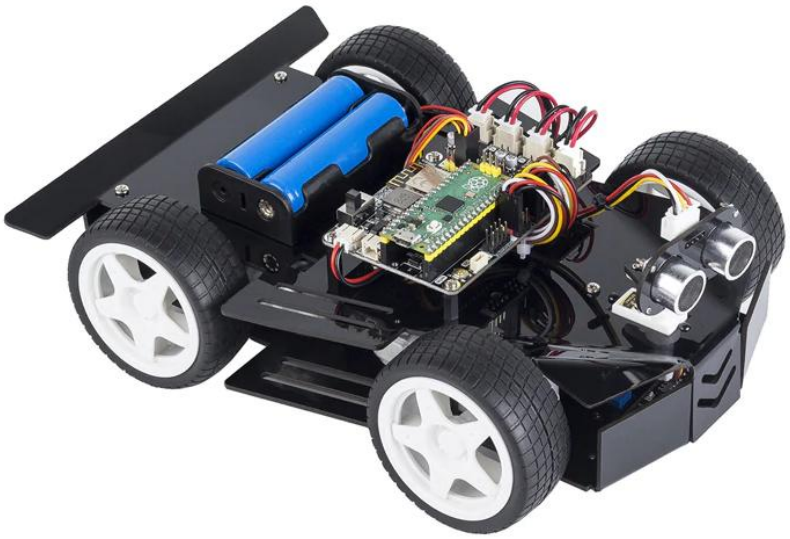
SunFounder pico_4wd_car

www.sunfounder.com

Feb 27, 2023

CONTENTS



The Pico-4wd is a Raspberry Pi Pico based, cool, robot car kit that everyone can have.

Equipped with greyscale sensor module and ultrasonic module, it can perform line tracking, cliff detection, follow and obstacle avoidance functions. The RGB boards assembled at the bottom and rear of the car make it the coolest spirit in the dark.

We have provided sample code based on MicroPython so you can get started quickly.

In addition, you can also use an app - SunFounder Controller - to DIY your own control methods! Let's Play!

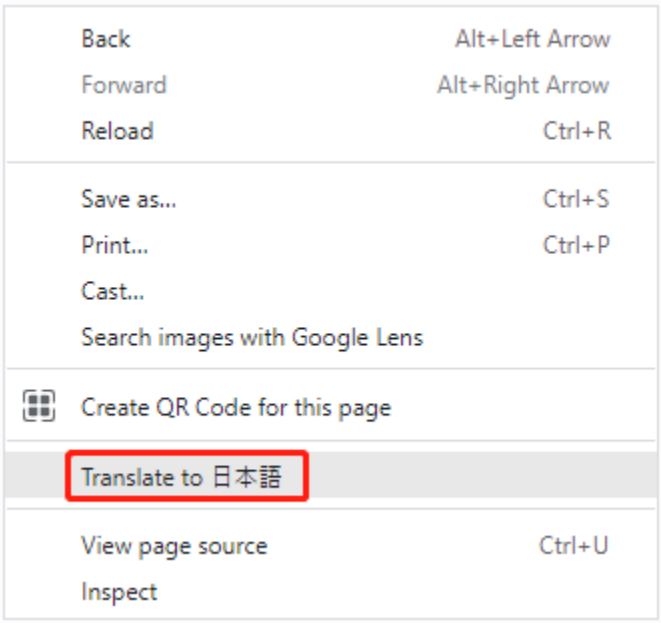
Here is the Email: cs@sunfounder.com.

About the display language

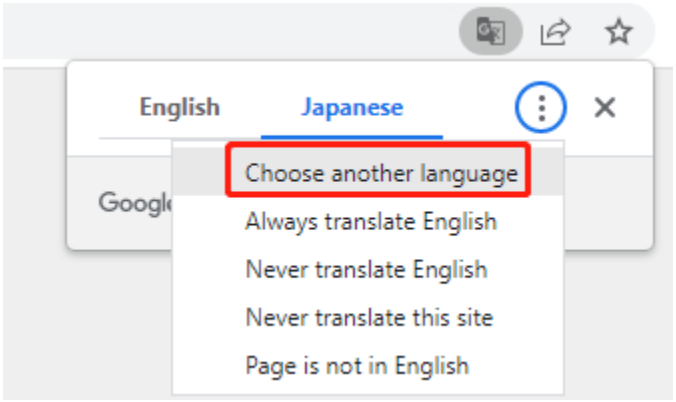
In addition to English, we are working on other languages for this course. Please contact service@sunfounder.com if you are interested in helping, and we will give you a free product in return. In the meantime, we recommend using Google Translate to convert English to the language you want to see.

The steps are as follows.

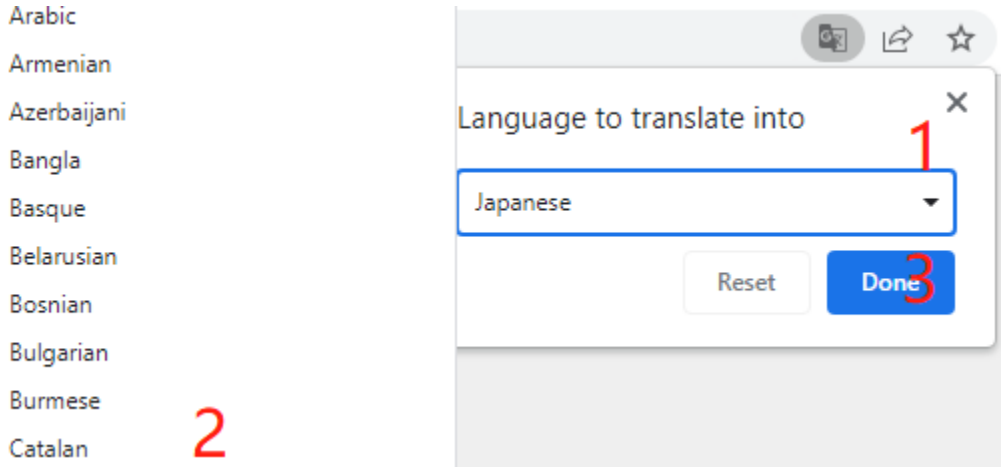
- In this course page, right-click and select **Translate to xx**. If the current language is not what you want, you can change it later.



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



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



Source Code

`SunFounder Pico-4wd Car Code`

Or check out the code at

CHAPTER

ONE

GET STARTED

In this section, you will learn all the hardware and software configurations needed to get Pico-4wd up and running, and it is recommended that you read them in order.

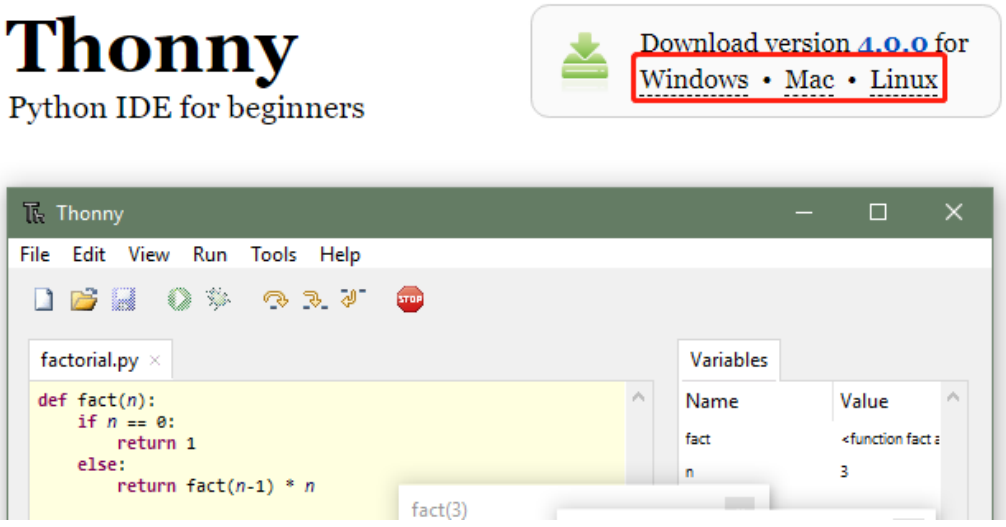
1.1 1. Install Thonny IDE

Before you can start to program Pico with MicroPython, you need an integrated development environment (IDE), here we recommend Thonny. Thonny comes with Python 3.7 built in, just one simple installer is needed and you're ready to learn programming.

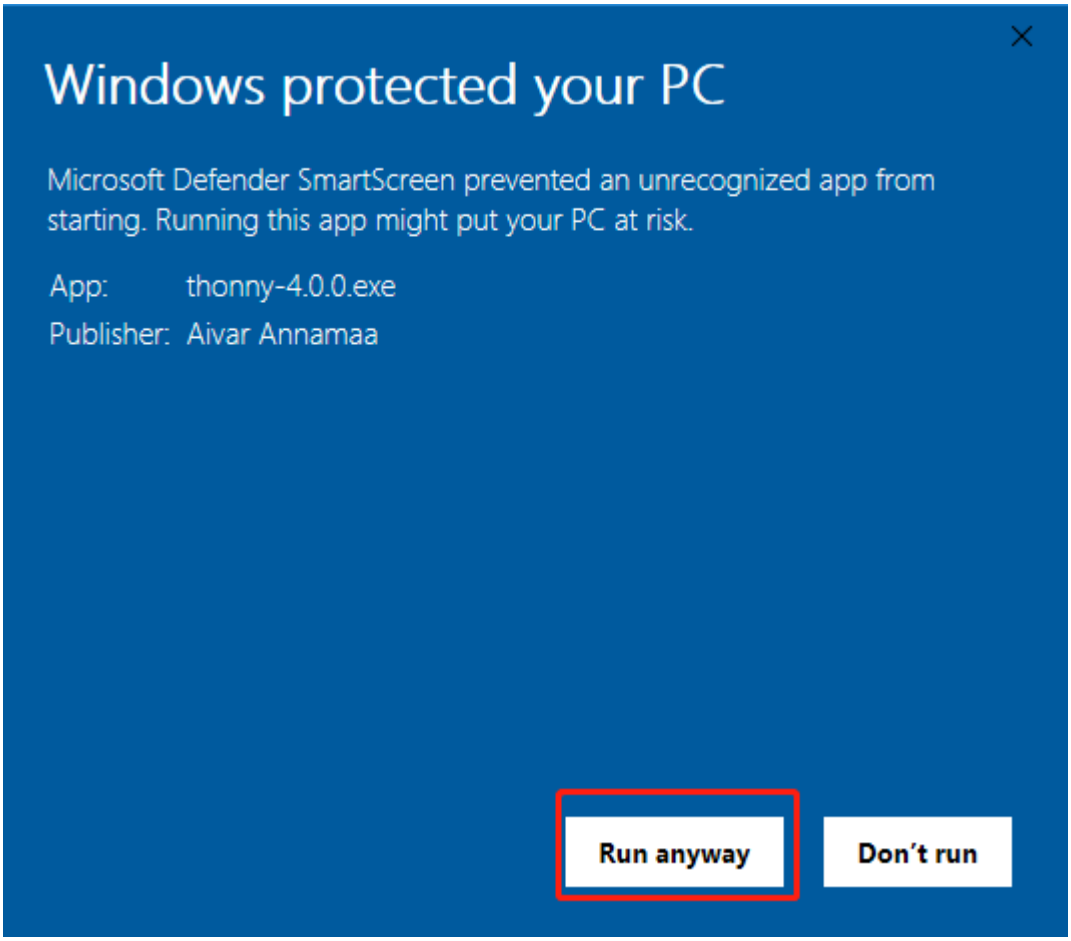
Note: Since the Raspberry Pi Pico interpreter only works with Thonny version 3.3.3 or later, you can skip this chapter if you have it; otherwise, please update or install it.

---

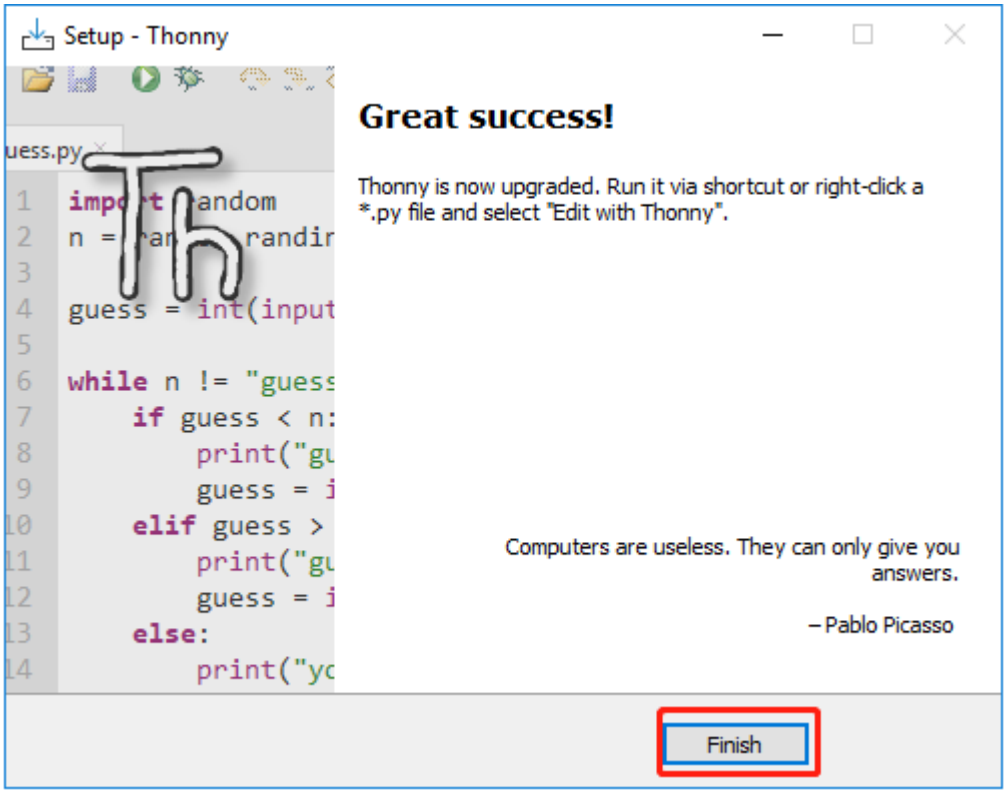
1. You can download it by visiting the website. Once open the page, you will see a light gray box in the upper right corner, click on the link that applies to your operating system.



2. The installers have been signed with a new certificate which hasn't built up its reputation yet. You may need to click through your browser warning (e.g. choose "Keep" instead of "Discard" in Chrome) and Windows Defender warning (**More info Run anyway**).



3. Next, click **Next** and **Install** to finish installing Thonny.



1.2 2. Install MicroPython on Your Pico

MicroPython is a software implementation of a programming language largely compatible with Python 3, written in C, that is optimized to run on a microcontroller.

MicroPython consists of a Python compiler to bytecode and a runtime interpreter of that bytecode. The user is presented with an interactive prompt (the REPL) to execute supported commands immediately. Included are a selection of core Python libraries; MicroPython includes modules which give the programmer access to low-level hardware.

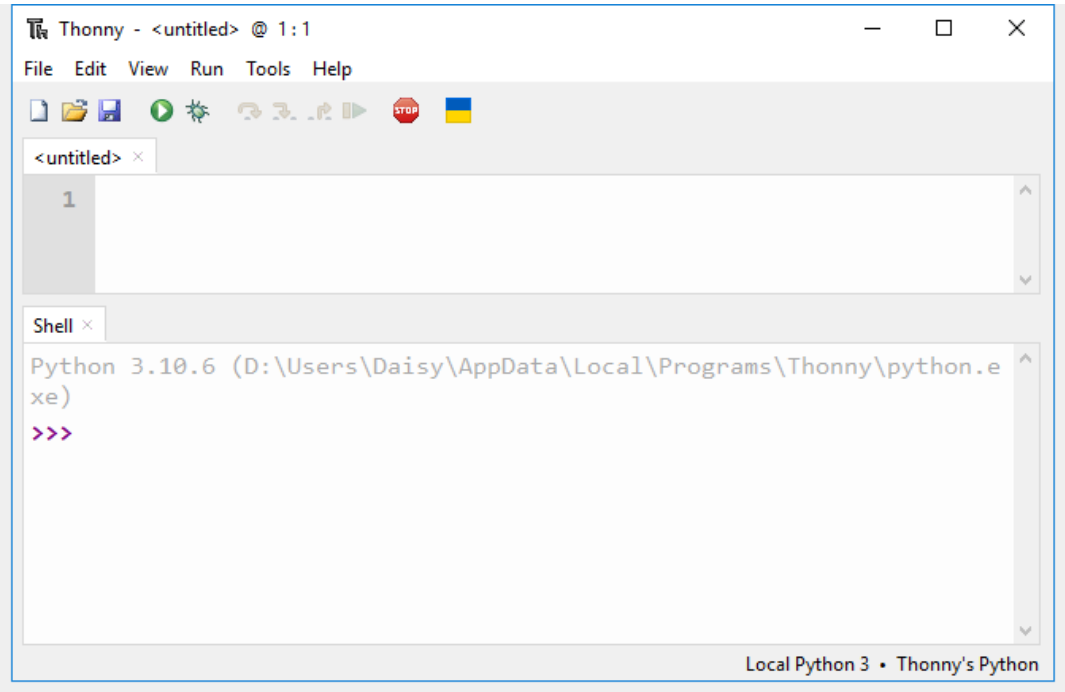
- Reference:

Now come to install MicroPython into Raspberry Pi Pico, Thonny IDE provides a very convenient way for you to install it with one click.

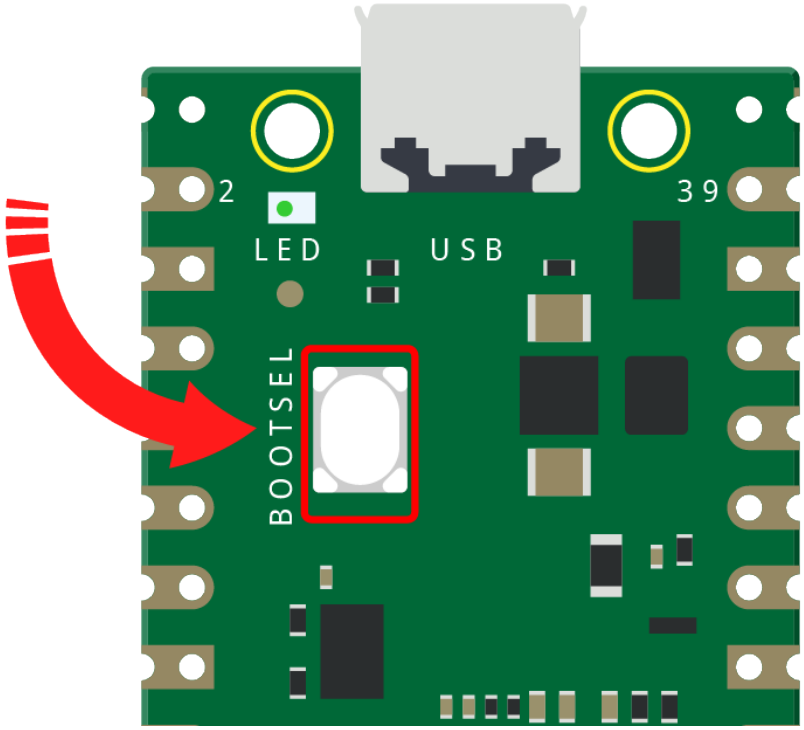
Note: If you do not wish to upgrade Thonny, you can use the Raspberry Pi official by dragging and dropping an `rp2_pico_xxxx.uf2` file into Raspberry Pi Pico.

---

1. Open Thonny IDE.

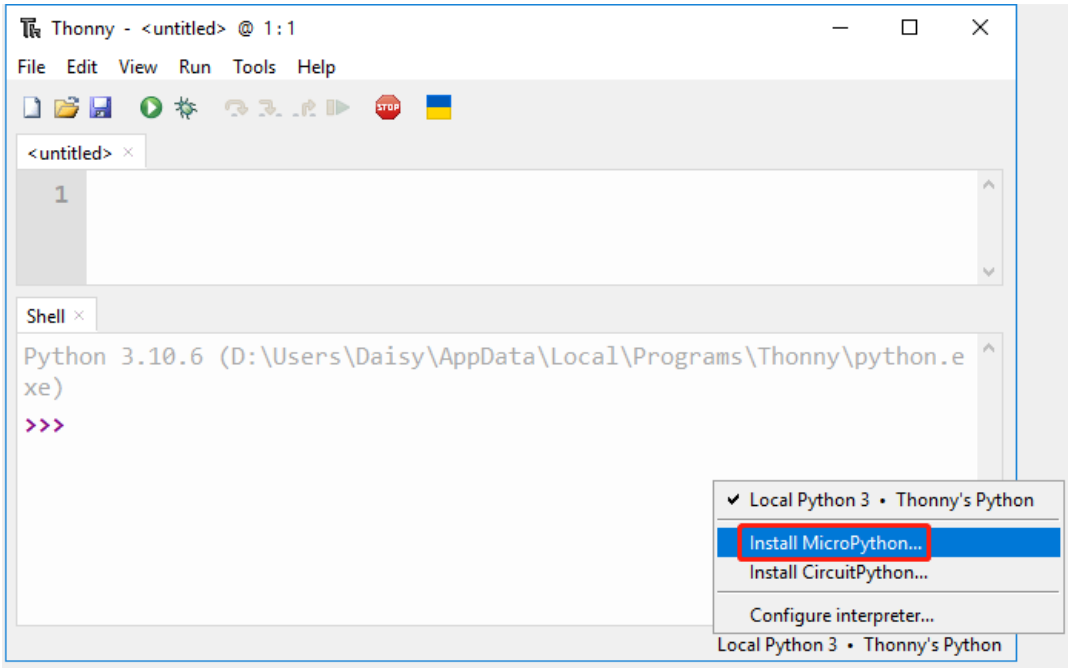


2. Press and hold the **BOOTSEL** button and then connect the Pico to computer via a Micro USB cable. Release the **BOOTSEL** button after your Pico is mount as a Mass Storage Device called **RPI-RP2**.

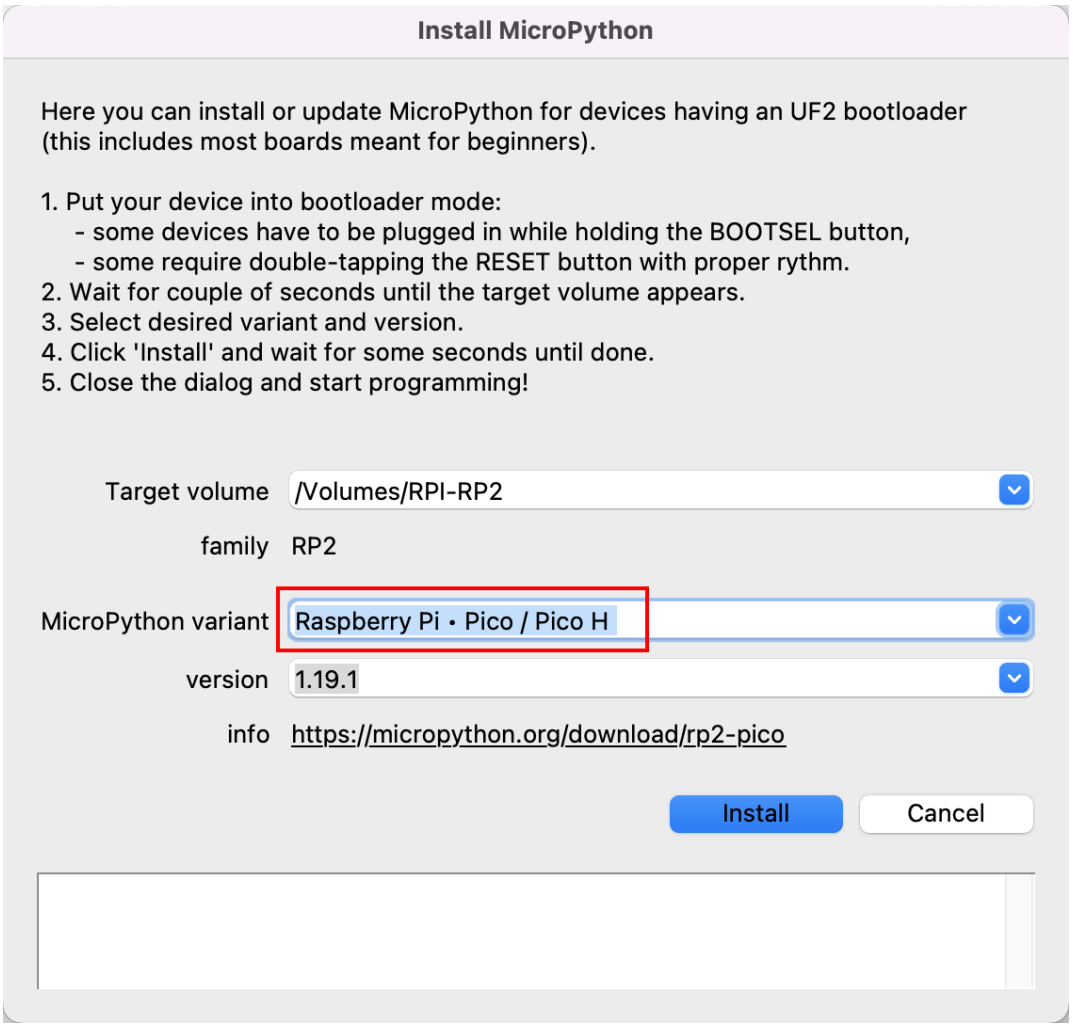


3. In the bottom right corner, click the interpreter selection button and select **Install Micropython**.

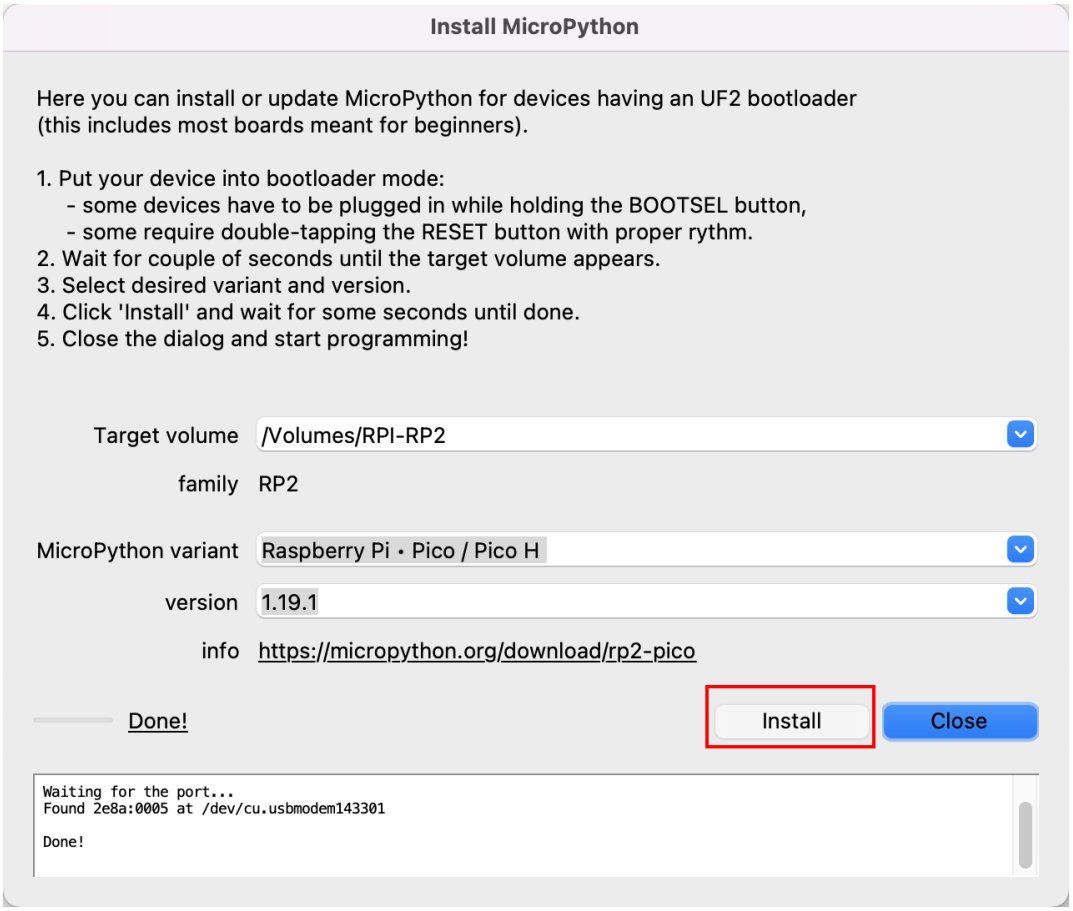
> **Note:** If your Thonny does not have this option, please update to the latest version.



