
SunFounder pisloth

www.sunfounder.com

Dec 29, 2021

CONTENTS

1	Component List and Assembly Instructions	3
2	About Robot HAT	5
3	Play with Ezblock	7
3.1	Quick Guide on Ezblock	7
3.2	Move	9
3.3	Don't Touch Me	12
3.4	Obstacle Avoidance	15
3.5	Emotional PiSloth	17
3.6	Dance	21
3.7	Let's Fight! Warrior!	25
3.8	Remote Control	29
3.9	Custom Step	32
3.10	Custom Action	38
4	Play with Python	41
4.1	Quick Guide on Python	41
4.2	Move	60
4.3	Dance	62
4.4	Obstacle Avoidance	65
4.5	Don't Touch Me	67
4.6	Let's Fight! Warrior!	69
4.7	Emotional PiSloth	71
4.8	Remote Control	74
4.9	Custom Step	76
4.10	Custom Action	79
5	Appendix	83
5.1	Filezilla Software	83
5.2	I2C Configuration	85
5.3	Remote Desktop	87
5.4	About the Battery	96
6	Thank You	99
7	Copyright Notice	101

Thanks for choosing our PiSloth.

PiSloth is a Raspberry Pi Bionic robot with an aluminum alloy structure. It can talk, dance, and even express emotions, such as happiness and excitement.

It has 22 different actions, such as: Stomp, Swing and MoonWalk, and you can customize the actions according to your needs. PiSloth's eyes consist of an ultrasonic sensor module that can be used to detect distance for obstacle avoidance and following functions.

In this tutorial, a list and assembly pdf, introduction to Robot HAT, and programming of PiSloth are included.

The programming part is divided into two chapters: *Play with Ezblock* & *Play with Python*, and each of them can get you started on making PiSloth work in way you want.

Ezblock Studio 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.

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.

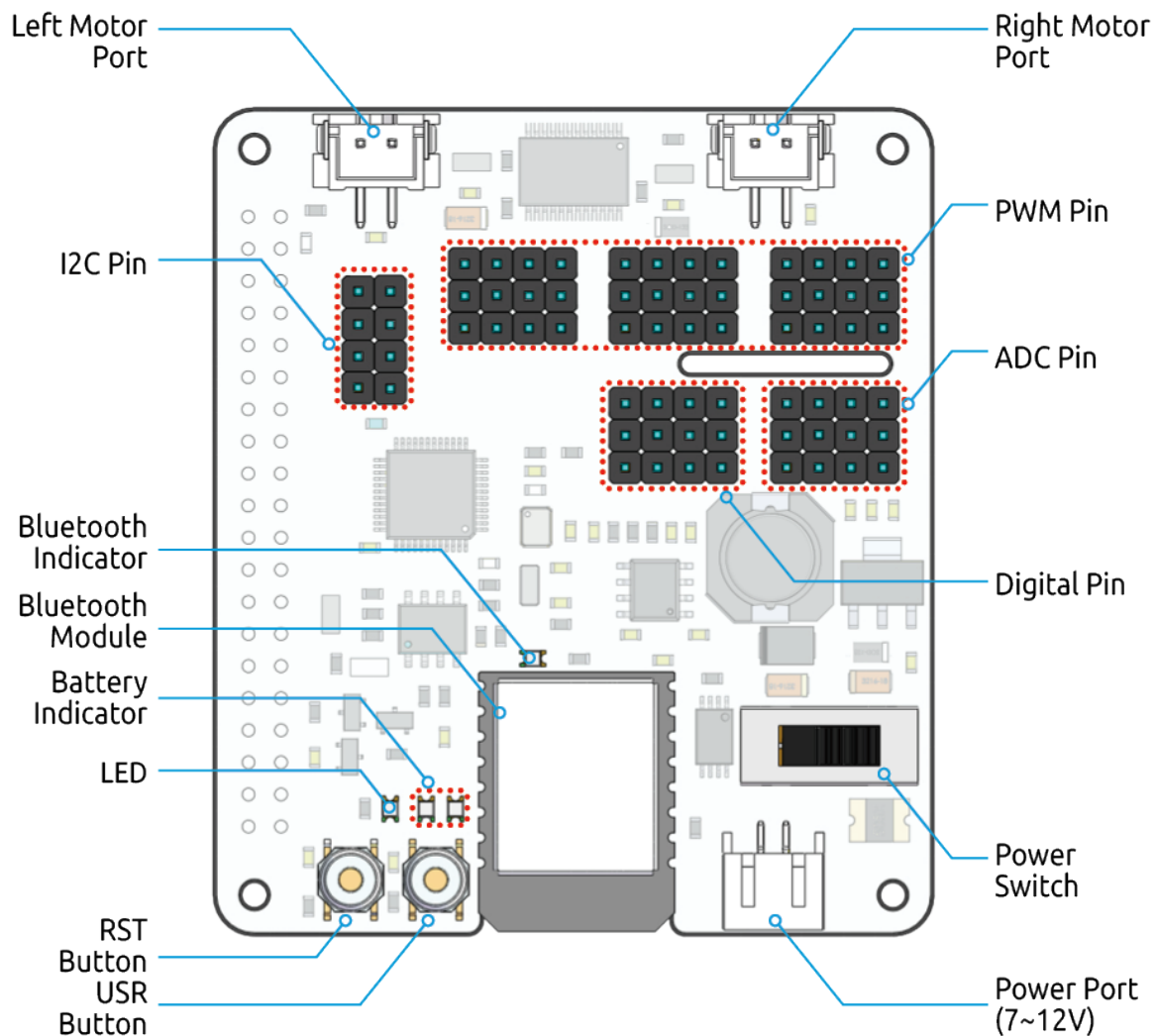
Please follow the steps on the PDF to assemble.

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 pisloth

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 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.

3.1 Quick Guide on Ezblock

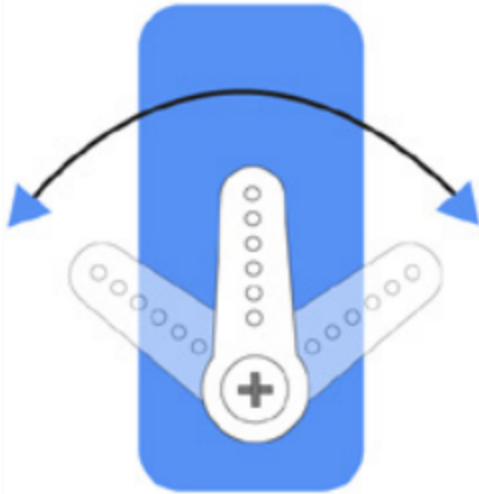
There are 2 parts here:

- *Before Assembling PiSloth* 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 PiSloth.

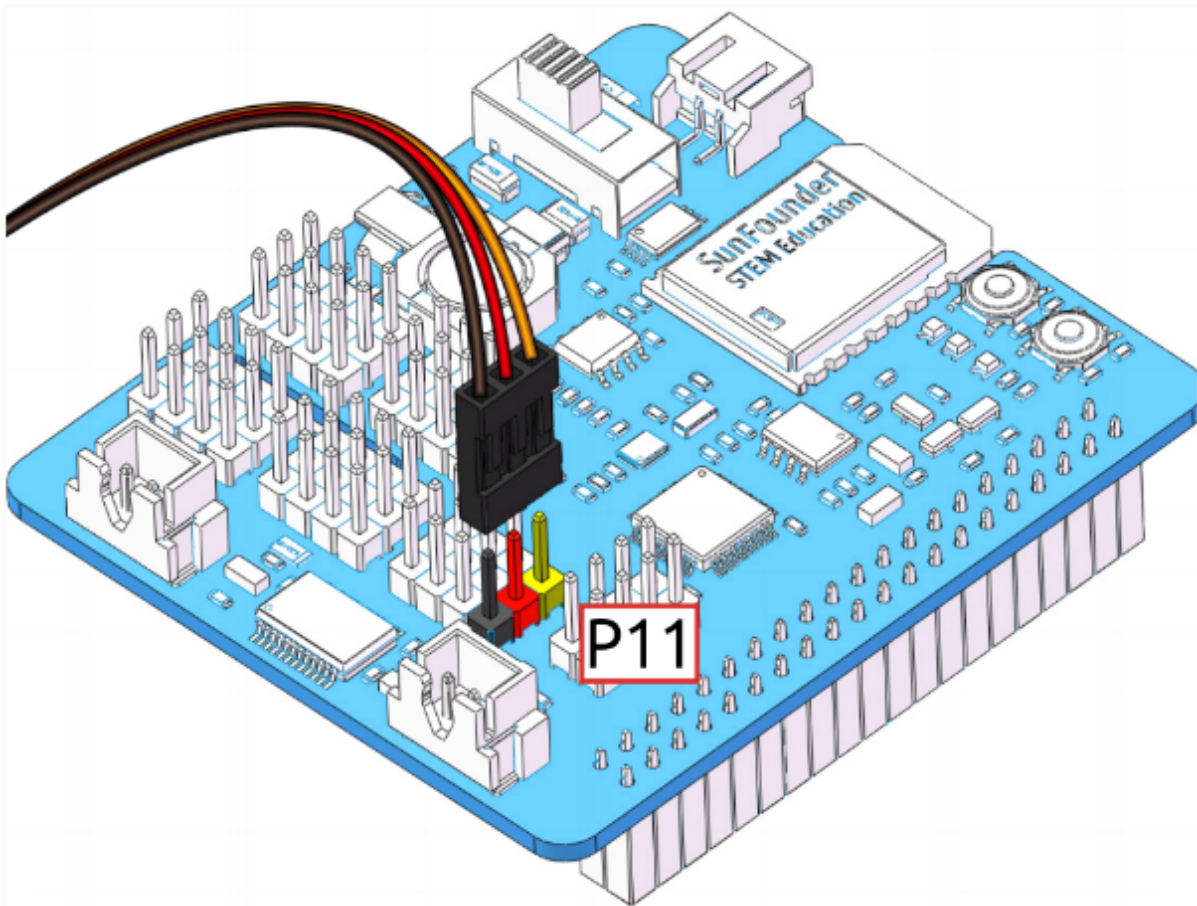
3.1.1 Before Assembling PiSloth

Before assembling the PiSloth, 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.

3.1.2 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 PiSloth, 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 PiSloth 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.

3.2 Move

This is the first project. PiSloth has woken up, and it moves freely.

Before programming, you need to learn the basic usage of Ezblock Studio from here.

- [Quick User Guide for Ezblock 3](#)
- [How to Create a New Project?](#)



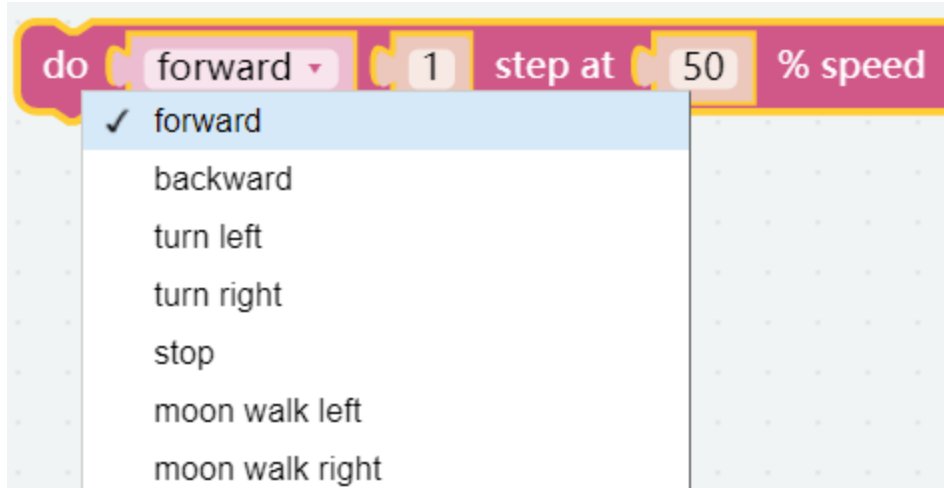
TIPS

This is the basic structure of the program, the Start block is used to do some initialization (even if no block is placed, it cannot be deleted) and the Forever block is, as the name suggests, a continuous loop that allows your program to change and respond.



This block is used to make PiSloth do a specific action several steps at a speed (%), for example, let PiSloth go forward 1 step at 50% speed.

Different actions can be selected from the drop down options, there are 22 in total.



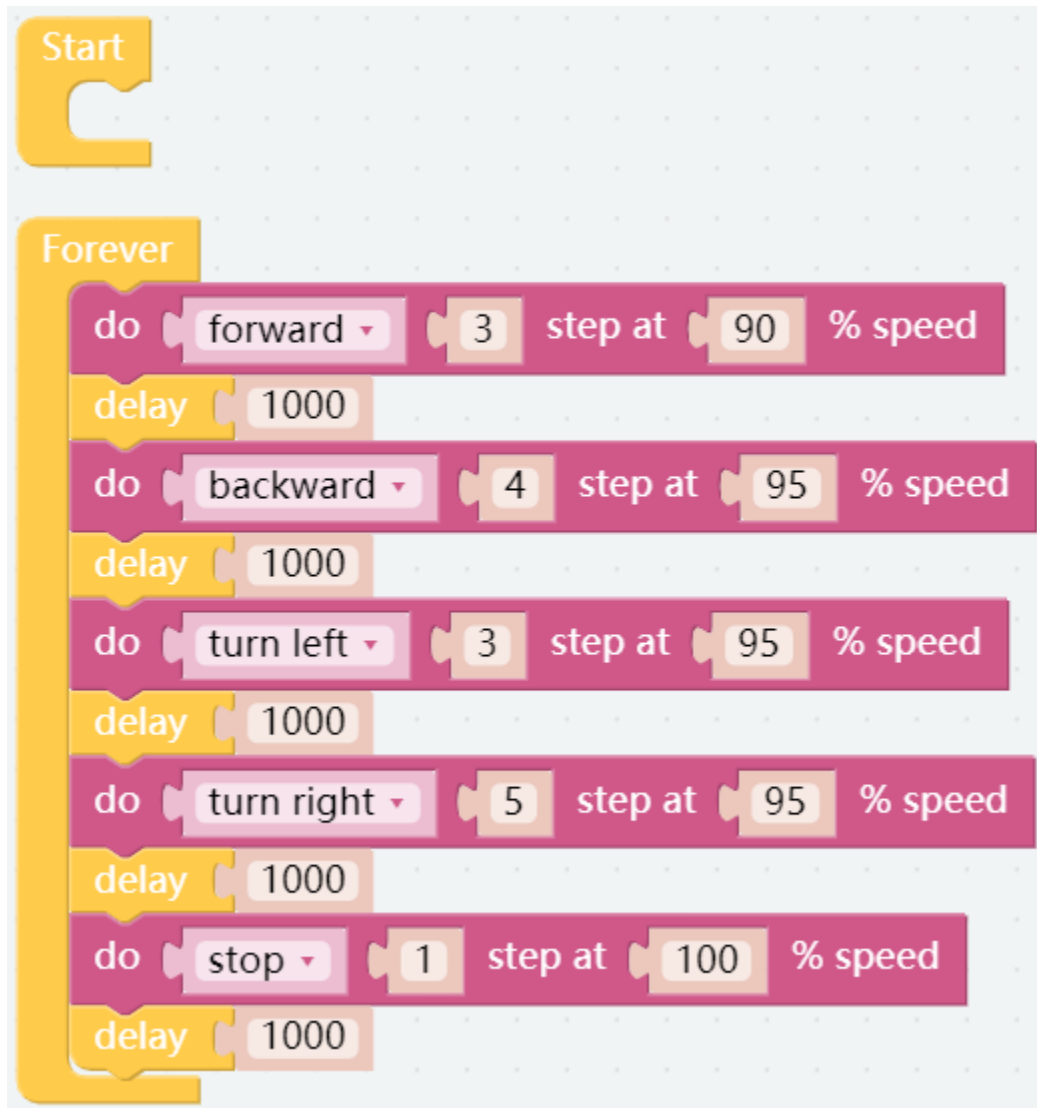
This is a block that sets the duration of the previous block, unit: ms.



EXAMPLE

After writing the code according to the following figure, click the download icon in the bottom right corner, you will see PiSloth move forward 3 steps, backward 4 steps, left 3 steps, right 5 steps, and finally stop. Since the whole code is placed inside the Forever block, PiSloth will repeat the above actions after stopping for a while.

You can try putting the code from the Forever block into the Start block and see what happens.



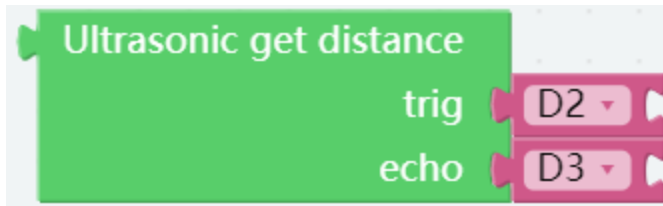
3.3 Don't Touch Me

If you don't meet PiSloth's needs, it will get angry and stay away from your touch.

TIPS

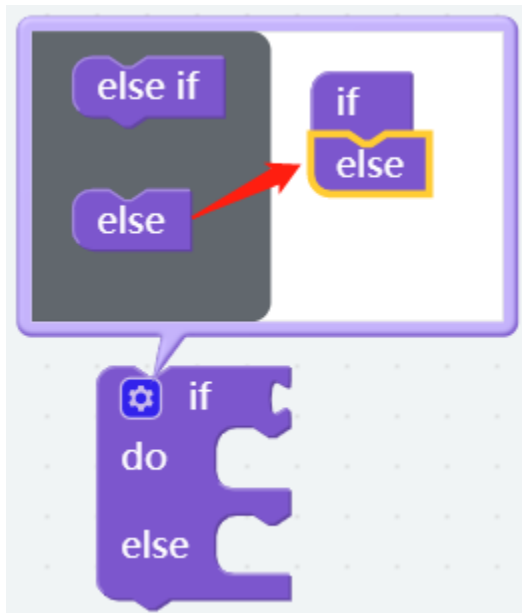
You can directly use this block to read the distance to the obstacle right ahead.

Note: When assembling, Trig and Echo are connected to D2 and D3 respectively, you also need to change them simultaneously when programming.



To achieve conditional judgment of “if” type, you need to use an **if do** block.

When you need to implement multiple conditional judgments, you will have to change **if do** into **if else do**. This can be achieved by clicking on the **setting** icon.



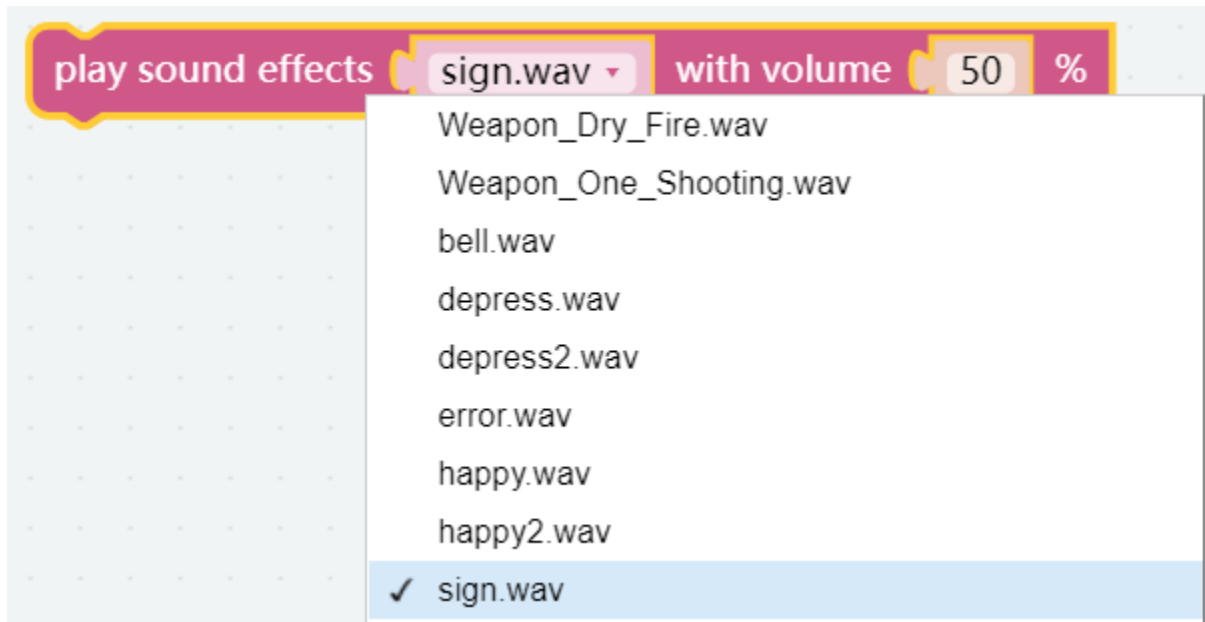
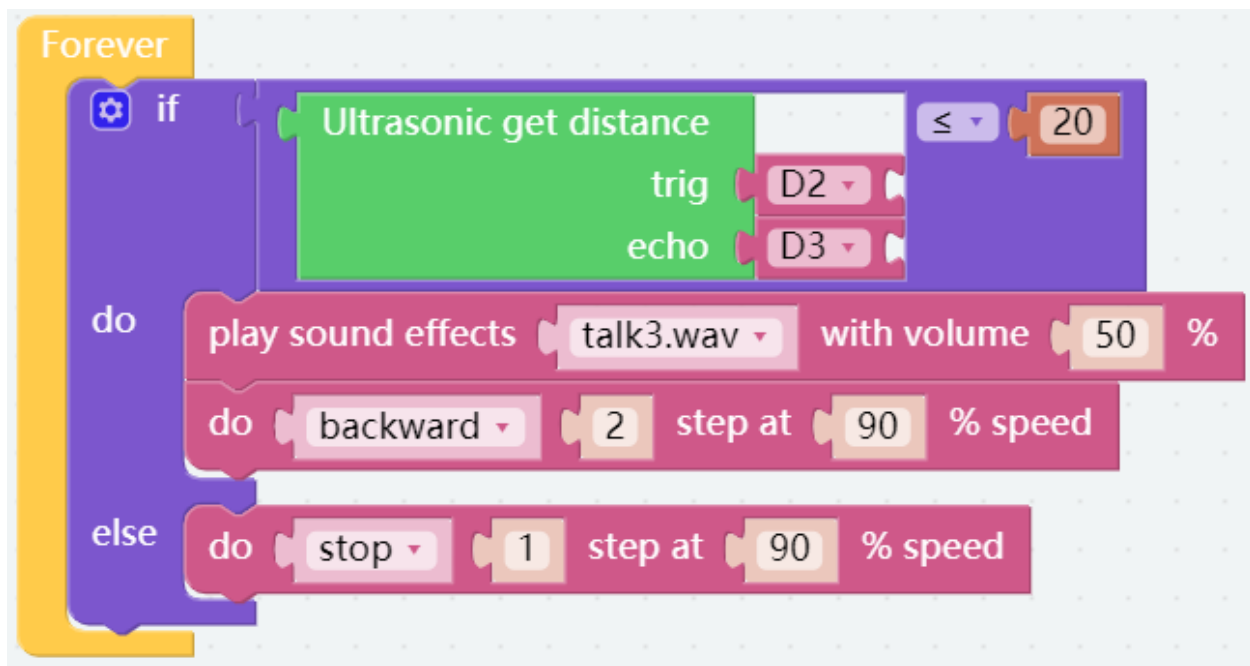
You need to use a conditional statements block in conjunction with if do. Judging conditions can be “=”, “>”, “<”, “”, “”, “”.



A number block.



This block can emit some preset sound effects, such as siren sound, gun sound and so on. The range of volume is 1~100.

**EXAMPLE**

3.4 Obstacle Avoidance

In this project, when PiSloth detects an obstacle, it will send a signal and look for another direction to move forward.

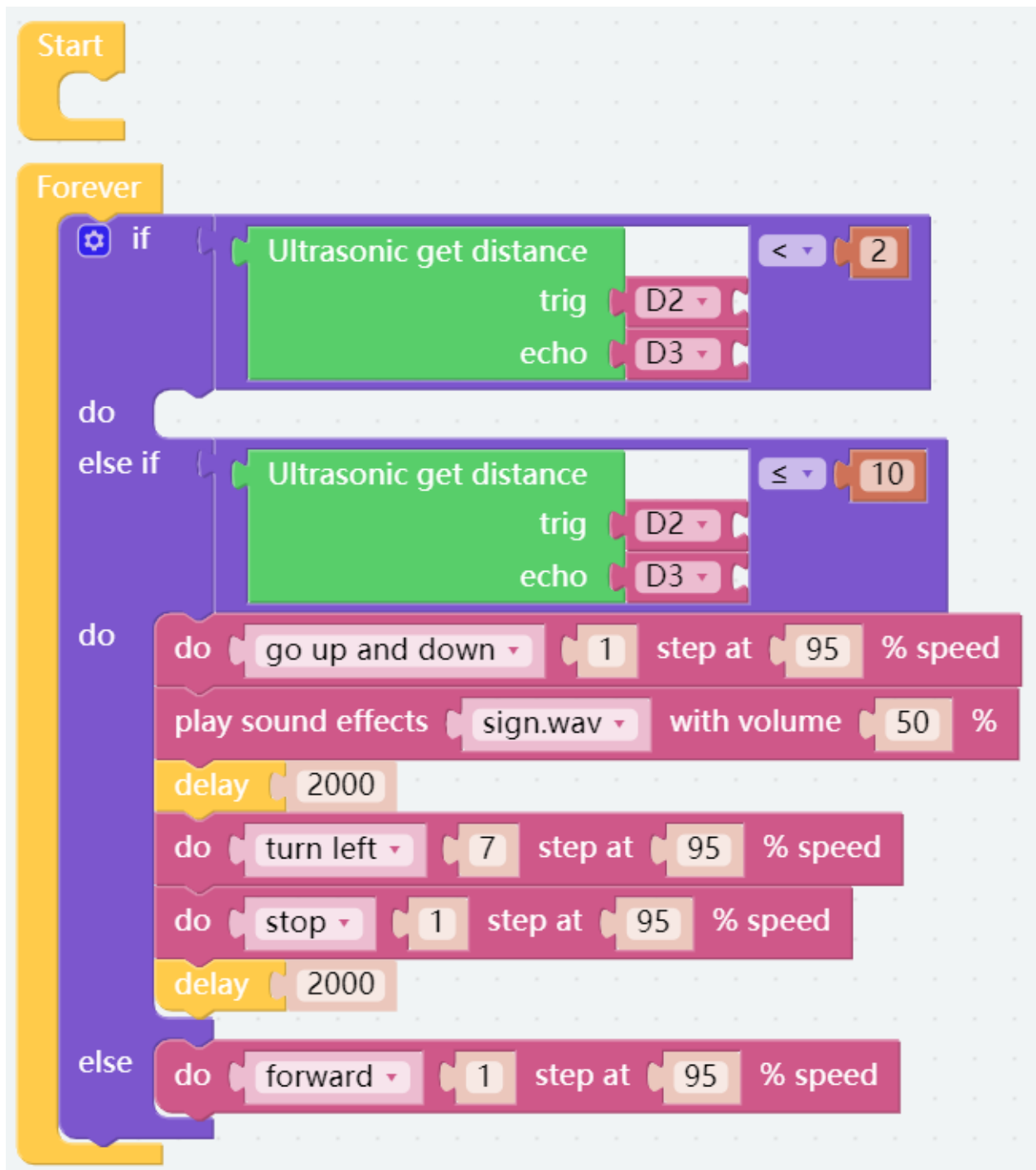
TIPS

This is based on the previous project *Don't Touch Me*, which adds autonomous judgment, so that PiSloth can actively avoid obstacles in front of it.

EXAMPLE

After the code runs, PiSloth 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 for 7 steps 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.

Since the effective detection distance of the ultrasonic sensor module is 2-400cm, when the detection distance is less than it will do nothing.



