
SunFounder PiCrawler Kit

www.sunfounder.com

Dec 28, 2021

CONTENTS

1	Component List and Assembly Instructions	3
2	About Robot HAT	5
3	Play with Python	7
3.1	Quick Guide on Python	7
3.2	Move	27
3.3	Keyboard Control	29
3.4	Sound Effect	31
3.5	Obstacle Avoidance	33
3.6	Computer Vision	35
3.7	Record Video	40
3.8	Bull Fight	42
3.9	Treasure Hunt	44
3.10	Pose	47
3.11	Adjust Posture	51
3.12	Record New Step	54
3.13	Twist	57
3.14	Emotional Robot	58
4	Play with Ezblock	61
4.1	Quick Guide on Ezblock	61
4.2	Move — Ezblock	63
4.3	Remote Control — Ezblock	66
4.4	Sound Effect — Ezblock	69
4.5	Obstacle Avoidance — Ezblock	72
4.6	Computer Vision — Ezblock	74
4.7	Bull Fight — Ezblock	76
4.8	Treasure Hunt — Ezblock	79
4.9	Pose — Ezblock	82
4.10	Adjust Posture — Ezblock	85
4.11	Record New Step — Ezblock	89
4.12	Twist — Ezblock	92
4.13	Emotional Robot — Ezblock	94
5	Appendix	97
5.1	Filezilla Software	97
5.2	I2C Configuration	99
5.3	Remote Desktop	101
5.4	About the Battery	110

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

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.

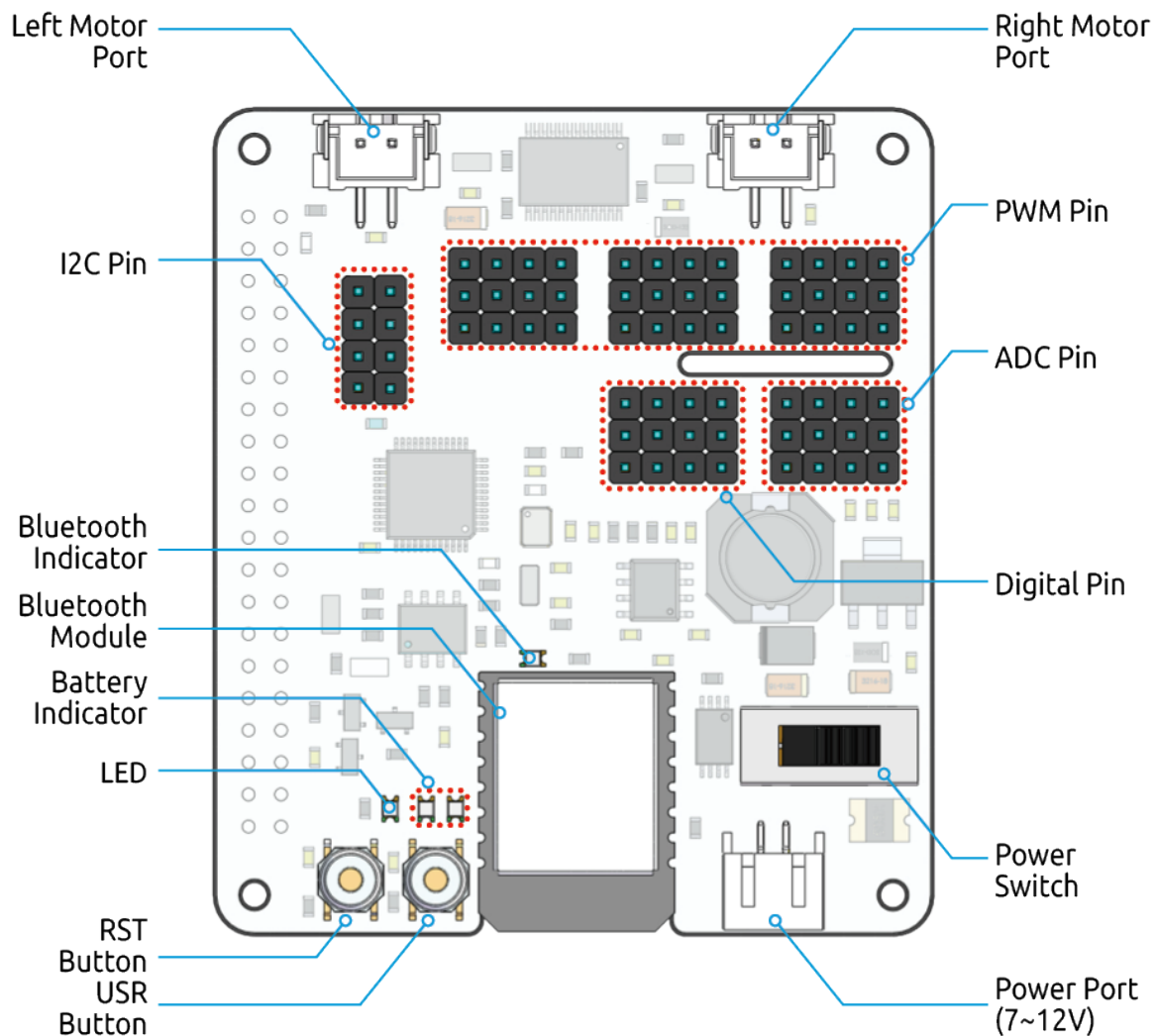
If the servo has been powered on, please do not turn the Servo shaft to avoid damage.

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.

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.

SunFounder PiCrawler Kit

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](#).

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.

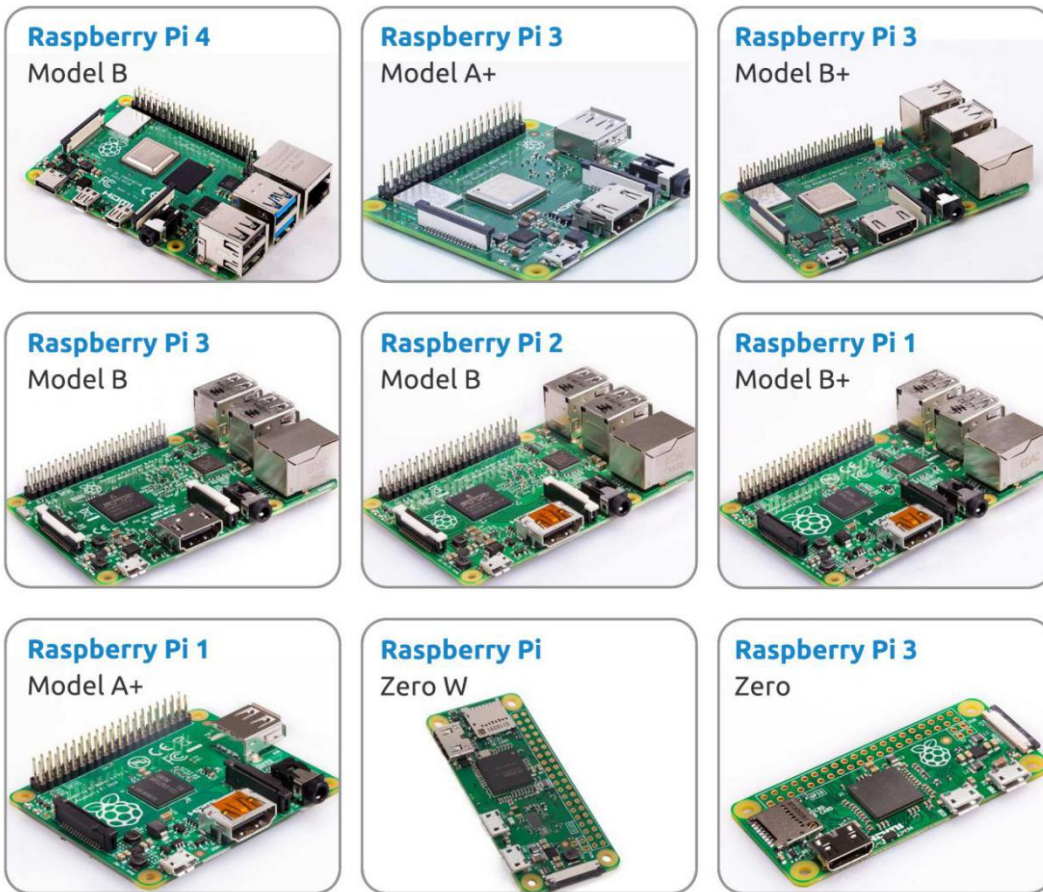
3.1.1 What Do We Need?

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.

Our kit applies to the following versions of the product of Raspberry Pi



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

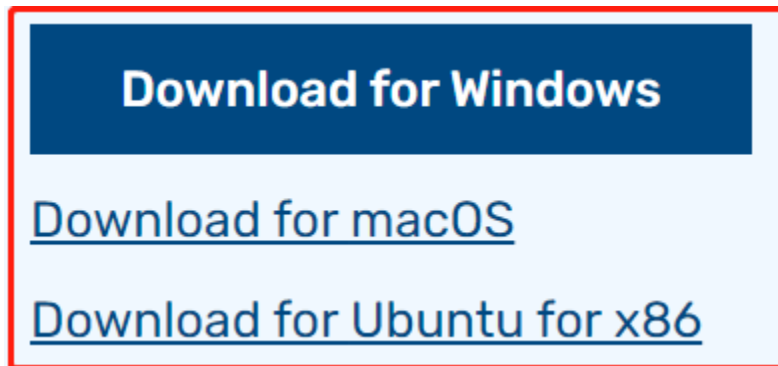
Required Components

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.

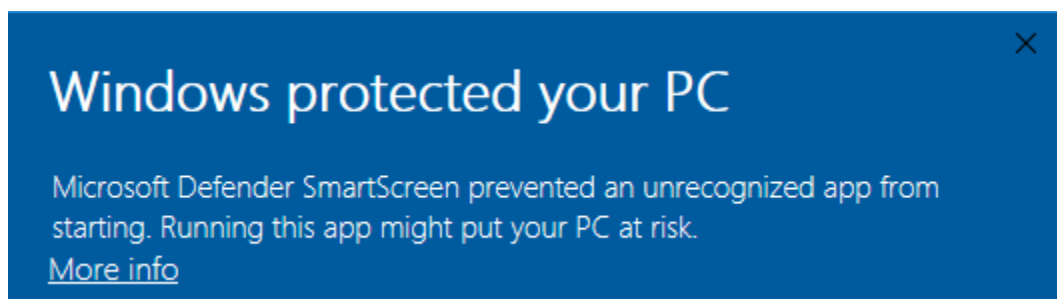
Visit the download page: <https://www.raspberrypi.org/software/>. Click on the link for the Raspberry Pi Imager that matches your operating system, when the download finishes, click it to launch the installer.



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.



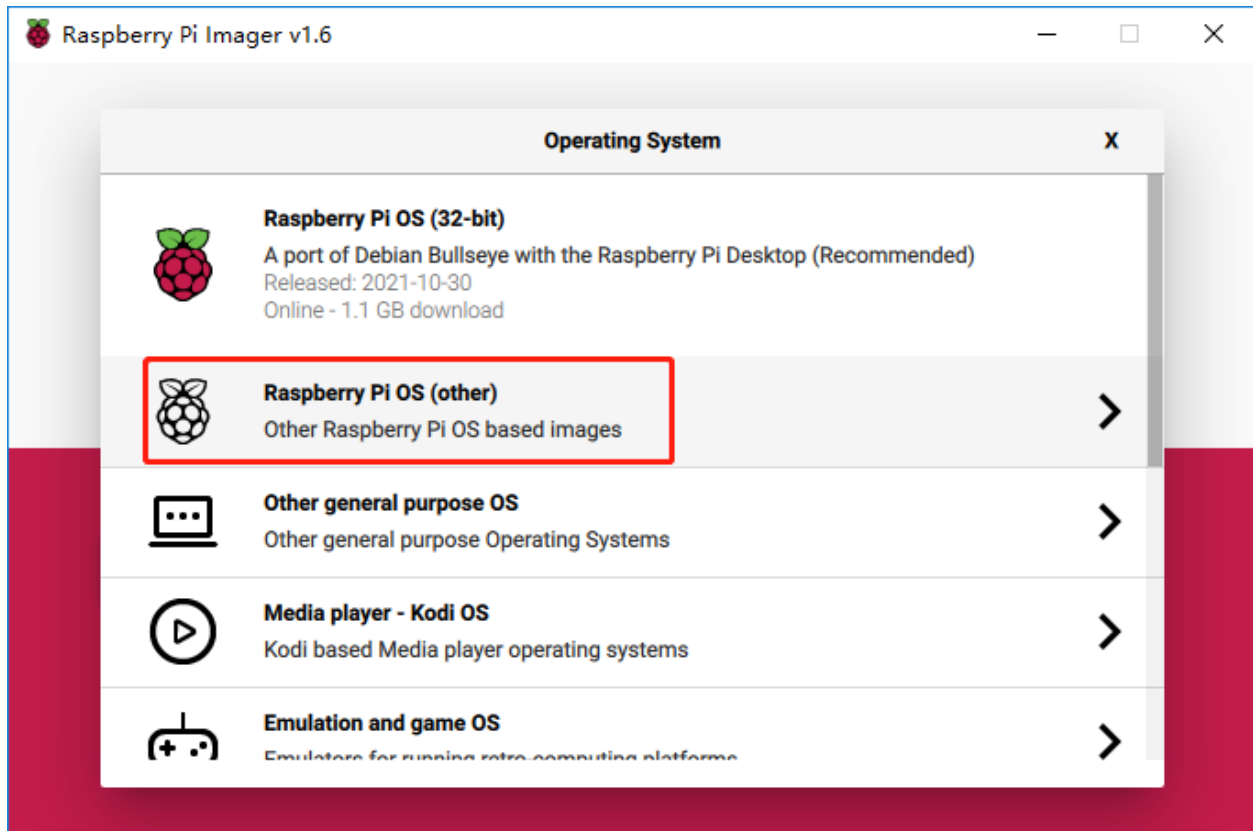
Step 3

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

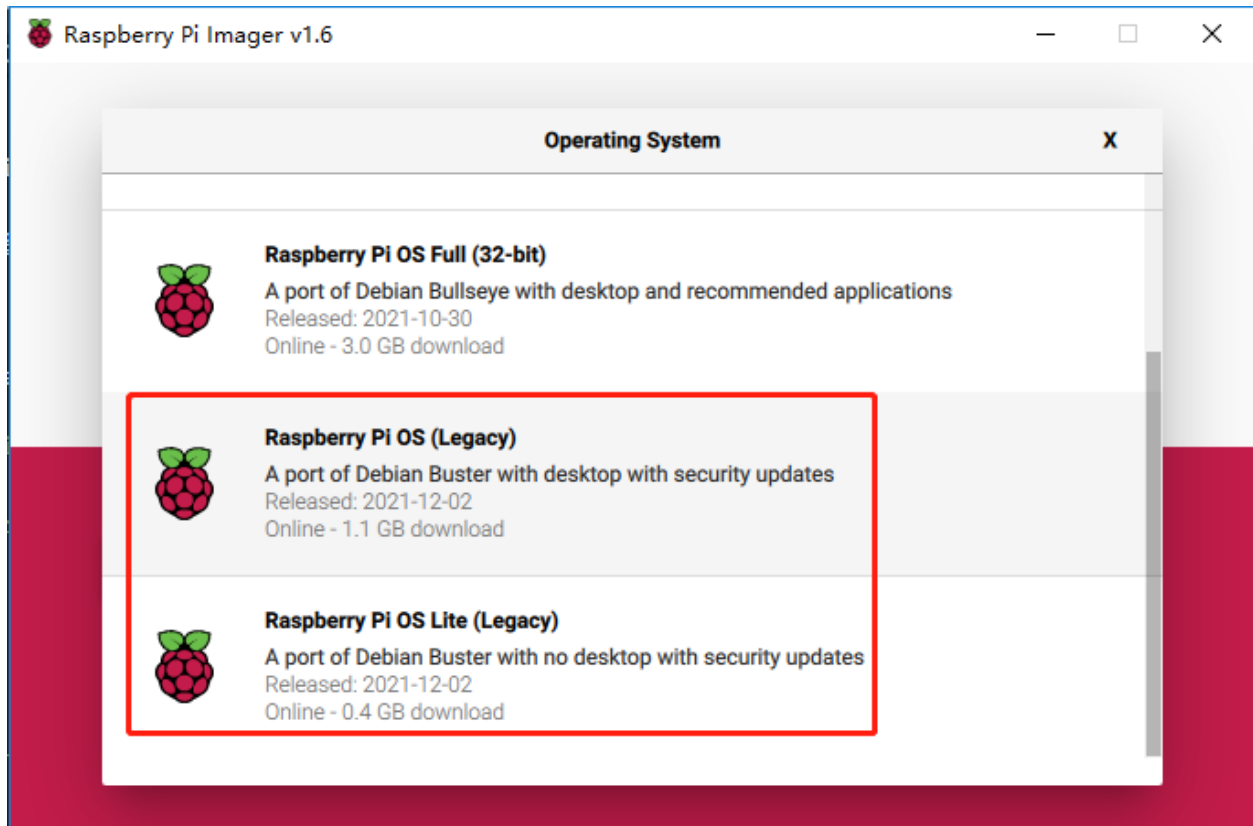
Step 4

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)**.

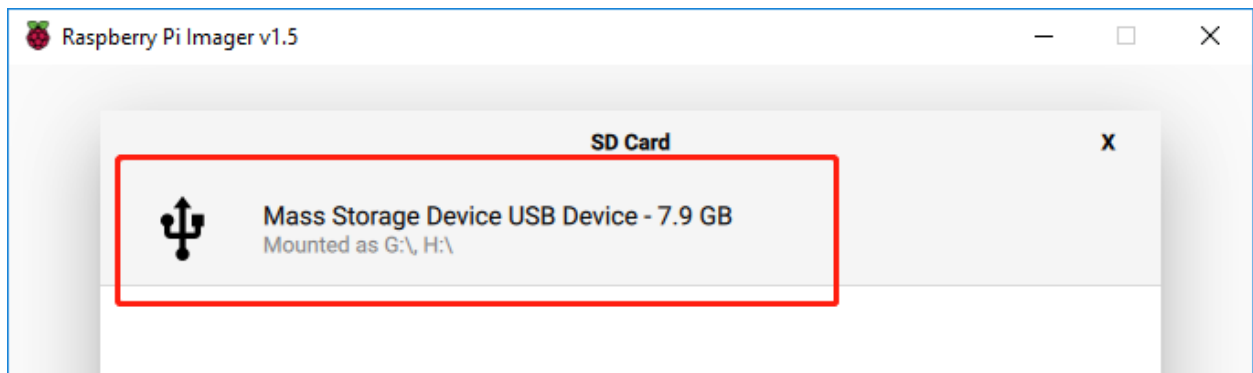


Scroll down to the end of the newly opened page and you will see **Raspberry Pi OS(Legacy)** and **Raspberry Pi OS Lite(Legacy)**, these are security updates for Debian Buster, the difference between them is with or without the desktop. It is recommended to install **Raspberry Pi OS(Legacy)**, the system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



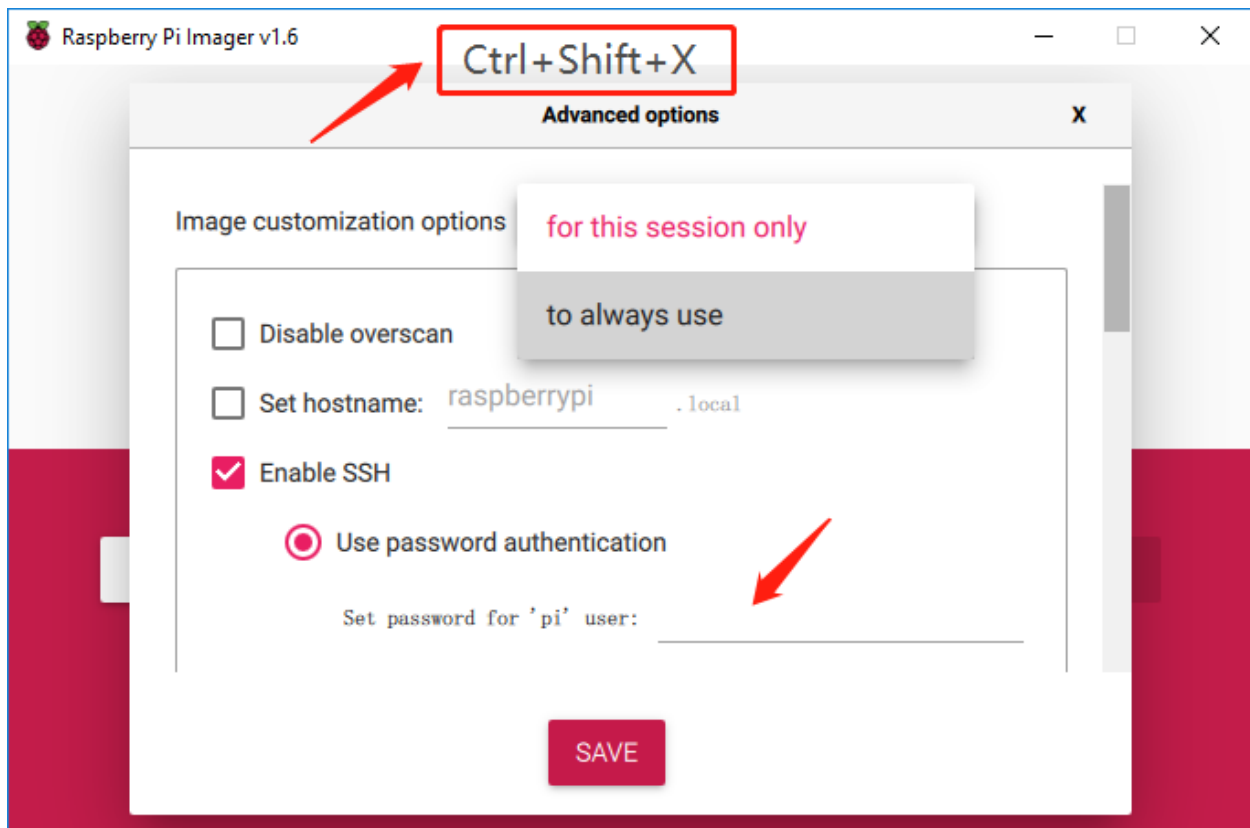
Step 5

Select the SD card you are using.



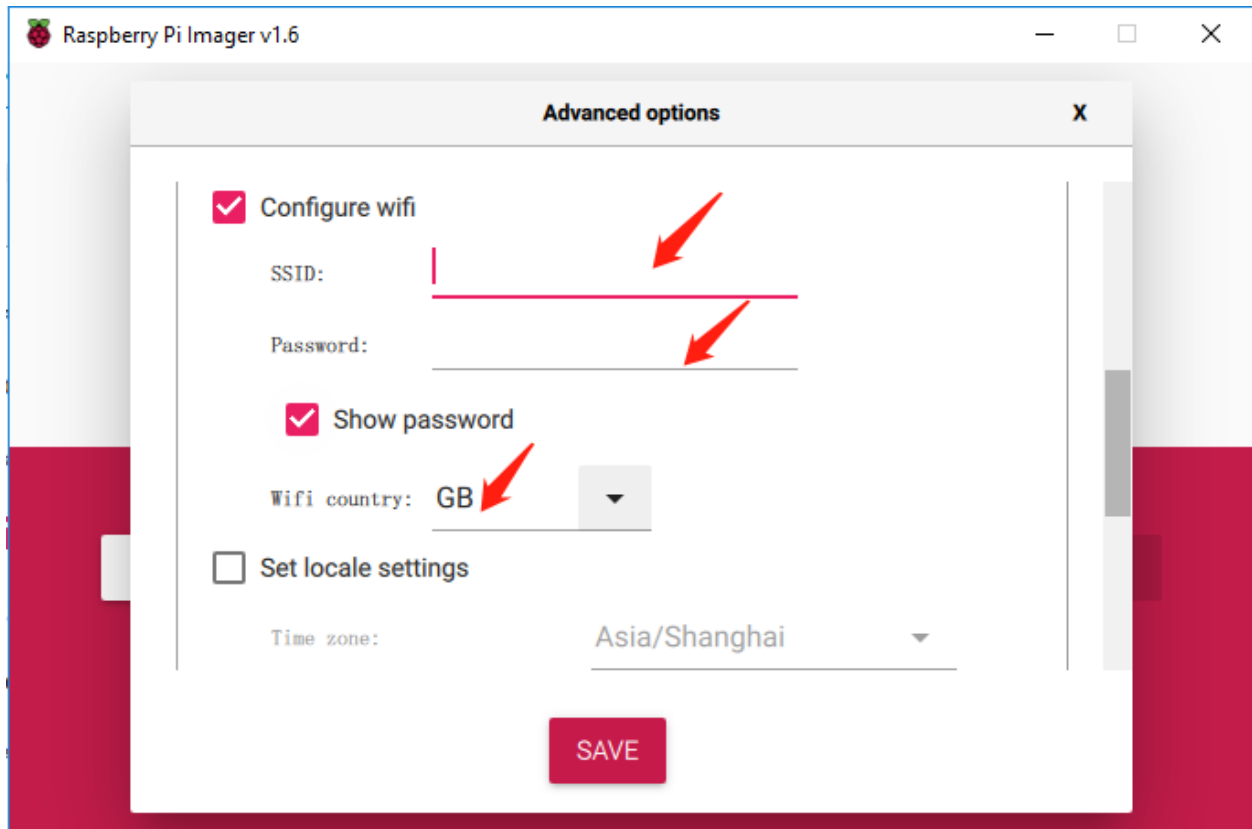
Step 6

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](https://en.wikipedia.org/wiki/ISO_3166-1_alpha-2#Officially_assigned_code_elements) 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

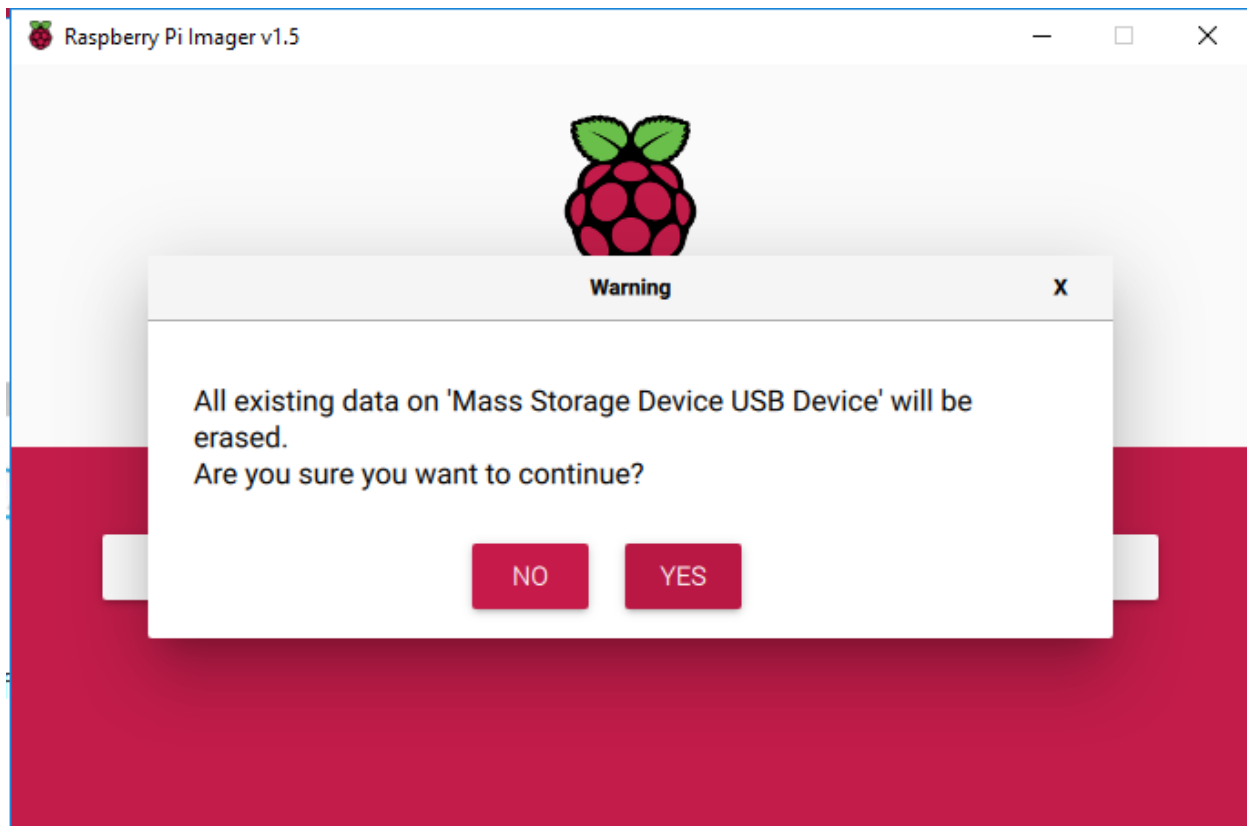


Step 7

Click the **WRITE** button.

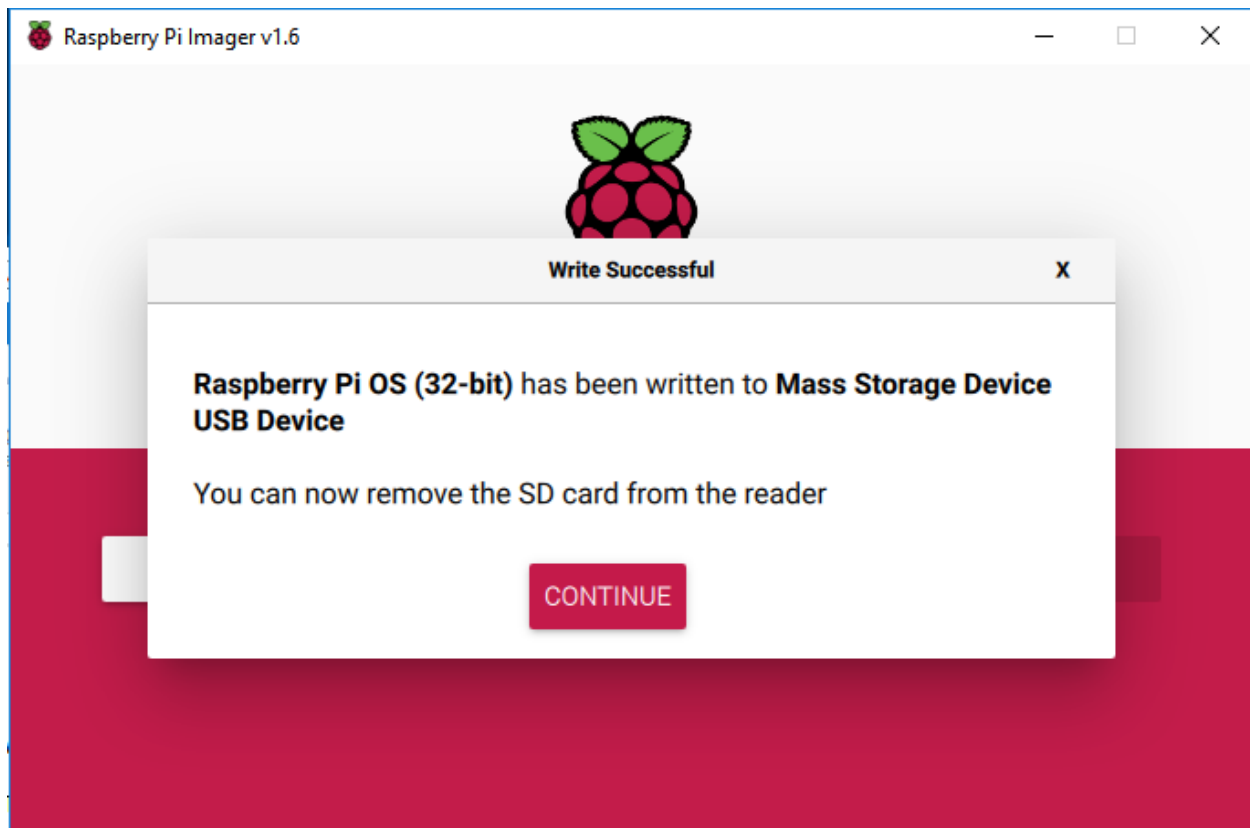
**Step 8**

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.