4. In the **Target volume**, the volume of the Pico you just plugged in will automatically appear, and in the **Micropython variant**, select **Raspberry Pi.Pico/Pico H**.



5. Click the **Install** button, wait for the installation to complete and then close this page.

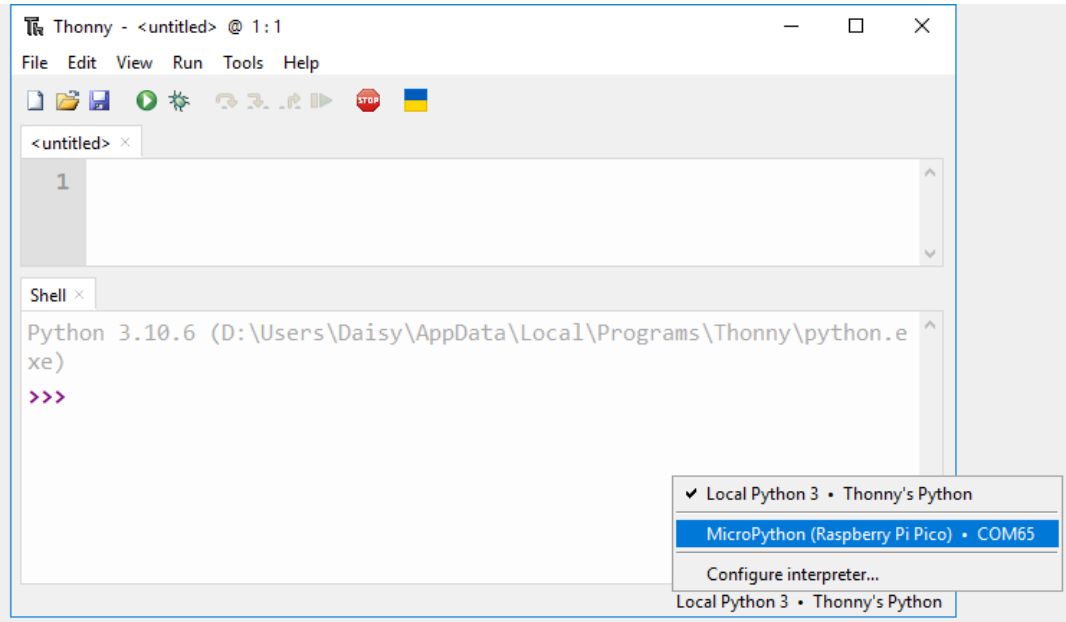


Congratulations, now your Raspberry Pi Pico is ready to go.

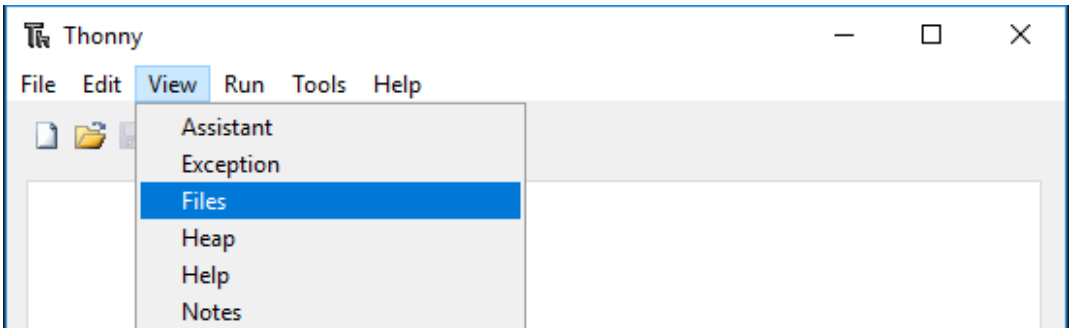
1.3 3. Upload the Libraries to Pico

Before using Pico-4wd Car, you need to upload its related libraries in Raspberry Pi Pico.

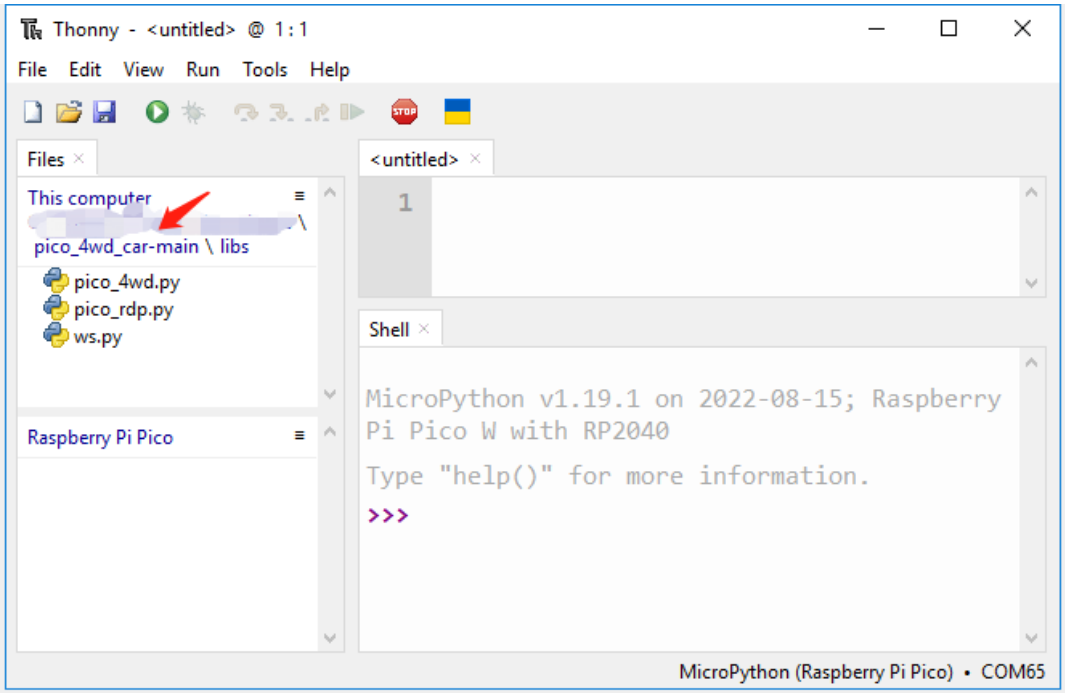
1. Download the relevant code from the link below.
    - `SunFounder Pico-4wd Car Code`
2. Open Thonny IDE and plug the Pico into your computer with a micro USB cable and click on the "MicroPython (Raspberry Pi Pico).COMXX" interpreter in the bottom right corner.



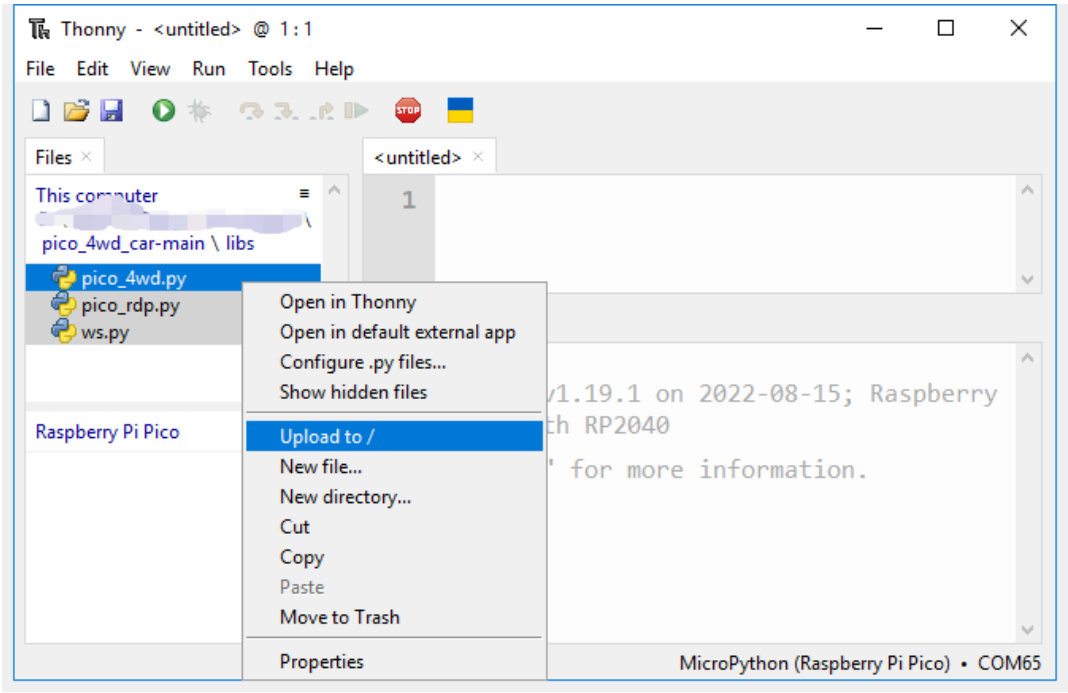
3. In the top navigation bar, click **View** -> **Files**.



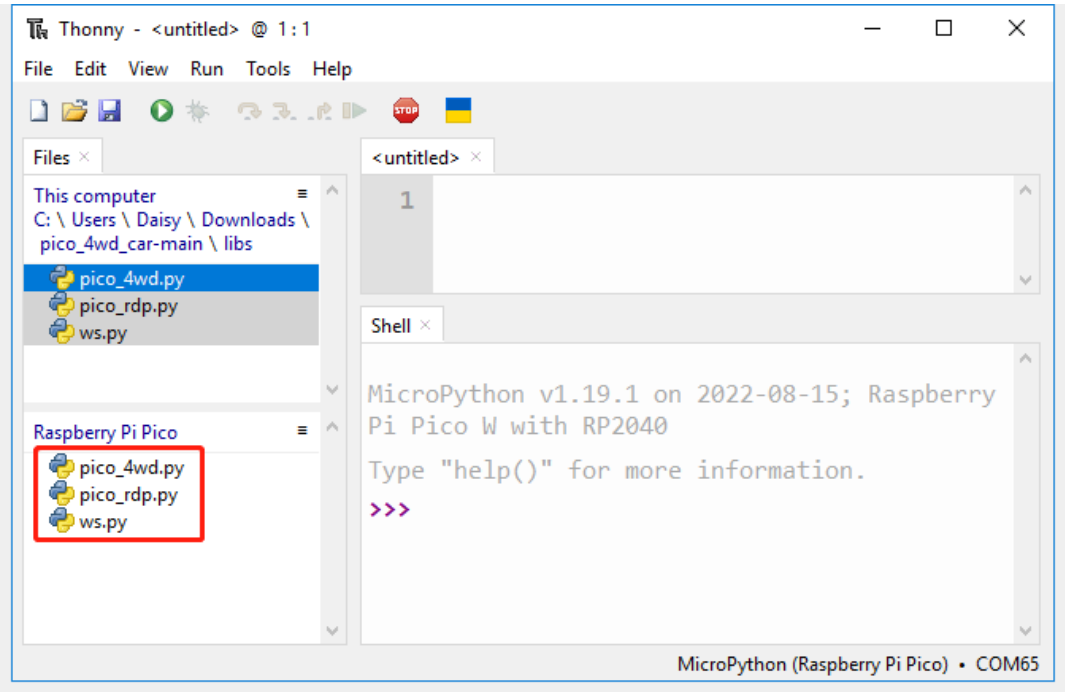
4. Switch the path to the folder where you downloaded the code package before, and then go to the `pico_4wd_car_main/libs` folder.



5. Select these 3 files, right-click and click **Upload to**, it will take a while to upload.



6. Now you will see the files you just uploaded inside your drive `Raspberry Pi Pico`.

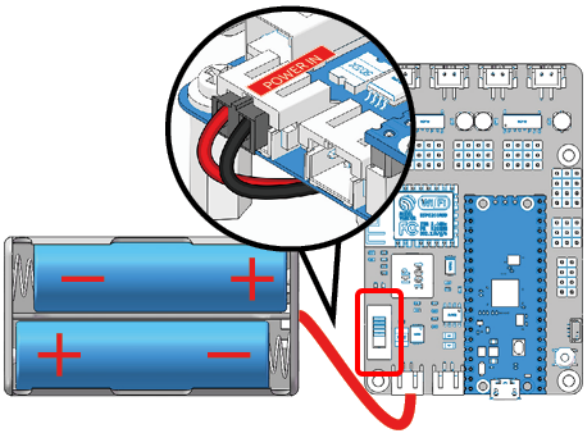


1.4 4. Test the Modules

This chapter is suitable for usability testing of the modules before assembly; or for final commissioning and maintenance of the Pico-4wd after assembly has been completed.

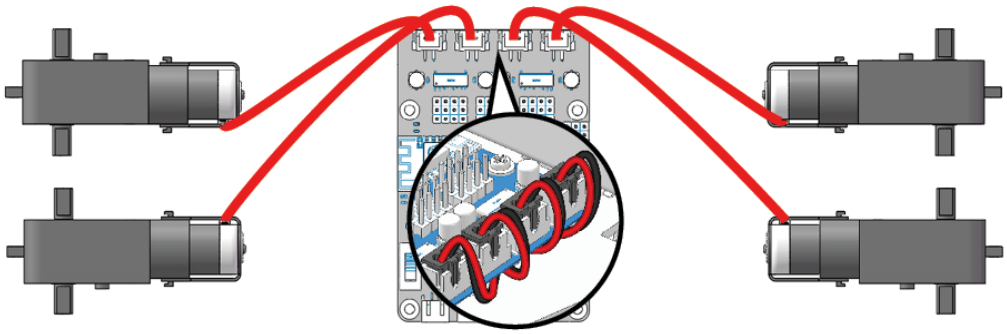
1.4.1 Power up the Pico RDP

In order to make the module work, you need to power up the Pico RDP and turn the power switch to ON.



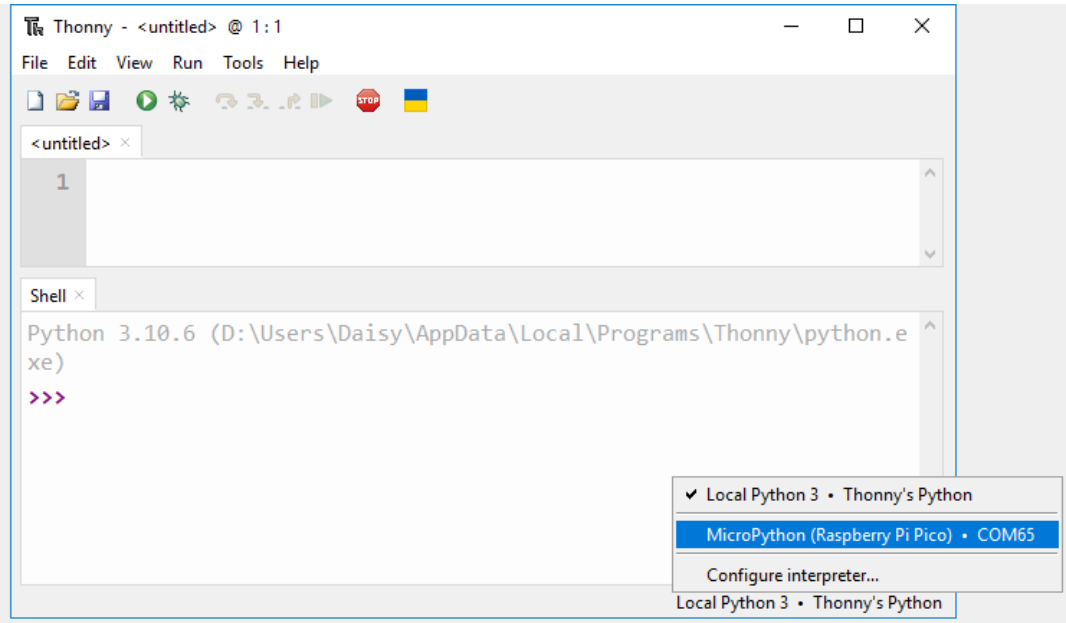
1.4.2 Test the Motors

1. Connect the 4 motors according to the diagram below.

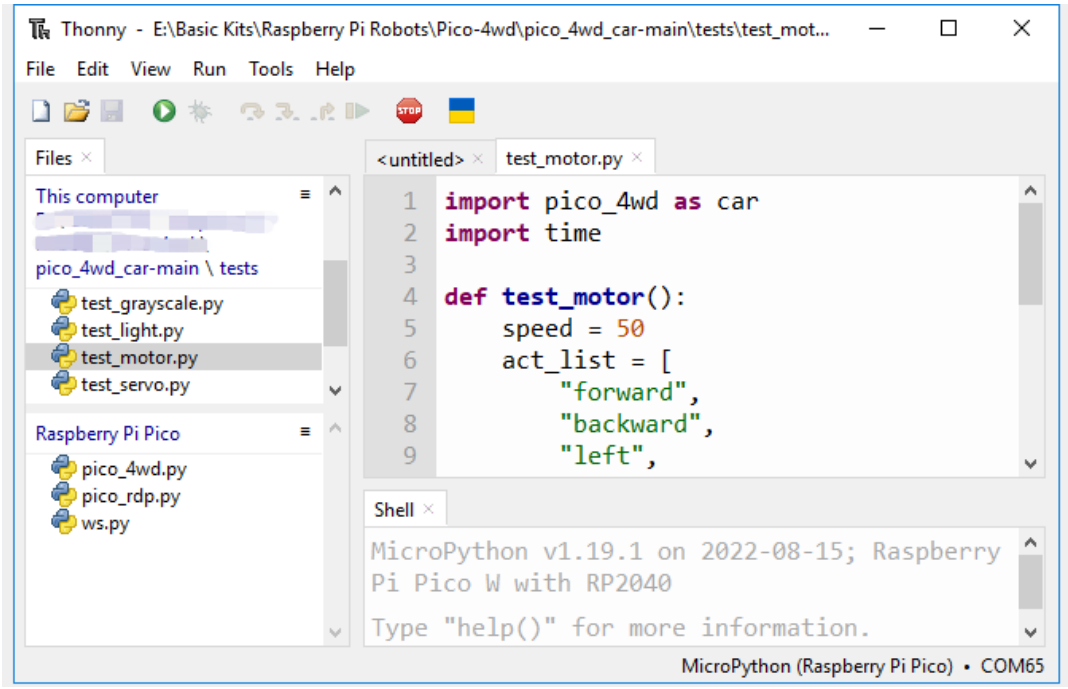


2. Select Correct Interpreter

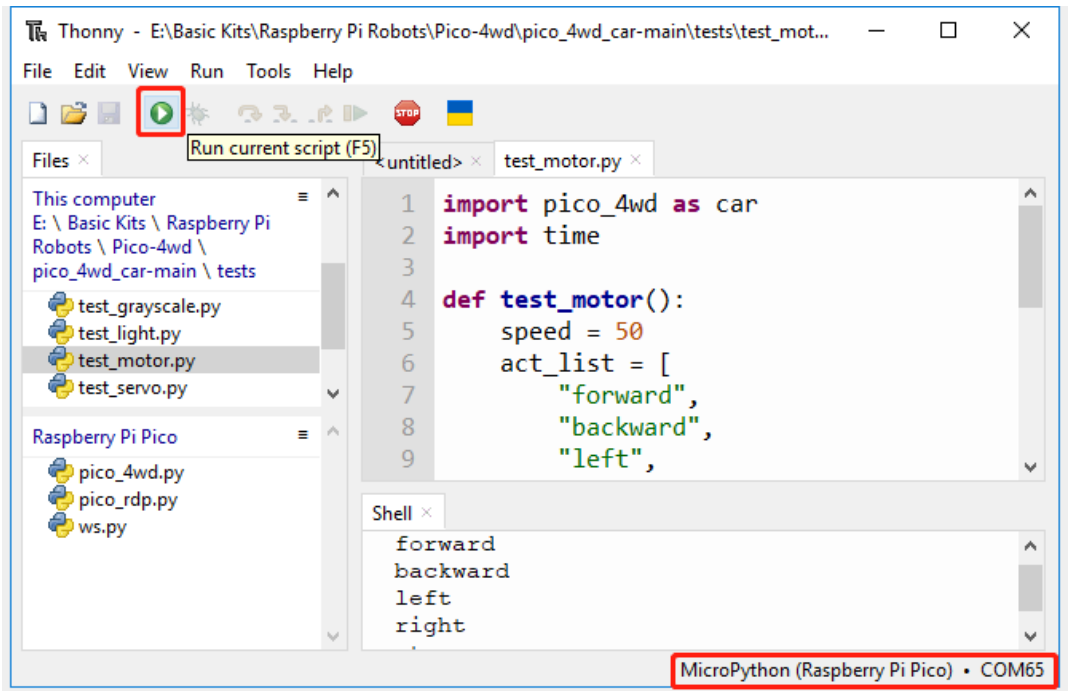
   Plug the Pico into your computer with a micro USB cable and select the "MicroPython (Raspberry Pi Pico).COMXX" interpreter in the bottom right corner.



3. Go to the `pico_4wd_car_main/tests` path and double click on `test_motor.py` to open it.



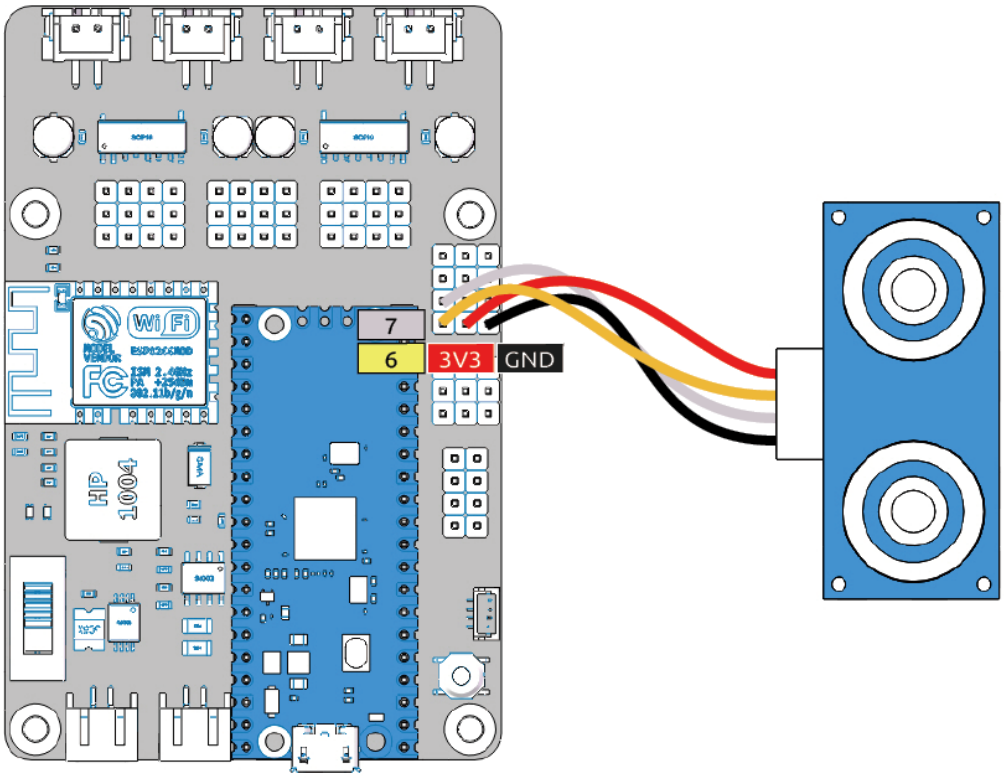
4. Click the **Run current script** button or just press F5 to run it.



