Welcome

Thank you for choosing Freenove products!
If you have not downloaded the zip file yet, please download it and unzip it via link below:
https://github.com/Freenove/Freenove_Bipedal_Robot_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

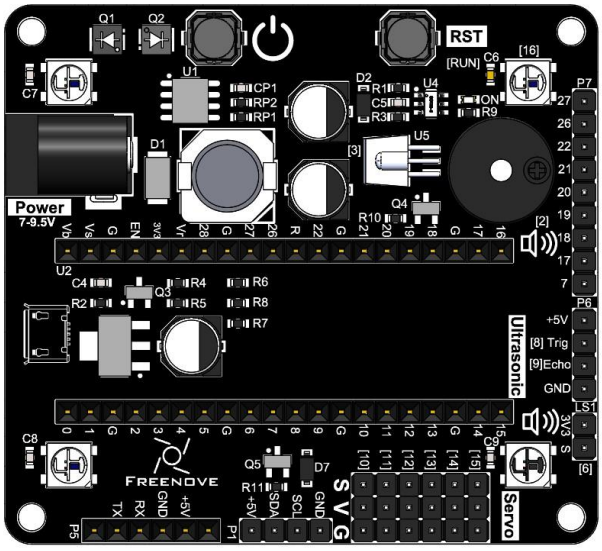
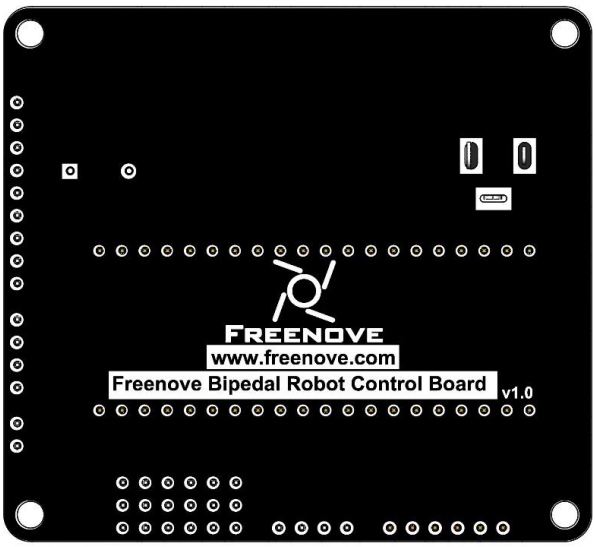
FREENOVE™

Contents

List

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

Raspberry Pi Pico (W) Robot Shield

Top	Bottom
	

Machinery Parts

M3*8
Screw

x10
Freenove

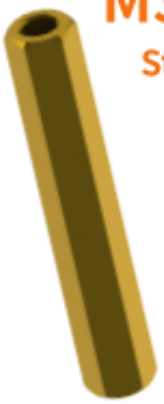
M3*18
Screw

x2
Freenove

M3
Nut

x2
Freenove

M3*35
Standoff



x4
Freenove

M2*12
Screw

x14
Freenove

M2
Nut

x17
Freenove

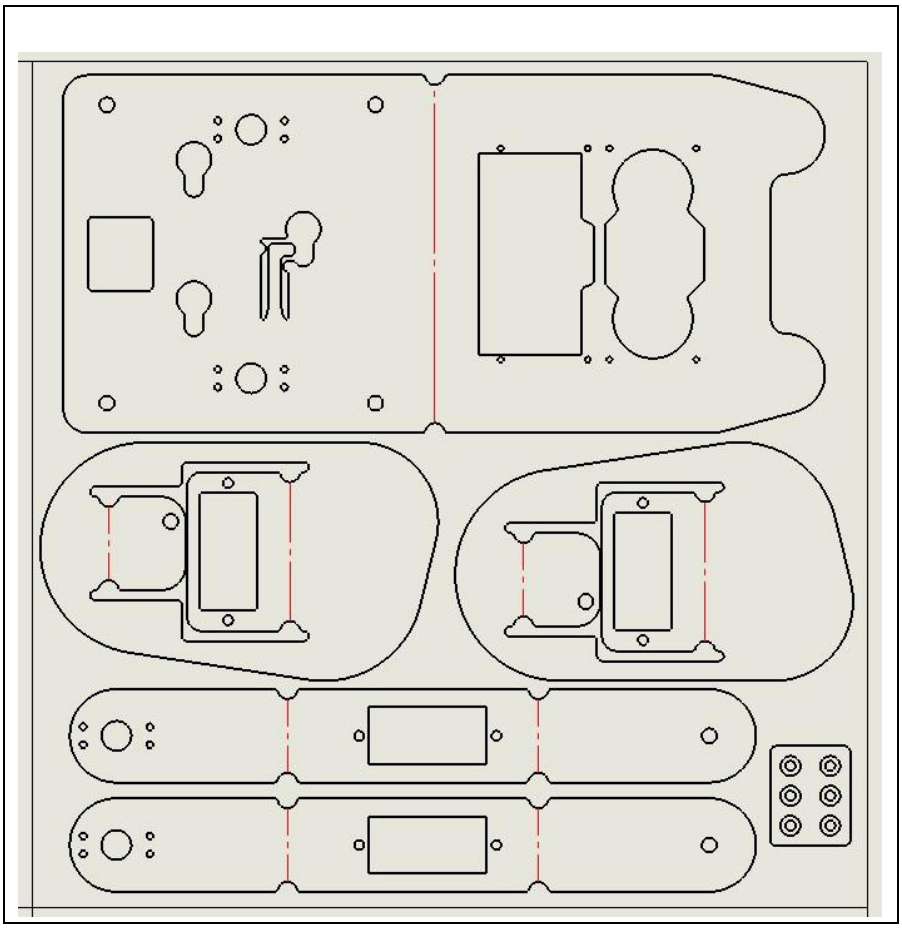
M1.4*5
self-tapping
Screw

x10
Freenove

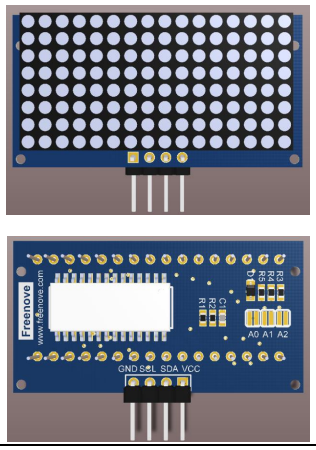
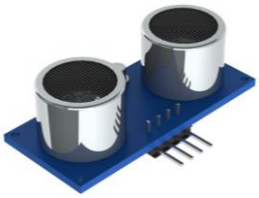
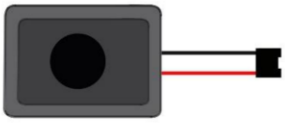
M2*6*0.5
Nut

x4
Freenove

Acrylic Parts



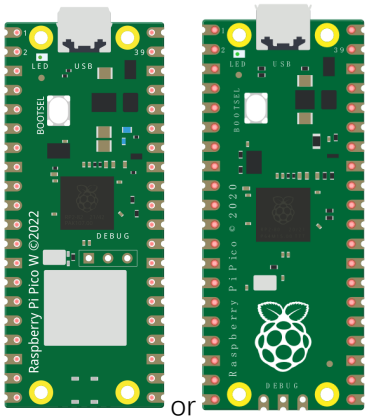
Electronic Parts

Dot Matrix Module x1 	Ultrasonic Module x1 
	Speaker x1 
Servo package x4	

Bluetooth module x1

Infrared emitter x1

Raspberry Pi Pico W x1 or Raspberry Pi Pico x1



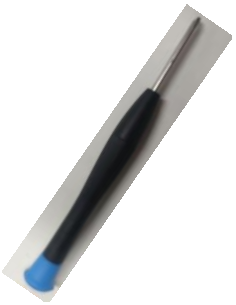
or

If you bought the kit without control board, please prepare one of the above board yourself.

