SunFounder PiCar-X Kit

www.sunfounder.com

Oct 21, 2025

CONTENTS

1 Assemble the PiCar-X **5**

2 Play with Python **7**

2.1 1. Quick Guide on Python . . . 7
2.1.1 1. What Else Do You Need? . . . 7
2.1.2 2. Installing the OS . . . 12
2.1.3 3. Power Supply for Raspberry Pi (Important) . . . 22
2.1.4 4. Set up Your Raspberry Pi . . . 24
2.1.5 5. Install All the Modules(Important) . . . 27
2.1.6 6. Enable I2C Interface(Important) . . . 31
2.1.7 7. Servo Adjust(Important) . . . 33
2.2 2. Basic Movement . . . 35
2.2.1 1. Calibrating the PiCar-X . . . 35
2.2.2 2. Let PiCar-X Move . . . 41
2.2.3 3. Keyboard Control . . . 44
2.2.4 4. Obstacle Avoidance . . . 46
2.2.5 5. Cliff Detection . . . 49
2.2.6 6. Line Tracking . . . 51
2.3 3. Computer Vision . . . 54
2.3.1 7. Computer Vision . . . 55
2.3.2 8. Stare at You . . . 62
2.3.3 9. Record Video . . . 64
2.3.4 10. Bull Fight . . . 66
2.3.5 11. Video Car . . . 69
2.3.6 12. Controlled by the APP . . . 72

3 Think · Talk · Drive — AI-Powered with Multi-LLMs **79**

3.1 13. Play Music and Sound Effects . . . 79
3.2 14. Voice Prompt Car with Espeak and Pico2Wave . . . 81
3.2.1 Before You Start . . . 81
3.2.2 1. Testing Espeak . . . 81
3.2.3 2. Testing Pico2Wave . . . 82
3.2.4 3. Voice Prompt Car . . . 83
3.2.5 Troubleshooting . . . 84
3.2.6 Comparison: Espeak vs Pico2Wave . . . 84
3.3 15. AI Storytelling Robot with Piper and OpenAI . . . 85
3.3.1 Before You Start . . . 85
3.3.2 1. Testing Piper . . . 85
3.3.3 2. Testing OpenAI TTS . . . 86
3.3.4 3. Storytelling Robot . . . 90

3.3.5 Troubleshooting . . . 91
3.3.6 Comparison of TTS Engines . . . 92
3.4 16. Voice Controlled Car with Vosk (Offline) . . . 93
3.4.1 Before You Start . . . 93
3.4.2 1. Check Your Microphone . . . 93
3.4.3 2. Test Vosk . . . 94
3.4.4 3. Voice Controlled Car . . . 95
3.4.5 Troubleshooting . . . 97
3.5 17. Text Vision Talk with Ollama . . . 98
3.5.1 Before You Start . . . 98
3.5.2 1. Install Ollama (LLM) and Download Model . . . 98
3.5.3 2. Test Ollama . . . 101
3.5.4 3. Vision Talk with Ollama . . . 102
3.5.5 Troubleshooting . . . 105
3.6 18. Connecting to Online LLMs . . . 106
3.6.1 Before You Start . . . 106
3.6.2 OpenAI . . . 106
3.6.3 Gemini . . . 110
3.6.4 Qwen . . . 114
3.6.5 Grok (xAI) . . . 117
3.6.6 DeepSeek . . . 120
3.6.7 Doubao . . . 122
3.6.8 General . . . 129
3.7 19. Local Voice Chatbot . . . 130
3.7.1 Before You Start . . . 130
3.7.2 Run the Code . . . 131
3.7.3 Code . . . 131
3.7.4 Code Analysis . . . 133
3.7.5 Troubleshooting & FAQ . . . 136
3.8 20. Treasure Hunt . . . 136
3.8.1 Before You Start . . . 137
3.8.2 Run the Code . . . 137
3.8.3 Game Rules . . . 137
3.8.4 Code . . . 137
3.8.5 How It Works . . . 140
3.8.6 Troubleshooting . . . 141
3.9 21. AI Voice Assistant Car . . . 141
3.9.1 Before You Start . . . 142
3.9.2 Run the Example . . . 142
3.9.3 What Will Happen . . . 143
3.9.4 Switching to Other LLMs or TTS . . . 143
3.9.5 Action & Sound Reference . . . 144
3.9.6 Troubleshooting . . . 145

4 Python Video Course **147**
4.1 Video A1: Starting with Raspbrry Pi . . . 147
4.2 Video A2: Assembly of the PICAR-X . . . 148
4.3 Video A3: Calibrate the PiCar-X . . . 149
4.4 Video 1: Motor Move and Steering Control . . . 150
4.5 Video 2: Controlling the PiCar-X using keyboard . . . 150
4.6 Video 3: Text to Speech . . . 151
4.7 Video 4: Obstacle Avoidance with Ultrasonic . . . 152
4.8 Video 5: Greyscale Line Tracking . . . 153
4.9 Video 6: Cliff Detection . . . 153

4.10 Video 7: PiCar-X Computer Vision . . . 154
4.11 Video 8: PiCar-X Stares at You . . . 155
4.12 Video 9: Recording Video . . . 156
4.13 Video 10: Bull Fight with PiCar-X . . . 156
4.14 Video 11: PiCar-X as Video Car . . . 157
4.15 Video 12: Treasure Hunt Game . . . 158
4.16 Video 12: Control PiCAR-X Robot Car use mobile app . . . 159

5 Play with Ezblock 161

5.1 Quick Guide on EzBlock . . . 161
5.2 Install and Configure EzBlock Studio . . . 165
5.3 Calibrate the Car . . . 166
5.4 Move . . . 170
5.5 Remote Control . . . 174
5.6 Test Ultrasonic Module . . . 176
5.7 Test Grayscale Module . . . 179
5.8 Color Detection . . . 180
5.9 Face Detection . . . 183
5.10 Sound Effect . . . 187
5.11 Background Music . . . 188
5.12 Say Hello . . . 190
5.13 Music Car . . . 193
5.14 Cliff Detection . . . 194
5.15 Minecart . . . 198
5.16 Minecart Plus . . . 202
5.17 Bullfight . . . 207
5.18 Beware of Pedestrians . . . 209
5.19 Traffic Sign Detection . . . 211
5.20 Orienteering . . . 215

6 Adjust Servo for Assembly 221

6.1 For Python Users . . . 222
6.2 For Ezblock Users . . . 222

7 Appendix 223

7.1 Filezilla Software . . . 223
7.2 PuTTY . . . 225
7.3 Install OpenSSH via Powershell . . . 227
7.4 Remote Desktop Access for Raspberry Pi . . . 229

8 Hardware 235

8.1 Robot HAT . . . 235
8.2 Camera Module . . . 236
8.3 Ultrasonic Module . . . 238
8.4 3-pin Battery . . . 240

9 FAQ 243

9.1 Q1: After installing Ezblock OS, the servo can’t turn to 0°? . . . 243
9.2 Q2: When using VNC, I am prompted that the desktop cannot be displayed at the moment? . . . 244
9.3 Q3: Why does the servo sometimes return to the middle position for no reason? . . . 244
9.4 Q4: About the Robot HAT Detailed Tutorial? . . . 244
9.5 Q5: About the Battery Charger? . . . 244
9.6 Q6: Camera Not Working? . . . 245

10 Copyright Notice 247

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

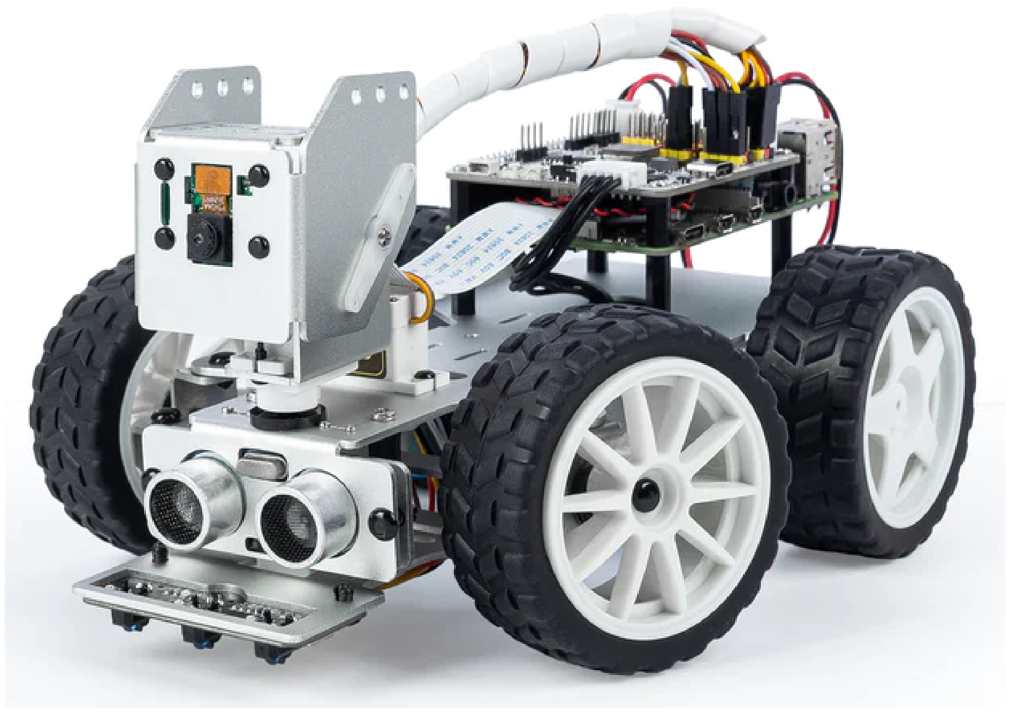
-

Thanks for choosing our .

Note: This document is available in the following languages.

-
-
-
-
-
-
-

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



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

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

CHAPTER

ONE

ASSEMBLE THE PICAR-X

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.

This video will walk you through the process of assembling your robot from scratch.

Note: The assembly steps in the video may differ slightly from the printed instructions you have. Please prioritize following the printed instructions. If any steps are unclear, you can refer to the video for further clarification.

In this tutorial, you will learn:

- **Preparation**: We'll introduce you to all the tools and parts needed, ensuring you're fully equipped before starting the assembly.
- **Assembly Steps**: We'll demonstrate each assembly step in a systematic manner.
- **Tips and Considerations**: Throughout the process, we'll share essential tips and tricks to help you avoid common mistakes and ensure your car operates smoothly.
- **Zeroing a Servo**: Before fixing each servo, it needs to be zeroed first. The steps for zeroing are to first install the Raspberry Pi OS, then install the required modules, and then run a script (set the angle of all PWM pins to 0). After that, plug in the servo wire to zero the servo.

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

CHAPTER

TWO

PLAY WITH PYTHON

For novices and beginners wishing to program in Python, some basic Python skills and familiarity with the Raspberry Pi OS are helpful. This section will guide you step by step — from setting up your Raspberry Pi, to moving PiCar-X, to using computer vision, and finally adding voice and AI interaction.

2.1 1. Quick Guide on Python

Learn how to set up your Raspberry Pi environment: install Raspberry Pi OS, configure Wi-Fi, and enable remote access so you can run Python code easily. If you already know how to use Raspberry Pi and access its command line, you may skip directly to the later parts.

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

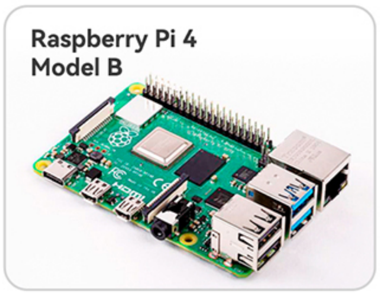
2.1.1 1. What Else Do You Need?

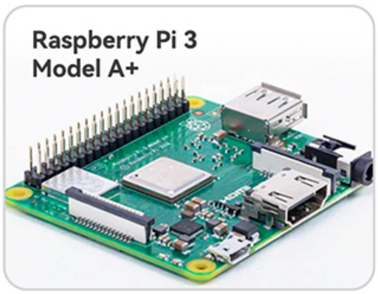
Before we start playing with PiCar-X, let's prepare the essential hardware. Think of these components as the **brain, heart, and senses** of PiCar-X — without them, the car cannot run properly.

Required Components

- **Raspberry Pi**

  The Raspberry Pi acts as the **brain** of PiCar-X, handling all computing, sensing, and control tasks.

  - **Compatible models**: Compatible with Raspberry Pi 5, 4, 3, and Raspberry Pi Zero 2 W (best on Pi 5 or Pi 4).
  - **Minimum**: **2GB RAM** — sufficient for all PiCar-X standard functions (movement, sensors, camera streaming) and for using **online AI services** such as OpenAI Whisper, TTS, or LLMs.
  - **Recommended**: **4GB RAM or more** — ensures smoother performance when running **local AI models** (e.g., Vosk speech recognition, Piper TTS, or lightweight LLMs) alongside camera streaming and control tasks.

- **Power Adapter**

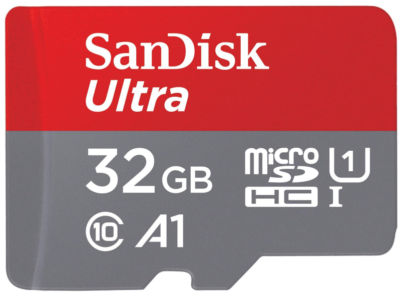
  PiCar-X comes with an **18650 battery pack** and a **Robot HAT** board featuring a built-in charging circuit.

Raspberry Pi
Official USB-C Power Supply
UK
USB-C
White
Get started:
raspberrypi.org/products/type-c-power-supply

- For charging, it is recommended to use a **5V 3A power supply**, such as the official **Raspberry Pi 15W USB-C adapter**.
- You may also use a **USB-C Power Delivery (PD) charger** or a **QC 2.0 fast charger**.
- A full charge typically takes about **2 hours** (from 0% to 100%).

- **Micro SD Card**

  The Raspberry Pi does not have a built-in hard drive. It boots and stores all files on a **Micro SD card**.



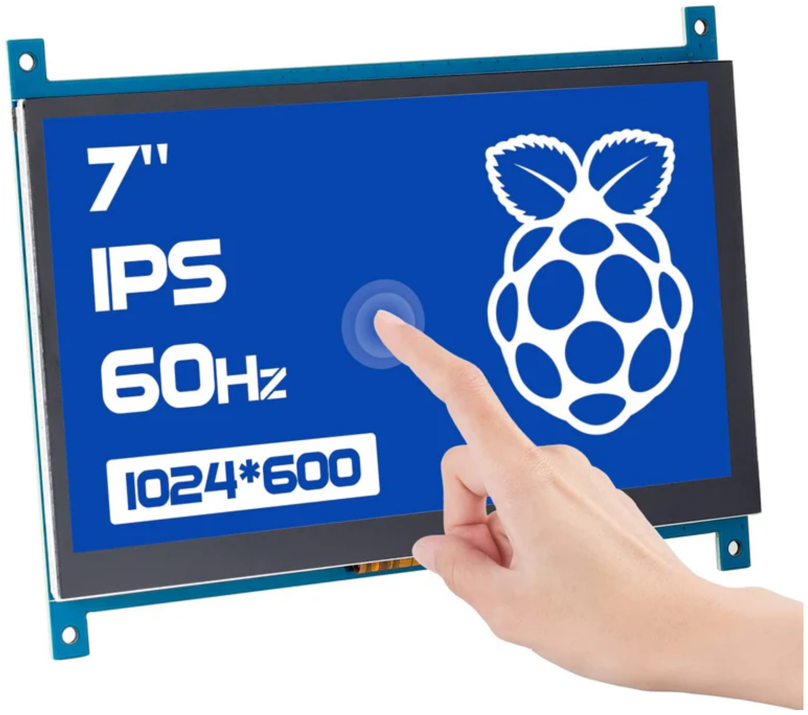
- Minimum: **16GB**
- Recommended: **32GB** for better stability
- Brand: Use reliable options such as **SanDisk** or **Samsung** to avoid read/write errors

Optional Components

Although not strictly required, the following peripherals will greatly improve your learning and debugging experience:

- **Monitor (HDMI or TV)**

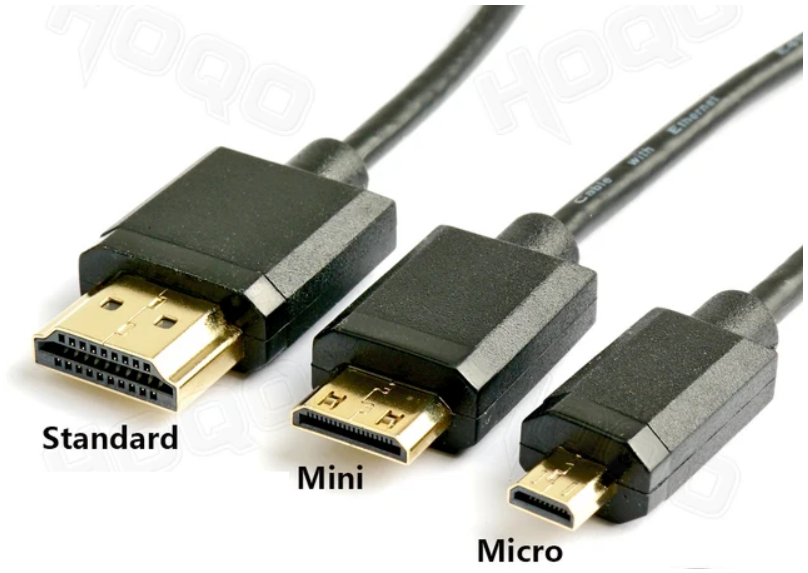
  For beginners, we strongly recommend a display with an HDMI input, so you can easily configure Raspberry Pi OS and run graphical programs.



- **HDMI Cable (Standard / Mini / Micro)**

  Different Raspberry Pi models use different HDMI connectors, be sure to check your Pi model and prepare the correct cable.

  - **Raspberry Pi 4 / 5**: Micro HDMI
  - **Raspberry Pi 3**: Standard HDMI
  - **Raspberry Pi Zero 2W**: Mini HDMI



- **Keyboard & Mouse**

  Very useful during the initial setup of Raspberry Pi OS. Later, you may switch to remote access (SSH/VNC), but for beginners we recommend preparing a basic USB or wireless set.

Tips for Preparation

- If you purchased the **PiCar-X kit**, most accessories are included, but you still need to prepare the Raspberry Pi board, Micro SD card, and power adapter separately.
- Not sure what to buy? The most stable and universal choice is: **Raspberry Pi 4 (2GB) + Official Power Supply + 32GB Micro SD card**.

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.

- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

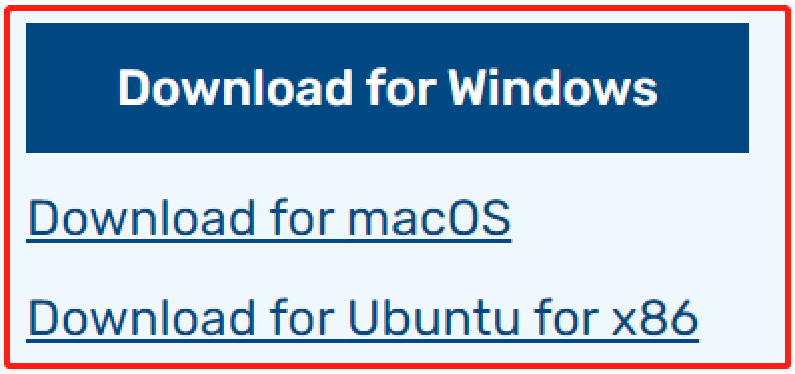
2.1.2 2. Installing the OS

Required Components

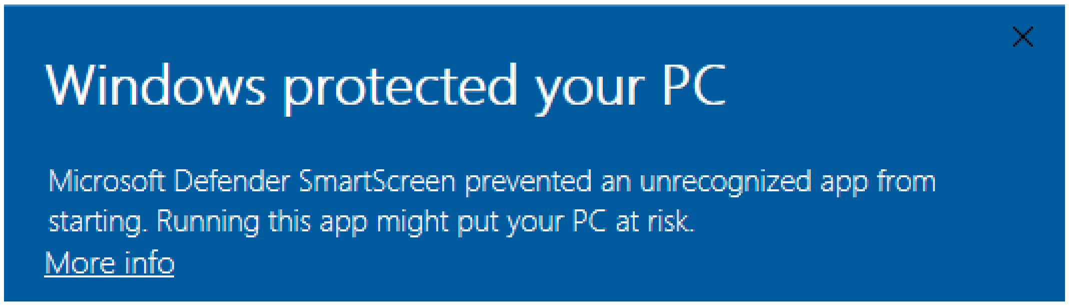
- A Personal Computer
- A Micro SD card and Reader

1. Install Raspberry Pi Imager

1. Visit the Raspberry Pi software download page at Raspberry Pi Imager. Choose the Imager version compatible with your operating system. Download and open the file to initiate installation.



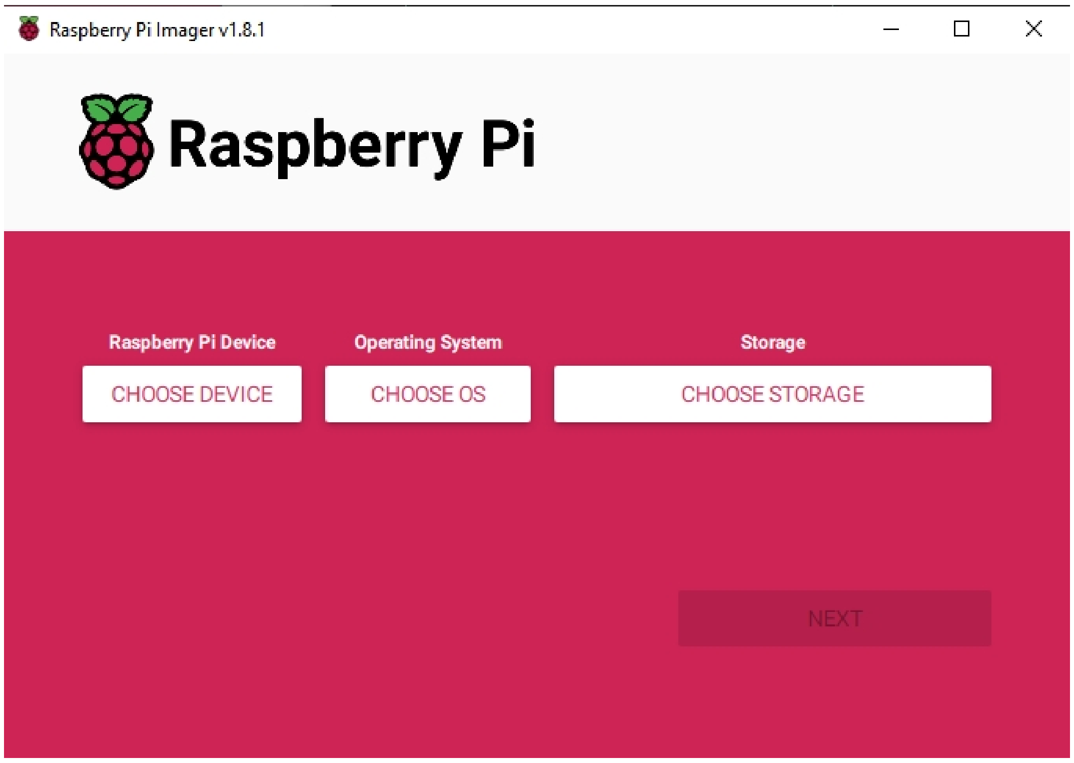
2. A security prompt may appear during installation, depending on your operating system. For example, Windows might display a warning message. In such cases, select **More info** and then **Run anyway**. Follow the on-screen guidance to complete the installation of the Raspberry Pi Imager.



Windows protected your PC

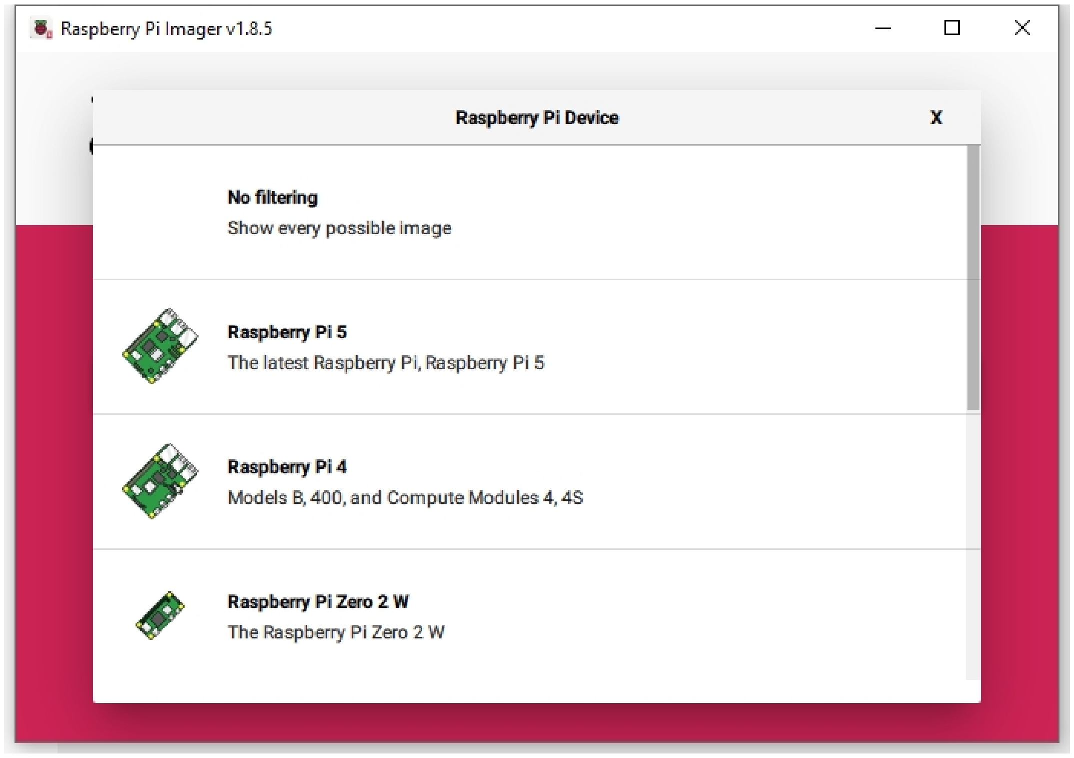
Microsoft Defender SmartScreen prevented an unrecognized app from starting. Running this app might put your PC at risk.

3. Launch the Raspberry Pi Imager application by clicking its icon or typing `rpi-imager` in your terminal.



2. Install OS to Micro SD Card

1. Insert your SD card into your computer or laptop using a Reader.
2. Within the Imager, click **Raspberry Pi Device** and select the Raspberry Pi model from the dropdown list.



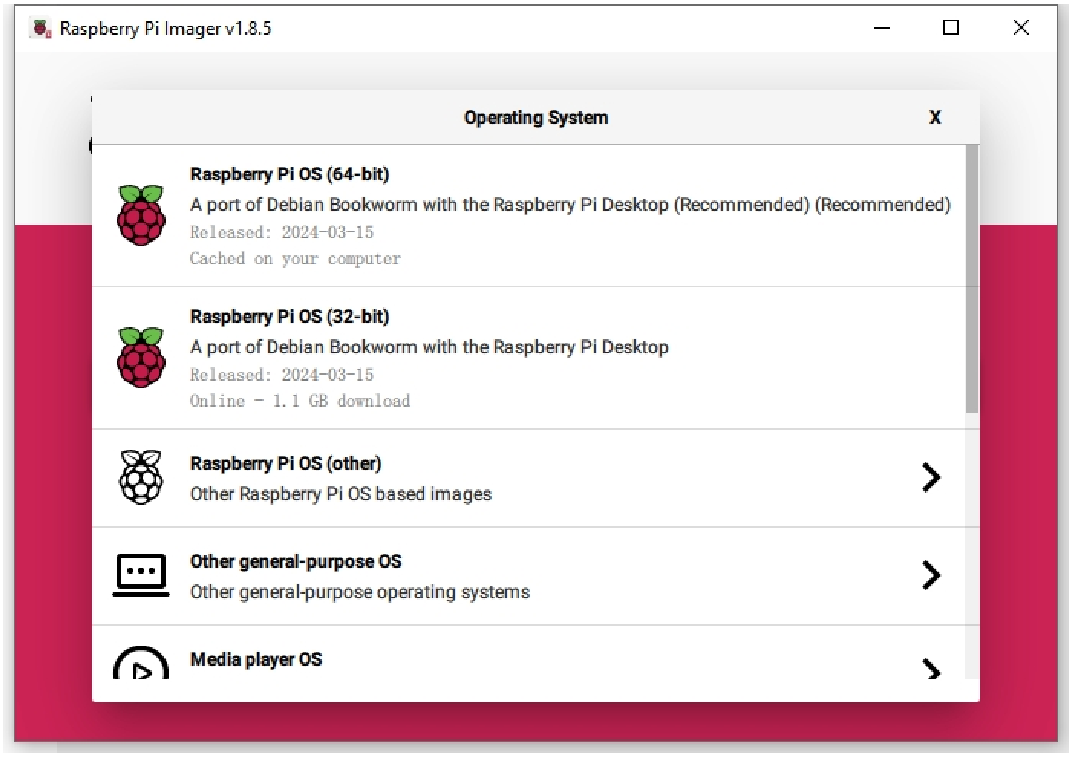
No filtering
Show every possible image

Raspberry Pi 5
The latest Raspberry Pi, Raspberry Pi 5

Raspberry Pi 4
Models B, 400, and Compute Modules 4, 4S

Raspberry Pi Zero 2 W
The Raspberry Pi Zero 2 W

3. Select **Operating System** and opt for the recommended operating system version.



Raspberry Pi OS (64-bit)
A port of Debian Bookworm with the Raspberry Pi Desktop (Recommended) (Recommended)
Released: 2024-03-15
Cached on your computer

Raspberry Pi OS (32-bit)
A port of Debian Bookworm with the Raspberry Pi Desktop
Released: 2024-03-15
Online - 1.1 GB download

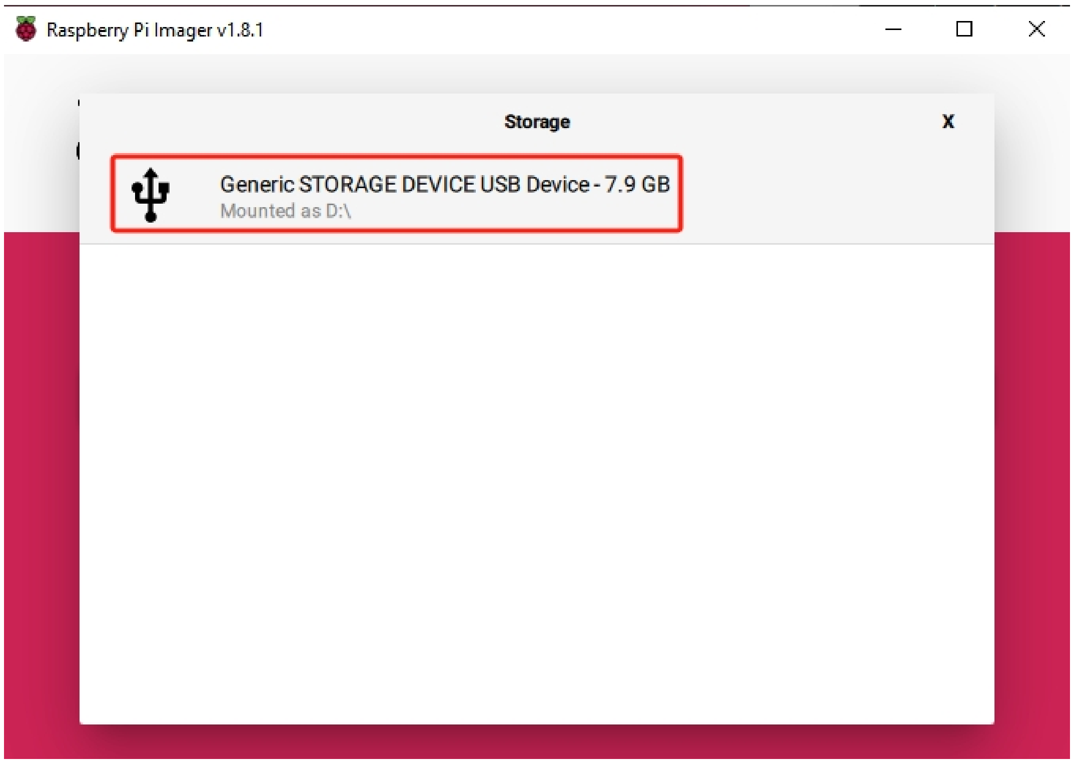
Raspberry Pi OS (other)
Other Raspberry Pi OS based images

Other general-purpose OS
Other general-purpose operating systems

Media player OS

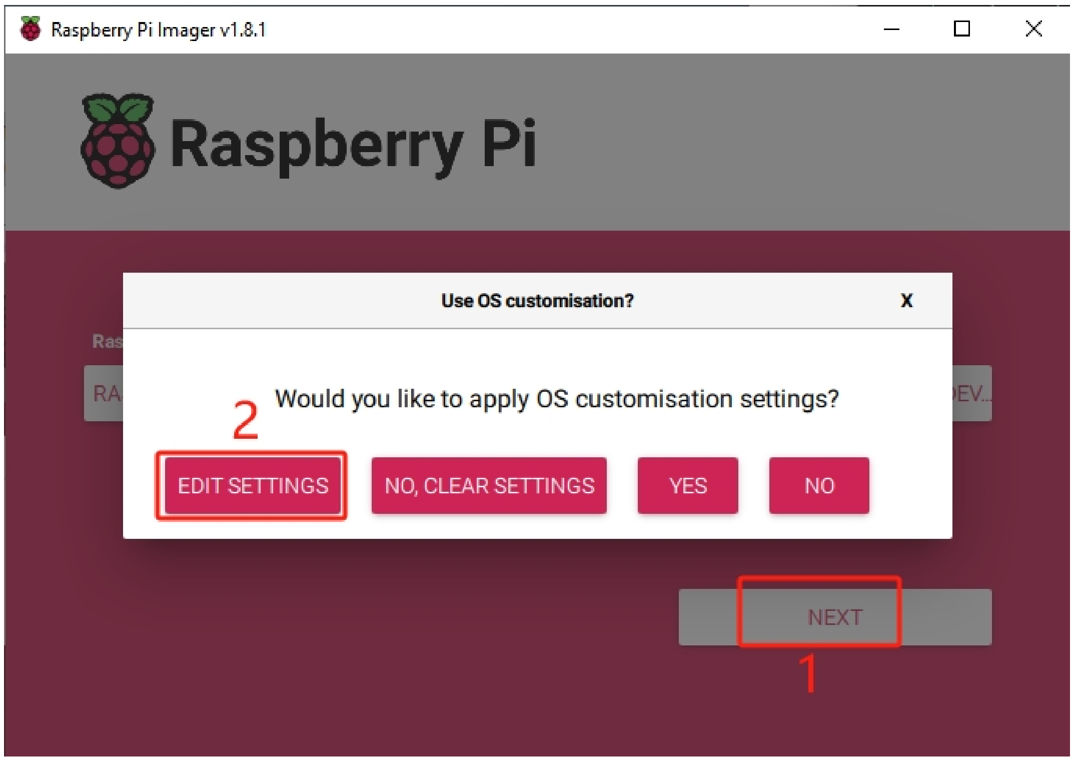
4. Click **Choose Storage** and select the appropriate storage device for the installation.

Note: Ensure you select the correct storage device. To avoid confusion, disconnect any additional storage devices if multiple ones are connected.


Raspberry Pi Imager v1.8.1
Storage
X
Generic STORAGE DEVICE USB Device - 7.9 GB
Mounted as D:\

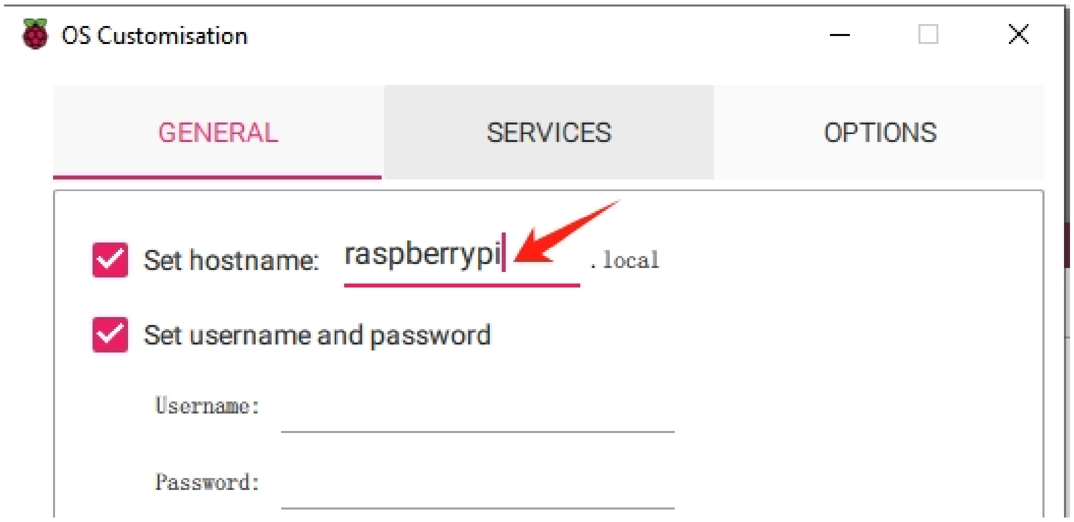
5. Click **NEXT** and then **EDIT SETTINGS** to tailor your OS settings.

Note: If you have a monitor for your Raspberry Pi, you can skip the next steps and click 'Yes' to begin the installation. Adjust other settings later on the monitor.


Raspberry Pi Imager v1.8.1
Raspberry Pi
Use OS customisation?
X
Would you like to apply OS customisation settings?
2
EDIT SETTINGS
NO, CLEAR SETTINGS
YES
NO
NEXT
1

6. Define a **hostname** for your Raspberry Pi.

   Note: The hostname is your Raspberry Pi's network identifier. You can access your Pi using `<hostname>.local` or `<hostname>.lan`.



OS Customisation

GENERAL | SERVICES | OPTIONS

☑ Set hostname: raspberrypi .local

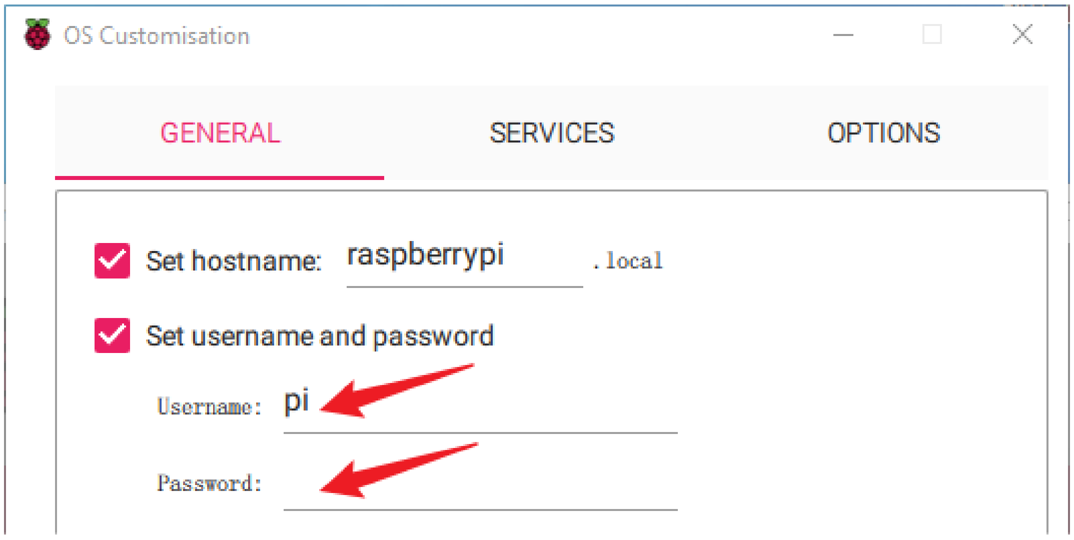
☑ Set username and password

Username:

Password:

7. Create a **Username** and **Password** for the Raspberry Pi's administrator account.

   Note: Establishing a unique username and password is vital for securing your Raspberry Pi, which lacks a default password.



OS Customisation

GENERAL | SERVICES | OPTIONS

☑ Set hostname: raspberrypi .local

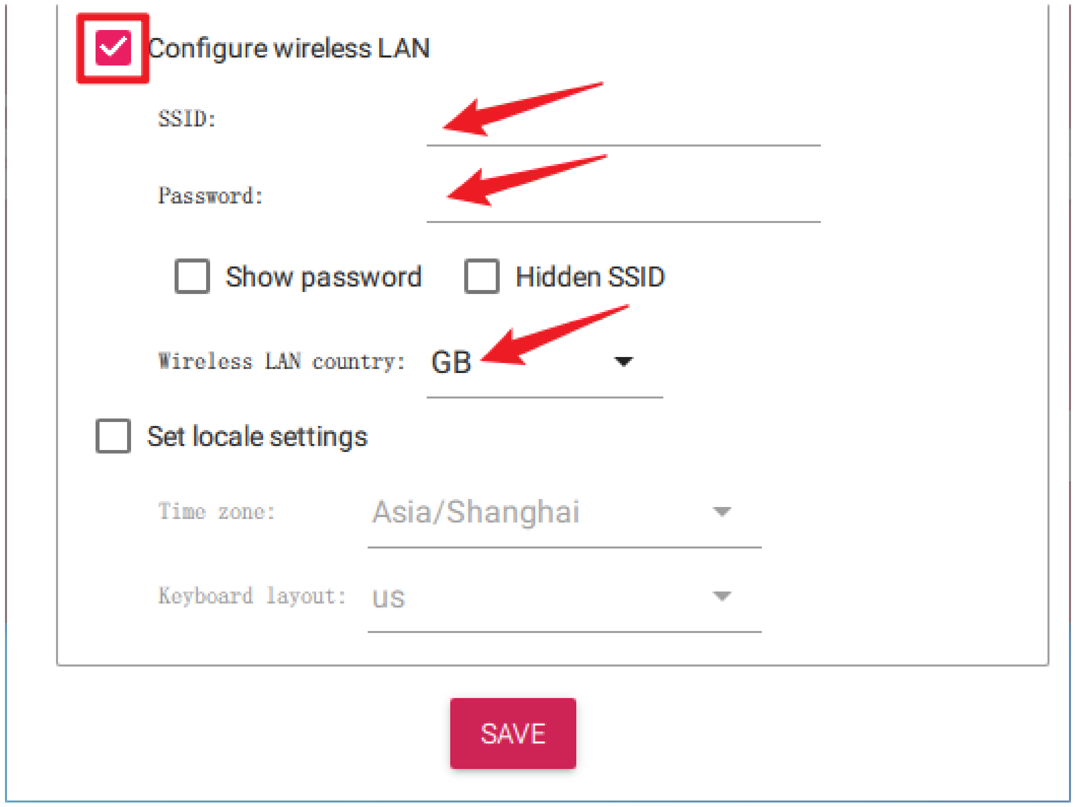
☑ Set username and password

Username: pi

Password:

8. Configure the wireless LAN by providing your network's **SSID** and **Password**.

   Note: Set the `Wireless LAN country` to the two-letter ISO/IEC alpha2 code corresponding to your location.


Configure wireless LAN

SSID:

Password:

Show password

Hidden SSID

Wireless LAN country: GB

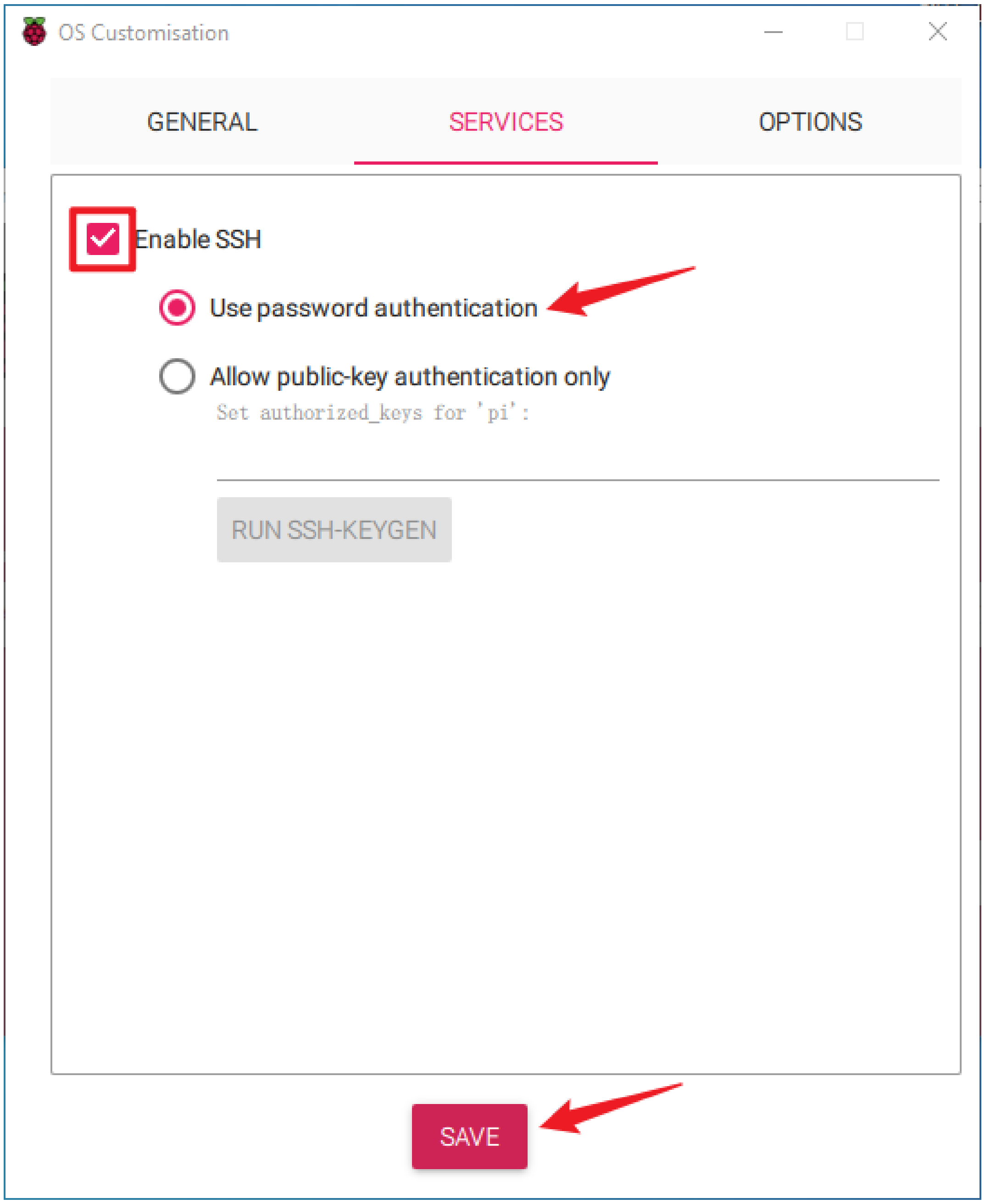
Set locale settings

Time zone: Asia/Shanghai

Keyboard layout: us

9. To remotely connect to your Raspberry Pi, enable SSH in the Services tab.

   - For **password authentication**, use the username and password from the General tab.
   - For public-key authentication, choose "Allow public-key authentication only". If you have an RSA key, it will be used. If not, click "Run SSH-keygen" to generate a new key pair.



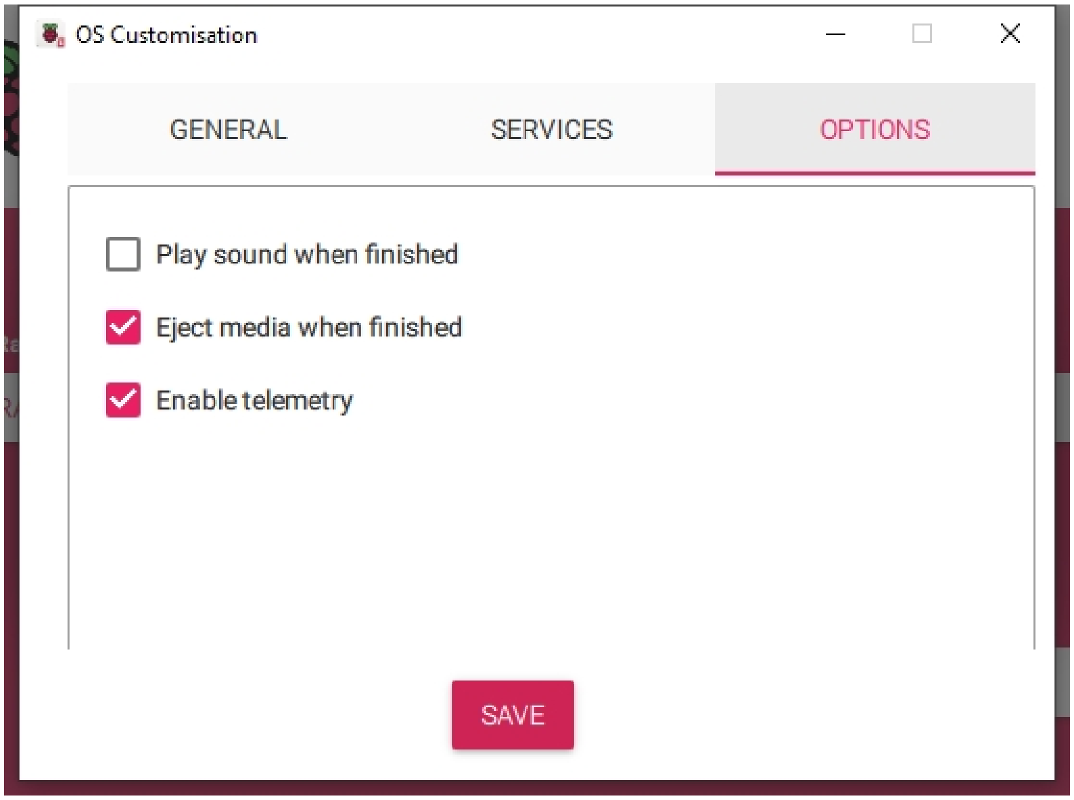
[✓] Enable SSH

[✓ Use password authentication]

Allow public-key authentication only

Set authorized_keys for 'pi':

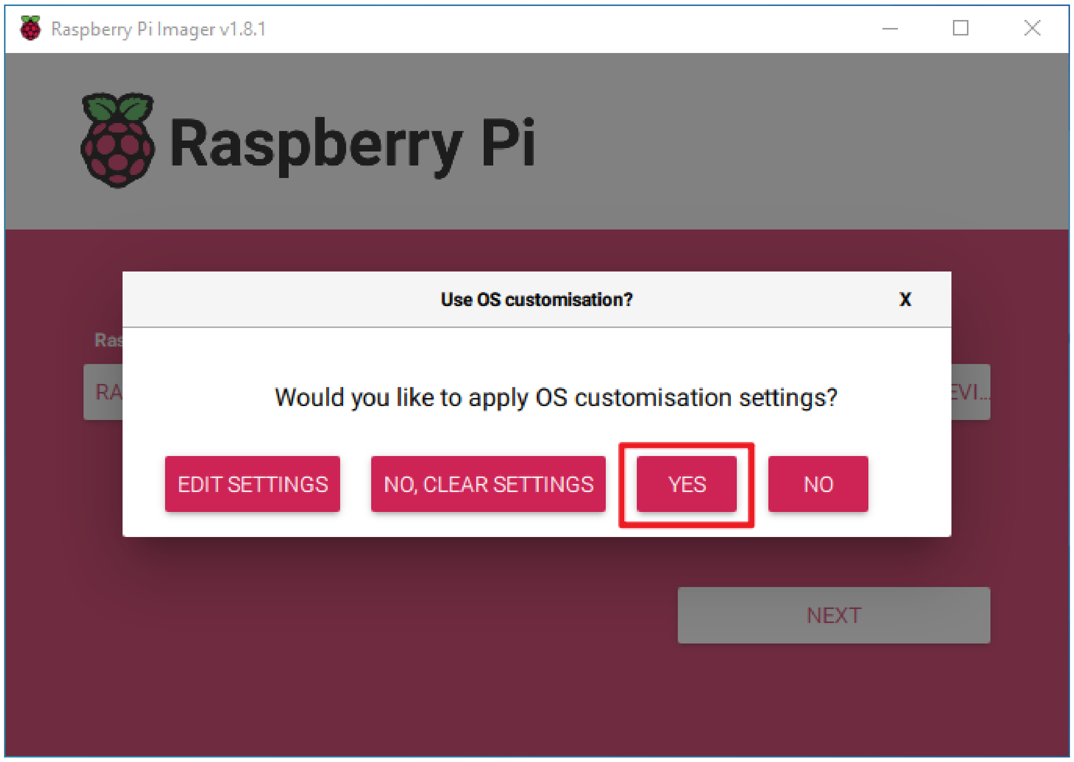
10. The **Options** menu lets you configure Imager's behavior during a write, including playing sound when finished, ejecting media when finished, and enabling telemetry.


OS Customisation

GENERAL SERVICES OPTIONS

- [ ] Play sound when finished
- [x] Eject media when finished
- [x] Enable telemetry

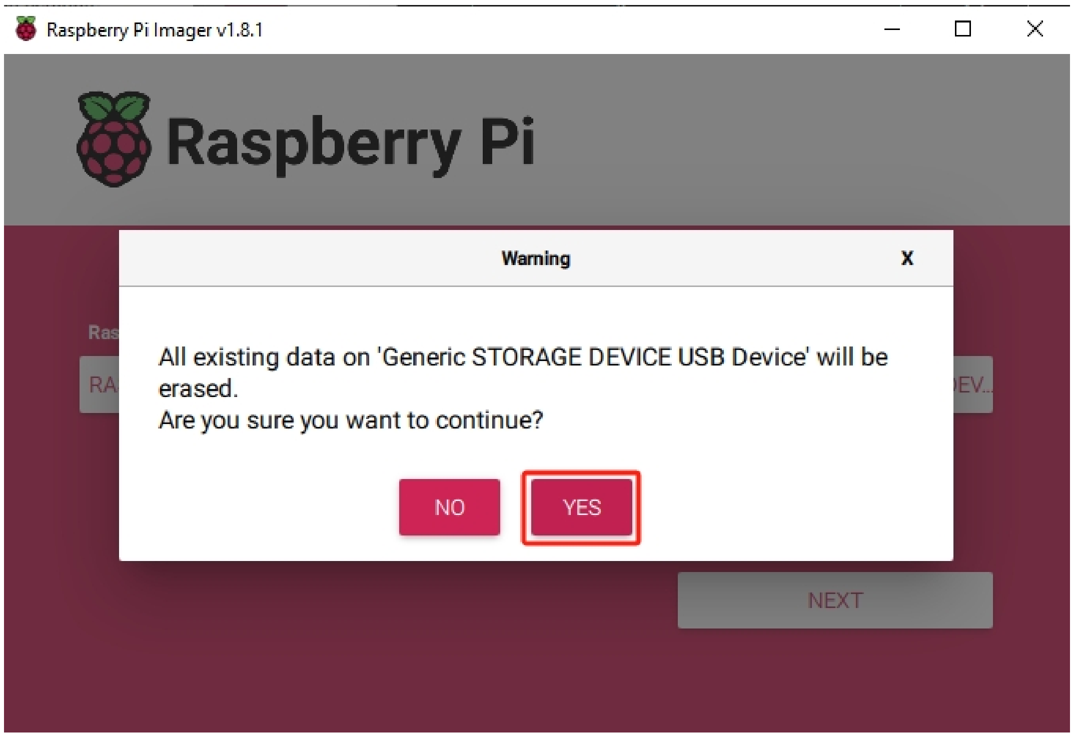
11. When you've finished entering OS customisation settings, click **Save** to save your customisation. Then, click **Yes** to apply them when writing the image.


Raspberry Pi Imager v1.8.1

Use OS customisation?

Would you like to apply OS customisation settings?

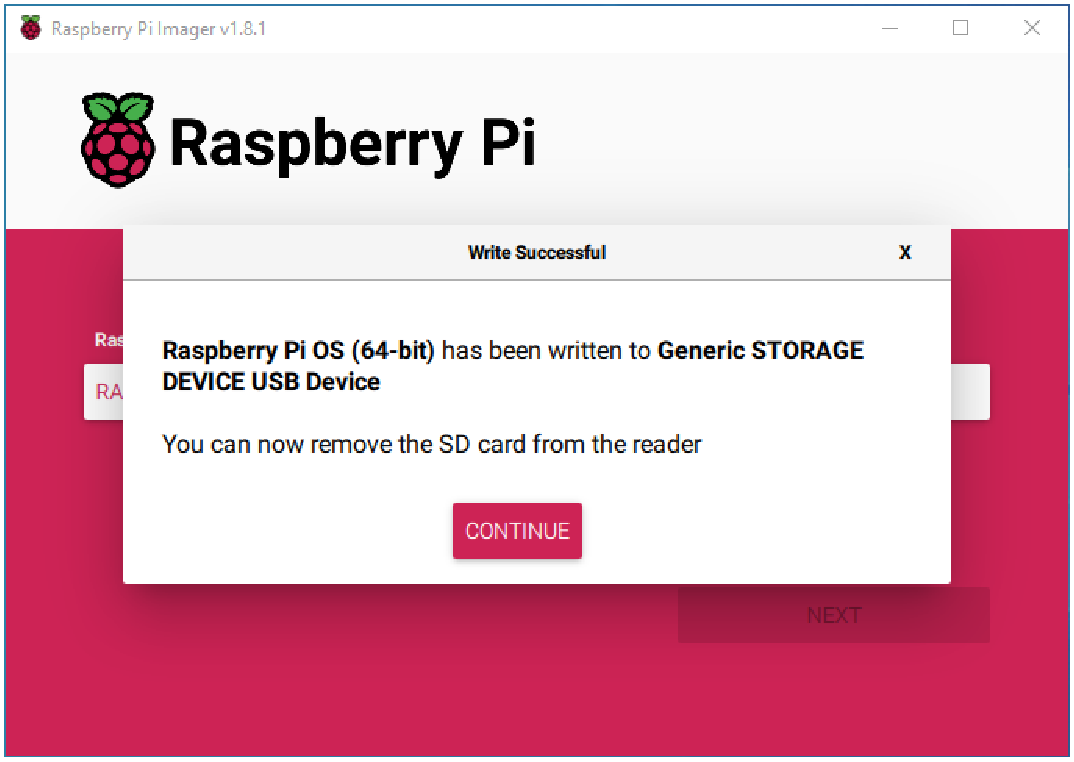
12. If the SD card contains existing data, ensure you back it up to prevent data loss. Proceed by clicking **Yes** if no backup is needed.


Raspberry Pi Imager v1.8.1

Warning

All existing data on 'Generic STORAGE DEVICE USB Device' will be erased.
Are you sure you want to continue?

13. When you see the “Write Successful” popup, your image has been completely written and verified. You’re now ready to boot a Raspberry Pi from the Micro SD Card!

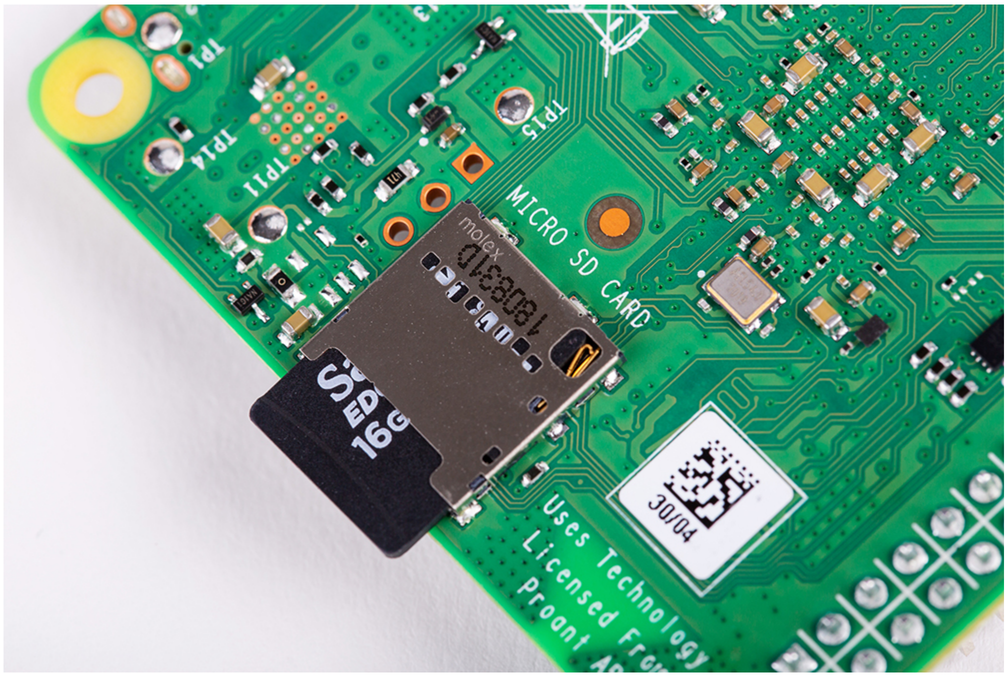

Raspberry Pi Imager v1.8.1

Write Successful

Raspberry Pi OS (64-bit) has been written to **Generic STORAGE DEVICE USB Device**

You can now remove the SD card from the reader

14. Now you can insert the SD card set up with Raspberry Pi OS into the microSD card slot located on the underside of the Raspberry Pi.



Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

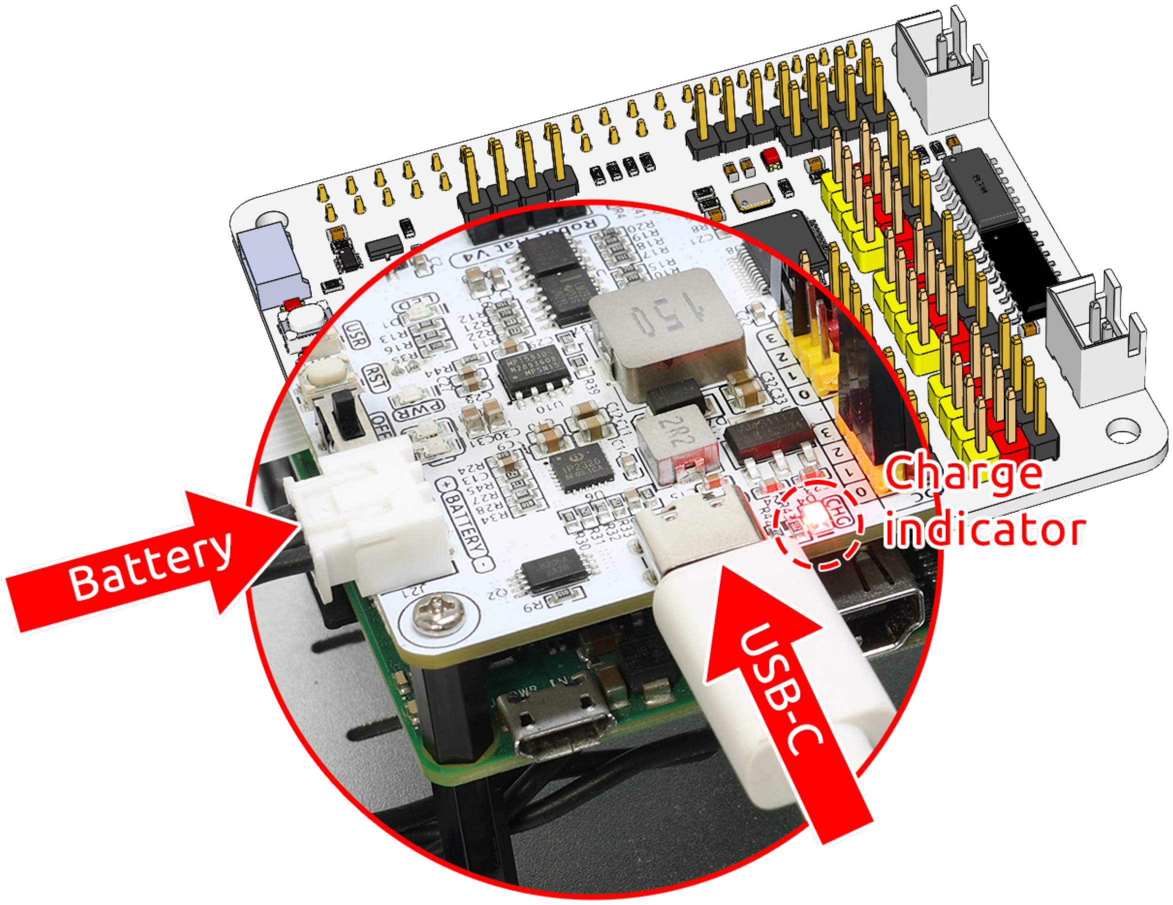
- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

2.1.3 3. Power Supply for Raspberry Pi (Important)

Charge

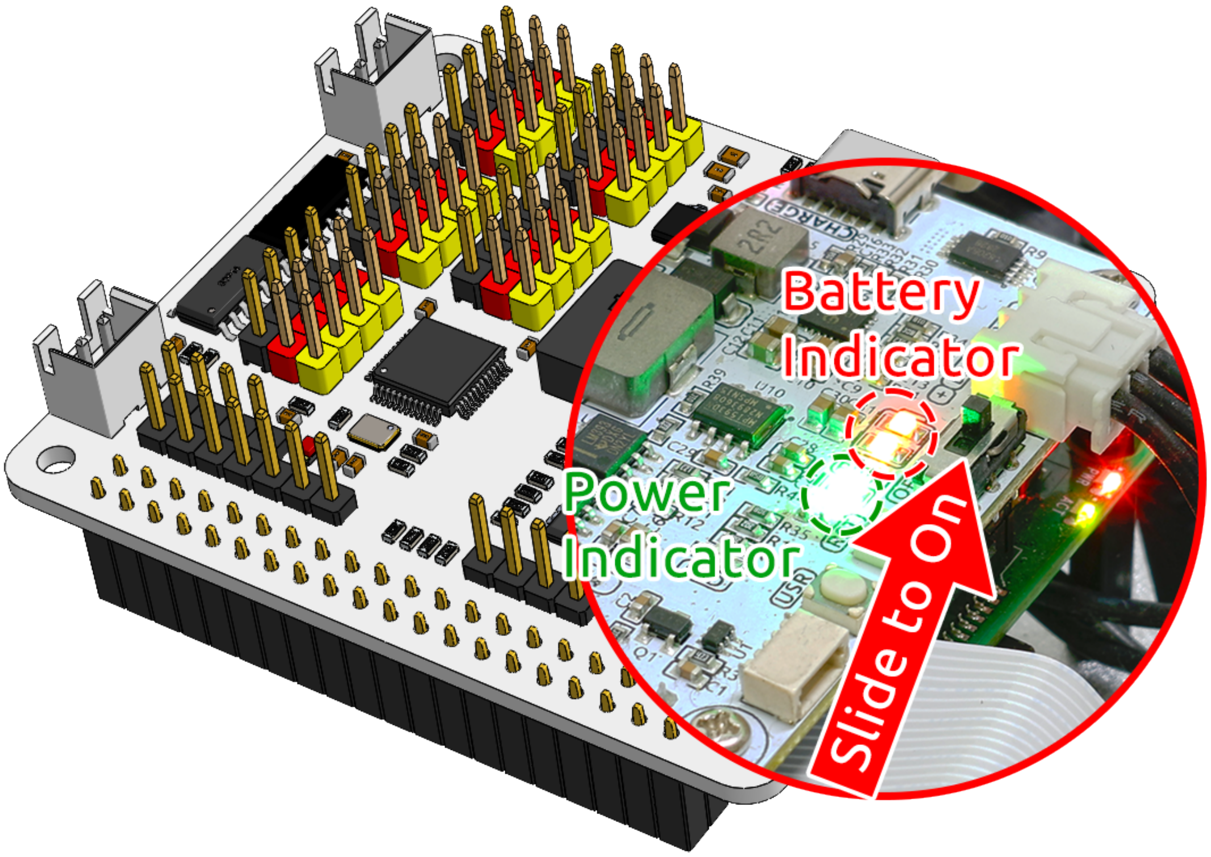
Insert the battery cable. Next, insert the USB-C cable to charge the battery. You will need to provide your own charger; we recommend a 5V 3A charger, or your commonly used smartphone charger will suffice.



Note: Connect an external Type-C power source to the Type-C port on the robot hat; it will immediately start charging the battery, and a red indicator light will illuminate.When the battery is fully charged, the red light will automatically turn off.

Power ON

Turn on the power switch. The Power indicator light and the battery level indicator light will illuminate.



Wait for a few seconds, and you will hear a slight beep, indicating that the Raspberry Pi has successfully booted.

Note: If both battery level indicator lights are off, please charge the battery. When you need extended programming or debugging sessions, you can keep the Raspberry Pi operational by inserting the USB-C cable to charge the battery simultaneously.

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

2.1.4 4. Set up Your Raspberry Pi

To start programming and controlling your PiCar-X, you first need to access your Raspberry Pi. This section will guide you through two common methods: using a monitor, keyboard, and mouse, or setting up a headless connection so you can log in remotely from another computer.

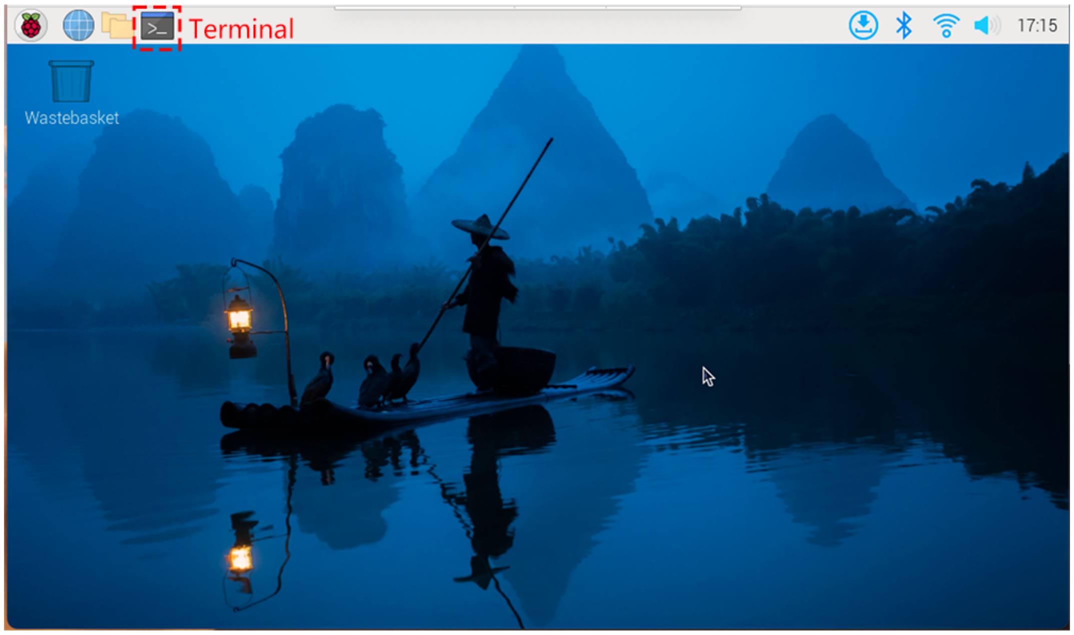
If You Have a Screen

Note: The Raspberry Pi Zero 2W installed on the Robot is not easy to connect to a screen. We recommend the **headless (no-screen)** method.

Required Components

- Raspberry Pi
- Power Adapter
- Micro SD card
- HDMI cable
- Screen
- Mouse
- Keyboard

1. Insert the microSD card into your Raspberry Pi.
2. Connect mouse, keyboard, and screen (for Pi 4/5 use **HDMI0**, the port nearest to the power input).
3. Power on the Raspberry Pi.
4. After a short while, Raspberry Pi OS desktop will appear and you can open a Terminal to enter commands.



If You Have No Screen (Headless Setup)

Without a monitor, you can configure and log into your Raspberry Pi remotely. This is the most convenient way to get started.

Required Components

- Raspberry Pi
- Power Adapter
- Micro SD card
- A computer on the same network

Tips

- Set the **Wireless LAN country** correctly using the ISO/IEC alpha-2 code (e.g., `US`, `UK`, `CN`); otherwise Wi-Fi will not work.
- Ensure that your Raspberry Pi and your computer are on the same local network.
- For a more reliable connection, use a direct network connection (Ethernet) whenever possible.

