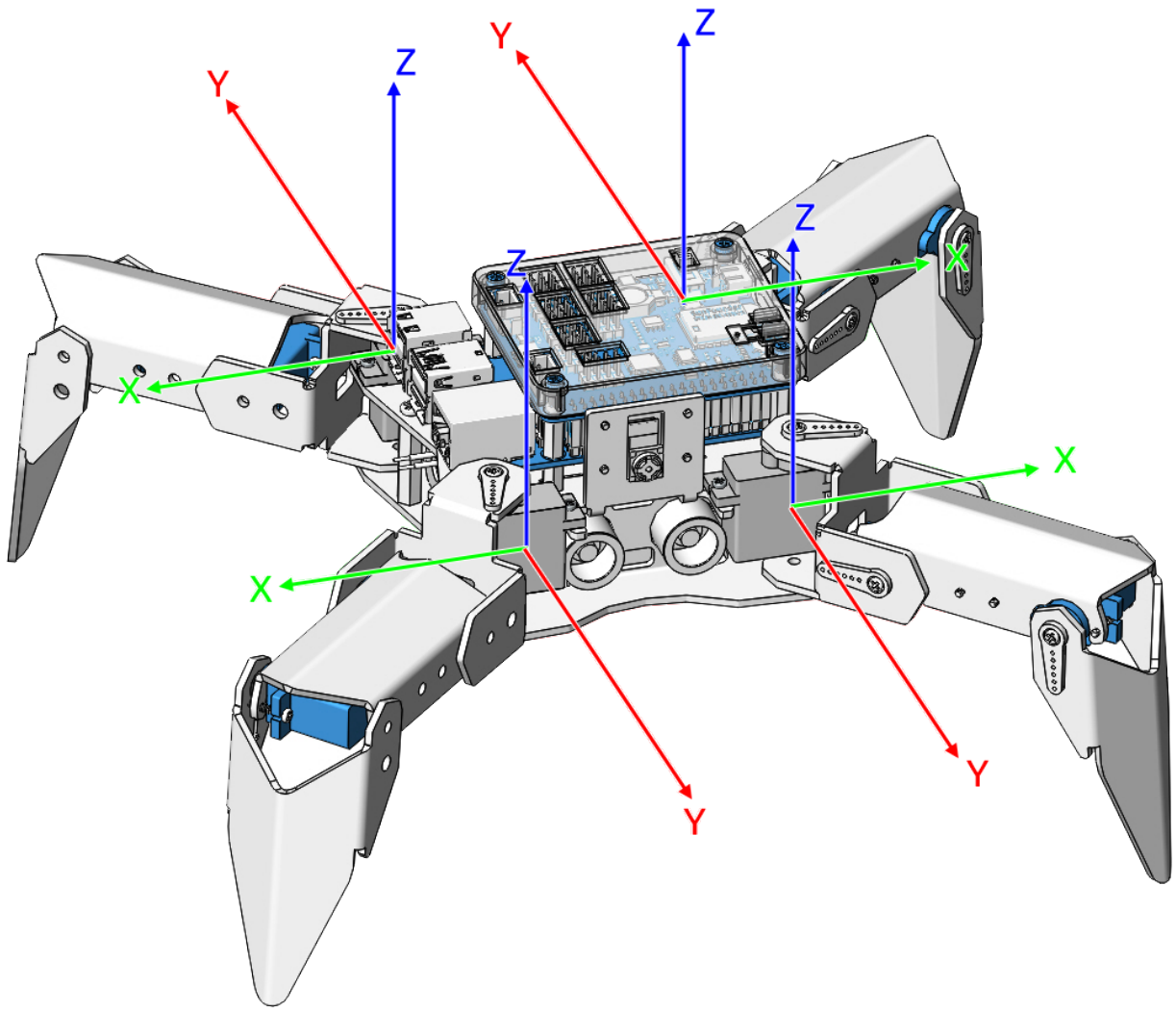
One: It can write strings, directly use the `step_list` dictionary in the `picrawler` library.

```
crawler.do_step('stand',speed)
# "speed" indicates the speed of the step, the range is 0~100.
```

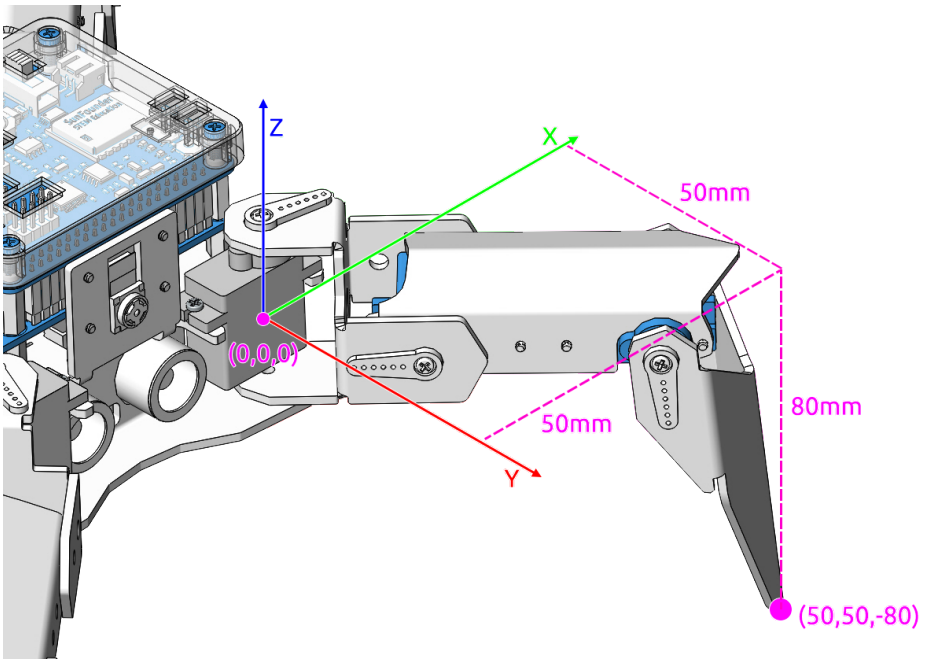
Second: It can also write an array of 4 coordinate values.

```
crawler.do_step([[50, 50, -80], [50, 50, -80],[80, 80, 0], [50, 50, -80]],speed)
# These four coordinates are used to control the four legs of right front, left front,
↪ left rear, and left rear respectively.
```

Each foot has an independent coordinate system. As shown below:



You need to measure the coordinates of each toe individually. As shown below:



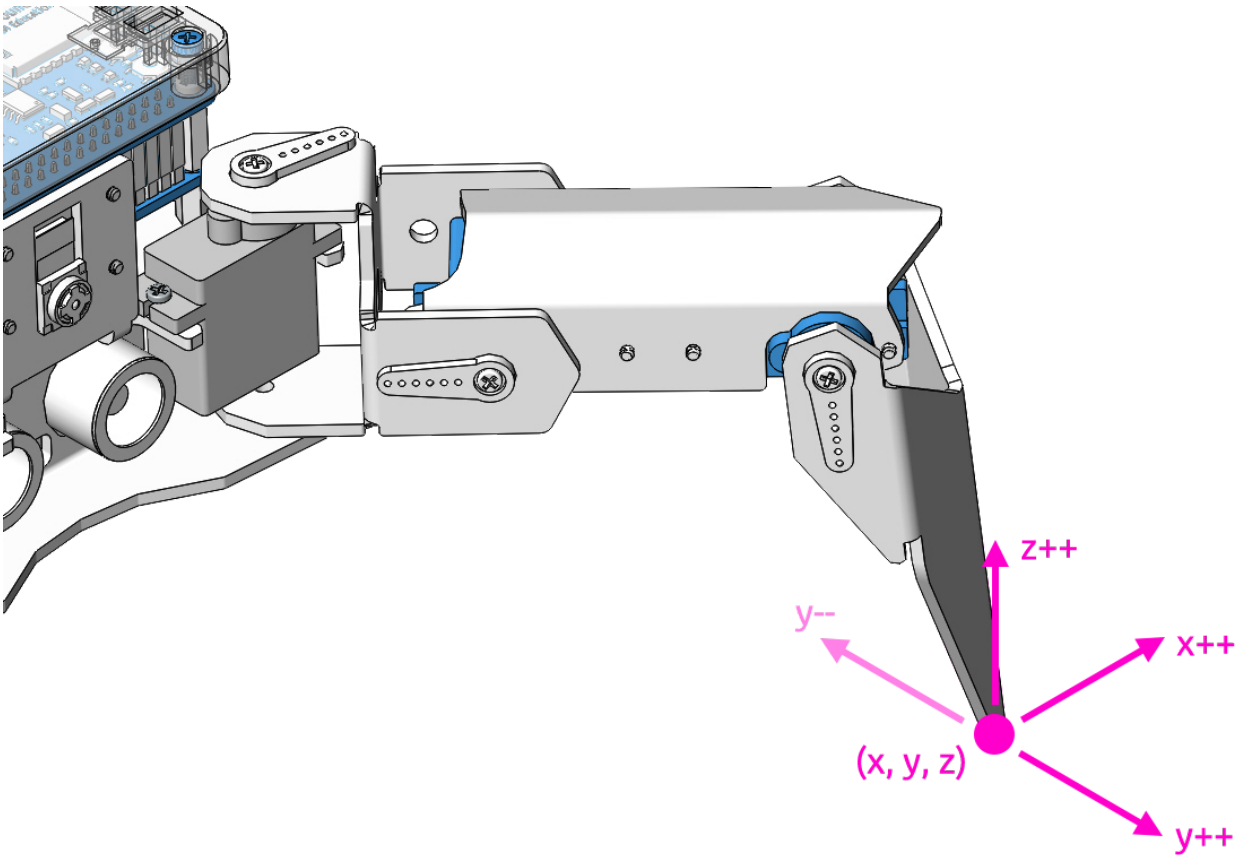
By the way: the `step_list` called in the first method also consists of an array containing 4 coordinate values.

```
step_list = {
    "stand":[
        [50, 50, -80],
        [50, 50, -80],
        [50, 50, -80],
        [50, 50, -80]
    ],
    "sit":[
        [50, 50, -33],
        [50, 50, -33],
        [50, 50, -33],
        [50, 50, -33]
    ],
}
```

3.12 Adjust Posture

In this example, we use the keyboard to control the PiCrawler foot by foot and assume the desired posture.

You can press the space bar to print out the current coordinate values. These coordinate values come in handy when you create unique actions for PiCrawler.



Run the Code

```
cd /home/pi/picrawler/examples
sudo python3 do_single_leg.py
```

After the code runs, please operate according to the prompt that pops up in the terminal.

- Press `1234` to select the feet separately, `1`: right front foot, `2`: left front foot, `3`: left rear foot, `4`: right rear foot
- Press `w`, `a`, `s`, `d`, `r`, and `f` to slowly control the PiCrawler's coordinate values.
- Press `space` to print all coordinate values.
- Press `esc` to exit.

Code

```
from picrawler import Picrawler
from time import sleep
import readchar

crawler = Picrawler([10,11,12,4,5,6,1,2,3,7,8,9])
speed = 80


manual = '''
-------- PiCrawler Controller ---------
    .......           .......
    <=|   2   |--|    1   |=>
    '''.......' ├       '.......'
```

(continues on next page)

(continued from previous page)

```
    ....... ├       .......
    <=|   3   |└------|   4   |=>
    ^^^^^^^           ^^^^^^^
    1: Select right front leg
    2: Select left front leg
    3: Select left rear leg
    4: Select right rear leg
    W: Y++          R: Z++
    A: X--          F: Z--
    S: Y--
    D: X++          Esc: Quit
'''
legs_list = ['right front', 'left front', 'left rear', 'right rear']

def main():
    leg = 0
    speed = 80
    step = 2
    print(manual)
    crawler.do_step('stand',speed)
    sleep(0.2)
    coordinate=crawler.current_step_all_leg_value()

    def show_info():
        print("\033[H\033[J",end='')  # clear terminal windows
        print(manual)
        print('%s : %s'%(leg+1, legs_list[leg]))
        print('coordinate: %s'%(coordinate))

    show_info()

    while True:
        # readkey
        key = readchar.readkey()
        key = key.lower()
        # select the leg
        if key in ('1234'):
            leg = int(key) - 1
            show_info()
        # move
        elif key in ('wsadrf'):
            if 'w' == key:
                coordinate[leg][1]=coordinate[leg][1] + step
            elif 's' == key:
                coordinate[leg][1]=coordinate[leg][1] - step
            elif 'a' == key:
                coordinate[leg][0]=coordinate[leg][0] - step
            elif 'd' == key:
                coordinate[leg][0]=coordinate[leg][0] + step
            elif 'r' == key:
                coordinate[leg][2]=coordinate[leg][2] + step
            elif 'f' == key:
                coordinate[leg][2]=coordinate[leg][2] - step

            crawler.do_single_leg(leg,coordinate[leg],speed)
            sleep(0.1)
            # coordinate=crawler.current_step_all_leg_value()
```

(continues on next page)

(continued from previous page)

```
            show_info()
        # quit
        elif key == readchar.key.CTRL_C or key in readchar.key.ESCAPE_SEQUENCES:
            print("\n Quit")
            break

        sleep(0.05)


if __name__ == "__main__":
    main()
```

- `current_step_all_leg_value()` : Returns the coordinate values of all legs.
- `do_single_leg(leg,coordinate,speed)` : Modify the coordinate value of a certain leg individually.

3.13 Record New Step

We use the keyboard to control PiCrawler to make several poses in turn, and record these poses. Replay them later.

Run the Code

```
cd /home/pi/picrawler/examples
sudo python3 record_new_step_by_keyboard.py
```

After the code runs, please operate according to the prompt that pops up in the terminal.

- Press `1234` to select the feet separately, `1`: right front foot, `2`: left front foot, `3`: left rear foot, `4`: right rear foot
- Press `w`, `a`, `s`, `d`, `r`, and `f` to slowly control the PiCrawler's coordinate values.
- Press `space` to print all coordinate values.
- Press `p` to have PiCrawler replay the recorded action.
- Press `esc` to exit.

Code

```
from picrawler import Picrawler
from time import sleep
import sys
import tty
import termios
import copy

crawler = Picrawler([10,11,12,4,5,6,1,2,3,7,8,9])
#crawler.set_offset([0,0,0,0,0,0,0,0,0,0,0,0])
speed = 80

def readchar():
    fd = sys.stdin.fileno()
    old_settings = termios.tcgetattr(fd)
    try:
        tty.setraw(sys.stdin.fileno())
        ch = sys.stdin.read(1)
    finally:
```

(continues on next page)

(continued from previous page)

```
        termios.tcsetattr(fd, termios.TCSADRAIN, old_settings)
    return ch


manual = '''
Press keys on keyboard to control PiSloth!
    w: Y++
    a: X--
    s: Y--
    d: X++
    r: Z++
    f: Z--
    1: Select right front leg
    2: Select left front leg
    3: Select left rear leg
    4: Select right rear leg
    space: Print all leg coodinate & Save this step
    p: Play all saved step
    esc: Quit
'''


new_step=[]

def save_new_step():
    new_step.append(copy.deepcopy(crawler.current_step_all_leg_value()))
    print(new_step)

def play_all_new_step():
    for step in new_step:
        crawler.do_step(step,speed)
        sleep(0.6)

def main():

    speed = 80
    print(manual)
    crawler.do_step('sit',speed)
    leg = 0
    coodinate=crawler.current_step_leg_value(leg)
    while True:
        key = readchar()
        key = key.lower()
        # print(key)
        if 'w' == key:
            coodinate[1]=coodinate[1]+2
        elif 's' == key:
            coodinate[1]=coodinate[1]-2
        elif 'a' == key:
            coodinate[0]=coodinate[0]-2
        elif 'd' == key:
            coodinate[0]=coodinate[0]+2
        elif 'r' == key:
            coodinate[2]=coodinate[2]+2
        elif 'f' == key:
            coodinate[2]=coodinate[2]-2
        elif '1' == key:
```

(continues on next page)

(continued from previous page)

```
            leg=0
            coodinate=crawler.current_step_leg_value(leg)
        elif '2' == key:
            leg=1
            coodinate=crawler.current_step_leg_value(leg)
        elif '3' == key:
            leg=2
            coodinate=crawler.current_step_leg_value(leg)
        elif '4' == key:
            leg=3
            coodinate=crawler.current_step_leg_value(leg)
        elif chr(32) == key:
            print("[[right front],[left front],[left rear],[right rear]]")
            print("saved new step")
            print(crawler.current_step_all_leg_value())
            save_new_step()
        elif 'p' == key:
            play_all_new_step()
        elif chr(27) == key:# 27 for ESC
            break

        sleep(0.05)
        crawler.do_single_leg(leg,coodinate,speed)
    print("\n q Quit")


if __name__ == "__main__":
    main()
```

How it works?

This project was born out of *Adjust Posture*. Added recording and replay functions.

The recording function is implemented by the following code.

```
new_step=[]

def save_new_step():
    new_step.append(copy.deepcopy(crawler.current_step_all_leg_value()))
    print(new_step)
```

Note: The assignment here needs to use the Deep Copy function, otherwise the `new_step` will not get a new array object when appending.

The replay function is implemented by the following code.

```
def play_all_new_step():
    for step in new_step:
        crawler.do_step(step,speed)
        sleep(0.6)
```

3.14 Twist

We already know how to make PiCrawler assume a specific pose, the next step is to combine the poses to form a continuous action.

Here, PiCrawler's four feet are up and down in twos, jumping with the music.

Run the Code

```
cd /home/pi/picrawler/examples
sudo python3 twist.py
```

Code

Note: You can **Modify/Reset/Copy/Run/Stop** the code below. But before that, you need to go to source code path like `picrawler\examples`. After modifying the code, you can run it directly to see the effect.

```
from picrawler import Picrawler
from time import sleep
from robot_hat import Music

music = Music()
crawler = Picrawler([10,11,12,4,5,6,1,2,3,7,8,9])
#crawler.set_offset([0,0,0,0,0,0,0,0,0,0,0,0])


def twist(speed):
     ## [right front],[left front],[left rear],[left rear]
    new_step=[[50, 50, -80], [50, 50, -80],[50, 50, -80], [50, 50, -80]]
    for i in range(4):
        for inc in range(30,60,5):
            rise = [50,50,(-80+inc*0.5)]
            drop = [50,50,(-80-inc)]

            new_step[i]=rise
            new_step[(i+2)%4] = drop
            new_step[(i+1)%4] = rise
            new_step[(i-1)%4] = drop
            crawler.do_step(new_step,speed)

def main():

    music.background_music('./musics/sports-Ahjay_Stelino.mp3')
    music.music_set_volume(20)

    while True:
        twist(speed=100)


if __name__ == "__main__":
    main()
```

How it works?

In this code, you need to pay attention to this part:

```
def twist(speed):
    ## [right front],[left front],[left rear],[right rear]
    new_step=[[50, 50, -80], [50, 50, -80],[50, 50, -80], [50, 50, -80]]
    for i in range(4):
        for inc in range(30,60,5):
            rise = [50,50,(-80+inc*0.5)]
            drop = [50,50,(-80-inc)]

            new_step[i]=rise
            new_step[(i+2)%4] = drop
            new_step[(i+1)%4] = rise
            new_step[(i-1)%4] = drop
            crawler.do_step(new_step,speed)
```

Simply put, it uses two layers of for loops to make the `new_step` array produce continuous and regular changes, and at the same time, `crawler.do_step()` executes the posture to form a continuous action.

You can intuitively get the coordinate value array corresponding to each pose from *Adjust Posture*.

In addition, the example also played background music. The implementation method is as follows.

Play music by importing the following libraries.

```
from robot_hat import Music
```

Declare a Music object.

```
music = Music()
```

Play the background music in the `picrawler/examples/musics` directory and set the volume to 20. You can also add music to the `musics` folder via *Filezilla Software*.

```
music.background_music('./musics/sports-Ahjay_Stelino.mp3')
music.music_set_volume(20)
```

Note: You can add different sound effects or music to `musics` or `sounds` folder via *Filezilla Software*.

3.15 Emotional Robot

This example shows several interesting custom actions of PiCrawler.

Run the Code

```
cd /home/pi/picrawler/examples
sudo python3 emotional_robot.py
```

Code

Note: You can **Modify/Reset/Copy/Run/Stop** the code below. But before that, you need to go to source code path like `picrawler\examples`. After modifying the code, you can run it directly to see the effect.

```
from picrawler import Picrawler
from time import sleep

crawler = Picrawler([10,11,12,4,5,6,1,2,3,7,8,9])
#crawler.set_offset([0,0,0,0,0,0,0,0,0,0,0,0])

def handwork(speed):

    basic_step = []
    basic_step = crawler.step_list.get("sit")
    left_hand  = crawler.mix_step(basic_step,0,[0,50,80])
    right_hand  = crawler.mix_step(basic_step,1,[0,50,80])
    two_hand = crawler.mix_step(left_hand,1,[0,50,80])

    crawler.do_step('sit',speed)
    sleep(0.6)
    crawler.do_step(left_hand,speed)
    sleep(0.6)
    crawler.do_step(two_hand,speed)
    sleep(0.6)
    crawler.do_step(right_hand,speed)
    sleep(0.6)
    crawler.do_step('sit',speed)
    sleep(0.6)

def twist(speed):

    new_step=[[50, 50, -80], [50, 50, -80],[50, 50, -80], [50, 50, -80]]
    for i in range(4):
        for inc in range(30,60,5):
            rise = [50,50,(-80+inc*0.5)]
            drop = [50,50,(-80-inc)]

            new_step[i]=rise
            new_step[(i+2)%4] = drop
            new_step[(i+1)%4] = rise
            new_step[(i-1)%4] = drop
            crawler.do_step(new_step,speed)

##"[[right front], [left front], [left rear], [left rear]]")

def pushup(speed):
    up=[[80, 0, -100], [80, 0, -100],[0, 120, -60], [0, 120, -60]]
    down=[[80, 0, -30], [80, 0, -30],[0, 120, -60], [0, 120, -60]]
    crawler.do_step(up,speed)
    sleep(0.6)
    crawler.do_step(down,speed)
    sleep(0.6)

def swimming(speed):
    for i in range(100):
        crawler.do_step([[100-i,i,0],[100-i,i,0],[0,120,-60+i/5],[0,100,-40-i/5]],
↪speed)

# main
def main():
    speed = 100
```

(continues on next page)

(continued from previous page)

```
    swimming(speed)
    pushup(speed)
    handwork(speed)
    twist(speed)

    sleep(0.05)

if __name__ == "__main__":
    main()
```

CHAPTER

FOUR

PLAY WITH EZBLOCK

For beginners and novices, EzBlock is a software development platform offered by SunFounder for Raspberry Pi. Ezbock offers two programming environments: a graphical environment and a Python environment.

It is available for almost all types of devices, including Mac, PC, and Android.

Here is a tutorial to help you complete EzBlock installation, download, and use.

4.1 Quick Guide on EzBlock

There are 2 parts here:

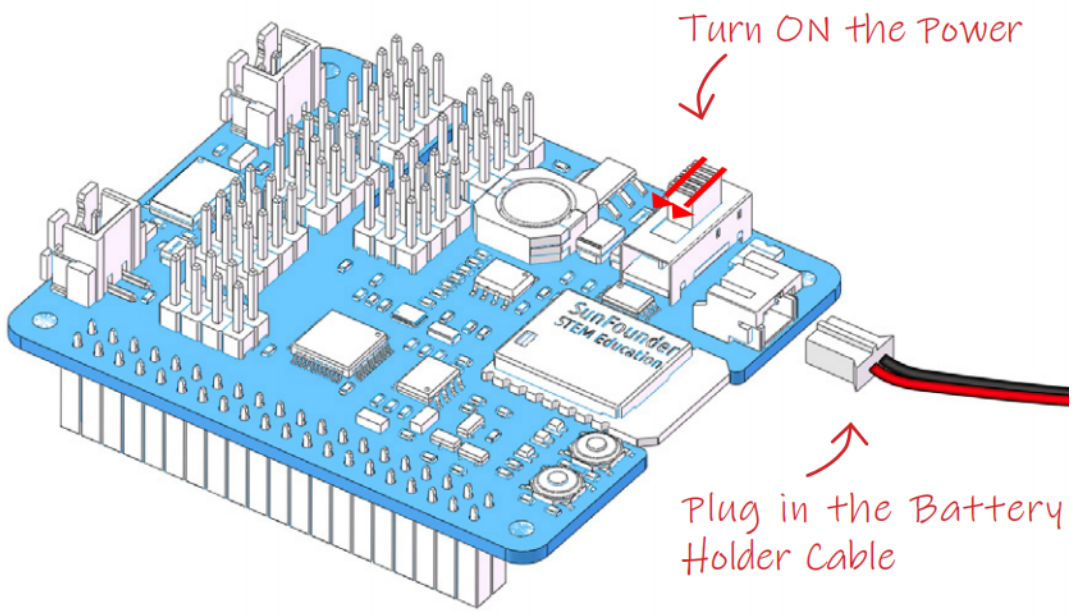
- *Servo Adjust* allows you to keep all the servos at 0 degrees to complete a proper and safe assembly (otherwise you will probably damage the servos).
- *Install and Configure EzBlock Studio* will guide you to download EzBlock Studio to play with your robot.

