SunFounder Pan-tilt HAT Kit

www.sunfounder.com

Jul 12, 2023

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.

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.

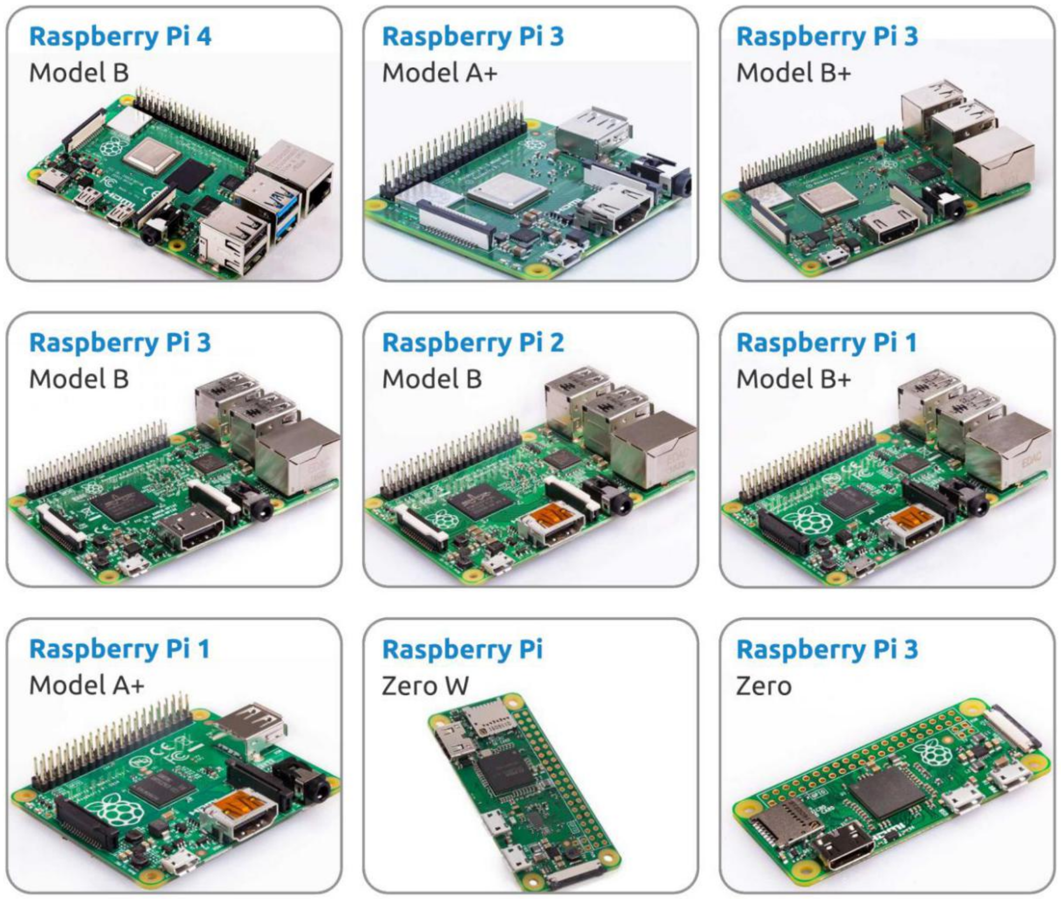
2.1.1 What Do We Need?

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

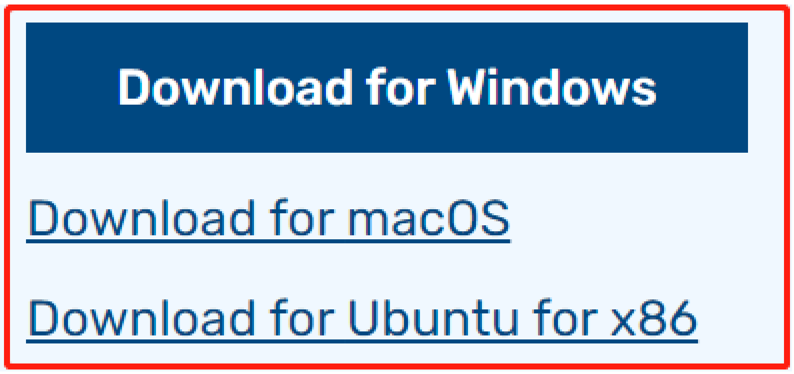
Required Components

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.

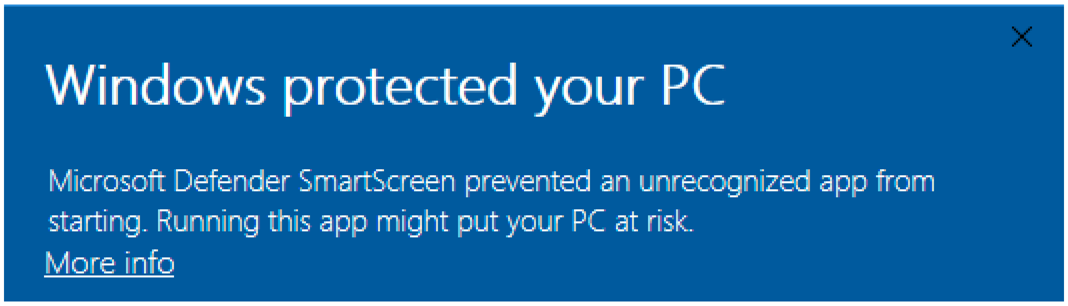
Visit the download page: https://www.raspberrypi.org/software/. Click on the link for the Raspberry Pi Imager that matches your operating system, when the download finishes, click it to launch the installer.



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

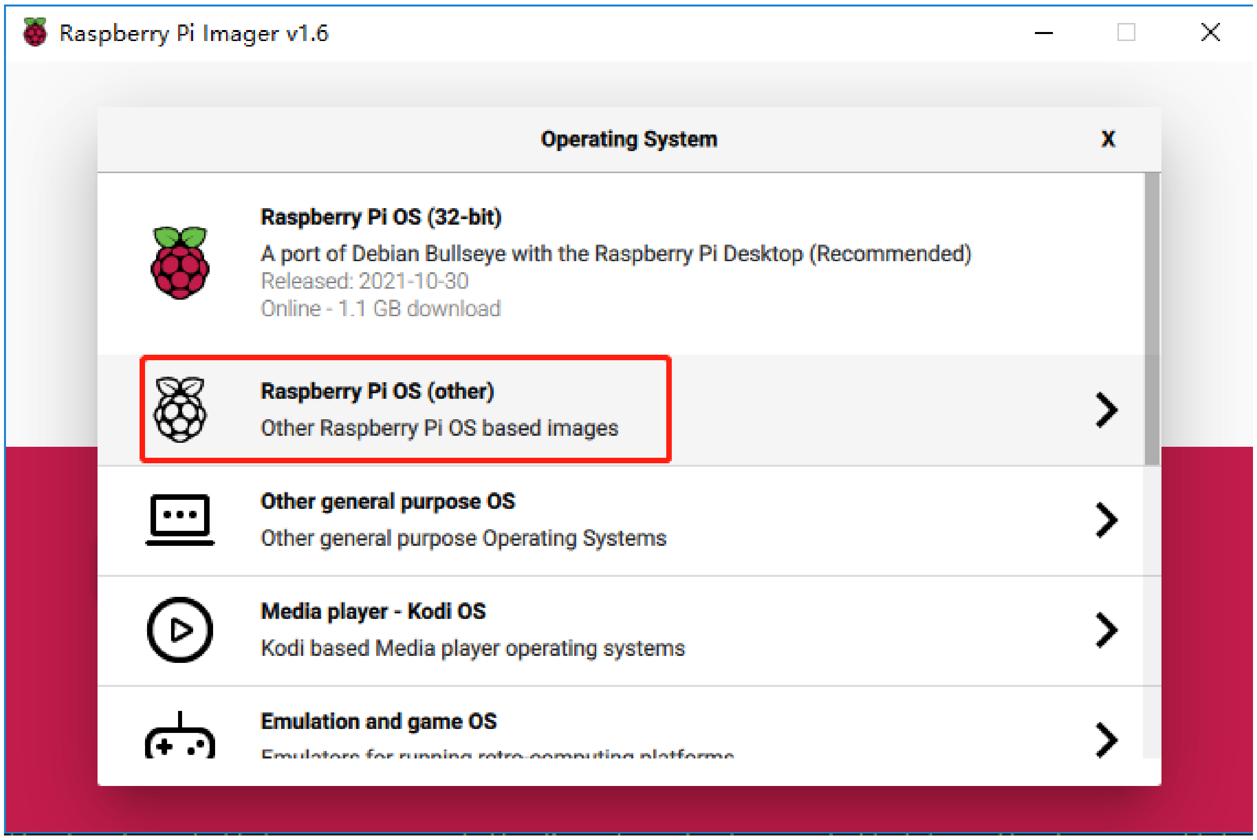
Step 3

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

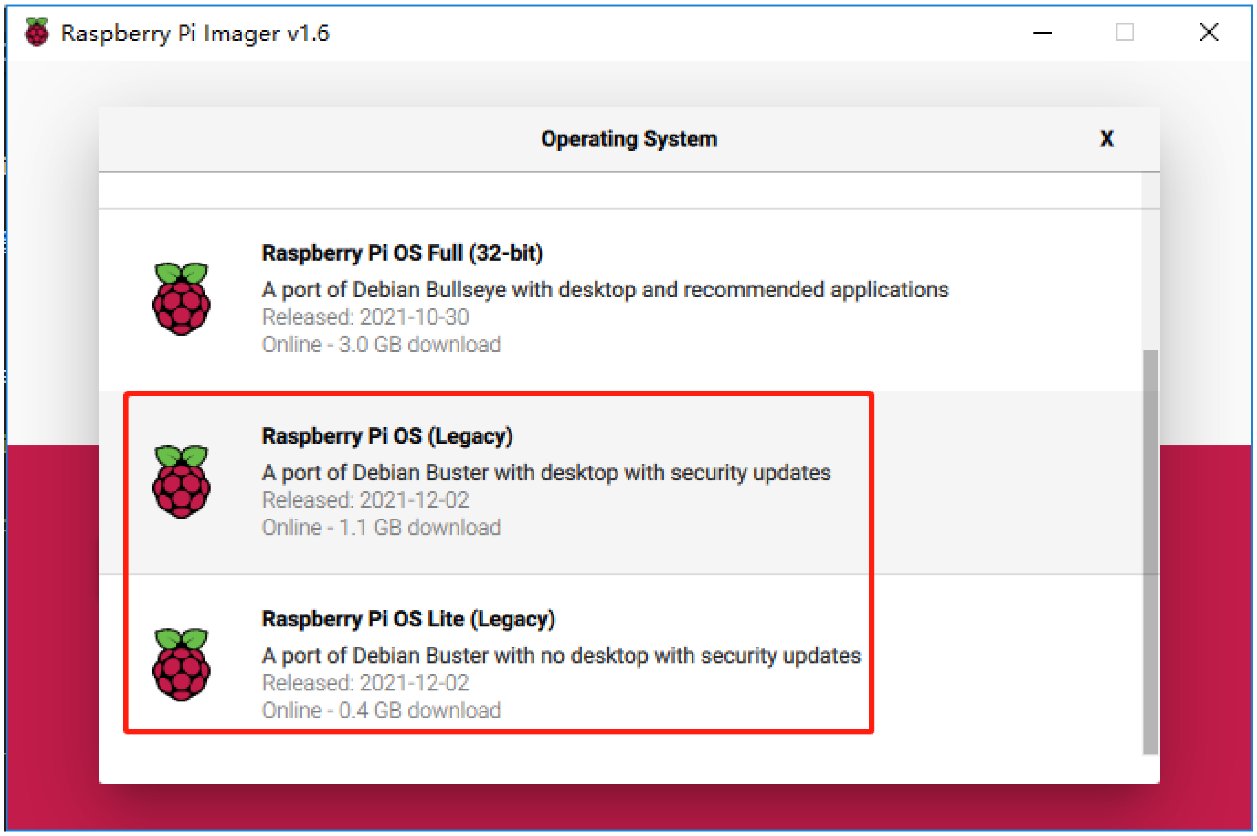
Step 4

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

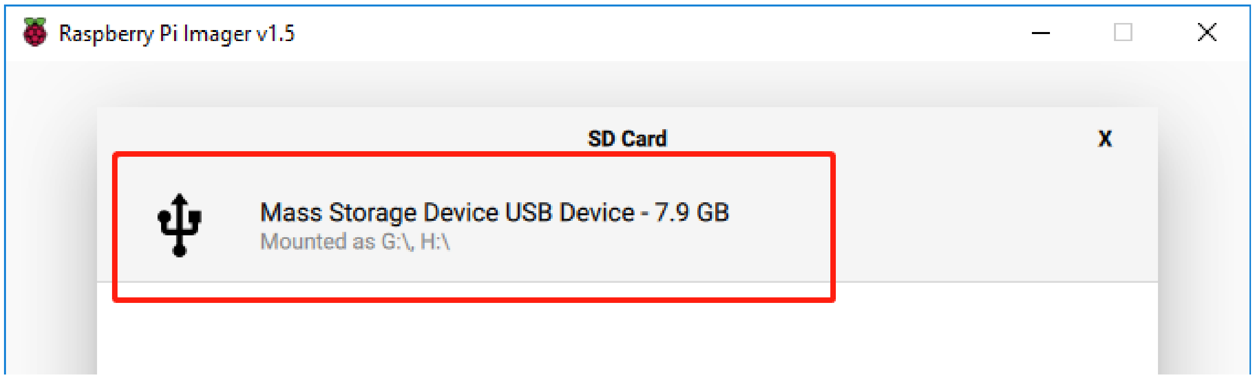
Raspberry Pi OS Full (32-bit)
A port of Debian Bullseye with desktop and recommended applications
Released: 2021-10-30
Online - 3.0 GB download

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

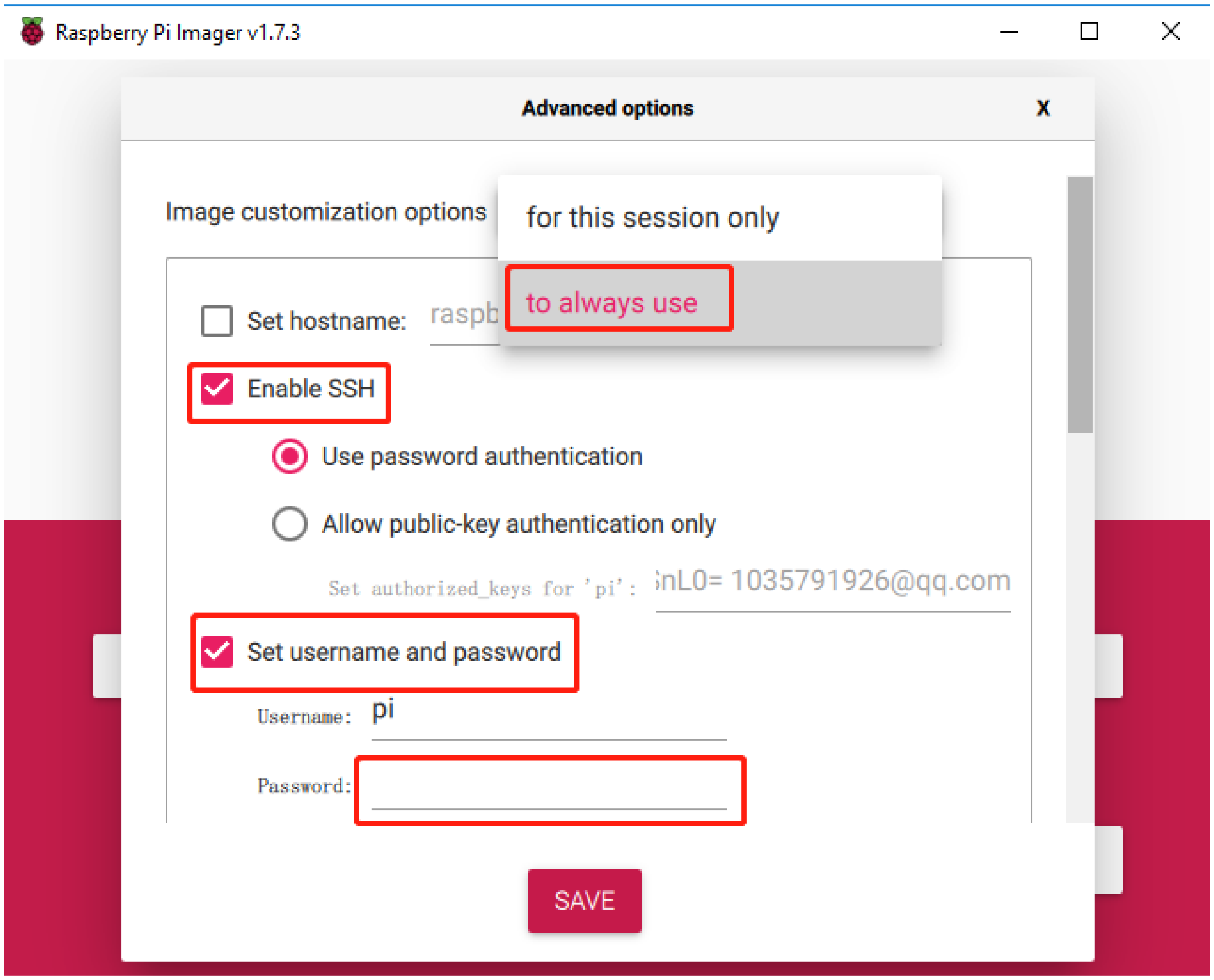
SD Card

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

Step 6

To open the advanced options page, click the setting button (appears after selecting operating system) or press Ctrl+Shift+X. Enable ssh and set the username and name. You can choose to always use this image customization options.

Note: When the Set hostname box is not checked, the default hostname will still be `raspberrypi`, and we will use this hostname to access the Raspberry Pi remotely.


Raspberry Pi Imager v1.7.3

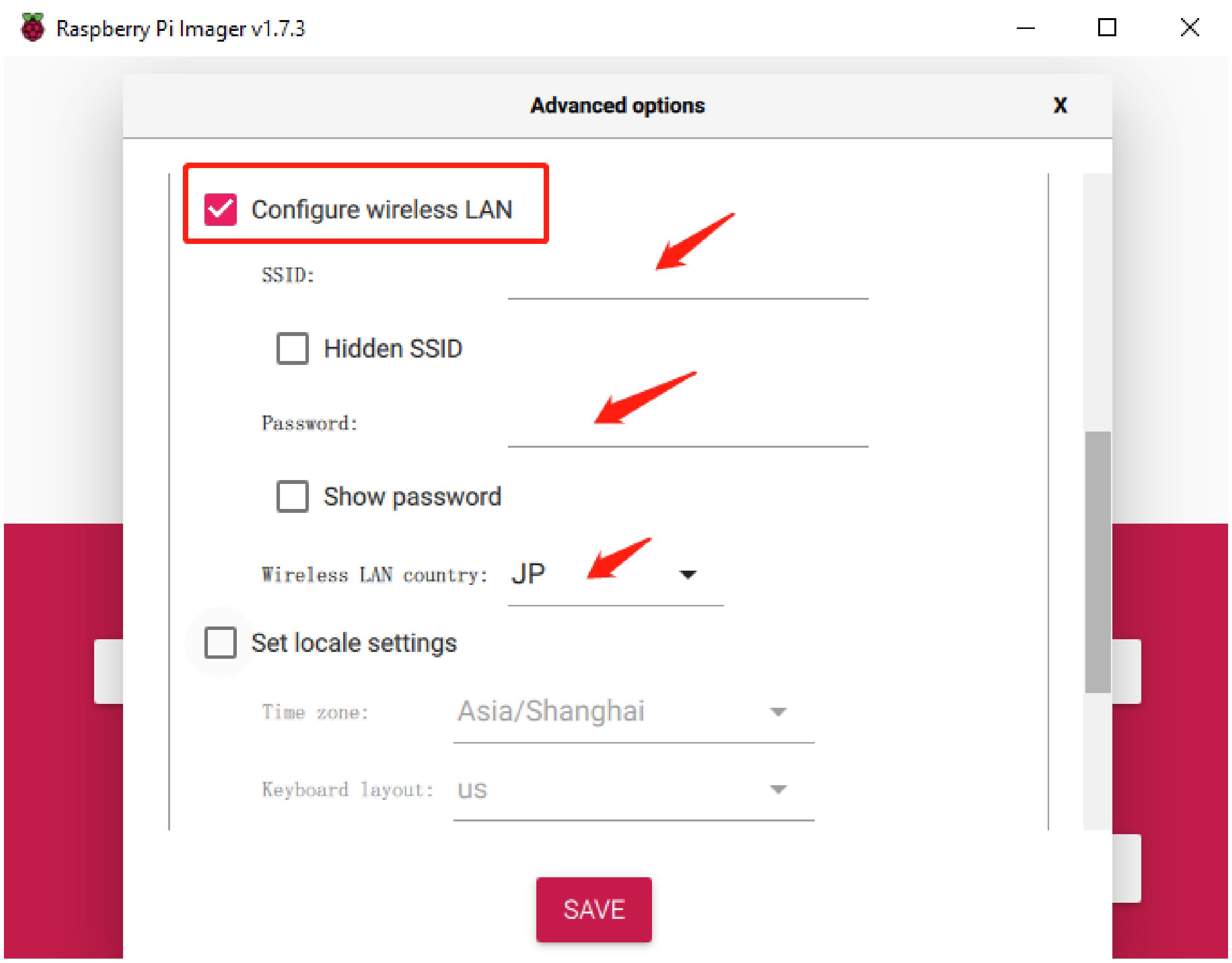
Advanced options

Image customization options: to always use

- [ ] Set hostname: raspb
- [✓] Enable SSH
  - [● Use password authentication]
  - ○ Allow public-key authentication only
  - Set authorized_keys for 'pi': 5nL0= 1035791926@qq.com
- [✓] Set username and password
  - Username: pi
  - Password:

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.



Configure wireless LAN [✓]

SSID:

Hidden SSID

Password:

Show password

Wireless LAN country: JP

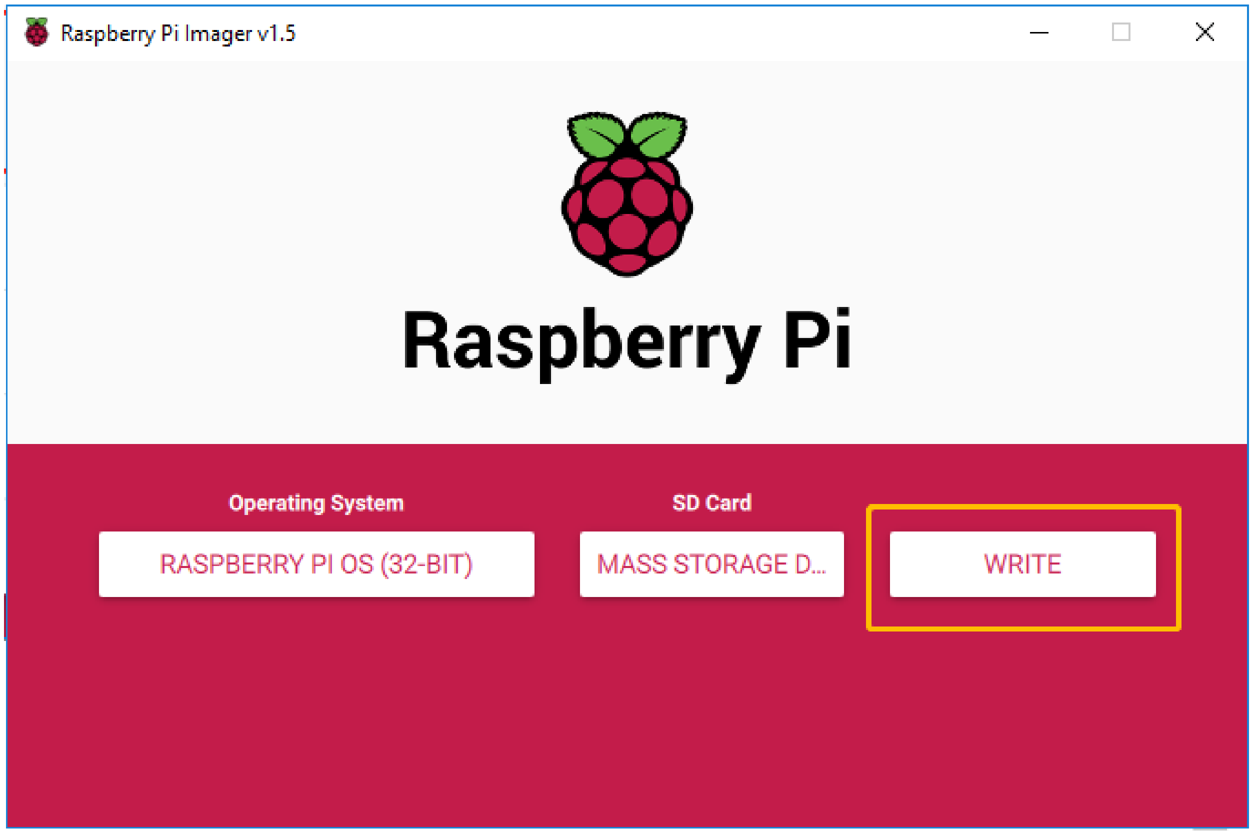
Set locale settings

Time zone: Asia/Shanghai

Keyboard layout: us

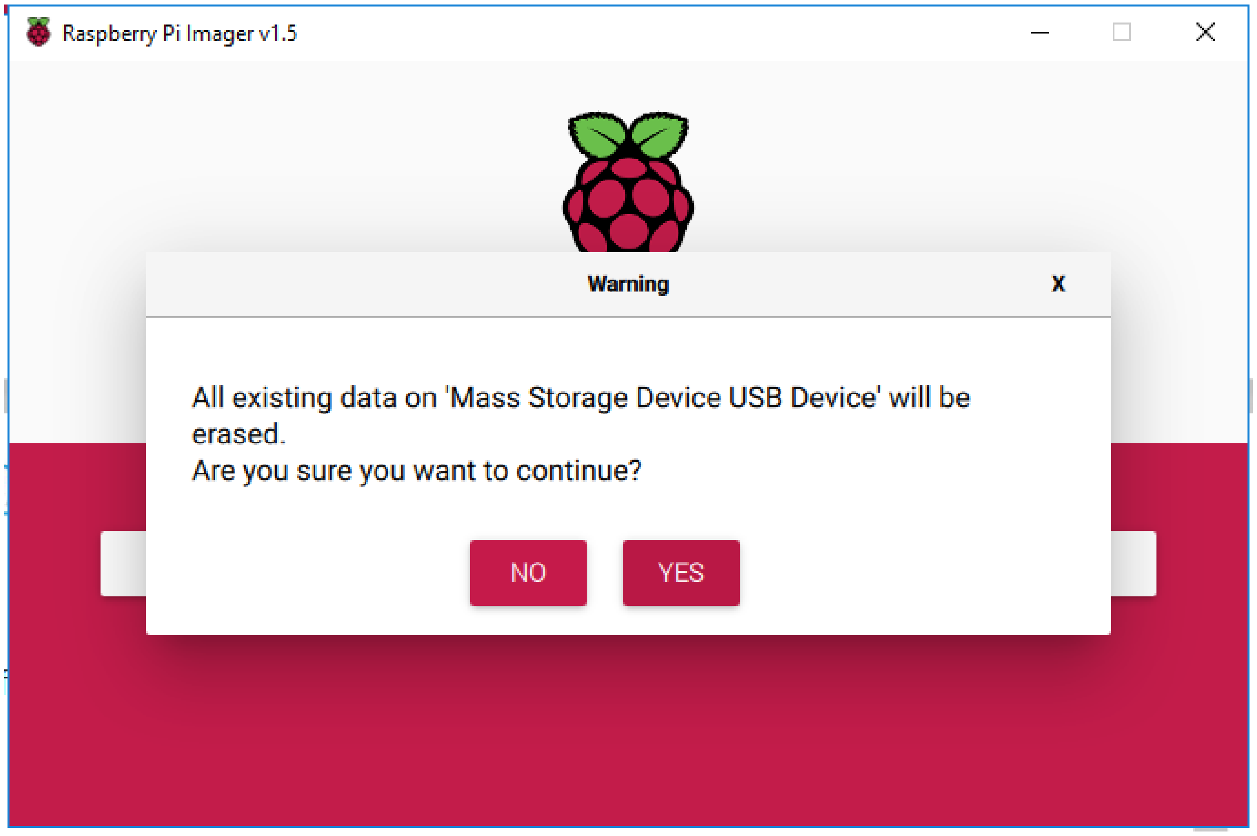
Step 7

Click the **WRITE** button.