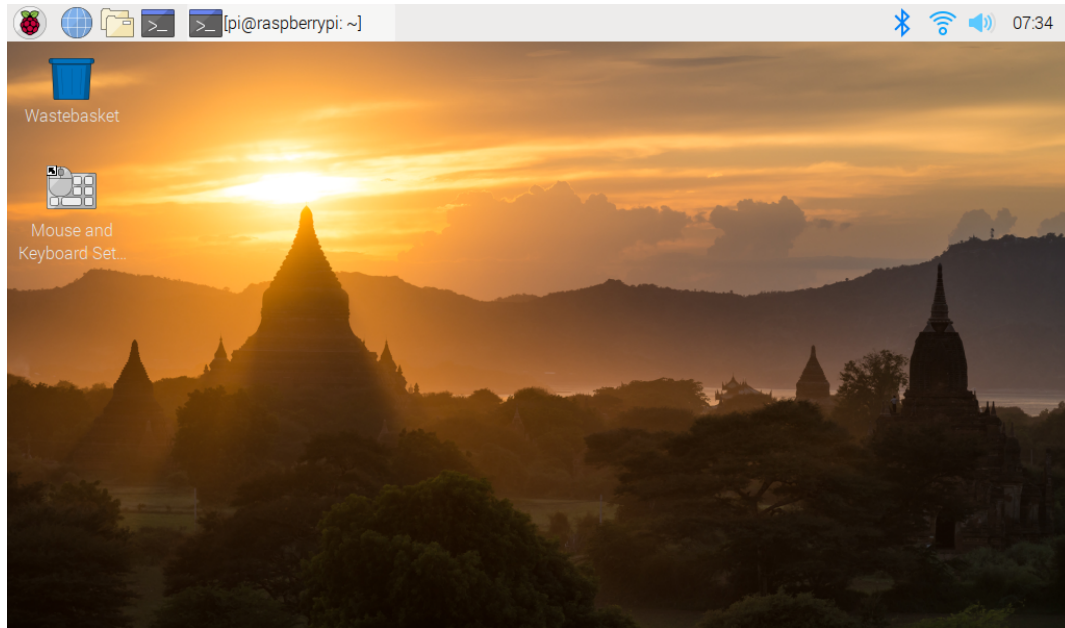
Required Components

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.

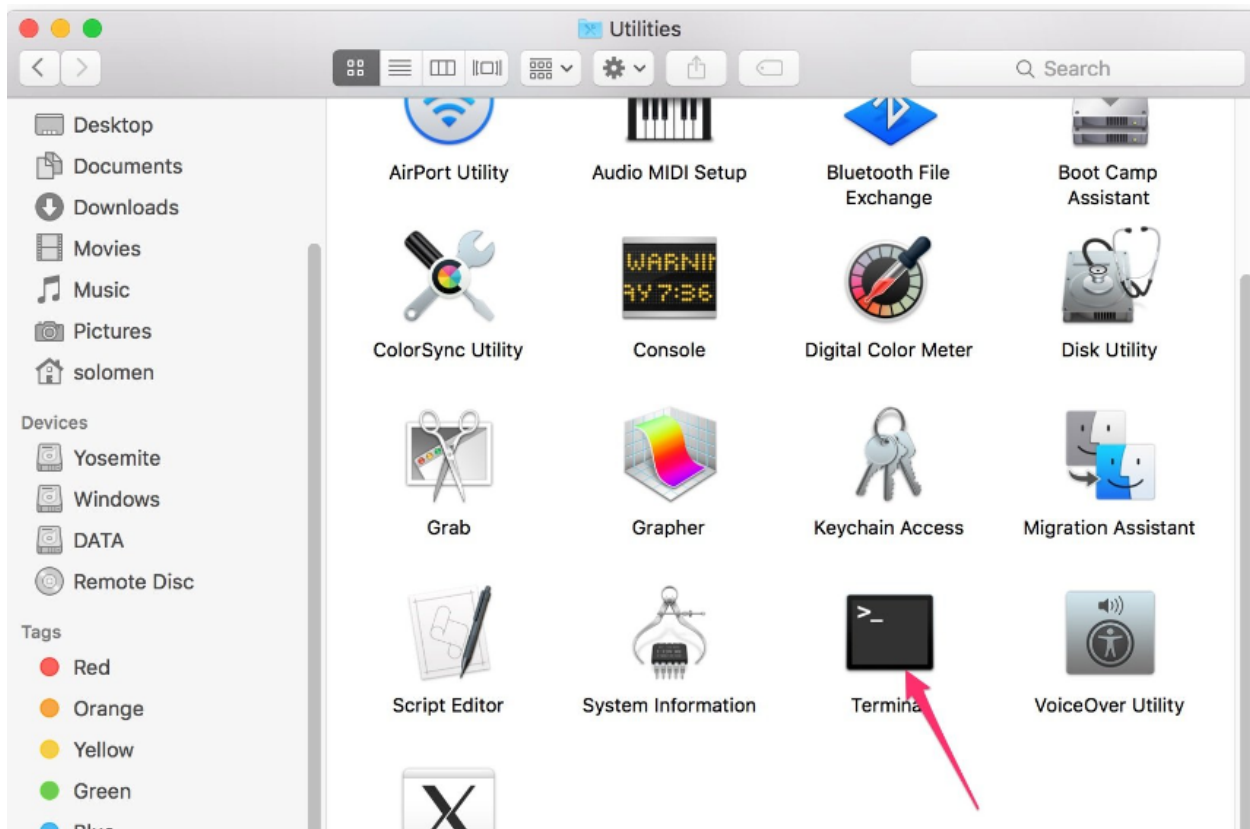
Use the SSH Remote Control

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.



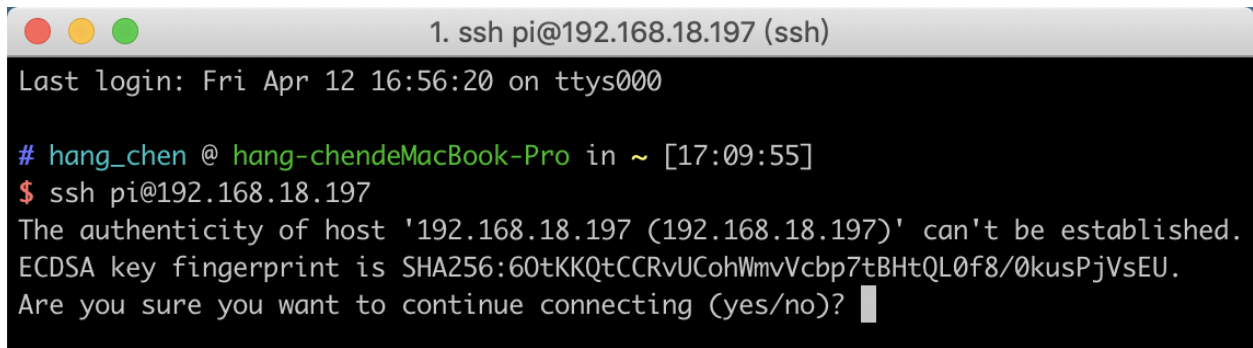
Step 2

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”.

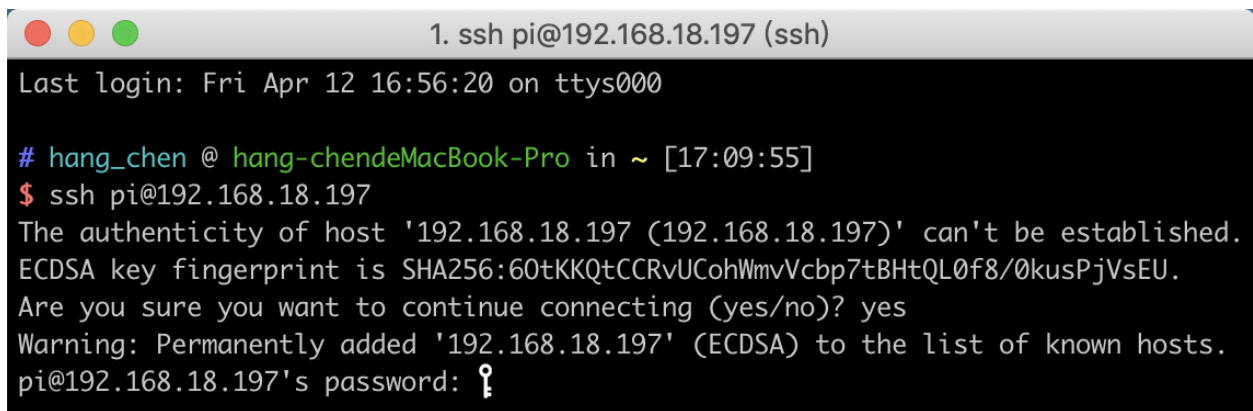


```
1. ssh pi@192.168.18.197 (ssh)
Last login: Fri Apr 12 16:56:20 on ttys000

# hang_chen @ hang-chendeMacBook-Pro in ~ [17:09:55]
$ ssh pi@192.168.18.197
The authenticity of host '192.168.18.197 (192.168.18.197)' can't be established.
ECDSA key fingerprint is SHA256:60tKKQtCCRvUCohWmvVcbp7tBHtQL0f8/0kusPjVsEU.
Are you sure you want to continue connecting (yes/no)?
```

Step 4

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

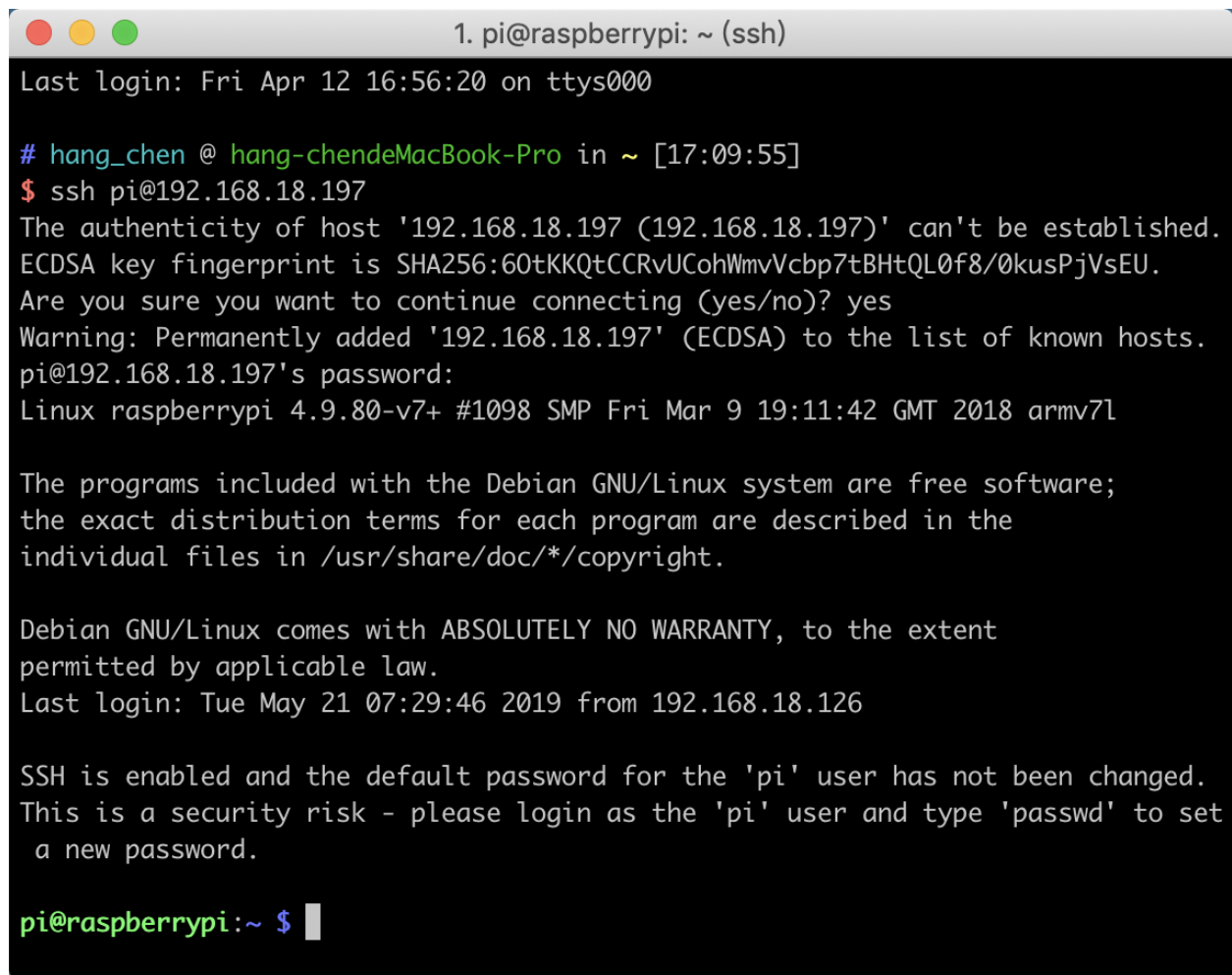


```
1. ssh pi@192.168.18.197 (ssh)
Last login: Fri Apr 12 16:56:20 on ttys000

# hang_chen @ hang-chendeMacBook-Pro in ~ [17:09:55]
$ ssh pi@192.168.18.197
The authenticity of host '192.168.18.197 (192.168.18.197)' can't be established.
ECDSA key fingerprint is SHA256:60tKKQtCCRvUCohWmvVcbp7tBHtQL0f8/0kusPjVsEU.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.18.197' (ECDSA) to the list of known hosts.
pi@192.168.18.197's password: 
```

Step 5

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

A terminal window titled '1. pi@raspberrypi: ~ (ssh)' with standard macOS window controls (red, yellow, green buttons). The terminal output shows a successful SSH connection from a Mac to a Raspberry Pi. It includes the last login time, the user's location and time, the SSH command used, host authenticity warnings, the password prompt, the Linux version, and a message about Debian GNU/Linux warranty and security. The prompt at the bottom is 'pi@raspberrypi:~ \$' with a cursor.

```
1. pi@raspberrypi: ~ (ssh)
Last login: Fri Apr 12 16:56:20 on ttys000

# hang_chen @ hang-chendeMacBook-Pro in ~ [17:09:55]
$ ssh pi@192.168.18.197
The authenticity of host '192.168.18.197 (192.168.18.197)' can't be established.
ECDSA key fingerprint is SHA256:60tKKQtCCRvUCohWmvVcbp7tBHtQL0f8/0kusPjVsEU.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.18.197' (ECDSA) to the list of known hosts.
pi@192.168.18.197's password:
Linux raspberrypi 4.9.80-v7+ #1098 SMP Fri Mar 9 19:11:42 GMT 2018 armv7l

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

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue May 21 07:29:46 2019 from 192.168.18.126

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

pi@raspberrypi:~ $
```

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

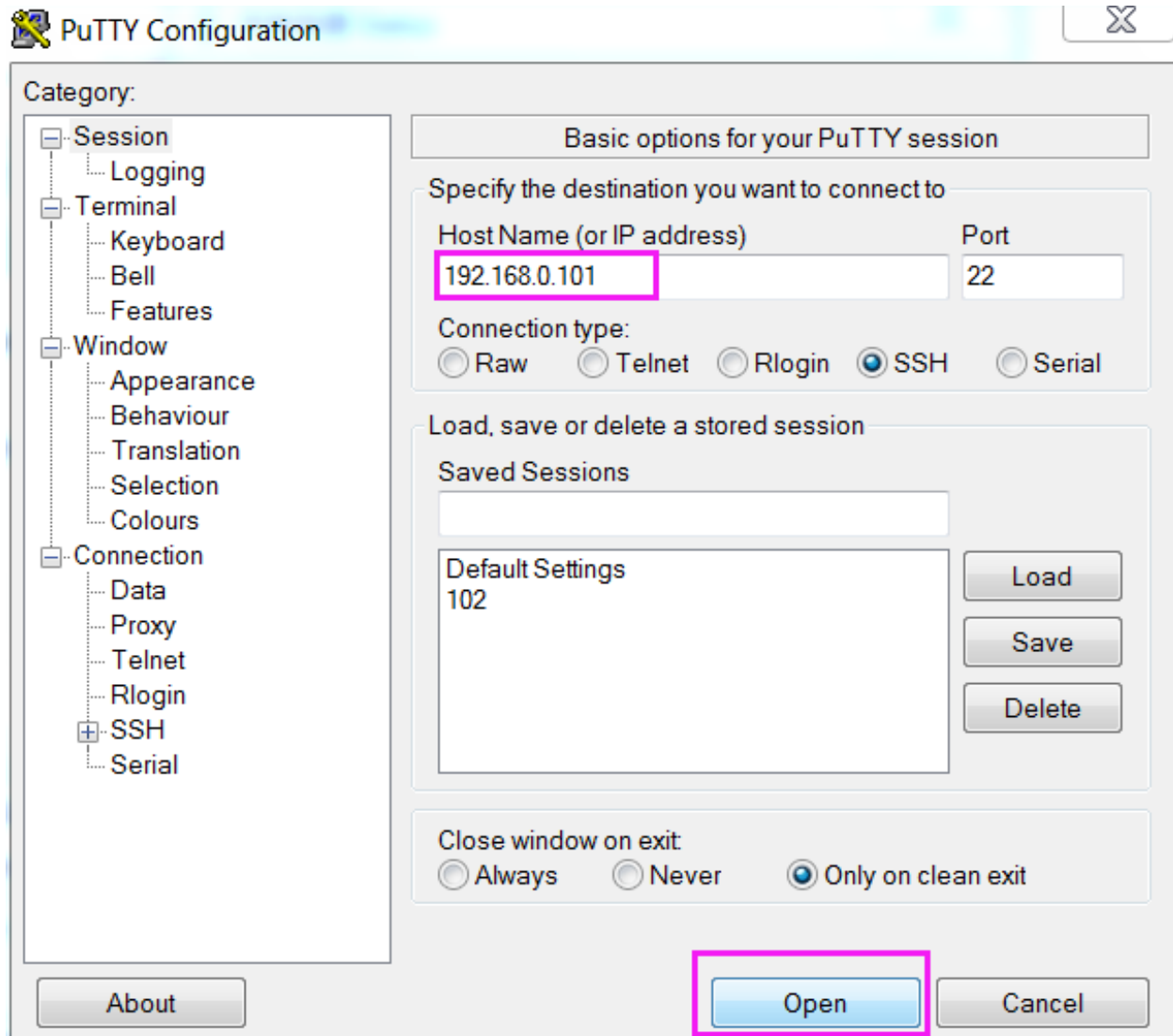
If you're a Windows user, you can use SSH with the application of some software. Here, we recommend **PuTTY**.

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

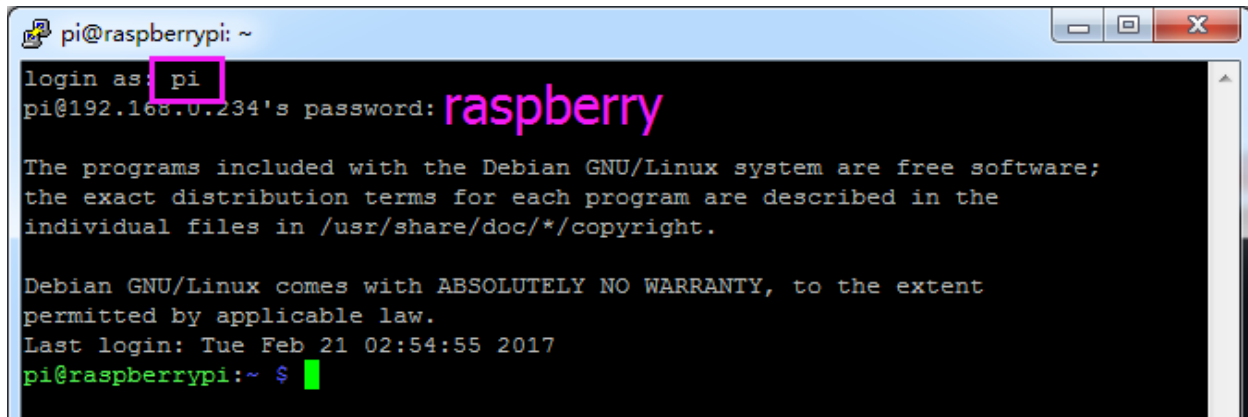
Click **Open**. Note that when you first log in to the Raspberry Pi with the IP address, there prompts a security reminder. Just click **Yes**.

Step 4

When the PuTTY window prompts “**login as:**”, type in “**pi**”(the user name of the RPi), and **password:** “**raspberrypi**” (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.



```
pi@raspberrypi: ~
login as: pi
pi@192.168.0.234's password: raspberrypi

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

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue Feb 21 02:54:55 2017
pi@raspberrypi:~ $
```

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

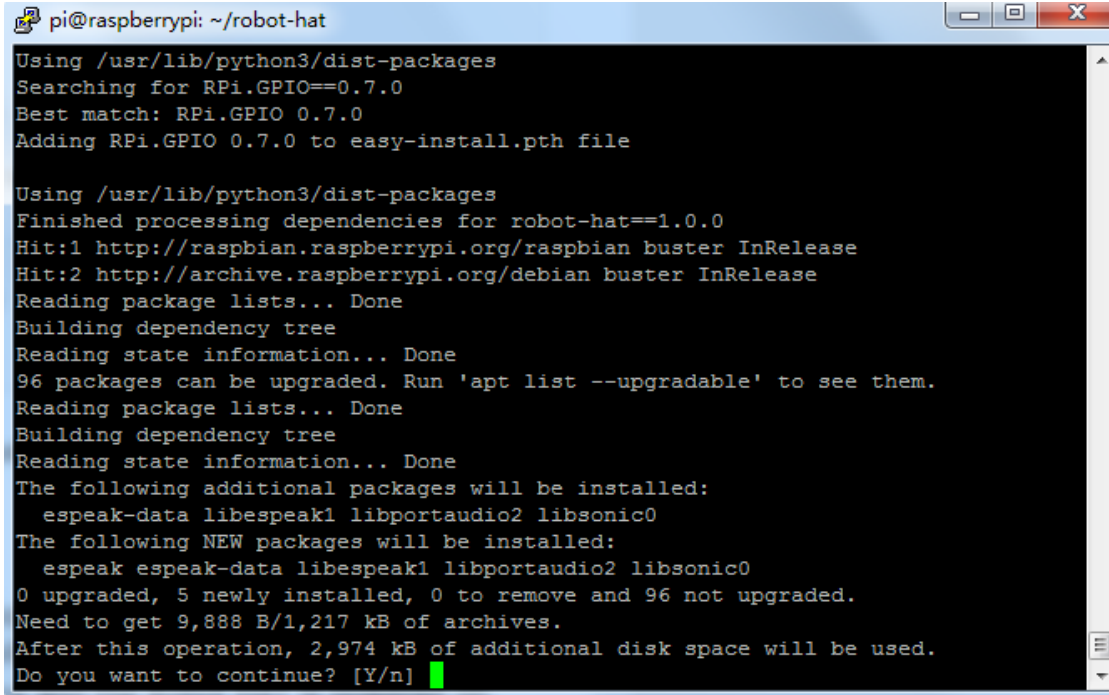
Download the Code Install 3 Libraries

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

Install `robot-hat` library 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.



```

pi@raspberrypi: ~/robot-hat
Using /usr/lib/python3/dist-packages
Searching for RPi.GPIO==0.7.0
Best match: RPi.GPIO 0.7.0
Adding RPi.GPIO 0.7.0 to easy-install.pth file

Using /usr/lib/python3/dist-packages
Finished processing dependencies for robot-hat==1.0.0
Hit:1 http://raspbian.raspberrypi.org/raspbian buster InRelease
Hit:2 http://archive.raspberrypi.org/debian buster InRelease
Reading package lists... Done
Building dependency tree
Reading state information... Done
96 packages can be upgraded. Run 'apt list --upgradable' to see them.
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  espeak-data libespeak1 libportaudio2 libsonic0
The following NEW packages will be installed:
  espeak espeak-data libespeak1 libportaudio2 libsonic0
0 upgraded, 5 newly installed, 0 to remove and 96 not upgraded.
Need to get 9,888 B/1,217 kB of archives.
After this operation, 2,974 kB of additional disk space will be used.
Do you want to continue? [Y/n]

```

Then download the code and install vilib library.

```

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

```

Then download the code and install picrawler library.

```

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 pisolot will have no sound.

```

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

```



```

pi@raspberrypi: ~/pisloth
File "/usr/local/lib/python3.7/dist-packages/robot_hat-1.0.0-py3.7.egg/robot_hat/robot.py", line 91, in servo_move
    time.sleep(step_delay)
KeyboardInterrupt
pi@raspberrypi:~/pisloth/examples $ cd ..
pi@raspberrypi:~/pisloth $ sudo bash i2samp.sh
Support for your operating system is experimental. Please visit
forums.adafruit.com if you experience issues with this product.

This script will install everything needed to use
i2s amplifier

--- Warning ---

Always be careful when running scripts and commands
copied from the internet. Ensure they are from a
trusted source.

If you want to see what this script does before
running it, you should run:
    \curl -sS github.com/adafruit/Raspberry-Pi-Installer-Scripts/i2samp

Do you wish to continue? [y/N] █

```

Type y and press Enter to continue running the script.

```

pi@raspberrypi: ~/pisloth
running it, you should run:
    \curl -sS github.com/adafruit/Raspberry-Pi-Installer-Scripts/i2samp

Do you wish to continue? [y/N] y

Checking hardware requirements...

Adding Device Tree Entry to /boot/config.txt
dtoverlay=hifiberry-dac
dtoverlay=i2s-mmap

Commenting out Blacklist entry in
/etc/modprobe.d/raspi-blacklist.conf

Disabling default sound driver
Configuring sound output

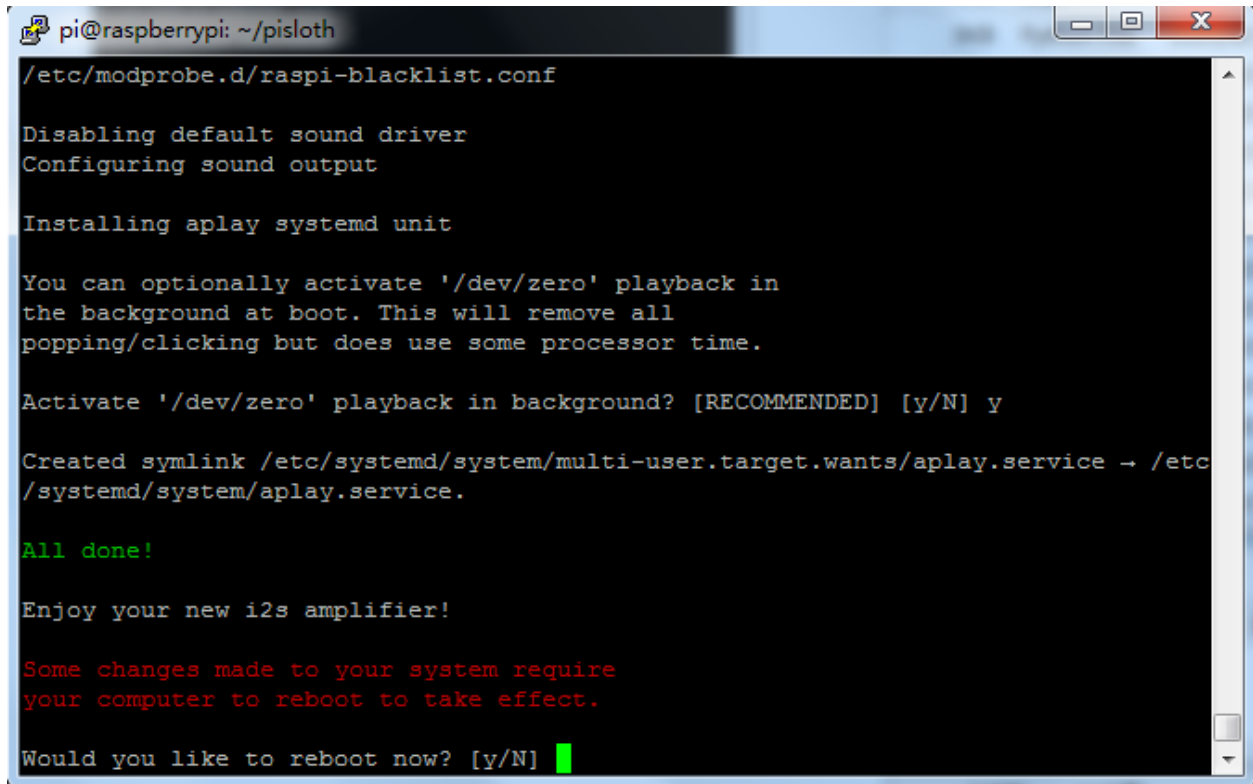
Installing aplay systemd unit

You can optionally activate '/dev/zero' playback in
the background at boot. This will remove all
popping/clicking but does use some processor time.

Activate '/dev/zero' playback in background? [RECOMMENDED] [y/N] █

```

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



```
pi@raspberrypi: ~/pisloth
/etc/modprobe.d/raspi-blacklist.conf

Disabling default sound driver
Configuring sound output

Installing aplay systemd unit

You can optionally activate '/dev/zero' playback in
the background at boot. This will remove all
popping/clicking but does use some processor time.

Activate '/dev/zero' playback in background? [RECOMMENDED] [y/N] y

Created symlink /etc/systemd/system/multi-user.target.wants/aplay.service -> /etc
/systemd/system/aplay.service.

All done!

Enjoy your new i2s amplifier!

Some changes made to your system require
your computer to reboot to take effect.

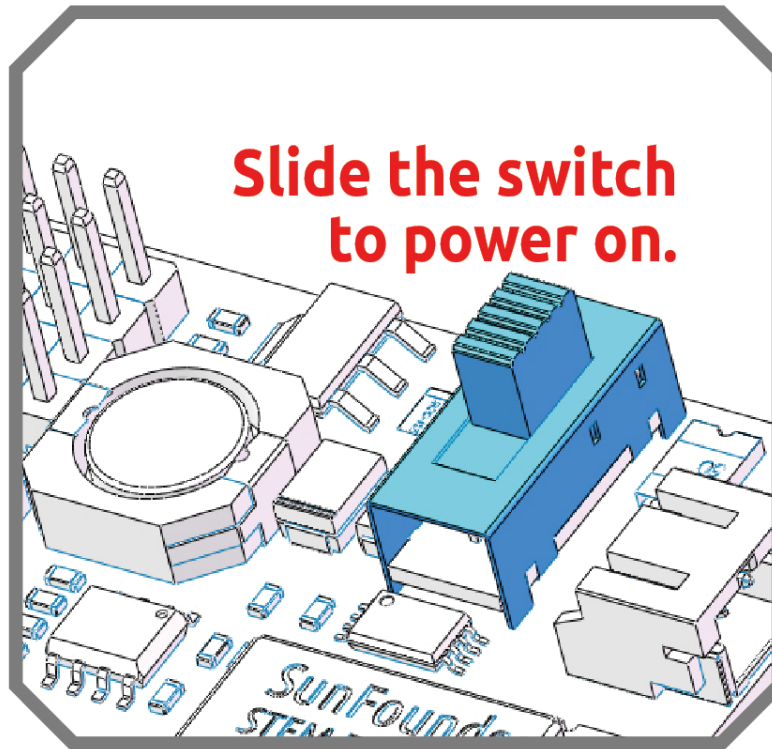
Would you like to reboot now? [y/N] █
```

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.

Run `servo_zeroing.py`

Because the servo is powered by the power supply on the Robot HAT, when you only supply power to the Raspberry Pi, the servo will not work. You need to make sure that the batteries are placed in the battery box and the Robot HAT is powered on.

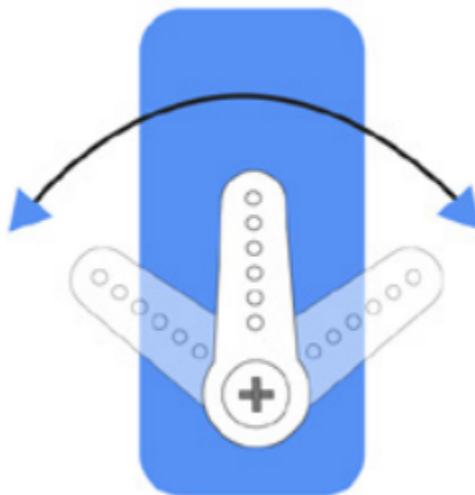


Before you run any code, enable your Raspberry Pi's I2C port, see: [I2C Configuration](#).

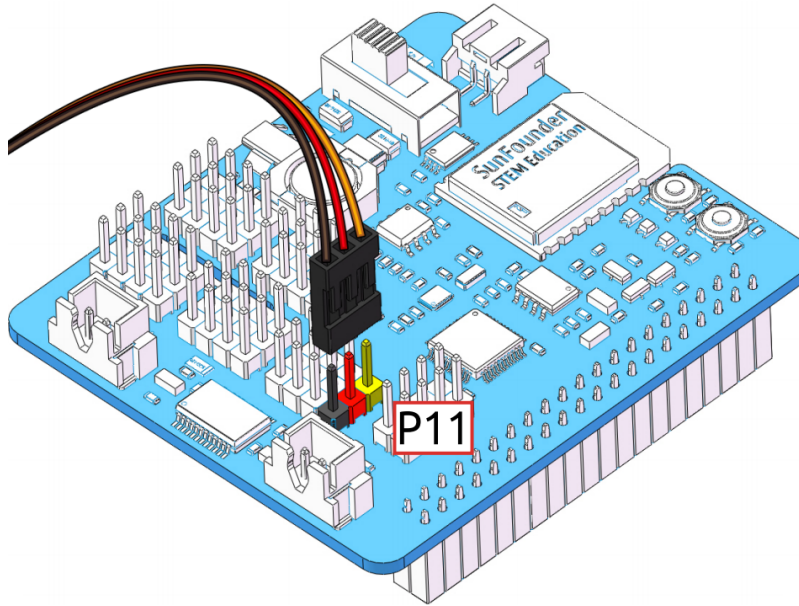
Now run the `servo_zeroing.py` in the `examples/` folder.

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

To make sure you can see that the servo has been set to 0°, you can insert a rocker arm in the servo shaft first and then turn the servo to another angle.



Now follow the diagram below and insert the servo to the P11 position.



So now if the servo arm returns after the servo arm returns, this function will take effect. If not, If not, please check the insertion direction of the servo cable and re-run the code.

Note: Before assembling each servo, you need to plug the servo pin into P11 and keep the power on.

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

3.2 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():
```

(continues on next page)

(continued from previous page)

```

speed = 100

while True:

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

3.3 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
```

Code

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

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 = 90

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

manual = '''
Press keys on keyboard to control PiSloth!
W: Forward
A: Turn left
S: Backward
D: Turn right
'''

def main():

    print(manual)

    while True:
        key = readchar()
        print(key)
        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:
```

(continues on next page)

(continued from previous page)

```

        crawler.do_action('turn right',1,speed)
    elif chr(27) == key:# 27 for ESC
        break

    sleep(0.05)
    print("\n q Quit")

if __name__ == "__main__":
    main()

```

How it works?

This function refers to the standard input stream and returns the first character of the data stream read.

- `tty.setraw(sys.stdin.fileno())` is to change the standard input stream to raw mode, that is, all characters will not be escaped during transmission, including special characters. Before changing the mode, backup the original mode, and restore it after the change.
- `old_settings = termios.tcgetattr(fd)` and `termios.tcsetattr(fd, termios.TCSADRAIN, old_settings)` plays the role of backup and restore.