Wires

Jumper Wire F/F(4 pin) x2

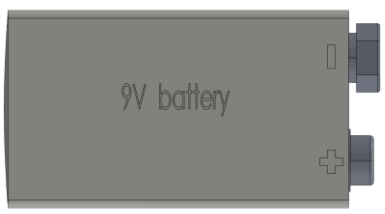
Tools

Cross screwdriver (3mm) x1 	Cross screwdriver (2mm)x1 	Cable Tidy x40cm 
M3 spanner 		

Required but NOT Contained Parts

This robot is powered with **9V alkaline batteries** or **9V NIMH rechargeable batteries** as power supply. Please note that 9V carbon batteries and 9V rechargeable lithium batteries **cannot** provide enough power for the robot. Therefore, it is important to pay attention to the battery model when purchasing batteries.

It is easy to find appropriate batteries on both eBay and Amazon. You can simply search 9V battery 6lr61 or 9V rechargeable NIMH battery on the platforms.



Preface

Welcome to use Freenove Bipedal Robot Kit for Raspberry Pi Pico (W). Following this tutorial, you can make a very cool robot 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 robot. 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.

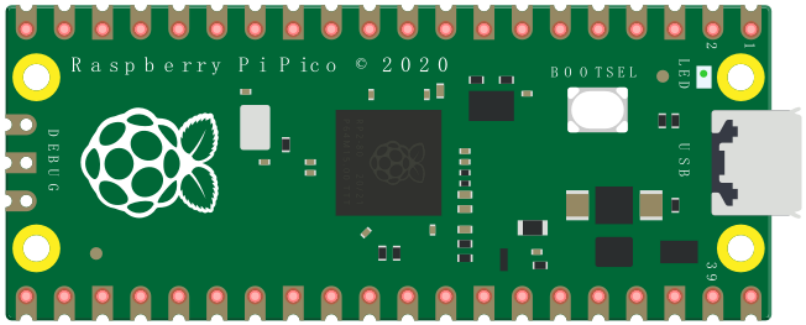
support@freenove.com

This robot does not require a high threshold for users. Even if you know little professional knowledge, you can make your own smart robot 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:

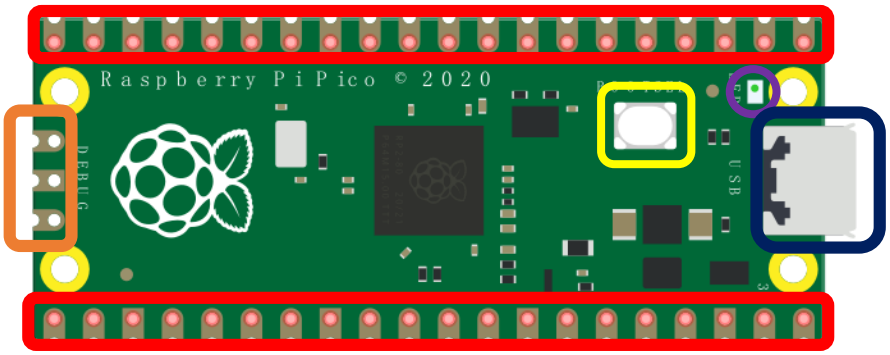
www.freenove.com or contact us to buy our kit designed for beginners: **Freenove Ultimate Kit for Raspberry Pi Pico.**

Raspberry Pi Pico

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

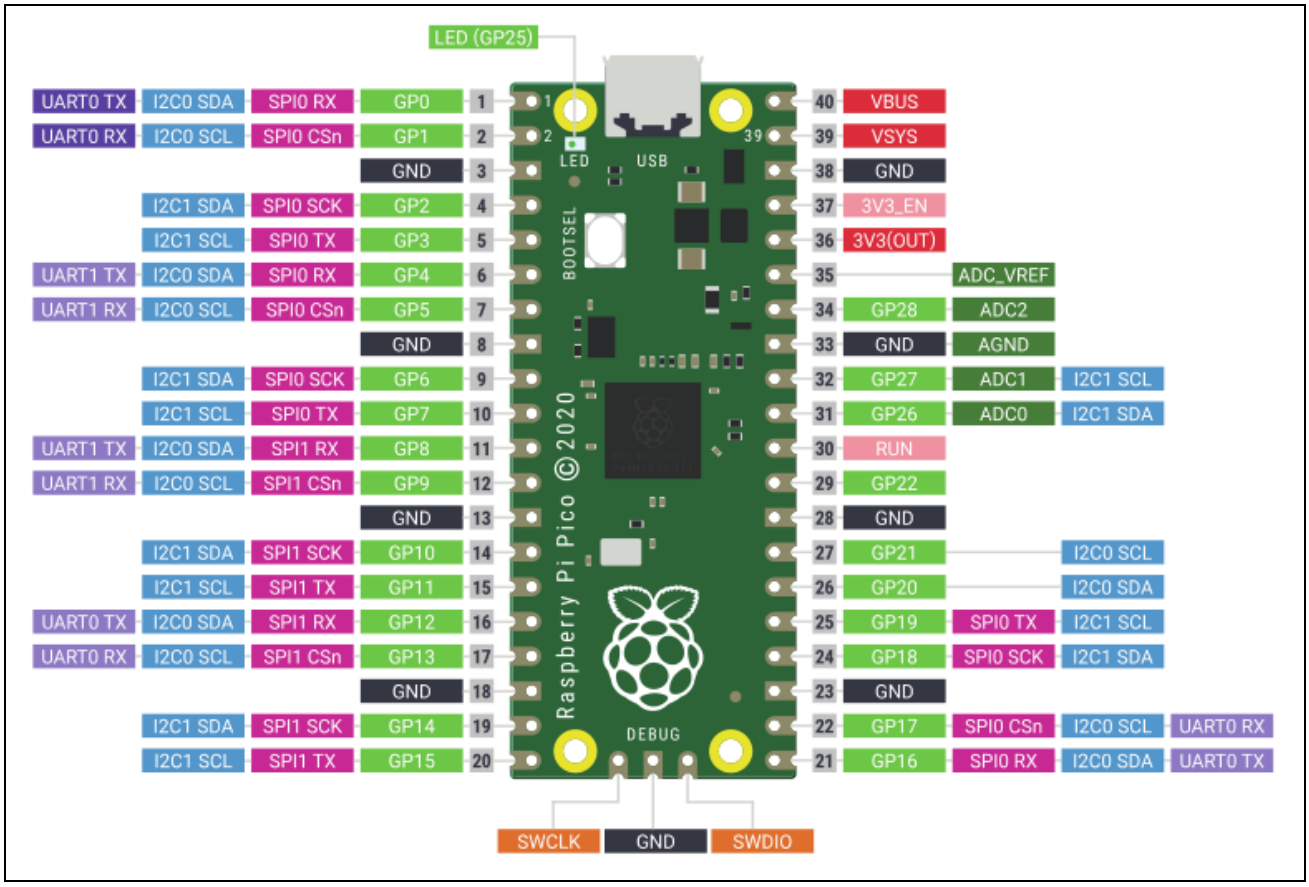


The hardware interfaces are distributed as follows:



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

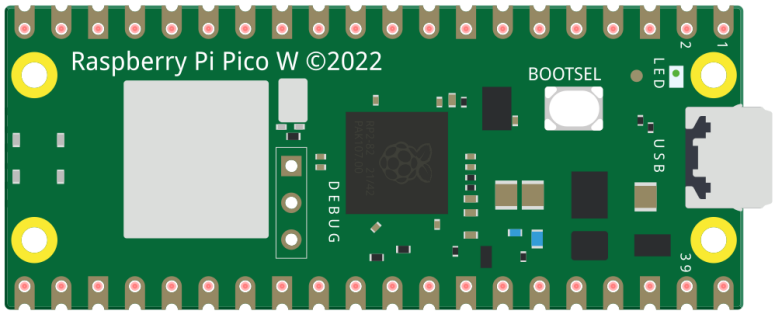
Function	Default
I2C Frequency	400000
I2C_SDA	Pin 4
I2C_SCL	Pin 5

SPI

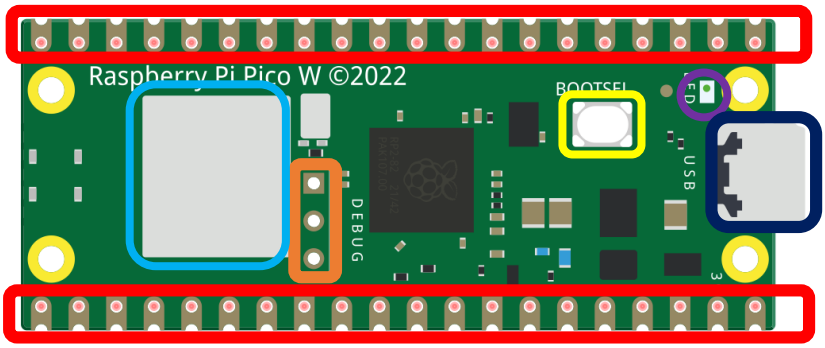
Function	Default
SPI_BAUDRATE	1000000
SPI_POLARITY	0
SPI_PHASE	0
SPI_BITS	8
SPI_FIRSTBIT	MSB
SPI_SCK	Pin 18
SPI_MOSI	Pin 19
SPI_MISO	Pin 16
SPI_SS	Pin 17

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.

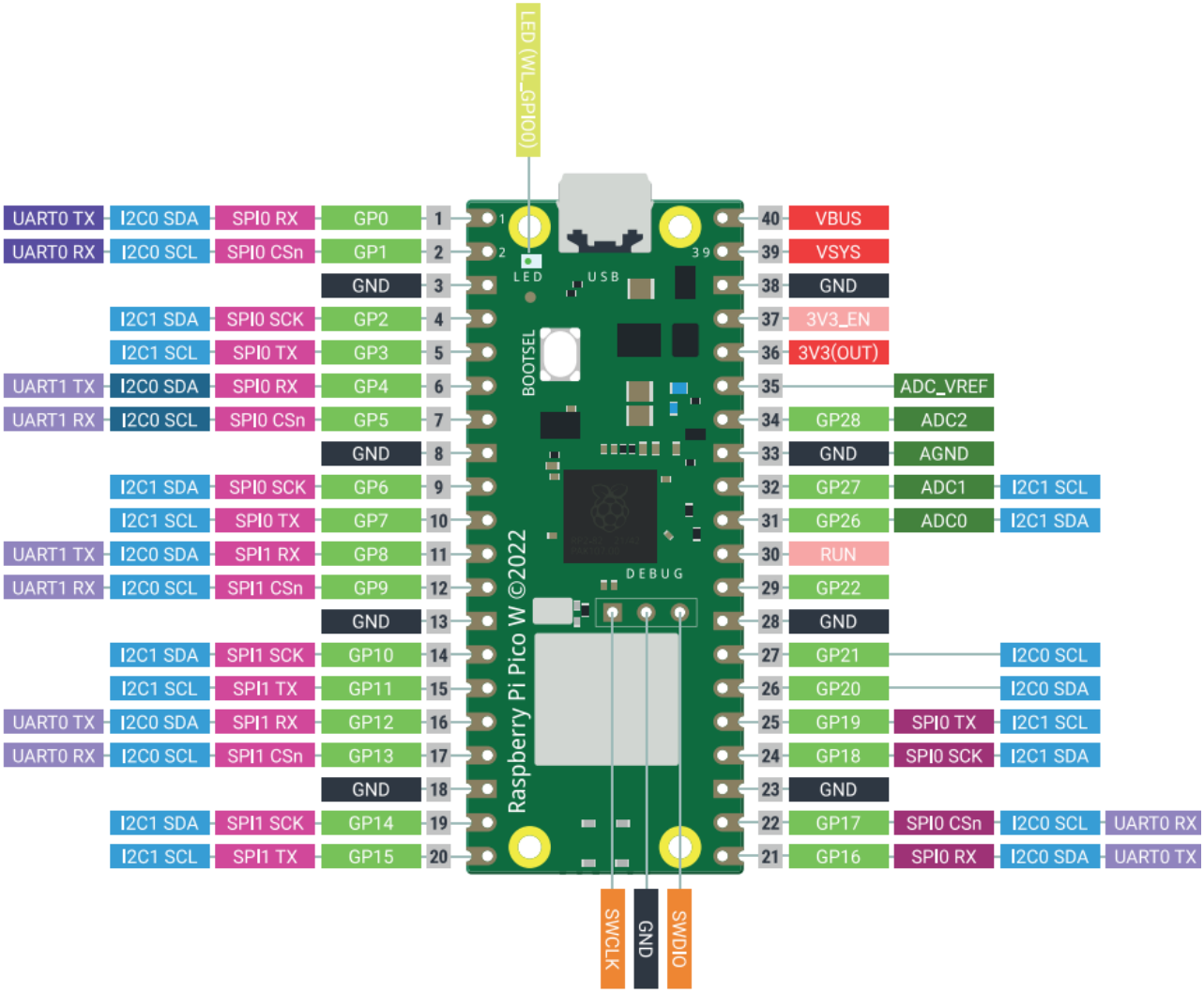


The hardware interfaces are distributed as follows:



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

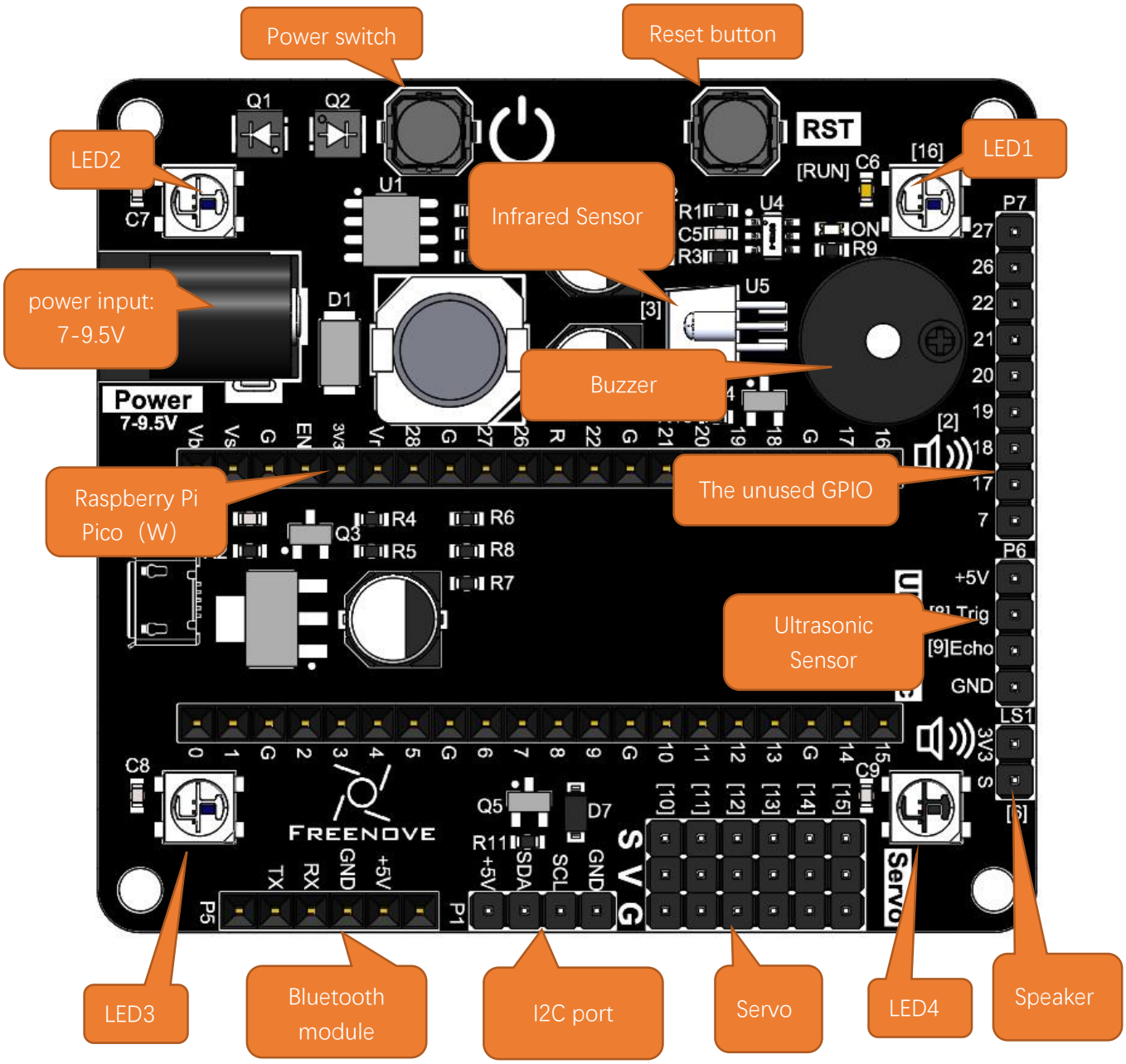
Pins of the Robot

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
GPIO10	Servo	Servo1
GPIO11		Servo2
GPIO12		Servo3
GPIO13		Servo4
GPIO14		Servo5
GPIO15		Servo6
GPIO8	Ultrasonic module	Trig
GPIO9		Echo
GPIO4	I2C port	SDA
GPIO5		SCL
GPIO16	WS2812	WS2812
GPIO28	Battery detection	A2
GPIO6	Speaker interface	Speaker
GPIO3	Infrared receiver port	IR
GPIO2	Buzzer port	Buzzer
GPIO1	Bluetooth module	RX
GPIO0		TX
GPIO7	Unused GPIO	GPIO7
GPIO17		GPIO17
GPIO18		GPIO18
GPIO19		GPIO19
GPIO20		GPIO20
GPIO21		GPIO21
GPIO22		GPIO22
GPIO26		GPIO26
GPIO27		GPIO27

Introduction to the Bipedal Robot

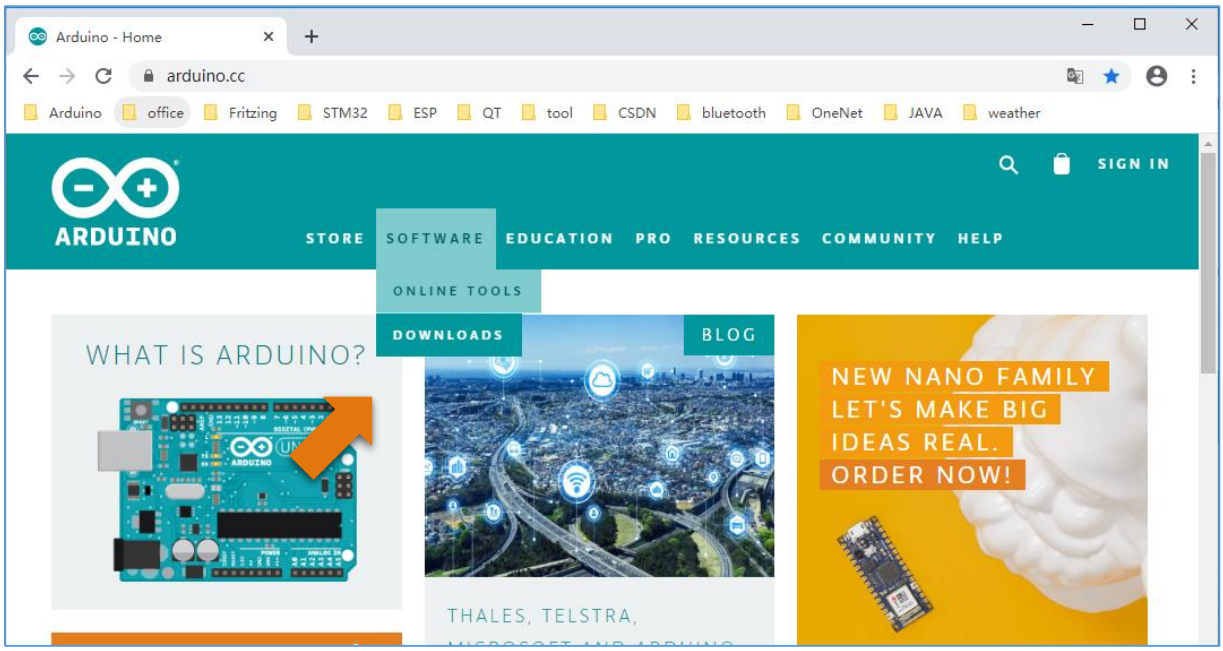
The function diagram of the Raspberry Pi Pico (W) bipedal robot is as follows:



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.

