SunFounder picar-x

www.sunfounder.com

Mar 05, 2024

CONTENTS

Thanks for choosing our PiCar-X.

Note: This document is available in the following languages.

-
-
-

Please click on the respective links to access the document in your preferred language.

Warning: We offer two versions of PiCar-X. It's crucial to note that the scripts in each version's online tutorial are not interchangeable.

To ensure the proper setup, you'll need to identify your version using the short link provided in your instruction sheet:

- If the link is "picar-x.rtfd.io", continue with this tutorial.
- If the link displays "picar-x-v20.rtfd.io", kindly follow the tutorial at .
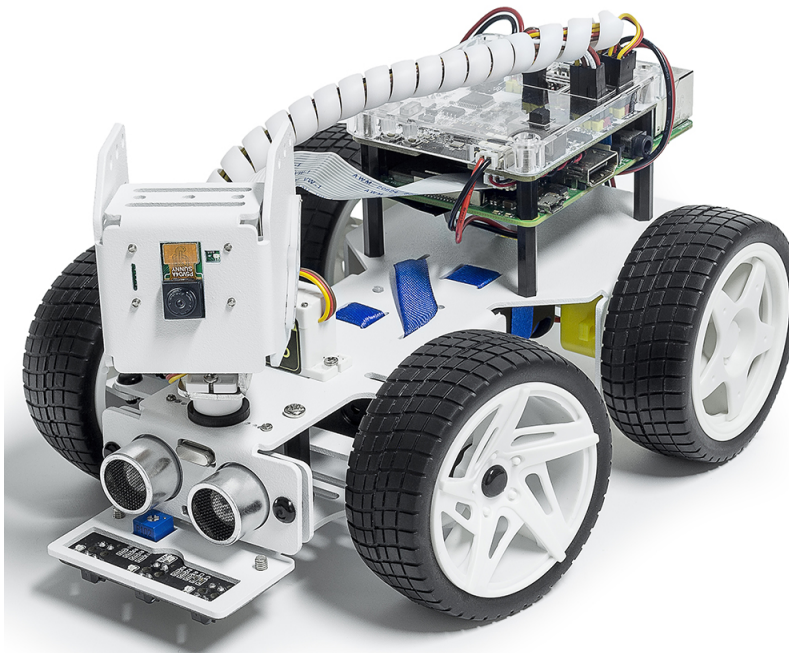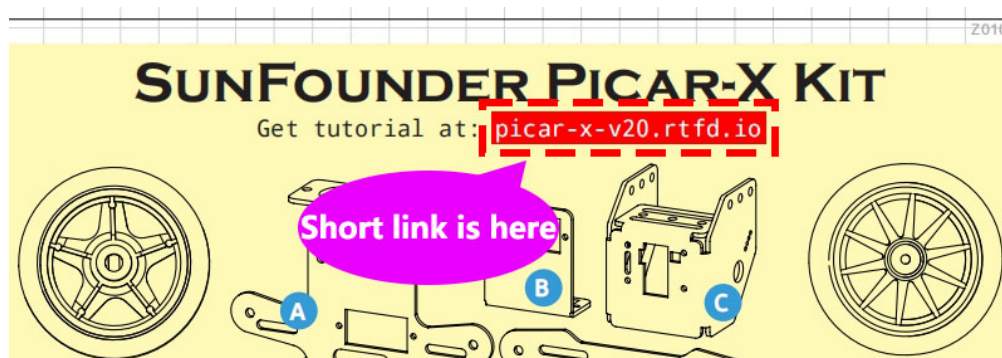
The PiCar-X is an AI-driven self-driving robot car for the Raspberry Pi platform, upon which the Raspberry Pi acts as the control center. The PiCar-X's 2-axis camera module, ultrasonic module, and line tracking modules can provide the functions of color/face/traffic-signs detection, automatic obstacle avoidance, automatic line tracking, etc.

PiCar-X has two programming languages: Blockly and Python. No matter what language you program in, you'll find detailed steps to teach you everything from configuring the Raspberry Pi to running the relevant example code.

- *Play with Python*

    – This chapter is for those who enjoy programming in Python or want to learn the Python language.

    – To get Picar-X working properly, you must install some libraries first.

    – The Raspberry Pi configuration and samples code for the PiCar-X are provided in this chapter.

    – An APP - SunFounder Controller is also provided to allow you to remotely control the PiCar-X on your mobile device.

- *Play with Ezblock*

    – In this section, you will use a Blockly based APP, Ezblock Studio, which, like Scratch, allows you to drag and drop blocks to make Picar-X move.

    – It is required to reinstall the SD card with the operating system we provide with pre-installed Ezblock environment before programming. It is recommended to use a new or unused TF card for this section.

    – Ezblock Studio is available for nearly all types of devices, including Macs, PCs, and Androids.

    – Ezblock Studio is a good choice if you are 6-12 years old, or don't have programming skills, or want to test Picar-X quickly.

Content

ONE

INTRODUCTION

1.1 The History of Self-driving Cars

Experiments have been conducted on self-driving cars since at least the 1920's. Promising trials took place in the 1950's, and work has proceeded forward ever since. The first self-sufficient and truly autonomous cars appeared in the 1980's, with Carnegie Mellon University's Navlab and ALV projects in 1984, and Mercedes-Benz and Bundeswehr University Munich's Eureka Prometheus Project in 1987. Since the late 1980's, numerous research organizations and major automakers have developed working autonomous vehicles, including: Mercedes-Benz, General Motors, Continental Automotive Systems, Autoliv Inc., Bosch, Nissan, Toyota, Audi, Volvo, Vislab from University of Parma, Oxford University, and Google. In July 2013, Vislab demonstrated BRAiVE, a vehicle that moved autonomously on a mixed traffic route open to the public. As of 2019, twenty-nine U.S. states have already passed laws permitting autonomous cars on public roadways.

Some UNECE members and EU members, including the UK, have enacted rules and regulations related to automated and fully automated cars. In Europe, cities in Belgium, France, Italy, and the UK have plans in place to operate transport systems for driverless cars, and Germany, the Netherlands, and Spain have already allowed the testing of robotic cars in public traffic. In 2020, the UK, the EU, and Japan are already on track to regulate automated cars.

- Reference: History of self-driving cars - Wikipedia

Today, self-driving cars are the closest technological revolution at hand. Some experts predict that by 2025, Level 4 cars are likely to enter the market. The Level 4 cars will allow drivers to divert their attention to something else entirely, eliminating the need to pay attention to traffic conditions as long as the system is functioning properly.

Level 4 reference:

- SAE Levels of Driving Automation™

- ABI Research Forecasts 8 Million Vehicles to Ship with SAE Level 3, 4 and 5 Autonomous Technology in 2025

Recent rapid advances in software (Artificial Intelligence, Machine Learning), hardware (GPUs, FPGAs, accelerometers, etc.), and cloud computing are driving this technological revolution forward.

- In October 2010, a driverless truck designed by the Italian technology company **Vislab** took three months to travel from Italy to China, with a total distance of 8, 077 miles.

- In April 2015, a car designed by **Delphi Automotive** traveled from San Francisco to New York , traversing 3,400 miles, completing 99 percent of that distance under computer control.

- In December 2018, **Alphabet**'s **Waymo** launched a level 4 self-driving taxi service in Arizona , where they had already been testing driverless cars since 2008. With no one in the driver's seat, the vehicles operated for more than a year and traveled over 10 million miles.

- In October 2020, **Baidu** fully opened its Apollo Robotaxi self-driving cab service in Beijing. The driving routes cover local residential, commercial, leisure, and industrial parks areas, and offer a fully autonomous driving system.

However, despite the massive amounts of data collected every day, including training data from real driving records and simulated scenarios, the complexity of AI models for self-driving cars has not been fully met.

According to RAND's report , reaching the appropriate level of autonomous learning requires training data from hundreds of millions, or even hundreds of billions of miles to establish a level of reliability.

So, while the future of self-driving cars is promising and exciting, there are still many more years of development to go before the technology has matured enough to become fully accessible to the self-driving car market.

The proven way to allow an emerging technology to quickly mature is to make it easily accessible to everyone by minimizing the market-entry requirements. This is SunFounders motivation for launching PiCar-X.

SunFounders goal is to help beginners, novices, and those who simply just want to learn about autonomous driving, to understand the development process, the technology, and the latest innovations in self-driving vehicles.

1.2 About PiCar-X
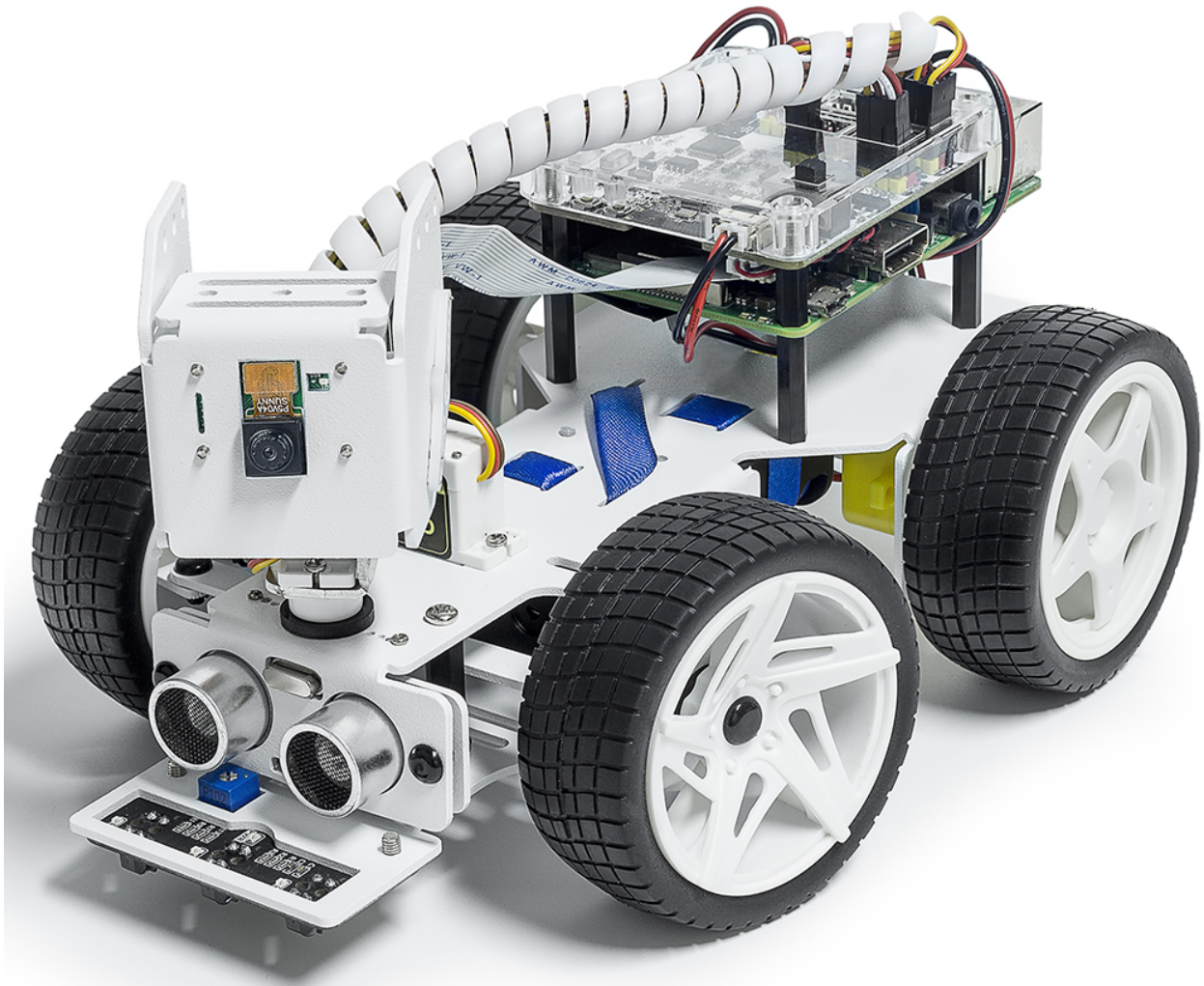
The PiCar-X is an AI-controlled self-driving robot car for the Raspberry Pi platform, upon which the Raspberry Pi acts as the control center. The PiCar-X's 2-axis camera module, ultrasonic module, and line tracking modules can provide the functions of color/face/traffic signs detection, automatic obstacle avoidance, automatic line tracking, etc.

With the SunFounder-designed Robot HAT board, the PiCar-X integrates left/right driving motors, servo motors for steering and the camera's pan/tilt functions, and pre-sets the Robot HAT's ADC, PWM, and Digital I2C pins to allow for extensions to the standard functionality of the Raspberry Pi. Both a speaker and a bluetooth chip have been engineered into the Robot HAT for remote control of Text-to-Speech, sound effects, or even background music functionality.

All of the PiCar-X functions, including GPIO control, computer vision, and deep learning, are implemented through the open sourced Python programming language, OpenCV's Computer Vision Library software, and Google's TensorFlow for deep learning frameworks. Other software has been included to optimize the PiCar-X capabilities, allowing the user a near-limitless learning environment.

1.3 Deep Learning and Neural Networks

To learn more about deep learning and Neural Networks, SunFounder recommends the following resources:

Machine Learning - Andrew Ng : This course provides a broad introduction to machine learning, datamining, and statistical pattern recognition.

Neural Networks and Deep Learning : This E-book covers both Neural Networks, a biologically-inspired programming paradigm that enables a computer to learn from observational data, and Deep learning, a powerful set of techniques for machine learning in neural networks.

Rethinking the Inception Architecture for Computer Vision : This high-level white-paper explores the methods users can scale up networks by utilizing added computations as efficiently as possible through factorized convolutions and aggressive regularization.

COMPONENT LIST AND ASSEMBLY INSTRUCTIONS

Before assembling the PiCar-X, please first verify that all parts and components have been included. If there are any missing or damaged components, please contact SunFounder immediately at service@sunfounder.com to resolve the issue as soon as possible.

Please follow the steps on the following PDF for assembly instructions:

[PDF]Component List and Assembly of PiCar-X.

Note: If the servos has been powered on through the Robot HAT after assembly, do not manually force the steering gear, as this could cause damage to the servo.

Note:

1. Before assembling, you need to buy 2 18650 batteries and fully charge them, refer to *About the Battery*.

2. Robot HAT cannot charge the battery, so you need to buy a battery charger at the same time.

ABOUT ROBOT HAT
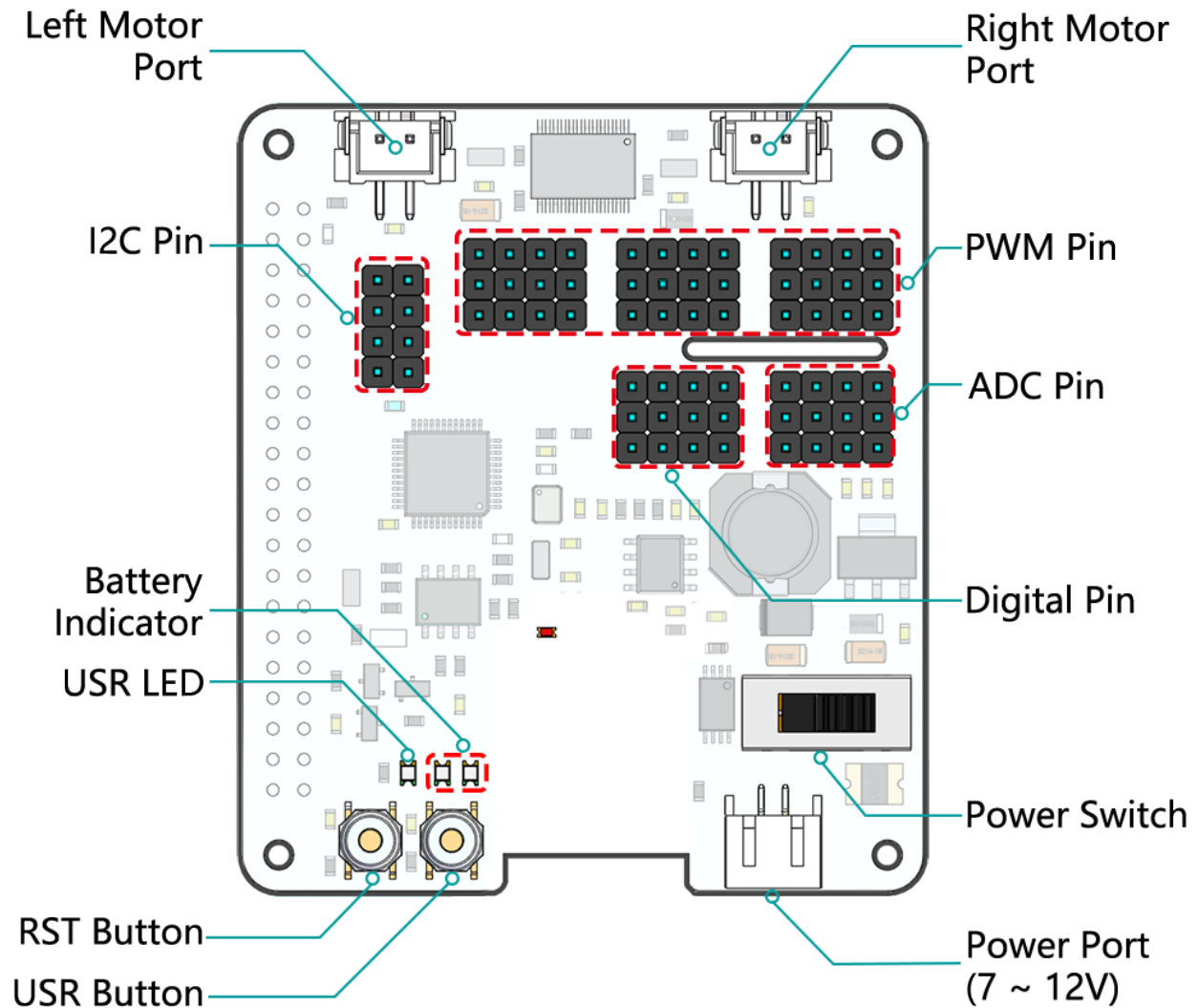
Left/Right Motor Port

- 2-channel XH2.54 motor ports.

- The left port is connected to GPIO 4 and the right port is connected to GPIO 5.

I2C Pin

- 2-channel I2C pins from Raspberry Pi.

PWM Pin

- 12-channel PWM pins, P0-P12.

ADC Pin

- 4-channel ADC pins, A0-A3.

Digital Pin

- 4-channel digital pins, D0-D3.

Battery Indicator

- Two LEDs light up when the voltage is higher than 7.8V.

- One LED lights up in the 6.7V to 7.8V range.

- Below 6.7V, both LEDs turn off.

USR LED

- Set by your program. (Outputting 1 turns the LED on; Outputting 0 turns it off.)

RST Button

- Short pressing RST Button causes program resetting.

- Long press RST Button till the LED lights up then release, and you will disconnect the Bluetooth.

USR Button

- The functions of USR Button can be set by your programming. (Pressing down leads to a input "0"; releasing produces a input "1".)

Power Switch

- Turn on/off the power of the robot HAT.

- When you connect power to the power port, the Raspberry Pi will boot up. However, you will need to switch the power switch to ON to enable Robot HAT.

Power Port

- 7-12V PH2.0 2pin power input.

- Powering the Raspberry Pi and Robot HAT at the same time.

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

PLAY WITH PYTHON

For novices and beginners wishing to program in Python, some basic Python programming skills and knowledge of the Raspberry Pi OS are needed. To start configuring the Raspberry Pi, please reference Quick Guide on Python:

4.1 Quick Guide on Python

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

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

4.1.1 What Do We Need?

Required Components

Raspberry Pi

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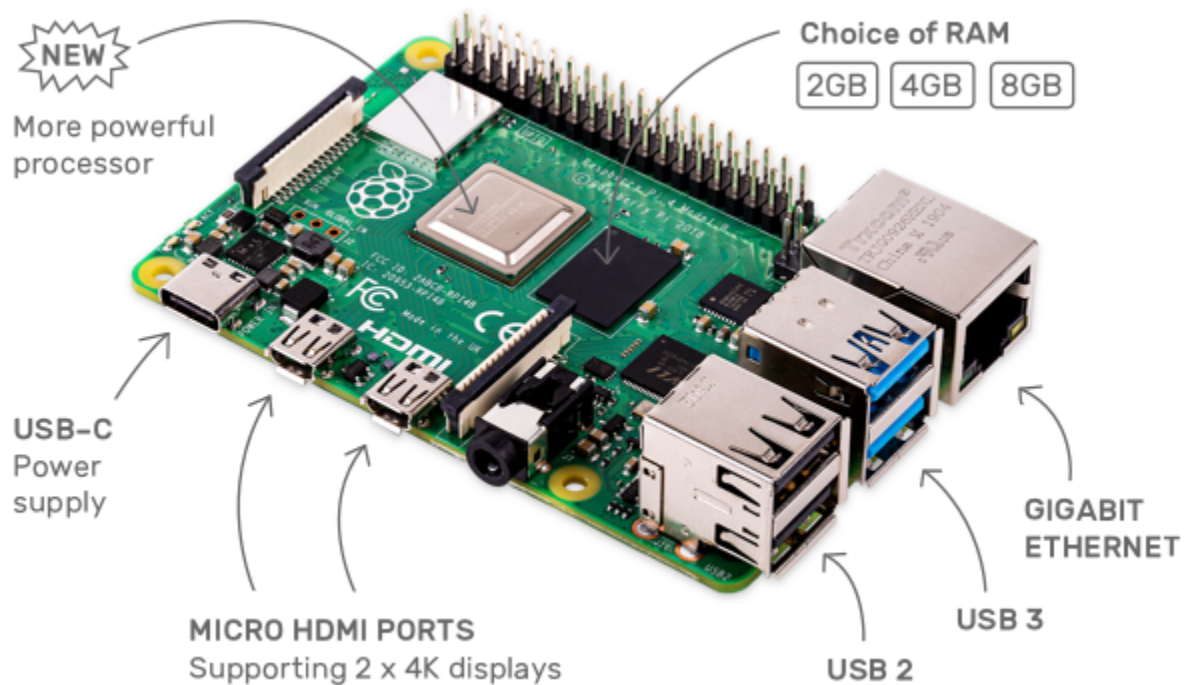
Power Adapter

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

Micro SD Card

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

Optional Components

Screen

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

Mouse & Keyboard

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

HDMI

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

Case

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

Sound or Earphone

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

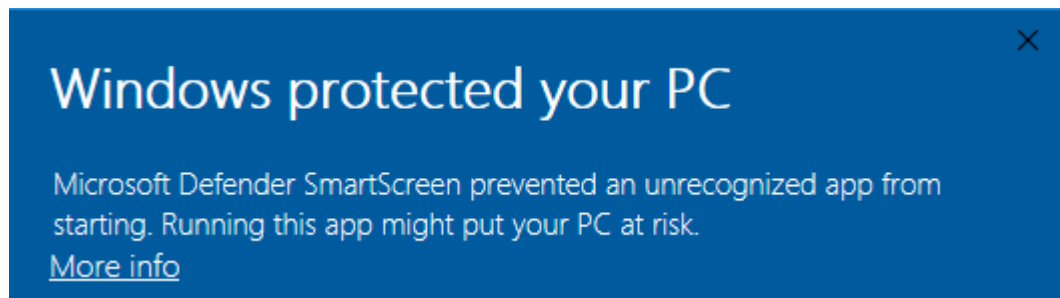### 4.1.2 Installing the OS

Required Components

- Raspberry Pi 4B/Zero 2 w/3B 3B+/2B/Zero W

- 1 x Personal Computer

- 1 x Micro SD card

Steps

1. Go to the Raspberry Pi software download page: Raspberry Pi Imager. Select the Imager version for your operating system. After downloading, open the file to start the installation.

Download for Windows

Download for macOS

Download for Ubuntu for x86
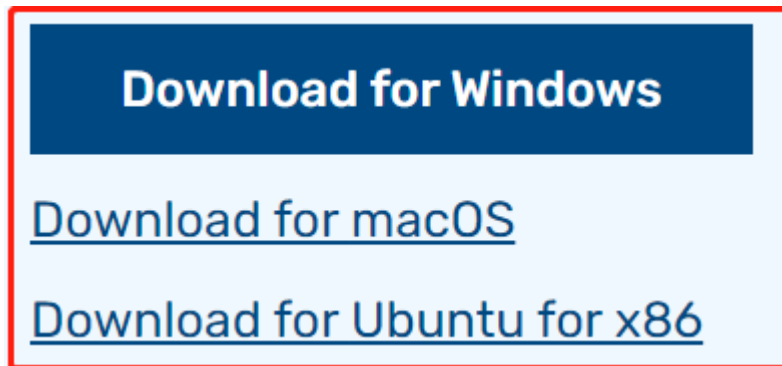
2. Upon launching the installer, your OS might display a security warning. For instance, Windows may show a caution message. If this occurs, select **More info** and then **Run anyway**. Follow the on-screen instructions to install the Raspberry Pi Imager.

Windows protected your PC

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

More info

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

4. Open the Raspberry Pi Imager application either by clicking its icon or executing `rpi-imager` in your terminal.
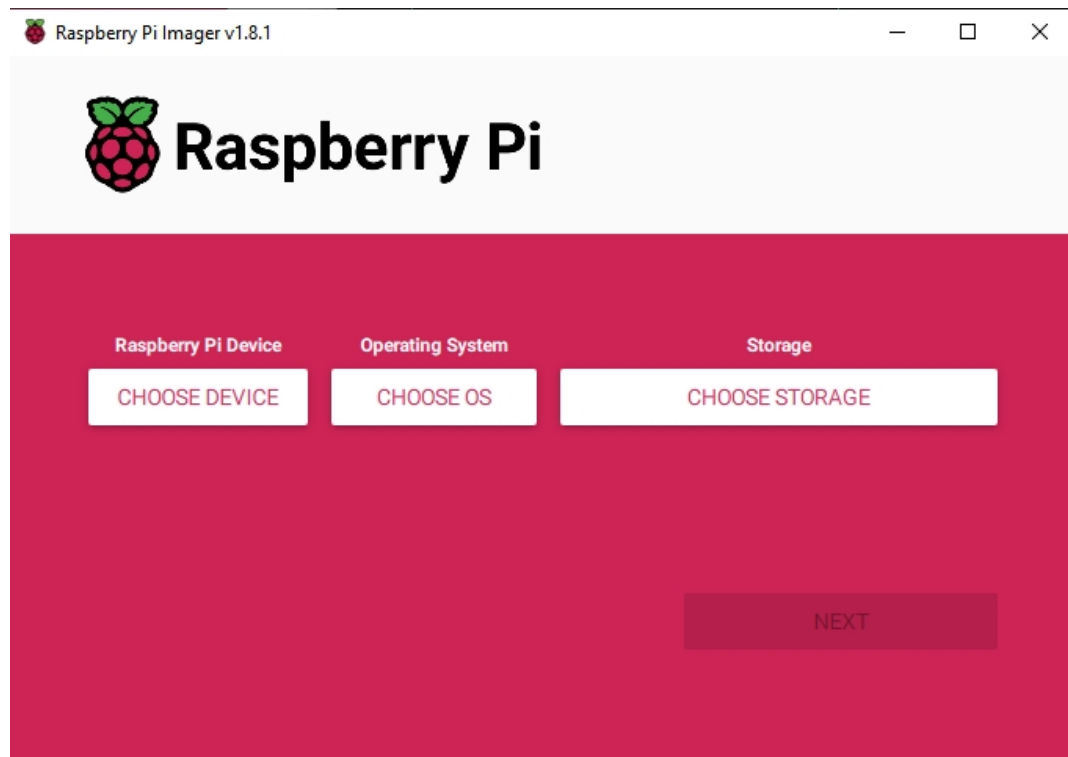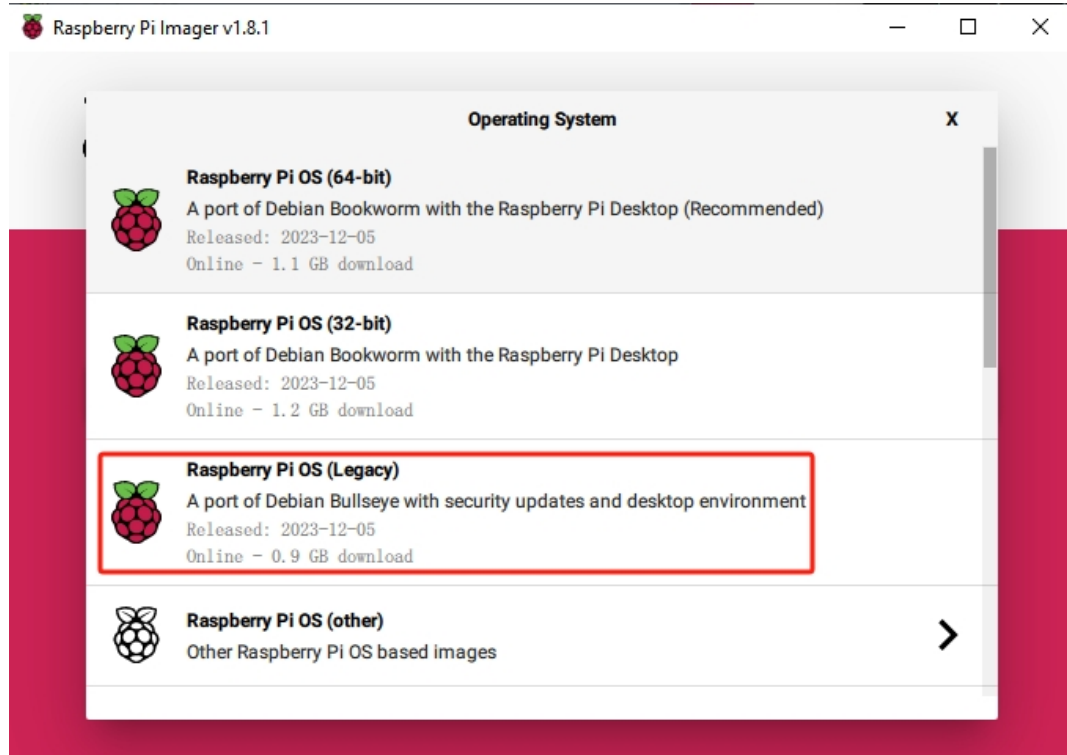
5. Click **CHOOSE DEVICE** and select your specific Raspberry Pi model from the list (Note: Raspberry Pi 5 is not applicable).
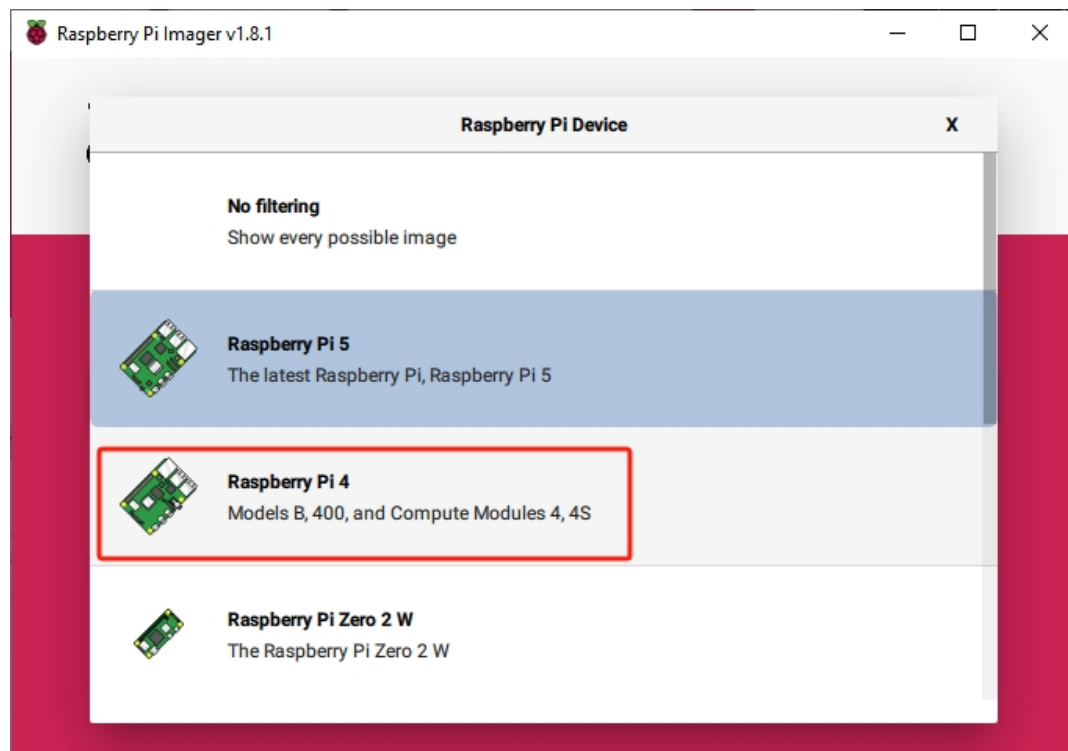
6. Select **CHOOSE OS** and then choose **Raspberry Pi OS (Legacy)**.

> **Warning:**
>
> - Please do not install the **Bookworm** version as the speaker will not work.
>
> - You need to install the **Raspberry Pi OS (Legacy)** version - **Debian Bullseye**.
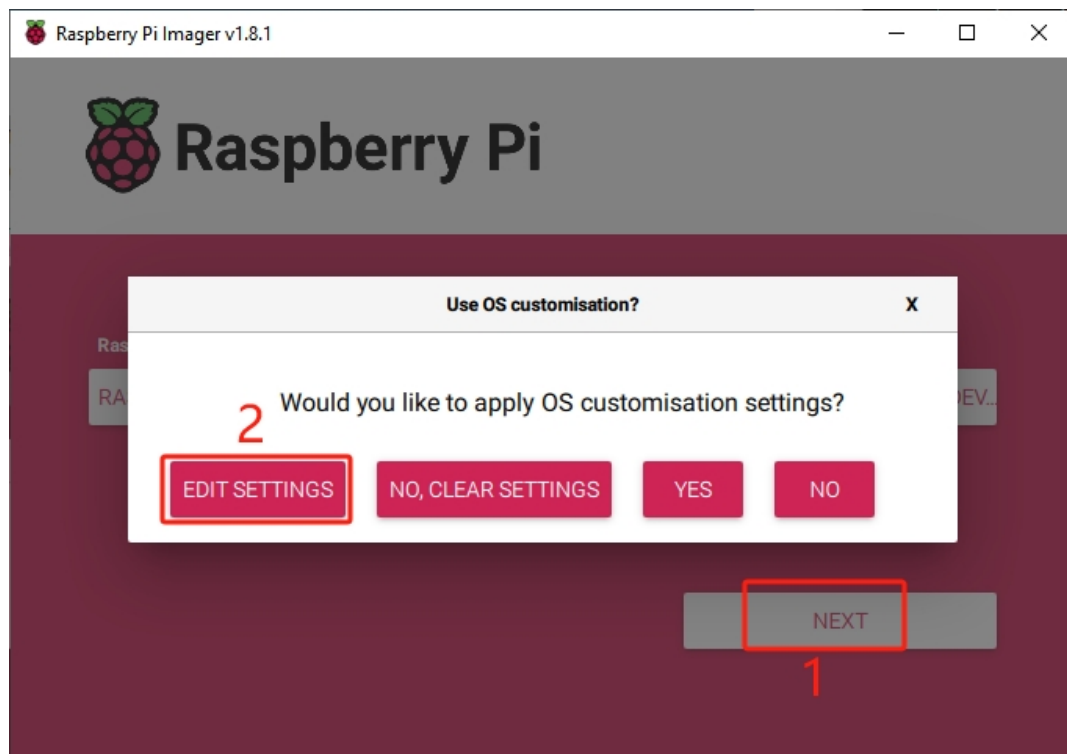

7. Click **Choose Storage** and pick the correct storage device for the installation.

Note: Be sure to select the correct device, especially if multiple storage devices are connected. Disconnect others if you're unsure.
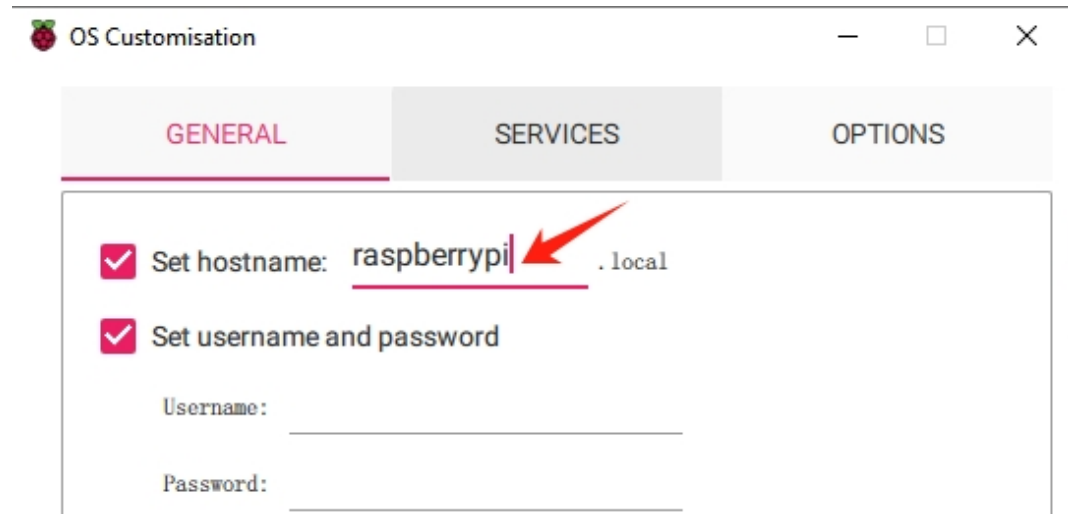
---
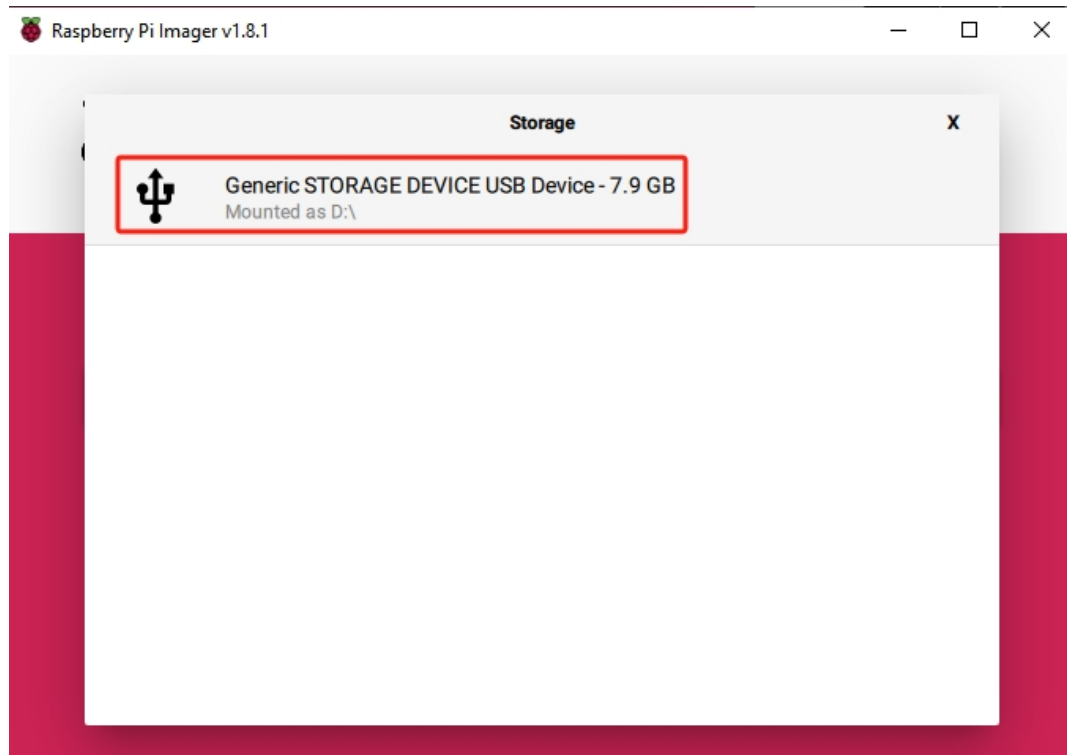
8. Press **NEXT** and select **EDIT SETTINGS** to customize your OS settings.

9. Set your Raspberry Pi's **hostname**.

Note: The hostname is what your Raspberry Pi uses to identify itself on the network. You can

connect to your Pi using *<hostname>.local* or *<hostname>.lan*.
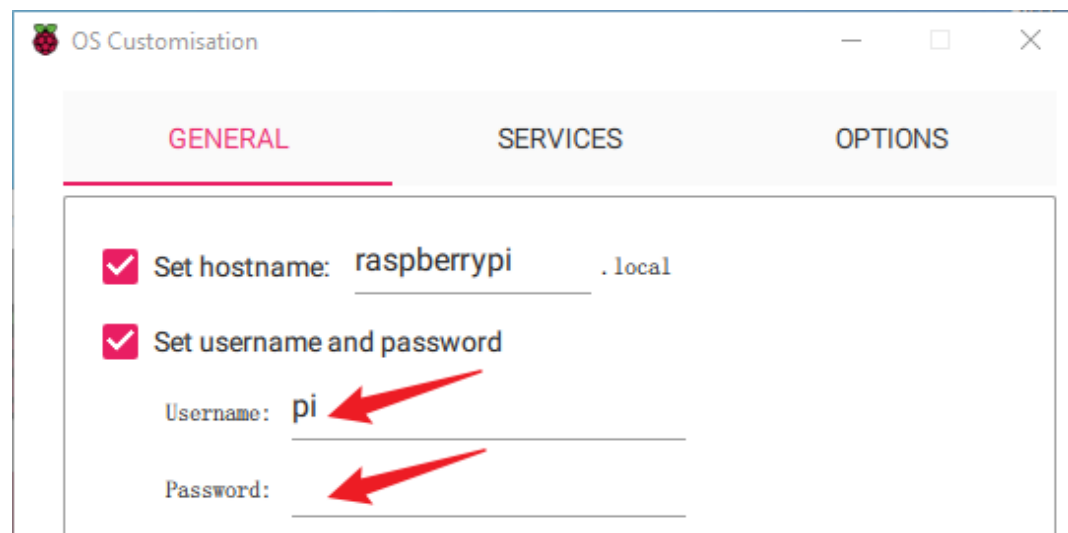
10. Create a **Username** and **Password** for the Raspberry Pi's administrator account.

> **Note:** Setting a unique username and password is crucial for security, as the Raspberry Pi does not have a default password.
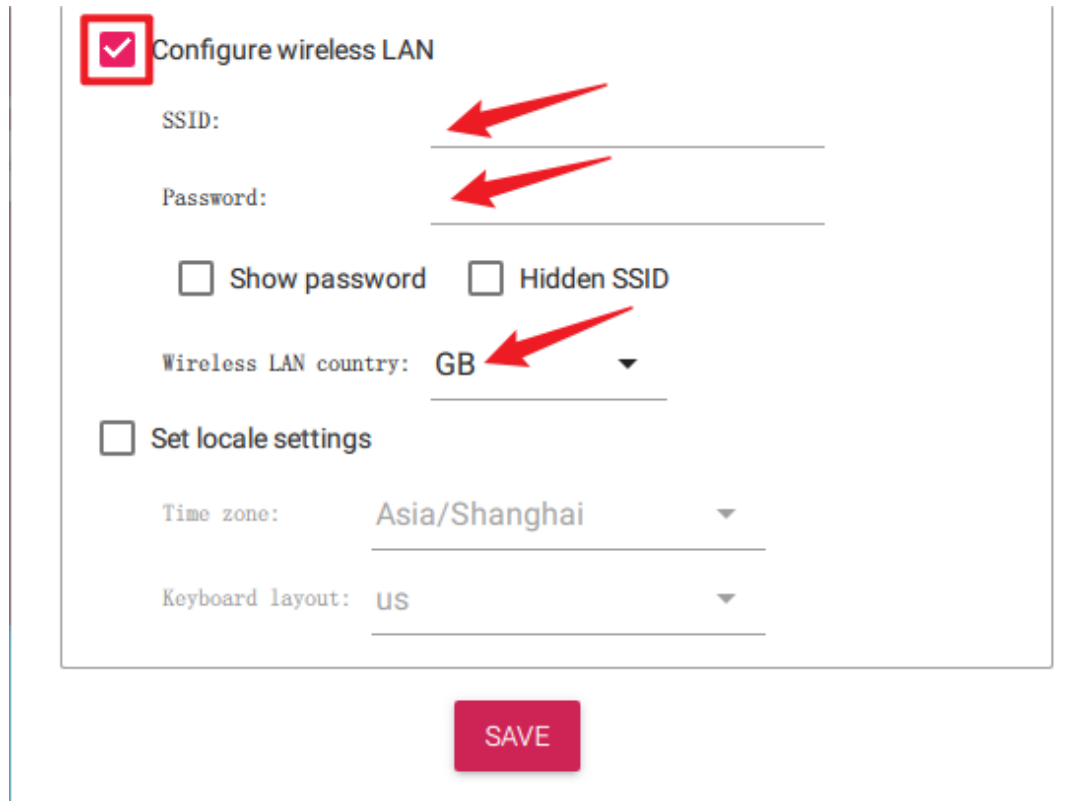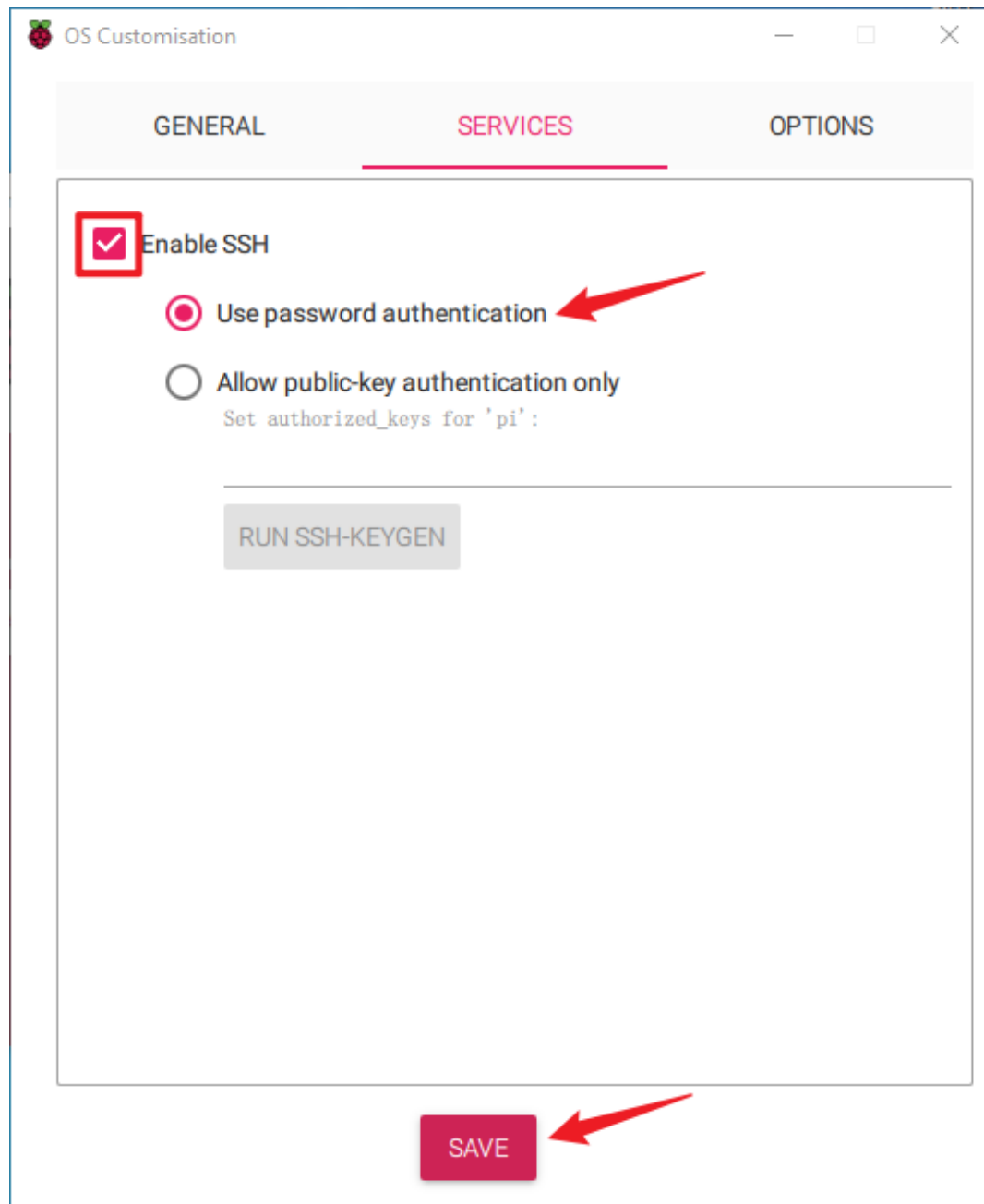

11. Set up wireless LAN by inputting your network's **SSID** and **Password**.

> **Note:** `Wireless LAN country` should be set the two-letter ISO/IEC alpha2 code for the country in which you are using your Raspberry Pi.

