
SunFounder picar-x

www.sunfounder.com

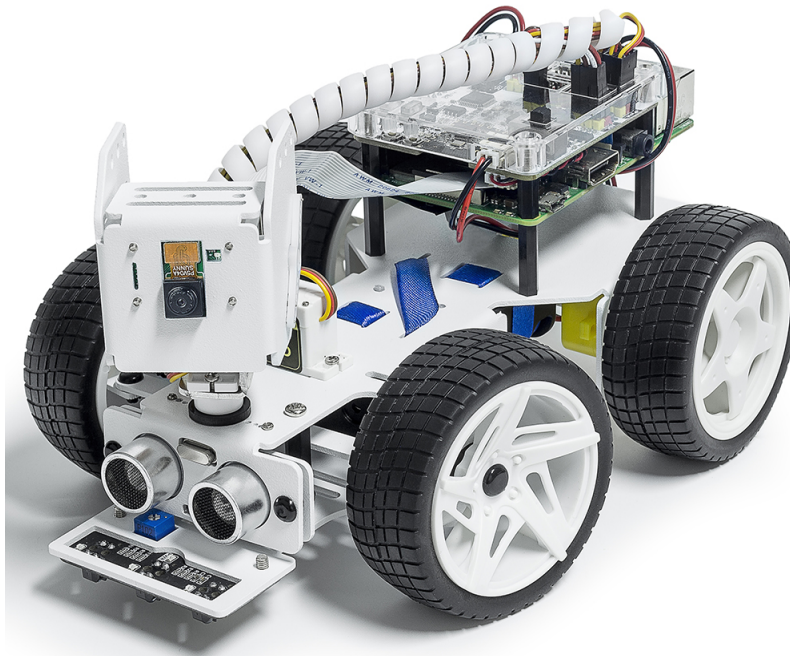
Aug 19, 2022

CONTENTS

1	Introduction	5
1.1	The History of Self-driving Cars	5
1.2	About PiCar-X	7
1.3	Deep Learning and Neural Networks	8
2	Component List and Assembly Instructions	9
3	About Robot HAT	11
4	Play with Python	13
4.1	Quick Guide on Python	13
4.1.1	What Do We Need?	13
4.1.2	Installing the OS	15
4.1.3	Set up Your Raspberry Pi	22
4.1.4	Install All the Modules	28
4.1.5	Servo Adjust	31
4.2	Calibrating the PiCar-X	33
4.3	Let PiCar-X Move	37
4.4	Obstacle Avoidance	39
4.5	Line Tracking	41
4.6	Text to Speech	42
4.7	Computer Vision	44
4.8	Color Detection	47
4.9	Face Detection	52
4.10	Video Car	55
4.11	Controlled by the APP	58
5	Play with Ezblock	67
5.1	Quick Guide on EzBlock	67
5.1.1	Servo Adjust	67
5.1.2	Install and Configure EzBlock Studio	69
5.2	Move	70
5.3	Remote Control	74
5.4	Test Ultrasonic Module	76
5.5	Test Grayscale Module	78
5.6	Color Detection	79
5.7	Face Detection	82
5.8	Sound Effect	85
5.9	Background Music	87
5.10	Say Hello	88

5.11	Music Car	91
5.12	Cliff Detection	92
5.13	Minecart	94
5.14	Minecart Plus	97
5.15	Bullfight	102
5.16	Beware of Pedestrians	104
5.17	Traffic Sign Detection	106
5.18	Orienteering	111
6	Appendix	117
6.1	I2C Configuration	117
6.2	Remote Desktop	119
6.2.1	VNC	119
6.2.2	XRDP	125
6.3	About the Battery	128
7	FAQ	131
8	Thank You	133
9	Copyright Notice	135

Thanks for choosing our PiCar-X.



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 Ezblokk*

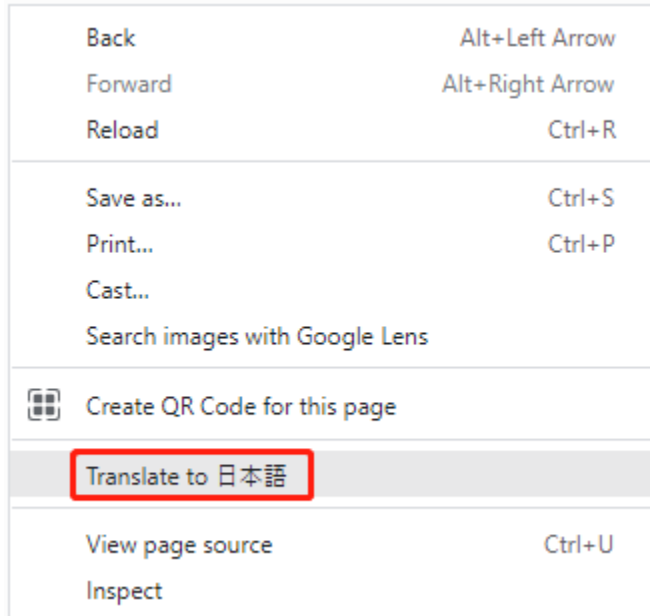
- In this section, you will use a Blockly based APP, Ezblokk Studio, which, like Scratch, allows you to drag and drop blocks to make Picar-X move.
- It is required to reinstall the SD card with the operating system we provide with pre-installed Ezblokk environment before programming. It is recommended to use a new or unused TF card for this section.
- Ezblokk Studio is available for nearly all types of devices, including Macs, PCs, and Androids.
- Ezblokk 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.

About the display language

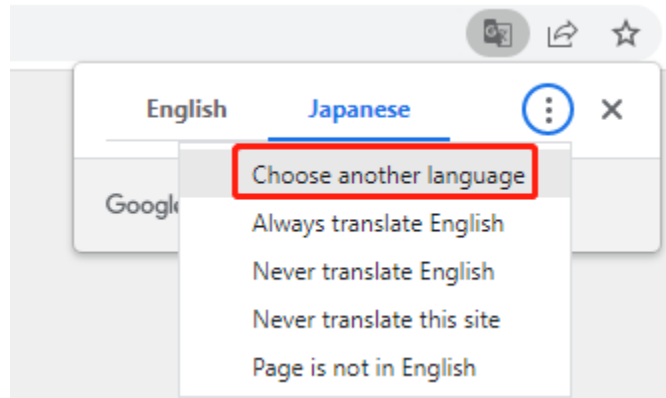
In addition to English, we are working on other languages for this course. Please contact service@sunfounder.com if you are interested in helping, and we will give you a free product in return. In the meantime, we recommend using Google Translate to convert English to the language you want to see.

The steps are as follows.

- In this course page, right-click and select **Translate to xx**. If the current language is not what you want, you can change it later.



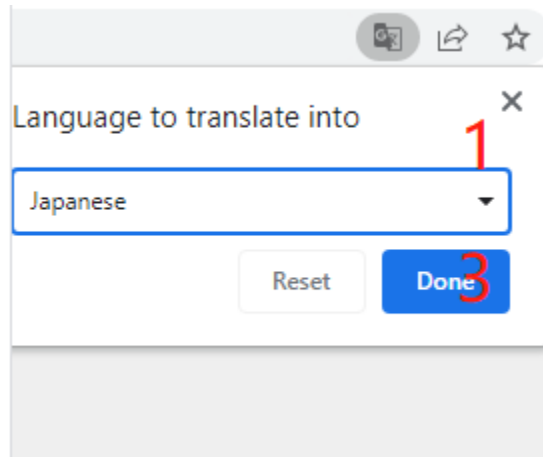
- There will be a language popup in the upper right corner. Click on the menu button to **choose another language**.



- Select the language from the inverted triangle box, and then click **Done**.

- Arabic
- Armenian
- Azerbaijani
- Bangla
- Basque
- Belarusian
- Bosnian
- Bulgarian
- Burmese
- Catalan

2



Content

INTRODUCTION

1.1 The History of Self-driving Cars

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

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

- Reference: [History of self-driving cars - Wikipedia](#)

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

Level 4 reference:

- [SAE Levels of Driving Automation™](#)
- [ABI Research Forecasts 8 Million Vehicles to Ship with SAE Level 3, 4 and 5 Autonomous Technology in 2025](#)



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

- In October 2010, a driverless truck designed by the Italian technology company **Vislab** took three months to travel from Italy to China, with a total distance of 8, 077 miles.
- In April 2015, a car designed by **Delphi Automotive** traveled from San Francisco to New York , traversing 3,400 miles, completing 99 percent of that distance under computer control.
- In December 2018, **Alphabet's Waymo** launched a level 4 self-driving taxi service in Arizona , where they had already been testing driverless cars since 2008. With no one in the driver's seat, the vehicles operated for more than a year and traveled over 10 million miles.
- In October 2020, **Baidu** fully opened its **Apollo Robotaxi self-driving cab service** in Beijing. The driving routes cover local residential, commercial, leisure, and industrial parks areas, and offer a fully autonomous driving system.

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

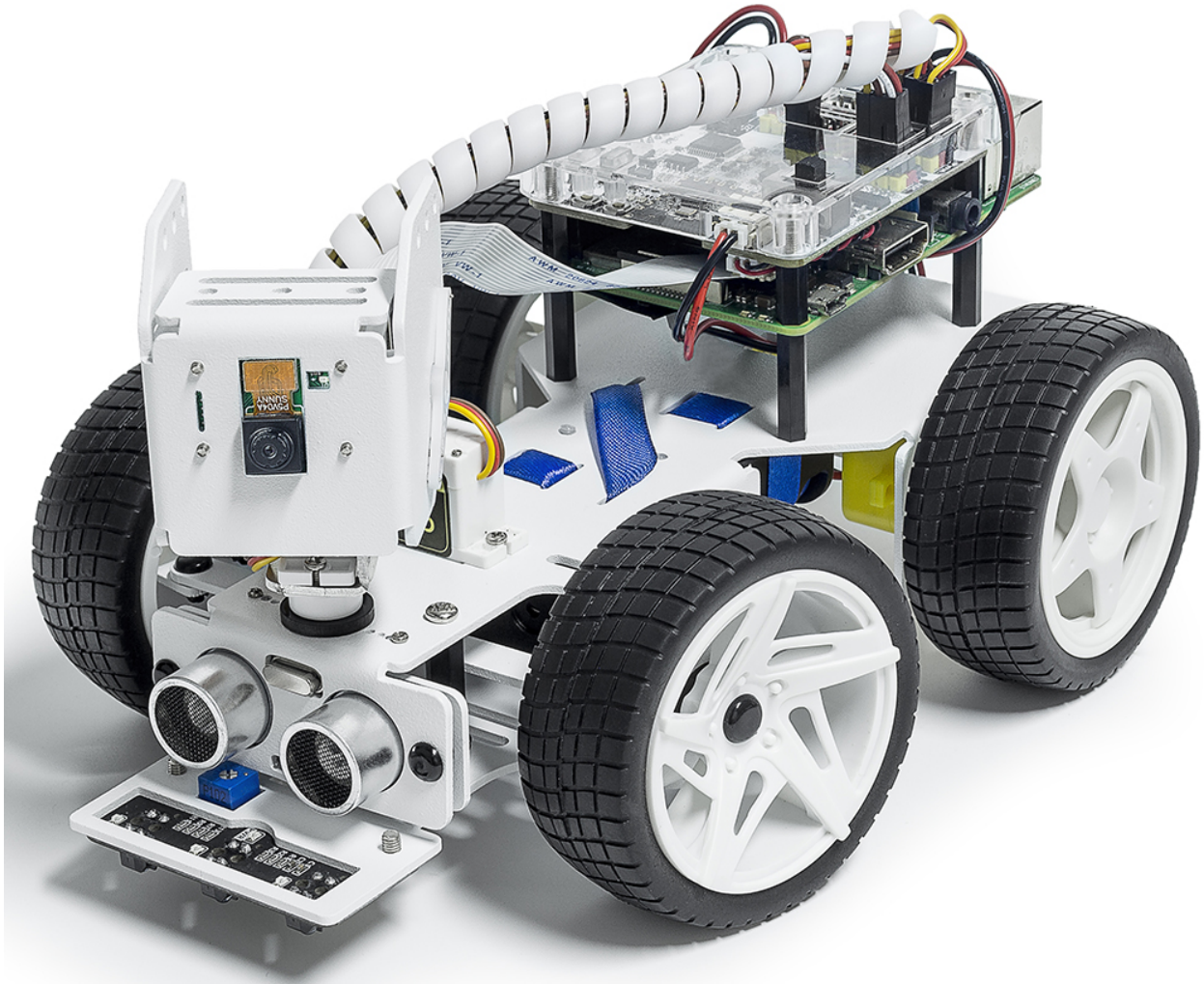
According to **RAND's report** , reaching the appropriate level of autonomous learning requires training data from hundreds of millions, or even hundreds of billions of miles to establish a level of reliability.

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

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

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

1.2 About PiCar-X



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

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

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

1.3 Deep Learning and Neural Networks

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

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

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

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

COMPONENT LIST AND ASSEMBLY INSTRUCTIONS

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

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

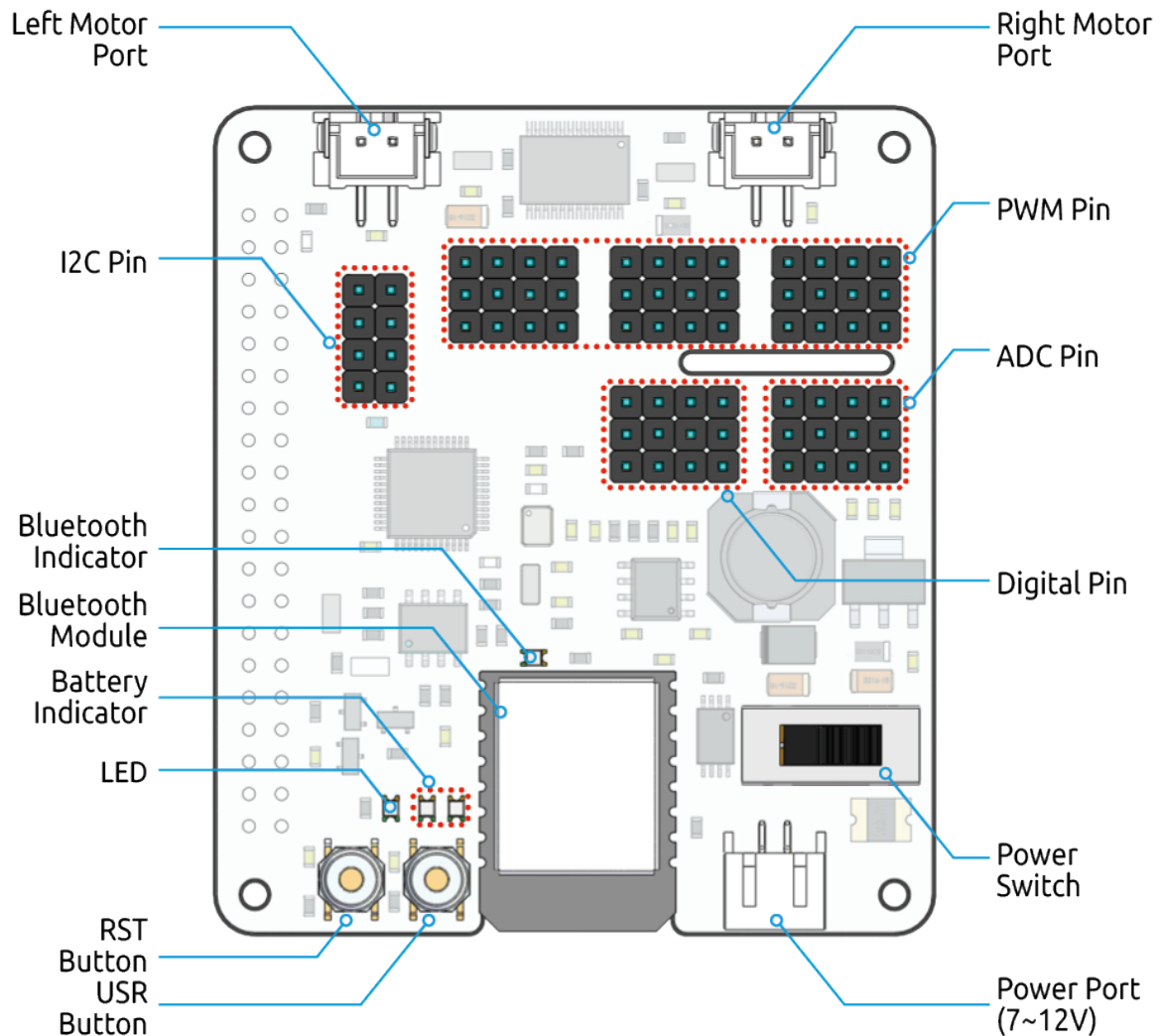
[PDF]Component List and Assembly of PiCar-X.

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

Note:

1. Before assembling, you need to buy 2 18650 batteries and fully charge them, refer to *About the Battery*.
 2. Robot HAT cannot charge the battery, so you need to buy a battery charger at the same time.
-

ABOUT ROBOT HAT



RST Button

- A short-press of the RST Button will cause any running programs to reset.
- A long-press of the RST Button until the LED lights up, and then releasing will disconnect the Robot HAT's Bluetooth chip.

USR Button

- The functions of the USR Button can be configured through programming. (Pressing down leads to a input of “0”, and releasing produces a input of “1”)

LED

- Configured through programming (Outputting ‘1’ turns the LED on, Outputting ‘0’ turns the LED off.)

Battery Indicator

- Battery voltage above 7.8V will light up the two indicator LEDs. Battery voltage ranging from 6.7V to 7.8V will only light up one LED, voltage below 6.7V will turn both LEDs off.

Bluetooth Indicator

- The Bluetooth indicator LED will stay on with a solid Bluetooth connection, and blink rapidly during a signal transmission. The LED will blink at 1-second intervals if the Bluetooth is disconnected.

Note: You can see more details in the [Robot HAT Documentation](#).

PLAY WITH PYTHON

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

4.1 Quick Guide on Python

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

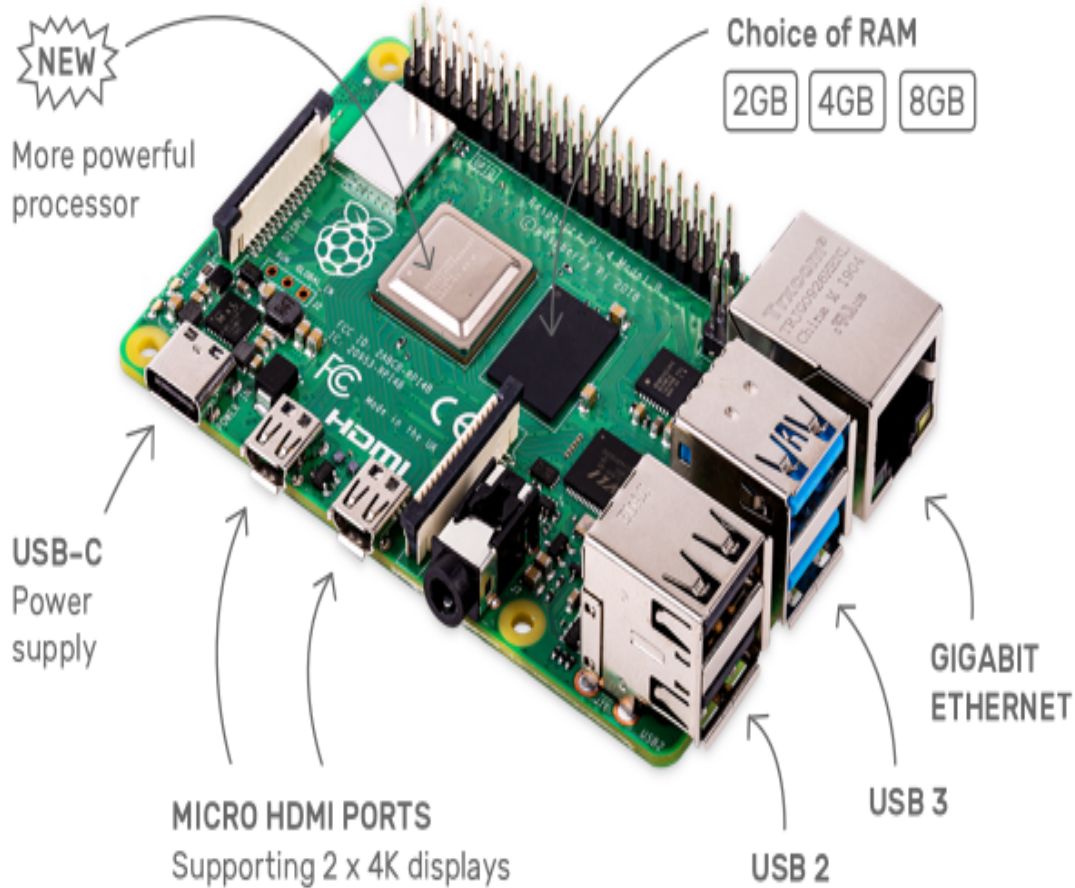
If you are familiar with Raspberry Pi and can open the command line successfully, then you can skip the first 3 parts and then complete the last part.

4.1.1 What Do We Need?

Required Components

Raspberry Pi

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



Power Adapter

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

Micro SD Card

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

Optional Components

Screen

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

Mouse & Keyboard

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

HDMI

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

Case

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

Sound or Earphone

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

4.1.2 Installing the OS

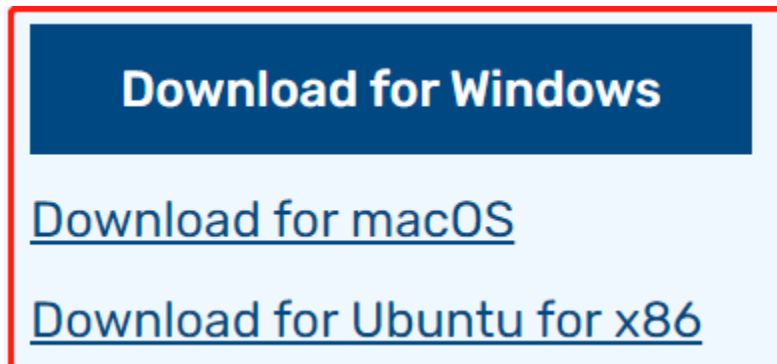
Required Components

Any Raspberry Pi	1 * Personal Computer
1 * Micro SD card	

Step 1

Raspberry Pi have developed a graphical SD card writing tool that works on Mac OS, Ubuntu 18.04 and Windows, and is the easiest option for most users as it will download the image and install it automatically to the SD card.

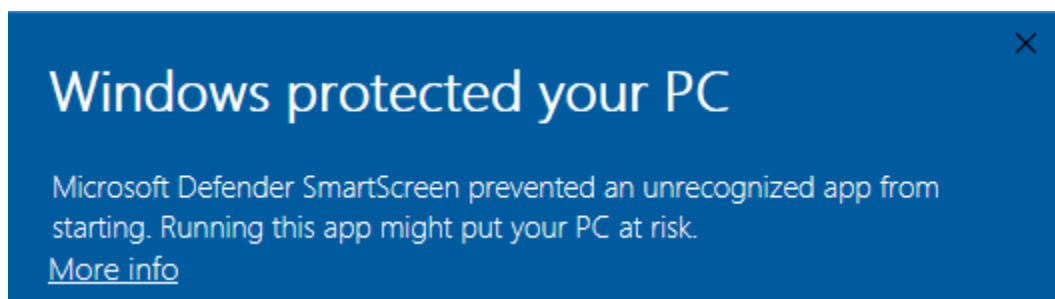
Visit the download page: <https://www.raspberrypi.org/software/>. Click on the link for the Raspberry Pi Imager that matches your operating system, when the download finishes, click it to launch the installer.



Step 2

When you launch the installer, your operating system may try to block you from running it. For example, on Windows I receive the following message:

If this pops up, click on **More info** and then **Run anyway**, then follow the instructions to install the Raspberry Pi Imager.



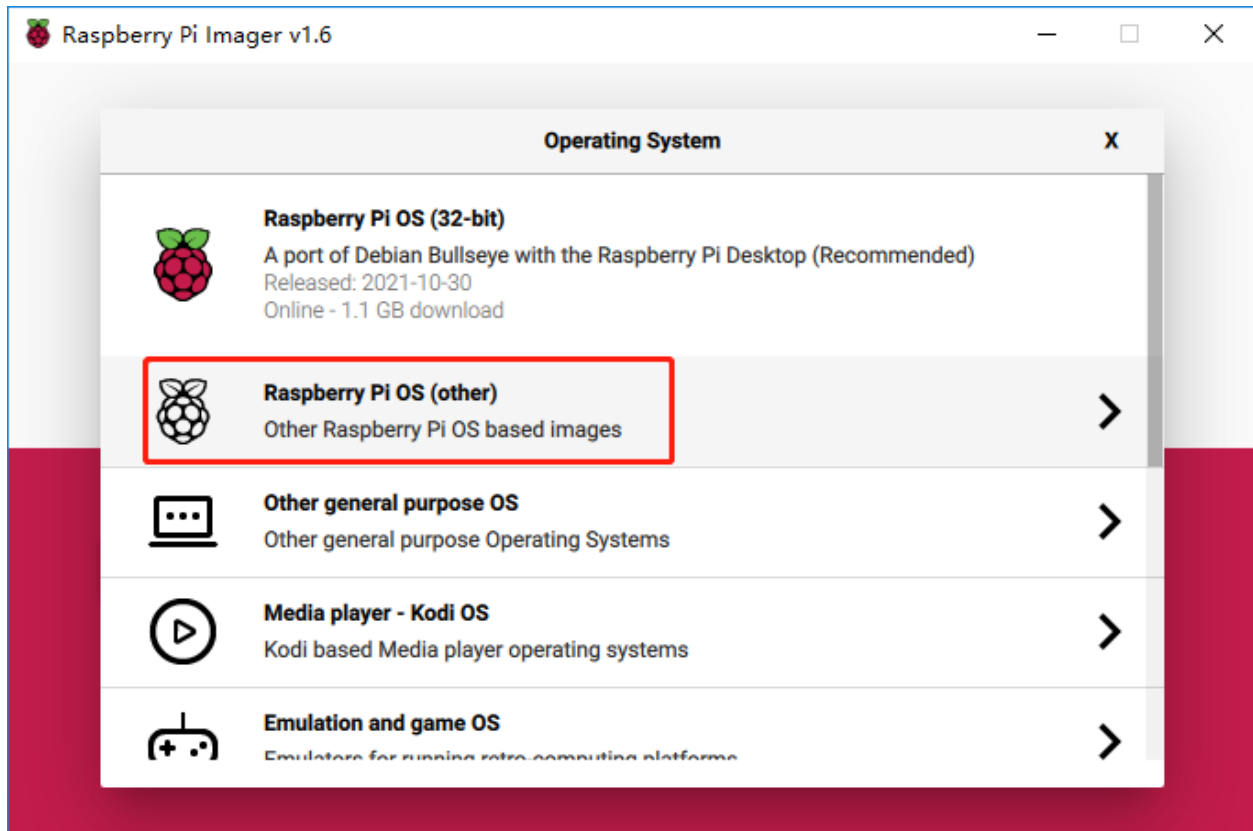
Step 3

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

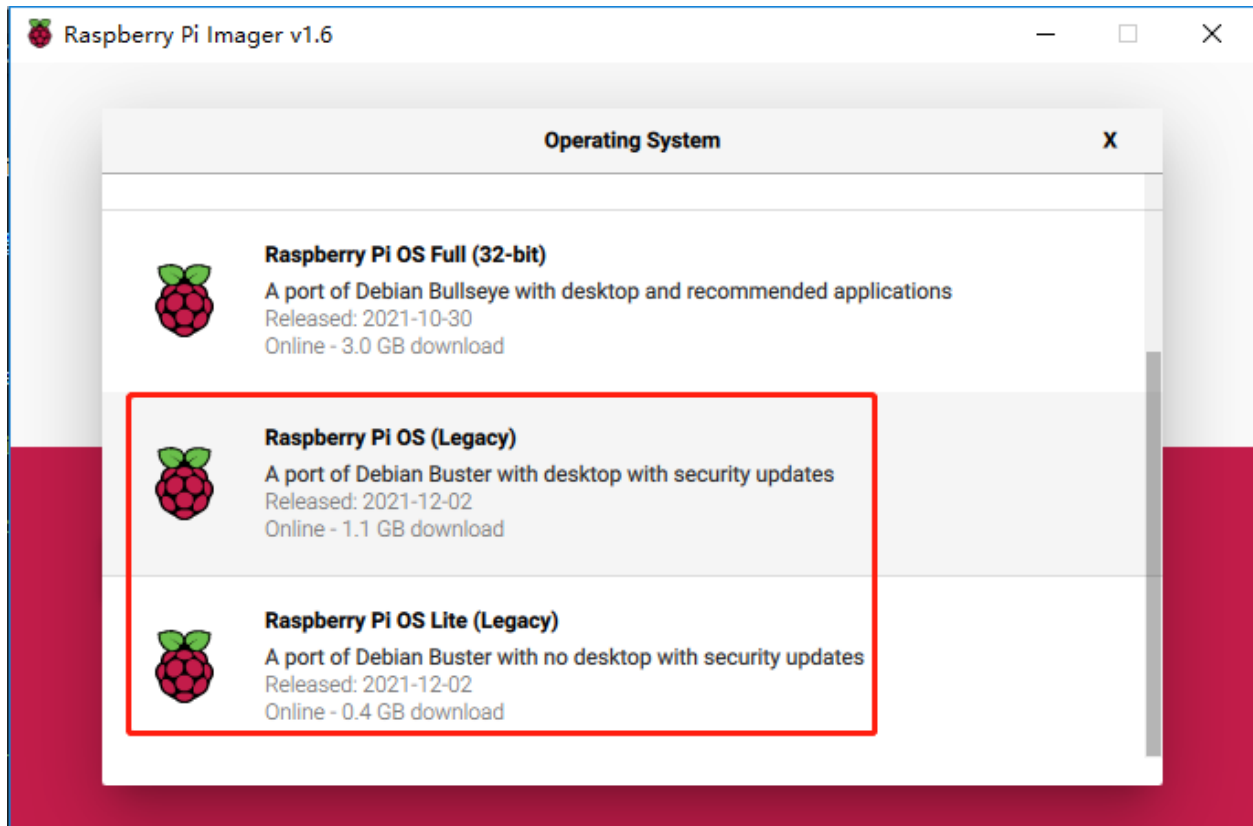
Step 4

Warning: Upgrading the Raspberry Pi OS to **Debian Bullseye** will cause some features to not work, so it is recommended to continue using the **Debian Buster** version.

In the Raspberry Pi Imager, click **CHOOSE OS** -> **Raspberry Pi OS(other)**.

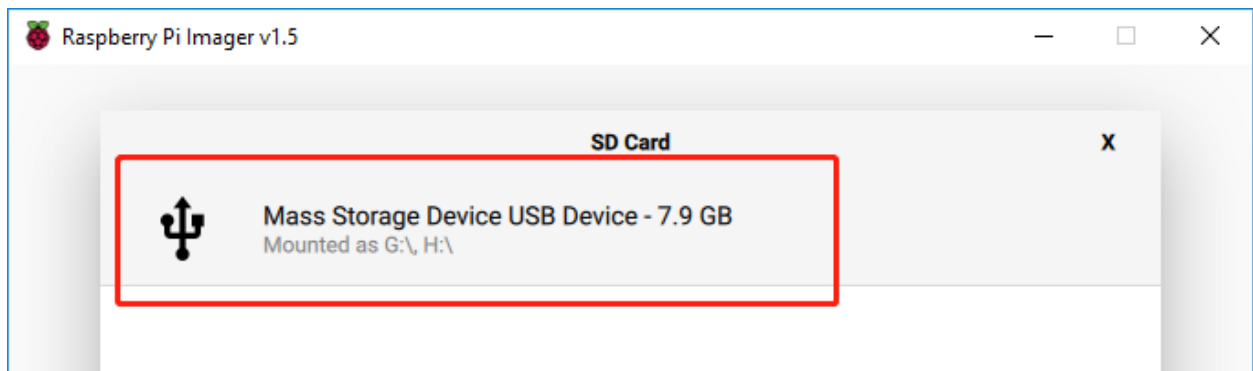


Scroll down to the end of the newly opened page and you will see **Raspberry Pi OS(Legacy)** and **Raspberry Pi OS Lite(Legacy)**, these are security updates for Debian Buster, the difference between them is with or without the desktop. It is recommended to install **Raspberry Pi OS(Legacy)**, this will allow you to go to the Raspberry Pi desktop to view the footage taken by the camera..



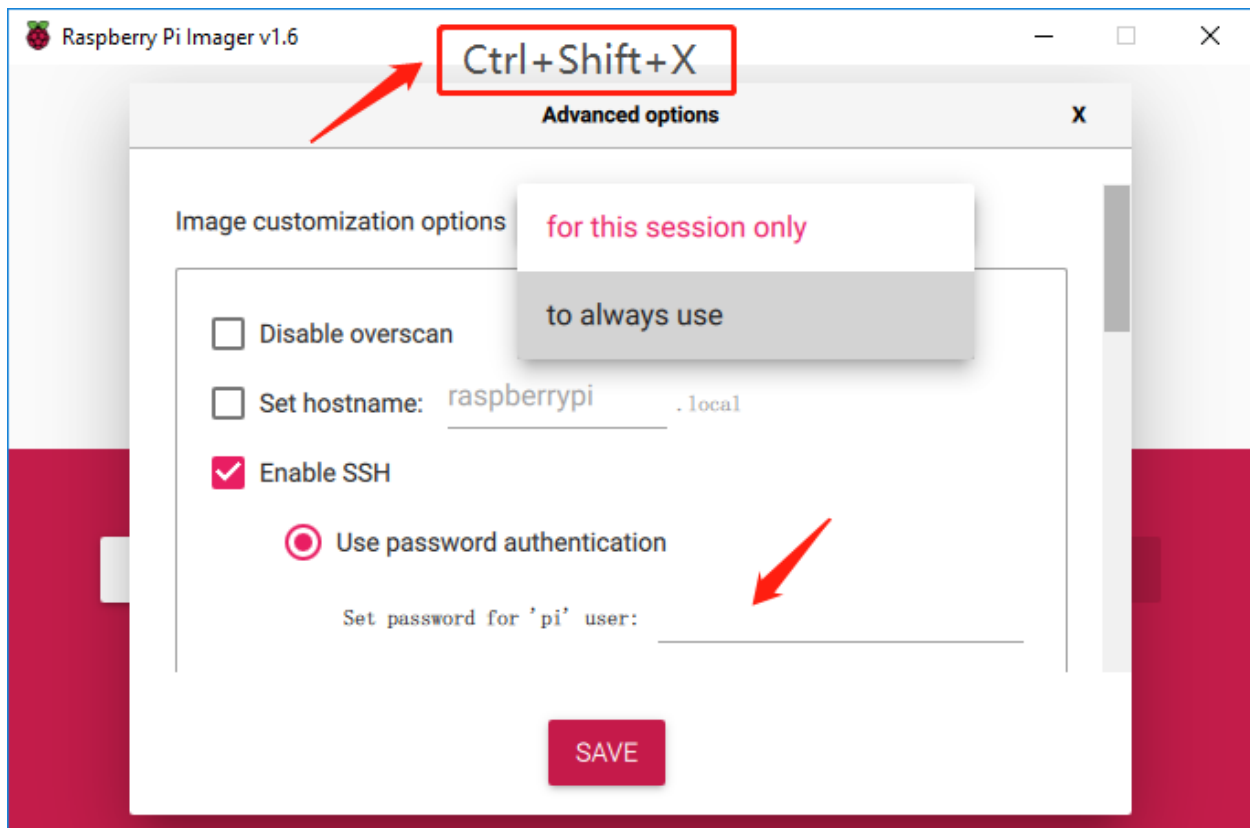
Step 5

Select the SD card you are using.



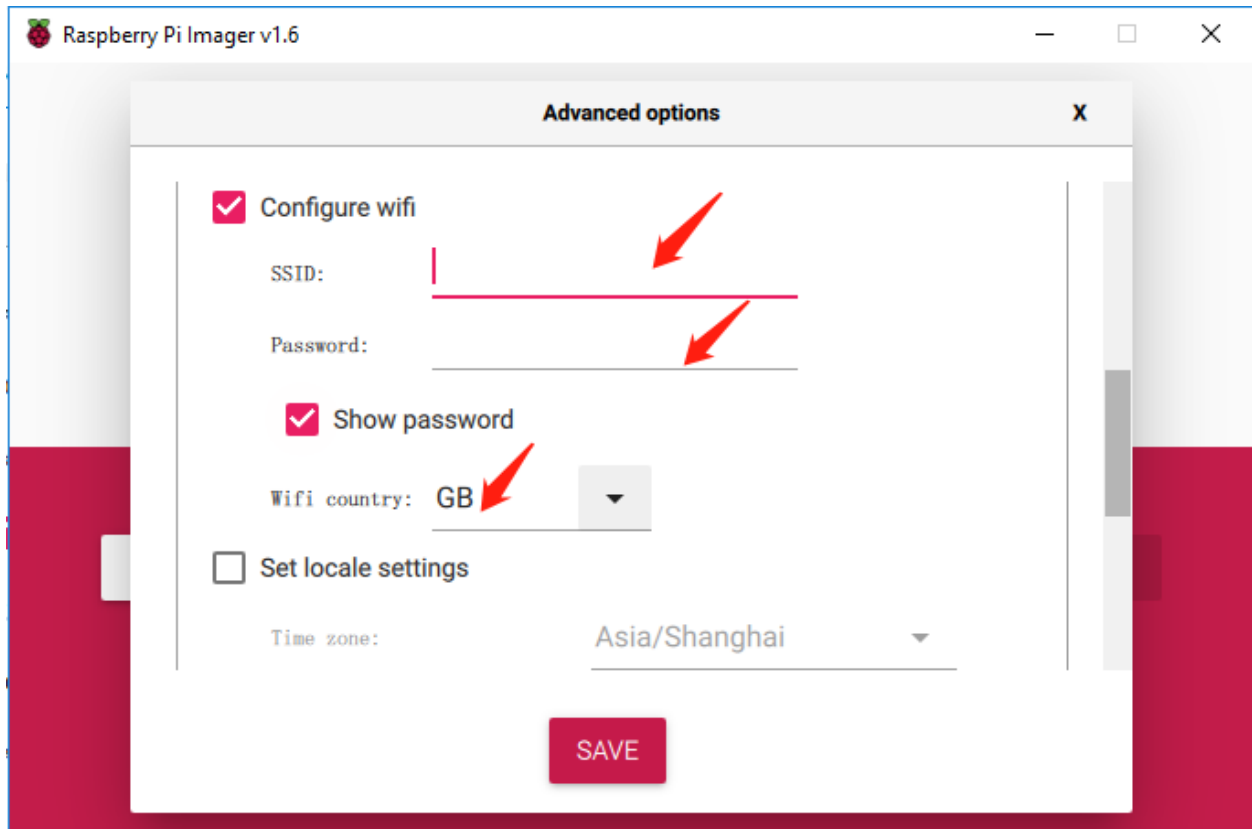
Step 6

Press **Ctrl+Shift+X** or click the **setting** button to open the **Advanced options** page to enable SSH and configure wifi, these 2 items must be set, the others depend on your choice . You can choose to always use this image customization options.



