
SunFounder picar-x

www.sunfounder.com

Dec 29, 2021

CONTENTS

1	Introduction	3
1.1	The History of Self-driving Cars	3
1.2	About PiCar-X	5
1.3	Deep Learning and Neural Networks	6
2	Component List and Assembly Instructions	7
3	About Robot HAT	9
4	Play with Ezblock	11
4.1	Quick Guide on Ezblock	11
4.1.1	Before Assembling PiCar-X	11
4.1.2	Before Programming With Ezblock	13
4.2	Projects	13
4.2.1	Move	14
4.2.2	Remote Control	17
4.2.3	Test Ultrasonic Module	19
4.2.4	Test Grayscale Module	20
4.2.5	Color Detection	21
4.2.6	Face Detection	24
4.2.7	Sound Effect	27
4.2.8	Background Music	29
4.2.9	Say Hello	30
4.2.10	Music Car	33
4.2.11	Cliff Detection	34
4.2.12	Minecart	35
4.2.13	Minecart Plus	38
4.2.14	Bullfight	43
4.2.15	Beware of Pedestrians	45
4.2.16	Traffic Sign Detection	47
4.2.17	Orienteering	52
5	Play with Python	59
5.1	Quick Guide on Python	59
5.1.1	What Do We Need?	59
5.1.2	Installing the OS	61
5.1.3	Set up Your Raspberry Pi	68
5.1.4	Download and Run the Code	74
5.2	Let PiCar-X Move	75
5.3	Obstacle Avoidance	78

5.4	Line Tracking	80
5.5	Text to Speech	81
5.6	Computer Vision	83
5.7	Color Detection	86
5.8	Face Detection	89
5.9	Video Car	93
6	Appendix	97
6.1	I2C Configuration	97
6.2	Remote Desktop	99
6.2.1	VNC	99
6.2.2	XRDP	105
6.3	About the Battery	108
7	FAQ	111
8	Thank You	113
9	Copyright Notice	115

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.

Included in this tutorial are a component list, an assembly diagram, an introduction to SunFounders Robot HAT, and programming tutorials for the PiCar-X in both Blockly-based and Python languages.

The programming sections are divided into two chapters: *Play with Ezblock* and *Play with Python*. Each programming chapter allows users to get started by making the PiCar-X perform basic movements, and proceed up to designing multiplayer games and beyond!

Ezblock Studio is a software development platform created by SunFounder for helping beginners and novices get started with programming on a Raspberry Pi platform. Ezblock Studio has two programming language environments: a Graphical User Interface environment, and a command-line Python environment.

Ezblock is available for nearly all types of devices, including Mac, PC, and Android. Ezblock has recently been upgraded to version 3.0, which uses WebSocket to provide Bluetooth and Wi-Fi support.

For users who only want to learn programming in Python 3.0, also included is an introduction to the fundamentals of the Raspberry Pi Operating System, and tutorials that provide basic Python programming skills.

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.

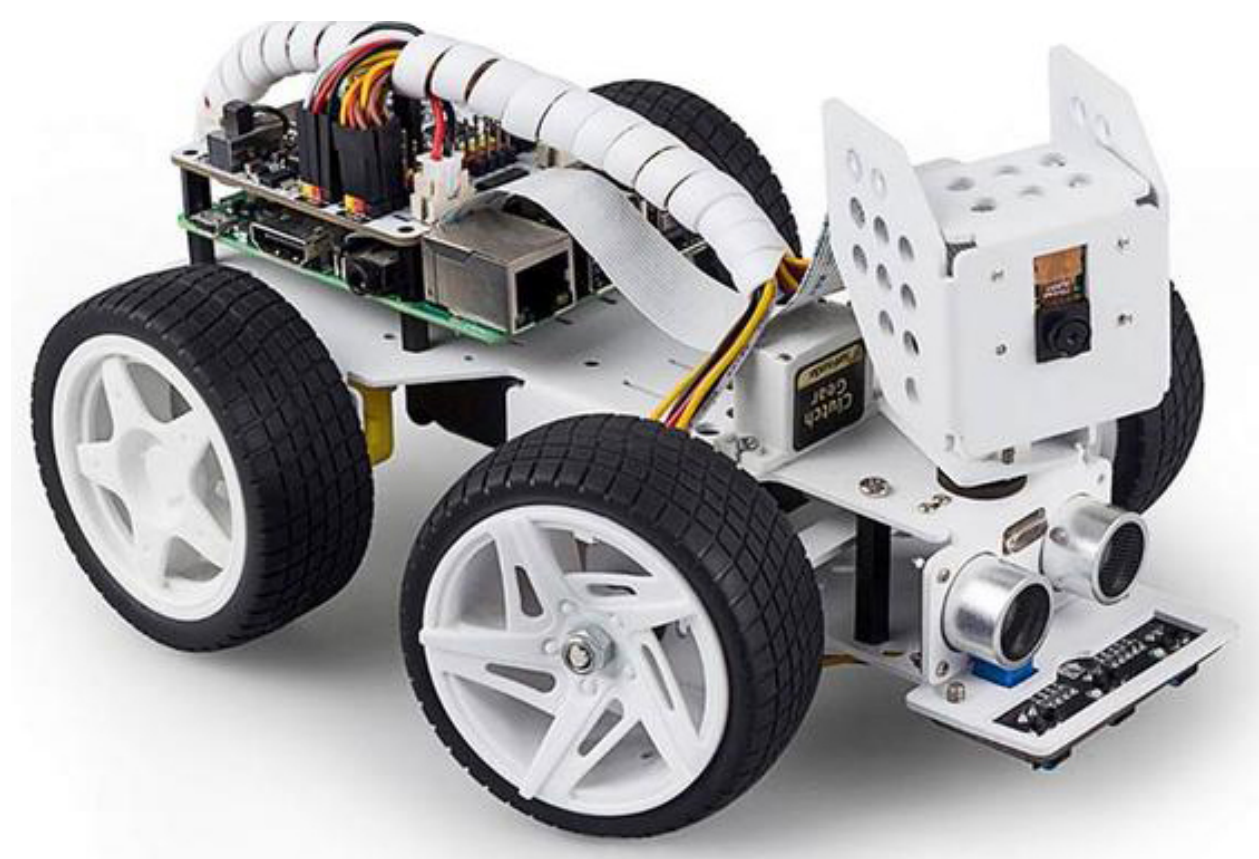
According to **RAND's report** , reaching the appropriate level of autonomous learning requires training data from hundreds of millions, or even hundreds of billions of miles to establish a level of reliability.

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.

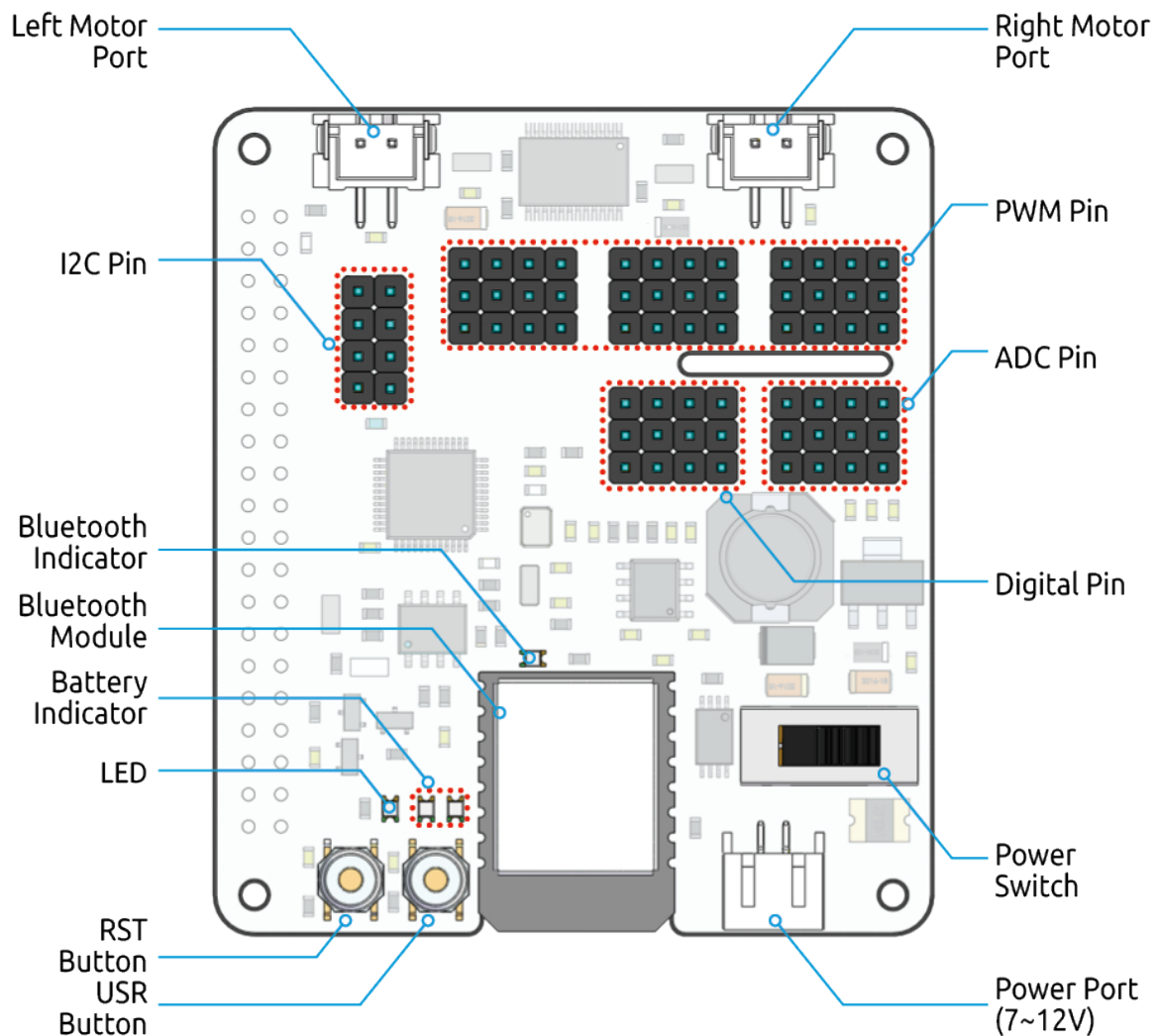
Please follow the steps on the following PDF for assembly instructions: [PDF]Component List and Assembly of PiCar-X.

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.

PLAY WITH EZBLOCK

Ezblock is a software development platform created by SunFounder for helping beginners and novices get started with programming on a Raspberry Pi platform. Ezblock has two programming language environments: a Graphical User Interface environment, and a command-line Python environment.

Ezblock is available for nearly all types of devices, including Mac, PC, and Android. Ezblock has recently been upgraded to version 3.0, which uses WebSocket to provide Bluetooth and Wi-Fi support.

Ezblock 3.0 has been engineered to optimize remote connections, provide an intuitive user-interface, and to remotely control a Raspberry Pi through Ezblock Studio.

4.1 Quick Guide on Ezblock

There are 2 parts here:

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

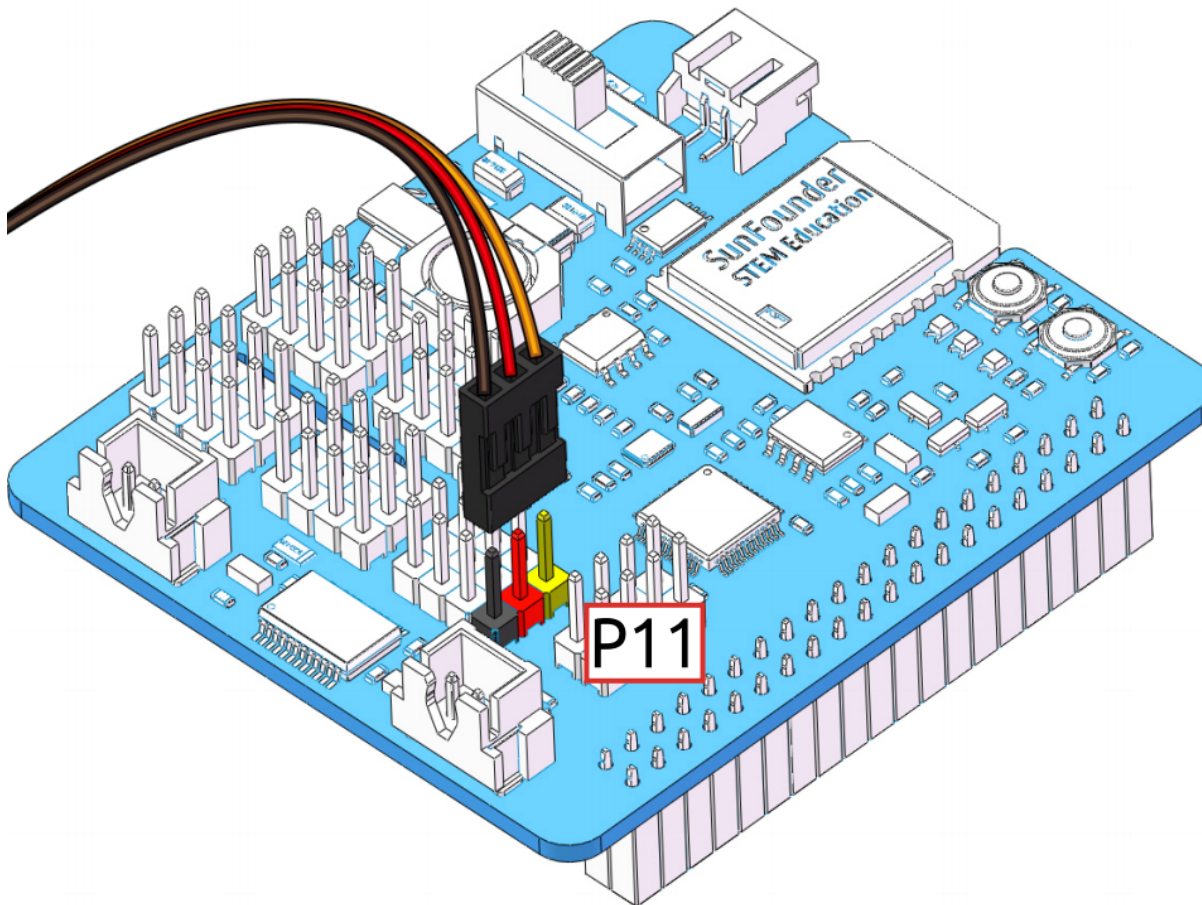
4.1.1 Before Assembling PiCar-X

Before assembling the PiCar-X, follow the instructions on how to install the Ezblock OS on an Micro SD card here: [Download and Install Ezblock OS](#).

After burning the Ezblock system on the Micro SD card, the P11 port on the Robot HAT is set to calibrate the servo angle to a 0° angle. To make sure the servo has been correctly set to 0°, first gently insert a rocker arm in the servo shaft, then slightly rotate the rocker arm to a different angle.



Next, insert the servo cable into the P11 port as shown below:



Turn on the **Power Switch** to the Robot HAT, and the servo arm should return to the 0° position. If the servo arm does not return to 0°, press the **RST** button to restart the Robot HAT.

Note: Before attaching each servo, plug the servo cable into P11 and turn the power on to set the servo to 0°. This function will become invalid after writing any programs to the Micro SD card.

4.1.2 Before Programming With Ezblock

First download Ezblock Studio 2, and then manually upgrade to Ezblock Studio 3 to begin programming.

For a detailed installation tutorial, please refer to: [Install Ezblock Studio](#).

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.

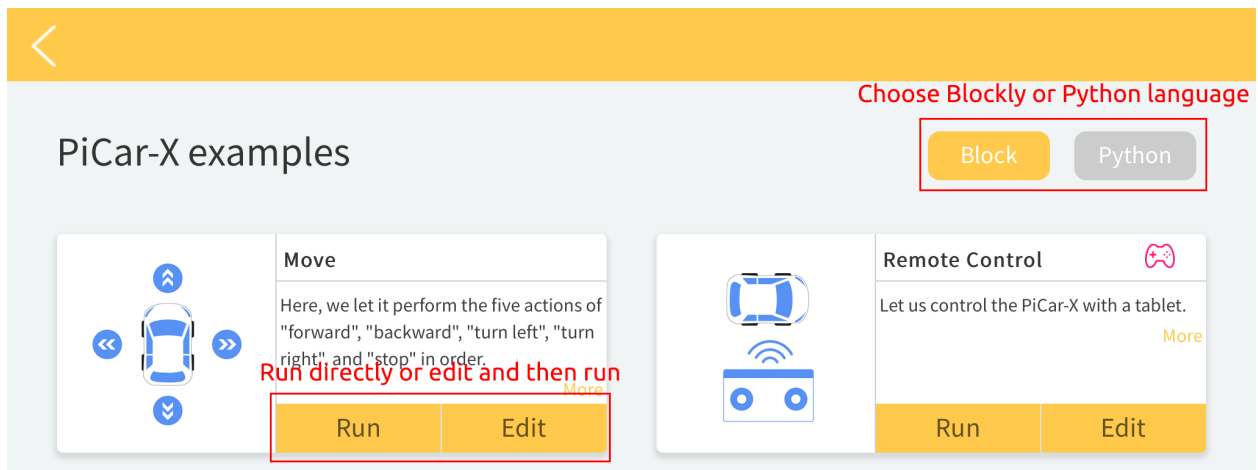
But if you think the assembly is perfect and no calibration is needed, you can also skip this step.

4.2 Projects

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

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

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



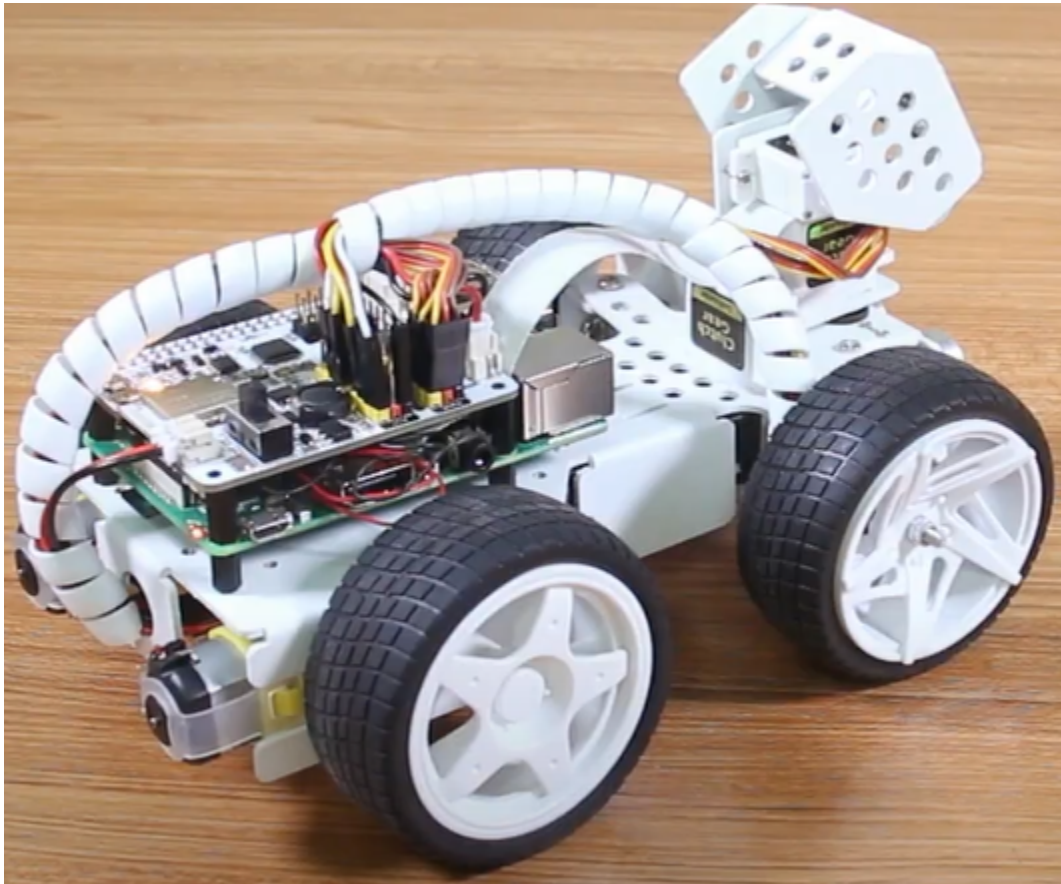
Basic

4.2.1 Move

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

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

- [Quick User Guide for Ezblock 3](#)
- [How to Create a New Project?](#)



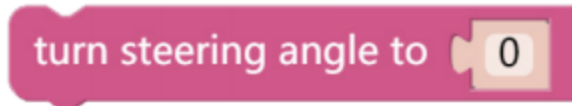
TIPS

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.



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.

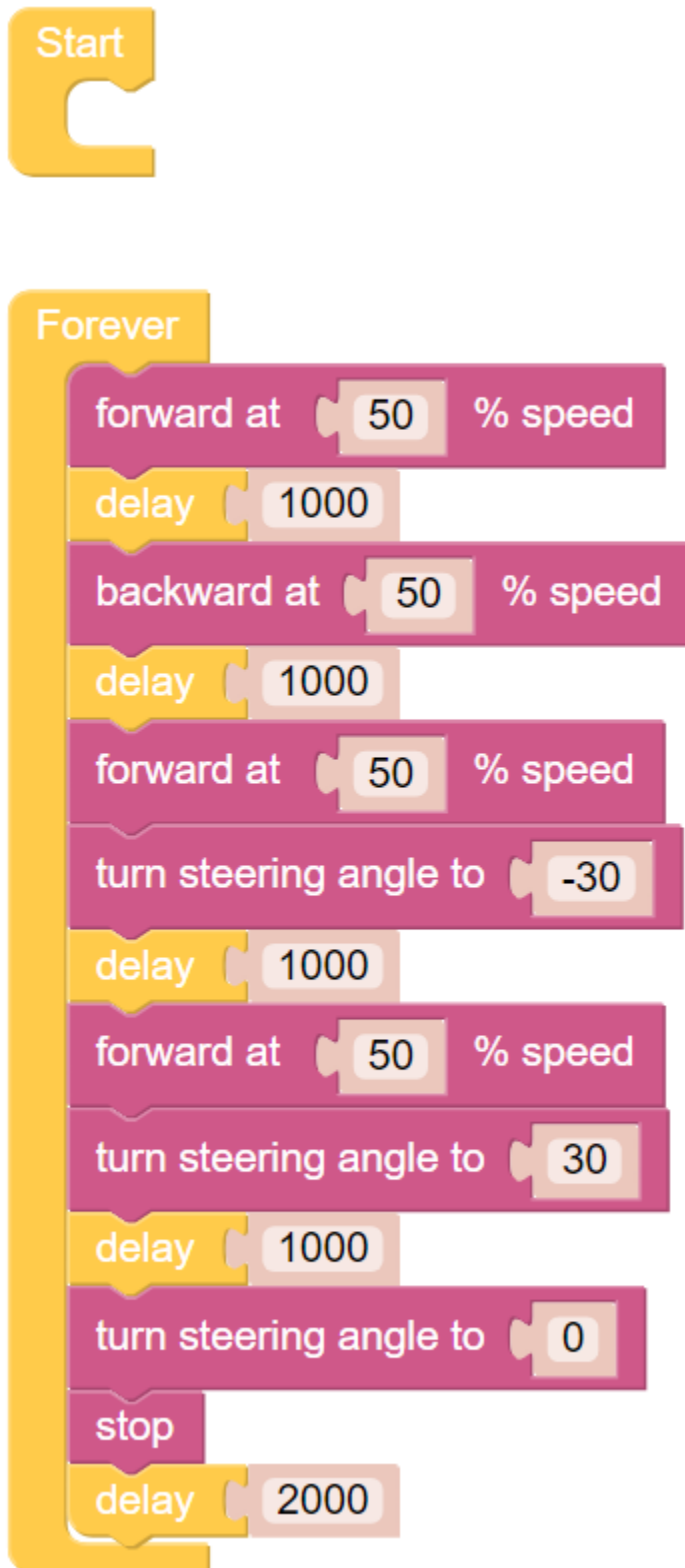


This block will cause a timed break between commands, based on milliseconds. In the example below, the PiCar-X will wait for 1 second (1000 milliseconds) before executing the next command.



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

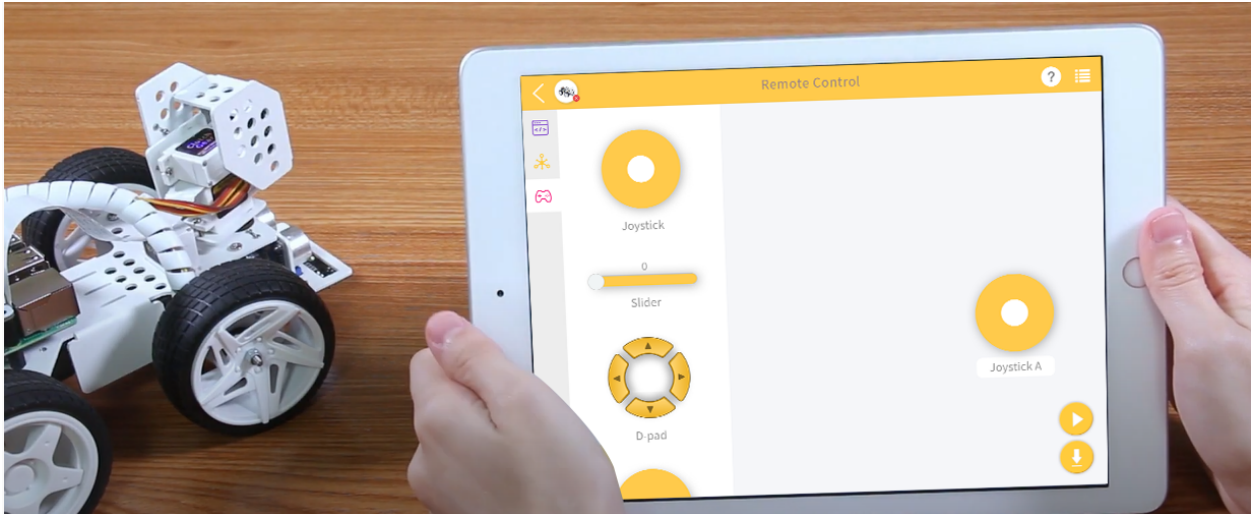
EXAMPLE



4.2.2 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:

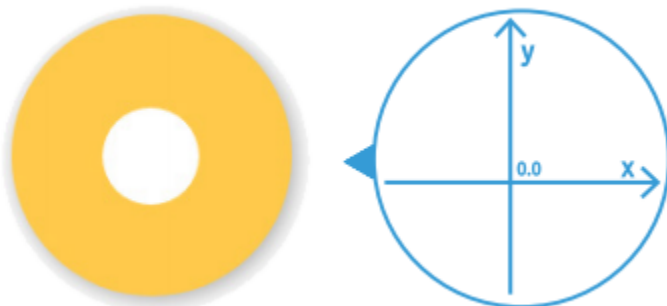
- [How to Use the Remote Control Function?](#)



TIPS



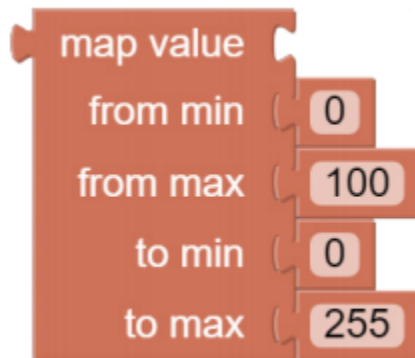
To use the remote control function, open the Remote Control page from the left side of the main page.



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”.

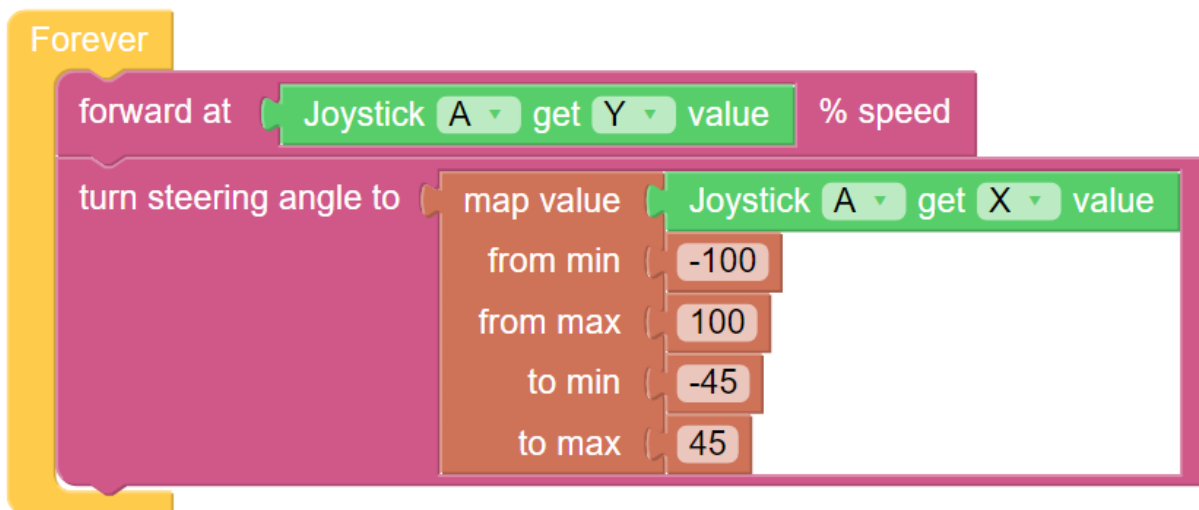


After dragging and dropping a widget on the remote control page, a new category-Remote with the above block will appear. This block reads the Joystick value in the Remote Control page. You can click the drop-down menu to switch to the Y-axis reading.



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".

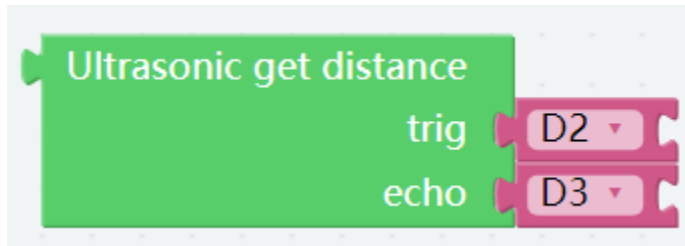
EXAMPLE



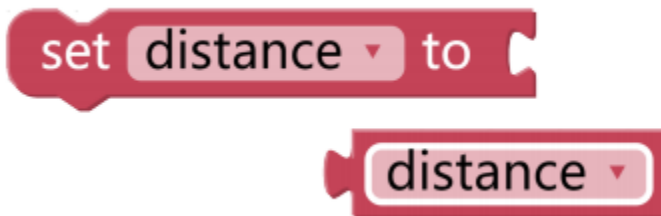
4.2.3 Test Ultrasonic Module

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.



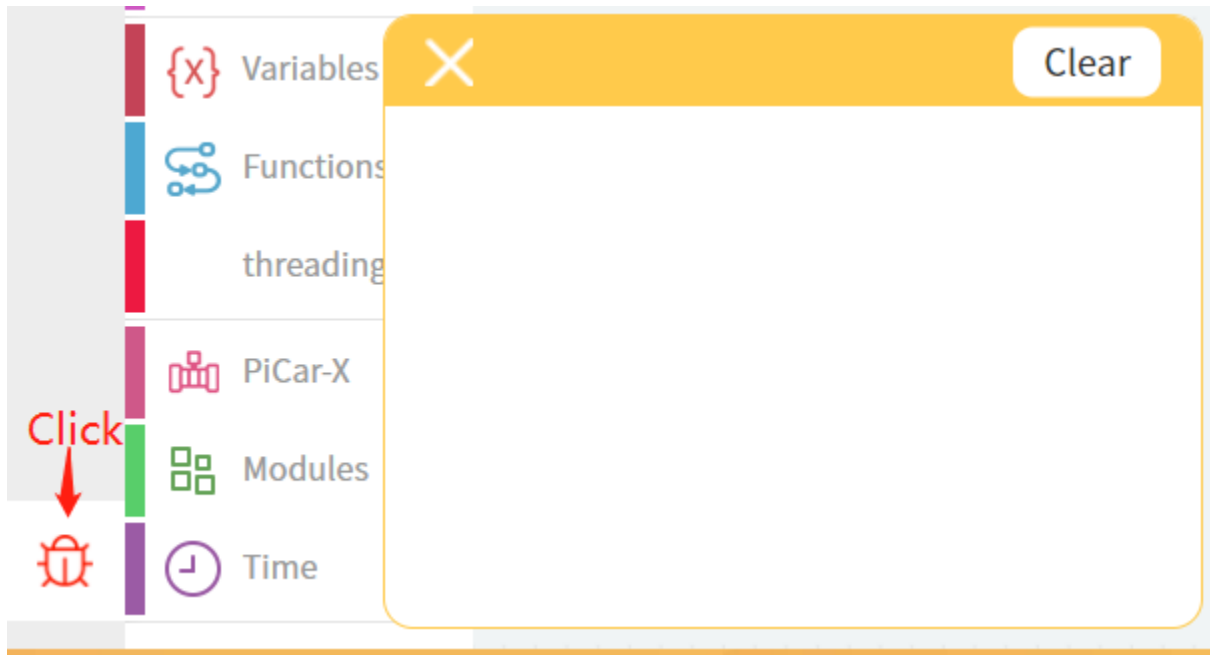
This program is simplified with a **Variable**. For example, when there are multiple functions in a program that each need to use the distance to an obstacle, a **Variable** can be used to report the same distance value to each function, instead of each function reading the same value separately.