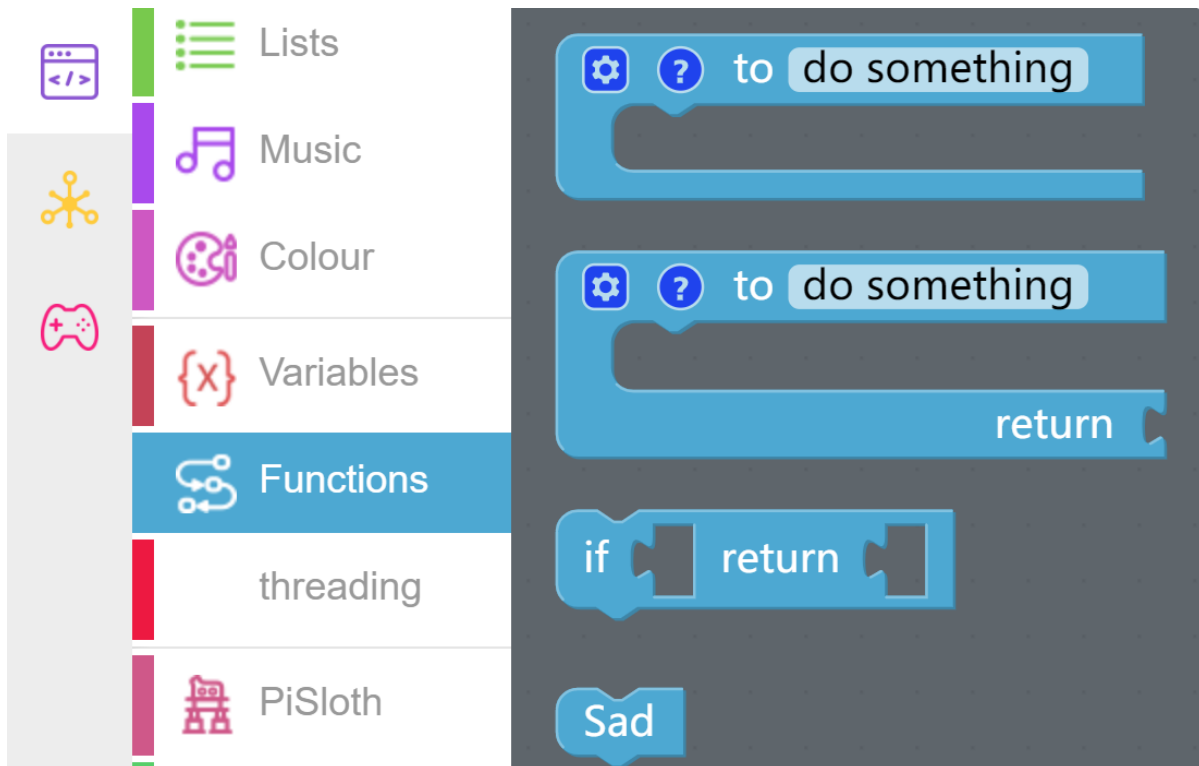
3.5 Emotional PiSloth

PiSloth is very emotional, sometimes happy, sometimes shy, sometimes confused.

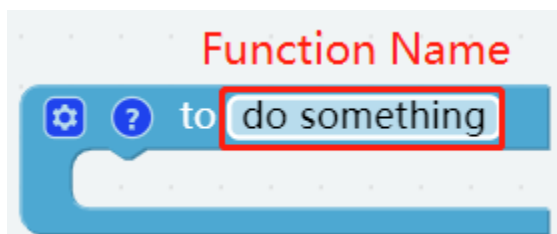
TIPS

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.

Click on the **Functions** category and select the appropriate function block, the function you created will also appear here.

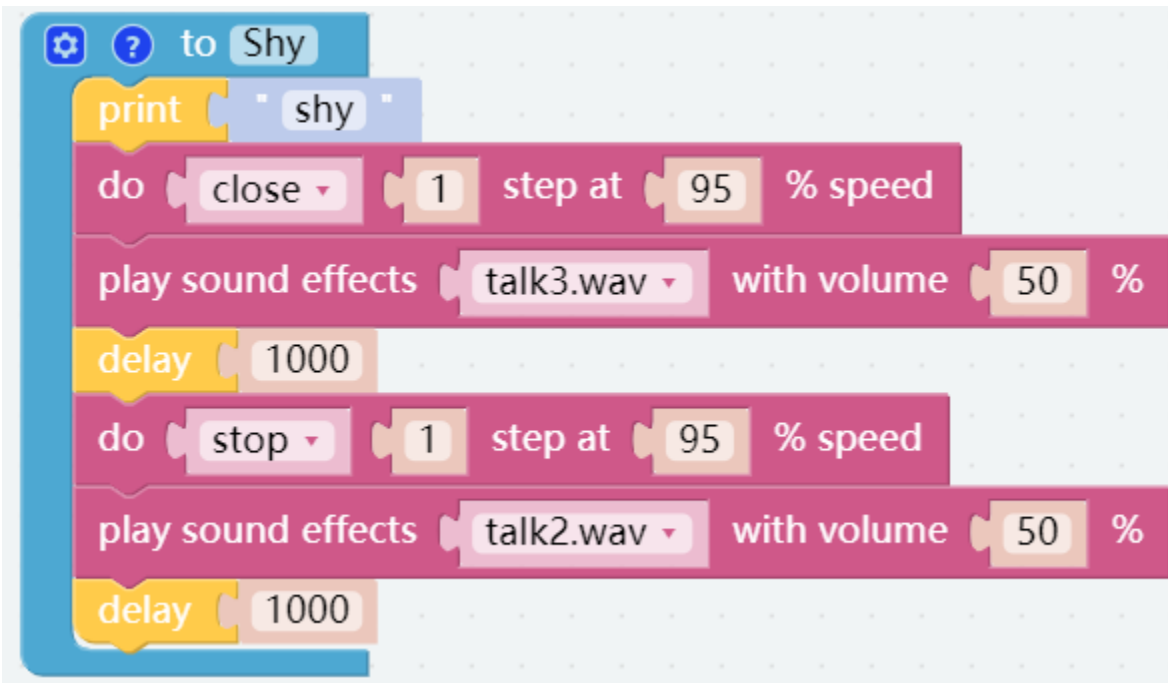
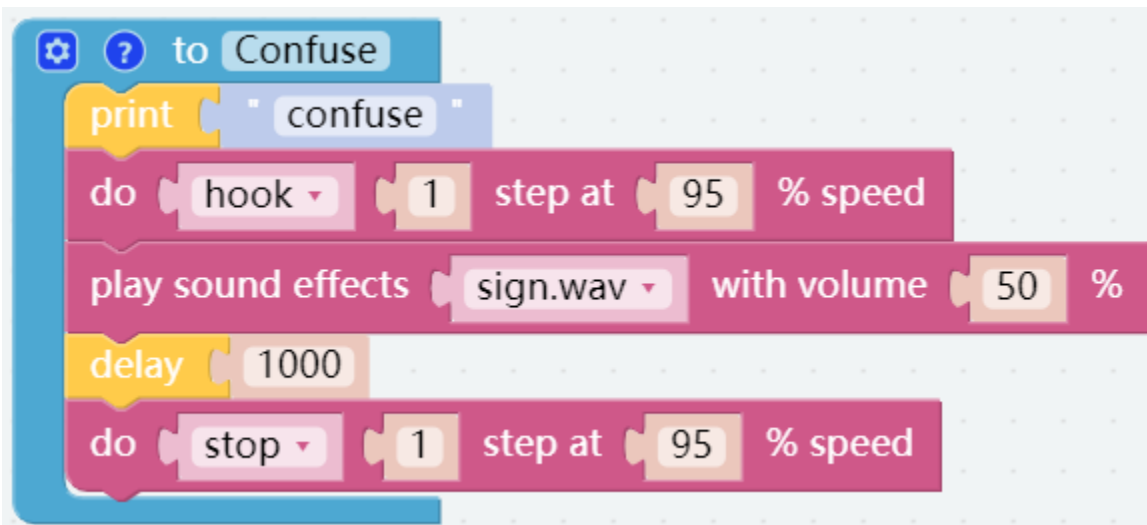


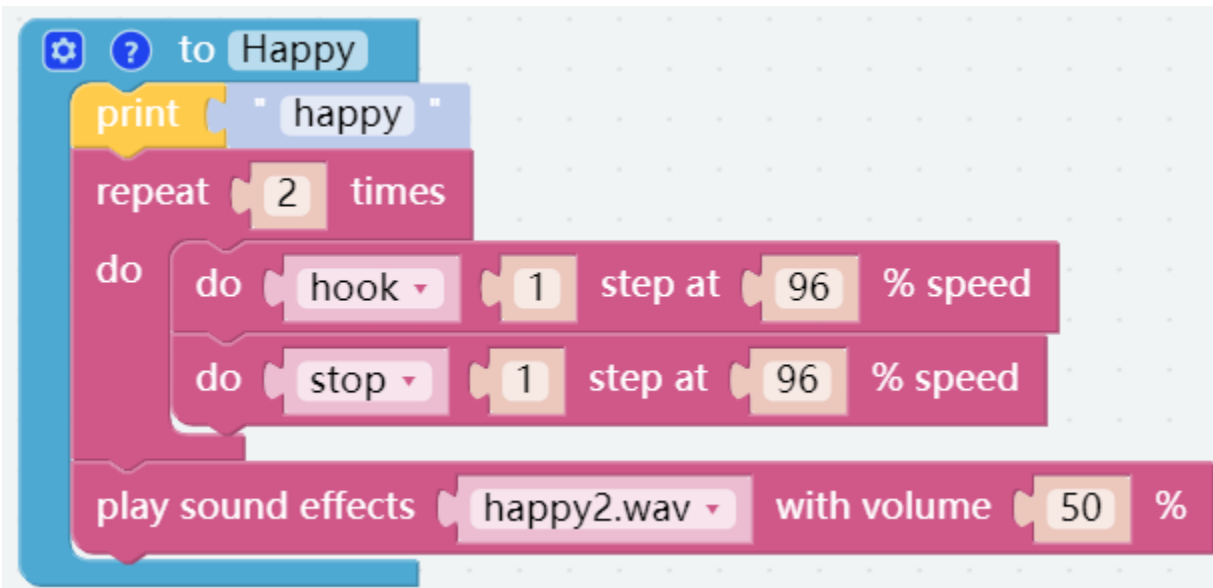
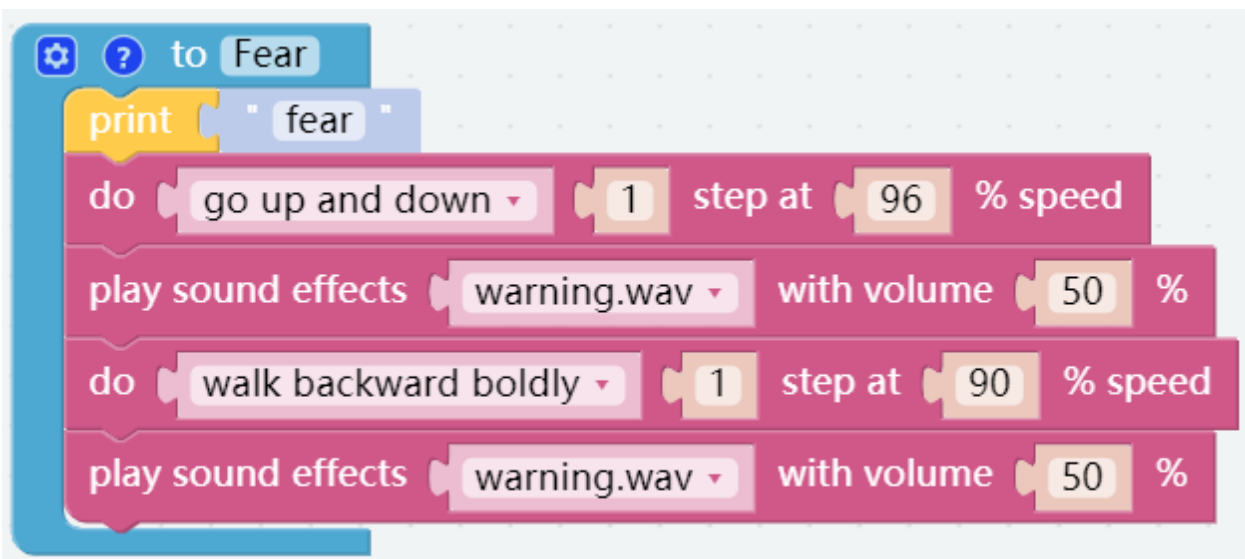
The **Function** block without output is used here.

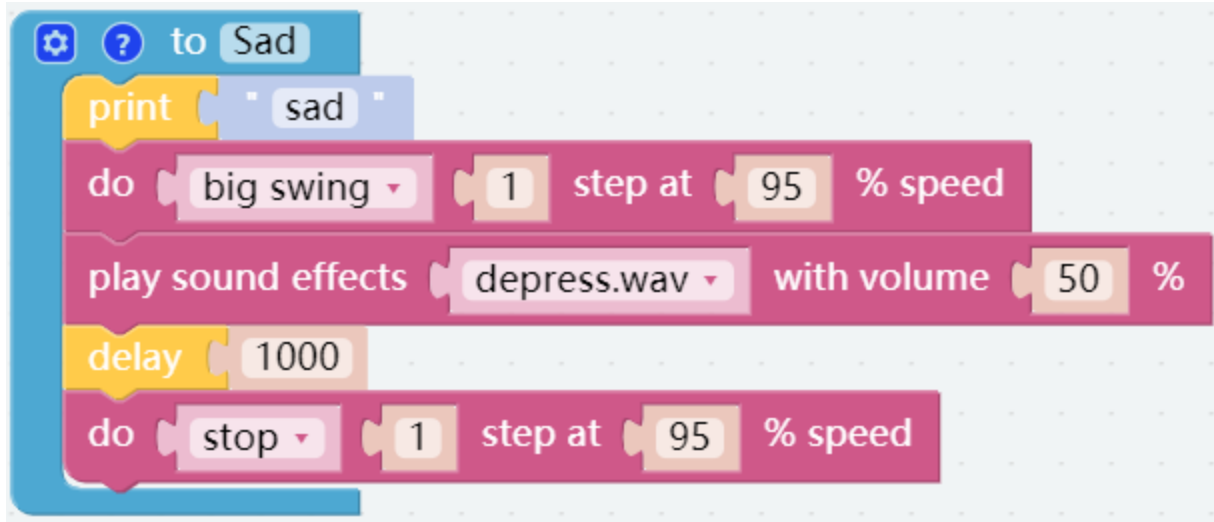
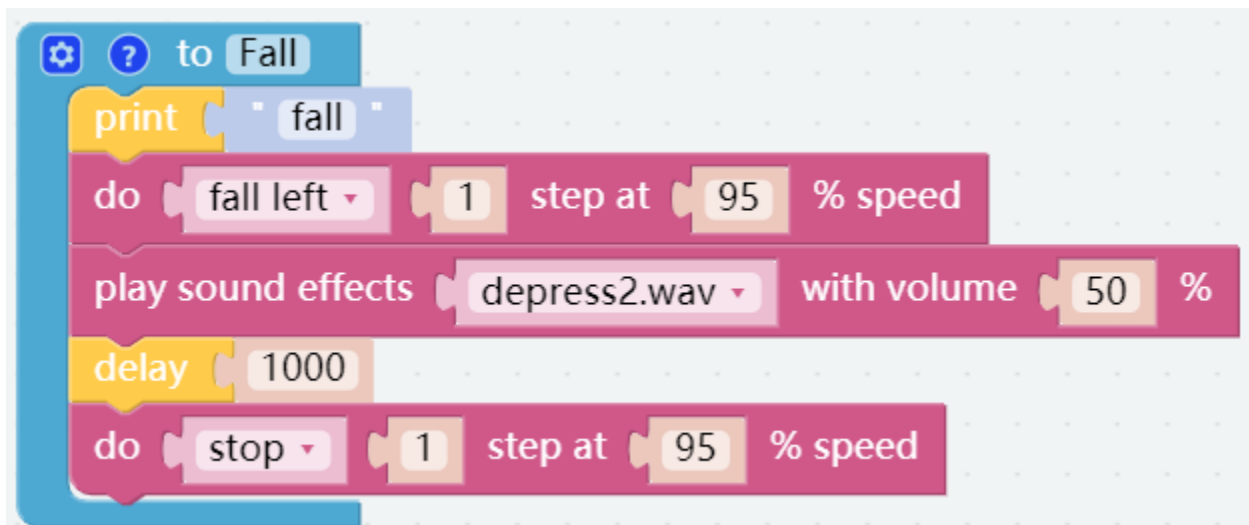


EXAMPLE

Shy

**Confuse****Happy**

**Fear****Sad**

**Fall**

Call all custom functions in the Forever block.



3.6 Dance

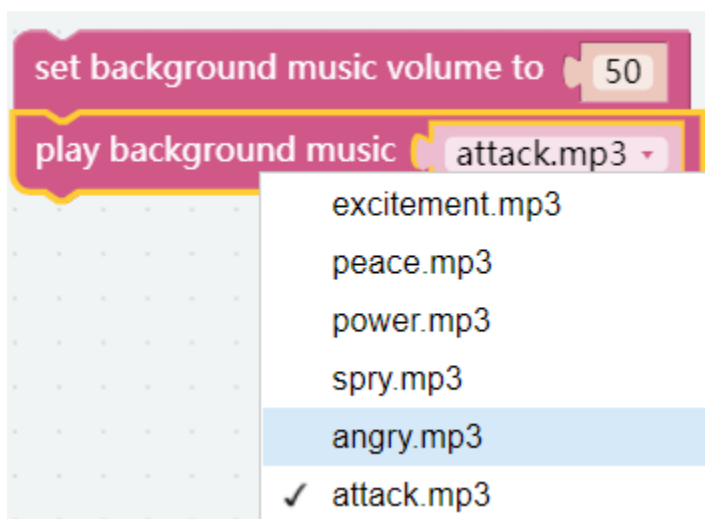
Now, PiSloth will show you his newly learned dance.



Note: You can download and print the [PDF Cartoon Mask](#) for your PiSloth.

TIPS

In addition to having PiSloth play sound effects and speak, it can also play set background music, and the volume of the background music can be adjusted (0%-100%).



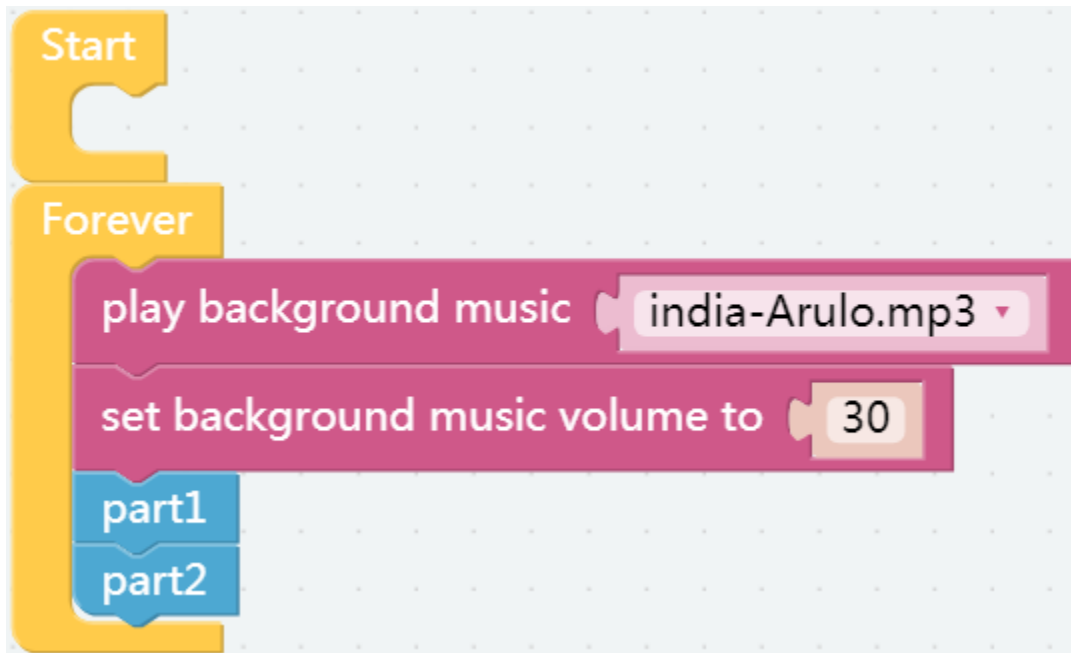
Repeat block can help you execute the same code multiple times to reduce code size.

**EXAMPLE**

The whole dance is divided into 2 parts, and PiSloth will finish these 2 parts with the music. If you don't pause the code, it will repeat the dance.

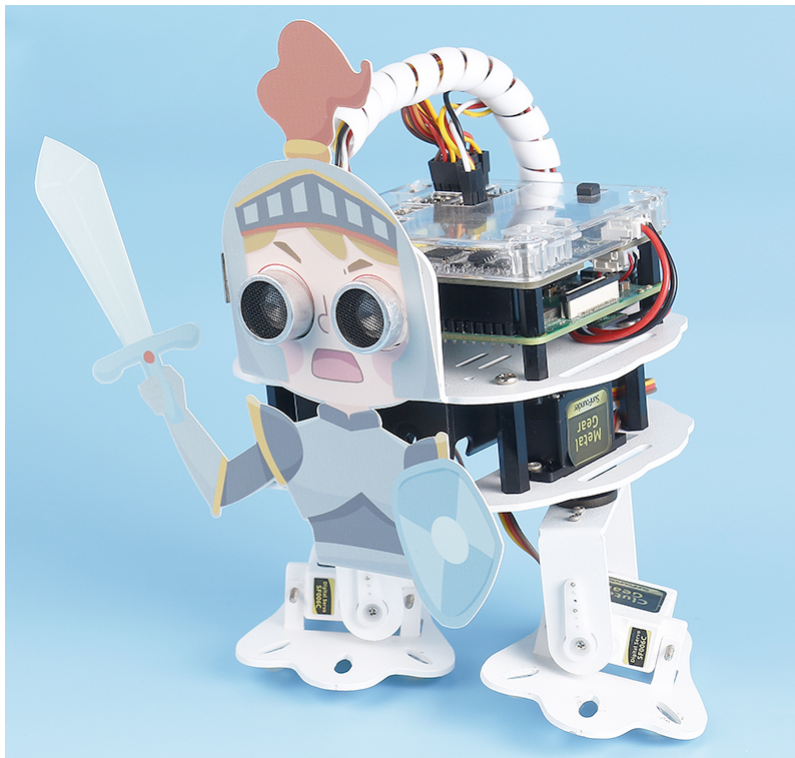






3.7 Let's Fight! Warrior!

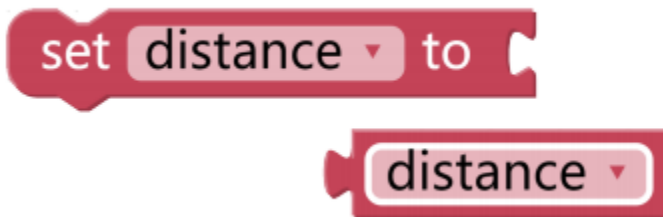
Here, PiSloth is a brave warrior, when it appears in front of the enemy, it will let out a roar and rush to the enemy.



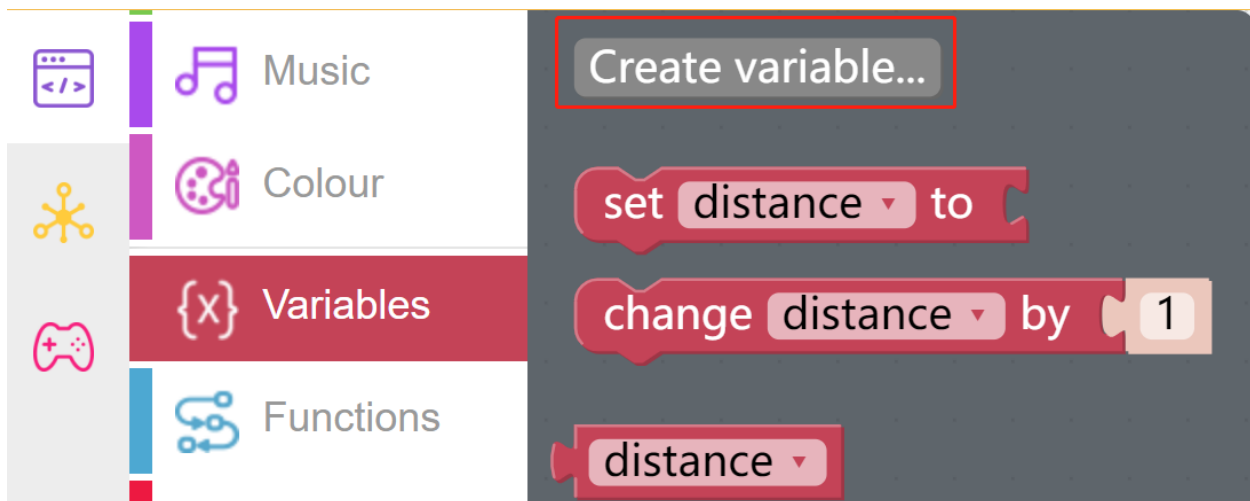
Note: You can download and print the [PDF Cartoon Mask](#) for your PiSloth.

TIPS

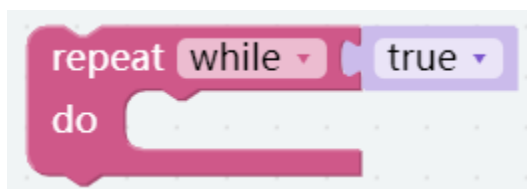
You may want to simplify your program with Variable. For example, when you have multiple functions that need to read the obstacle distance, you don't need to read the value for each function, just load the value into a variable and use it multiple times.



Click the **Create variable** button on the **Variables** category to create a variable named distance.

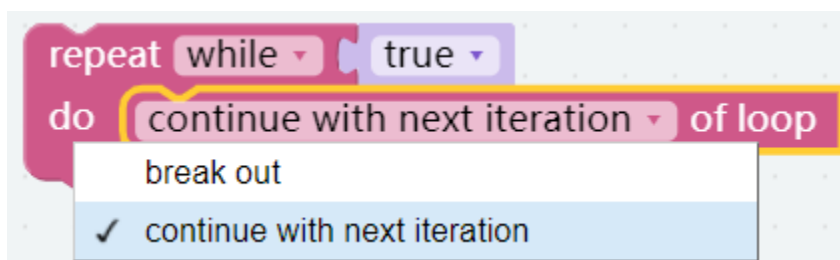


You can use this block to set up an endless loop.



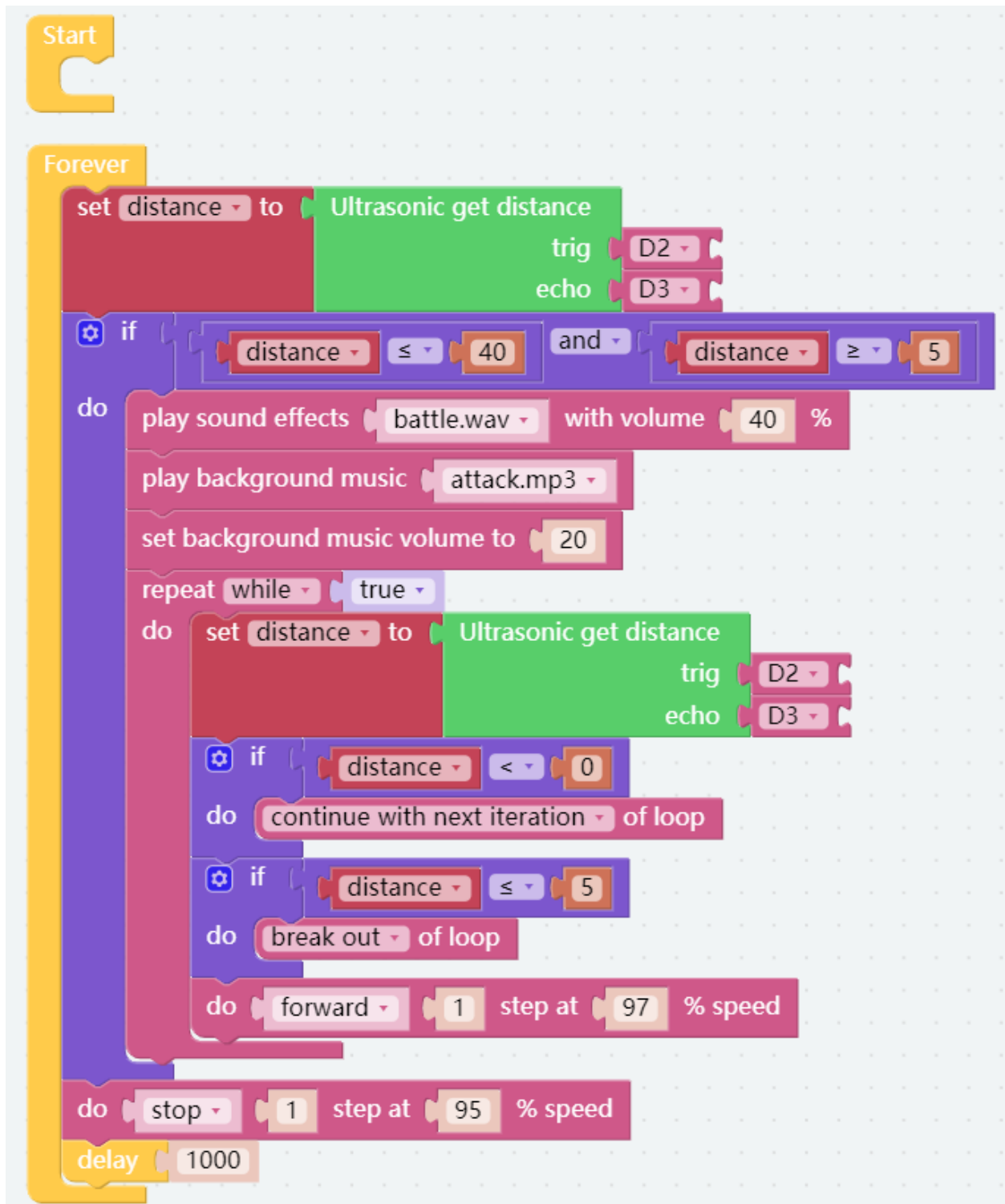
This is a block that jumps out of the loop, and it has two options and can be only used within a loop.

- **break out:** Jump out of the entire loop.
- **continue with next iteration:** Jump out of the current loop and enter the next loop.