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

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



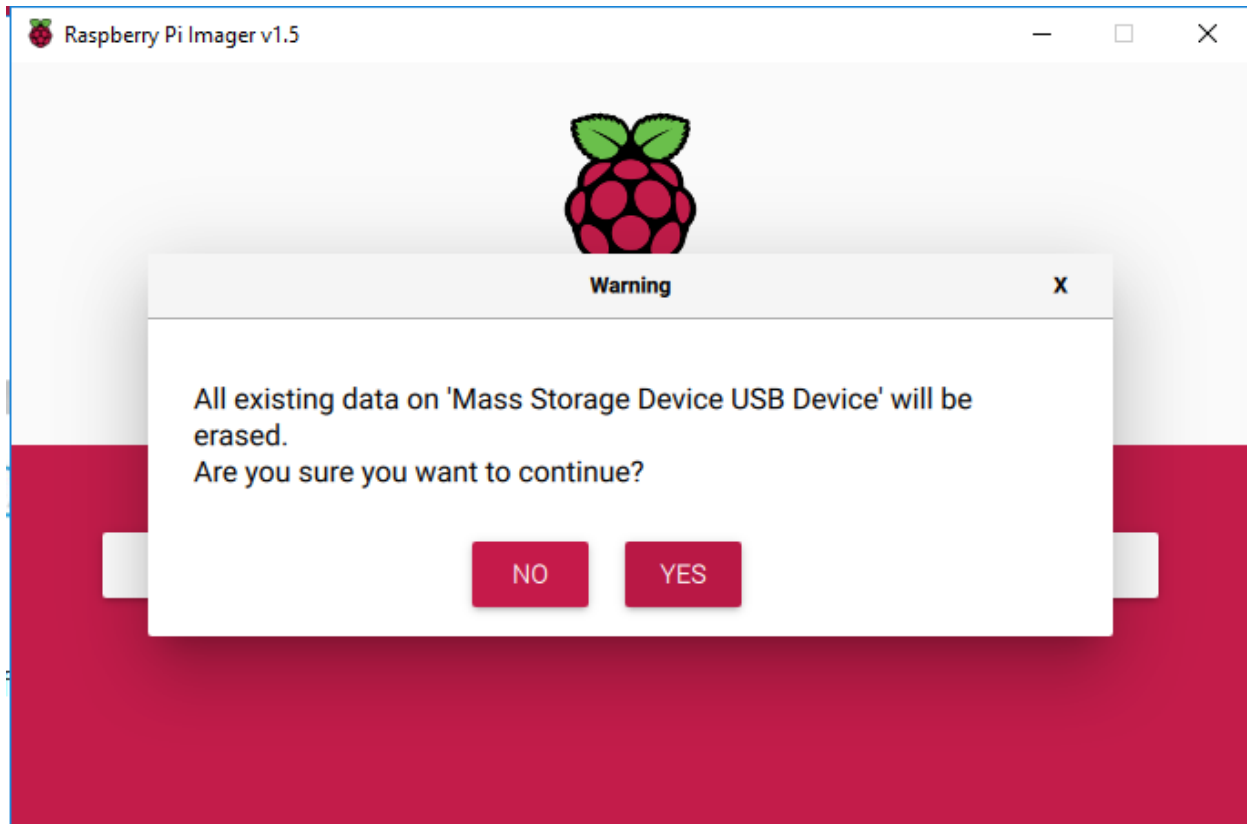
Step 7

Click the **WRITE** button.



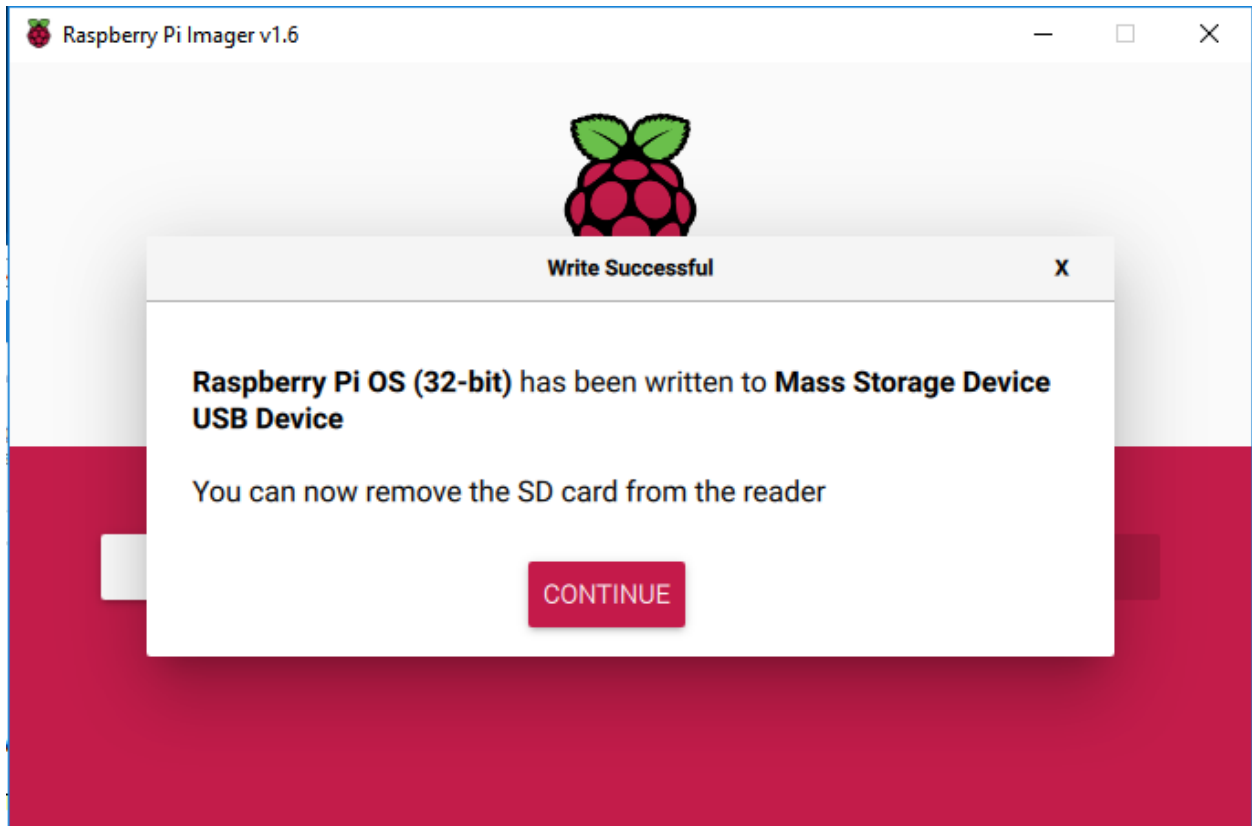
Step 8

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



Step 9

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



4.1.3 Set up Your Raspberry Pi

If You Have a Screen

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

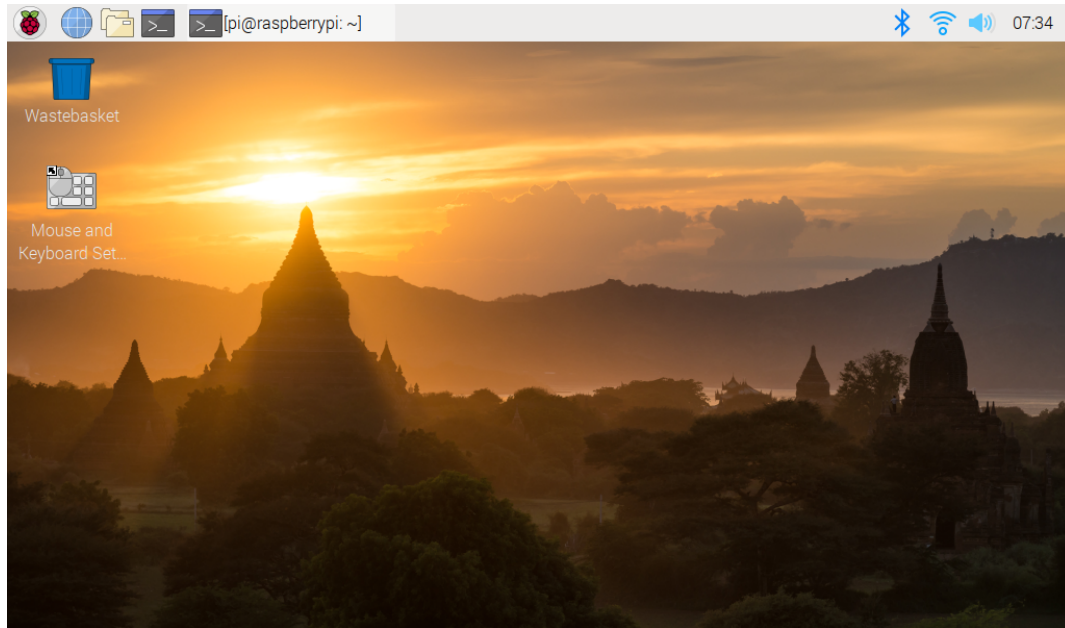
Required Components

Any Raspberry Pi	1 * Power Adapter
1 * Micro SD card	1 * Screen Power Adapter
1 * HDMI cable	1 * Screen
1 * Mouse	1 * Keyboard

1. Insert the SD card you've set up with Raspberry Pi OS into the micro SD card slot on the underside of your Raspberry Pi.
2. Plug in the Mouse and Keyboard.
3. Connect the screen to Raspberry Pi's HDMI port and make sure your screen is plugged into a wall socket and switched on.

Note: If you use a Raspberry Pi 4, you need to connect the screen to the HDMI0 (nearest the power in port).

4. Use the power adapter to power the Raspberry Pi. After a few seconds, the Raspberry Pi OS desktop will be displayed.



If You Have No Screen

If you don't have a display, you can log in to the Raspberry Pi remotely, but before that, you need to get the IP of the Raspberry Pi.

Get the IP Address

After the Raspberry Pi is connected to WIFI, we need to get the IP address of it. There are many ways to know the IP address, and two of them are listed as follows.

1. Checking via the router

If you have permission to log in the router(such as a home network), you can check the addresses assigned to Raspberry Pi on the admin interface of router.

The default hostname of the Raspberry Pi OS is **raspberrypi**, and you need to find it. (If you are using ArchLinuxARM system, please find alarmpi.)

2. Network Segment Scanning

You can also use network scanning to look up the IP address of Raspberry Pi. You can apply the software, **Advanced IP scanner** and so on.

Scan the IP range set, and the name of all connected devices will be displayed. Similarly, the default hostname of the Raspberry Pi OS is **raspberrypi**, if you haven't modified it.

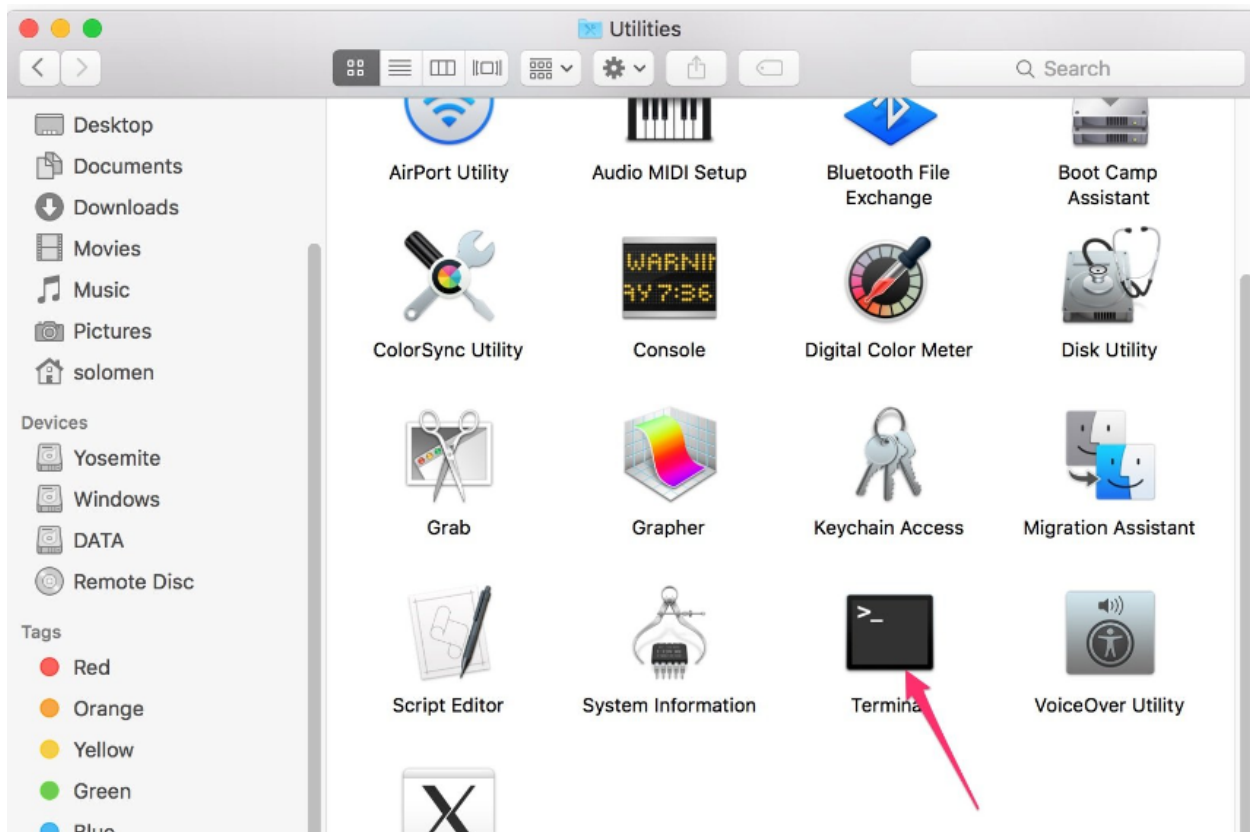
Use the SSH Remote Control

We can open the Bash Shell of Raspberry Pi by applying SSH. Bash is the standard default shell of Linux. The Shell itself is a program written in C that is the bridge linking the customers and Unix/Linux. Moreover, it can help to complete most of the work needed.

For Linux or/Mac OS X Users

Step 1

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



Step 2

Type in `ssh pi@ip_address` . “pi” is your username and “ip_address” is your IP address. For example:

```
ssh pi@192.168.18.197
```

Step 3

Input “yes”.


```
1. ssh pi@192.168.18.197 (ssh)
Last login: Fri Apr 12 16:56:20 on ttys000
# hang_chen @ hang-chendeMacBook-Pro in ~ [17:09:55]
$ ssh pi@192.168.18.197
The authenticity of host '192.168.18.197 (192.168.18.197)' can't be established.
ECDSA key fingerprint is SHA256:60tKKQtCCRvUCohWmvVcbp7tBHtQL0f8/0kusPjVsEU.
Are you sure you want to continue connecting (yes/no)?
```

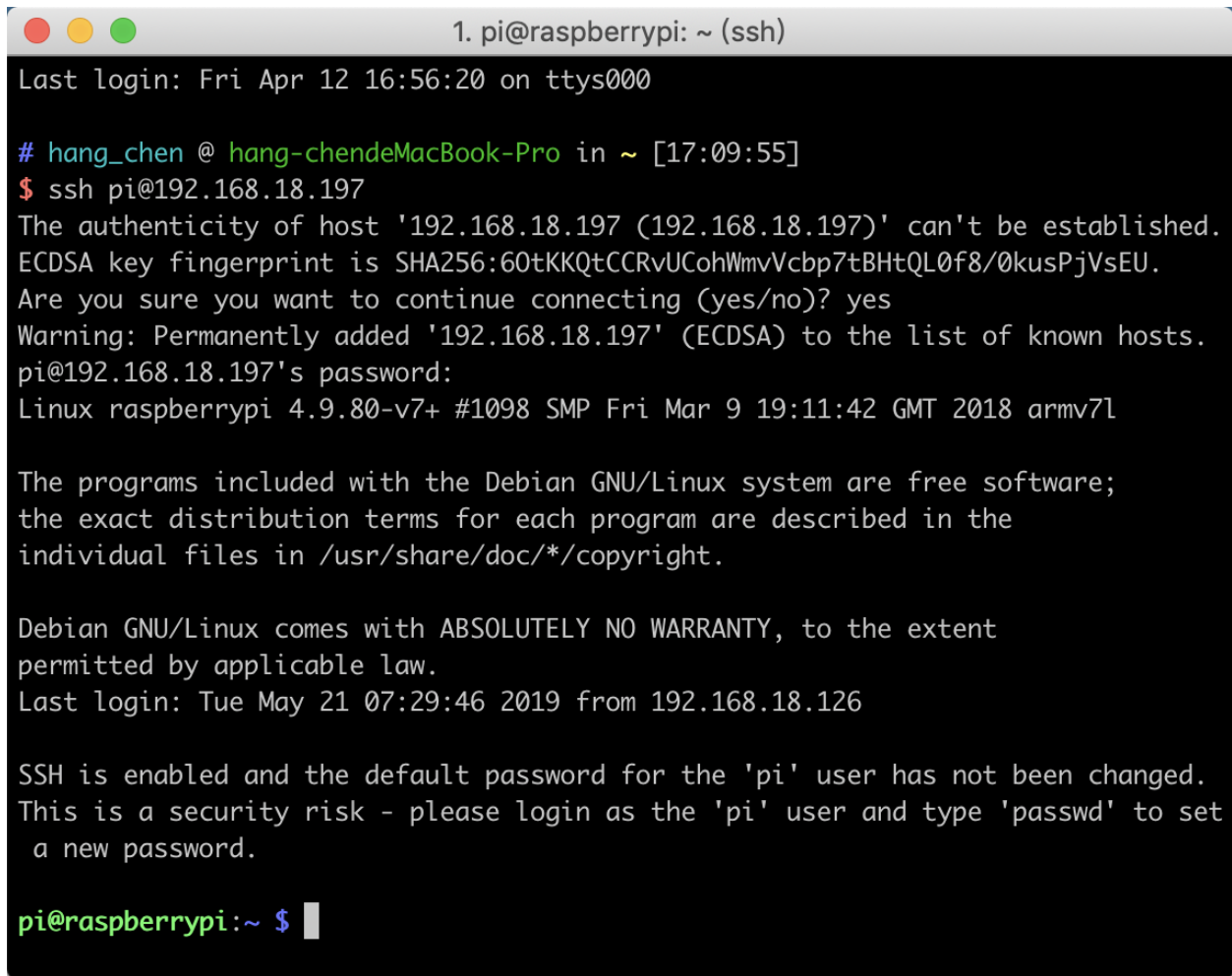
Step 4

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

```
1. ssh pi@192.168.18.197 (ssh)
Last login: Fri Apr 12 16:56:20 on ttys000
# hang_chen @ hang-chendeMacBook-Pro in ~ [17:09:55]
$ ssh pi@192.168.18.197
The authenticity of host '192.168.18.197 (192.168.18.197)' can't be established.
ECDSA key fingerprint is SHA256:60tKKQtCCRvUCohWmvVcbp7tBHtQL0f8/0kusPjVsEU.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.18.197' (ECDSA) to the list of known hosts.
pi@192.168.18.197's password: ?
```

Step 5

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



```
1. pi@raspberrypi: ~ (ssh)
Last login: Fri Apr 12 16:56:20 on ttys000
# hang_chen @ hang-chendeMacBook-Pro in ~ [17:09:55]
$ ssh pi@192.168.18.197
The authenticity of host '192.168.18.197 (192.168.18.197)' can't be established.
ECDSA key fingerprint is SHA256:60tKKQtCCRvUCohWmvVcbp7tBHtQL0f8/0kusPjVsEU.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '192.168.18.197' (ECDSA) to the list of known hosts.
pi@192.168.18.197's password:
Linux raspberrypi 4.9.80-v7+ #1098 SMP Fri Mar 9 19:11:42 GMT 2018 armv7l

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

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Tue May 21 07:29:46 2019 from 192.168.18.126

SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set
a new password.

pi@raspberrypi:~ $ █
```

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

For Windows Users

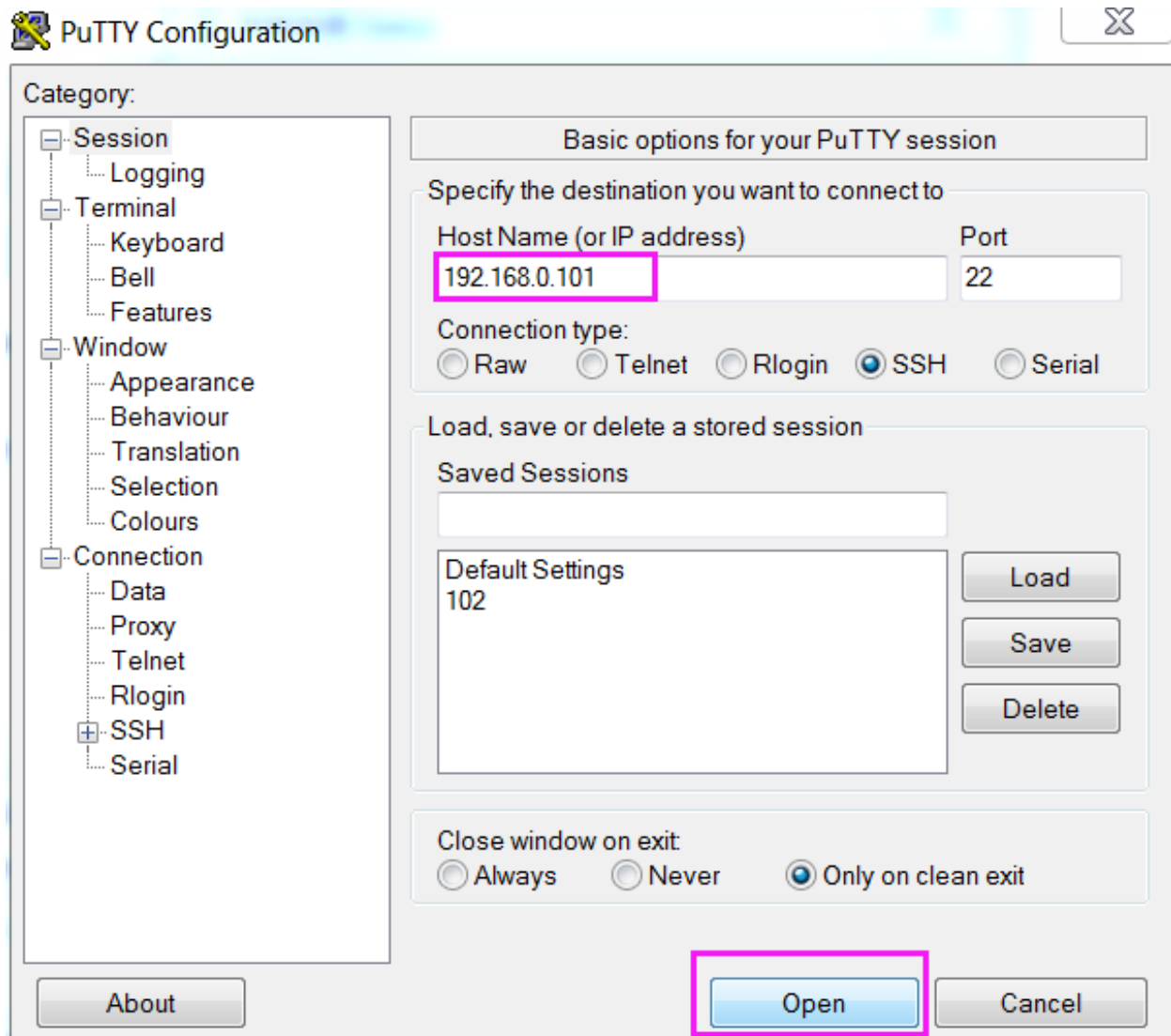
If you're a Windows user, you can use SSH with the application of some software. Here, we recommend **PuTTY**.

Step 1

Download PuTTY.

Step 2

Open PuTTY and click **Session** on the left tree-alike structure. Enter the IP address of the RPi in the text box under **Host Name (or IP address)** and **22** under **Port** (by default it is 22).

**Step 3**

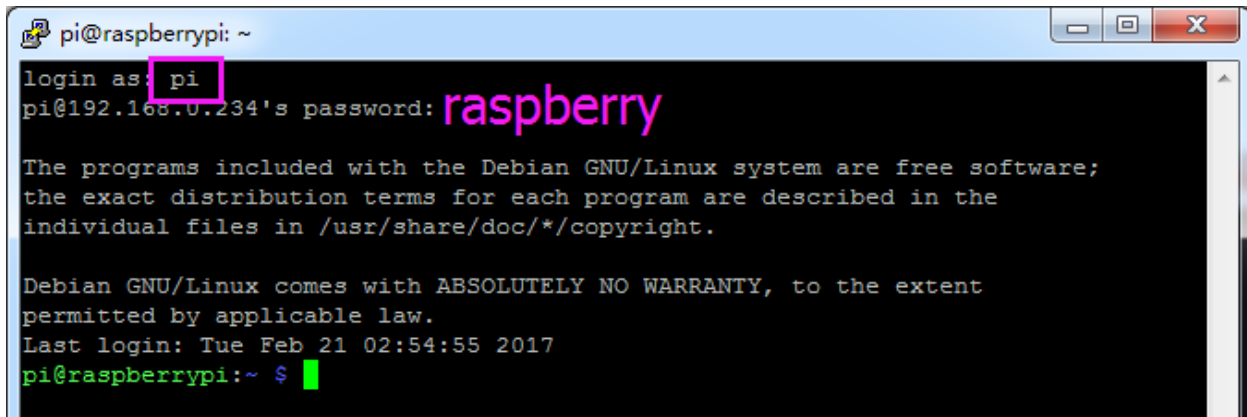
Click **Open**. Note that when you first log in to the Raspberry Pi with the IP address, there prompts a security reminder. Just click **Yes**.

Step 4

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

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

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



Step 5

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

Note: If you are not satisfied with using the command window to control the Raspberry Pi, you can also use the remote desktop function, which can help us manage the files in the Raspberry Pi easily.

For details on how to do this, please refer to *Remote Desktop*.

4.1.4 Install All the Modules

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

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

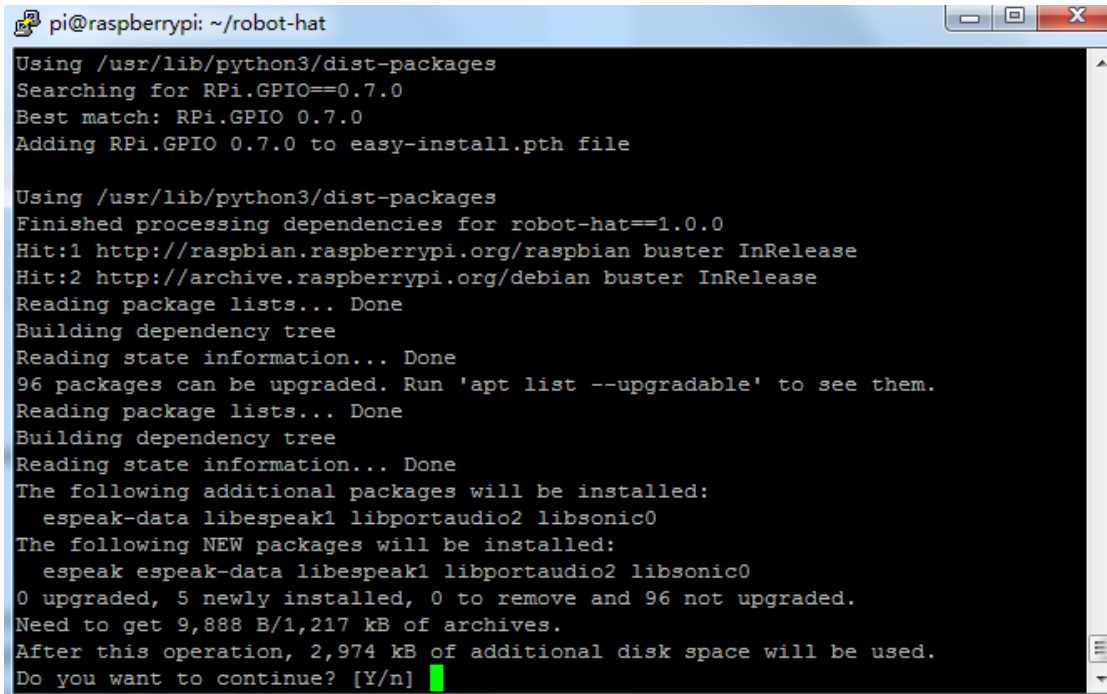
Note: Python3 related packages must be installed if you are installing the Lite version OS.

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

Install `robot-hat`.

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

Note: Running `setup.py` will download some necessary components. Your download may have failed due to network issues. You may need to download again at this point. See the following interface, type `Y` and press Enter.



```

pi@raspberrypi: ~/robot-hat
Using /usr/lib/python3/dist-packages
Searching for RPi.GPIO==0.7.0
Best match: RPi.GPIO 0.7.0
Adding RPi.GPIO 0.7.0 to easy-install.pth file

Using /usr/lib/python3/dist-packages
Finished processing dependencies for robot-hat==1.0.0
Hit:1 http://raspbian.raspberrypi.org/raspbian buster InRelease
Hit:2 http://archive.raspberrypi.org/debian buster InRelease
Reading package lists... Done
Building dependency tree
Reading state information... Done
96 packages can be upgraded. Run 'apt list --upgradable' to see them.
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  espeak-data libespeak1 libportaudio2 libsonic0
The following NEW packages will be installed:
  espeak espeak-data libespeak1 libportaudio2 libsonic0
0 upgraded, 5 newly installed, 0 to remove and 96 not upgraded.
Need to get 9,888 B/1,217 kB of archives.
After this operation, 2,974 kB of additional disk space will be used.
Do you want to continue? [Y/n]

```

Then download and install the vilib module.

```

cd /home/pi/
git clone https://github.com/sunfounder/vilib.git
cd vilib
sudo python3 install.py

```

Download and install the picar-x module.

```

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

```

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

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

```

cd /home/pi/picar-x
sudo bash i2samp.sh

```

```

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

This script will install everything needed to use
i2s amplifier

--- Warning ---

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

If you want to see what this script does before
running it, you should run:
    \curl -sS github.com/adafruit/Raspberry-Pi-Installer-Scripts/i2samp
Do you wish to continue? [y/N] █

```

Type `y` and press enter to continue running the script.

```

pi@raspberrypi: ~/pisloth
running it, you should run:
    \curl -sS github.com/adafruit/Raspberry-Pi-Installer-Scripts/i2samp
Do you wish to continue? [y/N] y
Checking hardware requirements...

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

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

Disabling default sound driver
Configuring sound output

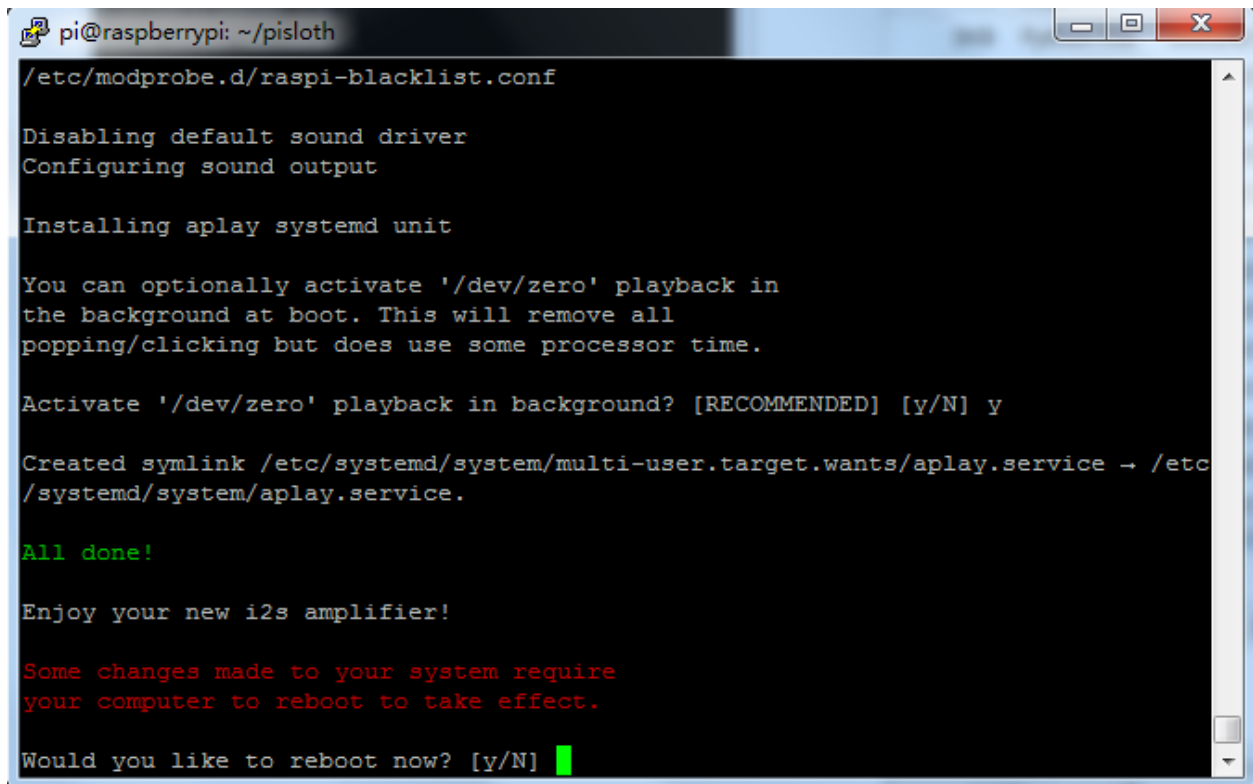
Installing aplay systemd unit

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

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

```

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



```
pi@raspberrypi: ~/pisloth
/etc/modprobe.d/raspi-blacklist.conf

Disabling default sound driver
Configuring sound output

Installing aplay systemd unit

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

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

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

All done!

Enjoy your new i2s amplifier!

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

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

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

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

4.1.5 Servo Adjust

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.

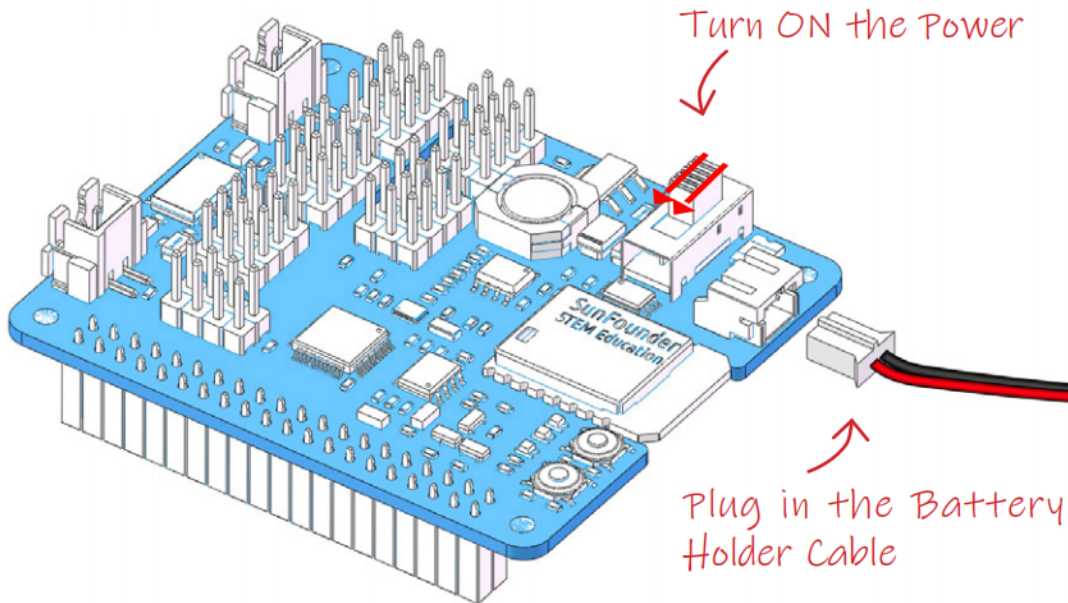
It is recommended that you *Calibrating the PiCar-X* after assembling it. The servo angle will be tilted due to possible deviations during assembly or limitations of the servo itself, so you can get the servo to a perfect state by calibrating it, usually the calibration angle is -5° ~ 5° .

But if the deviation angle is too big, you still have to go back to this section to set the servo angle to 0° , and then follow the instructions to reassemble the car.

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 to a different angle.



2. Follow the instructions on the assembly foldout, insert the battery holder cable and turn the power switch to the ON. Wait for 1-2 minutes, there will be a sound to indicate that the Raspberry Pi boots successfully.

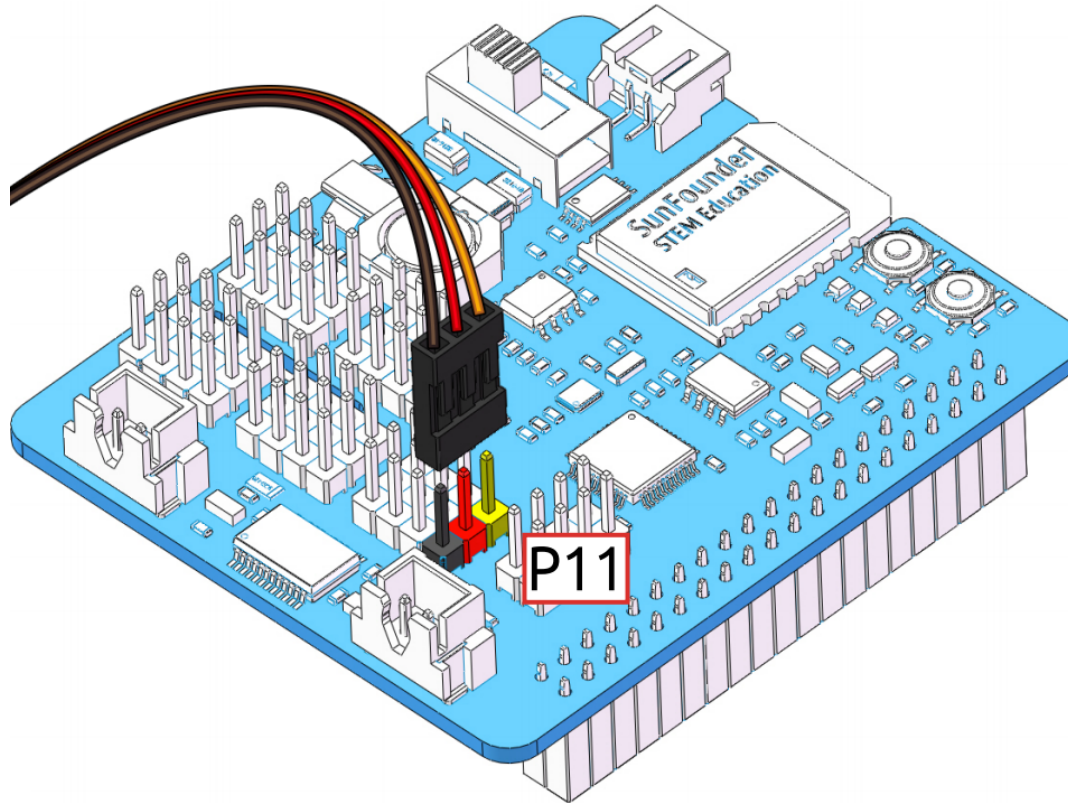


3. Now, run `servo_zeroing.py` in the `example/` folder.

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

Note: If you get an error, try re-enabling the Raspberry Pi's I2C port, see: [I2C Configuration](#).

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



5. At this point you will see the servo arm rotate to a specific position (0°). If the servo arm does not return to 0° , press the **RST** button to restart the Robot HAT.

Now you can continue the installation as instructed on the assembly foldout.

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° .

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

4.2 Calibrating the PiCar-X

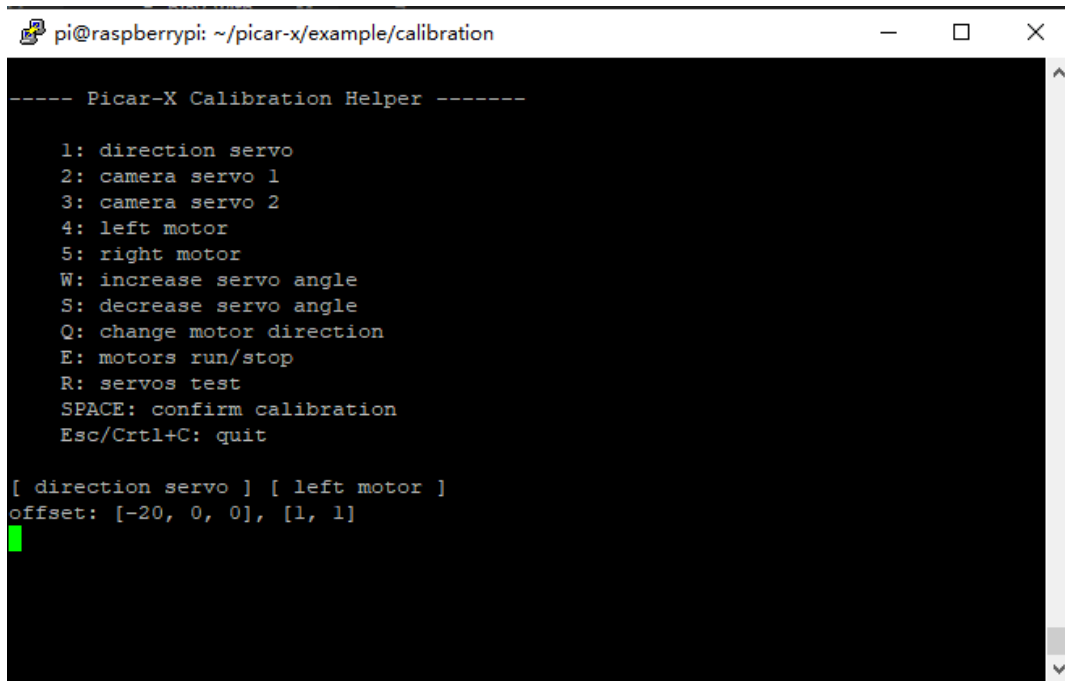
Some servo angles may be slightly tilted due to possible deviations during PiCar-X installation or limitations of the servos themselves, so you can calibrate them.

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 /home/pi/picar-x/example/calibration
sudo python3 calibration.py
```

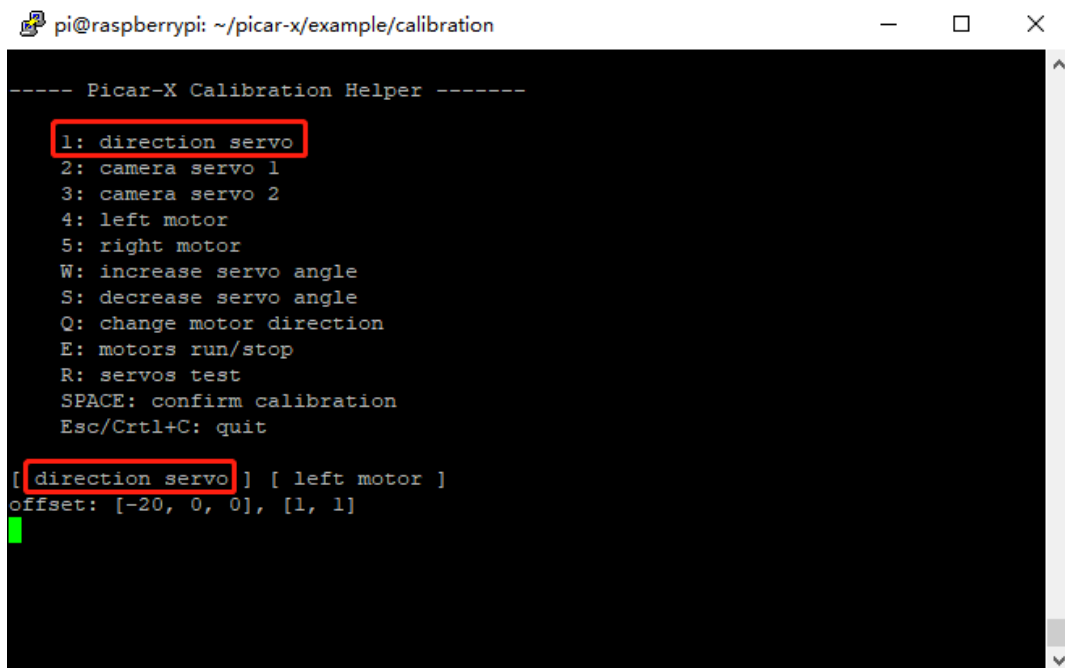
2. After running the code, you will see the following interface displayed in the terminal.



```
pi@raspberrypi: ~/picar-x/example/calibration
----- Picar-X Calibration Helper -----
1: direction servo
2: camera servo 1
3: camera servo 2
4: left motor
5: right motor
W: increase servo angle
S: decrease servo angle
Q: change motor direction
E: motors run/stop
R: servos test
SPACE: confirm calibration
Esc/Ctrl+C: quit

[ direction servo ] [ left motor ]
offset: [-20, 0, 0], [1, 1]
█
```

3. The R key is used to test whether the servo that controls the direction of the front wheel can work normally and is not damaged.
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.



```
pi@raspberrypi: ~/picar-x/example/calibration
----- Picar-X Calibration Helper -----
1: direction servo
2: camera servo 1
3: camera servo 2
4: left motor
5: right motor
W: increase servo angle
S: decrease servo angle
Q: change motor direction
E: motors run/stop
R: servos test
SPACE: confirm calibration
Esc/Ctrl+C: quit

[ direction servo ] [ left motor ]
offset: [-20, 0, 0], [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.