4.1.1 Servo Adjust

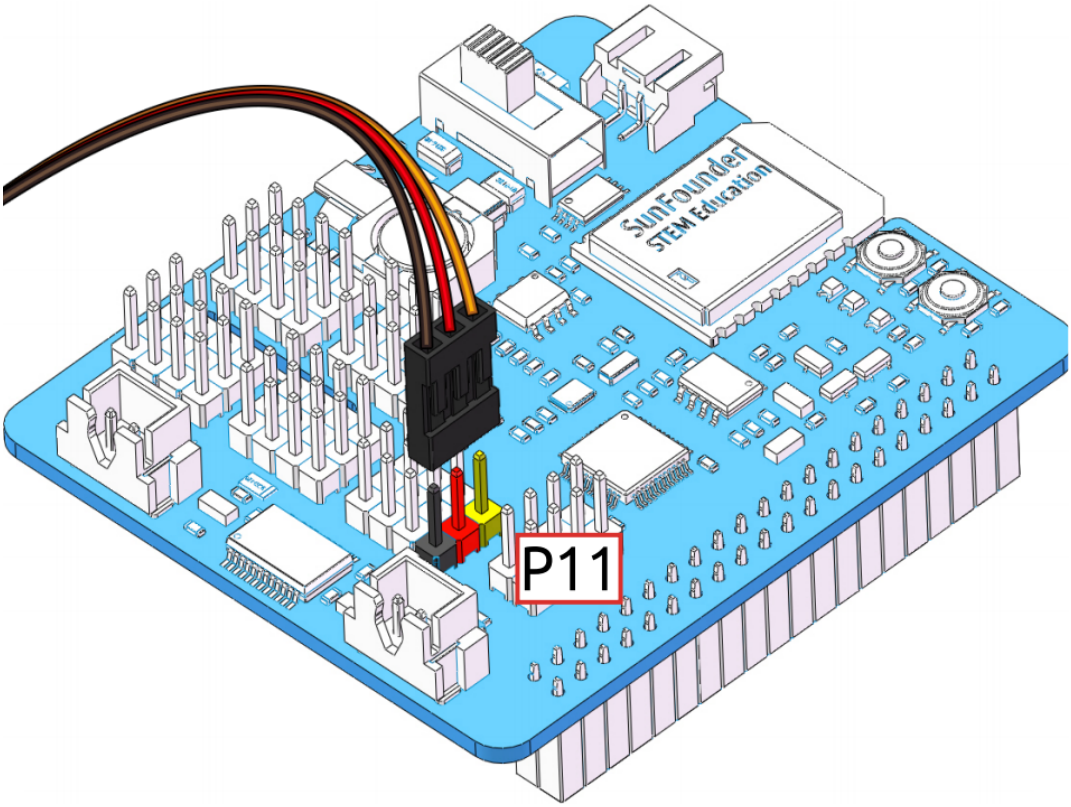
When assembling to the part with the servo, you need to keep the servo at 0° and secure it with the servo screw. Please follow the tutorial below to do this.

1. Firstly, Install EzBlock OS(3.1) onto a Micro SD card, once the installation is complete, insert it into the Raspberry Pi.
2. To ensure that the servo has been properly set to 0°, first insert the rocker arm into the servo shaft and then gently rotate the rocker arm to a different angle.

3. Follow the instructions on the assembly foldout, insert the battery holder cable and turn the power switch to the ON. Wait for 1-2 minutes, there will be a sound to indicate that the Raspberry Pi boots successfully.



4. Next, plug the servo cable into the P11 port as follows.



5. At this point you will see the servo arm rotate to a specific position (0°). If the servo arm does not return to 0°, press the RST button to restart the Robot HAT.
6. Now you can continue the installation as instructed on the assembly foldout.

Note:

- Do not unplug this servo cable before fastening this servo with the servo screw, you can unplug it after fastening.
- Do not turn the servo while it is powered on to avoid damage; if the servo shaft is inserted at the wrong angle, pull out the servo and reinsert it.
- Before assembling each servo, you need to plug the servo cable into P11 and turn on the power to set its angle to 0°.
- This zeroing function will be disabled if you download a program to the robot later with the EzBlock APP.

4.1.2 Install and Configure EzBlock Studio

As soon as the robot is assembled, you will need to carry out some basic operations.

- Install EzBlock Studio(3.1): Download and install EzBlock Studio on your device or use the web-based version.
- Connect the Product and EzBlock(3.1): Configure Wi-Fi, Bluetooth and calibrate before use.
- Open and Run Examples(3.1): View or run the related example directly.

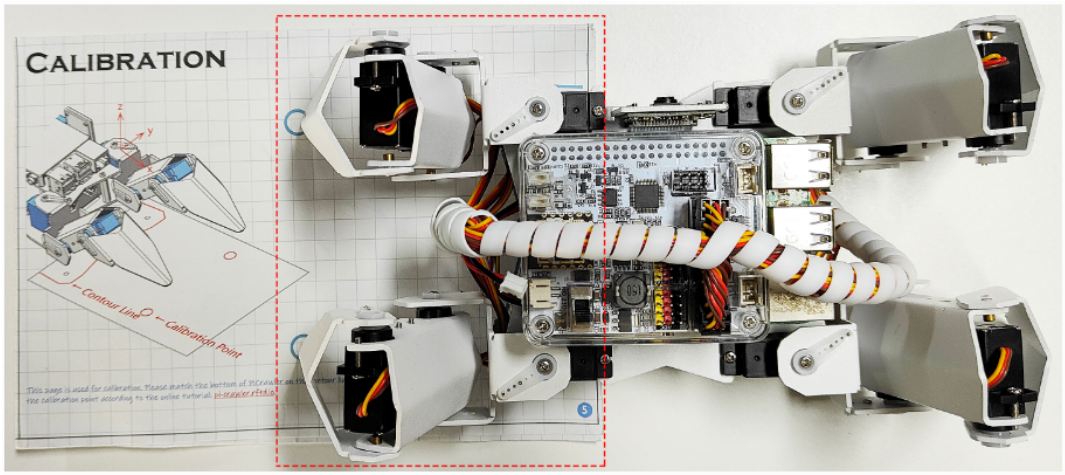
4.2 Calibrate the PiCrawler

Note: After you connect the PiCrawler, there will be a calibration step. This is because of possible deviations in the installation process or limitations of the servos themselves, making some servo angles slightly tilted, so you can calibrate them in this step.

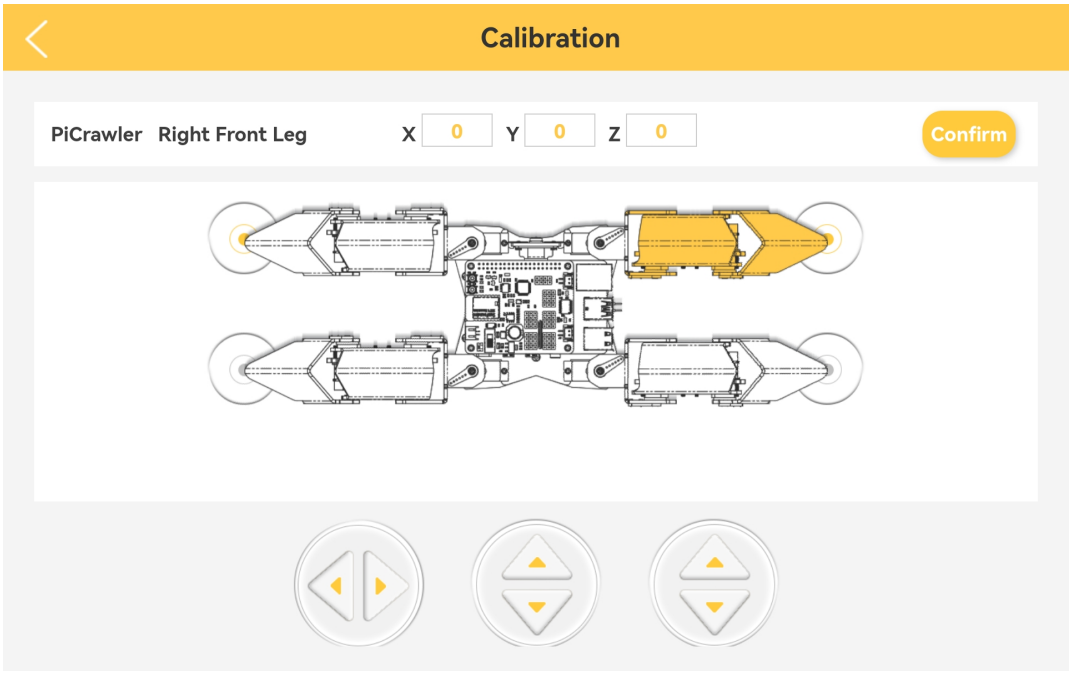
But if you think the assembly is perfect and no calibration is needed, you can also skip this step.

The calibration steps are as follows:

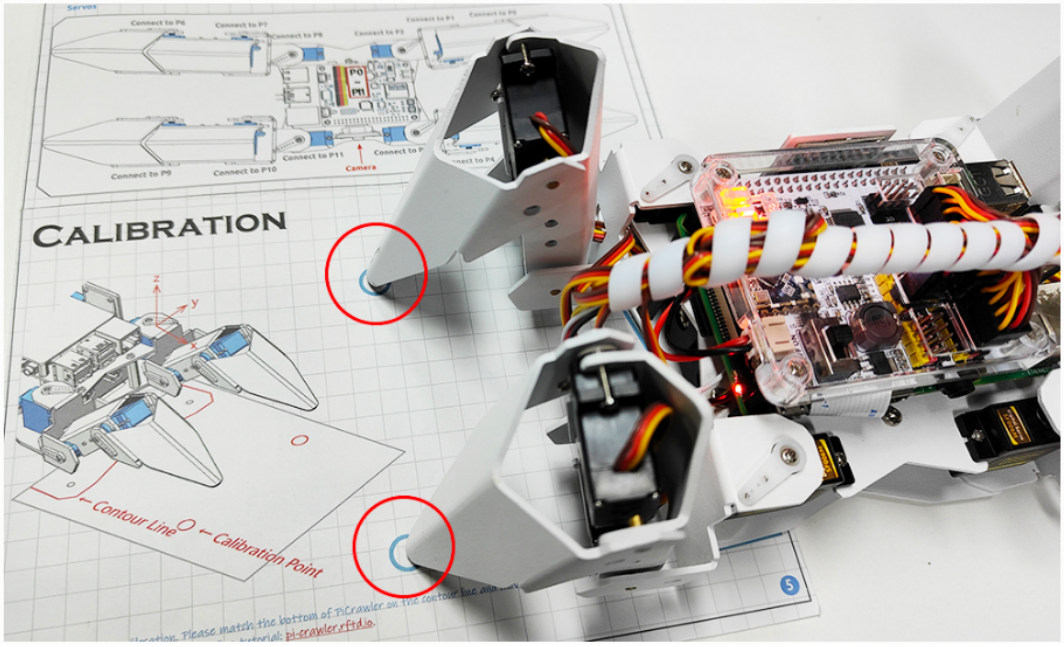
1. Take out the assembly leaflet, turn it to the last page, and lay it flat on the table. Then place the PiCrawler as shown below, aligning its bottom with the outline on the calibration chart.



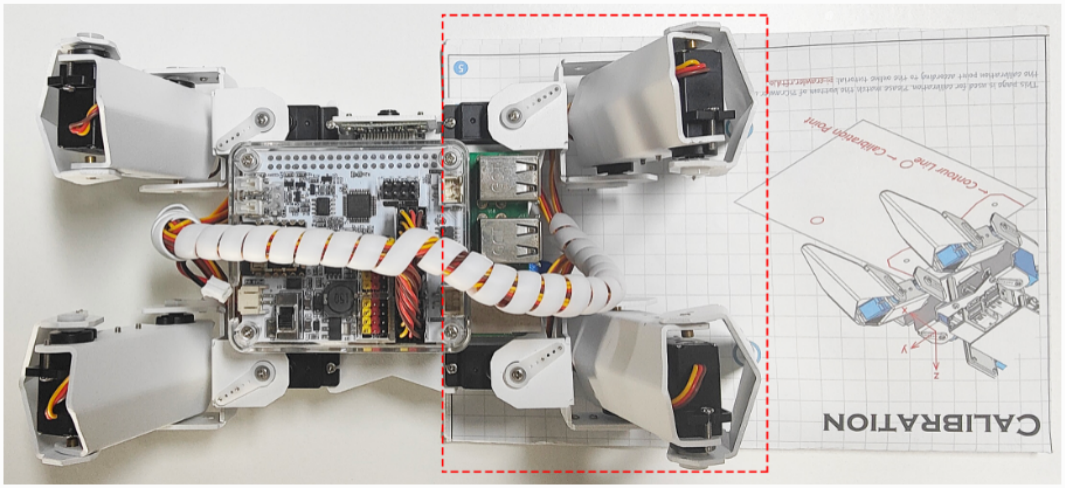
2. Go back to EzBlock Studio, select one foot on the left, then click the 3 sets of X, Y and Z buttons, and let the toes slowly align with the calibration point.
    - The calibration buttons are used for fine-tuning, and you need to press these buttons multiple times to see the pin position change.
    - It is recommended to click the up button of Z axis to lift the foot up first, then go to adjust X and Y.



3. Align the other foot on the left in the same way.



4. After calibrating the left two feet, change the calibration paper to the right, and calibrate the right two feet according to the above method.



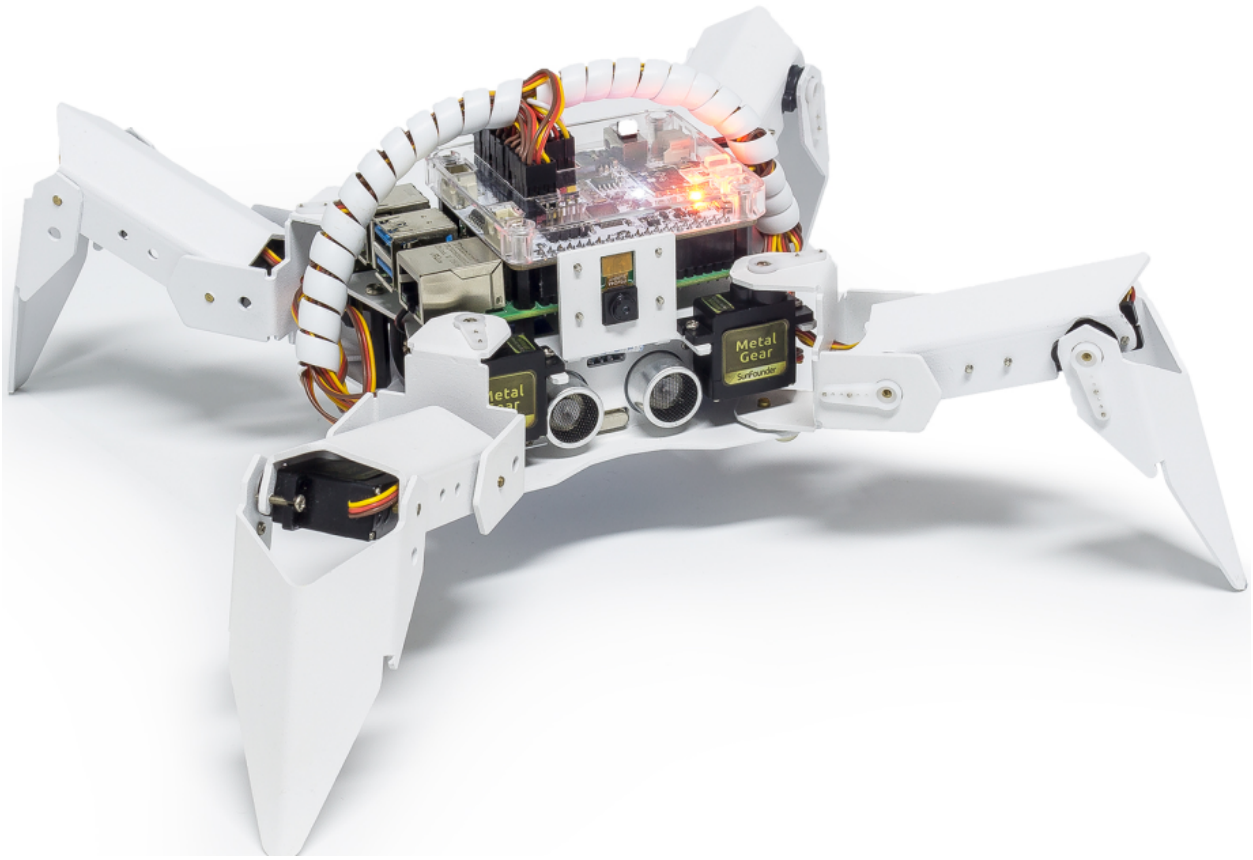
Projects

Here, we show you the projects of playing PiCrawler on Ezblock Studio. If you are new to these, you can refer to the code images inside each project to program, and can learn the use of blocks according to TIPS.

If you don't want to write these projects one by one, we have uploaded them to Ezblock Studio's Examples page and you can run them directly or edit them and run them later.

4.3 Move

This is PiCrawler's first project. Perform its most basic function - move.



Program

Note:

- You can write the program according to the following picture, please refer to the tutorial: How to Create a New Project?.
- Or find the code with the same name on the **Examples** page of the EzBlock Studio and click **Run** or **Edit** directly.

Click the Upload & Run button at the bottom right of the screen, and PiCrawler will execute “forward” and “backward” actions in sequence.

How it works?

First, you need to understand the program framework of Ezblock. as follows:

All Ezblock projects contain these two blocks. The **Start** block runs at the beginning of the program and is executed only once, and is often used to set variables; the **Forever** block runs after **Start**, and will be executed repeatedly, and is often used to implement main functions. If you delete these two blocks, you can drag them back from the **Basic** category on the left.

Next you need to understand the following blocks.

do action allows PiCrawler to perform basic actions. You can modify the options in the first groove. For example, select "Turn Left", "Back" and so on. The second groove can set the number of executions of the action, and only integer numbers greater than 0 can be written. The third groove can set the speed of the action, and only integers within 0~100 can be written.

do step is similar to **do action**, but it is not an action but a static posture. Such as "stand", "sit".

Both blocks can be dragged from the **PiCrawler** category on the left.

4.4 Remote Control

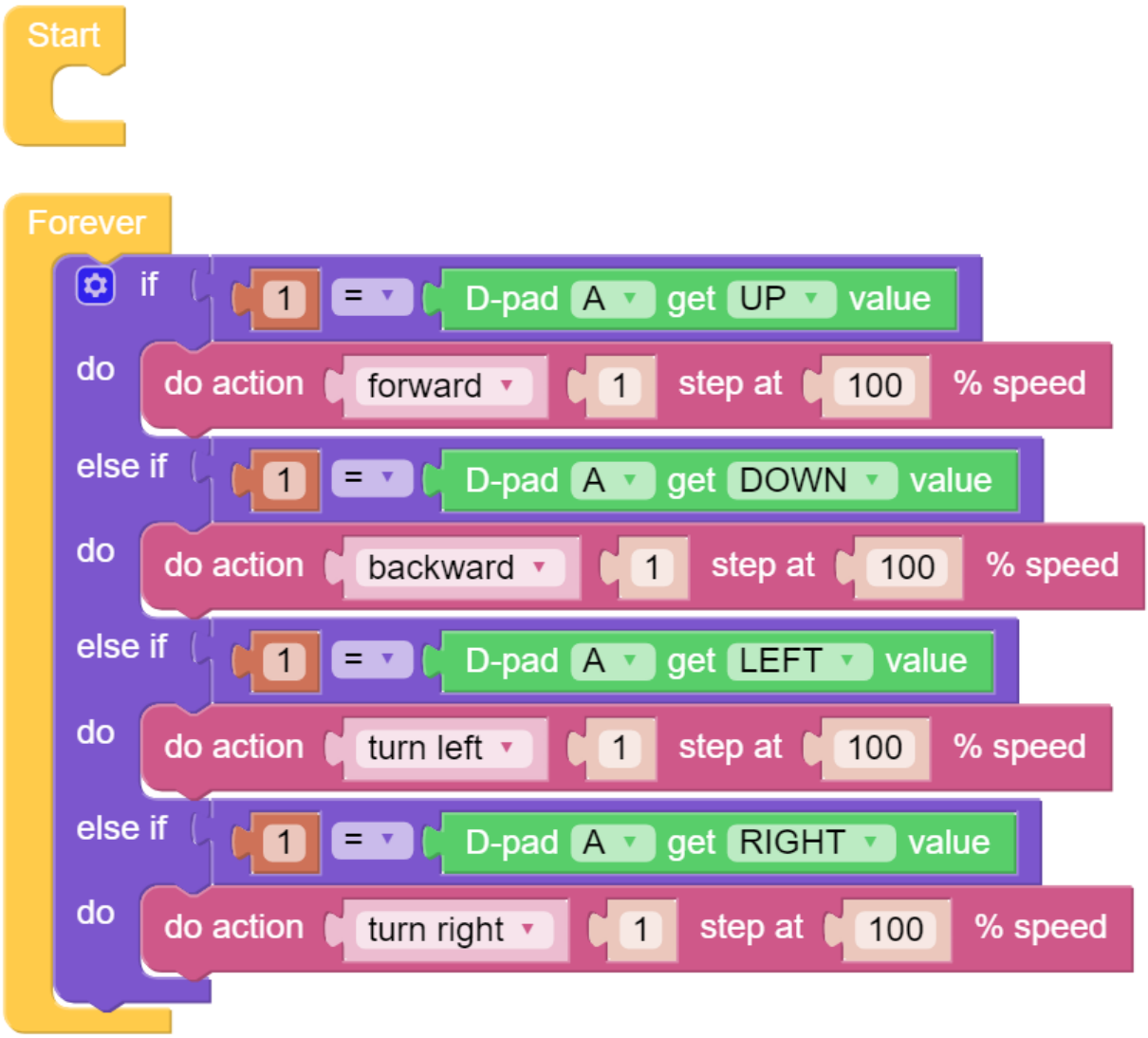
In this project, we will learn how to remote control the PiCrawler. You can control the PiCrawler to move forward, backward, left, and right.

Note: You can refer to How to Use the Remote Control Function?. Come and carry out this project smoothly.

Program

Note:

- You can write the program according to the following picture, please refer to the tutorial: How to Create a New Project?.
- Or find the code with the same name on the **Examples** page of the EzBlock Studio and click **Run** or **Edit** directly.



Switch to the Remote Control interface, and you will see the following widgets.

After the program is running, you can activate PiCrawler through D-Pad.

How it works?

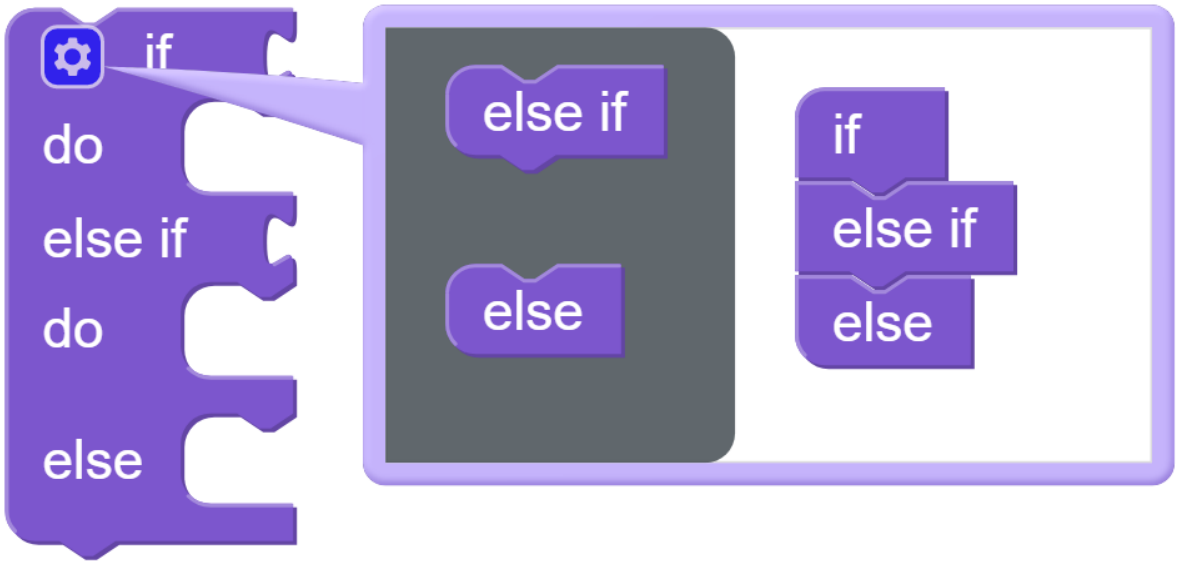
After dragging out the widget on the Remote Control interface, a category named **Remote** will appear in the block categories column of the programming interface.

Here we add the D-Pad widget, so the **D-Pad get value** block appears here.

D-Pad can be regarded as a four-in-one button. You can choose which button to read in the second groove of the block.

When the button is pressed, the value is “1”; when the button is not pressed, the value is “0”.

We used an **if** block (you can find it in the **Logic** category on the left) to make the PiCrawler move forward once when the **UP** button of the D-pad is pressed.



You can click the gear icon on the upper left of the block to modify the shape of the **if** block to realize multiple judgment branches.

if block is usually used with **=** block, **=** block can be modified to **>**, **<** and other conditions through the drop-down menu, please use it flexibly.