Connect via SSH

1. On your computer, open a terminal (Windows: **PowerShell**, macOS/Linux: **Terminal**) and type:

   ```
   ssh <username>@<hostname>.local
   # Example:
   ssh daisy@picarx.local
   ```

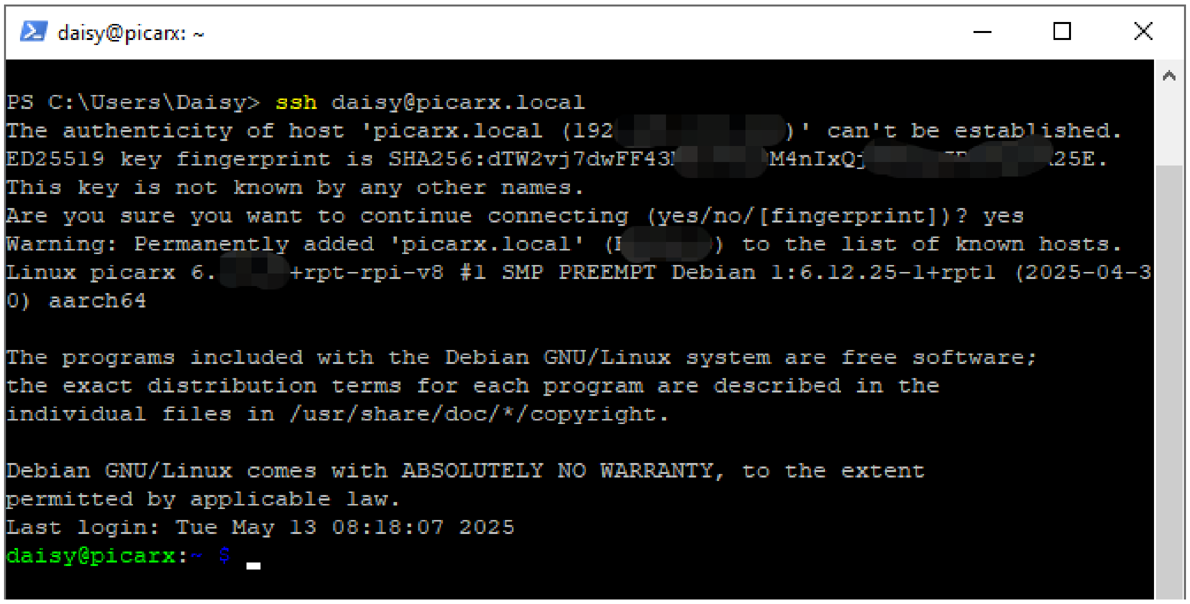
2. Alternatively, check your router's DHCP/client list, find the Pi's IP, and connect using:

   ```
   ssh <username>@<IP>

   # Example:

   ssh daisy@192.xxx.xx.xx
   ```

3. On first login, you will see a security prompt. Enter `yes` to proceed:
4. Enter the password you set in Raspberry Pi Imager. (Characters will not appear while typing; this is normal.)

   Note: The absence of visible characters when typing the password is a standard security feature. Just type carefully.

5. Once connected, your Raspberry Pi is ready for remote operations.



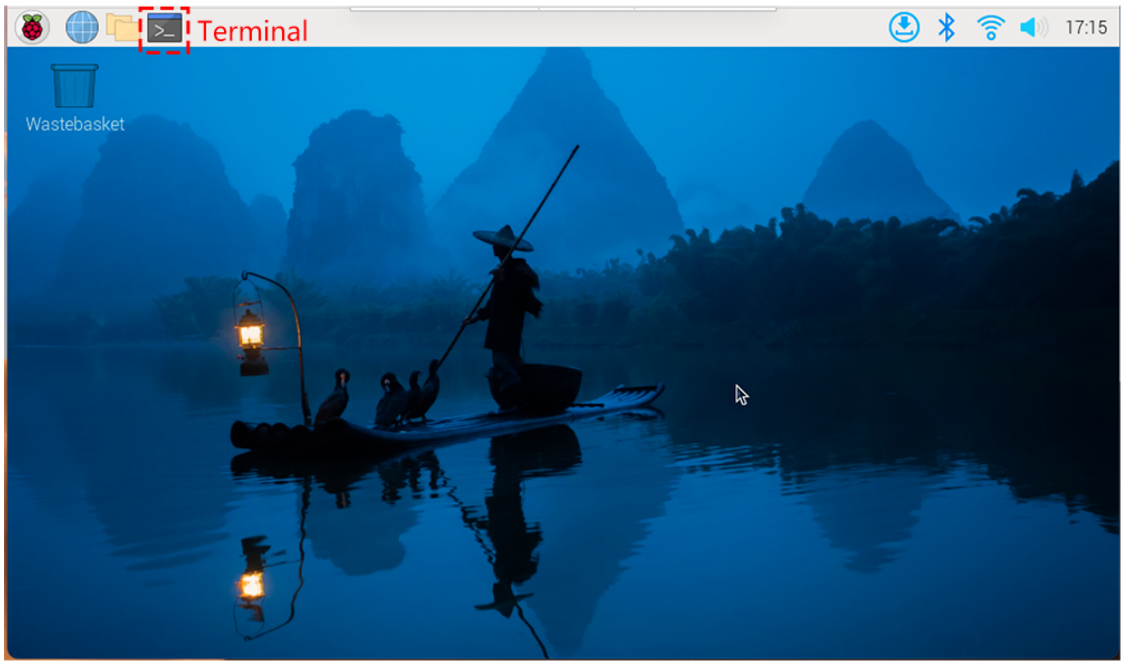
```
PS C:\Users\Daisy> ssh daisy@picarx.local
The authenticity of host 'picarx.local (192[redacted])' can't be established.
ED25519 key fingerprint is SHA256:dTW2vj7dwFF43[redacted]M4nIxQ[redacted]25E.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'picarx.local' ([redacted]) to the list of known hosts.
Linux picarx 6.[redacted]+rpt-rpi-v8 #1 SMP PREEMPT Debian 1:6.12.25-1+rpt1 (2025-04-3
0) 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: Tue May 13 08:18:07 2025
daisy@picarx:~ $
```

Troubleshooting

- **ssh: Could not resolve hostname ...**
  - Check that the hostname is correct.
  - If it still fails, use the Pi's IP address instead of `<hostname>.local`.
- **The term 'ssh' is not recognized... (Windows)**
  - Your system does not have OpenSSH installed. Install OpenSSH manually (see *Install OpenSSH via Powershell*), or use a third-party SSH client (see *PuTTY*).
- **Permission denied (publickey,password)**
  - Make sure you are using the username and password you configured in Raspberry Pi Imager.
- **Connection refused**
  - Wait 1–2 minutes after powering on.
  - Confirm that SSH was enabled in Raspberry Pi Imager.

Graphical Access Options

If you prefer a graphical interface instead of command line, you have two options:



- *Remote Desktop Access for Raspberry Pi*: Enable **VNC (Virtual Network Computing)** for a full desktop experience on your Pi.
- : Use **Raspberry Pi Connect** for secure remote access from anywhere, directly in a browser.

Now you cancontrol your Raspberry Pi without a monitor, either through SSH for command-line operations, or with VNC / Raspberry Pi Connect for a graphical desktop experience.

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

2.1.5 5. 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 2.5.x https://github.com/sunfounder/robot-hat.git --depth 1
cd robot-hat
sudo python3 install.py
```

Then download and install the `vilib` module.

```
cd ~/
git clone https://github.com/sunfounder/vilib.git --depth 1
cd vilib
sudo python3 install.py
```

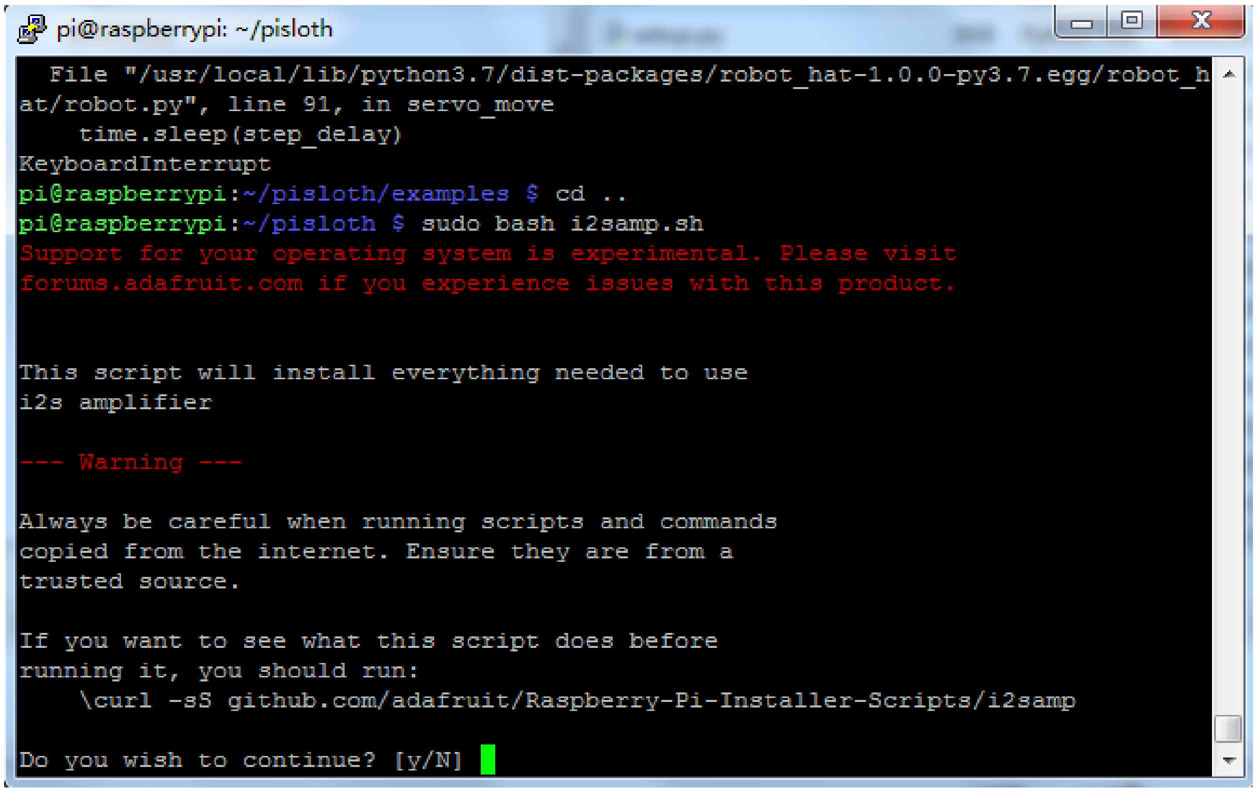
Download and install the `picar-x` module.

```
cd ~/
git clone -b 2.1.x https://github.com/sunfounder/picar-x.git --depth 1
cd picar-x
sudo pip3 install . --break
```

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

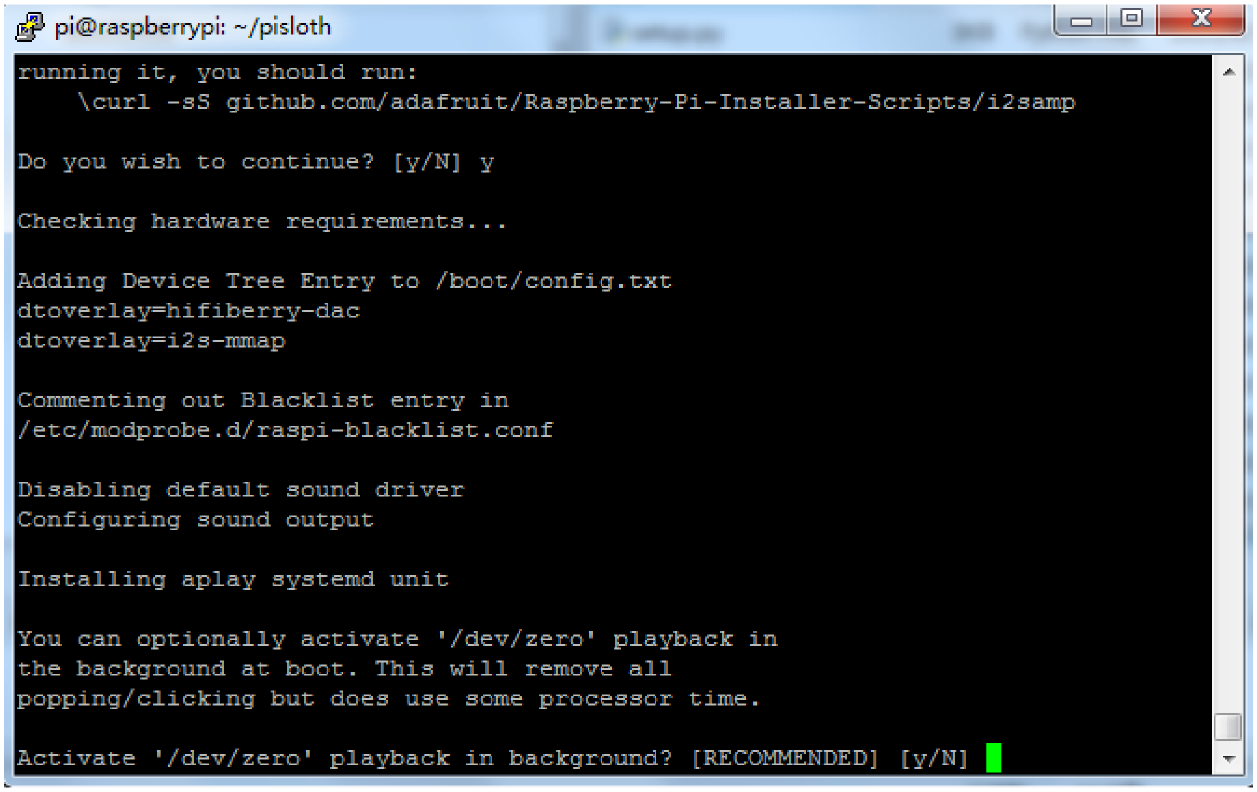
Finally, you need to run the script `i2samp.sh` to install the components required by the i2s amplifier, otherwise the picar-x will have no sound.

```
cd ~/robot-hat
sudo bash i2samp.sh
```



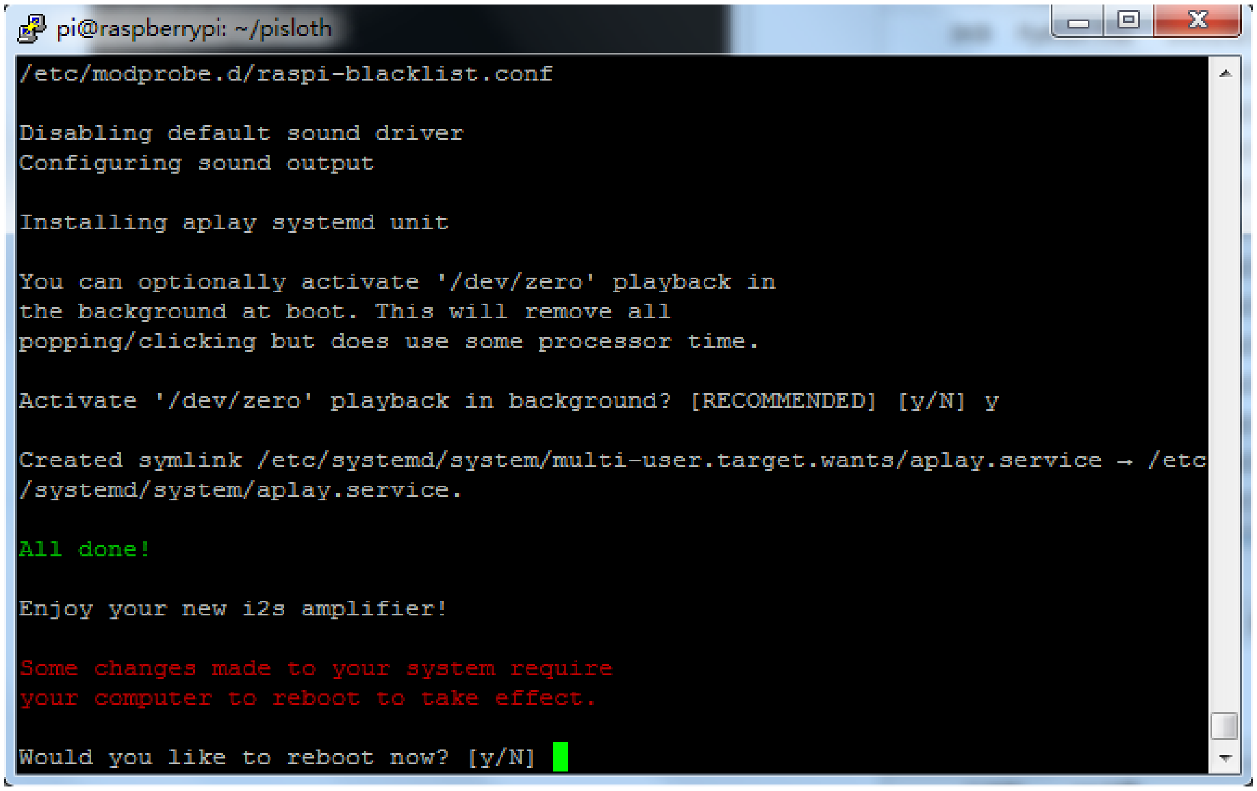
```
  File "/usr/local/lib/python3.7/dist-packages/robot_hat-1.0.0-py3.7.egg/robot_h
at/robot.py", line 91, in servo_move
    time.sleep(step_delay)
KeyboardInterrupt
pi@raspberrypi:~/pisloth/examples $ cd ..
pi@raspberrypi:~/pisloth $ sudo bash i2samp.sh
Support for your operating system is experimental.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


This script will install everything needed to use
i2s amplifier

--- Warning ---

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

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

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

Type y and press enter to continue running the script.



```
running it, you should run:
    \curl -sS github.com/adafruit/Raspberry-Pi-Installer-Scripts/i2samp

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

Checking hardware requirements...

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

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

Disabling default sound driver
Configuring sound output

Installing aplay systemd unit

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

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

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



```
/etc/modprobe.d/raspi-blacklist.conf

Disabling default sound driver
Configuring sound output

Installing aplay systemd unit

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

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

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

All done!

Enjoy your new i2s amplifier!

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

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

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

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

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

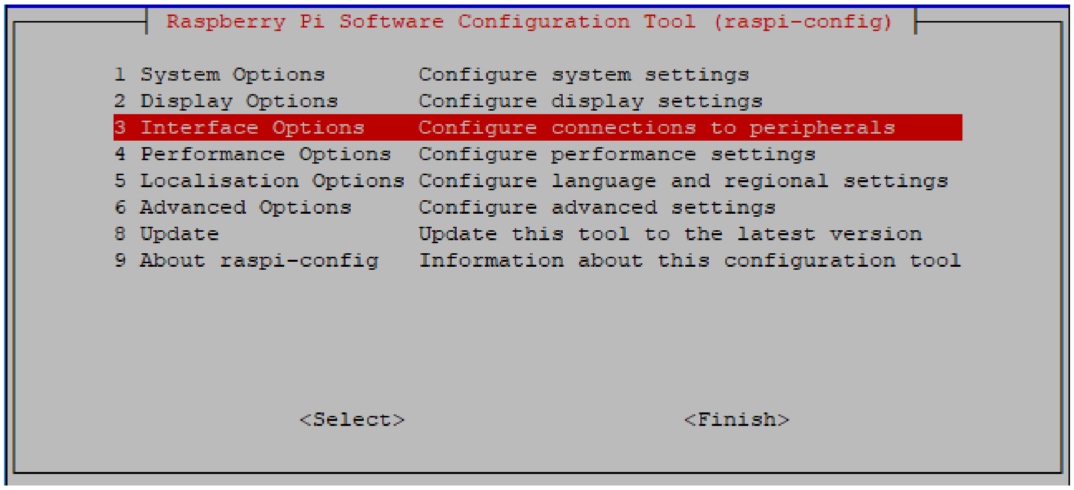
2.1.6 6. Enable I2C Interface(Important)

We will be using the Raspberry Pi's I2C interface. This interface should have been enabled when installing the `robot-hat` module earlier. To ensure everything is in order, let's check if it is indeed enabled.

1. Input the following command:

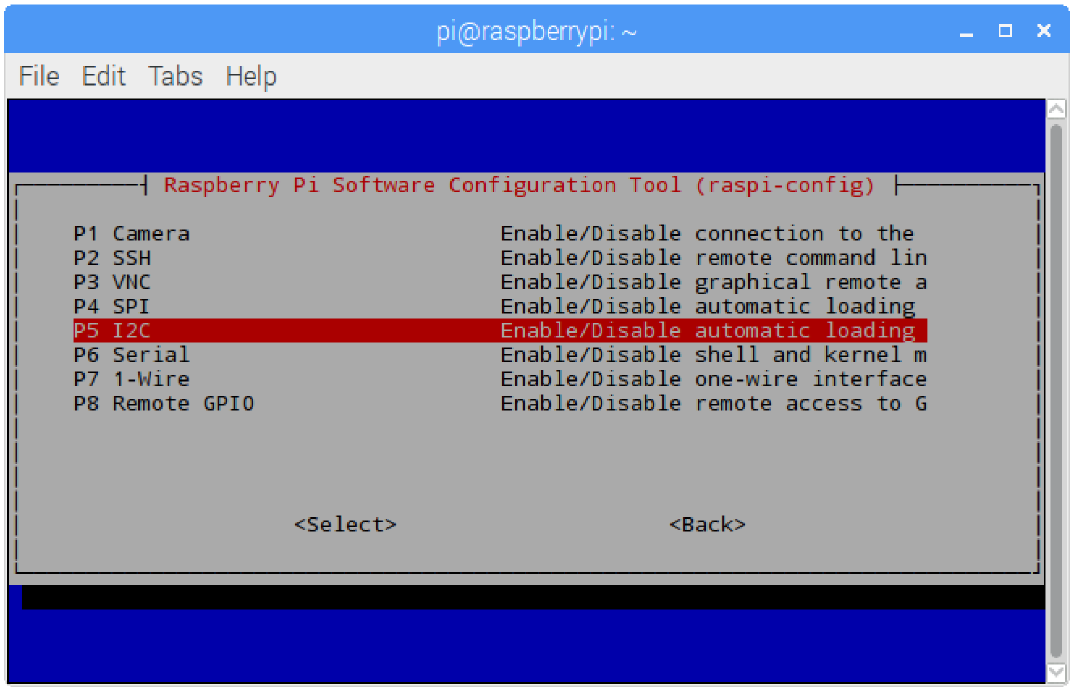
```
sudo raspi-config
```

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



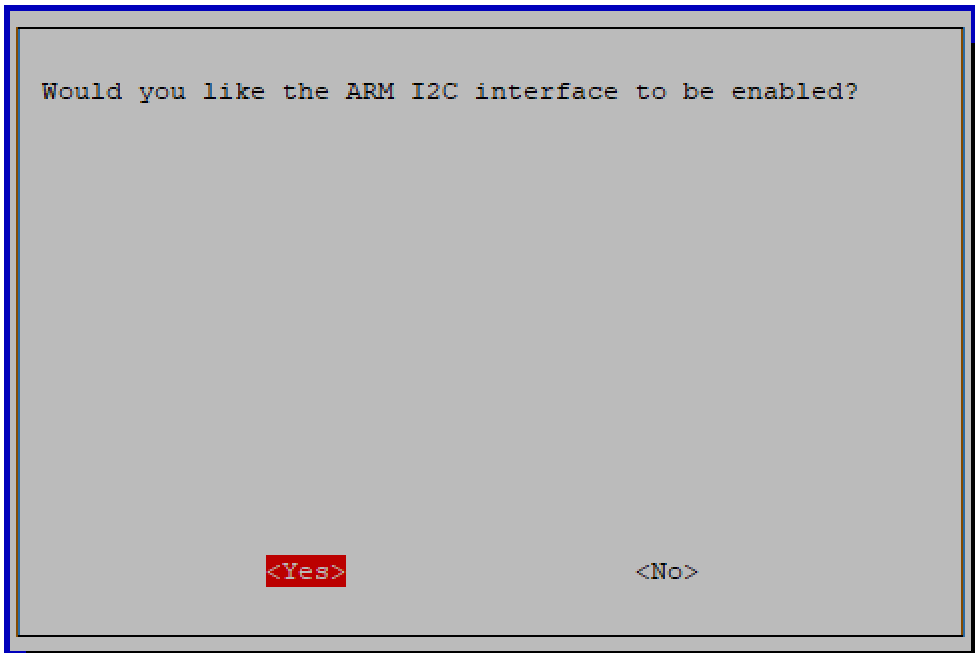
```
Raspberry Pi Software Configuration Tool (raspi-config)

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**.



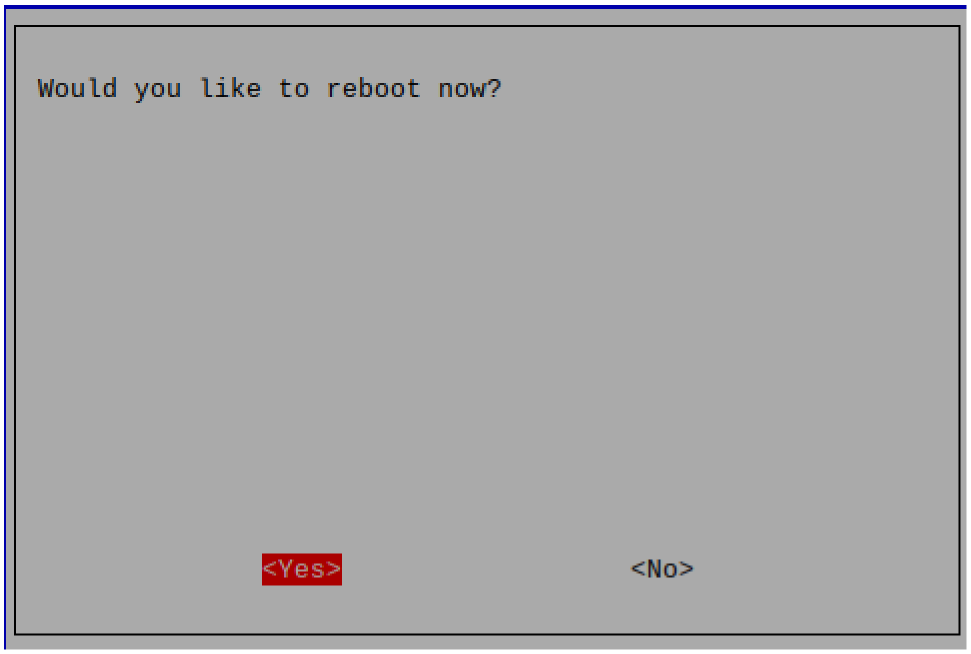
```
Raspberry Pi Software Configuration Tool (raspi-config)

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?

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?

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

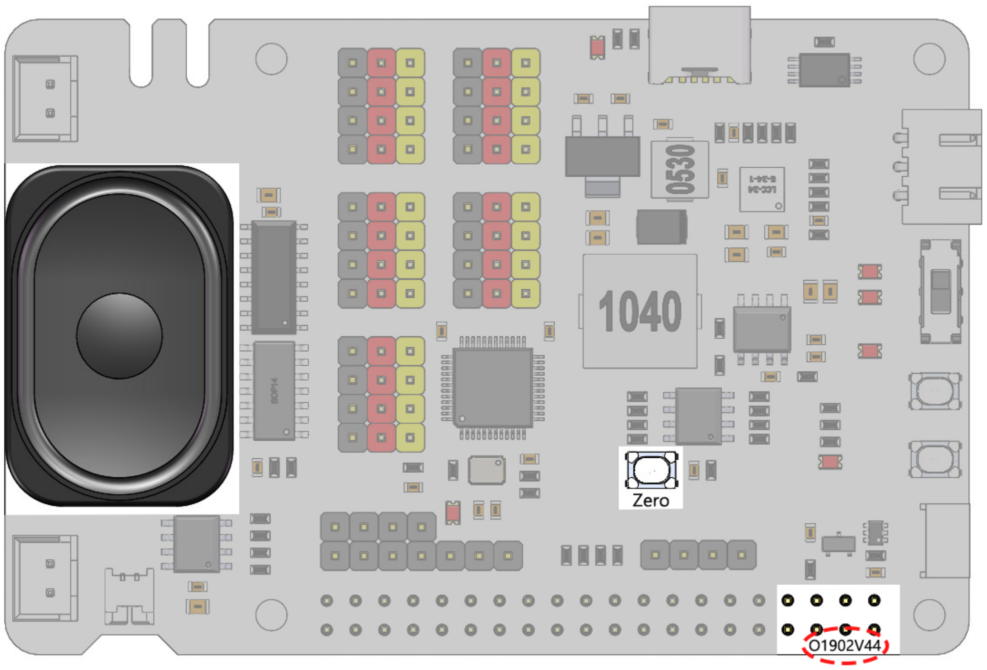
- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.

- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

2.1.7 7. Servo Adjust(Important)

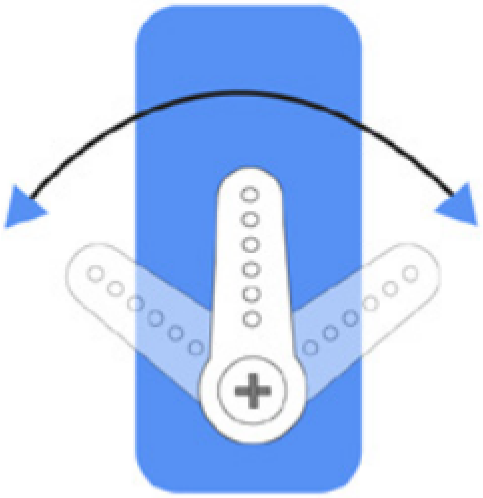
Note: If your Robot HAT is version V44 or higher (with the speaker located at the top of the board) and includes an onboard **Zero** button, you can skip this step and simply press the **Zero** button to activate the servo zeroing program.



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.

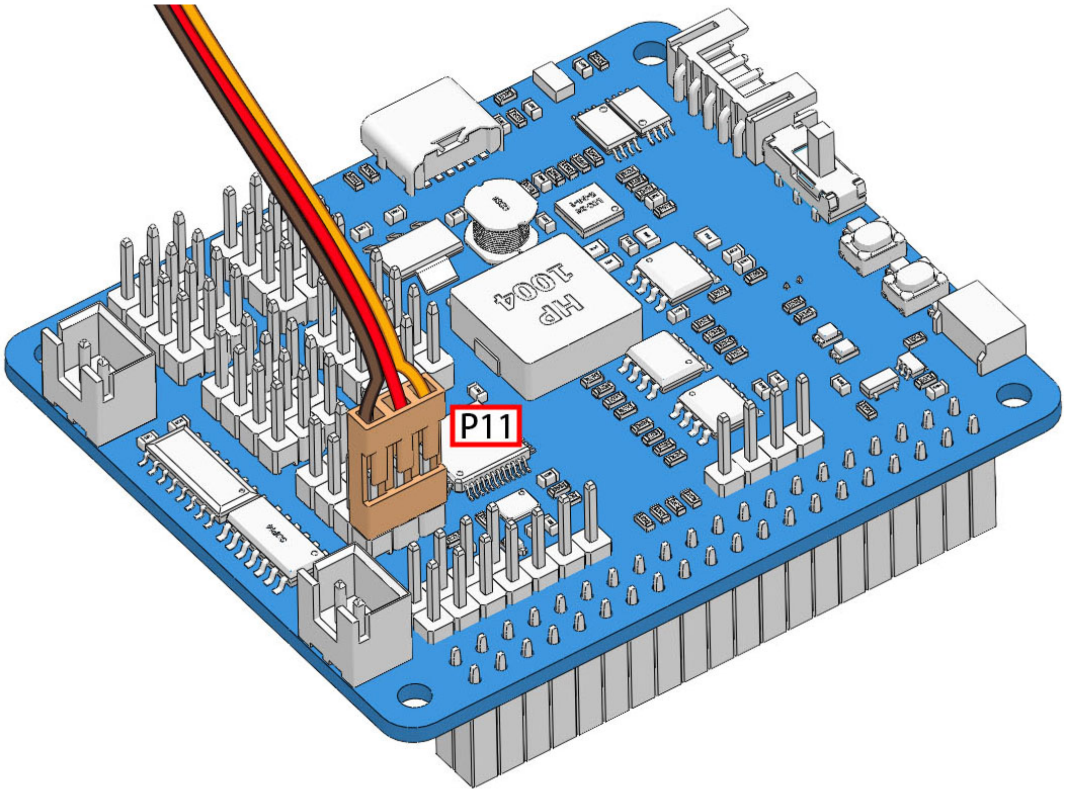
1. To ensure that the servo has been properly set to 0°, first insert the servo arm into the servo shaft and then gently rotate the rocker arm to a different angle. This servo arm is just to allow you to clearly see that the servo is rotating.



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°.

2.2 2. Basic Movement

After assembling your PiCar-X, start with simple movement programs. You will learn how to control the motors, drive forward/backward, turn, and use basic sensors for avoiding obstacles or following lines.

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

2.2.1 1. Calibrating the PiCar-X

Calibrate Motors & Servo

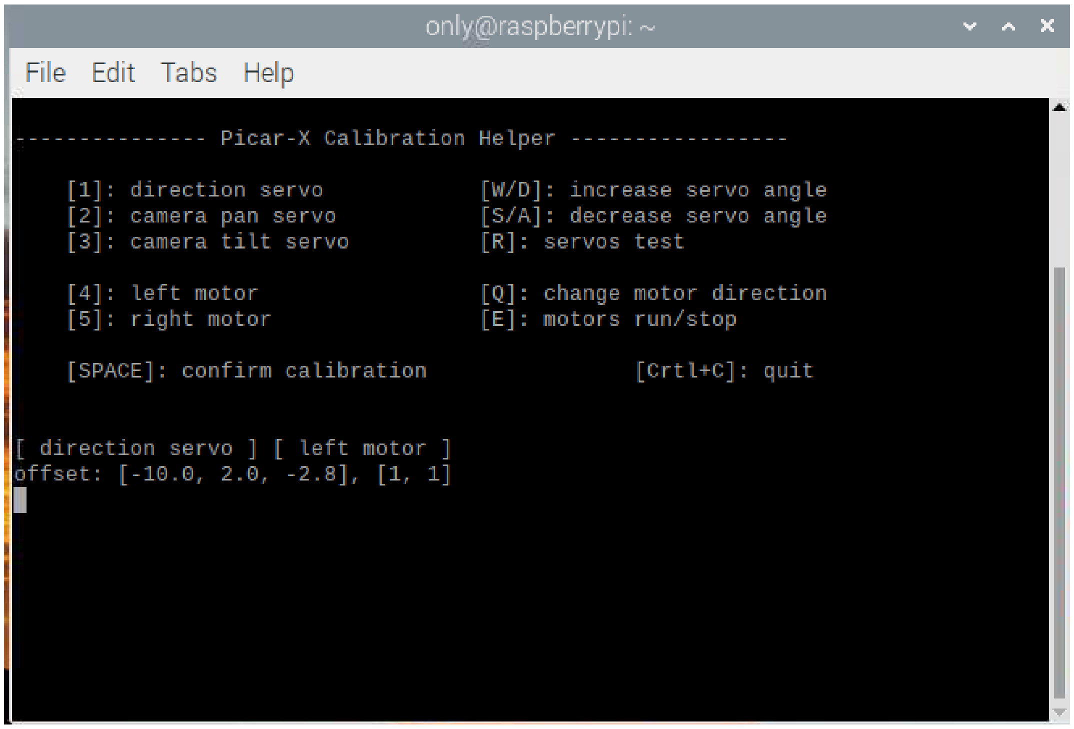
Some servo angles may be slightly tilted due to possible deviations during PiCar-X installation or limitations of the servos themselves, so you can calibrate them.

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

1. Run the `calibration.py`.

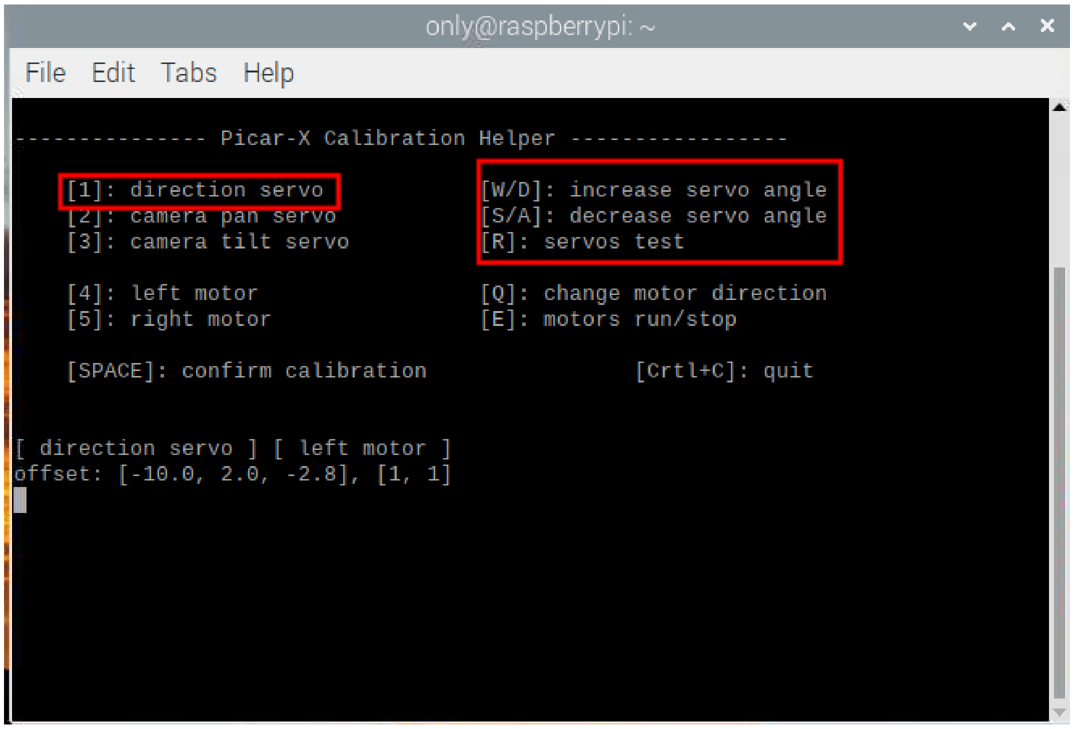
   ```
   cd ~/picar-x/example
   sudo python3 1.cali_servo_motor.py
   ```

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



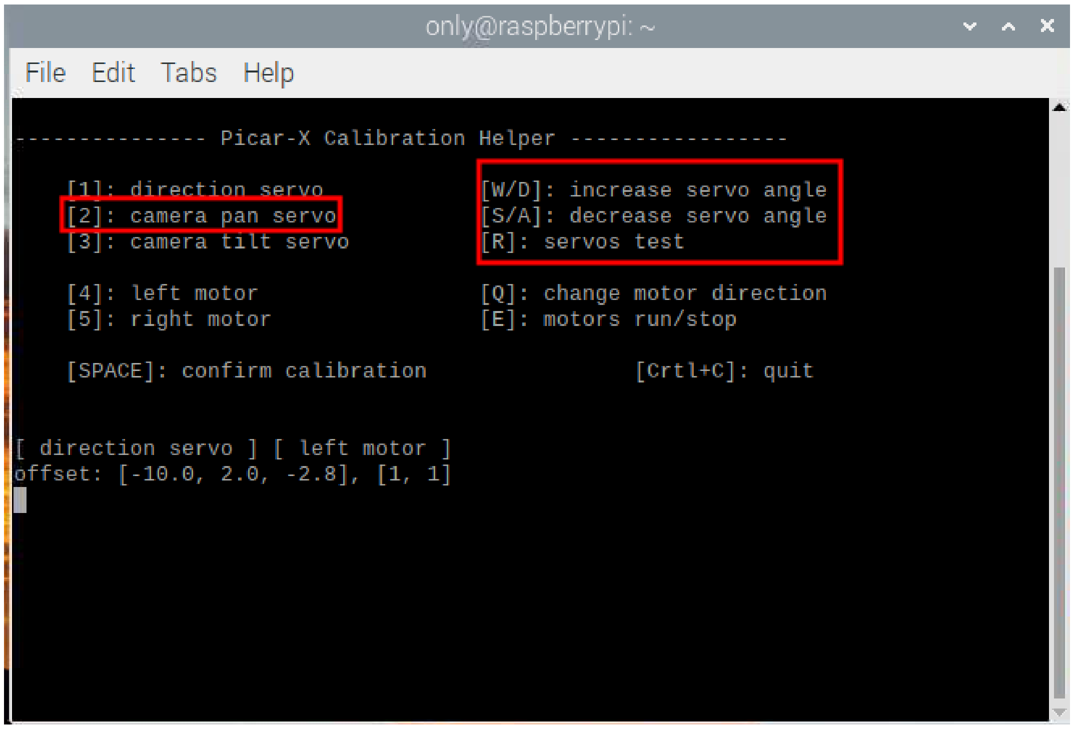
```
--------------- Picar-X Calibration Helper -----------------

    [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: [-10.0, 2.0, -2.8], [1, 1]
```

3. The R key is used to test if the 3 servos are working properly.
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.



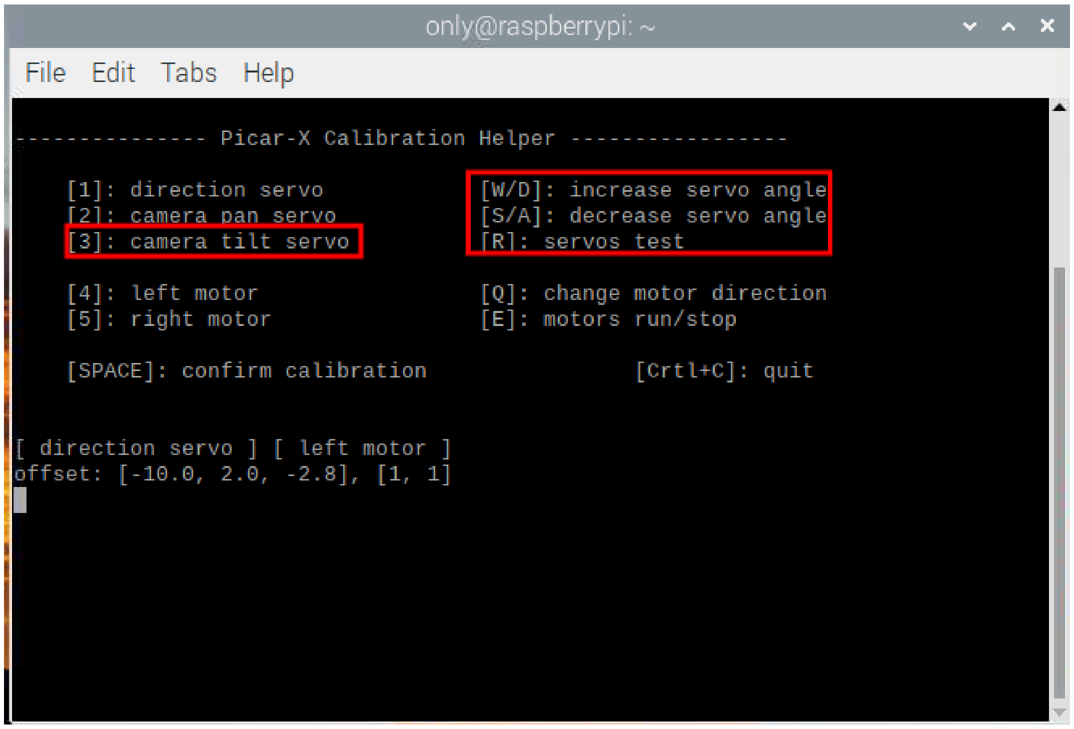
```
--------------- Picar-X Calibration Helper -----------------

    [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: [-10.0, 2.0, -2.8], [1, 1]
```

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.



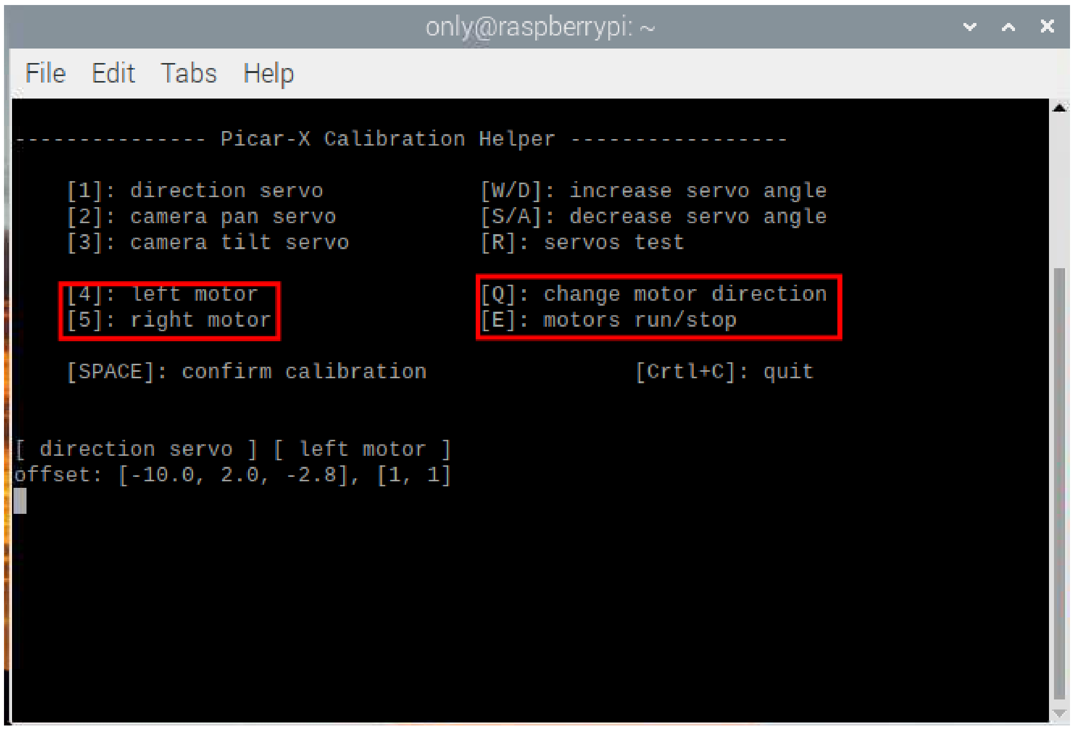
```
--------------- Picar-X Calibration Helper -----------------

    [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: [-10.0, 2.0, -2.8], [1, 1]
```

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.



```
--------------- 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: [-10.0, 2.0, -2.8], [1, 1]
```

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.

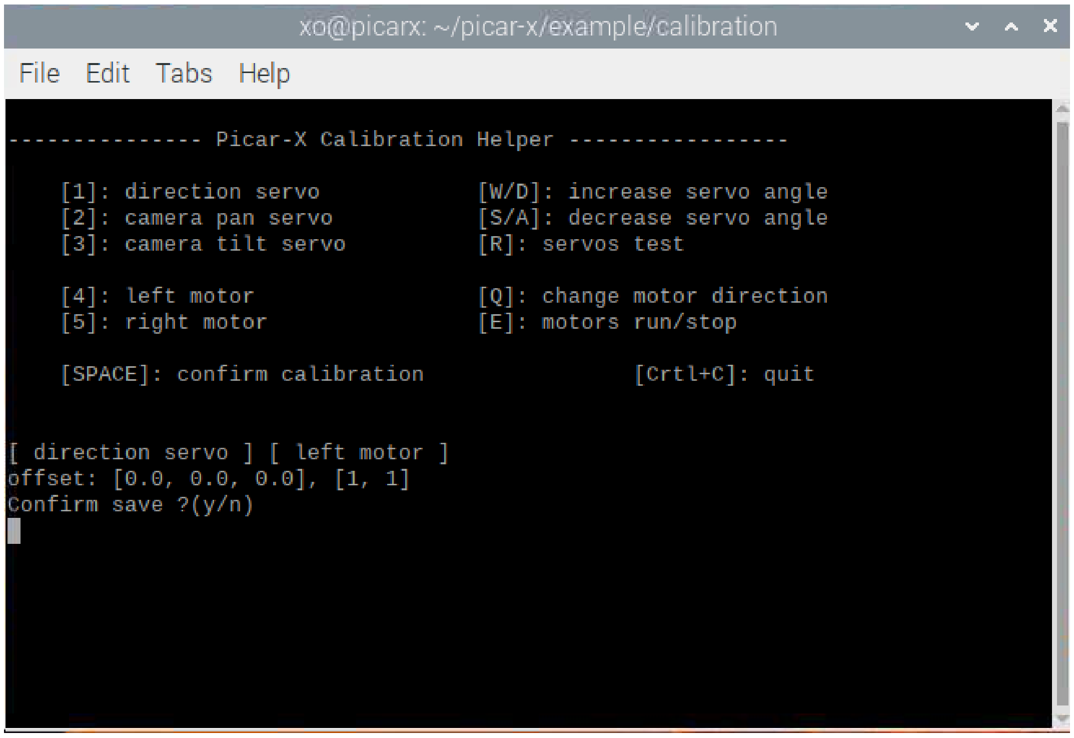


only@raspberrypi: ~

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: [-10.0, 2.0, -2.8], [1, 1]
```

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)
```

Calibrate Grayscale Module

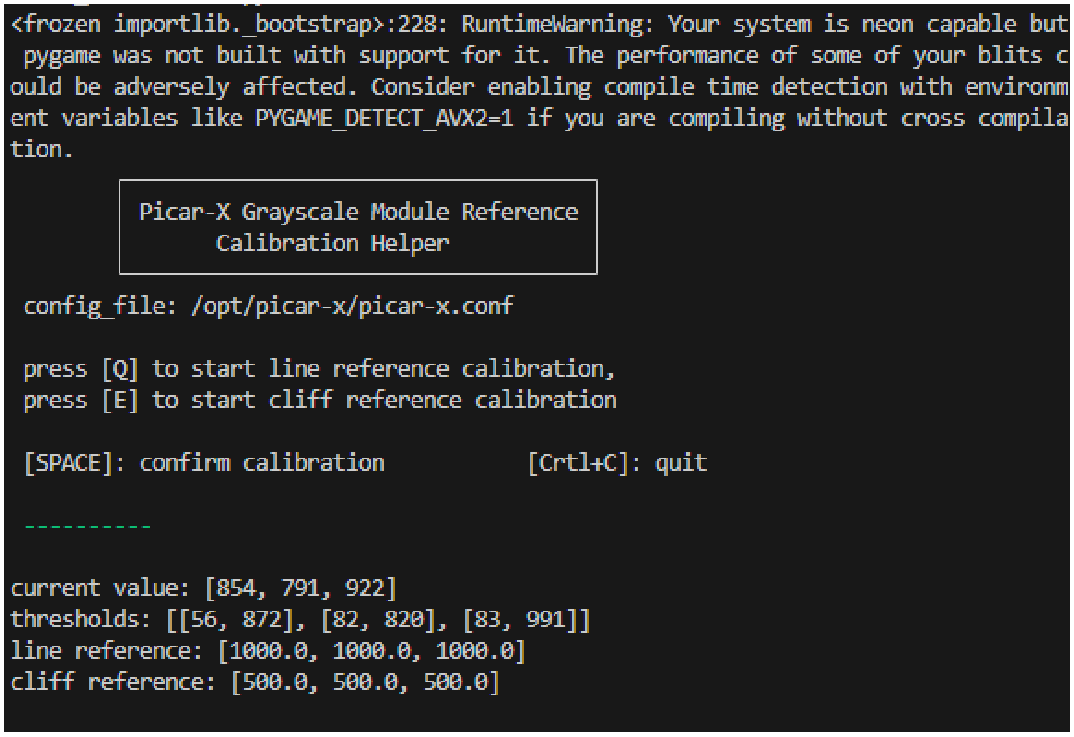
Due to varying environmental conditions and lighting situations, the preset parameters for the greyscale module might not be optimal. You can fine-tune these settings through this program to achieve better results.

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 code.

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

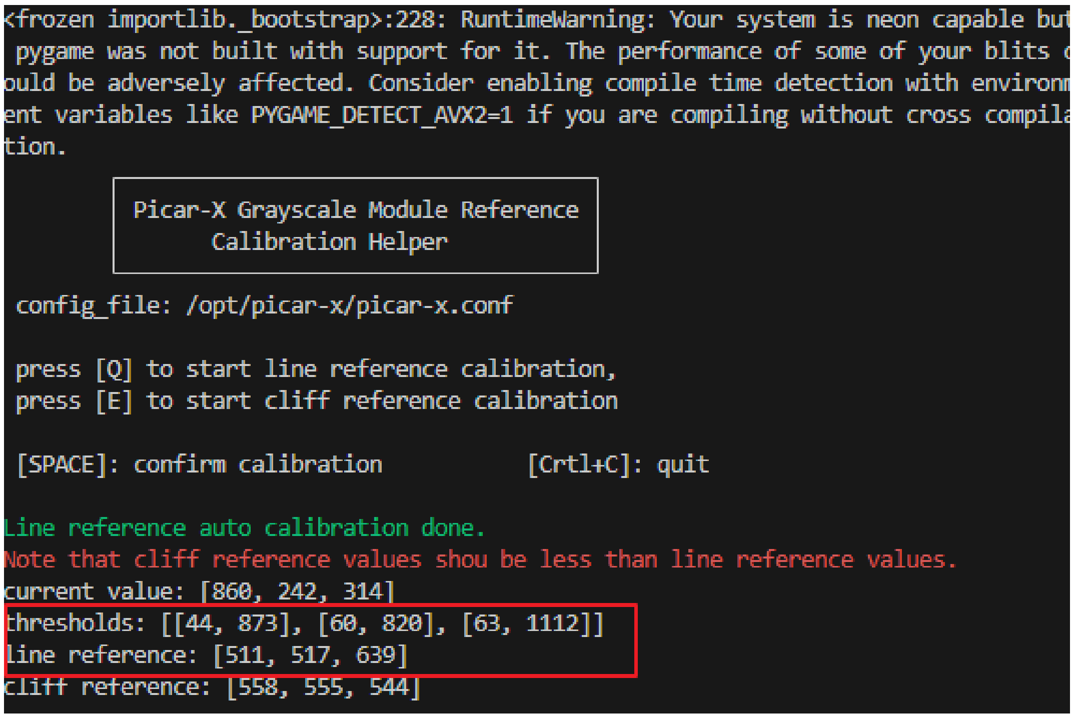
3. After running the code, you will see the following interface displayed in the terminal.



```
<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

 ----------

current value: [854, 791, 922]
thresholds: [[56, 872], [82, 820], [83, 991]]
line reference: [1000.0, 1000.0, 1000.0]
cliff reference: [500.0, 500.0, 500.0]
```

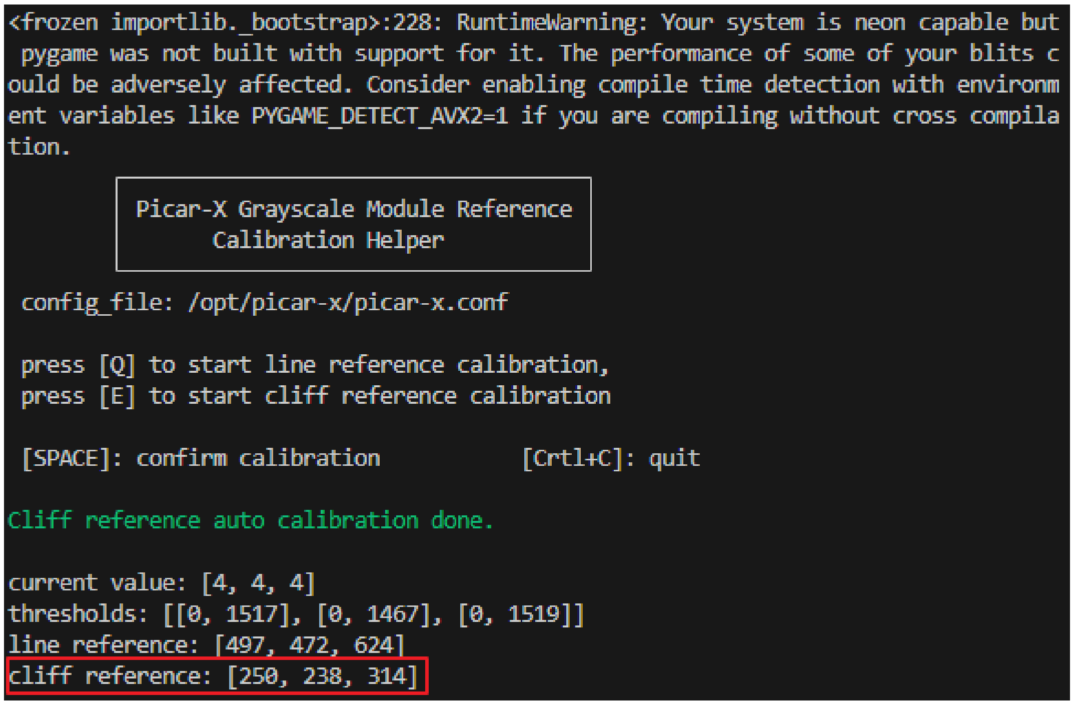
4. Press the "Q" key to initiate the greyscale calibration. You'll then observe the PiCar-X make minor movements to both the left and the right. During this process, each of the three sensors should sweep across the electrical tape at least once.

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.



```
<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

Line reference auto calibration done.
Note that cliff reference values shou be less than line reference values.
current value: [860, 242, 314]
thresholds: [[44, 873], [60, 820], [63, 1112]]
line reference: [511, 517, 639]
cliff reference: [558, 555, 544]
```

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.

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

2.2.2 2. 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 2.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.

```
from picarx import Picarx
import time


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

(continues on next page)

(continued from previous page)

```
        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.

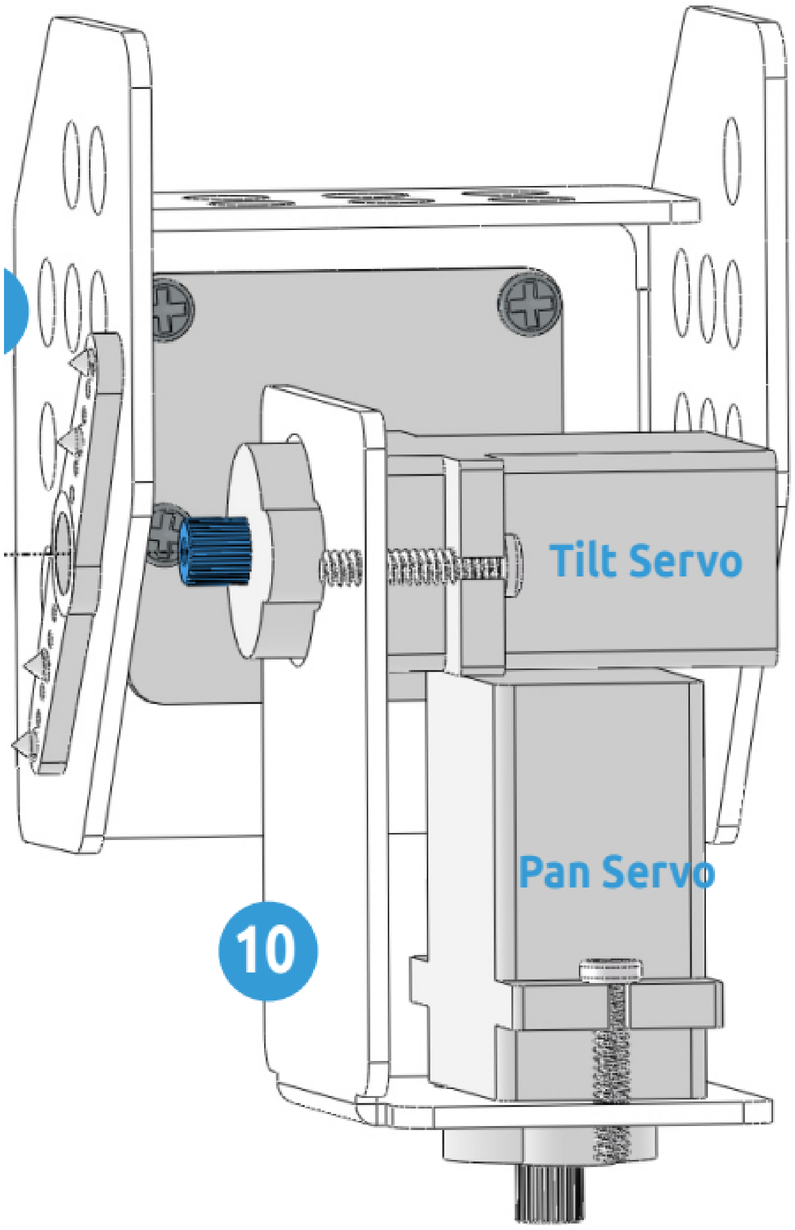
Now, the libraries to support the basic functionality of PiCar-X are imported. These lines will appear in all the examples that involve PiCar-X movement.

```
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.

```
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`.



Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

2.2.3 3. 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 3.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

```
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
        px = Picarx()
```

(continues on next page)

(continued from previous page)

```
        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.

```
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
```

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

2.2.4 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.

```
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.

```
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.

```
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.

```
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.

```
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.

```
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.

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

2.2.5 5. 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.

Run the Code

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

```
from picarx import Picarx
from time import sleep

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

last_state = "safe"

if __name__ == '__main__':
    try:
        while True:
```

(continues on next page)

(continued from previous page)

```
            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":
                    sleep(0.1)

            last_state = state

    except KeyboardInterrupt:
        print("\nKeyboardInterrupt: stop and exit")

    finally:
        px.stop()
        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.

2.2.6 6. 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 6.line_tracking.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.

```
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
    if _state == [0, 0, 0]:
```

(continues on next page)

(continued from previous page)

```
        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()

    except KeyboardInterrupt:
        print("\nKeyboardInterrupt: stop and exit")

    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.

  ```
  from picarx import Picarx
  from time import sleep

  ```

(continues on next page)

(continued from previous page)

```
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.

2.3 3. Computer Vision

Give your PiCar-X the ability to see using its camera. This section covers fun vision-based projects such as face tracking, recording, object interactions, and controlling the car via video or a mobile app.

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.

- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

2.3.1 7. Computer Vision

This project will officially enter the field of computer vision!

Run the Code

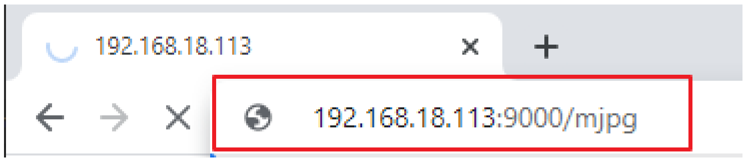
```
cd ~/picar-x/example
sudo python3 7.computer_vision.py
```

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

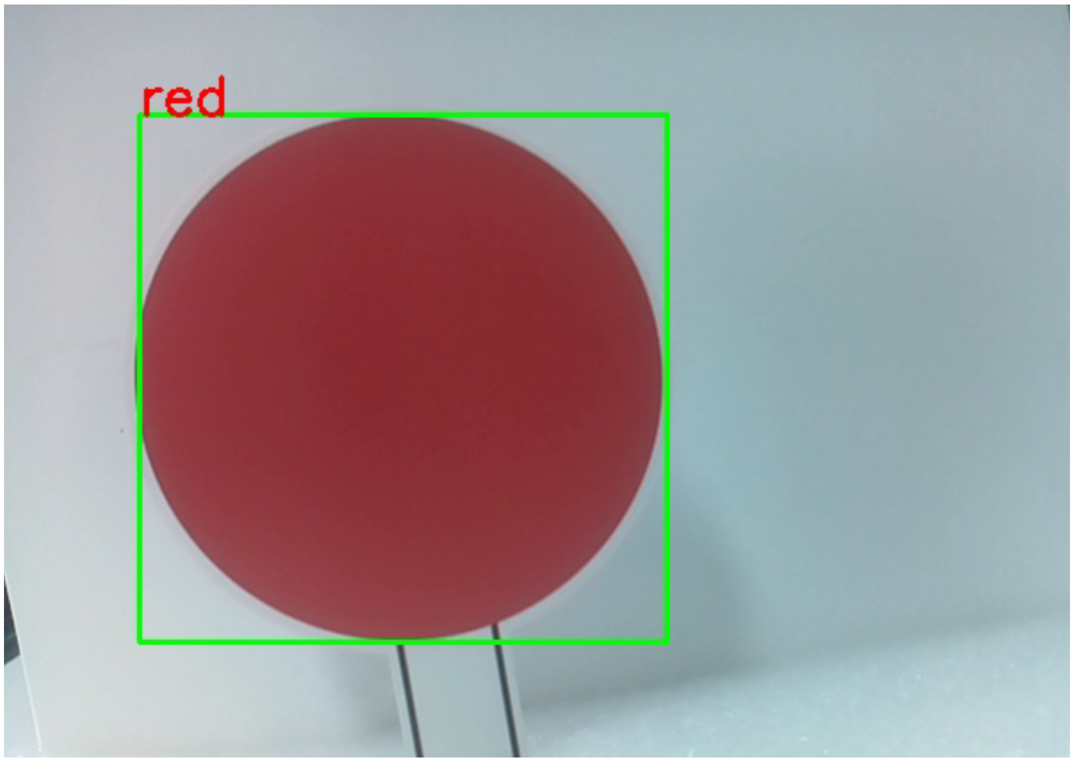
Please follow the prompts to activate the corresponding functions.

- **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**

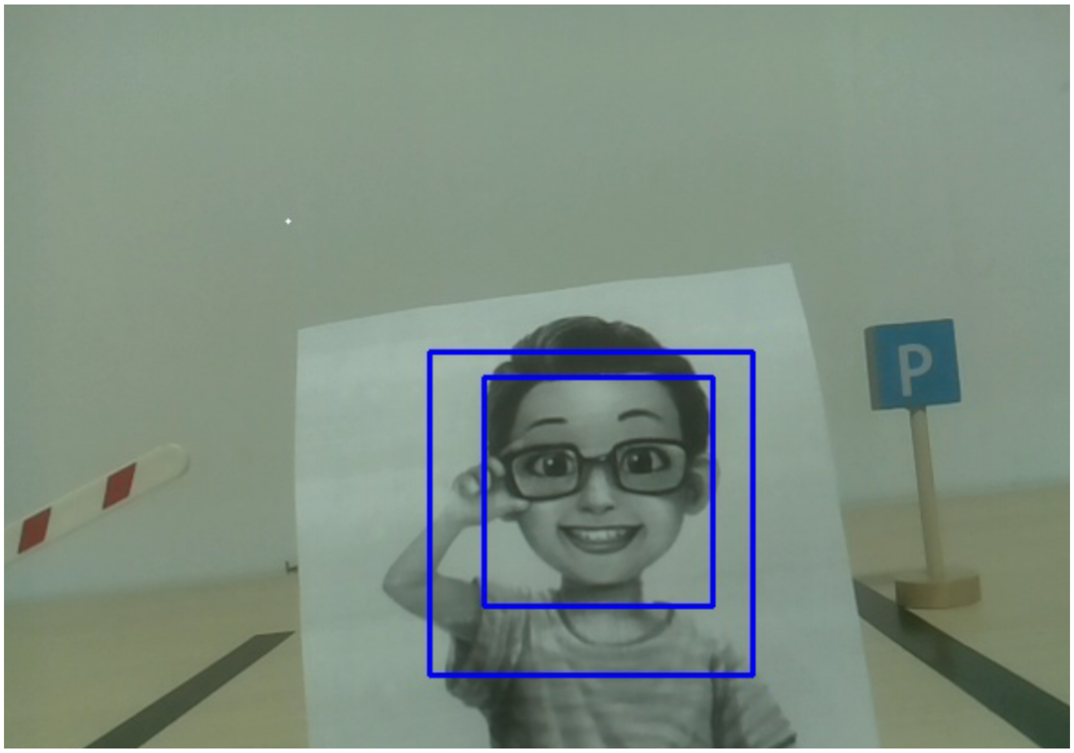
  Entering a number between `1~6` will detect one of the colors in "red, orange, yellow, green, blue, purple". Enter `0` to turn off color detection.

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

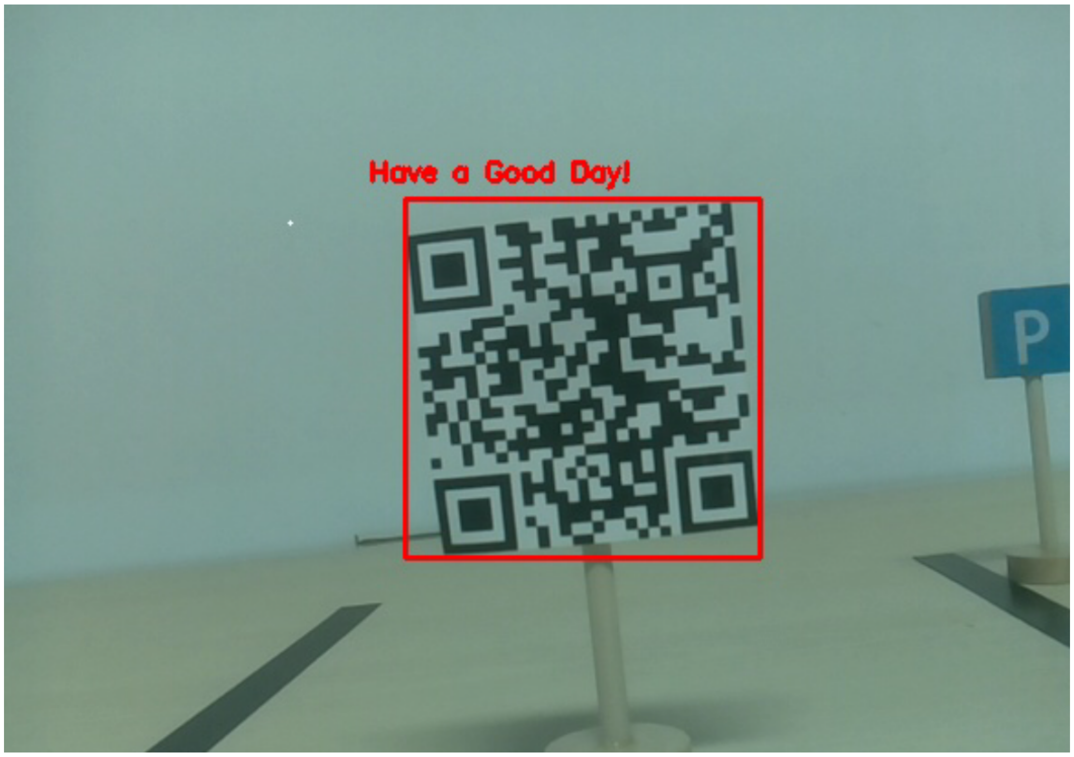
- **Face Detect**

  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

```
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
```

(continues on next page)

(continued from previous page)

```
        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':
```

(continues on next page)

(continued from previous page)

```
            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.

```
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

2.3.2 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

```
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
```

(continues on next page)

(continued from previous page)

```
                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()` .

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook!

Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

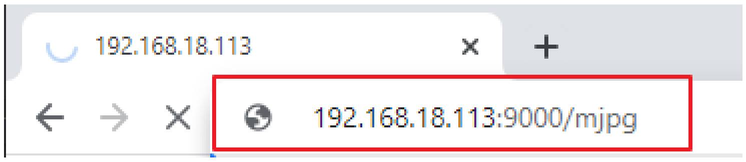
2.3.3 9. Record Video

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

```
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)

```

(continues on next page)

(continued from previous page)

```
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__":
```

(continues on next page)

(continued from previous page)

```
    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.