```

pi@raspberrypi: ~/picar-x/example/calibration
----- Picar-X Calibration Helper -----

1: direction servo
2: camera servo 1
3: camera servo 2
4: left motor
5: right motor
W: increase servo angle
S: decrease servo angle
Q: change motor direction
E: motors run/stop
R: servos test
SPACE: confirm calibration
Esc/Ctrl+C: quit

[ camera servo 1 ] [ left motor ]
offset: [-20, 20, 0], [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.

```

pi@raspberrypi: ~/picar-x/example/calibration
----- Picar-X Calibration Helper -----

1: direction servo
2: camera servo 1
3: camera servo 2
4: left motor
5: right motor
W: increase servo angle
S: decrease servo angle
Q: change motor direction
E: motors run/stop
R: servos test
SPACE: confirm calibration
Esc/Ctrl+C: quit

[ camera servo 2 ] [ left motor ]
offset: [-20, 20, 11], [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.

```
pi@raspberrypi: ~/picar-x/example/calibration
----- Picar-X Calibration Helper -----

1: direction servo
2: camera servo 1
3: camera servo 2
4: left motor
5: right motor
W: increase servo angle
S: decrease servo angle
Q: change motor direction
E: motors run/stop
R: servos test
SPACE: confirm calibration
Esc/Ctrl+C: quit

[ direction servo ] [ right motor ]
offset: [0, 20, 11], [1, 1]
```

- 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 `esc` to exit the program to complete the calibration.

```
pi@raspberrypi: ~/picar-x/example/calibration
----- Picar-X Calibration Helper -----

1: direction servo
2: camera servo 1
3: camera servo 2
4: left motor
5: right motor
W: increase servo angle
S: decrease servo angle
Q: change motor direction
E: motors run/stop
R: servos test
SPACE: confirm calibration
Esc/Ctrl+C: quit

[ direction servo ] [ left motor ]
offset: [0, 20, 11], [1, 1]
rConfirm save ?(y/n)
```

4.3 Let PiCar-X Move

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

Run the Code

```
cd /home/pi/picar-x/example
sudo python3 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)
        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)
```

(continues on next page)

(continued from previous page)

```
finally:  
    px.forward(0)
```

How it works?

The basic functionality of Parker 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.

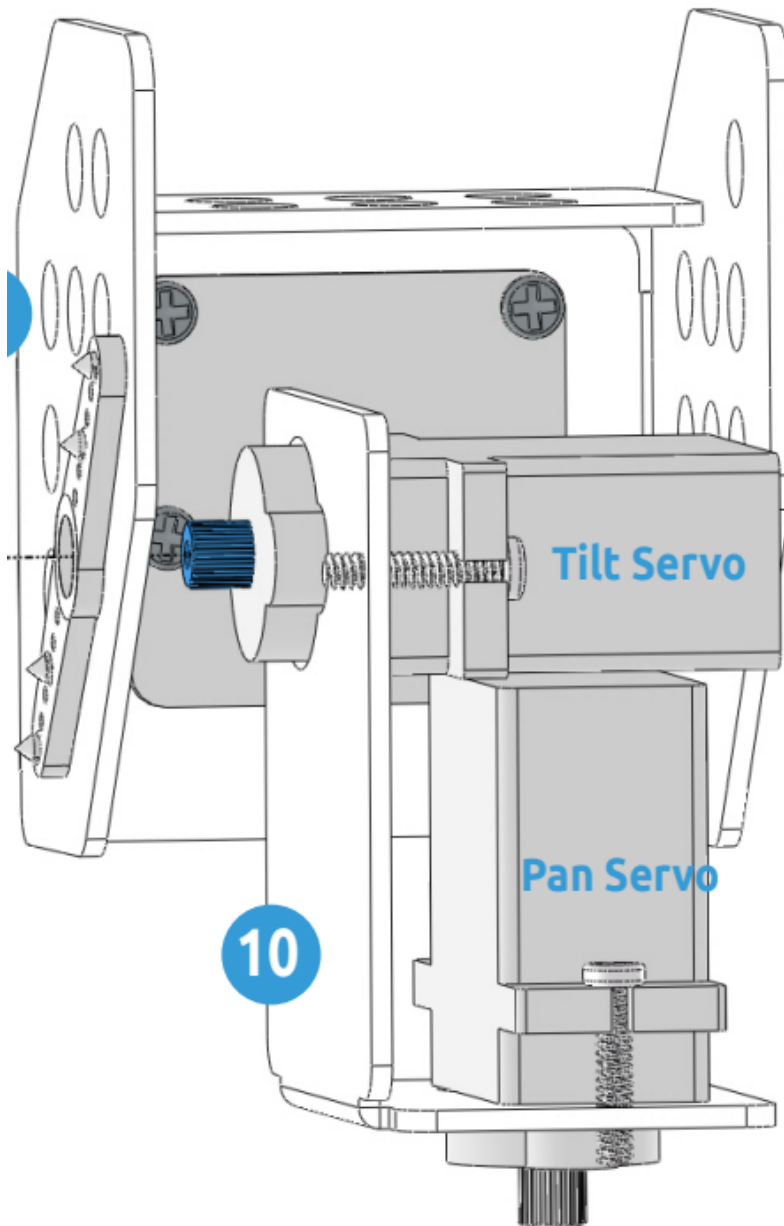
Now, the libraries to support the basic functionality of PiCar-X are imported. These lines will appear in all the examples that involve PiCar-X movement.

```
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_camera_servo1_angle`: Turns the Pan servo to a specific angle.
- `set_camera_servo2_angle`: Turns the Tilt servo to a specific angle.



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 /home/pi/picar-x/example
sudo python3 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 25cm, 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

def main():
    try:
        px = Picarx()
        # px = Picarx(ultrasonic_pins=['D2','D3']) # tring, echo
        px.forward(30)
        while True:
            distance = px.ultrasonic.read()
            print("distance: ",distance)
            if distance > 0 and distance < 300:
                if distance < 25:
                    px.set_dir_servo_angle(-35)
                else:
                    px.set_dir_servo_angle(0)
            finally:
                px.forward(0)

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

How it works?

The ultrasonic module is also imported in the `picarx` module, and we can use some of its encapsulated functions to detect distance.

```
from picarx import Picarx
```

Because the ultrasonic module is imported into the `picarx` module, we can directly use `px.ultrasonic.read()` to get the distance.

```
px = Picarx()
px.forward(30)
while True:
    distance = px.ultrasonic.read()
```

The following code snippet reads the distance value reported by the ultrasonic module, and if the distance is below 25cm (10 inches) it will set the steering servo from 0° (straight) to -35° (turn left).

```
while True:
    distance = px.ultrasonic.read()
    print("distance: ",distance)
    if distance > 0 and distance < 300:
        if distance < 25:
            px.set_dir_servo_angle(-35)
        else:
            px.set_dir_servo_angle(0)
```


4.5 Line Tracking

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

Run the Code

```
cd /home/pi/picar-x/example
sudo python3 minecart_plus.py
```

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

Code

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

```
from picarx import Picarx

if __name__ == '__main__':
    try:
        px = Picarx()
        # px = Picarx(grayscale_pins=['A0', 'A1', 'A2'])
        px_power = 10
        while True:
            gm_val_list = px.get_grayscale_data()
            print("gm_val_list:", gm_val_list)
            gm_status = px.get_line_status(gm_val_list)
            print("gm_status:", gm_status)

            if gm_status == 'forward':
                print(1)
                px.forward(px_power)

            elif gm_status == 'left':
                px.set_dir_servo_angle(12)
                px.forward(px_power)

            elif gm_status == 'right':
                px.set_dir_servo_angle(-12)
                px.forward(px_power)

            else:
                px.set_dir_servo_angle(0)
                px.stop()

    finally:
        px.stop()
```

How it works?

The grayscale sensor module `grayscale_module` is also imported in the `picarx` module, and we can use some of these methods to detect black lines.

The function to detect the black line 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_line_status()`: `get_line_status()`: This method will generate an action based on the values detected by the three probes. There are four types of actions: `forward`, `left`, `right`, and `stop`.

The trigger conditions for these actions are as follows: A value is assigned by default in the module as the threshold for detecting black or white. When the detection values of the three probes are all greater than the threshold, it means that the probes are sensing the color white, and no black line is detected, which makes the `get_line_status()` to generate a return value of `stop`.

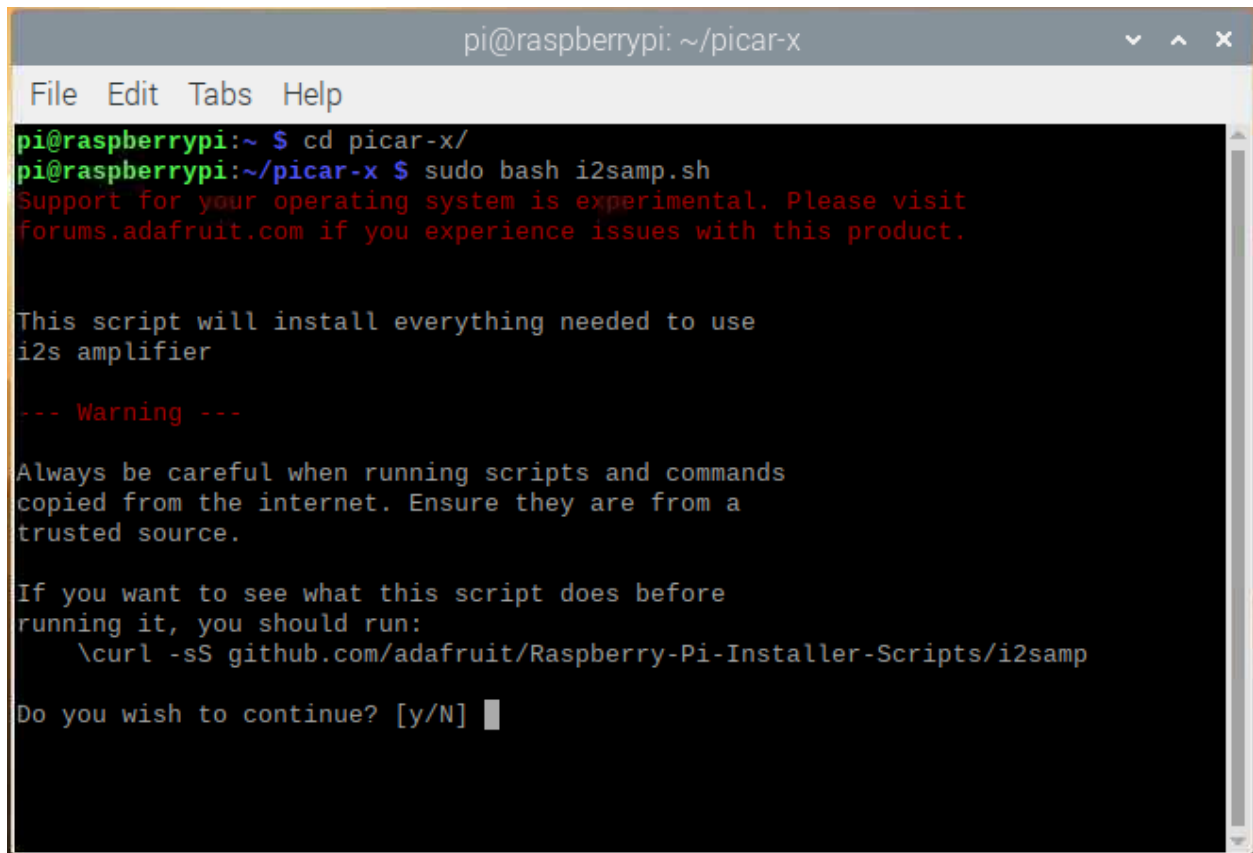
- If the right (and the first) probe detects a black line, `right` is returned;
- If the middle probe detects a black line, return `forward`;
- If the left probe detects a black line, `left` is returned.

4.6 Text to Speech

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

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

```
cd /home/pi/picar-x
sudo bash i2samp.sh
```



```
pi@raspberrypi: ~/picar-x
File Edit Tabs Help
pi@raspberrypi:~ $ cd picar-x/
pi@raspberrypi:~/picar-x $ 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] █
```

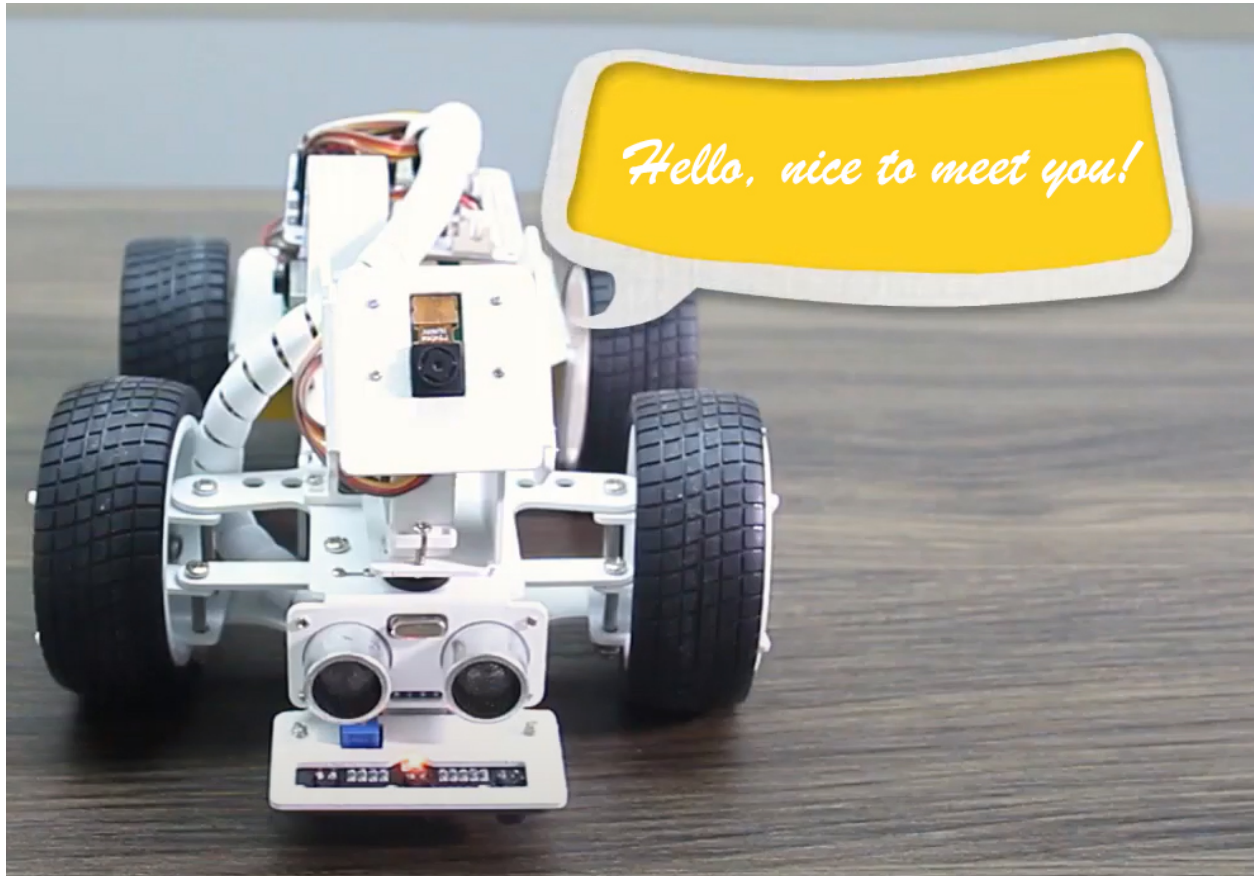
There will be several prompts asking to confirm the request. Respond to all prompts with a **Y**. After the changes have been made to the Raspberry Pi system, the computer will need to reboot for these changes to take effect.

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

Run the Code

```
cd /home/pi/picar-x/example
sudo python3 tts_example.py
```

After running the code, PiCar-X will say “Hello”, “Hi”, “Good bye”, “Nice to meet you”.



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 robot_hat import TTS

if __name__ == "__main__":
    words = ["Hello", "Hi", "Good bye", "Nice to meet you"]
    tts_robot = TTS()
    for i in words:
        print(i)
        tts_robot.say(i)
```

How it works?

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

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

```
from robot_hat import TTS
```

Create a string list `words`, then create an instantiated object of the `TTS()` class `tts_robot`, and finally use the `tts_robot.say()` function to speak the words in the list in speech.

```
words = ["Hello", "Hi", "Good bye", "Nice to meet you"]
tts_robot = TTS()
for i in words:
    print(i)
    tts_robot.say(i)
```

4.7 Computer Vision

This next project will officially enter the field of computer vision!

To perform the next four experiments, make sure to have completed the *Remote Desktop*. A remote connection via SSH will not display the camera images.

Run the Code

Note:

- This project requires access to the Raspberry Pi desktop to view the footage taken by the camera module.
- You can connect a screen to the PiCar-X or refer to the tutorial *Remote Desktop* to access it with VNC or XRDP.
- Once inside the Raspberry Pi desktop, open Terminal and type the following command to run it, or just open and run it with a Python editor.

```
cd /home/pi/picar-x/example
sudo python3 computer_vision.py
```

After the code is run, you will see a window open on your desktop showing the shot.

Code

```
import cv2
from picamera.array import PiRGBArray
from picamera import PiCamera
import time

with PiCamera() as camera:
    camera.resolution = (640, 480)
    camera.framerate = 24
    rawCapture = PiRGBArray(camera, size=camera.resolution)
    time.sleep(2)

    for frame in camera.capture_continuous(rawCapture, format="bgr", use_video_
↵port=True): # use_video_port=True
```

(continues on next page)

(continued from previous page)

```

img = frame.array
cv2.imshow("video", img) # OpenCV image show
rawCapture.truncate(0) # Release cache

k = cv2.waitKey(1) & 0xFF
if k == 27:
    break

print('quit ...')
cv2.destroyAllWindows()
camera.close()

```

How it works?

Photos are obtained with PiCamera. This package provides a pure Python interface to the Raspberry Pi camera.

- [PiCamera Docs](#)

Capturing an image to a file only requires specifying the name of the file to the output of whatever `capture()` method was required.

```

from time import sleep
from picamera import PiCamera

with PiCamera() as camera:
    camera.resolution = (640, 480)
    camera.start_preview()
    # Camera warm-up time
    sleep(2)
    camera.capture('foo.jpg')

```

This project uses the **capturing timelapse sequences** method. This method enables OpenCV to acquire sequential frames.

With this method, the camera captures images continually until it is told to stop. Images are automatically given unique names. The `sleep(x)` function controls the delay between captures.

```

from time import sleep
from picamera import PiCamera

with PiCamera() as camera:
    camera.resolution = (640, 480)
    camera.start_preview()
    sleep(2)

    for filename in camera.capture_continuous('img{counter:03d}.jpg'):
        print('Captured %s' % filename)
        sleep(10) # capture images with a 10s delay between each shot

```

In order to capture OpenCV objects, an image will be captured to Python's in-memory stream class: `BytesIO`. The `BytesIO` will convert the stream to a numpy array, and the program will read the array with OpenCV:

- [What is Numpy?](#)

```

import io
import time
import picamera
import cv2

```

(continues on next page)

(continued from previous page)

```

import numpy as np

# Create the in-memory stream
stream = io.BytesIO()
with picamera.PiCamera() as camera:
    camera.start_preview()
    time.sleep(2)
    camera.capture(stream, format='jpeg')
# Construct a numpy array from the stream
data = np.fromstring(stream.getvalue(), dtype=np.uint8)
# "Decode" the image from the array, preserving colour
image = cv2.imdecode(data, 1)
# OpenCV returns an array with data in BGR order. If you want RGB instead
# use the following...
image = image[:, :, ::-1]

```

To avoid the losses with JPEG encoding and decoding, use the classes in the `picamera.array` module. This will also potentially increase the speed of image processing.

As OpenCV images are simply numpy arrays arranged in BGR order, the `PiRGBArray` class, and simply capture with the 'bgr' format. Note: RGB data and BGR data are the same size and configuration, but have reversed color planes.

- `PiRGBArray`

```

import time
import picamera
import picamera.array
import cv2

with picamera.PiCamera() as camera:
    camera.start_preview()
    time.sleep(2)
    with picamera.array.PiRGBArray(camera) as stream:
        camera.capture(stream, format='bgr')
        # At this point the image is available as stream.array
        image = stream.array

```

Combined with the method of capturing timelapse sequences, these 3-dimensional RGB arrays are shown by OpenCV.

```

import cv2
from picamera.array import PiRGBArray
from picamera import PiCamera

#init camera
with PiCamera() as camera:
    camera.resolution = (640,480)
    camera.framerate = 24
    rawCapture = PiRGBArray(camera, size=camera.resolution)

    for frame in camera.capture_continuous(rawCapture, format="bgr", use_video_
↳port=True): # use_video_port=True
        img = frame.array
        cv2.imshow("video", img) # OpenCV image show
        rawCapture.truncate(0) # Release cache

        # click ESC key to exit.

```

(continues on next page)

(continued from previous page)

```
k = cv2.waitKey(1) & 0xFF
if k == 27:
    camera.close()
    break
```

There are many other ways to read video streams with OpenCV. The ones used in these examples are better suited for the next four PiCar-X tasks, such as *Color Detection* and *Face Detection*.

For more ways to use video streams, please reference: [OpenCV-Python Tutorials](#).

4.8 Color Detection

This project will add a color detection algorithm to the previous *Computer Vision* project.

- [PDF] [Color Cards](#)

Note: The printed colors may have a slightly different hue from the Python 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.



Run the Code

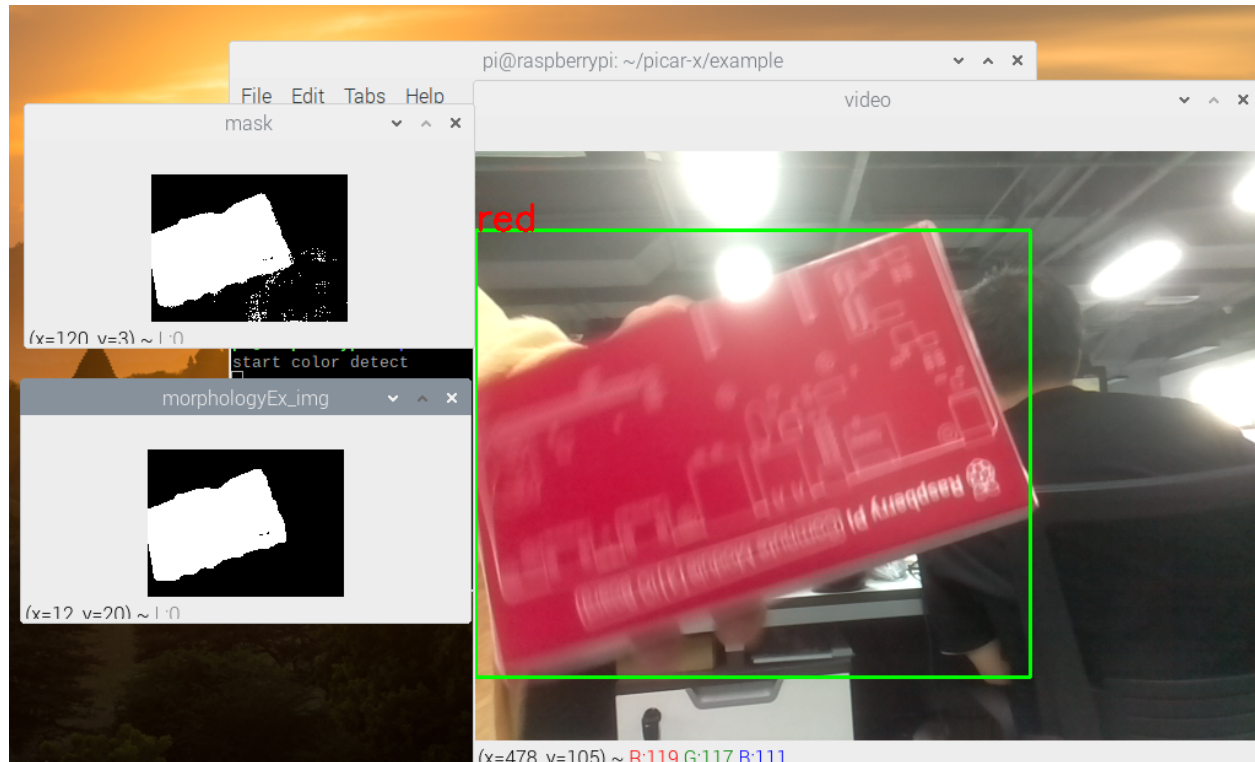
Note:

- This project requires access to the Raspberry Pi desktop to view the footage taken by the camera module.
- You can connect a screen to the PiCar-X or refer to the tutorial *Remote Desktop* to access it with VNC or XRDP.

- Once inside the Raspberry Pi desktop, open Terminal and type the following command to run it, or just open and run it with a Python editor.

```
cd /home/pi/picar-x/example
sudo python3 color_detect.py
```

When the code is run, if PiCar-X captures a red object, it will frame it out. You can also change the 'red' in the code to another color for detection.



Code

```
import cv2
from picamera.array import PiRGBArray
from picamera import PiCamera
import numpy as np
import time

color_dict = {'red':[0,4], 'orange':[5,18], 'yellow':[22,37], 'green':[42,85], 'blue':[92,
↪110], 'purple':[115,165], 'red_2':[165,180]} #Here is the range of H in the HSV
↪color space represented by the color

kernel_5 = np.ones((5,5),np.uint8) #Define a 5x5 convolution kernel with element_
↪values of all 1.

def color_detect(img,color_name):

    # The blue range will be different under different lighting conditions and can be_
    ↪adjusted flexibly. H: chroma, S: saturation v: lightness
    resize_img = cv2.resize(img, (160,120), interpolation=cv2.INTER_LINEAR) # In_
    ↪order to reduce the amount of calculation, the size of the picture is reduced to_
    ↪(160,120)
```

(continues on next page)

(continued from previous page)

```

hsv = cv2.cvtColor(resize_img, cv2.COLOR_BGR2HSV) # Convert from BGR
↳to HSV
color_type = color_name

mask = cv2.inRange(hsv,np.array([min(color_dict[color_type]), 60, 60]), np.
↳array([max(color_dict[color_type]), 255, 255]) ) # inRange()Make the ones
↳between lower/upper white, and the rest black
    if color_type == 'red':
        mask_2 = cv2.inRange(hsv, (color_dict['red_2'][0],0,0), (color_dict['red_2
↳'] [1],255,255))
        mask = cv2.bitwise_or(mask, mask_2)

morphologyEx_img = cv2.morphologyEx(mask, cv2.MORPH_OPEN, kernel_5,iterations=1)
↳ # Perform an open operation on the image

# Find the contour in morphologyEx_img, and the contours are arranged according
↳to the area from small to large.
_tuple = cv2.findContours(morphologyEx_img,cv2.RETR_EXTERNAL,cv2.CHAIN_APPROX_
↳SIMPLE)
# compatible with opencv3.x and opencv4.x
if len(_tuple) == 3:
    _, contours, hierarchy = _tuple
else:
    contours, hierarchy = _tuple

color_area_num = len(contours) # Count the number of contours

if color_area_num > 0:
    for i in contours: # Traverse all contours
        x,y,w,h = cv2.boundingRect(i) # Decompose the contour into the
↳coordinates of the upper left corner and the width and height of the recognition
↳object

        # Draw a rectangle on the image (picture, upper left corner coordinate,
↳lower right corner coordinate, color, line width)
        if w >= 8 and h >= 8: # Because the picture is reduced to a quarter of
↳the original size, if you want to draw a rectangle on the original picture to
↳circle the target, you have to multiply x, y, w, h by 4.
            x = x * 4
            y = y * 4
            w = w * 4
            h = h * 4
            cv2.rectangle(img, (x,y), (x+w,y+h), (0,255,0),2) # Draw a rectangular
↳frame
            cv2.putText(img,color_type, (x,y), cv2.FONT_HERSHEY_SIMPLEX, 1, (0,0,
↳255),2) # Add character description

    return img,mask,morphologyEx_img

with PiCamera() as camera:
    print("start color detect")
    camera.resolution = (640,480)
    camera.framerate = 24
    rawCapture = PiRGBArray(camera, size=camera.resolution)
    time.sleep(2)

    for frame in camera.capture_continuous(rawCapture, format="bgr",use_video_
↳port=True): # use_video_port=True

```

(continues on next page)

(continued from previous page)

```

img = frame.array
img,img_2,img_3 = color_detect(img,'red') # Color detection function
cv2.imshow("video", img) # OpenCV image show
cv2.imshow("mask", img_2) # OpenCV image show
cv2.imshow("morphologyEx_img", img_3) # OpenCV image show
rawCapture.truncate(0) # Release cache

k = cv2.waitKey(1) & 0xFF
# 27 is the ESC key, which means that if you press the ESC key to exit
if k == 27:
    break

print('quit ...')
cv2.destroyAllWindows()
camera.close()

```

How it works?

First, the range of H in the HSV color space is defined as a dictionary, which is convenient for the following color judgment algorithm:

```

color_dict = {'red':[0,4], 'orange':[5,18], 'yellow':[22,37], 'green':[42,85], 'blue':[92,
↪110], 'purple':[115,165], 'red_2':[165,180]}

```

Then, a convolution kernel of size 5x5 is defined, which will be used for morphological operations, like filtering.

```

kernel_5 = np.ones((5,5),np.uint8)

```

Next, the `color_detect()` function will process pictures in four steps:

1. Extract the data of the target color as a new binary image (array).
2. Performs advanced morphological transformations.
3. Finds contours in a binary image.
4. Draws a frame for the recognized object on the image.

```

def color_detect(img,color_name):

    # The blue range will be different under different lighting conditions and can be
    ↪adjusted flexibly. H: chroma, S: saturation v: lightness
    resize_img = cv2.resize(img, (160,120), interpolation=cv2.INTER_LINEAR) # In
    ↪order to reduce the amount of calculation, the size of the picture is reduced to
    ↪(160,120)
    hsv = cv2.cvtColor(resize_img, cv2.COLOR_BGR2HSV) # Convert from BGR
    ↪to HSV
    color_type = color_name

    mask = cv2.inRange(hsv,np.array([min(color_dict[color_type]), 60, 60]), np.
    ↪array([max(color_dict[color_type]), 255, 255]) ) # inRange()Make the ones
    ↪between lower/upper white, and the rest black
    if color_type == 'red':
        mask_2 = cv2.inRange(hsv, (color_dict['red_2'][0],0,0), (color_dict['red_2
    ↪'] [1],255,255))
        mask = cv2.bitwise_or(mask, mask_2)

    morphologyEx_img = cv2.morphologyEx(mask, cv2.MORPH_OPEN, kernel_5,iterations=1)
    ↪ # Perform an open operation on the image

```

(continues on next page)

(continued from previous page)

```

# Find the contour in morphologyEx_img, and the contours are arranged according_
↳to the area from small to large.
_tuple = cv2.findContours(morphologyEx_img, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_
↳SIMPLE)
# compatible with opencv3.x and opencv4.x
if len(_tuple) == 3:
    _, contours, hierarchy = _tuple
else:
    contours, hierarchy = _tuple

color_area_num = len(contours) # Count the number of contours

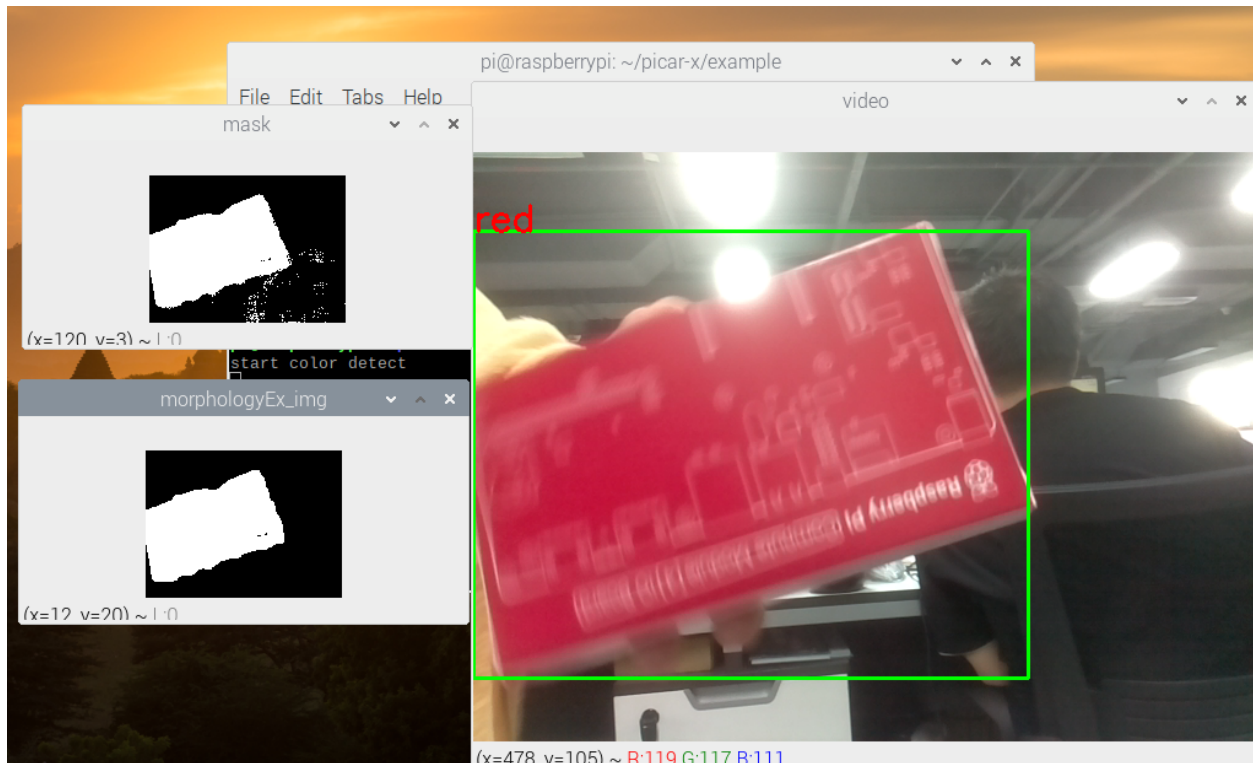
if color_area_num > 0:
    for i in contours: # Traverse all contours
        x,y,w,h = cv2.boundingRect(i) # Decompose the contour into the_
↳coordinates of the upper left corner and the width and height of the recognition_
↳object

        # Draw a rectangle on the image (picture, upper left corner coordinate,_
↳lower right corner coordinate, color, line width)
        if w >= 8 and h >= 8: # Because the picture is reduced to a quarter of_
↳the original size, if you want to draw a rectangle on the original picture to_
↳circle the target, you have to multiply x, y, w, h by 4.
            x = x * 4
            y = y * 4
            w = w * 4
            h = h * 4
            cv2.rectangle(img, (x,y), (x+w,y+h), (0,255,0), 2) # Draw a rectangular_
↳frame
            cv2.putText(img,color_type, (x,y), cv2.FONT_HERSHEY_SIMPLEX, 1, (0,0,
↳255), 2) # Add character description

return img,mask,morphologyEx_img

```

The `img`, `mask`, and `morphologyEx_img` are displayed in three windows to directly observe the processing results of each step.



For more information on morphology and contouring, please reference the following resources:

- [Opening operation - Wikipedia](#)
- [morphologyEx - OpenCV](#)
- [findContours - OpenCV](#)
- [Contour Features - OpenCV](#)

4.9 Face Detection

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

Note: To run this project, you must first complete *Remote Desktop*.

Run the Code

Note:

- This project requires access to the Raspberry Pi desktop to view the footage taken by the camera module.
- You can connect a screen to the PiCar-X or refer to the tutorial *Remote Desktop* to access it with VNC or XRDP.
- Once inside the Raspberry Pi desktop, open Terminal and type the following command to run it, or just open and run it with a Python editor.

