SunFounder PiCrawler Kit

www.sunfounder.com

Aug 02, 2022

CONTENTS

Thank you for choosing our Pan-tilt HAT.

Pan-Tilt HAT is a 2-axis Pan-Tilt kit for intelligent detection and free movement based on the Raspberry Pi. Built in a 5 megapixel camera and the Raspberry Pi extension board with 14 external interfaces, Pan-Tilt HAT can realize functions such as color detection, face detection, gesture detection and traffic sign detection, and can freely control the camera's turning around. The free turning of the Pan-Tilt is mainly realized by two servos. The movable angle ranges 0~180° from left to right, and 0~90° from up to down. Pan-Tilt HAT has 14 external interfaces, including 4 Analog ports, 4 PWM ports, 4 Digital ports, 2 I2C ports, etc.

This document includes the list and assembly pdf and programming.

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

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

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

Content

CHAPTER ONE

COMPONENT LIST AND ASSEMBLY INSTRUCTIONS

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

Please follow the steps on the PDF to assemble.

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

- `Component List and Assembly Instructions.`

CHAPTER

TWO

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*.

2.1 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.

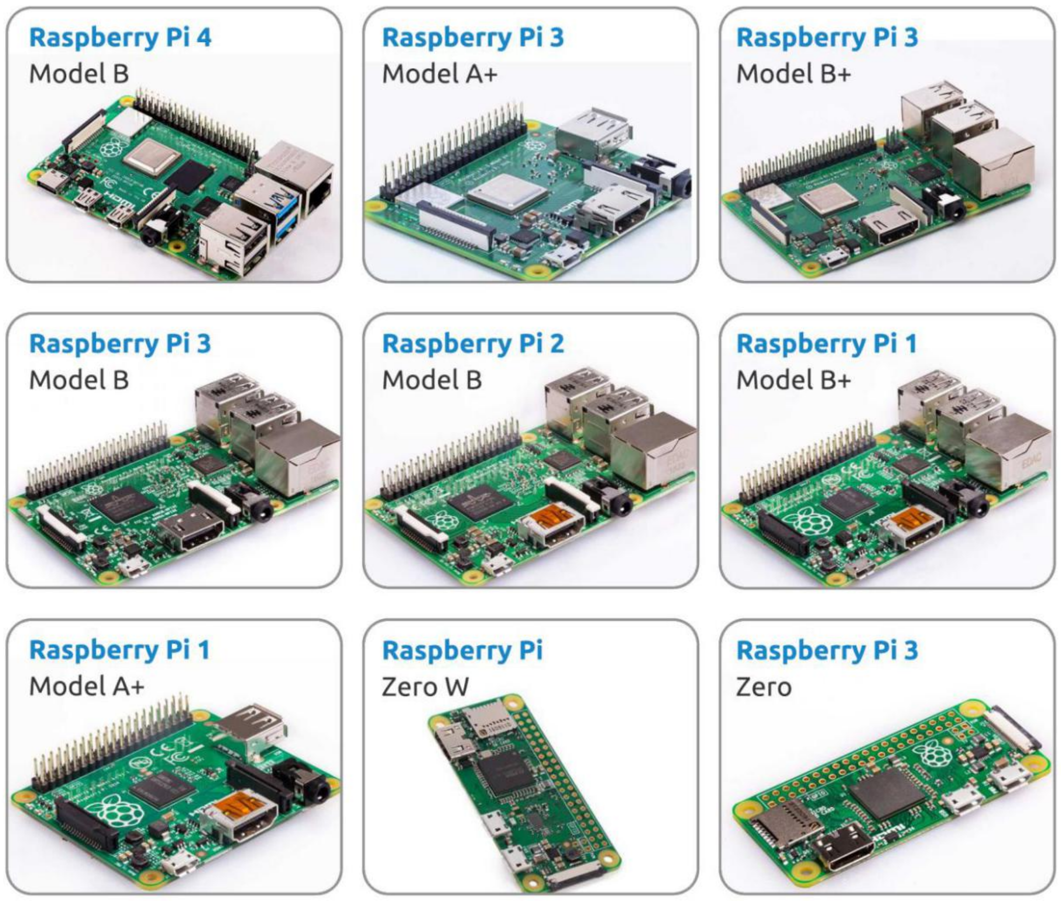
2.1.1 What Do We Need?

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.

2.1.2 Installing the OS

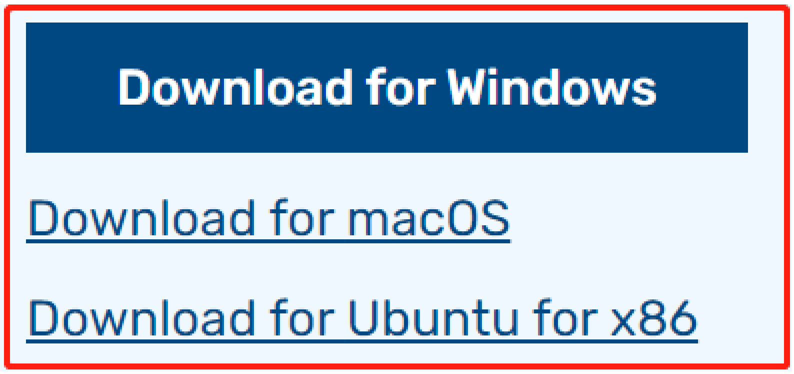
Required Components

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.

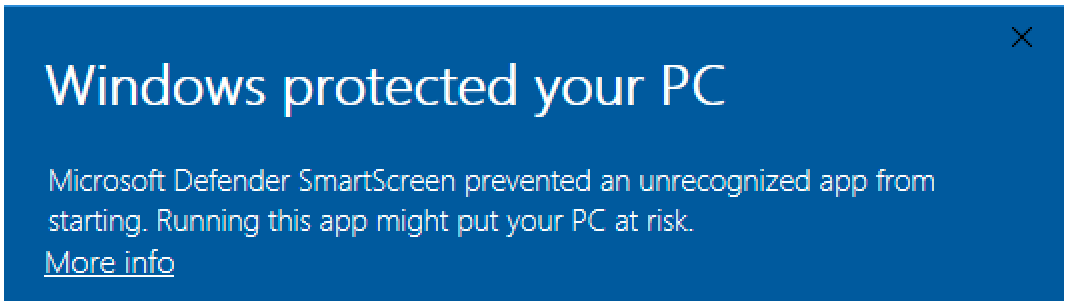
Visit the download page: https://www.raspberrypi.org/software/. Click on the link for the Raspberry Pi Imager that matches your operating system, when the download finishes, click it to launch the installer.



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.



Windows protected your PC

Microsoft Defender SmartScreen prevented an unrecognized app from starting. Running this app might put your PC at risk.
More info

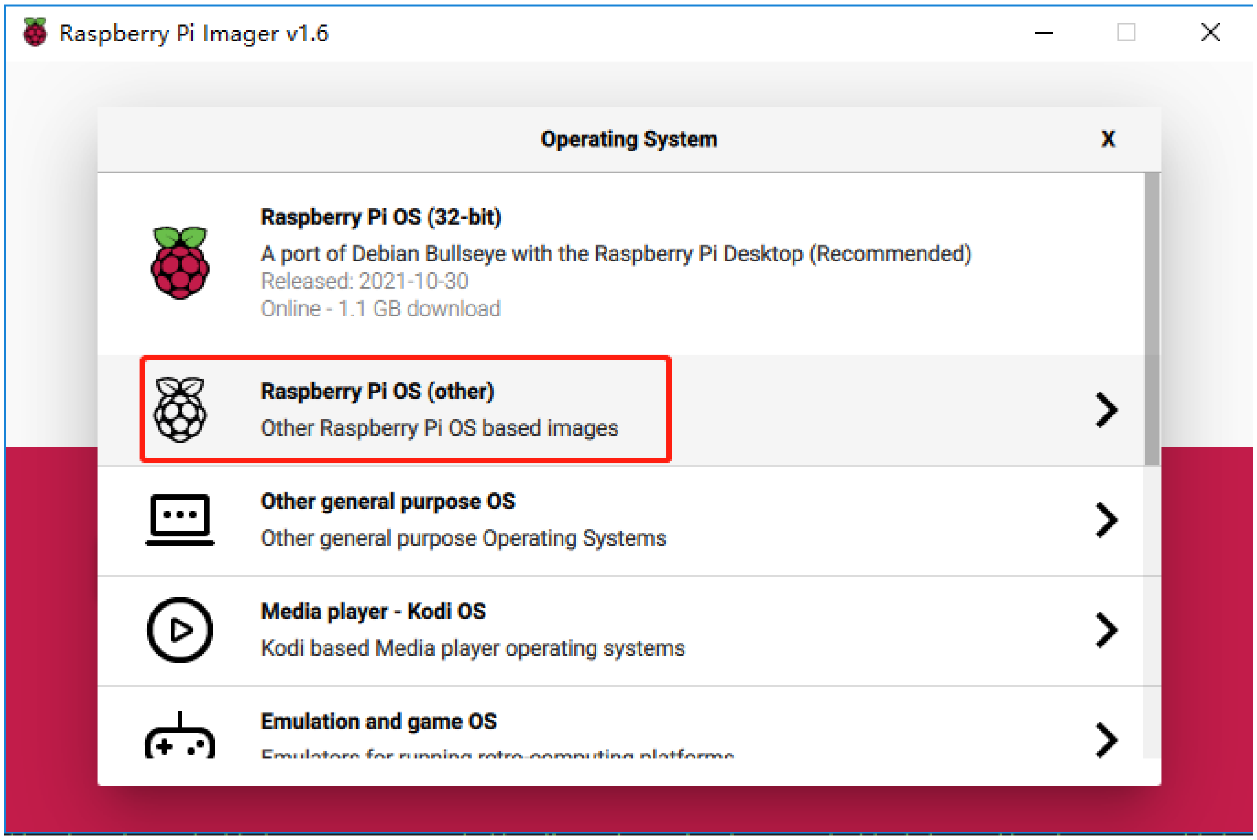
Step 3

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

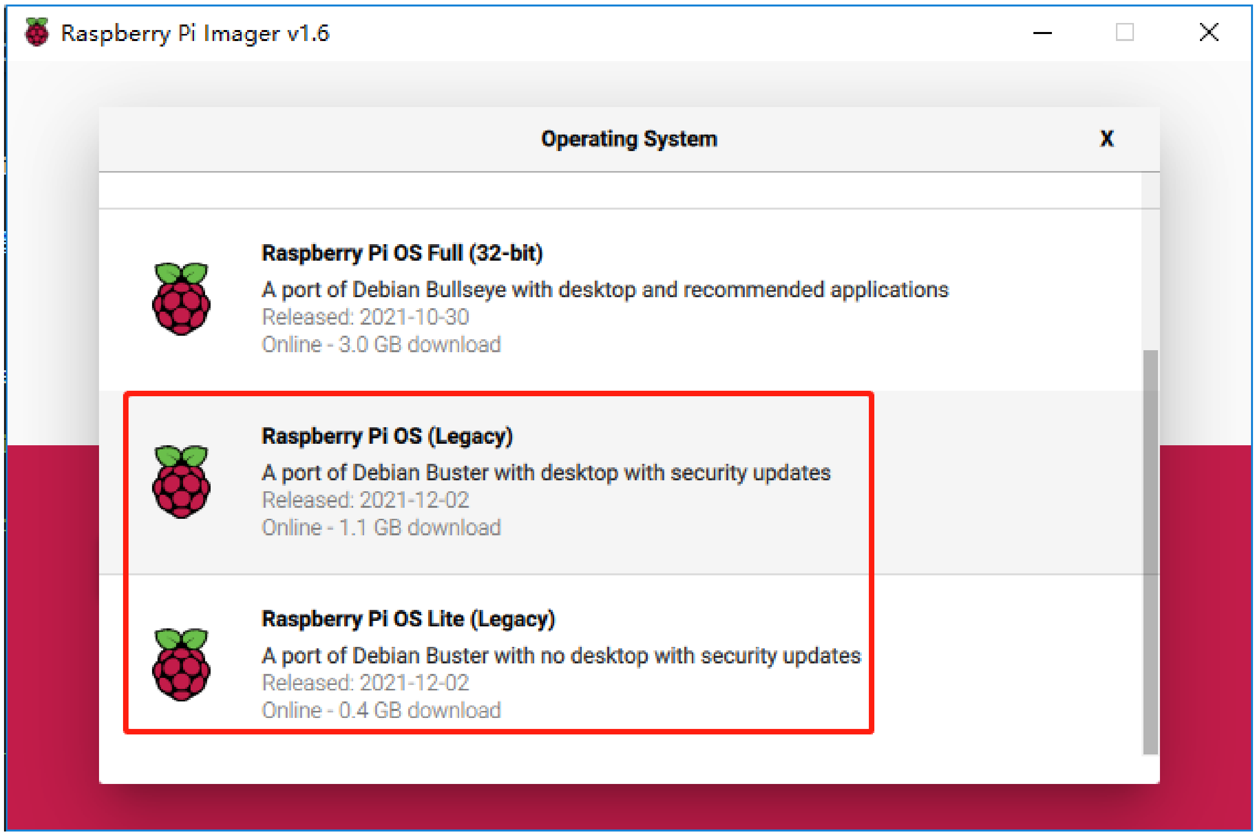
Step 4

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.


Raspberry Pi Imager v1.6

Operating System

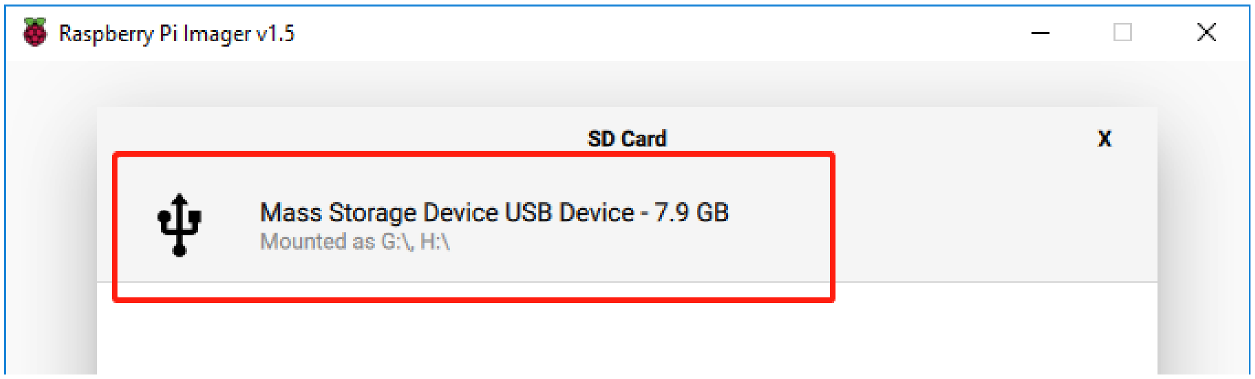
Raspberry Pi OS Full (32-bit)
A port of Debian Bullseye with desktop and recommended applications
Released: 2021-10-30
Online - 3.0 GB download

Raspberry Pi OS (Legacy)
A port of Debian Buster with desktop with security updates
Released: 2021-12-02
Online - 1.1 GB download

Raspberry Pi OS Lite (Legacy)
A port of Debian Buster with no desktop with security updates
Released: 2021-12-02
Online - 0.4 GB download

Step 5

Select the SD card you are using.


Raspberry Pi Imager v1.5

SD Card

Mass Storage Device USB Device - 7.9 GB
Mounted as G:\, H:\

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.

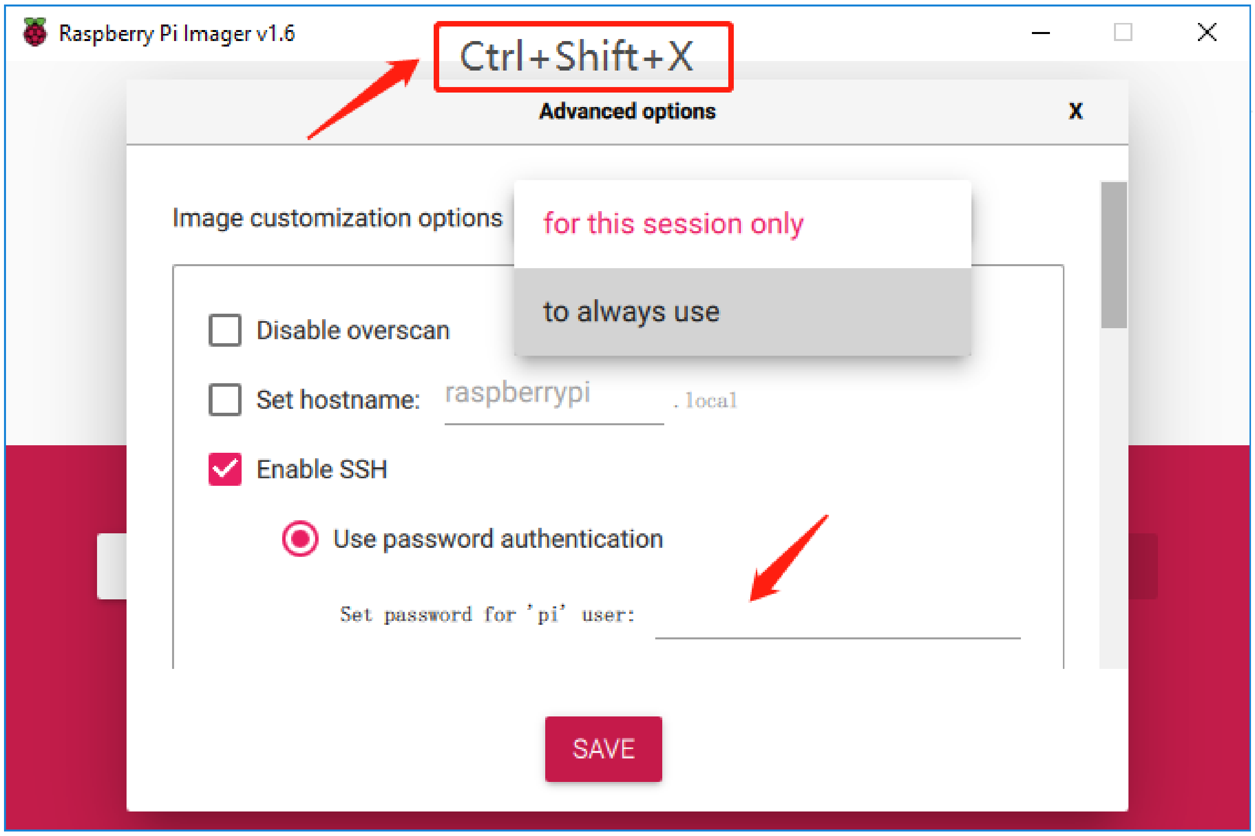
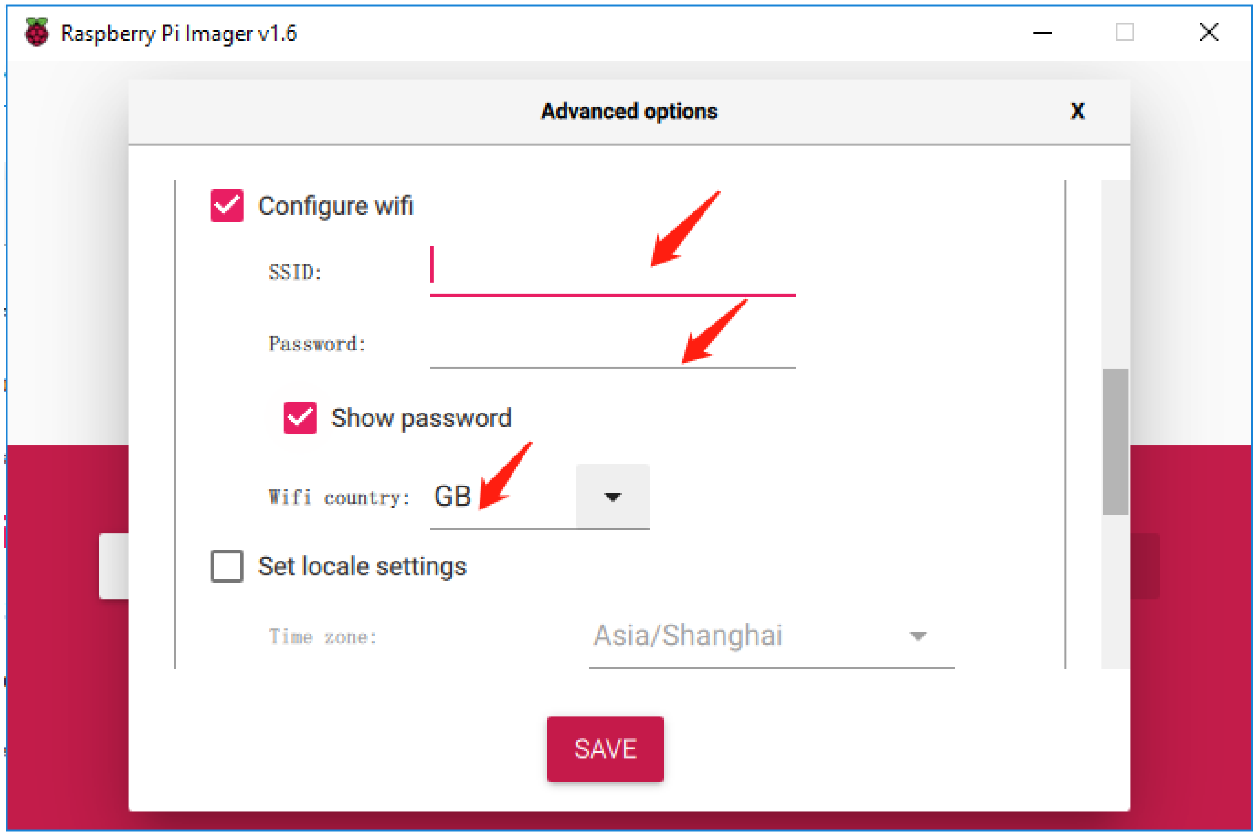


Image customization options

- [ ] Disable overscan
- [ ] Set hostname: raspberrypi .local
- [✓] Enable SSH
  - [✓] Use password authentication
  - Set password for 'pi' user:

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

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



Configure wifi [✓]

SSID:

Password:

Show password [✓]

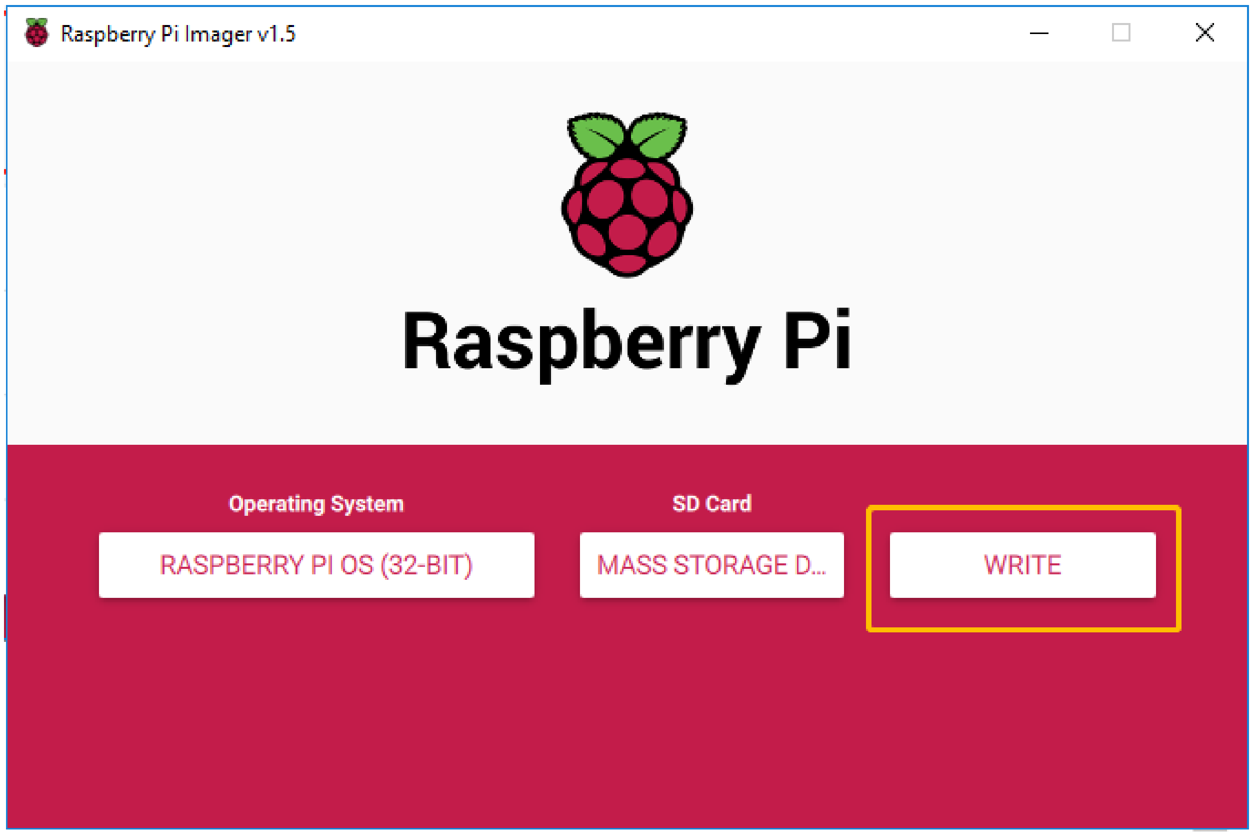
Wifi country: GB

Set locale settings [ ]

Time zone: Asia/Shanghai

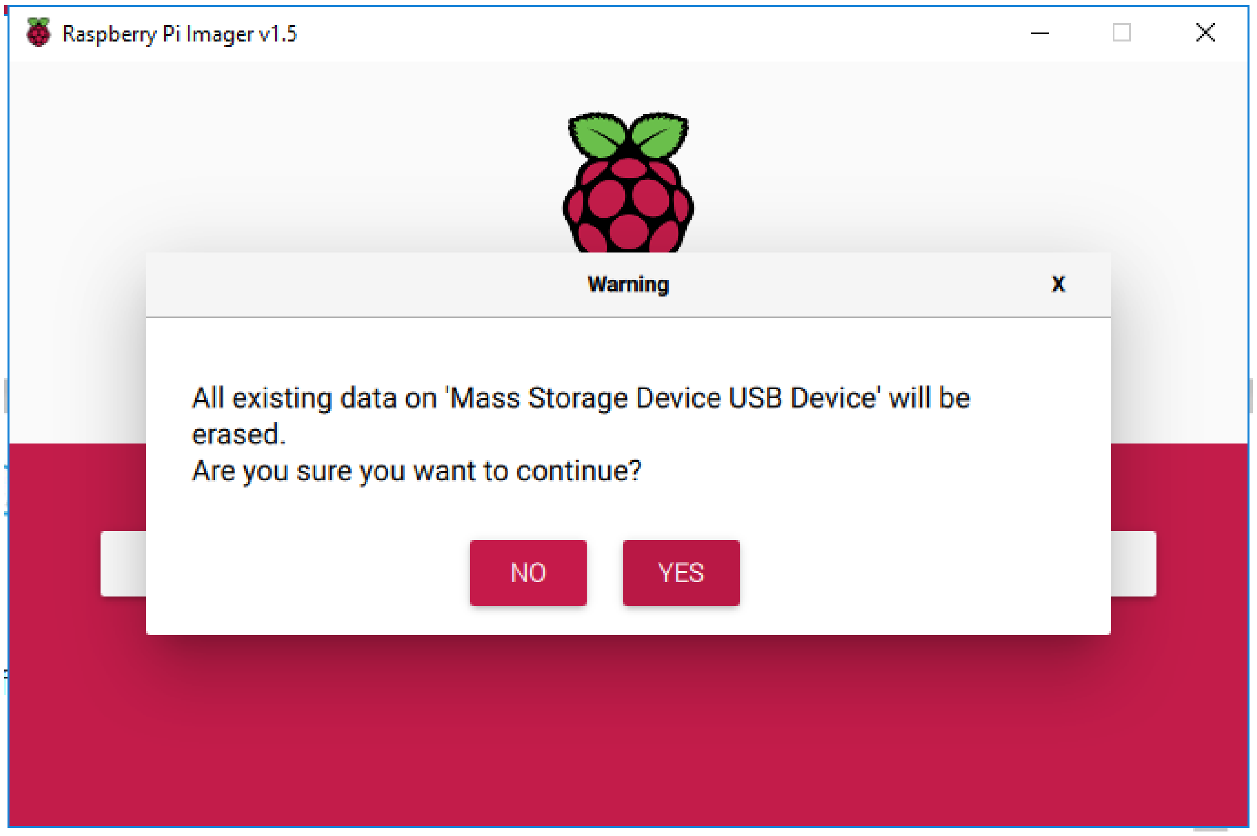
Step 7

Click the **WRITE** button.



Step 8

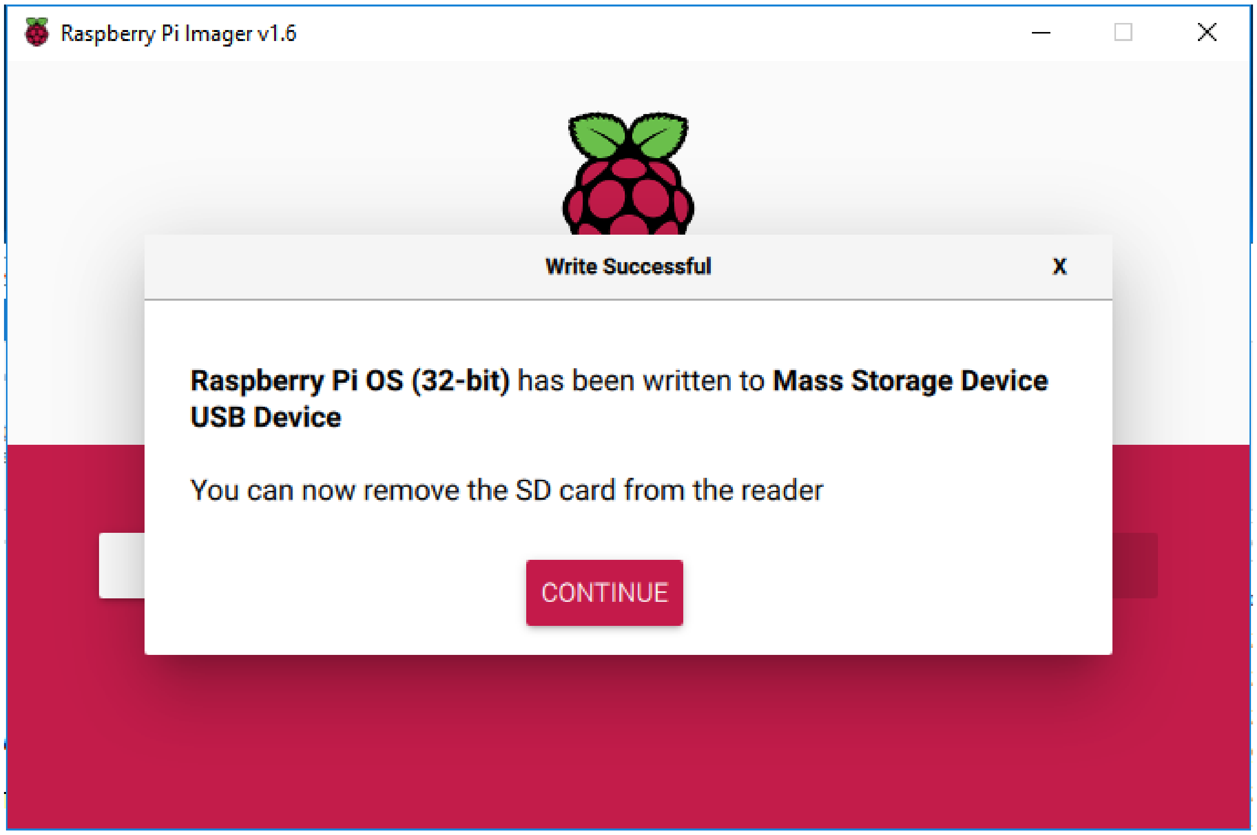
If your SD card currently has any files on it, you may wish to back up these files first to prevent you from permanently losing them. If there is no file to be backed up, click **Yes**.



All existing data on 'Mass Storage Device USB Device' will be erased.
Are you sure you want to continue?

Step 9

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



Raspberry Pi OS (32-bit) has been written to **Mass Storage Device USB Device**

You can now remove the SD card from the reader

2.1.3 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.

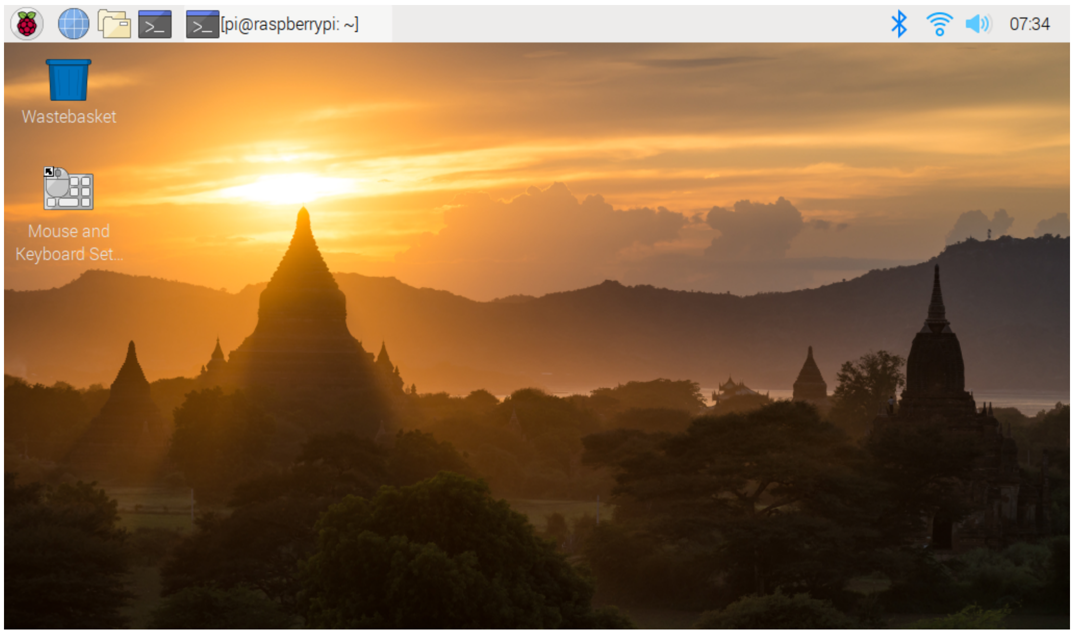
Required Components

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.

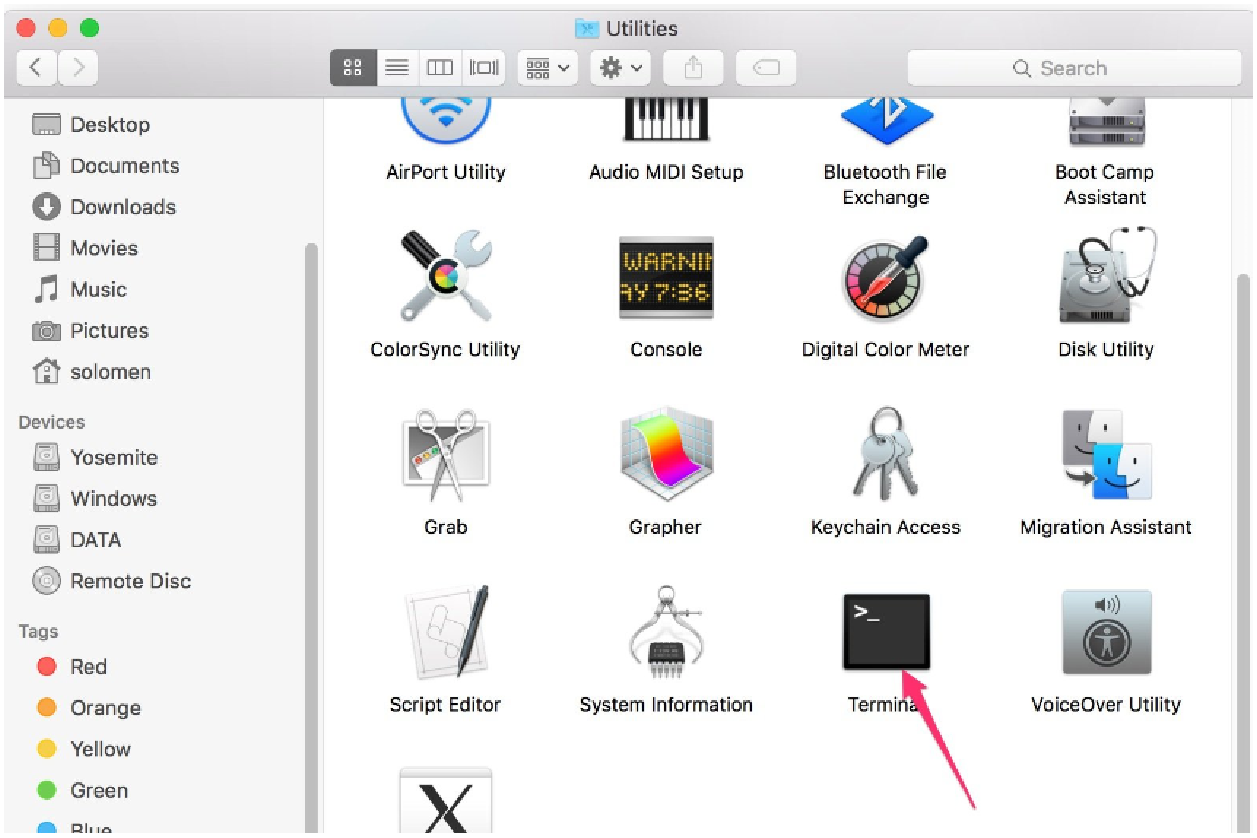
Use the SSH Remote Control

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.