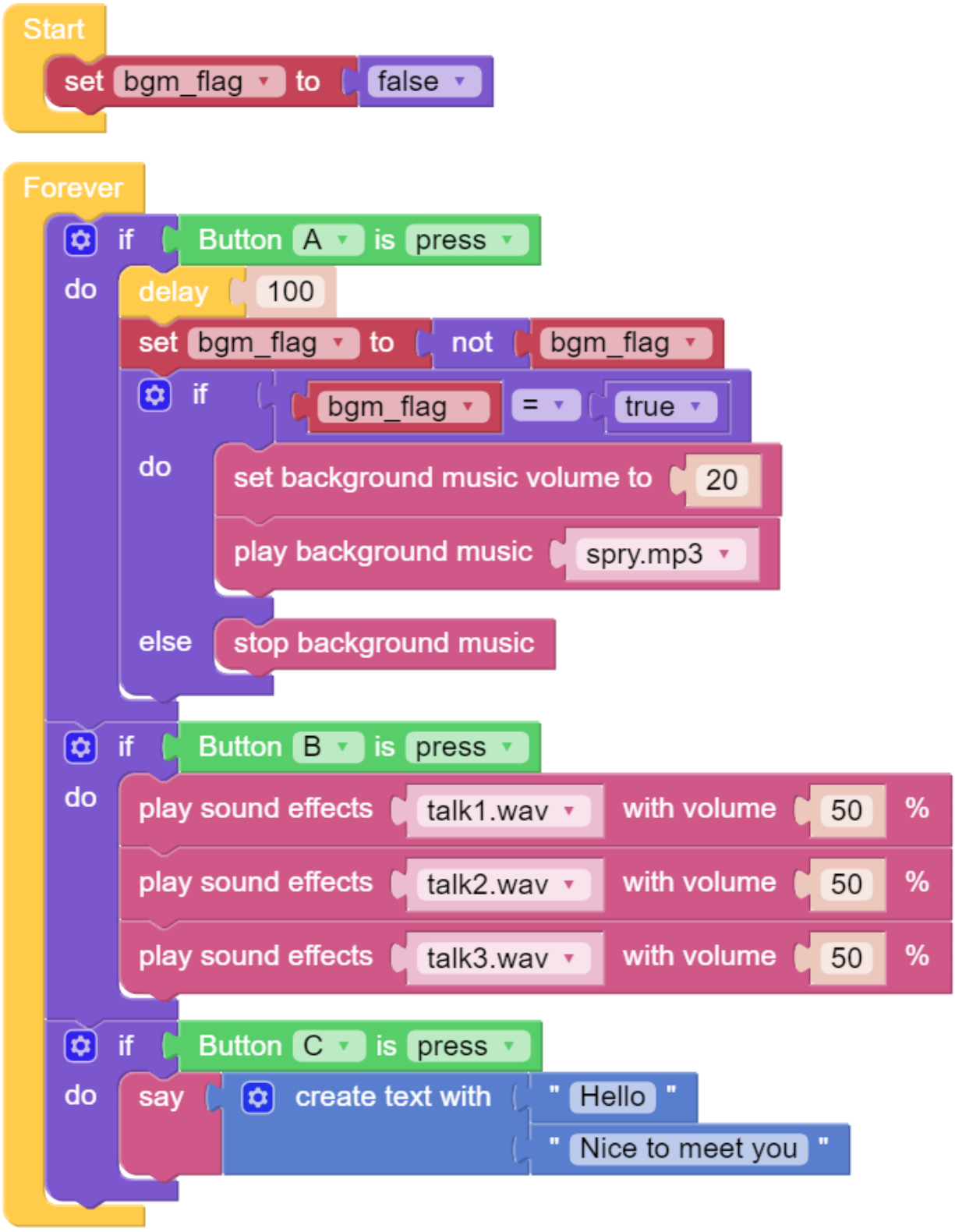
4.5 Sound Effect

In this example, we use PiCrawler's (to be precise, Robot HAT's) sound effects. It consists of three parts, namely **Muisc**, **Sound**, **Text to Speech**.

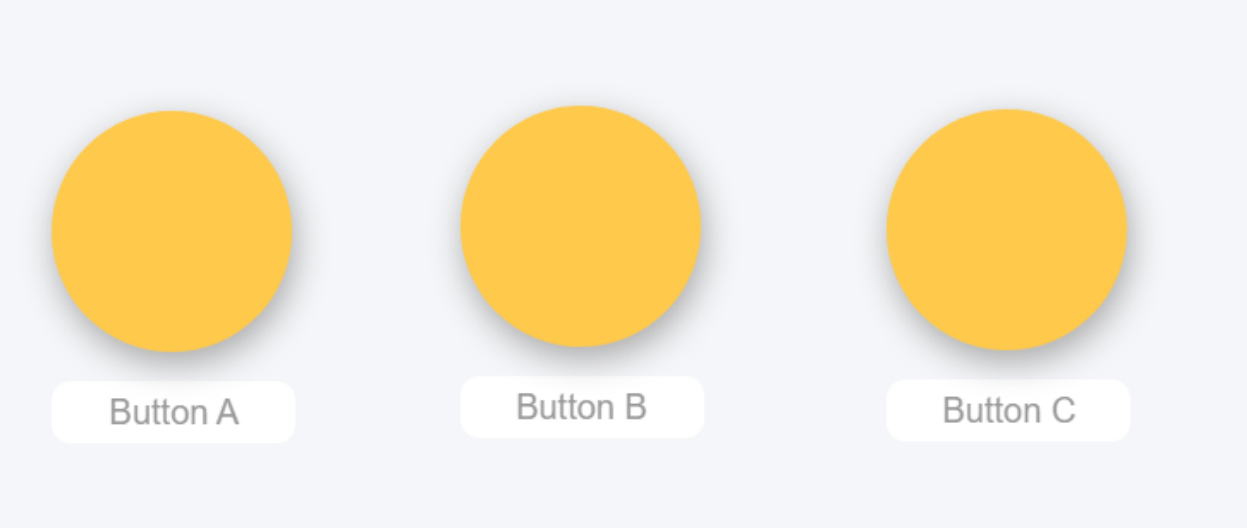
Program

Note:

- You can write the program according to the following picture, please refer to the tutorial: How to Create a New Project?.
- Or find the code with the same name on the **Examples** page of the EzBlock Studio and click **Run** or **Edit** directly.



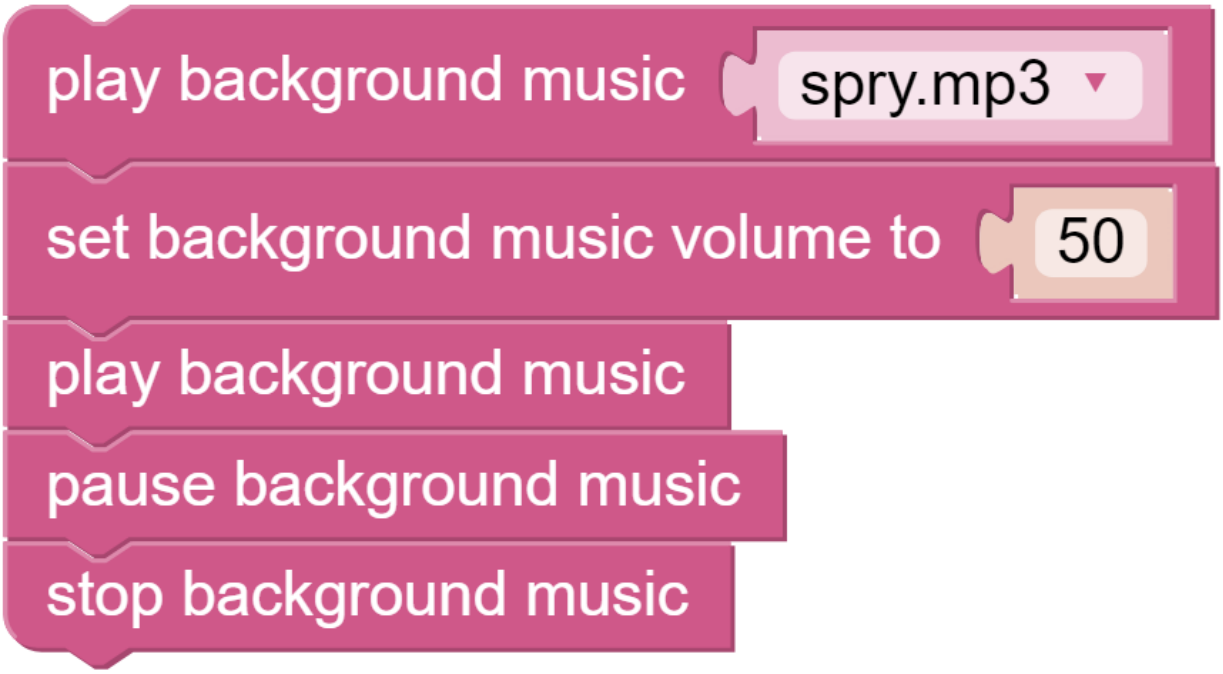
Switch to the Remote Control interface, and you will see the following widgets.



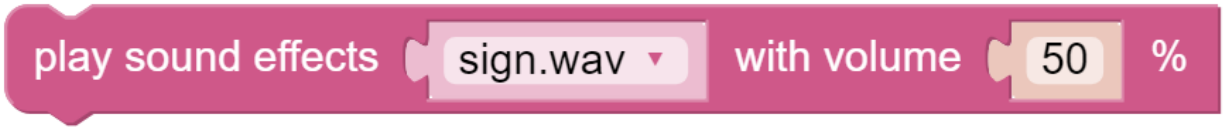
After the program is running, you can press different buttons to make PiCrawler sound.

How it works?

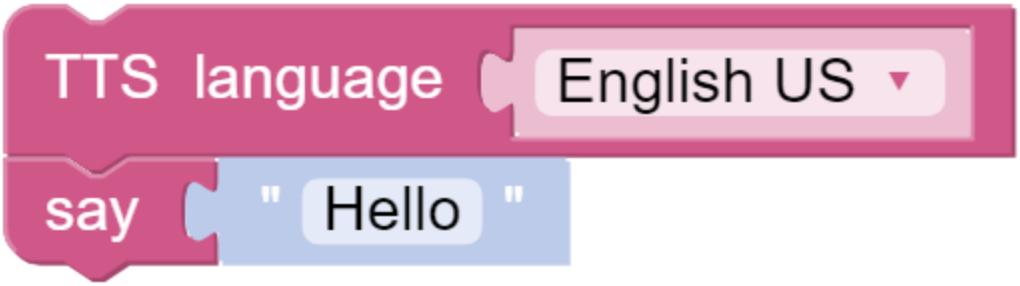
Functions related to background music include these:



Functions related to sound effects include these:

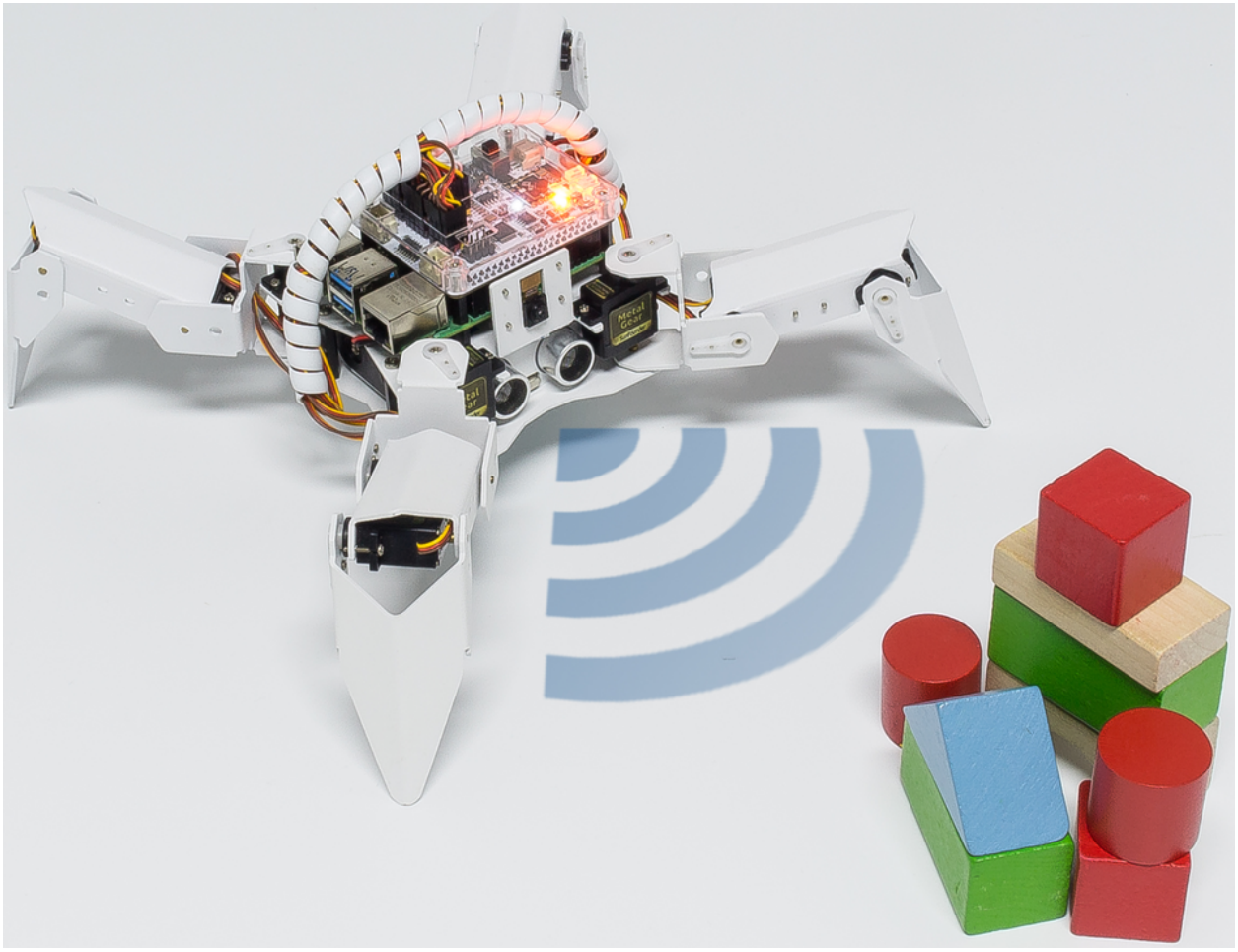


Functions related to Text to Speech include these:



4.6 Obstacle Avoidance

In this project, picrawler will use an ultrasonic module to detect obstacles in front. When PiCrawler detects an obstacle, it will send a signal and look for another direction to move forward.

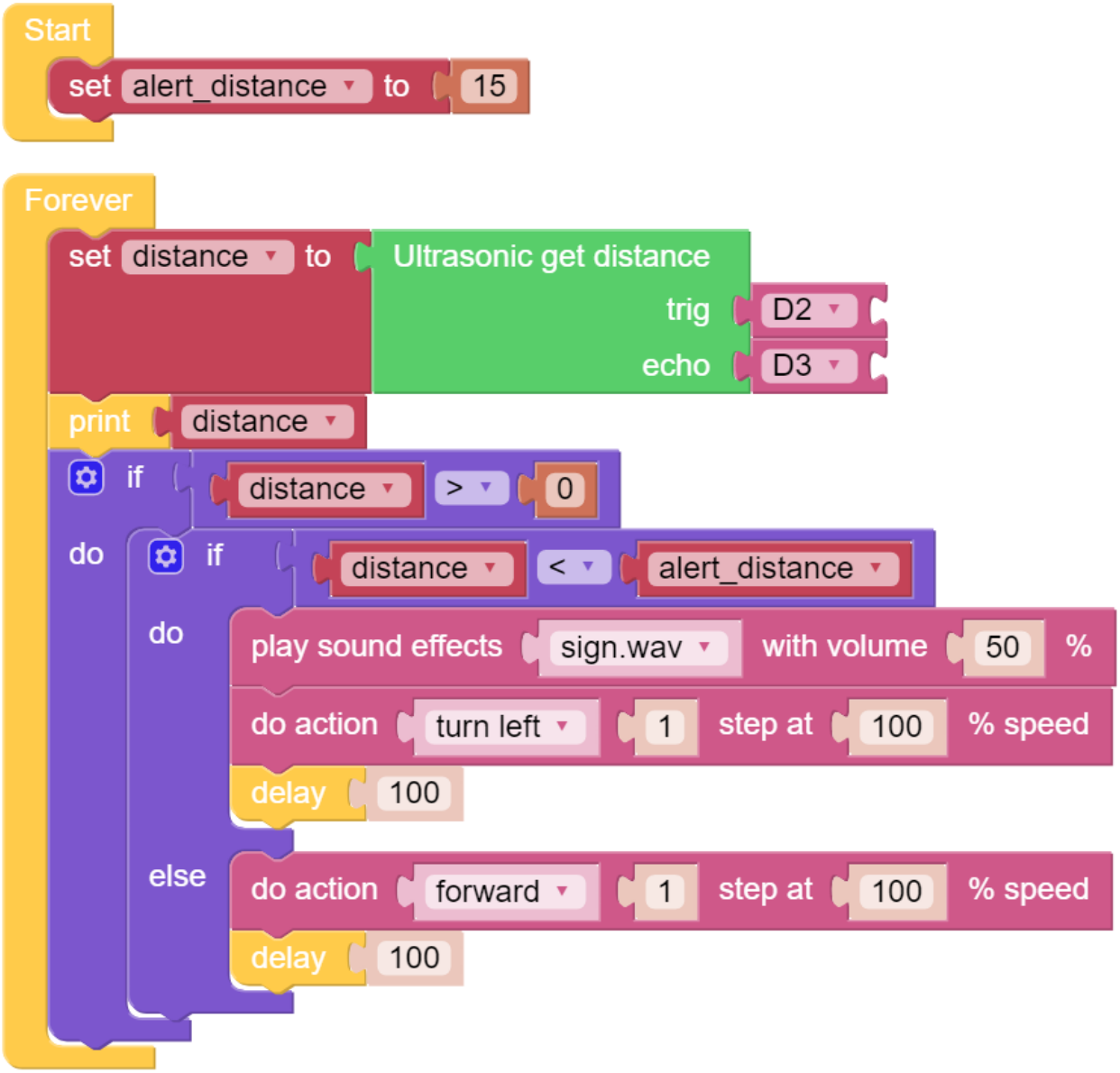


Program

Note:

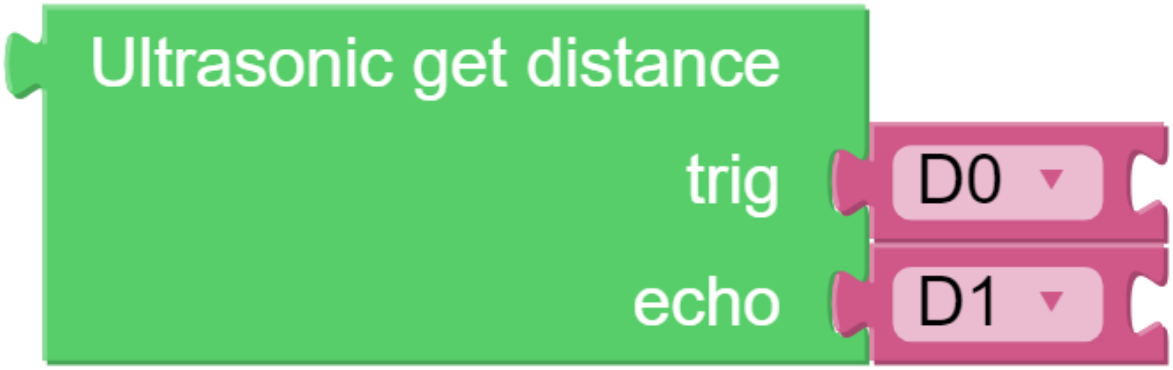
- You can write the program according to the following picture, please refer to the tutorial: How to Create a New Project?.

- Or find the code with the same name on the **Examples** page of the EzBlock Studio and click **Run** or **Edit** directly.



How it works?

You can find the following blocks in the **Module** category to achieve distance detection:



It should be noted that the two pins of the block should correspond to the actual wiring, that is, trig-D2, echo-D3.

Here is the main program.

- Read the `distance` detected by ultrasonic module and filter out the values less than 0 (When the ultrasonic module is too far from the obstacle or cannot read the data correctly, `distance<0` will appear).
- When the `distance` is less than `alert_distance` (the threshold value set earlier, which is 10), play the sound effect `sign.wav`. PiCrawler does `turn left` .
- When the `distance` is greater than `alert_distance`, PiCrawler will move `forward`.

4.7 Computer Vision

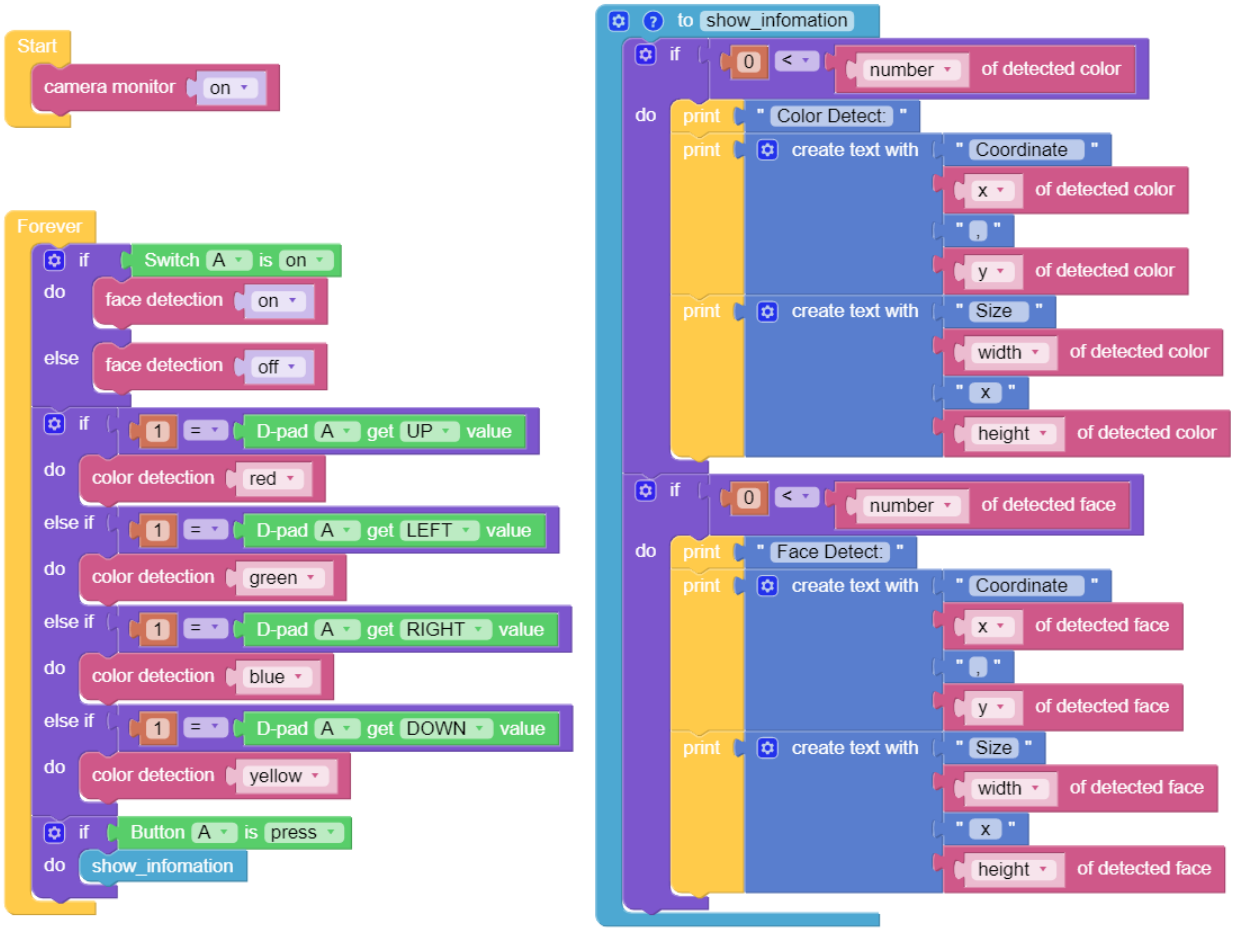
This project will officially enter the field of computer vision!

Note: You can read How to Use the Video Function?. Come and carry out this project smoothly.