Step 8

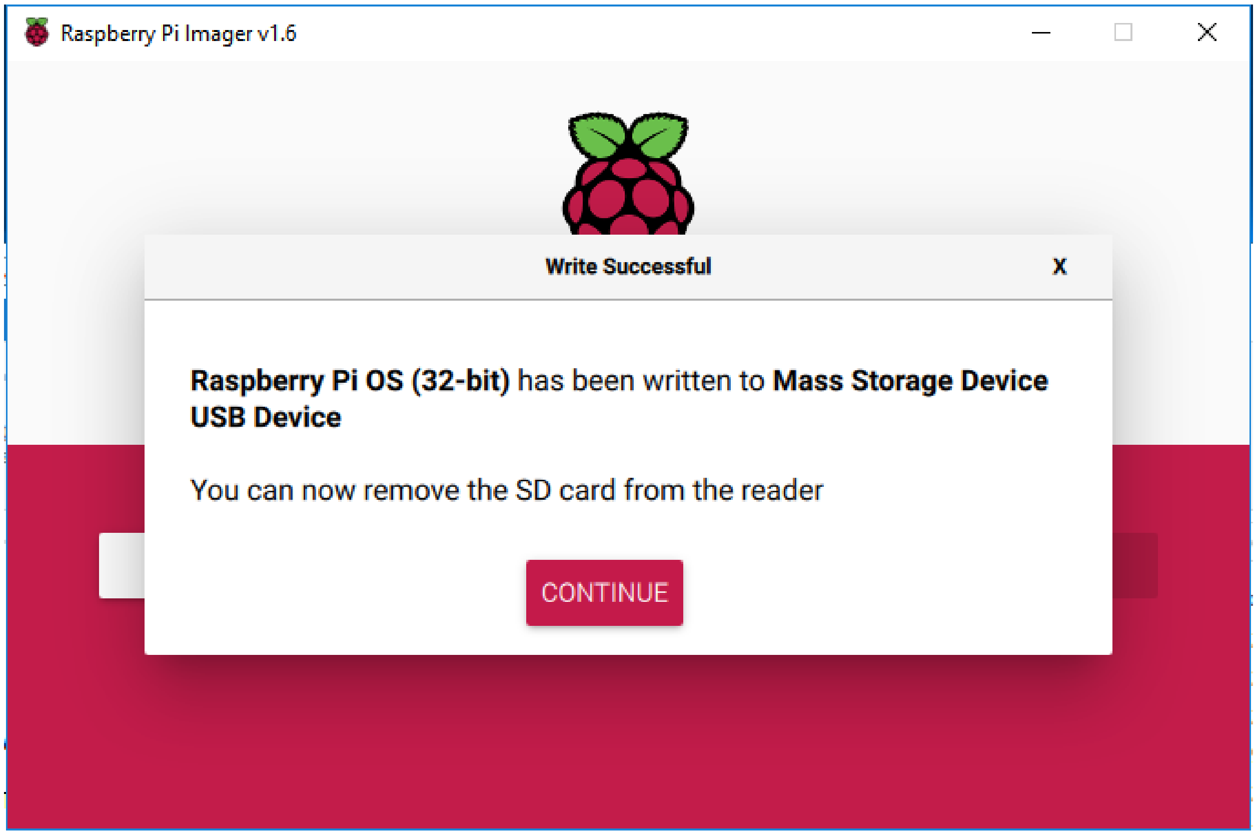
If your SD card currently has any files on it, you may wish to back up these files first to prevent you from permanently losing them. If there is no file to be backed up, click **Yes**.



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.

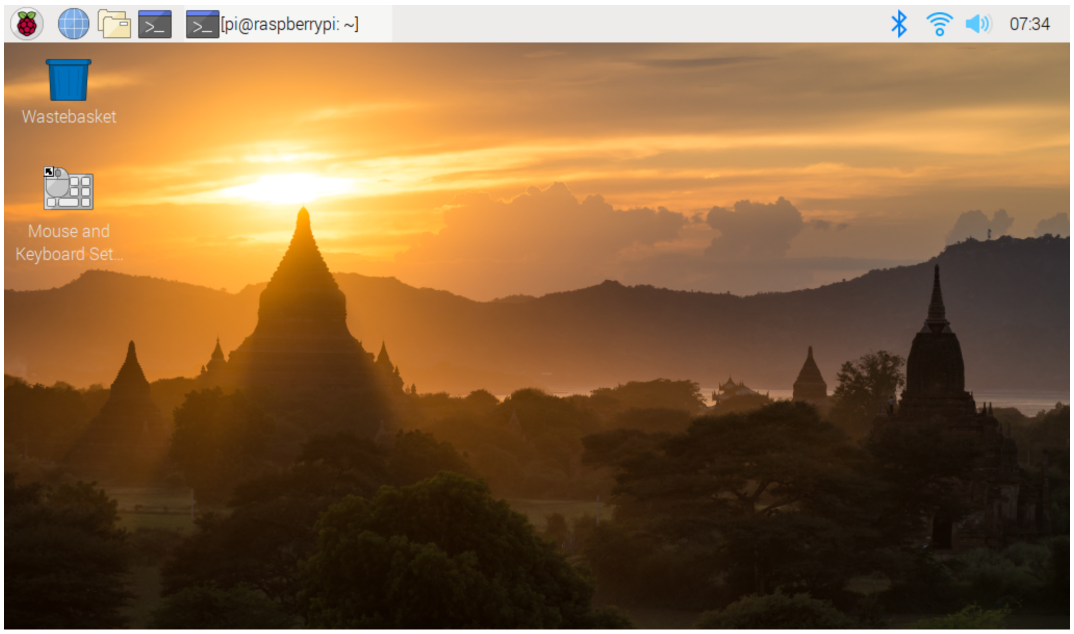
Required Components

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 monitor, you can remotely log into your Raspberry Pi.

You can apply the SSH command to open the Raspberry Pi's Bash shell. Bash is the standard default shell for Linux. The shell itself is a command (instruction) when the user uses Unix/Linux. Most of what you need to do can be done through the shell.

If you're not satisfied with using the command window to access your Raspberry Pi, you can also use the remote desktop feature to easily manage files on your Raspberry Pi using a GUI.

See below for detailed tutorials for each system.

Mac OS X user

For Mac users, accessing the Raspberry Pi desktop directly via VNC is more convenient than from the command line. You can access it via Finder by entering the set account password after enabling VNC on the Raspberry Pi side.

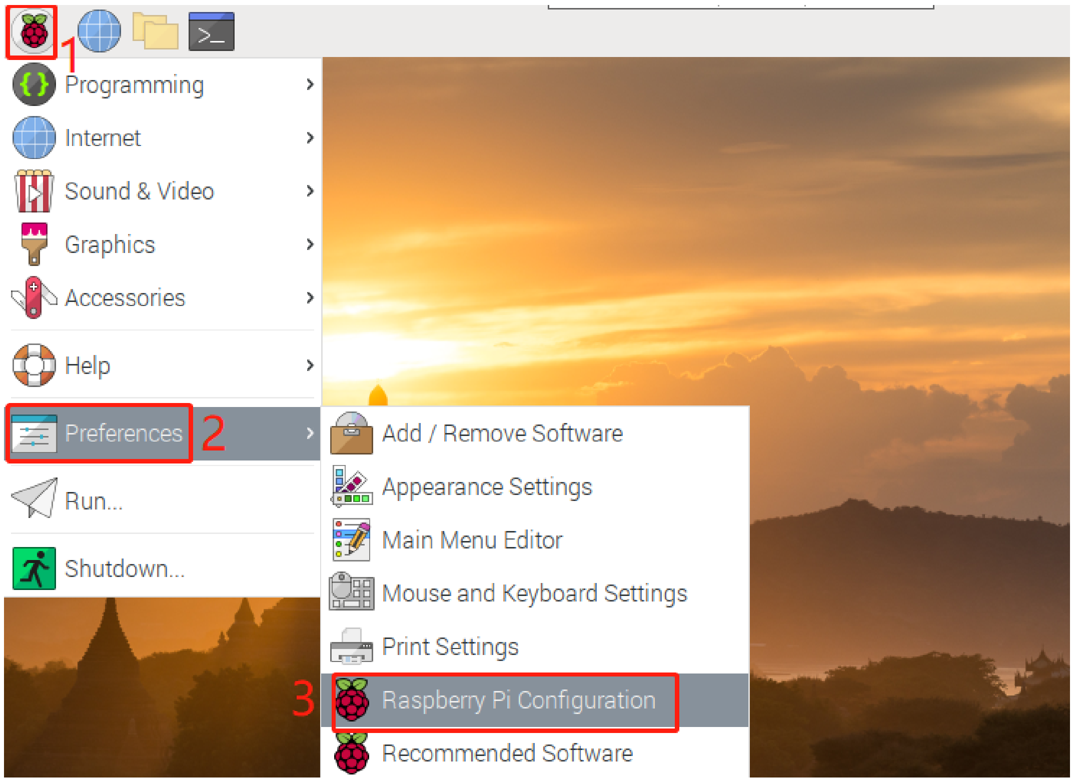
Note that this method does not encrypt communication between the Mac and Raspberry Pi. The communication will take place within your home or business network, so even if it's unprotected, it won't be an issue. However, if you are concerned about it, you can install a VNC application such as VNC® Viewer.

Alternatively it would be handy if you could use a temporary monitor (TV), mouse and keyboard to open the Raspberry Pi desktop directly to set up VNC. If not, it doesn't matter, you can also use the SSH command to open the Raspberry Pi's Bash shell and then using the command to set up the VNC.

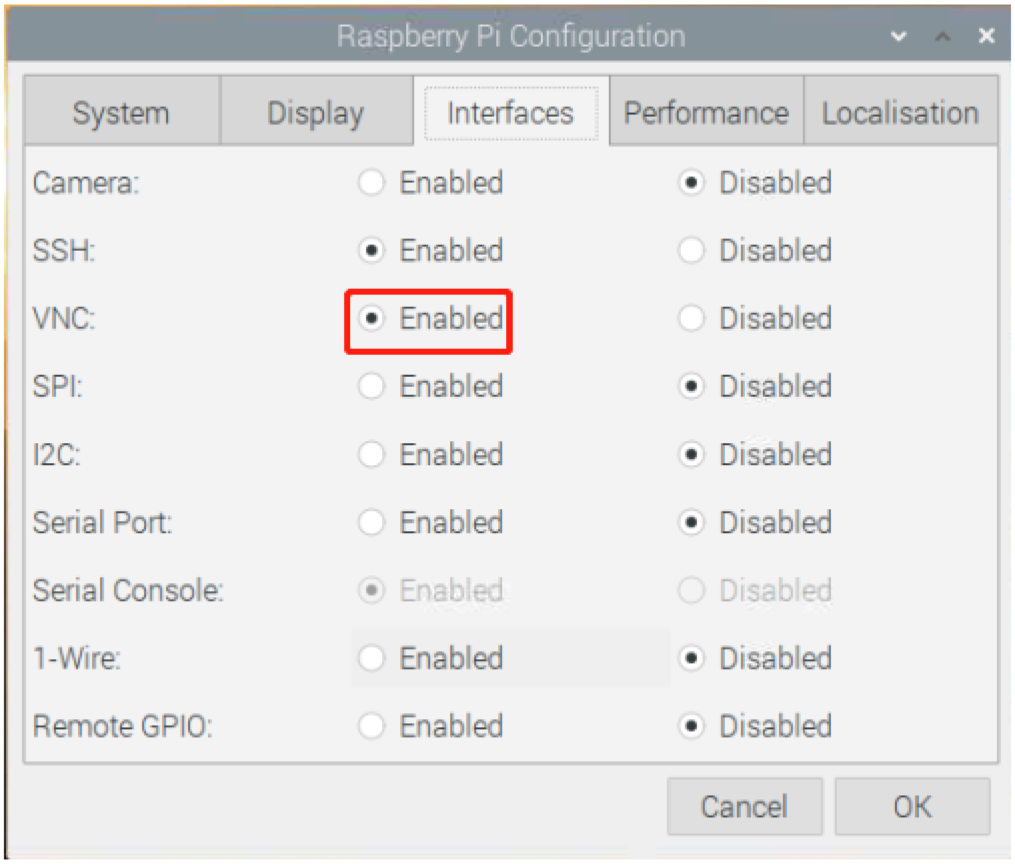
- *Have Temporarily Monitor (or TV)?*
- *Don't Have Temporarily Monitor (or TV)?*

Have Temporarily Monitor (or TV)?

1. Connect a monitor (or TV), mouse and keyboard to the Raspberry Pi and power it on. Select the menu according to the numbers in the figure.

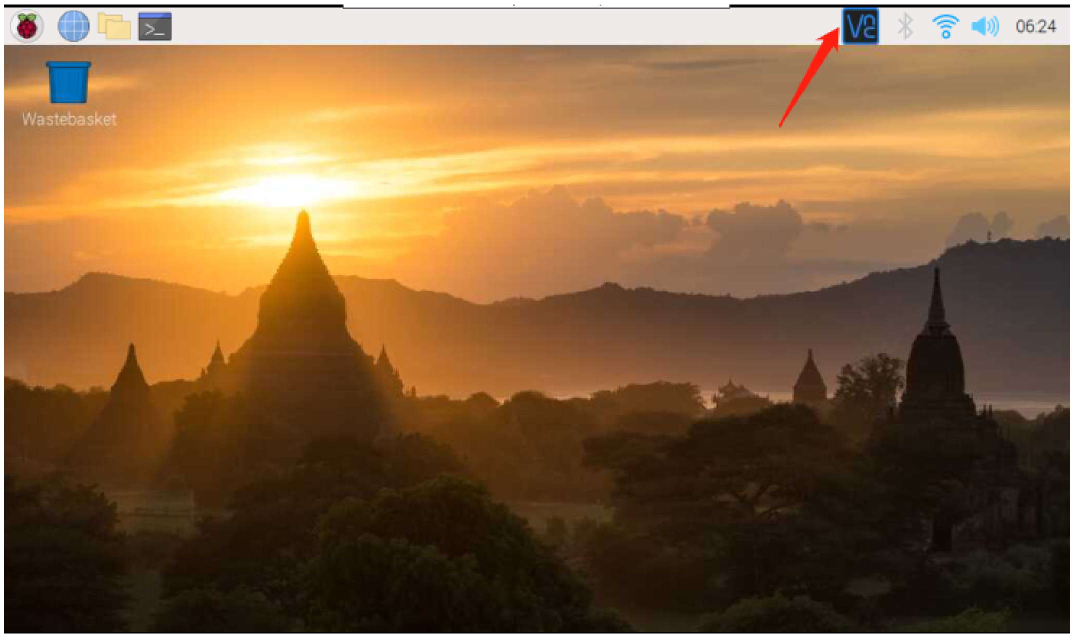


2. The following screen will be displayed. Set **VNC** to **Enabled** on the **Interfaces** tab, and click **OK**.

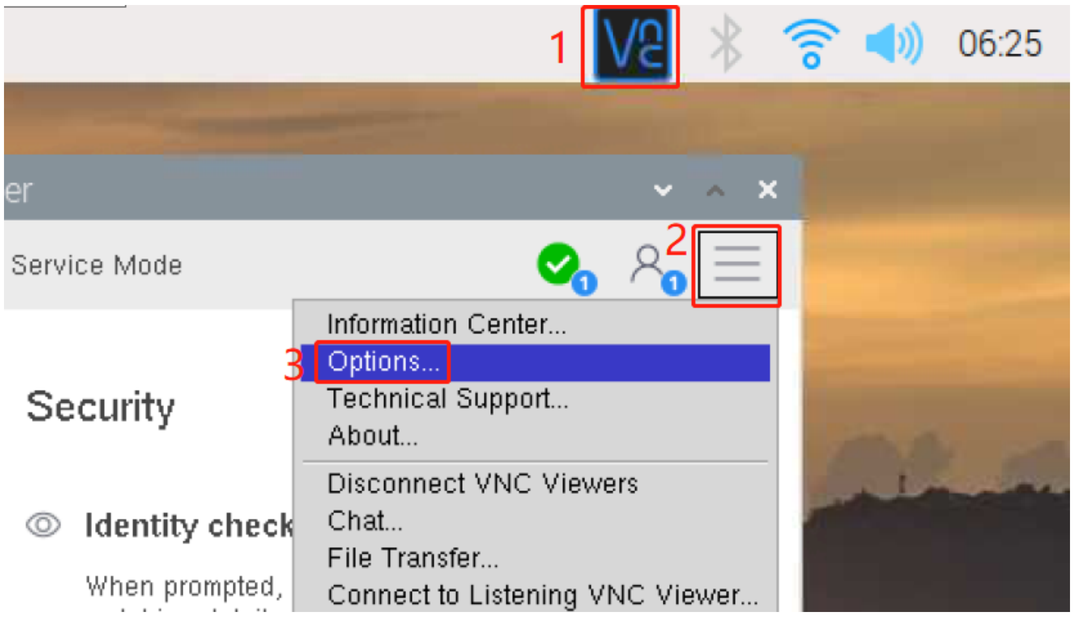


Camera:	○ Enabled	● Disabled
SSH:	● Enabled	○ Disabled
VNC:	● Enabled	○ Disabled
SPI:	○ Enabled	● Disabled
I2C:	○ Enabled	● Disabled
Serial Port:	○ Enabled	● Disabled
Serial Console:	● Enabled	○ Disabled
1-Wire:	○ Enabled	● Disabled
Remote GPIO:	○ Enabled	● Disabled

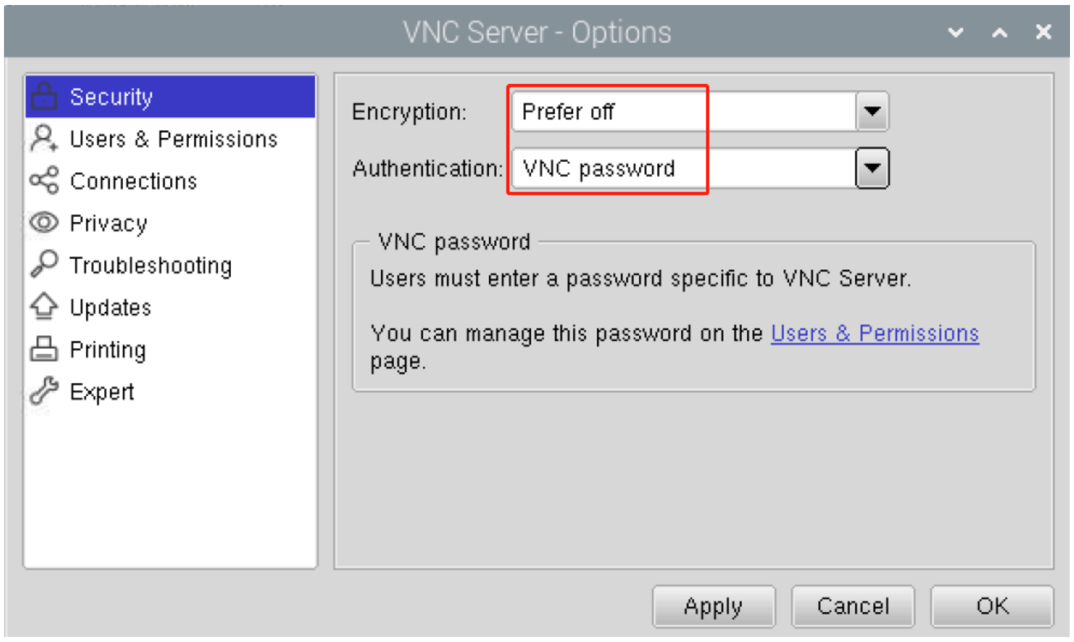
3. A VNC icon appears on the upper right of the screen and the VNC server starts.



4. Open the VNC server window by clicking on the **VNC** icon, then click on the **Menu** button in the top right corner and select **Options**.



5. You will be presented with the following screen where you can change the options.



Encryption: Prefer off

Authentication: VNC password

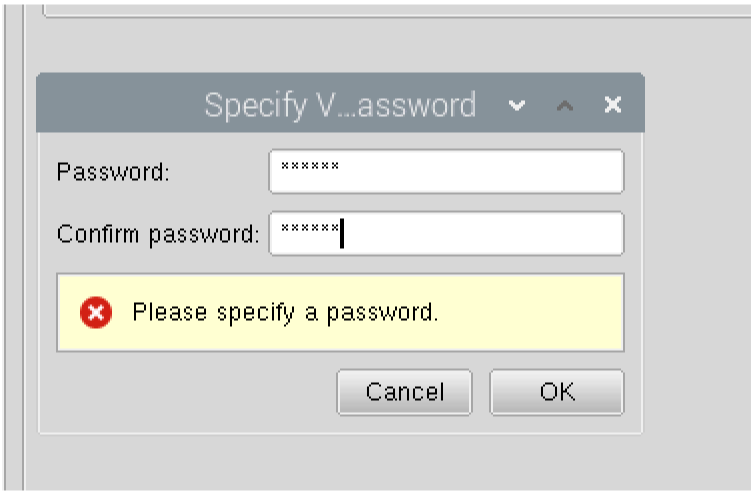
VNC password

Users must enter a password specific to VNC Server.

You can manage this password on the Users & Permissions page.

Set **Encryption** to **Prefer off** and **Authentication** to **VNC password**.

6. When you click the **OK** button, the password input screen is displayed. You can use the same password as the Raspberry pi password or a different password, so enter it and click **OK**.

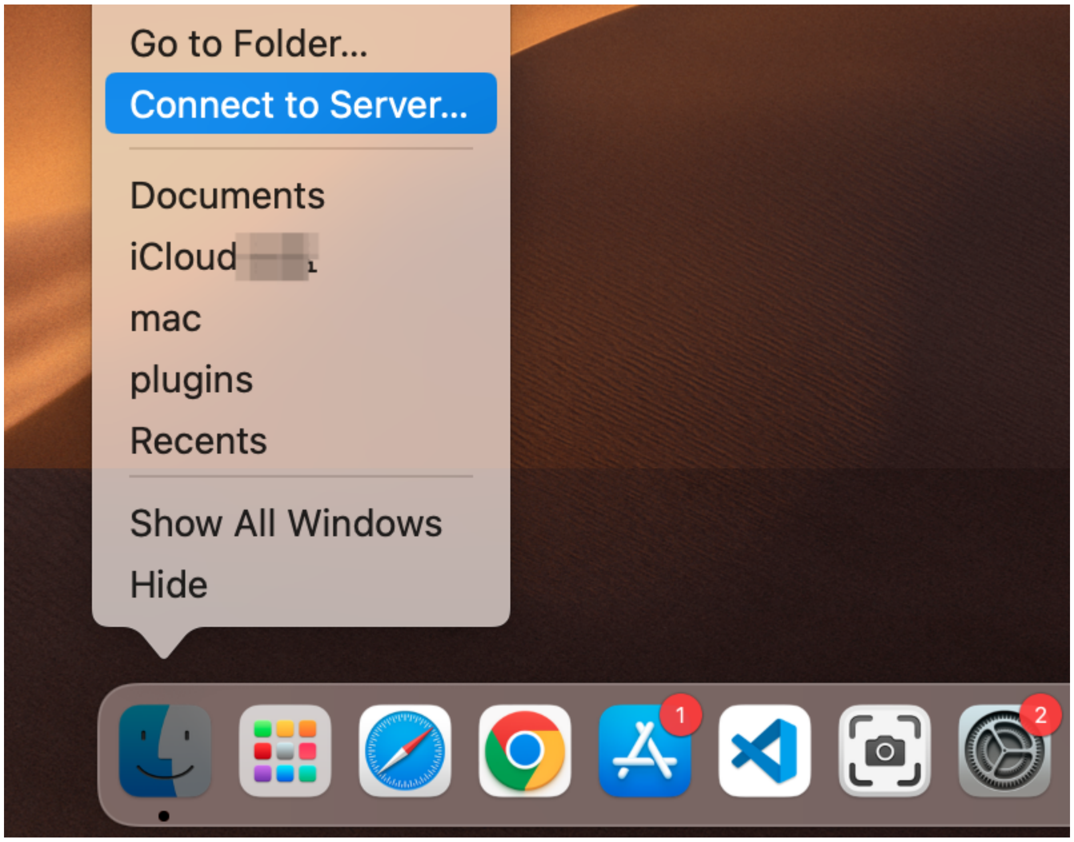


Please specify a password.

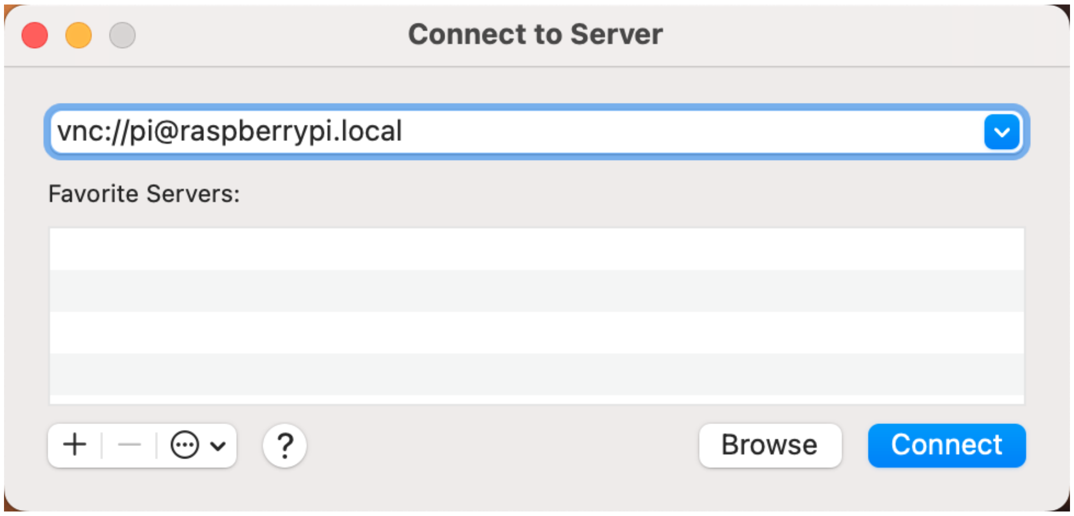
You are now ready to connect from your Mac. It's okay to disconnect the monitor.

From here, it will be the operation on the Mac side.

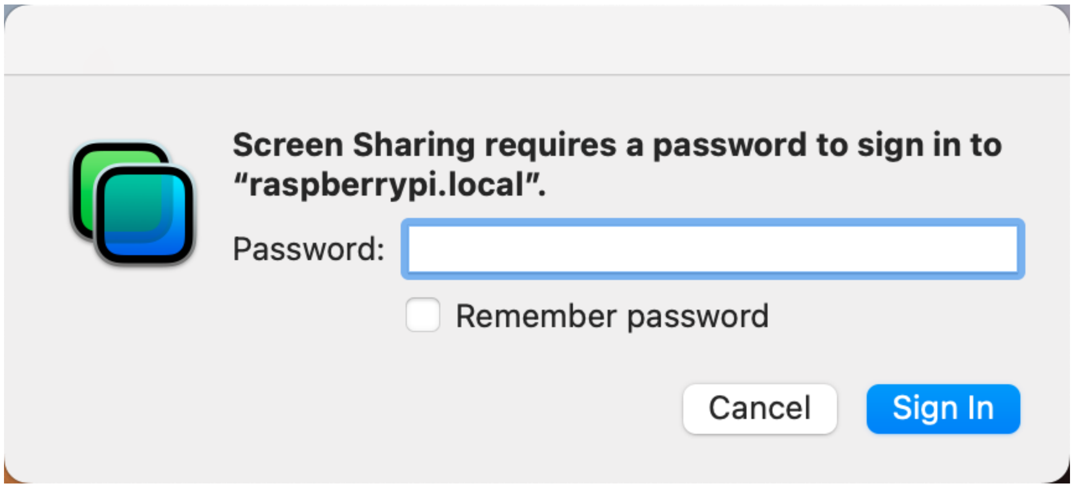
1. Now, select **Connect to Server** from the Finder's menu, which you can open by right-clicking.



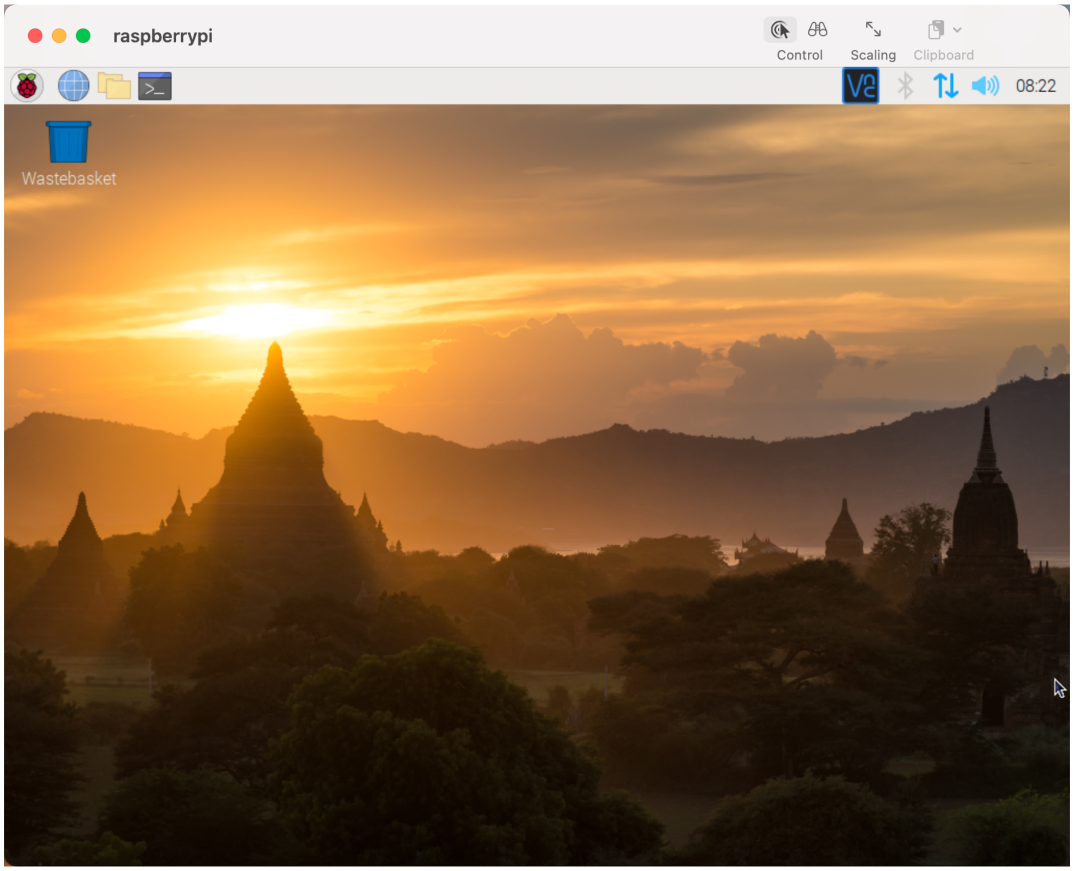
2. Type in `vnc://<username>@<hostname>.local` (or `vnc://<username>@<IP address>`). After entering, click **Connect**.