12. Click **SERVICES** and enable **SSH** for password-based remote access. Remember to click **Save**.

13. Confirm your choices by clicking **Yes**.
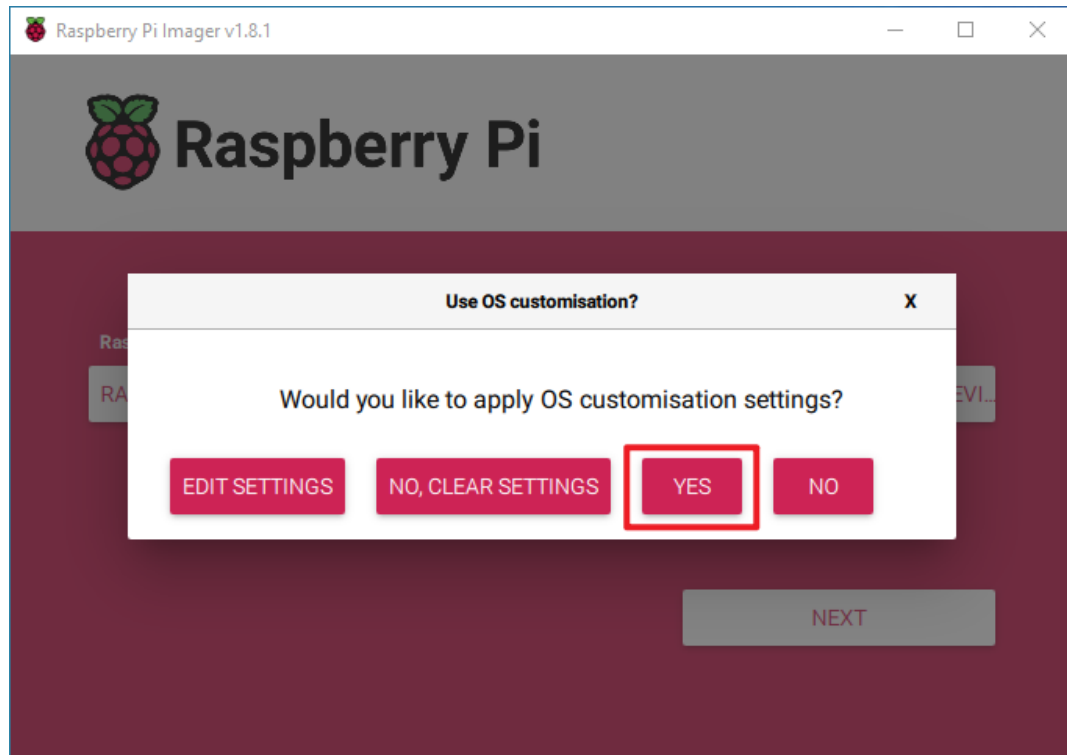
14. If your SD card has existing files, back them up to avoid data loss. Click **Yes** to proceed if no backup is necessary.
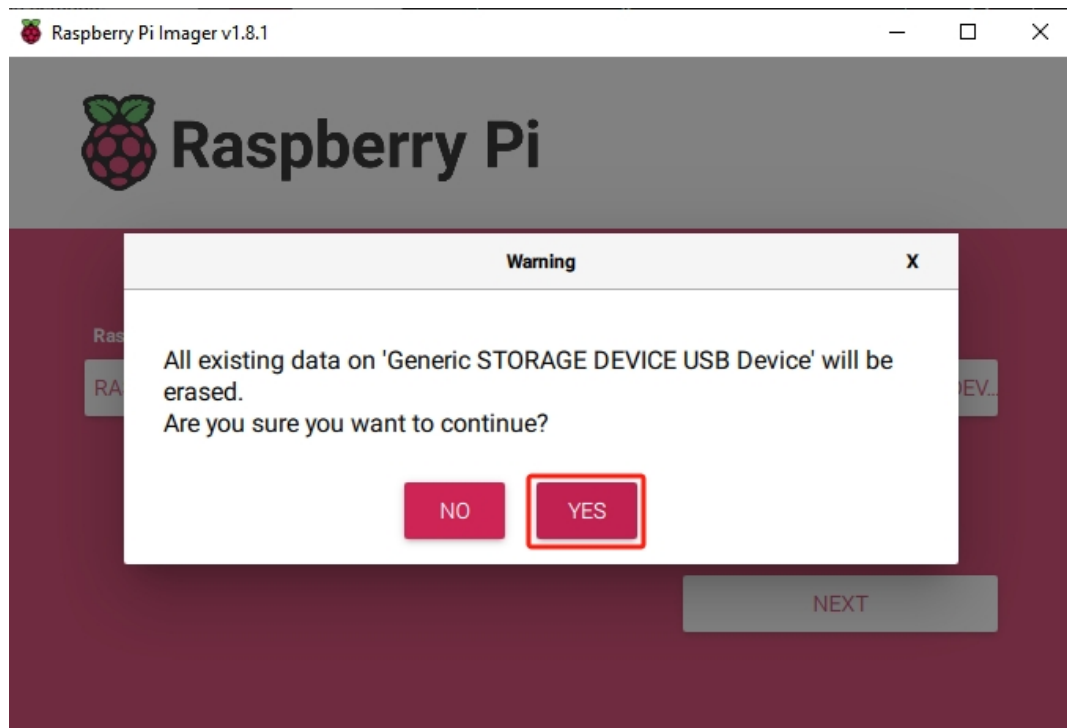
15. Wait as the OS is written to the SD card. Once completed, a confirmation window will appear.
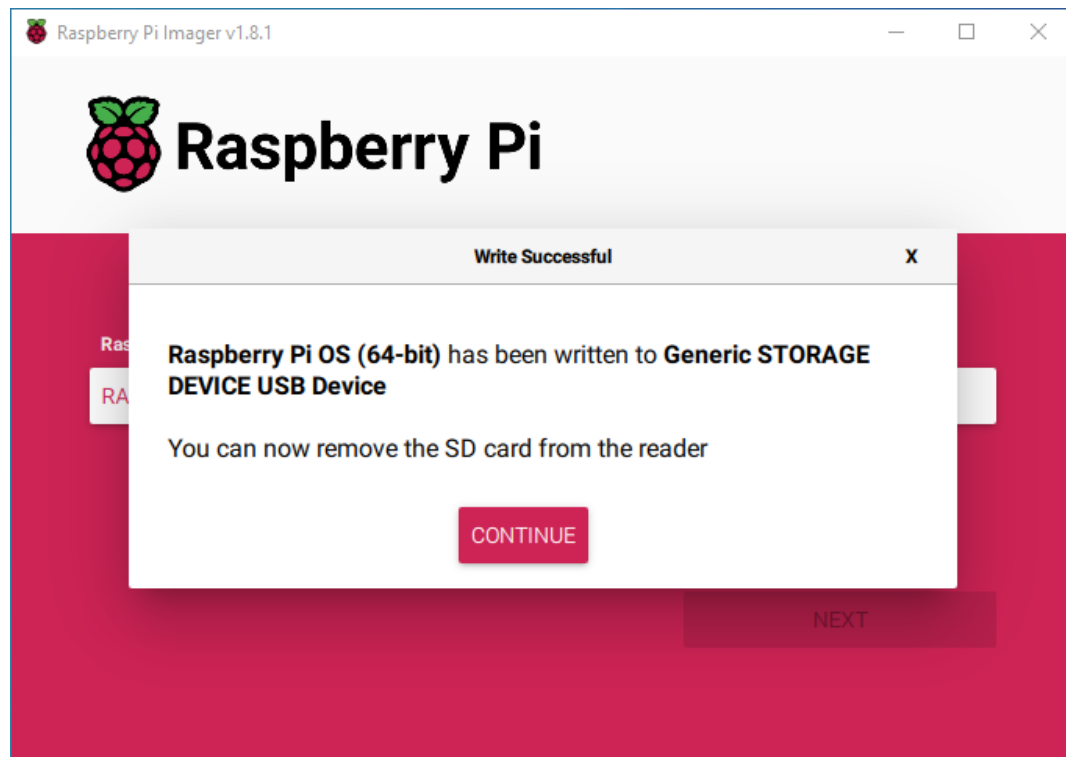
4.1.3 Set up Your Raspberry Pi

Power Supply for Raspberry Pi (Important)

1. Insert the SD card set up with Raspberry Pi OS into the microSD card slot located on the underside of the Raspberry Pi.
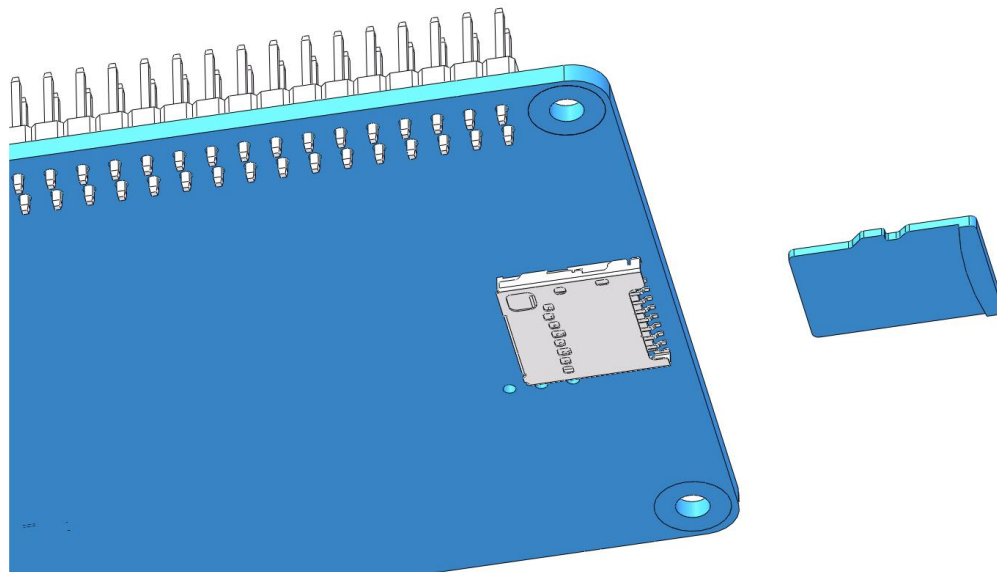
2. Following the assembly instructions, insert the battery cable and turn on the power switch. Next, insert the USB-C cable to power up the battery. Wait for 1-2 minutes, and you will hear a sound indicating that the Raspberry Pi has successfully booted.

Turn ON the Power

Plug in the Battery
Holder Cable
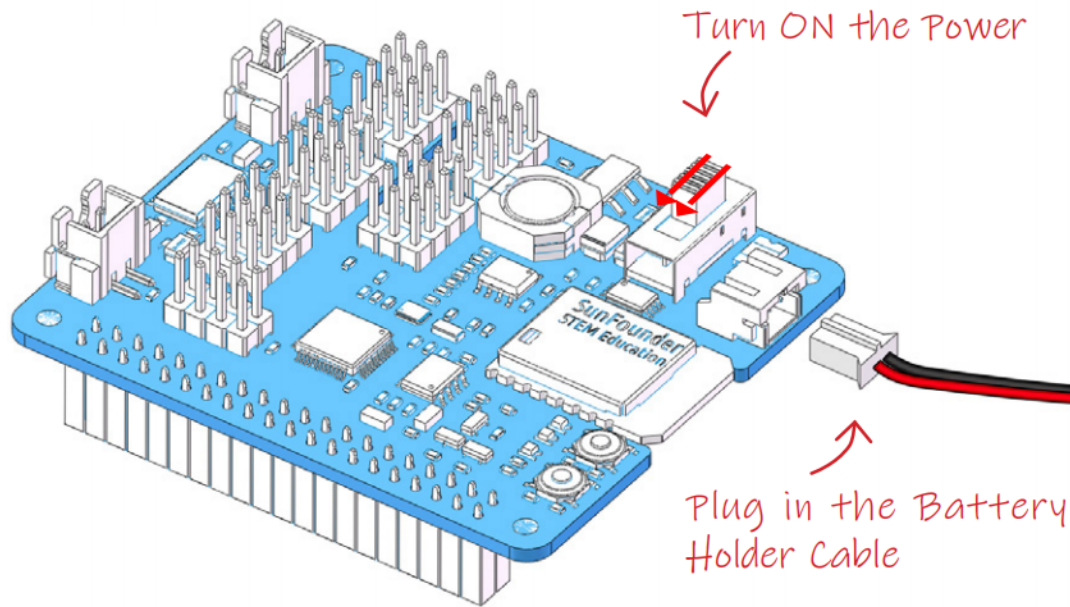
Note: It is recommended to leave the USB-C cable plugged in, as the subsequent software setup
process can take a considerable amount of time.

If You Have a Screen

Note: The Raspberry Pi ZERO installed on the Robot is not easy to connect to the screen, please use the method
without a screen to set it up.

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

Required Components

- Raspberry Pi 4B/3B 3B+/2B

- 1 * Power Adapter

- 1 * Micro SD card

- 1 * Screen Power Adapter

- 1 * HDMI cable

- 1 * Screen

- 1 * Mouse

- 1 * Keyboard

1. Plug in the Mouse and Keyboard.

2. 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).

---

3. Use the power adapter to power the Raspberry Pi. After a few seconds, the Raspberry Pi OS desktop will be displayed.
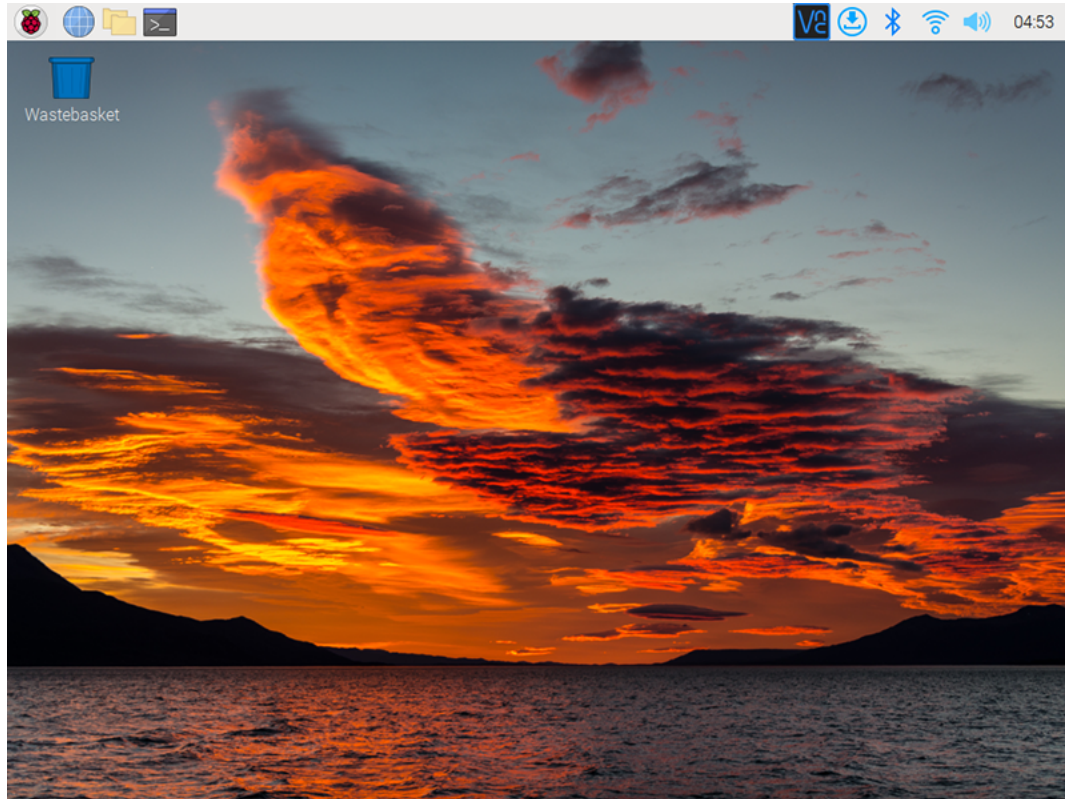
If You Have No Screen

If you don't have a monitor, you can remotely log into your Raspberry Pi.

Required Components

- - Raspberry Pi 4B/Zero 2 w/3B 3B+/2B/Zero W
- 1 * Power Adapter
- 1 * Micro SD card

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.
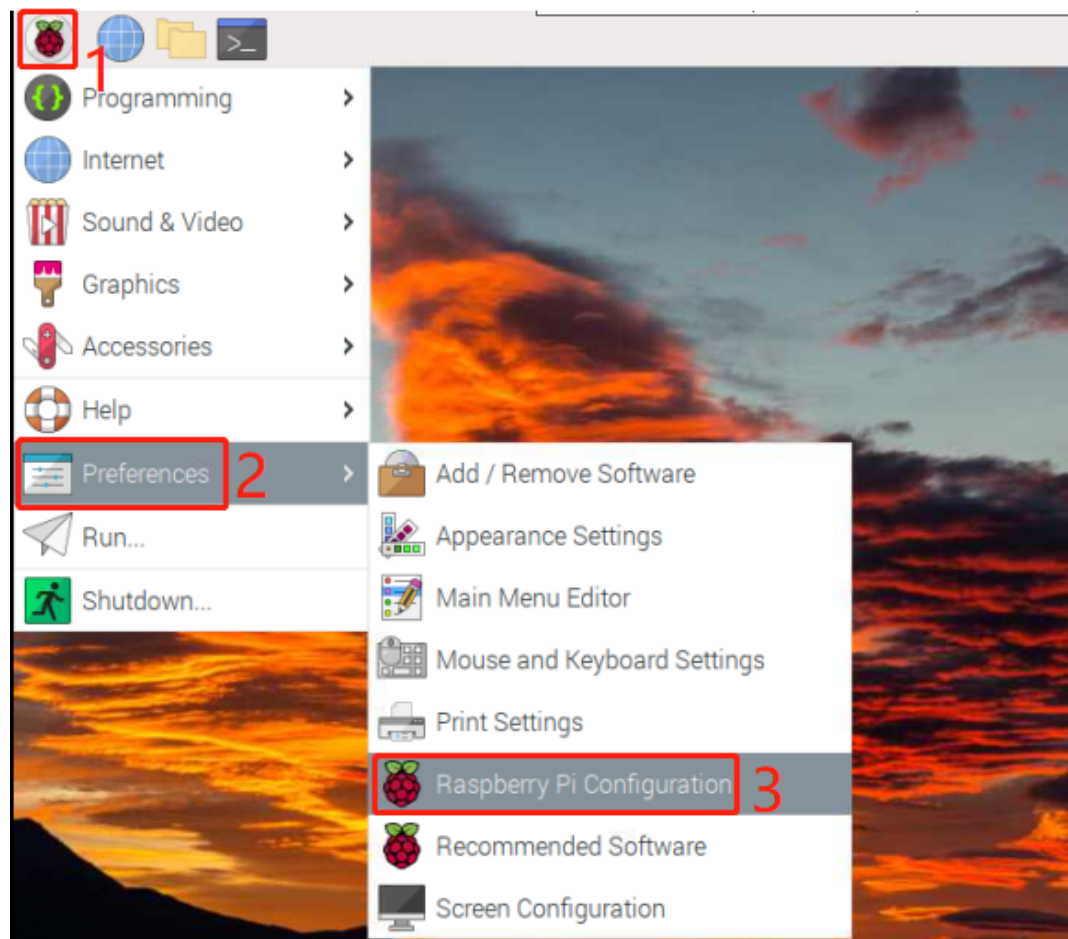
2. The following screen will be displayed. Set **VNC** to **Enabled** on the **Interfaces** tab, and click **OK**.
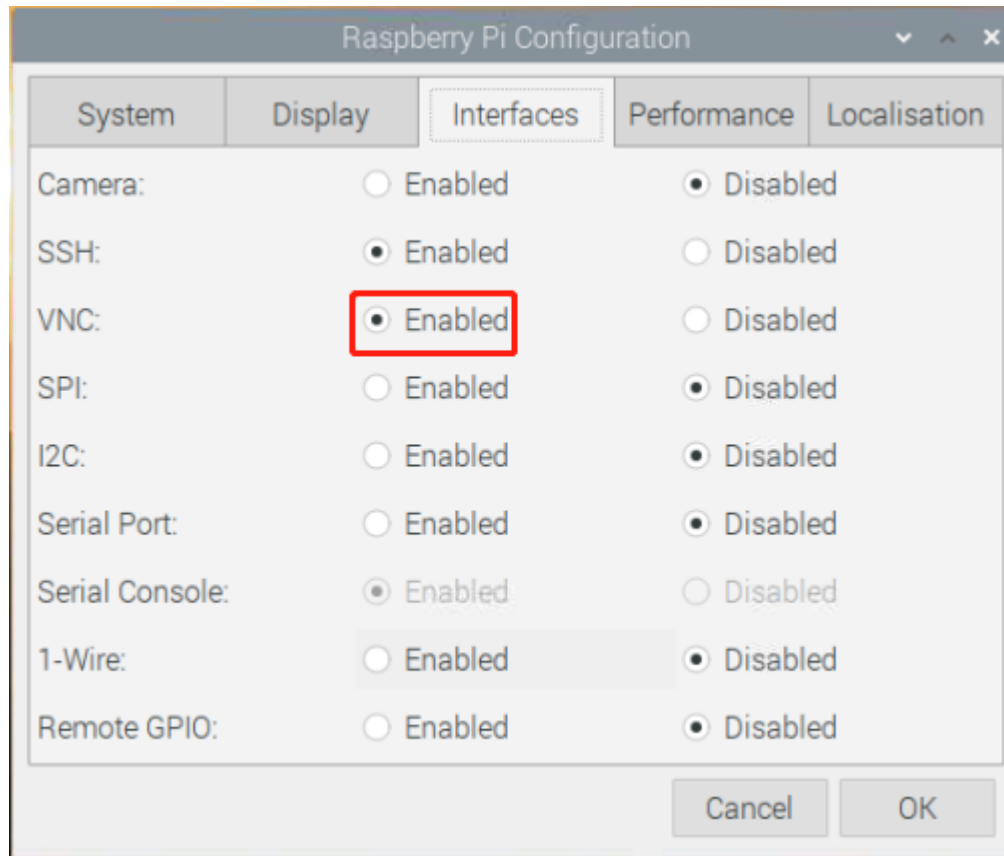
3. A VNC icon appears on the upper right of the screen and the VNC server starts.
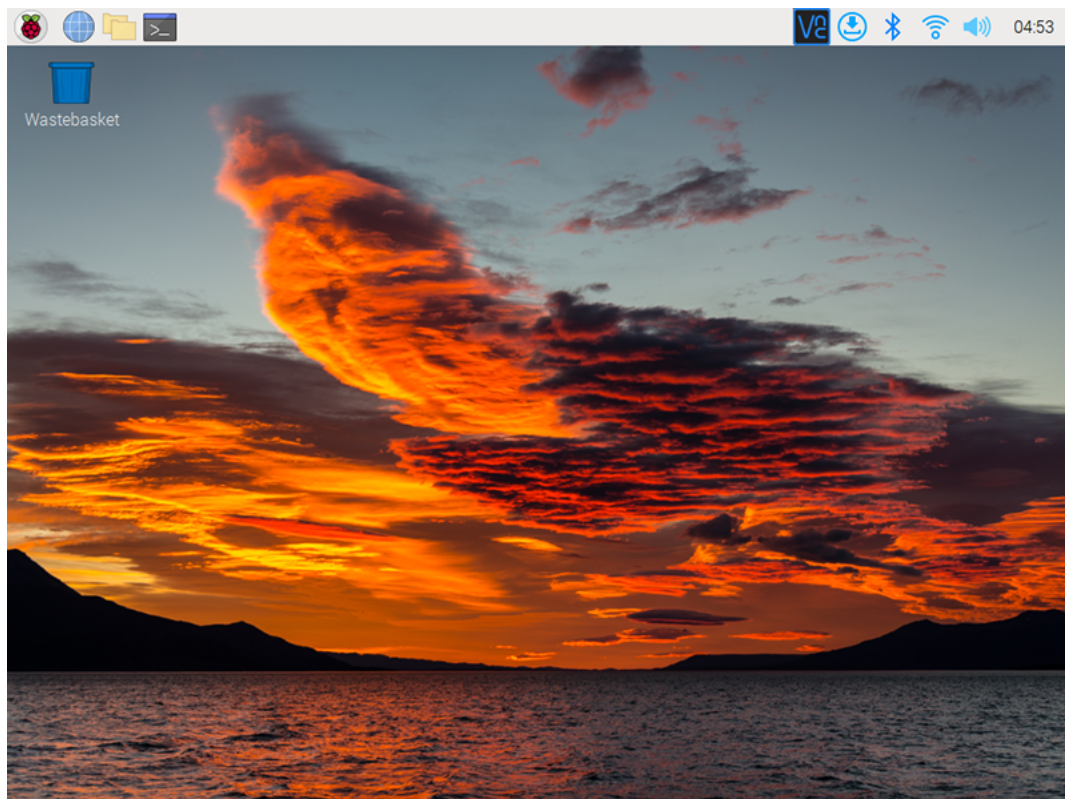
4. Open the VNC server window by clicking on the **VNC** icon, then click on the **Menu** button in the top right corner and select **Options**.
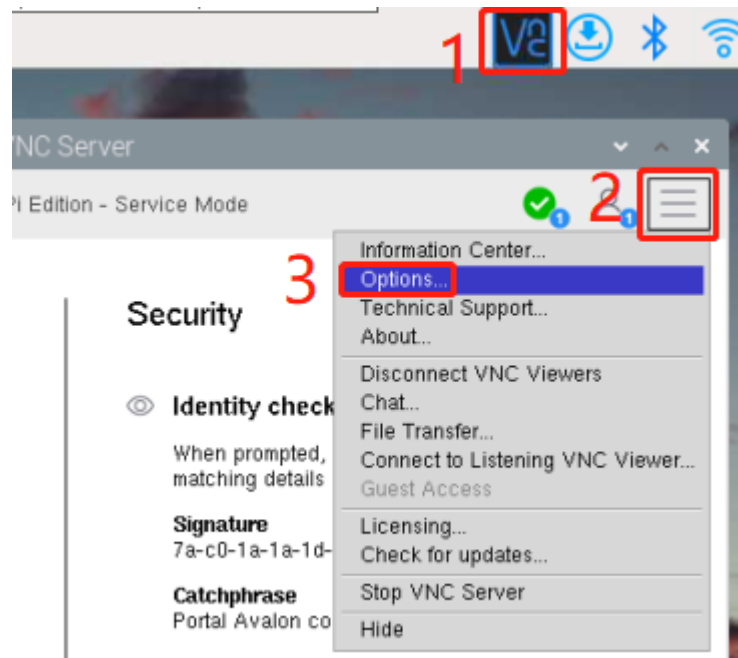
5. You will be presented with the following screen where you can change the options.
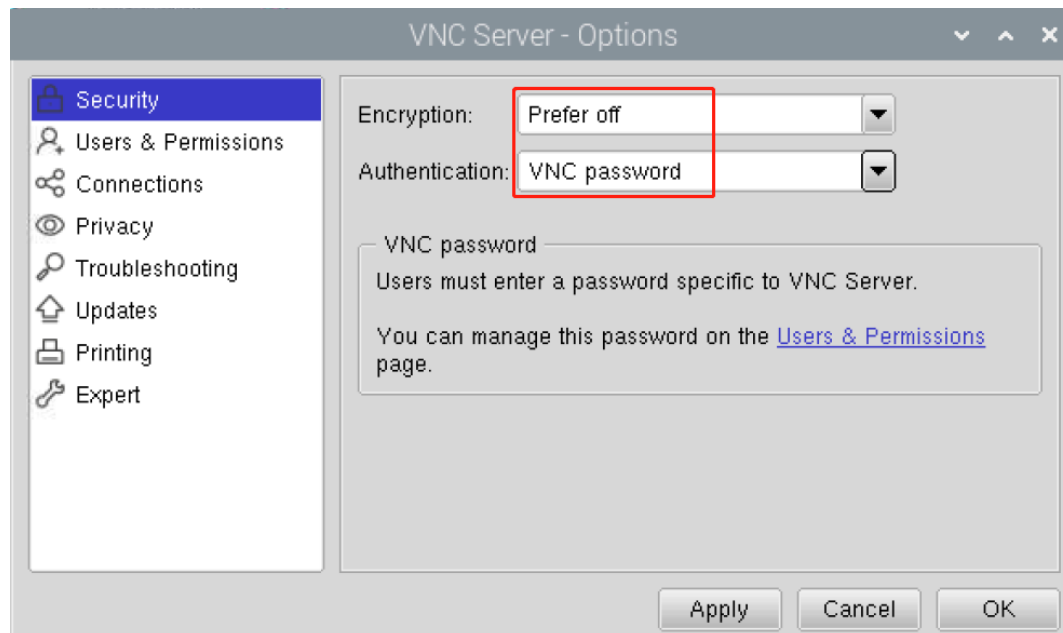
Set **Encryption** to **Prefer off** and **Authentication** to **VNC password**.

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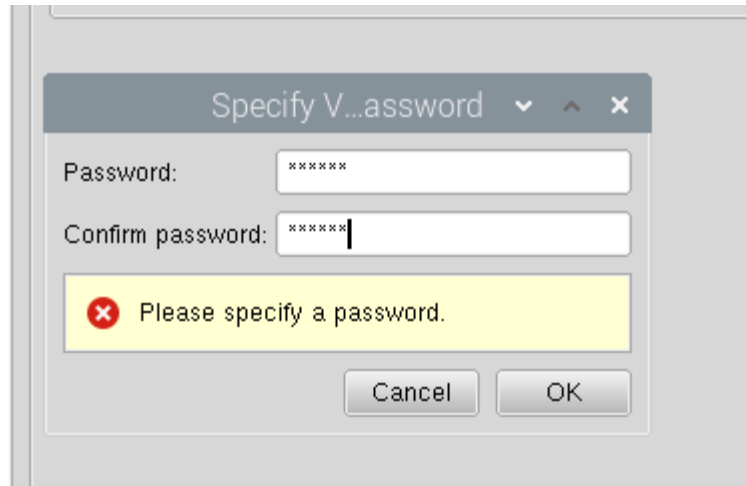
You are now ready to connect from your Mac. It's okay to disconnect the monitor.

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.
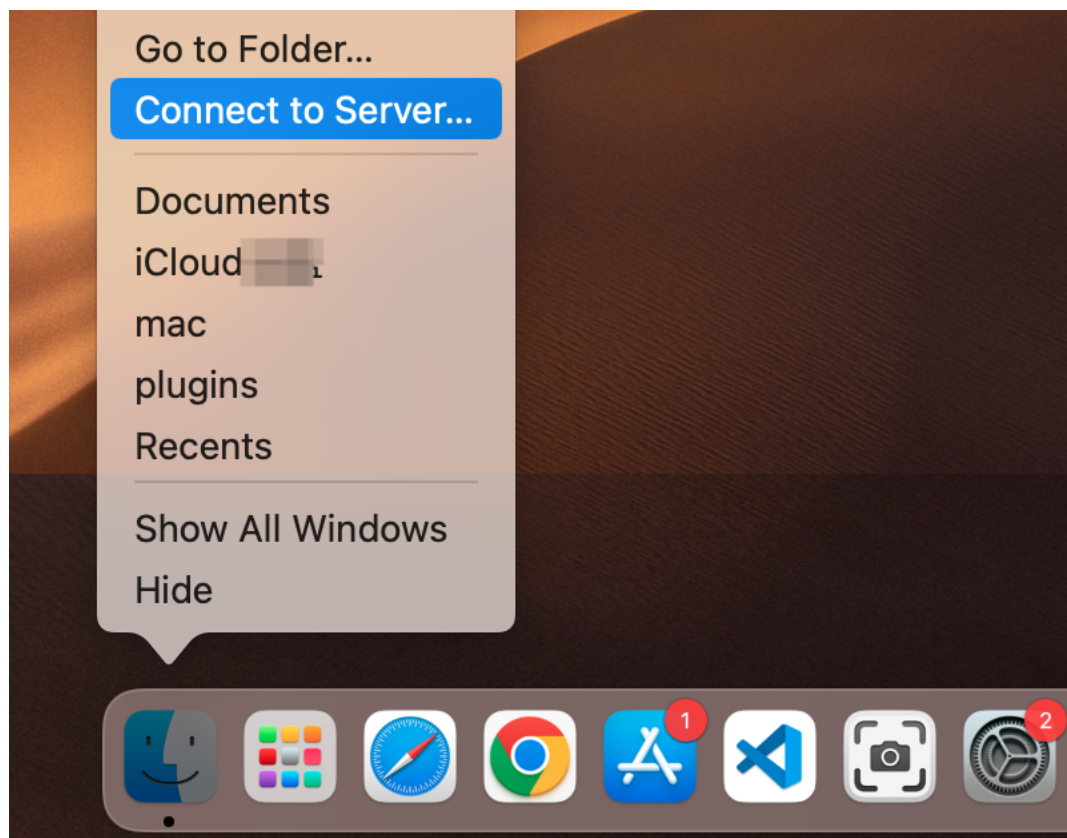
2. Type in `vnc://<username>@<hostname>.local` (or `vnc://<username>@<IP address>`). After entering, click **Connect**.
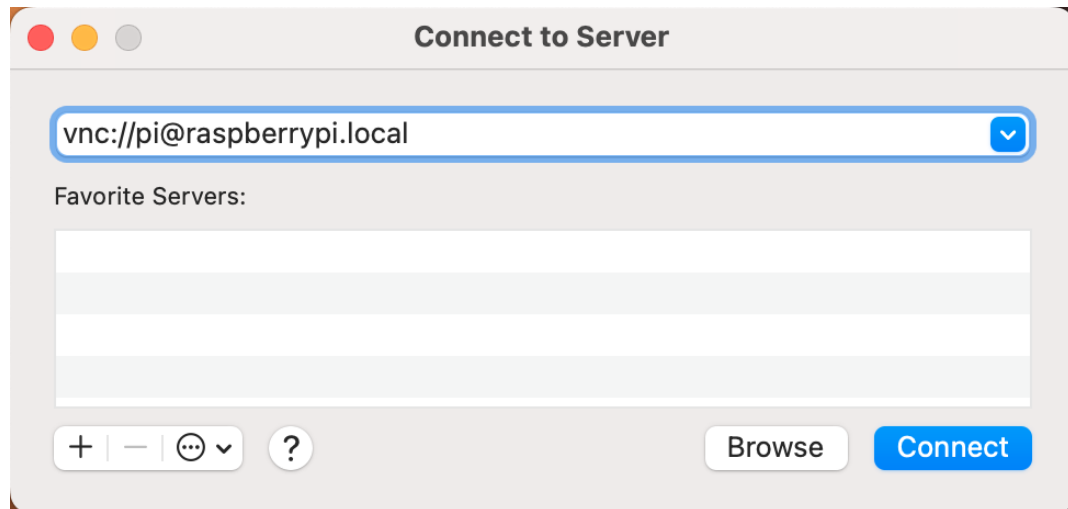
3. You will be asked for a password, so please enter it.
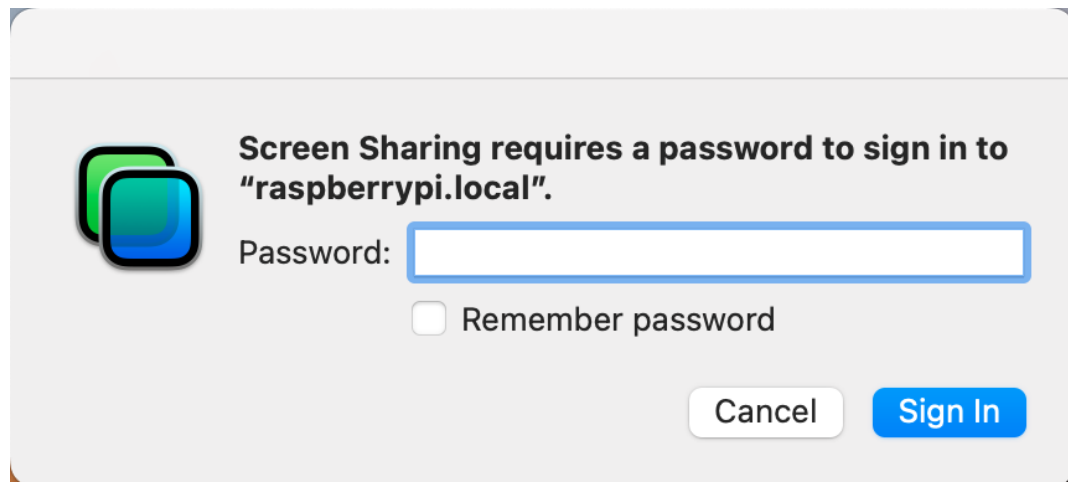
4. The desktop of the Raspberry pi will be displayed, and you will be able to operate it from the Mac as it is.
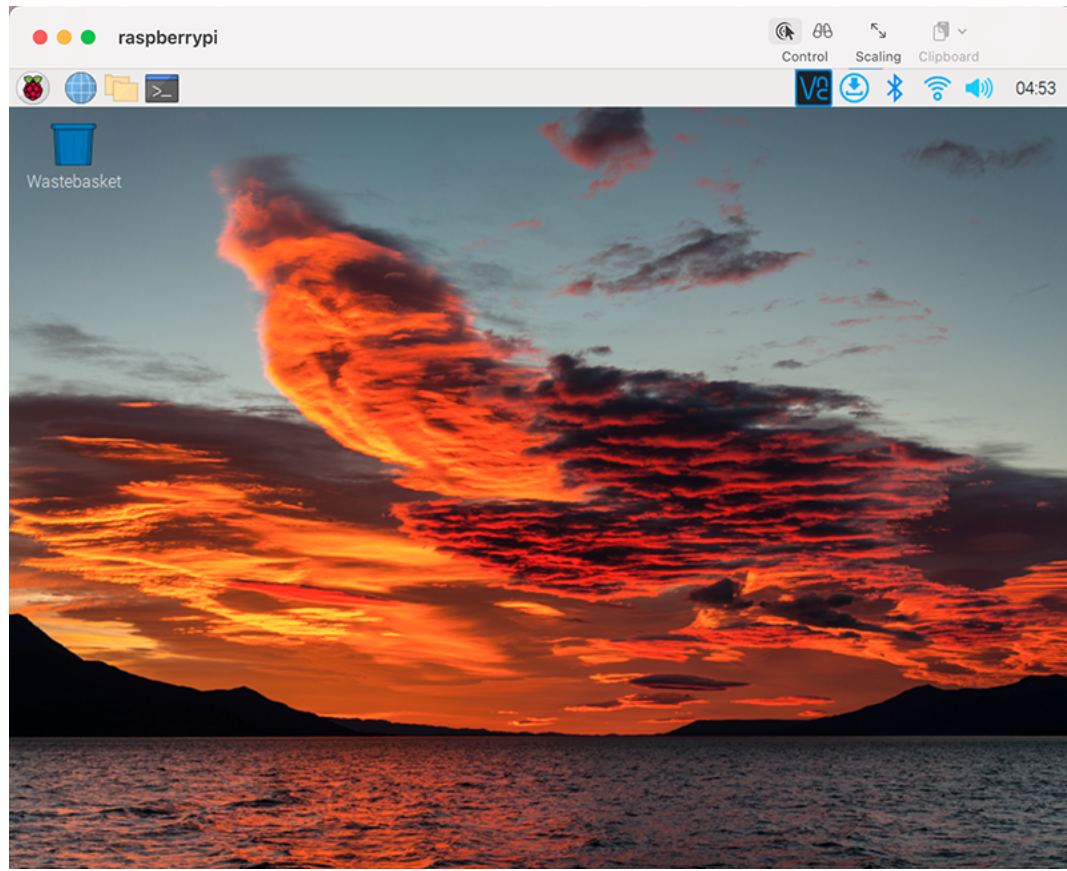
Don't Have Temporarily Monitor (or TV)?

- You can apply the SSH command to open the Raspberry Pi's Bash shell.

- Bash is the standard default shell for Linux.

- The shell itself is a command (instruction) 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
```
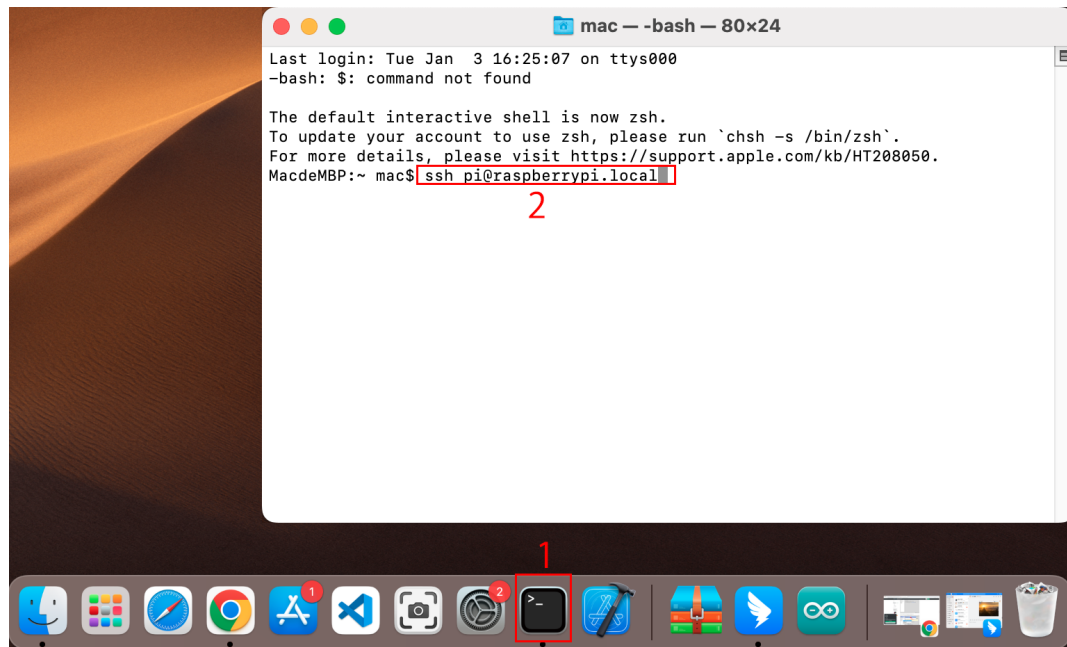
2. The following message will be displayed only when you log in for the first time, so enter **yes**.

```
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 **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 **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.