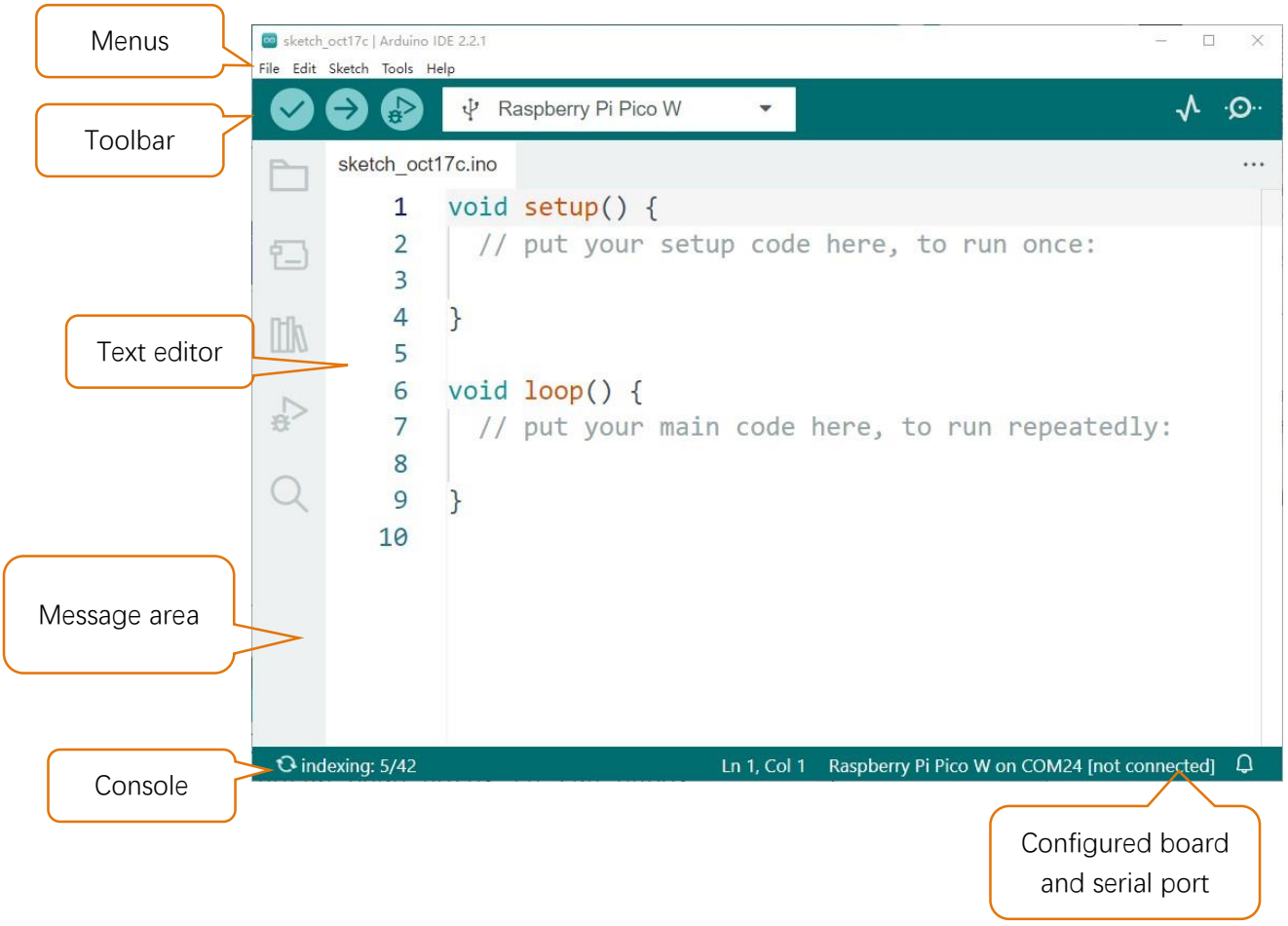


Select and download corresponding installer based on your operating system. If you are a Windows user, please select the "Windows Installer" to download and install the driver correctly.

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:



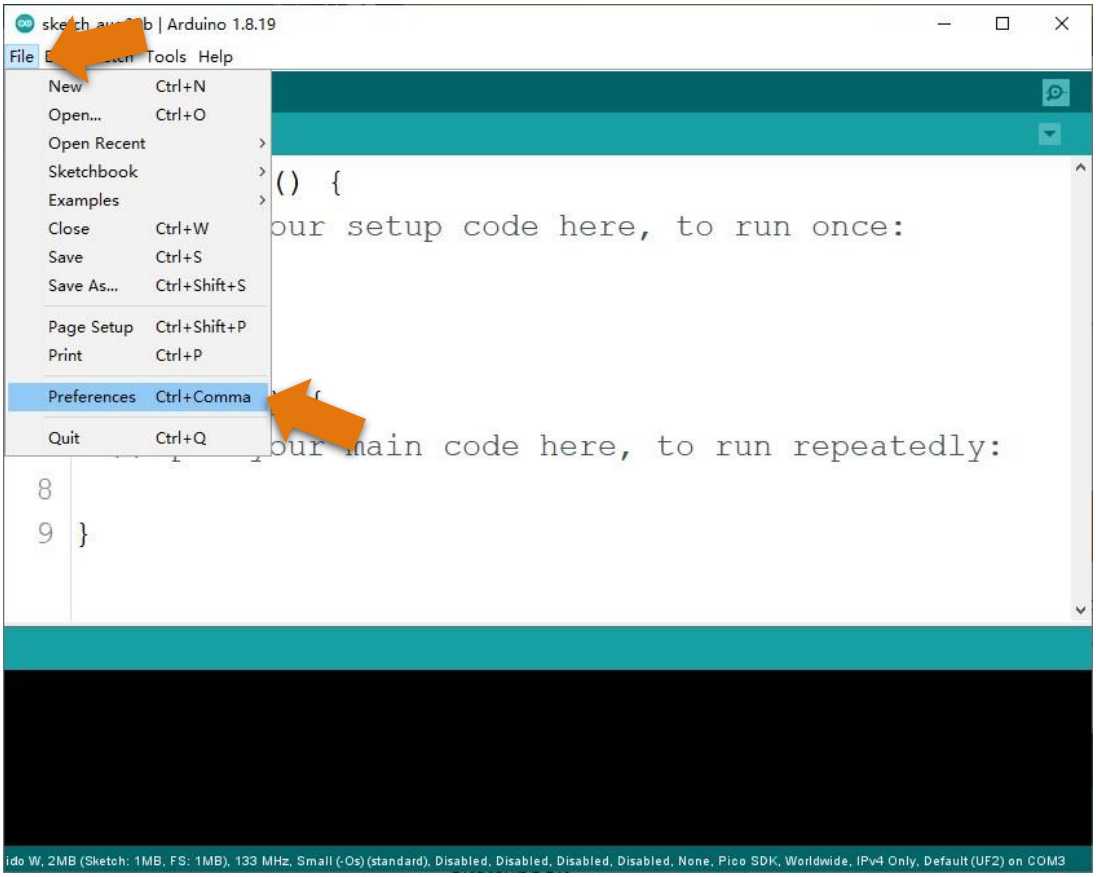
Programs written with Arduino Software (IDE) are called **sketches**. These sketches are written in the text editor and saved with the file extension**.ino**. The editor has features for cutting/pasting and searching/replacing text. The message area gives feedback while saving and exporting and also 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.

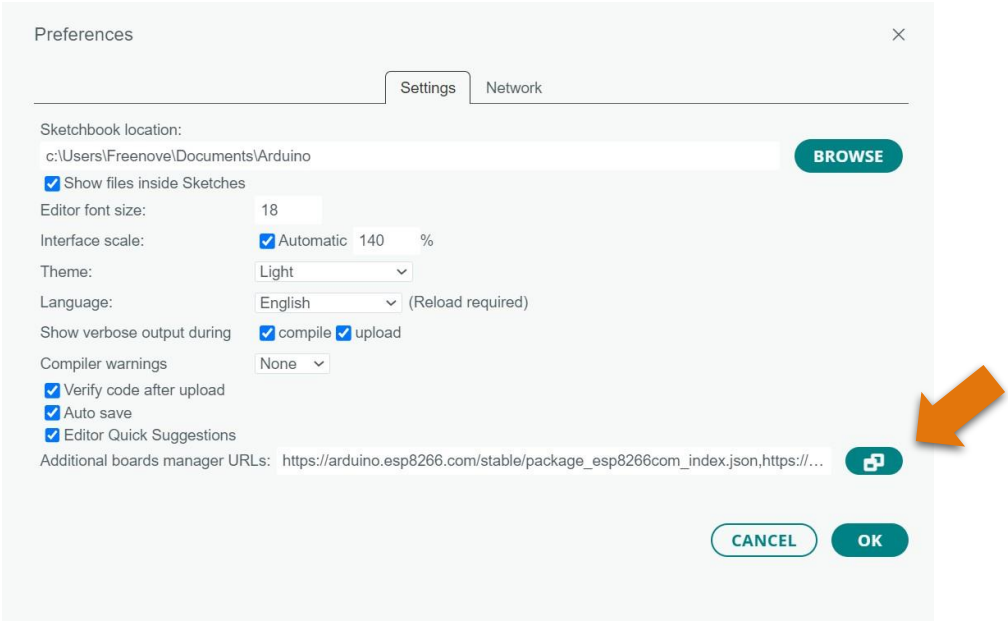
Additional commands are found within the five menus: File, Edit, Sketch, Tools, Help. The menus are context sensitive, which means only those items relevant to the work currently being carried out are available.