3. You will be asked for a password, so please enter it.


Screen Sharing requires a password to sign in to "raspberrypi.local".

4. The desktop of the Raspberry pi will be displayed, and you will be able to operate it from the Mac as it is.

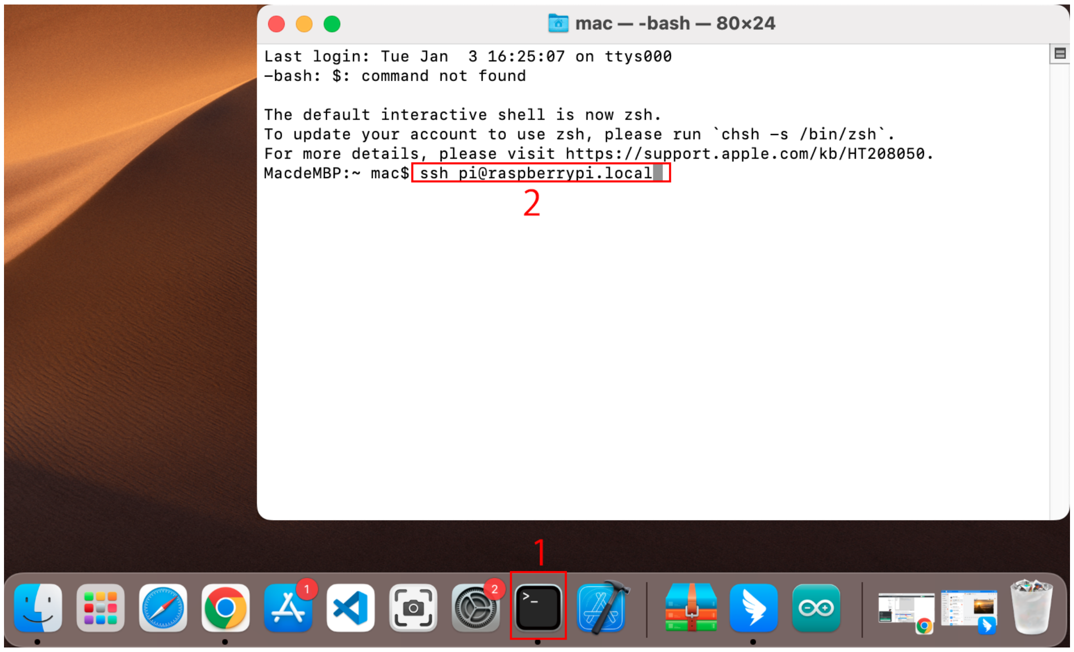


Don't Have Temporarily Monitor (or TV)?

- You can apply the SSH command to open the Raspberry Pi's Bash shell.
- Bash is the standard default shell for Linux.
- The shell is a command line interpreter (CLI) when the user uses Unix/Linux.
- Most of what you need to do can be done through the shell.
- After setting up the Raspberry pi side, you can access the desktop of the Raspberry Pi using the **Finder** from the Mac.

1. Type `ssh <username>@<hostname>.local` to connect to the Raspberry Pi.

```
ssh pi@raspberrypi.local
```



```
Last login: Tue Jan  3 16:25:07 on ttys000
-bash: $: command not found

The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
MacdeMBP:~ mac$ ssh pi@raspberrypi.local
```

2. The following message will be displayed only when you log in for the first time, so enter **yes**.

```
The authenticity of host 'raspberrypi.local␣
↪(2400:2410:2101:5800:635b:f0b6:2662:8cba)' can't be established.
ED25519 key fingerprint is SHA256:oo7x3ZSgAo032wD1tE8eW0fFM/
↪kmewIvRwkBys6XRwg.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])?
```

3. Enter the password for the Raspberry pi. The password you enter will not be displayed, so be careful not to make a mistake.

```
pi@raspberrypi.local's password:
Linux raspberrypi 5.15.61-v8+ #1579 SMP PREEMPT Fri Aug 26 11:16:44 BST␣
↪2022 aarch64

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: Thu Sep 22 12:18:22 2022
pi@raspberrypi:~ $
```

4. Set up your Raspberry Pi so that you can log in via VNC from your Mac once you have successfully logged into it. The first step is to update your operating system by running the following commands.

```
sudo apt update
sudo apt upgrade
```

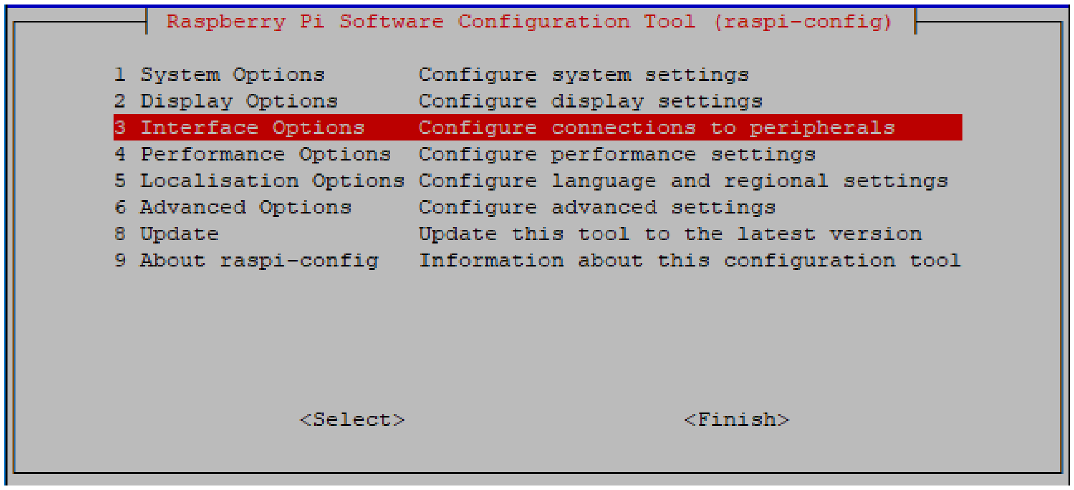
`Do you want to continue? [Y/n]`, Enter `Y` when prompted.

It may take some time for the update to finish. (It depends on the amount of updates at that time.)

5. Enter the following command to enable the **VNC Server**.

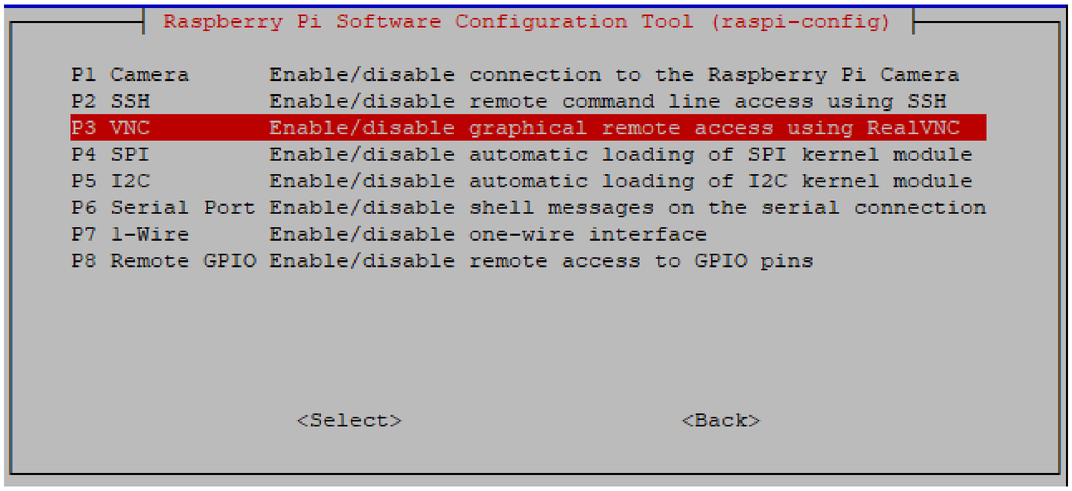
```
sudo raspi-config
```

6. The following screen will be displayed. Select **3 Interface Options** with the arrow keys on the keyboard and press the **Enter** key.



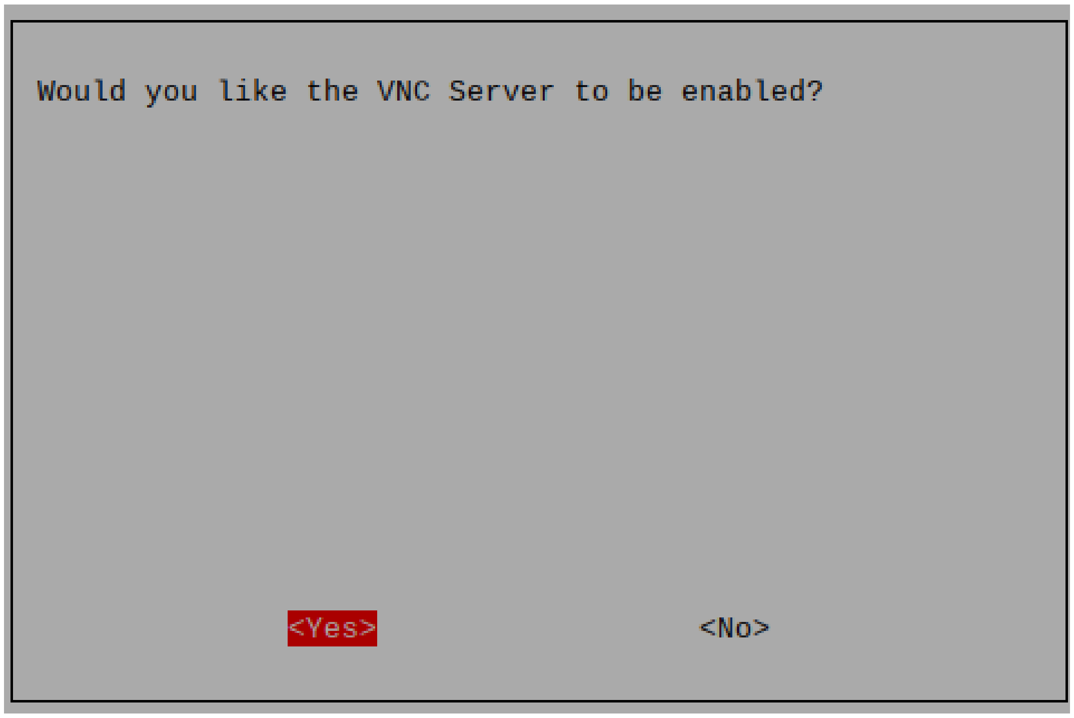
```
Raspberry Pi Software Configuration Tool (raspi-config)

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>
```

7. Then select **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>
```

8. Use the arrow keys on the keyboard to select **<Yes>** -> **<OK>** -> **<Finish>** to complete the setup.



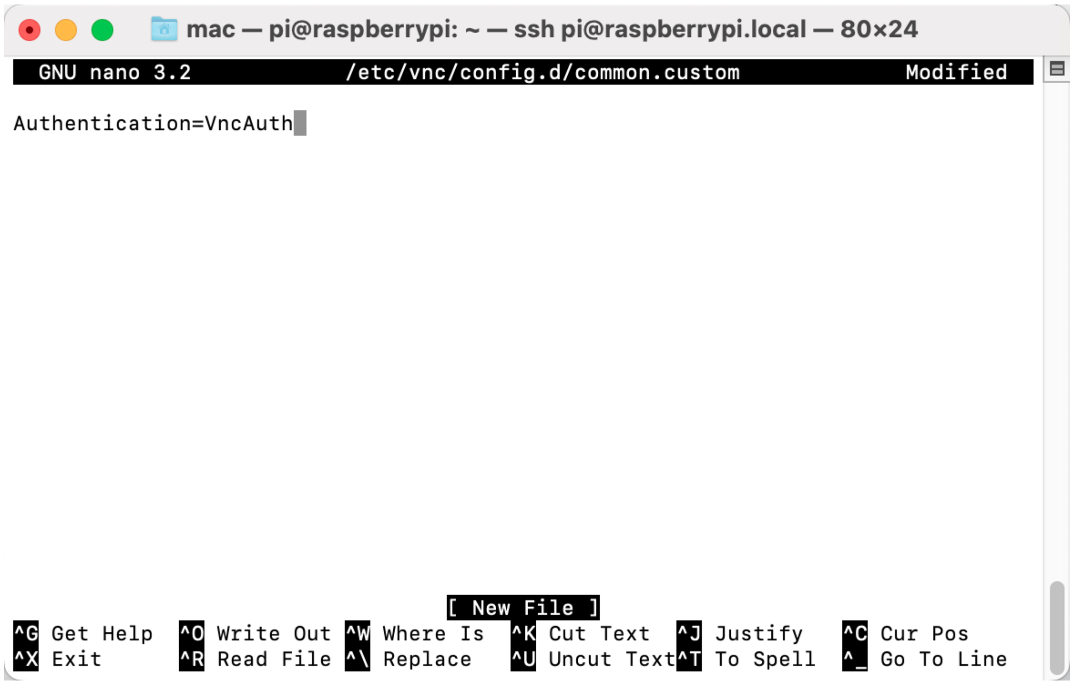
```
Would you like the VNC Server to be enabled?

                <Yes>                    <No>
```

9. Now that the VNC server has started, let's change the settings for connecting from a Mac.

   To specify parameters for all programs for all user accounts on the computer, create `/etc/vnc/config.d/common.custom`.

```
sudo nano /etc/vnc/config.d/common.custom
```

   After entering `Authentication=VncAuthenter`, press `Ctrl+X` -> `Y` -> `Enter` to save and exit.



```
mac — pi@raspberrypi: ~ — ssh pi@raspberrypi.local — 80×24
  GNU nano 3.2              /etc/vnc/config.d/common.custom              Modified

Authentication=VncAuth

                                  [ New File ]
^G Get Help  ^O Write Out ^W Where Is  ^K Cut Text  ^J Justify   ^C Cur Pos
^X Exit      ^R Read File ^\ Replace   ^U Uncut Text^T To Spell  ^_ Go To Line
```

10. In addition, set a password for logging in via VNC from a Mac. You can use the same password as the Raspberry

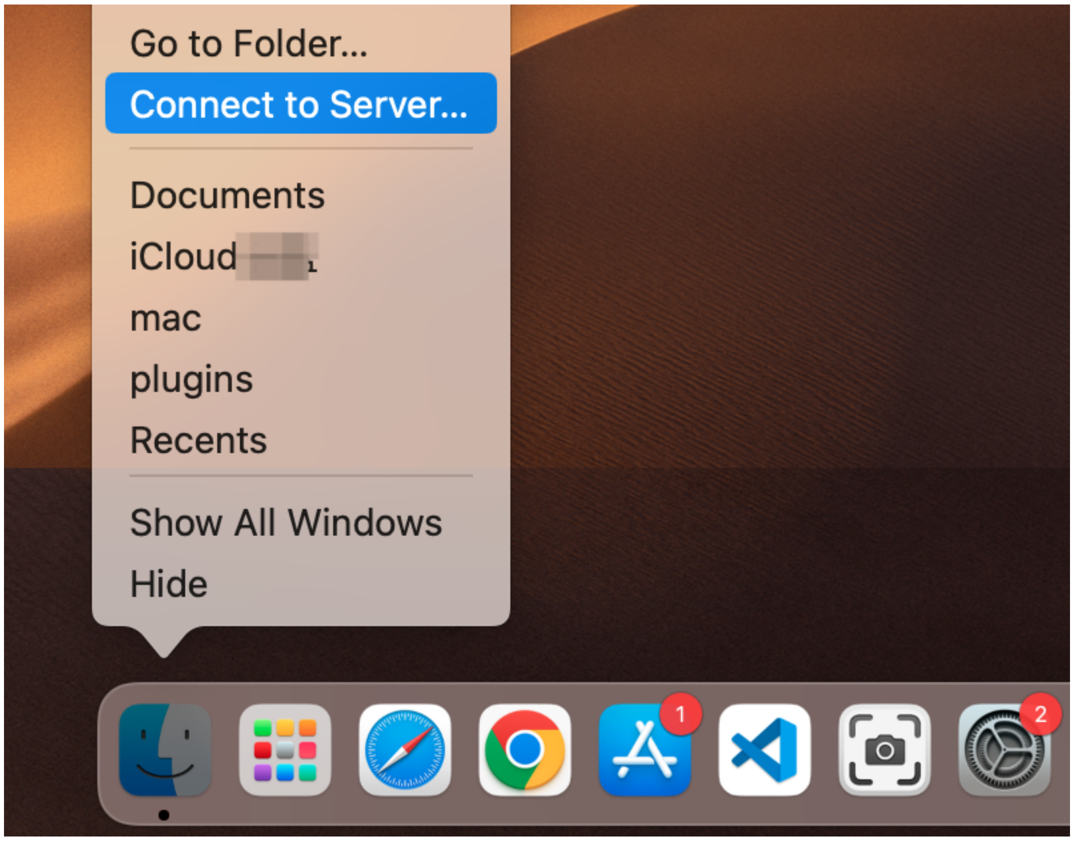
pi password or a different password.

```
sudo vncpasswd -service
```

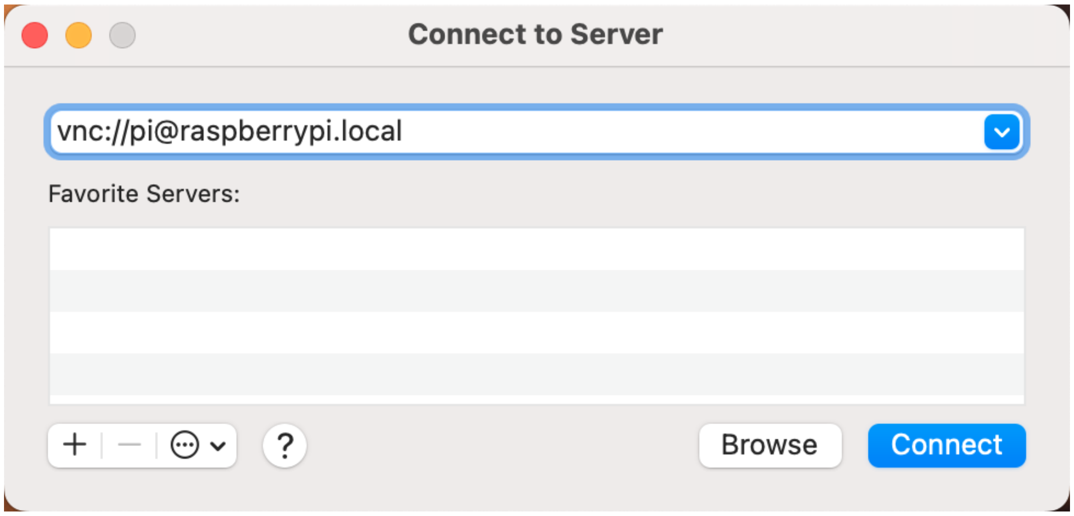
11. Once the setup is complete, restart the Raspberry Pi to apply the changes.

```
sudo sudo reboot
```

12. Now, select **Connect to Server** from the **Finder**'s menu, which you can open by right-clicking.

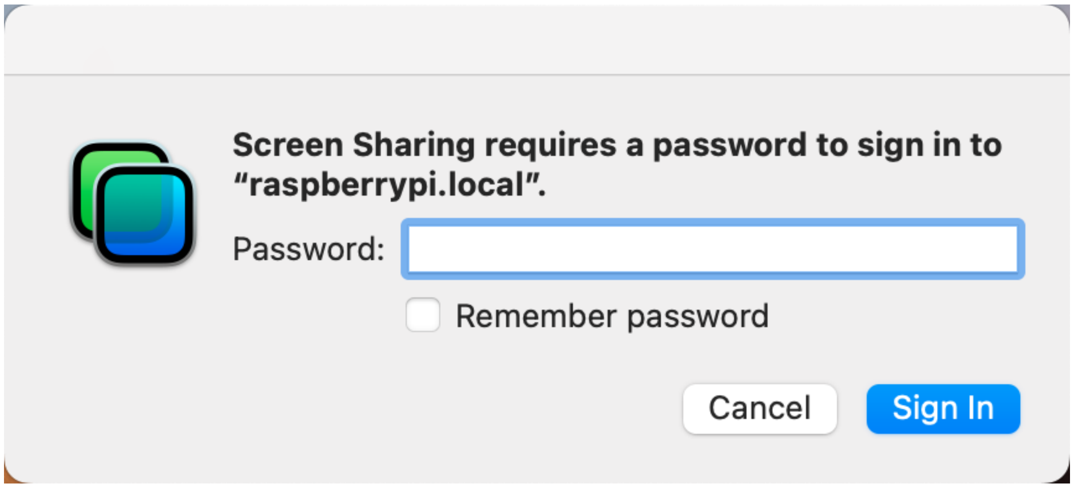


13. Type in `vnc://<username>@<hostname>.local` (or `vnc://<username>@<IP address>`). After entering, click **Connect**.



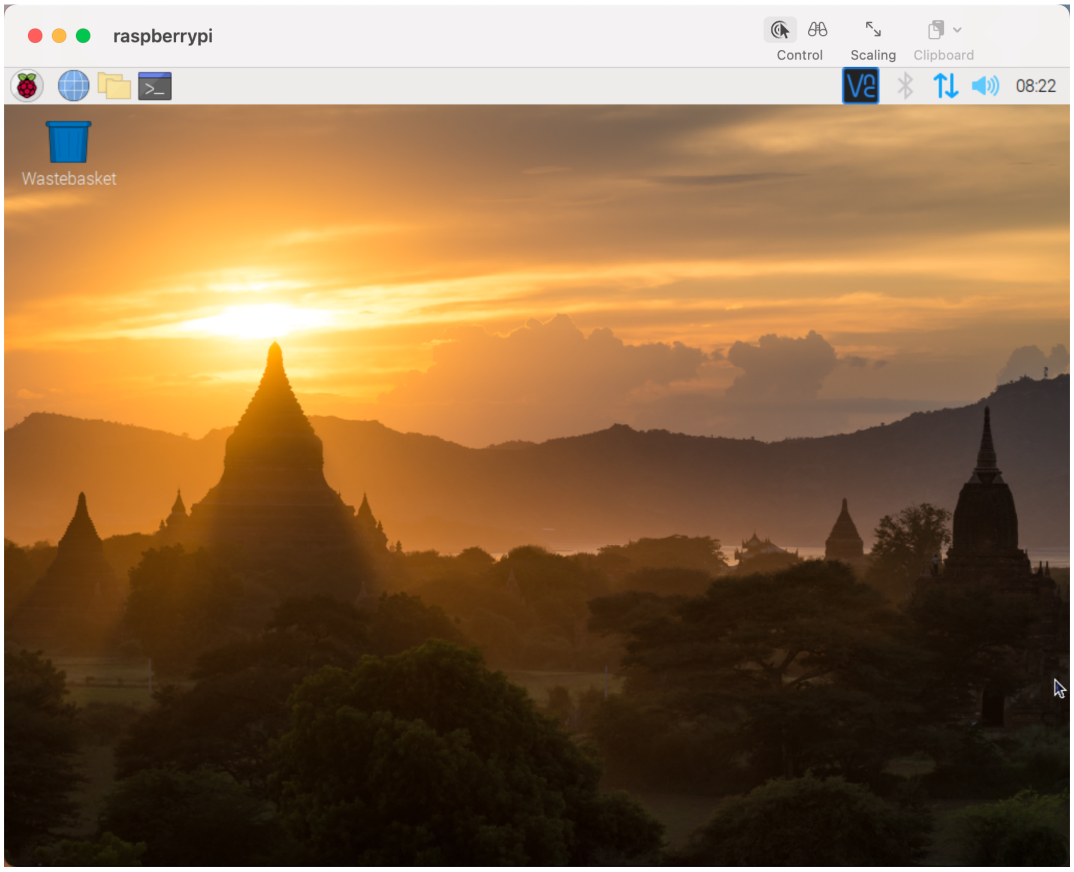
vnc://pi@raspberrypi.local

14. You will be asked for a password, so please enter it.


Screen Sharing requires a password to sign in to "raspberrypi.local".

Password:

15. The desktop of the Raspberry pi will be displayed, and you will be able to operate it from the Mac as it is.

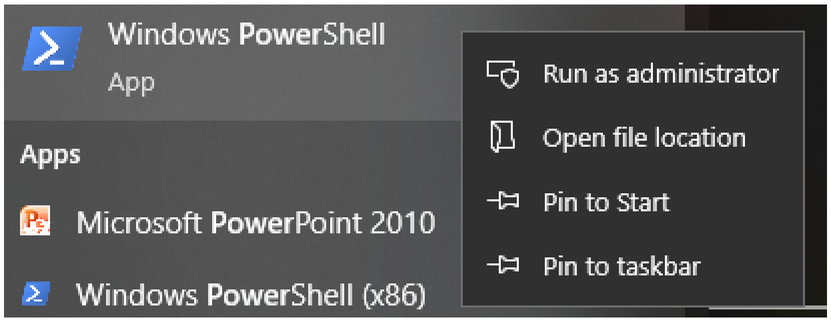


Windows Users

Login Raspberry Pi Remotely

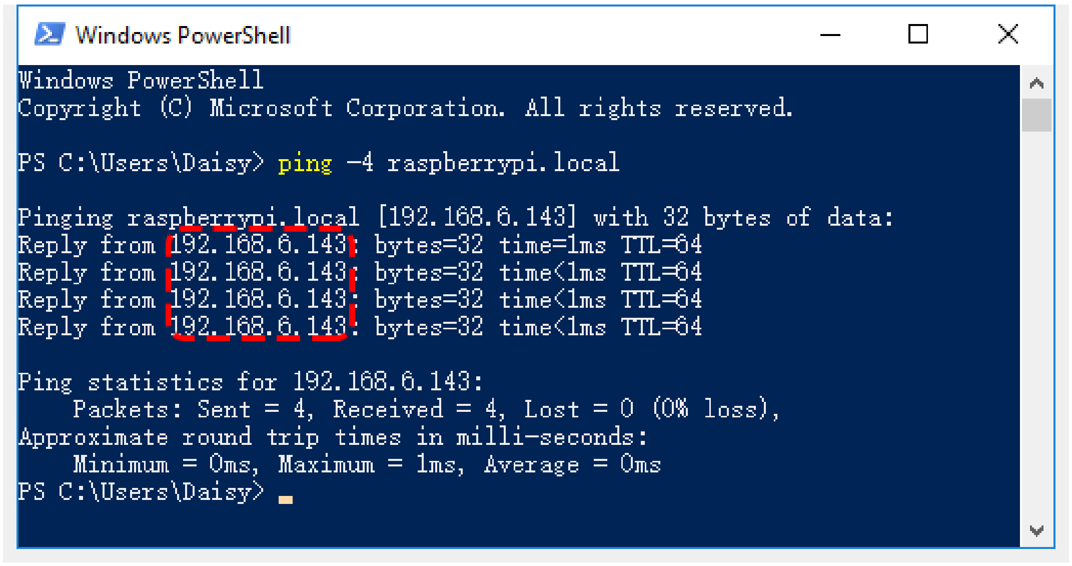
If you are using win10, you can use follow way to login Raspberry Pi remotely.

1. Type `powershell` in the search box of your Windows desktop, right click on the `Windows PowerShell`, and select `Run as administrator` from the menu that appears.



2. Then, check the IP address of your Raspberry Pi by typing `ping -4 <hostname>.local`.

```
ping -4 raspberrypi.local
```



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Users\Daisy> ping -4 raspberrypi.local

Pinging raspberrypi.local [192.168.6.143] with 32 bytes of data:
Reply from 192.168.6.143: bytes=32 time=1ms TTL=64
Reply from 192.168.6.143: bytes=32 time<1ms TTL=64
Reply from 192.168.6.143: bytes=32 time<1ms TTL=64
Reply from 192.168.6.143: bytes=32 time<1ms TTL=64

Ping statistics for 192.168.6.143:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 0ms, Maximum = 1ms, Average = 0ms
PS C:\Users\Daisy>
```

As shown above, you can see the Raspberry Pi's IP address after it has been connected to the network.

- If terminal prompts `Ping request could not find host pi.local. Please check the name and try again.`. Please follow the prompts to make sure the hostname you fill in is correct.
- Still can't get the IP? Check your network or WiFi configuration on the Raspberry Pi.