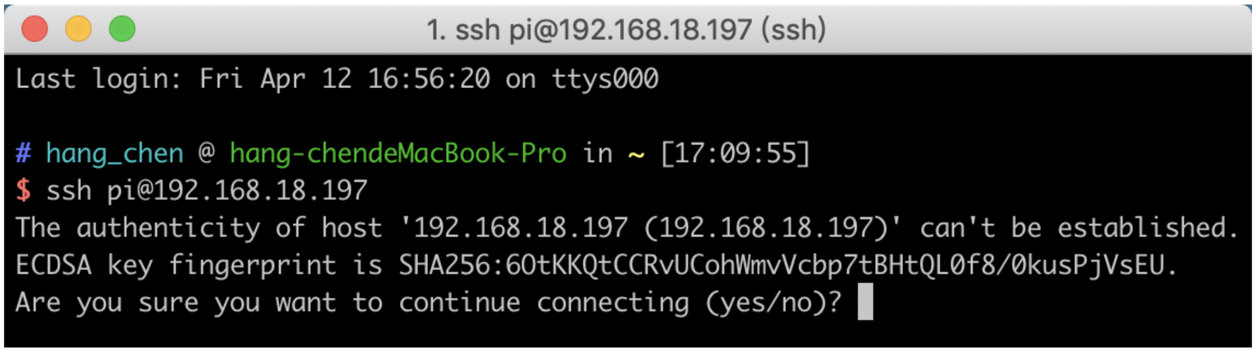
Step 2

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

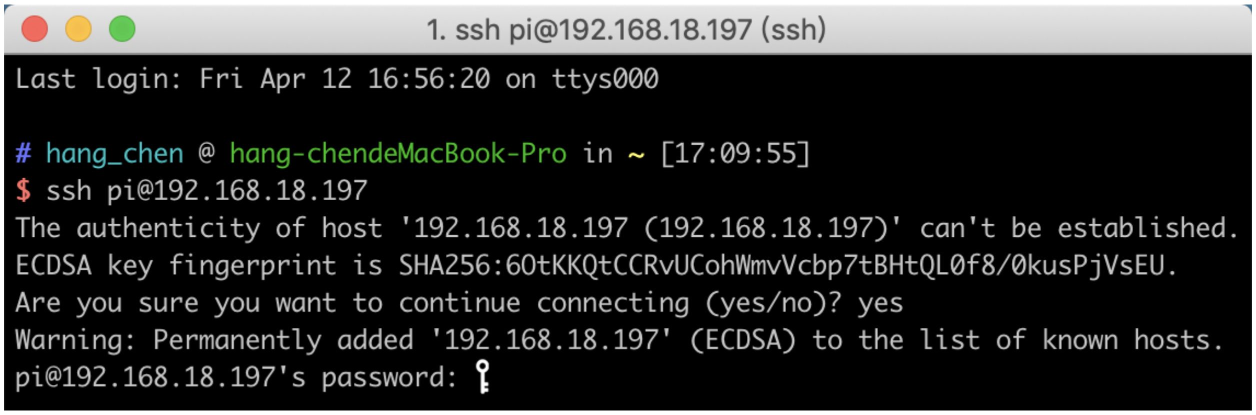
Input"yes".



```
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 SHA256:6OtKKQtCCRvUCohWmvVcbp7tBHtQL0f8/0kusPjVsEU.
Are you sure you want to continue connecting (yes/no)?
```

Step 4

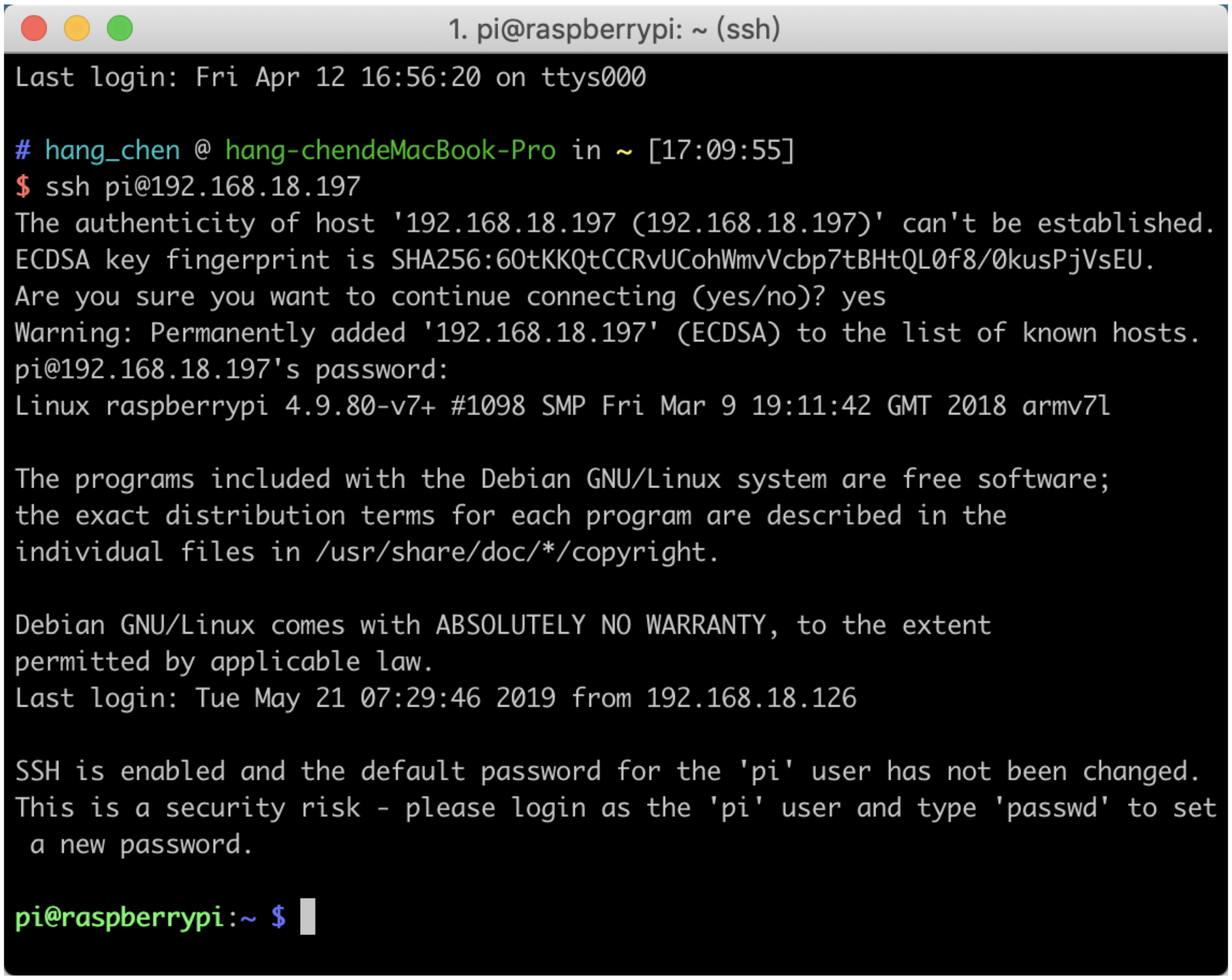
Input the passcode and the default password is **raspberry**.



```
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 SHA256:6OtKKQtCCRvUCohWmvVcbp7tBHtQL0f8/0kusPjVsEU.
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.



```
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 SHA256:6OtKKQtCCRvUCohWmvVcbp7tBHtQL0f8/0kusPjVsEU.
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

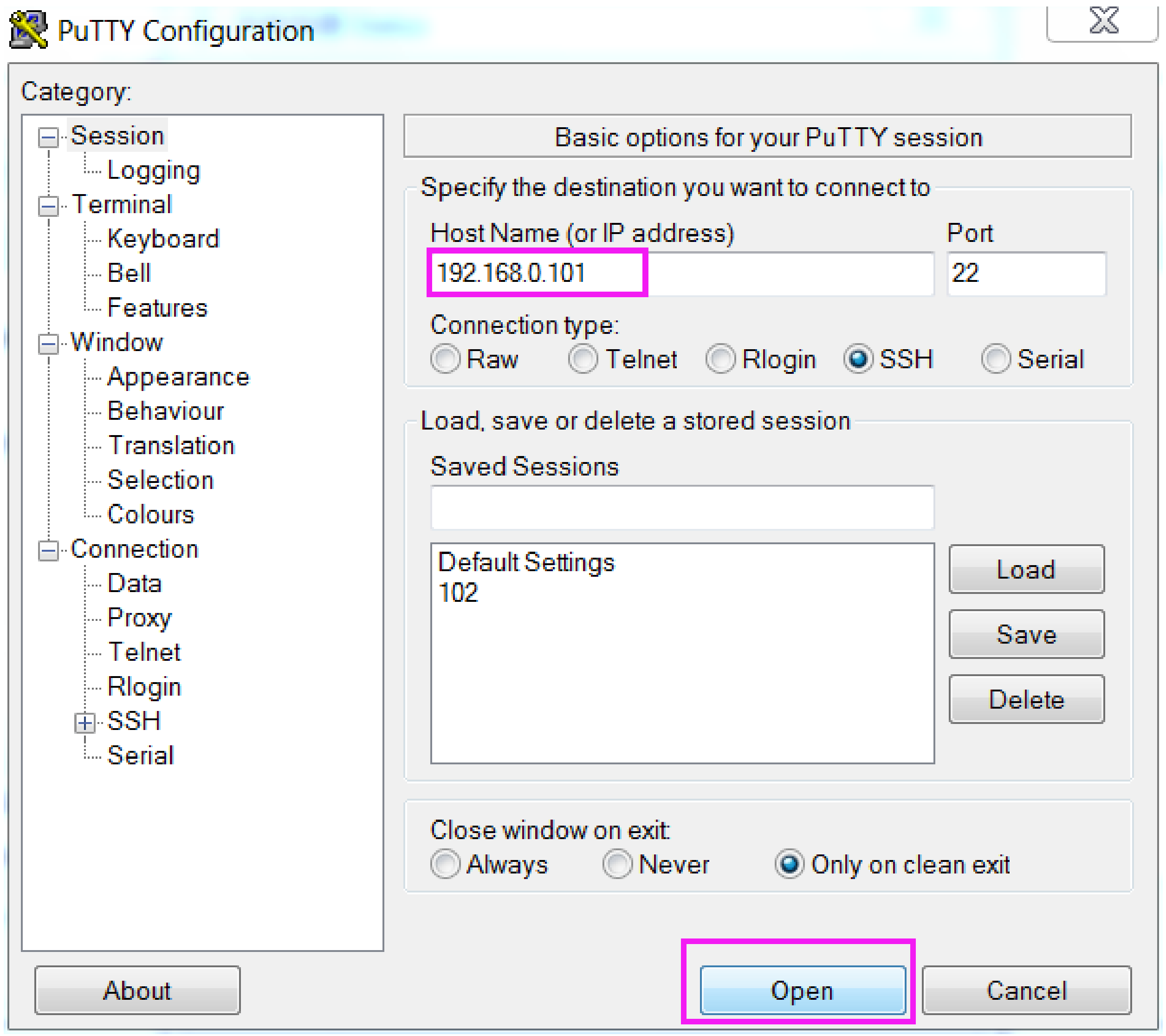
If you're a Windows user, you can use SSH with the application of some software. Here, we recommend **PuTTY**.

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).



Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): 192.168.0.101 Port: 22

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

Load, save or delete a stored session

Saved Sessions

Default Settings
102

Close window on exit: Always Never [✓ Only on clean exit]

Step 3

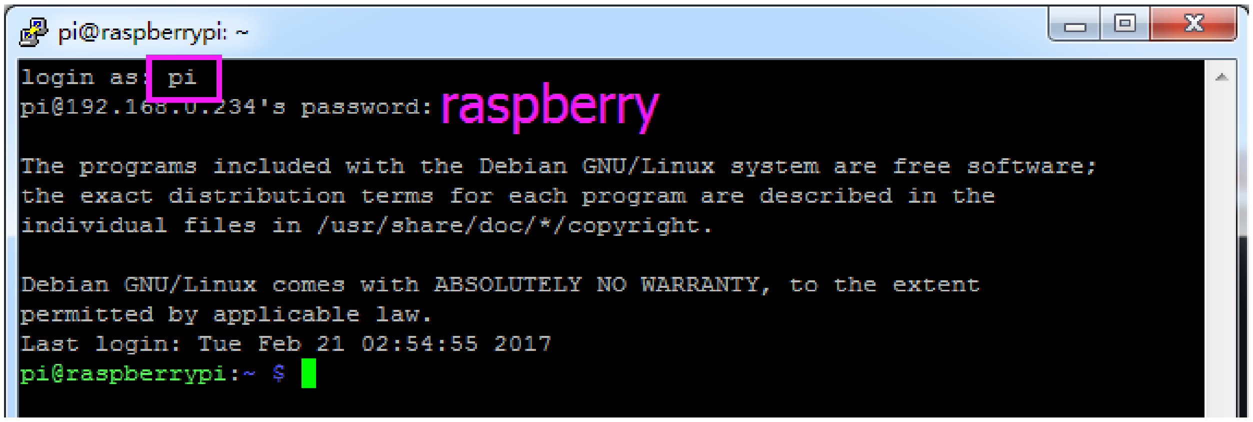
Click **Open**. Note that when you first log in to the Raspberry Pi with the IP address, there prompts a security reminder. Just click **Yes**.

Step 4

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

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

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



```
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*.

2.1.4 Download and Run the Code

Download the Code Install Libraries

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

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

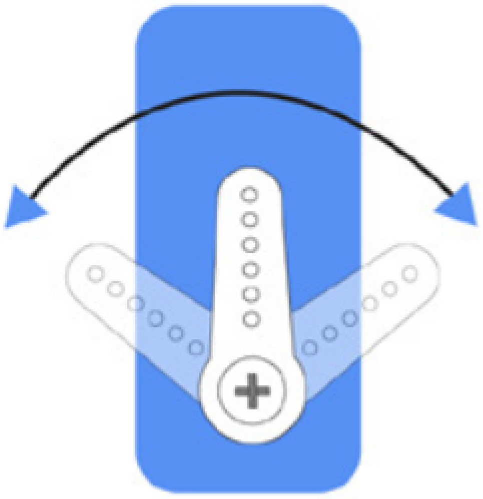
Note: Running `install.py` will download some necessary components. You may fail to download due to network problems. You may need to download again at this time.

Run `servo_zeroing.py`

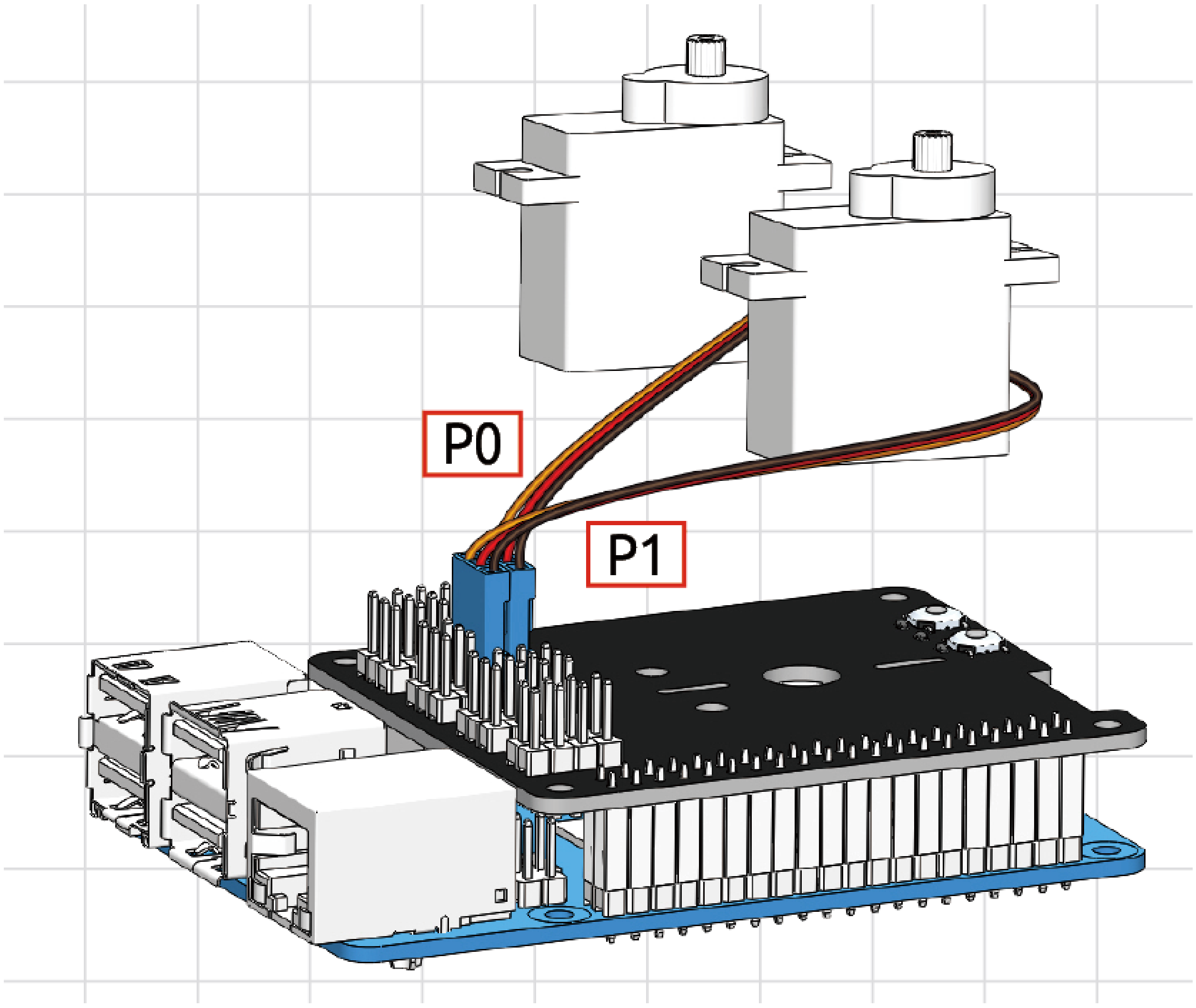
Run the `servo_zeroing.py` in the `examples/` folder.

```
cd /home/pi/pan-tilt-hat/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 P0/P1 position.



Now if the servo arm shifts and stops at a specific position, the function will take effect. If it is not, please check the insertion direction of the servo cable and re-run the code.

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

2.2 Take Photo

This is the first python example of Pan-tilt HAT.

After the program runs: 1. You can enter `http://<Your Raspberry Pi IP>:9000/mjpg` in the browser (such as chrome) to view the viewfinder screen. 2. Type `q` in Terminal and press Enter to take a photo.

Run the Code

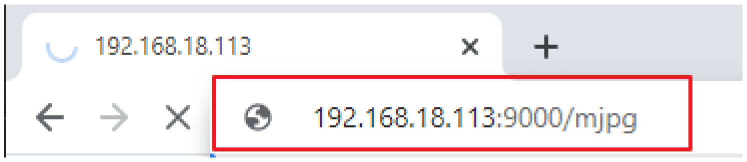
```
cd /home/pi/pan-tilt-hat/examples
sudo python3 take_photo.py
```

View the Image

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

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

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



Code

```python
from vilib import Vilib

def main():
    Vilib.camera_start(vflip=True,hflip=True)
    Vilib.display(local=True,web=True)

    path = "/home/pi/Pictures/"

    while True:
        key = input()
        if key == "q":
            Vilib.take_photo(photo_name="new_photo",path=path)
            print("Take Photo")
        if key == "g":
            Vilib.camera_close()
            break

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

How it works?

The content involved in this article is exactly the basic function of the `vilib` library. We have already installed this library in *Download and Run the Code*.

What you need to focus on is the following:

```
from vilib import Vilib
```

All functions related to computer vision are encapsulated in this library.

```
Vilib.camera_start(vflip=True,hflip=True)
```

Let the camera module enter the working state. If you modify the two parameters of this function to `False`, the screen will be flipped horizontally/vertically.

```
Vilib.camera_close()
```

Stop the camera module.

```
Vilib.display(local=True,web=True)
```

Allows you to see the picture taken by the camera module. Its parameter `local=True` is used to open the viewfinder in the Raspberry Pi desktop, which is suitable for *Remote Desktop* or the situation where a screen is provided for the Raspberry Pi. The parameter `web=True` allows you to view the image through the browser, which is the method suggested in this article. It is suitable for the situation where your PC and Raspberry Pi are connected to the same local area network.

```
Vilib.take_photo(photo_name="new_photo",path="/home/pi/Pictures/")
```

As the name suggests, this function takes pictures. The first parameter is the name of the generated image file, and the second parameter is the path where the file is located. They can all be customized.

Please see the next chapter for how to move and view the photos you have taken.

2.3 Check Photos

An easy way is to use *Filezilla Software* to drag and drop the photo files to the PC. However, its viewing experience is not very good. We recommend that you use Samba to turn the picture folder into an album that can be used in the local area network.

Install Samba

1. Run the command to set up Samba service.

```
sudo apt-get update
sudo apt-get install samba samba-common-bin
```

2. Configure Samba typing.

```
sudo nano /etc/samba/smb.conf
```

Note: Press ctrl+o to save what you modify in nano editor, ctrl+x to to exit.

Input the following content at the end of the file:

```
[share]
path = /home/pi/Pictures/ #This is your album path.
valid users = pi
browseable = yes
public = yes
writable = yes
```

3. Restart Samba service.

```
sudo service smbd restart
```

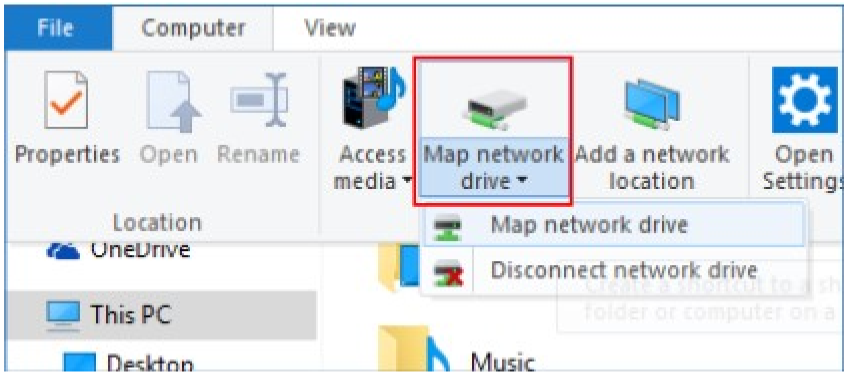
4. Add sharing account.

```
sudo smbpasswd -a pi
```

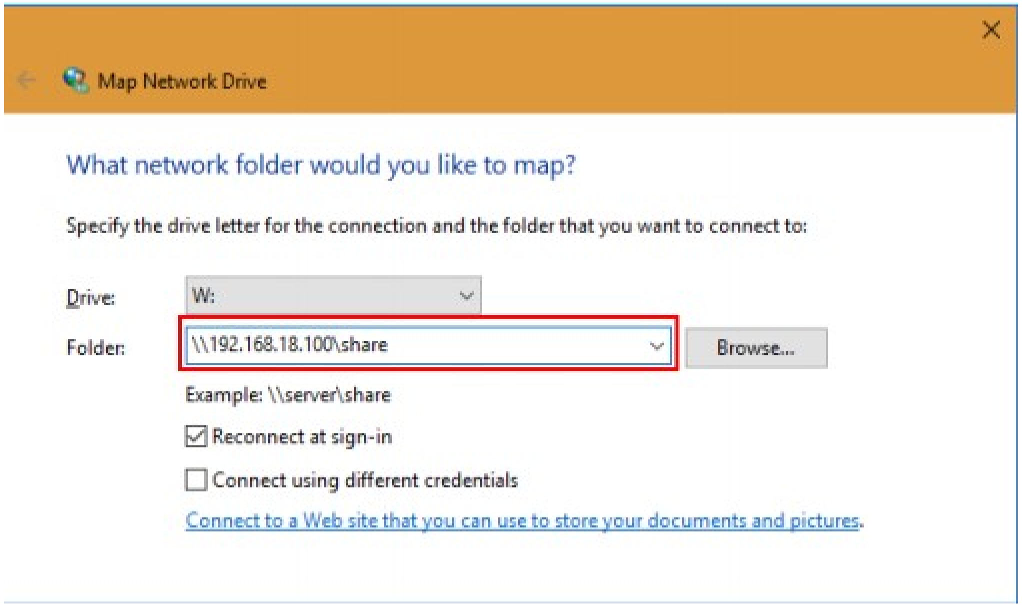
Note: A sharing account "pi" is created and you need to set your passcode.

Mount Photo Album to Windows

Under This PC, click **Map network drive**.



Type `\\hostname or IP address\the name of the shared files` in the path bar.


Map Network Drive

What network folder would you like to map?

Specify the drive letter for the connection and the folder that you want to connect to:

Drive: W:

Folder: \\192.168.18.100\share

Browse...

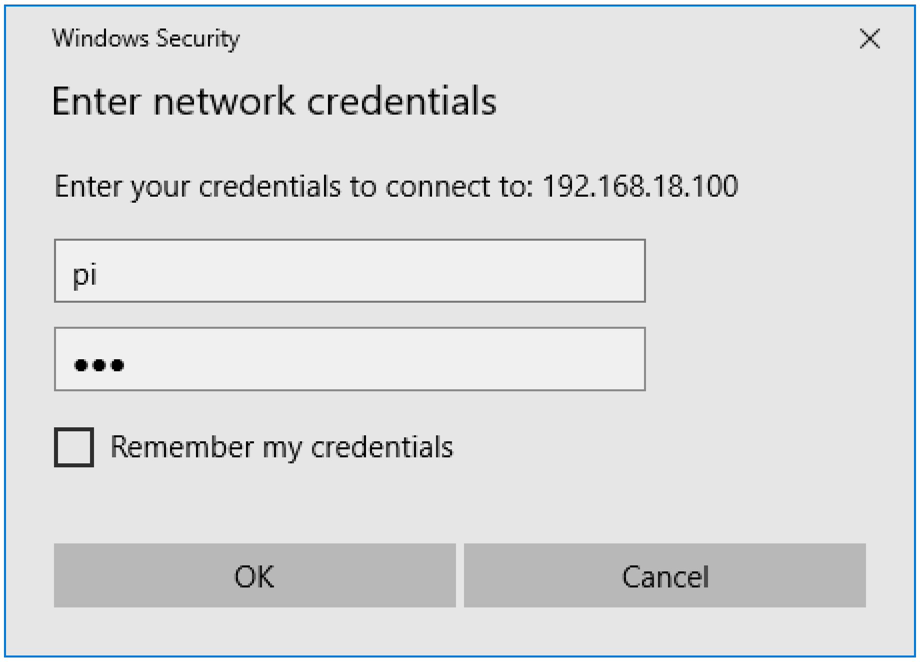
Example: \\server\share

☑ Reconnect at sign-in

☐ Connect using different credentials

Connect to a Web site that you can use to store your documents and pictures.

Type in the username and the password. Click OK button, and you can access the shared files.



Windows Security

Enter network credentials

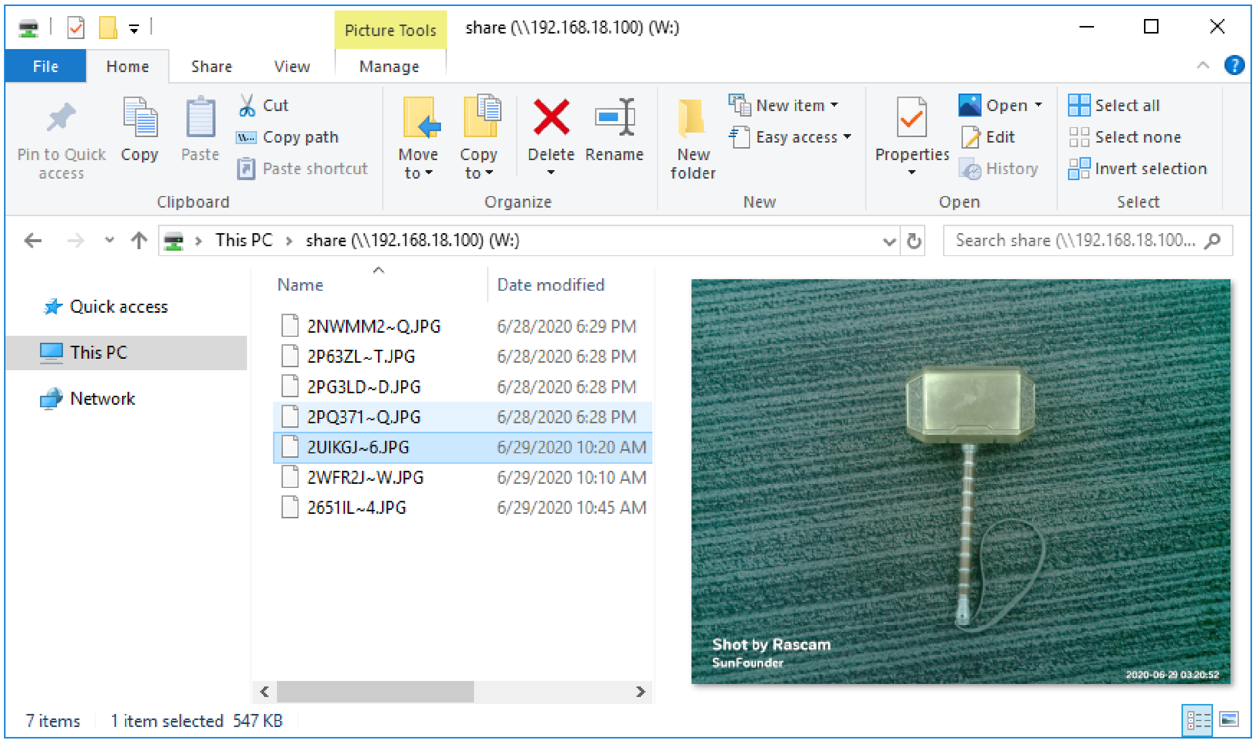
Enter your credentials to connect to: 192.168.18.100

pi

●●●

☐ Remember my credentials

The album will appear as a new volume under this PC.



share (\\192.168.18.100) (W:)

This PC > share (\\192.168.18.100) (W:)

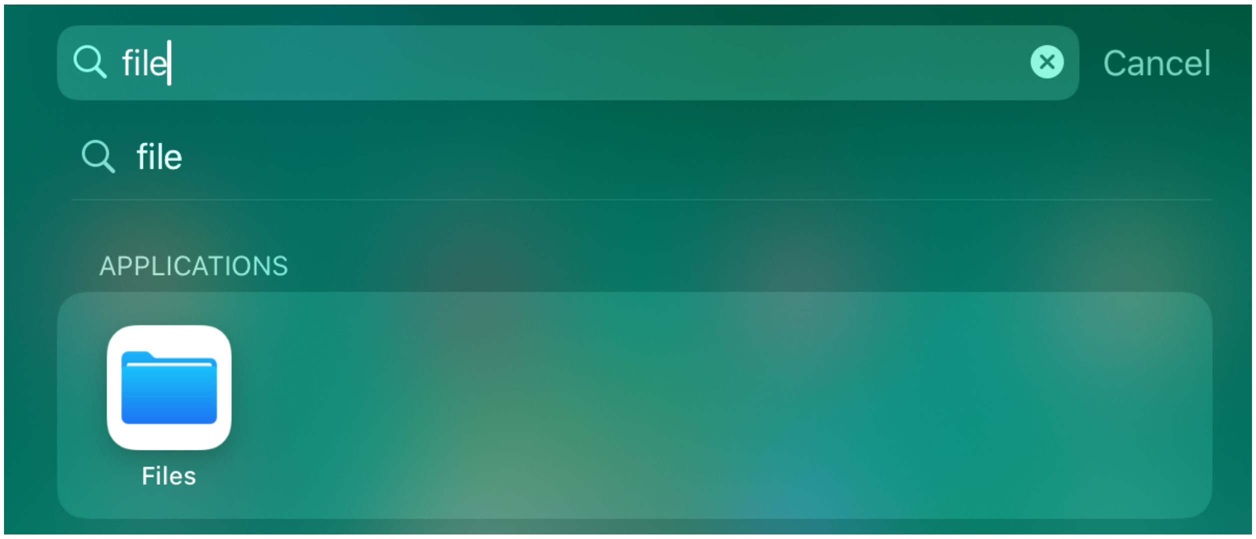
Name	Date modified
2NWMM2~Q.JPG	6/28/2020 6:29 PM
2P63ZL~T.JPG	6/28/2020 6:28 PM
2PG3LD~D.JPG	6/28/2020 6:28 PM
2PQ371~Q.JPG	6/28/2020 6:28 PM
2UIKGJ~6.JPG	6/29/2020 10:20 AM
2WFR2J~W.JPG	6/29/2020 10:10 AM
2651IL~4.JPG	6/29/2020 10:45 AM

7 items 1 item selected 547 KB

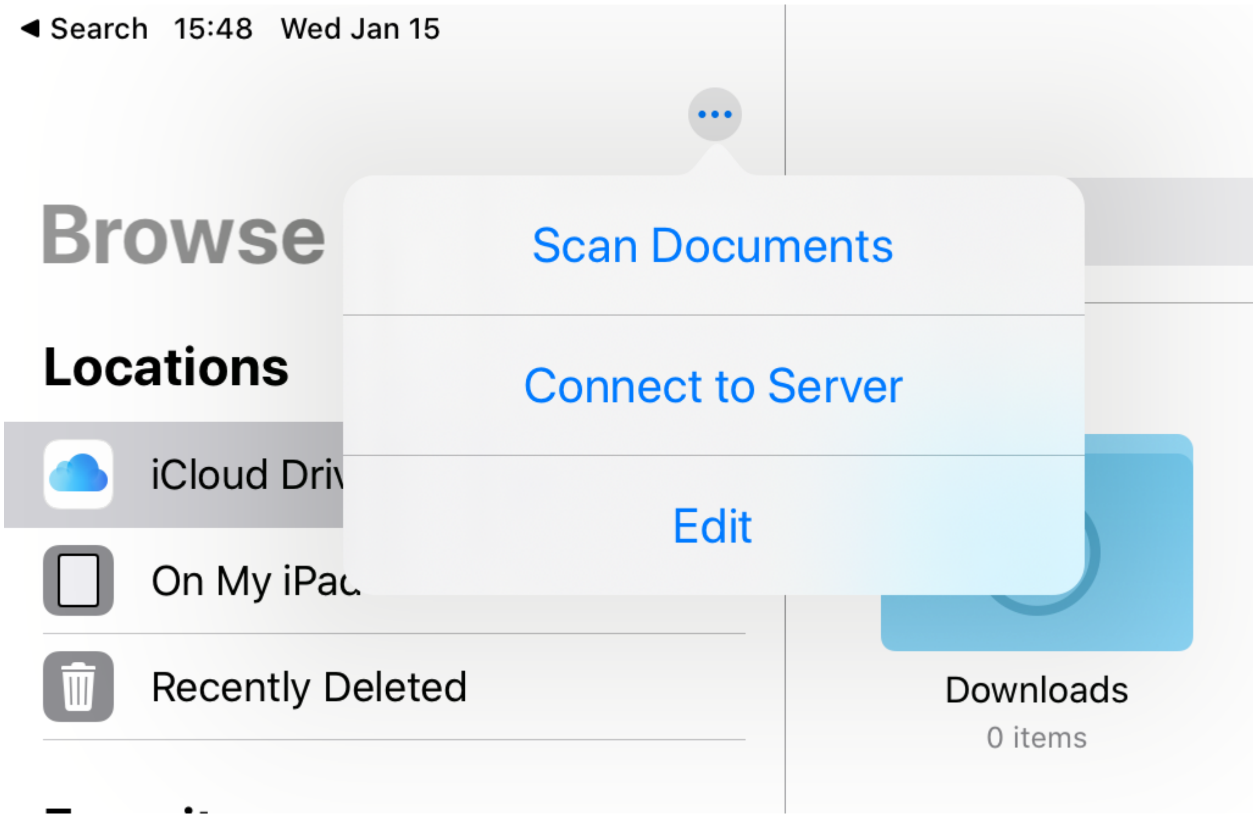
Mount Photo album to iOS

The system newer than **iOS 13.0** can directly mount storage in Files. The older versions of iOS can use APPs like DS File.

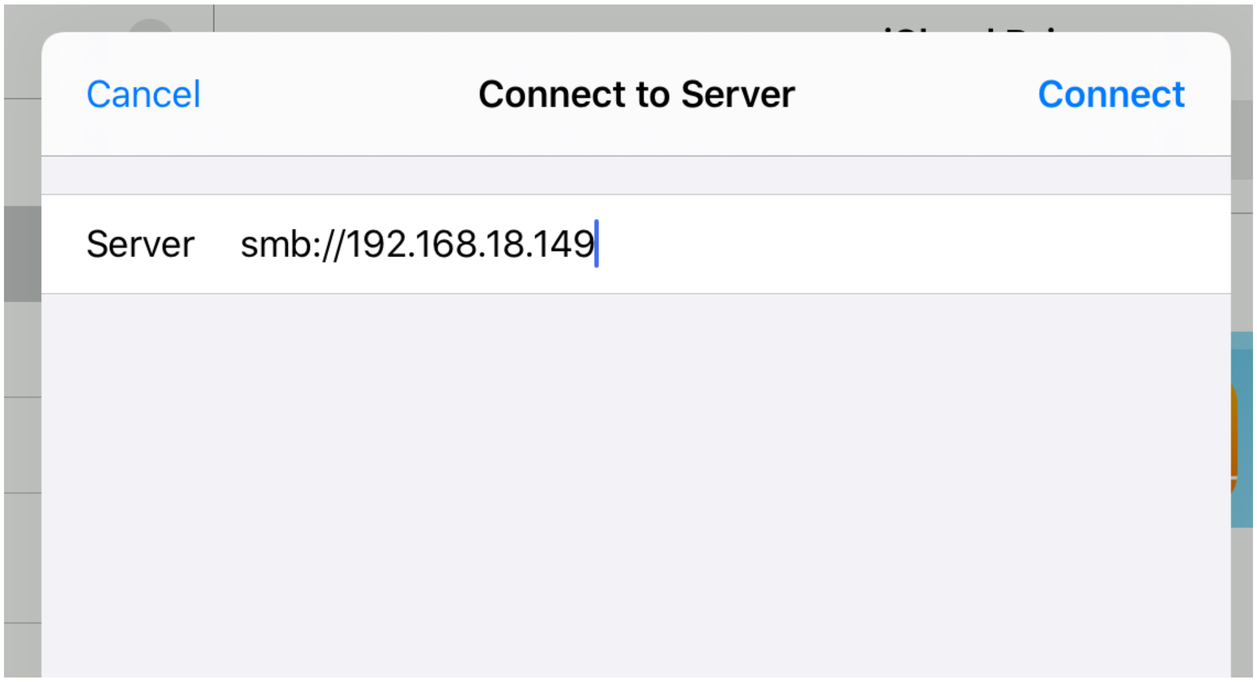
1. Open **Files**.



2. Tap **Connect to Server** in **Menu**.

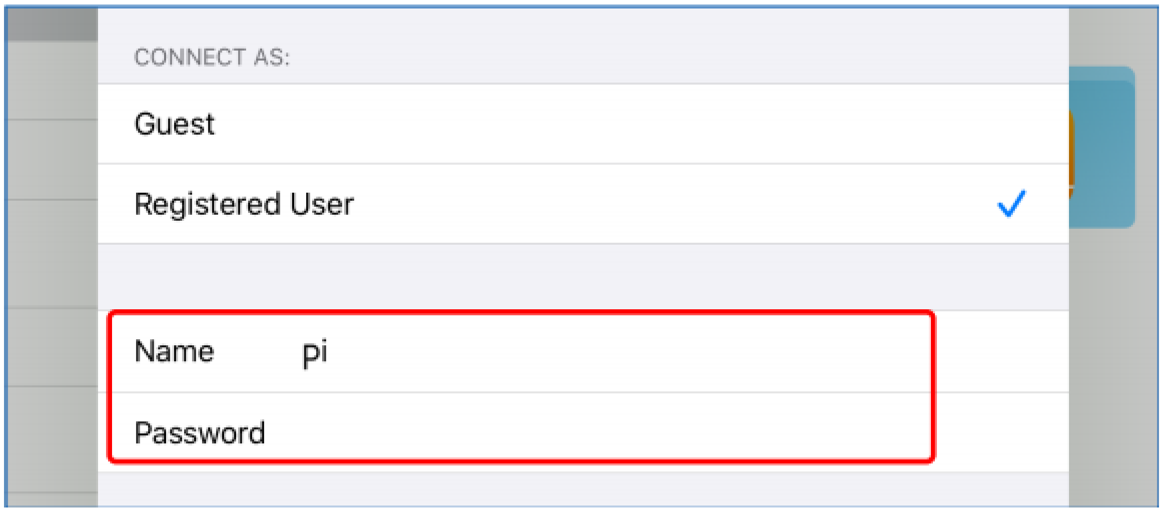


3. Enter your Rascam IP address.



Server smb://192.168.18.149

4. Log in.



CONNECT AS:

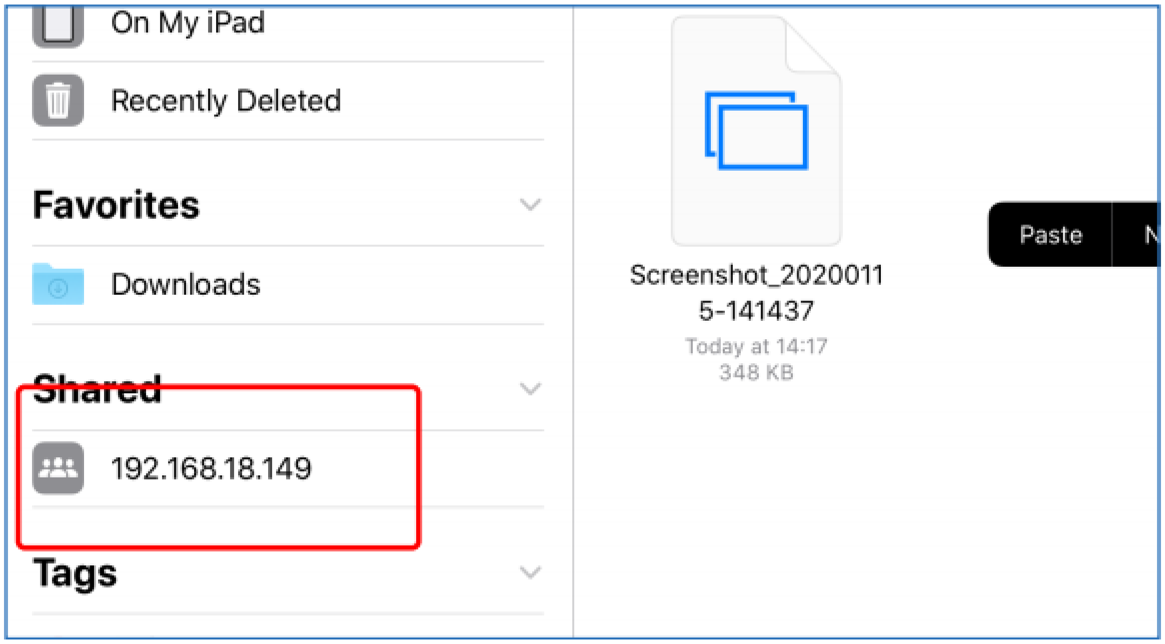
Guest

Registered User ✓

Name pi

Password

5. Now, you can directly access the photo album in File.



2.4 Continuous Shooting

This example allows us to use the pan-tilt for shooting/continuous shooting more conveniently.

By the way, a large number of still photos captured in continuous shooting can be used as frames to synthesize videos. This usage will be later in the chapter *Time Lapse Photography*.

Here you will use two windows at the same time: One is Terminal, where you will enter `wasd` to control the camera orientation, enter `q` to shoot, and enter `g` to exit shooting. If the program has not been terminated after exiting the shooting, please press `ctrl+c`. Another browser interface, after the program runs, you will need to enter `http://<Your Raspberry Pi IP>:9000/mjpg` in the PC browser (such as chrome) to view the viewfinder screen.

Run the Code

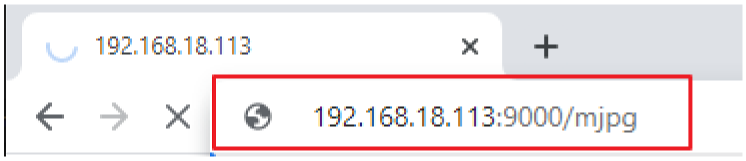
```
cd /home/pi/pan-tilt-hat/examples
sudo python3 continuous_shooting.py
```

View the Image

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

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

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



Code

```
from time import sleep,strftime,localtime
from vilib import Vilib
from sunfounder_io import PWM,Servo,I2C

import sys
import tty
import termios

# region  read keyboard
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 record value!
    W: up
    A: left
    S: right
    D: down
    Q: continuous_shooting

    G: Quit
'''
# endregion

# region init
I2C().reset_mcu()
sleep(0.01)

pan = Servo(PWM("P1"))
tilt = Servo(PWM("P0"))
panAngle = 0
tiltAngle = 0
pan.angle(panAngle)
tilt.angle(tiltAngle)

#endregion init

# region servo control
def limit(x,min,max):
    if x > max:
        return max
    elif x < min:
        return min
```

(continues on next page)

(continued from previous page)

```
    else:
        return x

def servo_control(key):
    global panAngle,tiltAngle
    if key == 'w':
        tiltAngle -= 1
        tiltAngle = limit(tiltAngle, -90, 90)
        tilt.angle(tiltAngle)
    if key == 's':
        tiltAngle += 1
        tiltAngle = limit(tiltAngle, -90, 90)
        tilt.angle(tiltAngle)
    if key == 'a':
        panAngle += 1
        panAngle = limit(panAngle, -90, 90)
        pan.angle(panAngle)
    if key == 'd':
        panAngle -= 1
        panAngle = limit(panAngle, -90, 90)
        pan.angle(panAngle)

# endregion

# continuous shooting
def continuous_shooting(path,interval_ms:int=50,number=10):
    print("continuous_shooting .. ")
    path=path+'/'+strftime("%Y-%m-%d-%H.%M.%S", localtime())
    for i in range(number):
        Vilib.take_photo(photo_name='%03d'%i,path=path)
        print("take_photo: %s"%i)
        sleep(interval_ms/1000)
    print("continuous_shooting done ")

def main():

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

    path = "/home/pi/Pictures/continuous_shooting"

    print(manual)
    while True:
        key = readchar()
        servo_control(key)
        if key == 'q':
            continuous_shooting(path,interval_ms=50,number=10)
        if key == 'g':
            Vilib.camera_close()
            break
        sleep(0.1)


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

How it works?

