Welcome

Thank you for choosing Freenove products!

About Battery

First, read the document **About_Battery.pdf** in the unzipped folder.
If you did not download the zip file, please download it and unzip it via link below:
https://github.com/Freenove/Freenove_4WD_Car_Kit_for_Raspberry_Pi_Pico

Get Support and Offer Input

Freenove provides free and responsive product and technical support, including but not limited to:

- Product quality issues
- Product use and build issues
- Questions regarding the technology employed in our products for learning and education
- Your input and opinions are always welcome
- We also encourage your ideas and suggestions for new products and product improvements

For any of the above, you may send us an email to:

support@freenove.com

Safety and Precautions

Please follow the following safety precautions when using or storing this product:

- Keep this product out of the reach of children under 6 years old.
- This product should be used only when there is adult supervision present as young children lack necessary judgment regarding safety and the consequences of product misuse.
- This product contains small parts and parts, which are sharp. This product contains electrically conductive parts. Use caution with electrically conductive parts near or around power supplies, batteries and powered (live) circuits.
- When the product is turned ON, activated or tested, some parts will move or rotate. To avoid injuries to hands and fingers, keep them away from any moving parts!
- It is possible that an improperly connected or shorted circuit may cause overheating. Should this happen, immediately disconnect the power supply or remove the batteries and do not touch anything until it cools down! When everything is safe and cool, review the product tutorial to identify the cause.
- Only operate the product in accordance with the instructions and guidelines of this tutorial, otherwise parts may be damaged or you could be injured.
- Store the product in a cool dry place and avoid exposing the product to direct sunlight.
- After use, always turn the power OFF and remove or unplug the batteries before storing.

About Freenove

Freenove provides open source electronic products and services worldwide.

Freenove is committed to assist customers in their education of robotics, programming and electronic circuits so that they may transform their creative ideas into prototypes and new and innovative products. To this end, our services include but are not limited to:

- Educational and Entertaining Project Kits for Robots, Smart Cars and Drones
- Educational Kits to Learn Robotic Software Systems for Arduino, Raspberry Pi, micro: bit and Raspberry Pi Pico W.
- Electronic Component Assortments, Electronic Modules and Specialized Tools
- **Product Development and Customization Services**

You can find more about Freenove and get our latest news and updates through our website:

http://www.freenove.com

Copyright

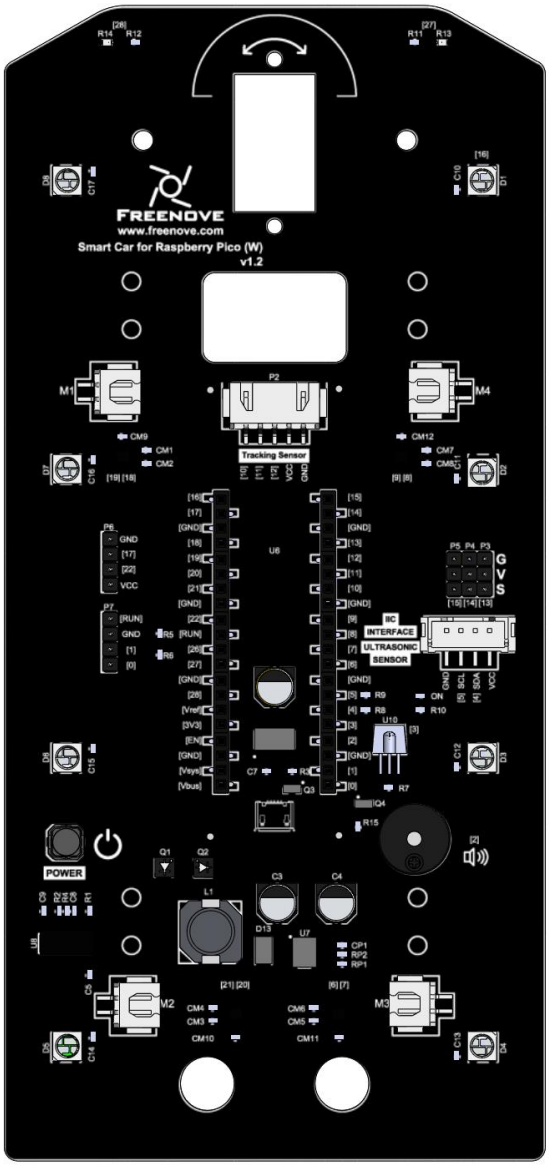
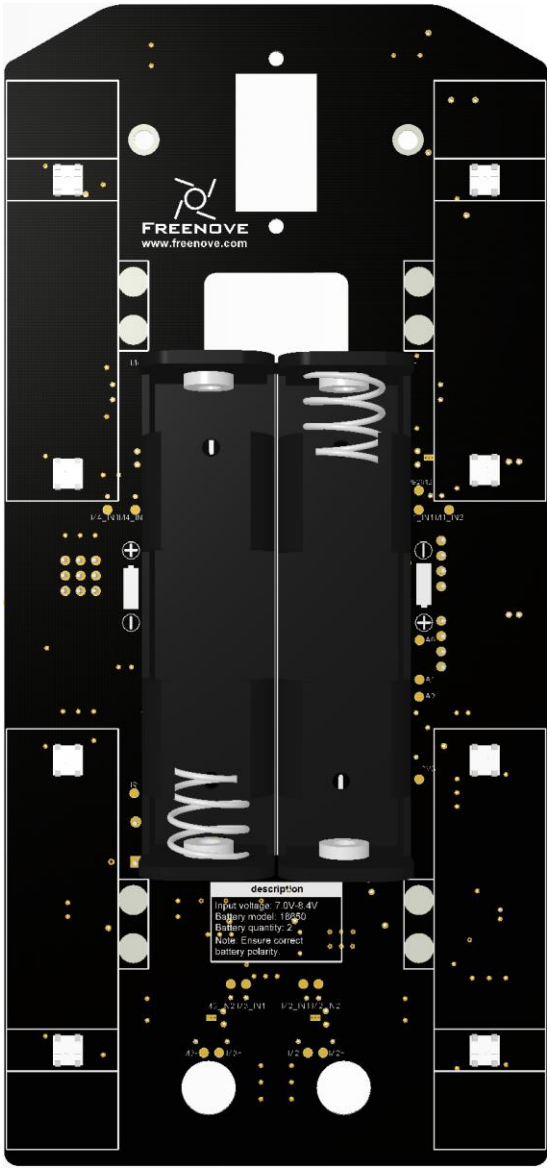
FREENOVE™

Contents

List

If you have any concerns, please feel free to contact us via support@freenove.com

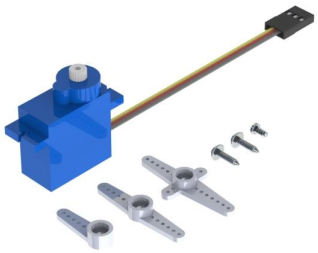
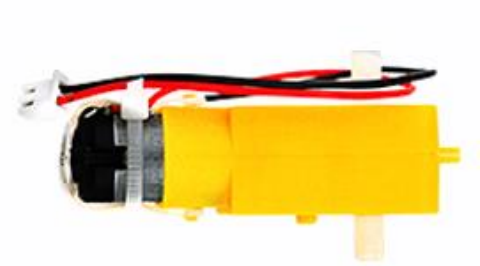
Raspberry Pi Pico (W) Car Shield

Top	Bottom
	

Machinery Parts

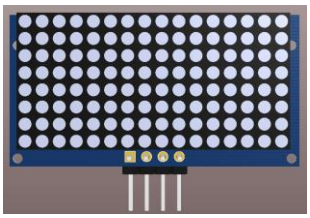
Transmission Parts

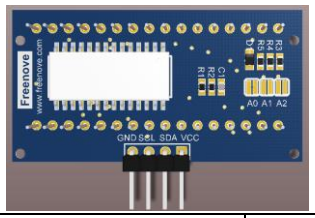
Servo package x1 	Driven wheel x4 
DC speed reduction motor x4  Caution: Do not remove the cable tie from the motor; otherwise, the motor cable may become detached.	Motor bracket package x4 

Electronic Parts

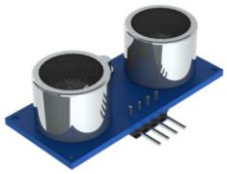
Line tracking module x1

Dot Matrix Module x1

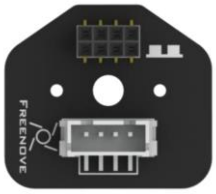




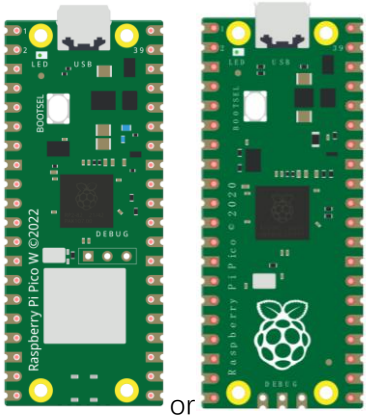
Ultrasonic Module x1



Ultrasonic module connector



Raspberry Pi Pico W x1 or Raspberry Pi Pico x1



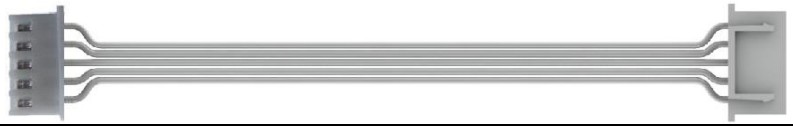
or

Infrared emitter x1

Wires

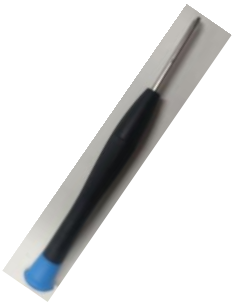
XH-2.54-4Pin cable x1

XH-2.54-5Pin cable x1



Tools

Cross screwdriver (3mm) x1



Black tape x1

Required but NOT Contained Parts

2 x 3.7V 18650 lithium **rechargeable** batteries with continuous discharge current >3A.

It is easier to find proper battery on eBay than Amazon. Search "18650 high drain" on eBay.

Preface

Welcome to use Freenove 4WD Car Kit for Raspberry Pi Pico (W). Following this tutorial, you can make a very cool car with many functions.

Based on the Raspberry Pi Pico (W) development board, a popular IoT control board, this kit uses the very popular Arduino IDE for programming, so you can share and exchange your experience and design ideas with enthusiasts all over the world. The parts in the kit include all electronic components, modules, and mechanical components required for making the car. They are all individually packaged. There are detailed assembly and debugging instructions in this book. If you encounter any problems, please feel free to contact us for fast and free technical support.

support@freenove.com

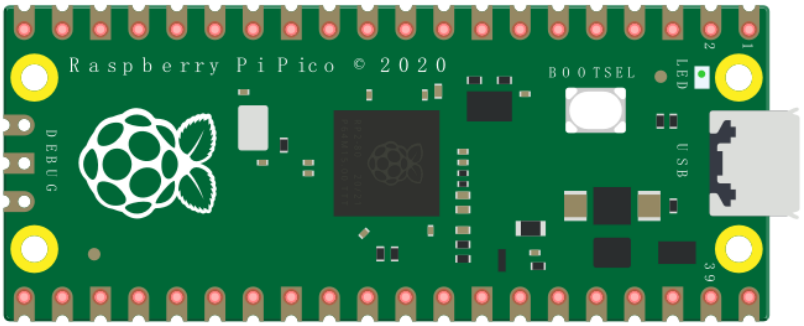
This car does not require a high threshold for users. Even if you know little professional knowledge, you can make your own smart car easily with the guidance of the tutorial. If you're really interested in Raspberry Pi Pico (W) and hope to learn how to program and build circuits, please visit our website: www.freenove.com

or contact us to buy our kit designed for beginners: **Freenove Ultimate Kit for Raspberry Pi Pico.**

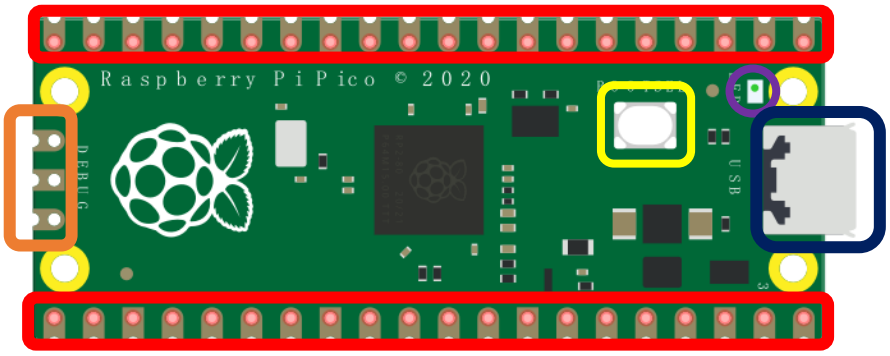
Raspberry Pi Pico

Raspberry Pi Pico applies to all chapters except Wireless in this tutorial.

Before learning Pico, we need to know about it. Here we use an imitated diagram of Pico, which ressembles the actual Pico.

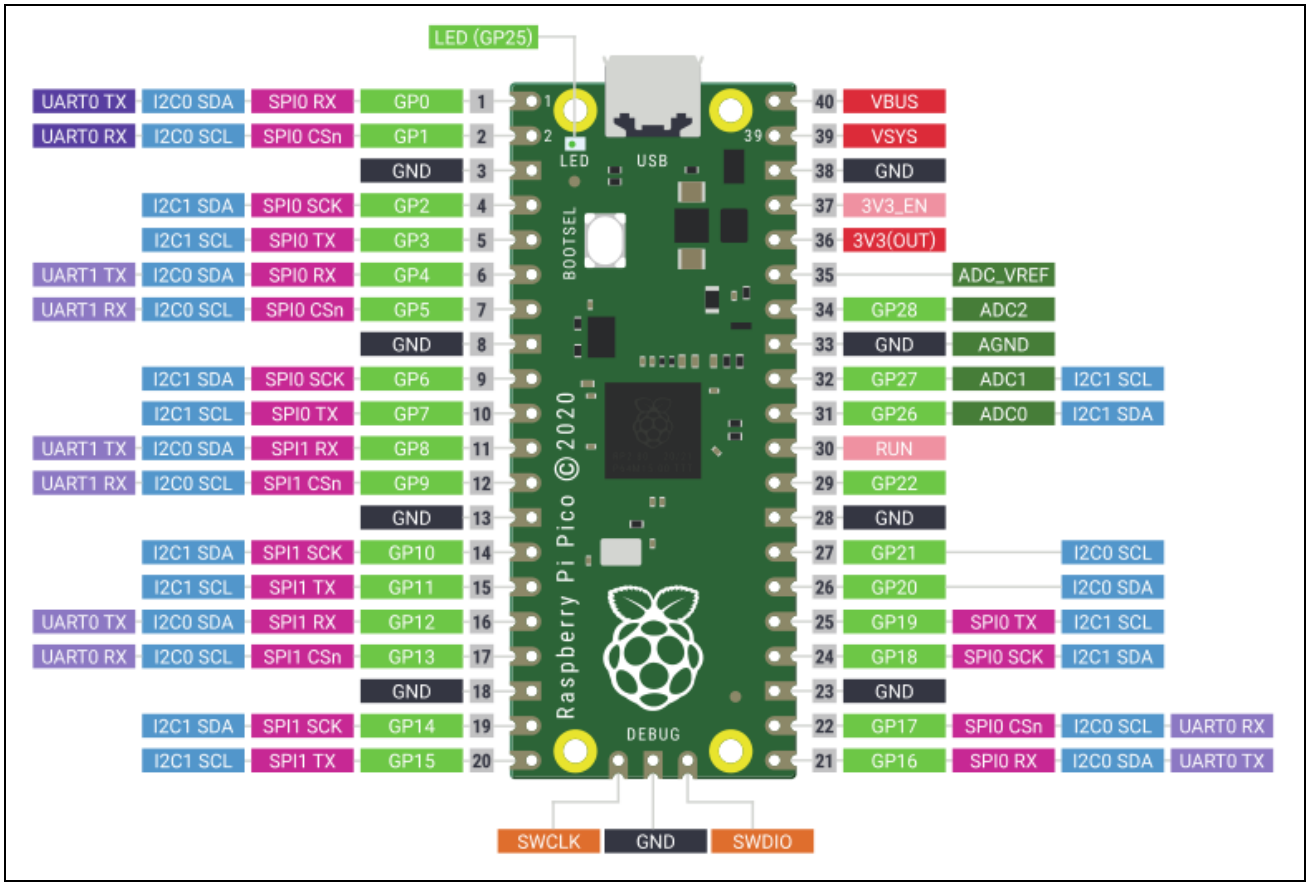


The hardware interfaces are distributed as follows:



Frame color	Description
	Pins
	BOOTSEL button
	USB port
	LED
	Debugging

Function definition of pins:



Color	Pins	Color	Pins
	GND		Power
	GPIO		ADC
	UART(defualt)		UART
	SPI		I2C
	System Control		Debugging

For details: https://datasheets.raspberrypi.org/pico/pico-datasheet.pdf

UART, I2C, SPI Default Pin

In Arduino IDE, the default pins of serial port are Pin0 and Pin1.
Note: Serial port is virtualized by RP2040. Therefore, when using the serial port, please enable the verification function of DTR. It can work under any baud rate.

UART

Function	Default
UART_BAUDRATE	X
UART_BITS	8
UART_STOP	1
UART_TX	Pin 0
UART_RX	Pin 1

I2C

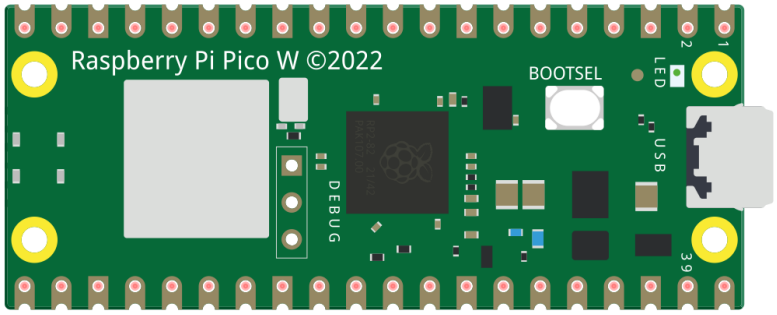
Function	Default
I2C Frequency	400000
I2C_SDA	Pin 4
I2C_SCL	Pin 5

SPI

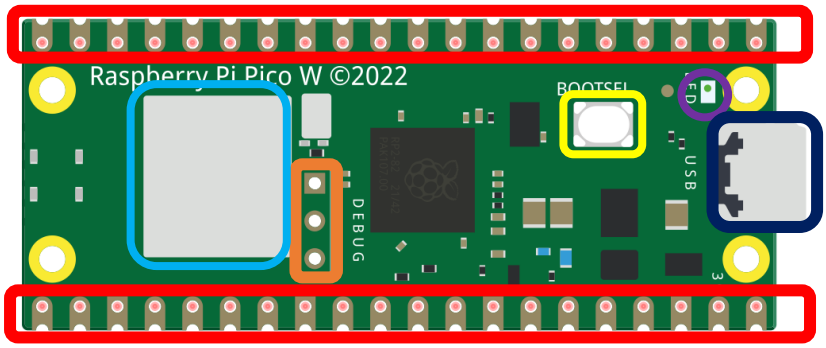
Function	Default
SPI_BAUDRATE	1000000
SPI_POLARITY	0
SPI_PHASE	0
SPI_BITS	8
SPI_FIRSTBIT	MSB
SPI_SCK	Pin 18
SPI_MOSI	Pin 19
SPI_MISO	Pin 16
SPI_SS	Pin 17

Raspberry Pi Pico W

Raspberry Pi Pico W adds CYW43439 as the WiFi function on the basis of Raspberry Pi Pico. It is connected to RP2040 chip through SPI interface.

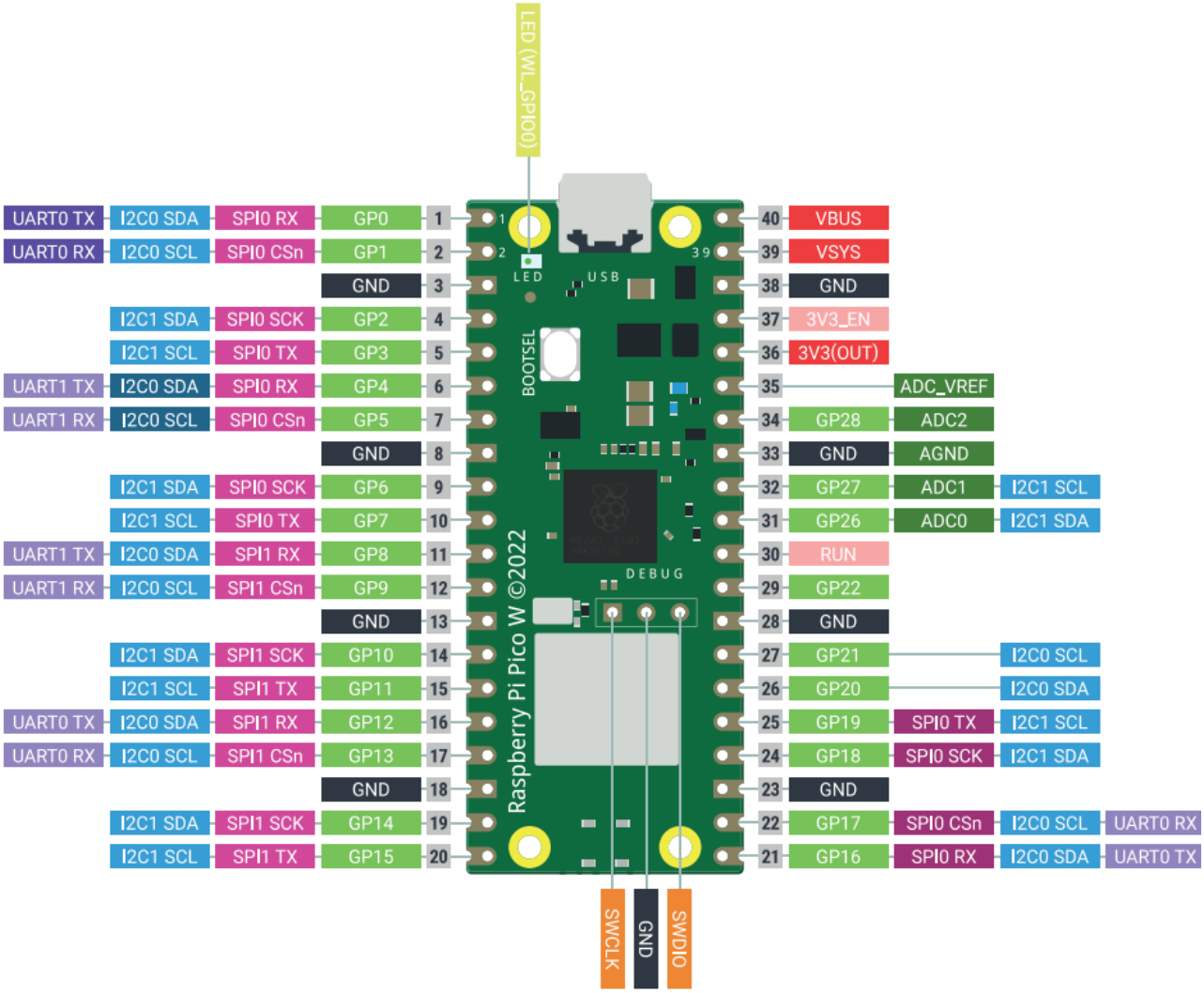


The hardware interfaces are distributed as follows:



Frame color	Description
	Pins
	BOOTSEL button
	USB port
	LED
	Debugging
	Wireless

Function definition of pins:



Color	Pins	Color	Pins
	GND		Power
	GPIO		ADC
	UART(defualt)		UART
	SPI		I2C
	System Control		Debugging

For details: https://datasheets.raspberrypi.com/picow/pico-w-datasheet.pdf

UART, I2C, SPI, Wireless Default Pin

In Arduino IDE, the default pins of serial port are Pin0 and Pin1.
Note: Serial port is virtualized by RP2040. Therefore, when using the serial port, please enable the verification function of DTR. It can work under any baud rate.

UART

Function	Default
UART_BAUDRATE	X
UART_BITS	8
UART_STOP	1
UART_TX	Pin 0
UART_RX	Pin 1

I2C

Function	Default
I2C Frequency	400000
I2C_SDA	Pin 4
I2C_SCL	Pin 5

SPI

Function	Default
SPI_BAUDRATE	1000000
SPI_POLARITY	0
SPI_PHASE	0
SPI_BITS	8
SPI_FIRSTBIT	MSB
SPI_SCK	Pin 18
SPI_MOSI	Pin 19
SPI_MISO	Pin 16
SPI_SS	Pin 17

Wireless

Function	Default
WL_ON	GPIO23
WL_D	GPIO24
WL_CLK	GPIO29_ADC
WL_CS	GPIO25

Pins of the Car

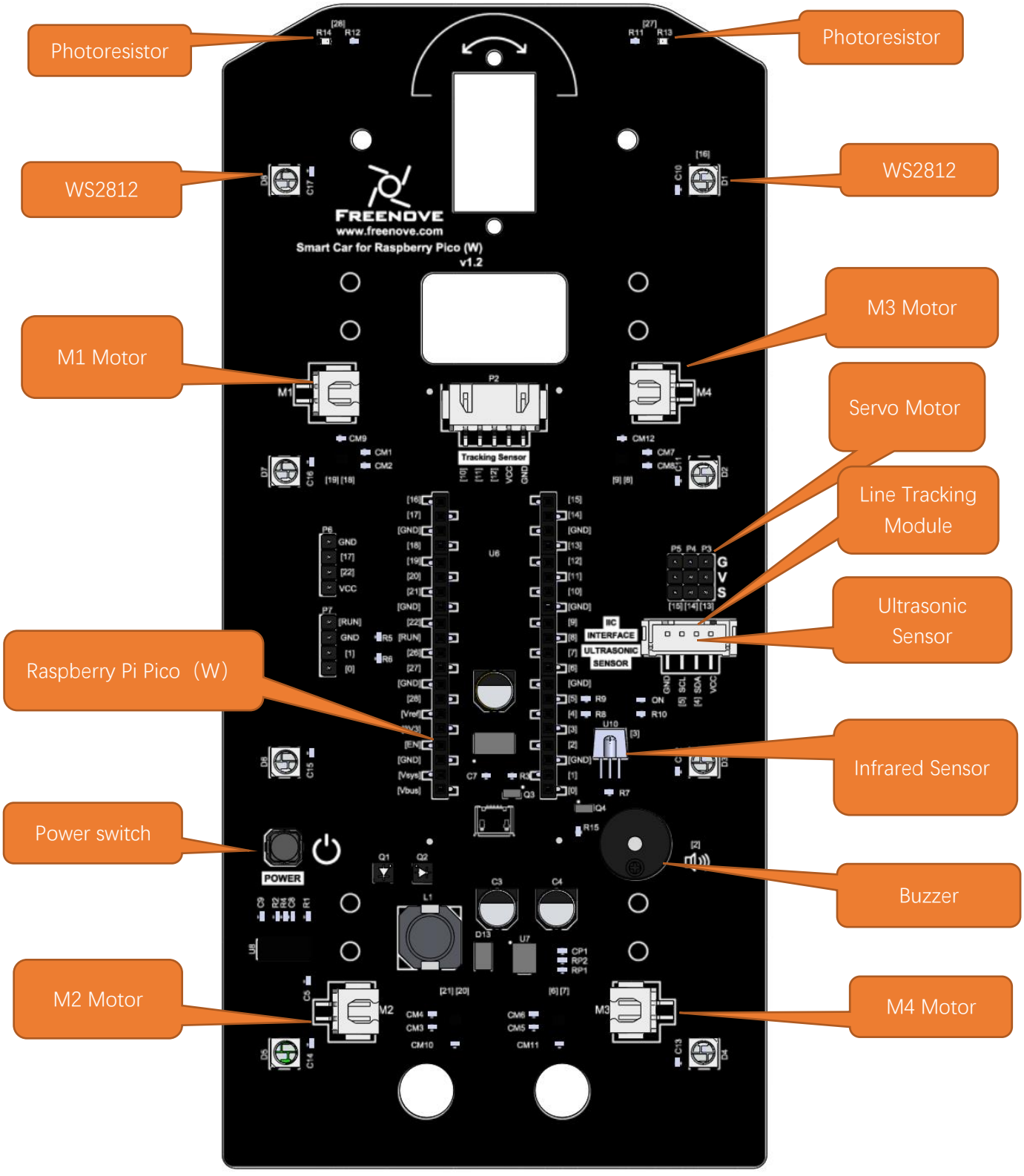
To learn what each GPIO corresponds to, please refer to the following table.

The functions of the pins are allocated as follows:

Pins of Raspberry Pi Pico W	Funtions	Description
GPIO18	Motor	M1_IN1
GPIO19		M1_IN2
GPIO20		M2_IN1
GPIO21		M2_IN2
GPIO6		M3_IN1
GPIO7		M3_IN2
GPIO8		M4_IN1
GPIO9		M4_IN2
GPIO13	Servo	Servo1
GPIO14		Servo2
GPIO15		Servo3
GPIO10	Tracking module	Track1
GPIO11		Track2
GPIO12		Track3
GPIO4	I2C port/Ultrasonic module interface	SDA/Trig
GPIO5		SCL/Echo
GPIO16	WS2812	WS2812
GPIO26	Battery detection	A0
GPIO27	Search light ADC port	A1
GPIO28	Search light ADC port	A2
GPIO22	Unused GPIO	GPIO22
GPIO17		GPIO17
GPIO3	Infrared receiver port	IR
GPIO2	Buzzer port	Buzzer
GPIO1	Serial port	RX
GPIO0		TX

Introduction to the Car

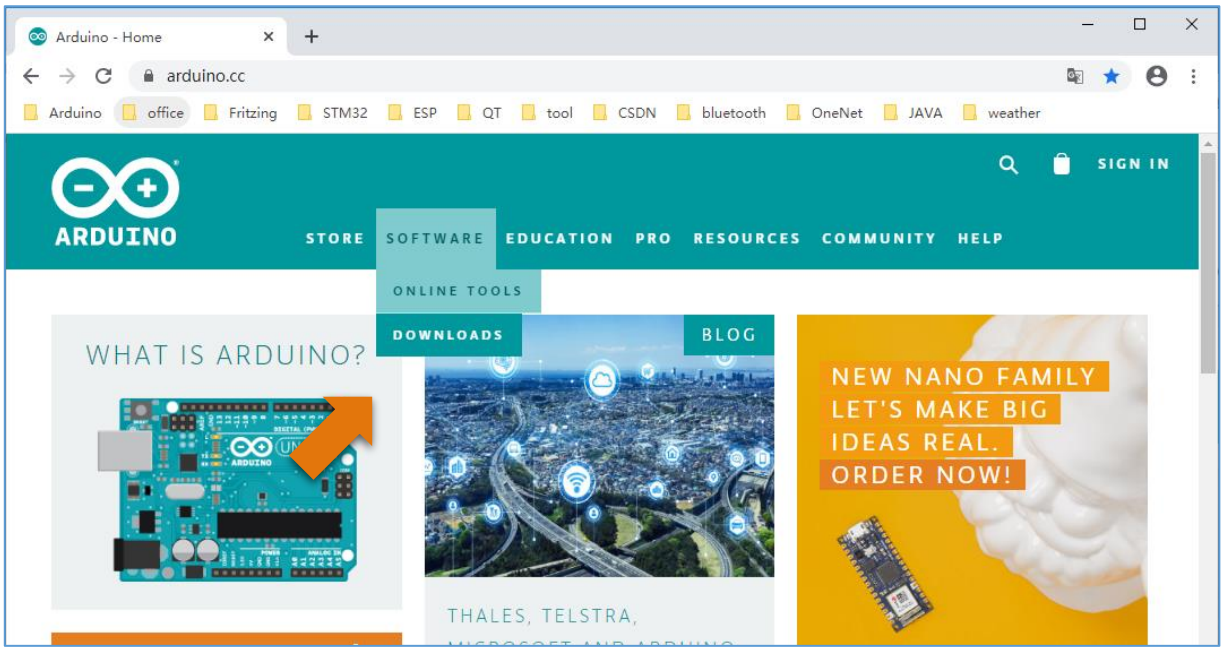
The function diagram of the Raspberry Pi Pico W car is as follows:



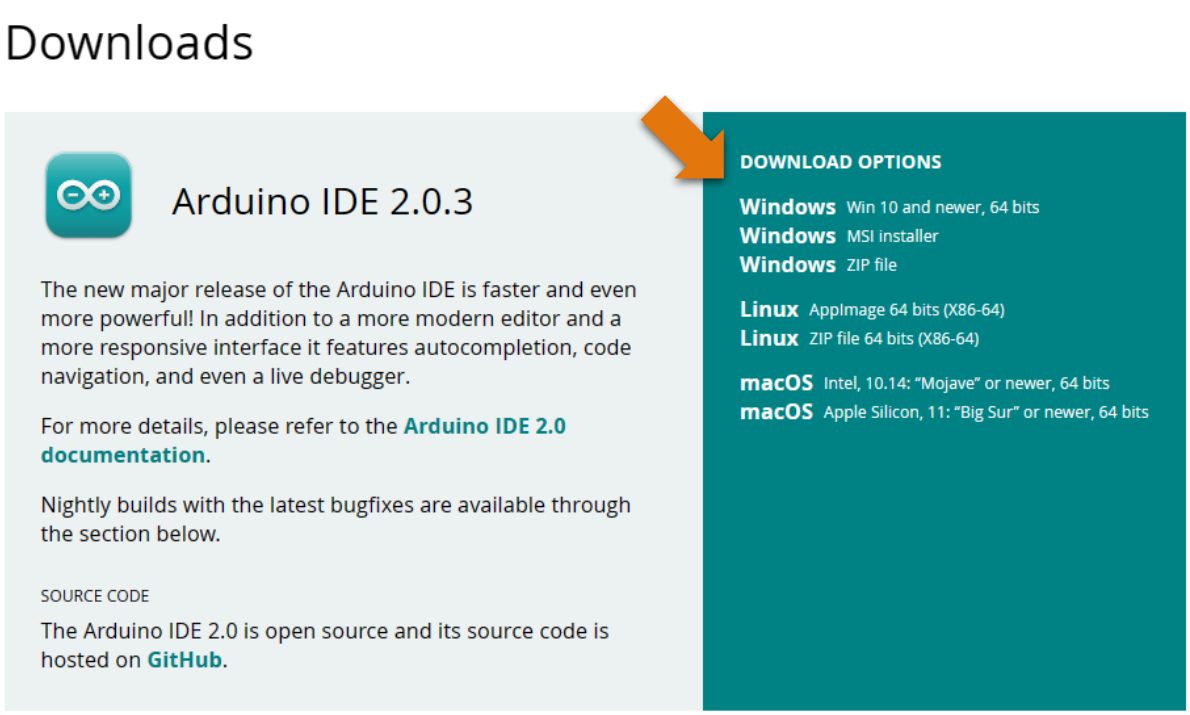
Chapter 0 Installation of Arduino IDE

Arduino Software

Arduino Software (IDE) is used to write and upload the code for Arduino Board.
First, install Arduino Software (IDE): visit https://www.arduino.cc, click "Download" to enter the download page.

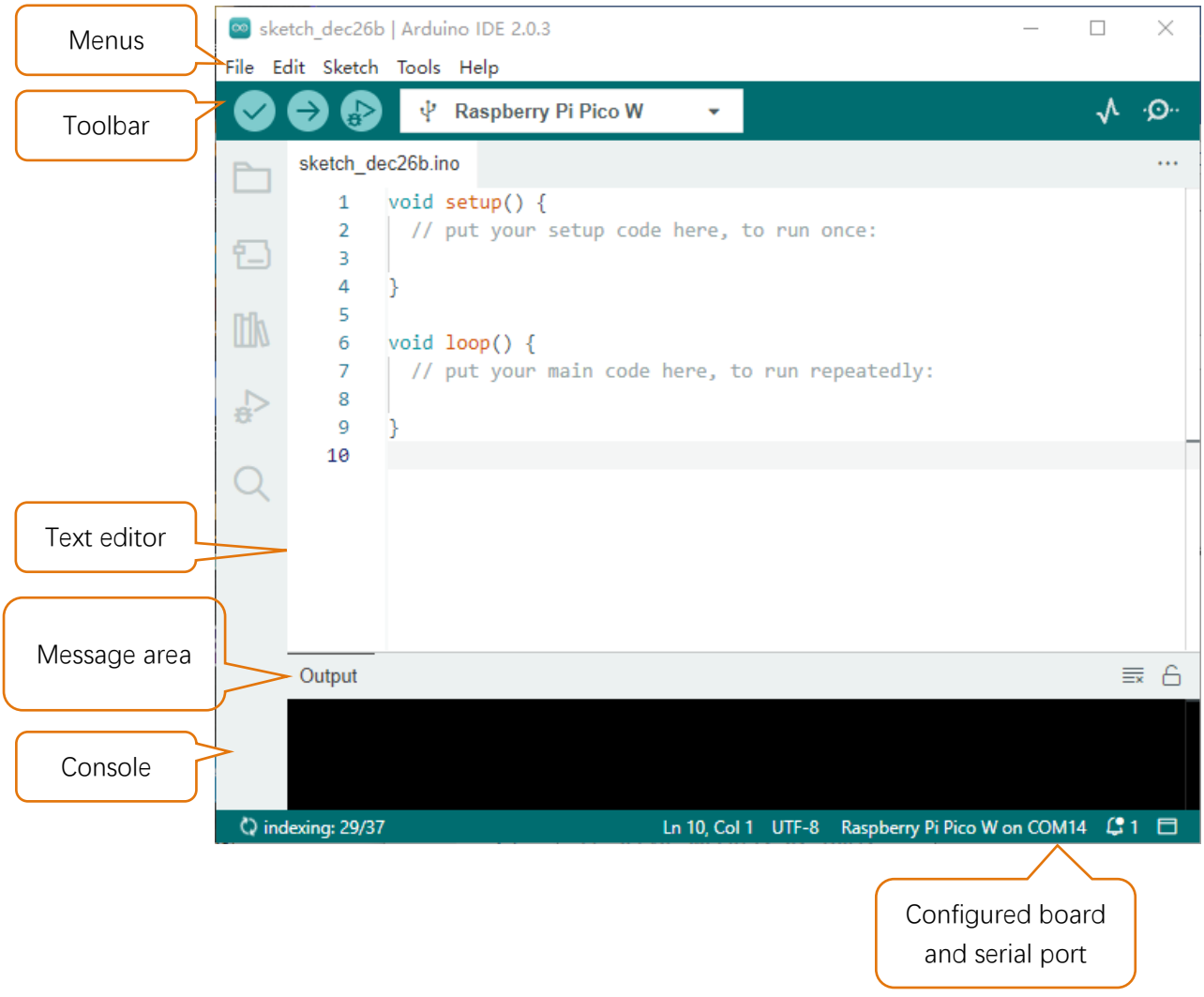


Select and download corresponding installer based on your operating system. If you are a Windows user, please select the "Windows Installer" to download and install the driver correctly.



After the download completes, run the installer. For Windows users, there may pop up an installation dialog during the installation. When it popes up, please allow the installation.
After installation is completed, an Arduino Software shortcut will be generated on the desktop. Run the Arduino Software.

The interface of Arduino Software is as follows:



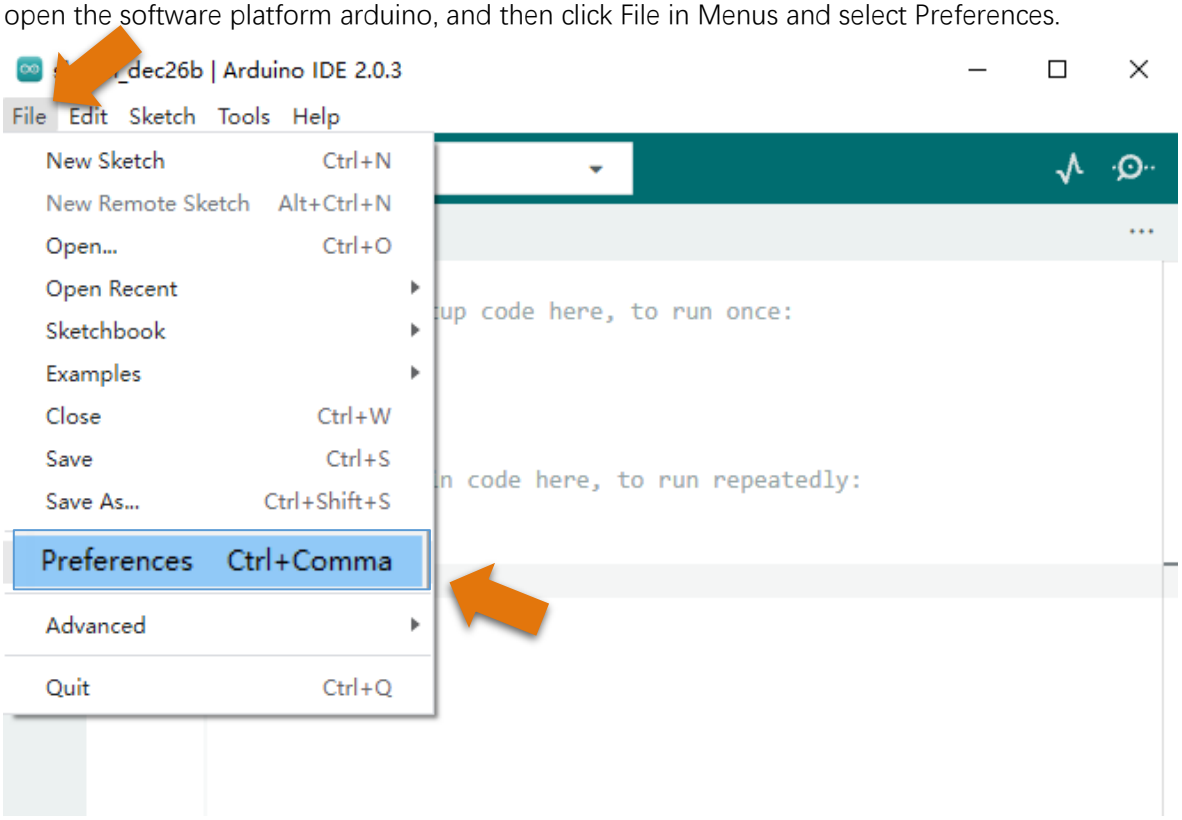
Programs written with Arduino Software (IDE) are called **sketches**. These sketches are written in the text editor and saved with the file extension**.ino**. The editor has features for cutting/pasting and searching/replacing text. The message area gives feedback while saving and exporting and also displays errors. The console displays text output by the Arduino Software (IDE), including complete error messages and other information. The bottom right-hand corner of the window displays the configured board and serial port. The toolbar buttons allow you to verify and upload programs, create, open, and save sketches, and open the serial monitor.

	Verify	Check your code for compile errors .
	Upload	Compile your code and upload them to the configured board.
	New	Create a new sketch.
	Open	Present a menu of all the sketches in your sketchbook. Clicking one will open it within the current window and overwrite its content.
	Save	Save your sketch.
	Serial Monitor	Open the serial monitor.

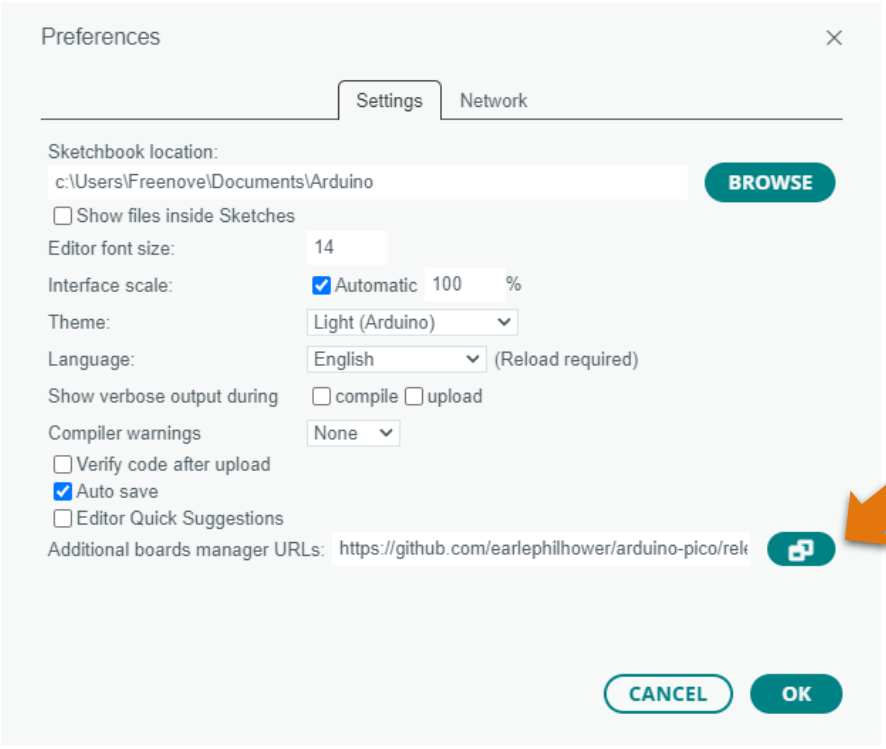
Additional commands are found within the five menus: File, Edit, Sketch, Tools, Help. The menus are context sensitive, which means only those items relevant to the work currently being carried out are available.

Environment Configuration

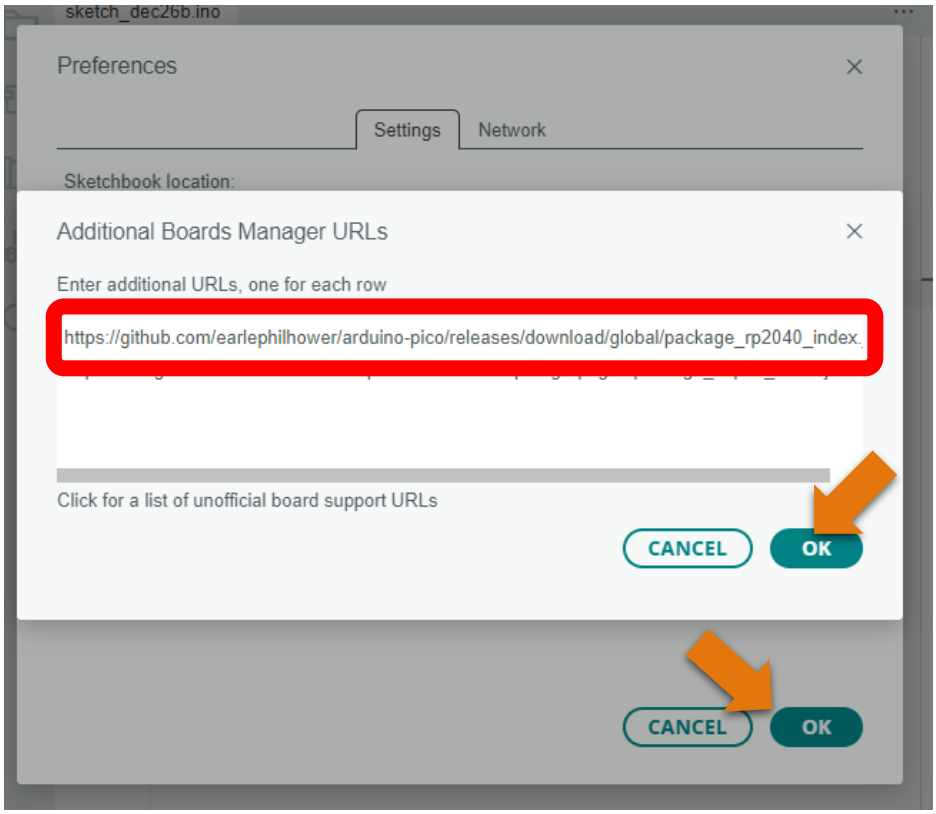
First, open the software platform arduino, and then click File in Menus and select Preferences.



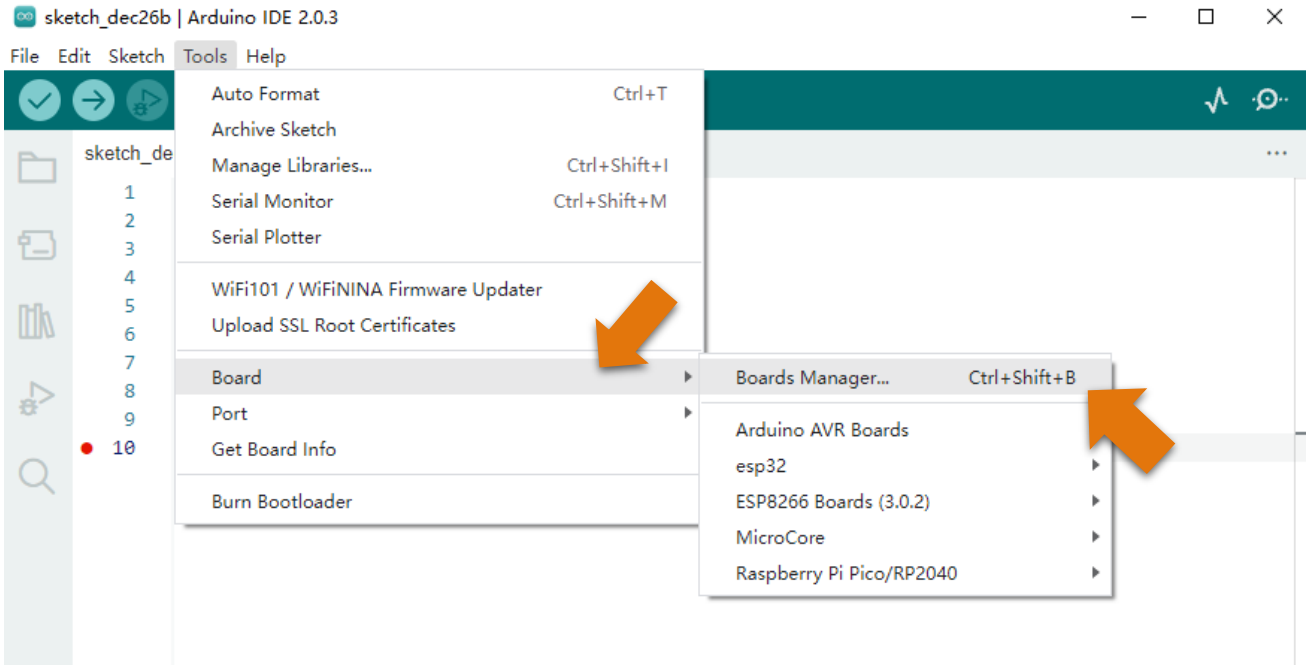
Second, click on the symbol behind "Additional Boards Manager URLs"



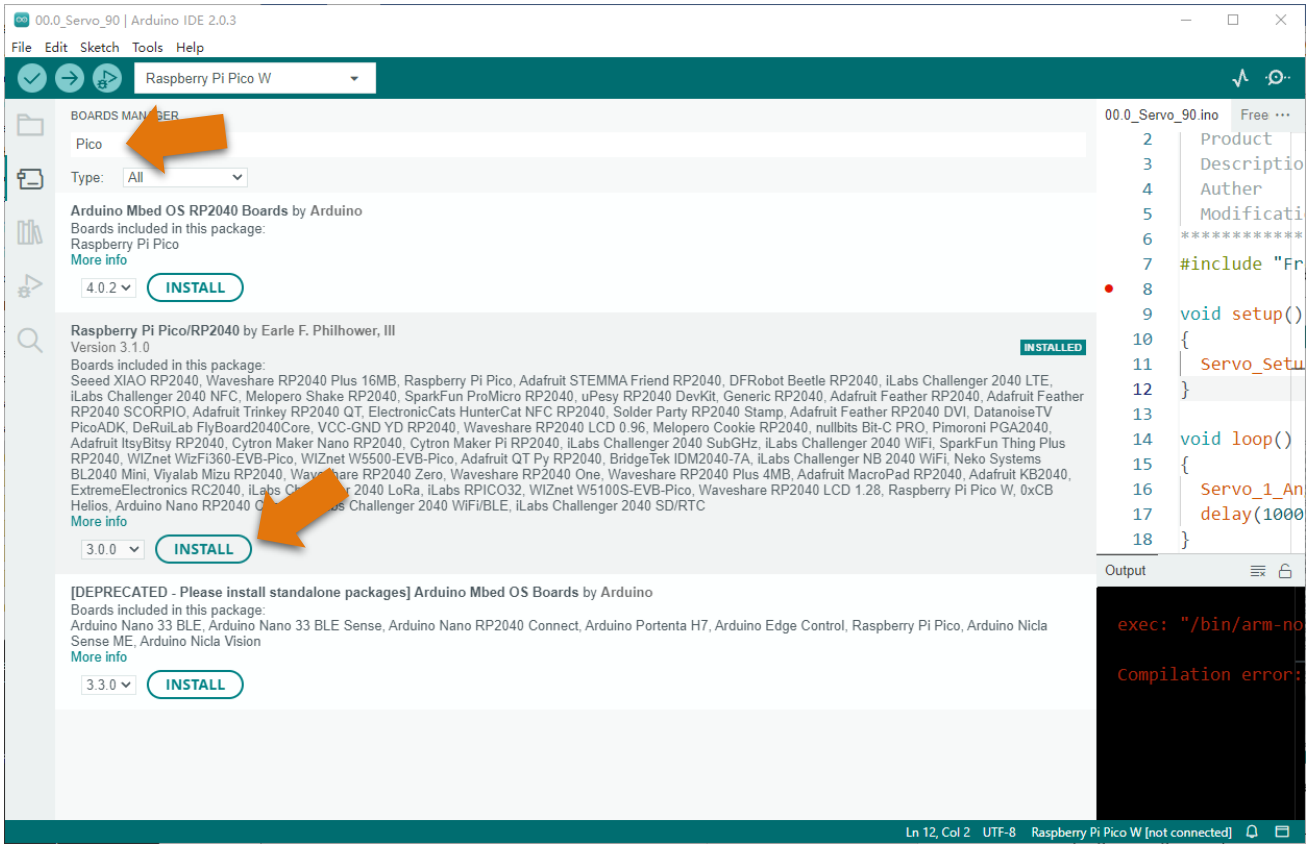
Third, fill in https://github.com/earlephilhower/arduino-pico/releases/download/global/package_rp2040_index.json in the new window, click OK, and click OK on the Preferences window again.



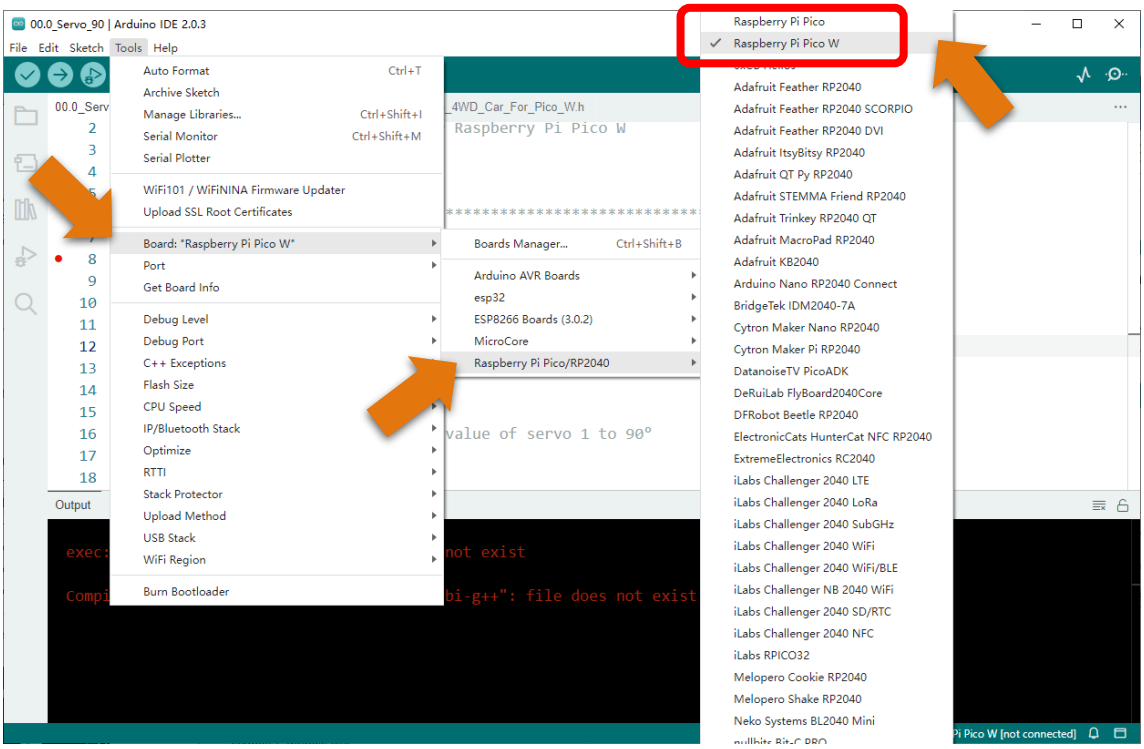
Fourth, click Tools in Menus, select Board:"ArduinoUno", and then select "Boards Manager".