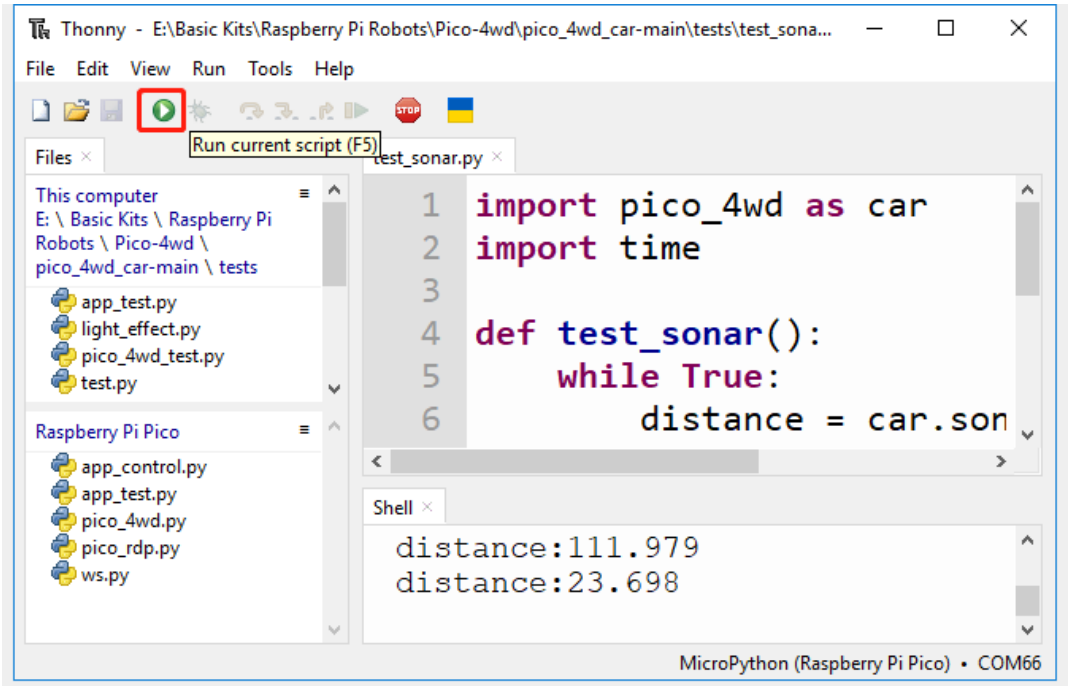
Now, you can see the four motors work in a regular pattern. If you complete the assembly, this code will make the Pico-4wd perform five movements: forward, backward, left, right and stop.

1.4.3 Test the Ultrasonic Module

1. Connect the ultrasonic module as shown below.



2. Run the `test_sonar.py` file under the path `pico_4wd_car_main/tests`.

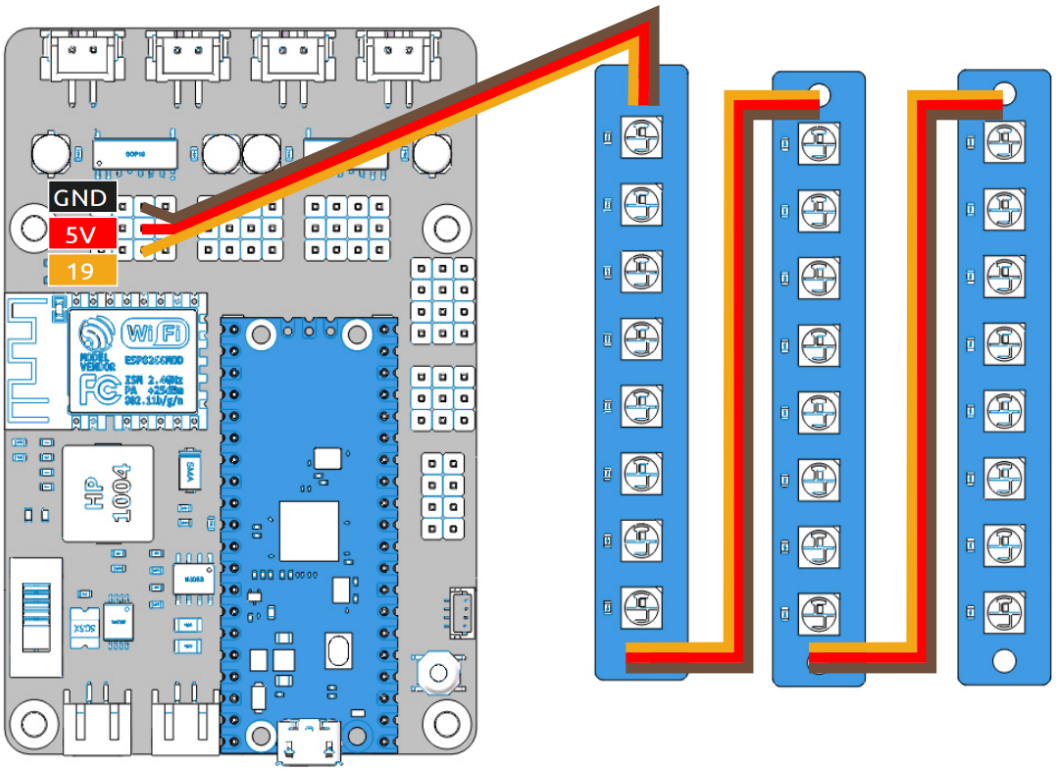


After running the code, the distance of ultrasonic detection will be displayed in the Shell. If the distance changes when

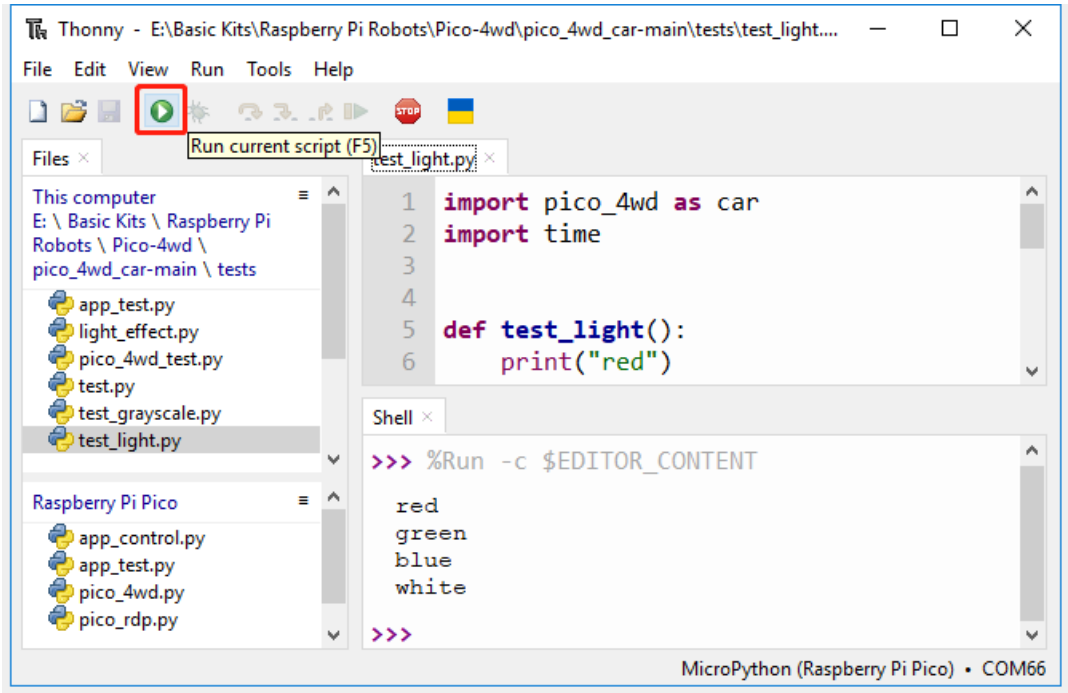
your hand is in front of the ultrasonic module, the module is working properly.

1.4.4 Test the RGB Boards

1. As shown below, connect the 3 RGB boards.



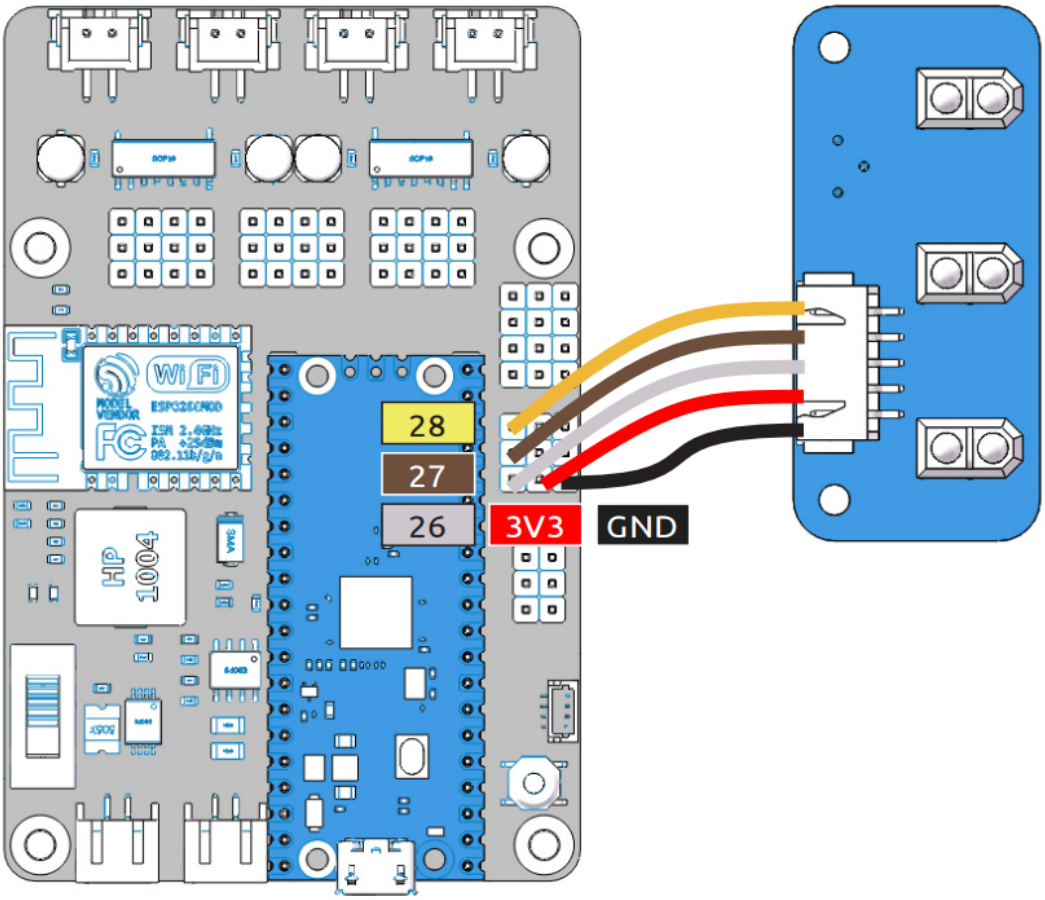
2. Run the `test_light.py` file in `pico_4wd_car_main/tests`.



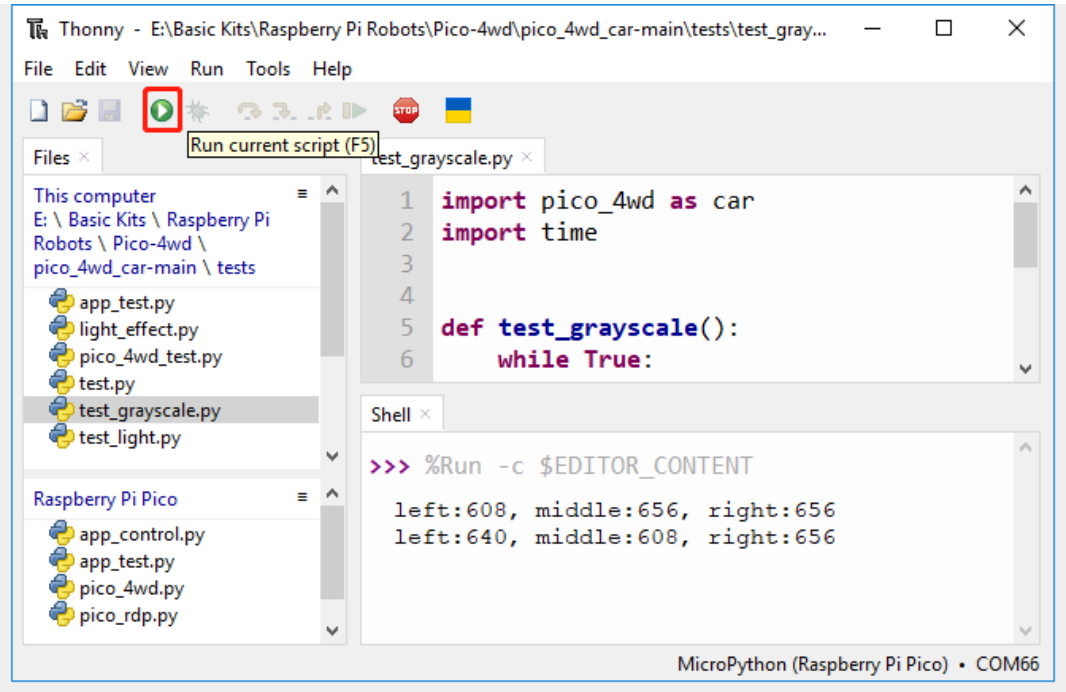
After the code is run, the 24 LEDs (all on the 3 RGB boards) to emit red, green, blue and white light in turn.

1.4.5 Test the Grayscale Sensor Module

1. Diagrammatically connect the Grayscale Sensor Module.



2. Run the `test_grayscale.py` file in `pico_4wd_car_main/tests`.

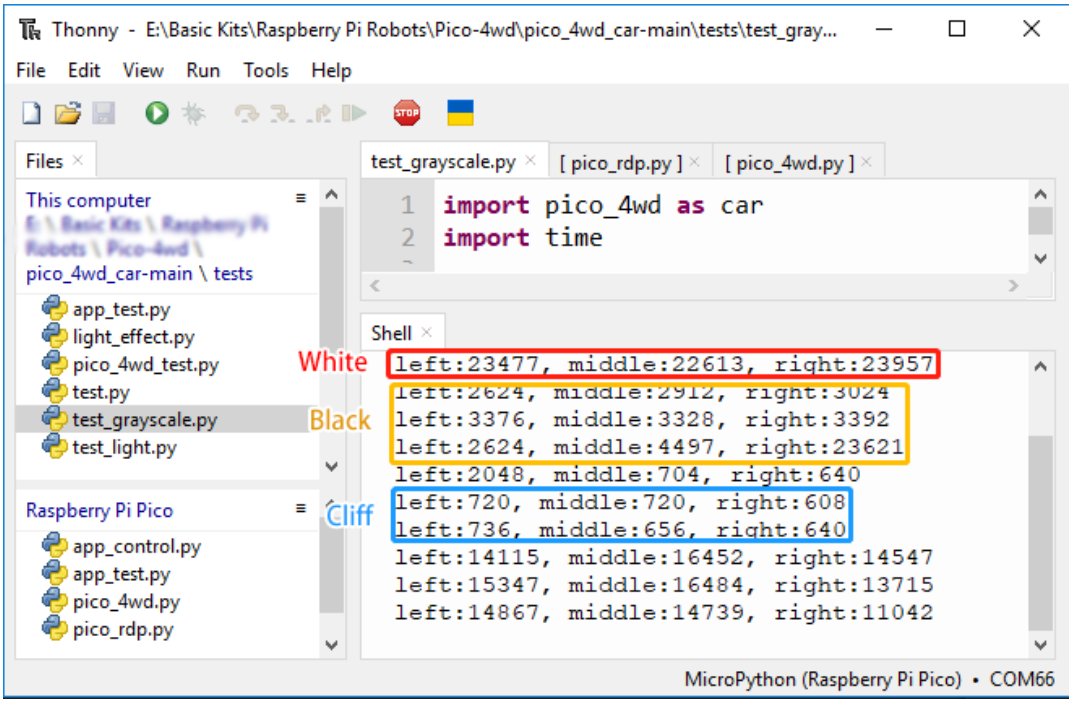


Upon running, you will see that the values in the Shell change when you place the grayscale module at a height of about 1cm above the different surfaces, indicating that it is working properly.

In general, the value for **white surfaces** > the value for **black surfaces** > the value for **overhanging** (grayscale module 10cm above the ground).

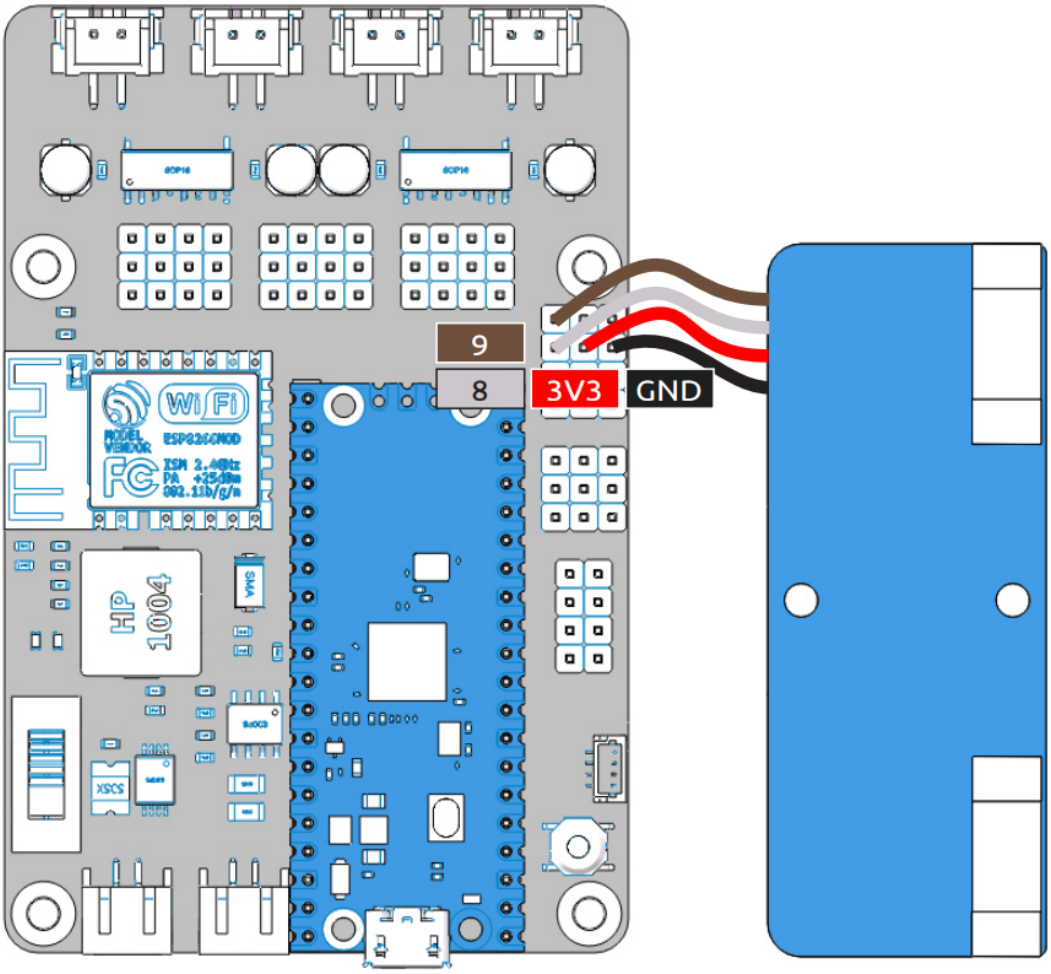
3. Because light intensities differ in different environments, the factory-set contrast may not be suitable for your current environment, which means the grayscale module cannot identify white and black lines well, so it needs to be calibrated.

    - Tape a small piece of black electrical tape to the ground or table.
    - Hold the grayscale module 1 cm above the table (this is about the same height as after assembly, so you can use it directly after calibration).
    - Keep `test_grayscale.py` running, then use a screwdriver to adjust the potentiometer on the grayscale module until the values printed on the Shell are relatively far apart. Based on the actual situation, your values should differ from mine.



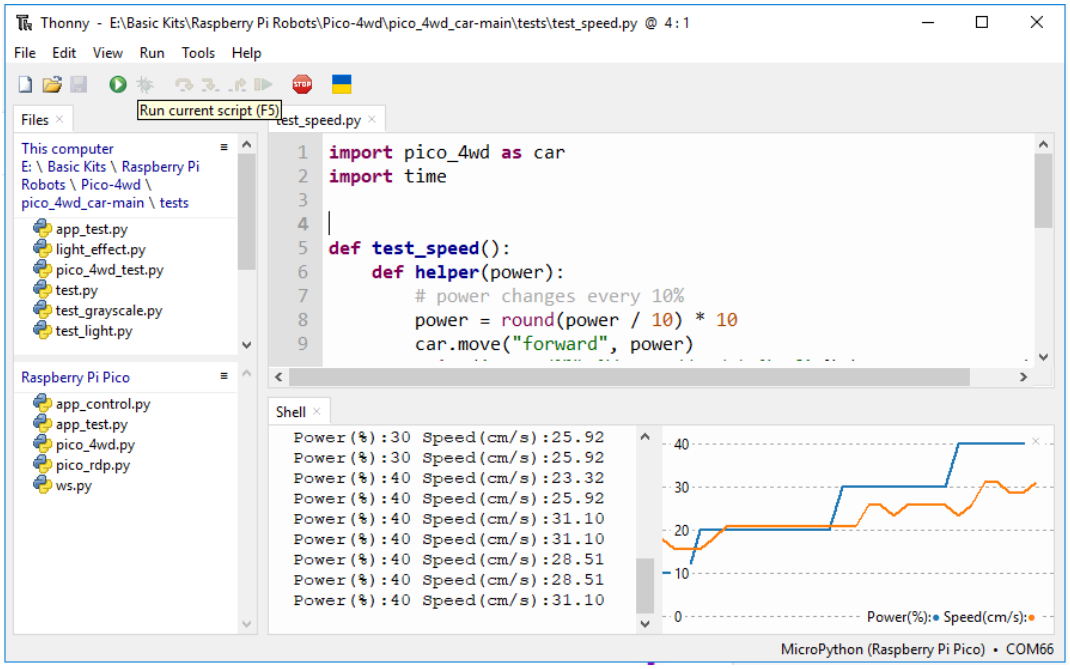
1.4.6 Test the Speed Module

1. Follow the diagram below to connect the Speed Module.



2. Run the `test_speed.py` file in `pico_4wd_car_main/tests`.

Note: The Thonny IDE contains a line graph tool, please open it by clicking **View > Plotter** in the navigation bar to help you see how the printed values are changing.

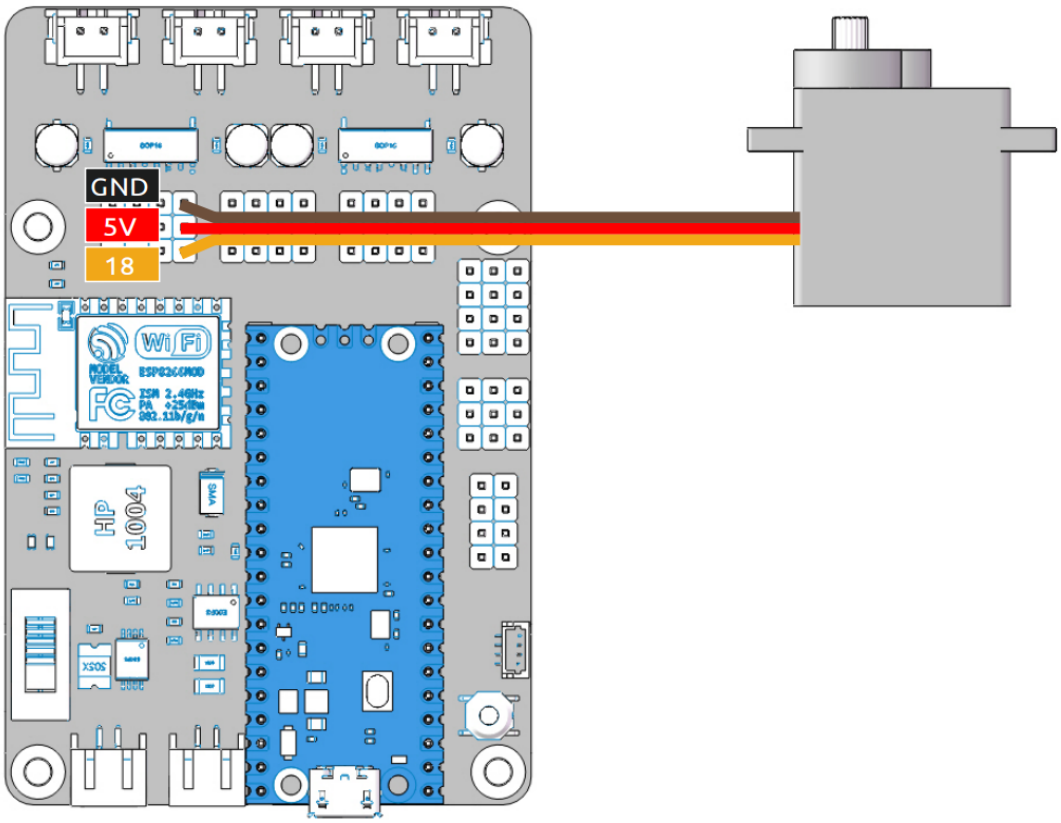


After the code runs, when you back and forth put the jammed paper into the U-shaped slot on the speed module/take it out. The Shell in Thonny IDE will print the current speed.