---

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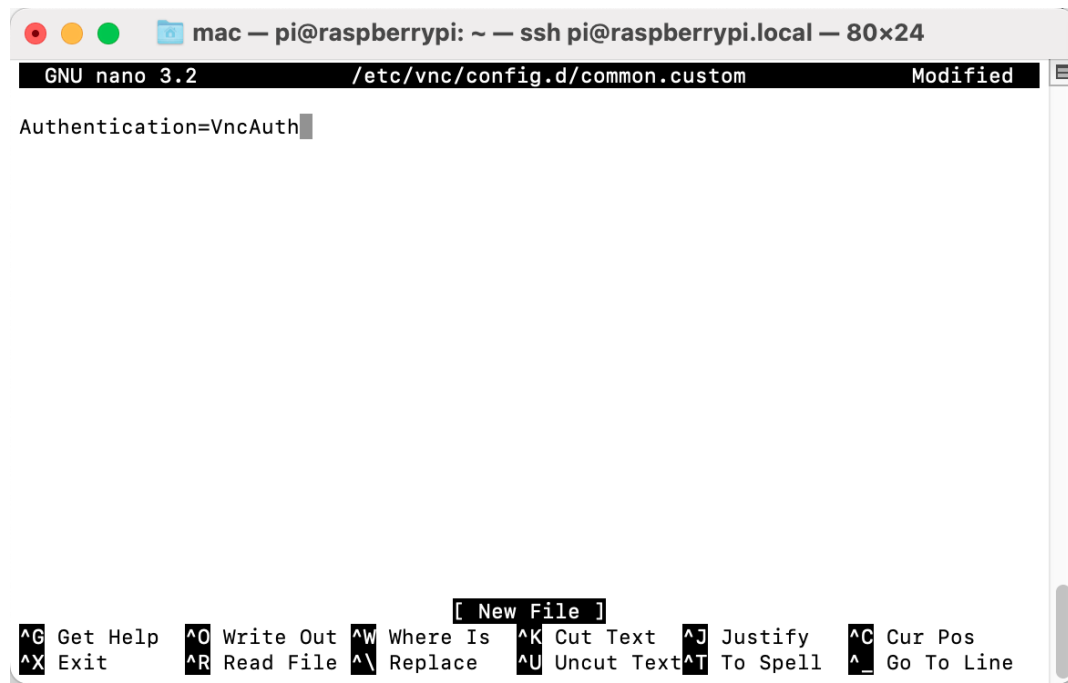
10. In addition, set a password for logging in via VNC from a Mac. You can use the same password as the Raspberry

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.
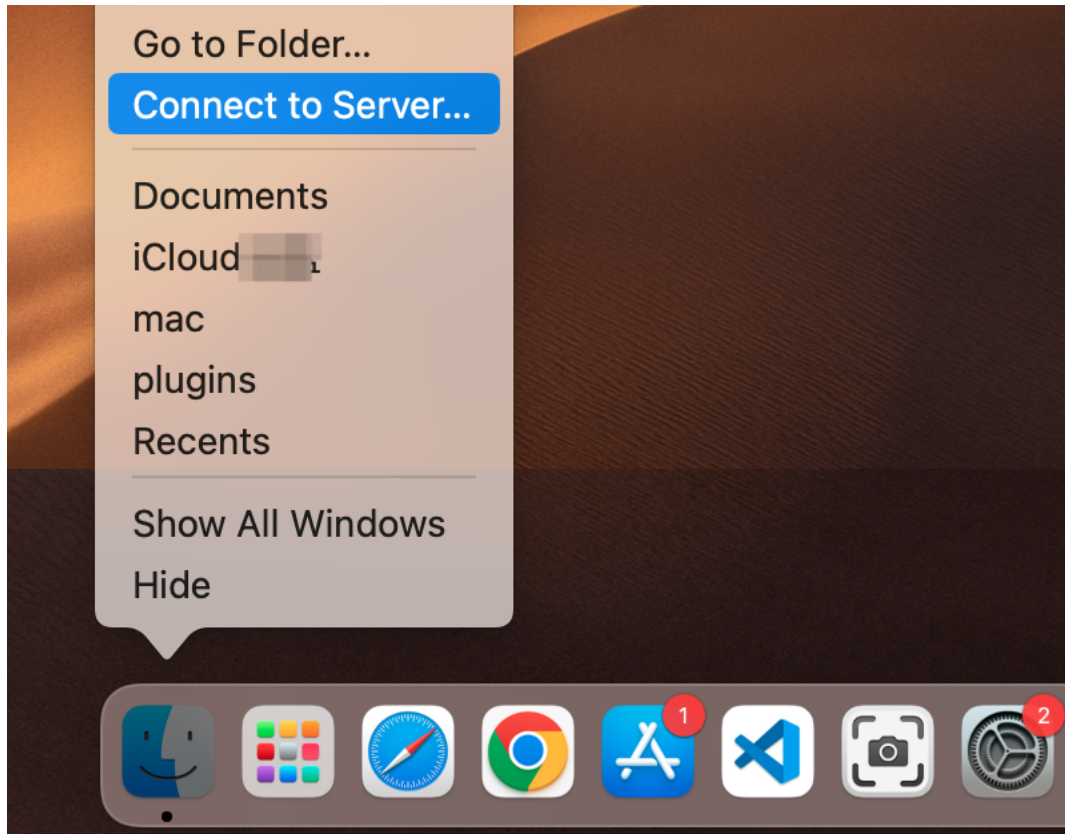
13. Type in `vnc://<username>@<hostname>.local` (or `vnc://<username>@<IP address>`). After entering, click **Connect**.
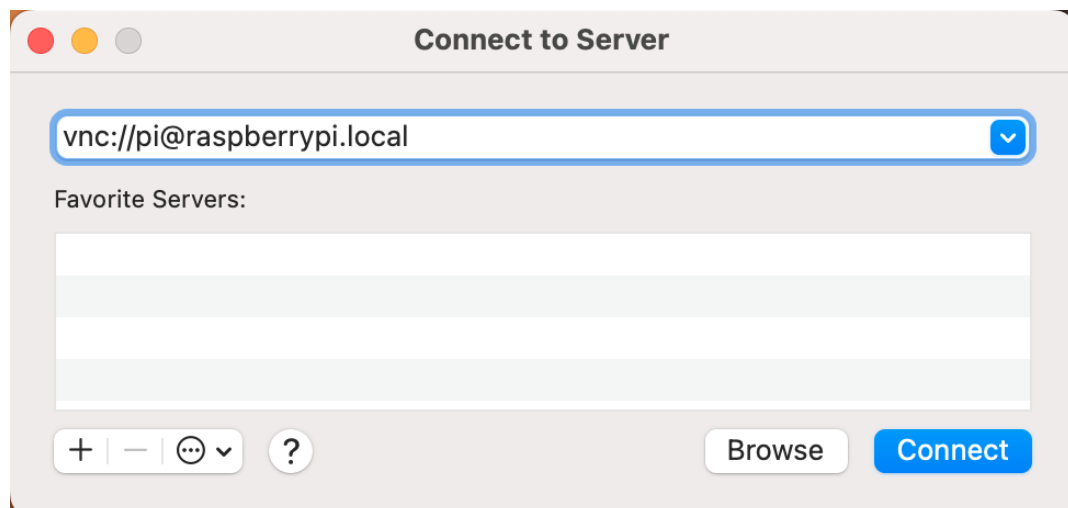
14. You will be asked for a password, so please enter it.
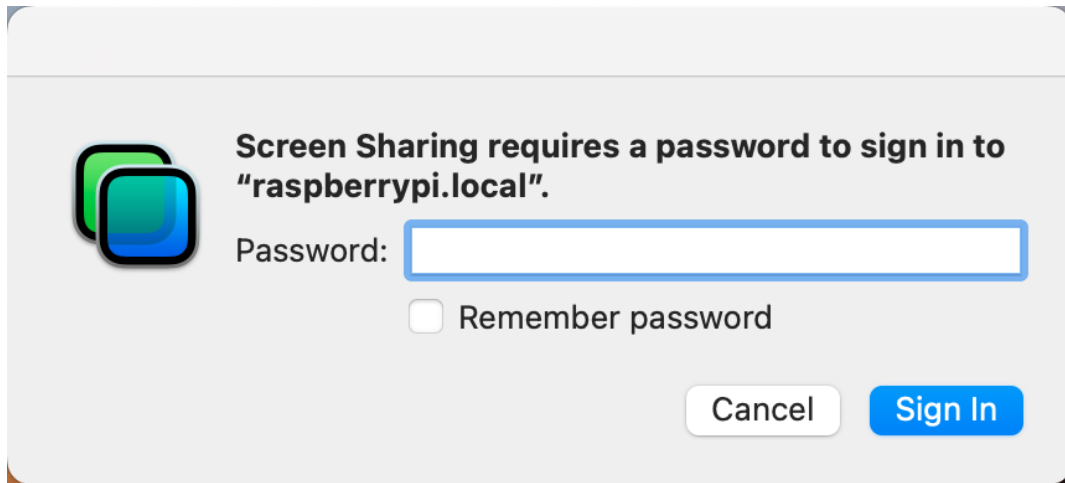
15. The desktop of the Raspberry pi will be displayed, and you will be able to operate it from the Mac as it is.

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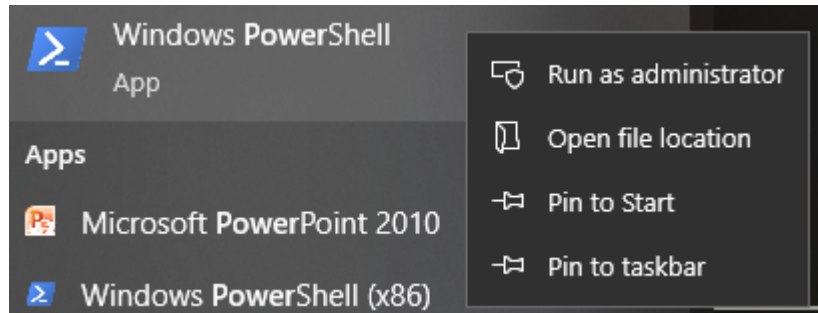
2. Then, check the IP address of your Raspberry Pi by typing in `ping -4 <hostname>.local`.

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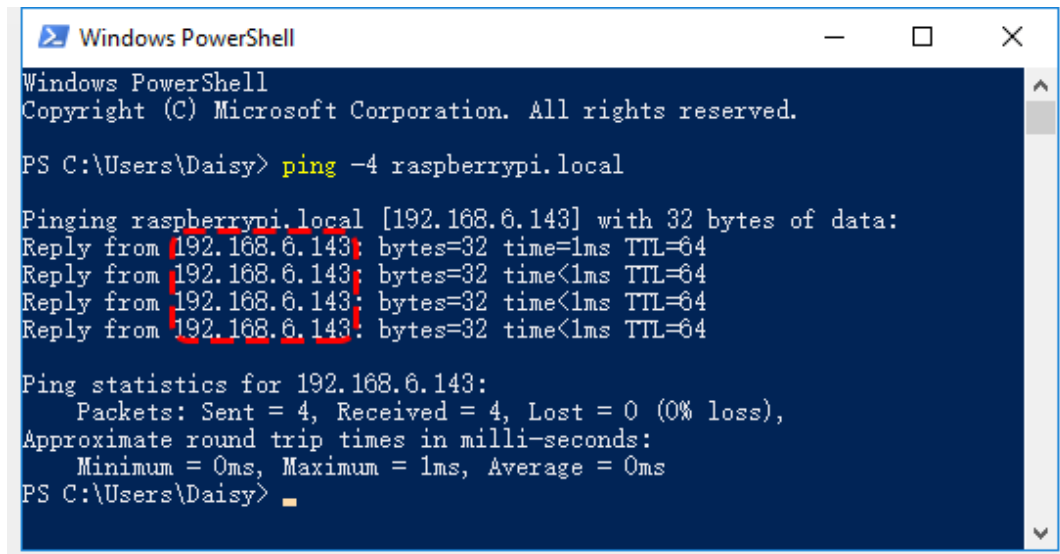

As shown above, you can see the Raspberry Pi's IP address after it has been connected to the network.
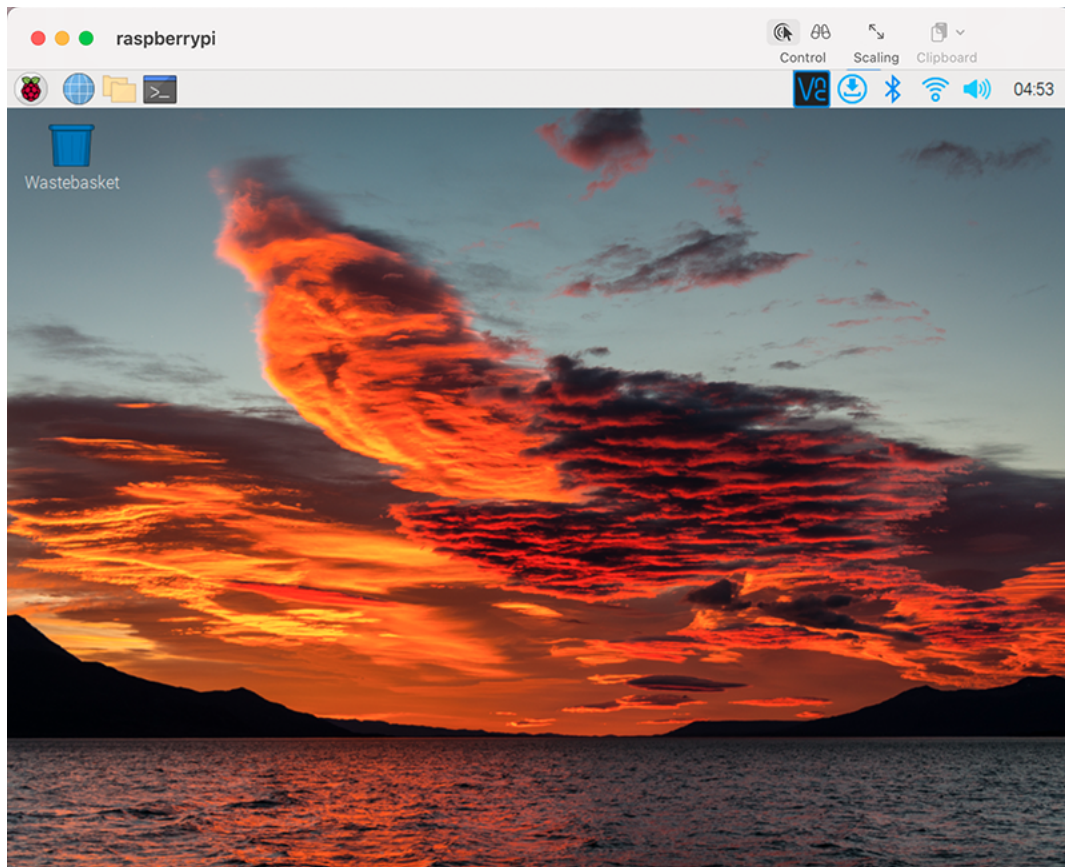
- If terminal prompts `Ping request could not find host pi.local. Please check the name and try again.`. Please follow the prompts to make sure the hostname you fill in is correct.

- 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.

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.
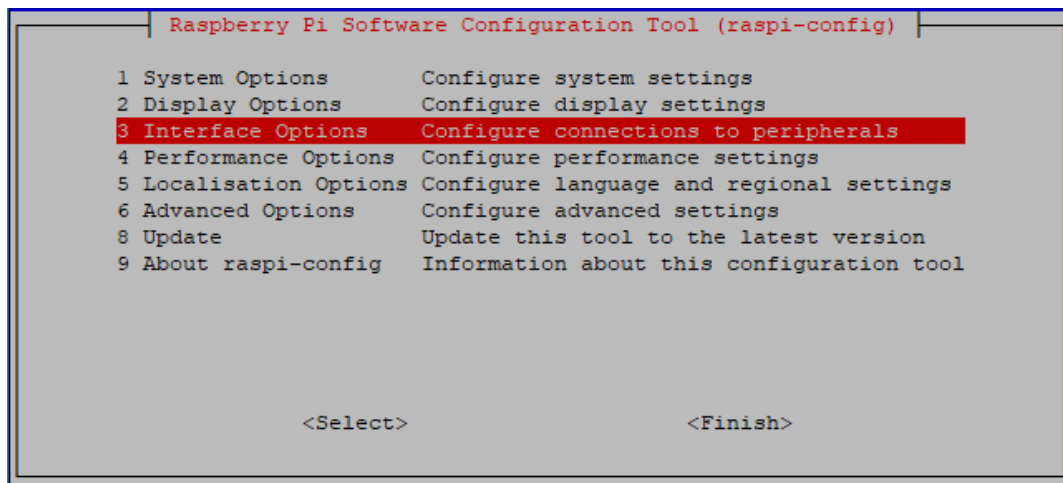
1. Input the following command:

```
sudo raspi-config
```

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

3. Then **VNC**.

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

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.

3. After entering your Raspberry Pi name and password, click **OK**.
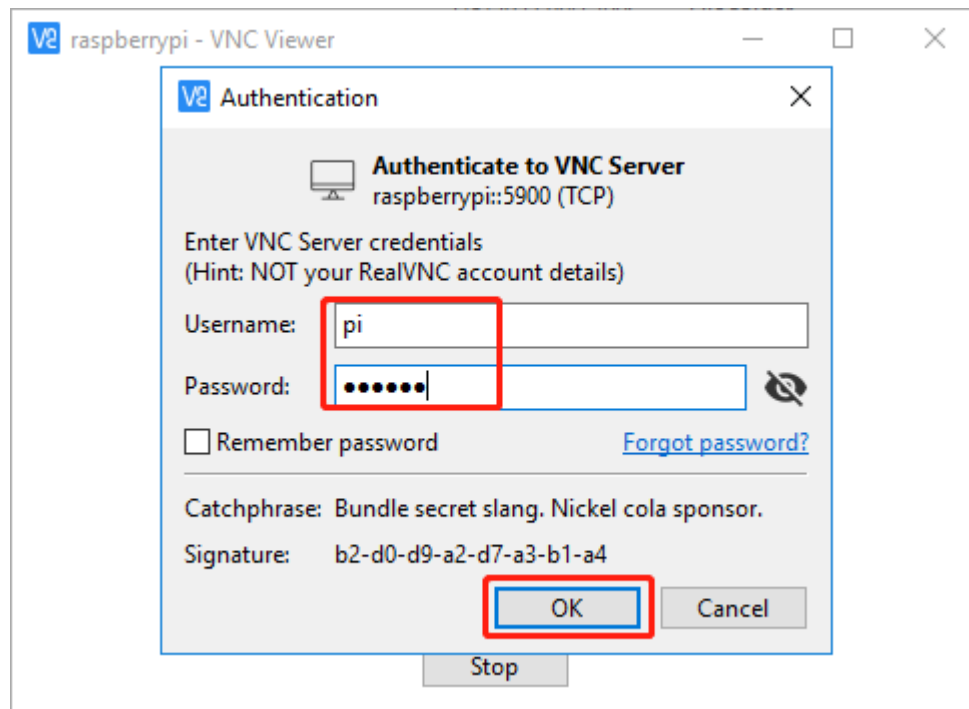
4. Now you can see the desktop of the Raspberry Pi.
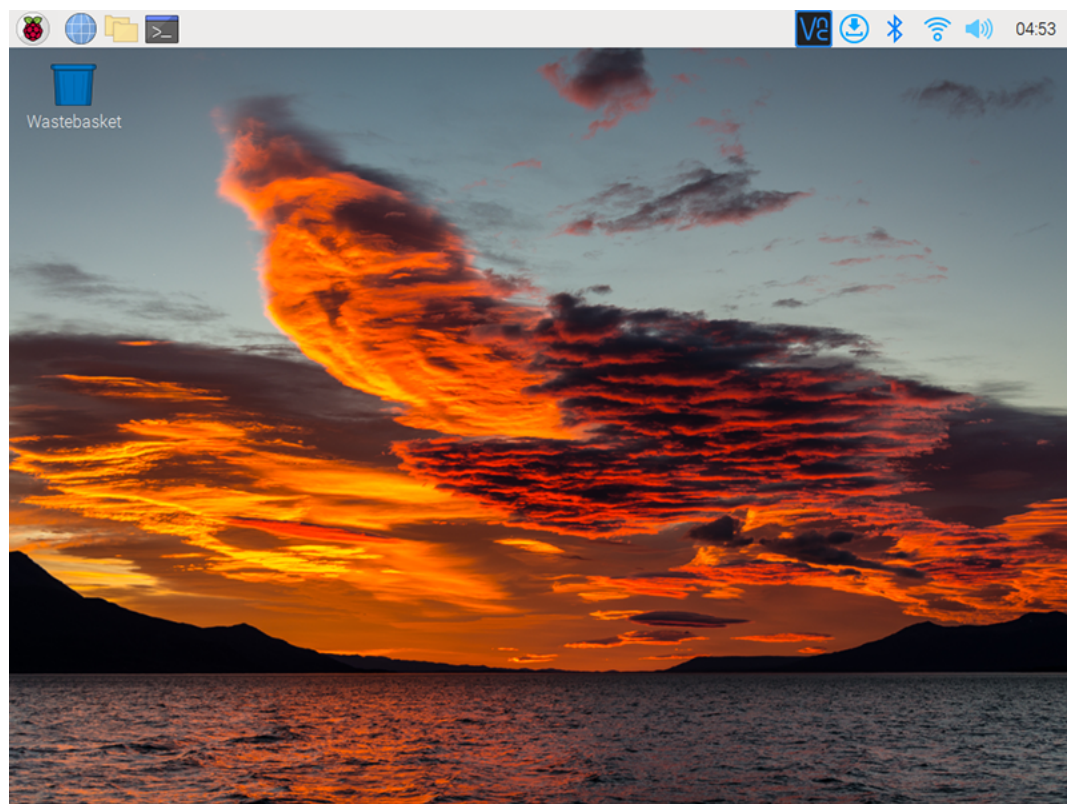
Linux /Unix Users

#. Go to **Applications**->**Utilities**, find the **Terminal**, and open it.
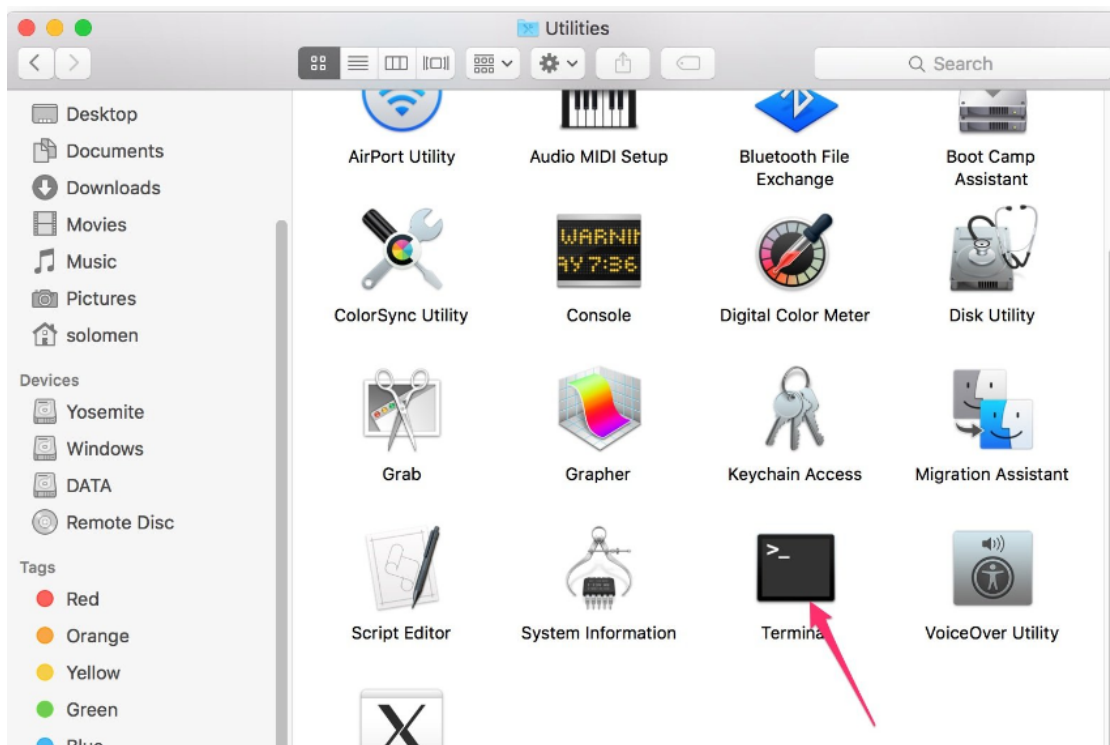
1. Check if your Raspberry Pi is on the same network by type in `ping <hostname>.local`.

```
ping raspberrypi.local
```
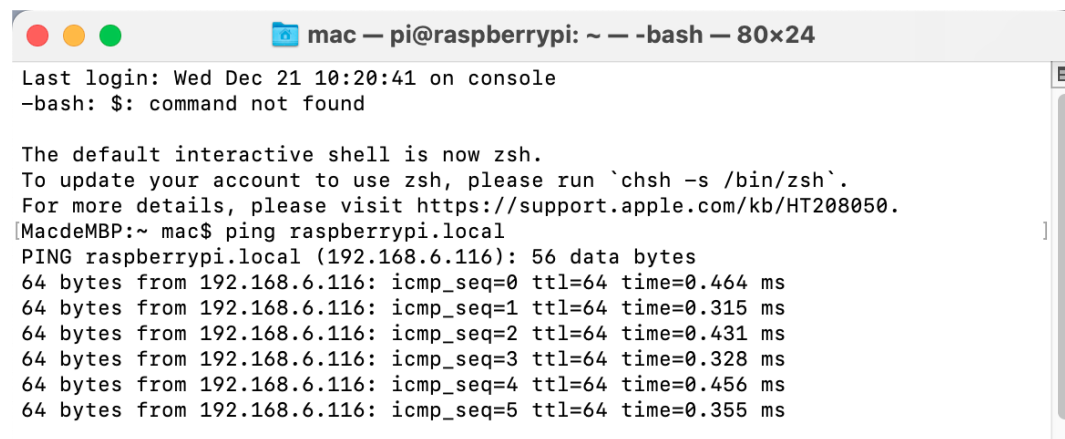

As shown above, you can see the Raspberry Pi's IP address after it has been connected to the network.

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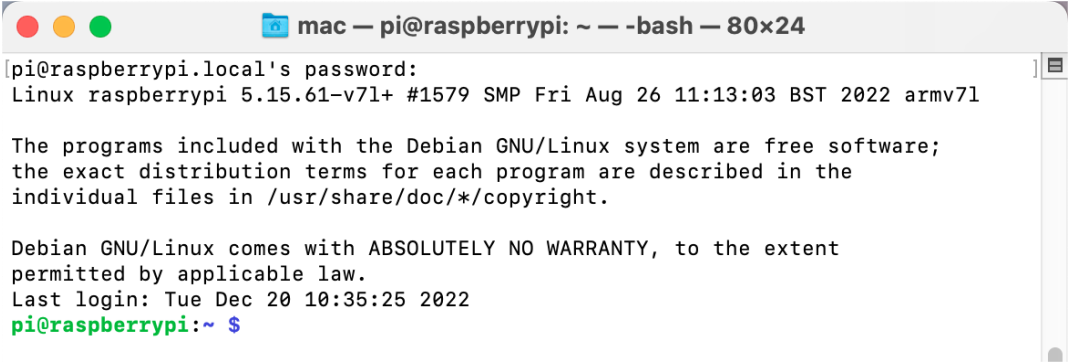
3. The following message will be displayed only when you log in for the first time, so enter `yes`.

```
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.

```
●●●                 mac — pi@raspberrypi: ~ — -bash — 80×24
[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:~ $
```

4.1.4 Install All the Modules(Important)

Make sure you are connected to the Internet and update your system:

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

Note: Python3 related packages must be installed if you are installing the Lite version OS.

```
sudo apt install git python3-pip python3-setuptools python3-smbus
```

Install `robot-hat`.

```
cd ~/
git clone -b v2.0 https://github.com/sunfounder/robot-hat.git
cd robot-hat
sudo python3 setup.py install
```

Then download and install the `vilib` module.

```
cd ~/
git clone -b picamera2 https://github.com/sunfounder/vilib.git
cd vilib
sudo python3 install.py
```
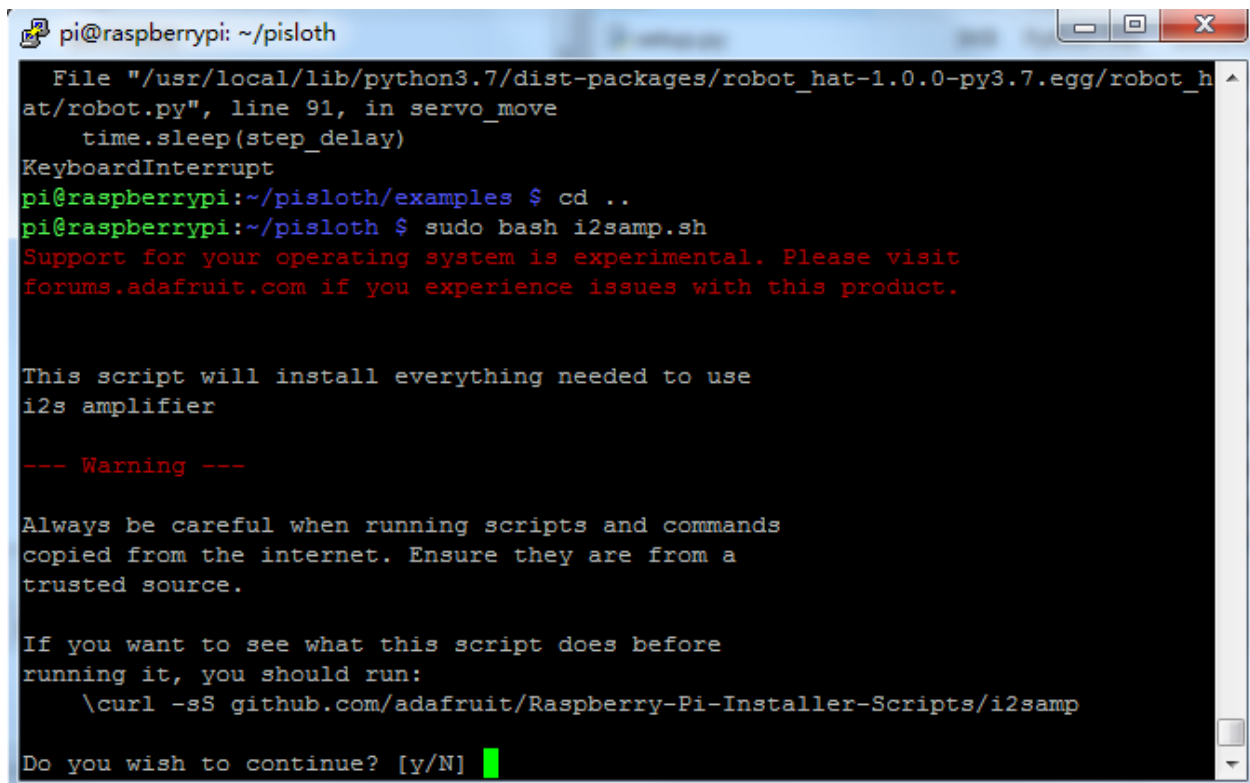
Download and install the `picar-x` module.

```
cd ~/
git clone -b v2.0 https://github.com/sunfounder/picar-x.git
cd picar-x
sudo python3 setup.py install
```

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

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

```
cd ~/picar-x
sudo bash i2samp.sh
```


Type y and press enter to continue running the script.

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

Type y and press enter to restart the Picar-X.

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

4.1.5 Enable I2C Interface(Important)

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

1. Input the following command:

```
sudo raspi-config
```

2. Choose **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 **I2C**.

```
                        pi@raspberrypi: ~                    _  □  ✕

File  Edit  Tabs  Help


    ┤ Raspberry Pi Software Configuration Tool (raspi-config) ├

    P1 Camera                   Enable/Disable connection to the
    P2 SSH                      Enable/Disable remote command lin
    P3 VNC                      Enable/Disable graphical remote a
    P4 SPI                      Enable/Disable automatic loading
    P5 I2C                      Enable/Disable automatic loading
    P6 Serial                   Enable/Disable shell and kernel m
    P7 1-Wire                   Enable/Disable one-wire interface
    P8 Remote GPIO              Enable/Disable remote access to G




                <Select>                        <Back>
```

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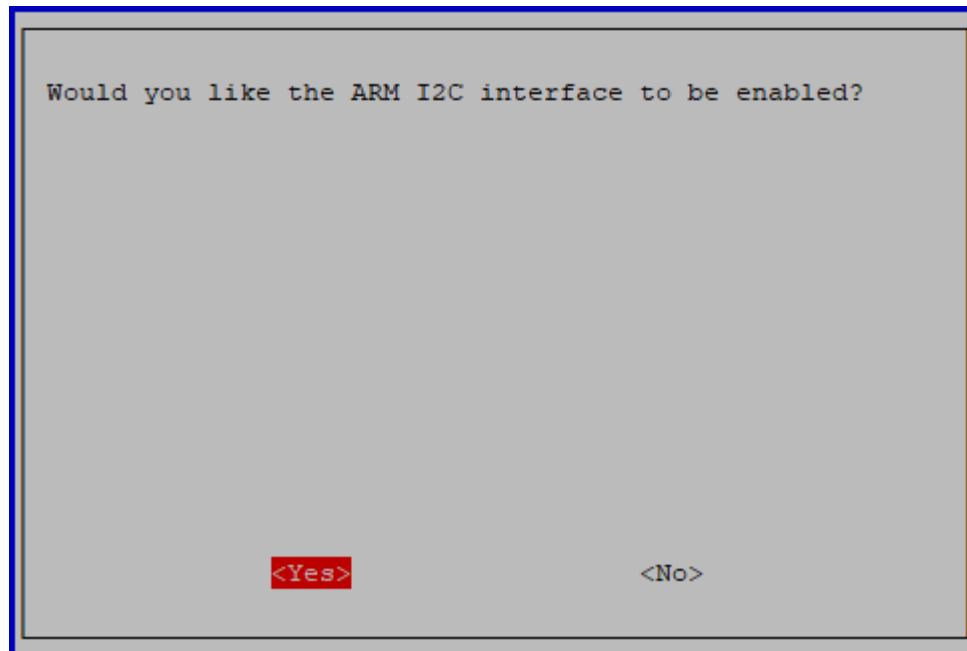
5. After you select **<Finish>**, a pop-up will remind you that you need to reboot for the settings to take effect, select **<Yes>**.

Would you like to reboot now?

<Yes>   <No>
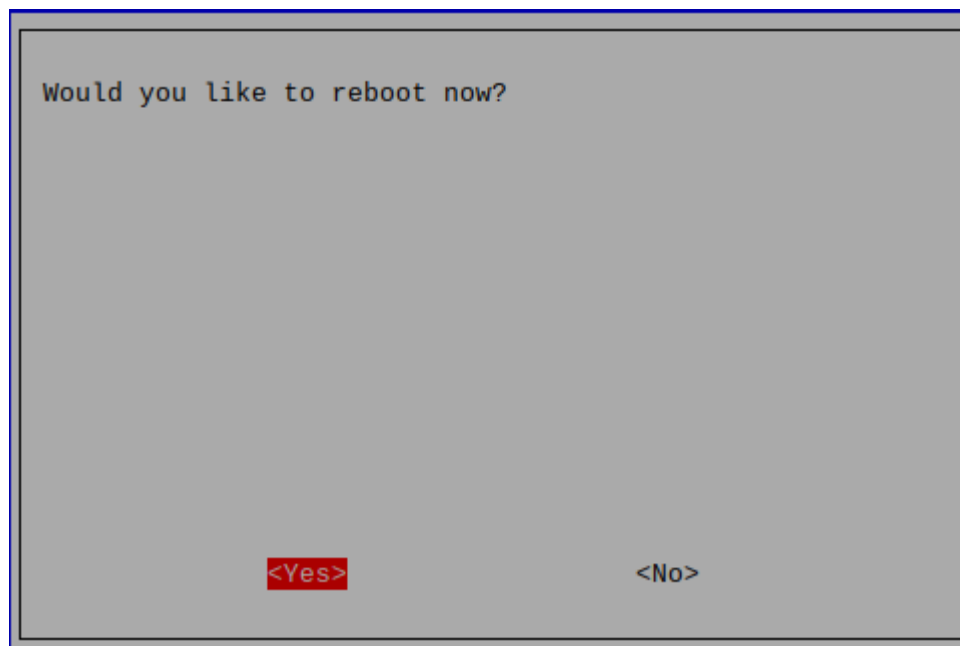
4.1.6 Servo Adjust(Important)

The angle range of the servo is -90~90, but the angle set at the factory is random, maybe 0°, maybe 45°; if we assemble it with such an angle directly, it will lead to a chaotic state after the robot runs the code, or worse, it will cause the servo to block and burn out.

So here we need to set all the servo angles to 0° and then install them, so that the servo angle is in the middle, no matter which direction to turn.

1. To ensure that the servo has been properly set to 0°, first insert the servo arm into the servo shaft and then gently rotate the rocker arm to a different angle. This servo arm is just to allow you to clearly see that the servo is rotating.

2. Now, run `servo_zeroing.py` in the `example/` folder.

```
cd ~/picar-x/example
sudo python3 servo_zeroing.py
```

3. Next, plug the servo cable into the P11 port as follows, at the same time you will see the servo arm rotate to a position(This is the 0° position, which is a random location and may not be vertical or parallel.).
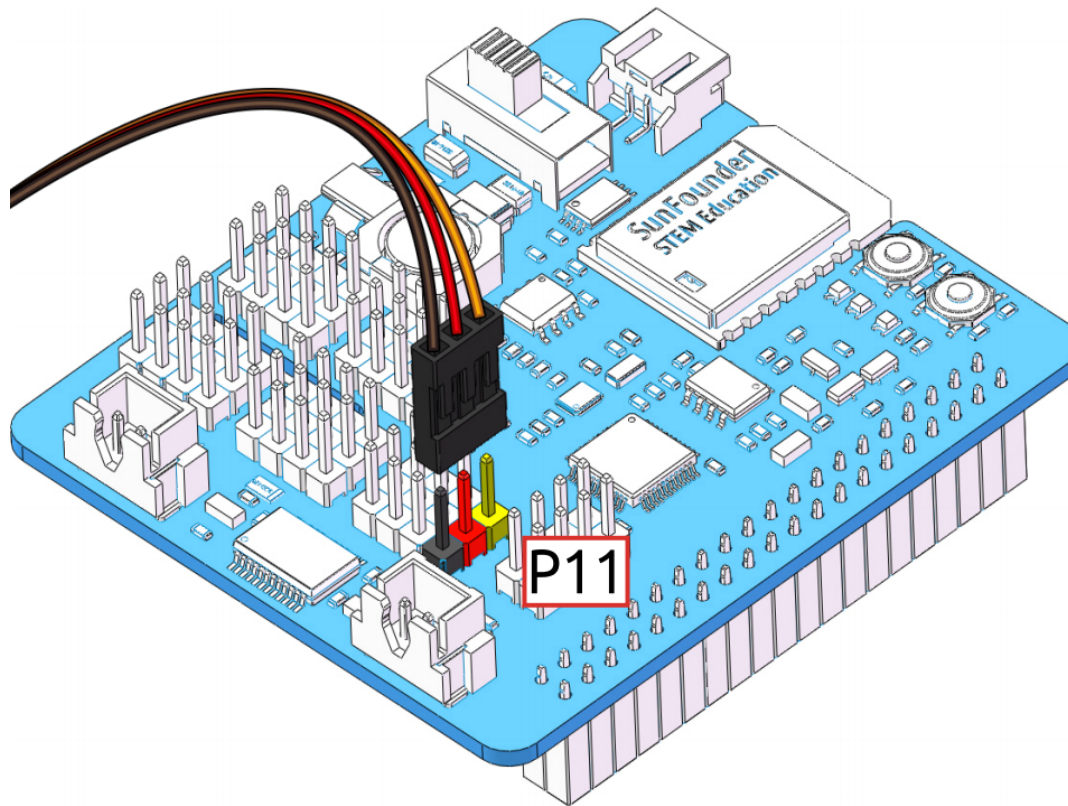
4. Now, remove the servo arm, ensuring the servo wire remains connected, and do not turn off the power. Then continue the assembly following the paper instructions.

Note:

- Do not unplug this servo cable before fixing it with the servo screw, you can unplug it after fixing it.

- Do not rotate the servo while it is powered on to avoid damage; if the servo shaft is not inserted at the right angle, pull the servo out and reinsert it.

- Before assembling each servo, you need to plug the servo cable into P11 and turn on the power to set its angle to 0°.

Video

In our assembly video from 6:25 to 8:48, there is also a detailed tutorial for this chapter. You can follow the video instructions directly.

Note:

- Please only watch the section from 6:25 to 8:48.

- The rest of the video is about PiCar-X 2.0, which may have some assembly steps different from the paper assembly instructions you received. Please refer to the paper manual for accuracy.

After the PiCar-X assembly is completed, try running the projects below:

4.2 0. Calibrating the PiCar-X

4.2.1 Calibrate Motors & Servo
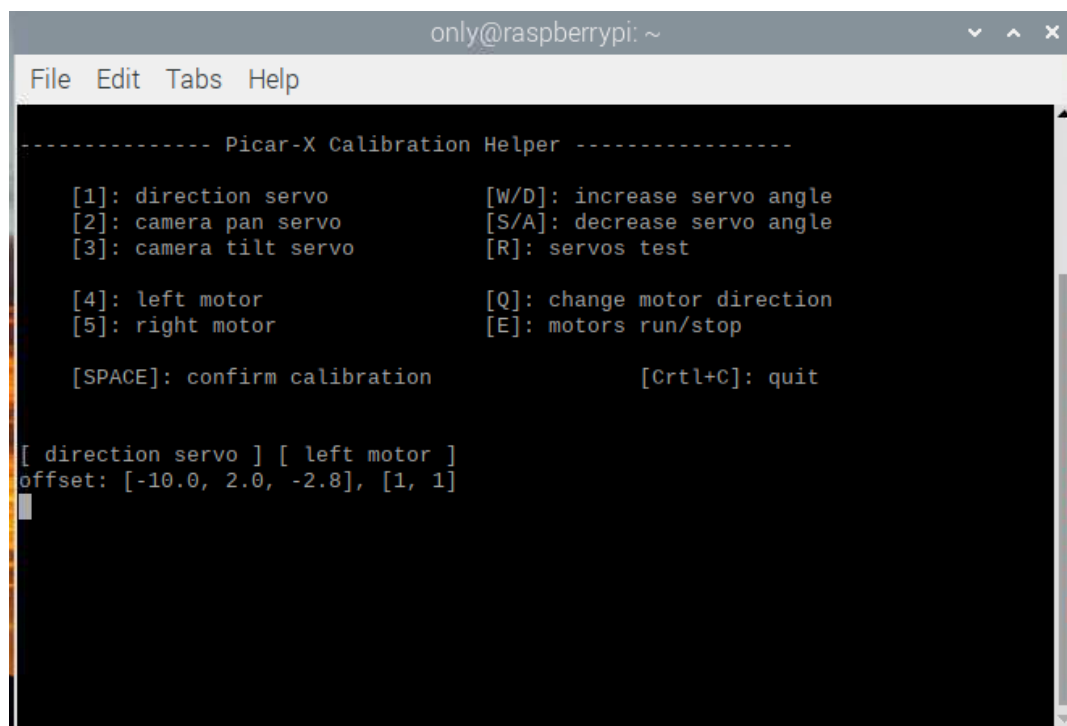
Some servo angles may be slightly tilted due to possible deviations during PiCar-X installation or limitations of the servos themselves, so you can calibrate them.

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

1. Run the `calibration.py`.

```
cd ~/picar-x/example/calibration
sudo python3 calibration.py
```

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

3. The R key is used to test if the 3 servos are working properly. After selecting a servo with the 1, 2 or 3 keys, then press the R key to test that servo.

4. Press the number key 1 to select the front wheel servo, and then press the W/S key to let the front wheel looks as forward as possible without skewing left and right.

5. Press the number key 2 to select the **Pan servo**, then press the W/S key to make the pan/tilt platform look straight ahead and not tilt left or right.

6. Press the number key 3 to select the **tilt servo**, then press the W/S key to make the pan/tilt platform look straight ahead and not tilt up and down.

7. Since the wiring of the motors may be reversed during installation, you can press E to test whether the car can move forward normally. If not, use the number keys 4 and 5 to select the left and right motors, then press the Q key to calibrate the rotation direction.

8. When the calibration is completed, press the Spacebar to save the calibration parameters. There will be a prompt to enter y to confirm, and then press Ctrl+C to exit the program to complete the calibration.

```
xo@picarx: ~/picar-x/example/calibration          ∨  ∧  ✕

File  Edit  Tabs  Help

--------------- Picar-X Calibration Helper ----------------

    [1]: direction servo          [W/D]: increase servo angle
    [2]: camera pan servo          [S/A]: decrease servo angle
    [3]: camera tilt servo         [R]: servos test

    [4]: left motor                [Q]: change motor direction
    [5]: right motor               [E]: motors run/stop

    [SPACE]: confirm calibration             [Crtl+C]: quit


[ direction servo ] [ left motor ]
offset: [0.0, 0.0, 0.0], [1, 1]
Confirm save ?(y/n)
```
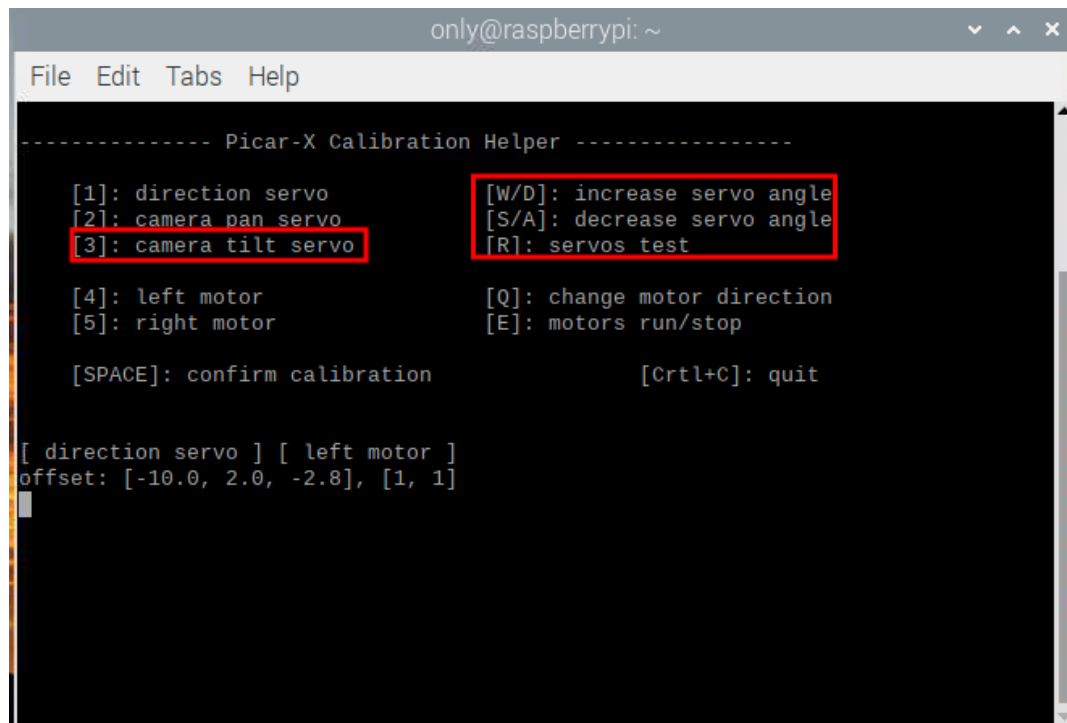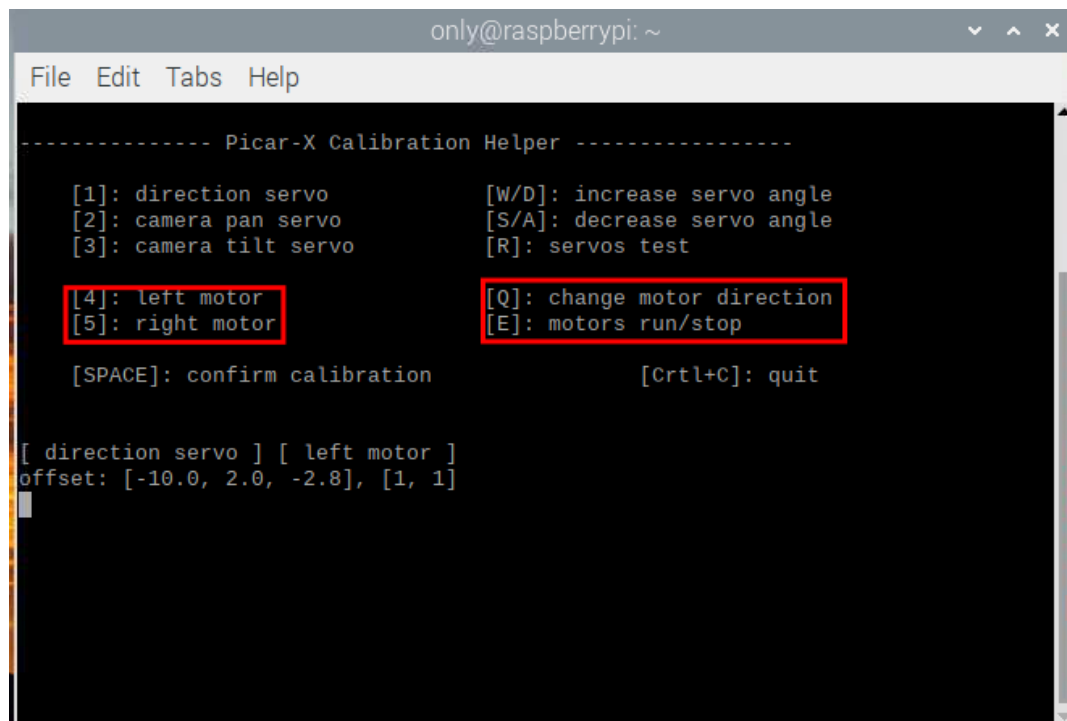
4.2.2 Calibrate Grayscale Module

Due to varying environmental conditions and lighting situations, the preset parameters for the greyscale module might not be optimal. You can fine-tune these settings through this program to achieve better results.

1. Lay down a strip of black electrical tape, about 15cm long, on a light-colored floor. Center your PiCar-X so that it straddles the tape. In this setup, the middle sensor of the greyscale module should be directly above the tape, while the two flanking sensors should hover over the lighter surface.

2. Run the grayscale_calibration.py.

```
cd ~/picar-x/example/calibration
sudo python3 grayscale_calibration.py
```

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

4. Press the "Q" key to initiate the greyscale calibration. You'll then observe the PiCar-X make minor movements to both the left and the right. During this process, each of the three sensors should sweep across the electrical tape at least once.

5. Additionally, you will notice three pairs of significantly different values appearing in the "threshold value" section, while the "line reference" will display two intermediate values, each representing the average of one of these pairs.

6. Next, suspend the PiCar-X in mid-air (or position it over a cliff edge) and press the "E" key. You'll observe that the "cliff reference" values are also updated accordingly.

```
<frozen importlib._bootstrap>:228: RuntimeWarning: Your system is neon capable but
 pygame was not built with support for it. The performance of some of your blits c
ould be adversely affected. Consider enabling compile time detection with environm
ent variables like PYGAME_DETECT_AVX2=1 if you are compiling without cross compila
tion.