Fifth, input "Pico" in the window below, and press Enter. click "Install" to install.



When finishing installation, click Tools in the Menus again and select Board: "Raspberry Pi Pico/RP2040", and then you can see information of Raspberry Pi Pico (W). Click "Raspberry Pi Pico W" so that the Raspberry Pi Pico W programming development environment is configured.

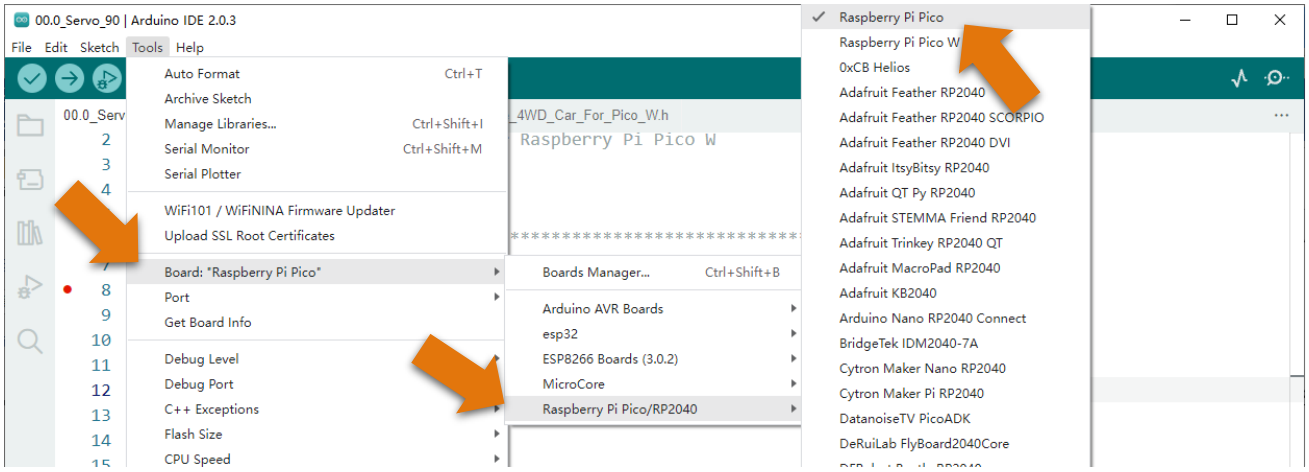


Additional Remarks

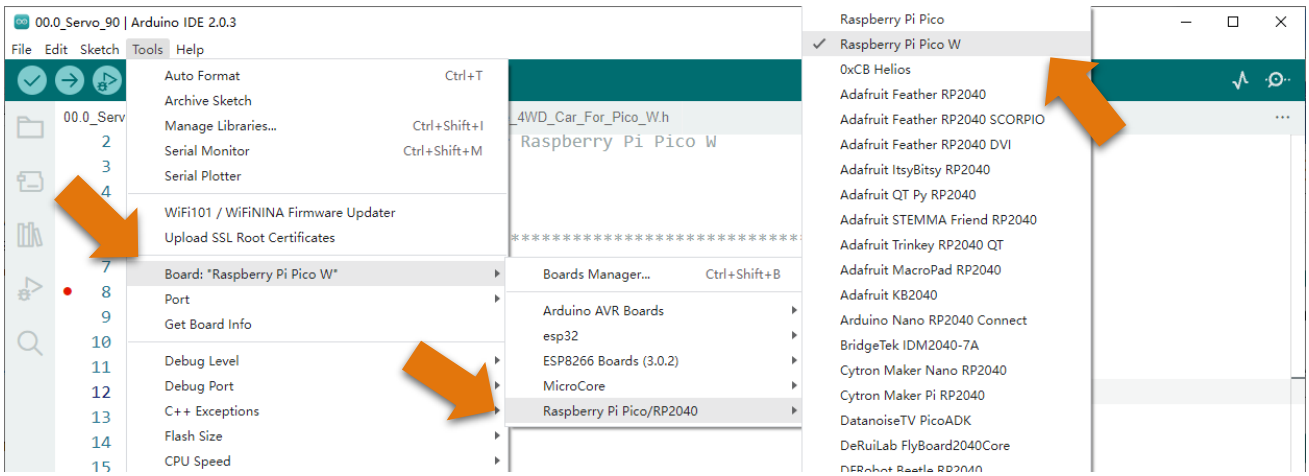
To finish this tutorial, you can use a Raspberry Pi Pico W or a Raspberry Pi Pico. The two boards have the same form factor, and their only difference is that Pico W features with an onboard single-band 4.802GHz wireless interface using Infineon CYW2. That is to say, the hardware Raspberry Pi Pico W is almost the same as the normal pico except for the wireless interface, which enables Pico W to work with WiFi (see chapter 7). If you use a normal Pico, you will not be able to finish the experiments in chapter 7. You can just skip that chapter.
No matter which board you use, the procedure for each project is the same. You only need to select the correct board and upload the corresponding code based on the board you use.

In this book, we use Raspberry Pi Pico W to illustrate the operation.

The followings show the configuration on Arduino for the use of each board.
For Raspberry Pi Pico:
Click Tools on Menu bar, click Board, select “Raspberry Pi Pico/RP2040”, and select Raspberry Pi Pico.



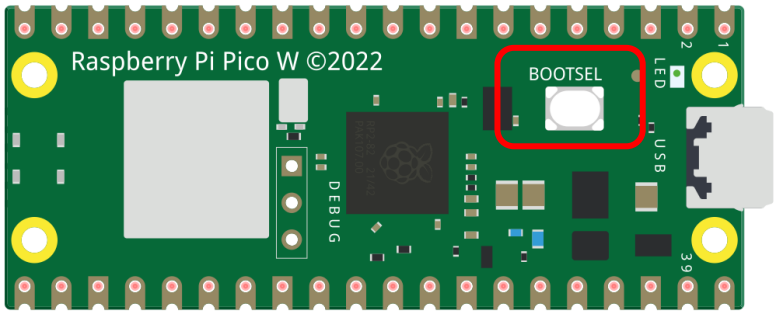
For Raspberry Pi Pico W:
Click Tools on Menu bar, click Board, select “Raspberry Pi Pico/RP2040”, and select Raspberry Pi Pico W.



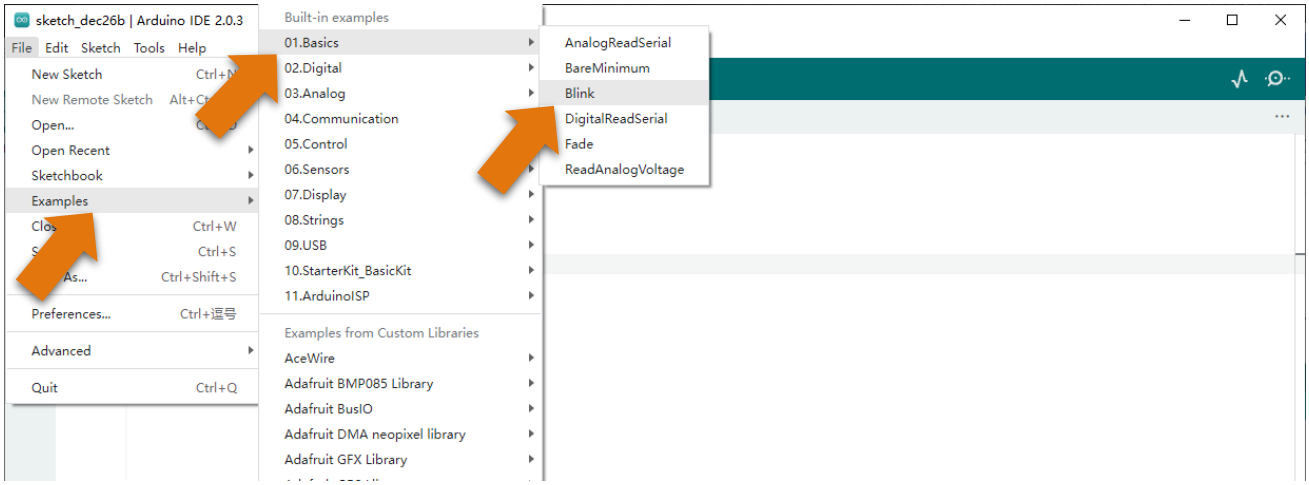
Uploading Adruino-compatible Firmware for Raspberry Pi Pico (W)

To program a new Raspberry Pi Pico (W) with Arduino, you need to upload an Adruino-compatible Firmware for it. Please refer to the following steps to cinfigure.

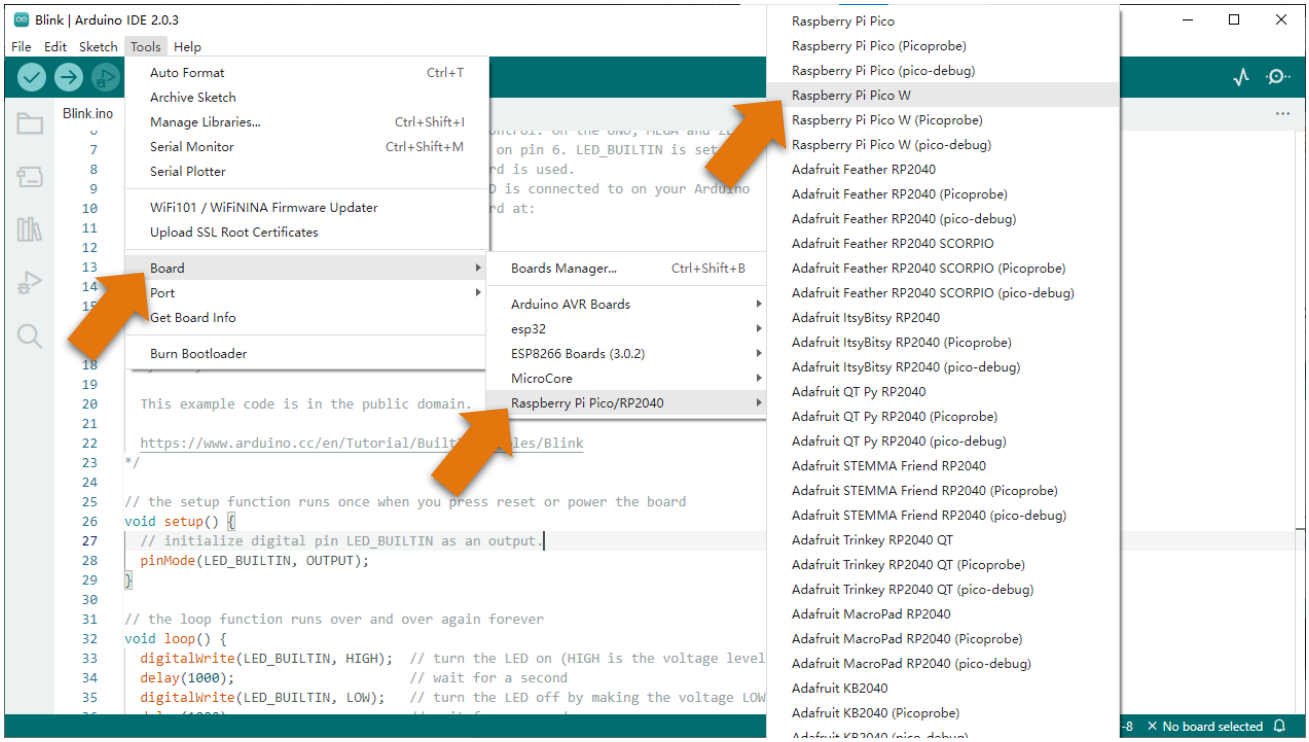
1. Disconnect Pico (W) from your computer. Press and hold the white button (BOOTSEL) on Pico (W) while connecting it to your computer. (Note: Be sure to hold the button before powering the Pico, otherwise the firmware will not download successfully.)



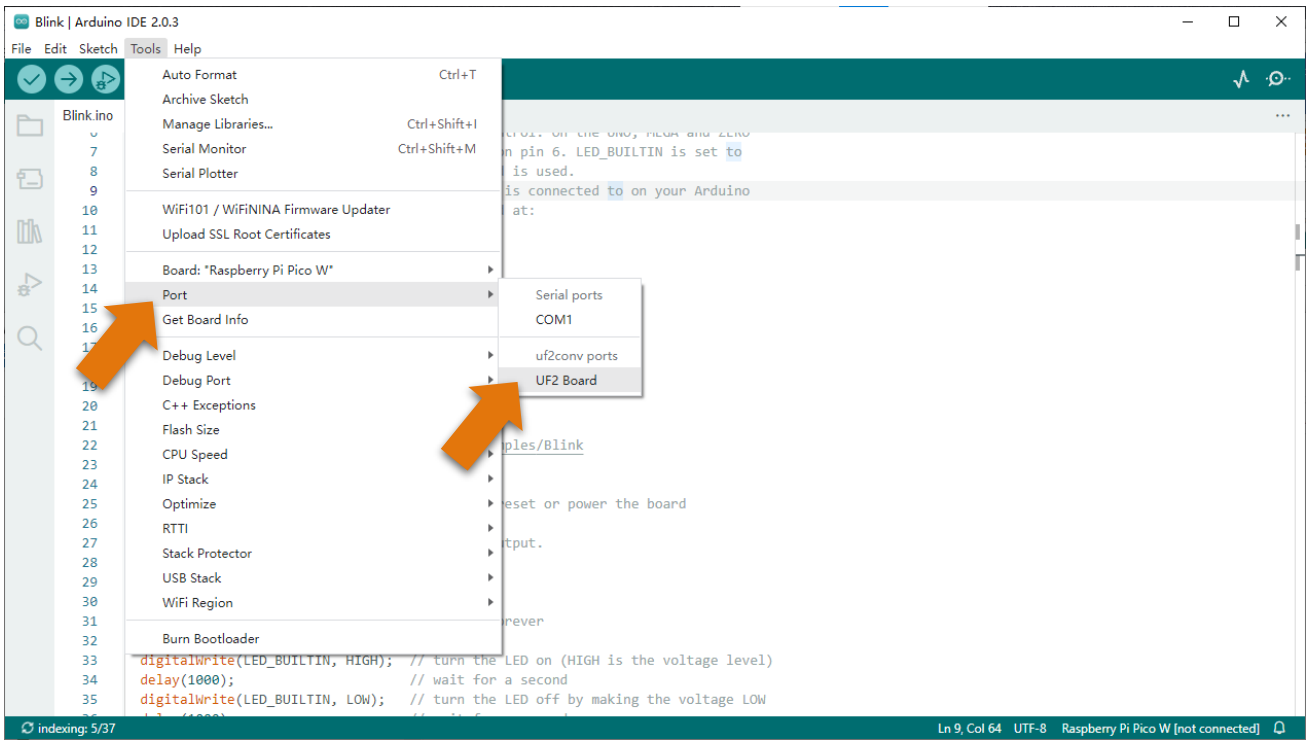
2. Open Arduino IDE. Click File>Examples>01.Basics>Blink.



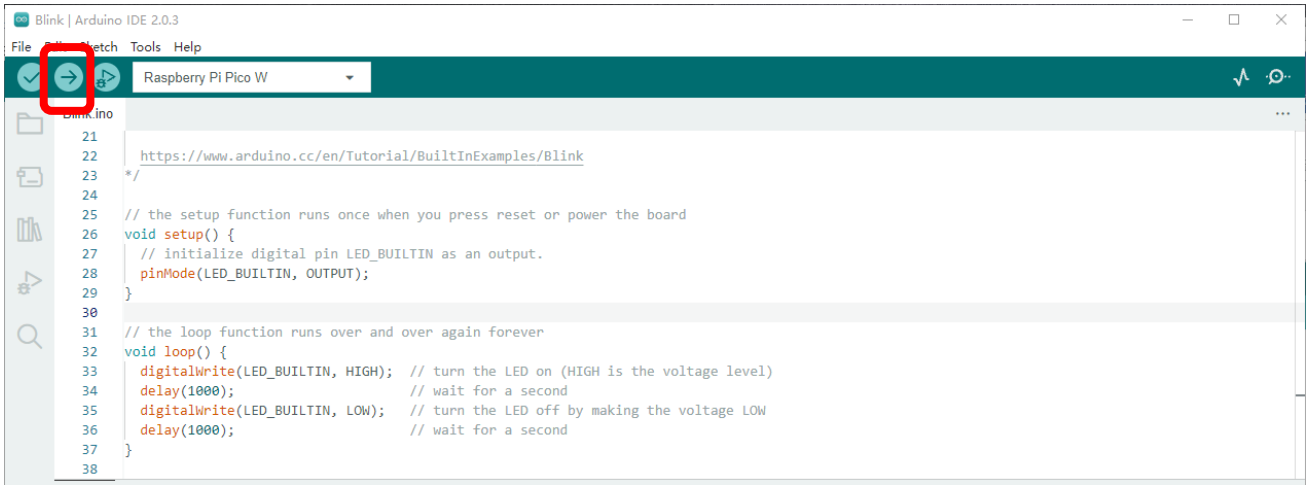
3. Click **Tools>Board>Raspberry Pi RP2040 Boards>Raspberry Pi Pico W**.



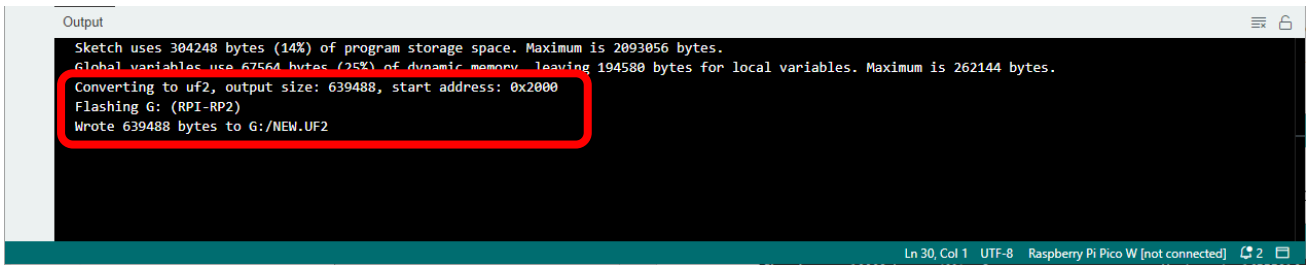
4. Click **Tools>Port>UF2 Board**.



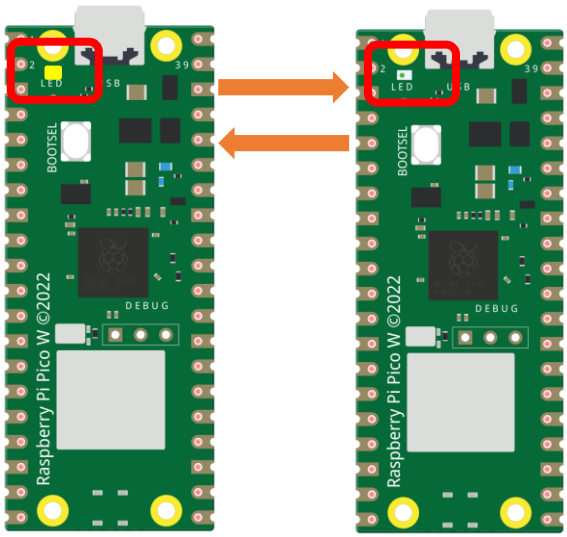
5. Upload sketch to Raspberry Pi Pico W.



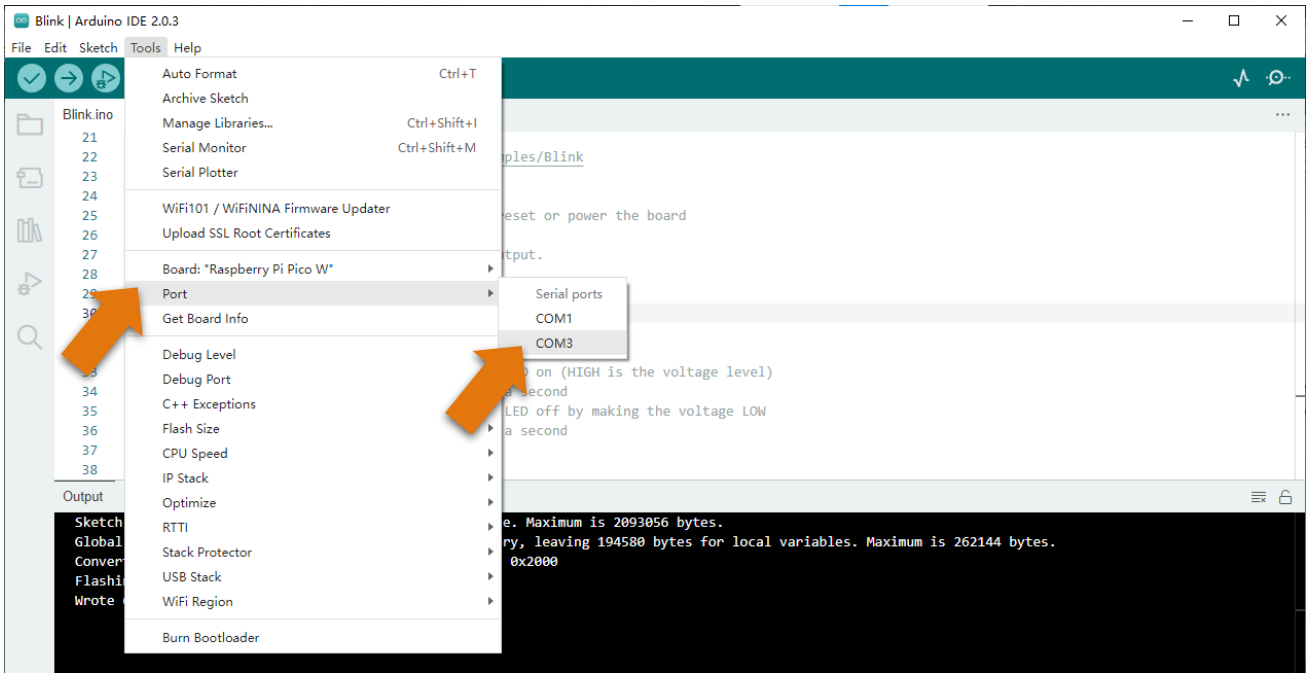
When the sketch finishes uploading, you can see messages as below.



The indicator on Pico W starts to flash.



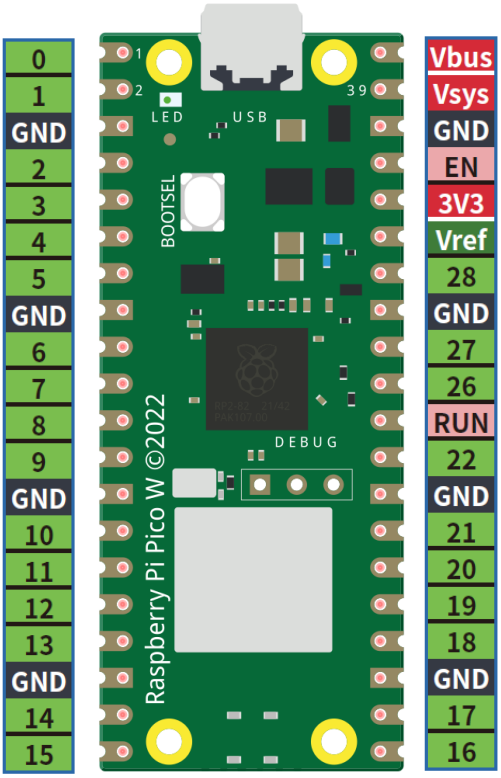
5, Click **Tools>Port>COMx (Raspberry Pi Pico W)**. X of COMx varies from different computers. Please select the correct one on your computer. In our case, it is COM3.



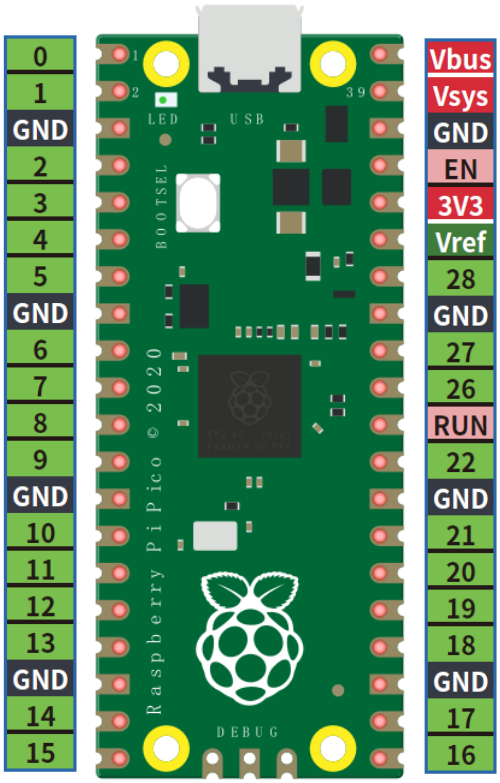
Note:

1. At the first use of Arduino to upload sketch for Pico (W), you need to select a port on the UF2 board. After that, each time before uploading sketch, please check whether the port has beed selected; otherwise, the downloading may fail.
2. Sometimes when used, Pico (W) may lose firmware due to the code and fail to work. At this point, you can upload firmware for Pico (W) as mentioned above.

To facilitate your learning, the pins of Pico W and Pico are shown below:



Raspberry Pi Pico W



Raspberry Pi Pico

Uploading the First Code

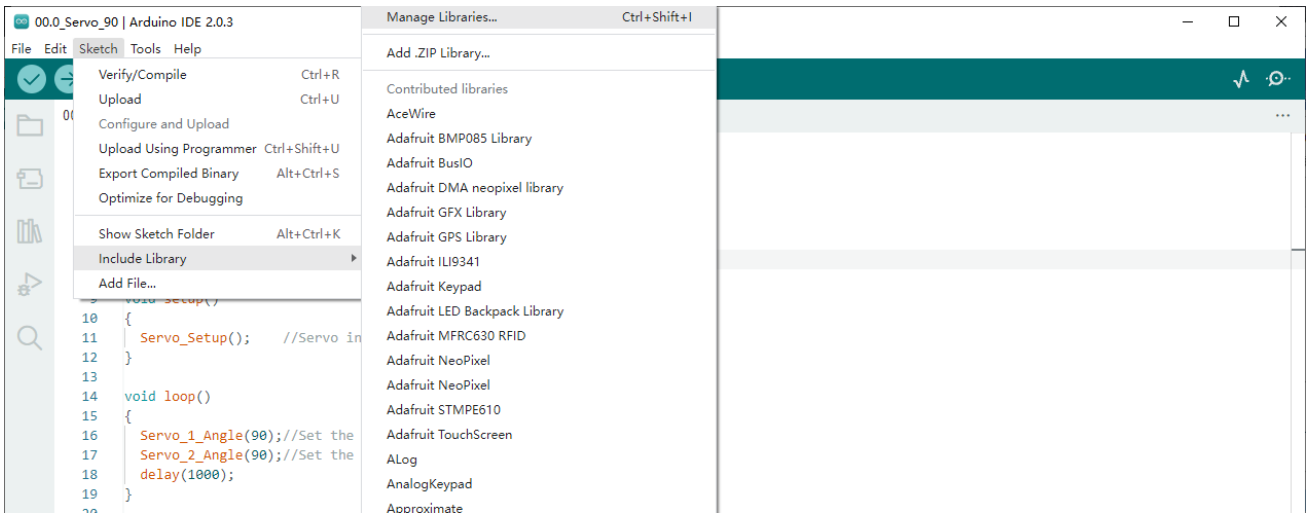
Here we take “00.0_Servo_90” in **“Freenove_4WD_Car_Kit_for_Raspberry_Pi_Pico\Ordinary_wheels\Sketches”** as an example.
The servo on the car is controlled by RP2040_PWM. Therefore, it is necessary to add the related libararies to Arduino IDE first.

How to Add Libraries

Method 2 is preferred.

Method 1

Open Arduino IDE, click Sketch on Menu bar ->Include Library -> Manage Libraries.

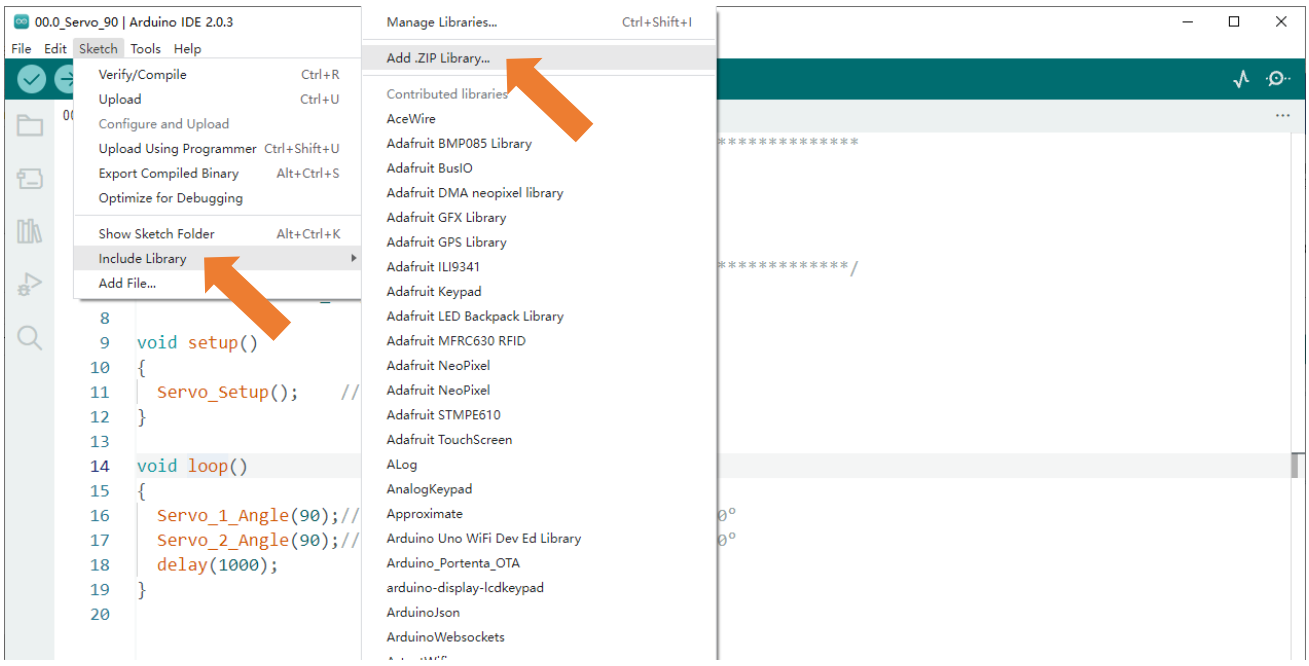


There is an input field on the right top of the pop-up window. Enter RP2040_PWM there and click to install the library boxed in the following figure.

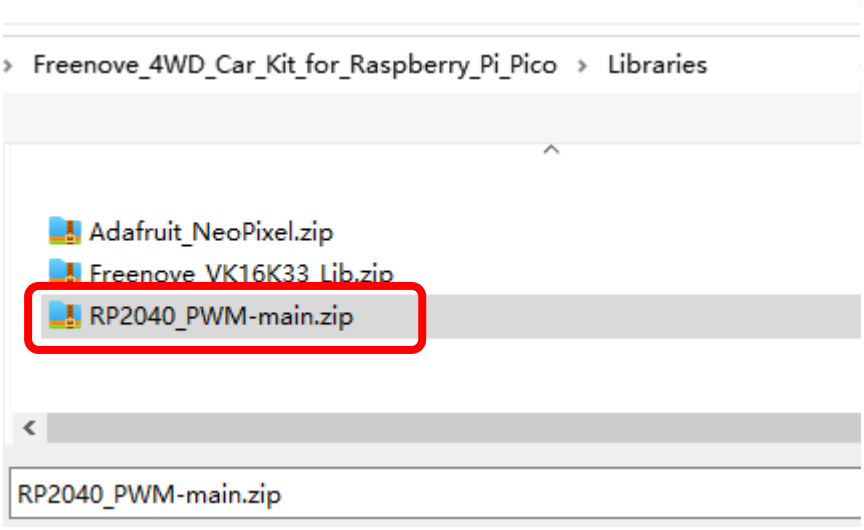
Wait for the installation to finish.

Method 2

Open Arduino IDE, click Sketch on Menu bar->Include Library ->Add .ZIP library.

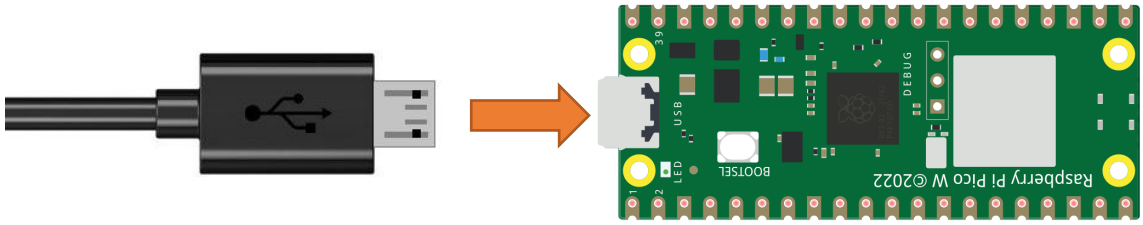


On the pop-up window, select RP2040_PWM-main.zip in Libraries folder under **"Freenove_4WD_Car_Kit_for_ Raspberry_Pi_Pico\Libraries"**, and then click Open.

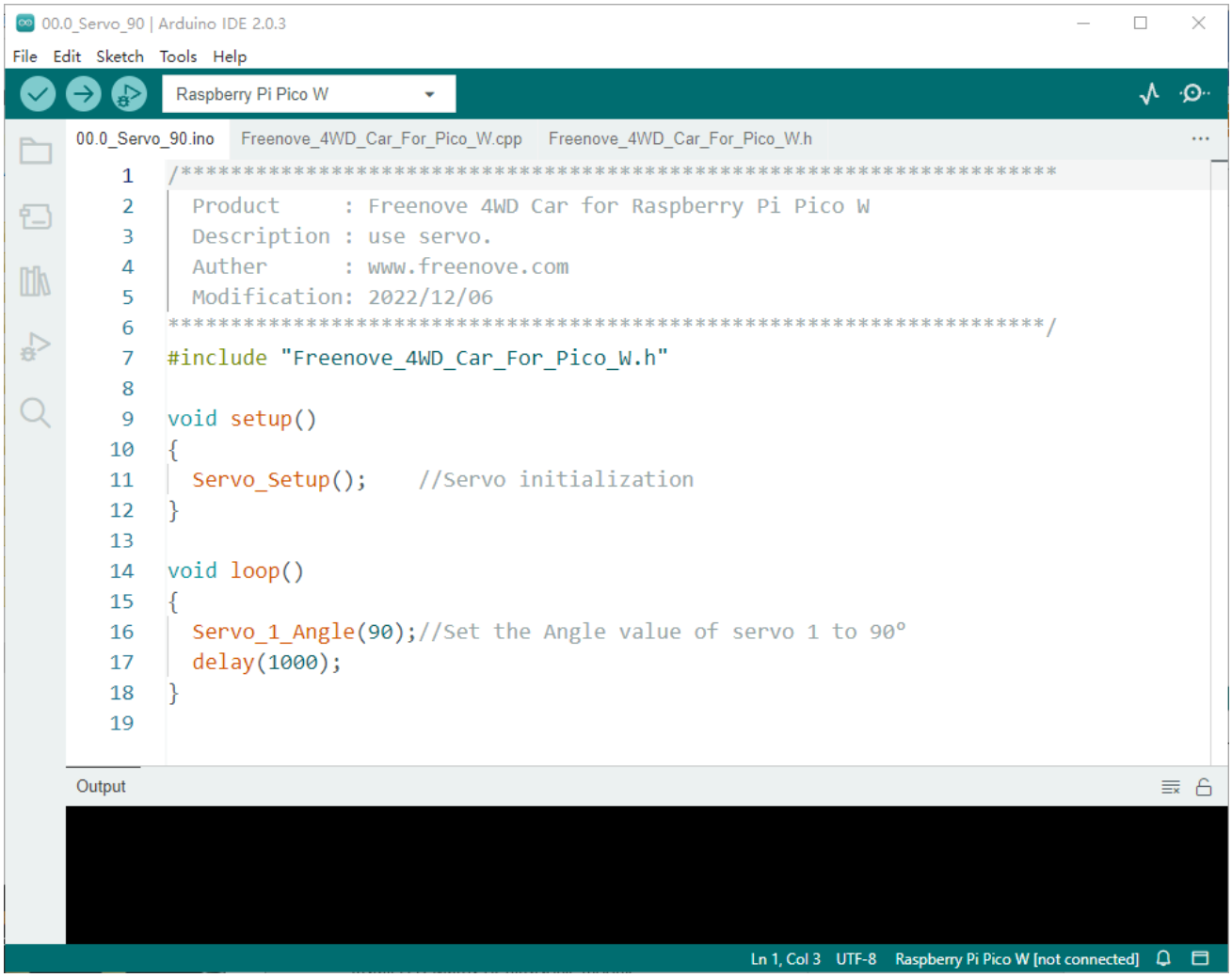

Freenove_4WD_Car_Kit_for_Raspberry_Pi_Pico
Libraries
Adafruit_NeoPixel.zip
Freenove_VK16K33_Lib.zip
RP2040_PWM-main.zip
RP2040_PWM-main.zip

How to Compile and Upload Code

Step 1. Connect your computer and Raspberry Pi Pico W with a USB cable.

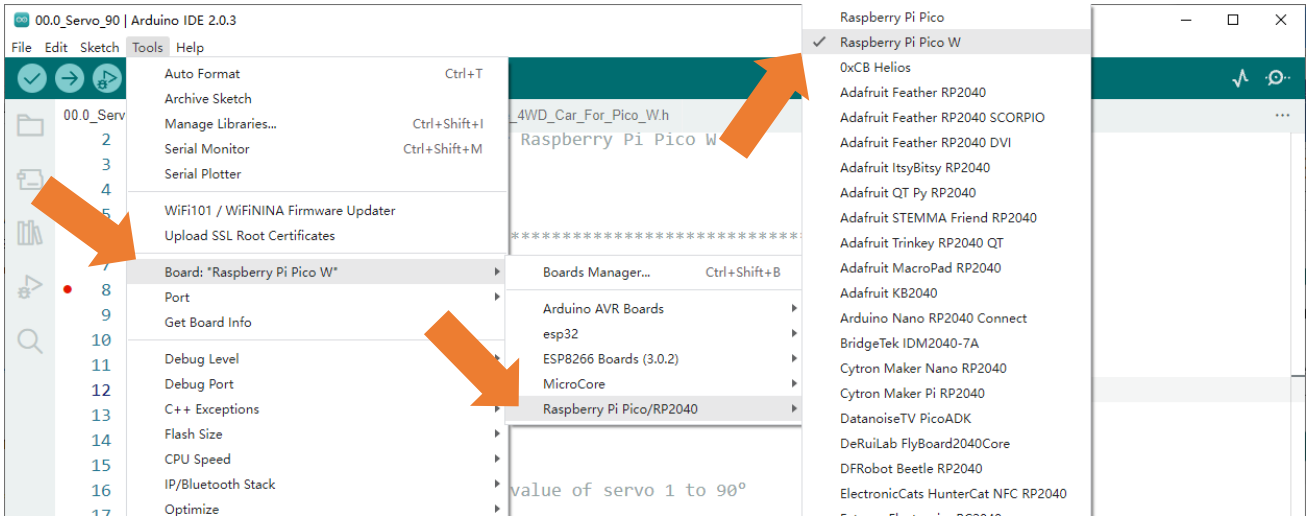


Step 2. Open "00.0_Servo_90" folder in **"Freenove_4WD_Car_Kit_for_Raspberry_Pi_Pico\Ordinary_wheels\Sketches"**, and double-click "00.0_Servo_90.ino". The code is to rotate the servo motors to 90°.



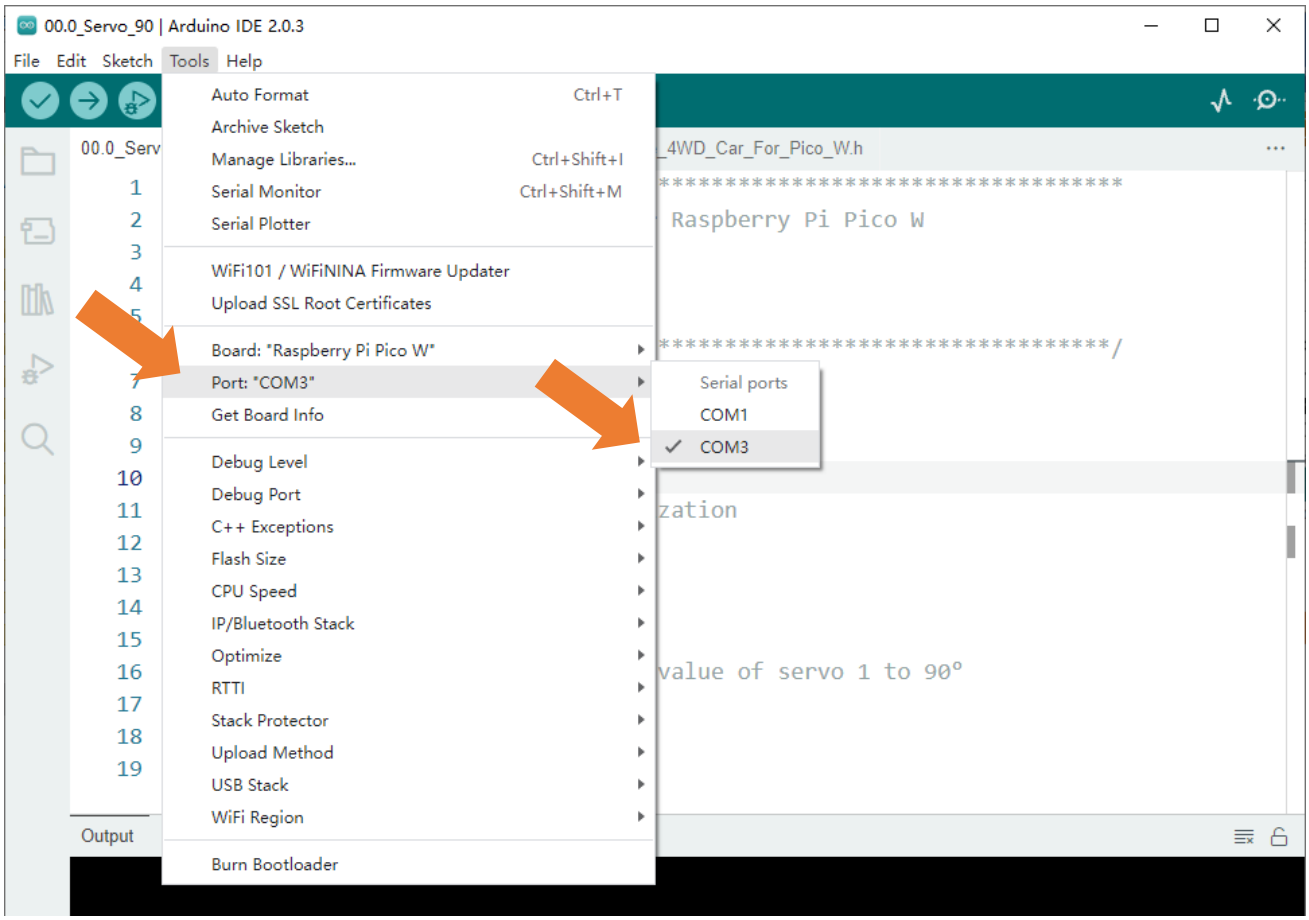
Step 3. Select development board.

Click Tools on Menu bar, click Board -> “Raspberry Pi Pico/RP2040” -> Raspberry Pi Pico W.

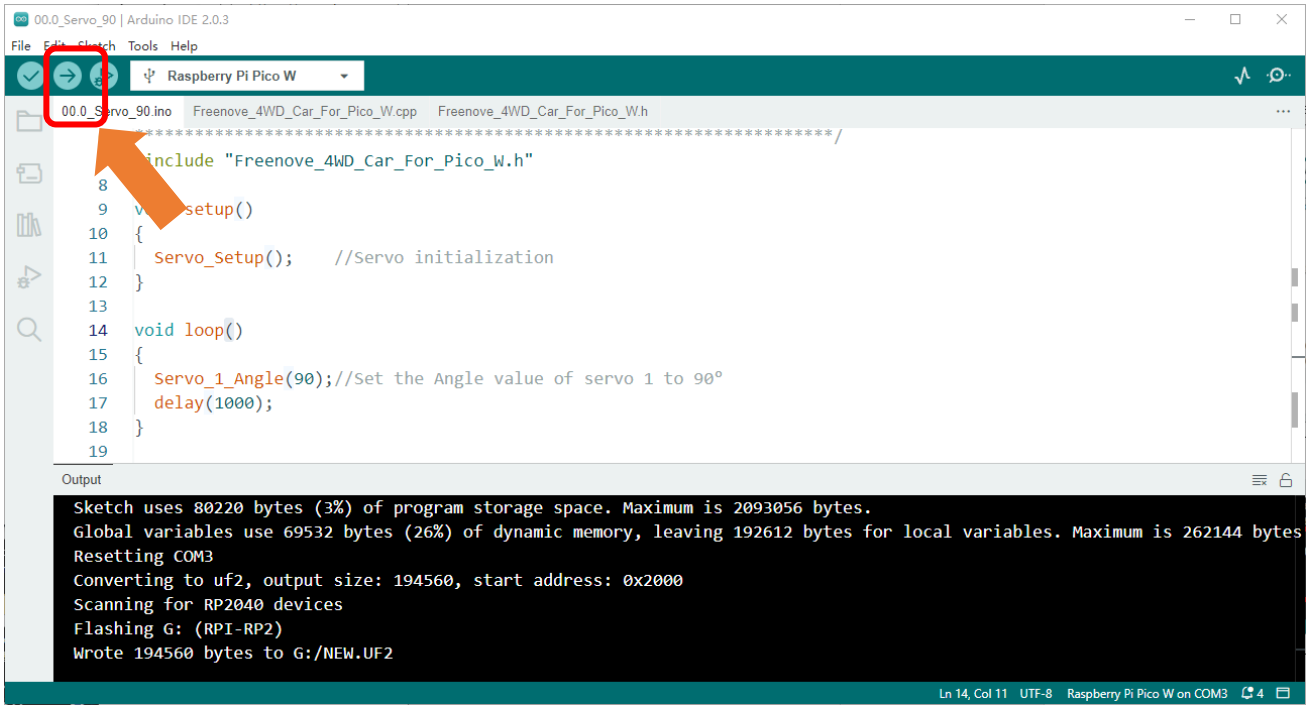


Step 4. Select serial port.

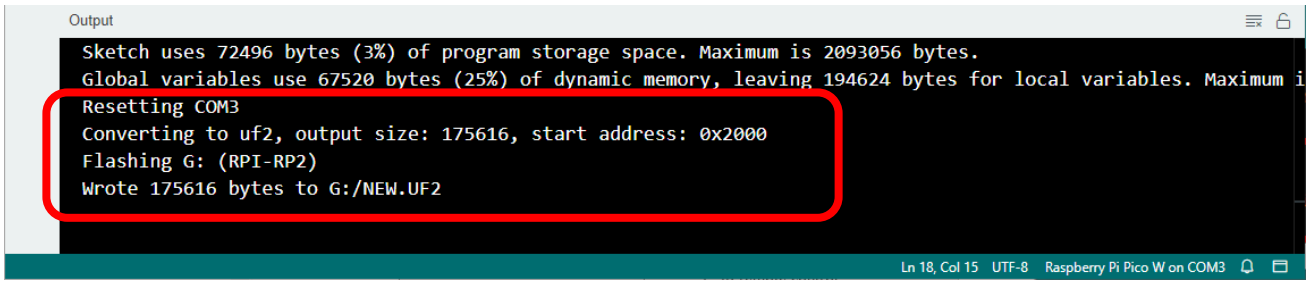
Click Tools on Menu bar, navigate your mouse to Port and select COMx on your computer. The value of COMx varies in different computers, but it will not affect the download function of Raspberry Pi Pico (W), as long as you select the correct one.



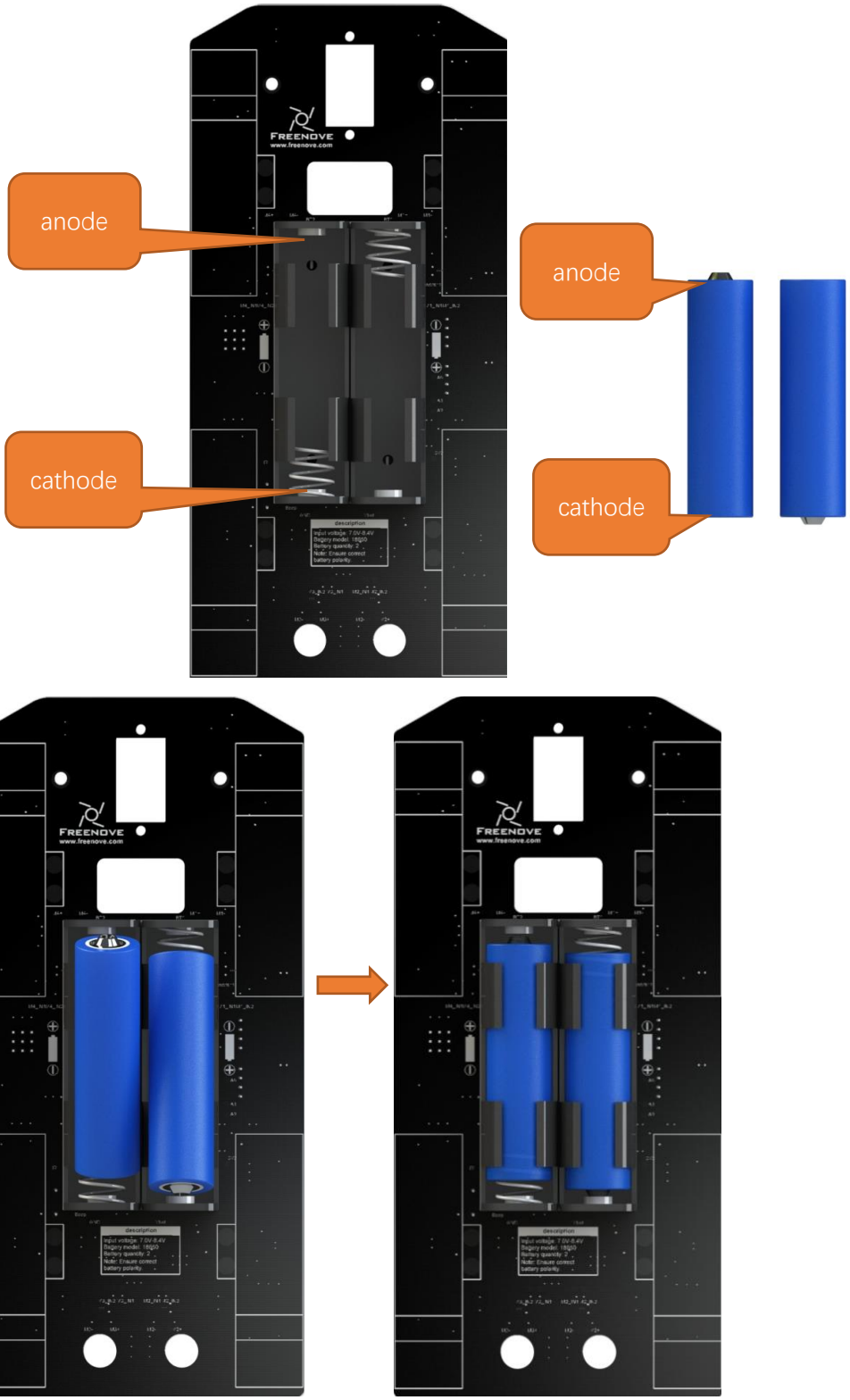
Click “Upload Using Programmer” and the program will be downloaded to Raspberry Pi Pico (W).



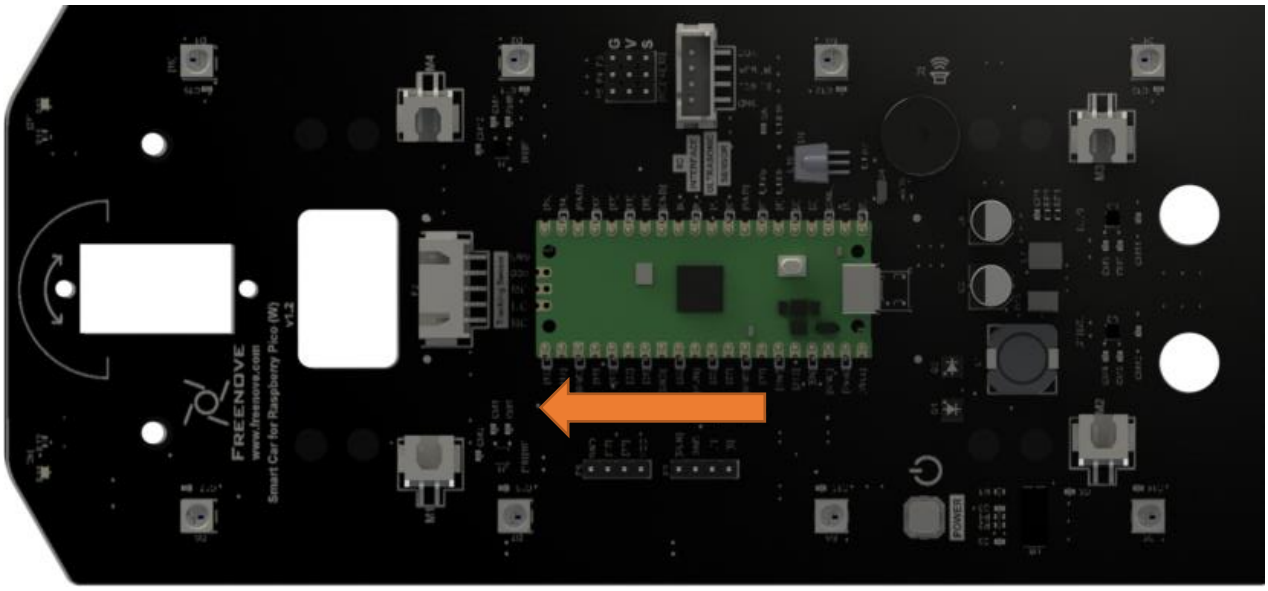
When you see the following content, it indicates that the program has been uploaded to Raspberry Pi Pico (W).