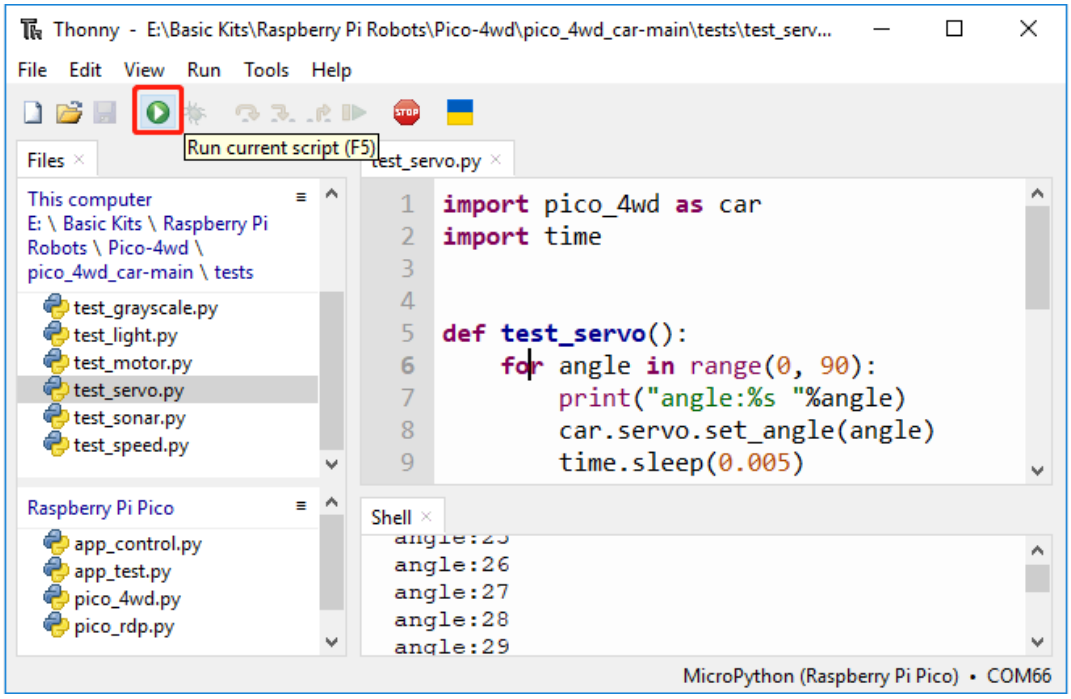
If you have already mounted it, this code will make the Pico-4wd move forward at variable speed and print out the motor power (as a percentage) and the travel speed (cm/s). To use it you should hover the car so that the motor rotation is not obstructed.

1.4.7 Test the Servo

1. Connect the servo according to the following diagram.



2. Run the `test_servo.py` file in `pico_4wd_car_main/tests`.



Insert a rocker arm first to observe the rotation of the servo. After clicking on the Run button, the servo will rotate left and right once and then stop at 0°.

Note: For the next chapter, *5. Assemble the Car*, the servo must be held at 0°, so after this code run, do not turn the servo shaft until the car is complete.

In the event that you accidentally turn the servo shaft, remove the rocker arm, run this code again, and then reassemble the servo.

1.5 5. Assemble the Car

Note: When you assemble the ultrasonic module, you need to keep the servo at 0°, which is done in *Test the Servo*.

If you accidentally turn the servo shaft, please take down the rocker arm, run `test_servo.py` code again, and then continue to assemble.

- `(PDF)Pico-4wd Assemble Instruction`

1.6 6. Examples

Here are some interesting and useful Pico-4wd projects you can follow the tutorials to accomplish.

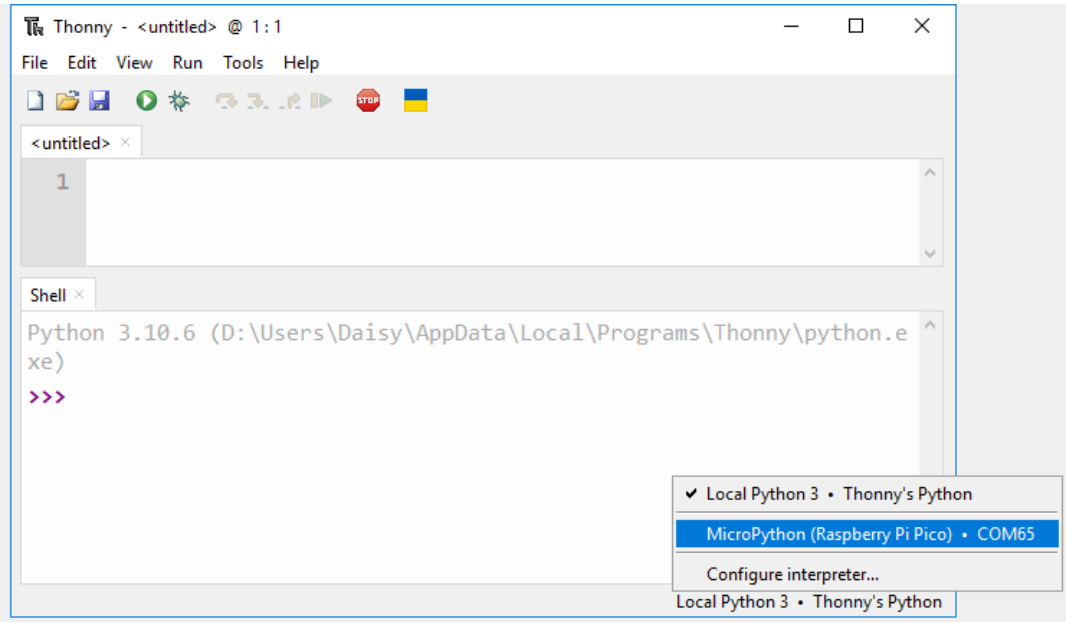
The flowcharts in each examples and the documentation can help you better understand the internal programming principles.

1.6.1 Motor Calibration

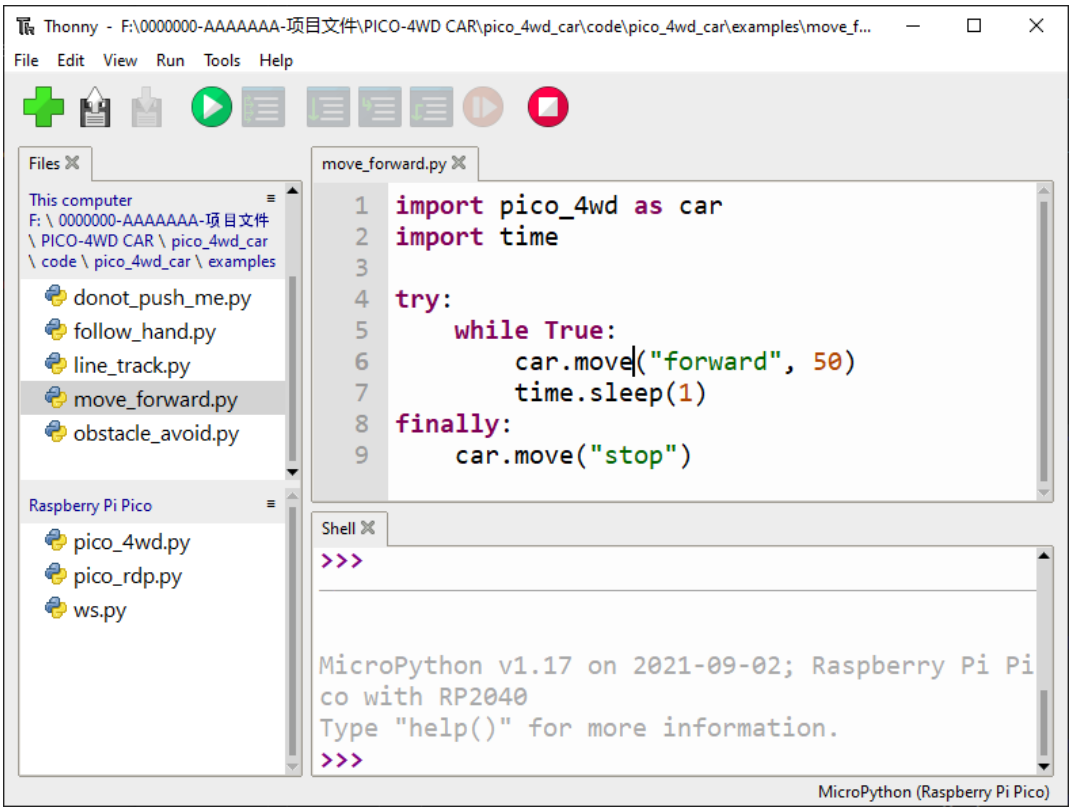
Because of assembly methods and other reasons, your motor may still need a calibration direction. Otherwise, when the car moves forward, it may turn left, turn right, or even go backwards. Please follow the following steps to complete the calibration.

How to do

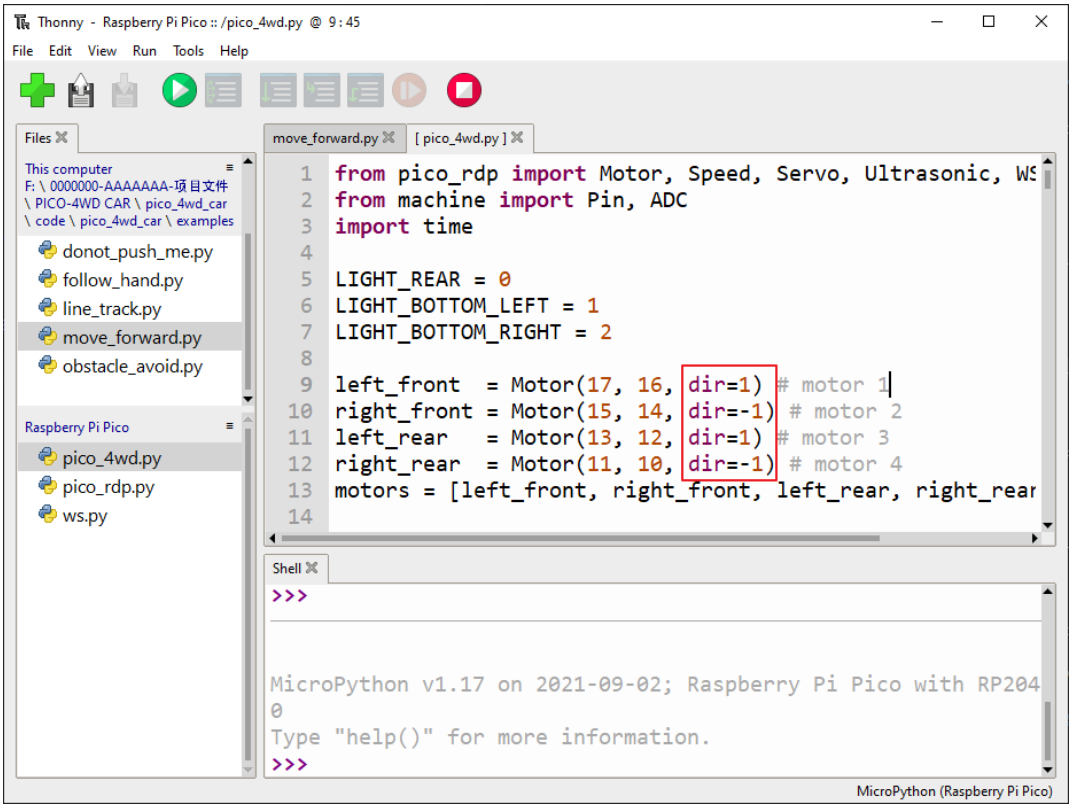
1. Select Correct Interpreter. Plug the Pico into your computer with a micro USB cable and select the "MicroPython (Raspberry Pi Pico).COMXX" interpreter in the bottom right corner.



2. Go to the `pico_4wd_car/examples` path and double click on `move_forward.py` to open it. This is a very simple example, which will let the car go forward.

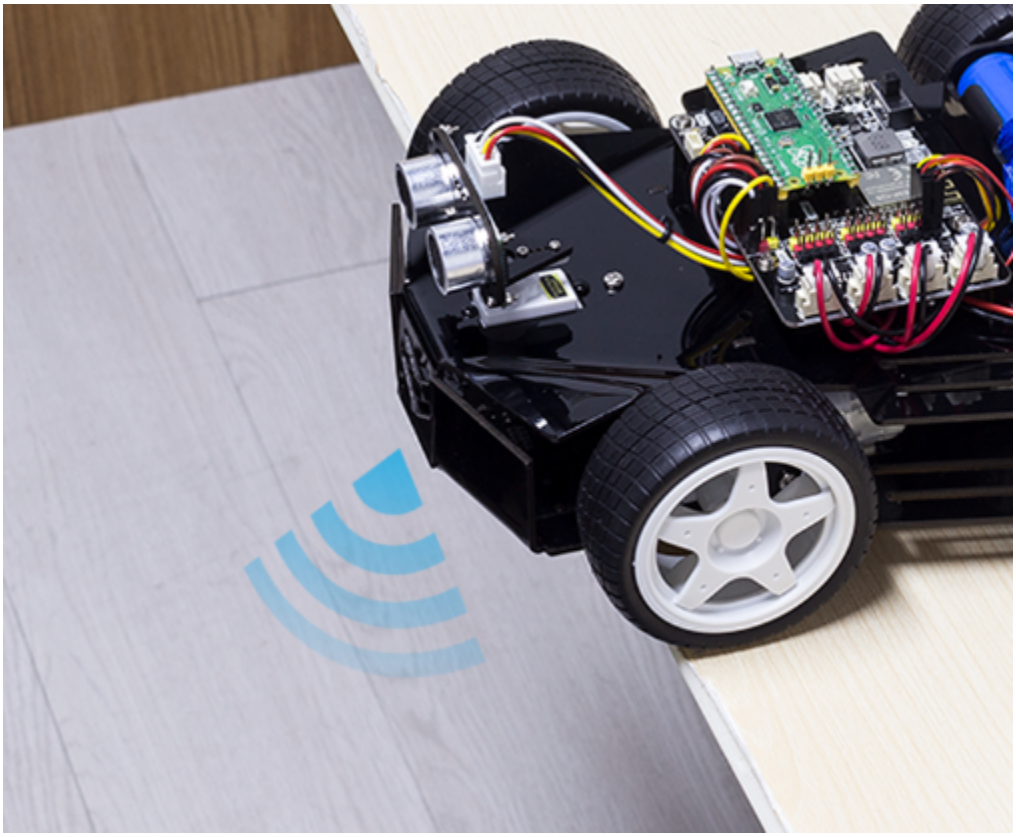


3. Observe whether the wheels of the car are driving forward and marked the wrong motor.
4. Open the `pico_4wd.py` file that was uploaded to the pico before, modify the wrong motor's `dir` to reverse the value and save it.



5. Run again `move_forward.py`, if the car is driving correctly, the calibration is completed.

1.6.2 Don't Push Me



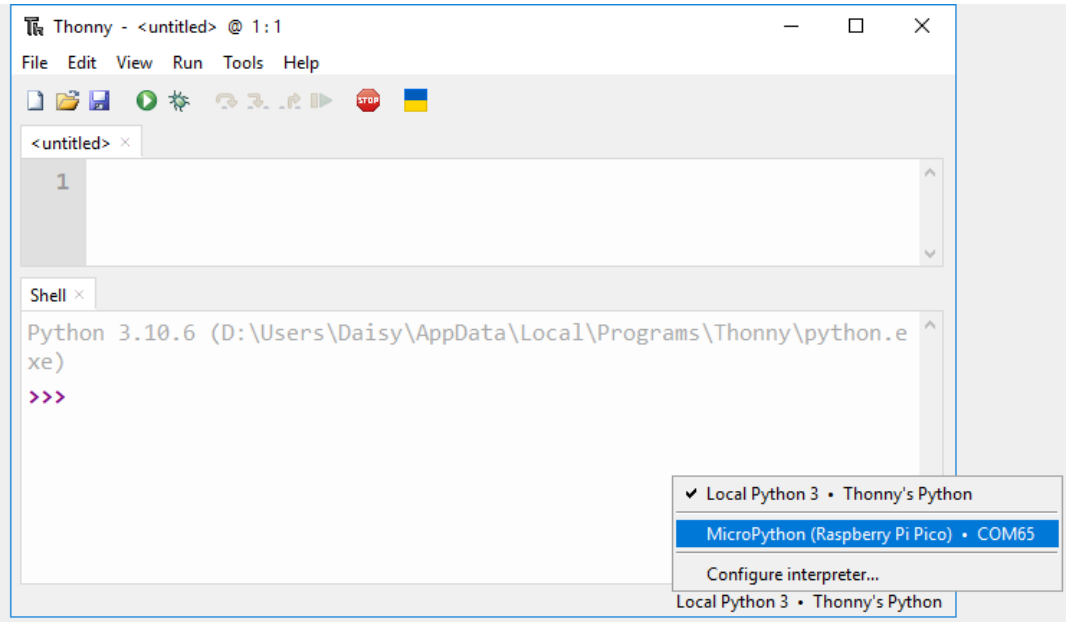
Let us give Pico-4wd a little **self-protection awareness** and let it learn to use its own grayscale module to avoid rushing down the cliff.

In this example, the car will be dormant. If you push it to a cliff, it will be awakened urgently, then back up, and shake its head to express dissatisfaction.

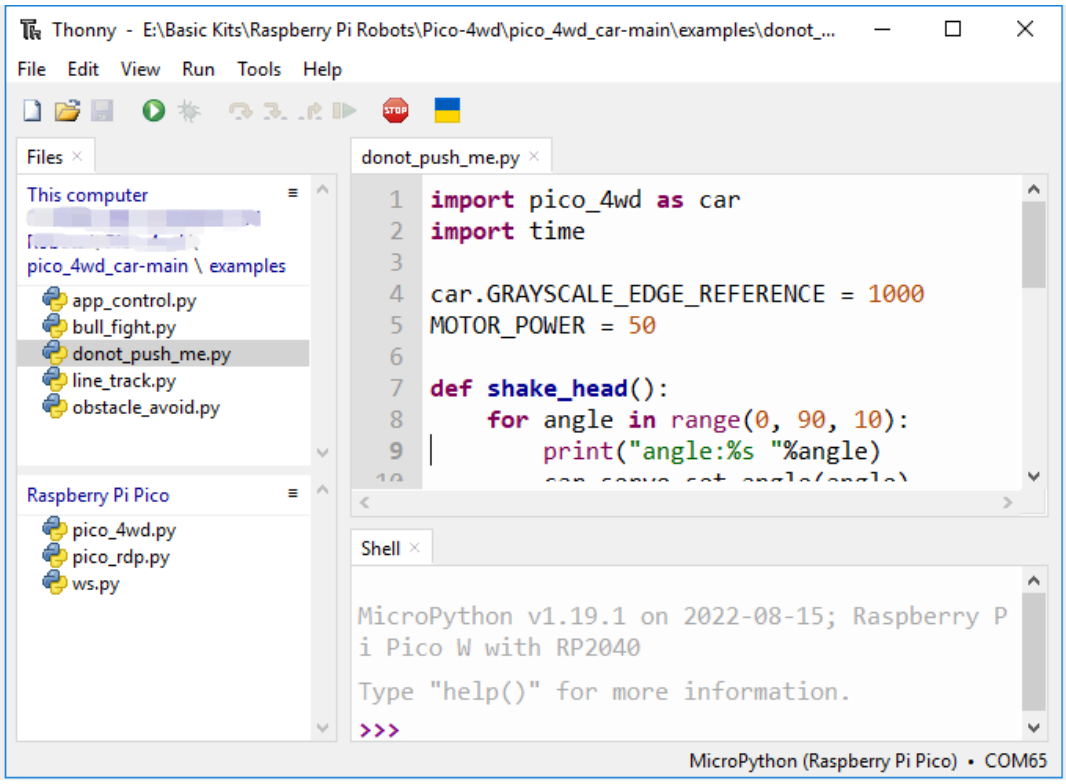
How to do

1. Select Correct Interpreter

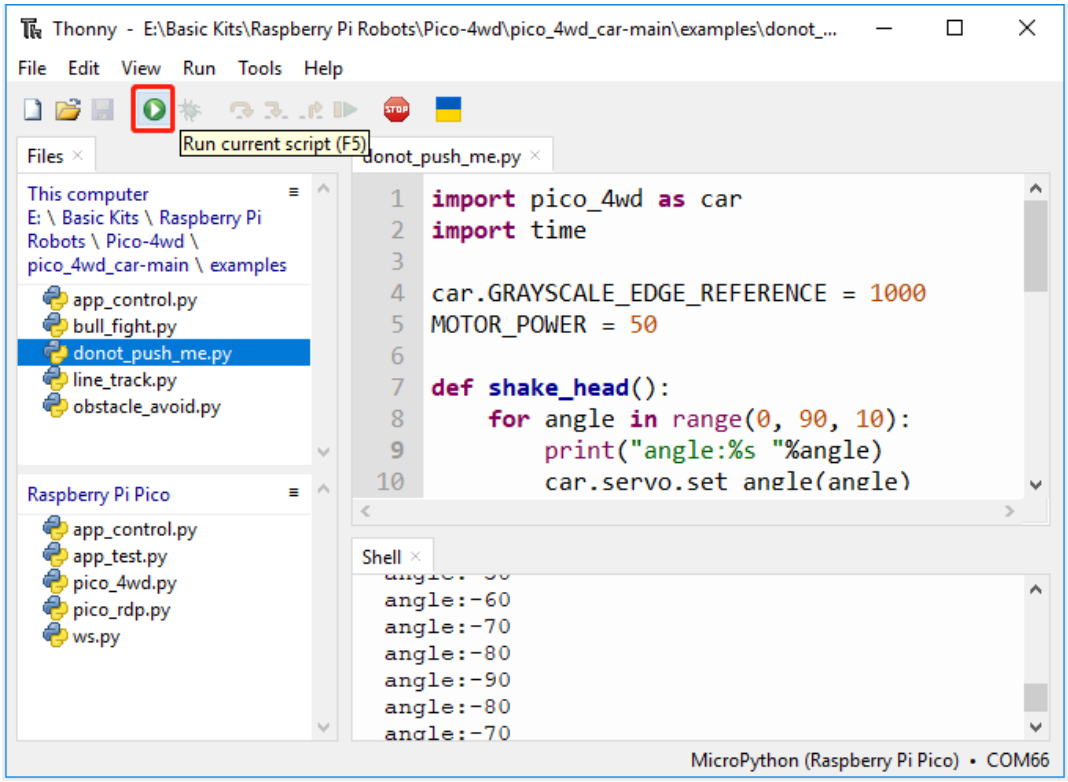
   Plug the Pico into your computer with a micro USB cable and select the "MicroPython (Raspberry Pi Pico).COMXX" interpreter in the bottom right corner.



2. Go to the `pico_4wd_car_main/examples` path and double click on `donot_push_me.py` to open it.



3. Click the **Run current script** button or just press F5 to run it.



After the code runs, if you push it to a cliff, it will be awakened urgently, then back up, and shake its head to express dissatisfaction.

Note: If the result is not satisfactory, please modify the fourth line `car.GRAYSCALE_EDGE_REFERENCE = 1000`.

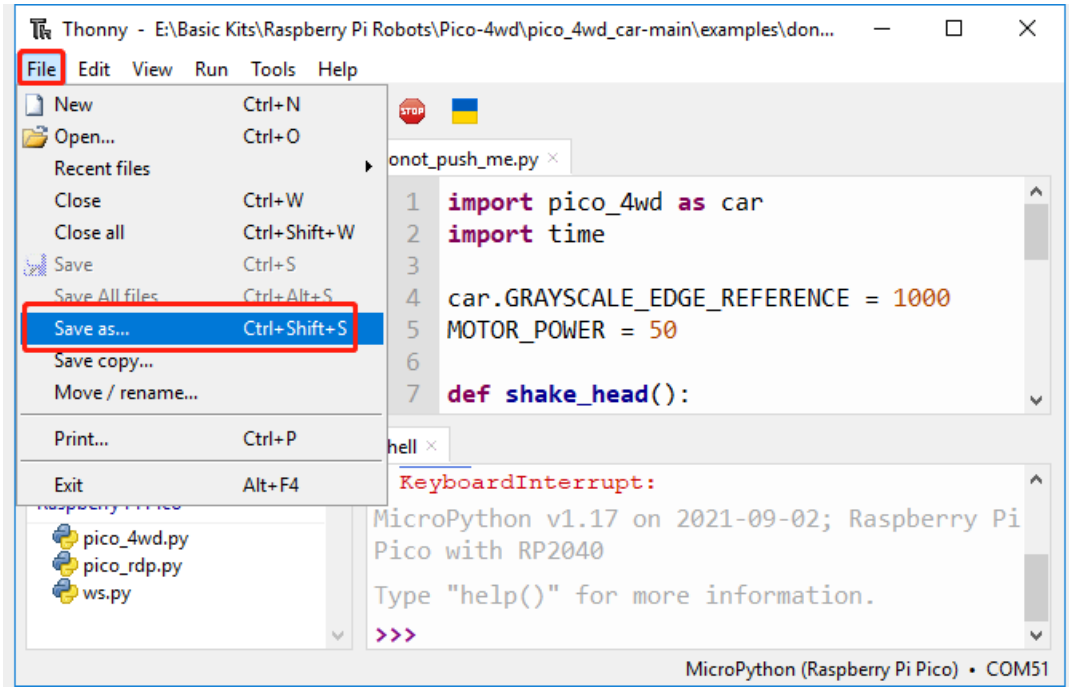
This is a threshold that tells the car if the grayscale sensor detected values are below 1000, then it should be considered a cliff and can move backward.

This threshold can be obtained in the *Test the Grayscale Sensor Module* section, for example, in my test, the detected cliff value is basically around 700, so here it is set to 1000, as long as it is lower than 1000, it can be considered to detect a cliff.

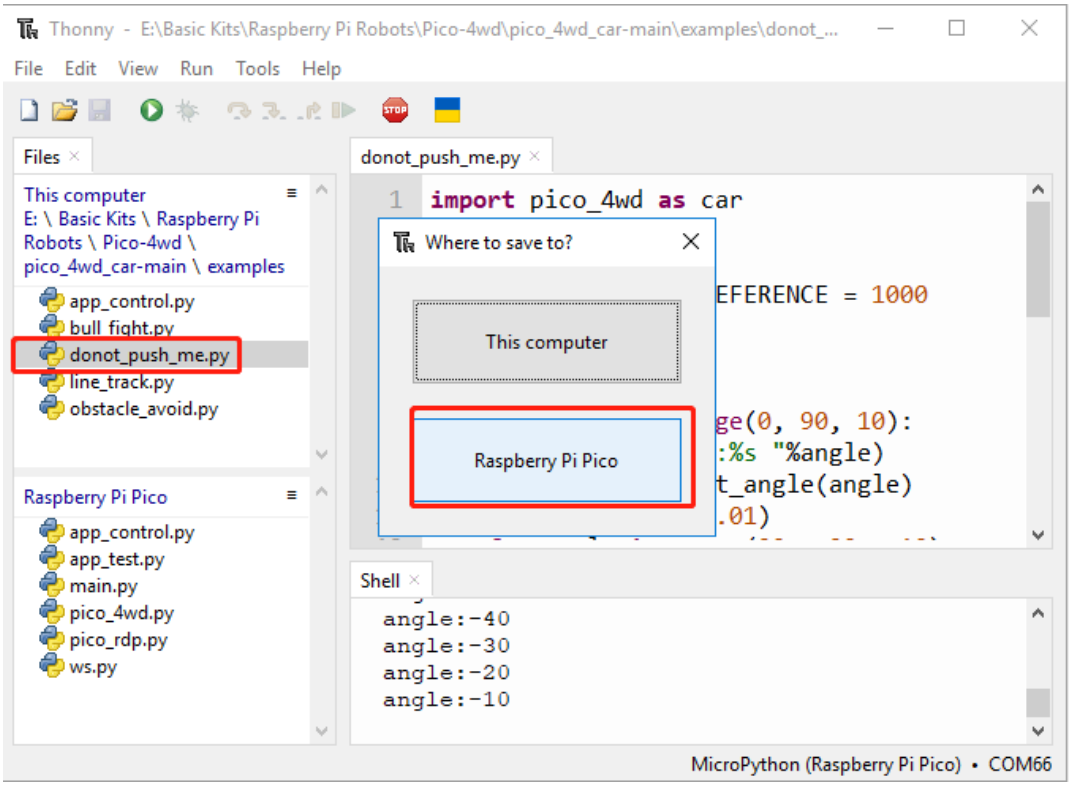
---

4. To enable Pico-4wd to run this code on boot, you need to save `donot_push_me.py` to the Raspberry Pi Pico as `main.py`, as follows.