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

2.3.4 10. Bull Fight

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

Run the Code

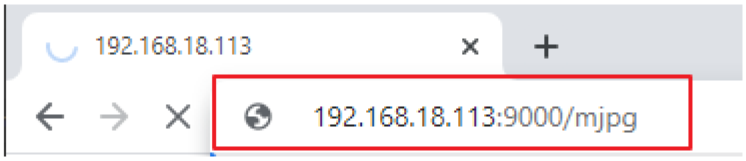
```
cd ~/picar-x/example
sudo python3 10.bull_fight.py
```

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`



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.

```
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.
            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)
```

(continues on next page)

(continued from previous page)

```
        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)`.

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.

- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

2.3.5 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.

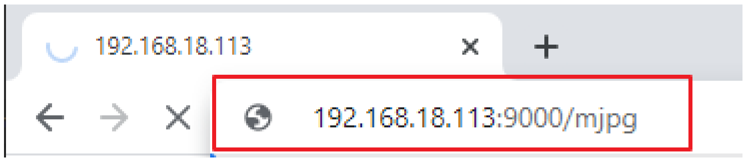
- O: speed up
- P: speed down
- W: forward
- S: backward
- A: turn left
- D: turn right
- F: stop
- T: take photo
- Ctrl+C: quit

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

```
#!/usr/bin/env python3

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

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':
```

(continues on next page)

(continued from previous page)

```
            px.set_dir_servo_angle(-30)
            px.forward(speed)
        elif operate == 'turn right':
            px.set_dir_servo_angle(30)
            px.forward(speed)



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'
```

(continues on next page)

(continued from previous page)

```
            # 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()
```

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

2.3.6 12. 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: *5. Install All the Modules(Important)*.

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

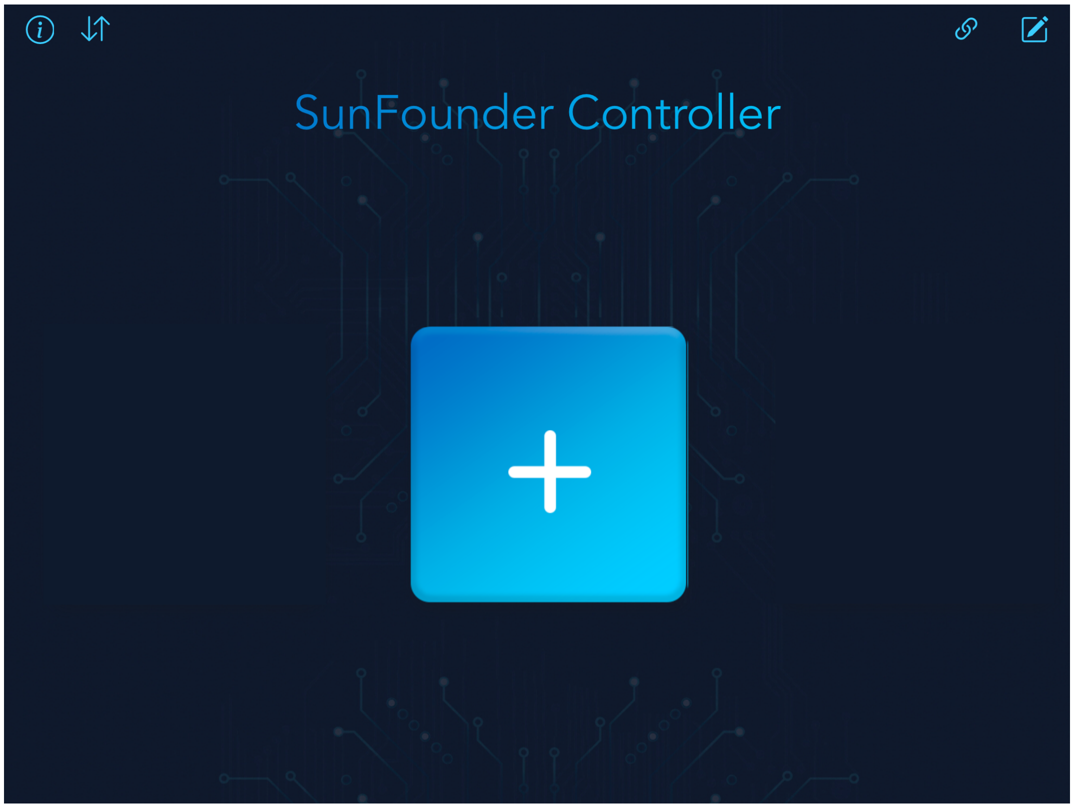
2. Run the code.

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

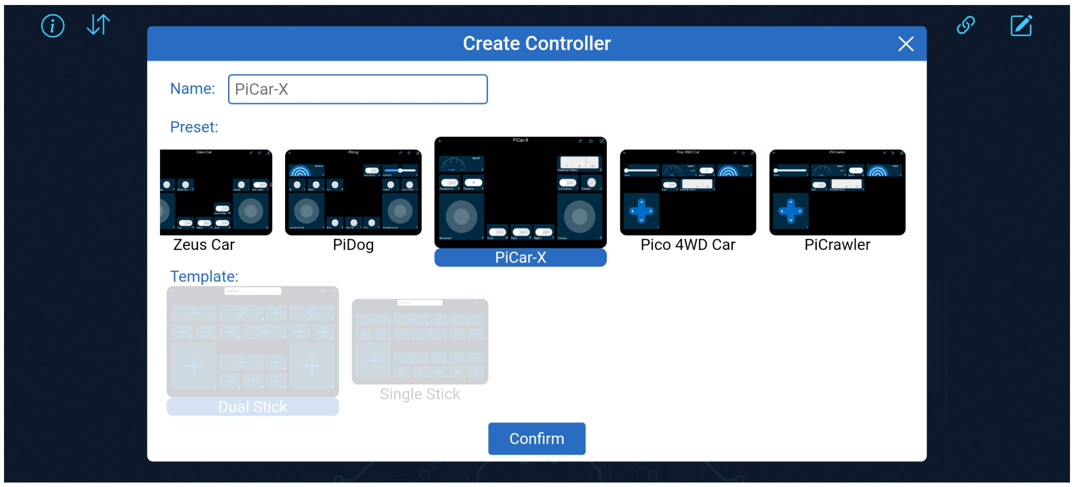
3. Install SunFounder Controller from **APP Store(iOS)** or **Google Play(Android)**.

4. Open and create a new controller.

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

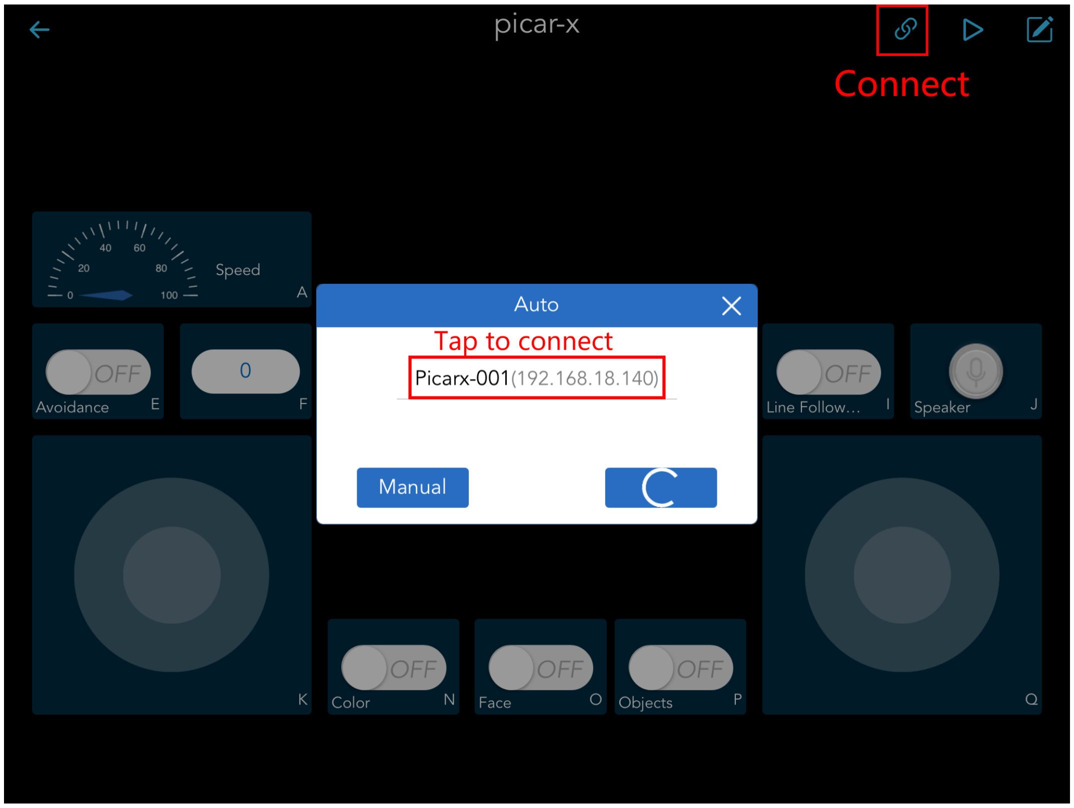
   

   There are preset controllers for some products in the Preset section, which you can use as needed. Here, we select **PiCar-X**.

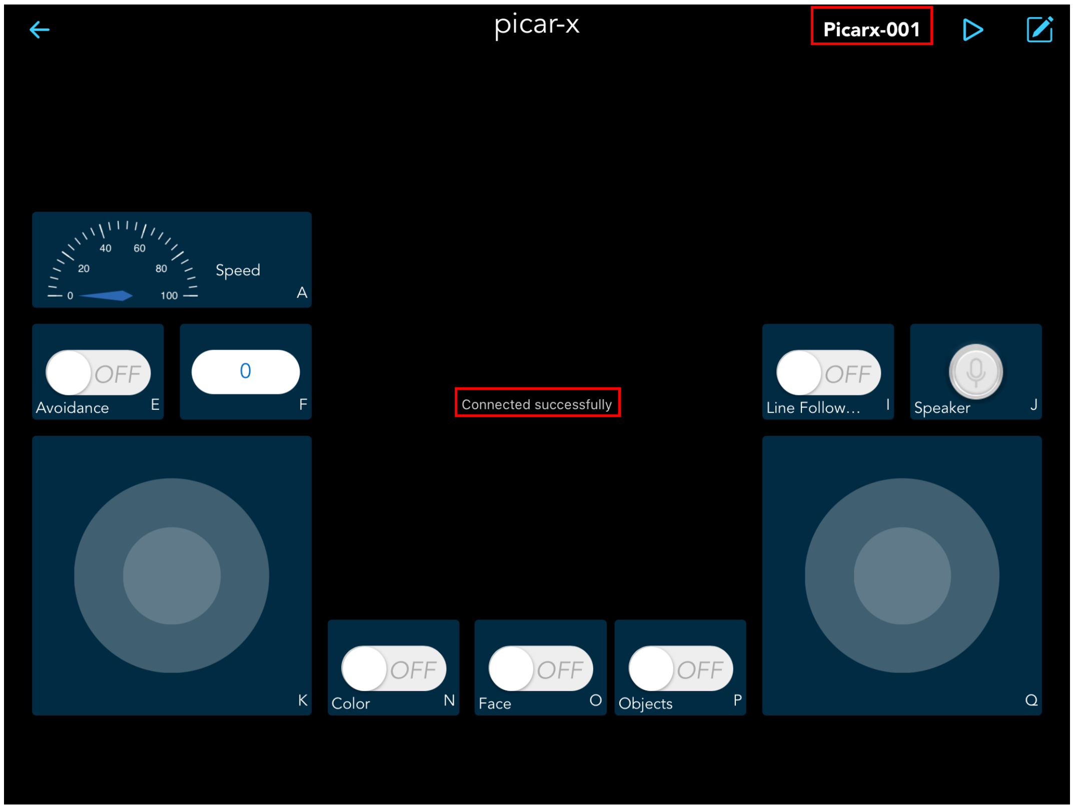


5. Connect to PiCar-x.

   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.

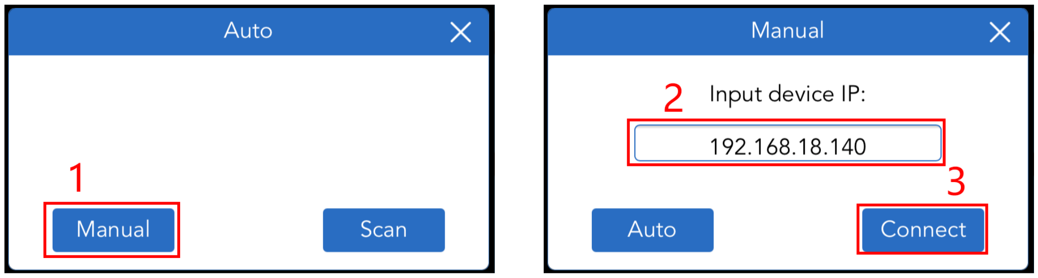


Once you click on the product name, the message "Connected Successfully" will appear and the product name will appear in the upper right corner.



Note:

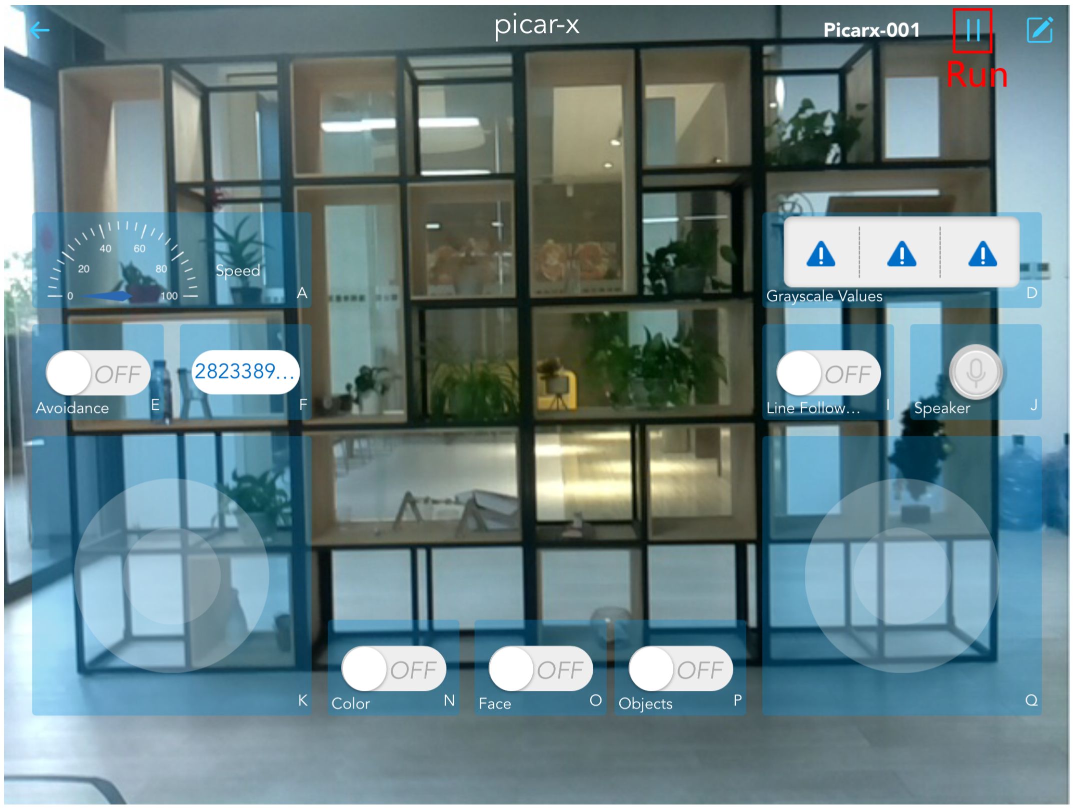
- You need to make sure that your mobile device is connected to the same LAN as PiCar-X.
- If it doesn't search automatically, you can also manually enter the IP to connect.



---

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.

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.

- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

CHAPTER

THREE

THINK · TALK · DRIVE — AI-POWERED WITH MULTI-LLMS

Go beyond movement and vision by adding **speech** and **AI**. Here you will explore text-to-speech (TTS), speech-to-text (STT), and large language models (LLMs) to make your PiCar-X talk, listen, and even chat with you like a smart robot.

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

3.1 13. Play Music and Sound Effects

In this project, you will learn how to make the PiCar-X play background music or sound effects. You can also play music files that you have stored.

Before You Start

Make sure you‘ve completed:

- *5. Install All the Modules(Important)* — Install `robot-hat`, `vilib`, `picar-x` modules, then run the script `i2samp.sh`.

Run the Code

```
cd ~/picar-x/example
sudo python3 13.sound_background_music.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
- q: Play/Stop Music

Code

```python
from time import sleep
from picarx.music import Music
import readchar

music = Music()

manual = '''
Input key to call the function!
    space: Play sound effect (Car horn)
    c: Play sound effect with threads
    q: Play/Stop Music
'''

def main():
    print(manual)

    flag_bgm = False
    music.music_set_volume(20)


    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)


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*.

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

3.2 14. Voice Prompt Car with Espeak and Pico2Wave

In this lesson, we'll use two built-in text-to-speech (TTS) engines on Raspberry Pi — **Espeak** and **Pico2Wave** — to make the PiCar-X talk.

These two engines are both simple and run offline, but they sound quite different:

- **Espeak**: very lightweight and fast, but the voice is robotic. You can adjust speed, pitch, and volume.
- **Pico2Wave**: produces a smoother and more natural voice than Espeak, but has fewer options to configure.

You'll hear the difference in **voice quality** and **features**, and then build a "voice prompt car" that announces its actions before moving.

3.2.1 Before You Start

Make sure you've completed:

- *5. Install All the Modules(Important)* — Install `robot-hat`, `vilib`, `picar-x` modules, then run the script `i2samp.sh`.

3.2.2 1. Testing Espeak

Espeak is a lightweight TTS engine included in Raspberry Pi OS. Its voice sounds robotic, but it is highly configurable: you can adjust volume, pitch, speed, and more.

Steps to try it out:

- Create a new file with the command:

```
cd ~/picar-x/example
sudo nano test_tts_espeak.py
```

- Then copy the example code into it. Press `Ctrl+X`, then `Y`, and finally `Enter` to save and exit.

```
from picarx.tts import Espeak

tts = Espeak()

# Optional voice tuning
# tts.set_amp(100)   # 0 to 200
# tts.set_speed(150) # 80 to 260
# tts.set_gap(5)     # 0 to 200
# tts.set_pitch(50)  # 0 to 99

# Quick hello (sanity check)
tts.say("Hello! I'm Espeak TTS.")
```

- Run the program with:

```
sudo python3 test_tts_espeak.py
```

- You should hear the PiCar-X say: “Hello! I’m Espeak TTS.”
- Uncomment the voice tuning lines in the code to experiment with how `amp`, `speed`, `gap`, and `pitch` affect the sound.

3.2.3 2. Testing Pico2Wave

Pico2Wave produces a more natural, human-like voice than Espeak. It’s simpler to use but less flexible — you can only change the language, not the pitch or speed.

Steps to try it out:

- Create a new file with the command:

```
cd ~/picar-x/example
sudo nano test_tts_pico2wave.py
```

- Then copy the example code into it. Press `Ctrl+X`, then `Y`, and finally `Enter` to save and exit.

```
from picarx.tts import Pico2Wave

tts = Pico2Wave()

tts.set_lang('en-US')  # en-US, en-GB, de-DE, es-ES, fr-FR, it-IT

# Quick hello (sanity check)
tts.say("Hello! I'm Pico2Wave TTS.")
```

- Run the program with:

```
sudo python3 test_tts_pico2wave.py
```

- You should hear the PiCar-X say: “Hello! I’m Pico2Wave TTS.”
- Try switching the language (for example, `es-ES` for Spanish) and listen to the difference.

3.2.4 3. Voice Prompt Car

Now let's combine **Pico2Wave** or **Espeak** with PiCar-X driving code to create a "voice prompt car": before every action, the car will announce what it's about to do.

Run the Code

```
cd ~/picar-x/example
sudo python3 14.voice_promt_car.py
```

Run this code and you'll see your PiCar-X drive forward, backward, and turn, each time announcing its move first. This makes your car safer, friendlier, and more interactive.

Code

```
from picarx import Picarx
from picarx.tts import Espeak
import time

# If you want to try Pico2Wave instead of Espeak, uncomment below:
# from picarx.tts import Pico2Wave
# tts = Pico2Wave()
# tts.set_lang('en-US')  # Options: en-US, en-GB, de-DE, es-ES, fr-FR, it-IT

px = Picarx()
tts = Espeak()

# Quick hello (test)
tts.say("Hello! I'm PiCar-X.")

def main():
    try:
        # Forward
        tts.say("Moving forward")
        px.forward(30)
        time.sleep(2)
        px.stop()

        # Backward
        tts.say("Moving backward")
        px.backward(30)
        time.sleep(2)
        px.stop()

        # Turn left
        tts.say("Turning left")
        px.set_dir_servo_angle(-20)
        px.forward(30)
        time.sleep(2)
        px.stop()
        px.set_dir_servo_angle(0)

        # Turn right
        tts.say("Turning right")
        px.set_dir_servo_angle(20)
```

(continues on next page)

(continued from previous page)

```
        px.forward(30)
        time.sleep(2)
        px.stop()
        px.set_dir_servo_angle(0)

    except KeyboardInterrupt:
        # Stop if interrupted
        px.stop()
    finally:
        # Reset to safe state
        px.stop()
        px.set_dir_servo_angle(0)

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

3.2.5 Troubleshooting

- **No sound when running Espeak or Pico2Wave**
  - Check that your speakers/headphones are connected and volume is not muted.
  - Run a quick test in terminal:

    ```
    espeak "Hello world"
    pico2wave -w test.wav "Hello world" && aplay test.wav
    ```

  If you hear nothing, the issue is with audio output, not your Python code.
- **Espeak voice sounds too fast or too robotic**
  - Try adjusting the parameters in your code:

    ```
    tts.set_speed(120)   # slower
    tts.set_pitch(60)    # different pitch
    ```

- **Permission denied when running code**
  - Try running with sudo:

    ```
    sudo python3 test_tts_espeak.py
    ```

3.2.6 Comparison: Espeak vs Pico2Wave

Feature	Espeak	Pico2Wave
Voice quality	Robotic, synthetic	More natural, human-like
Languages	Default English	Fewer, but common ones
Adjustable	Yes (speed, pitch, etc.)	No (only language)
Performance	Very fast, lightweight	Slightly slower, heavier

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

3.3 15. AI Storytelling Robot with Piper and OpenAI

In the previous lesson, we tried two built-in TTS engines on Raspberry Pi (**Espeak** and **Pico2Wave**). Now let's explore two more powerful options: **Piper** (offline, neural network-based) and **OpenAI TTS** (online, cloud-based).

- **Piper**: a local TTS engine that runs offline on Raspberry Pi.
- **OpenAI TTS**: an online service that provides very natural, human-like voices.

At the end, your PiCar-X will drive around and tell jokes like a little storyteller.

3.3.1 Before You Start

Make sure you've completed:

- *5. Install All the Modules(Important)* — Install `robot-hat`, `vilib`, `picar-x` modules, then run the script `i2samp.sh`.

3.3.2 1. Testing Piper

Steps to try it out:

1. Create a new file:

```
cd ~/picar-x/example
sudo nano test_tts_piper.py
```

2. Copy the example code below into the file. Press `Ctrl+X`, then `Y`, and finally `Enter` to save and exit.

```
from picarx.tts import Piper

tts = Piper()

# List supported languages
```

(continues on next page)

(continued from previous page)

```
print(tts.available_countrys())

# List models for English (en_us)
print(tts.available_models('en_us'))

# Set a voice model (auto-download if not already present)
tts.set_model("en_US-amy-low")

# Say something
tts.say("Hello! I'm Piper TTS.")
```

- `available_countrys()`: print supported languages.
- `available_models()`: list available models for that language.
- `set_model()`: set the voice model (downloads automatically if missing).
- `say()`: convert text to speech and play it.

3. Run the program:

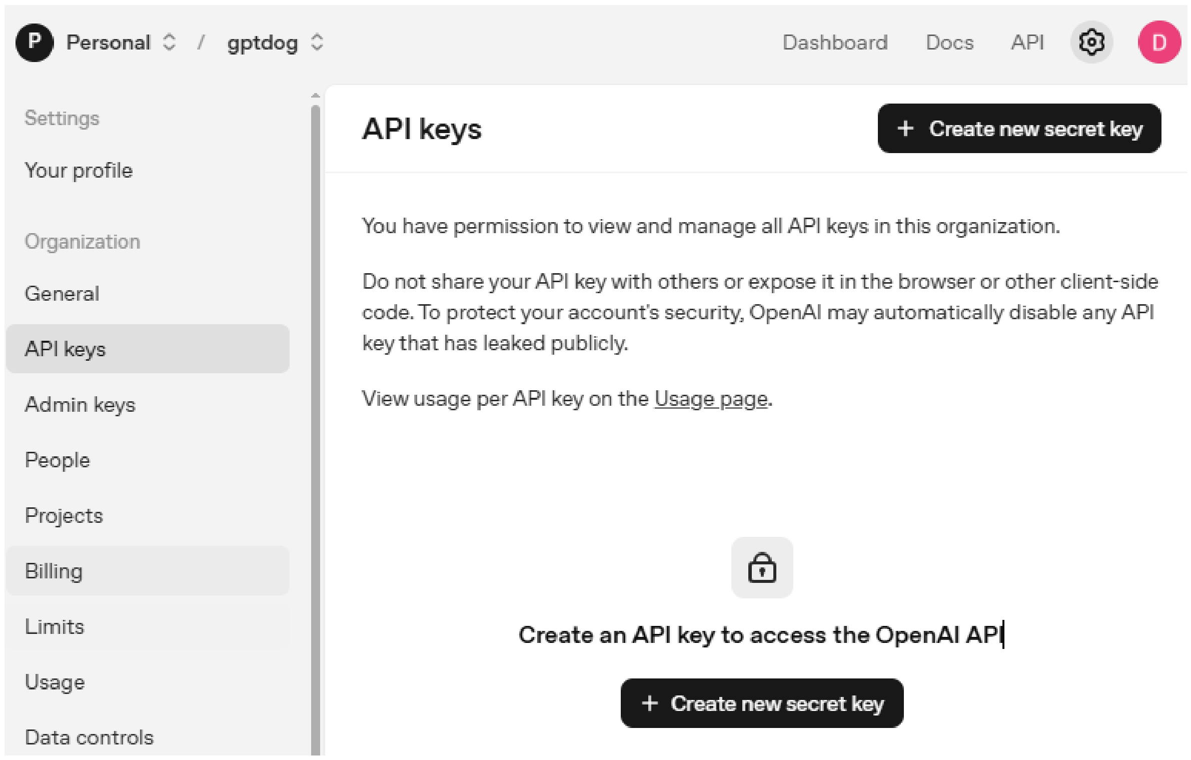
```
sudo python3 test_tts_piper.py
```

4. The first time you run it, the selected voice model will be downloaded automatically.
   - You should then hear the PiCar-X say: `Hello! I'm Piper TTS.`
   - You can change to another language model by calling `set_model()` with a different name.

3.3.3 2. Testing OpenAI TTS

Get and save your API Key

1. Go to and log in. On the **API keys** page, click **Create new secret key**.



API keys

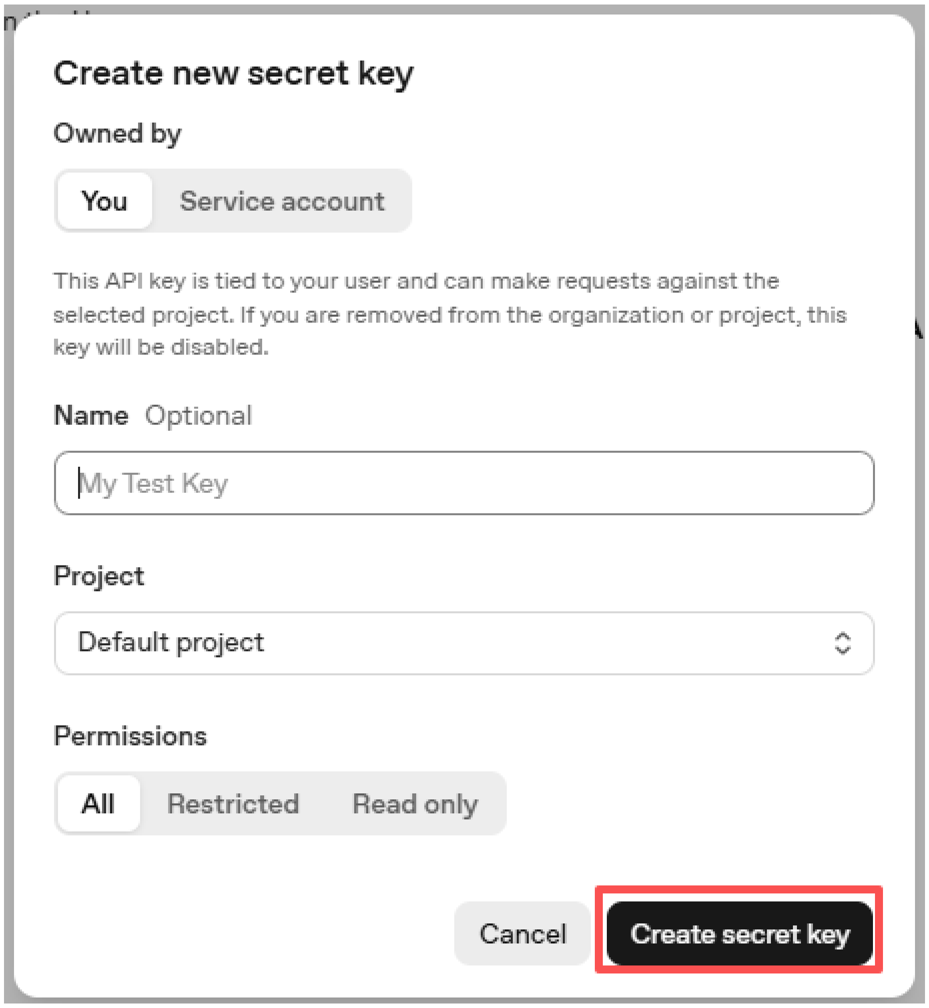
You have permission to view and manage all API keys in this organization.

Do not share your API key with others or expose it in the browser or other client-side code. To protect your account's security, OpenAI may automatically disable any API key that has leaked publicly.

View usage per API key on the Usage page.

Create an API key to access the OpenAI API

2. Fill in the details (Owner, Name, Project, and permissions if needed), then click **Create secret key**.


Create new secret key

Owned by

You | Service account

This API key is tied to your user and can make requests against the selected project. If you are removed from the organization or project, this key will be disabled.

Name Optional

My Test Key

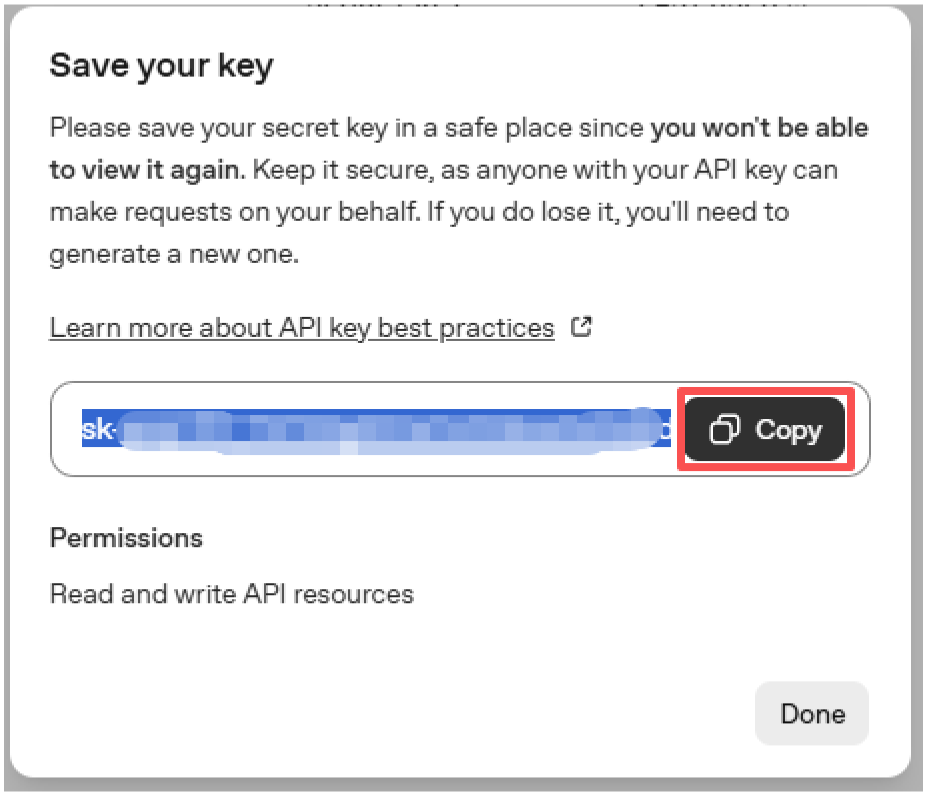
Project

Default project

Permissions

All | Restricted | Read only

3. Once the key is created, copy it right away — you won't be able to see it again. If you lose it, you must generate a new one.



Save your key

Please save your secret key in a safe place since **you won't be able to view it again**. Keep it secure, as anyone with your API key can make requests on your behalf. If you do lose it, you'll need to generate a new one.

Learn more about API key best practices

sk-[redacted]

Permissions

Read and write API resources

4. In your project folder (for example: `/picar-x/example`), create a file called `secret.py`:

```
cd ~/picar-x/example
sudo nano secret.py
```

5. Paste your key into the file like this:

```
# secret.py
# Store secrets here. Never commit this file to Git.
OPENAI_API_KEY = "sk-xxx"
```

Write and Run a Test Program

1. Create a new file:

```
cd ~/picar-x/example
sudo nano test_tts_openai.py
```

2. Copy the example code below into the file. Press `Ctrl+X`, then `Y`, and finally `Enter` to save and exit.

```
from picarx.tts import OpenAI_TTS
from secret import OPENAI_API_KEY   # or use the try/except version shown above

# Initialize OpenAI TTS
tts = OpenAI_TTS(api_key=OPENAI_API_KEY)
tts.set_model('gpt-4o-mini-tts')  # low-latency TTS model
tts.set_voice('alloy')            # pick a voice

# Quick hello (sanity check)
tts.say("Hello! I'm OpenAI TTS.")
```

3. Run the program:

   ```
   sudo python3 test_tts_openai.py
   ```

4. You should hear the PiCar-X say:

   ```
   Hello! I'm OpenAI TTS.
   ```

3.3.4 3. Storytelling Robot

Now that we have tested both **Piper** and **OpenAI TTS**, let's use them in a real project: a **storytelling robot car** that drives around while telling jokes.

In this program, the PiCar-X will:

- Greet you with TTS when it starts.
- Move forward and tell a first joke.
- Move forward again and tell a second joke.
- Finally drive backward, return "home," and say goodbye.

It's like having a little robot storyteller on wheels!

Run the code

```
cd ~/picar-x/example
sudo python3 15.storytelling_robot.py
```

Code

```
from picarx import Picarx
import time

# === TTS Configuration ===
# Default: Piper
from picarx.tts import Piper
tts = Piper()
tts.set_model("en_US-amy-low")  # use the voice model you installed

# Optional: switch to OpenAI TTS
# from picarx.tts import OpenAI_TTS
# from secret import OPENAI_API_KEY
# tts = OpenAI_TTS(api_key=OPENAI_API_KEY)
# tts.set_model("gpt-4o-mini-tts")  # low-latency TTS model
# tts.set_voice("alloy")            # choose a voice

# === PiCar-X Setup ===
px = Picarx()

# Quick hello (sanity check)
tts.say("Hello! I'm PiCar-X speaking with Piper.")

def main():
    try:
        # Leg 1
```

(continues on next page)

(continued from previous page)

```
        px.forward(30)
        time.sleep(3)
        px.stop()
        tts.say("Why can't your nose be twelve inches long? Because then it would be a␣
↪foot!")

        # Leg 2
        px.forward(30)
        time.sleep(3)
        px.stop()
        tts.say("Why did the cow go to outer space? To see the moooon!")

        # Wrap-up
        tts.say("That's all for today. Goodbye, let's go home and sleep.")
        px.backward(30)
        time.sleep(6)
        px.stop()

    except KeyboardInterrupt:
        px.stop()
    finally:
        px.stop()
        px.set_dir_servo_angle(0)

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

3.3.5 Troubleshooting

- **No module named 'secret'**

  This means `secret.py` is not in the same folder as your Python file. Move `secret.py` into the same directory where you run the script, e.g.:

  ```
  ls ~/picar-x/example
  # Make sure you see both: secret.py and your .py file
  ```

- **OpenAI: Invalid API key / 401**
  - Check that you pasted the full key (starts with `sk-`) and there are no extra spaces/newlines.
  - Ensure your code imports it correctly:

    ```
    from secret import OPENAI_API_KEY
    ```

  - Confirm network access on your Pi (try `ping api.openai.com`).
- **OpenAI: Quota exceeded / billing error**
  - You may need to add billing or increase quota in the OpenAI dashboard.
  - Try again after resolving the account/billing issue.
- **Piper: tts.say() runs but no sound**

  - Make sure a voice model is actually present:

    ```
    ls ~/.local/share/piper/voices
    ```

  - Confirm your model name matches exactly in code:

    ```
    tts.set_model("en_US-amy-low")
    ```

  - Check the audio output device/volume on your Pi (`alsamixer`), and that speakers are connected and powered.

- **ALSA / sound device errors (e.g., "Audio device busy" or "No such file or directory")**
  - Close other programs using audio.
  - Reboot the Pi if the device stays busy.
  - For HDMI vs. headphone jack output, select the correct device in Raspberry Pi OS audio settings.
- **Permission denied when running Python**
  - Try with `sudo` if your environment requires it:

    ```
    sudo python3 test_tts_piper.py
    ```

3.3.6 Comparison of TTS Engines

Table 1: Feature comparison: Espeak vs Pico2Wave vs Piper vs OpenAI TTS

Item	Espeak	Pico2Wave	Piper	OpenAI TTS
Runs on	Built-in on Raspberry Pi (offline)	Built-in on Raspberry Pi (offline)	Raspberry Pi / PC (offline, needs model)	Cloud (online, needs API key)
Voice quality	Robotic	More natural than Espeak	Natural (neural TTS)	Very natural / human-like
Controls	Speed, pitch, volume	Limited controls	Choose different voices/models	Choose model and voices
Languages	Many (quality varies)	Limited set	Many voices/languages available	Best in English (others vary by availability)
Latency / speed	Very fast	Fast	Real-time on Pi 4/5 with "low" models	Network-dependent (usually low latency)
Setup	Minimal	Minimal	Download `.onnx` + `.onnx.json` models	Create API key, install client
Best for	Quick tests, basic prompts	Slightly better offline voice	Local projects with better quality	Highest quality, rich voice options

- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

3.4 16. Voice Controlled Car with Vosk (Offline)

Vosk is a lightweight speech-to-text (STT) engine that supports many languages and runs fully **offline** on Raspberry Pi. You only need internet access once to download a language model. After that, everything works without a network connection.

In this lesson, we will:

- Check the microphone on Raspberry Pi.
- Install and test Vosk with a chosen language model.
- Build a **voice controlled PiCar-X** that listens for a wake word and then responds to commands like **forward**, **backward**, **left**, and **right**.

3.4.1 Before You Start

Make sure you‘ve completed:

- *5. Install All the Modules(Important)* — Install `robot-hat`, `vilib`, `picar-x` modules, then run the script `i2samp.sh`.

3.4.2 1. Check Your Microphone

Before using speech recognition, make sure your USB microphone works correctly.

1. List available recording devices:

   ```
   arecord -l
   ```

   Look for a line like `card 1: ... device 0`.

2. Record a short sample (replace `1,0` with the numbers you found):

   ```
   arecord -D plughw:1,0 -f S16_LE -r 16000 -d 3 test.wav
   ```

   - Example: if your device is `card 2, device 0`, use:

   ```
   arecord -D plughw:2,0 -f S16_LE -r 16000 -d 3 test.wav
   ```

3. Play it back to confirm the recording:

   ```
   aplay test.wav
   ```

4. Adjust microphone volume if needed:

```
alsamixer
```

- Press **F6** to select your USB microphone.
- Find the **Mic** or **Capture** channel.
- Make sure it is not muted (**[MM]** means mute, press `M` to unmute → should show **[OO]**).
- Use ↑ / ↓ arrow keys to change the recording volume.

3.4.3 2. Test Vosk

Steps to try it out:

1. Create a new file:

```
cd ~/picar-x/example
sudo nano test_stt_vosk.py
```

2. Copy the example code into it. Press `Ctrl+X`, then `Y`, and `Enter` to save and exit.

```
from picarx.stt import Vosk

vosk = Vosk(language="en-us")

print(vosk.available_languages)

while True:
    print("Say something")
    result = vosk.listen(stream=False)
    print(result)
```

3. Run the program:

```
sudo python3 test_stt_vosk.py
```

4. The first time you run this code with a new language, Vosk will **automatically download the language model** (by default it will download the **small** version). At the same time, it will also print out the list of supported languages. Then you will see:

```
vosk-model-small-en-us-0.15.zip: 100%|| 39.3M/39.3M [00:05<00:00, 7.85MB/s]
['ar', 'ar-tn', 'ca', 'cn', 'cs', 'de', 'en-gb', 'en-in', 'en-us', 'eo', 'es', 'fa',
↪ 'fr', 'gu', 'hi', 'it', 'ja', 'ko', 'kz', 'nl', 'pl', 'pt', 'ru', 'sv', 'te', 'tg
↪', 'tr', 'ua', 'uz', 'vn']
Say something
```

This means:

- The model file (`vosk-model-small-en-us-0.15`) has been downloaded.
- The list of supported languages has been printed.
- The system is now listening — say something into the PiCar-X microphone, and the recognized text will appear in the terminal.

Tips:

- Keep the microphone about 15–30 cm away.

- Pick a model that matches your language and accent.

Streaming Mode (optional)

You can also stream speech continuously to see partial results as you speak:

```
from picarx.stt import Vosk

vosk = Vosk(language="en-us")

while True:
    print("Say something")
    for result in vosk.listen(stream=True):
        if result["done"]:
            print(f"final:   {result['final']}")
        else:
            print(f"partial: {result['partial']}", end="\r", flush=True)
```

3.4.4 3. Voice Controlled Car

Now let's connect speech recognition to the PiCar-X!

We will use a **wake word** ("hey robot") so the car only listens for commands after being activated. This saves CPU and prevents unwanted triggers.

Run the code

```
cd ~/picar-x/example
sudo python3 16.voice_controlled_car.py
```

In this program, the car:

- Waits for the wake word **"hey robot"**.
- After that, you can speak naturally — as long as your sentence includes one of the keywords (**forward**, **backward**, **left**, **right**), the car will respond.

  For example:

  – "Can you move forward a little?" → the car moves forward.

  – "Please turn left now." → the car turns left.
- The command **"sleep"** stops the control loop and puts the car back into waiting mode.

Code

```
from picarx import Picarx
from picarx.stt import Vosk
import time

px = Picarx()
stt = Vosk(language="en-us")

WAKE_WORDS = ["hey robot"]

print('Say "hey robot" to wake me up! Then say: forward / backward / left / right. Say
↪"sleep" to stop listening.')
```

(continues on next page)

(continued from previous page)

```
try:
    while True:
        # --- wait for wake word once ---
        stt.wait_until_heard(WAKE_WORDS)
        print("Wake word detected. Listening for commands... (say 'sleep' to pause)")

        # --- command loop: multiple commands after one wake ---
        while True:
            res = stt.listen(stream=False)
            text = res.get("text", "") if isinstance(res, dict) else str(res)
            text = text.lower().strip()
            if not text:
                continue

            print("Heard:", text)

            if "sleep" in text:
                # pause command mode; go back to wait for wake word
                px.stop(); px.set_dir_servo_angle(0)
                print("Sleeping. Say 'hey robot' to wake me again.")
                break

            elif "forward" in text:
                px.set_dir_servo_angle(0)
                px.forward(30); time.sleep(1); px.stop()

            elif "backward" in text:
                px.set_dir_servo_angle(0)
                px.backward(30); time.sleep(1); px.stop()

            elif "left" in text:
                px.set_dir_servo_angle(-25)
                px.forward(30); time.sleep(1)
                px.stop(); px.set_dir_servo_angle(0)

            elif "right" in text:
                px.set_dir_servo_angle(25)
                px.forward(30); time.sleep(1)
                px.stop(); px.set_dir_servo_angle(0)
            # (ignore other words)

except KeyboardInterrupt:
    pass
finally:
    px.stop(); px.set_dir_servo_angle(0)
    print("Stopped and centered. Bye.")
```

3.4.5 Troubleshooting

- **No such file or directory (when running `arecord`)**

  You may have used the wrong card/device number. Run:

  ```
  arecord -l
  ```

  and replace `1,0` with the numbers shown for your USB microphone.

- **Recorded file has no sound**

  Open the mixer and check the microphone volume:

  ```
  alsamixer
  ```

  - Press **F6** to select your USB mic.
  - Make sure **Mic/Capture** is not muted (**[OO]** instead of **[MM]**).
  - Increase the level with ↑.

- **Vosk does not recognize speech**
  - Make sure the **language code** matches your model (e.g. `en-us` for English, `zh-cn` for Chinese).
  - Keep the microphone 15–30 cm away and avoid background noise.
  - Speak clearly and slowly.

- **Wake word ("hey robot") never triggers**
  - Say it in a natural tone, not too fast.
  - Check that the program prints recognized text at all. If not, the microphone is not working.

- **High latency / slow recognition**
  - The default auto-download is a **small model** (faster, but less accurate).
  - If it's still slow, close other programs to free CPU.

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

3.5 17. Text Vision Talk with Ollama

In this lesson, you will learn how to use **Ollama**, a tool for running large language and vision models locally. We will show you how to install Ollama, download a model, and connect PiCar-X to it.

With this setup, PiCar-X can take a camera snapshot and the model will **see and tell** — you can ask any question about the image, and the model will reply in natural language.

3.5.1 Before You Start

Make sure you've completed:

- *5. Install All the Modules(Important)* — Install `robot-hat`, `vilib`, `picar-x` modules, then run the script `i2samp.sh`.

3.5.2 1. Install Ollama (LLM) and Download Model

You can choose where to install **Ollama**:

- On your Raspberry Pi (local run)
- Or on another computer (Mac/Windows/Linux) in the **same local network**

Recommended models vs hardware

You can choose any model available on . Models come in different sizes (3B, 7B, 13B, 70B…). Smaller models run faster and require less memory, while larger models provide better quality but need powerful hardware.

Check the table below to decide which model size fits your device.

Model size	Min RAM Required	Recommended Hardware
~3B parameters	8GB (16GB better)	Raspberry Pi 5 (16GB) or mid-range PC/Mac
~7B parameters	16GB+	Pi 5 (16GB, just usable) or mid-range PC/Mac
~13B parameters	32GB+	Desktop PC / Mac with high RAM
30B+ parameters	64GB+	Workstation / Server / GPU recommended
70B+ parameters	128GB+	High-end server with multiple GPUs

Install on Raspberry Pi

If you want to run Ollama directly on your Raspberry Pi:

- Use a **64-bit Raspberry Pi OS**
- Strongly recommended: **Raspberry Pi 5 (16GB RAM)**

Run the following commands:

```
# Install Ollama
curl -fsSL https://ollama.com/install.sh | sh

# Pull a lightweight model (good for testing)
ollama pull llama3.2:3b

# Quick run test (type 'hi' and press Enter)
ollama run llama3.2:3b
```

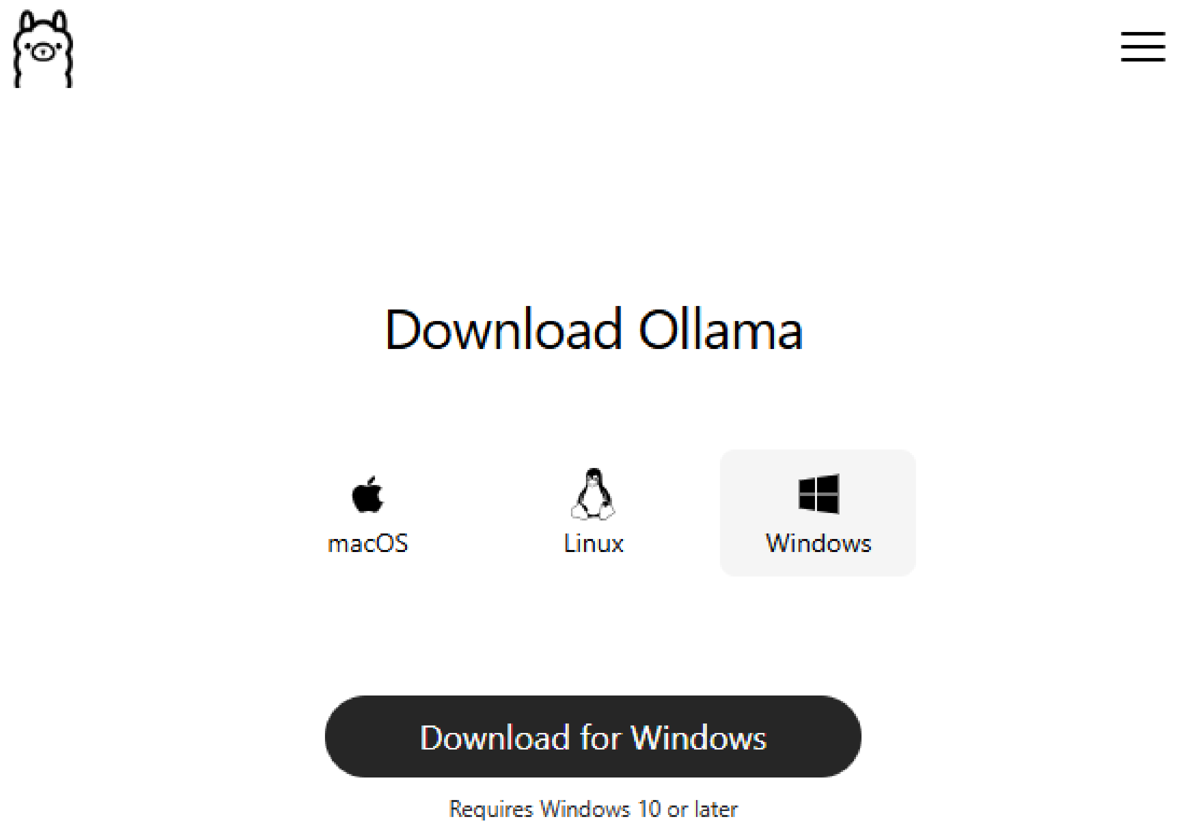
(continues on next page)

(continued from previous page)

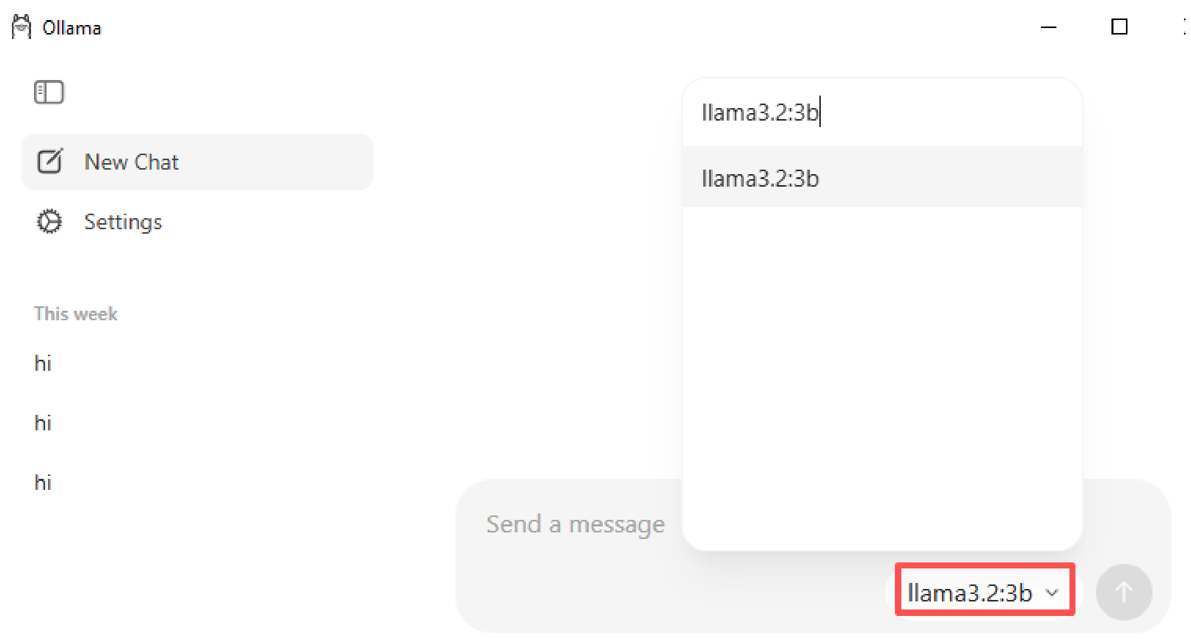
```
# Serve the API (default port 11434)
# Tip: set OLLAMA_HOST=0.0.0.0 to allow access from LAN
OLLAMA_HOST=0.0.0.0 ollama serve
```

Install on Mac / Windows / Linux (Desktop App)

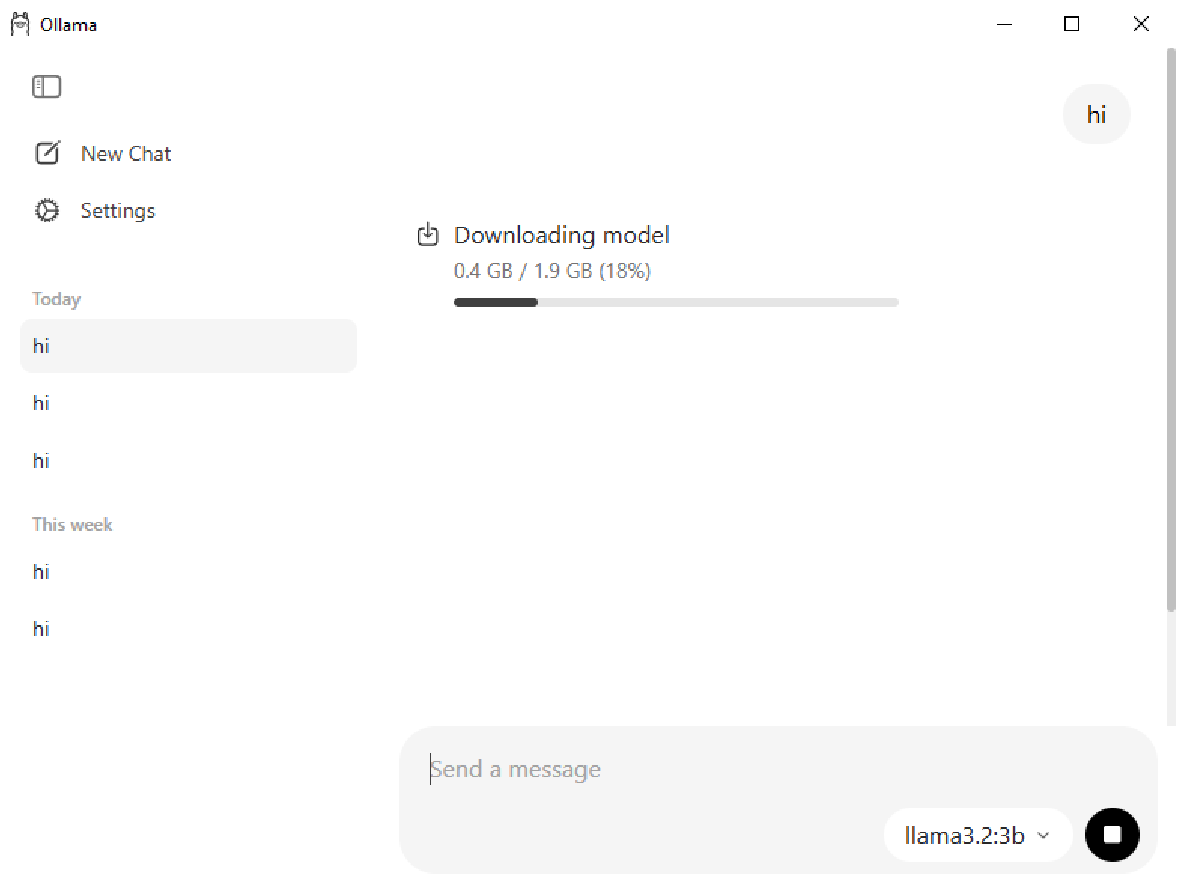
1. Download and install Ollama from



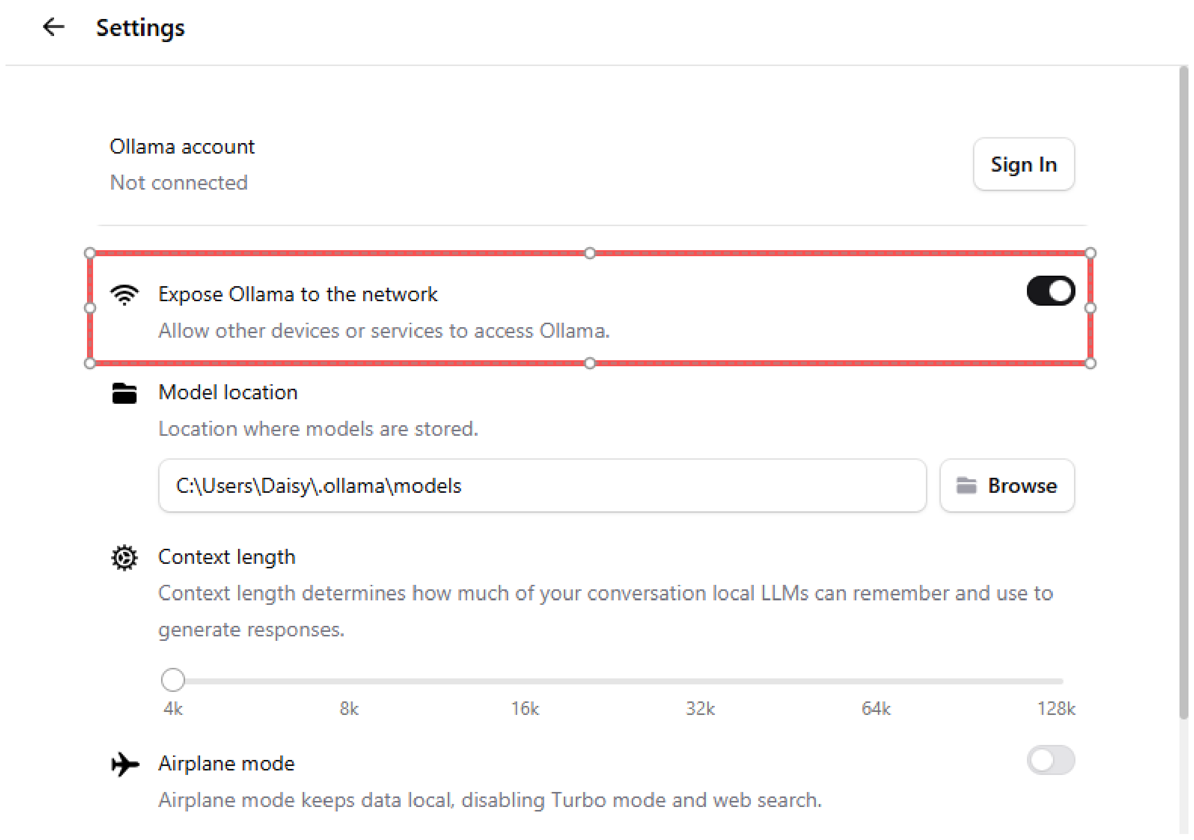
2. Open the Ollama app, go to the **Model Selector**, and use the search bar to find a model. For example, type `llama3.2:3b` (a small and lightweight model to start with).



3. After the download is complete, type something simple like "Hi" in the chat window, Ollama will automatically start downloading it when you first use it.



4. Go to **Settings** → enable **Expose Ollama to the network**. This allows your Raspberry Pi to connect to it over LAN.


Settings

Ollama account
Not connected

Expose Ollama to the network
Allow other devices or services to access Ollama.

Model location
Location where models are stored.

C:\Users\Daisy\.ollama\models

Context length
Context length determines how much of your conversation local LLMs can remember and use to generate responses.

Airplane mode
Airplane mode keeps data local, disabling Turbo mode and web search.

Warning: If you see an error like:

```
Error:  model requires more system memory ...
```

The model is too large for your machine. Use a **smaller model** or switch to a computer with more RAM.

3.5.3 2. Test Ollama

Once Ollama is installed and your model is ready, you can quickly test it with a minimal chat loop.

Steps

1. Create a new file:

```
cd ~/picar-x/example
nano test_llm_ollama.py
```

2. Paste the following code and save (`Ctrl+X` → `Y` → `Enter`):

```
from picarx.llm import Ollama

INSTRUCTIONS = "You are a helpful assistant."
WELCOME = "Hello, I am a helpful assistant. How can I help you?"

# If Ollama runs on the same Raspberry Pi, use "localhost".
# If it runs on another computer in your LAN, replace with that computer's IP␣
```

(continues on next page)

(continued from previous page)

```
↪address.
llm = Ollama(
    ip="localhost",
    model="llama3.2:3b"   # you can replace with any model
)

# Basic configuration
llm.set_max_messages(20)
llm.set_instructions(INSTRUCTIONS)
llm.set_welcome(WELCOME)

print(WELCOME)

while True:
    text = input(">>> ")
    if text.strip().lower() in {"exit", "quit"}:
        break

    # Response with streaming output
    response = llm.prompt(text, stream=True)
    for token in response:
        if token:
            print(token, end="", flush=True)
    print("")
```

3. Run the program:

```
python3 test_llm_ollama.py
```

4. Now you can chat with PiCar-X directly from the terminal.

   - You can choose **any model** available on , but smaller models (e.g. `moondream:1.8b`, `phi3:mini`) are recommended if you only have 8–16GB RAM.
   - Make sure the model you specify in the code matches the model you have already pulled in Ollama.
   - Type `exit` or `quit` to stop the program.
   - If you cannot connect, ensure that Ollama is running and that both devices are on the same LAN if you are using a remote host.

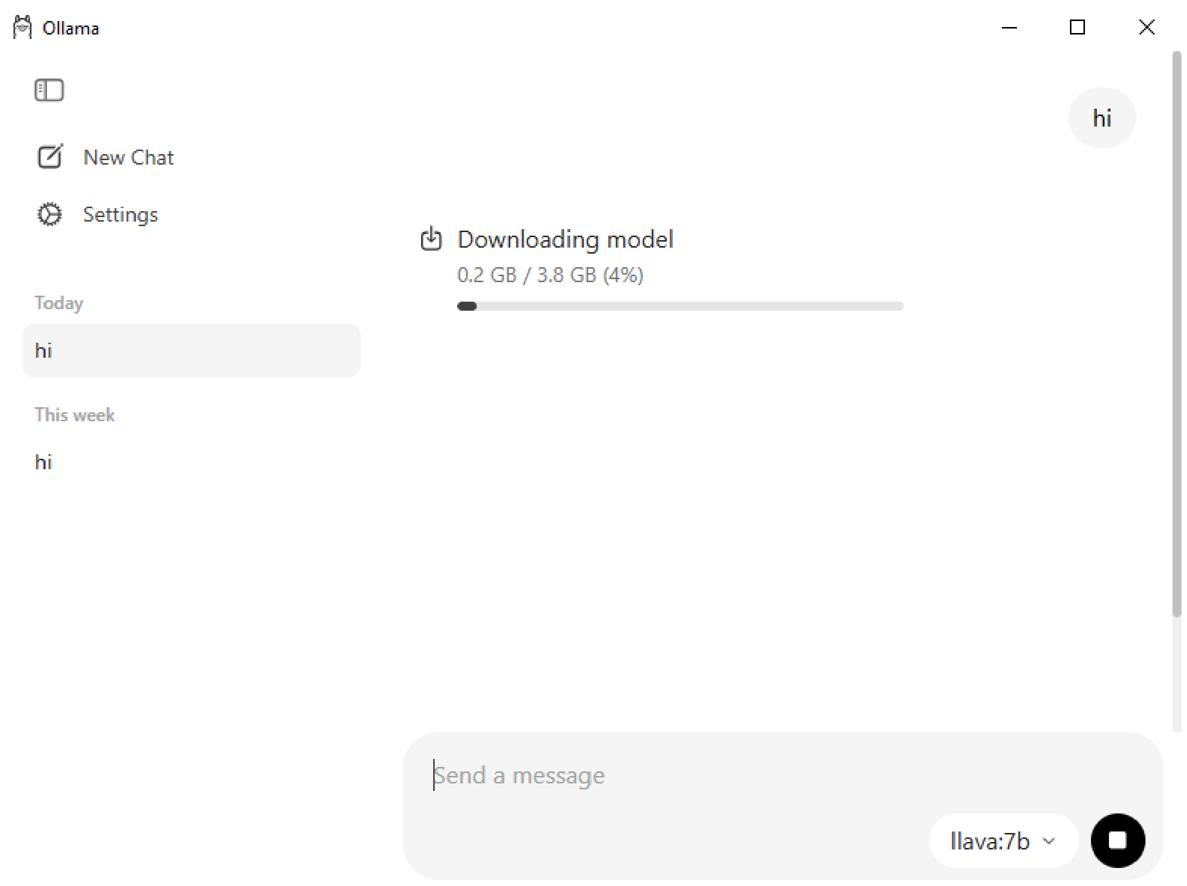
3.5.4 3. Vision Talk with Ollama

In this demo, the Pi camera takes a snapshot **each time you type a question**. The program sends **your typed text + the new photo** to a local vision model via Ollama, and then streams the model's reply in plain English. This is a minimal "see & tell" baseline you can later extend with color/face/QR checks.

Before You Start

1. Open the **Ollama** app (or run the service) and make sure a **vision-capable model** is pulled.

   - If you have enough memory (16GB RAM), you may try `llava:7b`.
   - If you only have **8GB RAM**, prefer a smaller model such as `moondream:1.8b` or `granite3.2-vision:2b`.



Downloading model
0.2 GB / 3.8 GB (4%)

Run the Demo

1. Go to the example folder and run the script:

```
cd ~/picar-x/example
python3 17.text_vision_talk.py
```

2. What happens when it runs:

   - The program prints a welcome line and waits for your input (>>>).
   - **Every time you type anything** (e.g., "hello", "Is there yellow?", "Any faces?", "What is on the desk?"), it:
     - **captures a photo** from the Pi camera (saved to `/tmp/llm-img.jpg`),
     - **sends your text + the photo** to the vision model via Ollama,
     - **streams back** the model's answer to the terminal.
   - Type `exit` or `quit` to end the program.

Code

```
from picarx.llm import Ollama
from picamera2 import Picamera2
import time

"""
You need to set up Ollama first.
```

(continues on next page)

(continued from previous page)

```
Note: At least 8GB RAM is recommended for small vision models (e.g., moondream:1.8b).
      For llava:7b, more memory is preferred (16GB).
"""

INSTRUCTIONS = "You are a helpful assistant."
WELCOME = "Hello, I am a helpful assistant. How can I help you?"