3. At this point you will be able to log in to your Raspberry Pi using the `ssh <username>@<hostname>.local` (or `ssh <username>@<IP address>`).

```
ssh pi@raspberrypi.local
```

Warning: If a prompt appears `The term 'ssh' is not recognized as the name of a cmdlet...`.

It means your system is too old and does not have ssh tools pre-installed, you need to manually *Install OpenSSH via Powershell*.

Or use a third party tool like *PuTTY*.

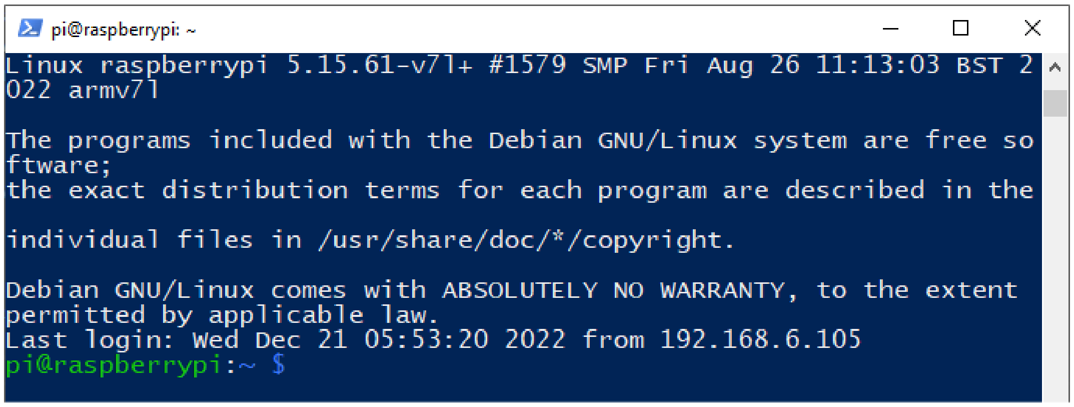
4. The following message will be displayed only when you log in for the first time, so enter `yes`.

```
The authenticity of host 'raspberrypi.local␣
↪(2400:2410:2101:5800:635b:f0b6:2662:8cba)' can't be established.
ED25519 key fingerprint is SHA256:oo7x3ZSgAo032wD1tE8eW0fFM/
↪kmewIvRwkBys6XRwg.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])?
```

5. Input the password you set before. (Mine is `raspberry`.)

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.

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



pi@raspberrypi: ~

```
Linux raspberrypi 5.15.61-v7l+ #1579 SMP Fri Aug 26 11:13:03 BST 2
022 armv7l

The programs included with the Debian GNU/Linux system are free so
ftware;
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: Wed Dec 21 05:53:20 2022 from 192.168.6.105
pi@raspberrypi:~ $
```

Remote Desktop

If you're not satisfied with using the command window to access your Raspberry Pi, you can also use the remote desktop feature to easily manage files on your Raspberry Pi using a GUI.

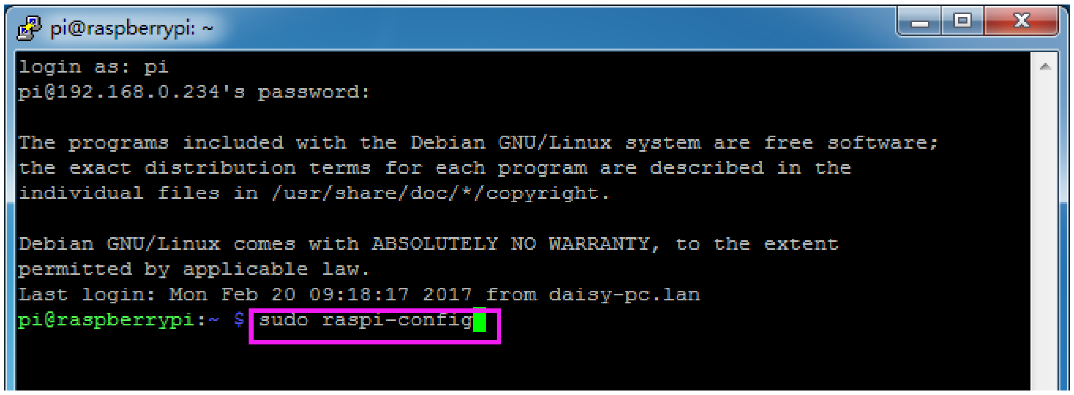
Here we use VNC® Viewer.

Enable VNC service

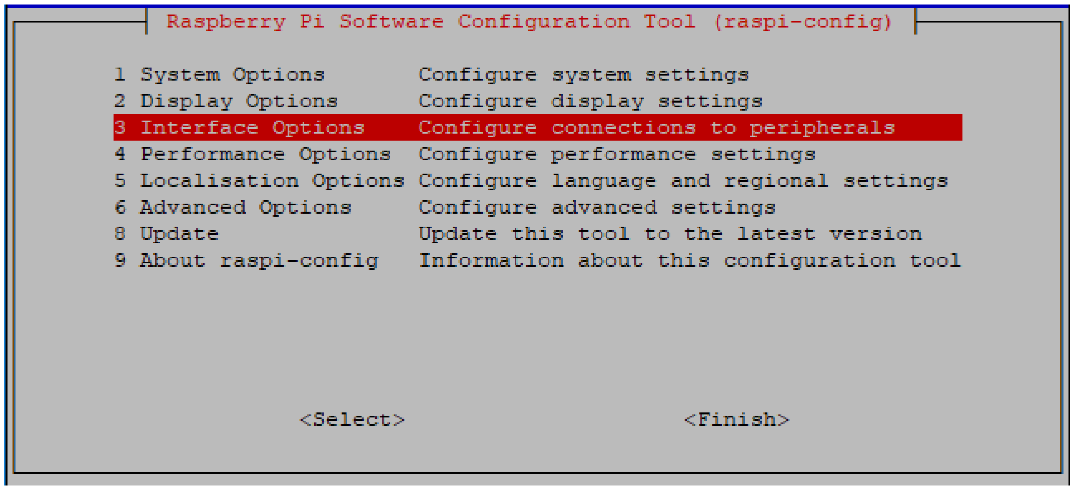
The VNC service has been installed in the system. By default, VNC is disabled. You need to enable it in config.

1. Input the following command:

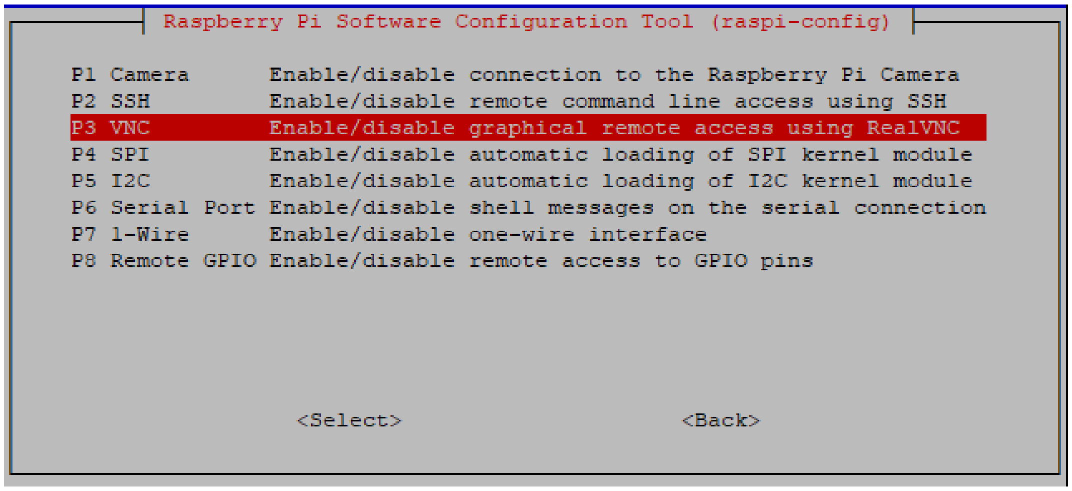
```
sudo raspi-config
```


```
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
```

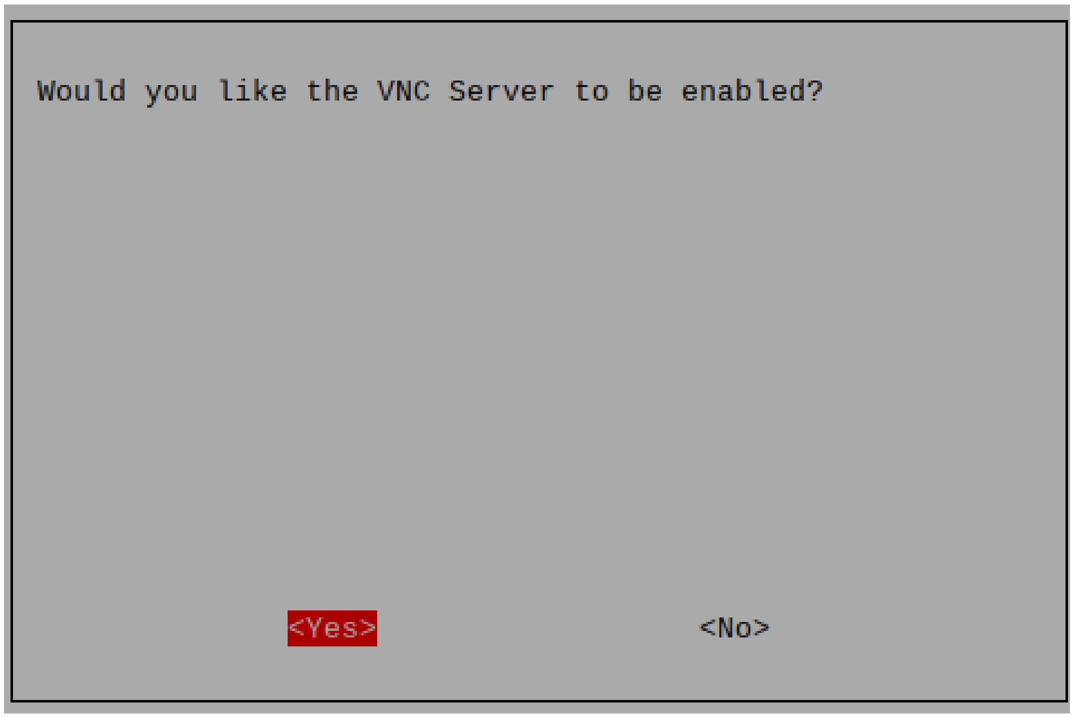
2. Choose **3 Interfacing Options** by press the down arrow key on your keyboard, then press the **Enter** key.


```
        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>
```

3. Then **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>
```

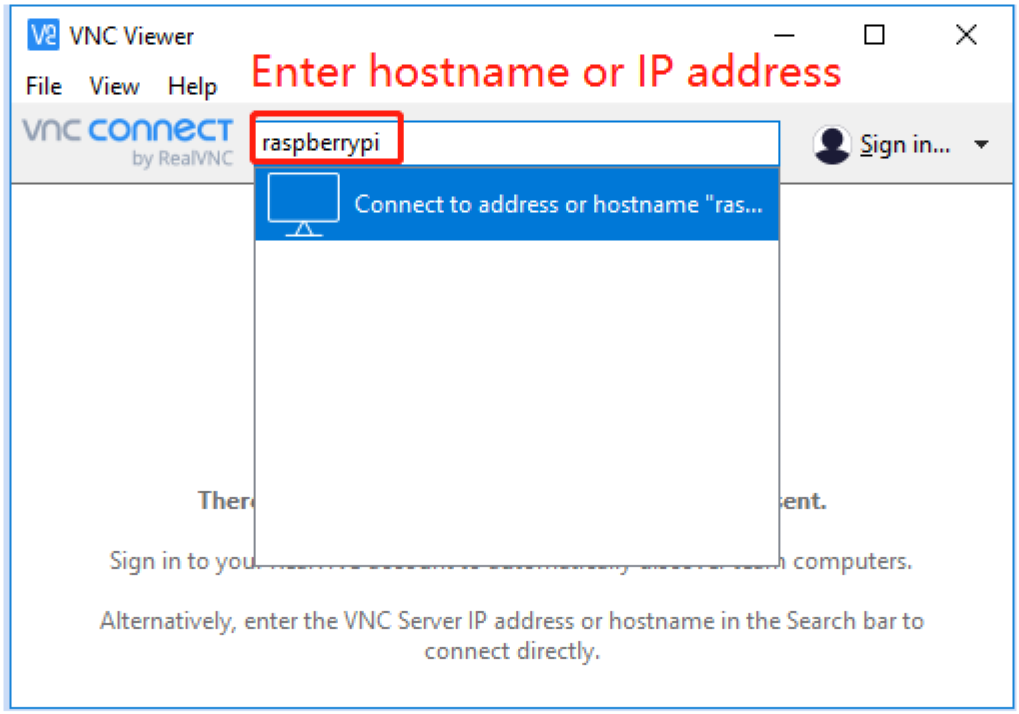
4. Use the arrow keys on the keyboard to select **<Yes>** -> **<OK>** -> **<Finish>** to complete the setup.



```
Would you like the VNC Server to be enabled?

                    <Yes>                    <No>
```

Login to VNC

1. You need to download and install the VNC Viewer on personal computer.
2. Open it once the installation is complete. Then, enter the host name or IP address and press Enter.

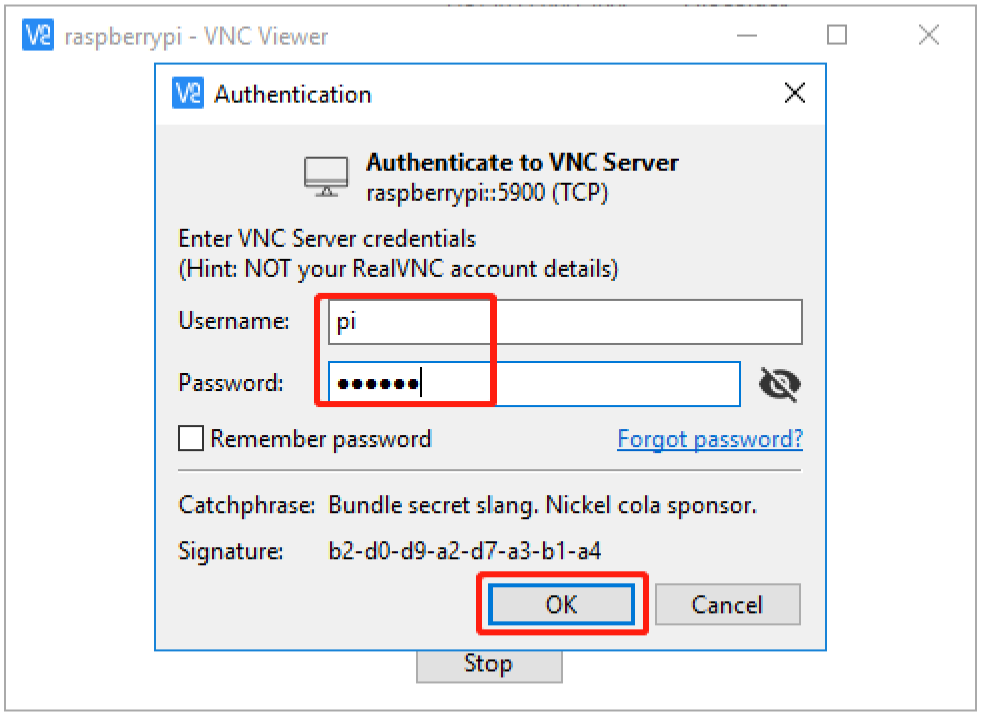

VNC Viewer
File
View
Help
Enter hostname or IP address
vnc connect
by RealVNC
raspberrypi
Sign in...
Connect to address or hostname "ras...

There… …ent.

Sign in to you… …computers.

Alternatively, enter the VNC Server IP address or hostname in the Search bar to connect directly.

3. After entering your Raspberry Pi name and password, click **OK**.


raspberrypi - VNC Viewer

Authentication

Authenticate to VNC Server
raspberrypi::5900 (TCP)

Enter VNC Server credentials
(Hint: NOT your RealVNC account details)

Username: pi

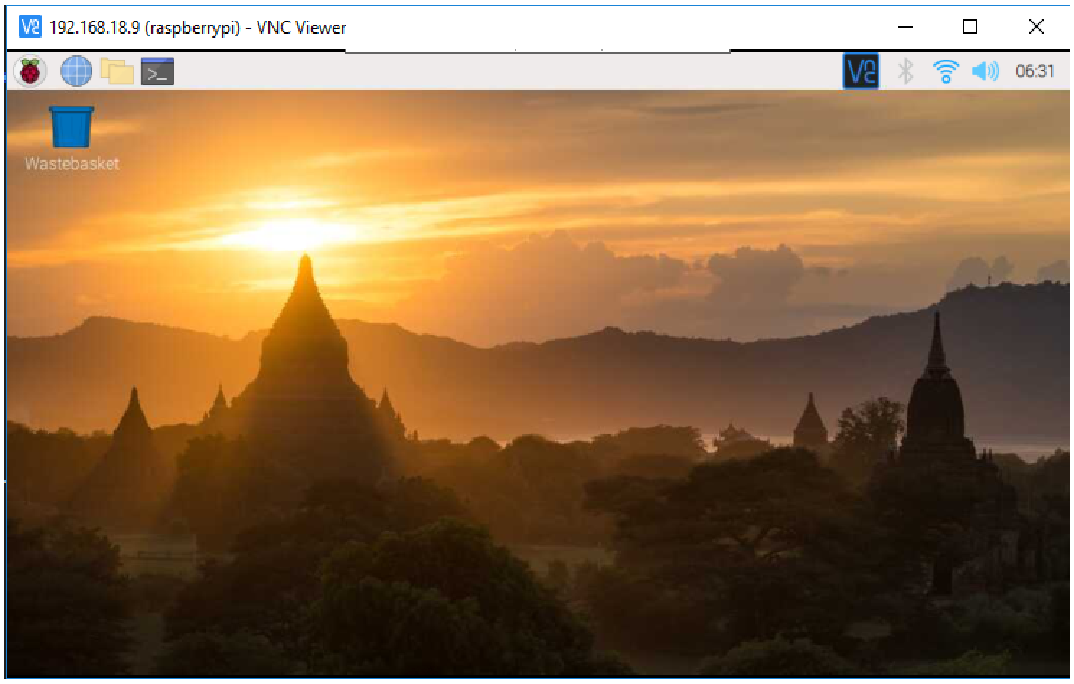
Password: ••••••

Remember password   Forgot password?

Catchphrase: Bundle secret slang. Nickel cola sponsor.

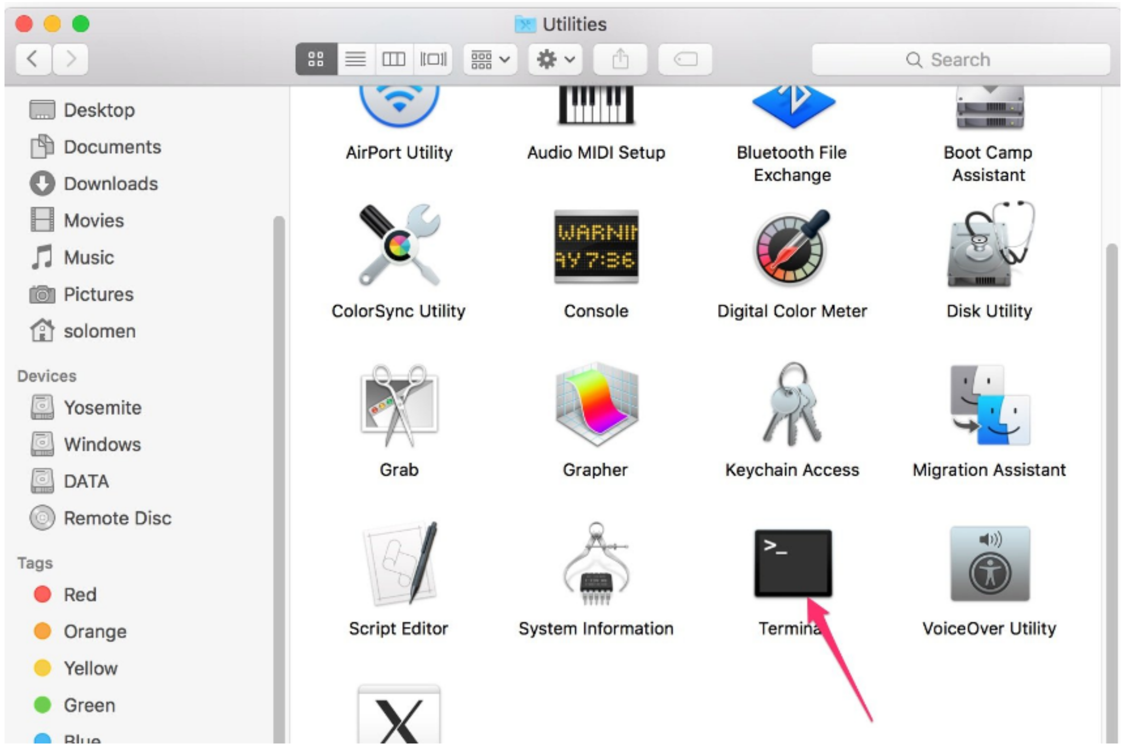
Signature: b2-d0-d9-a2-d7-a3-b1-a4

4. Now you can see the desktop of the Raspberry Pi.



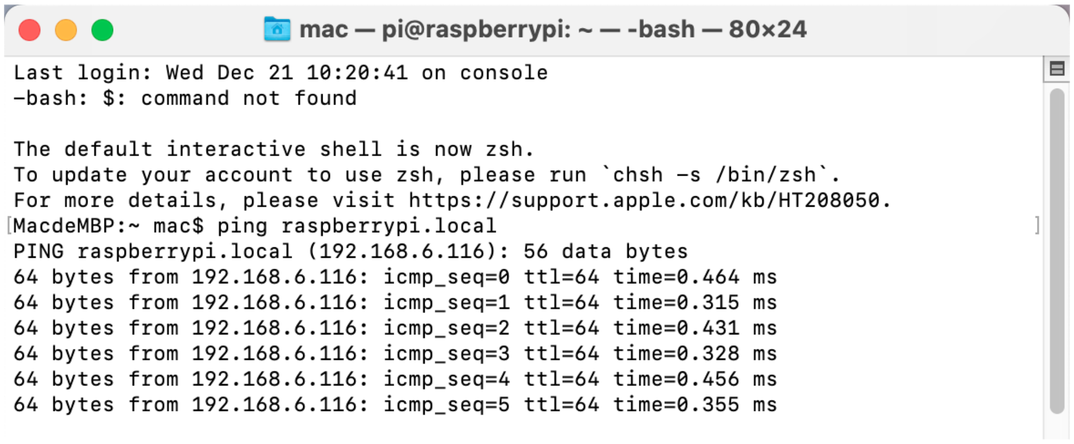
Linux /Unix Users

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



1. Check if your Raspberry Pi is on the same network by type in `ping <hostname>.local`.

```
ping raspberrypi.local
```



```
Last login: Wed Dec 21 10:20:41 on console
-bash: $: command not found

The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
MacdeMBP:~ mac$ ping raspberrypi.local
PING raspberrypi.local (192.168.6.116): 56 data bytes
64 bytes from 192.168.6.116: icmp_seq=0 ttl=64 time=0.464 ms
64 bytes from 192.168.6.116: icmp_seq=1 ttl=64 time=0.315 ms
64 bytes from 192.168.6.116: icmp_seq=2 ttl=64 time=0.431 ms
64 bytes from 192.168.6.116: icmp_seq=3 ttl=64 time=0.328 ms
64 bytes from 192.168.6.116: icmp_seq=4 ttl=64 time=0.456 ms
64 bytes from 192.168.6.116: icmp_seq=5 ttl=64 time=0.355 ms
```

As shown above, you can see the Raspberry Pi's IP address after it has been connected to the network.

- If terminal prompts `Ping request could not find host pi.local. Please check the name and try again.`. Please follow the prompts to make sure the hostname you fill in is correct.
- Still can't get the IP? Check your network or WiFi configuration on the Raspberry Pi.

2. Type in `ssh <username>@<hostname>.local` (or `ssh <username>@<IP address>`).

```
ssh pi@raspberrypi.local
```

Note: If a prompt appears `The term 'ssh' is not recognized as the name of a cmdlet...`.

It means your system is too old and does not have ssh tools pre-installed, you need to manually *Install OpenSSH via Powershell*.

Or use a third party tool like *PuTTY*.

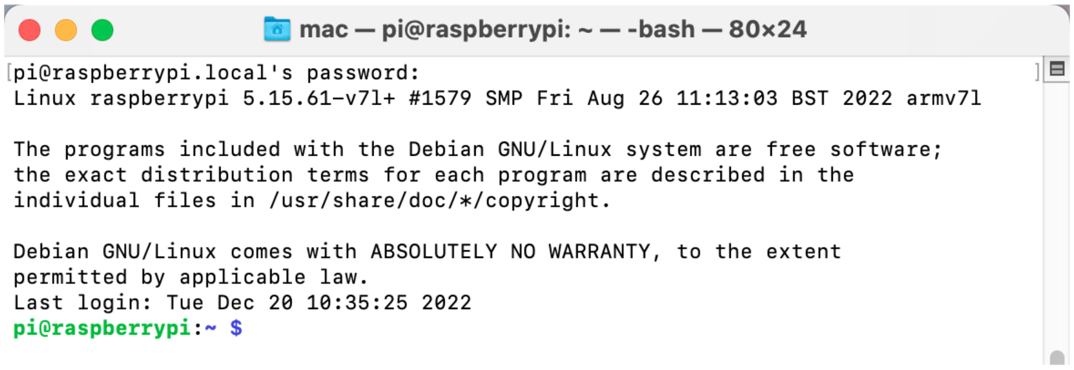
3. The following message will be displayed only when you log in for the first time, so enter `yes`.

```
The authenticity of host 'raspberrypi.local␣
↪(2400:2410:2101:5800:635b:f0b6:2662:8cba)' can't be established.
ED25519 key fingerprint is SHA256:oo7x3ZSgAo032wD1tE8eW0fFM/
↪kmewIvRwkBys6XRwg.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])?
```

4. Input the password you set before. (Mine is `raspberry`.)

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.

5. We now get the Raspberry Pi connected and are ready to go to the nextstep.