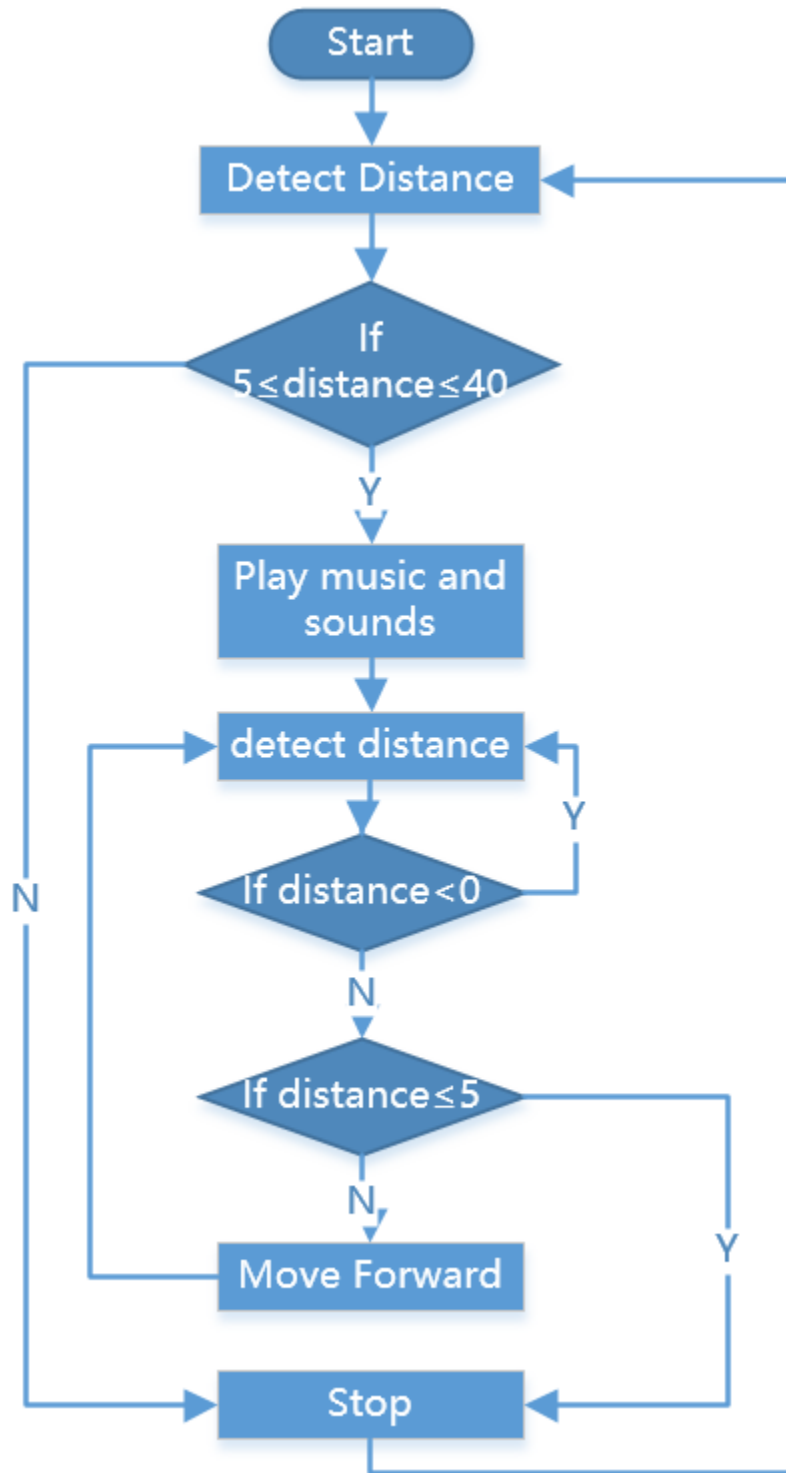


EXAMPLE

After the code is run, PiSloth will continuously detect the distance of the obstacle, when the distance is between 5 and 40, PiSloth will make a roaring sound and rush forward; when the distance of the obstacle is less than 5, PiSloth will stop.



Flow Chart



3.8 Remote Control

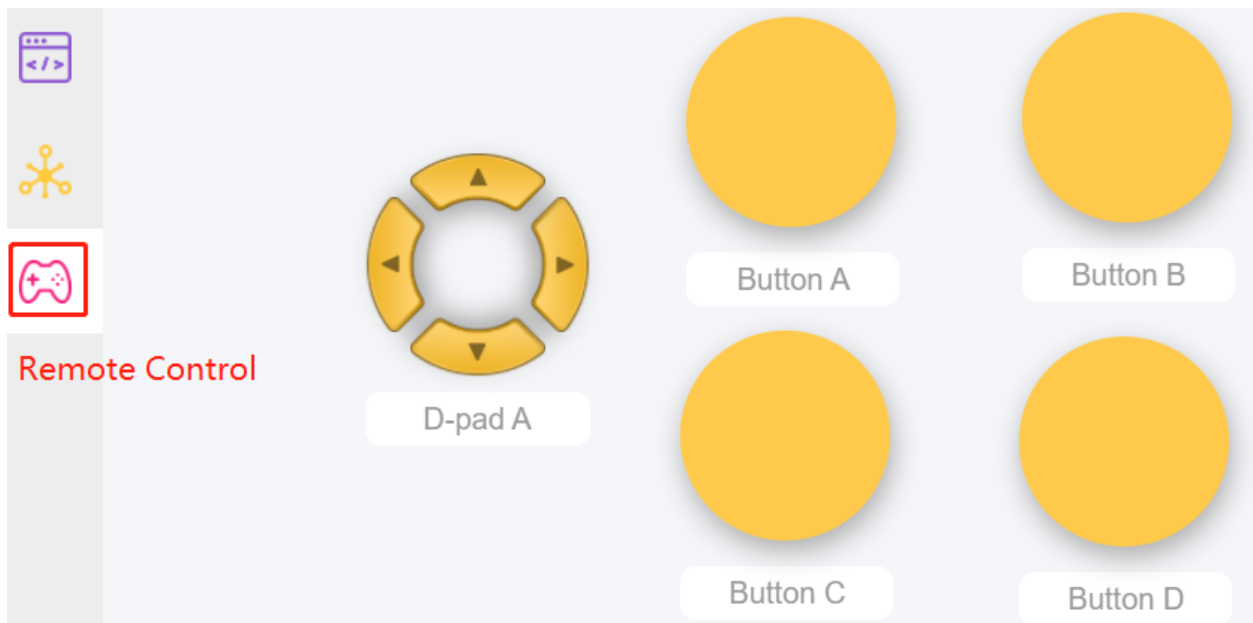
You can also use the widgets on Ezblock Studio to make PiSloth move.



- How to Use the Remote Control Function?

TIPS

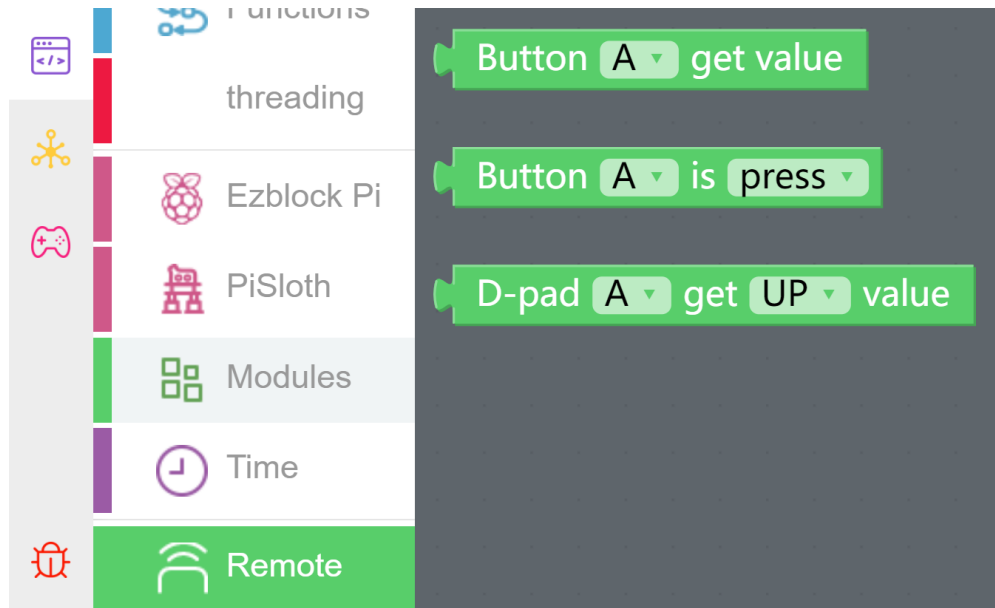
To use the remote control function, you need to enter the **Remote Control** page from the left side of main page, and then drag one D-pad and 4 buttons to the central area.



Back in the programming page, you will see an additional Remote category, and the D-pad and Button block appear

in it.

- **Button () get value:** This block is used to read the value of the buttons, if the button is pressed, the value is 1, otherwise it is 0.
- **Button () is (press/release):** This block and Button () get value = (0/1) have the same effect and can be used directly to determine whether a button is pressed or not.
- **D-pad () get () value:** This block is used to read the up/down/left/right (selected through the drop-down menu) pad values, press for 1 and release for 0.



EXAMPLE

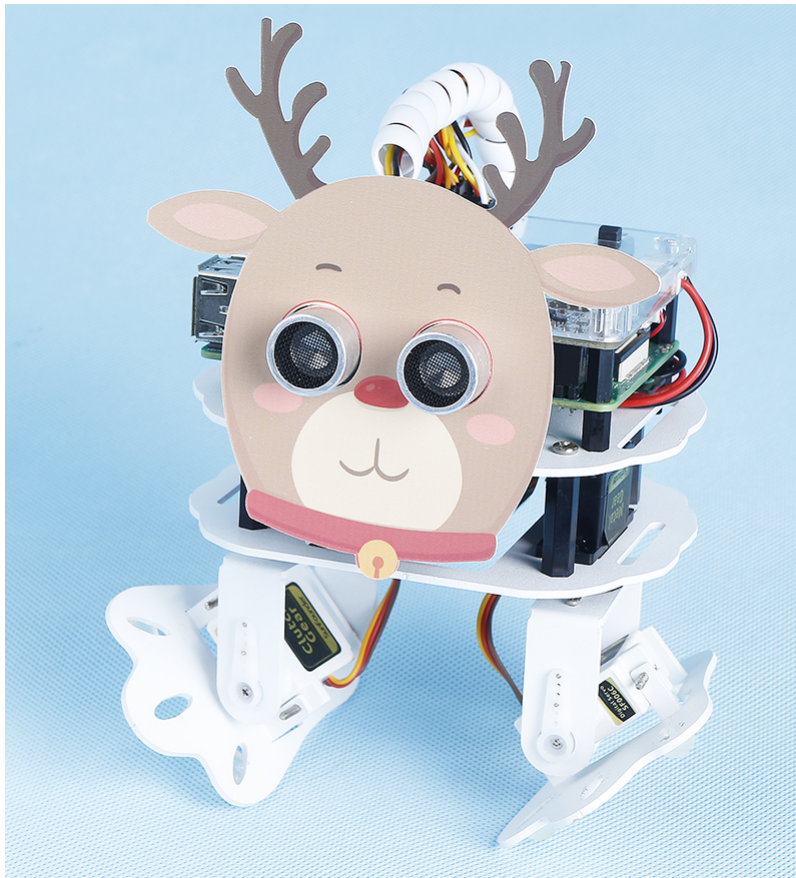


3.9 Custom Step

In the previous projects, we used a lot of actions that we wrote, so how are these actions composed and done? Generally speaking, an action is composed of one or more steps.

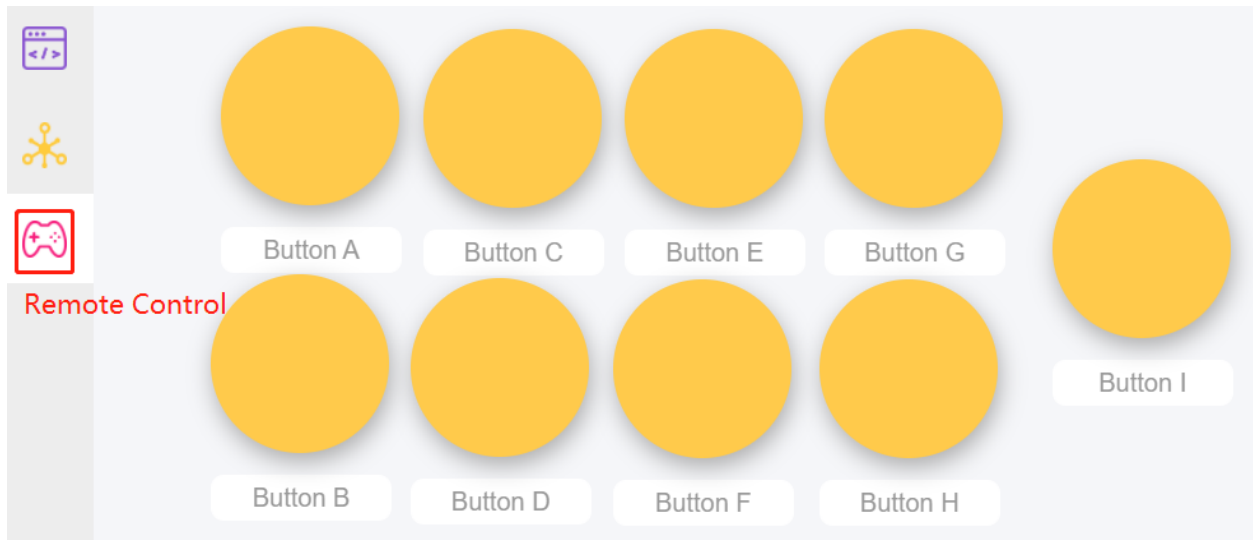
In this project, we will learn how to customize PiSloth's step.

All we have to do is to use the buttons in the remote control page to make PiSloth complete the step shown in the figure below, and then get the angles of the 4 Servos at that time.

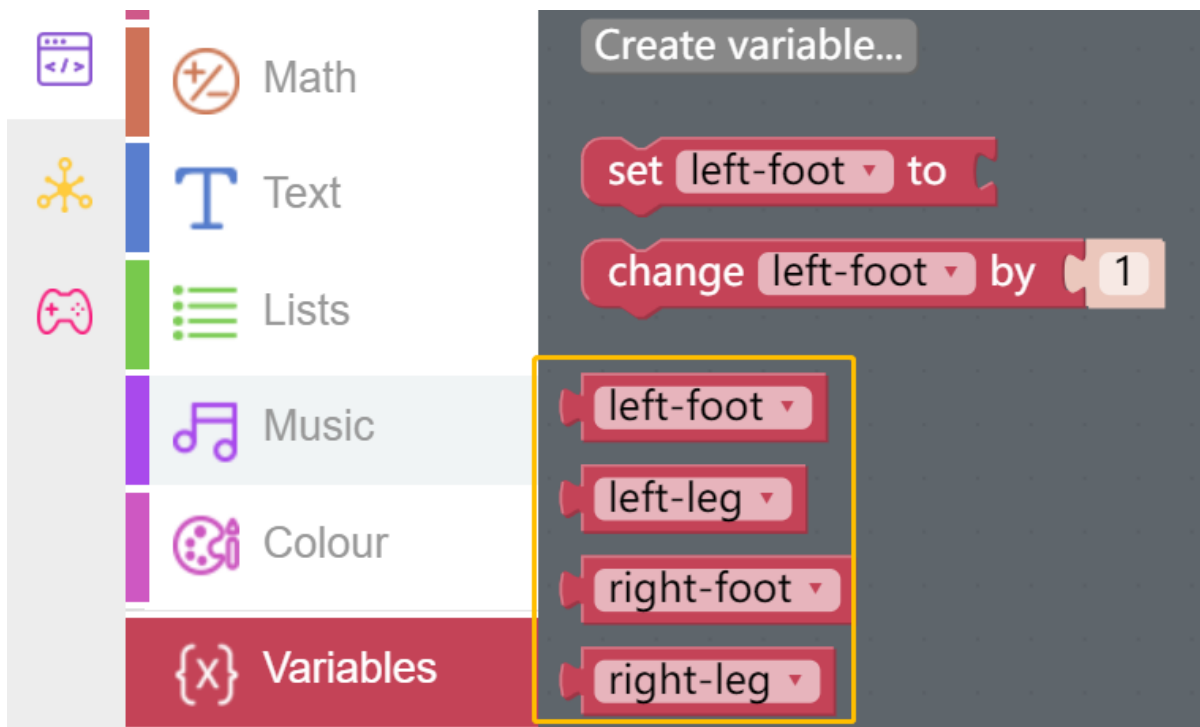


Note: You can download and print the [PDF Cartoon Mask](#) for your PiSloth.

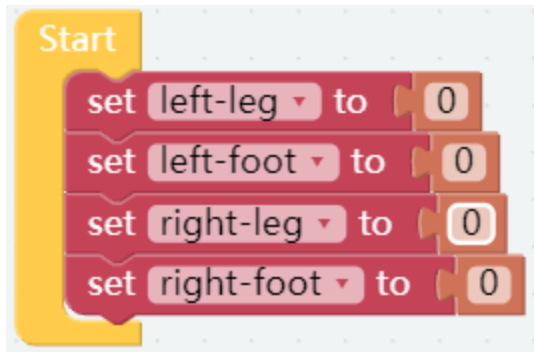
Step 1: Drag out 9 buttons in the Remote Control to control the rotation angles of the 4 Servos on the PiSloth.



Step 2: Create 4 variables to store the angles of the 4 Servos.

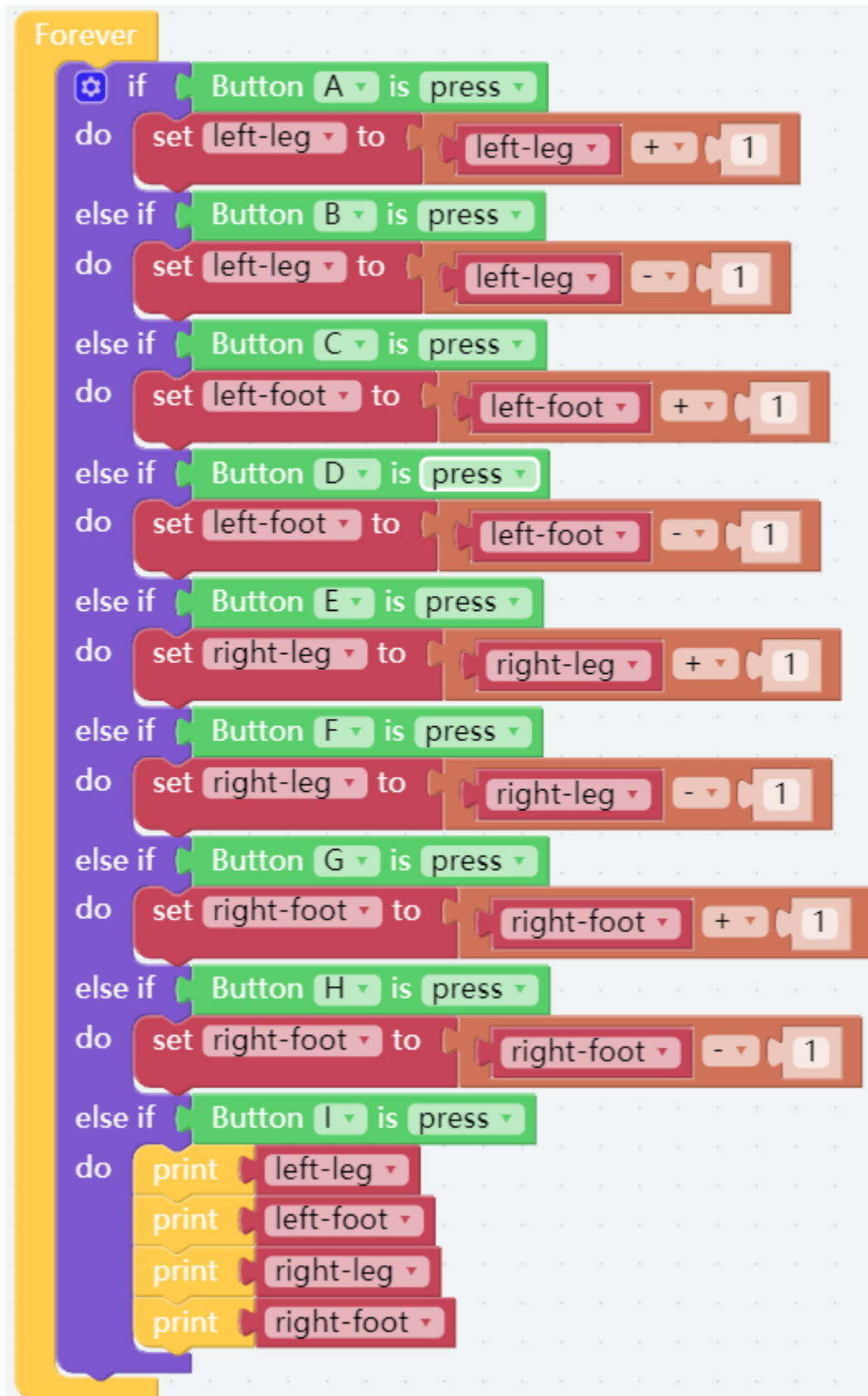


Then initialize the angle to 0.

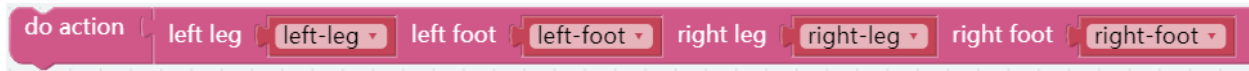


Step 3: Reads the values of the different buttons that are used to control the angles of the Servos.

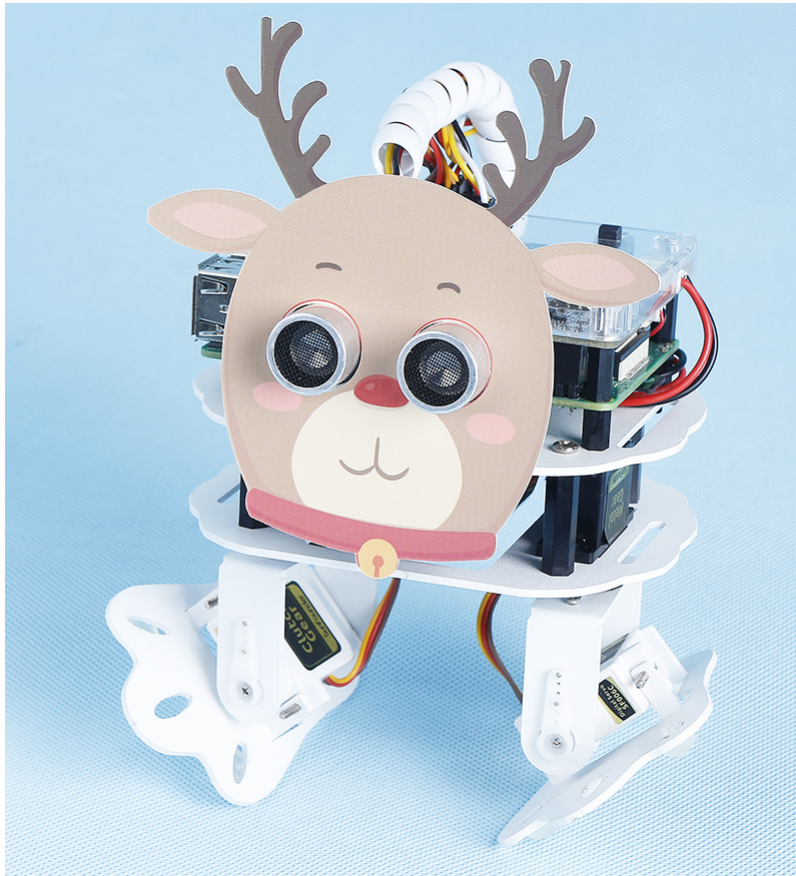
- **button AB** control the **left-leg**.
- **button CD** control the **left-foot**.
- **button EF** control the **right-leg**.
- **button GH** control the **right-foot**.
- Press **button I** and the angles of the 4 Servos will be printed in the Debug Monitor.



Step 4: At the end of the Forever block, fill in the angle values read into the 4 servos and use the **do action** block to make PiSloth do this step.

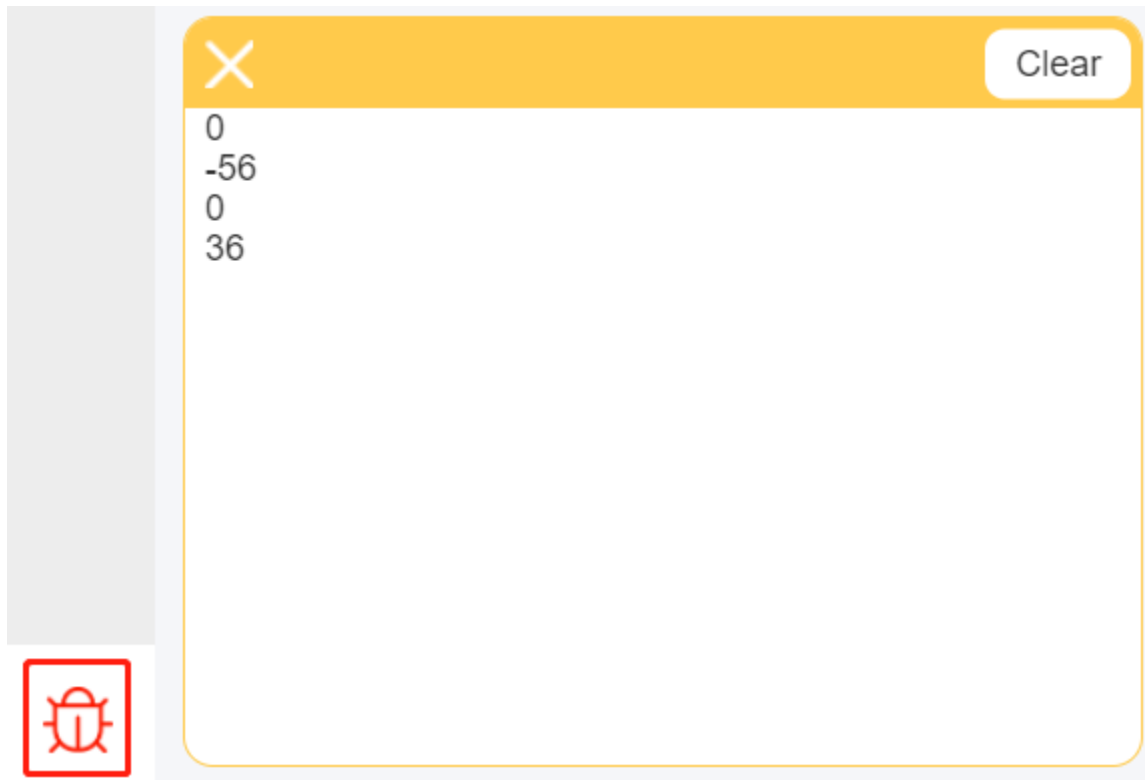


Step 5: Once the code is complete, click the **download** icon in the bottom right corner to download and run the code. Now we can click **button CD** and **button GH** (according to the actual code) to make PiSloth pose like this, you can also make it do other steps.