Create variable...

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

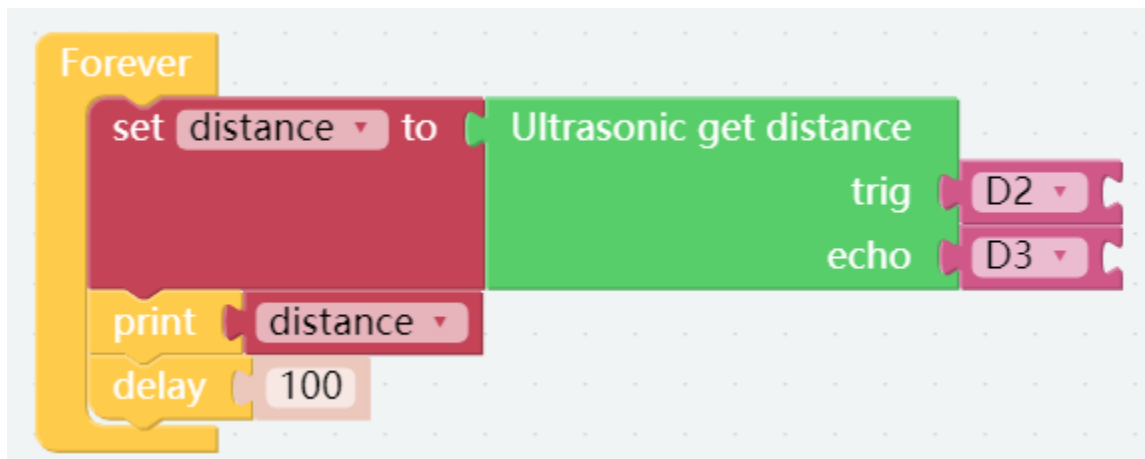


The **Print** function can print data such as variables and text for easy debugging.



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

EXAMPLE



4.2.4 Test Grayscale Module

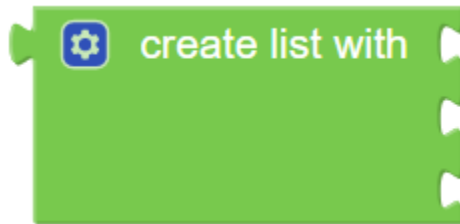
PiCar-X includes a Grayscale module for implementing line-following, cliff detection, and other fun experiments. The Grayscale module has three detection sensors that will each report a value according to the shade of color detected by the sensor. For example, a sensor reading the shade of pure black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

TIPS



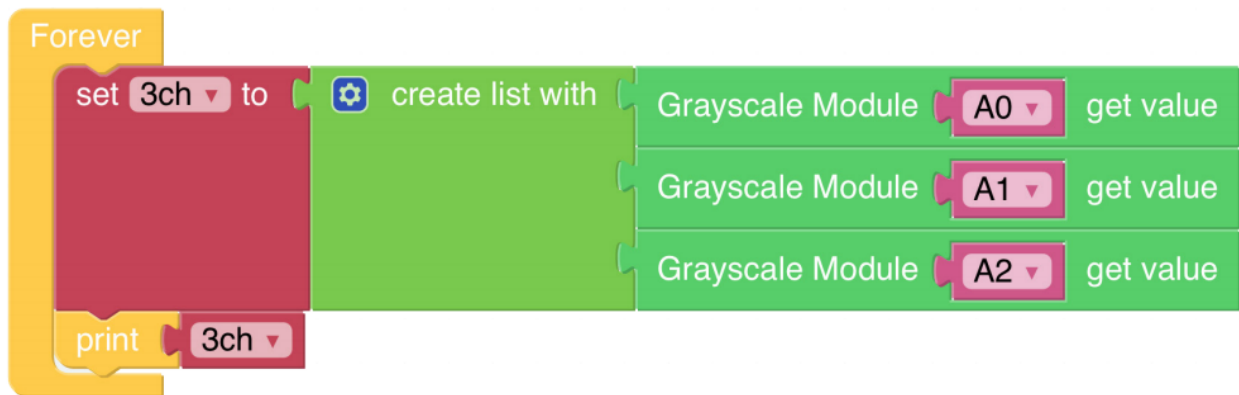
Use the **Grayscale module** block to read the value of one of the sensors. In the example above, the “A0” sensor is the

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



4.2.5 Color Detection

PiCar-X is a self-driving car with a built-in camera, which allows Ezbloc programs to utilize object detection and color recognition code. In this section, Ezbloc 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](#)

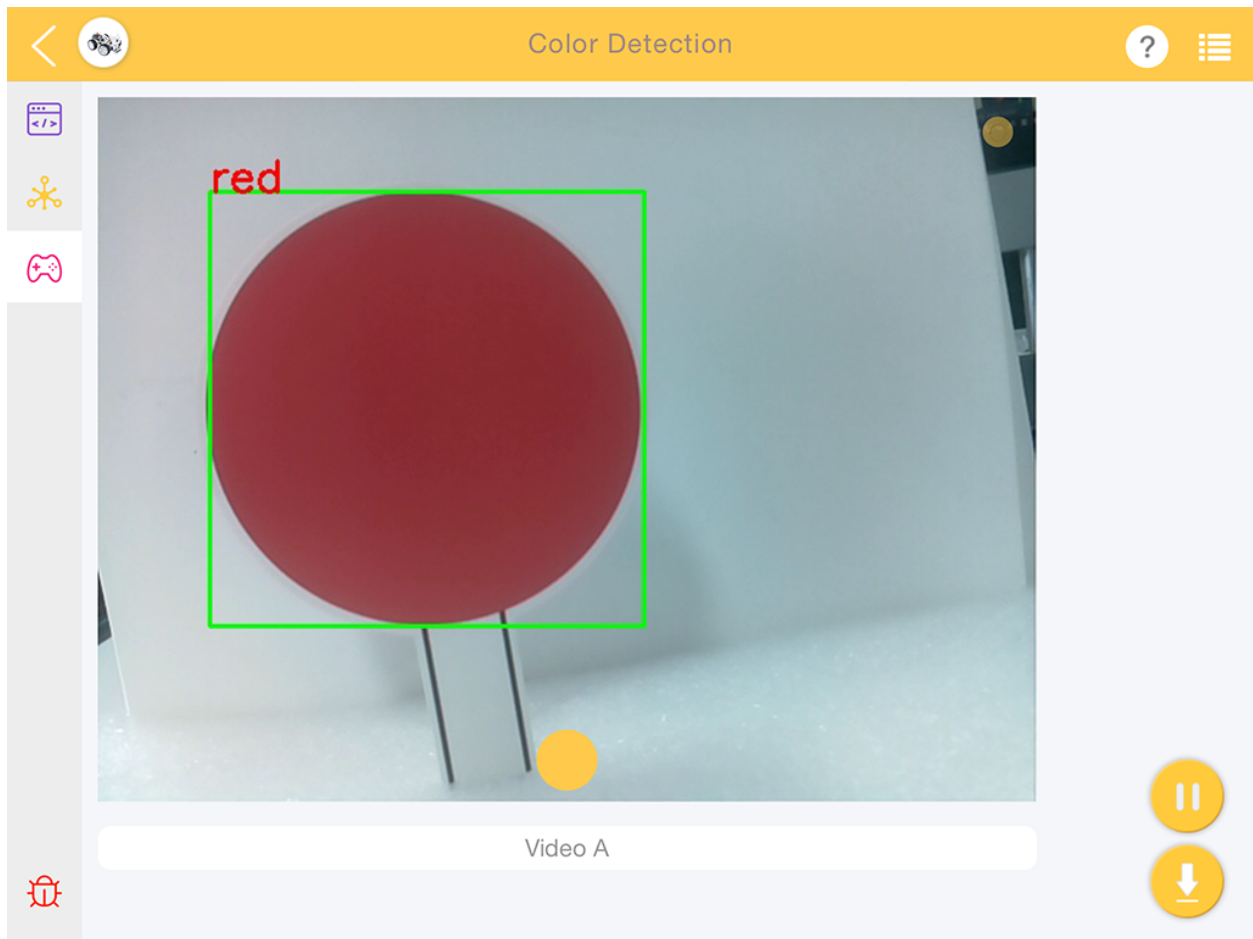
In this program, Ezbloc will first be told the Hue-Saturation-Value (HSV) space range of the color to be detected, then utilize OpenCV to process the colors in the HSV range to remove the background noise, and finally, box the matching color.

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

- [PDF]Color Cards



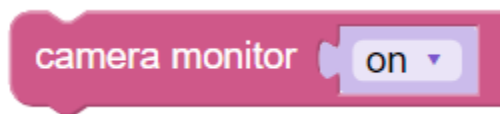
Note: The printed colors may have a slightly different hue from the Ezblock color models due to printer toner differences, or the printed medium, such as a tan-colored paper. This can cause a less accurate color recognition.



TIPS



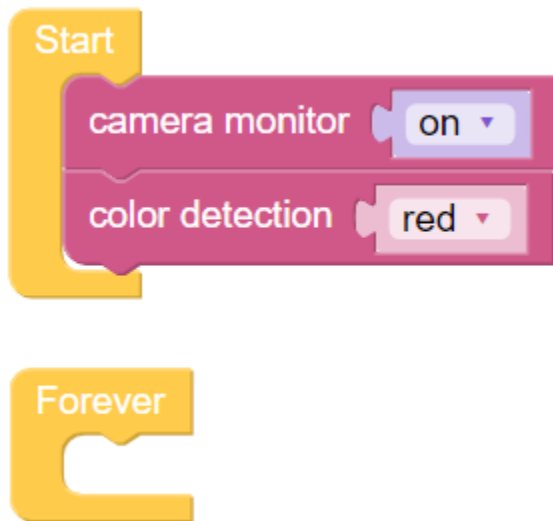
Drag the Video widget from the remote Control page, and it will generate a video monitor. For more information on how to use the Video widget, please reference the tutorial on Ezblock video here: [How to Use the Video Function?](#)



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

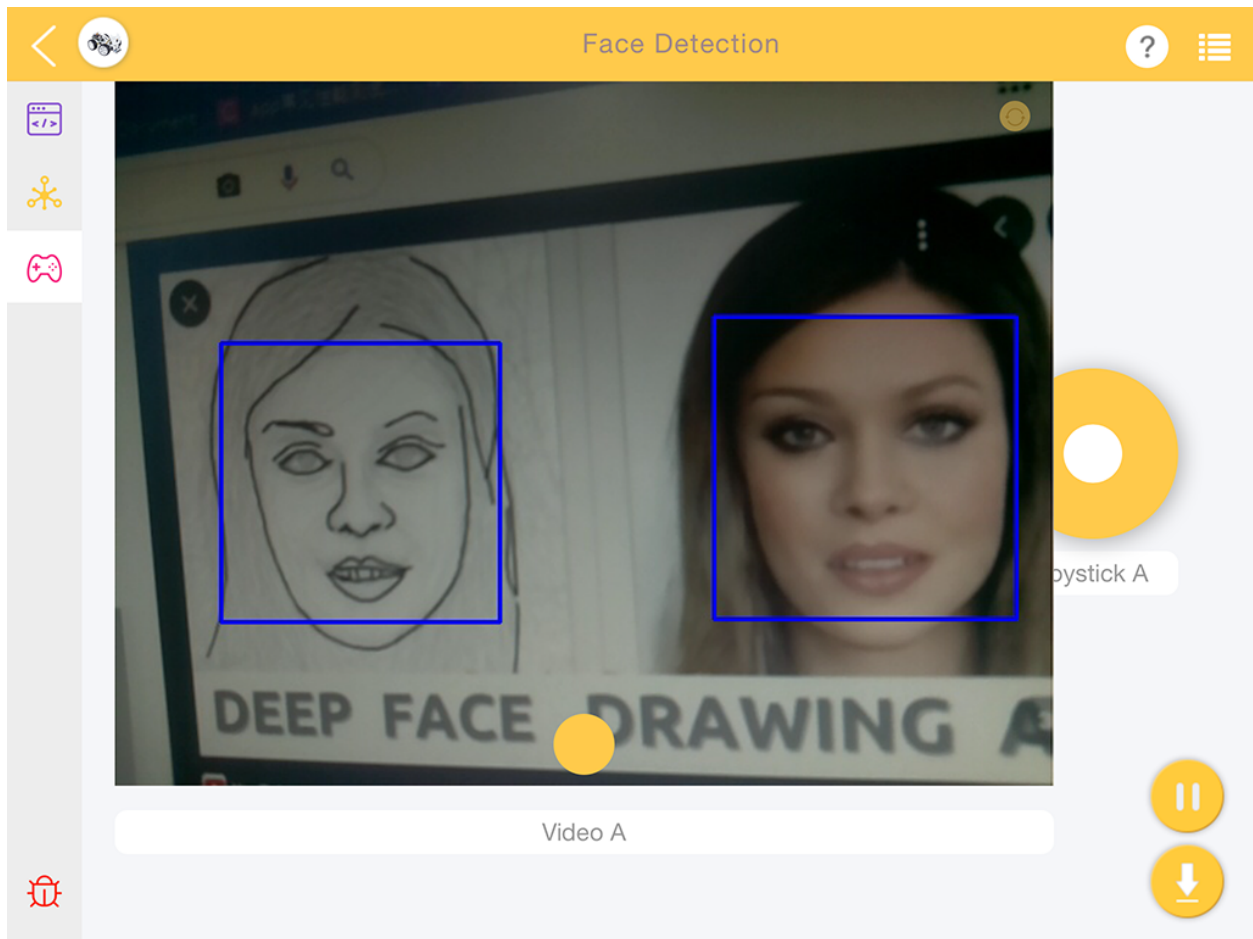


Use the **color detection** block to enable the color detection. Note: only one color can be detected at a time.

EXAMPLE**4.2.6 Face Detection**

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

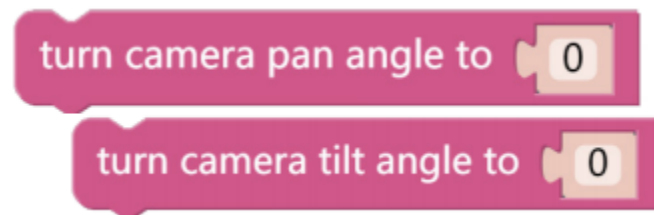
For more information on how to use the Video widget, please reference the tutorial on Ezblock video here: [How to Use the Video Function?](#)



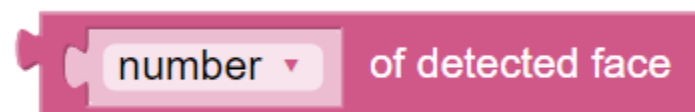
TIPS



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



These two blocks are used to adjust the orientation of the pan-tilt camera, similar to driving the PiCar-X in the [Remote Control](#) tutorial. As the value increases, the camera will rotate to the right, or upwards, a decreasing value will rotate the camera left, or downwards.

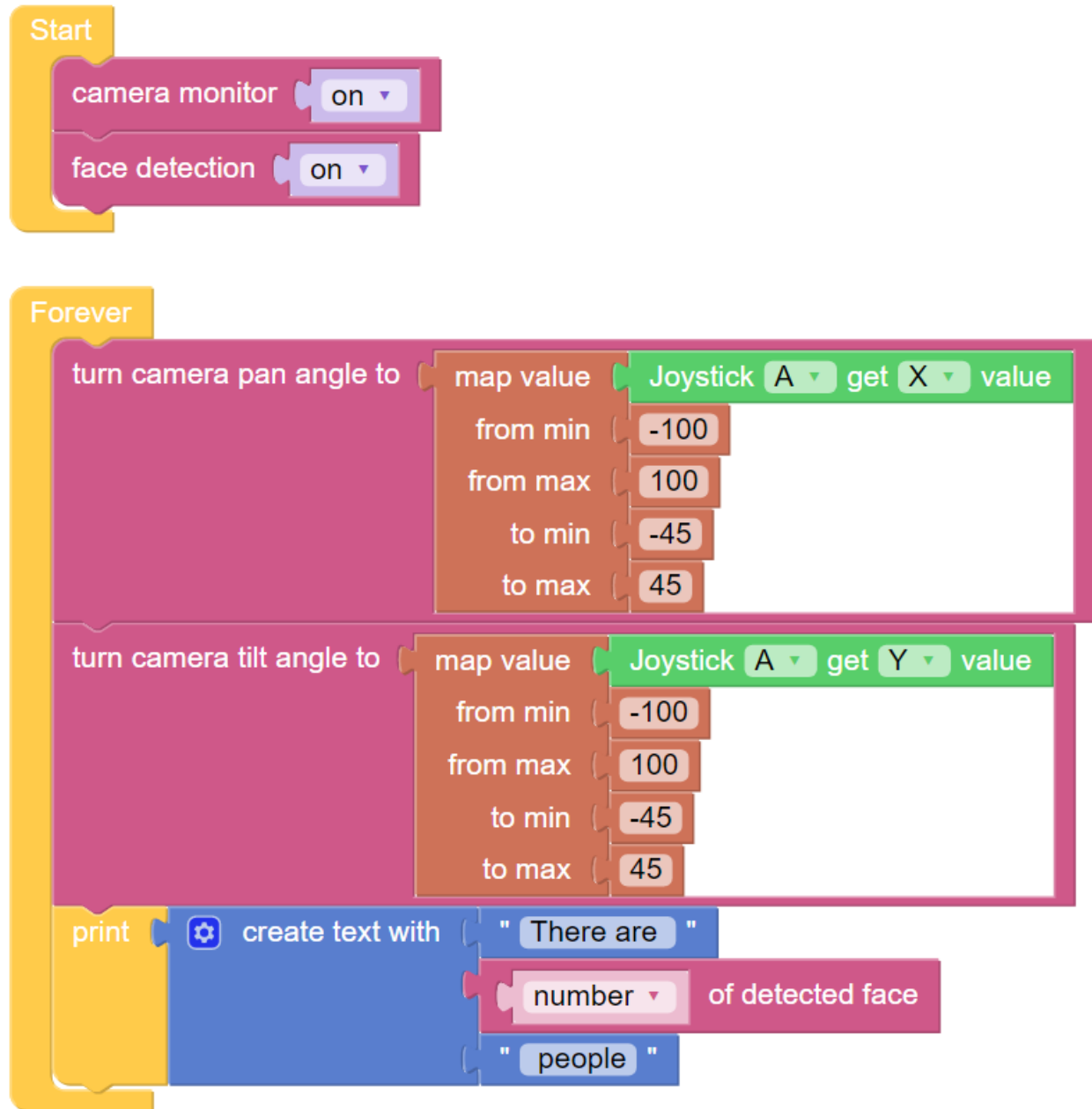


The image detection results are given through the of **detected face** block. Use the drop-down menu options to choose between reading the coordinates, size, or number of results from the image detection function.



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

EXAMPLE



4.2.7 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

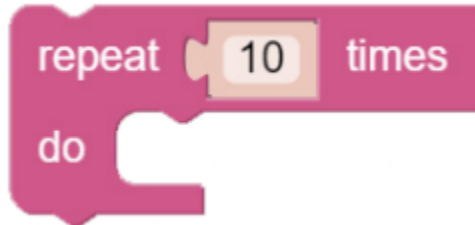


Use the **say** block with a **text** block to write a sentence for the PiCar-X to say. The **say** block can be used with text or

numbers.



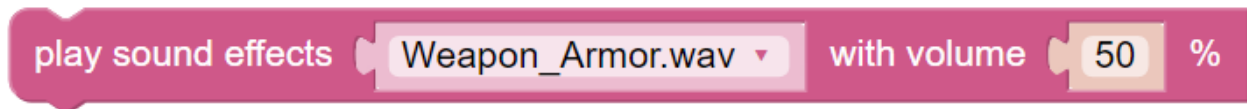
The **number** block.



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

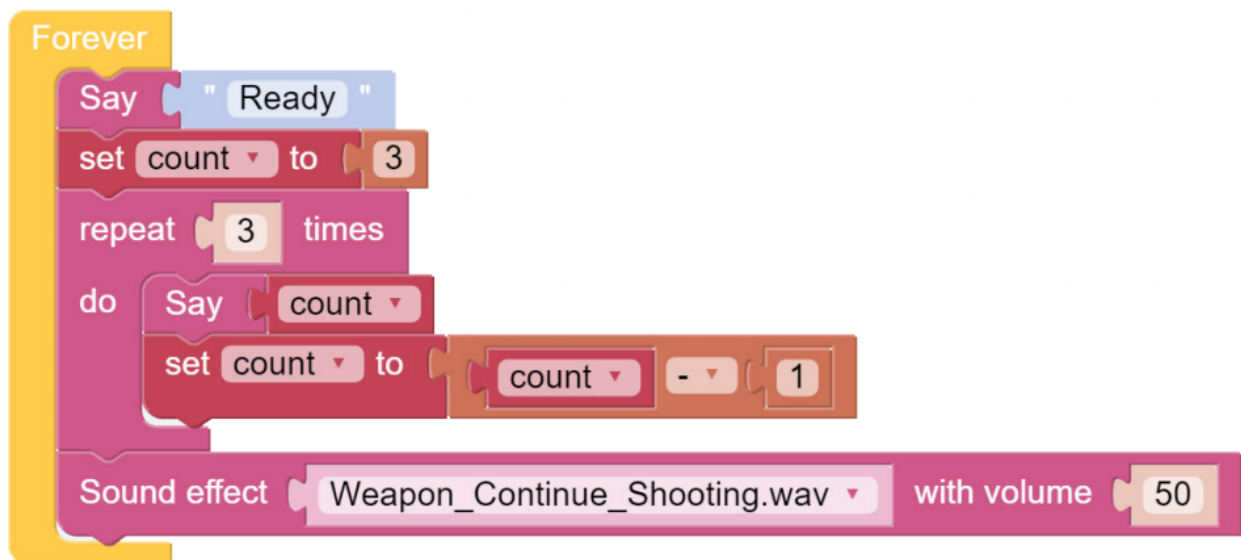


The **mathematical operation** block can perform typical mathematical functions, such as "+", "-", "x", and "÷".



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



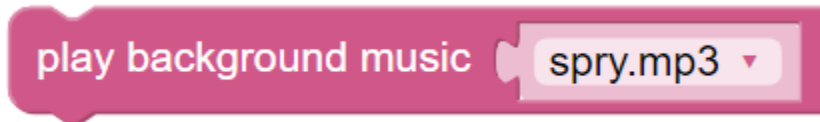
4.2.8 Background Music

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

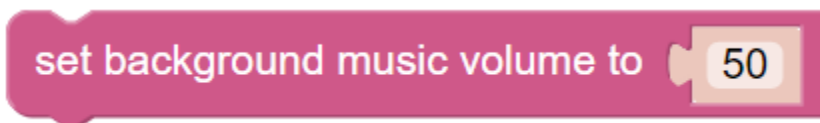
- [How to Use the Remote Control Function?](#)

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

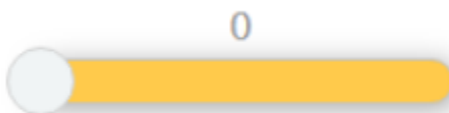
TIPS



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



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



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



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

EXAMPLE

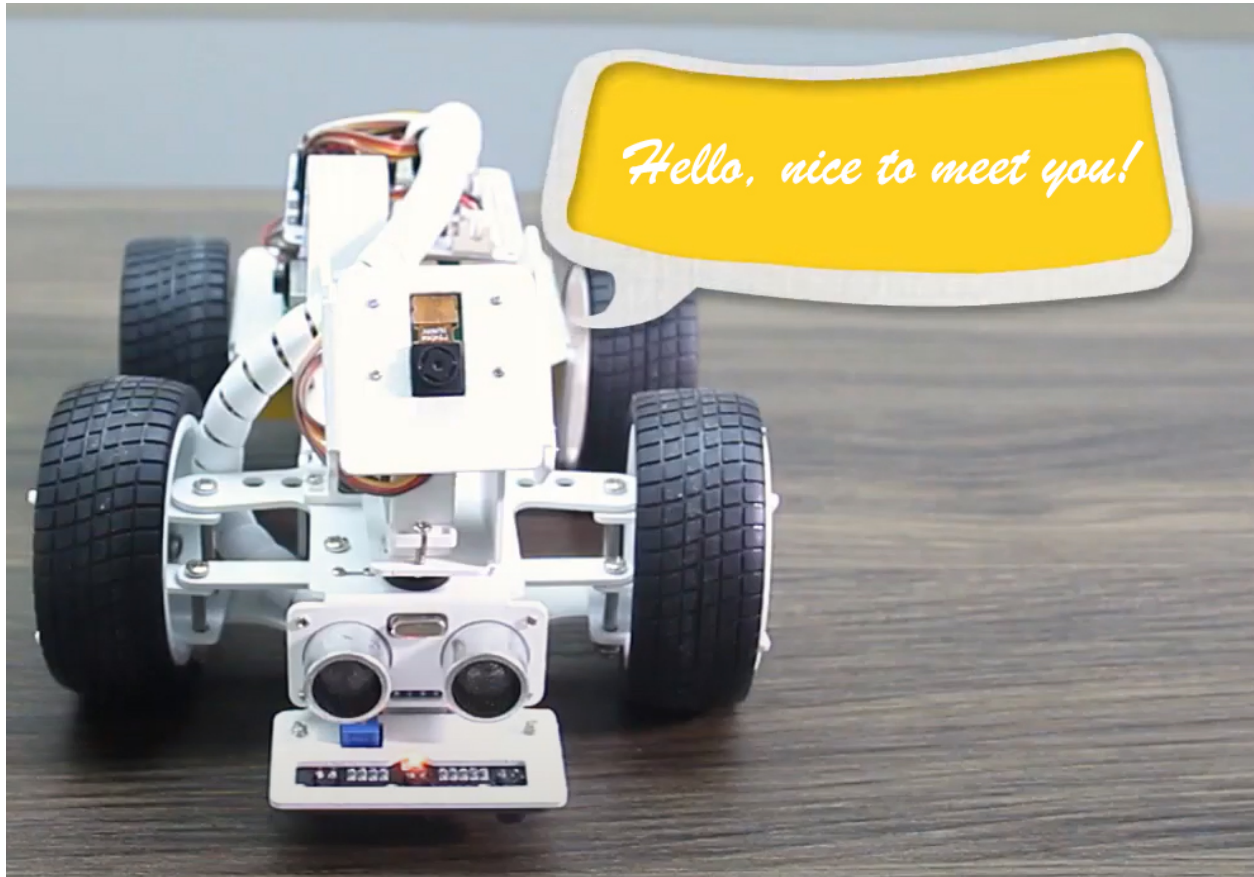


Advanced

4.2.9 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

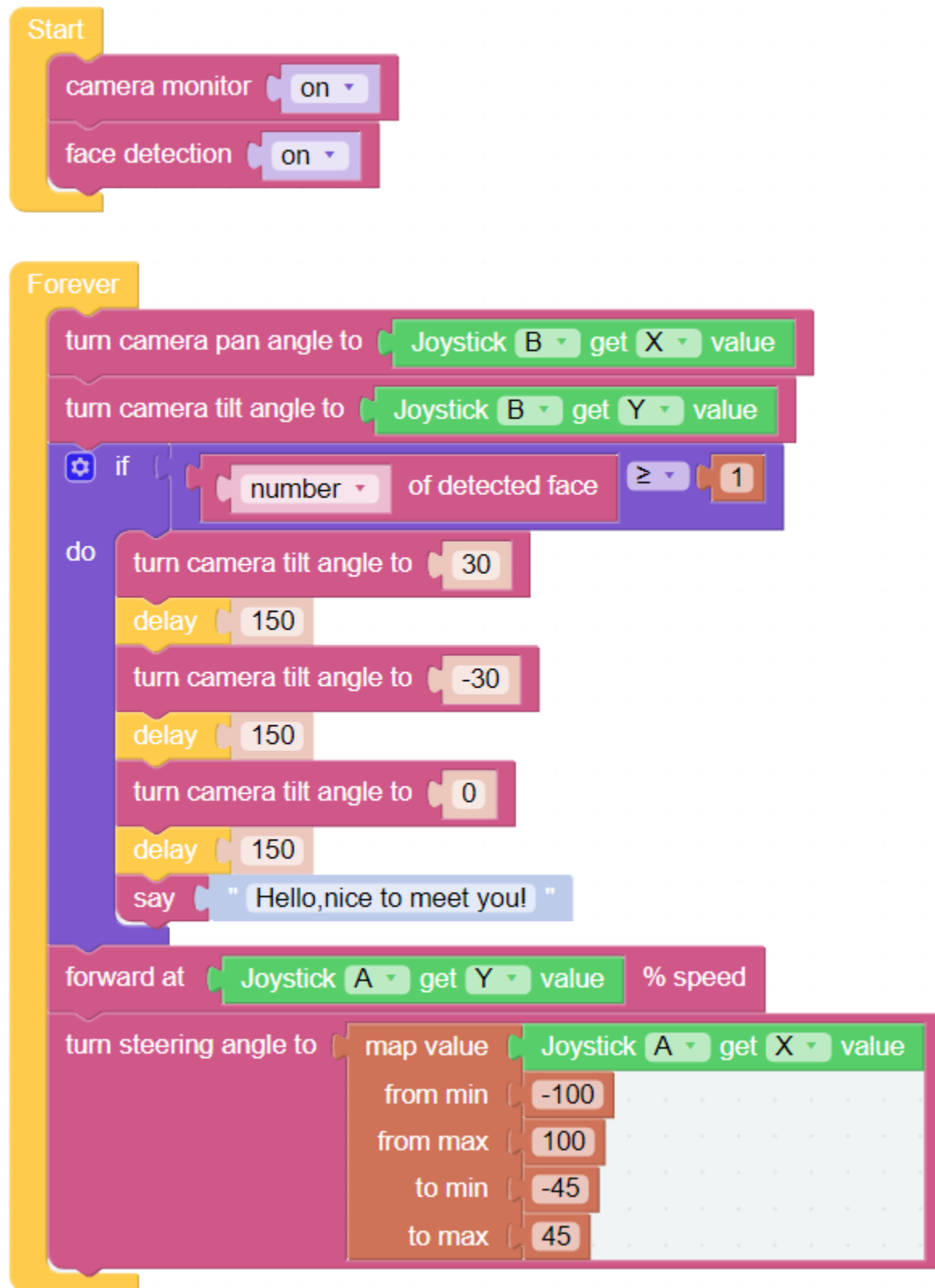


The **if do** block is used to nod politely once the conditional judgment of "if" is true.