```
pi@raspberrypi.local's password:
Linux raspberrypi 5.15.61-v7l+ #1579 SMP Fri Aug 26 11:13:03 BST 2022 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 Dec 20 10:35:25 2022
pi@raspberrypi:~ $
```

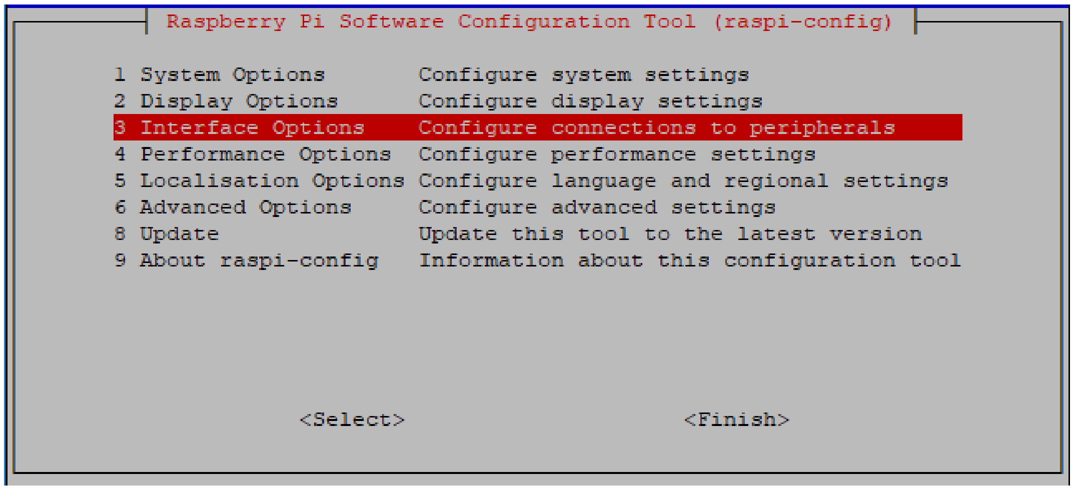
2.1.4 Enable I2C and Camera Interface

Here we are using the Raspberry Pi's I2C and Camera interfaces, but by default they are disabled, so we need to enable them first.

1. Input the following command:

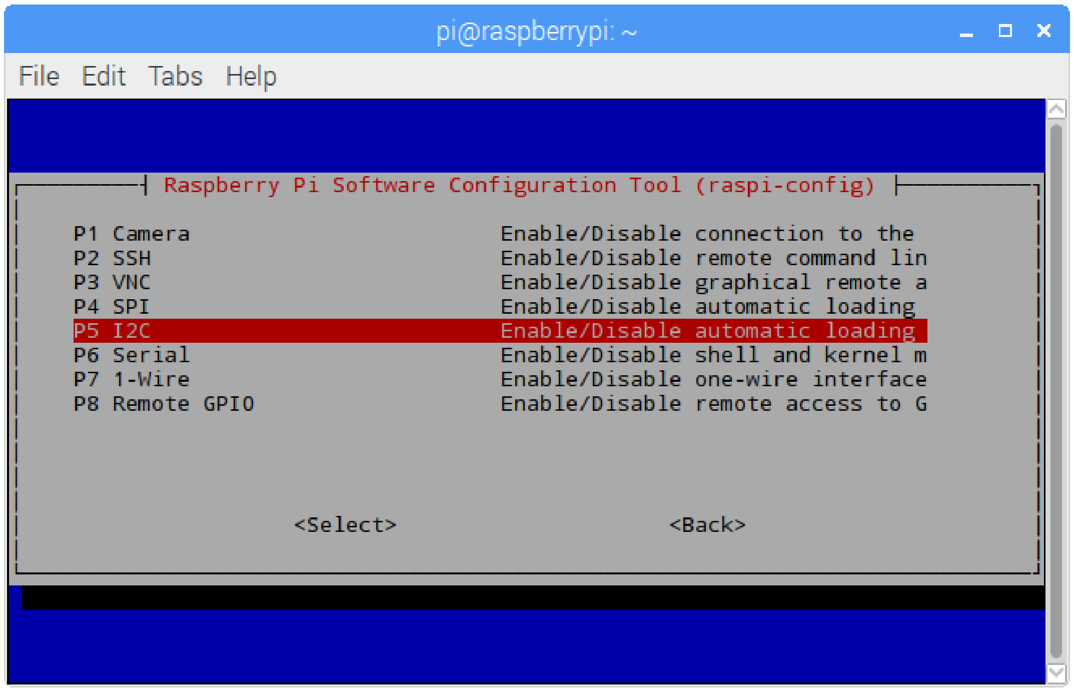
```
sudo raspi-config
```

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



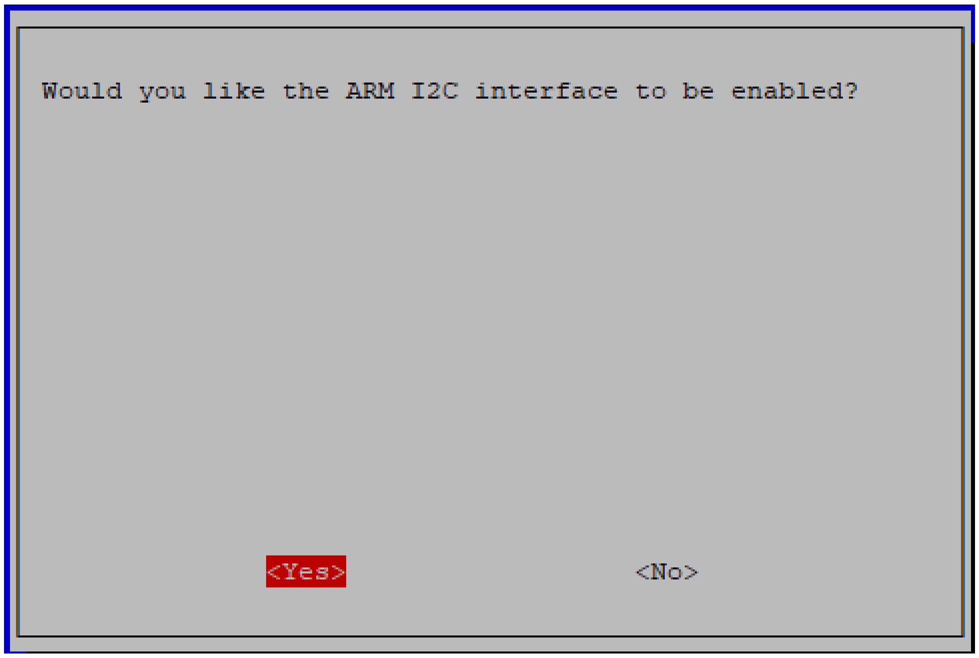
```
Raspberry Pi Software Configuration Tool (raspi-config)

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>
```

3. Then **P5 I2C**.



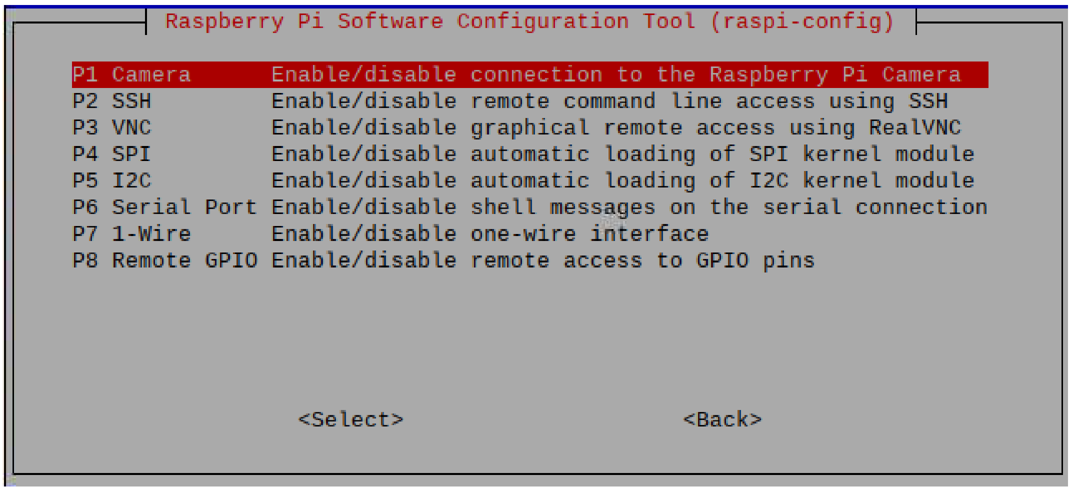
```
Raspberry Pi Software Configuration Tool (raspi-config)

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>
```

4. Use the arrow keys on the keyboard to select **<Yes>** -> **<OK>** to complete the setup of the I2C.



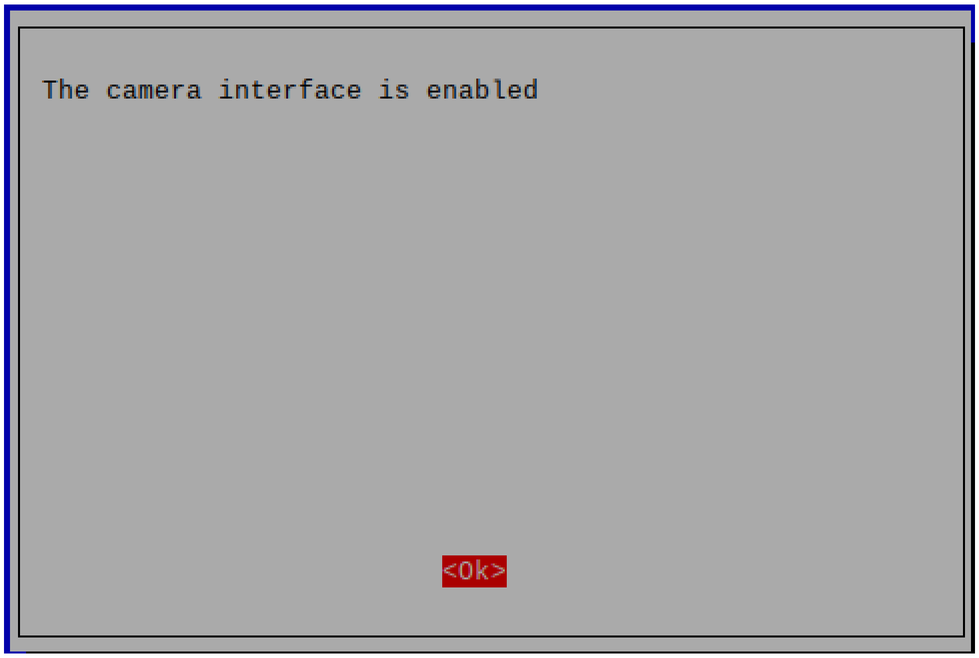
```
Would you like the ARM I2C interface to be enabled?

                <Yes>                          <No>
```

5. Go to **3 Interfacing Options** again and select **P1 Camera**.



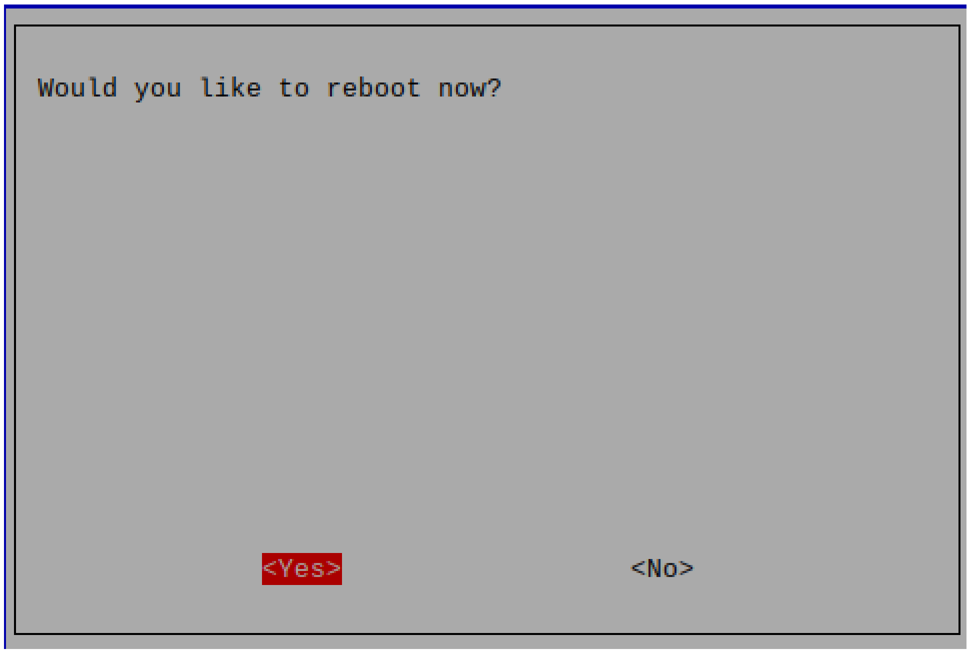
```
        Raspberry Pi Software Configuration Tool (raspi-config)

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>
```

6. Again select **<Yes>** -> **<OK>** to complete the setup.



```
The camera interface is enabled
```

7. After you select **<Finish>**, a pop-up will remind you that you need to reboot for the settings to take effect, select **<Yes>**.



```
Would you like to reboot now?
```

2.1.5 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
```

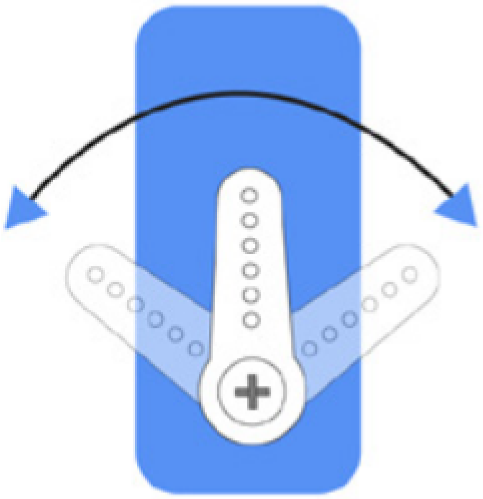
Note: Running `install.py` will download some necessary components. You may fail to download due to network problems. You may need to download again at this time.

Run `servo_zeroing.py`

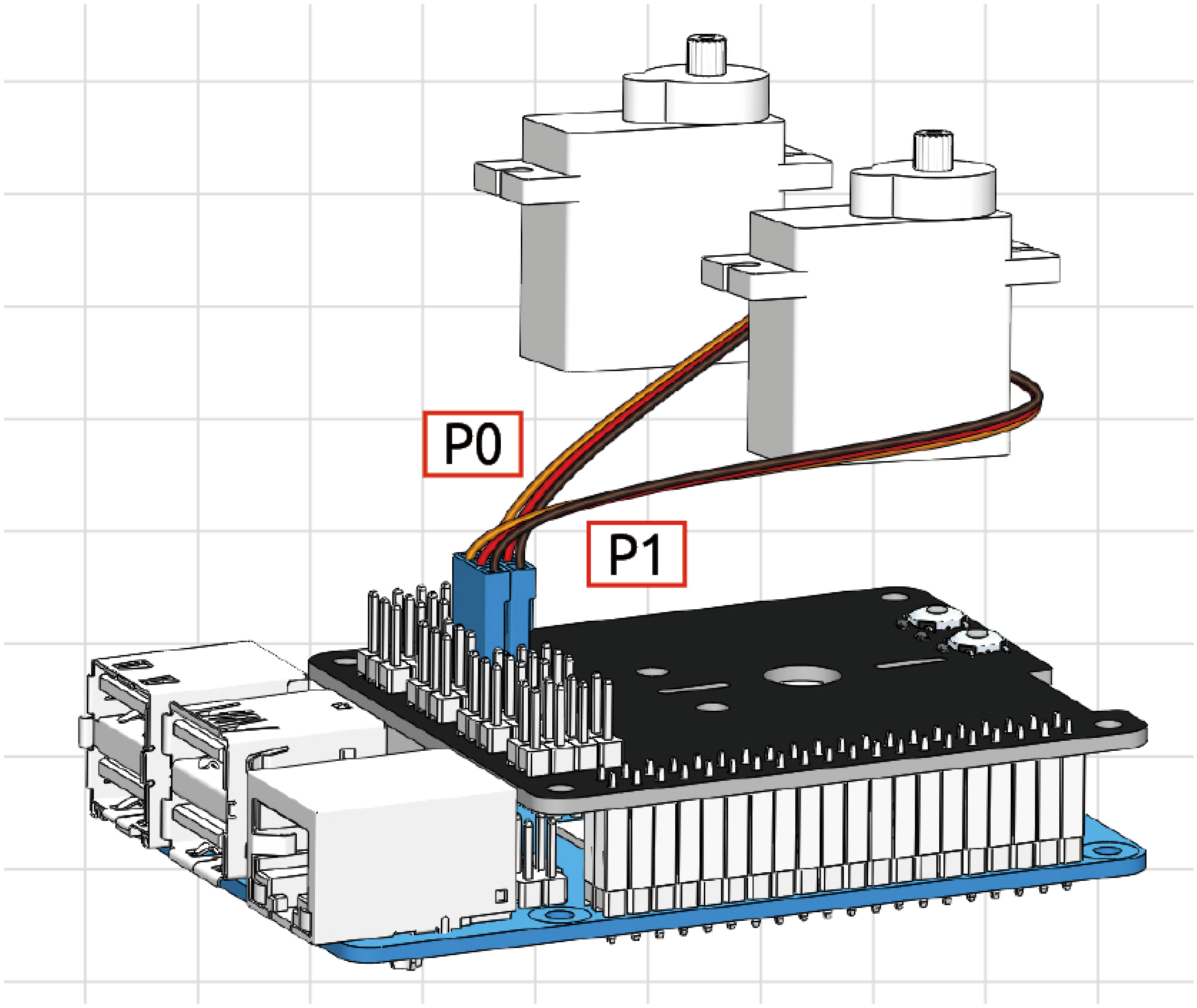
Run the `servo_zeroing.py` in the `examples/` folder.

```
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 12/13 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. Here you can use Pan-tilt HAT to take a photo and store it in `/home/pi/Pictures/vilib/`.

Run the Code

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

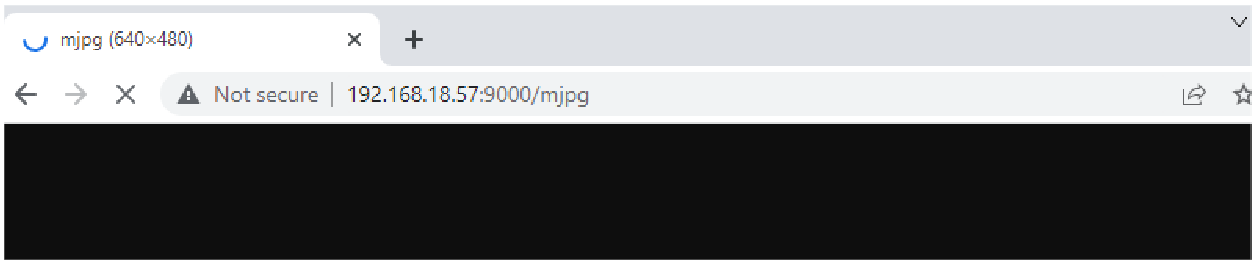
After the program runs: 1. You can enter `http://<Your Raspberry Pi IP>:9000/mjpg` in the browser (such as chrome) to view the viewfinder screen. 2. Type `q` in Terminal and press Enter to take a photo.

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

```
#!/usr/bin/env python3
from vilib import Vilib
import time

manual = '''
Press keys on keyboard to record value!
    Q: photo shoot
    G: Quit
'''

def main():
    path = "/home/pi/Pictures/vilib/"

    Vilib.camera_start(vflip=True,hflip=True)
    Vilib.display(local=True,web=True)
    time.sleep(2)

    print(manual)
    while True:
        try:
            key = input().lower()
            if key == "q":
                _time = time.strftime("%y-%m-%d_%H-%M-%S", time.localtime())
                Vilib.take_photo(photo_name=str(_time),path=path)
                print("The photo save as %s%s.jpg"%(path,_time))
            elif key == "g" :
                Vilib.camera_close()
                break
        except KeyboardInterrupt:
            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=str(_time),path=path)
```

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
```

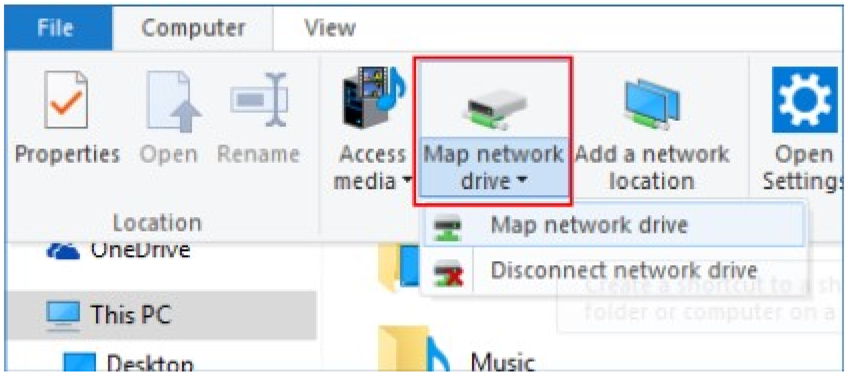
4. Add sharing account.

```
sudo smbpasswd -a pi
```

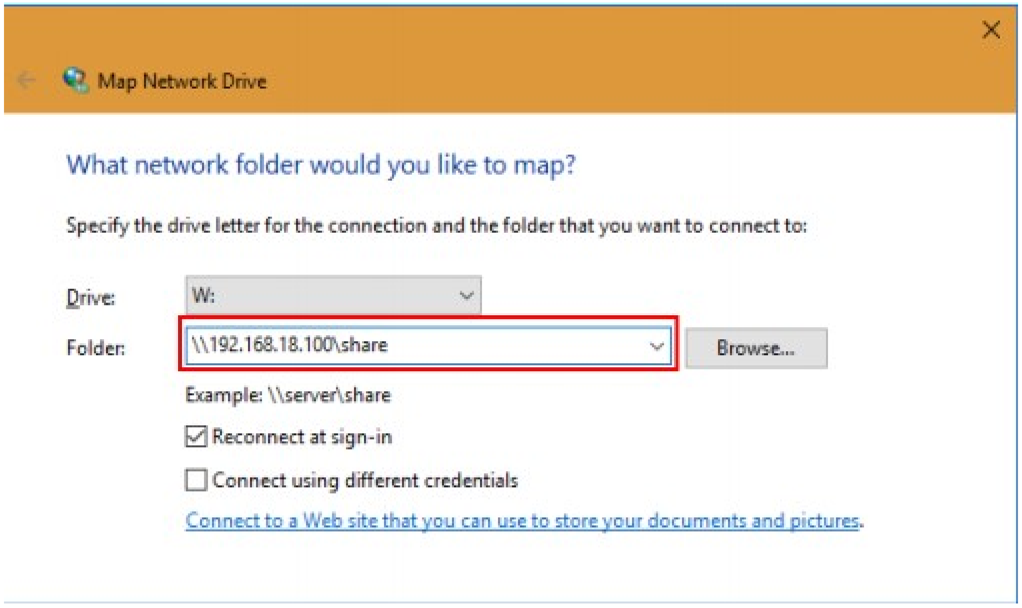
Note: A sharing account "pi" is created and you need to set your passcode.

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...

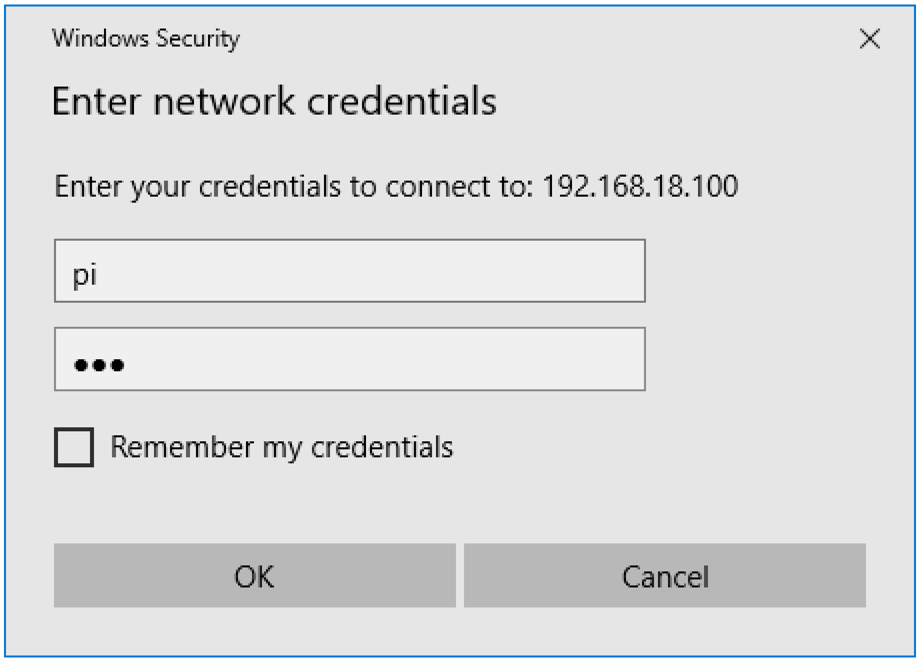
Example: \\server\share

☑ 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

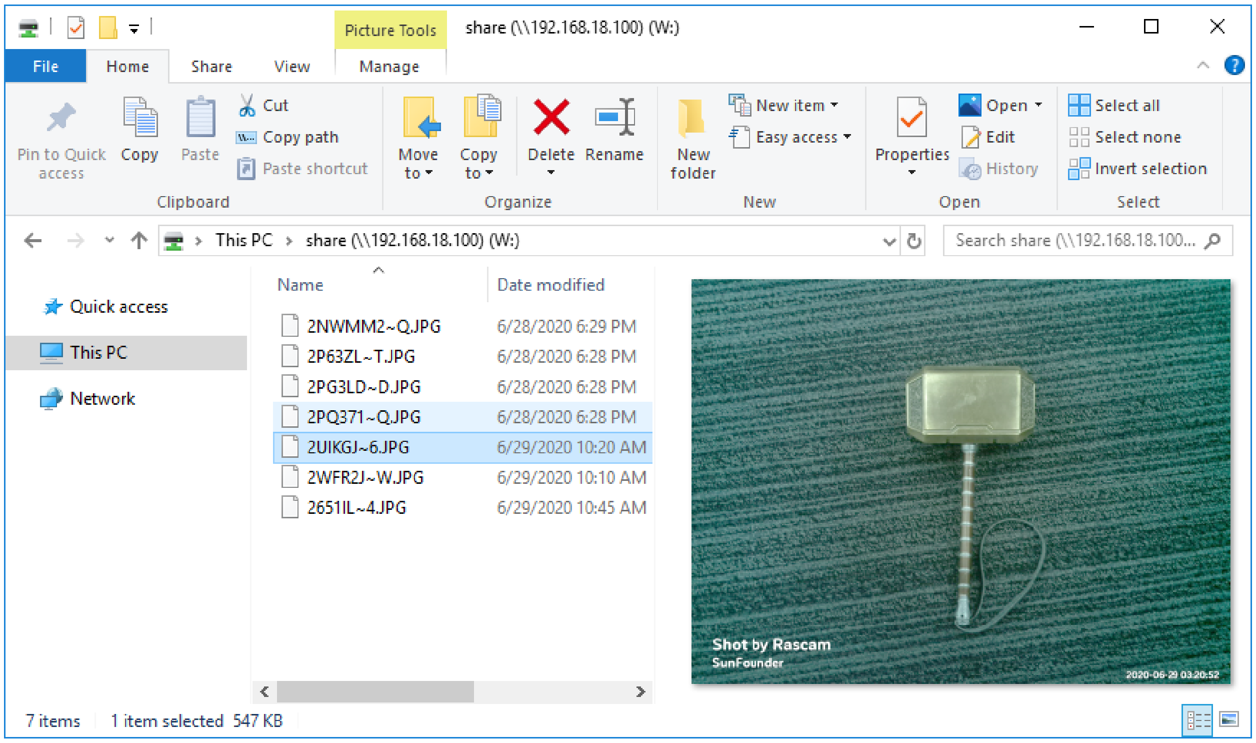
Enter your credentials to connect to: 192.168.18.100

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:)

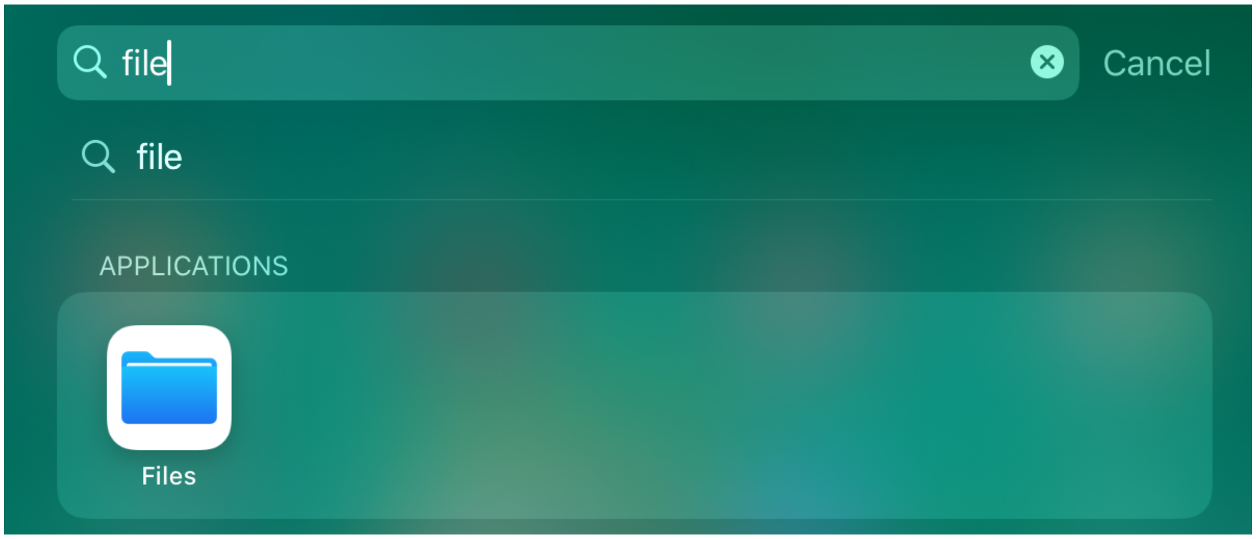
Name	Date modified
2NWMM2~Q.JPG	6/28/2020 6:29 PM
2P63ZL~T.JPG	6/28/2020 6:28 PM
2PG3LD~D.JPG	6/28/2020 6:28 PM
2PQ371~Q.JPG	6/28/2020 6:28 PM
2UIKGJ~6.JPG	6/29/2020 10:20 AM
2WFR2J~W.JPG	6/29/2020 10:10 AM
2651IL~4.JPG	6/29/2020 10:45 AM