The code in this article looks slightly complicated, we can split it into three parts:

- Keyboard input
- Servo control
- Take photos

1. First, let's look at the keyboard control part, which includes the following parts:

```
import sys
import tty
import termios

# region  read keyboard
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
# endregion

def main():
    while True:
        key = readchar()
        sleep(0.1)

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

   Its function is to make the terminal can obtain the keyboard input value in real time (without pressing enter), which is more convenient for practical operation.

2. Secondly, let's look at the steering gear control part, which consists of the following code:

```
from time import sleep
from sunfounder_io import PWM,Servo,I2C

### The readchar part is omitted here ###

# region init
I2C().reset_mcu()
sleep(0.01)

pan = Servo(PWM("P1"))
tilt = Servo(PWM("P0"))
panAngle = 0
tiltAngle = 0
pan.angle(panAngle)
tilt.angle(tiltAngle)
#endregion init

# region servo control
def limit(x,min,max):
    if x > max:
        return max
```

(continues on next page)

(continued from previous page)

```
    elif x < min:
        return min
    else:
        return x

def servo_control(key):
    global panAngle,tiltAngle
    if key == 'w':
        tiltAngle -= 1
        tiltAngle = limit(tiltAngle, -90, 90)
        tilt.angle(tiltAngle)
    if key == 's':
        tiltAngle += 1
        tiltAngle = limit(tiltAngle, -90, 90)
        tilt.angle(tiltAngle)
    if key == 'a':
        panAngle += 1
        panAngle = limit(panAngle, -90, 90)
        pan.angle(panAngle)
    if key == 'd':
        panAngle -= 1
        panAngle = limit(panAngle, -90, 90)
        pan.angle(panAngle)

# endregion

def main():
    while True:
        key = readchar()
        servo_control(key)

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

It seems to be a little bit more complicated, but after careful observation, you will find that most of this is the initialization and restriction of the position of the steering gear, which can be perfected according to personal preferences. Its main core is nothing more than the following lines:

```
from time import sleep
from sunfounder_io import PWM,Servo,I2C

I2C().reset_mcu()
sleep(0.01)

pan = Servo(PWM("P1"))
tilt = Servo(PWM("P0"))
panAngle = 0
tiltAngle = 0
pan.angle(panAngle)
tilt.angle(tiltAngle)
```

- Among them, `I2C().reset_mcu()` is used to reset Pan-tilt HAT, which can help you reduce many accidents. It is recommended to add it in every example of using a steering gear.
- And `tilt = Servo(PWM("P0"))` is used to init the servo object. Here, the servo connected to P0 is declared as an object named `tilt` .
- As for `tilt.angle(angle)` , it directly controls the tiltServo, which is the angle of the

servo connected to P0.

3. Finally, let's take a look at the photo section, which is roughly similar to *Take Photo*, but with the addition of continuous shooting.

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

### The readchar part & servo part is omitted here ###

# continuous shooting
def continuous_shooting(path,interval_ms:int=50,number=10):
    print("continuous_shooting .. ")
    path=path+'/'+strftime("%Y-%m-%d-%H.%M.%S", localtime())
    for i in range(number):
        Vilib.take_photo(photo_name='%03d'%i,path=path)
        print("take_photo: %s"%i)
        sleep(interval_ms*0.001)
    print("continuous_shooting done ")

def main():
    Vilib.camera_start(vflip=True,hflip=True)
    Vilib.display(local=True,web=True)

    path = "/home/pi/Pictures/continuous_shooting"

    while True:
        key = readchar()
        #servo_control(key)
        if key == 'q':
            continuous_shooting(path,interval_ms=50,number=10)
        if key == 'g':
            Vilib.camera_close()
            break
        sleep(0.1)

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

We have written a function `continuous_shooting(path,interval_ms=50, number=10)`, whose function is to execute a for loop and execute `Vilib.take_photo()` to achieve continuous shooting.

The photos produced by continuous shooting will be stored in a newly created folder, and the folder will be named according to the current time. Here you may be curious about the time-related functions `strftime()` and `localtime()`, then please see Time-Python Docs.

2.5 Time Lapse Photography

Some things happen too slowly for us to perceive, such as the legend of the flow of people, sunrise and sunset, and the blooming of flower buds. Time-lapse photography allows you to see these exciting things clearly.

You will use two windows at the same time in this project: One is Terminal, where you will enter `wasd` to control the camera orientation, enter `q` to record, then enter `e` to stop, and enter `g` to exit shooting. If the program has not been terminated after exiting the shooting, please press `ctrl+c`. Another browser interface, after the program runs, you will need to enter `http://<Your Raspberry Pi IP>:9000/mjpg` in the PC browser (such as chrome) to view the screen.

Run the Code

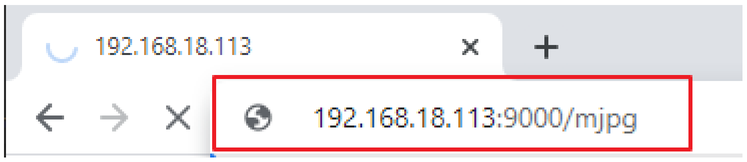
```
cd /home/pi/pan-tilt-hat/examples
sudo python3 time_lapse_photography
```

View the Image

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

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

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



Code

```python
from time import perf_counter, sleep,strftime,localtime
from vilib import Vilib
from sunfounder_io import PWM,Servo,I2C
import cv2
import os
import sys
import tty
import termios
import threading

# region  read keyboard
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 record value!
    W: up
    A: left
    S: right
    D: down
    Q: start Time-lapse photography
    E: stop
    G: Quit
'''
```

(continues on next page)

(continued from previous page)

```
# endregion

# region init
I2C().reset_mcu()
sleep(0.01)

pan = Servo(PWM("P1"))
tilt = Servo(PWM("P0"))
panAngle = 0
tiltAngle = 0
pan.angle(panAngle)
tilt.angle(tiltAngle)
# endregion

# # check dir
def check_dir(dir):
    if not os.path.exists(dir):
        try:
            os.makedirs(dir)
        except Exception as e:
            print(e)

# region servo control
def limit(x,min,max):
    if x > max:
        return max
    elif x < min:
        return min
    else:
        return x

def servo_control(key):
    global panAngle,tiltAngle
    if key == 'w':
        tiltAngle -= 1
        tiltAngle = limit(tiltAngle, -90, 90)
        tilt.angle(tiltAngle)
    if key == 's':
        tiltAngle += 1
        tiltAngle = limit(tiltAngle, -90, 90)
        tilt.angle(tiltAngle)
    if key == 'a':
        panAngle += 1
        panAngle = limit(panAngle, -90, 90)
        pan.angle(panAngle)
    if key == 'd':
        panAngle -= 1
        panAngle = limit(panAngle, -90, 90)
        pan.angle(panAngle)

# endregion servo control

# Video synthesis
def video_synthesis(name:str,input:str,output:str,fps=30,format='.jpg',
↪datetime=False):

    print('processing video, please wait ....')
```

(continues on next page)

(continued from previous page)

```
    # video parameter
    fourcc = cv2.VideoWriter_fourcc(*'XVID')
    out = cv2.VideoWriter(output+'/'+name, fourcc, fps, (640,480))
    width = 640
    height = 480

    # traverse

    for root, dirs, files in os.walk(input):
        print('%s pictures be processed'%len(files))
        files = sorted(files)
        for file in files:
            # print('Format:',os.path.splitext(file)[1])
            if os.path.splitext(file)[1] == format:
                # imread
                frame = cv2.imread(input+'/'+file)
                # add datetime watermark
                if datetime == True:
                    # print('name:',os.path.splitext(file)[1])
                    time = os.path.splitext(file)[0].split('-')
                    year = time[0]
                    month = time[1]
                    day = time[2]
                    hour = time[3]
                    minute = time[4]
                    second = time[5]
                    frame = cv2.putText(frame,'%s.%s.%s %s:%s:%s'%(year,month,day,
↪hour,minute,second),
                                        (width - 180, height - 25),cv2.FONT_HERSHEY_
↪SIMPLEX, 0.5,
                                        (255, 255, 255),1,cv2.LINE_AA)    # anti-
↪aliasing
                # write video
                out.write(frame)

    # release the VideoWriter object
    out.release()

    # remove photos
    os.system('sudo rm -r %s'%input)
    print('Done.The video save as %s/%s'%(output,name))

# keyboard scan thread
key = None
breakout_flag=False
def keyboard_scan():
    global key
    while True:
        key = None
        key = readchar()
        sleep(0.01)
        if breakout_flag==True:
            break

# continuous_shooting
def continuous_shooting(path,interval_ms:int=3000):
```

(continues on next page)

(continued from previous page)

```
    print('Start time-lapse photography, press the "e" key to stop')

    delay = 10 # ms

    count = 0
    while True:
        if count == interval_ms/delay:
            count = 0
            Vilib.take_photo(photo_name=strftime("%Y-%m-%d-%H-%M-%S", localtime()),
↪path=path)
        if key == 'e':
            break
        count += 1
        sleep(delay/1000)  # second

# main
def main():

    print(manual)

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

    sleep(1)
    t = threading.Thread(target=keyboard_scan)
    t.setDaemon(True)
    t.start()


    while True:
        servo_control(key)

        # time-lapse photography
        if key == 'q':

            #check path
            output  = "/home/pi/Pictures/time_lapse" # -o
            input = output+'/'+strftime("%Y-%m-%d-%H-%M-%S", localtime())
            check_dir(input)
            check_dir(output)

            # take_photo
            continuous_shooting(input,interval_ms=3000)

            # video_synthesis
            name=strftime("%Y-%m-%d-%H-%M-%S", localtime())+'.avi'
            video_synthesis(name=name,input=input,output=output,fps=30,format='.jpg',
↪datetime=True)

        # esc
        if key == 'g':
            Vilib.camera_close()
            global breakout_flag
            breakout_flag=True
            sleep(0.1)
            print('The program ends, please press CTRL+C to exit.')
            break
```

(continues on next page)

(continued from previous page)

```
        sleep(0.01)

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

How it works?

Similar to *Continuous Shooting*, this example also needs to be split for analysis. It includes the following parts:

- Servo control
- Key input
- Path management
- Shooting
- Video synthesis

1. **Servo Control**: It is exactly the same as Continuous Shooting, no need to repeat it.
2. **Key input**: Its implementation is consistent with Continuous Shooting (ie `readchar()`), but it is called by a separate thread. We extract the relevant code separately, as follows:

```
'''
Time-lapse photography based on the Raspistill command
'''
from time import sleep,

import sys
import tty
import termios
import threading

# region  read keyboard
def readchar():
    pass

# keyboard scan thread
key = None
breakout_flag=False

def keyboard_scan():
    global key
    while True:
        key = None
        key = readchar()
        sleep(0.01)
        if breakout_flag==True:
            break

# main
def main():

    t = threading.Thread(target=keyboard_scan)
    t.setDaemon(True)
    t.start()

    while True:
```

(continues on next page)

(continued from previous page)

```
        # esc
        if key == 'g':
            global breakout_flag
            breakout_flag=True
            sleep(0.1)
            print('The program ends, please press CTRL+C to exit.')
            break
        sleep(0.01)

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

Simply put, the `t = threading.Thread(target=keyboard_scan)` line of the main function generates a thread and calls the `keyboard_scan()` function. This function calls `readchar()` in a loop until the `breakout_flag` ends after being modified.

For details on the use of threads, please refer to Threading - Python Docs.

3. **Route Management**: Used to ensure that the file read and write path during shooting is correct. It includes the following:

```
import os

# # check dir
def check_dir(dir):
    if not os.path.exists(dir):
        try:
            os.makedirs(dir)
        except Exception as e:
            print(e)

# main
def main():

    while True:
        if key == 'q':
            #check path
            output = "/home/pi/Pictures/time_lapse" # -o
            input = output+'/'+strftime("%Y-%m-%d-%H-%M-%S", localtime())
            check_dir(input)
            check_dir(output)

            # take_photo
            # video_synthesis


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

The target directory for our output videos is `output`. And generating video requires a large number of temporary still photos, which are stored in `input`. The function of `check_dir()` is to check whether the target folder exists, and create it if it does not exist.

An `os` library is imported here, which allows python to use related functions of the operating system. Such as reading and writing files, creating files and directories, and manipulate paths. For details, please see OS - Python Docs.

4. **Shooting**: Similar to *Continuous Shooting*, the difference is that instead of writing a specific number of photos,

you manually press `e` to stop. This is achieved because the keyboard input is separated from the main program and runs on the thread separately.

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


# continuous_shooting
def continuous_shooting(path,interval_ms:int=3000):
    print('Start time-lapse photography, press the "e" key to stop')

    delay = 10 # ms
    count = 0
    while True:
        if count == interval_ms/delay:
            count = 0
            Vilib.take_photo(photo_name=strftime("%Y-%m-%d-%H-%M-%S",␣
↪localtime()),path=path)
        if key == 'e':
            break
        count += 1
        sleep(delay/1000) # second

# main
def main():

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

    while True:
        # time-lapse photography
        if key == 'q':
            #check path

            # take_photo
            continuous_shooting(input,interval_ms=3000)

            # video_synthesis

        # esc
        if key == 'g':
            Vilib.camera_close()
            break
        sleep(0.01)

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

5. **Video synthesis**: It uses the photos stored in the `input` path as frames, and generates a video output to the `output` path.

```
from time import perf_counter, sleep,strftime,localtime
from vilib import Vilib
import cv2
import os

# Video synthesis
def video_synthesis(name:str,input:str,output:str,fps=30,format='.jpg',
↪datetime=False):
```

(continues on next page)

(continued from previous page)

```
    print('processing video, please wait ....')

    # video parameter
    fourcc = cv2.VideoWriter_fourcc(*'XVID')
    out = cv2.VideoWriter(output+'/'+name, fourcc, fps, (640,480))
    width = 640
    height = 480

    # traverse

    for root, dirs, files in os.walk(input):
        print('%s pictures be processed'%len(files))
        file = sorted(files)
        for file in files:
            # print('Format:',os.path.splitext(file)[1])
            if os.path.splitext(file)[1] == format:
                # imread
                frame = cv2.imread(input+'/'+file)
                # add datetime watermark
                if datetime == True:
                    # print('name:',os.path.splitext(file)[1])
                    time = os.path.splitext(file)[0].split('-')
                    year = time[0]
                    month = time[1]
                    day = time[2]
                    hour = time[3]
                    minute = time[4]
                    second = time[5]
                    frame = cv2.putText(frame, '%s.%s.%s %s:%s:%s'%(year,
↪month,day,hour,minute,second),
                                        (width - 180, height - 25), cv2.
↪FONT_HERSHEY_SIMPLEX, 0.5,
                                        (255, 255, 255),1,cv2.LINE_AA)
↪# anti-aliasing
                # write video
                out.write(frame)

    # release the VideoWriter object
    out.release()

    # remove photos
    os.system('sudo rm -r %s'%input)
    print('Done.The video save as %s/%s'%(output,name))

# main
def main()
    while True:
        if key == 'q':
            #check path
            # take_photo

            # video_synthesis
            name=strftime("%Y-%m-%d-%H-%M-%S", localtime())+'.avi'
            video_synthesis(name=name,input=input,output=output,fps=30,
↪format='.jpg',datetime=True)
```

(continues on next page)

(continued from previous page)

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

Here, the video writer object is initialized first. The code show as below:

```
fourcc = cv2.VideoWriter_fourcc(*'XVID')
out = cv2.VideoWriter(output+'/'+name, fourcc, fps, (640,480))
```

This module is derived from OpenCV, please refer to VideoWriter-OpenCV Docs for details.

Then, loop through each frame to form a video:

```
for root, dirs, files in os.walk(input):
    print('%s pictures be processed'%len(files))
    files = sorted(files)
    for file in files:
        if os.path.splitext(file)[1] == format:
            # imread
            frame = cv2.imread(input+'/'+file)
            # add datetime watermark
            if datetime == True:
                time = os.path.splitext(file)[0].split('-')
                year = time[0]
                month = time[1]
                day = time[2]
                hour = time[3]
                minute = time[4]
                second = time[5]
                frame = cv2.putText(frame, '%s.%s.%s %s:%s:%s'%(year,
↪month,day,hour,minute,second),
                                    (width - 180, height - 25), cv2.FONT_
↪HERSHEY_SIMPLEX, 0.5,
                                    (255, 255, 255),1,cv2.LINE_AA)    #␣
↪anti-aliasing
            # write video
            out.write(frame)
```

After the video is processed, release the VideoWriter.

```
# release the VideoWriter object
out.release()
```

Finally delete the input folder. Of course, if you have enough space, comment out this line of code to keep the original picture.

```
# remove photos
os.system('sudo rm -r %s'%input)
```

2.6 Record Video

This example allows us to record a video.

Here you will use two windows at the same time: One is Terminal, you can type `wasd` here to control the orientation of the camera, type `q` to record/pause/continue recording, type `e` to stop recording, and type `g` to exit shooting . If the program has not been terminated after exiting the shooting, please type `ctrl+c`. Another browser interface, after the program runs, you will need to enter `http://<Your Raspberry Pi IP>:9000/mjpg` in the PC browser (such as chrome) to view the viewfinder screen.

Run the Code

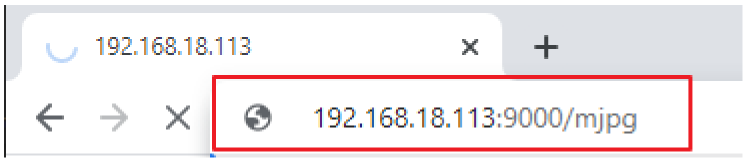
```
cd /home/pi/pan-tilt-hat/examples
sudo python3 record_video.py
```

View the Image

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

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

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



Code

```python
from time import sleep,strftime,localtime
from vilib import Vilib
from sunfounder_io import PWM,Servo,I2C

import sys
import tty
import termios

# region  read keyboard
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 = '''
```

(continues on next page)

(continued from previous page)

```
Press keys on keyboard to record value!
    W: up
    A: left
    S: right
    D: down
    Q: record/pause/continue
    E: stop

    G: Quit
'''
# endregion

# region init
I2C().reset_mcu()
sleep(0.01)

pan = Servo(PWM("P1"))
tilt = Servo(PWM("P0"))
panAngle = 0
tiltAngle = 0
pan.angle(0)
tilt.angle(0)

Vilib.rec_video_set["path"] = "/home/pi/Pictures/"
vname = strftime("%Y-%m-%d-%H.%M.%S", localtime())
Vilib.rec_video_set["name"] = vname

rec_flag = 'stop' # start,pause,stop
# endregion init

# rec control
def rec_control(key):
    global rec_flag

    if key == 'q' and rec_flag == 'stop':
        key = None
        rec_flag = 'start'
        Vilib.rec_video_run()
        print('rec start ...')
    if key == 'q' and rec_flag == 'start':
        key = None
        rec_flag = 'pause'
        Vilib.rec_video_pause()
        print('pause')
    if key == 'q' and rec_flag == 'pause':
        key = None
        rec_flag = 'start'
        Vilib.rec_video_start()
        print('continue')

    if key == 'e' and rec_flag != 'stop':
        Vilib.rec_video_stop()
        print('stop')
        print("The video saved as %s%s.avi"%(Vilib.rec_video_set["path"],vname))

# region servo control
def limit(x,min,max):
```

(continues on next page)

(continued from previous page)

```
    if x > max:
        return max
    elif x < min:
        return min
    else:
        return x

def servo_control(key):
    global panAngle,tiltAngle
    if key == 'w':
        tiltAngle -= 1
        tiltAngle = limit(tiltAngle, -90, 90)
        tilt.angle(tiltAngle)
    if key == 's':
        tiltAngle += 1
        tiltAngle = limit(tiltAngle, -90, 90)
        tilt.angle(tiltAngle)
    if key == 'a':
        panAngle += 1
        panAngle = limit(panAngle, -90, 90)
        pan.angle(panAngle)
    if key == 'd':
        panAngle -= 1
        panAngle = limit(panAngle, -90, 90)
        pan.angle(panAngle)

# endregion servo control


def main():

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

    print(manual)
    while True:
        key = readchar()
        # rec control
        rec_control(key)
        # servo control
        servo_control(key)
        # esc
        if key == 'g':
            Vilib.camera_close()
            break

        sleep(0.1)

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

How it works?

This article can be divided into three parts to analyze:

- Keyboard input
- Servo control

- Record video

The first two parts are consistent with *Continuous Shooting*. The record video function code is as follows:

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

# region init
Vilib.rec_video_set["path"] = "/home/pi/Pictures/"
vname = strftime("%Y-%m-%d-%H.%M.%S", localtime())
Vilib.rec_video_set["name"] = vname

rec_flag = 'stop' # start,pause,stop
# endregion init

# rec control
def rec_control(key):
    global rec_flag

    if key == 'q' and rec_flag == 'stop':
        key = None
        rec_flag = 'start'
        Vilib.rec_video_run()
        print('rec start ...')
    if key == 'q' and rec_flag == 'start':
        key = None
        rec_flag = 'pause'
        Vilib.rec_video_pause()
        print('pause')
    if key == 'q' and rec_flag == 'pause':
        key = None
        rec_flag = 'start'
        Vilib.rec_video_start()
        print('continue')

    if key == 'e' and rec_flag != 'stop':
        Vilib.rec_video_stop()
        print('stop')
        print("The video saved as %s%s.avi"%(Vilib.rec_video_set["path"],vname))


def main():

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

    while True:
        rec_control(key)
        if key == 'g':
            Vilib.camera_close()
            break

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

Parameters related to recording include the following:

- `Vilib.rec_video_set["path"]` The address where the video is saved
- `Vilib.rec_video_set["name"]` The name of the saved video

Functions related to recording include the following:

- `Vilib.rec_video_run()` Start recording
- `Vilib.rec_video_pause()` Pause recording
- `Vilib.rec_video_start()` Continue recording
- `Vilib.rec_video_stop()` Stop recording

2.7 Panorama

A panorama is a wide-angle view or representation of the physical space, which is composed of multiple standard photos stitched together to bring a visual wonder with a field of view far beyond the human eye.

In this article, you will use the slow rotation of the pan-tilt to obtain multiple photos to combine a long panorama picture.

Run the Code

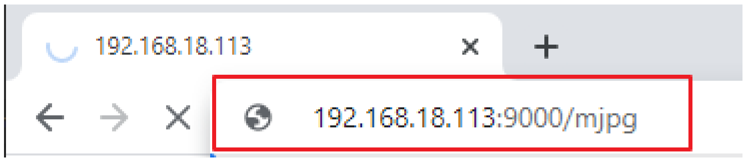
```
cd /home/pi/pan-tilt-hat/examples
sudo python3 panorama.py
```

View the Image

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

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

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



Code

```
from time import perf_counter, sleep,strftime,localtime
from vilib import Vilib
from sunfounder_io import PWM,Servo,I2C
import cv2
import os

import sys
import tty
import termios

# region  read keyboard
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 record value!
    Q: take photo
    G: Quit
'''
# endregion

# # check dir
def check_dir(dir):
    if not os.path.exists(dir):
        try:
            os.makedirs(dir)
        except Exception as e:
            print(e)

# region init
I2C().reset_mcu()
sleep(0.01)

pan = Servo(PWM("P1"))
tilt = Servo(PWM("P0"))
panAngle = 0
tiltAngle = 0
pan.angle(panAngle)
tilt.angle(tiltAngle )
# endregion

Status_info = {
    0: 'OK',
    1: 'ERR_NEED_MORE_IMGS',
    2: 'ERR_HOMOGRAPHY_EST_FAIL',
    3: 'ERR_CAMERA_PARAMS_ADJUST_FAIL',
}

def panorama_shooting(path):
    global panAngle,tiltAngle
```

(continues on next page)

(continued from previous page)

```
    temp_path = "/home/pi/picture/.temp/panorama"
    imgs =[]

    # check path
    check_dir(path)

    # take photo
    for a in range(panAngle,-81,-5):
        panAngle = a
        pan.angle(panAngle)
        sleep(0.1)

    num = 0
    for angle in range(-80,81,20):
        for a in range(panAngle,angle,1):
            panAngle = a
            pan.angle(a)
            sleep(0.1)
        sleep(0.5)
        # sleep(0.5)
        print(num,angle)
        Vilib.take_photo(photo_name='%s'%num,path=temp_path)
        sleep(0.2)
        num += 1

    # stitch image
    stitcher = cv2.Stitcher_create(cv2.Stitcher_SCANS)

    for index in range(num):
        imgs.append(cv2.imread('%s/%s.jpg'%(temp_path,index)))
    print('imgs num: %s'%len(imgs))

    status,pano = stitcher.stitch(imgs)

    # imwrite and imshow
    print('status: %s , %s'%(status,Status_info[status]))
    if status == 0:
        cv2.imwrite('%s/%s.jpg'%(path,strftime("%Y-%m-%d-%H.%M.%S", localtime())),
↪pano)
        cv2.imshow('panorama',pano)

    os.system('sudo rm -r %s'%temp_path)

# main

def main():

    print(manual)

    Vilib.camera_start(vflip=True,hflip=True)
    Vilib.display(local=True,web=True)
    sleep(0.1)

    path = "/home/pi/Pictures/panorama"
    while True:
        key = readchar()
```

(continues on next page)

(continued from previous page)

```
        # take photo
        if key == 'q':
            print("panorama shooting ...")
            panorama_shooting(path)

        # esc
        if key == 'g':
            print('Quit')
            Vilib.camera_close()
            break

        sleep(0.01)

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

How it works?

The core functions of this example are placed in the `panorama_shooting(path)` function. This function includes the following parts:

1. Path management: that is, `checkdir()` , I won't repeat it.
2. Photograph:

   ```
   # take photo
   for a in range(panAngle,-81,-5):
       panAngle = a
       pan.angle(panAngle)
       sleep(0.1)

   num = 0
   for angle in range(-80,81,20):
       for a in range(panAngle,angle,1):
           panAngle = a
           pan.angle(a)
           sleep(0.1)
       sleep(0.5)

       print(num,angle)
       Vilib.take_photo(photo_name='%s'%num,path=temp_path)
       sleep(0.2)
       num += 1
   ```

   Here, the two functions of taking pictures and steering gear control are put together. The pan servo starts to deflection slowly counterclockwise from the -80° position, and takes a picture every 20° deflection, until the 80° position ends. After execution, you will get 9 temporary photos, which are stored in the path `temp_path`.

3. Stitching photos:

   ```
   # stitch image
   stitcher = cv2.Stitcher_create(cv2.Stitcher_SCANS)

   for index in range(num):
       imgs.append(cv2.imread('%s/%s.jpg'%(temp_path,index)))
   print('imgs num: %s'%len(imgs))
   ```

(continues on next page)

(continued from previous page)

```
status,pano = stitcher.stitch(imgs)

# imwrite and imshow
print('status: %s , %s'%(status,Status_info[status]))
if status == 0:
    cv2.imwrite('%s/%s.jpg'%(path,strftime("%Y-%m-%d-%H.%M.%S",␣
↪localtime())),pano)
    cv2.imshow('panorama',pano)