```

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

```

Finally, according to the read keyboard characters, let PiCrawler do the actions we set.

```

key = readchar()
print(key)
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)
elif chr(27) == key:# 27 for ESC
    break

```

3.4 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.

Code

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

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 Speech
'''

def main():
    print(manual)

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

    while True:
        key = input()
        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()
```

(continues on next page)

(continued from previous page)

```
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.
- `music.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

3.5 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()

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:
```

(continues on next page)

(continued from previous page)

```

    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:
        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*.

3.6 Computer Vision

This project will officially enter the field of computer vision!

Run the Code

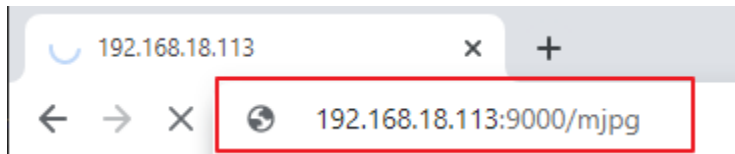
```
cd /home/pi/picrawler/examples
sudo python3 display.py
```

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`



Call the Function

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
```

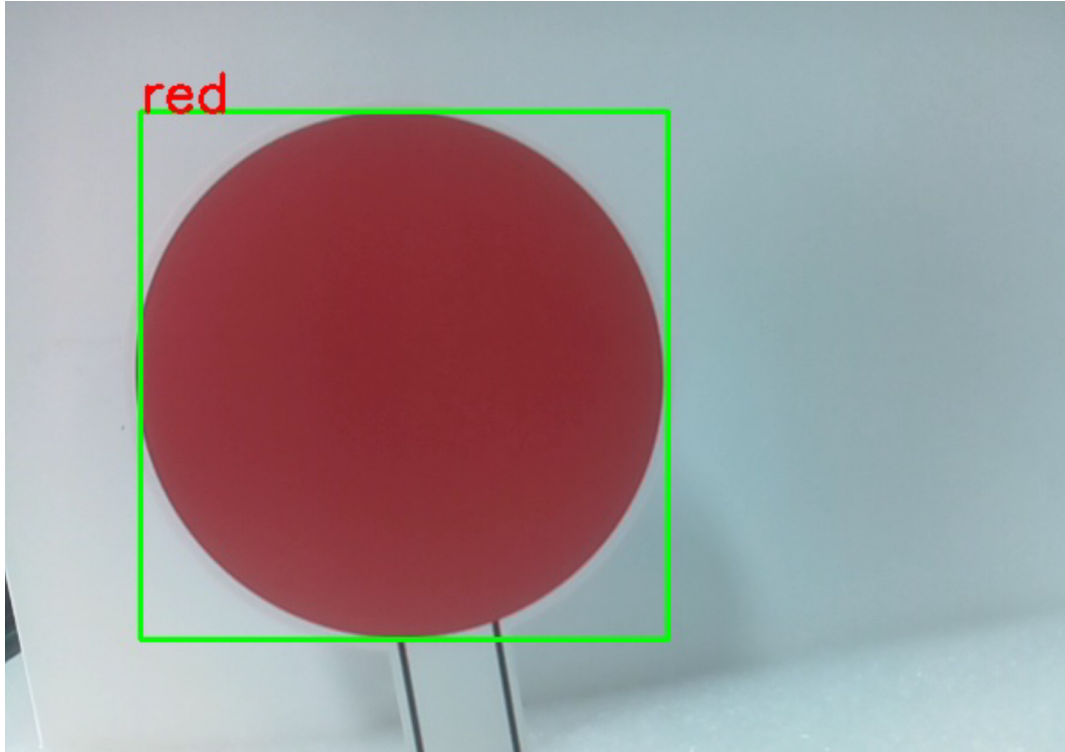
Please follow the prompts to activate the corresponding functions.

- **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**

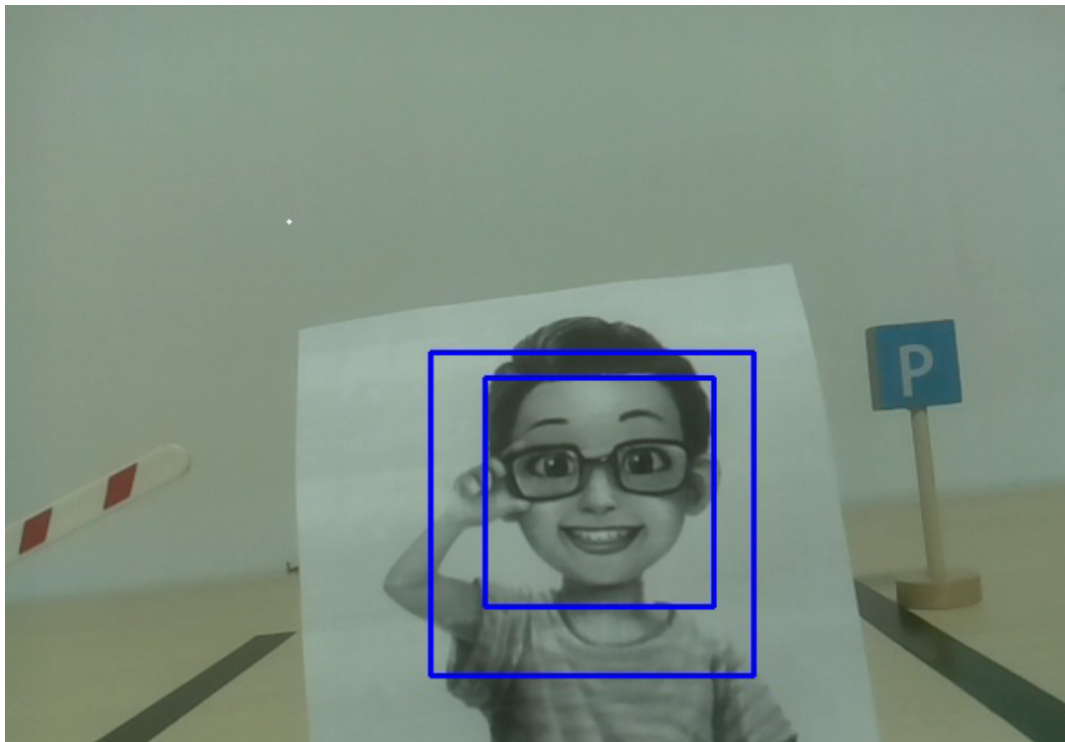
Entering a number between 1~6 will detect one of the colors in “red, orange, yellow, green, blue, purple”. Enter 0 to turn off color detection.



Note: You can download and print the PDF [Color Cards](#) for color detection.

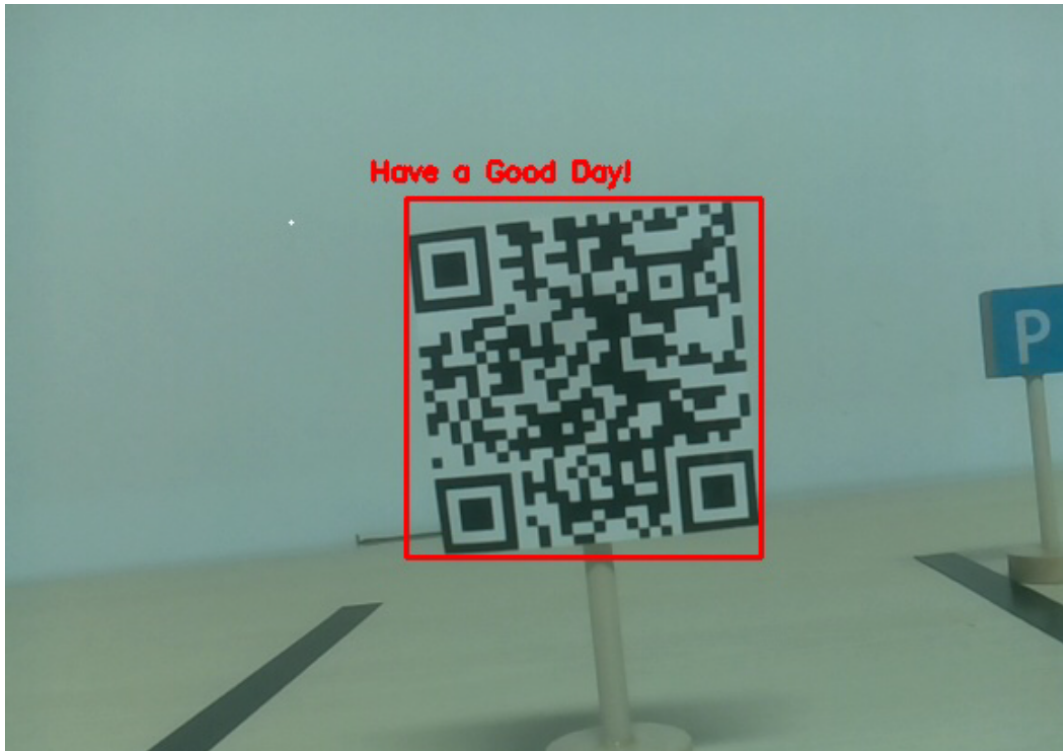
- **Face Detect**

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 vilib import Vilib
from time import sleep
import time

flag_face = False
flag_color = 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
'''
```

(continues on next page)

(continued from previous page)

```

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

def color_detect(color):
    print("detecting color : " + color)
    Vilib.color_detect(color)

def qrcode_detect():
    Vilib.qrcode_detect_switch(True)
    print("Waitting for QR code")
    while True:
        text = Vilib.detect_obj_parameter['qr_data']
        if text != "None":
            break
        sleep(0.5)
    print(text)
    sleep(0.5)
    Vilib.qrcode_detect_switch(False)

def take_photo():
    now = time.strftime('%Y-%m-%d-%H-%M-%S', time.localtime(time.time()))
    path = "/home/pi/Pictures/PiCrawler/"
    Vilib.take_photo('photo'+now, path)
    sleep(0.1)

def object_show():
    global flag_face, flag_color
    if flag_color is True and Vilib.detect_obj_parameter['color_n']!=0:
        color_coordinate = (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_coordinate, "Size", color_size)
        if flag_face is True and Vilib.detect_obj_parameter['human_n']!=0:
            human_coordinate = (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("[Human Detect] ", "Coordinate:", human_coordinate, "Size", human_size)

def main():
    Vilib.camera_start()
    Vilib.display()
    print(manual)

    global flag_face, flag_color

    while True:
        key = input()
        if key == "q":
            take_photo()
        elif key == "1":
            color_detect("red")

```

(continues on next page)

(continued from previous page)

```

        flag_color = True
    elif key == "2":
        color_detect("orange")
        flag_color = True
    elif key == "3":
        color_detect("yellow")
        flag_color = True
    elif key == "4":
        color_detect("green")
        flag_color = True
    elif key == "5":
        color_detect("blue")
        flag_color = True
    elif key == "6":
        color_detect("purple")
        flag_color = True
    elif key == "0":
        Vilib.color_detect_switch(False)
        flag_color = False
    elif key == "f":
        flag_face = not flag_face
        face_detect(flag_face)
    elif key == "r":
        qrcode_detect()
    elif key == "s":
        object_show()

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

3.7 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
```

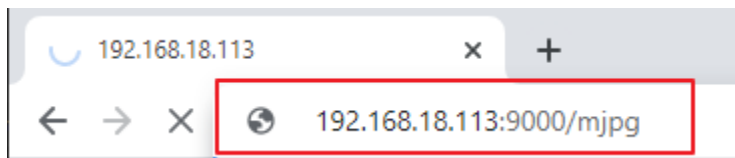

After the code runs, the recording function will start. You can input `w` to pause recording, input `q` to start/continue recording, and input `e` to stop recording.

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

```
from os import pardir
from time import sleep
from vilib import Vilib

def main():
    Vilib.camera_start()
    Vilib.display()

    Vilib.rec_video_set["path"] = "/home/pi/video/test/"
    vname = "vtest"
    Vilib.rec_video_run(vname)
    print('start rec ...')
    while True:
        if input() == 'q':
            Vilib.rec_video_start()
            print('continue')
        if input() == 'w':
            Vilib.rec_video_pause()
            print('pause')
        if input() == 'e':
            Vilib.rec_video_stop()
            print('stop')
            print("The video saved as", Vilib.rec_video_set["path"], vname)

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.

3.8 Bull Fight

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

Run the Code

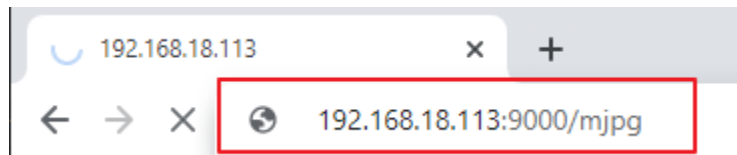
```
cd /home/pi/picrawler/examples
sudo python3 bull_fight.py
```

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")
```

(continues on next page)

(continued from previous page)

```
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')

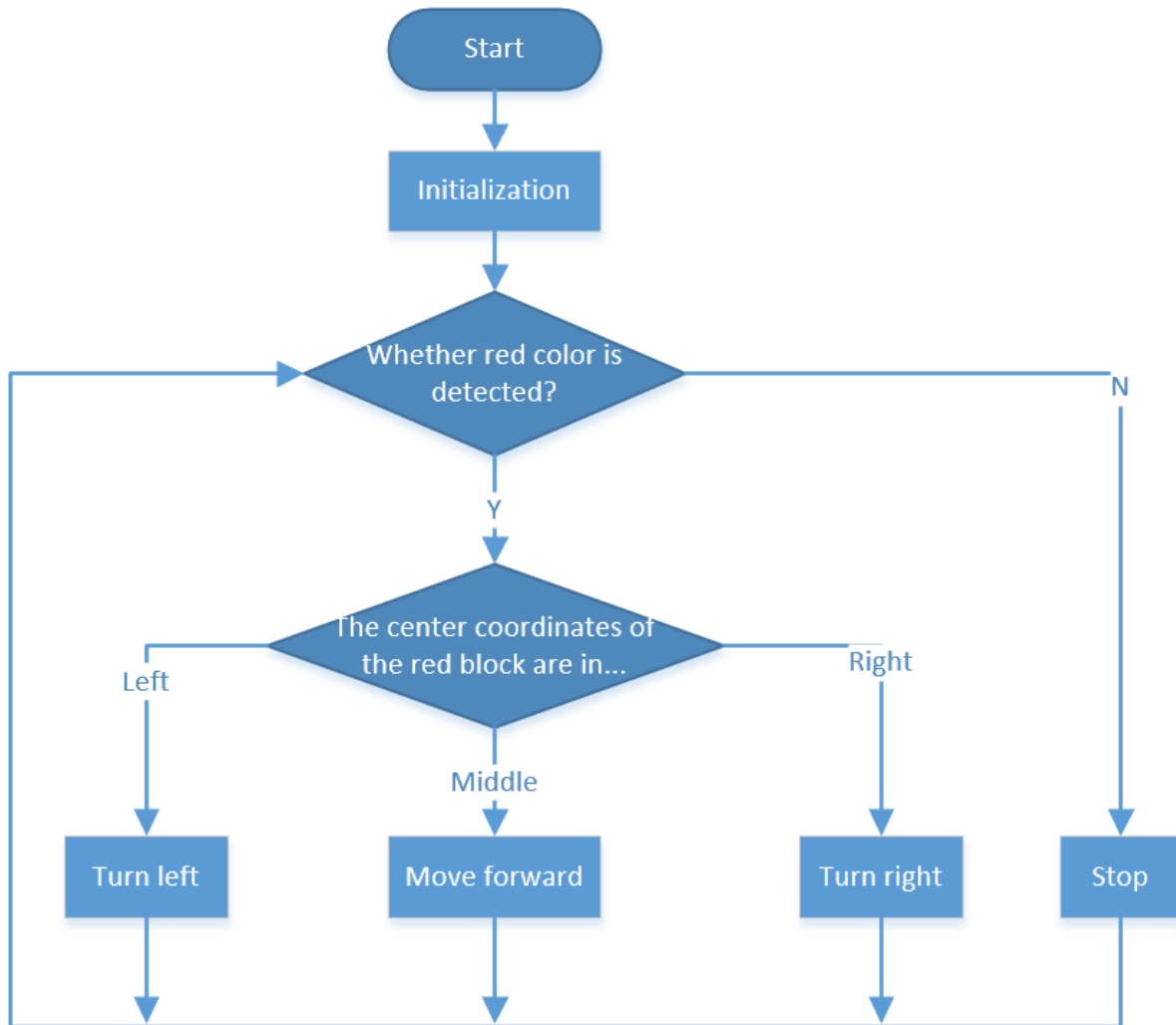
        if coordinate_x < 100:
            crawler.do_action('turn left',1,speed)
            sleep(0.05)
        elif coordinate_x > 220:
            crawler.do_action('turn right',1,speed)
            sleep(0.05)
        else :
            crawler.do_action('forward',2,speed)
            sleep(0.05)
    else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



3.9 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.

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: `https://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 sys
import tty
import termios
import random

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()

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

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"]
```

(continues on next page)

```

def renew_color_detect():
    global color
    color = random.choice(color_list)
    Vilib.color_detect(color)
    tts.say("Look for " + color)

def main():
    Vilib.camera_start()
    Vilib.display()
    speed = 100
    print(manual)

    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()

        key = readchar()
        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)
        elif chr(32) == key:
            tts.say("Look for " + color)
        elif chr(27) == key:# 27 for ESC
            break

        sleep(0.05)
    print("\n q Quit")

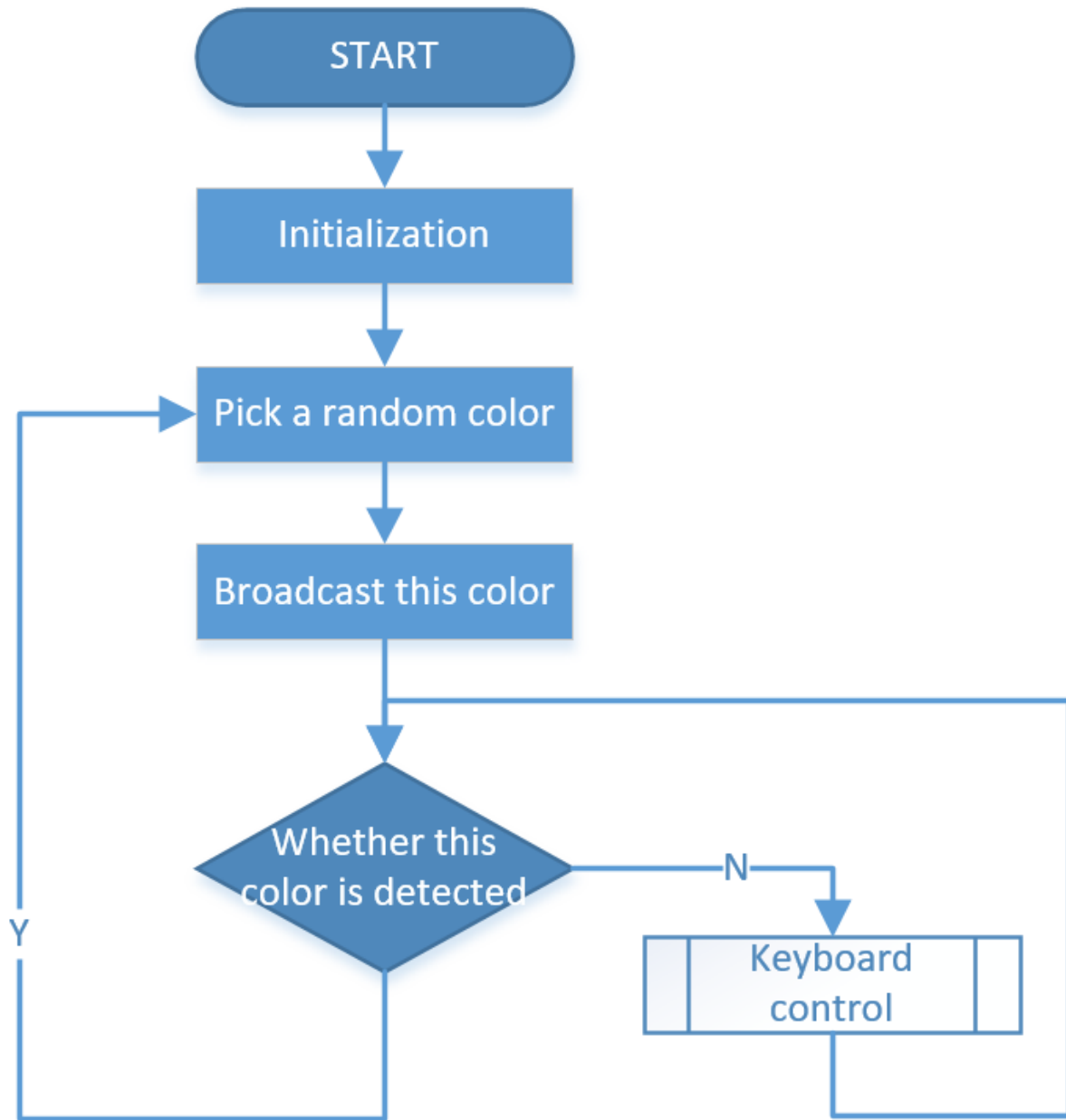
if __name__ == "__main__":
    main()

```

How it works?

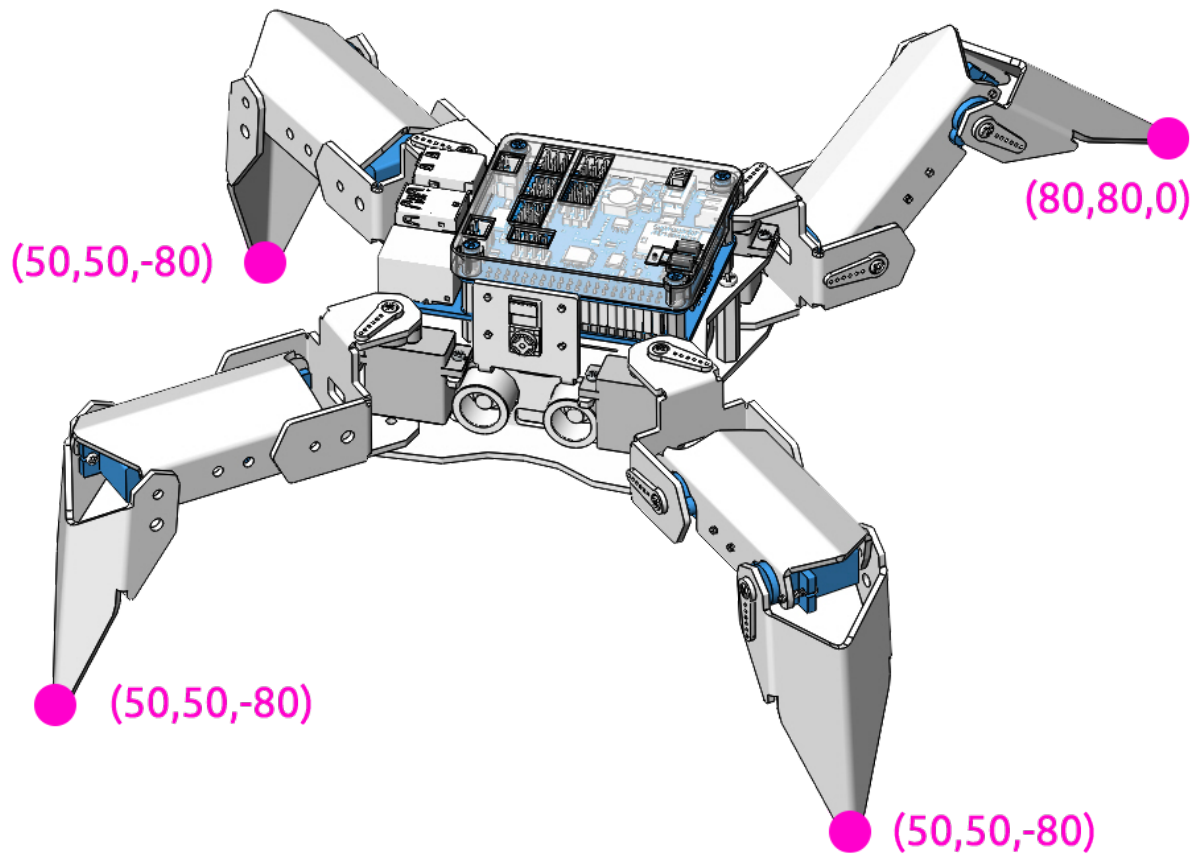
In general, this project combines the knowledge points of *Keyboard Control*, *Computer Vision* and *Sound Effect*.

Its flow is shown in the figure below:



3.10 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:

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

(continues on next page)

(continued from previous page)

```

    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.

```

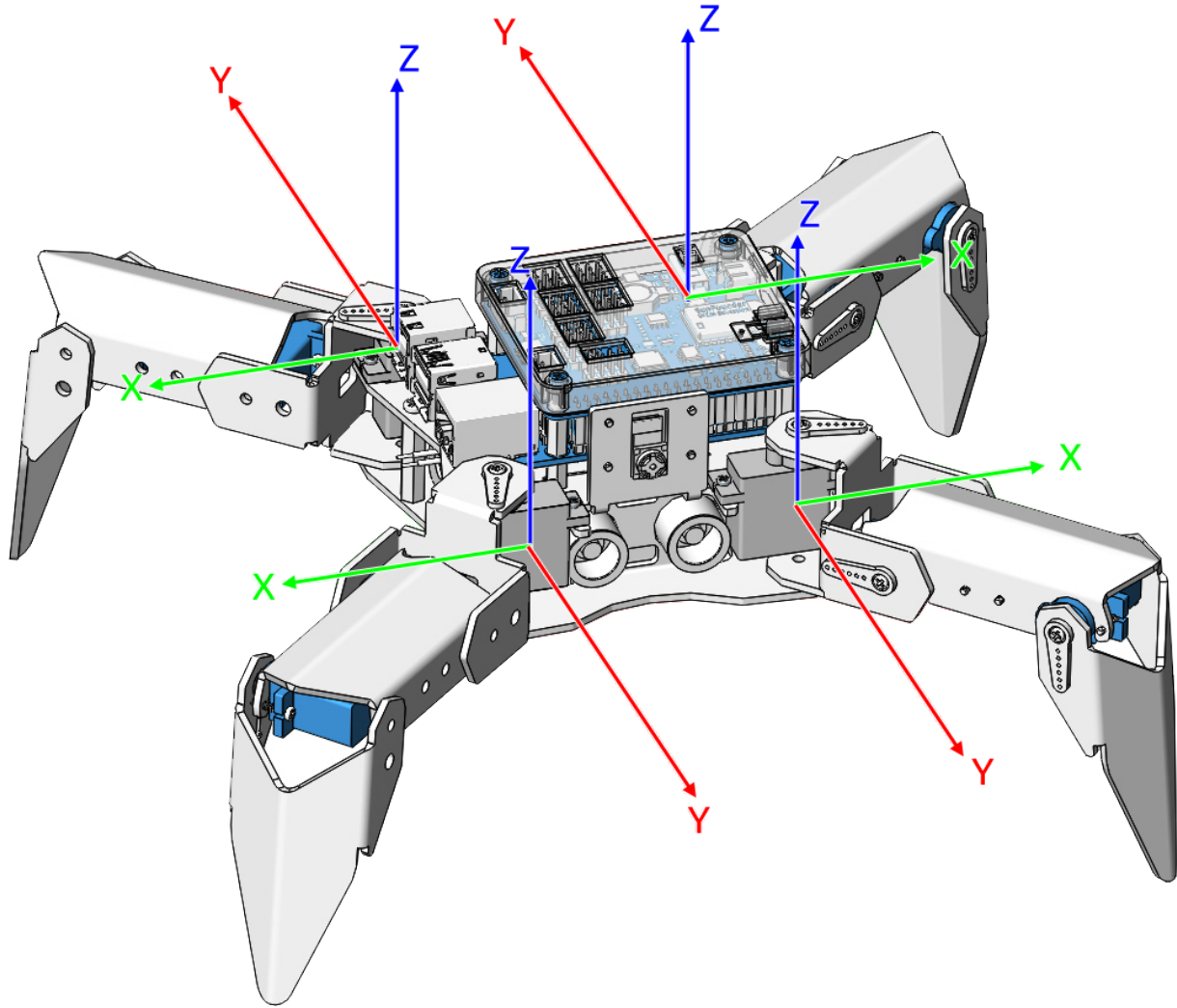
Second: It can also write an array of 4 coordinate values.

```

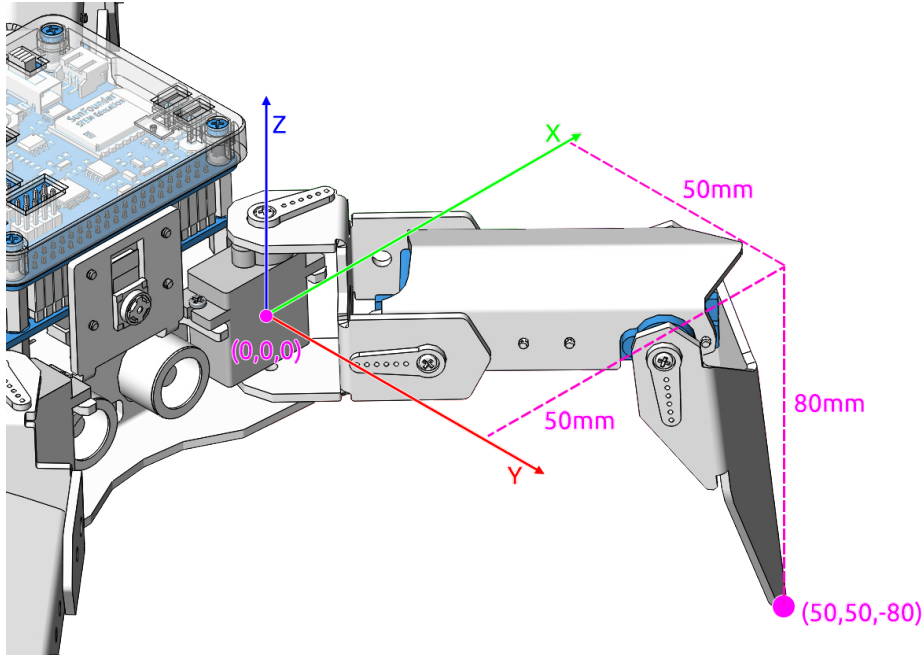
crawler.do_step([[50, 50, -80], [50, 50, -80], [80, 80, 0], [50, 50, -80]], speed)
# These four coordinates are used to control the four legs of right front, left front,
  ↳ left rear, and left rear respectively.

```

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



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



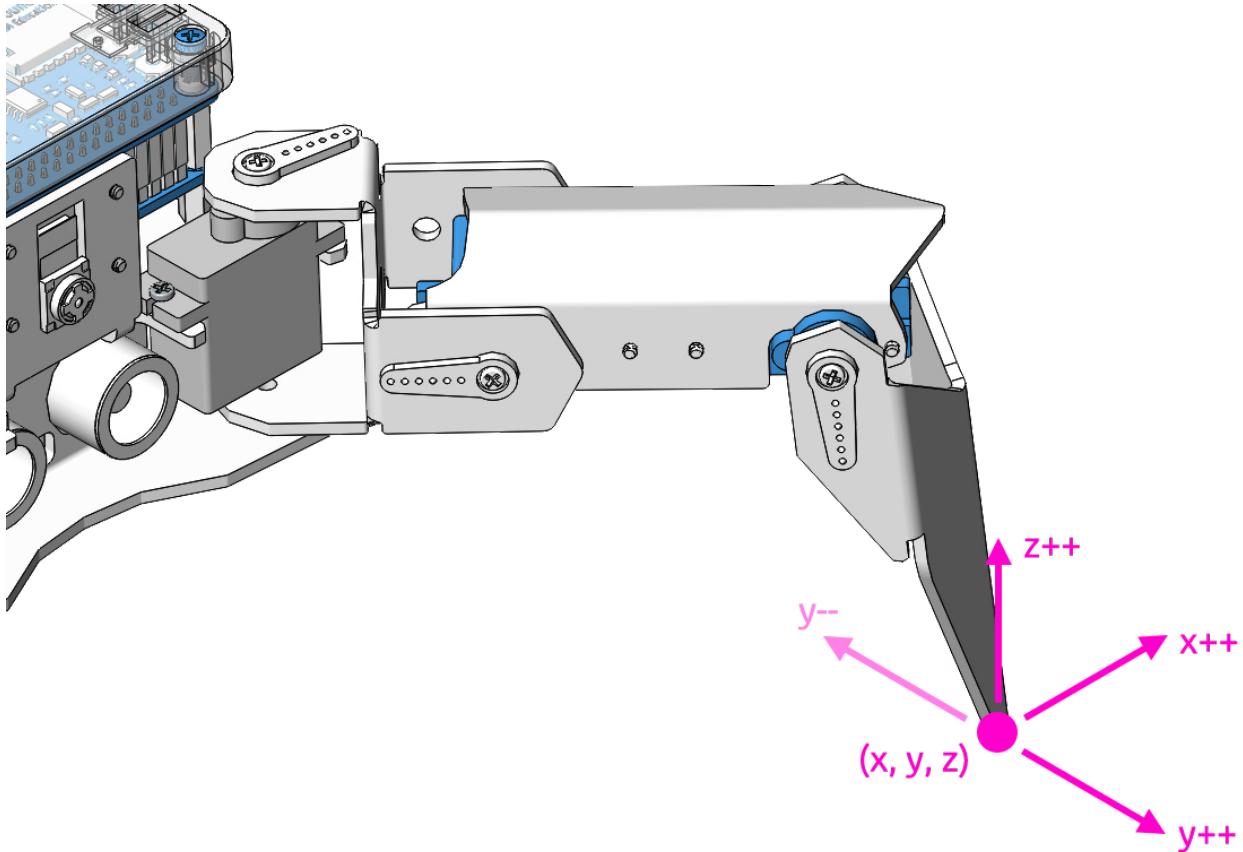
By the way: the `step_list` called in the first method also consists of an array containing 4 coordinate values.