7 items 1 item selected 547 KB

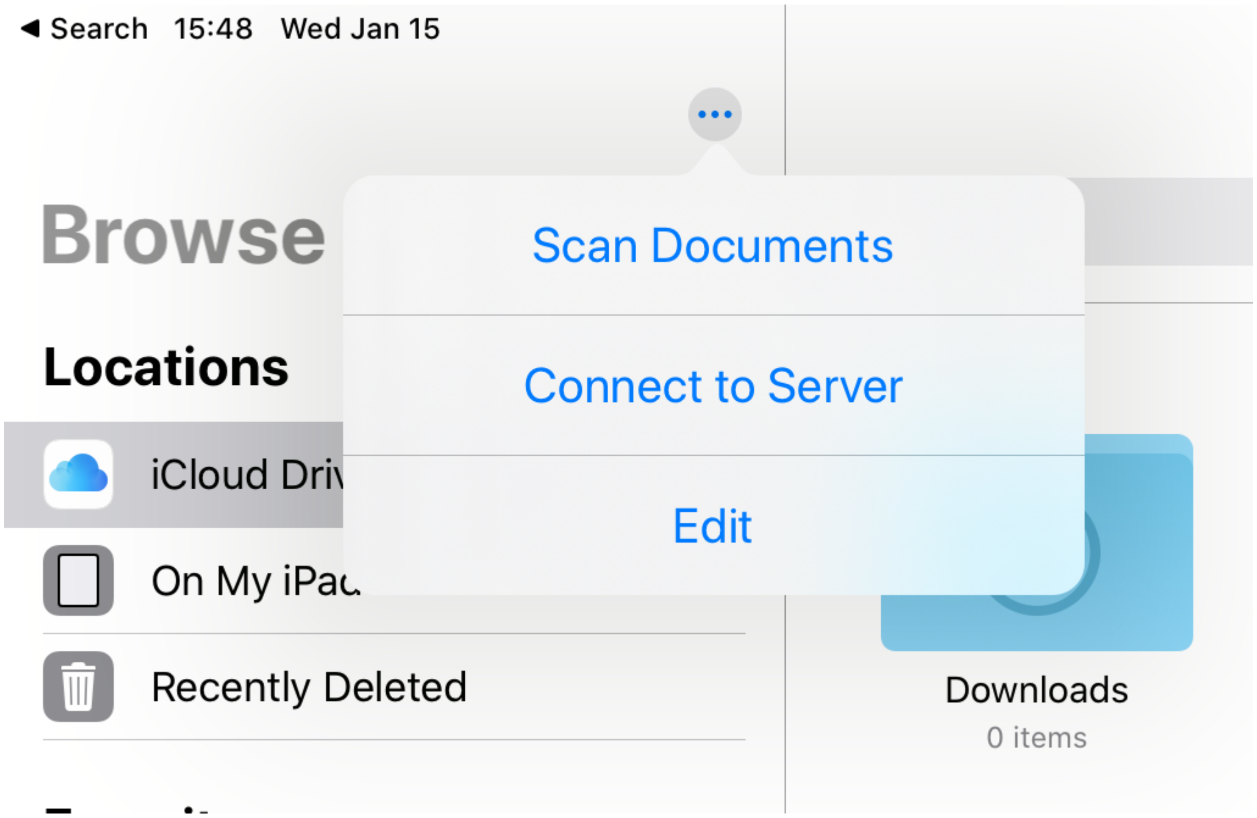
Mount Photo album to iOS

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

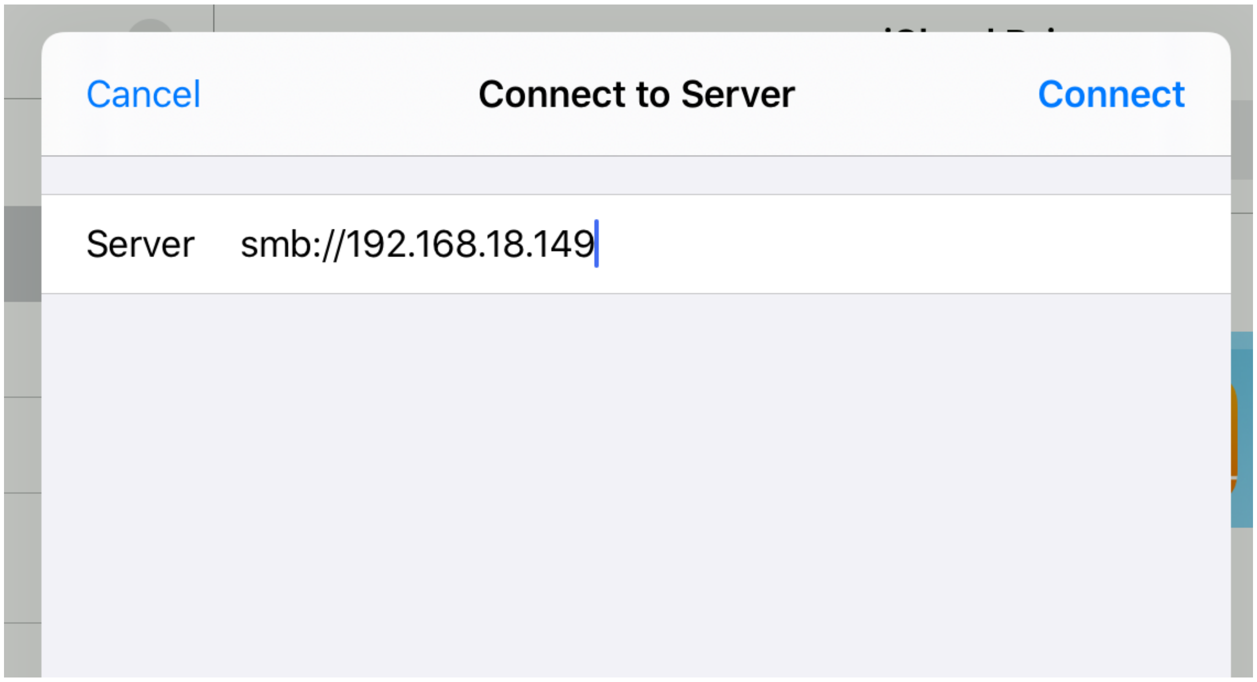
1. Open **Files**.



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

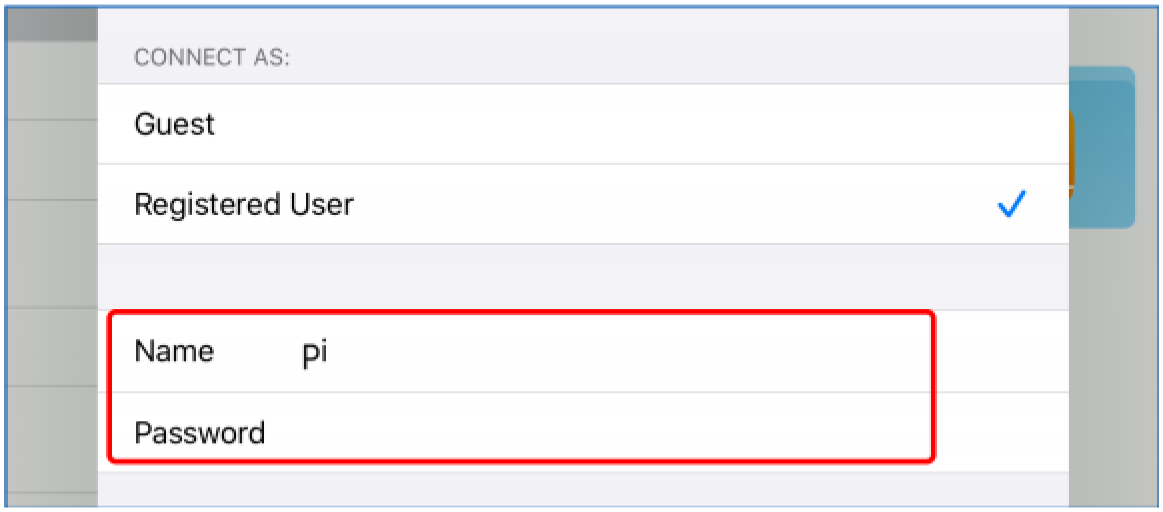


3. Enter your Rascam IP address.



Server smb://192.168.18.149

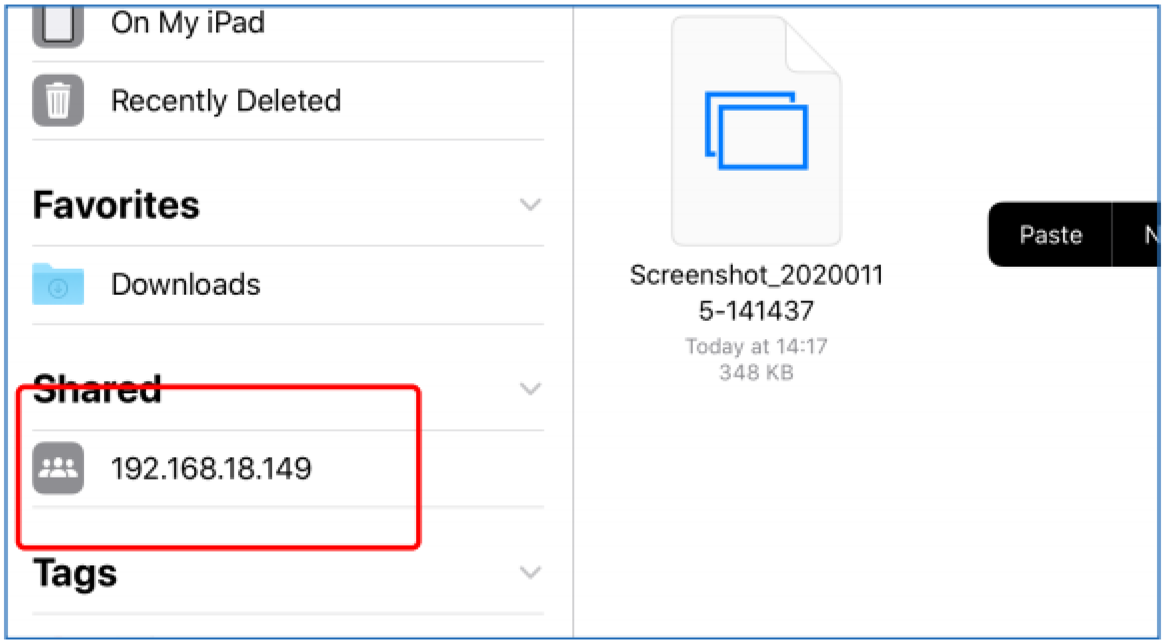
4. Log in.



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 is the web 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

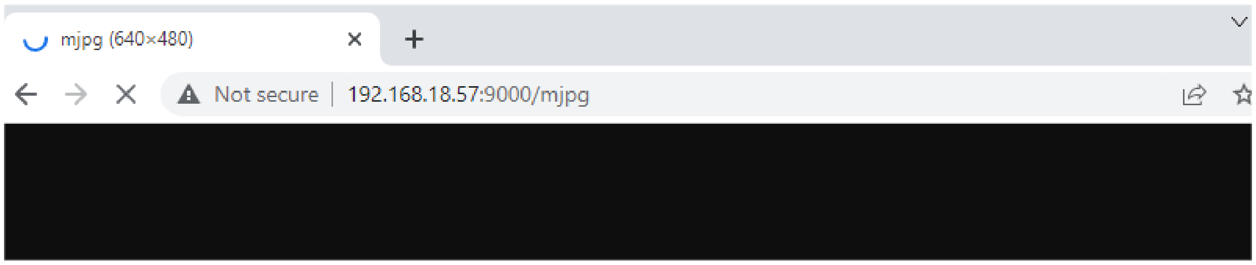
```
cd /home/pi/pan-tilt-hat/examples
sudo python3 continuous_shooting.py
```

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

```
#!/usr/bin/env python3
from time import sleep,strftime,localtime
from vilib import Vilib
import sys
sys.path.append('./')
from servo import Servo
import tty
import termios

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: down
    D: right
    Q: continuous_shooting
    G: Quit
'''

# region servos init
pan = Servo(pin=13, min_angle=-90, max_angle=90) # pan_servo_pin (BCM)
tilt = Servo(pin=12, min_angle=-90, max_angle=30) # be careful to limit the angle of␣
↪the steering gear
panAngle = 0
tiltAngle = 0
pan.set_angle(panAngle)
tilt.set_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, 30)
        tilt.set_angle(tiltAngle)
    if key == 's':
        tiltAngle += 1
        tiltAngle = limit(tiltAngle, -90, 30)
        tilt.set_angle(tiltAngle)
    if key == 'a':
        panAngle += 1
        panAngle = limit(panAngle, -90, 90)
        pan.set_angle(panAngle)
    if key == 'd':
        panAngle -= 1
        panAngle = limit(panAngle, -90, 90)
        pan.set_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,the pictures save as %s"%path)
    sleep(0.2)

def main():

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

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

    print(manual)
    while True:
        key = readchar().lower()
        servo_control(key)
        if key == 'q':
            continuous_shooting(path,interval_ms=50,number=10)
        elif key == 'g':
            Vilib.camera_close()
            break
        sleep(0.01)


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:

```python
import sys
import tty
import termios

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

def main():
    while True:
        key = readchar().lower()
        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:

```python
from time import sleep,strftime,localtime
from servo import Servo

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


# region servos init
pan = Servo(pin=13, min_angle=-90, max_angle=90) # pan_servo_pin (BCM)
tilt = Servo(pin=12, min_angle=-90, max_angle=30) # be careful to limit␣
↪the angle of the steering gear
panAngle = 0
tiltAngle = 0
pan.set_angle(panAngle)
tilt.set_angle(tiltAngle)
#endregion init

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

(continues on next page)

(continued from previous page)

```
        return min
    else:
        return x

def servo_control(key):
    global panAngle,tiltAngle
    if key == 'w':
        tiltAngle -= 1
        tiltAngle = limit(tiltAngle, -90, 30)
        tilt.set_angle(tiltAngle)
    if key == 's':
        tiltAngle += 1
        tiltAngle = limit(tiltAngle, -90, 30)
        tilt.set_angle(tiltAngle)
    if key == 'a':
        panAngle += 1
        panAngle = limit(panAngle, -90, 90)
        pan.set_angle(panAngle)
    if key == 'd':
        panAngle -= 1
        panAngle = limit(panAngle, -90, 90)
        pan.set_angle(panAngle)

# endregion

def main():
    while True:
        key = readchar().lower()
        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,strftime,localtime
from servo import Servo

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


# region servos init
pan = Servo(pin=13, min_angle=-90, max_angle=90) # pan_servo_pin (BCM)
tilt = Servo(pin=12, min_angle=-90, max_angle=30) # be careful to limit␣
↪the angle of the steering gear
panAngle = 0
tiltAngle = 0
pan.set_angle(panAngle)
tilt.set_angle(tiltAngle)
#endregion init
```

- And `tilt = Servo(pin=12, min_angle=-90, max_angle=30)` is used to init the servo object. Here, the servo connected to 12 is declared as an object named `tilt` .
- As for `tilt.set_angle(tiltAngle)` , it directly controls the tiltServo, which is the angle

of the servo connected to 12.

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.

```
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/1000)
    print("continuous_shooting done,the pictures save as %s"%path)
    sleep(0.2)

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().lower()
        #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

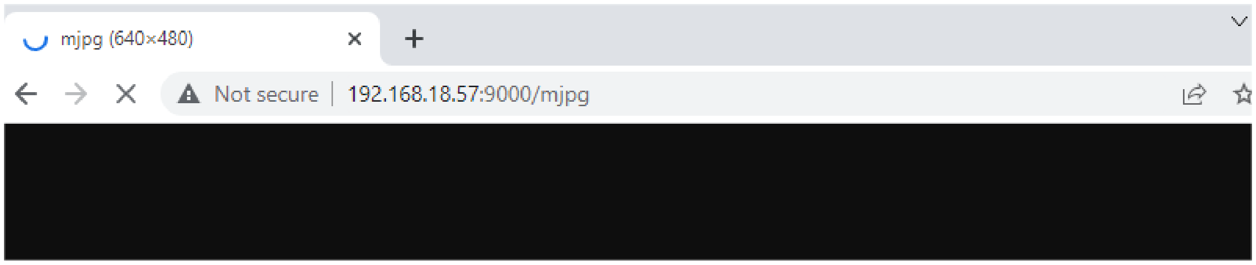
```
cd /home/pi/pan-tilt-hat/examples
sudo python3 time_lapse_photography
```

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

```
#!/usr/bin/env python3
'''
    Time-lapse photography based on the Raspistill command
'''
from time import time, sleep, strftime, localtime
from vilib import Vilib
import os
import sys
sys.path.append('./')
from servo import Servo
# import readchar
import cv2
import threading
import tty
import termios
```

(continues on next page)

(continued from previous page)

```
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
'''
# endregion

# region servos init
pan = Servo(pin=13, min_angle=-90, max_angle=90) # pan_servo_pin (BCM)
tilt = Servo(pin=12, min_angle=-90, max_angle=30) # be careful to limit the angle of␣
↪the steering gear
panAngle = 0
tiltAngle = 0
pan.set_angle(panAngle)
tilt.set_angle(tiltAngle)

#endregion init

# # 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
```

(continues on next page)

(continued from previous page)

```
        tiltAngle = limit(tiltAngle, -90, 30)
        tilt.set_angle(tiltAngle)
    if key == 's':
        tiltAngle += 1
        tiltAngle = limit(tiltAngle, -90, 30)
        tilt.set_angle(tiltAngle)
    if key == 'a':
        panAngle += 1
        panAngle = limit(panAngle, -90, 90)
        pan.set_angle(panAngle)
    if key == 'd':
        panAngle -= 1
        panAngle = limit(panAngle, -90, 90)
        pan.set_angle(panAngle)

# endregion servo control

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

    print('\nprocessing video, please wait ....')

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

    # traverse

    for root, dirs, files in os.walk(output):
        print('%s pictures need to 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(output+'/'+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
```

(continues on next page)

(continued from previous page)

```
                out.write(frame)

    # release the VideoWriter object
    out.release()
    # remove photos cache
    os.system('sudo rm -r %s'%output)
    print('\nDone.The video save as %s/%s'%(path,name))

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

# continuous_shooting
def continuous_shooting(path, interval_s=3, duration_s=3600):
    print('\nStart time-lapse photography, press the "e" key to stop')

    start_time = time()
    node_time = start_time

    while True:

        if time()-node_time > interval_s:
            node_time = time()
            Vilib.take_photo(photo_name=strftime("%Y-%m-%d-%H-%M-%S", localtime()),
↪path=path)
        if key == 'e' or time()-start_time > duration_s:
            break
        sleep(0.01) # second


# main
def main():
    global key


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

    sleep(2)
    print(manual)
    sleep(0.2)
    t = threading.Thread(target=keyboard_scan)
    t.setDaemon(True)
    t.start()

    path = "/home/pi/Videos/vilib/time_lapse"
    check_dir(path)

    while True:
```

(continues on next page)

(continued from previous page)

```
        servo_control(key)

        # time-lapse photography
        if key == 'q':
            # check path
            output = path+'/'+strftime("%Y-%m-%d-%H-%M-%S", localtime())
            check_dir(output)
            # take a picture every 3 seconds for 3600 seconds
            continuous_shooting(output, interval_s=3, duration_s=3600)
            # video_synthesis
            name=strftime("%Y-%m-%d-%H-%M-%S", localtime())
            video_synthesis(name=name,
                            output=output,
                            path=path,
                            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
        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
```

(continues on next page)

(continued from previous page)

```
# 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().lower()
        sleep(0.01)
        if breakout_flag==True:
            break

# main
def main():

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

    while True:
        # 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
```

(continues on next page)

(continued from previous page)

```
def main():
    path = "/home/pi/Videos/vilib/time_lapse"
    check_dir(path)

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

            # take_photo
            # video_synthesis


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

The target directory for our output videos is `path`. And generating video requires a large number of temporary still photos, which are stored in `output`. 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 time, sleep, strftime, localtime
from vilib import Vilib

# continuous_shooting
def continuous_shooting(path, interval_s=3, duration_s=3600):
    print('\nStart time-lapse photography, press the "e" key to stop')

    start_time = time()
    node_time = start_time

    while True:

        if time()-node_time > interval_s:
            node_time = time()
            Vilib.take_photo(photo_name=strftime("%Y-%m-%d-%H-%M-%S",␣
↪localtime()),path=path)
        if key == 'e' or time()-start_time > duration_s:
            break
        sleep(0.01) # second

# main
def main():

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

    while True:
        # time-lapse photography
```

(continues on next page)

(continued from previous page)

```
        if key == 'q':
            #check path

            # take a picture every 3 seconds for 3600 seconds
            continuous_shooting(output, interval_s=3, duration_s=3600)

            # 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 `output` path as frames, and generates a video output to the `path`.

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

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

    print('\nprocessing video, please wait ....')

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

    # traverse

    for root, dirs, files in os.walk(output):
        print('%s pictures need to 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(output+'/'+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]
```

(continues on next page)

(continued from previous page)

```
                     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 cache
    os.system('sudo rm -r %s'%output)
    print('\nDone.The video save as %s/%s'%(path,name))

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

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


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('-')
```

(continues on next page)

(continued from previous page)

```
                    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 cache
os.system('sudo rm -r %s'%output)
print('\nDone.The video save as %s/%s'%(path,name))
```

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

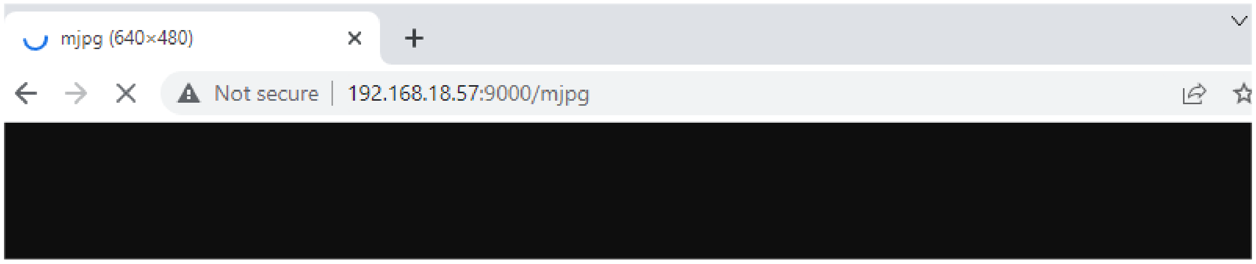
```
cd /home/pi/pan-tilt-hat/examples
sudo python3 record_video.py
```

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

```
#!/usr/bin/env python3
from time import sleep,strftime,localtime
from vilib import Vilib
import sys
sys.path.append('./')
from servo import Servo

import tty
import termios

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: record/pause/continue
    E: stop
    G: Quit
'''

# region init
pan = Servo(pin=13, min_angle=-90, max_angle=90) # pan_servo_pin (BCM)
tilt = Servo(pin=12, min_angle=-90, max_angle=30) # be careful to limit the angle of␣
↪the steering gear
panAngle = 0
tiltAngle = 0
pan.set_angle(panAngle)
tilt.set_angle(tiltAngle)

Vilib.rec_video_set["path"] = "/home/pi/Videos/vilib/" # set path
vname = None
rec_flag = 'stop' # start,pause,stop
#endregion init
```

(continues on next page)

(continued from previous page)

```
# rec control
def rec_control(key):
    global rec_flag, vname

    # start,pause
    if key == 'q':
        key = None
        if rec_flag == 'stop':
            rec_flag = 'start'
            # set name
            vname = strftime("%Y-%m-%d-%H.%M.%S", localtime())
            Vilib.rec_video_set["name"] = vname
            # start record
            Vilib.rec_video_run()
            Vilib.rec_video_start()
            print('rec start ...')
        elif rec_flag == 'start':
            rec_flag = 'pause'
            Vilib.rec_video_pause()
            print('pause')
        elif rec_flag == 'pause':
            rec_flag = 'start'
            Vilib.rec_video_start()
            print('continue')
    # stop
    elif key == 'e' and rec_flag != 'stop':
        key = None
        rec_flag = 'stop'
        Vilib.rec_video_stop()
        print("The video saved as %s%s.avi"%(Vilib.rec_video_set["path"],vname),end='\
↪n')

# 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, 30)
        tilt.set_angle(tiltAngle)
    if key == 's':
        tiltAngle += 1
        tiltAngle = limit(tiltAngle, -90, 30)
        tilt.set_angle(tiltAngle)
    if key == 'a':
        panAngle += 1
        panAngle = limit(panAngle, -90, 90)
        pan.set_angle(panAngle)
    if key == 'd':
```

(continues on next page)

(continued from previous page)

```
        panAngle -= 1
        panAngle = limit(panAngle, -90, 90)
        pan.set_angle(panAngle)

# endregion servo control


def main():

    Vilib.camera_start(vflip=True,hflip=True)
    Vilib.display(local=True,web=True)
    sleep(2)
    print(manual)
    while True:
        key = readchar().lower()
        # 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:

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

# region init
Vilib.rec_video_set["path"] = "/home/pi/Videos/vilib/" # set path
vname = None
rec_flag = 'stop' # start,pause,stop
# endregion init

# rec control
def rec_control(key):
    global rec_flag, vname

    # start,pause
    if key == 'q':
        key = None
        if rec_flag == 'stop':
```

(continues on next page)

(continued from previous page)

```
            rec_flag = 'start'
            # set name
            vname = strftime("%Y-%m-%d-%H.%M.%S", localtime())
            Vilib.rec_video_set["name"] = vname
            # start record
            Vilib.rec_video_run()
            Vilib.rec_video_start()
            print('rec start ...')
        elif rec_flag == 'start':
            rec_flag = 'pause'
            Vilib.rec_video_pause()
            print('pause')
        elif rec_flag == 'pause':
            rec_flag = 'start'
            Vilib.rec_video_start()
            print('continue')
    # stop
    elif key == 'e' and rec_flag != 'stop':
        key = None
        rec_flag = 'stop'
        Vilib.rec_video_stop()
        print("The video saved as %s%s.avi"%(Vilib.rec_video_set["path"],vname),end='\
↪n')


def main():

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

    while True:
        key = readchar().lower()
        # rec control
        rec_control(key)
        # servo control

        # esc
        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

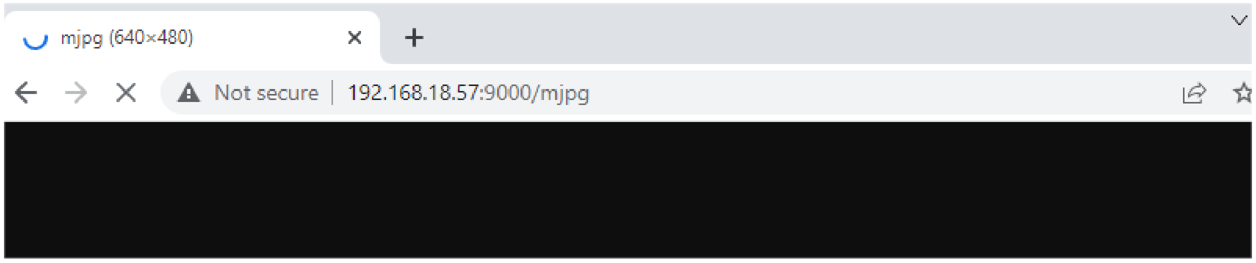
```
cd /home/pi/pan-tilt-hat/examples
sudo python3 panorama.py
```

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

```
#!/usr/bin/env python3
from time import sleep,strftime,localtime
from vilib import Vilib
import sys
sys.path.append('./')
from servo import Servo
import cv2
```

(continues on next page)

(continued from previous page)

```
import os

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 panoramic 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
pan = Servo(pin=13, min_angle=-90, max_angle=90) # pan_servo_pin (BCM)
tilt = Servo(pin=12, min_angle=-90, max_angle=30) # be careful to limit the angle of␣
↪the steering gear
panAngle = 0
tiltAngle = 0
pan.set_angle(panAngle)
tilt.set_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

    temp_path = "/home/pi/Pictures/vilib/panorama/.temp/"
    imgs =[]

    # check path
    check_dir(path)
    check_dir(temp_path)
```

(continues on next page)

(continued from previous page)

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

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

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

# main

def main():
    path = "/home/pi/Pictures/vilib/panorama"
    Vilib.camera_start(vflip=True,hflip=True)
    Vilib.display(local=True,web=True)
    sleep(2)

    print(manual)
    while True:
        key = readchar()
        # take photo
        if key == 'q':
            print("panorama shooting ...")
            panorama_shooting(path)
        # esc
        if key == 'g':
            print('Quit')
```

(continues on next page)

(continued from previous page)

```
            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()`.
2. Photograph:

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

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

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))

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)
```

(continues on next page)

(continued from previous page)

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

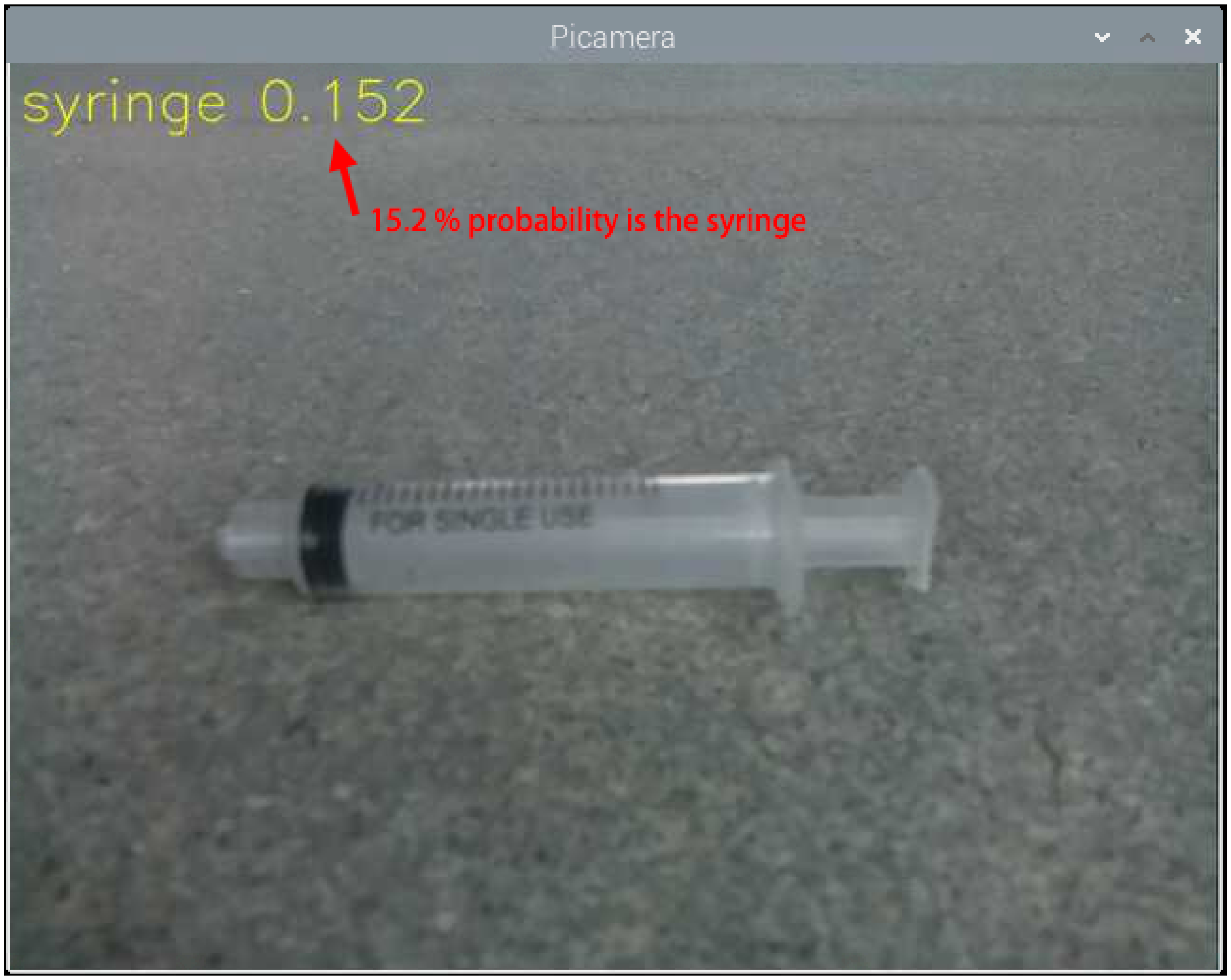
These photos are added to an array `imgs`, and then call OpenCV's Stitcher module (ie `status, pano = stitcher.stitch(imgs)`) to merge them into a panorama `pano`. Finally, use `cv2.imwrite()` to write `pano` into the storage space, and delete the temporary photos and their paths.