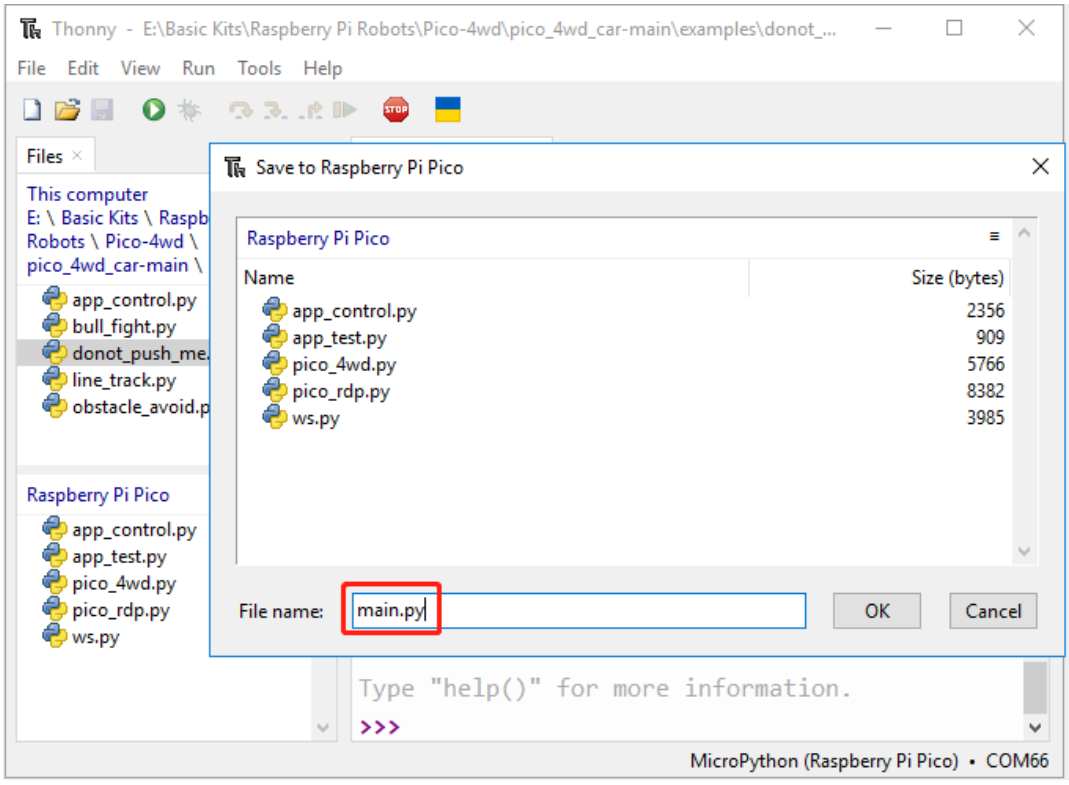
    - Stop the script from running and click **File -> Save as**.



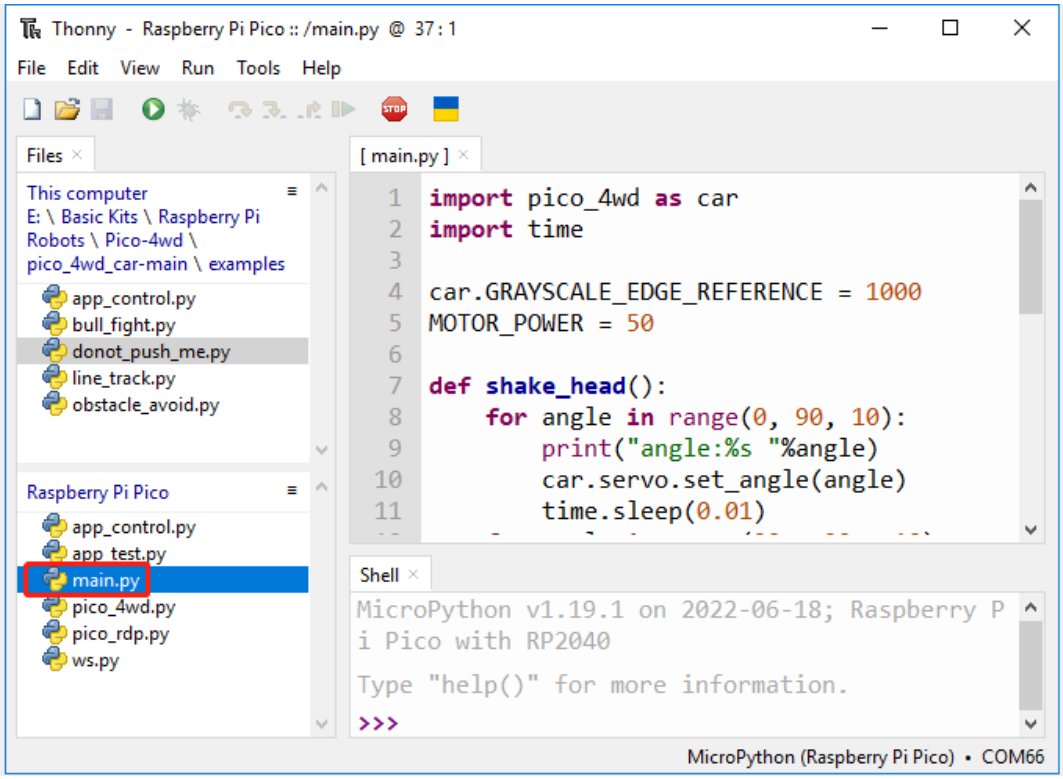
- Select **Raspberry Pi Pico** in the popup window that appears.



- Set the file name to `main.py`.

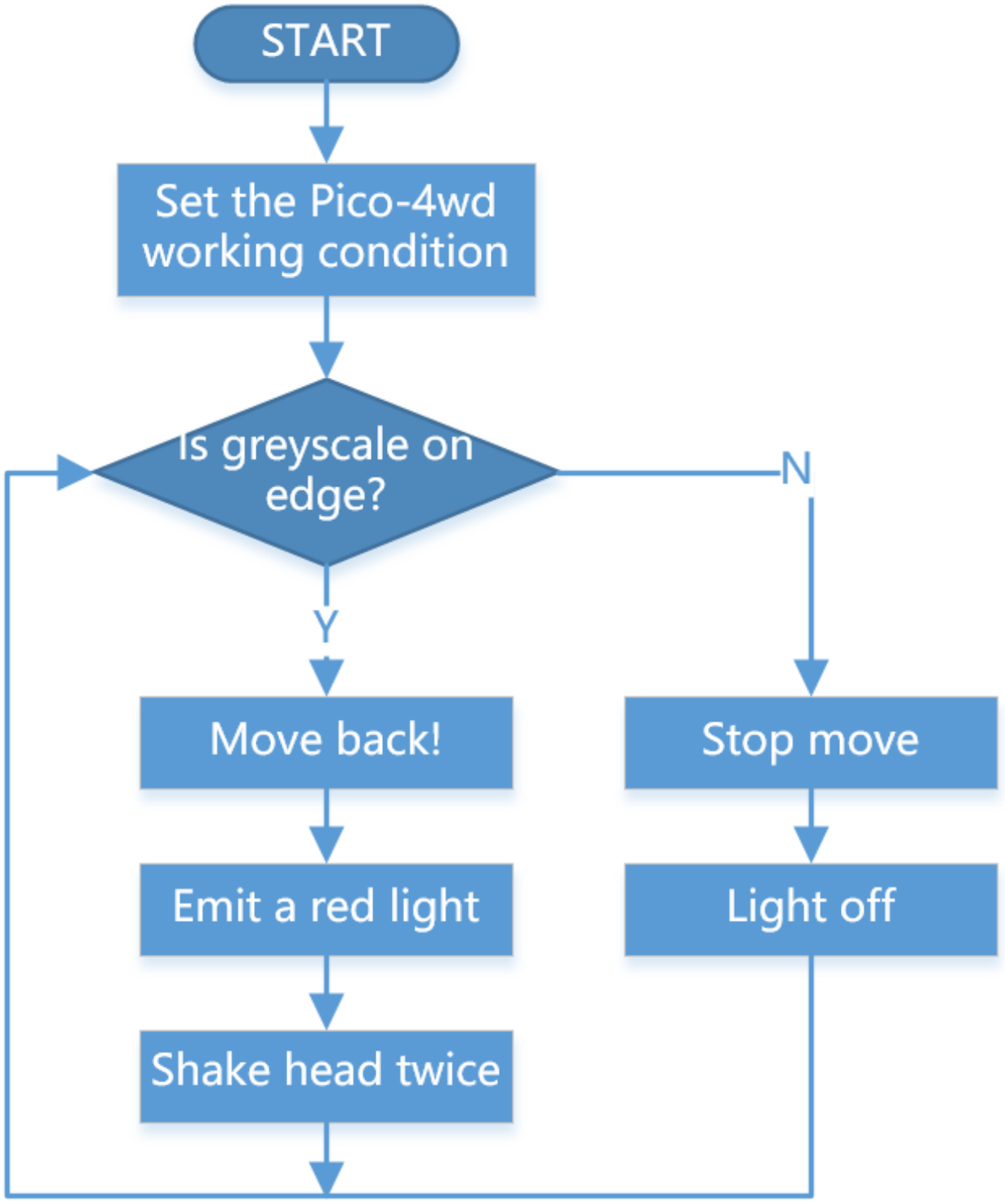


- After clicking **OK**, the Raspberry Pi Pico will have an additional `main.py` file.

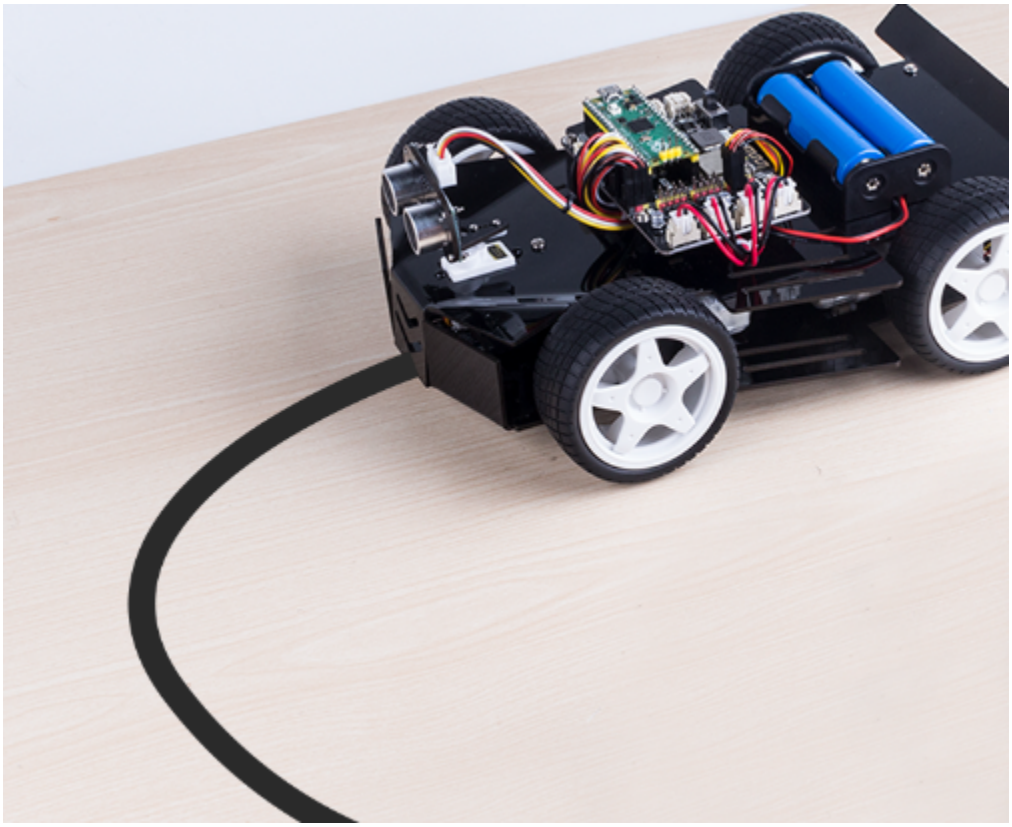


- Afterwards, you can unplug the USB cable, turn on the power switch on the car, and the script `main.py` will begin running.

How it Work?


START
Set the Pico-4wd working condition
Is greyscale on edge?
N
Y
Move back!
Stop move
Emit a red light
Light off
Shake head twice

1.6.3 Line Track

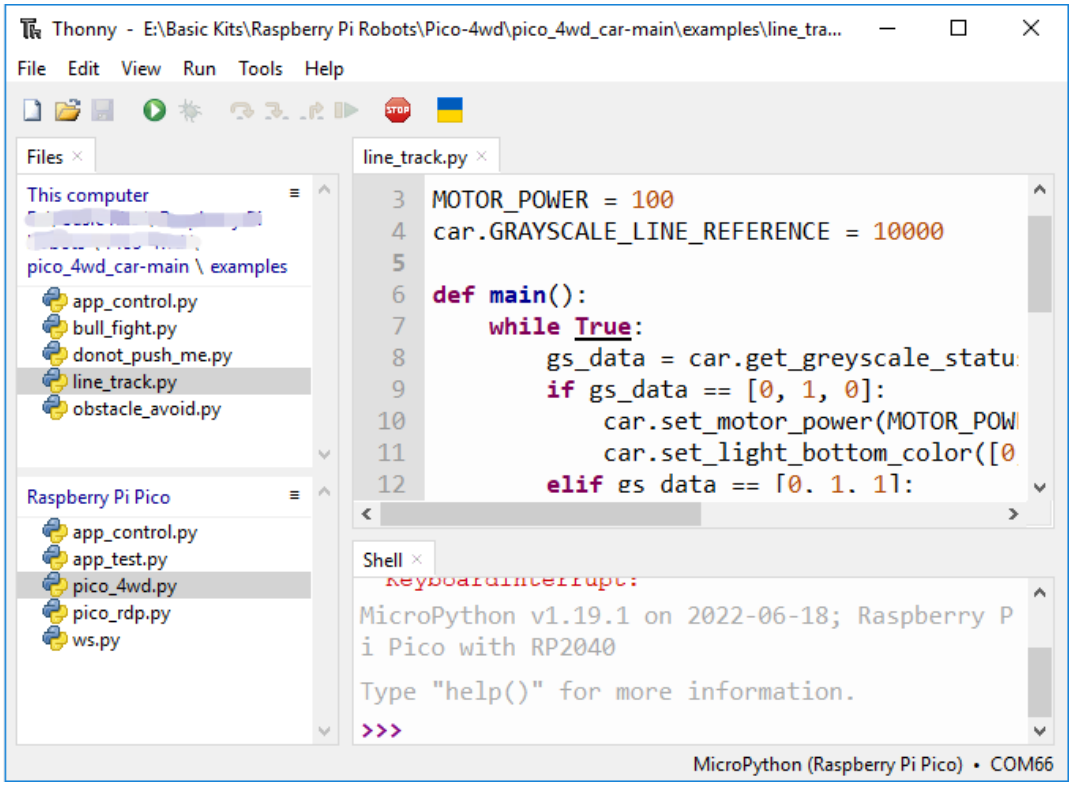


Let Pico-4wd walk on its exclusive avenue! Tape a line on a light-colored ground (or table) with black insulating tape. Run this script and you will see Pico-4wd track the line to forward.

> **Warning:** When pasting this line, there should be no sharp turns so that the car does not drive off the path.

How to do?

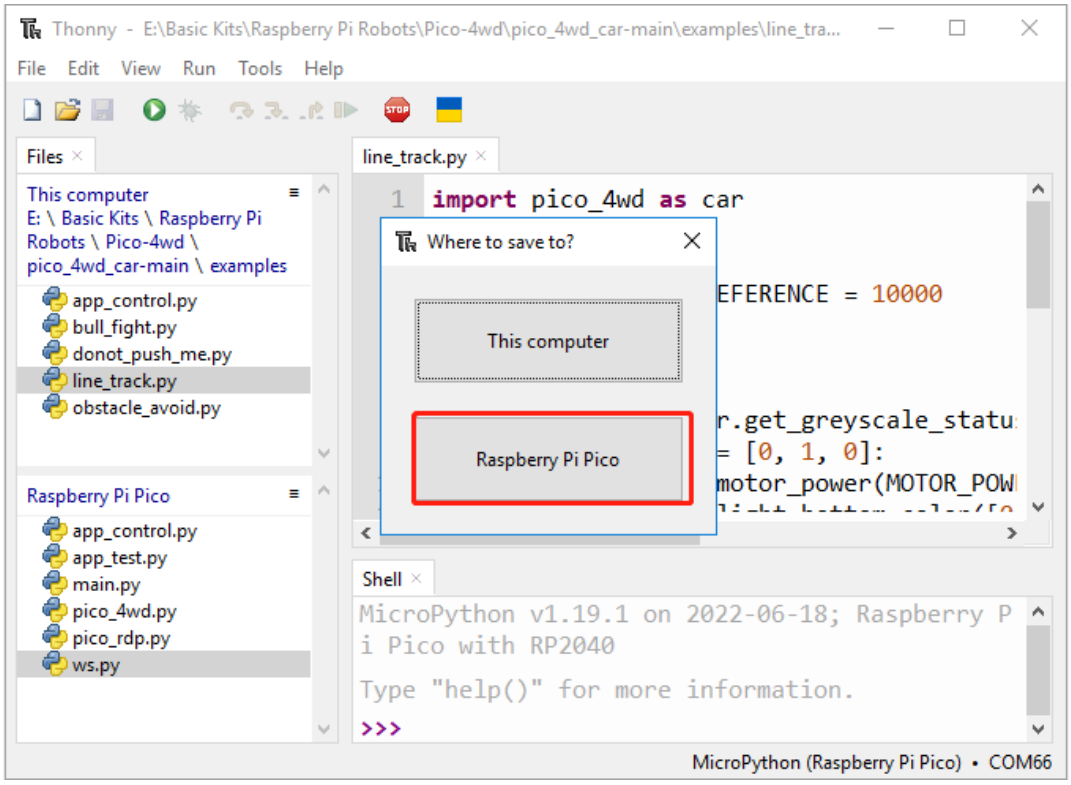
1. Open the `line_track.py` file under the path of `pico_4wd_car_main\examples`.



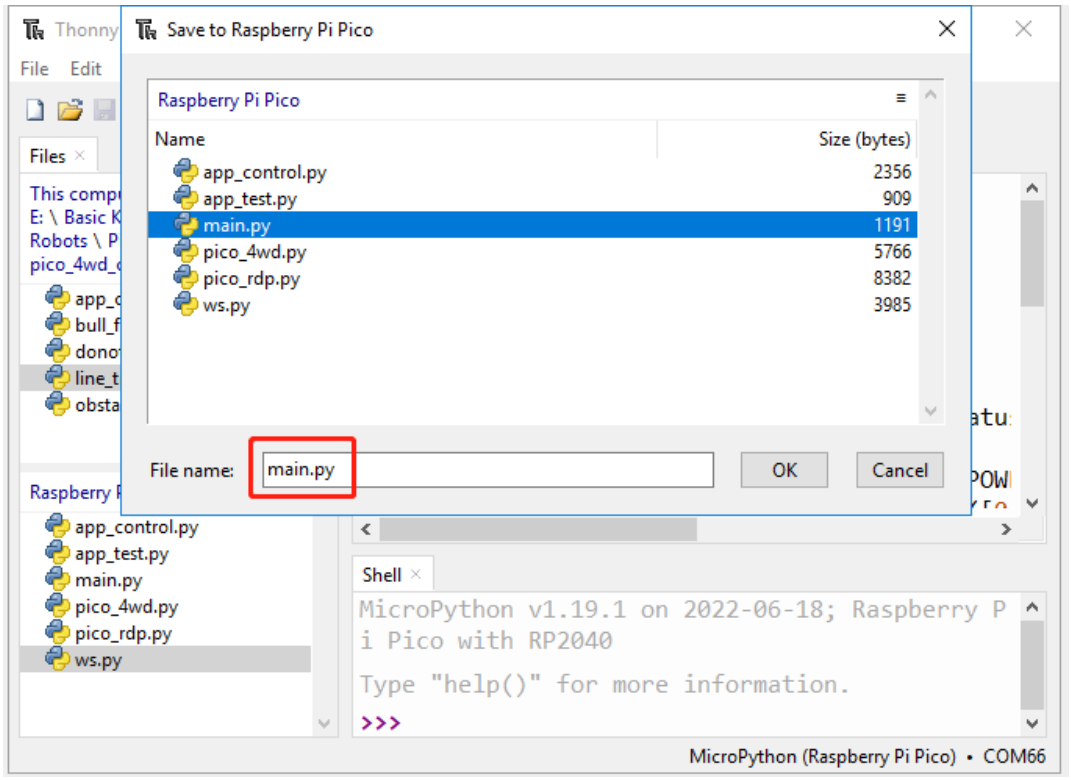
2. The following value should be modified based on the results of the *Test the Grayscale Sensor Module* section. It should be between the values of black and white surfaces. For example, in my tests, black surfaces generally around 3000 and white surfaces are around 20000, so 10000 is the appropriate threshold value. According to the actual situation, you can modify it to other values according to the actual situation.

```
car.GRAYSCALE_LINE_REFERENCE = 10000
```

3. Click **File** -> **Save as** or press `Ctrl+Shift+S`, and then select **Raspberry Pi Pico** in the popup window that appears.



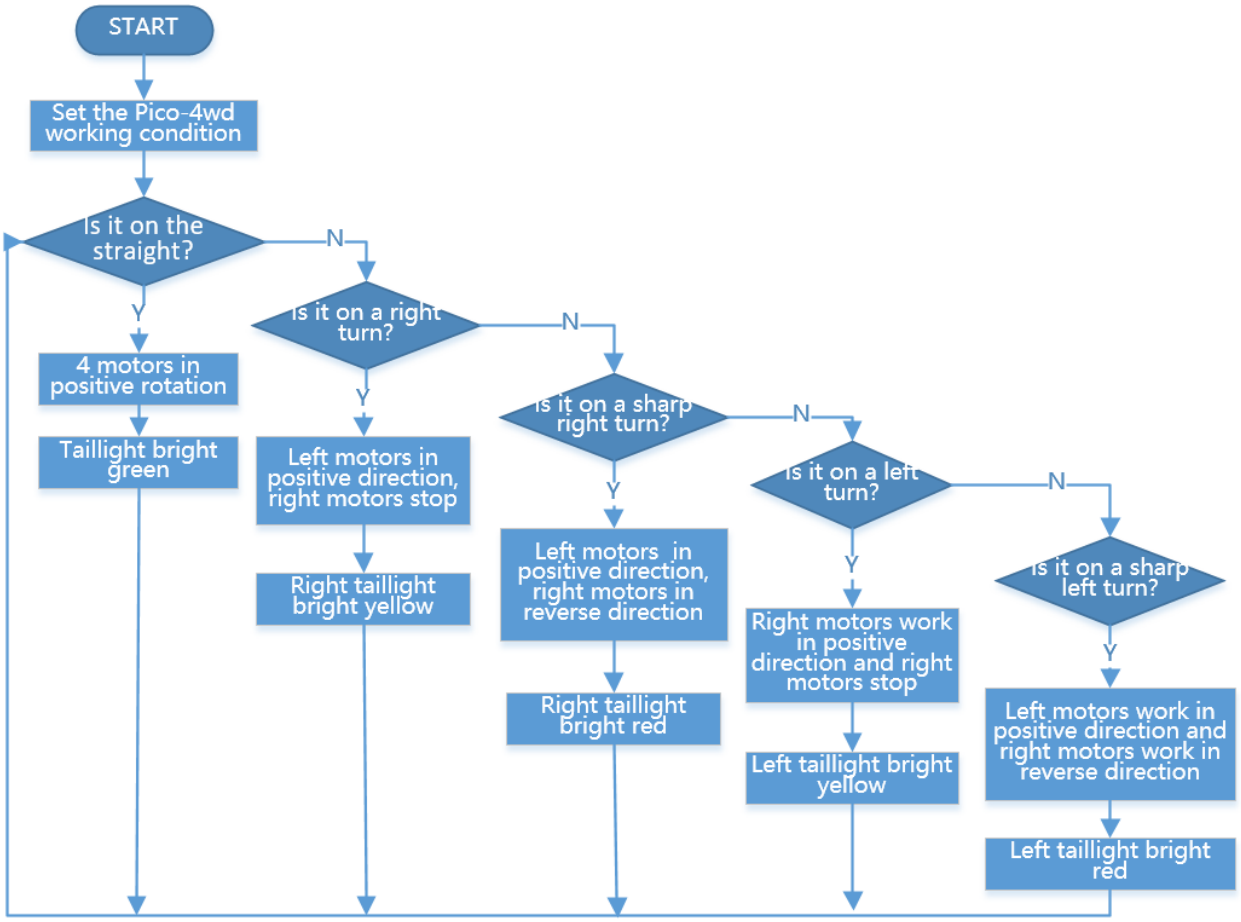
4. Set the file name to `main.py`. If you already have the `main.py` file in your Pico, it will prompt to overwrite the `main.py` file.