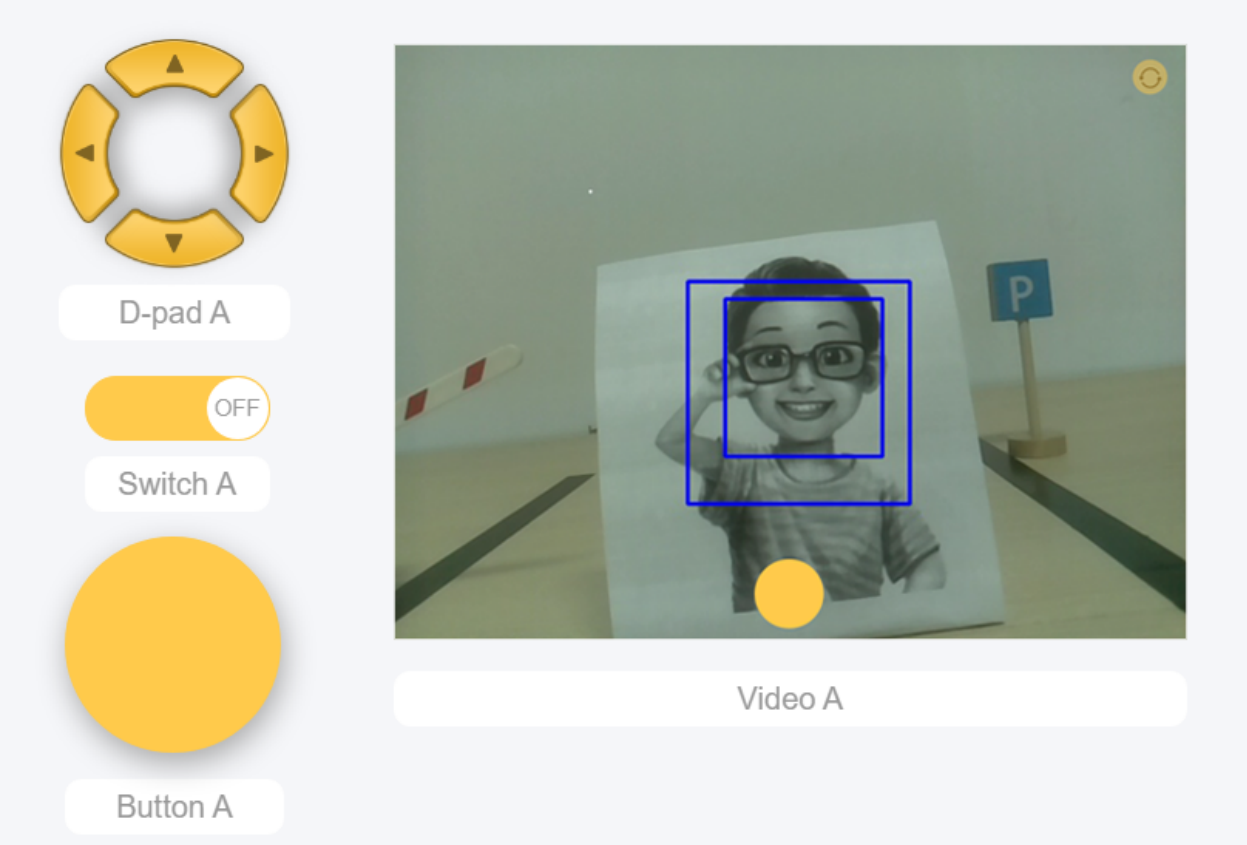
Program

Note:

- You can write the program according to the following picture, please refer to the tutorial: How to Create a New Project?.
- Or find the code with the same name on the **Examples** page of the EzBlock Studio and click **Run** or **Edit** directly.



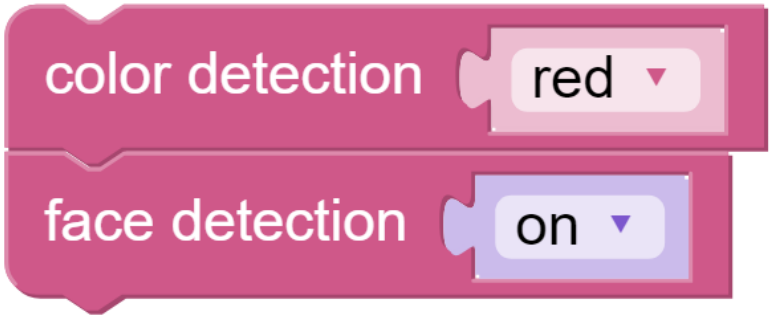
Switch to the Remote Control interface, and you will see the following widgets.



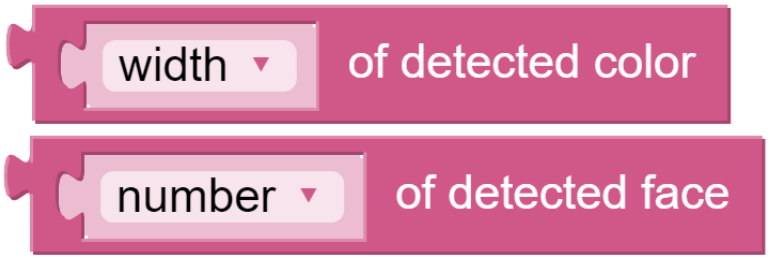
After the program is running, you can switch the slider widget to turn on/off the face detection; click the D-Pad to select the color of the detection; click the button to print the detection result.

How it works?

This block is used to enable the camera module.



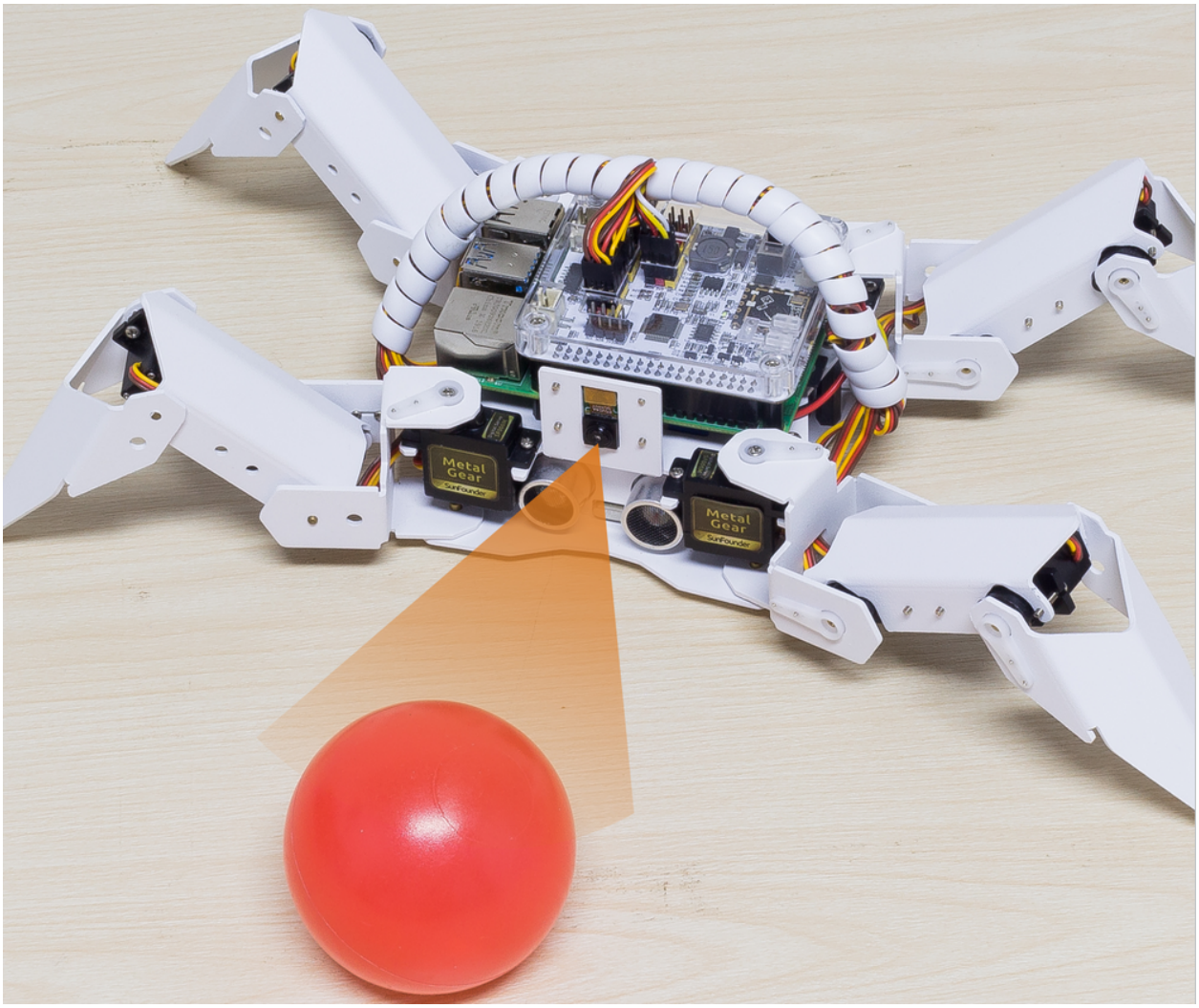
These two blocks are used to enable the face detection/color detection function.



These two blocks are used to output information. The detection result has five output values, namely coordinate x value, coordinate y value, width, height, and number.

4.8 Bull Fight

Make PiCrawler an angry bull! Use its camera to track and rush the red cloth!

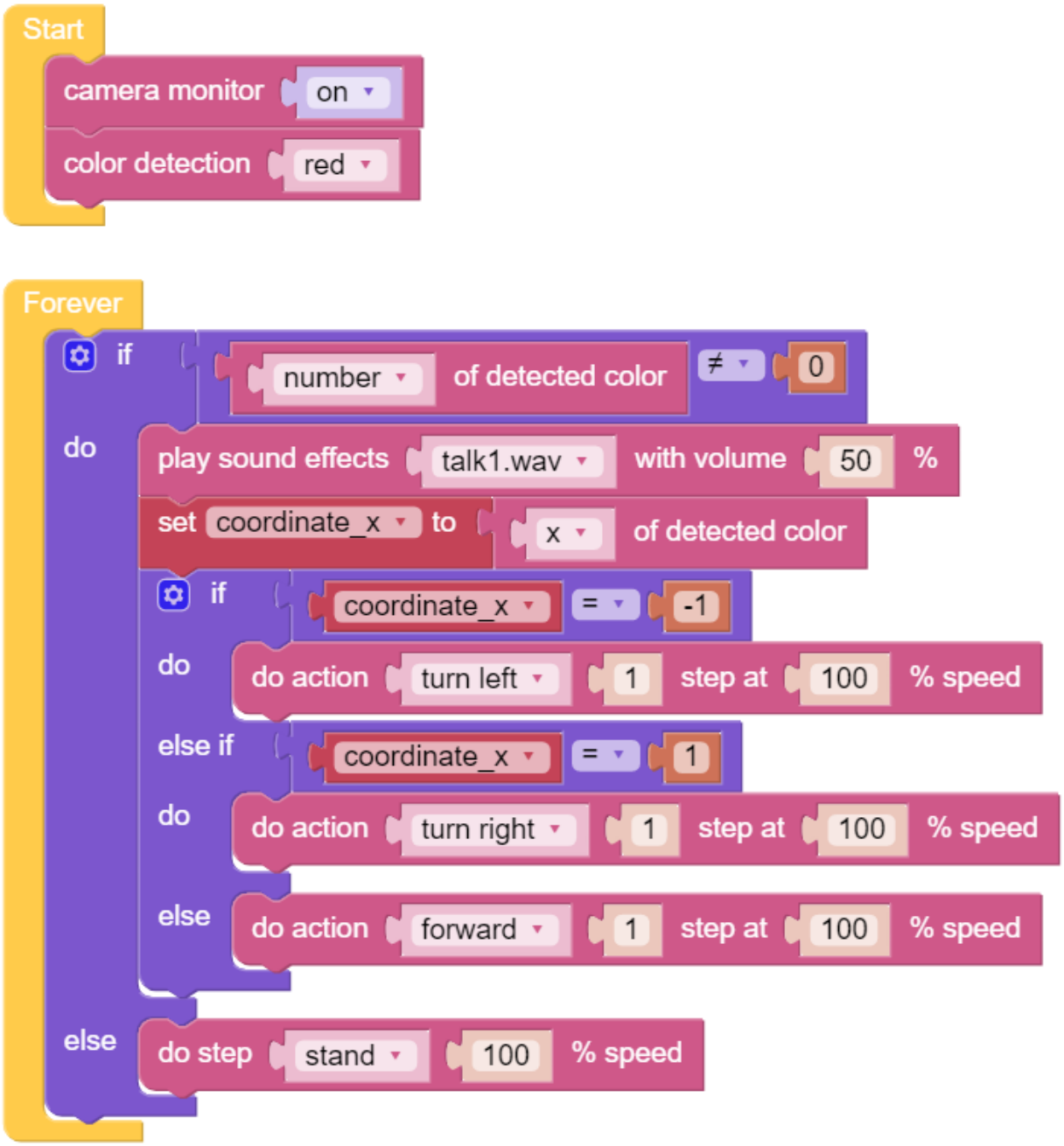


Note: You can download and print the `PDF Color Cards` for color detection.

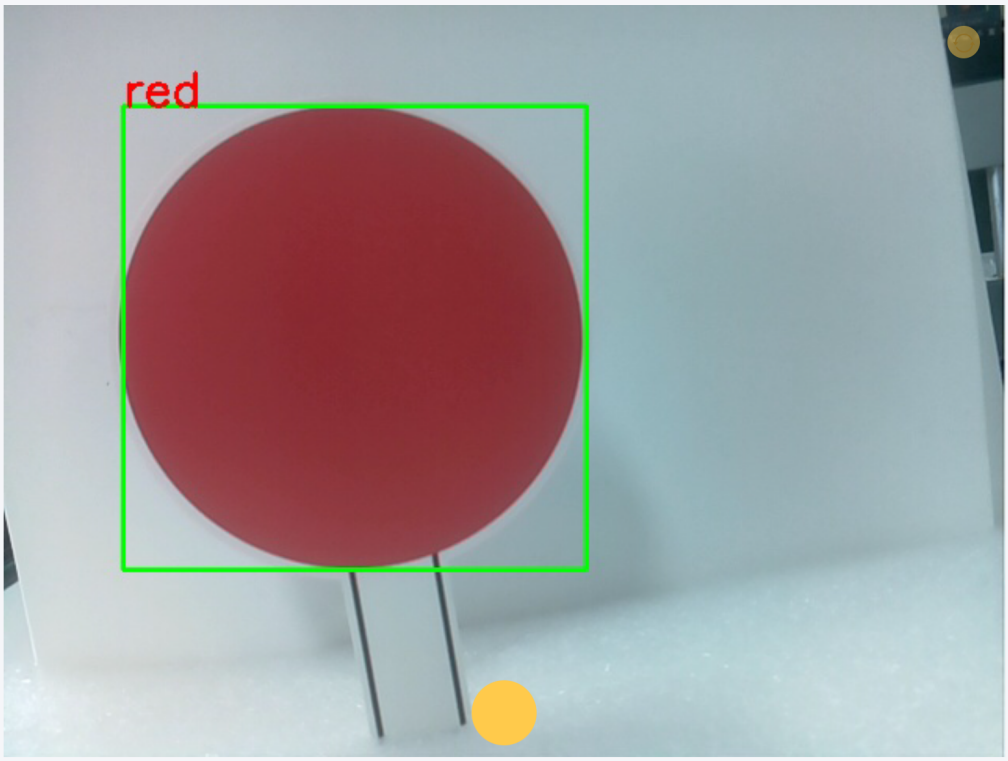
Program

Note:

- You can write the program according to the following picture, please refer to the tutorial: How to Create a New Project?.
- Or find the code with the same name on the **Examples** page of the EzBlock Studio and click **Run** or **Edit** directly.



Switch to the Remote Control interface, you will see the following screen.

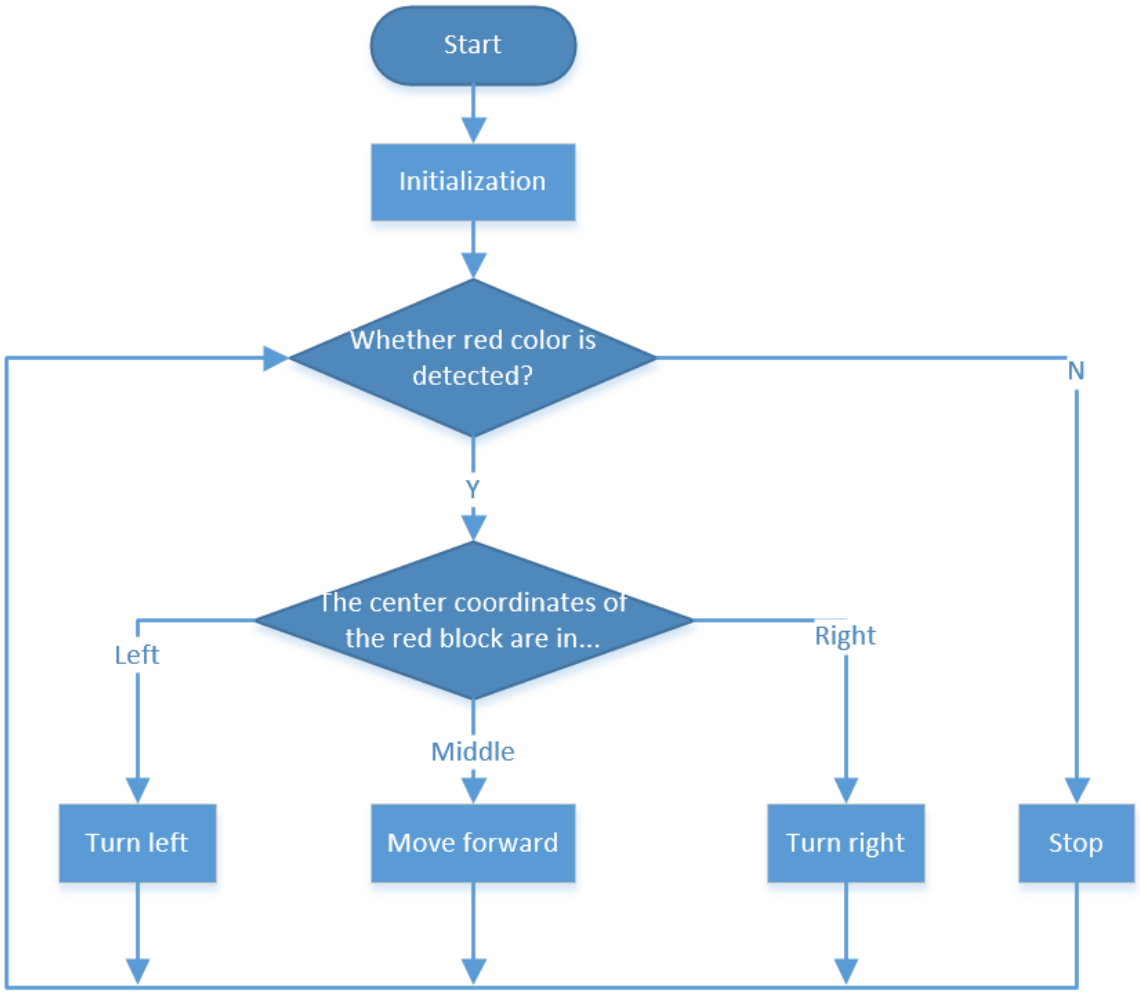


Video A

How it works?

In general, this project combines the knowledge points of *Move*, *Computer Vision* and *Sound Effect*.

Its flow is shown in the figure below:



4.9 Treasure Hunt

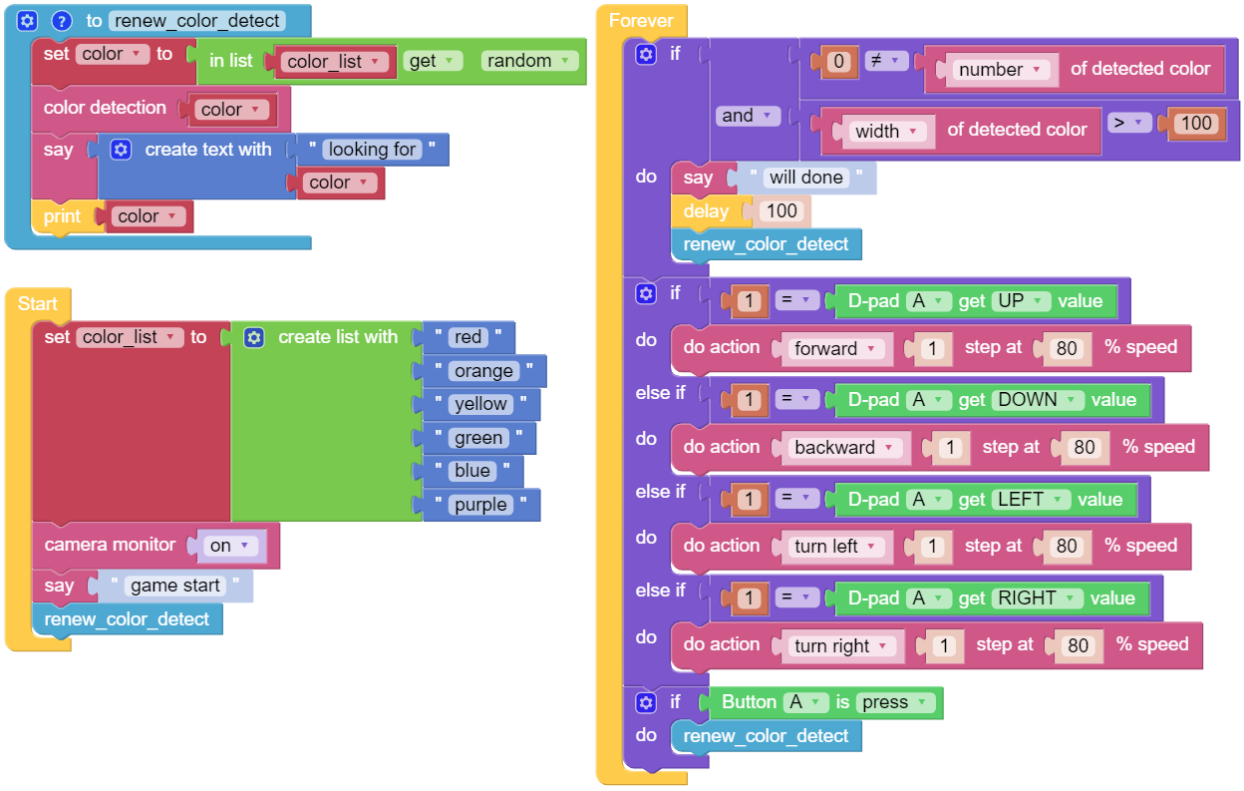
Arrange a maze in your room and place six different color cards in six corners. Then control PiCrawler to search for these color cards one by one!

Note: You can download and print the `PDF Color Cards` for color detection.

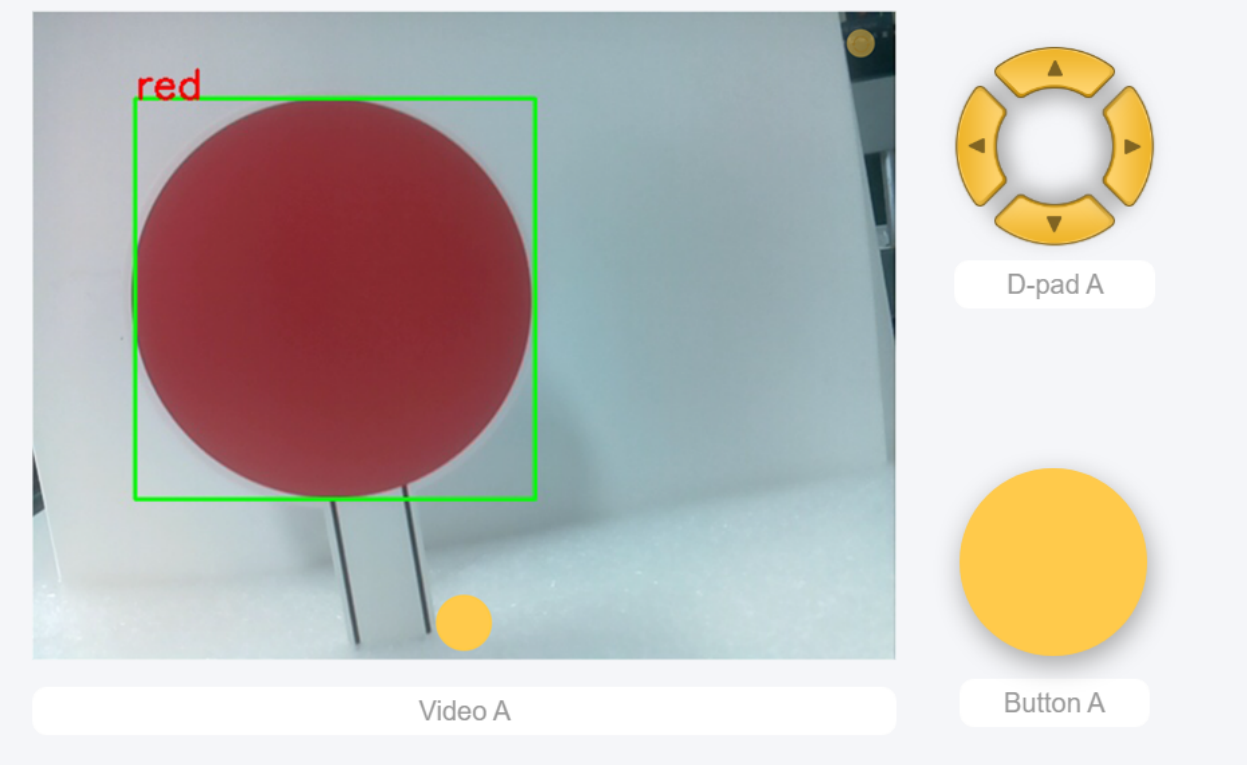
Program

Note:

- You can write the program according to the following picture, please refer to the tutorial: How to Create a New Project?.
- Or find the code with the same name on the **Examples** page of the EzBlock Studio and click **Run** or **Edit** directly.