# If Ollama runs on the same Pi, use "localhost".
# If it runs on another computer in your LAN, replace with that computer's IP.
llm = Ollama(
    ip="localhost",           # e.g., "192.168.100.145" if remote
    model="llava:7b"          # change to "moondream:1.8b" or "granite3.2-vision:2b" for␣
↪8GB RAM
)

# Basic configuration
llm.set_max_messages(20)
llm.set_instructions(INSTRUCTIONS)
llm.set_welcome(WELCOME)

# Init camera
camera = Picamera2()
config = camera.create_still_configuration(
    main={"size": (1280, 720)},
)
camera.configure(config)
camera.start()
time.sleep(2)

print(WELCOME)

while True:
    input_text = input(">>> ")
    if input_text.strip().lower() in {"exit", "quit"}:
        break

    # Capture image
    img_path = "/tmp/llm-img.jpg"
    camera.capture_file(img_path)

    # Response with stream (text + image)
    response = llm.prompt(input_text, stream=True, image_path=img_path)
    for next_word in response:
        if next_word:
            print(next_word, end="", flush=True)
    print("")
```

3.5.5 Troubleshooting

- **I get an error like: `model requires more system memory ...`.**
  - This means the model is too large for your device.
  - Use a smaller model such as `moondream:1.8b` or `granite3.2-vision:2b`.
  - Or switch to a machine with more RAM and expose Ollama to the network.
- **The code cannot connect to Ollama (connection refused).**

  Check the following:
  - Make sure Ollama is running (`ollama serve` or the desktop app is open).
  - If using a remote computer, enable **Expose to network** in Ollama settings.
  - Double-check that the `ip="..."` in your code matches the correct LAN IP.
  - Confirm both devices are on the same local network.
- **My Pi camera does not capture anything.**
  - Verify that `Picamera2` is installed and working with a simple test script.
  - Check that the camera cable is properly connected and enabled in `raspi-config`.
  - Ensure your script has permission to write to the target path (`/tmp/llm-img.jpg`).
- **The output is too slow.**
  - Smaller models reply faster, but with simpler answers.
  - You can lower the camera resolution (e.g., 640×480 instead of 1280×720) to speed up image processing.
  - Close other programs on your Pi to free up CPU and RAM.

3.6 18. Connecting to Online LLMs

In this lesson, we'll learn how to connect your PiCar-X (or Raspberry Pi) to different **online Large Language Models (LLMs)**. Each provider requires an API key and offers different models you can choose from.

We'll cover how to:

- Create and save your API keys safely.
- Pick a model that fits your needs.
- Run our example code to chat with the models.

Let's go step by step for each provider.

3.6.1 Before You Start

Make sure you've completed:

- *5. Install All the Modules(Important)* — Install `robot-hat`, `vilib`, `picar-x` modules, then run the script `i2samp.sh`.

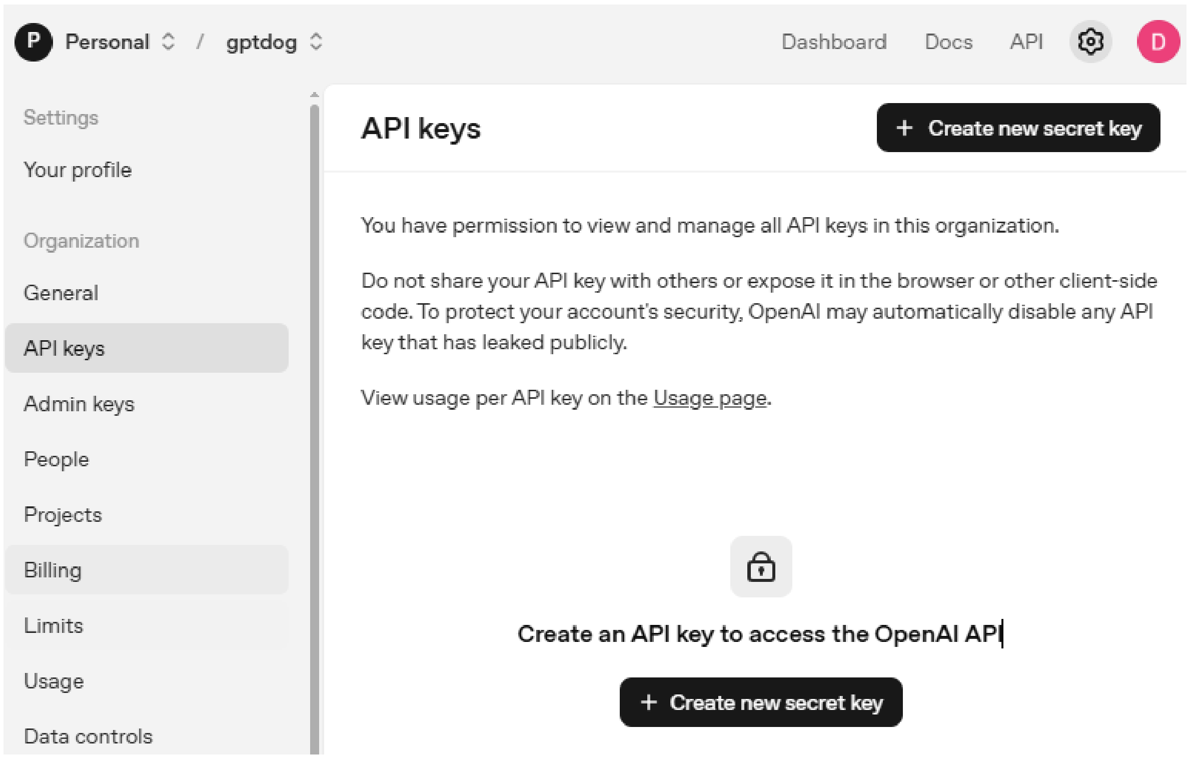
3.6.2 OpenAI

OpenAI provides powerful models like **GPT-4o** and **GPT-4.1** that can be used for both text and vision tasks.

Here's how to set it up:

Get and Save your API Key

1. Go to and log in. On the **API keys** page, click **Create new secret key**.



API keys

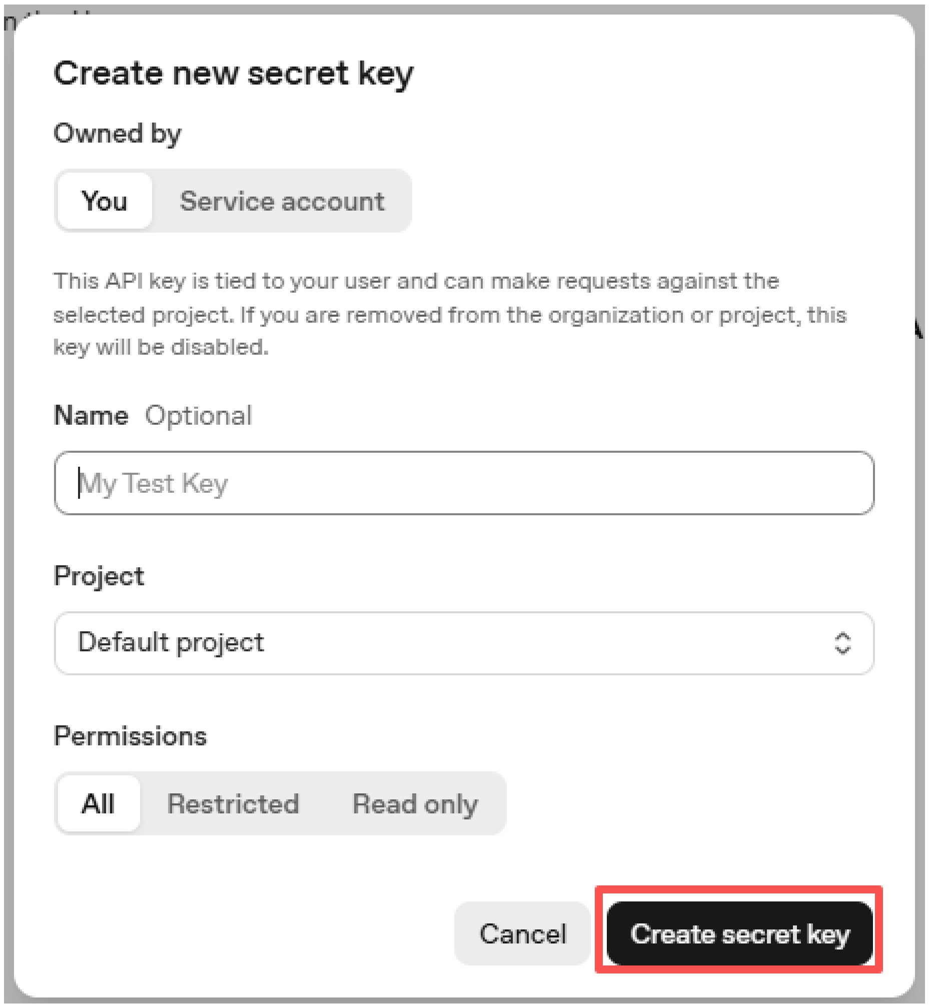
You have permission to view and manage all API keys in this organization.

Do not share your API key with others or expose it in the browser or other client-side code. To protect your account's security, OpenAI may automatically disable any API key that has leaked publicly.

View usage per API key on the Usage page.

Create an API key to access the OpenAI API

2. Fill in the details (Owner, Name, Project, and permissions if needed), then click **Create secret key**.



Create new secret key

Owned by

You | Service account

This API key is tied to your user and can make requests against the selected project. If you are removed from the organization or project, this key will be disabled.

Name Optional

My Test Key

Project

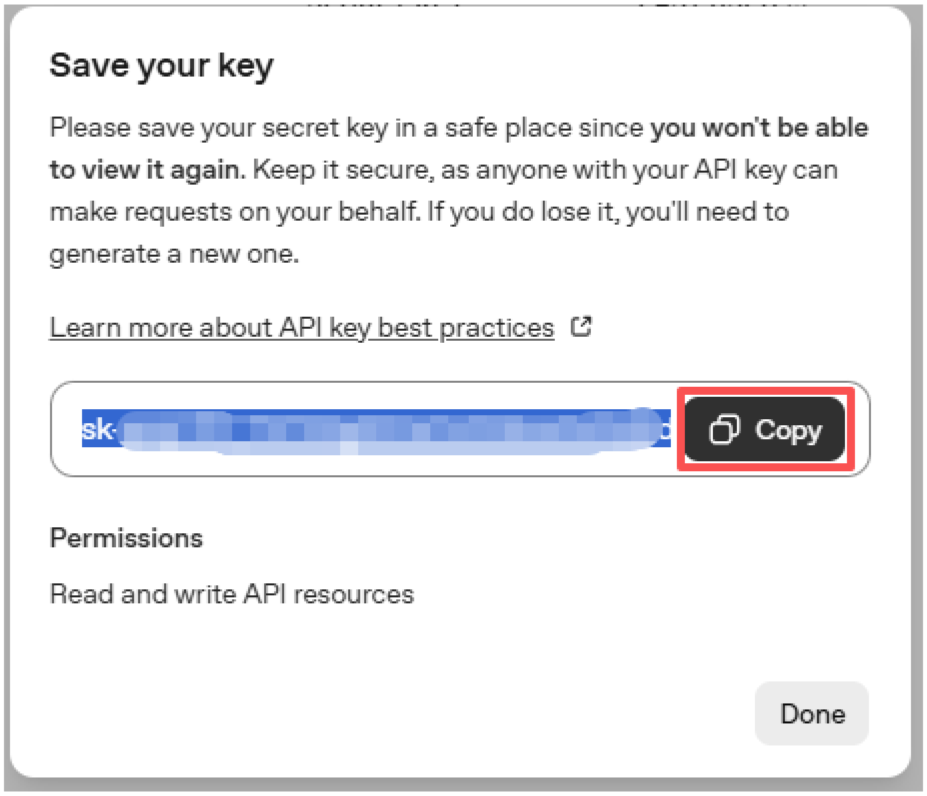
Default project

Permissions

All | Restricted | Read only

Cancel | Create secret key

3. Once the key is created, copy it right away — you won't be able to see it again. If you lose it, you'll need to generate a new one.



Save your key

Please save your secret key in a safe place since **you won't be able to view it again**. Keep it secure, as anyone with your API key can make requests on your behalf. If you do lose it, you'll need to generate a new one.

Learn more about API key best practices

sk-[redacted]

Permissions

Read and write API resources

4. In your project folder (for example: `/picar-x/example`), create a file called `secret.py`:

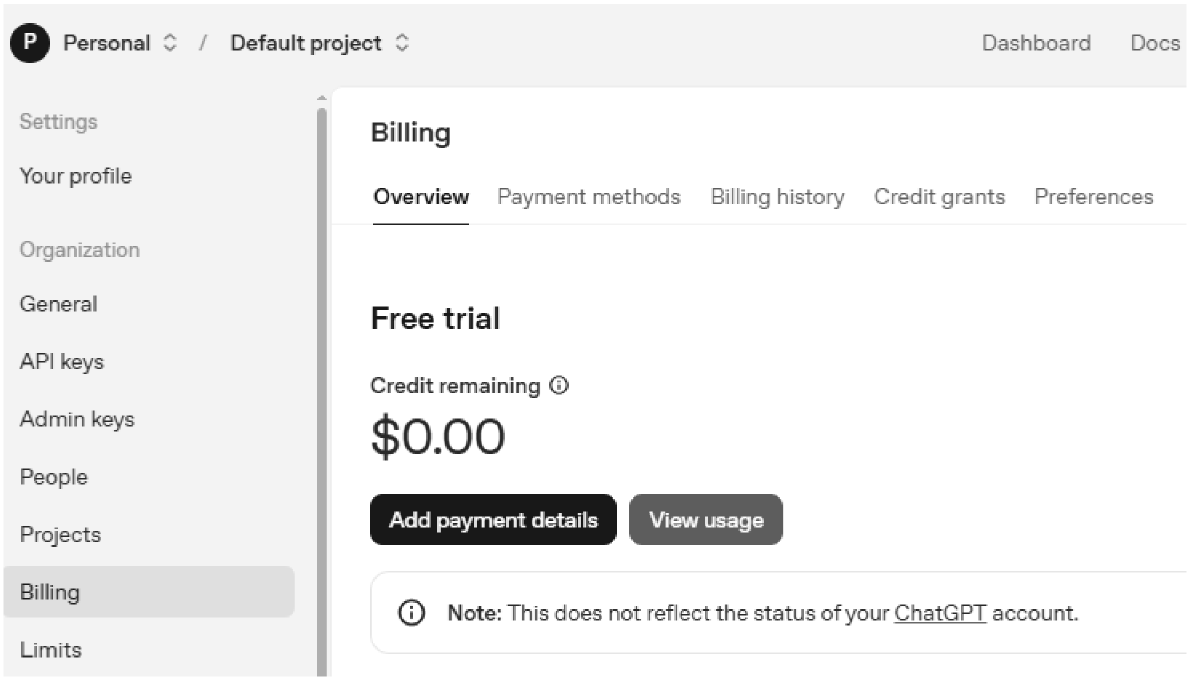
```
cd ~/picar-x/example
sudo nano secret.py
```

5. Paste your key into the file like this:

```
# secret.py
# Store secrets here. Never commit this file to Git.
OPENAI_API_KEY = "sk-xxx"
```

Enable billing and check models

1. Before using the key, go to the **Billing** page in your OpenAI account, add your payment details, and top up a small amount of credits.



Billing

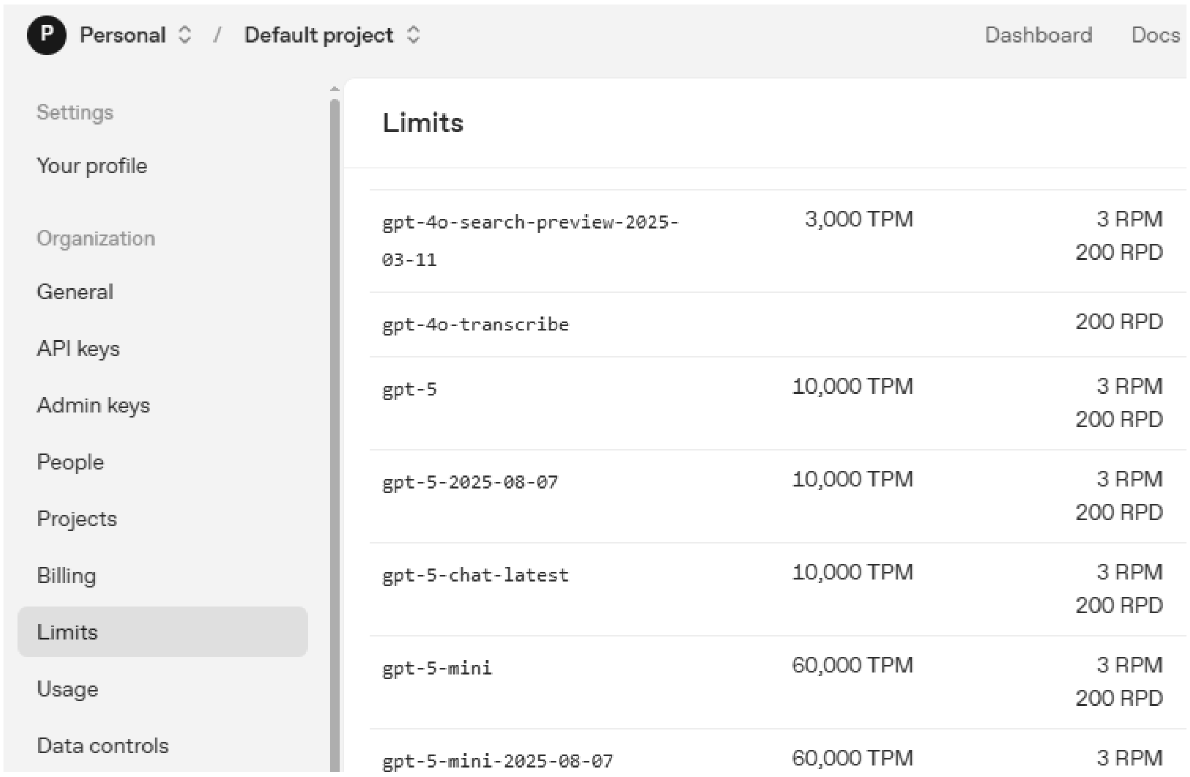
Free trial

Credit remaining

$0.00

Note: This does not reflect the status of your ChatGPT account.

2. Then go to the **Limits** page to check which models are available for your account and copy the exact model ID to use in your code.



Limits

gpt-4o-search-preview-2025-03-11	3,000 TPM	3 RPM 200 RPD
gpt-4o-transcribe		200 RPD
gpt-5	10,000 TPM	3 RPM 200 RPD
gpt-5-2025-08-07	10,000 TPM	3 RPM 200 RPD
gpt-5-chat-latest	10,000 TPM	3 RPM 200 RPD
gpt-5-mini	60,000 TPM	3 RPM 200 RPD
gpt-5-mini-2025-08-07	60,000 TPM	3 RPM

Test with example code

1. Open sample code:

```
cd ~/picar-x/example
sudo nano 18.online_llm_test.py
```

2. Replace the content with the code below, and update `model="xxx"` to the model you want (for example, `gpt-4o`):

```python
from picarx.llm import OpenAI
from secret import OPENAI_API_KEY

INSTRUCTIONS = "You are a helpful assistant."
WELCOME = "Hello, I am a helpful assistant. How can I help you?"

llm = OpenAI(
    api_key=OPENAI_API_KEY,
    model="gpt-4o",
)
```

   Save and exit (`Ctrl+X`, then `Y`, then `Enter`).

3. Finally, run the test:

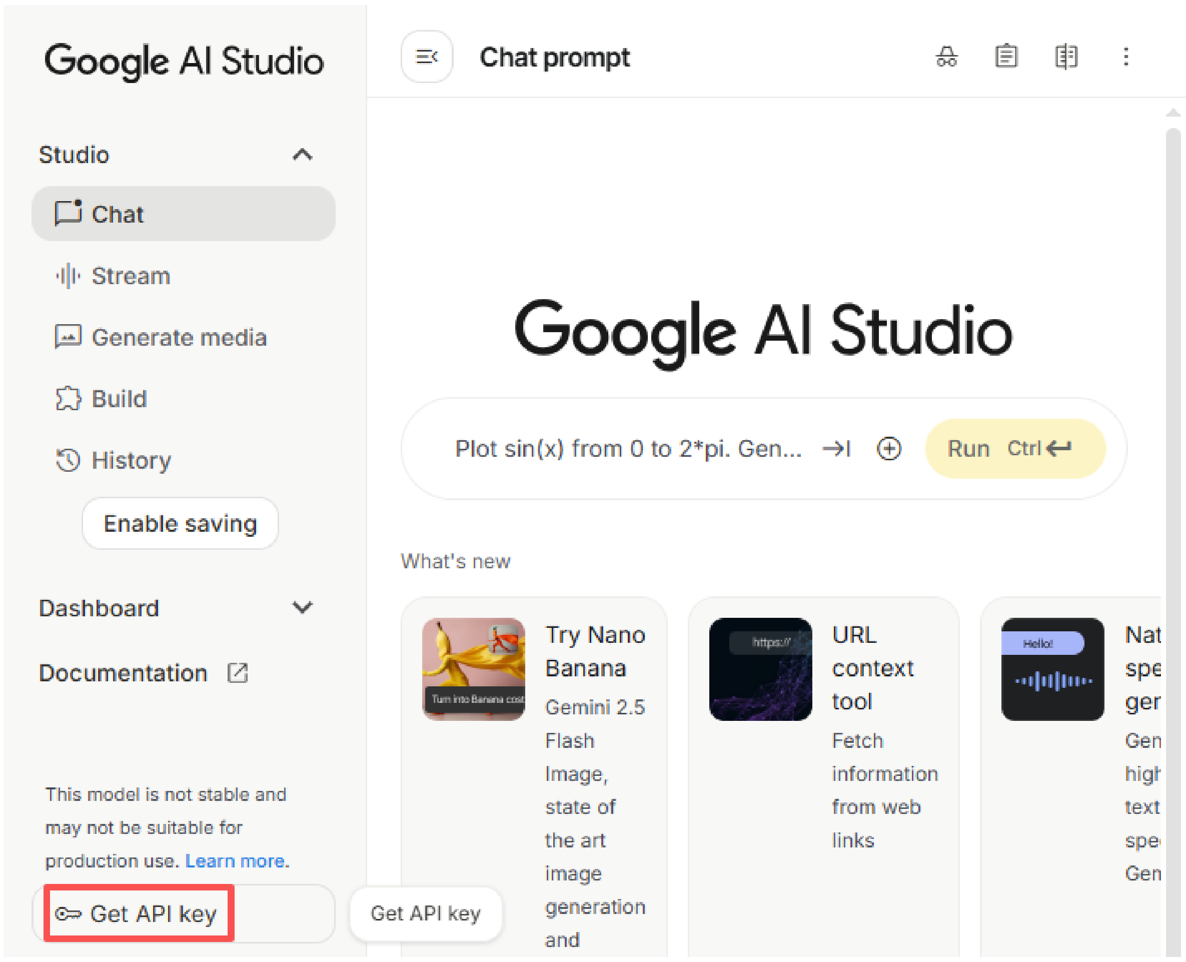
```
sudo python3 18.online_llm_test.py
```

3.6.3 Gemini

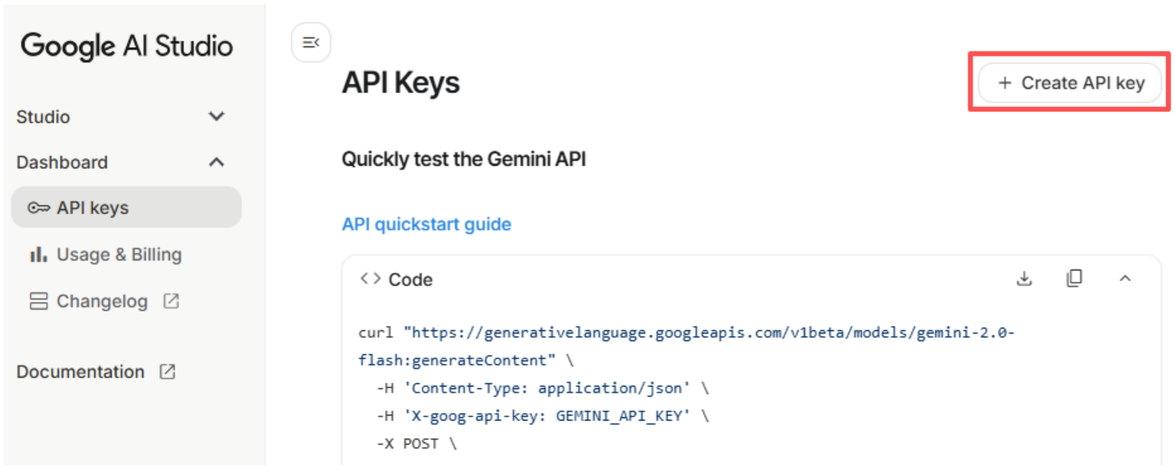
Gemini is Google's family of AI models. It's fast and great for general-purpose tasks.

Get and Save your API Key

1. Log in to , then go to the API Keys page.

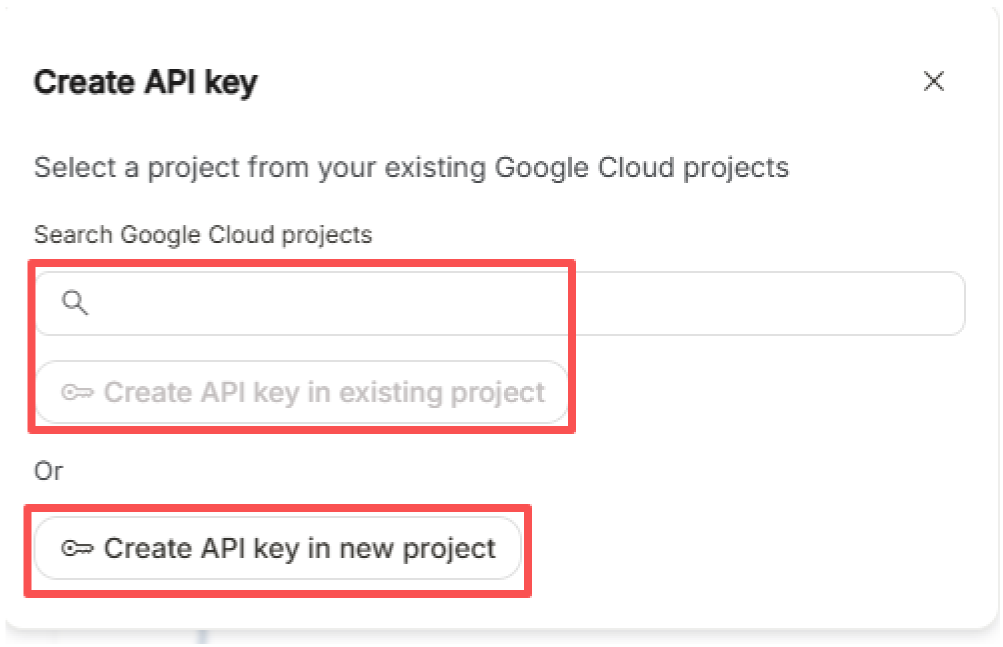

Google AI Studio
Studio
Chat
Stream
Generate media
Build
History
Enable saving
Dashboard
Documentation
This model is not stable and may not be suitable for production use. Learn more.
Get API key
Get API key
Chat prompt
Google AI Studio
Plot sin(x) from 0 to 2*pi. Gen...
Run Ctrl
What's new
Try Nano Banana
Gemini 2.5 Flash Image, state of the art image generation and
URL context tool
Fetch information from web links

2. Click the **Create API key** button in the top-right corner.



```
curl "https://generativelanguage.googleapis.com/v1beta/models/gemini-2.0-
flash:generateContent" \
  -H 'Content-Type: application/json' \
  -H 'X-goog-api-key: GEMINI_API_KEY' \
  -X POST \
```

3. You can create a key for an existing project or a new one.



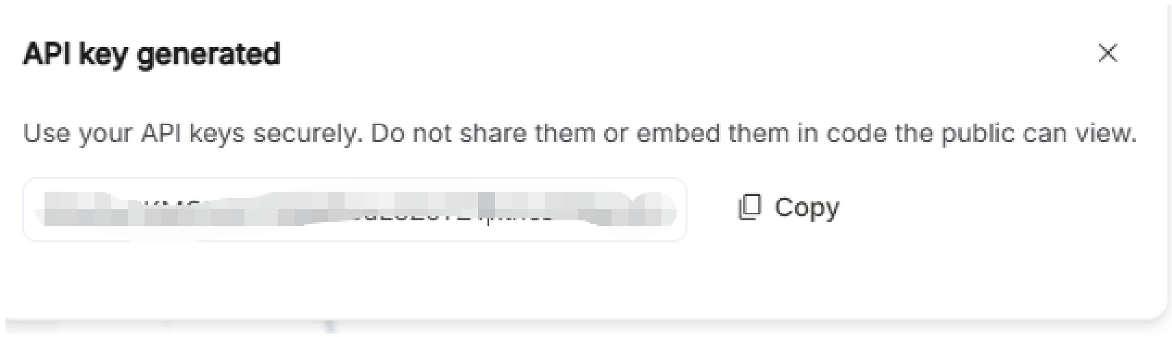
Create API key

Select a project from your existing Google Cloud projects

Search Google Cloud projects

Create API key in existing project
Or
Create API key in new project

4. Copy the generated API key.



API key generated

Use your API keys securely. Do not share them or embed them in code the public can view.

[redacted]

5. In your project folder:

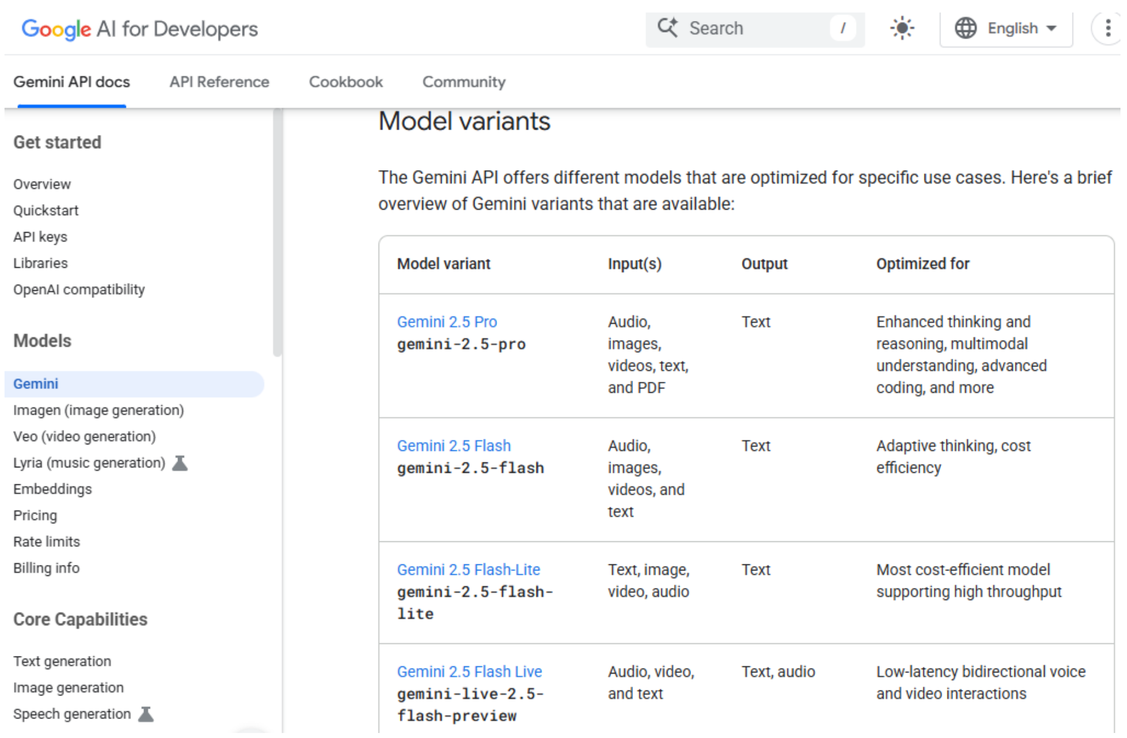
```
cd ~/picar-x/example
sudo nano secret.py
```

6. Paste the key:

```
# secret.py
# Store secrets here. Never commit this file to Git.
GEMINI_API_KEY = "AIxxx"
```

Check available models

Go to the official page, here you'll see the list of models, their exact API IDs, and which use case each one is optimized for.



Model variants

The Gemini API offers different models that are optimized for specific use cases. Here's a brief overview of Gemini variants that are available:

Model variant	Input(s)	Output	Optimized for
Gemini 2.5 Pro `gemini-2.5-pro`	Audio, images, videos, text, and PDF	Text	Enhanced thinking and reasoning, multimodal understanding, advanced coding, and more
Gemini 2.5 Flash `gemini-2.5-flash`	Audio, images, videos, and text	Text	Adaptive thinking, cost efficiency
Gemini 2.5 Flash-Lite `gemini-2.5-flash-lite`	Text, image, video, audio	Text	Most cost-efficient model supporting high throughput
Gemini 2.5 Flash Live `gemini-live-2.5-flash-preview`	Audio, video, and text	Text, audio	Low-latency bidirectional voice and video interactions

Test with example code

1. Open the test file:

```
cd ~/picar-x/example
sudo nano 18.online_llm_test.py
```

2. Replace the content with the code below, and update `model="xxx"` to the model you want (for example, `gemini-2.5-flash`):

```
from picarx.llm import Gemini
from secret import GEMINI_API_KEY

INSTRUCTIONS = "You are a helpful assistant."
WELCOME = "Hello, I am a helpful assistant. How can I help you?"

llm = Gemini(
    api_key=GEMINI_API_KEY,
    model="gemini-2.5-flash",
)
```

3. Save and run:

```
sudo python3 18.online_llm_test.py
```

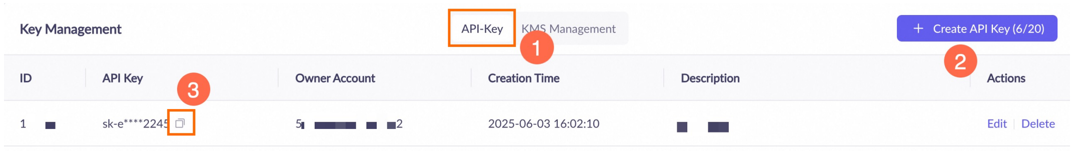
3.6.4 Qwen

Qwen is a family of large language and multimodal models provided by Alibaba Cloud. These models support text generation, reasoning, and multimodal understanding (such as image analysis).

Get an API Key

To call Qwen models, you need an **API Key**. Most international users should use the **DashScope International (Model Studio)** console. Mainland China users can instead use the **Bailian ()** console.

- **For International Users**
  1. Go to the official page on **Alibaba Cloud**.
  2. Sign in or create an **Alibaba Cloud** account.
  3. Navigate to **Model Studio** (choose Singapore or Beijing region).
     - If an "Activate Now" prompt appears at the top of the page, click it to activate Model Studio and receive the free quota (Singapore only).
     - Activation is free — you will only be charged after your free quota is used.
     - If no activation prompt appears, the service is already active.
  4. Go to the **Key Management** page. On the **API Key** tab, click **Create API Key**.
  5. After creation, copy your API Key and keep it safe.

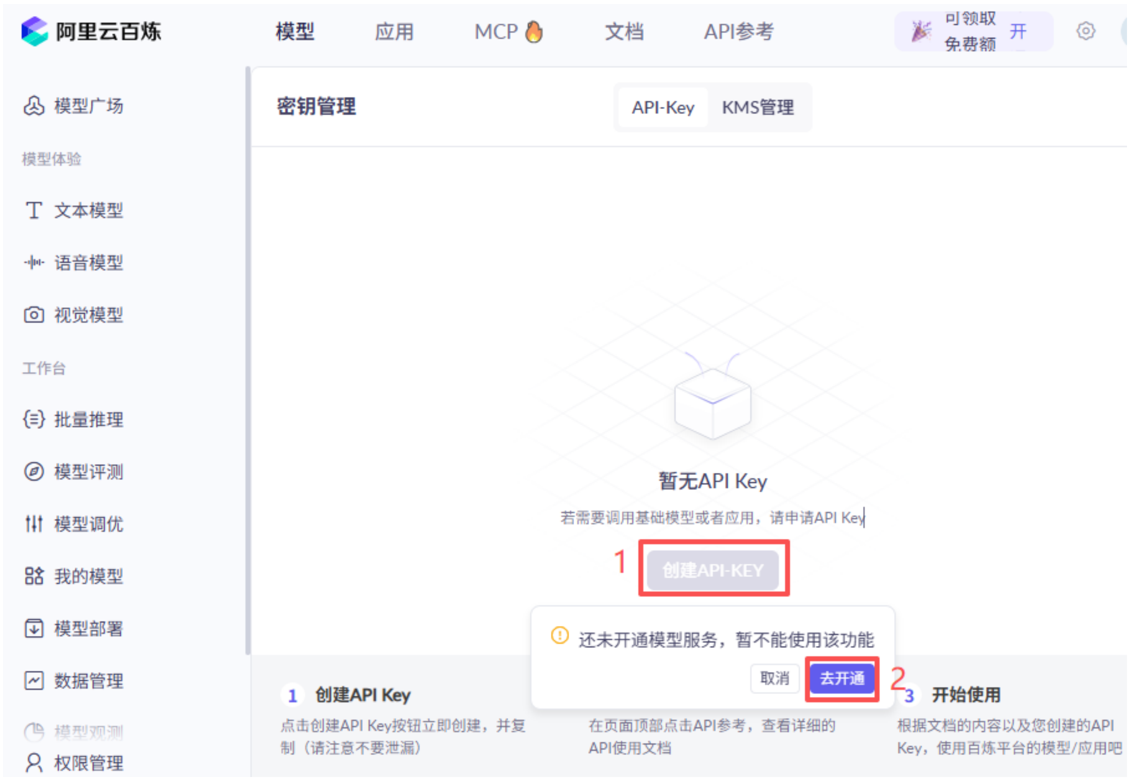


ID	API Key	Owner Account	Creation Time	Description	Actions
1 [redacted]	sk-e****2245	5[redacted]2	2025-06-03 16:02:10	[redacted]	Edit | Delete

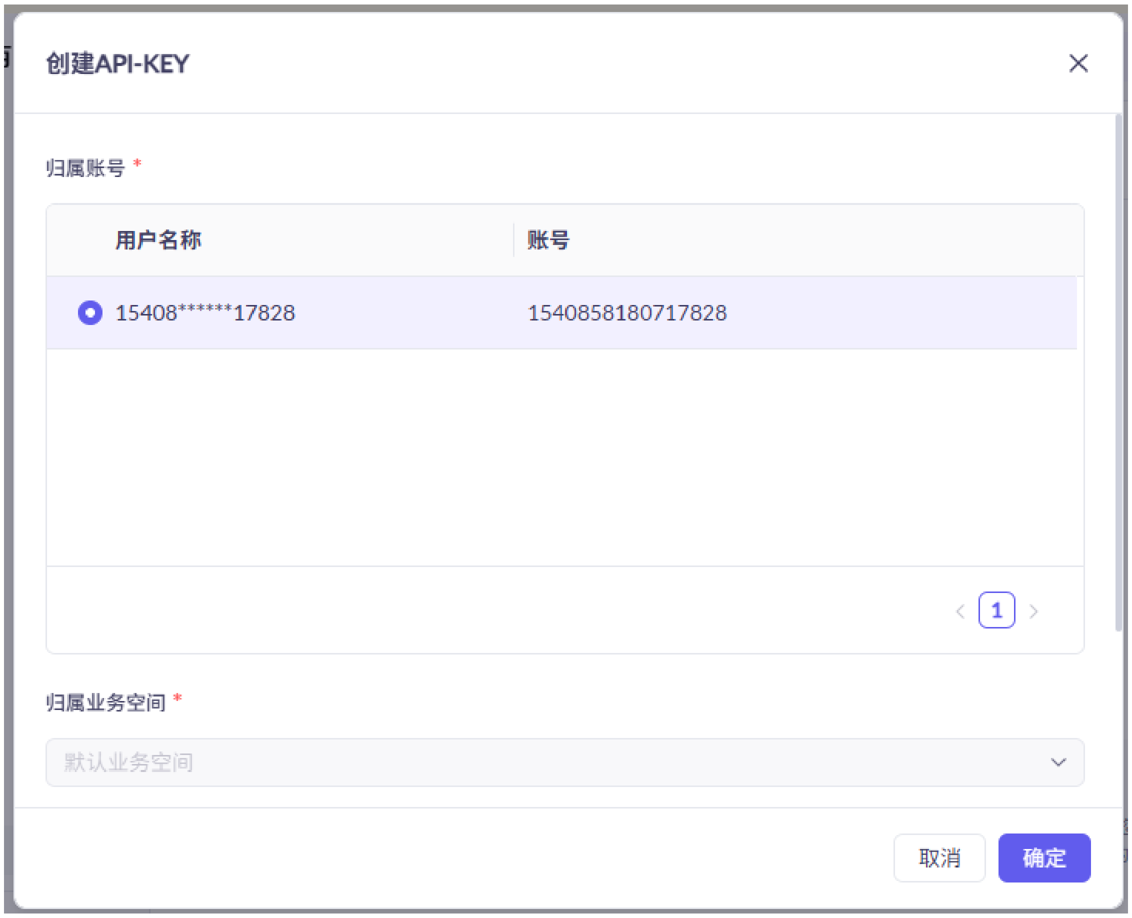
Note: Users in Hong Kong, Macau, and Taiwan should also choose the **International (Model Studio)** option.

- **For Mainland China Users**

  If you are in Mainland China, you can use the **Alibaba Cloud Bailian ()** console instead:
  1. Log in to (Bailian console) and complete account verification.
  2. Select **Create API Key**. If prompted that model services are not activated, click **Activate**, agree to the terms, and claim your free quota. After activation, the **Create API Key** button will be enabled.



3. Click **Create API Key** again, check your account, and then click **Confirm**.



创建API-KEY

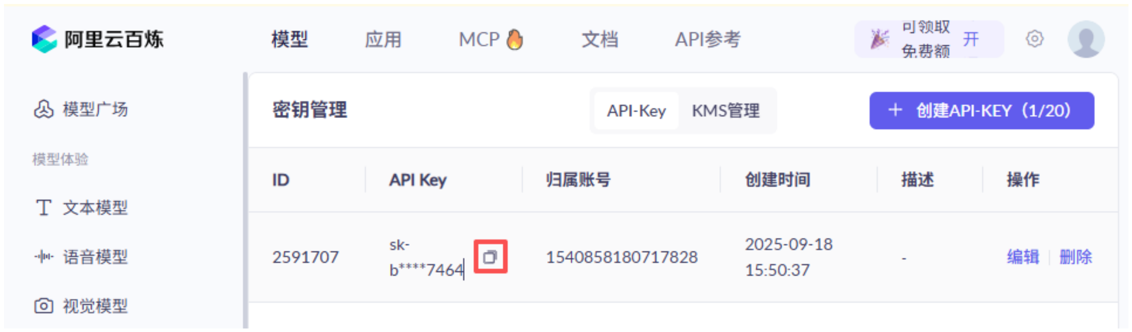
归属账号 *

用户名称	账号
● 15408******17828	1540858180717828

归属业务空间 *

默认业务空间

4. Once created, copy your API Key.



ID	API Key	归属账号	创建时间	描述	操作
2591707	sk-b****7464	1540858180717828	2025-09-18 15:50:37	-	编辑 | 删除

Save your API Key

1. In your project folder:

```
cd ~/picar-x/example
sudo nano secret.py
```

2. Paste your key like this:

```
# secret.py
# Store secrets here. Never commit this file to Git.

QWEN_API_KEY = "sk-xxx"
```

Test with example code

1. Open the test file:

```
cd ~/picar-x/example
sudo nano 18.online_llm_test.py
```

2. Replace the content with the code below, and update `model="xxx"` to the model you want (for example, `qwen-plus`):

```
from picarx.llm import Qwen
from secret import QWEN_API_KEY

INSTRUCTIONS = "You are a helpful assistant."
WELCOME = "Hello, I am a helpful assistant. How can I help you?"

llm = Qwen(
    api_key=QWEN_API_KEY,
    model="qwen-plus",
)
```

3. Run with:

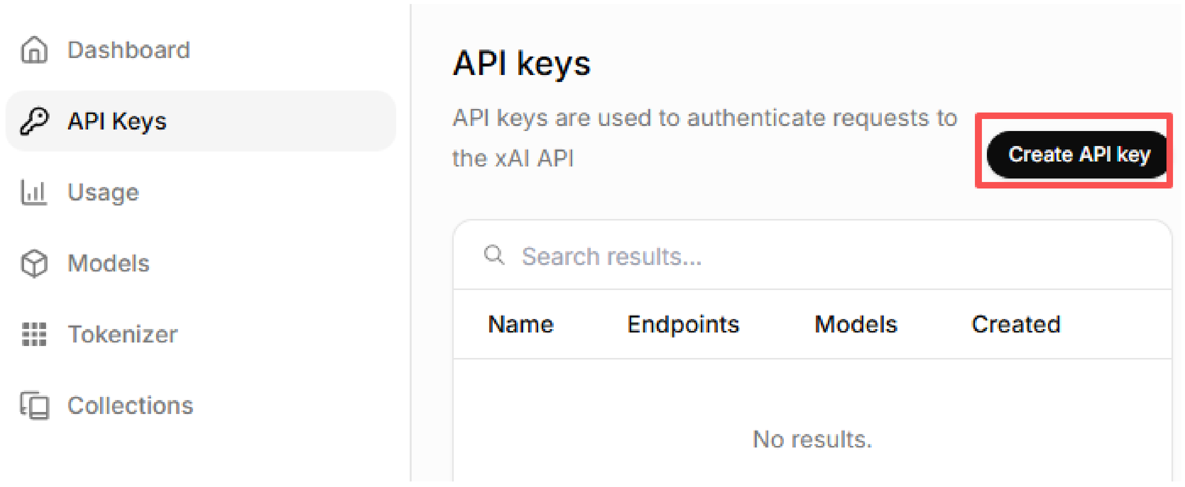
```
sudo python3 18.online_llm_test.py
```

3.6.5 Grok (xAI)

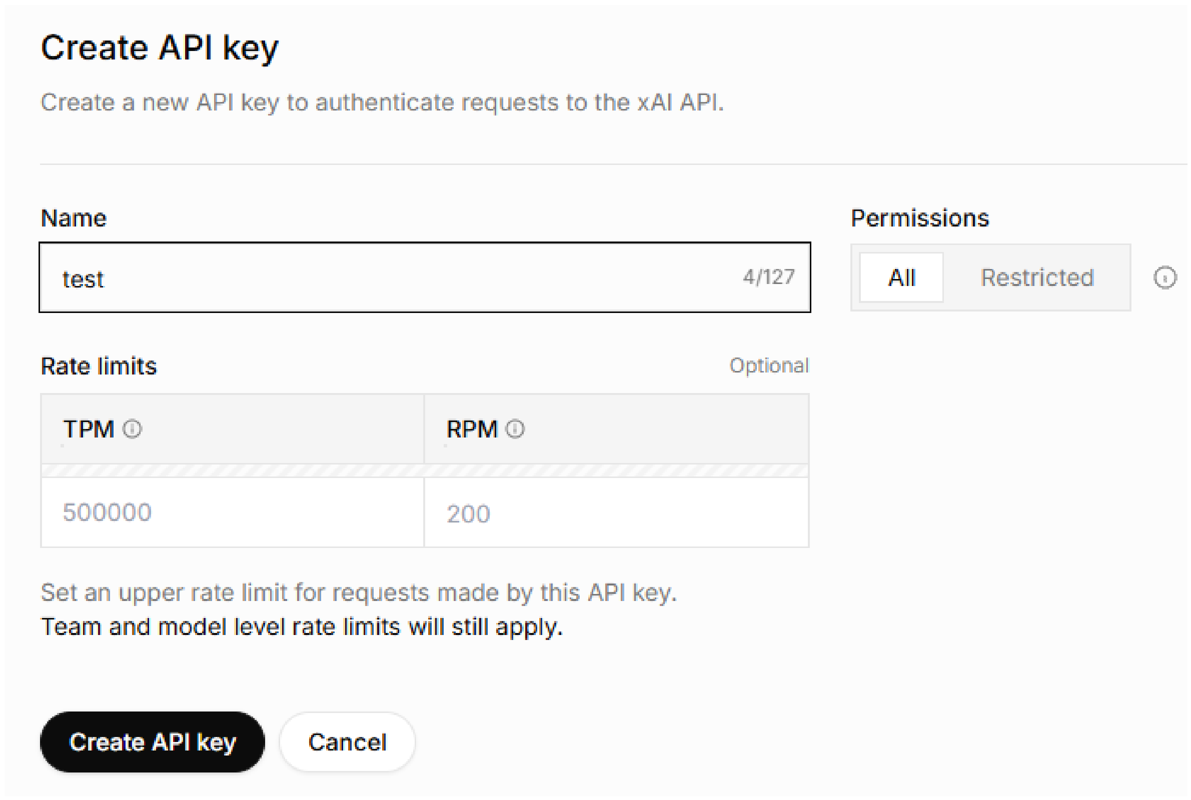
Grok is xAI's conversational AI, created by Elon Musk's team. You can connect to it through the xAI API.

Get and Save your API Key

1. Sign up for an account here: . Add some credits to your account first — otherwise the API won't work.
2. Go to the API Keys page, click **Create API key**.



3. Enter a name for the key, then click **Create API key**.



Create API key

Create a new API key to authenticate requests to the xAI API.

Name: test 4/127

Permissions: All Restricted

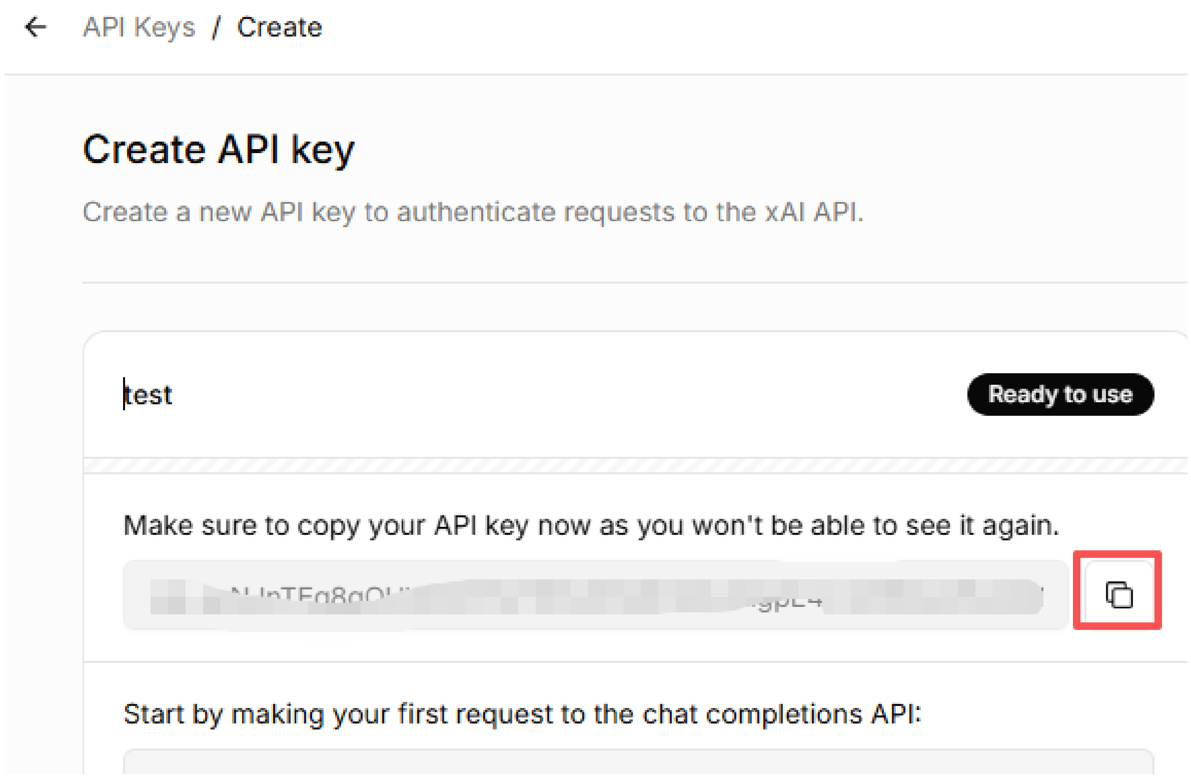
Rate limits Optional

TPM	RPM
500000	200

Set an upper rate limit for requests made by this API key.
Team and model level rate limits will still apply.

Create API key Cancel

4. Copy the generated key and keep it safe.



← API Keys / Create

Create API key

Create a new API key to authenticate requests to the xAI API.

test Ready to use

Make sure to copy your API key now as you won't be able to see it again.

[redacted]

Start by making your first request to the chat completions API:

5. In your project folder:

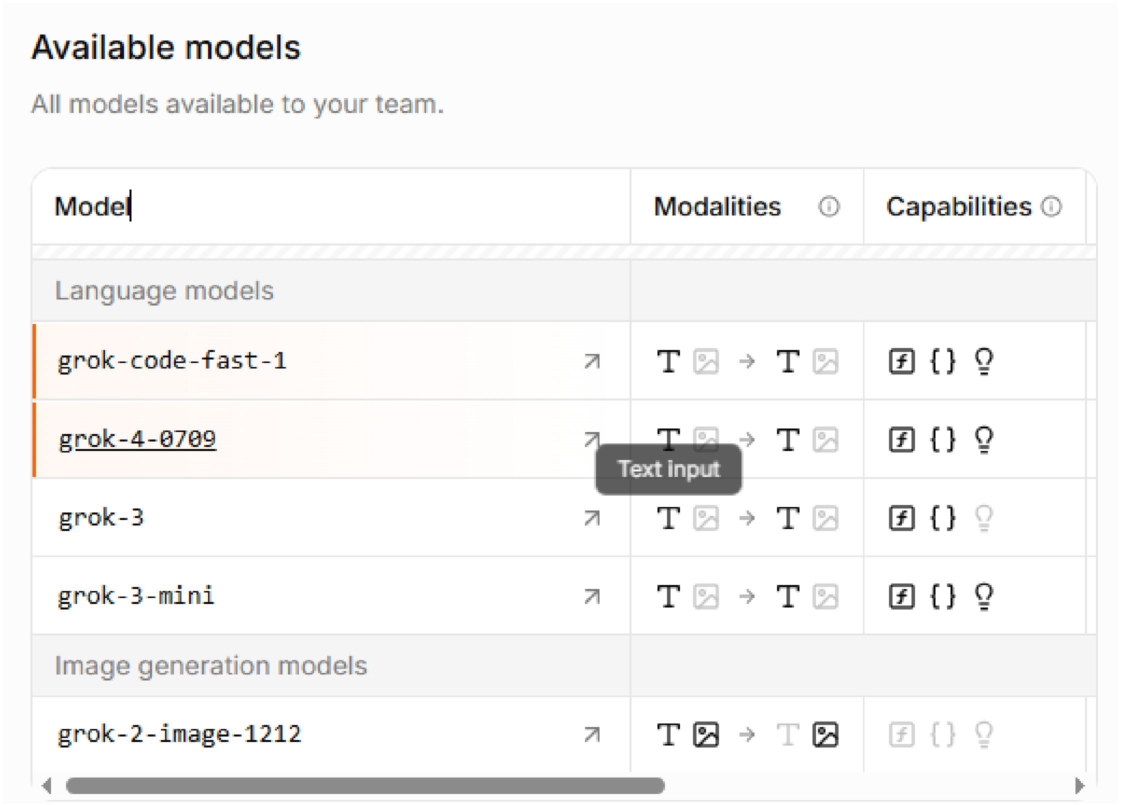
```
cd ~/picar-x/example
sudo nano secret.py
```

6. Paste your key like this:

```
# secret.py
# Store secrets here. Never commit this file to Git.

GROK_API_KEY = "xai-xxx"
```

Check available models

Go to the Models page in the xAI console. Here you can see all the models available to your team, along with their exact API IDs — use these IDs in your code.



Available models

All models available to your team.

Model	Modalities	Capabilities
Language models		
grok-code-fast-1	T ▨ → T ▨	ƒ {} 💡
grok-4-0709	T ▨ → T ▨	ƒ {} 💡
grok-3	T ▨ → T ▨	ƒ {}
grok-3-mini	T ▨ → T ▨	ƒ {} 💡
Image generation models		
grok-2-image-1212	T ▨ → ▨	

Test with example code

1. Open the test file:

```
cd ~/picar-x/example
sudo nano 18.online_llm_test.py
```

2. Replace the content with the code below, and update `model="xxx"` to the model you want (for example, `grok-4-latest`):

```
from picarx.llm import Grok
from secret import GROK_API_KEY

INSTRUCTIONS = "You are a helpful assistant."
```

(continues on next page)

(continued from previous page)

```
WELCOME = "Hello, I am a helpful assistant. How can I help you?"

llm = Grok(
    api_key=GROK_API_KEY,
    model="grok-4-latest",
)
```

3. Run with:

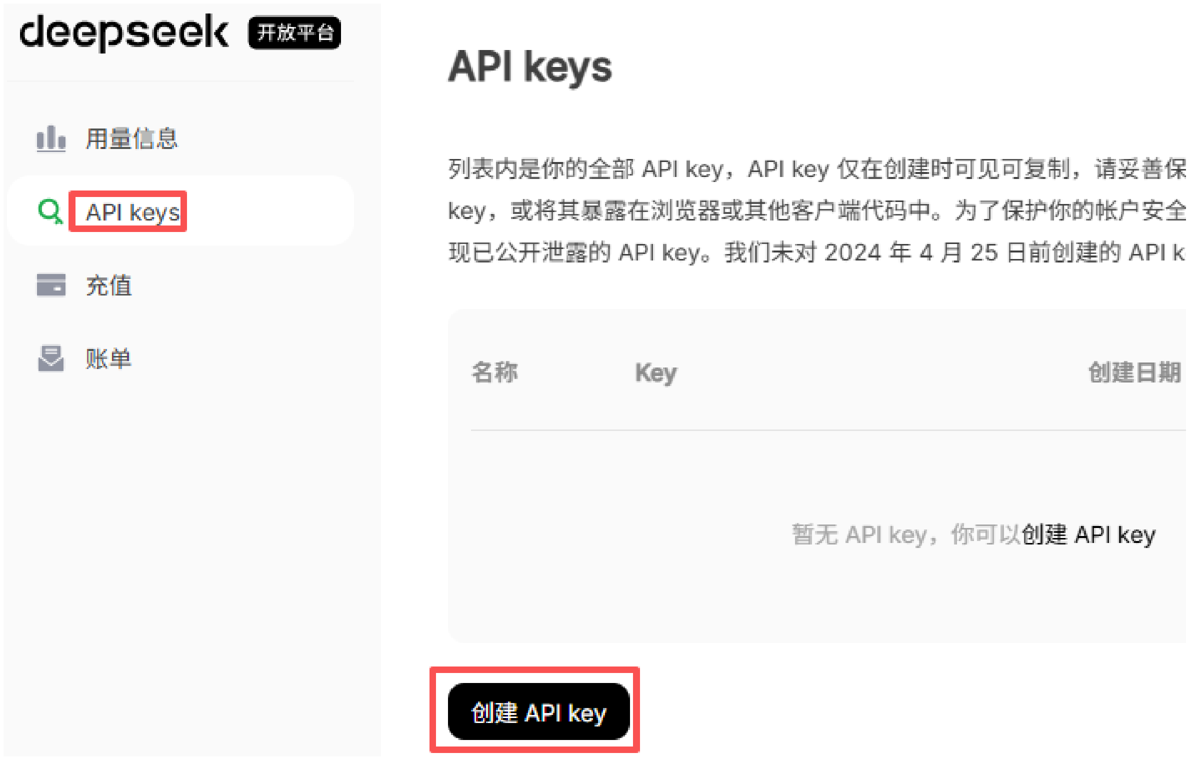
```
sudo python3 18.online_llm_test.py
```

3.6.6 DeepSeek

DeepSeek is a Chinese LLM provider that offers affordable and capable models.

Get and Save your API Key

1. Log in to .
2. In the top-right menu, select **API Keys** → **Create API Key**.

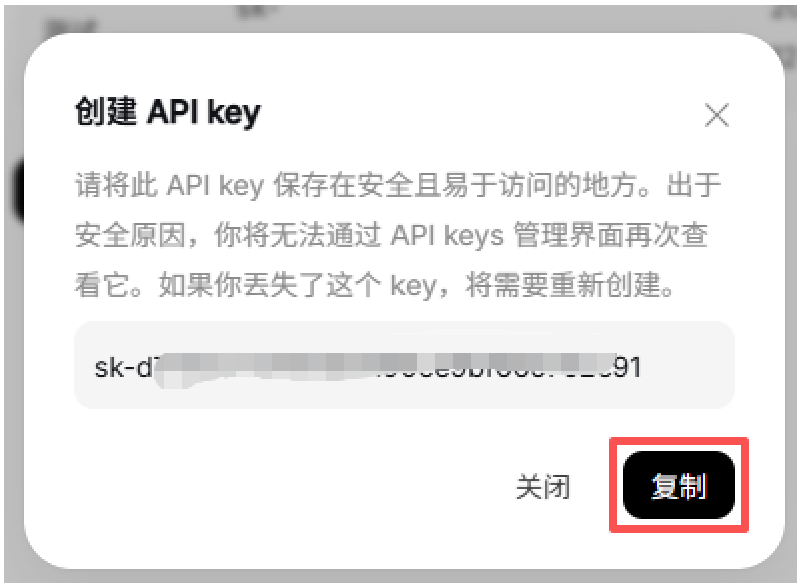


列表内是你的全部 API key，API key 仅在创建时可见可复制，请妥善保
key，或将其暴露在浏览器或其他客户端代码中。为了保护你的帐户安全
现已公开泄露的 API key。我们未对 2024 年 4 月 25 日前创建的 API k

名称	Key	创建日期

暂无 API key，你可以创建 API key

3. Enter a name, click **Create**, then copy the key.



创建 API key

请将此 API key 保存在安全且易于访问的地方。出于安全原因，你将无法通过 API keys 管理界面再次查看它。如果你丢失了这个 key，将需要重新创建。

sk-d[redacted]91

关闭 复制

4. In your project folder:

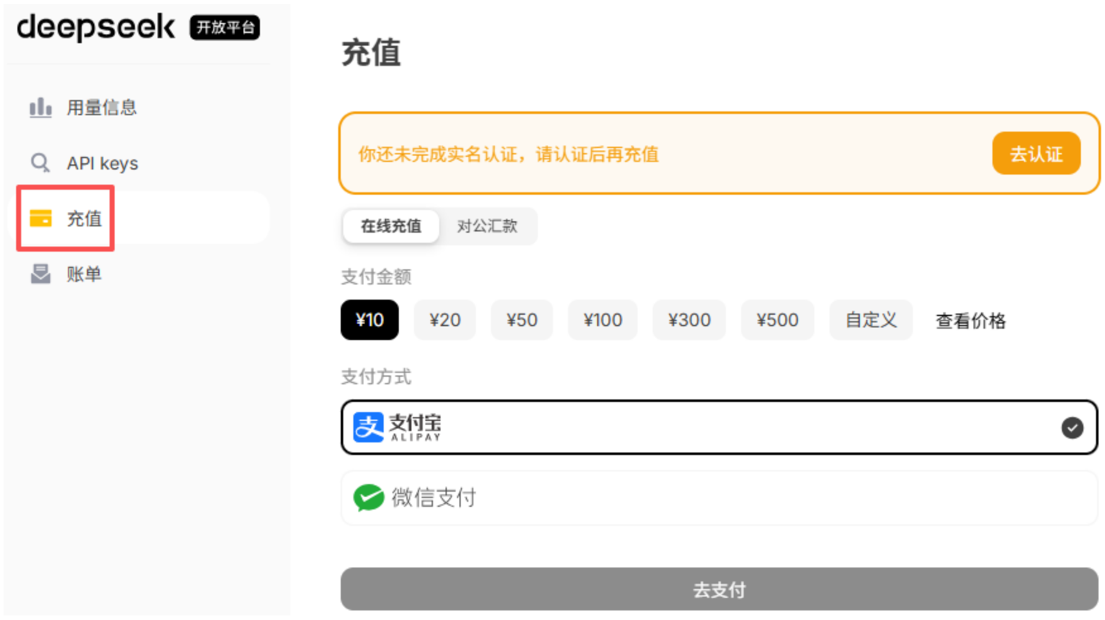
```
cd ~/picar-x/example
sudo nano secret.py
```

5. Add your key:

```
# secret.py
DEEPSEEK_API_KEY = "sk-xxx"
```

Enable billing

You'll need to recharge your account first. Start with a small amount (like ¥10 RMB).



deepseek 开放平台

用量信息

API keys

充值

账单

充值

你还未完成实名认证，请认证后再充值 去认证

在线充值 对公汇款

支付金额

¥10 ¥20 ¥50 ¥100 ¥300 ¥500 自定义 查看价格

支付方式

支付宝 ALIPAY

微信支付

去支付

Available models

At the time of writing (2025-09-12), DeepSeek offers:

- `deepseek-chat`
- `deepseek-reasoner`

Test with example code

1. Open the test file:

```
cd ~/picar-x/example
sudo nano 18.online_llm_test.py
```

2. Replace the content with the code below, and update `model="xxx"` to the model you want (for example, `deepseek-chat`):

```python
from picarx.llm import Deepseek
from secret import DEEPSEEK_API_KEY

INSTRUCTIONS = "You are a helpful assistant."
WELCOME = "Hello, I am a helpful assistant. How can I help you?"

llm = Deepseek(
    api_key=DEEPSEEK_API_KEY,
    model="deepseek-chat",
    max_messages=20,
)
```

3. Run:

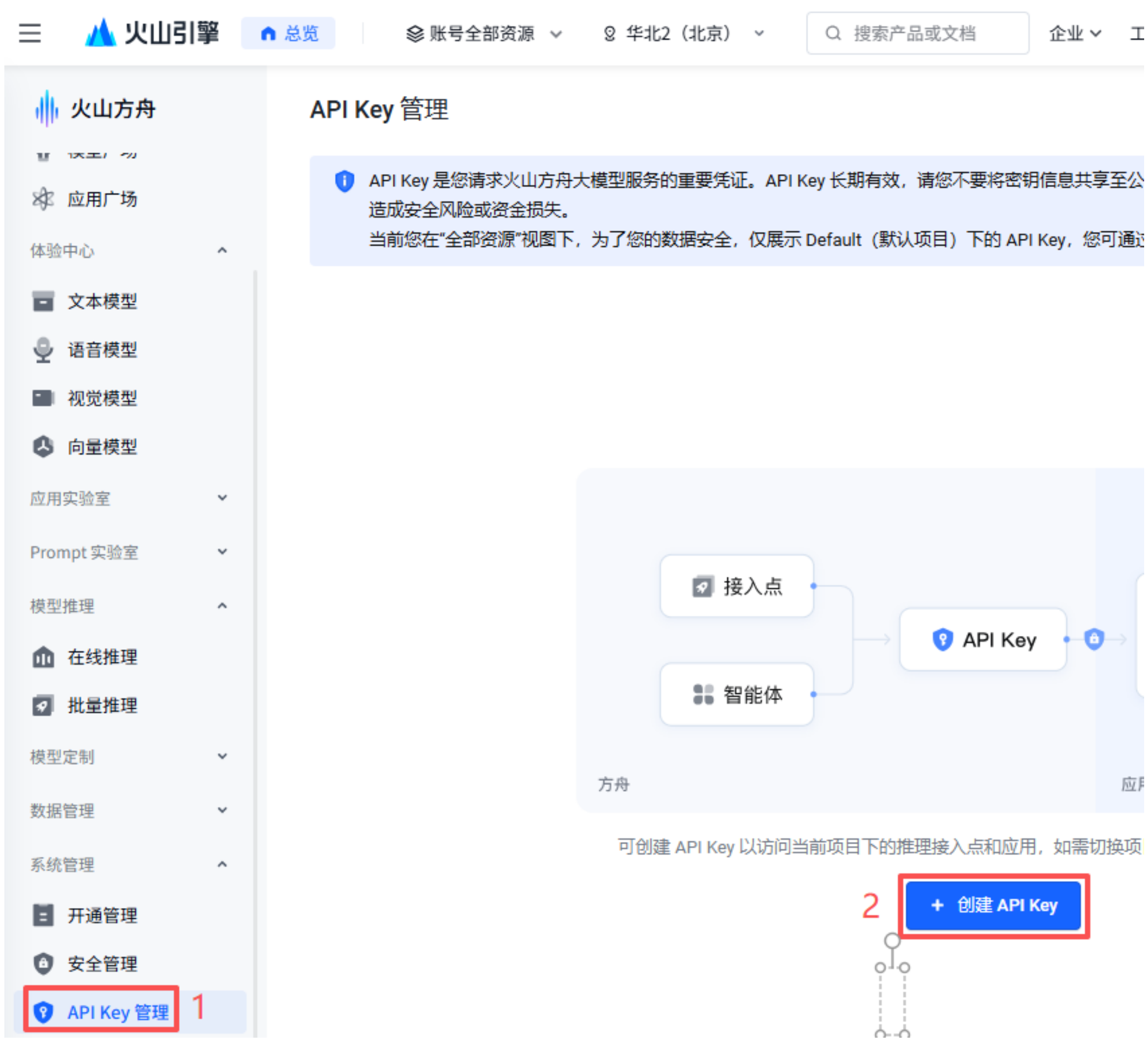
```
sudo python3 18.online_llm_test.py
```

3.6.7 Doubao

Doubao is ByteDance's AI model platform (Volcengine Ark).

Get and Save your API Key

1. Log in to .
2. In the left menu, scroll down to **API Key Management** → **Create API Key**.

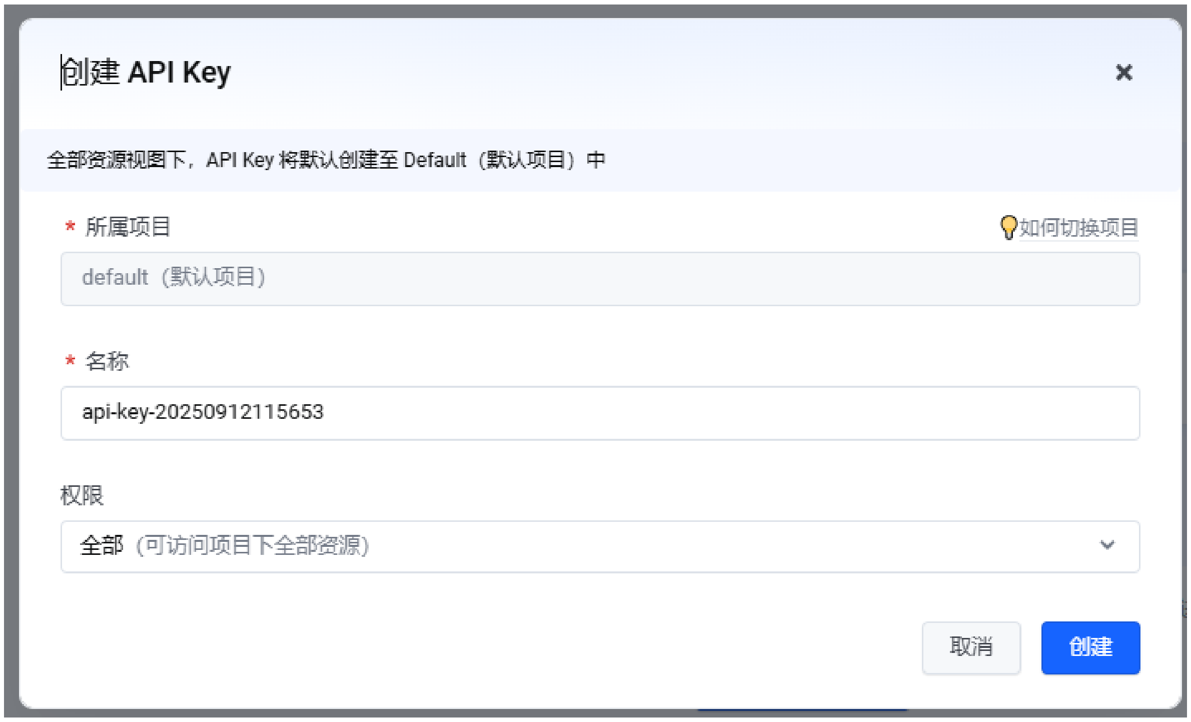


API Key 管理

API Key 是您请求火山方舟大模型服务的重要凭证。API Key 长期有效，请您不要将密钥信息共享至公
造成安全风险或资金损失。
当前您在"全部资源"视图下，为了您的数据安全，仅展示 Default（默认项目）下的 API Key，您可通

可创建 API Key 以访问当前项目下的推理接入点和应用，如需切换项

3. Choose a name and click **Create**.



创建 API Key

全部资源视图下，API Key 将默认创建至 Default（默认项目）中

* 所属项目

default（默认项目）

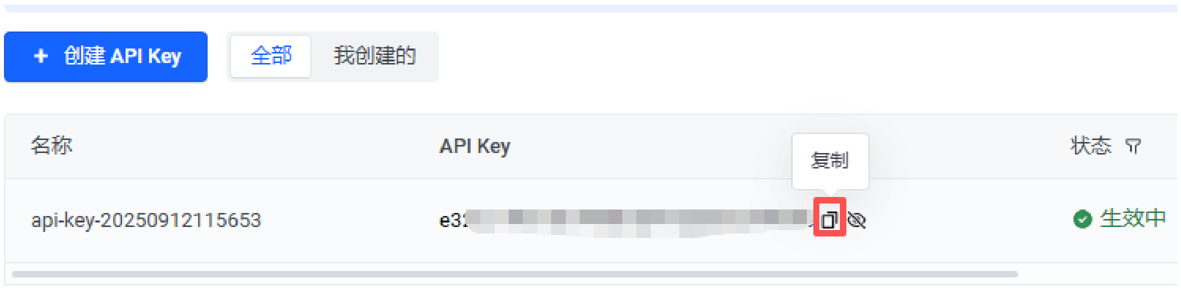
* 名称

api-key-20250912115653

权限

全部（可访问项目下全部资源）

4. Click the **Show API Key** icon and copy it.



名称	API Key	状态
api-key-20250912115653	e3[redacted]	生效中

5. In your project folder:

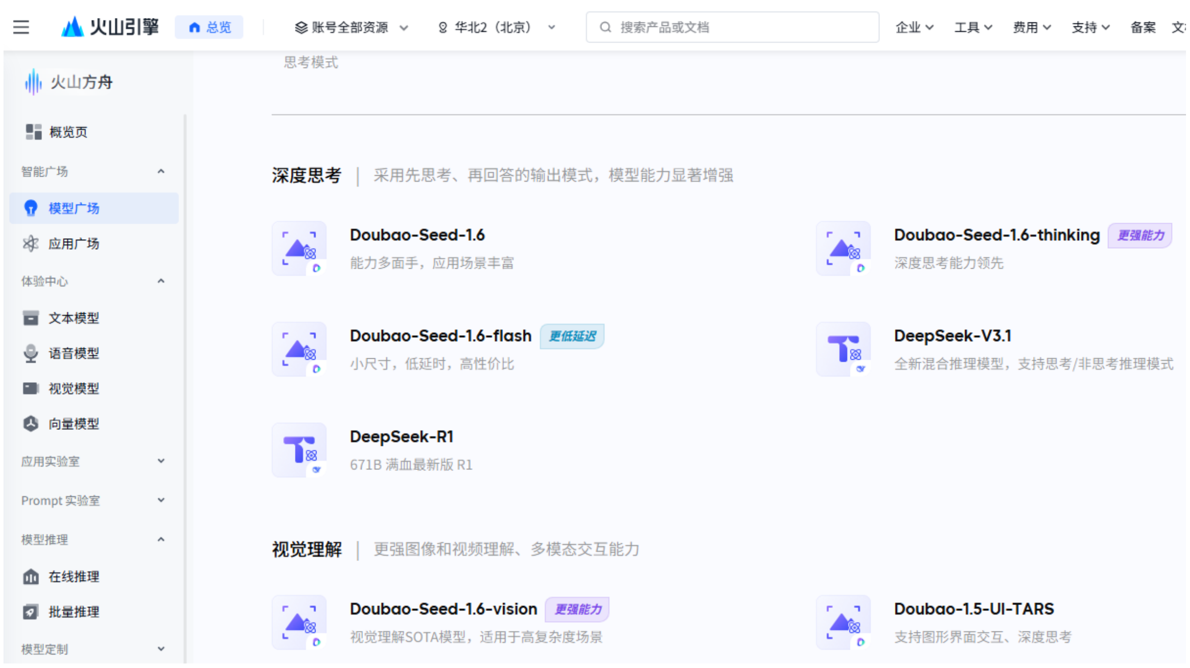
```
cd ~/picar-x/example
sudo nano secret.py
```

6. Add your key:

```
# secret.py
DOUBAO_API_KEY = "xxx"
```

Choose a model

1. Go to the model marketplace and pick a model.



火山引擎 总览 账号全部资源 华北2（北京） 搜索产品或文档 企业 工具 费用 支持 备案 文

火山方舟

概览页

智能广场

模型广场

应用广场

体验中心

文本模型

语音模型

视觉模型

向量模型

应用实验室

Prompt 实验室

模型推理

在线推理

批量推理

模型定制

思考模式

深度思考 | 采用先思考、再回答的输出模式，模型能力显著增强

Doubao-Seed-1.6
能力多面手，应用场景丰富

Doubao-Seed-1.6-thinking 更强能力
深度思考能力领先

Doubao-Seed-1.6-flash 更低延迟
小尺寸，低延时，高性价比

DeepSeek-V3.1
全新混合推理模型，支持思考/非思考推理模式

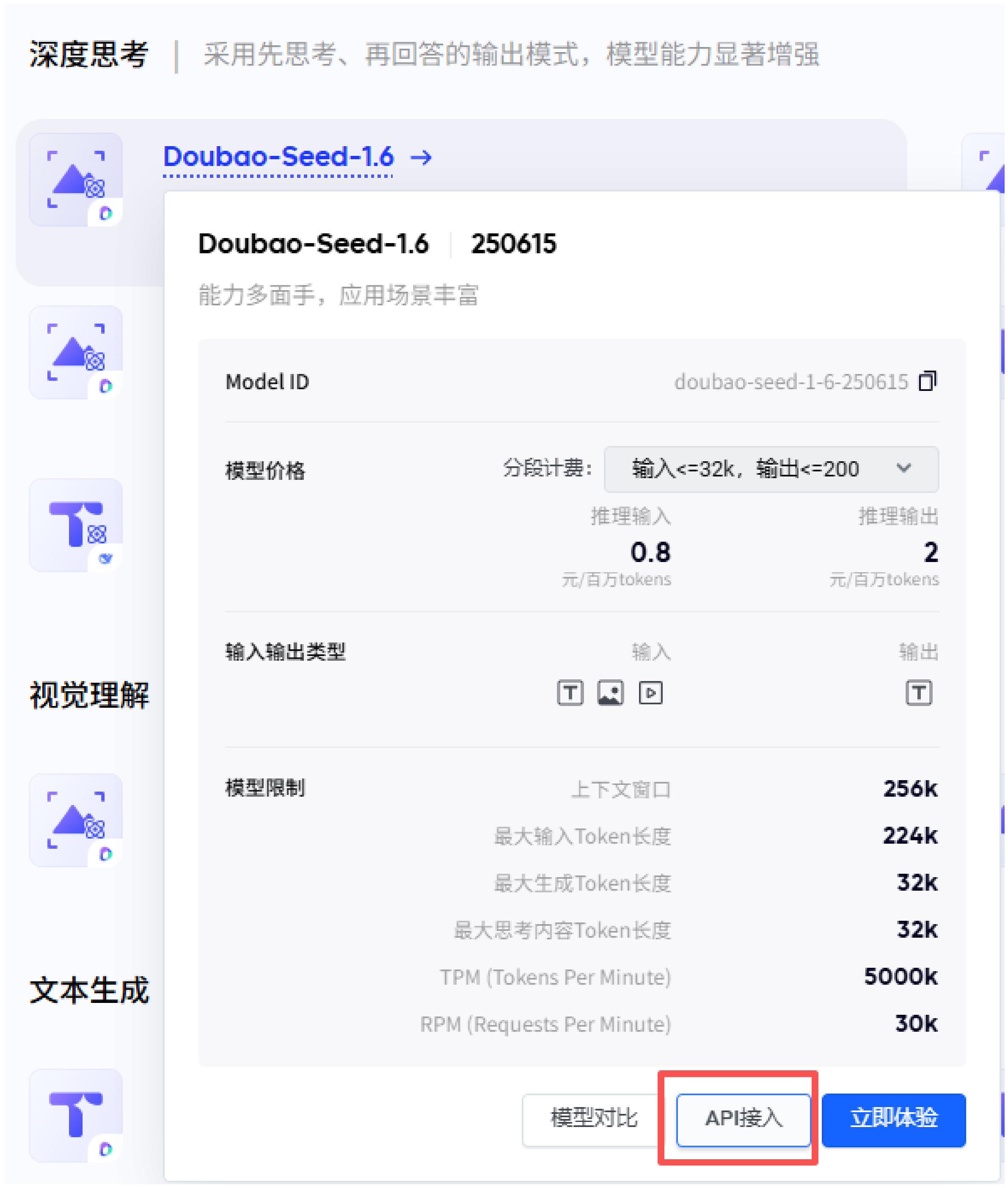
DeepSeek-R1
671B 满血最新版 R1

视觉理解 | 更强图像和视频理解、多模态交互能力

Doubao-Seed-1.6-vision 更强能力
视觉理解SOTA模型，适用于高复杂度场景

Doubao-1.5-UI-TARS
支持图形界面交互、深度思考

2. For example, choose **Doubao-seed-1.6**, then click **API** .



深度思考 | 采用先思考、再回答的输出模式，模型能力显著增强

视觉理解

文本生成

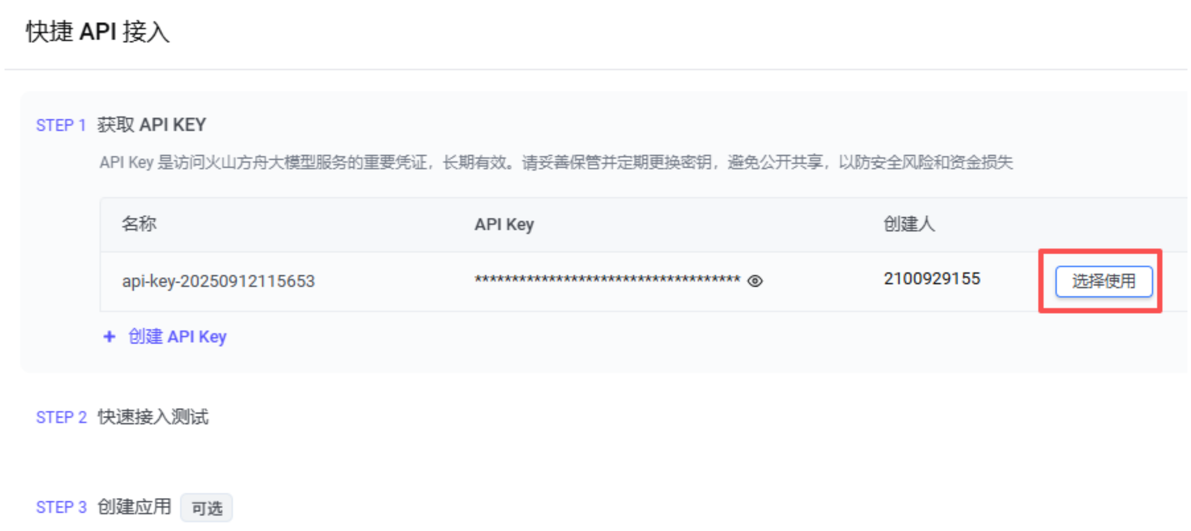
Doubao-Seed-1.6 →

Doubao-Seed-1.6 | 250615

能力多面手，应用场景丰富

Model ID		doubao-seed-1-6-250615
模型价格	分段计费：	输入<=32k，输出<=200
	推理输入	推理输出
	0.8 元/百万tokens	2 元/百万tokens
输入输出类型	输入	输出
模型限制	上下文窗口	256k
	最大输入Token长度	224k
	最大生成Token长度	32k
	最大思考内容Token长度	32k
	TPM (Tokens Per Minute)	5000k
	RPM (Requests Per Minute)	30k

3. Select your API Key and click **Use API**.


快捷 API 接入

STEP 1 获取 API KEY

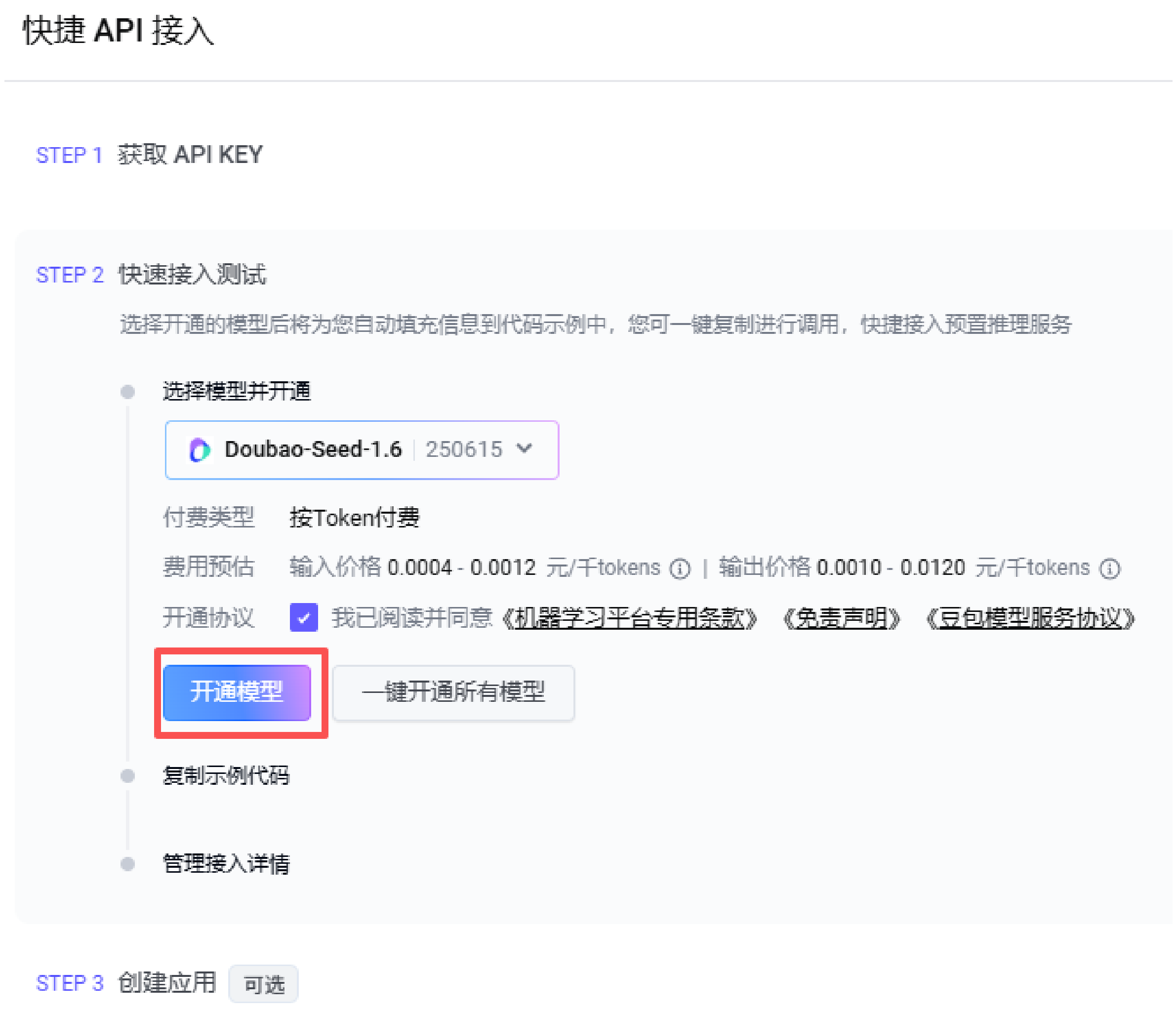
API Key 是访问火山方舟大模型服务的重要凭证，长期有效。请妥善保管并定期更换密钥，避免公开共享，以防安全风险和资金损失

名称	API Key	创建人	
api-key-20250912115653	************************************	2100929155	选择使用

STEP 2 快速接入测试

STEP 3 创建应用 可选

4. Click **Enable Model**.


快捷 API 接入

STEP 1 获取 API KEY

STEP 2 快速接入测试

选择开通的模型后将为您自动填充信息到代码示例中，您可一键复制进行调用，快捷接入预置推理服务

选择模型并开通

Doubao-Seed-1.6 | 250615

付费类型 按Token付费

费用预估 输入价格 0.0004 - 0.0012 元/千tokens | 输出价格 0.0010 - 0.0120 元/千tokens

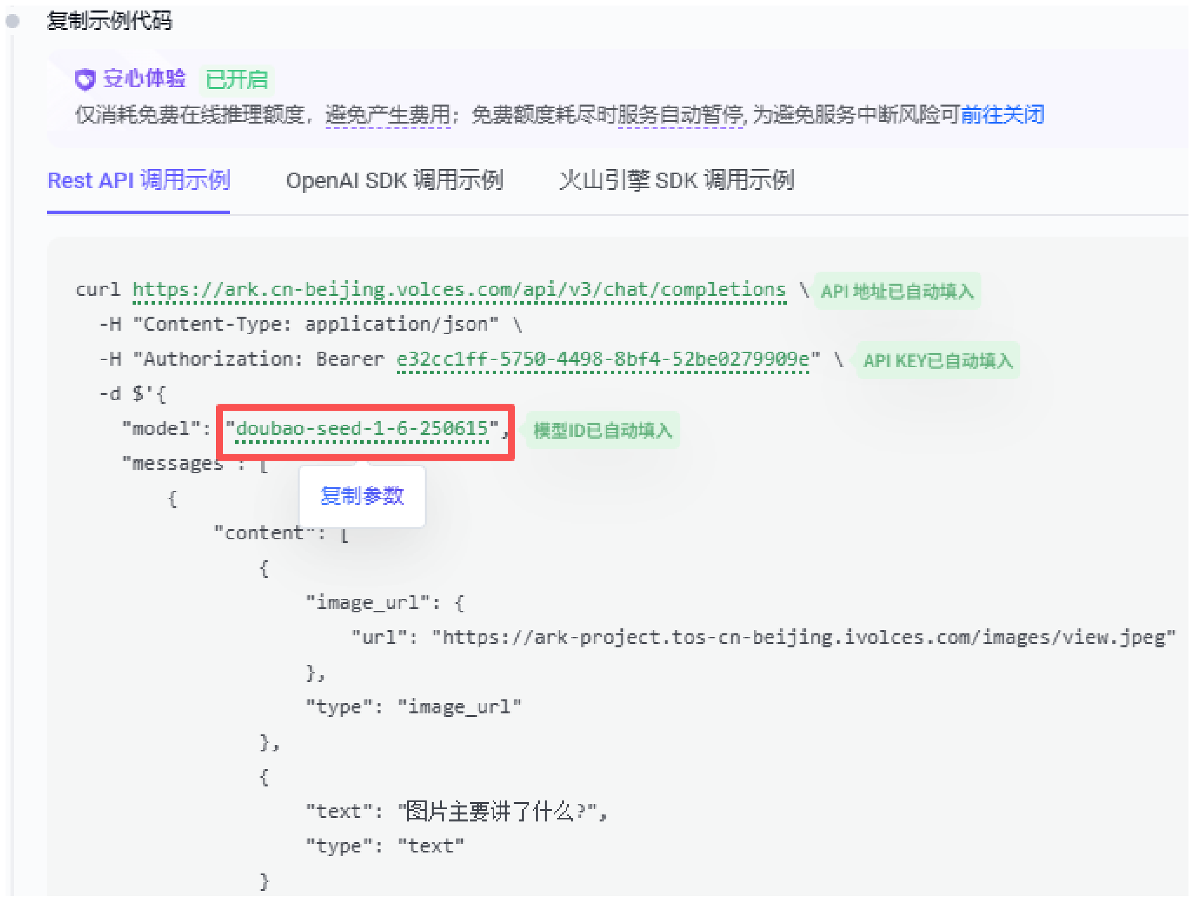
开通协议 ✓ 我已阅读并同意《机器学习平台专用条款》《免责声明》《豆包模型服务协议》

复制示例代码

管理接入详情

STEP 3 创建应用 可选

5. Hover over the model ID to copy it.



```
curl https://ark.cn-beijing.volces.com/api/v3/chat/completions \
  -H "Content-Type: application/json" \
  -H "Authorization: Bearer e32cc1ff-5750-4498-8bf4-52be0279909e" \
  -d $'{
    "model": "doubao-seed-1-6-250615",
    "messages": [
        {
            "content": [
                {
                    "image_url": {
                        "url": "https://ark-project.tos-cn-beijing.ivolces.com/images/view.jpeg"
                    },
                    "type": "image_url"
                },
                {
                    "text": "图片主要讲了什么?",
                    "type": "text"
                }
```

Test with example code

1. Open the test file:

```
cd ~/picar-x/example
sudo nano 18.online_llm_test.py
```

2. Replace the content with the code below, and update `model="xxx"` to the model you want (for example, `doubao-seed-1-6-250615`):

```
from picarx.llm import Doubao
from secret import DOUBAO_API_KEY

INSTRUCTIONS = "You are a helpful assistant."
WELCOME = "Hello, I am a helpful assistant. How can I help you?"

llm = Doubao(
    api_key=DOUBAO_API_KEY,
    model="doubao-seed-1-6-250615",
)
```

3. Run with:

```
sudo python3 18.online_llm_test.py
```

3.6.8 General

This project supports connecting to multiple LLM platforms through a unified interface. We have built-in compatibility with:

- **OpenAI** (ChatGPT / GPT-4o, GPT-4, GPT-3.5)
- **Gemini** (Google AI Studio / Vertex AI)
- **Grok** (xAI)
- **DeepSeek**
- **Qwen** ()
- **Doubao** ()

In addition, you can connect to **any other LLM service that is compatible with the OpenAI API format**. For those platforms, you will need to manually obtain your **API Key** and the correct **base_url**.

Get and Save Your API Key

1. Obtain an **API Key** from the platform you want to use. (See each platform's official console for details.)
2. In your project folder, create a new file:

```
cd ~/picar-x/example
nano secret.py
```

3. Add your key into `secret.py`:

```
# secret.py
API_KEY = "your_api_key_here"
```

Warning: Keep your API Key private. Do not upload `secret.py` to public repositories.

Test With Example Code

1. Open the test file:

```
cd ~/picar-x/example
sudo nano 18.online_llm_test.py
```

2. Replace the content of a Python file with the following example, and fill in the correct `base_url` and `model` for your platform:

Note: About `base_url`: We support the **OpenAI API format**, as well as any API that is **compatible** with it. Each provider has its own `base_url`. Please check their documentation.

```
from picarx.llm import LLM
from secret import API_KEY

INSTRUCTIONS = "You are a helpful assistant."
WELCOME = "Hello, I am a helpful assistant. How can I help you?"

llm = LLM(
```

(continues on next page)

(continued from previous page)

```
    base_url="https://api.example.com/v1",  # fill in your provider's base_url
    api_key=API_KEY,
    model="your-model-name-here",           # choose a model from your provider
)
```

3. Run the program:

```
python3 18.online_llm_test.py
```

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

3.7 19. Local Voice Chatbot

In this lesson, you will combine everything you've learned — **speech recognition (STT)**, **text-to-speech (TTS)**, and a **local LLM (Ollama)** — to build a fully offline **voice chatbot** that runs on your PiCar-X system.

The workflow is simple:

1. **Listen** — The microphone captures your speech and transcribes it with **Vosk**.
2. **Think** — The text is sent to a local **LLM** running on Ollama (e.g., `llama3.2:3b`).
3. **Speak** — The chatbot answers aloud using **Piper TTS**.

This creates a **hands-free conversational robot** that can understand and respond in real time.

3.7.1 Before You Start

Make sure you have prepared the following:

- *5. Install All the Modules(Important)* — Install `robot-hat`, `vilib`, `picar-x` modules, then run the script `i2samp.sh`.
- Tested **Piper TTS** (*1. Testing Piper*) and chosen a working voice model.
- Tested **Vosk STT** (*2. Test Vosk*) and chosen the right language pack (e.g., `en-us`).
- Installed **Ollama** (*1. Install Ollama (LLM) and Download Model*) on your Pi or another computer, and downloaded a model such as `llama3.2:3b` (or a smaller one like `moondream:1.8b` if memory is limited).

3.7.2 Run the Code

1. Open the example script:

```
cd ~/picar-x/example
sudo nano 19.local_voice_chatbot.py
```

2. Update the parameters as needed:

   - `stt = Vosk(language="en-us")`: Change this to match your accent/language package (e.g., `en-us`, `zh-cn`, `es`).
   - `tts.set_model("en_US-amy-low")`: Replace with the Piper voice model you verified in *1. Testing Piper*.
   - `llm = Ollama(ip="localhost", model="llama3.2:3b")`: Update both `ip` and `model` to your own setup.
     - `ip`: If Ollama runs on the **same Pi**, use `localhost`. If Ollama runs on another computer in your LAN, enable **Expose to network** in Ollama and set `ip` to that computer's LAN IP.
     - `model`: Must exactly match the model name you downloaded/activated in Ollama.

3. Run the script:

```
cd ~/picar-x/example
sudo python3 19.local_voice_chatbot.py
```

4. After running, you should see:

   - The bot greets you with a spoken welcome message.
   - It waits for speech input.
   - Vosk transcribes your speech into text.
   - The text is sent to Ollama, which streams back a reply.
   - The reply is cleaned (removing hidden reasoning) and spoken aloud by Piper.
   - Stop the program anytime with `Ctrl+C`.

3.7.3 Code

```python
import re
import time
from picarx.llm import Ollama
from picarx.stt import Vosk
from picarx.tts import Piper

# Initialize speech recognition
stt = Vosk(language="en-us")

# Initialize TTS
tts = Piper()
tts.set_model("en_US-amy-low")
```

(continues on next page)

(continued from previous page)

```
# Instructions for the LLM
INSTRUCTIONS = (
    "You are a helpful assistant. Answer directly in plain English. "
    "Do NOT include any hidden thinking, analysis, or tags like <think>."
)
WELCOME = "Hello! I'm your voice chatbot. Speak when you're ready."

# Initialize Ollama connection
llm = Ollama(ip="localhost", model="llama3.2:3b")
llm.set_max_messages(20)
llm.set_instructions(INSTRUCTIONS)

# Utility: clean hidden reasoning
def strip_thinking(text: str) -> str:
    if not text:
        return ""
    text = re.sub(r"<\s*think[^>]*>.*?<\s*/\s*think\s*>", "", text, flags=re.DOTALL|re.
↪IGNORECASE)
    text = re.sub(r"<\s*thinking[^>]*>.*?<\s*/\s*thinking\s*>", "", text, flags=re.
↪DOTALL|re.IGNORECASE)
    text = re.sub(r"```(?:\s*thinking)?\s*.*?```", "", text, flags=re.DOTALL|re.
↪IGNORECASE)
    text = re.sub(r"\[/?thinking\]", "", text, flags=re.IGNORECASE)
    return re.sub(r"\s+\n", "\n", text).strip()

def main():
    print(WELCOME)
    tts.say(WELCOME)

    try:
        while True:
            print("\n Listening... (Press Ctrl+C to stop)")

            # Collect final transcript from Vosk
            text = ""
            for result in stt.listen(stream=True):
                if result["done"]:
                    text = result["final"].strip()
                    print(f"[YOU] {text}")
                else:
                    print(f"[YOU] {result['partial']}", end="\r", flush=True)

            if not text:
                print("[INFO] Nothing recognized. Try again.")
                time.sleep(0.1)
                continue

            # Query Ollama with streaming
            reply_accum = ""
            response = llm.prompt(text, stream=True)
            for next_word in response:
                if next_word:
```

(continues on next page)

(continued from previous page)

```
                    print(next_word, end="", flush=True)
                    reply_accum += next_word
            print("")

            # Clean and speak
            clean = strip_thinking(reply_accum)
            if clean:
                tts.say(clean)
            else:
                tts.say("Sorry, I didn't catch that.")

            time.sleep(0.05)

    except KeyboardInterrupt:
        print("\n[INFO] Stopping...")
    finally:
        tts.say("Goodbye!")
        print("Bye.")

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

3.7.4 Code Analysis

Imports and global setup

```
import re
import time
from picarx.llm import Ollama
from picarx.stt import Vosk
from picarx.tts import Piper
```

Brings in the three subsystems you built earlier: **Vosk** for speech-to-text (STT), **Ollama** for the LLM, and **Piper** for text-to-speech (TTS).

Initialize STT (Vosk)

```
stt = Vosk(language="en-us")
```

Loads the Vosk model for US English. Change the language code (e.g., `zh-cn`, `es`) to match your voice pack for better accuracy.

Initialize TTS (Piper)

```
tts = Piper()
tts.set_model("en_US-amy-low")
```

Creates a Piper engine and selects a specific voice. Pick a model you've tested in *1. Testing Piper*. Lower-quality voices are faster and use less CPU.

LLM instructions and welcome line

```
INSTRUCTIONS = (
    "You are a helpful assistant. Answer directly in plain English. "
    "Do NOT include any hidden thinking, analysis, or tags like <think>."
)
WELCOME = "Hello! I'm your voice chatbot. Speak when you're ready."
```

Two key UX choices:

- Keep **answers short and direct** (helps with TTS clarity).
- Explicitly forbid hidden "chain-of-thought" tags to reduce noisy outputs.

Connect to Ollama and set conversation scope

```
llm = Ollama(ip="localhost", model="llama3.2:3b")
llm.set_max_messages(20)
llm.set_instructions(INSTRUCTIONS)
```

- `ip="localhost"` assumes the Ollama server runs on the same Pi. If it runs on another LAN machine, put that computer's **LAN IP** and enable *Expose to network* in Ollama.
- `set_max_messages(20)` keeps a short conversational history. Lower this if memory/latency is tight.

Strip hidden reasoning / tags before speaking

```
def strip_thinking(text: str) -> str:
    if not text:
        return ""
    text = re.sub(r"<\s*think[^>]*>.*?<\s*/\s*think\s*>", "", text, flags=re.DOTALL|re.
↪IGNORECASE)
    text = re.sub(r"<\s*thinking[^>]*>.*?<\s*/\s*thinking\s*>", "", text, flags=re.
↪DOTALL|re.IGNORECASE)
    text = re.sub(r"```(?:\s*thinking)?\s*.*?```", "", text, flags=re.DOTALL|re.
↪IGNORECASE)
    text = re.sub(r"\[/?thinking\]", "", text, flags=re.IGNORECASE)
    return re.sub(r"\s+\n", "\n", text).strip()
```

Some models may emit internal-style tags (e.g., `<think>...`). This function removes those so your TTS **only** speaks the final answer.

Tip: If you see other artifacts on screen (because you stream raw tokens), this function already ensures **spoken** output stays clean.

Main loop: greet once, then listen → think → speak

```
print(WELCOME)
tts.say(WELCOME)
```

Greets the user via terminal and speaker. Happens once at startup.

Listen (streaming STT with live partials)

```
print("\n Listening... (Press Ctrl+C to stop)")

text = ""
for result in stt.listen(stream=True):
    if result["done"]:
```

(continues on next page)

(continued from previous page)

```
        text = result["final"].strip()
        print(f"[YOU] {text}")
    else:
        print(f"[YOU] {result['partial']}", end="\r", flush=True)
```

- `stream=True` yields **partial** transcripts for immediate feedback and a **final** transcript when the utterance ends.
- The final recognized text is stored in `text` and printed once.

Guard: If nothing was recognized, you skip the LLM call:

```
if not text:
    print("[INFO] Nothing recognized. Try again.")
    time.sleep(0.1)
    continue
```

This avoids sending empty prompts to the model (saves time and tokens).

Think (LLM) with streamed printing

```
reply_accum = ""
response = llm.prompt(text, stream=True)
for next_word in response:
    if next_word:
        print(next_word, end="", flush=True)
        reply_accum += next_word
print("")
```

- Sends the final transcript to the local LLM and **prints tokens as they arrive** for low latency.
- Meanwhile, you accumulate the full reply in `reply_accum` for post-processing.

Note: If you'd rather **not** show raw tokens, set `stream=False` and just print the final string.

Speak (clean first, then TTS once)

```
clean = strip_thinking(reply_accum)
if clean:
    tts.say(clean)
else:
    tts.say("Sorry, I didn't catch that.")
```

- Cleans the final text to remove hidden tags, then **speaks exactly once**.
- Keeping TTS to a single pass avoids repeated prompts like "[LLM] / [SAY]".

Exit and teardown

```
except KeyboardInterrupt:
    print("\n[INFO] Stopping...")
finally:
    tts.say("Goodbye!")
    print("Bye.")
```

Use **Ctrl+C** to stop. The bot says a short goodbye to signal a clean exit.

3.7.5 Troubleshooting & FAQ

- **Model is too large (memory error)**

  Use a smaller model like `moondream:1.8b` or run Ollama on a more powerful computer.
- **No response from Ollama**

  Make sure Ollama is running (`ollama serve` or desktop app open). If remote, enable **Expose to network** and check IP address.
- **Vosk not recognizing speech**

  Verify your microphone works. Try another language pack (`zh-cn`, `es` etc.) if needed.
- **Piper silent or errors**

  Confirm the chosen voice model is downloaded and tested in *1. Testing Piper*.
- **Answers too long or off-topic**

  Edit `INSTRUCTIONS` to add: **"Keep answers short and to the point."**

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

3.8 20. Treasure Hunt

In this lesson, you will turn your PiCar-X into a **treasure hunter robot**. Arrange a maze in your room and place six different color cards in different corners. Your PiCar-X will **search, recognize, and celebrate** when it finds the target color.

This project combines three skills you've learned so far:

- **Computer Vision** – detecting colored cards with the Pi camera.
- **Keyboard Control** – driving the robot manually through the maze.
- **Speech Feedback** – Pico2Wave announces the target color and success.

It's a fun game that shows how robots can **see, think, and act** just like treasure hunters!

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

3.8.1 Before You Start

Make sure you‘ve completed:

- *5. Install All the Modules(Important)* — Install `robot-hat`, `vilib`, `picar-x` modules, then run the script `i2samp.sh`.

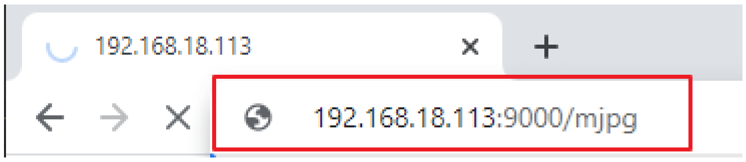
3.8.2 Run the Code

```
cd ~/picar-x/example
sudo python3 20.treasure_hunt.py
```

After running, you’ll see a message like this:

```
* Running on http://0.0.0.0:9000/ (Press CTRL+C to quit)
```

Then, open `http://<your IP>:9000/mjpg` in your browser to view the live video feed. Example: `http://192.168.18.113:9000/mjpg`



3.8.3 Game Rules

1. The robot randomly selects a **target color** and says: **“Look for red!”**
2. You drive PiCar-X with the keyboard:
   - `w` = forward
   - `a` = turn left
   - `s` = backward
   - `d` = turn right
   - `space` = repeat target
   - `Ctrl+C` = quit
3. When the camera sees the target color card, PiCar-X says **“Well done!”**
4. A new target color is chosen, and the hunt continues!

3.8.4 Code

```
#!/usr/bin/env python3

from picarx import Picarx
from vilib import Vilib
from picarx.tts import Pico2Wave

from time import sleep
```

(continues on next page)

(continued from previous page)

```python
import threading
import readchar
import random

# -----------------------
# Settings
# -----------------------
COLORS = ["red", "orange", "yellow", "green", "blue", "purple"]
DETECTION_WIDTH_THRESHOLD = 100  # how wide the color blob must be
DRIVE_SPEED = 80
TURN_ANGLE = 30

MANUAL = """
Press keys to control PiCar-X:
  w: forward    a: turn left    s: backward    d: turn right
  space: repeat target          Ctrl+C: quit
"""

# -----------------------
# Init
# -----------------------
px = Picarx()

tts = Pico2Wave()
tts.set_lang("en-US")

current_color = "red"
key = None
lock = threading.Lock()

def say(line: str):
    print(f"[SAY] {line}")
    tts.say(line)

def renew_color_detect():
    """Choose a new target color and start detection."""
    global current_color
    current_color = random.choice(COLORS)
    Vilib.color_detect(current_color)
    say(f"Look for {current_color}!")

def key_scan_thread():
    """Background thread reading keys."""
    global key
    while True:
        k = readchar.readkey()
        # Map special keys before lowercasing
        if k == readchar.key.SPACE:
            mapped = "space"
        elif k == readchar.key.CTRL_C:
            mapped = "quit"
        else:
```

(continues on next page)

(continued from previous page)

```
            mapped = k.lower()

        with lock:
            key = mapped

        if mapped == "quit":
            return
        sleep(0.01)

def car_move(k: str):
    if k == "w":
        px.set_dir_servo_angle(0)
        px.forward(DRIVE_SPEED)
    elif k == "s":
        px.set_dir_servo_angle(0)
        px.backward(DRIVE_SPEED)
    elif k == "a":
        px.set_dir_servo_angle(-TURN_ANGLE)
        px.forward(DRIVE_SPEED)
    elif k == "d":
        px.set_dir_servo_angle(TURN_ANGLE)
        px.forward(DRIVE_SPEED)

def main():
    global key

    # Start camera and web preview
    Vilib.camera_start(vflip=False, hflip=False)
    Vilib.display(local=False, web=True)
    sleep(0.8)

    print(MANUAL.strip())
    say("Game start!")
    sleep(0.1)
    renew_color_detect()

    # Start keyboard thread (modern style)
    key_thread = threading.Thread(target=key_scan_thread, daemon=True)
    key_thread.start()

    try:
        while True:
            # Check detection: if target color present and wide enough
            if (Vilib.detect_obj_parameter.get("color_n", 0) != 0 and
                Vilib.detect_obj_parameter.get("color_w", 0) > DETECTION_WIDTH_
↪THRESHOLD):
                say("Well done!")
                sleep(0.1)
                renew_color_detect()

            # Take a snapshot of the last key (and clear it)
            with lock:
```

(continues on next page)

(continued from previous page)

```
                k = key
                key = None

            # Handle movement / actions
            if k in ("w", "a", "s", "d"):
                car_move(k)
                sleep(0.5)
                px.stop()
            elif k == "space":
                say(f"Look for {current_color}!")
            elif k == "quit":
                print("\n[INFO] Quit requested.")
                break

            sleep(0.05)

    except KeyboardInterrupt:
        print("\n[INFO] Stopped by user.")
    finally:
        try:
            Vilib.camera_close()
        except Exception:
            pass
        px.stop()
        say("Goodbye!")
        sleep(0.2)

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

3.8.5 How It Works

1. **Initialization**
   - Import modules and configure PiCar-X, camera, and TTS.
   - Set color list, speed, and steering angle.
2. **Target Selection**
   - `renew_color_detect()` randomly picks a target color.
   - The robot announces the target with Pico2Wave.
3. **Keyboard Control**
   - `key_scan_thread()` runs in the background to capture keys.
   - Keys `w, a, s, d` control motion; `space` repeats target.
4. **Color Detection**
   - Camera constantly checks if the target color is visible.
   - If the detected blob is large enough, PiCar-X celebrates.
5. **Main Loop**

- Continuously handles movement, detection, and feedback.
- Cleanly stops the robot and camera when quitting.

3.8.6 Troubleshooting

- **Camera feed not working?**

  Run `libcamera-hello` to check if the Pi camera is connected properly.

- **Robot doesn't detect colors?**

  Ensure the cards are printed clearly and placed in good lighting. Try adjusting `DETECTION_WIDTH_THRESHOLD`.

- **No voice feedback?**

  Check that `pico2wave` is installed and your audio output is configured.

- **Car doesn't move?**

  Verify PiCar-X power is on and the motor calibration is correct.

By completing this lesson, you've built a **mini treasure hunt game** with PiCar-X, combining **vision, control, and interaction** into one project!

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

3.9 21. AI Voice Assistant Car

This lesson turns your PiCar-X into an **AI-powered voice assistant on wheels**. The robot can wake up to your voice, recognize what you say, talk back with emotion, and act out its "feelings" through movements, gestures, and lights.

You'll build a **fully interactive voice assistant car** using:

- **LLM** - Large Language Model (OpenAI GPT or Doubao).
- **STT** - Speech-to-Text (voice to text).
- **TTS** - Text-to-Speech (text to voice).
- **Sensors + Actions** - Ultrasonic, camera, and built-in expressive actions.

3.9.1 Before You Start

Make sure you‘ve completed:

- *5. Install All the Modules(Important)* — Install `robot-hat`, `vilib`, `picar-x` modules, then run the script `i2samp.sh`.
- *1. Testing Piper* — Check the supported languages of **Piper TTS**.
- *2. Test Vosk* — Check the supported languages of **Vosk STT**.
- *18. Connecting to Online LLMs* — This step is **very important**: obtain your **OpenAI** or **Doubao** API key, or the API key for any other supported LLM.

You should already have:

- A working **microphone** and **speaker** on your PiCar-X.
- A **valid API key** stored in `secret.py`.
- A stable network connection (a **wired connection** is recommended for better stability).

3.9.2 Run the Example

Both language versions are placed in the same directory:

```
cd ~/picar-x/example
```

English version (OpenAI GPT, instructions in English):

```
sudo python3 21.voice_active_car_gpt.py
```

- LLM: `OpenAI GPT-4o-mini`
- TTS: `en_US-ryan-low` (Piper)
- STT: Vosk (`en-us`)

Wake word:

```
"Hey buddy"
```

—

Chinese version (Doubao, instructions in Chinese):

```
sudo python3 21.voice_active_car_doubao_cn.py
```

- LLM: `Doubao-seed-1-6-250615`
- TTS: `zh_CN-huayan-x_low` (Piper)
- STT: Vosk (`cn`)

Wake word:

```
"  "
```

Note: You can modify the **wake word** and **robot name** in the code: `NAME = "Buddy"` or `NAME = ""` `WAKE_WORD = ["hey buddy"]` or `WAKE_WORD = [" "]`

3.9.3 What Will Happen

When you run this example successfully:

- The robot **waits for the wake word** (e.g., “Hey Buddy” / “ ”).
- When it hears the wake word:
  - LEDs will **blink** and stay on.
  - The robot **greets you** with a cheerful voice.
- It then starts **listening to your voice** in real time.
- After recognizing what you said, it:
  - Sends your speech to the **LLM** (OpenAI or Doubao).
  - **Thinks** and blinks LED while processing.
  - Replies with **TTS voice**.
  - Executes **corresponding actions** (e.g., nodding, turning, celebrating).
- If you approach it too closely, the ultrasonic sensor:
  - Triggers an auto **backward** move for safety.
  - Interrupts the current round with a warning response.

Example interaction

```
You: Hey Buddy
Robot: Hi there!

You: Turn left and look around.
Robot: Roger that, turning my head left like a curious cat!
ACTIONS: turn_left, look_left
```

3.9.4 Switching to Other LLMs or TTS

You can easily switch to other LLMs, TTS, or STT languages with just a few edits:

- Supported LLMs:
  - OpenAI
  - Doubao
  - Deepseek
  - Gemini
  - Qwen

  - Grok
- *1. Testing Piper* — Check the supported languages of **Piper TTS**.
- *2. Test Vosk* — Check the supported languages of **Vosk STT**.

To switch, simply modify the initialization part in the code:

```
from picarx.llm import Gemini as LLM
llm = LLM(api_key="YOUR_KEY", model="gemini-pro")

# Set models and languages
TTS_MODEL = "en_US-ryan-low"
STT_LANGUAGE = "en-us"
```

3.9.5 Action & Sound Reference

Below are the **action keywords** the LLM can return (after the `ACTIONS:` line) and what they do on the robot.

Action	What it does (per preset_actions.py)	Effect / Notes
`shake head`	Quickly swings camera pan angle rightleft in diminishing steps, then centers.	"No" gesture; wheels remain stopped.
`nod`	Bobs camera tilt updown twice, then centers.	"Yes" gesture; wheels remain stopped.
`wave hands`	Tilts camera, then steers left/right twice (±25°) and centers.	Playful wave (uses steering servo as "arms").
`resist`	Small tilt; alternates (steer ±15°, pan ±15°) 3 times; stops and centers.	"Refuse"/defensive motion.
`act cute`	Head tilt down; quick forward/back micro-shuffles (short motor pulses), then reset.	Bouncy "cute" move; very short motions.
`rub hands`	Repeated small steering oscillation (±6°) five times; reset.	Mimics "rubbing hands together".
`think`	Smooth pan right + tilt down + steer right sweep; brief hold; small poised pose; reset.	Used as a single "thinking" animation.
`twist body`	Three cycles of short forward/stop/pan-left/steer-left, then short backward/stop/pan-right/steer-right.	Gives a body "twist" vibe.
`celebrate`	Tilt up; two right pan/steer flourishes, then two left flourishes; returns to center.	Festive, symmetrical flourish.
`depressed`	Series of downward tilt pulses with varying angles and pauses; ends after a long beat and resets.	"Sad" posture sequence.

Movement & Utility

Action	What it does	Notes
`forward`	Drive forward at low speed for ~1 second, then stop.	Implemented by `forward(car)` (5% speed + 1s).
`backward`	Drive backward at low speed for ~1 second, then stop.	Implemented by `backward(car)` (5% speed + 1s).

Sound Effects

Sound	What it does	Notes
`honking`	Plays `car-double-horn.wav` asynchronously (volume ~100).	Triggered via `Music.sound_play_threading`.
`start engine`	Plays `car-start-engine.wav` asynchronously (volume ~50).	Boot/ready cue.

Sensor Triggers (Automatic)

- **Ultrasonic proximity**
  - Trigger: distance < 10 cm
  - Side effect: auto `backward` + disable image for this round
  - Injected message: `<<<Ultrasonic sense too close: {distance}cm>>>`

Lifecycle Hooks (LED Indicators)

- `before_listen` → blink twice (ready to listen)
- `before_think` → blinking (thinking)
- `before_say` → LED on (speaking)
- `after_say` → wait for actions → LED off
- `on_stop` → stop actions, close devices

3.9.6 Troubleshooting

- **The robot doesn't respond to wake word**
  - Check if the microphone works.
  - Ensure `WAKE_ENABLE = True`.
  - Adjust wake word to match your pronunciation.
- **No sound from the speaker**
  - Verify TTS model setup.
  - Test Piper or Espeak manually.
  - Check speaker connection and volume.
- **API Key error or timeout**
  - Check your key in `secret.py`.
  - Ensure network connection.
  - Confirm the LLM is supported.
- **Picar-X doesn't move or act**

  - Check that the action name matches `actions_dict`.
  - Verify motor and servo connections.
- **Ultrasonic sensor keeps triggering unexpectedly.**
  - Check sensor installation height and angle.
  - Adjust the `TOO_CLOSE` distance threshold in code.

CHAPTER

FOUR

PYTHON VIDEO COURSE

This video course is an online Python tutorial specifically designed for interactive learning. It includes three introductory videos that cover the essential foundations: starting with setting up the Raspberry Pi, assembling the PiCar-X, and then installing the necessary robot modules. This initial phase ensures that everything is prepared before beginning the various projects with the PiCar-X.

Following the introductory section, the course features 12 project videos. These projects progressively develop skills starting from basic movement of the PiCar-X, keyboard control, Text-to-Speech (TTS), obstacle avoidance, line tracking, to applications in computer vision. The course then advances to more complex projects that combine multiple functionalities. Besides teaching you to run examples from the online tutorial, the videos also provide additional extensions to each topic, allowing for a deeper understanding of each feature.

Get Started

The course begins with three introductory videos that lay the foundation for working with the PiCar-X. These videos cover:

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

4.1 Video A1: Starting with Raspbrry Pi

This is the first video for the PiCar-X.

This video provides a comprehensive tutorial on setting up a Raspberry Pi for use with the PiCar-X robot car. It covers:

- The features of the PiCar-X.
- How to image the Raspberry Pi OS.
- Various programming methods including using HDMI, PowerShell, Remote Desktop, and SunFounder Create Agent.

- Installing necessary robot modules.

If you are a beginner, it is suggested to follow the steps one by one. If you are more familiar with Raspberry Pi, then you can directly follow the latter part of the video to install the necessary modules.

Video

Related On-line Tutorials

- *1. Quick Guide on Python*

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

4.2 Video A2: Assembly of the PICAR-X

In this video, the focus is on assembling the PICAR-X.

Key aspects of the tutorial include:

- **Unboxing**: Introduction of the kit components, including the assembly instruction sheet, tools, modules, and motors.
- **Assembly Guide**: Detailed steps for assembling the PICAR-X, starting from attaching screws to the Raspberry Pi, connecting cables, and setting up the robot hat.
- **Servo Calibration**: Using SunFounder's Create Agent for zeroing the servos.
- **Motor and Battery Installation**: Attaching motors to the frame and securing the battery.
- **Sensor and Camera Installation**: Installing and connecting the grayscale sensor, ultrasonic sensor, and camera.
- **Wiring and Final Setup**: Organizing and connecting wires for various components, followed by final checks and tidy wire management.

The video is designed to guide you through each step of the assembly process, ensuring a thorough understanding of how each component fits together.

Video

Related On-line Tutorials

- *Assemble the PiCar-X*

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

4.3 Video A3: Calibrate the PiCar-X

This third video in the series focuses on calibrating the PiCar-X and introducing the Robot HAT. The tutorial is divided into several key sections:

- **Greyscale Sensor Calibration**: Instructions on accessing and calibrating the greyscale sensor.
- **DC Motors and Servo Calibration**: Steps for calibrating the DC motors for direction and speed, as well as the steering and camera pan/tilt servos.
- **Script Automation at Startup**: How to set scripts to run automatically at the Raspberry Pi's startup.
- **Robot HAT Overview**: Detailed overview of the Robot HAT, including its features and functionalities.

This tutorial is crucial for those looking to fine-tune their PiCar-X and understand the technical aspects of the Robot HAT.

Video

Related On-line Tutorials

- *1. Calibrating the PiCar-X*

Projects

Following the setup, the course dives into twelve project-based videos. These videos progressively enhance the capabilities of the PiCar-X, starting from basic movements to more complex tasks, including:

4.4 Video 1: Motor Move and Steering Control

This video serves as the first project tutorial in the PiCar-X series, focusing on how to control the motors and steering servo of the PiCar-X. The key contents include:

- **Starting and Stopping Motors**: Demonstrates how to control the movement of the PiCar-X, including moving forward and backward.
- **Motor Speed Control**: Shows how to increase and decrease the speed of the motors.
- **Steering Control**: Teaches how to turn left and right by controlling the front steering servo.
- **Script Execution**: Illustrates how to run a program automatically when the Raspberry Pi starts.
- **Move.py Code Analysis**: Offers an in-depth explanation of the Move.py script, detailing the logic behind controlling motors and steering.
- **Testing and Debugging**: Practical testing to ensure both the code and hardware function correctly.

This video is an excellent practical guide for beginners to PiCar-X, providing detailed instructions on controlling motors and steering, laying a solid foundation for further project development.

Video

Related On-line Tutorials

- *2. Let PiCar-X Move*

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

4.5 Video 2: Controlling the PiCar-X using keyboard

In this video tutorial, you'll learn how to control the PiCar-X robot using a keyboard. It covers:

- **Basic Control**: Demonstrates controlling the PiCar-X with keyboard commands - "W" for forward, "S" for backward, "A" for left turns, and "D" for right turns.
- **Tilt and Pan of Camera**: Teaches controlling the mounted camera's tilt and pan using "I" (up), "K" (down), "J" (left), and "L" (right).
- **Code Explanation**: Provides a detailed explanation of the Python code for keyboard control, including library imports and control logic.
- **Running the Code**: Shows how to execute the Python code for keyboard control, including connecting to the PiCar-X.

- **Exiting Control**: Explains exiting keyboard control by pressing "Ctrl + C", returning the robot to its default state.

This tutorial is ideal for beginners and robotics enthusiasts, offering clear instructions and a hands-on demonstration for controlling the PiCar-X robot with a keyboard.

Video

Related On-line Tutorials

- *3. Keyboard Control*

4.6 Video 3: Text to Speech

This tutorial covers the text-to-speech features of the PiCar-X robot:

- **Robot Setup and Initial Movement**: Demonstrates the robot's readiness and initial movement, including stopping and obstacle detection.
- **Text-to-Speech Capabilities**: Shows how the robot can speak predefined phrases and count down using text-to-speech technology.
- **Custom Script Demonstration**: Runs a custom script where the robot talks, moves, reacts to obstacles, and makes U-turns.
- **Playing Music**: Teaches how to play music files on the robot using Python scripts.

The lesson offers an in-depth tutorial on integrating text-to-speech functionality into the PiCar-X robot, including practical demonstrations and code details.

Video

Related On-line Tutorials

- *13. Play Music and Sound Effects*

4.7 Video 4: Obstacle Avoidance with Ultrasonic

This video tutorial covers obstacle avoidance using an ultrasonic sensor in the PiCar-X robot:

- **Introduction to Ultrasonic Sensor**: Explains how to use the ultrasonic sensor for measuring distance and controlling the car's movement.
- **Python Code Walkthrough**: Details the Python code for obstacle avoidance, including variable definitions and conditional movement logic.
- **Creating and Running Custom Scripts**: Shows how to write and execute custom scripts for U-turns and obstacle reactions.
- **Practical Demonstrations**: Provides demonstrations of the robot's obstacle avoidance capabilities on different surfaces.
- **Code Saving and Updating**: Explains how to save and update scripts on the robot.

This lesson offers an essential guide to implementing ultrasonic sensor-based obstacle avoidance in the PiCar-X robot.

Video

Related On-line Tutorials

- *4. Obstacle Avoidance*

4.8 Video 5: Greyscale Line Tracking

This video tutorial explores greyscale line tracking using the PiCar-X robot:

- **Line Detection**: Demonstrates how the robot detects a black line on a white surface using a greyscale sensor, allowing it to track or drive over the line.
- **Python Code Explanation**: Explains the Python code involved in line tracking, detailing the import of modules, creation of objects, and logic for responding to line detection.
- **Practical Demonstrations**: Provides demonstrations of the robot detecting and following lines on different surfaces.
- **Troubleshooting and Tweaking**: Discusses the need for tweaking and fixing the code for better line tracking performance.

This lesson offers a comprehensive guide on implementing greyscale line tracking in the PiCar-X, including practical demonstrations, code walkthroughs, and troubleshooting tips.

Video

Related On-line Tutorials

- *6. Line Tracking*

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

4.9 Video 6: Cliff Detection

This tutorial provides essential insights into programming and utilizing cliff detection in the PiCar-X robot.

- **Introduction**: Explains the use of a grayscale sensor in PiCar-X for cliff detection by measuring surface reflections.
- **Python Code Walkthrough**: Covers the code for cliff detection, detailing sensor setup, and program logic.
- **Demonstration and Testing**: Shows the robot detecting cliffs and reacting, with steps on running and testing the code.
- **Auto-Start Setup**: Guides on configuring the Raspberry Pi to run the cliff detection script on boot.
- **Practical Application**: Demonstrates the robot's response to different surfaces and edges, highlighting its cliff detection capability.

Video

Related On-line Tutorials

- *5. Cliff Detection*

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

4.10 Video 7: PiCar-X Computer Vision

This video tutorial focuses on computer vision capabilities in the PiCar-X:

- **Introduction to Computer Vision**: Teaches how to detect hand movement, fingers, colors, faces, and read QR codes using the PiCar-X equipped with a camera.
- **Remote Desktop and Python Code Execution**: Demonstrates using VNC for remote desktop access to the Raspberry Pi and running Python code for computer vision tasks.
- **Color Detection and Photo Taking**: Shows how to detect different colors and take photos using camera controls.
- **QR Code and Face Detection**: Explains how to switch between QR code reading and face detection features.
- **Object Detection**: Discusses the use of built-in functions from SunFounder's VI library for object detection.
- **Viewing Video on Browser and Mobile**: Teaches how to stream the camera feed to a browser or mobile phone, ensuring they are connected to the same network as the PiCar-X.
- **Hand Detection**: Covers the use of hand detection feature from the VI library and running corresponding Python code.
- **Code Editing and Running**: Demonstrates creating, editing, and running Python scripts for various computer vision tasks.

This lesson offers a comprehensive guide to exploring computer vision features in the PiCar-X, including practical demonstrations of color detection, QR code reading, face detection, and hand movement tracking.

Video

Related On-line Tutorials

- *7. Computer Vision*

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

4.11 Video 8: PiCar-X Stares at You

This tutorial teaches how to use a camera and servo on the PiCar-X robot for object tracking:

- **Camera-Servo Integration**: Demonstrates connecting a camera with a servo for movement tracking.
- **Python Code Overview**: Explains the code for controlling the PiCar-X's camera movements.
- **Starting Camera & Video Display**: Shows starting the camera and displaying the video feed.
- **Face Tracking**: Details how the robot tracks a human face, adjusting its camera angles.
- **Demonstrations**: Includes demonstrations of the robot's camera following and adjusting to movements.
- **Video Streaming**: Covers streaming the camera feed to a browser or mobile phone.

This lesson provides a succinct guide on enabling the PiCar-X to track and focus on objects or faces using its camera.

Video

Related On-line Tutorials

- *8. Stare at You*

4.12 Video 9: Recording Video

This tutorial offers a concise yet detailed guide on recording and managing videos using the PiCar-X robot.

- **Overview**: Teaches how to record HD 1080p videos using the PiCar-X, a Raspberry Pi self-driving robot car kit.
- **Documentation and Code**: Guides through the PiCar-X documentation for video recording and explains the Python script for recording control (start, pause, continue, stop).
- **Remote Desktop Connection**: Demonstrates how to connect remotely to the Raspberry Pi for video recording.
- **Practical Recording Demonstration**: Shows running the video recording script and operating it through a remote desktop.
- **Video Playback**: Illustrates locating and playing back recorded videos to check quality.
- **Additional Features**: Introduces PiCamera2 GitHub Repository for more recording options like time-lapse and easy capture.
- **Simple Video Scripting**: Highlights creating and running basic video recording scripts with different formats and configurations.

Video

Related On-line Tutorials

- *9. Record Video*

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

4.13 Video 10: Bull Fight with PiCar-X

This tutorial covers using the PiCar-X Robot for a "bullfight" game, focusing on color detection and movement:

- **Bull Fight Concept**: Uses PiCar-X's camera to track and follow red color, with pan and tilt control.
- **Python Code Overview**: Details the programming behind color detection and robotic movements.
- **Remote Desktop Access**: Demonstrates managing the PiCar-X's code via remote desktop.
- **Color Tracking Mechanics**: Explains how the robot tracks red objects, adjusting its movement accordingly.
- **Demonstration**: Showcases the PiCar-X in action, pursuing a red target.

This lesson provides a comprehensive guide on programming the PiCar-X for a color-tracking game, complete with code explanations and a live demonstration.

Video

Related On-line Tutorials

- *10. Bull Fight*

4.14 Video 11: PiCar-X as Video Car

This tutorial teaches using PiCar-X as a video car with camera control:

- **Control Mechanics**: Introduces keyboard controls for movement - forward, backward, left, right, and stop.
- **Video Car Features**: Focuses on using PiCar-X for driving, adjusting speed, turning, and taking pictures.
- **Python Code Overview**: Briefly explains the Python code for controlling the car and camera.
- **Remote Desktop and Program Setup**: Shows setting up the car's program via remote desktop and running it at startup.
- **Live Demonstration**: Includes demonstrations of controlling the PiCar-X with keyboard inputs and using its camera.
- **Photo Capture and Viewing**: Teaches capturing and viewing photos taken by the PiCar-X.

The lesson provides a concise overview of operating the PiCar-X as a video car, emphasizing hands-on control and camera usage.

Video

Related On-line Tutorials

- *11. Video Car*

- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

4.15 Video 12: Treasure Hunt Game

The tutorial provides an engaging and educational experience in programming and robotics with the PiCar-X.

- **Overview**: Demonstrates a treasure hunt game where the PiCar-X robot detects and navigates towards randomly selected colors.
- **Python Code**: Detailed explanation of the Python code used for color detection, robot movement control, and text-to-speech announcements.
- **Gameplay**: The robot moves using keyboard inputs to find and reach a target color, which is announced via text-to-speech.
- **Practical Demo**: Shows the robot in action, successfully identifying and moving towards different colors like red, yellow, and blue.
- **Game Exit Instructions**: Covers how to safely exit the game, including stopping the robot and shutting down the camera.

Video

Related On-line Tutorials

- *20. Treasure Hunt*

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

4.16 Video 12: Control PiCAR-X Robot Car use mobile app

The tutorial provides a comprehensive guide on using the mobile app for the Picar-X Raspberry Pi robot from SunFounder.

1. **Overview**: Explains the setup and usage of the SunFounder Picar-X robot car kit, including how to use the mobile app to control the robot.
2. **Initial Setup**: Demonstrates the initial setup steps, including installing necessary modules and setting up the PowerShell for SSH access.
3. **App Integration**: Shows how to customize the mobile app interface with various widgets such as button switches, joysticks, and digital displays.
4. **Python Code Execution**: Guides through the process of running Python scripts to control the robot, including setting up the IP address and executing the scripts.
5. **Camera Setup**: Explains how to set up and view the camera feed from the robot, ensuring proper connection and functionality.
6. **App Installation and Configuration**: Details the steps to install and configure the SunFounder controller app, connect to the robot, and control its movements.
7. **Practical Demonstration**: Provides a hands-on demonstration of controlling the robot using the app, including moving the camera and navigating the robot.
8. **Debugging and Troubleshooting**: Offers tips on troubleshooting common issues, ensuring smooth operation and connectivity of the robot and app.

Video

Related On-line Tutorials

- *20. Treasure Hunt*

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

CHAPTER

FIVE

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.

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

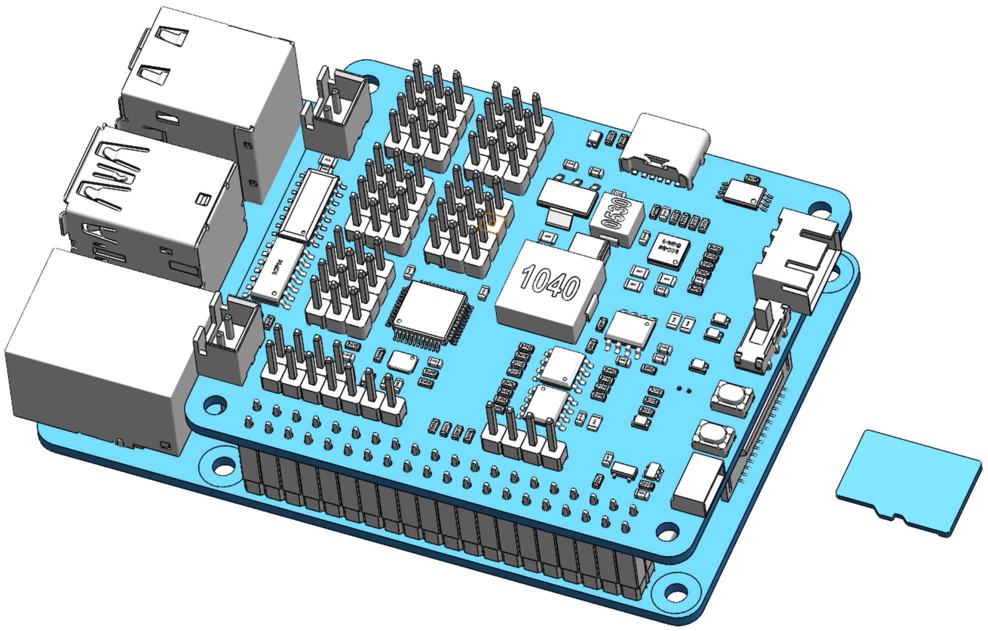
5.1 Quick Guide on EzBlock

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.

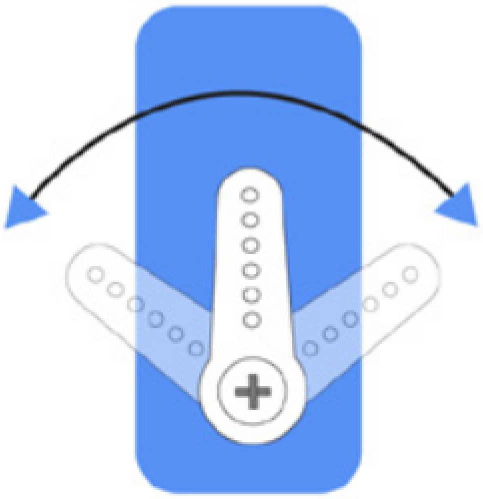
So here we need to set all the servo angles to 0° and then install them, so that the servo angle is in the middle, no matter which direction to turn.

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.

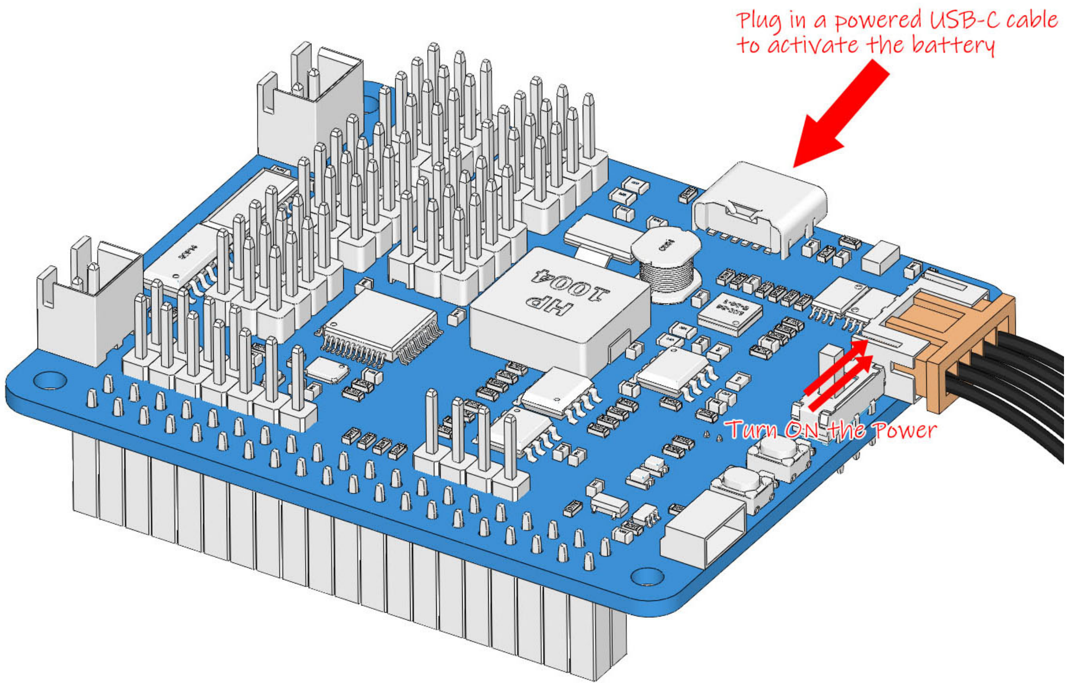
   Note: After the installation is complete, please return to this page.



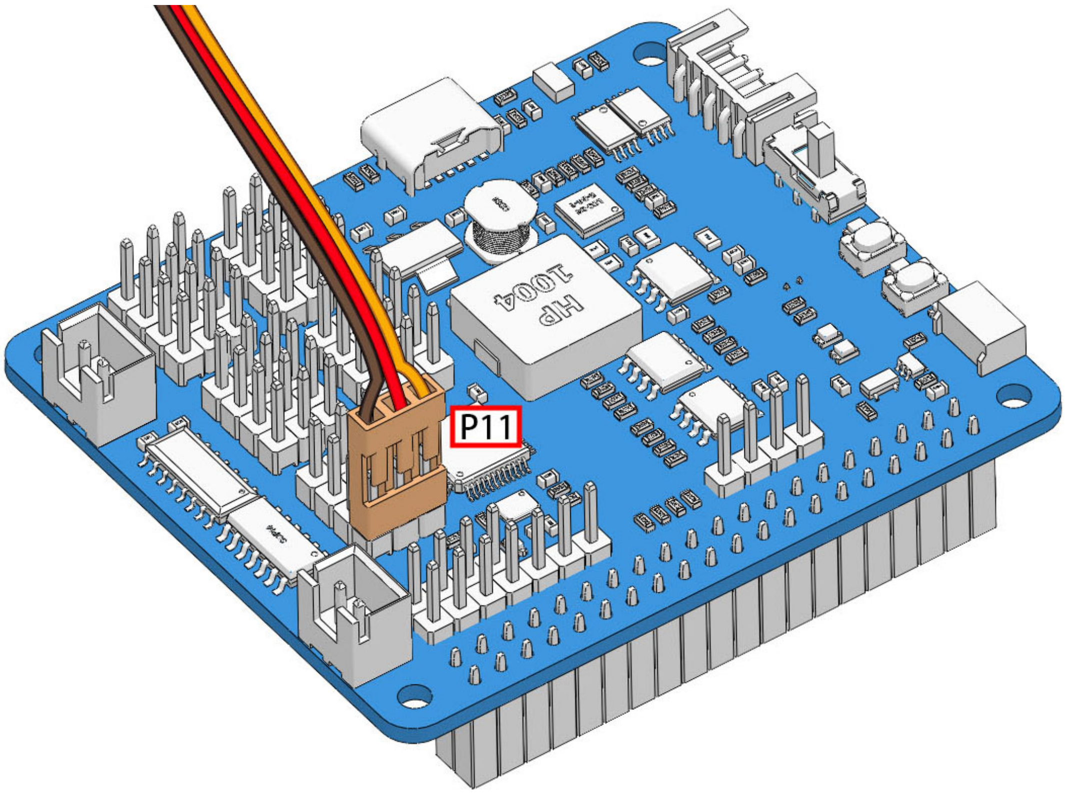
2. To ensure that the servo has been properly set to 0°, first insert the servo arm into the servo shaft and then gently rotate the rocker arm to a different angle. This servo arm is just to allow you to clearly see that the servo is rotating.



3. Follow the instructions on the assembly foldout, insert the battery cable and turn the power switch to the ON. Then plug in a powered USB-C cable to activate the battery. Wait for 1-2 minutes, there will be a sound to indicate that the Raspberry Pi boots successfully.

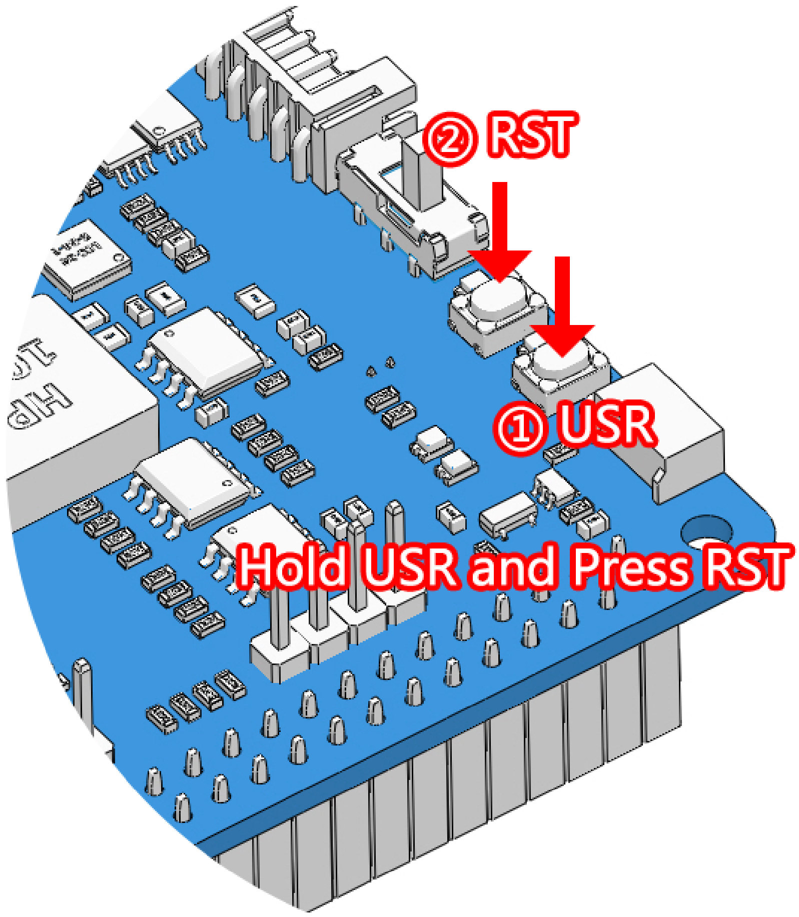


4. Next, plug the servo cable into the P11 port as follows.



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.

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.

- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

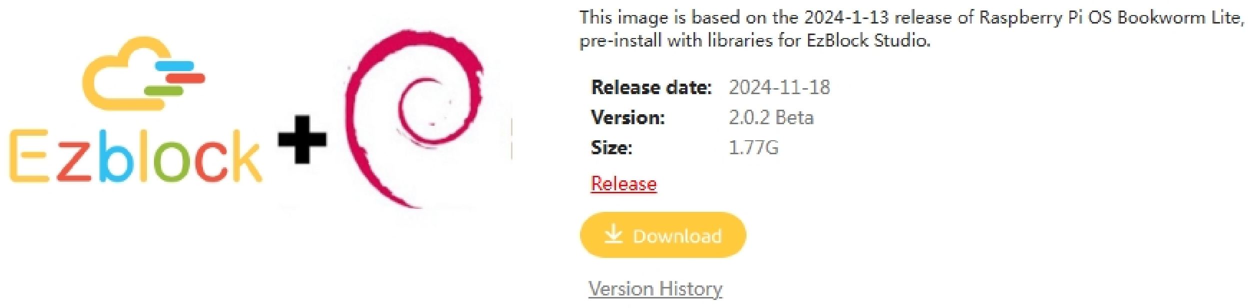
Ready to explore and create with us? Click [] and join today!

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.

Note: If you are using the Raspberry Pi 5, please download the Beat version. Have any questions or issues during using? please don't hesitate to contact us.


This image is based on the 2024-1-13 release of Raspberry Pi OS Bookworm Lite, pre-install with libraries for EzBlock Studio.

Release date:	2024-11-18
Version:	2.0.2 Beta
Size:	1.77G

- 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.

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.

- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

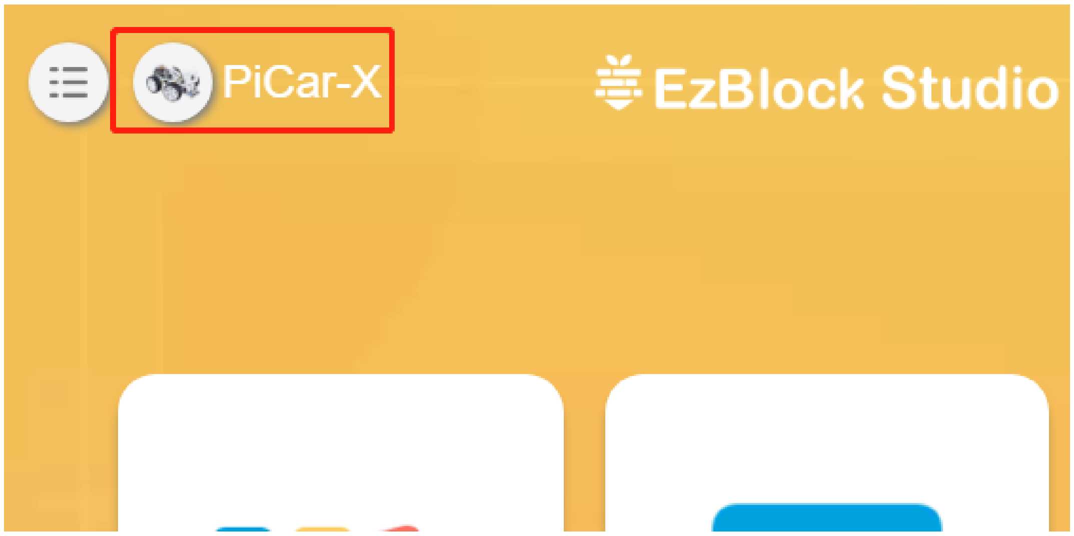
5.3 Calibrate the Car

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.

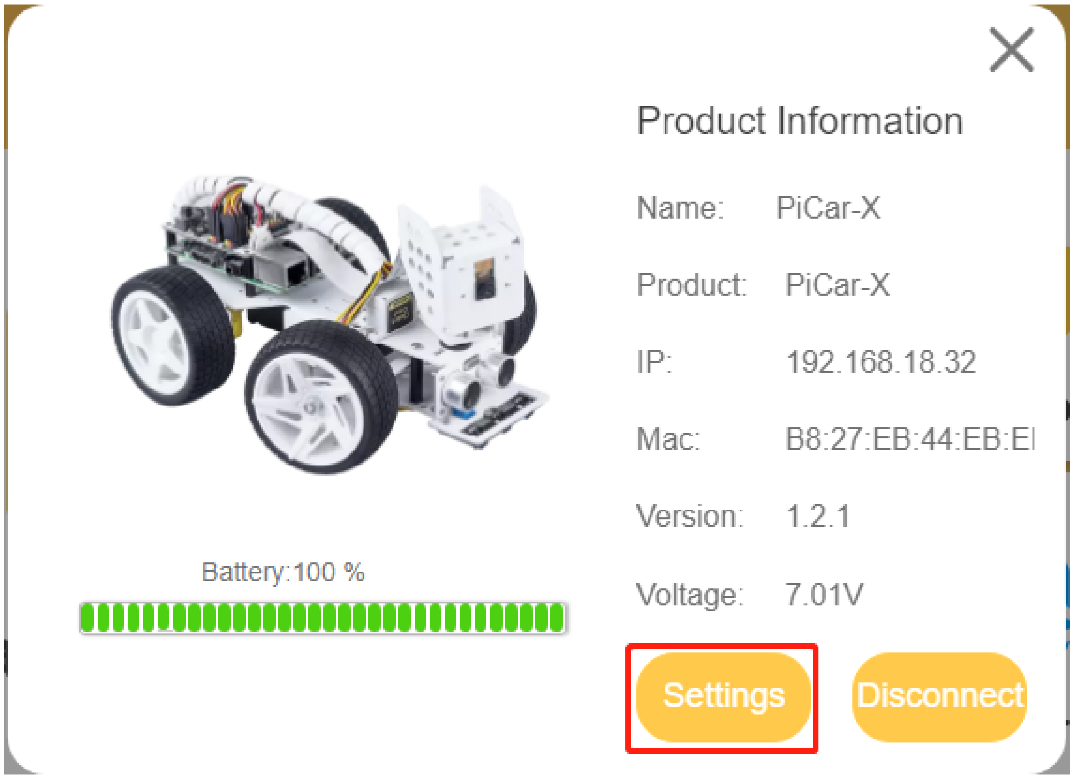
But if you think the assembly is perfect and no calibration is needed, you can also skip this step.

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

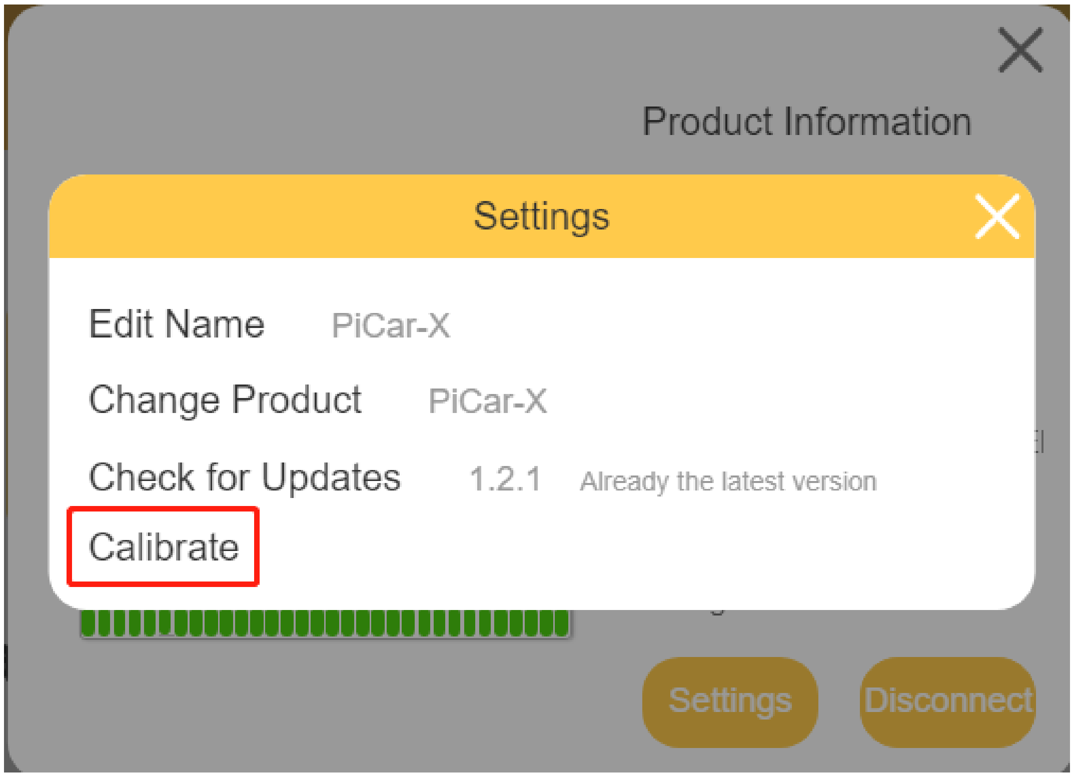
1. You can open the product detail page by clicking the connect icon in the upper left corner.



2. Click the **Settings** button.



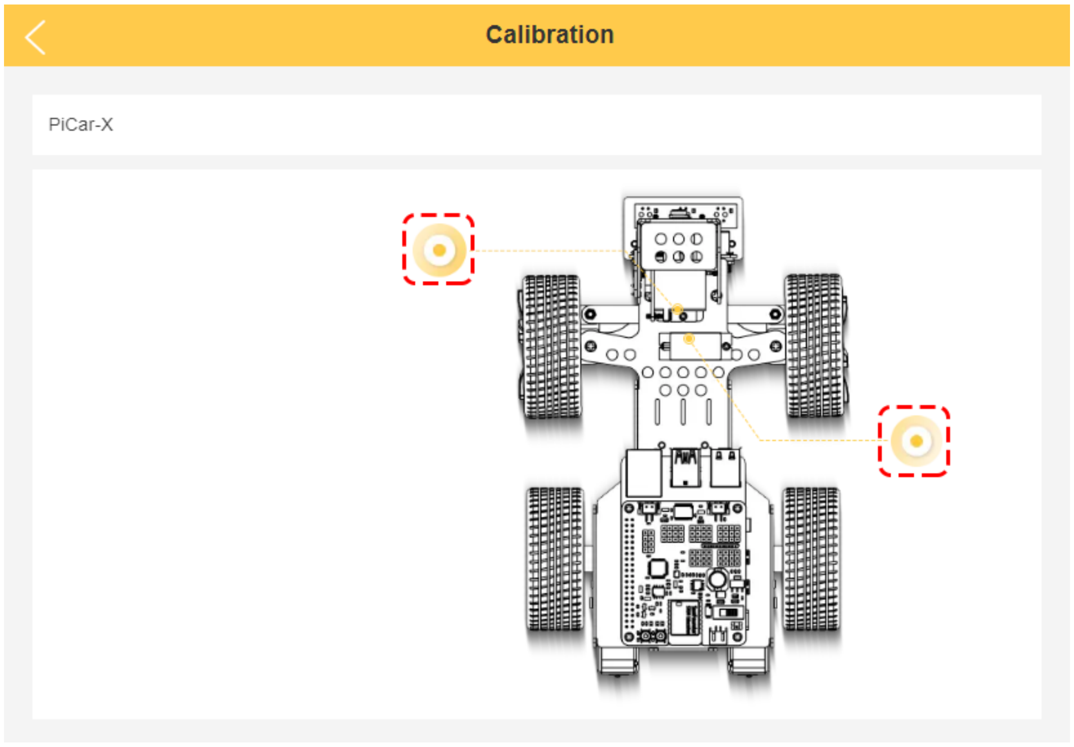
3. On this page, you can change the product name, product type, view the app version or calibrate the robot. Once you click on **Calibrate** you can go to the calibration page.



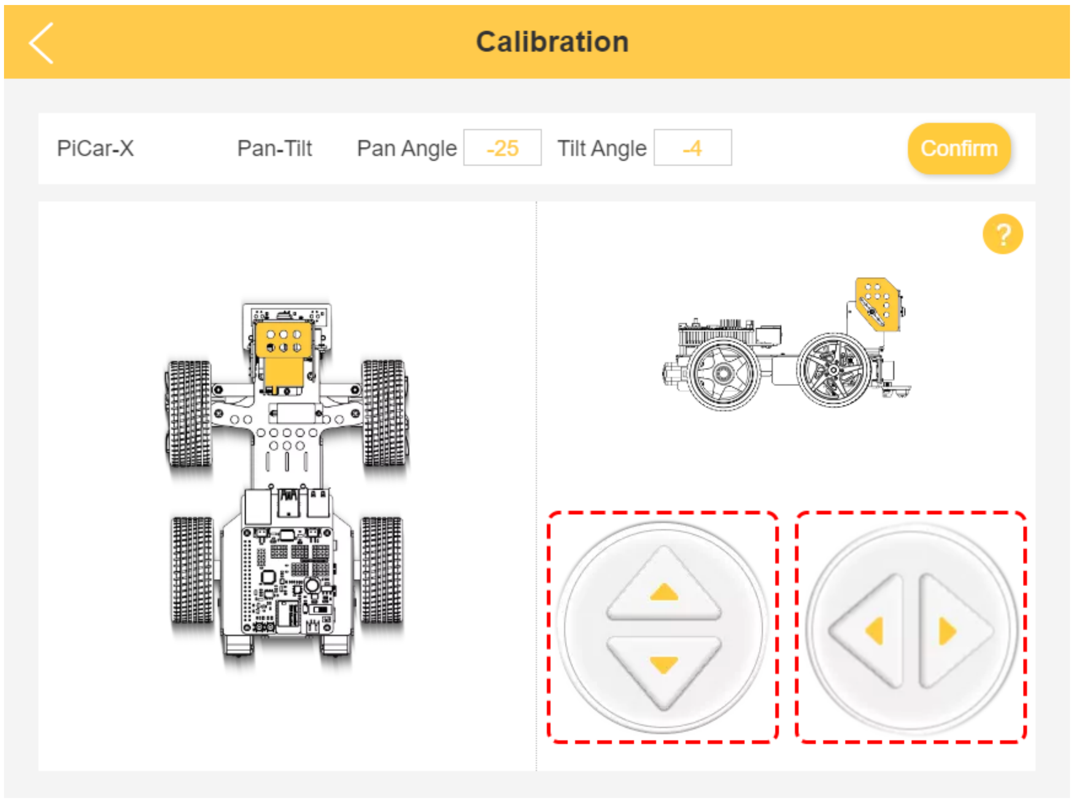
The calibration steps are as follows:

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

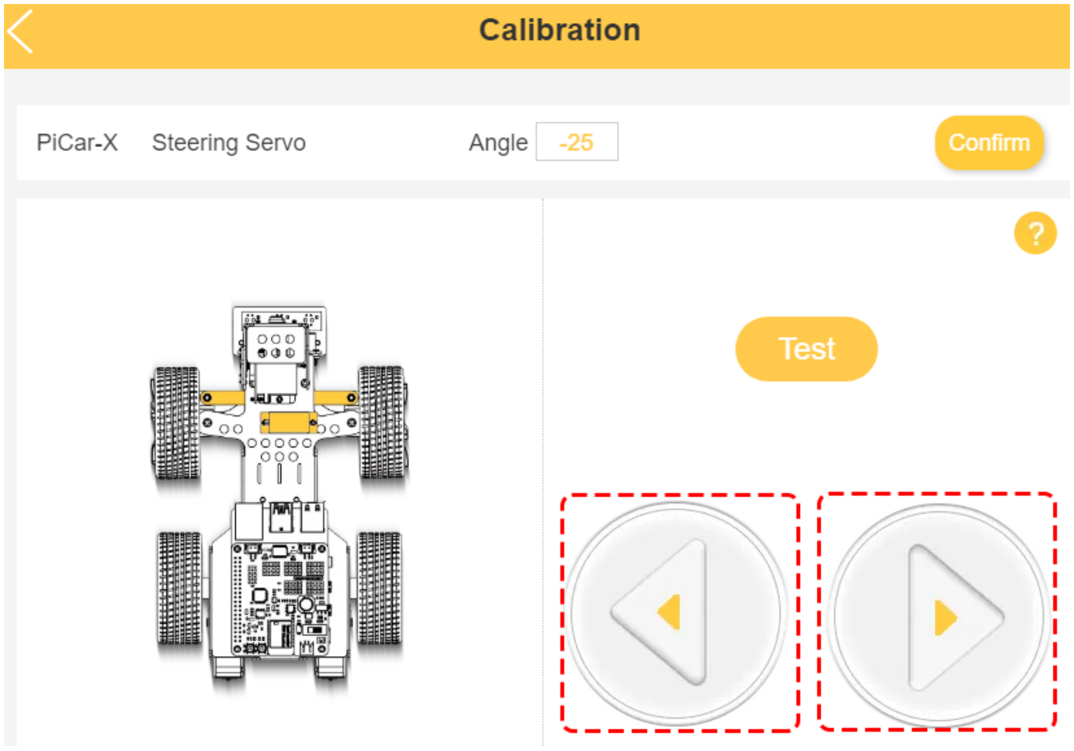
   Note: Calibrating is a micro-adjustment process. It is recommended to take the part off and reassemble it if you click a button to the limit and the part is still off.

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**.

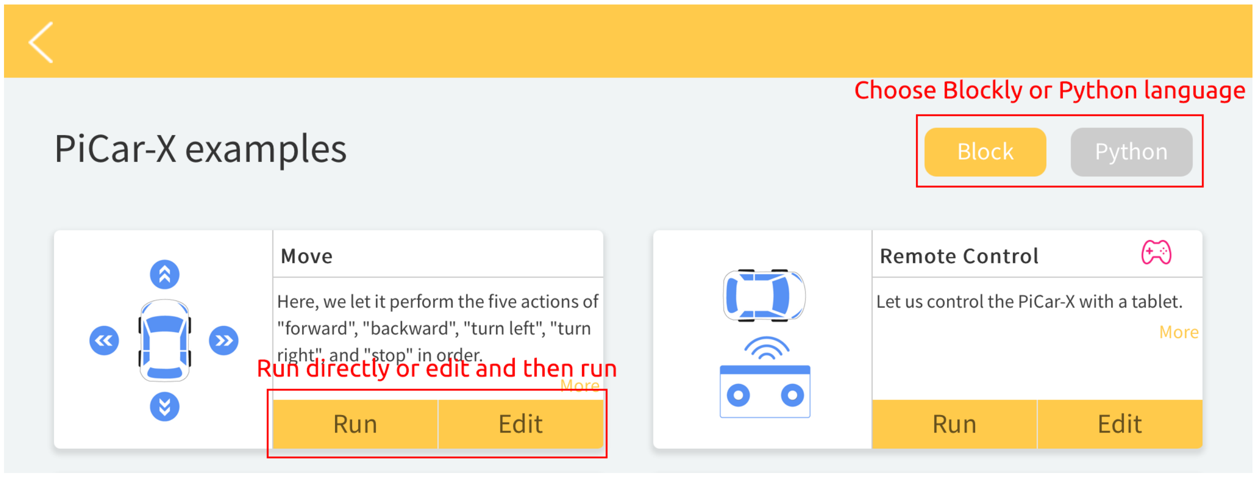


Projects

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.



Here, we let it perform the five actions of "forward", "backward", "turn left", "turn right", and "stop" in order.

Let us control the PiCar-X with a tablet.

Basic

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

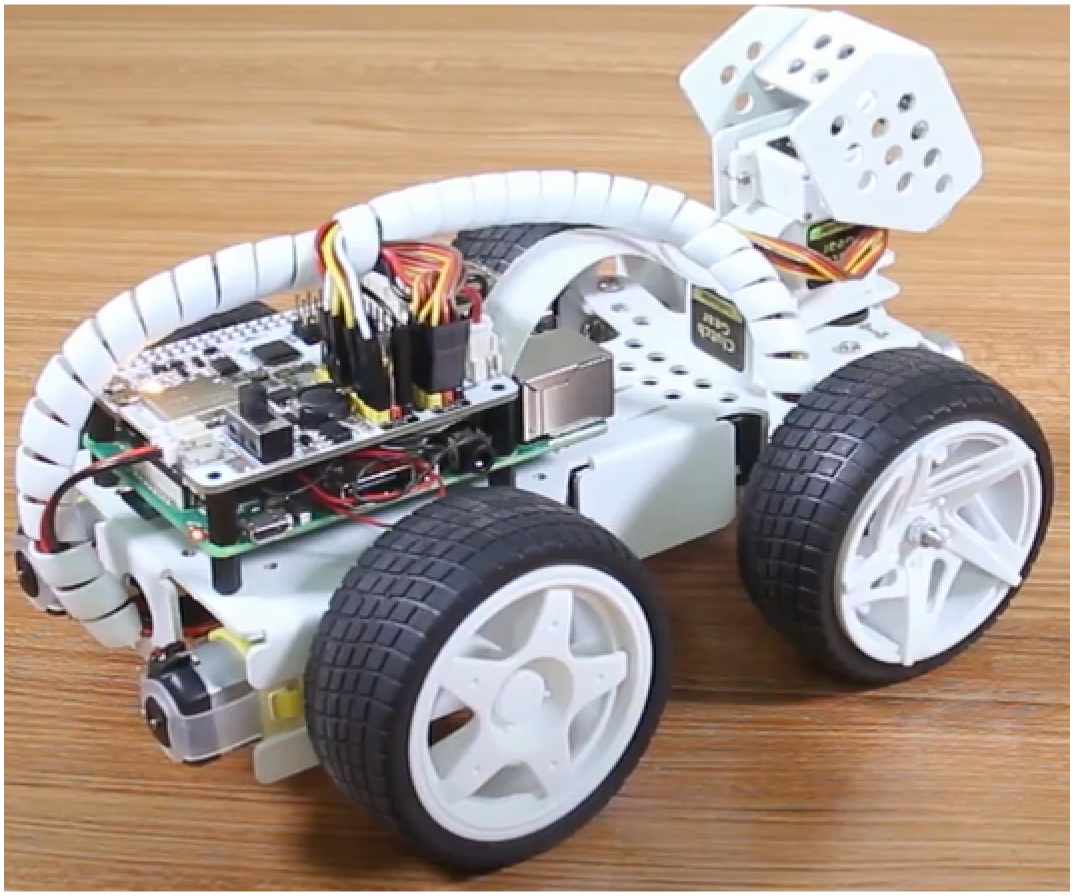
Ready to explore and create with us? Click [] and join today!

5.4 Move

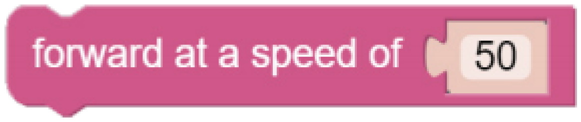
This first project teaches how to program movement actions for the PiCar-X. In this project, the program will tell the PiCar-X to execute five actions in order: “forward”, “backward”, “turn left”, “turn right”, and “stop”.

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

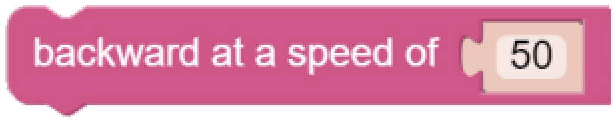
- How to Create a New Project?



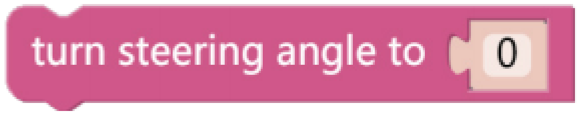
TIPS



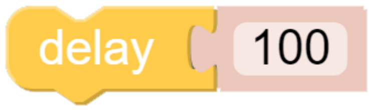
This block will make the PiCar-X move forward at a speed based on a percentage of available power. In the example below "50" is 50% of power, or half-speed.



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.



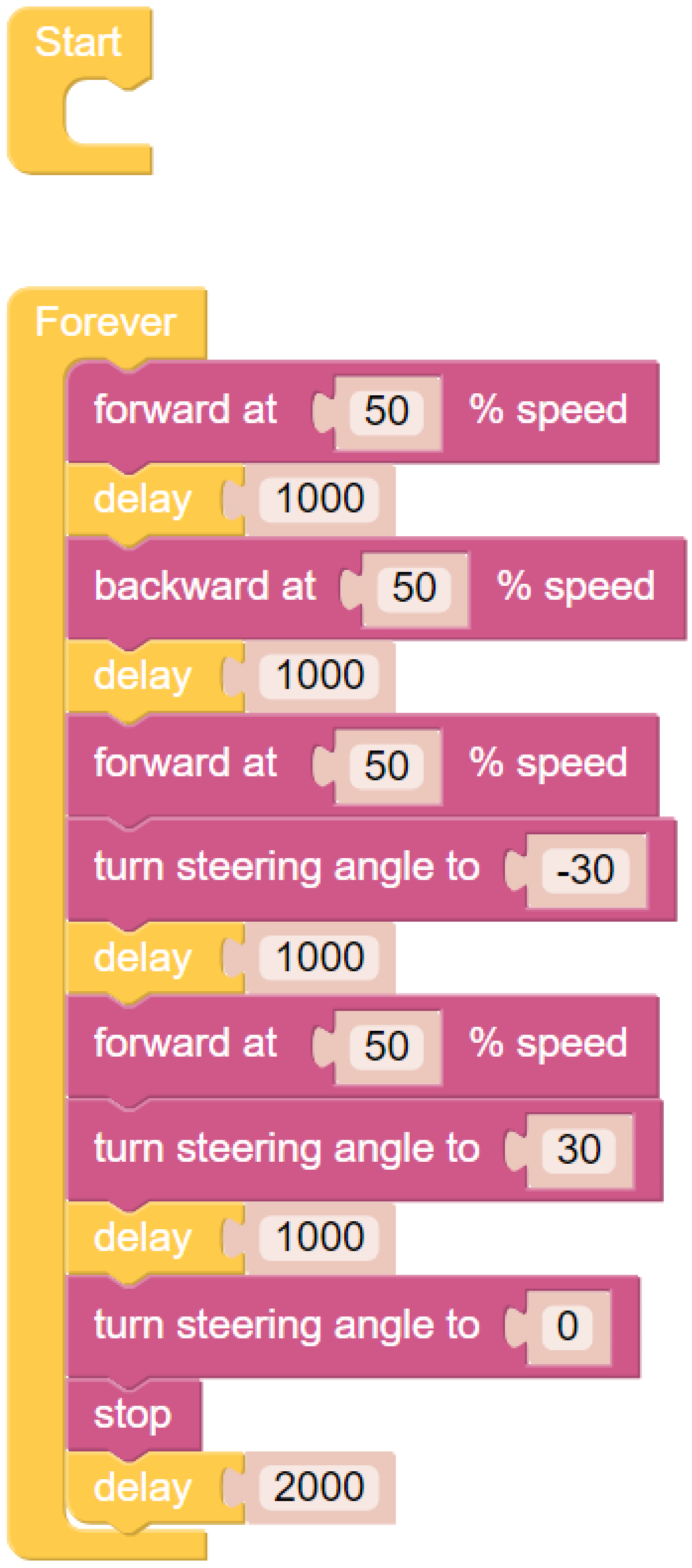
This block will cause a timed break between commands, based on milliseconds. In the example below, the PiCar-X will wait for 1 second (1000 milliseconds) before executing the next command.

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
```