        Picar-X Grayscale Module Reference
                Calibration Helper

config_file: /opt/picar-x/picar-x.conf

press [Q] to start line reference calibration,
press [E] to start cliff reference calibration

[SPACE]: confirm calibration            [Crtl+C]: quit

Cliff reference auto calibration done.

current value: [4, 4, 4]
thresholds: [[0, 1517], [0, 1467], [0, 1519]]
line reference: [497, 472, 624]
cliff reference: [250, 238, 314]
```

7. Once you've verified that all the values are accurate, press the "space" key to save the data. You can then exit the program by pressing Ctrl+C.

4.3 1. Let PiCar-X Move

This is the first project, let's test the basic movement of Picar-X.

Run the Code

```
cd ~/picar-x/example
sudo python3 1.move.py
```

After running the code, PiCar-X will move forward, turn in an S-shape, stop and shake its head.

Code

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

```python
from picarx import Picarx
import time


if __name__ == "__main__":
    try:
        px = Picarx()
```

```python
        px.forward(30)
        time.sleep(0.5)
        for angle in range(0,35):
            px.set_dir_servo_angle(angle)
            time.sleep(0.01)
        for angle in range(35,-35,-1):
            px.set_dir_servo_angle(angle)
            time.sleep(0.01)
        for angle in range(-35,0):
            px.set_dir_servo_angle(angle)
            time.sleep(0.01)
        px.forward(0)
        time.sleep(1)

        for angle in range(0,35):
            px.set_camera_servo1_angle(angle)
            time.sleep(0.01)
        for angle in range(35,-35,-1):
            px.set_camera_servo1_angle(angle)
            time.sleep(0.01)
        for angle in range(-35,0):
            px.set_camera_servo1_angle(angle)
            time.sleep(0.01)
        for angle in range(0,35):
            px.set_camera_servo2_angle(angle)
            time.sleep(0.01)
        for angle in range(35,-35,-1):
            px.set_camera_servo2_angle(angle)
            time.sleep(0.01)
        for angle in range(-35,0):
            px.set_camera_servo2_angle(angle)
            time.sleep(0.01)

    finally:
        px.forward(0)
```

How it works?

The basic functionality of PiCar-X is in the `picarx` module, Can be used to control steering gear and wheels, and will make the PiCar-X move forward, turn in an S-shape, or shake its head.

Now, the libraries to support the basic functionality of PiCar-X are imported. These lines will appear in all the examples that involve PiCar-X movement.

```python
from picarx import Picarx
import time
```

The following function with the `for` loop is then used to make PiCar-X move forward, change directions, and move the camera's pan/tilt.

```python
px.forward(speed)
px.set_dir_servo_angle(angle)
px.set_camera_servo1_angle(angle)
px.set_camera_servo2_angle(angle)
```

- `forward()`: Orders the PiCar-X go forward at a given `speed`.

- `set_dir_servo_angle`: Turns the Steering servo to a specific `angle`.

- `set_cam_pan_angle`: Turns the Pan servo to a specific `angle`.

- `set_cam_tilt_angle`: Turns the Tilt servo to a specific `angle`.
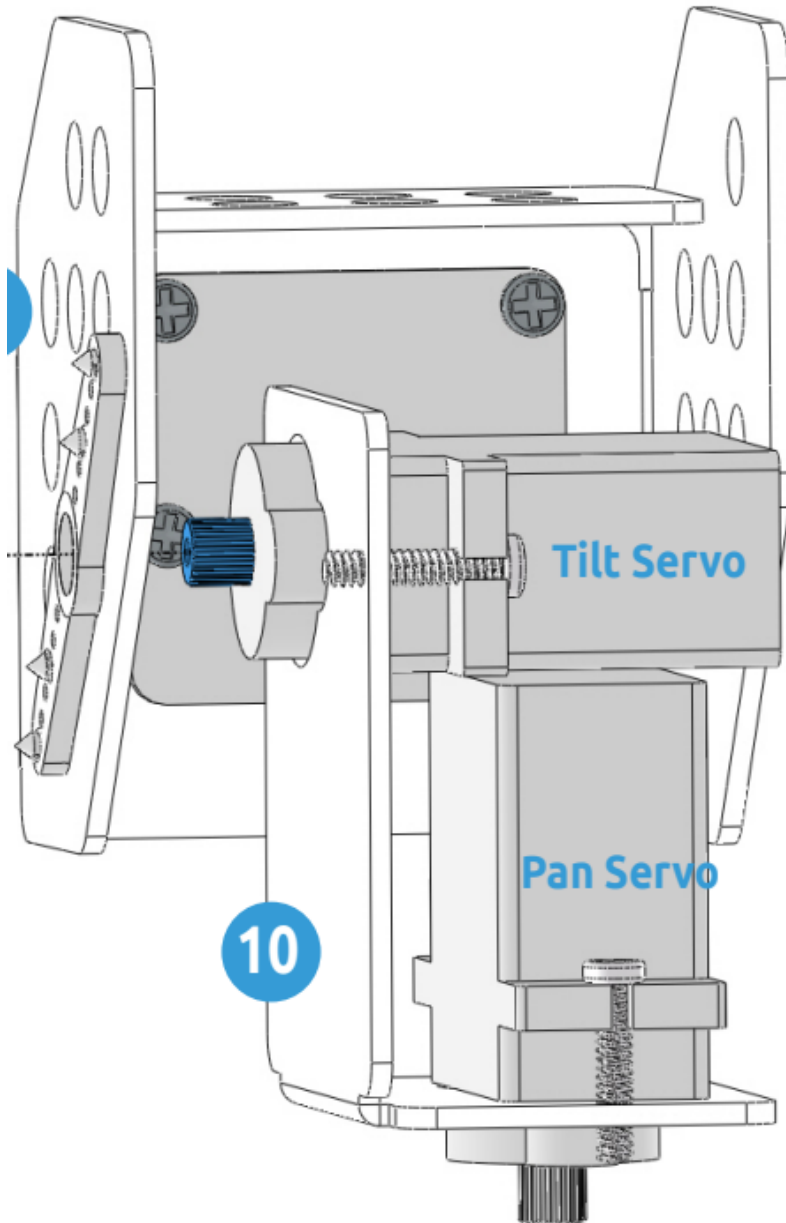
4.4 2. Keyboard Control

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

Run the Code

```
cd ~/picar-x/example
sudo python3 2.keyboard_control.py
```

Press keys on keyboard to control PiCar-X!

- w: Forward

- a: Turn left

- s: Backward

- d: Turn right

- i: Head up

- k: Head down

- j: Turn head left

- l: Turn head right

- ctrl + c: Press twice to exit the program

Code

```python
from picarx import Picarx
from time import sleep
import readchar

manual = '''
Press keys on keyboard to control PiCar-X!
    w: Forward
    a: Turn left
    s: Backward
    d: Turn right
    i: Head up
    k: Head down
    j: Turn head left
    l: Turn head right
    ctrl+c: Quit
'''

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


if __name__ == "__main__":
    try:
        pan_angle = 0
        tilt_angle = 0
```

(continues on next page)

```python
    px = Picarx()
    show_info()
    while True:
        key = readchar.readkey()
        key = key.lower()
        if key in('wsadikjl'):
            if 'w' == key:
                px.set_dir_servo_angle(0)
                px.forward(80)
            elif 's' == key:
                px.set_dir_servo_angle(0)
                px.backward(80)
            elif 'a' == key:
                px.set_dir_servo_angle(-35)
                px.forward(80)
            elif 'd' == key:
                px.set_dir_servo_angle(35)
                px.forward(80)
            elif 'i' == key:
                tilt_angle+=5
                if tilt_angle>35:
                    tilt_angle=35
            elif 'k' == key:
                tilt_angle-=5
                if tilt_angle<-35:
                    tilt_angle=-35
            elif 'l' == key:
                pan_angle+=5
                if pan_angle>35:
                    pan_angle=35
            elif 'j' == key:
                pan_angle-=5
                if pan_angle<-35:
                    pan_angle=-35

            px.set_cam_tilt_angle(tilt_angle)
            px.set_cam_pan_angle(pan_angle)
            show_info()
            sleep(0.5)
            px.forward(0)

        elif key == readchar.key.CTRL_C:
            print("\n Quit")
            break

    finally:
        px.set_cam_tilt_angle(0)
        px.set_cam_pan_angle(0)
        px.set_dir_servo_angle(0)
        px.stop()
        sleep(.2)
```

How it works?

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

```python
while True:
    key = readchar.readkey()
    key = key.lower()
    if key in('wsadikjl'):
        if 'w' == key:
            pass
        elif 's' == key:
            pass
        elif 'a' == key:
            pass
        elif 'd' == key:
            pass
        elif 'i' == key:
            pass
        elif 'k' == key:
            pass
        elif 'l' == key:
            pass
        elif 'j' == key:
            pass

    elif key == readchar.key.CTRL_C:
        print("\n Quit")
        break
```

4.5 3. Text to Speech & Sound Effect

In this example, we use PiCar-X's (to be precise, Robot HAT's) sound effects. It consists of three parts, namely Muisc, Sound, Text to Speech.
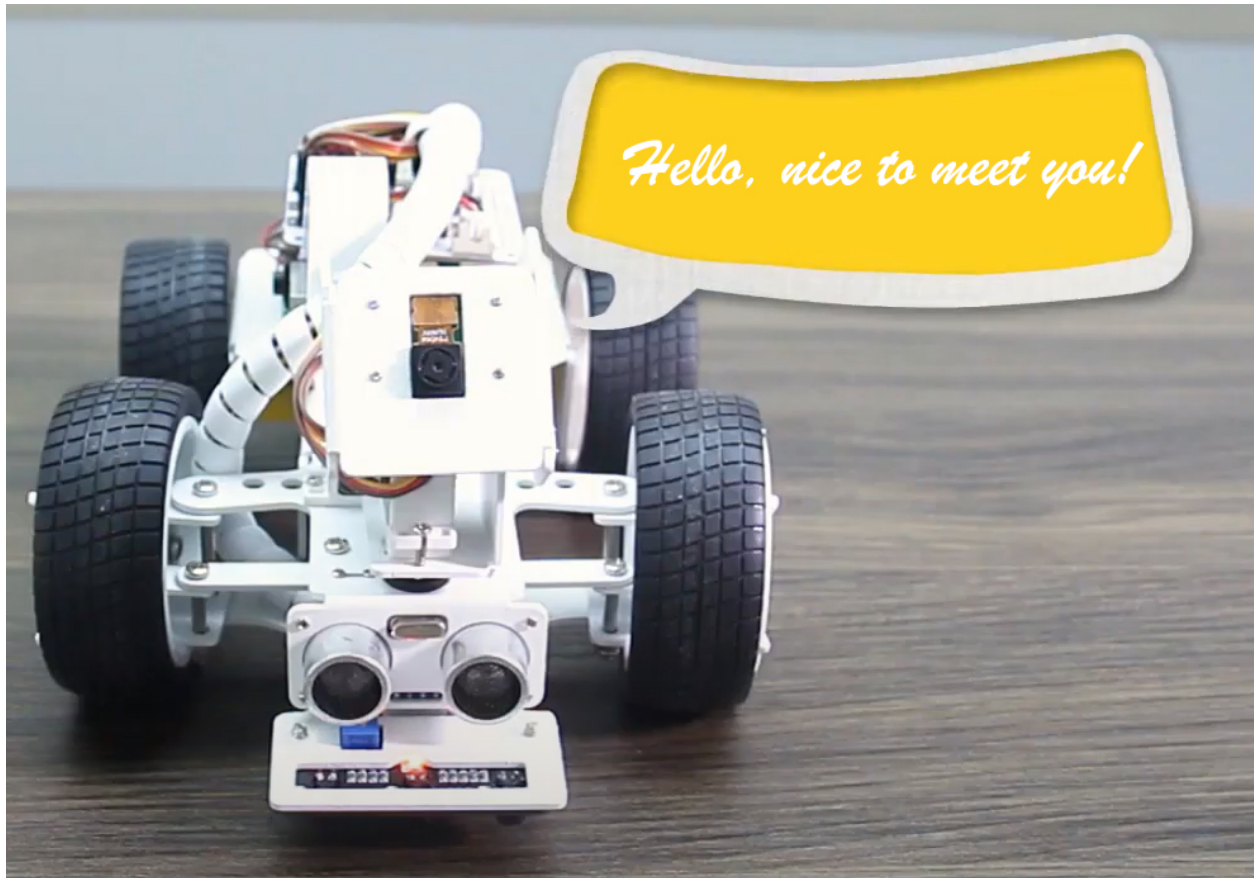
Install i2samp

Before using the Text-to-Speech (TTS) and Sound Effect functions, first activate the speaker so that it will be enabled and can make sounds.

Run `i2samp.sh` in the **picar-x** folder, and this script will install everything needed to use i2s amplifier.

```
cd ~/picar-x
sudo bash i2samp.sh
```

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

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

Run the Code

```
cd ~/picar-x/example
sudo python3 3.tts_example.py
```

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

Input key to call the function!

- space: Play sound effect (Car horn)

- c: Play sound effect with threads

- t: Text to speak (Say Hello)

- q: Play/Stop Music

Code

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

music = Music()
```

(continues on next page)

```python
tts = TTS()

manual = '''
Input key to call the function!
    space: Play sound effect (Car horn)
    c: Play sound effect with threads
    t: Text to speak
    q: Play/Stop Music
'''


def main():
    print(manual)

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


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

        elif key == readchar.key.SPACE:
            music.sound_play('../sounds/car-double-horn.wav')
            sleep(0.05)

        elif key == "c":
            music.sound_play_threading('../sounds/car-double-horn.wav')
            sleep(0.05)

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

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

How it works?

Functions related to background music include these:

- `music = Music()` : Declare the object.

- `music.music_set_volume(20)` : Set the volume, the range is 0~100.

- `music.music_play('../musics/slow-trail-Ahjay_Stelino.mp3')` : Play music files, here is the **slow-trail-Ahjay_Stelino.mp3** file under the `../musics` path.

- `music.music_stop()` : Stop playing background music.

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

Functions related to sound effects include these:

- `music = Music()`

- `music.sound_play('../sounds/car-double-horn.wav')` : Play the sound effect file.

- `muisc.sound_play_threading('../sounds/car-double-horn.wav')` : Play the sound effect file in a new thread mode without suspending the main thread.

The eSpeak software is used to implement the functions of TTS.

Import the TTS module in robot_hat, which encapsulates functions that convert text to speech.

Functions related to Text to Speech include these:

- `tts = TTS()`

- `tts.say(words)` : Text audio.

- `tts.lang("en-US")` : Set the language.

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

Table 1: Language

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

4.6 4. Obstacle Avoidance

In this project, PiCar-X will detect obstacles in front of it while moving forward, and when the obstacles are too close, it will change the direction of moving forward.

Run the Code

```
cd ~/picar-x/example
sudo python3 4.avoiding_obstacles.py
```

After running the code, PiCar-X will walk forward.

If it detects that the distance of the obstacle ahead is less than 20cm, it will go backward.

If there is an obstacle within 20 to 40cm, it will turn left.

If there is no obstacle in the direction after turning left or the obstacle distance is greater than 25cm, it will continue to move forward.

Code

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

```python
from picarx import Picarx
import time

POWER = 50
SafeDistance = 40   # > 40 safe
DangerDistance = 20 # > 20 && < 40 turn around,
                    # < 20 backward

def main():
    try:
        px = Picarx()
        # px = Picarx(ultrasonic_pins=['D2','D3']) # tring, echo

        while True:
            distance = round(px.ultrasonic.read(), 2)
            print("distance: ",distance)
            if distance >= SafeDistance:
                px.set_dir_servo_angle(0)
                px.forward(POWER)
            elif distance >= DangerDistance:
                px.set_dir_servo_angle(30)
                px.forward(POWER)
                time.sleep(0.1)
            else:
                px.set_dir_servo_angle(-30)
                px.backward(POWER)
                time.sleep(0.5)

    finally:
        px.forward(0)


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

How it works?

- Importing the Picarx Module and Initializing Constants:

    This section of the code imports the `Picarx` class from the `picarx` module, which is essential for controlling the Picarx robot. Constants like `POWER`, `SafeDistance`, and `DangerDistance` are defined, which will be used later in the script to control the robot's movement based on distance measurements.

    ```python
    from picarx import Picarx
    import time

    POWER = 50
    SafeDistance = 40 # > 40 safe
    DangerDistance = 20 # > 20 && < 40 turn around,
    # < 20 backward
    ```

- Main Function Definition and Ultrasonic Sensor Reading:

  The `main` function is where the Picarx robot is controlled. An instance of `Picarx` is created, which activates the robot's functionalities. The code enters an infinite loop, constantly reading the distance from the ultrasonic sensor. This distance is used to determine the robot's movement.

```python
def main():
try:
px = Picarx()

    while True:
        distance = round(px.ultrasonic.read(), 2)
        # [Rest of the logic]
```

- Movement Logic Based on Distance:

  The robot's movement is controlled based on the `distance` read from the ultrasonic sensor. If the `distance` is greater than `SafeDistance`, the robot moves forward. If the distance is between `DangerDistance` and `SafeDistance`, it slightly turns and moves forward. If the `distance` is less than `DangerDistance`, the robot reverses while turning in the opposite direction.

```python
if distance >= SafeDistance:
    px.set_dir_servo_angle(0)
    px.forward(POWER)
elif distance >= DangerDistance:
    px.set_dir_servo_angle(30)
    px.forward(POWER)
    time.sleep(0.1)
else:
    px.set_dir_servo_angle(-30)
    px.backward(POWER)
    time.sleep(0.5)
```

- Safety and Cleanup with the 'finally' Block:

  The `try...finally` block ensures safety by stopping the robot's motion in case of an interruption or error. This is a crucial part for preventing uncontrollable behavior of the robot.

```python
try:
# [Control logic]
finally:
px.forward(0)
```

- Execution Entry Point:

  The standard Python entry point `if __name__ == "__main__":` is used to run the main function when the script is executed as a standalone program.

```python
if name == "main":
    main()
```

In summary, the script uses the Picarx module to control a robot, utilizing an ultrasonic sensor for distance measurement. The robot's movement is adapted based on these measurements, ensuring safe operation through careful control and a safety mechanism in the finally block.

4.7 5. Line Tracking

This project will use the Grayscale module to make the PiCar-X move forward along a line. Use dark-colored tape to make a line as straight as possible, and not too curved. Some experimenting might be needed if the PiCar-X is derailed.

Run the Code

```
cd ~/picar-x/example
sudo python3 5.minecart_plus.py
```

After running the code, PiCar-X will move forward along a line.

Code

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

```python
from picarx import Picarx
from time import sleep

px = Picarx()
# px = Picarx(grayscale_pins=['A0', 'A1', 'A2'])

# Please run ./calibration/grayscale_calibration.py to Auto calibrate grayscale values
# or manual modify reference value by follow code
# px.set_line_reference([1400, 1400, 1400])

current_state = None
px_power = 10
offset = 20
last_state = "stop"

def outHandle():
    global last_state, current_state
    if last_state == 'left':
        px.set_dir_servo_angle(-30)
        px.backward(10)
    elif last_state == 'right':
        px.set_dir_servo_angle(30)
        px.backward(10)
    while True:
        gm_val_list = px.get_grayscale_data()
        gm_state = get_status(gm_val_list)
        print("outHandle gm_val_list: %s, %s"%(gm_val_list, gm_state))
        currentSta = gm_state
        if currentSta != last_state:
            break
    sleep(0.001)

def get_status(val_list):
    _state = px.get_line_status(val_list)  # [bool, bool, bool], 0 means line, 1 means
    →background
```

(continues on next page)

```python
    if _state == [0, 0, 0]:
        return 'stop'
    elif _state[1] == 1:
        return 'forward'
    elif _state[0] == 1:
        return 'right'
    elif _state[2] == 1:
        return 'left'

if __name__=='__main__':
    try:
        while True:
            gm_val_list = px.get_grayscale_data()
            gm_state = get_status(gm_val_list)
            print("gm_val_list: %s, %s"%(gm_val_list, gm_state))

            if gm_state != "stop":
                last_state = gm_state

            if gm_state == 'forward':
                px.set_dir_servo_angle(0)
                px.forward(px_power)
            elif gm_state == 'left':
                px.set_dir_servo_angle(offset)
                px.forward(px_power)
            elif gm_state == 'right':
                px.set_dir_servo_angle(-offset)
                px.forward(px_power)
            else:
                outHandle()
    finally:
        px.stop()
        print("stop and exit")
        sleep(0.1)
```

How it works?

This Python script controls a Picarx robot car using grayscale sensors for navigation. Here's a breakdown of its main components:

- Import and Initialization:

    The script imports the Picarx class for controlling the robot car and the sleep function from the time module for adding delays.

    An instance of Picarx is created, and there's a commented line showing an alternative initialization with specific grayscale sensor pins.

```python
from picarx import Picarx
from time import sleep

px = Picarx()
```

- Configuration and Global Variables:

current_state, px_power, offset, and last_state are global variables used to track and control the car's movement. px_power sets the motor power, and offset is used for adjusting the steering angle.

```
current_state = None
px_power = 10
offset = 20
last_state = "stop"
```

- outHandle Function:

    This function is called when the car needs to handle an 'out of line' scenario.

    It adjusts the car's direction based on last_state and checks the grayscale sensor values to determine the new state.

```
def outHandle():
    global last_state, current_state
    if last_state == 'left':
        px.set_dir_servo_angle(-30)
        px.backward(10)
    elif last_state == 'right':
        px.set_dir_servo_angle(30)
        px.backward(10)
    while True:
        gm_val_list = px.get_grayscale_data()
        gm_state = get_status(gm_val_list)
        print("outHandle gm_val_list: %s, %s"%(gm_val_list, gm_state))
        currentSta = gm_state
        if currentSta != last_state:
            break
    sleep(0.001)
```

- get_status Function:

    It interprets the grayscale sensor data (val_list) to determine the car's navigation state.

    The car's state can be 'forward', 'left', 'right', or 'stop', based on which sensor detects the line.

```
def get_status(val_list):
    _state = px.get_line_status(val_list)  # [bool, bool, bool], 0 means
→line, 1 means background
    if _state == [0, 0, 0]:
        return 'stop'
    elif _state[1] == 1:
        return 'forward'
    elif _state[0] == 1:
        return 'right'
    elif _state[2] == 1:
        return 'left'
```

- Main Loop:

    The while True loop continuously checks the grayscale data and adjusts the car's movement accordingly.

    Depending on the gm_state, it sets the steering angle and movement direction.

```
if __name__=='__main__':
    try:
        while True:
            gm_val_list = px.get_grayscale_data()
            gm_state = get_status(gm_val_list)
            print("gm_val_list: %s, %s"%(gm_val_list, gm_state))

            if gm_state != "stop":
                last_state = gm_state

            if gm_state == 'forward':
                px.set_dir_servo_angle(0)
                px.forward(px_power)
            elif gm_state == 'left':
                px.set_dir_servo_angle(offset)
                px.forward(px_power)
            elif gm_state == 'right':
                px.set_dir_servo_angle(-offset)
                px.forward(px_power)
            else:
                outHandle()
```

- Safety and Cleanup:

    The `try...finally` block ensures the car stops when the script is interrupted or finished.

```
finally:
px.stop()
print("stop and exit")
sleep(0.1)
```

In summary, the script uses grayscale sensors to navigate the Picarx robot car. It continuously reads the sensor data to determine the direction and adjusts the car's movement and steering accordingly. The outHandle function provides additional logic for situations where the car needs to adjust its path significantly.

4.8 6. Cliff Detection

Let us give PiCar-X a little self-protection awareness and let it learn to use its own grayscale module to avoid rushing down the cliff.

In this example, the car will be dormant. If you push it to a cliff, it will be awakened urgently, then back up, and say "danger".

Run the Code

```
cd ~/picar-x/example
sudo python3 6.cliff_detection.py
```

Code

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

```python
from picarx import Picarx
from time import sleep
from robot_hat import TTS

tts = TTS()
tts.lang("en-US")


px = Picarx()
# px = Picarx(grayscale_pins=['A0', 'A1', 'A2'])
# manual modify reference value
px.set_cliff_reference([200, 200, 200])

current_state = None
px_power = 10
offset = 20
last_state = "safe"

if __name__=='__main__':
    try:
        while True:
            gm_val_list = px.get_grayscale_data()
            gm_state = px.get_cliff_status(gm_val_list)
            # print("cliff status is:  %s"%gm_state)

            if gm_state is False:
                state = "safe"
                px.stop()
            else:
                state = "danger"
                px.backward(80)
                if last_state == "safe":
                    tts.say("danger")
                    sleep(0.1)
            last_state = state

    finally:
        px.stop()
        print("stop and exit")
        sleep(0.1)
```

How it works?

The function to detect the cliff looks like this:

- `get_grayscale_data()`: This method directly outputs the readings of the three sensors, from right to left. The brighter the area, the larger the value obtained.

- `get_cliff_status(gm_val_list)`: This method compares the readings from the three probes and outputs a result. If the result is true, it is detected that there is a cliff in front of the car.

4.9 7. Computer Vision

This project will officially enter the field of computer vision!

Run the Code

```
cd ~/picar-x/example
sudo python3 7.display.py
```

View the Image

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

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

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

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

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

Please follow the prompts to activate the corresponding functions.

- **Take Photo**

    Type q in the terminal and press Enter. The picture currently seen by the camera will be saved (if the color detection function is turned on, the mark box will also appear in the saved picture). You can see

these photos from the /home/{username}/Pictures/ directory of the Raspberry Pi. You can use tools such as *Filezilla Software* to transfer photos to your PC.

- **Color Detect**

    Entering a number between 1~6 will detect one of the colors in "red, orange, yellow, green, blue, purple". Enter 0 to turn off color detection.
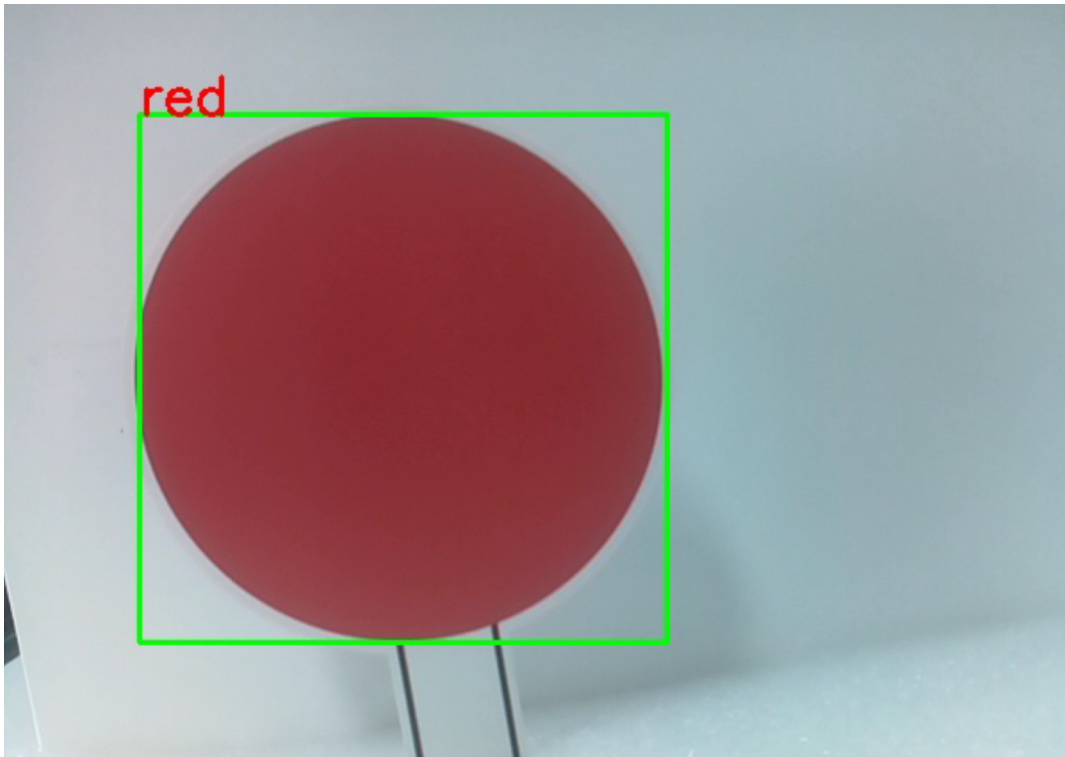
Note: You can download and print the PDF Color Cards for color detection.

- **Face Detect**

    Type f to turn on face detection.

- **QR Code Detect**

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

- **Display Information**

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

Code

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

flag_face = False
flag_color = False
qr_code_flag = False

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

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

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


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

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

```python
        sleep(0.5)
    Vilib.qrcode_detect_switch(False)


def take_photo():
    _time = strftime('%Y-%m-%d-%H-%M-%S',localtime(time()))
    name = 'photo_%s'%_time
    username = os.getlogin()

    path = f"/home/{username}/Pictures/"
    Vilib.take_photo(name, path)
    print('photo save as %s%s.jpg'%(path,name))


def object_show():
    global flag_color, flag_face

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

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


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

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

    while True:
        # readkey
        key = readchar.readkey()
        key = key.lower()
        # take photo
        if key == 'q':
```

```python
            take_photo()
        # color detect
        elif key != '' and key in ('0123456'):  # '' in ('0123') -> True
            index = int(key)
            if index == 0:
                flag_color = False
                Vilib.color_detect('close')
            else:
                flag_color = True
                Vilib.color_detect(color_list[index]) # color_detect(color:str -> color_
→name/close)
                print('Color detect : %s'%color_list[index])
        # face detection
        elif key =="f":
            flag_face = not flag_face
            face_detect(flag_face)
        # qrcode detection
        elif key =="r":
            qr_code_flag = not qr_code_flag
            if qr_code_flag == True:
                if qrcode_thread == None or not qrcode_thread.is_alive():
                    qrcode_thread = threading.Thread(target=qrcode_detect)
                    qrcode_thread.setDaemon(True)
                    qrcode_thread.start()
            else:
                if qrcode_thread != None and qrcode_thread.is_alive():
                # wait for thread to end
                    qrcode_thread.join()
                    print('QRcode Detect: close')
        # show detected object information
        elif key == "s":
            object_show()

        sleep(0.5)


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

How it works?

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

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

Functions related to "object detection":

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

- `Vilib.color_detect(color)` : For color detection, only one color detection can be performed at the same time. The parameters that can be input are: `"red"`, `"orange"`, `"yellow"`, `"green"`, `"blue"`, `"purple"`

- `Vilib.color_detect_switch(False)` : Switch OFF color detection

- `Vilib.qrcode_detect_switch(False)` : Switch ON/OFF QR code detection, Returns the decoded data of the QR code.

- `Vilib.gesture_detect_switch(False)` : Switch ON/OFF gesture detection

- `Vilib.traffic_sign_detect_switch(False)` : Switch ON/OFF traffic sign detection

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

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

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

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

- `color_x`: the x value of the center coordinate of the detected color block, the range is 0~320

- `color_y`: the y value of the center coordinate of the detected color block, the range is 0~240

- `color_w`: the width of the detected color block, the range is 0~320

- `color_h`: the height of the detected color block, the range is 0~240

- `color_n`: the number of detected color patches

- `human_x`: the x value of the center coordinate of the detected human face, the range is 0~320

- `human_y`: the y value of the center coordinate of the detected face, the range is 0~240

- `human_w`: the width of the detected human face, the range is 0~320

- `human_h`: the height of the detected face, the range is 0~240

- `human_n`: the number of detected faces

- `traffic_sign_x`: the center coordinate x value of the detected traffic sign, the range is 0~320

- `traffic_sign_y`: the center coordinate y value of the detected traffic sign, the range is 0~240

- `traffic_sign_w`: the width of the detected traffic sign, the range is 0~320

- `traffic_sign_h`: the height of the detected traffic sign, the range is 0~240

- `traffic_sign_t`: the content of the detected traffic sign, the value list is *['stop','right','left','forward']*

- `gesture_x`: The center coordinate x value of the detected gesture, the range is 0~320

- `gesture_y`: The center coordinate y value of the detected gesture, the range is 0~240

- `gesture_w`: The width of the detected gesture, the range is 0~320

- `gesture_h`: The height of the detected gesture, the range is 0~240

- `gesture_t`: The content of the detected gesture, the value list is *["paper","scissor","rock"]*

- `qr_date`: the content of the QR code being detected

- `qr_x`: the center coordinate x value of the QR code to be detected, the range is 0~320

- `qr_y`: the center coordinate y value of the QR code to be detected, the range is 0~240

- `qr_w`: the width of the QR code to be detected, the range is 0~320

- `qr_h`: the height of the QR code to be detected, the range is 0~320

4.10 8. Stare at You

This project is also based on the *7. Computer Vision* project, with the addition of face detection algorithms.

When you appear in front of the camera, it will recognize your face and adjust its gimbal to keep your face in the center of the frame.

You can view the screen at `http://<your IP>:9000/mjpg`.

Run the Code

```
cd ~/picar-x/example
sudo python3 8.stare_at_you.py
```

When the code is run, the car's camera will always be staring at your face.

Code

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

px = Picarx()

def clamp_number(num,a,b):
    return max(min(num, max(a, b)), min(a, b))

def main():
    Vilib.camera_start()
    Vilib.display()
    Vilib.face_detect_switch(True)
    x_angle =0
    y_angle =0
    while True:
        if Vilib.detect_obj_parameter['human_n']!=0:
            coordinate_x = Vilib.detect_obj_parameter['human_x']
            coordinate_y = Vilib.detect_obj_parameter['human_y']

            # change the pan-tilt angle for track the object
            x_angle +=(coordinate_x*10/640)-5
            x_angle = clamp_number(x_angle,-35,35)
            px.set_cam_pan_angle(x_angle)

            y_angle -=(coordinate_y*10/480)-5
            y_angle = clamp_number(y_angle,-35,35)
            px.set_cam_tilt_angle(y_angle)

            sleep(0.05)

        else :
            pass
            sleep(0.05)

if __name__ == "__main__":
    try:
```

```
    main()

    finally:
        px.stop()
        print("stop and exit")
        sleep(0.1)
```

How it works?

These lines of code in `while True` make the camera follow the face.

```
while True:
    if Vilib.detect_obj_parameter['human_n']!=0:
        coordinate_x = Vilib.detect_obj_parameter['human_x']
        coordinate_y = Vilib.detect_obj_parameter['human_y']

        # change the pan-tilt angle for track the object
        x_angle +=(coordinate_x*10/640)-5
        x_angle = clamp_number(x_angle,-35,35)
        px.set_cam_pan_angle(x_angle)

        y_angle -=(coordinate_y*10/480)-5
        y_angle = clamp_number(y_angle,-35,35)
        px.set_cam_tilt_angle(y_angle)
```

1. Check if there is a detected human face

   ```
   Vilib.detect_obj_parameter['human_n'] != 0
   ```

2. If a human face is detected, obtain the coordinates (`coordinate_x` and `coordinate_y`) of the detected face.

3. Calculate new pan and tilt angles (`x_angle` and `y_angle`) based on the detected face's position and adjust them to follow the face.

4. Limit the pan and tilt angles within the specified range using the `clamp_number` function.

5. Set the camera's pan and tilt angles using `px.set_cam_pan_angle()` and `px.set_cam_tilt_angle()` .

4.11 9. Record Video

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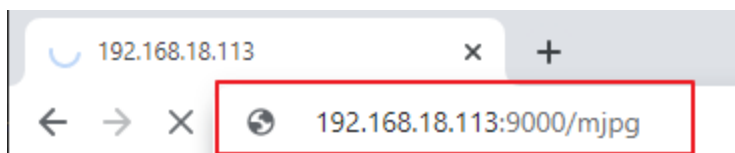
Run the Code

```
cd ~/picar-x/example
sudo python3 9.record_video.py
```

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

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

- Press q to begin recording or pause/continue, e to stop recording or save.

- If you want to exit the program, press ctrl+c.

Code

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

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

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

def main():
    rec_flag = 'stop' # start,pause,stop
    vname = None
    username = os.getlogin()

    Vilib.rec_video_set["path"] = f"/home/{username}/Videos/" # set path

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

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

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

        sleep(0.1)

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

How it works?

Functions related to recording include the following:

- `Vilib.rec_video_run(video_name)` : Started the thread to record the video. `video_name` is the name of the video file, it should be a string.

- `Vilib.rec_video_start()`: Start or continue video recording.

- `Vilib.rec_video_pause()`: Pause recording.

- `Vilib.rec_video_stop()`: Stop recording.

`Vilib.rec_video_set["path"] = f"/home/{username}/Videos/"` sets the storage location of video files.

4.12 10. Bull Fight

Make PiCar-X an angry bull! Use its camera to track and rush the red cloth!

Run the Code

```
cd ~/picar-x/example
sudo python3 10.bull_fight.py
```

View the Image

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

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

```
Use a production WSGI server instead.
* Debug mode: off
* Running on http://0.0.0.0:9000/ (Press CTRL+C to quit)
```

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

Code

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

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

px = Picarx()

def clamp_number(num,a,b):
return max(min(num, max(a, b)), min(a, b))

def main():
    Vilib.camera_start()
    Vilib.display()
    Vilib.color_detect("red")
    speed = 50
    dir_angle=0
    x_angle =0
    y_angle =0
    while True:
        if Vilib.detect_obj_parameter['color_n']!=0:
            coordinate_x = Vilib.detect_obj_parameter['color_x']
            coordinate_y = Vilib.detect_obj_parameter['color_y']

            # change the pan-tilt angle for track the object
            x_angle +=(coordinate_x*10/640)-5
            x_angle = clamp_number(x_angle,-35,35)
            px.set_cam_pan_angle(x_angle)

            y_angle -=(coordinate_y*10/480)-5
            y_angle = clamp_number(y_angle,-35,35)
            px.set_cam_tilt_angle(y_angle)

            # move
            # The movement direction will change slower than the pan/tilt direction
            # change to avoid confusion when the picture changes at high speed.
```

```python
            if dir_angle > x_angle:
                dir_angle -= 1
            elif dir_angle < x_angle:
                dir_angle += 1
            px.set_dir_servo_angle(x_angle)
            px.forward(speed)
            sleep(0.05)

        else :
            px.forward(0)
            sleep(0.05)


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

    finally:
        px.stop()
        print("stop and exit")
        sleep(0.1)
```

How it works?

You need to pay attention to the following three parts of this example:

1. Define the main function:

    - Start the camera using `Vilib.camera_start()`.

    - Display the camera feed using `Vilib.display()`.

    - Enable color detection and specify the target color as "red" using `Vilib.color_detect("red")`.

    - Initialize variables: `speed` for car movement speed, `dir_angle` for the direction angle of the car's movement, `x_angle` for the camera's pan angle, and `y_angle` for the camera's tilt angle.

2. Enter a continuous loop (while True) to track the red-colored object:

    - Check if there is a detected red-colored object (`Vilib.detect_obj_parameter['color_n'] != 0`).

    - If a red-colored object is detected, obtain its coordinates (`coordinate_x` and `coordinate_y`).

    - Calculate new pan and tilt angles (`x_angle` and `y_angle`) based on the detected object's position and adjust them to track the object.

    - Limit the pan and tilt angles within the specified range using the `clamp_number` function.

    - Set the camera's pan and tilt angles using `px.set_cam_pan_angle()` and `px.set_cam_tilt_angle()` to keep the object in view.

3. Control the car's movement based on the difference between dir_angle and `x_angle`:

    - If `dir_angle` is greater than `x_angle`, decrement `dir_angle` by 1 to gradually change the direction angle.

    - If `dir_angle` is less than `x_angle`, increment `dir_angle` by 1.

    - Set the direction servo angle using `px.set_dir_servo_angle()` to steer the car's wheels accordingly.

    - Move the car forward at the specified speed using `px.forward(speed)`.

4.13 11. Video Car

This program will provide a First Person View from the PiCar-X! Use the keyboards WSAD keys to control the direction of movement, and the O and P to adjust the speed.

Run the Code

```
cd ~/picar-x/example
sudo python3 11.video_car.py
```

Once the code is running, you can see what PiCar-X is shooting and control it by pressing the following keys.
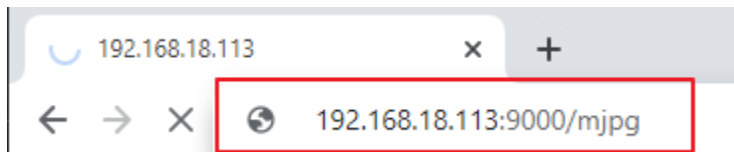
- O: speed up

- P: speed down

- W: forward

- S: backward

- A: turn left

- D: turn right

- F: stop

- T: take photo

- Ctrl+C: quit

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
#!/usr/bin/env python3

from robot_hat.utils import reset_mcu
from picarx import Picarx
from vilib import Vilib
from time import sleep, time, strftime, localtime
import readchar
```

(continues on next page)

```python
import os
user = os.getlogin()
user_home = os.path.expanduser(f'~{user}')

reset_mcu()
sleep(0.2)

manual = '''
Press key to call the function(non-case sensitive):

    O: speed up
    P: speed down
    W: forward
    S: backward
    A: turn left
    D: turn right
    F: stop
    T: take photo

    Ctrl+C: quit
'''


px = Picarx()

def take_photo():
    _time = strftime('%Y-%m-%d-%H-%M-%S',localtime(time()))
    name = 'photo_%s'%_time
    path = f"{user_home}/Pictures/picar-x/"
    Vilib.take_photo(name, path)
    print('\nphoto save as %s%s.jpg'%(path,name))


def move(operate:str, speed):

    if operate == 'stop':
        px.stop()
    else:
        if operate == 'forward':
            px.set_dir_servo_angle(0)
            px.forward(speed)
        elif operate == 'backward':
            px.set_dir_servo_angle(0)
            px.backward(speed)
        elif operate == 'turn left':
            px.set_dir_servo_angle(-30)
            px.forward(speed)
        elif operate == 'turn right':
            px.set_dir_servo_angle(30)
            px.forward(speed)
```

```python
def main():
    speed = 0
    status = 'stop'

    Vilib.camera_start(vflip=False,hflip=False)
    Vilib.display(local=True,web=True)
    sleep(2)  # wait for startup
    print(manual)

    while True:
        print("\rstatus: %s , speed: %s    "%(status, speed), end='', flush=True)
        # readkey
        key = readchar.readkey().lower()
        # operation
        if key in ('wsadfop'):
            # throttle
            if key == 'o':
                if speed <=90:
                    speed += 10
            elif key == 'p':
                if speed >=10:
                    speed -= 10
                if speed == 0:
                    status = 'stop'
            # direction
            elif key in ('wsad'):
                if speed == 0:
                    speed = 10
                if key == 'w':
                    # Speed limit when reversing,avoid instantaneous current too large
                    if status != 'forward' and speed > 60:
                        speed = 60
                    status = 'forward'
                elif key == 'a':
                    status = 'turn left'
                elif key == 's':
                    if status != 'backward' and speed > 60: # Speed limit when reversing
                        speed = 60
                    status = 'backward'
                elif key == 'd':
                    status = 'turn right'
            # stop
            elif key == 'f':
                status = 'stop'
            # move
            move(status, speed)
        # take photo
        elif key == 't':
            take_photo()
        # quit
```

```
        elif key == readchar.key.CTRL_C:
            print('\nquit ...')
            px.stop()
            Vilib.camera_close()
            break

        sleep(0.1)


if __name__ == "__main__":
    try:
        main()
    except Exception as e:
        print("error:%s"%e)
    finally:
        px.stop()
        Vilib.camera_close()
```

4.14 12. Treasure Hunt

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

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

Run the Code

```
cd ~/picar-x/example
sudo python3 12.treasure_hunt.py
```

View the Image

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

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

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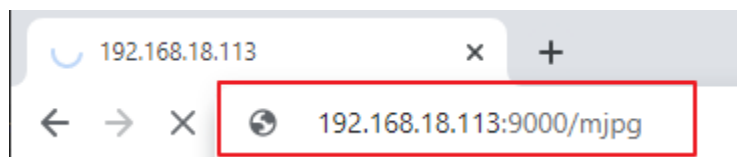
Code

```python
from picarx import Picarx
from time import sleep
from robot_hat import Music,TTS
from vilib import Vilib
import readchar
import random
import threading


px = Picarx()

music = Music()
tts = TTS()

manual = '''
Press keys on keyboard to control Picar-X!
    w: Forward
    a: Turn left
    s: Backward
    d: Turn right
    space: Say the target again
    ctrl+c: Quit
'''


color = "red"
color_list=["red","orange","yellow","green","blue","purple"]

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

key = None
lock = threading.Lock()
def key_scan_thread():
    global key
    while True:
        key_temp = readchar.readkey()
        print('\r',end='')
        with lock:
            key = key_temp.lower()
            if key == readchar.key.SPACE:
                key = 'space'
            elif key == readchar.key.CTRL_C:
                key = 'quit'
                break
        sleep(0.01)

def car_move(key):
    if 'w' == key:
        px.set_dir_servo_angle(0)
        px.forward(80)
    elif 's' == key:
```

(continues on next page)

```python
            px.set_dir_servo_angle(0)
            px.backward(80)
        elif 'a' == key:
            px.set_dir_servo_angle(-30)
            px.forward(80)
        elif 'd' == key:
            px.set_dir_servo_angle(30)
            px.forward(80)


def main():
    global key
    Vilib.camera_start(vflip=False,hflip=False)
    Vilib.display(local=False,web=True)
    sleep(0.8)
    print(manual)

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

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

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

        with lock:
            if key != None and key in ('wsad'):
                car_move(key)
                sleep(0.5)
                px.stop()
                key = None
            elif key == 'space':
                tts.say("Look for " + color)
                key = None
            elif key == 'quit':
                _key_t.join()
                print("\n\rQuit")
                break

        sleep(0.05)

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

Chapter 4. Play with Python