os.system('sudo rm -r %s'%temp_path)
```

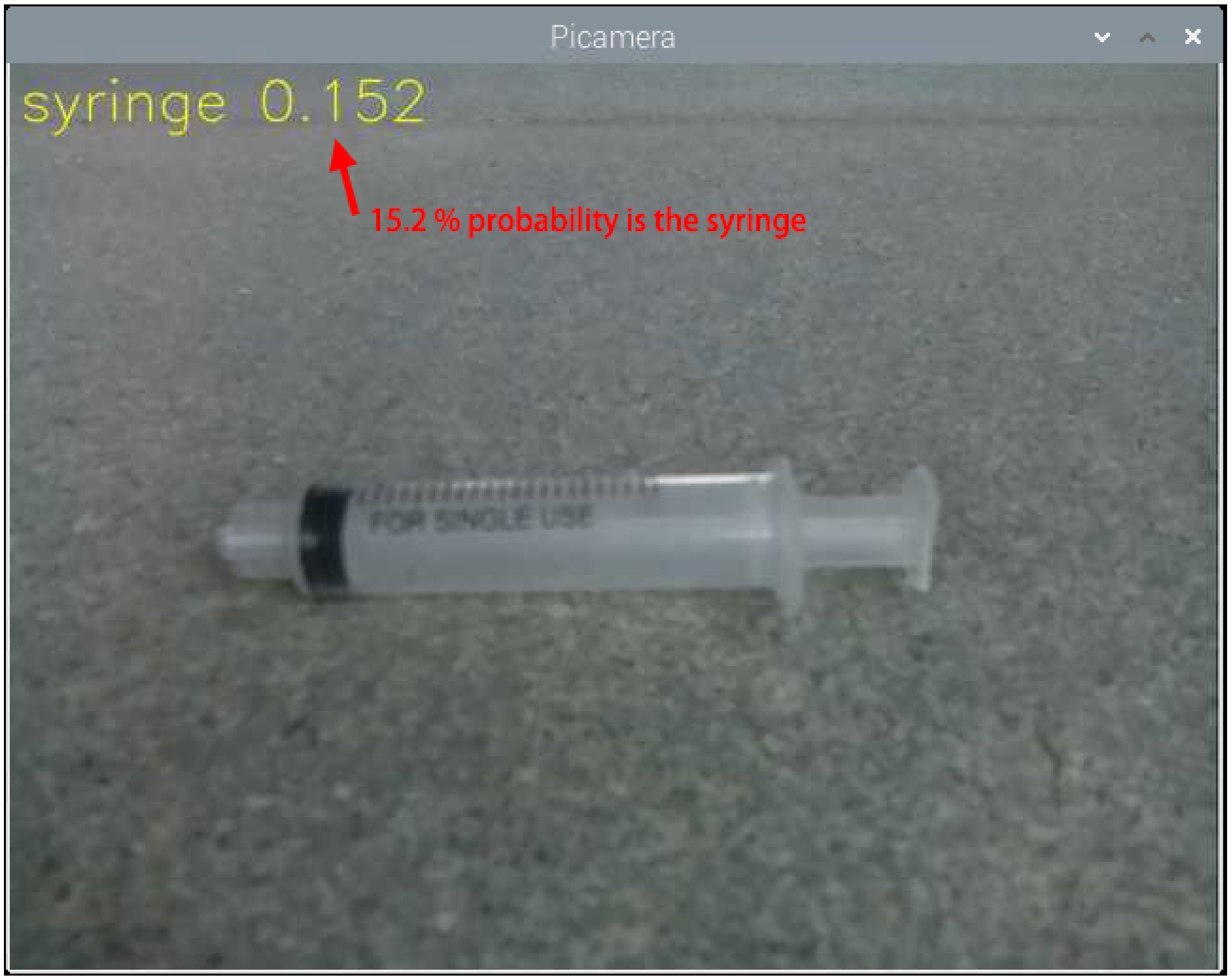
These photos are added to an array `imgs`, and then call OpenCV's Stitcher module (ie `status, pano = stitcher.stitch(imgs)`) to merge them into a panorama `pano`. Finally, use `cv2.imwrite()` to write `pano` into the storage space, and delete the temporary photos and their paths.

For more details, please see Stitcher-Docs and Image file reading and writing-OpenCV.

2.8 Image Classification

When you see a rabbit, you will directly say its name. The computer will tell you that 91% of it may be a rabbit, 4% of it may be a cat, 3% of it may be a dog, and 2% of it may be something else.

We use image classification to reveal the cognitive model of computer vision. Image classification is a basic task. A trained model is used to identify images representing various objects, and then the images are assigned to specific tags to classify the images.



Run the Code

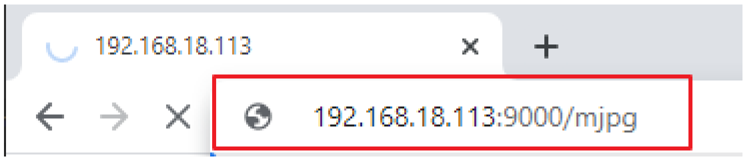
```
cd /home/pi/pan-tilt-hat/examples
sudo python3 image_classification.py
```

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 vilib import Vilib

def main():
    Vilib.camera_start(vflip=True,hflip=True)
    Vilib.display(local=True,web=True)
    Vilib.image_classify_set_model(path='/home/pi/pan-tilt-hat/models/mobilenet_v1_0.
↪25_224_quant.tflite')
    Vilib.image_classify_set_labels(path='/home/pi/pan-tilt-hat/models/labels_
↪mobilenet_quant_v1_224.txt')
    Vilib.image_classify_switch(True)

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

How it works?

The use of this feature is very simple. You only need to pay attention to the following three lines of code:

- `Vilib.image_classify_set_model(path)` : Load the trained model file.
- `Vilib.image_classify_set_labels(path)` : Load the corresponding label file.
- `Vilib.image_classify_switch(True)` : Start the image classifier.

Here, we directly use Tensorflow pre-trained model, which is an image classification model that includes thousands of objects. You can open the label file (`/home/pi/pan-tilt-hat/models/labels_mobilenet_quant_v1_224.txt`) to see which objects are included.

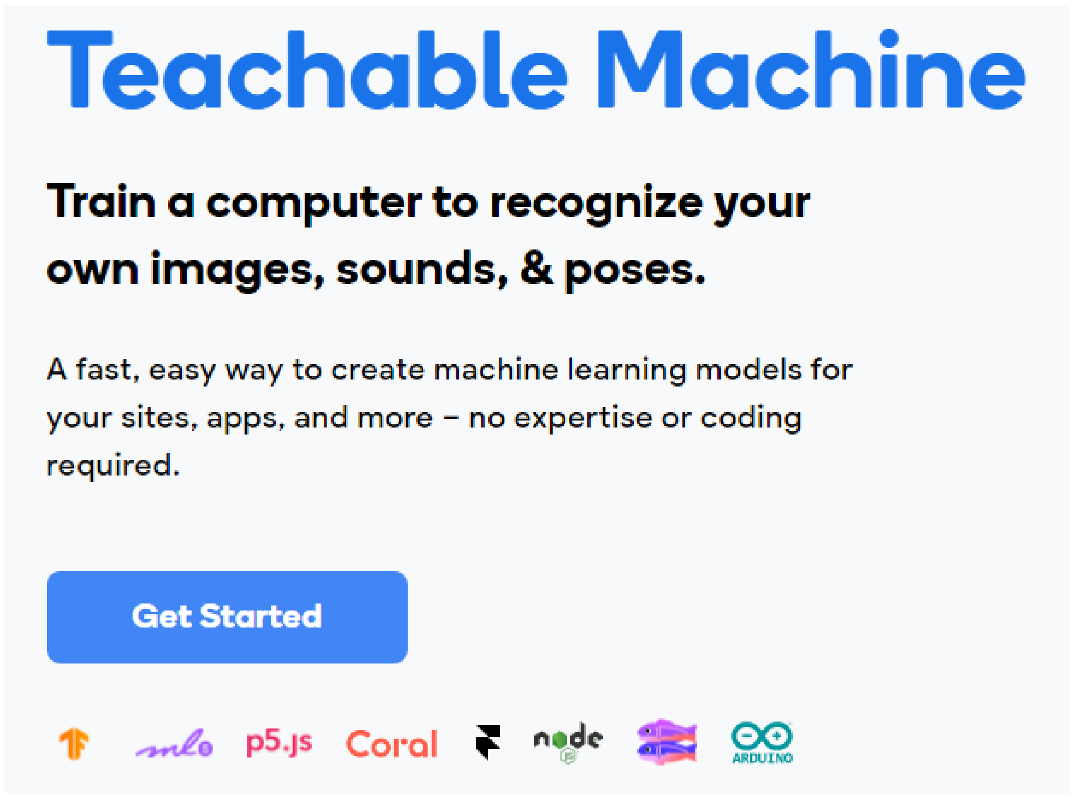
In addition to the built-in models in this article, you can also download the pre-trained image classification model on TensorFlow Hub. It should be noted that these models may not be suitable for your project, please use them as appropriate.

If you want to try to create your own model. We strongly recommend that you use Teachable Machine. It is a web-based tool that allows everyone to create machine learning models quickly, easily, and accessible. Please click Get start on the webpage to start training your model.

Note: Raspberry Pi may not be able to use Teachable Machine smoothly. You will need to prepare a PC or laptop equipped with a camera.

Model Training

1. Open Teachable Machine, you will see an obvious Get Start on the web page, click on it.


Teachable Machine

Train a computer to recognize your own images, sounds, & poses.

A fast, easy way to create machine learning models for your sites, apps, and more – no expertise or coding required.

2. Select Image Project (Audio Project and Pose Project are not applicable here). You will be prompted to choose Standard image model or Embedded image model. The former has a higher accuracy rate and the latter has a faster speed. We recommend choosing the first one.

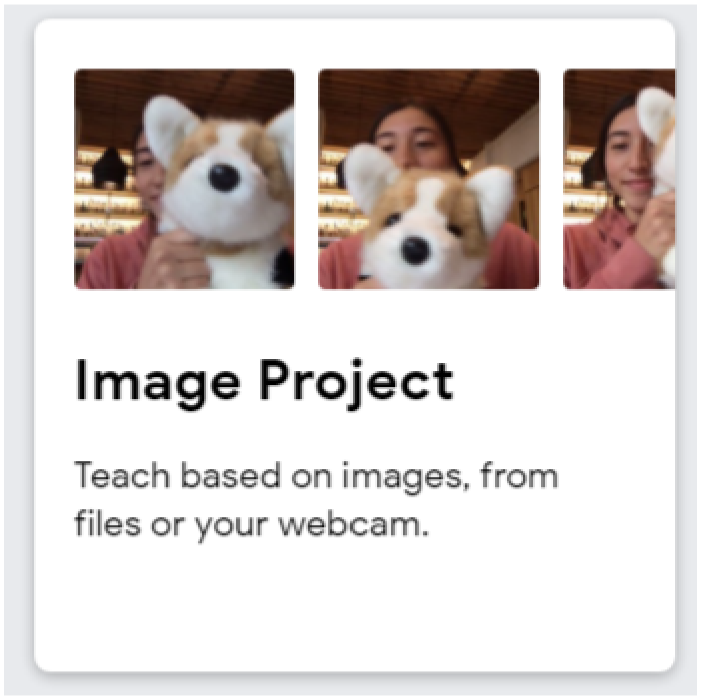
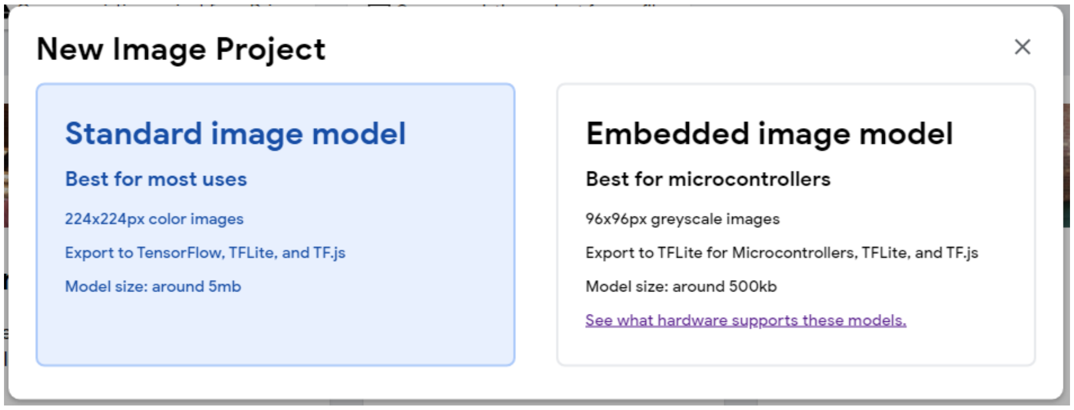


Image Project

Teach based on images, from files or your webcam.



New Image Project

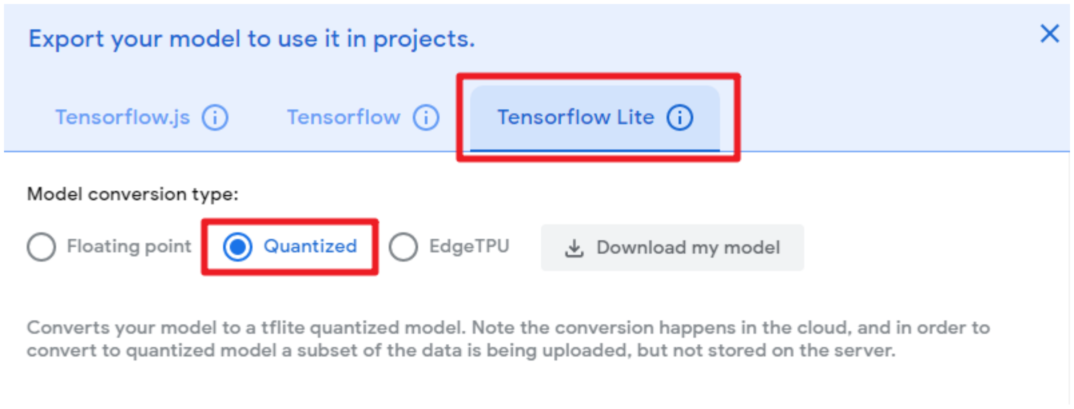
Standard image model
Best for most uses
224x224px color images
Export to TensorFlow, TFLite, and TF.js
Model size: around 5mb

Embedded image model
Best for microcontrollers
96x96px greyscale images
Export to TFLite for Microcontrollers, TFLite, and TF.js
Model size: around 500kb
See what hardware supports these models.

3. Train the model. Teachable Machine provides a detailed video step-by-step explanation, please see:

Note: The video after 0:55 is the content of the other two projects and is not applicable here.

Note: The export settings applicable to this project are shown in the figure:



Export your model to use it in projects.

Tensorflow.js | Tensorflow | Tensorflow Lite

Model conversion type:

Floating point [✓ Quantized] EdgeTPU

Download my model

Converts your model to a tflite quantized model. Note the conversion happens in the cloud, and in order to convert to quantized model a subset of the data is being uploaded, but not stored on the server.

4. Unzip the downloaded zip file, you will be able to see the model file and label file, their formats are `.tflite` and `.txt` respectively. Use *Filezilla Software* to copy them to the `/home/pi/pan-tilt-hat/models/` directory of the Raspberry Pi.
5. Modify the two lines of the sample code in this article, and change them to your model and label.

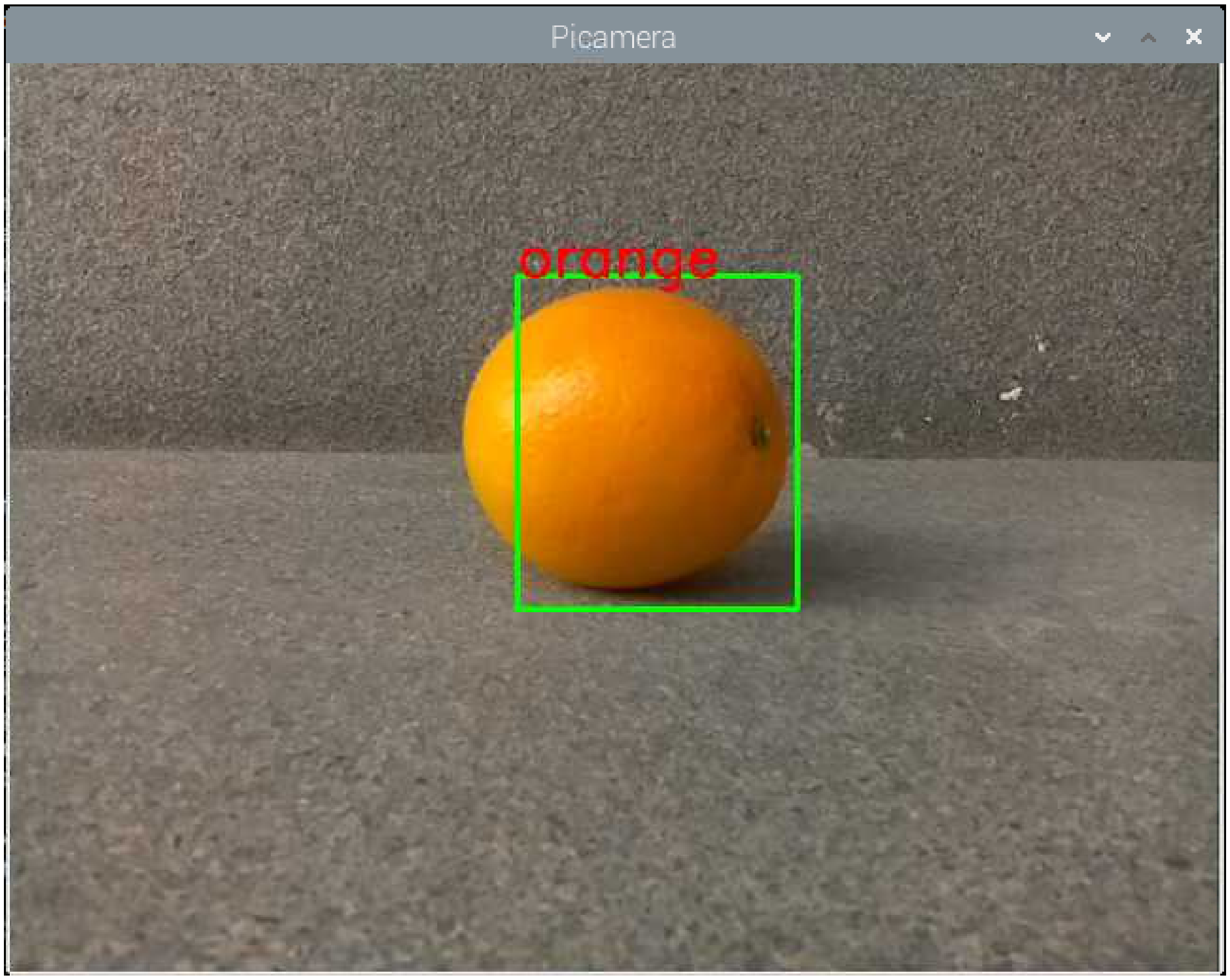
```
Vilib.image_classify_set_model(path='/home/pi/pan-tilt-hat/models/your_
↪model.tflite')
Vilib.image_classify_set_labels(path='/home/pi/pan-tilt-hat/models/your_
↪label.txt')
```

6. Re-run the example. It will recognize the objects in your training model.

2.9 Color Detection

Say a color and mark it in the field of view. This is not difficult for most humans, because we have been trained in this way since we were young.

For computers, thanks to deep learning, such tasks can also be accomplished. In this project, there is an algorithm that can find a certain color (6 kinds in total), such as finding "orange".



Run the Code

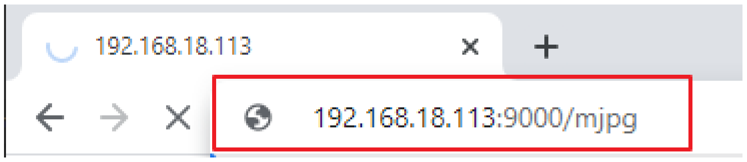
```
cd /home/pi/pan-tilt-hat/examples
sudo python3 color_detection.py
```

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

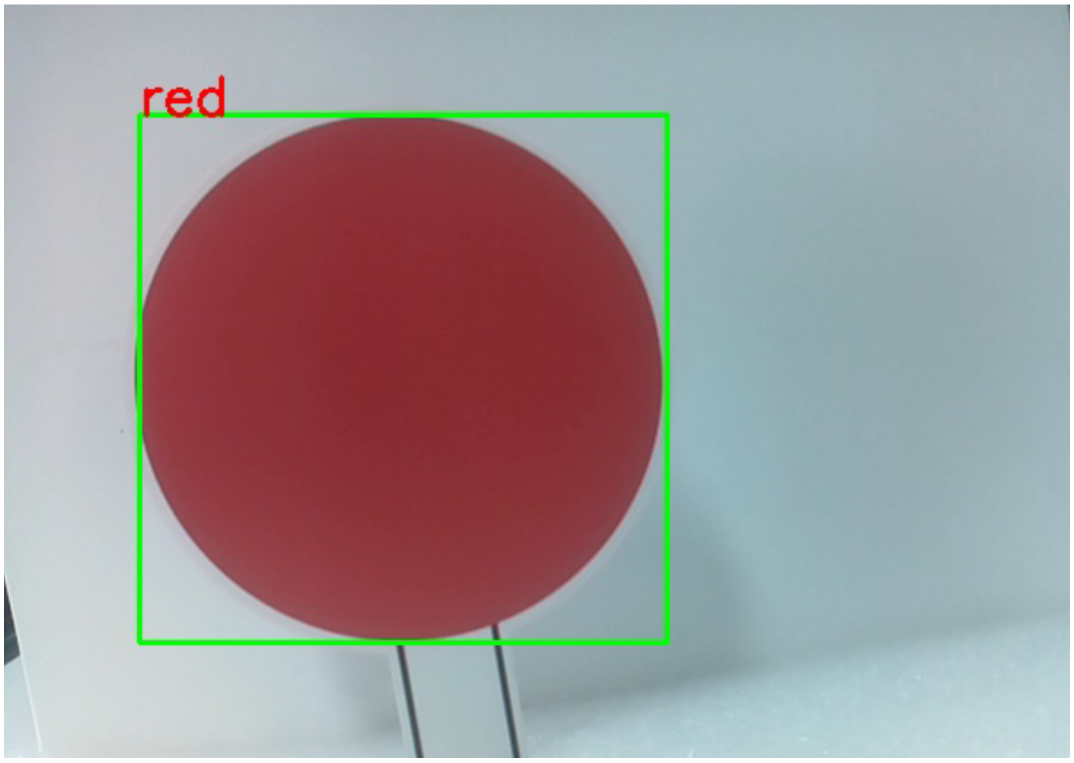
After the program runs, you will see the following information in the final:

```
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
s: Display detected object information
```

Please follow the prompts to activate the corresponding functions.

- **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.

- **Display Information**

  Entering `s` will print the information of the 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

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
    s: Display detected object information
'''

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

def show_info():
    if flag_color is True and Vilib.detect_obj_parameter['color_n']!=0:
        color_coodinate = (Vilib.detect_obj_parameter['color_x'],Vilib.detect_obj_
↪parameter['color_y'])
        color_size = (Vilib.detect_obj_parameter['color_w'],Vilib.detect_obj_
↪parameter['color_h'])
        print("Coordinate:",color_coodinate,"Size",color_size)

def main():
    Vilib.camera_start(vflip=True,hflip=True)
    Vilib.display(local=True,web=True)
    print(manual)

    global flag_color

    while True:
        key = input()
        if key == "1":
            color_detect("red")
            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")
```

(continues on next page)

(continued from previous page)

```
            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 == "s":
            show_info()

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(vflip=True,hflip=True)
Vilib.display(local=True,web=True)
```

Functions related to "color 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

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:

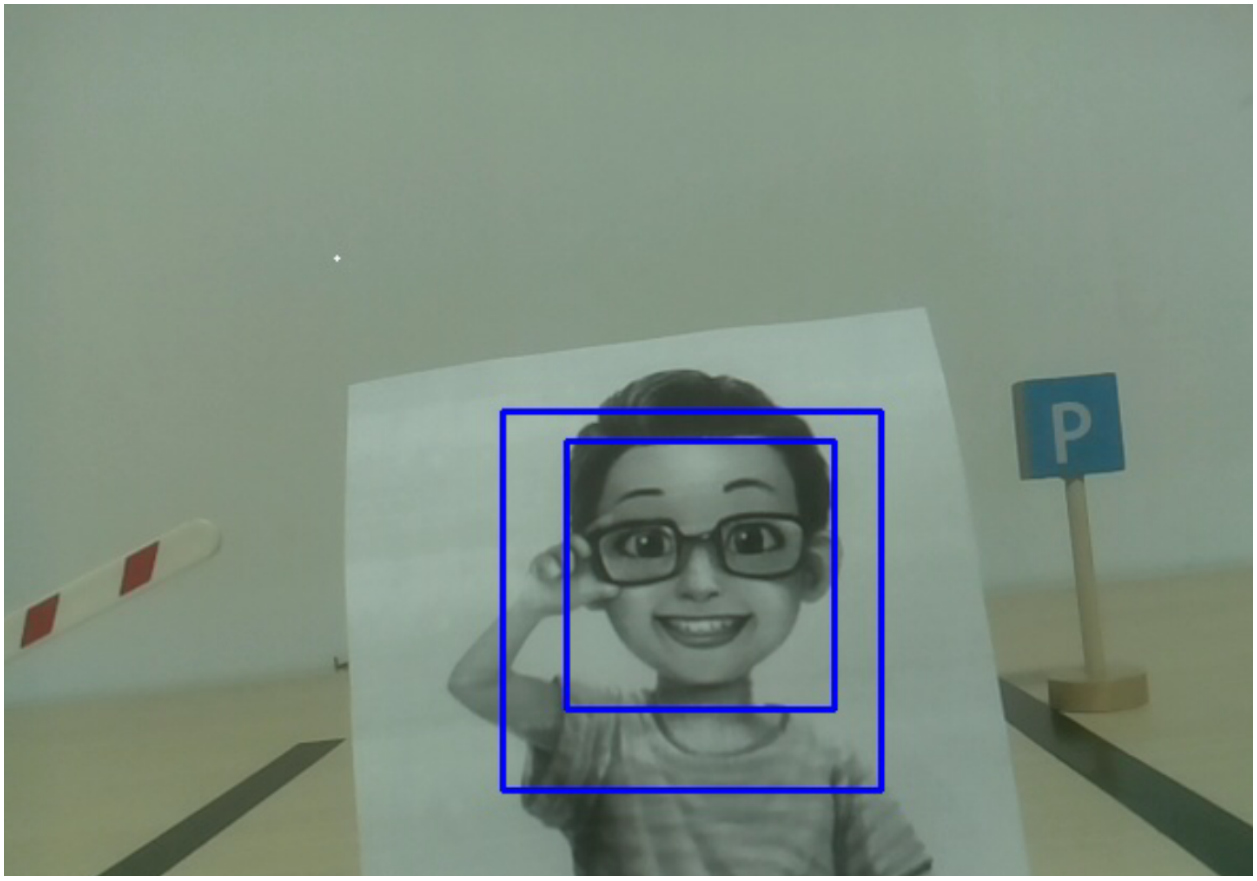
```
Vilib.detect_obj_parameter['color_x']
```

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

2.10 Face Detection

Face detection is the first step in building a face recognition system. It will uniformly calibrate the face according to the key points in the face (such as the position of the eyes, the position of the nose, the position of the mouth, the contour points of the face, etc.) and mark the area containing the face. This technology is widely used in security and other scenarios.



Run the Code

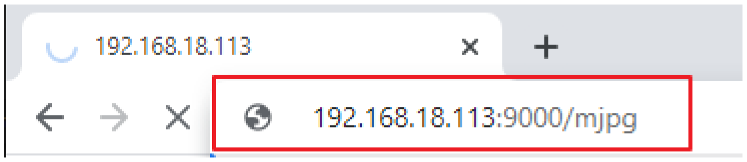
```
cd /home/pi/pan-tilt-hat/examples
sudo python3 face_detection.py
```

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 vilib import Vilib
from time import sleep


def object_show():
    print("Number: ",Vilib.detect_obj_parameter['human_n'])
    human_coodinate = (Vilib.detect_obj_parameter['human_x'],Vilib.detect_obj_
↪parameter['human_y'])
    human_size = (Vilib.detect_obj_parameter['human_w'],Vilib.detect_obj_parameter[
↪'human_h'])
    print("Coordinate:",human_coodinate,"Size",human_size)

def main():
    Vilib.camera_start(vflip=True,hflip=True)
    Vilib.display(local=True,web=True)
    Vilib.human_detect_switch(True)

    while True:
        object_show()
        sleep(0.2)


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

How it works?

Functions related to human face detection:

- `Vilib.human_detect_switch(True)` : Switch ON/OFF face detection

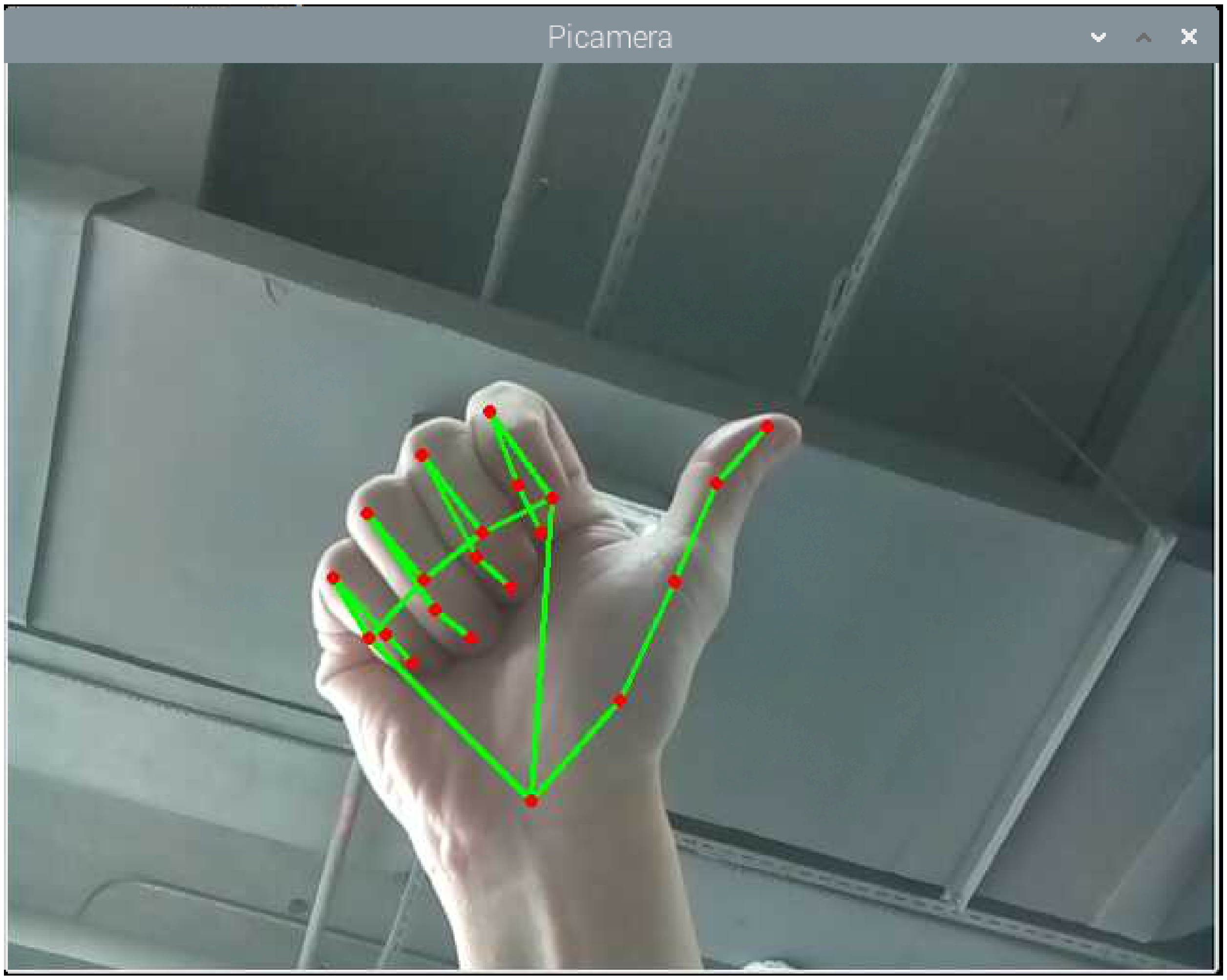
The keys of the dictionary and their uses are shown in the following list:

- `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

2.11 Hands Detection

Hand detection is a research hotspot in the field of human-computer interaction, allowing people to issue instructions to the computer without the mouse and keyboard. For example, use your fingers to command the robot, or play a guessing game with the robot, and so on.

In this project, you will see that the hand in front of the camera is recognized and the coordinates of the index finger will be printed.



Run the Code

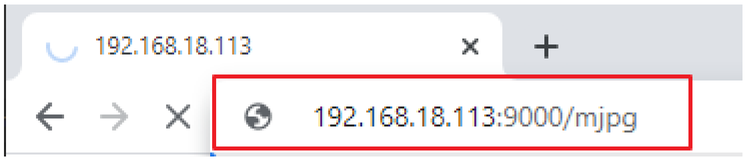
```
cd /home/pi/pan-tilt-hat/examples
sudo python3 hands_detection.py
```

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`



192.168.18.113:9000/mjpg

Code

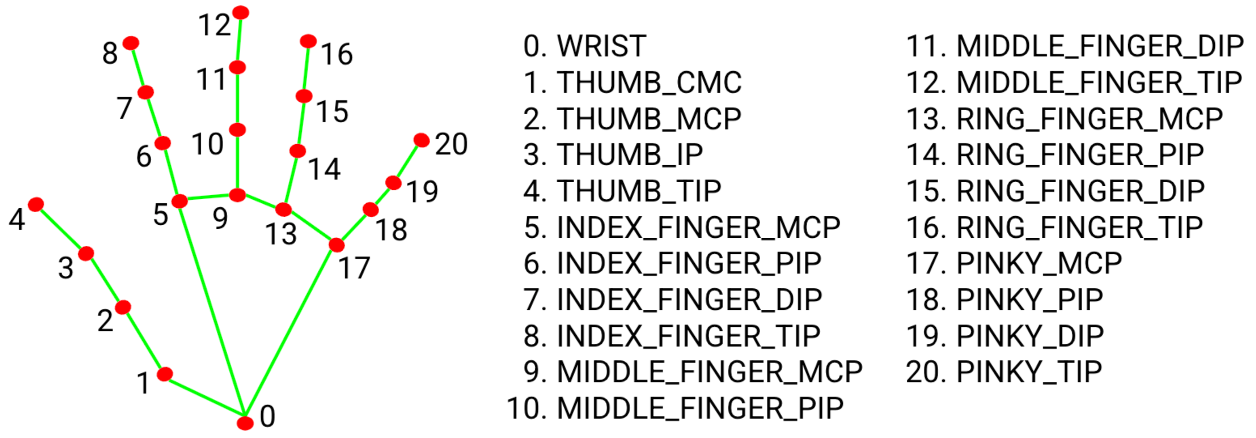
```python
from vilib import Vilib
from time import sleep

def main():
    Vilib.camera_start(vflip=True,hflip=True)
    Vilib.display(local=True,web=True)
    Vilib.hands_detect_switch(True)
    joints = []
    while True:
        joints = Vilib.detect_obj_parameter['hands_joints']

        if isinstance(joints,list) and len(joints) == 21:
            print(joints[8])

        sleep(1)


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

How it works?

This function has been encapsulated in the `vilib` library, execute `Vilib.hands_detect_switch(True)` to start hand detection.

The information detected by the target will be stored in the `Vilib.detect_obj_parameter['hands_joints']`.

The three-dimensional coordinates of the 21 joints of the hand are stored here, such as the `joints[8]` printed in this article, that is, the coordinates of the index finger. The serial numbers of all joints are shown in the figure below.



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

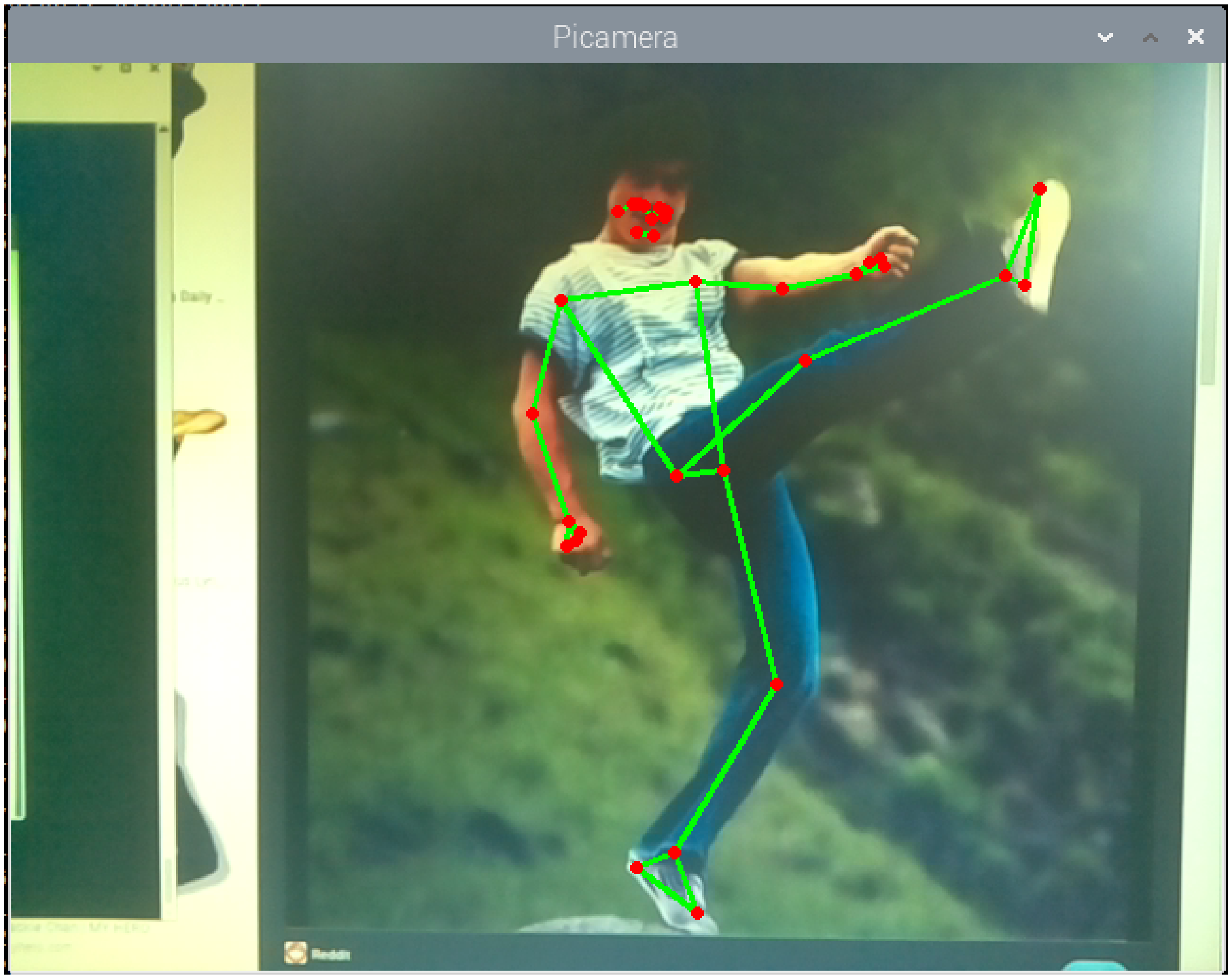
The coordinates of these joints are composed of [x,y,z]. x and y are normalized to 0.0 ~ 1.0 by the width and height of the image (rather than the specific pixel position), the upper left corner of the screen is [0.0, 0.0], and the lower right corner is [1.0, 1.0]. The z coordinate represents the depth, and the wrist depth is the origin. The smaller the value, the closer the landmark is to the camera. The size of z uses roughly the same ratio as x.

This feature is based on MediaPipe Google, please see more knowledge in Hands - MediaPipe .

2.12 Pose Detection

Like hand detection, posture detection is also part of the field of human-computer interaction. It can be used to judge action instructions, identify dances, quantify exercises and other scenes.

In this project, you will see that the human body in front of the camera is recognized and the coordinates of the left wrist are printed.



Run the Code

```
cd /home/pi/pan-tilt-hat/examples
sudo python3 pose_detection.py
```

View the Image

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

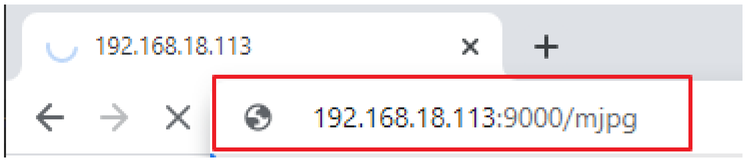
```
No desktop !
* Serving Flask app "vilib.vilib" (lazy loading)
* Environment: production
```

(continues on next page)

(continued from previous page)

```
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

```python
from time import sleep
from vilib import Vilib


def main():

    Vilib.camera_start(vflip=True,hflip=True)
    Vilib.display(local=True,web=True)
    Vilib.pose_detect_switch(True)

    joints = []
    while True:
        joints = Vilib.detect_obj_parameter['body_joints']

        if isinstance(joints,list) and len(joints) == 33:
            print(joints[15])

        sleep(1)
```

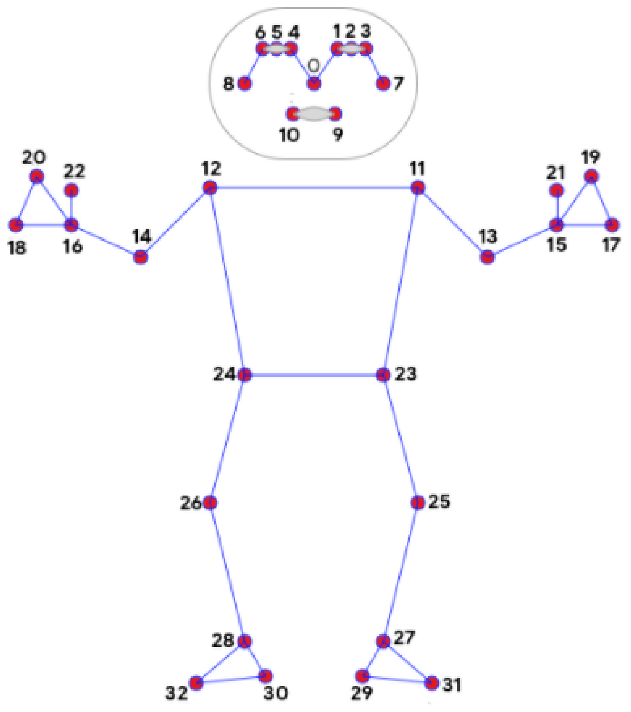
if __name__ == "__main__": main()

How it works?

This function has been encapsulated in the `vilib` library, execute `Vilib.pose_detect_switch(True)` to start gesture detection.

The information detected by the target will be stored in the `Vilib.detect_obj_parameter['body_joints']`.

The three-dimensional coordinates of the 33 joints of the hand are stored here, such as the `joints[15]` printed in this article, which is the coordinates of the left wrist. The serial numbers of all joints are shown in the figure below.



0. nose
1. left_eye_inner
2. left_eye
3. left_eye_outer
4. right_eye_inner
5. right_eye
6. right_eye_outer
7. left_ear
8. right_ear
9. mouth_left
10. mouth_right
11. left_shoulder
12. right_shoulder
13. left_elbow
14. right_elbow
15. left_wrist
16. right_wrist
17. left_pinky
18. right_pinky
19. left_index
20. right_index
21. left_thumb
22. right_thumb
23. left_hip
24. right_hip
25. left_knee
26. right_knee
27. left_ankle
28. right_ankle
29. left_heel
30. right_heel
31. left_foot_index
32. right_foot_index

The coordinates of these joints are composed of [x,y,z,visiblity]. x and y are normalized to 0.0 ~ 1.0 by the width and height of the image (rather than the specific pixel position), the upper left corner of the screen is [0.0, 0.0], and the lower right corner is [1.0, 1.0].

The z coordinate represents the depth, with the crotch (between 23 and 24) depth as the origin. The smaller the value, the closer the landmark is to the camera. The size of z uses roughly the same ratio as x. visibility is a value indicating the possibility that the joint is visible (existing and not occluded) in the image, and its range is between 0.0 and 1.0.

This feature is based on MediaPipe Google, please see more knowledge in Pose - MediaPipe .

CHAPTER

THREE

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.

3.1 Quick Guide on Ezblock

There are 2 parts here:

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

3.1.1 Before Programming With Ezblock

First download Ezblock Studio.

For a detailed installation tutorial, please refer to: Install Ezblock Studio.

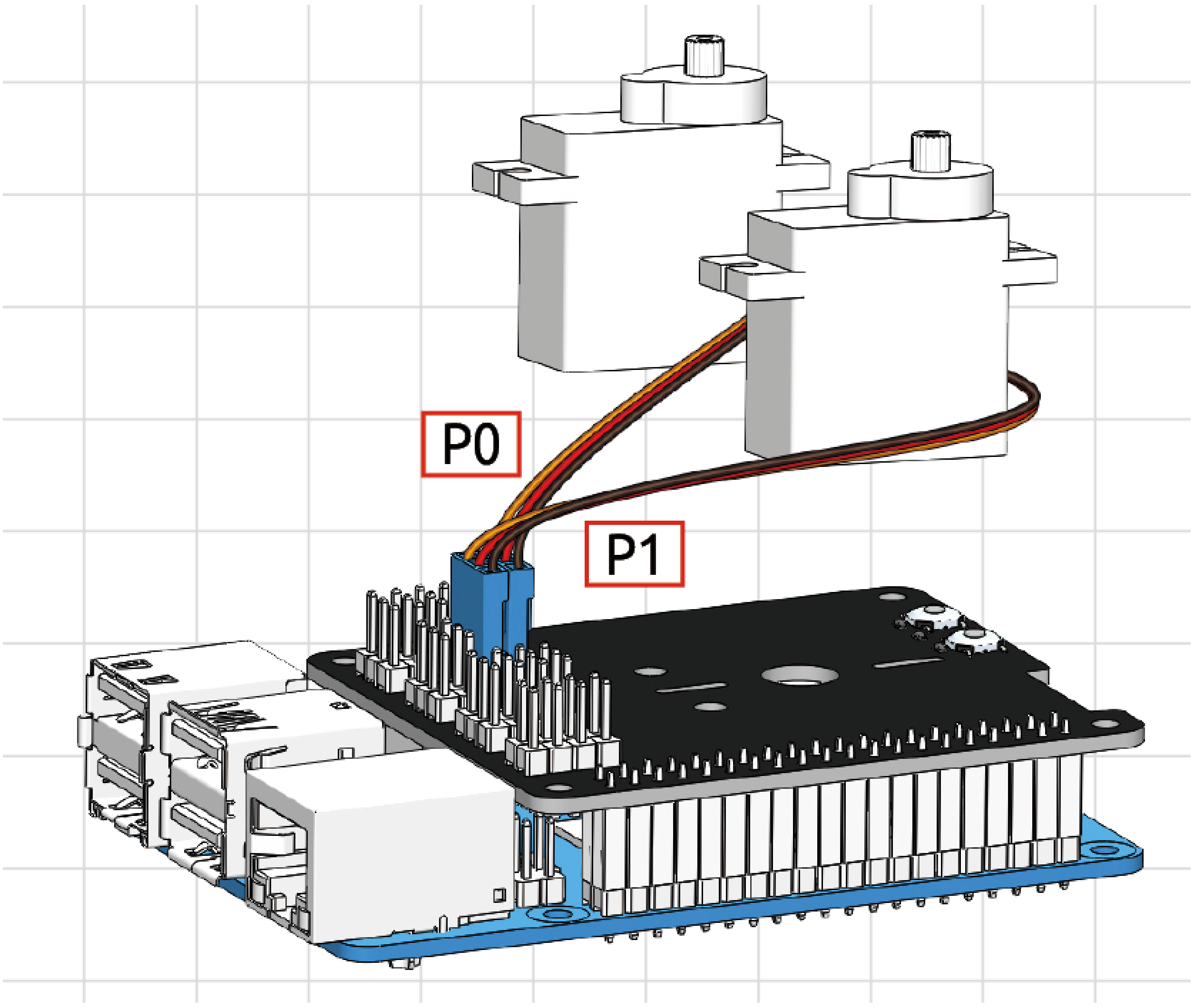
Note: After you connect the Pan-tilt HAT, there will be a calibration step. This is because of possible deviations in the installation process or limitations of the servos themselves, making some servo angles slightly tilted, so you can calibrate them in this step.

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

3.1.2 Before Assembling Pan-tilt HAT

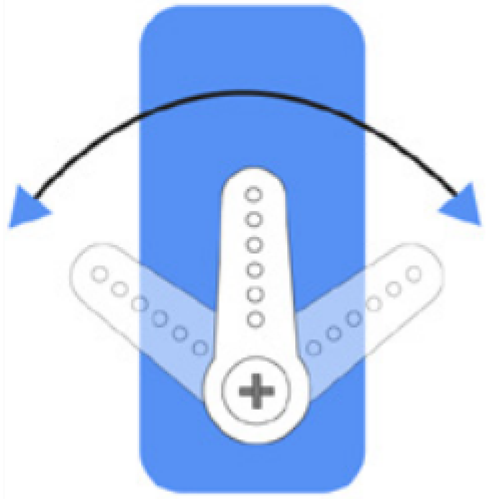
Before assembling the Pan-tilt HAT, follow the instructions on how to install the Ezblock OS on an Micro SD card here: Download and Install Ezblock OS.

Temporarily combine the HAT and Raspberry Pi. Run “Color Detection” Example (This example will put the servo at the zero position).



Insert the servo cable into the P0 & P1 port. Now the servo arm should return to the 0° position.

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.