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

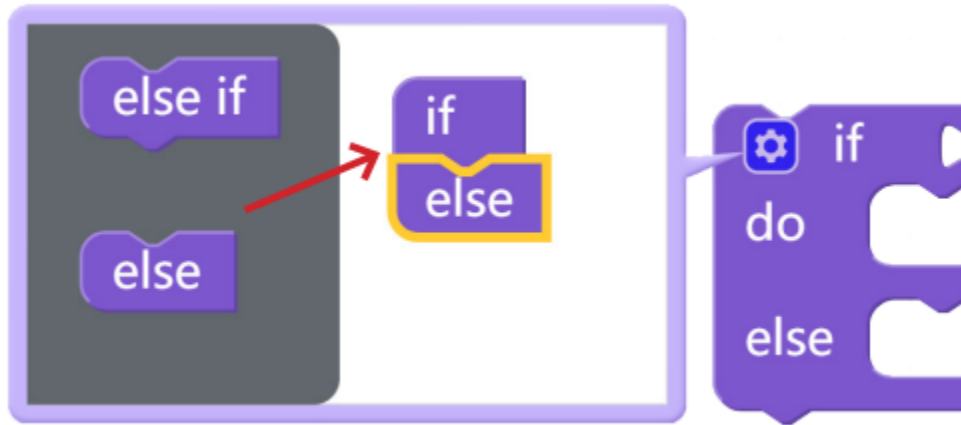
EXAMPLE



4.2.10 Music Car

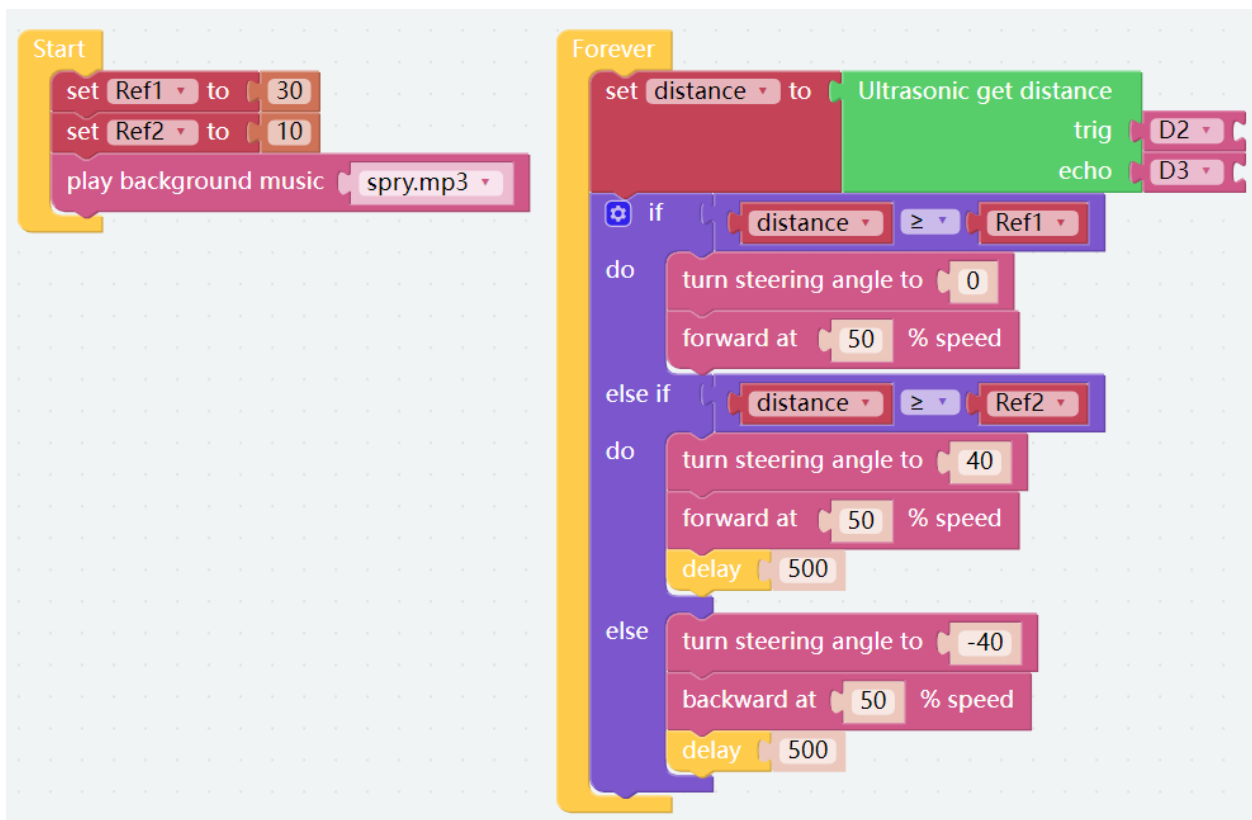
This project will turn the PiCar-X into a music car that will travel around your home, playing cheerful music. This project will also show how the PiCar-X avoids hitting walls with the built-in ultrasonic sensor.

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.

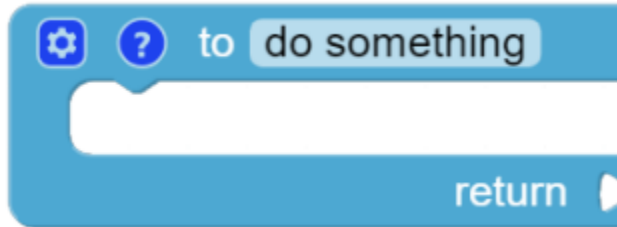
EXAMPLE



4.2.11 Cliff Detection

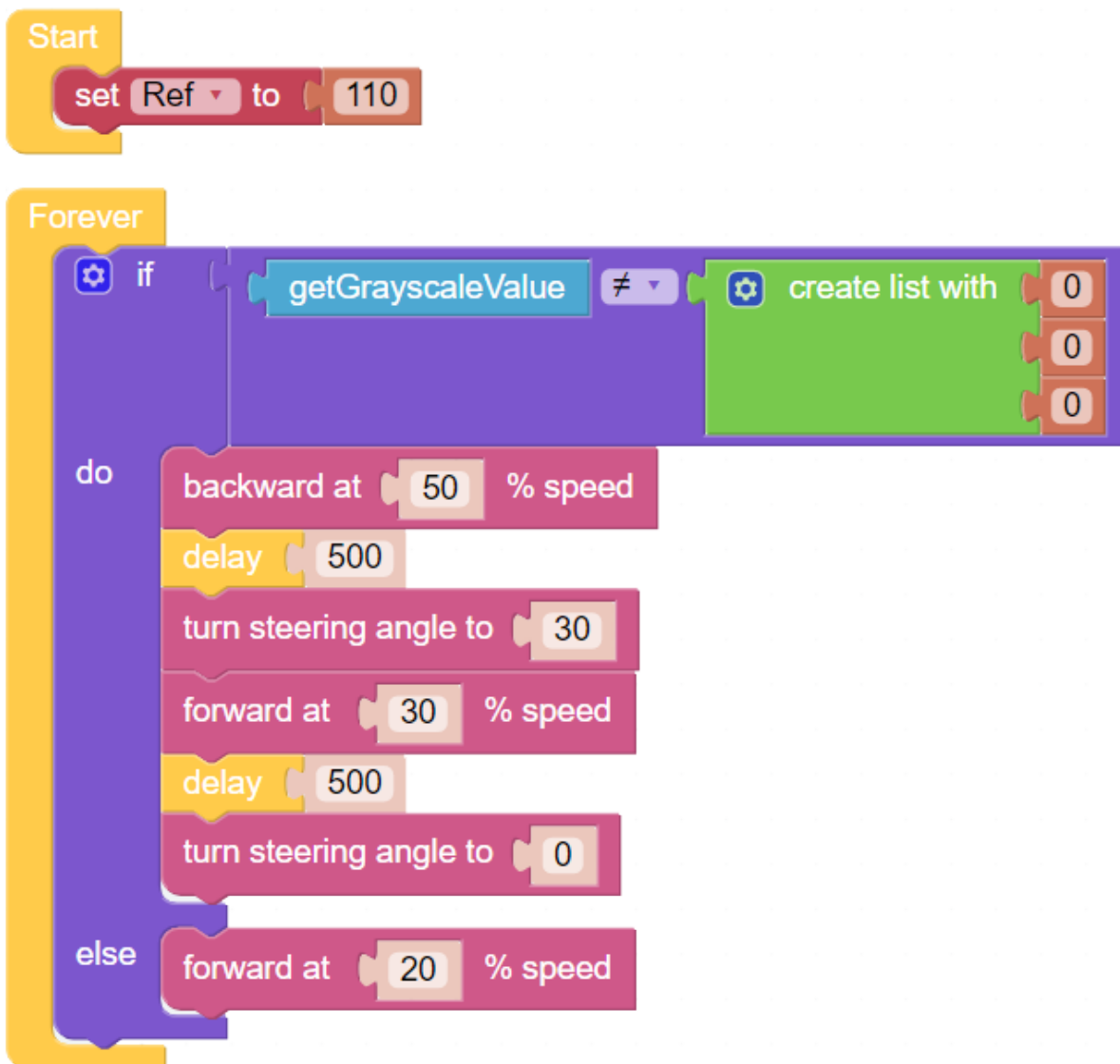
This project will use the **grayscale module** to prevent the PiCar-X from falling off a cliff while it is moving freely around your home. This is an essential project for houses with staircases.

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



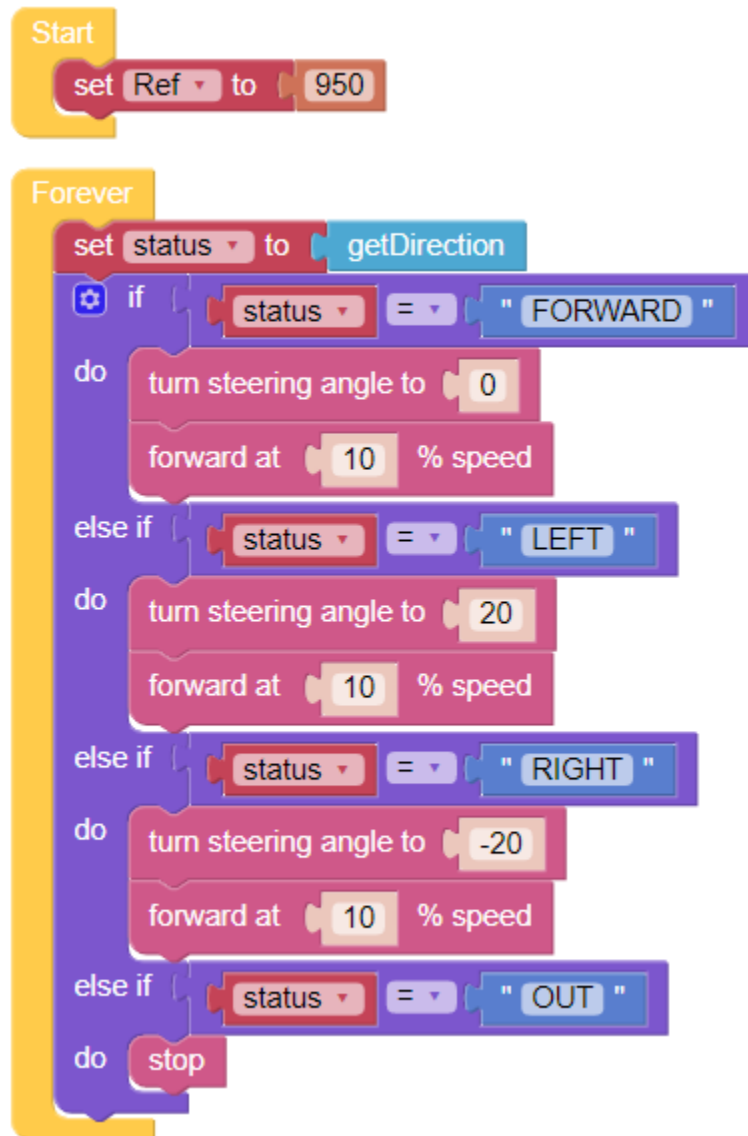


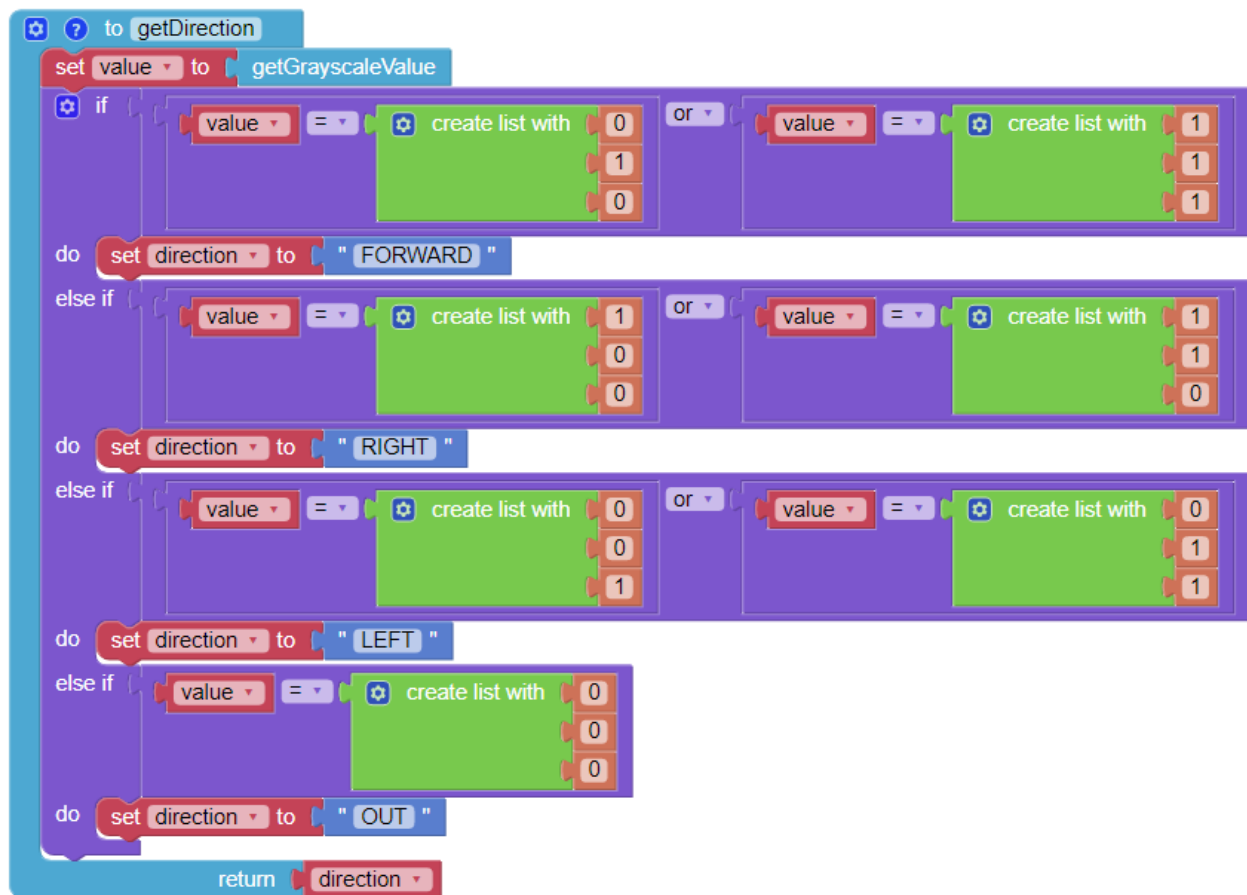
4.2.12 Minecart

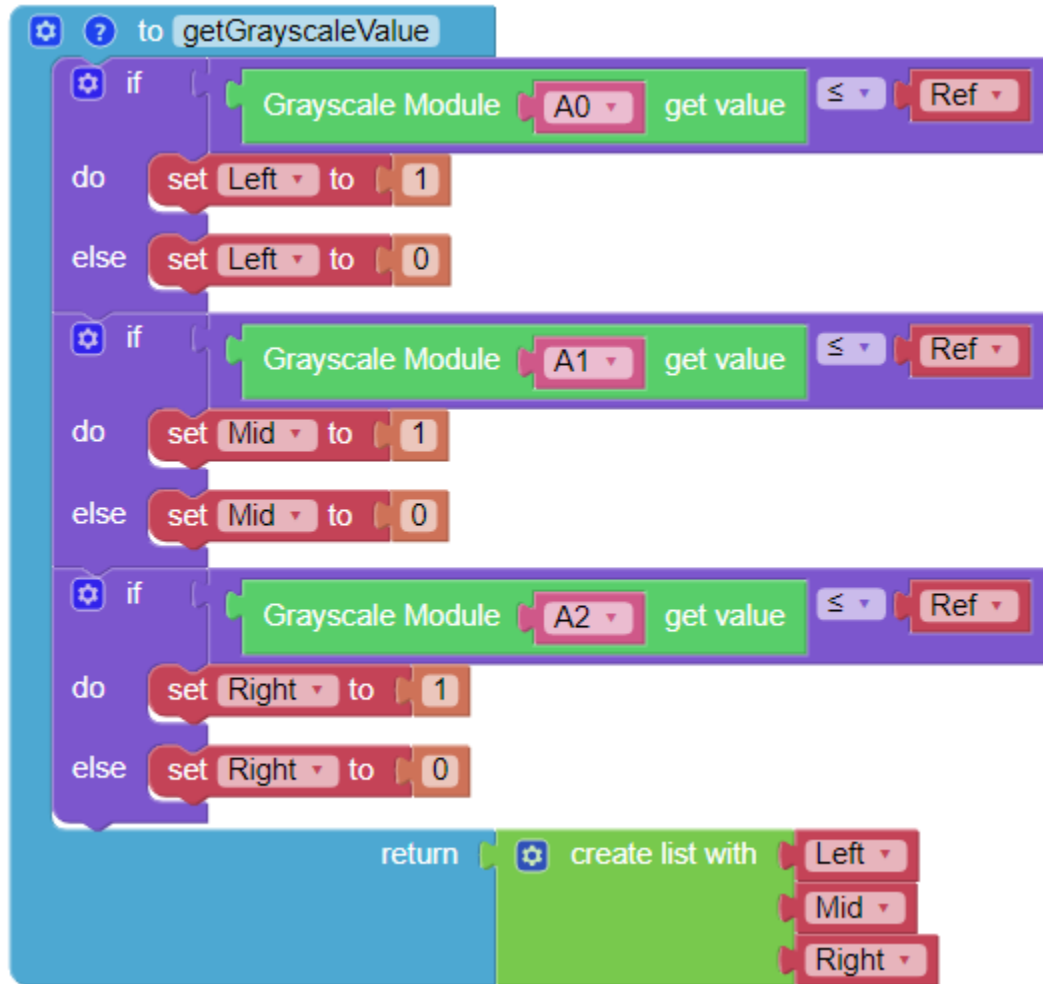
Let's make a minecart project! This project will use the Grayscale module to make the PiCar-X move forward along a track. Use dark-colored tape to make a track on the ground as straight as possible, and not too curved. Some experimenting might be needed if the PiCar-X becomes derailed.

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 derailed, the Grayscale module will no longer detect the dark-colored tape track, and the PiCar-X will come to a stop.

EXAMPLE

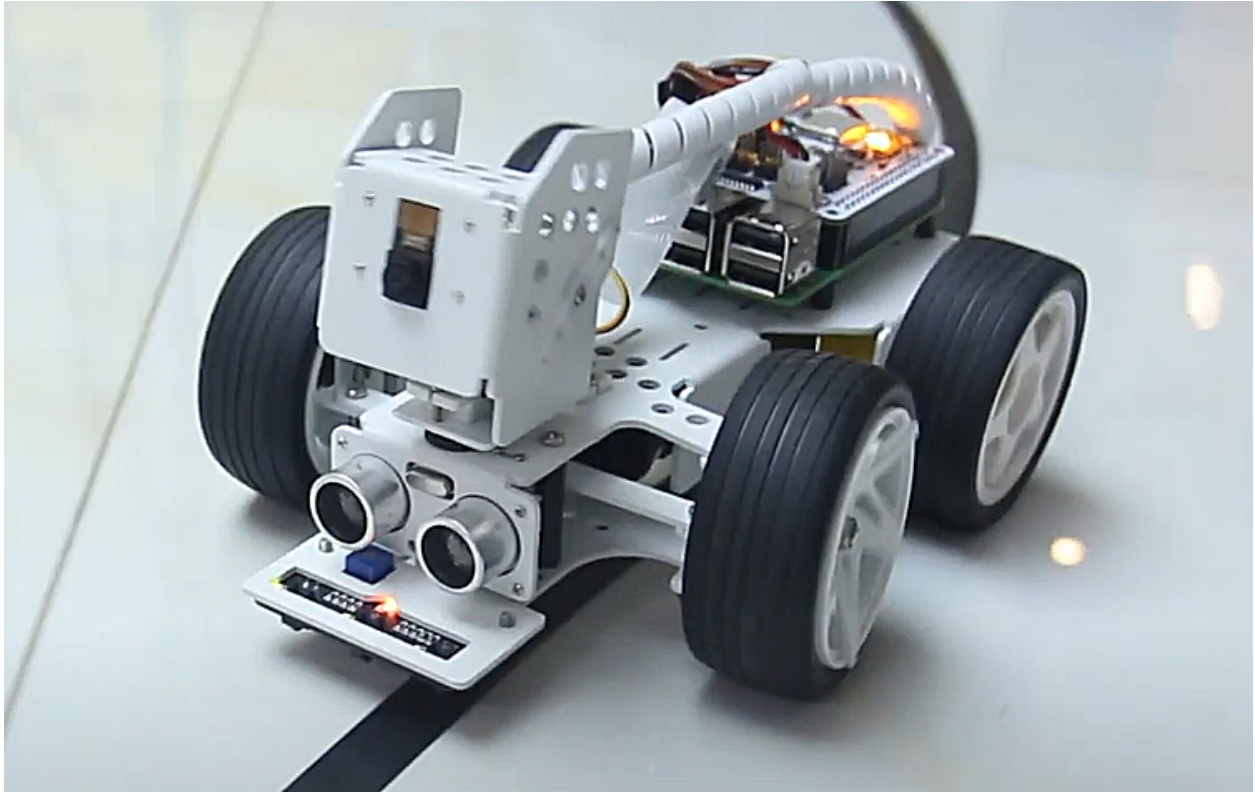
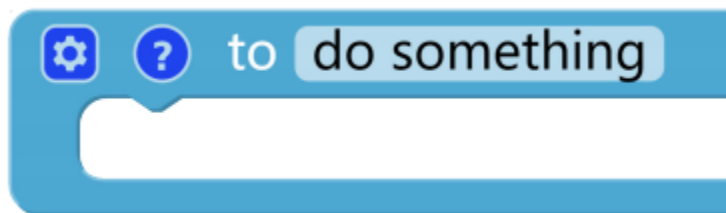






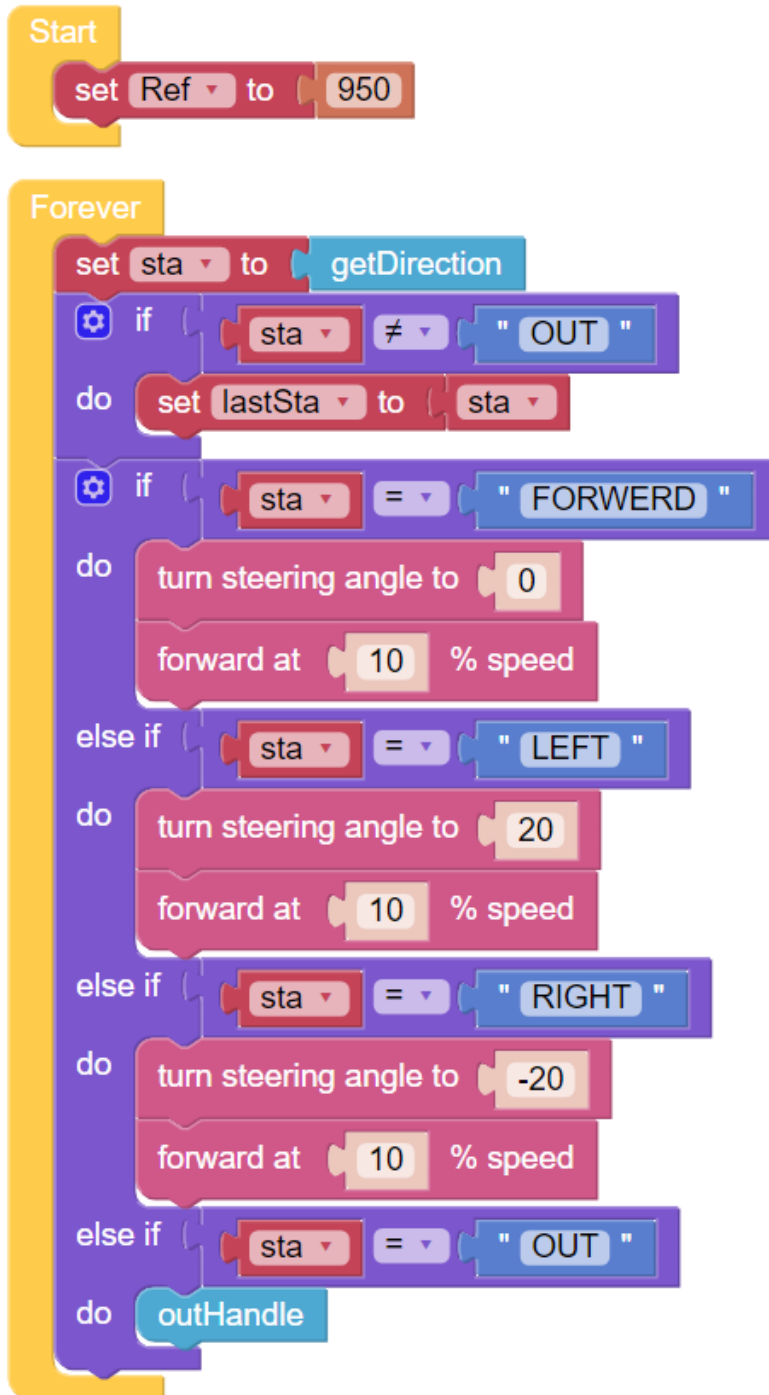
4.2.13 Minecart Plus

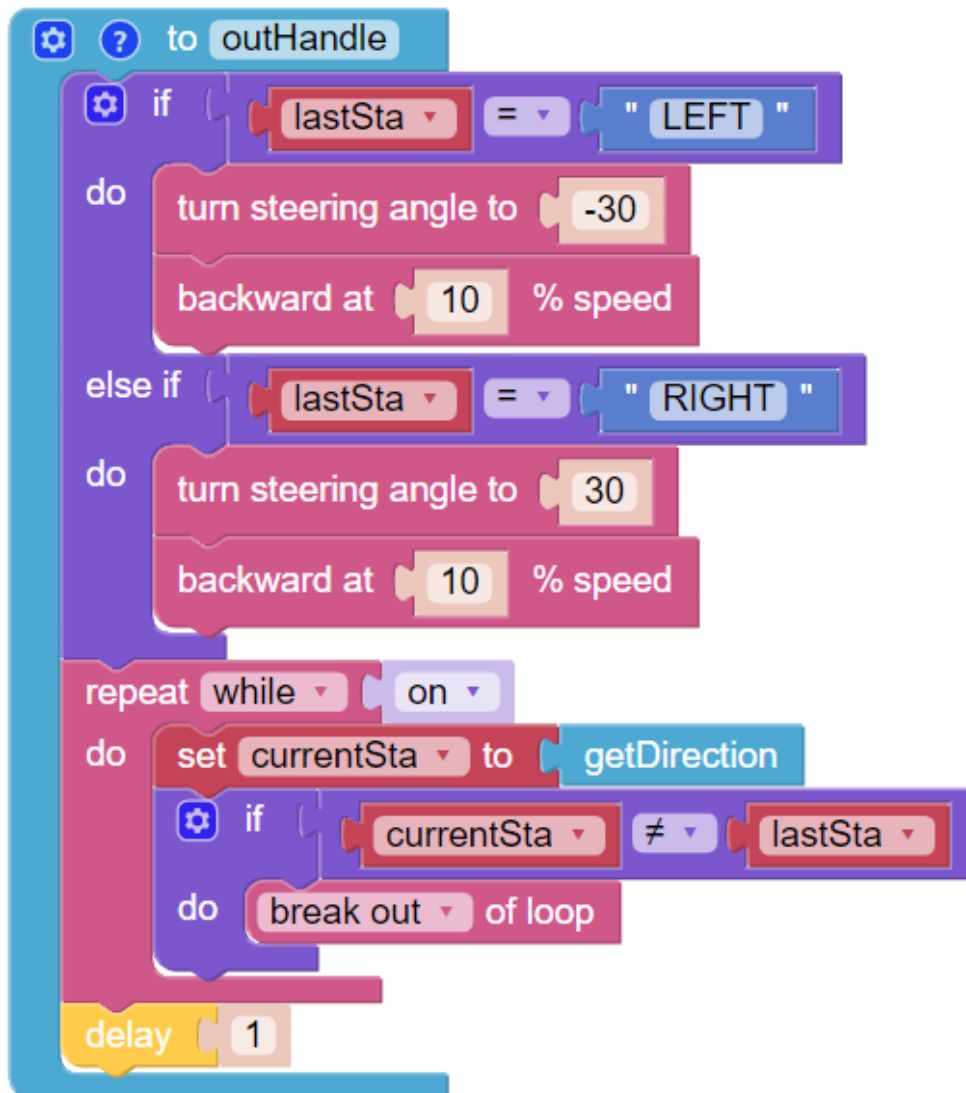
In this project, derailment recovery has been added to the *Minecart* project to let the PiCar-X adapt and recover from a more severe curve.

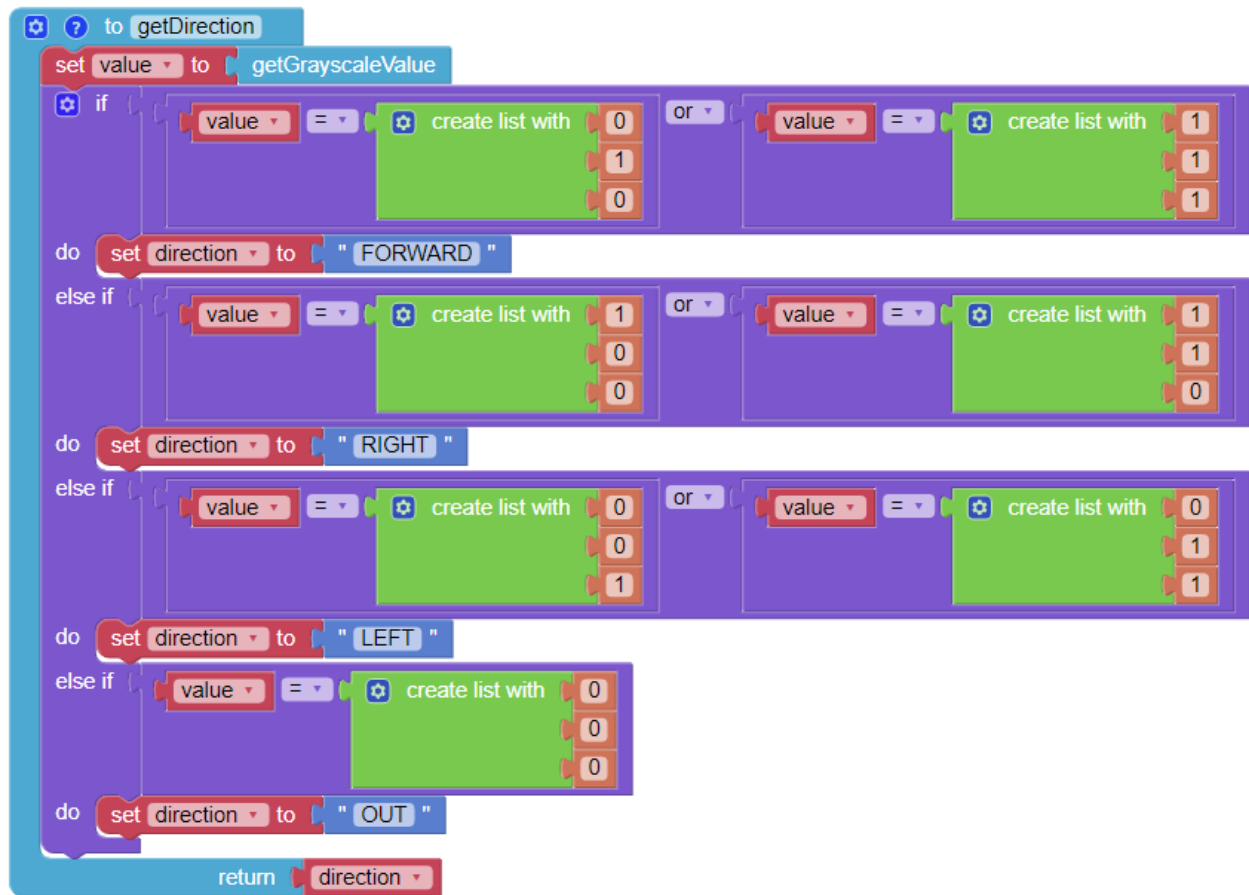
**TIPS**

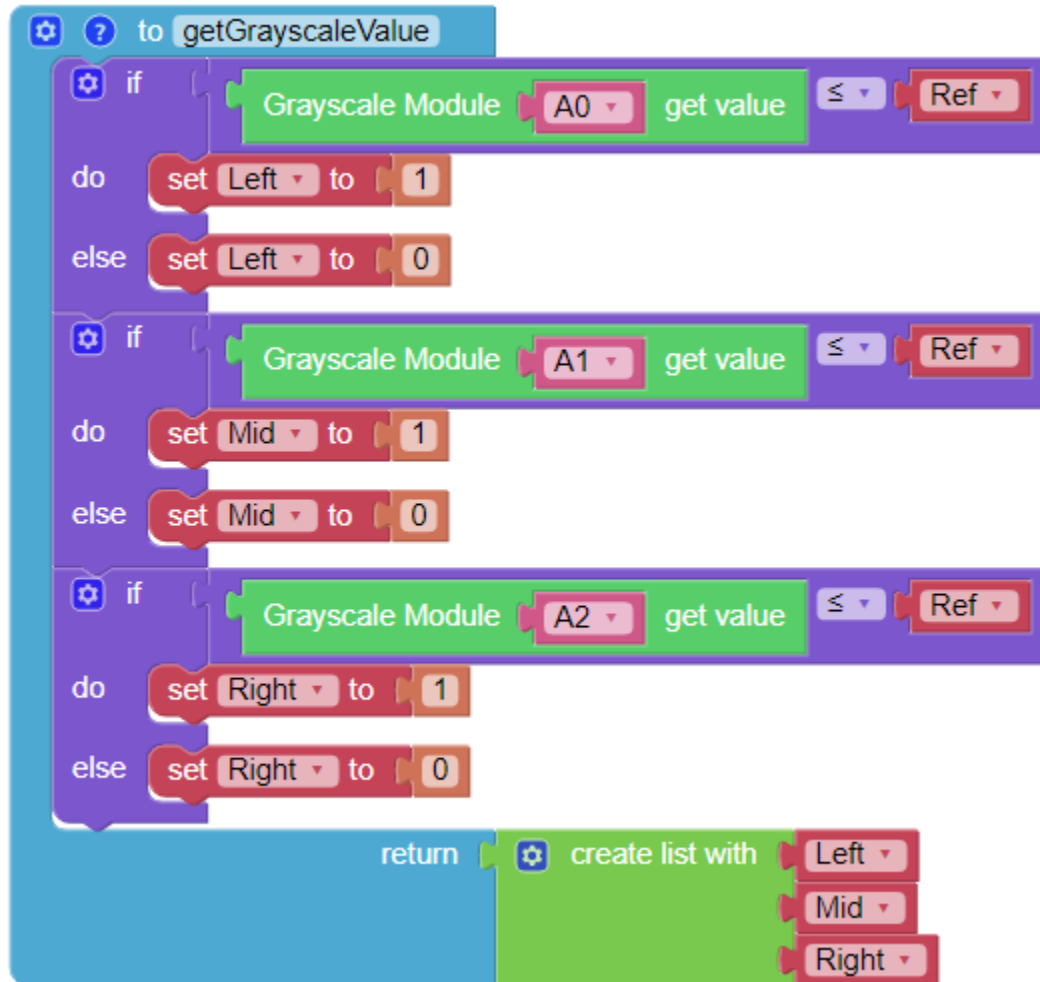
Use another **to do something** block to allow the PiCar-X to back up and recover from a sharp curve. Note that the new **to do something** function does not return any values, but is used just for reorienting the PiCar-X.