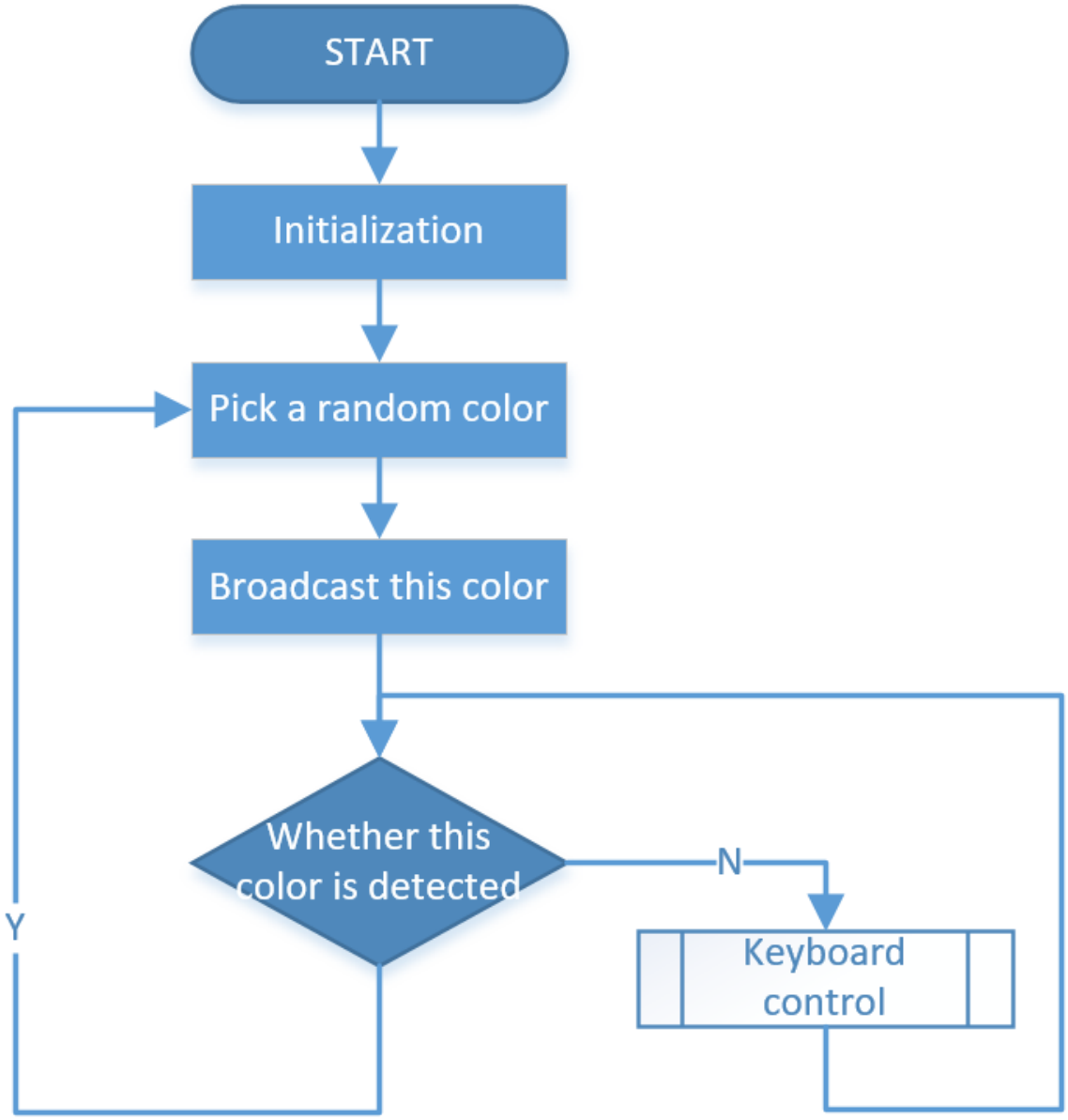
Switch to the Remote Control interface, and you will see the following widgets.



How it works?

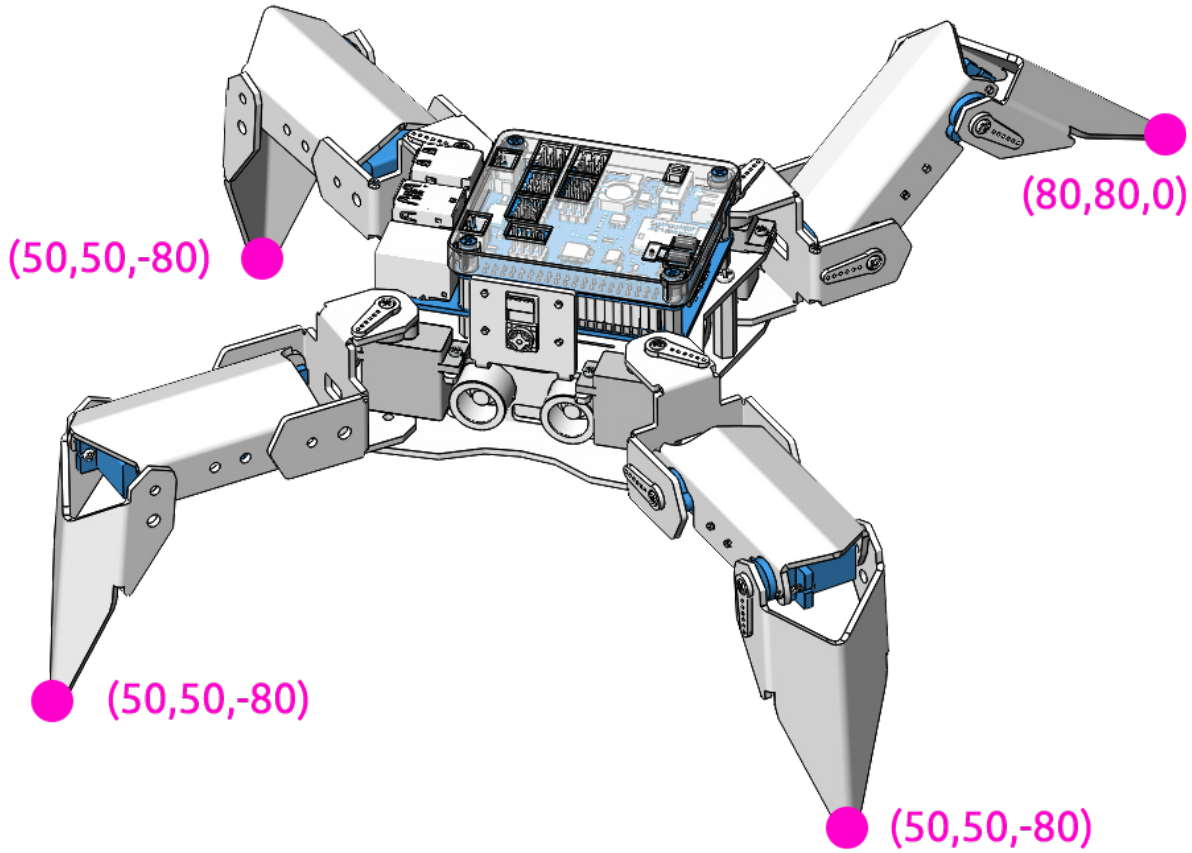
In general, this project combines the knowledge points of *Remote Control*, *Computer Vision* and *Sound Effect*.

Its flow is shown in the figure below:



4.10 Pose

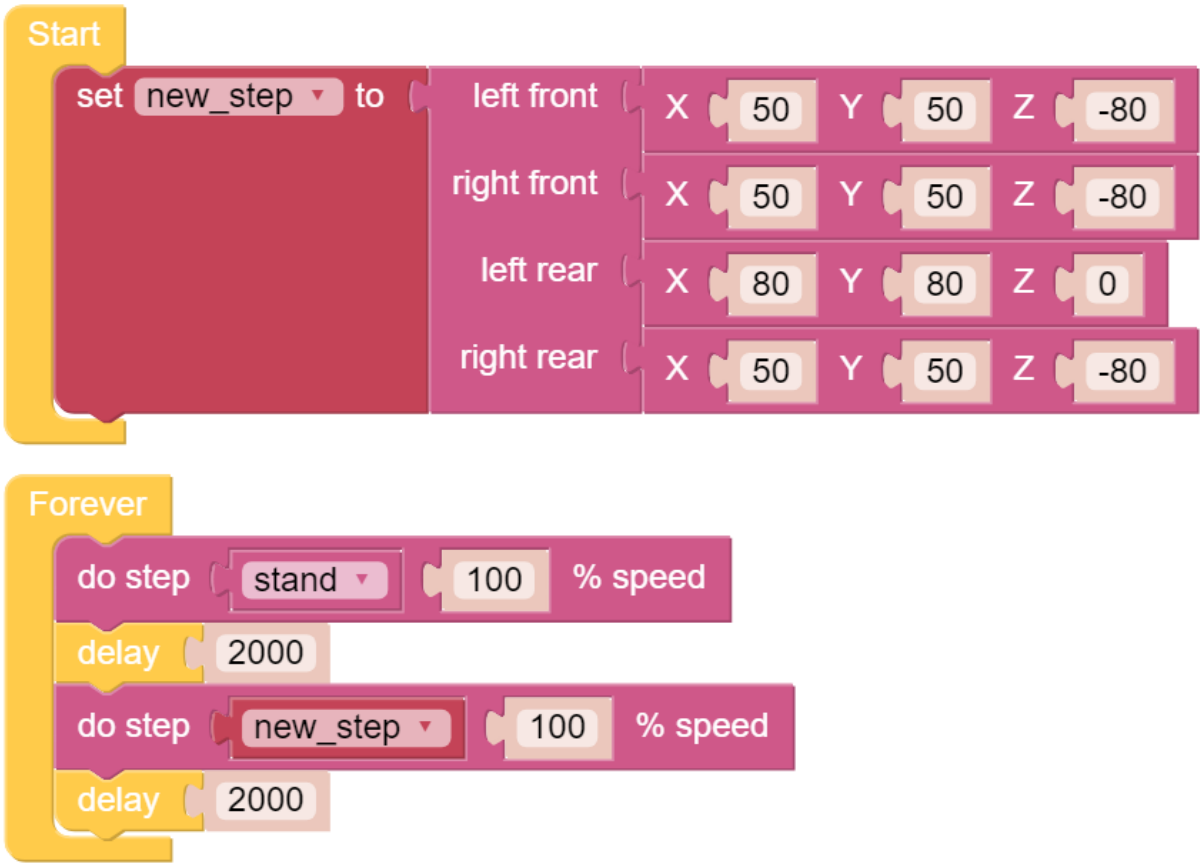
PiCrawler can assume a specific posture by writing a coordinate array. Here it assumes a raised right rear foot posture.



Program

Note:

- You can write the program according to the following picture, please refer to the tutorial: How to Create a New Project?.
- Or find the code with the same name on the **Examples** page of the EzBlock Studio and click **Run** or **Edit** directly.



How it works?

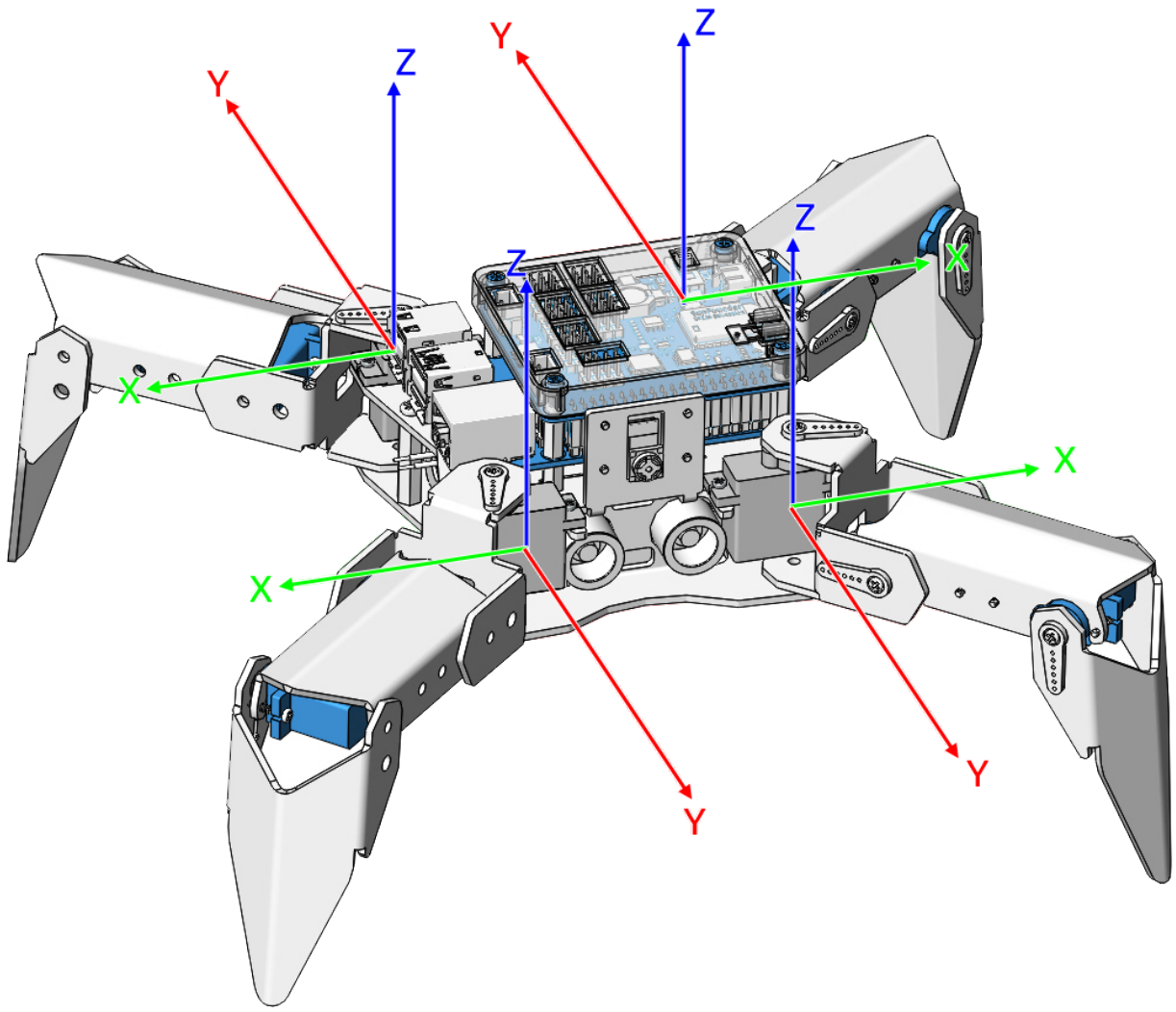
In this code, the code you need to pay attention to is this **do step**.

It has two uses:

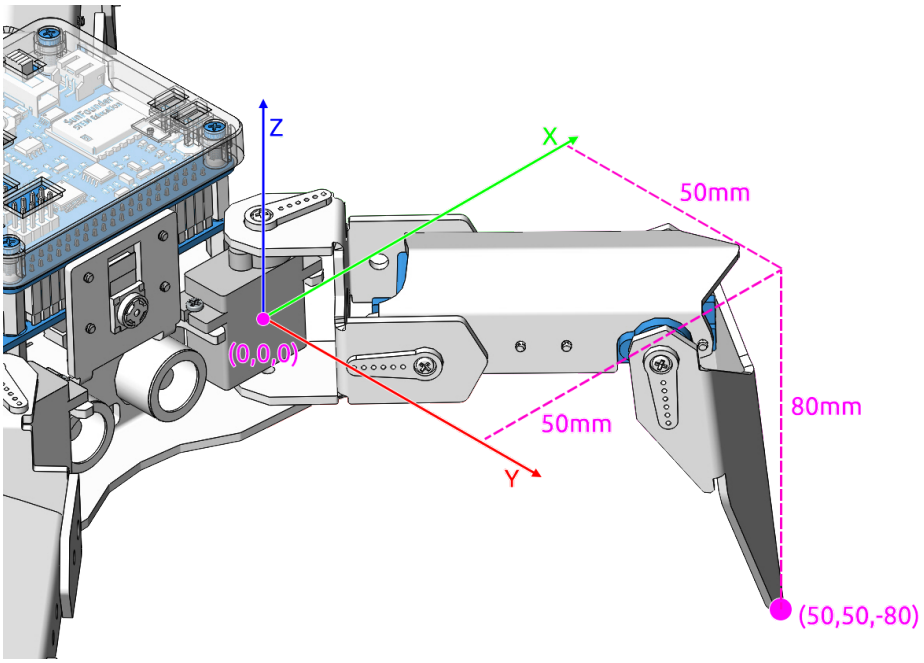
One: It can directly use **stand** or **sit**.

Second: It can also write an array of 4 coordinate values.

Each foot has an independent coordinate system. As shown below:



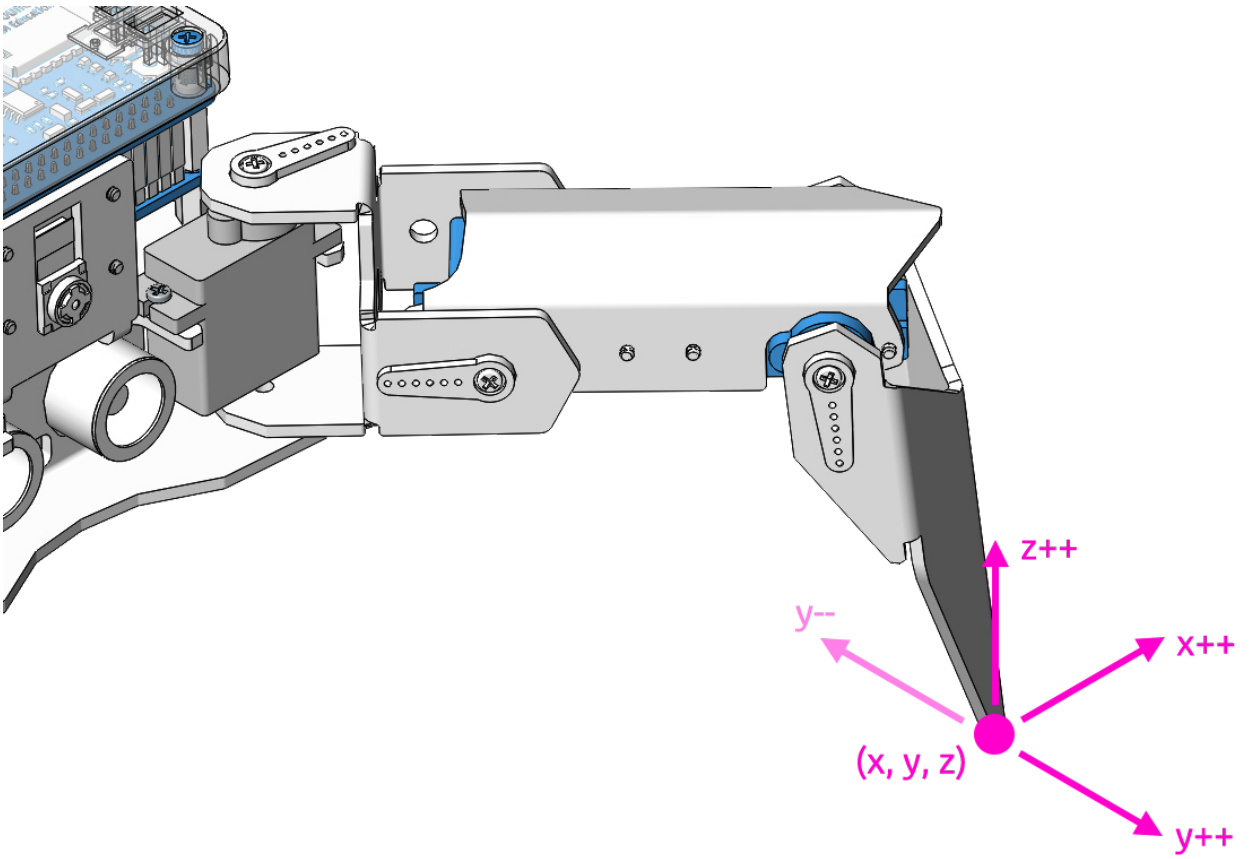
You need to measure the coordinates of each toe individually. As shown below:



4.11 Adjust Posture

In this example, we use the remote function to control the PiCrawler foot by foot and assume the desired posture.

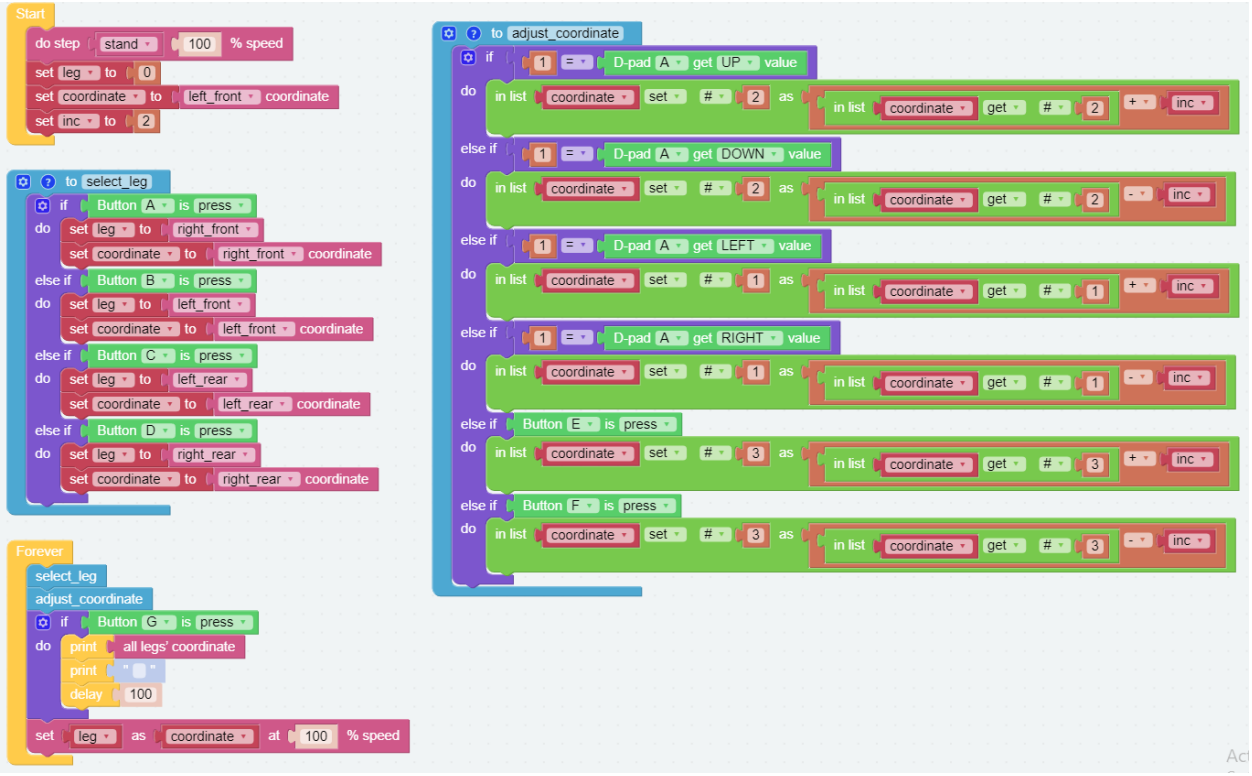
You can tap the button to print out the current coordinate values. These coordinate values come in handy when you create unique actions for PiCrawler.



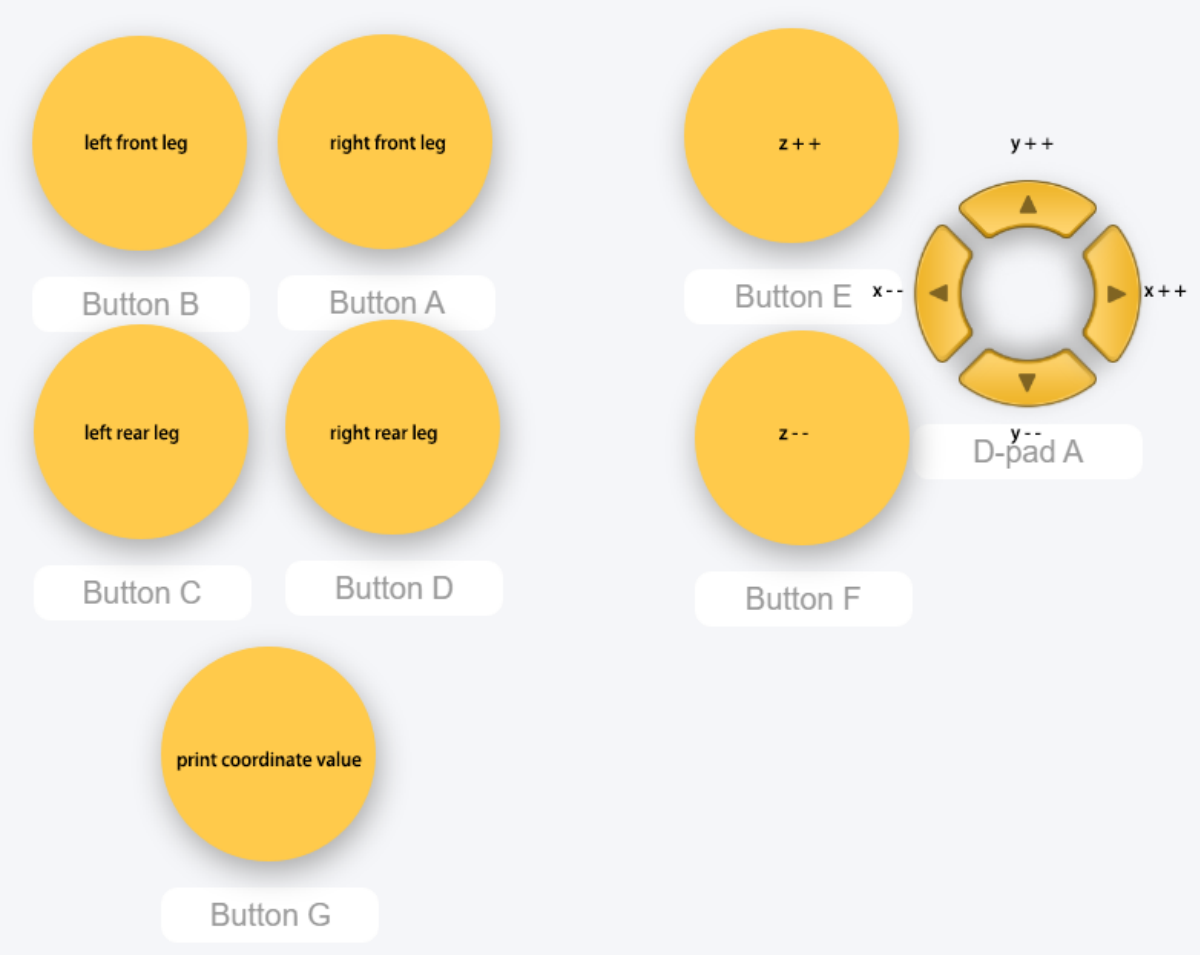
Program

Note:

- You can write the program according to the following picture, please refer to the tutorial: How to Create a New Project?.
- Or find the code with the same name on the **Examples** page of the EzBlock Studio and click **Run** or **Edit** directly.