```python
    except KeyboardInterrupt:
        pass
    except Exception as e:
        print(f"ERROR: {e}")
    finally:
        Vilib.camera_close()
        px.stop()
        sleep(.2)
```

How it works?

To understand the basic logic of this code, you can focus on the following key parts:

1. **Initialization and Imports:** Import statements at the beginning of the code to understand the libraries being used.

2. **Global Variables:** Definitions of global variables, such as `color` and `key`, which are used throughout the code to track the target color and keyboard input.

3. `renew_color_detect()` : This function selects a random color from a list and sets it as the target color for detection. It also uses text-to-speech to announce the selected color.

4. `key_scan_thread()` : This function runs in a separate thread and continuously scans for keyboard input, updating the `key` variable with the pressed key. It uses a lock for thread-safe access.

5. `car_move(key)` : This function controls the movement of the PiCar-X based on the keyboard input (`key`). It sets the direction and speed of the robot's movement.

6. `main()` :The primary function that orchestrates the overall logic of the code. It does the following:

   - Initializes the camera and starts displaying the camera feed.

   - Creates a separate thread to scan for keyboard input.

   - Announces the start of the game using text-to-speech.

   - Enters a continuous loop to:

     - Check for detected colored objects and trigger actions when a valid object is detected.

     - Handle keyboard input to control the robot and interact with the game.

   - Handles quitting the game and exceptions like KeyboardInterrupt.

   - Ensures that the camera is closed and the PiCar-X is stopped when exiting.

By understanding these key parts of the code, you can grasp the fundamental logic of how the PiCar-X robot responds to keyboard input and detects and interacts with objects of a specific color using the camera and audio output capabilities.

4.15 13. Controlled by the APP

The SunFounder controller is used to control Raspberry Pi/Pico based robots.

The APP integrates Button, Switch, Joystick, D-pad, Slider and Throttle Slider widgets; Digital Display, Ultrasonic Radar, Grayscale Detection and Speedometer input widgets.

There are 17 areas A-Q , where you can place different widgets to customize your own controller.

In addition, this application provides a live video streaming service.

Let's customize a PiCar-X controller using this app.

How to do?

1. Install the `sunfounder-controller` module.

   The `robot-hat`, `vilib`, and `picar-x` modules need to be installed first, for details see: *Install All the Modules(Important)*.

   ```
   cd ~
   git clone https://github.com/sunfounder/sunfounder-controller.git
   cd ~/sunfounder-controller
   sudo python3 setup.py install
   ```

2. Run the code.

   ```
   cd ~/picar-x/example
   sudo python3 13.app_control.py
   ```

3. Install SunFounder Controller from **APP Store(iOS)** or **Google Play(Android)**.

4. Open and create a new controller.

   Create a new controller by clicking on the + sign in the SunFounder Controller APP.

   There are preset controllers for some products in the Preset section, which you can use as needed. Here, we select **PiCar-X**.
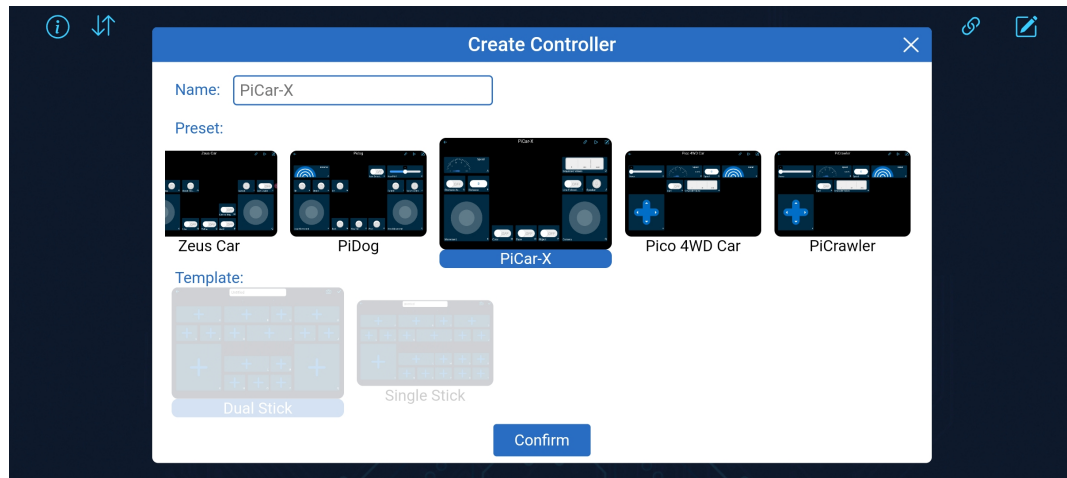
5. Connect to PiCar-x.

   When you click the **Connect** button, it will automatically search for robots nearby. Its name is defined in `picarx_control.py` and it must be running at all times.
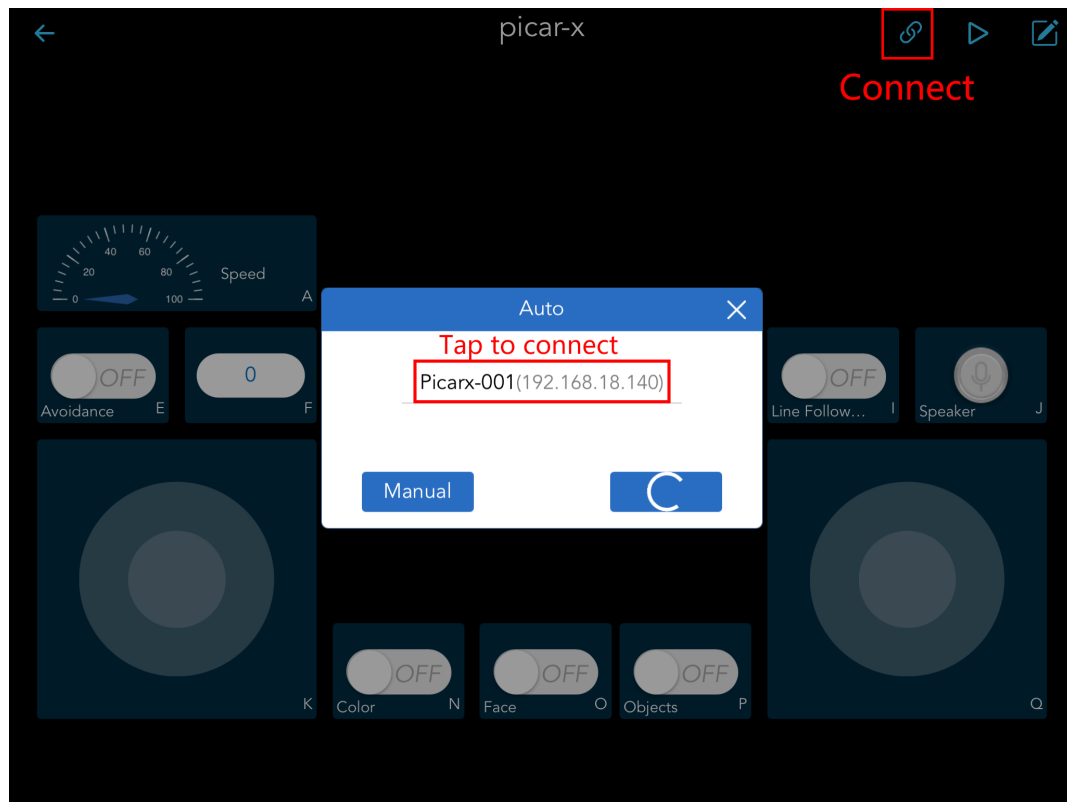
   Once you click on the product name, the message "Connected Successfully" will appear and the product name will appear in the upper right corner.
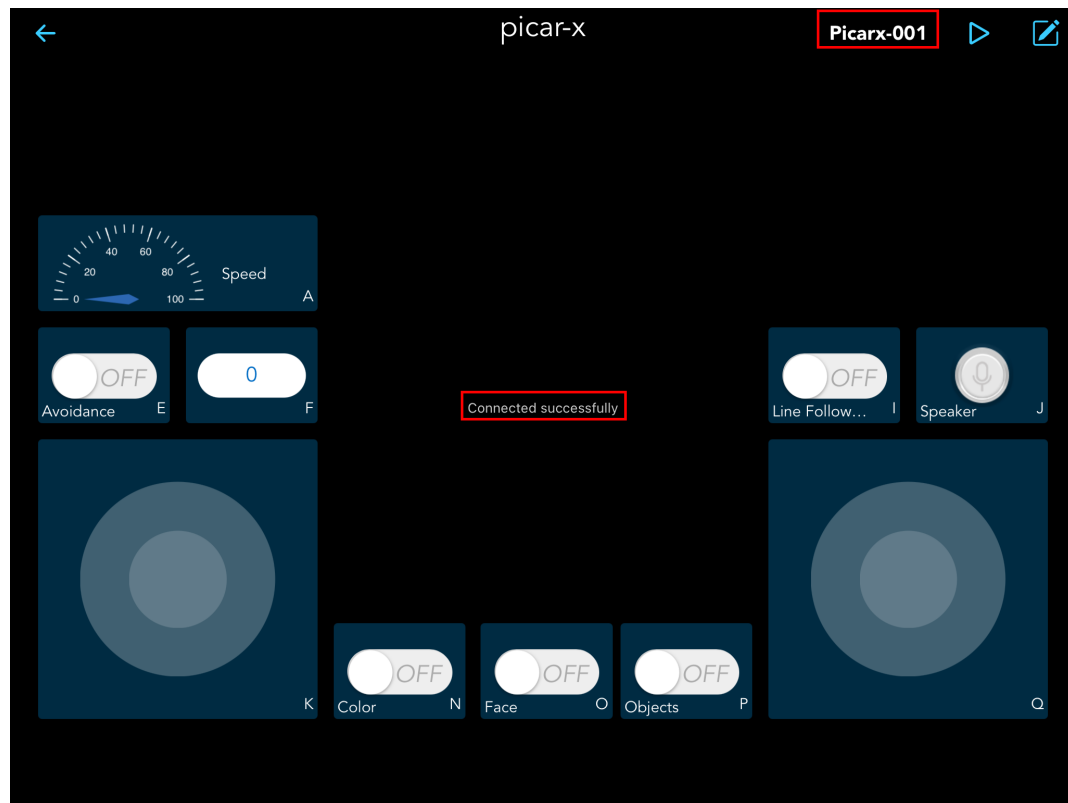
Note:

- You need to make sure that your mobile device is connected to the same LAN as PiCar-X.

- If it doesn't search automatically, you can also manually enter the IP to connect.
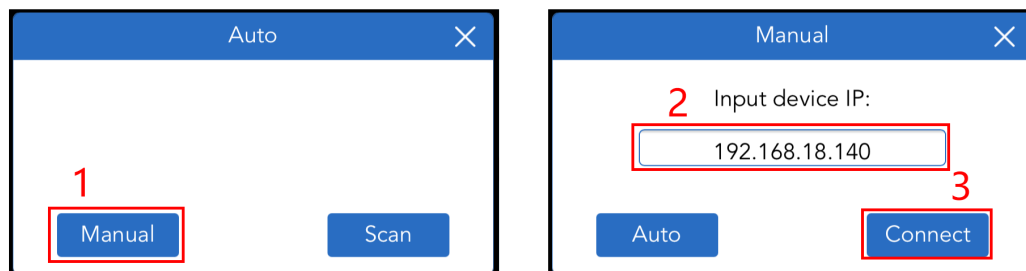
6. Run this controller.

   Click the **Run** button to start the controller, you will see the footage of the car shooting, and now you can control your PiCar-X with these widgets.
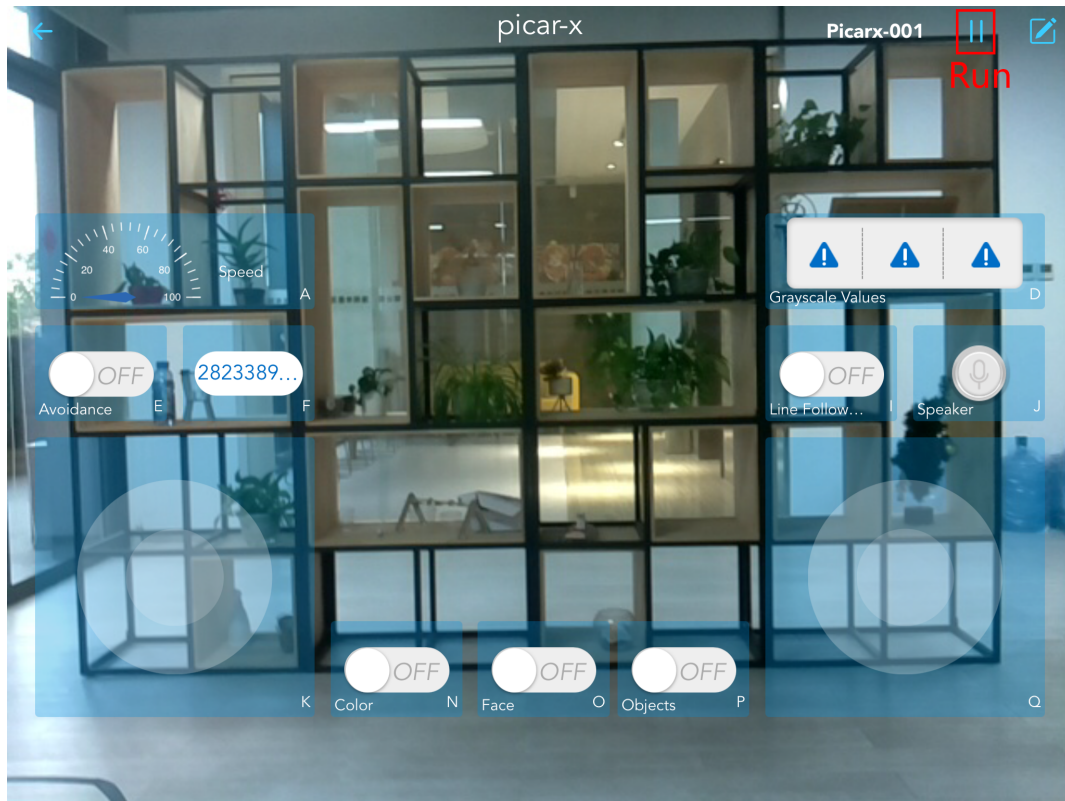
Here are the functions of the widgets.

- **A**: Show the current speed of the car.

- **E**: turn on the obstacle avoidance function.

- **I**: turn on the line following function.

- **J**: voice recognition, press and hold this widget to start speaking, and it will show the recognized voice when you release it. We have set `forward`, `backard`, `left` and `right` 4 commands in the code to control the car.

- **K**: Control forward, backward, left, and right motions of the car.

- **Q**: turn the head(Camera) up, down, left and right.

- **N**: Turn on the color recognition function.

- **O**: Turn on the face recognition function.

- **P**: Turn on the object recognition function, it can recognize nearly 90 kinds of objects, for the list of models, please refer to: https://github.com/sunfounder/vilib/blob/master/workspace/coco_labels.txt.

PLAY WITH EZBLOCK

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

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

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

5.1 Quick Guide on EzBlock

The angle range of the servo is -90~90, but the angle set at the factory is random, maybe 0°, maybe 45°; if we assemble it with such an angle directly, it will lead to a chaotic state after the robot runs the code, or worse, it will cause the servo to block and burn out.

So here we need to set all the servo angles to 0° and then install them, so that the servo angle is in the middle, no matter which direction to turn.

1. Firstly, Install EzBlock OS (EzBlock's own tutorials) onto a Micro SD card, once the installation is complete, insert it into the Raspberry Pi.

> **Note:** After the installation is complete, please return to this page.
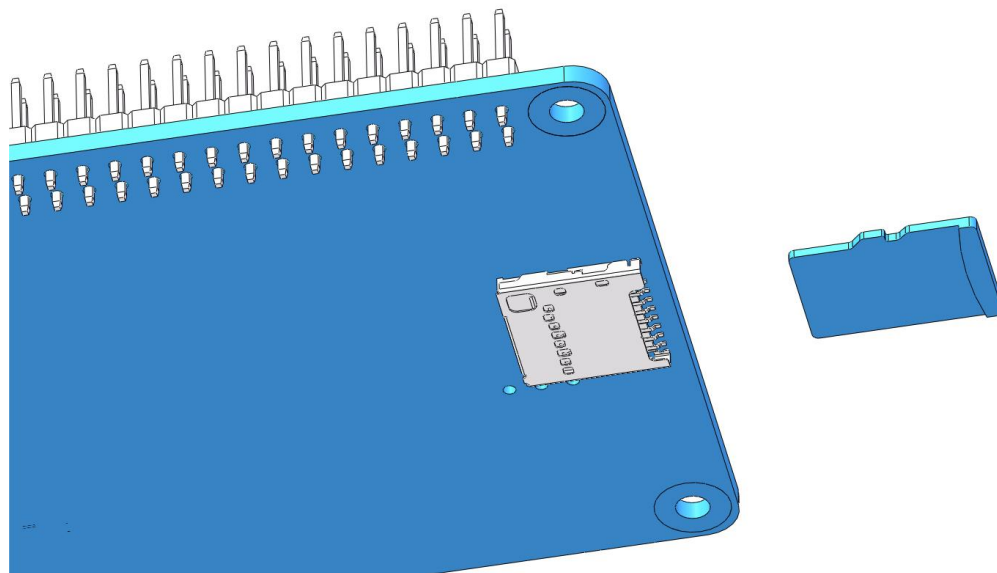
2. To ensure that the servo has been properly set to 0°, first insert the servo arm into the servo shaft and then gently rotate the rocker arm to a different angle. This servo arm is just to allow you to clearly see that the servo is rotating.
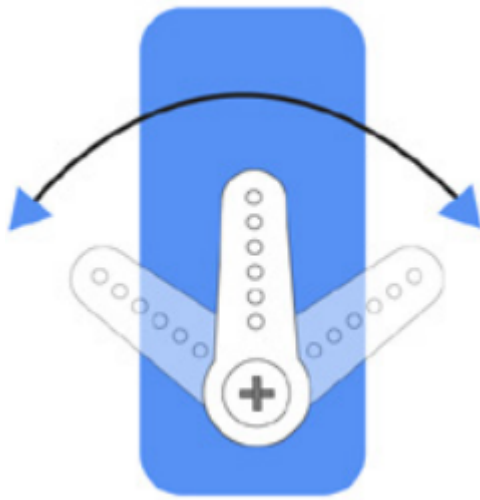
3. Follow the instructions on the assembly foldout, insert the battery holder cable and turn the power switch to the ON. Wait for 1-2 minutes, there will be a sound to indicate that the Raspberry Pi boots successfully.
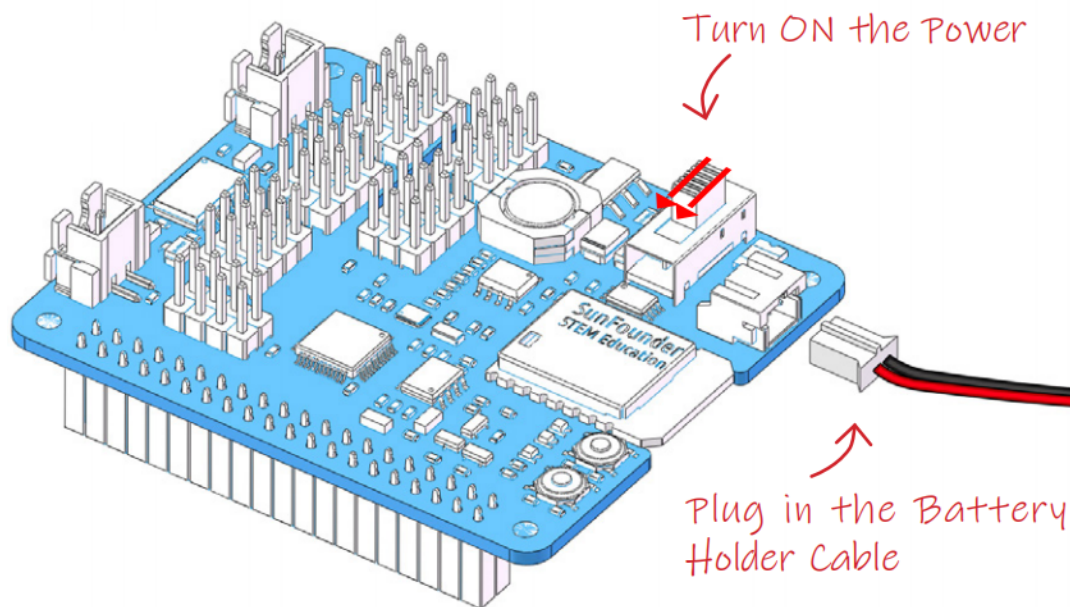
4. Next, plug the servo cable into the P11 port as follows.
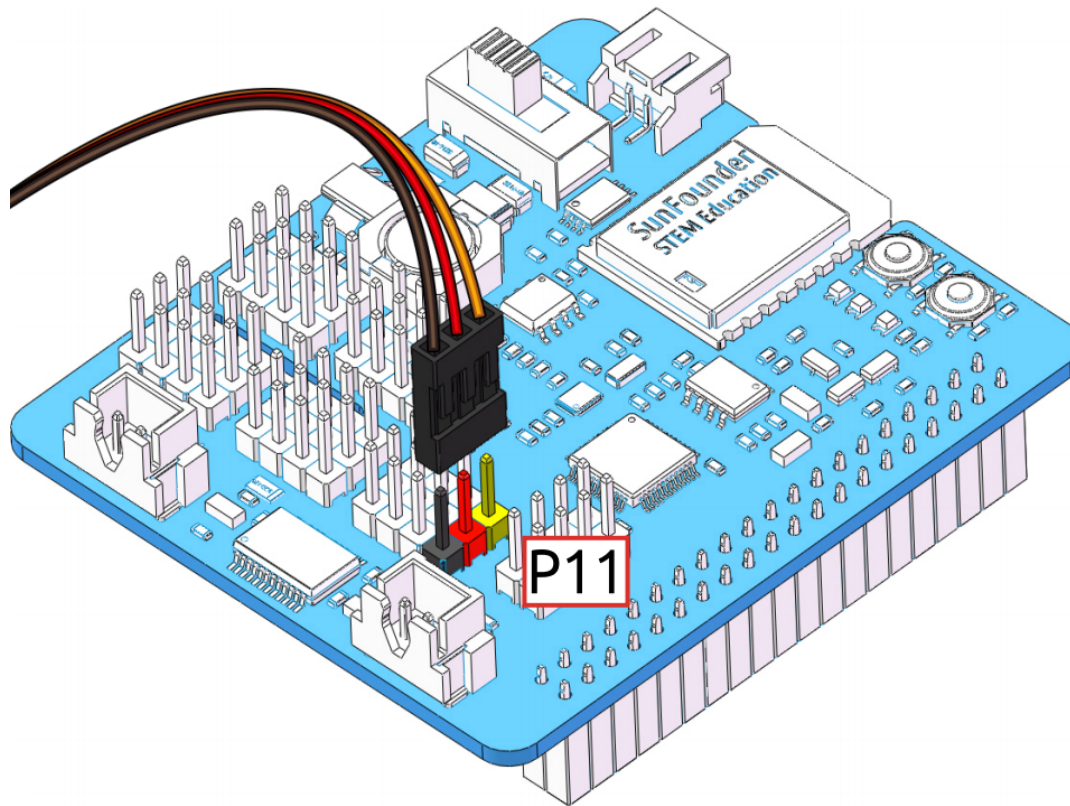
5. Press and hold the **USR** key, then press the **RST** key to execute the servo zeroing script within the system. When you see the servo arm rotate to a position(This is the 0° position, which is a random location and may not be vertical or parallel.), it indicates that the program has run.

> **Note:** This step only needs to be done once; afterward, simply insert other servo wires, and they will automatically zero.
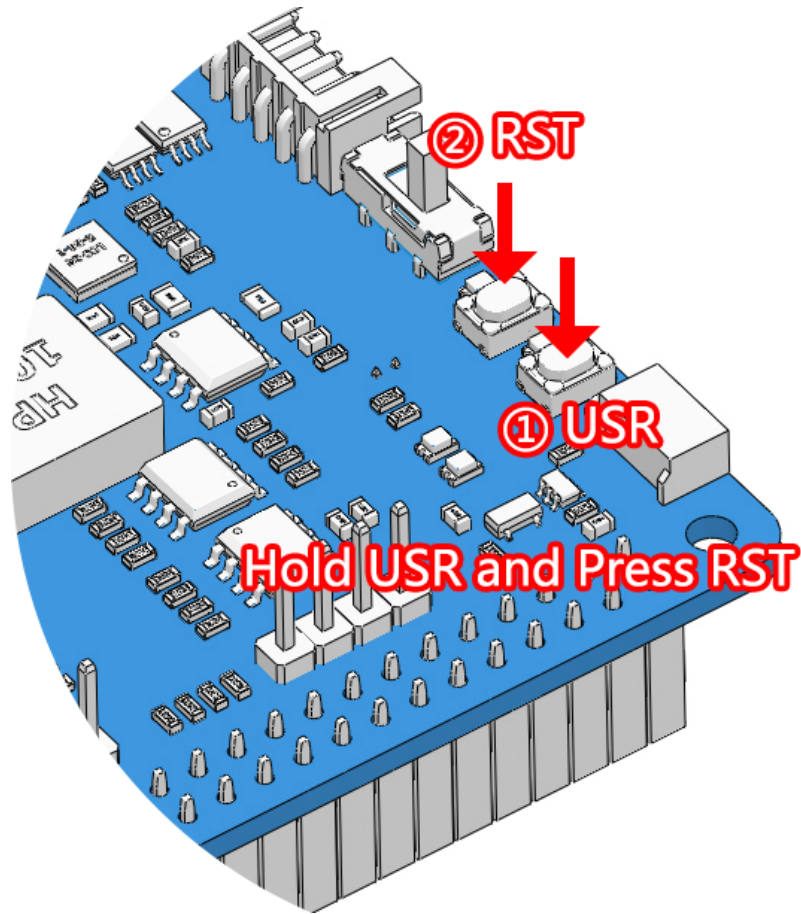
6. Now, remove the servo arm, ensuring the servo wire remains connected, and do not turn off the power. Then continue the assembly following the paper assembly instructions.

Note:

- Do not unplug this servo cable before fastening this servo with the servo screw, you can unplug it after fastening.

- Do not turn the servo while it is powered on to avoid damage; if the servo shaft is inserted at the wrong angle, pull out the servo and reinsert it.

- Before assembling each servo, you need to plug the servo cable into P11 and turn on the power to set its angle to 0°.

- This zeroing function will be disabled if you download a program to the robot later with the EzBlock APP.

5.2 Install and Configure EzBlock Studio

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

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

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

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

5.3 Calibrate the Car

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

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

Note: If you want to recalibrate the robot during use, please follow the steps below.

1. You can open the product detail page by clicking the connect icon in the upper left corner.
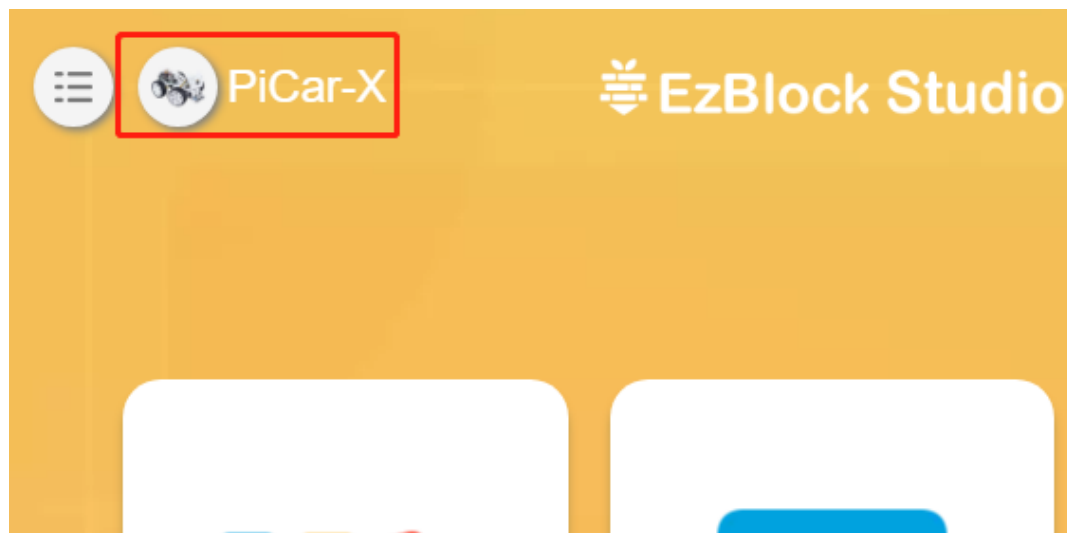
2. Click the **Settings** button.

3. On this page, you can change the product name, product type, view the app version or calibrate the robot. Once you click on **Calibrate** you can go to the calibration page.
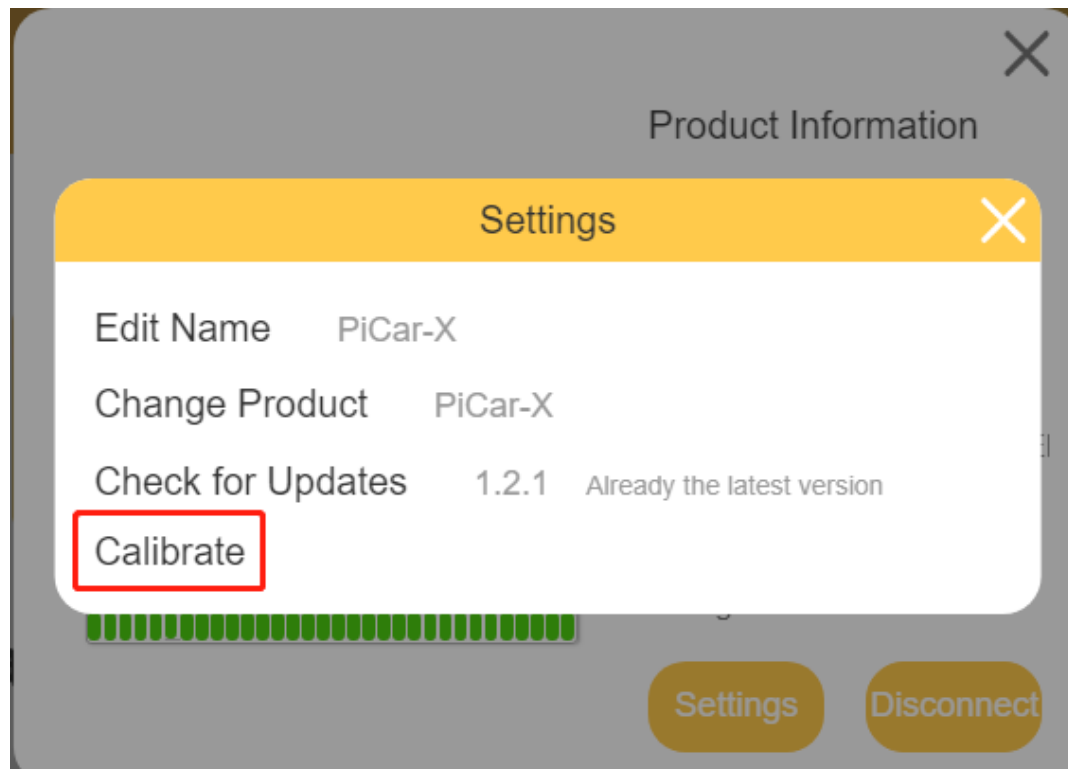
The calibration steps are as follows:

1. Once you get to the calibration page, there will be two prompt points telling you where to calibrate.

Note: Calibrating is a micro-adjustment process. It is recommended to take the part off and reassemble it if you click a button to the limit and the part is still off.

---
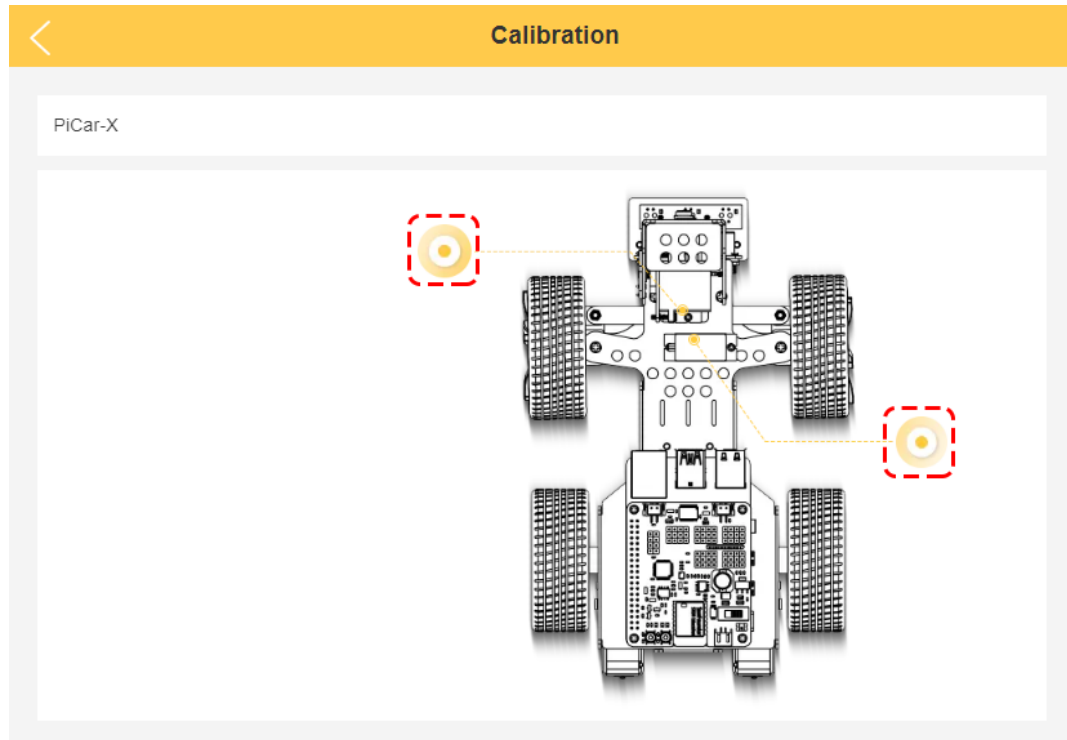