```
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]
  ],
  1,
}
```

3.11 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.

Code

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

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

(continues on next page)

(continued from previous page)

```

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 coordinate
ESC: Quit
'''

def main():

    speed = 80
    print(manual)
    crawler.do_step('stand', speed)
    leg = 0
    coordinate=crawler.current_step_leg_value(leg)
    while True:
        key = readchar()
        print(key)
        if 'w' == key:
            coordinate[1]=coordinate[1]+5
        elif 's' == key:
            coordinate[1]=coordinate[1]-5
        elif 'a' == key:
            coordinate[0]=coordinate[0]-5
        elif 'd' == key:
            coordinate[0]=coordinate[0]+5
        elif 'r' == key:
            coordinate[2]=coordinate[2]+5
        elif 'f' == key:
            coordinate[2]=coordinate[2]-5
        elif '1' == key:
            leg=0
            coordinate=crawler.current_step_leg_value(leg)
        elif '2' == key:
            leg=1
            coordinate=crawler.current_step_leg_value(leg)
        elif '3' == key:
            leg=2
            coordinate=crawler.current_step_leg_value(leg)
        elif '4' == key:
            leg=3
            coordinate=crawler.current_step_leg_value(leg)
        elif chr(32) == key:
            print("[[right front],[left front],[left rear],[right rear]]")
            print(crawler.current_step_all_leg_value())

        elif chr(27) == key: # 27 for ESC
            break

```

(continues on next page)

(continued from previous page)

```

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

if __name__ == "__main__":
    main()

```

How it works?

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

```

current_step_leg_value(leg)
current_step_all_leg_value()
do_single_leg(leg, coordinate, speed)

```

- `current_step_leg_value(leg)` : Returns the coordinate value of the corresponding leg. The parameter `leg` can be 0, 1, 2, 3 four values, corresponding to right front, left front, left rear, left rear four values respectively leg.
- `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.

3.12 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

```

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

```

(continues on next page)

(continued from previous page)

```

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 coordinate & 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
    coordinate=crawler.current_step_leg_value(leg)
    while True:
        key = readchar()
        print(key)
        if 'w' == key:
            coordinate[1]=coordinate[1]+2
        elif 's' == key:
            coordinate[1]=coordinate[1]-2
        elif 'a' == key:
            coordinate[0]=coordinate[0]-2
        elif 'd' == key:
            coordinate[0]=coordinate[0]+2
        elif 'r' == key:
            coordinate[2]=coordinate[2]+2
        elif 'f' == key:
            coordinate[2]=coordinate[2]-2
        elif '1' == key:
            leg=0
            coordinate=crawler.current_step_leg_value(leg)
        elif '2' == key:
            leg=1

```

(continues on next page)

(continued from previous page)

```

        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)

```


3.13 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.

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](#).

3.14 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()
```

PLAY WITH EZBLOCK

Ezblock is a development platform developed by SunFounder designed for beginners to lower the barriers to getting started with Raspberry Pi. It has two programming languages: Graphical and Python, and available on almost all different types of devices. With Bluetooth and Wi-Fi support, you can download code, remote control a Raspberry Pi, on Ezblock Studio.

Ezblock has been upgraded to the third version. Compared with the Bluetooth communication of Ezblock 2, Ezblock 3 uses Websocket to communicate, which is much faster.

So Ezblock 3 is used here.

4.1 Quick Guide on Ezblock

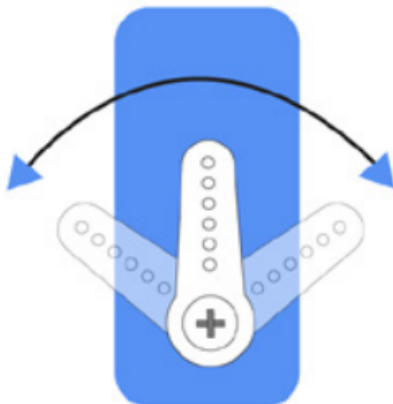
There are 2 parts here:

- *Before Assembling PiCrawler* allows you to keep all the servos at 0 degrees to complete a proper and safe assembly (otherwise you will probably damage the servos).
- *Before Programming With Ezblock* will guide you to download Ezblock Studio to play with PiCrawler.

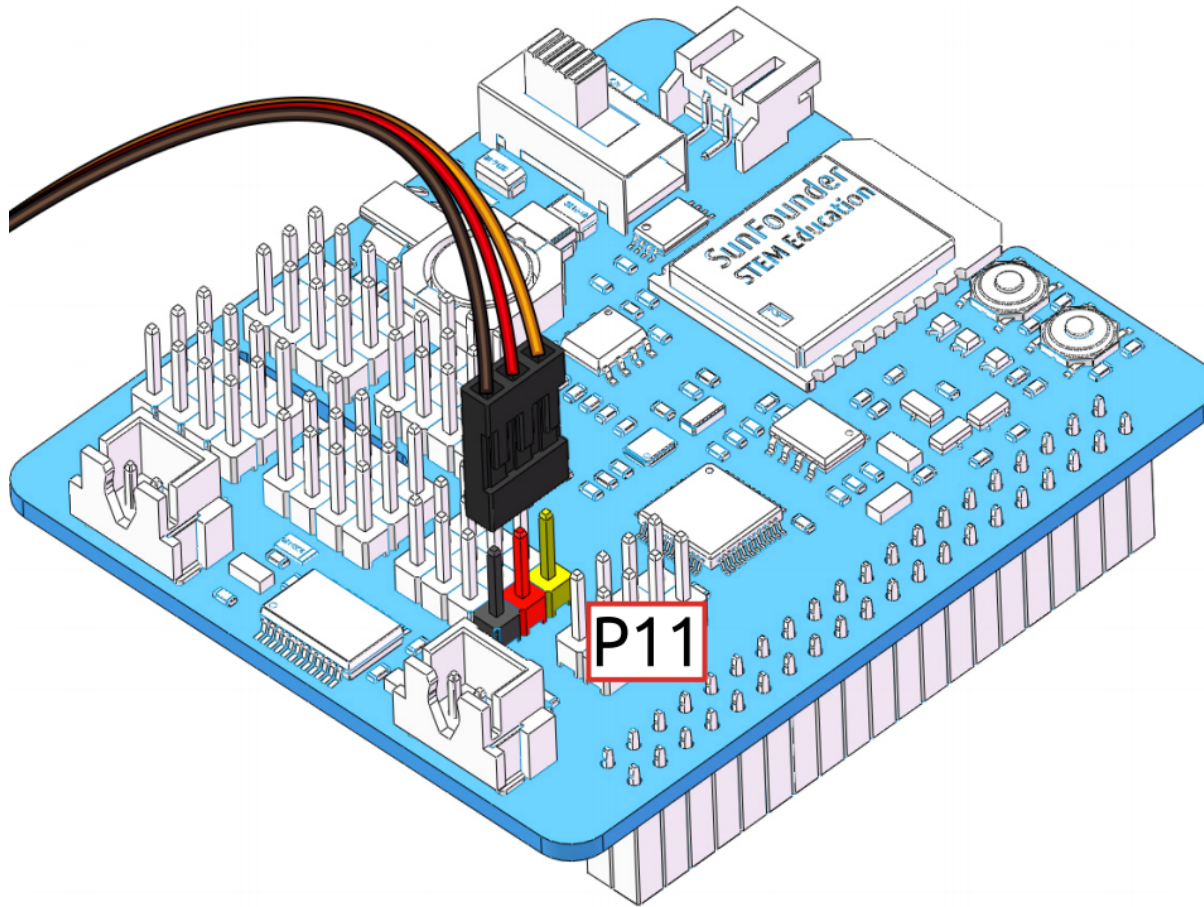
4.1.1 Before Assembling PiCrawler

Before assembling the PiCrawler, follow the instructions on how to install the Ezblock OS on an Micro SD card here: [Download and Install Ezblock OS](#).

After burning the Ezblock system on the SD-card, the P11 port on the Robot HAT is set to calibrate the servo angle to a 0° angle. To make sure the servo has been correctly set to 0°, first gently insert a rocker arm in the servo shaft, then slightly rotate the rocker arm to a different angle.



Next, insert the servo cable into the P11 port as shown below:



Turn on the **Power Switch** to the Robot HAT, and the servo arm should return to the 0° position. If the servo arm does not return to 0°, press the **RST** button to restart the Robot HAT.

Note: Before attaching each servo, plug the servo cable into P11 and turn the power on to set the servo to 0°. This function will become invalid after writing any programs to the Micro SD card.

4.1.2 Before Programming With Ezblock

First download Ezblock Studio 2, and then manually upgrade to Ezblock Studio 3 to begin programming.

For a detailed installation tutorial, please refer to: [Install Ezblock Studio](#).

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.

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.

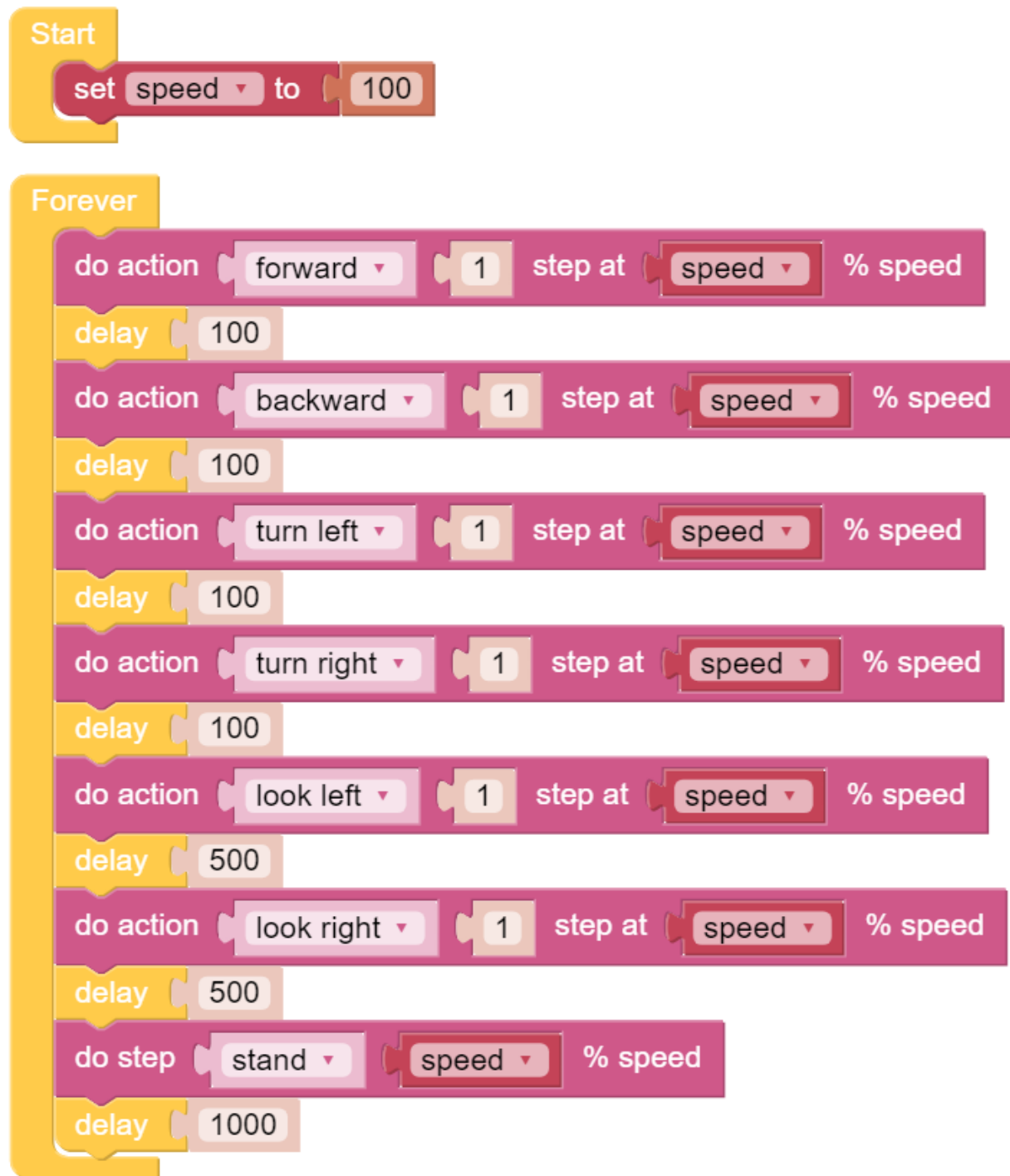
4.2 Move — Ezblock

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

Note: If you are not yet familiar with the use of Ezblock, please refer to the [Quick User Guide](#). You need to know **How to connect the product and Ezblock Studio?** and **How to Open and Run examples?** to proceed with this project smoothly.

Program

After opening the example, you can see the following code block.



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

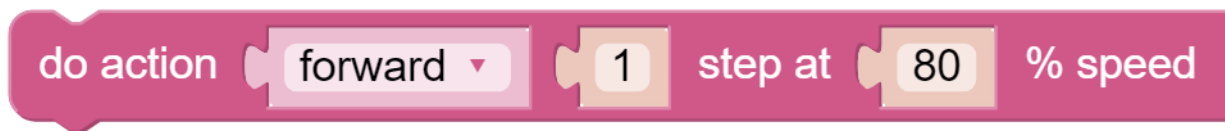
How it works?

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.

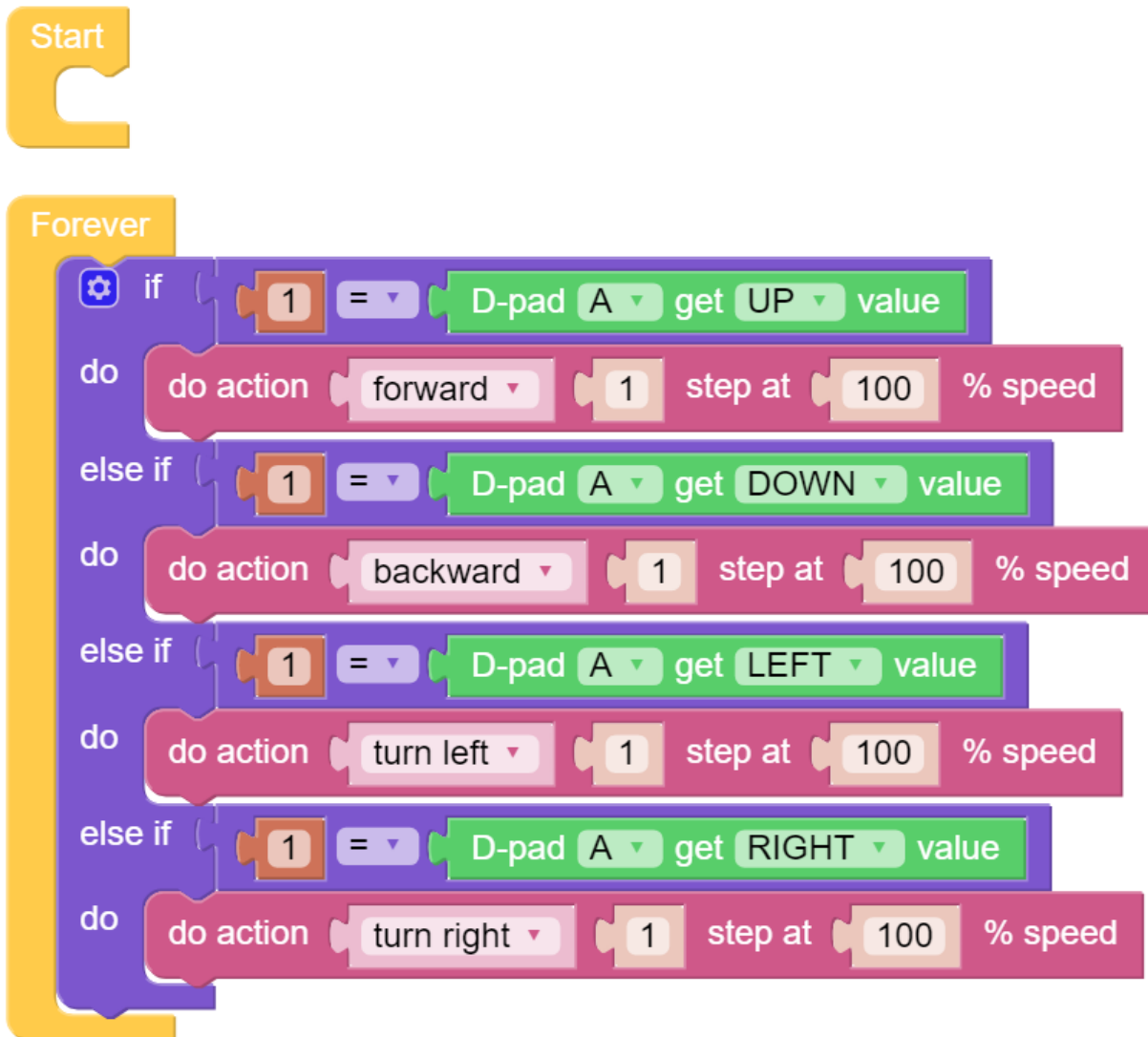
4.3 Remote Control — Ezblock

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

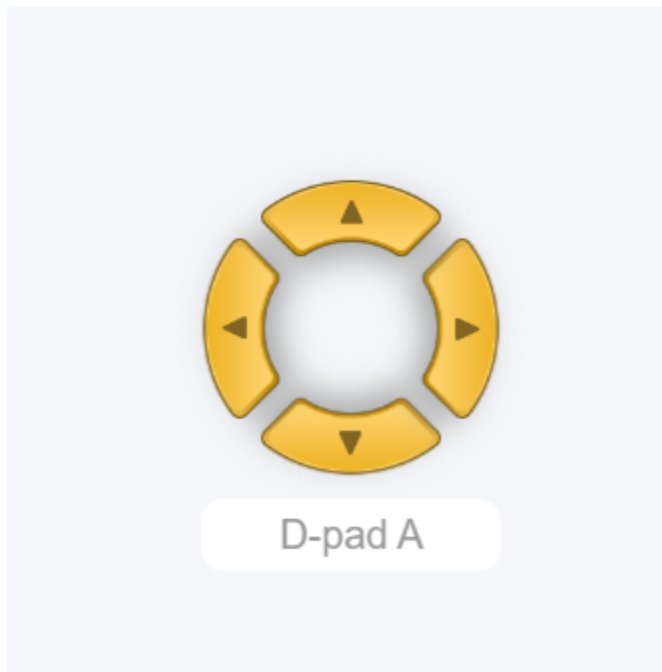
Note: You can refer to [How to Use the Remote Control Function?](#). Come and carry out this project smoothly.

Program

After opening the example, you can see the following code block.



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

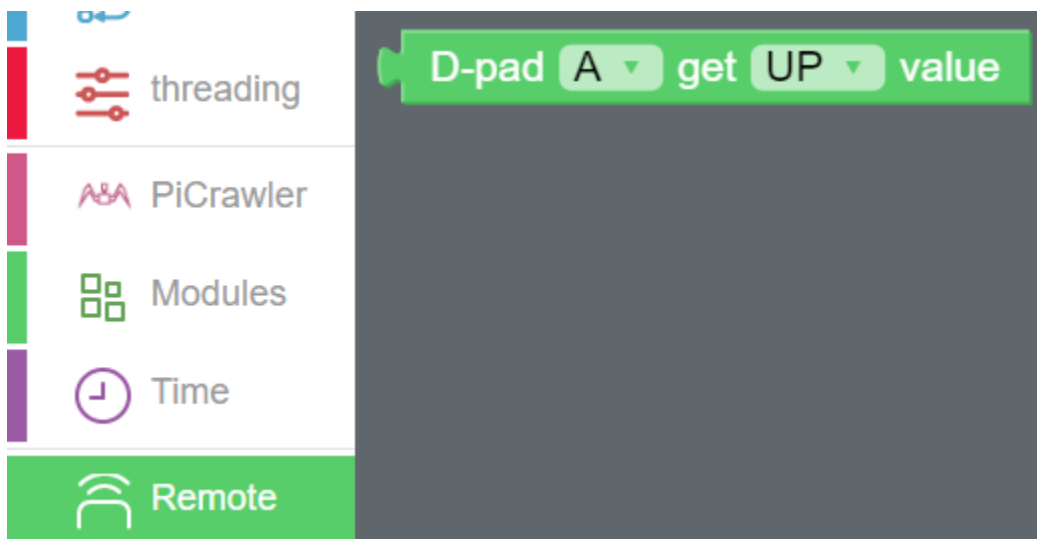


After the program is running, you can activate PiCrawler through D-Pad.

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.

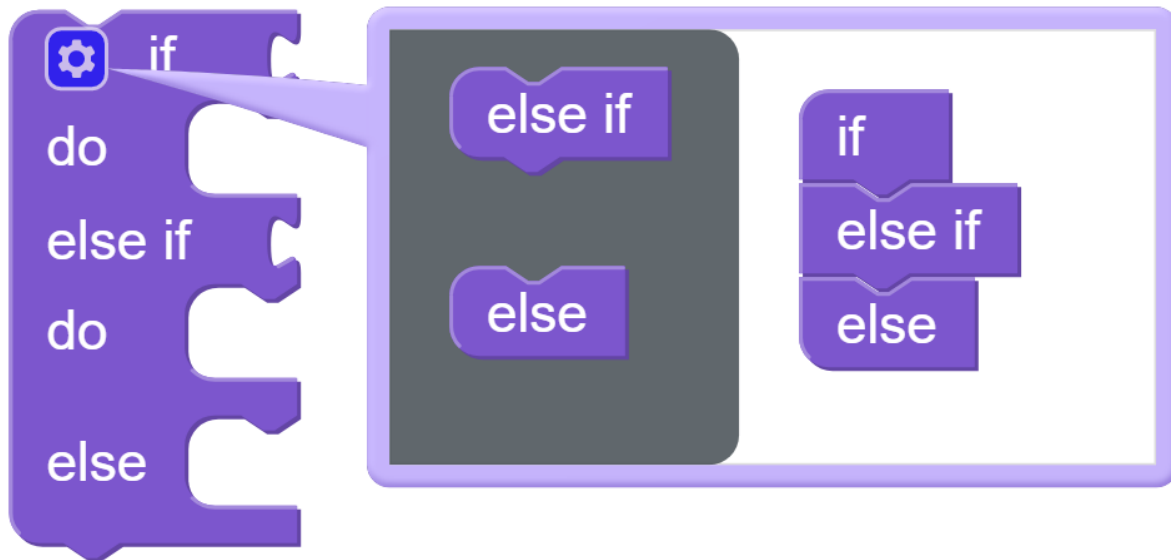
Here we add the D-Pad widget, so the **D-Pad get value** block appears here.



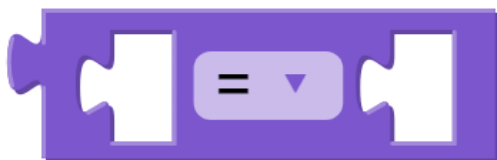
D-Pad can be regarded as a four-in-one button. You can choose which button to read in the second groove of the block. When the button is pressed, the value is “1”; when the button is not pressed, the value is “0”.



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.



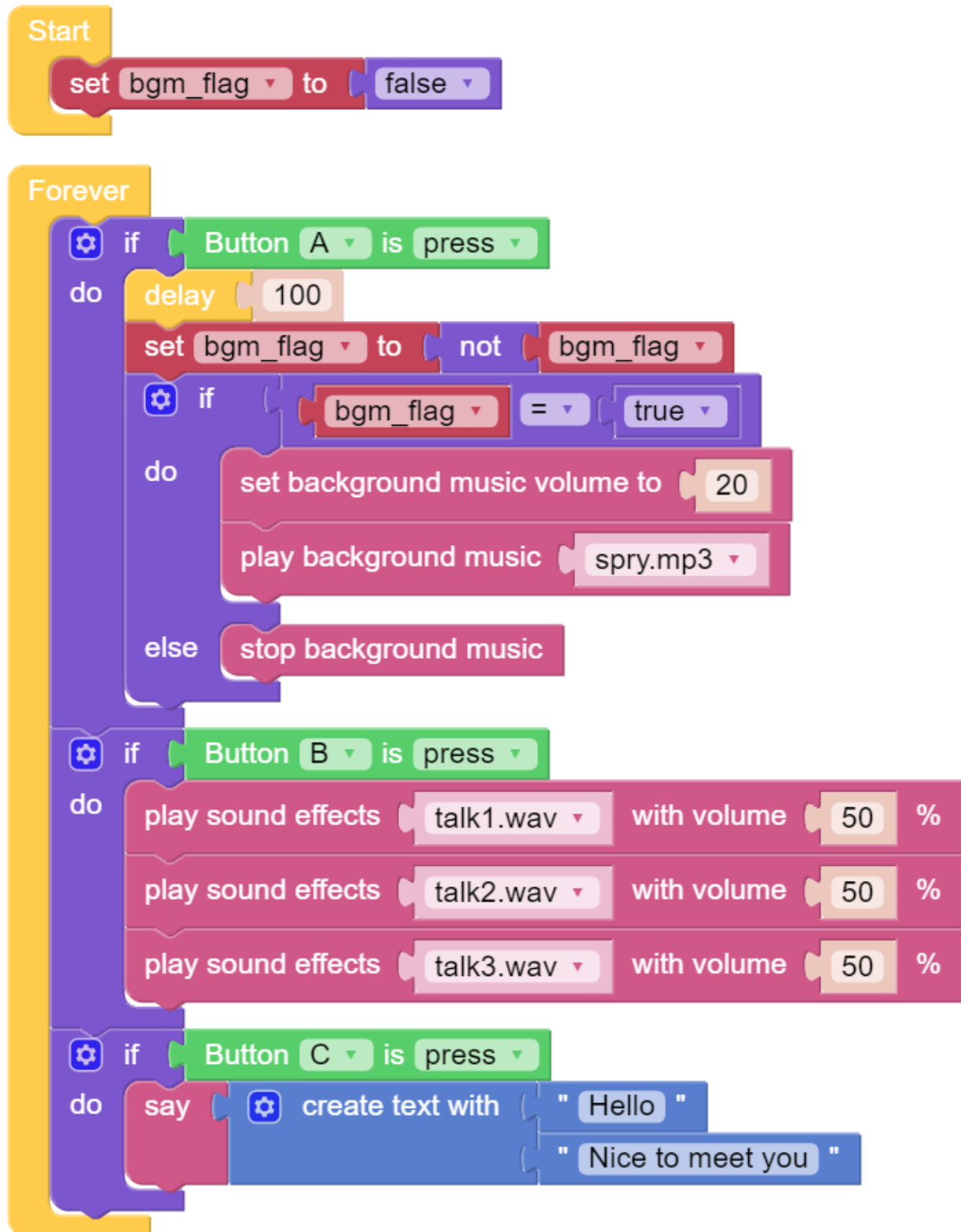
if block is usually used with = block, = block can be modified to >, < and other conditions through the drop-down menu, please use it flexibly.

4.4 Sound Effect — Ezblock

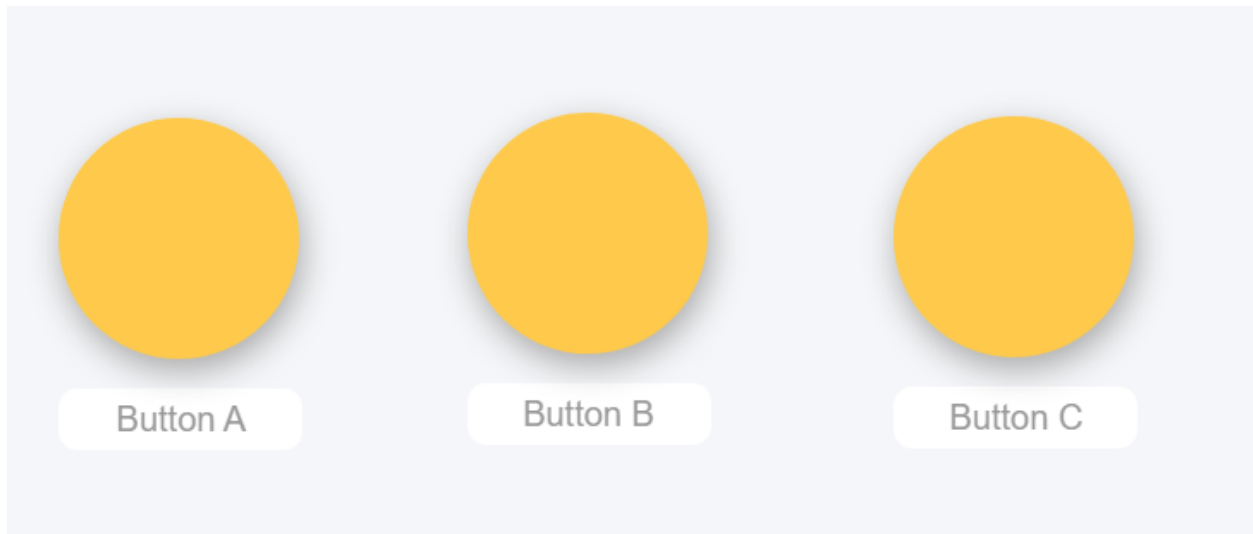
In this example, we use PiCrawler's (to be precise, Robot HAT's) sound effects. It consists of three parts, namely **Muisc**, **Sound**, **Text to Speech**.

Program

After opening the example, you can see the following code block.