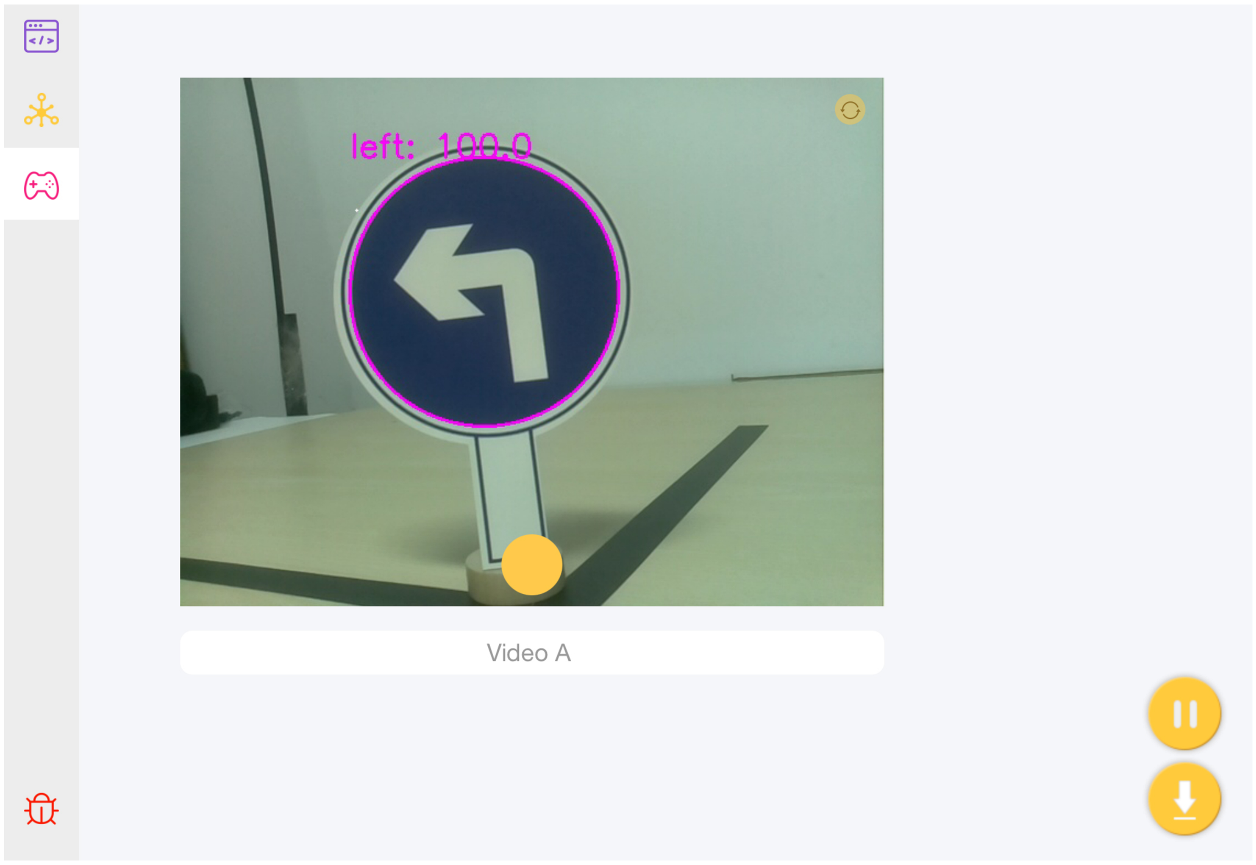
Projects

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.

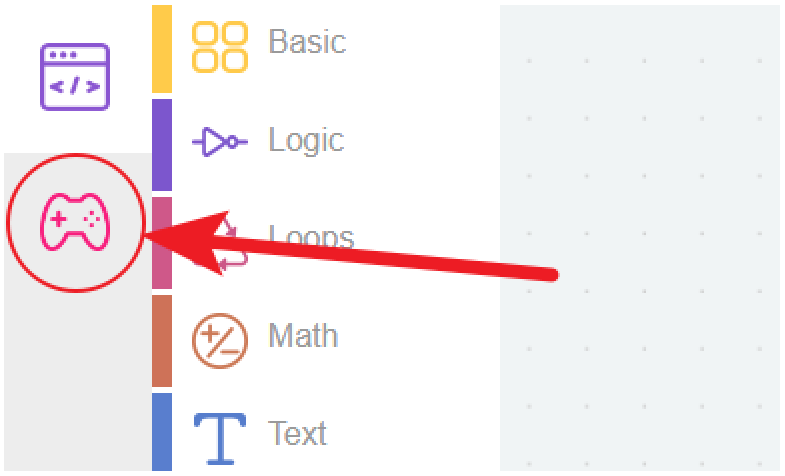
3.2 Traffic Sign Detection - Ezblock

Firstly, we learn how to use Pan-Tilt HAT to detect traffic signs. During this operation, we can master how to enable and use RPi camera.



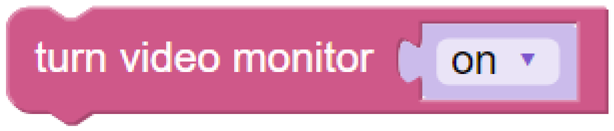
TIPS

Click the this icon to enter the Bluetooth control page to enable the camera monitor.

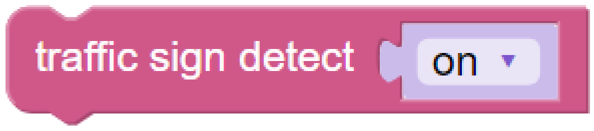


Here we drag a Video from the page, and it will generate a monitor.

You need to open the image in Video by turning video monitor to on. Turning it to off will close the image (but not object detection).



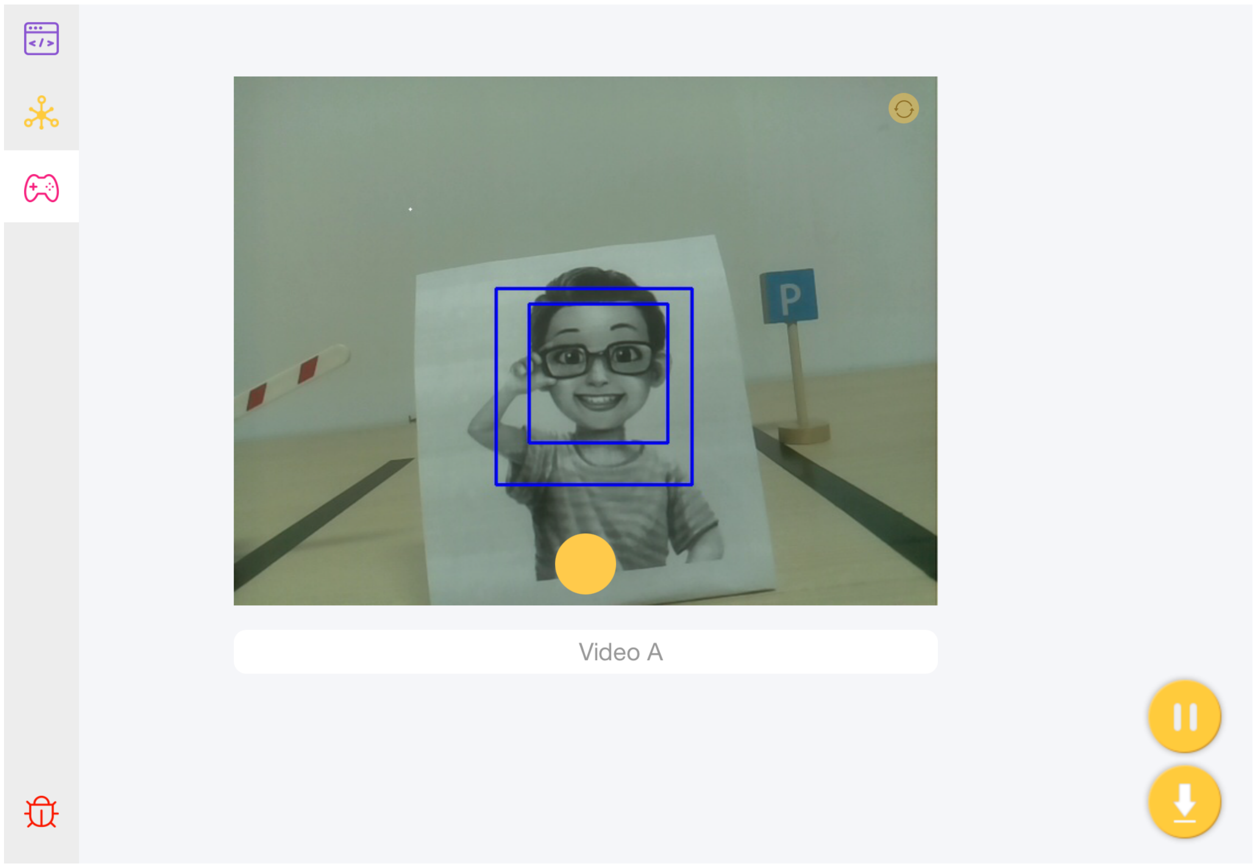
Turn on traffic sign detection.



EXAMPLE

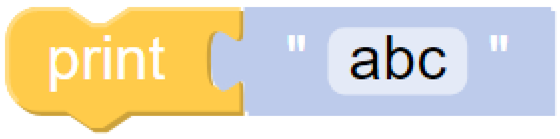
3.3 Face Detection - Ezblock

Next, it's time to go in to learn about the face detection of Pan-Tilt HAT. The Pan-Tilt HAT can print the number of the detected faces on the debug monitor.

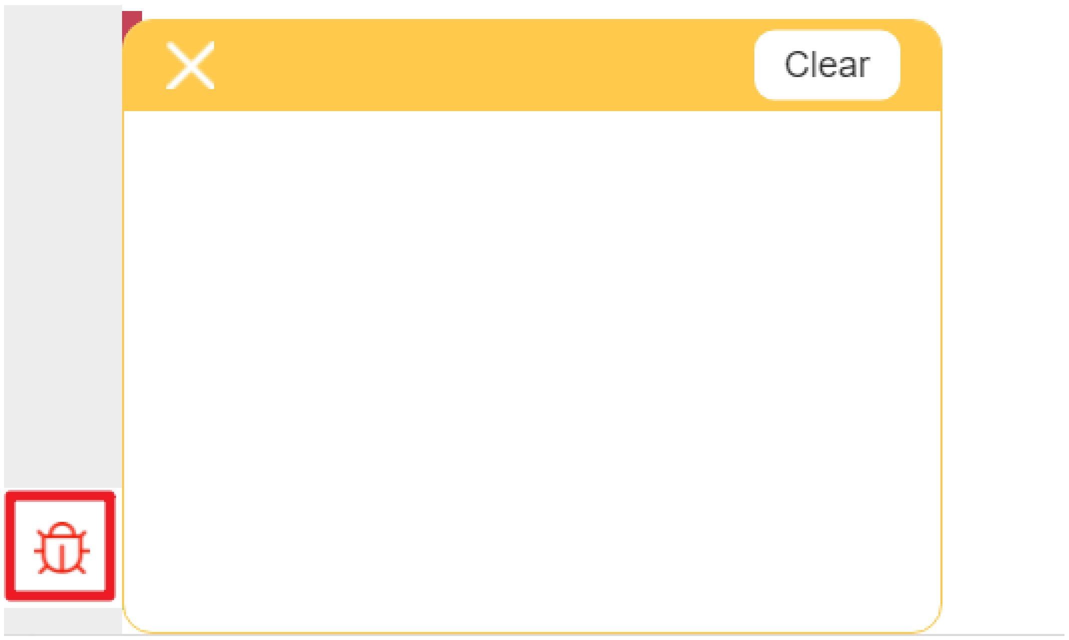


TIPS

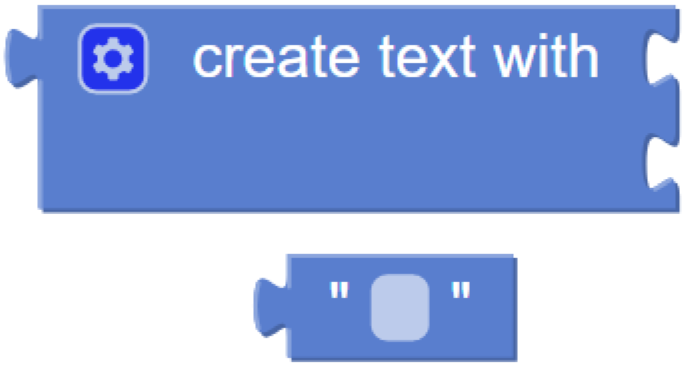
A print block can be used to print data or text on the debug monitor.



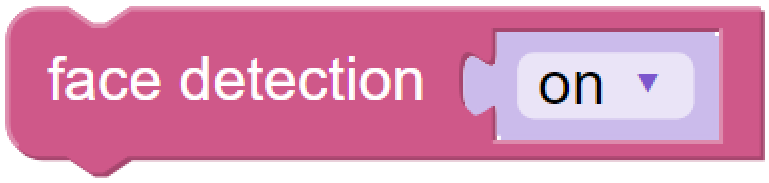
The data printed by the Print block will appear in the Debug Monitor on the left. In other pages, you can also click on the Debug Monitor in the upper right corner.



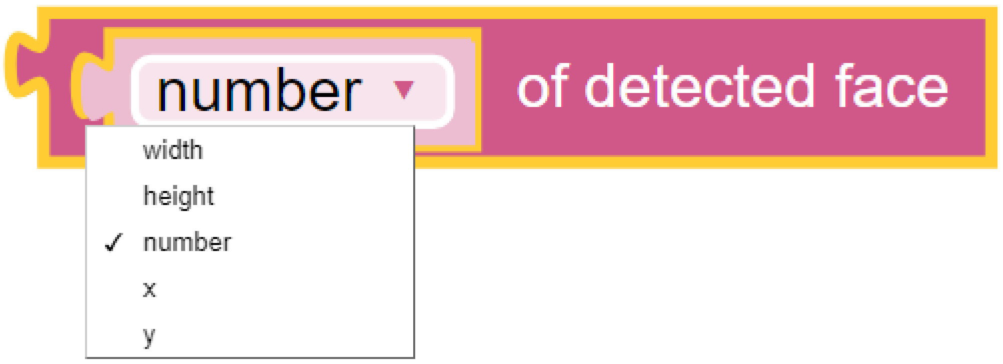
You may want to use text block to print the combination of texts and data at once.



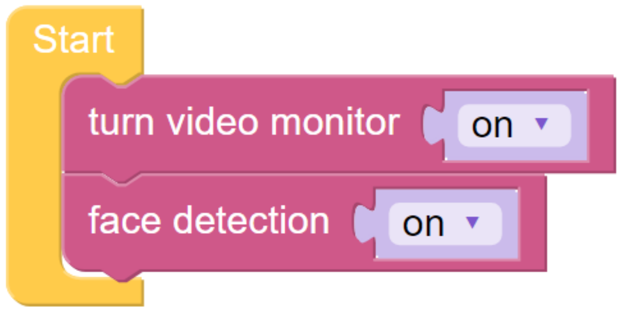
You need to turn on face detection.

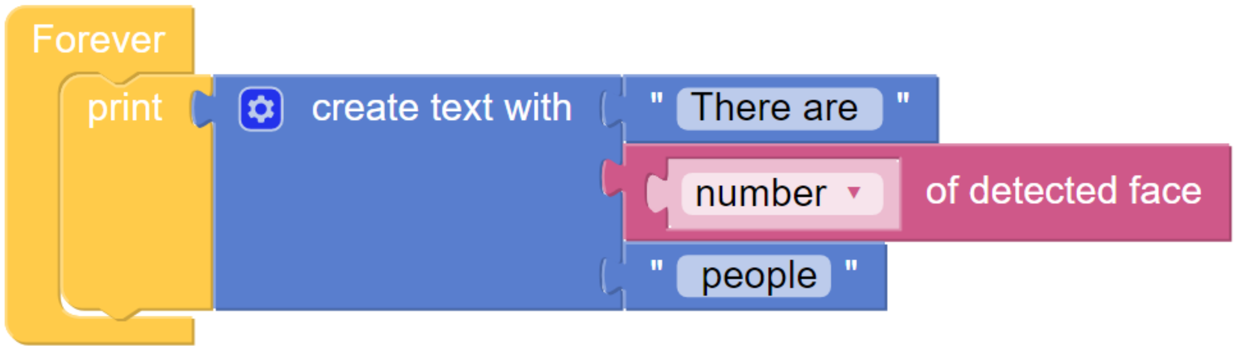


You can read the results of face detection through this block, modify the drop-down menu options, and choose to read the coordinates, size or number of the results of face detection.


number
of detected face
width
height
✓ number
x
y

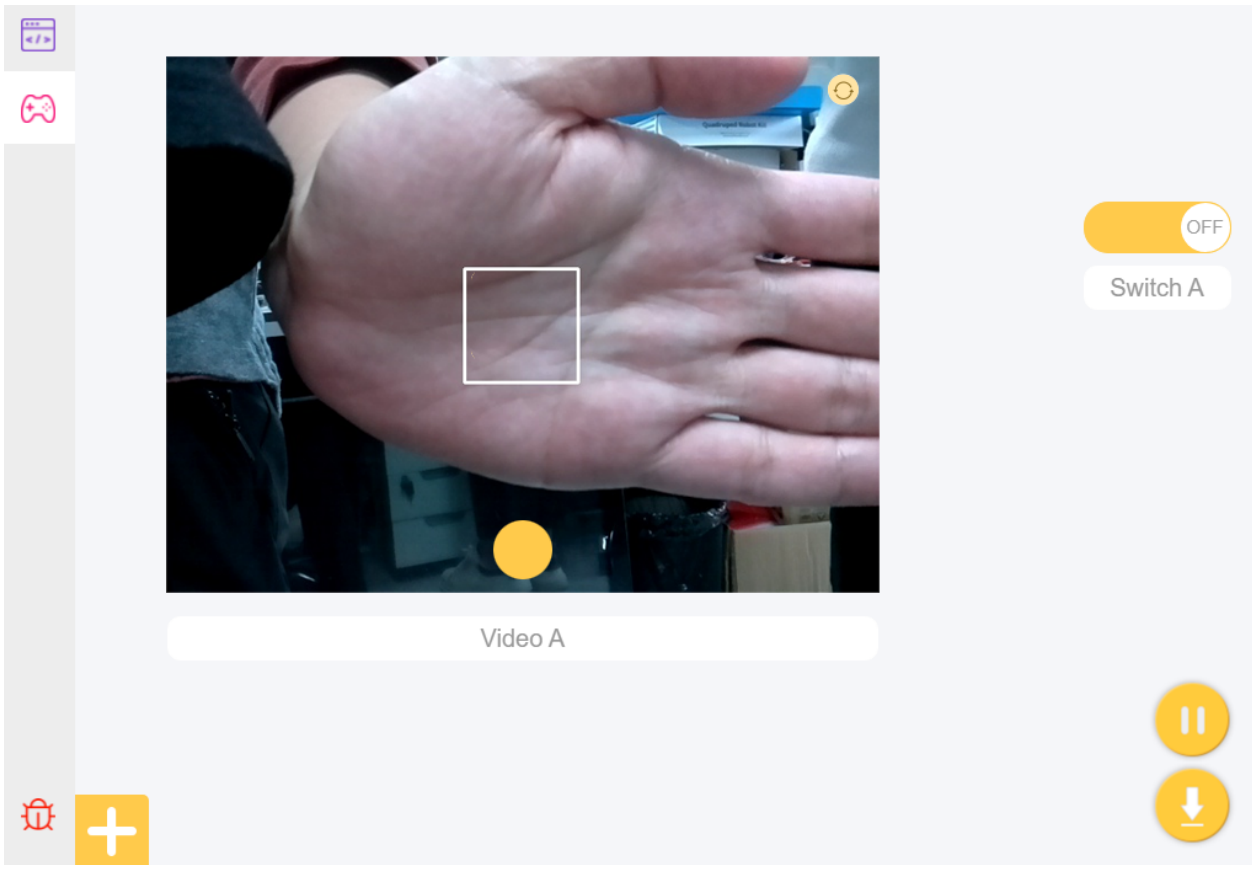
EXAMPLE


Start
turn video monitor on
face detection on


Forever
print
create text with
" There are "
number of detected face
" people "

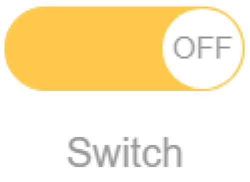
3.4 Gesture Calibration - Ezblock

Gesture detection is easily influenced by environmental factors, so we need first to calibrate gestures then enable gesture detection.

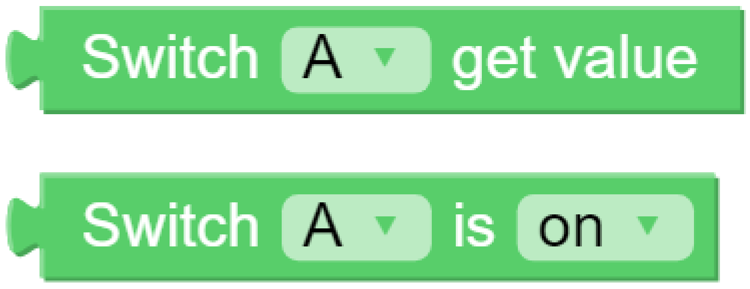


TIPS

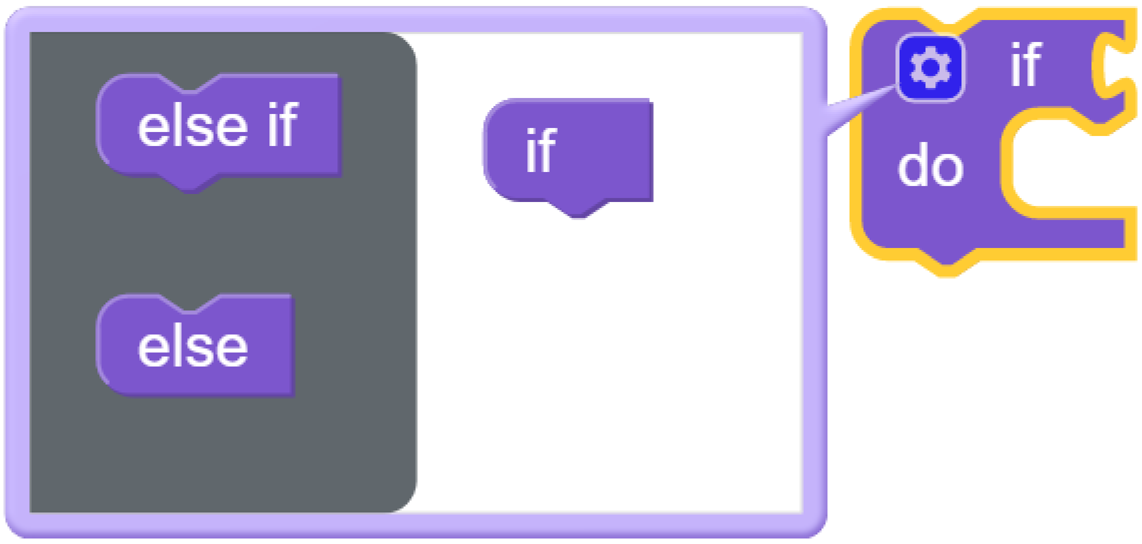
Here we drag a Switch from the Bluetooth control page. After that a Remote category will appear.



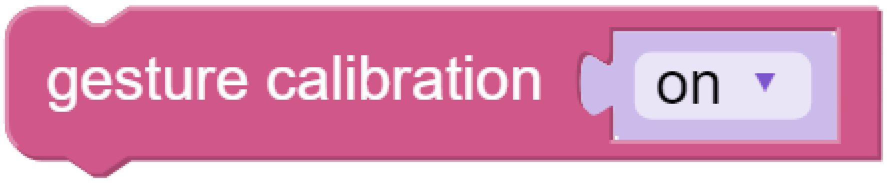
This blocks read the switch state in the Bluetooth control page.



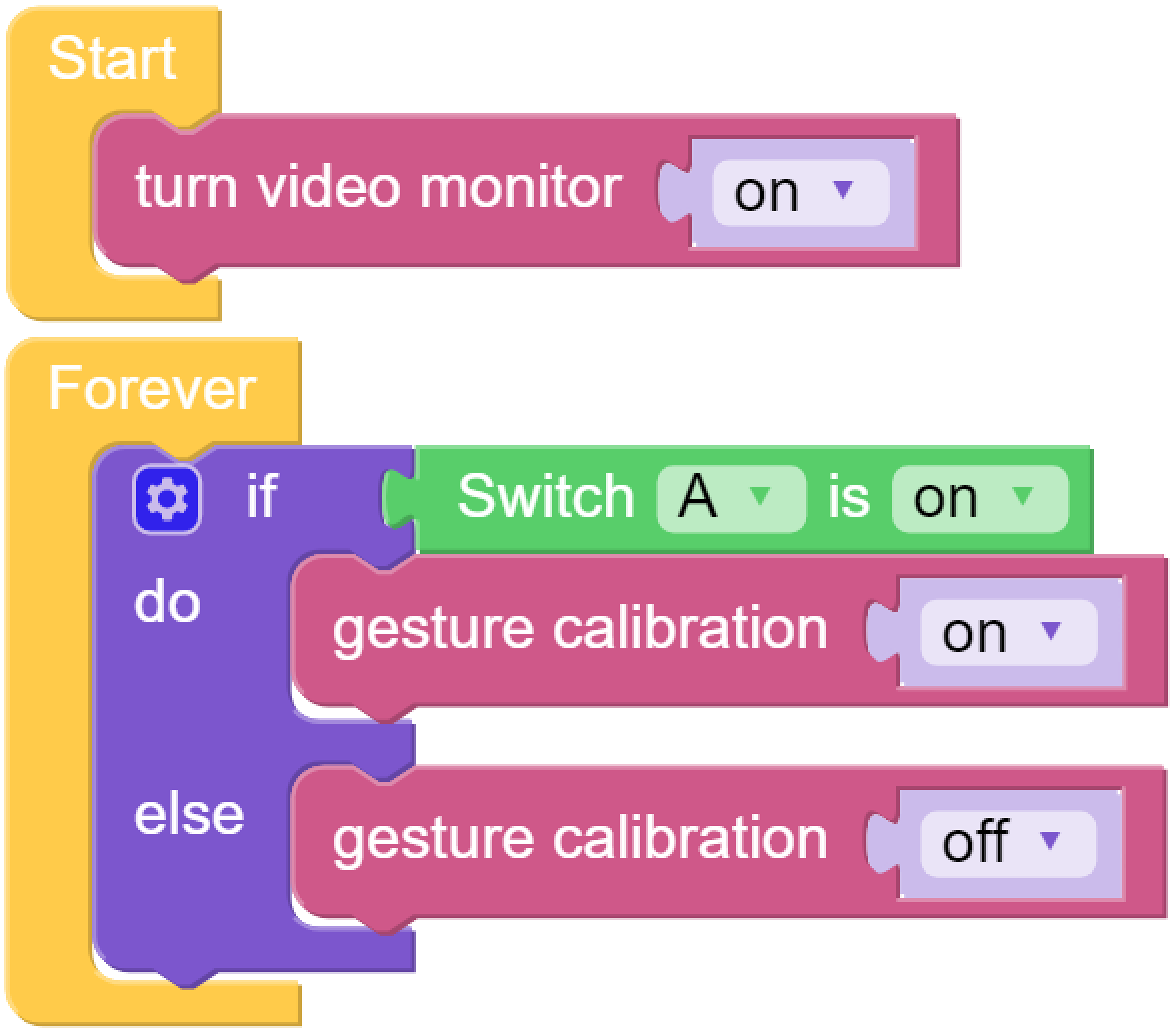
To achieve conditional judgment of "if" type, you need to use an if do block.



Calibrate your gesture by using this block. In the drop-down menu, choose on, the calibration will be on; otherwise, the calibration will be off.



EXAMPLE

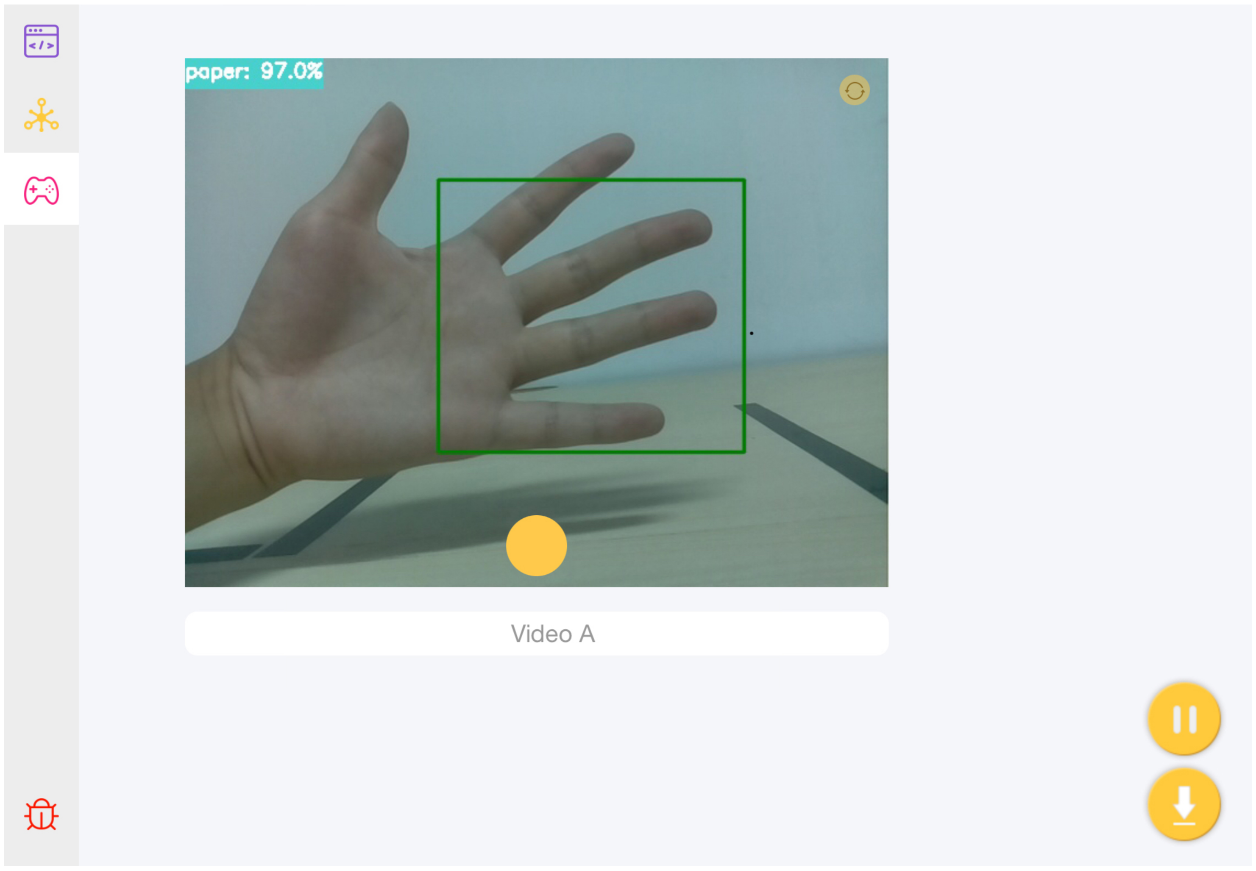


After the code is uploaded, let the switch "ON" , there will appear a white square in the center of camera monitor. Put out your hand and place the palm in the square. Let it "OFF" when you finish the calibration.

Note: When the gesture is being calibrated, the camera should avoid being directly radiated by light. After calibration is complete, you need to press the reset key on the Pan-Tilt HAT to take effect.

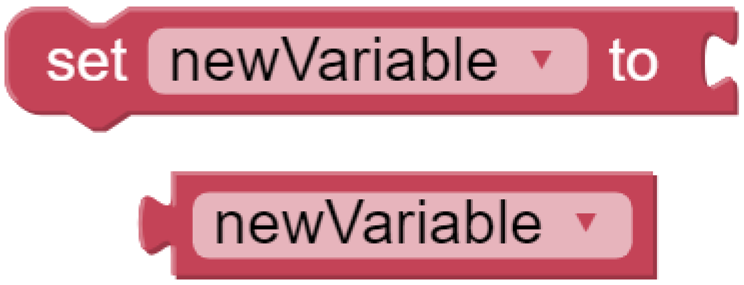
3.5 Gesture Detection - Ezblock

Completed the gesture calibration, we can start using Pan-Tilt HAT to detect our gestures. Now, what can be detected by our Pan-Tilt HAT includes rock, scissor, paper.



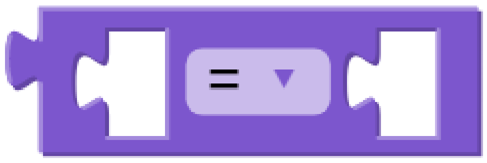
TIPS

You may want to simplify your program with Variable. For example, when you need to use the values of the type of gesture repeatedly, just assign them to variables for repeated use instead of reading repeatedly.

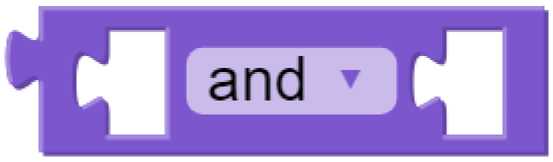


Click the Create variable button on the Variables category to create a new variable.

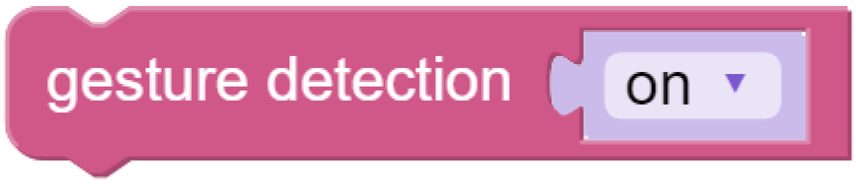
This block is used to do conditional judgement. Judging conditions can be "=", ">", "<", "", "", "".



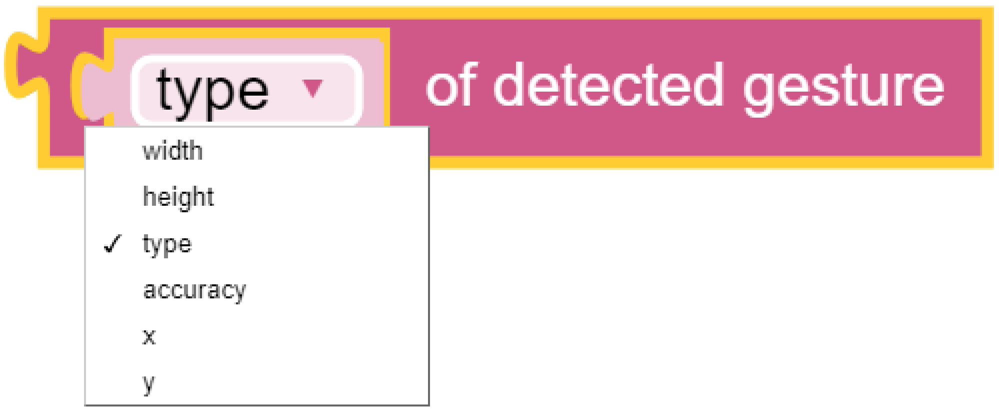
This block is used to do "logical judgment" and judging conditions can be "and", "or" etc.



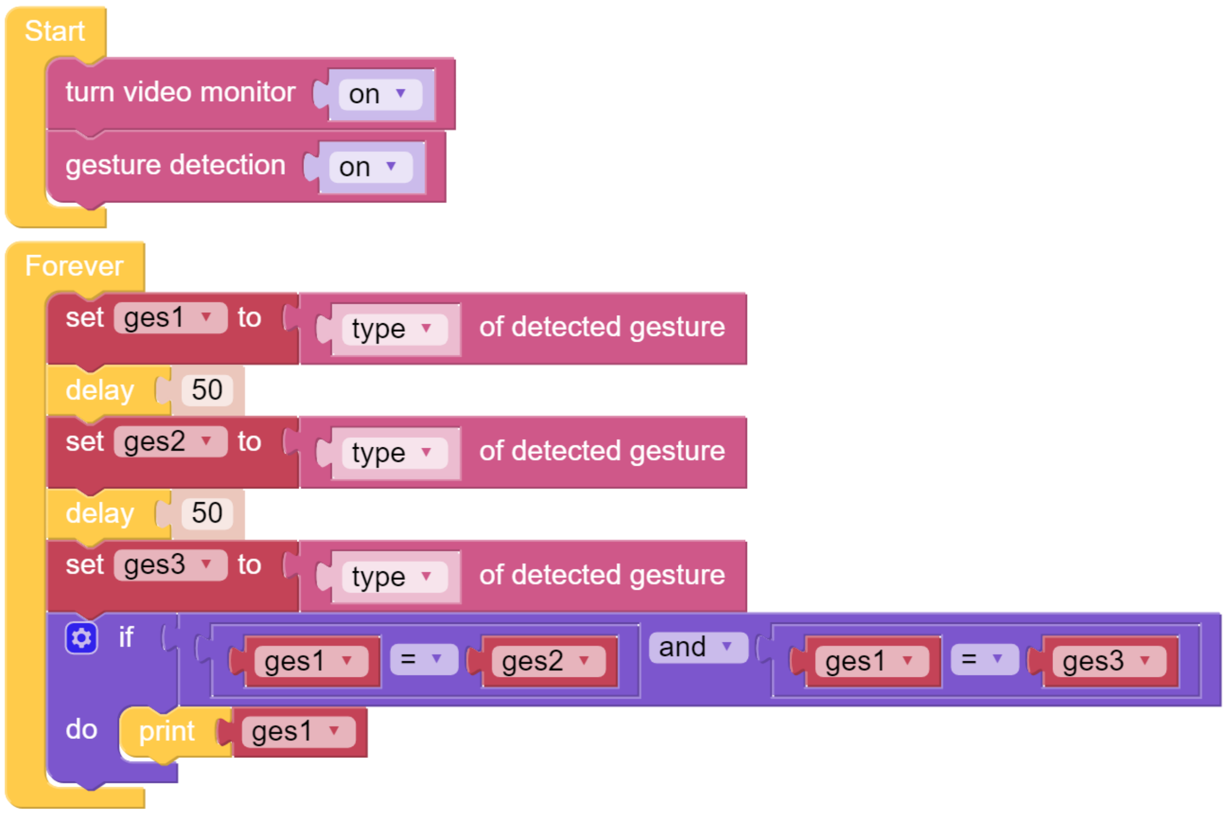
You need to turn on gesture detection.



You can read the results of the detected gesture through this block. In the drop-down menu, choose to read the coordinates, size, type or accuracy of the detected gesture.