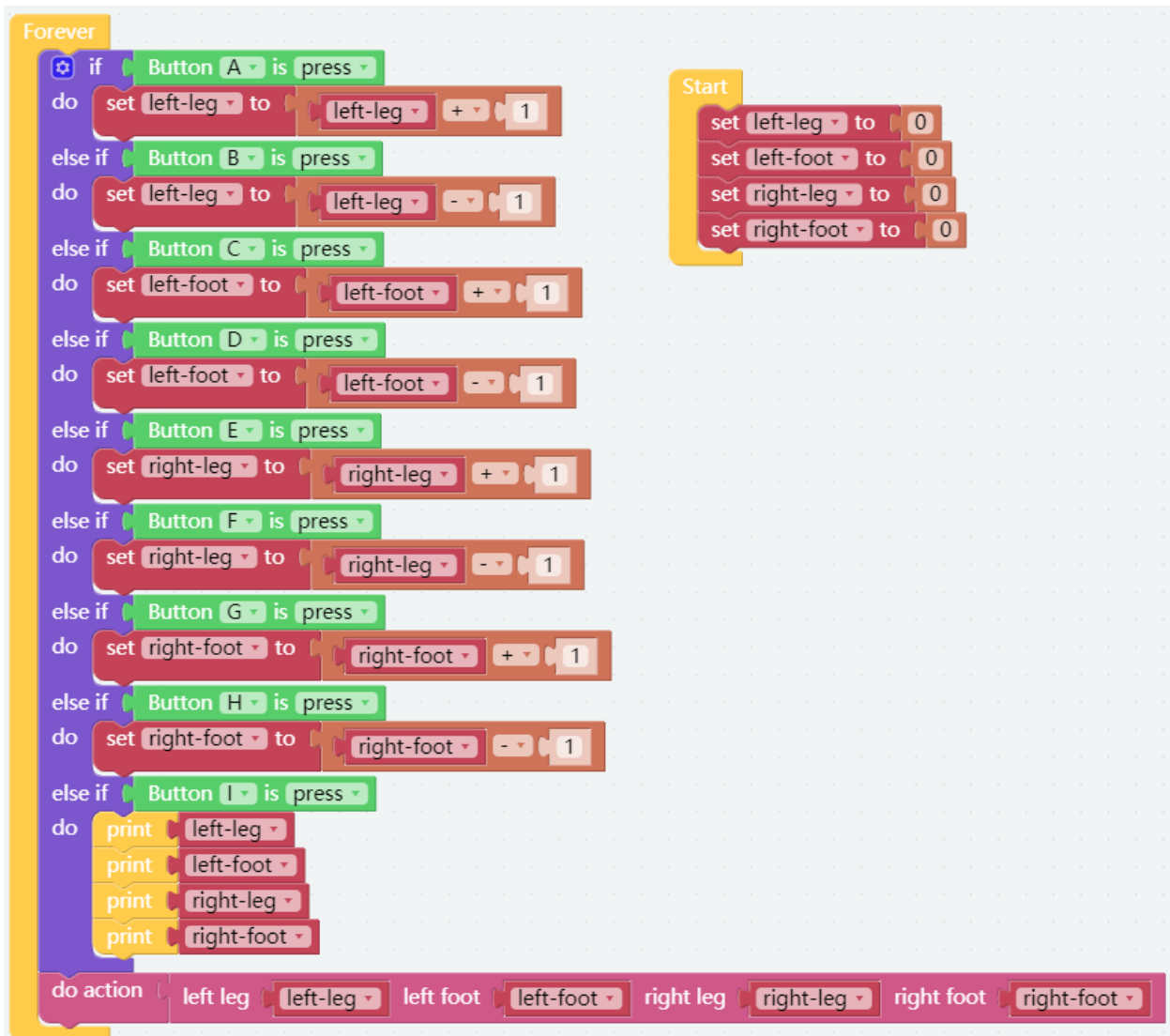


Step 6: Click on the Debug Monitor icon in the bottom left corner, and you will see the angle of the 4 servos in the Debug Monitor at that moment when you press **button I**.

Note: Some times more than 2 sets of data may appear because if you click **button I** for a little longer, Ezblock will think **button I** was clicked 2 times. You can clear the data and click button I again.



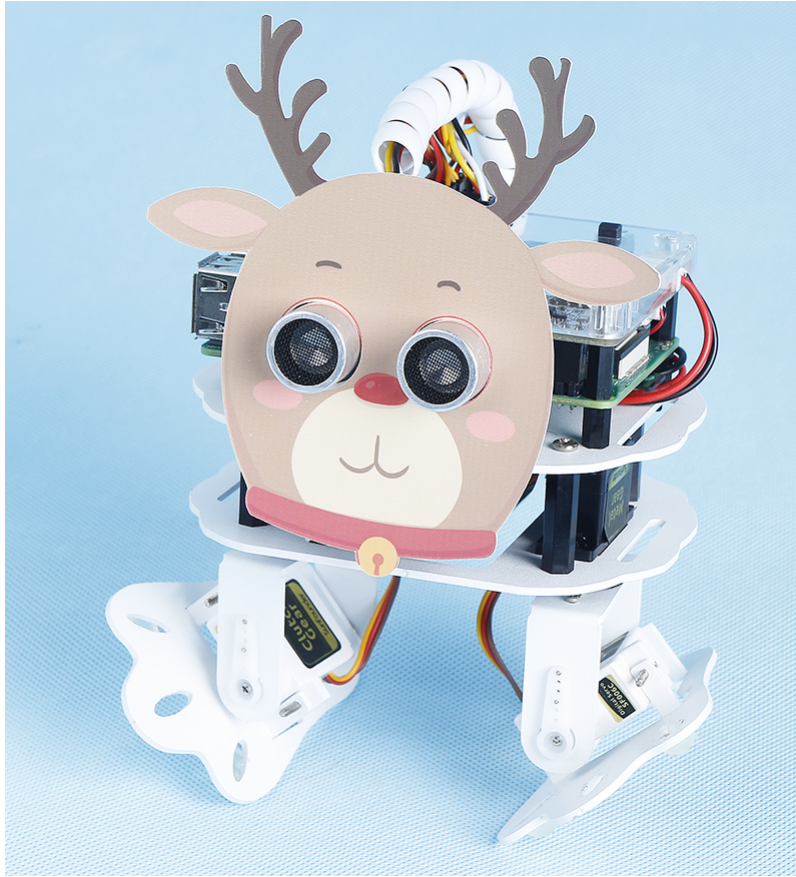
The complete code is as follows:



3.10 Custom Action

In the previous project, we were able to give PiSloth custom steps, so how do we combine these steps into actions?

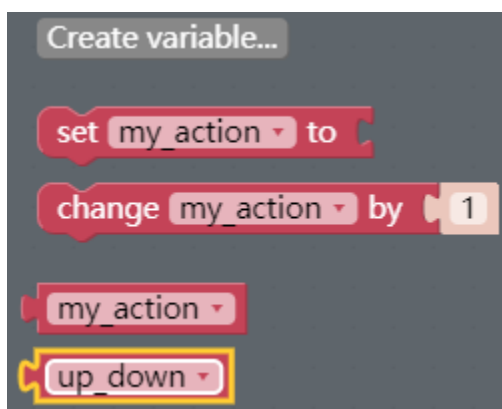
For example, have PiSloth make the step from the previous project and then return to the initial position.



Note: You can download and print the [PDF Cartoon Mask](#) for your PiSloth.

TIPS

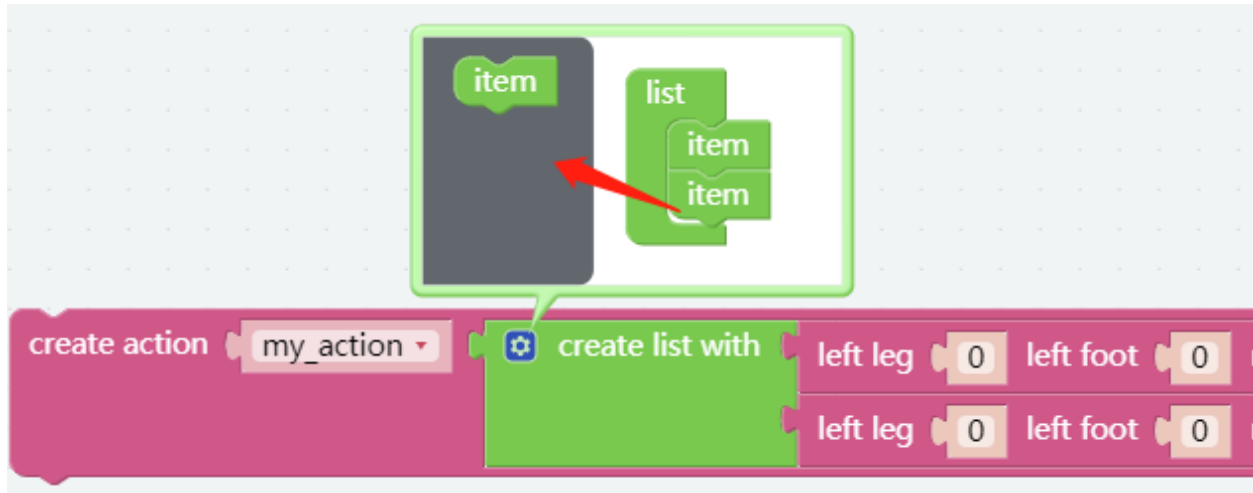
Create a variable **up_down** to store this action.



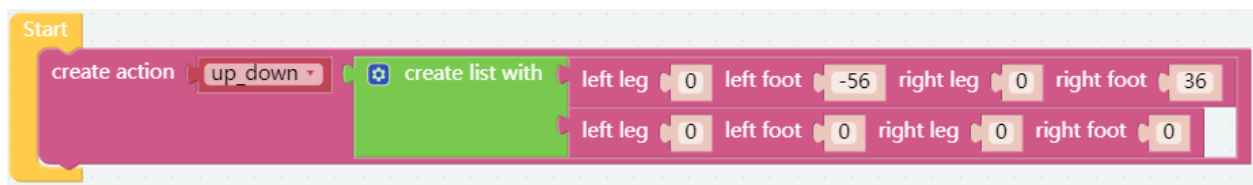
You can use the **create action** block to make PiSloth do this action. These values represent the angles of the 4 Servos on the PiSloth. The range is (-90~90).



Here you can increase or decrease the number of items by dragging it.



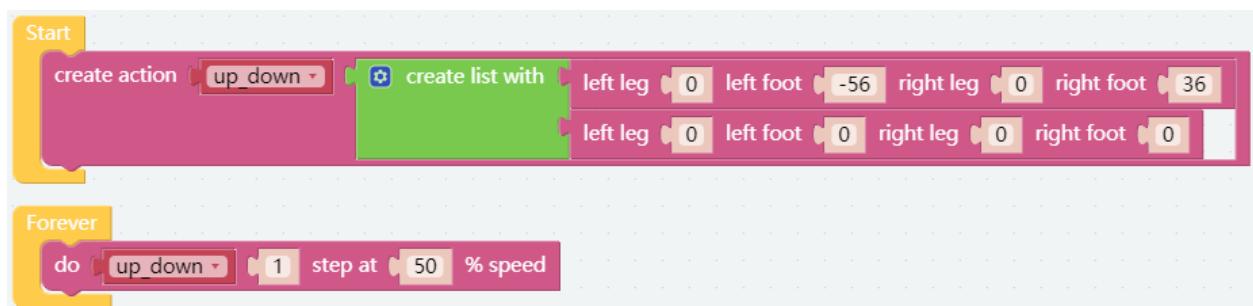
Fill in the angle obtained in the previous project and name this action **up_down** (drag it from **Variables** category).



Use the **do** block to make PiSloth do this action once at 50% speed.



EXAMPLE



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](#).

4.1 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.

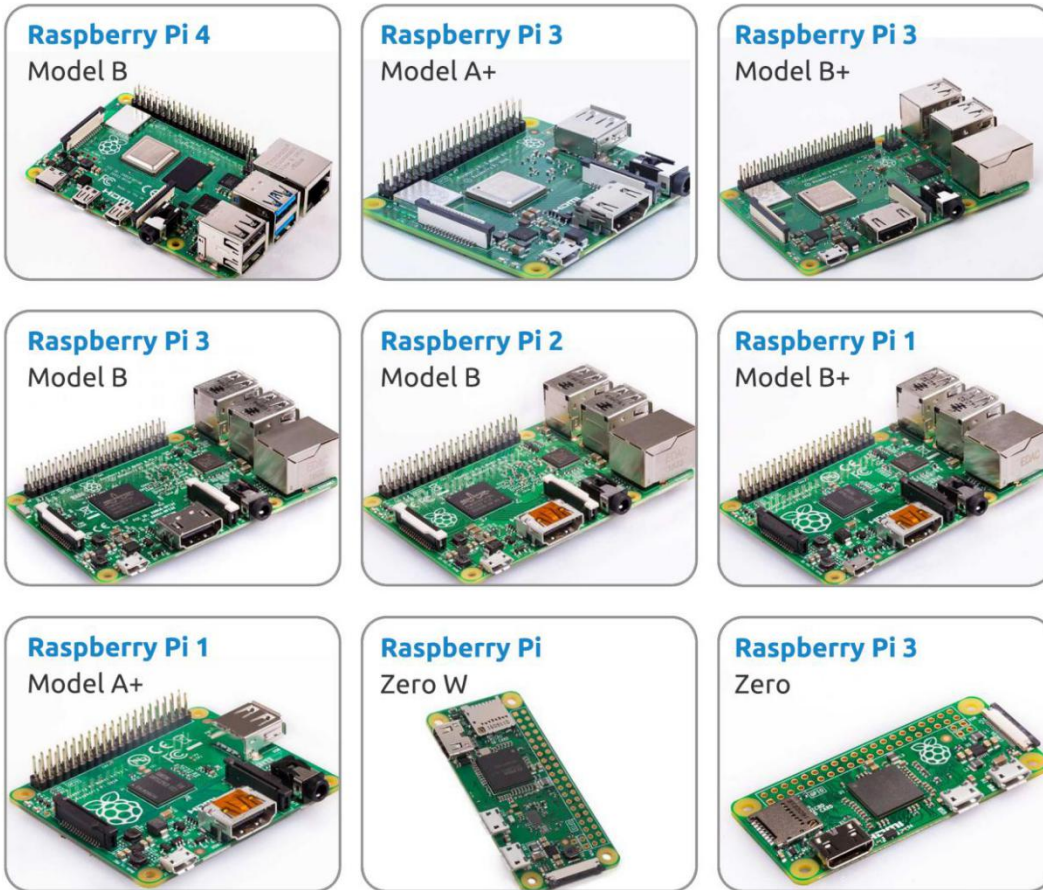
4.1.1 What Do We Need?

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.

4.1.2 Installing the OS

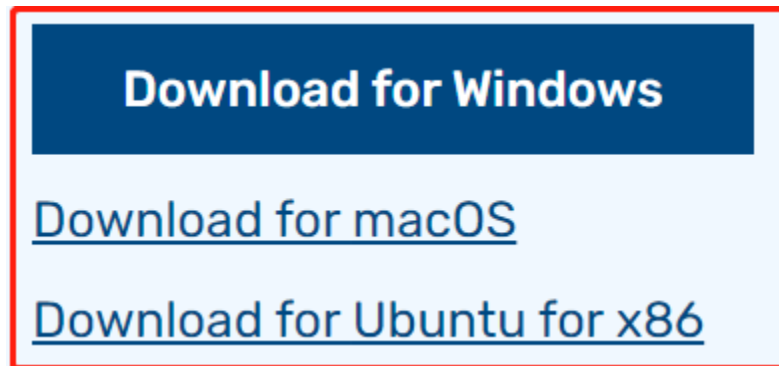
Required Components

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.

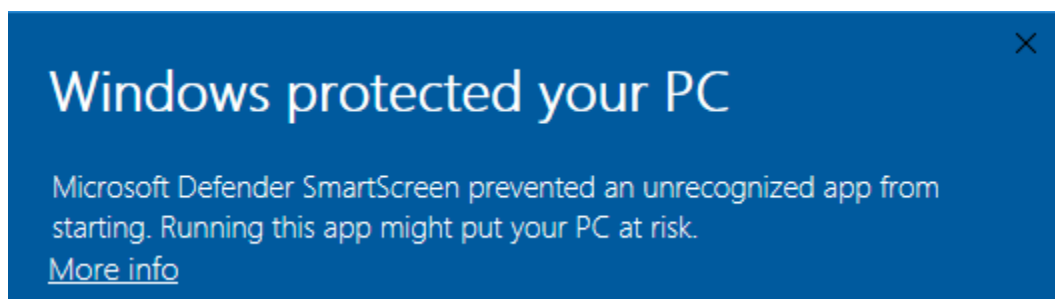
Visit the download page: <https://www.raspberrypi.org/software/>. Click on the link for the Raspberry Pi Imager that matches your operating system, when the download finishes, click it to launch the installer.



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.

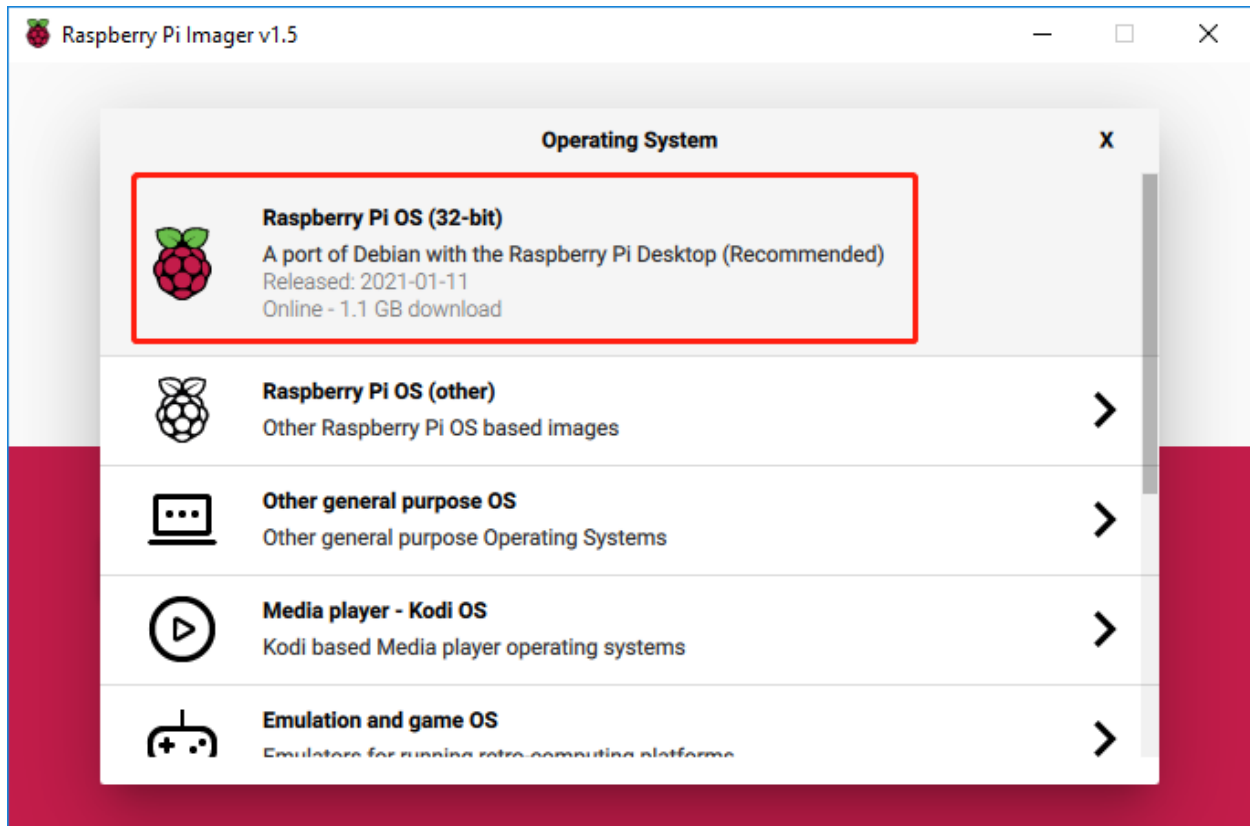


Step 3

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

Step 4

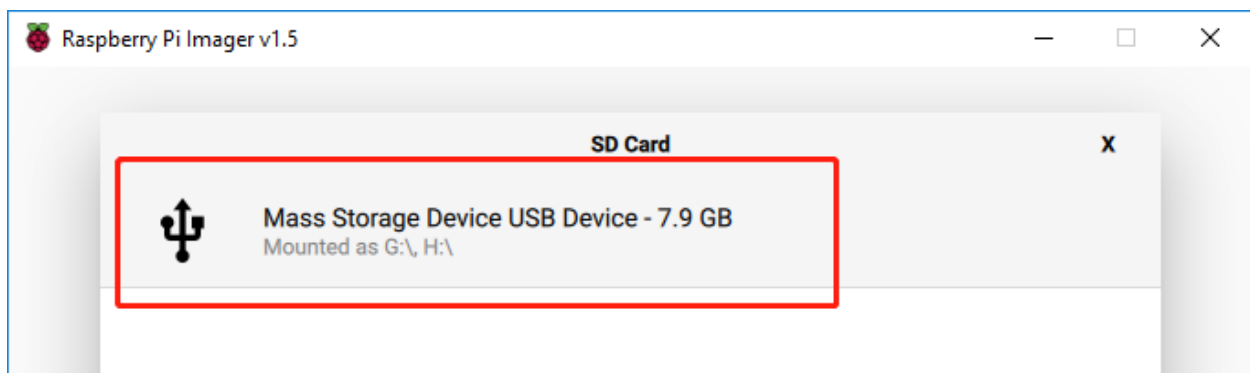
In the Raspberry Pi Imager, select the OS that you want to install and the SD card you would like to install it on.

**Note:**

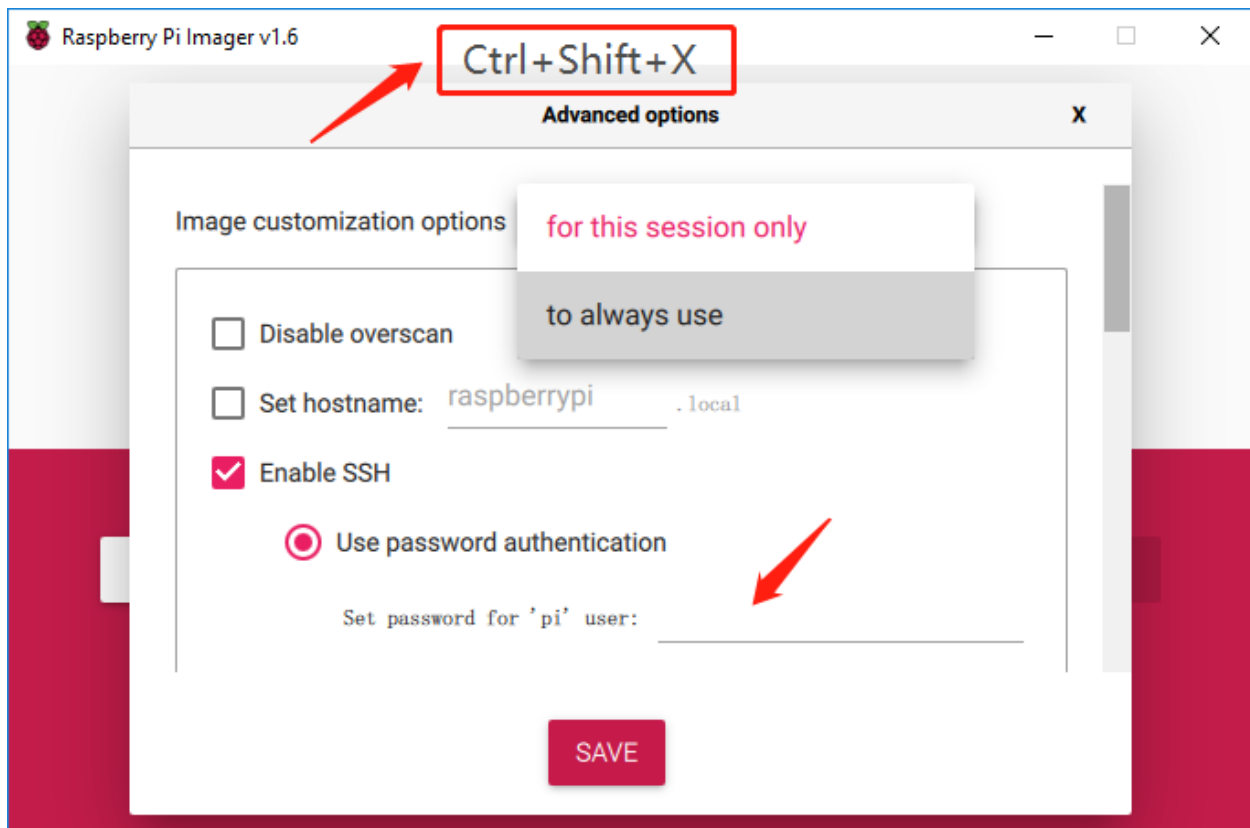
- 1) You will need to be connected to the internet the first time.
- 2) That OS will then be stored for future offline use (lastdownload.cache, C:/Users/yourname/AppData/Local/Raspberry Pi/Imager/cache). So the next time you open the software, it will have the display “Released: date, cached on your computer”.

Step 5

Select the SD card you are using.

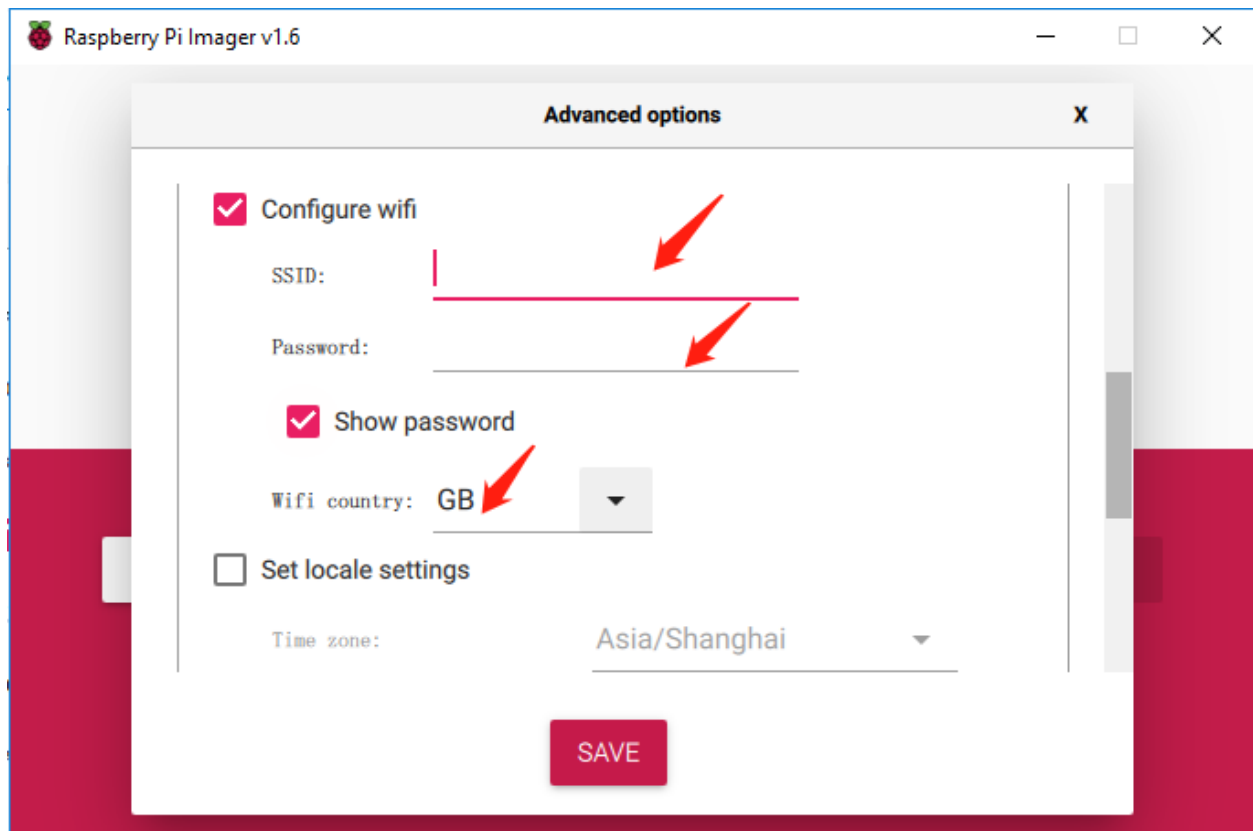
**Step 6**

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



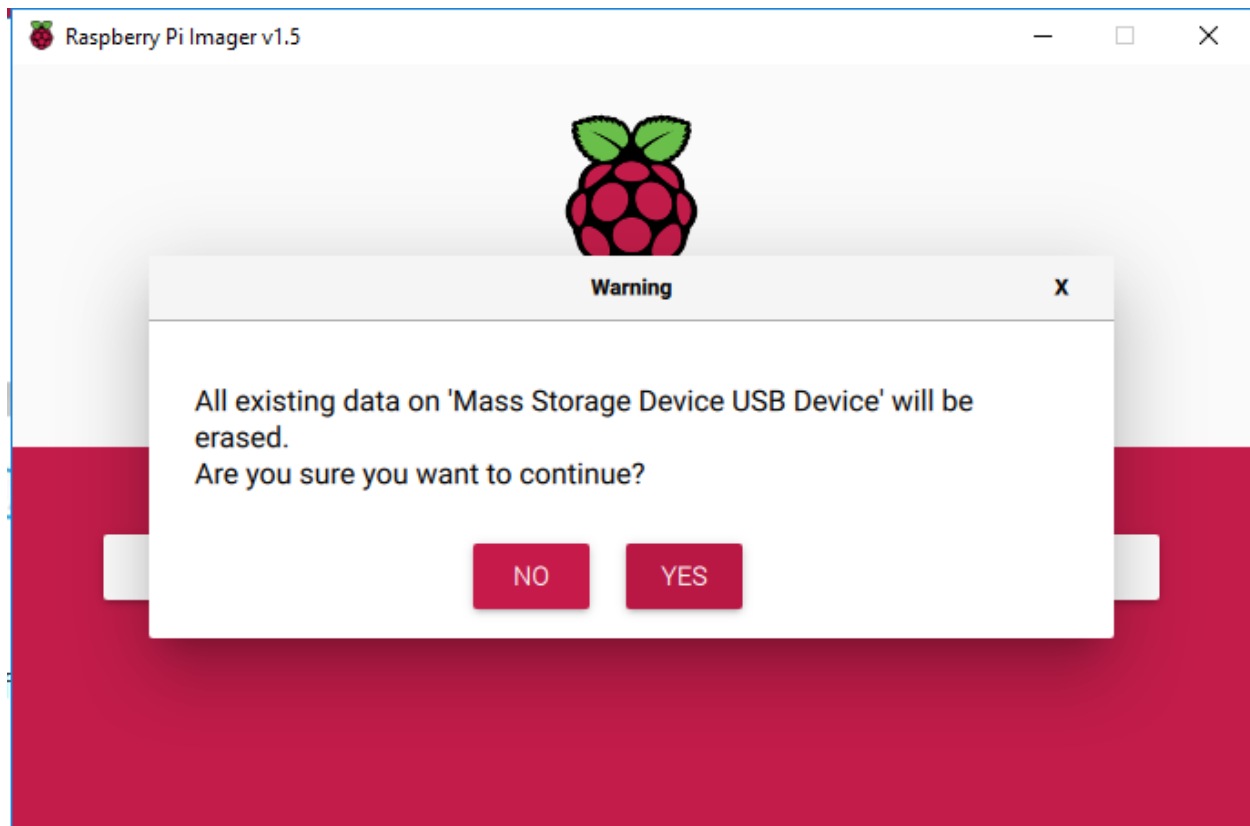
Step 7

Click the **WRITE** button.