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

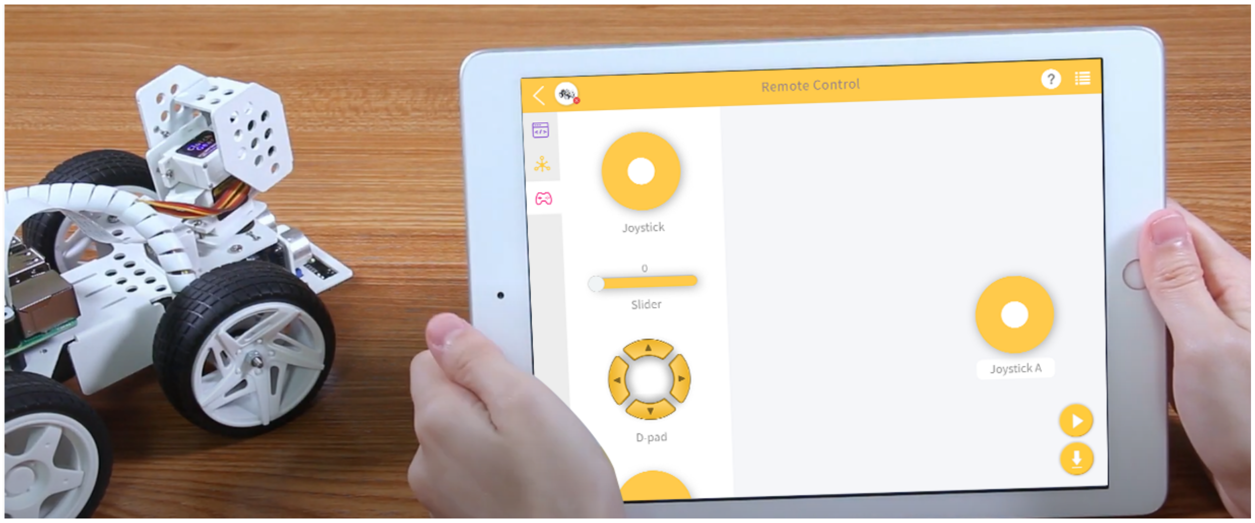
- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

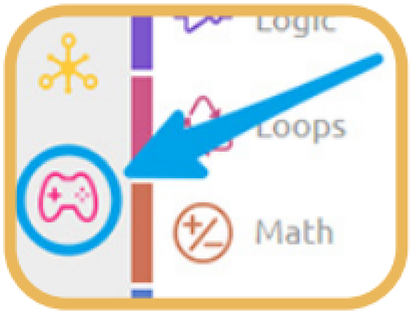
5.5 Remote Control

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:

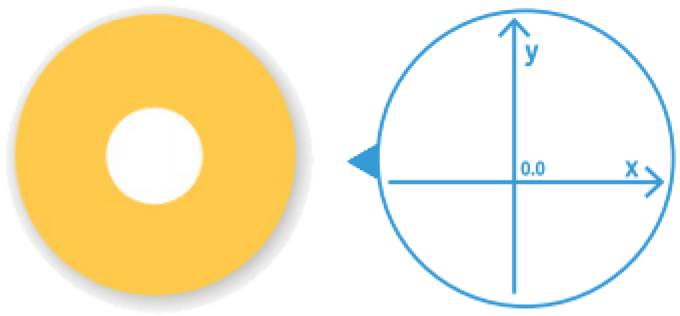
- How to Use the Remote Control Function?



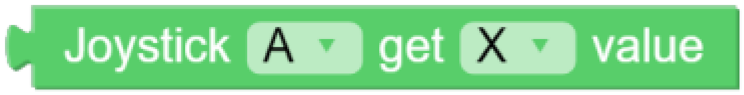
TIPS



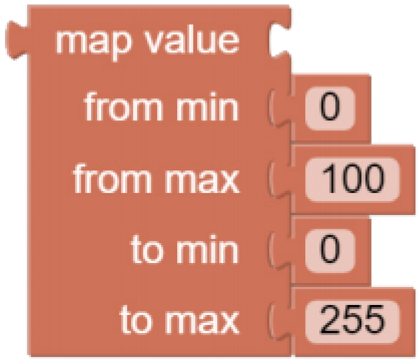
To use the remote control function, open the Remote Control page from the left side of the main page.



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.

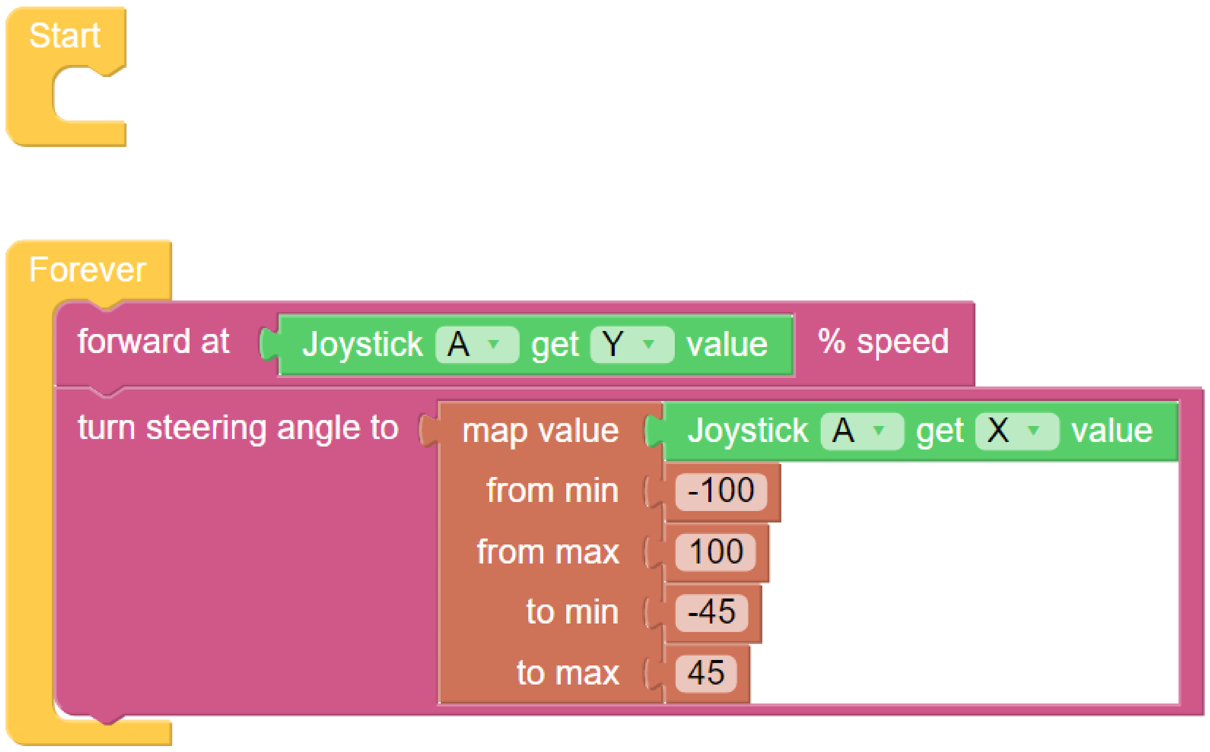


The map value block can remap a number from one range to another. If the range is set to 0 to 100, and the map value number is 50, then it is at a 50% position of the range, or "50". If the range is set to 0 to 255 and the map value number is 50, then it is at a 50% position of the range, or "127.5".

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.


Start
Forever
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

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

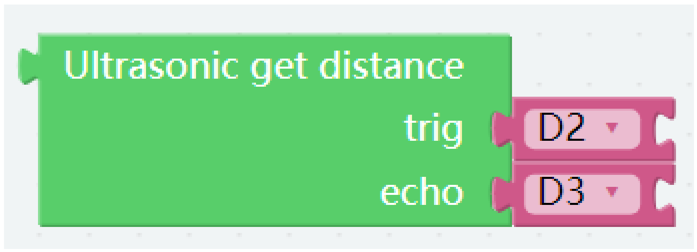
- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

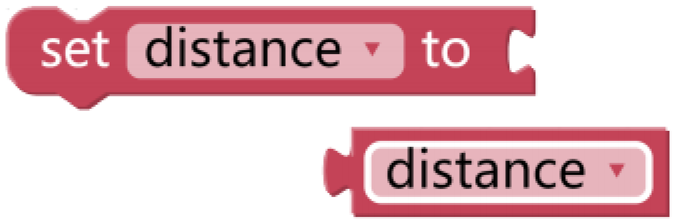
5.6 Test Ultrasonic Module

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

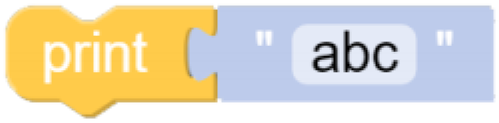


The **Ultrasonic get distance** block will read the distance from the PiCar-X to an obstacle directly ahead.

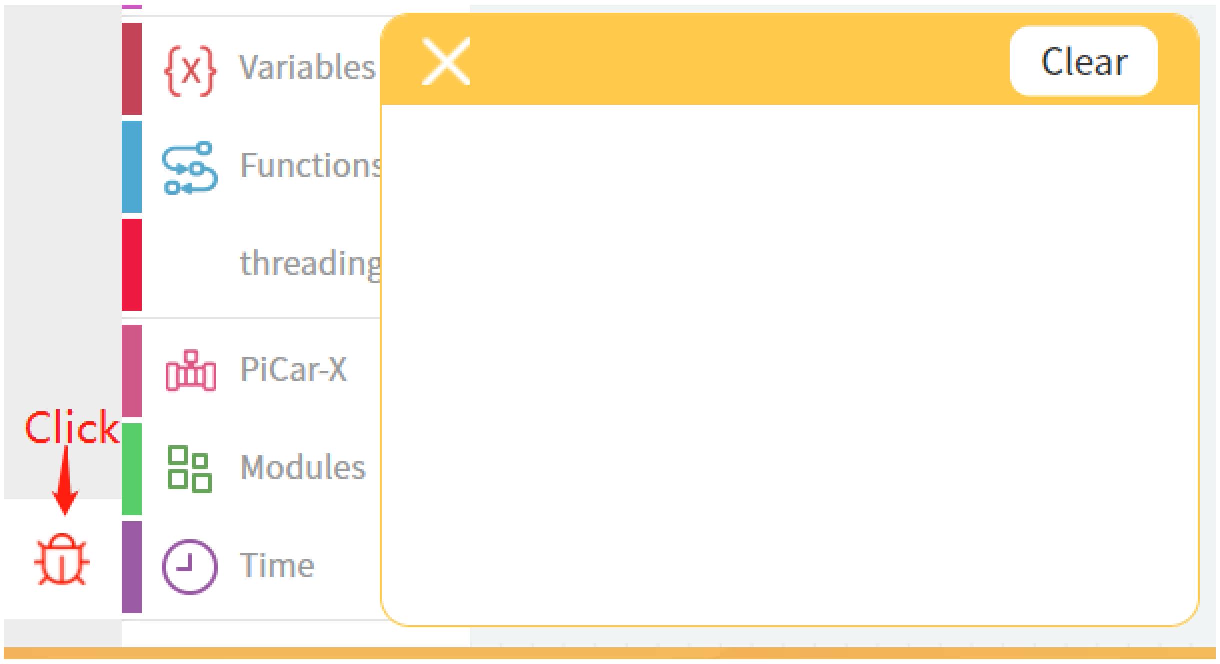


This program is simplified with a **Variable**. For example, when there are multiple functions in a program that each need to use the distance to an obstacle, a **Variable** can be used to report the same distance value to each function, instead of each function reading the same value separately.

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.

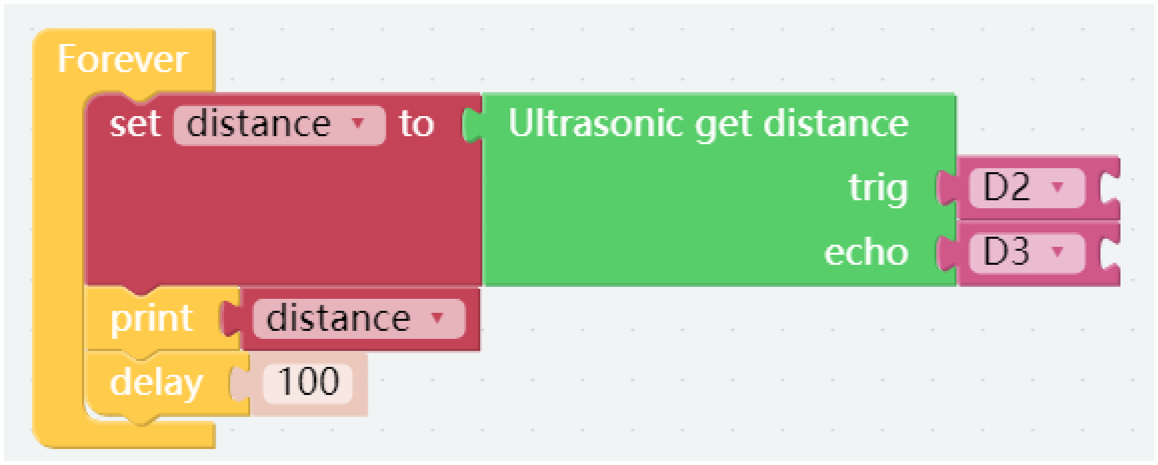


Once the code is running, enable the debug monitor by clicking the **Debug** icon in the bottom left corner.

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.



Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.

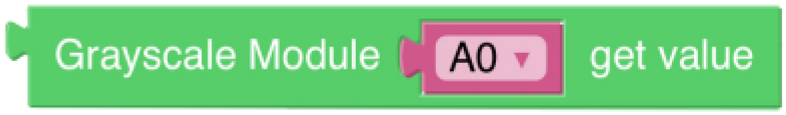
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

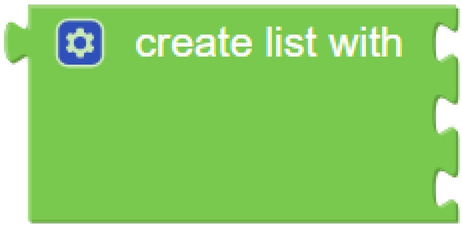
5.7 Test Grayscale Module

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).

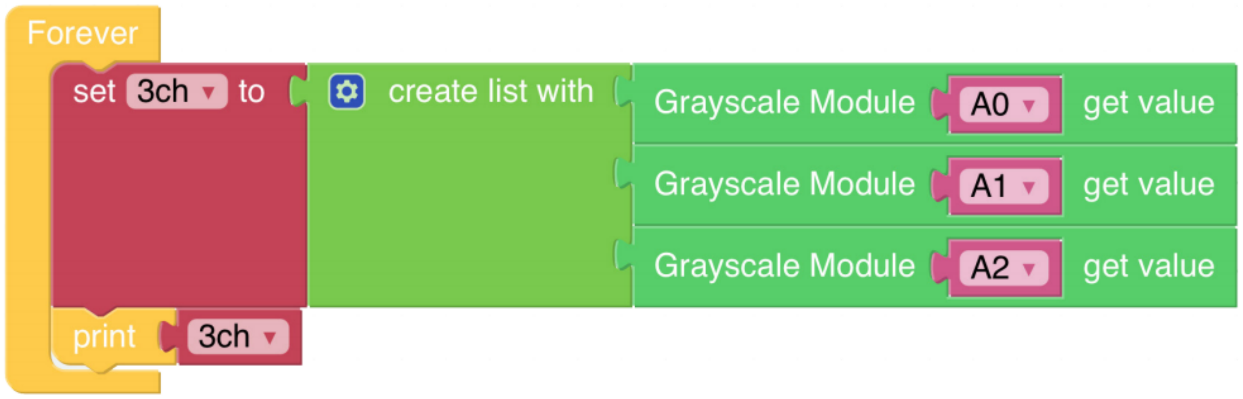


The program is simplified with a **create list with** block. A **List** is used in the same way as a single **Variable**, but in this case a **List** is more efficient than a single **Variable** because the **Grayscale module** will be reporting more than one sensor value. The **create list with** block 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.


Forever
set 3ch to
create list with
Grayscale Module A0 get value
Grayscale Module A1 get value
Grayscale Module A2 get value
print 3ch

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

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.

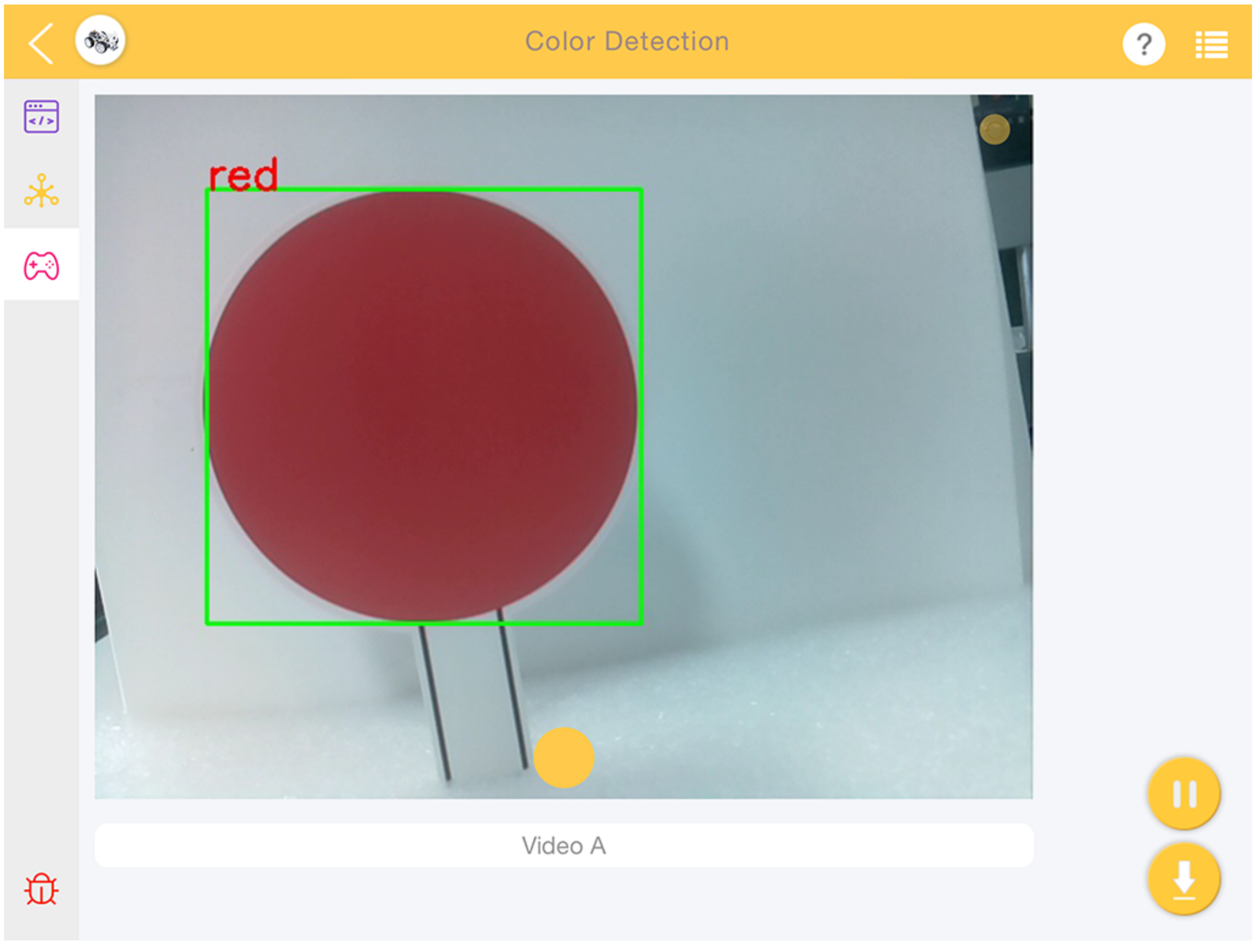
Note: Before attempting this section, make sure that the Raspberry Pi Camera's FFC cable is properly and securely connected. For detailed instructions on securely connecting the FCC cable, please reference: *Assemble the PiCar-X*.

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.

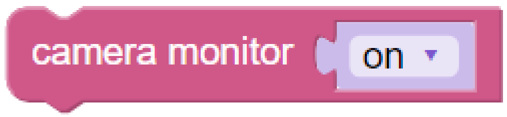
- `[PDF]Color Cards`

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.

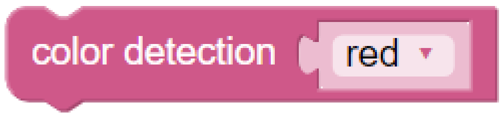


TIPS

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.

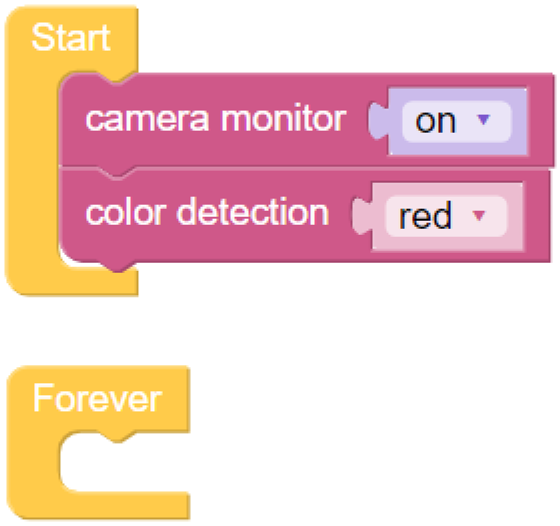


Use the **color detection** block to enable the color detection. Note: only one color can be detected at a time.

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.



Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

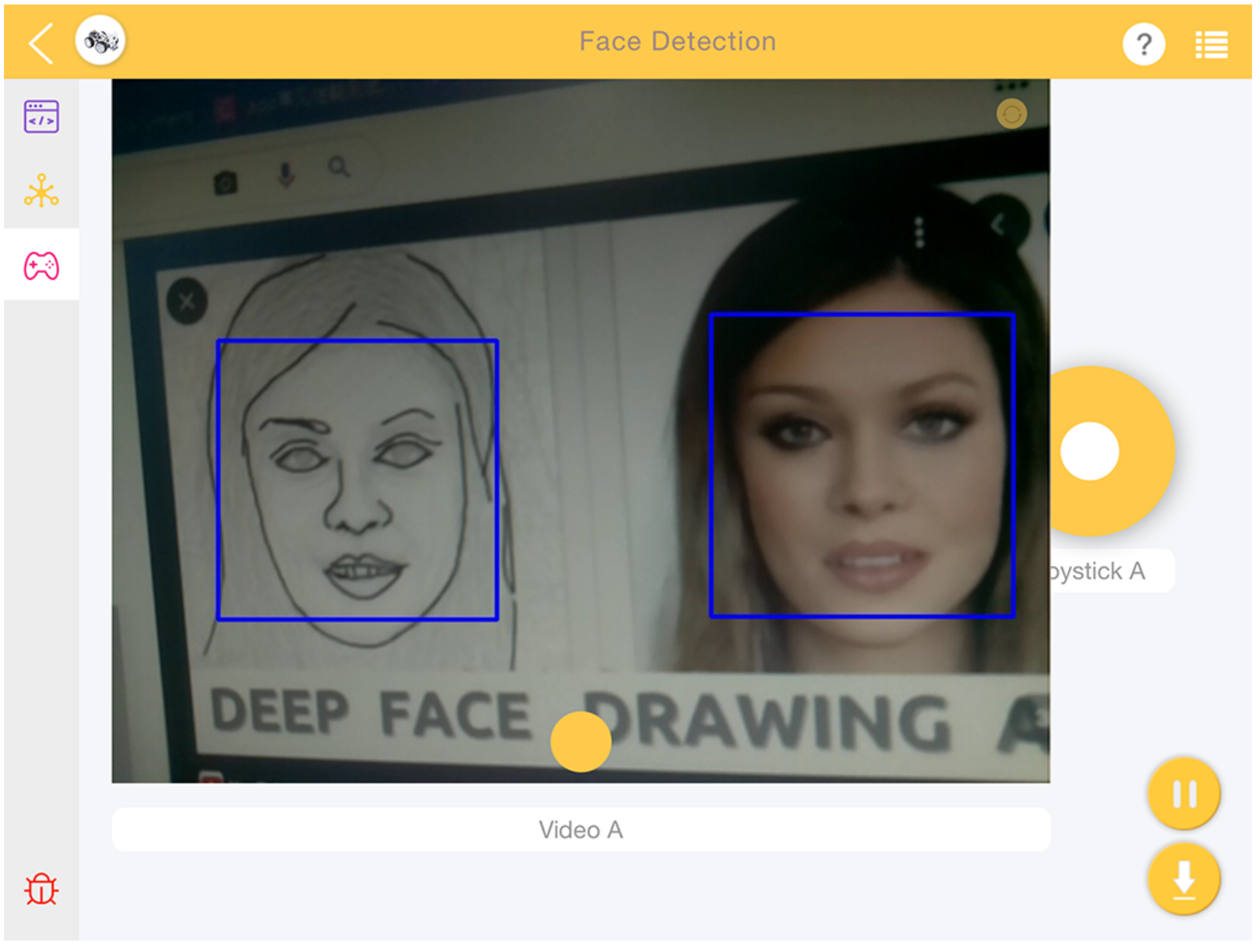
- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

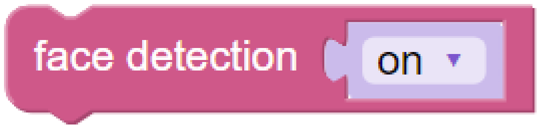
5.9 Face Detection

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.

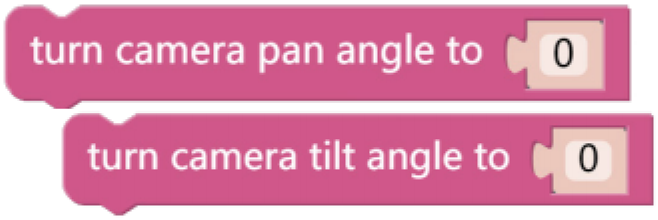
For more information on how to use the Video widget, please reference the tutorial on Ezblock video here: How to Use the Video Function?.



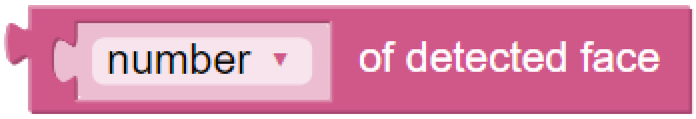
TIPS



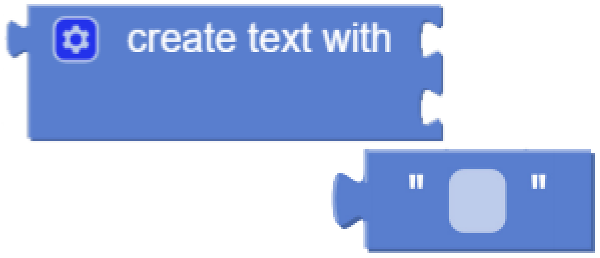
Set the **face detection** widget to **on** to enable facial detection.



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.

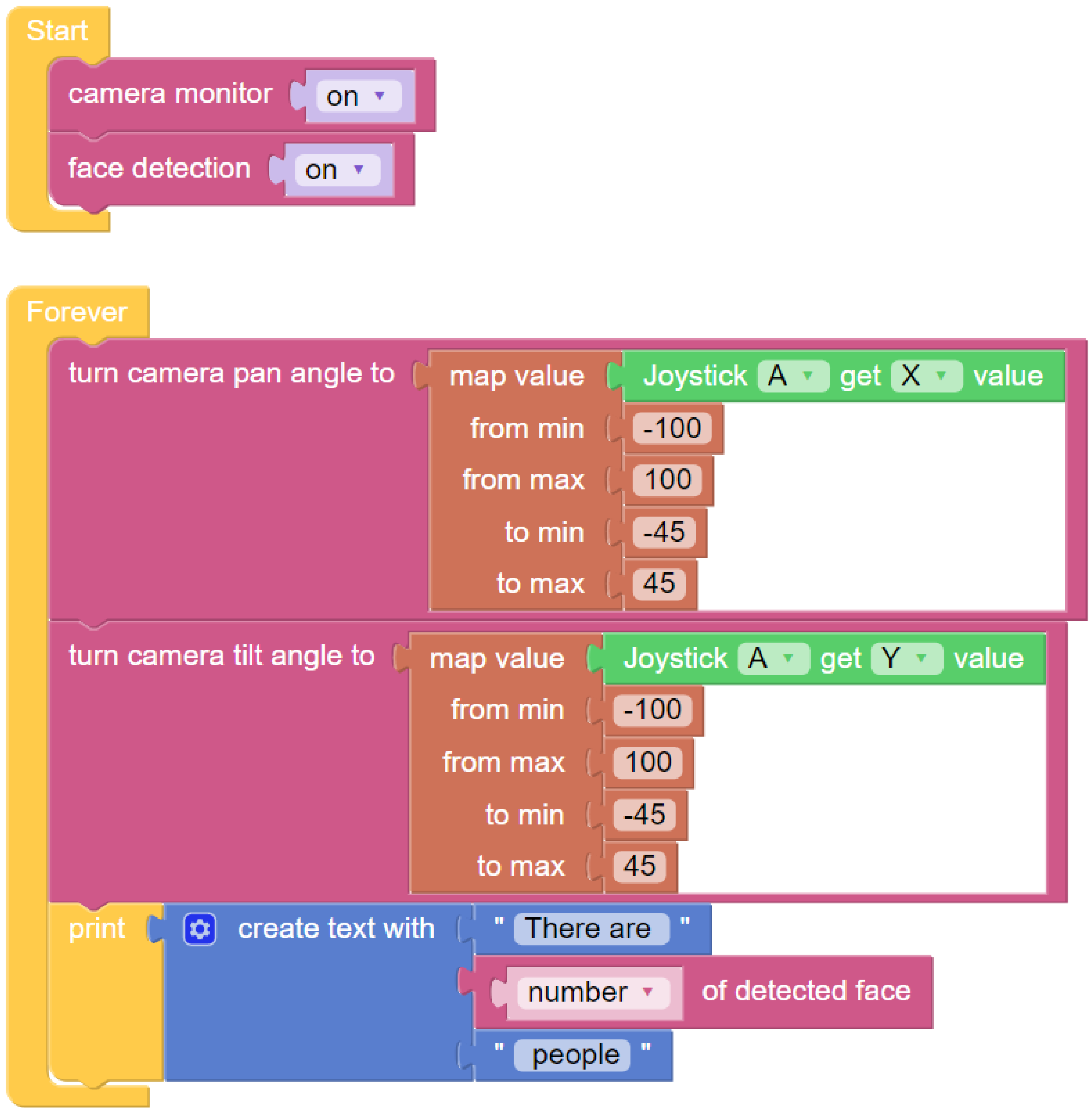


Use the **create text with** block to print the combination of **text** and of **detected face** data.

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.



```
Start
  camera monitor  on
  face detection  on

Forever
  turn camera pan angle to  map value  Joystick A get X value
                            from min   -100
                            from max   100
                            to min     -45
                            to max     45
  turn camera tilt angle to  map value  Joystick A get Y value
                             from min   -100
                             from max   100
                             to min     -45
                             to max     45
  print  create text with  " There are "
                           number of detected face
                           " people "
```

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

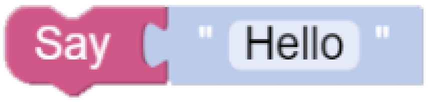
- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

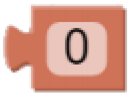
5.10 Sound Effect

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.

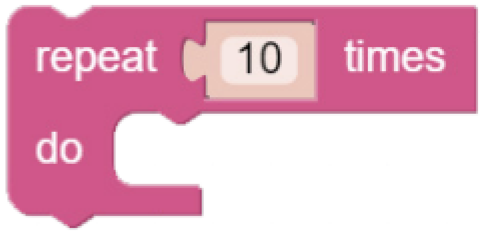
TIPS



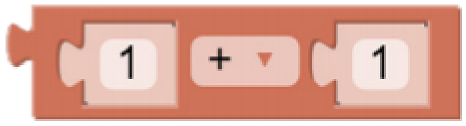
Use the **say** block with a **text** block to write a sentence for the PiCar-X to say. The **say** block can be used with text or numbers.



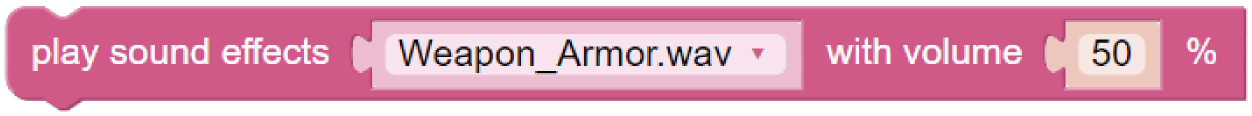
The **number** block.



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 “÷ “.



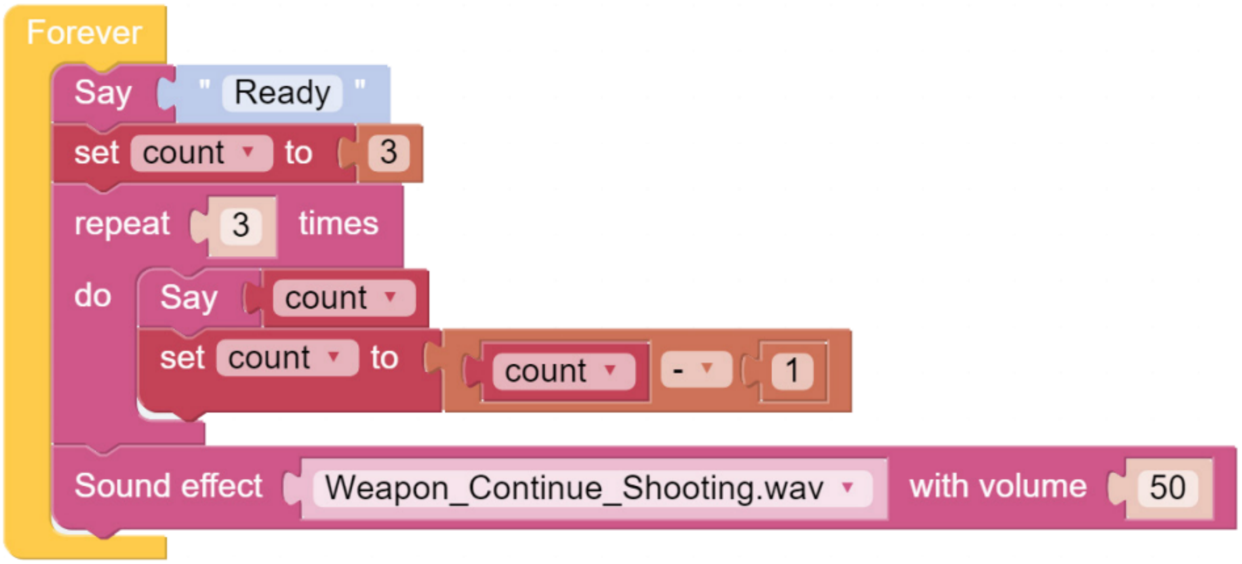
The play **sound effects - with volume - %** block has preset sound effects, such as a siren sound, a gun sound, and others. The range of the volume can be set from 0 to 100.

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.



Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

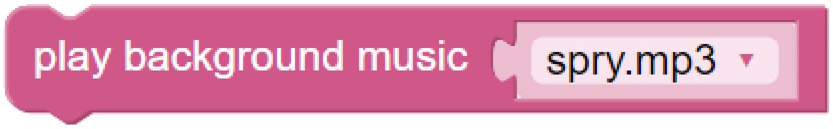
5.11 Background Music

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.

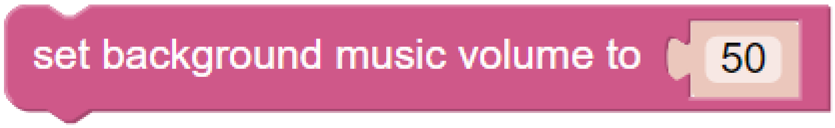
- How to Use the Remote Control Function?

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

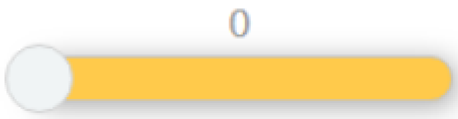
TIPS



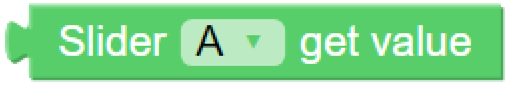
The **play background music** block will need to be added to the **Start** function. Use the drop-down menu to choose different background music for the PiCar-X to play.



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.

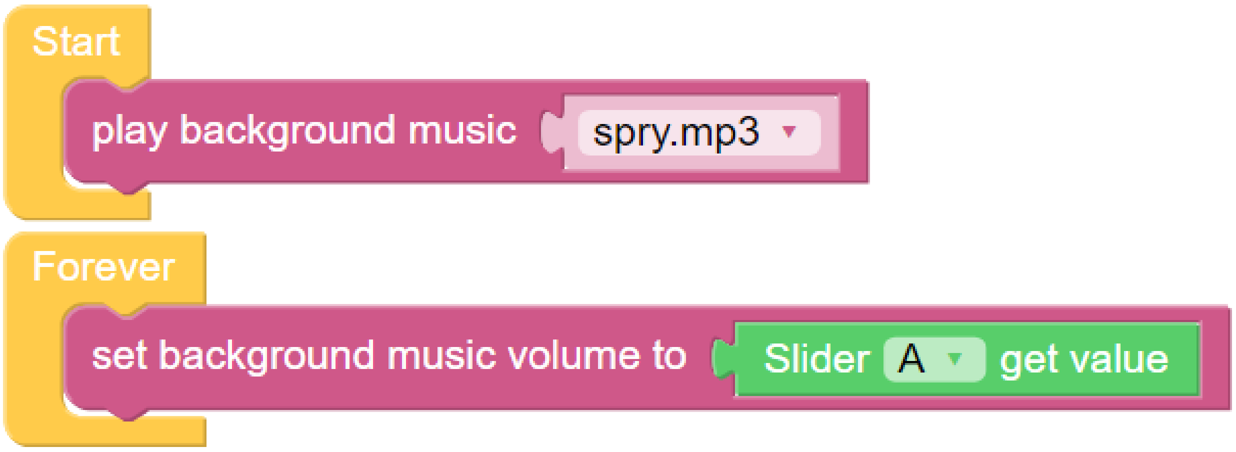


The **slider [A] get value** block will read the slider value. The example above has slider 'A' selected. If there are multiple sliders, use the drop-down menu to select the appropriate one.

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.



Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.

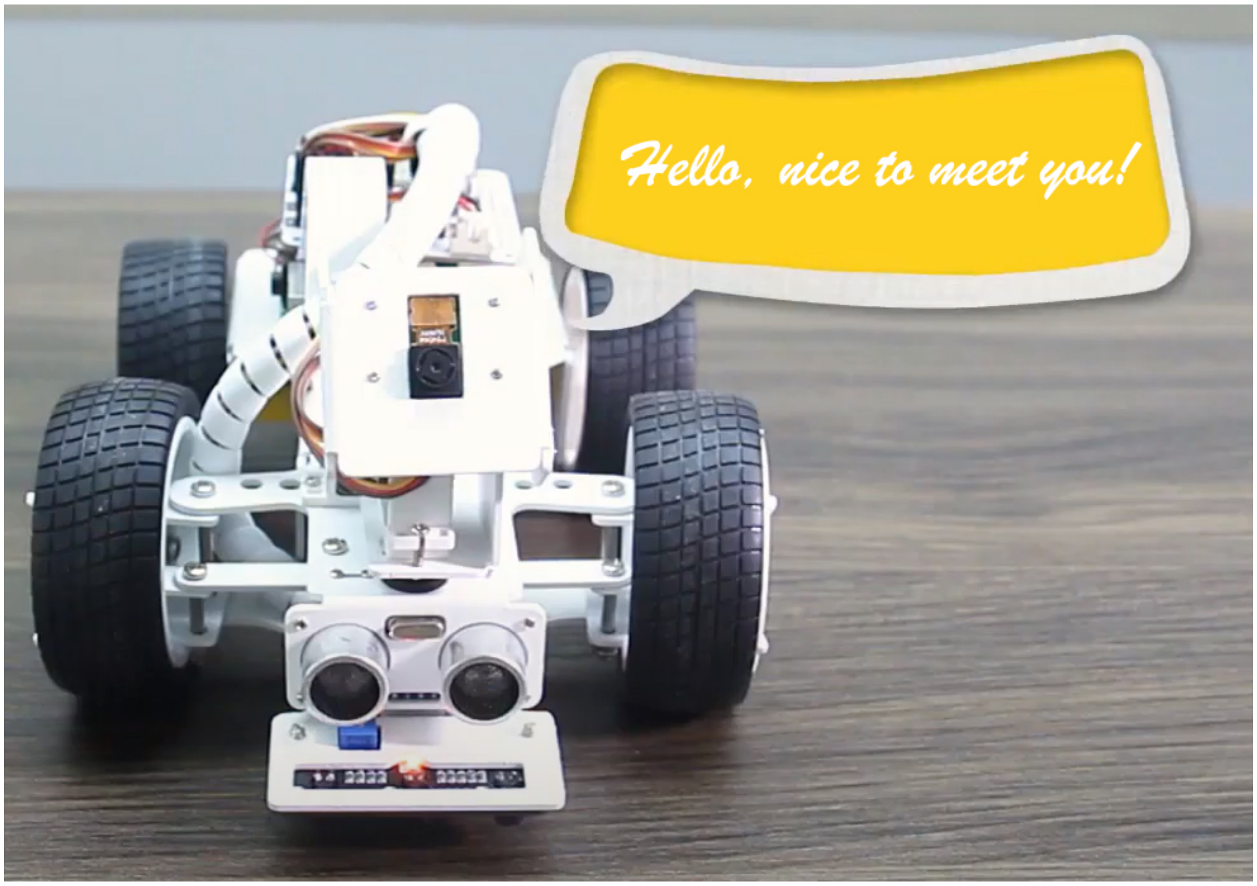
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

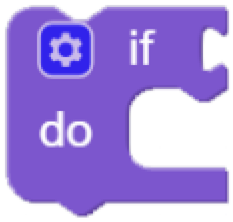
5.12 Say Hello

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!".

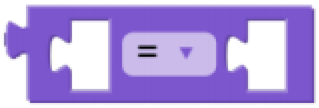
- How to Use the Video Function?
- How to Use the Remote Control Function?



TIPS



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

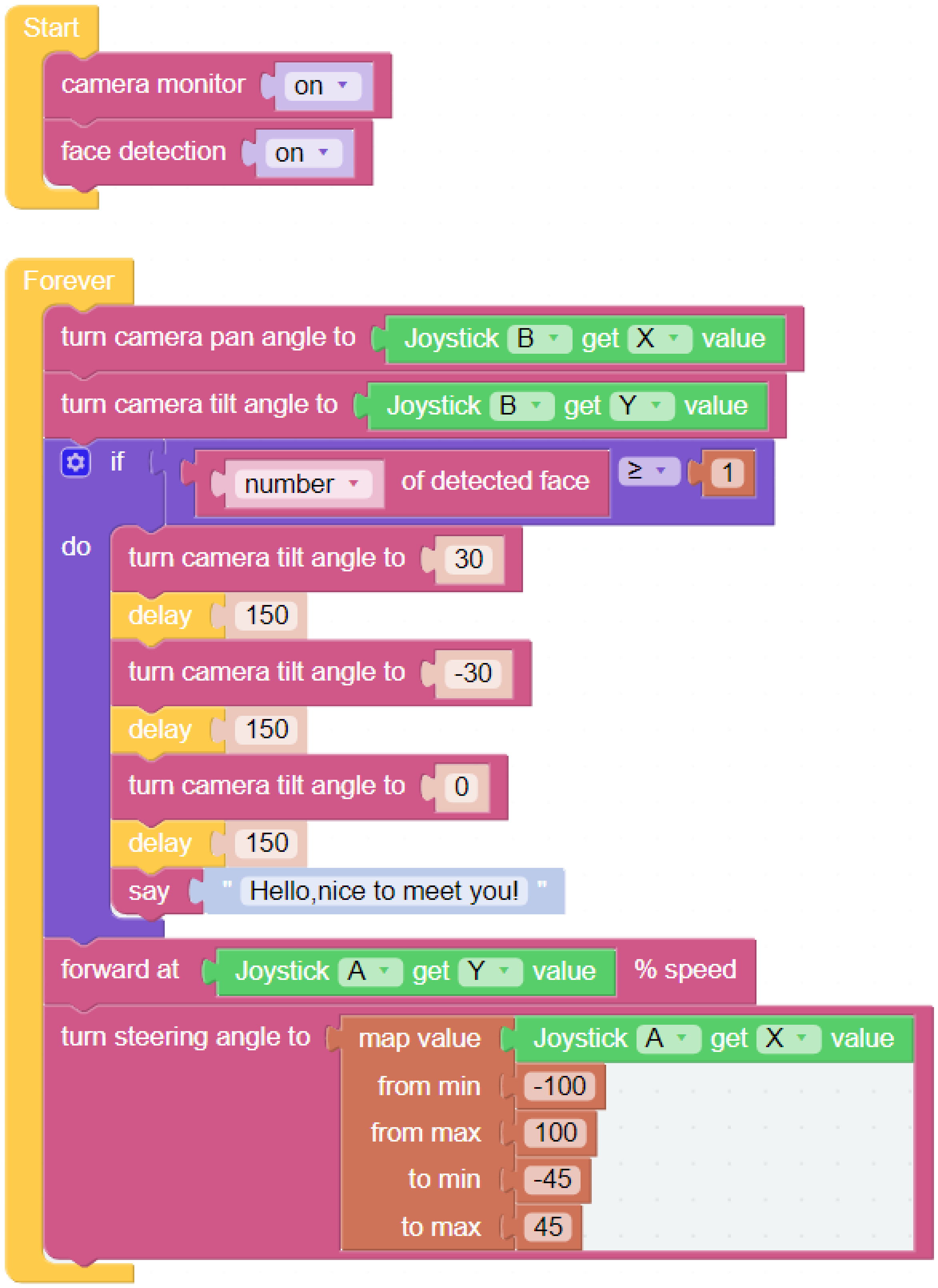


The **conditional statements** block is used in conjunction with the **if do** block. The conditions can be “=”, “>”, “<”, ” “, ” “, or ” “.

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
```

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

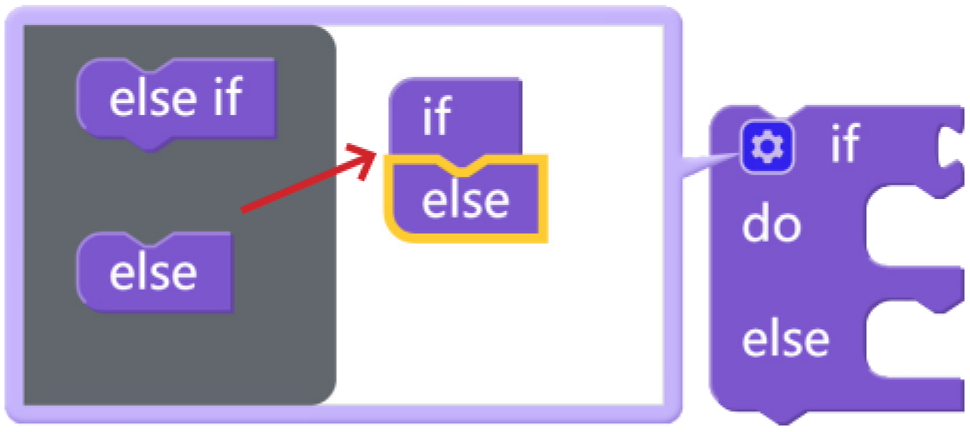
- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

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

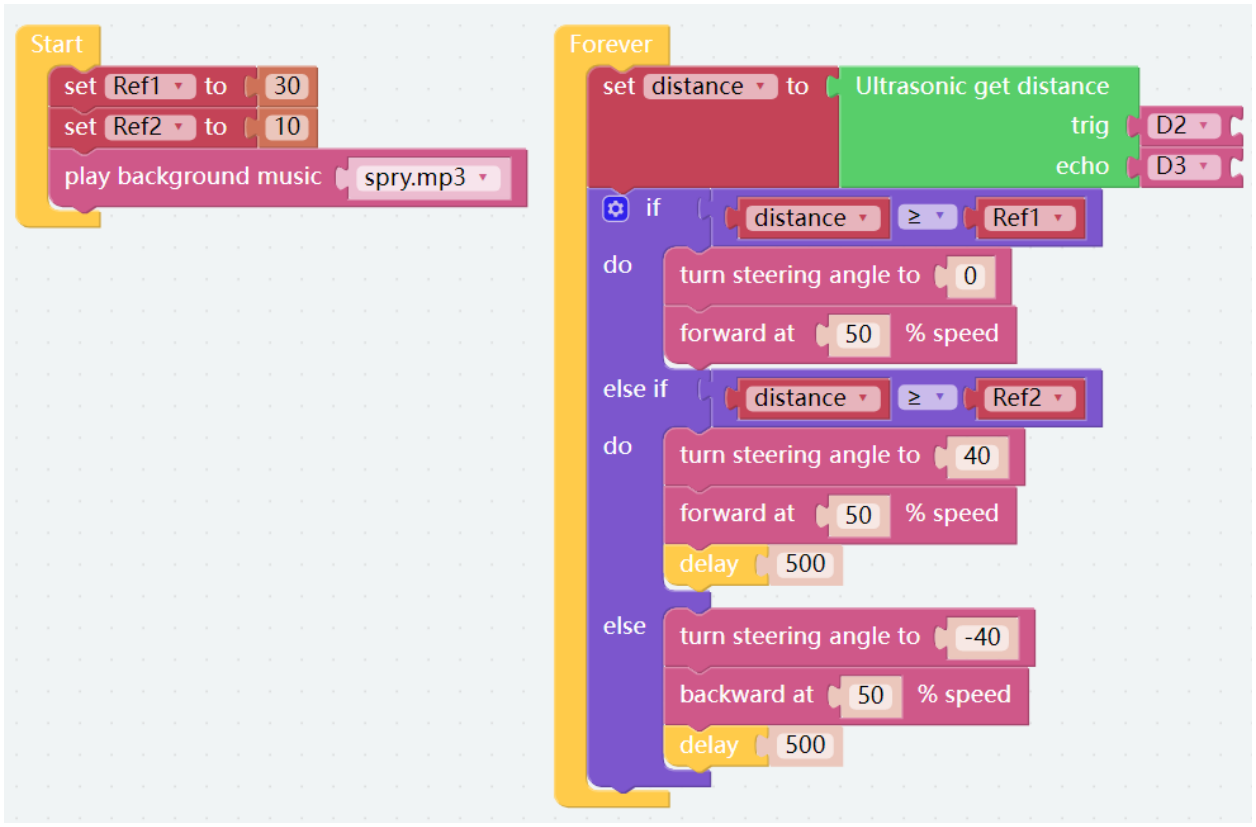


To implement multiple conditional judgments, change the simple if do block into an if else do / else if do block. This is done by clicking on the setting icon as shown above.

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.