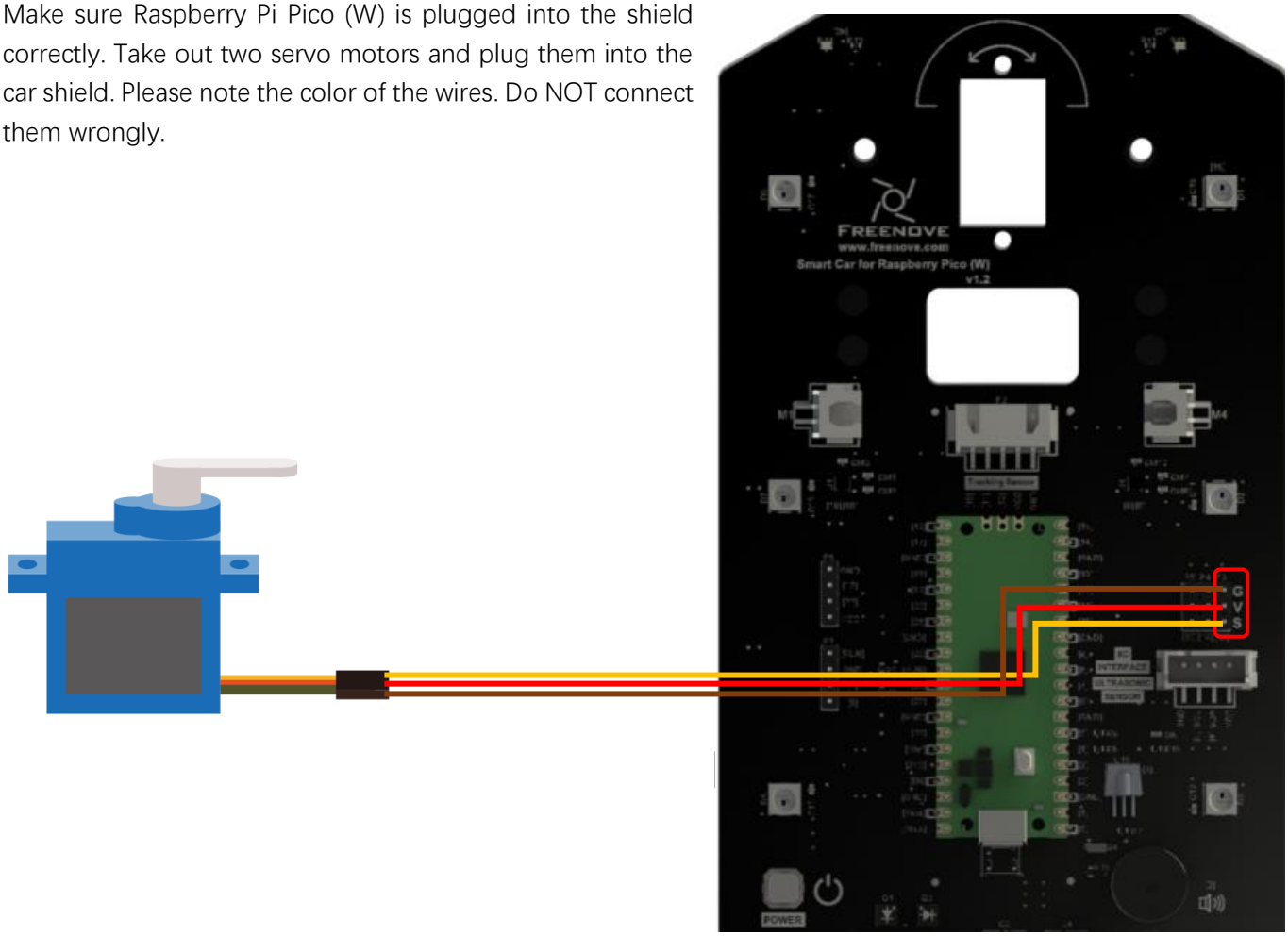
The car needs to be assembled with batteries installed. When installing them, please follow the silk print on the board.



Plug the Raspberry Pi Pico (W) to the car shield. Pay attention to the orientation of Raspberry Pi Pico (W).



Make sure Raspberry Pi Pico (W) is plugged into the shield correctly. Take out two servo motors and plug them into the car shield. Please note the color of the wires. Do NOT connect them wrongly.



Turn ON the switch and the servo will keep at 90°.

Chapter 1 Assembling Smart Car

If you have any concerns, please feel free to contact us via support@freenove.com

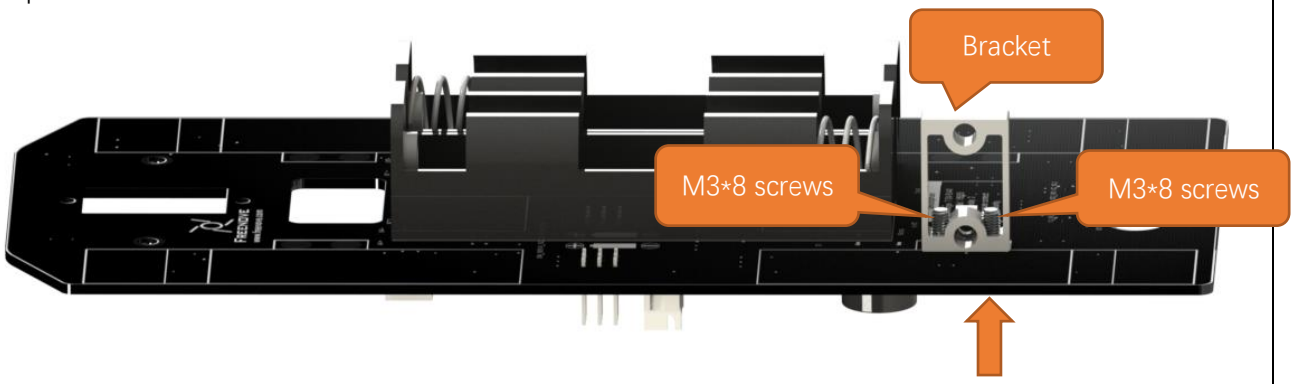
Assembling the Car

Installing Motor and Wheels

There are 4 bracket packages to fix motors, each containg an aluminum bracket, two M3*30 screws, two M3*8 screws and two M3 nuts, as shown below:

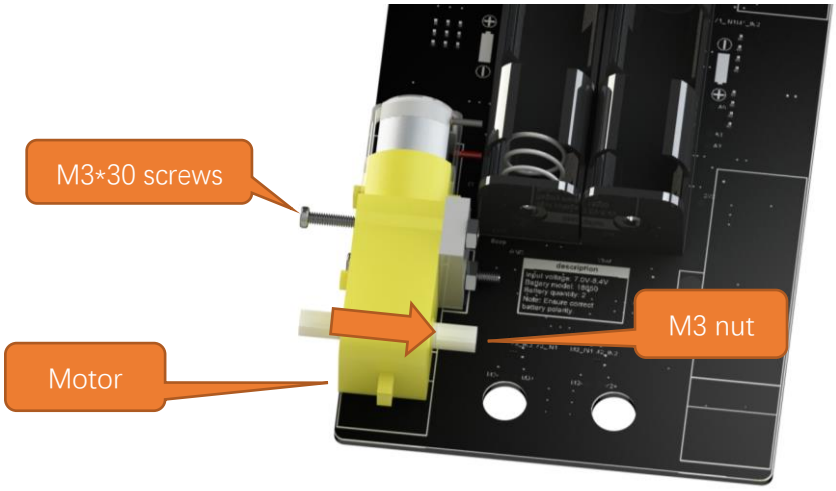
Installation steps:

Step 1 Fix the bracket on the shield.



Put the bottom of the car up, use two M3*8 screws to fix the bracket on the shield.

Step 2 Install the motor to the bracket.

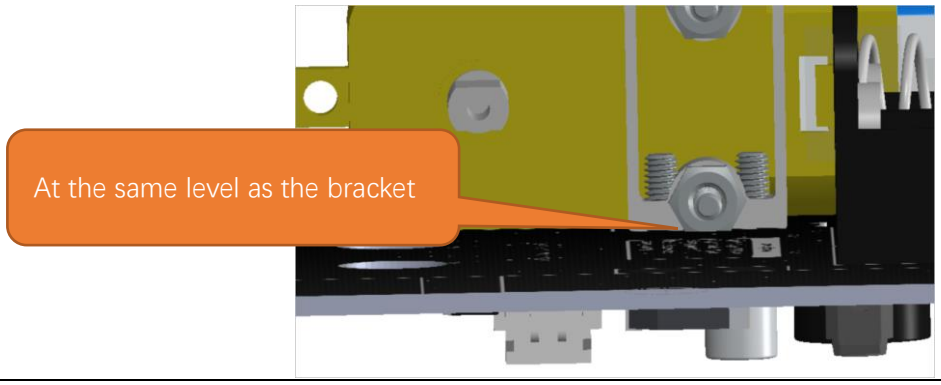


Use two M3*30 screws and two M3 nuts to fix the motor on the bracket.

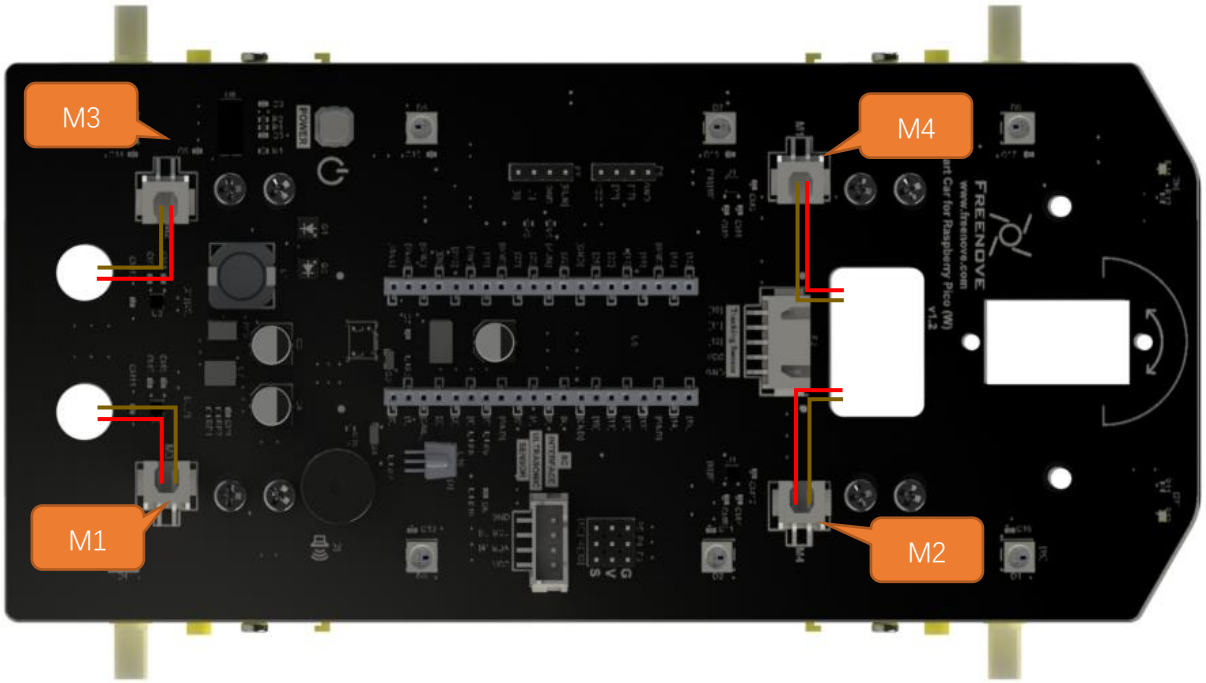
Caution: Do not remove the cable tie from the motor; otherwise, the motor cable may become detached.

If you perform step 2 before step 1, please check whether the M3 nut located under the motor in step 2 is at the same level as the bracket.

If the M3 nut below is not horizontal, it may easily cause damage to the PCB surface during installation step 1, resulting in a short circuit on the control board.

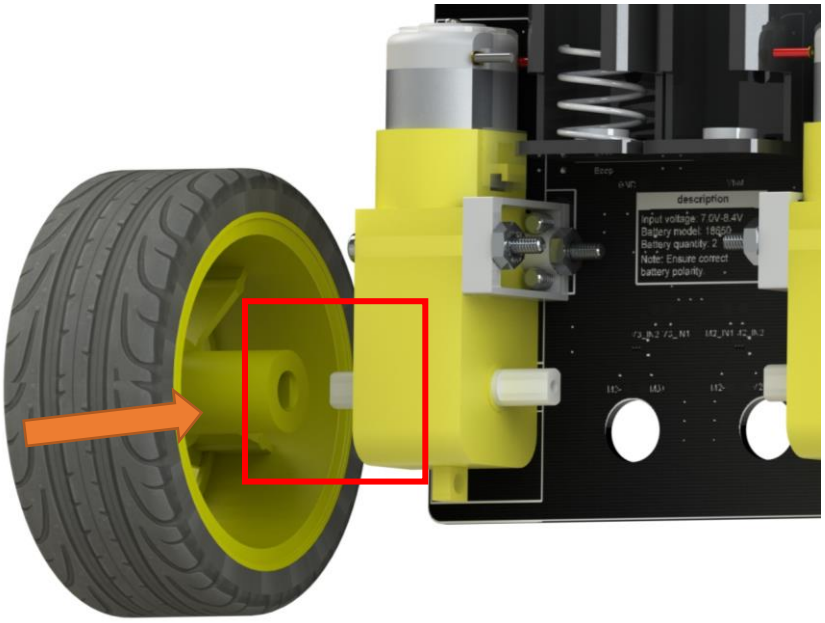


Step 3 Connect the motor wire to the corresponding socket.

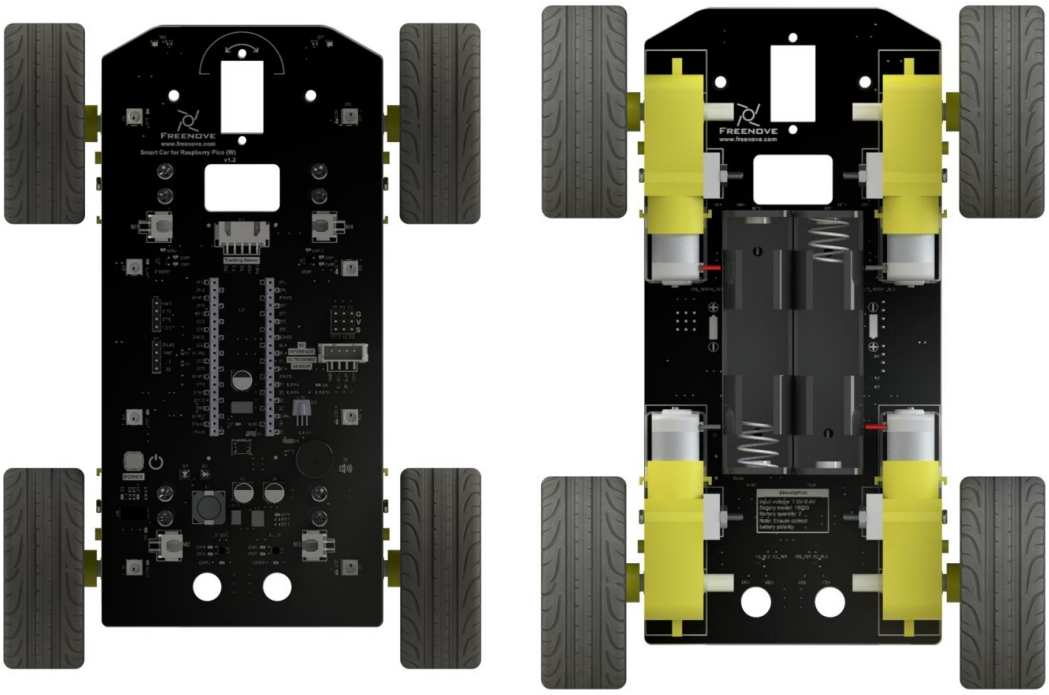


Put the motor wires through the holes on the shield and connect them to the corresponding sockets on the top.

Step 4 Install the wheel to the motor.



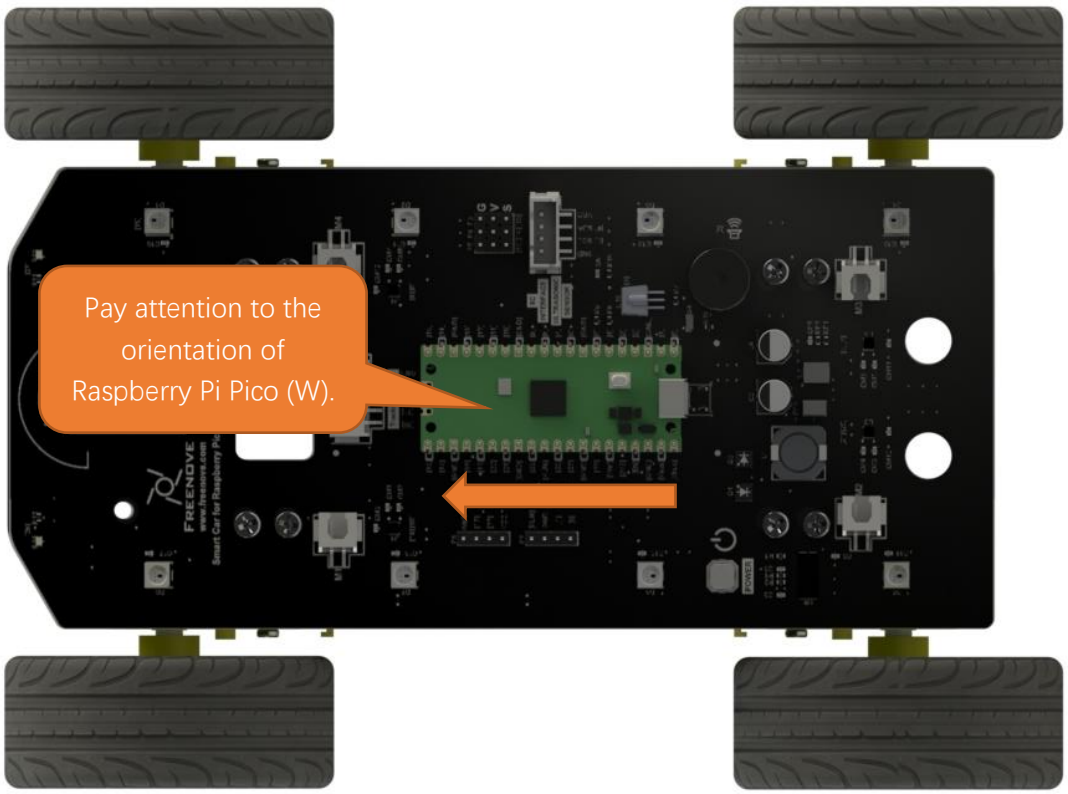
Note: The hole is not round. Please align the hole and fix the wheel to the motor.



The installation of the four wheels is similar. Repeat the above installation steps to complete the installation.

Plugging in Raspberry Pi Pico (W)

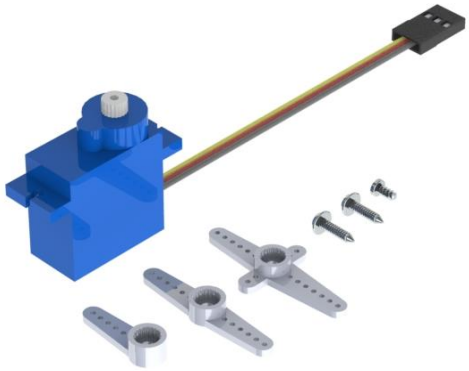
Plug Raspberry Pi Pico (W) into the shield.



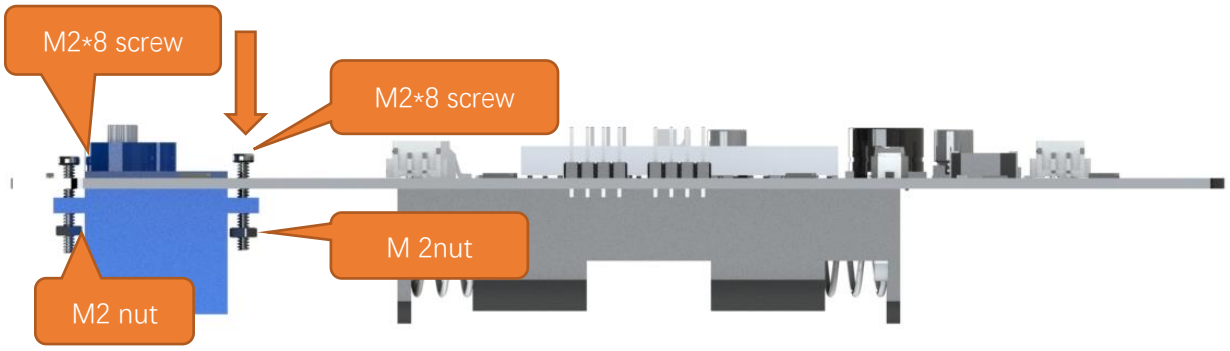
Please pay attention to the orientation of Raspberry Pi Pico (W). Do NOT reverse it; Otherwise, it may burn the Raspberry Pi Pico (W).

Installing Servo

Servo package: one servo, three rocker arms, two M2*8 screws and one M2*4 screw.

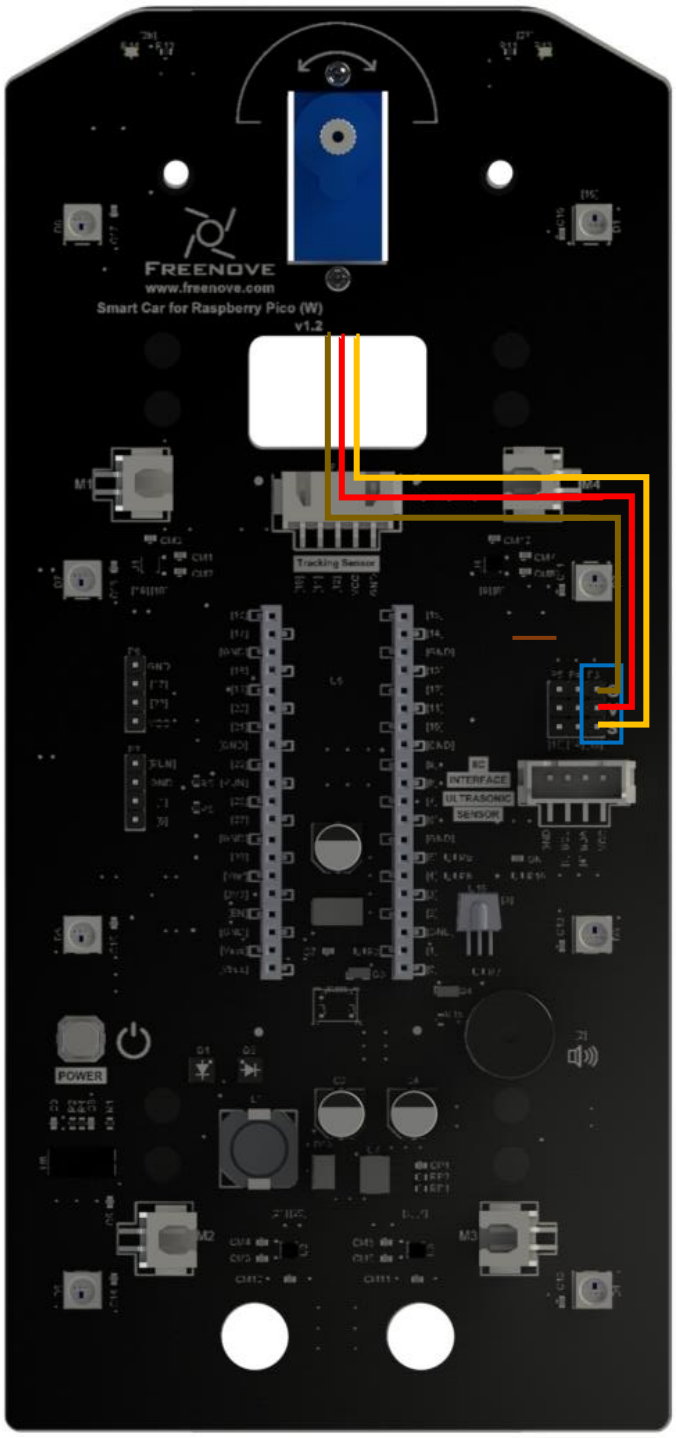


Step 1 Fix servo1 to the shield.



Use two M2*16 screws and two M2 nuts to mount the servo1 on the shield. Pay attention to the servo's direction.

Step 5 Wiring of the servor motor

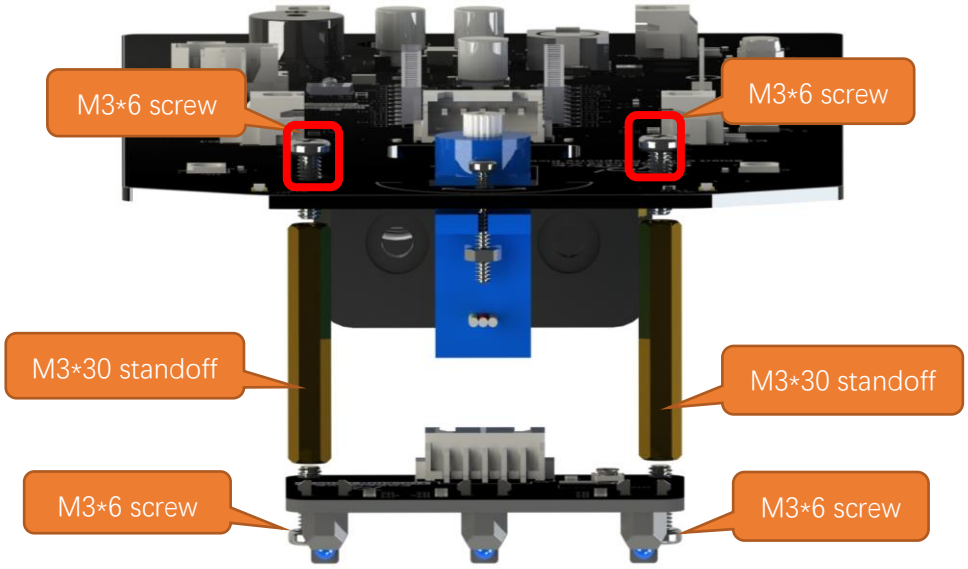


Connect servo1 to P3 interfaces on the shield.
Pay attention to the color of the wires.
Do not connect them wrongly.

Connect servo to P3 interfaces on the shield respectively.
Pay attention to the color of the wires. Do not connect them wrongly.

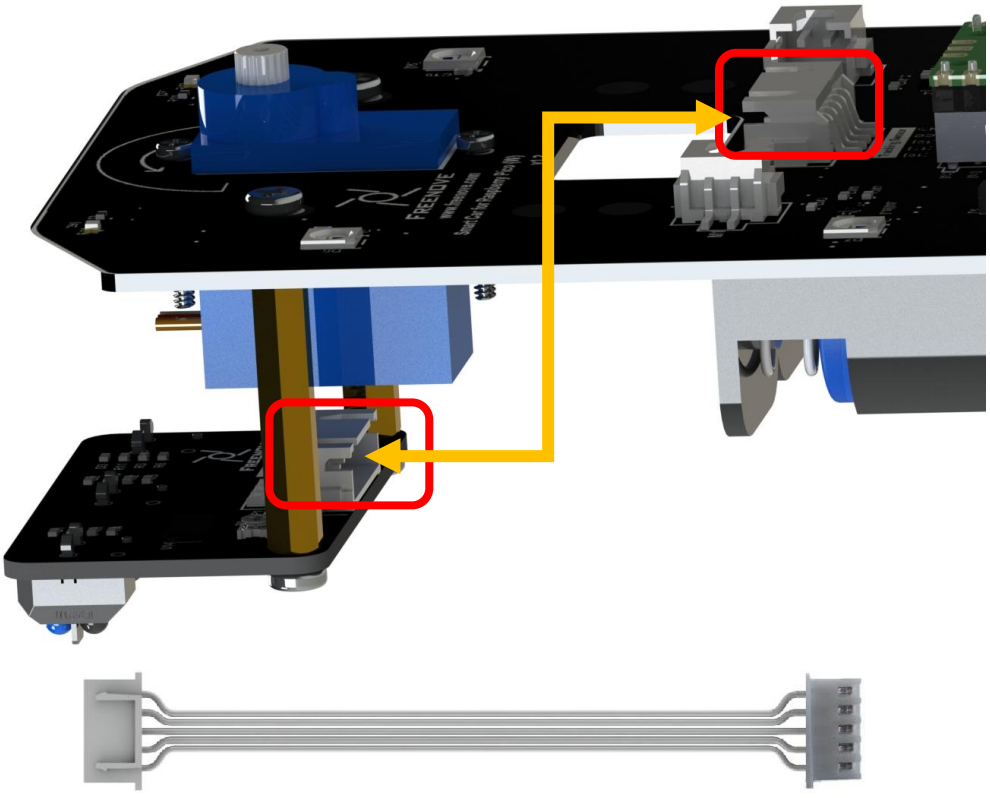
Installing Line Tracking Module

Step 1 Installing line tracking module



First, use two M3*6 screws to fix two M3*30 standoffs to the bottom of the car, and then use two M3*6 screws to fix the line tracking module to the standoffs.

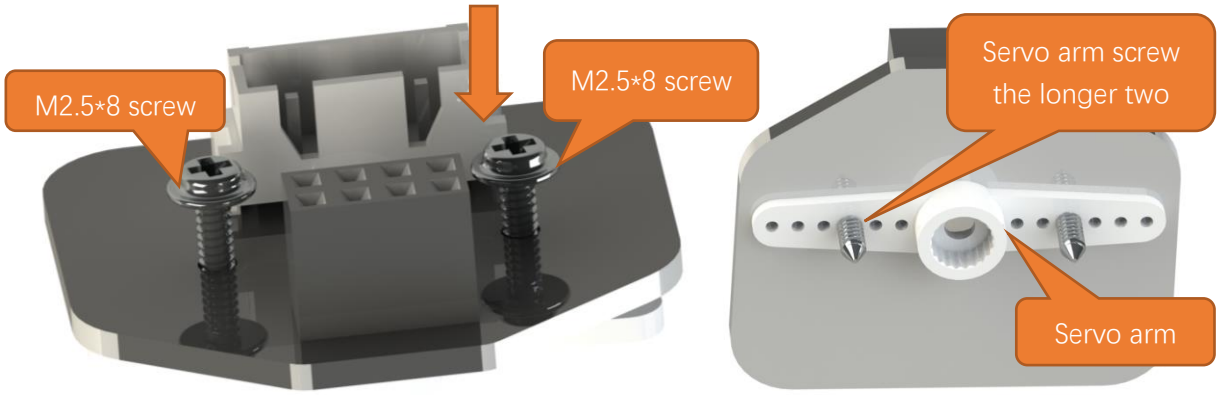
Step 2 Connect the cable to the tracking module



Use cable to connect the two connectors marked above.

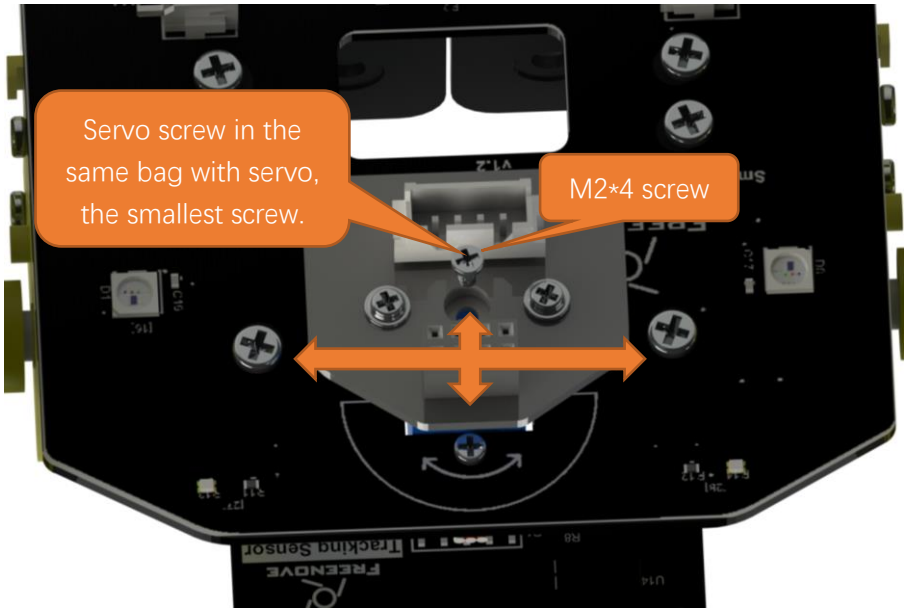
Mounting the Servo Arm to Module Connector.

Step 1 Mount the servo arm to acoustic wave module or expression module connector. The screw and arm are in the same bag with servo.

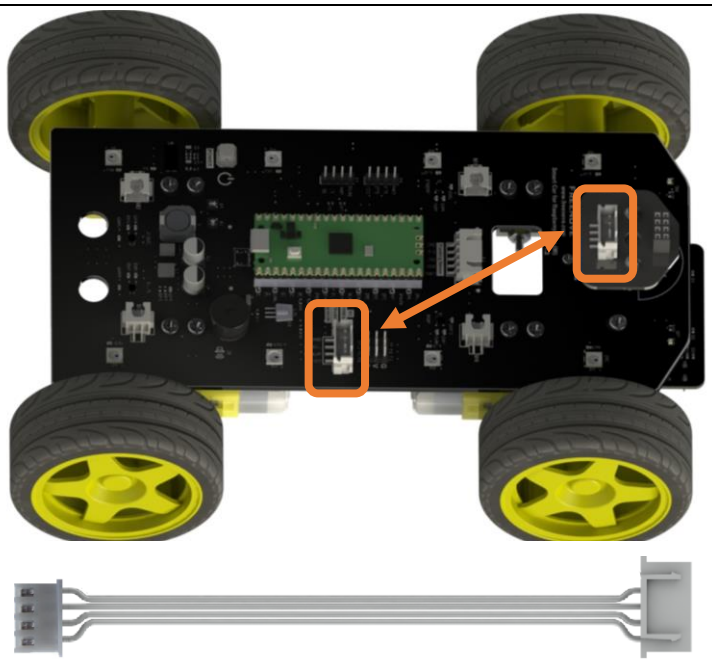


Step 2 Connect servo arm to servo. Make sure servo is installed at 90 degrees.

Note: Before attache the rocker arm to servo, please adjust the servo at 90°.You can refer to here.

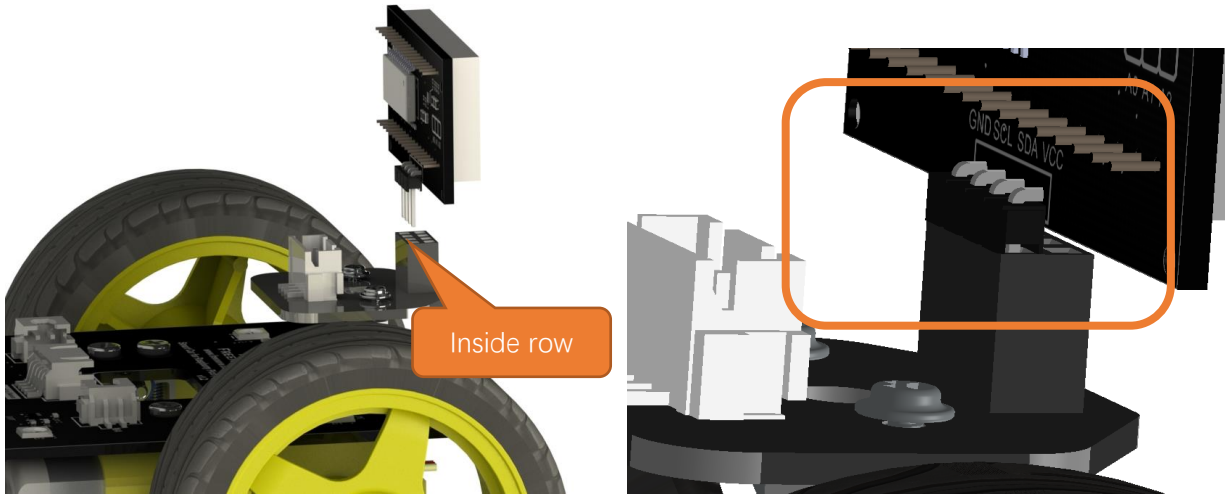


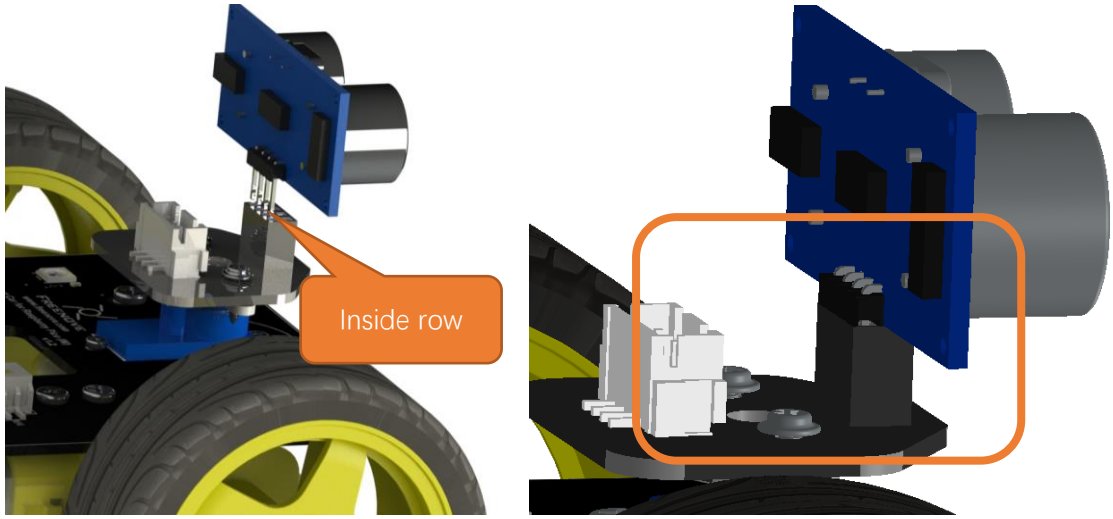
Step 3 Connect the two connectors marked below with a cable.



Plugging in LED Matrix or ultrasonic module.

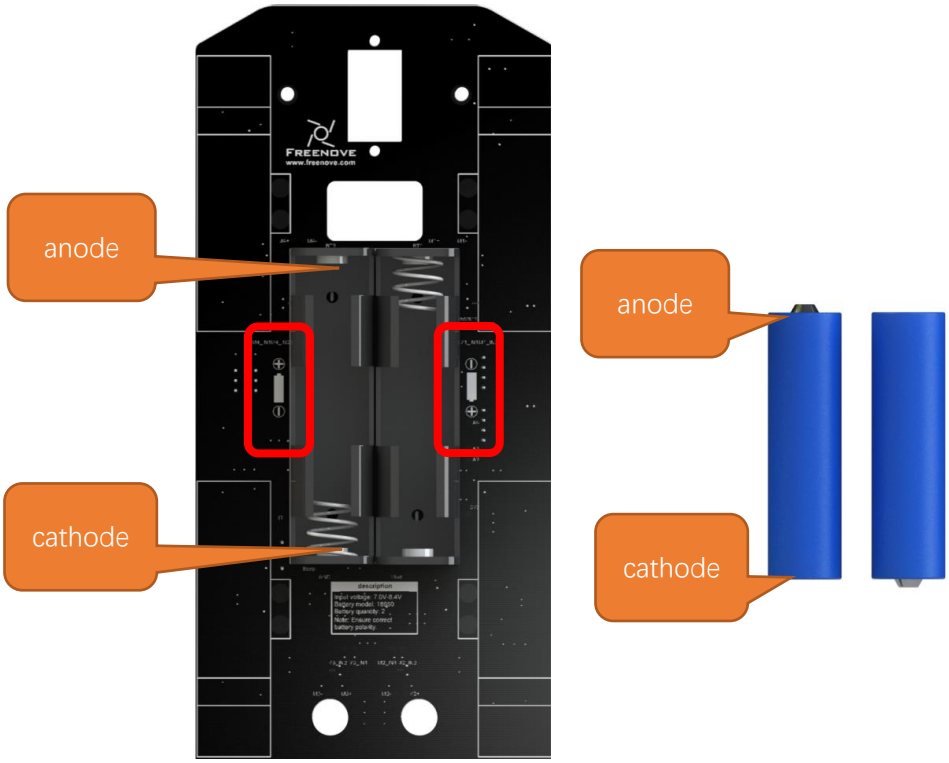
Step 1 Install LED Matrix or ultrasonic module.

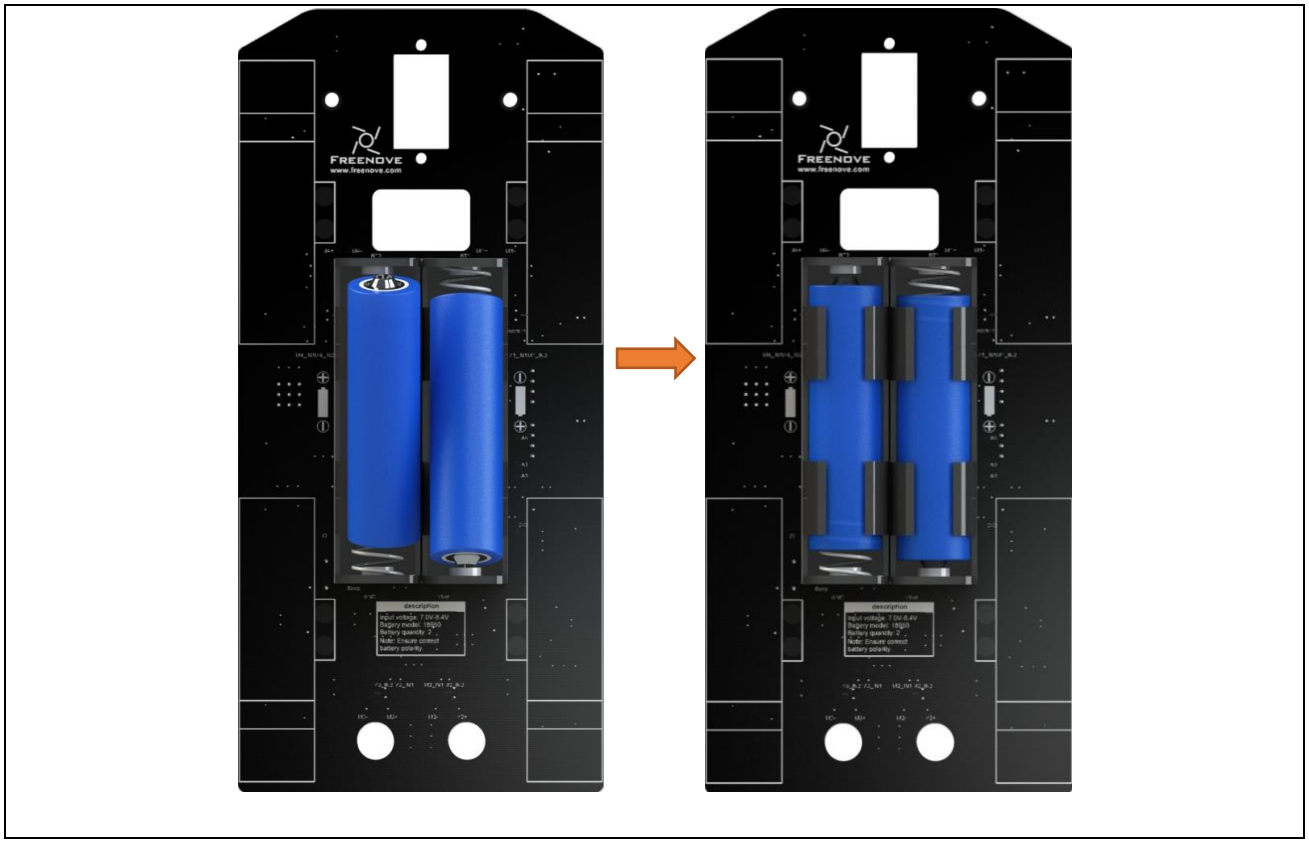




Installing two 18650 batteries

Step 1 Installed with batteries. When installing them, please following the silk print on the board.

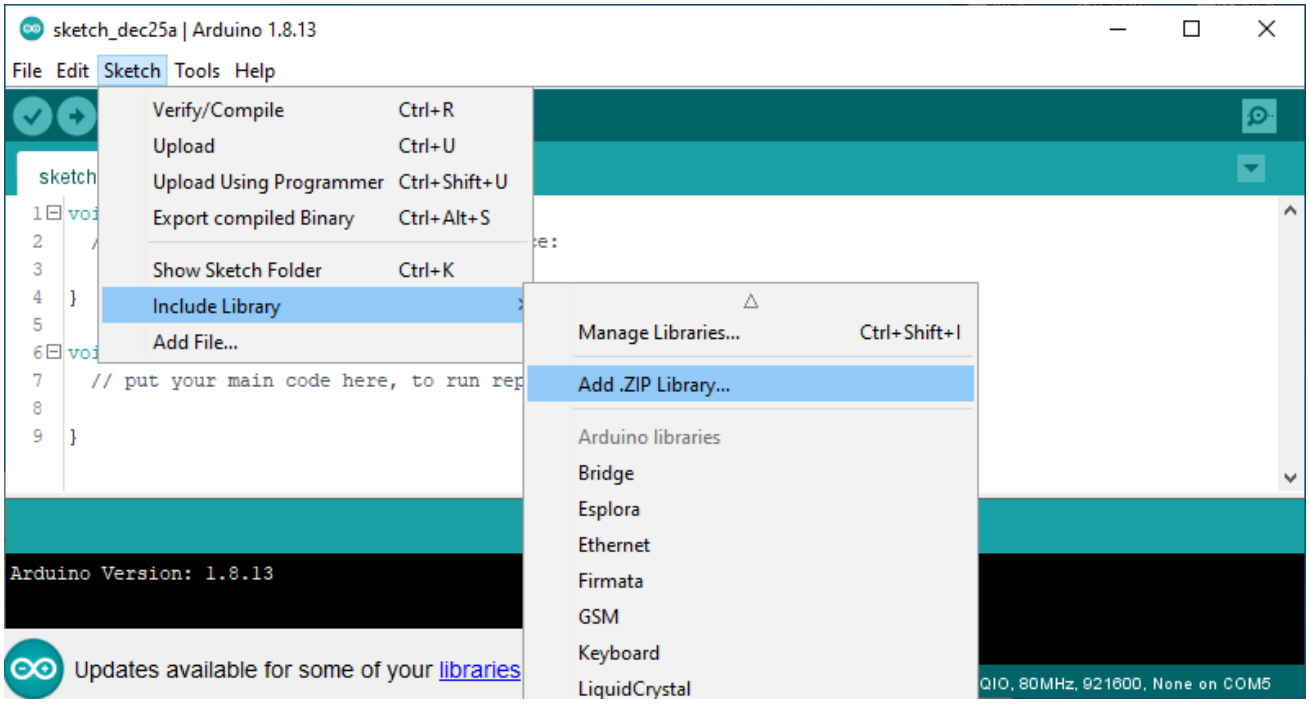



FREENOVE
www.freenove.com
FREENOVE
www.freenove.com

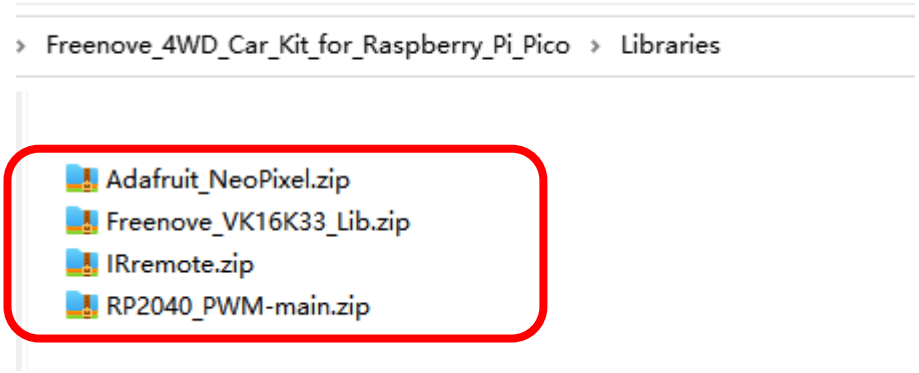
How to Play

Add libraries

Open the **Arduino IDE**, Click **Sketch** on the menu bar ->**Include Library** -> **Add .ZIP Library...**



In the new pop-up window, select **Freenove_4WD_Car_Kit_for_Raspberry_Pi_Pico_W\Libraries**, select every Library, click **Open**, and repeat this process until you have installed all six Libraries into the Arduino.



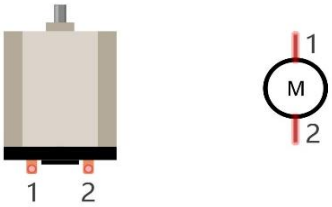
Chapter 2 Module test

If you have any concerns, please feel free to contact us via support@freenove.com

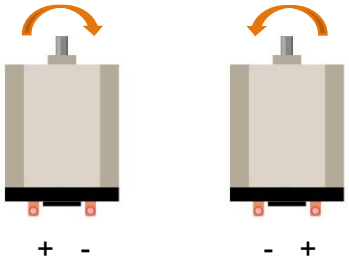
2.1 Motor

Motor

A motor is a device that converts electrical energy into mechanical energy. Motor consists of two parts: stator and rotor. When motor works, the stationary part is stator, and the rotating part is rotor. Stator is usually the outer case of motor, and it has terminals to connect to the power. Rotor is usually the shaft of motor, and can drive other mechanical devices to run. Diagram below is a small DC motor with two pins.

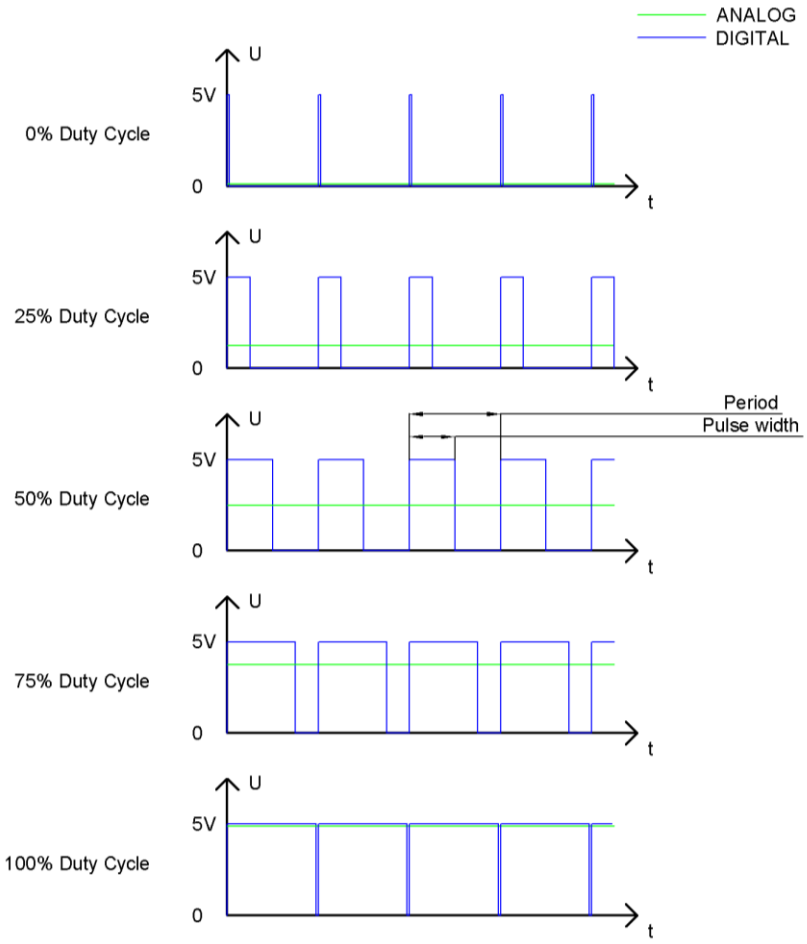


When a motor is connected to the power supply, it will rotate in one direction. Reverse the polarity of power supply, then the motor rotates in opposite direction.



PWM

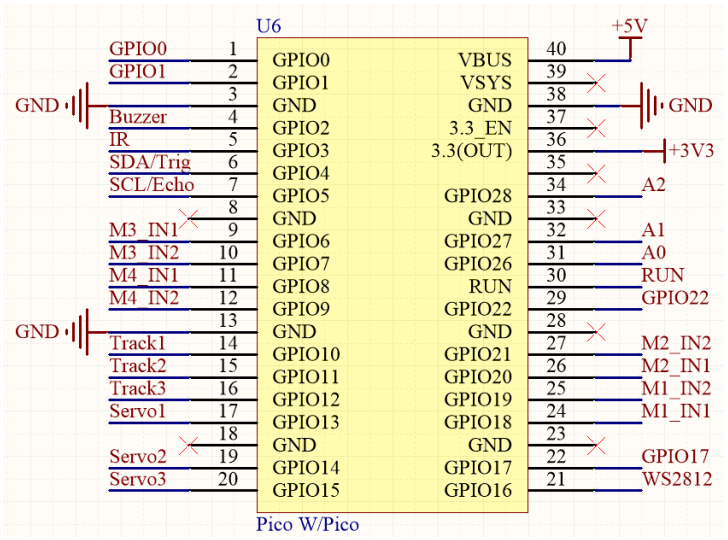
PWM, Pulse Width Modulation, uses digital pins to send certain frequencies of square waves, that is, the output of high levels and low levels, which alternately last for a while. The total time for each set of high levels and low levels is generally fixed, which is called the period (the reciprocal of the period is frequency). The time of high level outputs are generally called “pulse width”, and the duty cycle is the percentage of the ratio of pulse duration, or pulse width (PW) to the total period (T) of the waveform.
The longer the output of high levels last, the larger the duty cycle and the higher the corresponding voltage in analog signal will be. The following figures show how the analogs signal voltage vary between 0V-5V (high level is 5V) corresponding to the pulse width 0%-100%:

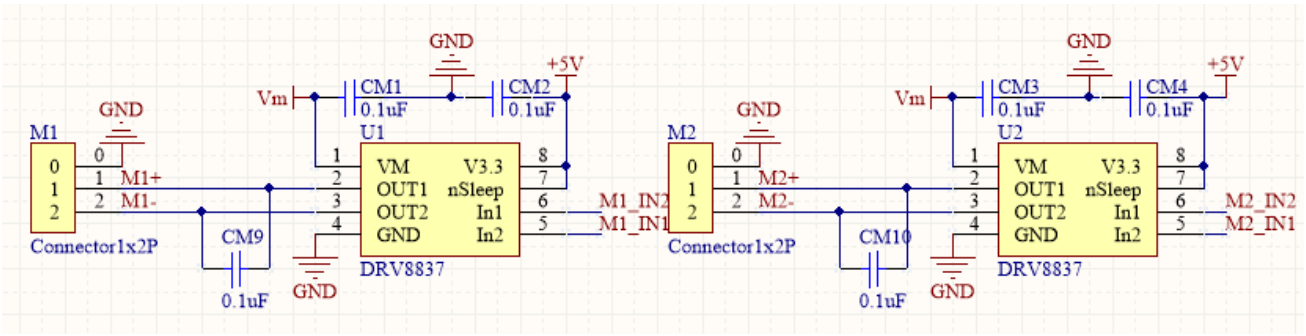


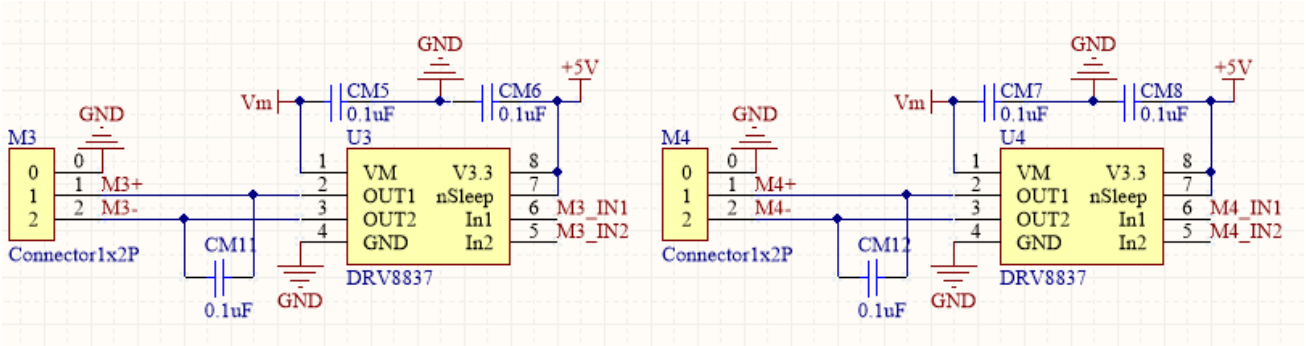
The longer the PWM duty cycle is, the higher the output power will be. Now that we understand this relationship, we can use PWM to control the brightness of an LED or the speed of DC motor and so on.