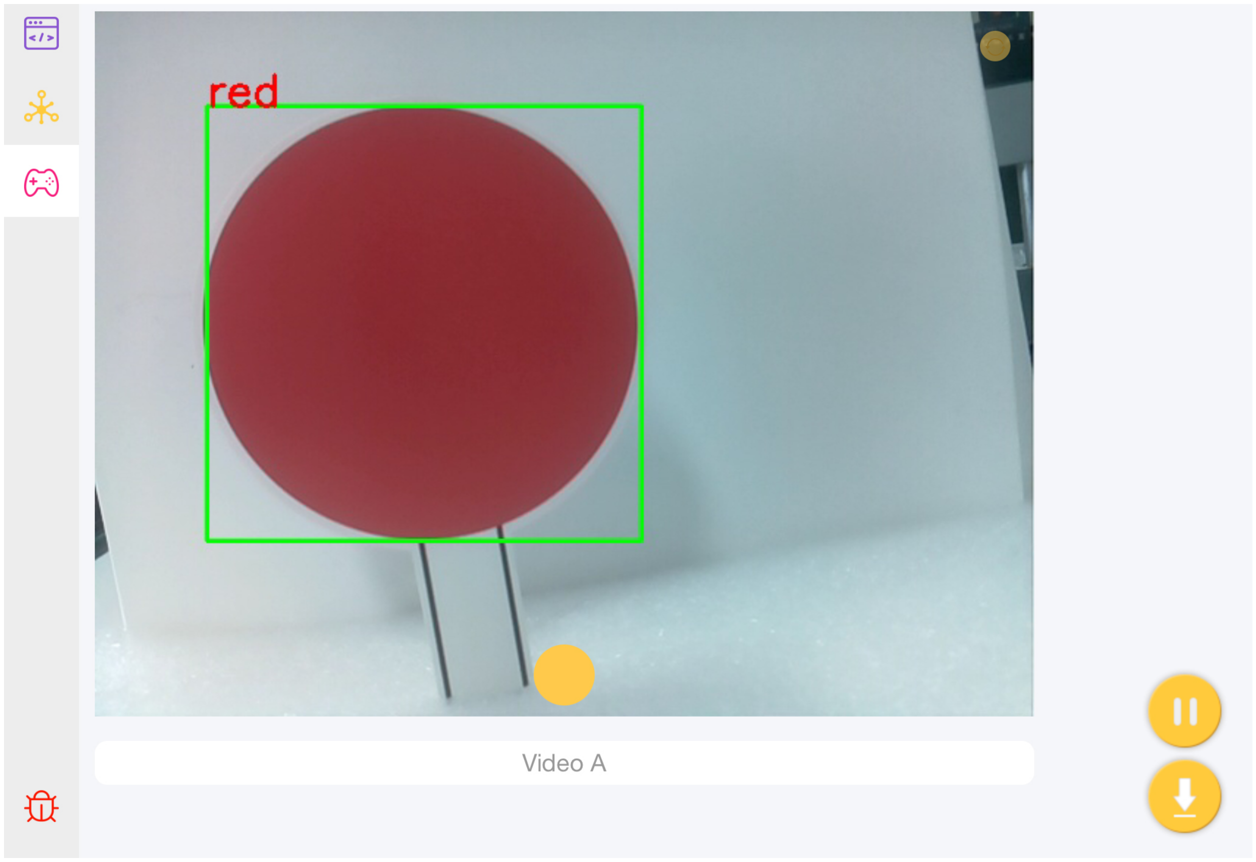
EXAMPLE



```
Start
  turn video monitor [on]
  gesture detection [on]

Forever
  set [ges1] to [type] of detected gesture
  delay 50
  set [ges2] to [type] of detected gesture
  delay 50
  set [ges3] to [type] of detected gesture
  if [ges1] [=] [ges2] [and] [ges1] [=] [ges3]
  do print [ges1]
```

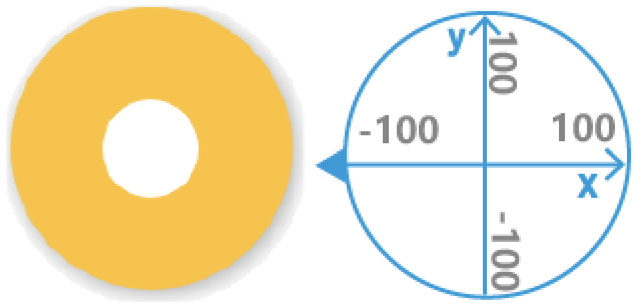
3.6 Color Detection - Ezblock

Let us learn the color detection of Pan-Tilt HAT. Let's adjust the direction of the camera with the joystick and check the color detection results displayed on the camera monitor.

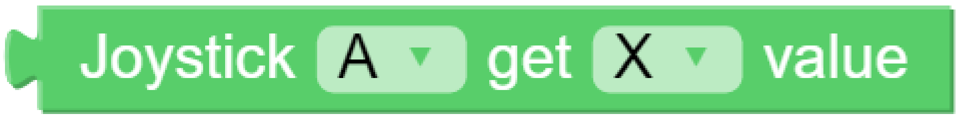


TIPS

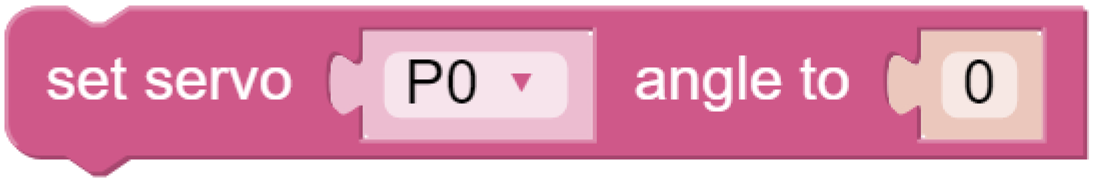
Drag the joystick to the center area. When using it, drag the white point in it to generate coordinates (-100~100).



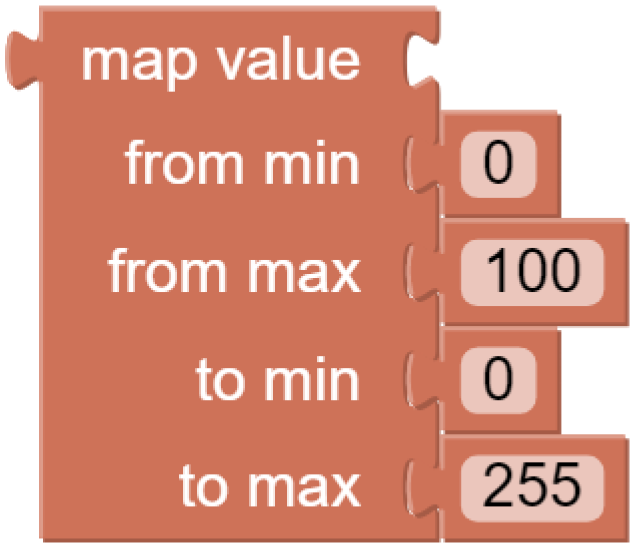
This block reads the Joystick value in the Bluetooth control page. You can click the drop-down menu to switch to the Y-axis reading.



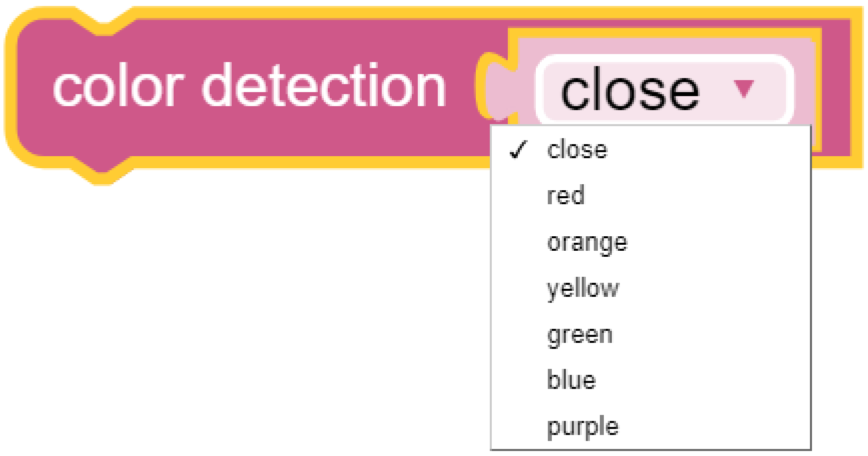
This block is used to drive the servo arm to rotate in a certain direction.



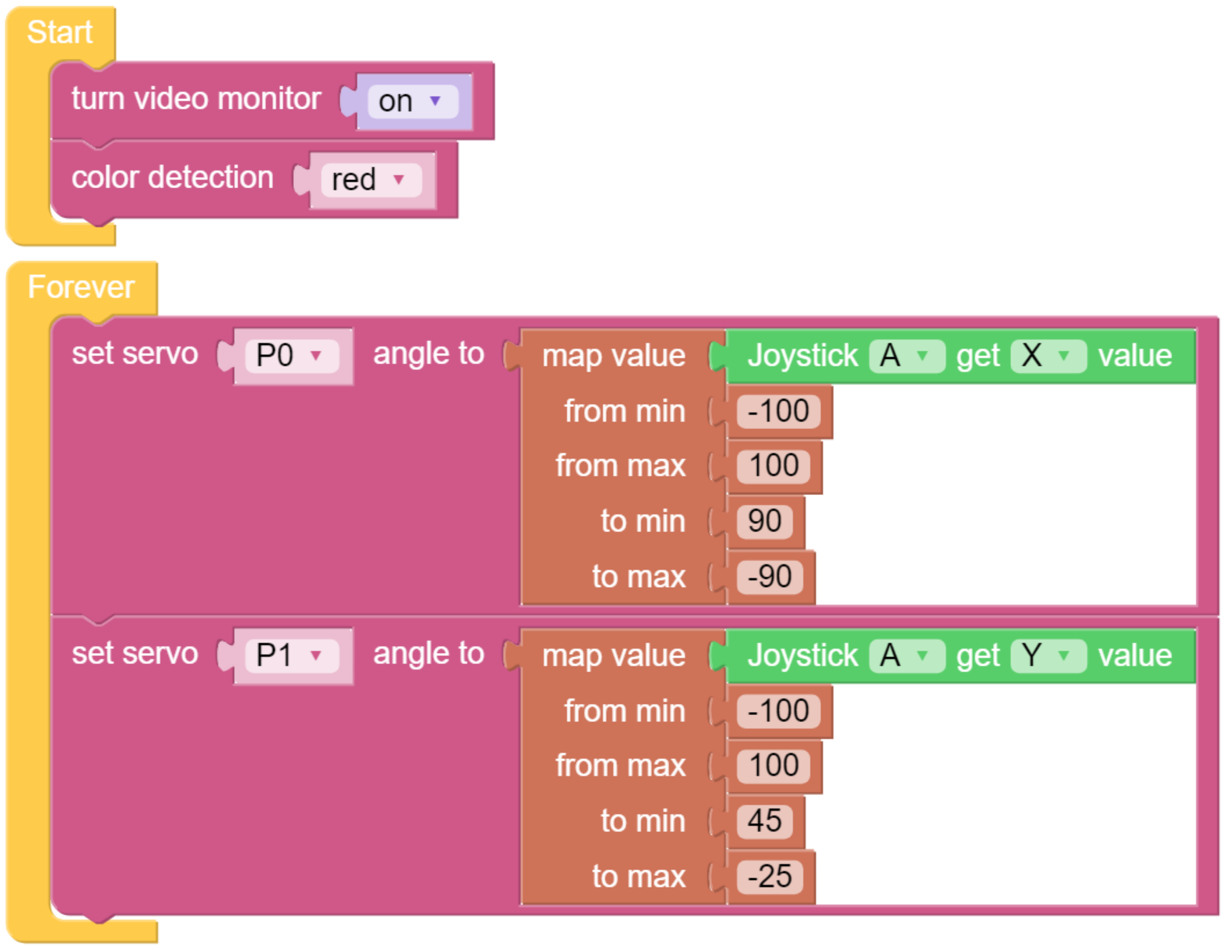
The map block can remap a number from one range to another. If a number is 50it is at 50% position of the range of 0~100; then if we map it to the range 0~255 via the map block, the number will be 127.5.



You can use this block to choose the detected color. Only one color can be detected for each detection.



EXAMPLE

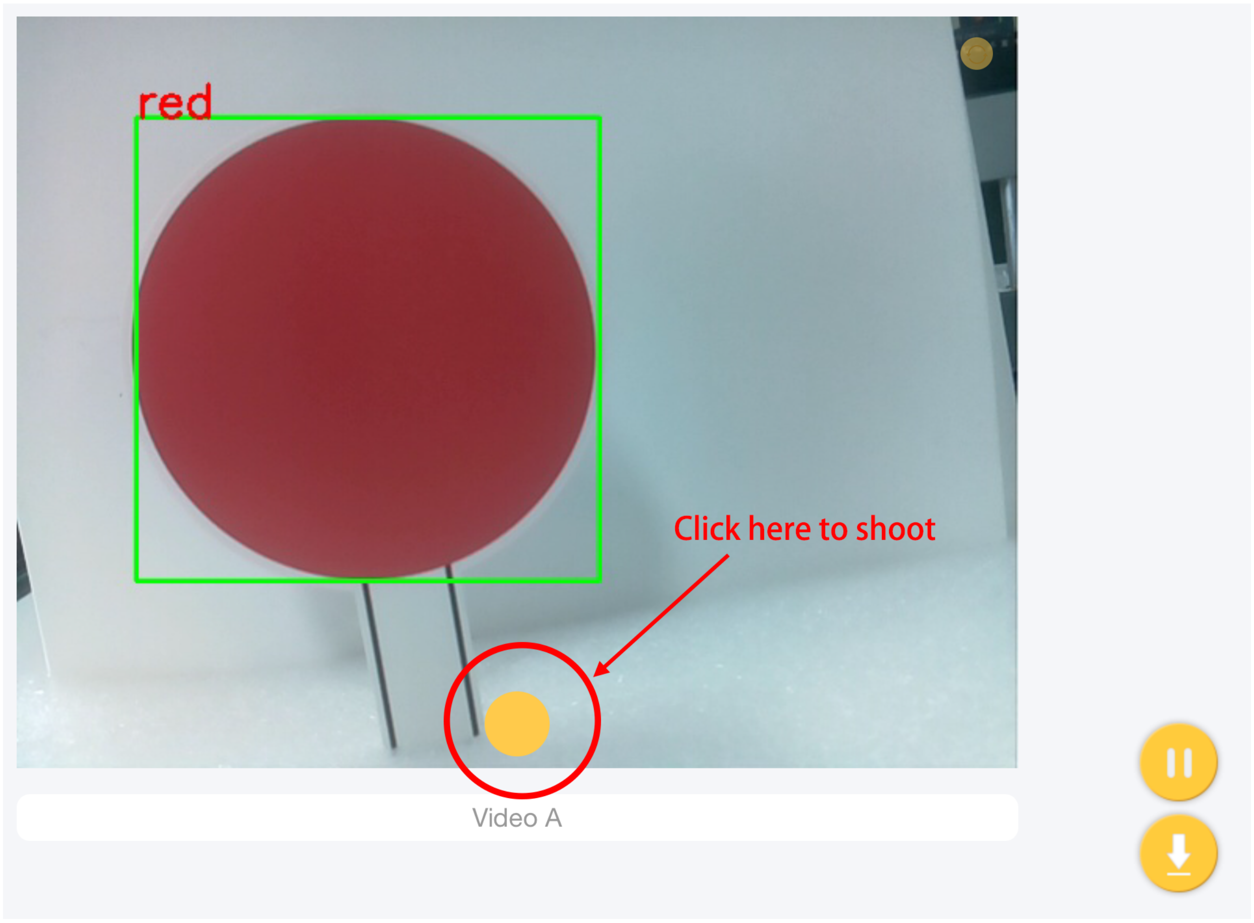


```
Start
  turn video monitor on
  color detection red
Forever
  set servo P0 angle to map value Joystick A get X value
                        from min -100
                        from max 100
                        to min 90
                        to max -90
  set servo P1 angle to map value Joystick A get Y value
                        from min -100
                        from max 100
                        to min 45
                        to max -25
```

3.7 Take Photo - Ezblock

Pan-Tilt HAT can take a photo. We use button to simulate a shutter, and once you press it, you can get a photo. This method will store the photos in the Raspberry Pi. In some projects that use algorithms to process images, this feature will be useful.

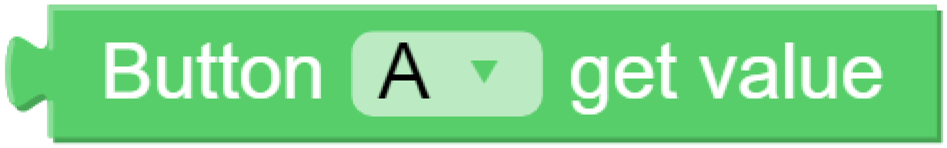
If you need to store the photos directly on the tablet (or computer), please click the dot on the Video widget.

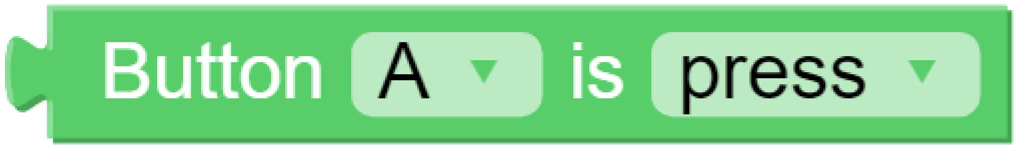


TIPS

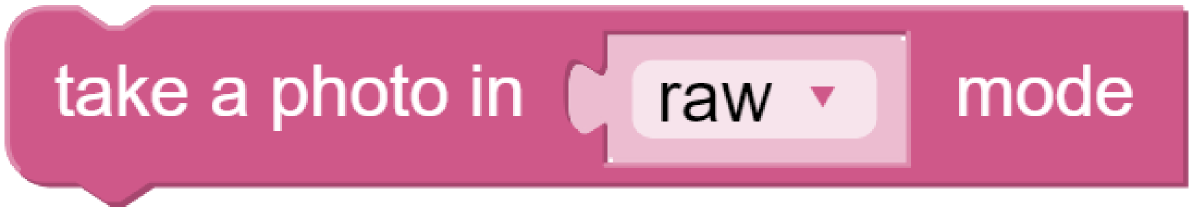
Drag a Button to the central area.

Judge if the button is pressed.



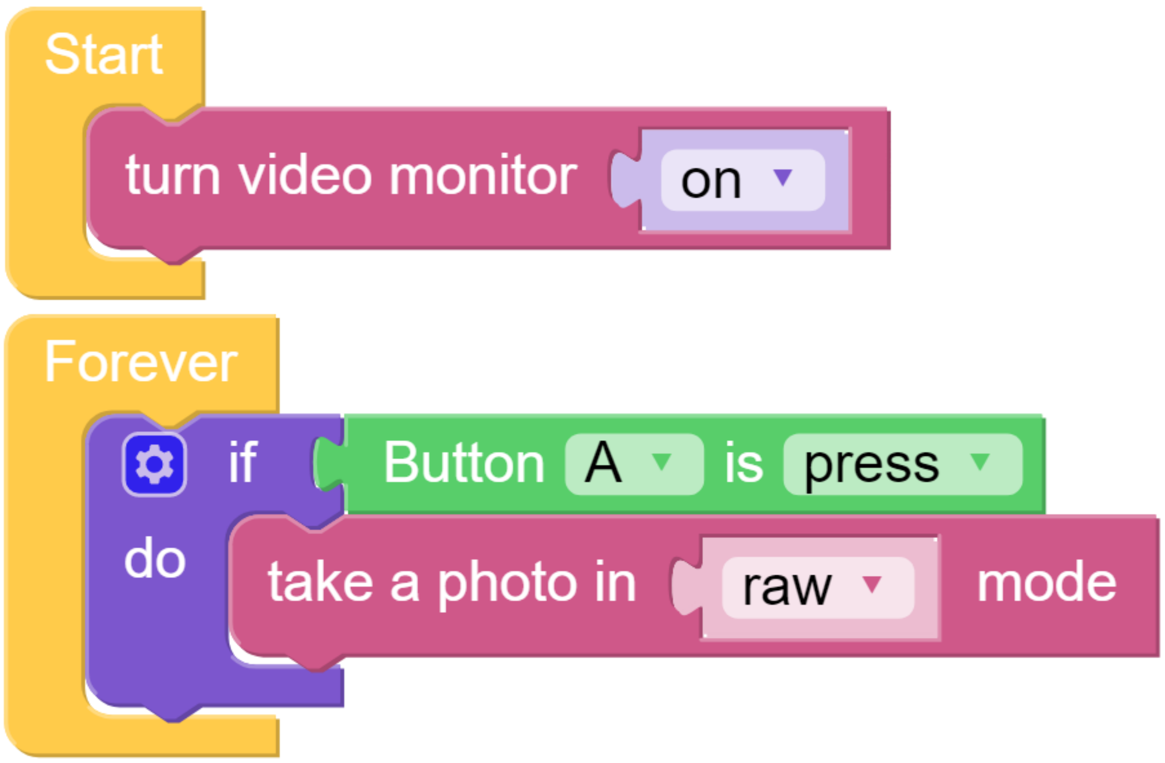


Use this block to take a photo. In the drop-down menu, choose raw or opencv.



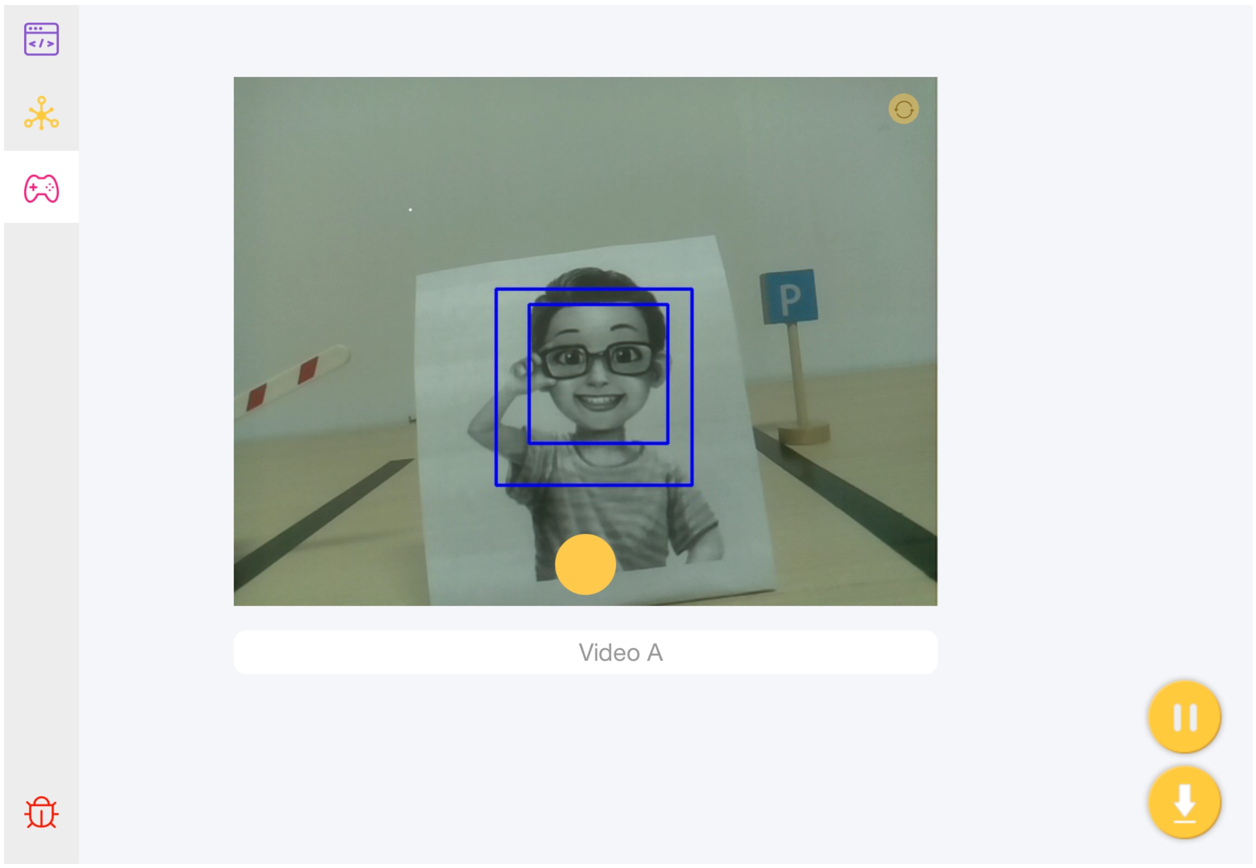
Note: Before you choose opencv, enable at least one kind of detection (color/face/gesture/traffic sign detection). During your shooting, the information of detected objects appears with a frame; for example, if the color detection (red) is enabled, on the photo, the red area will be marked with a frame and the information "red" will be indicated.

EXAMPLE

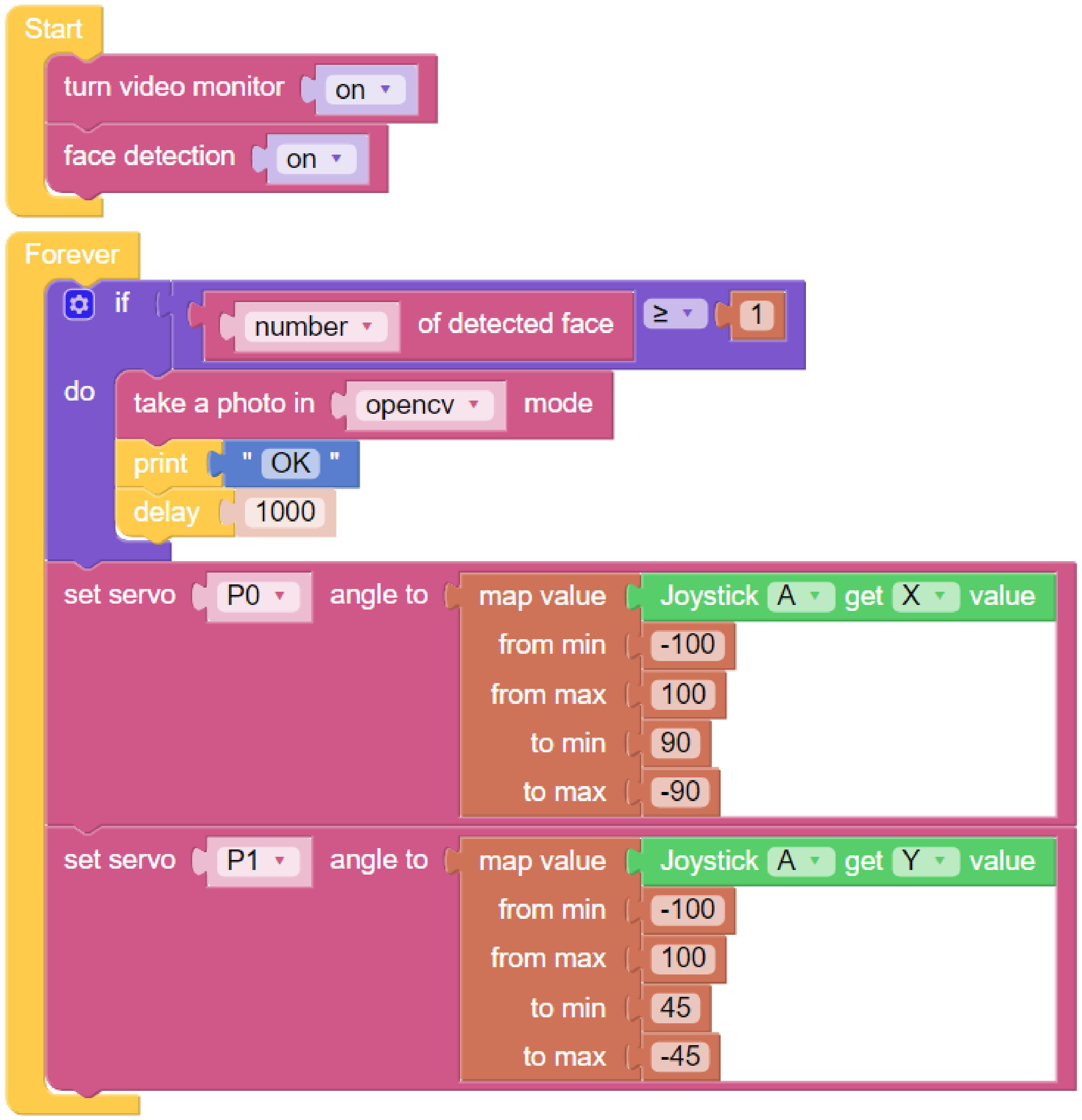


3.8 Say Cheese

Let's try to implement a simple project with Pan-Tilt HAT: remotely control Pan-Tilt HAT's movement. When Pan-Tilt HAT sees people, it will take a photo then print "OK".



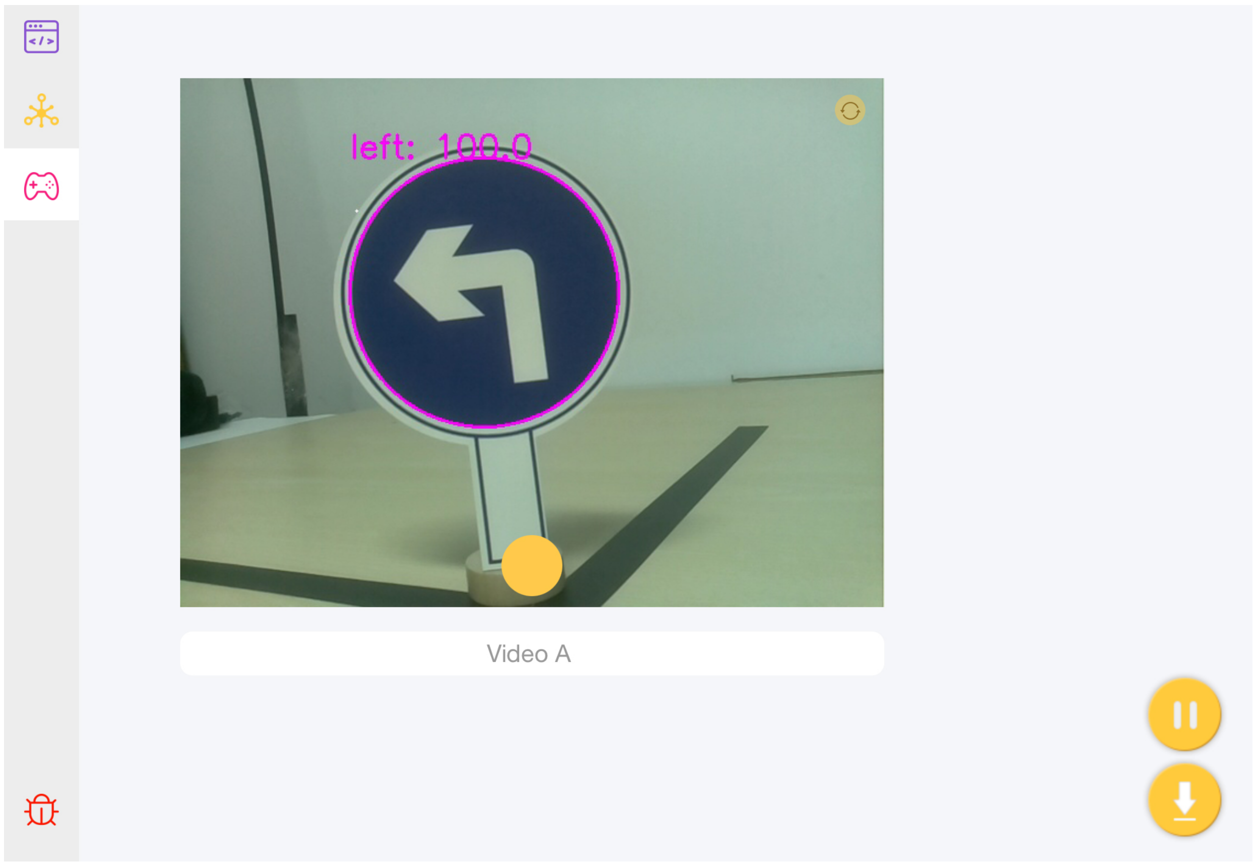
EXAMPLE



```
Start
    turn video monitor on
    face detection on

Forever
    if number of detected face ≥ 1
    do
        take a photo in opencv mode
        print " OK "
        delay 1000
    set servo P0 angle to map value Joystick A get X value
                          from min -100
                          from max 100
                          to min 90
                          to max -90
    set servo P1 angle to map value Joystick A get Y value
                          from min -100
                          from max 100
                          to min 45
                          to max -45
```

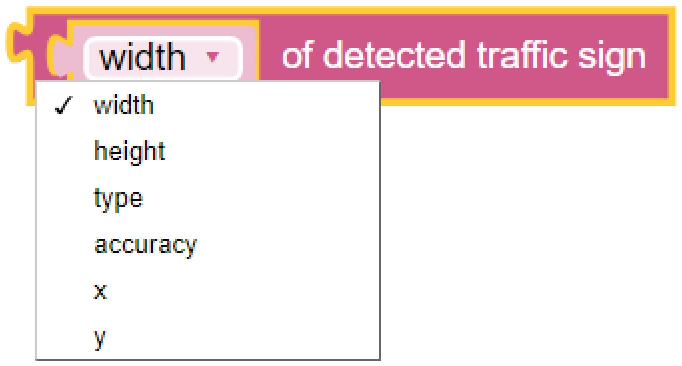
3.9 Get Order - Ezblock

Now Pan-Tilt HAT gets orders from traffic signs. when getting order “left”, it turns left.

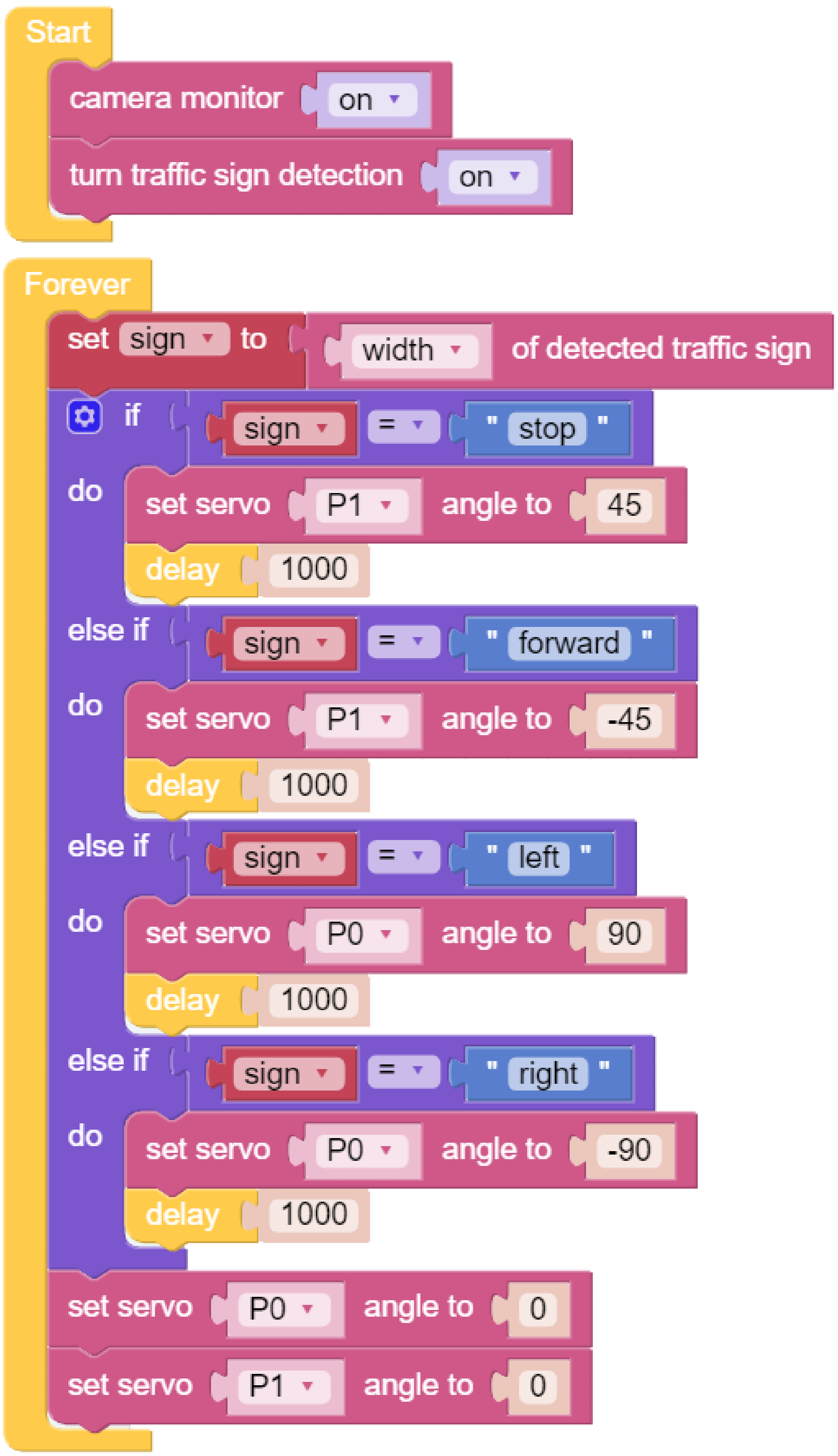


TIPS

You can detect the traffic signs through this block, modify the drop-down menu options, and choose to read the coordinates, size, type or accuracy of the detected traffic signs.



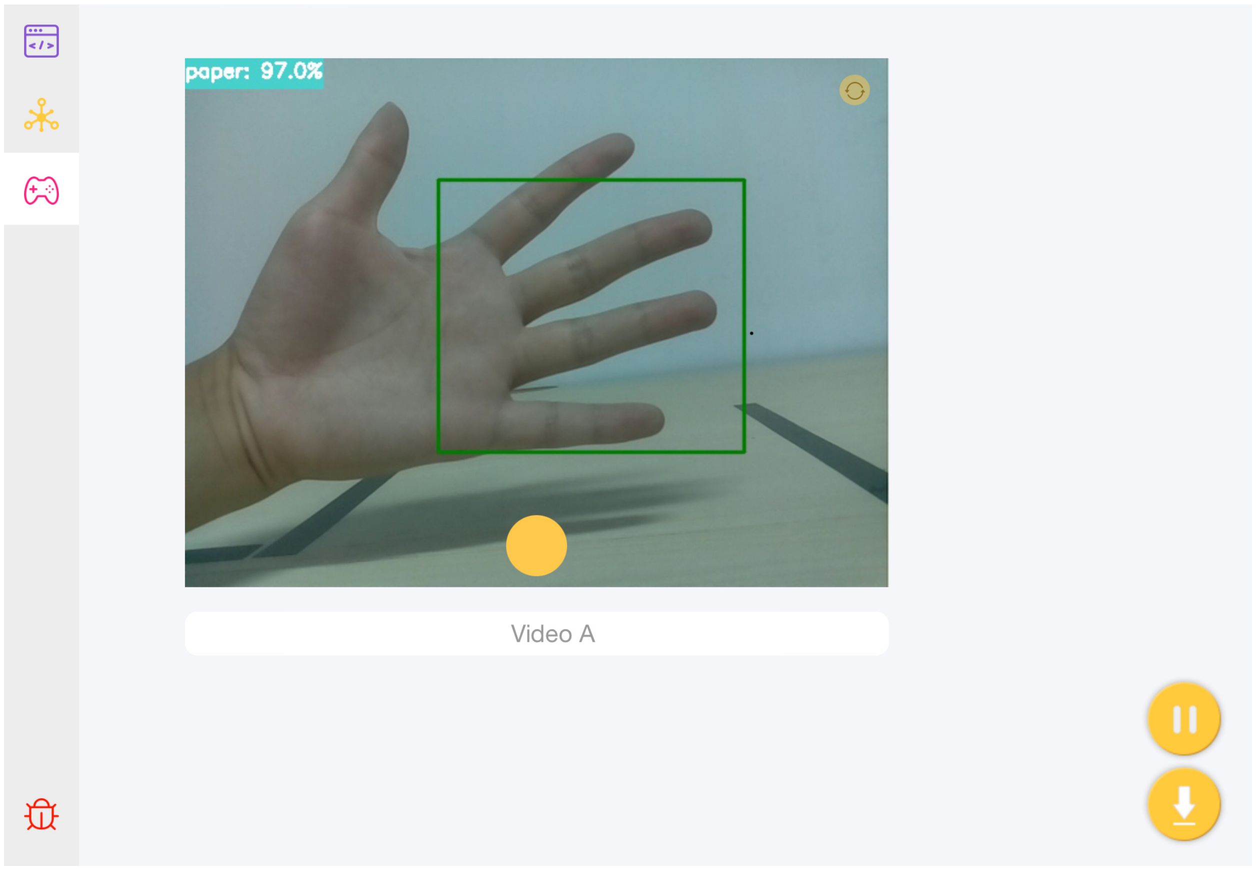
EXAMPLE



```
Start
    camera monitor  on
    turn traffic sign detection  on

Forever
    set sign to  width  of detected traffic sign
    if  sign = " stop "
    do  set servo  P1  angle to  45
        delay  1000
    else if  sign = " forward "
    do  set servo  P1  angle to  -45
        delay  1000
    else if  sign = " left "
    do  set servo  P0  angle to  90
        delay  1000
    else if  sign = " right "
    do  set servo  P0  angle to  -90
        delay  1000
    set servo  P0  angle to  0
    set servo  P1  angle to  0
```

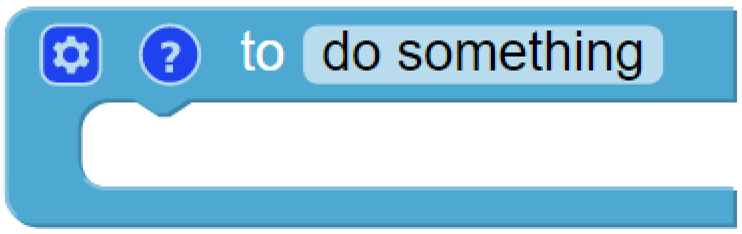
3.10 Play Rock-paper-scissor

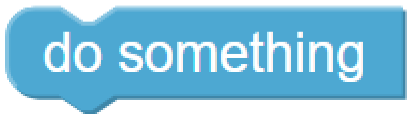
Let's play Rock-Paper-Scissor with Pan-Tilt HAT. Pan-Tilt HAT can judge who is the winner according to your gestures and print the result.