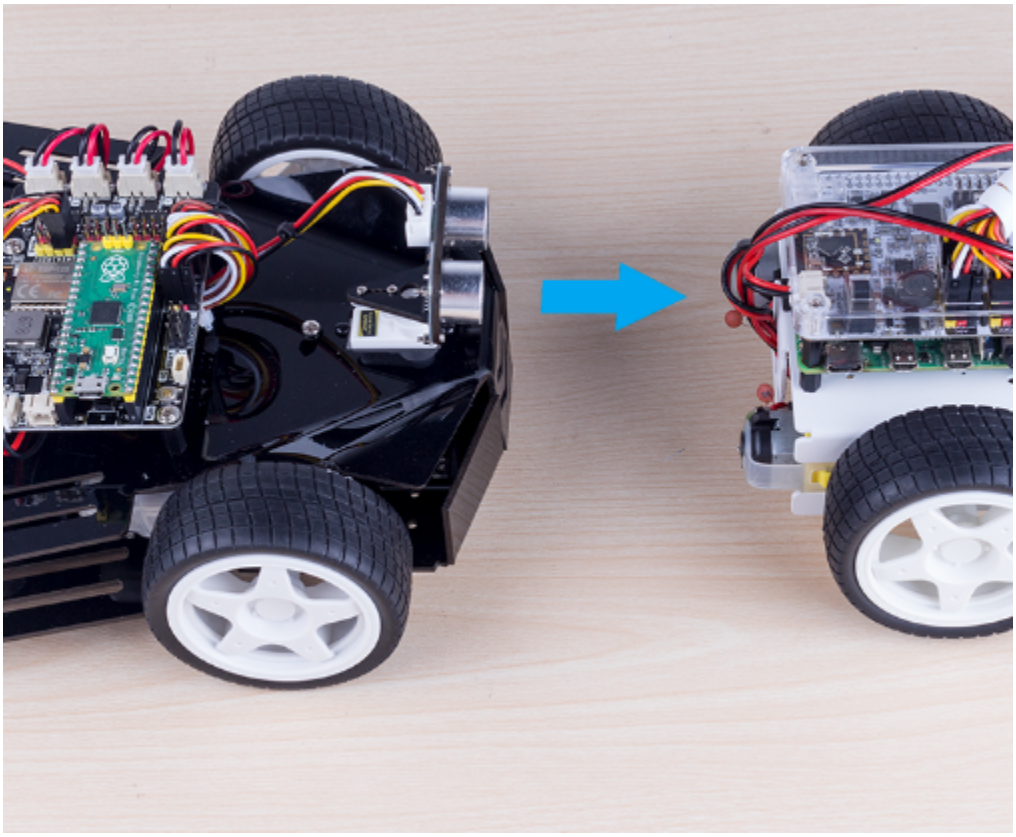
#. Now you can unplug the USB cable, place the car on the taped line, turn on the power switch and it will track the

line. The script also includes some lighting effects: when the Pico-4wd is going straight, the taillight (the RGB panel on the tail) will light up green; when turning, one side of the taillight will light up yellow. During sharp turns, one side of the taillight will light up red.

How it Works?



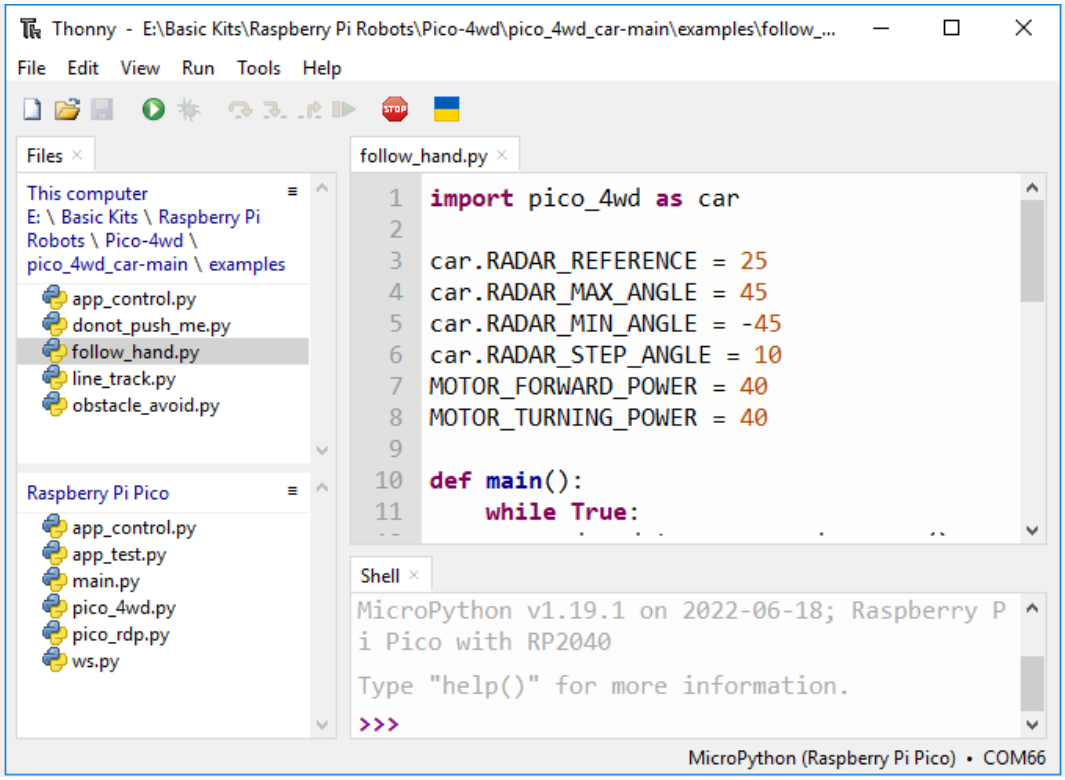
1.6.4 Follow Your Hand



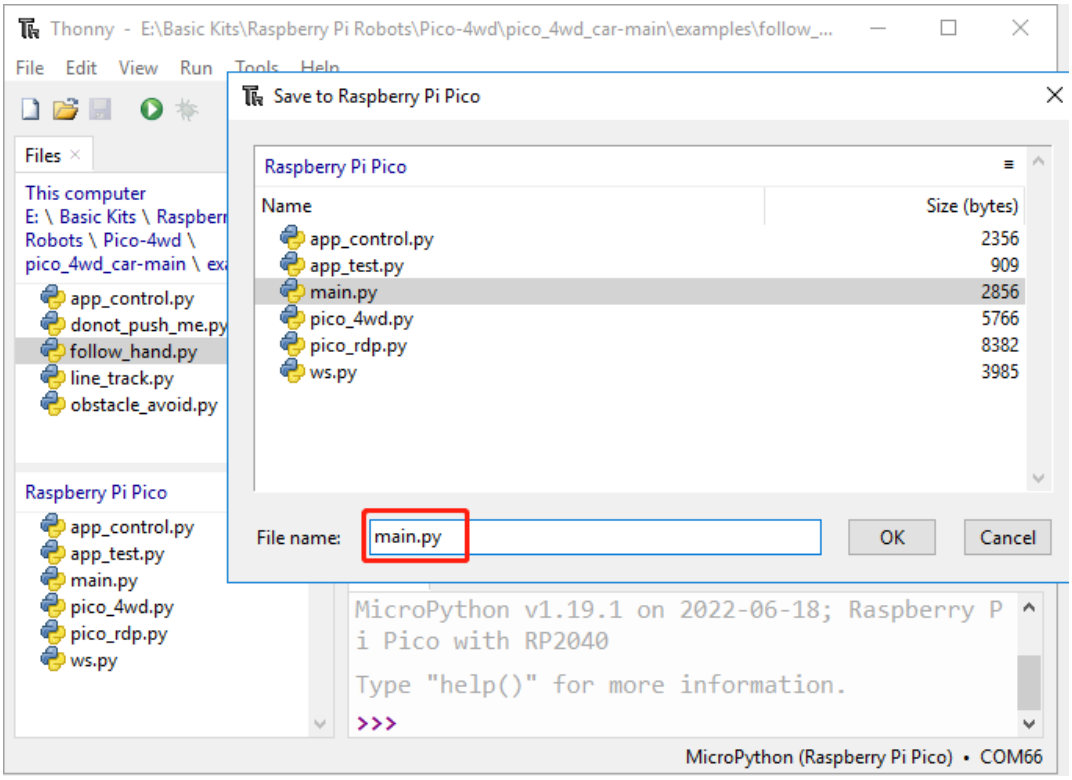
Here, Pico-4wd will follow your hand moving forward.

How to do?

1. Open the `follow_hand.py` file under the path of `pico_4wd_car_main\examples`.

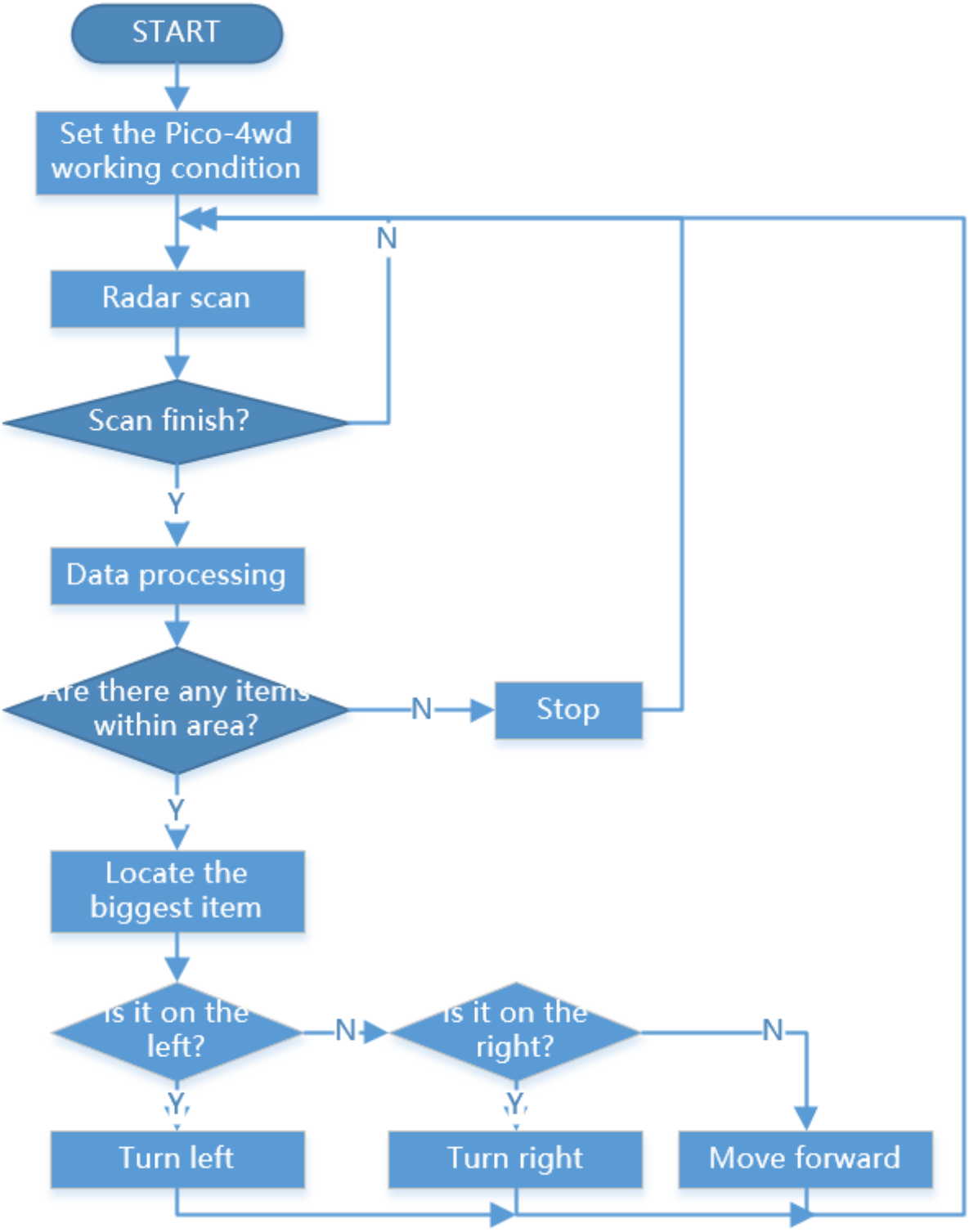


2. Click **File** -> **Save as** or press `Ctrl+Shift+S` to save `donot_push_me.py` to the Raspberry Pi Pico as `main.py`.

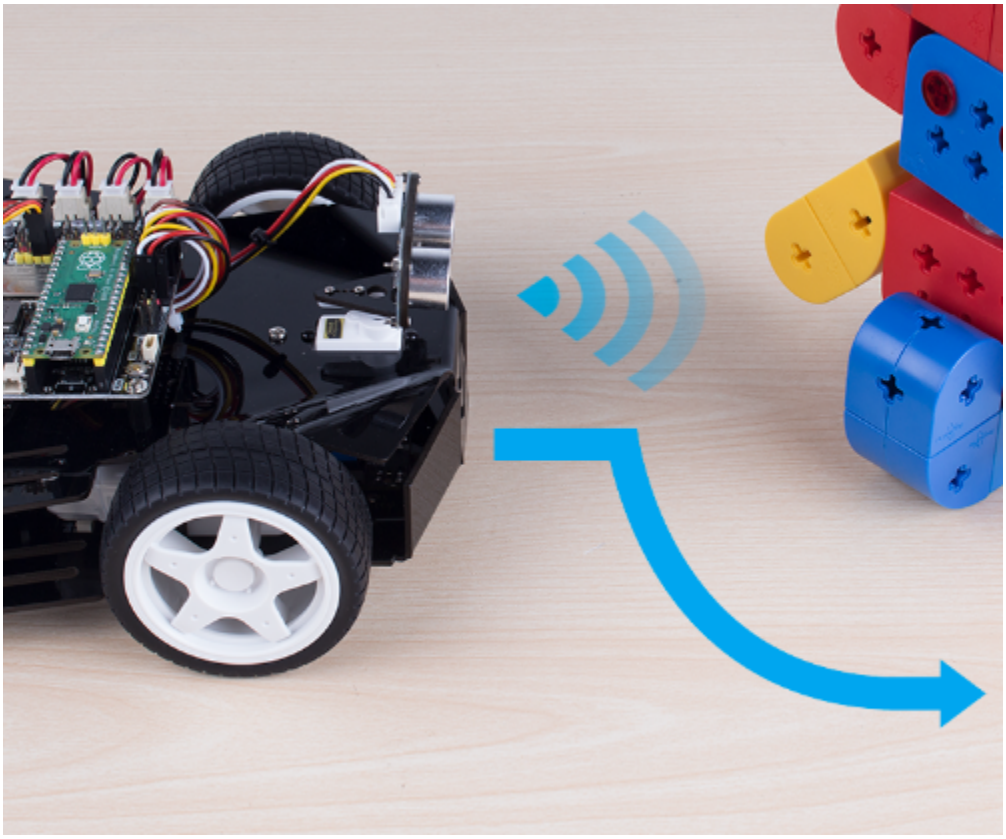


3. Unplug the USB cable and turn on the power switch. When you put your hand in front of it, it will follow.

How it works?


START
Set the Pico-4wd working condition
Radar scan
Scan finish?
N
Y
Data processing
Are there any items within area?
N
Stop
Y
Locate the biggest item
Is it on the left?
N
Is it on the right?
N
Y
Y
Turn left
Turn right
Move forward

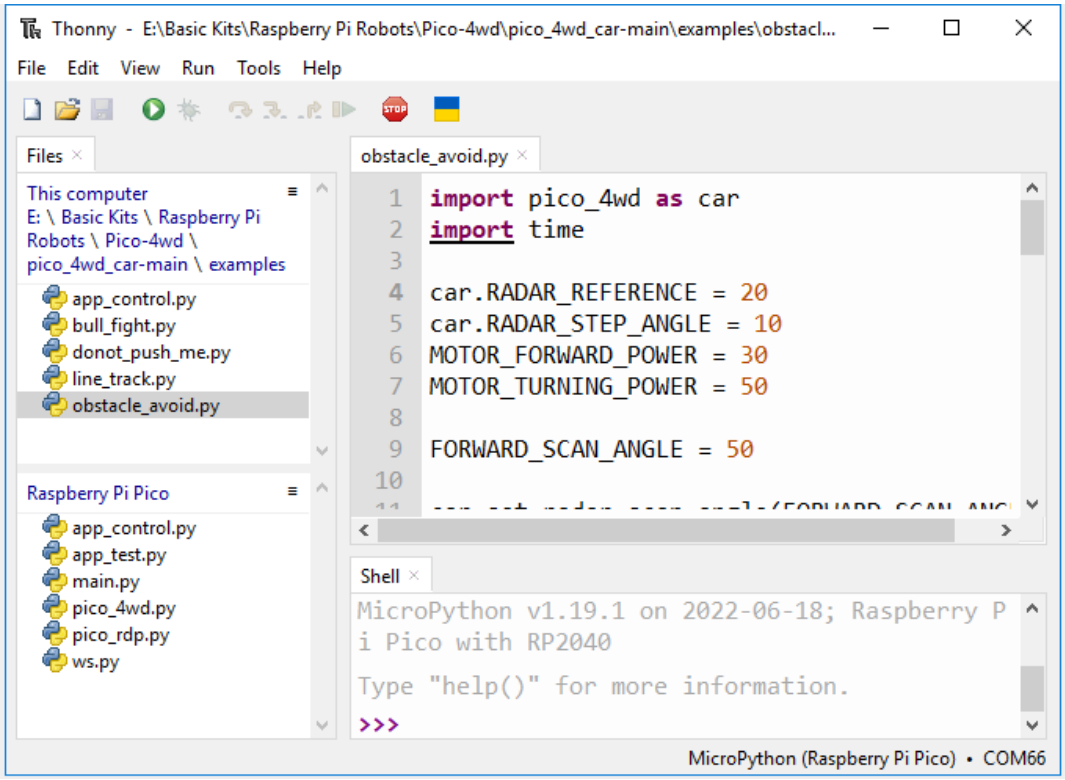
1.6.5 Obstacle Avoid



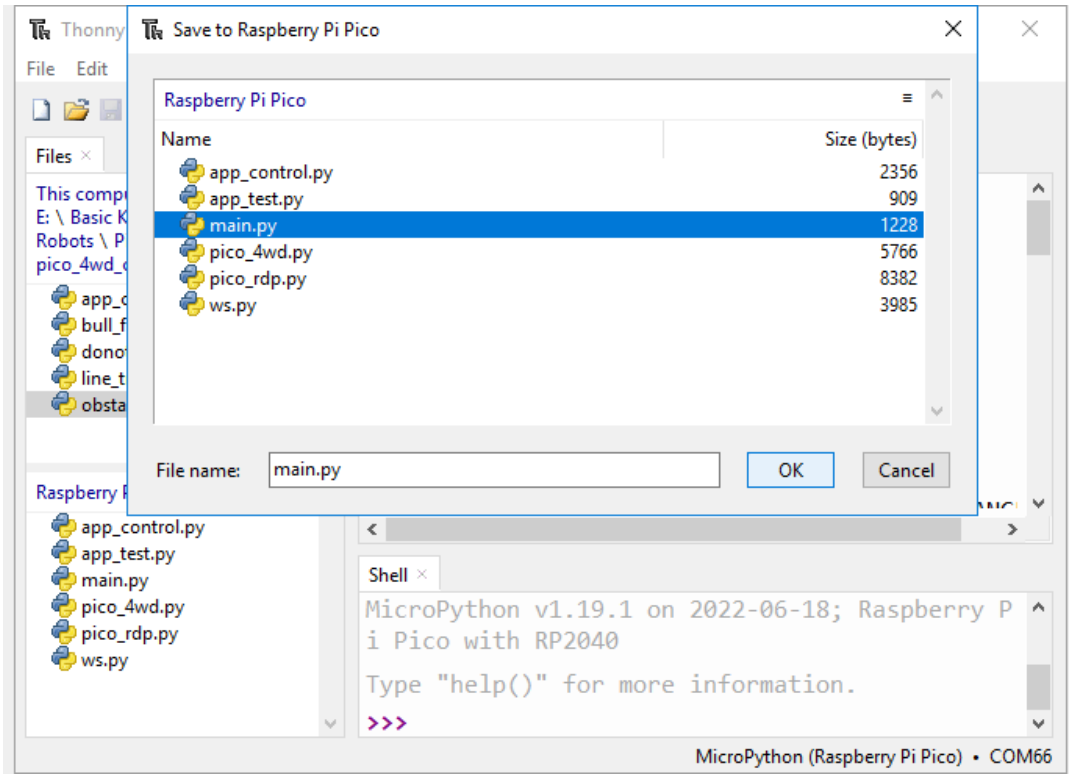
Let Pico-4wd do a challenging task: automatically avoid obstacles! When an obstacle is detected, instead of simply backing up, the radar scans the surrounding area and finds the widest way to move forward.

How to do?

1. Open the `obstacle_avoid.py` file under the path of `pico_4wd_car_main\examples`.

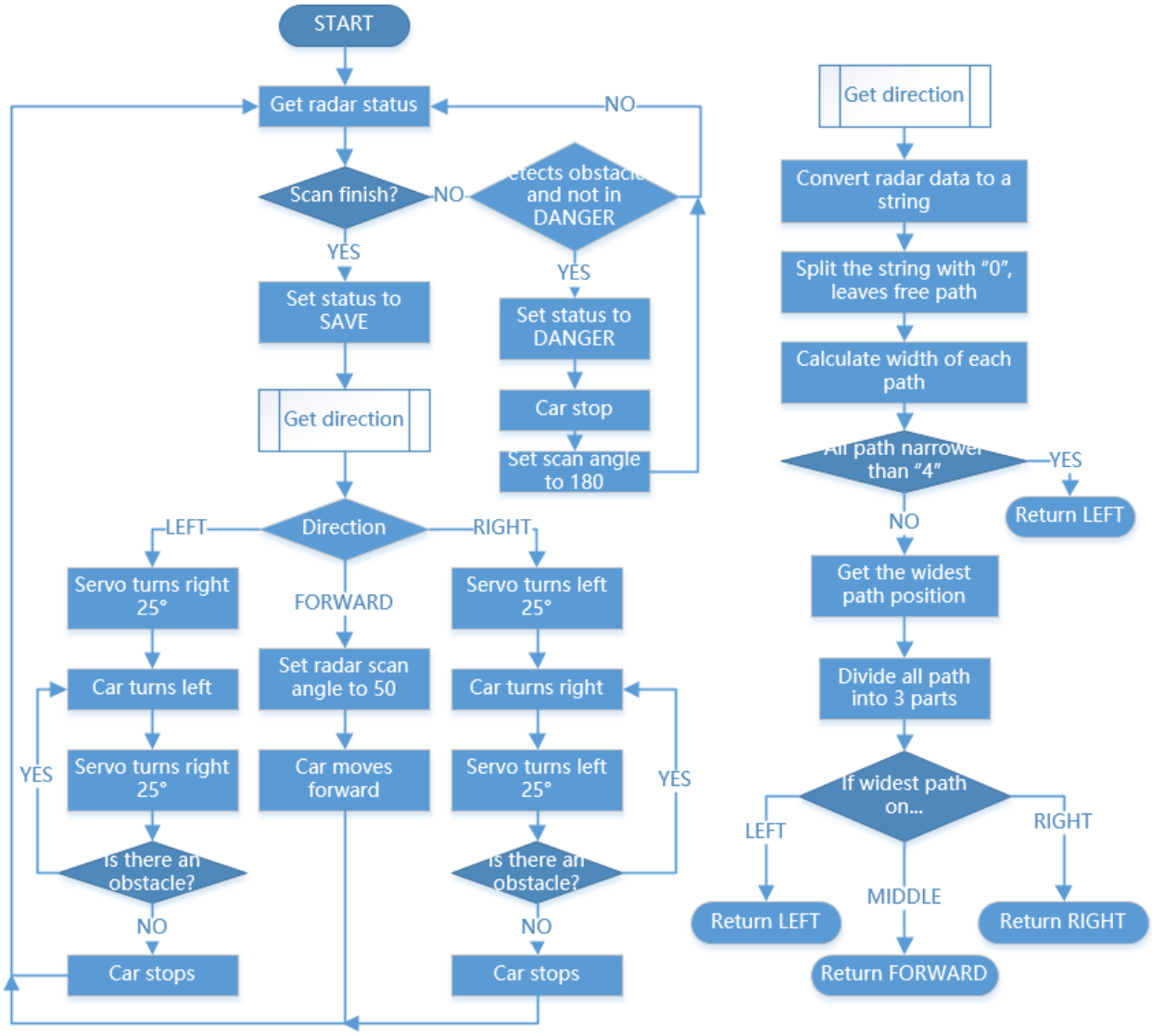


2. Click **File** -> **Save as** or press `Ctrl+Shift+S` to save `donot_push_me.py` to the Raspberry Pi Pico as `main.py`.



3. Unplug the USB cable and turn on the power switch. When you place it on the ground, it avoids the obstacles and keeps going.

How it Works?



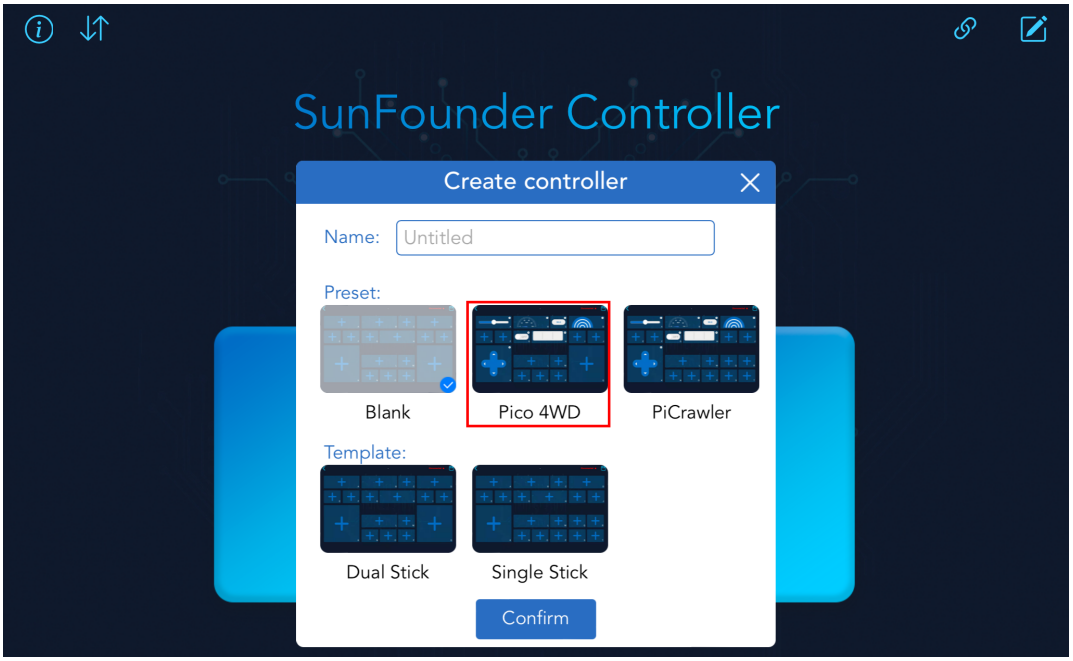
1.6.6 Control the Car with APP

This section will guide you through building a remote project using the Sunfounder Controller APP, which means you can use your phone/tablet to control your Pico-4wd car.