Schematic

The PWM control signal of this car is provided by Pico (W) with the RP2040_PWM library. RP2040_PWM library, the PWM library of Pico (W), uses Hardware-PWM module on RP2040-based boards to create and output the PWM of any GPIO pin. These purely hardware-based PWM channels can generate from very low (lowest is 7.5Hz * (F_CPU/125)) to very high PWM frequencies (in the MHz range, up to (F_CPU/2) or 62.5MHz). The reason why we use this library is that it is based on hardware-PWM channels, so its operation will not be blocked by bad-behaving software functions / tasks.







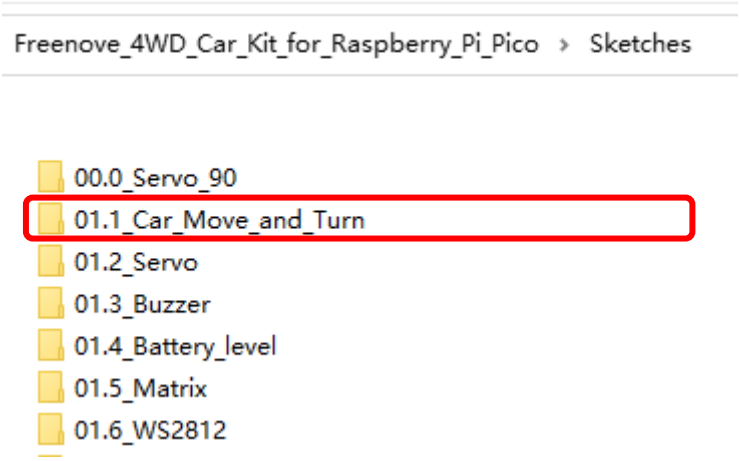
Pins of Raspberry Pi Pico W	Funtions	Description
GPIO18	Motor	M1_IN1
GPIO19		M1_IN2
GPIO20		M2_IN1
GPIO21		M2_IN2
GPIO6		M3_IN1
GPIO7		M3_IN2
GPIO8		M4_IN1
GPIO9		M4_IN2

As can be seen from the figure above, pins 18 and 19 are configured to control motor M1, pins 20 and 21 to control M2, pins 6 and 7 to control M3, and pins 8 and 9 to control M4.

Mx_IN1	Mx_IN2	Rotating direction of the wheels
1	0	Forward
0	1	backward

Sketch

Next, we download the code to Raspberry Pi Pico (W) to test the motor. Open "01.1_Car_Move_and_Turn" folder under **"Freenove_4WD_Car_Kit_for_Raspberry_Pi_Pico\Ordinary_wheels\Sketches"** and double-click "01.1_Car_Move_and_Turn.ino".



Code

```
#include "Freenove_4WD_Car_For_Pico_W.h"
void setup()
{
  Servo_Setup();
  Motor_Setup();       // Motor initialization
}

void loop()
{
  Motor_Move(50 , 50);     //go forward
  delay(1000);
  Motor_Move(0 , 0);         //stop
  delay(1000);
  Motor_Move(-50 , -50); //go back
  delay(1000);
  Motor_Move(0 , 0);         //stop
  delay(1000);
  Motor_Move(-50 , 50);   //turn left
  delay(1000);
  Motor_Move(0 , 0);         //stop
  delay(1000);
  Motor_Move(50 , -50);   //turn right
  delay(1000);
  Motor_Move(0 , 0);         //stop
  delay(1000);
```

```
26  }
```

After downloading the code, please place the car to a relatively open area. Turn on the power switch and you can see the car go forward, backward, turn left and turn right repeatedly.

Code Explanation

If you are not familiar with Arduino IDE, you can visit https://www.arduino.cc/reference/en/ to learn more.

Add the header file of the car. Each time before controlling the car, please add header file first.

```
1  #include "Freenove_4WD_Car_For_Pico_W.h"
```

Motor_Setup function, initialize the driver chip of the motor.

```
5    Motor_Setup();        // Motor initialization
```

Loop function can be used repeatedly in the program. Here we call Motor_Move function to control the car to move forward, backward, turn left and turn right repeatedly.

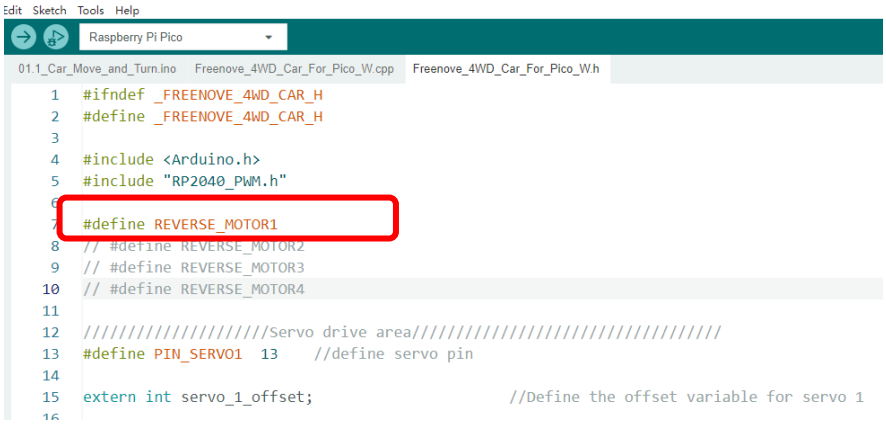
```
8   void loop()
9   {
10    Motor_Move(50 , 50);      //go forward
11    delay(1000);
12    Motor_Move(0 , 0);          //stop
13    delay(1000);
14    Motor_Move(-50 , -50); //go back
15    delay(1000);
16    Motor_Move(0 , 0);          //stop
17    delay(1000);
18    Motor_Move(-50 , 50);    //turn left
19    delay(1000);
20    Motor_Move(0 , 0);          //stop
21    delay(1000);
22    Motor_Move(50 , -50);    //turn right
23    delay(1000);
24    Motor_Move(0 , 0);          //stop
25    delay(1000);
26  }
```

Motor_Move_Init function is to control the car. The four parameters m1_speed, m2_speed, m3_speed, m4_speed range from -100 to 100. When the value is positive number, the motors move forward. Otherwise, they move backwards. m1_speed, m2_speed represent the two motors on the left and m3_speed, m4_speed embody those on the right.

```
    void Motor_Move_Init(int m1_speed, int m2_speed, int m3_speed, int m4_speed);
```

The Motor_Move function is used to control the car, which sets the left two motors as one group and the right two motors as the other, with the motors in the same group featuring the same speed.

If your car is moving in the opposite direction of the programmed direction, you can change the motor rotation direction of the motors through micro definition. Take the motor M1 as an example, modify the macro definition in the "Freenove_4WD_Car_For_Pico_W.h" file to "#define REVERSE_MOTOR1".

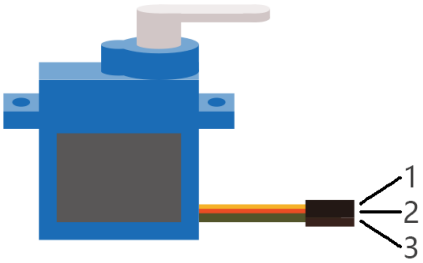


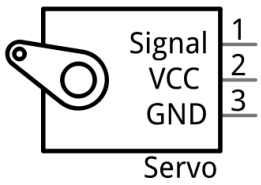

```
void Motor_Move(int Left_speed, int Right_speed) {
  int lf, lb, rf, rb;
  lf = lb = Left_speed;
  rf = rb = Right_speed;
#ifdef REVERSE_MOTOR1
  lf = -Left_speed;
#endif
#ifdef REVERSE_MOTOR2
  lb = -Left_speed;
#endif
#ifdef REVERSE_MOTOR3
  rf = -Right_speed;
#endif
#ifdef REVERSE_MOTOR4
  rb = -Right_speed;
#endif
  Motor_Move_Init(lf, lb, rf, rb);
}
```

2.2 Servo

Servo

Servo is a compact package, which consists of a DC motor, a set of reduction gears to provide torque, a sensor and control circuit board. Most servos only have a 180-degree range of motion via their "horn". Servos can output higher torque than a simple DC motor alone and they are widely used to control motion in model cars, model airplanes, robots, etc. Servos have three wire leads that usually terminate to a male or female 3-pin plug. Two leads are for electric power: positive (2-VCC, Red wire), negative (3-GND, Brown wire), and the signal line (1-Signal, Orange wire), as represented in the Servo provided in your Kit.





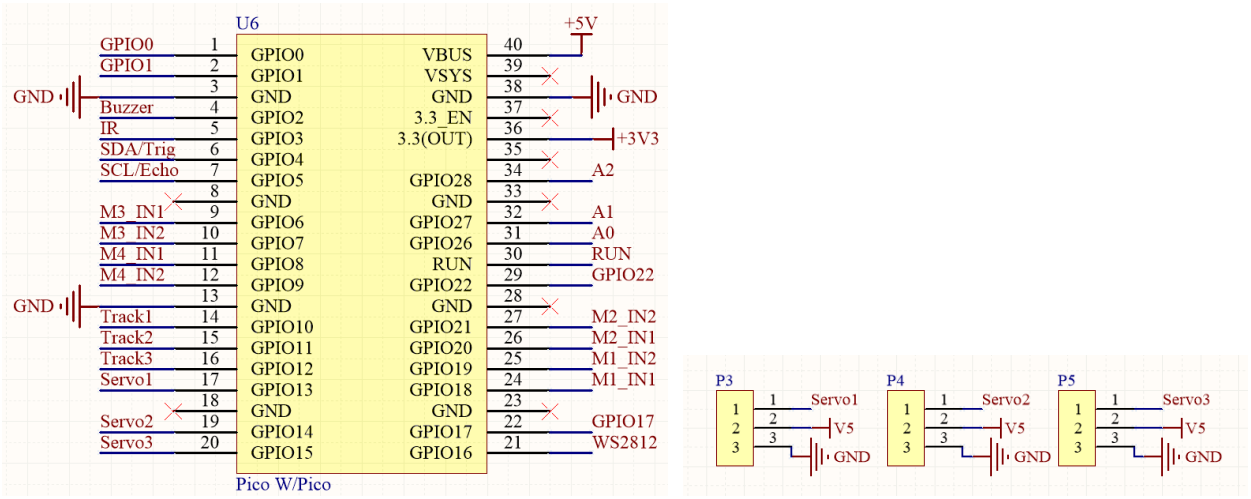
We will use a 50Hz PWM signal with a duty cycle in a certain range to drive the Servo. The lasting time of 0.5ms-2.5ms of PWM single cycle high level corresponds to the servo angle 0 degrees - 180 degree linearly. Part of the corresponding values are as follows:

High level time	Servo angle
0.5ms	0 degree
1ms	45 degree
1.5ms	0 degree
2ms	45 degree
2.5ms	180 degree

When you change the servo signal value, the servo will rotate to the designated angle.

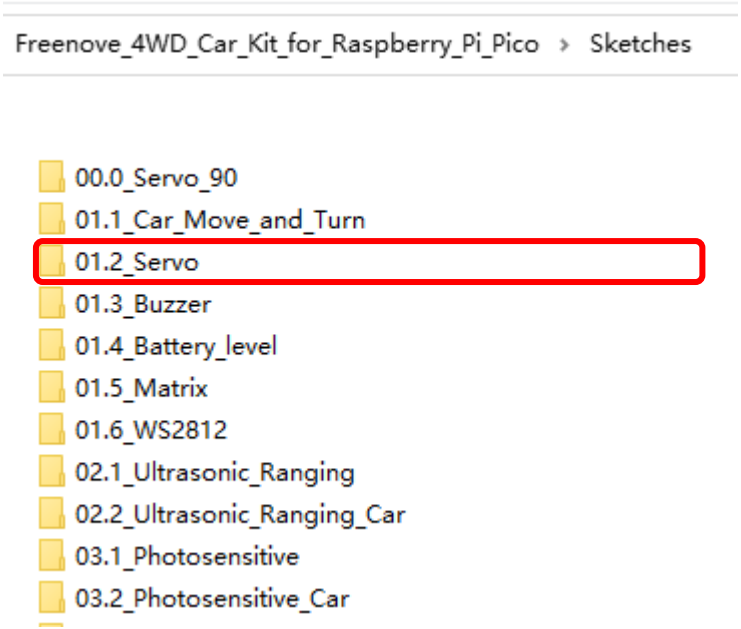
Schematic

Pico (W), based on the RP2040_PWM library, provides the servos' driving signals.



Sketch

Download the code to Raspberry Pi Pico (W) to make servo sweep back and forth.
Open "01.2_Servo" in **"Freenove_4WD_Car_Kit_for_Raspberry_Pi_Pico\Ordinary_wheels\Sketches"** and double-click "01.2_Servo.ino".



Code

```
#include "Freenove_4WD_Car_For_Pico_W.h"
void setup()
{
  Servo_Setup();       //Servo initialization
  Servo_1_Angle(90);   //Set servo 1 Angle
  delay(1000);
}

void loop()
{
  // Servo 1 motion path; 90°- 30°- 150°- 90°
   Servo_Sweep(1, 90, 30);
   Servo_Sweep(1, 30, 150);
   Servo_Sweep(1, 150, 90);
}
```

Code Explanation

Add header file of servos. Each time before controlling servos, please add header file first.

```
#include "Freenove_4WD_Car_For_Pico_W.h"
```

Initialize the servo driven chip.

```
Servo_Setup();       //Servo initialization
```

Set the angle of the two servos to 90°.

```
  Servo_1_Angle(90);   //Set servo 1 Angle
```

Sweep function of the servos. Servo_id stands for the servo number. Angle_start represents the starting position of the servo when sweeping while angle_end is the ending position. Note: the sweeping range of servo1 is 30-150°.

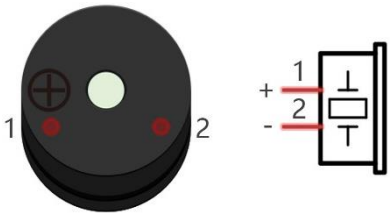
```
void Servo_Sweep(int servo_id, int angle_start, int angle_end);
```

2.3 Buzzer

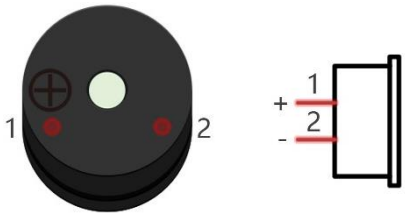
Buzzer

Buzzer is a sounding component, which is widely used in electronic devices such as calculator, electronic warning clock and alarm. Buzzer has two types: active and passive. Active buzzer has oscillator inside, which will sound as long as it is supplied with power. Passive buzzer requires external oscillator signal (generally use PWM with different frequency) to make a sound.

Active buzzer



Passive buzzer



Active buzzer is easy to use. Generally, it can only make a specific frequency of sound. Passive buzzer requires an external circuit to make a sound, but it can be controlled to make a sound with different frequency. The resonant frequency of the passive buzzer is 2kHz, which means the passive buzzer is loudest when its resonant frequency is 2kHz.

How to identify active and passive buzzer

1. Usually, there is a label on the surface of active buzzer covering the vocal hole, but this is not an absolute judgment method.
2. Active buzzers are more complex than passive buzzers in their manufacture. There are many circuits and crystal oscillator elements inside active buzzers; all of this is usually protected with a waterproof coating (and a housing) exposing only its pins from the underside. On the other hand, passive buzzers do not have protective coatings on their underside. From the pin holes viewing of a passive buzzer, you can see the circuit board, coils, and a permanent magnet (all or any combination of these components depending on the model.

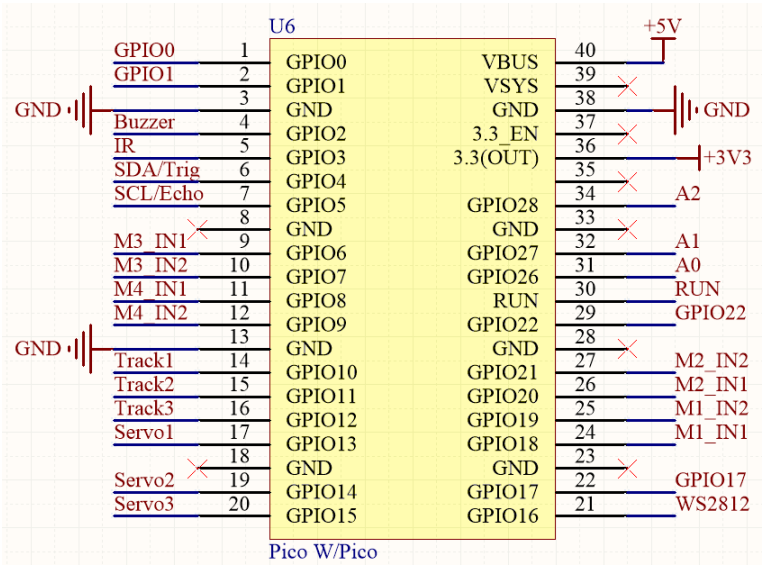
Active buzzer

Passive buzzer

The buzzer used in this car is a passive buzzer that can make sounds with different frequence.

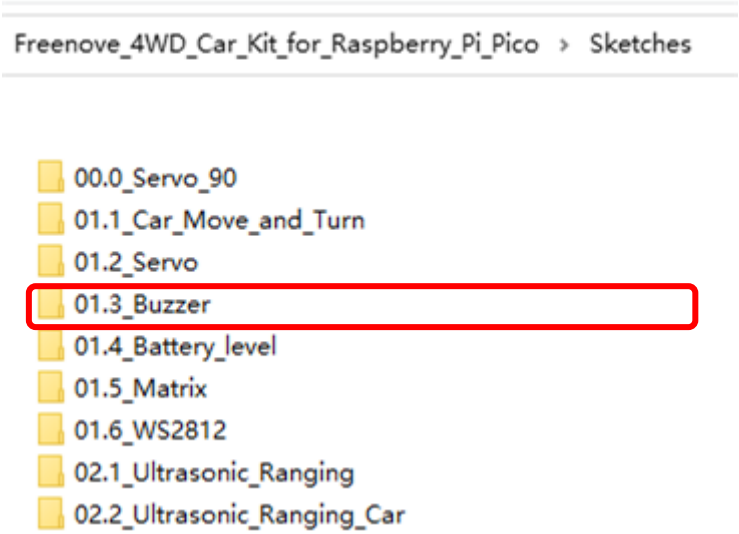
Schematic

As we can see, the buzzer is controlled by GPIO2 of Raspberry Pi Pico W. When the buzzer receives PWM signal, NPN will be activated to make the buzzer sound. When the buzzer receives no signal, it will be controlled at low level by R2 and NPN will not be activated, so the buzzer will not make any sounds.



Sketch

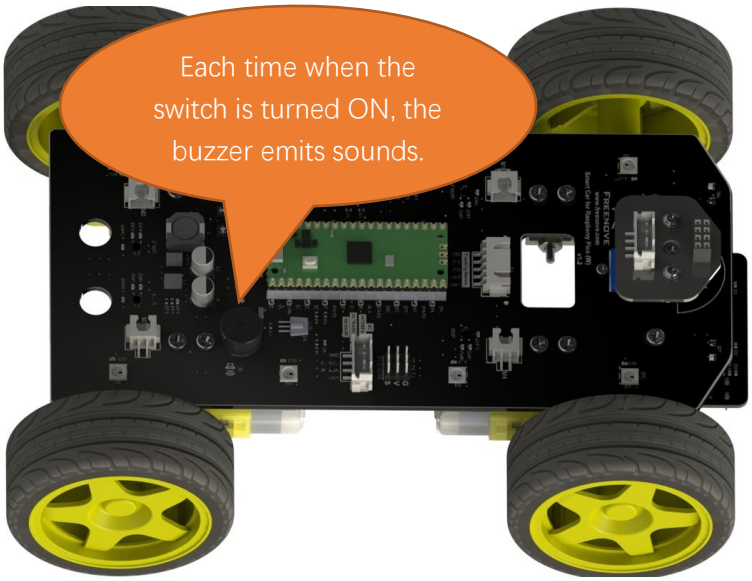
In this section, we will test the buzzer to make it sound like an alarm.
Open "01.3_Buzzer" folder in the **"Freenove_4WD_Car_Kit_for_Raspberry_Pi_Pico\Ordinary_wheels\Sketches"**, and then double-click "01.3_Buzzer.ino".



Code

```
#include "Freenove_4WD_Car_For_Pico_W.h"

void setup() {
  Buzzer_Setup();     //Buzzer initialization function
  Buzzer_Alert(4, 3);//Control the buzzer to sound
}

void loop() {
  delay(1000);
}
```

After the program is downloaded to Raspberry Pi Pico (W), the buzzer emits four short beeps, repeating for three times.



Code Explanation

Configure the PWM of Raspberry Pi Pico (W) to associate it with GPIO2 pin to control the buzzer to make sounds.

```
void Buzzer_Setup(void);                    //Buzzer initialization
```

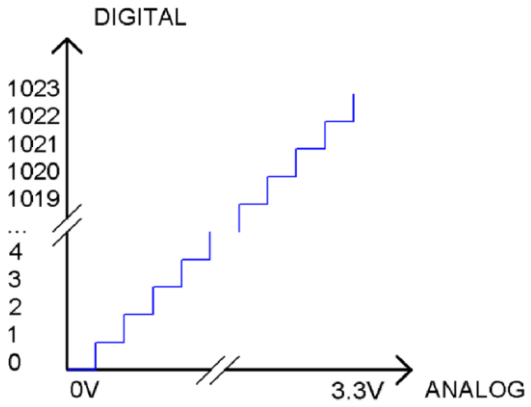
Control the buzzer to sound regularly. Parameter beat represents the number of times the buzzer sounds in each sounding cycle and rebeat represents how many cycles the buzzer sounds.

```
void Buzzer_Alert(int beat, int rebeat);//Buzzer alarm function
```

2.4 ADC Module

ADC

ADC is an electronic integrated circuit used to convert analog signals such as voltages to digital or binary form consisting of 1s and 0s. The range of our ADC on Raspberry Pi Pico (W) is 10 bits, that means the resolution is 2^10=1024, and it represents a range (at 3.3V) will be divided equally to 1024 parts. The rage of analog values corresponds to ADC values. So the more bits the ADC has, the denser the partition of analog will be and the greater the precision of the resulting conversion.



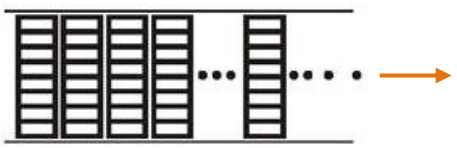
Subsection 1: the analog in rang of 0V---3.3/1023 V corresponds to digital 0;
Subsection 2: the analog in rang of 3.3/1023 V---2*3.3 /1023V corresponds to digital 1;
The following analog will be divided accordingly.
The conversion formula is as follows:

$$ADC\ Value = \frac{\text{Analog Voltage}}{3.3} * 1023$$

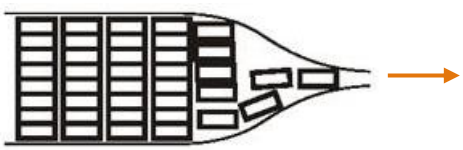
Serial Communication

Serial communication uses one data cable to transfer data one bit by another in turn. Parallel communication means that the data is transmitted simultaneously on multiple cables. Serial communication takes only a few cables to exchange information between systems, which is especially suitable for computers to computers, long distance communication between computers and peripherals. Parallel communication is faster, but it requires more cables and higher cost, so it is not appropriate for long distance communication.

Parallel communication

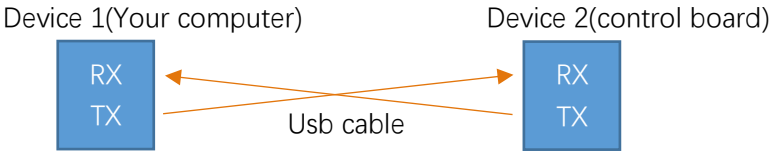


Serial communication



Serial communication generally refers to the Universal Asynchronous Receiver/Transmitter (UART), which is commonly used in electronic circuit communication. It has two communication lines; one is responsible for

sending data (TX line) and the other for receiving data (RX line). The serial communication connections of two devices use is as follows:

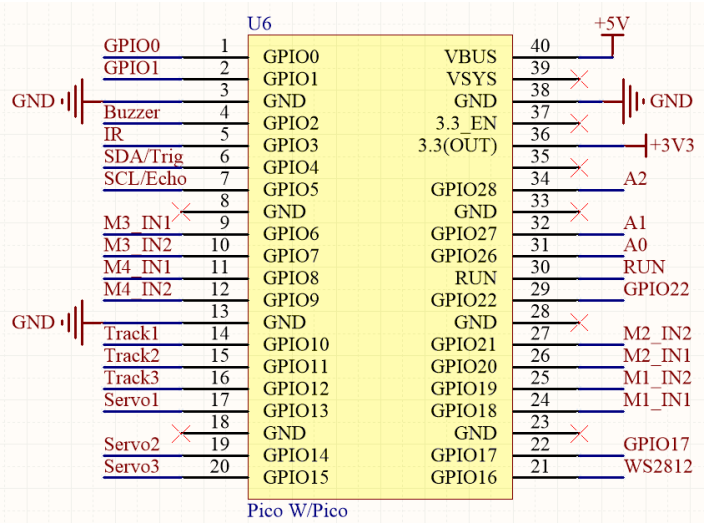


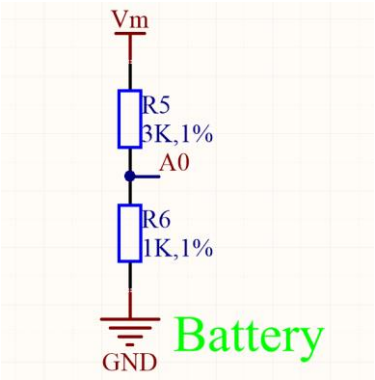
For serial communication, the **baud rate in both sides must be the same**. The baud rates commonly used are 9600 and 115200.

Computer identifies serial devices connected to your computer as COMx. We can use the Serial Monitor window of Arduino Software to communicate with Freenove control board.

Schematic

As we can see, the car reads the voltage of the batteries through GPIO26 of Raspberry Pi Pico (W).

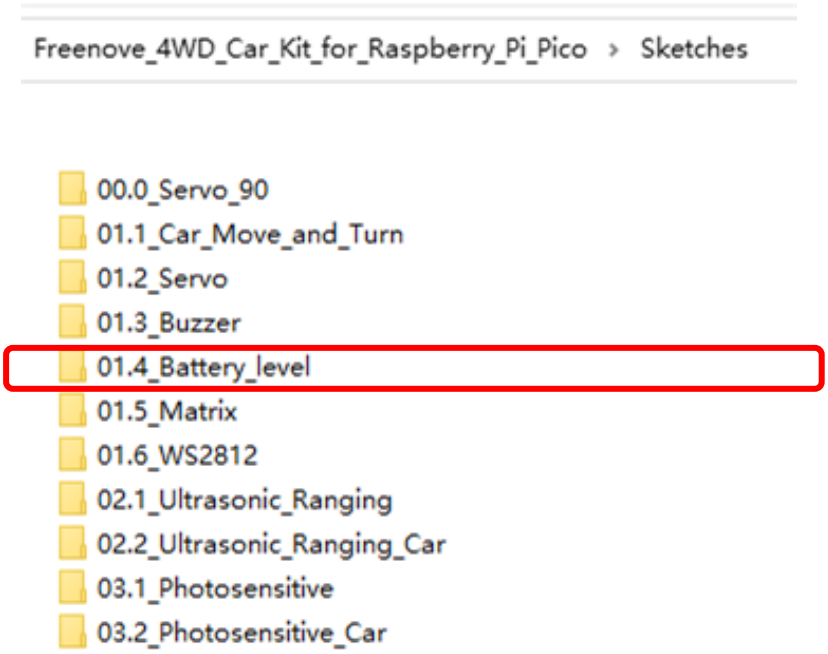




The voltage acquisition range of GPIO26 on Raspberry Pi Pico W is 0-3.3V, while the car is powered by two 18650-lithium batteries, whose voltage is 8.4V when they are fully charged, which exceeds the acquisition range of Raspberry Pi Pico (W). Therefore, after passing through the voltage divider circuit composed of R3 and R4, the voltage at A0 will be about 1/4 of the battery voltage, 8.4/4=2.1V, which is within the voltage collection range of GPIO26.

Sketch

In this section, we will use GPIO26 of Raspberry Pi Pico (W) to read the voltage value of the batteries and print it on serial monitor. Open "01.4_Battery_level" folder in **"Freenove_4WD_Car_Kit_for_Raspberry_Pi_Pico\Ordinary_wheels\Sketches"** and then double-click "01.4_Battery_level.ino".



Code

```
#include "Freenove_4WD_Car_For_Pico_W.h"

void setup() {
  Serial.begin(115200);                     //Set the Serial Baud rate
}
void loop() {
  Serial.print("Battery ADC : ");
  Serial.println(Get_Battery_Voltage_ADC());//Gets the battery ADC value
  Serial.print("Battery Voltage : ");
  Serial.print(Get_Battery_Voltage());      //Get the battery voltage value
  Serial.println("V");
  delay(300);
}
```

Code Explanation

Activate the serial port and set the baud rate to 115200.

```
  Serial.begin(115200);                     //Set the Serial Baud rate
```

Get ADC sampling value of GPIO26 and return it. The ADC has a range of 0-1023. The voltage range collected is 0-3.3V.

```
  int Get_Battery_Voltage_ADC(void);    //Gets the battery ADC value
```

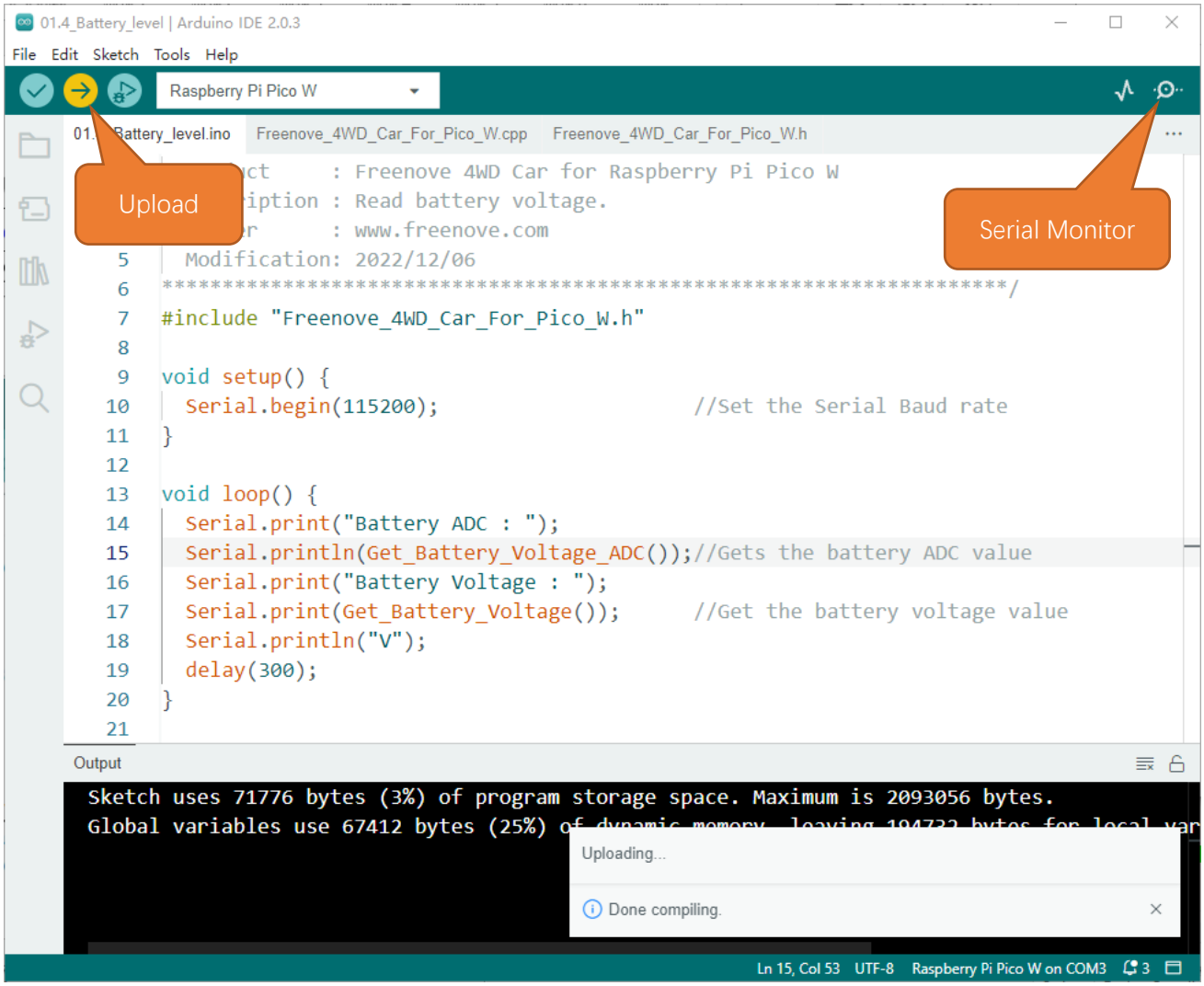
Calculate the voltage of batteris and return it.

```
float Get_Battery_Voltage(void);      //Get the battery voltage value
```

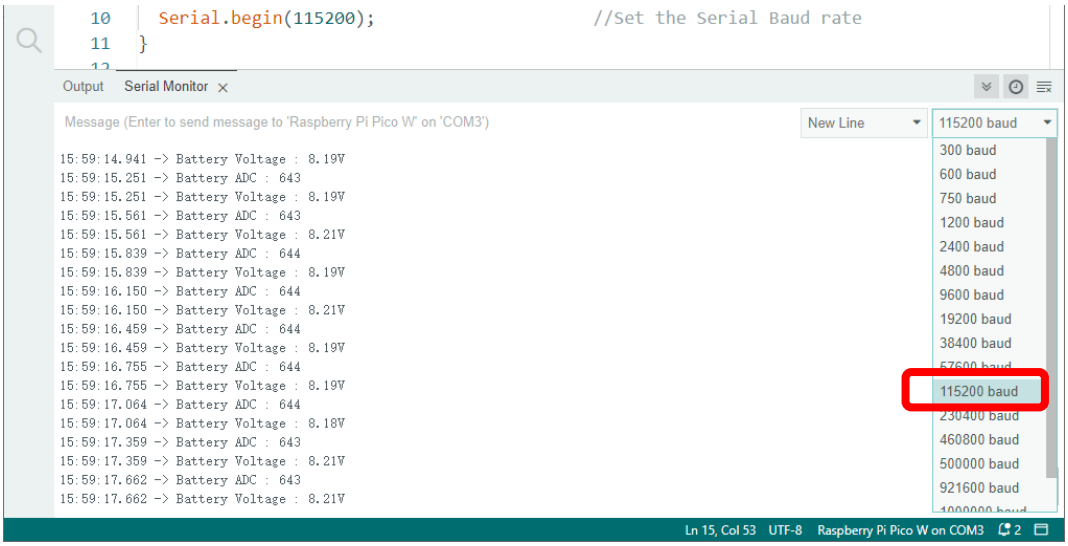
The default battery voltage coefficient is 3. Users can modify it by calling this function.

```
void Set_Battery_Coefficient(float coefficient);//Set the partial pressure coefficient
```

Click "Upload" to upload the code to Pico (W). After uploading successfully, click Serial Monitor



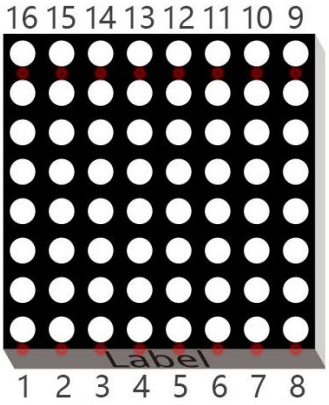
Set baud rate to 115200.



2.5 LED Matrix

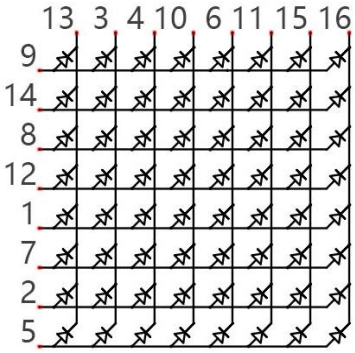
LED Matrix

A LED matrix is a rectangular display module that consists of a uniform grid of LEDs. The following is an 8X8 monochrome LED matrix containing 64 LEDs (8 rows by 8 columns).

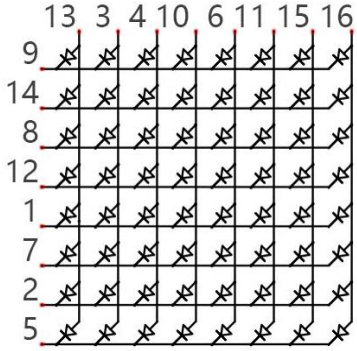


In order to facilitate the operation and reduce the number of ports required to drive this component, the positive poles of the LEDs in each row and negative poles of the LEDs in each column are respectively connected together inside the LED matrix module, which is called a common anode. There is another arrangement type. Negative poles of the LEDs in each row and the positive poles of the LEDs in each column are respectively connected together, which is called a common cathode.

Connection mode of common anode

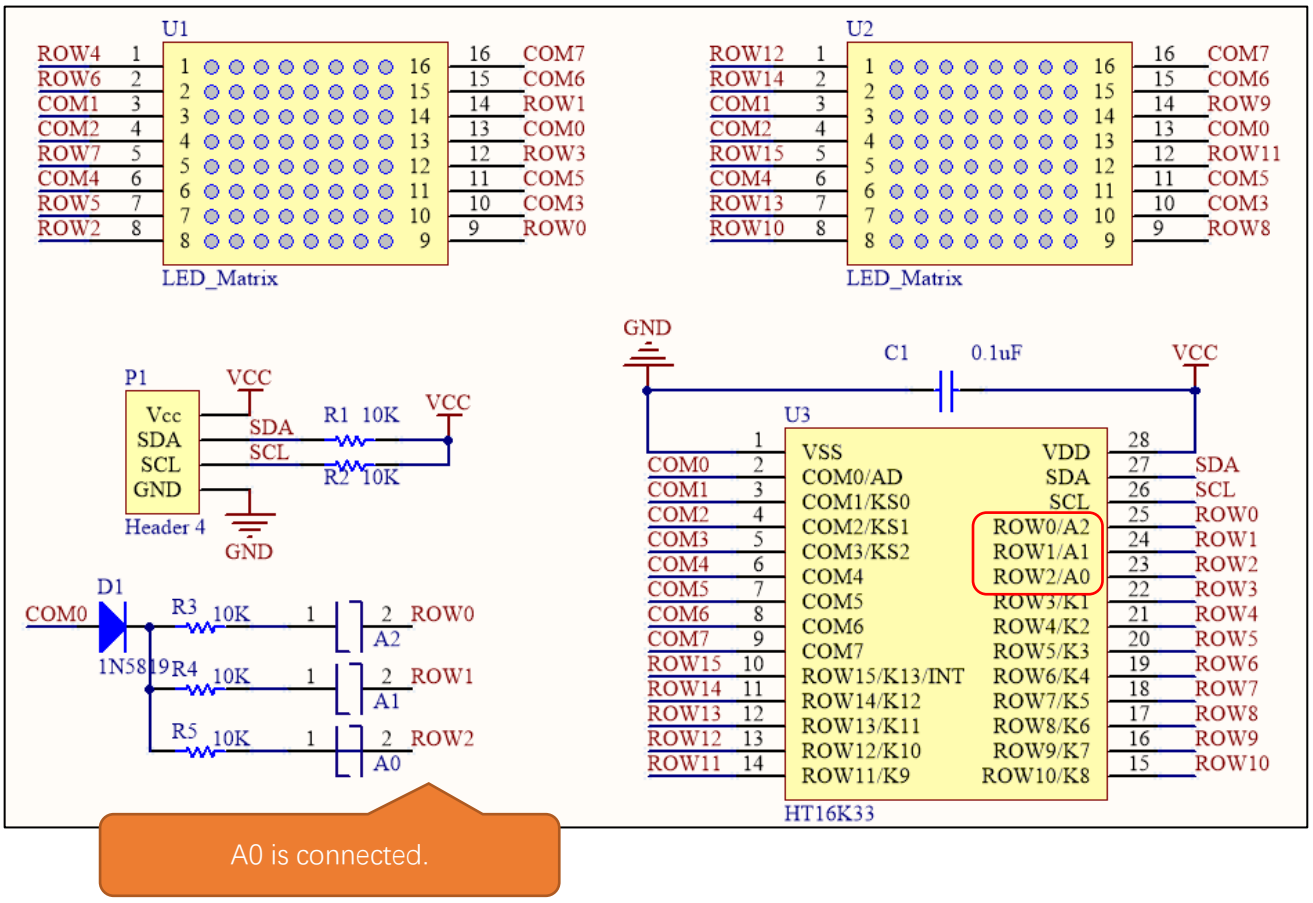


Connection mode of common cathode

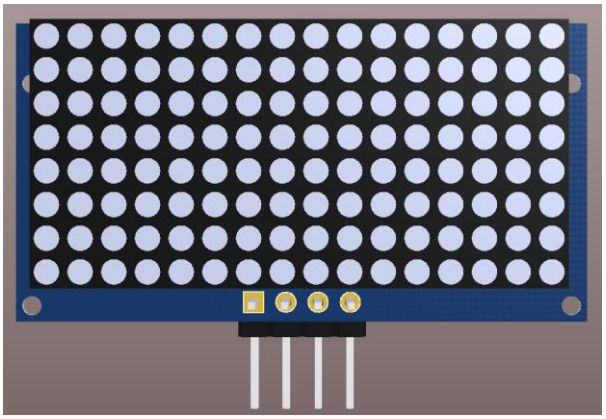
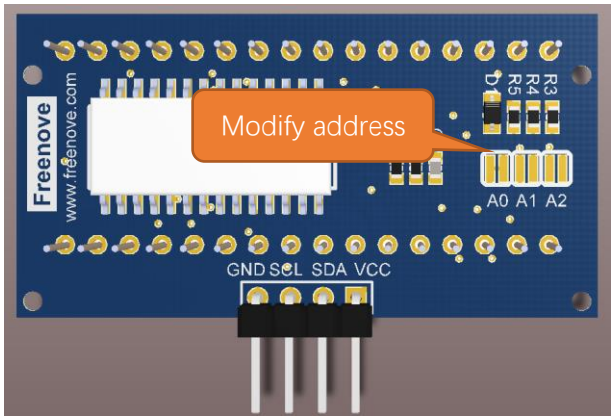


Schematic

For this tutorial, the LED matrix module is individual and it is driven by IIC chip.



The LED matrix is common anode. As we can see from the schematic above, the anode of LED matrix is connected to ROWx of HT16K33 chip, and the cathode is connected to COMx. The address of HT16K33 chip is (0x70+[A2:A0]), and the default address of LED matrix is 0x71. If you want to change the address, you can use a knife to cut the connecting line in the middle of A0, or connect A1/A2.

Front of LED Matrix	Back of LED Matrix
	

We divide the LED matrix into two sides and display "+" on the left and "o" on the right. As shown below, yellow stands for lit LED while other colors reprensent the OFF LED.

1															
2				1	1						1	1			
3				1	1					1			1		
4		1	1	1	1	1	1			1				1	
5		1	1	1	1	1	1			1				1	
6				1	1						1			1	
7				1	1							1	1		
8															

Below, the table on the left corresponds to the "+" above, and the table on the right corresponds to the "o" above.

Row	Binary	Hexadecimal
1	0000 0000	0x00
2	0001 1000	0x18
3	0001 1000	0x18
4	0111 1110	0x7e
5	0111 1110	0x7e
6	0001 1000	0x18
7	0001 1000	0x18
8	0000 0000	0x00

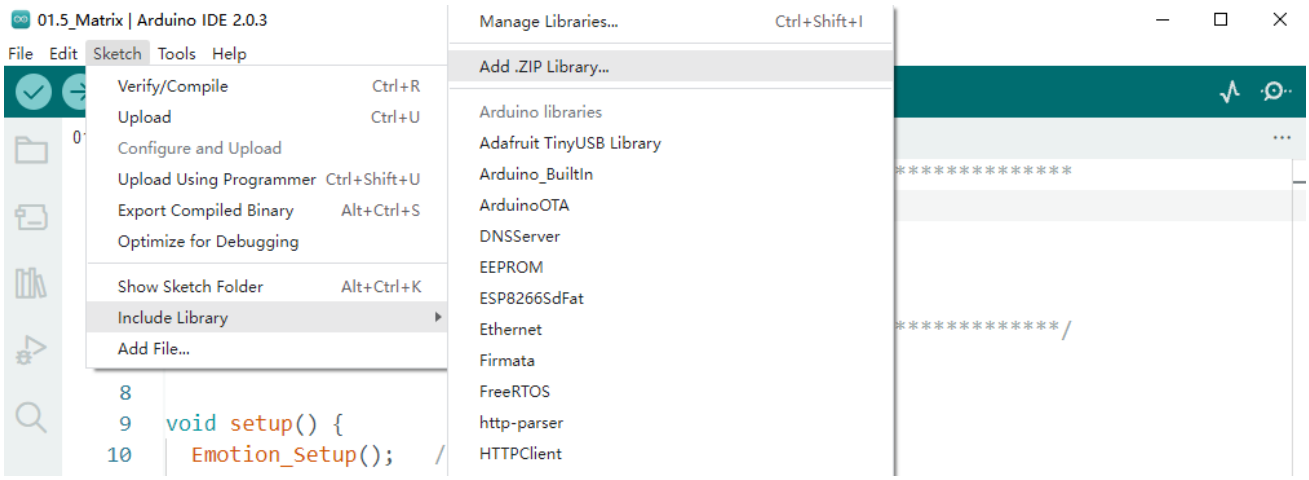
Row	Binary	Hexadecimal
1	0000 0000	0x00
2	0001 1000	0x18
3	0010 0100	0x24
4	0100 0010	0x42
5	0100 0010	0x42
6	0010 0100	0x24
7	0001 1000	0x18
8	0000 0000	0x00

Sketch

The LED matrix is controlled by HT16K33 chip. Therefore, before opening the program, we need to install Freenove_VK16K33_Lib library in advance.

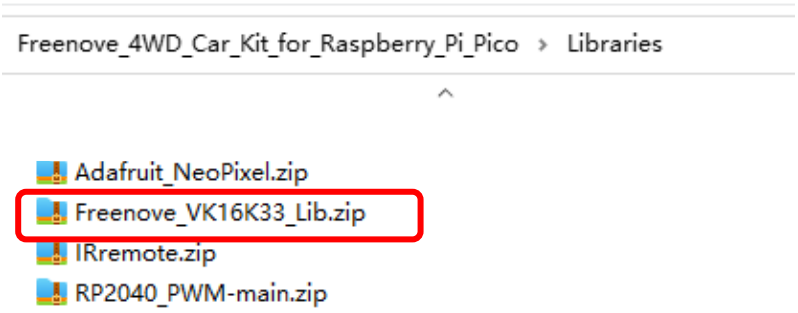
Install Freenove_VK16K33_Lib Library

Click Sketch and select Add .ZIP Library in Include.



Select "Freenove_VK16K33_Lib.zip" in the folder Libraries of the folder

"Freenove_4WD_Car_Kit_for_Raspberry_Pi_Pico".



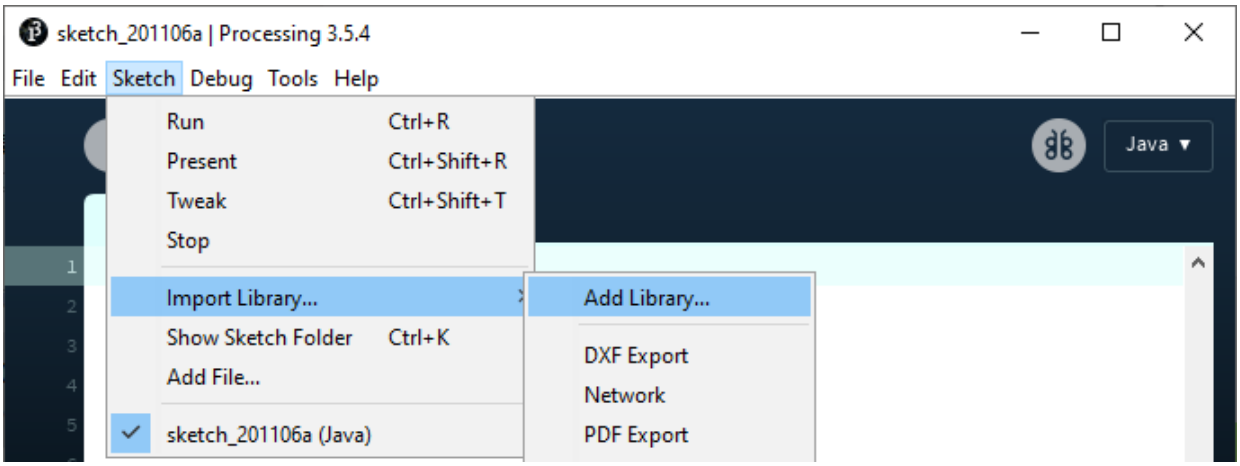
Install Processing

In this tutorial, we use Processing to build a simple Led Matrix platform.

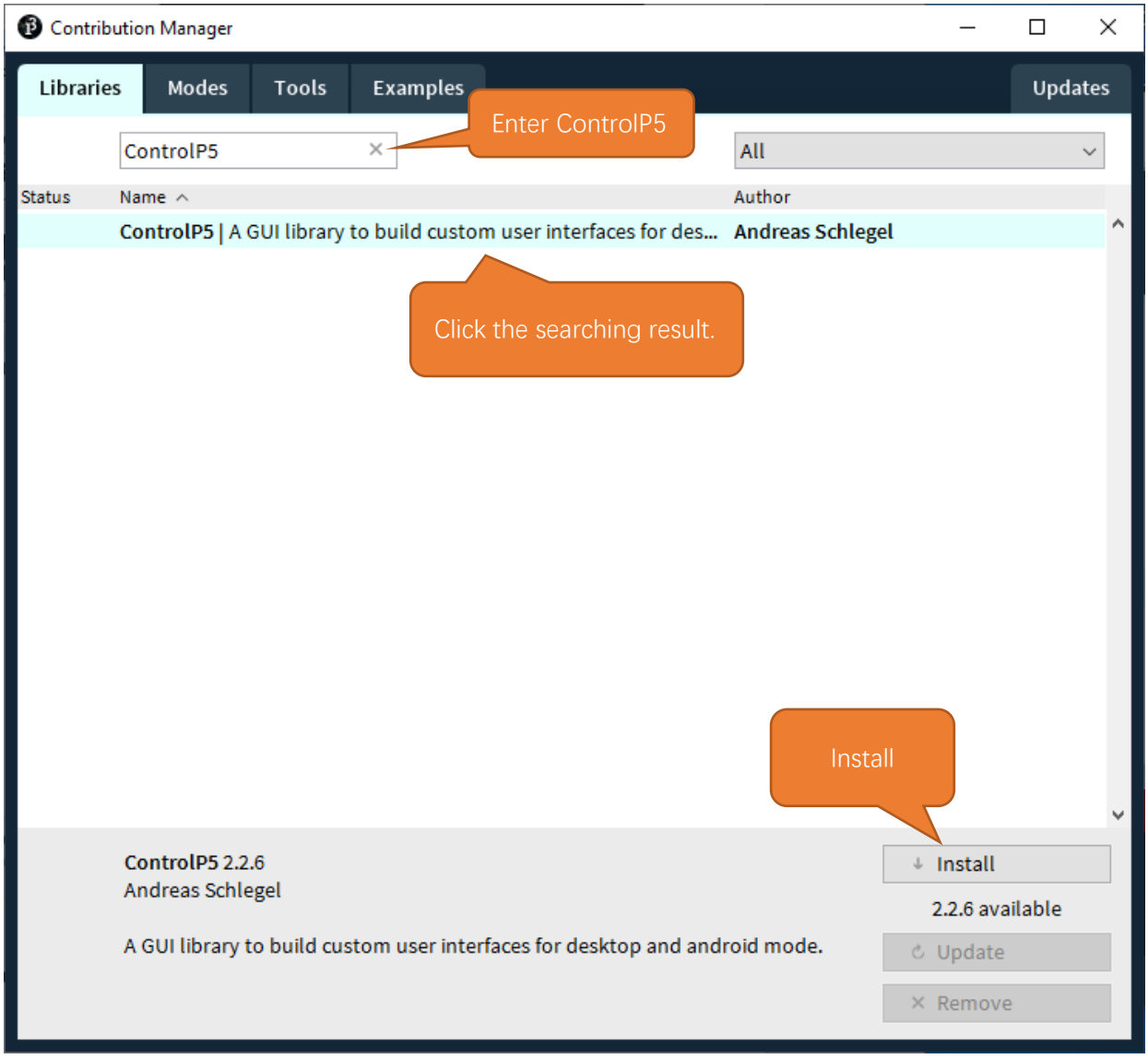
If you have not installed Processing, you can download it by clicking https://processing.org/download/. You can choose an appropriate version to download according to your PC system.

Unzip the downloaded file to your computer. Click "processing.exe" as the figure below to run this software.

In the interface of Processing, click Sketch on Menu bar, select “Import Library...”and then click “Add Library...”.

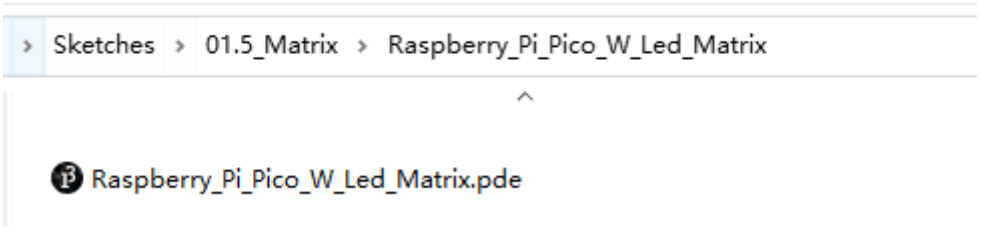


Enter “ControlP5” in the input field of the pop-up window. Click the searching result and then click “install”

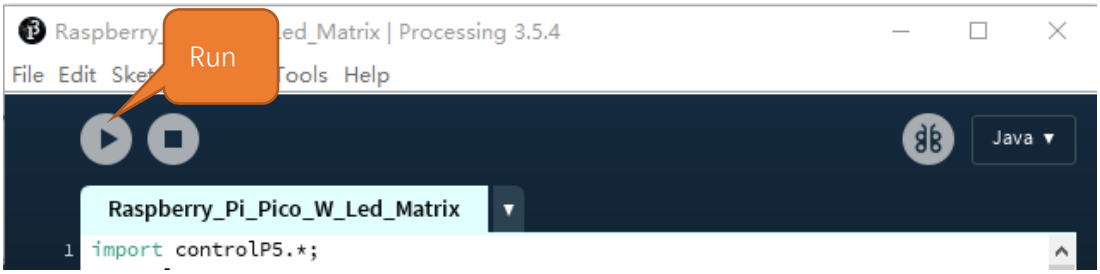


When the installation finishes, restart Processing.

Open the folder Raspberry_Pi_Pico_W_Led_Matrix in 01.5_Matrix of the **"Freenove_4WD_Car_Kit_for_Raspberry_Pi_Pico\Ordinary_wheels\Sketches"**. Here we take Windows as an example. Click to open Raspberry_Pi_Pico_W_Led_Matrix.pde.