EXAMPLE









4.2.14 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.

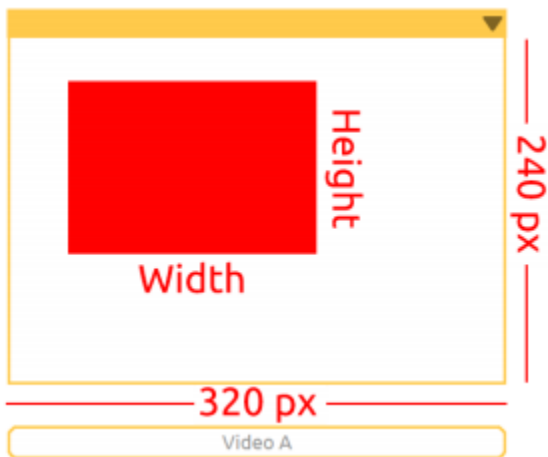
TIPS



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.

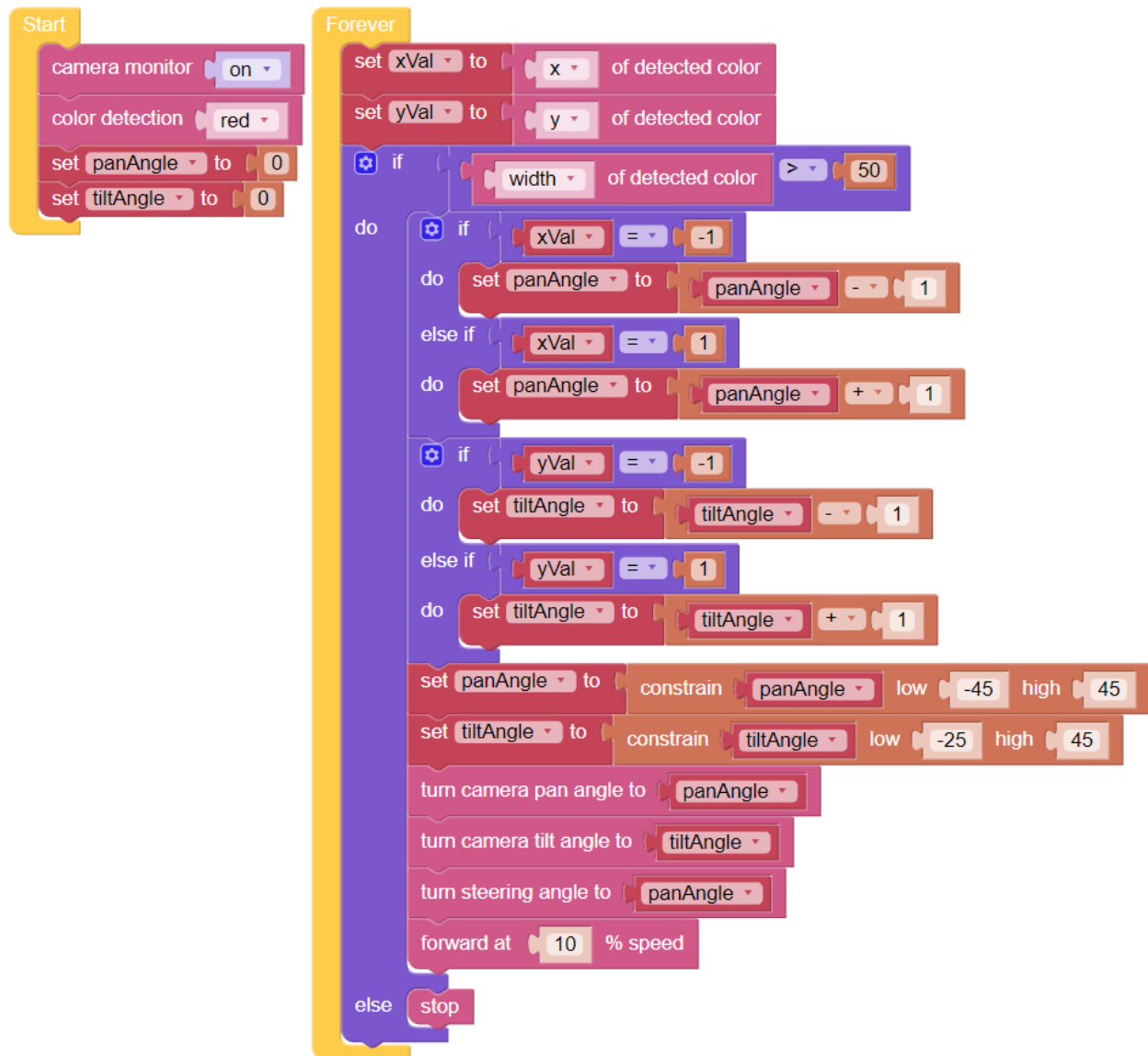
$(-1,1)$	$(0,1)$	$(1,1)$
$(-1,0)$	$(0,0)$	$(1,0)$
$(-1,-1)$	$(0,-1)$	$(1,-1)$

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



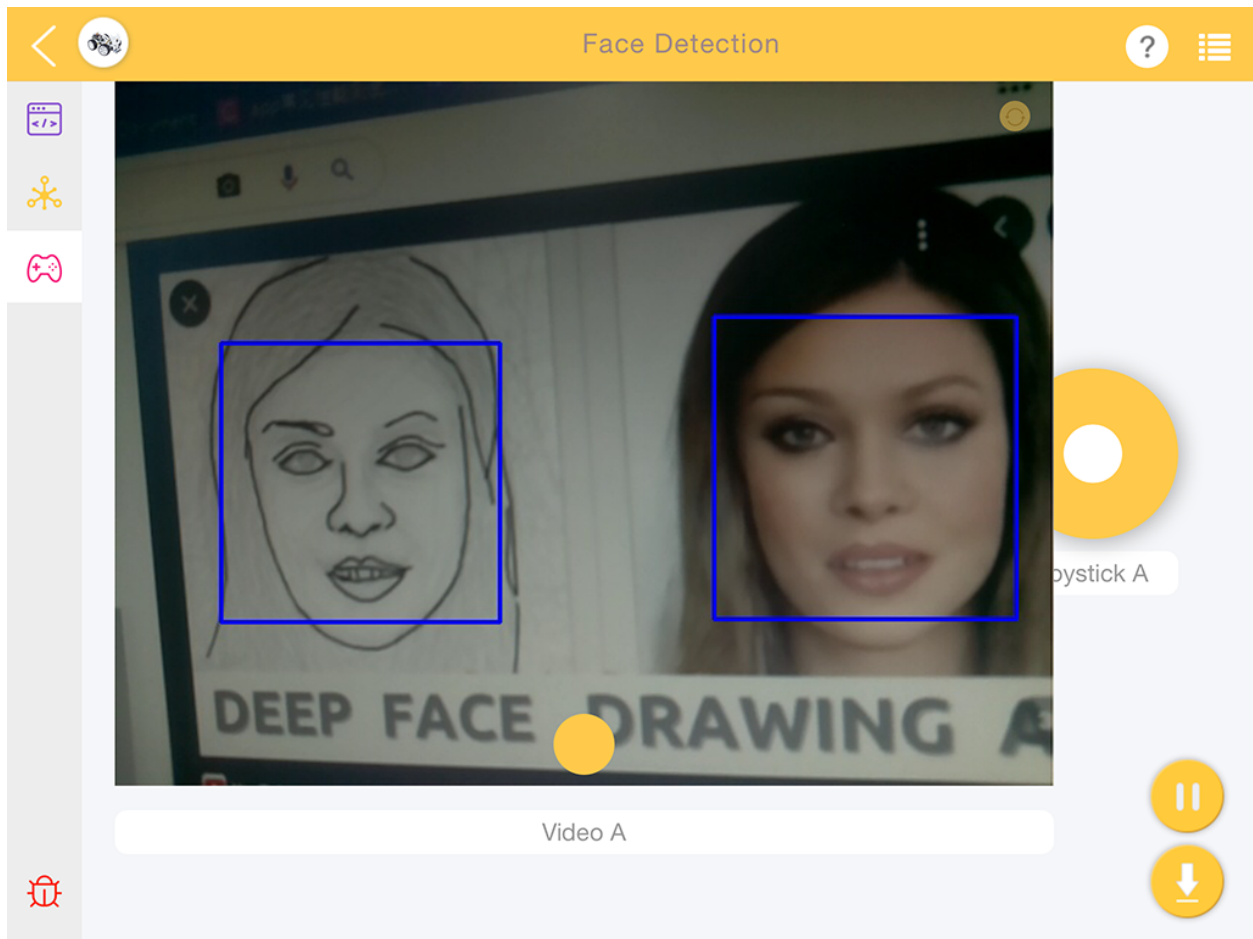
4.2.15 Beware of Pedestrians

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.

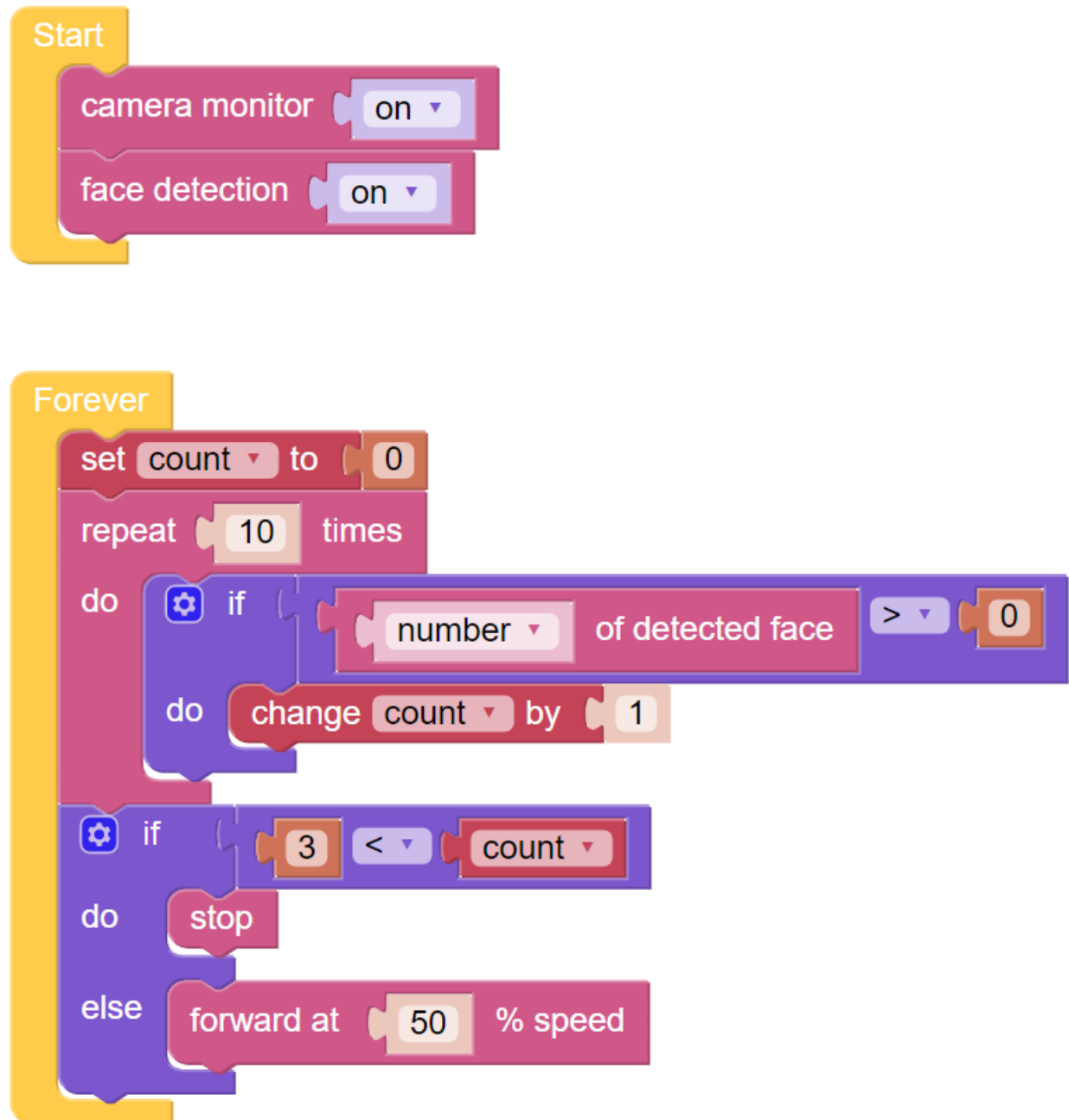
Once the program is running, hold a photo of a person in front of the PiCar-X. The Video Monitor will detect the person's face, and the PiCar-X will automatically come to a stop.

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 Video Function](#)



EXAMPLE



4.2.16 Traffic Sign Detection

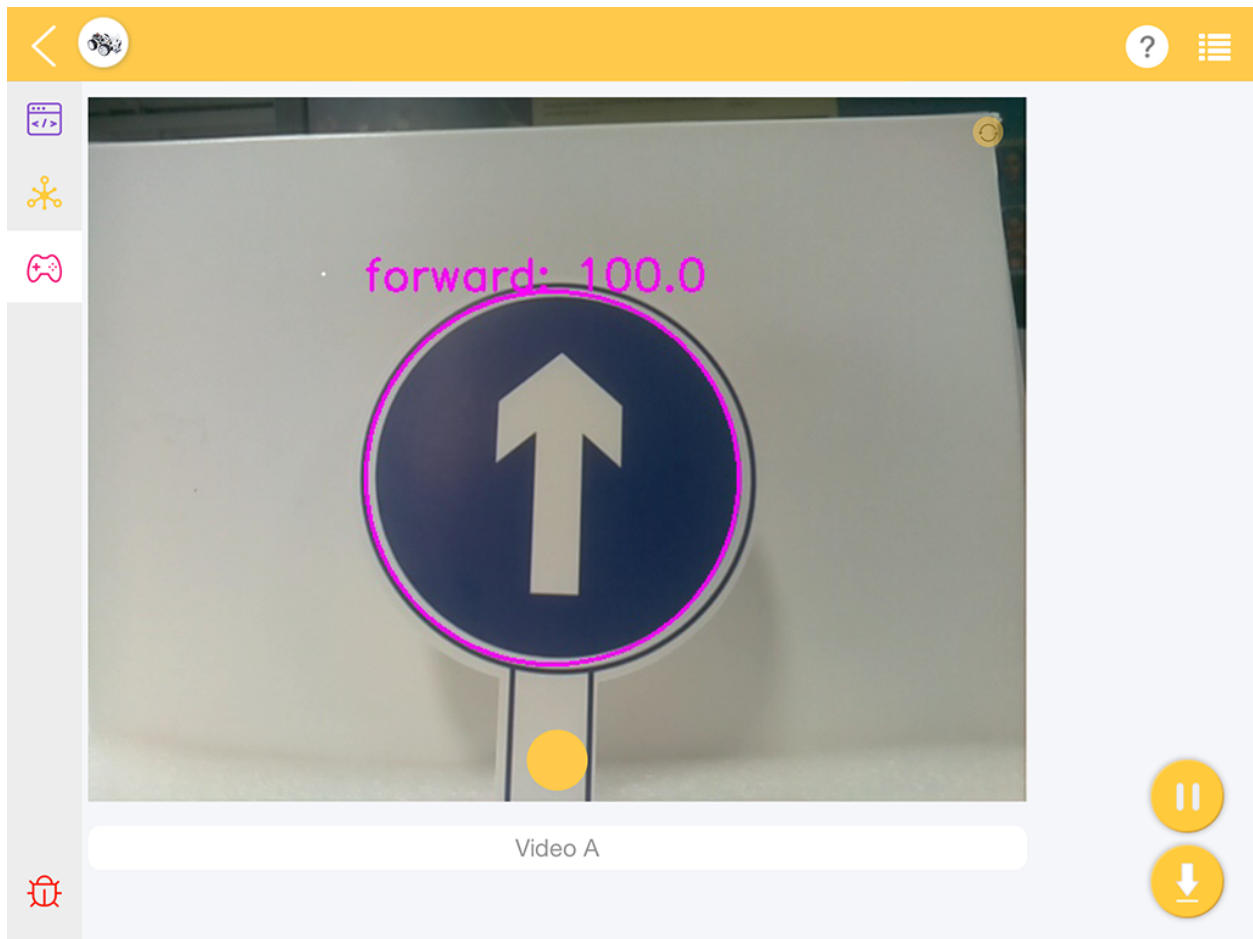
This project uses an image detection function to look for traffic signs, and make the PiCar-X follow the instructions on the sign. The turn **traffic sign detection [on]** block will recognize 4 different traffic sign models included in the printable PDF below. When the PiCar-X detects a **STOP** sign, 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

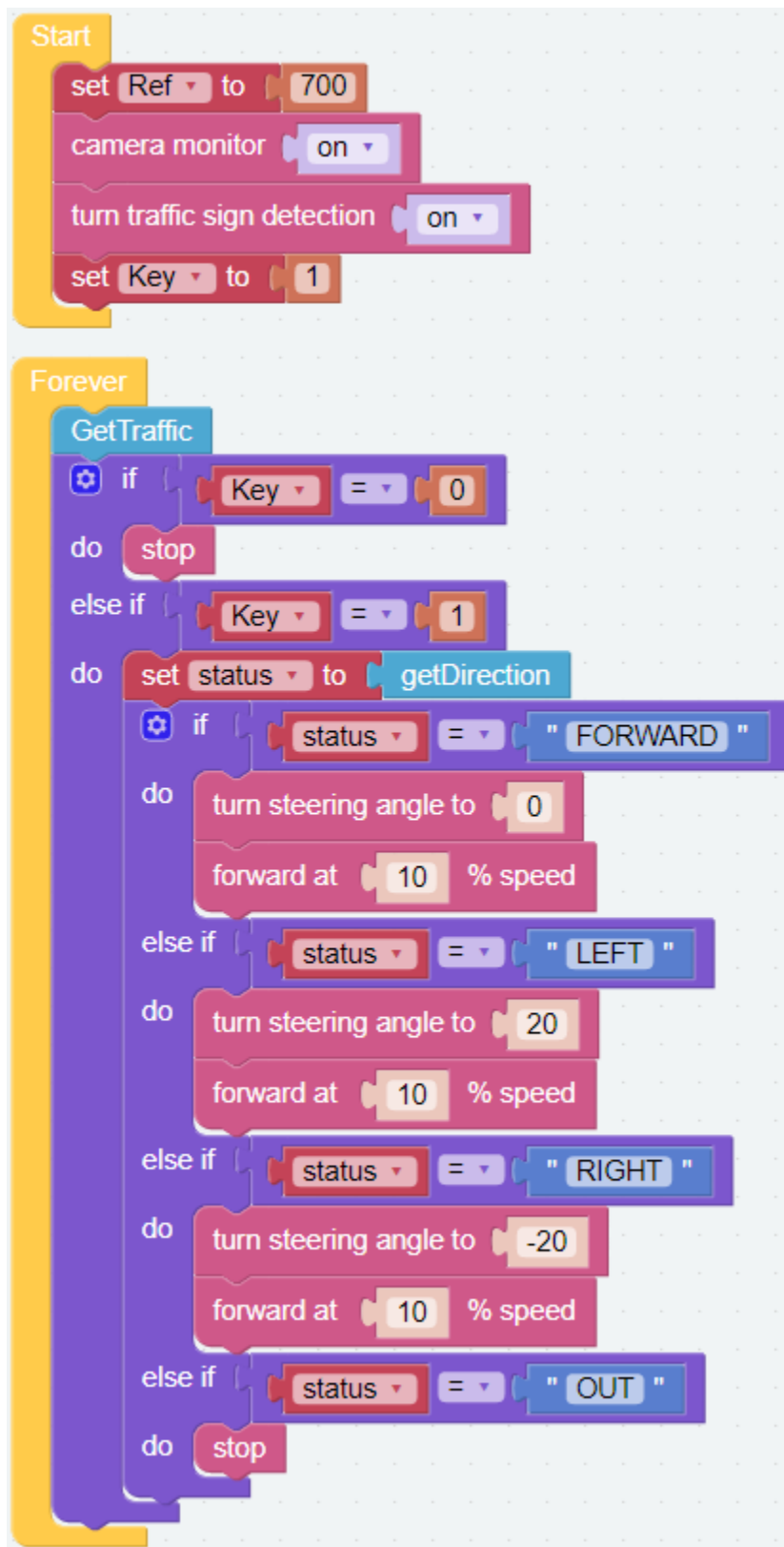


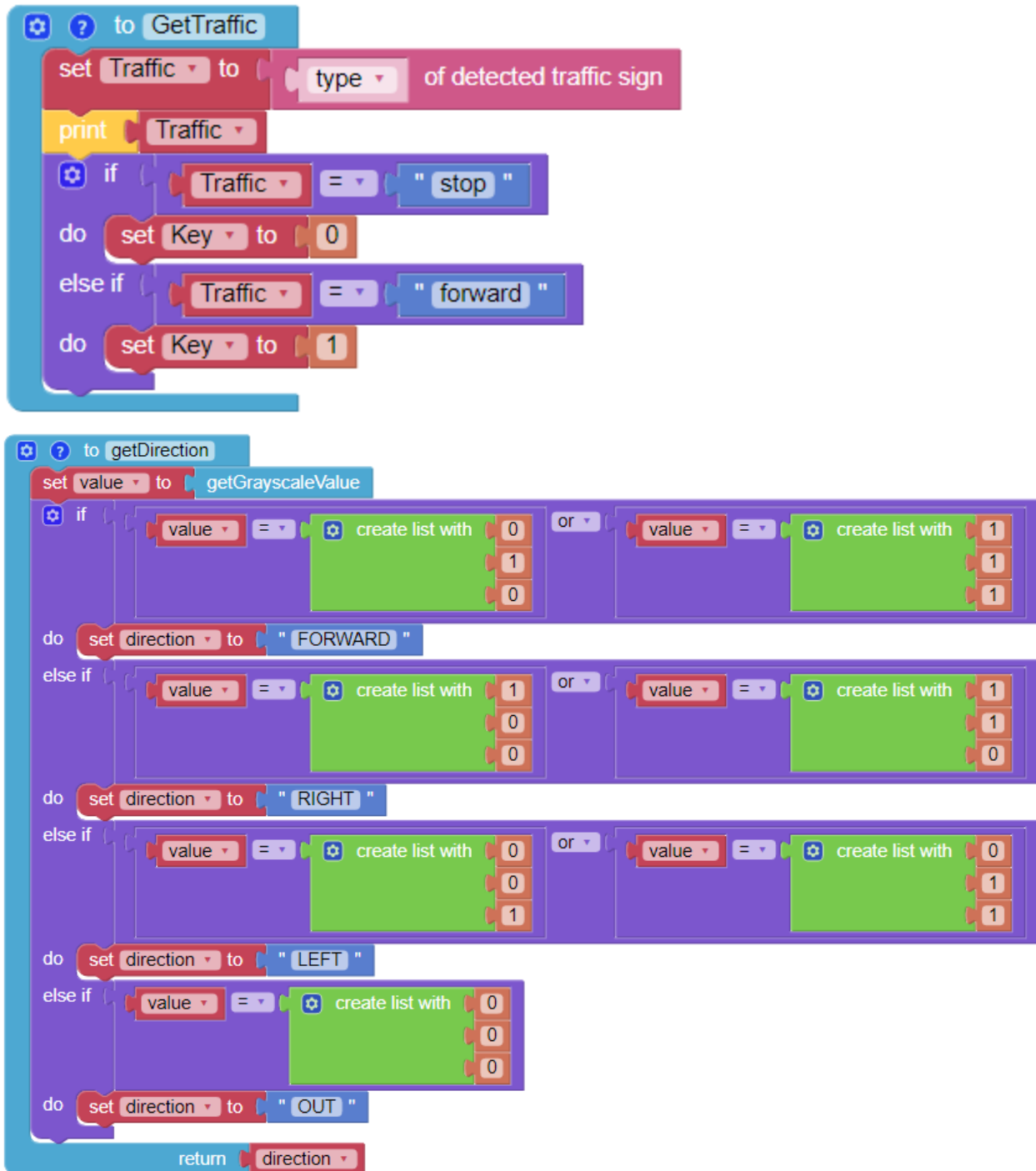
Note: The printed traffic sign colors may have a slightly different hue from the Ezblock color models due to printer toner differences, or the printed medium, such as a tan-colored paper. This can cause a less accurate color recognition.

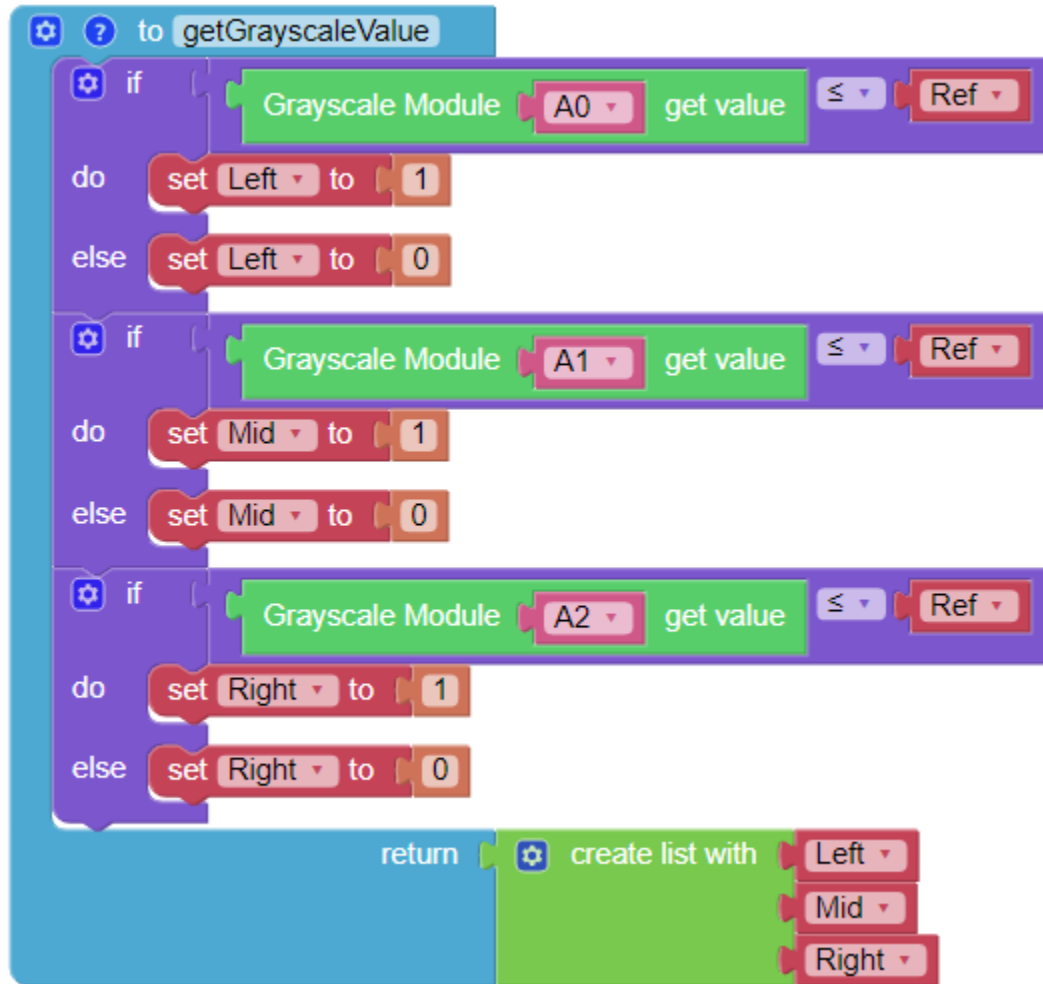
This project is based on [Minecart](#) , but instead of using the grayscale sensor, the PiCar-X uses an algorithm for traffic sign detection. The detection results can be viewed via the Video Monitor in Ezblock Studio.



TIPS







4.2.17 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](#)

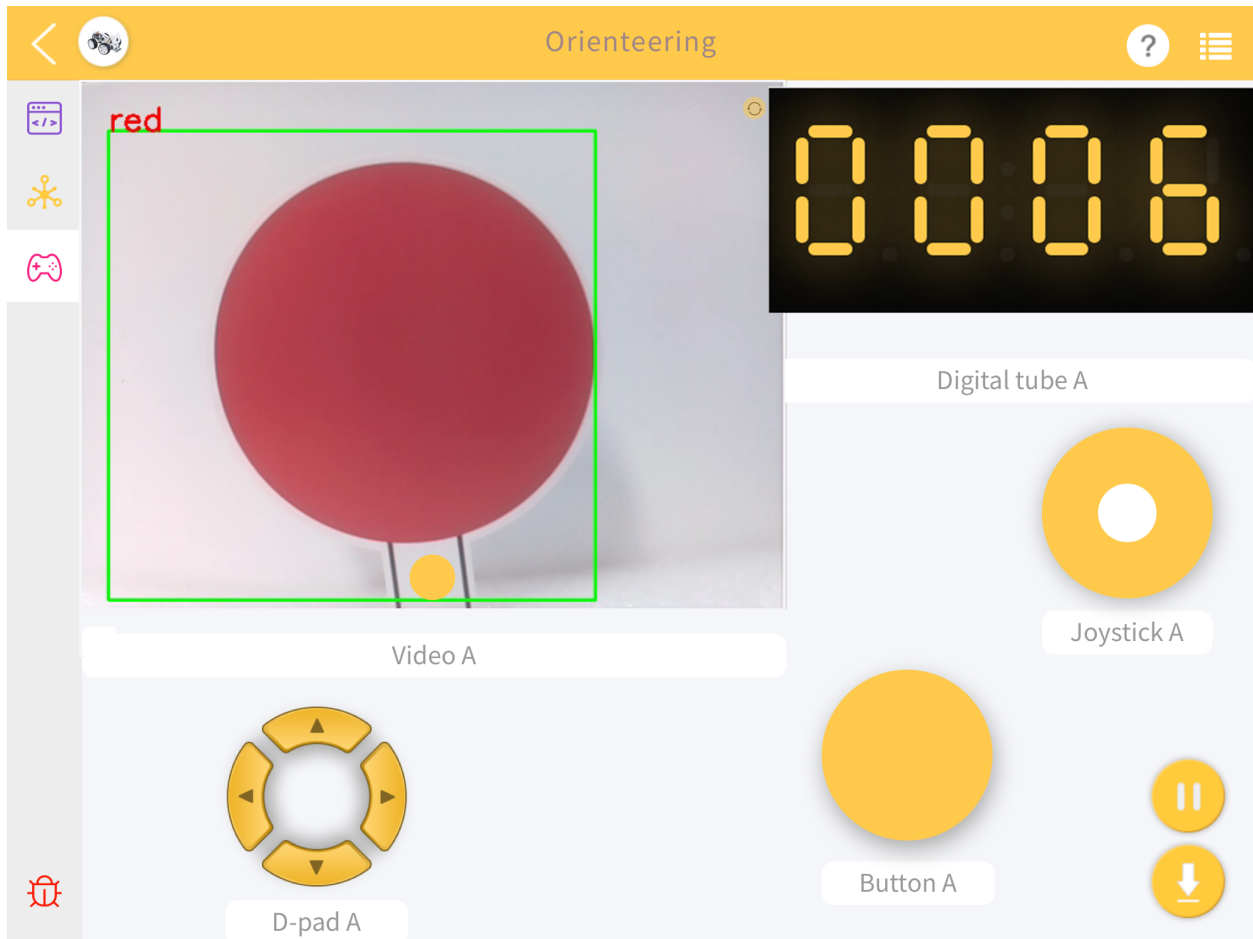


Note: The printed colors may have a slightly different hue from the Ezblock color models due to printer toner differences, or the printed medium, such as a tan-colored paper. This can cause a less accurate color recognition.