Environment Configuration

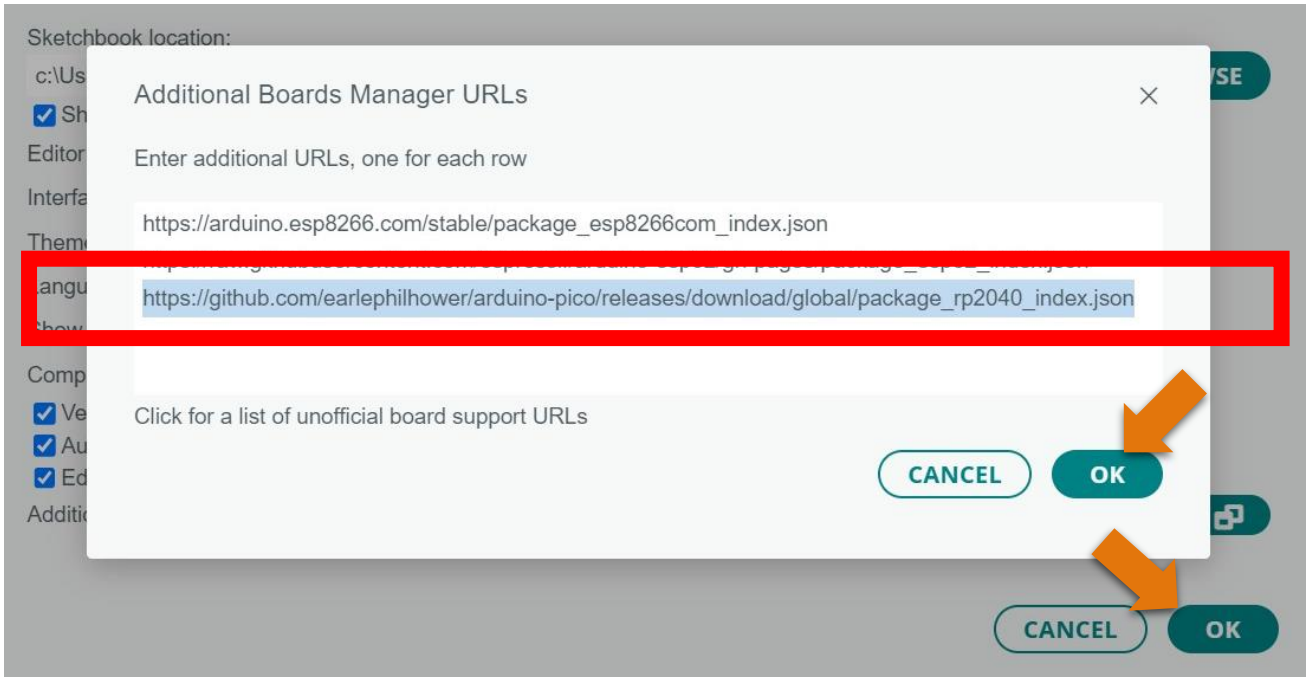
First, open the software platform arduino, and then click File in Menus and select Preferences.



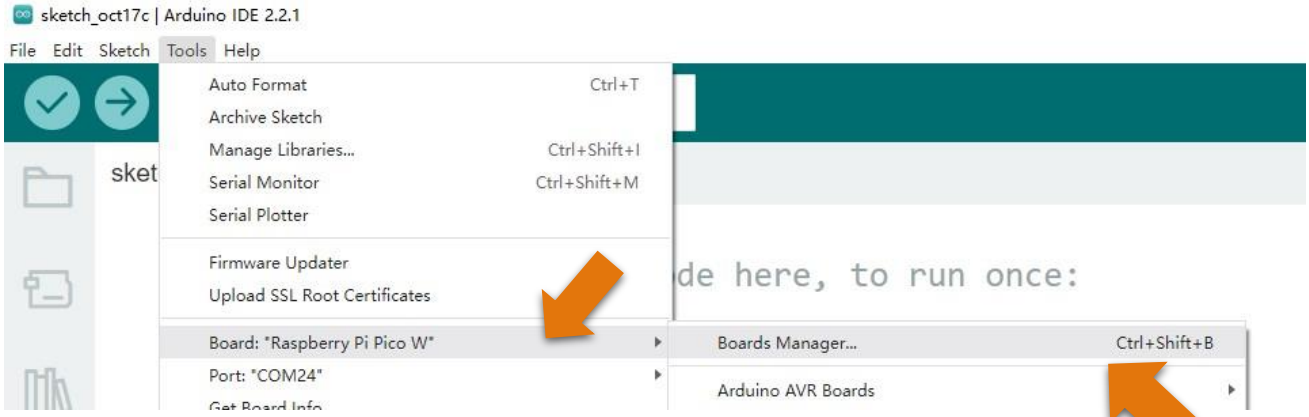
Second, click on the symbol behind "Additional Boards Manager URLs"



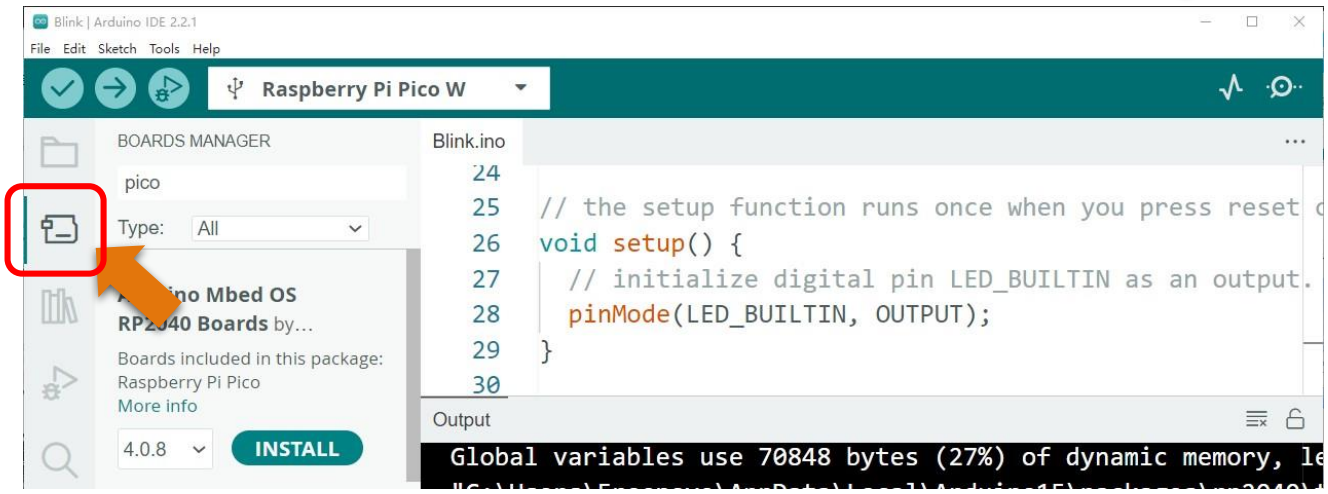
Third, fill in https://github.com/earlephilhower/arduino-pico/releases/download/global/package_rp2040_index.json in the new window, click OK, and click OK on the Preferences window again.