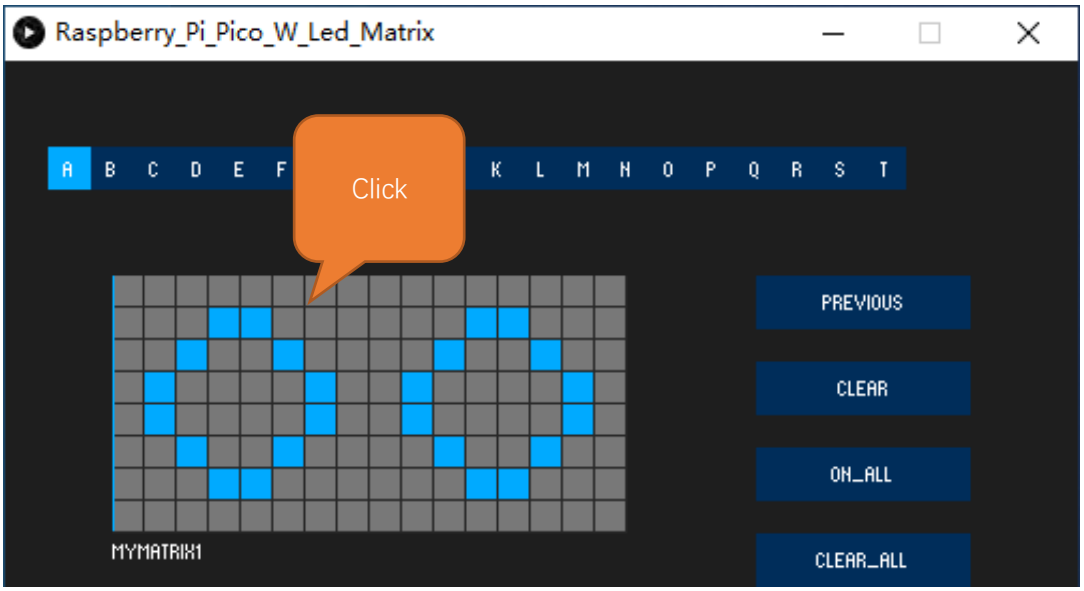


Click "Run"

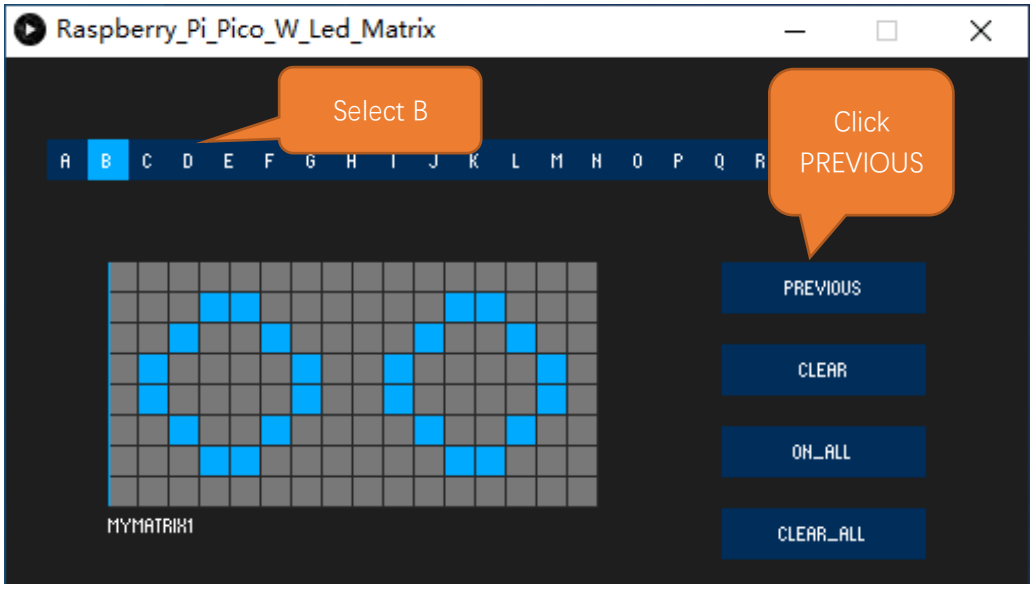


There are 20 pages from A to T. Select Page A.

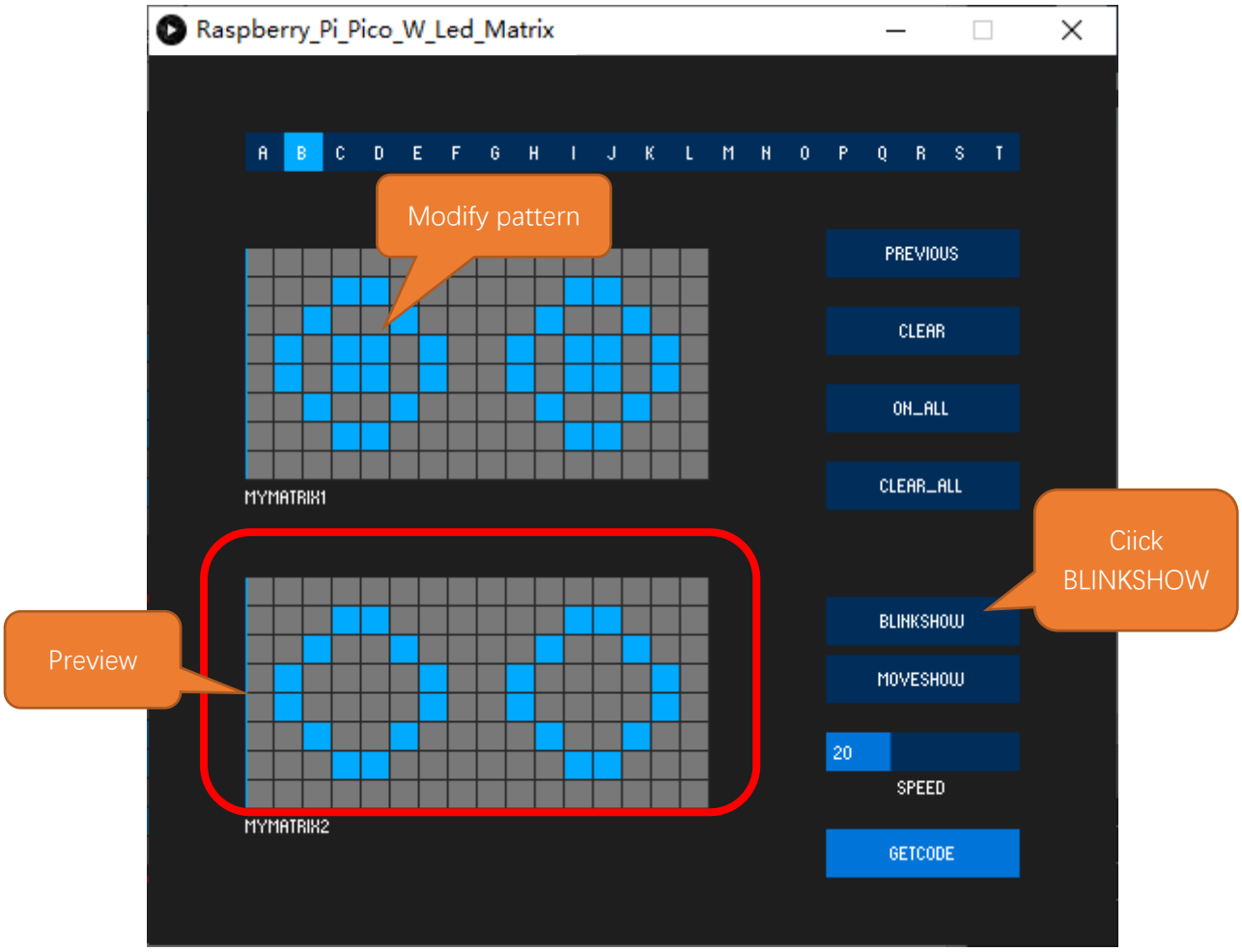
You can design your own pattern by clicking on the squares.



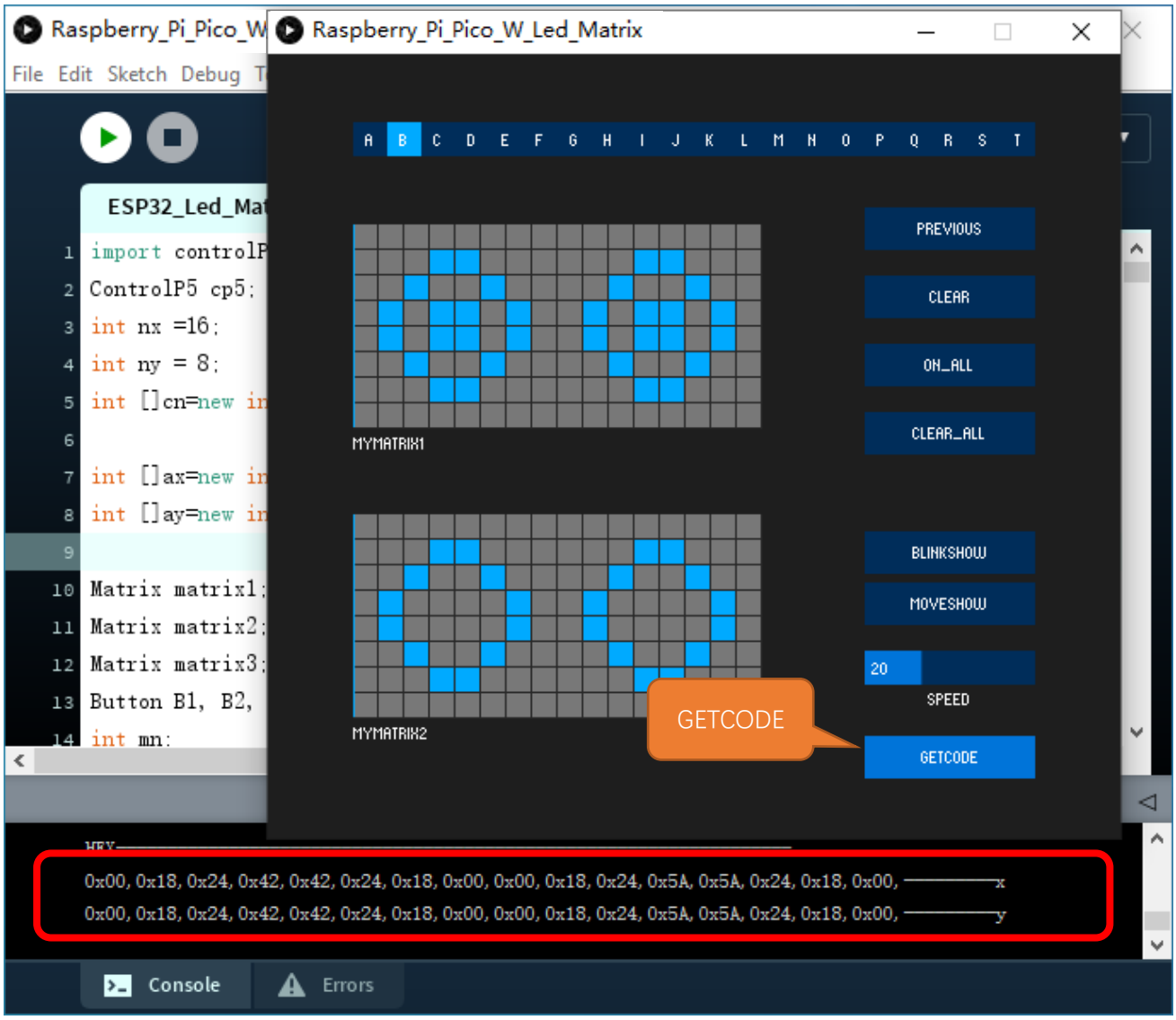
Next select Page B and click PREVIOUS, which copies the previous pattern to B.



Click the squares to modify the pattern, and then click BLINKSHOW, you can browse the overlay effect of different pages.



Cick GETCODE to generate array.



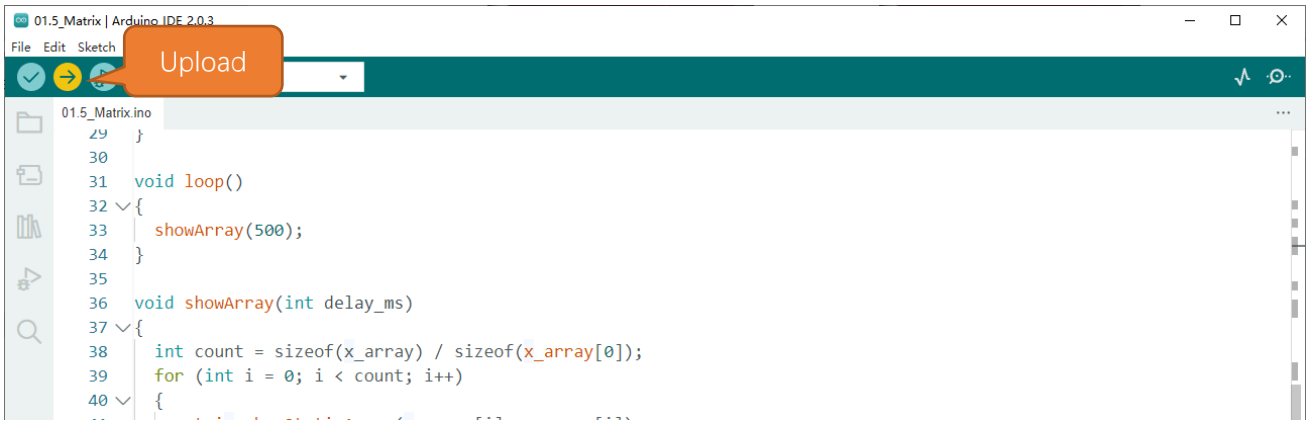
The data on the left of the LED matrix are stored together and end with "----x", and the data on the right are stored together and end with "----y". Copy these two sets of dot matrix data and replace the array content in "01.5_Matrix.ino".

Open the folder “01.5_Matrix” in the **“Freenove_4WD_Car_Kit_for_Raspberry_Pi_Pico\Ordinary_wheels\Sketches”** and double click “01.5_Matrix.ino”

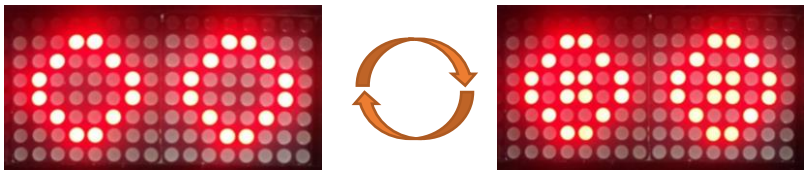
Code

```
#include "Freenove_VK16K33_Lib.h"

Freenove_VK16K33 matrix = Freenove_VK16K33();

byte x_array[][8] = {//Put the data into the left LED matrix
  ////////////////////////////////////////////
  0x00, 0x18, 0x24, 0x42, 0x42, 0x24, 0x18, 0x00,
  0x00, 0x18, 0x24, 0x5A, 0x5A, 0x24, 0x18, 0x00,
  ////////////////////////////////////////////
};

byte y_array[][8] = {//Put the data into the right LED matrix
  ////////////////////////////////////////////
  0x00, 0x18, 0x24, 0x42, 0x42, 0x24, 0x18, 0x00,
  0x00, 0x18, 0x24, 0x5A, 0x5A, 0x24, 0x18, 0x00,
  ////////////////////////////////////////////////
};

void setup()
{
  matrix.init(0x71);
  matrix.setBlink(VK16K33_BLINK_OFF);
}

void loop()
{
  showArray(500);
}

void showArray(int delay_ms)
{
  int count = sizeof(x_array) / sizeof(x_array[0]);
  for (int i = 0; i < count; i++)
  {
    matrix.showStaticArray(x_array[i], y_array[i]);
    delay(delay_ms);
  }
}
```


Copy and paste the array generated by the auxiliary applet to the program, and then click upload.



You can see the LED matrix keep blinking.



Code Explanation

Add the header file of LED matrix. Each time before controlling LED matrix, please add its header file first.

```
1  #include "Freenove_VK16K33_Lib.h"
```

Apply for an Freenove_VK16K33 object and name it matrix.

```
3  Freenove_VK16K33 matrix = Freenove_VK16K33();
```

Define IIC address and IIC pins of HT16K33 chip. Call init() function to initialize it and call setBlink() to set the LED matrix not blink.

```
3  Freenove_VK16K33 matrix = Freenove_VK16K33();
...  ...
21   matrix.init(0x71);
22   matrix.setBlink(VK16K33_BLINK_OFF);
```

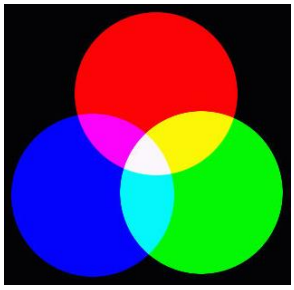
Define count to calculate the number of one-dimensional arrays contained in the two-dimensional x_array, and use the for loop to call the showStaticArray() function to continuously display the content of LED matrix.

```
32   int count = sizeof(x_array) / sizeof(x_array[0]);
33   for (int i = 0; i < count; i++)
34   {
35     matrix.showStaticArray(x_array[i], y_array[i]);
36     delay(delay_ms);
37   }
```

2.6 LED

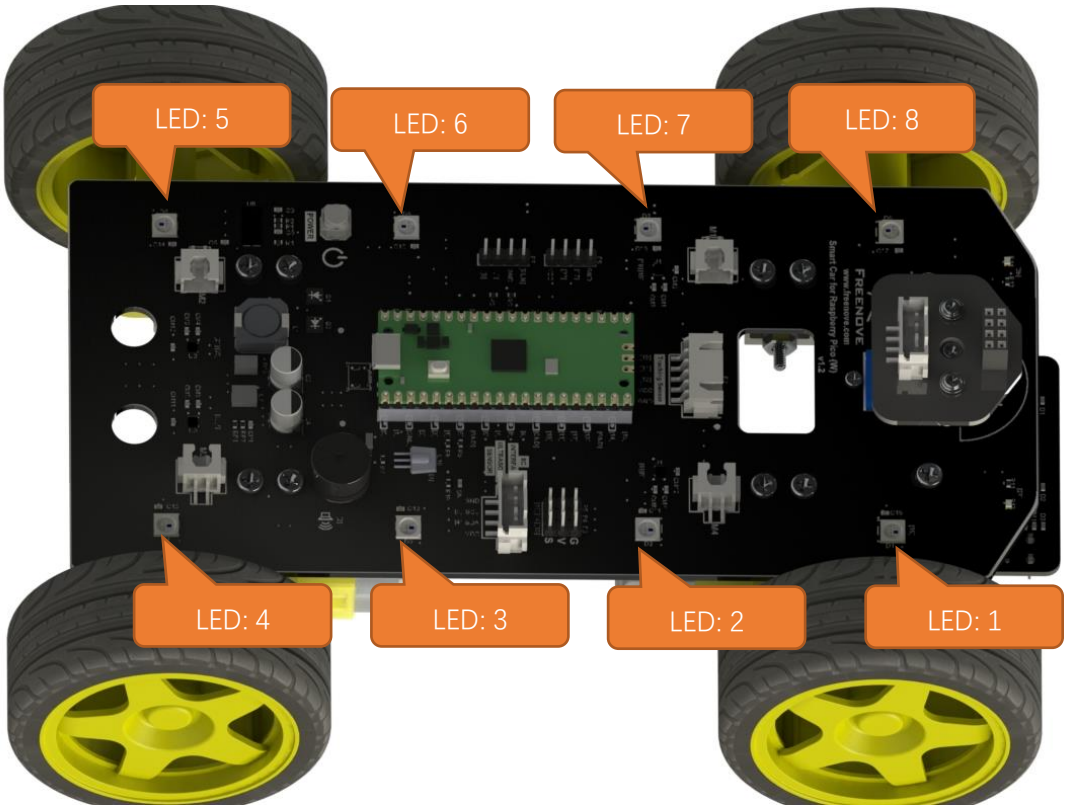
LED

Red, green, and blue are called the three primary colors. When you combine these three primary colors of different brightness, it can produce almost all kinds of visible light.



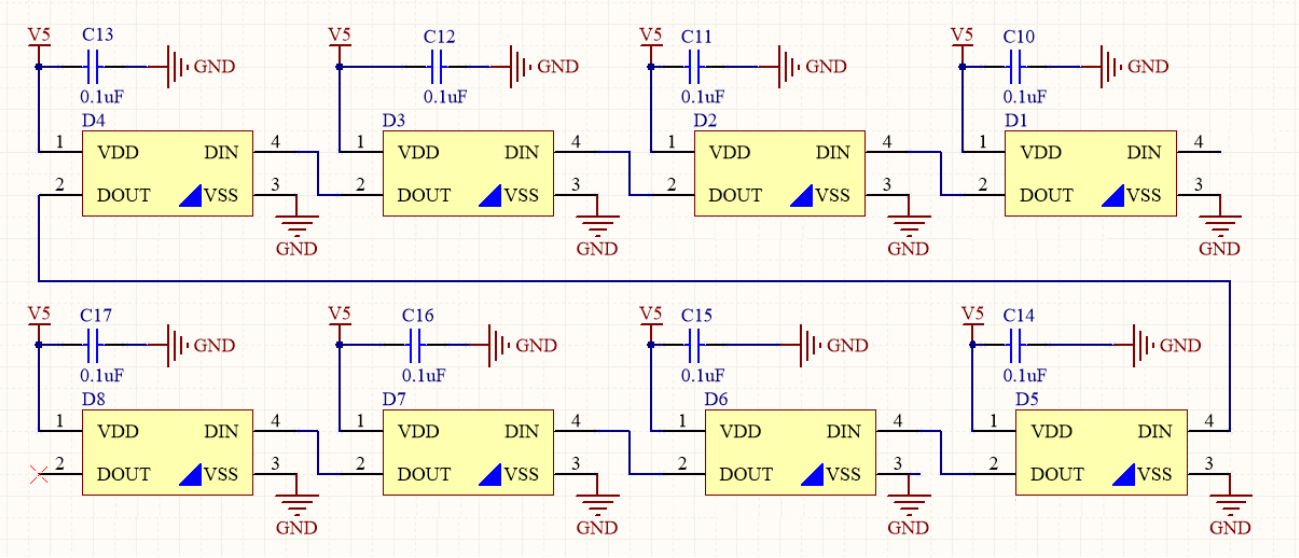
RGB

The LED of the car is composed of eight LED, each of which is controlled by one pin and supports cascading. Each LED can emit three basic colors of red, green and blue, and supports 256-level brightness adjustment, which means that each LED can emit 2^24=16,777,216 different colors.



Schematic

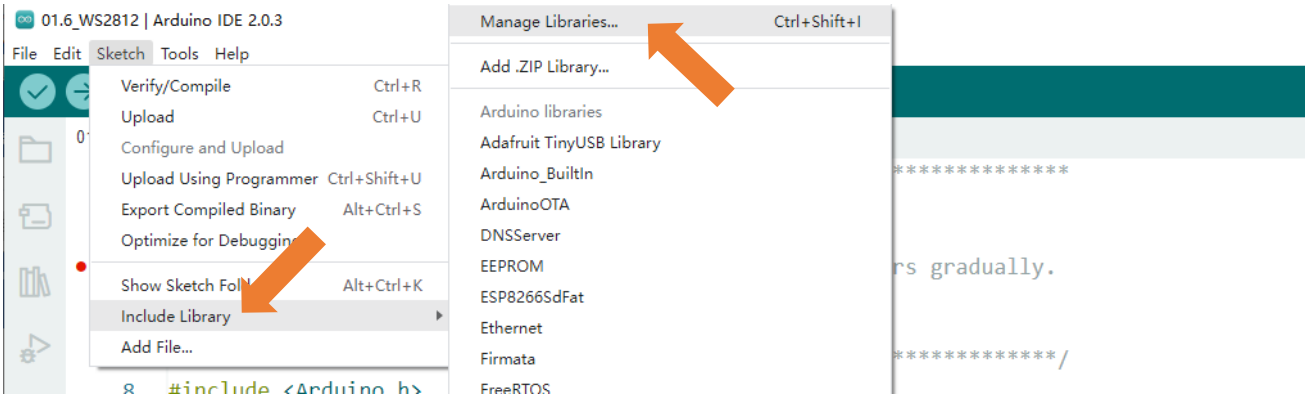
As shown below, the DOUT of each LED is connected with DIN of the next LED, and the 8 LED can be controlled to emit colorful colors by inputting control signals through LED.



Sketch

Before programming, please make sure the LED driver library has been installed. If not, please install it as follows.

Open Arduino IDE, select Sketch on Menu bar, navigate the mouse to Include library and click Manage Libraries.



Enter "**Adafruit_NeoPixel**" in the input field of the pop-up window, find it and then click Install.

Wait for the installation to finish.

Next, we will download the code to Raspberry Pi Pico W to test the LED. Open the folder "01.6_WS2812" in the **"Freenove_4WD_Car_Kit_for_Raspberry_Pi_Pico\Ordinary_wheels\Sketches"** and then double click "01.6_WS2812.ino".

Code

```
#include <Arduino.h>
#include <Adafruit_NeoPixel.h>
#include "Freenove_4WD_Car_WS2812.h"

void setup() {
 WS2812_Setup();      //WS2812 initialization
}

void loop() {
  WS2812_Show(5);
}
```

Another part of the code

```
#define WS2812_PIN 16        //Control pins for Pico W
#define LEDS_COUNT 8
Adafruit_NeoPixel ws2812_strip(LEDS_COUNT, WS2812_PIN, NEO_GRB + NEO_KHZ800);
//WS2812 initialization function
void WS2812_Setup(void)
{
  ws2812_strip.begin();
  ws2812_strip.setBrightness(20);
  ws2812_close();
}
//WS2812 non-blocking display function
void WS2812_Show(int mode)
{
  switch (mode)
  {
    case 0://Close the WS2812
      ws2812_close();
      break;
    case 1:
      ws2812_rgb();
      break;
    case 2:
      ws2812_following();
      break;
    case 3:
      ws2812_blink();
      break;
    case 4:
      ws2812_breathe();
      break;
    case 5:
```

```
32          ws2812_rainbow();
33          break;
34        default:
35          break;
36      }
37  }
38  int rainbow_count = 0;
39  //WS2812 rainbow display
40  void ws2812_rainbow(void)
41  {
42    ws2812_strip.setBrightness(20);
43    ws2812_strip_time_next = millis();
44    if (ws2812_strip_time_next - ws2812_strip_time_now > 5)
45    {
46      ws2812_strip_time_now = ws2812_strip_time_next;
47      rainbow_count++;
48      for (int i = 0; i < 12; i++)
49        ws2812_strip.setPixelColor(11 - i, Wheel((i * 256 / 12 + rainbow_count) & 255));
50      ws2812_strip.show();
51      if (rainbow_count > 255)
52        rainbow_count = 0;
53    }
54  }
```

Download the code to the Raspberry Pi Pico W, turn ON the power switch and the LED on the car will emit lights like rainbow.

Code Explanation:

Add the header file of LED. Each time before controlling LED, please add its header file.

```
2   #include <Adafruit_NeoPixel.h>
```

Set the number of LED, define the control pin and channel. Instantiate a LED object and name it strip.

```
#define WS2812_PIN 16         //Control pins for Pico W
#define LEDS_COUNT 8
Adafruit_NeoPixel ws2812_strip(LEDS_COUNT, WS2812_PIN, NEO_GRB + NEO_KHZ800);
```

Initialize LED, set their brightness to be 10. The range of brightness is 0-255.

```
  ws2812_strip.begin();            //Initialize LED
  ws2812_strip.setBrightness(10);  //Set the brightness of LED
```

Set the color of LED. Note: The color will not be displayed immediately after it is set.

```
ws2812_strip.setPixelColor(11 - i, Wheel((i * 256 / 12 + rainbow_count) & 255));
```

Display the color of LED. After setting the color, you need to call show function to display it.

```
ws2812_strip.show();
```

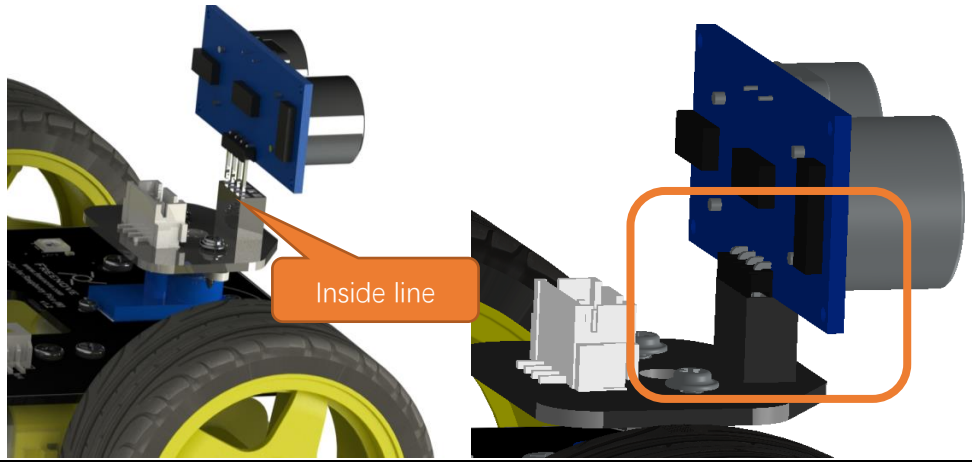
Chapter 3 Ultrasonic Obstacle Avoidance Car

This chapter requires replacing the LED matrix with the ultrasonic module. Please follow the tutorial to replace it first. In this tutorial, only chapters 3, 6, and 7 apply the ultrasonic module. When work on other chapters, please change to use the LED matrix.

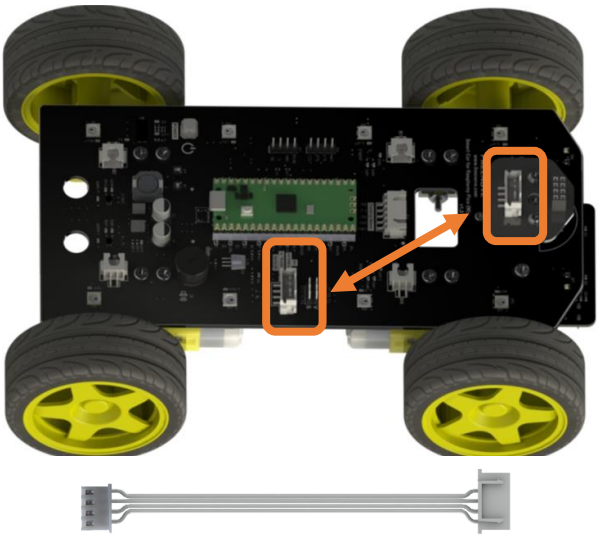
3.1 Ultrasonic Module

Replace the ultrasonic module.

Step 1 Installing the ultrasonic module.



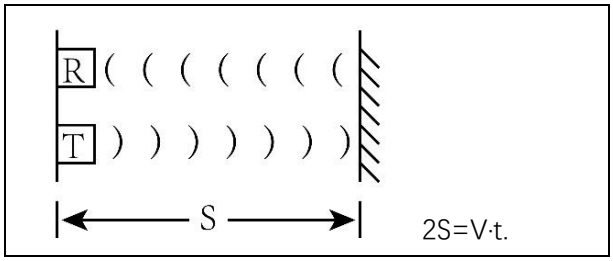
Use cable to connect the two connectors marked below.



Module	Gnd	Echo	Trig	Vcc
Car	GND	ECHO	TRIG	VCC
	GND	5	4	VCC

Ultrasonic Module

The ultrasonic ranging module uses the principle that ultrasonic waves will be sent back when encounter obstacles. We can measure the distance by counting the time interval between sending and receiving of the ultrasonic waves, and the time difference is the total time of the ultrasonic wave's journey from being transmitted to being received. Because the speed of sound in air is a constant, about v=340m/s, we can calculate the distance between the ultrasonic ranging module and the obstacle: s=vt/2.



The ultrasonic ranging module integrates both an ultrasonic transmitter and a receiver. The transmitter is used to convert electrical signals (electrical energy) into high frequency (beyond human hearing) sound waves (mechanical energy) and the function of the receiver is opposite of this. The picture and the diagram of the ultrasonic ranging module are shown below:

Pin description:

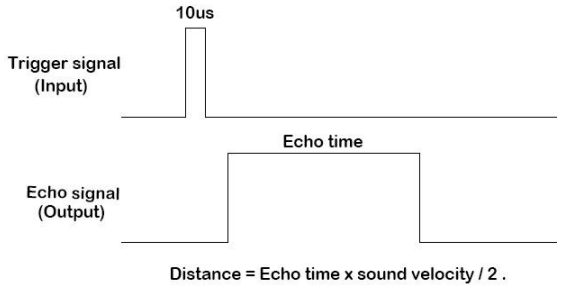
Pin	Description
VCC	Power supply pin
Trig	Trigger pin
Echo	Echo pin
GND	GND

Technical specs:

Working voltage: 5V Working current: 12mA

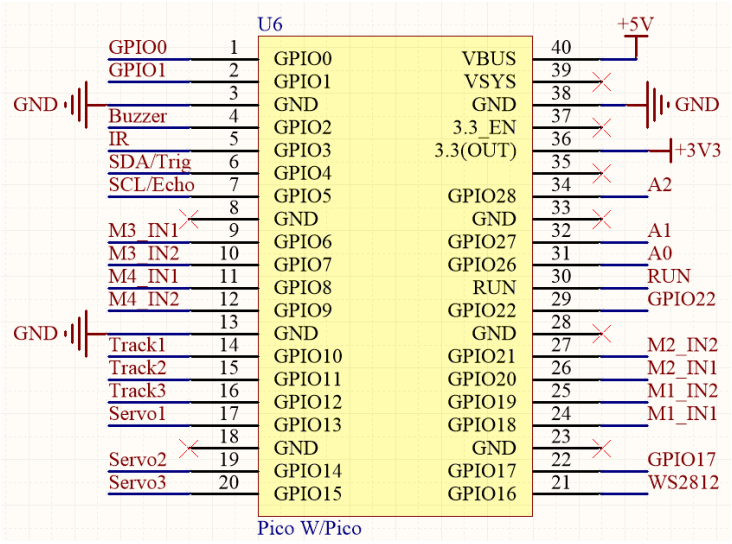
Minimum measured distance: 2cm Maximum measured distance: 200cm

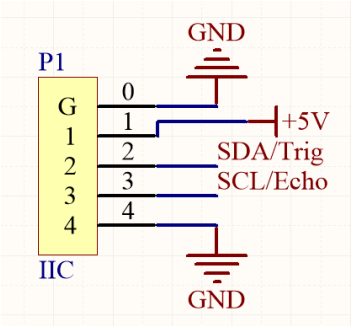
Instructions for use: output a high-level pulse in Trig pin lasting for least 10us, the module begins to transmit ultrasonic waves. At the same time, the Echo pin is pulled up. When the module receives the returned ultrasonic waves from encountering an obstacle, the Echo pin will be pulled down. The duration of high level in the Echo pin is the total time of the ultrasonic wave from transmitting to receiving, s=vt/2.



Schematic

The ultrasonic module is plugged in the front of the car and is connected to the Raspberry Pi Pico (W) development board by means of wiring. As can be seen from the figure below, Raspberry Pi Pico (W) uses GPIO4 and GPIO5 to control the Trig and Echo pins of the ultrasonic module.





Sketch

When the power of the car is turned ON, every module will be initialized and the servo motor will rotate to 90°. The ultrasonic data will be obtained and printed through serial port as the servo motor rotates.

Open the folder "02.1_Ultrasonic_Ranging" in **"Freenove_4WD_Car_Kit_for_Raspberry_Pi_Pico\Ordinary_wheels\Sketches"** and double click "02.1_Ultrasonic_Ranging.ino"

Code

```
#include <Arduino.h>
#include "Freenove_4WD_Car_For_Pico_W.h"

void setup() {
  Serial.begin(115200);//Open the serial port and set the baud rate to 115200
  Ultrasonic_Setup();  //Ultrasonic module initialization
  Servo_Setup();       //Servo initialization
  Servo_1_Angle(90);   //Set the initial value of Servo 1 to 90 degrees
  delay(500);          //Wait for the servo to arrive at the specified location
}

void loop() {
  Servo_1_Angle(150);  //Turn servo 1 to 150 degrees
  Serial.print("Distance: " + String(Get_Sonar()) + "\n");//Print ultrasonic distance
  delay(500);
```

```
16
17    Servo_1_Angle(90);    //Turn servo 1 to 90 degrees
18    Serial.print("Distance: " + String(Get_Sonar()) + "\n");//Print ultrasonic distance
19    delay(500);
20
21    Servo_1_Angle(30);    //Turn servo 1 to 30 degrees
22    Serial.print("Distance: " + String(Get_Sonar()) + "\n");//Print ultrasonic distance
23    delay(500);
24
25    Servo_1_Angle(90);    //Turn servo 1 to 90 degrees
26    Serial.print("Distance: " + String(Get_Sonar()) + "\n");//Print ultrasonic distance
27    delay(500);
28  }
```

Code Explanation:

Initialize ultrasonic module and Servo.

```
6     Ultrasonic_Setup();  //Ultrasonic module initialization
7     Servo_Setup();       // initialization
```

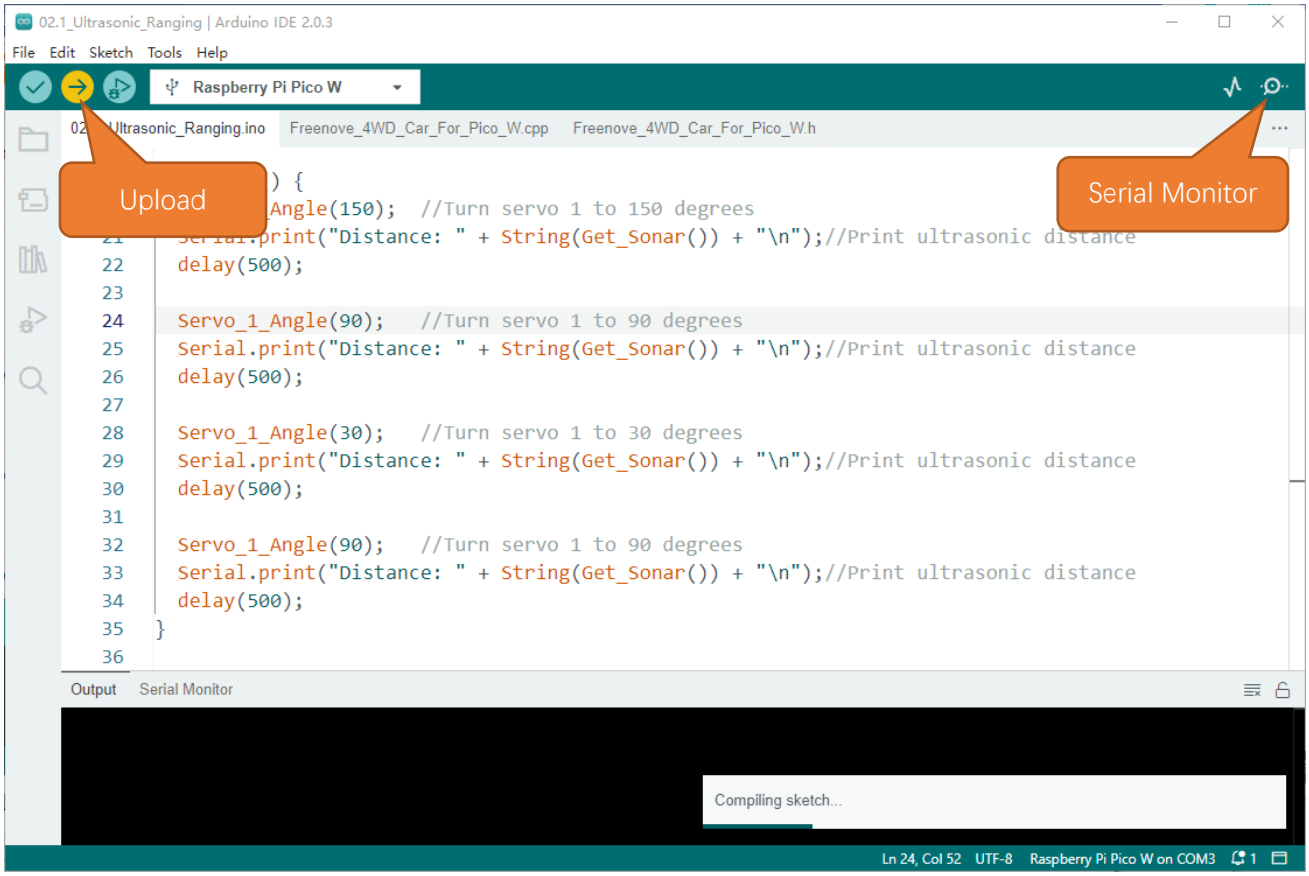
Obtain the distance between ultrasonic module and the obstacle and return a real number data in cm.

```
15    Get_Sonar()
```

Rotate the angle of the servo motor 1 and cooperate with the ultrasonic module to obtain distance data.

```
13    Servo_1_Angle(150);  //Turn servo 1 to 150 degrees
...   ...
17    Servo_1_Angle(90);   //Turn servo 1 to 90 degrees
...   ...
21    Servo_1_Angle(30);   //Turn servo 1 to 30 degrees
...   ...
25    Servo_1_Angle(90);   //Turn servo 1 to 90 degrees
```

Click “Upload” to upload the code to Raspberry Pi Pico (W). After uploading successfully, open Serial Monitor.



Set the baud rate as 115200.

3.2 Ultrasonic car

After starting the car, the ultrasound acquires data in various directions, makes judgments based on the data in each direction, and controls the car to avoid obstacles.

Sketch

Open the folder "02.2_Ultrasonic_Ranging_Car"in **"Freenove_4WD_Car_Kit_for_Raspberry_Pi_Pico\Ordinary_wheels\Sketches"** and double click "02.2_Ultrasonic_Ranging_Car.ino".

Code

```
#include <Arduino.h>
#include "Freenove_4WD_Car_For_Pico_W.h"

void setup() {
  Ultrasonic_Setup();           //Initialize the ultrasonic module
  Motor_Setup();                //motor initialization
  Servo_Setup();                //servo initialization
}
void loop()
{
  Ultrasonic_Car();
}
```

The code of automatic obstacle avoidance vehicle is as follows:

```
void Ultrasonic_Car() {
  int distance[3], tempDistance[3][5], sumDisntance;
  static u8 leftToRight = 0, servoAngle = 0, lastServoAngle = 0;  //
  const u8 scanAngle[2][3] = { { 150, 90, 30 }, { 30, 90, 150 } };
  int speedOffset = oa_VoltageCompensationToSpeed;
  for (int i = 0; i < 3; i++) {
    servoAngle = scanAngle[leftToRight][i];
    Servo_1_Angle(servoAngle);
    if (lastServoAngle != servoAngle) {
      delay(100);
    }
    lastServoAngle = servoAngle;
    for (int j = 0; j < COUNT_GET_SONAR; j++) {
      tempDistance[i][j] = Get_Sonar();
      delayMicroseconds(2 * SONIC_TIMEOUT);
      sumDisntance += tempDistance[i][j];
    }
    if (leftToRight == 0) {
      distance[i] = sumDisntance / COUNT_GET_SONAR;
```

```
20      } else {
21        distance[2 - i] = sumDisntance / COUNT_GET_SONAR;
22      }
23      sumDisntance = 0;
24    }
25  leftToRight = (leftToRight + 1) % 2;
26    Serial.println("Sonar : " + String(distance[0]) + " " + String(distance[1]) + " " +
    String(distance[2]));
27    if (distance[1] < OBSTACLE_DISTANCE){   //Too little distance ahead
28      if (distance[0] > distance[2] && distance[0] > OBSTACLE_DISTANCE){ //Left distance is
    greater than right distance
29        Motor_Move(-(SONAR_MODE_CRUISE_SPEED + speedOffset), -(SONAR_MODE_CRUISE_SPEED +
    speedOffset));  //Move back
30        delay(100);
31        Motor_Move(-(SONAR_MODE_CRUISE_SPEED + speedOffset), (SONAR_MODE_CRUISE_SPEED +
    speedOffset));
32      } else if (distance[0] < distance[2] && distance[2] > OBSTACLE_DISTANCE){ //Right distance
    is greater than left distance
33        Motor_Move(-(SONAR_MODE_CRUISE_SPEED + speedOffset), -(SONAR_MODE_CRUISE_SPEED +
    speedOffset));  //Move back
34        delay(100);
35        Motor_Move((SONAR_MODE_CRUISE_SPEED + speedOffset), -(SONAR_MODE_CRUISE_SPEED +
    speedOffset));
36      } else {  //Get into the dead corner, move back, then turn.
37        Motor_Move(-(SONAR_MODE_CRUISE_SPEED + speedOffset), -(SONAR_MODE_CRUISE_SPEED +
    speedOffset));
38        delay(100);
39        Motor_Move(-(SONAR_MODE_CRUISE_SPEED + speedOffset), (SONAR_MODE_CRUISE_SPEED +
    speedOffset));
40      }
41    } else {
42      if (distance[0] < OBSTACLE_DISTANCE_LOW){      //Obstacles on the left front.
43        Motor_Move(-(SONAR_MODE_CRUISE_SPEED + speedOffset), -(SONAR_MODE_CRUISE_SPEED +
    speedOffset));  //Move back
44        delay(100);
45        Motor_Move((70 + speedOffset), (20 + speedOffset));
46      } else if (distance[2] < OBSTACLE_DISTANCE_LOW){    //Obstacles on the right front.
47        Motor_Move(-(SONAR_MODE_CRUISE_SPEED + speedOffset), -(SONAR_MODE_CRUISE_SPEED +
    speedOffset));  //Move back
48        delay(100);
49        Motor_Move((20 + speedOffset), (70 + speedOffset));
50      } else {  //Cruising
51        Motor_Move((30 + speedOffset), (30 + speedOffset));
52      }
```

```
  }
}
```

Code Explanation:

The PWM value represents different speeds when the voltage of battery changes, so we set a speedOffset to compensate the difference.

```
void oa_CalculateVoltageCompensation() {
  Get_Battery_Voltage();
  float voltageOffset = BAT_VOL_STANDARD - batteryVoltage;
  oa_VoltageCompensationToSpeed = voltageOffset * OA_SPEED_OFFSET_PER_V;
}
```

OA SPEED OFFSET PER V: The default value is three, which is a tested value. You can try to test what is the most proper value.

We need control servo to 150°, 90°, 30°, 90°, 150°…

Get one ultrasonic module value (distance) at one angle.

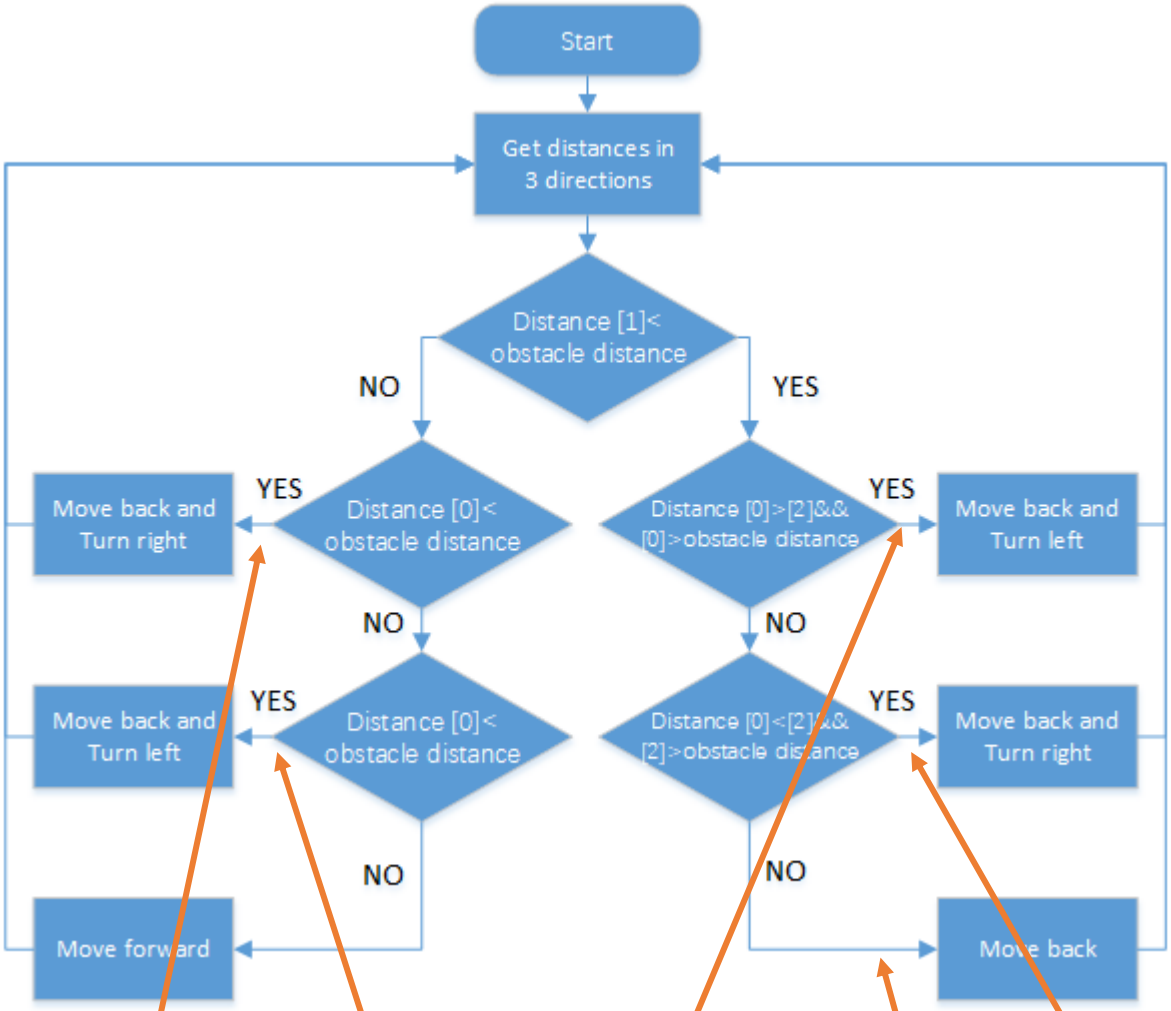
```
int distance[3], tempDistance[3][5], sumDisntance;
  static u8 leftToRight = 0, servoAngle = 0, lastServoAngle = 0;
  const u8 scanAngle[2][3] = { { 150, 90, 30 }, { 30, 90, 150 } };
  int speedOffset = oa_VoltageCompensationToSpeed;
  for (int i = 0; i < 3; i++) {
    servoAngle = scanAngle[leftToRight][i];
    Servo_1_Angle(servoAngle);
    if (lastServoAngle != servoAngle) {
      delay(100);
    }
    lastServoAngle = servoAngle;
    for (int j = 0; j < COUNT_GET_SONAR; j++) {
      tempDistance[i][j] = Get_Sonar();
      delayMicroseconds(2 * SONIC_TIMEOUT);
      sumDisntance += tempDistance[i][j];
    }
    if (leftToRight == 0) {
      distance[i] = sumDisntance / COUNT_GET_SONAR;
    } else {
      distance[2 - i] = sumDisntance / COUNT_GET_SONAR;
    }
    sumDisntance = 0;
  }
  leftToRight = (leftToRight + 1) % 2;
```

Then make the car react according to the distances above. Please refer to the flow chart for its logic.

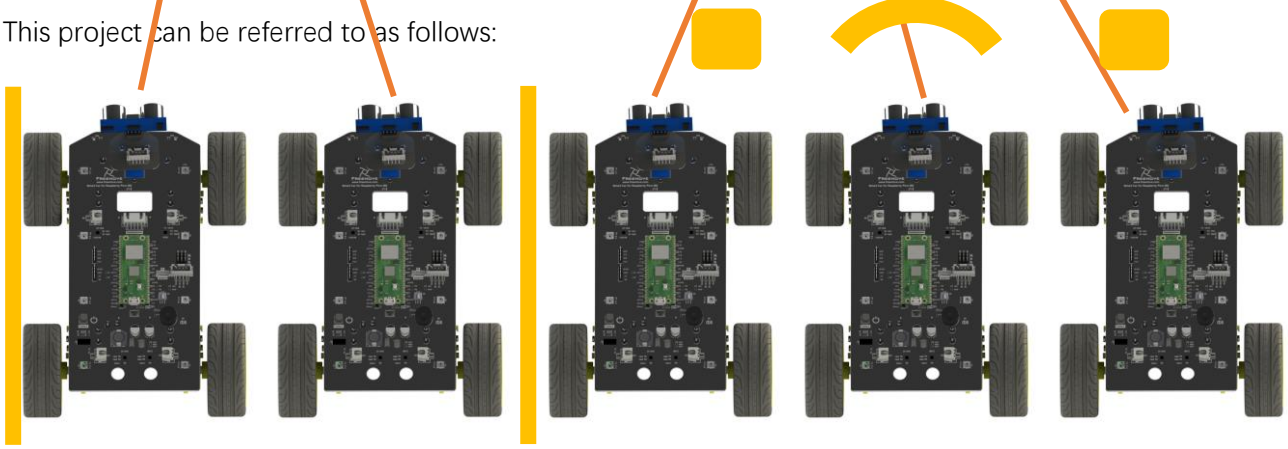
```
  if (distance[1] < OBSTACLE_DISTANCE){   //Too little distance ahead
    if (distance[0] > distance[2] && distance[0] > OBSTACLE_DISTANCE){ //Left distance is
greater than right distance
      Motor_Move(-(SONAR_MODE_CRUISE_SPEED + speedOffset), -(SONAR_MODE_CRUISE_SPEED +
```

```
speedOffset));  //Move back
      delay(100);
      Motor_Move(-(SONAR_MODE_CRUISE_SPEED + speedOffset), (SONAR_MODE_CRUISE_SPEED +
speedOffset));
    } else if (distance[0] < distance[2] && distance[2] > OBSTACLE_DISTANCE){ //Right distance
is greater than left distance
      Motor_Move(-(SONAR_MODE_CRUISE_SPEED + speedOffset), -(SONAR_MODE_CRUISE_SPEED +
speedOffset));  //Move back
      delay(100);
      Motor_Move((SONAR_MODE_CRUISE_SPEED + speedOffset), -(SONAR_MODE_CRUISE_SPEED +
speedOffset));
    } else {  //Get into the dead corner, move back, then turn.
      Motor_Move(-(SONAR_MODE_CRUISE_SPEED + speedOffset), -(SONAR_MODE_CRUISE_SPEED +
speedOffset));
      delay(100);
      Motor_Move(-(SONAR_MODE_CRUISE_SPEED + speedOffset), (SONAR_MODE_CRUISE_SPEED +
speedOffset));
    }
  } else {
    if (distance[0] < OBSTACLE_DISTANCE_LOW){      //Obstacles on the left front.
      Motor_Move(-(SONAR_MODE_CRUISE_SPEED + speedOffset), -(SONAR_MODE_CRUISE_SPEED +
speedOffset));  //Move back
      delay(100);
      Motor_Move((70 + speedOffset), (20 + speedOffset));
    } else if (distance[2] < OBSTACLE_DISTANCE_LOW){    //Obstacles on the right front.
      Motor_Move(-(SONAR_MODE_CRUISE_SPEED + speedOffset), -(SONAR_MODE_CRUISE_SPEED +
speedOffset));  //Move back
      delay(100);
      Motor_Move((20 + speedOffset), (70 + speedOffset));
    } else {  //Cruising
      Motor_Move((30 + speedOffset), (30 + speedOffset));
    }
  }
}
```

Code logic is as shown in the flow chart below:
Distance[0] is the distance to obstacle at the left.
Distance[1] is the distance to obstacle at the center.
Distance[2] is the distance to obstacle at the left.



This project can be referred to as follows:



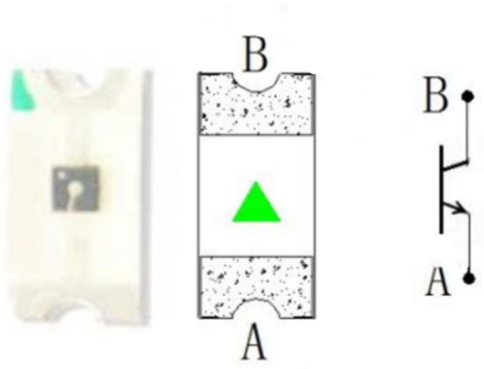
Chapter 4 Light Tracing Car

4.1 Photoresistor ADC

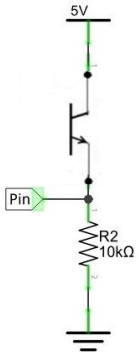
The photoresistor is very sensitive to the amount of light present. We can use this feature to make a light-tracing car. The car is controlled to trun toward the light source by reading the ADC values of the two photoresistors at the head of the car. Before we start, let us learn how to read the photoresistor value.