2. Click on the left prompt point to calibrate the PiCar-X's Pan-Tilt(the camera part). By using the two sets of buttons on the right, you can slowly adjust the Pan-Tilt's orientation, as well as view their angles. When the adjustment is complete, click on **Confirm**.
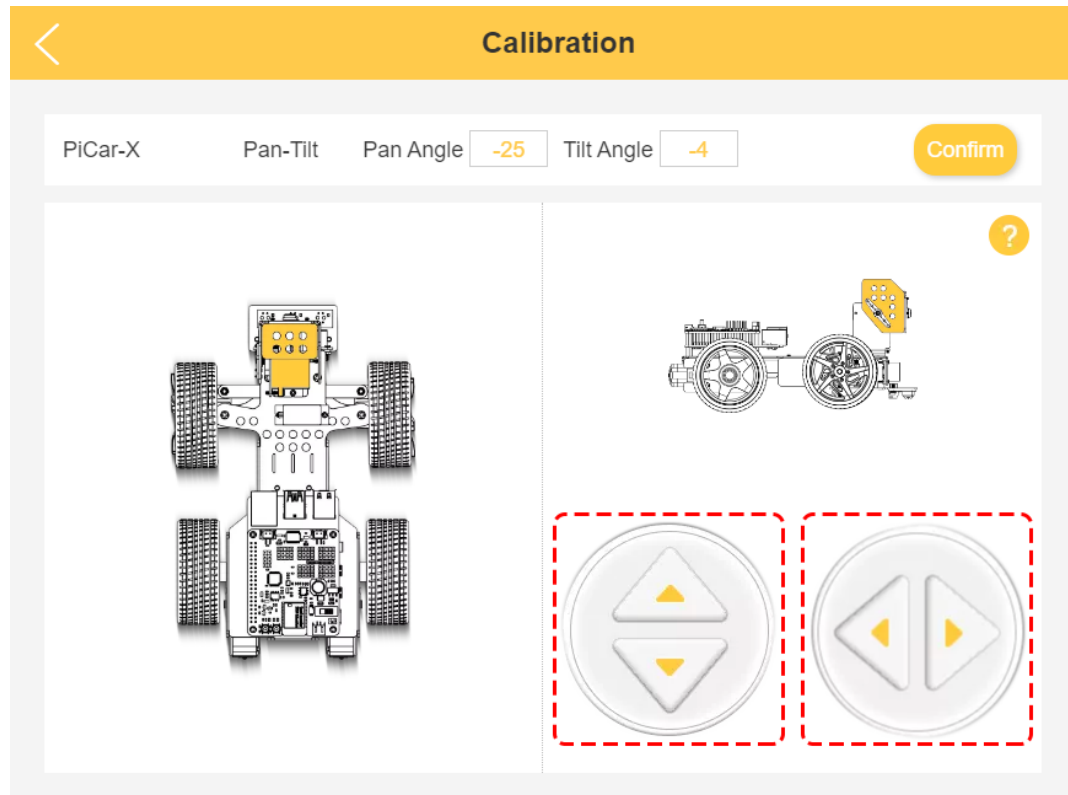
3. To calibrate the front wheel orientation, click on the right prompt point. Use the two buttons on the right to get the front wheel facing straight ahead. When the adjustment is done, click on **Confirm**.
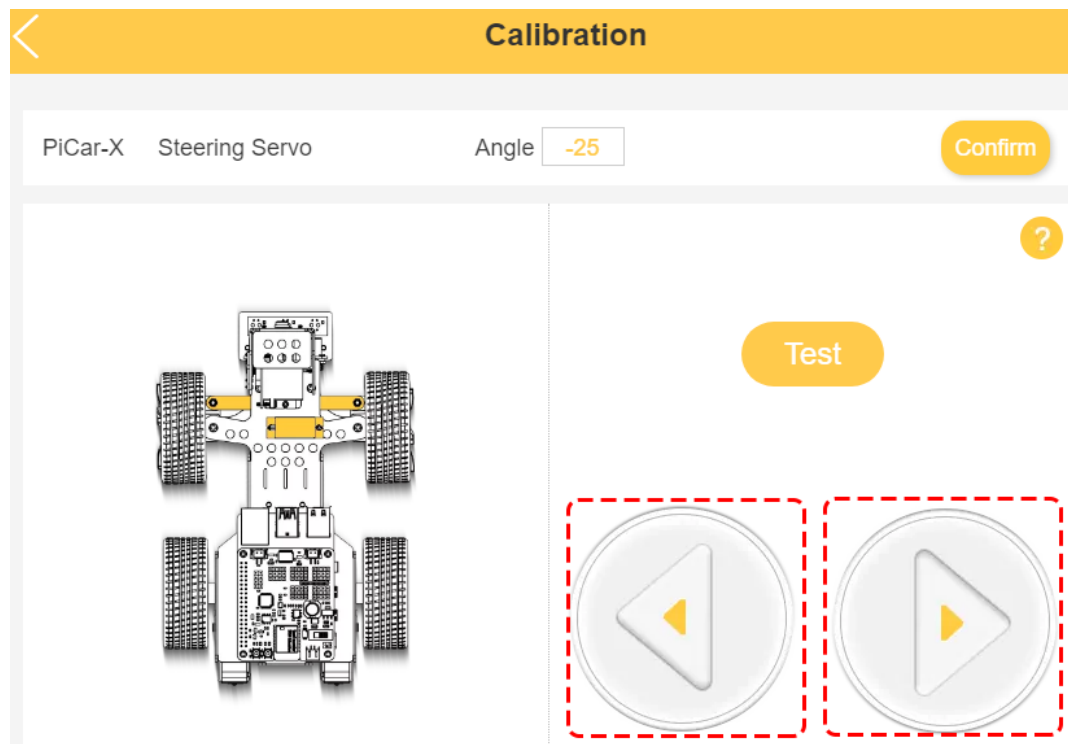
Projects

This section begins with basic programming functions for the PiCar-X, and continues through to creating more advanced programs in Ezblock Studio. Each tutorial contains TIPS that introduce new functions, allowing users to write the corresponding program. There is also a complete reference code in the Example section that can be directly used. We suggest attempting the programming without using the code in the Example sections, and enjoy the fun experience of overcoming the challenges!

All of the Ezblock projects have been uploaded to Ezblock Studio's Examples page. From the Examples page, users can run the programs directly, or edit the examples and save them into the users My Projects folder.

The Examples page allows users to choose between Block or Python language. The projects in this section only explain Block language, for an explanation of the Python code, please review this file to help you understand the Python code.
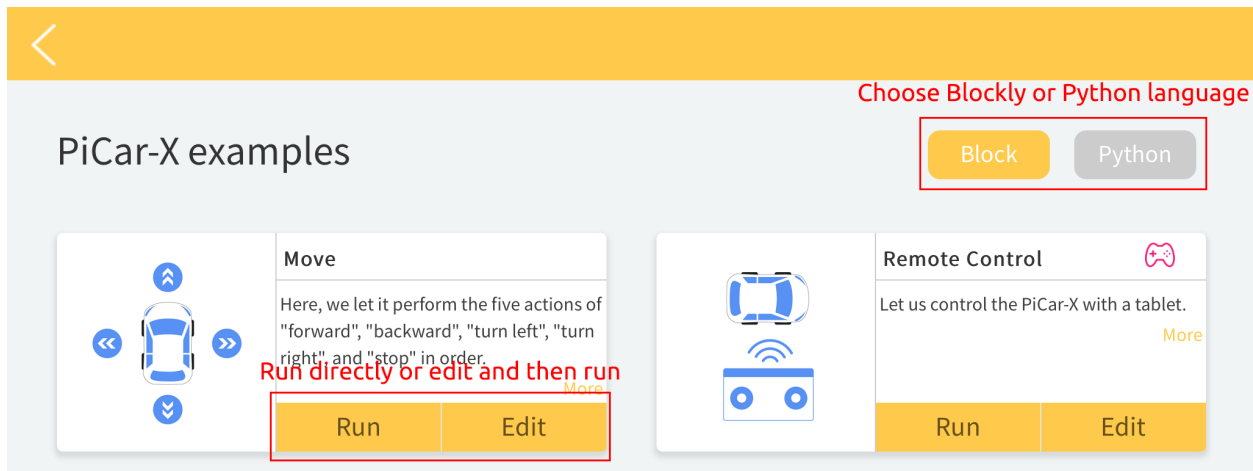
Basic

5.4 Move

This first project teaches how to program movement actions for the PiCar-X. In this project, the program will tell the PiCar-X to execute five actions in order: "forward", "backward", "turn left", "turn right", and "stop".

To learn the basic usage of Ezblock Studio, please read through the following two sections:

- How to Create a New Project?
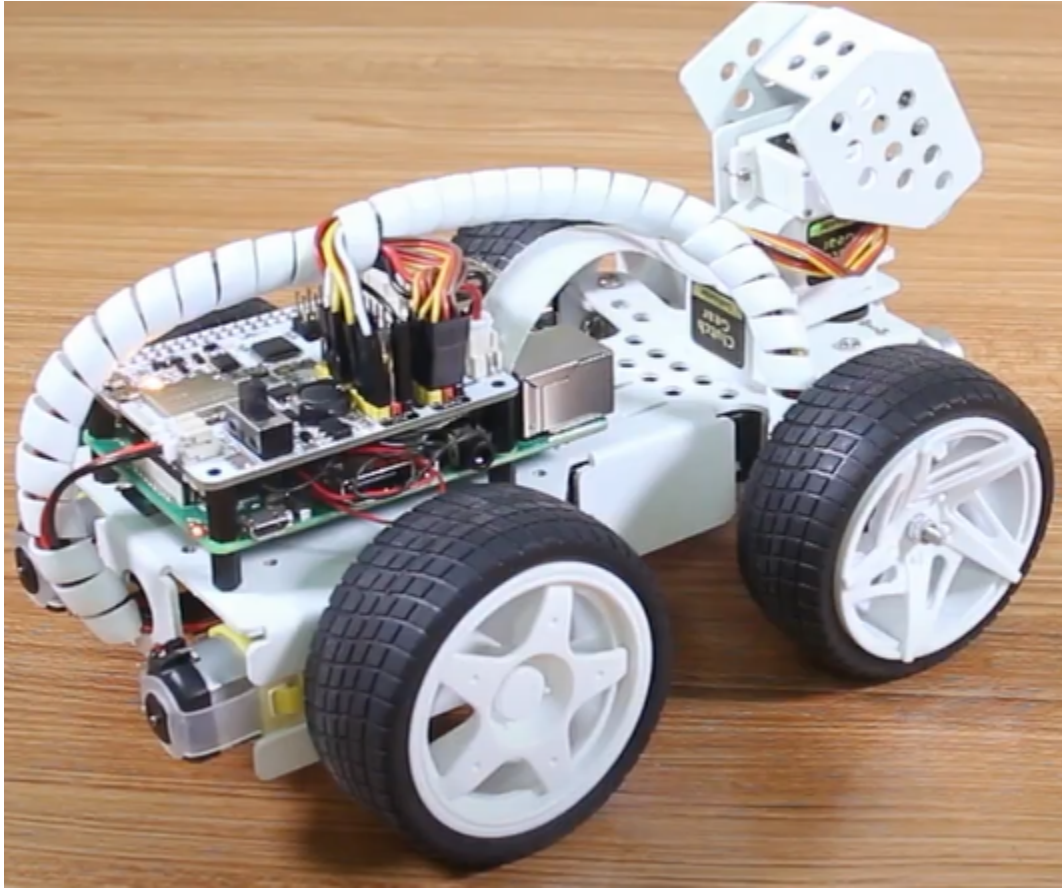
TIPS

This block will make the PiCar-X move forward at a speed based on a percentage of available power. In the example below "50" is 50% of power, or half-speed.

This block will make the PiCar-X move backward at a speed based on a percentage of available power.

This block adjusts the orientation of the front wheels. The range is "-45" to "45". In the example below, "-30" means the wheels will turn 30° to the left.

This block will cause a timed break between commands, based on milliseconds. In the example below, the PiCar-X will wait for 1 second (1000 milliseconds) before executing the next command.

This block will bring the PiCar-X to a complete stop.

EXAMPLE

Note:

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

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

Start

Forever
> forward at `50` % speed
> delay `1000`
> backward at `50` % speed
> delay `1000`
> forward at `50` % speed
> turn steering angle to `-30`
> delay `1000`
> forward at `50` % speed
> turn steering angle to `30`
> delay `1000`
> turn steering angle to `0`
> stop
> delay `2000`
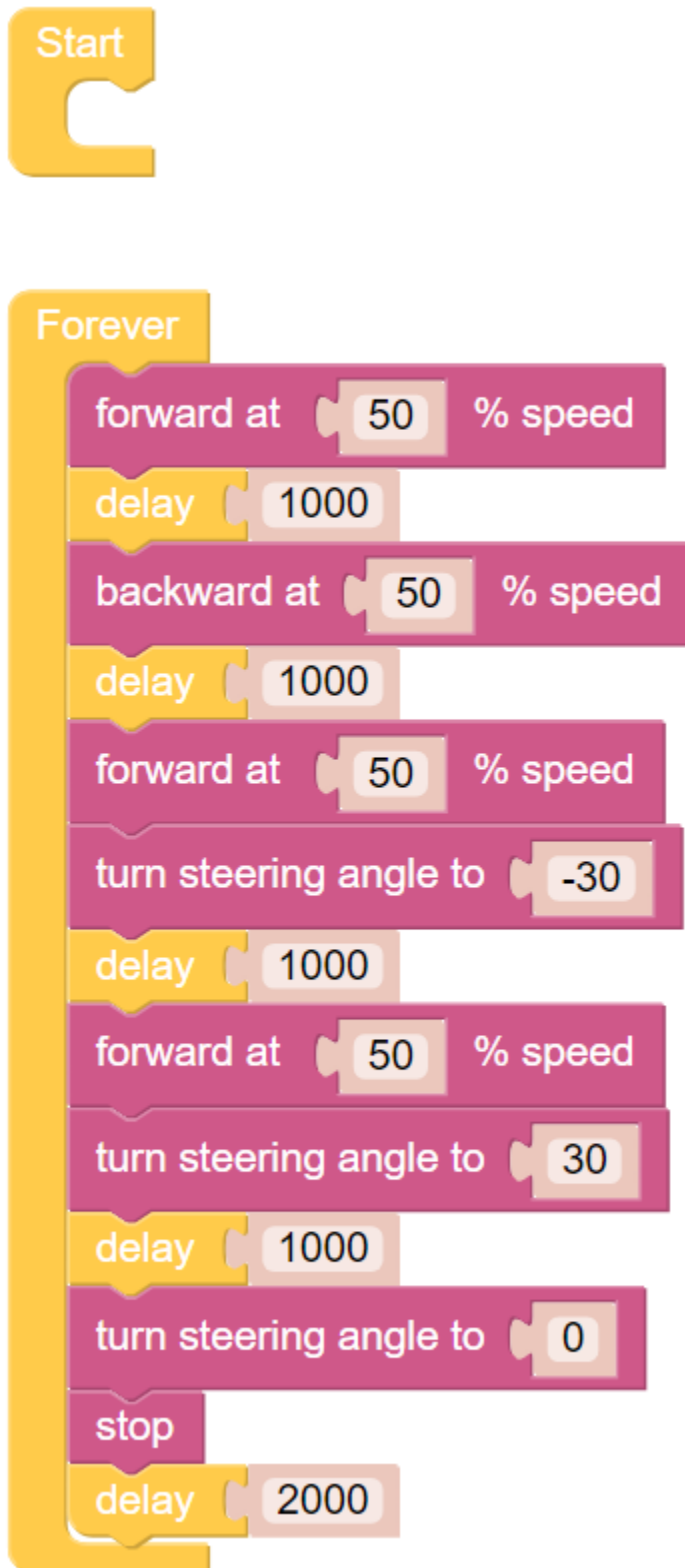
5.5 Remote Control

This project will teach how to remotely control the PiCar-X with the Joystick widget. Note: After dragging and dropping the Joystick widget from the Remote Control page, use the "Map" function to calibrate the Joysticks X-axis and Y-axis readings. For more information on the Remote Control function, please reference the following link:

- How to Use the Remote Control Function?
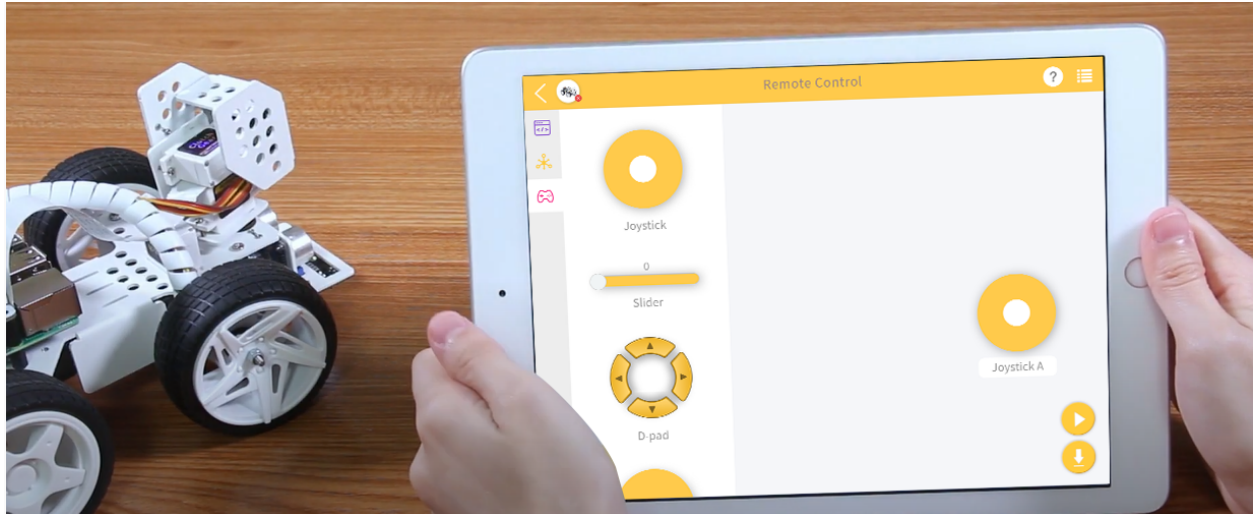
TIPS

To use the remote control function, open the Remote Control page from the left side of the main page.
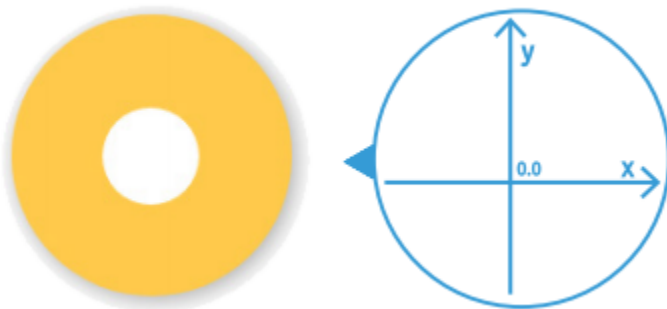
Drag a Joystick to the central area of the Remote Control page. Toggling the white point in the center, and gently dragging in any direction will produce an (X,Y) coordinate. The range of the X-axis or Y-axis is defaulted to "-100"

to "100". Toggling the white point and dragging it directly to the far left of the Joystick will result in an X value of "-100" and a Y value of "0".

After dragging and dropping a widget on the remote control page, a new category-Remote with the above block will appear. This block reads the Joystick value in the Remote Control page. You can click the drop-down menu to switch to the Y-axis reading.
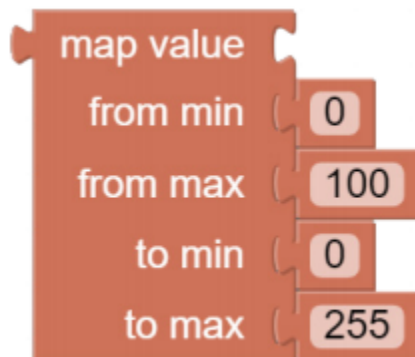
The map value block can remap a number from one range to another. If the range is set to 0 to 100, and the map value number is 50, then it is at a 50% position of the range, or "50". If the range is set to 0 to 255 and the map value number is 50, then it is at a 50% position of the range, or "127.5".
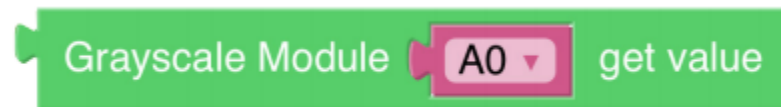
EXAMPLE

Note:

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

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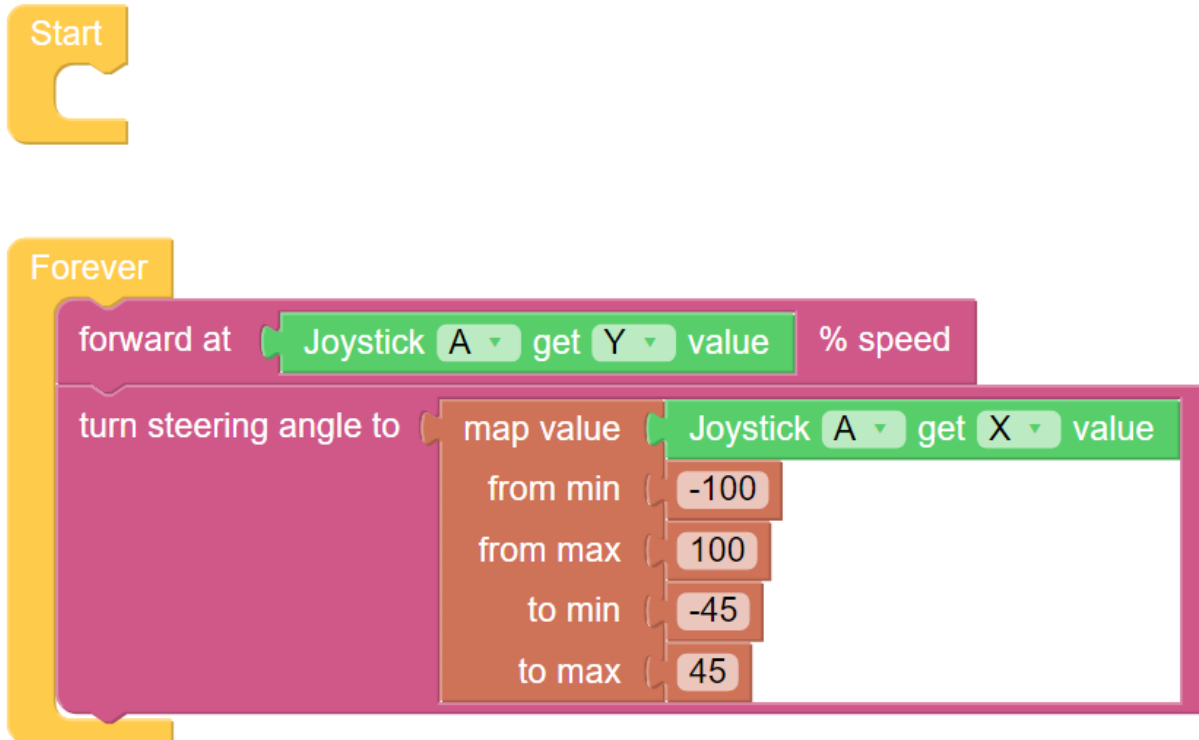
5.6 Test Ultrasonic Module

PiCar-X has a built-in Ultrasonic Sensor module that can be used for obstacle avoidance and automatic object-following experiments. In this lesson the module will read a distance in centimeters (24 cm = 1 inch), and **Print** the results in a **Debug** window.

TIPS
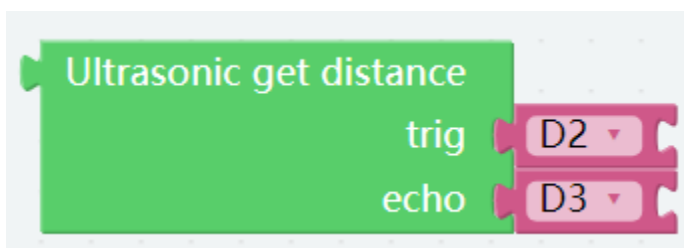
The **Ultrasonic get distance** block will read the distance from the PiCar-X to an obstacle directly ahead.
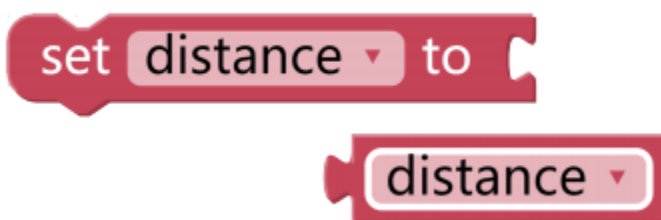
This program is simplified with a **Variable**. For example, when there are multiple functions in a program that each need to use the distance to an obstacle, a **Variable** can be used to report the same distance value to each function, instead of each function reading the same value separately.

Click the **Create variable…** button on the **Variables** category, and use the drop-down arrow to select the variable named "distance".

The **Print** function can print data such as variables and text for easy debugging.
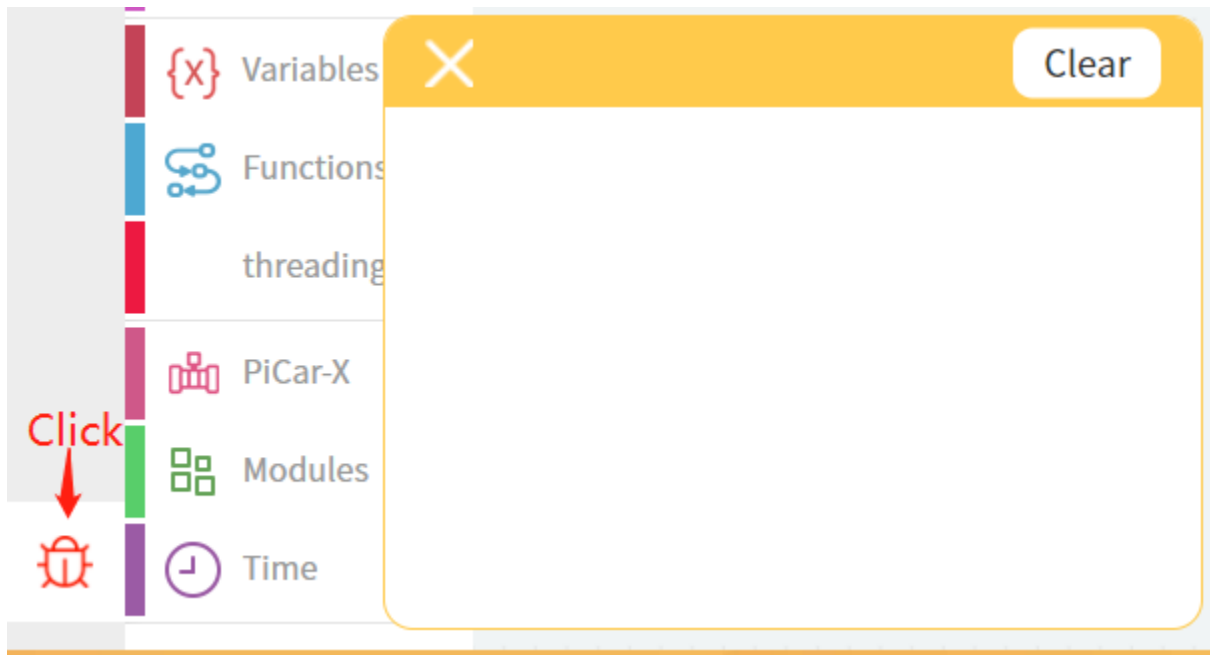
Once the code is running, enable the debug monitor by clicking the **Debug** icon in the bottom left corner.

EXAMPLE

Note:

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

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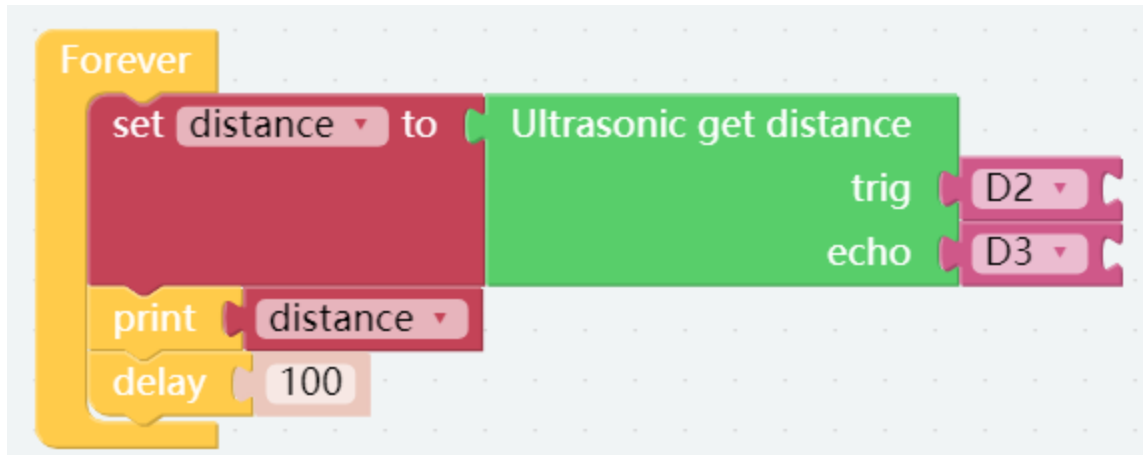
5.7 Test Grayscale Module

PiCar-X includes a Grayscale module for implementing line-following, cliff detection, and other fun experiments. The Grayscale module has three detection sensors that will each report a value according to the shade of color detected by the sensor. For example, a sensor reading the shade of pure black will return a value of "0".

TIPS

Use the **Grayscale module** block to read the value of one of the sensors. In the example above, the "A0" sensor is the sensor on the far left of the PiCar-X. Use the drop-down arrow to change the sensor to "A1" (center sensor), or "A2" (far right sensor).
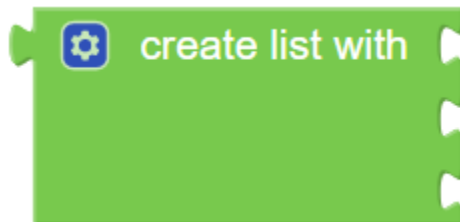
The program is simplified with a **create list with** block. A **List** is used in the same way as a single **Variable**, but in this case a **List** is more efficient than a single **Variable** because the **Grayscale module** will be reporting more than one sensor value. The **create list with** block will create separate **Variables** for each sensor, and put them into a List.

EXAMPLE

Note:

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

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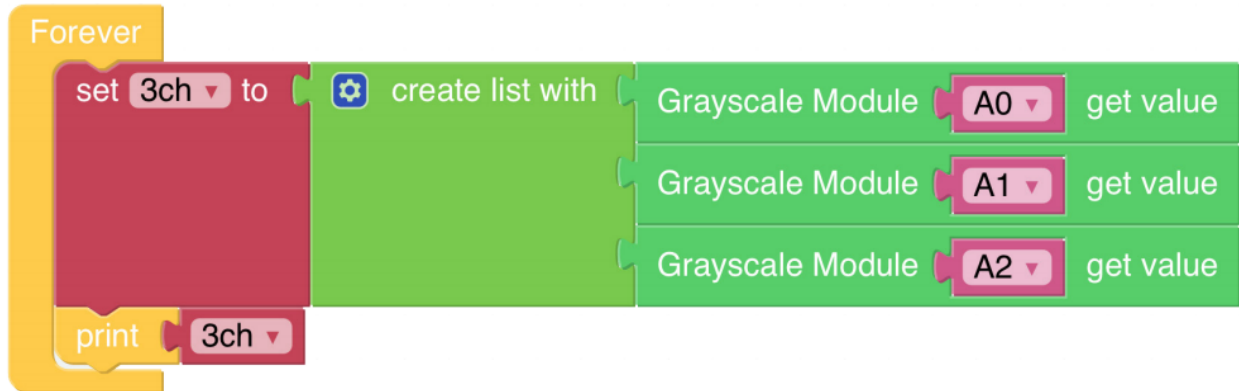
5.8 Color Detection

PiCar-X is a self-driving car with a built-in camera, which allows Ezblock programs to utilize object detection and color recognition code. In this section, Ezblock will be used to create a program for color detection.

Note: Before attempting this section, make sure that the Raspberry Pi Camera's FFC cable is properly and securely connected. For detailed instructions on securely connecting the FCC cable, please reference: *Component List and Assembly Instructions*.


In this program, Ezblock will first be told the Hue-Saturation-Value (HSV) space range of the color to be detected, then utilize OpenCV to process the colors in the HSV range to remove the background noise, and finally, box the matching color.

Ezblock includes 6 color models for PiCar-X, "red", "orange", "yellow", "green", "blue", and "purple". Color cards have been prepared in the following PDF, and will need to be printed on a color printer.

- [PDF]Color Cards

Note: The printed colors may have a slightly different hue from the Ezblock color models due to printer toner differences, or the printed medium, such as a tan-colored paper. This can cause a less accurate color recognition.
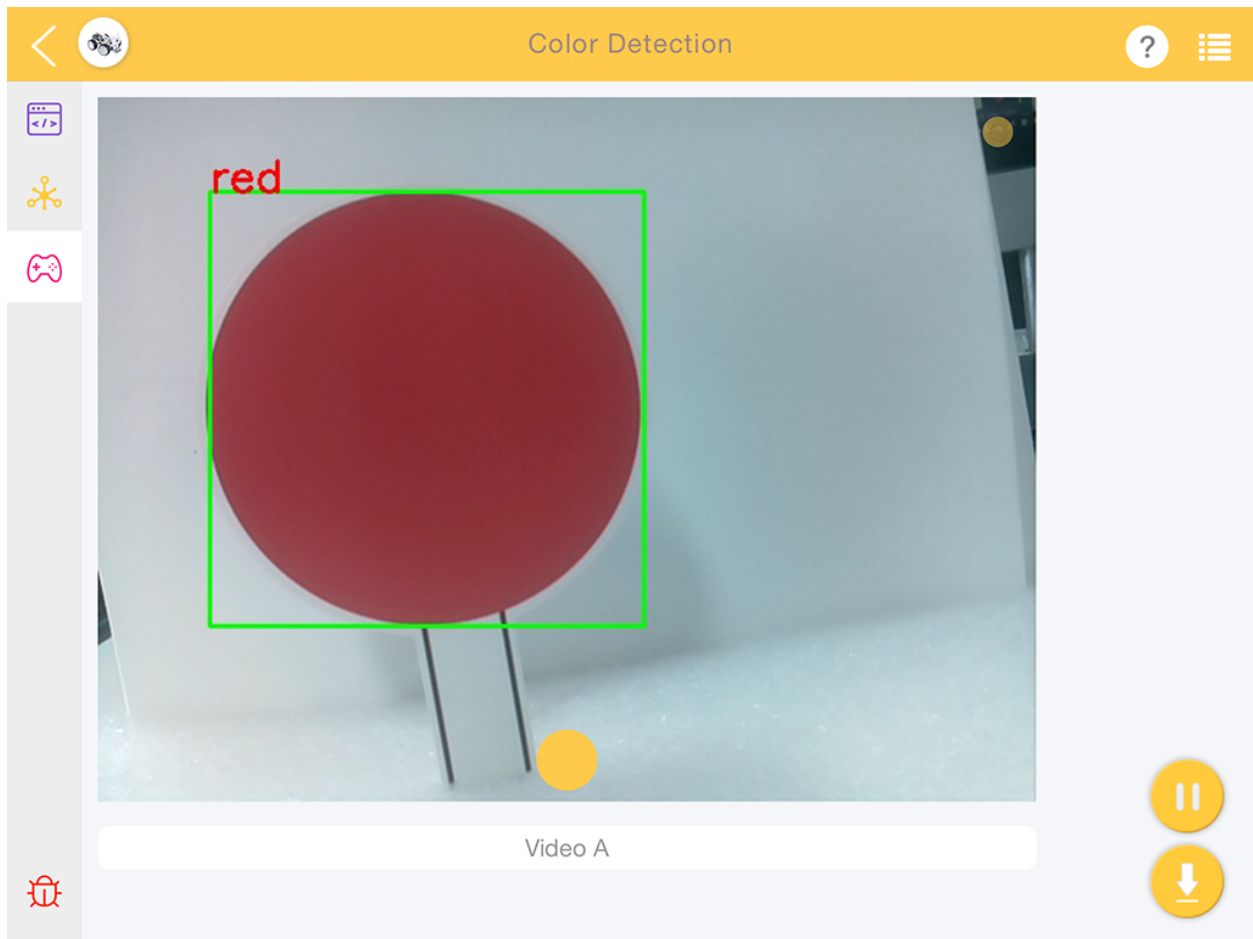
TIPS

Drag the Video widget from the remote Control page, and it will generate a video monitor. For more information on how to use the Video widget, please reference the tutorial on Ezblock video here: How to Use the Video Function?.
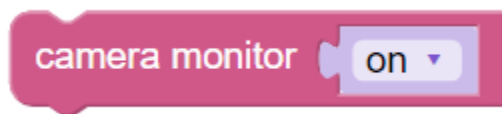

Enable the video monitor by setting the **camera monitor** block to **on**. Note: Setting the **camera monitor** to **off** will close the monitor, but object detection will still be available.