```
Start
    set Ref1 to 30
    set Ref2 to 10
    play background music spry.mp3

Forever
    set distance to Ultrasonic get distance trig D2 echo D3
    if distance ≥ Ref1
    do
        turn steering angle to 0
        forward at 50 % speed
    else if distance ≥ Ref2
    do
        turn steering angle to 40
        forward at 50 % speed
        delay 500
    else
        turn steering angle to -40
        backward at 50 % speed
        delay 500
```

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

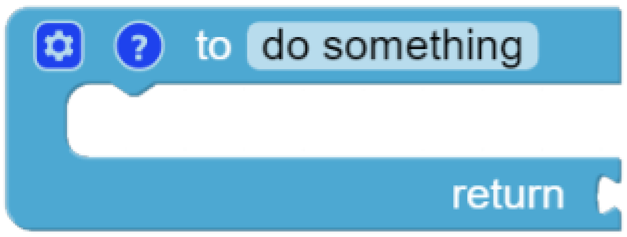
- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

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

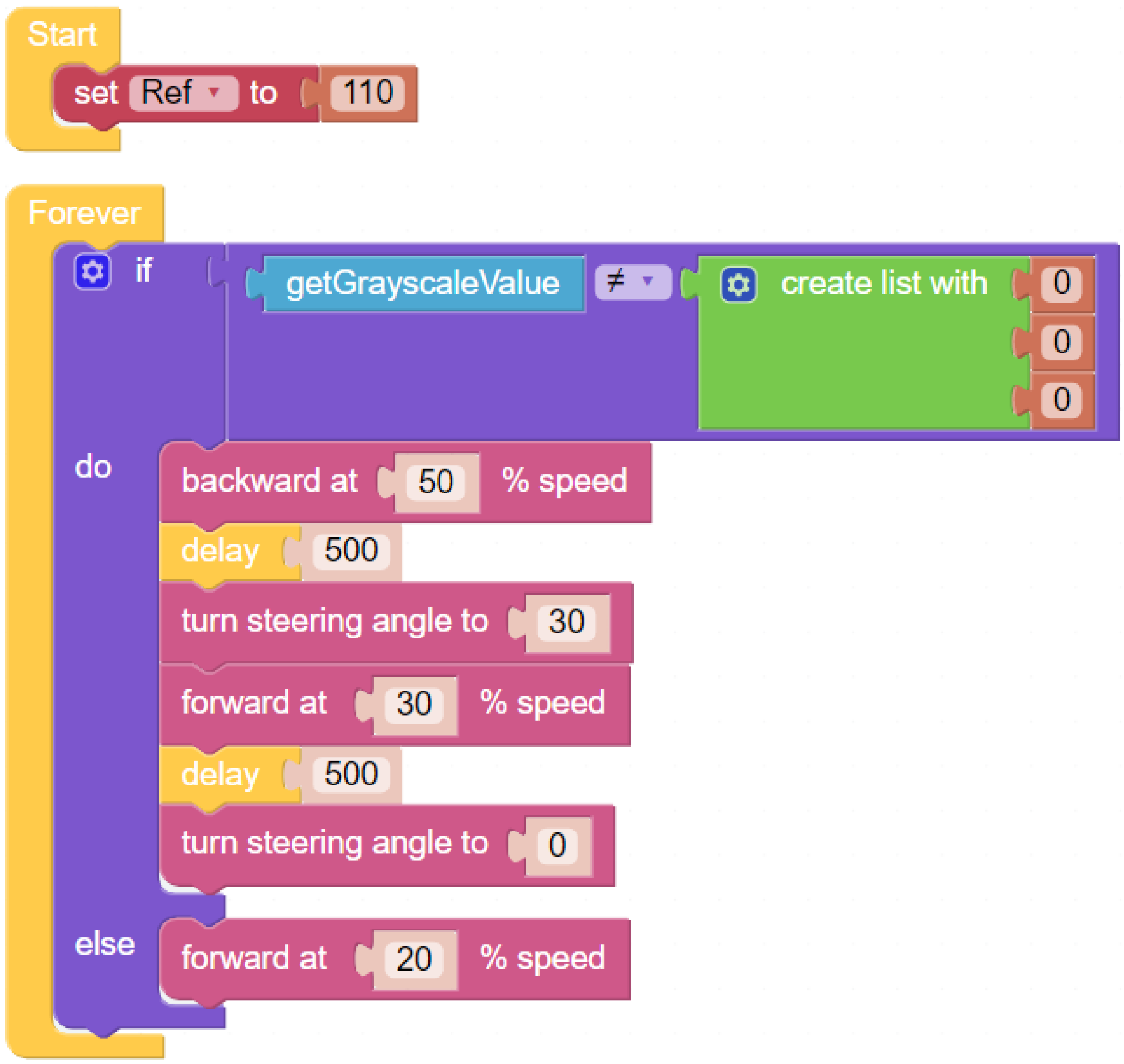


The **grayscale module** will be performing the same operation multiple times. To simplify the program, this project introduces a **function** that will return a **list** variable to the **do forever** block.

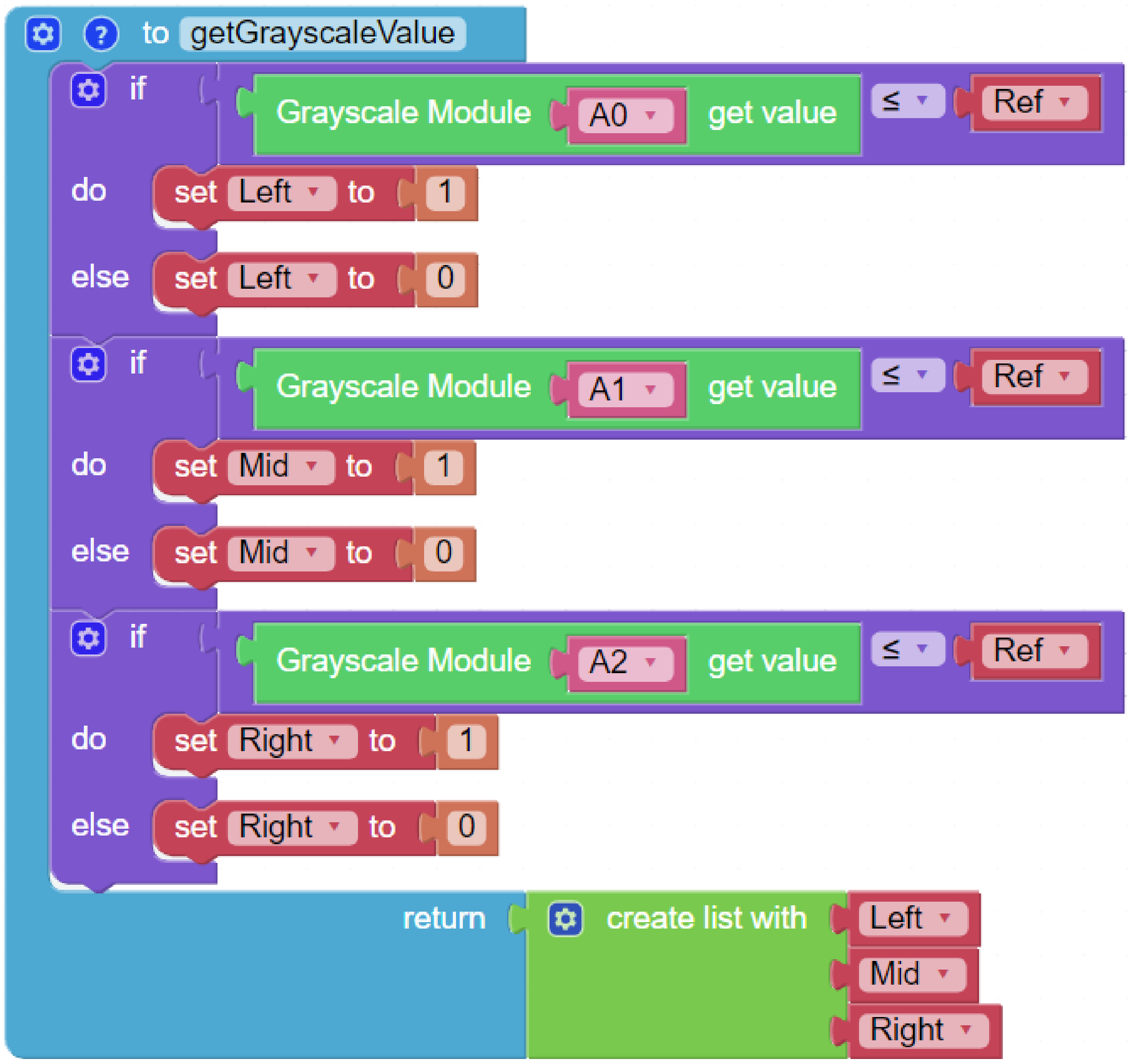
EXAMPLE

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.



```
Start
    set Ref to 110

Forever
    if getGrayscaleValue ≠ create list with 0 0 0
    do  backward at 50 % speed
        delay 500
        turn steering angle to 30
        forward at 30 % speed
        delay 500
        turn steering angle to 0
    else
        forward at 20 % speed
```



```
to getGrayscaleValue
    if Grayscale Module A0 get value ≤ Ref
    do   set Left to 1
    else set Left to 0
    if Grayscale Module A1 get value ≤ Ref
    do   set Mid to 1
    else set Mid to 0
    if Grayscale Module A2 get value ≤ Ref
    do   set Right to 1
    else set Right to 0
    return create list with Left
                            Mid
                            Right
```

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

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.

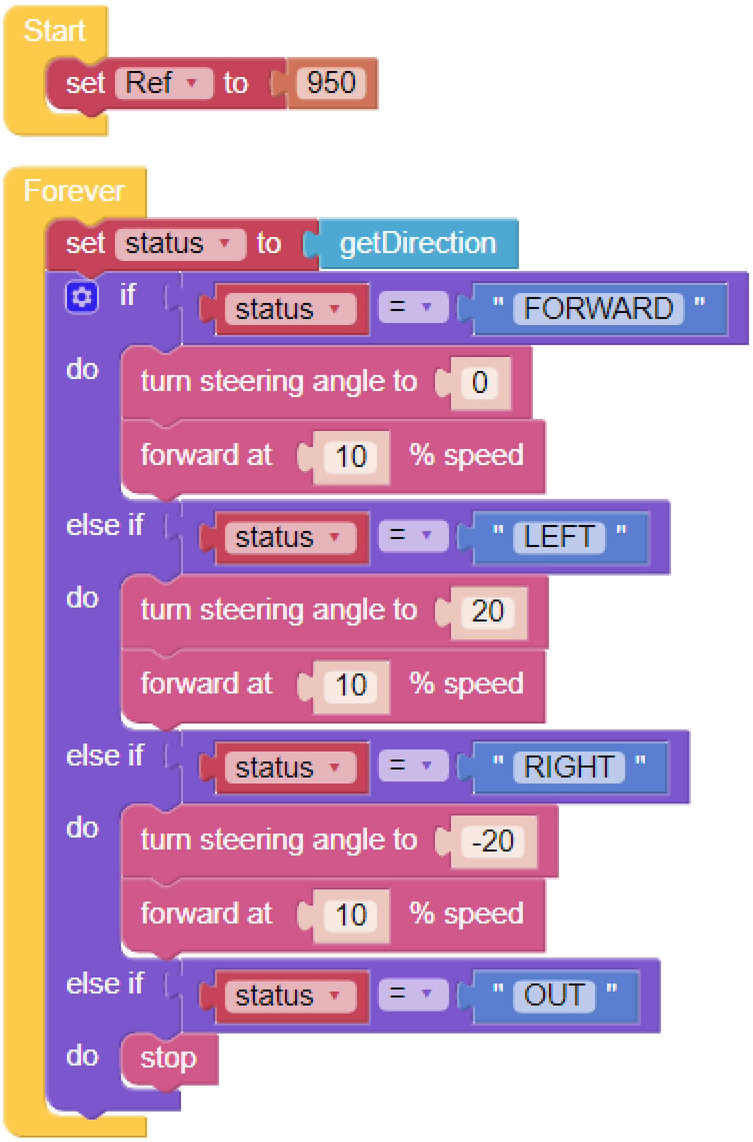
TIPS

- **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.

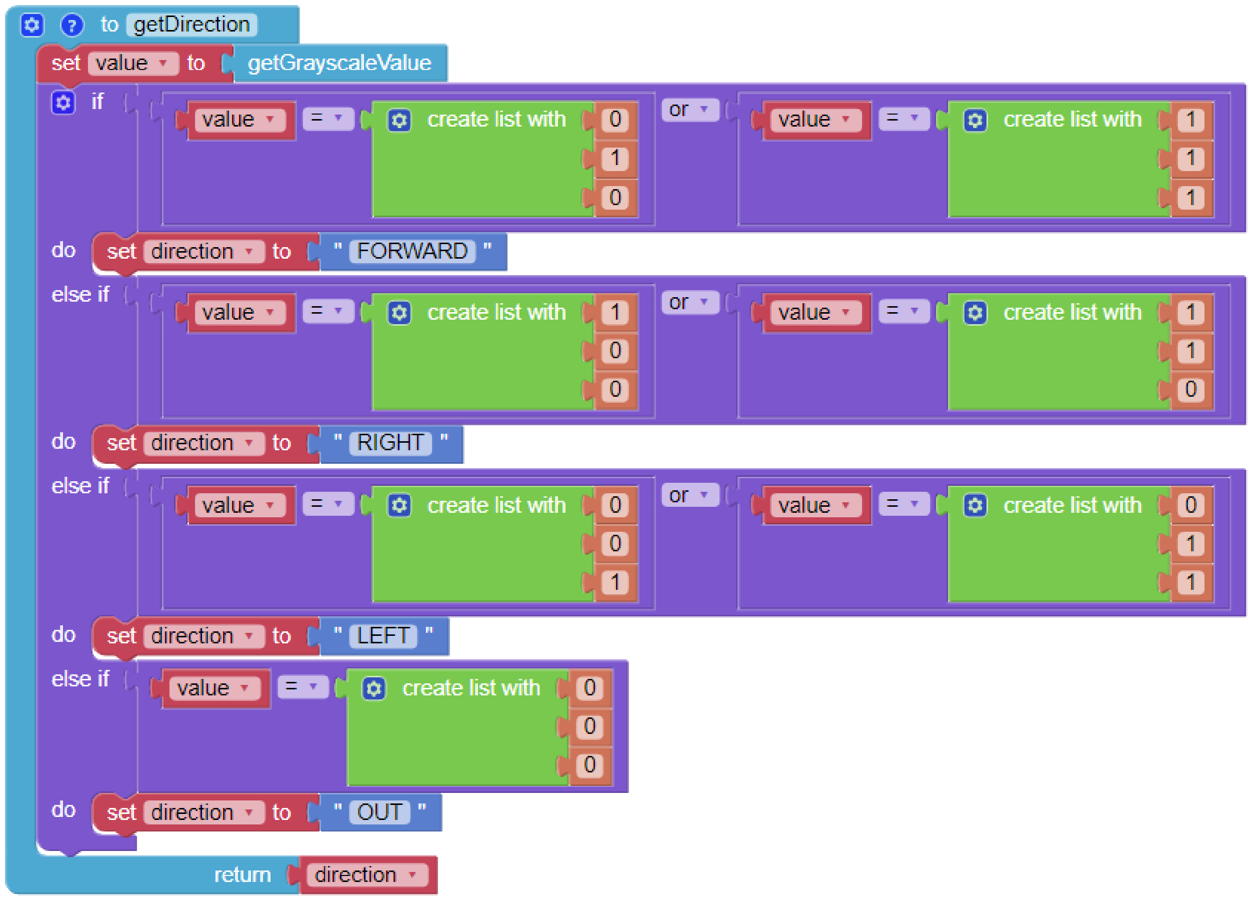
EXAMPLE

Note:

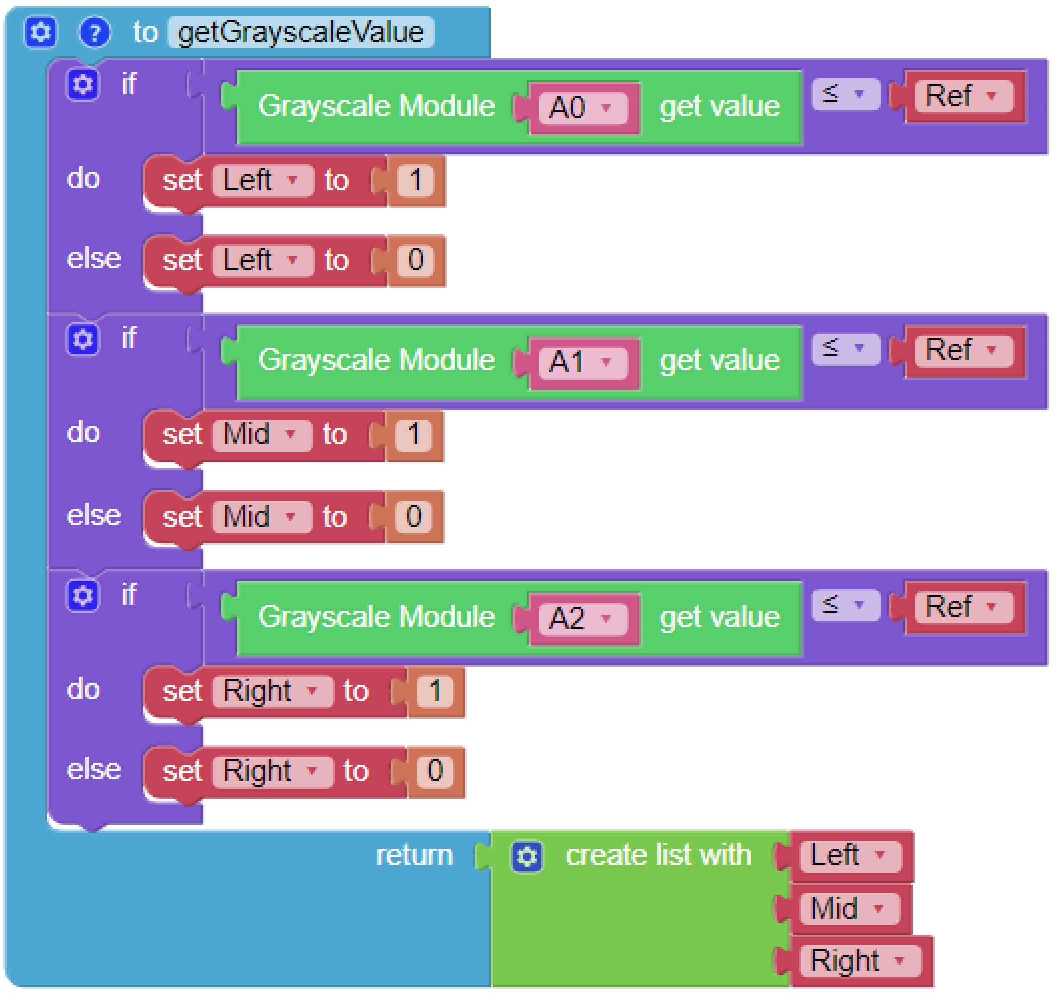
- You can write the program according to the following picture, please refer to the tutorial: How to Create a New Project?.
- Or find the code with the same name on the **Examples** page of the EzBlock Studio and click **Run** or **Edit** directly.



```
Start
    set Ref to 950

Forever
    set status to getDirection
    if status = "FORWARD"
    do  turn steering angle to 0
        forward at 10 % speed
    else if status = "LEFT"
    do  turn steering angle to 20
        forward at 10 % speed
    else if status = "RIGHT"
    do  turn steering angle to -20
        forward at 10 % speed
    else if status = "OUT"
    do  stop
```



```
to getDirection
  set value to getGrayscaleValue
  if value = create list with 0, 1, 0 or value = create list with 1, 1, 1
  do set direction to "FORWARD"
  else if value = create list with 1, 0, 0 or value = create list with 1, 1, 0
  do set direction to "RIGHT"
  else if value = create list with 0, 0, 1 or value = create list with 0, 1, 1
  do set direction to "LEFT"
  else if value = create list with 0, 0, 0
  do set direction to "OUT"
  return direction
```



```
to getGrayscaleValue
    if  Grayscale Module A0 get value ≤ Ref
    do  set Left to 1
    else  set Left to 0
    if  Grayscale Module A1 get value ≤ Ref
    do  set Mid to 1
    else  set Mid to 0
    if  Grayscale Module A2 get value ≤ Ref
    do  set Right to 1
    else  set Right to 0
    return  create list with  Left
                              Mid
                              Right
```

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

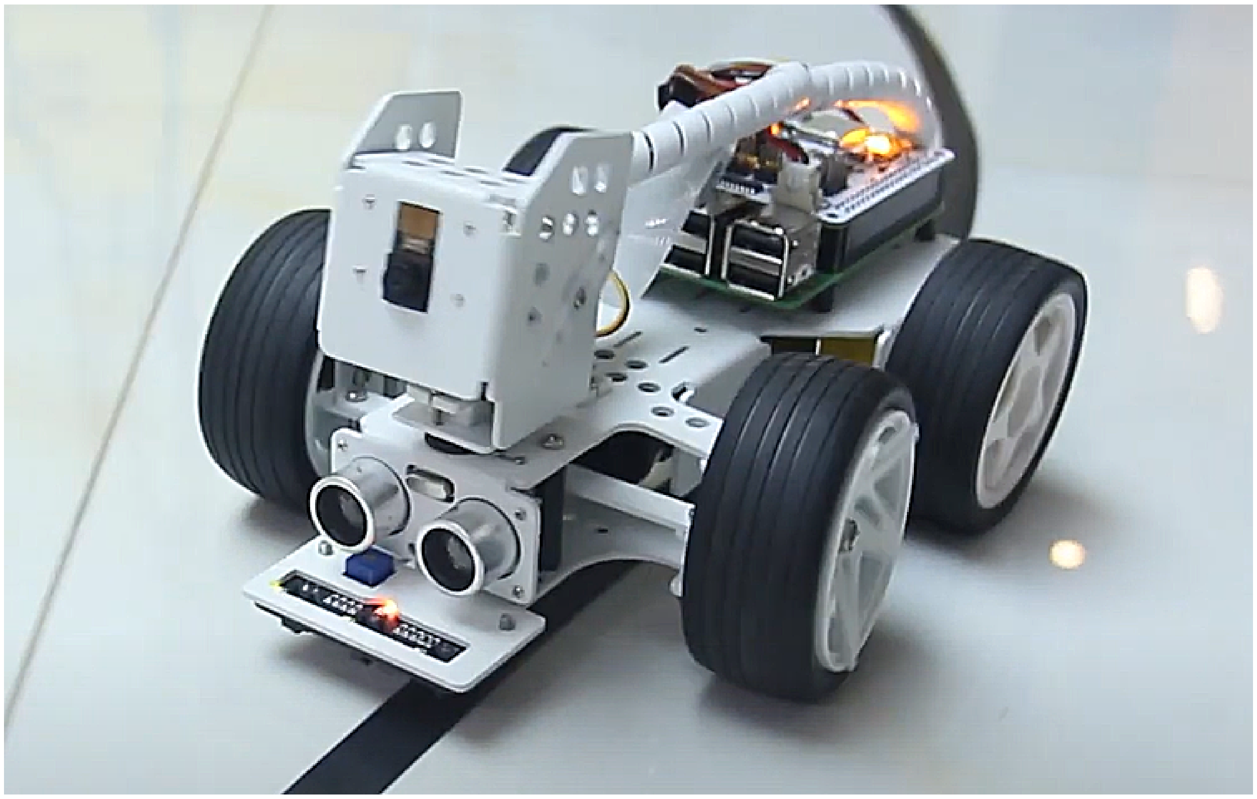
Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

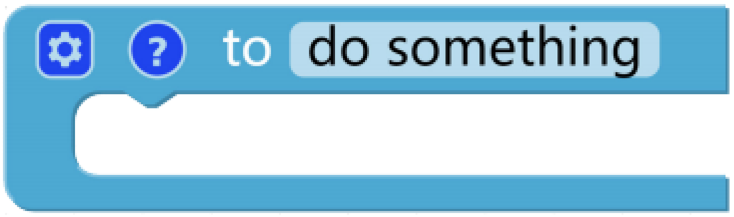
5.16 Minecart Plus

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.

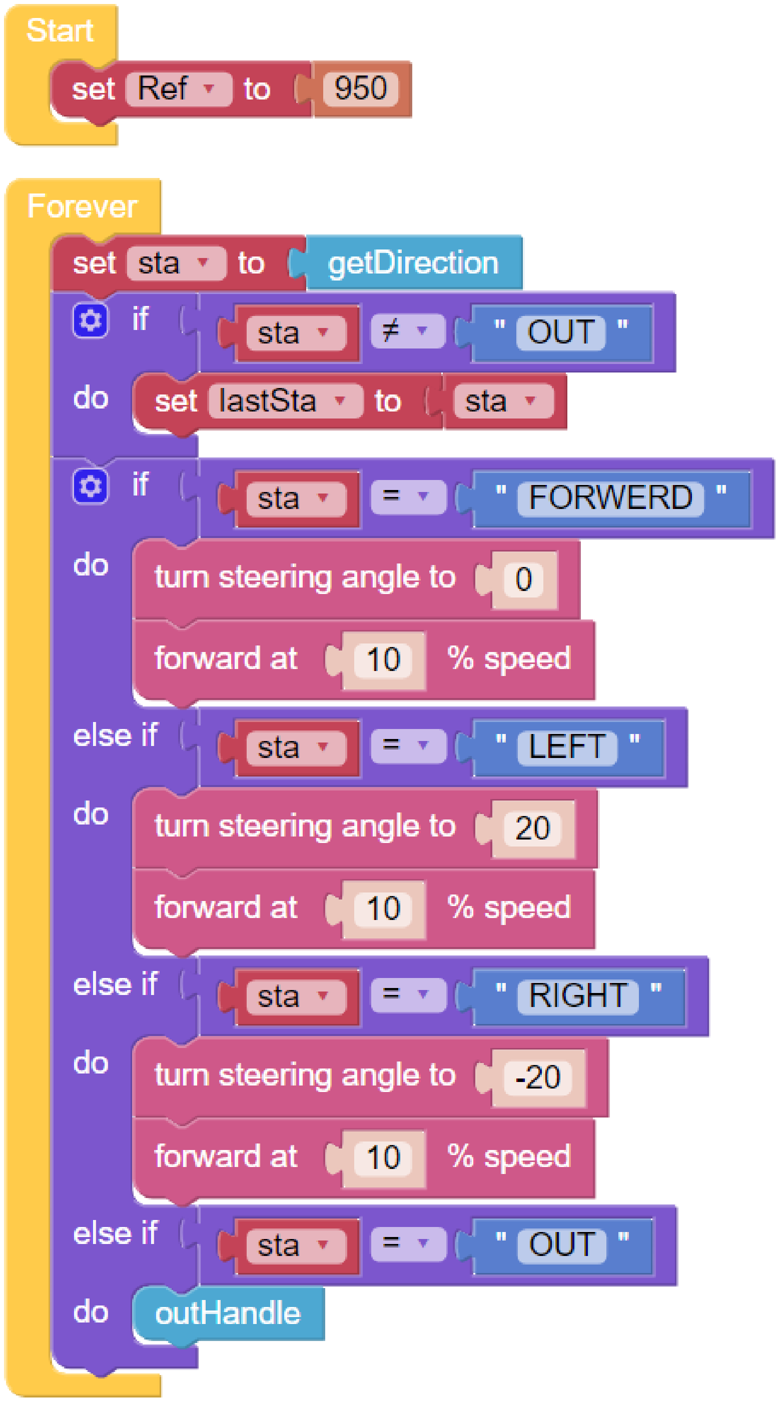


2. **Set ref to ()** block is used to set the grayscale threshold, you need to modify it according to the actual situation. You can go ahead and run *Test Grayscale Module* to see the values of the grayscale module on the white and black surfaces, and fill in their middle values in this block.

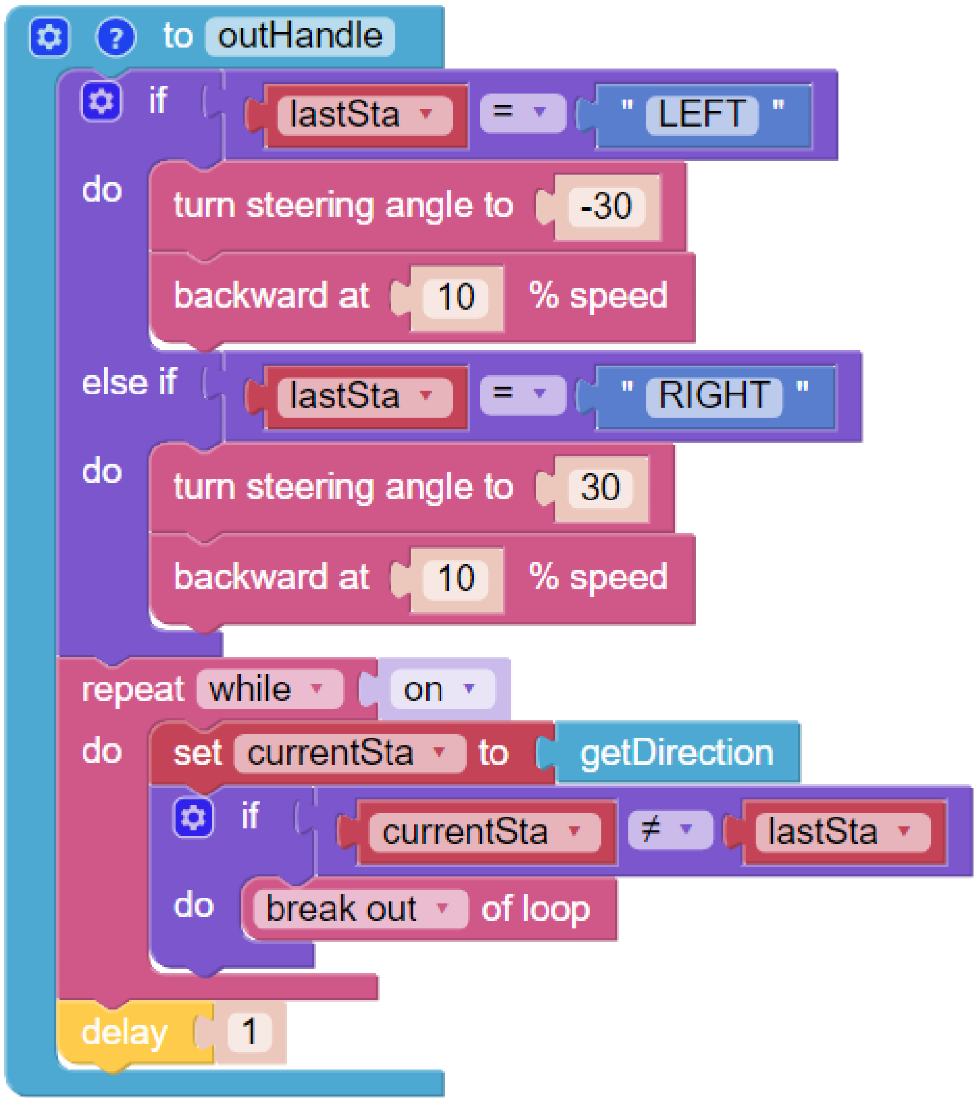
EXAMPLE

Note:

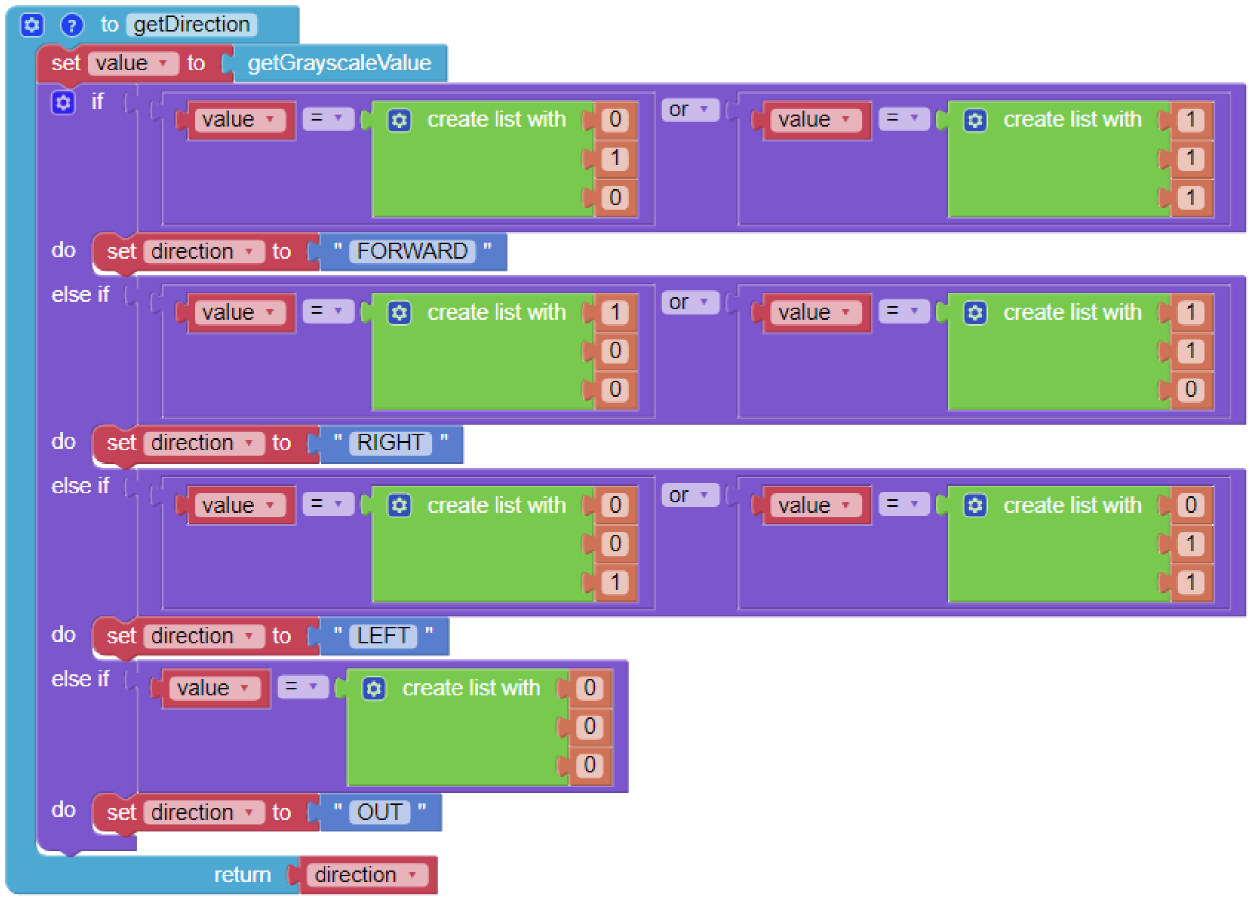
- You can write the program according to the following picture, please refer to the tutorial: How to Create a New Project?.
- Or find the code with the same name on the **Examples** page of the EzBlock Studio and click **Run** or **Edit** directly.



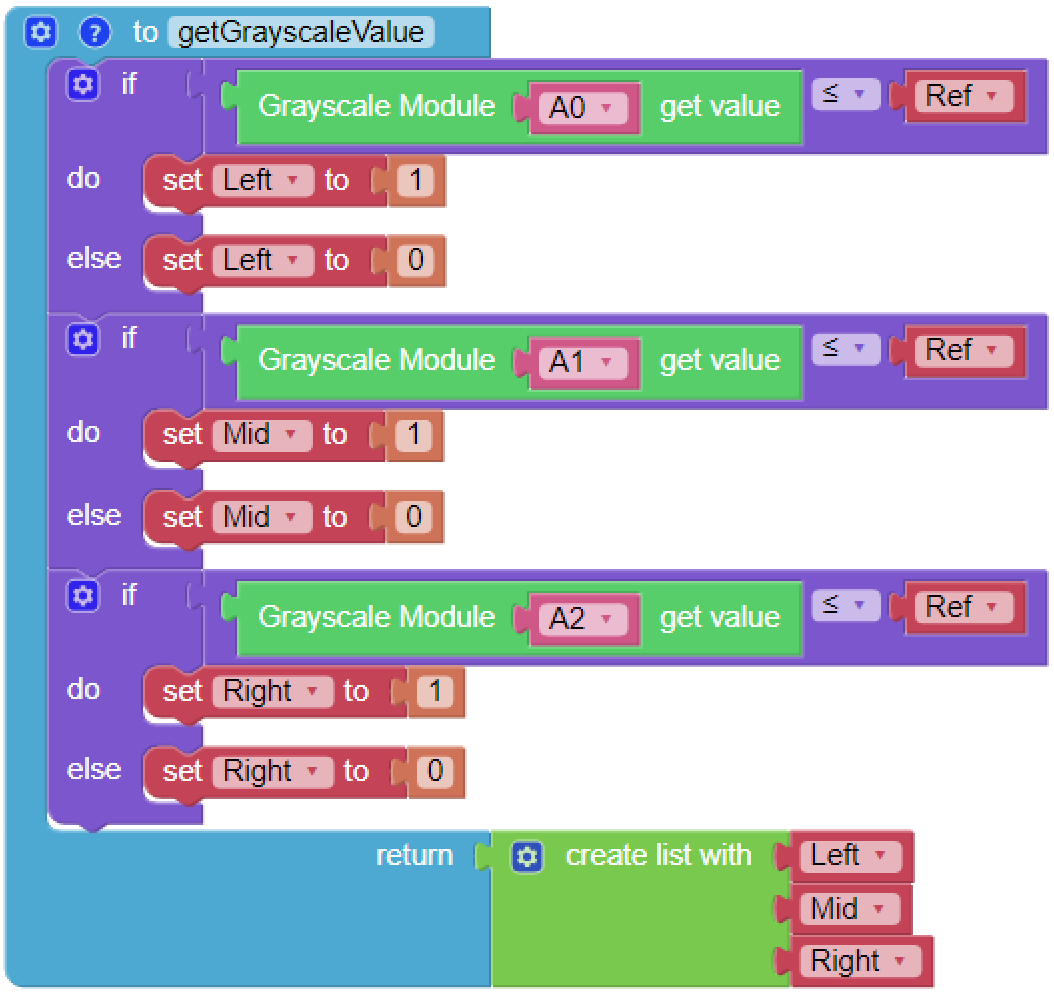
```
Start
    set Ref to 950

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
```



```
to outHandle
    if lastSta = " LEFT "
    do  turn steering angle to -30
        backward at 10 % speed
    else if lastSta = " RIGHT "
    do  turn steering angle to 30
        backward at 10 % speed
    repeat while on
    do  set currentSta to getDirection
        if currentSta ≠ lastSta
        do  break out of loop
    delay 1
```



```
to getDirection
  set value to getGrayscaleValue
  if value = create list with 0 1 0  or  value = create list with 1 1 1
  do set direction to " FORWARD "
  else if value = create list with 1 0 0  or  value = create list with 1 1 0
  do set direction to " RIGHT "
  else if value = create list with 0 0 1  or  value = create list with 0 1 1
  do set direction to " LEFT "
  else if value = create list with 0 0 0
  do set direction to " OUT "
  return direction
```



```
to getGrayscaleValue
    if  Grayscale Module A0 get value ≤ Ref
    do    set Left to 1
    else  set Left to 0
    if  Grayscale Module A1 get value ≤ Ref
    do    set Mid to 1
    else  set Mid to 0
    if  Grayscale Module A2 get value ≤ Ref
    do    set Right to 1
    else  set Right to 0
    return  create list with  Left
                              Mid
                              Right
```

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

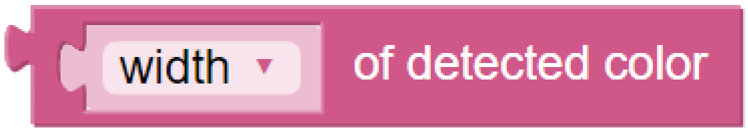
Ready to explore and create with us? Click [] and join today!

5.17 Bullfight

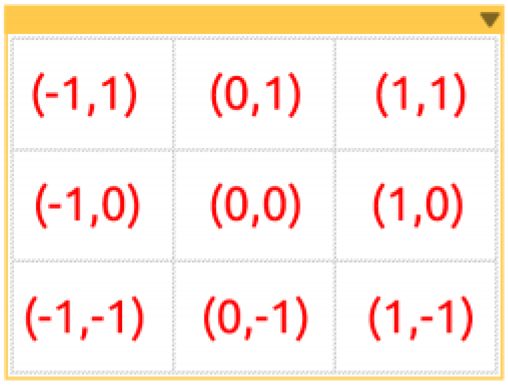
Turn PiCar-X into an angry bull! Prepare a red cloth, such as a handkerchief, and become a Bullfighter. When the PiCar-X chases after the red cloth, be careful not to get hit!

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.

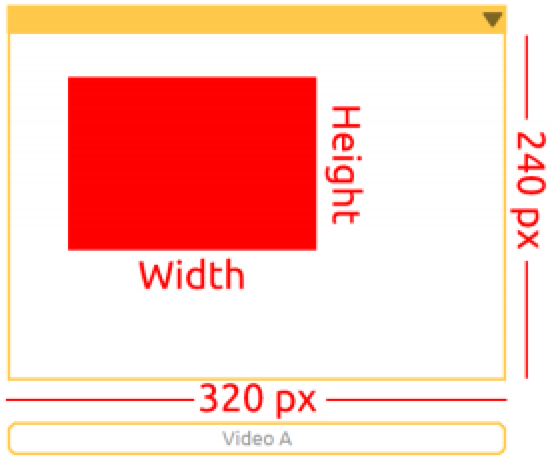
TIPS



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).

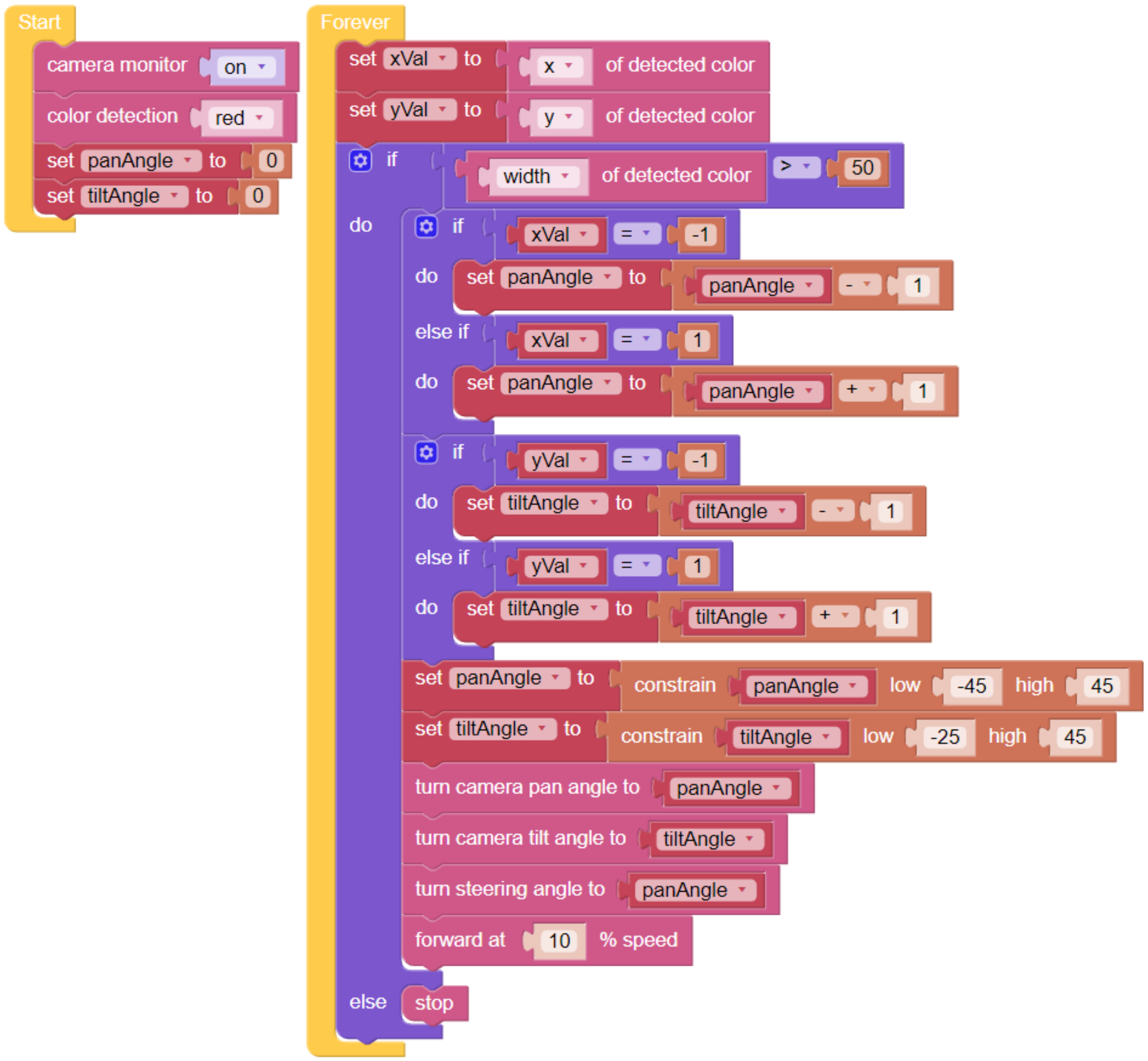


The "object detection" will detect the Width and Height of the graphic. If multiple targets are identified, the dimensions of the largest target will be recorded.

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.



```
Start
  camera monitor on
  color detection red
  set panAngle to 0
  set tiltAngle to 0

Forever
  set xVal to x of detected color
  set yVal to y of detected color
  if width of detected color > 50
  do
    if xVal = -1
    do  set panAngle to panAngle - 1
    else if xVal = 1
    do  set panAngle to panAngle + 1
    if yVal = -1
    do  set tiltAngle to tiltAngle - 1
    else if yVal = 1
    do  set tiltAngle to tiltAngle + 1
    set panAngle to constrain panAngle low -45 high 45
    set tiltAngle to constrain tiltAngle low -25 high 45
    turn camera pan angle to panAngle
    turn camera tilt angle to tiltAngle
    turn steering angle to panAngle
    forward at 10 % speed
  else stop
```

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.

- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

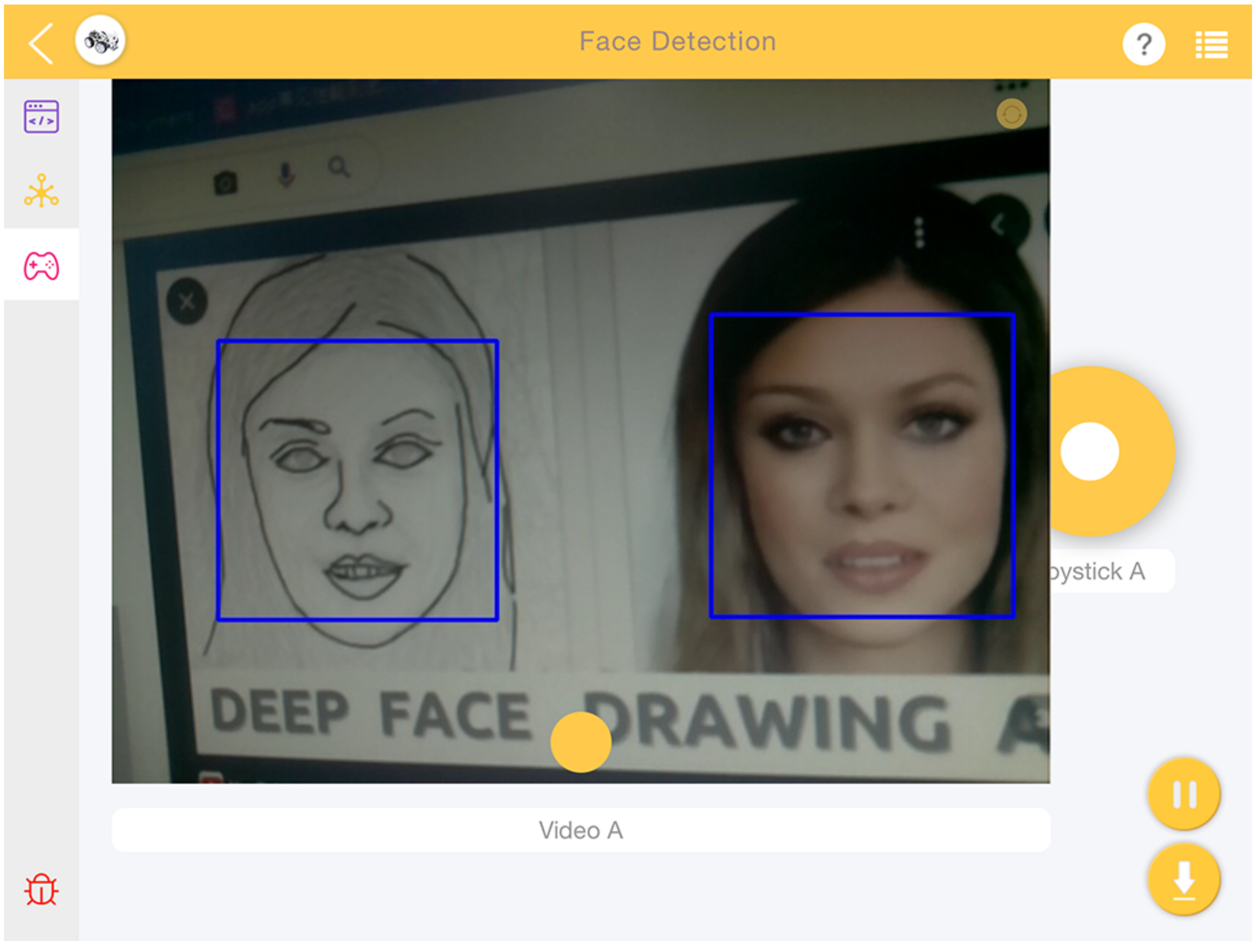
5.18 Beware of Pedestrians

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.

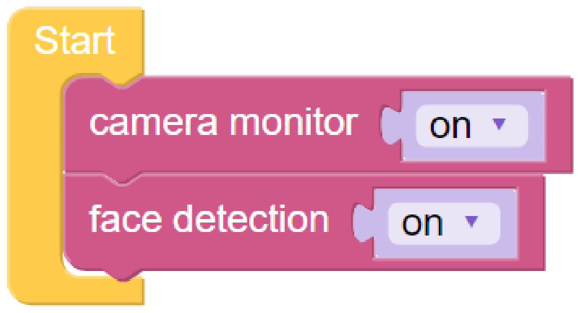
- How to Use the Remote Control Function?



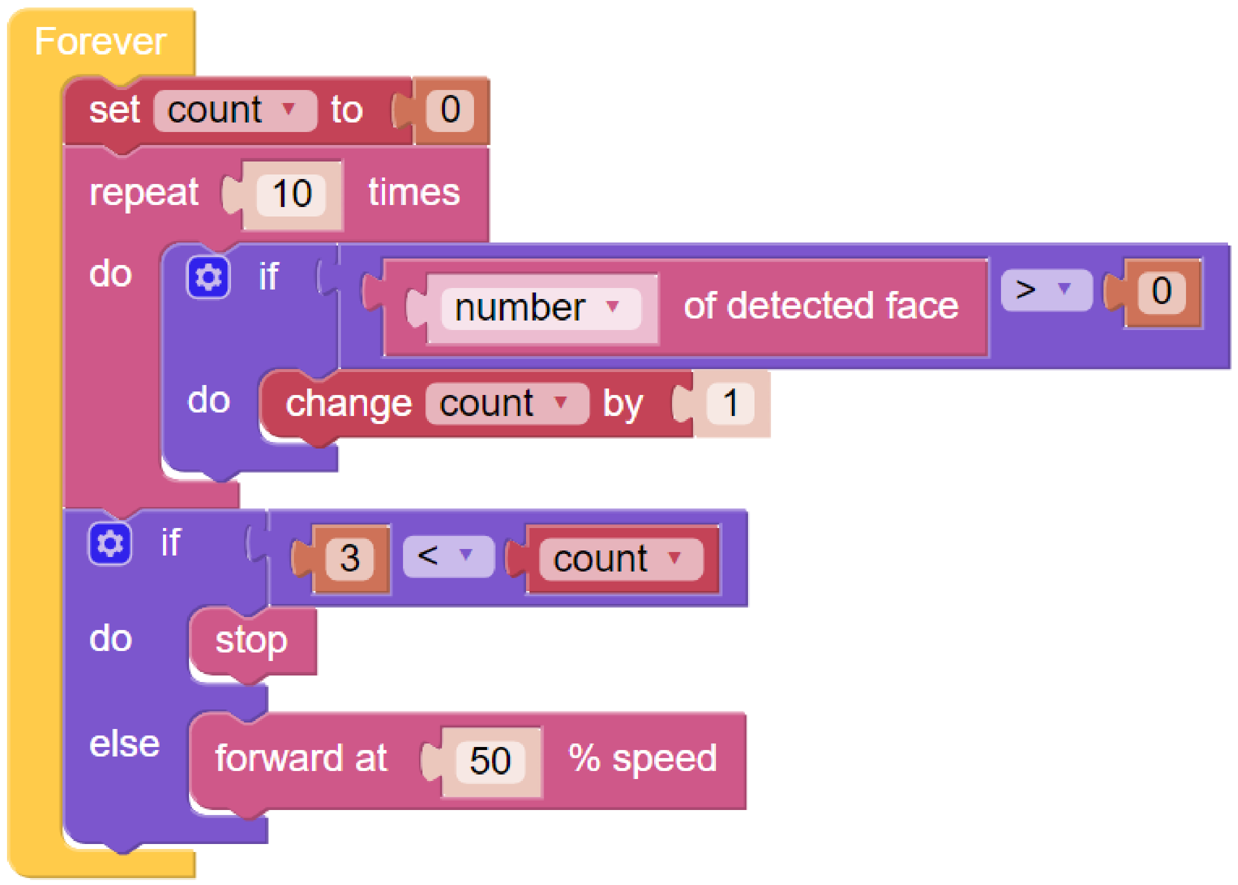
EXAMPLE

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.



```
Start
    camera monitor on
    face detection on
```



```
Forever
    set count to 0
    repeat 10 times
    do
        if number of detected face > 0
        do
            change count by 1
    if 3 < count
    do
        stop
    else
        forward at 50 % speed
```

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

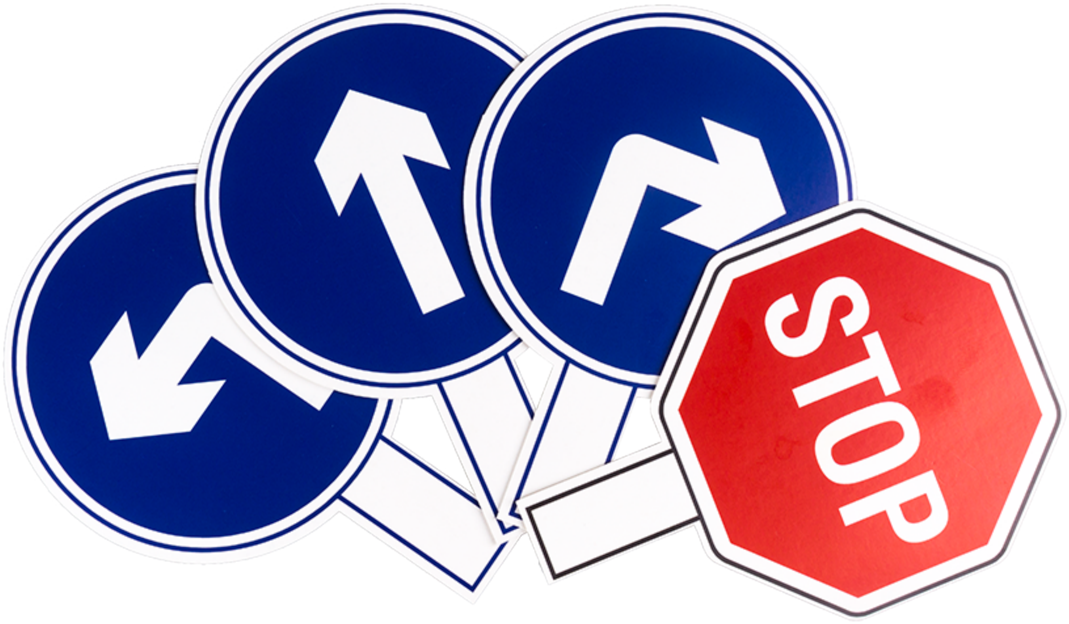
5.19 Traffic Sign Detection

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`

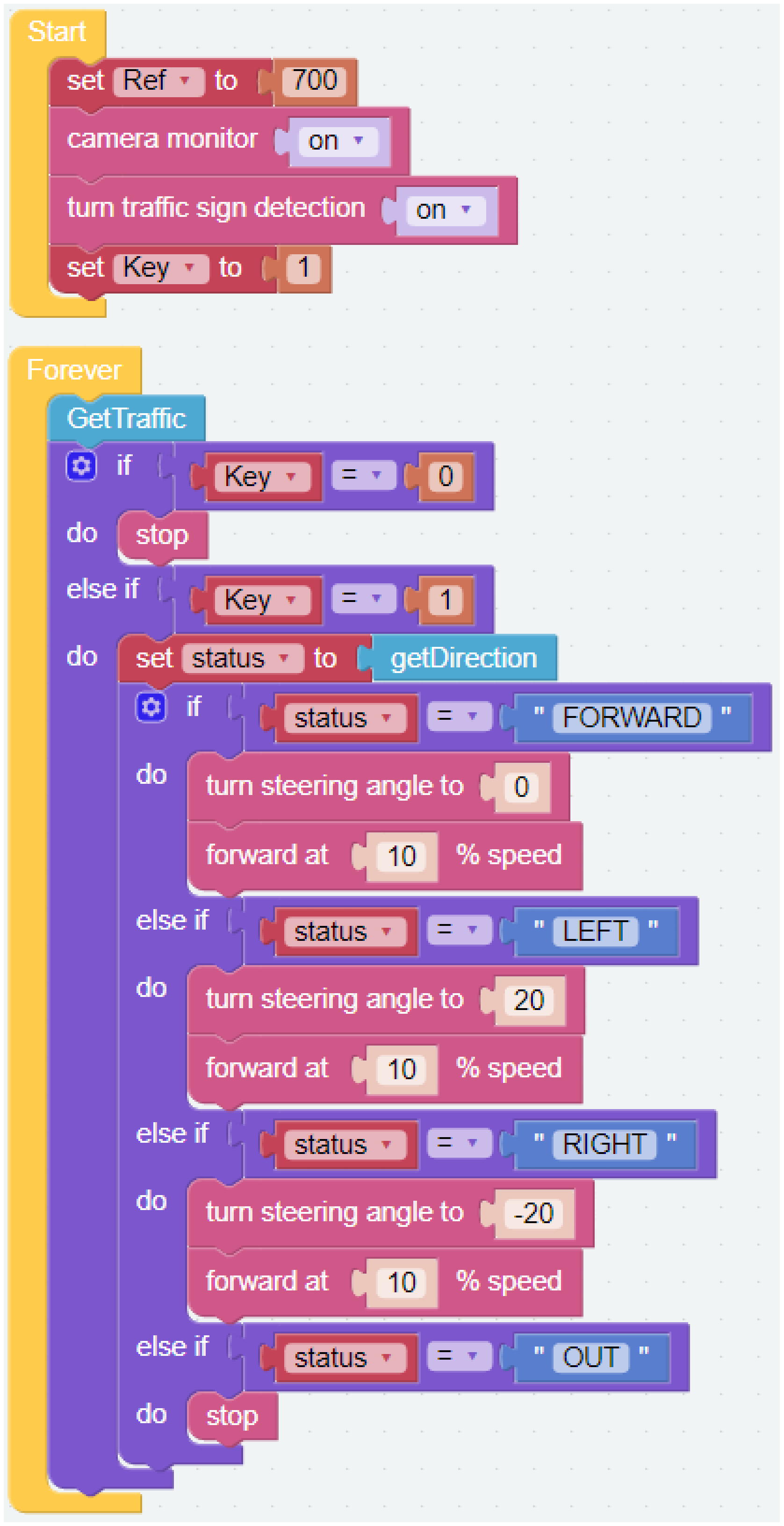
2. **Set ref to ()** block is used to set the grayscale threshold, you need to modify it according to the actual situation. You can go ahead and run *Test Grayscale Module* to see the values of the grayscale module on the white and black surfaces, and fill in their middle values in this block.

EXAMPLE

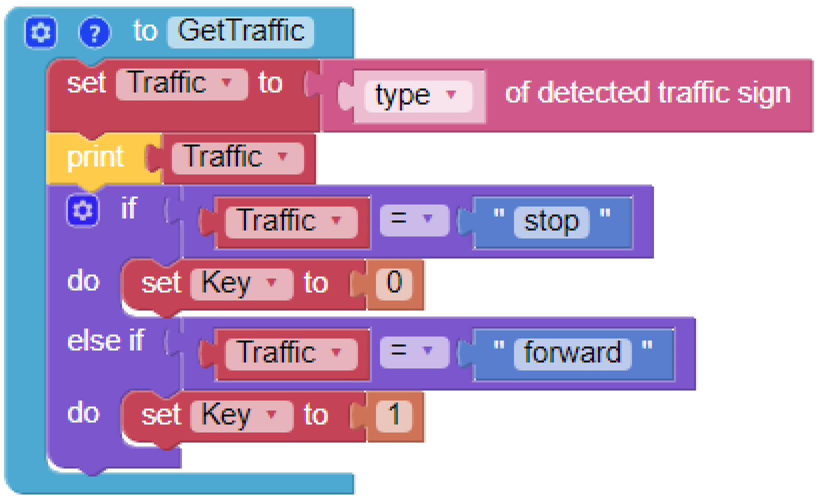
Note:

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

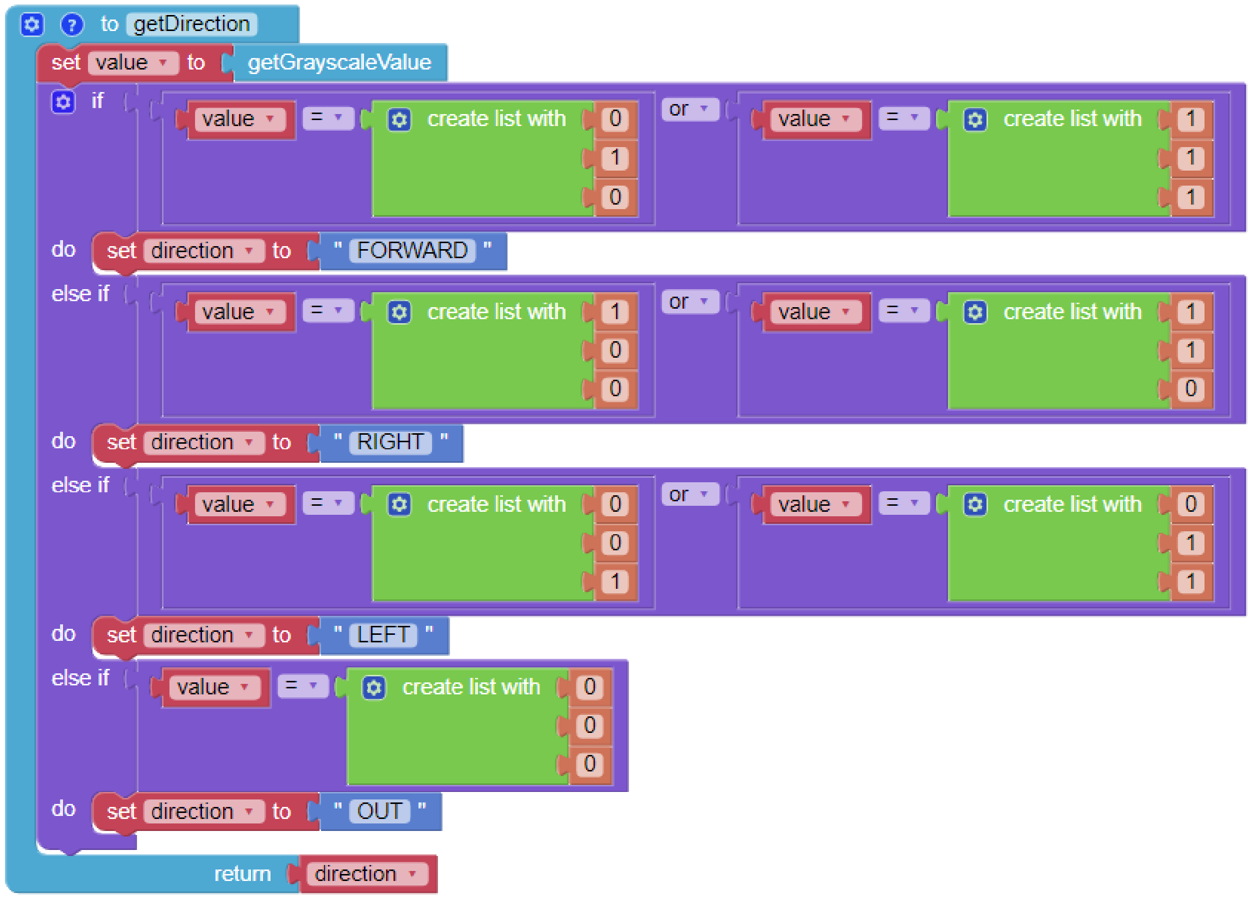
- Or find the code with the same name on the **Examples** page of the EzBlock Studio and click **Run** or **Edit** directly.



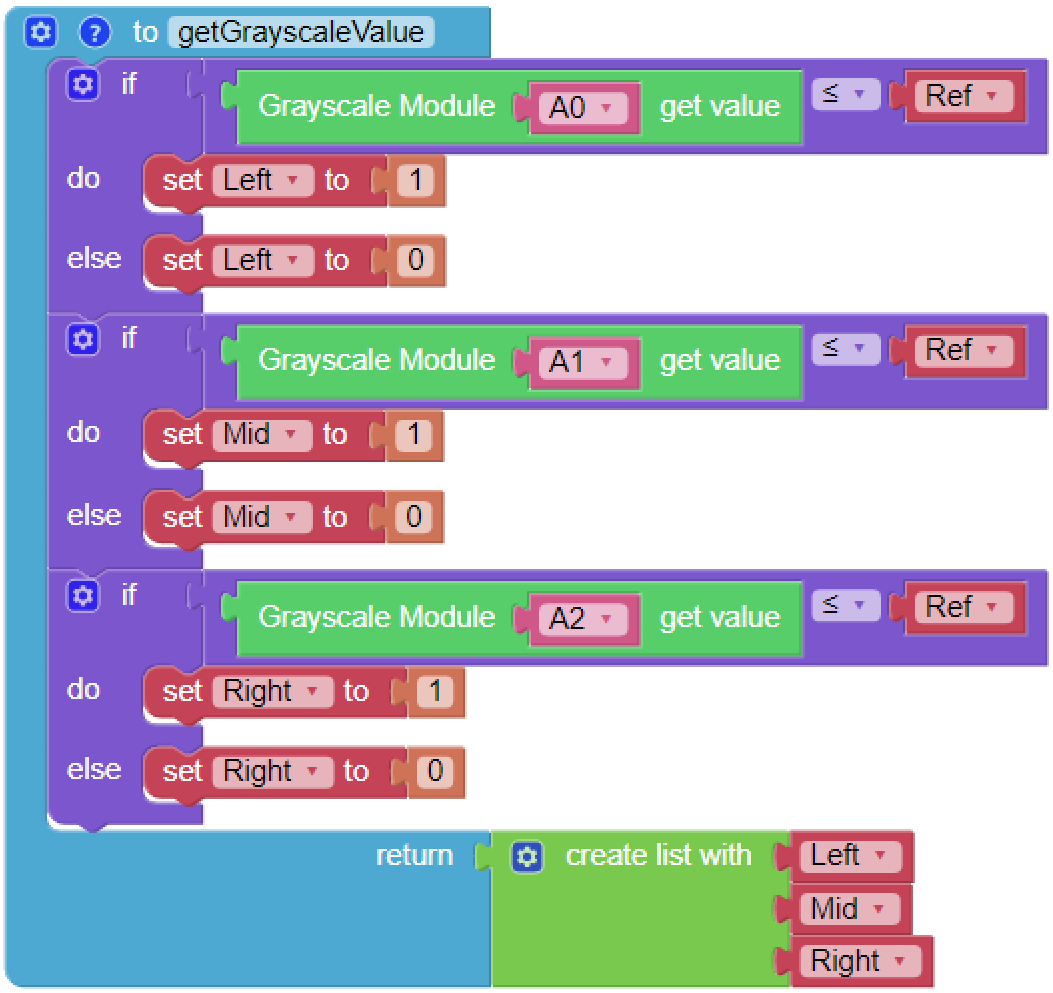
```
Start
    set Ref to 700
    camera monitor on
    turn traffic sign detection on
    set Key to 1

Forever
    GetTraffic
    if Key = 0
    do stop
    else if Key = 1
    do set status to getDirection
        if status = " FORWARD "
        do turn steering angle to 0
           forward at 10 % speed
        else if status = " LEFT "
        do turn steering angle to 20
           forward at 10 % speed
        else if status = " RIGHT "
        do turn steering angle to -20
           forward at 10 % speed
        else if status = " OUT "
        do stop
```



```
to GetTraffic
    set Traffic to type of detected traffic sign
    print Traffic
    if Traffic = " stop "
    do  set Key to 0
    else if Traffic = " forward "
    do  set Key to 1
```



```
to getDirection
    set value to getGrayscaleValue
    if value = create list with 0 1 0 or value = create list with 1 1 1
    do  set direction to " FORWARD "
    else if value = create list with 1 0 0 or value = create list with 1 1 0
    do  set direction to " RIGHT "
    else if value = create list with 0 0 1 or value = create list with 0 1 1
    do  set direction to " LEFT "
    else if value = create list with 0 0 0
    do  set direction to " OUT "
    return direction
```