Photoresistor

A photoresistor is a NPN photosensitive triode. The outou
A Photoresistor is simply a light sensitive resistor. A Photoresistor's output current will change in proportion to the ambient light detected, thus changing the voltage value. With this characteristic, we can use a Photoresistor to detect light intensity. The Photoresistor and its electronic symbol are as follows.



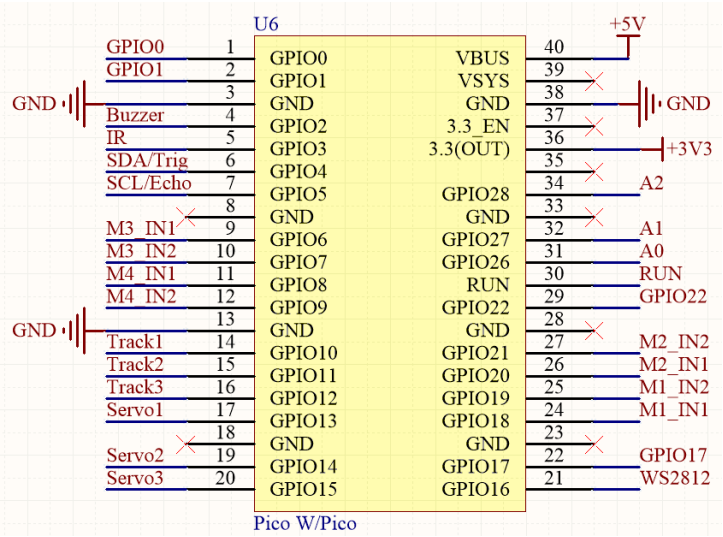
The circuit below is used to detect the change of a photoresistor's resistance value:

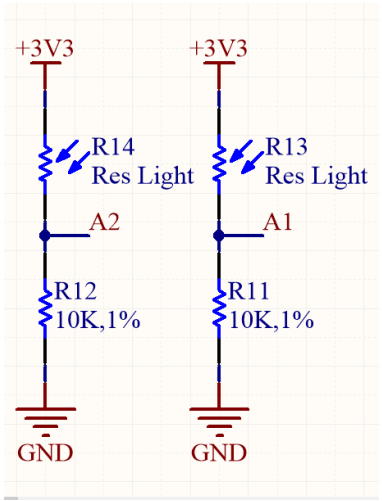


In the above circuit, when a photoresistor's resistance vale changes due to a change in light intensity, the voltage between the photoresistor and resistor R1 will also change. Therefore, the intensity of the light can be obtained by measuring this voltage.

Schematic

Photosensitive sensors are distributed on both sides of the front of the car, and the ADC value collection range of A1 and A2 is 0-1023. It can be seen from the circuit that photoresistors R13 and R14 are two independent light collection channels. When the photoresistor is in an absolutely dark environment, its corresponding ADC value is close to 0; and when it receives absolutely bright light, the value is close to 1023. When the photosensitive sensor receives light of a certain brightness, its corresponding ADC value is between 0-1023. Therefore, if the brightness of the light received by the two photoresistors is different, the values read by the ADC will have a large difference. Utilizing this difference, the car can realize the function of finding light.





Sketch

Next, we download the code to Raspberry Pi Pico W to test the photoresistors. Open the folder "03.1_Photosensitive" in **"Freenove_4WD_Car_Kit_for_Raspberry_Pi_Pico\Ordinary_wheels\Sketches"** and double click "03.1_Photosensitive.ino"

Code

```
#define Left_PHOTOSENSITIVE_PIN 28  //Define the pins that Raspberry Pi Pico reads photosensitive
#define Right_PHOTOSENSITIVE_PIN 27 //Define the pins that Raspberry Pi Pico reads photosensitive
int getLeftPhotosensitiveADCValue;  //Define a variable to store the left photosensitive adc value
int getRightPhotosensitiveADCValue; //Define a variable to store the right photosensitive adc value

void setup()
{
  pinMode(Left_PHOTOSENSITIVE_PIN, INPUT);//Configure the pins for input mode
  pinMode(Right_PHOTOSENSITIVE_PIN, INPUT);//Configure the pins for input mode
```

```
10    Serial.begin(115200);     //Initialize the serial port and set the baud rate to 115200
11  }
12
13  void loop()
14  {
15    getLeftPhotosensitiveADCValue = analogRead(Left_PHOTOSENSITIVE_PIN);
16    getRightPhotosensitiveADCValue = analogRead(Right_PHOTOSENSITIVE_PIN);
17    Serial.print("The photosensitive ADC on the left: ");
18    Serial.println(getLeftPhotosensitiveADCValue);       //Print the photosensitive ADC value on
    the left
19    Serial.print("The photosensitive ADC on the right: ");
20    Serial.println(getRightPhotosensitiveADCValue);      //Print the photosensitive ADC value on
    the right
21    delay(500);
22  }
```

Code Explanation:

Define the pin to read photoresistors.

```
1   #define Left_PHOTOSENSITIVE_PIN 28  //Define the pins that Raspberry Pi Pico reads
    photosensitive
2   #define Right_PHOTOSENSITIVE_PIN 27 //Define the pins that Raspberry Pi Pico reads
    photosensitive
```

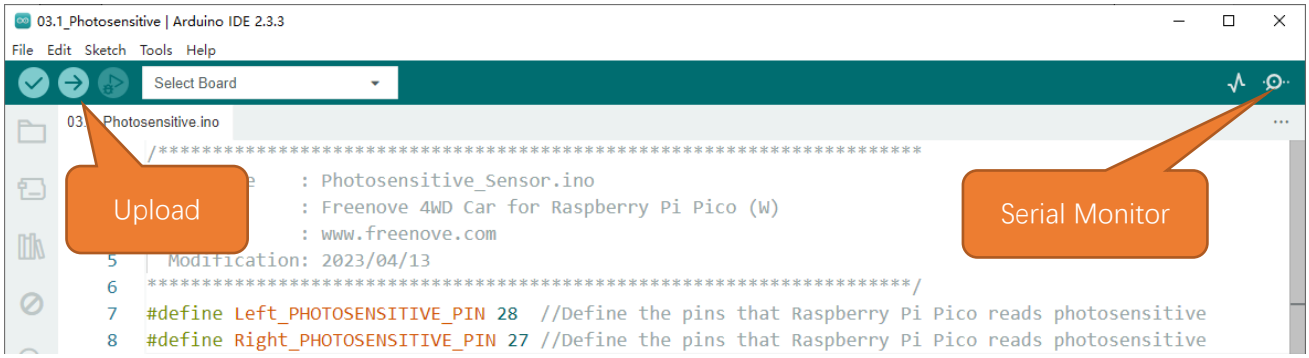
Call analogRead to read the differential value of the photoresistors and store it in the photosensitiveADC.

```
15    getLeftPhotosensitiveADCValue = analogRead(Left_PHOTOSENSITIVE_PIN);
16    getRightPhotosensitiveADCValue = analogRead(Right_PHOTOSENSITIVE_PIN);
```

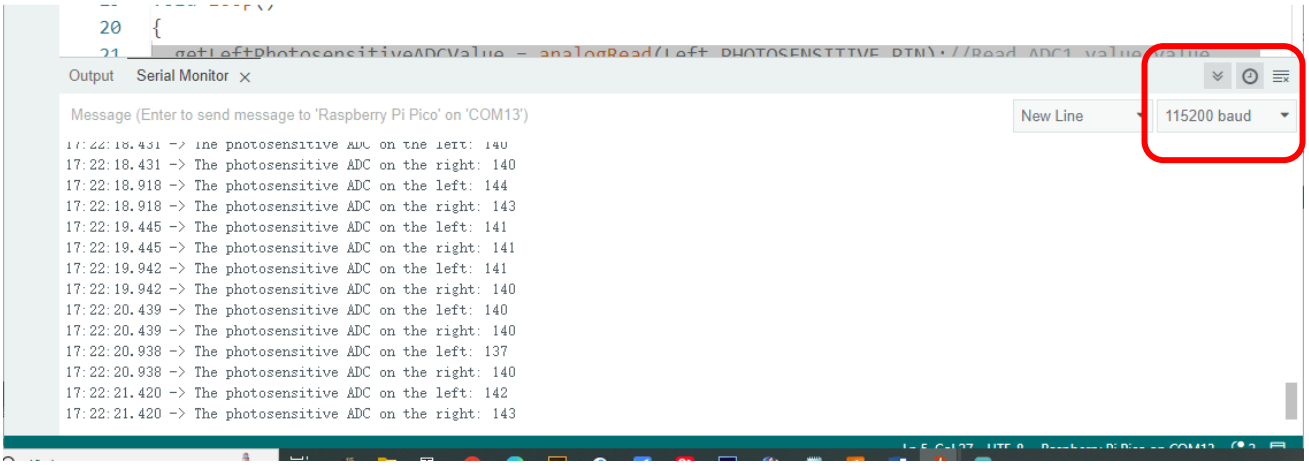
Print the photoresistor value through the serial port.

```
17    Serial.print("The photosensitive ADC on the left: ");
18    Serial.println(getLeftPhotosensitiveADCValue);       //Print the photosensitive ADC value on
    the left
19    Serial.print("The photosensitive ADC on the right: ");
20    Serial.println(getRightPhotosensitiveADCValue);      //Print the photosensitive ADC value on
    the right
```

Click “Upload” to upload the code to Raspberry Pi Pico W. When finishes uploading, click Serial Monitor.

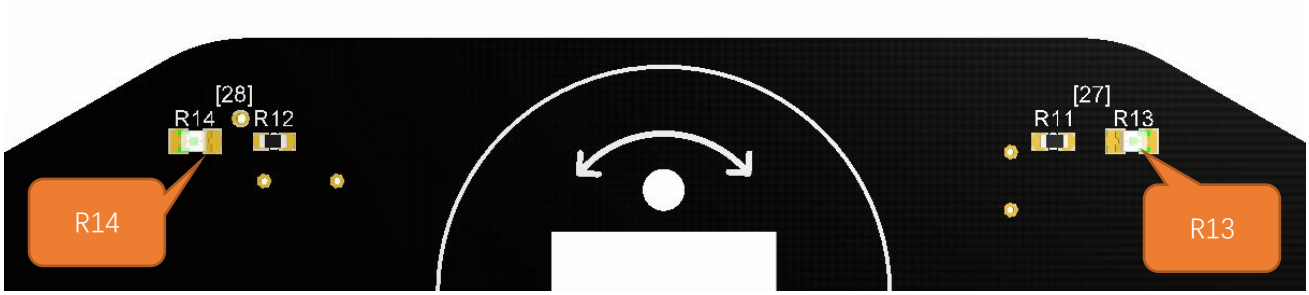


Set baud rate as 115200.


20 {
Output Serial Monitor
Message (Enter to send message to 'Raspberry Pi Pico' on 'COM13')
New Line
115200 baud
17:22:18.431 -> The photosensitive ADC on the right: 140
17:22:18.918 -> The photosensitive ADC on the left: 144
17:22:18.918 -> The photosensitive ADC on the right: 143
17:22:19.445 -> The photosensitive ADC on the left: 141
17:22:19.445 -> The photosensitive ADC on the right: 141
17:22:19.942 -> The photosensitive ADC on the left: 141
17:22:19.942 -> The photosensitive ADC on the right: 140
17:22:20.439 -> The photosensitive ADC on the left: 140
17:22:20.439 -> The photosensitive ADC on the right: 140
17:22:20.938 -> The photosensitive ADC on the left: 137
17:22:20.938 -> The photosensitive ADC on the right: 140
17:22:21.420 -> The photosensitive ADC on the left: 142
17:22:21.420 -> The photosensitive ADC on the right: 143

4.2 Light Tracing Car

It can be seen from the circuit board that R13 and R14 are photosensitive sensors on the right and left respectively. Set the ADC value read by photosensitive R13 to ADC1, and the ADC value read by photosensitive R14 to ADC2. When the light shines on R13, the resistance value of R13 gets smaller, and the ADC1 value increases, and the same is true for R14. Therefore, when ADC1 and ADC2 are greater than the set value at the same time, it means that there is a light source, and the car moves forward, otherwise, the car stops moving. When the car moves forward, detect the values of ADC1 on the right and ADC2 on the left. When ADC1 is greater than ADC2, it indicates that the light source is on the right side of the car, so the car turns right. When ADC1 is to be smaller than ADC2, the car turns left. When the difference between the light intensity on the left and the the right changes, the speed of the car changes accordingly.



Sketch

When car is powered ON, the ADC value of the current environment will be obtained. After initialization, the buzzer will sound once to remind users to test the light-tracing function. When users approach the car with light source, the car will turn following the light source.

Open the folder "03.2_Photosensitive_Car" in **"Freenove_4WD_Car_Kit_for_Raspberry_Pi_Pico\Ordinary_wheels\Sketches"** and double click "03.2_Photosensitive_Car.ino"

Code

```
#include <Arduino.h>
#include "Freenove_4WD_Car_For_Pico_W.h"

void setup() {
  Buzzer_Setup();
  Photosensitive_Setup();  //Photosensitive initialization
  Motor_Setup();           //Motor initialization
  Buzzer_Alert(1, 1);      //The buzzer beeps to prompt the user to start playing
  Serial.begin(115200);    //Initialize the serial port and set the baud rate to 115200
}

void loop() {
```

```
13    Light_Car();
14  }
```

The code of light tracing car is as follows:

```
1   void Light_Car() {
2       if (isLightModeFirstStarting) {
3         isLightModeFirstStarting = false;
4         Motor_Move(40, 40);
5         delay(200);
6       }
7       int leftLightValue = getLeftPhotosensitiveADCValue();
8       int rightLightValue = getRightPhotosensitiveADCValue();
9       int diffLightValue = (leftLightValue - rightLightValue) / 8;
10      if (leftLightValue > LIGHT_MIN_MOVED && rightLightValue > LIGHT_MIN_MOVED) {
11        int lsp = constrain(LIGHT_MODE_CRUISE_SPEED + diffLightValue, -100, 100);
12        int rsp = constrain(LIGHT_MODE_CRUISE_SPEED - diffLightValue, -100, 100);
13        Serial.println("sp: " + String(lsp) + " " + String(rsp) + " " + String(diffLightValue));
14        Motor_Move(lsp, rsp);
15      } else {
16        Motor_Move(0, 0);
17      }
18  }
```

Code Explanation:

Set the sensitivity of the photoresistor. You can adjust the value according to the change of the environment. The value is recommended to be between 50-150.

```
    #define LIGHT_MIN_MOVED 50
```

Initialize buzzer, photoresistor and motor.

```
5     Buzzer_Setup();
6     Photosensitive_Setup();  //Photosensitive initialization
7     Motor_Setup();           //Motor initialization
```

Get the ADC value of ambient light during initialization.

```
7       int leftLightValue = getLeftPhotosensitiveADCValue();
8       int rightLightValue = getRightPhotosensitiveADCValue();
```

Adjust the steering of the car according to the location of the light source.

```
10      if (leftLightValue > LIGHT_MIN_MOVED && rightLightValue > LIGHT_MIN_MOVED) {
11        int lsp = constrain(LIGHT_MODE_CRUISE_SPEED + diffLightValue, -100, 100);
12        int rsp = constrain(LIGHT_MODE_CRUISE_SPEED - diffLightValue, -100, 100);
13        Serial.println("sp: " + String(lsp) + " " + String(rsp) + " " + String(diffLightValue));
14        Motor_Move(lsp, rsp);
15      } else {
16        Motor_Move(0, 0);
17      }
```

Chapter 5 Line Tracking Car

5.1 Line tracking sensor

Track Sensor

There are three Reflective Optical Sensors on this car. When the infrared light emitted by infrared diode shines on the surface of different objects, the sensor will receive light with different intensities after reflection.
As we know, black objects absorb light better. So when black lines are drawn on the white plane, the sensor can detect the difference. The sensor can also be called Line Tracking Sensor.
When conducting experiments in this chapter, please replace the locomotive with LED Matrix as per the instructions.
Warning:
Reflective Optical Sensor (including Line Tracking Sensor) should be avoided using in environment with infrared interference, like sunlight. Sunlight contains a lot of invisible light such as infrared and ultraviolet. Under environment with intense sunlight, Reflective Optical Sensor cannot work normally.
The following table shows the values of all cases when three Tracking Sensors detect objects of different colors. Among them, black objects or no objects were detected to represent 1, and white objects were detected to represent 0.

Left	Middle	Right	Value(binary)	Value(decimal)
0	0	0	000	0
0	0	1	001	1
0	1	0	010	2
0	1	1	011	3
1	0	0	100	4
1	0	1	101	5
1	1	0	110	6
1	1	1	111	7

Sketch

The line tracking module is connected to the Raspberry Pi Pico (W). The Raspberry PI Pico (W) tracks whether the three channels of the module are triggered by acquiring the existence of the line, and prints it out through the serial port.

Open the folder " 04.1_Tracking_Sensor " in **"Freenove_4WD_Car_Kit_for_Raspberry_Pi_Pico\Ordinary_wheels\Sketches"** and double click "04.1_Tracking_Sensor.ino".

Code

```
1  #include <Arduino.h>
```

```
#include "Freenove_4WD_Car_For_Pico_W.h"

void setup() {
  Serial.begin(115200); //set baud rate
  Track_Setup();
}

void loop() {
  Track_Read();
  Serial.print("Sensor Value (L / M / R / ALL) : ");
  for (int i = 0; i < 4; i++) {
    Serial.print(sensorValue[i]);
    Serial.print('\t');
  }
  Serial.print('\n');
  delay(500);
}
```

Code Explanation:

Trace module initialization

```
6   Track_Setup();
```

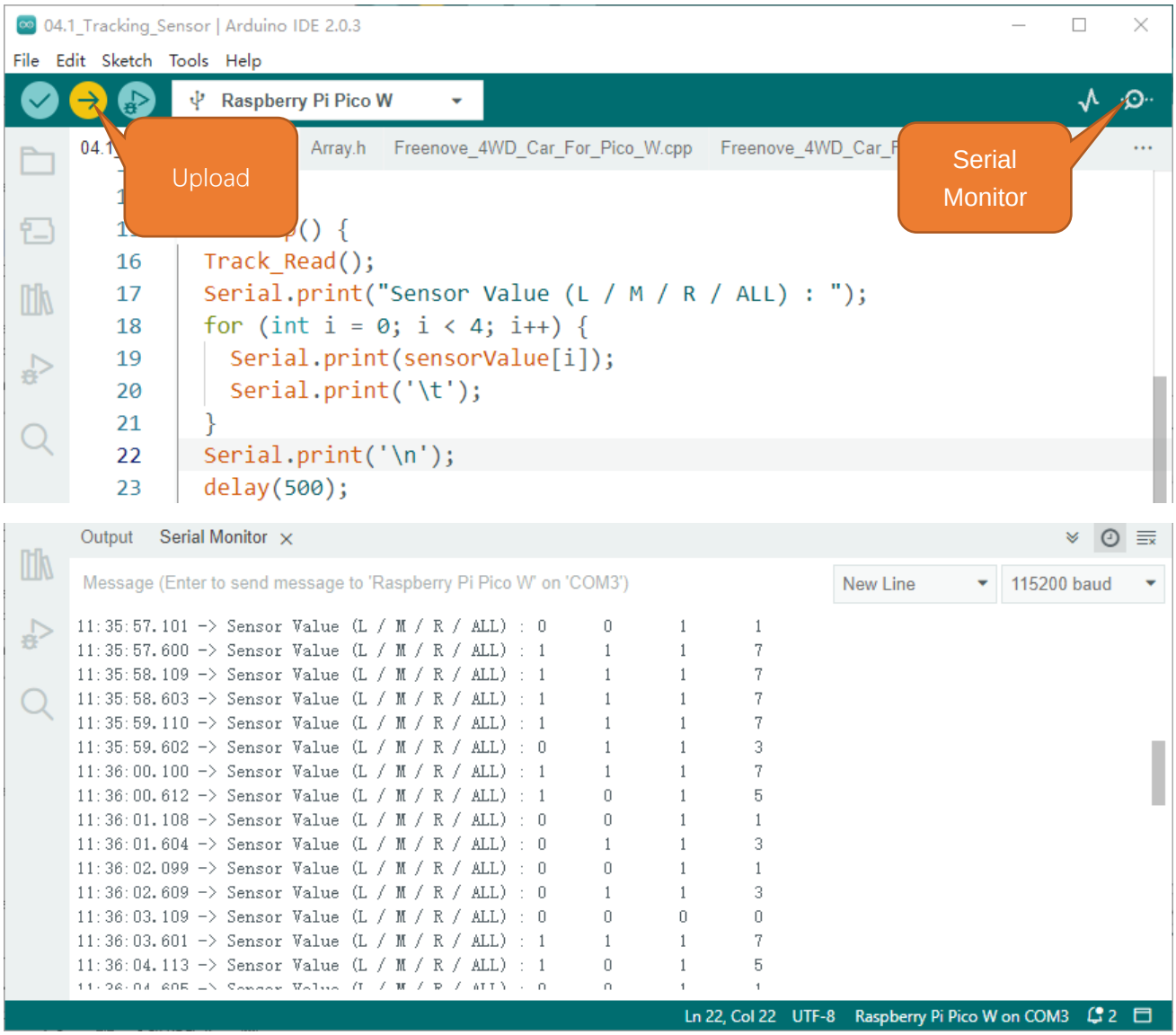
The function that obtains the tracking module feedback value. After calling this function, Raspberry Pi Pico W will store the data in the array sensorValue[].

```
10  Track_Read();
```

Print the obtained feedback value through serial port.

```
11  Serial.print("Sensor Value (L / M / R / ALL) : ");
12  for (int i = 0; i < 4; i++) {
13    Serial.print(sensorValue[i]);
14    Serial.print('\t');
15  }
16  Serial.print('\n');
```

Click "Upload" to upload to code to Raspberry Pi Pico (W) development board. After uploading successfully, click "Serial Monitor".

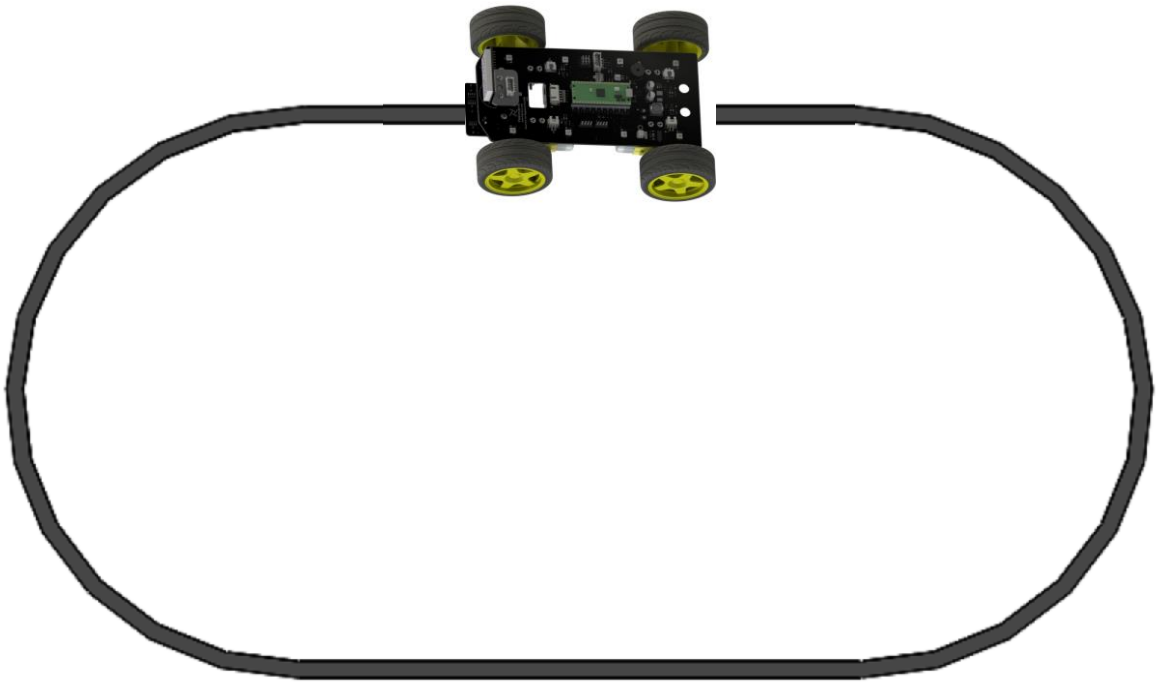


5.2 Line Tracking Car

The car will make different actions according to the value transmitted by the line-tracking sensor.

Left	Middle	Right	Value(binary)	Value(decimal)	Action
0	0	0	000	0	Move Forward
0	0	1	001	1	Turn Right
0	1	0	010	2	Move Forward
0	1	1	011	3	Turn Right
1	0	0	100	4	Turn Left
1	0	1	101	5	Move Forward
1	1	0	110	6	Turn Left
1	1	1	111	7	Stop

Turn on the power. Use a black tape to build a line and then put your car on it as below.



Sketch

Open the folder "04.2_Track_Car" in **"Freenove_4WD_Car_Kit_for_Raspberry_Pi_Pico_W\Sketches"** and double click "04.2_Track_Car.ino".

Code

```
#include <Arduino.h>
#include "Freenove_4WD_Car_For_Pico_W.h"

#define SPEED_LV4    ( 90 )
#define SPEED_LV3    ( 75 )
```

```
6  #define SPEED_LV2    ( 60 )
7  #define SPEED_LV1    ( 45 )
8
9  void setup(){
10   Track_Setup();   //Trace module initialization
11   Motor_Setup();   //Motor drive initialization
12   Emotion_Setup(); //Emotion initialization
13 }
14
15 void loop(){
16   Track_Read();
17   switch (sensorValue[3])
18   {
19     case 2:   //010
20     case 5:   //101
21       showArrow(1, 100);
22       Motor_Move(SPEED_LV1 , SPEED_LV1);    //Move Forward
23       break;
24     case 0:   //000
25       Motor_Move(SPEED_LV1 , SPEED_LV1);  //Move Forward
26       break;
27     case 7:   //111
28       eyesBlink1(100);
29       Motor_Move(0 , 0);    //Stop
30       break;
31     case 4:   //100
32     case 6:   //110
33       wheel(2, 100);
34       Motor_Move(-SPEED_LV4 , SPEED_LV3);//Turn Left
35       break;
36     case 1:   //001
37     case 3:   //011
38       wheel(1, 100);
39       Motor_Move(SPEED_LV3 , -SPEE-_LV4);  //Turn Right
40       break;
41     default:
42       break;
43   }
44 }
```

Code Explanation:

Initialize PCF8574 chip, Motor, Emotion module.

```
10   Track_Setup();   //Trace module initialization
11   Motor_Setup();   //Motor drive initialization
12   Emotion_Setup(); //Emotion initialization
```

The function that obtains the feedback value from the tracking module. After calling this function, Raspberry Pi Pico W will store the data in the array sensorValue[].

```
16    Track_Read();
```

Control the car to move forward, turn left, turn right, stop, etc. based on the value of line tracking module.

```
17    switch (sensorValue[3])
18    {
19      case 2:   //010
20      case 5:   //101
21        showArrow(1, 100);
22        Motor_Move(SPEED_LV1 , SPEED_LV1);     //Move Forward
23        break;
24      case 0:   //000
25        Motor_Move(SPEED_LV1 , SPEED_LV1);  //Move Forward
26        break;
27      case 7:   //111
28        eyesBlink1(100);
29        Motor_Move(0 , 0);     //Stop
30        break;
31      case 4:   //100
32      case 6:   //110
33        wheel(2, 100);
34        Motor_Move(-SPEED_LV4 , SPEED_LV3);//Turn Left
35        break;
36      case 1:   //001
37      case 3:   //011
38        wheel(1, 100);
39        Motor_Move(SPEED_LV3 , -SPEE-_LV4);  //Turn Right
40        break;
41      default:
42        break;
43    }
```

Note: The tracks made of black tape vary among individuals. If your car cannot move along the black tape, you can modify the parameters in Motor_Move().

Chapter 6 Infrared Car

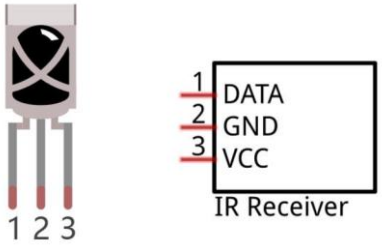
6.1 Introduction of infrared reception function

Infrared Remote

An infrared(IR) remote control is a device with a certain number of buttons. Pressing down different buttons will make the infrared emission tube, which is located in the front of the remote control, send infrared ray with different command. Infrared remote control technology is widely used in electronic products such as TV, air conditioning, etc. Thus making it possible for you to switch TV programs and adjust the temperature of the air conditioning when away from them. The remote control we use is shown below:

Infrared receiver

An infrared(IR) receiver is a component which can receive the infrared light, so we can use it to detect the signal emitted by the infrared remote control. DATA pin here outputs the received infrared signal.



When you use the infrared remote control, the infrared remote control sends a key value to the receiving circuit according to the pressed keys. We can program the Raspberry Pi Pico W to do things like lighting, when a key value is received.

The following is the key value that the receiving circuit will receive when each key of the infrared remote control is pressed.

ICON	KEY Value	ICON	KEY Value
	BA45FF00		F20DFF00
	B847FF00		F30CFF00
	BB44FF00		E718FF00
	BF40FF00		A15EFF00
	BC43FF00		F708FF00
	F807FF00		E31CFF00
	EA15FF00		A55AFF00
	F609FF00		BD42FF00
	E916FF00		AD52FF00
	E619FF00		B54AFF00

This sketch uses the infrared receiving tube to receive the value sent form the infrared remote control, and print it out via the serial port.

Sketch

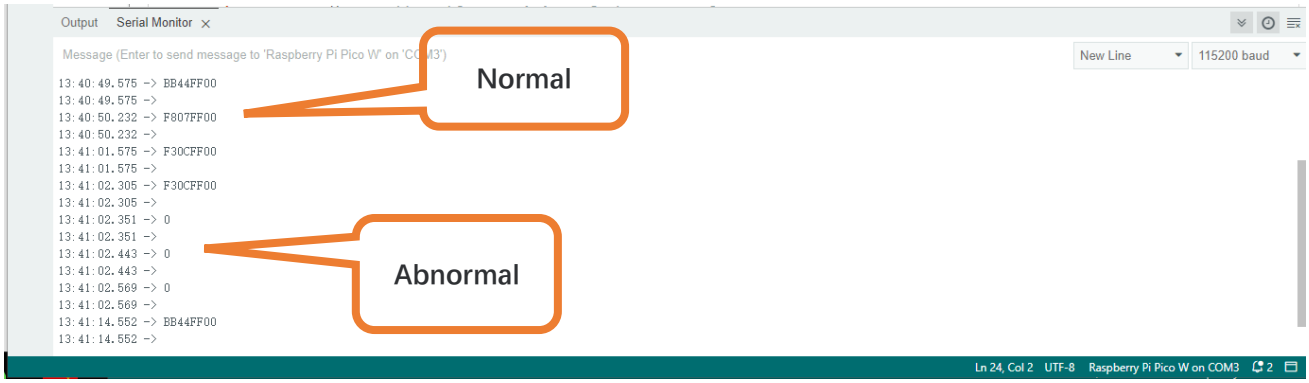
Each time when you press the infrared remote control, the car will print the received infrared coding value through serial port.

Open the folder “05.1_IR_Receiver”in **“Freenove_4WD_Car_Kit_for_Raspberry_Pi_Pico\Ordinary_wheels\Sketches”** and double click “05.1_IR_Receiver.ino”.

Code

```
#include <IRremote.hpp>
#define IR_Pin 3  // Infrared receiving pin
#define ENABLE_LED_FEEDBACK true
#define DISABLE_LED_FEEDBACK false

void setup() {
  Serial.begin(115200);     // Initialize the serial port and set the baud rate to 115200
  IrReceiver.begin(IR_Pin, DISABLE_LED_FEEDBACK);  // Start the receiver
  Serial.print("IRrecvDemo is now running and waiting for IR message on Pin ");
  Serial.println(IR_Pin);  //print the infrared receiving pin
}

void loop() {
  if (IrReceiver.decode()) {
    unsigned long value = IrReceiver.decodedIRData.decodedRawData;
    Serial.println(value, HEX);  // Print "old" raw data
    IrReceiver.resume();     // Enable receiving of the next value
  }
}
```

Download the code to Raspberry Pi Pico (W), open the serial port monitor, set the baud rate to 115200, press the IR remote control, the pressed keys value will be printed out through the serial port. As shown in the following figure: (Note that if the remote control button is long pressed, the infrared receiving circuit receives a "0".)



First, include header file. Each time you use the infrared sensor, you need to include the header file at the beginning of the program.

```
#include <IRremote.hpp>
```

Second, define an infrared receive pin and the infrared sensor is initialized.

```
#define IR_Pin 3                    // Infrared receiving pin
...
  IrReceiver.begin(IR_Pin, DISABLE_LED_FEEDBACK);  // Start the receiver
```

Finally, **IrReceiver.decode()** is used to determine whether an infrared signal has been received, returning true/1 if an infrared signal has been received, or false/0 if no infrared signal has been received; If an infrared signal is received, the received infrared coded value is printed through the serial port.

Please note that **IrReceiver.resume()** must be applied to release the infrared receiver function each time data are received. Otherwise, the infrared receiver function can only be used once and data cannot be received

next time.

```
14    if (IrReceiver.decode()) {
15      unsigned long value = IrReceiver.decodedIRData.decodedRawData;
16      Serial.println(value, HEX);  // Print "old" raw data
17      IrReceiver.resume();     // Enable receiving of the next value
18    }
```

6.2 Infrared Car

Based on the previous section, we use the infrared remote control to control the car. Press the black button on the infrared remote control to control the car to move forward, backward, turn left, and turn right. Press other buttons and the cart stops moving.

Sketch

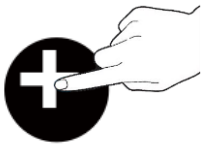
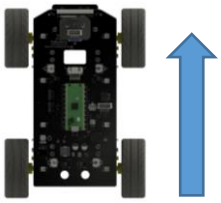
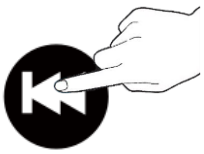
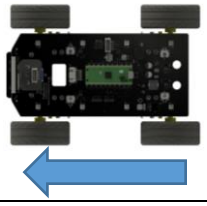
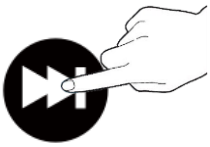
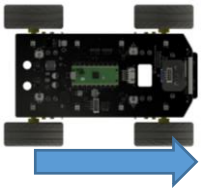
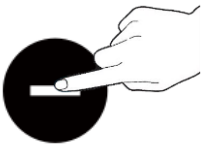
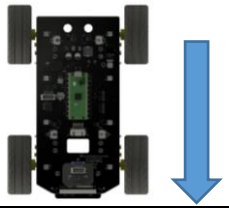
Open the folder "05.2_IR_Receiver_Car" in the **"Freenove_4WD_Car_Kit_for_Raspberry_Pi_Pico\Ordinary_wheels\Sketches"** and double click "05.2_IR_Receiver_Car.ino".

Code

```
#include <Arduino.h>
#include <IRremote.hpp>
#include "Freenove_4WD_Car_For_Pico_W.h"
#define IR_Pin 3
#define ENABLE_LED_FEEDBACK true
#define DISABLE_LED_FEEDBACK false
int motor_speed = 50;
int emotionMode = 4;
int motor_flag = 0;

void setup() {
  Serial.begin(115200);
  Motor_Setup();                                  //Motor initialization
  Emotion_Setup();                                //Initializes Emotion module
  IrReceiver.begin(IR_Pin, DISABLE_LED_FEEDBACK);  // Start the receiver
}

void loop() {
  if (IrReceiver.decode()) {
    unsigned long value = IrReceiver.decodedIRData.decodedRawData;
    handleControl(value);         // Handle the commands from remote control
    Serial.println(value, HEX);  // Print "old" raw data
    Serial.println();
    IrReceiver.resume();  // Enable receiving of the next value
  }
  showEmotion(emotionMode);
}

void handleControl(unsigned long value) {
  // Handle the commands
```

```
  switch (value) {
    case 0xBF40FF00:  // Receive the number '+'
      motor_flag = 1;
      Motor_Move(motor_speed , motor_speed);  // Go forward
      delay(200);
      Motor_Move(0 , 0);
      break;

    case 0xE619FF00:  // Receive the number '-'
      motor_flag = 2;
      Motor_Move(-motor_speed , -motor_speed);  // Back up
      delay(200);
      Motor_Move(0 , 0);
      break;

    case 0xF807FF00:  // Receive the number '|<<'
      motor_flag = 3;
      Motor_Move(-motor_speed , motor_speed);  // Turn left
      delay(200);
      Motor_Move(0 , 0);
      break;

    case 0xF609FF00:  // Receive the number '>>|'
      motor_flag = 4;
      Motor_Move(motor_speed , -motor_speed);  // Turn right
      delay(200);
      Motor_Move(0 , 0);
      break;

    case 0xE916FF00:  // Receive the number '0'
      emotionMode = millis() % 10;
      break;

    default:
      motor_flag = 0;
      Motor_Move(0 , 0);  //stop
      break;
  }
}
```

Compile and upload the code to Raspberry Pi Pico (W). When pressing "0" of the infrared remote control, the expression module will randomly display the content dynamically.

		Move forward
		Turn left
		Turn right
		Move back

Code Explanation:

Variable **IrReceiver.DecodedIRData.DecodedRawData** holds the infrared remote control encoding information, call handleccontrol function performs different code value corresponding action. After each execution of the program, call the **IrReceiver.resume()** function to release the infrared pin. If you do not call this function, you cannot use the infrared receiving and decoding functions again.

```
18  void loop() {
19    if (IrReceiver.decode()) {
20      unsigned long value = IrReceiver.decodedIRData.decodedRawData;
21      handleControl(value);         // Handle the commands from remote control
22      Serial.println(value, HEX);  // Print "old" raw data
23      Serial.println();
24      IrReceiver.resume();  // Enable receiving of the next value
25    }
26    showEmotion(emotionMode);
27  }
```

Infrared key code value processing function, receives instructions sent by the infrared remote control, and execute the corresponding program.

```
29  void handleControl(unsigned long value) {
30    // Handle the commands
31    switch (value) {
32      case 0xBF40FF00:  // Receive the number '+'
...       ...
39      case 0xE619FF00:  // Receive the number '-'
...       ...
```

```
46      case 0xF807FF00:  // Receive the number '|<<'
...      ...
53      case 0xF609FF00:  // Receive the number '>>|'
...      ...
60      case 0xE916FF00:  // Receive the number '0'
...      ...
64      default:        // Control the car to stop moving
...      ...
68    }
69  }
```

6.3 Multi-Functional Infrared Car

Following the previous section, we now integrate other functions into the infrared car, and most of the car's functions can be controlled by the infrared remote.

Sketch

Open the folder "05.3_Multi_Functional_Car" in the **"Freenove_4WD_Car_Kit_for_Raspberry_Pi_Pico\Ordinary_wheels\Sketches"** and double click "05.3_Multi_Functional_Car.ino".

Code

```
#include <Arduino.h>
#include <Adafruit_NeoPixel.h>
#include <IRremote.hpp>
#include "Freenove_4WD_Car_For_Pico_W.h"
#include "Freenove_4WD_Car_Emotion.h"
#include "Freenove_VK16K33_Lib.h"
#include "Freenove_4WD_Car_WS2812.h"

#define IR_Pin 3  // Infrared receiving pin
#define ENABLE_LED_FEEDBACK          true
#define DISABLE_LED_FEEDBACK         false

static int servo_1_angle = 90;
int emotion_flag = 0;
int ws2812_flag = 0;

int CAR_MODE_VOL = 0;
int LASt_CAR_MODE_VOL = 0;

void setup() {
  Serial.begin(115200);  //Turn on the serial port monitor and set the baud rate to 115200
  Motor_Setup();         //Initialize motor
  Servo_Setup();         //Initialize servo
  Buzzer_Setup();        //Initialize the buzzer
  WS2812_Setup();        //WS2812 initialization
  Emotion_and_Ultrasonic_Setup();
  IrReceiver.begin(IR_Pin, DISABLE_LED_FEEDBACK); // Start the receiver
  Servo_1_Angle(servo_1_angle);
  delay(100);
}

```

```
void loop() {
  if (IrReceiver.decode()) {
      unsigned long value=IrReceiver.decodedIRData.decodedRawData;
      handleControl(value);       // Handle the commands from remote control
      Serial.println(value, HEX); // Print "old" raw data
      Serial.println();
      IrReceiver.resume(); // Enable receiving of the next value
  }
  Emotion_and_Ultrasonic_Setup();
  oa_CalculateVoltageCompensation();
  if (Check_Module_value == MATRIX_IS_EXIST) {
    Emotion_Show(emotion_task_mode);  //Led matrix display function
  } else if (Check_Module_value == SONAR_IS_ESIST) {
  }
  Car_Select(carFlag);
  WS2812_Show(ws2812_task_mode);  //Car color lights display function
}

void handleControl(unsigned long value) {
  // Handle the commands
  switch (value) {
    case 0xBF40FF00:  // Receive the number '+'
      Motor_Move(50, 50);
      delay(300);
      Motor_Move(0, 0);
      break;
    case 0xE619FF00:  // Receive the number '-'
      Motor_Move(-50, -50);
      delay(300);
      Motor_Move(0, 0);
      break;
    case 0xF807FF00:  // Receive the number '|<<'
      Motor_Move(-50, 50);
      delay(300);
      Motor_Move(0, 0);
      break;
    case 0xF609FF00:  // Receive the number '>>|'
      Motor_Move(50, -50);
      delay(300);
      Motor_Move(0, 0);
      break;
    case 0xEA15FF00:  // Receive the number '▶'
      Motor_Move(0, 0);
      break;
```

```
76      case 0xE916FF00:  // Receive the number '0'
77        servo_1_angle = servo_1_angle + 10;
78        Servo_1_Angle(servo_1_angle);
79        break;
80      case 0xF30CFF00:  // Receive the number '1'
81        servo_1_angle = servo_1_angle - 10;
82        Servo_1_Angle(servo_1_angle);
83        break;
84      case 0xF708FF00:  // Receive the number '4'
85        servo_1_angle = 90;
86        Servo_1_Angle(servo_1_angle);
87        break;
88      case 0xF20DFF00:  // Receive the number 'C'
89        isLightModeFirstStarting = true;
90        Servo_1_Angle(90);
91        Car_SetMode(1);
92        break;
93      case 0xA15EFF00:  // Receive the number '3'
94        Servo_1_Angle(90);
95        Car_SetMode(2);
96        break;
97      case 0xA55AFF00:  // Receive the number '6'
98        Servo_1_Angle(90);
99        if (Check_Module_value == SONAR_IS_ESIST) {
100         LASt_CAR_MODE_VOL = 1;
101         Car_SetMode(3);
102       }
103       if (Check_Module_value != SONAR_IS_ESIST) {
104         Buzzer_Variable(2000, 50, 2);
105         Car_SetMode(0);
106         Motor_Move(0, 0);
107       }
108       break;
109     case 0xB54AFF00:  // Receive the number '9'
110       emotion_flag = 0;
111       Emotion_SetMode(emotion_flag);
112       Servo_1_Angle(90);
113       Car_SetMode(0);
114       Motor_Move(0, 0);
115       break;
116     case 0xBB44FF00:  // Receive the number 'TEST'
117       Buzzer_Variable(2000, 100, 1);
118       break;
119     case 0xE718FF00:  // Receive the number '2'
```

```
120         if (Check_Module_value == MATRIX_IS_EXIST) {
121           emotion_flag = emotion_flag + 1;
122           if (emotion_flag > 7) {
123             emotion_flag = 0;
124           }
125           Emotion_SetMode(emotion_flag);  //Display
126           Serial.print(" \n matrix is exist !!! \n");
127         }
128         if (Check_Module_value != MATRIX_IS_EXIST) {
129           Buzzer_Variable(2000, 50, 2);
130           Serial.print(" \n sonar is exist !!! \n");
131         }
132         break;
133       case 0xE31CFF00:  // Receive the number '5'
134         emotion_flag = 0;
135         Emotion_SetMode(emotion_flag);
136         break;
137       case 0xBD42FF00:  // Receive the number '7'
138         ws2812_flag = ws2812_flag + 1;
139         if (ws2812_flag >= 6) {
140           ws2812_flag = 0;
141         }
142         WS2812_SetMode(ws2812_flag);
143         break;
144       case 0xAD52FF00:  // Receive the number '8'
145         ws2812_flag = 0;
146         WS2812_SetMode(ws2812_flag);
147         break;
148       case 0xFFFFFFFF:  // Remain unchanged
149         break;
150       default:
151         break;
152     }
153 }
```

After the code is successfully uploaded, turn on the power of the car and use the infrared remote to control the car and other functions. The corresponding keys and their functions are shown in the following table:

ICON	KEY Value	Function	ICON	KEY Value	Function
+	BF40FF00	Move forward	2	E718FF00	Change the expression display mode
⏮	F807FF00	Turn left	5	E31CFF00	Turn off emoticons
⏭	F609FF00	Turn light	7	BD42FF00	Change the display mode of the WS2812
−	E619FF00	Move back	8	AD52FF00	Turn off WS2812 display
▶	EA15FF00	Stop the car	C	F20DFF00	Light tracing car mode
0	E916FF00	Control servo turn left	3	A15EFF00	Line tracking car mode
1	F30CFF00	Control servo turn right	6	A55AFF00	Ultrasonic obstacle avoidance car mode
4	F708FF00	Control servo turn to 90°	9	B54AFF00	Manual control mode
TEST	BB44FF00	Control the buzzer			

Since the ultrasonic module shares the interface with the LED Matrix, only one of them can be used at the same time. When the function of the sent button conflicts with the current head module, the buzzer will sound twice continuously. The situations can be as follows:

When "6" is pressed, if the LED Matrix is connected to the head, the buzzer will sound twice.

When "2" is pressed, if the head is connected to an ultrasonic module, the buzzer will sound twice.

When you hear the buzzer beep twice, please replace the module on the head to continue the experiment.

Code Explanation:

Add the header file for the car.

```
#include <Arduino.h>
#include <Adafruit_NeoPixel.h>
#include <IRremote.hpp>
#include "Freenove_4WD_Car_For_Pico_W.h"
#include "Freenove_4WD_Car_Emotion.h"
#include "Freenove_VK16K33_Lib.h"
#include "Freenove_4WD_Car_WS2812.h"
```

Initialize each function of the car.

```
  Serial.begin(115200);  //Turn on the serial port monitor and set the baud rate to 115200
  Motor_Setup();         //Initialize motor
  Servo_Setup();         //Initialize servo
  Buzzer_Setup();        //Initialize the buzzer
  WS2812_Setup();        //WS2812 initialization
  Emotion_and_Ultrasonic_Setup();
  IrReceiver.begin(IR_Pin, DISABLE_LED_FEEDBACK); // Start the receiver
  Servo_1_Angle(servo_1_angle);
  delay(100);
```

Infrared key code value processing function receives instructions sent by the infrared remote control and execute the corresponding program.

```
void handleControl(unsigned long value) {
  // Handle the commands
  switch (value) {
    case 0xBF40FF00:  // Receive the number '+'
      ...
    case 0xE619FF00:  // Receive the number '-'
      ...
    case 0xF807FF00:  // Receive the number '|<<'
      ...
    case 0xF609FF00:  // Receive the number '>>|'
      ...
    case 0xEA15FF00:  // Receive the number '▶'
      ...
    case 0xE916FF00:  // Receive the number '0'
      ...
    case 0xF30CFF00:  // Receive the number '1'
      ...
    case 0xF708FF00:  // Receive the number '4'
      ...
    case 0xF20DFF00:  // Receive the number 'C'
      ...
    case 0xA15EFF00:  // Receive the number '3'
      ...
```

```
97      case 0xA55AFF00:  // Receive the number '6'
...       ...
109     case 0xB54AFF00:  // Receive the number '9'
...       ...
116     case 0xBB44FF00:  // Receive the number 'TEST'
...       ...
119     case 0xE718FF00:  // Receive the number '2'
...       ...
133     case 0xE31CFF00:  // Receive the number '5'
...       ...
137     case 0xBD42FF00:  // Receive the number '7'
...       ...
144     case 0xAD52FF00:  // Receive the number '8'
...       ...
148     case 0xFFFFFFFF:  // Remain unchanged
149       break;
150     default:
151       break;
152   }
153 }
```

Chapter 7 WiFi Car(Only for Pico W)

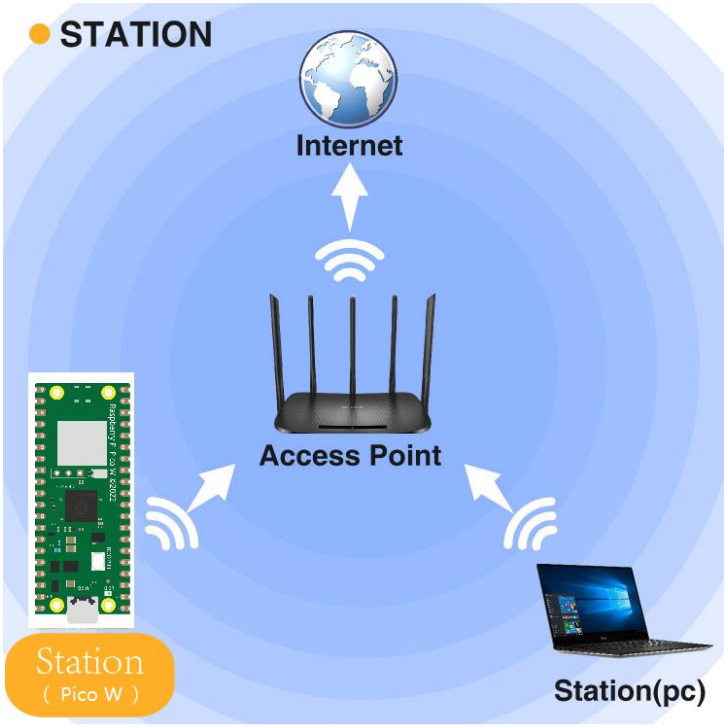
7.1 WiFi Sending and Receiving Data

This chapter requires the use of a Raspberry Pico W. If you use a Pico, you will not be able to finish projects in this chapter.

Before programming, we need to have a basic understanding of WiFi.

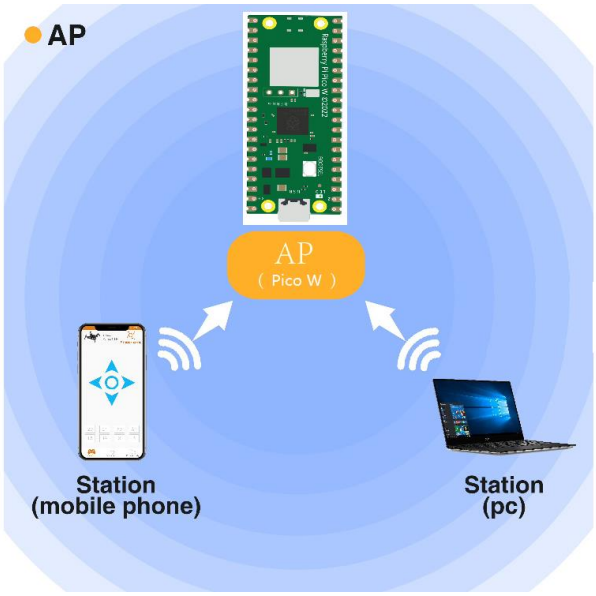
Station mode

When Raspberry Pi Pico W is in Station mode, it acts as a WiFi client. It can connect to the router network and communicate with other devices on the same LAN. As shown below, the PC is connected to a wireless router, and if Raspberry Pi Pico W wants to communicate with the PC, it needs to be connected to the same router.



AP mode

When Raspberry Pi Pico W selects AP mode, it creates a hotspot network that is separate from the Internet and waits for other WiFi devices to connect. As shown in the figure below, Raspberry Pi Pico W is used as a hotspot. If a mobile phone or PC wants to communicate with Raspberry Pi Pico W, it must be connected to the hotspot of Raspberry Pi Pico W. Only after a connection is established with Raspberry Pi Pico W can they communicate.


AP
AP
(Pico W)
Station
(mobile phone)
Station
(pc)

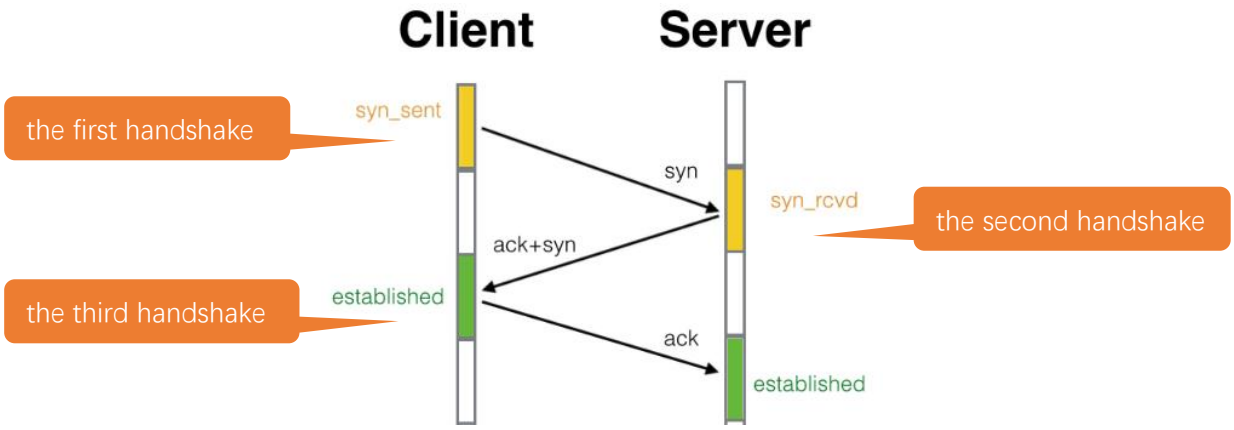
AP+Station mode

In addition to AP mode and station mode, Raspberry Pi Pico W can also use AP mode and station mode at the same time. This mode includes the functions of the previous two modes. Turn on Raspberry Pi Pico W's station mode, connect it to the router network, and it can communicate with the Internet via the router. At the same time, turn on its AP mode to create a hotspot network and other WiFi devices can choose to connect to the router network or the hotspot network to communicate with Raspberry Pi Pico W.

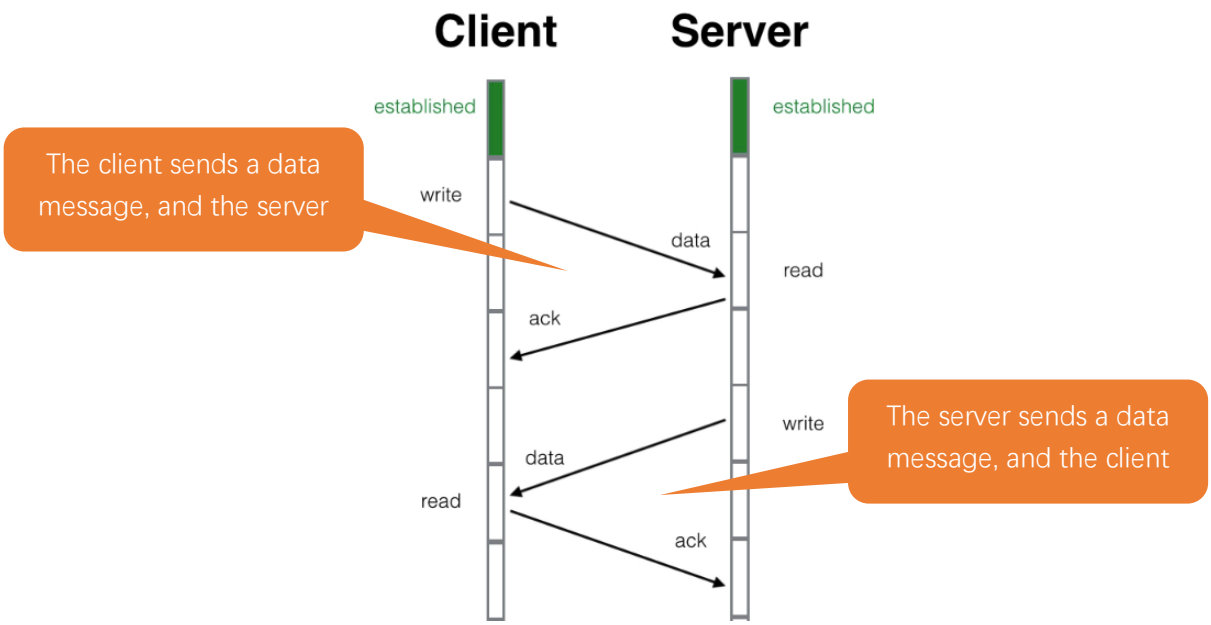
TCP connection

Before transmitting data, TCP needs to establish a logical connection between the sending end and the receiving end. It provides reliable and error-free data transmission between the two computers. In the TCP connection, the client and the server must be clarified. The client sends a connection request to the server, and each time such a request is proposed, a "three-times handshake" is required.