Fourth, click Tools on the Menu Bar > Board > "Boards Manager".

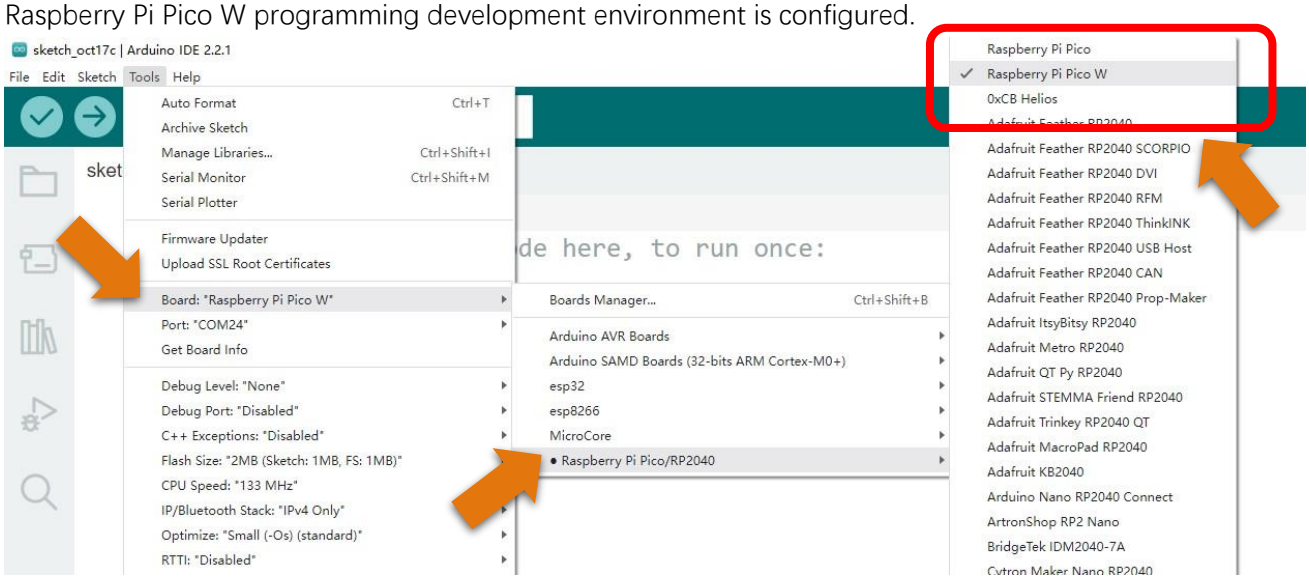


Or click the shortcut icon on the left.



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.

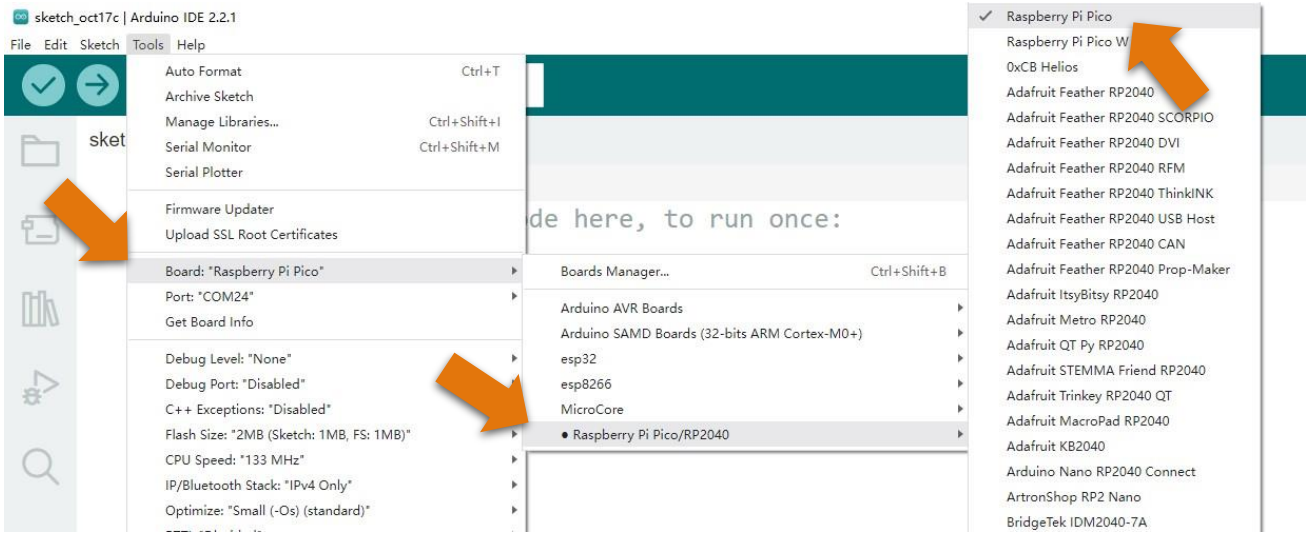


Additional Remarks

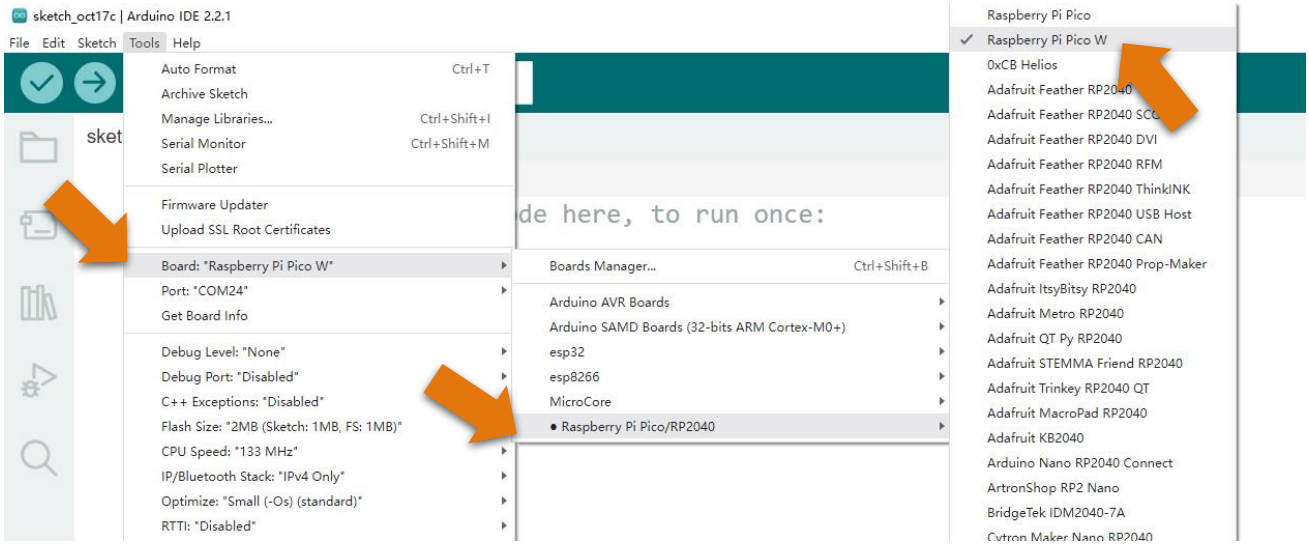
To finish this tutorial, you can use a Raspberry Pi Pico W or a Raspberry Pi Pico. The two boards have the same form factor, and their only difference is that Pico W features with an onboard single-band 4.802GHz wireless interface using Infineon CYW2. That is to say, the hardware Raspberry Pi Pico W is almost the same as the normal pico except for the wireless interface.
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.