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.

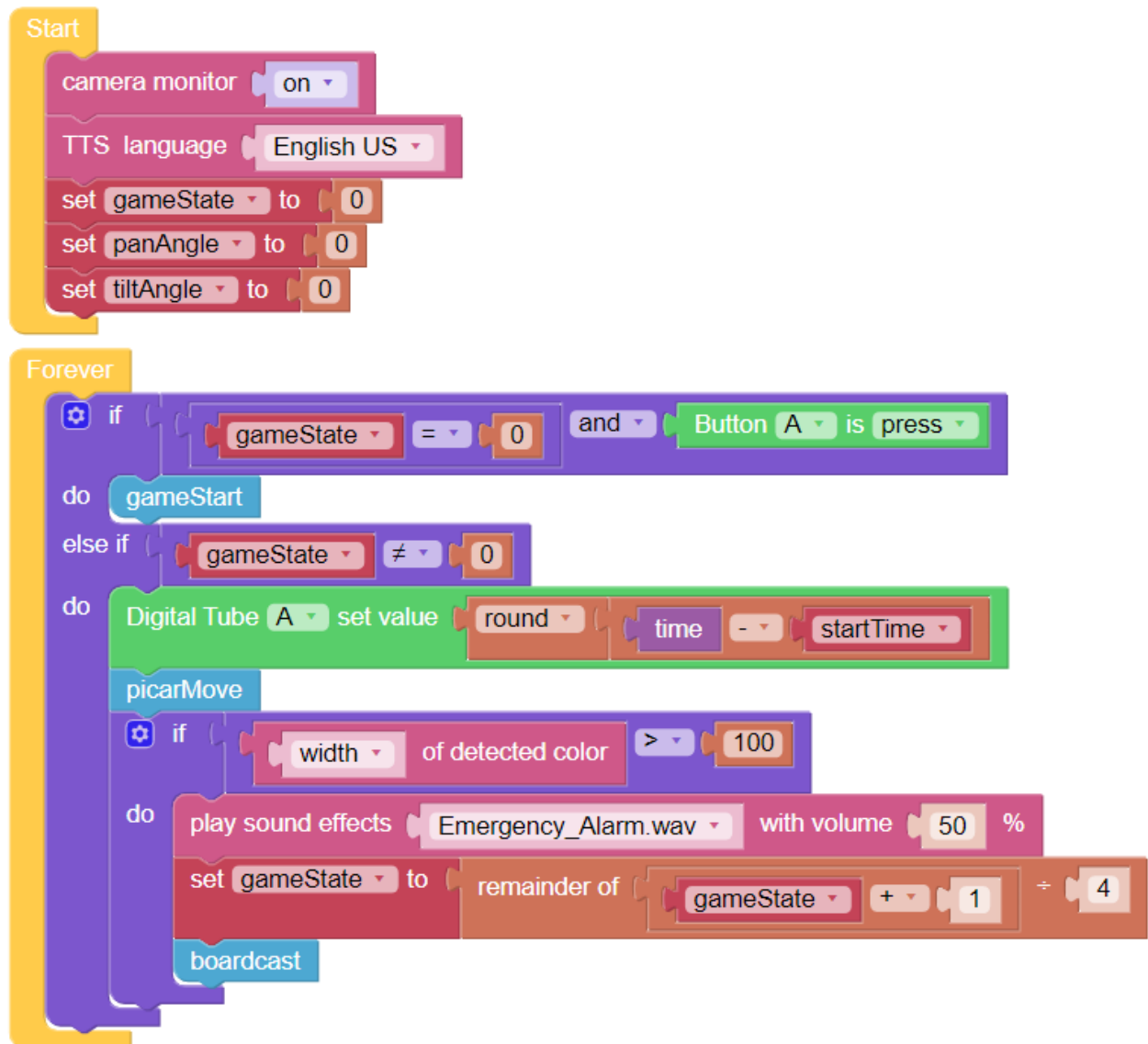
The objective is to help the PiCar-X find each of the three colors in as short of a time as possible.

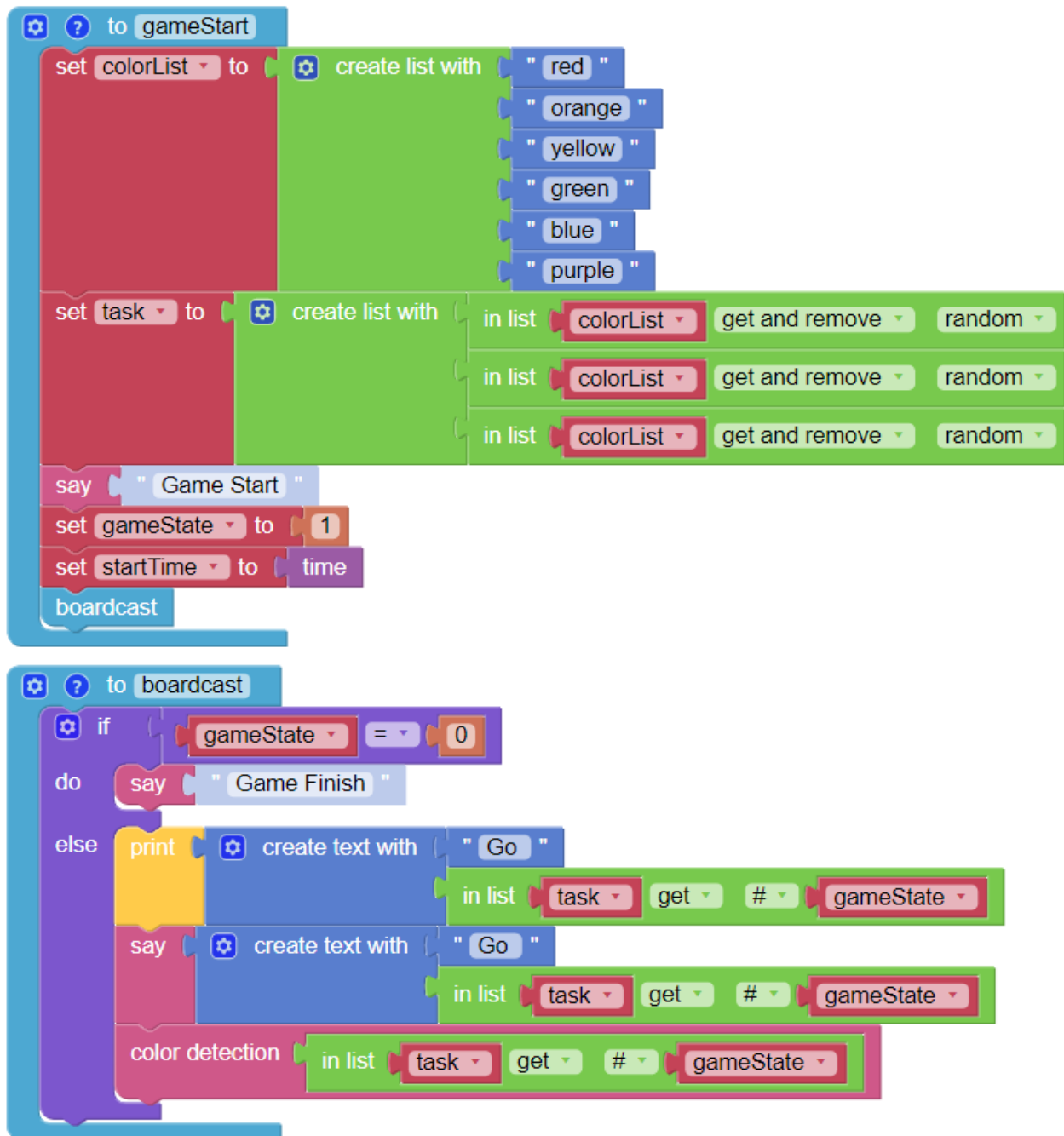
Place PiCar-X in the middle of the field and click the Button on the Remote Control page to start the game.

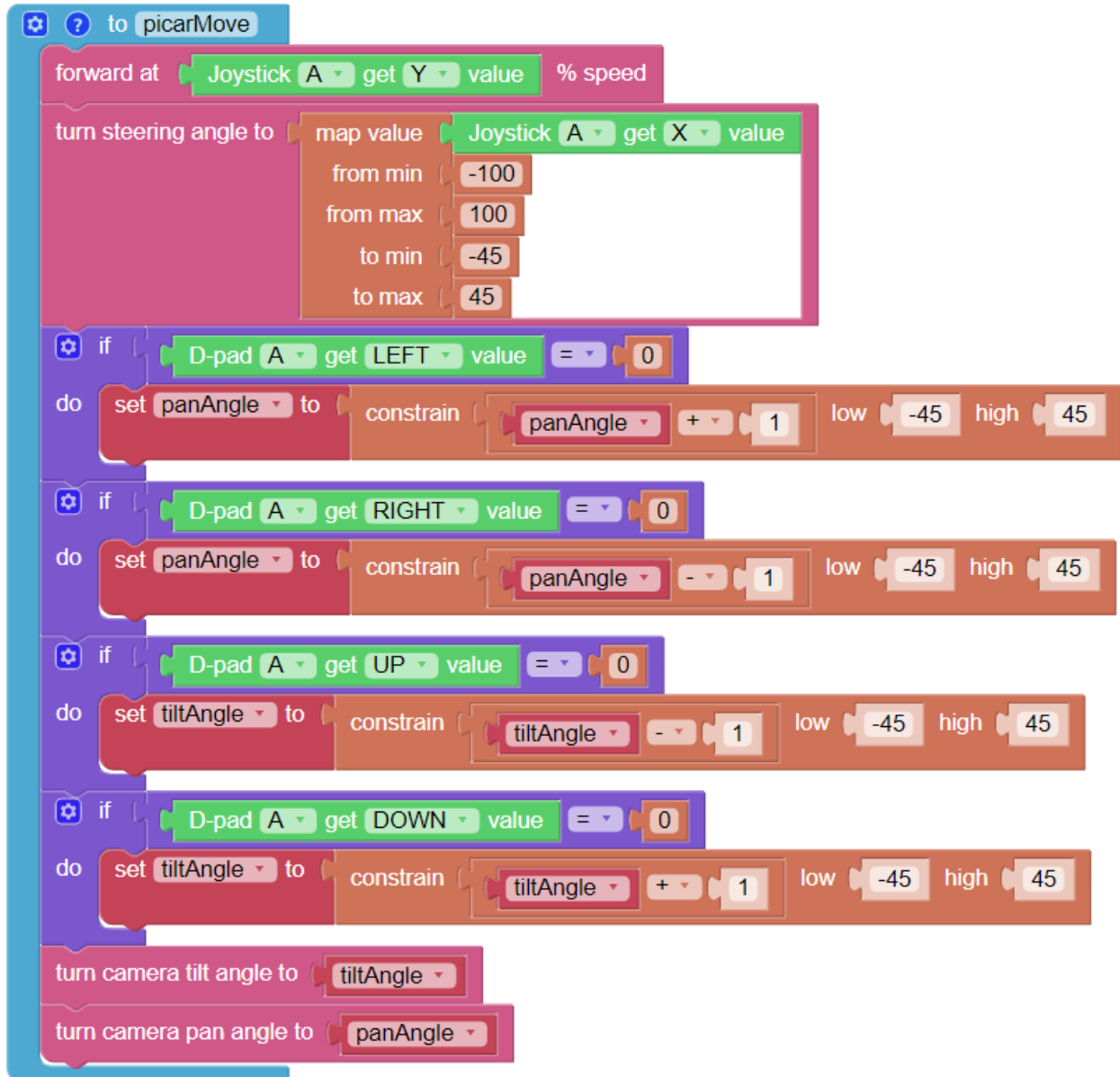


Take turns playing this game with friends to see who can help PiCar-X complete the objective the fastest!

EXAMPLE







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:

5.1 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.

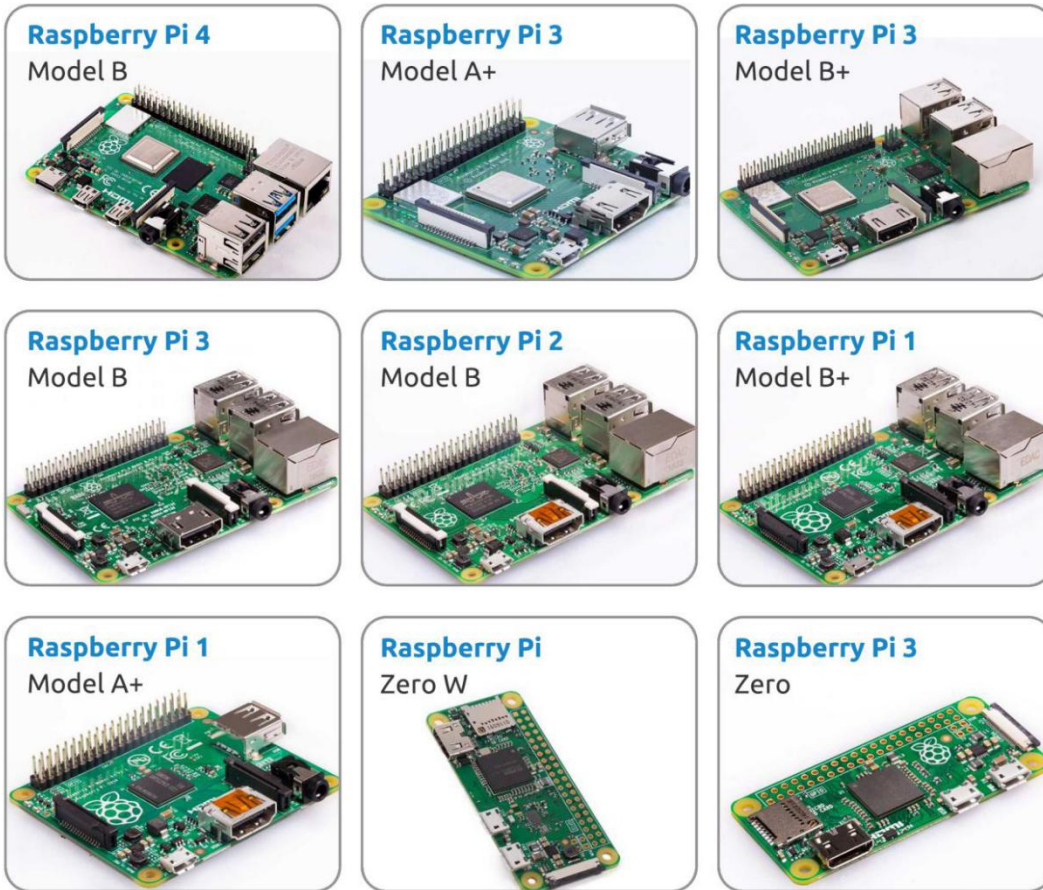
5.1.1 What Do We Need?

Required Components

Raspberry Pi

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

Our kit applies to the following versions of the product of Raspberry Pi



Power Adapter

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

Micro SD Card

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

Optional Components

Screen

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

Mouse & Keyboard

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

HDMI

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

Case

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

Sound or Earphone

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

5.1.2 Installing the OS

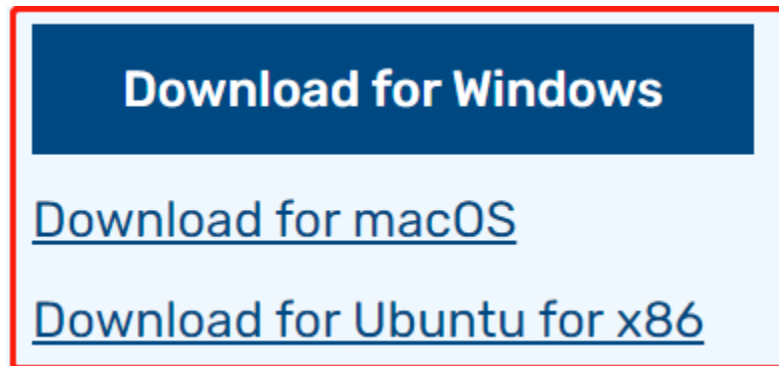
Required Components

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.

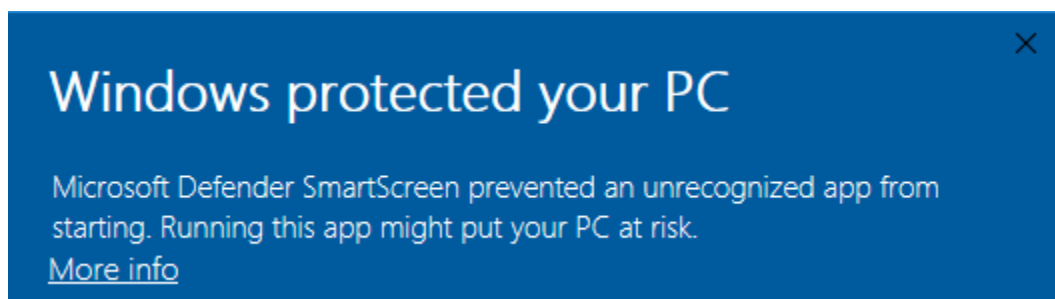
Visit the download page: <https://www.raspberrypi.org/software/>. Click on the link for the Raspberry Pi Imager that matches your operating system, when the download finishes, click it to launch the installer.



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.



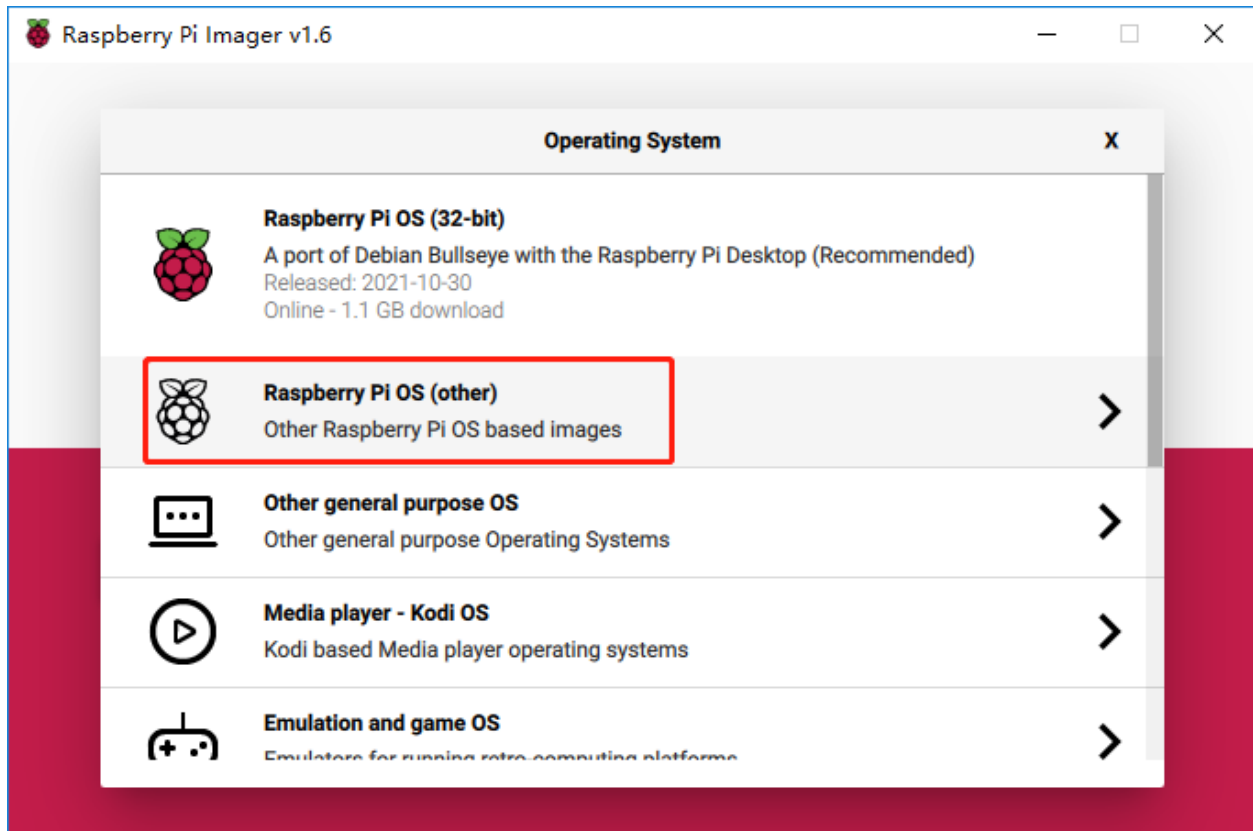
Step 3

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

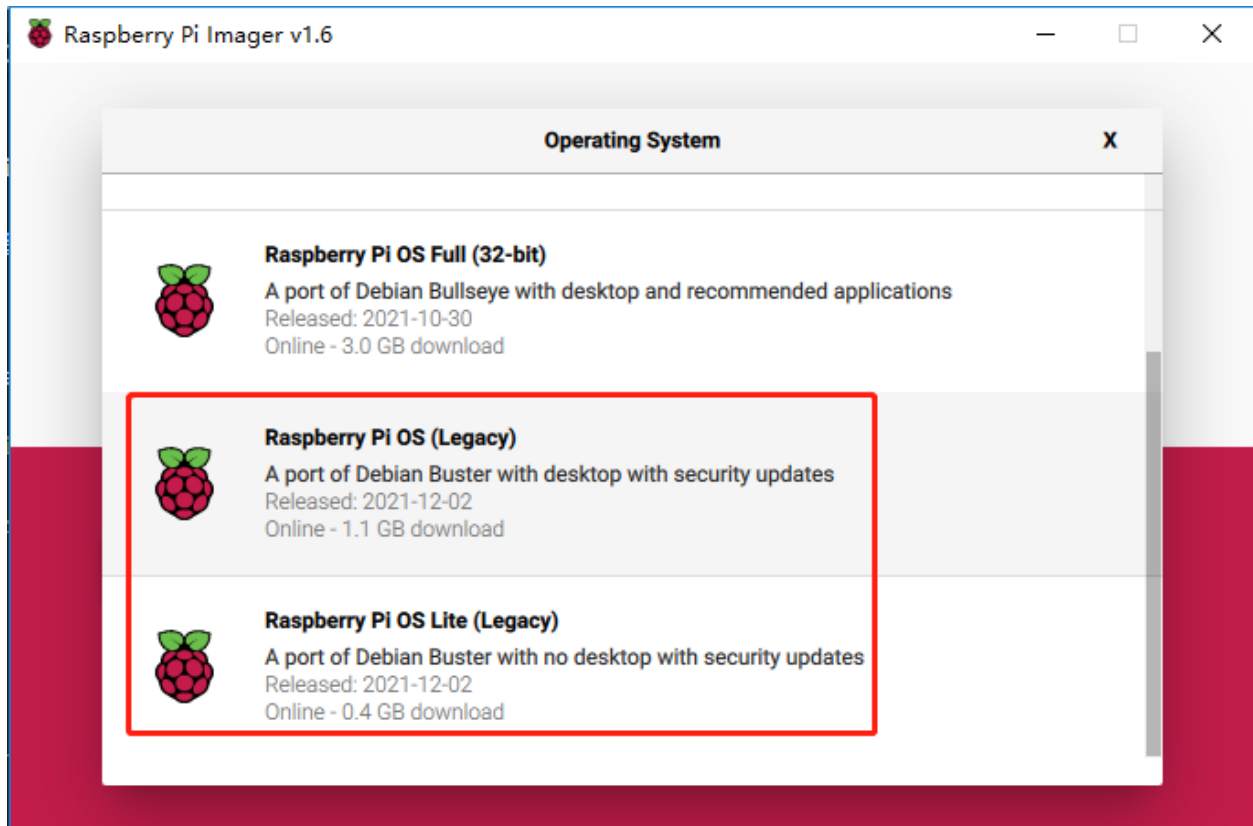
Step 4

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)**.

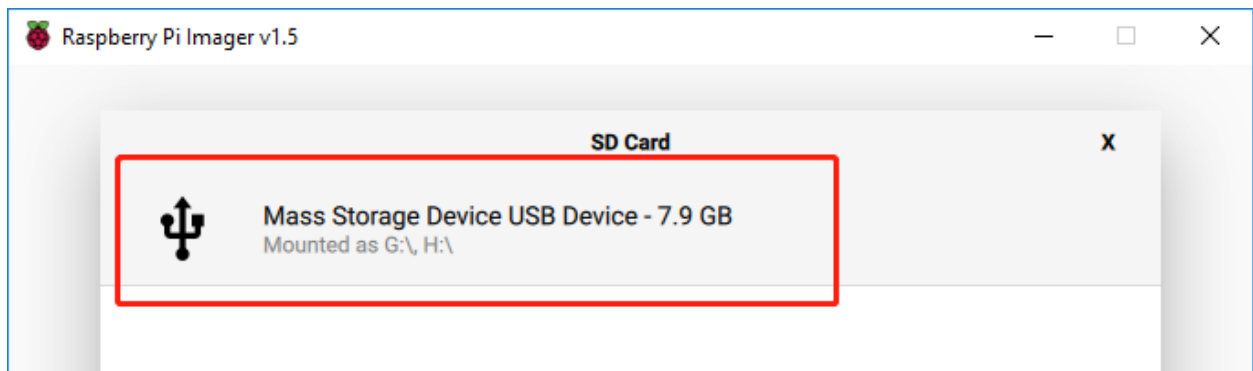


Scroll down to the end of the newly opened page and you will see **Raspberry Pi OS(Legacy)** and **Raspberry Pi OS Lite(Legacy)**, these are security updates for Debian Buster, the difference between them is with or without the desktop. It is recommended to install **Raspberry Pi OS(Legacy)**, the system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



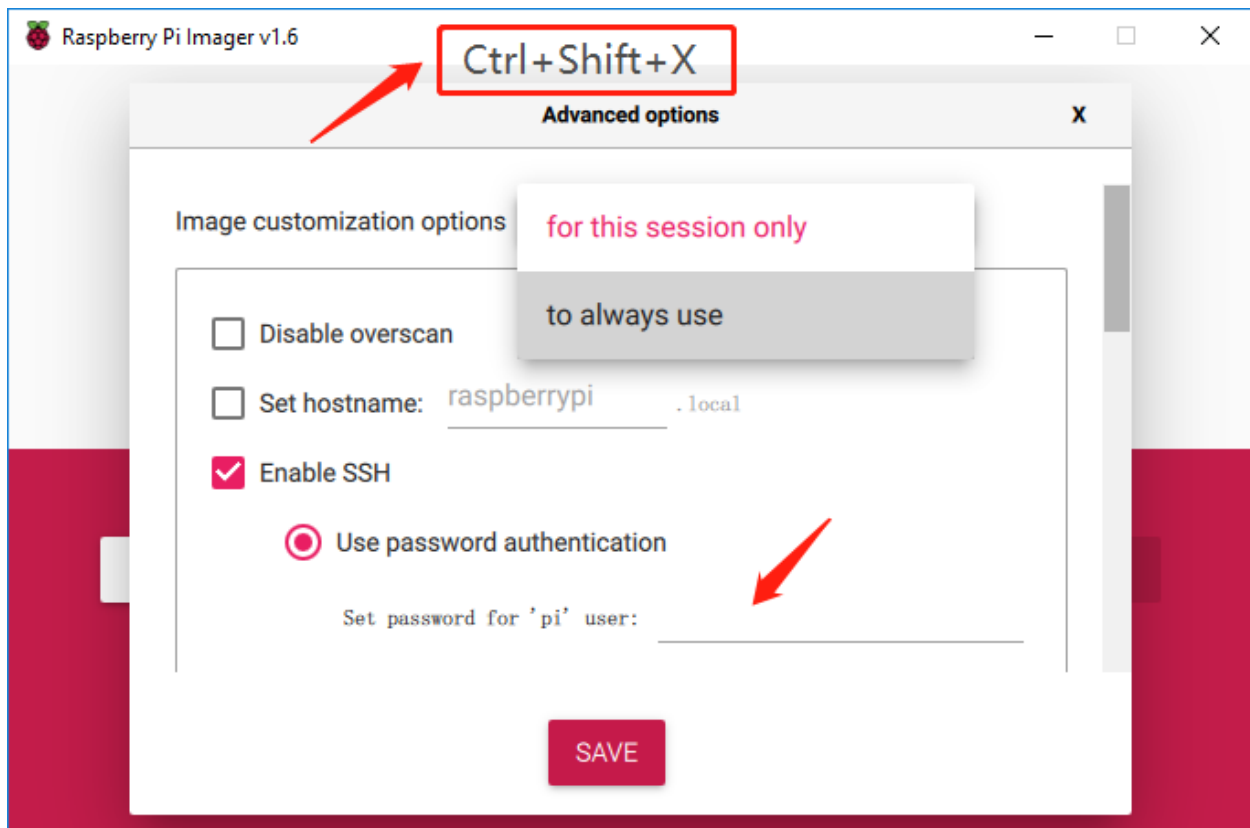
Step 5

Select the SD card you are using.