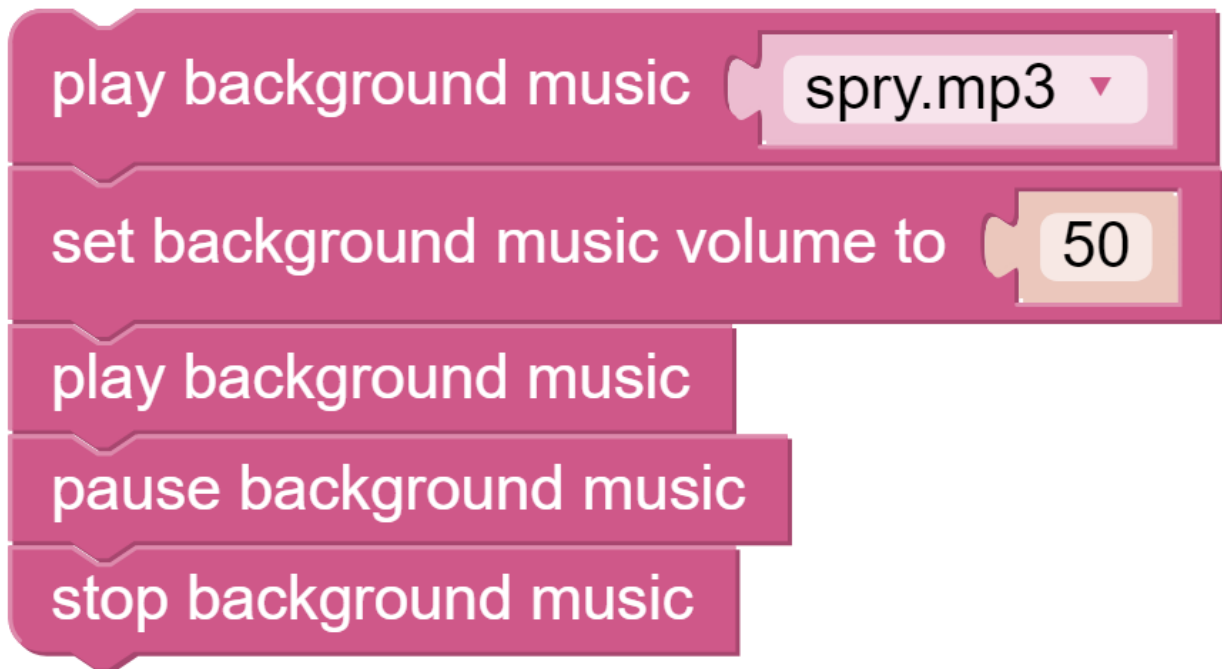
Switch to the Remote Control interface, and you will see the following widgets.



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

How it works?

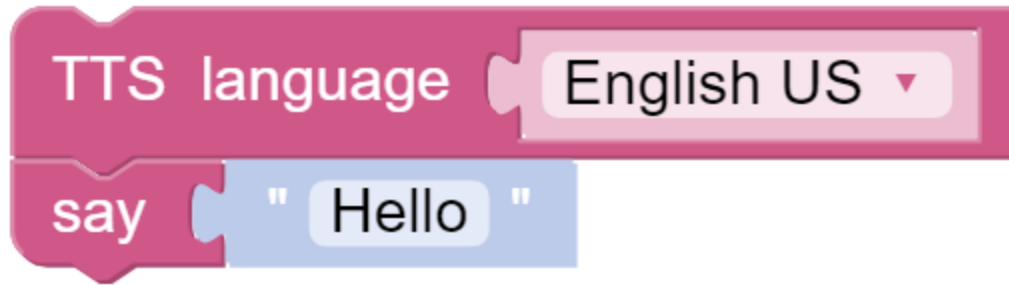
Functions related to background music include these:



Functions related to sound effects include these:



Functions related to Text to Speech include these:

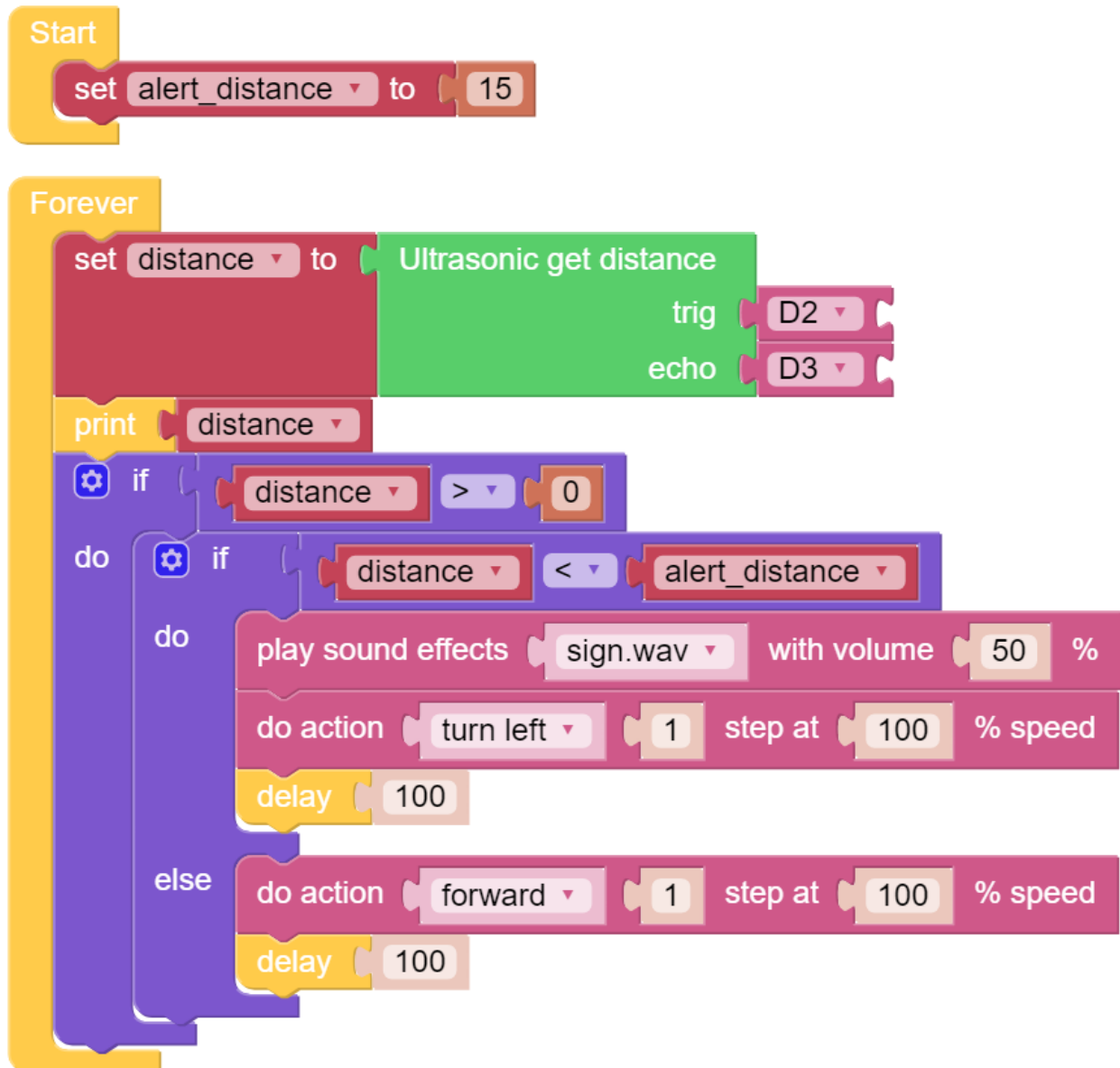


4.5 Obstacle Avoidance — Ezblock

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.

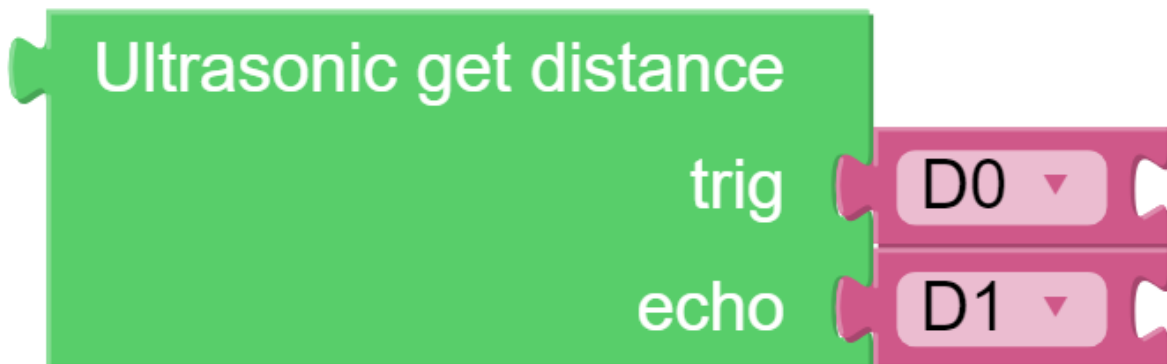
Program

After opening the example, you can see the following code block.



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, $\text{distance} < 0$ 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.

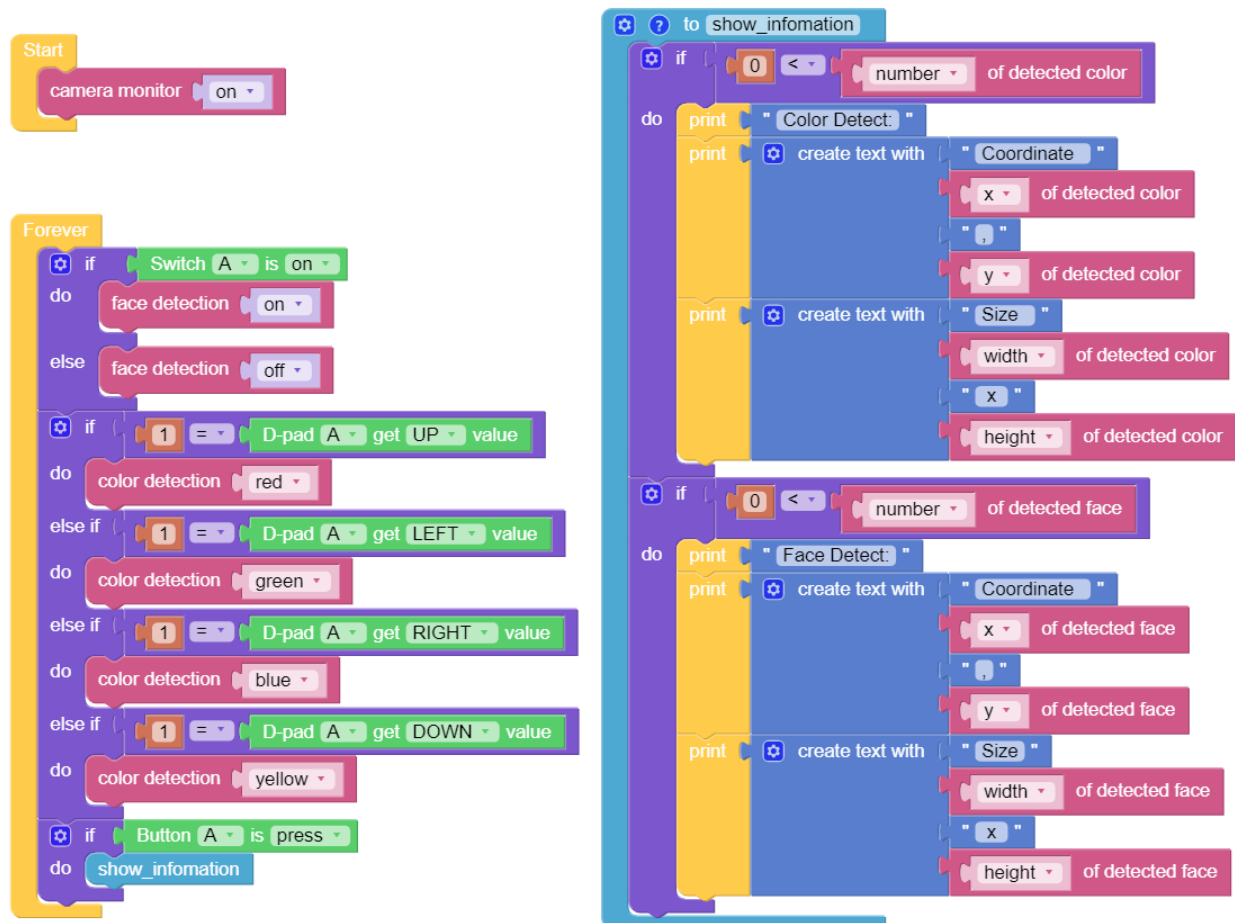
4.6 Computer Vision — Ezblock

This project will officially enter the field of computer vision!

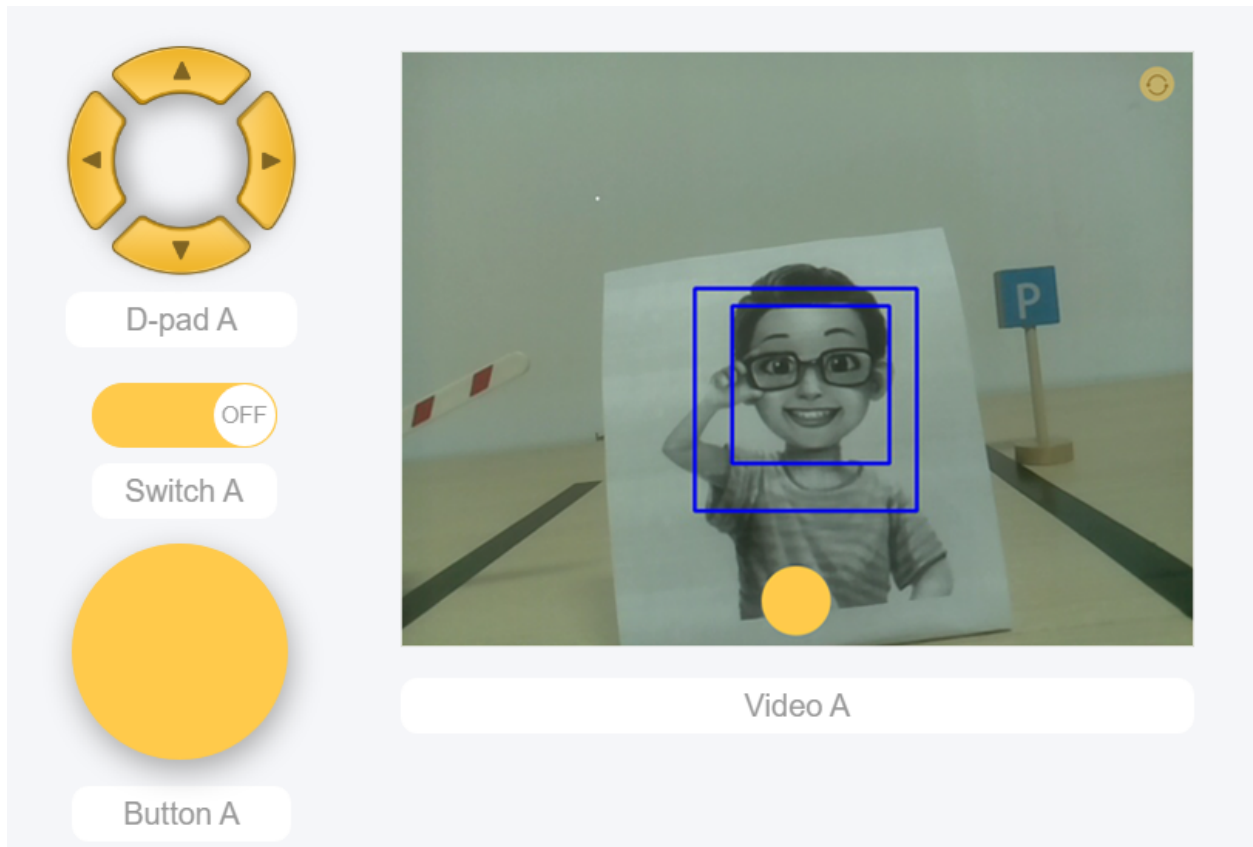
Note: You can read [How to Use the Video Function?](#). Come and carry out this project smoothly.

Program

After opening the example, you can see the following code block.



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

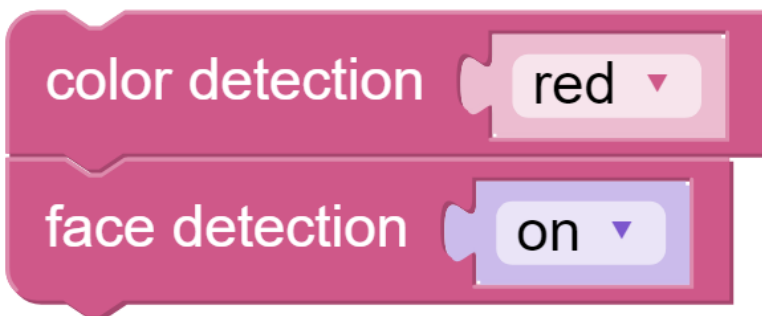


After the program is running, you can switch the slider widget to turn on/off the face detection; click the D-Pad to select the color of the detection; click the button to print the detection result.

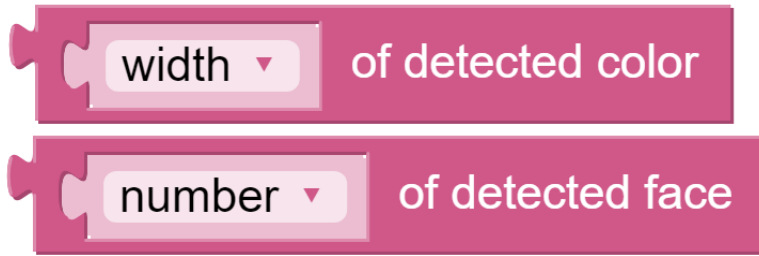
How it works?



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.

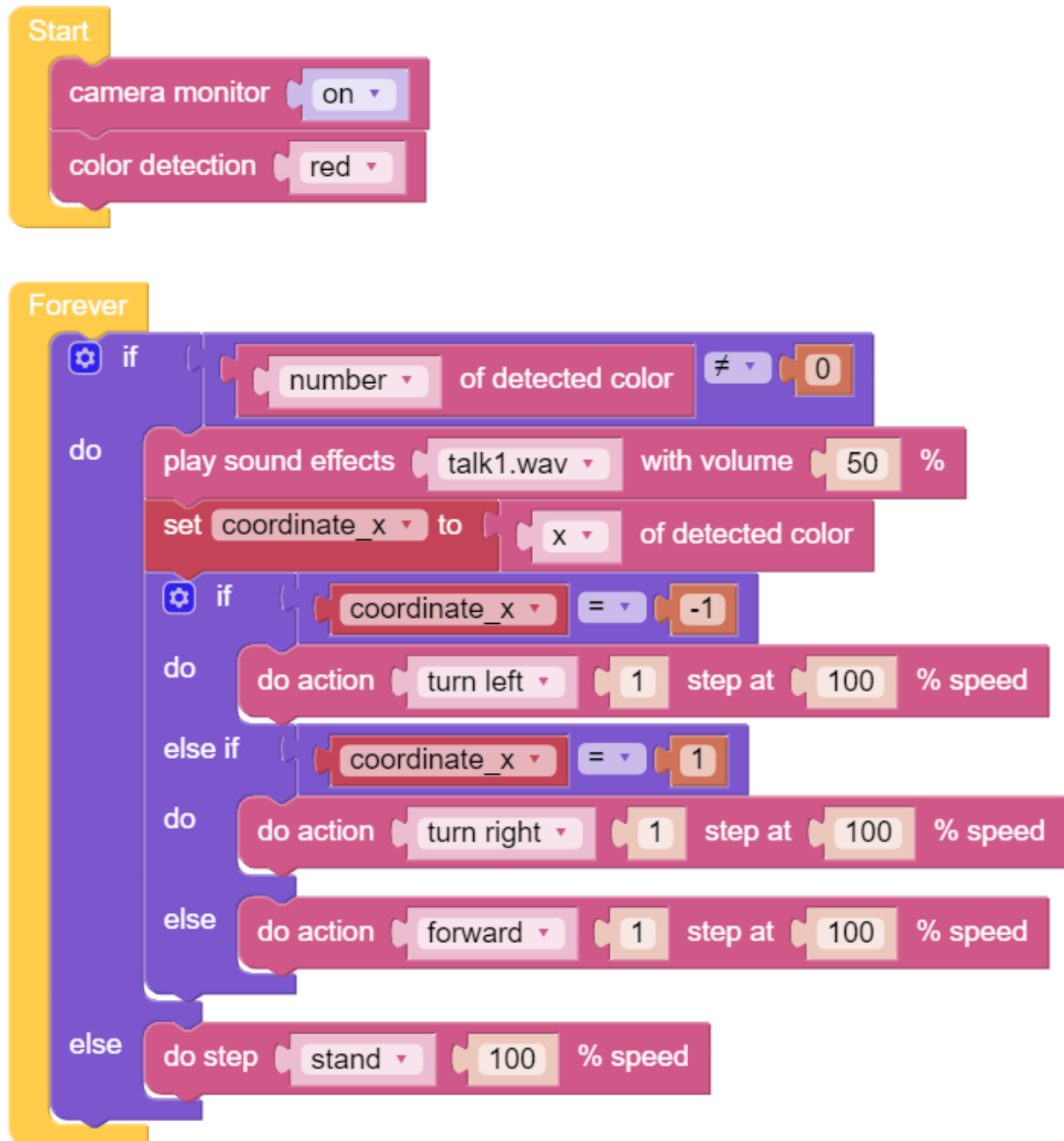
4.7 Bull Fight — Ezbloc

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

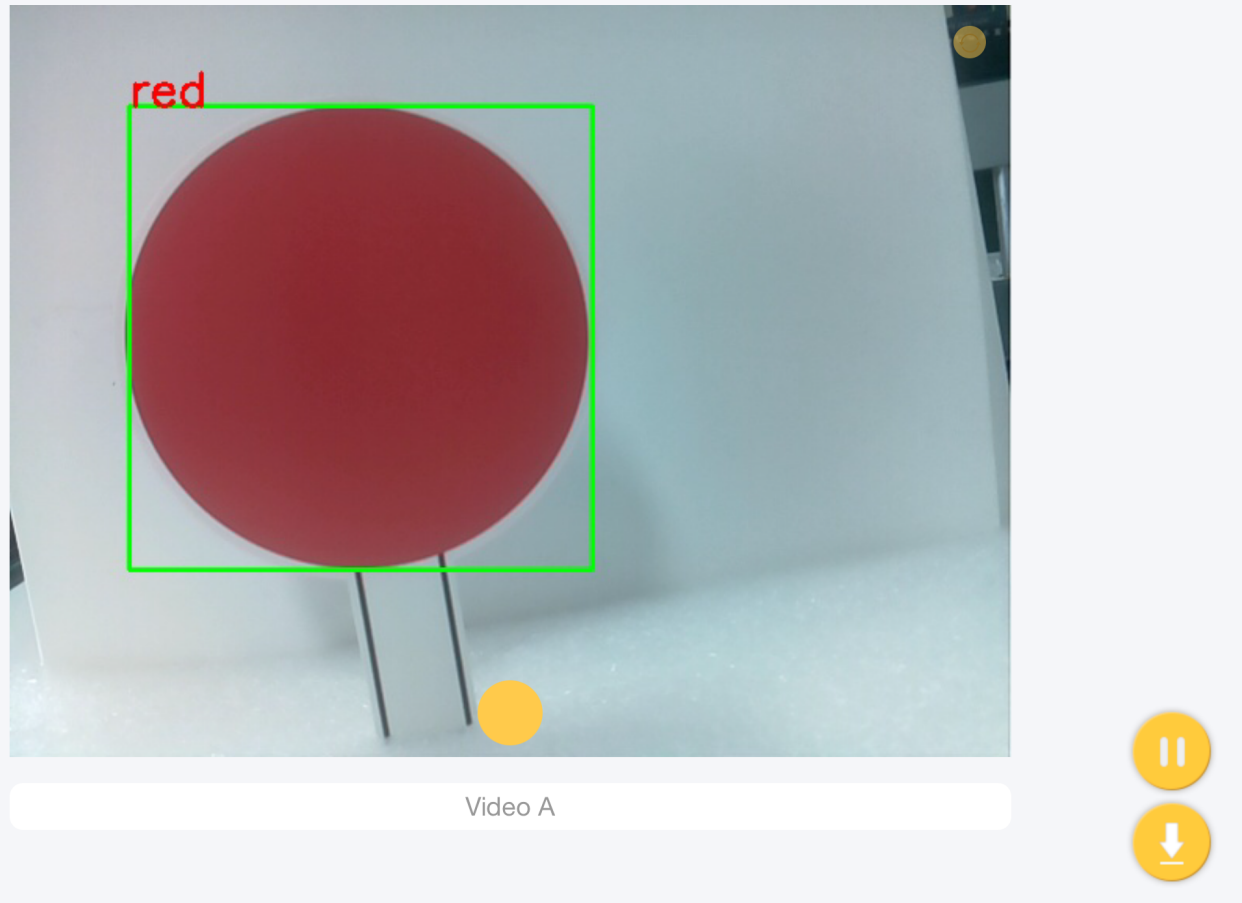
Note: You can download and print the [PDF Color Cards](#) for color detection.

Program

After opening the example, you can see the following code block.



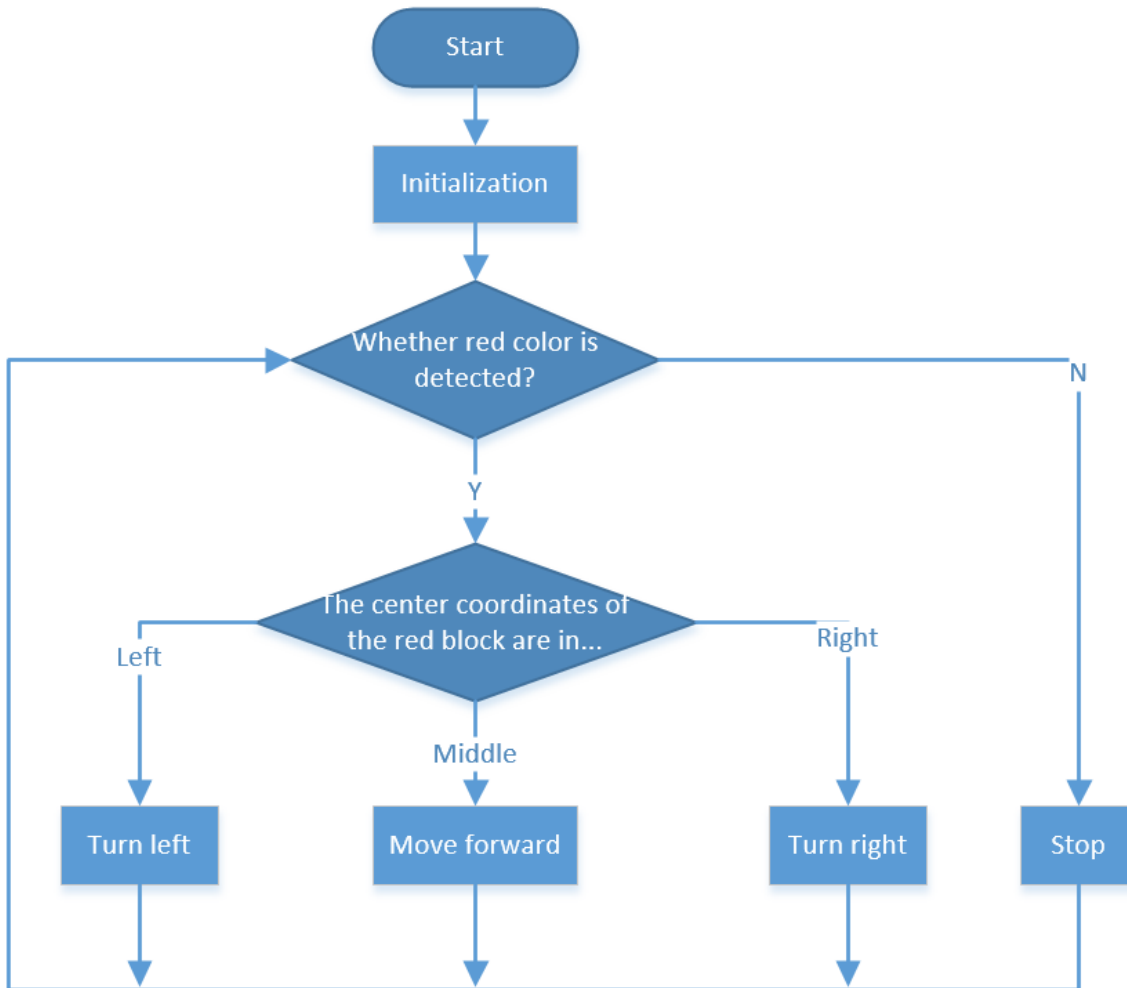
Switch to the Remote Control interface, you will see the following screen.



How it works?

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

Its flow is shown in the figure below:



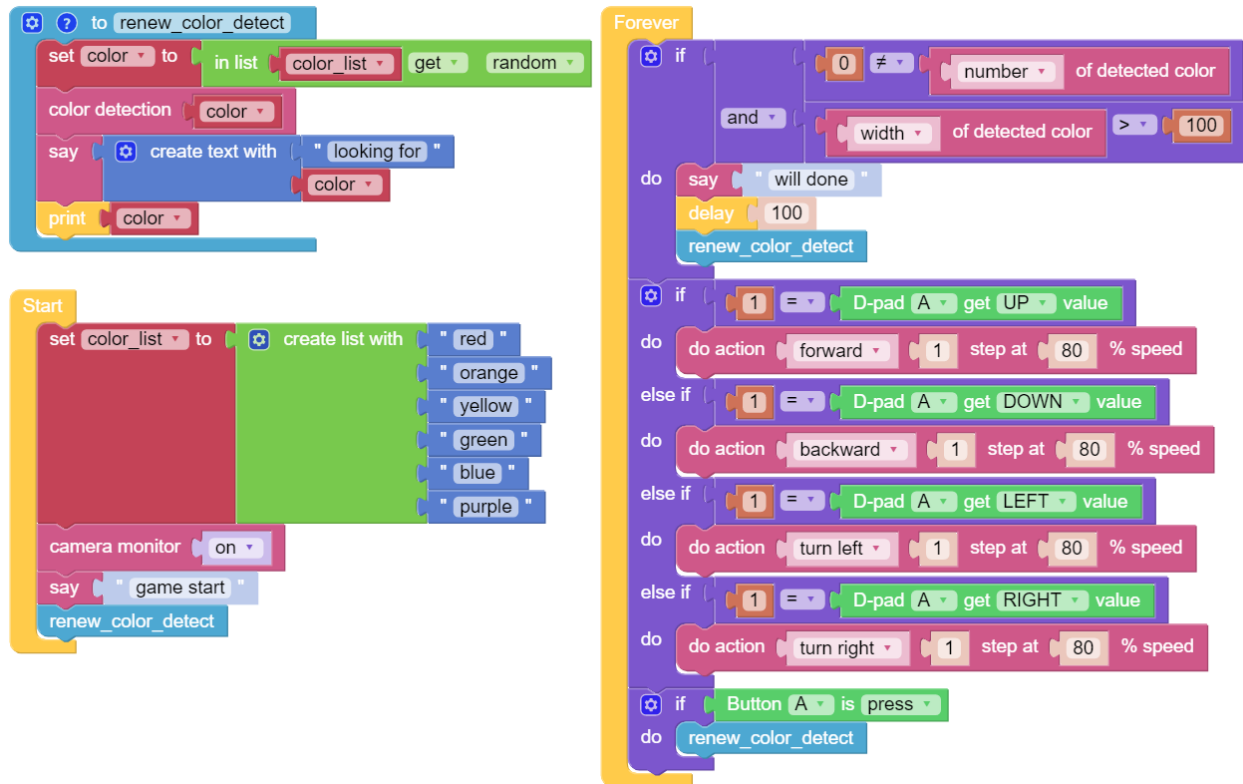
4.8 Treasure Hunt — Ezblock

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!

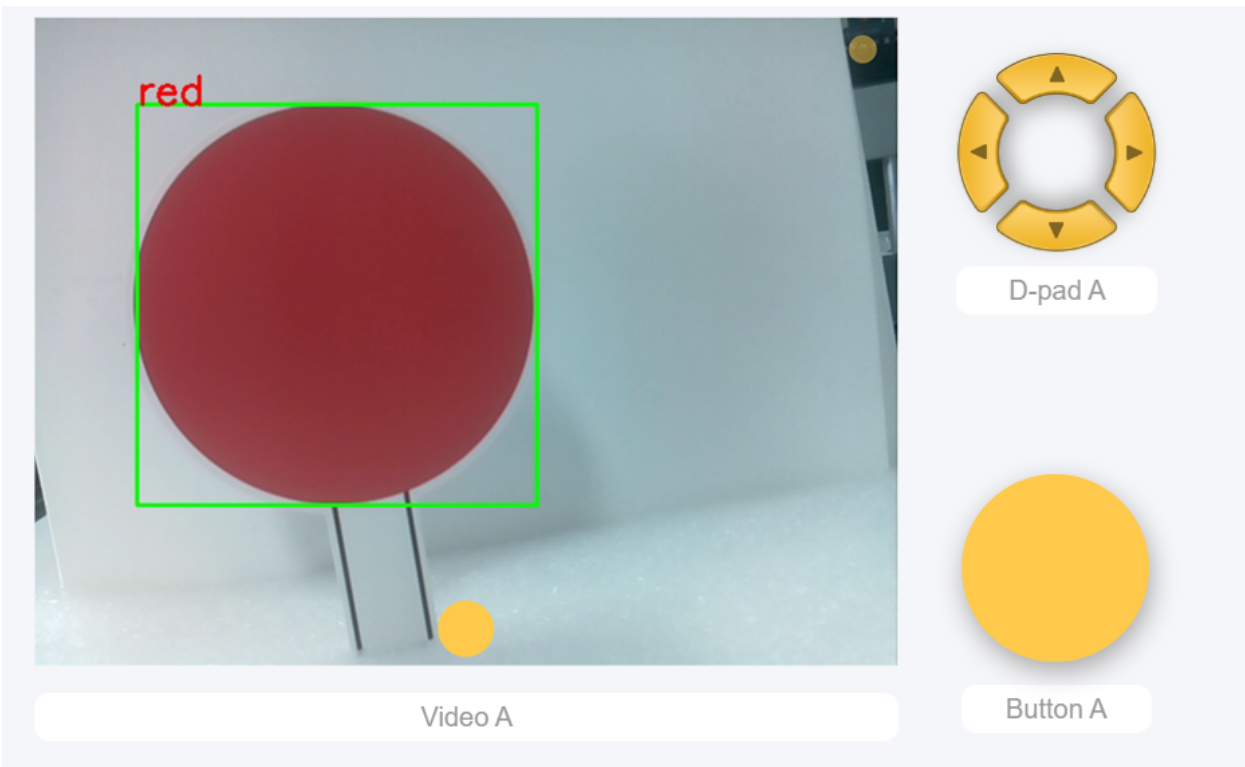
Note: You can download and print the PDF [Color Cards](#) for color detection.