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

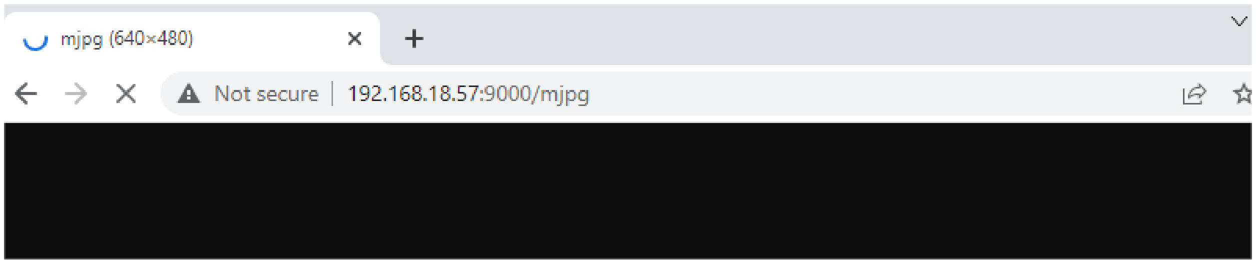
```
cd /home/pi/pan-tilt-hat/examples
sudo python3 image_classification.py
```

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

```python
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.

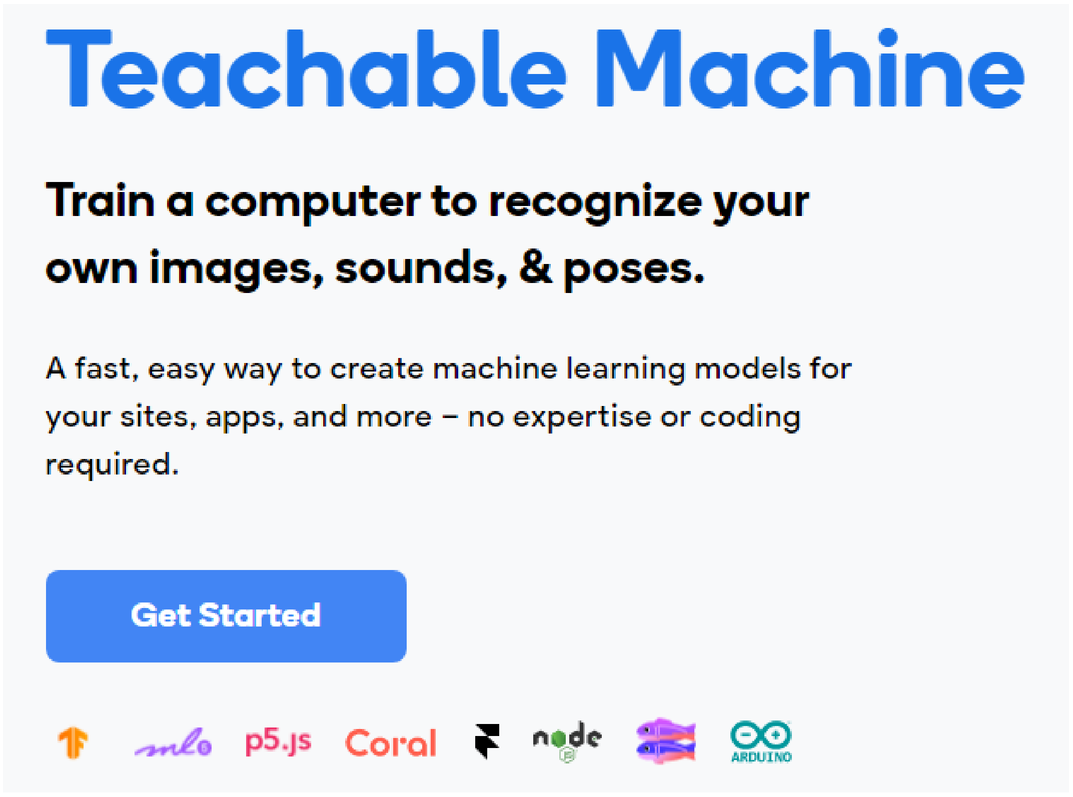
In addition to the built-in models in this article, you can also download the pre-trained image classification model on TensorFlow Hub. It should be noted that these models may not be suitable for your project, please use them as appropriate.

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.

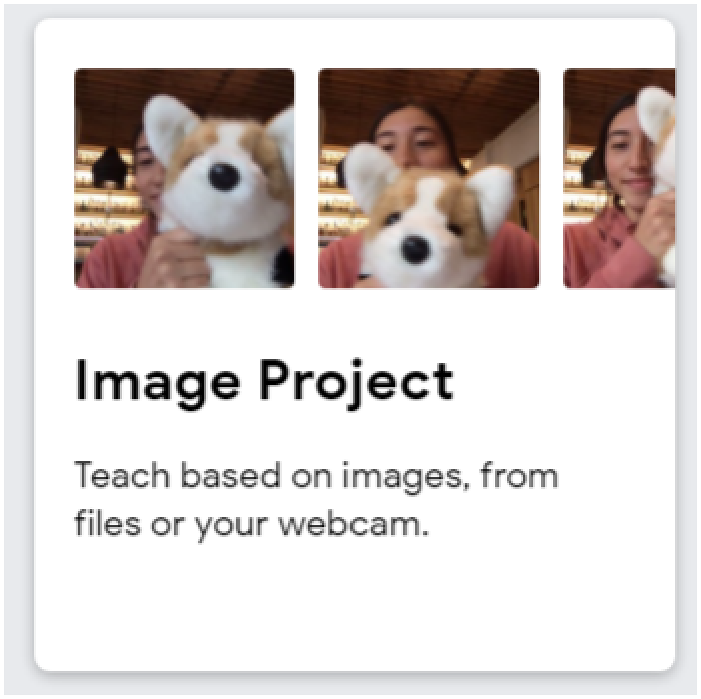
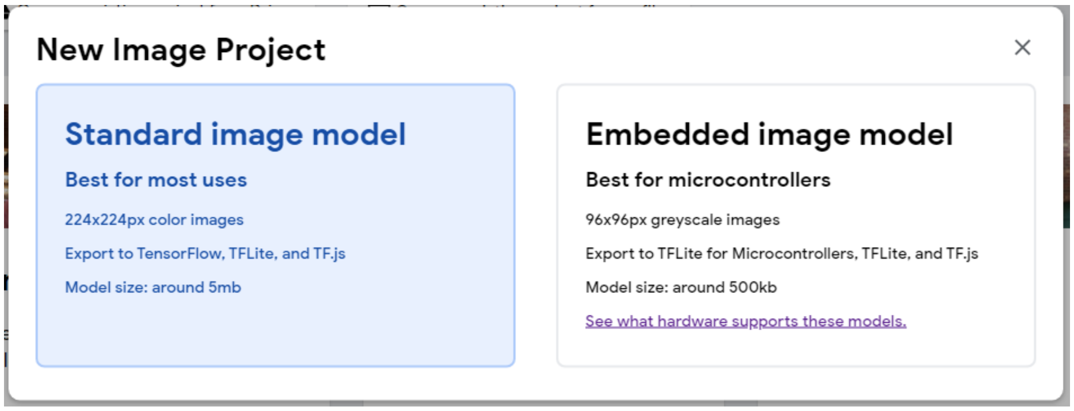


Image Project

Teach based on images, from files or your webcam.



New Image Project

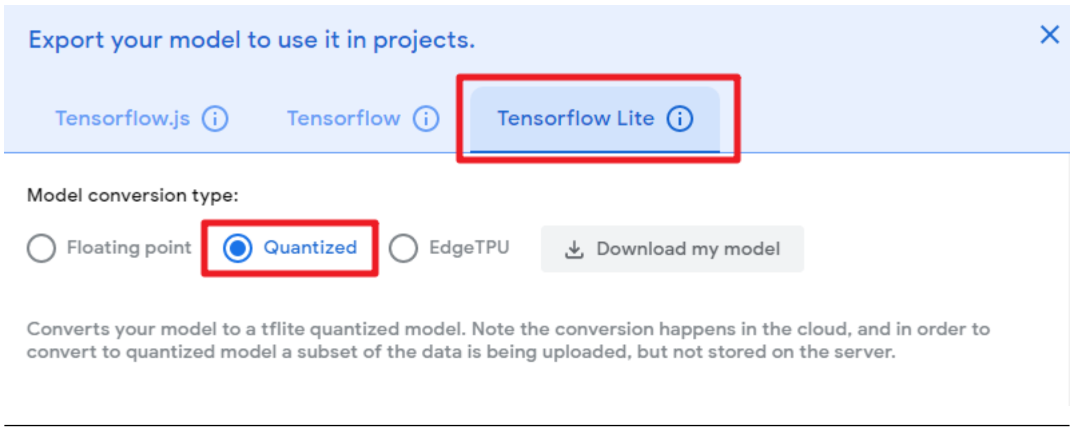
Standard image model
Best for most uses
224x224px color images
Export to TensorFlow, TFLite, and TF.js
Model size: around 5mb

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.

Model conversion type:

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.

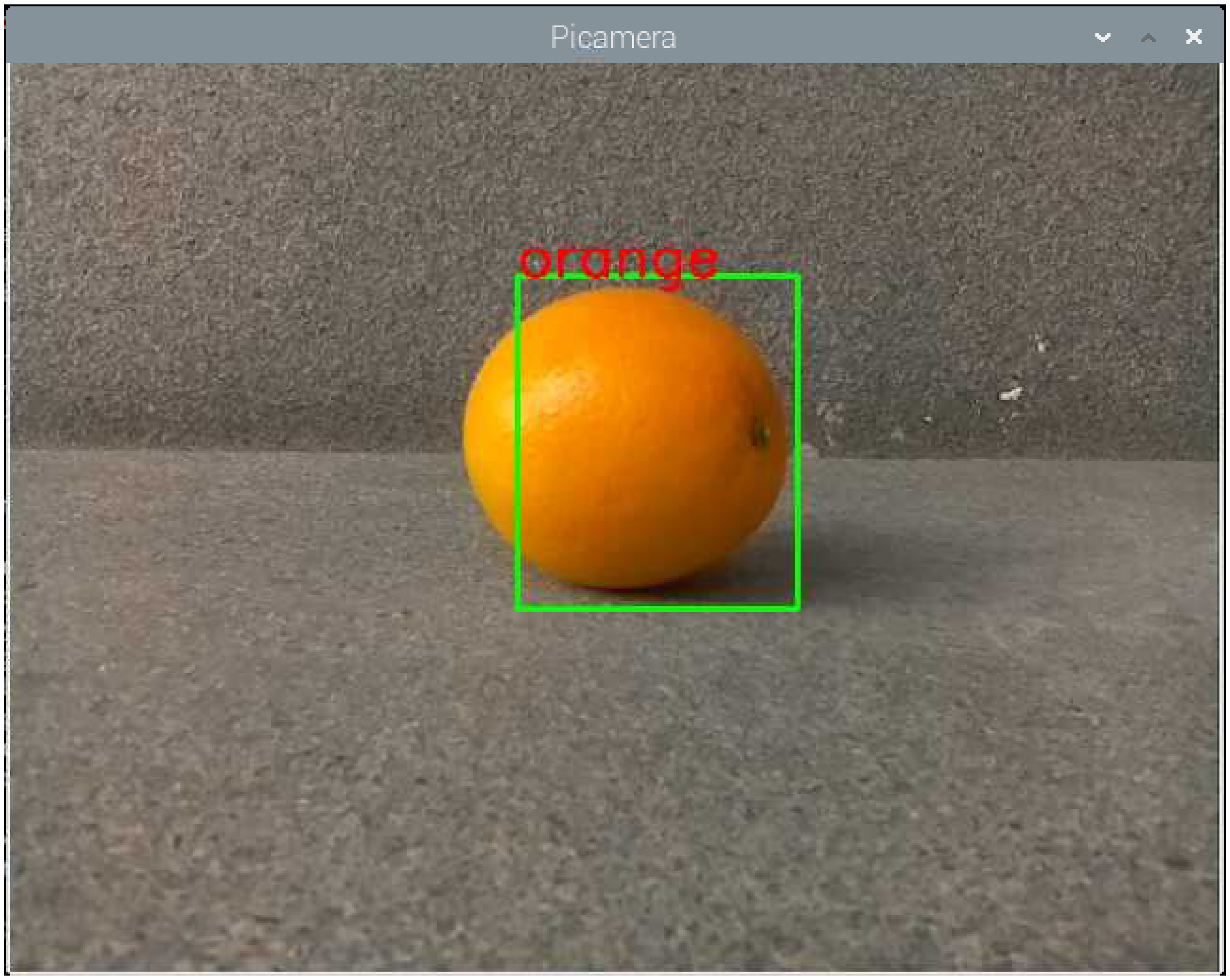
```
Vilib.image_classify_set_model(path='/home/pi/pan-tilt-hat/models/your_
↪model.tflite')
Vilib.image_classify_set_labels(path='/home/pi/pan-tilt-hat/models/your_
↪label.txt')
```

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

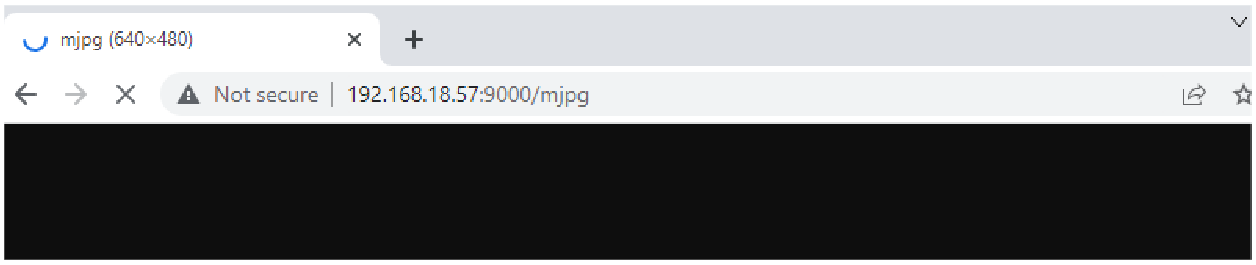
```
cd /home/pi/pan-tilt-hat/examples
sudo python3 color_detection.py
```

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

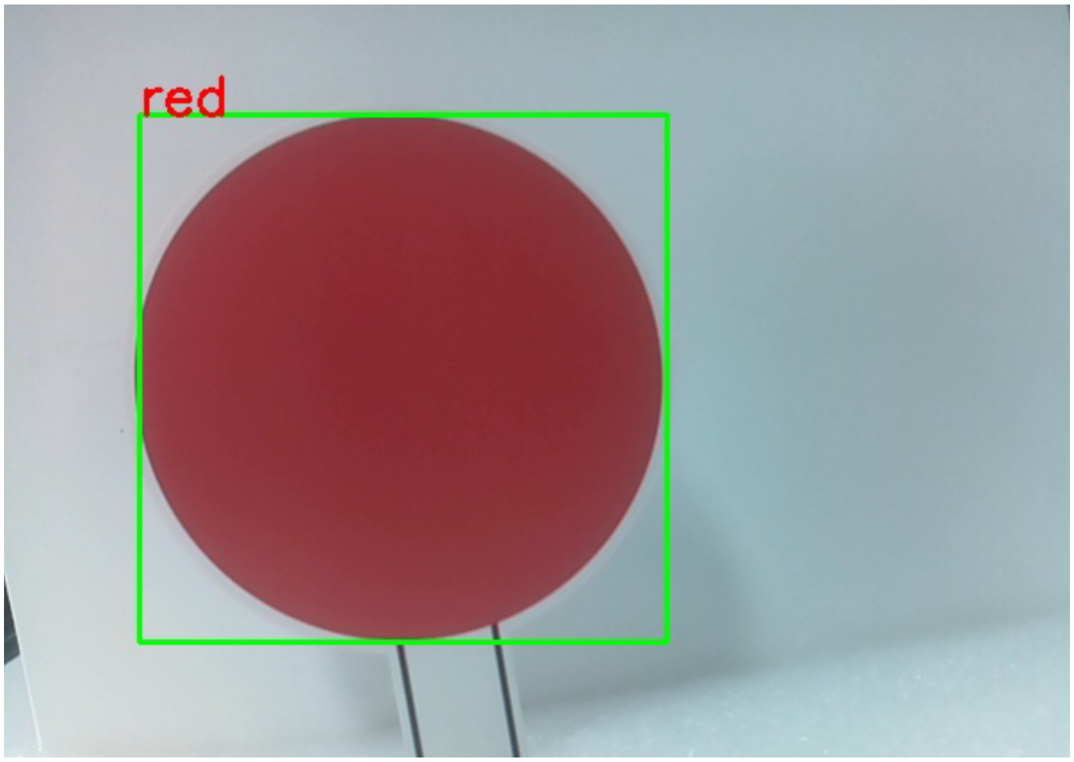
After the program runs, you will see the following information in the final:

```
Input key to call the function!
    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
    Q: Photo shoot
    G: Quitn
```

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

```
#!/usr/bin/env python3
from vilib import Vilib
import time

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

manual = '''
Input key to call the function!
    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
    Q: Photo shoot
    G: Quit
'''

def color_detect(color):
    if color == 'close':
        print("Color detect off!")
        Vilib.color_detect_switch(False)
    else:
        print("detecting color :" + color)
        Vilib.color_detect(color)


def show_info():
    if color_flag == 'close':
        print("Color detection is turned off !")
    else:
        if 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)
        else:
            print("No %s detected!"%color_flag)


def main():
    global color_flag
    path = "/home/pi/Pictures/vilib/color_detection/"
```

(continues on next page)

(continued from previous page)

```
    Vilib.camera_start(vflip=True,hflip=True)
    Vilib.display(local=True,web=True)
    time.sleep(2)

    print(manual)
    while True:
        try:
            key = input().lower()
            if key in ['0', '1', '2', '3', '4', '5', '6']:
                color_flag = color_list[int(key)]
                color_detect(color_flag)
            elif key == "s":
                show_info()
            elif key == 'q':
                _time = time.strftime("%y-%m-%d_%H-%M-%S", time.localtime())
                Vilib.take_photo(photo_name=str(_time),path=path)
                print("The photo save as %s%s.jpg"%(path,_time))
            elif key == "g" :
                Vilib.camera_close()
                break
        except KeyboardInterrupt:
            Vilib.camera_close()
            break


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']
```

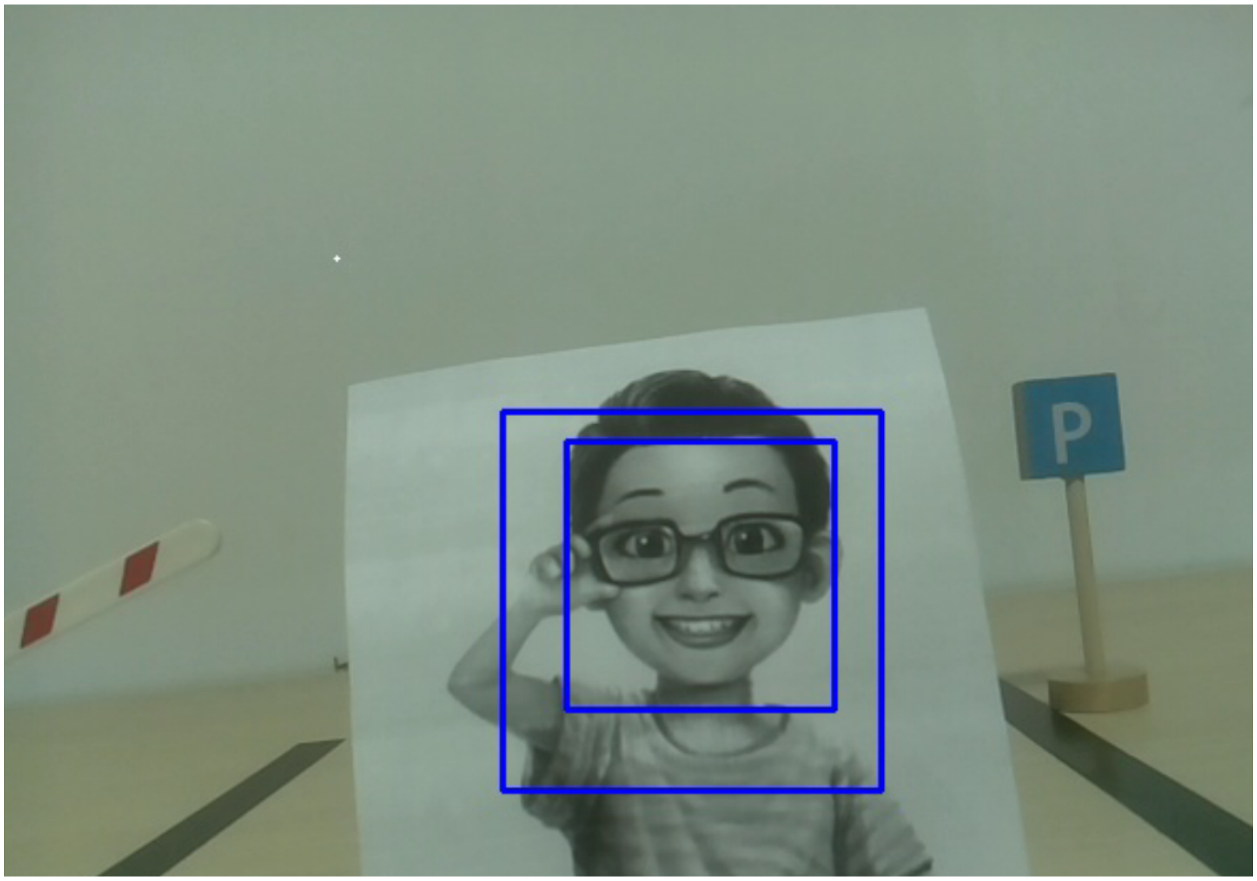
The keys of the dictionary and their uses are shown in the following list:

- `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

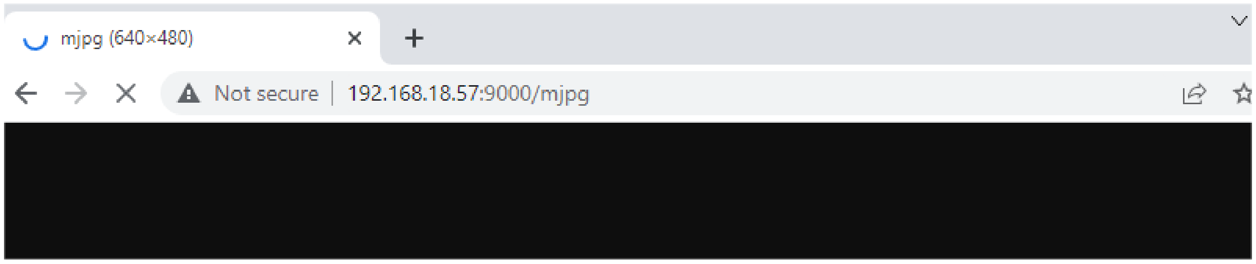
```
cd /home/pi/pan-tilt-hat/examples
sudo python3 face_detection.py
```

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

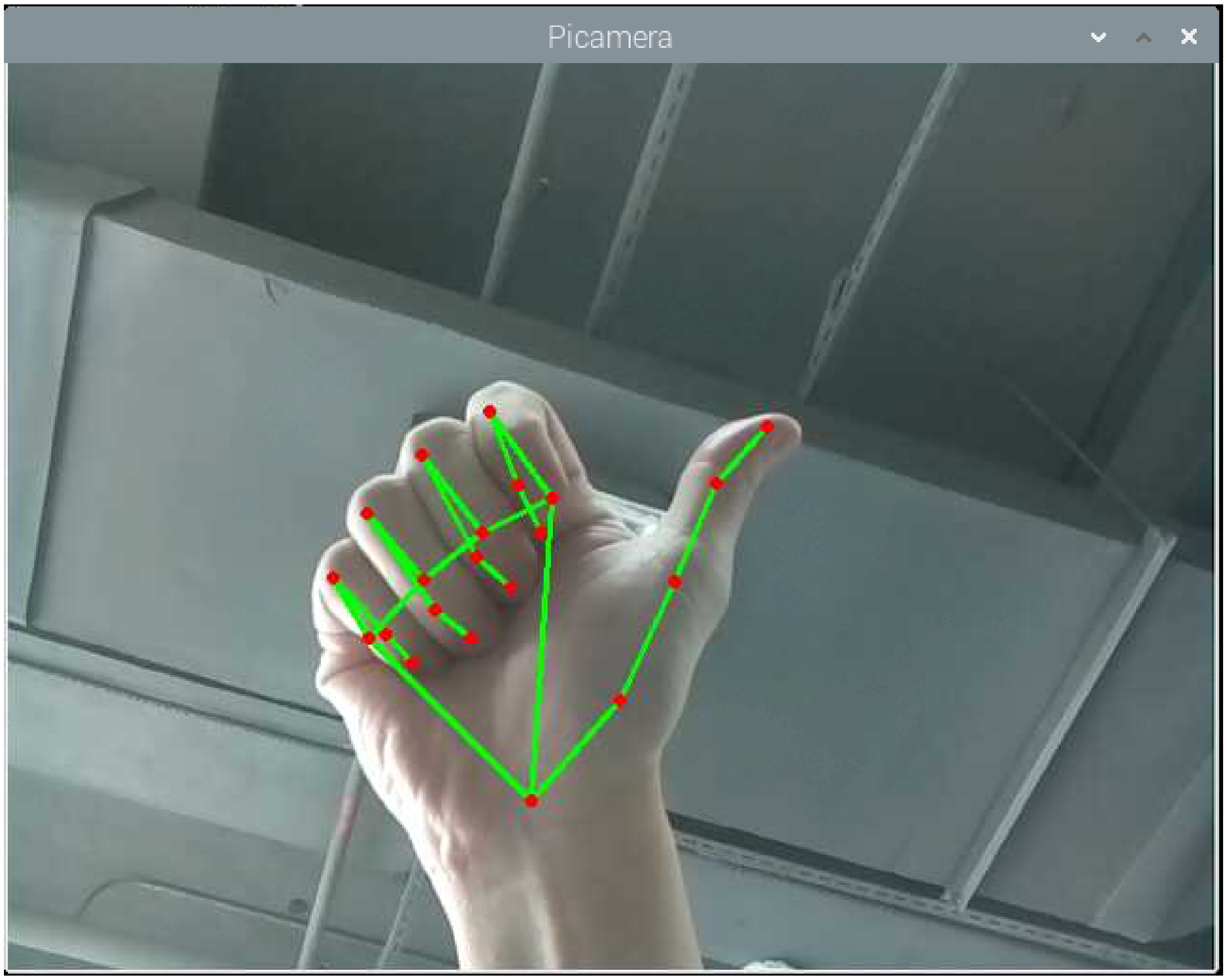
The keys of the dictionary and their uses are shown in the following list:

- `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:

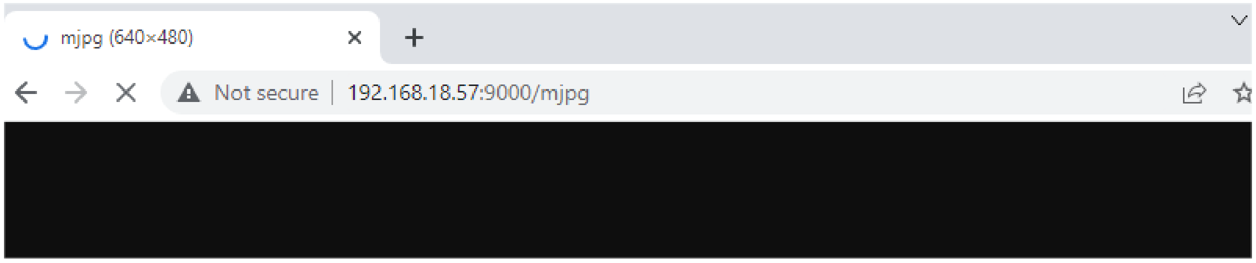
```
No desktop !
* Serving Flask app "vilib.vilib" (lazy loading)
* Environment: production
WARNING: Do not use the development server in a production environment.
Use a production WSGI server instead.
```

(continues on next page)

(continued from previous page)