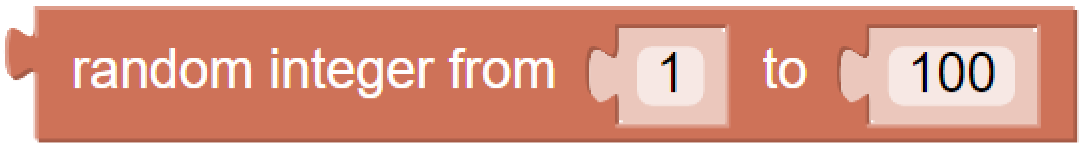
TIPS

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.

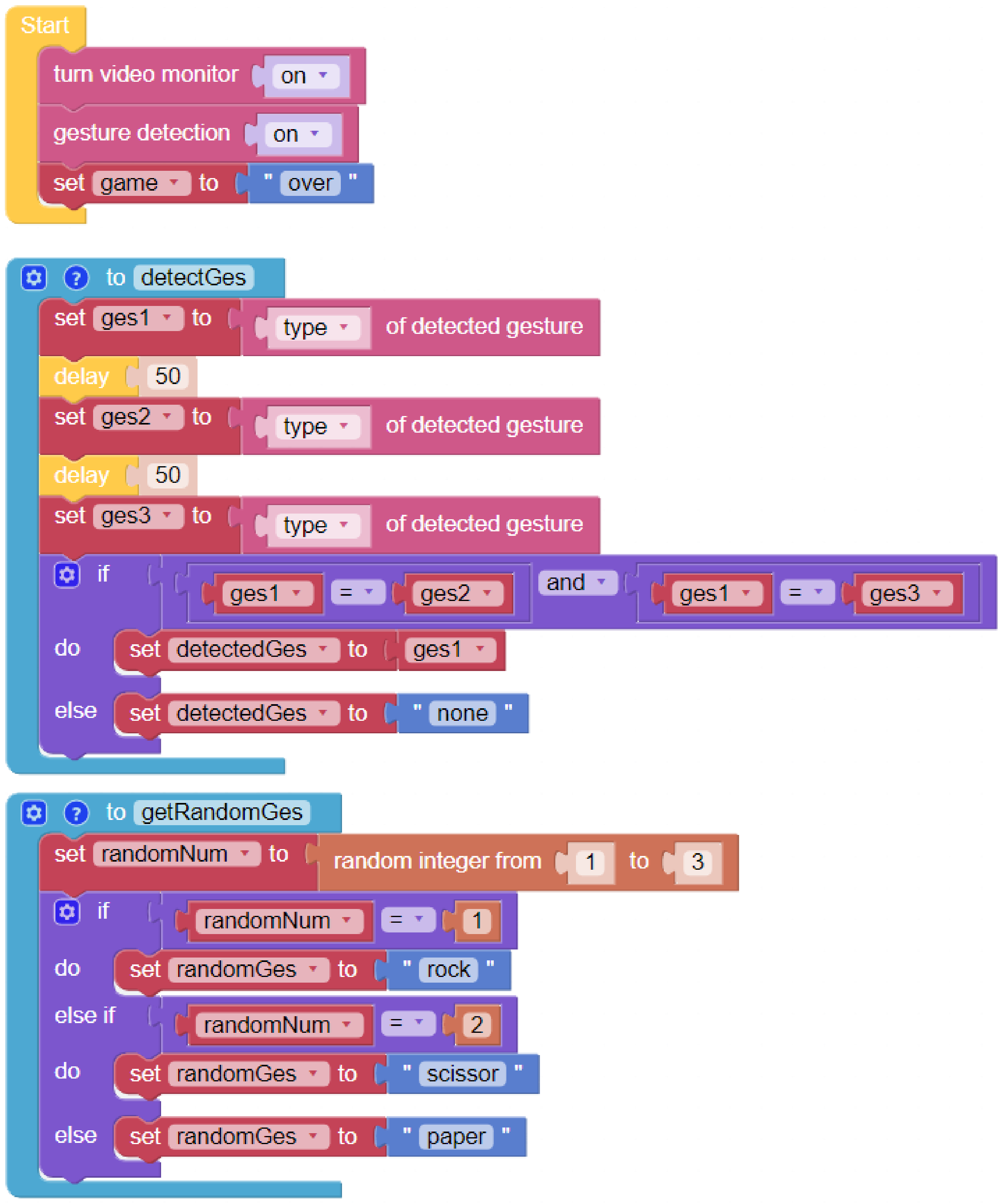




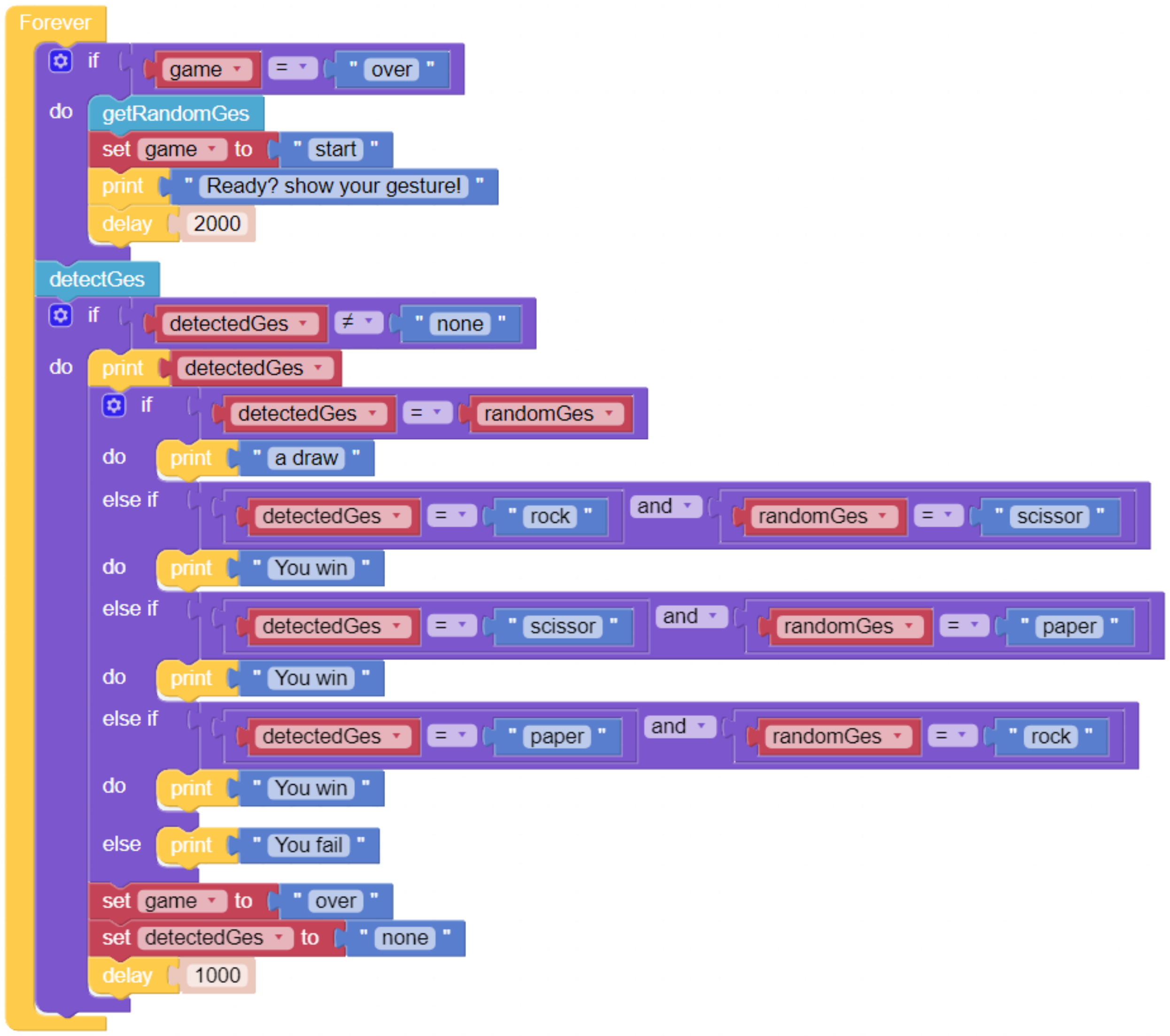
This block produces a random integer in a certain range and you can modify the upper and lower limits.



EXAMPLE



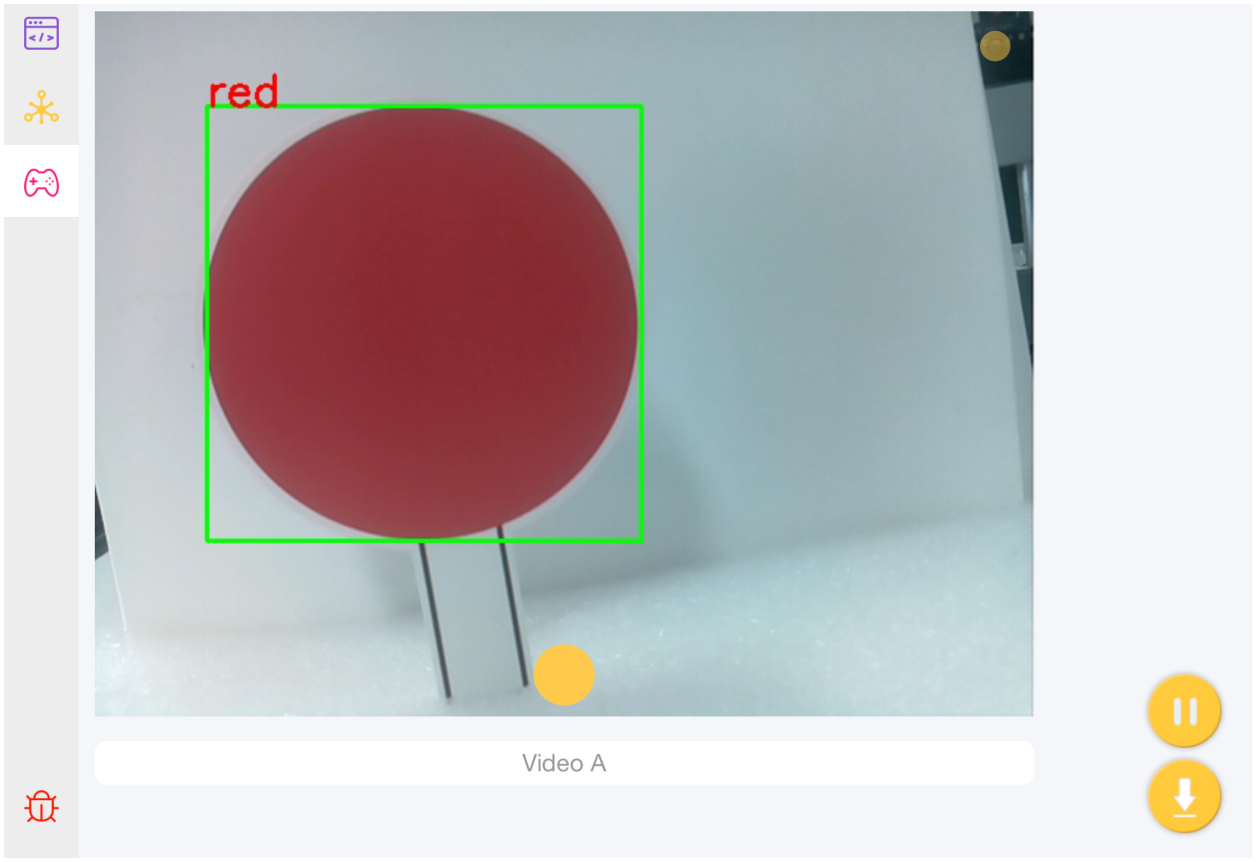
```
Start
    turn video monitor  on
    gesture detection  on
    set game to " over "

to detectGes
    set ges1 to  type  of detected gesture
    delay 50
    set ges2 to  type  of detected gesture
    delay 50
    set ges3 to  type  of detected gesture
    if   ges1 = ges2  and  ges1 = ges3
    do     set detectedGes to  ges1
    else   set detectedGes to " none "

to getRandomGes
    set randomNum to  random integer from 1 to 3
    if       randomNum = 1
    do       set randomGes to " rock "
    else if  randomNum = 2
    do       set randomGes to " scissor "
    else     set randomGes to " paper "
```



```
Forever
    if  game = "over"
    do  getRandomGes
        set game to "start"
        print "Ready? show your gesture!"
        delay 2000
    detectGes
    if  detectedGes ≠ "none"
    do  print detectedGes
        if  detectedGes = randomGes
        do  print "a draw"
        else if  detectedGes = "rock" and randomGes = "scissor"
        do  print "You win"
        else if  detectedGes = "scissor" and randomGes = "paper"
        do  print "You win"
        else if  detectedGes = "paper" and randomGes = "rock"
        do  print "You win"
        else  print "You fail"
        set game to "over"
        set detectedGes to "none"
        delay 1000
```

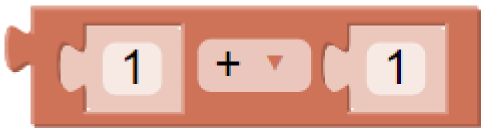
3.11 Color Tracking

Prepare a small red ball and place it directly in front of the camera of Pan-Tilt HAT. The camera can consistently track the ball. This project may be little harder than the previous projects.

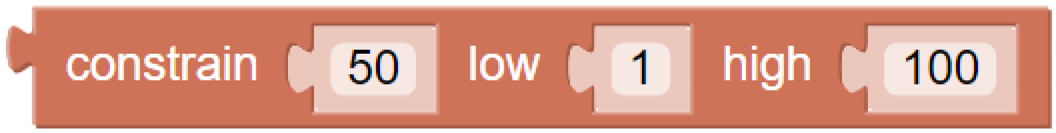


TIPS

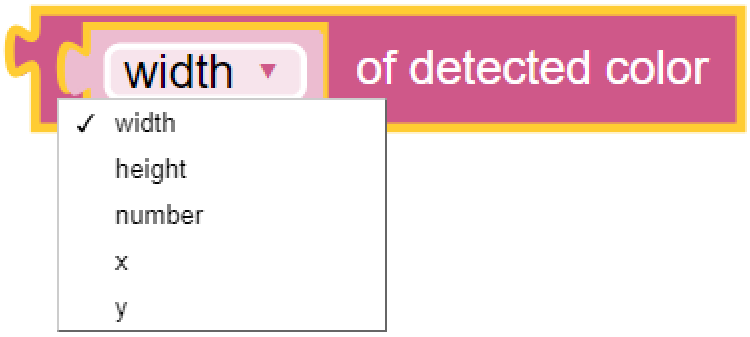
Mathematical operation block can perform “+ , - , x , ÷”.



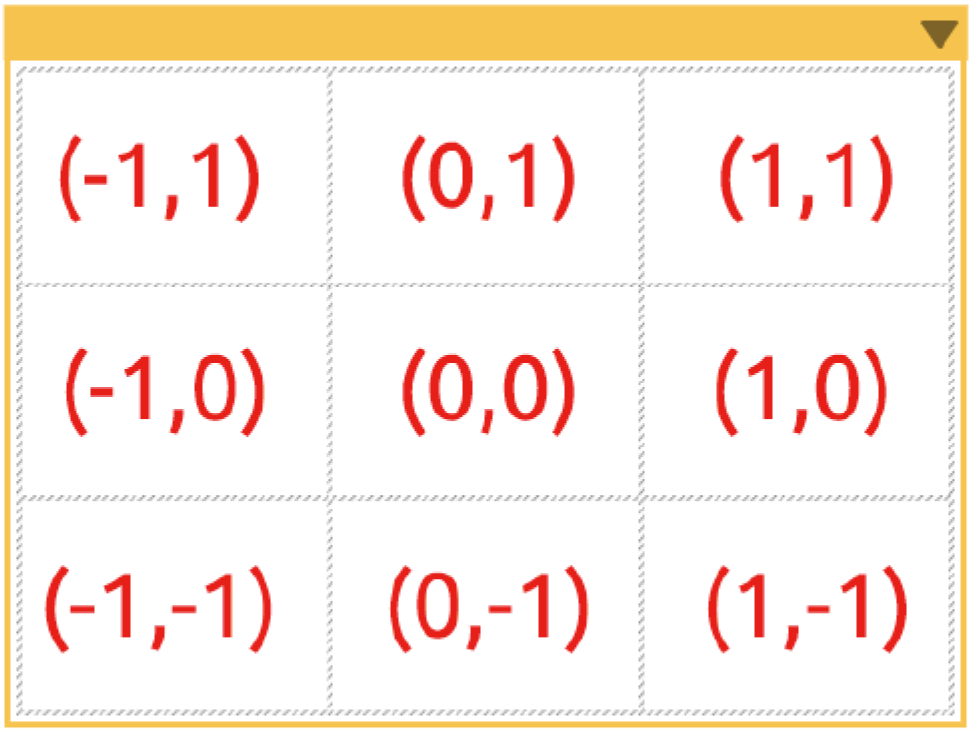
This block is often used together with variables to limit their ranges.



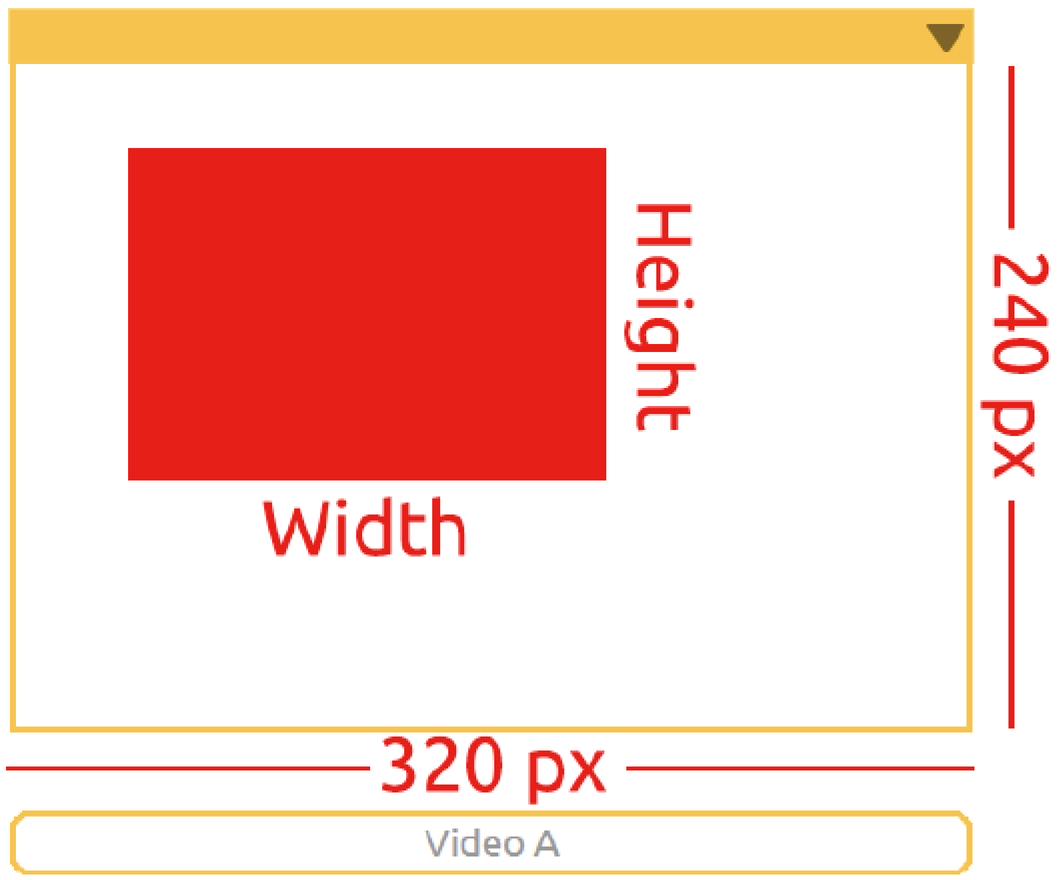
You can get the information of detected color through this block. Modify the drop-down menu options, and choose to read the coordinates, size or number.



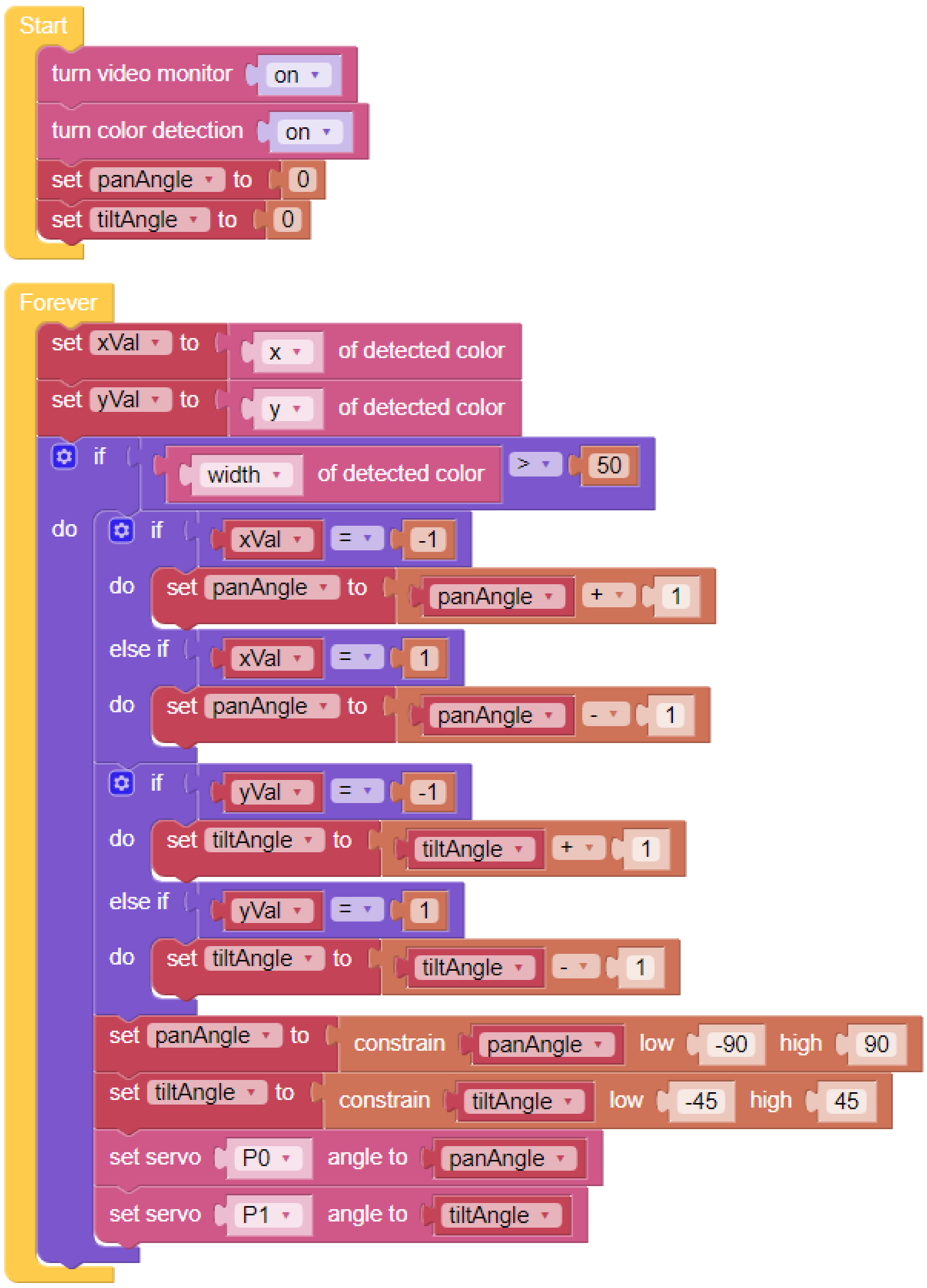
The “object detection” can output the detected coordinate value (x, y) based on the center point of the graphic. The screen is divided into a 3x3 grid, as shown on the left.



The “object detection” can detect the size (Width & Height) of the graphic.



EXAMPLE



```
Start
    turn video monitor  on
    turn color detection  on
    set panAngle to  0
    set tiltAngle to  0

Forever
    set xVal to  x of detected color
    set yVal to  y of detected color
    if  width of detected color  >  50
    do
        if  xVal  =  -1
        do  set panAngle to  panAngle  +  1
        else if  xVal  =  1
        do  set panAngle to  panAngle  -  1
        if  yVal  =  -1
        do  set tiltAngle to  tiltAngle  +  1
        else if  yVal  =  1
        do  set tiltAngle to  tiltAngle  -  1
        set panAngle to  constrain  panAngle  low  -90  high  90
        set tiltAngle to  constrain  tiltAngle  low  -45  high  45
        set servo  P0  angle to  panAngle
        set servo  P1  angle to  tiltAngle
```

CHAPTER

FOUR

APPENDIX

4.1 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.

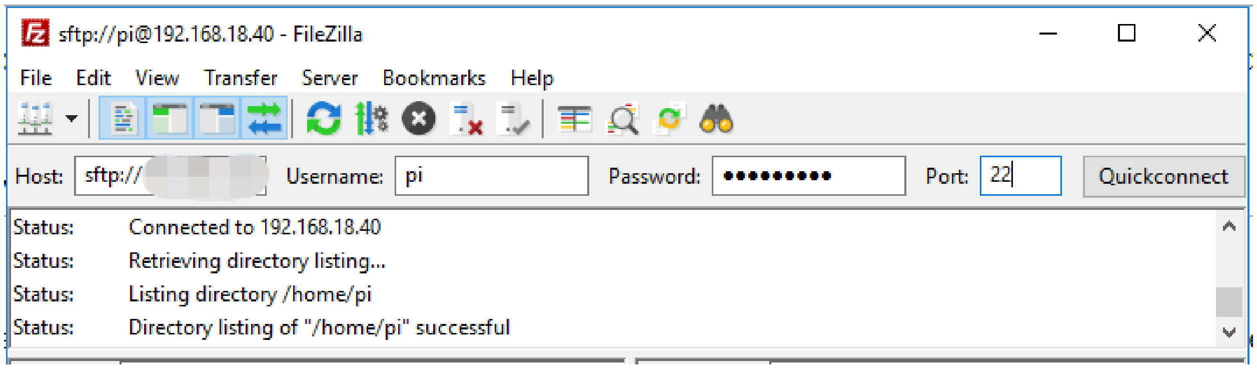
Filezilla is an open source software that not only supports FTP, but also FTP over TLS (FTPS) and SFTP. We can use Filezilla to upload local files (such as pictures and audio, etc.) to the Raspberry Pi, or download files from the Raspberry Pi to the local.

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

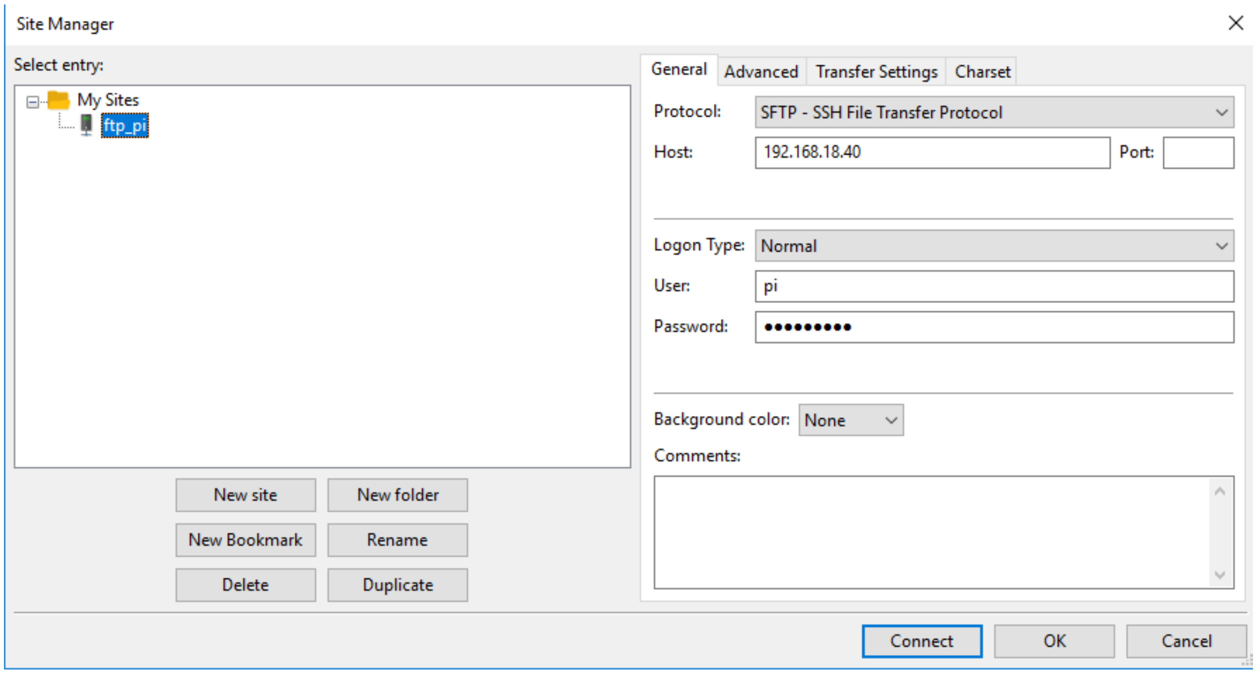
After a quick install open it up and now connect it to an FTP server. It has 3 ways to connect, here we use the **Quick Connect** bar. Enter the **hostname/IP**, **username**, **password** and **port (22)**, then click **Quick Connect** or press **Enter** to connect to the server.


sftp://pi@192.168.18.40 - FileZilla
File Edit View Transfer Server Bookmarks Help
Quickconnect

Host: sftp://
Username: pi
Password: •••••••••
Port: 22

```
Status:     Connected to 192.168.18.40
Status:     Retrieving directory listing...
Status:     Listing directory /home/pi
Status:     Directory listing of "/home/pi" successful
```

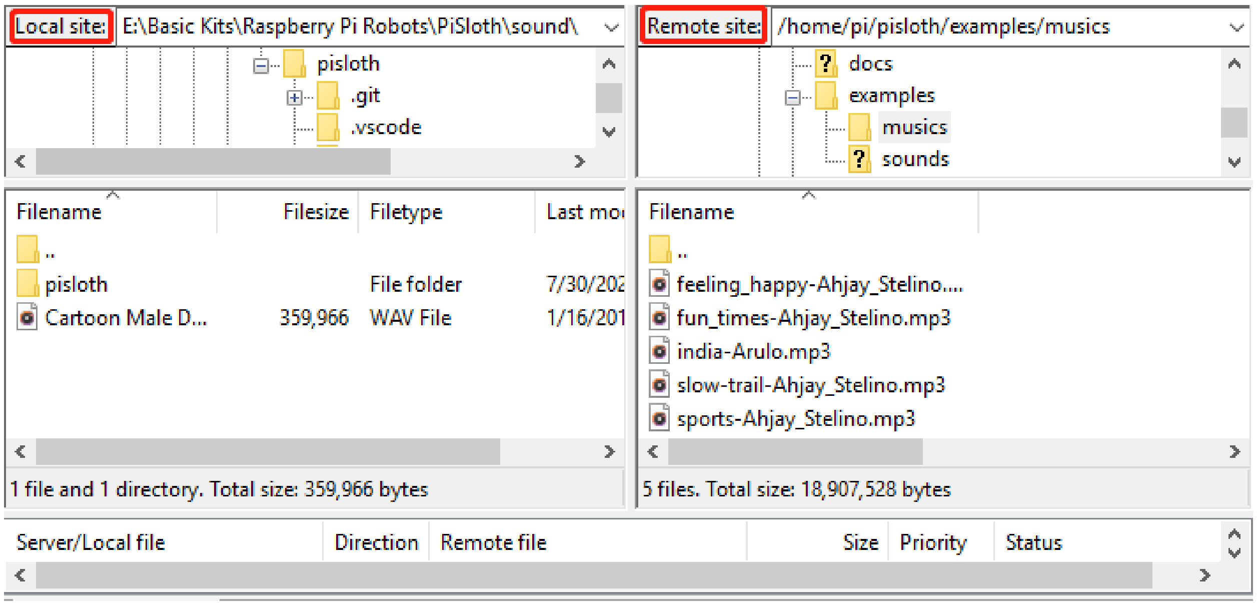
Note: Quick Connect is a good way to test your login information. If you want to create a permanent entry, you can select **File**-> **Copy Current Connection to Site Manager** after a successful Quick Connect, enter the name and click **OK**. Next time you will be able to connect by selecting the previously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



Protocol: SFTP - SSH File Transfer Protocol
Host: 192.168.18.40
Port:
Logon Type: Normal
User: pi
Password: •••••••••
Background color: None

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.



```
Filename           Filesize  Filetype     Last mo
..
pisloth                      File folder  7/30/202
Cartoon Male D...  359,966   WAV File     1/16/201
1 file and 1 directory. Total size: 359,966 bytes
```

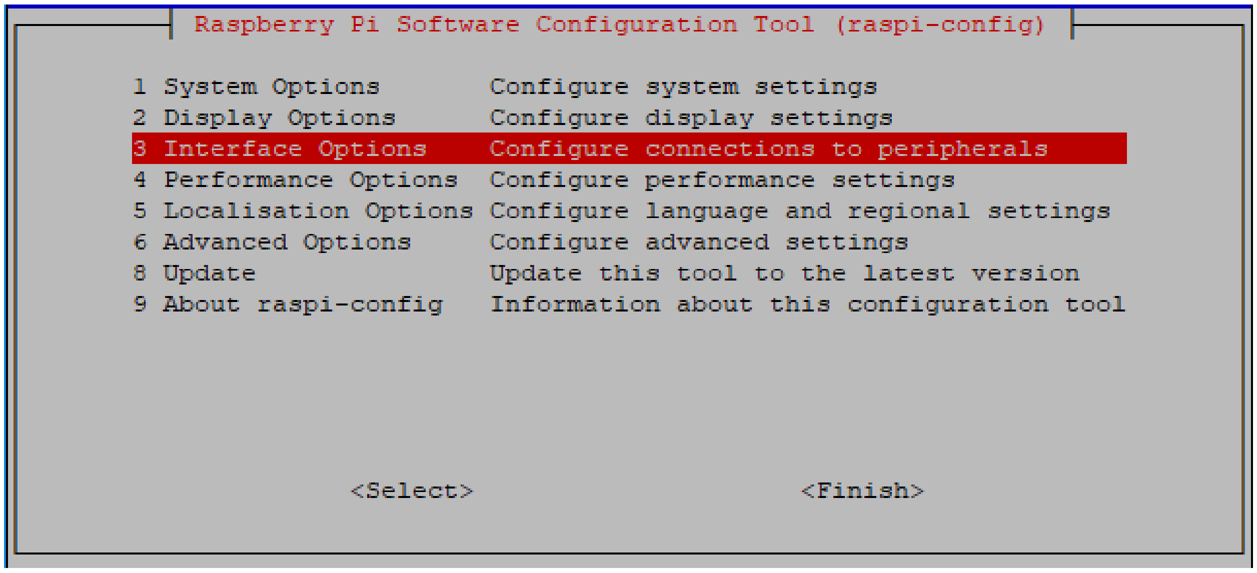
```
Filename
..
feeling_happy-Ahjay_Stelino....
fun_times-Ahjay_Stelino.mp3
india-Arulo.mp3
slow-trail-Ahjay_Stelino.mp3
sports-Ahjay_Stelino.mp3
5 files. Total size: 18,907,528 bytes
```

4.2 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).

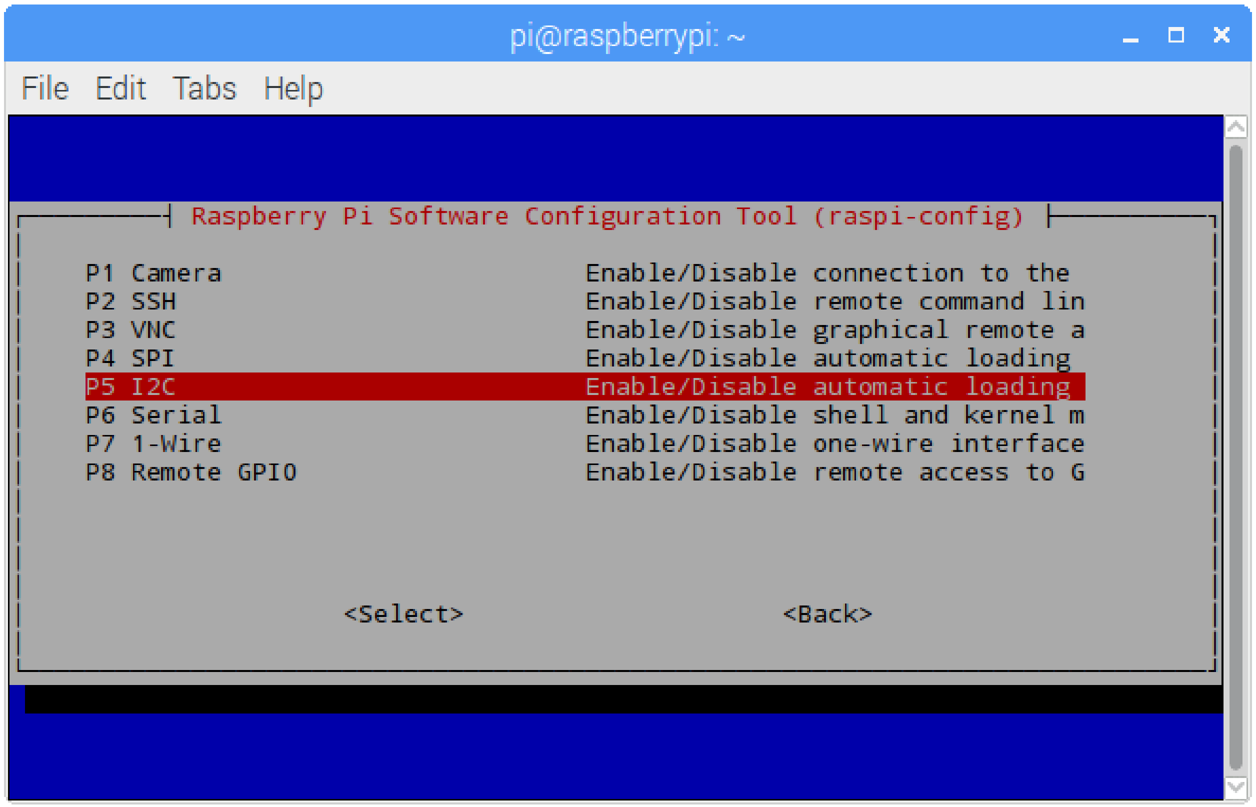
```
sudo raspi-config
```

3 Interfacing options



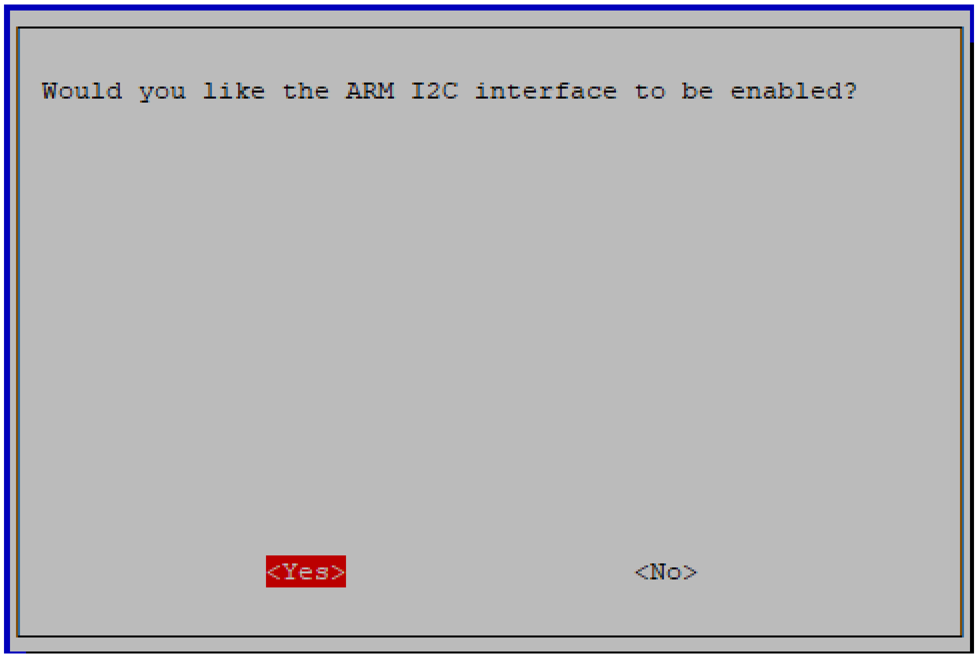
```
Raspberry Pi Software Configuration Tool (raspi-config)

1 System Options        Configure system settings
2 Display Options       Configure display settings
3 Interface Options     Configure connections to peripherals
4 Performance Options   Configure performance settings
5 Localisation Options  Configure language and regional settings
6 Advanced Options      Configure advanced settings
8 Update                Update this tool to the latest version
9 About raspi-config    Information about this configuration tool

            <Select>                     <Finish>
```

P5 I2C



```
pi@raspberrypi: ~
File  Edit  Tabs  Help

┌──────────┤ Raspberry Pi Software Configuration Tool (raspi-config) ├──────────┐
│                                                                                │
│    P1 Camera                          Enable/Disable connection to the         │
│    P2 SSH                             Enable/Disable remote command lin        │
│    P3 VNC                             Enable/Disable graphical remote a        │
│    P4 SPI                             Enable/Disable automatic loading         │
│    P5 I2C                             Enable/Disable automatic loading         │
│    P6 Serial                          Enable/Disable shell and kernel m        │
│    P7 1-Wire                          Enable/Disable one-wire interface        │
│    P8 Remote GPIO                     Enable/Disable remote access to G        │
│                                                                                │
│                  <Select>                     <Back>                           │
└────────────────────────────────────────────────────────────────────────────────┘
```

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



```
Would you like the ARM I2C interface to be enabled?

               <Yes>                    <No>
```

4.3 Remote Desktop

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

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

4.3.1 VNC

You can use the function of remote desktop through VNC.