```
cd /home/pi/picar-x/example
sudo python3 human_face_detect.py
```

After the code is run, the face will be checked out in the screen.

Code

```
import cv2
from picamera.array import PiRGBArray
from picamera import PiCamera
import time

def human_face_detect(img):
    resize_img = cv2.resize(img, (320,240), interpolation=cv2.INTER_LINEAR) #
    ↪ In order to reduce the amount of calculation, resize the image to 320 x 240 size
    gray = cv2.cvtColor(resize_img, cv2.COLOR_BGR2GRAY) # Convert to grayscale
    faces = face_cascade.detectMultiScale(gray, 1.3, 2) # Detect faces on
    ↪ grayscale images
    face_num = len(faces) # Number of detected faces
    if face_num > 0:
        for (x,y,w,h) in faces:
            x = x*2 # Because the image is reduced to one-half of the original size,
            ↪ the x, y, w, and h must be multiplied by 2.
            y = y*2
            w = w*2
            h = h*2
            cv2.rectangle(img, (x,y), (x+w,y+h), (255,0,0), 2) # Draw a rectangle on the
            ↪ face

    return img

with PiCamera() as camera:
    print("start human face detect")
    camera.resolution = (640,480)
    camera.framerate = 24
    rawCapture = PiRGBArray(camera, size=camera.resolution)
    time.sleep(2)

    for frame in camera.capture_continuous(rawCapture, format="bgr", use_video_
    ↪ port=True): # use_video_port=True
        img = frame.array
        img = human_face_detect(img)
        cv2.imshow("video", img) #OpenCV image show
        rawCapture.truncate(0) # Release cache

        k = cv2.waitKey(1) & 0xFF
        # 27 is the ESC key, which means that if you press the ESC key to exit
        if k == 27:
            break

    print('quit ...')
    cv2.destroyAllWindows()
    camera.close()
```

How it works?

In the same path as this project (picar-x/example/), put a file haarcascade_frontalhumanface_default.xml. This file is a face detection model file trained in OpenCV.

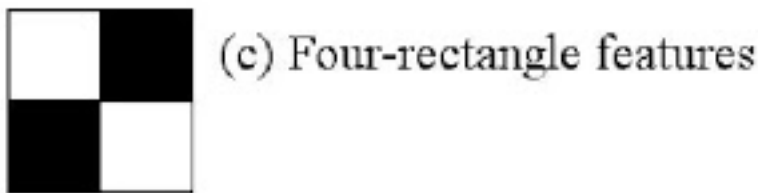
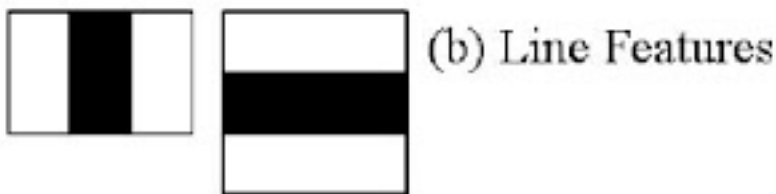
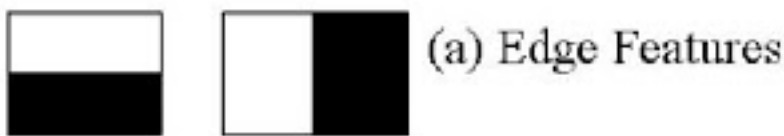
This file is called by **Cascade Classifier** of OpenCV.

```
face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
```

Object Detection using Haar feature-based cascade classifiers is an effective object detection method proposed by Paul Viola and Michael Jones in their paper, “Rapid Object Detection using a Boosted Cascade of Simple Features” in 2001.

This is a machine learning based approach, where a cascade function is trained from a large quantity of positive and negative images, and then used to detect objects in other images.

When working with face detection, the algorithm will initially need a large quantity of positive images (images of faces) and negative images (images without faces) to train the classifier. From there, the facial features will then need to be extracted. For this, Haar features shown in the below image are used, similar to the convolutional kernel. Each feature is a single value obtained by subtracting the sum of pixels under the white rectangle, from the sum of pixels under the black rectangle.



- Cascade Classifier
- Cascade Classifier Training

The `human_face_detect()` function processes pictures in three steps:

1. Convert picture to grayscale.
2. Detect the human face on the grayscale image to obtain the bounding rectangle of the detected face.
3. Draws a frame for the recognized object on the image.

```
def human_face_detect(img):
    resize_img = cv2.resize(img, (320,240), interpolation=cv2.INTER_LINEAR) # To_
    ↪ reduce the amount of calculation, the image size is reduced.
    gray = cv2.cvtColor(resize_img, cv2.COLOR_BGR2GRAY) # Convert picture to_
    ↪ grayscale.
    faces = face_cascade.detectMultiScale(gray, 1.3, 2) # Obtain the bounding_
    ↪ rectangle of the detected face.
```

(continues on next page)

(continued from previous page)

```

face_num = len(faces)
max_area = 0
if face_num > 0:
    for (x,y,w,h) in faces: # Because the picture is reduced during operation,
↪the increase now go back.
        x = x*2
        y = y*2
        w = w*2
        h = h*2
        cv2.rectangle(img, (x,y), (x+w,y+h), (255,0,0),2) # Draw a frame for the
↪recognized object on the image.

return img

```

- detectMultiScale - OpenCV

4.10 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

Note:

- This project requires access to the Raspberry Pi desktop to view the footage taken by the camera module.
- You can connect a screen to the PiCar-X or refer to the tutorial [Remote Desktop](#) to access it with VNC or XRDP.
- Once inside the Raspberry Pi desktop, open Terminal and type the following command to run it, or just open and run it with a Python editor.

```

cd /home/pi/picar-x/example
sudo python3 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
- ESC / Ctrl+C: quit

code

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

print('Please run under desktop environment (eg: vnc) to display the image window')

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

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
ESC / Ctrl+C: quit
'''

px = Picarx()

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

def move(operate:str, speed):

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

def main():
    speed = 0
```

(continues on next page)

(continued from previous page)

```

status = 'stop'

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

while True:
    print("\rstatus: %s , speed: %s" % (status, speed), end='', flush=True)
    # readkey
    key = readchar.readkey().lower()
    # operation
    if key in ('wsadfop'):
        # throttle
        if key == 'o':
            if speed <=90:
                speed += 10
        elif key == 'p':
            if speed >=10:
                speed -= 10
            if speed == 0:
                status = 'stop'
        # direction
        elif key in ('wsad'):
            if speed == 0:
                speed = 10
            if key == 'w':
                # Speed limit when reversing,avoid instantaneous current too large
                if status != 'forward' and speed > 60:
                    speed = 60
                status = 'forward'
            elif key == 'a':
                status = 'turn left'
            elif key == 's':
                if status != 'backward' and speed > 60: # Speed limit when
↳reversing
                    speed = 60
                status = 'backward'
            elif key == 'd':
                status = 'turn right'

        # stop
        elif key == 'f':
            status = 'stop'
        # move
        move(status, speed)
    # take photo
    elif key == 't':
        take_photo()
    # quit
    elif key == readchar.key.CTRL_C or key in readchar.key.ESCAPE_SEQUENCES:
        print('\nquit ...')
        px.stop()
        Vilib.camera_close()
        break