Three-times handshake: In the TCP protocol, during the preparation phase of sending data, the client and the server interact three times to ensure the reliability of the connection, which is called "three-times handshake".
The first handshake, the client sends a connection request to the server and waits for the server to confirm.
The second handshake, the server sends a response back to the client informing that it has received the connection request.
The third handshake, the client sends a confirmation message to the server again to confirm the connection.



TCP is a connection-oriented, low-level transmission control protocol. After TCP establishes a connection, the client and server can send and receive messages to each other, and the connection will always exist as long as the client or server does not initiate disconnection. Each time one party sends a message, the other party will reply with an ack signal.

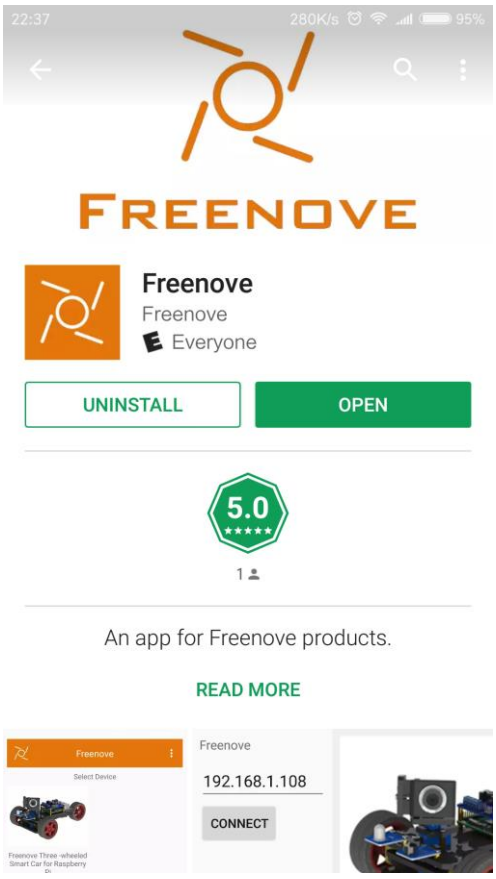


Install Freenove app

There are three ways to install our app.

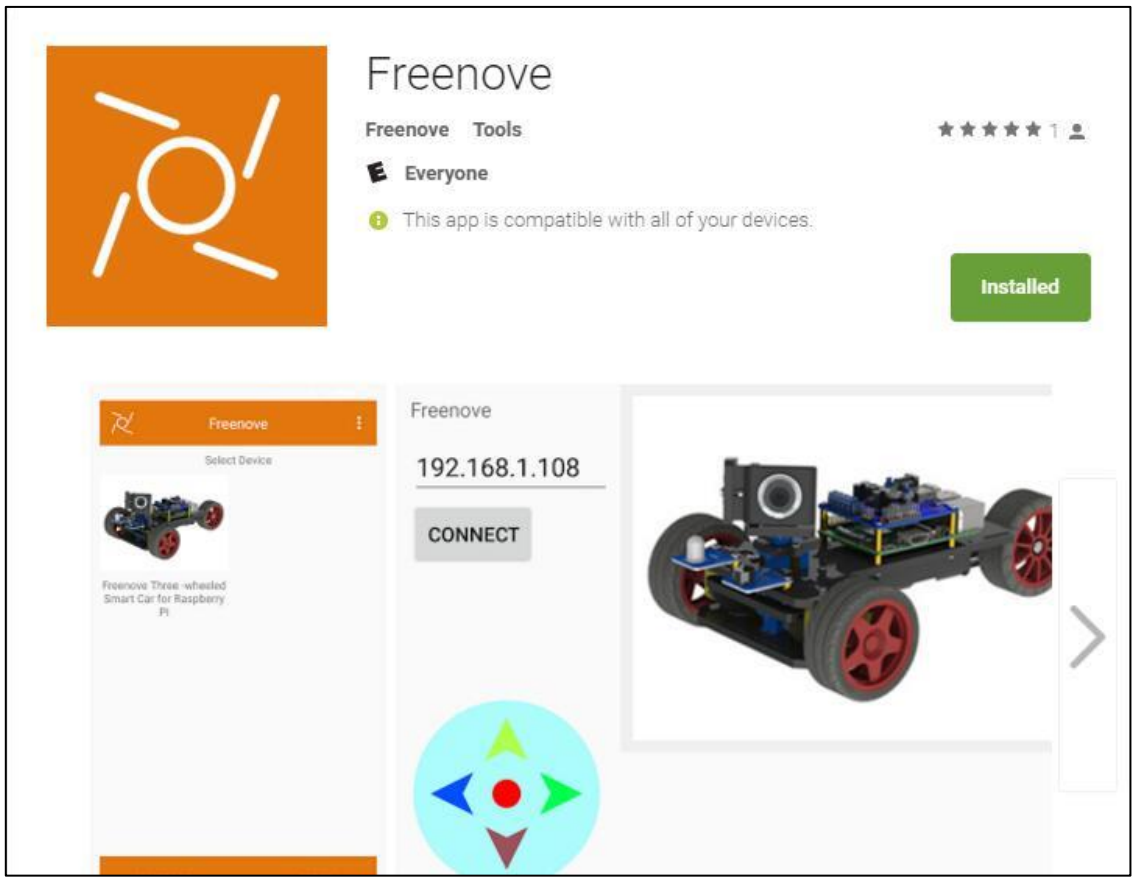
Method 1

Use Google play to search “Freenove”, download and install.



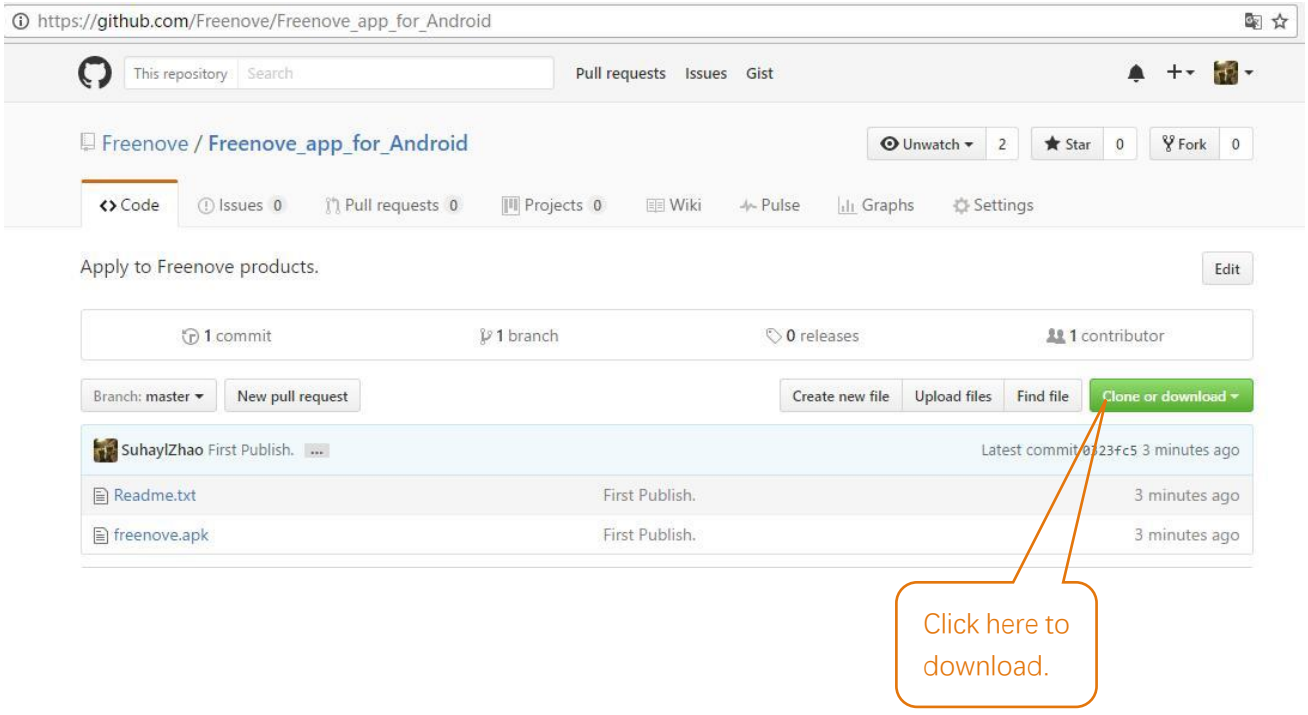
Method 2

Visit https://play.google.com/store/apps/details?id=com.freenove.suhayl.Freenove, and click install.



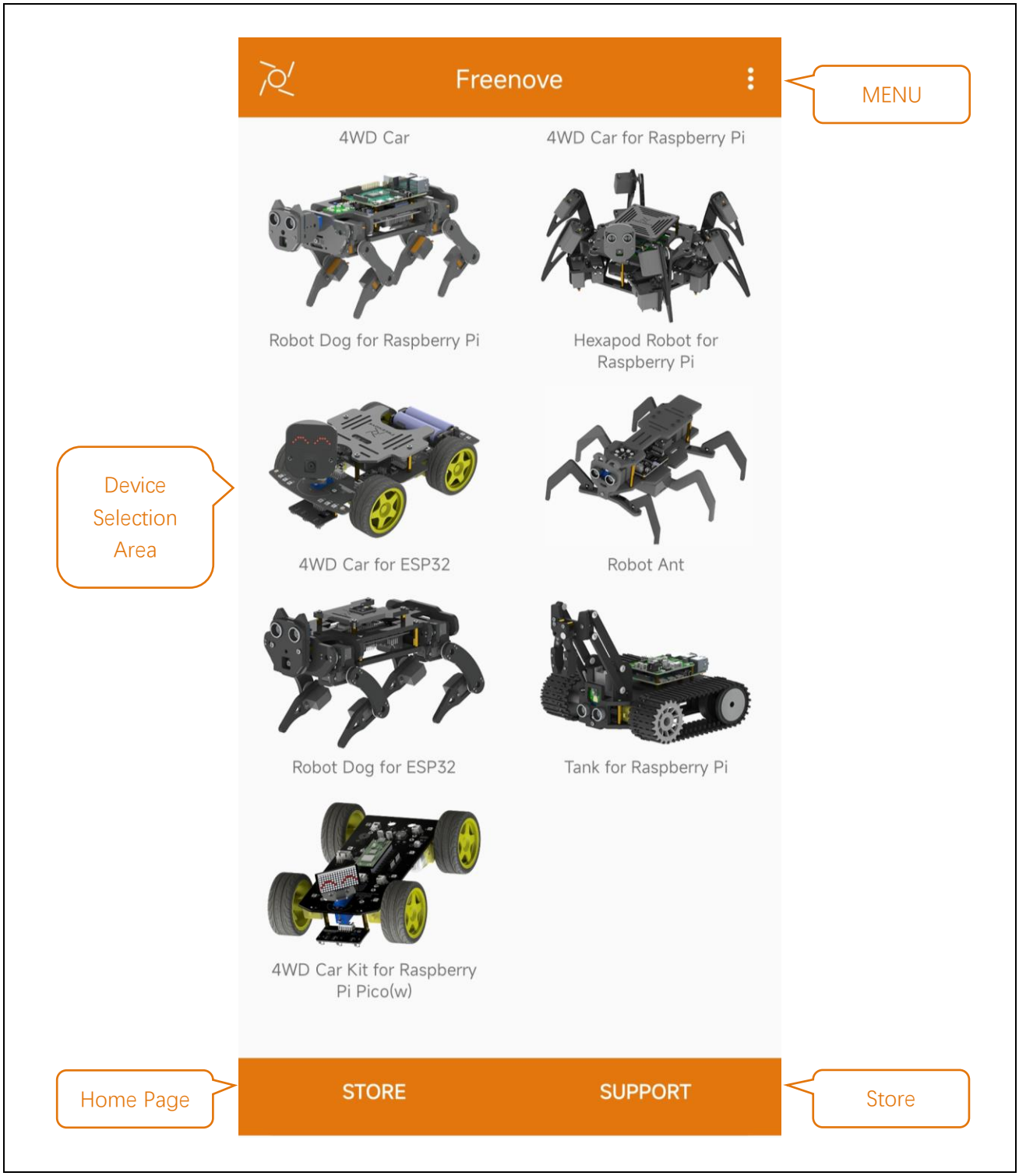
Method 3

Visit https://github.com/Freenove/Freenove_app_for_Android, download the files in this library, and install freenove.apk to your Android phone manually.



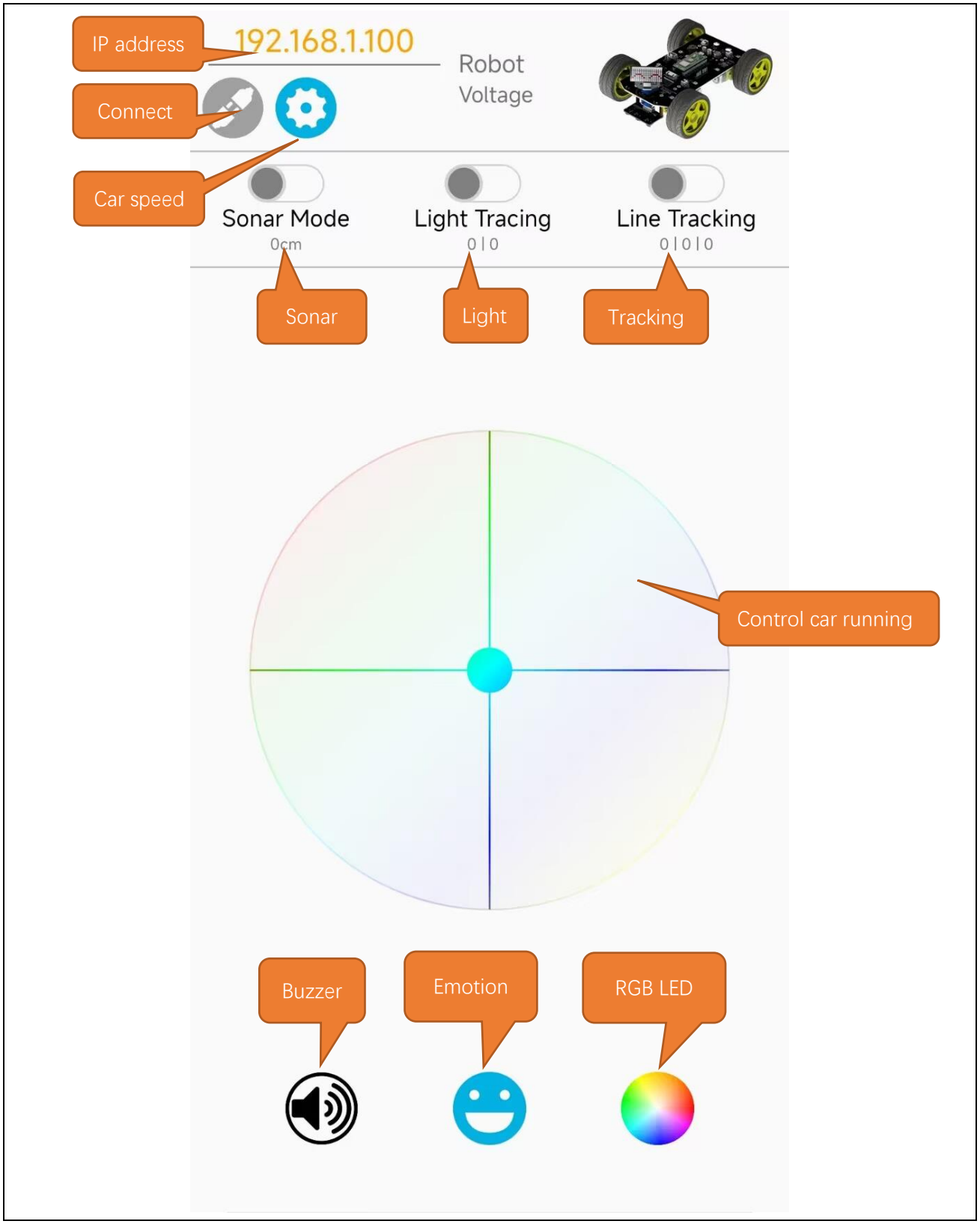
Menu

Open application “Freenove”, as shown below:



Introduction to the APP

In this chapter, we use Freenove 4WD Car for Raspberry Pi Pico W, so it is necessary to understand the interface of this mode.



Sketch

Open "06.1_WiFi_APP_TcpServer" in **"Freenove_4WD_Car_Kit_for_Raspberry_Pi_Pico\Ordinary_wheels\Sketches"** and double click "06.1_WiFi_APP_TcpServer.ino".

Before uploading the code, you need to modify it according to the name and password of your router.

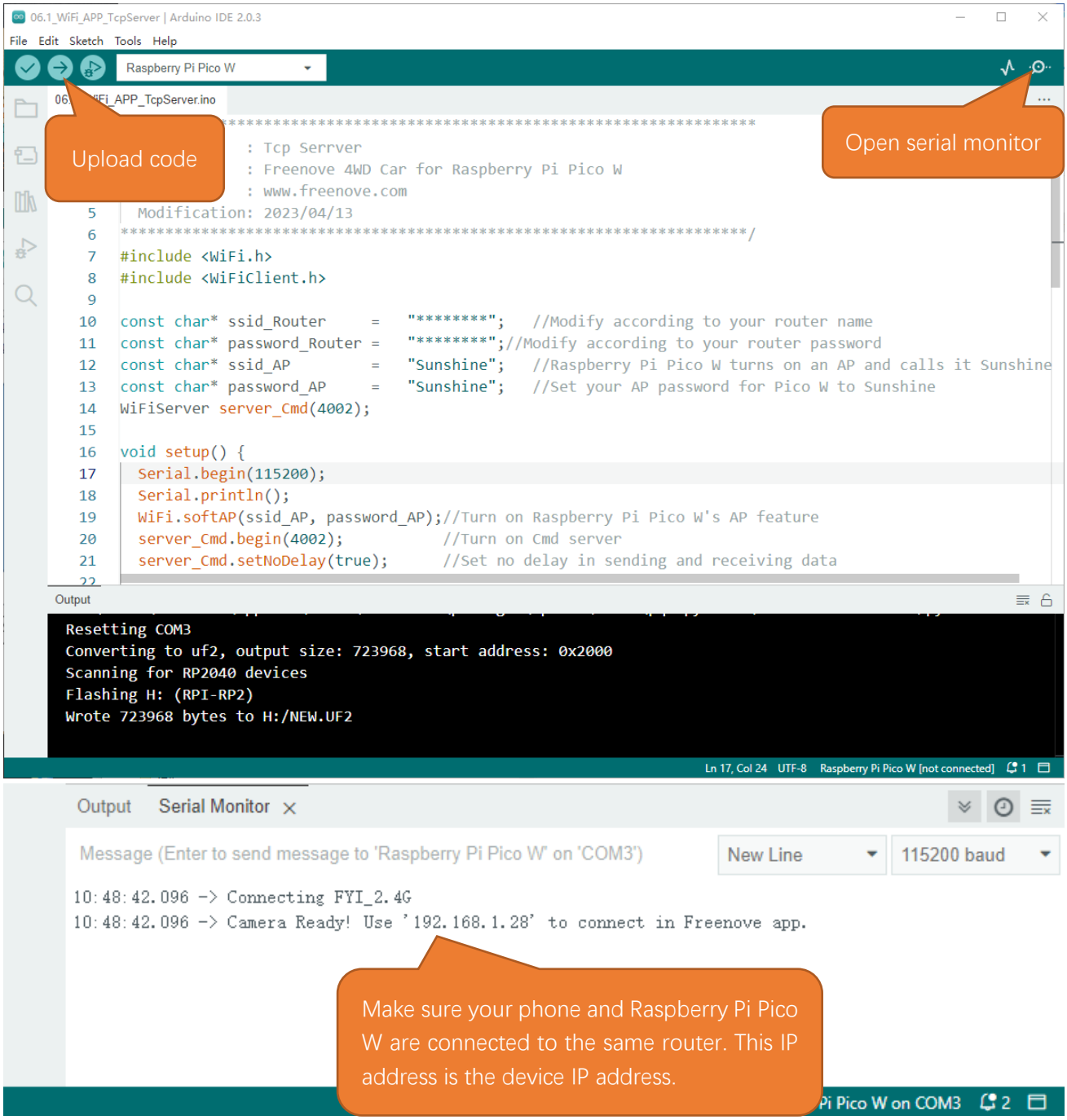
Code

```
#include <WiFi.h>
#include <WiFiClient.h>

const char* ssid_Router     =   "********";//Modify according to your router name
const char* password_Router =   "********";//Modify according to your router password
const char* ssid_AP         =   "Sunshine";// Raspberry Pi Pico W turns on an AP and calls it Sunshine
const char* password_AP     =   "Sunshine";//Set your AP password for Pico W to Sunshine
WiFiServer server_Cmd(4002);

void setup() {
  Serial.begin(115200);
  Serial.println();
  WiFi.softAP(ssid_AP, password_AP);//Turn on Raspberry Pi Pico W's AP feature
  server_Cmd.begin(4000);           //Turn on Cmd server
  server_Cmd.setNoDelay(true);      //Set no delay in sending and receiving data

  WiFi.disconnect(true);
  WiFi.begin(ssid_Router, password_Router);//Make a connector request to the router
  Serial.print("Connecting ");
  Serial.print(ssid_Router);
  int timeout=0;
  while(WiFi.connected() != true) { //If the connection fails, wait half a second for another connection request
    delay(500);
    Serial.print(".");
    timeout++;
    if(timeout==10)
      break;
  }

  Serial.print("\nCamera Ready! Use '");
  //Serial.print(WiFi.softAPIP());
  //Serial.print(" or ");
  Serial.print(WiFi.localIP());
```

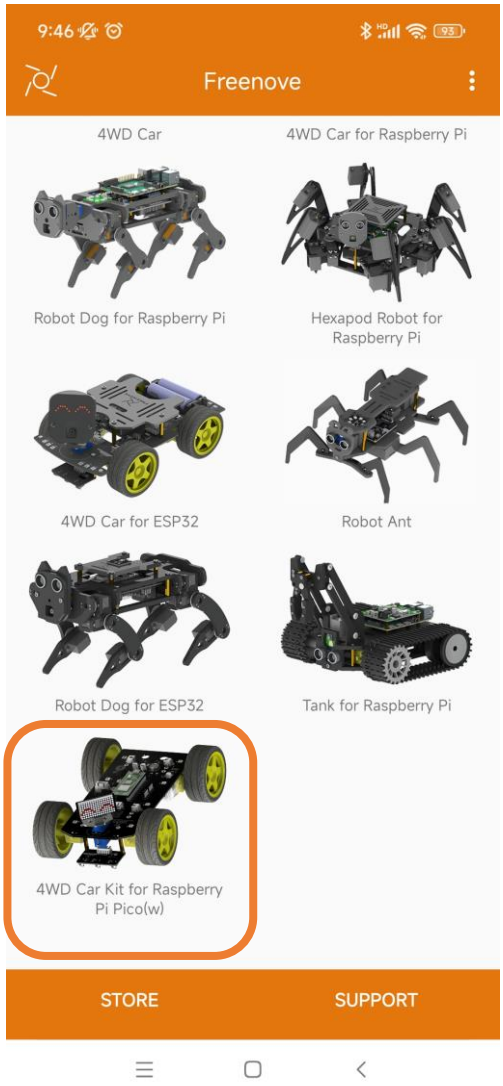
Modify according to the name and password of your

```
35    Serial.println("' to connect in Freenove app.");
36  }
37
38  void loop() {
39    WiFiClient client = server_Cmd.available();//listen for incoming clients
40    if (client) {                             //if you get a client,
41      Serial.println("Command Server connected to a client.");
42      while (client.connected()) {//loop while the client's connected
43        if (client.available()) { //if there's bytes to read from the client,
44          String dataBuffer = client.readStringUntil('\n') + String("\n"); //read data
45          Serial.print(dataBuffer);//print it out the serial monitor
46        }
47      }
48      client.stop();// close the connection:
49      Serial.println("Command Client Disconnected.");
50    }
51  }
```

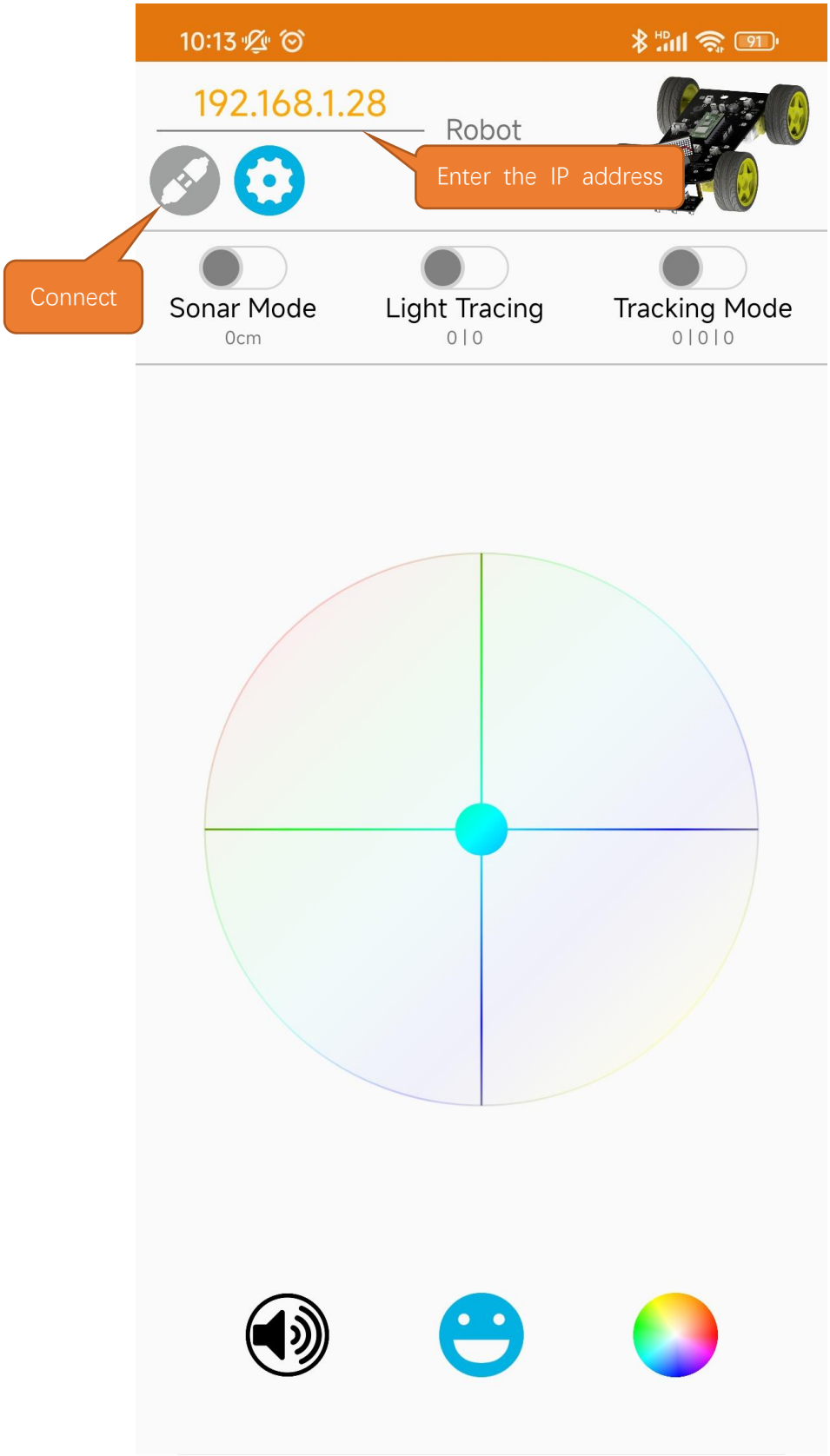
Uplode the code to Raspberry Pi Pico W car and open serial monitor.

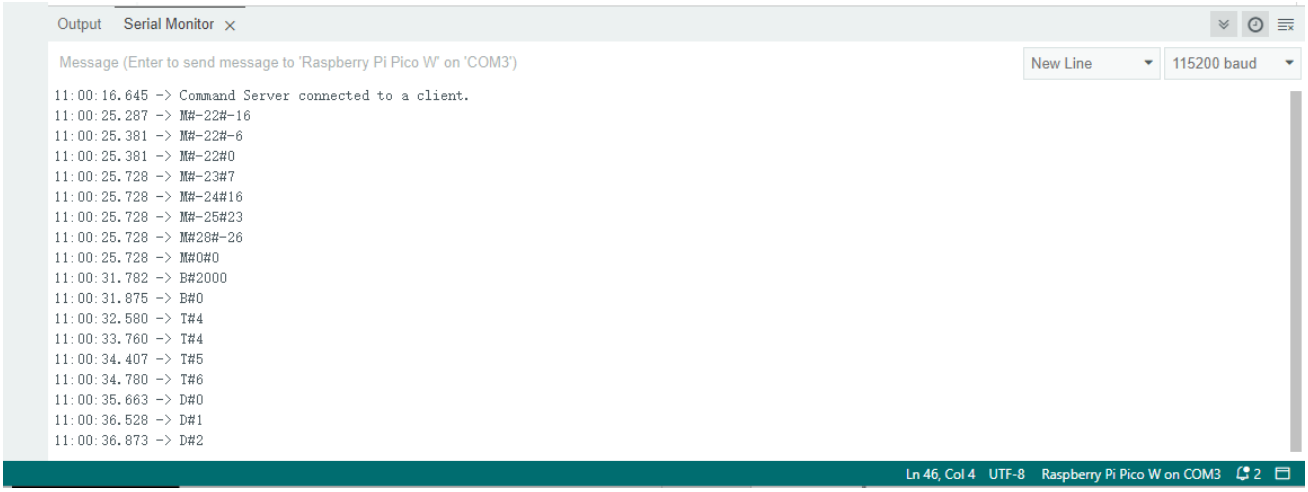


Open mobliephone app and select Freenove 4WD Car for Raspberry Pi Pico (W).



Make sure your mobile phone and Raspberry Pi Pico W car are connected to the same router. According to the aforementioned IP address, enter the corresponding IP address, and then click Connect. Tap the button on the screen and you can see the data on the serial monitor.


10:13
192.168.1.28
Robot
Enter the IP address
Connect
Sonar Mode
0cm
Light Tracing
0 | 0
Tracking Mode
0 | 0 | 0



Code Explanation:

Add the library functions of WiFi. Each time before using WiFi, please add these library functions.

```
1  #include <WiFi.h>
2  #include <WiFiClient.h>
```

Define four pointer variables to store information.

```
4  const char* ssid_Router     =   "********";//Modify according to your router name
5  const char* password_Router =   "********";//Modify according to your router password
6  const char* ssid_AP         =   "Sunshine";// Raspberry Pi Pico W turns on an AP and calls it
   Sunshine
7  const char* password_AP     =   "Sunshine";//Set your AP password for Pico W to Sunshine
```

Turn ON AP feature of Raspberry Pi Pico W. When the function is called, you can search a WiFi signal named “Sunshine” in your phone.

```
14   WiFi.softAP(ssid_AP, password_AP);//Turn on Raspberry Pi Pico W's AP feature
```

Mobile applications use a server for transmitting commands. Set Raspberry Pi Pico W to start a server. The server with port 4002 is used to send commands. Call the setNoDelay function to send and receive data directly.

```
14   server_Cmd.begin(4002);          //Turn on Cmd server
15   server_Cmd.setNoDelay(true);     //Set no delay in sending and receiving data
```

Connect Raspberry Pi Pico W to the router. After the router is successfully connected, Raspberry Pi Pico W will print the IP address information to the serial monitor.

```
22   int timeout=0;
23   while(WiFi.connected() != true) { //If the connection fails, wait half a second for another
   connection request
24     delay(500);
25     Serial.print(".");
26     timeout++;
27     if(timeout==10)
28       break;
29   }
30
31   Serial.print("\nCamera Ready! Use '");
32   //Serial.print(WiFi.softAPIP());
```

```
33    //Serial.print(" or ");
34    Serial.print(WiFi.localIP());
35    Serial.println("' to connect in Freenove app.");
```

Define a WiFi client object to monitor whether the server has a client requesting access signal.

```
39  WiFiClient client = server_Cmd.available();//listen for incoming clients
```

Determine whether there is a client connected to the server.

```
42    while (client.connected()) {//loop while the client's connected
```

Call available function to query whether the client has sent data to the server.

```
43      if (client.available()) { //if there's bytes to read from the client,
```

Call readStringUntil function to read a line of data from the client and print it to the serial monitor.

```
44          String dataBuffer = client.readStringUntil('\n') + String("\n"); //read data
45          Serial.print(dataBuffer);//print it out the serial monitor
```

Call stop() function to close the client connection.

```
48      client.stop();// close the connection:
```

Reference

Class Station

Every time when using WiFi, you need to include header file "WiFi.h.".

begin(ssid, password,channel, bssid, connect): Raspberry Pi Pico W is used as Station to connect hotspot.

ssid: WiFi hotspot name

password: WiFi hotspot password

channel: WiFi hotspot channel number; communicating through specified channel; optional parameter

bssid: mac address of WiFi hotspot, optional parameter

connect: blloean optional parameter, defaulting to true. If set as false, then Pico W won't connect WiFi.

config(local_ip, gateway, subnet, dns1, dns2): set static local IP address.

local_ip: station fixed IP address.

subnet: subnet mask

dns1,dns2: optional parameter, defines IP address of domain name server

status: obtain the connection status of WiFI

local IP(): obtian IP address in Station mode

disconnect(): disconnect wifi

setAutoConnect(boolen): set automatic connection Every time Raspberry Pi Pico W is powered on, it will connect WiFi automatically.

setAutoReconnect(boolen): set automatic reconnection Every time Raspberry Pi Pico W disconnects from WiFi, it will reconnect to WiFi automatically.

Class AP

Every time when using WiFi, you need to include header file "WiFi.h.".

softAP(ssid, password, channel, ssid_hidden, max_connection):

ssid: WiFi hotspot name

password: WiFi hotspot password

channel: Number of WiFi connection channels, range 1-13. The default is 1.

ssid_hidden: Whether to hide WiFi name from other devices. The default is not hide.

max_connection: Maximum number of WiFi connected devices. The range is 1-4. The default is 4.

softAPConfig(local_ip, gateway, subnet): set static local IP address.

local_ip: station fixed IP address.

Gateway: gateway IP address

Subnet: subnet mask

softAP(): obtian IP address in AP mode

softAPdisconnect (): disconnect AP mode.

Class Client

Every time when using Client, you need to include header file "WiFi.h."

connect(ip, port, timeout)/connect(*host, port, timeout): establish a TCP connection.

ip, *host: ip address of target server

port: port number of target server

timeout: connection timeout

connected(): judge whether client is connecting. If return value is 1, then connect successfully; If return value is 0, then fail to connect.

stop(): stop tcp connection

print(): send data to server connecting to client

available(): return to the number of bytes readable in receive buffer, if no, return to 0 or -1.

read(): read one byte of data in receive buffer

readString(): read string in receive buffer

Class Server

Every time use Server functionality, we need to include header file"WiFi.h".

WiFiServer(uint16_t port=80, uint8_t max_clients=4): create a TCP Server.

port: ports of Server; range from 0 to 65535 with the default number as 80.

max_clients: maximum number of clients with default number as 4.

begin(port): start the TCP Server.

port: ports of Server; range from 0 to 65535 with the default number as 0.

setNoDelay(bool nodelay): whether to turn off the delay sending functionality.

nodelay: true stands for forbidden Nagle algorithm.

close(): close tcp connection.

stop(): stop tcp connection.

7.2 WiFi Car by APP

Based on the previous sector, we now integrate other functions with WiFi, so that most of the car's functions can be controlled with phone APP.

Sketch

Open the folder "06.2_Multi_Functional_Car" in Freenove_4WD_Car_Kit_for_Raspberry Pi Pico\Ordinary_wheels\Sketches and double click "06.2_Multi_Functional_Car.ino".
Before compiling the code, please modify the code according to the name and password of your WiFi.
To enable Raspberry Pi Pico W's own router functionality, set line 37 to WiFi_Setup(1); to connect Raspberry Pi Pico W to the router, set line 37 to WiFi_Setup(0).
In the code we provide, the default is: WiFi_Setup(0).
At this point, you need to make sure your phone and the Raspberry PI Pico W car are connected to the same router.

Code

```
#include <Arduino.h>
#include <Adafruit_NeoPixel.h>
#include <WiFi.h>
#include <WiFiClient.h>
#include <Wire.h>
#include "Freenove_4WD_Car_WiFi.h"
#include "Freenove_4WD_Car_Emotion.h"
#include "Freenove_4WD_Car_WS2812.h"
#include "Freenove_4WD_Car_For_Pico_W.h"

#define UPLOAD_VOL_TIME 3000
long int lastUploadVoltageTime;
#define UPLOAD_LIGHTVOL_TIME 500
long int lastUploadLIGHTADCTime;
#define UPLOAD_LINEVOL_TIME 200
long int lastUploadLINEVOLTime;
long int lastUploadSONARVOLTime;
int buzzer_frequency = 0;
int CAR_MODE_VOL = 0;
int LASt_CAR_MODE_VOL = 0;

char CmdArray[8];
int paramters[8];
int sendOnceModuleCheck = 1;
int cmdFlag;

```

```
//WiFi parameter setting
char ssid_Router[] =     "********";       //Modify according to your router name
char password_Router[] = "********";       //Modify according to your router password
char ssid_AP[] =         "Sunshine";       //Pico W turns on an AP and calls it Sunshine
char password_AP[] =     "Sunshine";       //Set your AP password for Pico W to Sunshine
WiFiServer server_Cmd(4002);

void setup() {
  Buzzer_Setup();                  //Buzzer initialization
  Serial.begin(115200);
  WiFi_Setup(0);                   // Set mode. When the parameter is 1, AP mode is enabled.
If the parameter is 0, the STA mode is enabled.
  server_Cmd.begin(4002);          //Start the command server
  server_Cmd.setNoDelay(true);     //Set no delay in sending and receiving data
  Wire.begin();
  Motor_Setup();                   //Motor initialization
  Servo_Setup();                   //Servo initialization
  WS2812_Setup();                  //WS2812 initialization
  Emotion_and_Ultrasonic_Setup();  //Emotion initialization or Ultrasonic initialization
  Photosensitive_Setup();          //Light initialization
  Track_Setup();                   //Track initialization
  Servo_1_Angle(90);
  delay(500);
}

void loop() {
  WiFiClient client = server_Cmd.available();  //listen for incoming clients
  if (client) {                                //if you get a client
    Serial.println("Cmd_Server connected to a client.");
    while (client.connected()) {
      if (sendOnceModuleCheck == 1) {
        if (Check_Module_value == SONAR_IS_ESIST)
          cmdFlag = SONAR_IS_ESIST;
        else
          cmdFlag = MATRIX_IS_EXIST;
        sendOnceModuleCheck = 0;
        client.print("A#" + String(cmdFlag) + "\n");
      }                                //loop while the client's connected
      if (client.available()) {        //if there's bytes to read from the client
        String inputStringTemp = client.readStringUntil('\n');  //Read the command by WiFi
        Serial.println(inputStringTemp);            //Print out the command received by WiFi
        int string_length = inputStringTemp.length() + 1;
        char str[string_length];
        inputStringTemp.toCharArray(str, string_length);
```

```
70          Get_Command(str);
71          if (CmdArray[0] == CMD_LED_MOD)  //Set the display mode of car colored lights
72          {
73            WS2812_SetMode(paramters[1]);
74          }
75          if (CmdArray[0] == CMD_LED)  //Set the color and brightness of the car lights
76          {
77            WS2812_Set_Color_1(paramters[1], paramters[2], paramters[3], paramters[4]);
78          }
79          if (CmdArray[0] == CMD_MATRIX_MOD)  //Set the display mode of the LED matrix
80          {
81            if (Check_Module_value == MATRIX_IS_EXIST) {
82              Emotion_SetMode(paramters[1]);  //Display
83              Serial.print(" \n matrix is exist !!! \n");
84            }
85            if (Check_Module_value != MATRIX_IS_EXIST) {
86              client.print("A#" + String(SONAR_IS_ESIST) + "\n");
87              Serial.print(" \n sonar is exist !!! \n");
88            }
89          }
90          if (CmdArray[0] == CMD_BUZZER)  //Buzzer control command
91          {
92            tone(2, paramters[1]);
93          }
94          if (CmdArray[0] == CMD_POWER) {  //Power query command
95            float battery_voltage = Get_Battery_Voltage();
96            client.print(CMD_POWER);
97            client.print(INTERVAL_CHAR);
98            client.print(battery_voltage);
99            client.print(ENTER);
100         }
101         if (CmdArray[0] == CMD_MOTOR) {  //Network control car movement command
102           Car_SetMode(0);
103           if (paramters[1] == 0 && paramters[2] == 0) {
104             Motor_Move(0, 0);  //Stop the car
105           } else               //If the parameters are not equal to 0
106           {
107             Motor_Move(paramters[1], paramters[2]);
108           }
109         }
110         if (CmdArray[0] == CMD_SERVO) {  //Network control servo motor movement command
111           Servo_1_Angle(paramters[1]);
112         }
113         if (CmdArray[0] == CMD_CAR_MODE) {  //Car command Mode
```

```
            oa_CalculateVoltageCompensation();
            if (paramters[1] == CAR_MODE_LIGHT_TRACING)  //Light seeking car command
            {
              LASt_CAR_MODE_VOL = 1;
              Car_SetMode(1);
            } else if (paramters[1] == CAR_MODE_LINE_TRACKING)  //Tracking car command
            {
              LASt_CAR_MODE_VOL = 1;
              Car_SetMode(2);
            } else if (paramters[1] == CAR_MODE_SONAR)  //Ultrasonic car command
            {
              if (Check_Module_value == SONAR_IS_ESIST) {
                LASt_CAR_MODE_VOL = 1;
                Car_SetMode(CAR_MODE_SONAR);
              }
              if (Check_Module_value != SONAR_IS_ESIST) {
                Car_SetMode(CAR_MODE_MANUAL);
                client.print("A#" + String(MATRIX_IS_EXIST) + "\n");
              }
            } else if (paramters[1] == 0) {
              LASt_CAR_MODE_VOL = 2;
              Car_SetMode(0);
            }
            if (CAR_MODE_VOL != LASt_CAR_MODE_VOL) {
              if (CAR_MODE_VOL == 1) {
                emotion_task_mode = 0;  //Display
                Motor_Move(0, 0);  //Stop the car
                Servo_1_Angle(90);
                delay(100);
                isLightModeFirstStarting = true;
              }
              CAR_MODE_VOL = LASt_CAR_MODE_VOL;
            }
          }
          //Clears the command array and parameter array
          memset(CmdArray, 0, sizeof(CmdArray));
          memset(paramters, 0, sizeof(paramters));
        }
        if (millis() - lastUploadVoltageTime > UPLOAD_VOL_TIME) {
          float battery_voltage = Get_Battery_Voltage();
          client.print("P#" + String(battery_voltage) + "\n");
          lastUploadVoltageTime = millis();
        }
        Emotion_and_Ultrasonic_Setup();
```

```
158         if (Check_Module_value == MATRIX_IS_EXIST) {
159           Emotion_Show(emotion_task_mode);  //Led matrix display function
160         } else if (Check_Module_value == SONAR_IS_ESIST) {
161         }
162         WS2812_Show(ws2812_task_mode);  //Car color lights display function
163         Car_Select(carFlag);            //Pico W Car mode selection function
164         if (carFlag == CAR_MODE_LIGHT_TRACING) {
165           if (millis() - lastUploadLIGHTADCTime > UPLOAD_LIGHTVOL_TIME) {
              float leftADCVoltage = getLeftPhotosensitiveADCValue();
166           float rightADCVoltage = getRightPhotosensitiveADCValue();
167           client.print("C#1#" + String(leftADCVoltage) + "#" + String(rightADCVoltage) +
168 "\n");
169           lastUploadLIGHTADCTime = millis();
170         }
171       }
172       if (carFlag == CAR_MODE_LINE_TRACKING) {
            if (millis() - lastUploadLINEVOLTime > UPLOAD_LINEVOL_TIME) {
173           Track_Read();
174           client.print("C#2#" + String(sensorValue[0]) + String(sensorValue[1]) +
175 String(sensorValue[2]) + "\n");
176           lastUploadLINEVOLTime = millis();
177         }
178       }
179       if (carFlag == CAR_MODE_SONAR && Check_Module_value == SONAR_IS_ESIST) {
180         float distance = Get_Sonar();
181         client.print("C#3#" + String(distance) + "\n");
182         lastUploadSONARVOLTime = millis();
183       }
184     }
185     client.stop();  //close the connection:
186     Serial.println("Command Client Disconnected.");
187     sendOnceModuleCheck = 1;
188     // rp2040.restart();
189
190   }
191 }
192
193 void Get_Command(char *string) {
194   char *token = strtok(string, INTERVAL_CHAR);
195   CmdArray[0] = token[0];        //Put the command into an array
196   for (int i = 0; i < 5; i++) {
197     if (token != NULL) {
198       token = strtok(NULL, INTERVAL_CHAR);
199     }
```

```
200     paramters[I + 1] = atoi(token); //Put the paramters into an array and convert them to
201 integers
202   }
203 }
```

Code Explanation:

Add the driver header files of the car.

The three header files starting with "WiFi" contain the WiFi driver configuration function of Raspberry Pi Pico W.

Freenove_4WD_Car_WiFi.h Freenove_4WD_Car_WiFi.h contains command information for WiFi signal transmission and camera pin configuration for Raspberry Pi Pico W.

Freenove_4WD_Car_Emotion.h contains the driver configuration function of the LED matrix module.

Freenove_4WD_Car_WS2812.h contains the driver configuration function of the RGB LEDs.

Freenove_4WD_Car_For_Pico_W.h contains the driver configuration function of the car.

```
1 #include <Arduino.h>
2 #include <Adafruit_NeoPixel.h>
3 #include <WiFi.h>
4 #include <WiFiClient.h>
5 #include <Wire.h>
6 #include "Freenove_4WD_Car_WiFi.h"
7 #include "Freenove_4WD_Car_Emotion.h"
8 #include "Freenove_4WD_Car_WS2812.h"
9 #include "Freenove_4WD_Car_For_Pico_W.h"
```

Before each code upload, you can change the following ssid_Router and password_Router based on your WiFi name and password. The program uses STA mode by default. You need to make sure your phone is connected to the same router as your Raspberry PI Pico W car.

When you use AP mode, change the code to WiFi_Setup(1), and the Raspberry PI Pico W will create a WiFi named "Sunshine" corresponding to the WiFi configuration. Note: Sunshine is not connected to the Internet. So if you choose to connect it, your phone will disconnect from the Internet.

```
28   char ssid_Router[] =     "********";    //Modify according to your router name
29   char password_Router[] = "********";    //Modify according to your router password
30   char ssid_AP[] =         "Sunshine";    //Pico W turns on an AP and calls it Sunshine
31   char password_AP[] =     "Sunshine";    //Set your AP password for Pico W to Sunshine
```

Open a TCP server port. Port 4002 is used to receive commands from the car.

```
33 WiFiServer server_Cmd(4002);
```

Call the WiFi_Init() to initialize the WiFi parameters. Call the WiFi_Setup() to initialize the Raspberry Pi Pico W's WiFi.

AP mode starts when the WiFi_Setup parameter is 1, and STA mode when the parameter is 0. After Raspberry Pi Pico W starts WIFI, it prints the IP information on the serial monitor.

```
37 WiFi_Setup(0);                    // Set mode. When the parameter is 1, AP mode is enabled. If
   the parameter is 0, the STA mode is enabled.
```

Initialize the car's buzzer, LED matrix module, RGB LEDs, WiFi, motor, servo, photoresistor and line tracking modules.

```
34 void setup() {
35   Buzzer_Setup();                  //Buzzer initialization
```

```
36    Serial.begin(115200);
37    WiFi_Setup(0);                    // Set mode. When the parameter is 1, AP mode is enabled.
    If the parameter is 0, the STA mode is enabled.
38    server_Cmd.begin(4002);           //Start the command server
39    server_Cmd.setNoDelay(true);      //Set no delay in sending and receiving data
40    Wire.begin();
41    Motor_Setup();                    //Motor initialization
42    Servo_Setup();                    //Servo initialization
43    WS2812_Setup();                   //WS2812 initialization
44    Emotion_and_Ultrasonic_Setup();   //Emotion initialization or Ultrasonic initialization
45    Photosensitive_Setup();           //Light initialization
46    Track_Setup();                    //Track initialization
47    Servo_1_Angle(90);
48    delay(500);
49  }
```

Monitor TCP server port 4002. If a client connects to this port, Raspberry PI Pico W will print messages of WiFi connection on the serial port. When a client connects to this port, it receives information from the client until the client disconnects. When Raspberry PI Pico W detects that the client is disconnected, it closes the TCP connection and prints a WiFi disconnect prompt message.

```
52    WiFiClient client = server_Cmd.available();  //listen for incoming clients
53    if (client) {                               //if you get a client
54      Serial.println("Cmd_Server connected to a client.");
55      while (client.connected()) {
...       ...
181     }
182     client.stop();  //close the connection:
183     Serial.println("Command Client Disconnected.");
184     sendOnceModuleCheck = 1;
185     // rp2040.restart();
186   }
```

Get_Command(), the command parsing function. Every time this function is used, the command received by TCP will be sent to this function through inputStringTemp, and the command and parameters will be parsed and stored in the global variable array CmdArray and paramters.

```
193  void Get_Command(char *string) {
194    char *token = strtok(string, INTERVAL_CHAR);
195    CmdArray[0] = token[0];         //Put the command into an array
196    for (int i = 0; i < 5; i++) {
197      if (token != NULL) {
198        token = strtok(NULL, INTERVAL_CHAR);
199      }
200      paramters[I + 1] = atoi(token); //Put the paramters into an array and convert them to
201  integers
202    }
```

```
203 }
```

Judge the client's data and use different functions of the car according to different commands.

```
71          if (CmdArray[0] == CMD_LED_MOD)  //Set the display mode of car colored lights
...           ...
75          if (CmdArray[0] == CMD_LED)  //Set the color and brightness of the car lights
...           ...
79          if (CmdArray[0] == CMD_MATRIX_MOD)  //Set the display mode of the LED matrix
...                 ...
90          if (CmdArray[0] == CMD_BUZZER)  //Buzzer control command
...           ...
94          if (CmdArray[0] == CMD_POWER)  //Power query command
...           ...
101         if (CmdArray[0] == CMD_MOTOR)  //Network control car movement command
...           ...
110         if (CmdArray[0] == CMD_SERVO) {  //Network control servo motor movement command
...           ...
113         if (CmdArray[0] == CMD_CAR_MODE) {  //Car command Mode
...           ...
```

The following three functions run in a non-blocking manner. Call the Emotion_Show() function to control the LED matrix, call the WS2812_Show() function to control the RGB LEDs of the car, and call the Car_Select() function to control the mode of the car.

```
159         Emotion_Show(emotion_task_mode);  //Led matrix display function
...         ...
162       WS2812_Show(ws2812_task_mode);  //Car color lights display function
163       Car_Select(carFlag);            //Pico W Car mode selection function
```

7.3 WiFi Car by PC

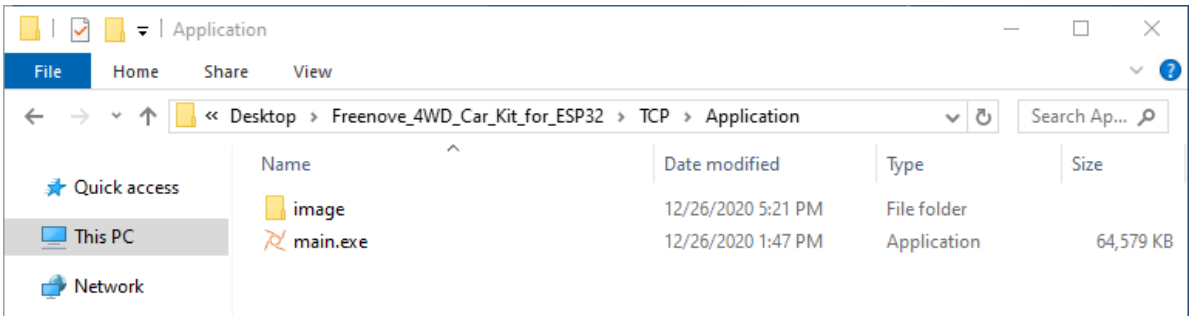
In addition to using a mobile phone to control the WiFi car, we also provide users with a host computer to control the car. In this chapter, the car still uses the code in section 7.2

Run client on windows system

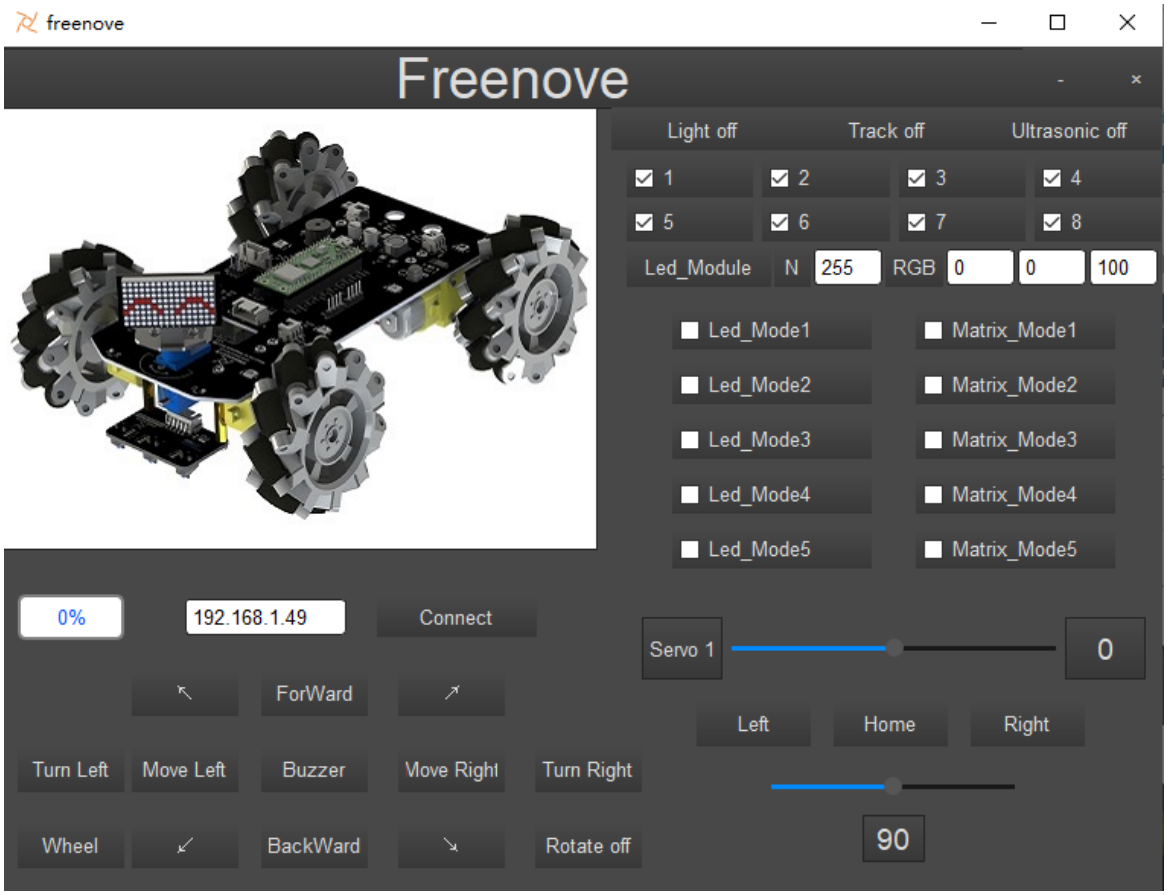
There are two ways to run client on Windows.