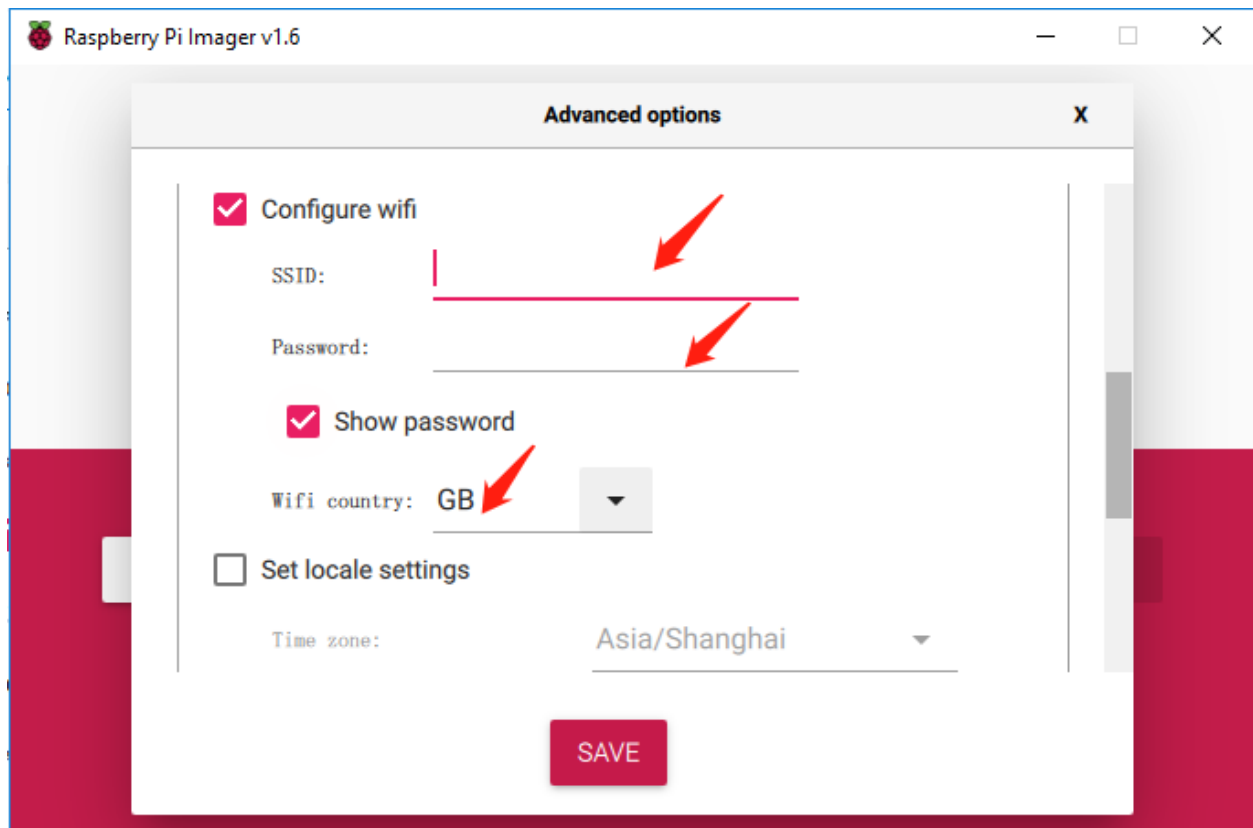
Step 6

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



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

Note: **wifi country** should be set the two-letter [ISO/IEC alpha2 code](https://en.wikipedia.org/wiki/ISO_3166-1_alpha-2#Officially_assigned_code_elements) 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

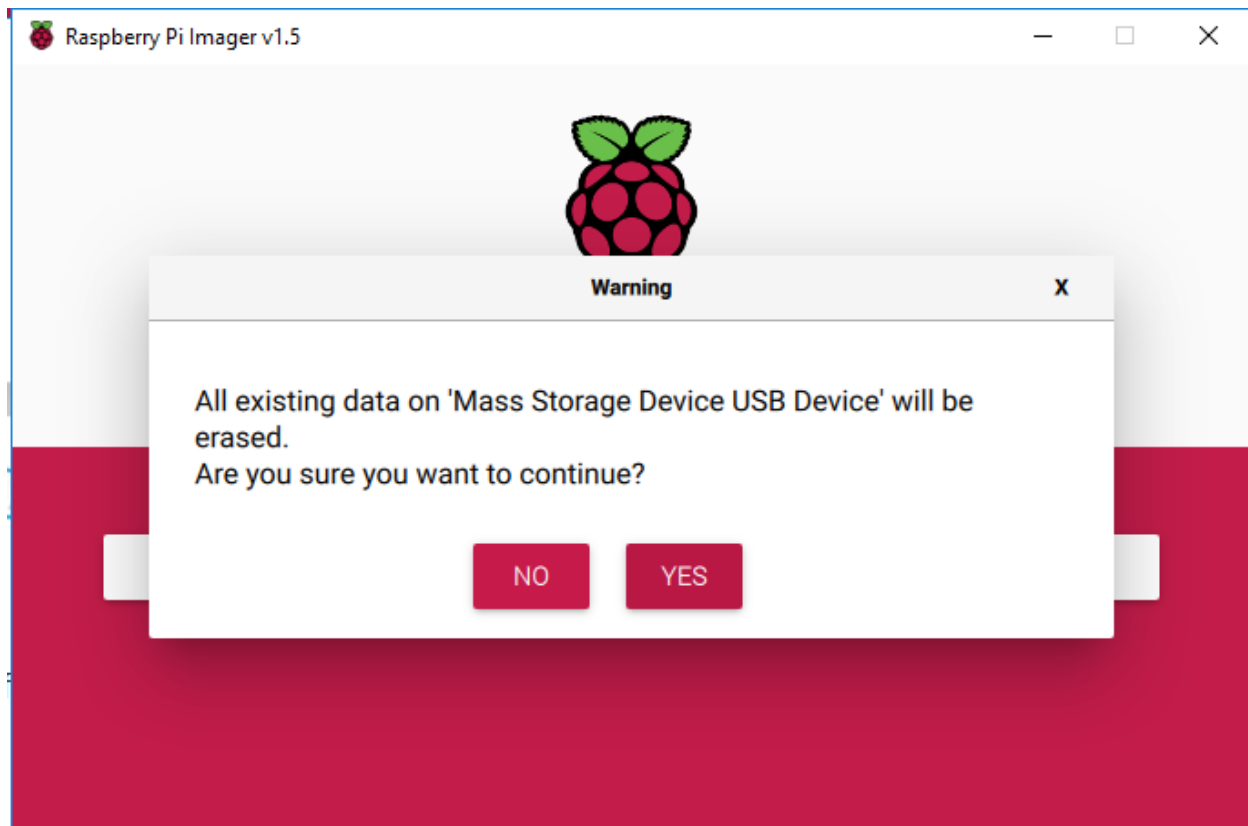


Step 7

Click the **WRITE** button.

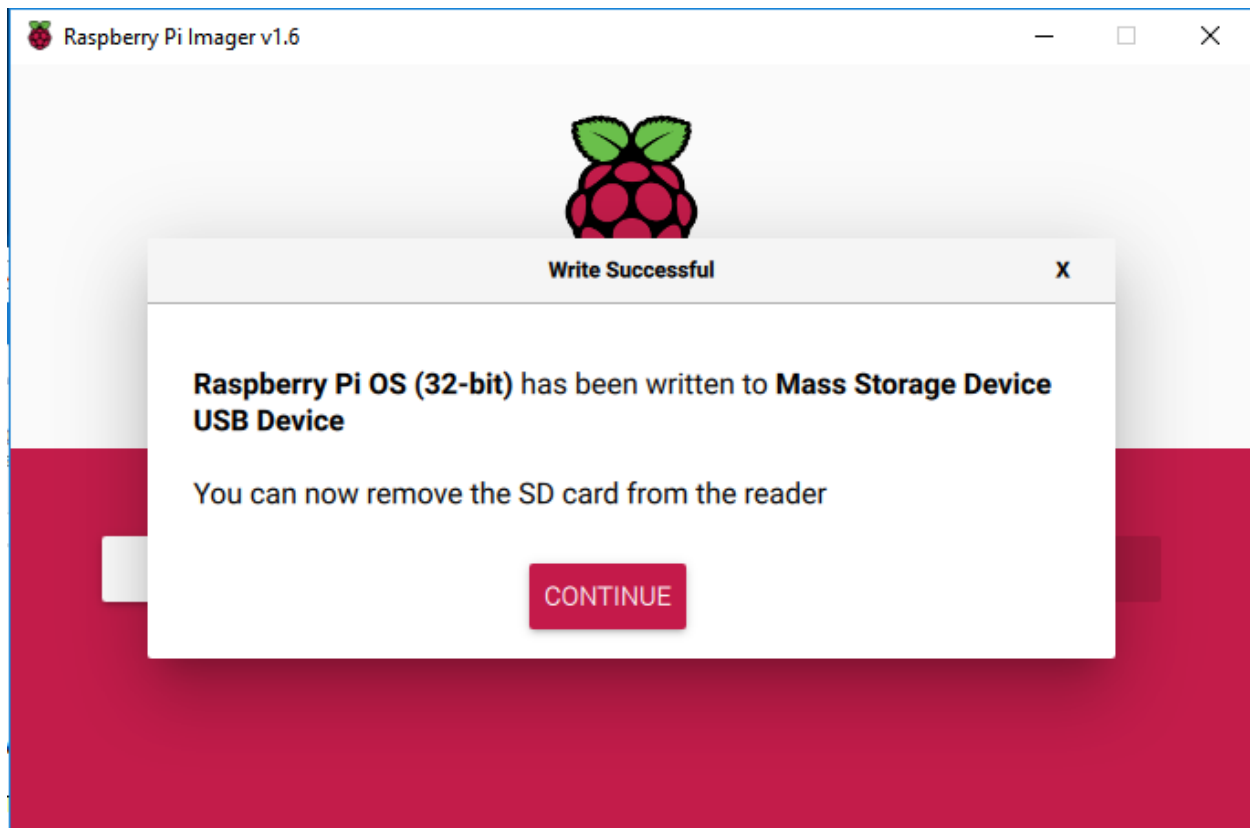
**Step 8**

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.



5.1.3 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.

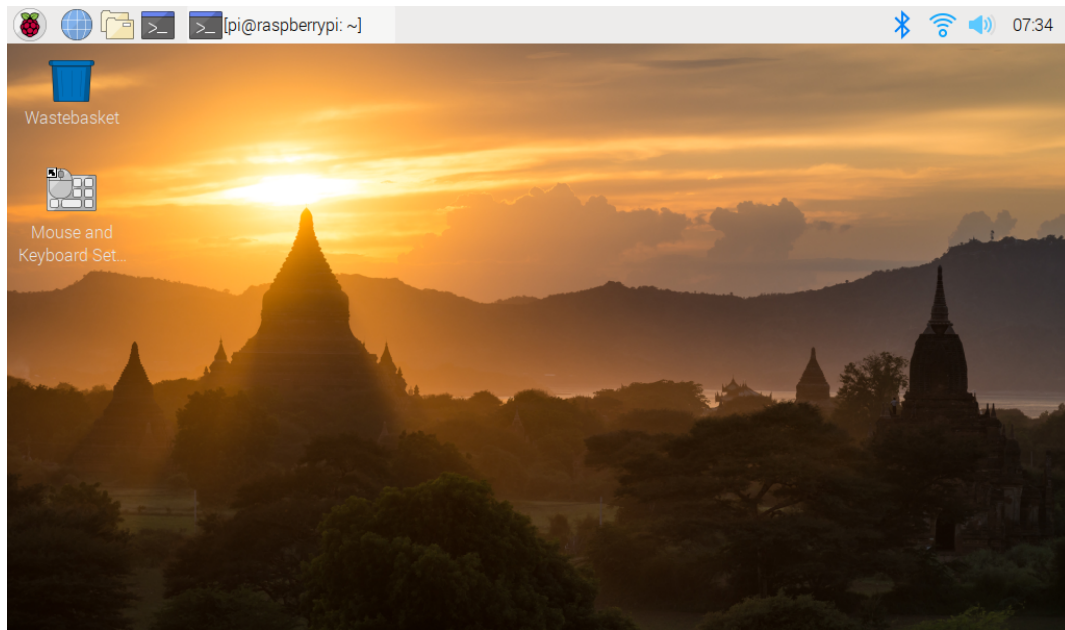
Required Components

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.

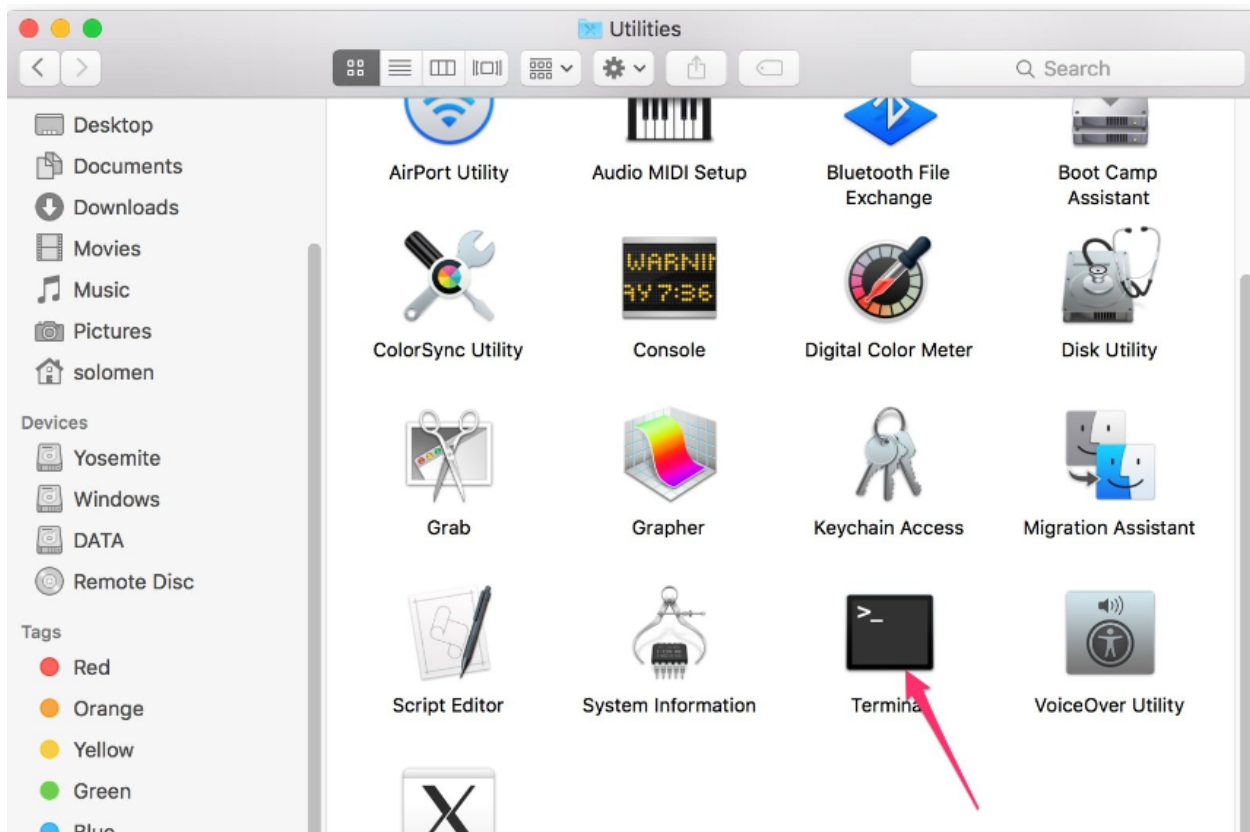
Use the SSH Remote Control

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.



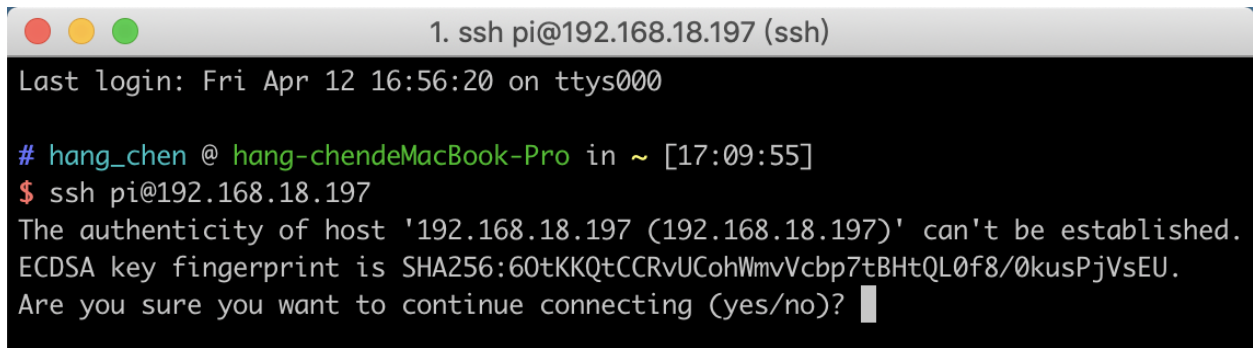
Step 2

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”.

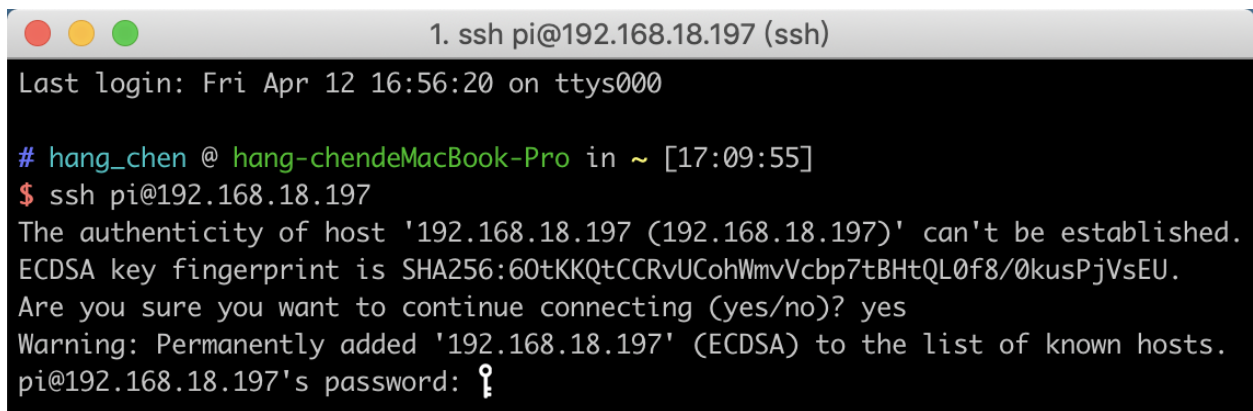


```
1. ssh pi@192.168.18.197 (ssh)
Last login: Fri Apr 12 16:56:20 on ttys000

# 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**.

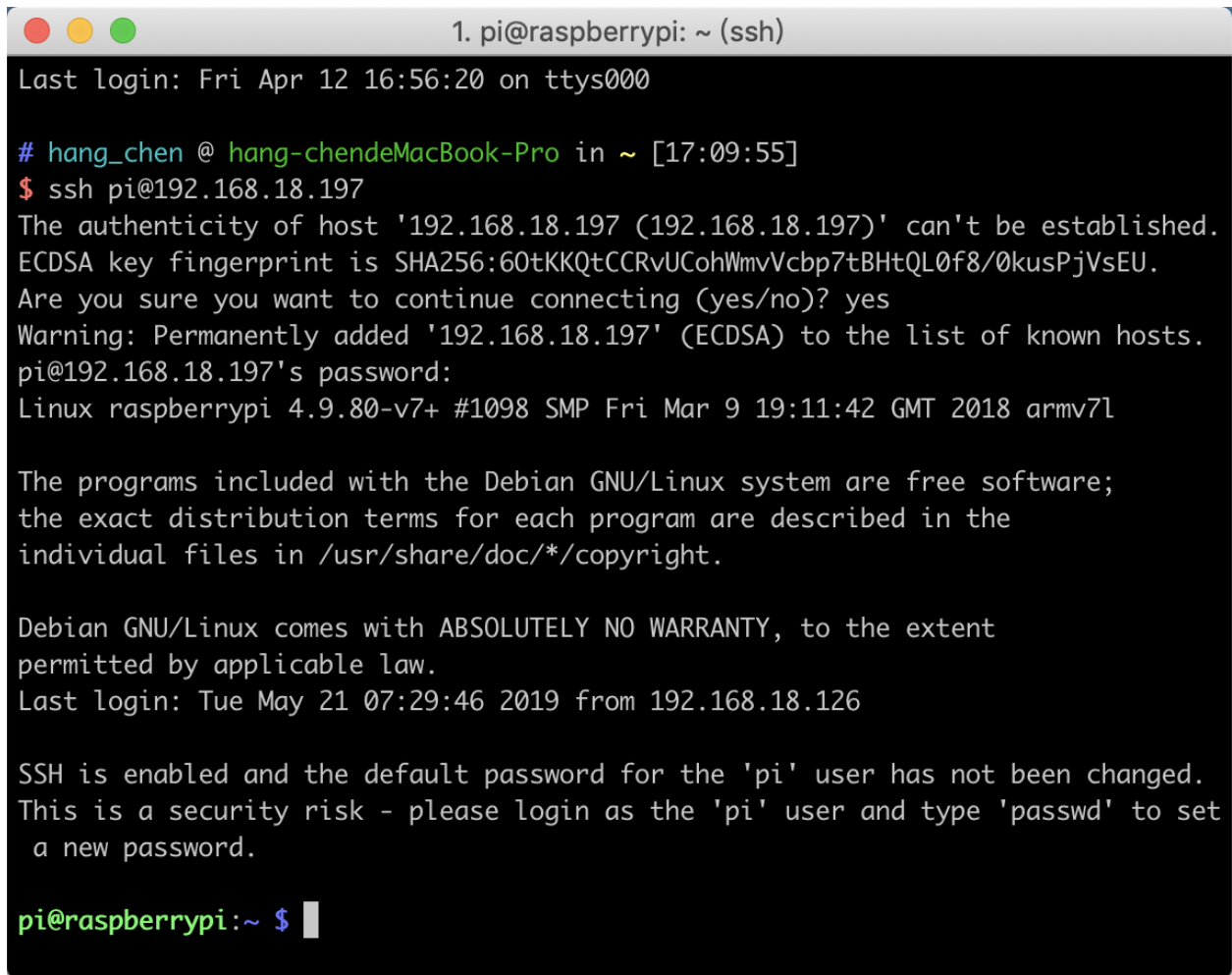


```
1. ssh pi@192.168.18.197 (ssh)
Last login: Fri Apr 12 16:56:20 on ttys000

# 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

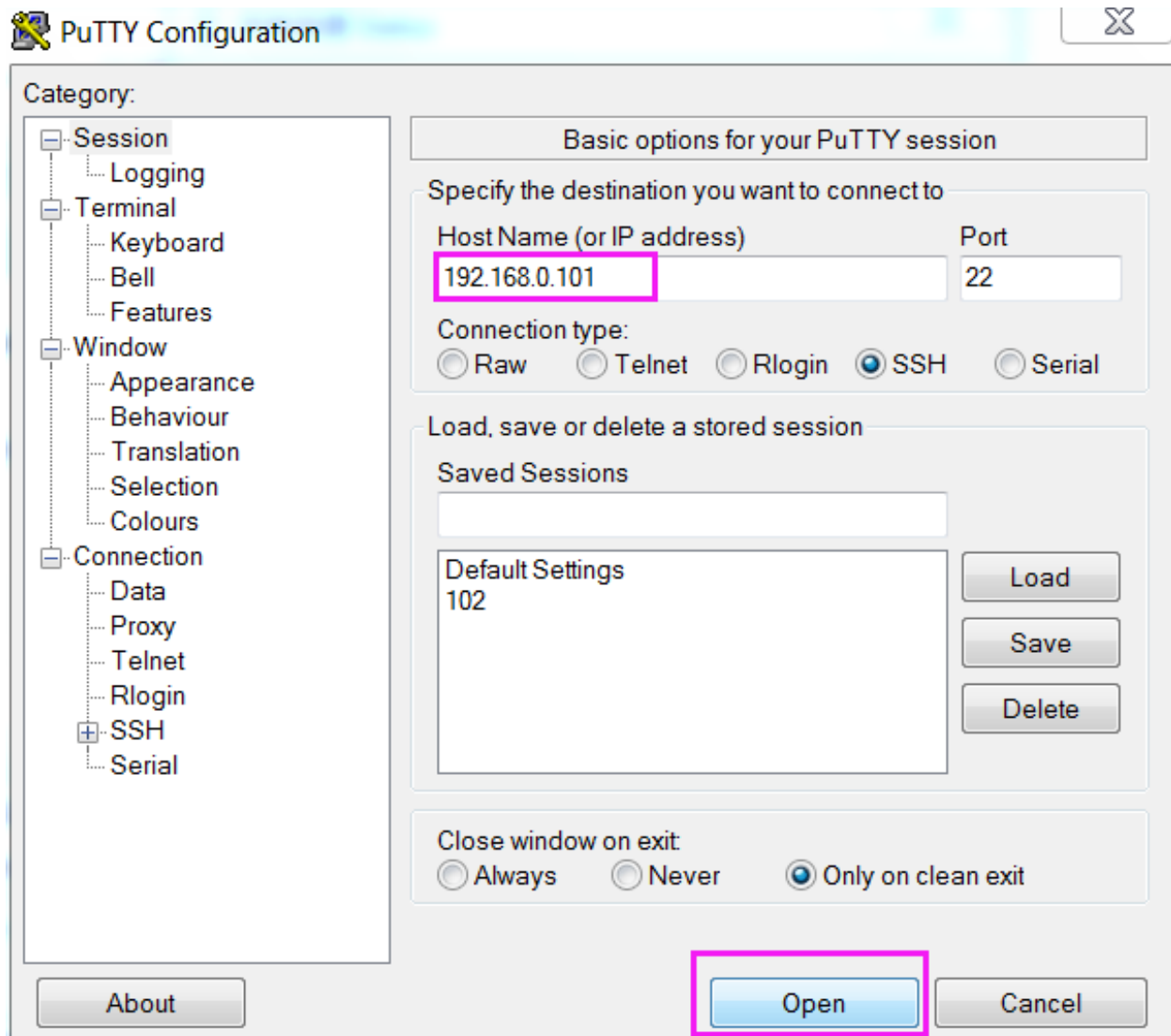
If you're a Windows user, you can use SSH with the application of some software. Here, we recommend **PuTTY**.

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

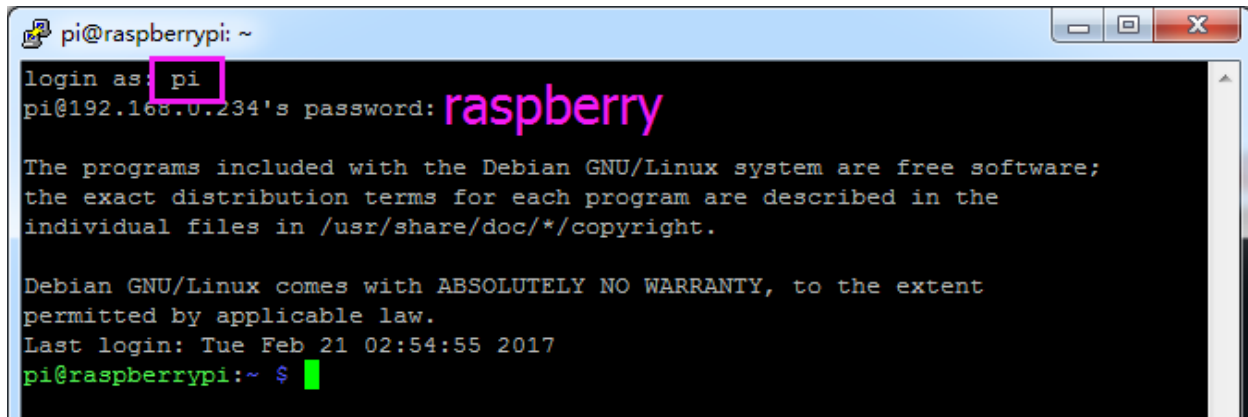
Click **Open**. Note that when you first log in to the Raspberry Pi with the IP address, there prompts a security reminder. Just click **Yes**.

Step 4

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

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

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



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

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

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

Step 5

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

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](#).

5.1.4 Download and Run the Code

Download the PiCar-X files by using `git clone` in the command line.

First change the directory to `/home/pi/` via `cd` command .

```
cd /home/pi/
```

Then clone the repository from github via `git clone` command .

```
git clone -b v2.0 https://github.com/sunfounder/picar-x.git
```

Run install.py

Enter the following two commands to run the `install.py` file in the `picar-x` folder.

```
cd picar-x
```

```
sudo python3 install.py
```

The `install.py` file will finish the installation of the required Python libraries, and configure the Raspberry Pi for PiCar-X operations.

```

pi@raspberrypi: ~/picar-x
File Edit Tabs Help
pi@raspberrypi:~ $ cd picar-x
pi@raspberrypi:~/picar-x $ sudo python3 install.py
EzBlock service install process starts
Install dependency
- update apt-get... Done
- install python3-pip... Done
- install i2c-tools... Done
- install espeak... Done
- install wiringpi... Done
- install python3-pyaudio... Done
- install libatlas-base-dev... Done
- install libjasper-dev... Done
- install libqt4-test... Done
- install libwebp6... Done
- install libtiff5... Done
- install libopenexr23...

```

Warning: The `install.py` program may encounter some **Errors** due to network connectivity. If there is an error prompt, please check the network and re-run `install.py` until all processes show **Done** and the prompt **Finished** appears at the end.

This step can take several minutes. After the file is fully executed and the prompt **Finished** appears, please restart the Raspberry Pi.

```
sudo reboot
```

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

5.2 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.

```
import sys
sys.path.append(r'/home/pi/picar-x/lib')
from utils import reset_mcu
```

(continues on next page)

(continued from previous page)

```

reset_mcu()

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)

    finally:
        px.forward(0)

```

How it works?

The basic functions of PiCar-X are in the `picarx.py` program file, under the path `/home/pi/picar-x/lib`. The `picarx.py` program is used for controlling the servo 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.

```

import sys
sys.path.append(r'/home/pi/picar-x/lib')
from utils import reset_mcu
reset_mcu()

```

(continues on next page)

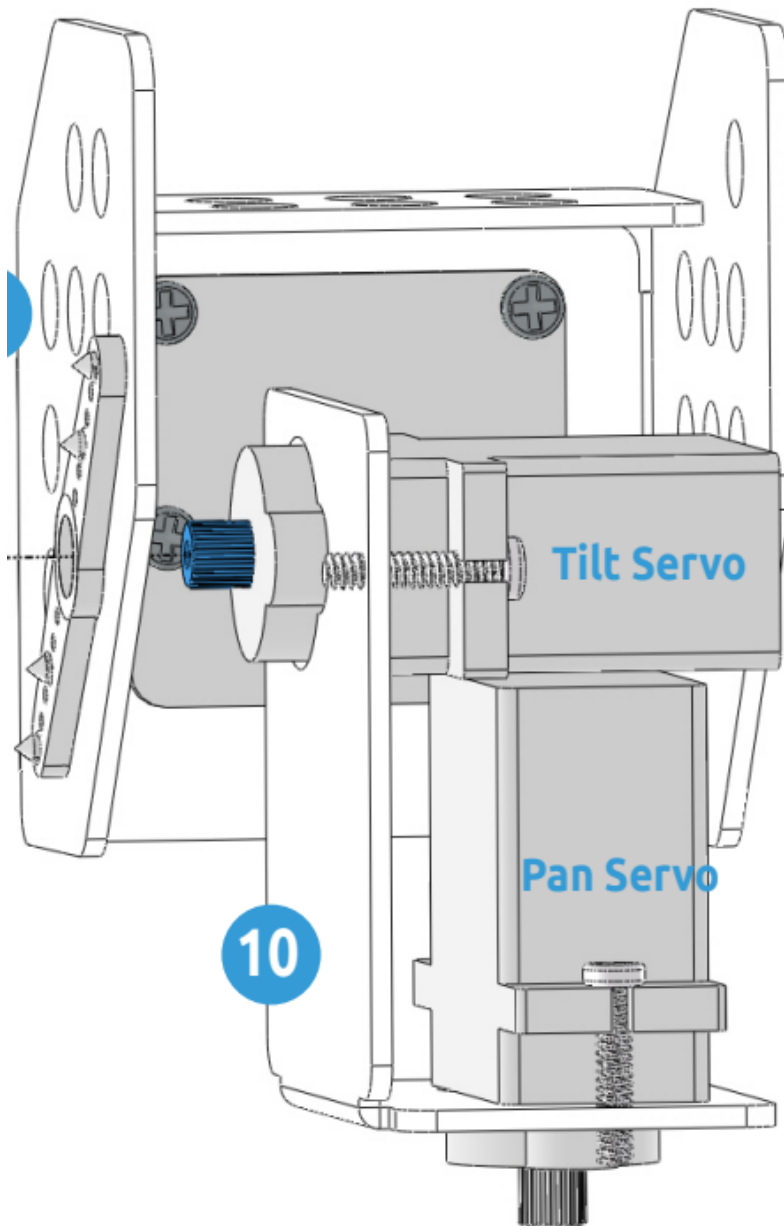
(continued from previous page)

```
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.



5.3 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.

```
import sys
sys.path.append(r'/home/pi/picar-x/lib')
from utils import reset_mcu
reset_mcu()

from picarx import Picarx
from ultrasonic import Ultrasonic
from pin import Pin

if __name__ == "__main__":
    try:
        trig_pin = Pin("D2")
        echo_pin = Pin("D3")
        sonar = Ultrasonic(trig_pin, echo_pin)
        px = Picarx()
        px.forward(30)
        while True:
            distance = sonar.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)
```

How it works?

The distance function is established by importing the Ultrasonic library.

```
from ultrasonic import Ultrasonic
```

Then the pins on the ultrasonic module are initialized.

```
trig_pin = Pin("D2")
echo_pin = Pin("D3")
sonar = Ultrasonic(trig_pin, echo_pin)
```

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

(continues on next page)

(continued from previous page)

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

5.4 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 mincart_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.

```
import sys
sys.path.append(r'/home/pi/picar-x/lib')
from utils import reset_mcu
reset_mcu()
from grayscale_module import Grayscale_Module
from picarx import Picarx

if __name__ == '__main__':
    try:
        gm = Grayscale_Module(500)
        px = Picarx()
        px_power = 10
        while True:
            gm_val_list = gm.get_grayscale_data()
            print("gm_val_list:", gm_val_list)
            gm_status = gm.get_line_status(gm_val_list)
            print("gm_status:", gm_status)

            if gm_status == 'forward':
                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()
```

(continues on next page)

(continued from previous page)

```
finally:  
    px.stop()
```

How it works?

To use the grayscale sensor module, import the `Grayscale_Module` library.

The `Grayscale_Module` library has two methods:

- `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: when generating the object, a number is assigned as a threshold. For example, `gm = Grayscale_Module(500)` will generate a `gm` object with a threshold of 500. When the detection value of all three probes is 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.

5.5 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.