    sleep(0.1)

```

(continues on next page)

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

4.11 Controlled by the APP

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

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

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

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

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

How to do?

1. Install the sunfounder-controller module.

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

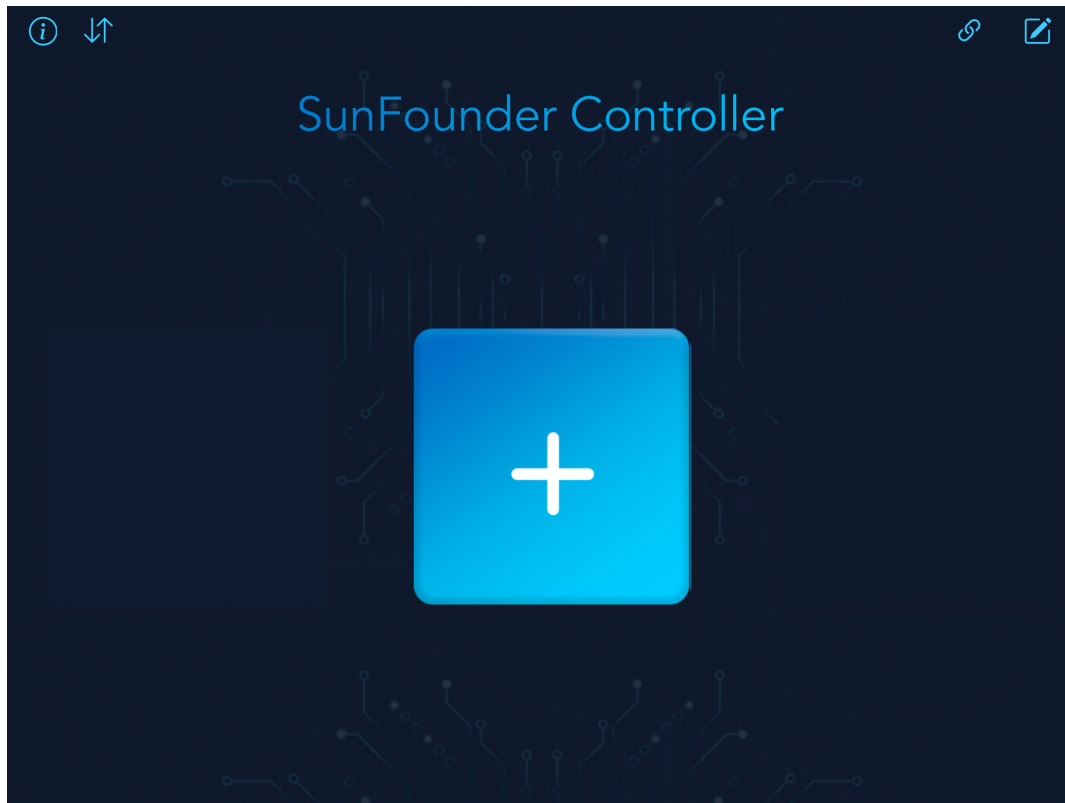
```
cd ~
git clone https://github.com/sunfounder/sunfounder-controller.git
cd ~/sunfounder-controller
sudo python3 setup.py install
```

2. Run the code.

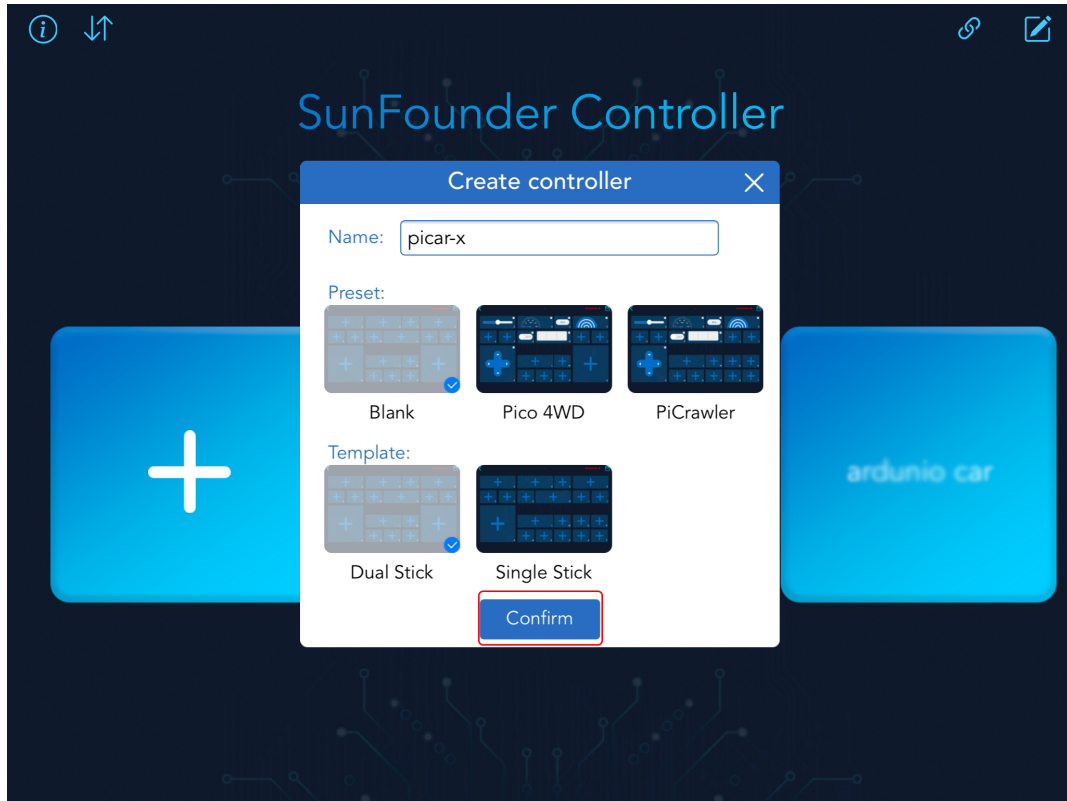
```
cd ~/sunfounder-controller/examples
sudo python3 picarx_control.py
```

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.

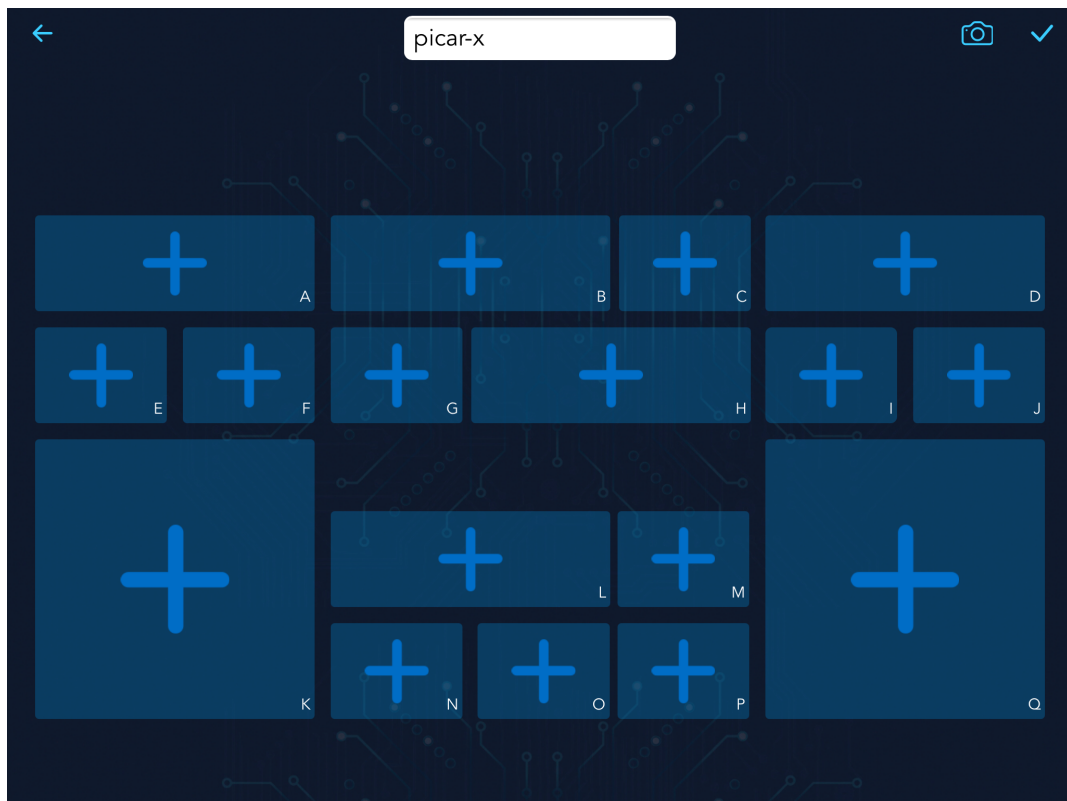


Give it a name and select the Controller type. There are preset controllers for some products in the Preset section, which you can use as needed. You can also customize your own controller by following the steps below.



5. Add different widgets to this controller.

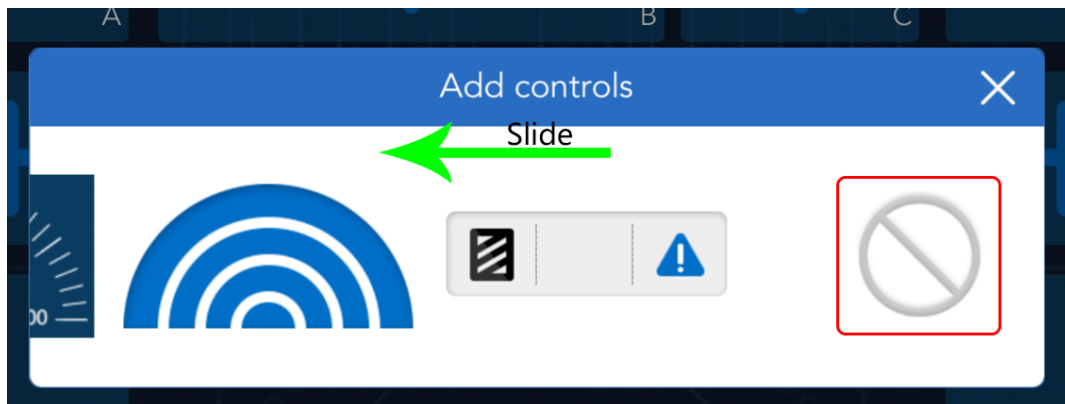
You can add different types and shapes of widgets to the **A-Q** 17 small areas inside this controller.



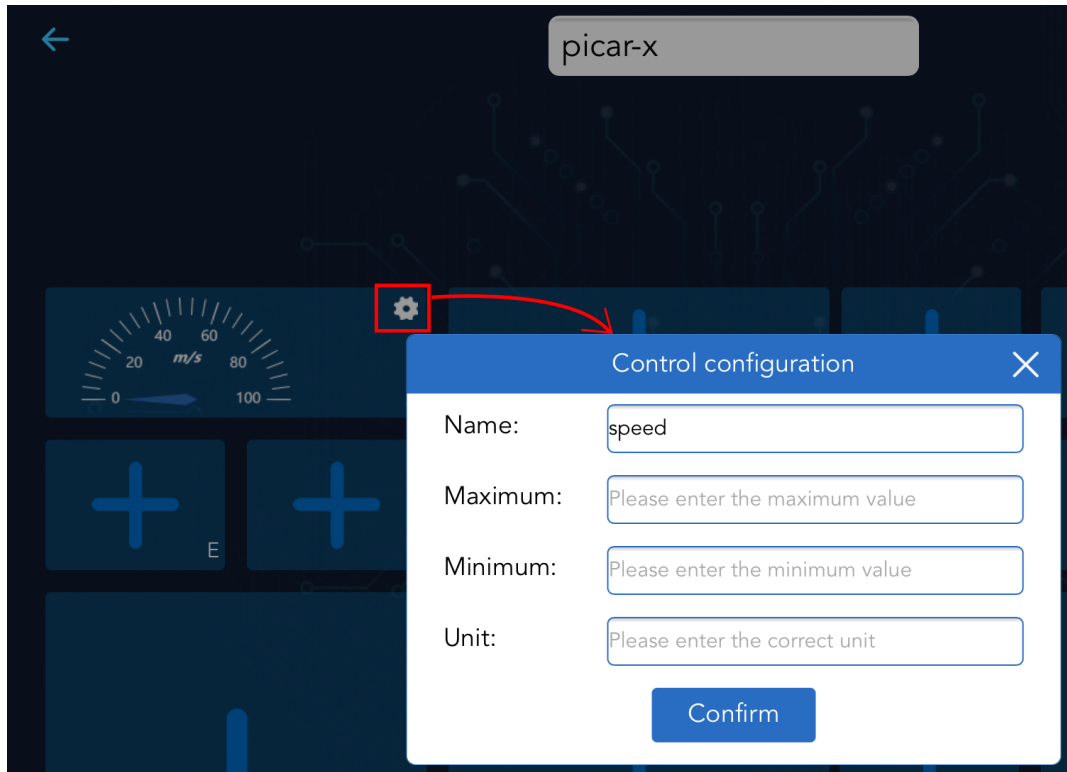
In the **A** area, add an ****Speedometer**** widget to display the car's speed.



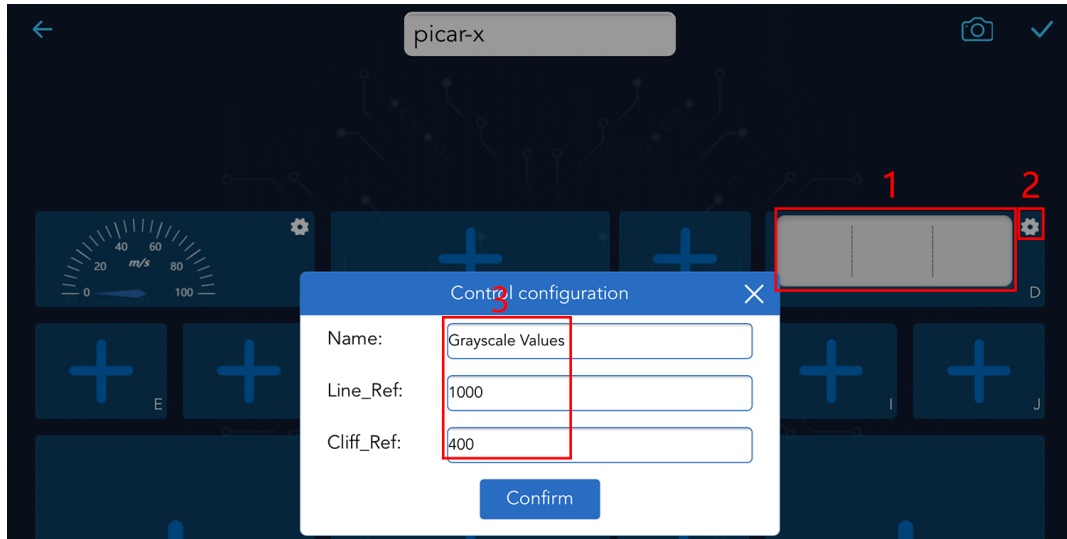
Note: You can delete the widget you have selected by clicking on it, swiping left to find the **Delete** button, and clicking on it.



Set the name, maximum and minimum values and units by clicking the **Settings** icon in the upper right corner.



Set your current environment's `Line_Ref` and `Cliff_Ref` for the **Grayscale Detection** widget in the **D** area.

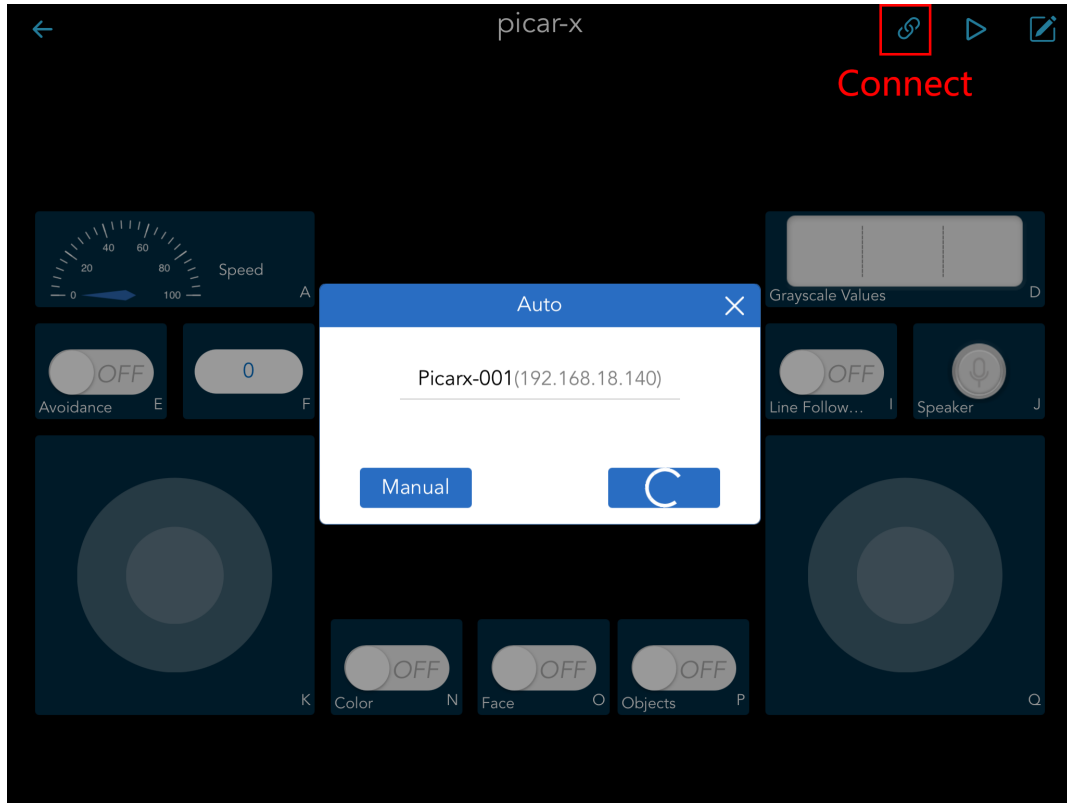


Lastly, add the remaining widgets and click the top right button to save.

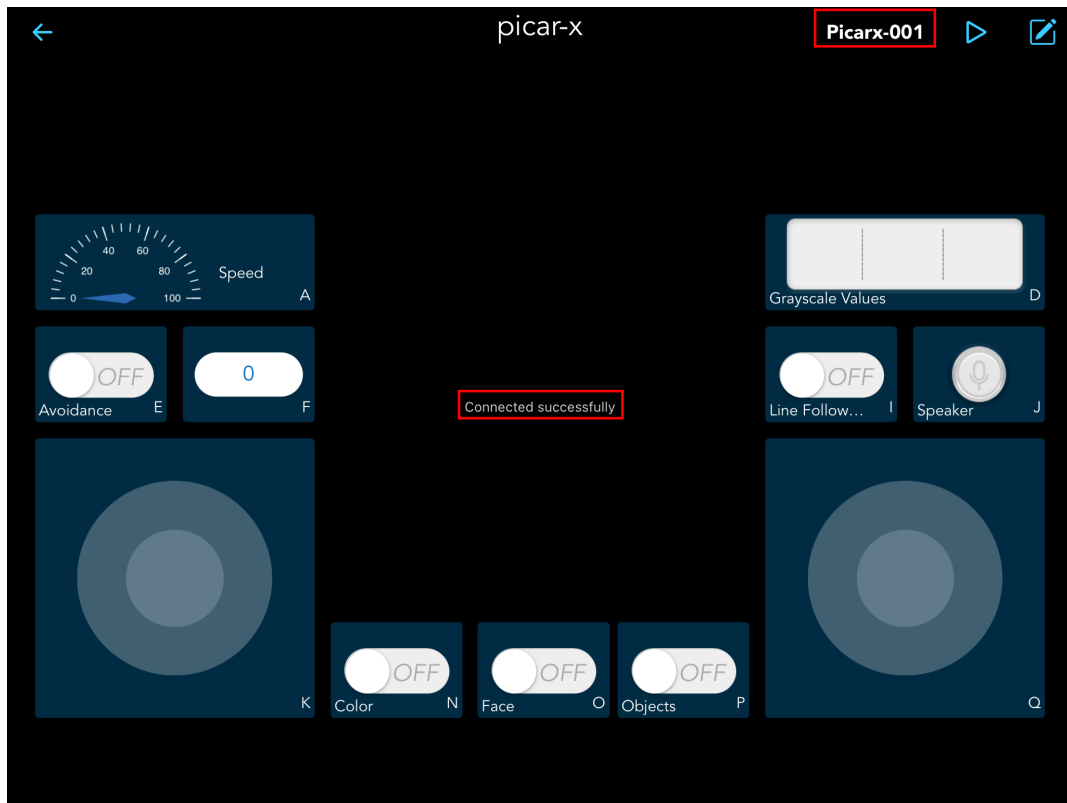


6. Connect to PiCar-x.

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.

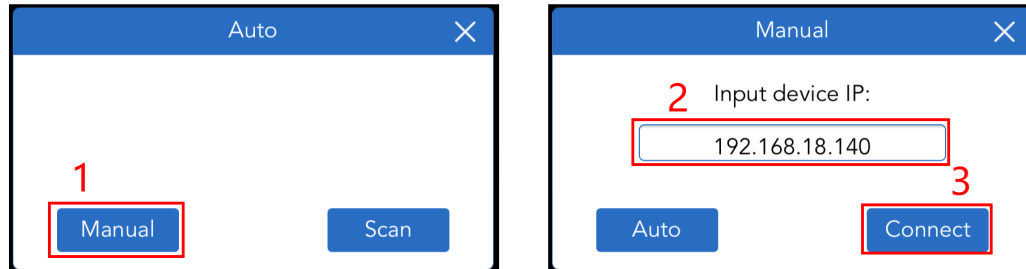


Once you click on the product name, the message “Connected Successfully” will appear and the product name will appear in the upper right corner.



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.



7. 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.
- **D:** Show the data of the three sensors on the grayscale module, which have three states: **black**: black line detected; **white**: white detected; **exclamation point**: cliff detected.
- **E:** turn on the obstacle avoidance function.

- **I**: turn on the line following function.
- **J**: voice recognition, press and hold this widget to start speaking, and it will show the recognized voice when you release it. We have set `forward`, `backard`, `left` and `right` 4 commands in the code to control the car.
- **K**: Control forward, backward, left, and right motions of the car.
- **Q**: turn the head(Camera) up, down, left and right.
- **N**: Turn on the color recognition function.
- **O**: Turn on the face recognition function.
- **P**: Turn on the object recognition function, it can recognize nearly 90 kinds of objects, for the list of models, please refer to: https://github.com/sunfounder/vilib/blob/master/workspace/coco_labels.txt.

PLAY WITH EZBLOCK

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

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

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

5.1 Quick Guide on EzBlock

There are 2 parts here:

- *Servo Adjust* allows you to keep all the servos at 0 degrees to complete a proper and safe assembly (otherwise you will probably damage the servos).
- *Install and Configure EzBlock Studio* will guide you to download EzBlock Studio to play with your robot.

5.1.1 Servo Adjust

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.

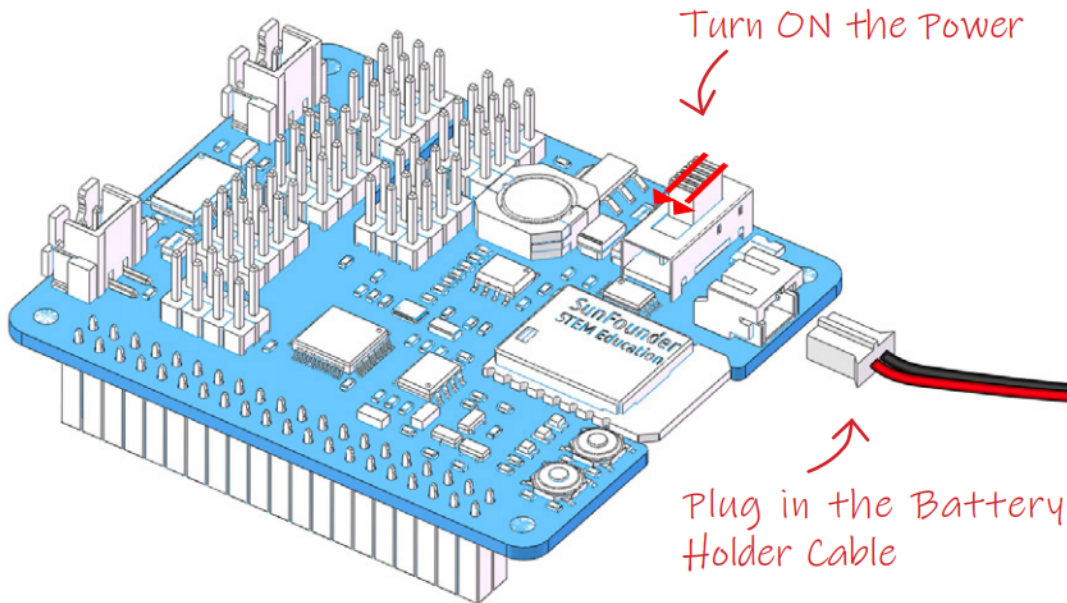
It is recommended that you follow the instructions on the APP to calibrate the picar-x after assembling it. The servo angle will be tilted due to possible deviations during assembly or limitations of the servo itself, so you can get the servo to a perfect state by calibrating it, usually the calibration angle is -5~5°.

But if the deviation angle is too big, you still have to go back to this section to set the servo angle to 0°, and then follow the instructions to reassemble the car.

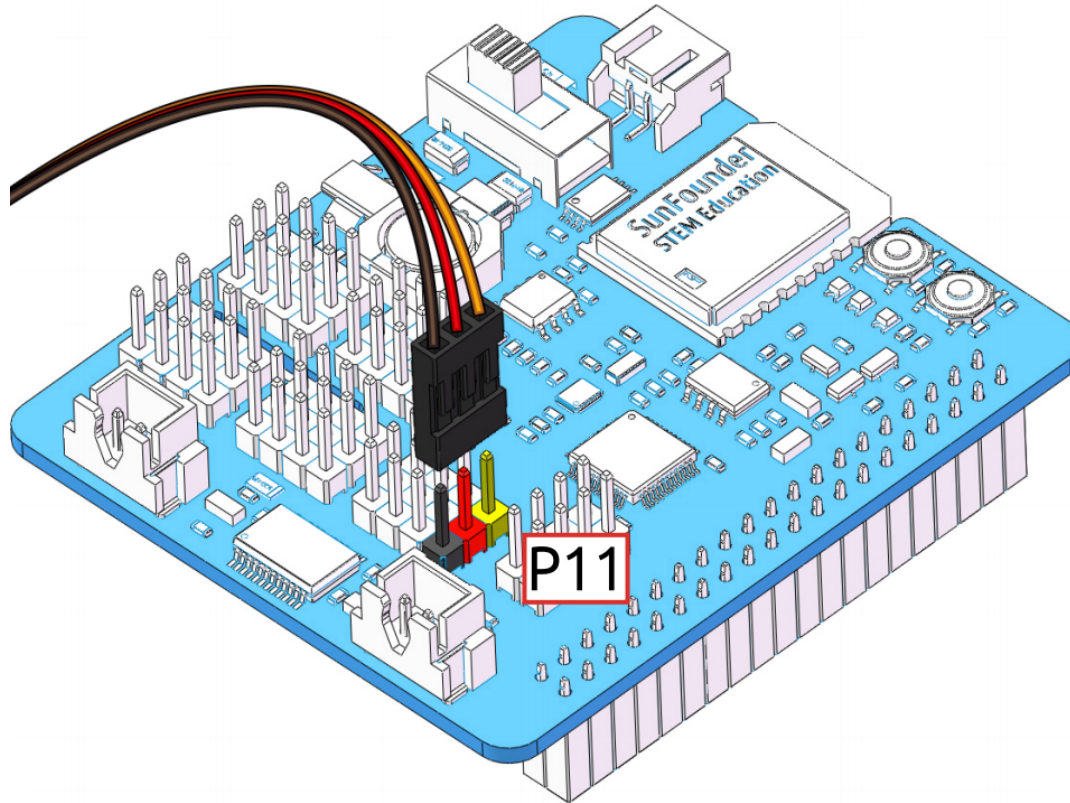
1. Firstly, [Install EzBlock OS\(3.1\)](#) onto a Micro SD card, once the installation is complete, insert it into the Raspberry Pi.
2. To ensure that the servo has been properly set to 0°, first insert the rocker arm into the servo shaft and then gently rotate the rocker arm to a different angle.



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



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



5. At this point you will see the servo arm rotate to a specific position (0°). If the servo arm does not return to 0° , press the RST button to restart the Robot HAT.
6. Now you can continue the installation as instructed on the assembly foldout.

Note:

- Do not unplug this servo cable before fastening this servo with the servo screw, you can unplug it after fastening.
 - Do not turn the servo while it is powered on to avoid damage; if the servo shaft is inserted at the wrong angle, pull out the servo and reinsert it.
 - Before assembling each servo, you need to plug the servo cable into P11 and turn on the power to set its angle to 0° .
 - This zeroing function will be disabled if you download a program to the robot later with the EzBlock APP.
-

5.1.2 Install and Configure EzBlock Studio

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

- **Install EzBlock Studio(3.1):** Download and install EzBlock Studio on your device or use the web-based version.
- **Connect the Product and EzBlock(3.1):** Configure Wi-Fi, Bluetooth and calibrate before use.
- **Open and Run Examples(3.1):** 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.

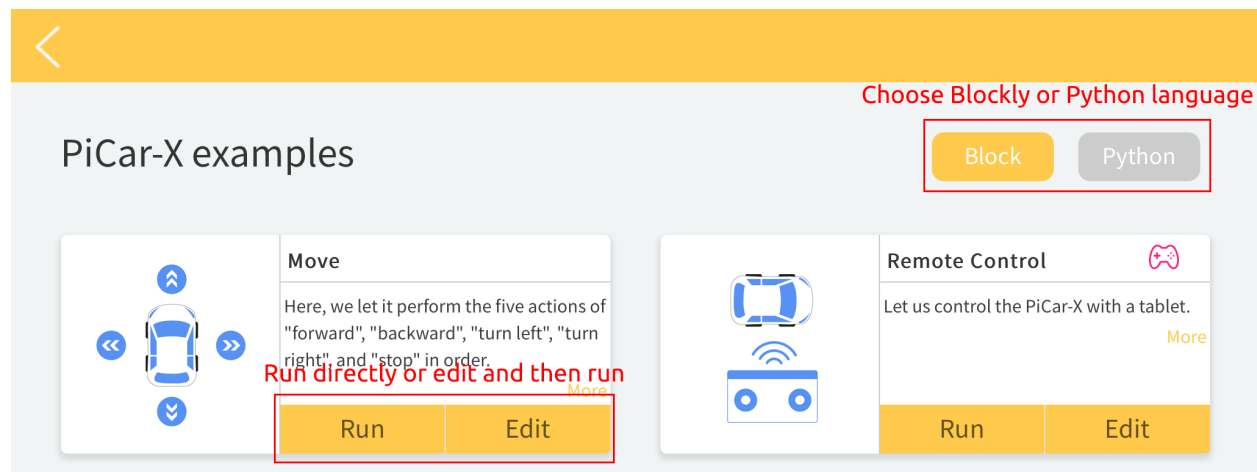
But if you think the assembly is perfect and no calibration is needed, you can also skip this step.

Projects

This section begins with basic programming functions for the PiCar-X, and continues through to creating more advanced programs in Ezbloc 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 Ezbloc projects have been uploaded to Ezbloc Studio's Examples page. From the Examples page, users can run the programs directly, or edit the examples and save them into the users My Projects folder.

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



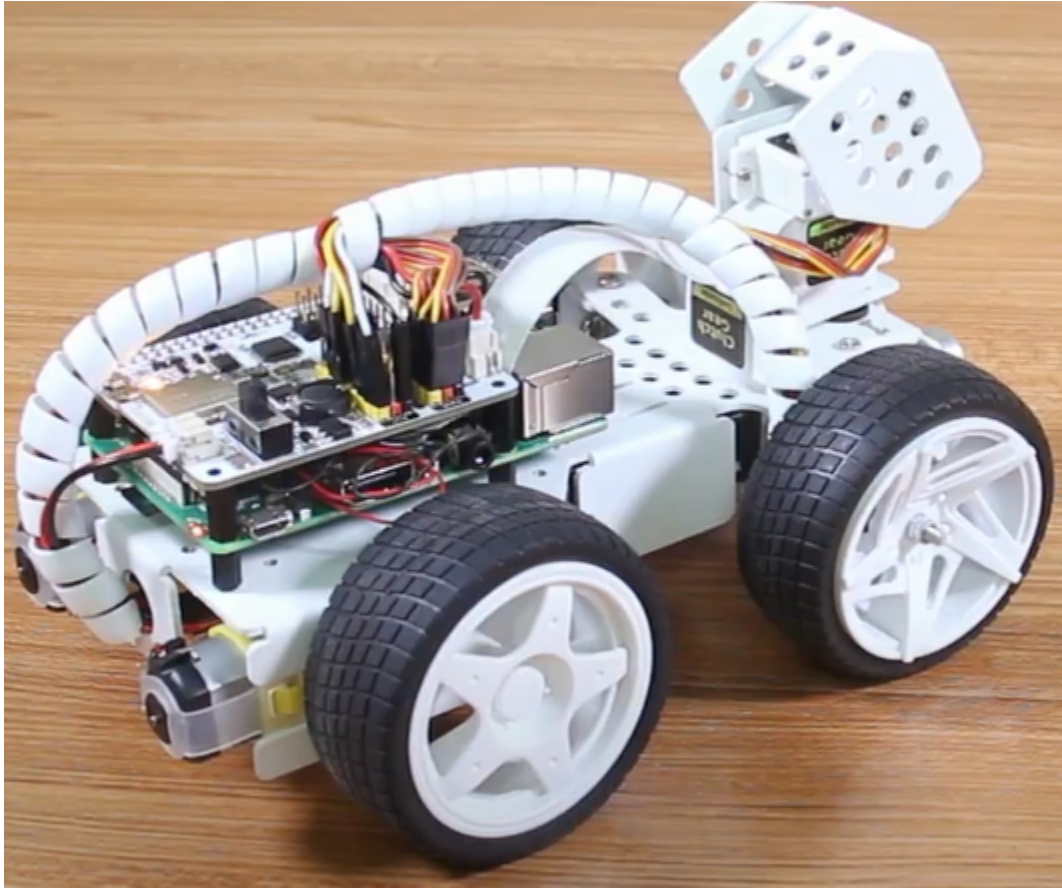
Basic

5.2 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 Ezbloc Studio, please read through the following two sections:

- [How to Create a New Project?](#)



TIPS

forward at a speed of 50

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.

backward at a speed of 50

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

turn steering angle to 0

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.

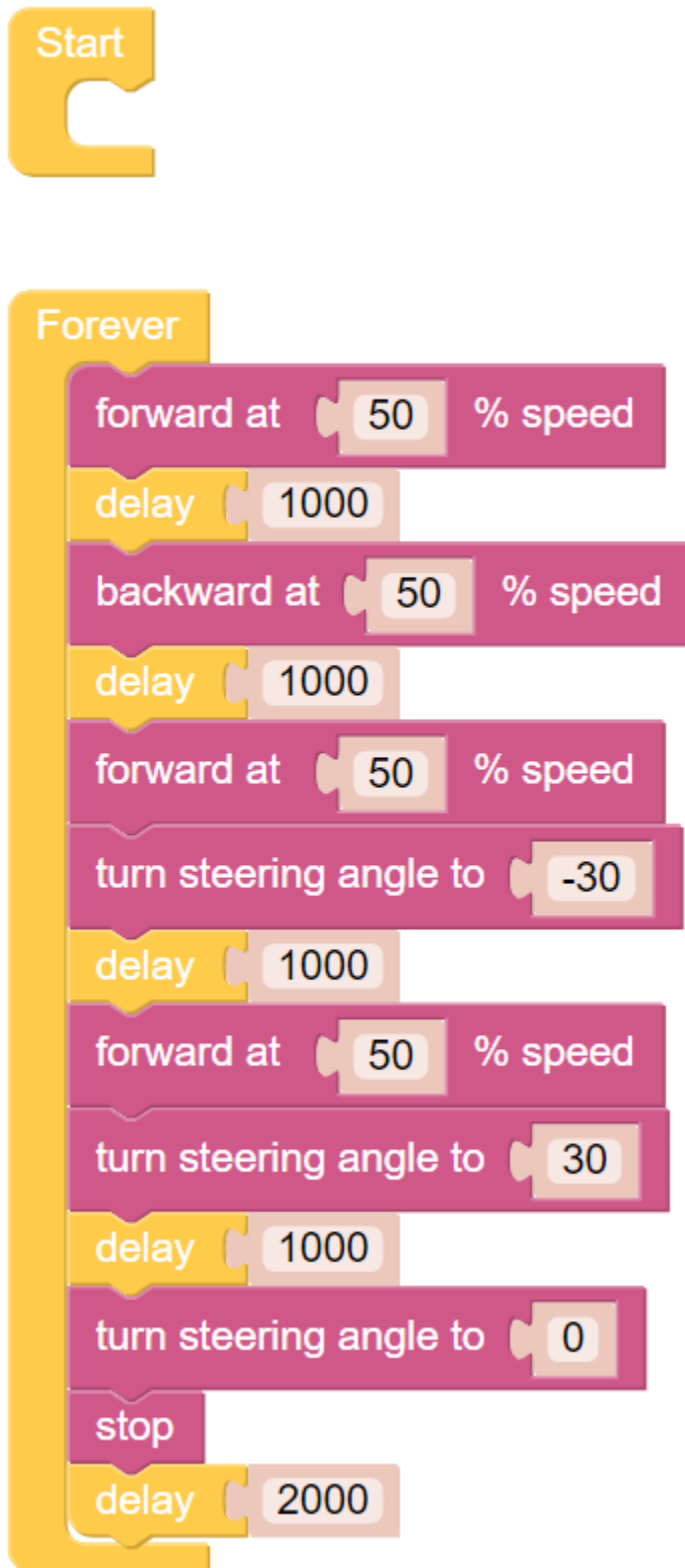


This block will bring the PiCar-X to a complete stop.

EXAMPLE

Note:

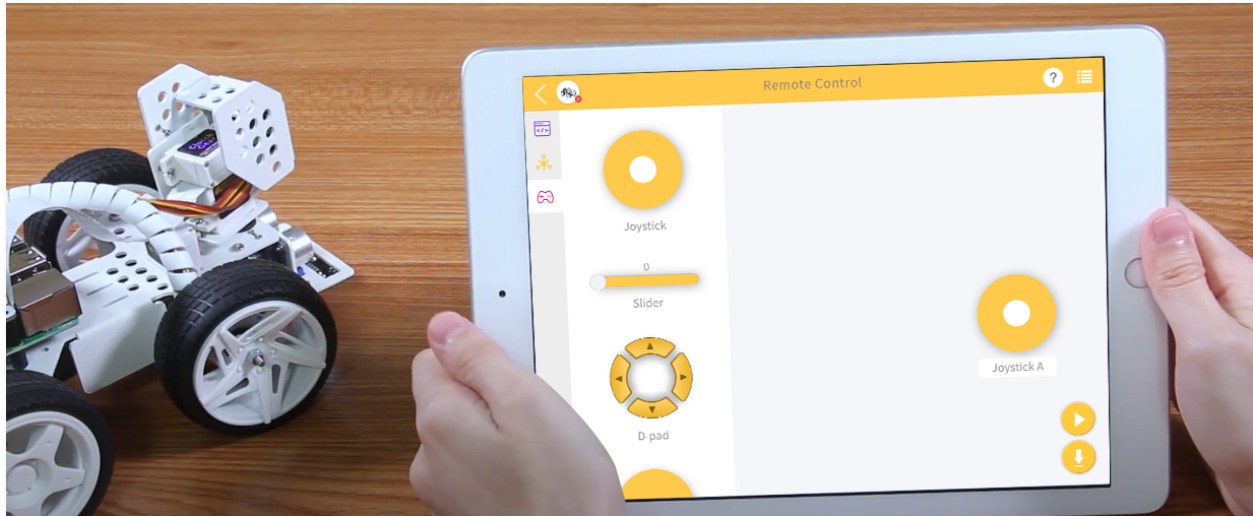
- You can write the program according to the following picture, please refer to the tutorial: [How to Create a New Project?](#).
 - Or find the code with the same name on the **Examples** page of the EzBlock Studio and click **Run** or **Edit** directly.
-



5.3 Remote Control

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

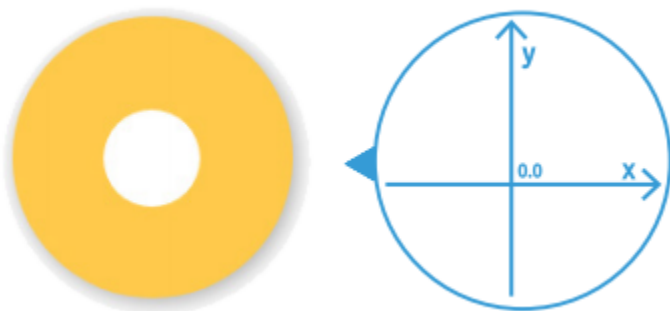
- [How to Use the Remote Control Function?](#)



TIPS



To use the remote control function, open the Remote Control page from the left side of the main page.

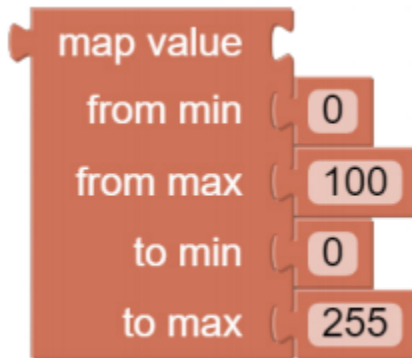


Drag a Joystick to the central area of the Remote Control page. Toggling the white point in the center, and gently dragging in any direction will produce an (X,Y) coordinate. The range of the X-axis or Y-axis is defaulted to “-100”

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



After dragging and dropping a widget on the remote control page, a new category-Remote with the above block will appear. This block reads the Joystick value in the Remote Control page. You can click the drop-down menu to switch to the Y-axis reading.

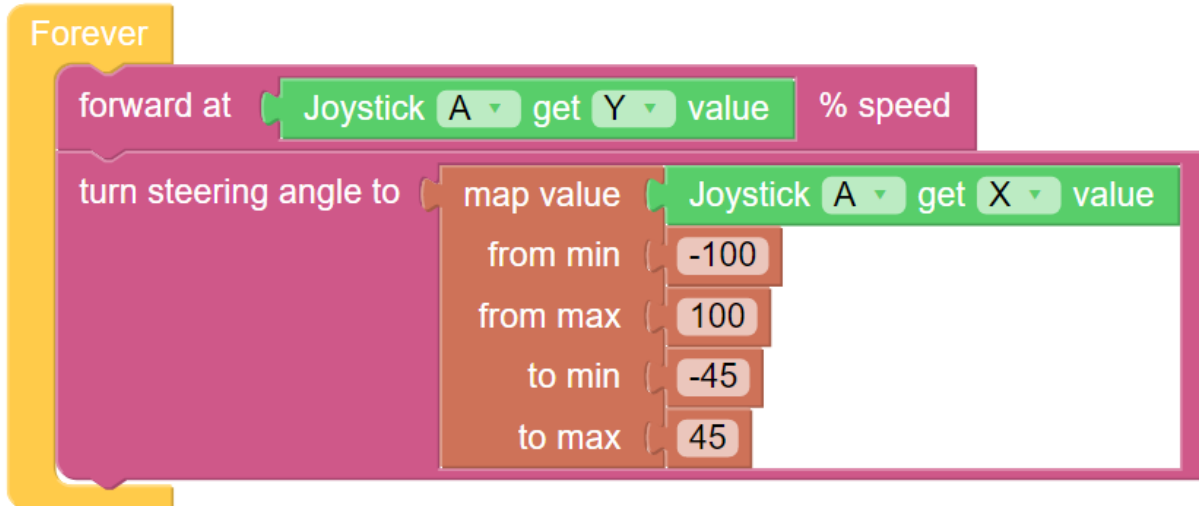


The map value block can remap a number from one range to another. If the range is set to 0 to 100, and the map value number is 50, then it is at a 50% position of the range, or “50”. If the range is set to 0 to 255 and the map value number is 50, then it is at a 50% position of the range, or “127.5”.

EXAMPLE

Note:

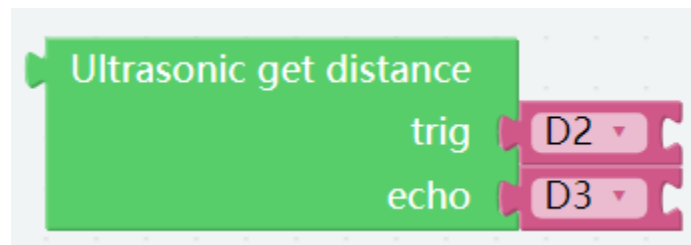
- You can write the program according to the following picture, please refer to the tutorial: [How to Create a New Project?](#).
- Or find the code with the same name on the **Examples** page of the EzBlock Studio and click **Run** or **Edit** directly.



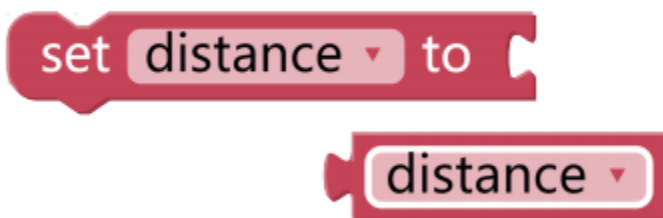
5.4 Test Ultrasonic Module

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.



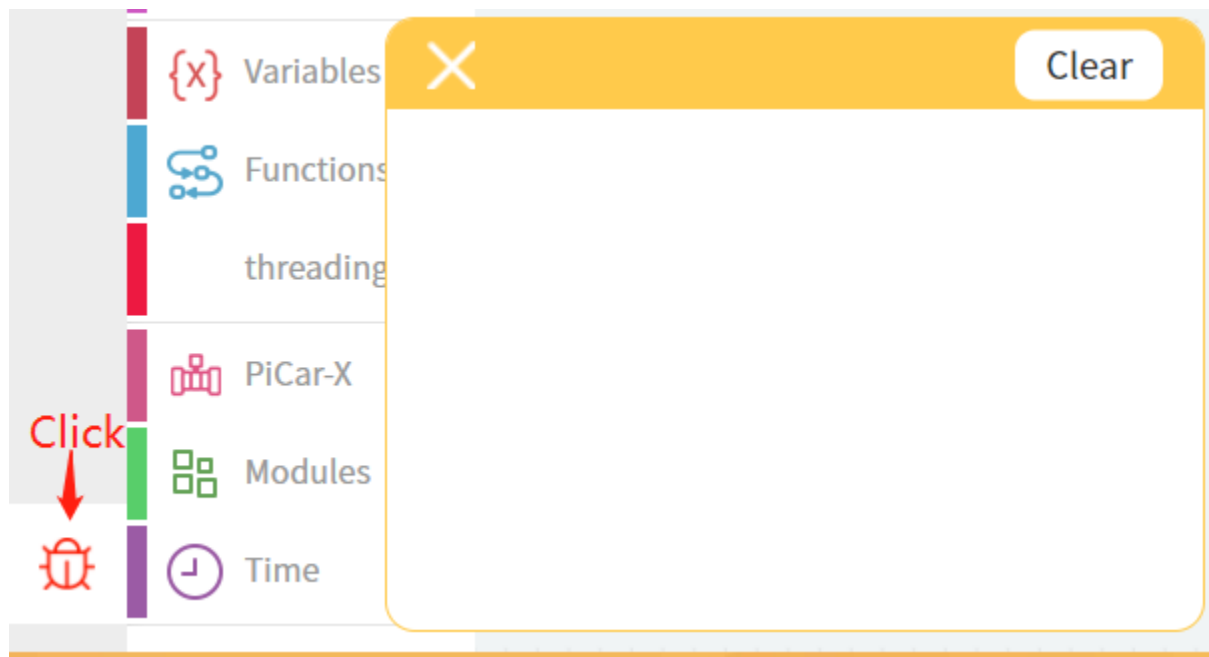
This program is simplified with a **Variable**. For example, when there are multiple functions in a program that each need to use the distance to an obstacle, a **Variable** can be used to report the same distance value to each function, instead of each function reading the same value separately.

Create variable...

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



The **Print** function can print data such as variables and text for easy debugging.

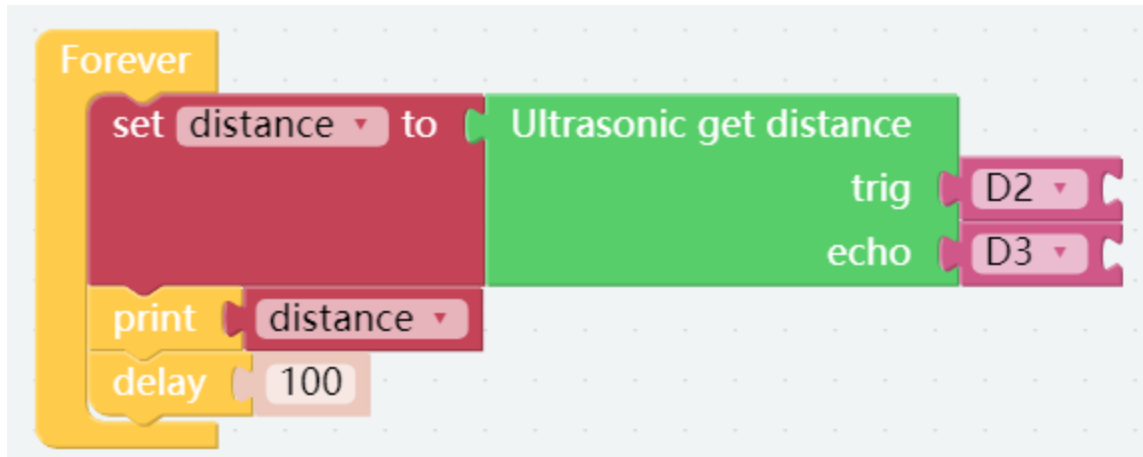


Once the code is running, enable the debug monitor by clicking the **Debug** icon in the bottom left corner.

EXAMPLE

Note:

- You can write the program according to the following picture, please refer to the tutorial: [How to Create a New Project?](#).
- Or find the code with the same name on the **Examples** page of the EzBlock Studio and click **Run** or **Edit** directly.



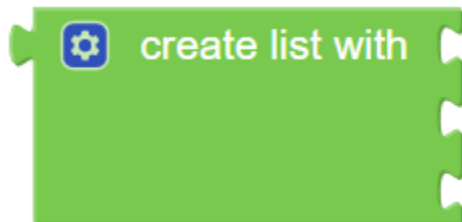
5.5 Test Grayscale Module

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

TIPS



Use the **Grayscale module** block to read the value of one of the sensors. In the example above, the “A0” sensor is the sensor on the far left of the PiCar-X. Use the drop-down arrow to change the sensor to “A1” (center sensor), or “A2” (far right sensor).

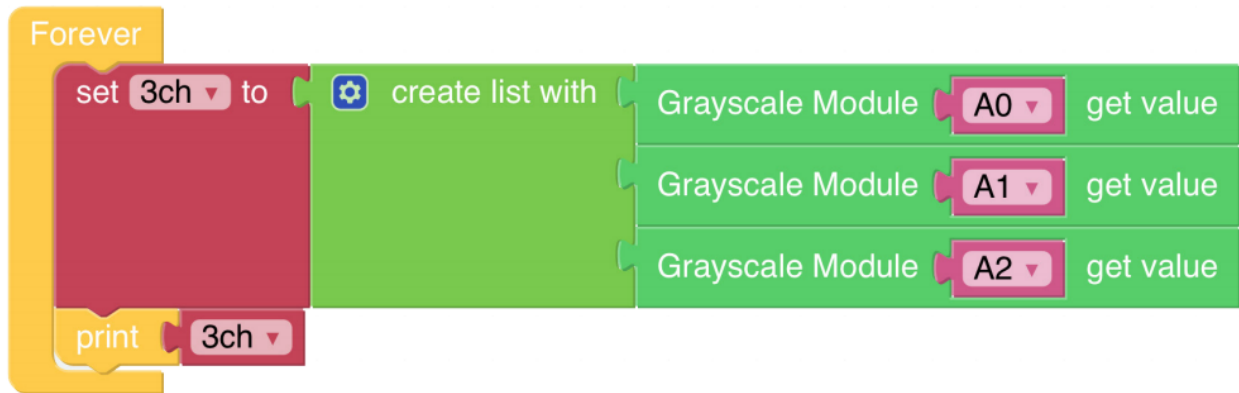


The program is simplified with a **create list with** block. A **List** is used in the same way as a single **Variable**, but in this case a **List** is more efficient than a single **Variable** because the **Grayscale module** will be reporting more than one sensor value. The **create list with** block will create separate **Variables** for each sensor, and put them into a List.

EXAMPLE

Note:

- You can write the program according to the following picture, please refer to the tutorial: [How to Create a New Project?](#).
- Or find the code with the same name on the **Examples** page of the EzBlock Studio and click **Run** or **Edit** directly.



5.6 Color Detection

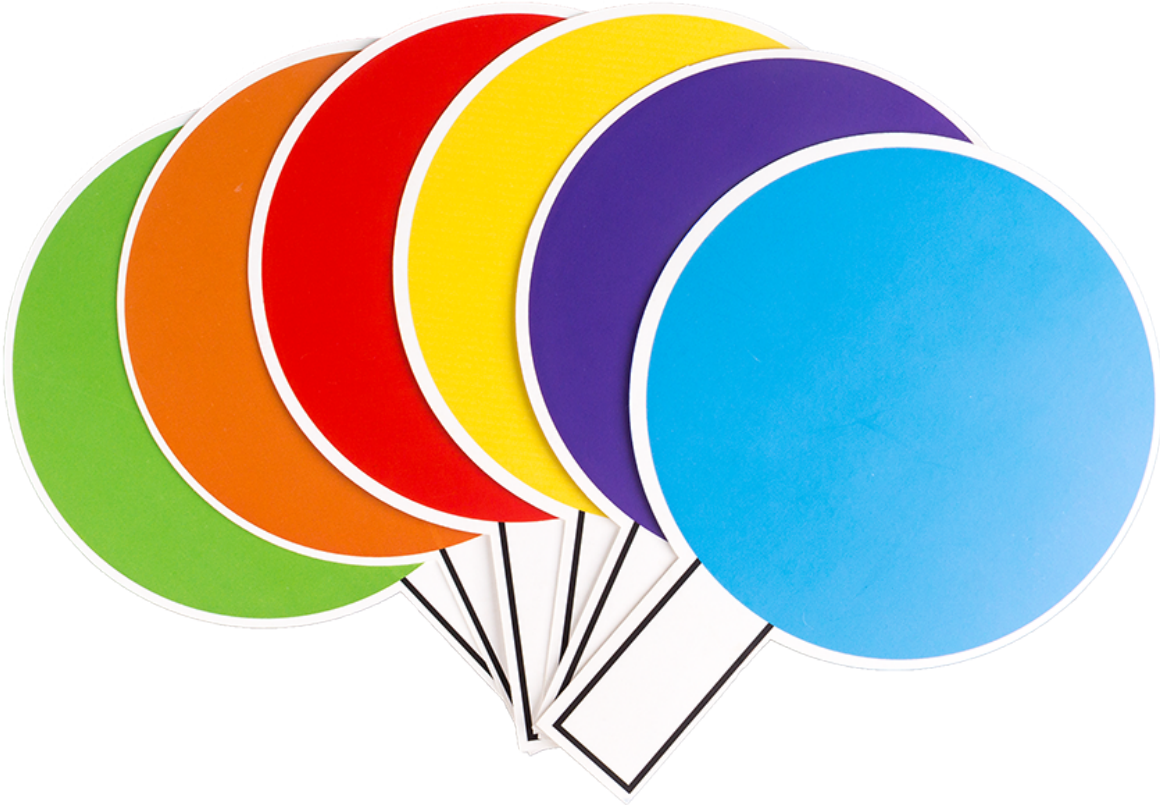
PiCar-X is a self-driving car with a built-in camera, which allows Ezblock programs to utilize object detection and color recognition code. In this section, Ezblock will be used to create a program for color detection.

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

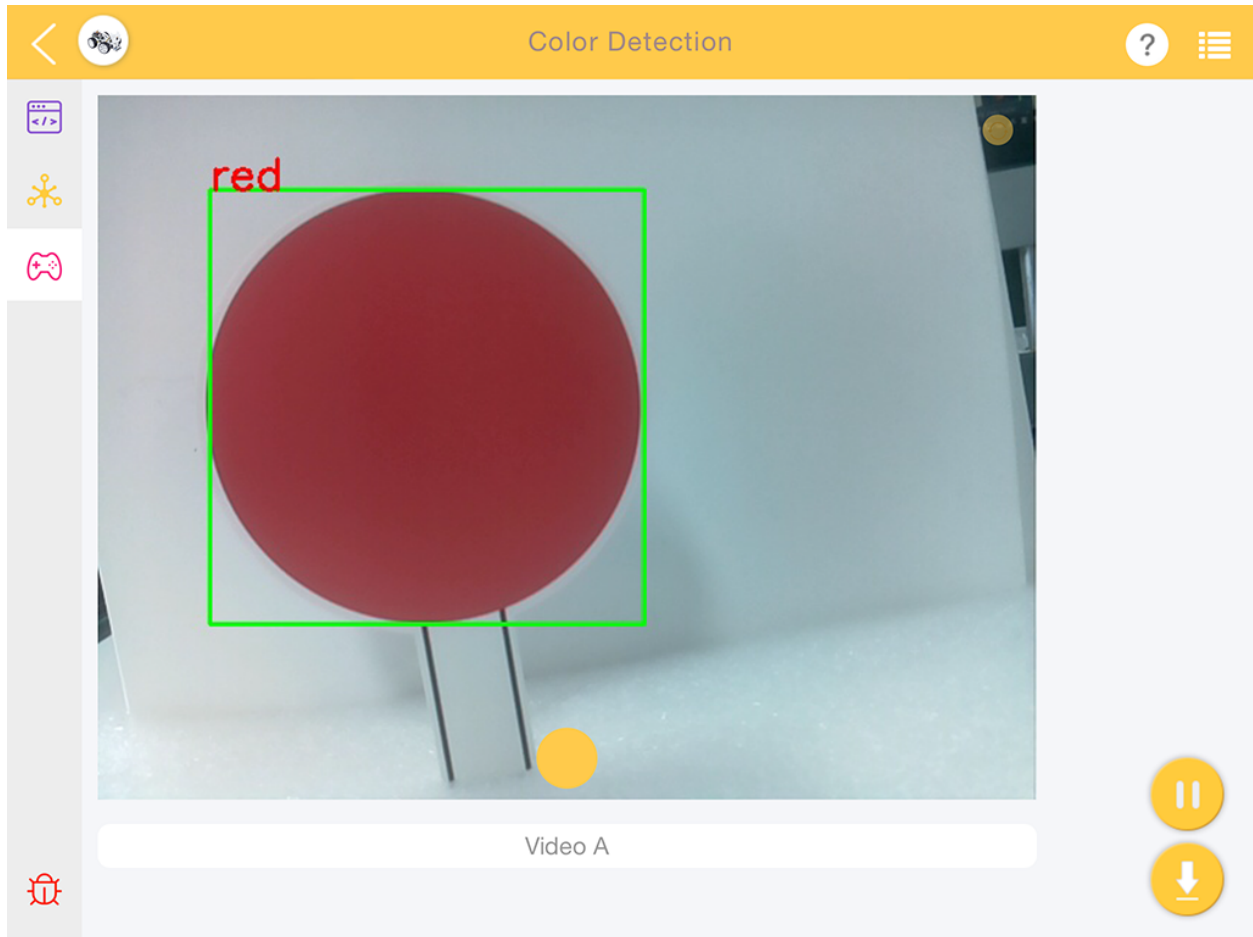
In this program, Ezblock will first be told the Hue-Saturation-Value (HSV) space range of the color to be detected, then utilize OpenCV to process the colors in the HSV range to remove the background noise, and finally, box the matching color.

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

- [\[PDF\]Color Cards](#)



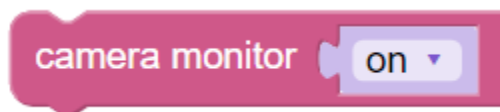
Note: The printed colors may have a slightly different hue from the Ezbloc color models due to printer toner differences, or the printed medium, such as a tan-colored paper. This can cause a less accurate color recognition.



TIPS



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



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

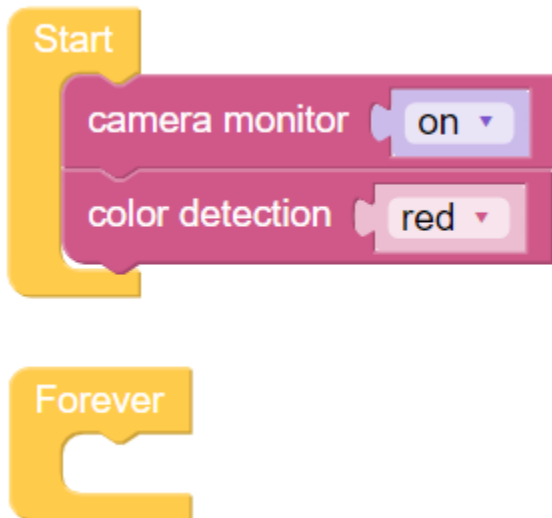


Use the **color detection** block to enable the color detection. Note: only one color can be detected at a time.

EXAMPLE

Note:

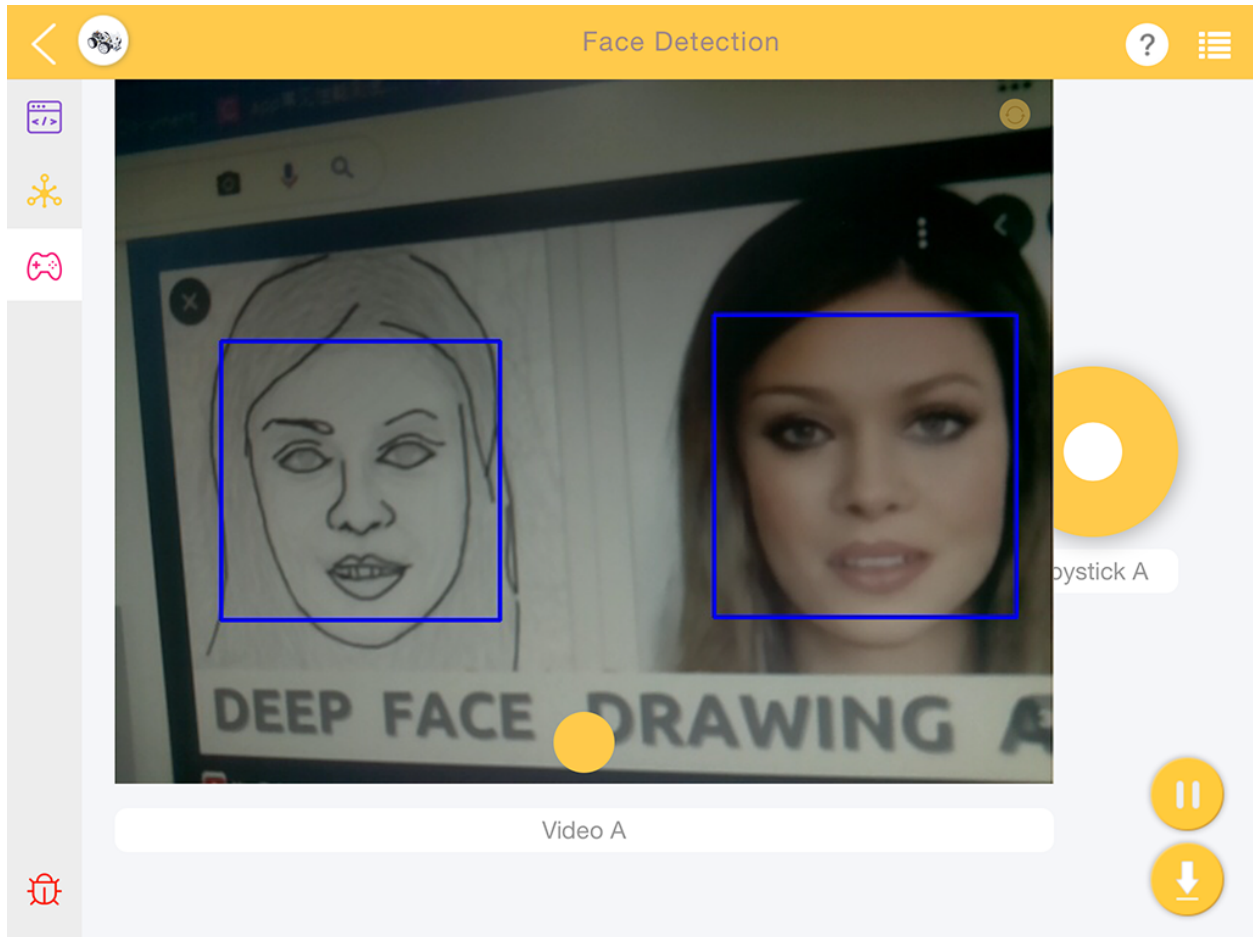
- You can write the program according to the following picture, please refer to the tutorial: [How to Create a New Project?](#).
 - Or find the code with the same name on the **Examples** page of the EzBlock Studio and click **Run** or **Edit** directly.
-



5.7 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

face detection on

Set the **face detection** widget to **on** to enable facial detection.

turn camera pan angle to 0

turn camera tilt angle to 0

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.

number of detected face

The image detection results are given through the of **detected face** block. Use the drop-down menu options to choose between reading the coordinates, size, or number of results from the image detection function.

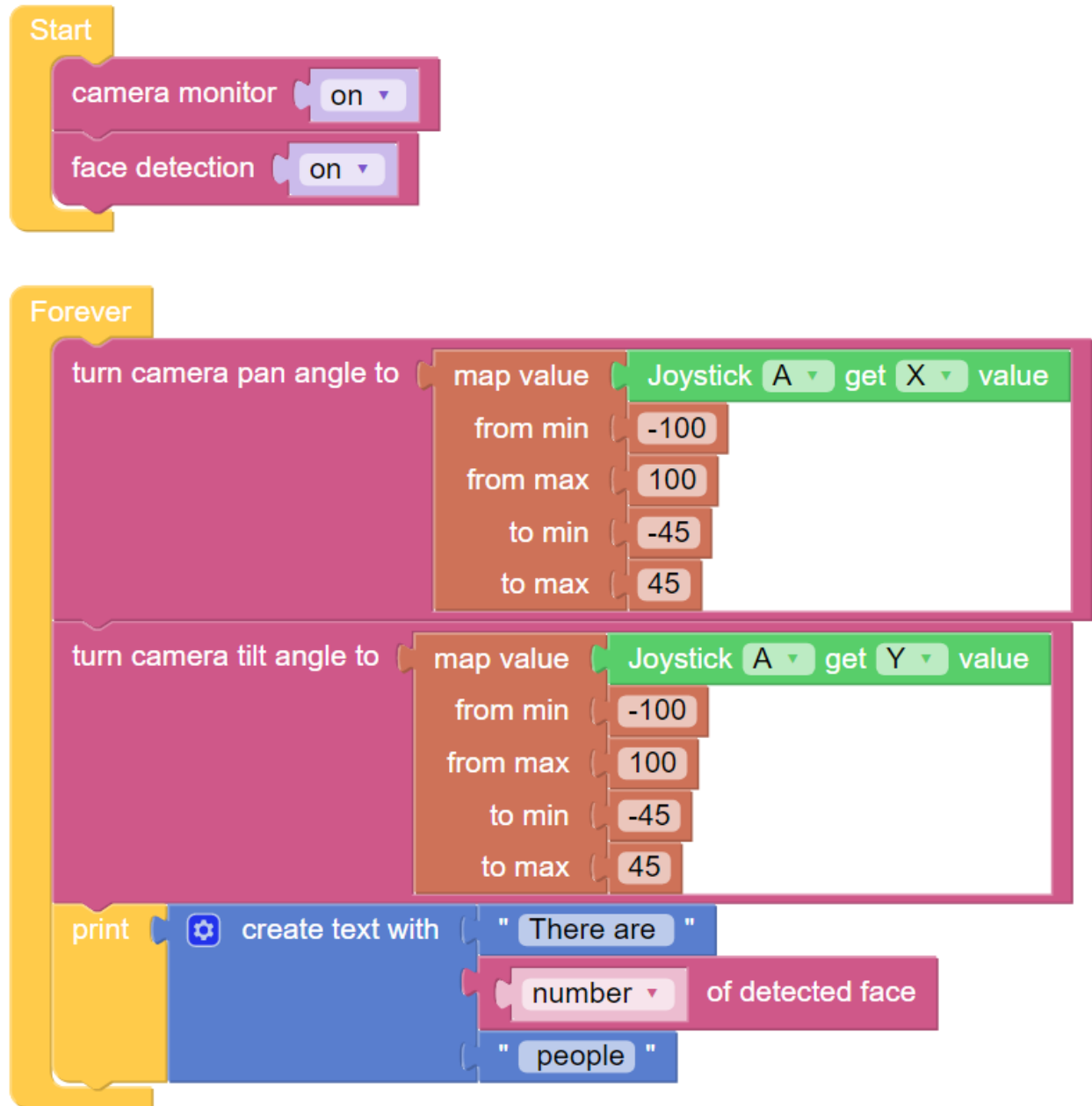


Use the **create text with** block to print the combination of **text** and of **detected face** data.

EXAMPLE

Note:

- You can write the program according to the following picture, please refer to the tutorial: [How to Create a New Project?](#).
 - Or find the code with the same name on the **Examples** page of the EzBlock Studio and click **Run** or **Edit** directly.
-



5.8 Sound Effect

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

TIPS



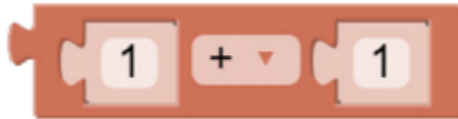
Use the **say** block with a **text** block to write a sentence for the PiCar-X to say. The **say** block can be used with text or numbers.



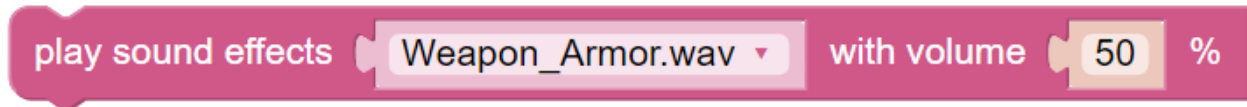
The **number** block.



Using the **repeat** block will repeatedly execute the same statement, which reduces the size of the code.



The **mathematical operation** block can perform typical mathematical functions, such as "+", "-", "x", and "÷".

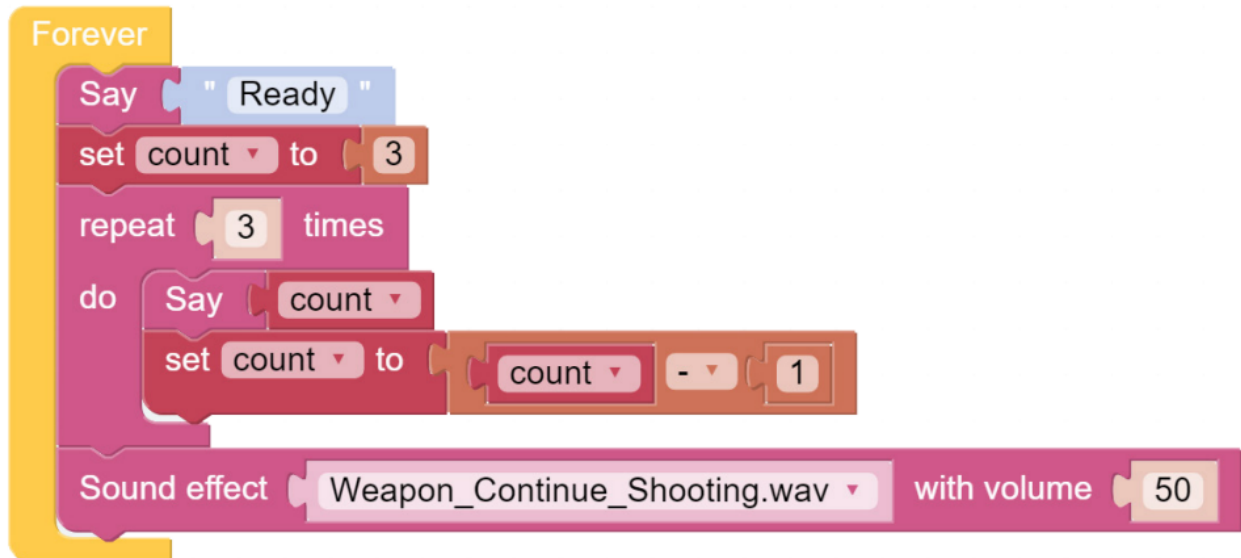


The play **sound effects - with volume - %** block has preset sound effects, such as a siren sound, a gun sound, and others. The range of the volume can be set from 0 to 100.

EXAMPLE

Note:

- You can write the program according to the following picture, please refer to the tutorial: [How to Create a New Project?](#).
 - Or find the code with the same name on the **Examples** page of the EzBlock Studio and click **Run** or **Edit** directly.
-



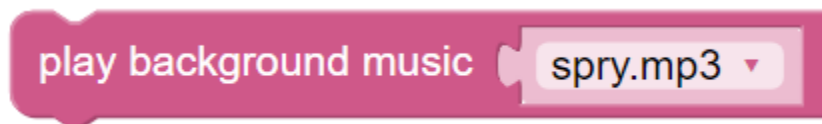
5.9 Background Music

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.

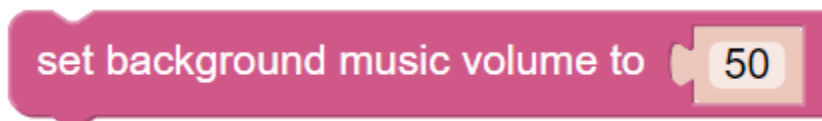
- [How to Use the Remote Control Function?](#)

For a detailed tutorial on Ezblocs remote control functions, please reference the [Remote Control](#) tutorial.

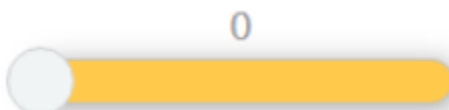
TIPS



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



The block **set background music volume to** will adjust the volume between the range of 0 to 100.



Drag a **Slider** bar from the **Remote Control** page to adjust music volume.



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

EXAMPLE

Note:

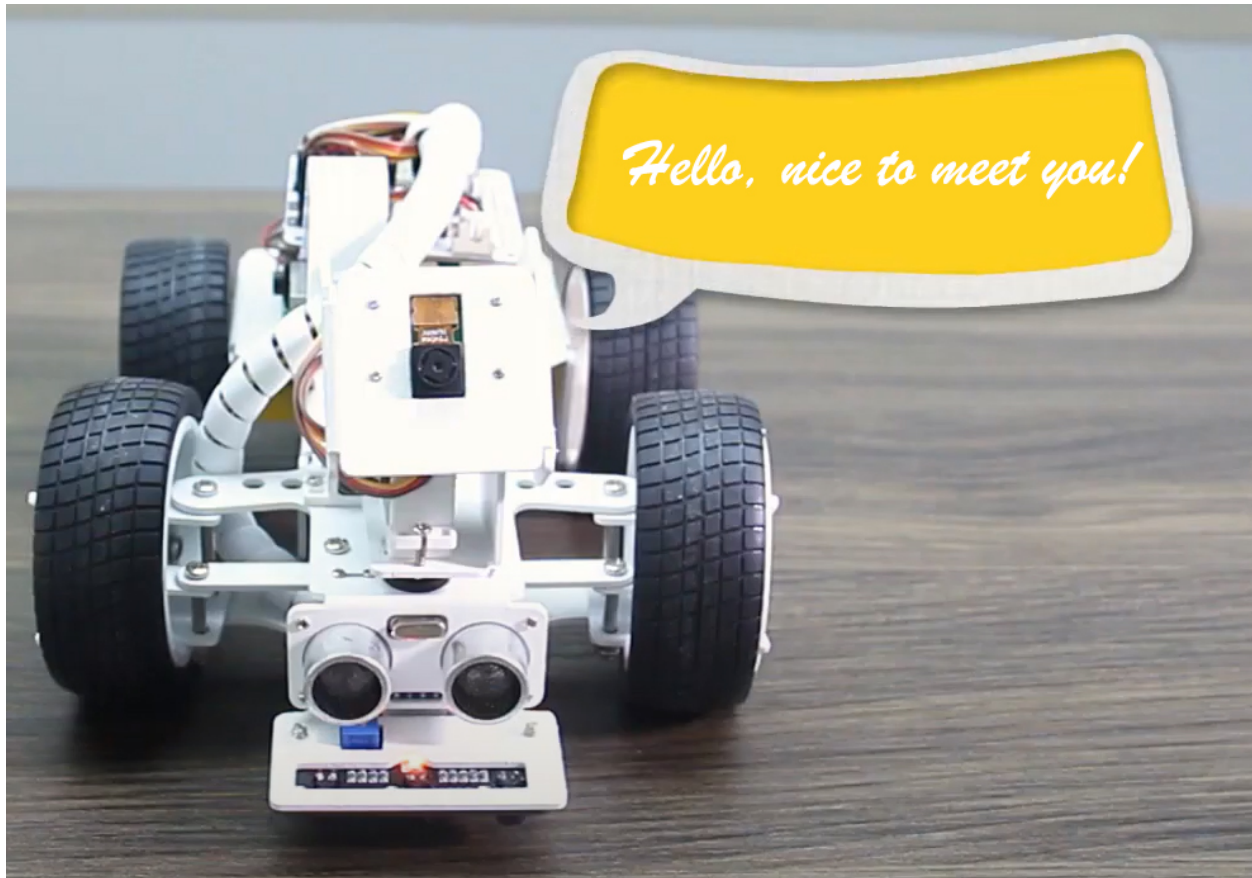
- You can write the program according to the following picture, please refer to the tutorial: [How to Create a New Project?](#).
 - Or find the code with the same name on the **Examples** page of the EzBlock Studio and click **Run** or **Edit** directly.
-



5.10 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.

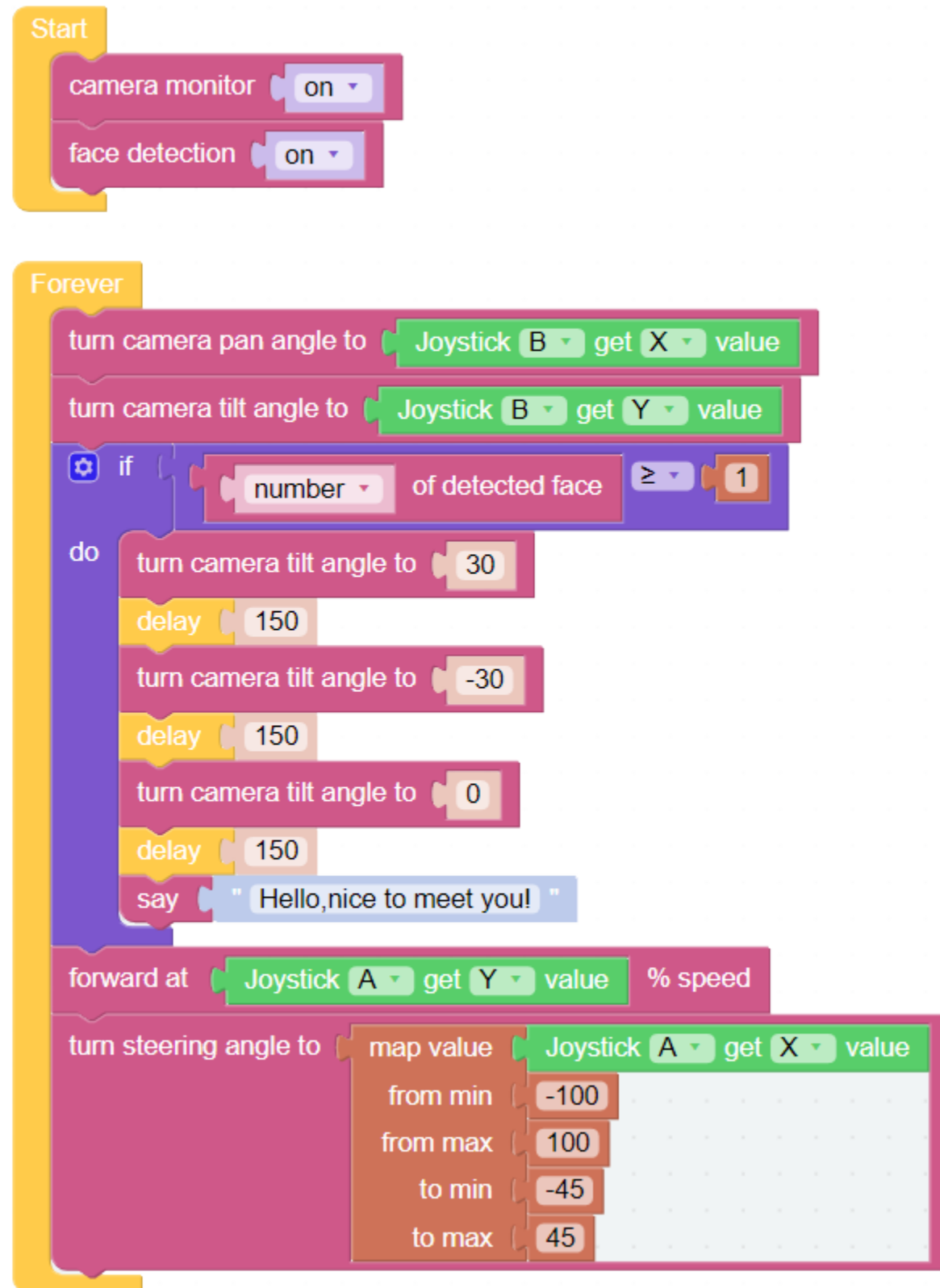


The **conditional statements** block is used in conjunction with the **if do** block. The conditions can be “=”, “>”, “<”, “,” “,” “,” or “.”.

EXAMPLE

Note:

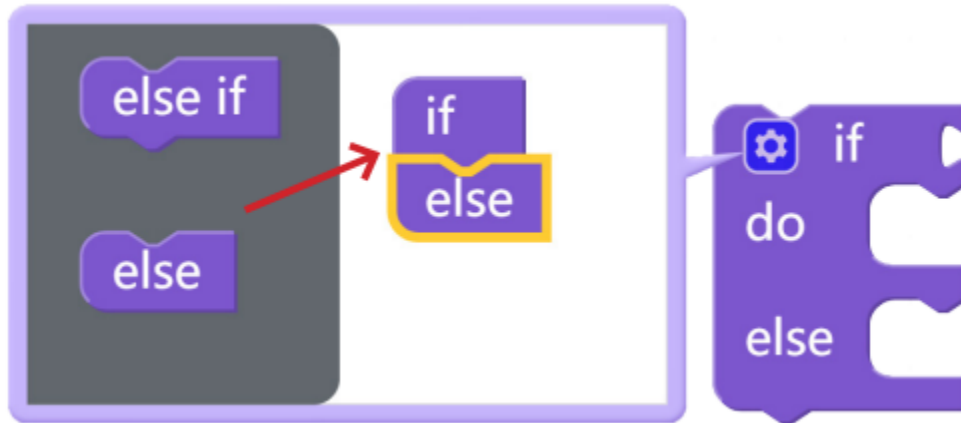
- You can write the program according to the following picture, please refer to the tutorial: [How to Create a New Project?](#).
- Or find the code with the same name on the **Examples** page of the EzBlock Studio and click **Run** or **Edit** directly.



5.11 Music Car

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

TIPS

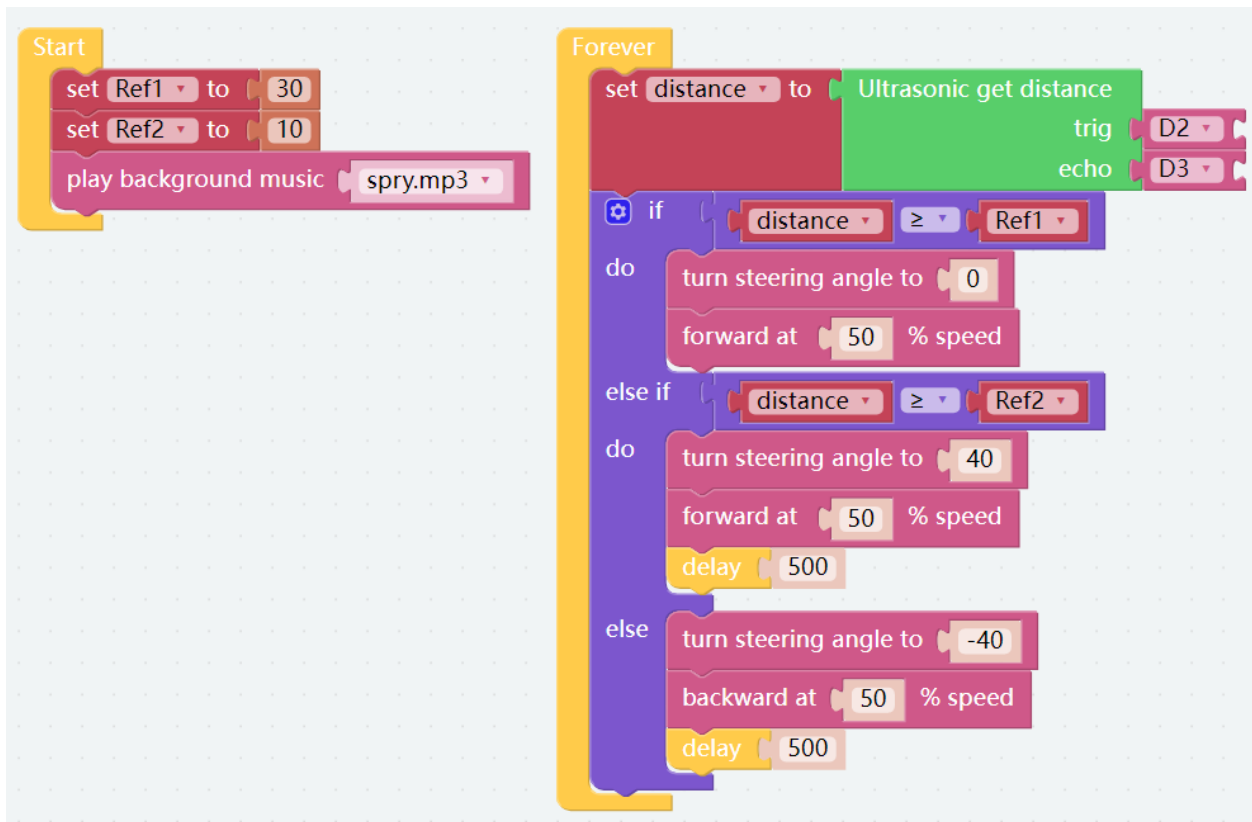


To implement multiple conditional judgments, change the simple if do block into an if else do / else if do block. This is done by clicking on the setting icon as shown above.

EXAMPLE

Note:

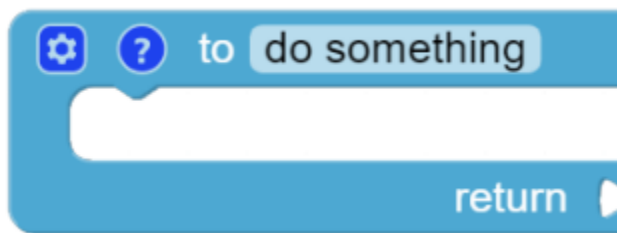
- You can write the program according to the following picture, please refer to the tutorial: [How to Create a New Project?](#).
- Or find the code with the same name on the **Examples** page of the EzBlock Studio and click **Run** or **Edit** directly.



5.12 Cliff Detection

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

TIPS

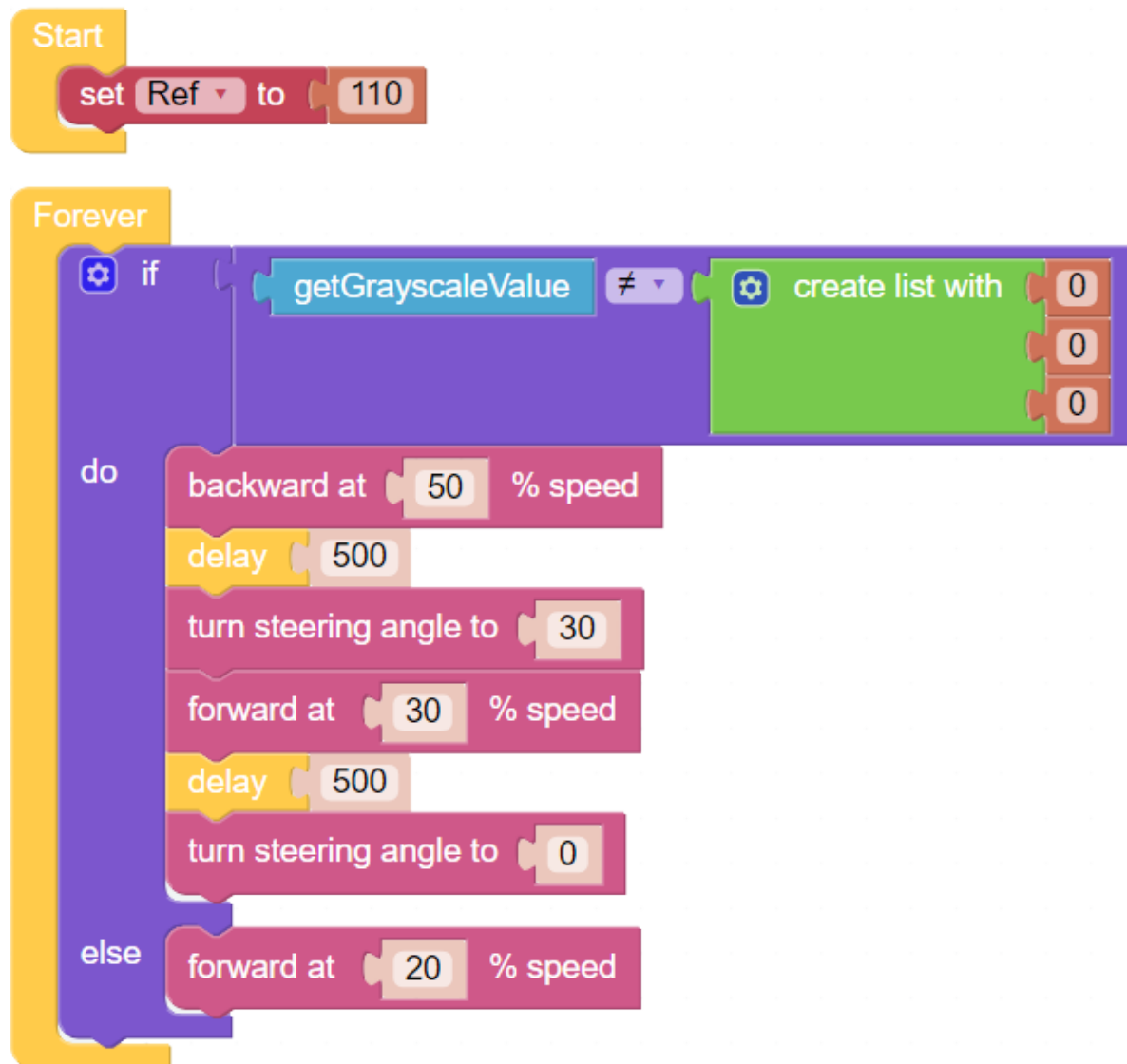


The **grayscale module** will be performing the same operation multiple times. To simplify the program, this project introduces a **function** that will return a **list** variable to the **do forever** block.

EXAMPLE

Note:

- You can write the program according to the following picture, please refer to the tutorial: [How to Create a New Project?](#).
- Or find the code with the same name on the **Examples** page of the EzBlock Studio and click **Run** or **Edit** directly.





5.13 Minecart

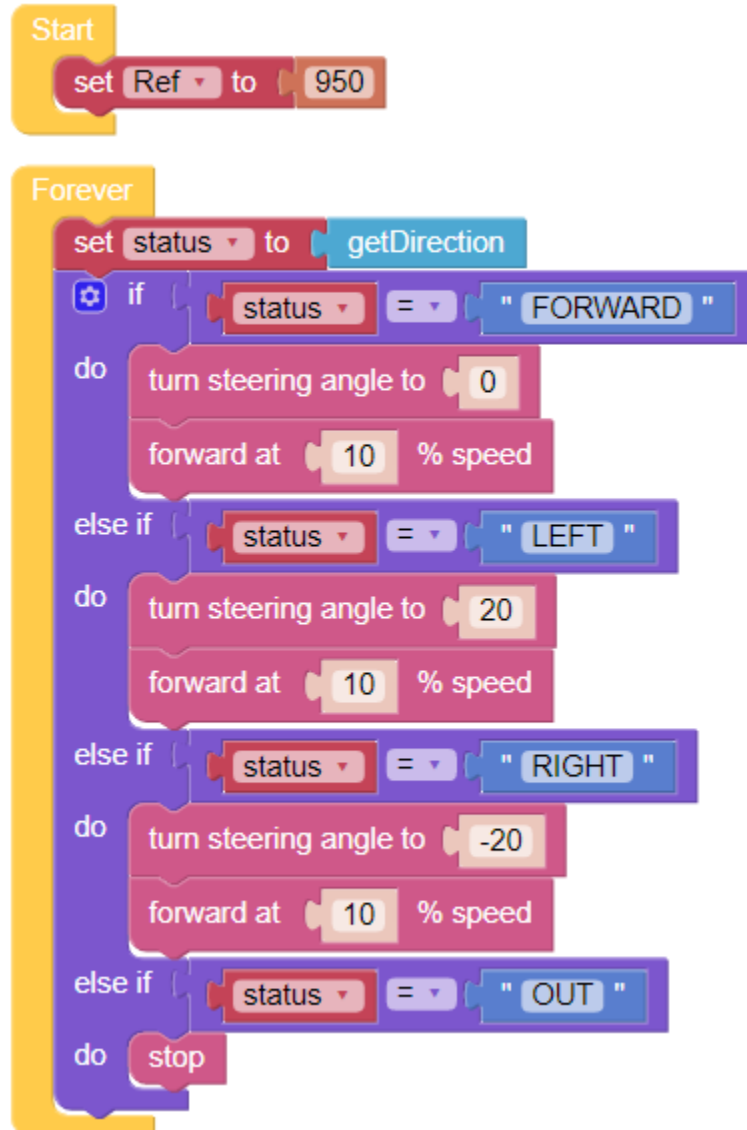
Let's make a minecart project! This project will use the Grayscale module to make the PiCar-X move forward along a track. Use dark-colored tape to make a track on the ground as straight as possible, and not too curved. Some experimenting might be needed if the PiCar-X becomes derailed.

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.

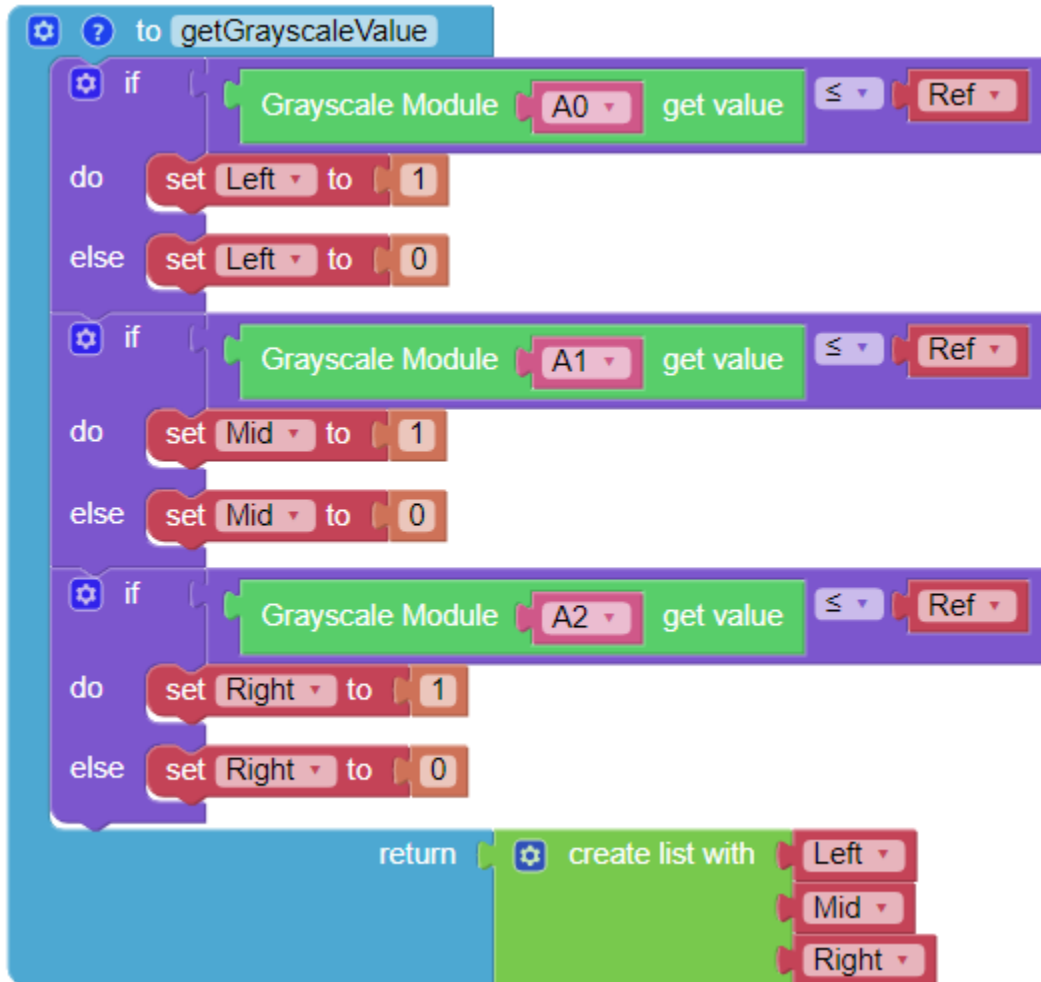
EXAMPLE

Note:

- You can write the program according to the following picture, please refer to the tutorial: [How to Create a New Project?](#).
- Or find the code with the same name on the **Examples** page of the EzBlock Studio and click **Run** or **Edit** directly.

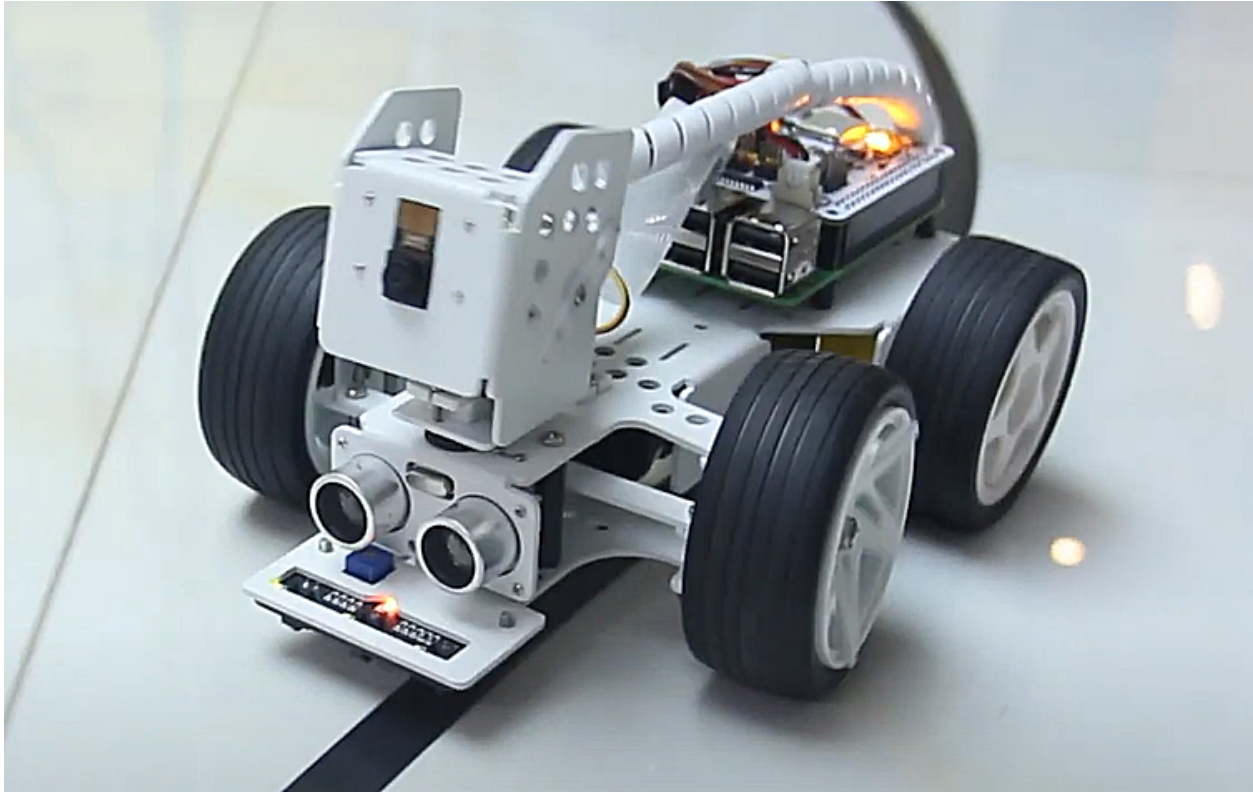