```
* 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

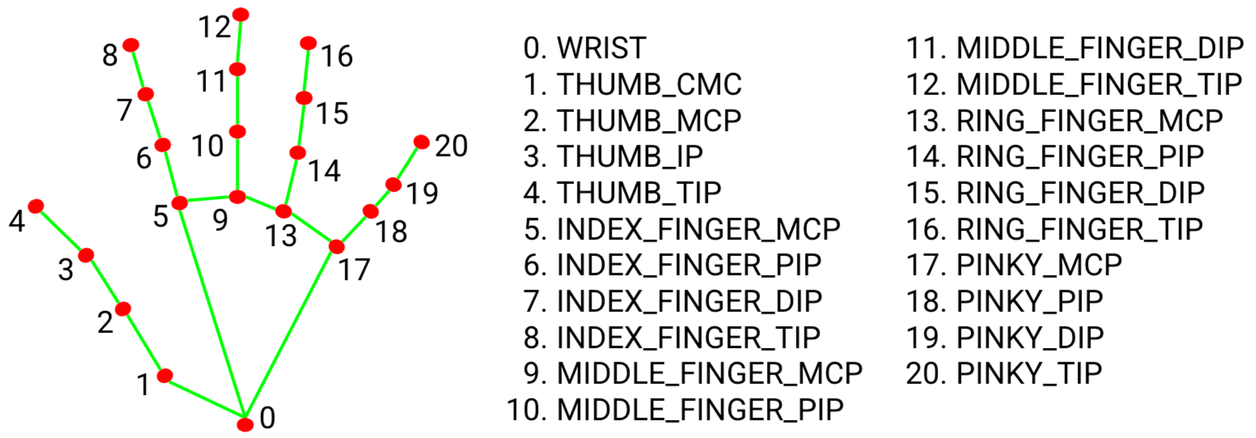
```python
from vilib import Vilib
from time import sleep

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

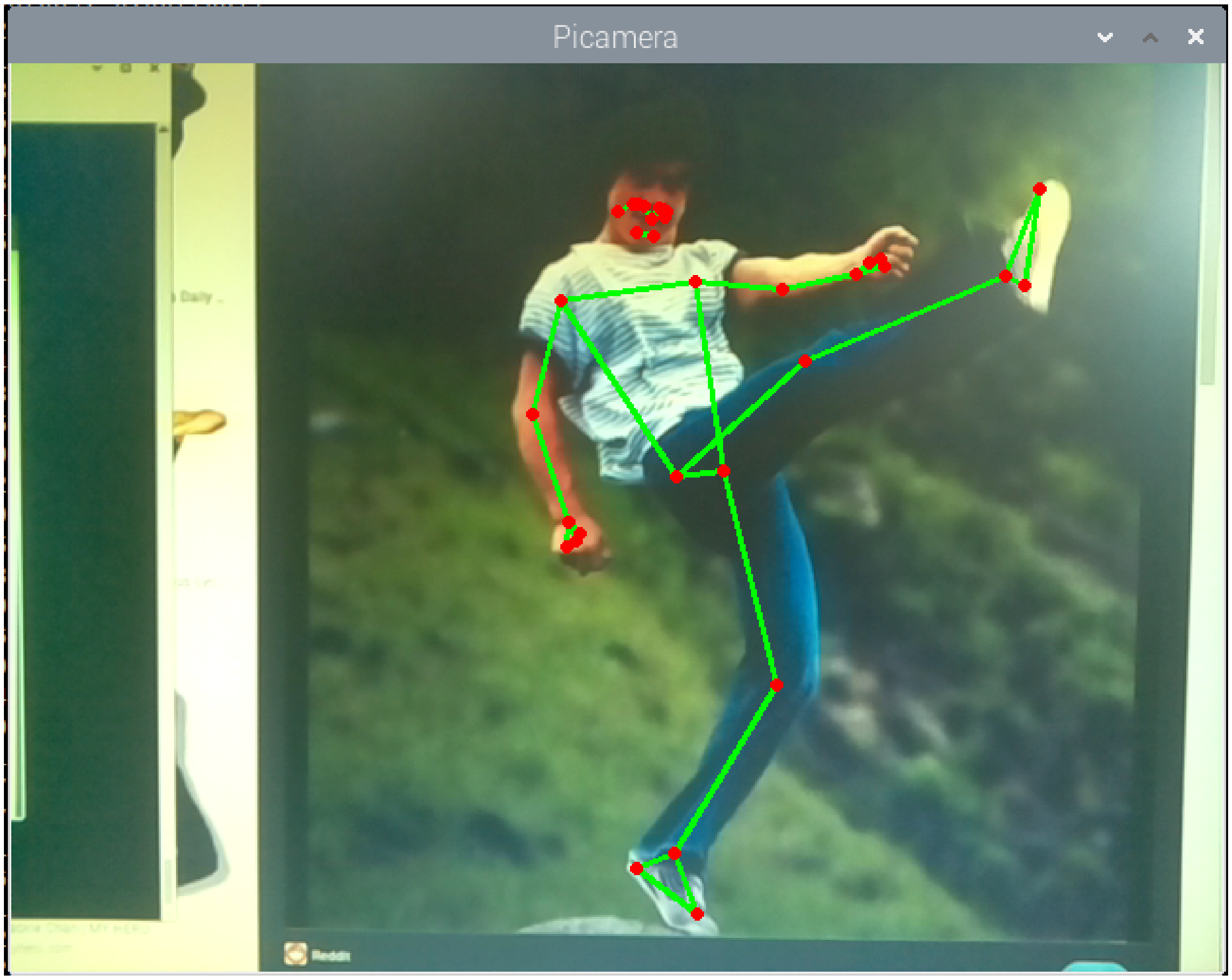
The coordinates of these joints are composed of [x,y,z]. x and y are normalized to 0.0 ~ 1.0 by the width and height of the image (rather than the specific pixel position), the upper left corner of the screen is [0.0, 0.0], and the lower right corner is [1.0, 1.0]. The z coordinate represents the depth,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

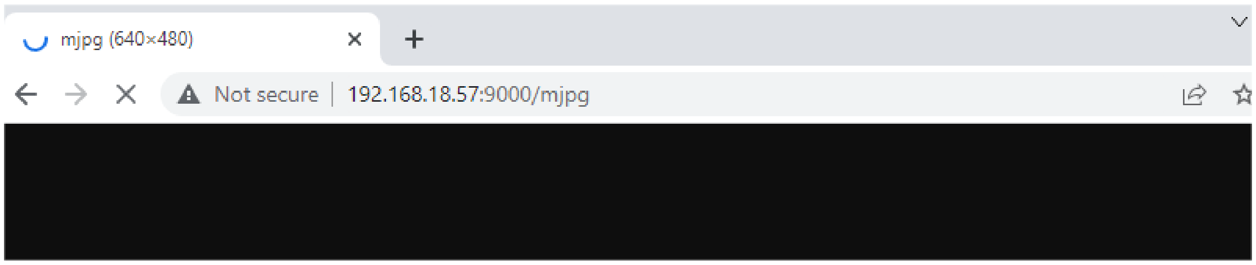
```
cd /home/pi/pan-tilt-hat/examples
sudo python3 pose_detection.py
```

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
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

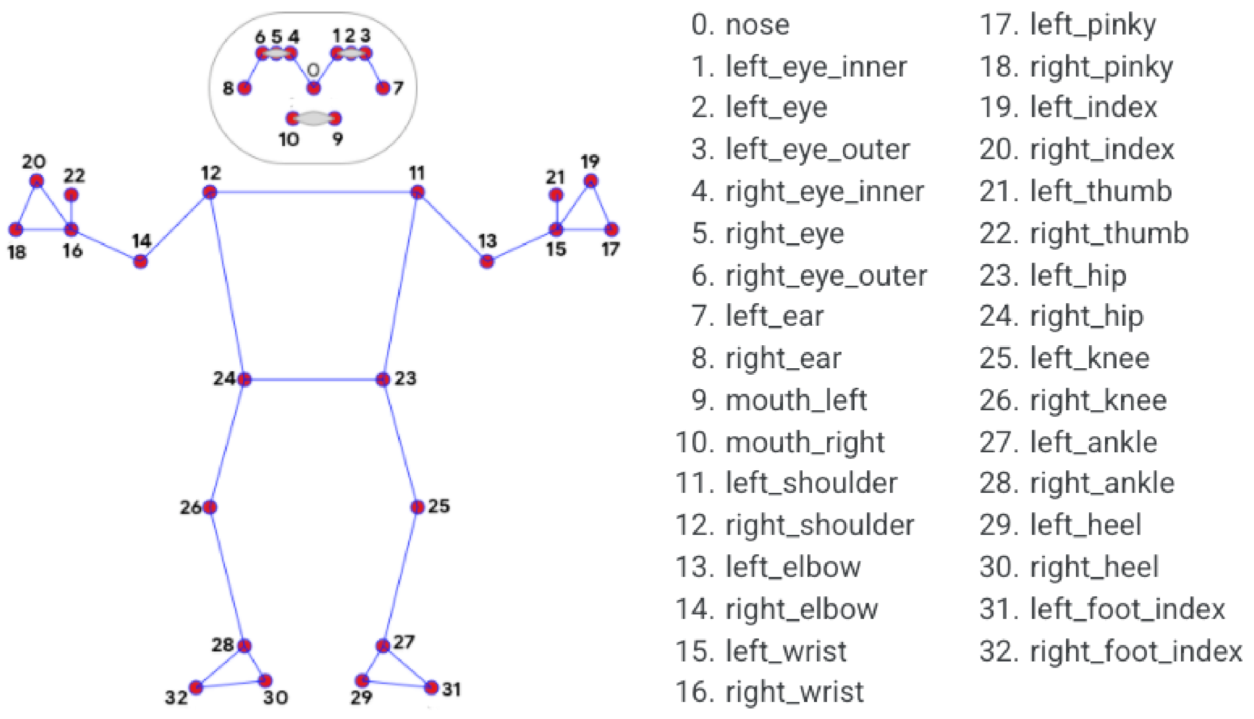
```
from time import sleep
from vilib import Vilib


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

APPENDIX

3.1 PuTTY

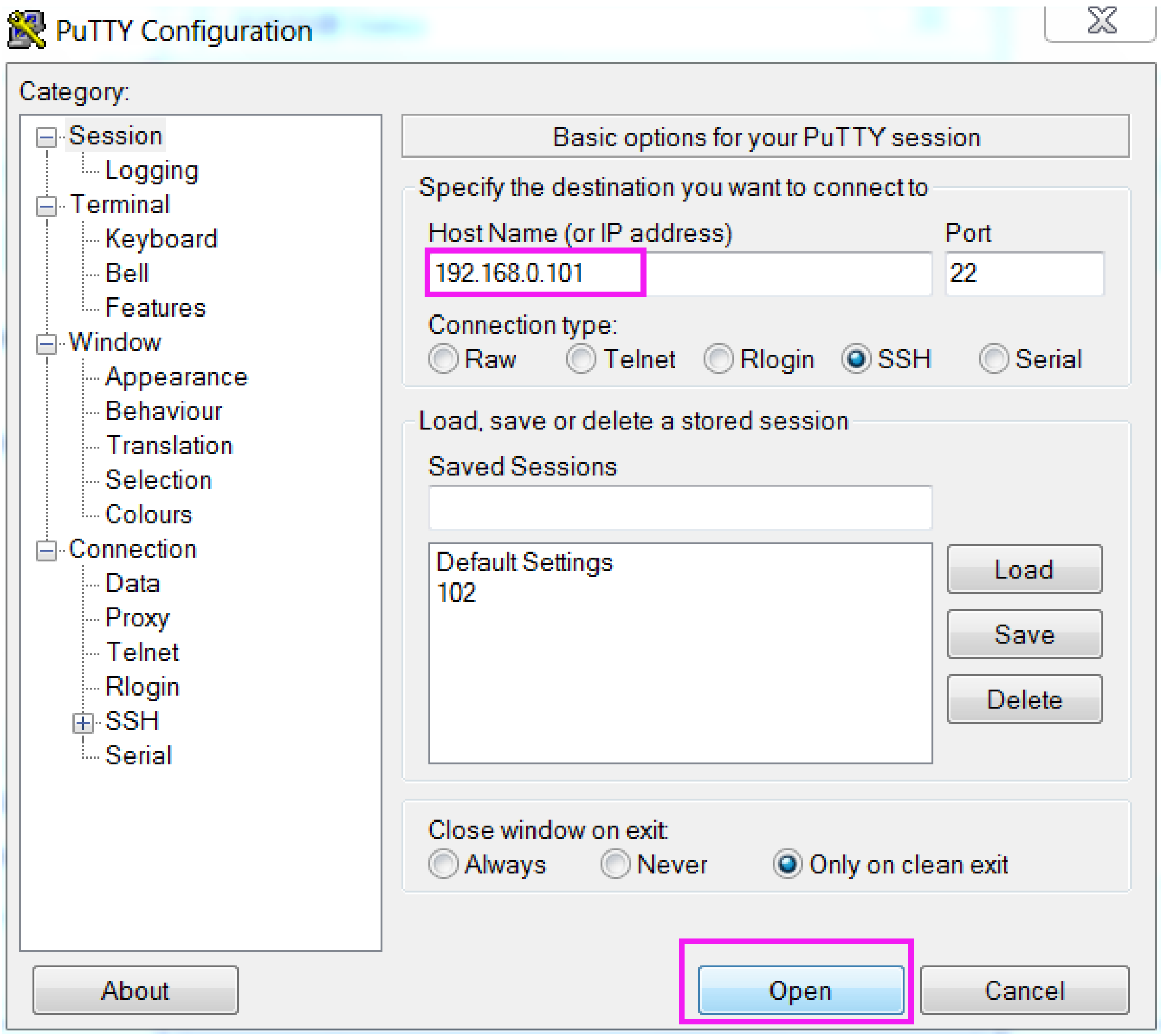
If you are a Windows user, you can use some applications of SSH. Here, we recommend **PuTTY**.

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).



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

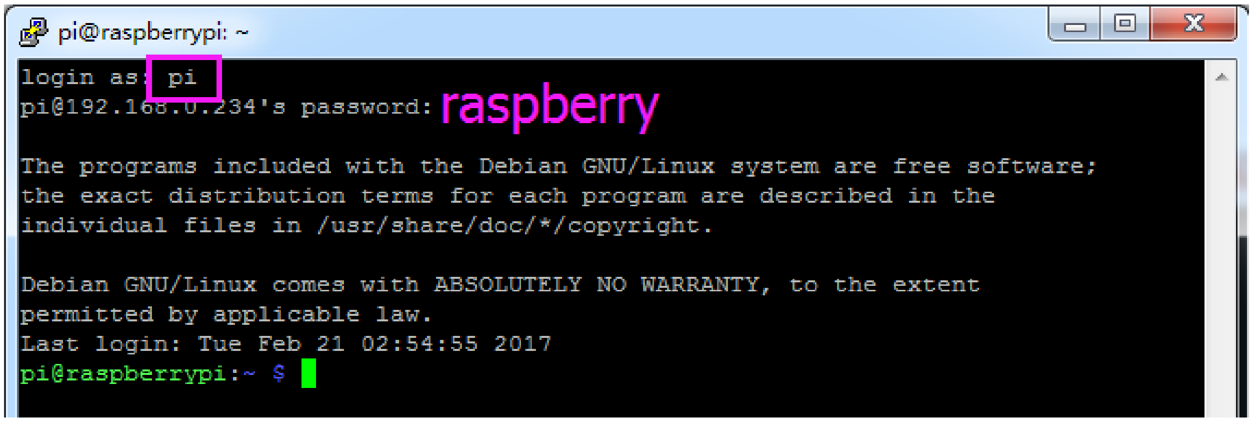
Click **Open**. Note that when you first log in to the Raspberry Pi with the IP address, there prompts a security reminder. Just click **Yes**.

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.



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

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

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

Step 5

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

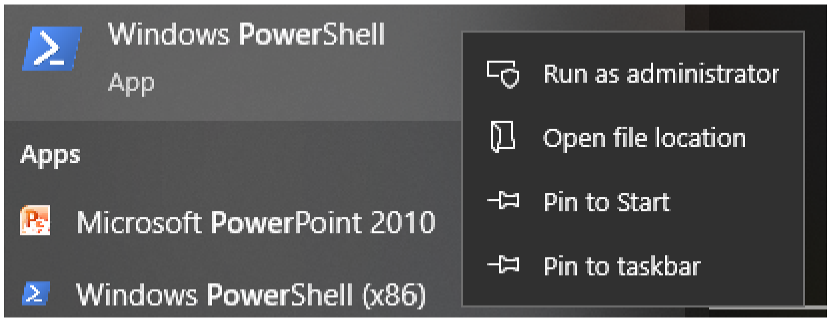
3.2 Install OpenSSH via Powershell

When you use `ssh <username>@<hostname>.local` (or `ssh <username>@<IP address>`) to connect to your Raspberry Pi, but the following error message appears.

```
ssh: The term 'ssh' is not recognized as the name of a cmdlet, function,␣
↪script file, or operable program. Check the
spelling of the name, or if a path was included, verify that the path is␣
↪correct and try again.
```

It means your computer system is too old and does not have OpenSSH pre-installed, you need to follow the tutorial below to install it manually.

1. Type `powershell` in the search box of your Windows desktop, right click on the `Windows PowerShell`, and select `Run as administrator` from the menu that appears.



2. Use the following command to install `OpenSSH.Client`.

```
Add-WindowsCapability -Online -Name OpenSSH.Client~~~~0.0.1.0
```

3. After installation, the following output will be returned.

```
Path          :
Online        : True
RestartNeeded : False
```

4. Verify the installation by using the following command.

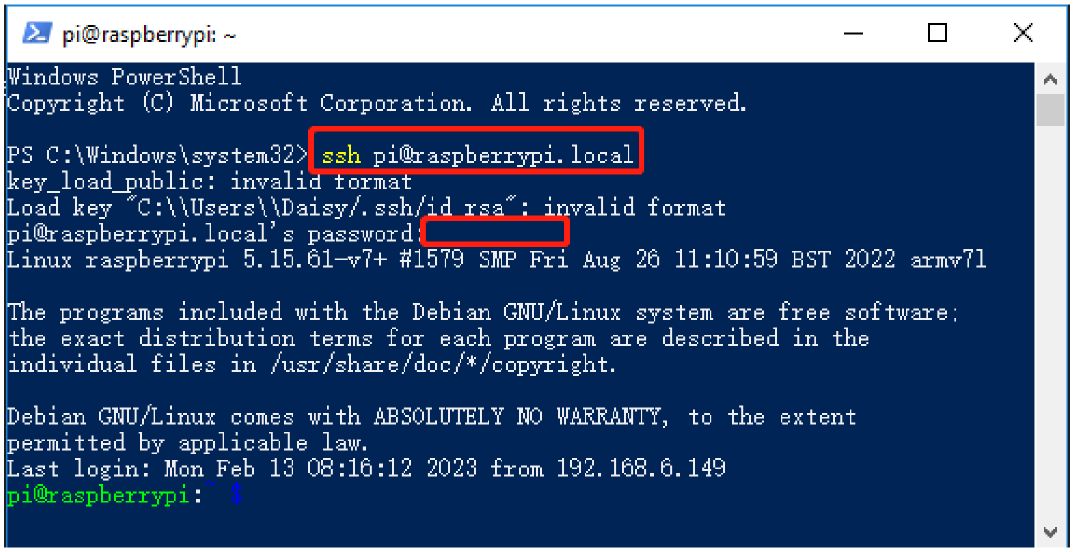
```
Get-WindowsCapability -Online | Where-Object Name -like 'OpenSSH*'
```

5. It now tells you that `OpenSSH.Client` has been successfully installed.

```
Name  : OpenSSH.Client~~~~0.0.1.0
State : Installed

Name  : OpenSSH.Server~~~~0.0.1.0
State : NotPresent
```

Warning: If the above prompt does not appear, it means that your Windows system is still too old, and you are advised to install a third-party SSH tool, like *PuTTY*.

6. Now restart PowerShell and continue to run it as administrator. At this point you will be able to log in to your Raspberry Pi using the `ssh` command, where you will be prompted to enter the password you set up earlier.



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Windows\system32> ssh pi@raspberrypi.local
key_load_public: invalid format
Load key "C:\\Users\\Daisy/.ssh/id_rsa": invalid format
pi@raspberrypi.local's password:
Linux raspberrypi 5.15.61-v7+ #1579 SMP Fri Aug 26 11:10:59 BST 2022 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: Mon Feb 13 08:16:12 2023 from 192.168.6.149
pi@raspberrypi:~ $
```

3.3 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.

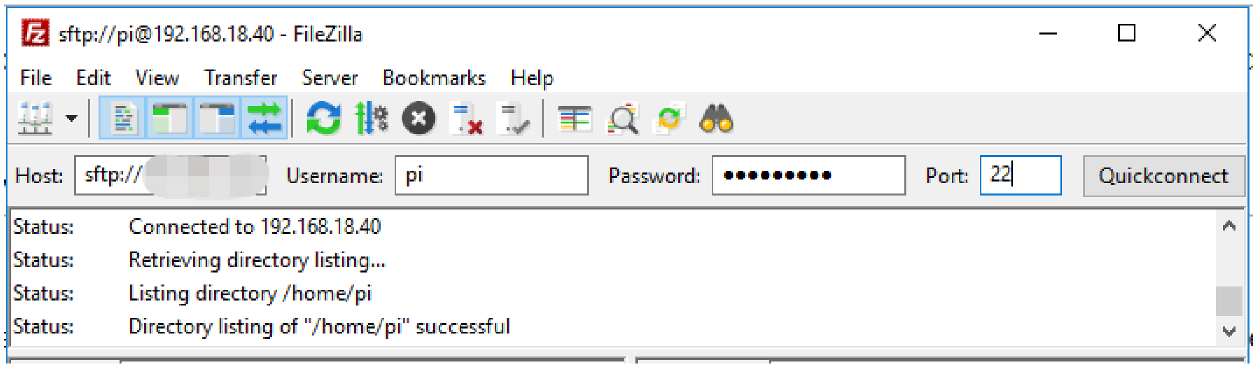
Filezilla is an open source software that not only supports FTP, but also FTP over TLS (FTPS) and SFTP. We can use Filezilla to upload local files (such as pictures and audio, etc.) to the Raspberry Pi, or download files from the Raspberry Pi to the local.

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

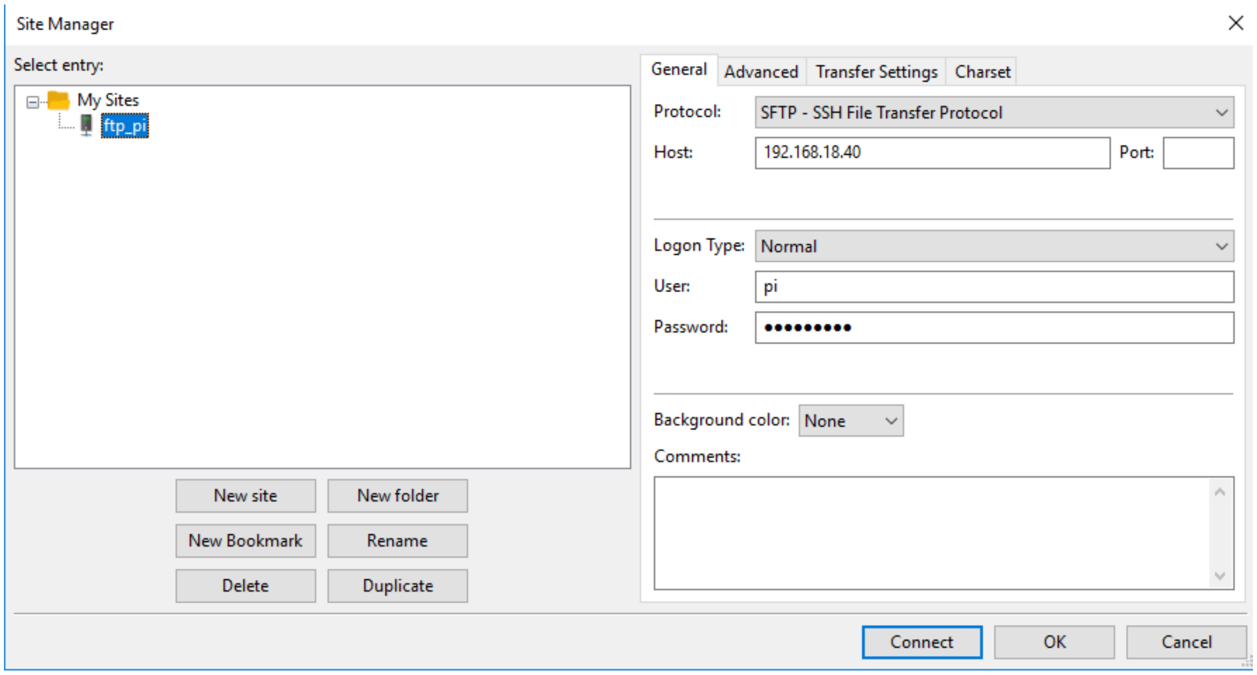
After a quick install open it up and now connect it to an FTP server. It has 3 ways to connect, here we use the **Quick Connect** bar. Enter the **hostname/IP**, **username**, **password** and **port (22)**, then click **Quick Connect** or press **Enter** to connect to the server.


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
```

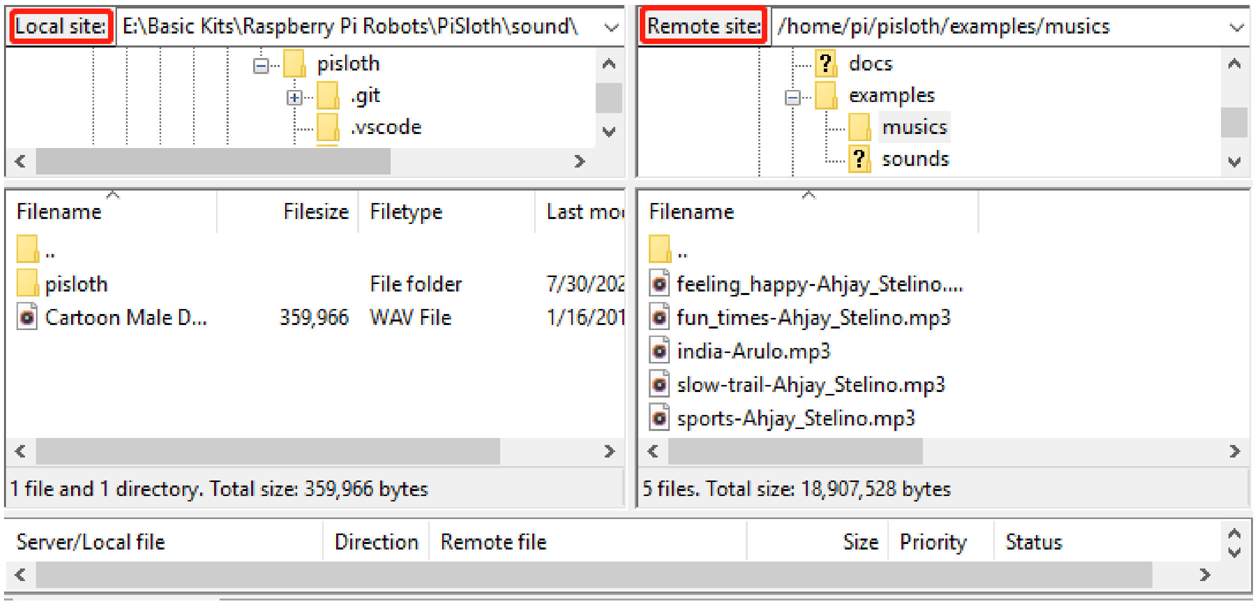
Note: Quick Connect is a good way to test your login information. If you want to create a permanent entry, you can select **File-> Copy Current Connection to Site Manager** after a successful Quick Connect, enter the name and click **OK**. Next time you will be able to connect by selecting the previously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

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.


Local site: E:\Basic Kits\Raspberry Pi Robots\PiSloth\sound\

pisloth
.git
.vscode

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

Remote site: /home/pi/pisloth/examples/musics

docs
examples
musics
sounds

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

Server/Local file	Direction	Remote file	Size	Priority	Status

CHAPTER FOUR

COPYRIGHT NOTICE