Switch to the Remote Control interface, and you will see the following widgets.



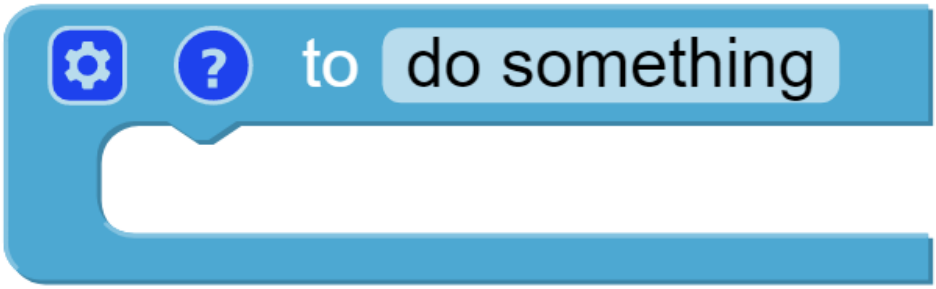
How it works?

What you need to pay attention to in this project are the following three blocks:

Modify the coordinate value of a certain leg individually.

Returns the coordinate value of the corresponding leg.

You may want to simplify the program with Functions, especially when you perform the same operation multiple times. Putting these operations into a newly declared function can greatly facilitate your use.



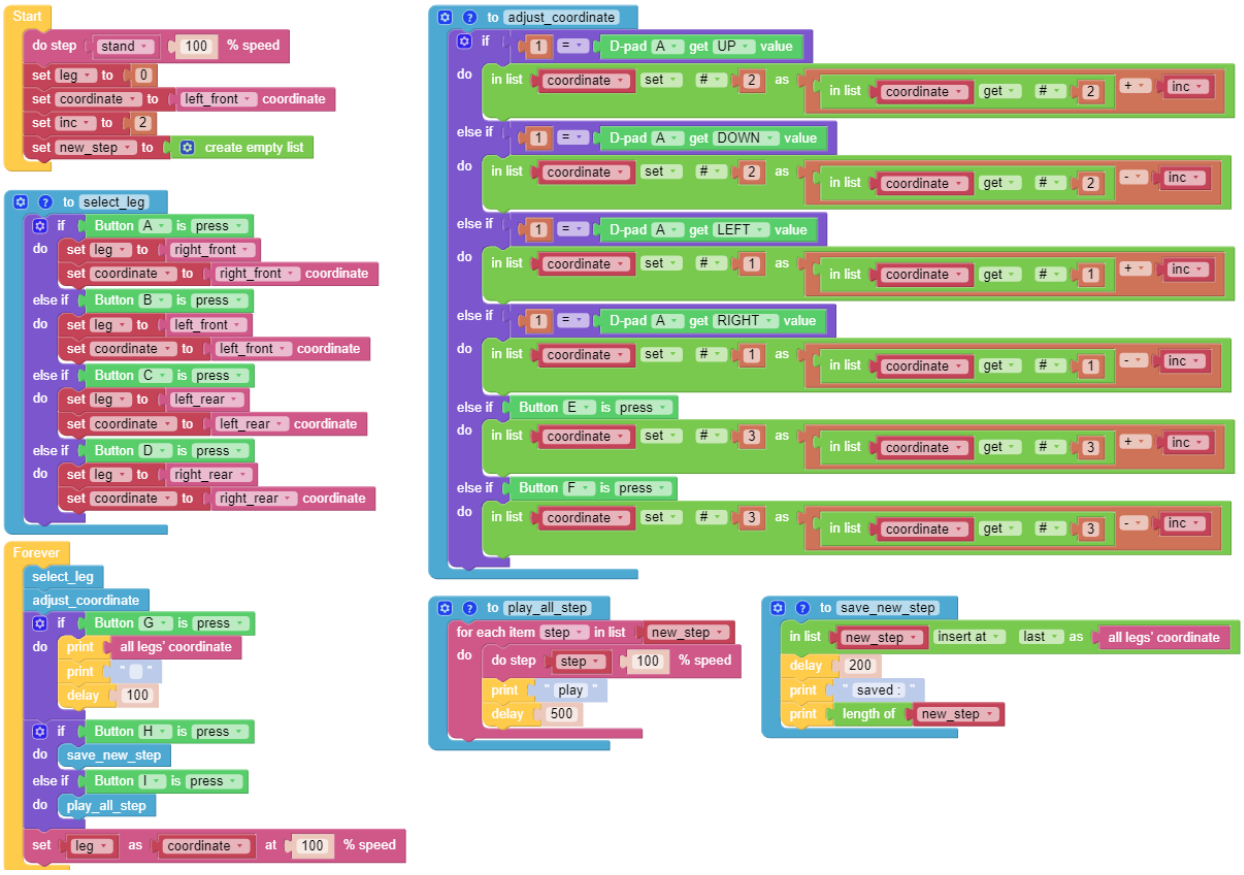
4.12 Record New Step

We use the remote function to control PiCrawler to make several poses in turn, and record these poses. Replay them later.

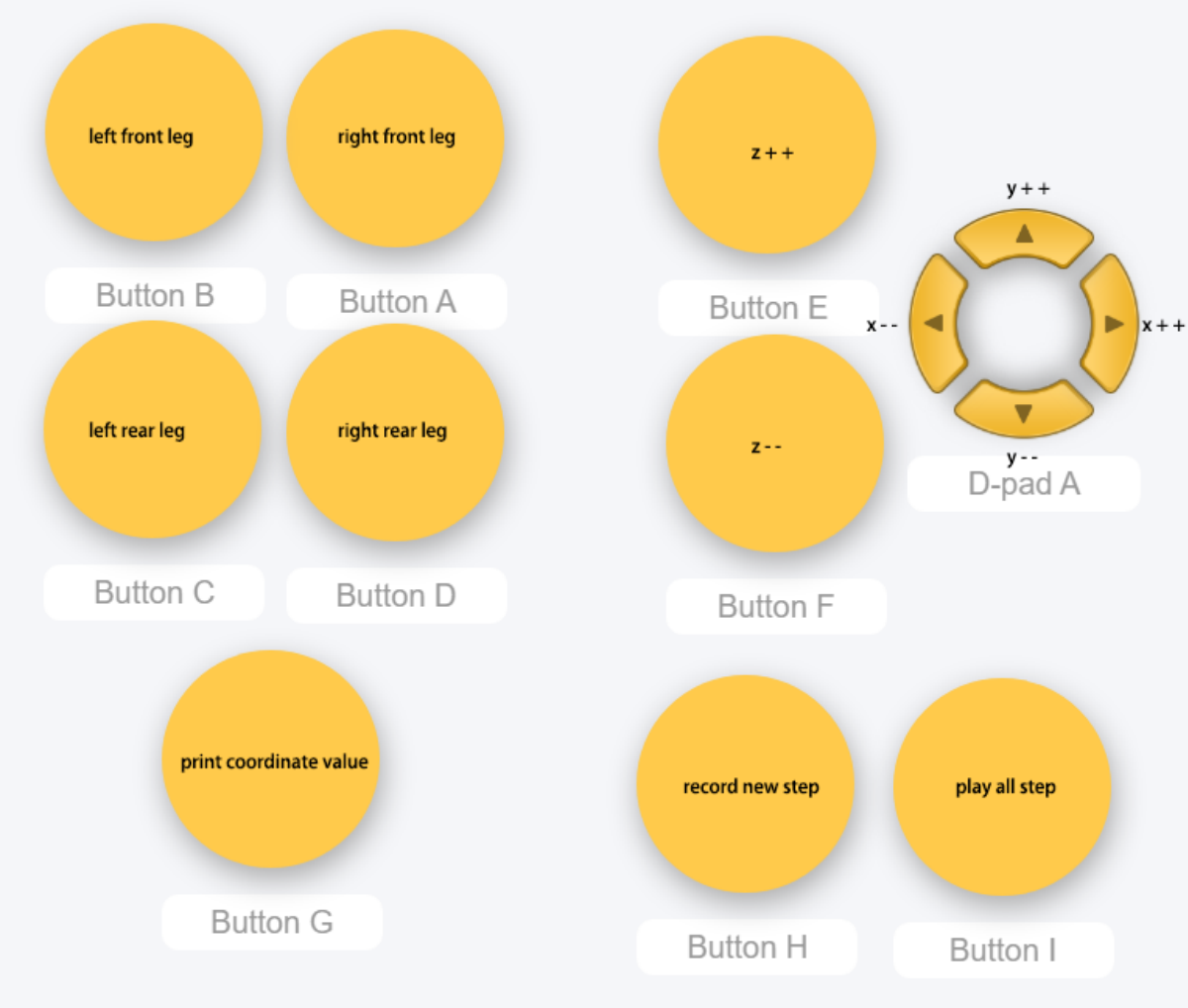
Program

Note:

- You can write the program according to the following picture, please refer to the tutorial: How to Create a New Project?.
- Or find the code with the same name on the **Examples** page of the EzBlock Studio and click **Run** or **Edit** directly.



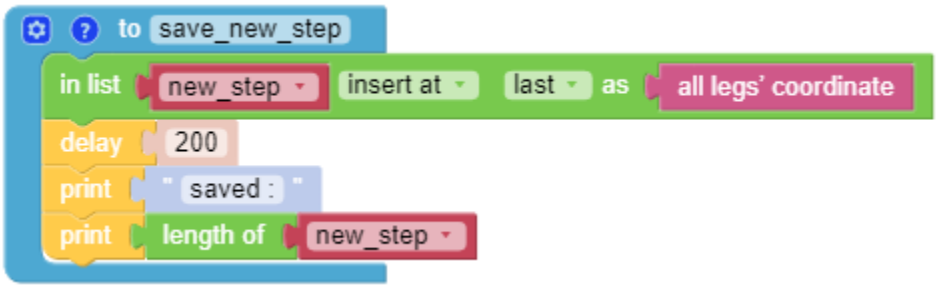
Switch to the Remote Control interface, and you will see the following widgets.



How it works?

This project was born out of *Adjust Posture*. Added recording and replay functions.

The recording function is implemented by the following code.



The replay function is implemented by the following code.

4.13 Twist

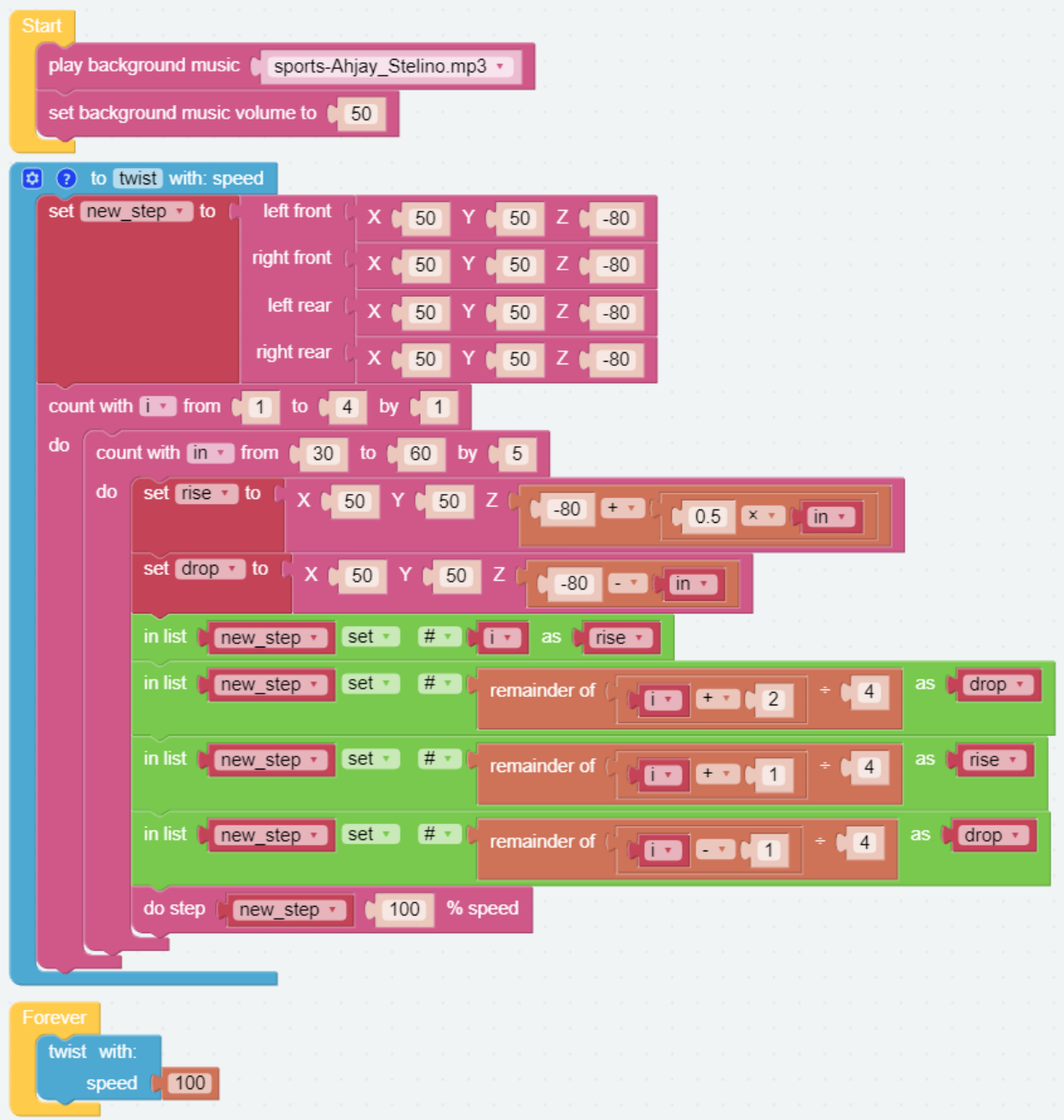
We already know how to make PiCrawler assume a specific pose, the next step is to combine the poses to form a continuous action.

Here, PiCrawler's four feet are up and down in twos, jumping with the music.

Program

Note:

- You can write the program according to the following picture, please refer to the tutorial: How to Create a New Project?.
- Or find the code with the same name on the **Examples** page of the EzBlock Studio and click **Run** or **Edit** directly.

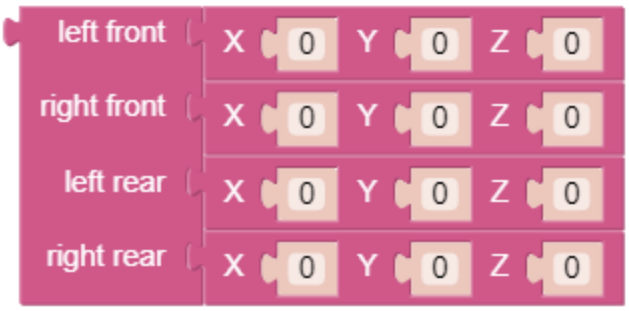


How it works?

It uses two layers of for loops to make the `new_step` array produce continuous and regular changes, and at the same time, **do step** executes the posture to form a continuous action.

You can intuitively get the coordinate value array corresponding to each pose from *Adjust Posture*.

One thing you need to pay attention to is this coordinate matrix block:



It is essentially a two-dimensional array, which can be processed by blocks in the **List** category. Its structure is `[[right front],[left front],[left rear],[right rear]]`. In other words, in this example, `new_step#1` corresponds to the right front; `new_step#2` corresponds to the left front; `new_step#3` corresponds to the left rear; and `new_step#4` corresponds to right rear.

4.14 Emotional Robot

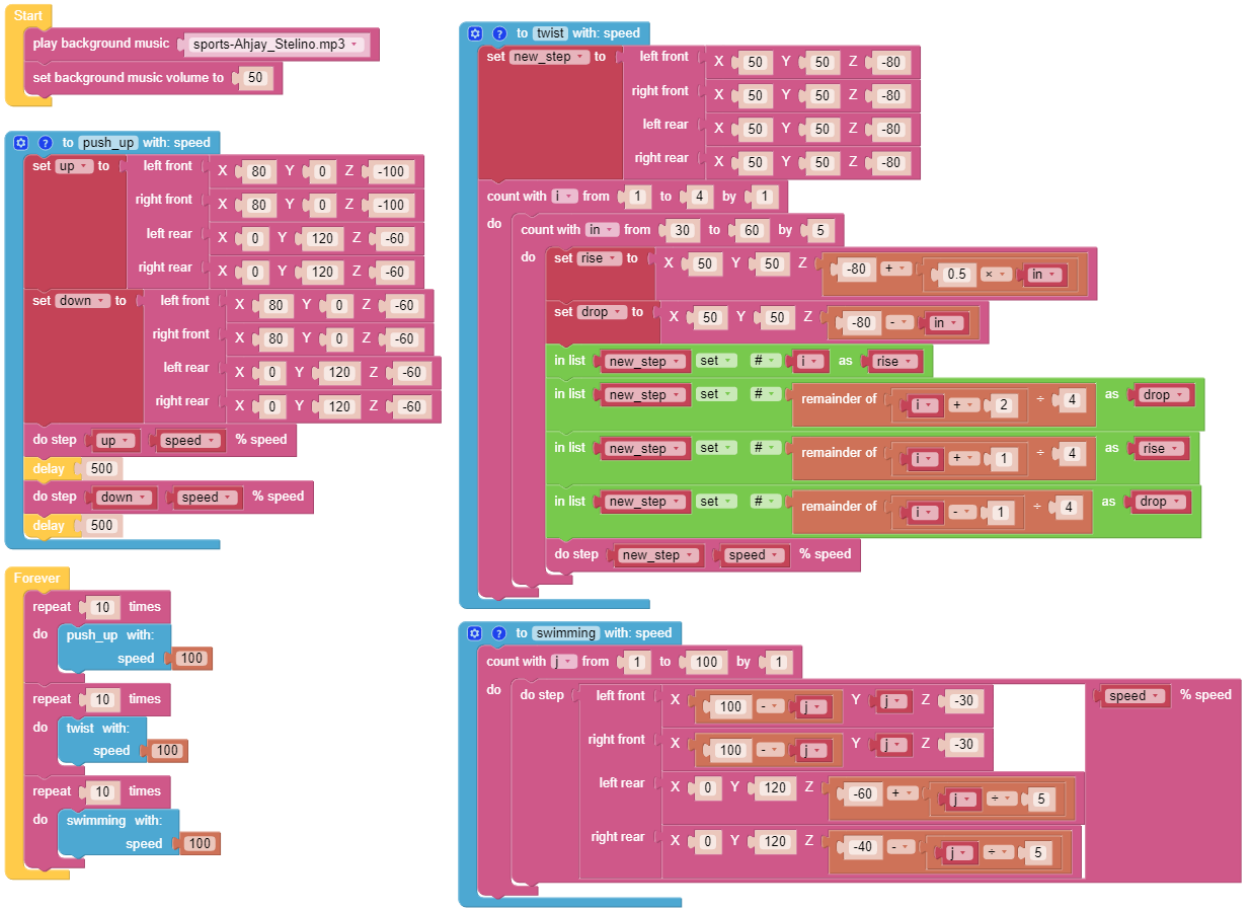
This example shows several interesting custom actions of PiCrawler.

It is a supplementary example of *Twist*.

Program

Note:

- You can write the program according to the following picture, please refer to the tutorial: How to Create a New Project?.
- Or find the code with the same name on the **Examples** page of the EzBlock Studio and click **Run** or **Edit** directly.


Start
play background music sports-Ahjay_Stelino.mp3
set background music volume to 50
to push_up with: speed
set up to left front X 80 Y 0 Z -100
right front X 80 Y 0 Z -100
left rear X 0 Y 120 Z -60
right rear X 0 Y 120 Z -60
set down to left front X 80 Y 0 Z -60
right front X 80 Y 0 Z -60
left rear X 0 Y 120 Z -60
right rear X 0 Y 120 Z -60
do step up speed % speed
delay 500
do step down speed % speed
delay 500
Forever
repeat 10 times
do push_up with: speed 100
repeat 10 times
do twist with: speed 100
repeat 10 times
do swimming with: speed 100
to twist with: speed
set new_step to left front X 50 Y 50 Z -80
right front X 50 Y 50 Z -80
left rear X 50 Y 50 Z -80
right rear X 50 Y 50 Z -80
count with i from 1 to 4 by 1
do count with in from 30 to 60 by 5
do set rise to X 50 Y 50 Z -80 + 0.5 × in
set drop to X 50 Y 50 Z -80 - in
in list new_step set # i as rise
in list new_step set # remainder of i + 2 ÷ 4 as drop
in list new_step set # remainder of i + 1 ÷ 4 as rise
in list new_step set # remainder of i - 1 ÷ 4 as drop
do step new_step speed % speed
to swimming with: speed
count with j from 1 to 100 by 1
do do step left front X 100 - j Y j Z -30
right front X 100 - j Y j Z -30
left rear X 0 Y 120 Z -60 + j ÷ 5
right rear X 0 Y 120 Z -40 - j ÷ 5
speed % speed

CHAPTER

FIVE

APPENDIX

5.1 Filezilla Software

The File Transfer Protocol (FTP) is a standard communication protocol used for the transfer of computer files from a server to a client on a computer network.

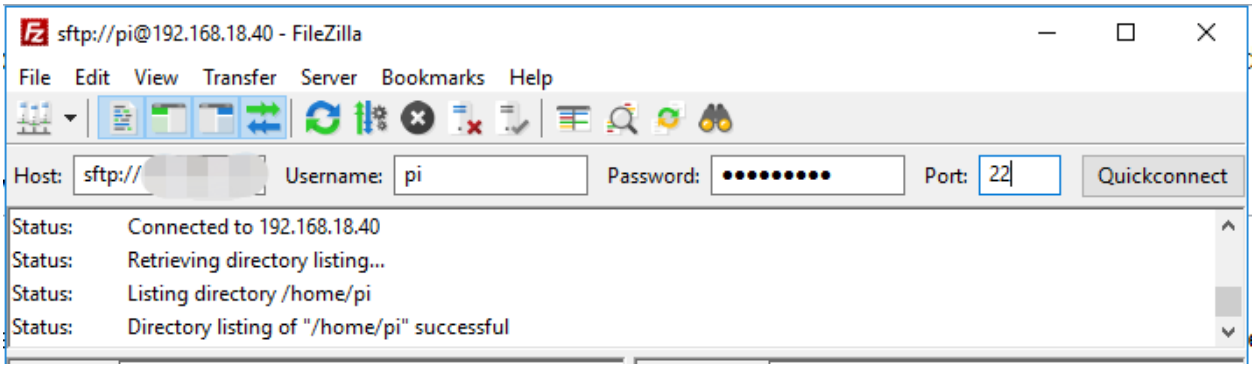
Filezilla is an open source software that not only supports FTP, but also FTP over TLS (FTPS) and SFTP. We can use Filezilla to upload local files (such as pictures and audio, etc.) to the Raspberry Pi, or download files from the Raspberry Pi to the local.

Step 1: Download Filezilla.