Use the **color detection** block to enable the color detection. Note: only one color can be detected at a time.

EXAMPLE

Note:

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

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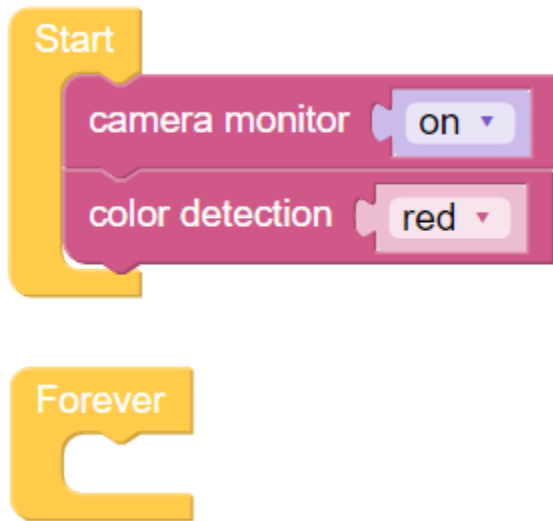
5.9 Face Detection

In addition to color detection, PiCar-X also includes a face detection function. In the following example the Joystick widget is used to adjust the direction of the camera, and the number of faces will be displayed in the debug monitor.

For more information on how to use the Video widget, please reference the tutorial on Ezblock video here: How to Use the Video Function?.
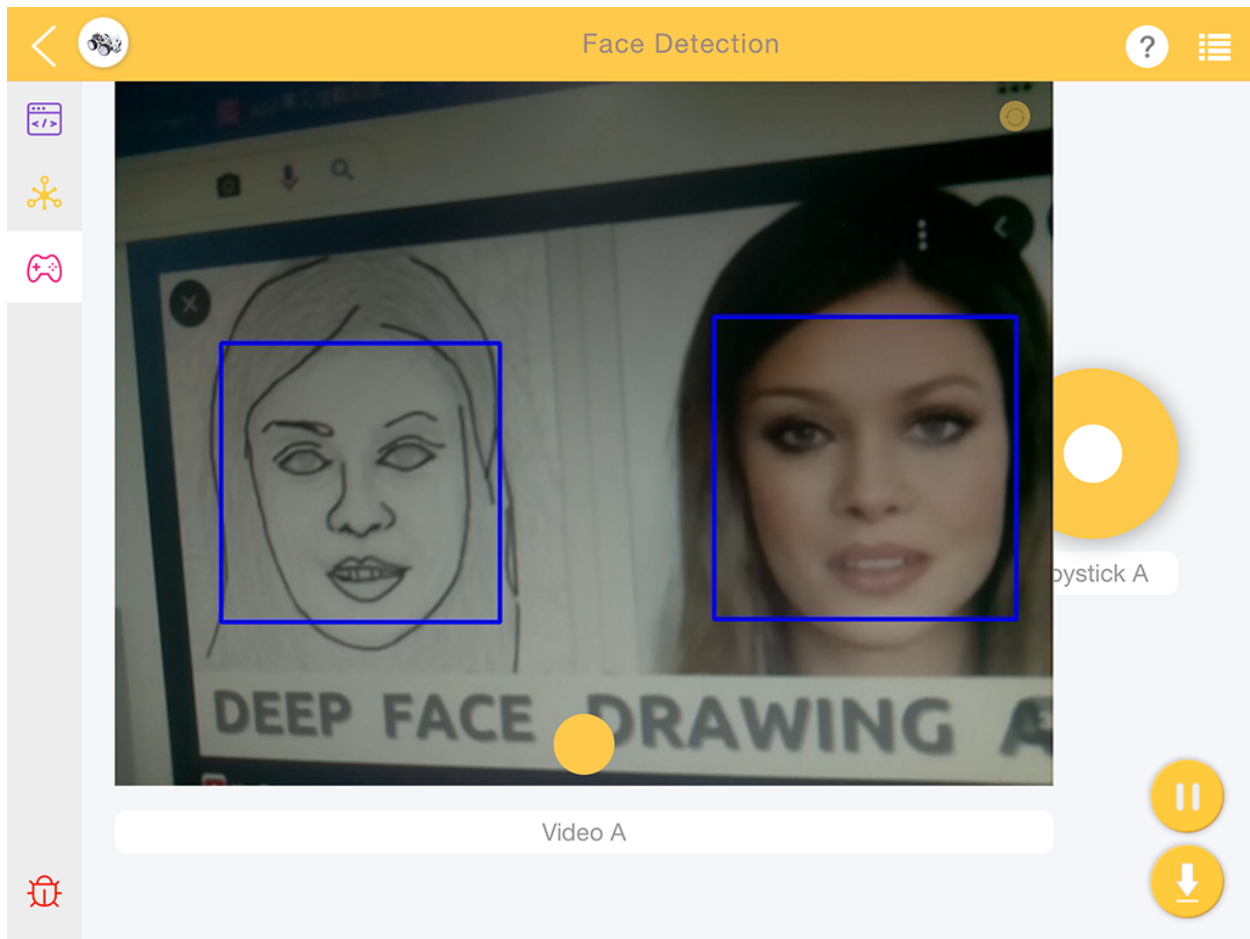
TIPS

Set the **face detection** widget to **on** to enable facial detection.
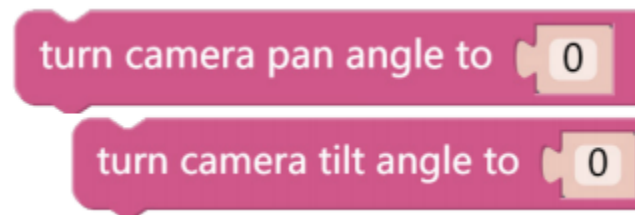
These two blocks are used to adjust the orientation of the pan-tilt camera, similar to driving the PiCar-X in the *Remote Control* tutorial. As the value increases, the camera will rotate to the right, or upwards, a decreasing value will rotate the camera right, or downwards.
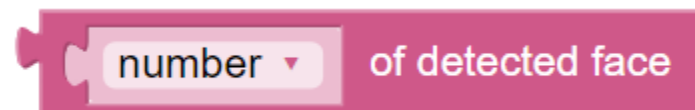
The image detection results are given through the of **detected face** block. Use the drop-down menu options to choose between reading the coordinates, size, or number of results from the image detection function.

Use the **create text with** block to print the combination of **text** and of **detected face** data.

EXAMPLE

Note:

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

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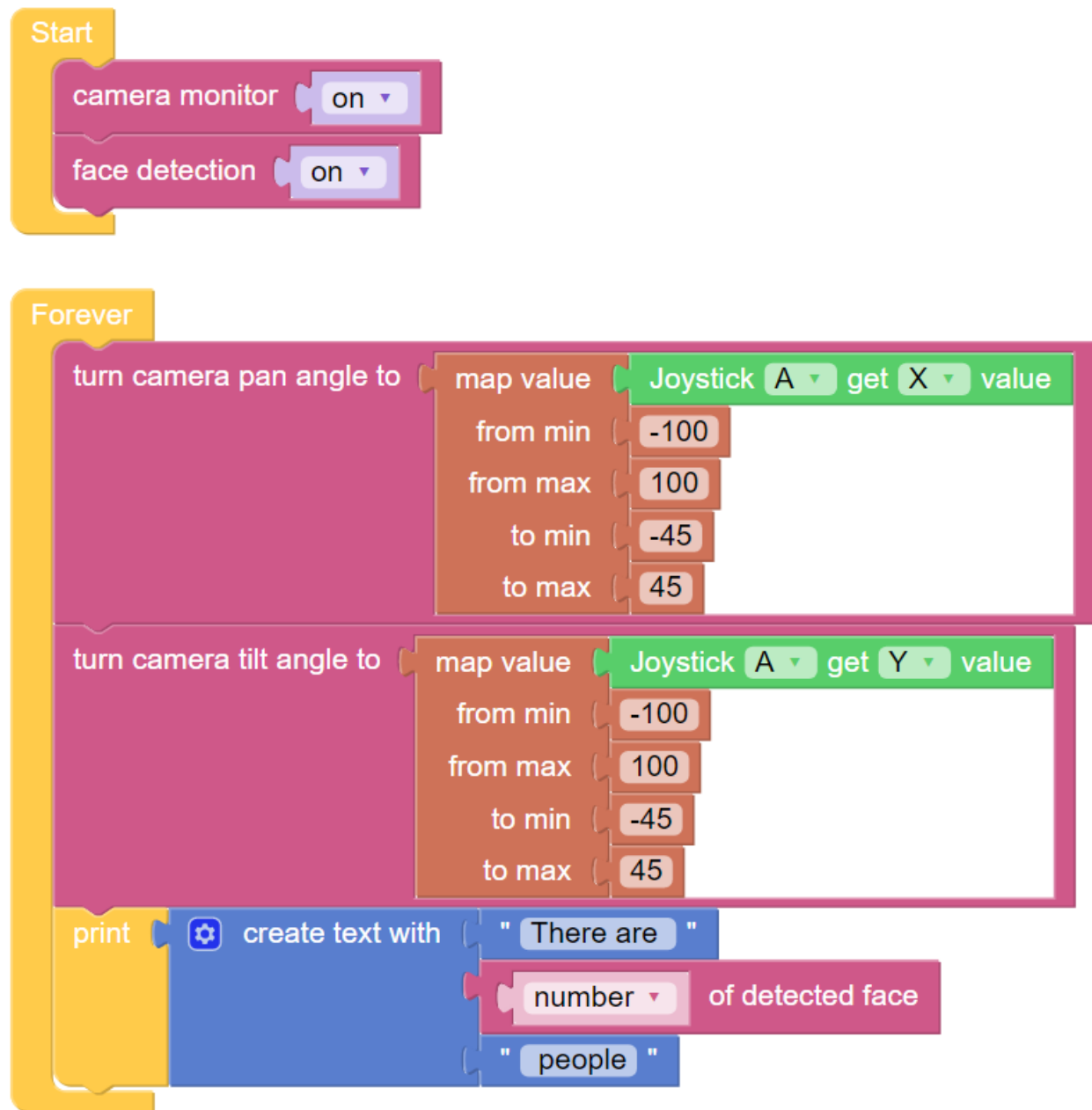
5.10 Sound Effect

PiCar-X has a built-in speaker that can be used for audio experiments. Ezblock allows users to enter text to make the PiCar-X speak, or make specific sound effects. In this tutorial, the PiCar-X will make the sound of a gun firing after a 3-second countdown, using a do/while function.

TIPS

Use the **say** block with a **text** block to write a sentence for the PiCar-X to say. The **say** block can be used with text or numbers.

The **number** block.
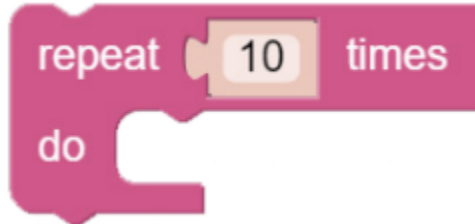
Using the **repeat** block will repeatedly execute the same statement, which reduces the size of the code.
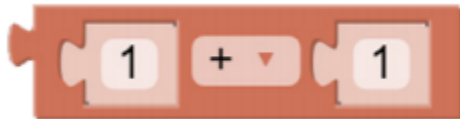
The **mathematical operation** block can perform typical mathematical functions, such as "+", "-", "x", and "÷ ".
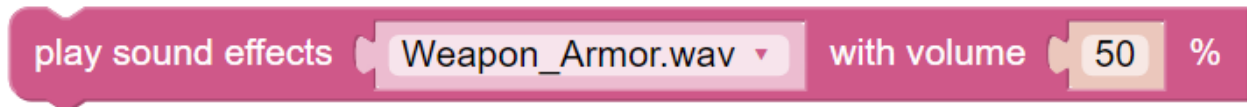
The play **sound effects - with volume - %** block has preset sound effects, such as a siren sound, a gun sound, and others. The range of the volume can be set from 0 to 100.

EXAMPLE

Note:

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

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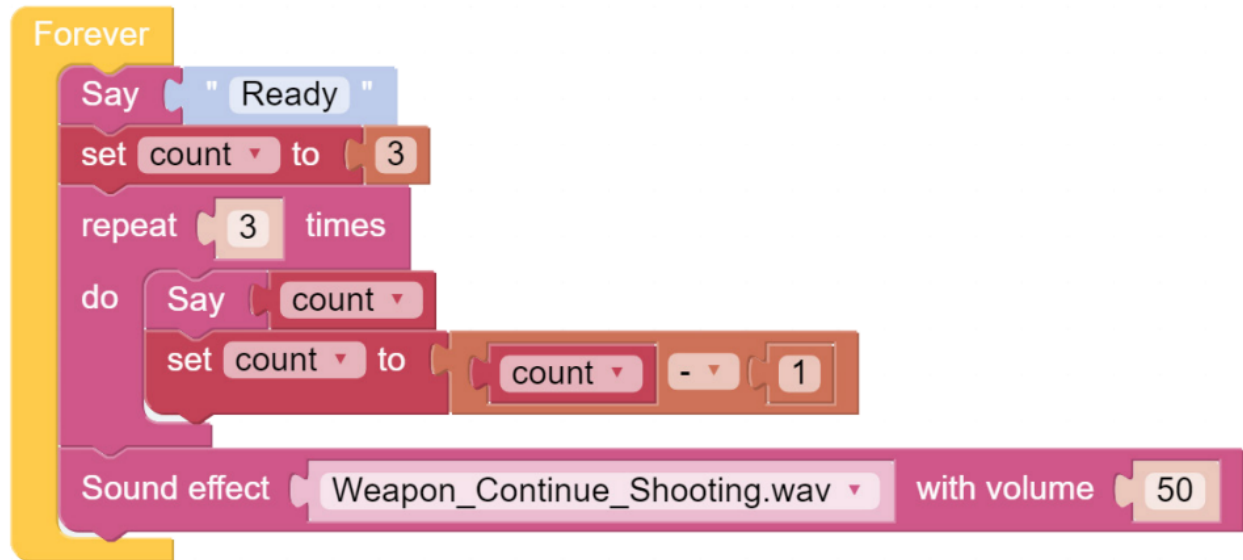
5.11 Background Music

In addition to programming the PiCar-X to play sound effects or text-to-speech (TTS), the PiCar-X will also play background music. This project will also use a **Slider** widget for adjusting the music volume.
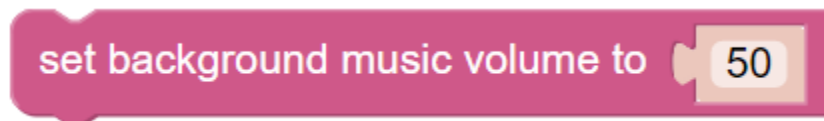
- How to Use the Remote Control Function?

For a detailed tutorial on Ezblocks remote control functions, please reference the *Remote Control* tutorial.

TIPS
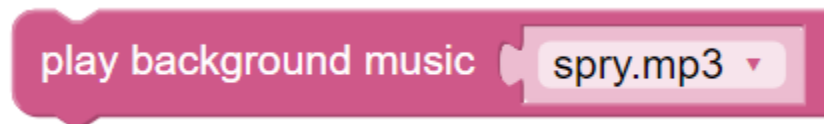
The **play background music** block will need to be added to the **Start** function. Use the drop-down menu to choose different background music for the PiCar-X to play.

The block **set background music volume to** will adjust the volume between the range of 0 to 100.
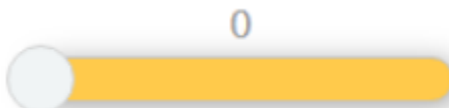
Drag a **Slider** bar from the **Remote Control** page to adjust music volume.

The **slider [A] get value** block will read the slider value. The example above has slider 'A' selected. If there are multiple sliders, use the drop-down menu to select the appropriate one.

EXAMPLE

Note:

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

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

5.12 Say Hello

This project will combine several functions from the preceding projects. The PiCar-X movement will be remotely controlled, and the PiCar's camera will be remotely controlled by using two joystick controllers. When PiCar recognizes someone's face, it will nod politely and then say "Hello!".

- How to Use the Video Function?
- How to Use the Remote Control Function?
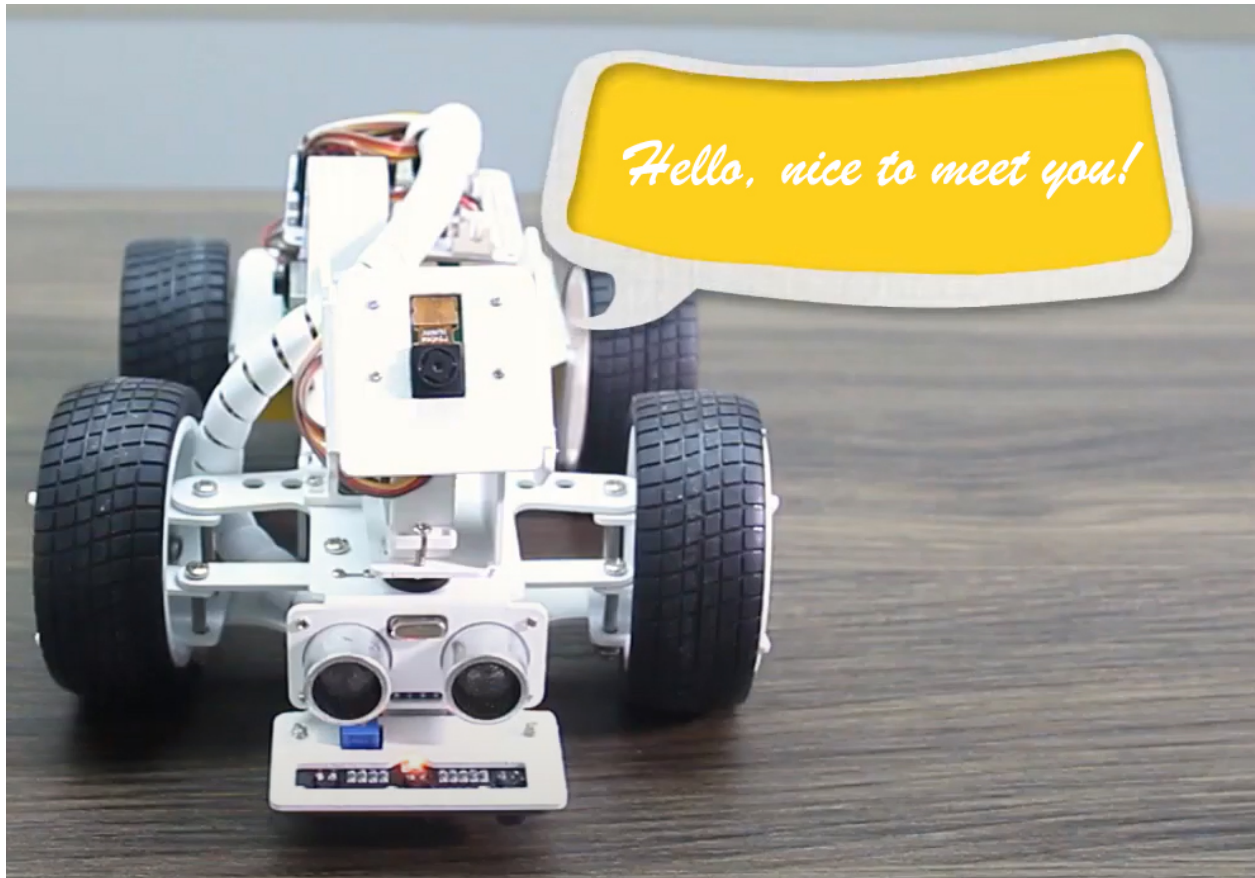
TIPS

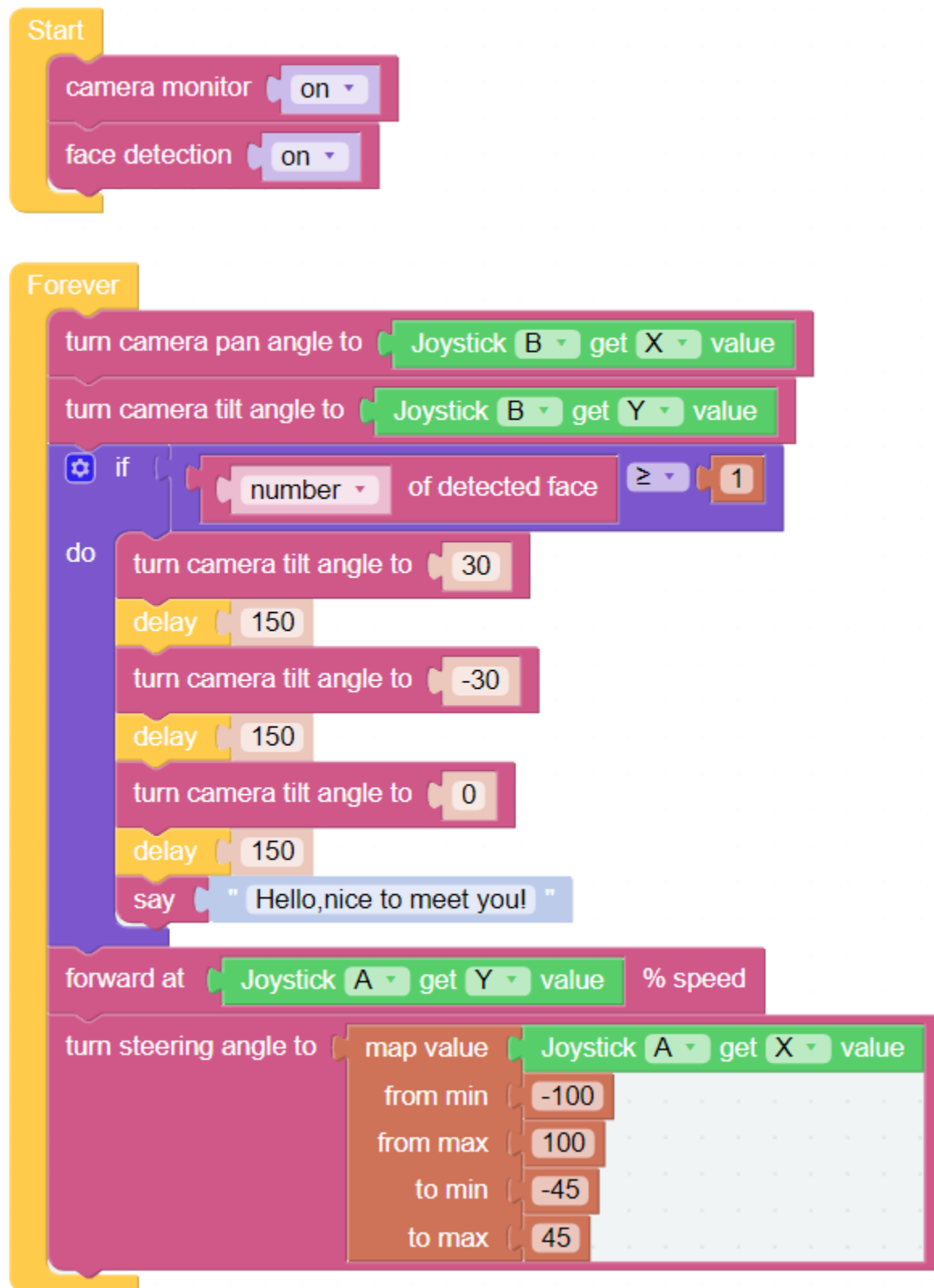The **if do** block is used to nod politely once the conditional judgment of "if" is true.

The **conditional statements** block is used in conjunction with the **if do** block. The conditions can be "=", ">", "<", " ", " ", or " ".
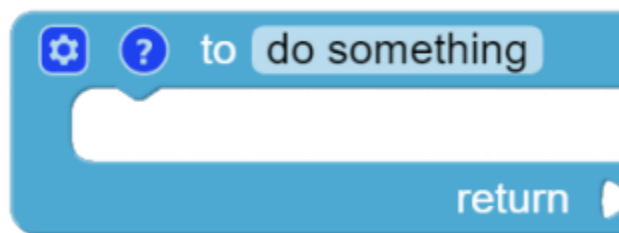
EXAMPLE

Note:

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

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

Start
camera monitor | on ▾
face detection | on ▾

Forever
turn camera pan angle to | Joystick B ▾ get X ▾ value
turn camera tilt angle to | Joystick B ▾ get Y ▾ value
⚙ if | number ▾ of detected face | ≥ ▾ | 1
do turn camera tilt angle to | 30
delay | 150
turn camera tilt angle to | -30
delay | 150
turn camera tilt angle to | 0
delay | 150
say | " Hello,nice to meet you! "
forward at | Joystick A ▾ get Y ▾ value | % speed
turn steering angle to | map value | Joystick A ▾ get X ▾ value
from min | -100
from max | 100
to min | -45
to max | 45

5.13 Music Car

This project will turn the PiCar-X into a music car that will travel around your home, playing cheerful music. This project will also show how the PiCar-X avoids hitting walls with the built-in ultrasonic sensor.

TIPS
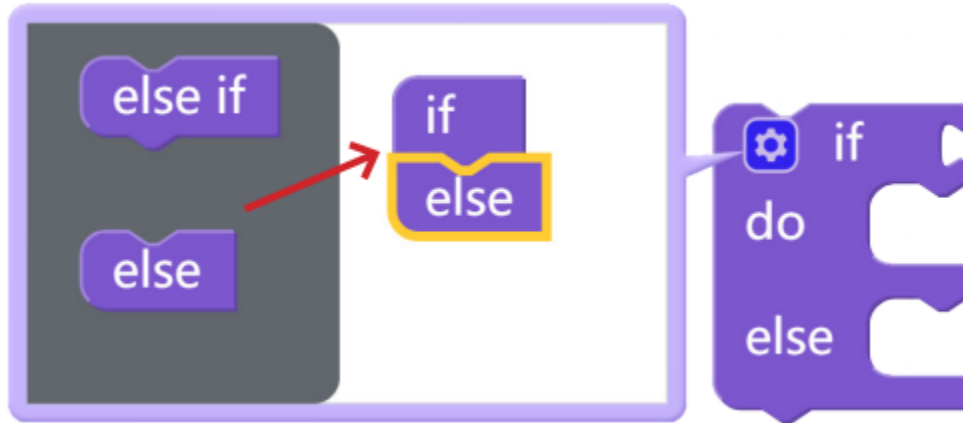
To implement multiple conditional judgments, change the simple if do block into an if else do / else if do block. This is done by clicking on the setting icon as shown above.

EXAMPLE

Note:

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

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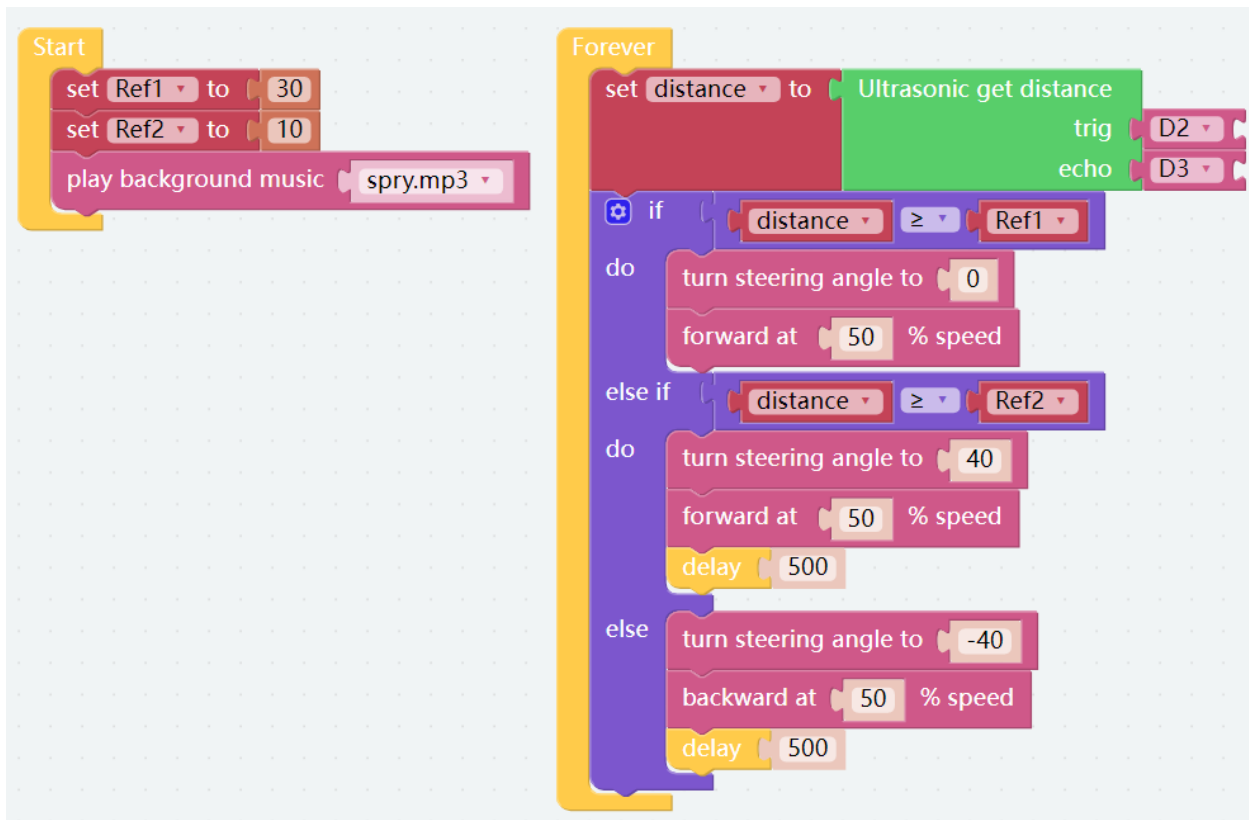
5.14 Cliff Detection

This project will use the **grayscale module** to prevent the PiCar-X from falling off a cliff while it is moving freely around your home. This is an essential project for houses with staircases.

TIPS

The **grayscale module** will be performing the same operation multiple times. To simplify the program, this project introduces a **function** that will return a **list** variable to the **do forever** block.

EXAMPLE

Note:

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

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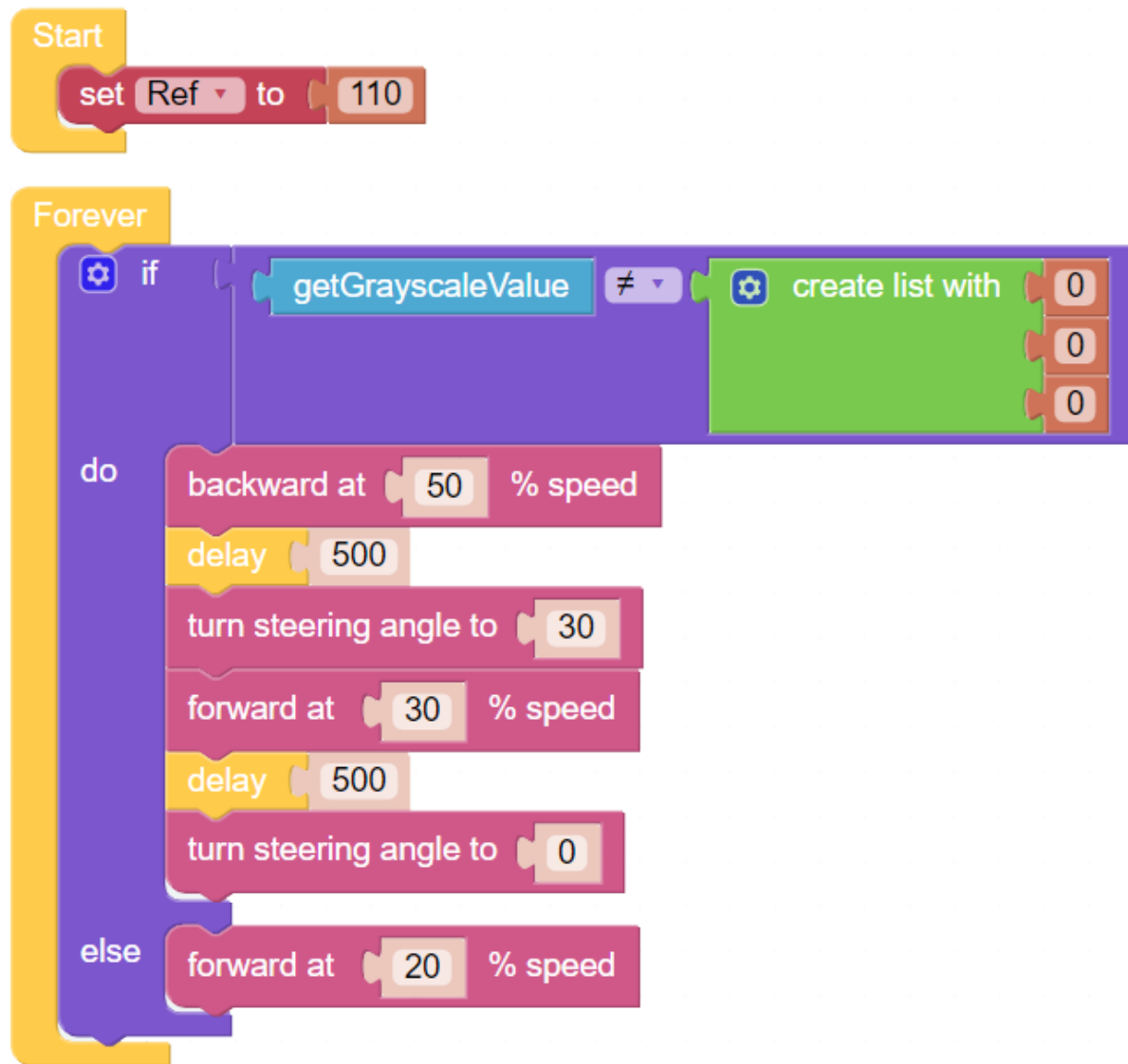
5.15 Minecart

Let's make a minecart project! This project will use the Grayscale module to make the PiCar-X move forward along a track. Use dark-colored tape to make a track on the ground as straight as possible, and not too curved. Some experimenting might be needed if the PiCar-X becomes derailed.

When moving along the track, the probes on the left and right sides of the Grayscale module will detect light-colored ground, and the middle probe will detect the track. If the track has an arc, the probe on the left or right side of the sensor will detect the dark-colored tape, and turn the wheels in that direction. If the minecart reaches the end of the track or derails, the Grayscale module will no longer detect the dark-colored tape track, and the PiCar-X will come to a stop.

TIPS

- **Set ref to** () block is used to set the grayscale threshold, you need to modify it according to the actual situation. You can go ahead and run *Test Grayscale Module* to see the values of the grayscale module on the white and

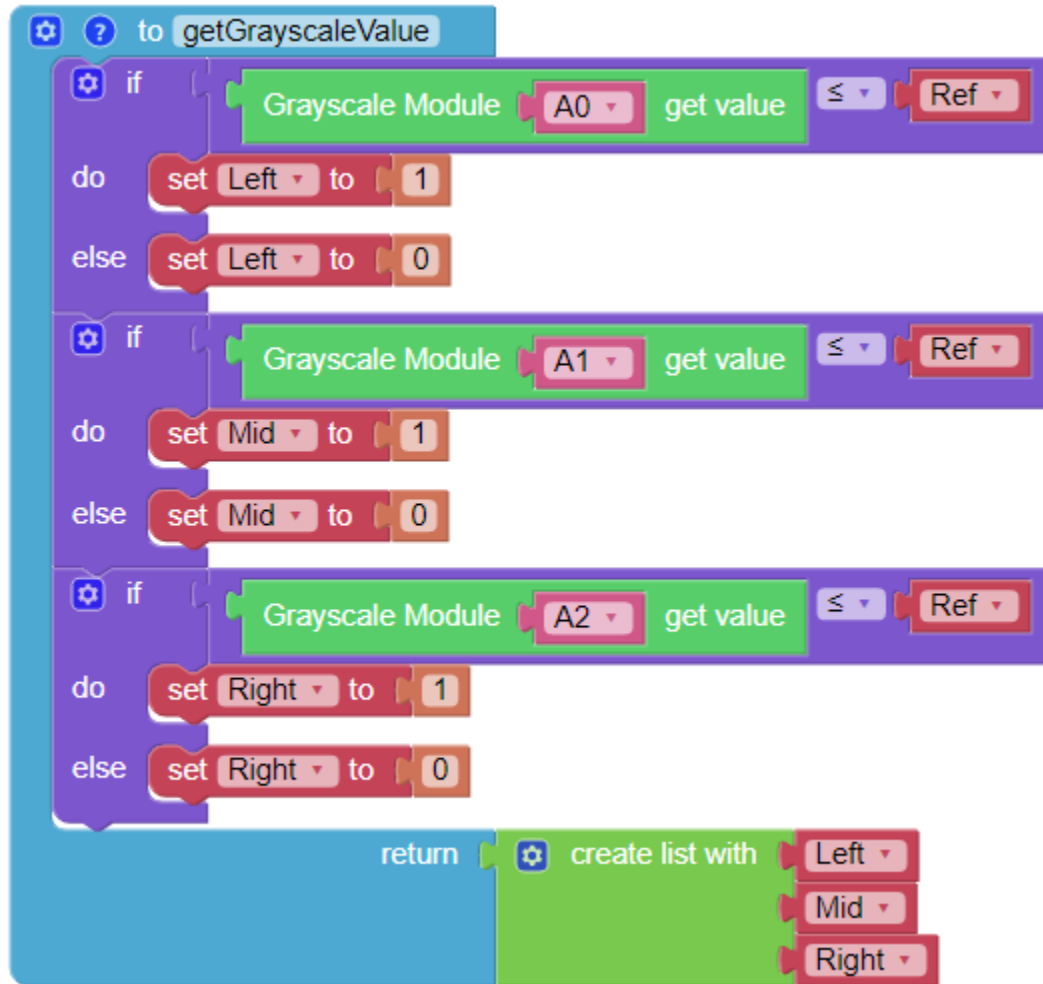black surfaces, and fill in their middle values in this block.

EXAMPLE

Note:

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

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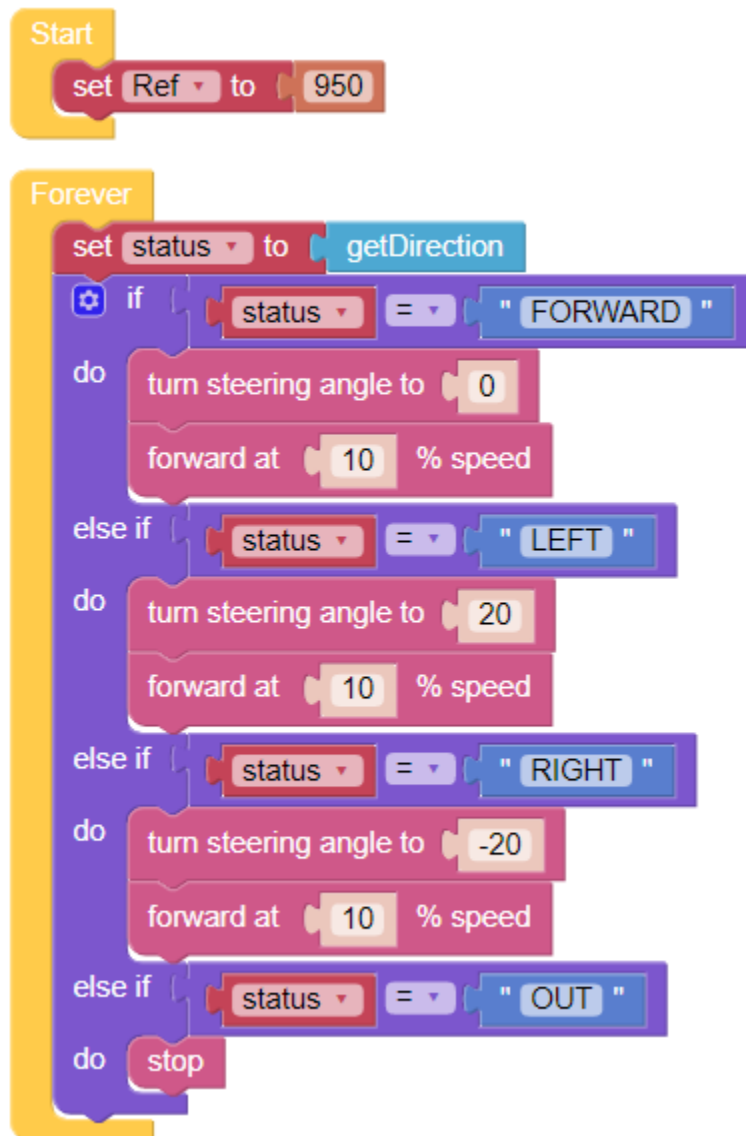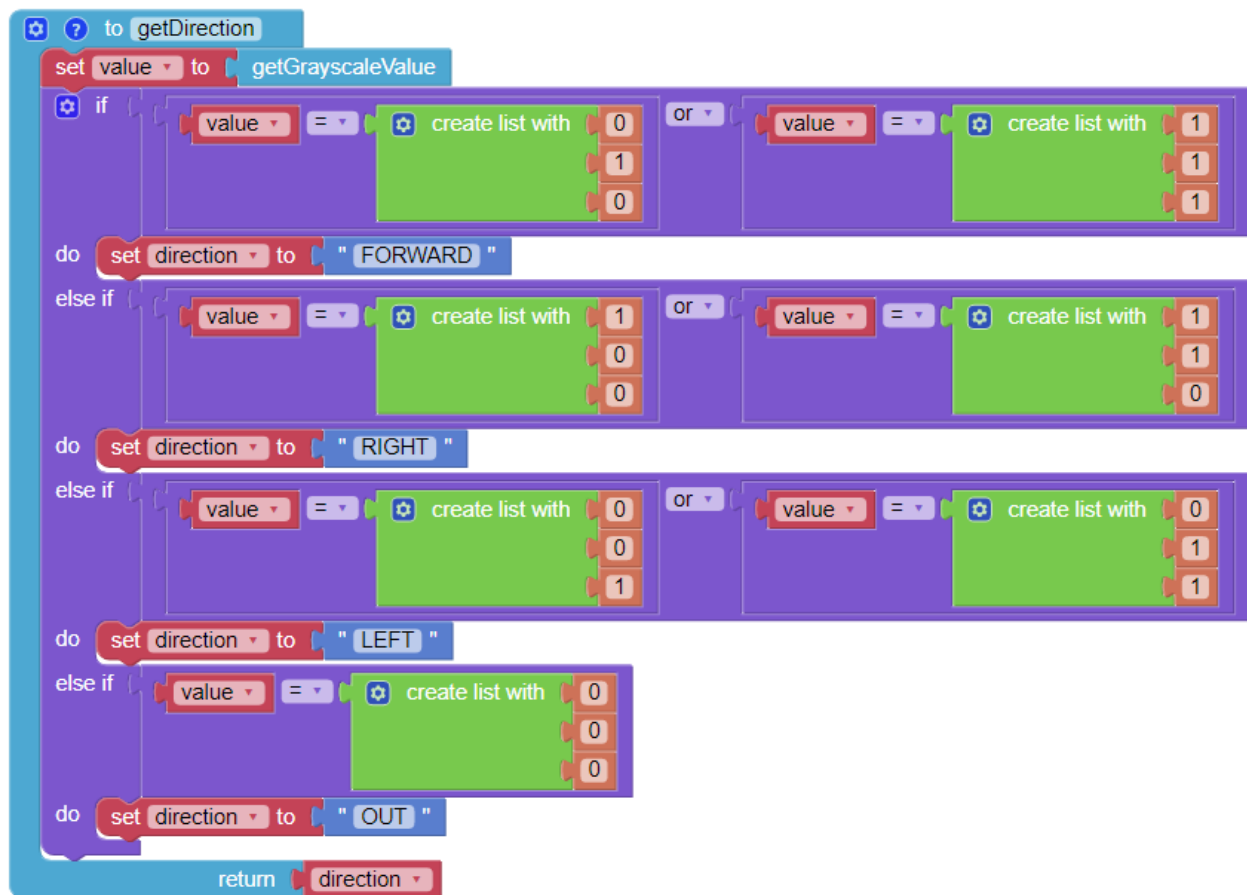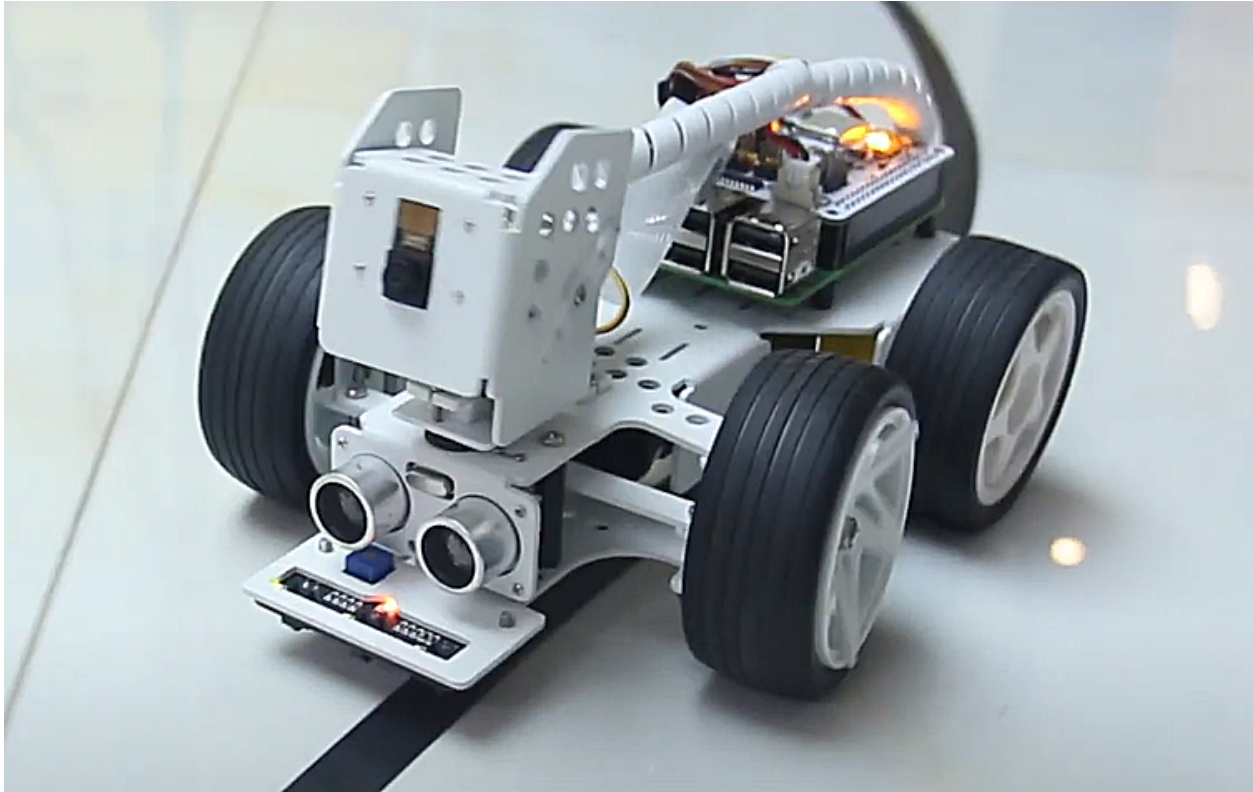
5.16 Minecart Plus

In this project, derailment recovery has been added to the *Minecart* project to let the PiCar-X adapt and recover from a more severe curve.

TIPS

1. Use another **to do something** block to allow the PiCar-X to back up and recover from a sharp curve. Note that the new **to do something** function does not return any values, but is used just for reorienting the PiCar-X.
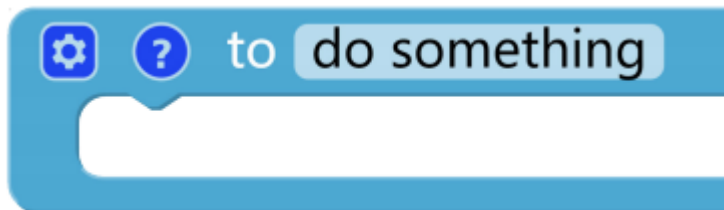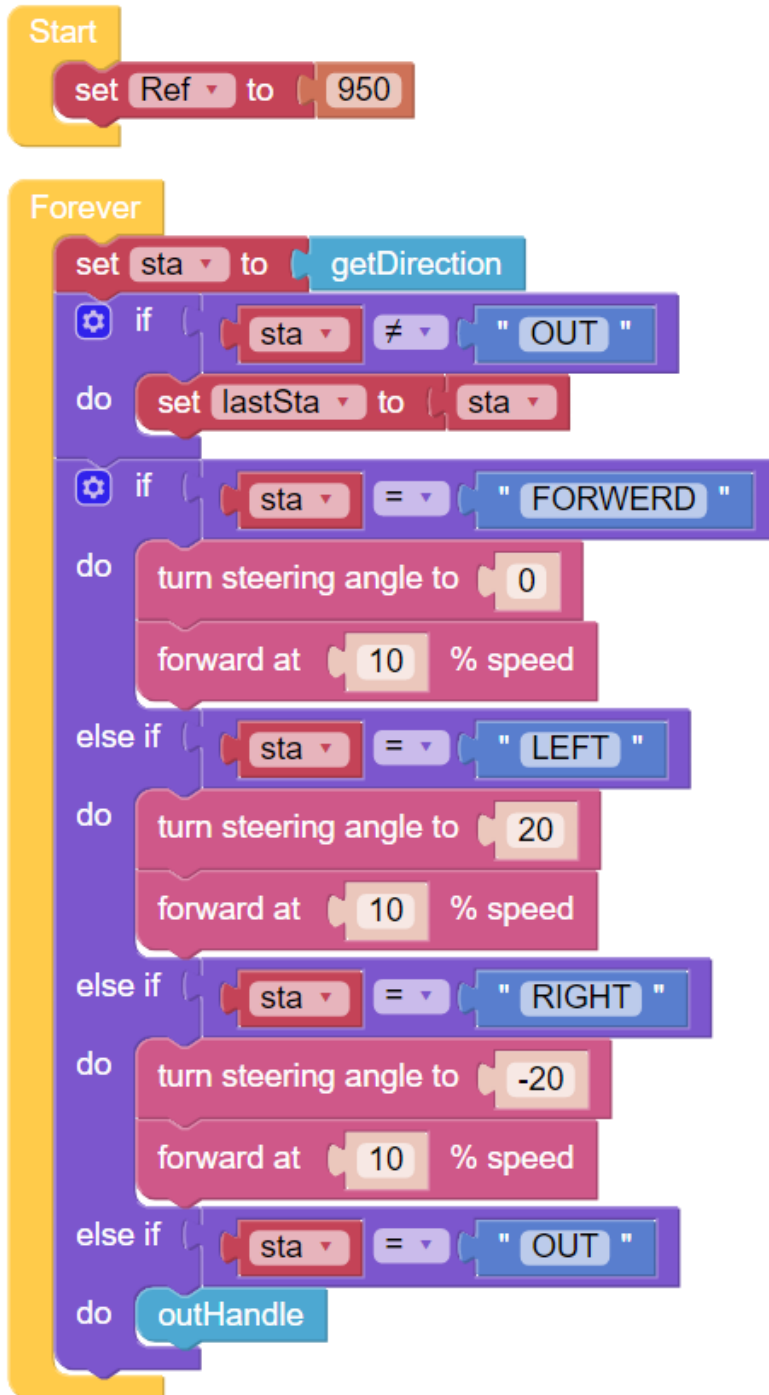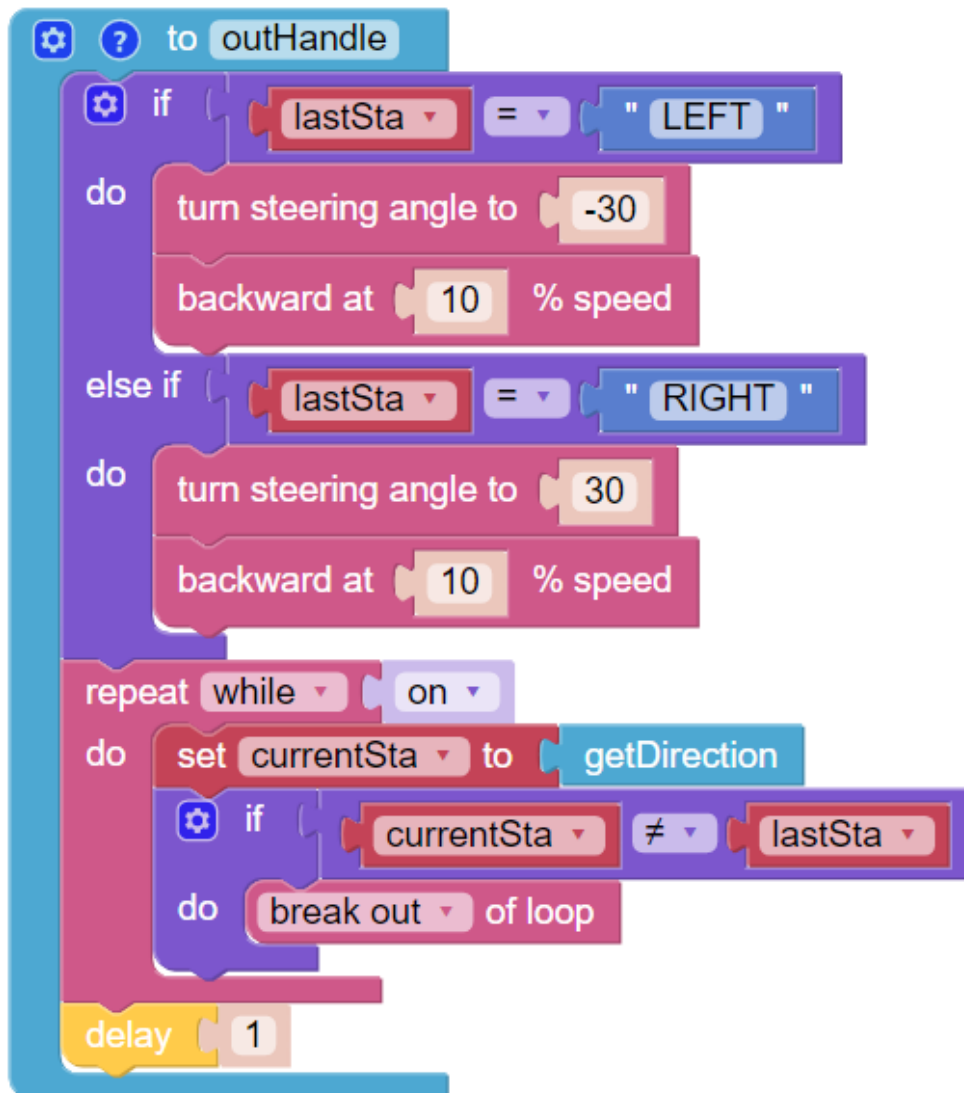

2. **Set ref to ()** block is used to set the grayscale threshold, you need to modify it according to the actual situation. You can go ahead and run *Test Grayscale Module* to see the values of the grayscale module on the white and black surfaces, and fill in their middle values in this block.
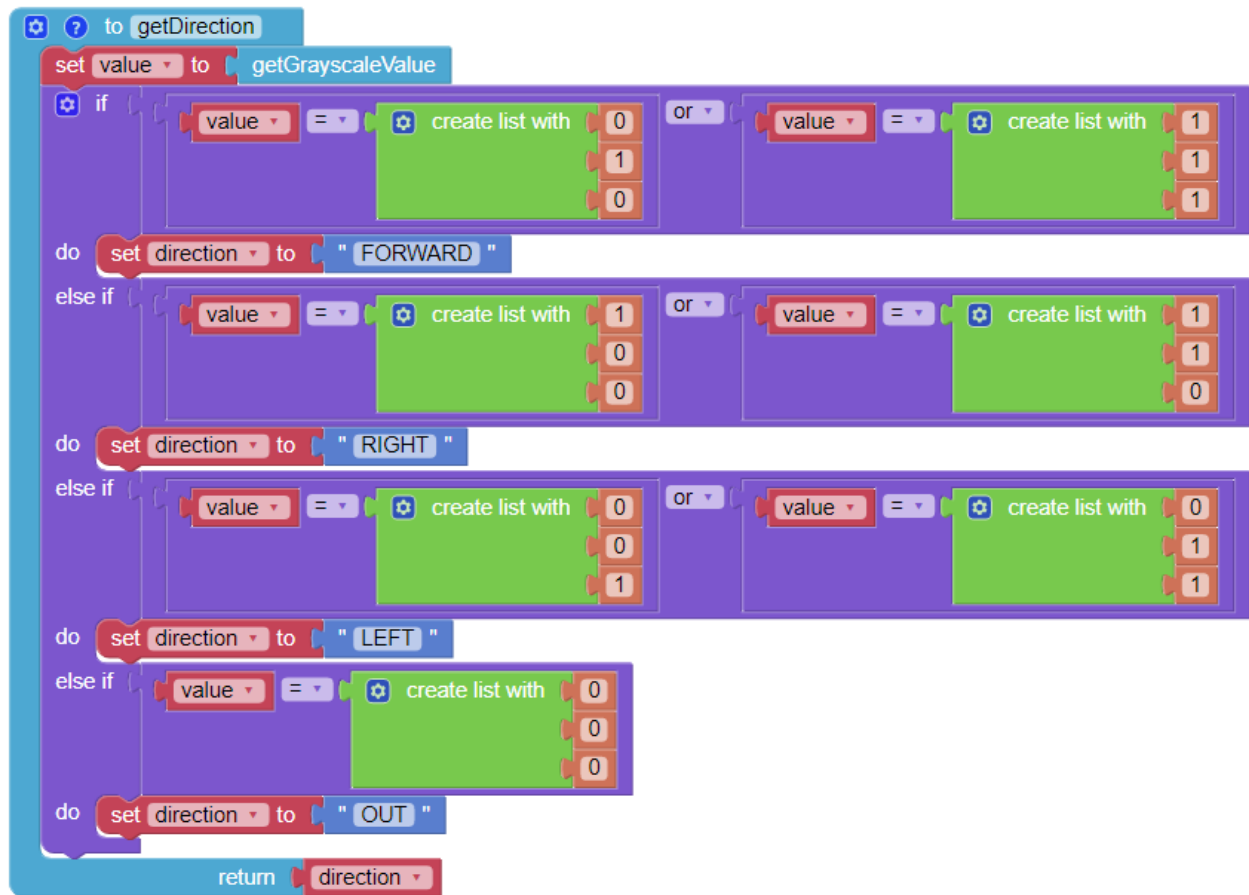
EXAMPLE

Note:

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

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