```

import sys
sys.path.append(r'/home/pi/picar-x/lib')
from tts import TTS

if __name__ == "__main__":
    words = ["Hello", "Hi", "Good bye", "Nice to meet you"]
    tts_robot = TTS()
    tts_robot.lang("en-US")
    for i in words:

```

(continues on next page)

(continued from previous page)

```
print(i)
tts_robot.say(i)
```

How it works?

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

Use `lang("")` to set the language, and `say("")` for text audio.

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

Table 1: Language

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

5.6 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

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

Code

```
# coding=utf-8
import cv2
from picamera.array import PiRGBArray
from picamera import PiCamera

#init camera
camera = PiCamera()
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

```

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

camera = PiCamera()
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

camera = PiCamera()
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
import numpy as np

# Create the in-memory stream
stream = io.BytesIO()
with picamera.PiCamera() as camera:
    camera.start_preview()
    time.sleep(2)

```

(continues on next page)

(continued from previous page)

```

    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
camera = PiCamera()
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.
    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](#).

5.7 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

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

Code

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

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

(continues on next page)

(continued from previous page)

```

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

def color_detect(img,color_name):

    resize_img = cv2.resize(img, (160,120), interpolation=cv2.INTER_LINEAR)
    hsv = cv2.cvtColor(resize_img, cv2.COLOR_BGR2HSV)

    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]) )
    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)

    contours, hierarchy = cv2.findContours(morphologyEx_img,cv2.RETR_EXTERNAL,cv2.
↪CHAIN_APPROX_SIMPLE)

    color_area_num = len(contours)

    if color_area_num > 0:
        for i in contours:
            x,y,w,h = cv2.boundingRect(i)

            if w >= 8 and h >= 8:
                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)
                cv2.putText(img,color_type,(x,y), cv2.FONT_HERSHEY_SIMPLEX, 1,(0,0,
↪255),2)

        return img,mask,morphologyEx_img

#init camera
print("start color detect")
camera = PiCamera()
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):
    img = frame.array
    img,img_2,img_3 = color_detect(img,'red')
    cv2.imshow("video", img)
    cv2.imshow("mask", img_2)
    cv2.imshow("morphologyEx_img", img_3)
    rawCapture.truncate(0)

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

```

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 processes 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):

    resize_img = cv2.resize(img, (160,120), interpolation=cv2.INTER_LINEAR) # To_
↪reduce the amount of calculation, the image size is reduced.
    hsv = cv2.cvtColor(resize_img, cv2.COLOR_BGR2HSV) # Convert color from BGR to HSV

    color_type = color_name

    ### Extract the data of the target color as a new binary image (array).
    mask = cv2.inRange(hsv,np.array([min(color_dict[color_type]), 60, 60]), np.
↪array([max(color_dict[color_type]), 255, 255]))
    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) # In HSV, red is divided into two_
↪sections, which need to be combined.

    ### Performs advanced morphological transformations
    morphologyEx_img = cv2.morphologyEx(mask, cv2.MORPH_OPEN, kernel_5,iterations=1) _
↪ # Perform open operation

    ### Finds contours in a binary image.
    contours, hierarchy = cv2.findContours(morphologyEx_img,cv2.RETR_EXTERNAL,cv2.
↪CHAIN_APPROX_SIMPLE)
    color_area_num = len(contours) # Count the number of contours

    if color_area_num > 0:
        for i in contours:
            x,y,w,h = cv2.boundingRect(i) # Let (x,y) be the top-left coordinate of_
↪the rectangle and (w,h) be its width and height.

            ### Draw a frame for the recognized object on the image
            if w >= 8 and h >= 8: # Because the picture is reduced during operation,_
↪the increase now go back
                x = x * 4
                y = y * 4
                w = w * 4
                h = h * 4
```

(continues on next page)

(continued from previous page)

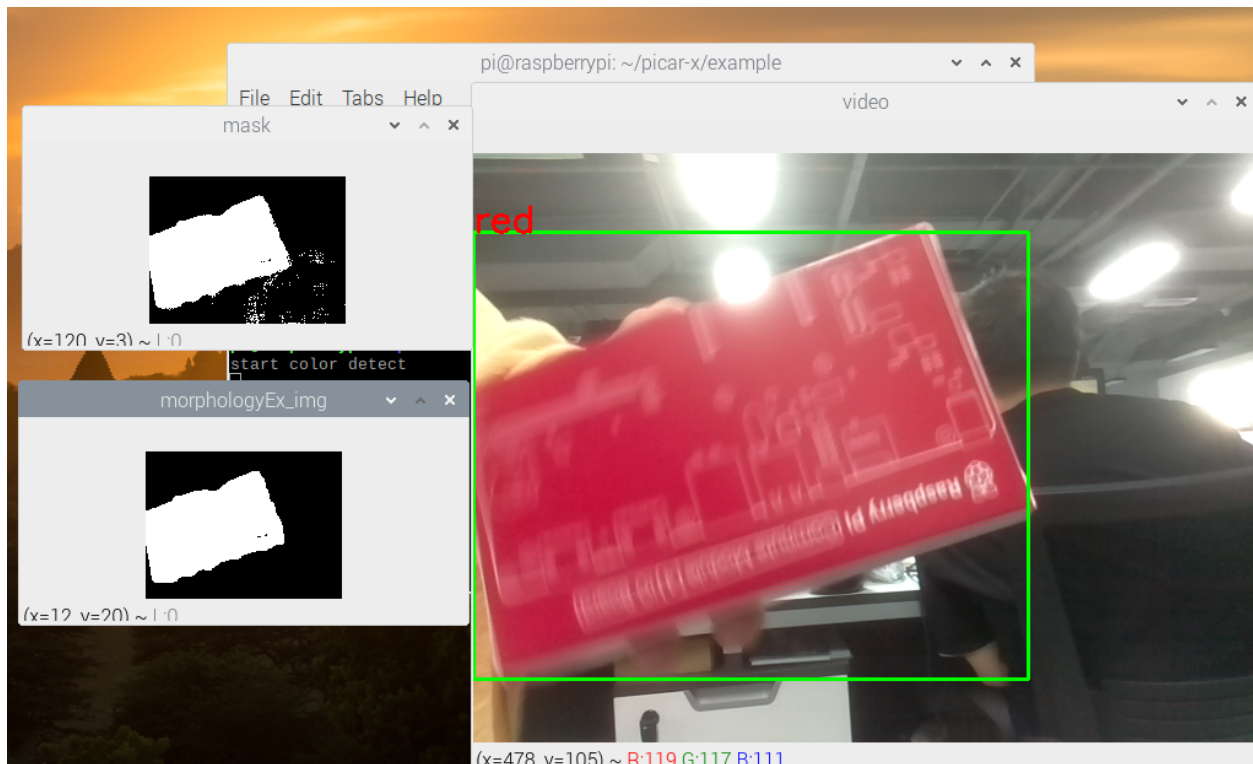
```

        cv2.rectangle(img, (x,y), (x+w,y+h), (0,255,0),2) # Draw a frame
        cv2.putText(img,color_type,(x,y), cv2.FONT_HERSHEY_SIMPLEX, 1,(0,0,
↪255),2) # Add 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.

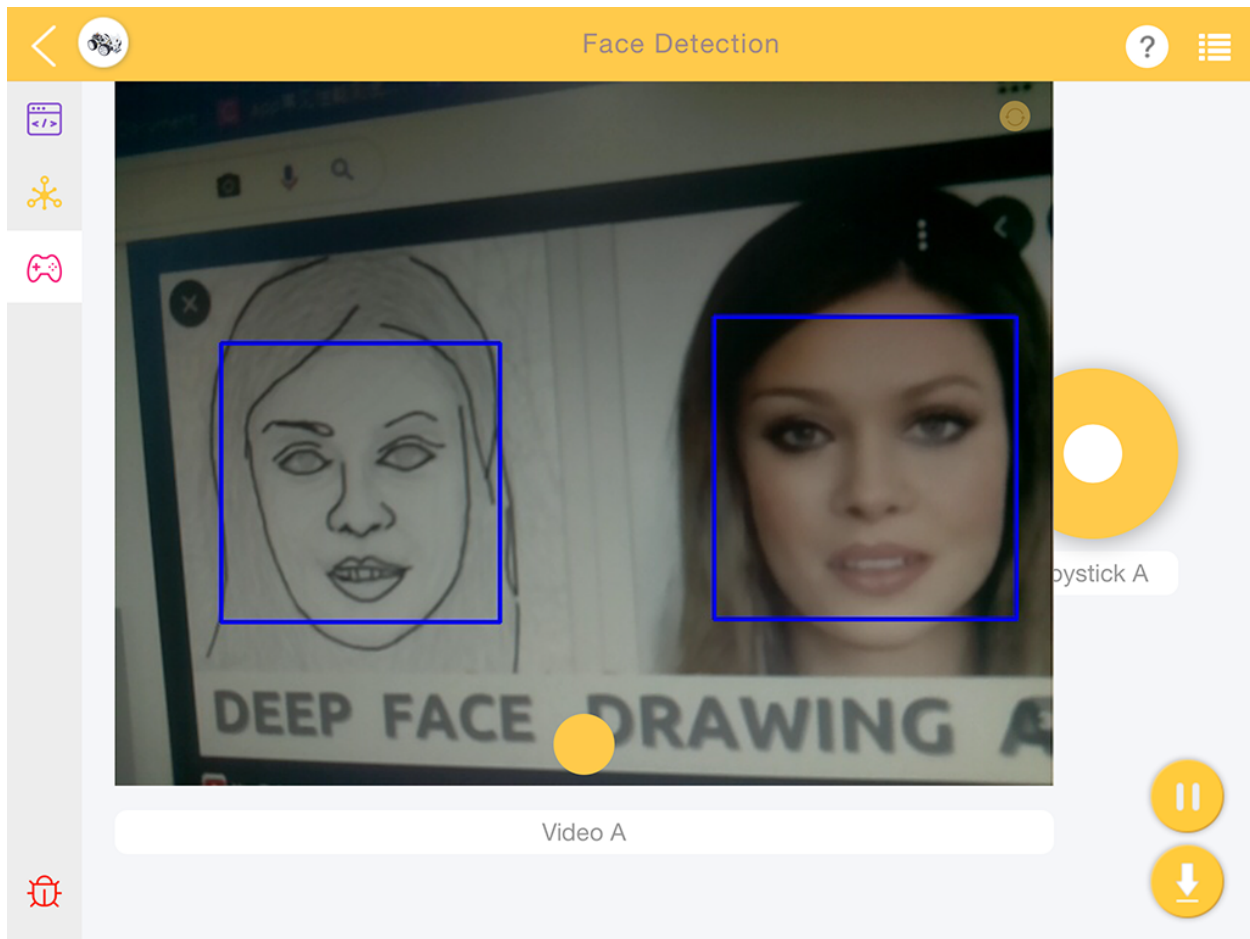


For more information on morphology and contouring, please reference the following resources:

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

5.8 Face Detection

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



Run the Code

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

Code

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

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

def human_face_detect(img):
    resize_img = cv2.resize(img, (320,240), interpolation=cv2.INTER_LINEAR)
    gray = cv2.cvtColor(resize_img, cv2.COLOR_BGR2GRAY)
    faces = face_cascade.detectMultiScale(gray, 1.3, 2)

    face_num = len(faces)
    max_area = 0
    if face_num > 0:
        for (x,y,w,h) in faces:
            x = x*2
            y = y*2
            w = w*2
```

(continues on next page)

(continued from previous page)

```

        h = h*2
        cv2.rectangle(img, (x,y), (x+w,y+h), (255,0,0),2)

    return img

#init camera
print("start human face detect")
camera = PiCamera()
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):
    img = frame.array
    img = human_face_detect(img)
    cv2.imshow("video", img)
    rawCapture.truncate(0)

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

```

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.

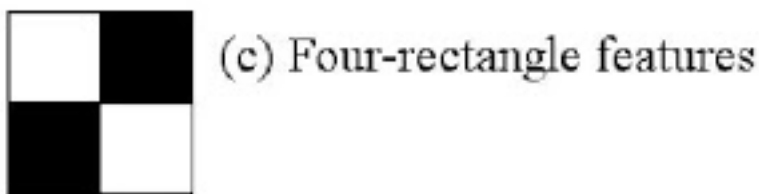
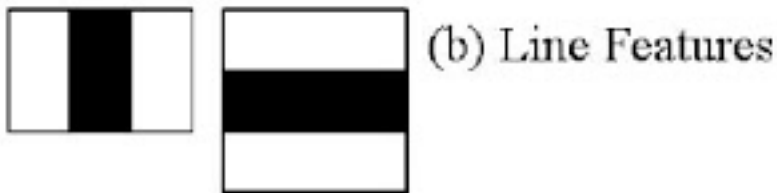
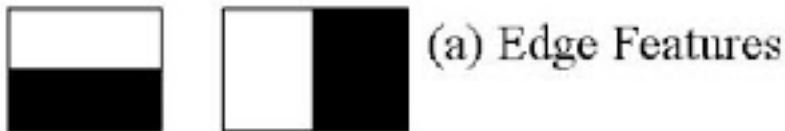
This file is called by **Cascade Classifier** of OpenCV.

```
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.

    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

5.9 Video Car

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

Run the Code

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

code

```
import sys
sys.path.append(r'/home/pi/picar-x/lib')

from utils import reset_mcu
reset_mcu()
from picarx import Picarx
from utils import run_command
import datetime

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

camera = PiCamera()
camera.resolution = (640,480)
camera.framerate = 24
camera.image_effect = "none"  #"none", "negative", "solarize", "emboss", "posterise",
↪ "cartoon",
rawCapture = PiRGBArray(camera, size=camera.resolution)

power_val = 0
px = Picarx()

try:
    while True:
        for frame in camera.capture_continuous(rawCapture, format="bgr", use_video_
↪ port=True): # use_video_port=True
            img = frame.array
            cv2.imshow("video", img)

            k = cv2.waitKey(1) & 0xFF
            if k == 27:
                camera.close()
                continue
            elif k == ord('o'):
                if power_val <=90:
                    power_val += 10
                    print("power_val:",power_val)
            elif k == ord('p'):
                if power_val >=10:
                    power_val -= 10
                    print("power_val:",power_val)
            elif k == ord('w'):
                # print("w:",)
```

(continues on next page)

(continued from previous page)

```

        px.set_dir_servo_angle(0)
        px.forward(power_val)
    elif k == ord('a'):
        px.set_dir_servo_angle(-30)
        px.forward(power_val)
    elif k == ord('s'):
        px.set_dir_servo_angle(0)
        px.backward(power_val)
    elif k == ord('d'):
        px.set_dir_servo_angle(30)
        px.forward(power_val)
    elif k == ord('f'):
        px.stop()

    elif k == ord('t'):
        camera.close()
        break
    rawCapture.truncate(0)

    print("take a photo wait...")
    picture_time = datetime.datetime.now().strftime('%Y-%m-%d_%H-%M-%S')
    picture_path = '/home/pi/Pictures/' + picture_time + '.jpg'

    a_t = "sudo raspistill -t 250 -w 2592 -h 1944 " + " -o " + picture_path

    print(a_t)
    run_command(a_t)

    # Vilib.shuttle_button()
    camera = PiCamera()
    camera.resolution = (640,480)
    camera.framerate = 24
    camera.image_effect = "none" # "none", "negative", "solarize", "emboss",
    → "posterize", "cartoon",
    rawCapture = PiRGBArray(camera, size=camera.resolution)
finally:
    px.stop()
    camera.close()

```

How it works?

Most of this code is described in *Computer Vision* and *Let PiCar-X Move*, with the addition of the OpenCV `waitKey()` function.

```
k = cv2.waitKey(1) & 0xFF
```

The `waitKey()` is a function that waits for key-press events, and is also the only method to obtain and process events in HighGUI. This function will only work if at least one HighGUI window has been created and is active.

- High-level GUI
- `waitKey` - OpenCV

```

print("take a photo wait...")
picture_time = datetime.datetime.now().strftime('%Y-%m-%d_%H-%M-%S')
picture_path = '/home/pi/Pictures/' + picture_time + '.jpg'

```

(continues on next page)

(continued from previous page)

```
a_t = "sudo raspistill -t 250 -w 2592 -h 1944 " + " -o " + picture_path
print(a_t)
run_command(a_t)
```

The purpose of these lines of code is to capture still photos using the Raspberry Pi camera module.

These codes are external to the main loop, and they are executed when OpenCV receives a **T** key on the keyboard, which breaks the main loop.

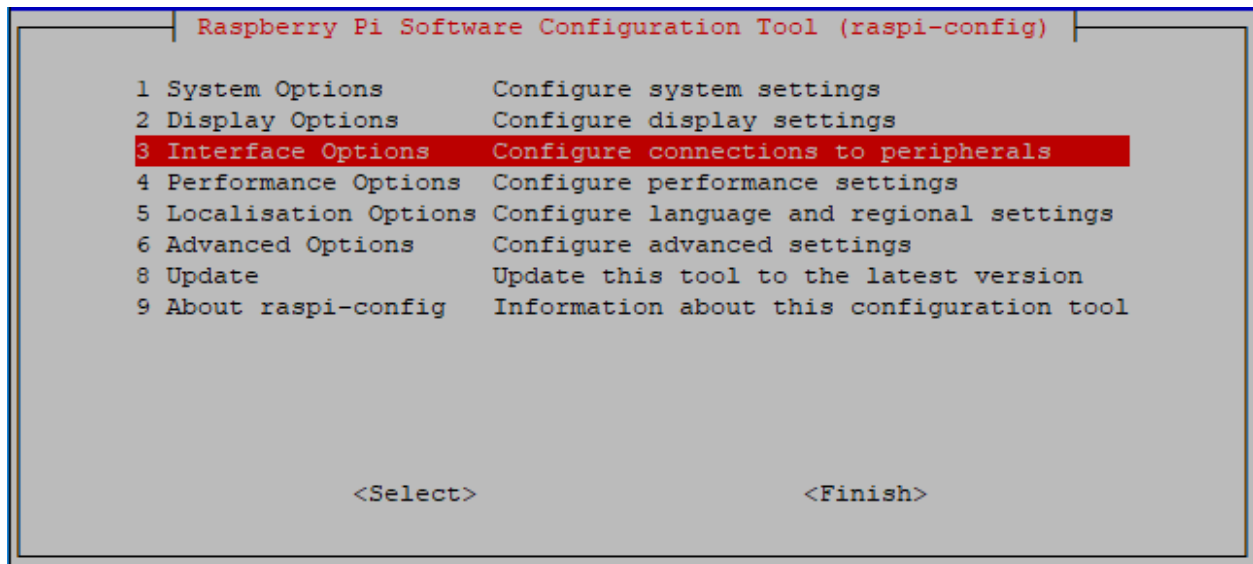
- `raspistill`

6.1 I2C Configuration

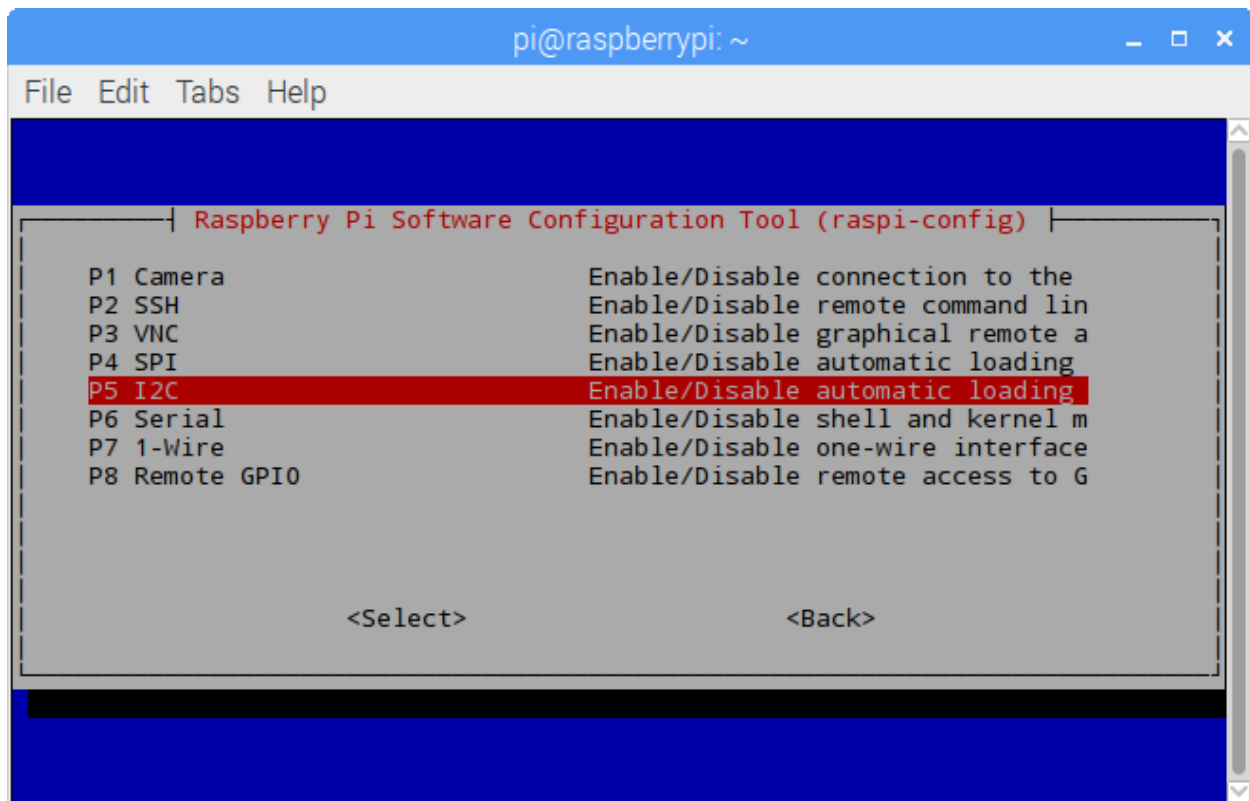
Enable the I2C port of your Raspberry Pi (If you have enabled it, skip this; if you do not know whether you have done that or not, please continue).

```
sudo raspi-config
```

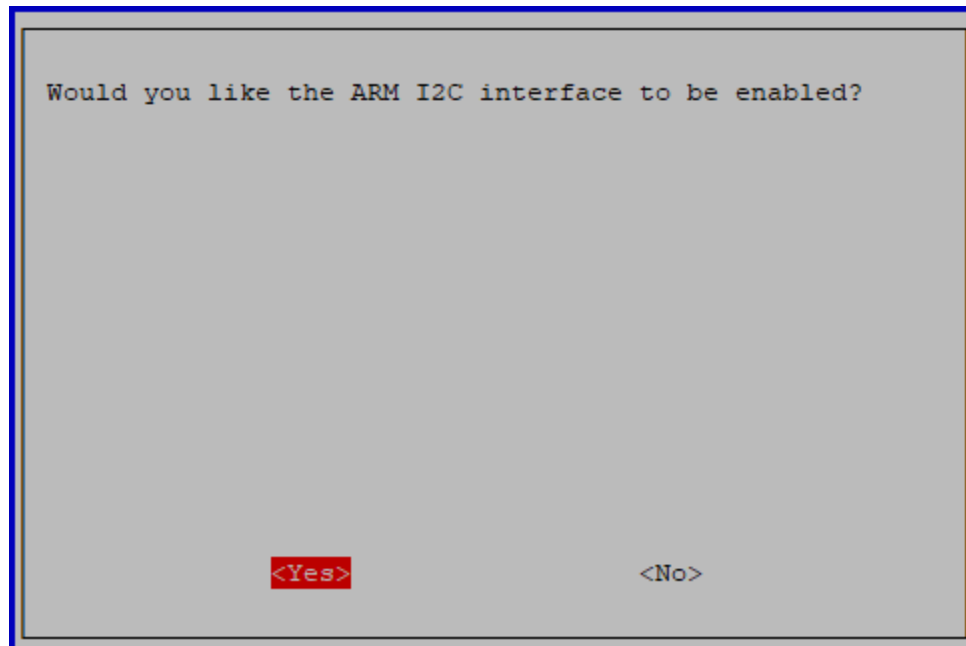
3 Interfacing options



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.

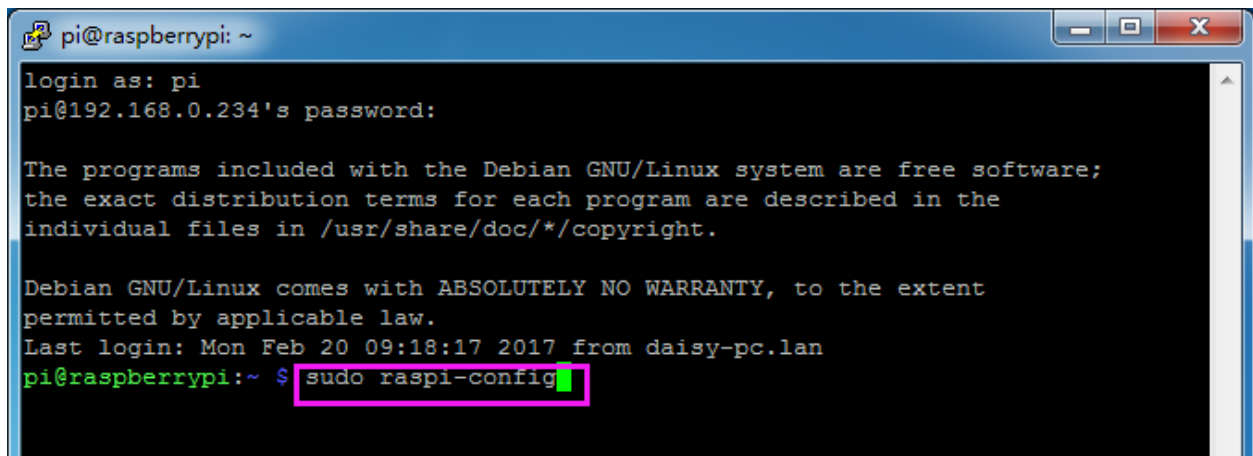
Enable VNC service

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

Step 1

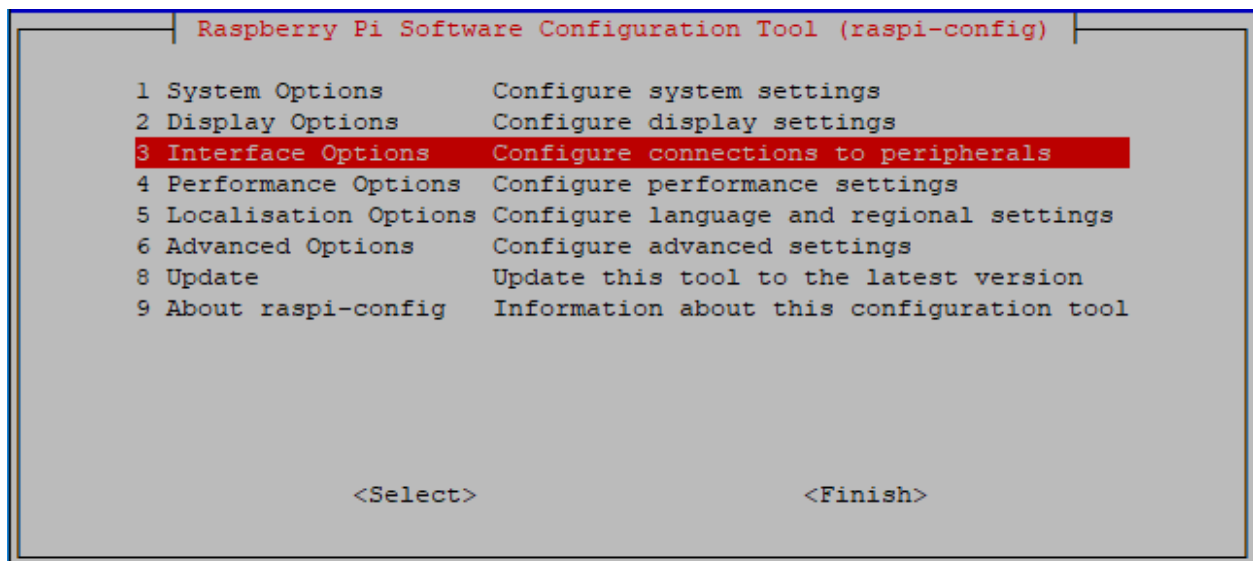
Input the following command:

```
sudo raspi-config
```



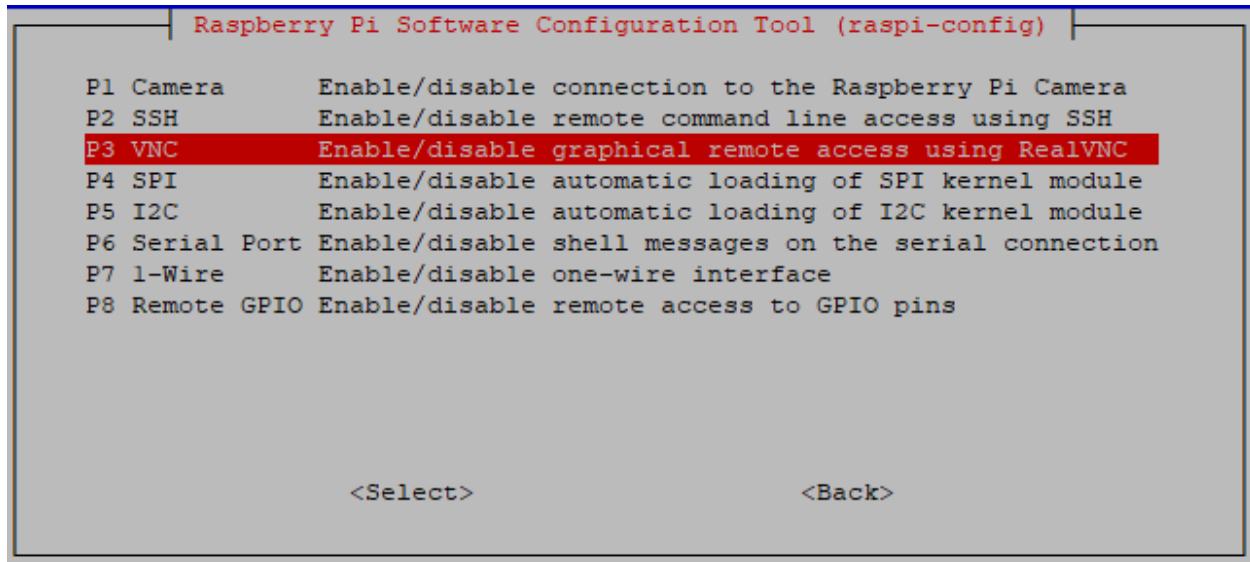
Step 2

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



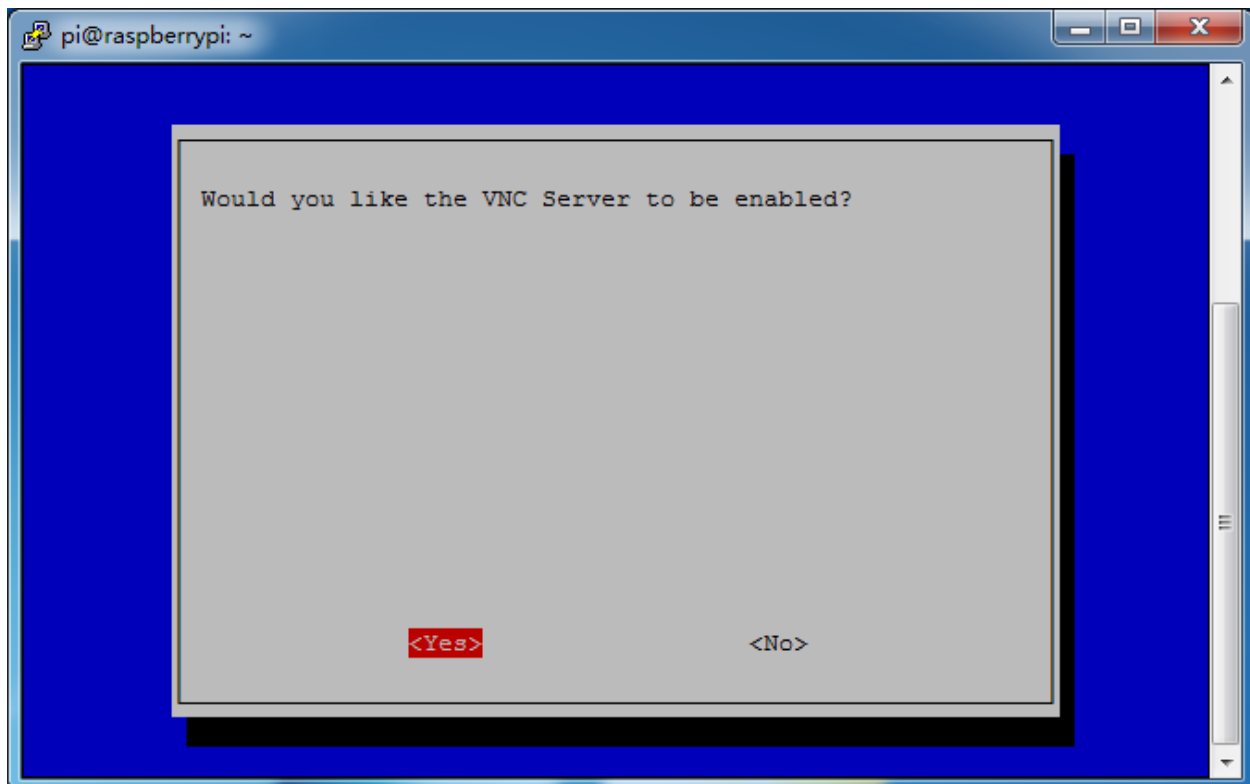
Step 3

P3 VNC



Step 4

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



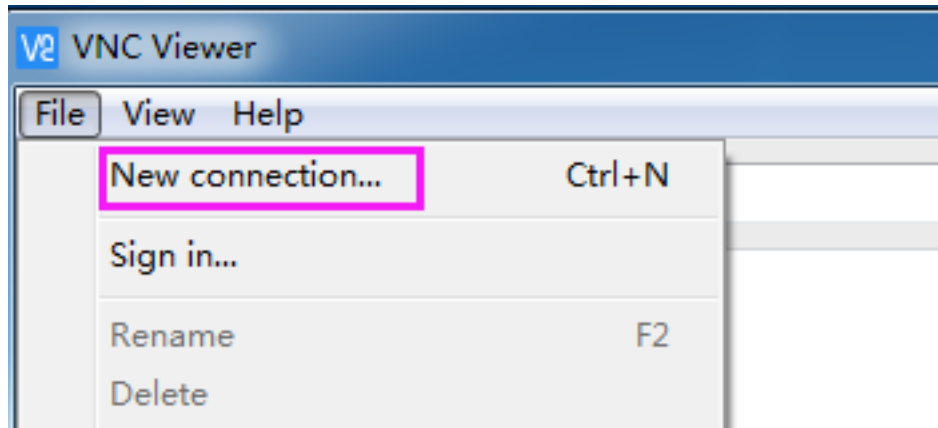
Login to VNC

Step 1

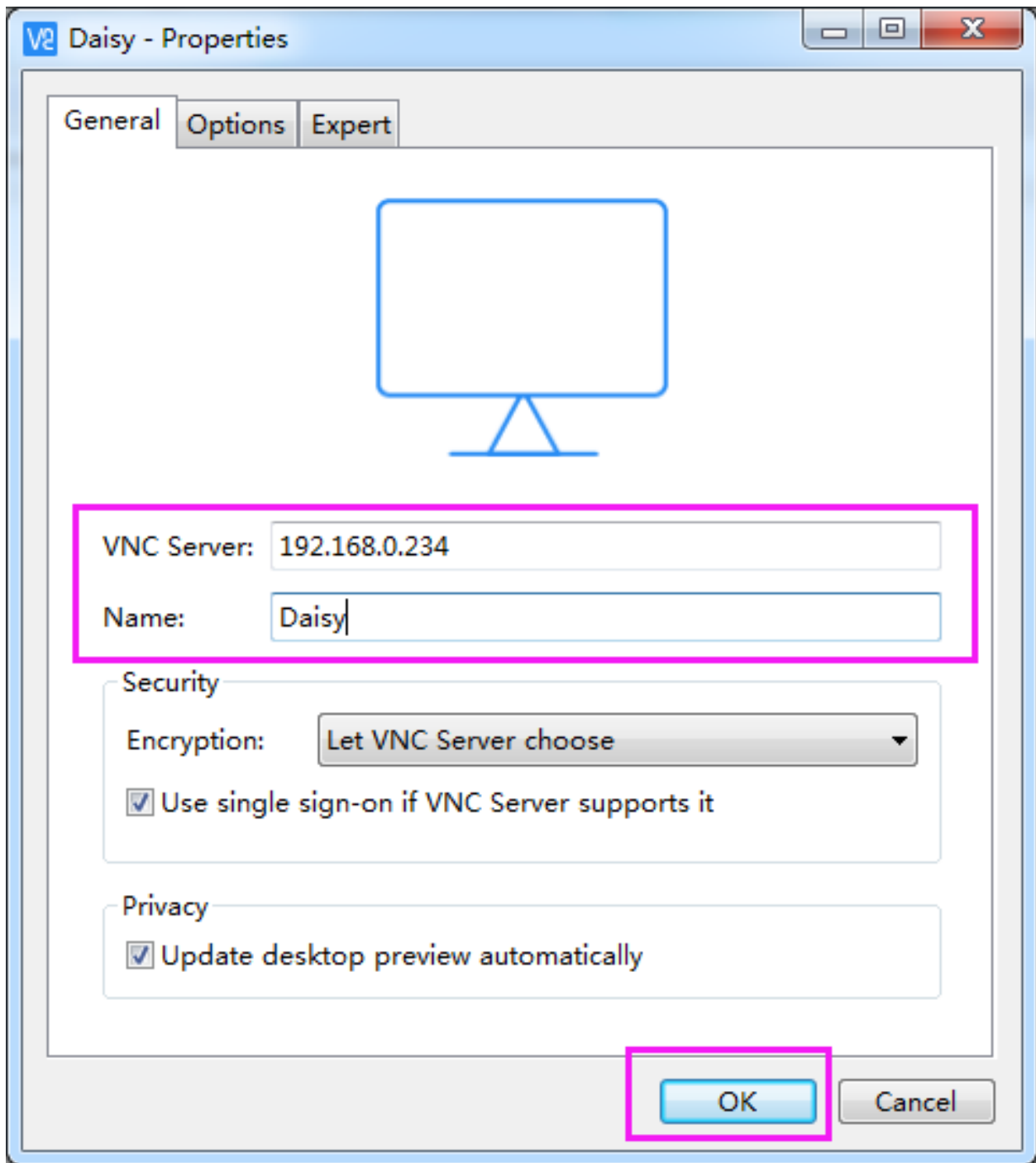
You need to download and install the [VNC Viewer](#) on personal computer. After the installation is done, open it.

Step 2

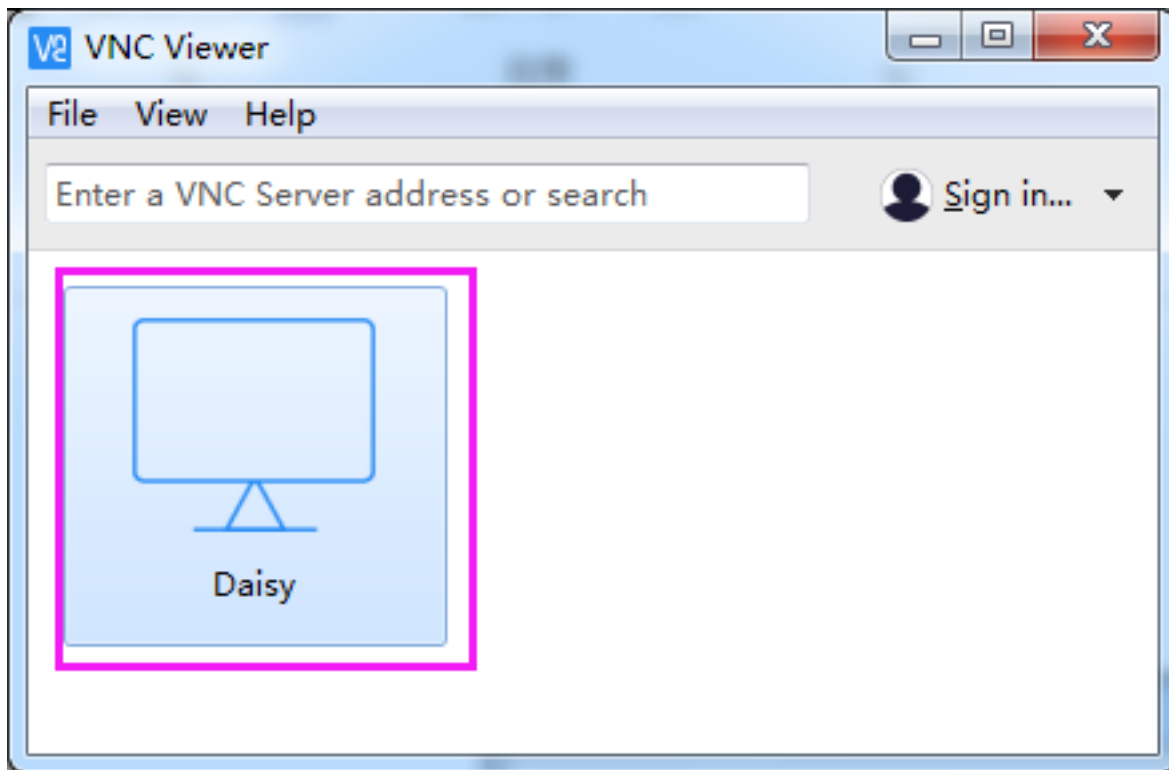
Then select “**New connection**”.

**Step 3**

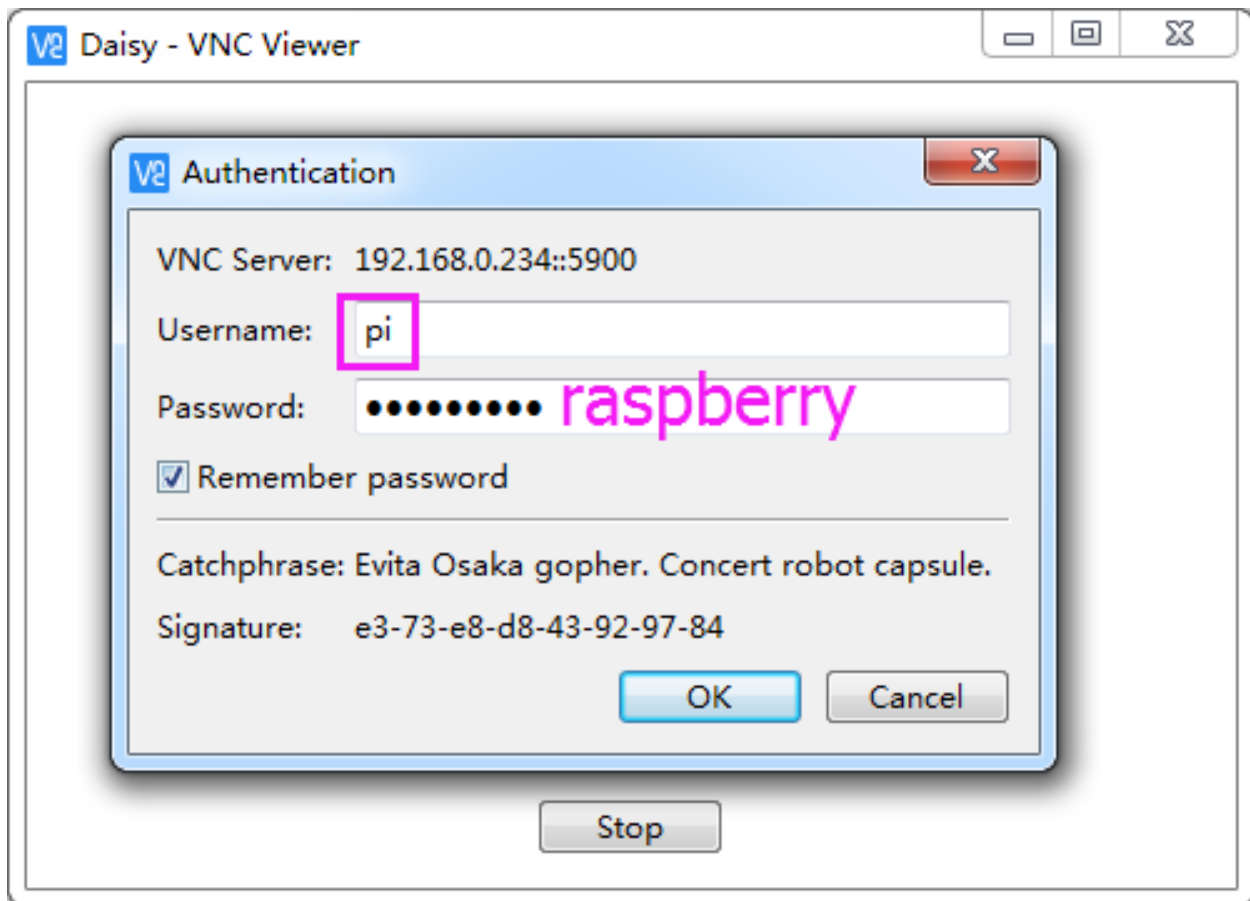
Input IP address of Raspberry Pi and any **Name**.

**Step 4**

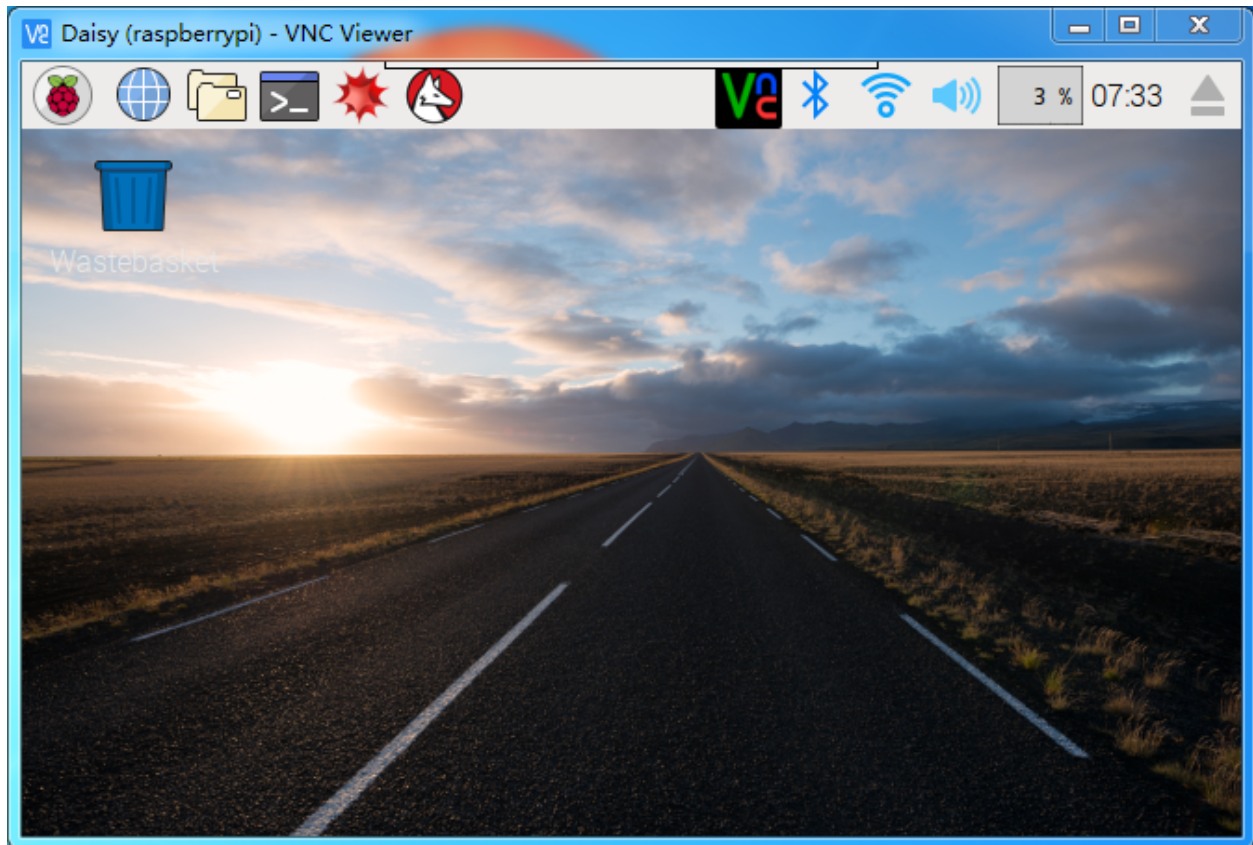
Double click the **connection** just created:

**Step 5**

Enter Username (**pi**) and Password (**raspberry** 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

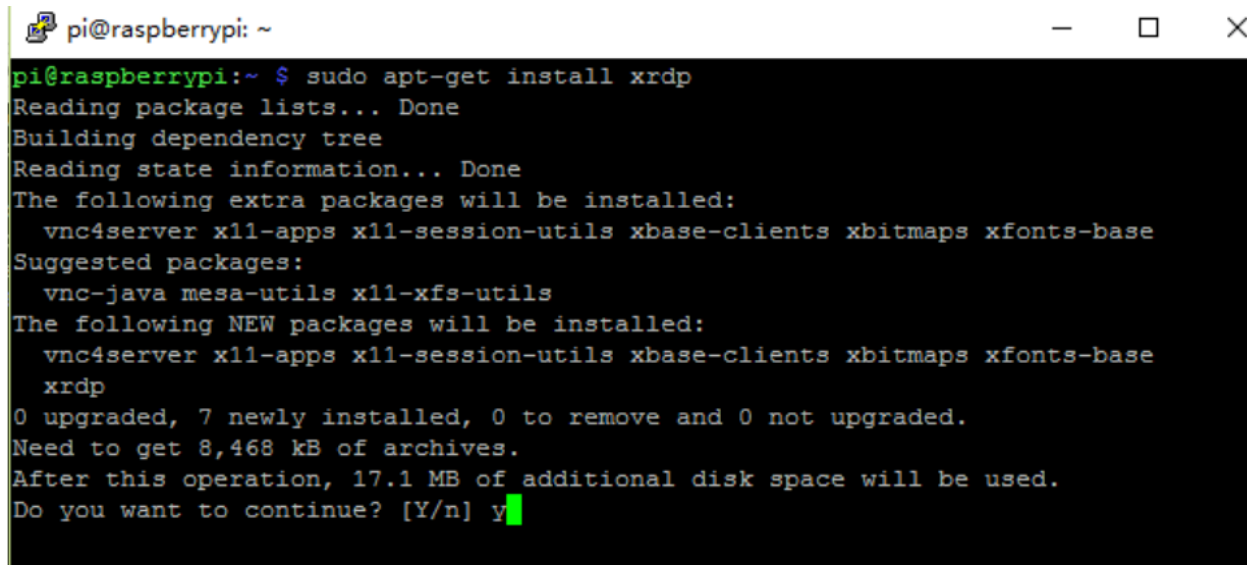
Input the following instructions to install XRDP.

```
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.

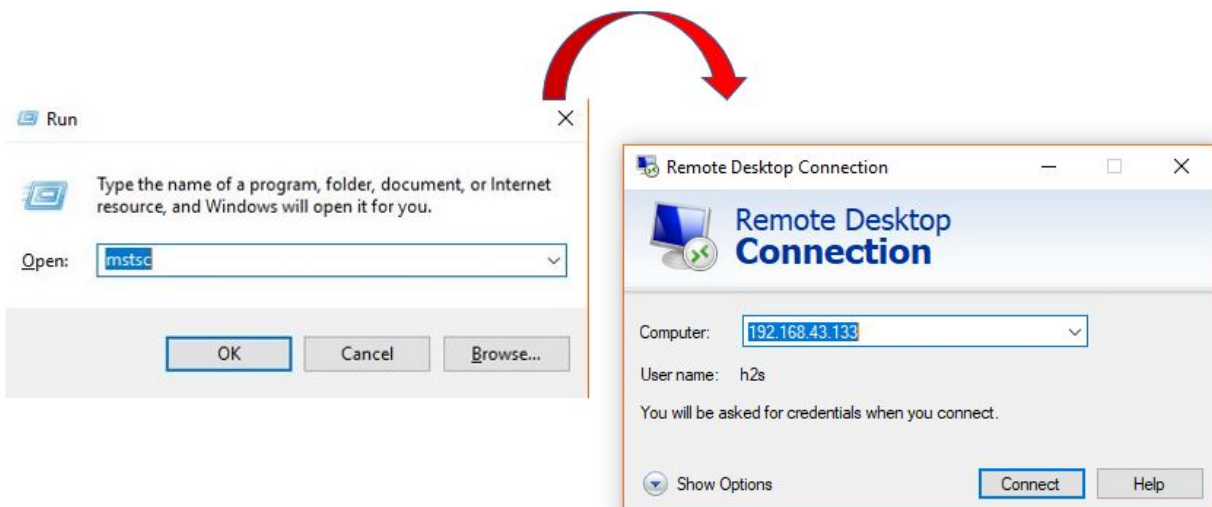
Login to XRDP

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.

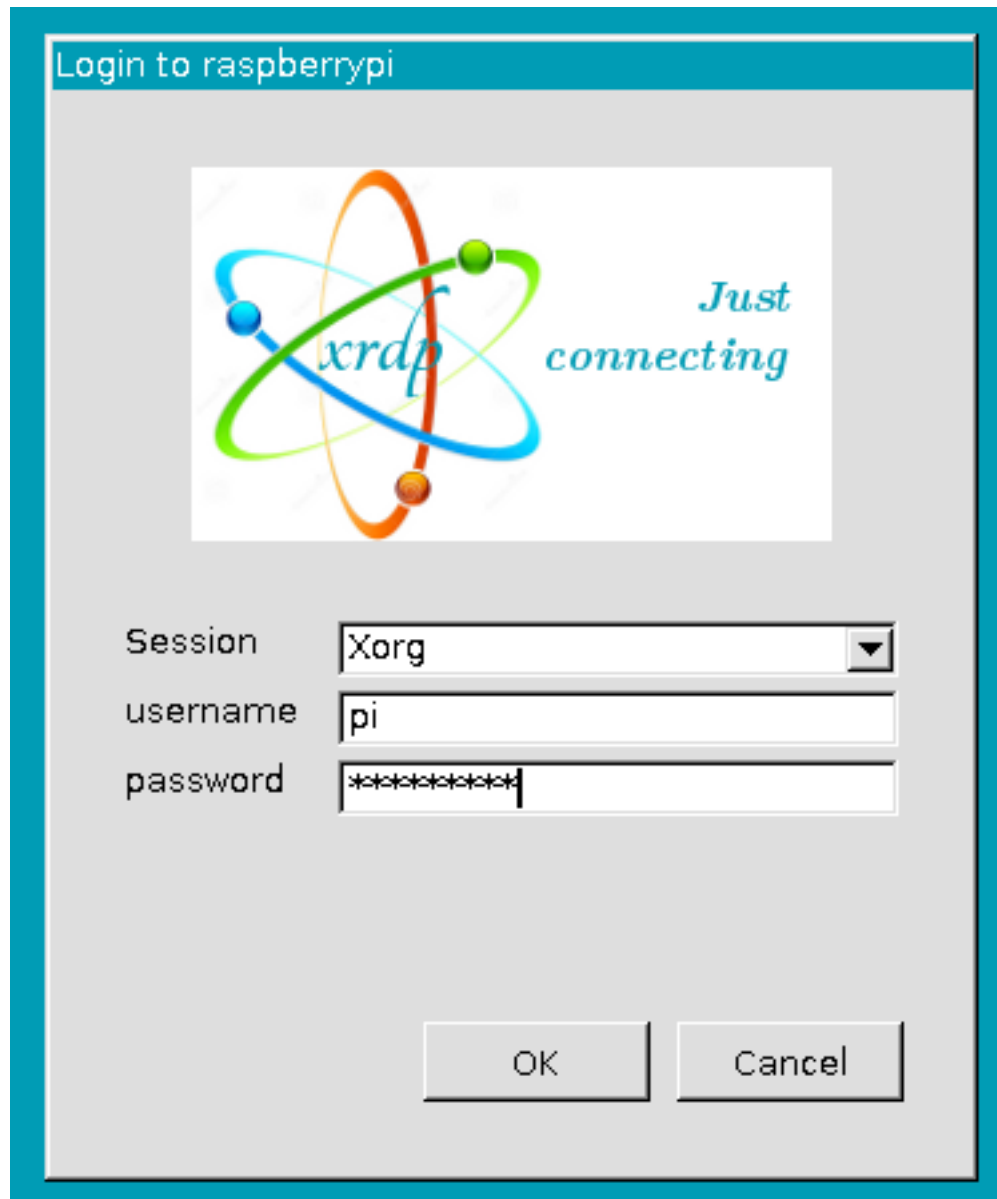
Step 2

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”.



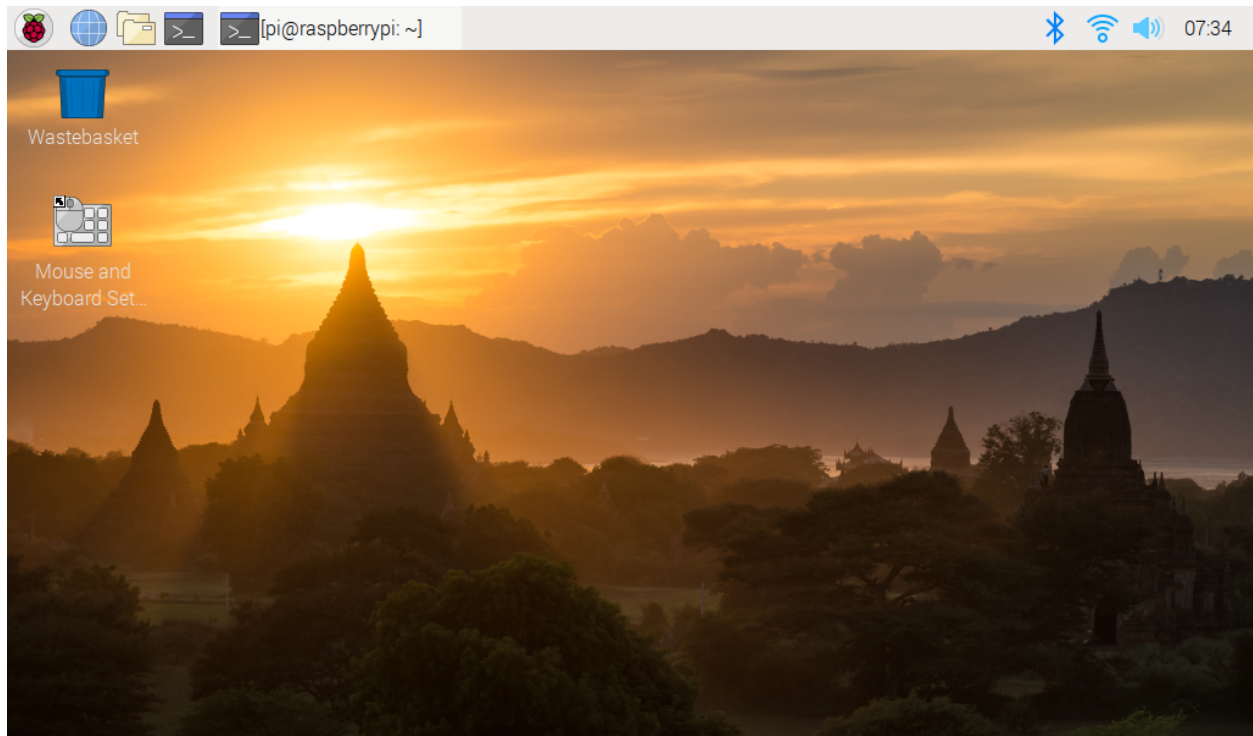
Step 3

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

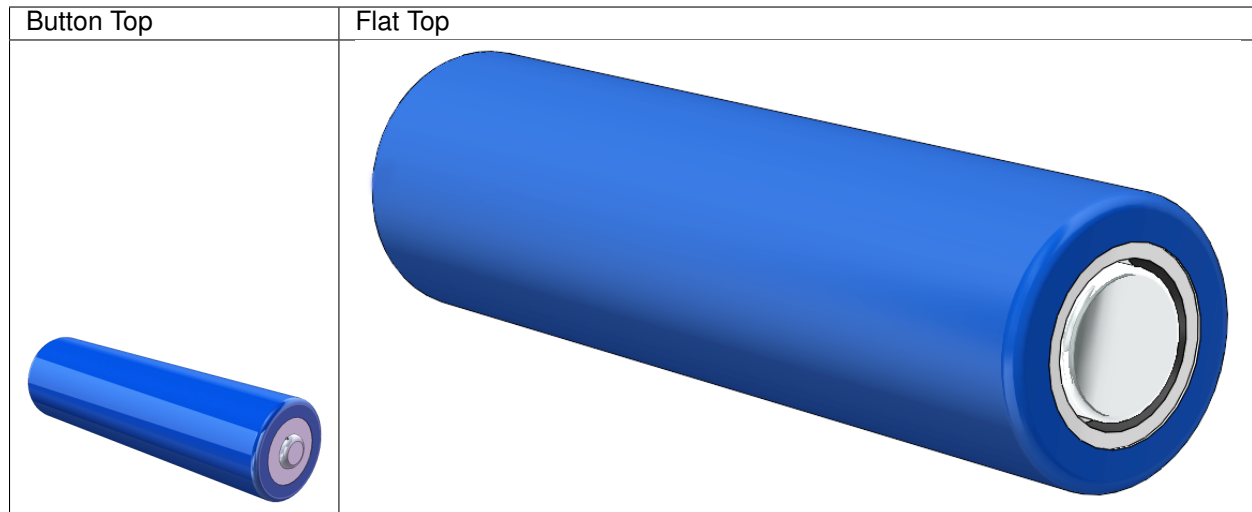
- 3.7V
- 18650
- Rechargeable
- Li-ion Battery
- Button Top
- No Protective Board

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.

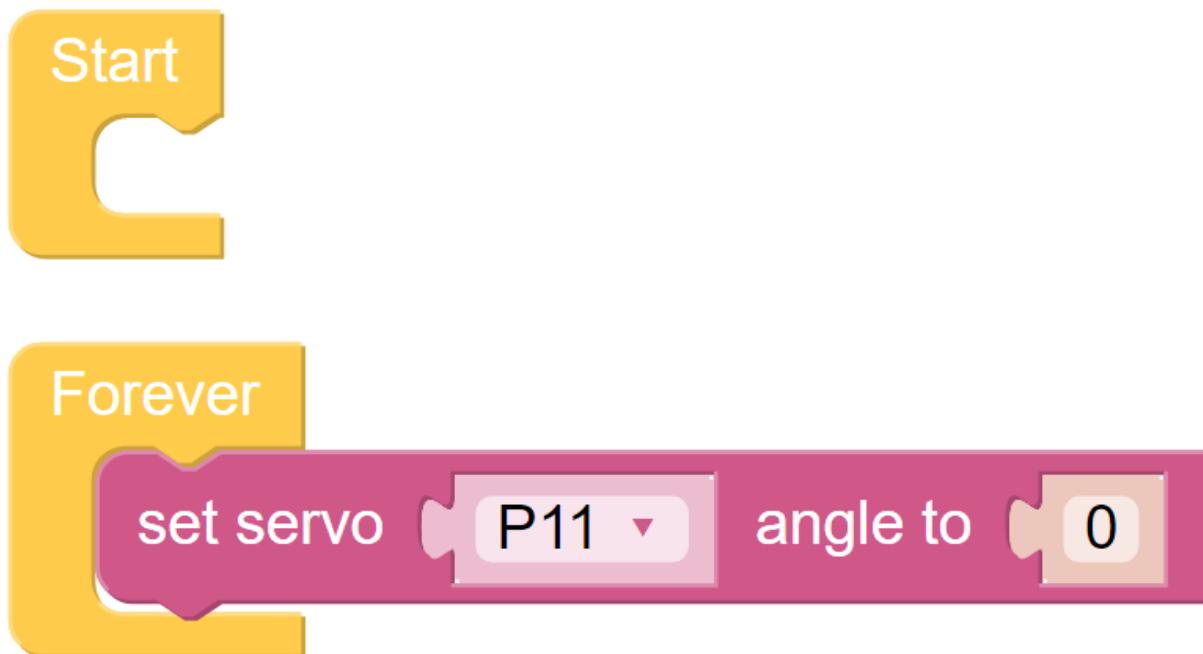
Battery capacity?

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.

FAQ

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

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



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

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

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

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

After a period of power failure, if no PWM signal is given to the 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.