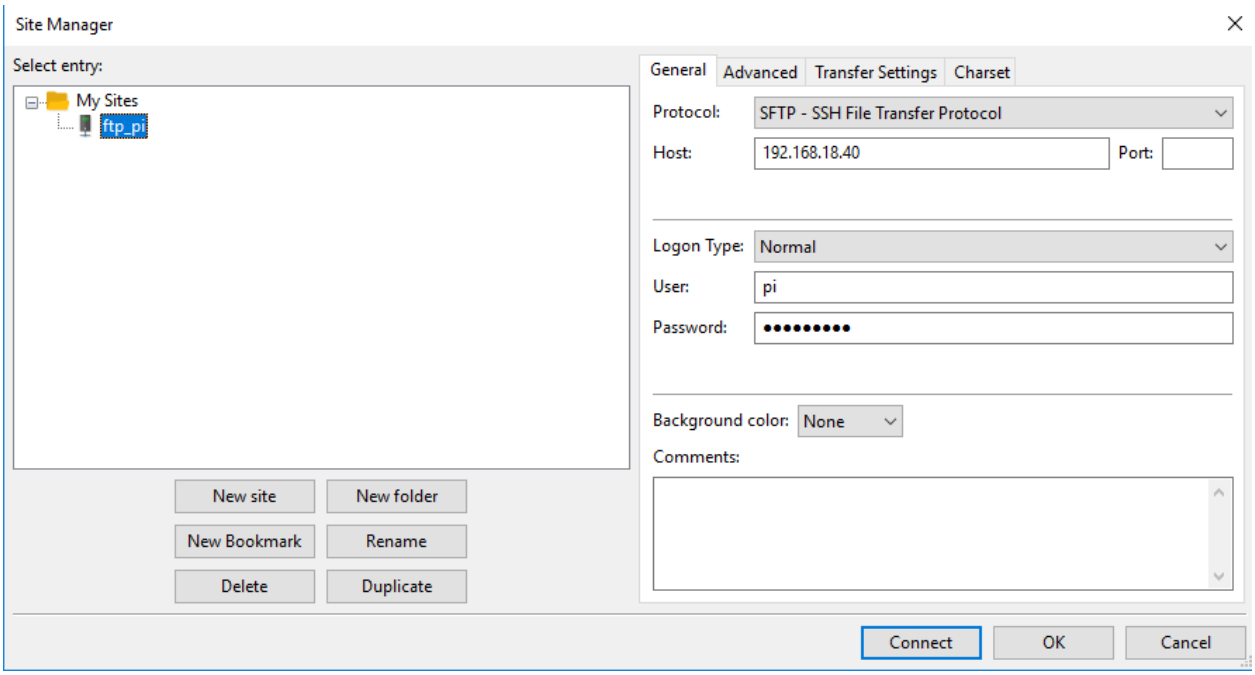
Download the client from Filezilla's official website, Filezilla has a very good tutorial, please refer to: Documentation - Filezilla.

Step 2: Connect to Raspberry Pi

After a quick install open it up and now connect it to an FTP server. It has 3 ways to connect, here we use the **Quick Connect** bar. Enter the **hostname/IP**, **username**, **password** and **port (22)**, then click **Quick Connect** or press **Enter** to connect to the server.

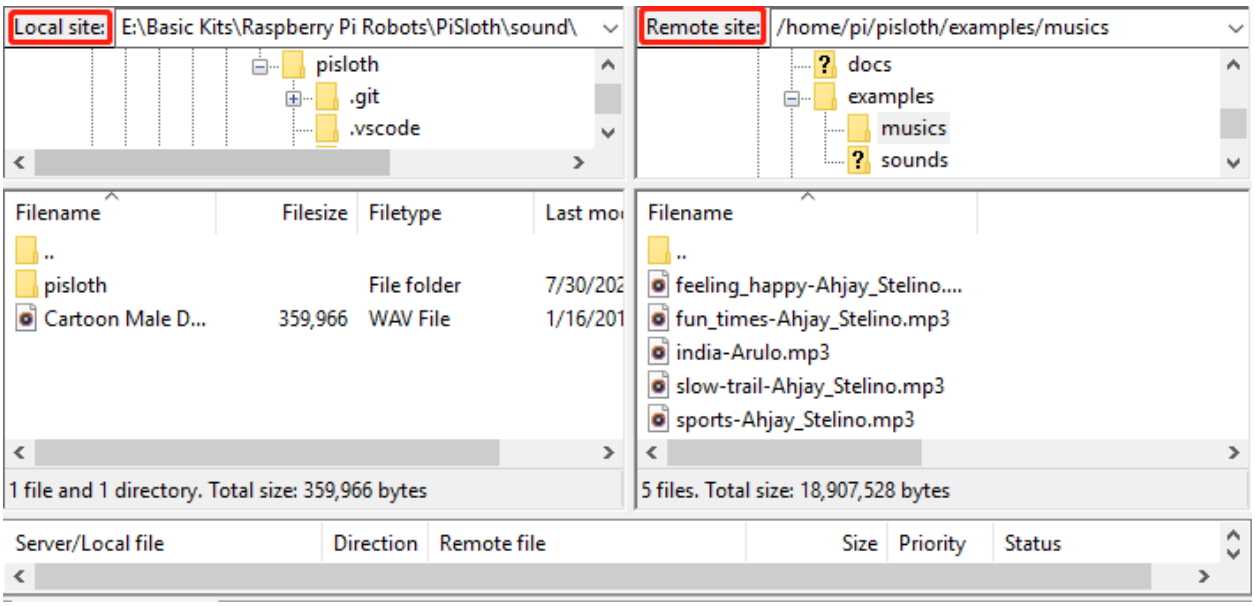


Note: Quick Connect is a good way to test your login information. If you want to create a permanent entry, you can select **File-> Copy Current Connection to Site Manager** after a successful Quick Connect, enter the name and click **OK**. Next time you will be able to connect by selecting the previously saved site inside **File -> Site Manager**.



Step 3: Upload/download files.

You can upload local files to Raspberry Pi by dragging and dropping them, or download the files inside Raspberry Pi files locally.

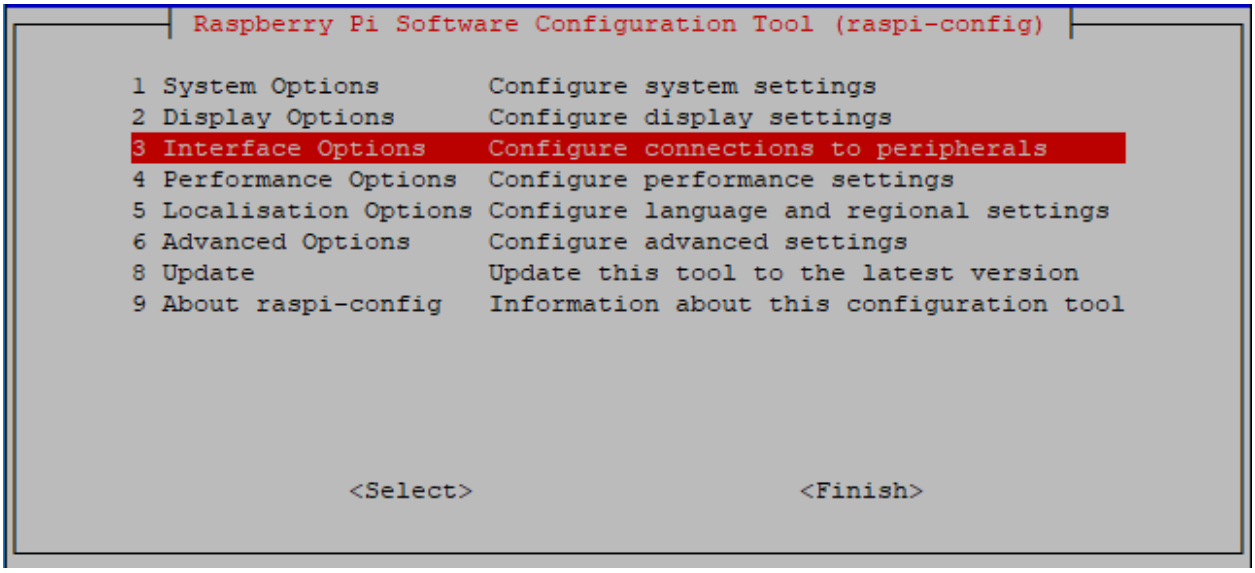


5.2 I2C Configuration

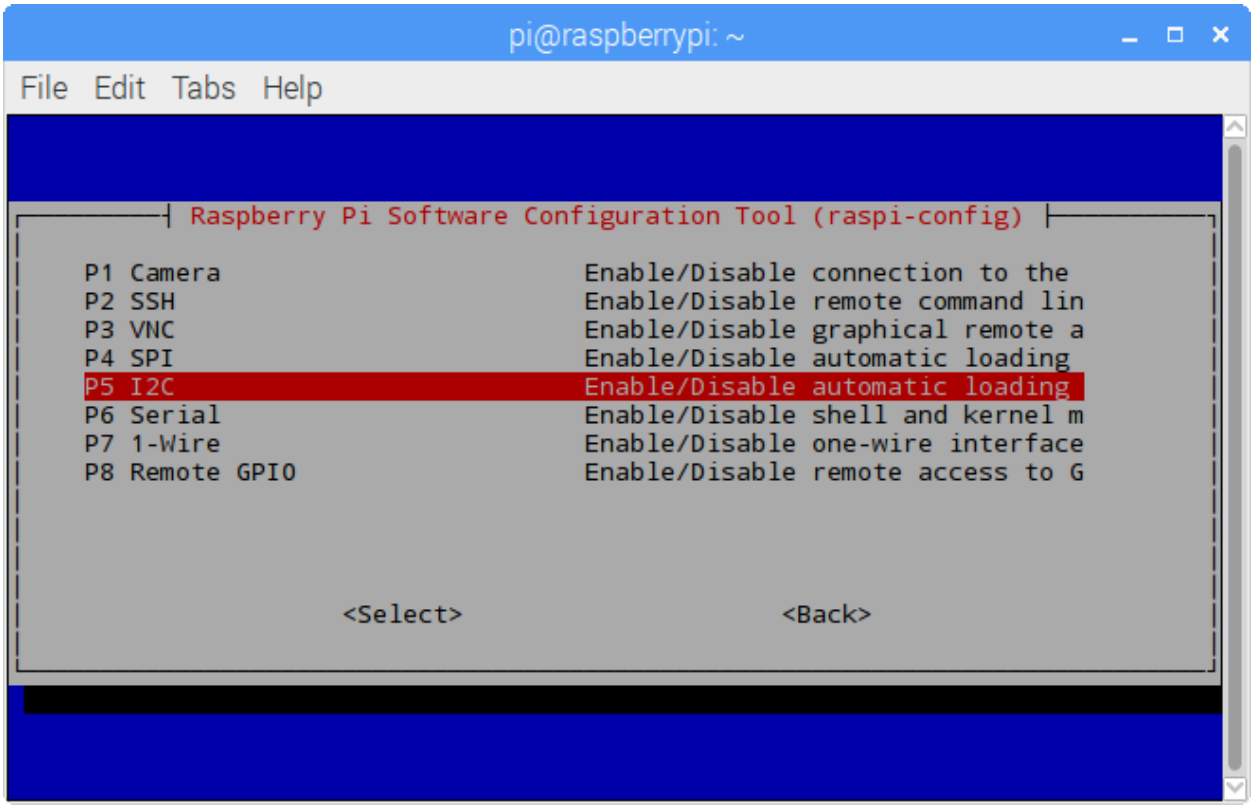
Enable the I2C port of your Raspberry Pi (If you have enabled it, skip this; if you do not know whether you have done that or not, please continue).

```
sudo raspi-config
```

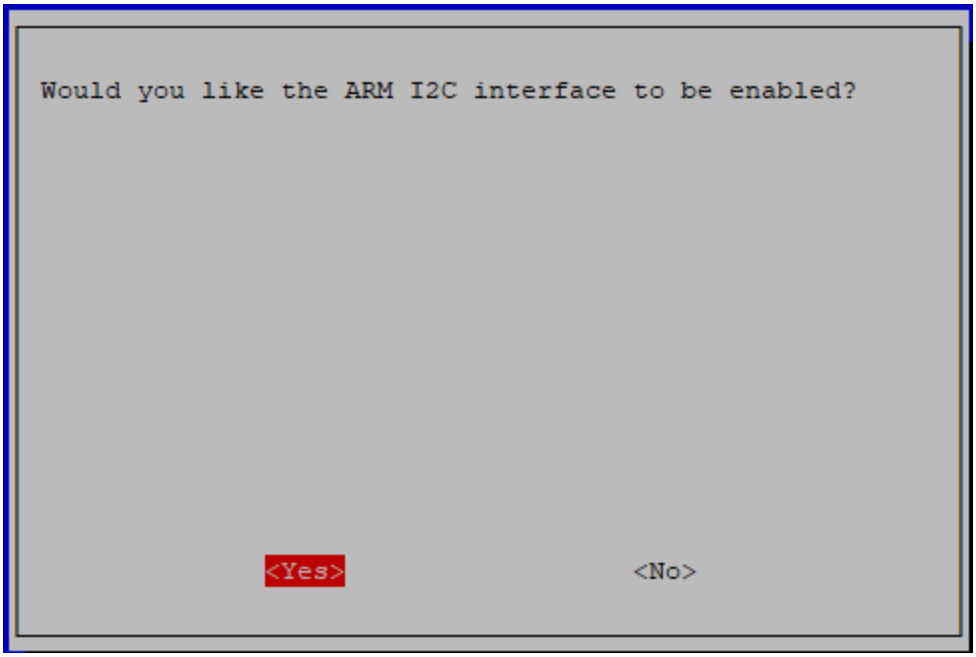
3 Interfacing options



P5 I2C



<Yes>, then <Ok> -> <Finish>.



5.3 Remote Desktop

There are two ways to control the desktop of the Raspberry Pi remotely:

VNC and **XRDP**, you can use any of them.

5.3.1 VNC

You can use the function of remote desktop through VNC.

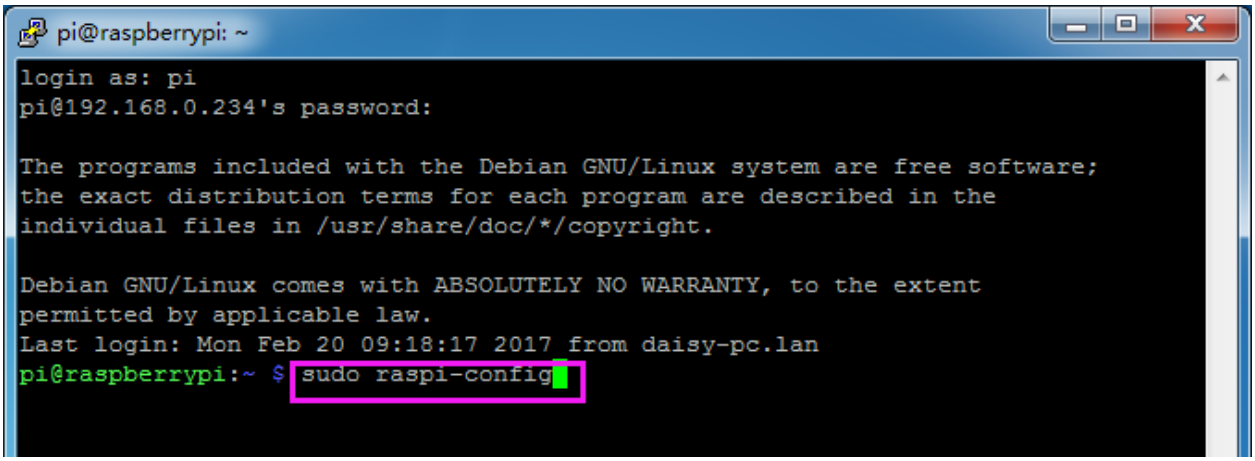
Enable VNC service

The VNC service has been installed in the system. By default, VNC is disabled. You need to enable it in config.

Step 1

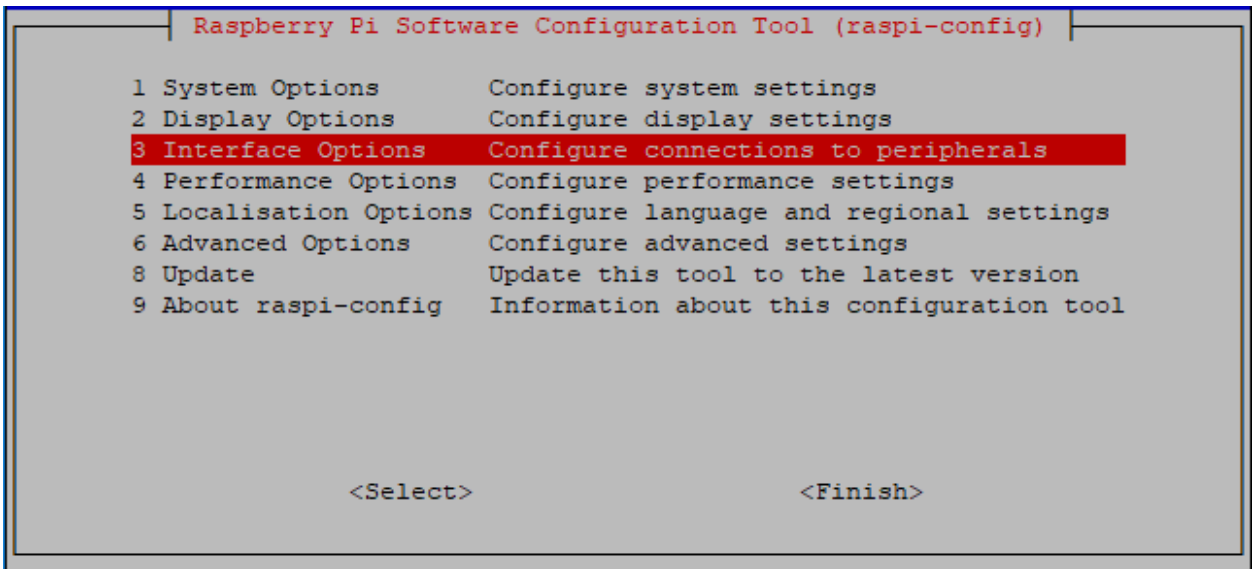
Input the following command:

```
sudo raspi-config
```



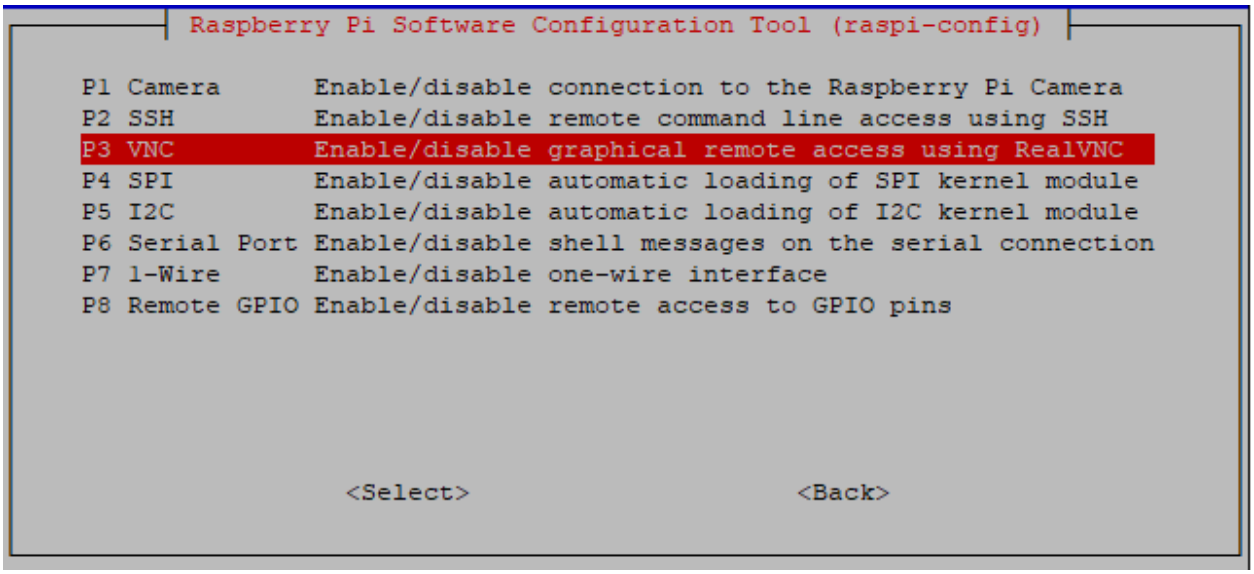
Step 2

Choose **3 Interfacing Options** by press the down arrow key on your keyboard, then press the **Enter** key.



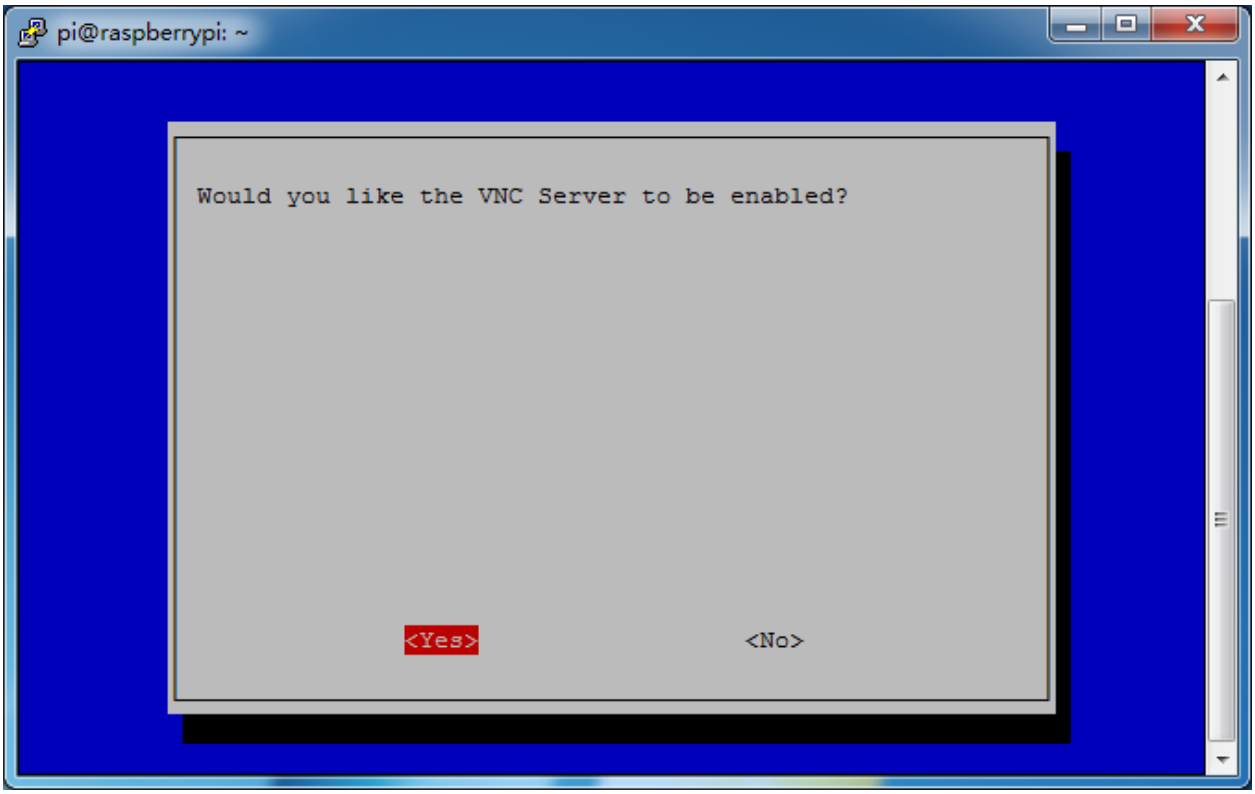
Step 3

P3 VNC



Step 4

Select **Yes -> OK -> Finish** to exit the configuration.



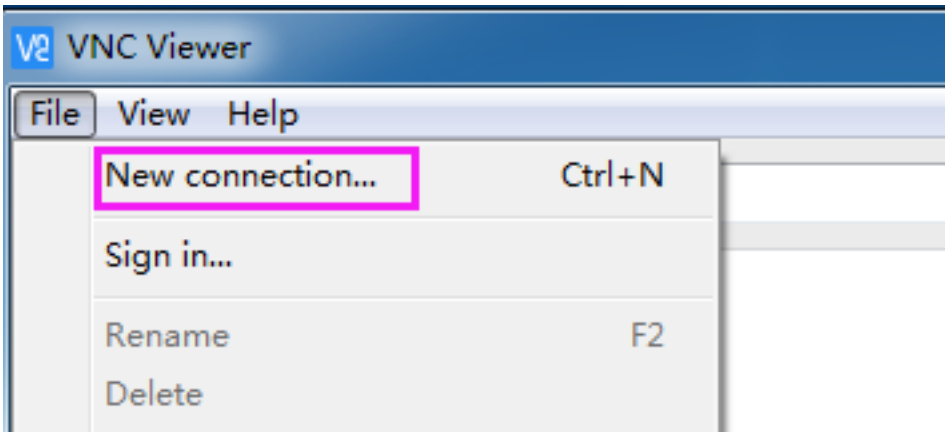
Login to VNC

Step 1

You need to download and install the VNC Viewer on personal computer. After the installation is done, open it.