1. Setup SunFounder Controller

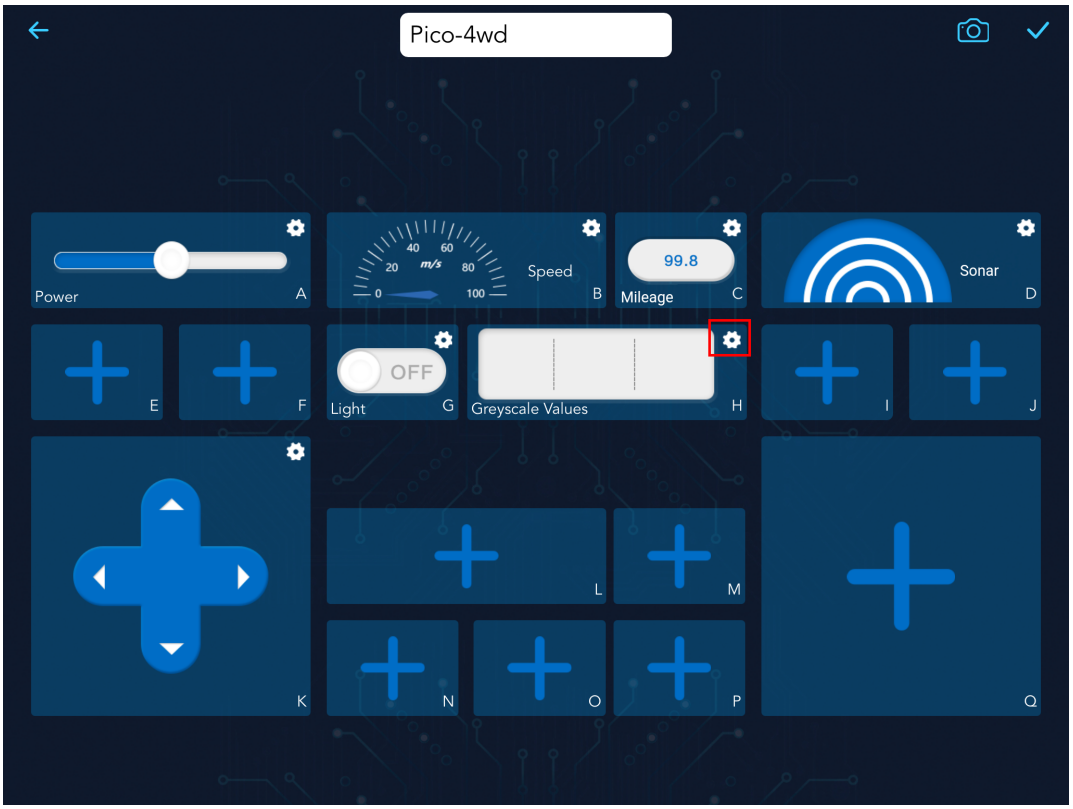
1. Install APP from APP Store(iOS) or Google Play(Android).
2. Open SunFounder Controller and click on the **+** to create a new controller.

3. We have preset controller for Pico-4wd, you can choose it directly.

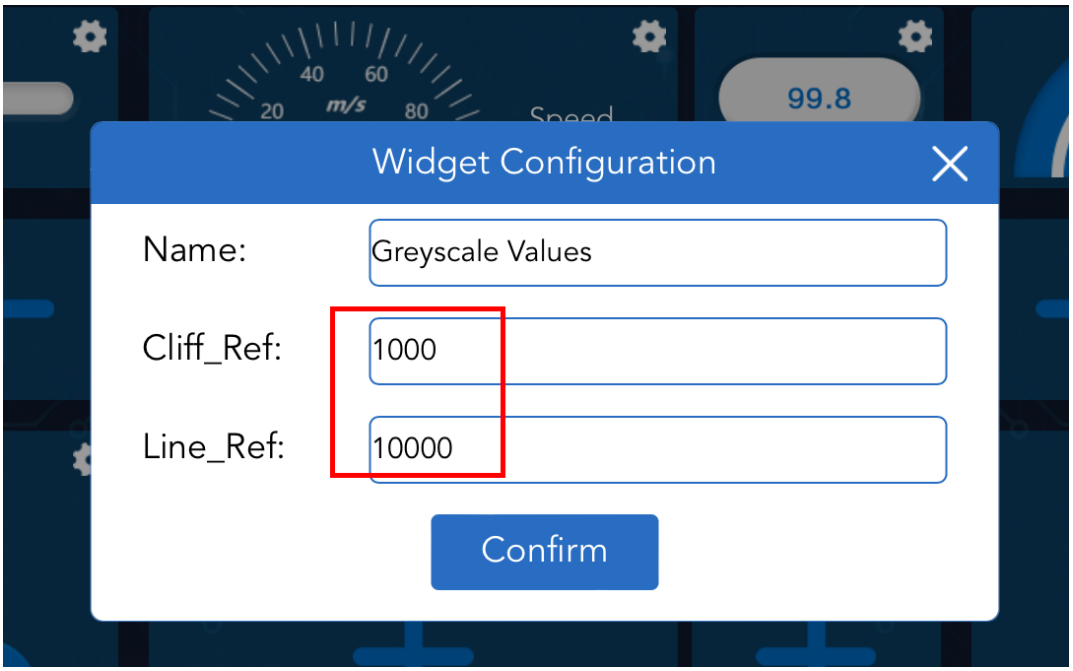


4. Define a name for this Controller and click **Confirm**.

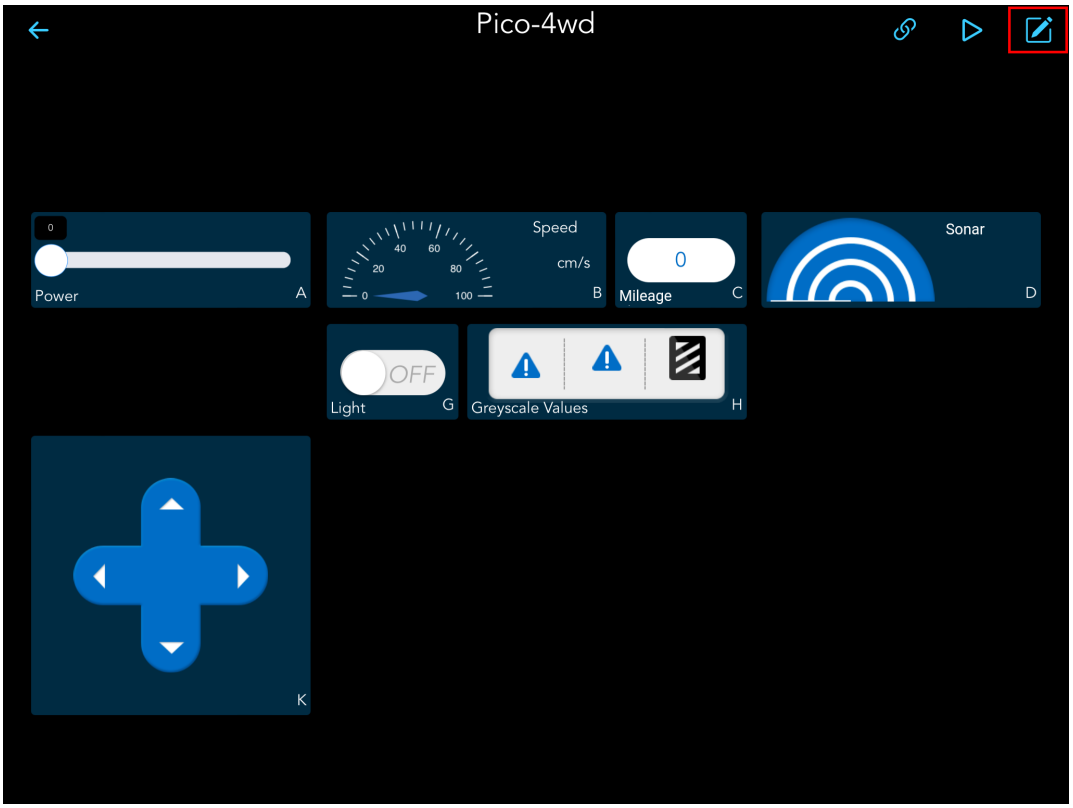
5. Now that you are inside this preset controller, click the Grayscale Values widget's Settings button to change its reference value.



6. The following thresholds should be modified based on the values obtained in the *Test the Grayscale Sensor Module* section.



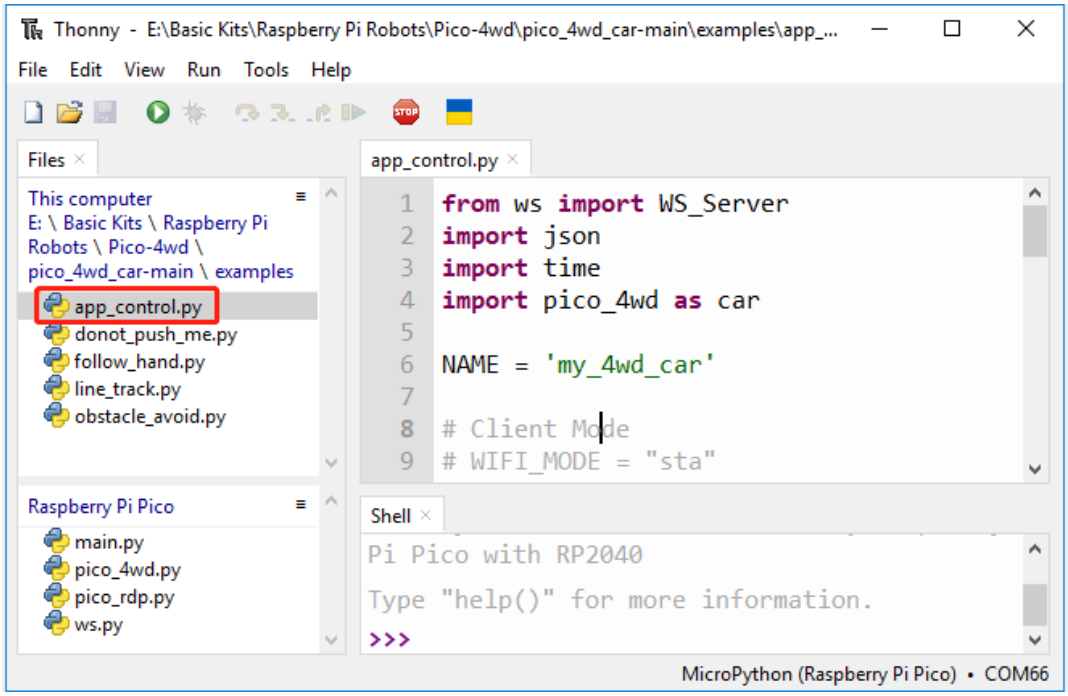
7. Once you have set it up, click the Save button in the upper right corner to save it. If you need to make any changes, click the Edit button again.



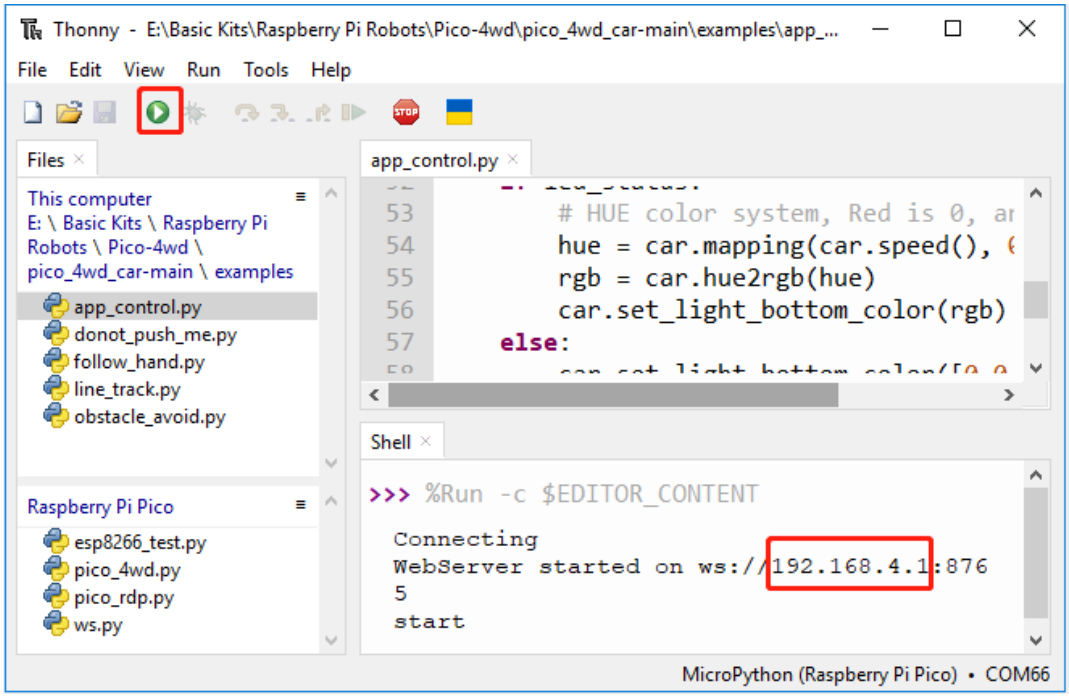
2. Make the app_control.py file run on boot

In order for Pico-4wd to be able to run specific scripts without being connected to a computer, and then be controlled by SunFounder Controller. You need to save the specific script to the Raspberry Pi Pico with the name main.py, as follows.

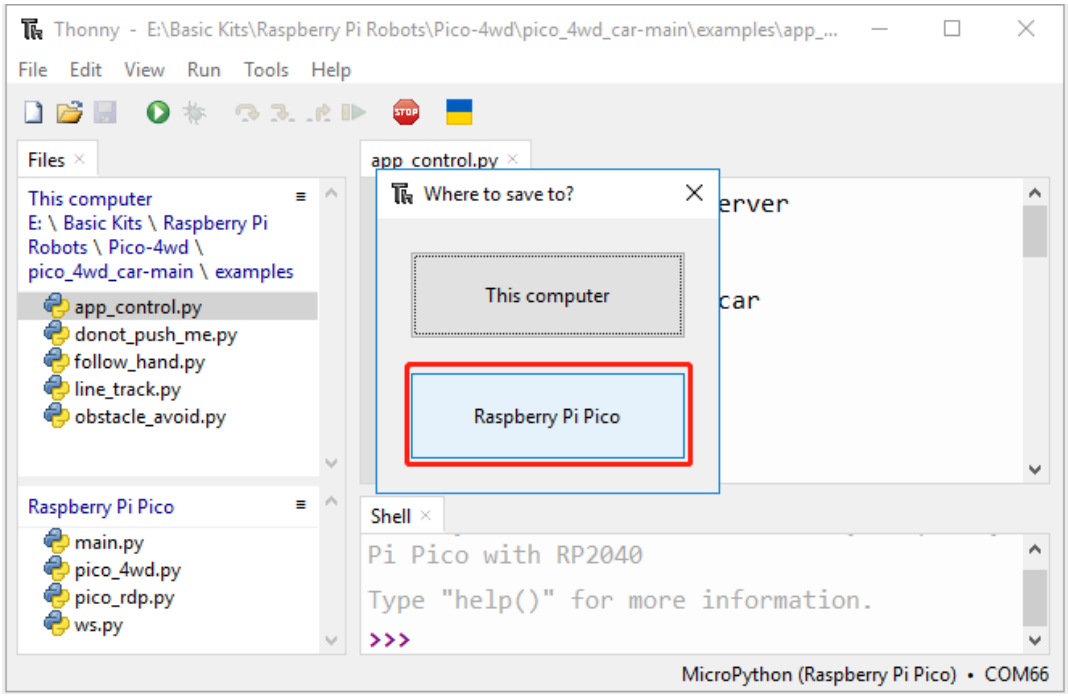
1. Open the `app_control.py` file under the path of `pico_4wd_car_main\examples`, then click "run current script" button or just press F5 to run it.



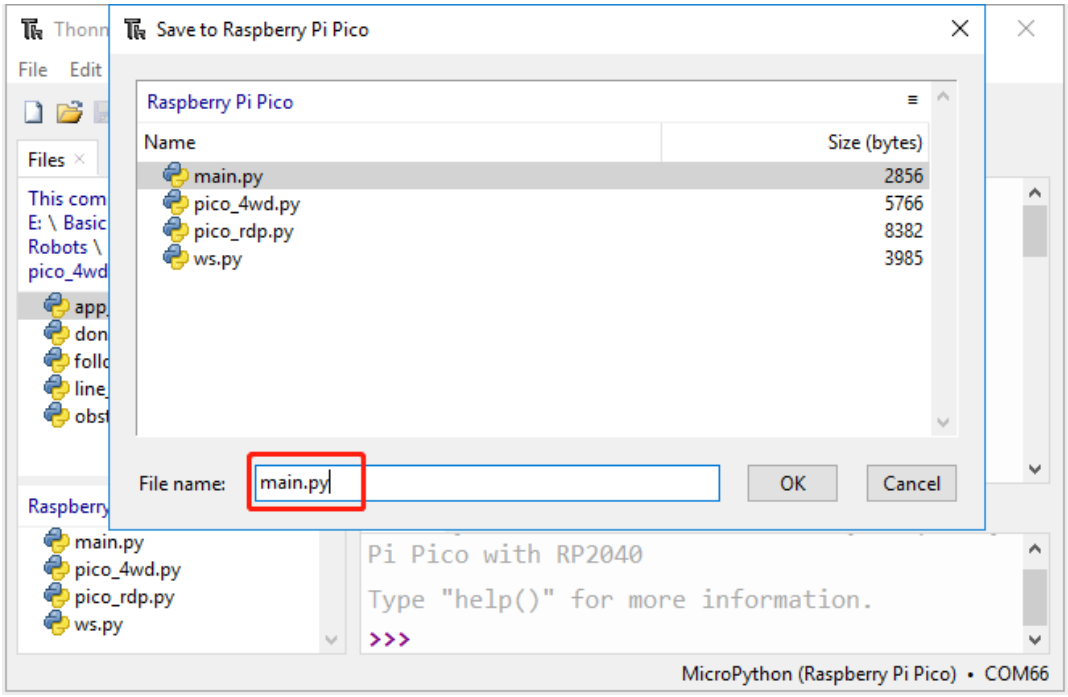
2. If you run the code, you will see this **IP** in the shell; remember it, as it may be used to connect to SunFounder Controller later on.



3. Stop the script from running and click **File -> Save as** or press `Ctrl+Shift+S`, then select **Raspberry Pi Pico** in the popup window that appears. If this pop-up does not appear on yours, make sure you have plugged the Pico into your computer with a micro USB cable and select the "MicroPython (Raspberry Pi Pico).COMXX" interpreter in the bottom right corner.



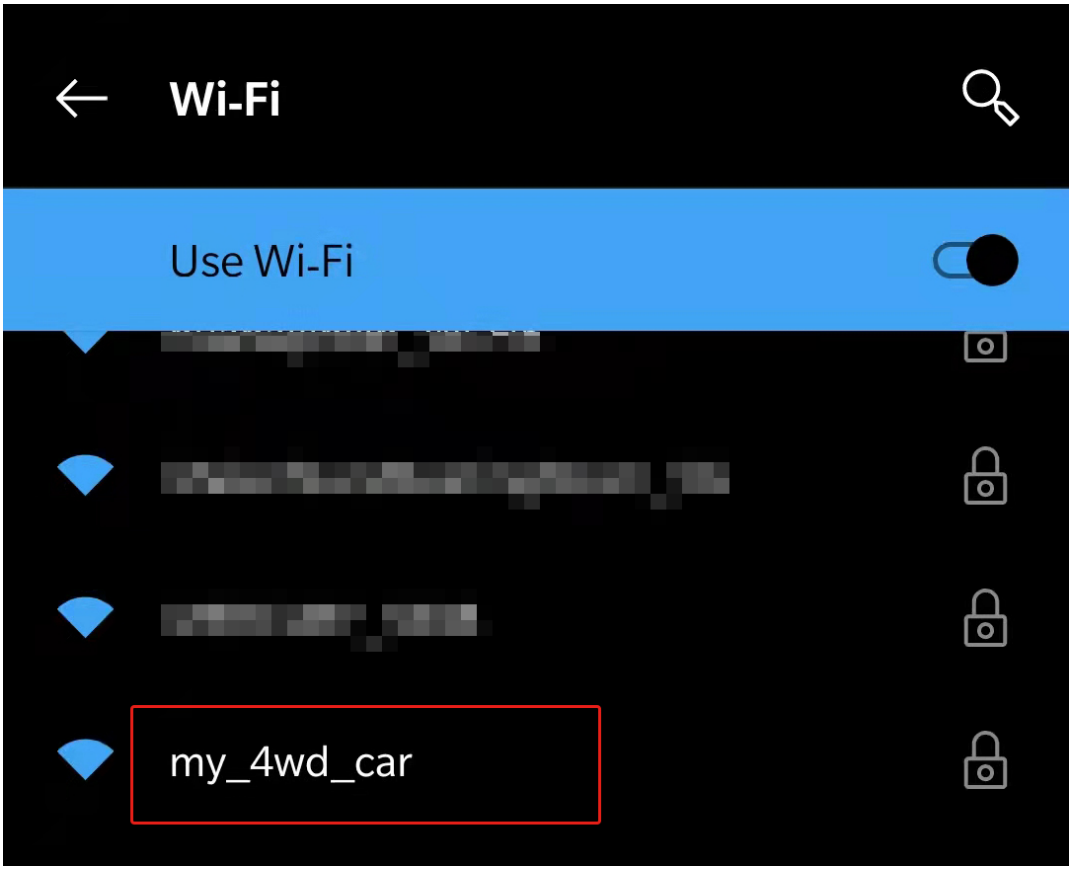
4. Set the file name to `main.py`. If you already have the same file in your Pico, it will prompt to overwrite it.



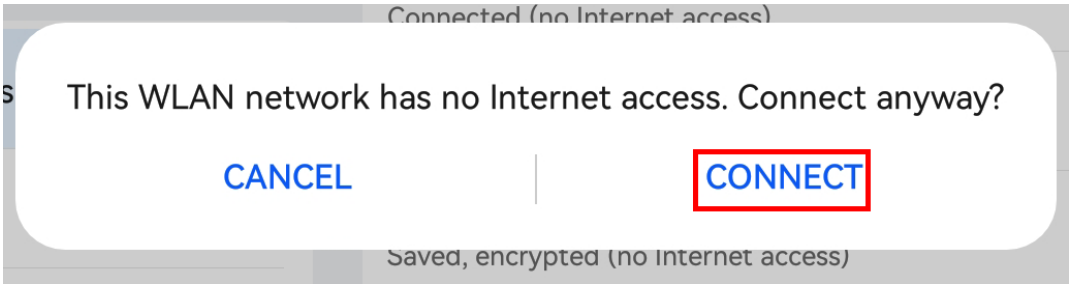
Now you can unplug the USB cable, turn on the power switch of Pico-4wd, and Pico-4wd will automatically run this `main.py` script.

3. Connect to Pico 4WD car

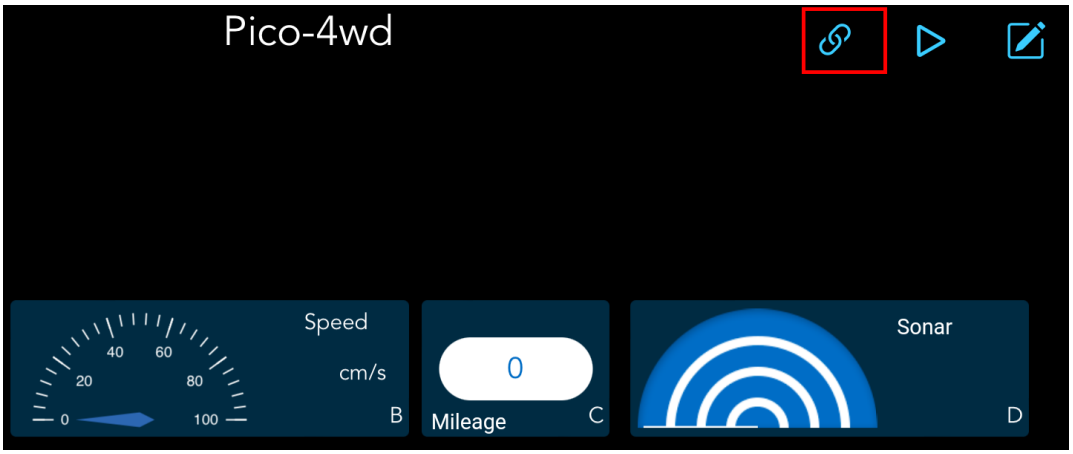
1. Find `my_4wd_car` on the WLAN of the mobile phone (tablet), enter the password `12345678` and connect to it.