Program

After opening the example, you can see the following code block.



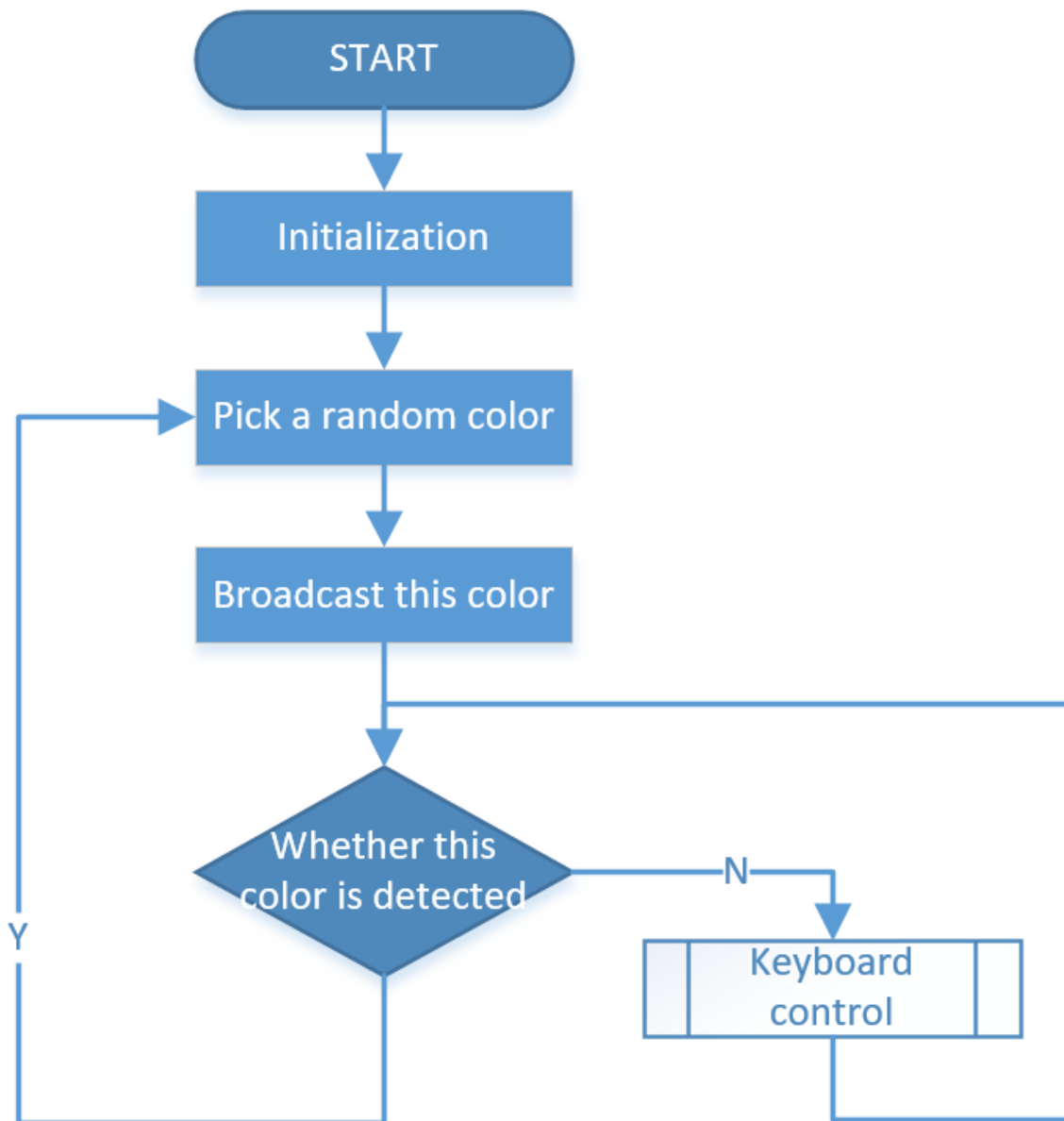
Switch to the Remote Control interface, and you will see the following widgets.



How it works?

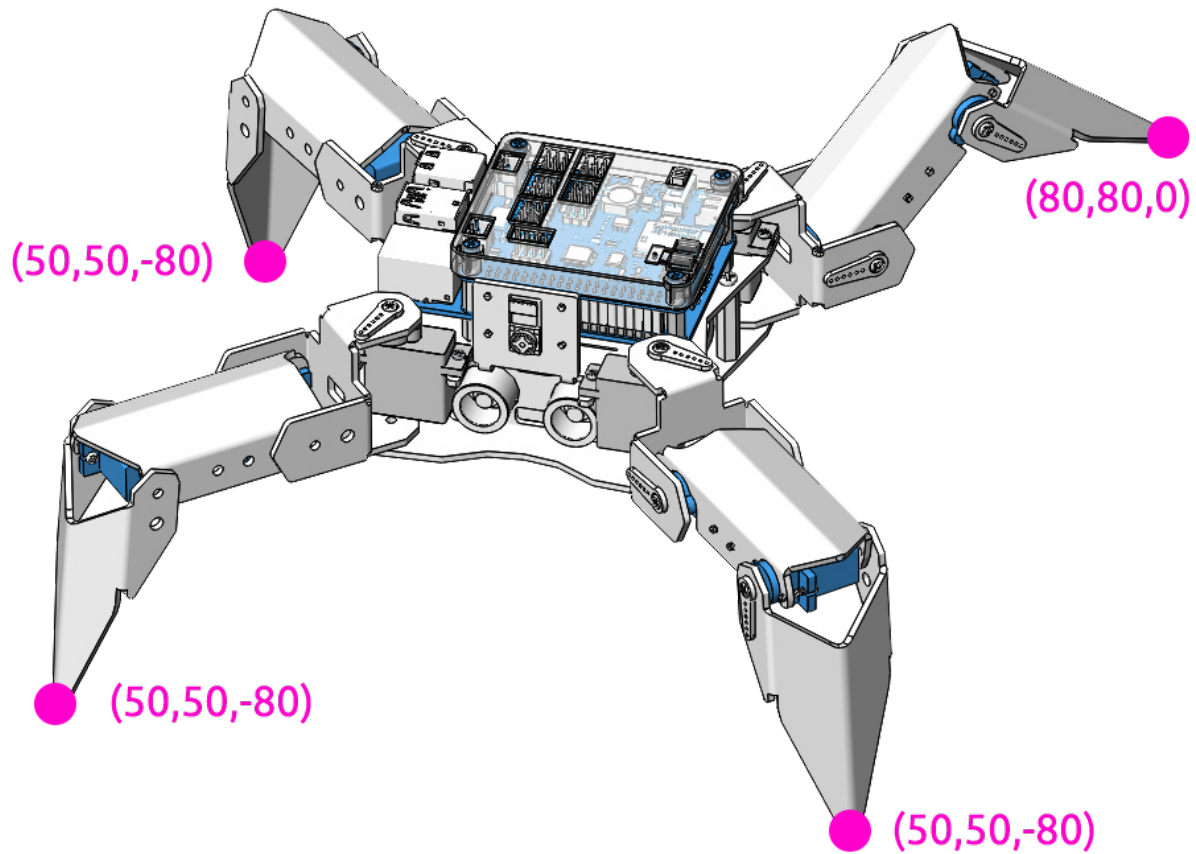
In general, this project combines the knowledge points of *Remote Control — Ezblock*, *Computer Vision — Ezblock* and *Sound Effect — Ezblock*.

Its flow is shown in the figure below:



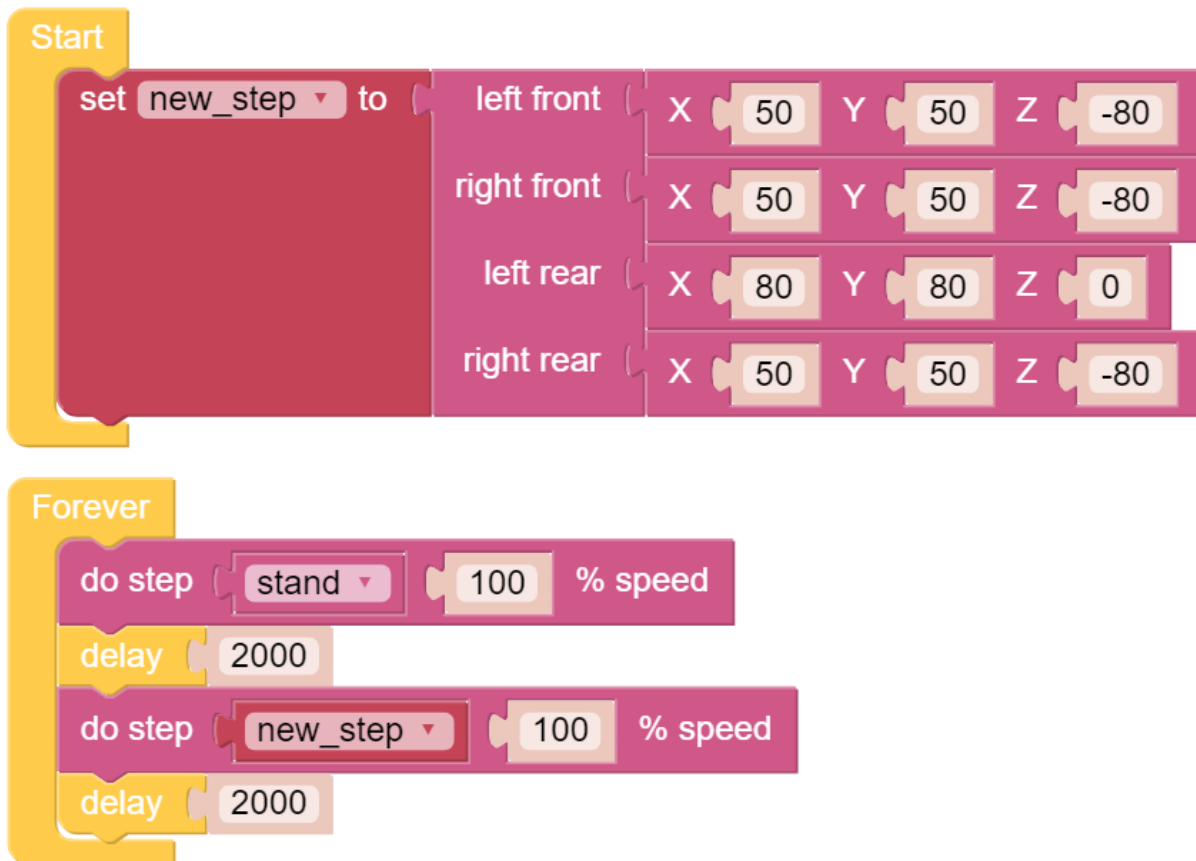
4.9 Pose — Ezblock

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



Program

After opening the example, you can see the following code block.



How it works?

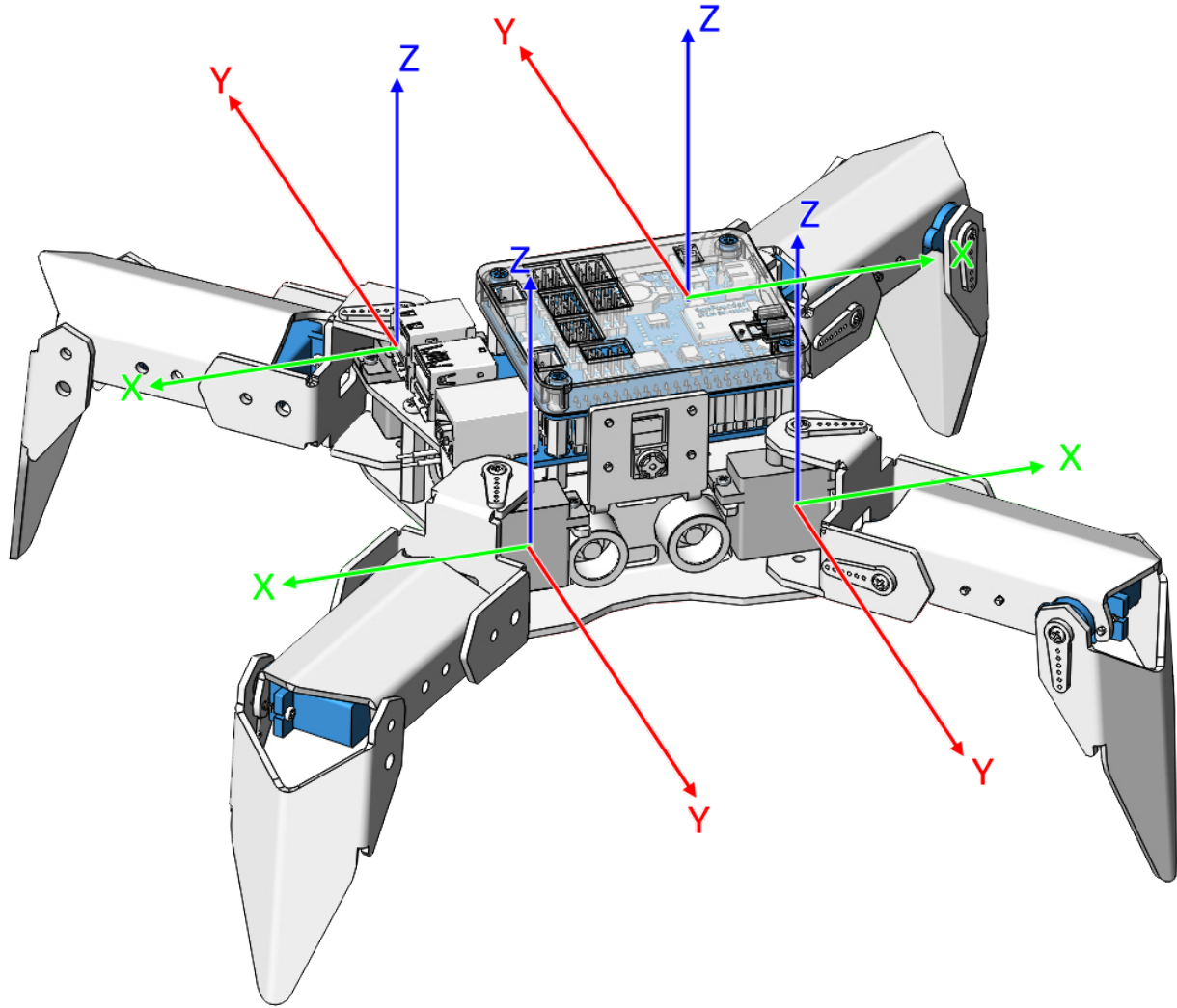
In this code, the code you need to pay attention to is this **do step**.

It has two uses:

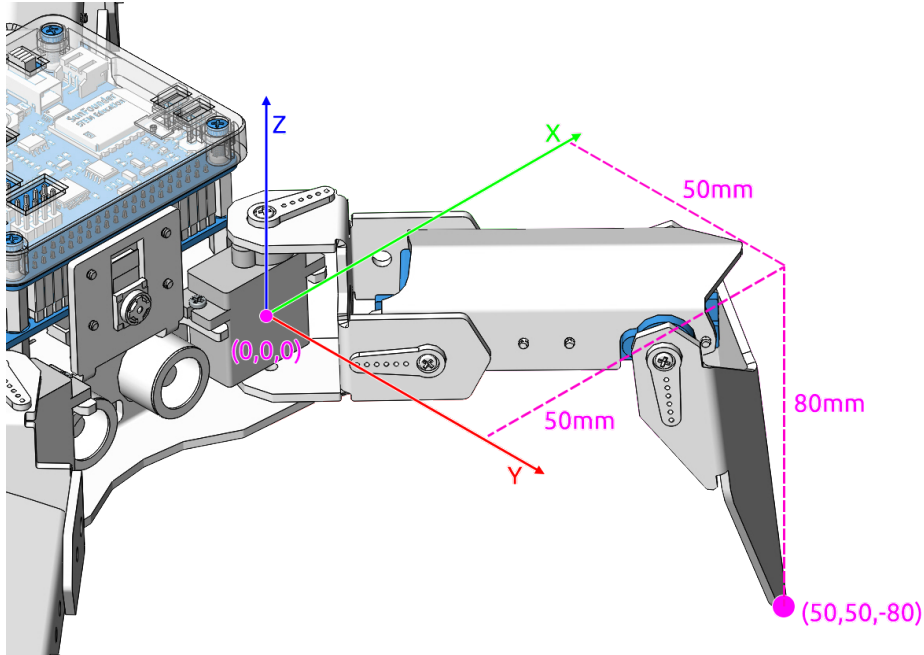
One: It can directly use **stand** or **sit**.

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

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



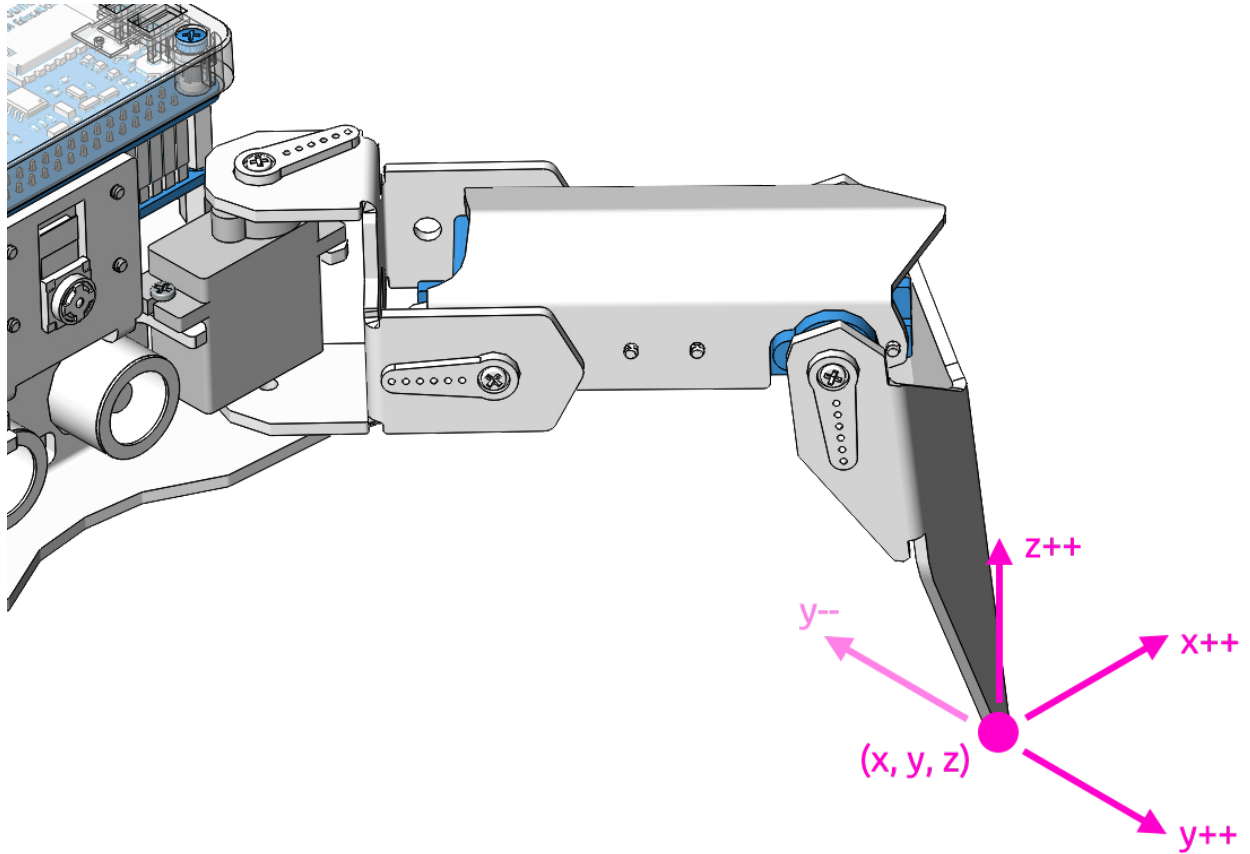
You need to measure the coordinates of each toe individually. As shown below:



4.10 Adjust Posture — Ezblock

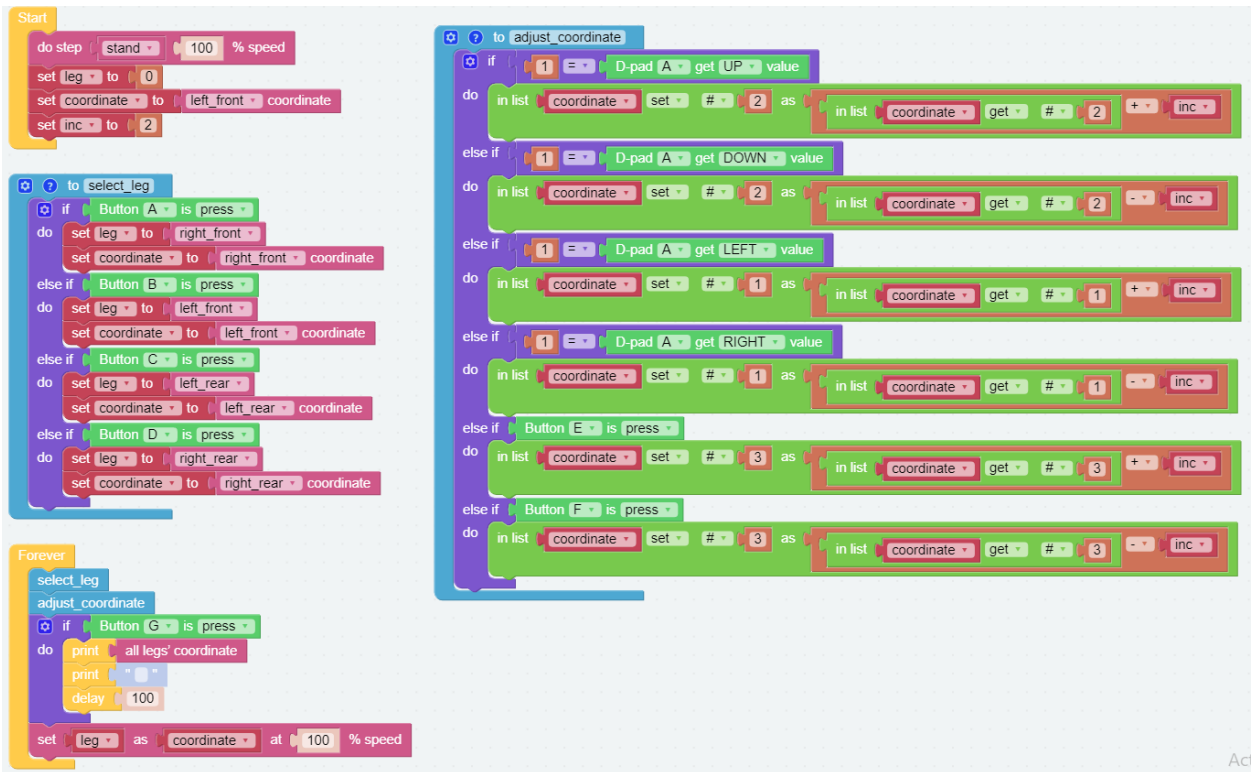
In this example, we use the remote function to control the PiCrawler foot by foot and assume the desired posture.

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.

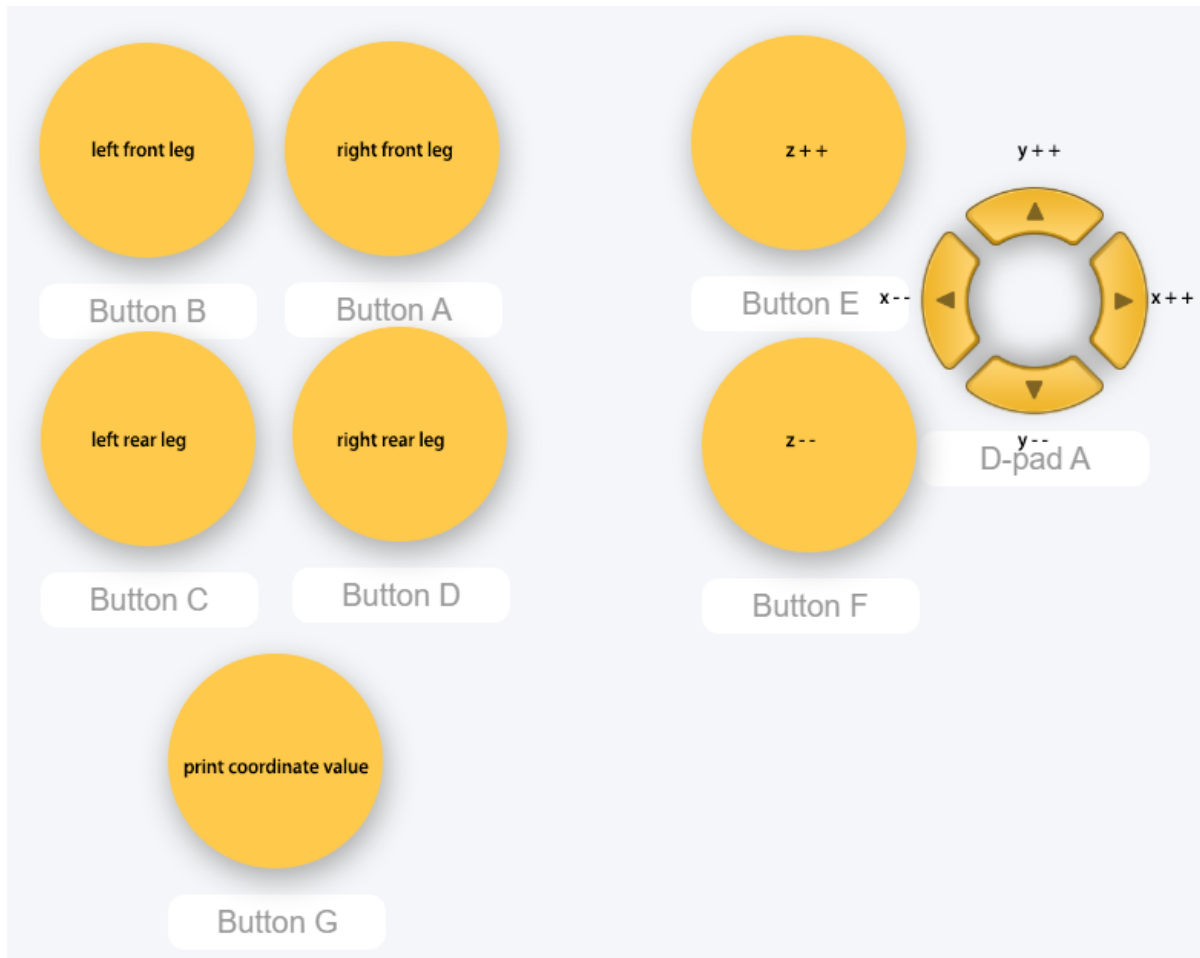


Program

After opening the example, you can see the following code block.



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:



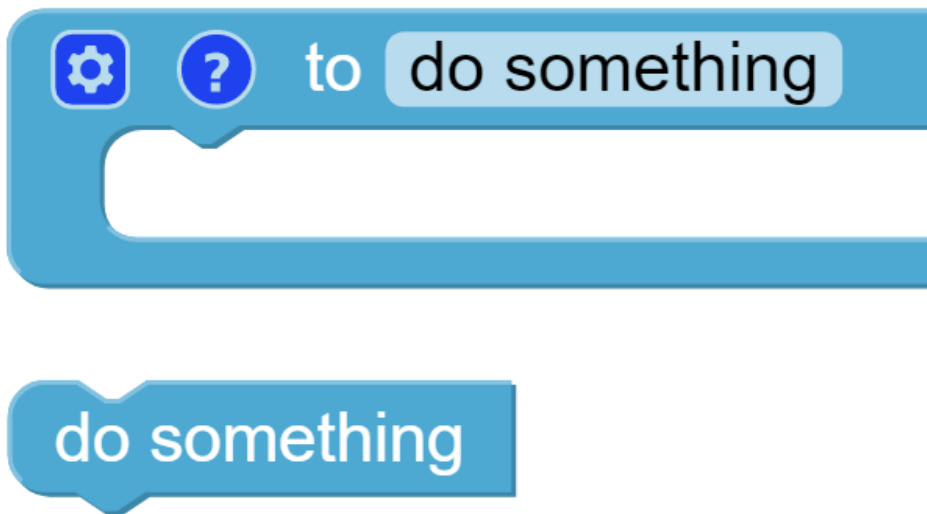
Modify the coordinate value of a certain leg individually.



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.

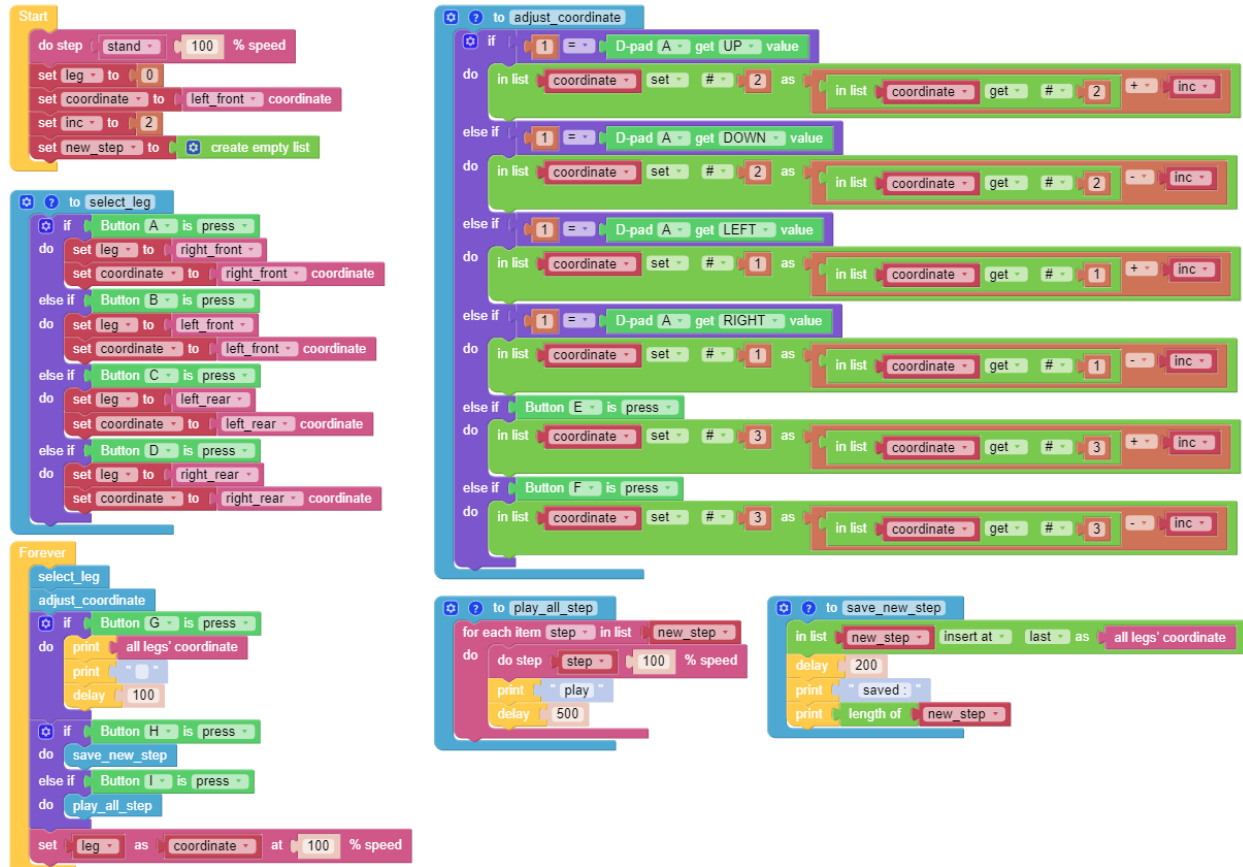


4.11 Record New Step — Ezblock

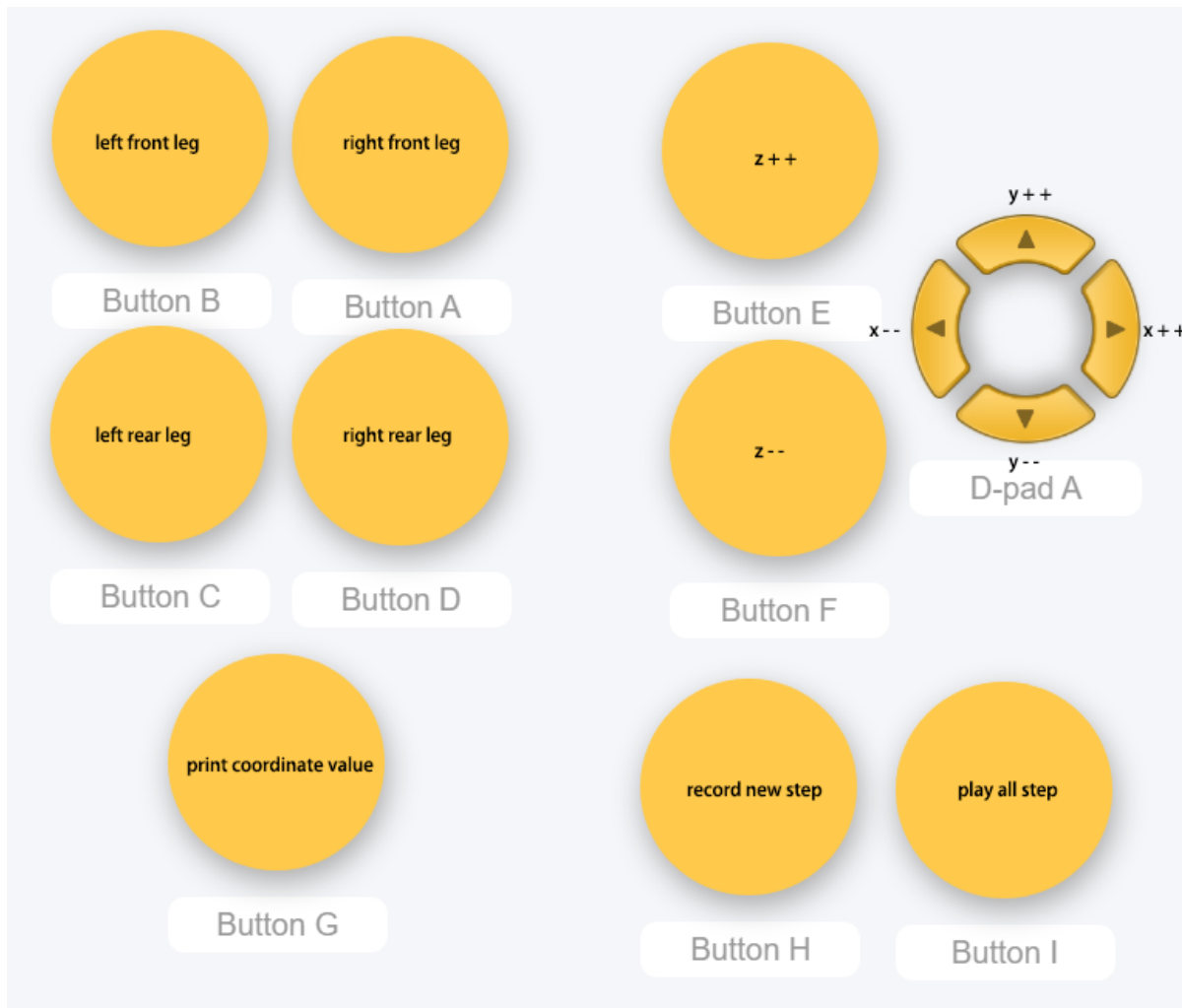
We use the remote function to control PiCrawler to make several poses in turn, and record these poses. Replay them later.