```
to getDirection
  set value to getGrayscaleValue
  if (value = create list with 0) or (value = create list with 1)
    do set direction to "FORWARD"
  else if (value = create list with 1) or (value = create list with 0)
    do set direction to "RIGHT"
  else if (value = create list with 0) or (value = create list with 1)
    do set direction to "LEFT"
  else if (value = create list with 0)
    do set direction to "OUT"
  return direction
```

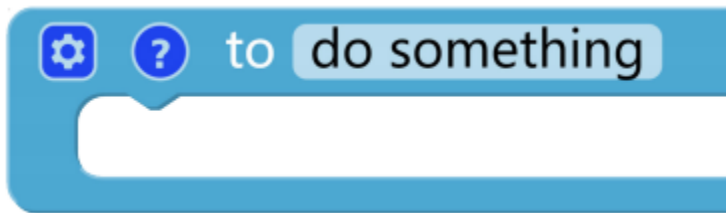



5.14 Minecart Plus

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

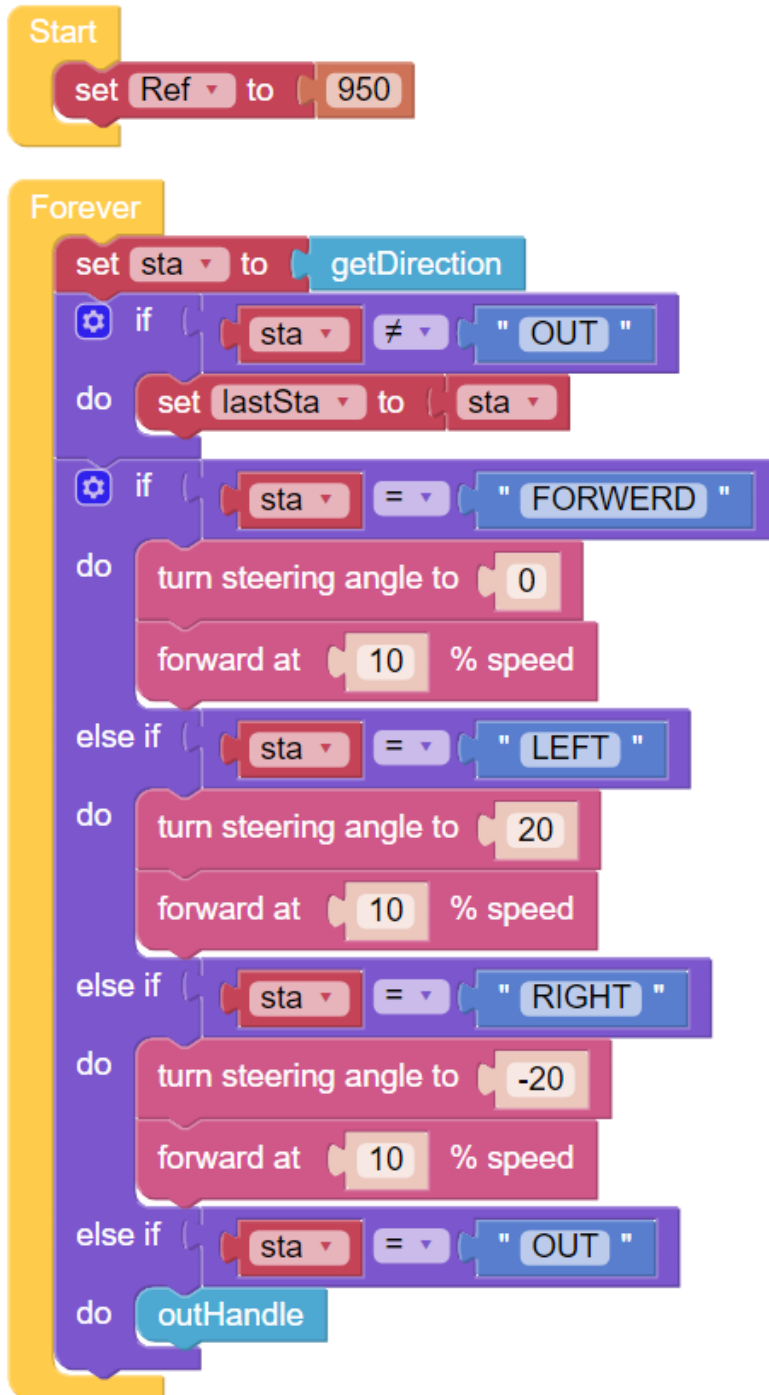


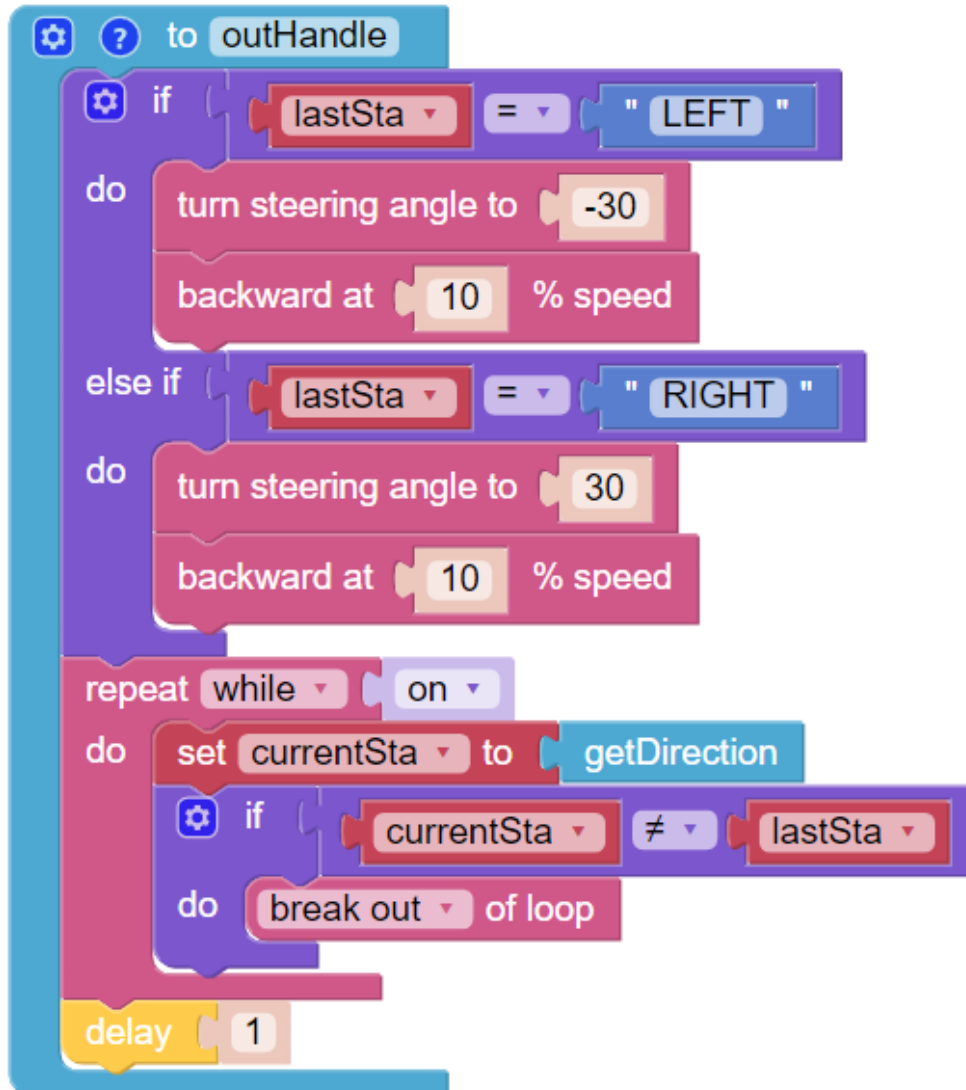
Use another **to do something** block to allow the PiCar-X to back up and recover from a sharp curve. Note that the new **to do something** function does not return any values, but is used just for reorienting the PiCar-X.

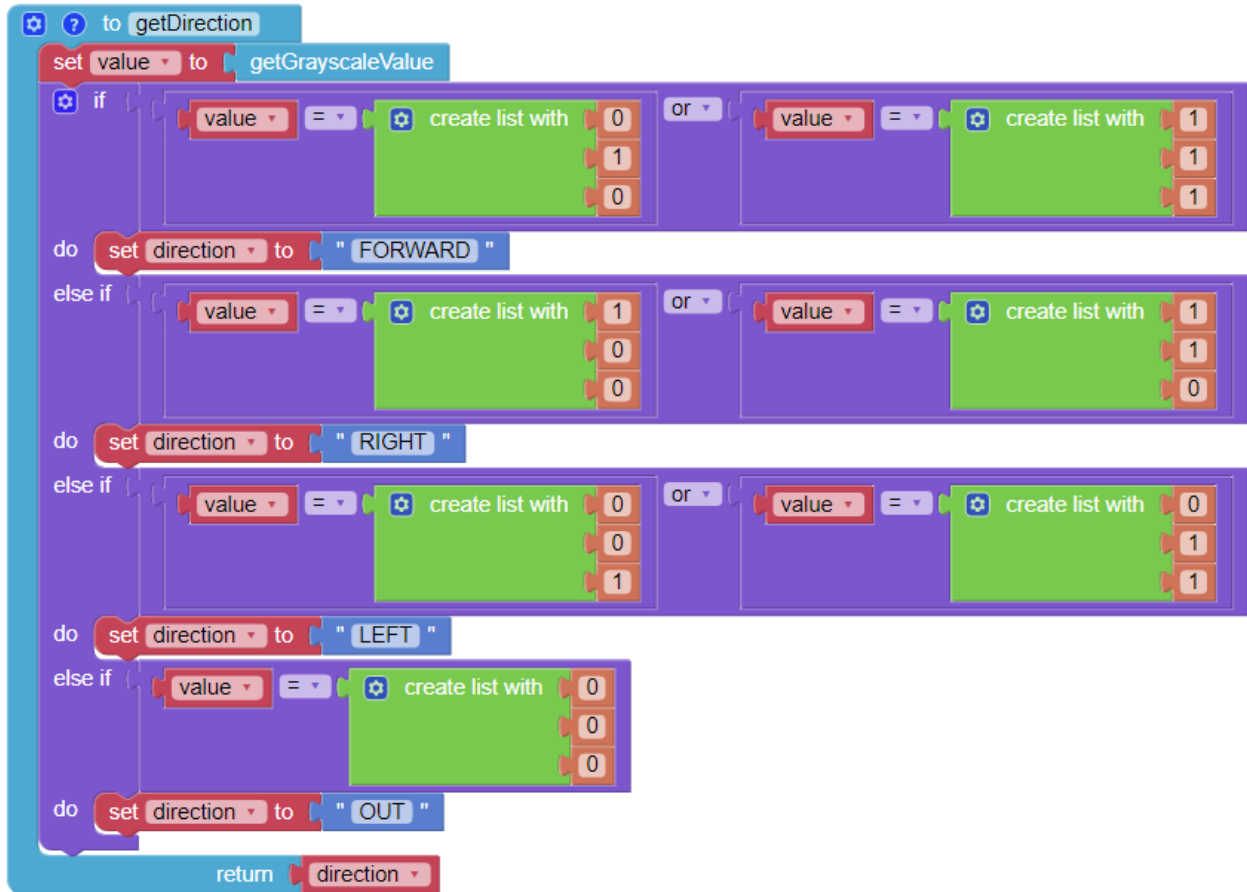
EXAMPLE

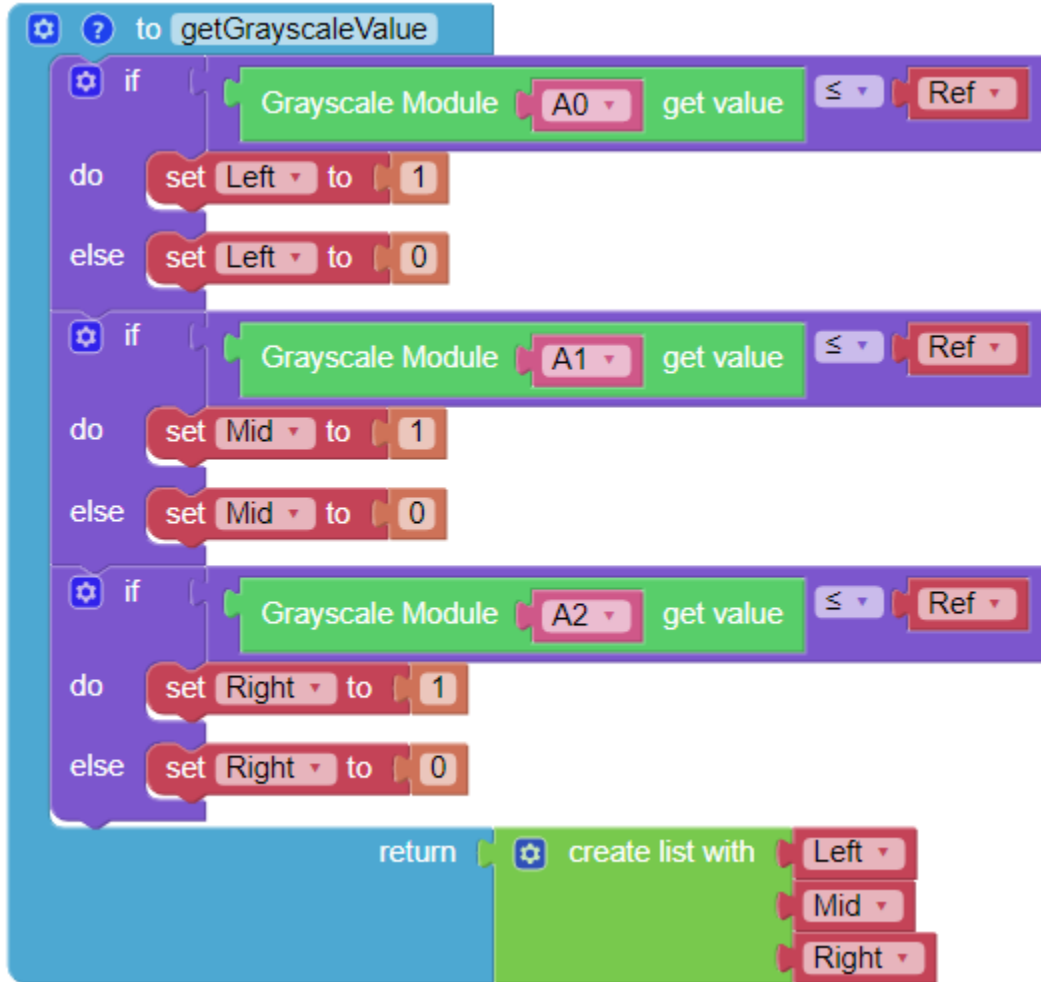
Note:

- You can write the program according to the following picture, please refer to the tutorial: [How to Create a New Project?](#).
 - Or find the code with the same name on the **Examples** page of the EzBlock Studio and click **Run** or **Edit** directly.
-









5.15 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.

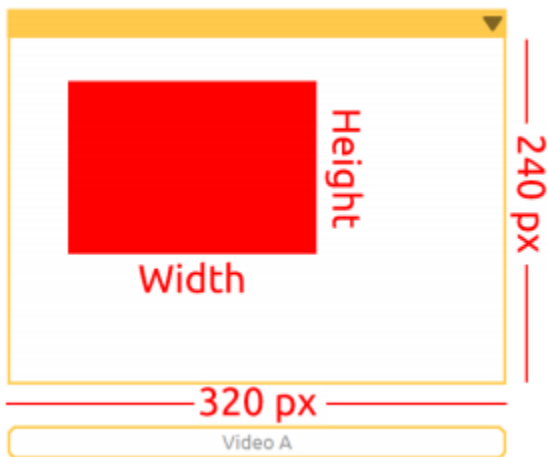
TIPS



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.

(-1,1)	(0,1)	(1,1)
(-1,0)	(0,0)	(1,0)
(-1,-1)	(0,-1)	(1,-1)

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

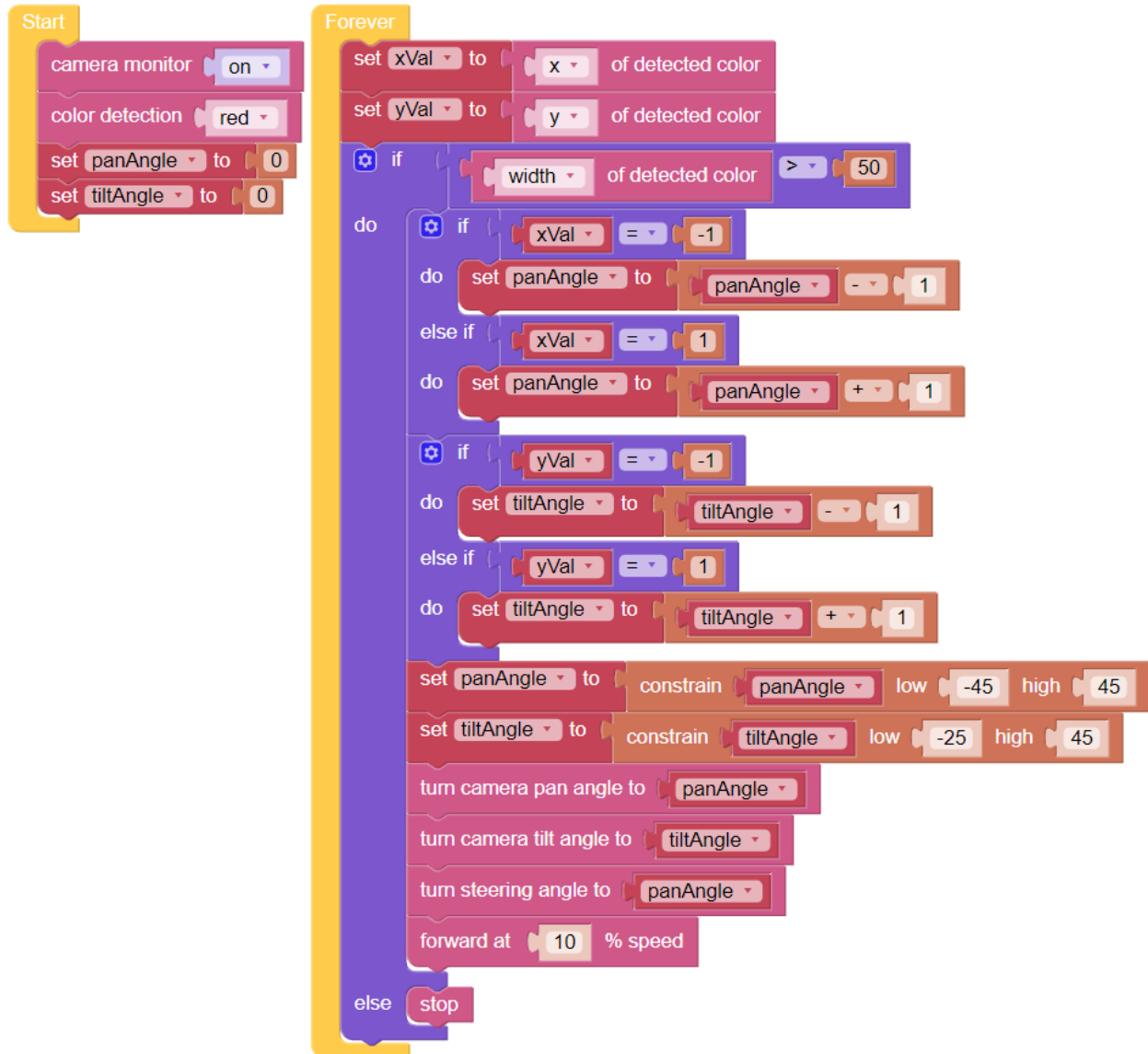


The “object detection” will detect the Width and Height of the graphic. If multiple targets are identified, the dimensions of the largest target will be recorded.

EXAMPLE

Note:

- You can write the program according to the following picture, please refer to the tutorial: [How to Create a New Project?](#).
- Or find the code with the same name on the **Examples** page of the EzBlock Studio and click **Run** or **Edit** directly.



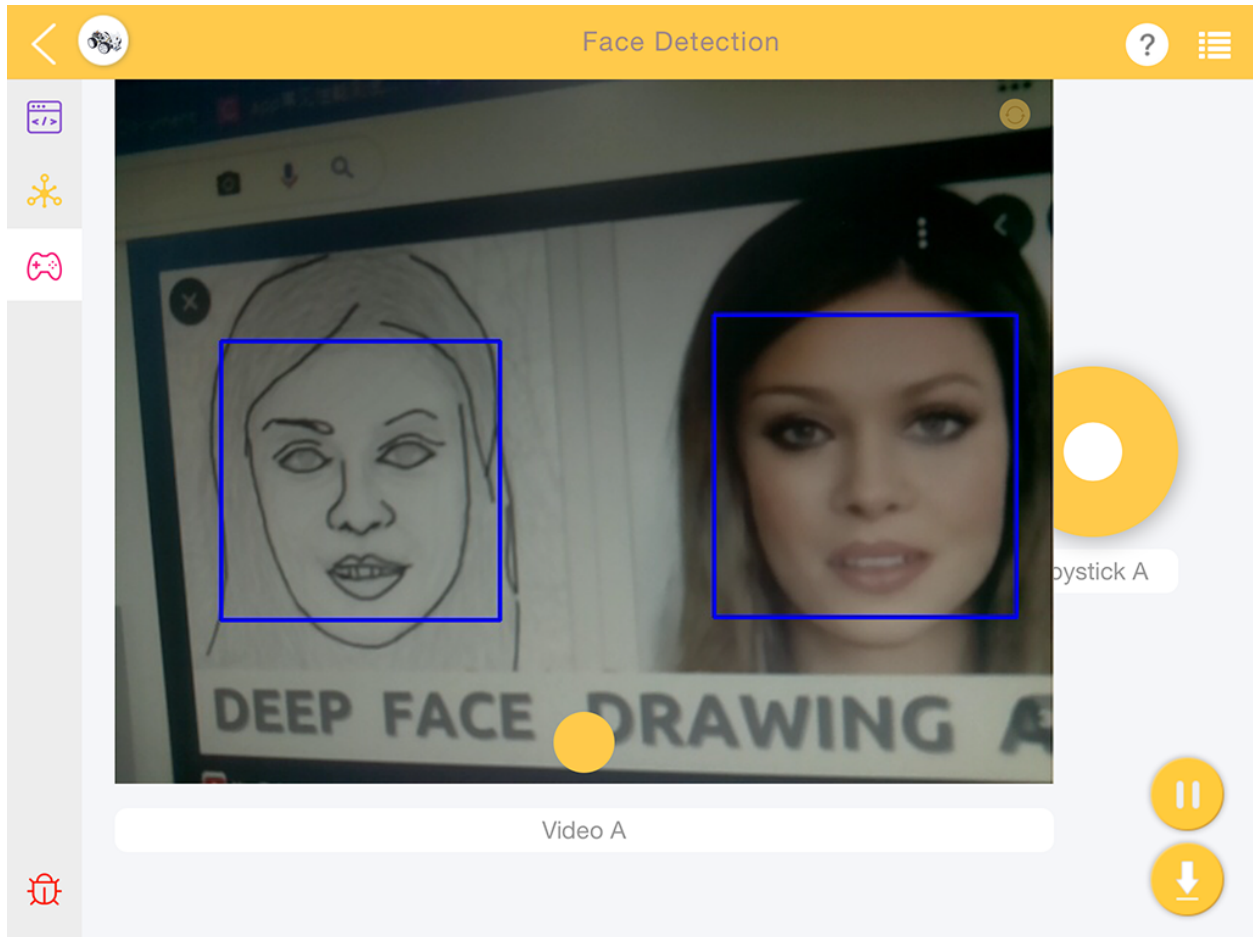
5.16 Beware of Pedestrians

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

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

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

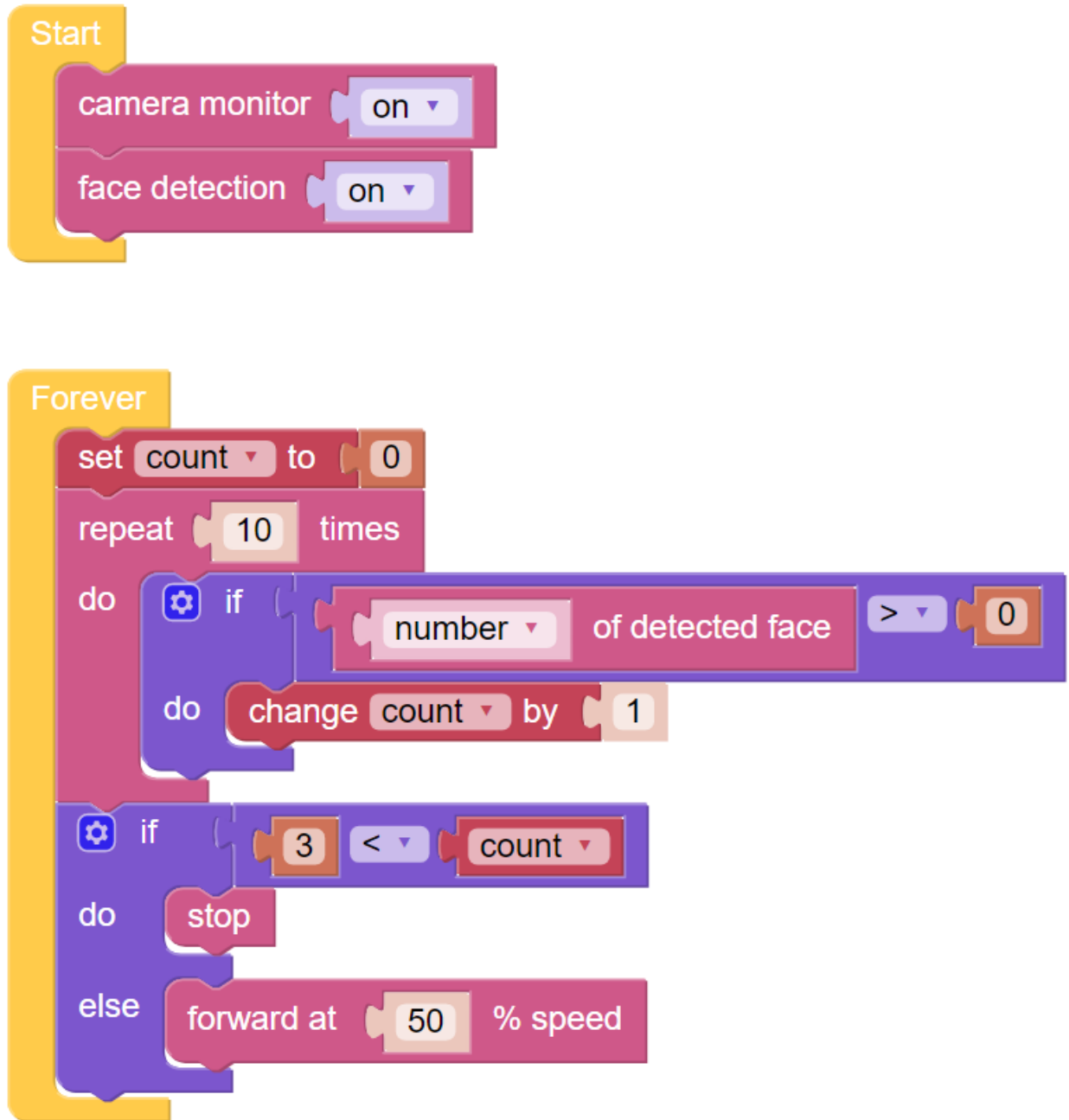
- [How to Use the Remote Control Function?](#)



EXAMPLE

Note:

- You can write the program according to the following picture, please refer to the tutorial: [How to Create a New Project?](#).
- Or find the code with the same name on the **Examples** page of the EzBlock Studio and click **Run** or **Edit** directly.



5.17 Traffic Sign Detection

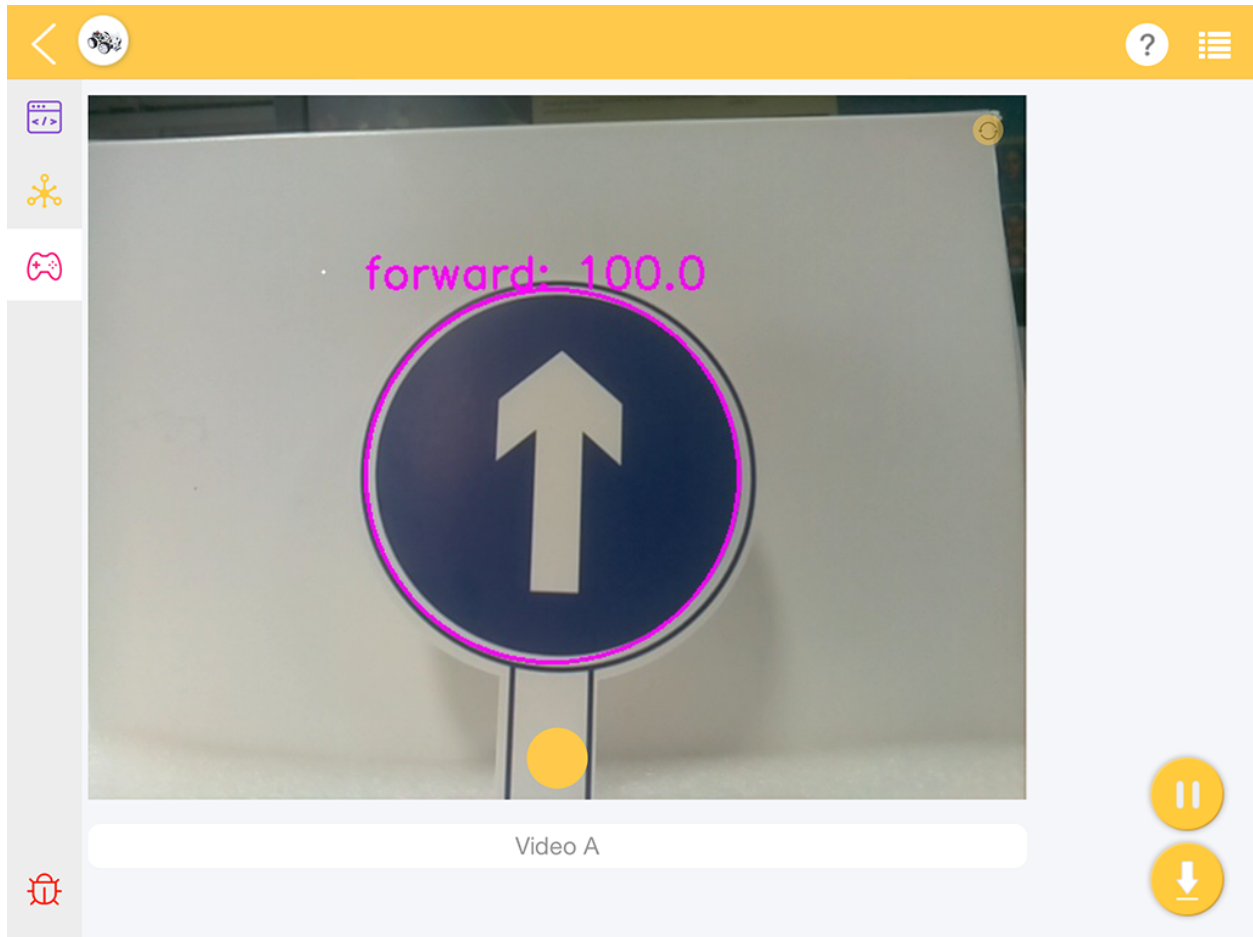
This project uses an image detection function to look for traffic signs, and make the PiCar-X follow the instructions on the sign. The turn **traffic sign detection [on]** block will recognize 4 different traffic sign models included in the printable PDF below. When the PiCar-X detects a **STOP** sign, it will come to a stop, a **FORWARD** sign will make it drive forward, and the **LEFT** or **RIGHT** arrows will make it turn that direction.

- [PDF]Traffic Sign Cards



Note: The printed traffic sign 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.

This project is based on *Minecart*, but instead of using the grayscale sensor, the PiCar-X uses an algorithm for traffic sign detection. The detection results can be viewed via the Video Monitor in Ezblock Studio.



TIPS

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
```