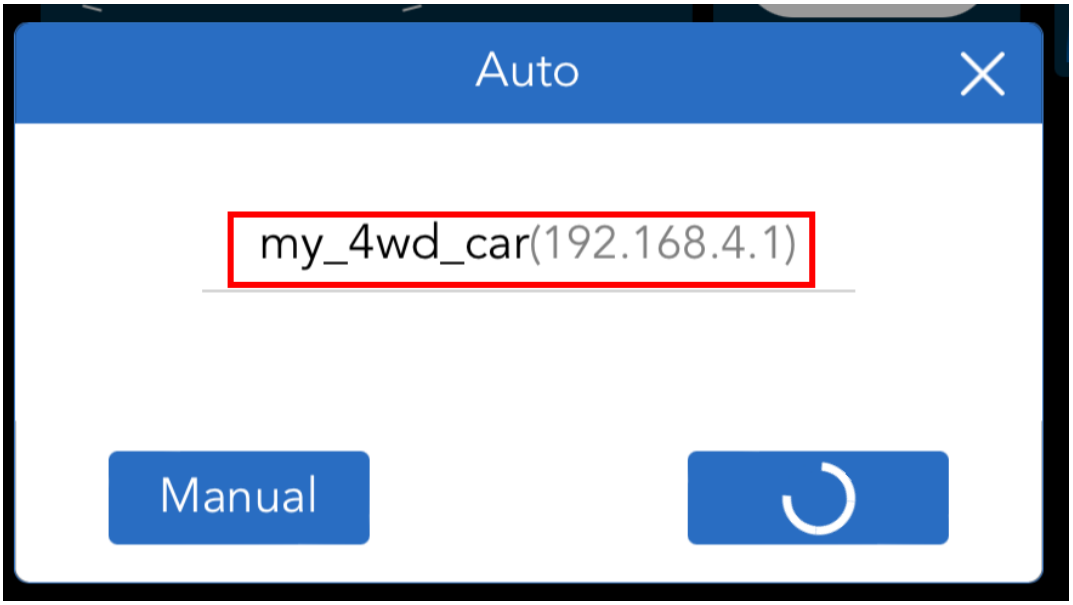
2. The default connection mode in `app_control.py` is . So after you connect, there will be a prompt telling you that there is no Internet access on this WLAN network, please choose to continue connecting.



3. Now go back to SunFounder Controller, when you click the **Connect** button, it will automatically search for robots nearby.

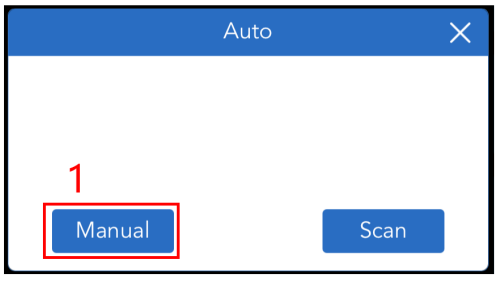


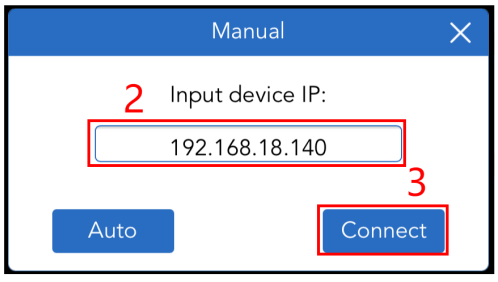
4. Click on the `my_4wd_car`.



Note:

- You need to make sure that your mobile device is connected to the `my_4wd_car` LAN.
- If it doesn't search automatically, you can also manually enter your car's IP to connect.



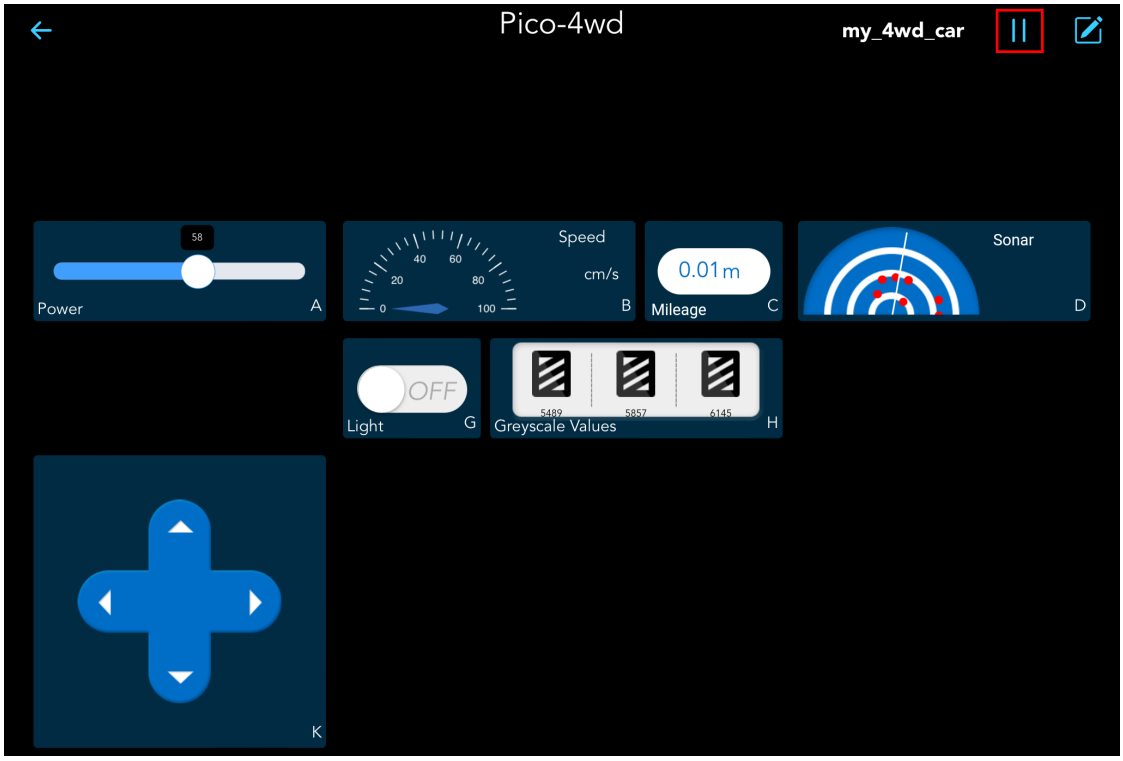


Once you click on `my_4wd_car`, the message "Connected Successfully" will appear and the product name will

appear in the upper right corner.

4. Run this Controller

Click the **Run** button to start the controller.



Here are the functions of the widgets.

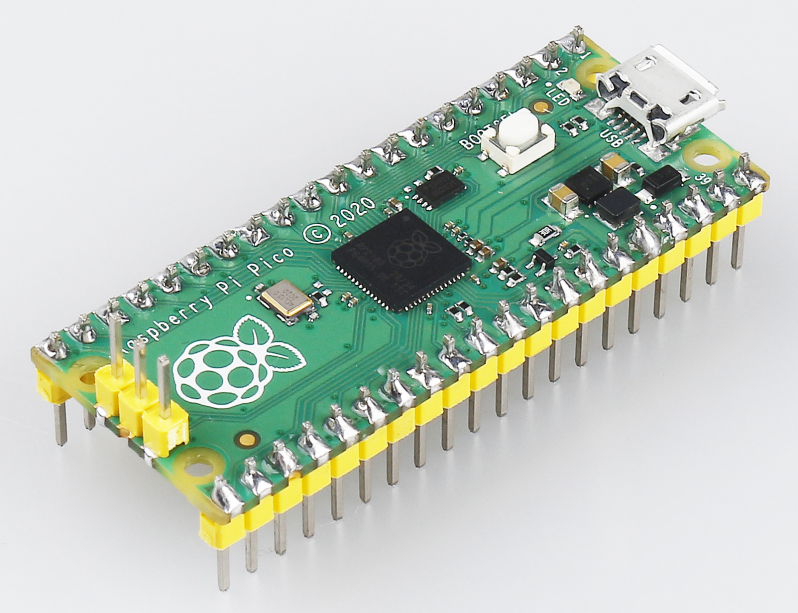
- A: Drive the car at 0~100% power. Before controlling the car movement with the K widget, you must set the A widget to 30% or more.
- B: The display of the car moving speed, unit: cm/s.
- C: Display of car speed in digital format.
- D: Radar display of obstacles detected by ultrasonic module.
- G: Turn on/off WS2812 RGB board.
- H: Show the data of the three sensors on the grayscale module, which have three states: black block: black line detected; white: white detected; exclamation point: cliff detected.
- K: Control forward, backward, left, and right motions of the car.

CHAPTER

TWO

APPENDIX

2.1 Introduction to Raspberry Pi Pico



The Raspberry Pi Pico is a microcontroller board based on the Raspberry Pi RP2040 microcontroller chip.

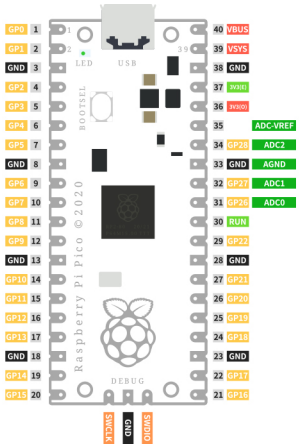
Whether you want to learn the MicroPython programming language, take the first step in physical computing, or want to build a hardware project, Raspberry Pi Pico – and its amazing community – will support you every step of the way. In the project, it can control anything, from LEDs and buttons to sensors, motors, and even other microcontrollers.

Features

- 21 mm × 51 mm form factor
- RP2040 microcontroller chip designed by Raspberry Pi in the UK
- Dual-core Arm Cortex-M0+ processor, flexible clock running up to 133 MHz
- 264KB on-chip SRAM
- 2MB on-board QSPI Flash
- 26 multifunction GPIO pins, including 3 analog inputs
- 2 × UART, 2 × SPI controllers, 2 × I2C controllers, 16 × PWM channels
- 1 × USB 1.1 controller and PHY, with host and device support

- 8 × Programmable I/O (PIO) state machines for custom peripheral support
- Supported input power 1.8–5.5V DC
- Operating temperature -20°C to +85°C
- Castellated module allows soldering direct to carrier boards
- Drag-and-drop programming using mass storage over USB
- Low-power sleep and dormant modes
- Accurate on-chip clock
- Temperature sensor
- Accelerated integer and floating-point libraries on-chip

Pico's Pins



Name	Description	Function
GP0-GP28	General-purpose input/output pins	Act as either input or output and have no fixed purpose of their own
GND	0 volts ground	Several GND pins around Pico to make wiring easier.
RUN	Enables or diables your Pico	Start and stop your Pico from another microcontroller.
GPxx_ADCx	General-purpose input/output or analog input	Used as an analog input as well as a digital input or output – but not both at the same time.
ADC_VREF	Analog-to-digital converter (ADC) voltage reference	A special input pin which sets a reference voltage for any analog inputs.
AGND	Analog-to-digital converter (ADC) 0 volts ground	A special ground connection for use with the ADC_VREF pin.
3V3(O)	3.3 volts power	A source of 3.3V power, the same voltage your Pico runs at internally, generated from the VSYS input.
3v3(E)	Enables or disables the power	Switch on or off the 3V3(O) power, can also switches your Pico off.
VSYS	2-5 volts power	A pin directly connected to your Pico's internal power supply, which cannot be switched off without also switching Pico off.
VBUS	5 volts power	A source of 5V power taken from your Pico's micro USB port, and used to power hardware which needs more than 3.3V.

The best place to find everything you need to get started with your Raspberry Pi Pico.

Or you can click on the links below:

- Raspberry Pi Pico product brief
- Raspberry Pi Pico datasheet
- Getting started with Raspberry Pi Pico: C/C++ development
- Raspberry Pi Pico C/C++ SDK
- API-level Doxygen documentation for the Raspberry Pi Pico C/C++ SDK
- Raspberry Pi Pico Python SDK
- Raspberry Pi RP2040 datasheet
- Hardware design with RP2040
- Raspberry Pi Pico design files
- Raspberry Pi Pico STEP file

2.2 Introduction to Pico RDP

The Pico Robotics Development Platform (RDP) is a Wi-Fi extension module for the Raspberry Pi Pico designed by SunFounder.

It integrates industry-leading Wi-Fi solutions, rich peripheral interfaces, and supports multiple compilers for development.

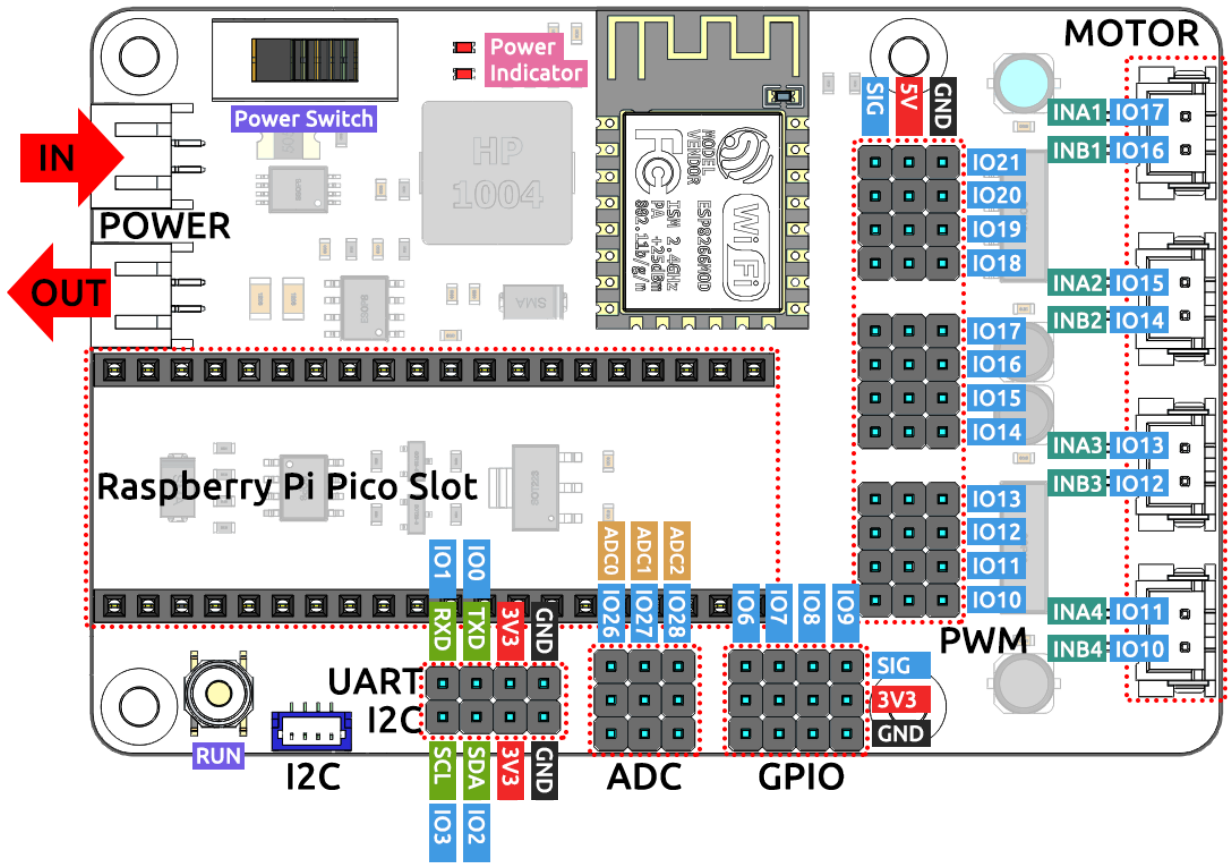
It also has IO expansion interface circuitry, LED power indicator circuitry, voltage measurement circuitry, and an on-board 4-channel DC motor driver circuit.

When you use Pico RDP for development and debugging, you can connect peripherals as needed, and the rich external interfaces can make your projects more interesting.

Features

- Microcontroller: Raspberry Pi Pico module
- Wi-Fi: ESP8266 Wi-Fi module, 802.11 b/g/n (802.11n, speeds up to 150 Mbps), 2.4 GHz ~ 2.5 GHz frequency range
- RUN button: reset button
- Input voltage: 7.0-30.0V (PH2.0-2P)
- Output voltage: 7.0-30.0V (PH2.0-2P), 5.0V, 3.3V
- Output current: 5V/5A, 3.3V/1A
- One channel SH1.0-4P port: I2C port.
- Four channel XH2.54-4P port: DC motor port
- 12 x PWM channel, 3 x ADC channel, 4 x GPIO pins.
- One channel SH1.0-4P port: I2C port. Compatible with QwIIC and STEMMA QT

Pico RDP's Pins



Here is the schematic of the Pico RDP: `PDF Pico RDP Schematic`.

2.3 Schematic and Structure Drawing

Schematic

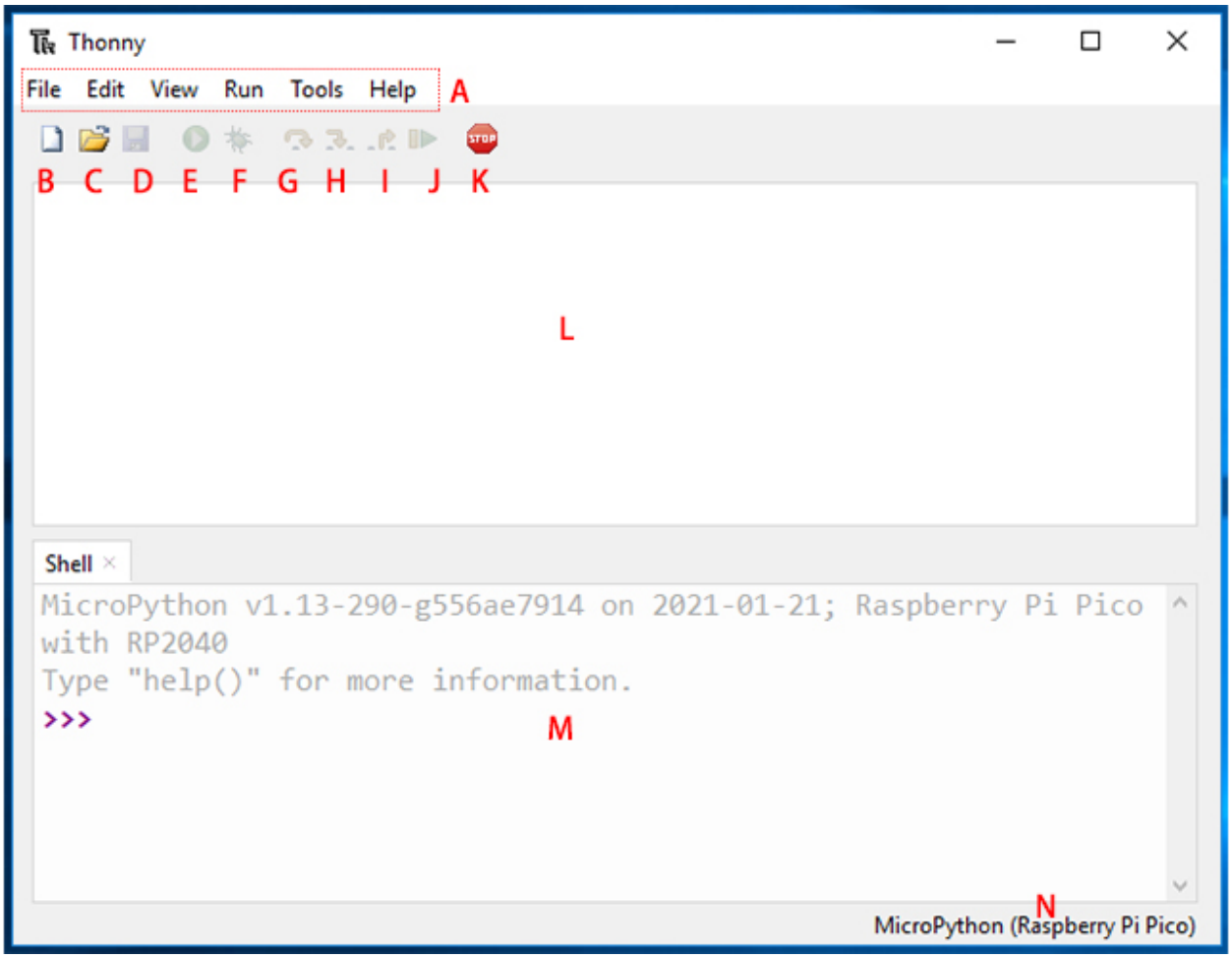
- `Grayscale Module`
- `RGB Board`
- `Pico RDP`
- `Speed Module`

Structure Drawing

- `Lower Plate`
- `Upper Plate`
- `The Whole Car`

2.4 Thonny IDE Introduction

- Ref:



- **A**: The menu bar that contains the file New, Save, Edit, View, Run, Debug, etc.
- **B**: This paper icon allows you to create a new file.
- **C**: The folder icon allows you to open files that already exist in your computer or Raspberry Pi Pico, if your Pico is already plugged into your computer.
- **D**: Click on the floppy disk icon to save the code. Similarly, you can choose whether to save the code to your computer or to the Raspberry Pi Pico.
- **E**: The play icon allows you to run the code. If you have not saved the code, save the code before it can run.
- **F**: The Debug icon allows you to debug your code. Inevitably, you will encounter errors when writing code. Errors can take many forms, sometimes using incorrect syntax, sometimes incorrect logic. Debugging is the tool for finding and investigating errors.

Note: The Debug tool cannot be used when MicroPython (Raspberry Pi Pico) is selected as the interpreter.

If you want to debug your code, you need to select the interpreter as the default interpreter and save as to your computer after debugging.

Finally, select the MicroPython (Raspberry Pi Pico) interpreter again, click the save as button, and re-save the debugged code to your Raspberry Pi Pico.

- The G, H and I arrow icons allow you to run the program step by step, but can only be started after clicking on the Degug icon. As you click on each arrow, you will notice that the yellow highlighted bar will indicate the line or section of Python that is currently evaluating.
  - **G**: Take a big step, which means jumping to the next line or block of code.
  - **H**: Take a small step means expressing each component in depth.
  - **I**: Exit out of the debugger.
- **J**: Click it to return from debug mode to play mode.
- **K**: Use the stop icon to stop running code.
- **L**: Script Area, where you can write your Python code.
- **M**: Python Shell, where you can type a single command, and when you press the Enter key, the single command will run and provide information about the running program. This is also known as REPL, which means "Read, Evaluate, Print, and Loop."
- **N**: Interpreter, where the current version of Python used to run your program is displayed, can be changed manually to another version by clicking on it.

Note: NO MicroPython(Raspberry Pi Pico) Interpreter Option ?

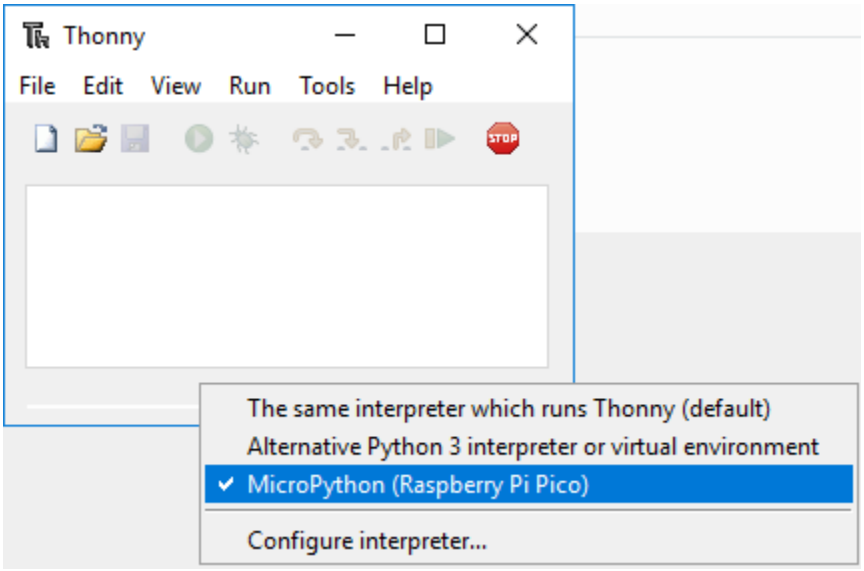
- Check that your Pico is plugged into your computer via a USB cable.
- The Raspberry Pi Pico interpreter is only available in version 3.3.3 or higher version of Thonny. If you are running an older version, please update.

CHAPTER

THREE

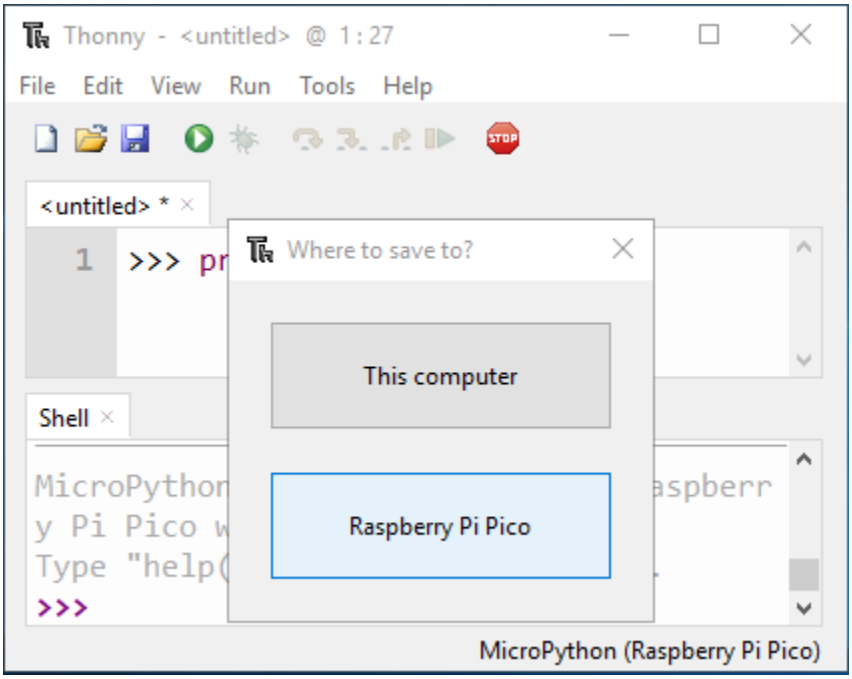
FAQ

3.1 Q1: NO MicroPython(Raspberry Pi Pico) Interpreter Option on Thonny IDE?



- Check that your Pico is plugged into your computer via a USB cable.
- Check that you have installed MicroPython for Pico (*2. Install MicroPython on Your Pico*).
- The Raspberry Pi Pico interpreter is only available in version 3.3.3 or higher version of Thonny. If you are running an older version, please update (*1. Install Thonny IDE*).
- Plug in/out the micro USB cable sveral times.

3.2 Q2: Cannot open Pico code or save code to Pico via Thonny IDE?



- Check that your Pico is plugged into your computer via a USB cable.
- Check that you have selected the Interpreter as **MicroPython (Raspberry Pi Pico)**.

CHAPTER

FOUR

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.

CHAPTER

FIVE

COPYRIGHT NOTICE