Program

After opening the example, you can see the following code block.



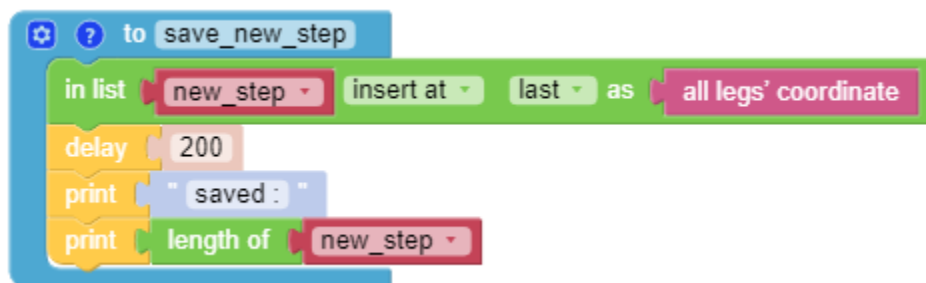
Switch to the Remote Control interface, and you will see the following widgets.



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.



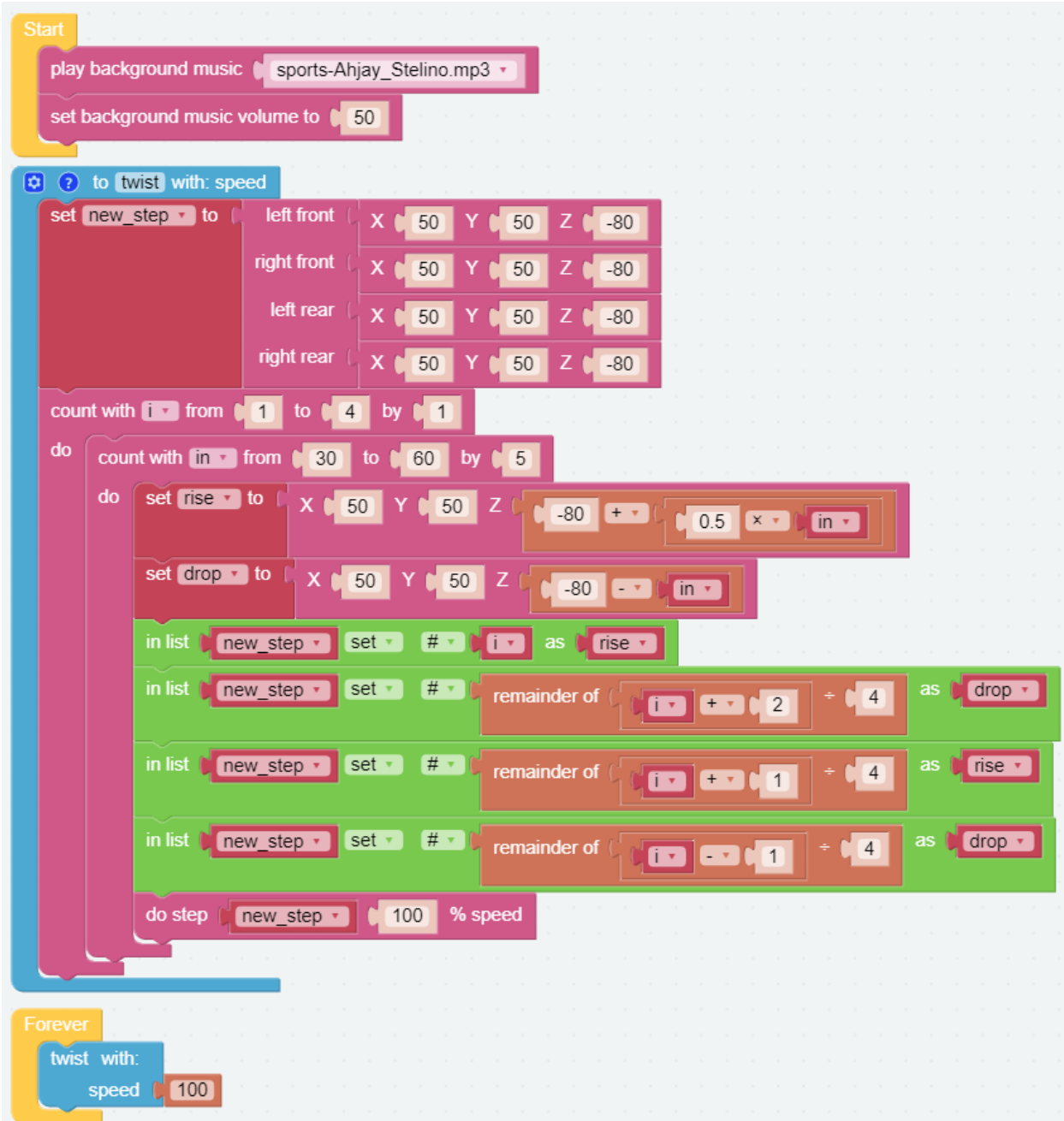
4.12 Twist — Ezblock

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

After opening the example, you can see the following code block.

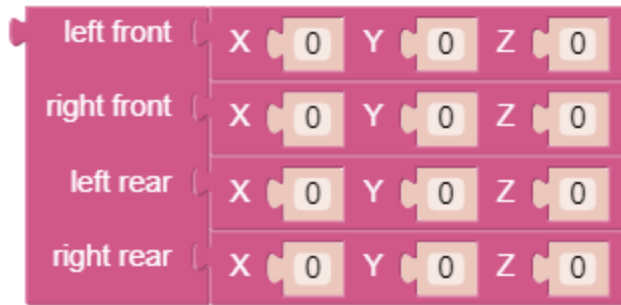


How it works?

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 — Ezblock](#).

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



left front	X	0	Y	0	Z	0
right front	X	0	Y	0	Z	0
left rear	X	0	Y	0	Z	0
right rear	X	0	Y	0	Z	0

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.

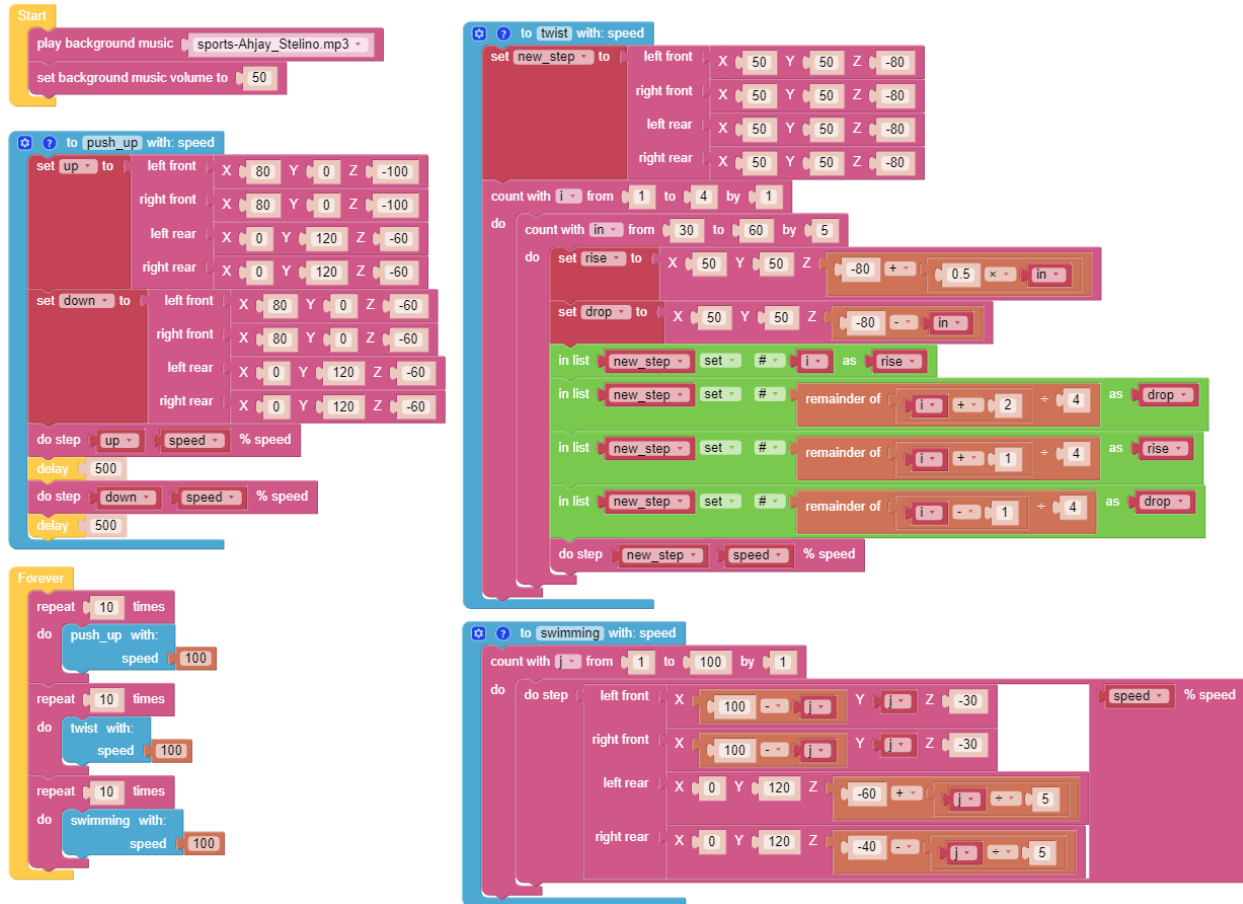
4.13 Emotional Robot — Ezblock

This example shows several interesting custom actions of PiCrawler.

It is a supplementary example of *Twist — Ezblock*.

Program

After opening the example, you can see the following code block.



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.

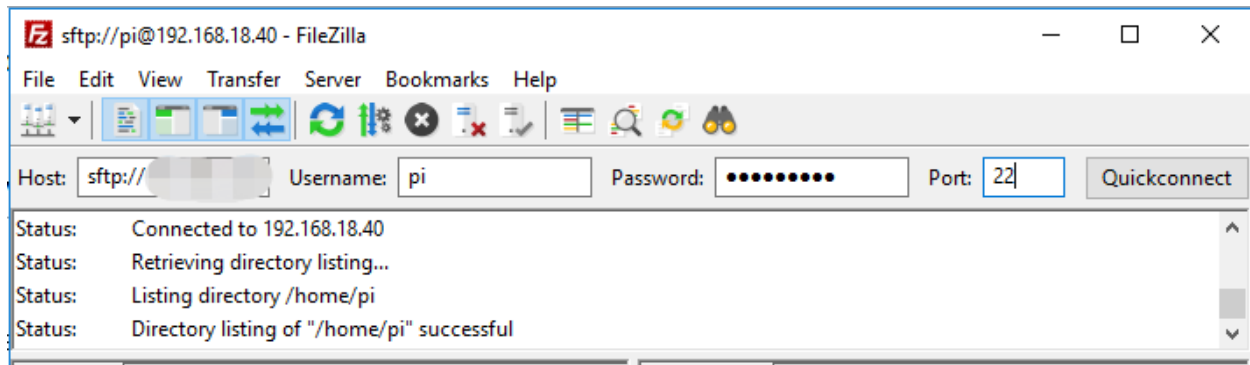
Filezilla is an open source software that not only supports FTP, but also FTP over TLS (FTPS) and SFTP. We can use Filezilla to upload local files (such as pictures and audio, etc.) to the Raspberry Pi, or download files from the Raspberry Pi to the local.

Step 1: Download Filezilla.

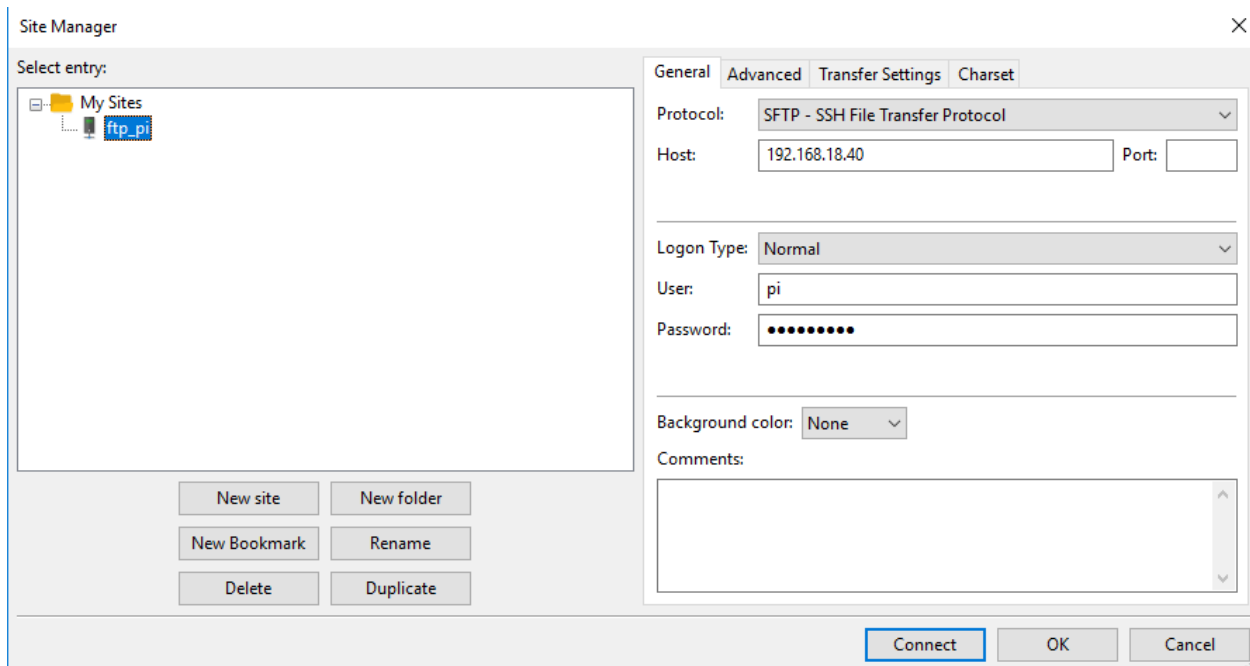
Download the client from [Filezilla's official website](#), Filezilla has a very good tutorial, please refer to: [Documentation - Filezilla](#).

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.

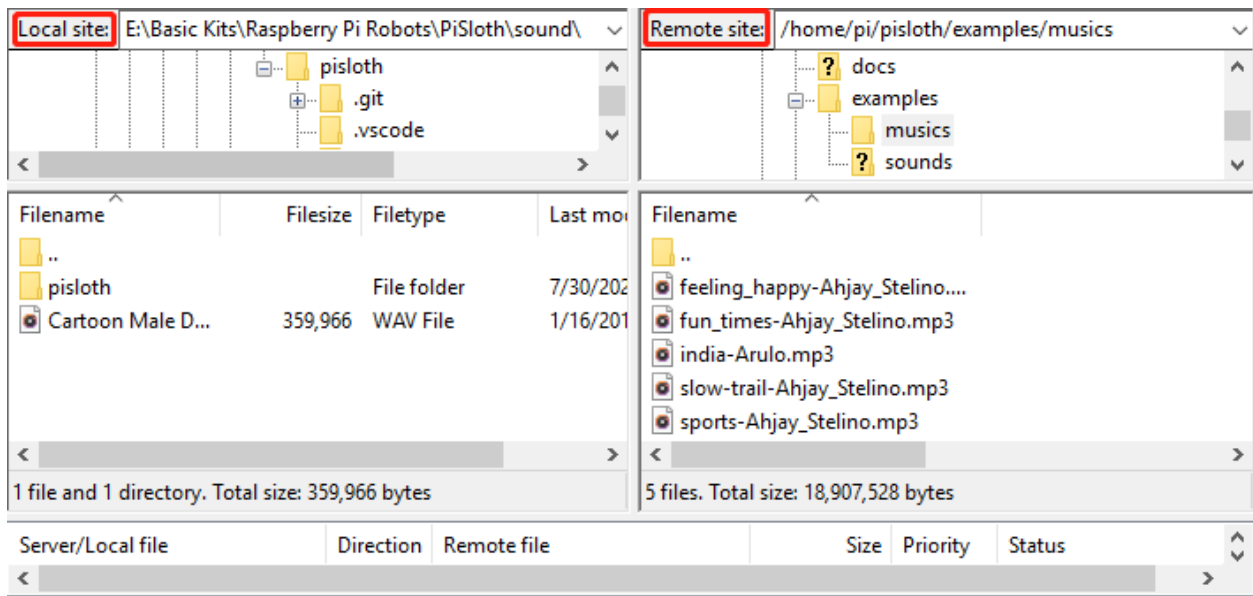


Note: Quick Connect is a good way to test your login information. If you want to create a permanent entry, you can select **File-> Copy Current Connection to Site Manager** after a successful Quick Connect, enter the name and click **OK**. Next time you will be able to connect by selecting the previously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

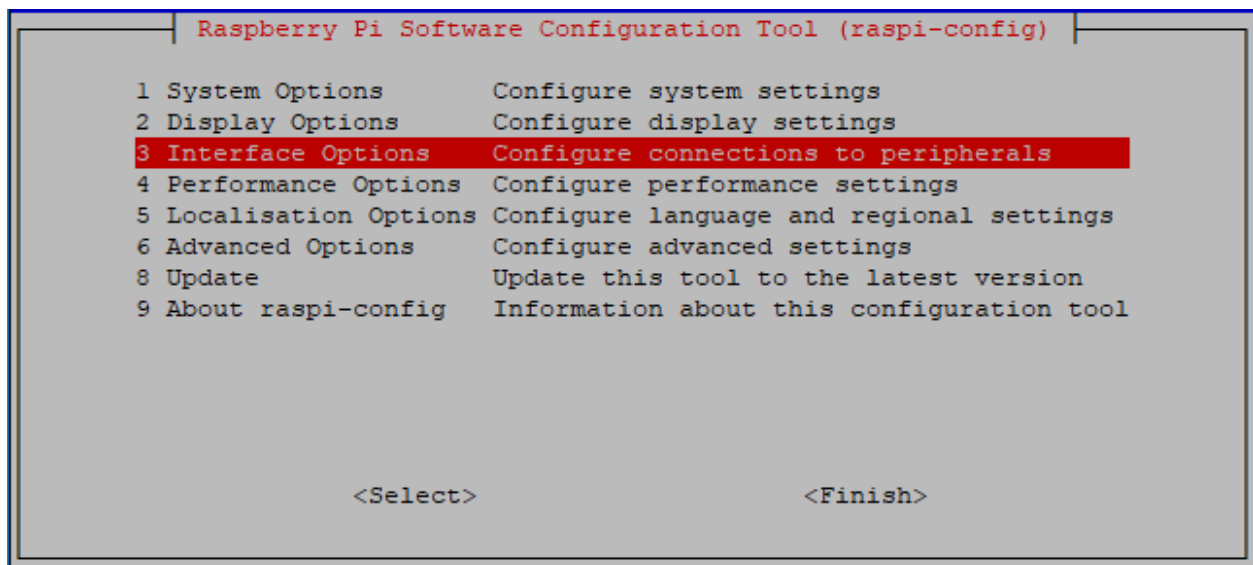


5.2 I2C Configuration

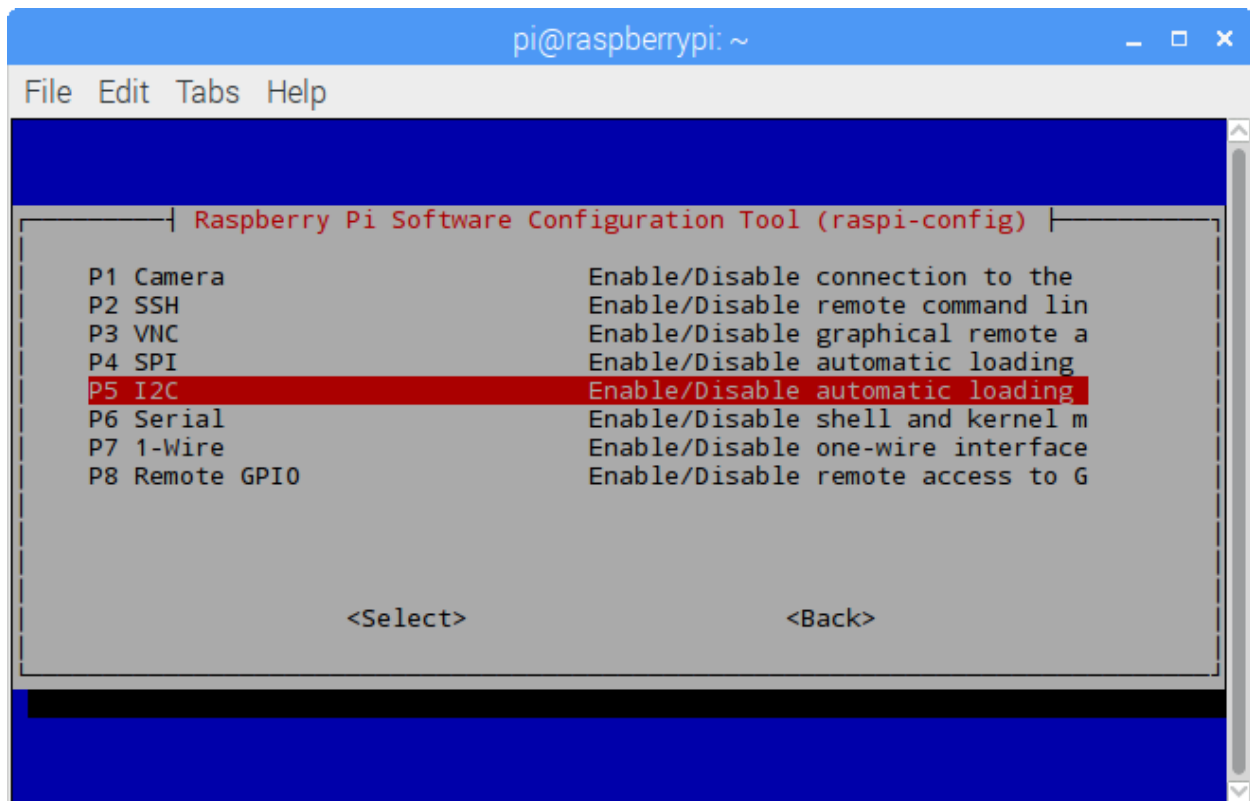
Enable the I2C port of your Raspberry Pi (If you have enabled it, skip this; if you do not know whether you have done that or not, please continue).

```
sudo raspi-config
```

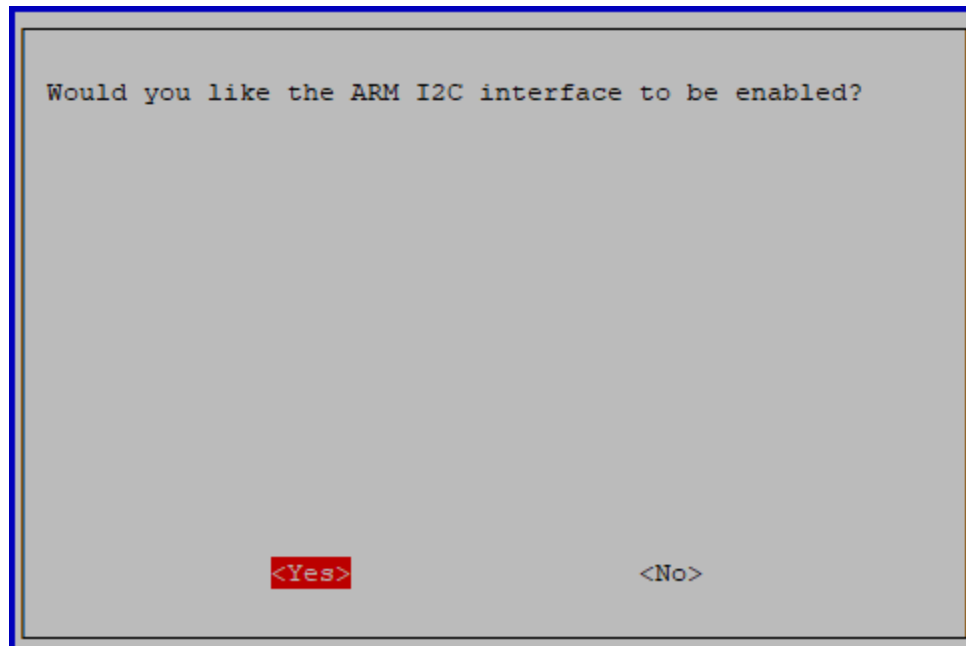
3 Interfacing options



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.

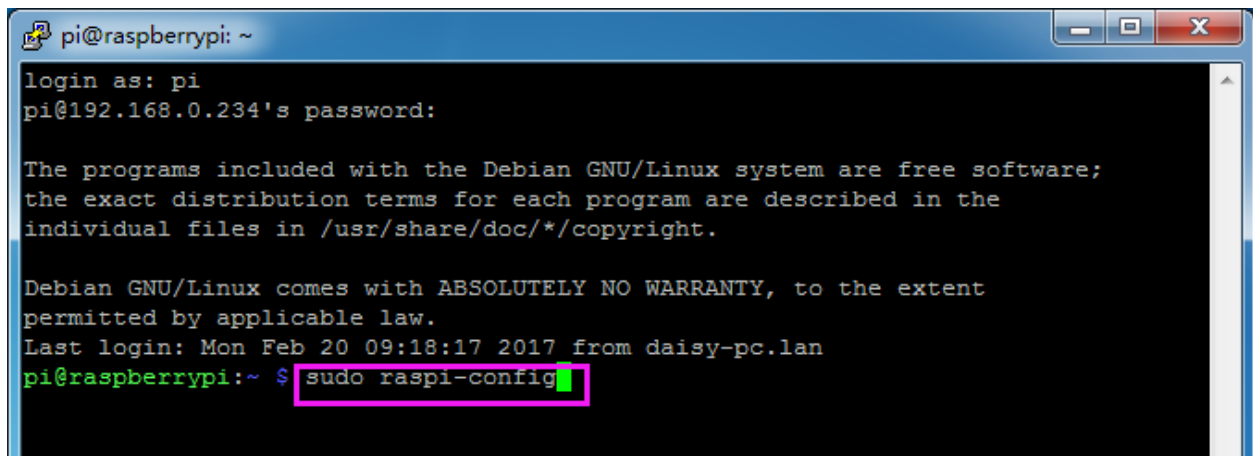
Enable VNC service

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

Step 1

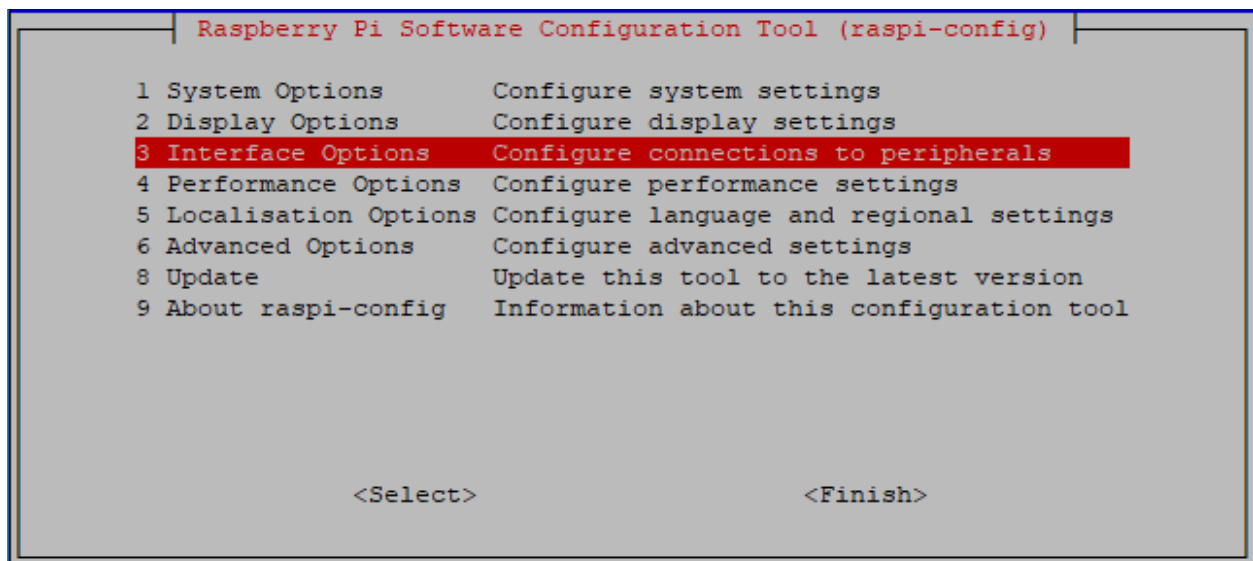
Input the following command:

```
sudo raspi-config
```