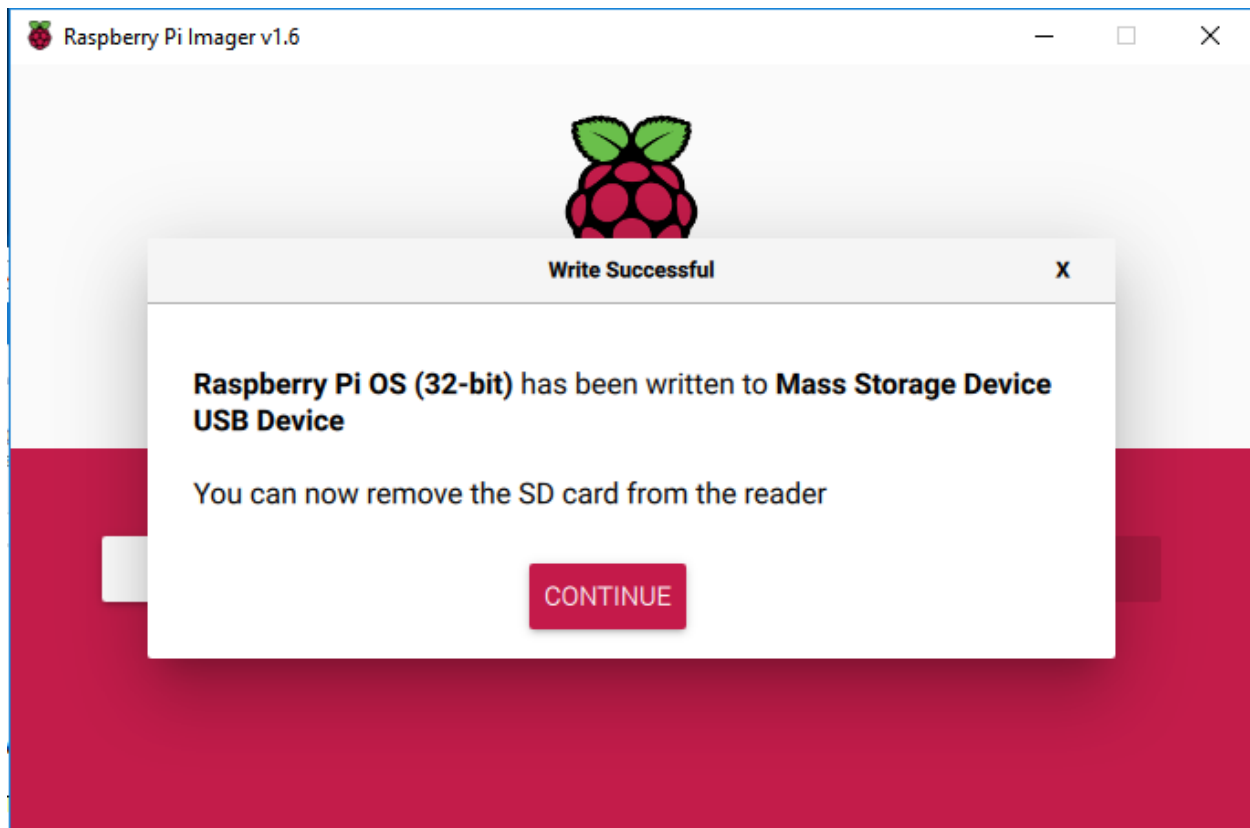
Step 8

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.



4.1.3 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.

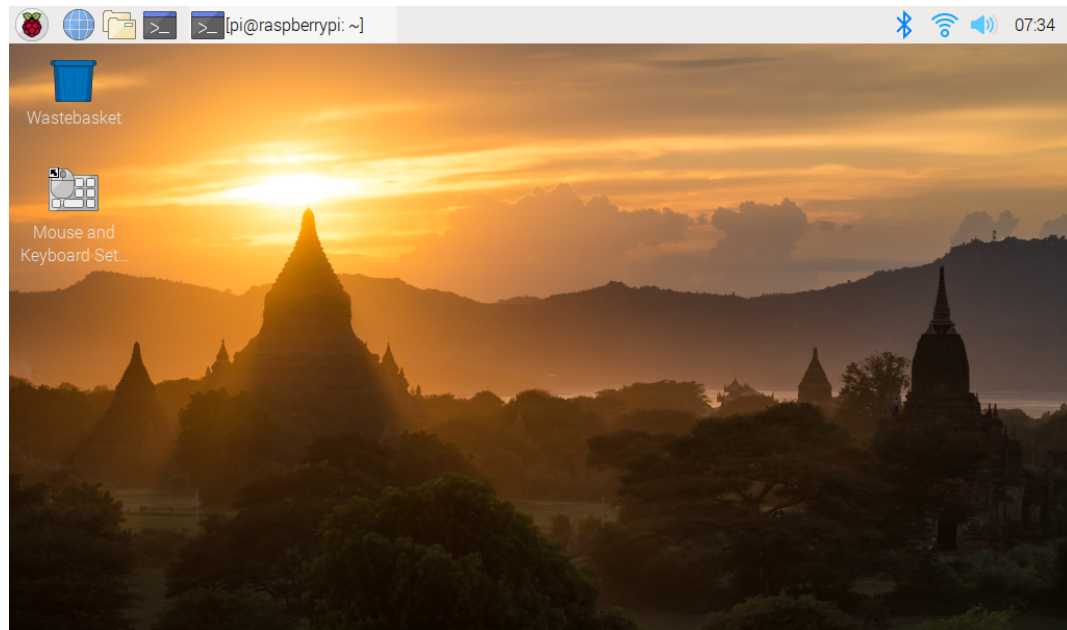
Required Components

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.

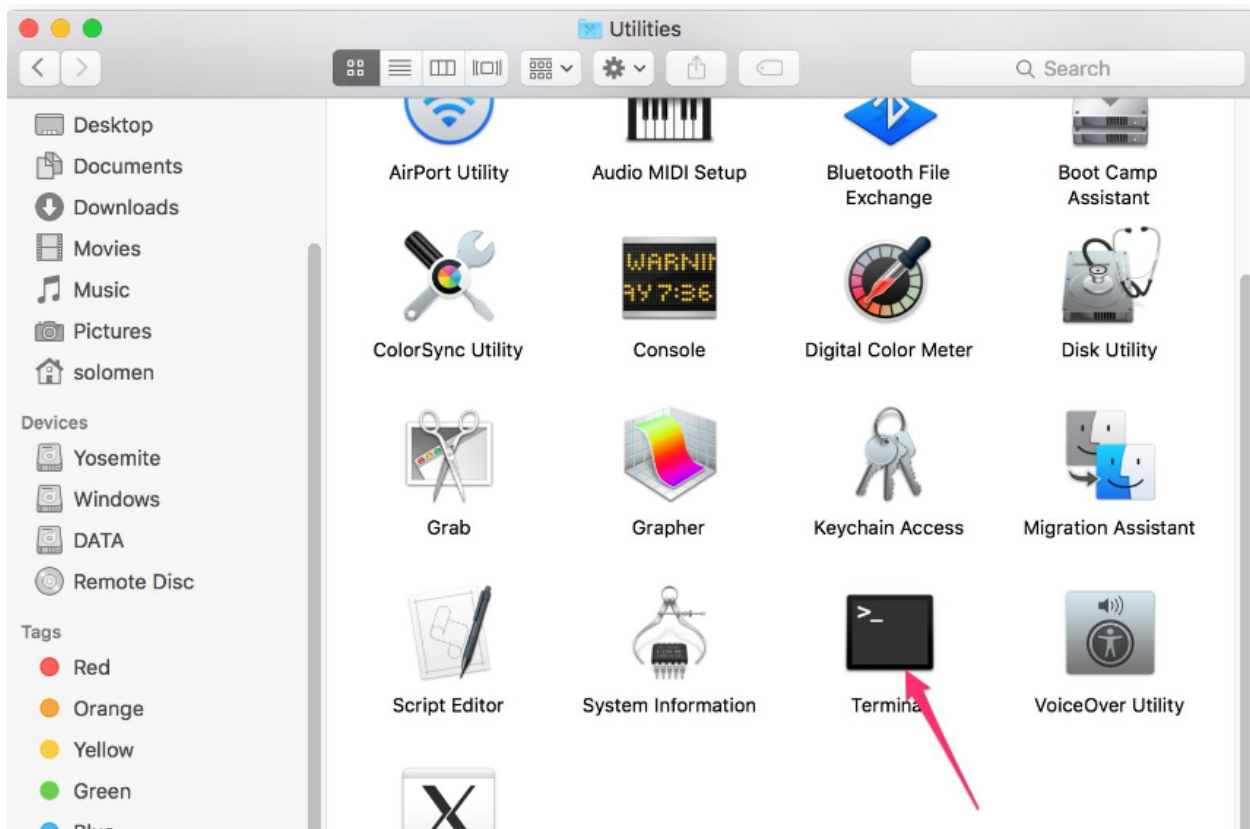
Use the SSH Remote Control

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.



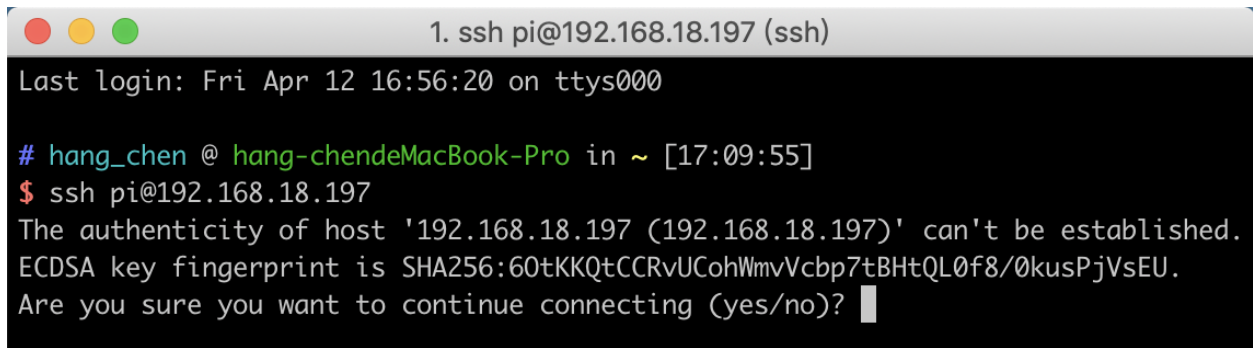
Step 2

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”.

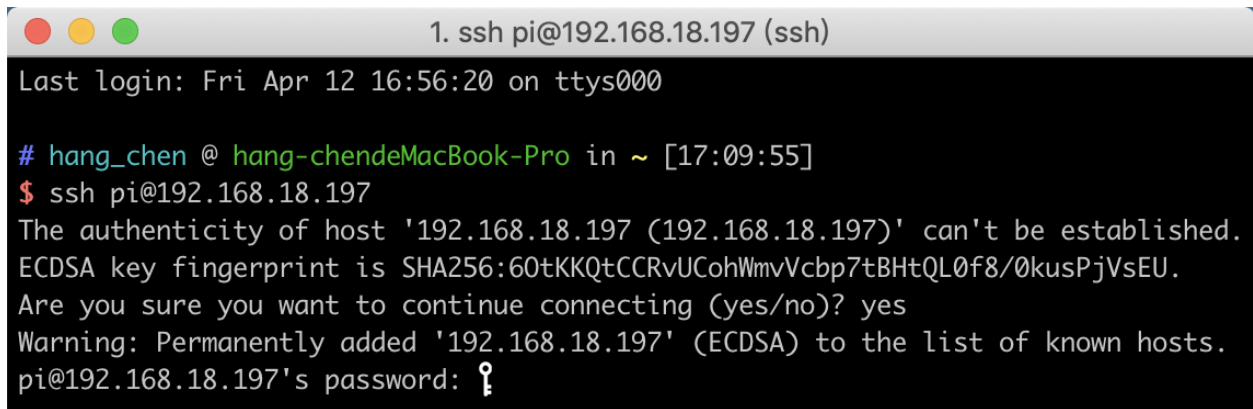


```
1. ssh pi@192.168.18.197 (ssh)
Last login: Fri Apr 12 16:56:20 on ttys000

# 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**.

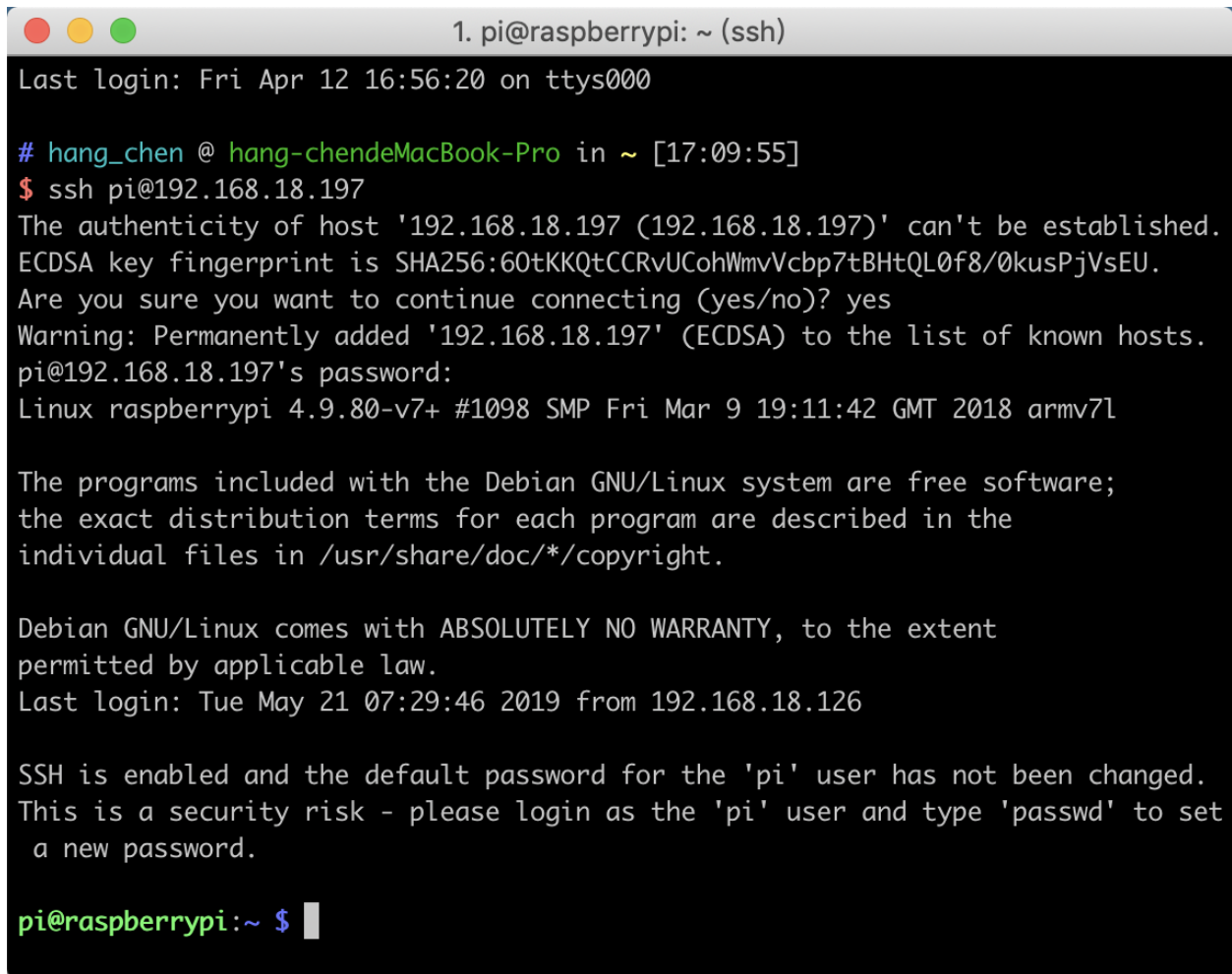


```
1. ssh pi@192.168.18.197 (ssh)
Last login: Fri Apr 12 16:56:20 on ttys000

# 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. The terminal shows a user 'hang_chen' at 'hang-chendeMacBook-Pro' initiating an SSH connection to 'pi@192.168.18.197'. It displays the host's ECDSA key fingerprint, asks for confirmation to continue, and shows the warning that the host has been added to the known hosts list. The password is entered (not visible), and the system version 'Linux raspberrypi 4.9.80-v7+ #1098 SMP Fri Mar 9 19:11:42 GMT 2018 armv7l' is shown. It then displays the Debian GNU/Linux license notice and the last login time. A message states that SSH is enabled but the default password for the 'pi' user has not been changed, advising a password change. The prompt 'pi@raspberrypi:~ \$' is shown at the bottom.

```
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

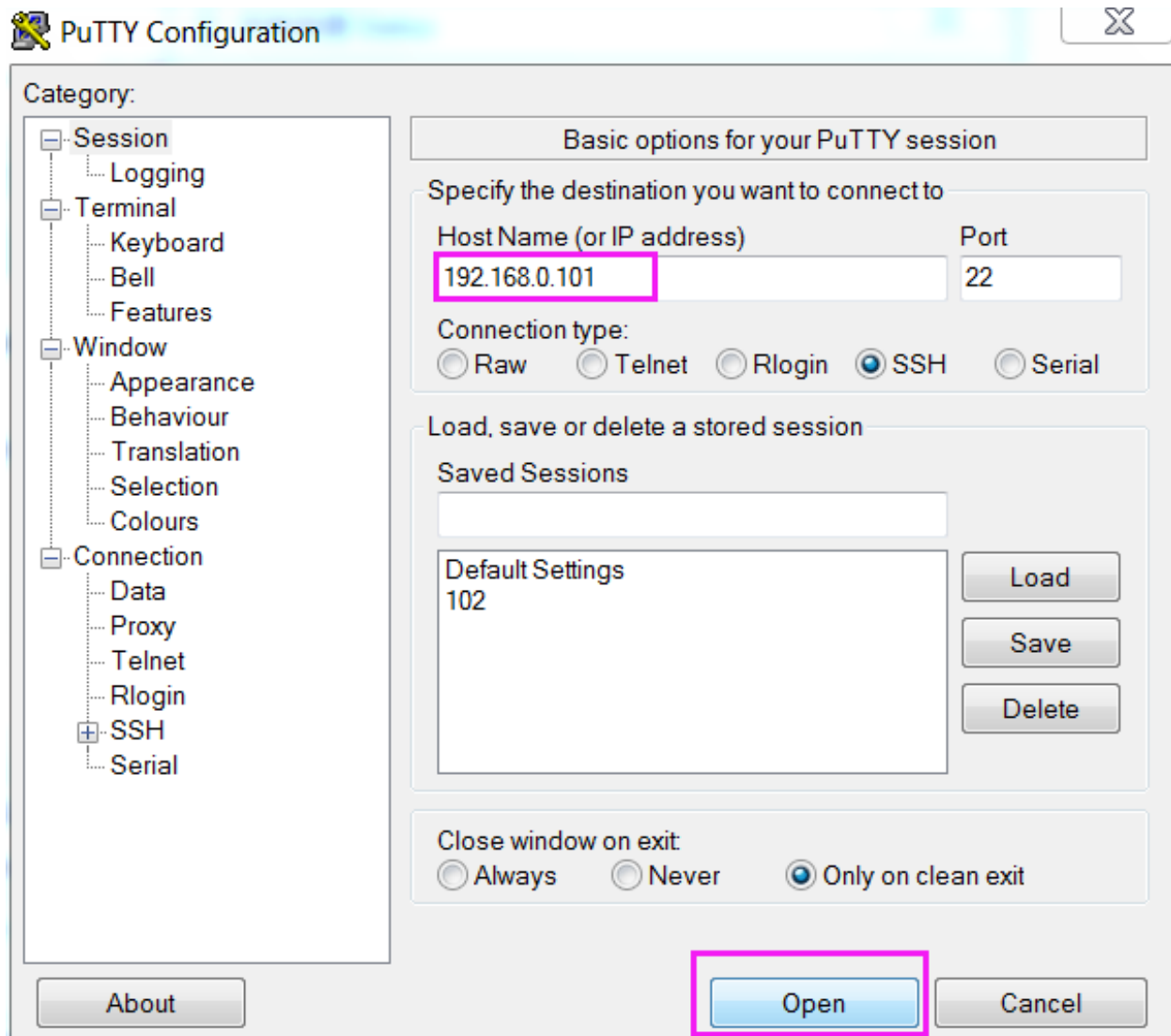
If you're a Windows user, you can use SSH with the application of some software. Here, we recommend **PuTTY**.

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

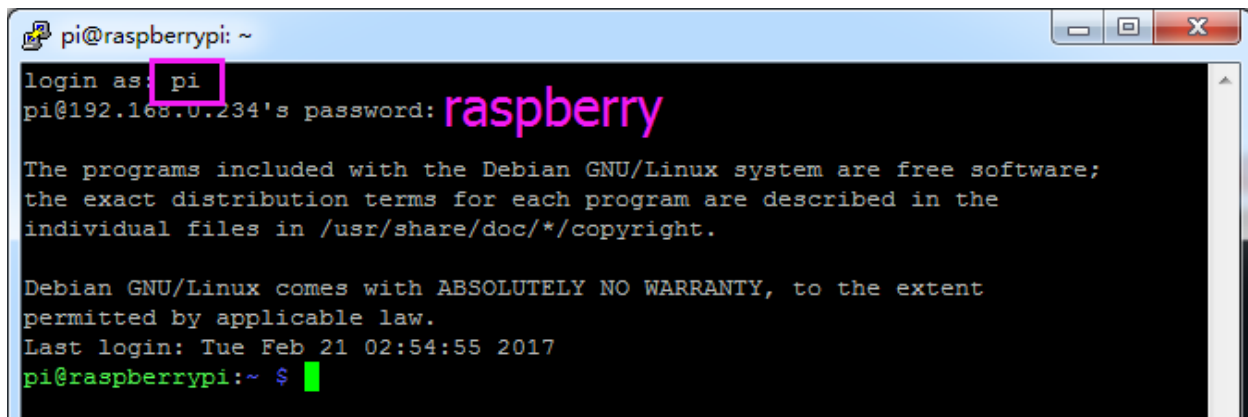
Click **Open**. Note that when you first log in to the Raspberry Pi with the IP address, there prompts a security reminder. Just click **Yes**.

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: raspberry

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](#).

4.1.4 Download and Run the Code

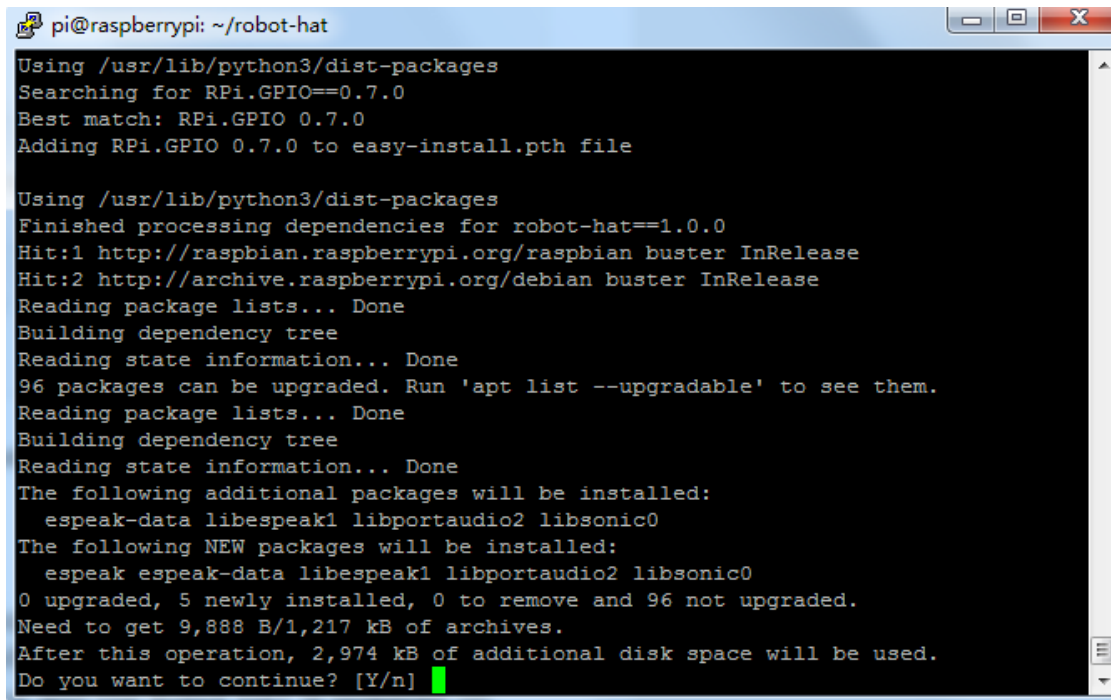
Download the Code Install 2 Libraries

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 pisloth library.

```
cd /home/pi/
git clone -b v2.0 https://github.com/sunfounder/pisloth.git
cd pisloth
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 pisloth will have no sound.

```
cd /home/pi/pisloth
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

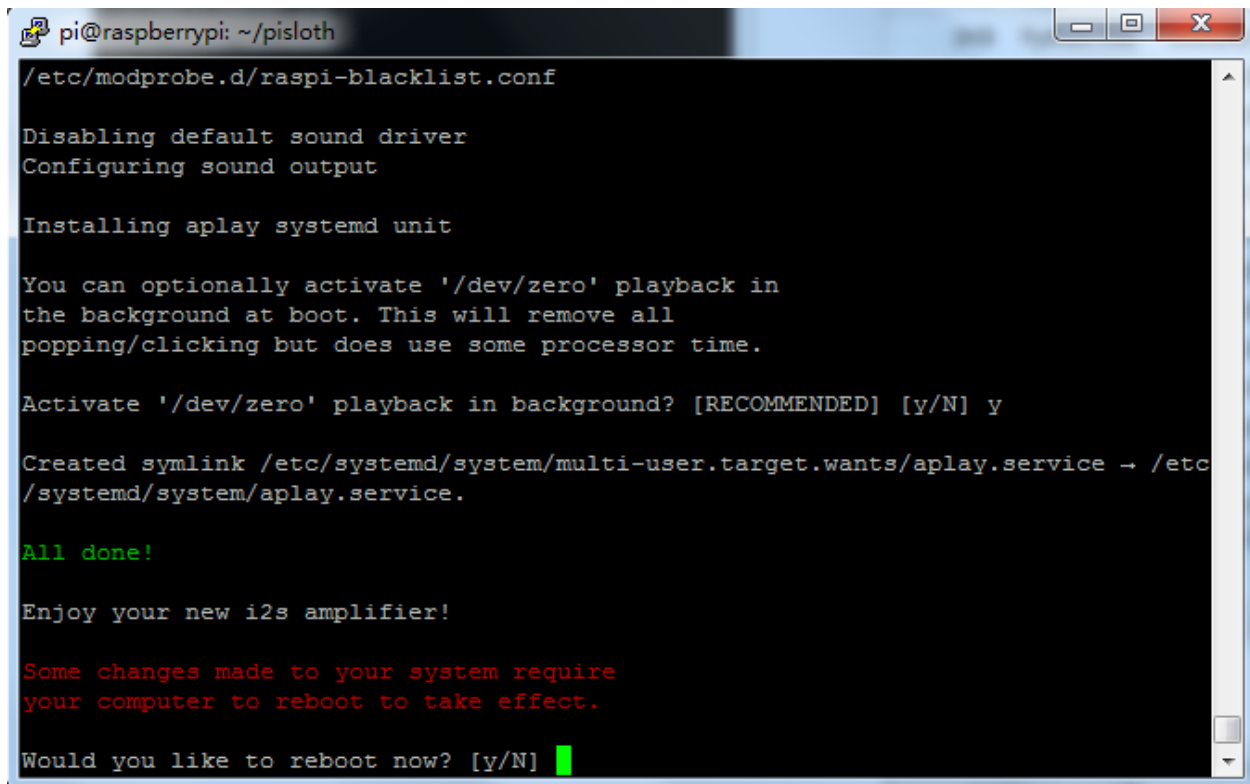
Installing aplay systemd unit

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.

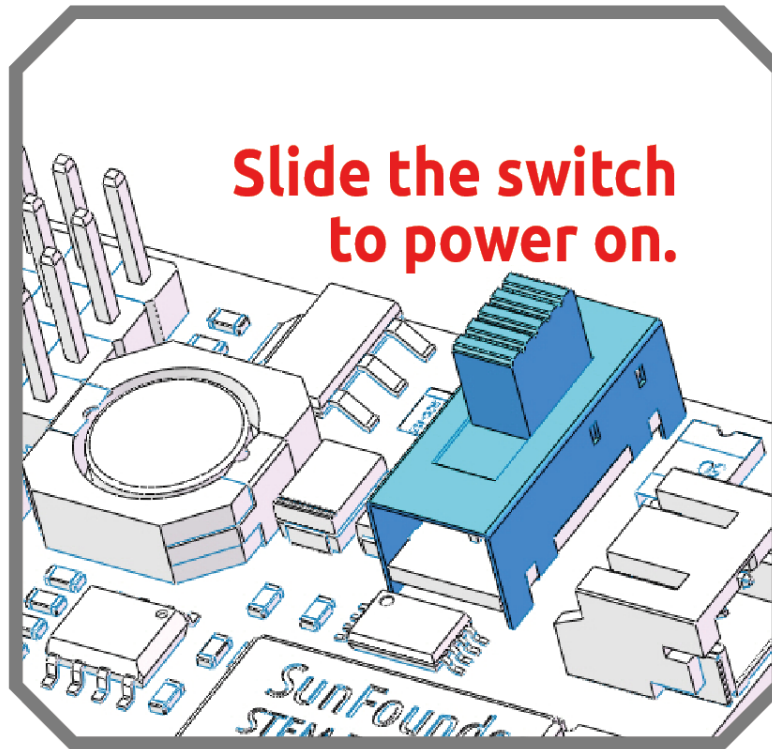
Would you like to reboot now? [y/N] █
```

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.