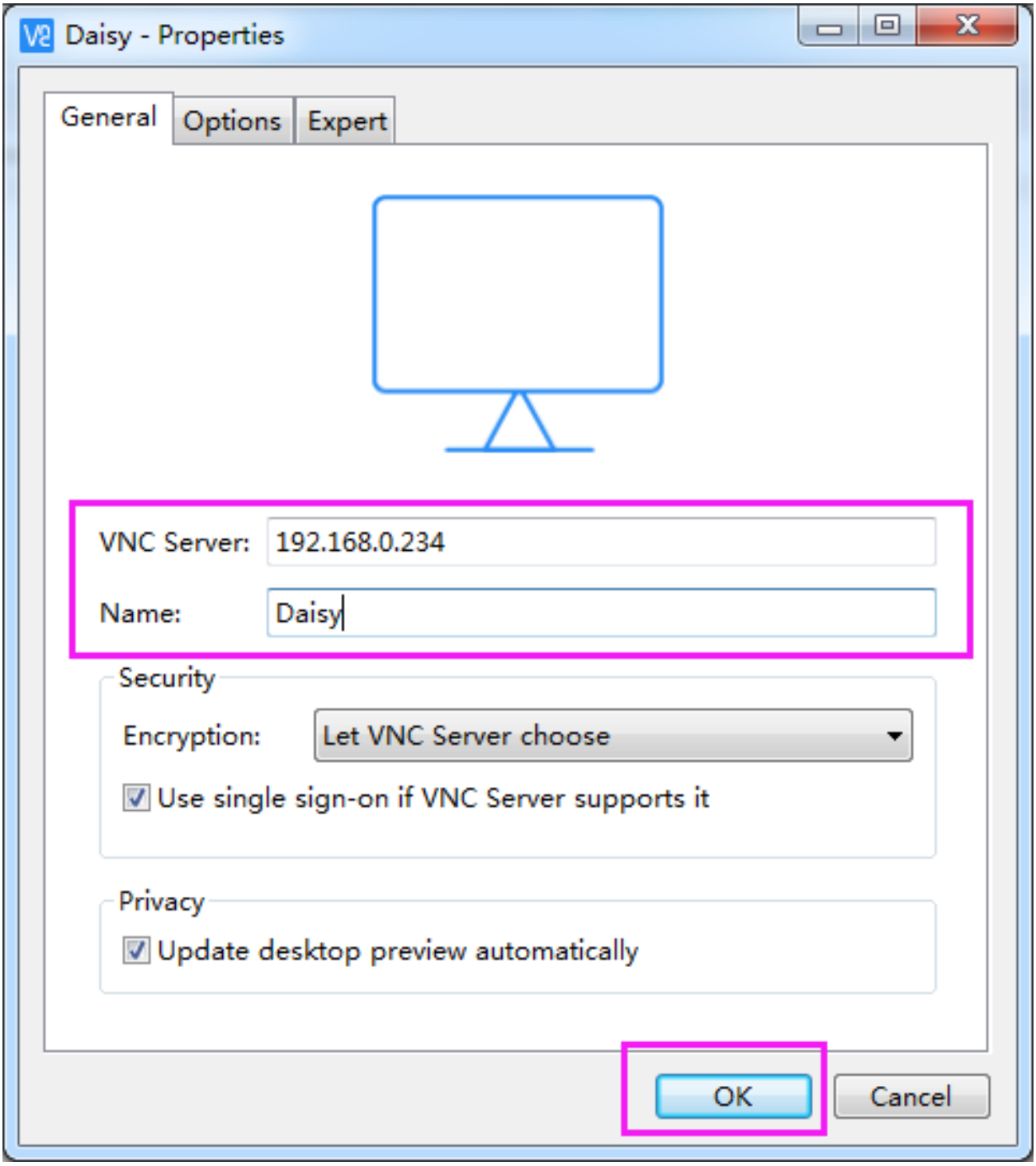
Step 2

Then select "**New connection**".



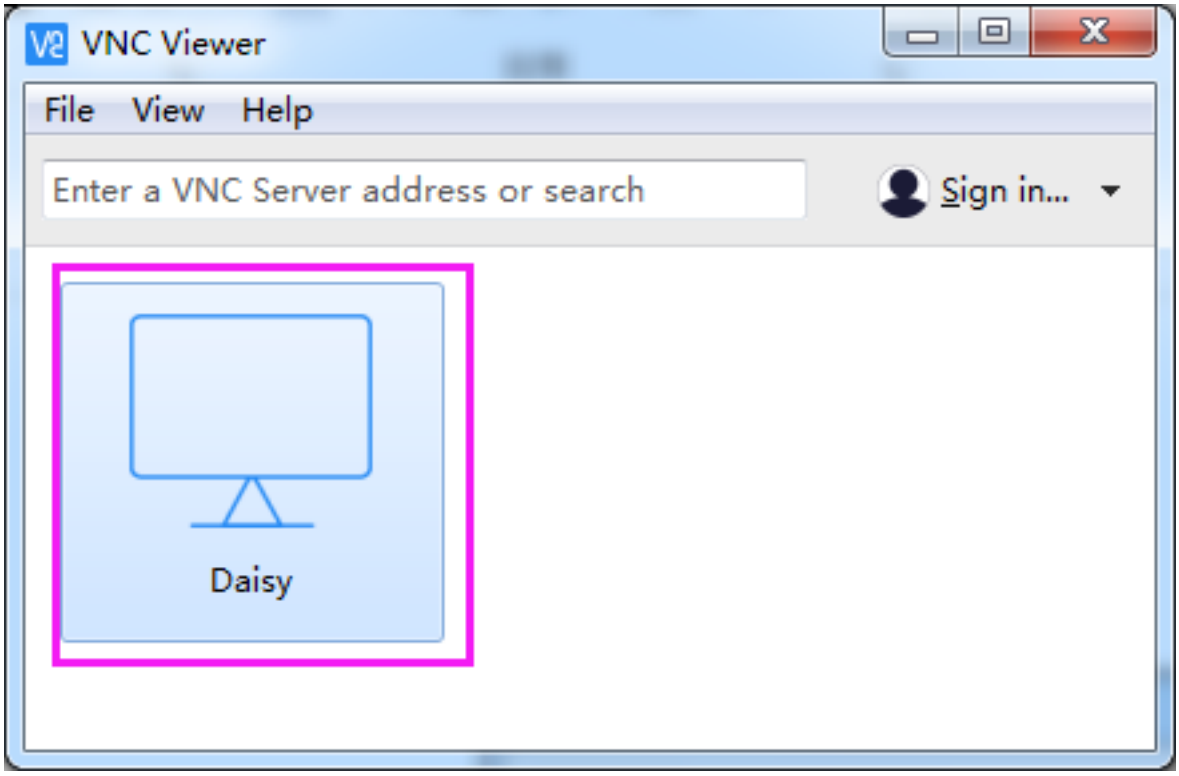
Step 3

Input IP address of Raspberry Pi and any **Name**.



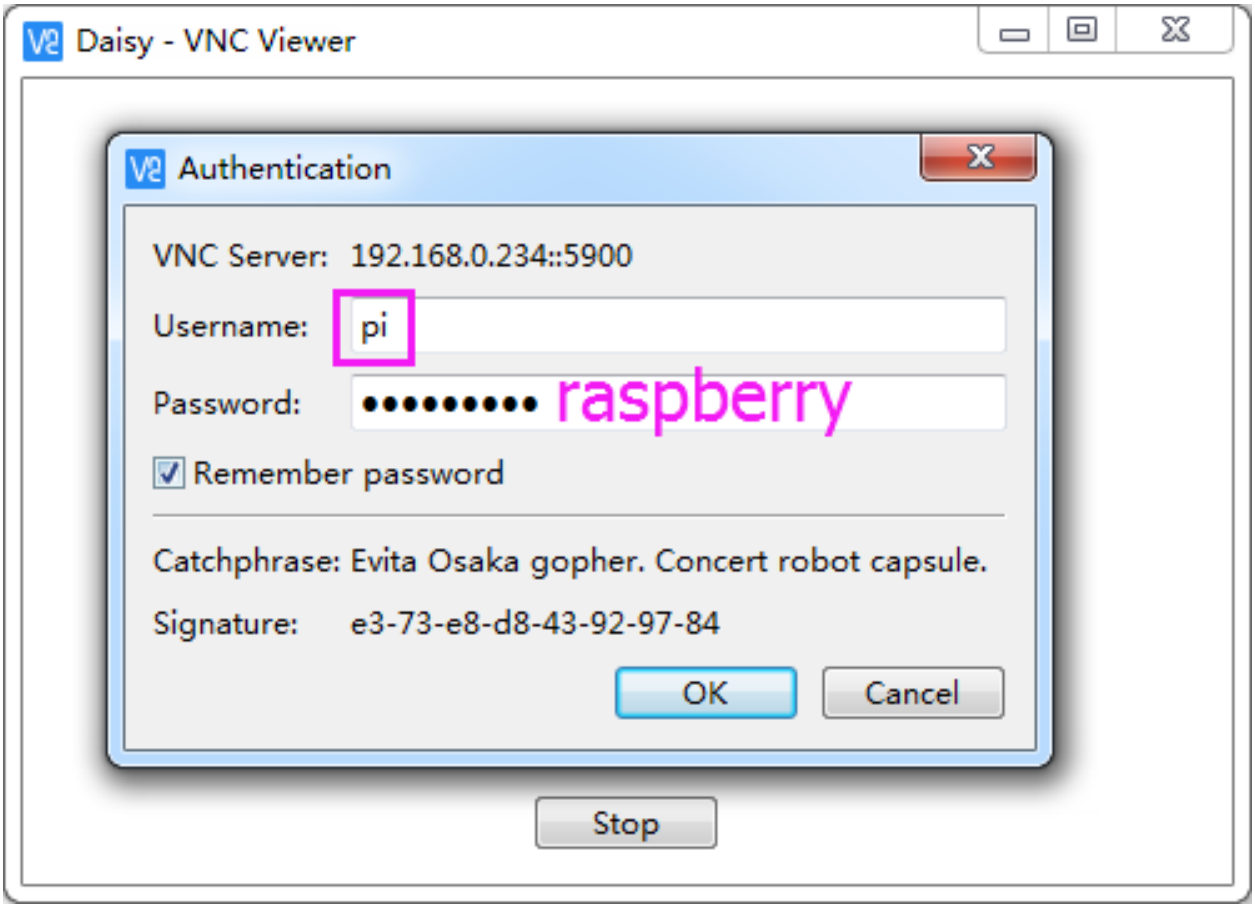
Step 4

Double click the **connection** just created:



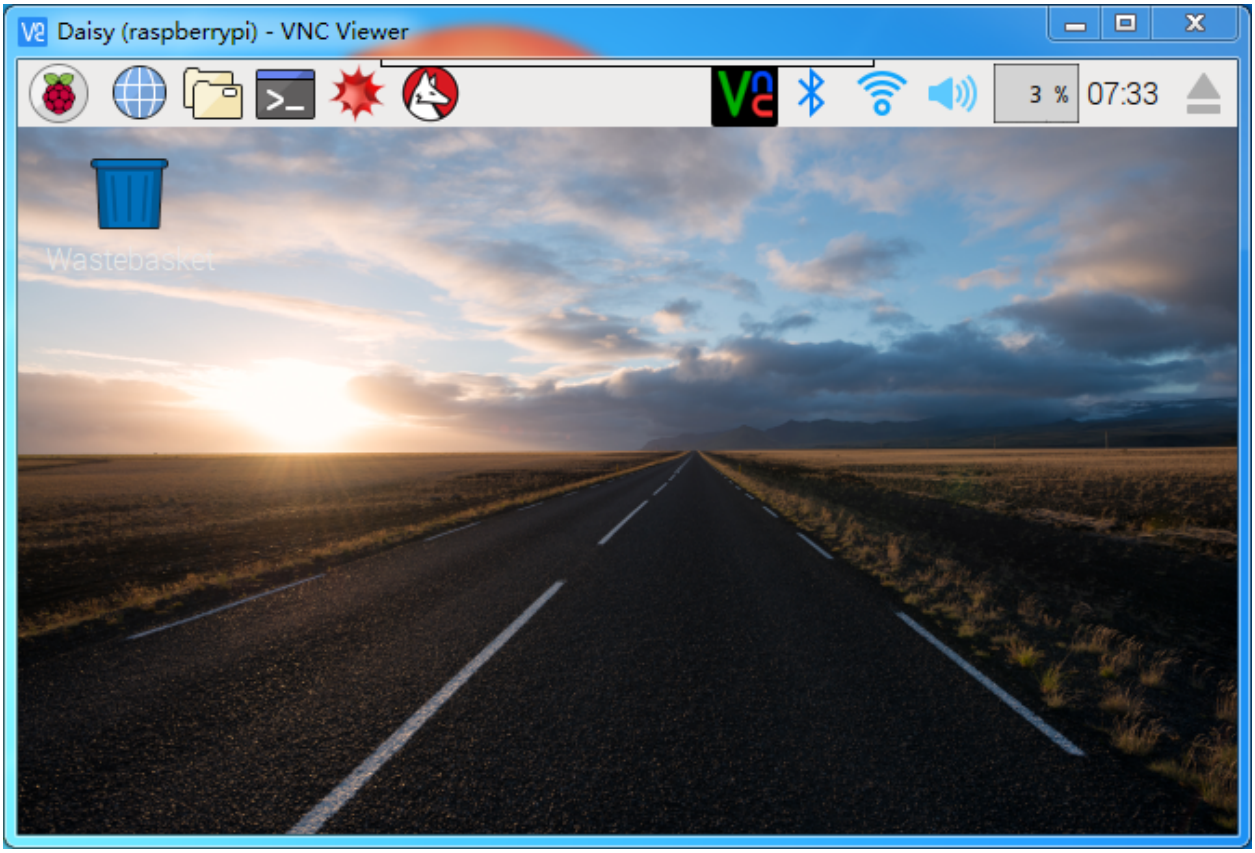
Step 5

Enter Username (**pi**) and Password (**raspberry** by default).



Step 6

Now you can see the desktop of the Raspberry Pi:



That's the end of the VNC part.

5.3.2 XRDP

Another method of remote desktop is XRDP, it provides a graphical login to remote machines using RDP (Microsoft Remote Desktop Protocol).

Install XRDP

Step 1

Login to Raspberry Pi by using SSH.

Step 2

Input the following instructions to install XRDP.

```
sudo apt-get update
sudo apt-get install xrdp
```

Step 3

Later, the installation starts.

Enter "Y", press key "Enter" to confirm.

```
pi@raspberrypi: ~
pi@raspberrypi:~ $ sudo apt-get install xrdp
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following extra packages will be installed:
  vnc4server x11-apps x11-session-utils xbase-clients xbitmaps xfonts-base
Suggested packages:
  vnc-java mesa-utils x11-xfs-utils
The following NEW packages will be installed:
  vnc4server x11-apps x11-session-utils xbase-clients xbitmaps xfonts-base
  xrdp
0 upgraded, 7 newly installed, 0 to remove and 0 not upgraded.
Need to get 8,468 kB of archives.
After this operation, 17.1 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

Step 4

Finished the installation, you should login to your Raspberry Pi by using Windows remote desktop applications.

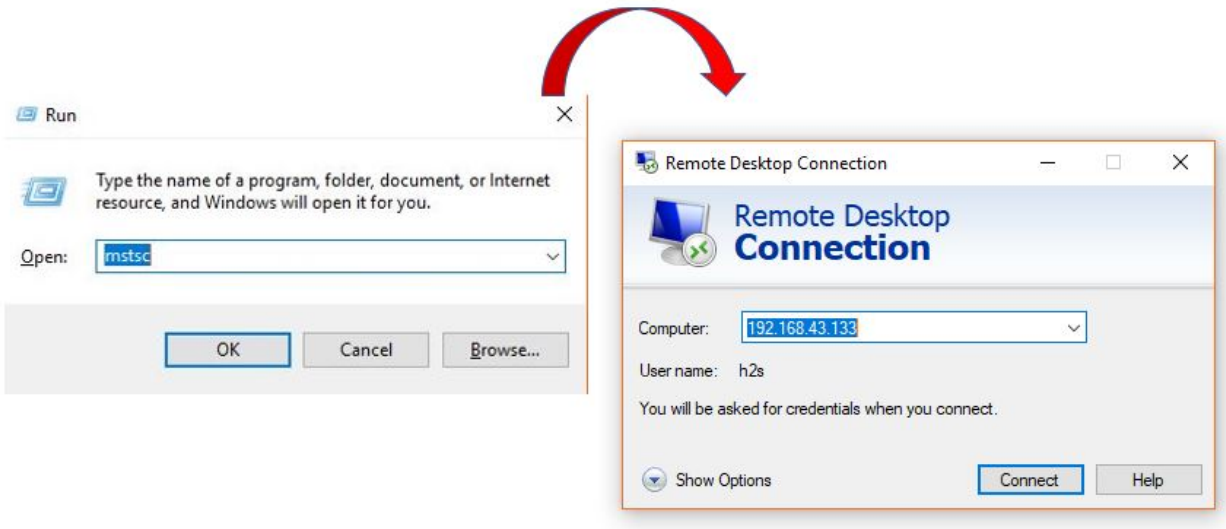
Login to XRDP

Step 1

If you are a Windows user, you can use the Remote Desktop feature that comes with Windows. If you are a Mac user, you can download and use Microsoft Remote Desktop from the APP Store, and there is not much difference between the two. The next example is Windows remote desktop.

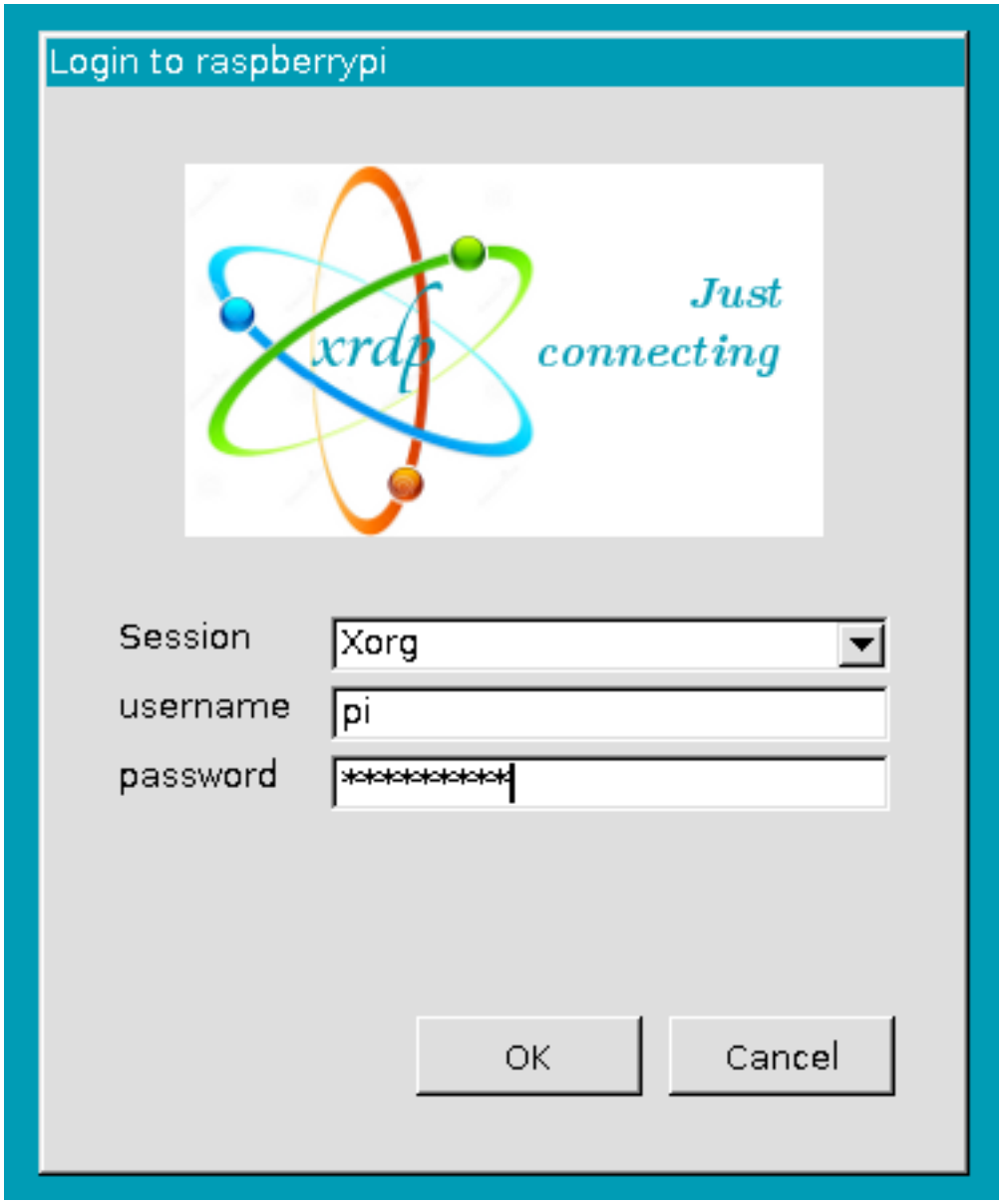
Step 2

Type in "**mstsc**" in Run (WIN+R) to open the Remote Desktop Connection, and input the IP address of Raspberry Pi, then click on "Connect".



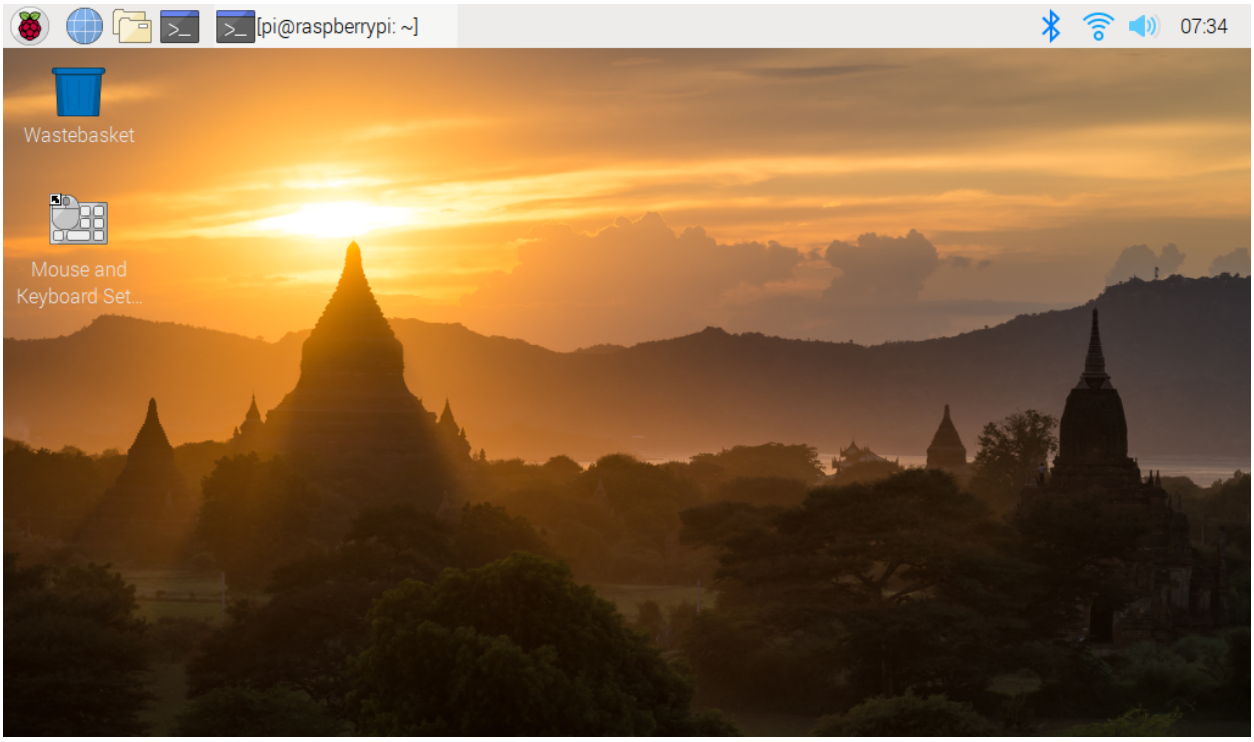
Step 3

Then the xrdp login page pops out. Please type in your username and password. After that, please click “OK”. At the first time you log in, your username is “pi” and the password is “raspberry”.



Step 4

Here, you successfully login to RPi by using the remote desktop.



5.4 About the Battery

Applicable Parameters

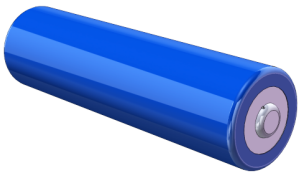
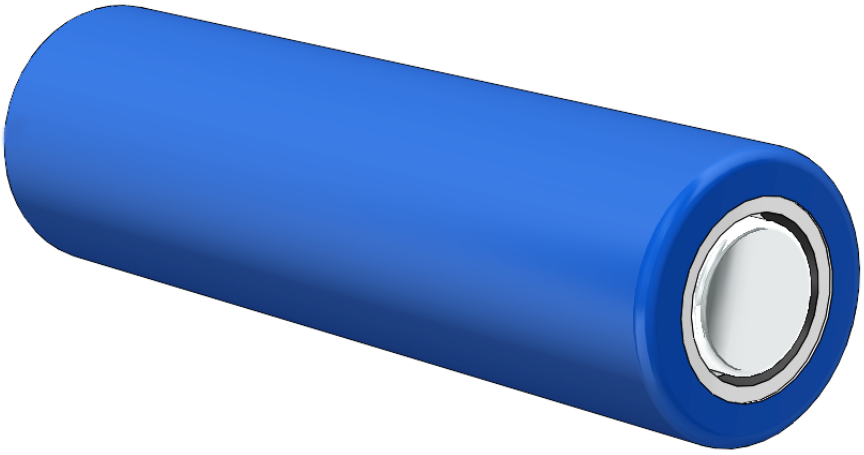
- 3.7V
- 18650
- Rechargeable
- Li-ion Battery
- Button Top
- No Protective Board

Note:

- Robot HAT cannot charge the battery, so you need to buy a battery charger.
- When the two power indicators on the Robot HAT are off, it means the power is too low and the batteries need to be charged.

Button Top vs Flat Top?

Please choose battery with button top to ensure a good connection between the battery and the battery holder.

Button Top	Flat Top
	

No protective board?

You are recommend to use 18650 batteries without a protective board. Otherwise, the robot may be cut power and stop running because of the overcurrent protection of the protective board.

Battery capacity?

In order to keep the robot working for a long time, use large-capacity batteries as much as possible. It is recommended to purchase batteries with a capacity of 3000mAh and above.

CHAPTER

SIX

COPYRIGHT NOTICE