Option 1 Run the executable file directly.

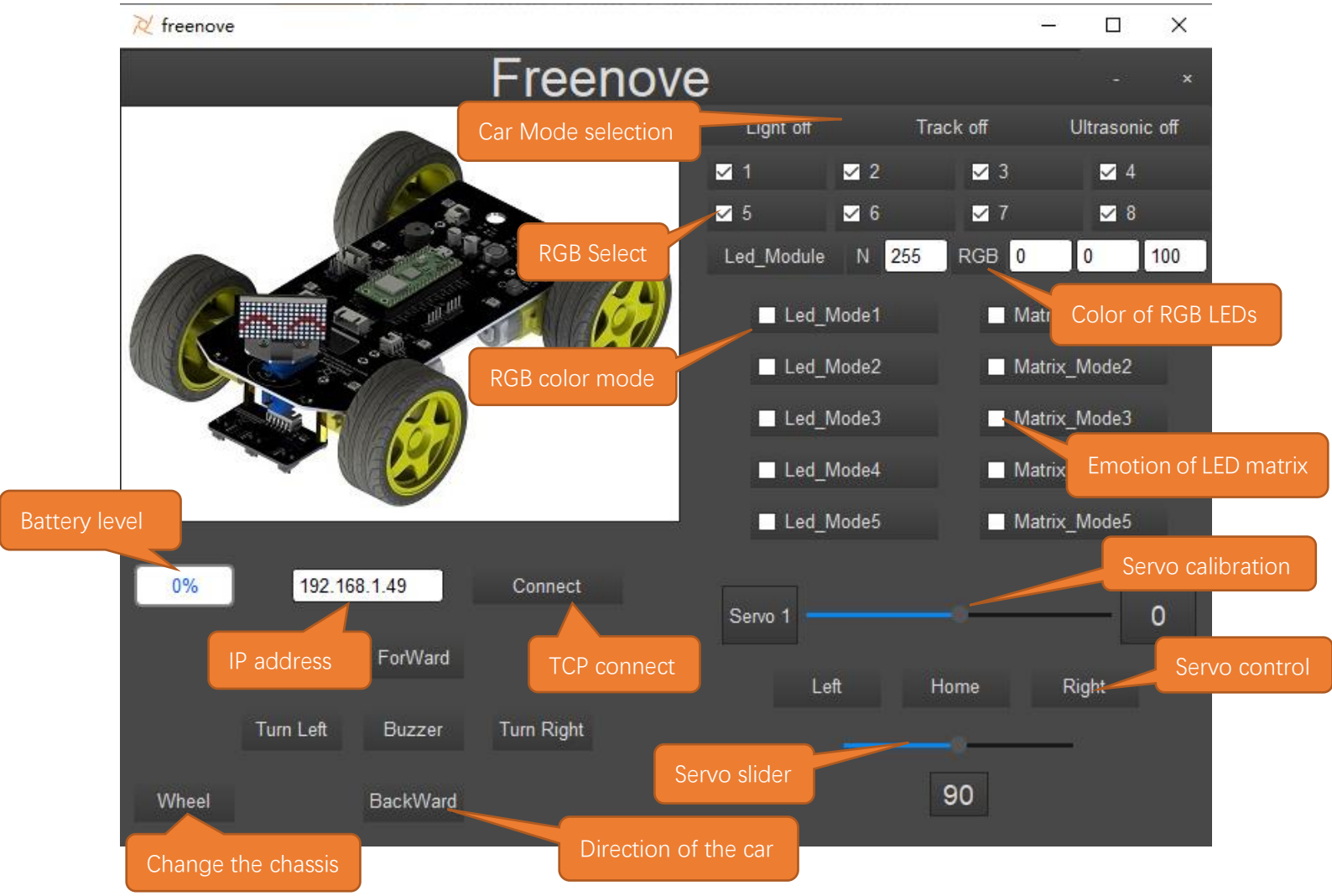
Open main.exe in Freenove_4WD_Car_Kit_for_Raspberry_Pi_Pico\TCP\Application.



The interface of client is as follows:



Note: You Can See the control end of the car, but this time need to click on the lower right corner of the button “Wheel” to enter our car control interfaceThe function of the client is as follows：



According to the prompt message printed on the Raspberry Pi Pico W serial monitor, enter the IP address into "IP Address" and click "Connect", and you can control the Raspberry Pi Pico W car through this client

Option 2 Install python3 and some related python libraries to run client.

If you want to modify the client, please follow this section.

Install python3

Download and install Python3 package.

https://www.python.org/downloads/windows/

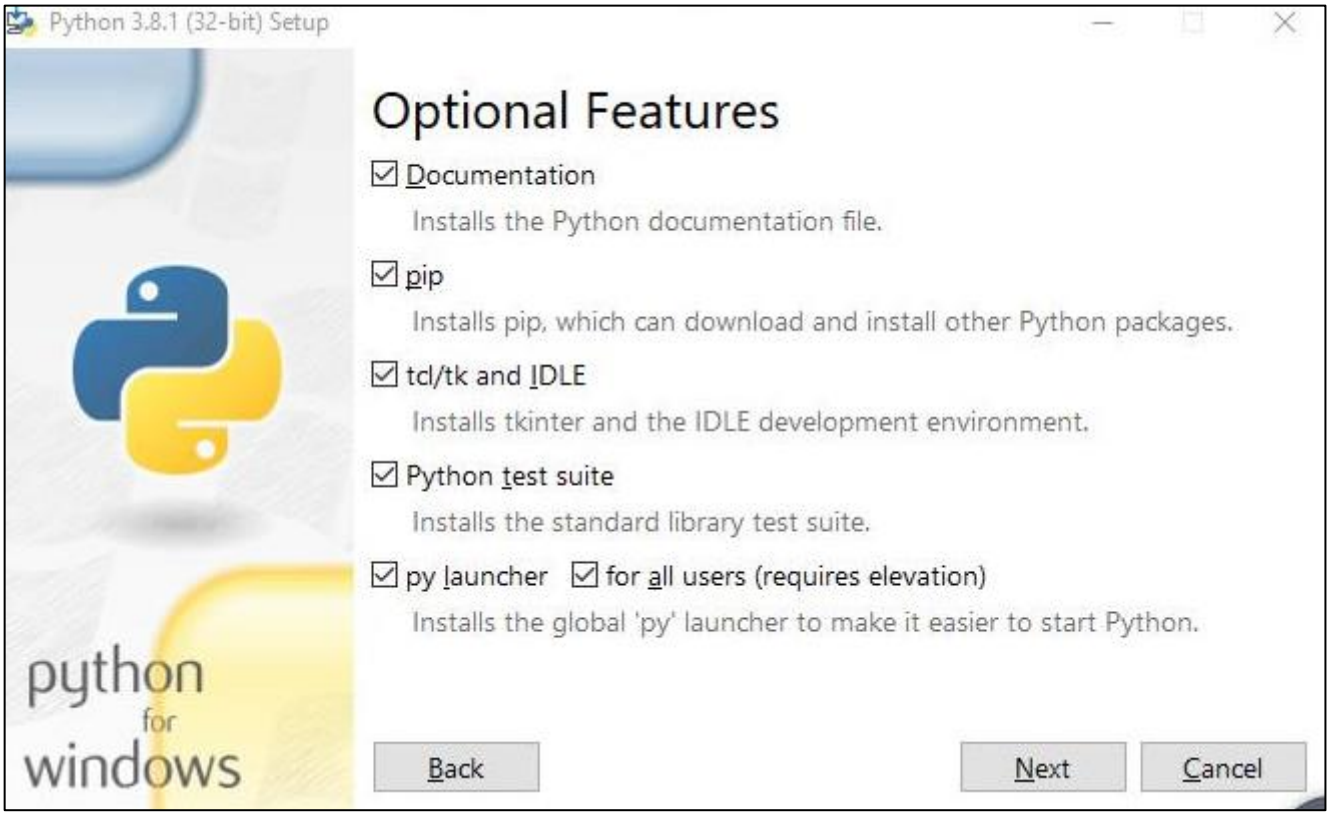
Select Python3.8.1.

Version	Operating System	Description
Gzipped source tarball	Source release	
XZ compressed source tarball	Source release	
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later
Windows help file	Windows	
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64
Windows x86 embeddable zip file	Windows	
Windows x86 executable installer	Windows	
Windows x86 web-based installer	Windows	

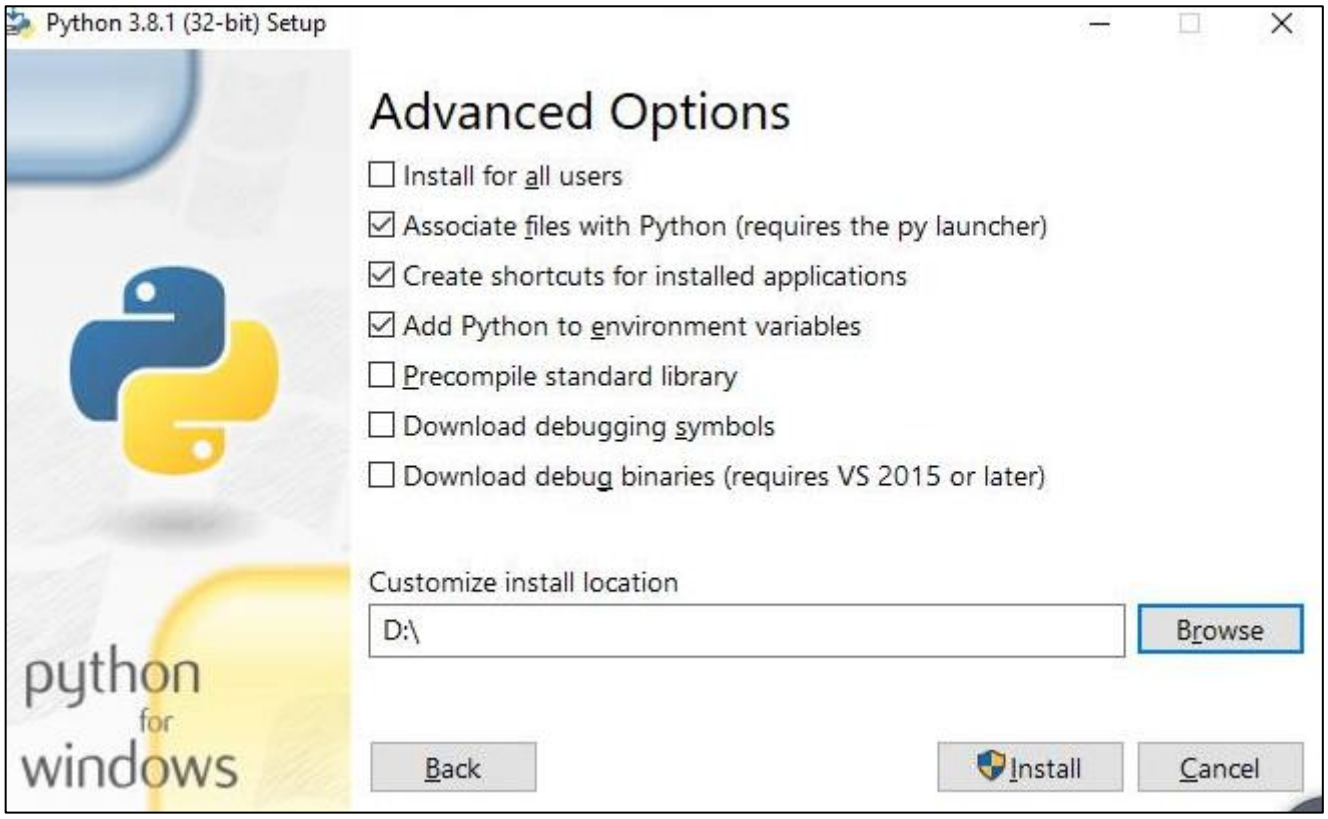
Click Windows x86 executable installer to download. After downloading, click to install.

Please note that "Add Python 3.8 to PATH" MUST be check.

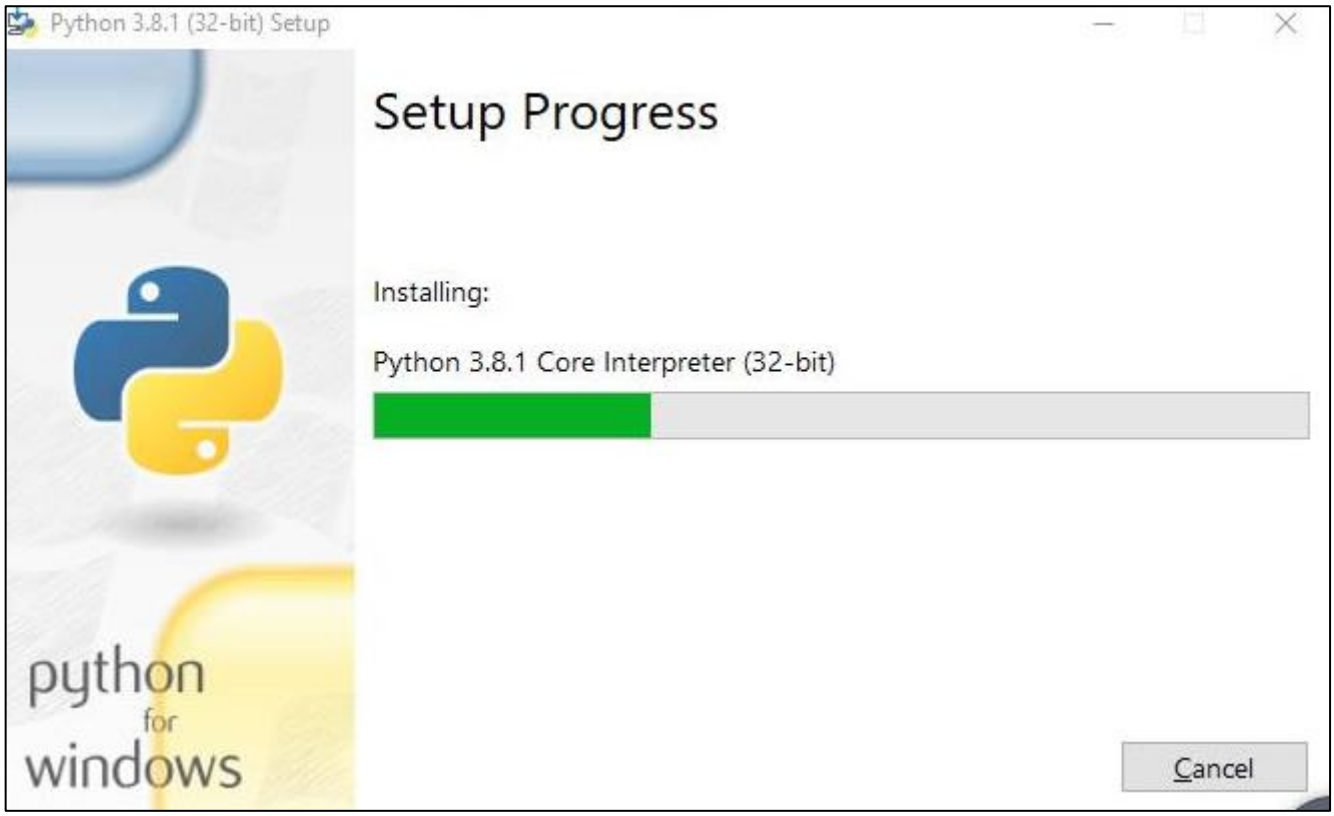
Check all the options and then click "Next".



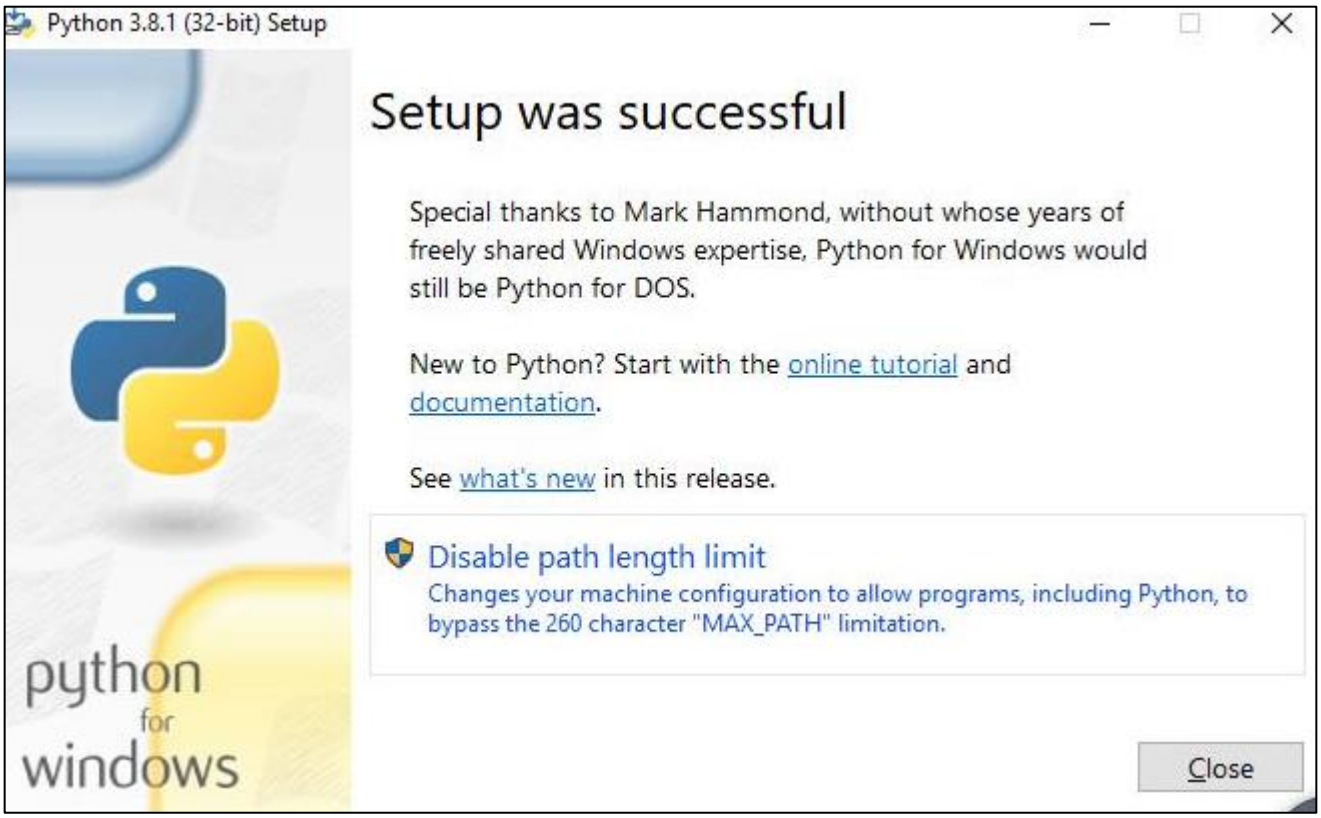
Here you can select the installation path of Python. We install it at D drive. If you are a novice, you can select the default path.



Wait for it to finish installing.

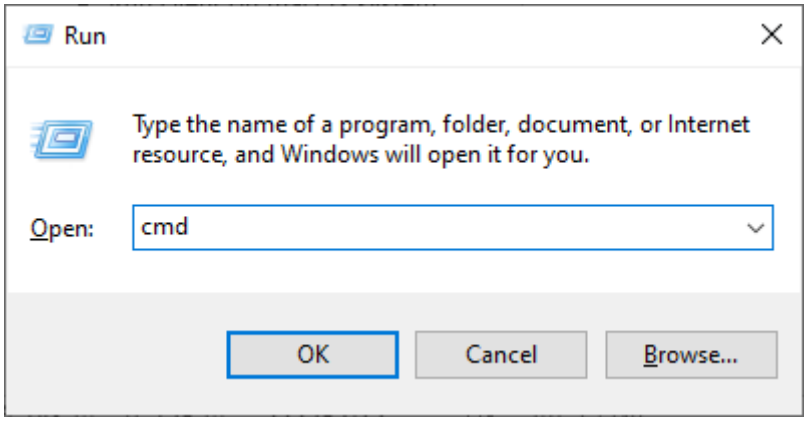


Now the installation is finished.



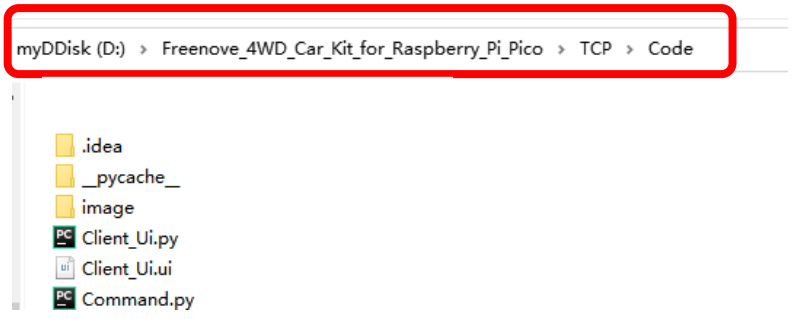
Install PyQt5、opencv、numpy and other libraries.

Press “Ctrl” and “R” on the keyboard at the same time will pop up a window.



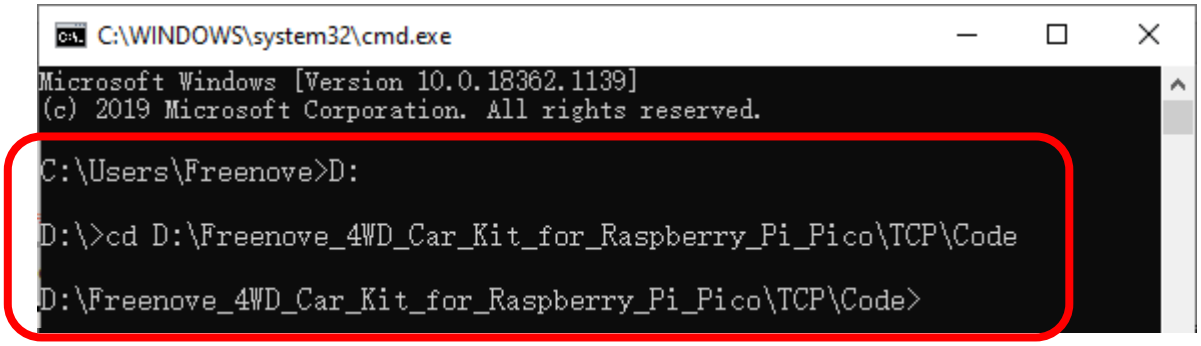
Enter **cmd** in the pop-up window and click “OK”.

Use the command to enter the storage location of Freenove_4WD_Car_Kit_for_Raspberry_Pi_Pico Here we assume it is saved in D drive.

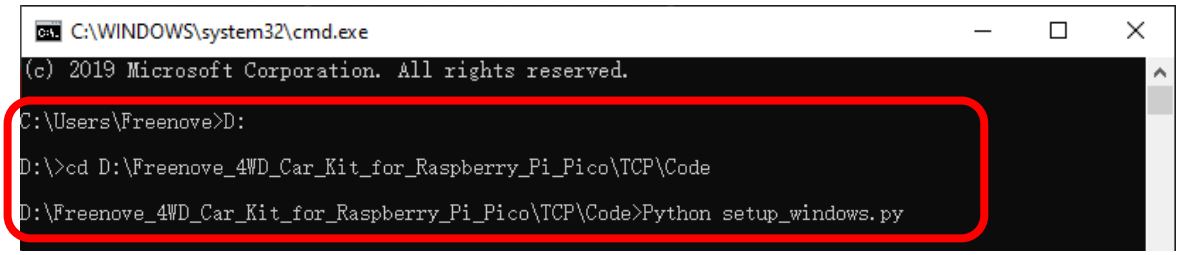


The address depends on where you saved Freenove_4WD_Car_Kit_for_Raspberry_Pi_Pico. You need to modify it based on your own storage location.

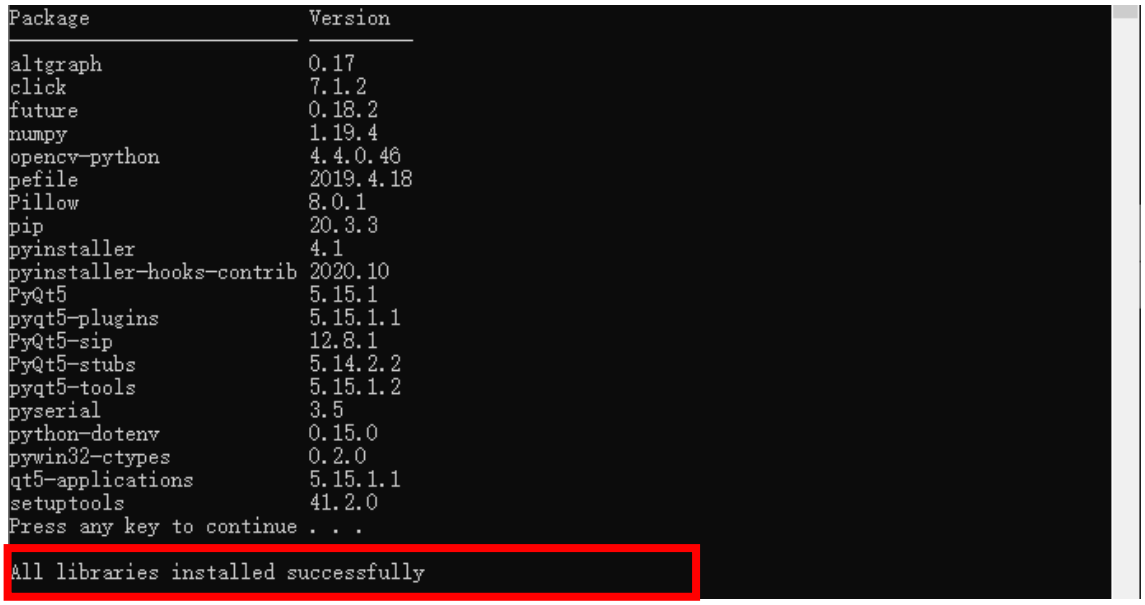
1. Input “D:”, press enter to enter D drive, and then input “cd D:\Freenove_4WD_Car_Kit_for_Raspberry_Pi_Pico\TCP\Code”, press enter to enter the folder.



2. Input “Python setup_windows.py” and press enter. If it fails, you can input “Python3 setup_windows.py” to try again.



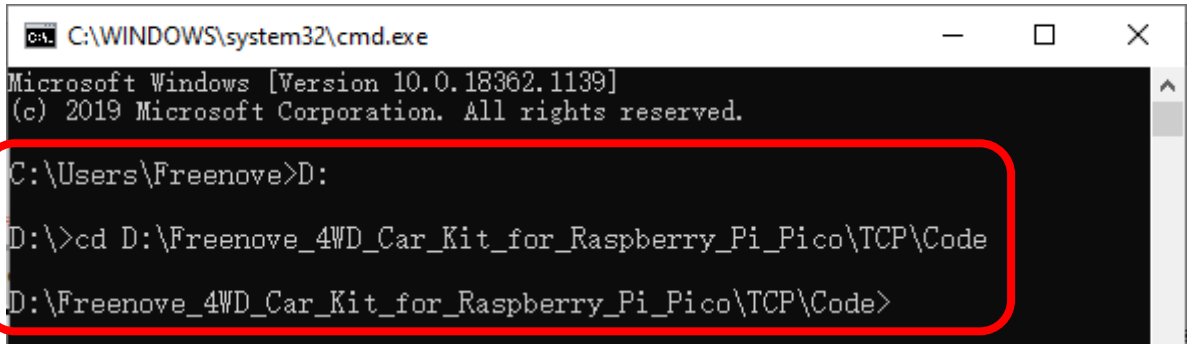
3. Wait for it to finish installing and press any key according to the prompt. When it displays “All libraries installed successfully”, it indicates the environment is installed successfully.



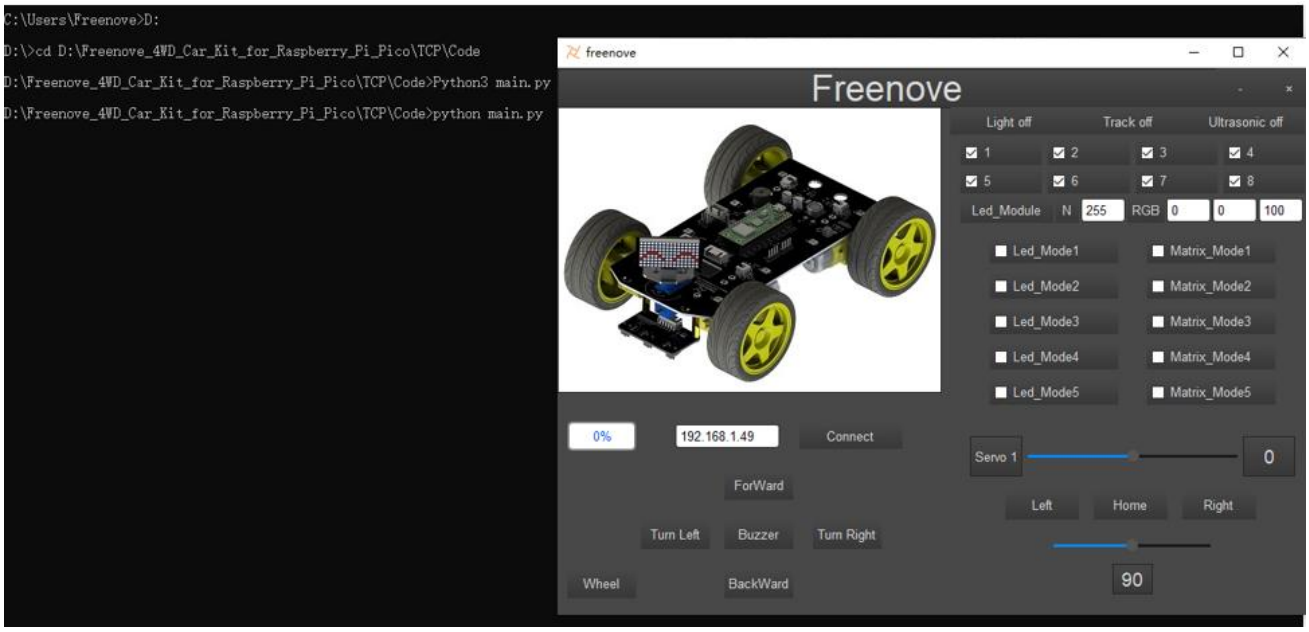
If not all libraries are installed successfully, it will prompt "Some libraries have not been installed. Please run Python3 setup_windows.py again', and then you need to enter and execute the command again. Most installation failures are caused by poor networks. You can Check your network before installation.

Run main.py

1. Input “D:”, press enter to enter D drive and then input “cd D:\Freenove_4WD_Car_Kit_for_Raspberry_Pi_Pico\TCP\Code”, press enter to enter the folder.



2. Input “Python main.py” and press enter. If it fails, you can input “Python3 main.py” to try again.

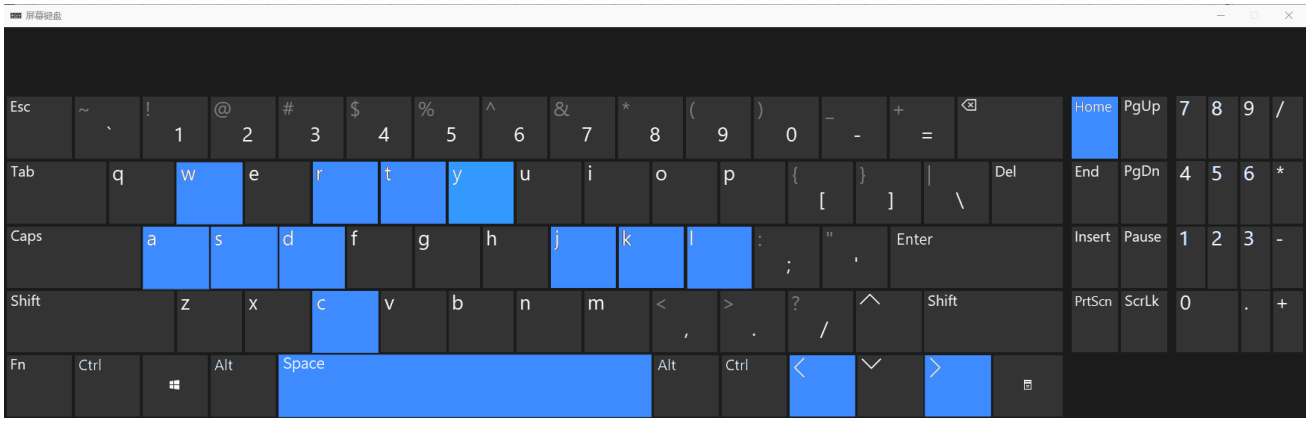


After the program runs, you can control the car through this window.

If you want to modify the code, you can modify the files in TCP folder.

Control the car with keyboard

The car can be controlled by clicking the client. And it can also be controlled by pressing keys on your keyboard.



The following is the corresponding operation of the buttons and keys.

Button on Client	Key	Action
ForWard	W	Move
BackWard	S	Back off
Turn Left	A	Turn left
Turn Right	D	Turn right
Left	left arrow	Turn servo left
Right	right arrow	Turn servo right
Home	Home	Turn servo back Home
Connect/ Disconnect	C	On/off Connection
Buzzer	Space	On/off Buzzer
Led_Mode 1,2,3,4,5	L	Switch RGB Led Mode
Matrix_Mode 1,2,3,4,5	K	Switch Matrix Mode
Matrix_Static Mode	J	Switch Matrix Static Mode
Light Car Mode	R	On/off Light Car Mode
Track Car Mode	T	On/off Track Car Mode
Ultrasonic Car Mode	Y	On/Off Ultrasonic Car Mode

The function of SliderBar is below:

SliderBar	Function
Servo 1	SliderBar Servo 1 is used to slightly adjust the angle. If the servo is not fully centered during installation, you can slightly tune it via the SliderBar.

Other control information:

Control	Function
IP address Edit box	Enter IP address of Raspberry Pi Pico W.
Power box	Show power level.
N,R,G,B Edit box	Control the color of LED selected.
Button “Light On/Off”	Turn on the light seeking function of the car.
Button “Track On/Off”	Turn on the tracking function of the car.
Button“Ultrasonic On/Off”	Turn on the car's obstacle avoidance function.

Run client on macOS system

Here we take MacOS 10.13 as an example. To run client on MacOS, you need to install some software and libraries. MacOS 10.13 comes with python2 but not python3. However, the projects in this program need to be run with python3, so you need to install it first.

Install python3

Download installation package, link: https://www.python.org/downloads/

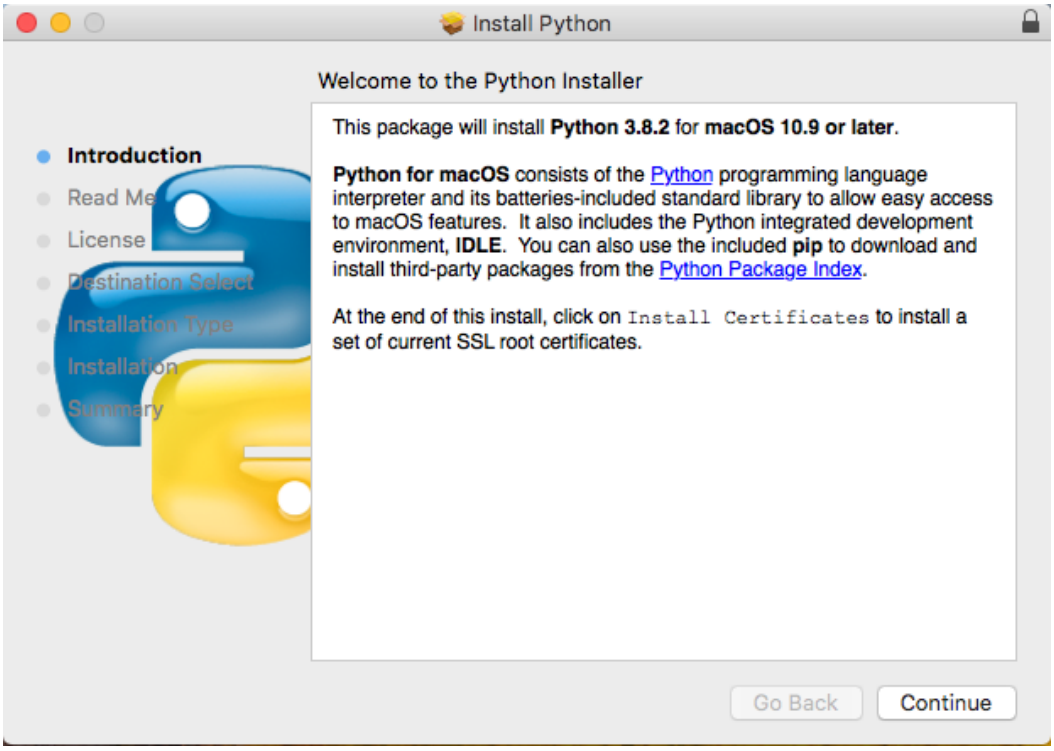
Click Python 3.8.2.

Release version	Release date	
Python 3.8.2	Feb. 24, 2020	Download
Python 3.8.1	Dec. 18, 2019	Download
Python 3.7.6	Dec. 18, 2019	Download
Python 3.6.10	Dec. 18, 2019	Download
Python 3.5.9	Nov. 2, 2019	Download
Python 3.5.8	Oct. 29, 2019	Download

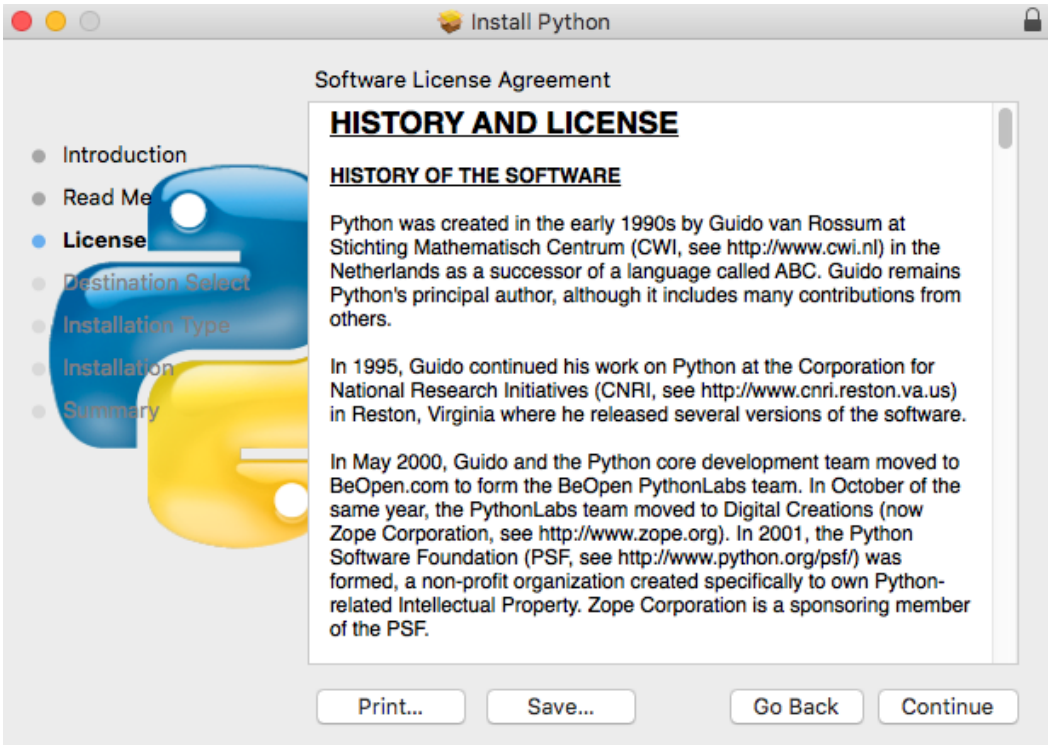
At the bottom of the page, click macOS 64-bit installer and download installation package.

Version	Operating System	Description
Gzipped source tarball	Source release	
XZ compressed source tarball	Source release	
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later
Windows help file	Windows	
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64T/x64
Windows x86-64 executable installer	Windows	for AMD64/EM64T/x64
Windows x86-64 web-based installer	Windows	for AMD64/EM64T/x64
Windows x86 embeddable zip file	Windows	
Windows x86 executable installer	Windows	
Windows x86 web-based installer	Windows	

Click Continue.

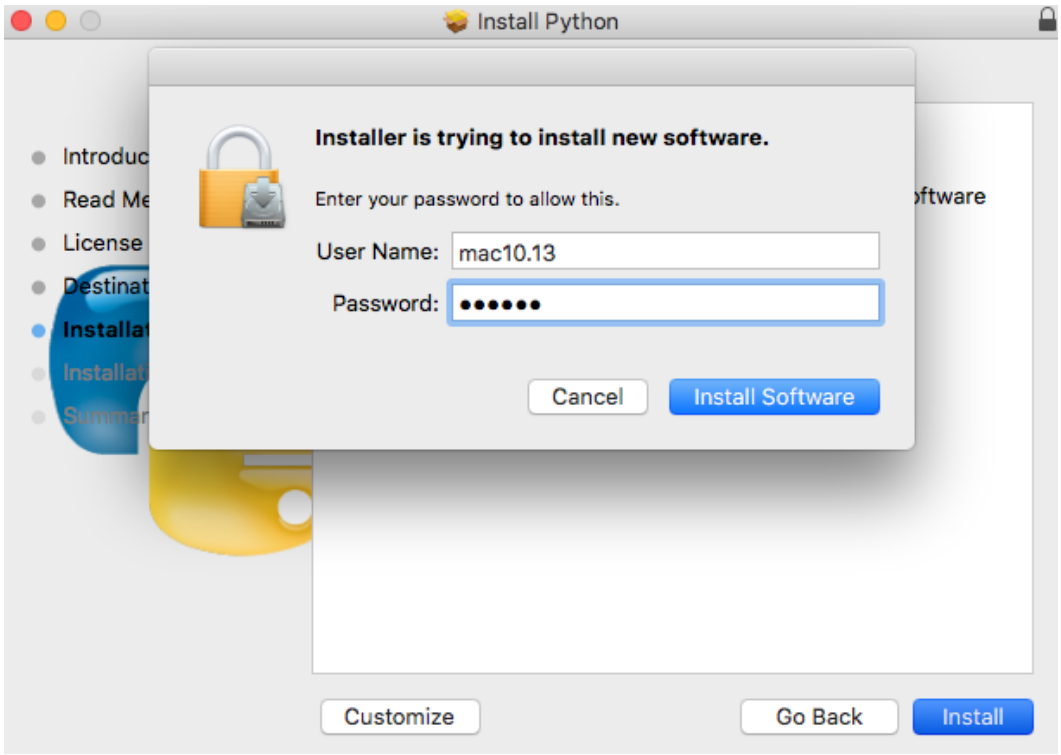


Click Continue

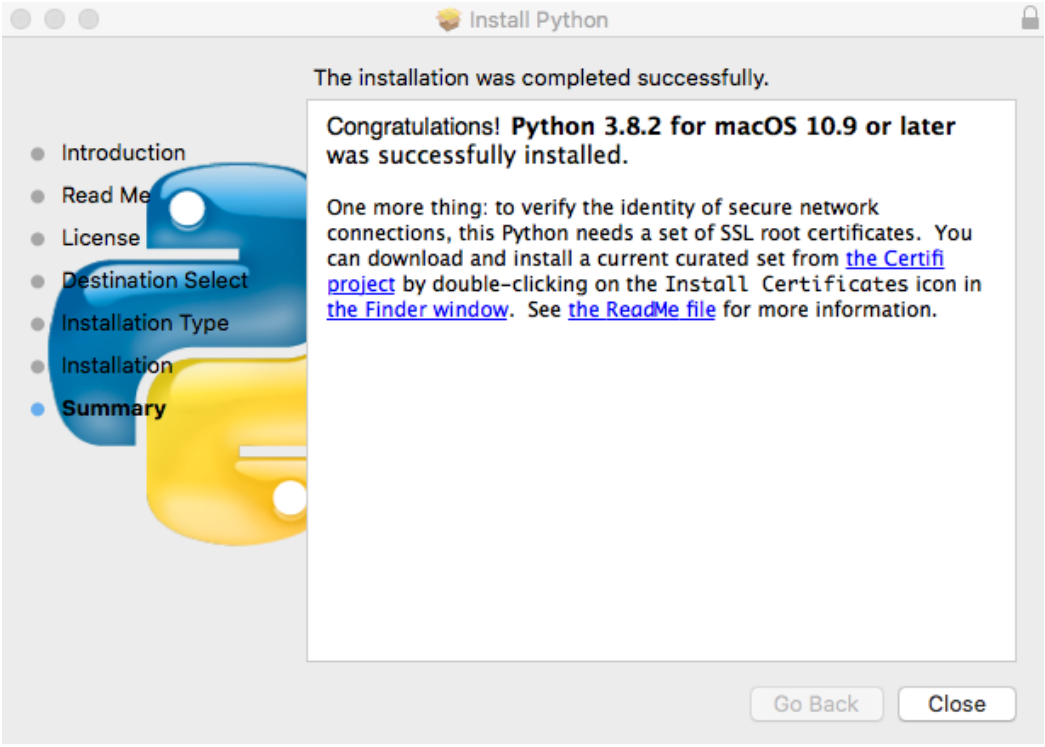


Click Agree.

Click Install. If your computer has a password, enter the password and Install Software.



Now the installation succeeds.



You can find it in Applications.

Install PyQt5、opencv、numpy and other libraries

If you have not yet downloaded the code of Raspberry Pi Pico W car to your macOS, you can download it from the following link: https://github.com/Freenove/Freenove_4WD_Car_Kit_for_Raspberry_Pi_Pico.git
After downloading, you can find it in **Downloads** of Macintosh HD.
Open **Terminal**.

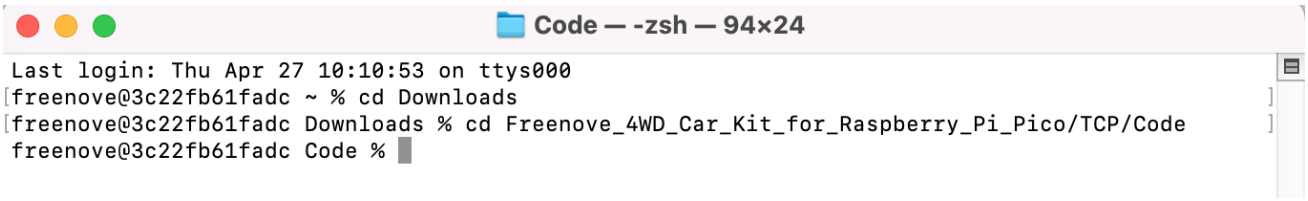
Enter the following commands in Terminal

1. Enter "**Downloads**" (where the code locates). If your code locates in a different path, you need to enter the location of your code.

```
cd Downloads
```

2. Enter the directory where setup_mac.py locates

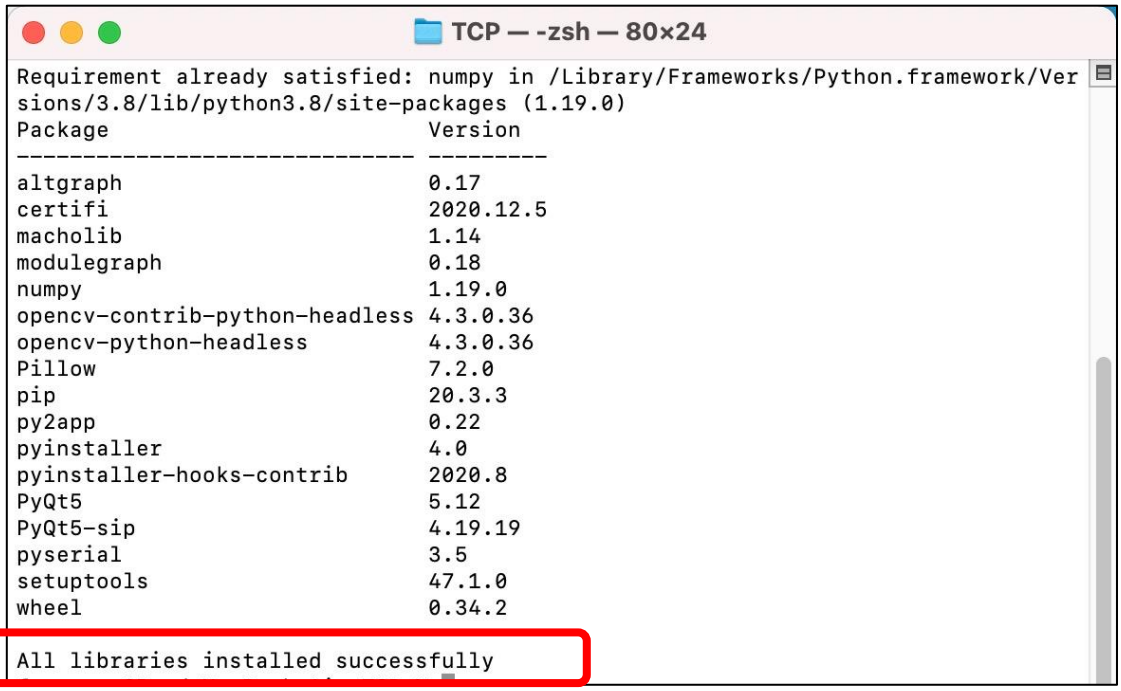
```
cd Freenove_4WD_Car_Kit_for_Raspberry_Pi_Pico/TCP/Code
```



3. Run setup_macos.py:

```
python3 setup_macos.py
```

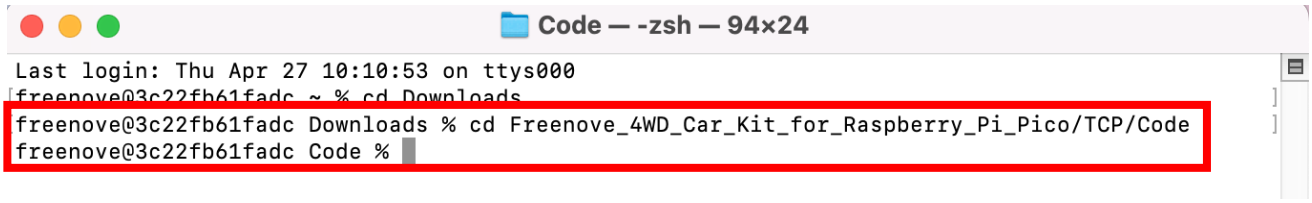
The installation takes some time. Please wait with patience. When installs successfully, it will print "All libraries installed successfully":



Note: If not all libraries are installed successfully, it will prompt “Some libraries have not been installed”. Please input and run Python3 setup_macos.py again. Most installation failures are caused by poor networks. You can Check your network before installation

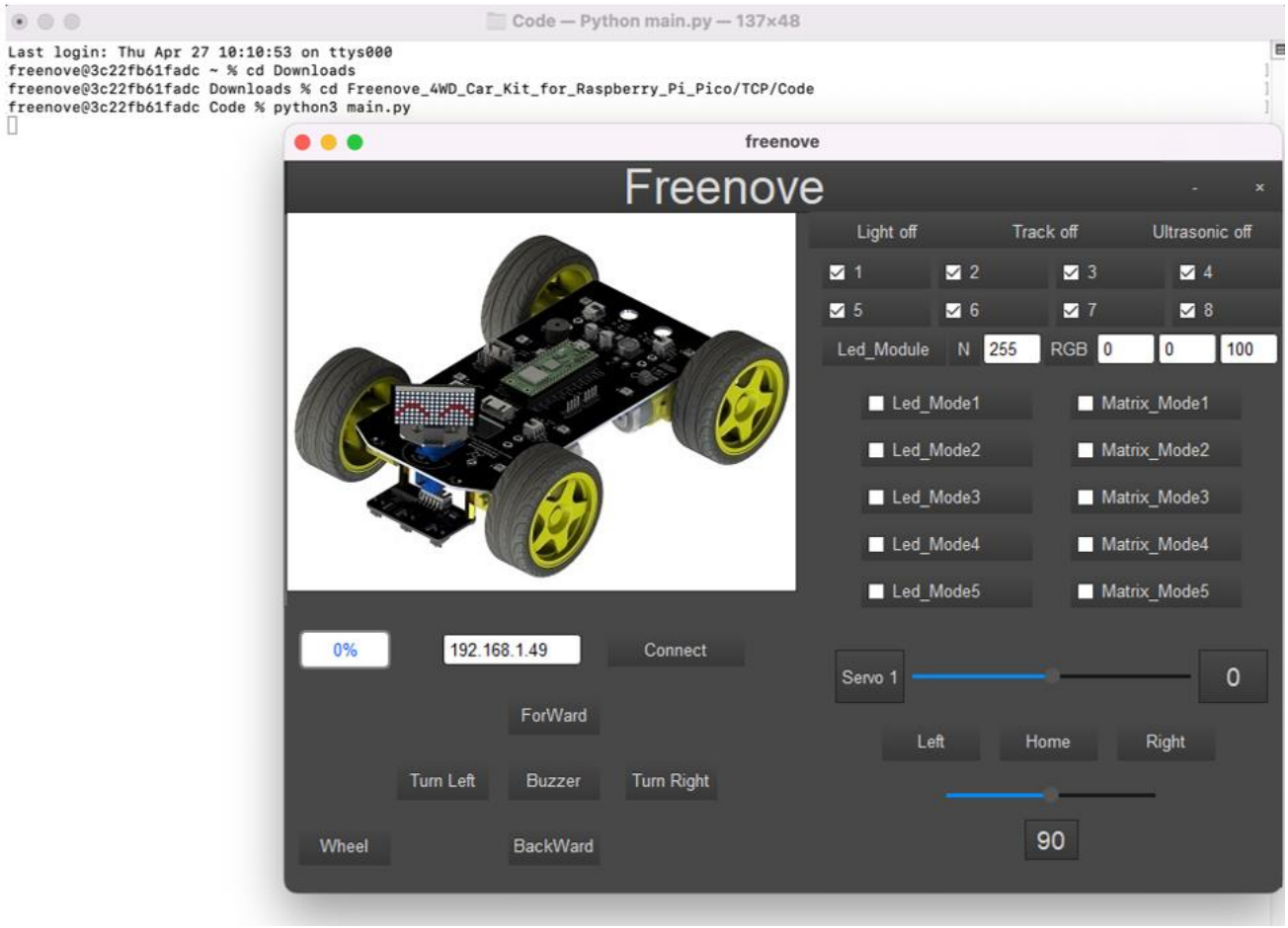
Run main.py

Following the previous steps, after the installation completes, you are now in the directory where setup_mac.py is located. (Freenove_4WD_Car_Kit_for_Raspberry_Pi_Pico/TCP/Code)



Enter the following command to run main.py.

```
python3 main.py
```



The way to control macOS System client is the same as Windows (Control).

What's next?

Thank you again for choosing Freenove products.

THANK YOU for participating in this learning experience!

We have reached the end of this Tutorial. If you find errors, omissions, or you have suggestions and/or questions about the Tutorial or component contents of this Kit, please feel free to contact us:
support@freenove.com

We will make every effort to make changes and correct errors as soon as feasibly possible and publish a revised version.

If you want to learn more about Arduino, ESP32, Raspberry Pi Pico W, Raspberry Pi, Smart Cars, Robotics and other interesting products in science and technology, please continue to visit our website. We will continue to launch fun, cost effective, innovative and exciting products.
http://www.freenove.com/

Thank you again for choosing Freenove products.