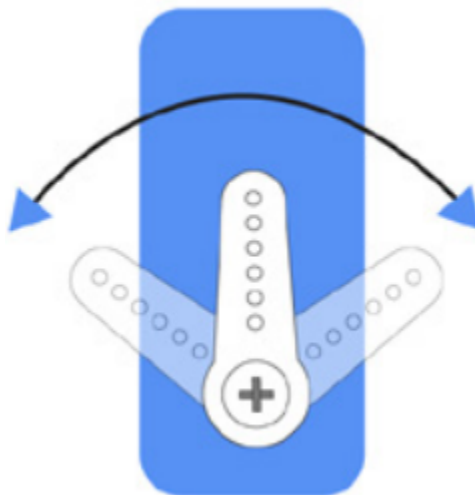


Before you run any code, enable your Raspberry Pi's I2C port, see: [I2C Configuration](#).

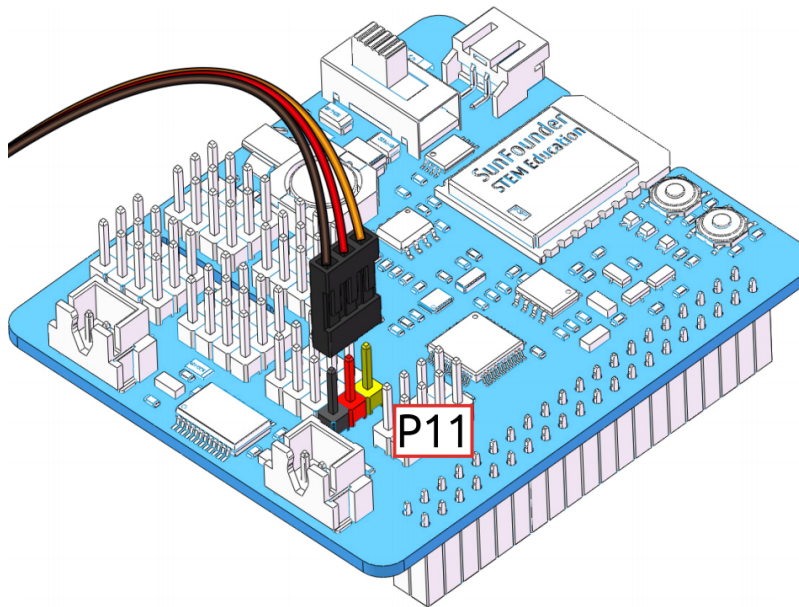
Now run the `servo_zeroing.py` in the `examples/` folder.

```
cd /home/pi/pisloth/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.



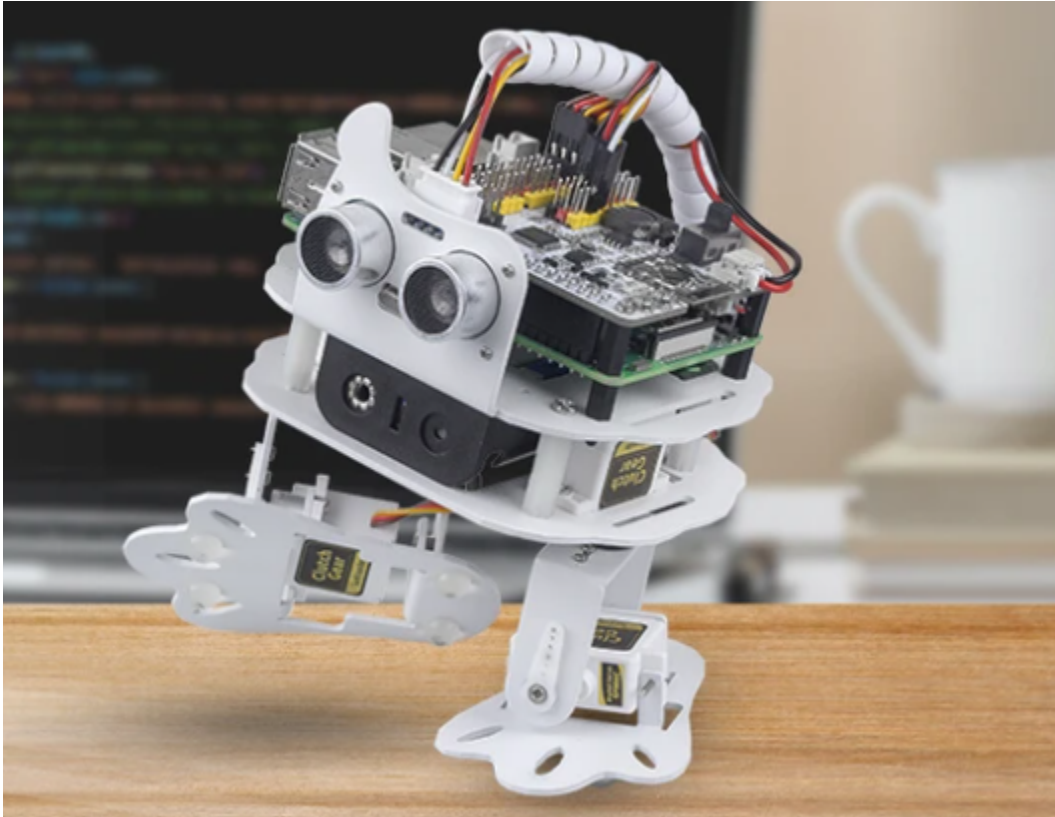
So now if the servo arm returns after the servo arm returns, this function will take effect. If not, If not, please check the insertion direction of the servo cable and re-run the code.

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.

4.2 Move

This is the first project. PiSloth has woken up, and it moves freely.



Run the Code

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

After running the code, you will see PiSloth move left 7 steps, forward 5 steps, right 7 steps, and forward 5 steps.

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 pisloth import Sloth

sloth = Sloth([1,2,3,4])
sloth.set_offset([0,0,0,0])

def main():
    sloth.do_action('turn left', 7, 90)
    sloth.do_action('forward', 5, 90)
    sloth.do_action('turn right', 7, 90)
    sloth.do_action('forward', 5, 90)

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

How it works?

4.2. Move

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

```
from pisloth import Sloth
```

Then instantiate the `Sloth` class.

```
sloth = Sloth([1,2,3,4])
sloth.set_offset([0,0,0,0])
```

Finally use the `sloth.do_action()` function to make PiSloth move.

```
sloth.do_action('turn left', 7, 90)
sloth.do_action('forward', 5, 90)
sloth.do_action('turn right', 7, 90)
sloth.do_action('forward', 5, 90)
```

In general, all actions of PiSloth can be implemented with the `sloth.do_action()` function. It has four parameters:

- `motion_name` is the name of specific actions, including: forward, turn right, turn left, backward, stand, moon walk left, moon walk right, hook, big swing, swing, walk boldly, walk backward boldly, walk shyly, walk backward shyly, stomp right, stomp left, close, open, tiptoe left, tiptoe right, fall left, fall right.
- `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.
- `bpm` means rhythm, we will use it later in the [Dance](#) project.

Note: You can add different sound effects or music to `musics` or `sounds` folder via [Filezilla Software](#).

4.3 Dance

Now, PiSloth will show you its newly learned dance.



Note: You can download and print the [PDF Cartoon Mask](#) for your PiSloth.

Run the Code

```
cd /home/pi/pisloth/examples
sudo python3 dancing.py
```

The whole dance is divided into 2 parts, and PiSloth will finish these 2 parts with the music. If you don't stop the code, it will repeat the dance.

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 pisloth import Sloth
from robot_hat import Music
from robot_hat import Ultrasonic
from robot_hat import Pin
import time
import os

music = Music()

sloth = Sloth([1,2,3,4])
```

(continues on next page)

(continued from previous page)

```

sloth.set_offset([0,0,0,0])

def main():

    music.background_music('./musics/india-Arulo.mp3')
    music.music_set_volume(20)
    sloth.do_action('stomp left',3,bpm=129)
    sloth.do_action('stomp right',3,bpm=129)
    sloth.do_action('moon walk left',3,bpm=129)
    sloth.do_action('moon walk right',3,bpm=129)
    for i in range(3):
        sloth.do_action('swing',1,bpm=129)
        sloth.do_action('stand',1,bpm=129)
    for i in range(3):
        sloth.do_action('close',1,bpm=129)
        sloth.do_action('stand',1,bpm=129)
        sloth.do_action('open',1,bpm=129)
        sloth.do_action('stand',1,bpm=129)
    sloth.do_action('tiptoe left',2,bpm=129)
    sloth.do_action('tiptoe right',2,bpm=129)

    sloth.do_action('stomp left',3,bpm=129)
    sloth.do_action('stomp rihgt',3,bpm=129)
    sloth.do_action('moon walk left',3,bpm=129)
    sloth.do_action('moon walk right',3,bpm=129)
    for i in range(3):
        sloth.do_action('hook',1,bpm=129)
        sloth.do_action('stand',1,bpm=129)
    for i in range(4):
        sloth.do_action('swing',1,bpm=129)
        sloth.do_action('big swing',1,bpm=129)
        sloth.do_action('swing',1,bpm=129)
        sloth.do_action('stand',1,bpm=129)

    sloth.do_action('tiptoe right',2,bpm=129)
    sloth.do_action('stand',2,bpm=129)

    music.music_stop()
    time.sleep(10)

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

```

How it works?

You can make PiSloth play music by importing the following libraries.

```
from robot_hat import TTS, Music
```

Play the background music in the `pisloth/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/india-Arulo.mp3')
music.music_set_volume(20)
```

In general, all actions of PiSloth can be implemented with the `sloth.do_action()` function. It has four parameters:

- **motion_name** is the name of specific actions, including: forward, turn right, turn left, backward, stand, moon walk left, moon walk right, hook, big swing, swing, walk boldly, walk backward boldly, walk shyly, walk backward shyly, stomp right, stomp left, close, open, tiptoe left, tiptoe right, fall left, fall right.
- **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.
- **bpm** means rhythm, the bpm parameter here affects the interval time of PiSloth movement. The higher the value, the shorter the interval time. When we know the beat of a song through the **bpm calculator**, we can make PiSloth dance to the music.

For music bpm, if you want to know more, you can refer to: <https://en.wikipedia.org/wiki/Tempo>

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

4.4 Obstacle Avoidance

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

Run the Code

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

After the code runs, PiSloth 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 `pisloth\examples`. After modifying the code, you can run it directly to see the effect.

```
from pisloth import Sloth
from robot_hat import TTS, Music
from robot_hat import Ultrasonic
from robot_hat import Pin
import time
import os

tts = TTS()
music = Music()

sloth = Sloth([1,2,3,4])
```

(continues on next page)

(continued from previous page)

```

sloth.set_offset([0,0,0,0])
sonar = Ultrasonic(Pin("D2") ,Pin("D3"))

alert_distance = 10

def main():
    distance = sonar.read()
    if distance < 0:
        pass
    elif distance <= alert_distance:
        try:
            music.sound_effect_threading('./sounds/sign.wav')
        except Exception as e:
            print(e)
        sloth.do_action('hook', 1,95)
        time.sleep(0.5)
        sloth.do_action('stand', 1,95)
        time.sleep(0.5)
        sloth.do_action('turn left',7,90)
        sloth.do_action('stand', 1,95)
        time.sleep(0.2)
    else :
        sloth.do_action('forward', 1,90)

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. PiSloth does hook, stand, left turn and stand in sequence.
- When the distance is greater than alert_distance, PiSloth will move forward.

```

distance = sonar.read()
if distance < 0:
    pass
elif distance <= alert_distance:
    try:
        music.sound_effect_threading('./sounds/sign.wav')
    except Exception as e:
        print(e)
    sloth.do_action('hook', 1,95)

```

(continues on next page)

(continued from previous page)

```

time.sleep(0.5)
sloth.do_action('stand', 1,95)
time.sleep(0.5)
sloth.do_action('turn left',7,90)
sloth.do_action('stand', 1,95)
time.sleep(0.2)
else :
    sloth.do_action('forward', 1,90)

```

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

4.5 Don't Touch Me

If you don't meet PiSloth's needs, it will get angry and stay away from your touch.

Run the Code

```

cd /home/pi/pisloth/examples
sudo python3 dont_touch_me.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 `pisloth/examples`. After modifying the code, you can run it directly to see the effect.

```

from pisloth import Sloth
from robot_hat import Music
from robot_hat import Ultrasonic
from robot_hat import Pin
import time
import os

music = Music()

sloth = Sloth([1,2,3,4])
sloth.set_offset([0,0,0,0])
sonar = Ultrasonic(Pin("D2"), Pin("D3"))

alert_distance = 20

def main():
    distance = sonar.read()
    print(distance)
    if distance <= alert_distance :
        try:
            music.sound_effect_threading('./sounds/talk3.wav')
        except Exception as e:
            print(e)
            sloth.do_action('backward', 2, 90)
    else:

```

(continues on next page)

(continued from previous page)

```
sloth.do_action('stand', 1, 90)
time.sleep(1)

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

How it works?

Instantiate various classes of Music, Sloth and Ultrasonic to be used.

```
music = Music()

sloth = Sloth([1,2,3,4])
sloth.set_offset([0,0,0,0])
sonar = Ultrasonic(Pin("D2") ,Pin("D3"))
```

Here is the main program.

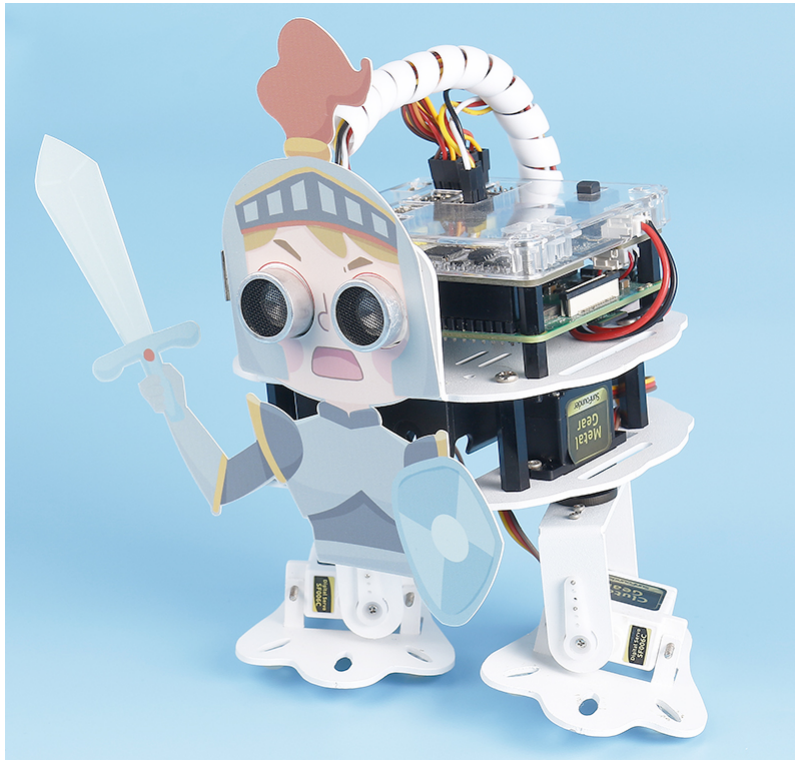
- Read the distance detected by the ultrasonic module and print it.
- When the distance is less than or equal to alert_distance (the threshold value set earlier, which is 20), play the sound effect talk3.wav and move backward.
- When the distance is greater than alert_distance, PiSloth will Stand.

```
distance = sonar.read()
print(distance)
if distance <= alert_distance :
    try:
        music.sound_effect_threading('./sounds/talk3.wav')
    except Exception as e:
        print(e)
    sloth.do_action('backward', 2, 90)
else:
    sloth.do_action('stand', 1, 90)
time.sleep(1)
```

Note: You can add different sound effects or music to musics or sounds folder via [Filezilla Software](#).

4.6 Let's Fight! Warrior!

Here, PiSloth is a brave warrior, when it appears in front of the enemy, it will let out a roar and rush to the enemy.



Note: You can download and print the [PDF Cartoon Mask](#) for your PiSloth.

Run the Code

```
cd /home/pi/pislloth/examples
sudo python3 lets_fight.py
```

After the code is run, PiSloth will continuously detect the distance of the obstacle, when the distance is between 5 and 40, PiSloth will make a roaring sound and rush forward; when the distance of the obstacle is less than 5, PiSloth will stop.

Code

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

```
from pislloth import Sloth
from robot_hat import Music
from robot_hat import Ultrasonic
from robot_hat import Pin
import time
import os
```

(continues on next page)

(continued from previous page)

```

music = Music()

sloth = Sloth([1,2,3,4])
sloth.set_offset([0,0,0,0])
sonar = Ultrasonic(Pin("D2"), Pin("D3"))

alert_distance = 40
contact_distance = 5

def main():
    distance = sonar.read()
    if distance <= alert_distance and distance >= contact_distance :
        try:
            music.sound_effect_play('./sounds/battle.wav')
            music.background_music('./musics/attack.mp3')
            music.music_set_volume(20)
        except Exception as e:
            print(e)
        while True:
            distance = sonar.read()
            print(distance)
            if distance < 0:
                continue
            if distance <= contact_distance:
                break
            sloth.do_action('forward', 1, 90)
        sloth.do_action('stand', 1, 90)
        time.sleep(1)

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

```

How it works?

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 between 5 and 40, PiSloth will play warning.wav and attack.mp3 and move forward.
- When the distance is less than 5, PiSloth will keep the stand position.

```

distance = sonar.read()
if distance <= alert_distance and distance >= contact_distance :
    try:
        music.sound_effect_play('./sounds/battle.wav')
        music.background_music('./musics/attack.mp3')
        music.music_set_volume(20)
    except Exception as e:
        print(e)
    while True:
        distance = sonar.read()
        print(distance)

```

(continues on next page)

(continued from previous page)

```

    if distance < 0:
        continue
    if distance <= contact_distance:
        break
    sloth.do_action('forward', 1, 95)
sloth.do_action('stand', 1, 90)
time.sleep(1)

```

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

4.7 Emotional PiSloth

PiSloth is very emotional, sometimes happy, sometimes shy, sometimes confused.

Run the Code

```

cd /home/pi/pislloth/examples
sudo python3 emotional_pislloth.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 `pislloth/examples`. After modifying the code, you can run it directly to see the effect.

```

from pislloth import Sloth
from robot_hat import TTS, Music
import time

tts = TTS()
music = Music()

sloth = Sloth([1, 2, 3, 4])
sloth.set_offset([0, 0, 0, 0])

def confuse():
    try:
        music.sound_effect_threading('./sounds/sign.wav')
    except Exception as e:
        print(e)
    sloth.do_action('hook', 1, 90)

def happy():
    try:
        music.sound_effect_threading('./sounds/happy2.wav')
    except Exception as e:
        print(e)
    for i in range(3):
        sloth.do_action('hook', 1, 90)
        sloth.do_action('stand', 1, 90)

def fear():

```

(continues on next page)

(continued from previous page)

```

    try:
        music.sound_effect_threading('./sounds/warning.wav')
    except Exception as e:
        print(e)
    sloth.do_action('hook', 1, 90)
    sloth.do_action('stand', 1, 90)
    try:
        music.sound_effect_threading('./sounds/warning.wav')
    except Exception as e:
        print(e)
    sloth.do_action('walk backward boldly', 1, 90)
    sloth.do_action('stand', 1, 90)

def sad():
    try:
        music.sound_effect_threading('./sounds/depress.wav')
    except Exception as e:
        print(e)
    sloth.do_action('big swing', 1, 90)

def angry():
    try:
        music.sound_effect_threading('./sounds/error.wav')
    except Exception as e:
        print(e)
    sloth.do_action('walk backward boldly', 1, 90)
    sloth.do_action('stand', 1, 90)

def fail():
    try:
        music.sound_effect_threading('./sounds/depress2.wav')
    except Exception as e:
        print(e)
    sloth.do_action('fall left', 1, 90)

def shy():
    try:
        music.sound_effect_threading('./sounds/talk3.wav')
    except Exception as e:
        print(e)
    sloth.do_action('close', 1, 90)
    time.sleep(1)
    try:
        music.sound_effect_threading('./sounds/talk2.wav')
    except Exception as e:
        print(e)
    sloth.do_action('stand', 1, 90)

def main():

    print("shy")
    shy()
    time.sleep(1)
    sloth.do_action('stand', 1, 90)
    time.sleep(2)

    print("confuse")

```

(continues on next page)

(continued from previous page)

```

confuse()
time.sleep(1)
sloth.do_action('stand', 1, 90)
time.sleep(2)

print("happy")
happy()
time.sleep(1)
sloth.do_action('stand', 1, 90)
time.sleep(2)

print("fear")
fear()
time.sleep(1)
sloth.do_action('stand', 1, 90)
time.sleep(2)

print("sad")
sad()
time.sleep(1)
sloth.do_action('stand', 1, 90)
time.sleep(2)

print("angry")
angry()
time.sleep(1)
sloth.do_action('stand', 1, 90)
time.sleep(2)

print("fail")
fail()
time.sleep(1)
sloth.do_action('stand', 1, 90)
time.sleep(2)

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

```

How it works?

In this project, actions + sound effects are combined into different emotional actions, and you can also modify them yourself.

Note: This `fail` action will make the PiSloth fall, be careful not to let it fall off the table and break it.

You can add different sound effects or music to `musics` or `sounds` folder via [Filezilla Software](#).

4.8 Remote Control

In this project, we will learn how to use the keyboard to remotely control the PiSloth. You can control the PiSloth to move up, down, left, and right and speak through specific keys.

Run the Code

```
cd /home/pi/pisloth/examples
sudo python3 keyboard_control.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 `pisloth\examples`. After modifying the code, you can run it directly to see the effect.

```
from pisloth import Sloth
from robot_hat import Music
from robot_hat import TTS
import sys
import tty
import termios
import time

sloth = Sloth([1,2,3,4])
tts = TTS()
music = Music()
sloth.set_offset([0,0,0,0])

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
    1: Sound effect: talk1
    2: Sound effect: talk2
    3: Sound effect: talk3
    4: Sound effect: depress2
    Q: Say: "Oh hello there"
    E: Say: "bye"
    ESC: Quit
'''

def main():
```

(continues on next page)

(continued from previous page)

```

print(manual)
while True:
    key = readchar()
    # print(key)
    if key == "w":
        sloth.do_action('forward', 1, 90)
    elif key == "a":
        sloth.do_action('turn left', 1, 90)
    elif key == "s":
        sloth.do_action('backward', 1, 90)
    elif key == "d":
        sloth.do_action('turn right', 1, 90)
    elif key == "1":
        music.sound_effect_play('./sounds/talk1.wav')
    elif key == "2":
        music.sound_effect_play('./sounds/talk2.wav')
    elif key == "3":
        music.sound_effect_play('./sounds/talk3.wav')
    elif key == "4":
        music.sound_effect_play('./sounds/depress.wav')
    elif key == "q":
        tts.say("Oh hello there")
    elif key == "e":
        tts.say("bye")
    elif key == chr(27): # 27 for ESC
        break
    time.sleep(0.05)
print("\nQuit")

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 PiSloth do the actions we set, call the `tts.say()` function to speak or play the sound effects prepared in advance.

```
key = readkey()
# print(key)
if key == "w":
    sloth.do_action('forward', 1, 90)
elif key == "a":
    sloth.do_action('turn left', 1, 90)
elif key == "s":
    sloth.do_action('backward', 1, 90)
elif key == "d":
    sloth.do_action('turn right', 1, 90)
elif key == "1":
    music.sound_effect_play('./sounds/talk1.wav')
elif key == "2":
    music.sound_effect_play('./sounds/talk2.wav')
elif key == "3":
    music.sound_effect_play('./sounds/talk3.wav')
elif key == "4":
    music.sound_effect_play('./sounds/depress.wav')
elif key == "q":
    tts.say("Oh hello there")
elif key == "e":
    tts.say("bye")
elif key == chr(27): # 27 for ESC
    break
```

Note: You can add different sound effects or music to `musics` or `sounds` folder via [Filezilla Software](#).

4.9 Custom Step

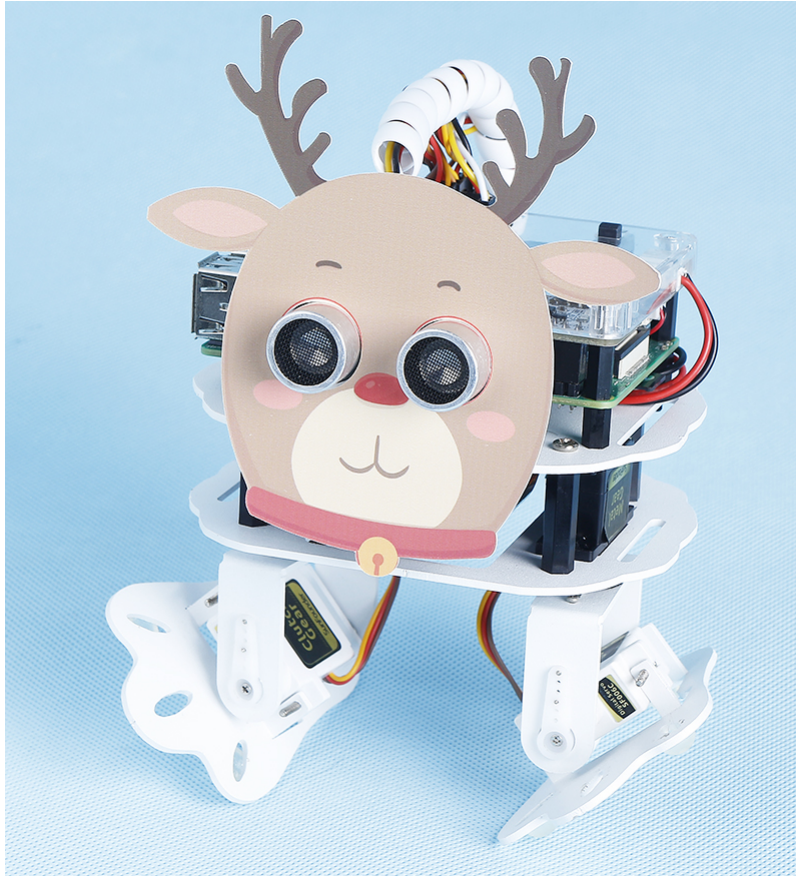
In the previous projects, we used a lot of actions that we wrote, so how are these actions composed and done? Generally speaking, an action is composed of one or more steps.

In this project, we will learn how to customize PiSloth's step.

Run the Code

```
cd /home/pi/pisloth/examples
sudo python3 custom_step.py
```

After the code runs, press the **key ZX** and **key NM** (according to the actual code) to make PiSloth do the step shown in the figure, and you can also let it do other steps.



Note: You can download and print the [PDF Cartoon Mask](#) for your PiSloth.

Press the **key SPACE** to print the angle of the 4 servos at this time. You need to record these angle values, which will be used in the next project *Custom Action*.