The image shows a Scratch script for a PICAR-X robot. It starts with a 'Start' block containing four initialization steps: setting a 'Ref' variable to 700, turning the 'camera monitor' on, turning 'traffic sign detection' on, and setting a 'Key' variable to 1. The main logic is in a 'Forever' loop. Inside the loop, a 'GetTraffic' block is called. An 'if' statement checks the 'Key' variable. If 'Key' is 0, the robot stops. If 'Key' is 1, the robot enters a 'do' block where it sets a 'status' variable to the value of 'getDirection'. Another 'if' statement then checks the 'status' variable. If 'status' is 'FORWARD', the robot turns its steering angle to 0 and moves forward at 10% speed. If 'status' is 'LEFT', it turns the steering angle to 20 degrees and moves forward at 10% speed. If 'status' is 'RIGHT', it turns the steering angle to -20 degrees and moves forward at 10% speed. If 'status' is 'OUT', the robot stops.

```
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
```



5.18 Orienteering

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

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

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

- [\[PDF\]Color Cards](#)

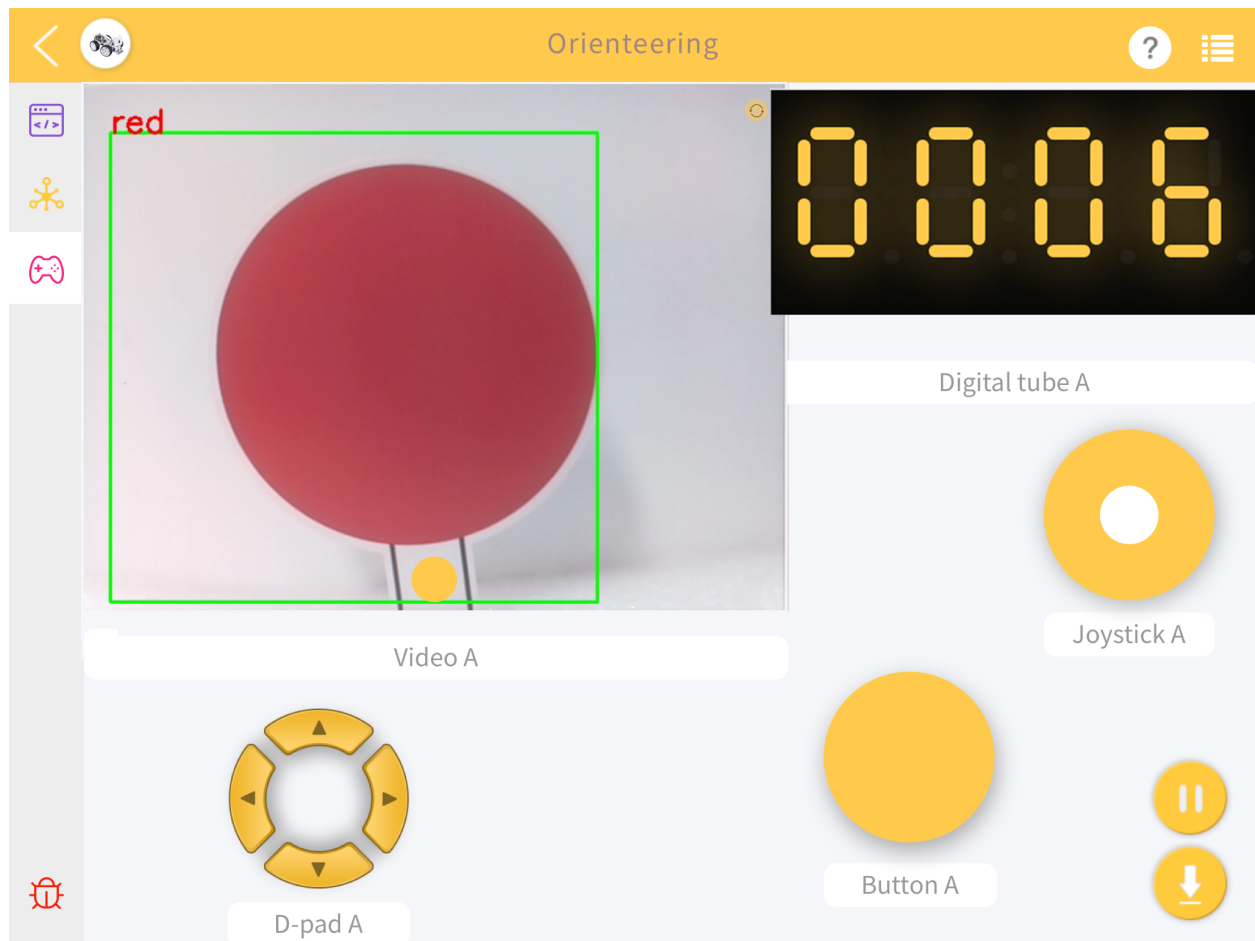


Note: The printed colors may have a slightly different hue from the Ezblock color models due to printer toner differences, or the printed medium, such as a tan-colored paper. This can cause a less accurate color recognition.

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

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

Place PiCar-X in the middle of the field and click the Button on the Remote Control page to start the game.

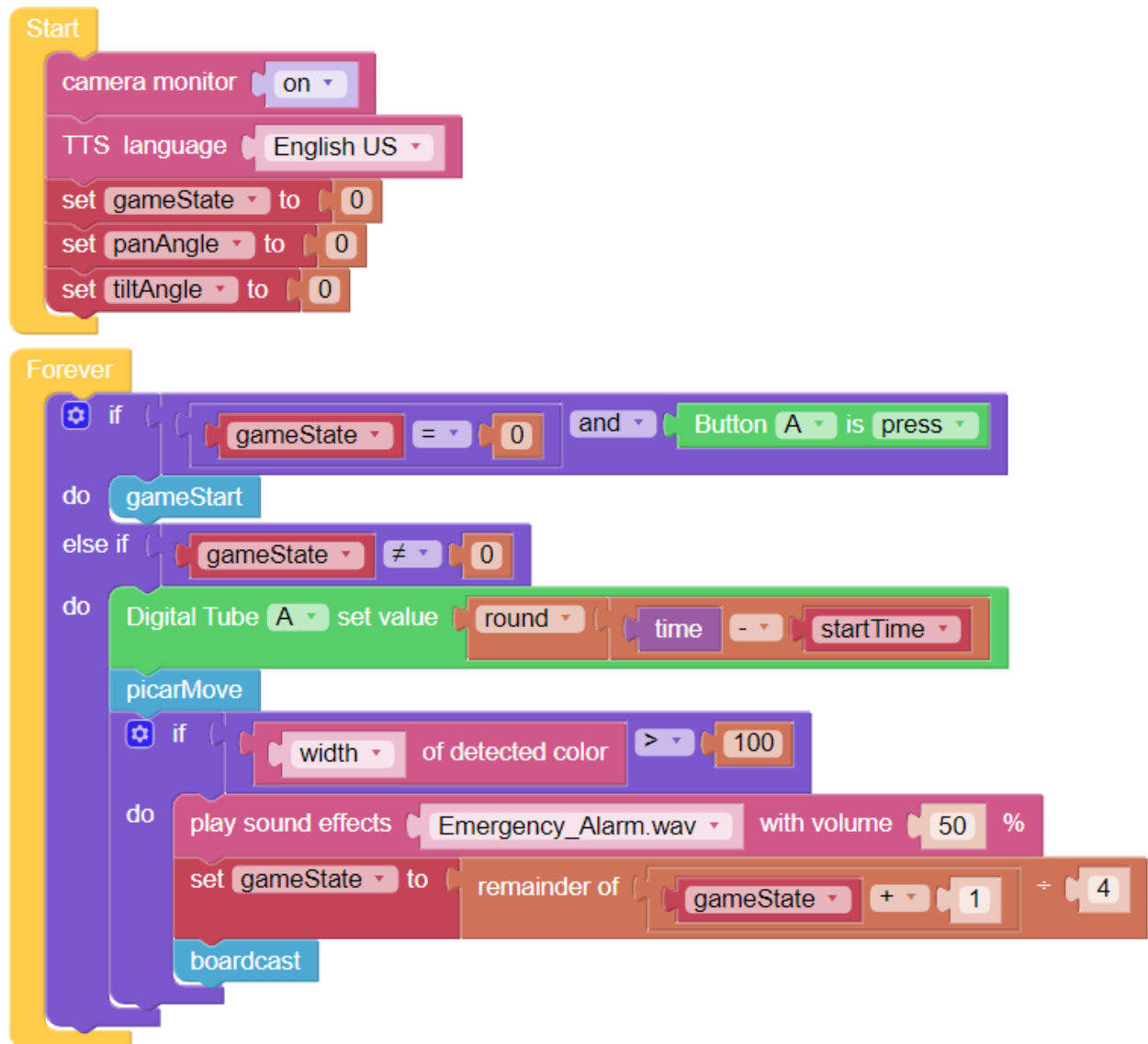


Take turns playing this game with friends to see who can help PiCar-X complete the objective the fastest!

EXAMPLE

Note:

- You can write the program according to the following picture, please refer to the tutorial: [How to Create a New Project?](#).
- Or find the code with the same name on the **Examples** page of the EzBlock Studio and click **Run** or **Edit** directly.



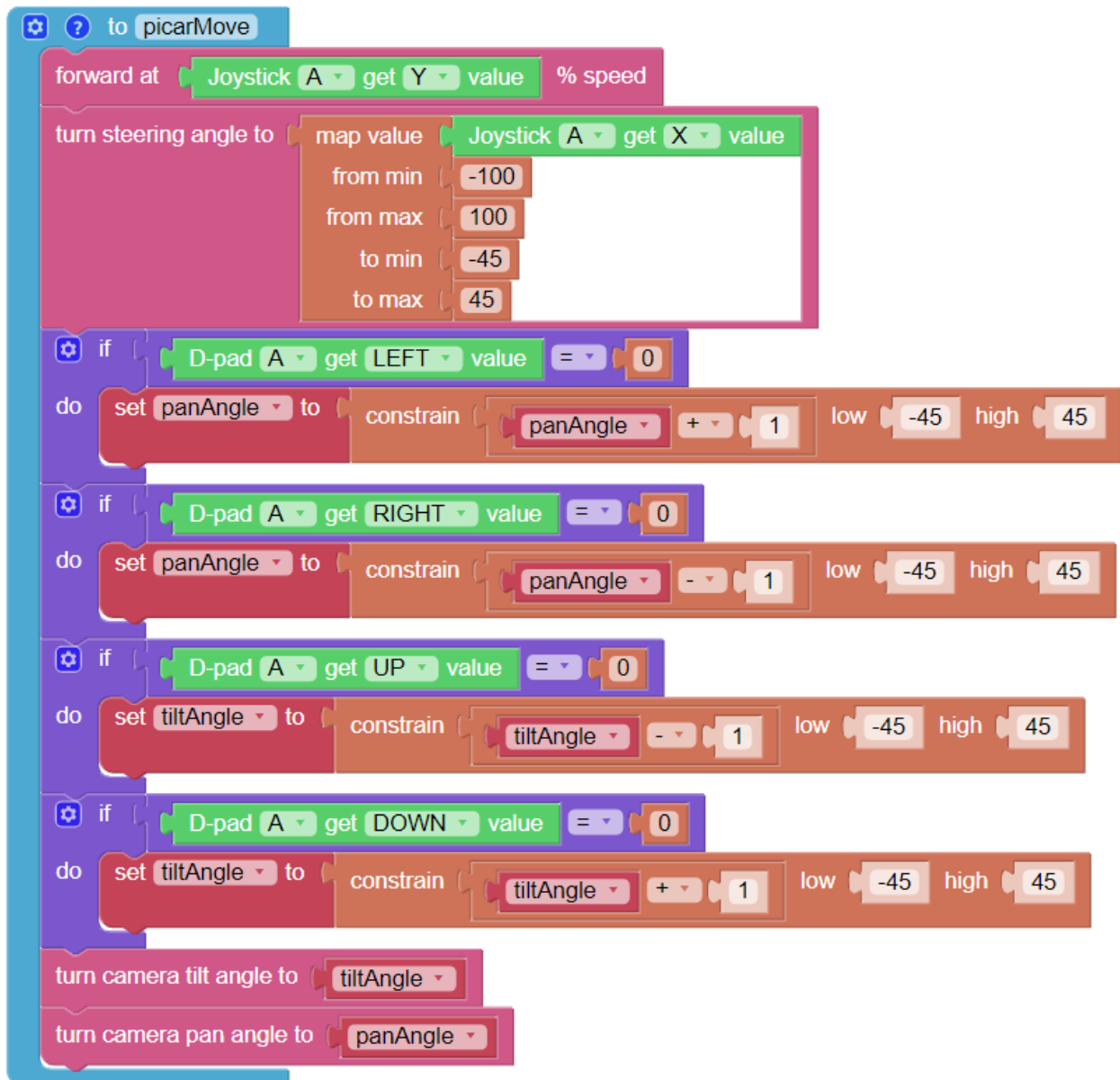
```

to 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
  broadcast

to broadcast
  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
  
```

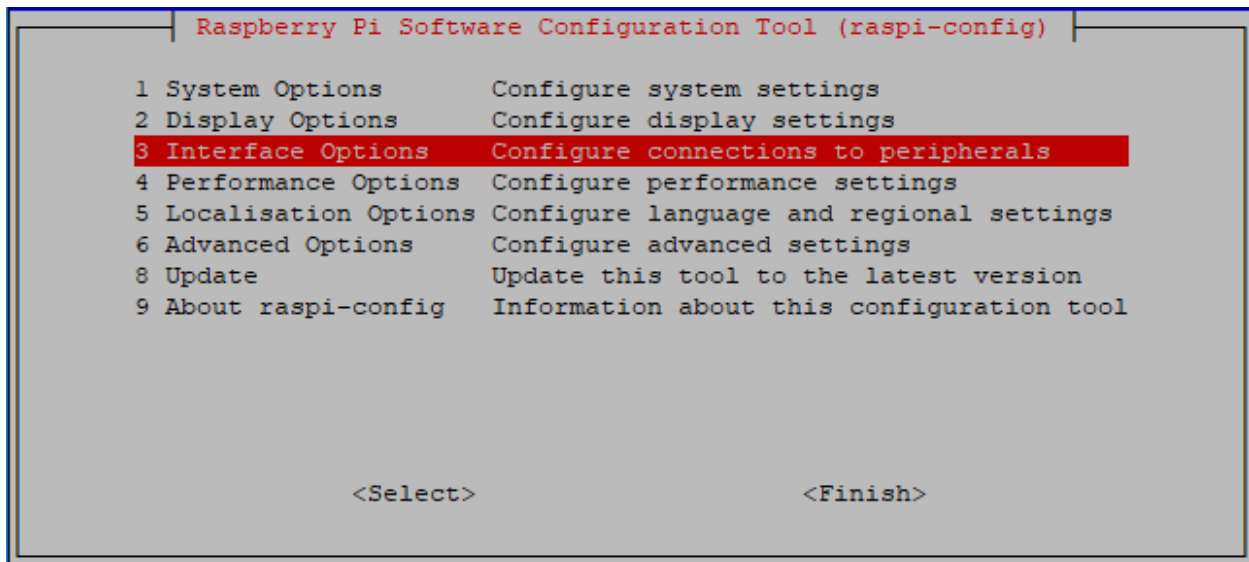


6.1 I2C Configuration

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

```
sudo raspi-config
```

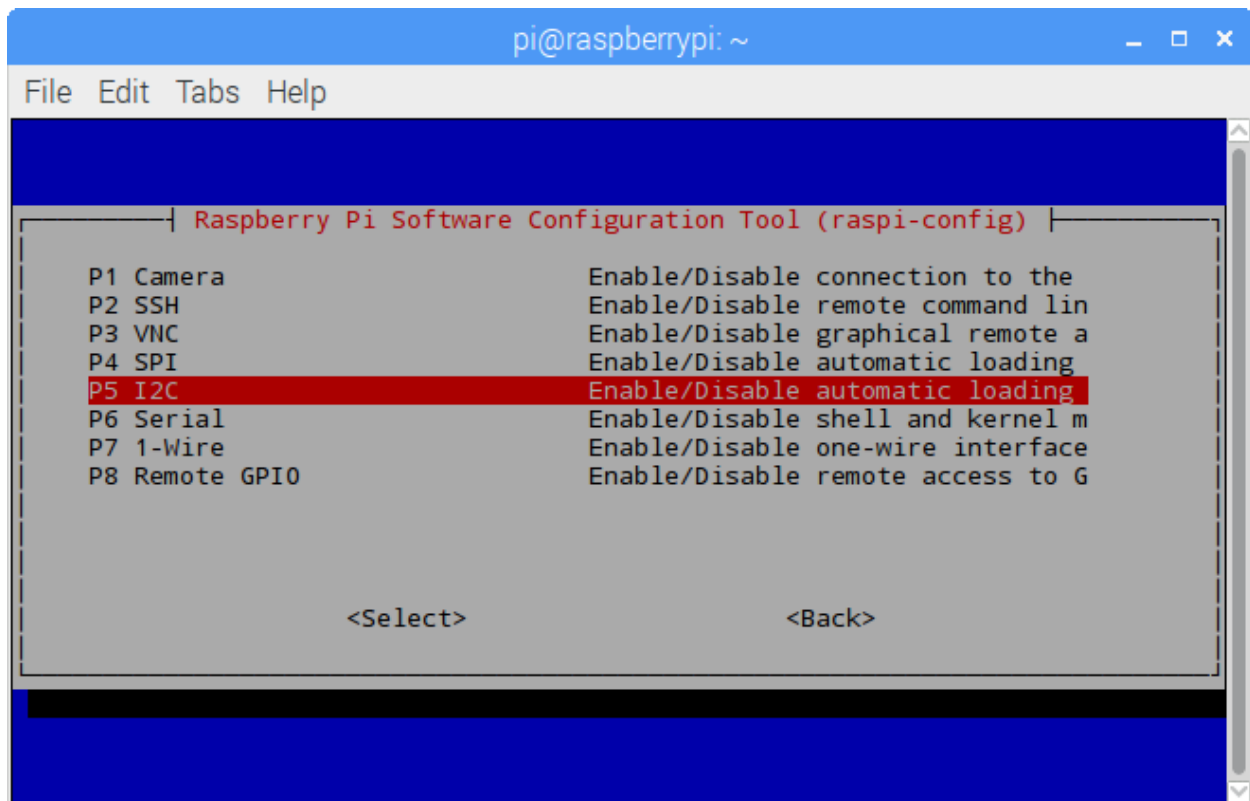
3 Interfacing options



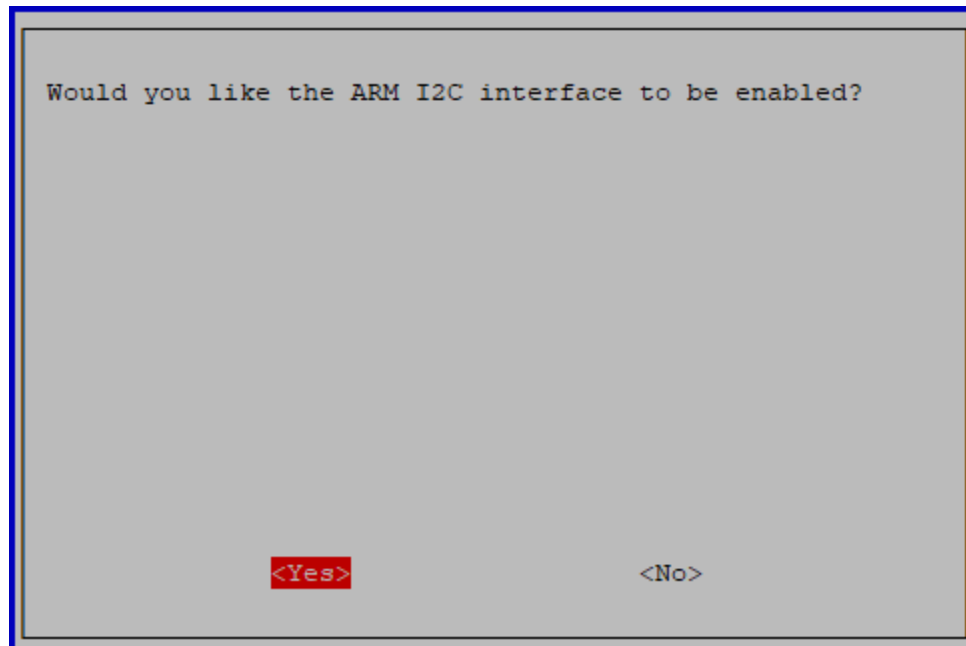
```
Raspberry Pi Software Configuration Tool (raspi-config)
1 System Options          Configure system settings
2 Display Options        Configure display settings
3 Interface Options       Configure connections to peripherals
4 Performance Options    Configure performance settings
5 Localisation Options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

<Select>                <Finish>
```

P5 I2C



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



6.2 Remote Desktop

There are two ways to control the desktop of the Raspberry Pi remotely: VNC and XRDP, you can use any of them.