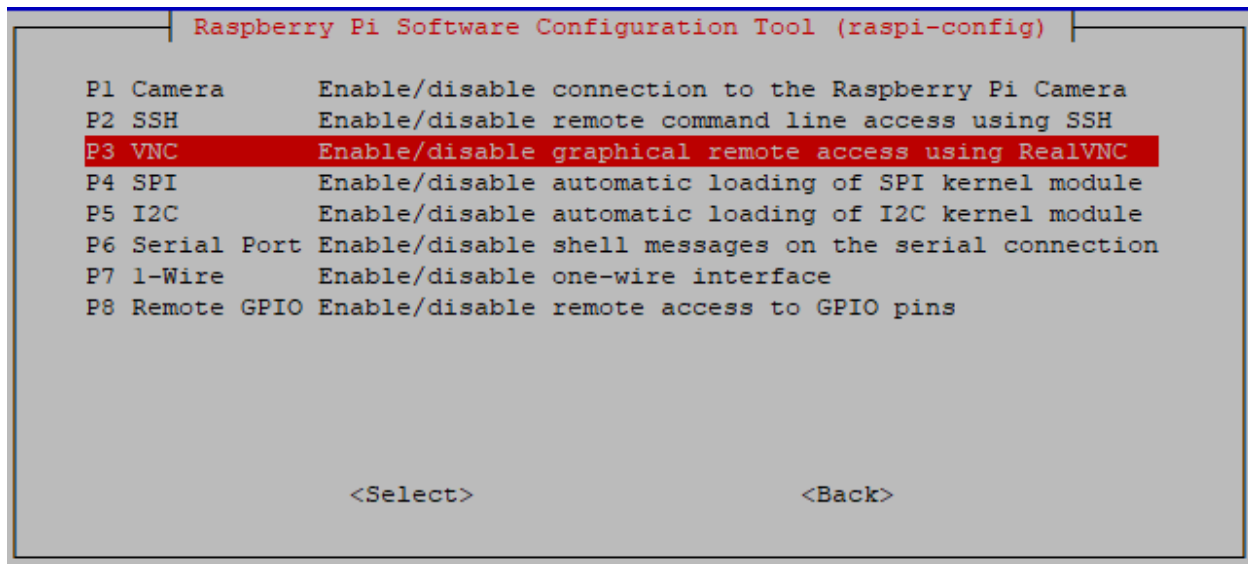
Step 2

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



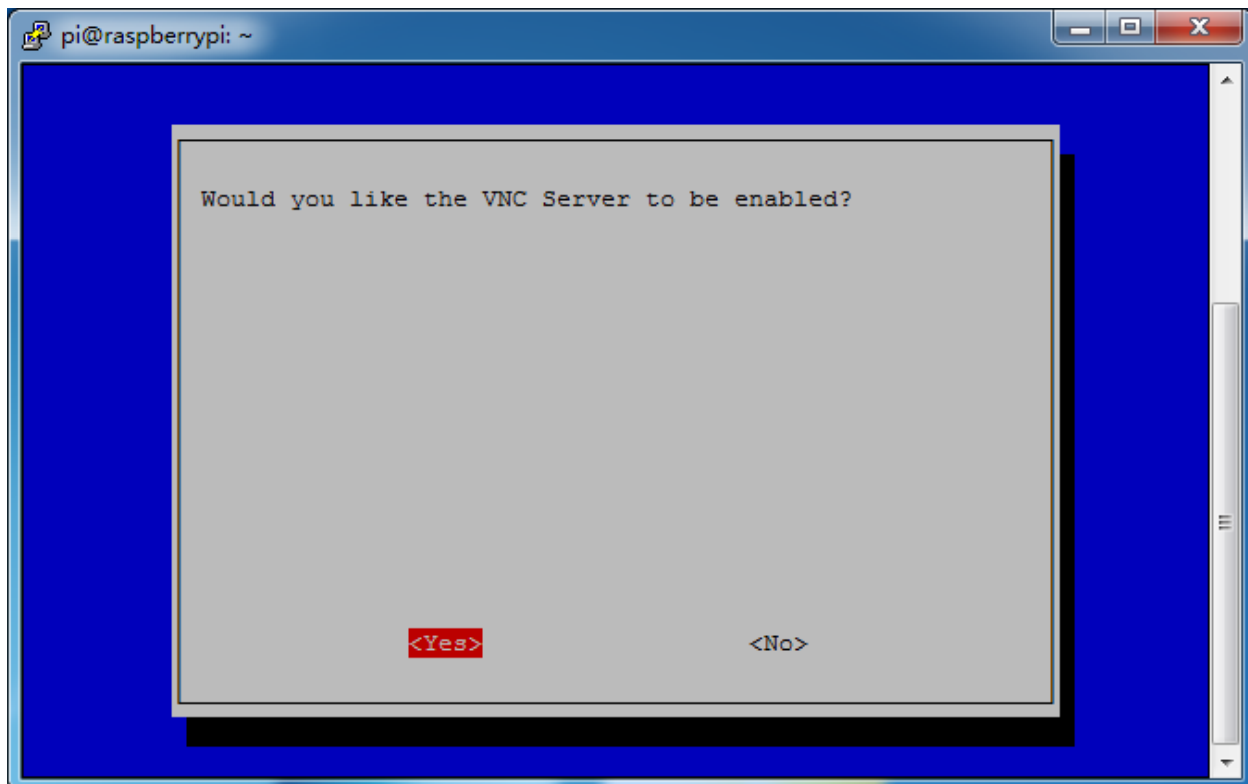
Step 3

P3 VNC



Step 4

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



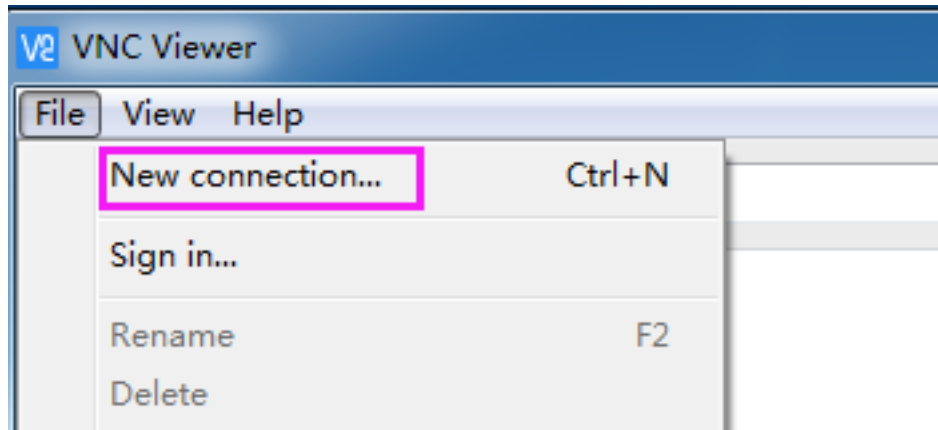
Login to VNC

Step 1

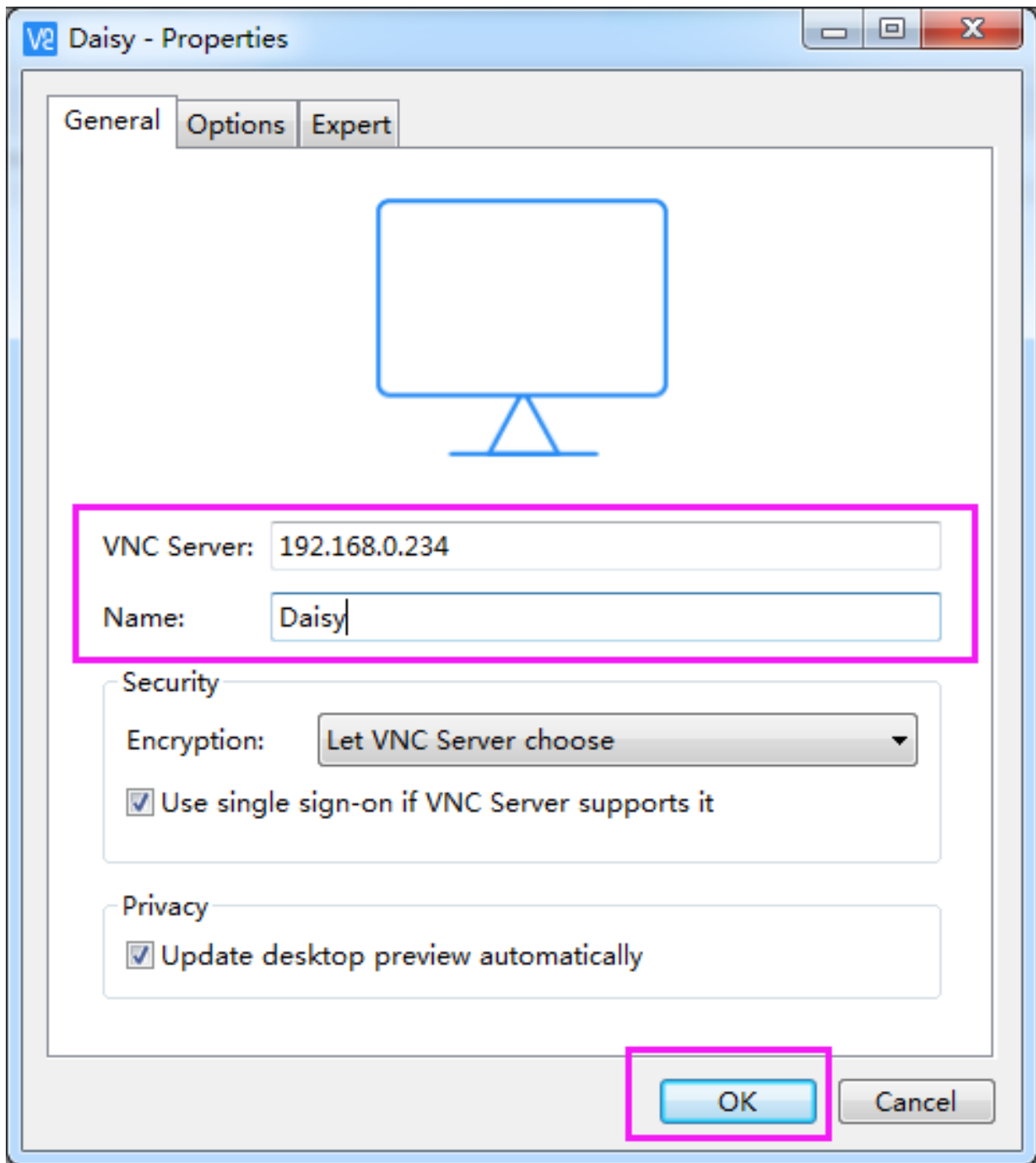
You need to download and install the [VNC Viewer](#) on personal computer. After the installation is done, open it.

Step 2

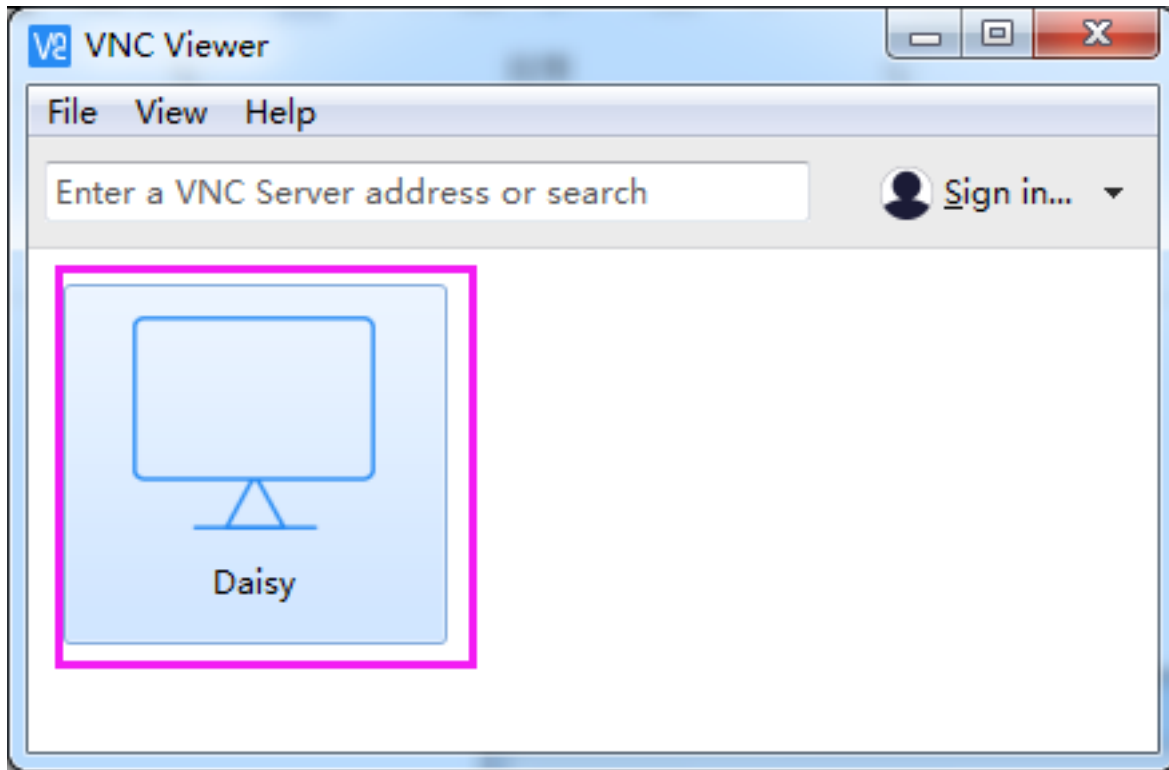
Then select “**New connection**”.

**Step 3**

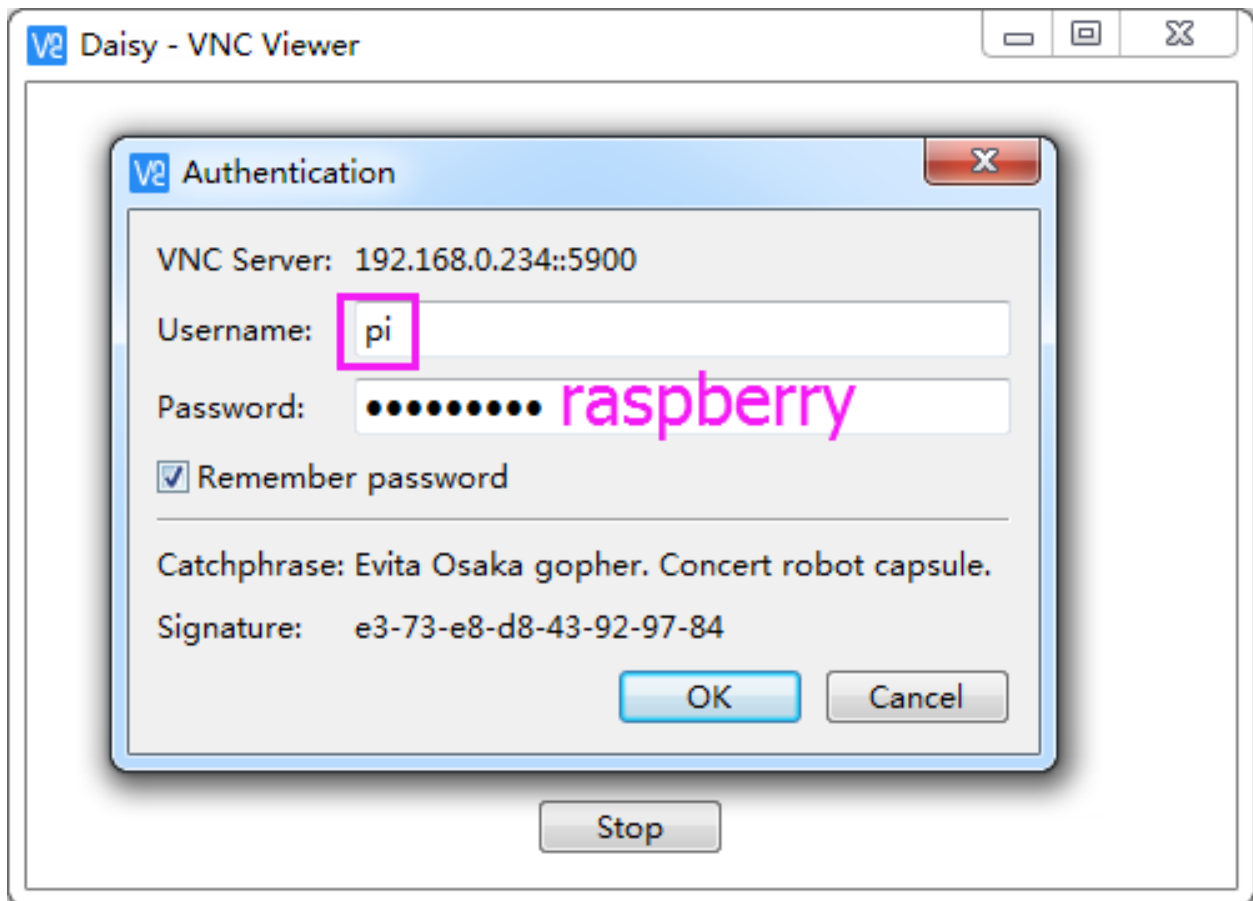
Input IP address of Raspberry Pi and any **Name**.

**Step 4**

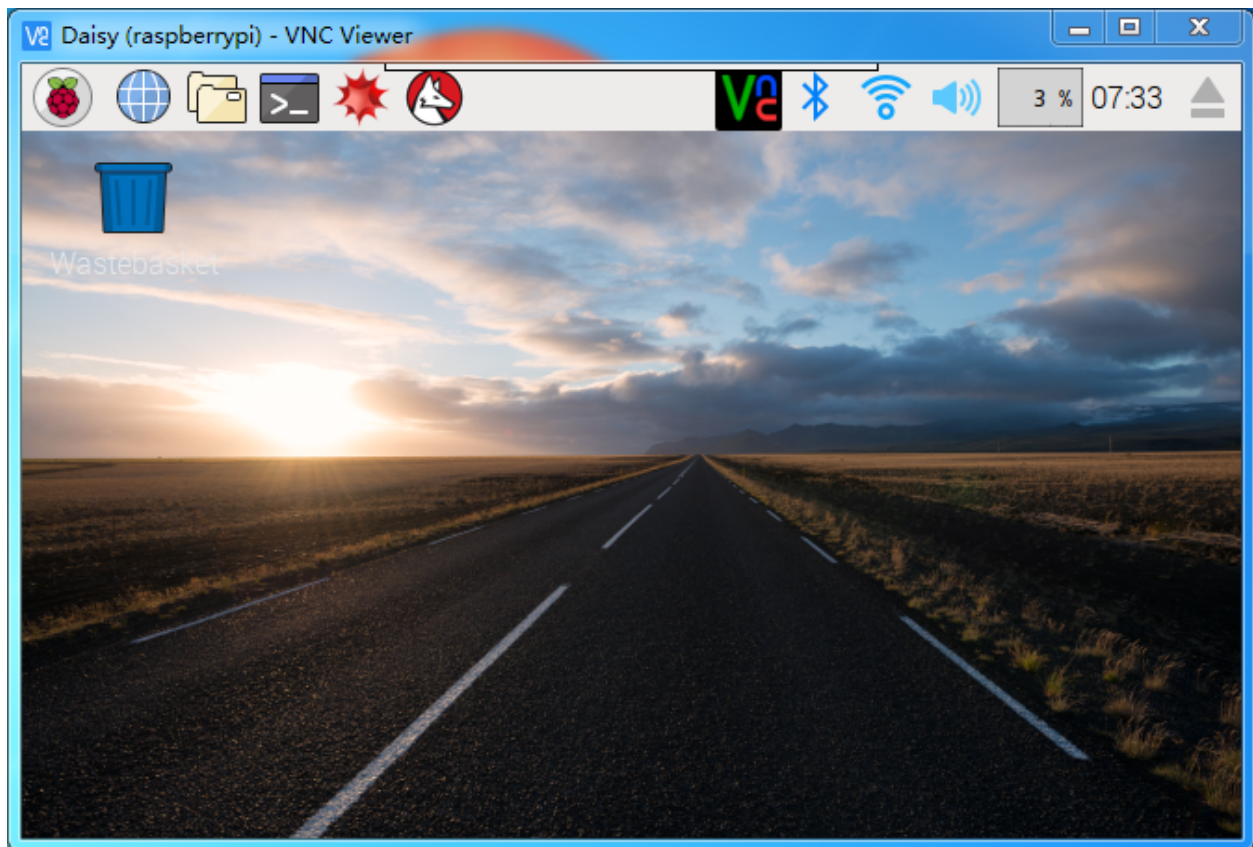
Double click the **connection** just created:

**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

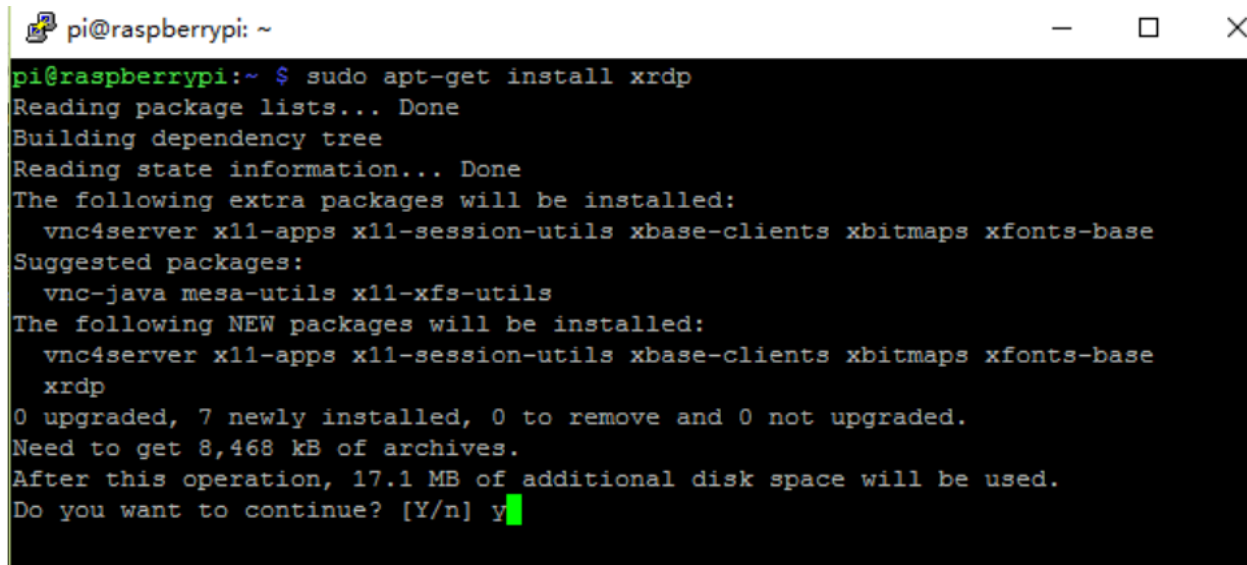
Input the following instructions to install XRDP.

```
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.

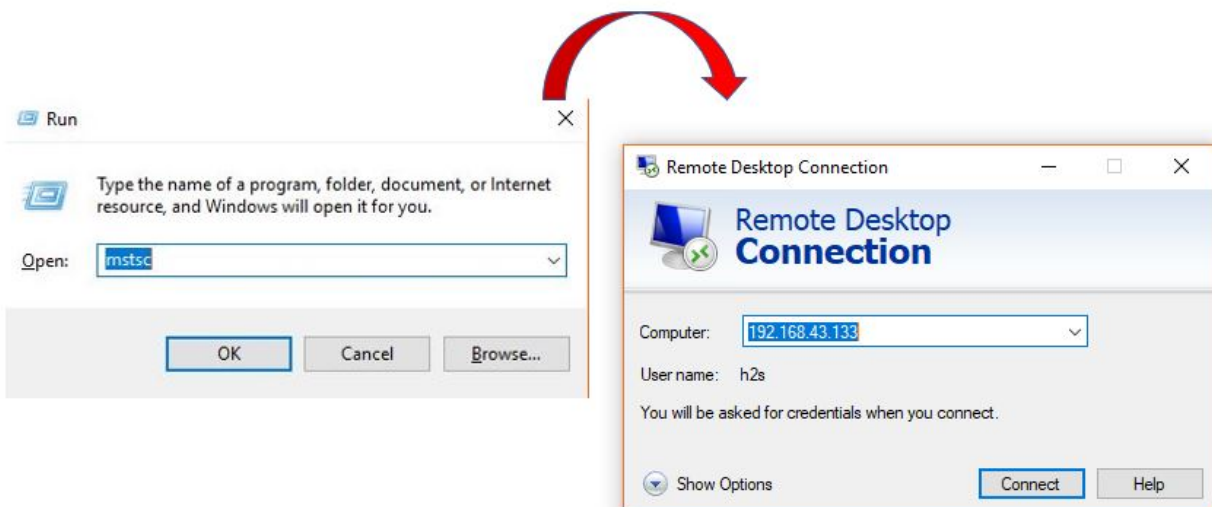
Login to XRDP

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.

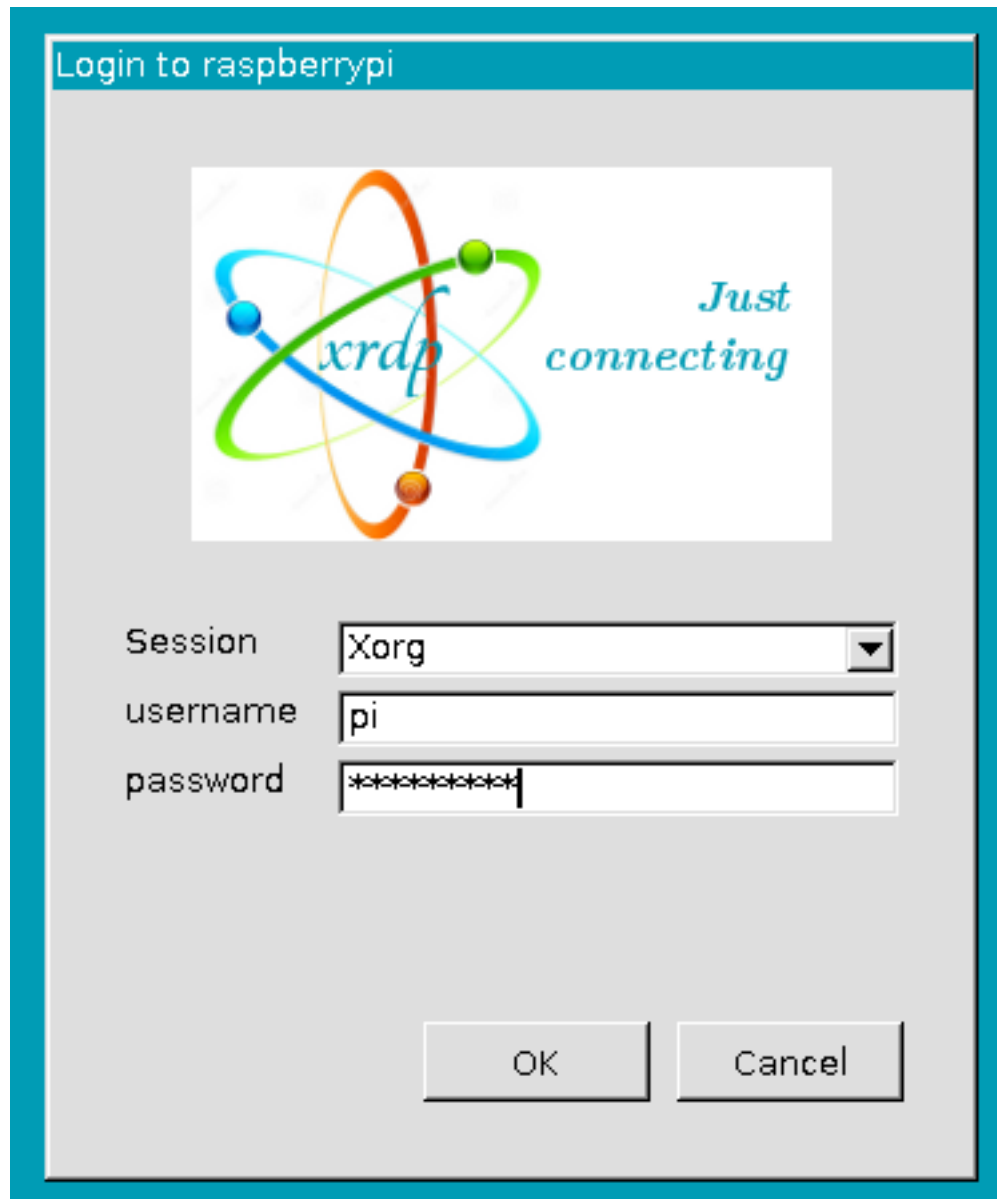
Step 2

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”.



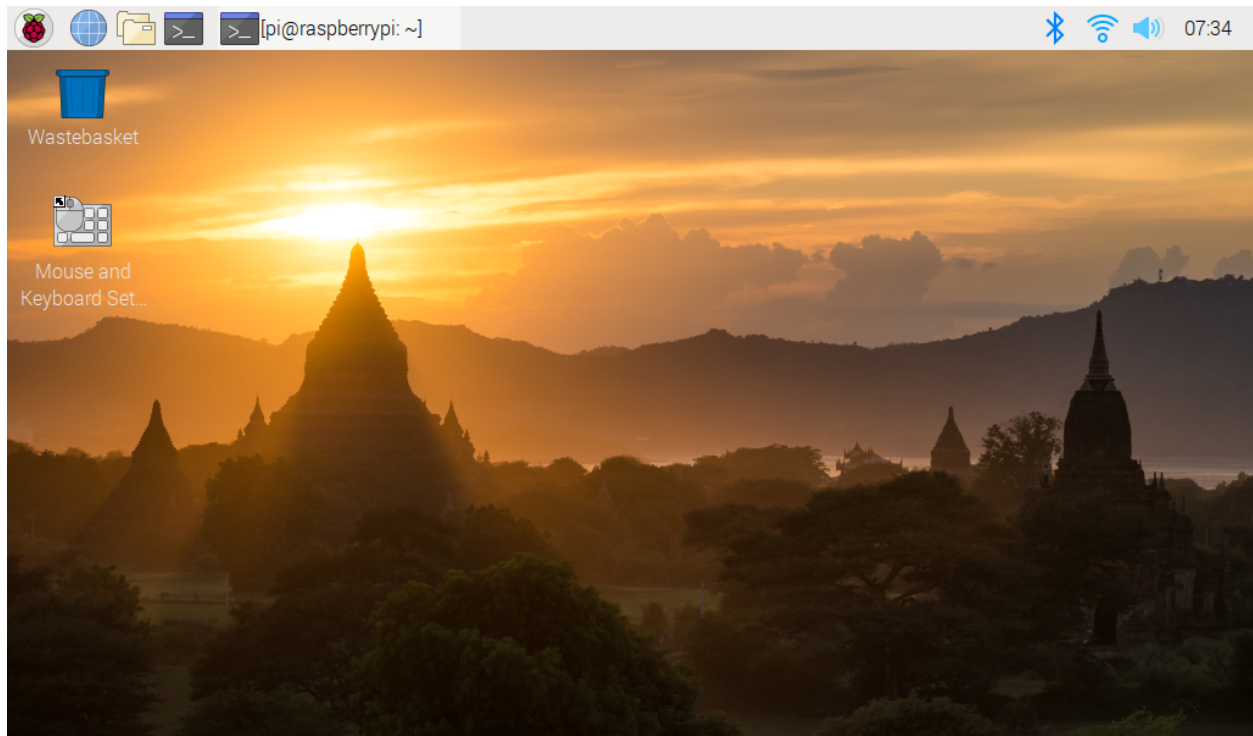
Step 3

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 “raspberrypi”.



Step 4

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



5.4 About the Battery

Applicable Parameters

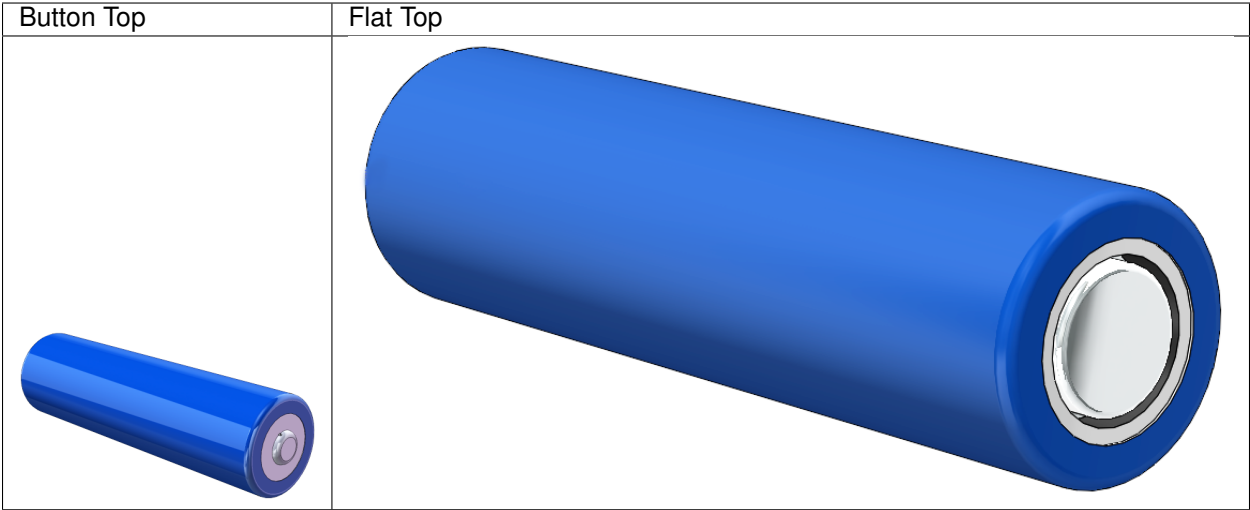
- 3.7V
- 18650
- Rechargeable
- Li-ion Battery
- Button Top
- No Protective Board

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.



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.

COPYRIGHT NOTICE

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