```
to getGrayscaleValue
    if   Grayscale Module A0 get value ≤ Ref
    do   set Left to 1
    else set Left to 0
    if   Grayscale Module A1 get value ≤ Ref
    do   set Mid to 1
    else set Mid to 0
    if   Grayscale Module A2 get value ≤ Ref
    do   set Right to 1
    else set Right to 0
    return create list with Left
                            Mid
                            Right
```

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

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.

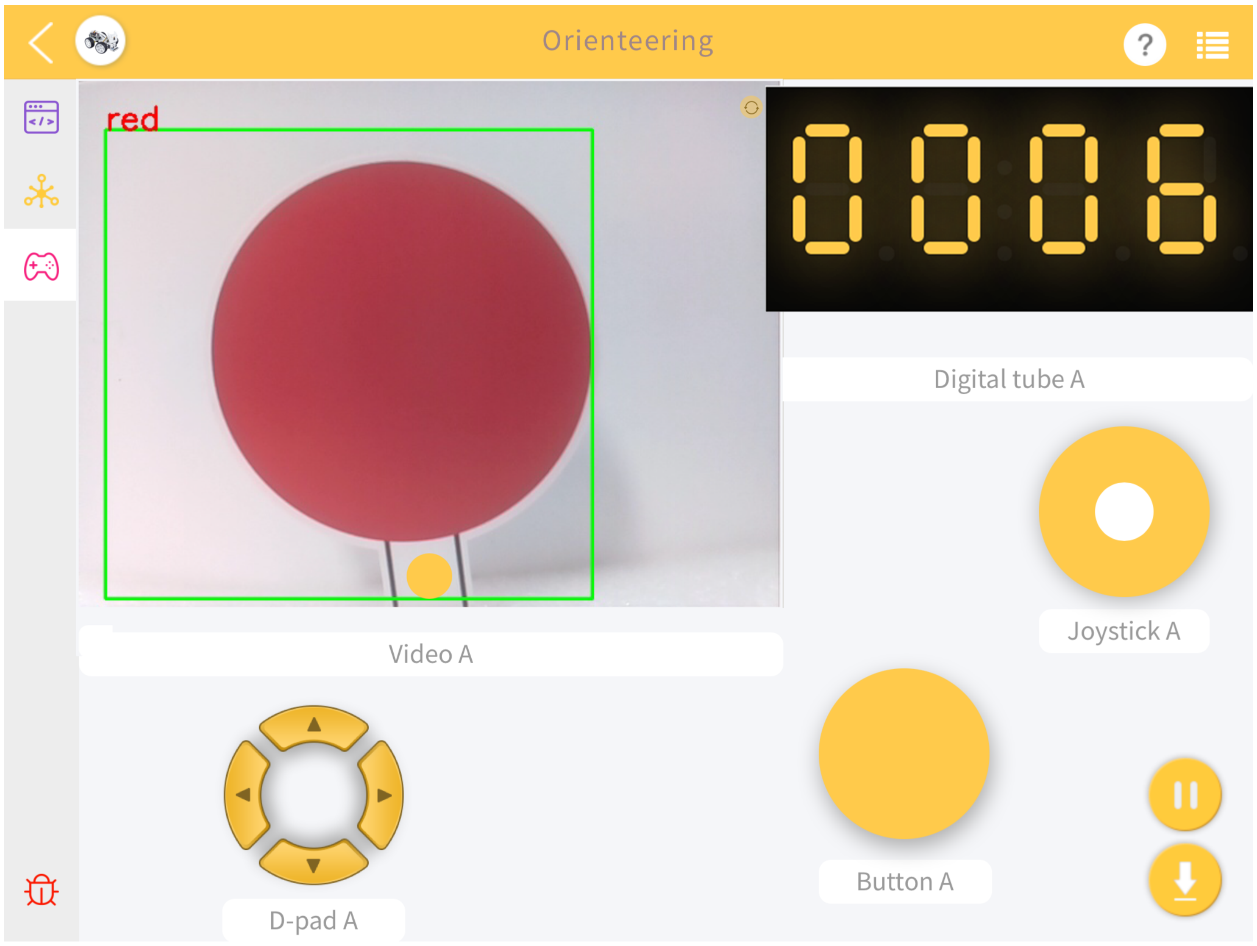
- `[PDF]Color Cards`

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.

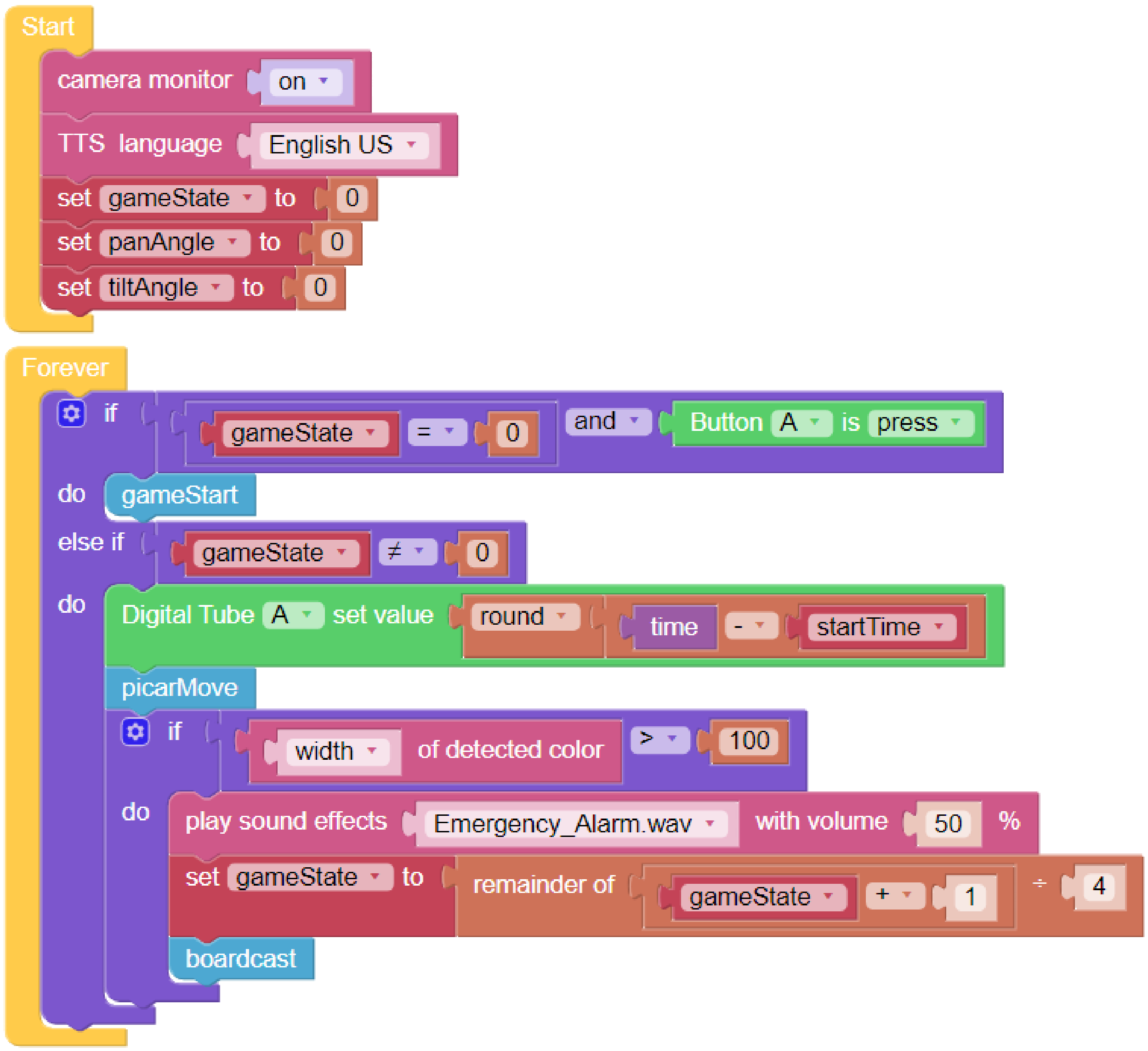


Take turns playing this game with friends to see who can help PiCar-X complete the objective the fastest!

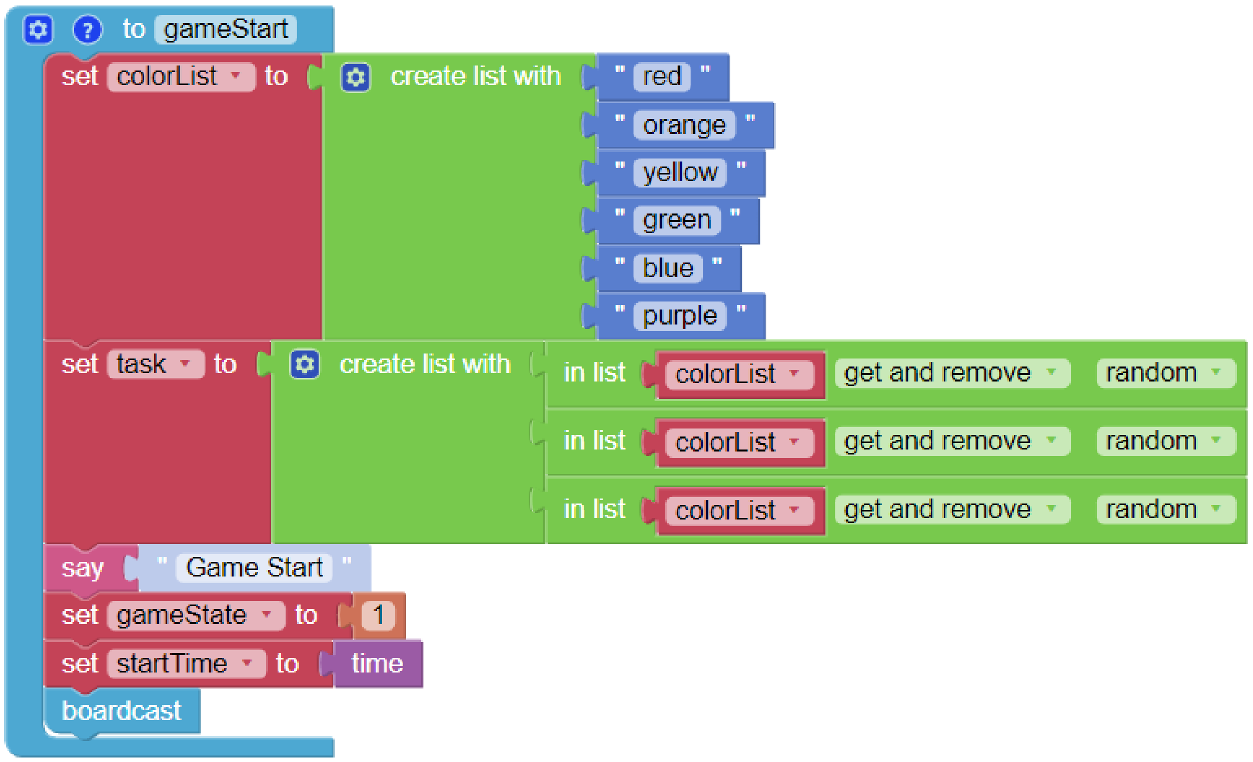
EXAMPLE

Note:

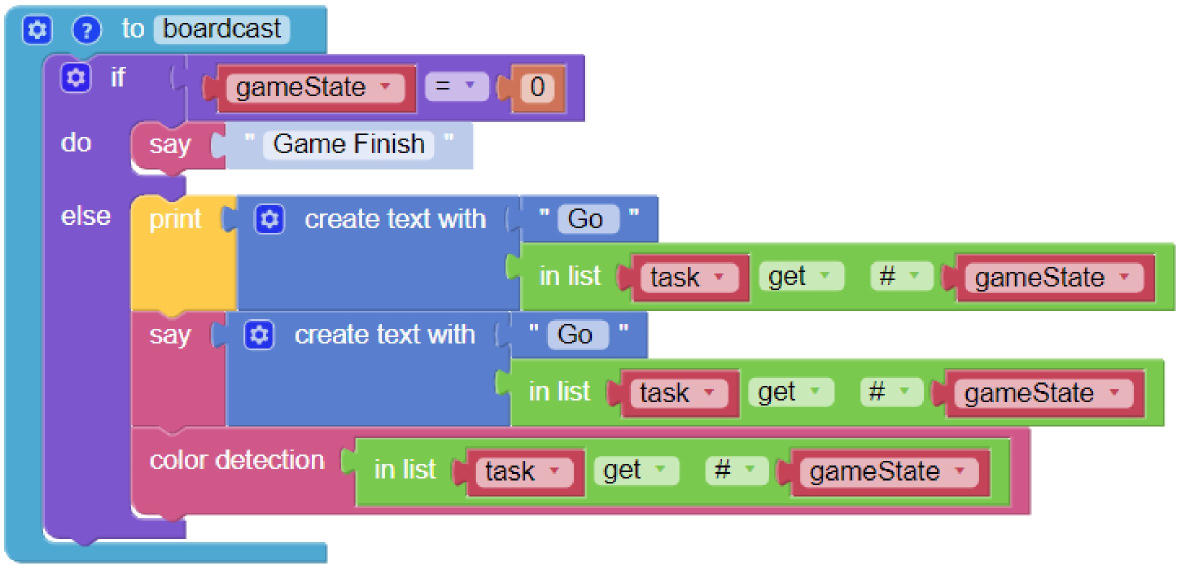
- You can write the program according to the following picture, please refer to the tutorial: How to Create a New Project?.
- Or find the code with the same name on the **Examples** page of the EzBlock Studio and click **Run** or **Edit** directly.



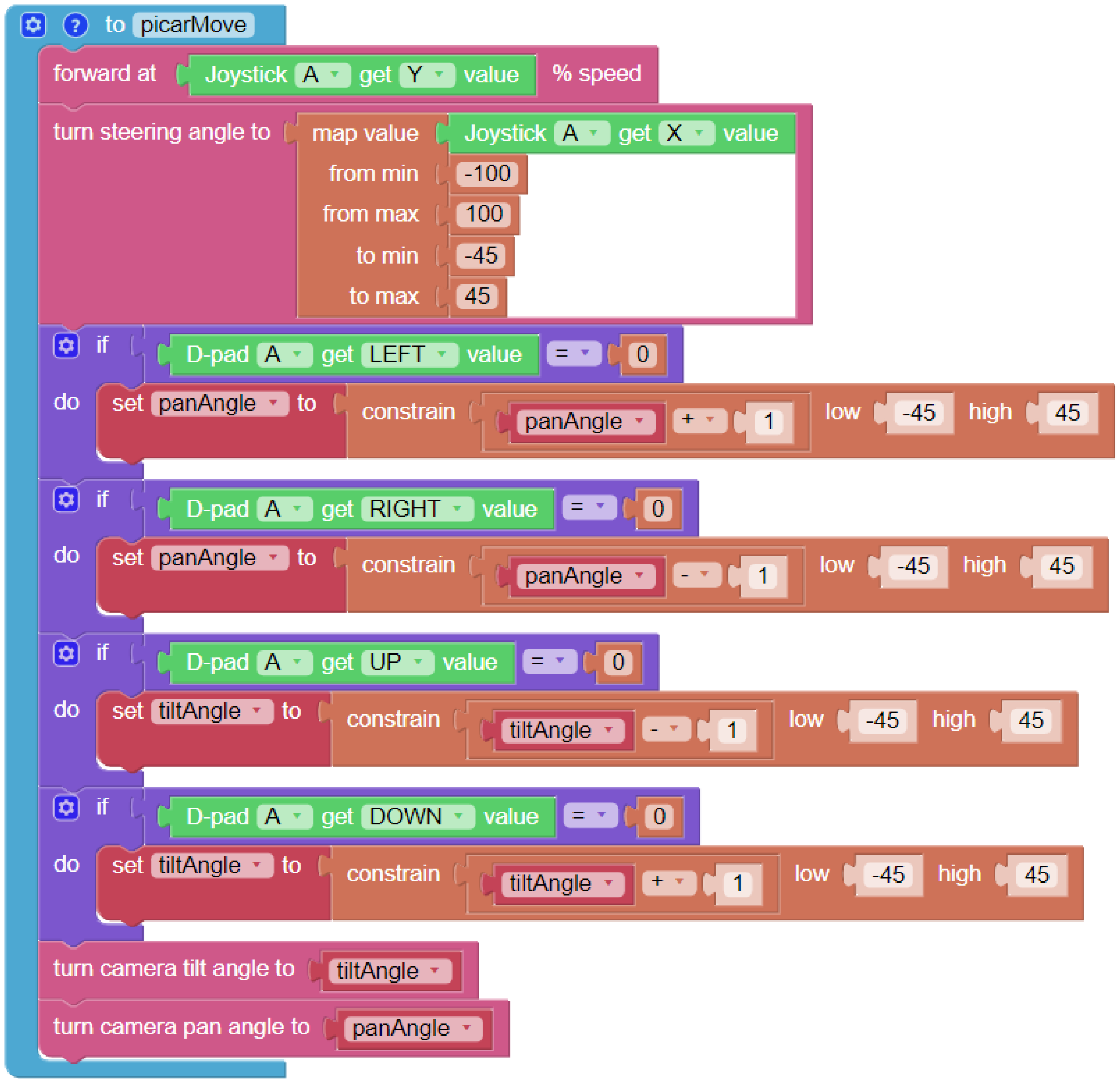
```
Start
  camera monitor on
  TTS language English US
  set gameState to 0
  set panAngle to 0
  set tiltAngle to 0

Forever
  if gameState = 0 and Button A is press
  do gameStart
  else if gameState ≠ 0
  do Digital Tube A set value round time - startTime
     picarMove
     if width of detected color > 100
     do play sound effects Emergency_Alarm.wav with volume 50 %
        set gameState to remainder of gameState + 1 ÷ 4
        boardcast
```



```
to gameStart
    set colorList to create list with "red" "orange" "yellow" "green" "blue" "purple"
    set task to create list with
        in list colorList get and remove random
        in list colorList get and remove random
        in list colorList get and remove random
    say "Game Start"
    set gameState to 1
    set startTime to time
    boardcast
```



```
to boardcast
    if gameState = 0
    do  say "Game Finish"
    else
        print create text with "Go"
                               in list task get # gameState
        say create text with "Go"
                             in list task get # gameState
        color detection in list task get # gameState
```



```
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
```

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

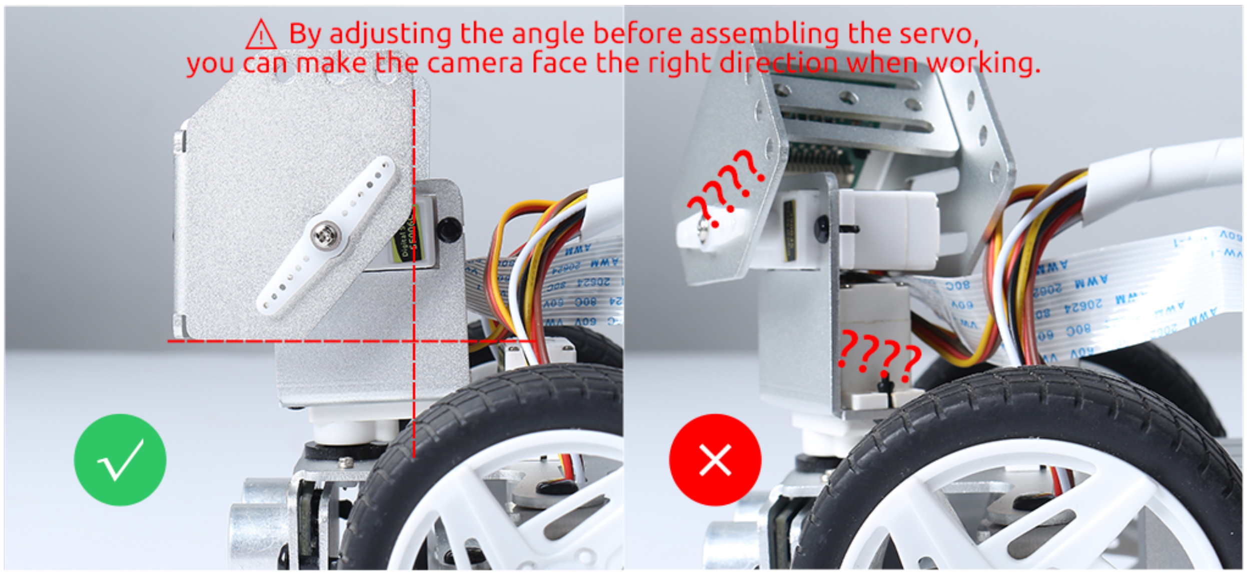
Ready to explore and create with us? Click [] and join today!

CHAPTER

SIX

ADJUST SERVO FOR ASSEMBLY

Before assembling the servo, it is essential to set the angle to zero. Since servo motors have a limited range of motion, setting the angle to zero degrees ensures that the servo starts in its initial position and avoids exceeding its range when powered on. Failing to set the servo to zero beforehand may cause it to attempt to move beyond its allowed range when powered, potentially damaging both the servo and the mechanical system it's attached to. This step is crucial to guarantee safe and proper operation of the servo motor.


⚠ By adjusting the angle before assembling the servo,
you can make the camera face the right direction when working.
????
????

6.1 For Python Users

Refer to *1. Quick Guide on Python* to complete the installation of Raspberry Pi OS and adjust the servo angles.

6.2 For Ezblock Users

Note: If you are using a Raspberry Pi 5, our graphical programming software, EzBlock, is not supported.

Once you have installed the Ezblock system, the P11 pin can be used to adjust the servo. For more details, please refer to *Quick Guide on EzBlock*.

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

CHAPTER SEVEN

APPENDIX

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

7.1 Filezilla Software

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

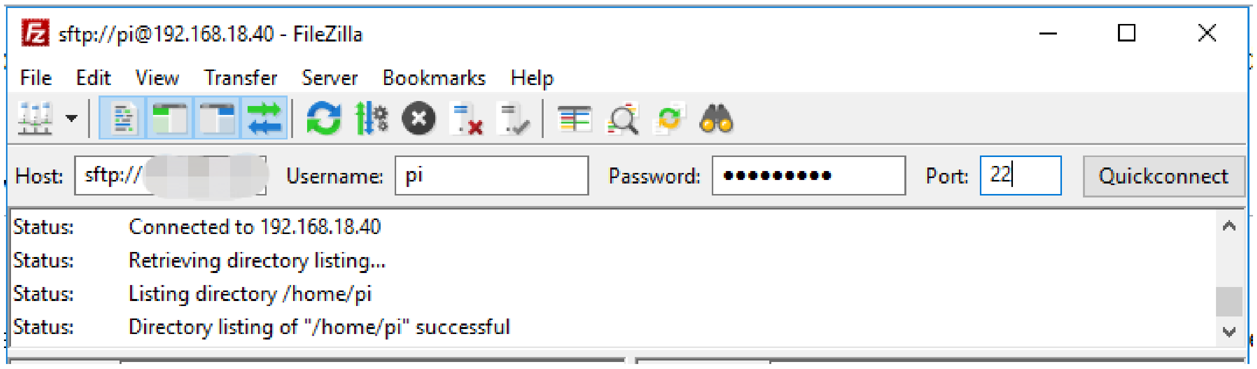
Filezilla is an open source software that not only supports FTP, but also FTP over TLS (FTPS) and SFTP. We can use Filezilla to upload local files (such as pictures and audio, etc.) to the Raspberry Pi, or download files from the Raspberry Pi to the local.

Step 1: Download Filezilla.

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

Step 2: Connect to Raspberry Pi

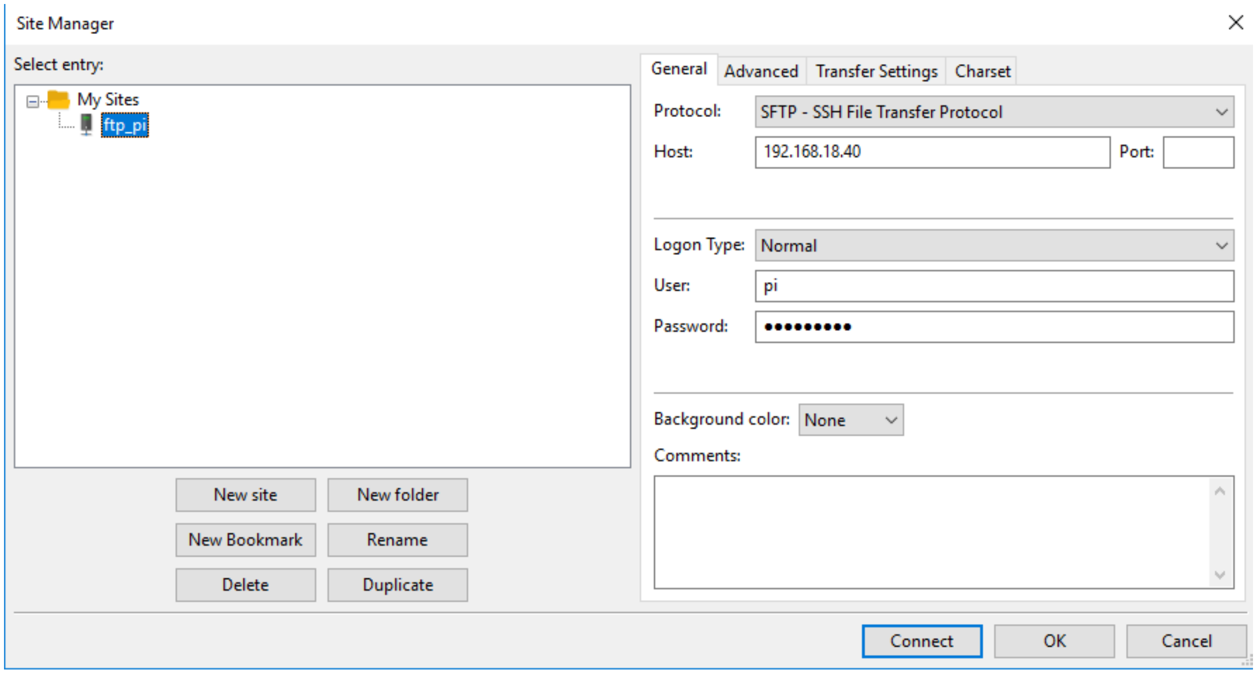
After a quick install open it up and now connect it to an FTP server. It has 3 ways to connect, here we use the **Quick Connect** bar. Enter the **hostname/IP**, **username**, **password** and **port (22)**, then click **Quick Connect** or press **Enter** to connect to the server.


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

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

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

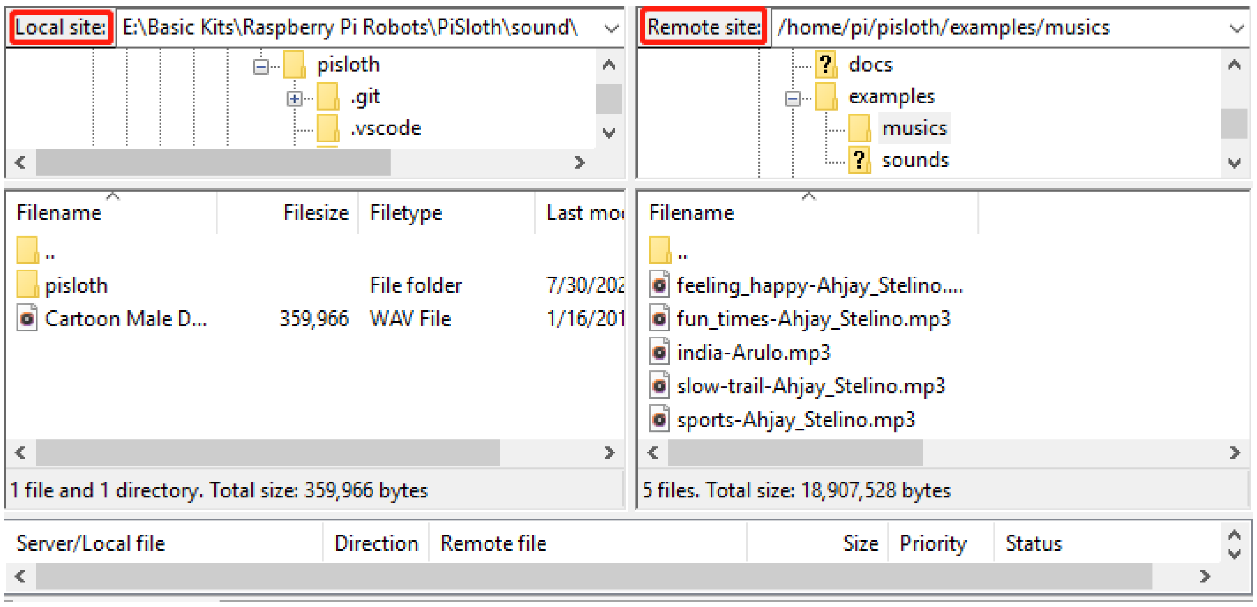
Note: Quick Connect is a good way to test your login information. If you want to create a permanent entry, you can select **File-> Copy Current Connection to Site Manager** after a successful Quick Connect, enter the name and click **OK**. Next time you will be able to connect by selecting the previously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



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

Step 3: Upload/download files.

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


Local site: E:\Basic Kits\Raspberry Pi Robots\PiSloth\sound\
pisloth
.git
.vscode
Remote site: /home/pi/pisloth/examples/musics
docs
examples
musics
sounds

Filename	Filesize	Filetype	Last mo
..			
pisloth		File folder	7/30/202
Cartoon Male D...	359,966	WAV File	1/16/201

1 file and 1 directory. Total size: 359,966 bytes

Filename
..
feeling_happy-Ahjay_Stelino....
fun_times-Ahjay_Stelino.mp3
india-Arulo.mp3
slow-trail-Ahjay_Stelino.mp3
sports-Ahjay_Stelino.mp3

5 files. Total size: 18,907,528 bytes

Server/Local file	Direction	Remote file	Size	Priority	Status

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

7.2 PuTTY

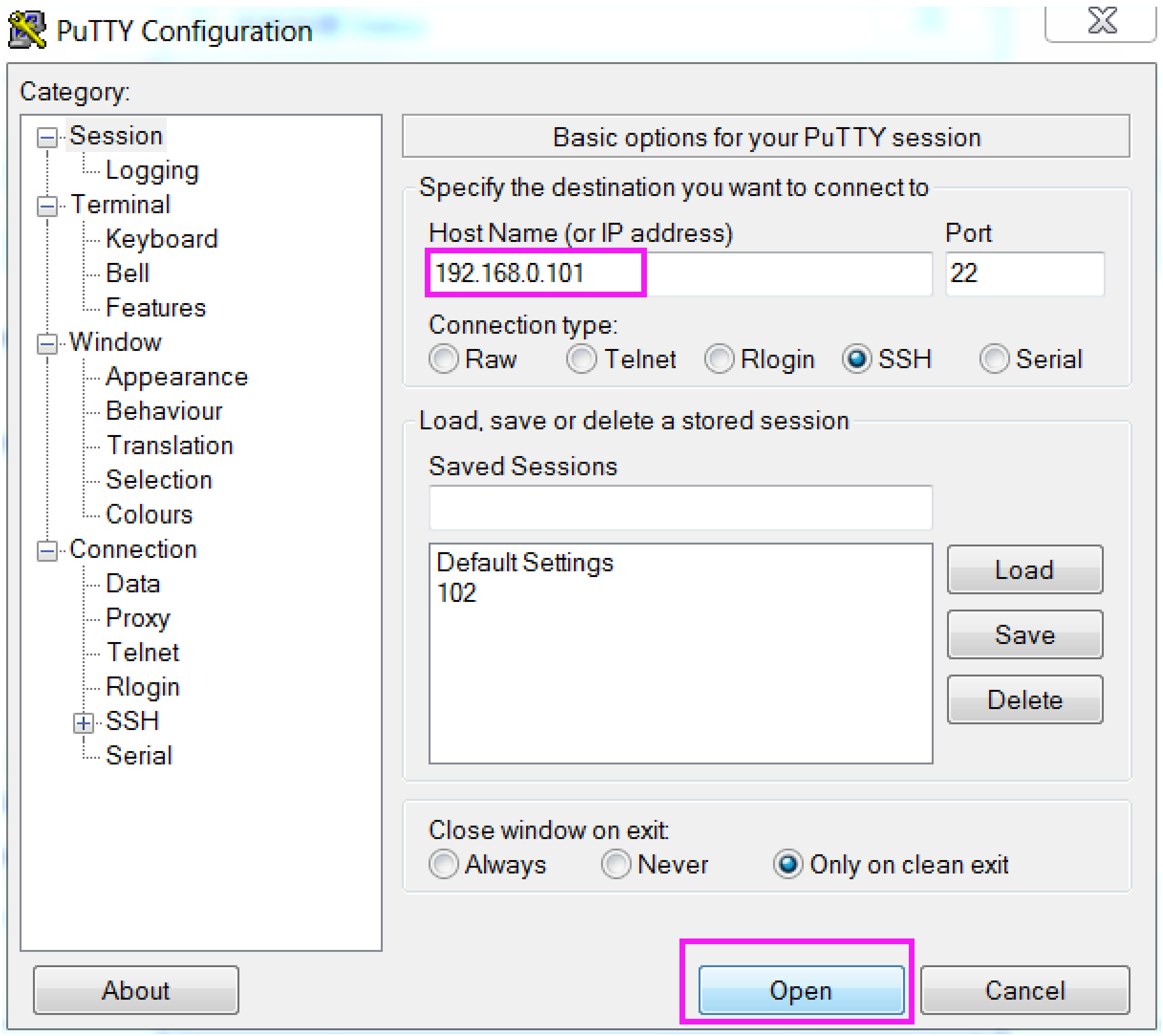
If you are a Windows user, you can use some applications of SSH. Here, we recommend PuTTY.

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).



Host Name (or IP address): 192.168.0.101
Port: 22
Connection type: Raw Telnet Rlogin [✓ SSH] Serial
Saved Sessions:
Default Settings
102
Close window on exit: Always Never [✓ Only on clean exit]

Step 3

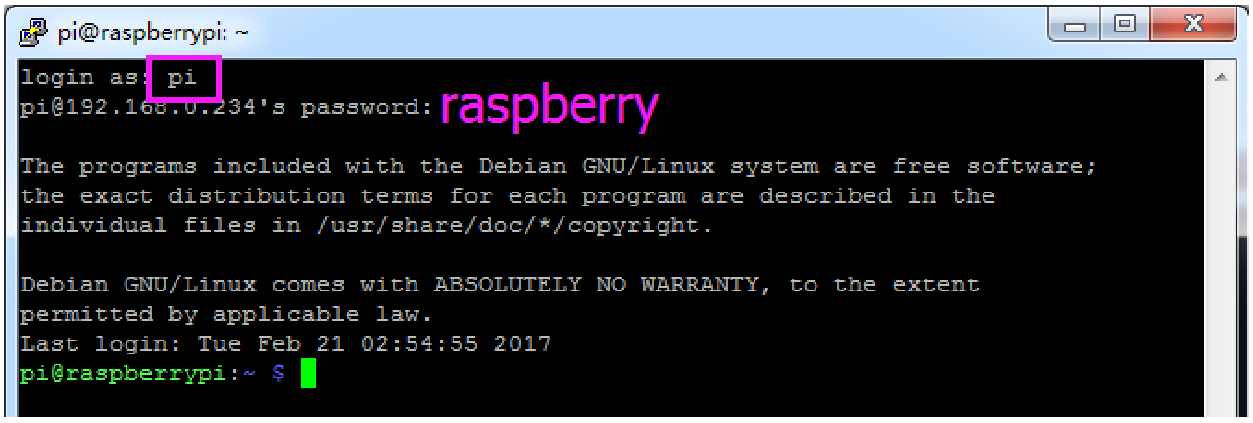
Click **Open**. Note that when you first log in to the Raspberry Pi with the IP address, there prompts a security reminder. Just click **Yes**.

Step 4

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

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

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



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

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

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

Step 5

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

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

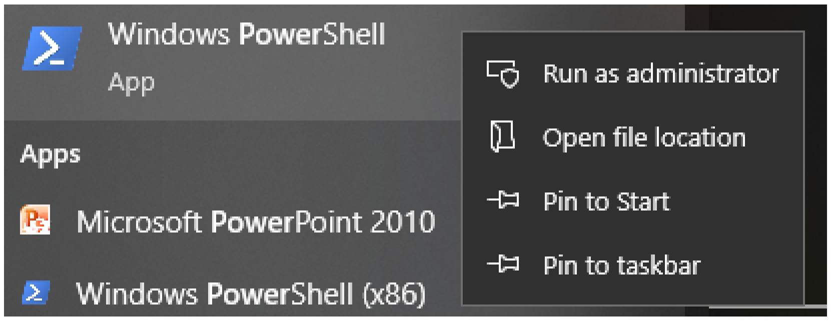
7.3 Install OpenSSH via Powershell

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.

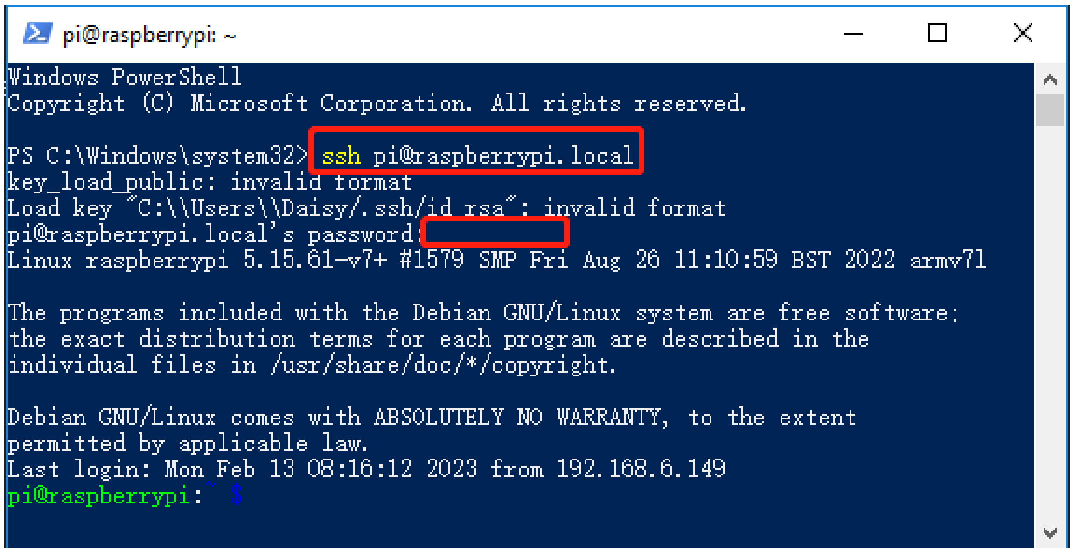
```
Get-WindowsCapability -Online | Where-Object Name -like 'OpenSSH*'
```

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.



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

PS C:\Windows\system32> ssh pi@raspberrypi.local
key_load_public: invalid format
Load key "C:\\Users\\Daisy/.ssh/id_rsa": invalid format
pi@raspberrypi.local's password:
Linux raspberrypi 5.15.61-v7+ #1579 SMP Fri Aug 26 11:10:59 BST 2022 armv7l

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

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Feb 13 08:16:12 2023 from 192.168.6.149
pi@raspberrypi:~ $
```

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

7.4 Remote Desktop Access for Raspberry Pi

For those preferring a graphical user interface (GUI) over command-line access, the Raspberry Pi supports remote desktop functionality. This guide will walk you through setting up and using VNC (Virtual Network Computing) for remote access.

We recommend using VNC® Viewer for this purpose.

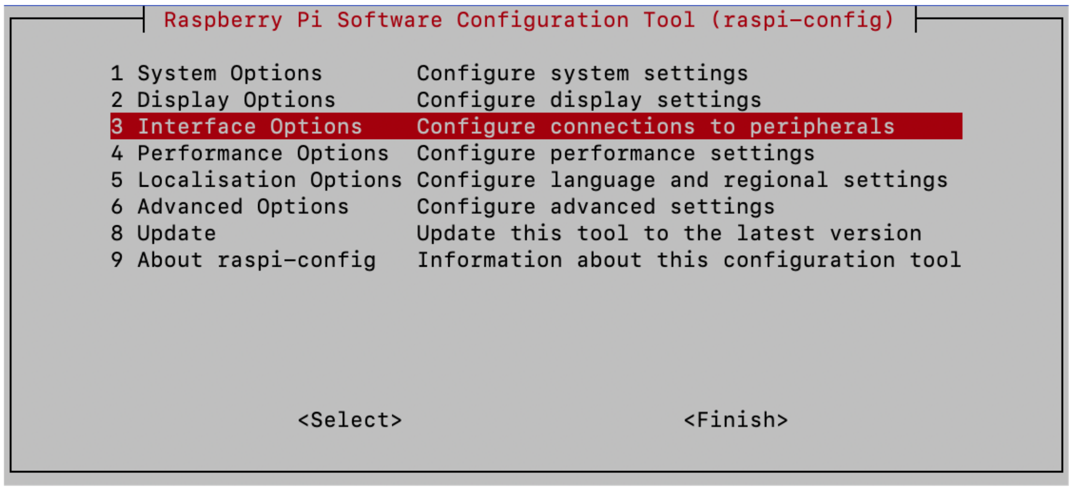
Enabling VNC Service on Raspberry Pi

VNC service comes pre-installed in the Raspberry Pi OS but is disabled by default. Follow these steps to enable it:

1. Enter the following command in the Raspberry Pi terminal:

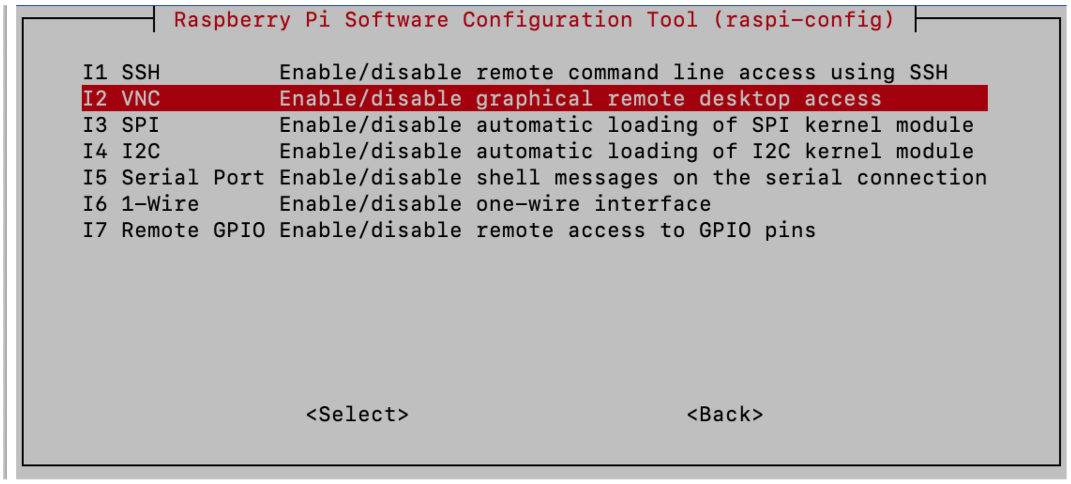
```
sudo raspi-config
```

2. Navigate to **Interfacing Options** using the down arrow key, then press **Enter**.



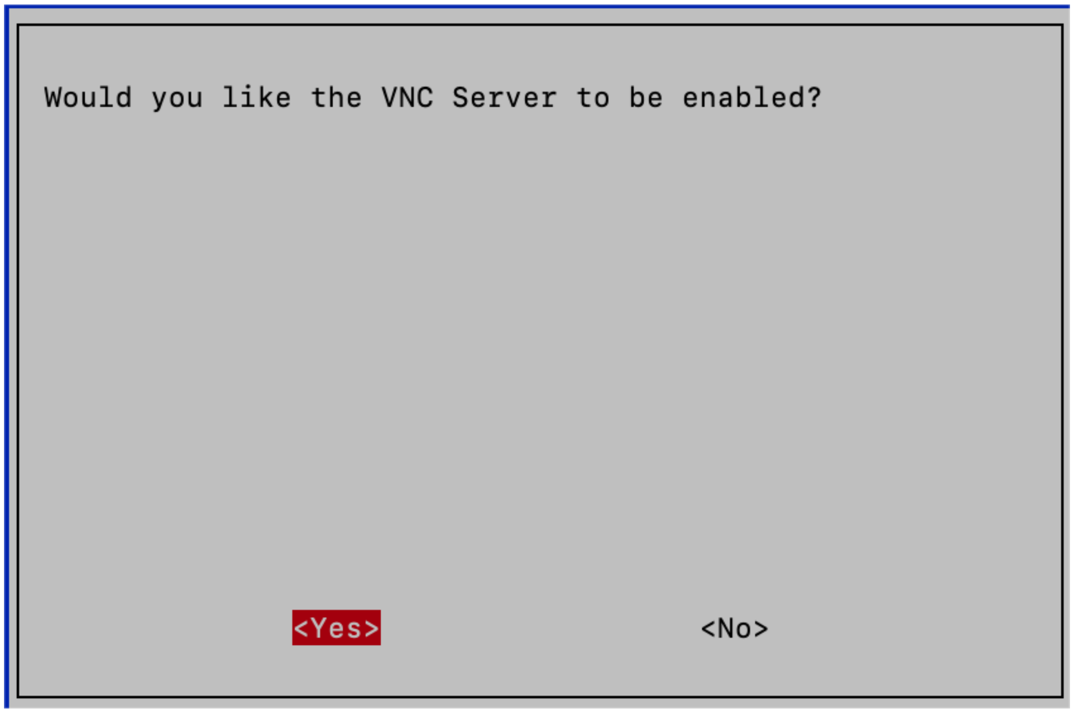
```
Raspberry Pi Software Configuration Tool (raspi-config)

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. Select **VNC** from the options.



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

I1 SSH         Enable/disable remote command line access using SSH
I2 VNC         Enable/disable graphical remote desktop access
I3 SPI         Enable/disable automatic loading of SPI kernel module
I4 I2C         Enable/disable automatic loading of I2C kernel module
I5 Serial Port Enable/disable shell messages on the serial connection
I6 1-Wire      Enable/disable one-wire interface
I7 Remote GPIO Enable/disable remote access to GPIO pins

              <Select>                      <Back>
```

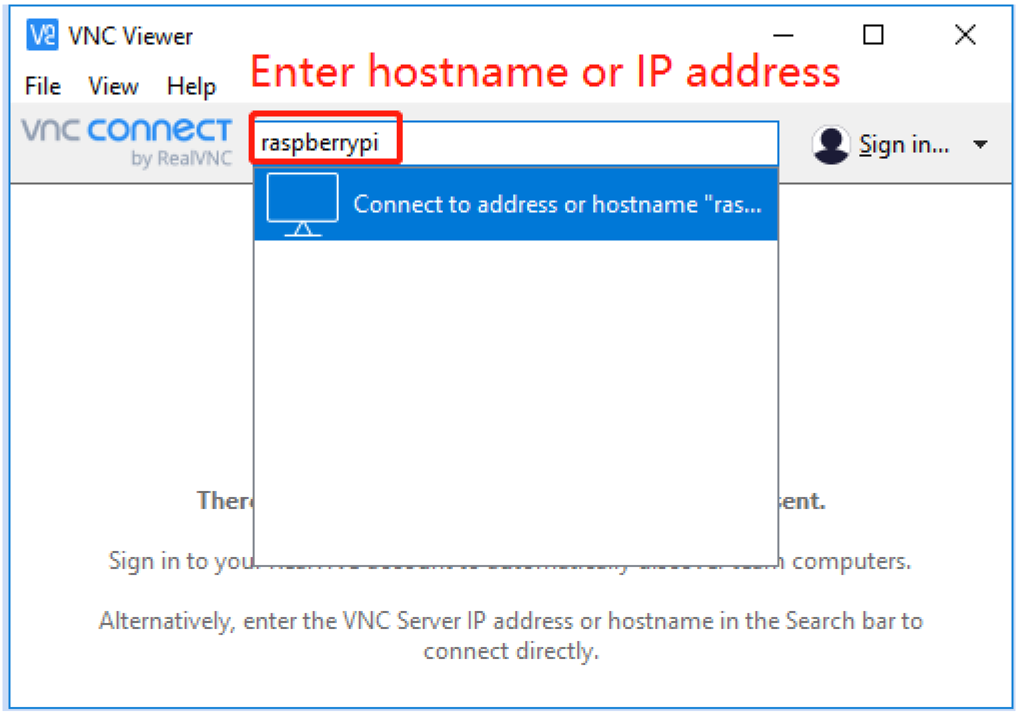
4. Use the arrow keys to choose **<Yes>** -> **<OK>** -> **<Finish>** and finalize the VNC service activation.



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

              <Yes>                    <No>
```

Logging in via VNC Viewer

1. Download and install VNC Viewer on your personal computer.
2. Once installed, launch VNC Viewer. Enter the hostname or IP address of your Raspberry Pi and press Enter.

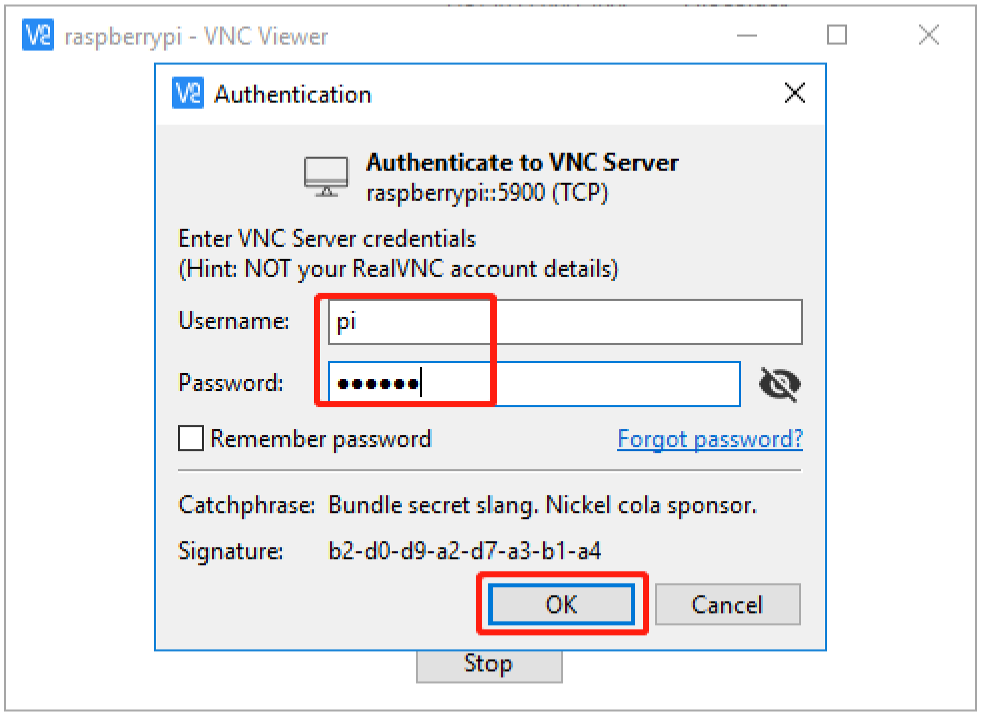


There... ...ent.

Sign in to you... ...computers.

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

3. When prompted, enter your Raspberry Pi's username and password, then click **OK**.


raspberrypi - VNC Viewer

Authentication

Authenticate to VNC Server
raspberrypi::5900 (TCP)

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

Username: pi

Password: ••••••

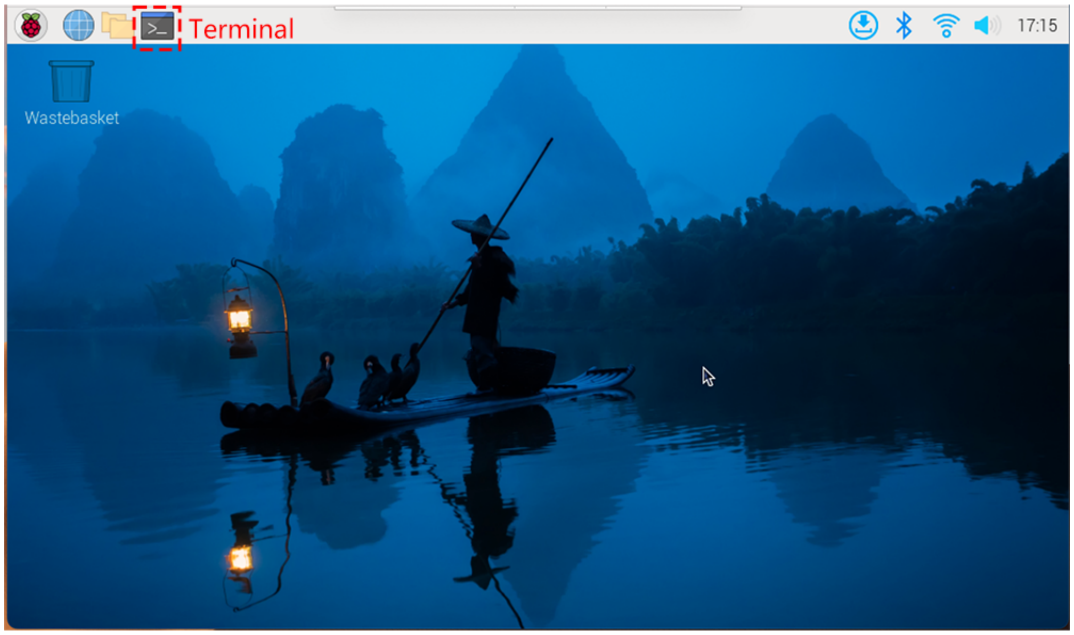
Remember password

Forgot password?

Catchphrase: Bundle secret slang. Nickel cola sponsor.

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

4. After a few seconds, the Raspberry Pi OS desktop will be displayed. Now you can open the Terminal to start entering commands.



Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.

- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

CHAPTER

EIGHT

HARDWARE

When you are writing code, you may need to know how each module works or the role of each pin, then please see this chapter.

In this chapter you will find a description of each module's function, technical parameters and working principle.

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

8.1 Robot HAT

is a multifunctional expansion board that allows Raspberry Pi to be quickly turned into a robot. An MCU is on board to extend the PWM output and ADC input for the Raspberry Pi, as well as a motor driver chip, I2S audio module and mono speaker. As well as the GPIOs that lead out of the Raspberry Pi itself.

It also comes with a Speaker, which can be used to play background music, sound effects and implement TTS functions to make your project more interesting.

Accepts 7-12V power input with 2 battery indicators, 1 charge indicator and 1 power indicator. The board also has a user available LED and a button for you to quickly test some effects.

For detailed instructions, please refer to: .

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

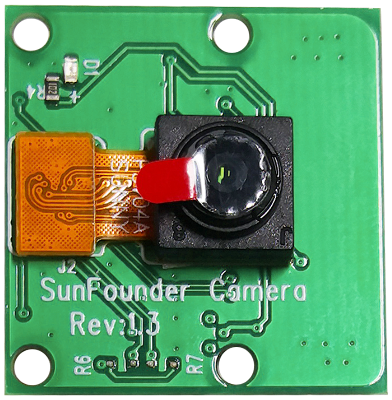
- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.

- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

8.2 Camera Module

Description



This is a 5MP Raspberry Pi camera module with OV5647 sensor. It's plug and play, connect the included ribbon cable to the CSI (Camera Serial Interface) port on your Raspberry Pi and you're ready to go.

The board is small, about 25mm x 23mm x 9mm, and weighs 3g, making it ideal for mobile or other size and weight-critical applications. The camera module has a native resolution of 5 megapixels and has an on-board fixed focus lens that captures still images at 2592 x 1944 pixels, and also supports 1080p30, 720p60 and 640x480p90 video.

Note: The module is only capable of capturing pictures and videos, not sound.

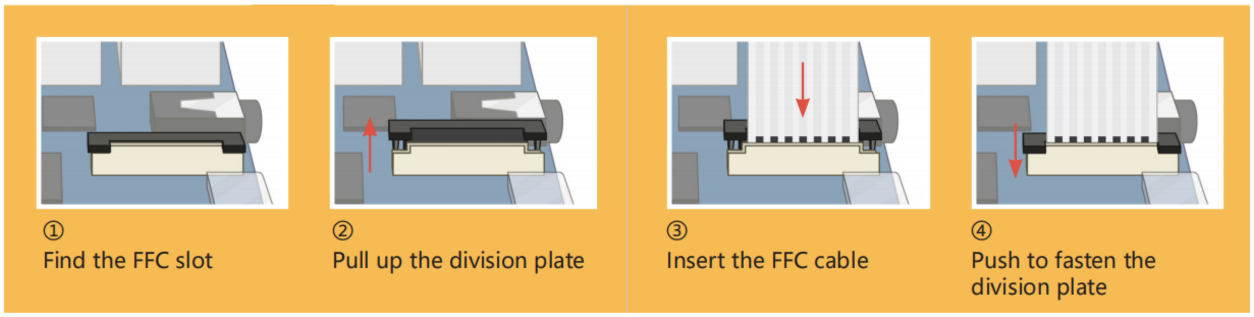
Specification

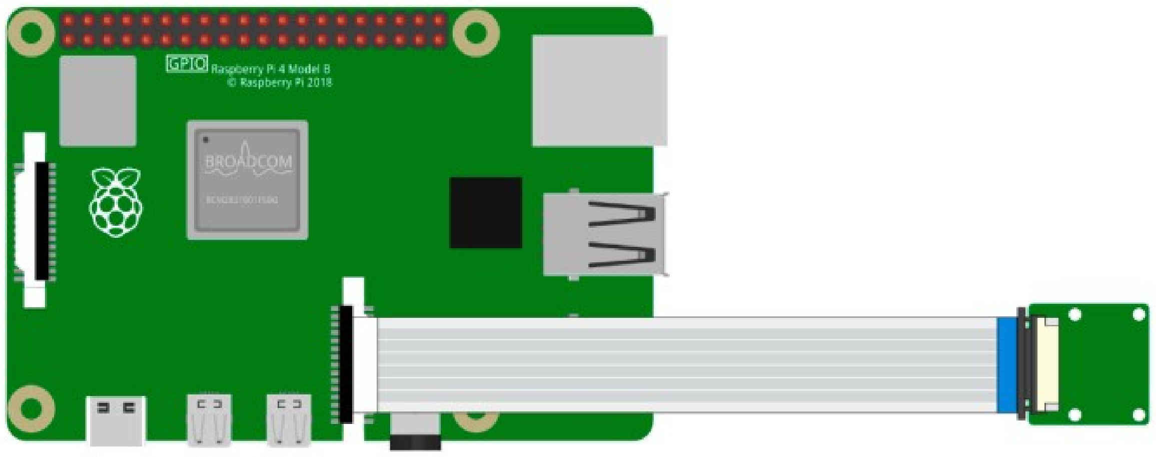
- **Static Images Resolution**: 2592×1944
- **Supported Video Resolution**: 1080p/30 fps, 720p/ 60fps and 640 x480p 60/90 video recording
- **Aperture (F)**: 1.8
- **Visual Angle**: 65 degree
- **Dimension**: 24mmx23.5mmx8mm
- **Weight**: 3g
- **Interface**: CSI connector
- **Supported OS**: Raspberry Pi OS(latest version recommended)

Assemble the Camera Module

On the camera module or Raspberry Pi, you will find a flat plastic connector. Carefully pull out the black fixing switch until the fixing switch is partially pulled out. Insert the FFC cable into the plastic connector in the direction shown and push the fixing switch back into place.

If the FFC wire is installed correctly, it will be straight and will not pull out when you gently pull on it. If not, reinstall it again.


① Find the FFC slot
② Pull up the division plate
③ Insert the FFC cable
④ Push to fasten the division plate



Warning: Do not install the camera with the power on, it may damage your camera.

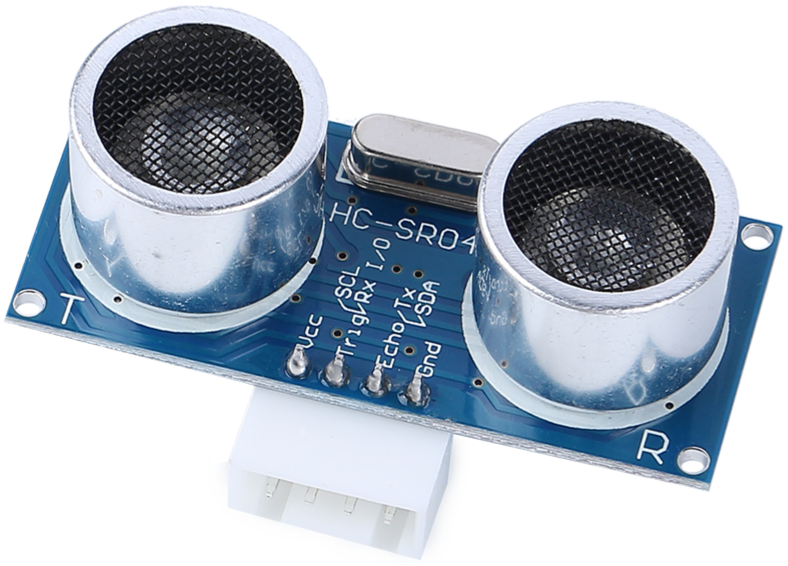
Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

8.3 Ultrasonic Module



- **TRIG**: Trigger Pulse Input
- **ECHO**: Echo Pulse Output
- **GND**: Ground
- **VCC**: 5V Supply

This is the HC-SR04 ultrasonic distance sensor, providing non-contact measurement from 2 cm to 400 cm with a range accuracy of up to 3 mm. Included on the module is an ultrasonic transmitter, a receiver and a control circuit.

You only need to connect 4 pins: VCC (power), Trig (trigger), Echo (receive) and GND (ground) to make it easy to use for your measurement projects.

Features

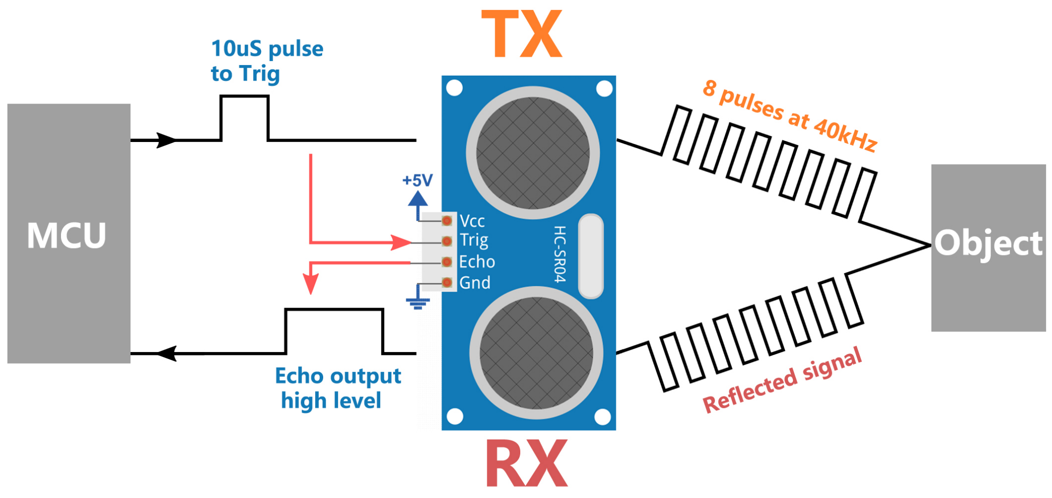
- Working Voltage: DC5V
- Working Current: 16mA
- Working Frequency: 40Hz
- Max Range: 500cm
- Min Range: 2cm
- Trigger Input Signal: 10uS TTL pulse
- Echo Output Signal: Input TTL lever signal and the range in proportion
- Connector: XH2.54-4P
- Dimension: 46x20.5x15 mm

Principle

The basic principles are as follows:

- Using IO trigger for at least 10us high level signal.
- The module sends an 8 cycle burst of ultrasound at 40 kHz and detects whether a pulse signal is received.

- Echo will output a high level if a signal is returned; the duration of the high level is the time from emission to return.
- Distance = (high level time x velocity of sound (340M/S)) / 2



Formula:

- us / 58 = centimeters distance
- us / 148 = inch distance
- distance = high level time x velocity (340M/S) / 2

Application Notes

- This module should not be connected under power up, if necessary, let the module's GND be connected first. Otherwise, it will affect the work of the module.
- The area of the object to be measured should be at least 0.5 square meters and as flat as possible. Otherwise, it will affect results.

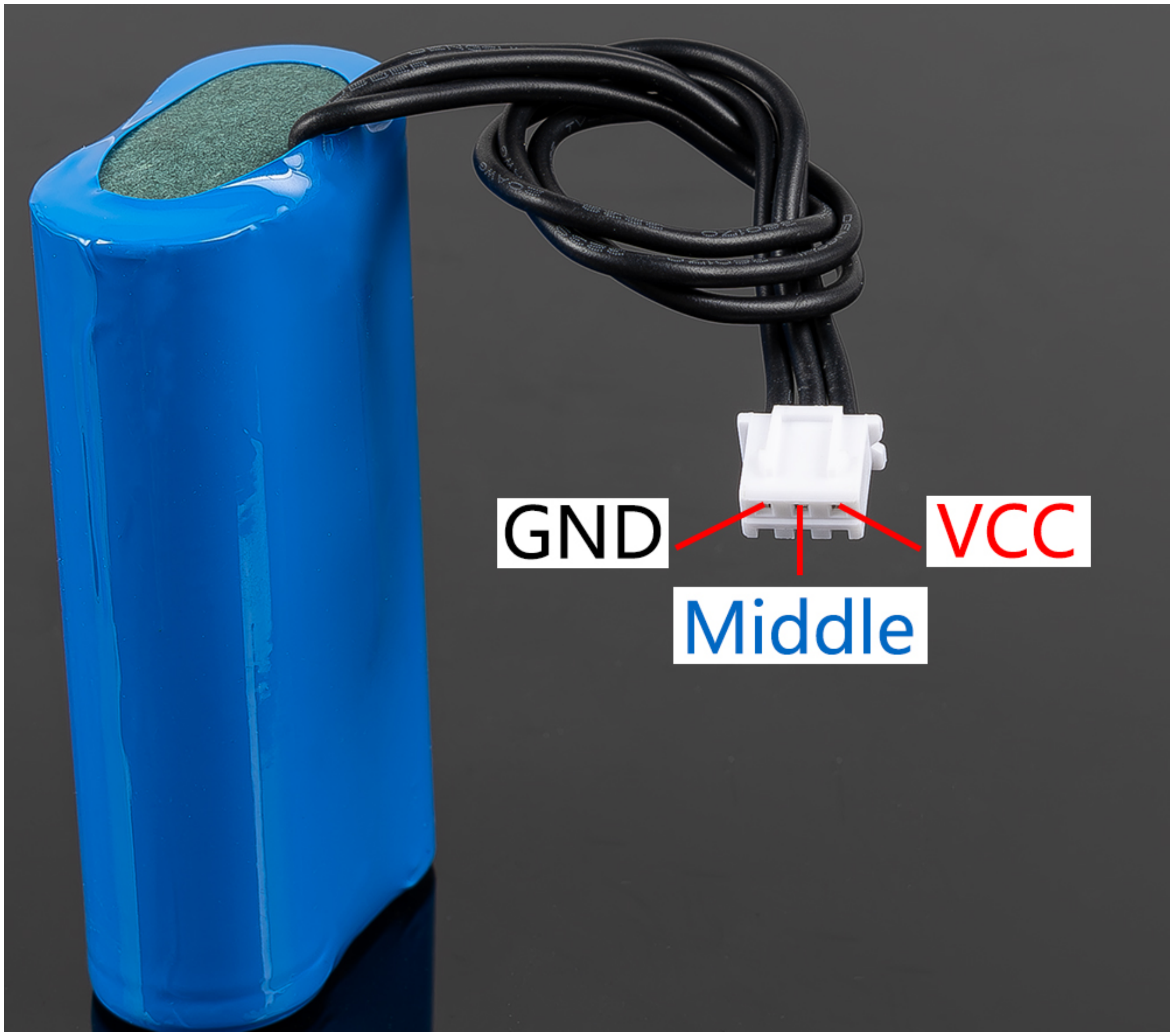
Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

8.4 3-pin Battery



- **VCC**: Battery positive terminal, here there are two sets of VCC and GND is to increase the current and reduce the resistance.
- **Middle**: To balance the voltage between the two cells and thus protect the battery.
- **GND**: Negative battery terminal.

This is a custom battery pack made by SunFounder consisting of two 18650 batteries with a capacity of 2000mAh. The connector is XH2.54 3P, which can be charged directly after being inserted into the Robot HAT.

Features

- Composition: Li-ion
- Battery Capacity: 2000mAh, 14.8Wh
- Battery Weight: 90.8g
- Number of Cells: 2

- Connector: XH2.54 3P
- Over-discharge protection: 6.0V

Note: Hello, welcome to the SunFounder Raspberry Pi & Arduino & ESP32 Enthusiasts Community on Facebook! Dive deeper into Raspberry Pi, Arduino, and ESP32 with fellow enthusiasts.

Why Join?

- **Expert Support**: Solve post-sale issues and technical challenges with help from our community and team.
- **Learn & Share**: Exchange tips and tutorials to enhance your skills.
- **Exclusive Previews**: Get early access to new product announcements and sneak peeks.
- **Special Discounts**: Enjoy exclusive discounts on our newest products.
- **Festive Promotions and Giveaways**: Take part in giveaways and holiday promotions.

Ready to explore and create with us? Click [] and join today!

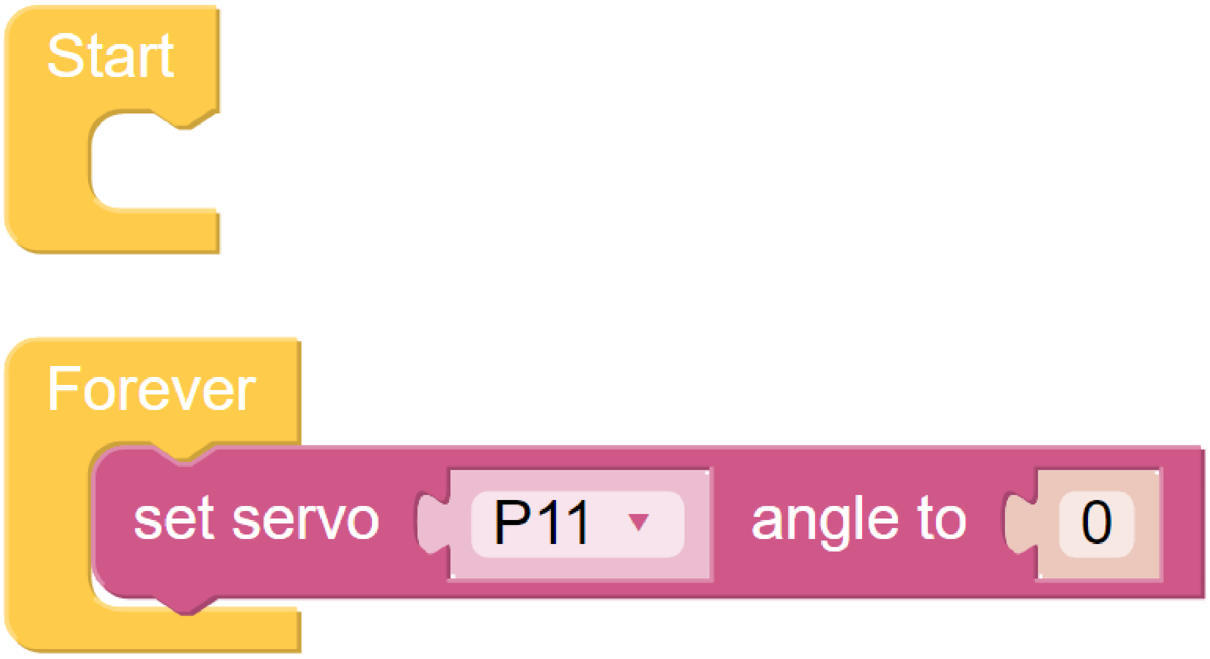
CHAPTER

NINE

FAQ

9.1 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.



9.2 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.

9.3 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 servo, the servo will automatically return to its original position.

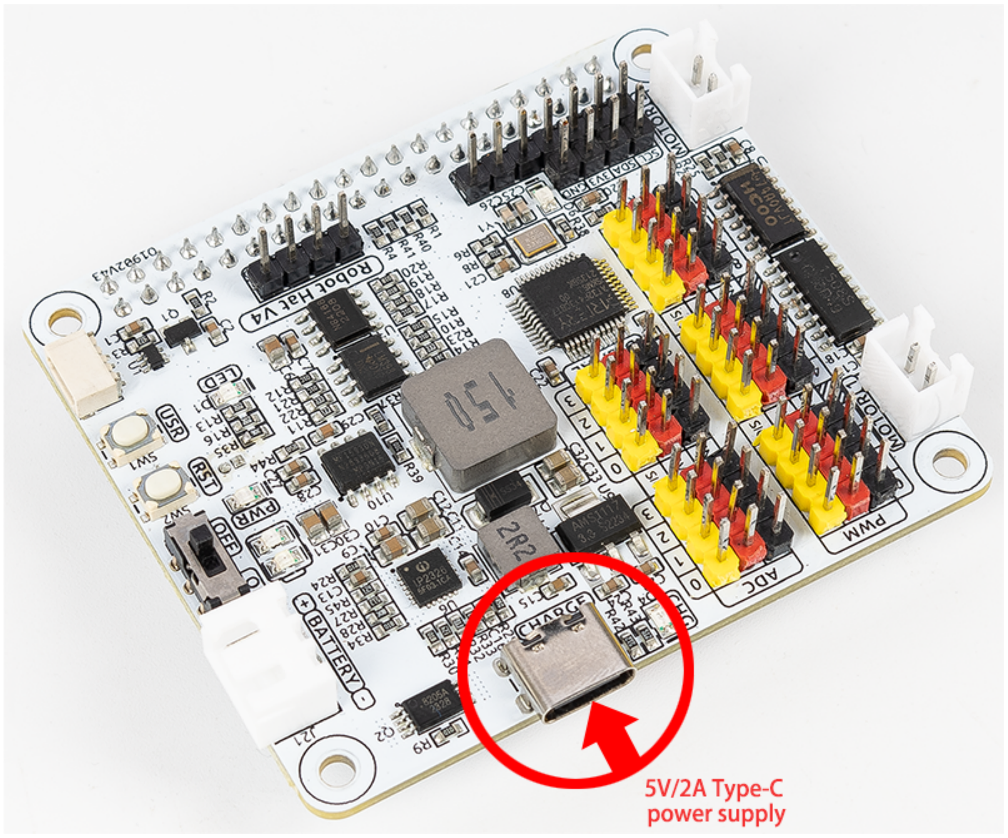
9.4 Q4: About the Robot HAT Detailed Tutorial?

You can find a comprehensive tutorial about the Robot HAT here, including information on its hardware and API.

-

9.5 Q5: About the Battery Charger?

To charge the battery, simply connect a 5V/2A Type-C power supply to the Robot Hat's power port. There's no need to turn on the Robot Hat's power switch during charging. You can also use the device while charging the battery.



During charging, the input power is boosted by the charging chip to charge the battery and simultaneously supply the DC-DC converter for external use, with a charging power of approximately 10W. If external power consumption remains high for an extended period, the battery may supplement the power supply, similar to using a phone while charging. However, be mindful of the battery's capacity to avoid completely depleting it during simultaneous charging and usage.

9.6 Q6: Camera Not Working?

If the camera is not displaying or displaying incorrectly, follow these troubleshooting steps:

1. Ensure the FPC cable of the camera is securely connected. It is recommended to reconnect the camera and then power on the device.
2. Use the following command to check if the camera is recognized.

```
libcamera-hello
```

CHAPTER

TEN

COPYRIGHT NOTICE