```
Start
    set  Ref ▾  to    950

Forever
    set  sta ▾  to    getDirection
    ⚙ if      sta ▾  ≠ ▾    " OUT "
      do   set  lastSta ▾  to    sta ▾

    ⚙ if      sta ▾  = ▾    " FORWERD "
      do   turn steering angle to   0
           forward at   10   % speed
    else if      sta ▾  = ▾    " LEFT "
      do   turn steering angle to   20
           forward at   10   % speed
    else if      sta ▾  = ▾    " RIGHT "
      do   turn steering angle to   -20
           forward at   10   % speed
    else if      sta ▾  = ▾    " OUT "
      do   outHandle
```
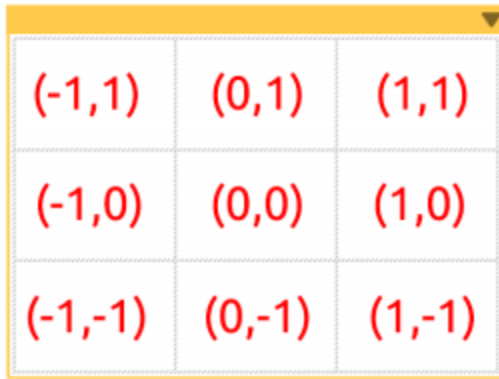
5.17 Bullfight

Turn PiCar-X into an angry bull! Prepare a red cloth, such as a handkerchief, and become a Bullfighter. When the PiCar-X chases after the red cloth, be careful not to get hit!

Note: This project is more advanced than the preceding projects. The PiCar-X will need to use the color detection function to keep the camera facing towards the red cloth, then the body orientation will need to automatically adjust in response to the direction that the camera is facing.


TIPS

Begin with adding the **color detection [red]** block to the **Start** widget to make the PiCar-X look for a red-colored object. In the forever loop, add the **[width] of detected color** block to transform the input into an "object detection" grid.

The "object detection" will output the detected coordinates in (x, y) values, based on the center point of the camera image. The screen is divided into a 3x3 grid, as shown below, so if the red cloth is kept in the top left of the cameras' image, the (x, y) coordinates will be (-1, 1).
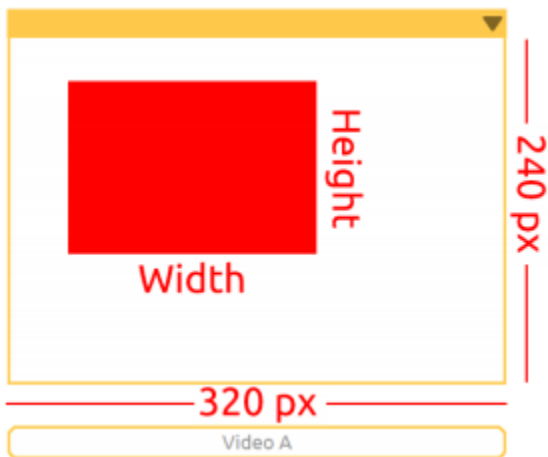

The "object detection" will detect the Width and Height of the graphic. If multiple targets are identified, the dimensions of the largest target will be recorded.

EXAMPLE

Note:

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

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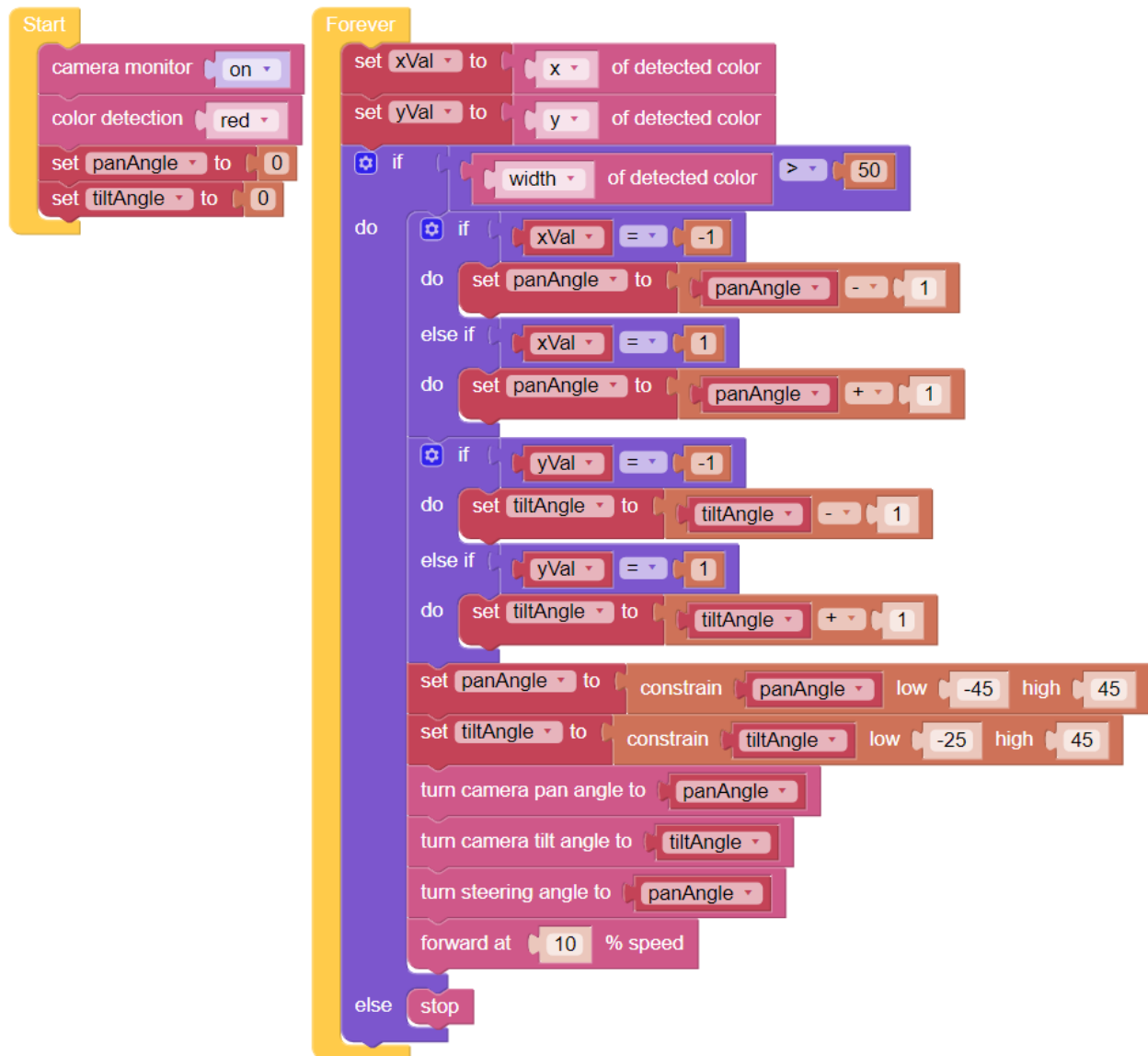
5.18 Beware of Pedestrians

This project will make the PiCar-X perform appropriate measures based on road conditions. While driving, the PiCar-X will come to a complete stop if a pedestrian is detected in its path.

Once the program is running, hold a photo of a person in front of the PiCar-X. The Video Monitor will detect the person's face, and the PiCar-X will automatically come to a stop.

To simulate driving safety protocols, a judgment procedure is created that will send a **[count]** value to a **if do else** block. The judgement procedure will look for a human face 10 times, and if a face does appear it will increment **[count]** by +1. When **[count]** is larger than 3, the PiCar-X will stop moving.

- How to Use the Remote Control Function?
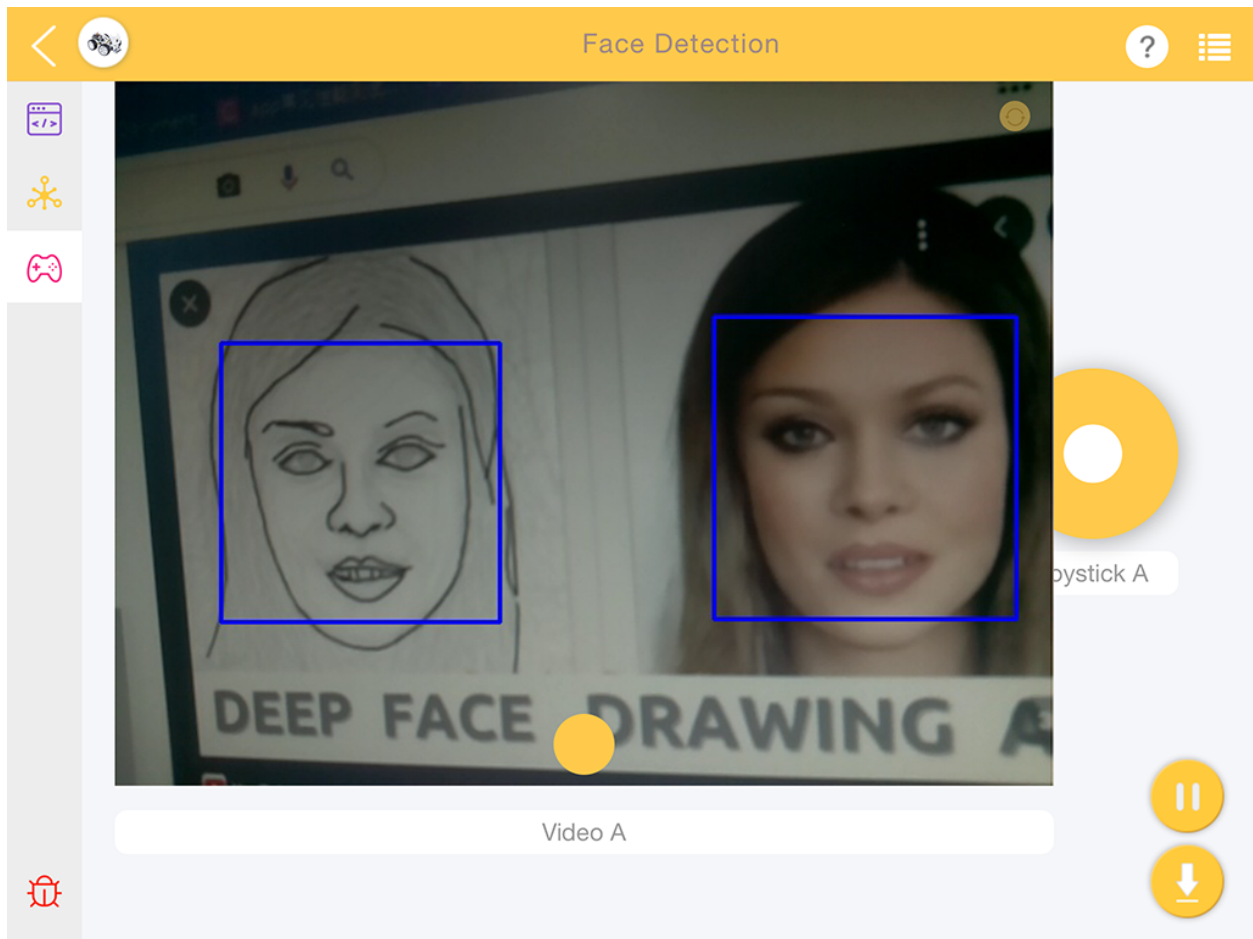
EXAMPLE

Note:

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

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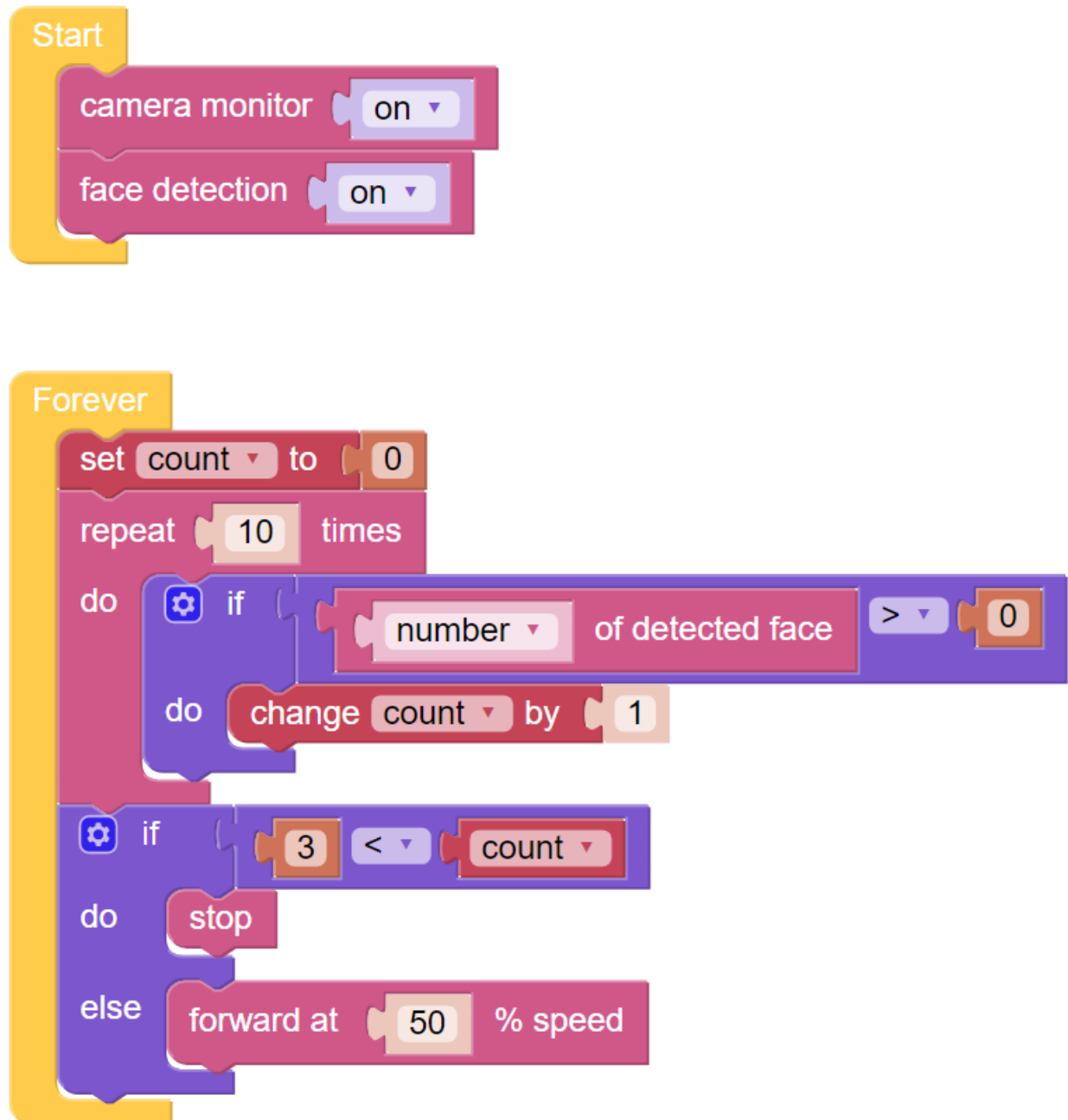
5.19 Traffic Sign Detection

In addition to color, face detection, PiCar-X can also do traffic sign detection.

Now let's combine this traffic sign detection with the line following function. Let PiCar-X track the line, and when you put the Stop sign in front of it, it will stop. When you place a Forward sign in front of it, it will continue to move forward.

TIPS

1. PiCar will recognize 4 different traffic sign models included in the printable PDF below.

- [PDF]Traffic Sign Cards

2. **Set ref to ()** block is used to set the grayscale threshold, you need to modify it according to the actual situation. You can go ahead and run *Test Grayscale Module* to see the values of the grayscale module on the white and black surfaces, and fill in their middle values in this block.
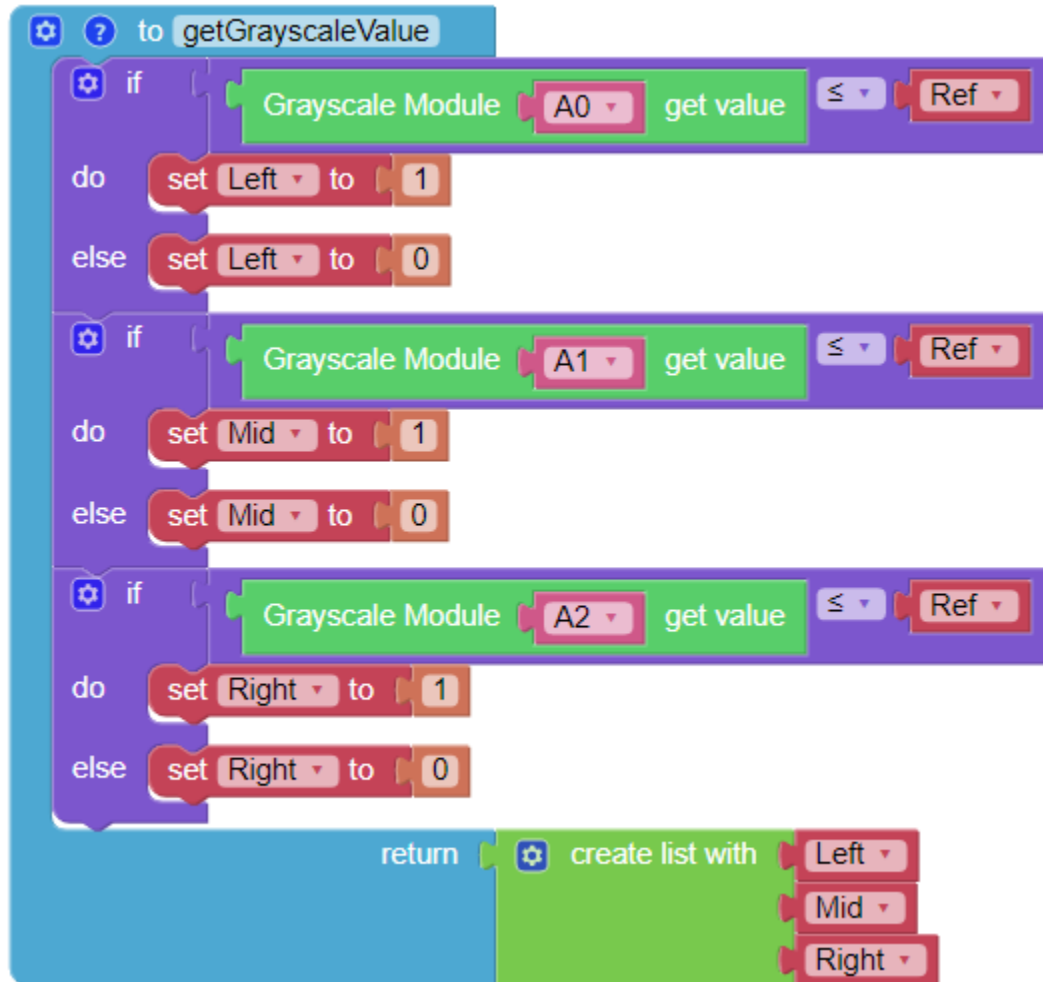
EXAMPLE

Note:

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

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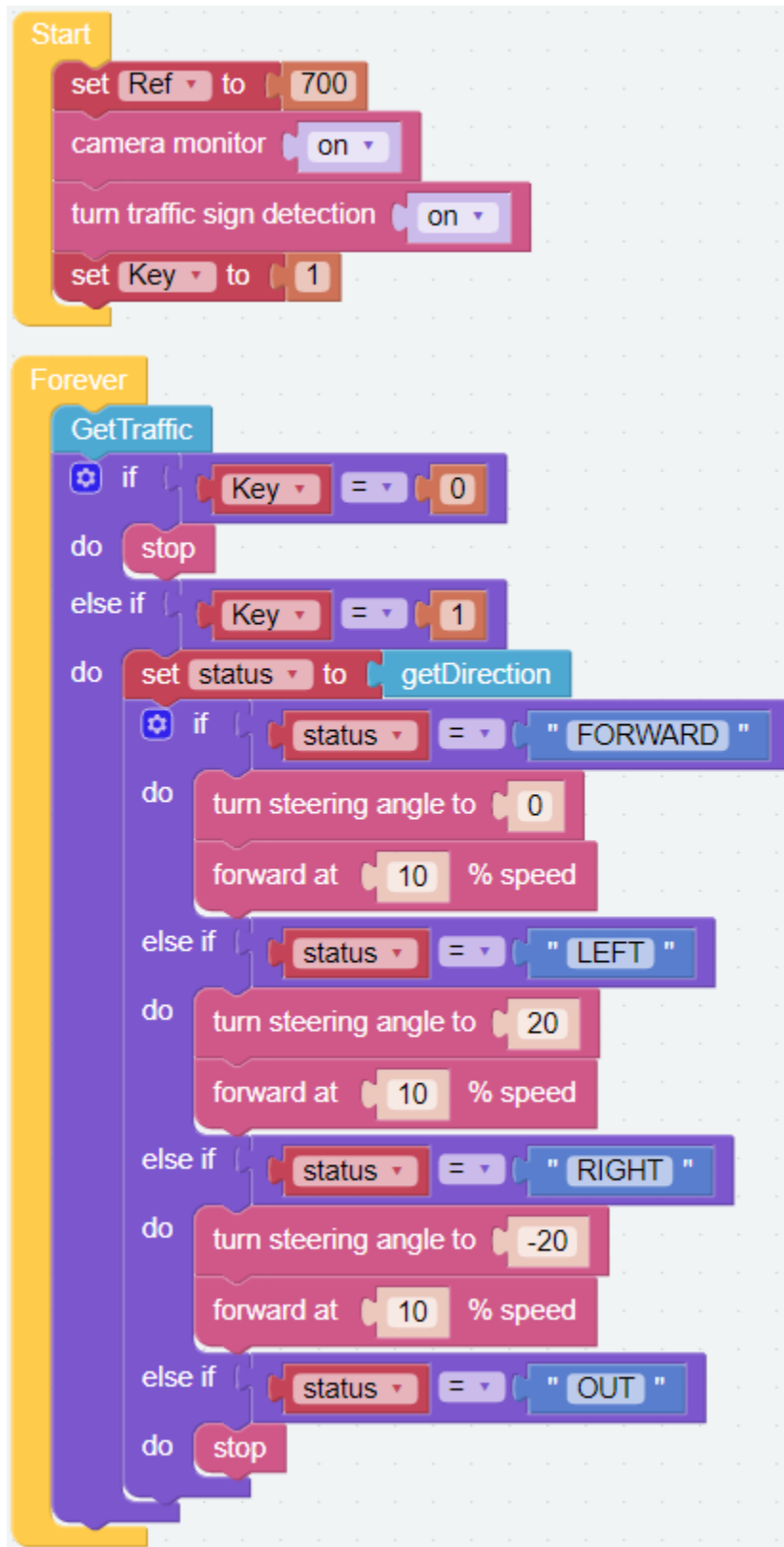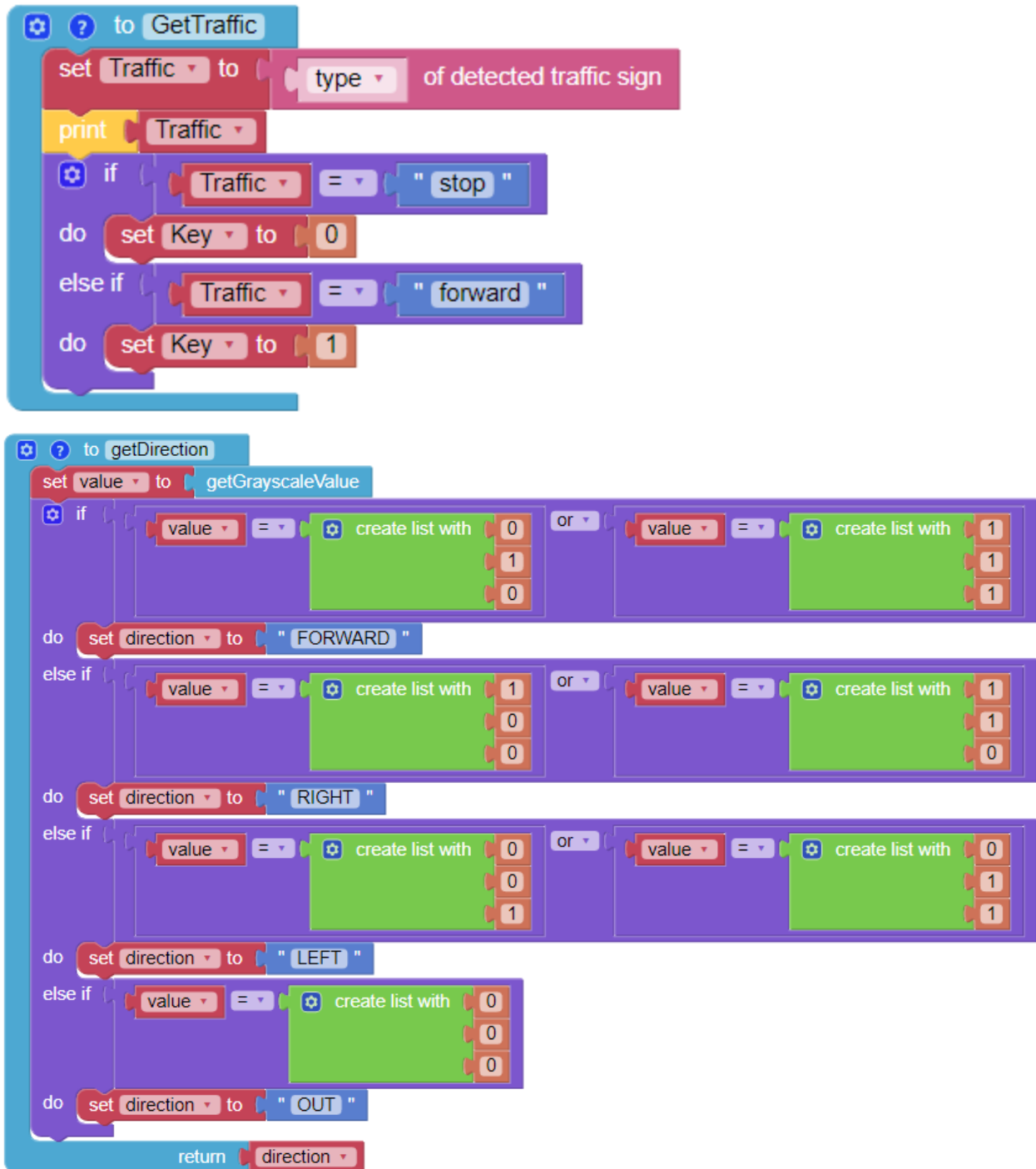
5.20 Orienteering

This project uses the remote control function to guide the PiCar-X through a competitive scavenger hunt!

First, set up either an obstacle course, or a maze, or even an empty room that the PiCar-X can drive through. Then, randomly place six markers along the route, and put a color-card at each of the six markers for the PiCar-X to find.

The six color models for PiCar-X are: red, orange, yellow, green, blue and purple, and are ready to print from a colored printer from the PDF below.

- [PDF]Color Cards

Note: The printed colors may have a slightly different hue from the Ezblock color models due to printer toner differences, or the printed medium, such as a tan-colored paper. This can cause a less accurate color recognition.

The PiCar-X will be programmed to find three of the six colors in a random order, and will be using the TTS function to announce which color to look for next.

The objective is to help the PiCar-X find each of the three colors in as short of a time as possible.

Place PiCar-X in the middle of the field and click the Button on the Remote Control page to start the game.
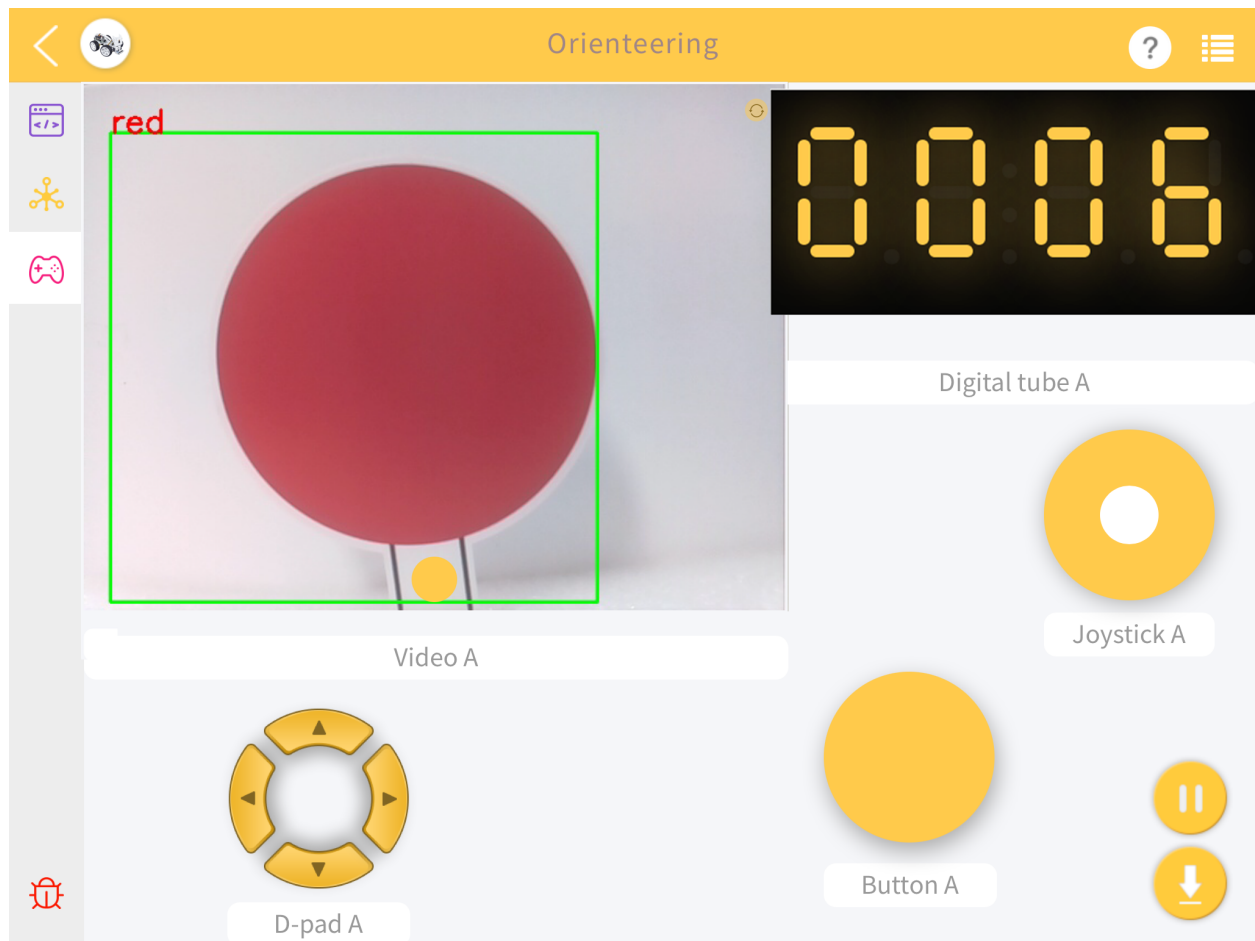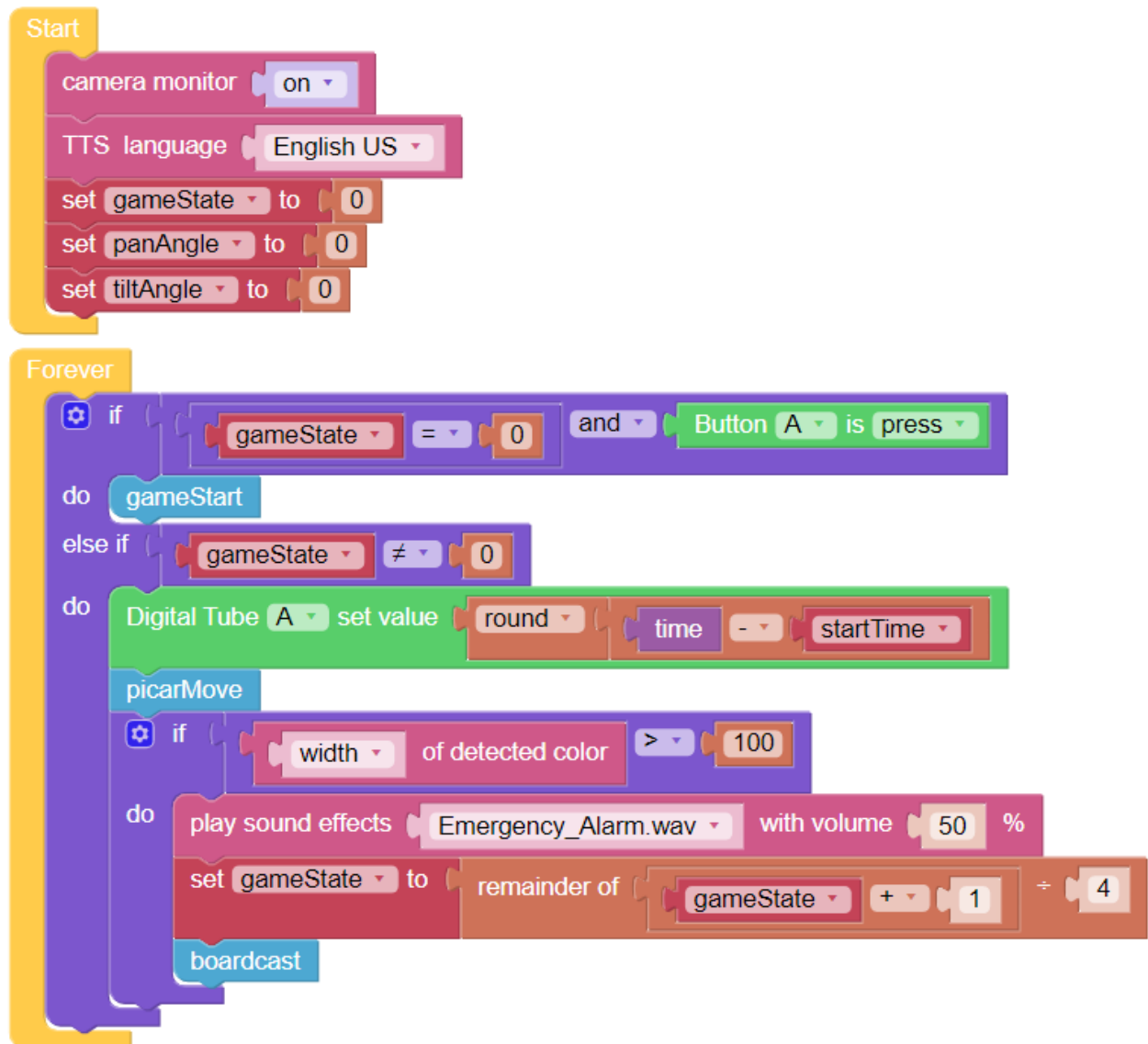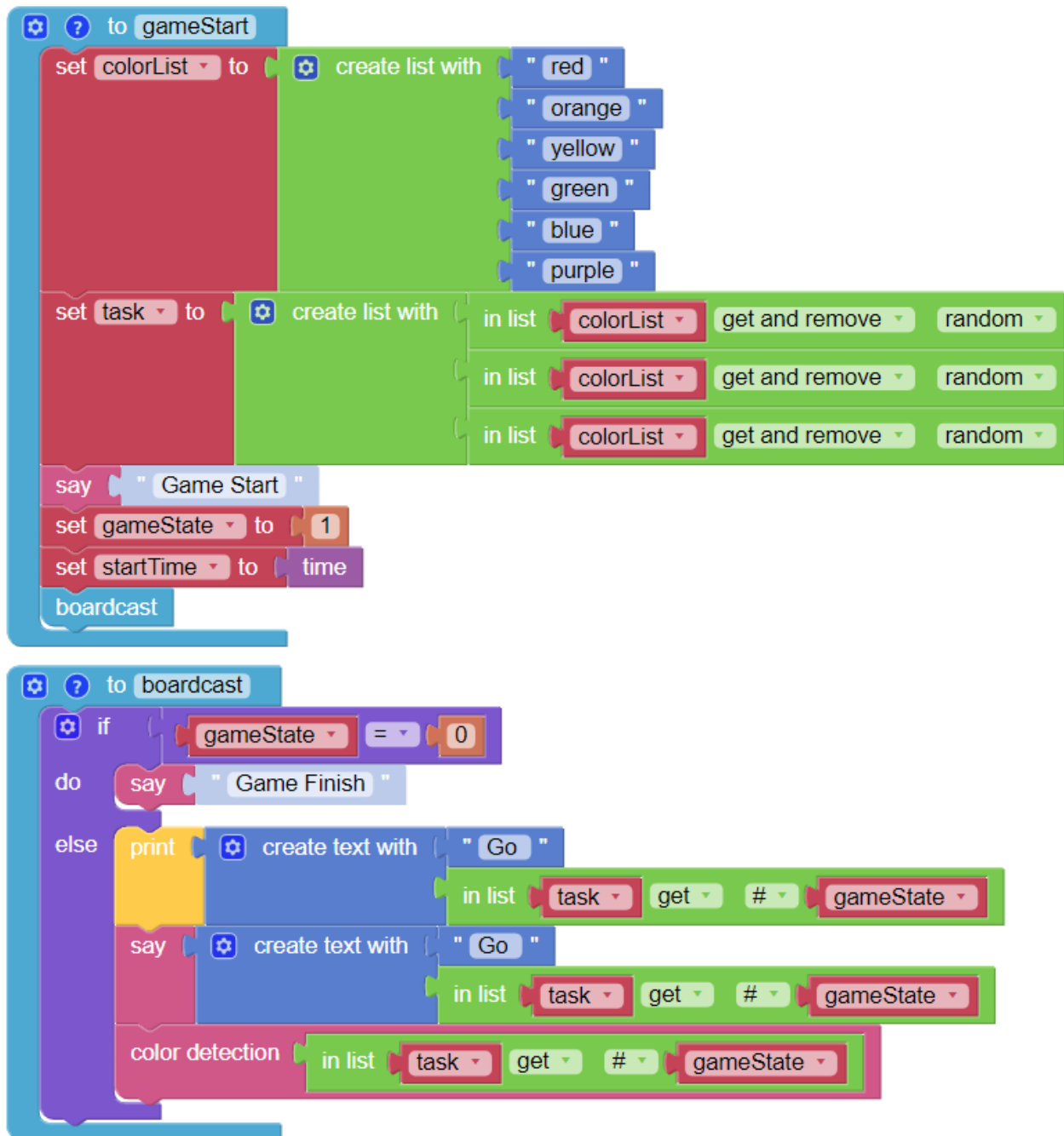
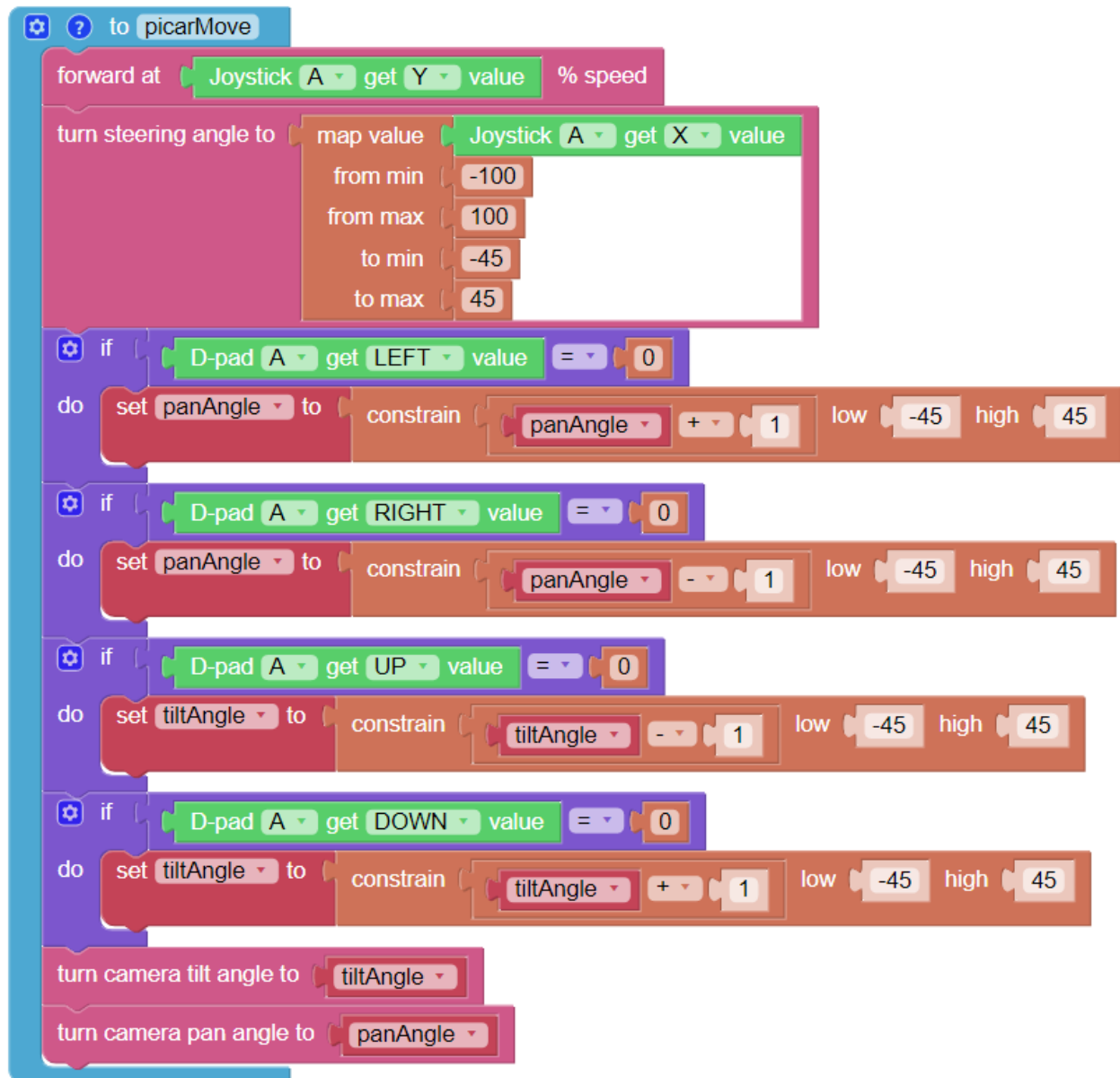Take turns playing this game with friends to see who can help PiCar-X complete the objective the fastest!

EXAMPLE

Note:

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

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

```
to picarMove
    forward at ( Joystick A get Y value    % speed
    turn steering angle to ( map value ( Joystick A get X value
                            from min ( -100
                            from max ( 100
                            to min ( -45
                            to max ( 45
    if ( D-pad A get LEFT value = ( 0
    do  set panAngle to ( constrain ( panAngle + 1    low ( -45  high ( 45
    if ( D-pad A get RIGHT value = ( 0
    do  set panAngle to ( constrain ( panAngle - 1    low ( -45  high ( 45
    if ( D-pad A get UP value = ( 0
    do  set tiltAngle to ( constrain ( tiltAngle - 1    low ( -45  high ( 45
    if ( D-pad A get DOWN value = ( 0
    do  set tiltAngle to ( constrain ( tiltAngle + 1    low ( -45  high ( 45
    turn camera tilt angle to ( tiltAngle
    turn camera pan angle to ( panAngle
```

APPENDIX

6.1 Filezilla Software

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

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

Step 1: Download Filezilla.

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

Step 2: Connect to Raspberry Pi

After a quick install open it up and now connect it to an FTP server. It has 3 ways to connect, here we use the **Quick Connect** bar. Enter the **hostname/IP**, **username**, **password** and **port (22)**, then click **Quick Connect** or press **Enter** to connect to the server.
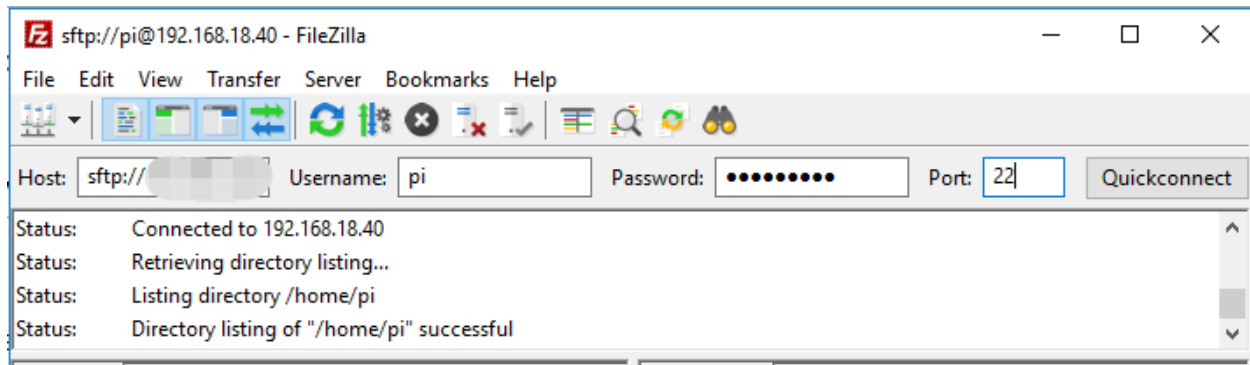
Note: Quick Connect is a good way to test your login information. If you want to create a permanent entry, you can select **File**-> **Copy Current Connection to Site Manager** after a successful Quick Connect, enter the name and click **OK**. Next time you will be able to connect by selecting the previously saved site inside **File -> Site Manager**.
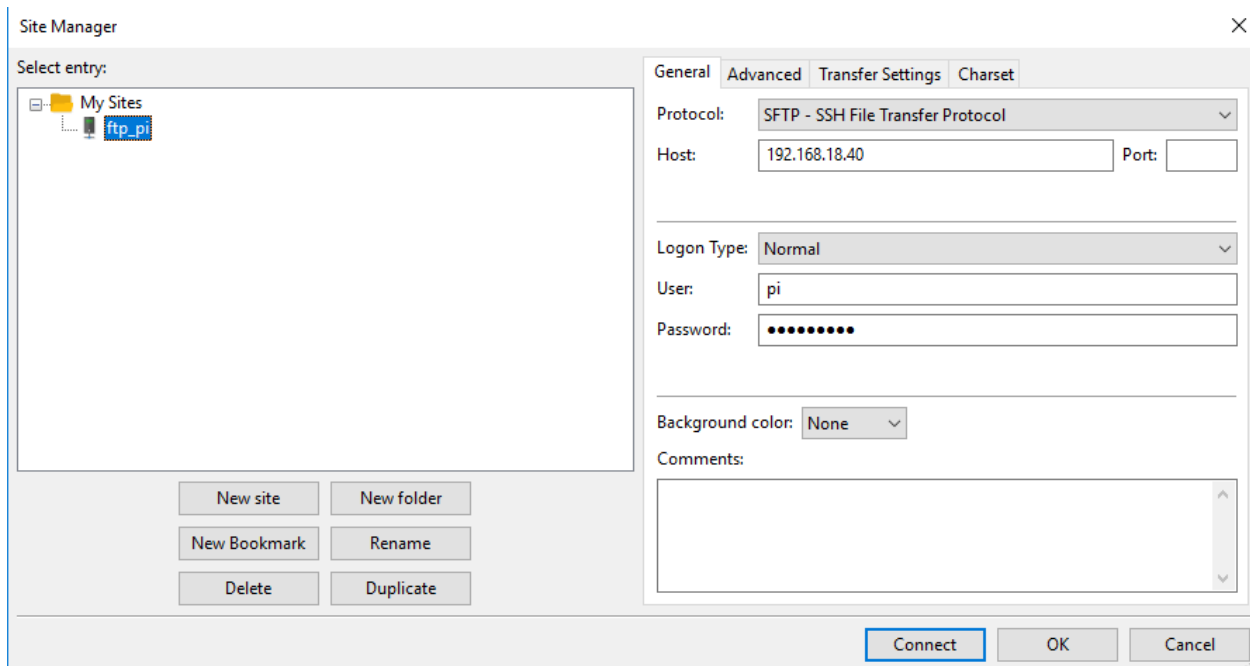
Step 3: Upload/download files.

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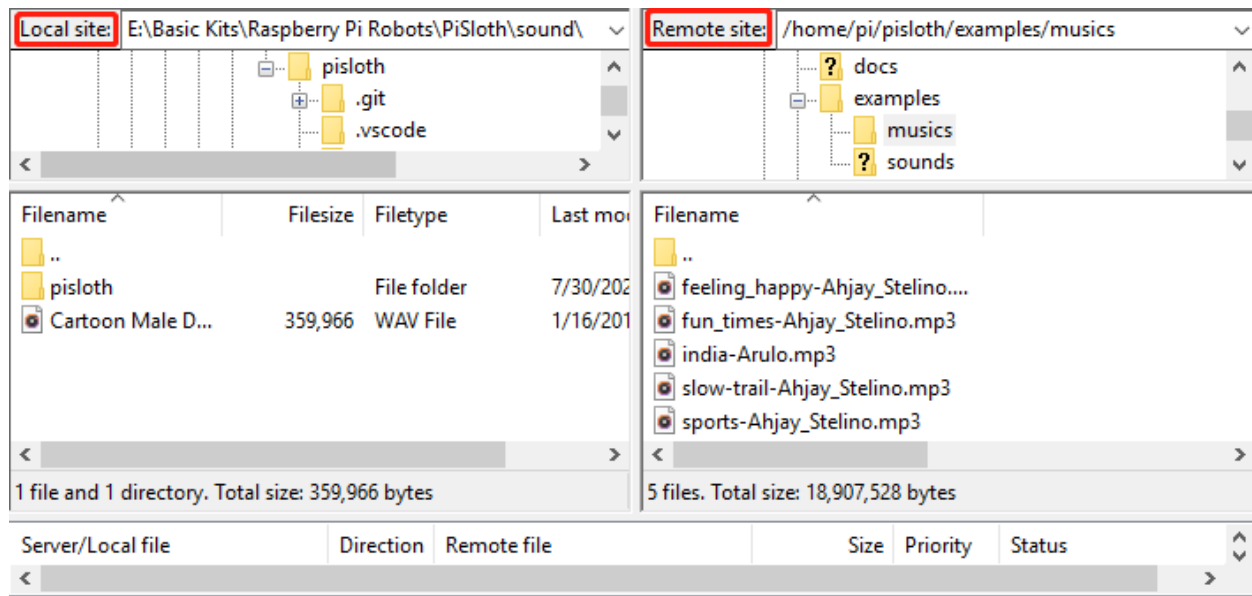
6.2 PuTTY

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).
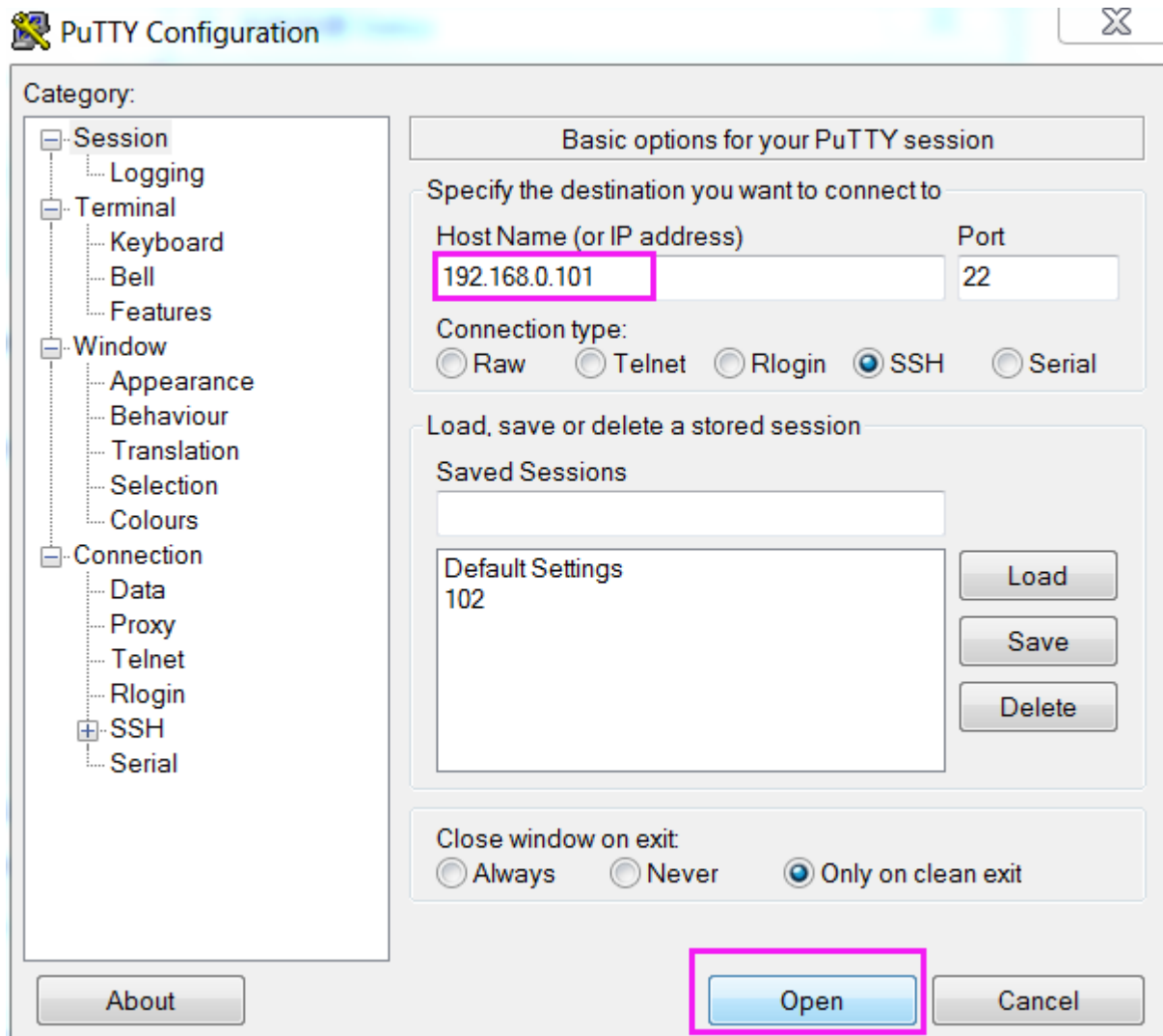
Step 3
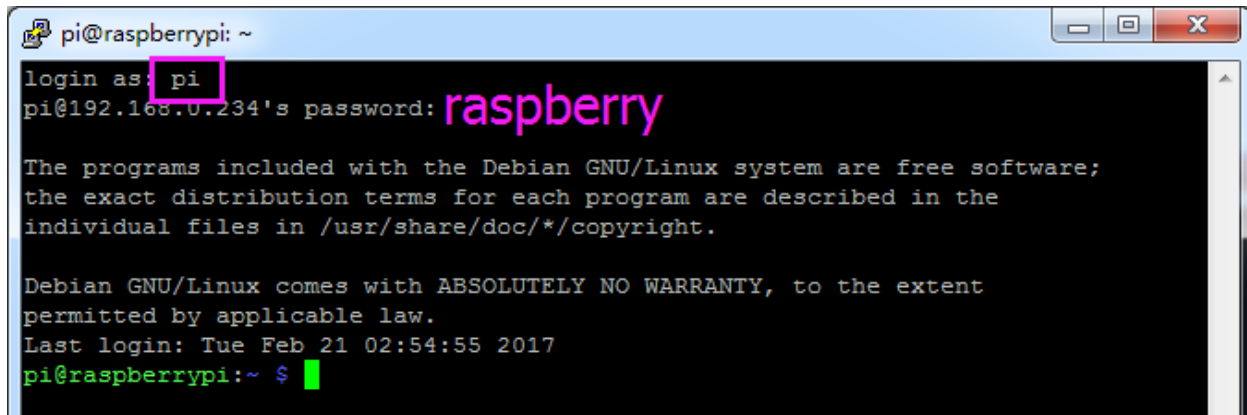
Click **Open**. Note that when you first log in to the Raspberry Pi with the IP address, there prompts a security reminder. Just click **Yes**.

Step 4

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

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

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

Step 5

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

6.3 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.
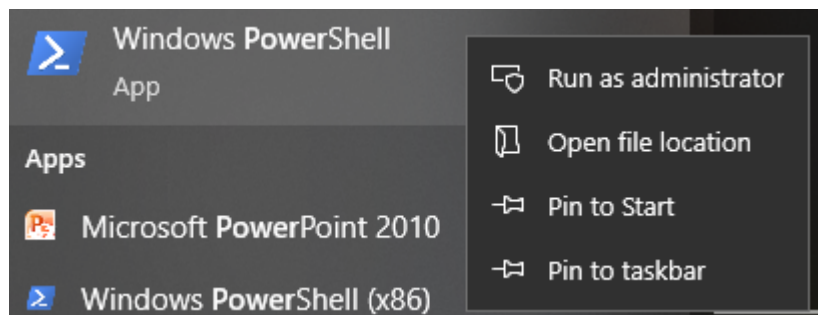
2. Use the following command to install `OpenSSH.Client`.

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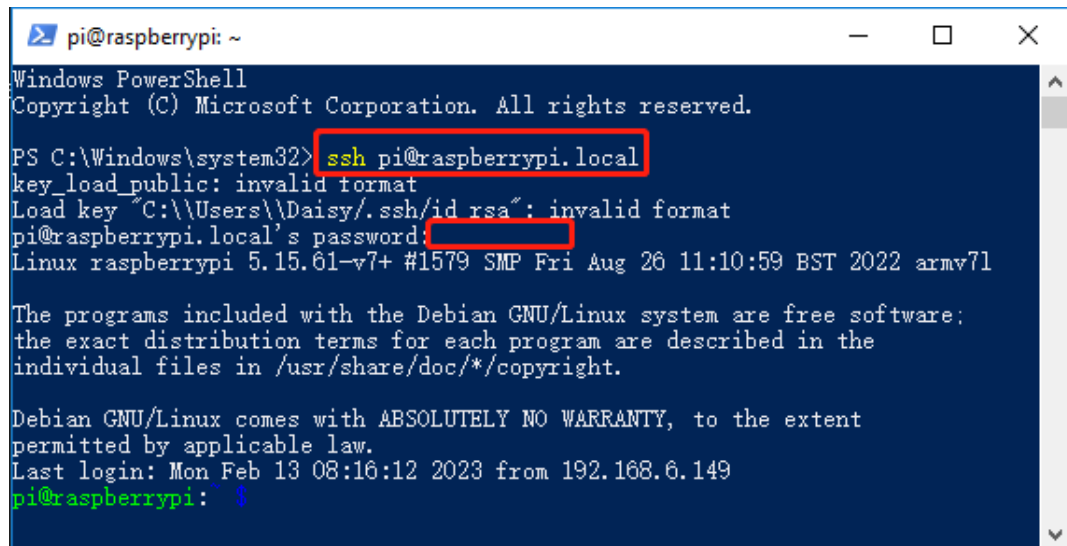
5. It now tells you that `OpenSSH.Client` has been successfully installed.

```
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.

## 6.4 About the Battery

Applicable Parameters

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

Note:

- Robot HAT cannot charge the battery, so you need to buy a battery charger.

- When the two power indicators on the Robot HAT are off, it means the power is too low and the batteries need to be charged.

Button Top vs Flat Top?

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

Button Top	Flat Top
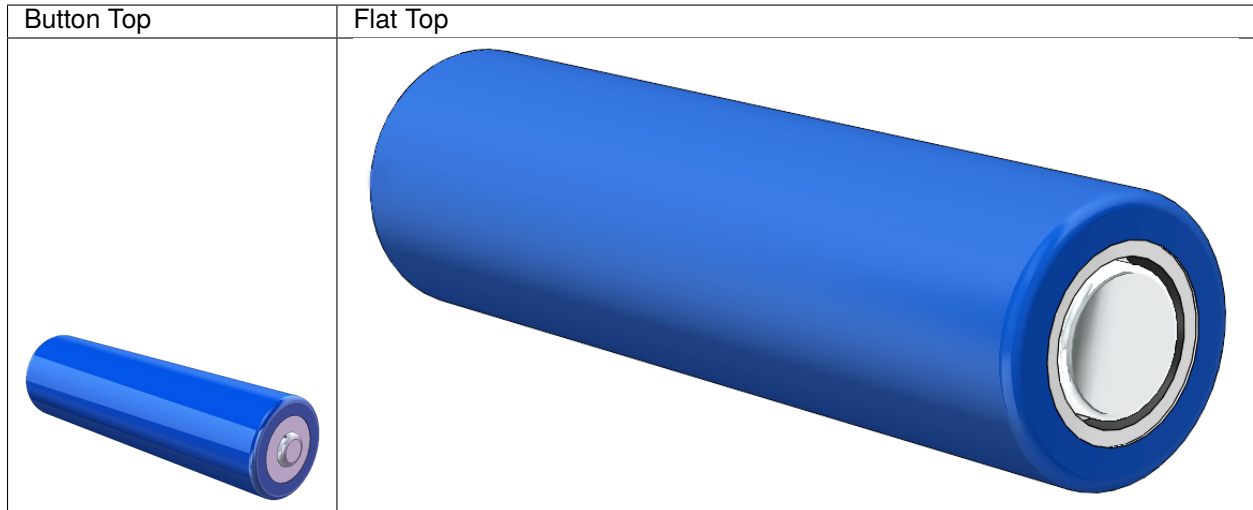
No protective board?

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

Battery capacity?

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

FAQ

Q1: After installing Ezblock OS, the servo can't turn to 0°?

1) Check if the servo cable is properly connected and if the Robot HAT power is on.

2) Press Reset button.

3) If you have already run the program in Ezblock Studio, the custom program for P11 is no longer available. You can refer to the picture below to manually write a program in Ezblock Studio to set the servo angle to 0.
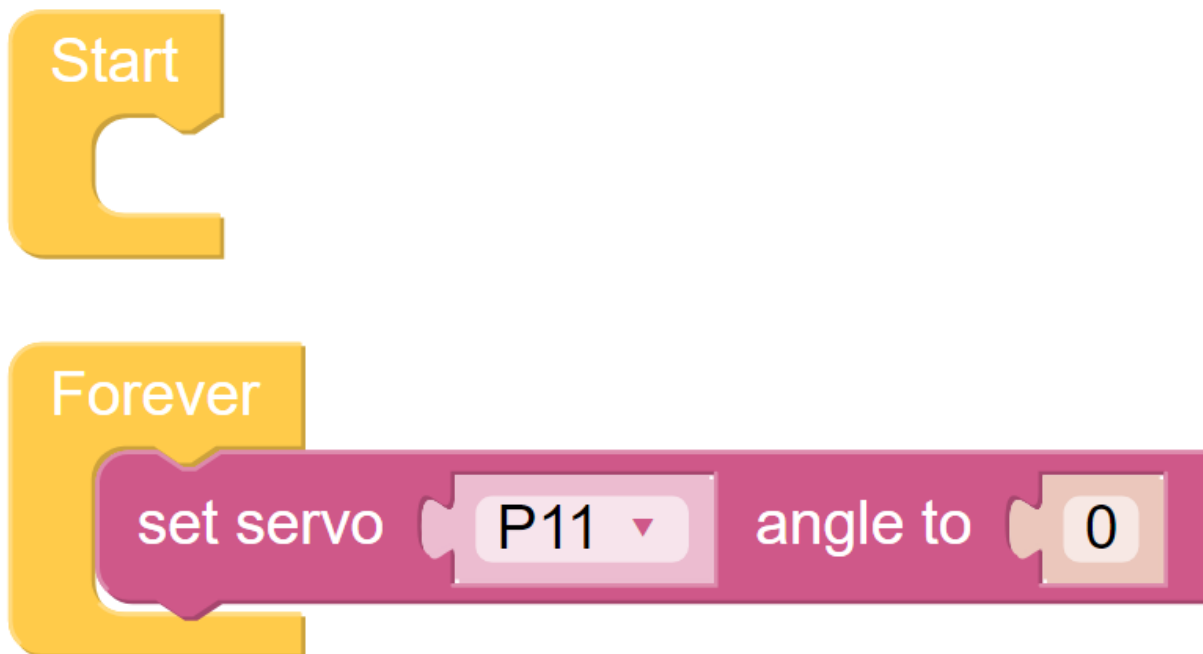
Q2: When using VNC, I am prompted that the desktop cannot be displayed at the moment?

In Terminal, type `sudo raspi-config` to change the resolution.

Q3: Why does the servo sometimes return to the middle position for no reason?

When the servo is blocked by a structure or other object and cannot reach its intended position, the servo will enter the power-off protection mode in order to prevent the servo from being burned out by too much current.

After a period of power failure, if no PWM signal is given to the servothe servo will automatically return to its original position.

EIGHT

THANK YOU

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

Particular Thanks

- Len Davisson

- Kalen Daniel

- Juan Delacosta

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

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

NINE

COPYRIGHT NOTICE