6.2.1 VNC

You can use the function of remote desktop through VNC.

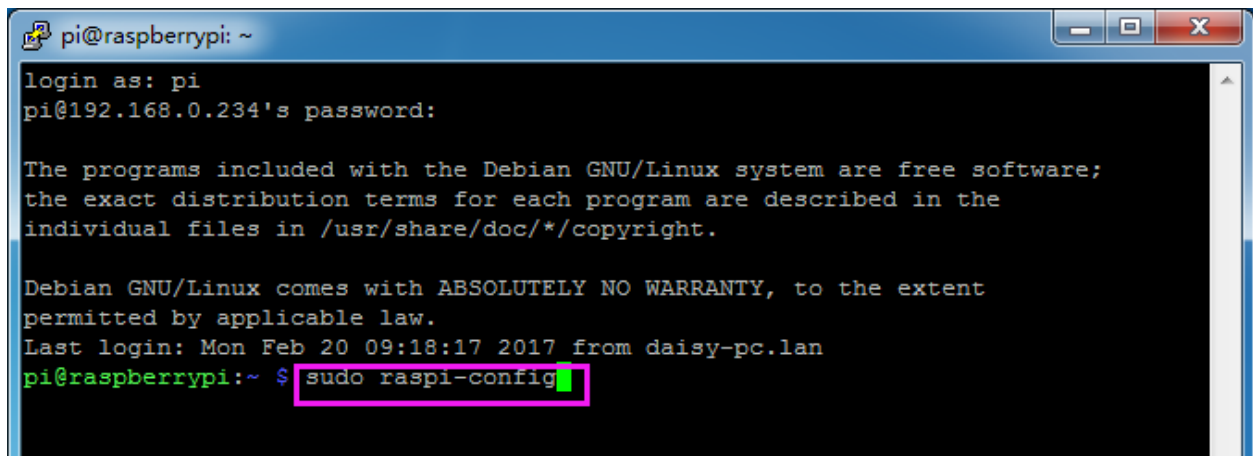
Enable VNC service

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

Step 1

Input the following command:

```
sudo raspi-config
```



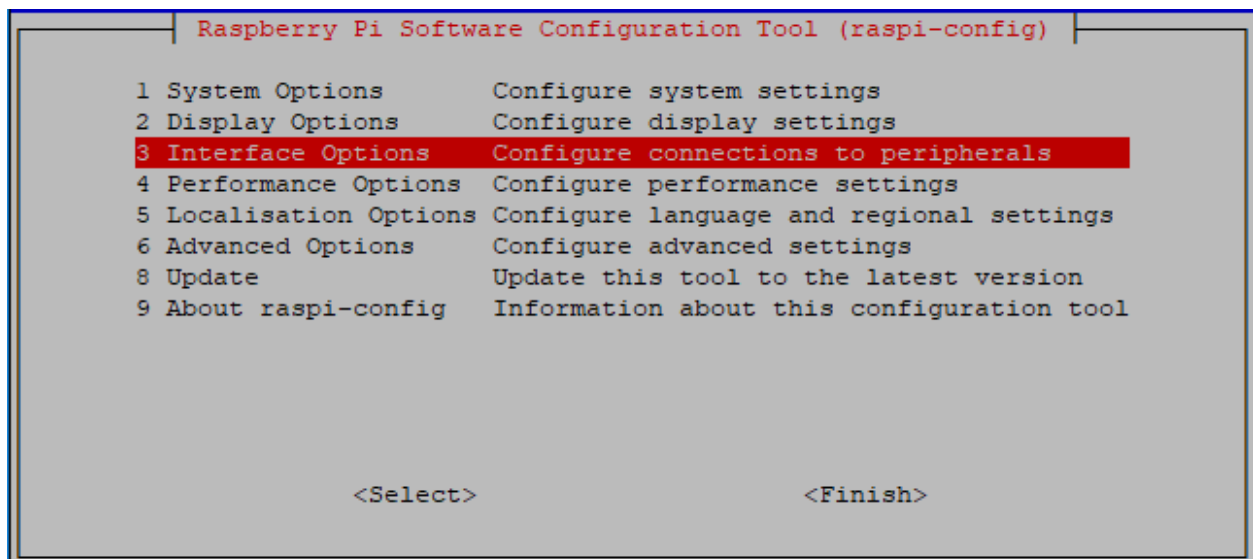
```
pi@raspberrypi: ~
login as: pi
pi@192.168.0.234's password:

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

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Feb 20 09:18:17 2017 from daisy-pc.lan
pi@raspberrypi:~ $ sudo raspi-config
```

Step 2

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



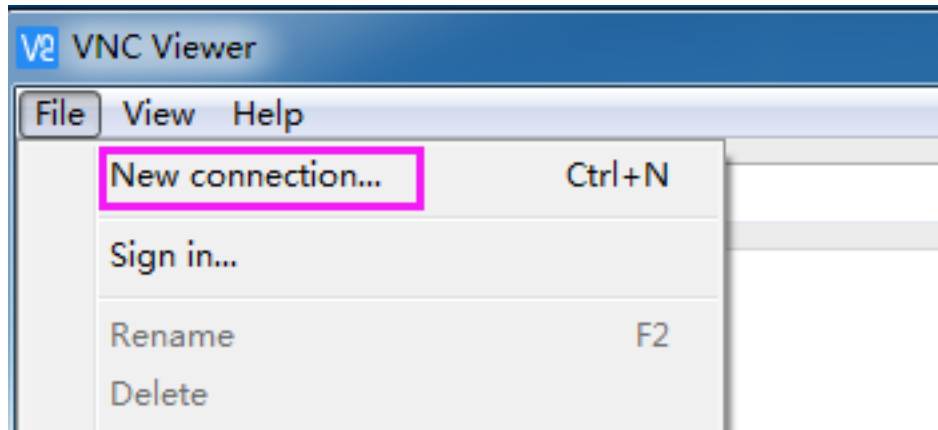
```
Raspberry Pi Software Configuration Tool (raspi-config)

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

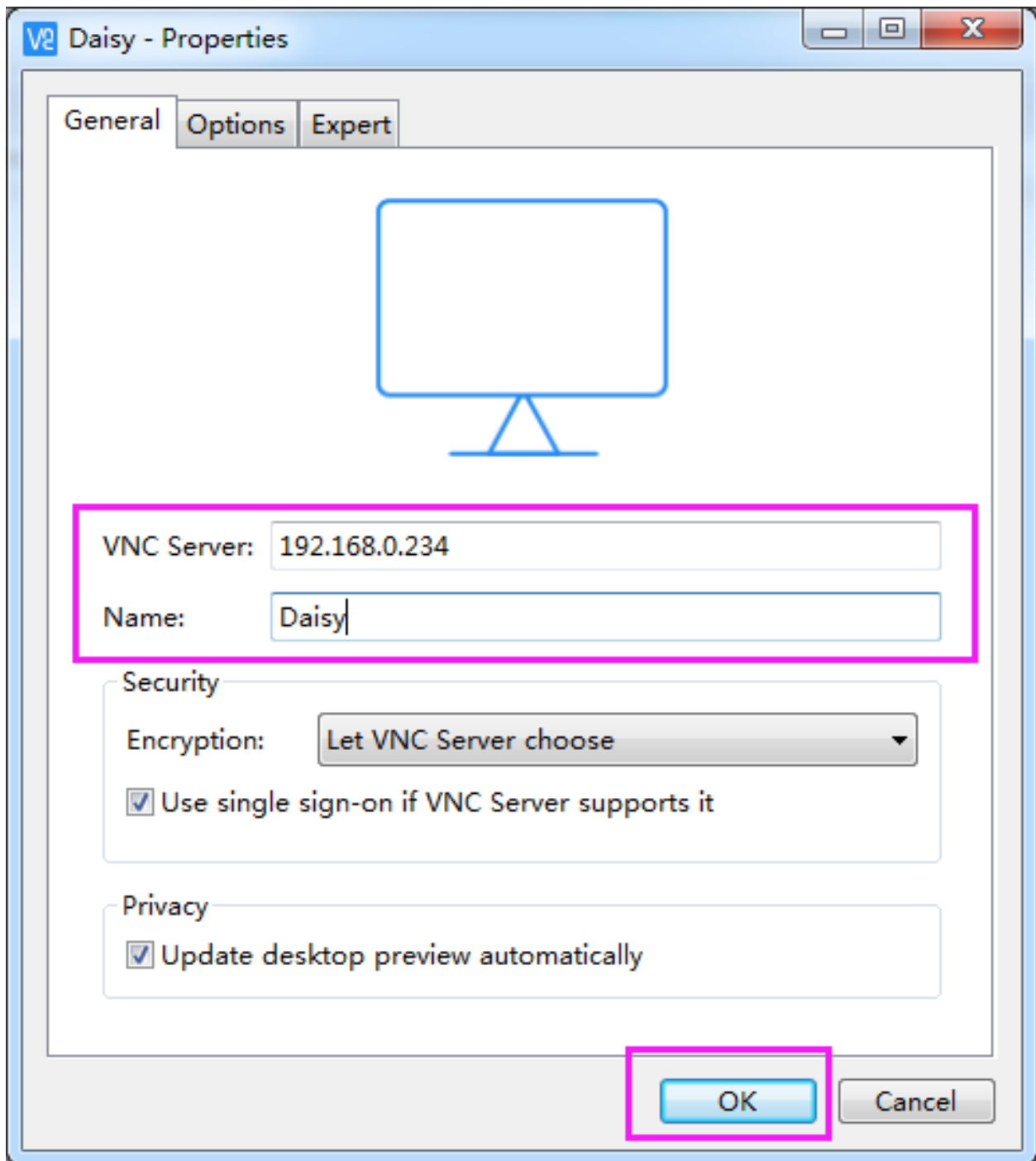
<Select>                <Finish>
```


Step 2

Then select “**New connection**”.

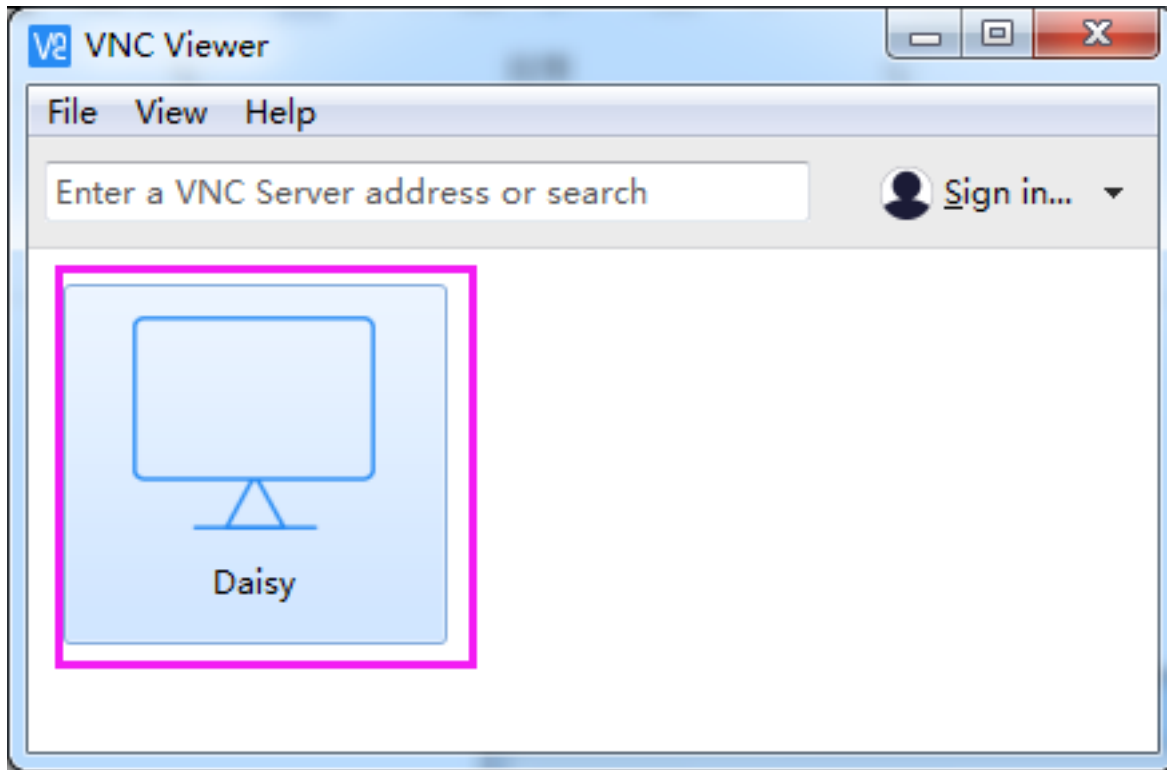
**Step 3**

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

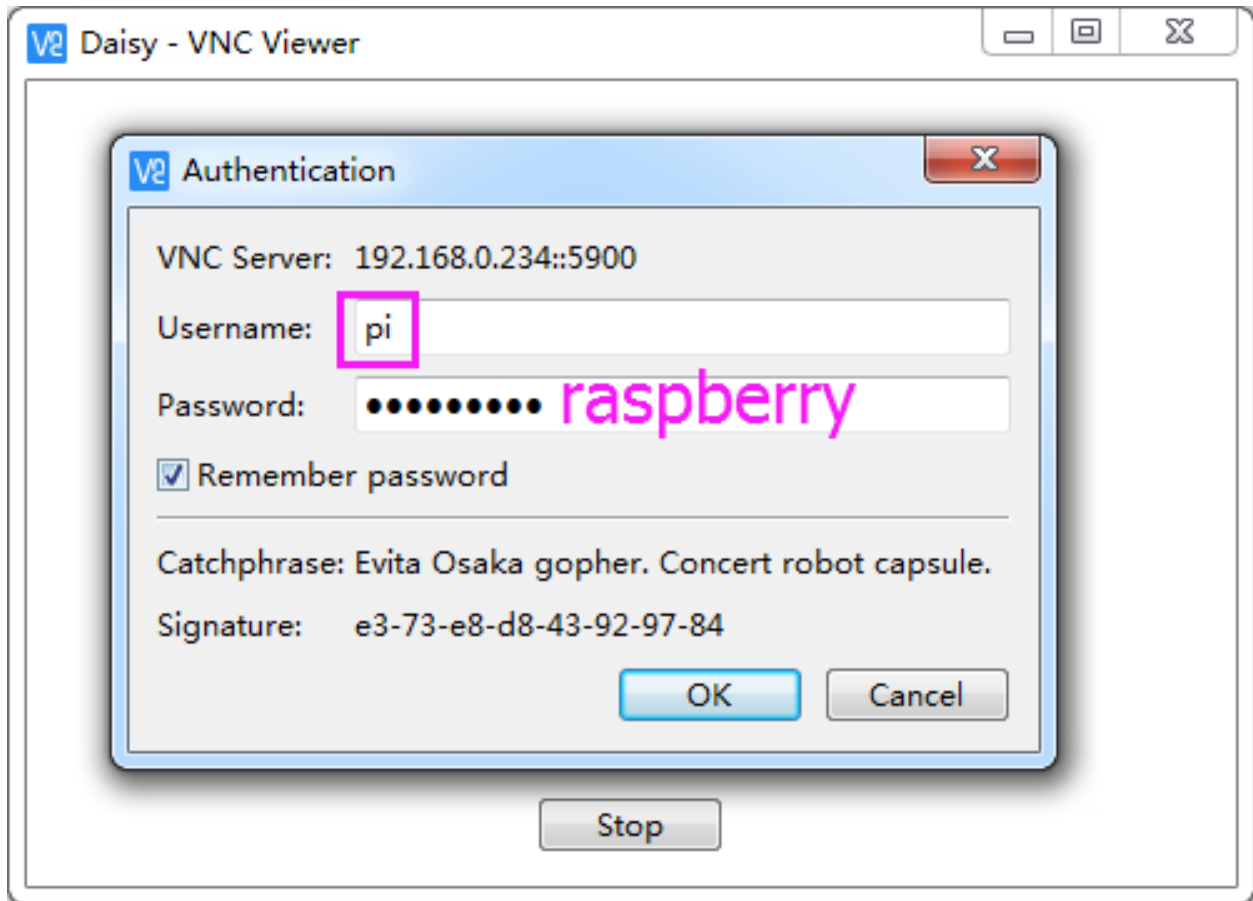


Step 4

Double click the **connection** just created:

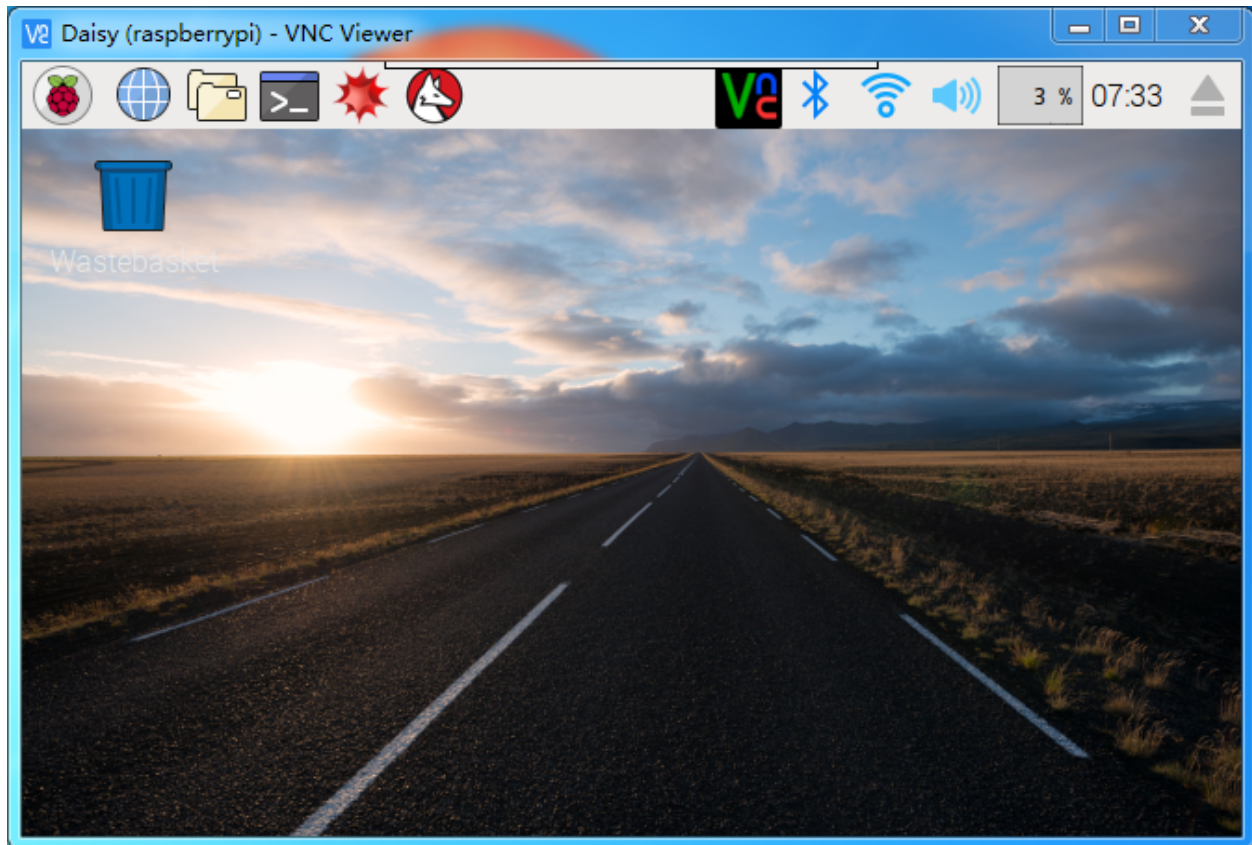
**Step 5**

Enter Username (**pi**) and Password (**raspberr**y by default).



Step 6

Now you can see the desktop of the Raspberry Pi:



That's the end of the VNC part.

6.2.2 XRDP

Another method of remote desktop is XRDP, it provides a graphical login to remote machines using RDP (Microsoft Remote Desktop Protocol).

Install XRDP

Step 1

Login to Raspberry Pi by using SSH.

Step 2

Input the following instructions to install XRDP.

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

Step 3

Later, the installation starts.

Enter “Y”, press key “Enter” to confirm.

```
pi@raspberrypi: ~  
pi@raspberrypi:~ $ sudo apt-get install xrdp  
Reading package lists... Done  
Building dependency tree  
Reading state information... Done  
The following extra packages will be installed:  
  vnc4server x11-apps x11-session-utils xbase-clients xbitmaps xfonts-base  
Suggested packages:  
  vnc-java mesa-utils x11-xfs-utils  
The following NEW packages will be installed:  
  vnc4server x11-apps x11-session-utils xbase-clients xbitmaps xfonts-base  
  xrdp  
0 upgraded, 7 newly installed, 0 to remove and 0 not upgraded.  
Need to get 8,468 kB of archives.  
After this operation, 17.1 MB of additional disk space will be used.  
Do you want to continue? [Y/n] y
```

Step 4

Finished the installation, you should login to your Raspberry Pi by using Windows remote desktop applications.

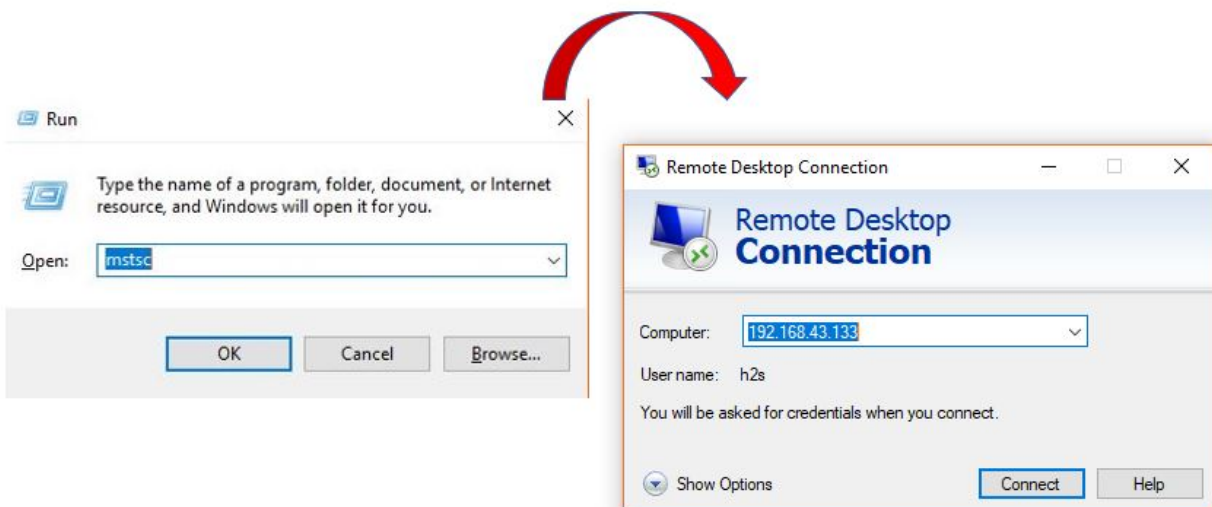
Login to XRDP

Step 1

If you are a Windows user, you can use the Remote Desktop feature that comes with Windows. If you are a Mac user, you can download and use Microsoft Remote Desktop from the APP Store, and there is not much difference between the two. The next example is Windows remote desktop.

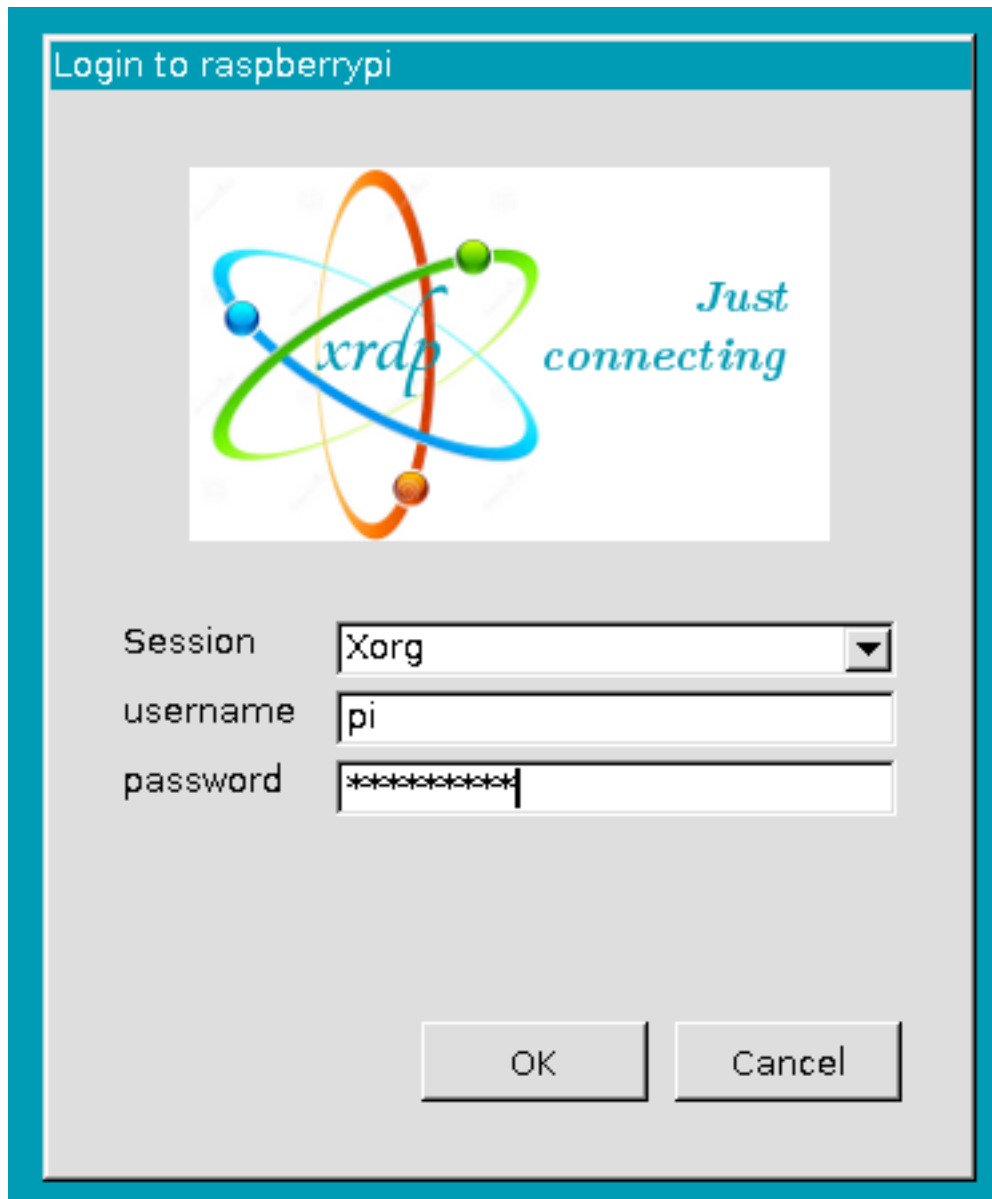
Step 2

Type in “mstsc” in Run (WIN+R) to open the Remote Desktop Connection, and input the IP address of Raspberry Pi, then click on “Connect”.



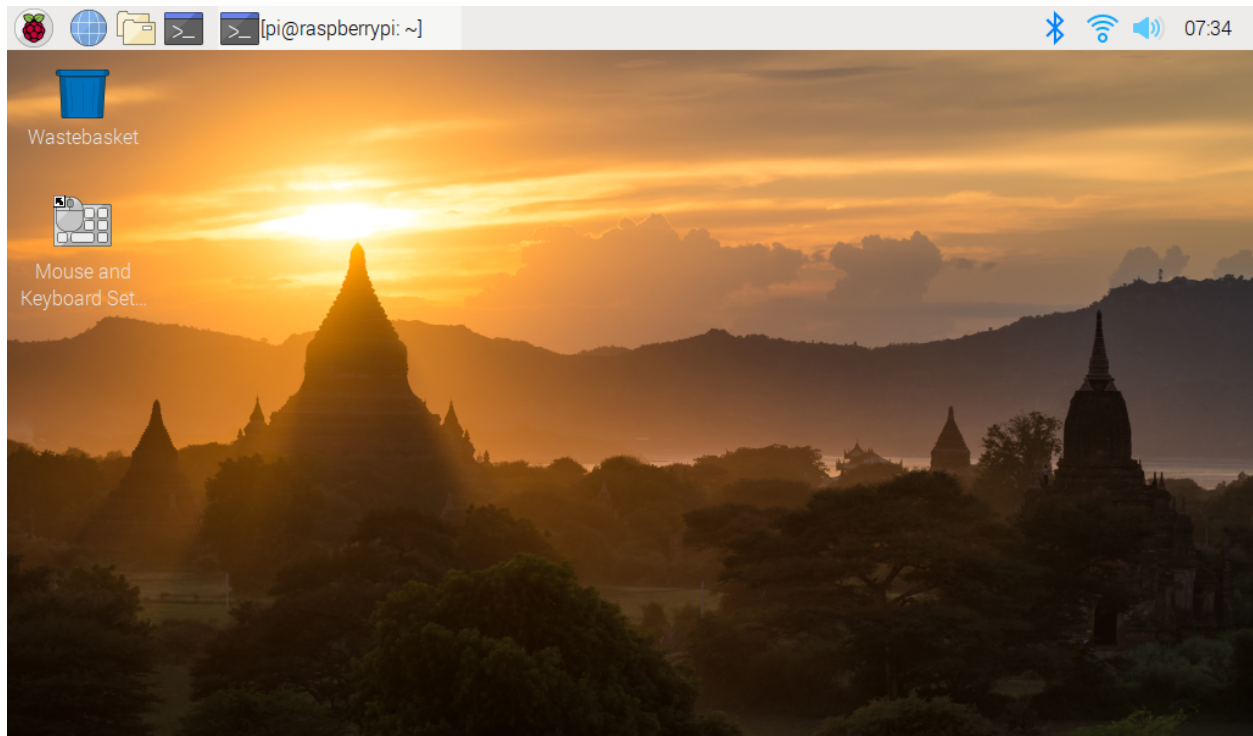
Step 3

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



Step 4

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



6.3 About the Battery

Applicable Parameters

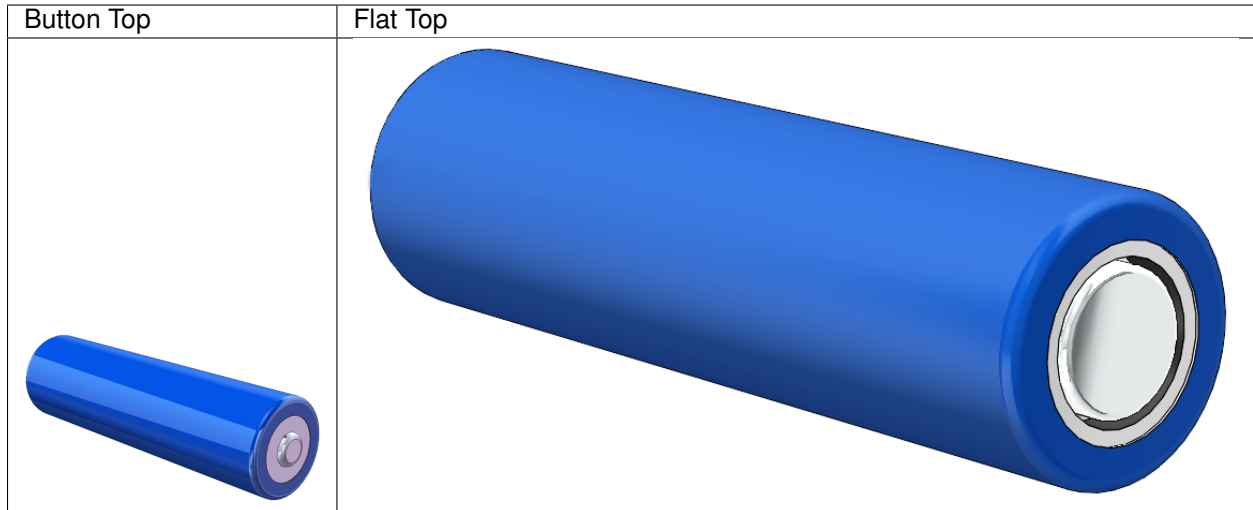
- 3.7V
- 18650
- Rechargeable
- Li-ion Battery
- Button Top
- No Protective Board

Note:

- Robot HAT cannot charge the battery, so you need to buy a battery charger.
 - When the two power indicators on the Robot HAT are off, it means the power is too low and the batteries need to be charged.
-

Button Top vs Flat Top?

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

**No protective board?**

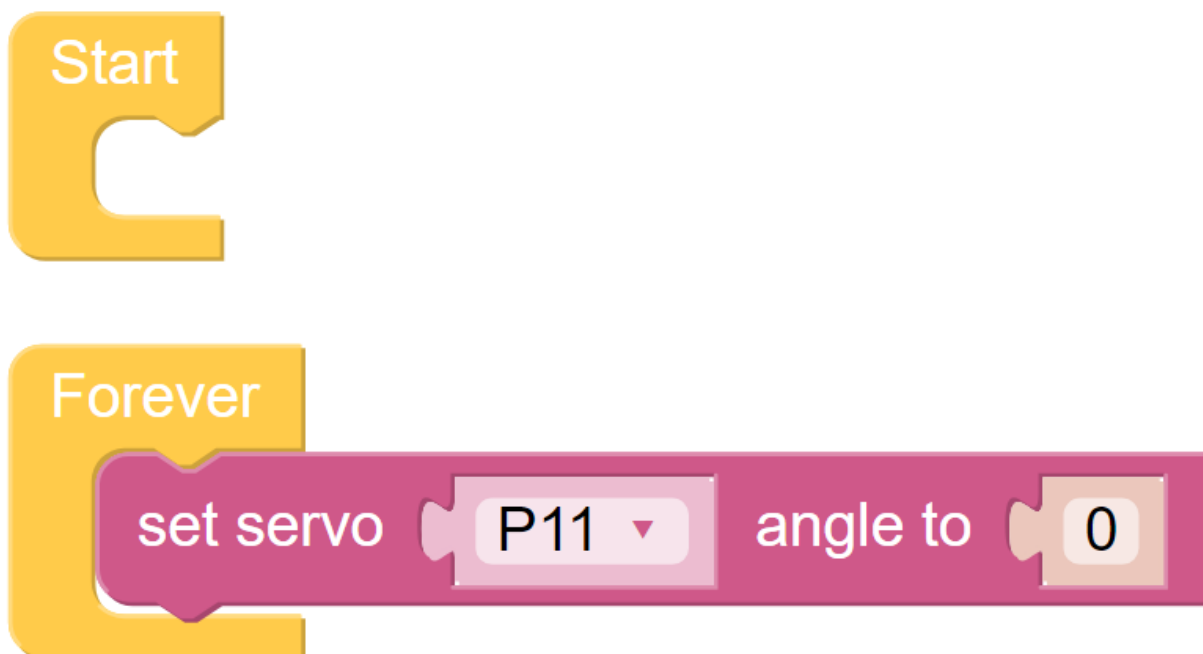
You are recommend to use 18650 batteries without a protective board. Otherwise, the robot may be cut power and stop running because of the overcurrent protection of the protective board.

Battery capacity?

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

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

- 1) Check if the servo cable is properly connected and if the Robot HAT power is on.
- 2) Press Reset button.
- 3) If you have already run the program in Ezblock Studio, the custom program for P11 is no longer available. You can refer to the picture below to manually write a program in Ezblock Studio to set the servo angle to 0.



Q2: When using VNC, I am prompted that the desktop cannot be displayed at the moment?

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

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

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

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

THANK YOU

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

Particular Thanks

- Len Davisson
- Kalen Daniel
- Juan Delacosta

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

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

COPYRIGHT NOTICE

All contents including but not limited to texts, images, and code in this manual are owned by the SunFounder Company. You should only use it for personal study, investigation, enjoyment, or other non-commercial or nonprofit purposes, under the related regulations and copyrights laws, without infringing the legal rights of the author and relevant right holders. For any individual or organization that uses these for commercial profit without permission, the Company reserves the right to take legal action.