```
pi@raspberrypi:~/pisloth/examples $ sudo python3 custom_step.py

Press keys on keyboard to control PiSloth!
Q: Increase left-up servo angle
W: Decrease left-up servo angle
Z: Increase left-down servo angle
X: Decrease left-down servo angle
I: Increase right-up servo angle
O: Decrease right-up servo angle
N: Increase right-down servo angle
M: Decrease right-down servo angle
SPACE: Print all servo angle
ESC: Quit

0 -45 0 40
```

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 pisloth import Sloth
# from robot_hat import Music
# from robot_hat import TTS
from robot_hat import PWM
from robot_hat import Servo

import sys
import tty
import termios
import time

sloth = Sloth([1,2,3,4])
# tts = TTS()
# music = Music()
sloth.set_offset([0,0,0,0])

right_leg_servo = Servo(PWM('P0'))
right_foot_servo = Servo(PWM('P1'))
left_leg_servo = Servo(PWM('P2'))
left_foot_servo = Servo(PWM('P3'))

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!
Q: Increase the servo angle of the left leg
W: Decrease the servo angle of the left leg
Z: Increase the servo angle of the left foot
X: Decrease the servo angle of the left foot
I: Increase the servo angle of the right leg
O: Decrease the servo angle of the right leg
N: Increase the servo angle of the right foot
M: Decrease the servo angle of the right foot
SPACE: Print all angle
ESC: Quit
'''

def main():
    print(manual)

    left_leg=0
    left_foot=0
    right_leg=0
    right_foot=0
    while True:
        key = readchar()
        # print(key)
        if key == "q":

```

(continues on next page)

(continued from previous page)

```

        left_leg = left_leg+5
    elif key == "w":
        left_leg = left_leg-5
    elif key == "z":
        left_foot = left_foot+5
    elif key == "x":
        left_foot = left_foot-5
    elif key == "i":
        right_leg = right_leg+5
    elif key == "o":
        right_leg = right_leg-5
    elif key == "n":
        right_foot = right_foot+5
    elif key == "m":
        right_foot = right_foot-5
    elif key == chr(32): # 32 for space
        print(right_leg, right_foot, left_leg, left_foot)
    elif key == chr(27): # 27 for ESC
        break

    right_leg_servo.angle(right_leg)
    right_foot_servo.angle(right_foot)
    left_leg_servo.angle(left_leg)
    left_foot_servo.angle(left_foot)
    # time.sleep(0.05)

    print("\nQuit")

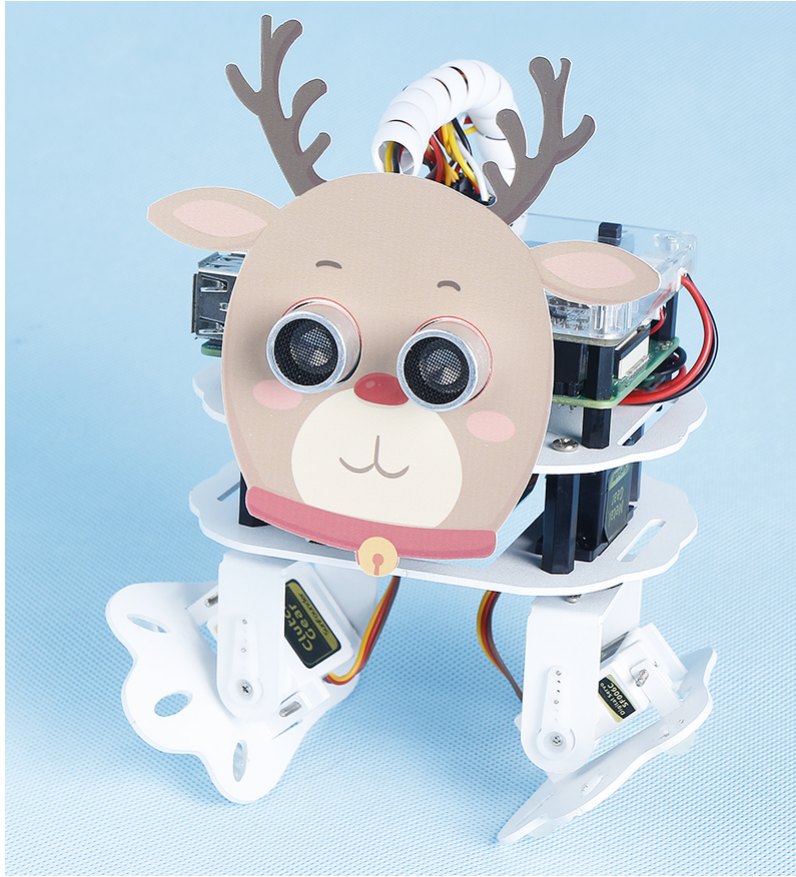
if __name__ == "__main__":
    main()

```

4.10 Custom Action

In the previous project, we were able to give PiSloth custom steps, so how do we combine these steps into actions?

For example, have PiSloth make the step from the previous project and then return to the initial position.



Note: You can download and print the [PDF Cartoon Mask](#) for your PiSloth.

Step 1: Go to the `/home/pi/pisloth/examples` path.

```
cd /home/pi/pisloth/examples
```

Step 2: Open `custom_action.py` with the following command.

```
nano custom_action.py
```

Step 3: Modify the angle in `sloth.add_action()`, each group represents a step, and only 2 steps are set here. You can set multiple steps as needed.

```
sloth.add_action("my_action", [
    [ 0, -45, 0, 40],
    [0, 0, 0, 0]
])
```

Step 4: Run this code.

```
sudo python3 custom_action.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 pislloth\examples. After modifying the code, you can run it directly to see the effect.

```
from pislloth import Sloth
import time

sloth = Sloth([1,2,3,4])
sloth.add_action("my_action", [
    [ 0,-45 ,0, 40],
    [0,  0, 0,  0]
])

def main():
    sloth.do_action("my_action", 1, 80)
    time.sleep(1)

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


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.

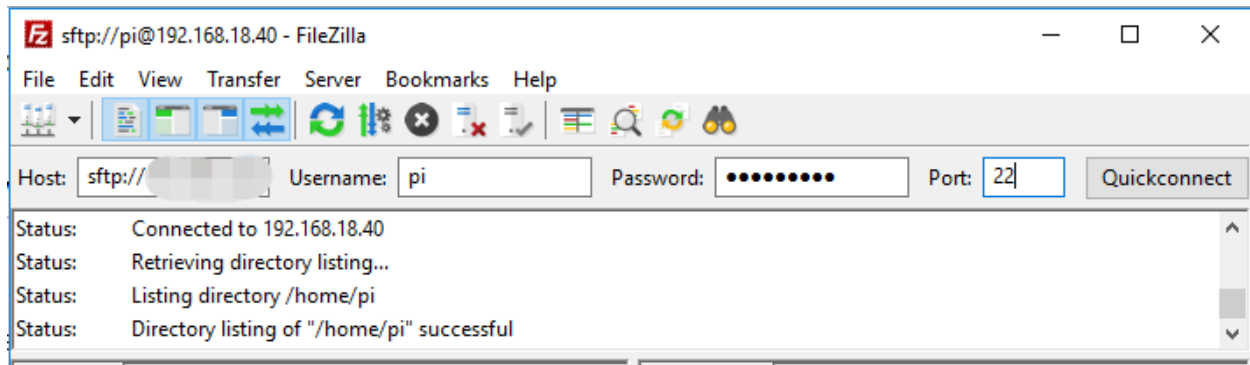
Filezilla is an open source software that not only supports FTP, but also FTP over TLS (FTPS) and SFTP. We can use Filezilla to upload local files (such as pictures and audio, etc.) to the Raspberry Pi, or download files from the Raspberry Pi to the local.

Step 1: Download Filezilla.

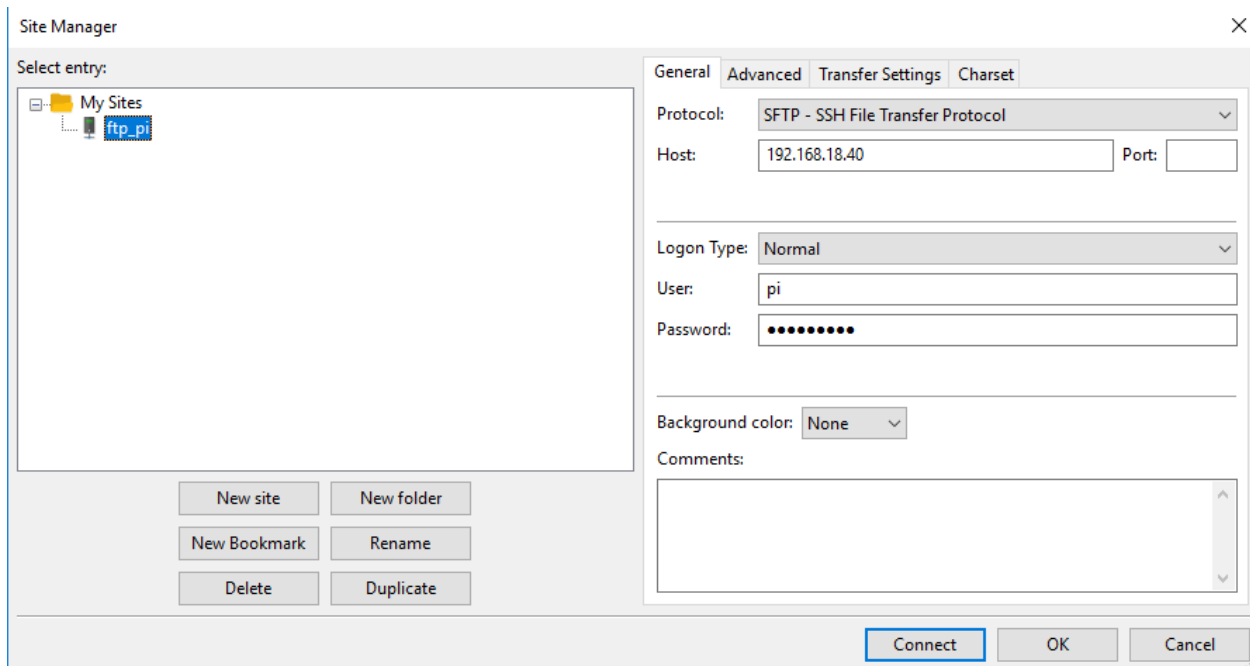
Download the client from [Filezilla's official website](#), Filezilla has a very good tutorial, please refer to: [Documentation - Filezilla](#).

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.

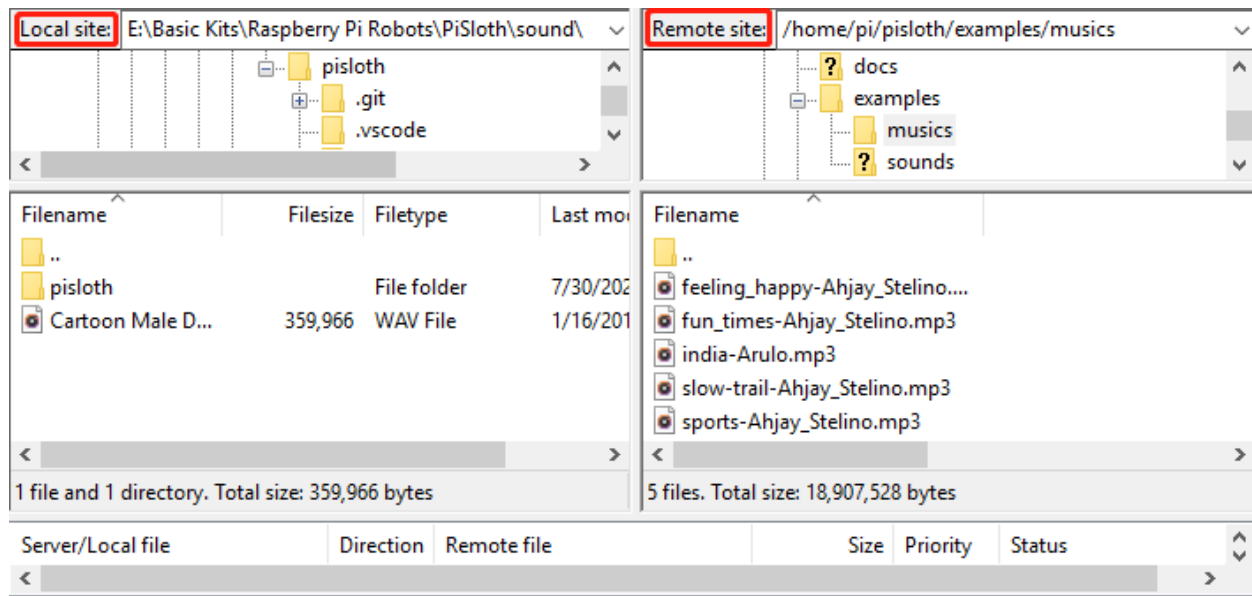


Note: Quick Connect is a good way to test your login information. If you want to create a permanent entry, you can select **File-> Copy Current Connection to Site Manager** after a successful Quick Connect, enter the name and click **OK**. Next time you will be able to connect by selecting the previously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

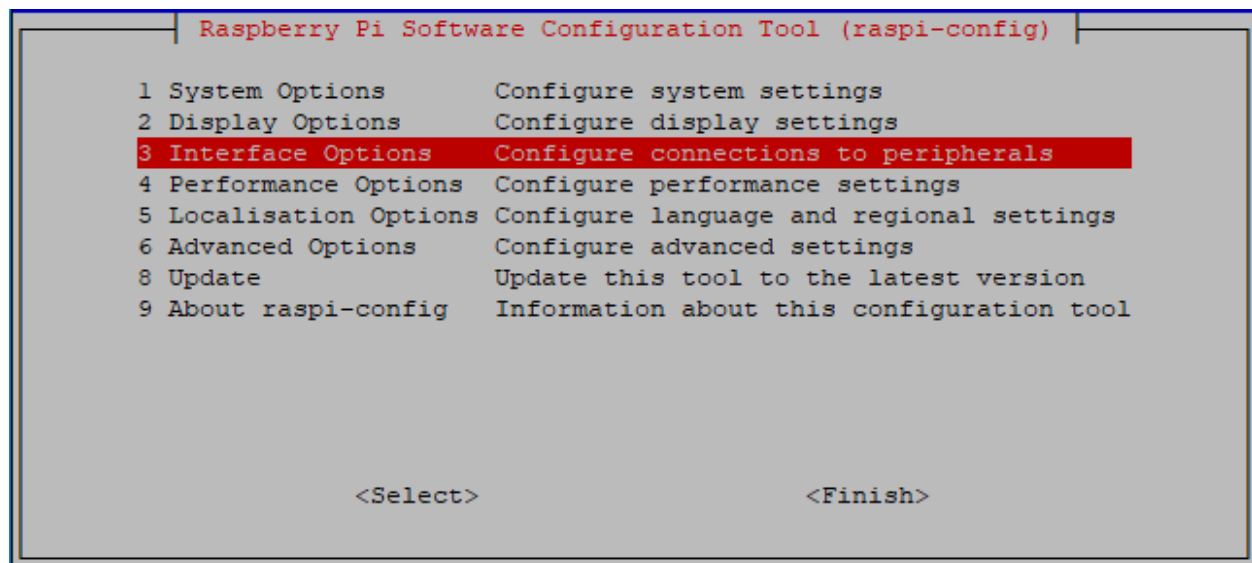


5.2 I2C Configuration

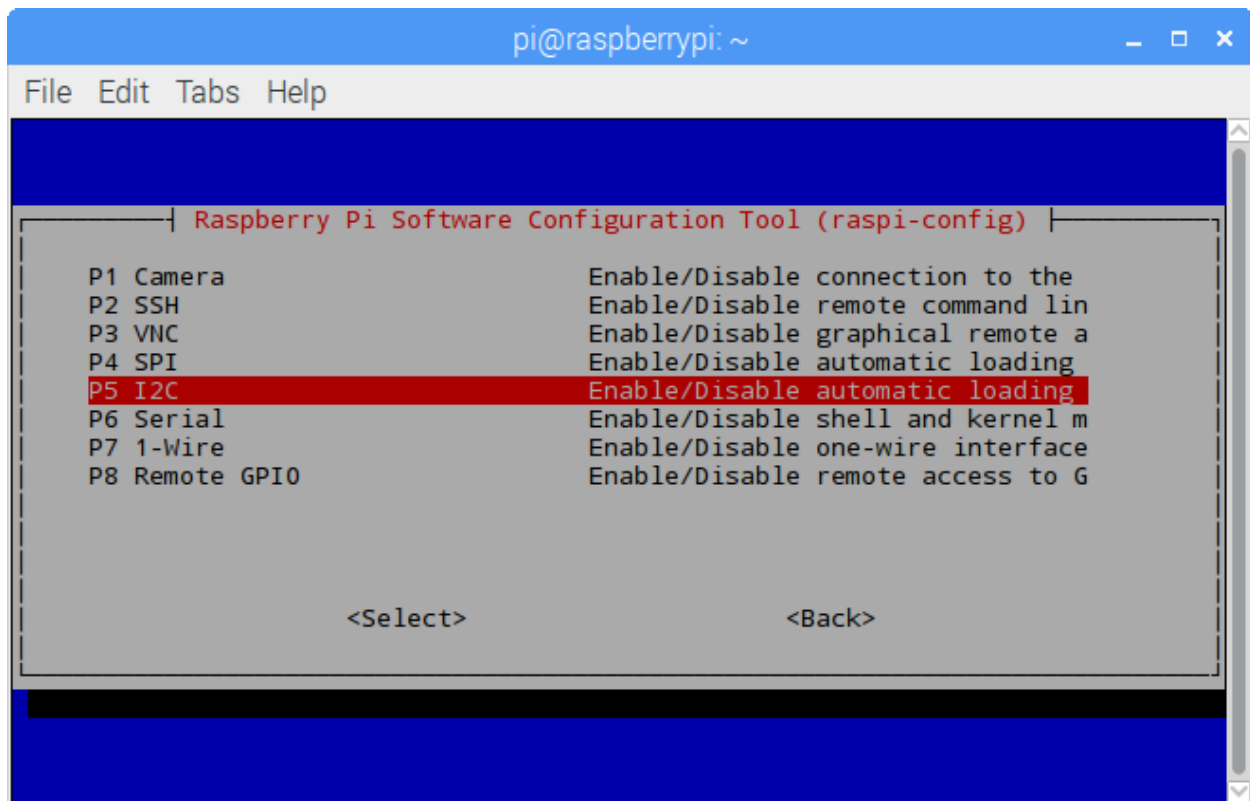
Enable the I2C port of your Raspberry Pi (If you have enabled it, skip this; if you do not know whether you have done that or not, please continue).

```
sudo raspi-config
```

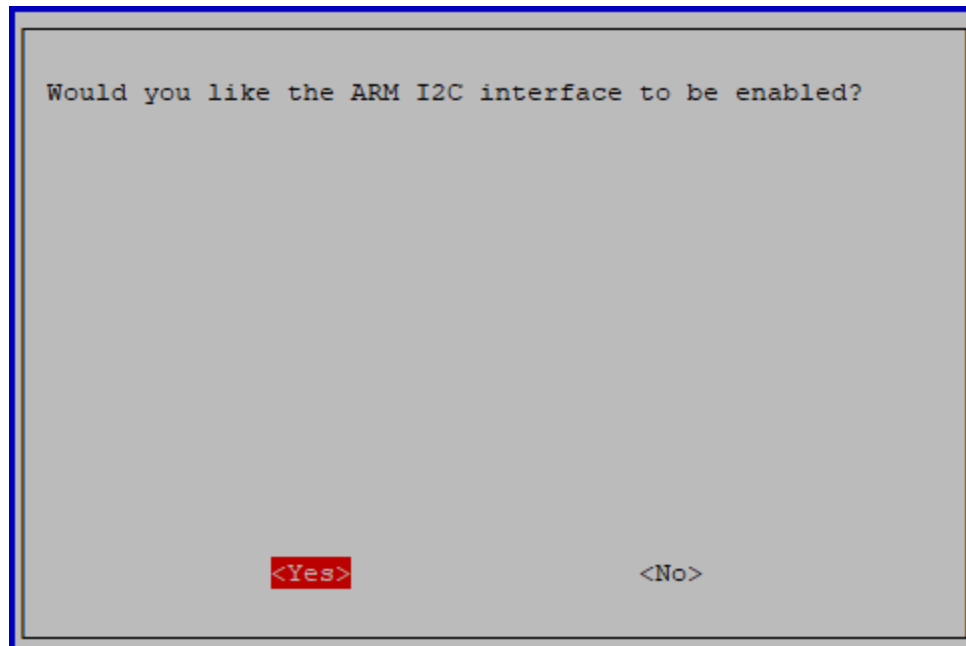
3 Interfacing options



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.

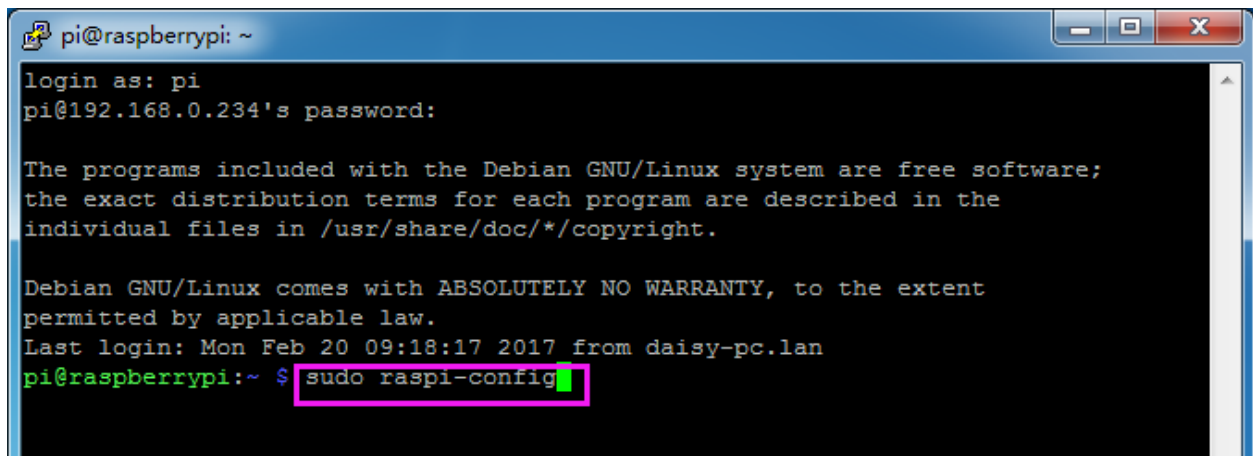
Enable VNC service

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

Step 1

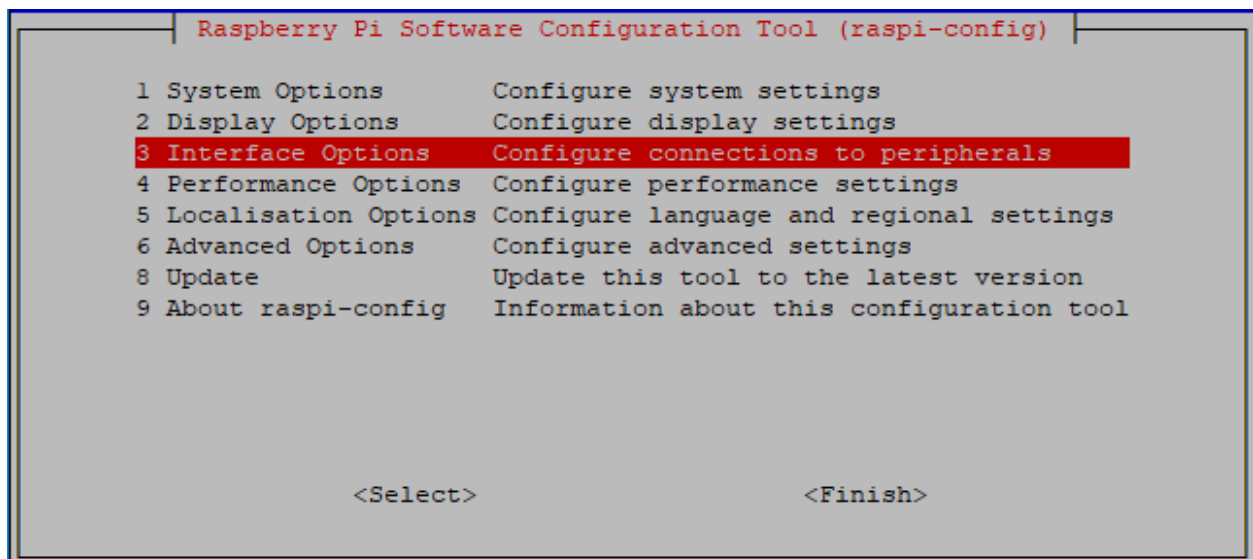
Input the following command:

```
sudo raspi-config
```