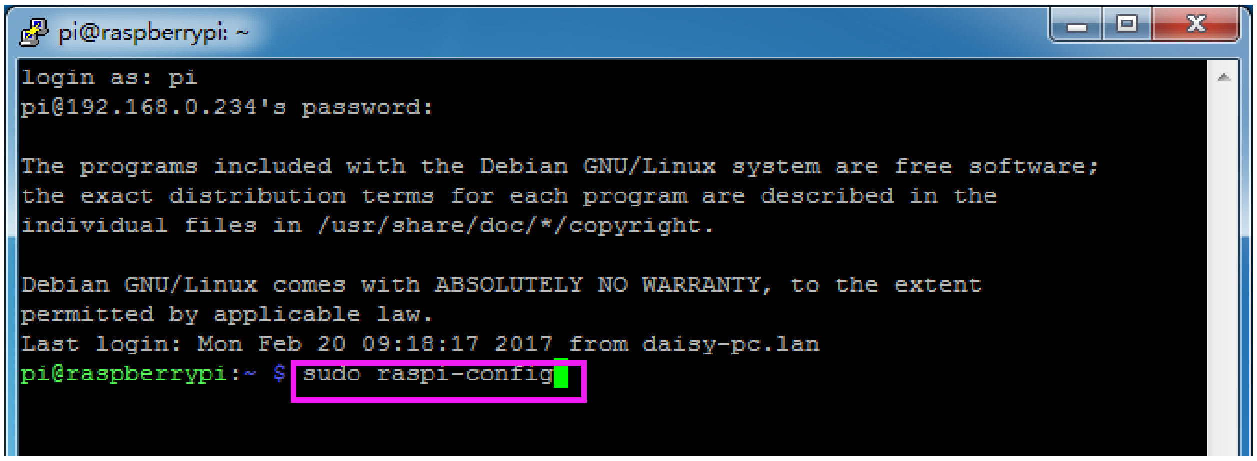
Enable VNC service

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:

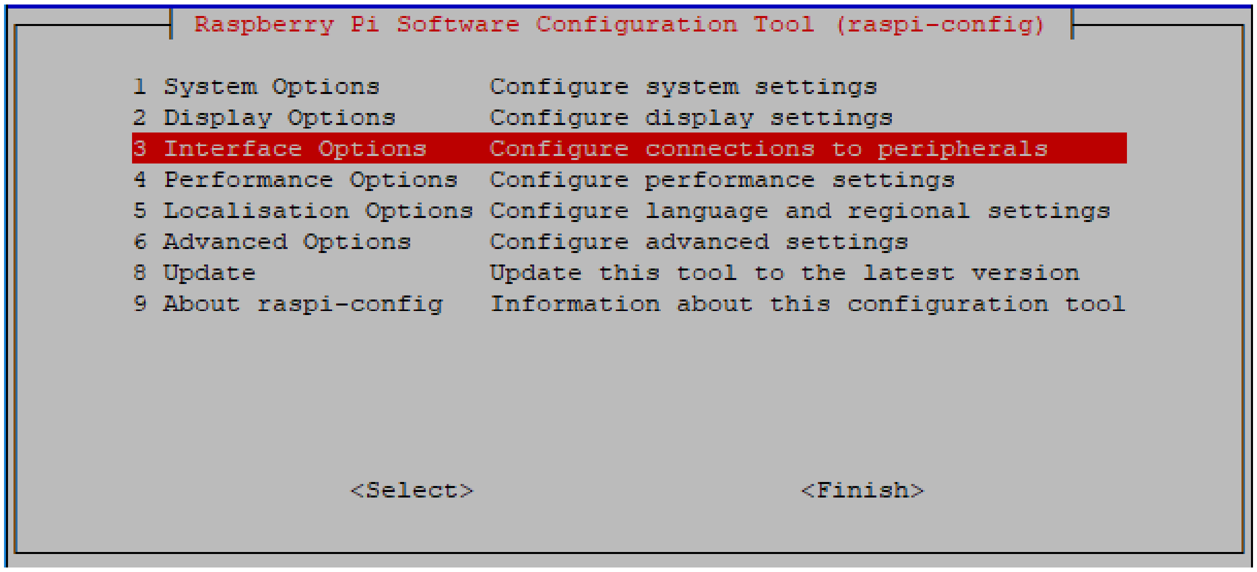
```
sudo raspi-config
```



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

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: Mon Feb 20 09:18:17 2017 from daisy-pc.lan
pi@raspberrypi:~ $ sudo raspi-config
```

Step 2

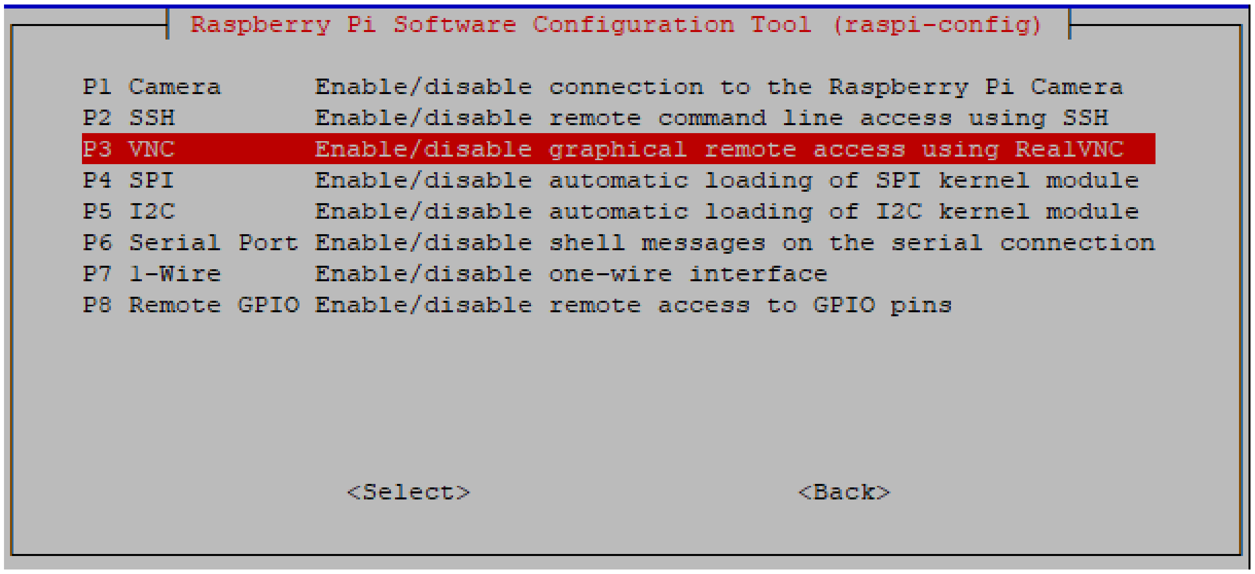
Choose **3 Interfacing Options** by press the down arrow key on your keyboard, then press the **Enter** key.



```
Raspberry Pi Software Configuration Tool (raspi-config)

 1 System Options        Configure system settings
 2 Display Options       Configure display settings
 3 Interface Options     Configure connections to peripherals
 4 Performance Options   Configure performance settings
 5 Localisation Options  Configure language and regional settings
 6 Advanced Options      Configure advanced settings
 8 Update                Update this tool to the latest version
 9 About raspi-config    Information about this configuration tool

          <Select>                     <Finish>
```

Step 3

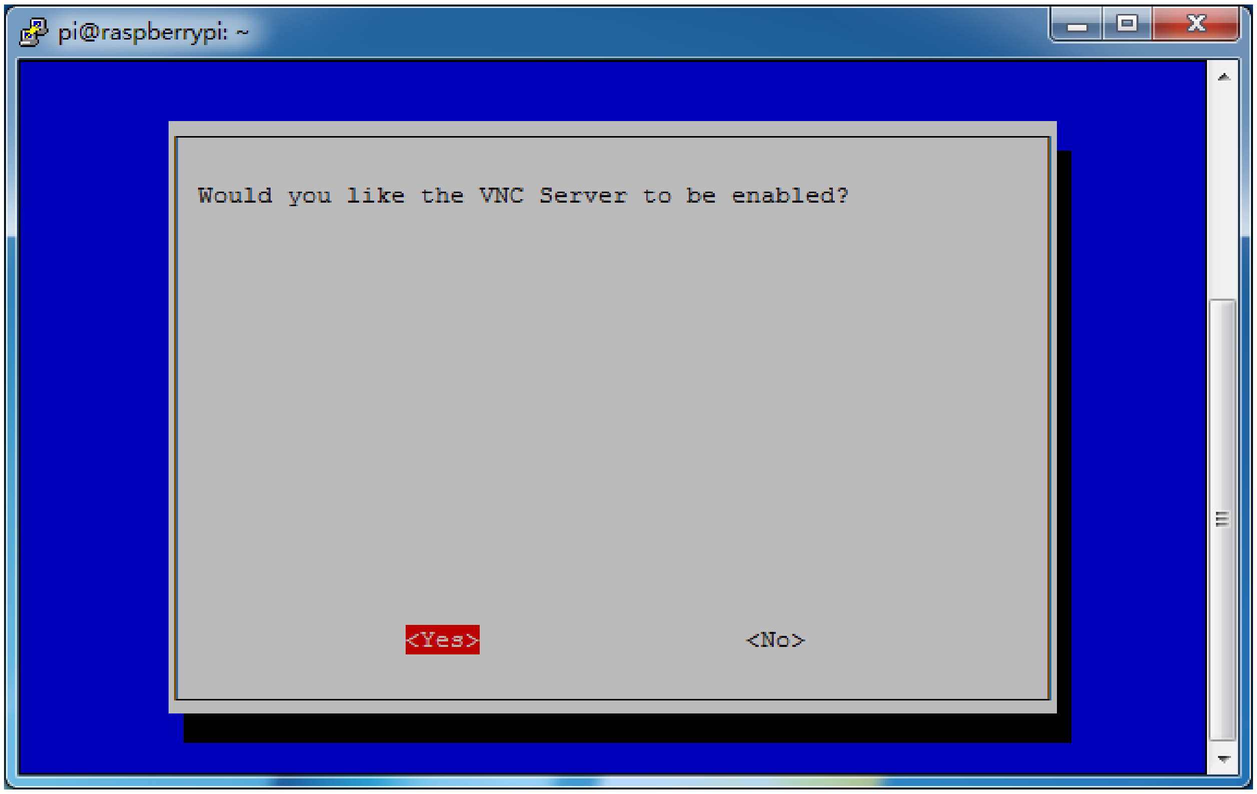
P3 VNC



```
Raspberry Pi Software Configuration Tool (raspi-config)

P1 Camera      Enable/disable connection to the Raspberry Pi Camera
P2 SSH         Enable/disable remote command line access using SSH
P3 VNC         Enable/disable graphical remote access using RealVNC
P4 SPI         Enable/disable automatic loading of SPI kernel module
P5 I2C         Enable/disable automatic loading of I2C kernel module
P6 Serial Port Enable/disable shell messages on the serial connection
P7 1-Wire      Enable/disable one-wire interface
P8 Remote GPIO Enable/disable remote access to GPIO pins

                 <Select>                     <Back>
```

Step 4

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



```
pi@raspberrypi: ~

Would you like the VNC Server to be enabled?

          <Yes>                 <No>
```

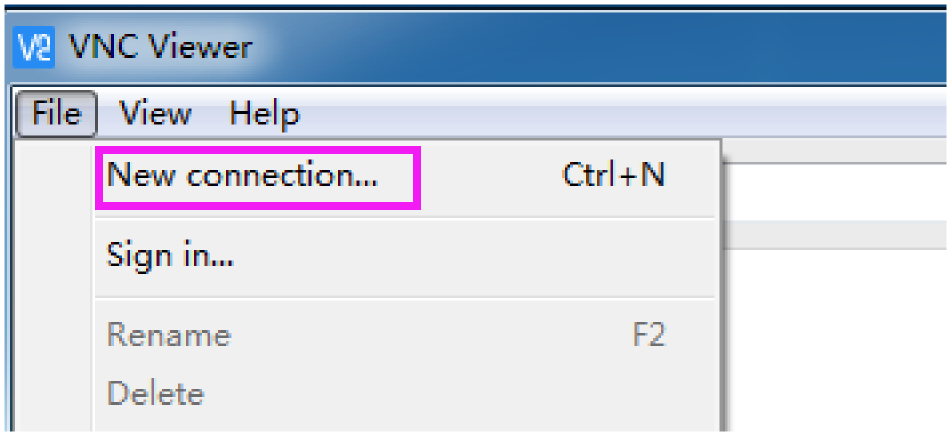
Login to VNC

Step 1

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

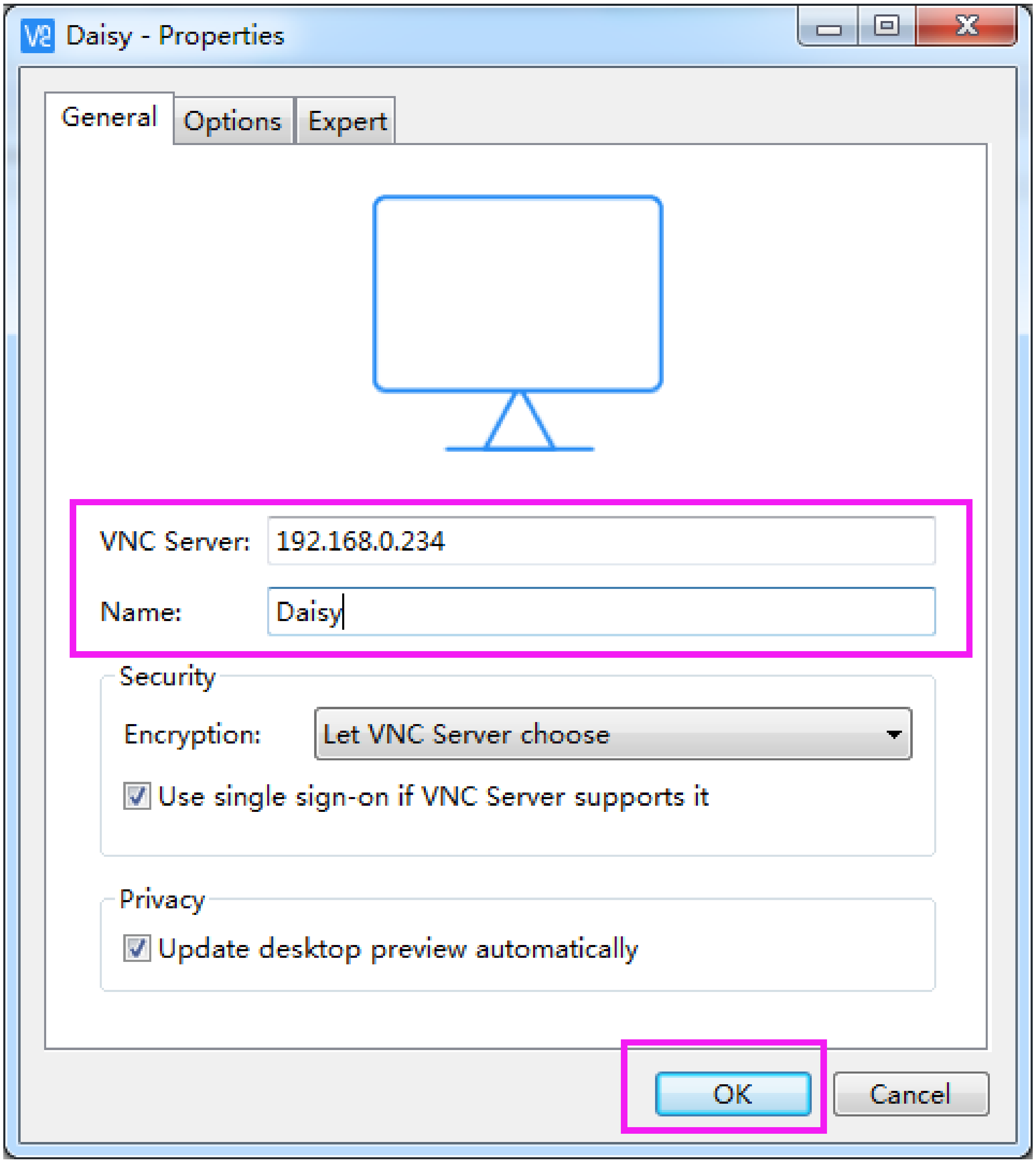
Step 2

Then select "**New connection**".



Step 3

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



VNC Server: 192.168.0.234

Name: Daisy

Security

Encryption: Let VNC Server choose

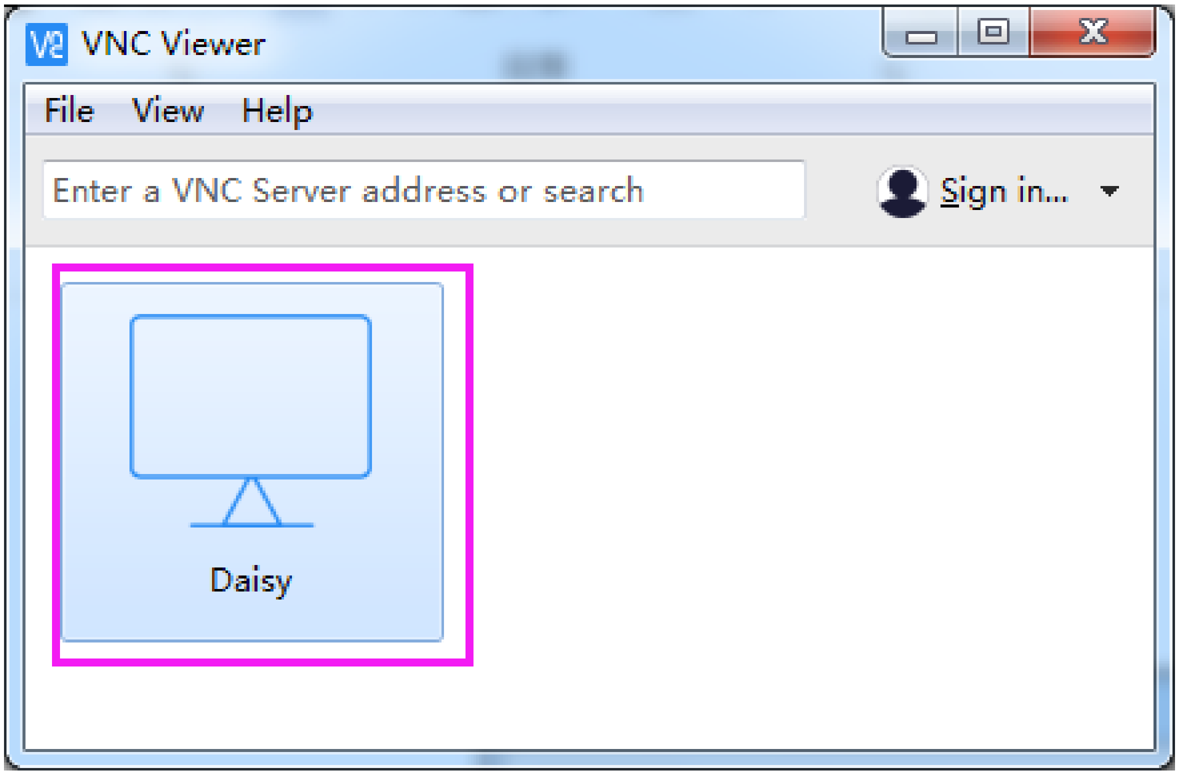
☑ Use single sign-on if VNC Server supports it

Privacy

☑ Update desktop preview automatically

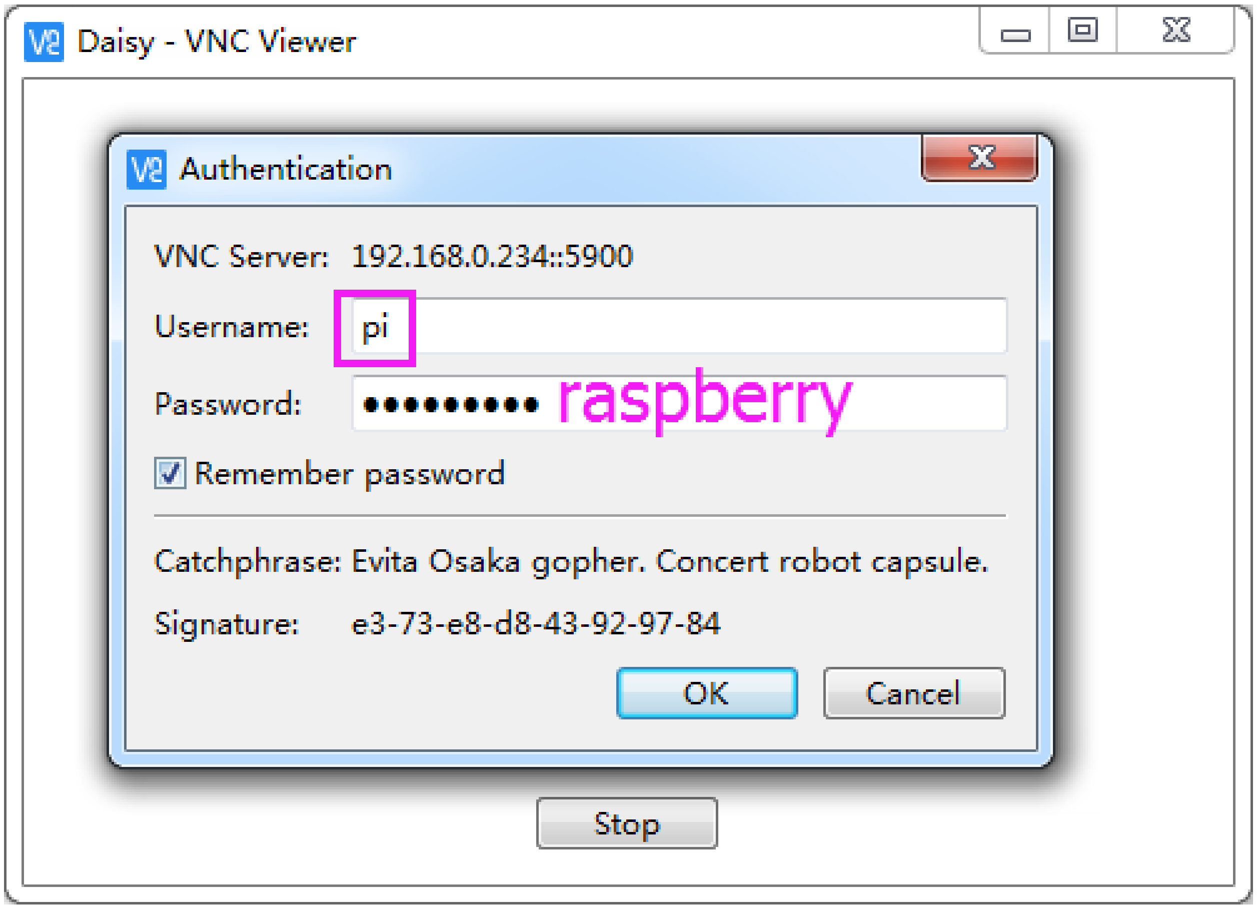
Step 4

Double click the **connection** just created:



Step 5

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



VNC Server: 192.168.0.234::5900

Username: pi

Password: •••••••••• raspberry

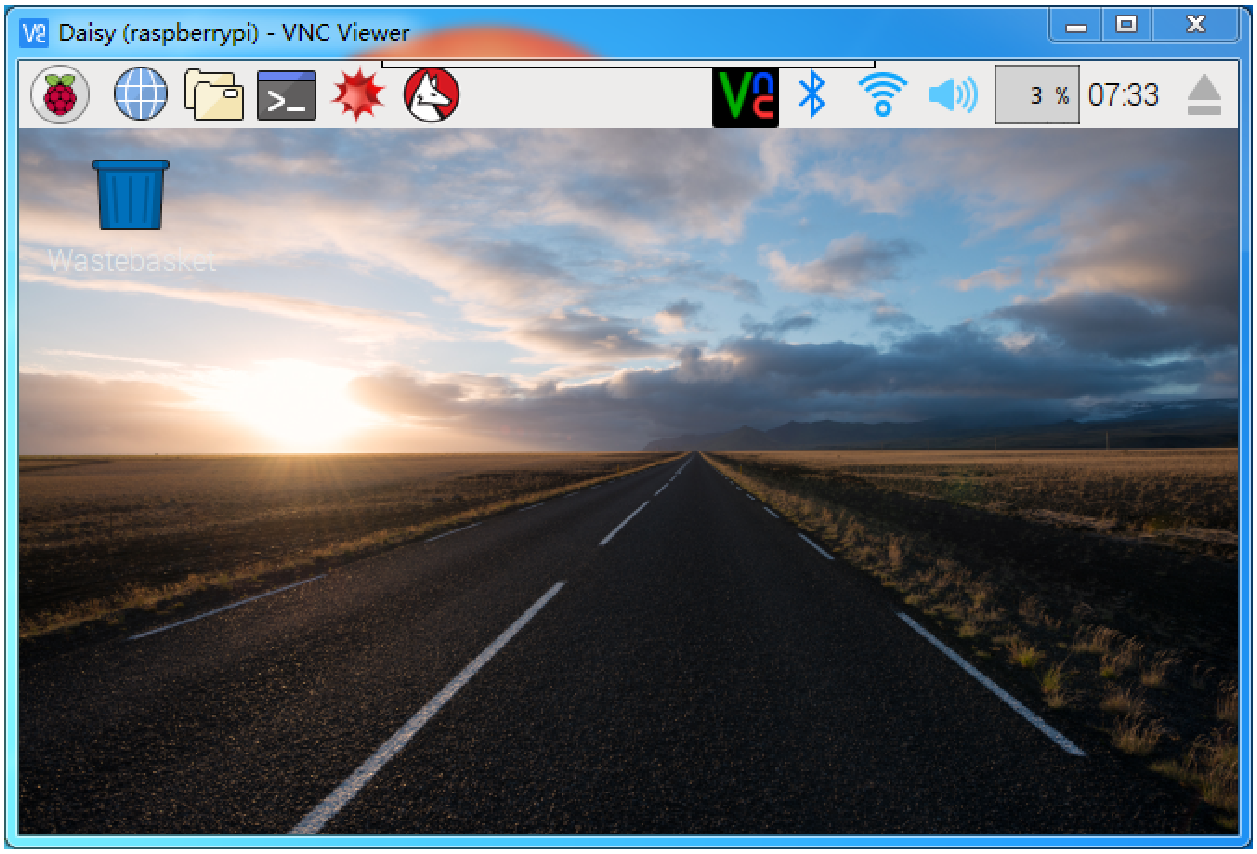
☑ Remember password

Catchphrase: Evita Osaka gopher. Concert robot capsule.

Signature: e3-73-e8-d8-43-92-97-84

Step 6

Now you can see the desktop of the Raspberry Pi:



That's the end of the VNC part.

4.3.2 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

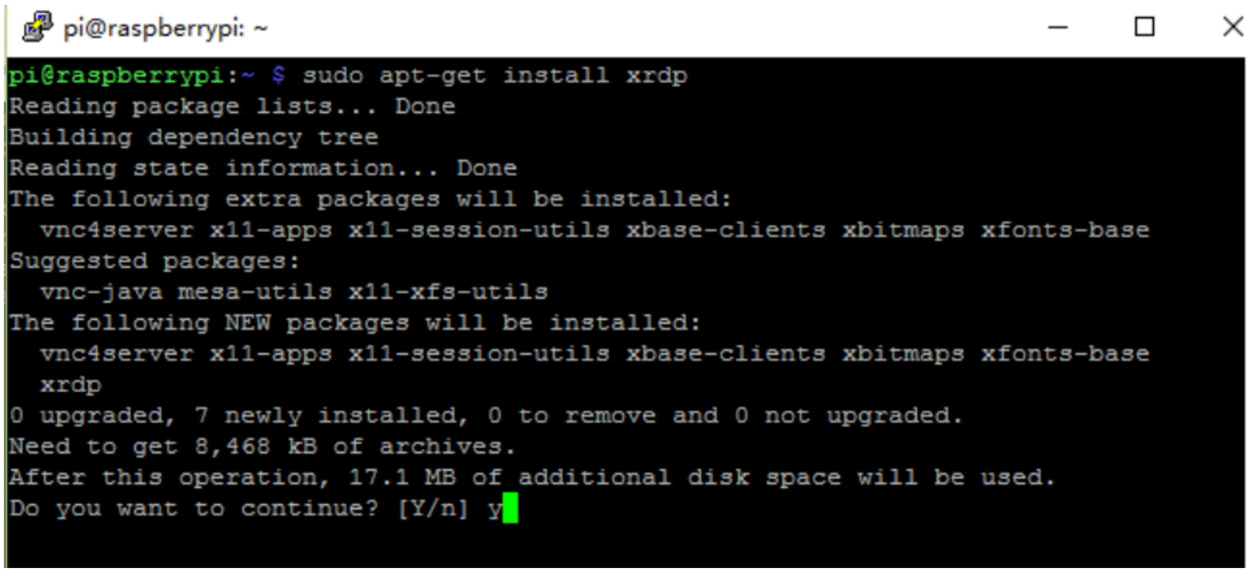
Input the following instructions to install XRDP.

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

Step 3

Later, the installation starts.

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



```
pi@raspberrypi:~ $ 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.

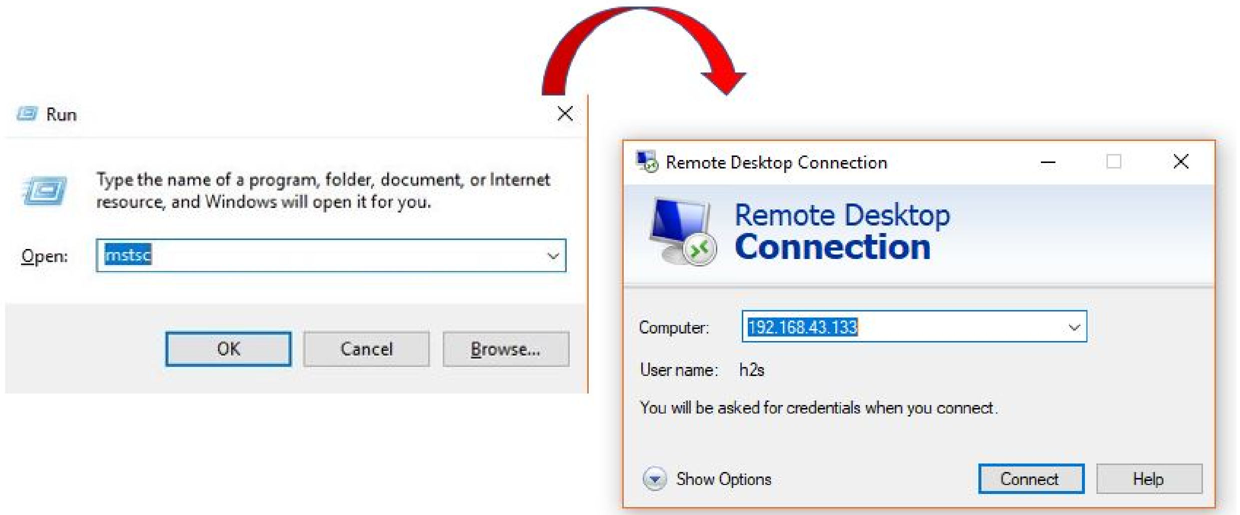
Login to XRDP

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".



Run

Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.

Open: mstsc

Remote Desktop Connection

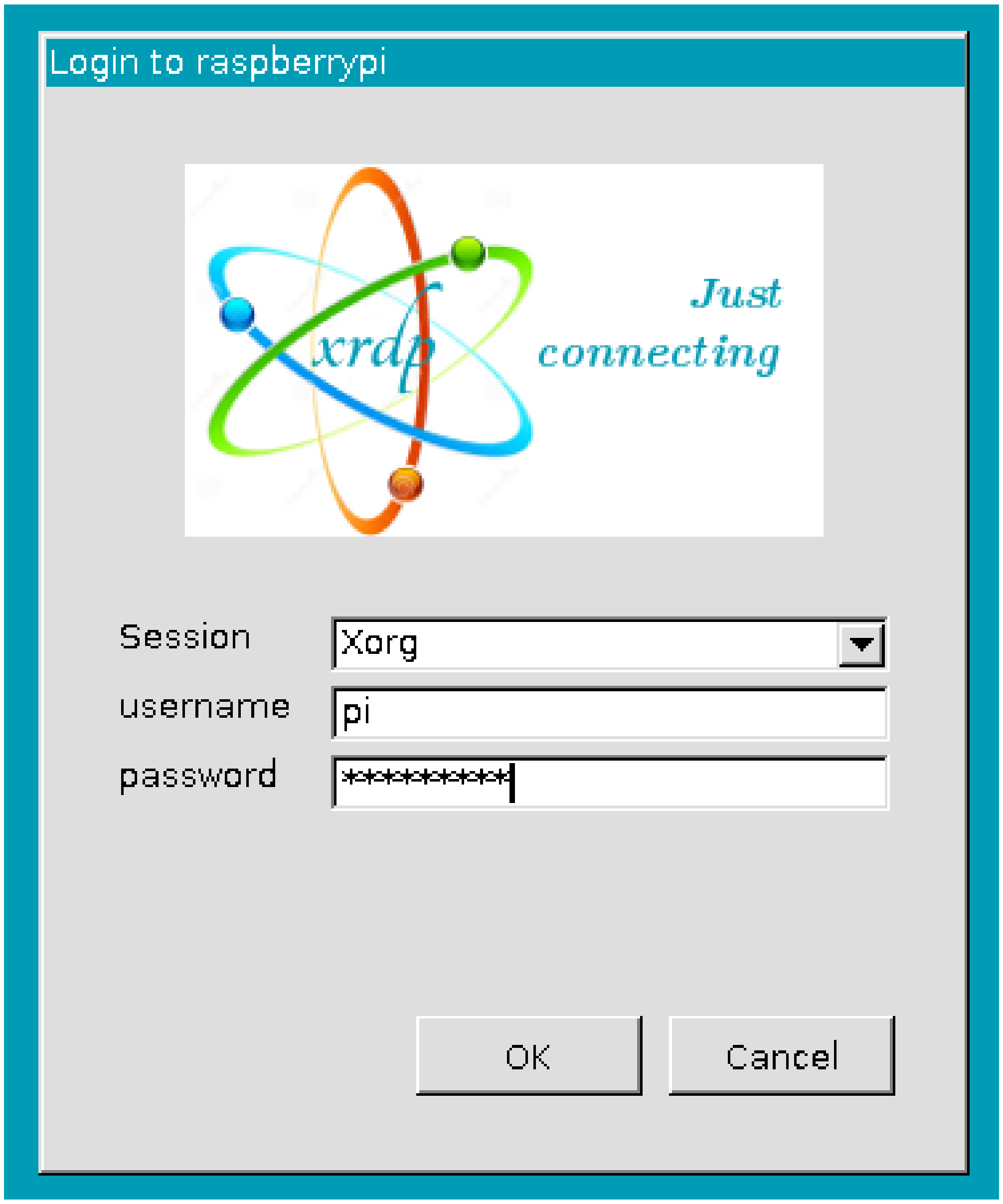
Computer: 192.168.43.133

User name: h2s

You will be asked for credentials when you connect.

Step 3

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


Login to raspberrypi

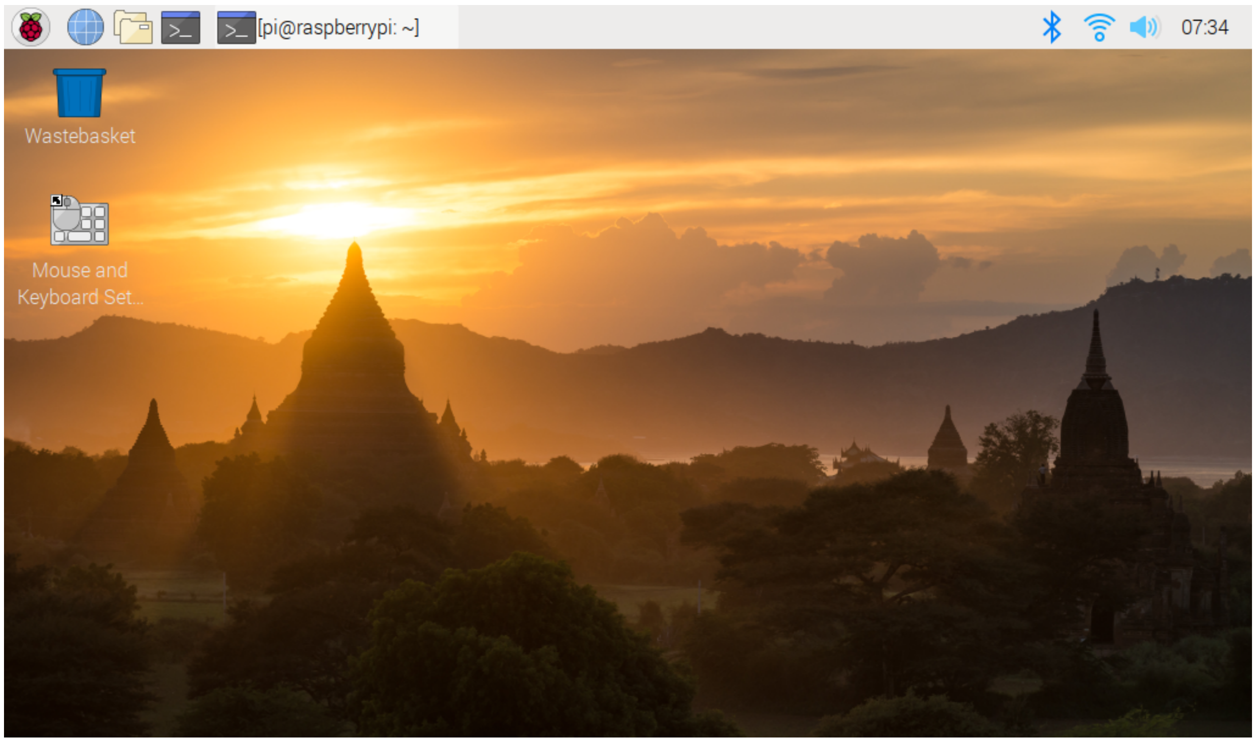
Session Xorg

username pi

password **********

Step 4

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


[pi@raspberrypi: ~]
07:34
Wastebasket
Mouse and Keyboard Set...

CHAPTER

FIVE

COPYRIGHT NOTICE