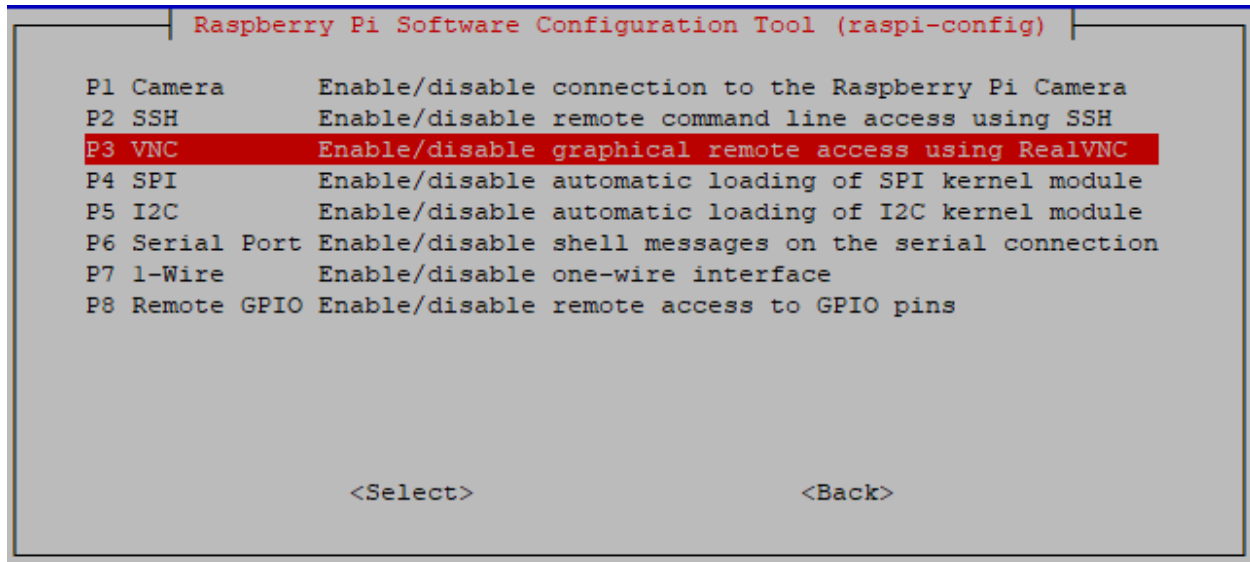
Step 2

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



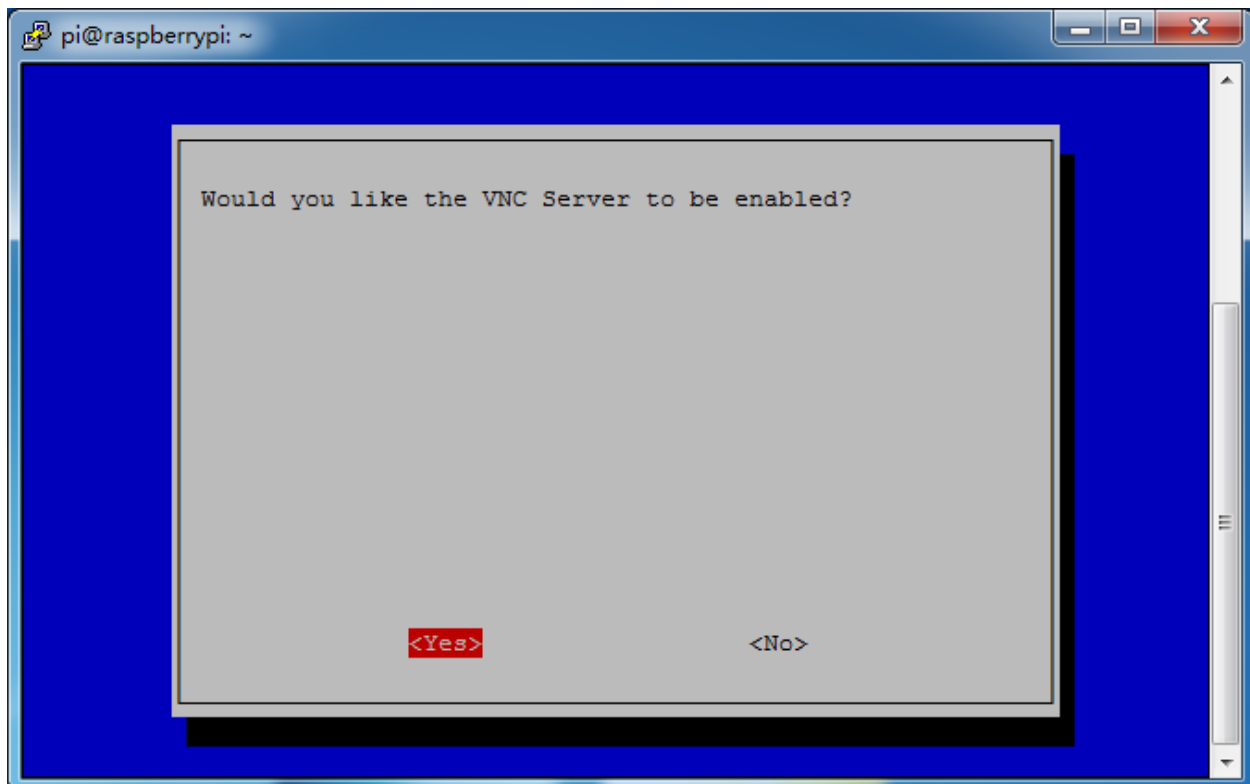
Step 3

P3 VNC



Step 4

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



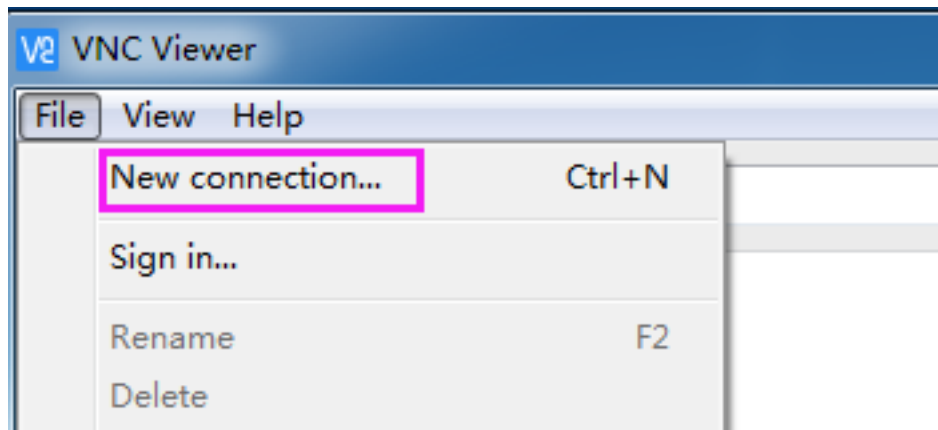
Login to VNC

Step 1

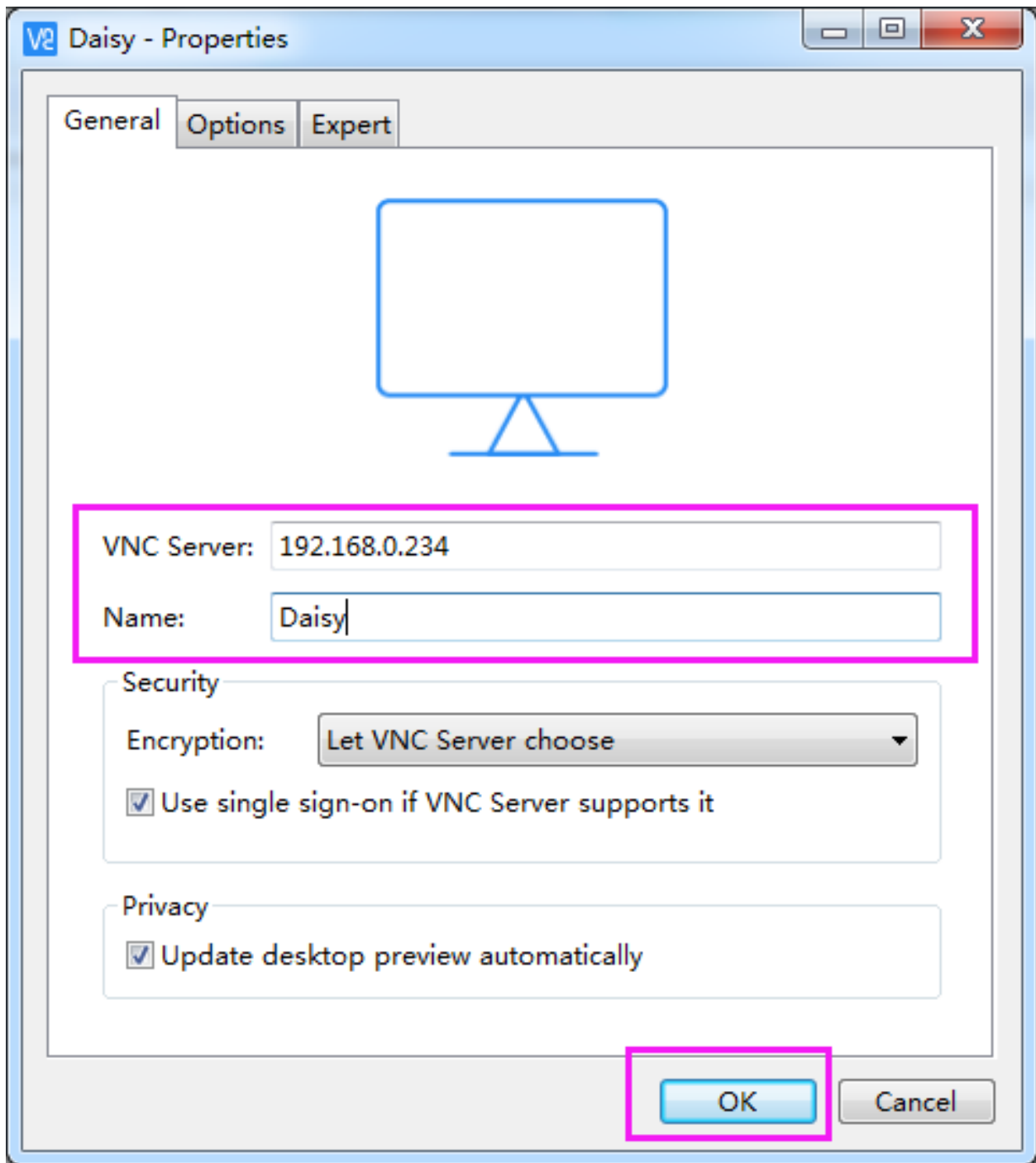
You need to download and install the [VNC Viewer](#) on personal computer. After the installation is done, open it.

Step 2

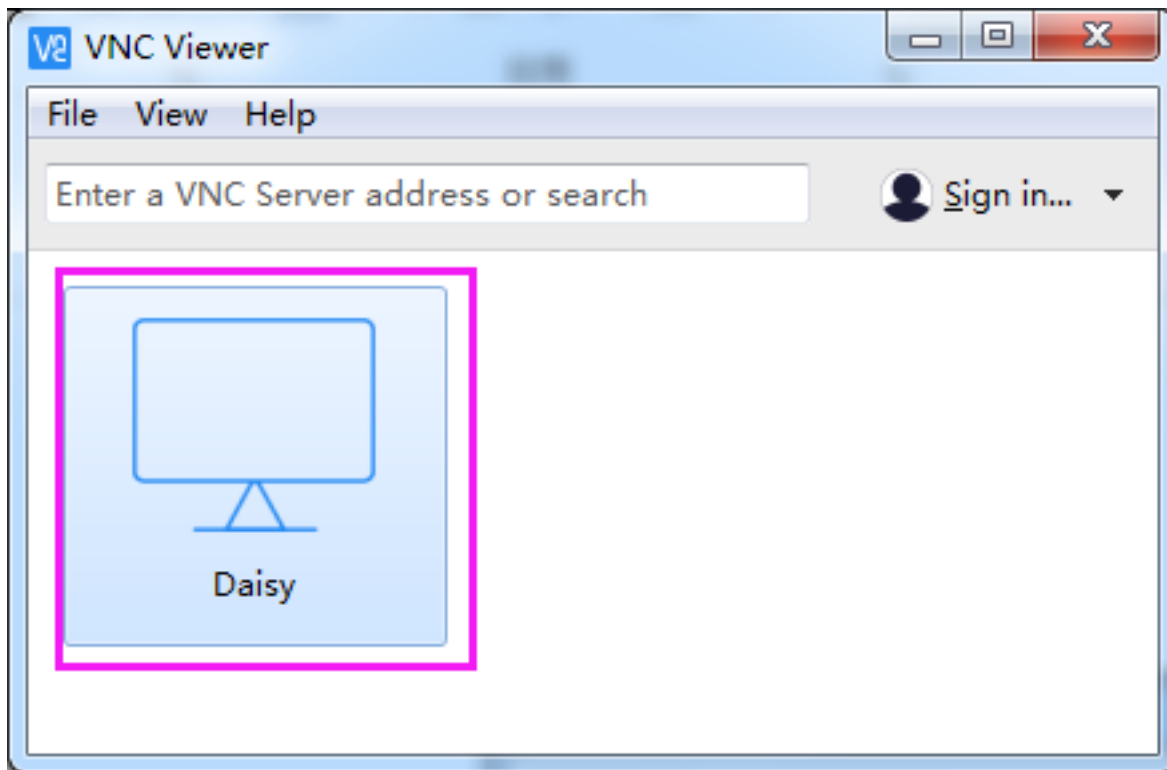
Then select “**New connection**”.

**Step 3**

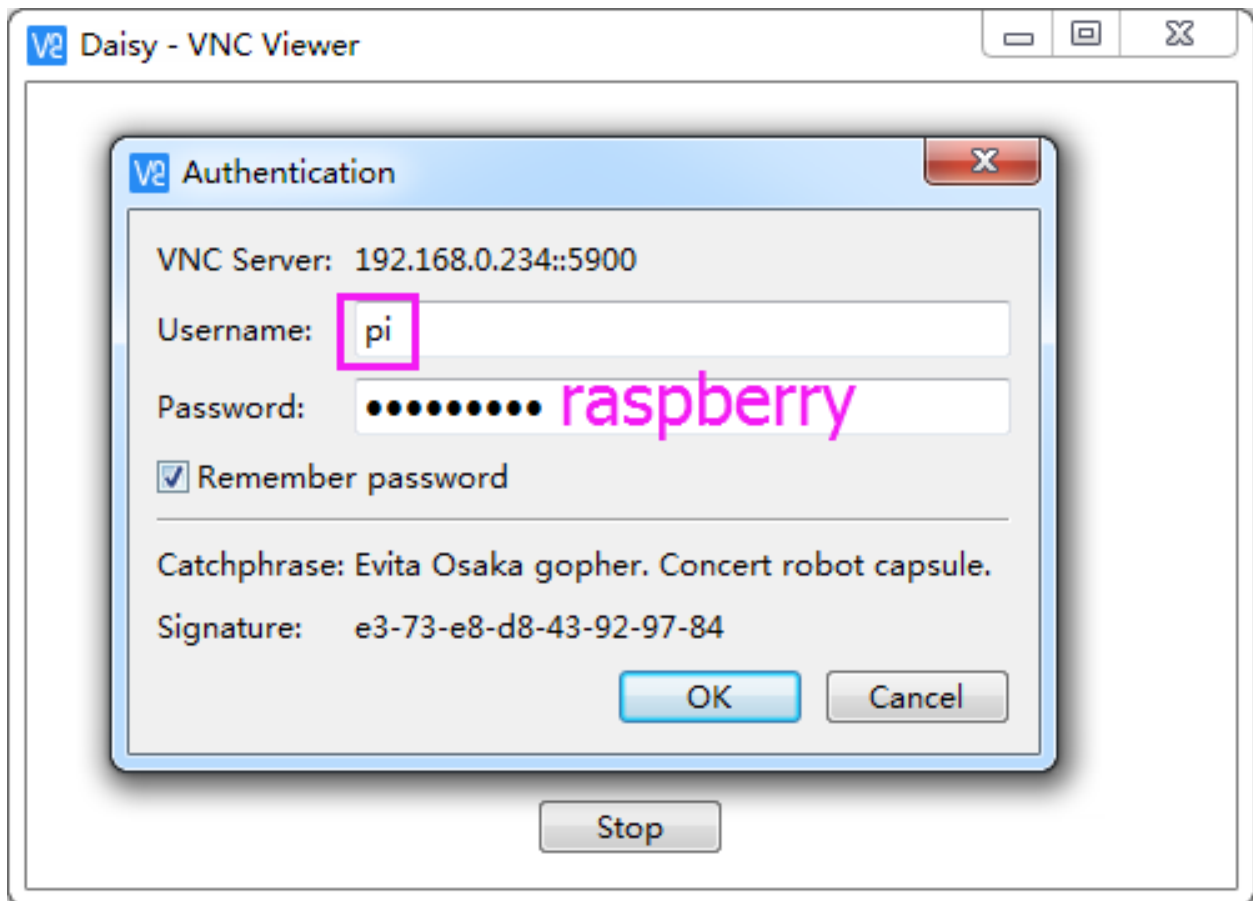
Input IP address of Raspberry Pi and any **Name**.

**Step 4**

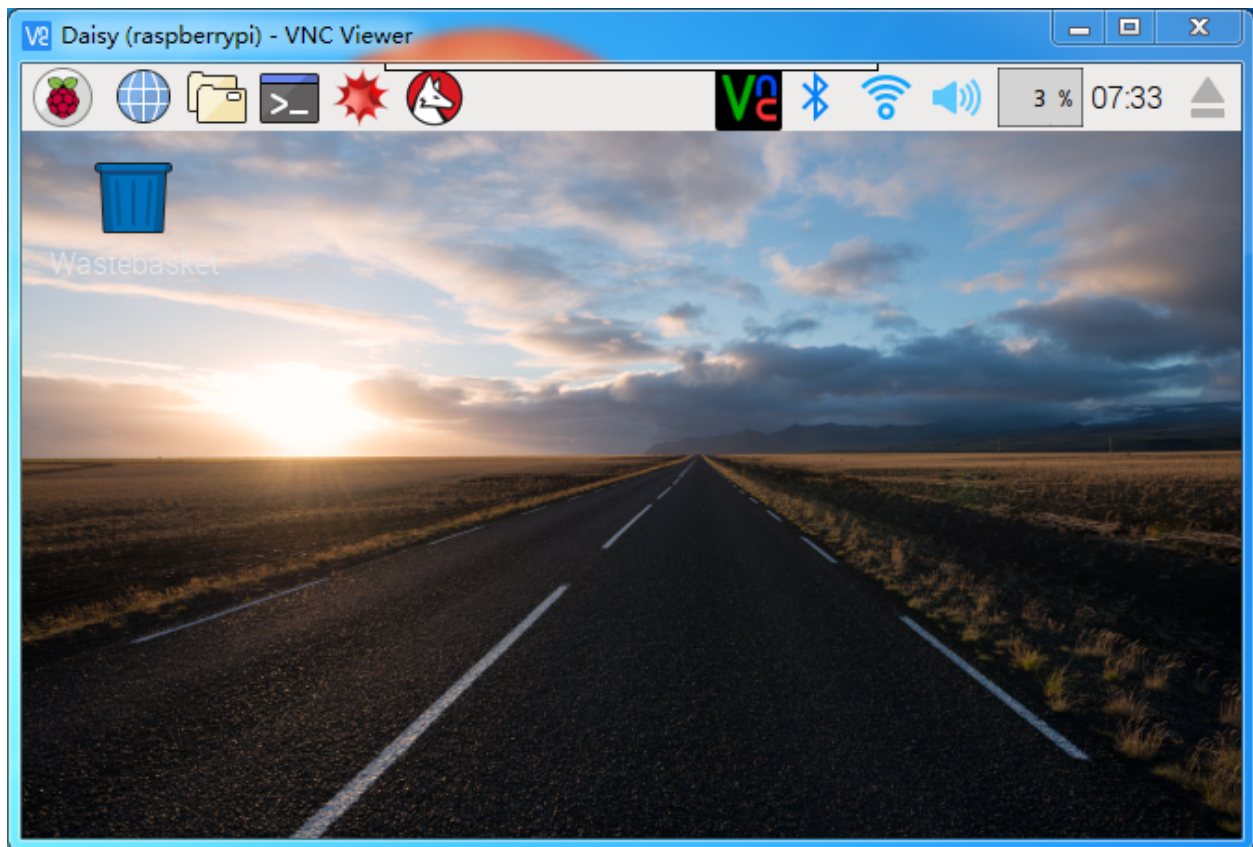
Double click the **connection** just created:

**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

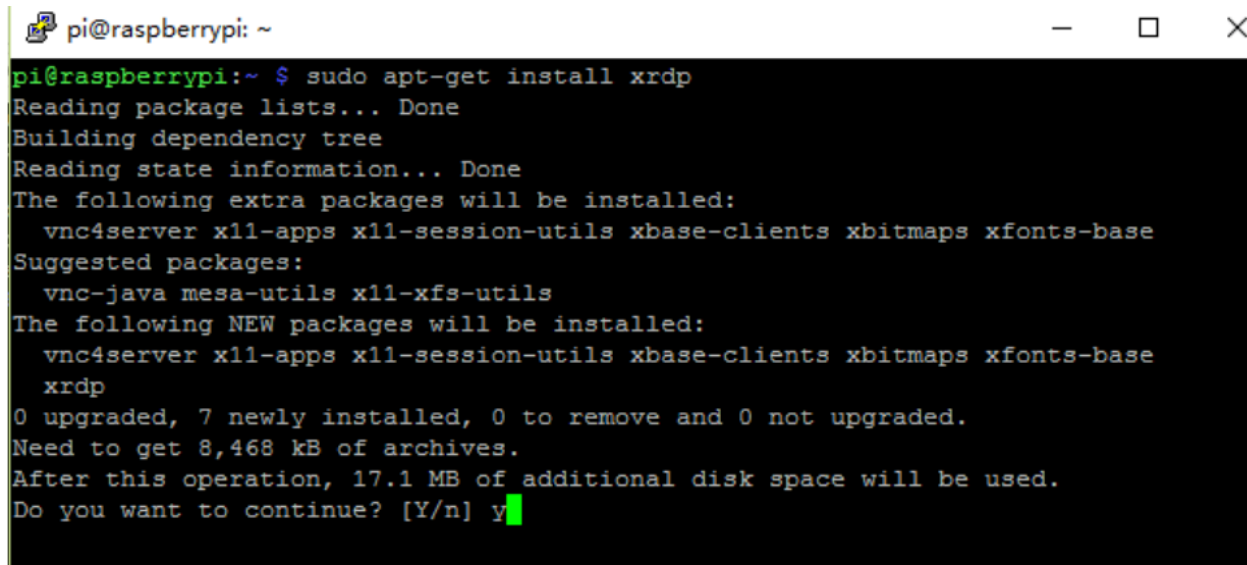
Input the following instructions to install XRDP.

```
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.

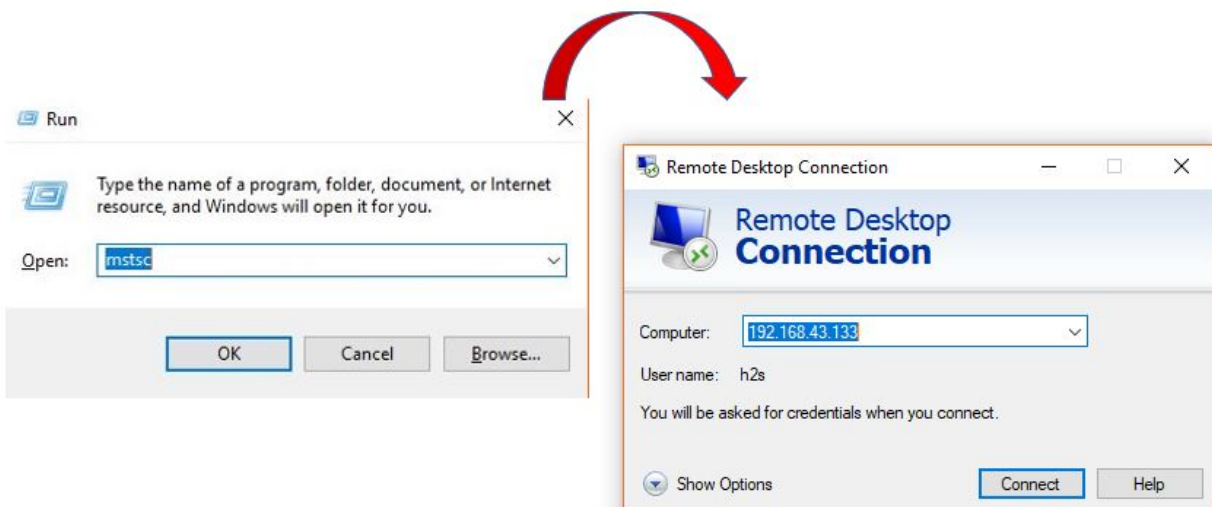
Login to XRDP

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.

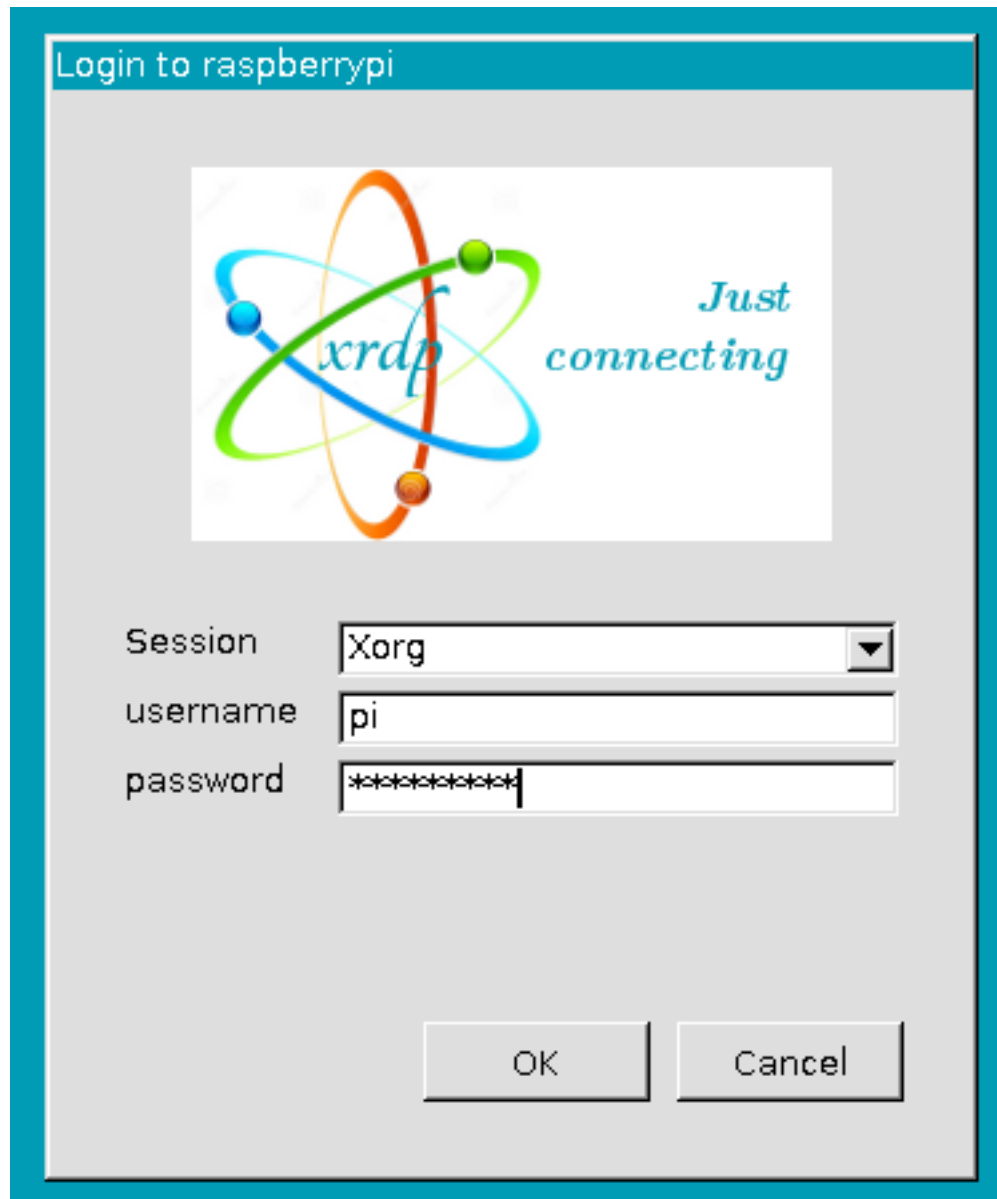
Step 2

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”.



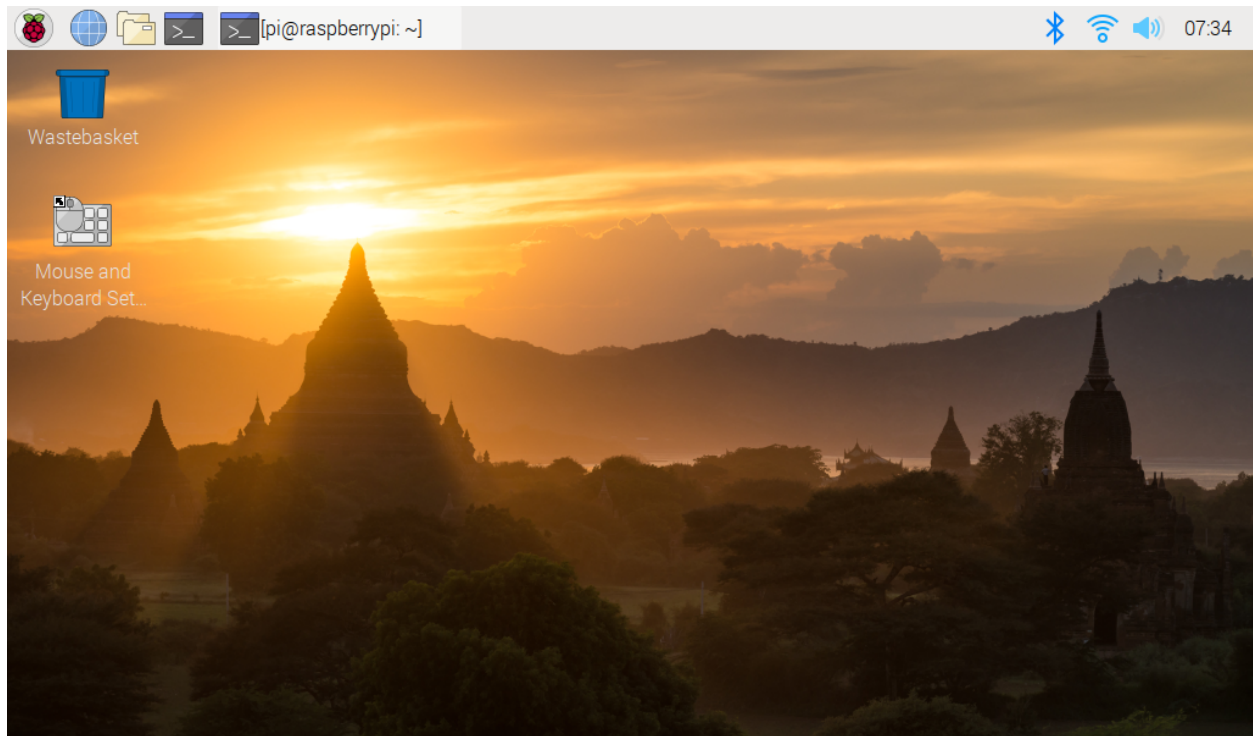
Step 3

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

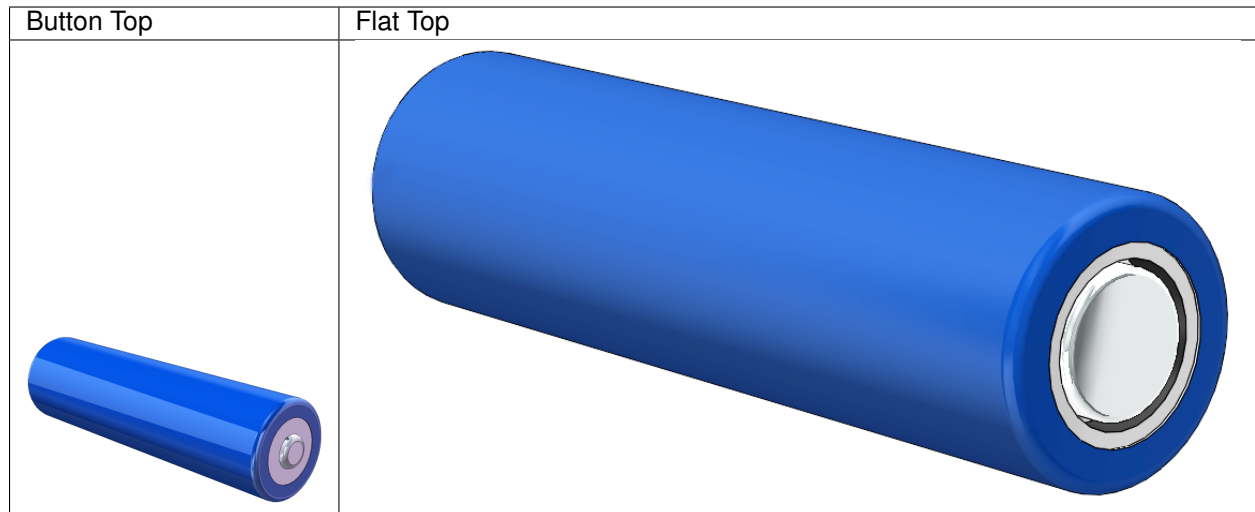
- 3.7V
- 18650
- Rechargeable
- Li-ion Battery
- Button Top
- No Protective Board

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.

THANK YOU

Thanks to the evaluators who evaluated our products, the veterans who provided suggestions for the tutorial, and the users who have been following and supporting us. Your valuable suggestions to us are our motivation to provide better products!

Particular Thanks

- Len Davisson
- Kalen Daniel
- Juan Delacosta

Now, could you spare a little time to fill out this questionnaire?

Note: After submitting the questionnaire, please go back to the top to view the results.

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.