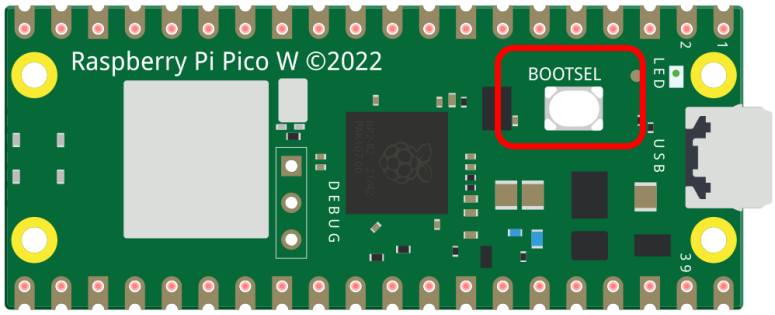
For Raspberry Pi Pico W:
Click Tools on Menu bar, click Board, select “Raspberry Pi Pico/RP2040”, and select Raspberry Pi Pico W.



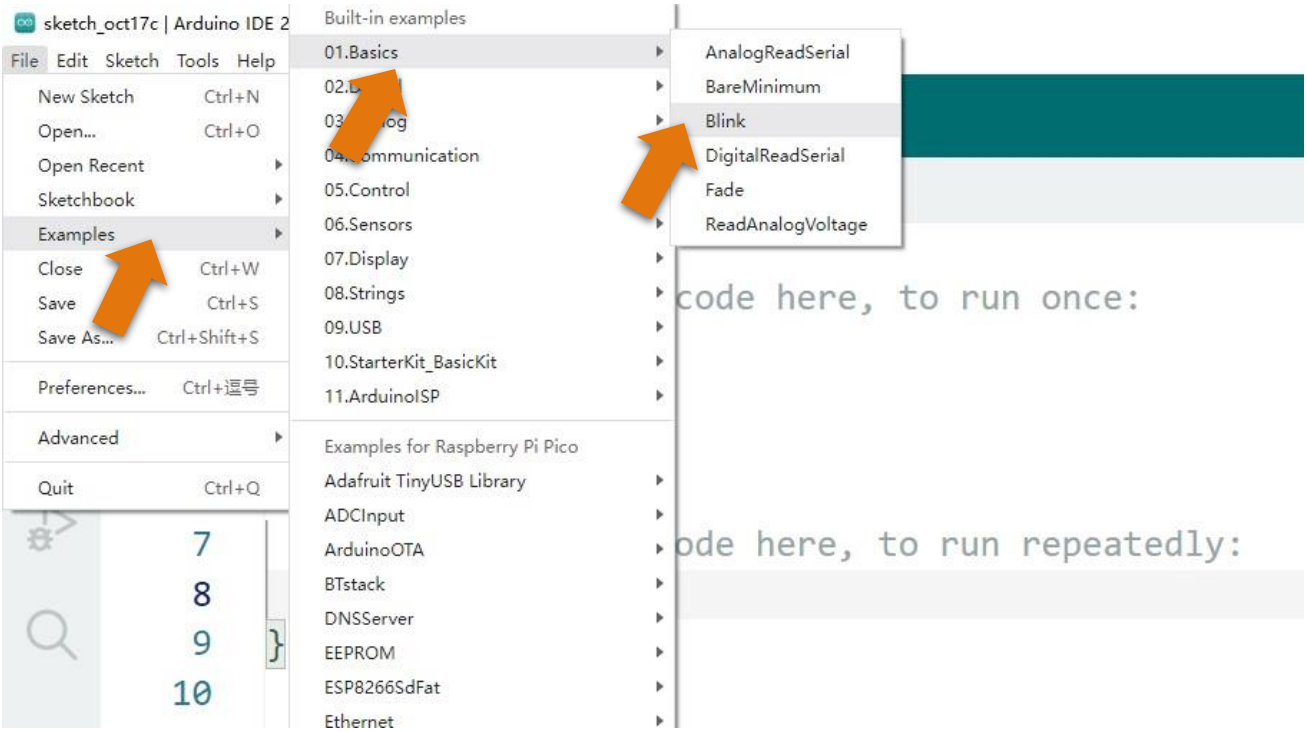
Uploading Adruino-compatible Firmware for Raspberry Pi Pico (W)

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.

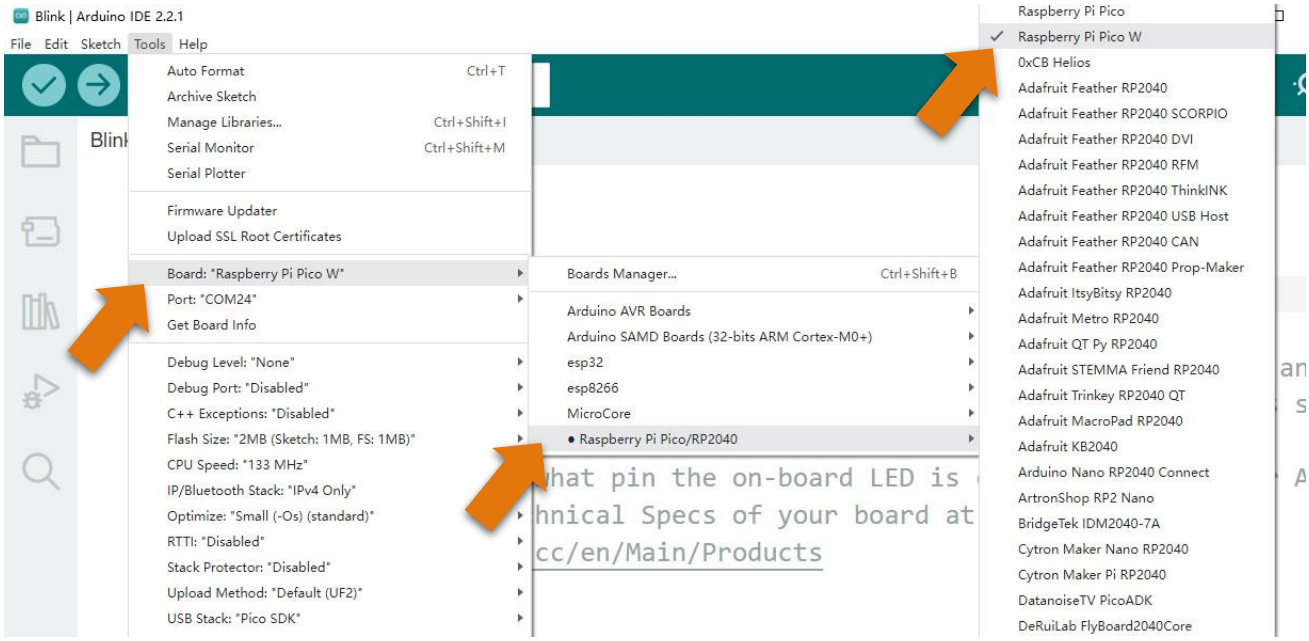
1. Disconnect Pico (W) from your computer. Press and hold the white button (BOOTSEL) on Pico (W) while connecting it to your computer. (Note: Be sure to hold the button before powering the Pico, otherwise the firmware will not download successfully.)



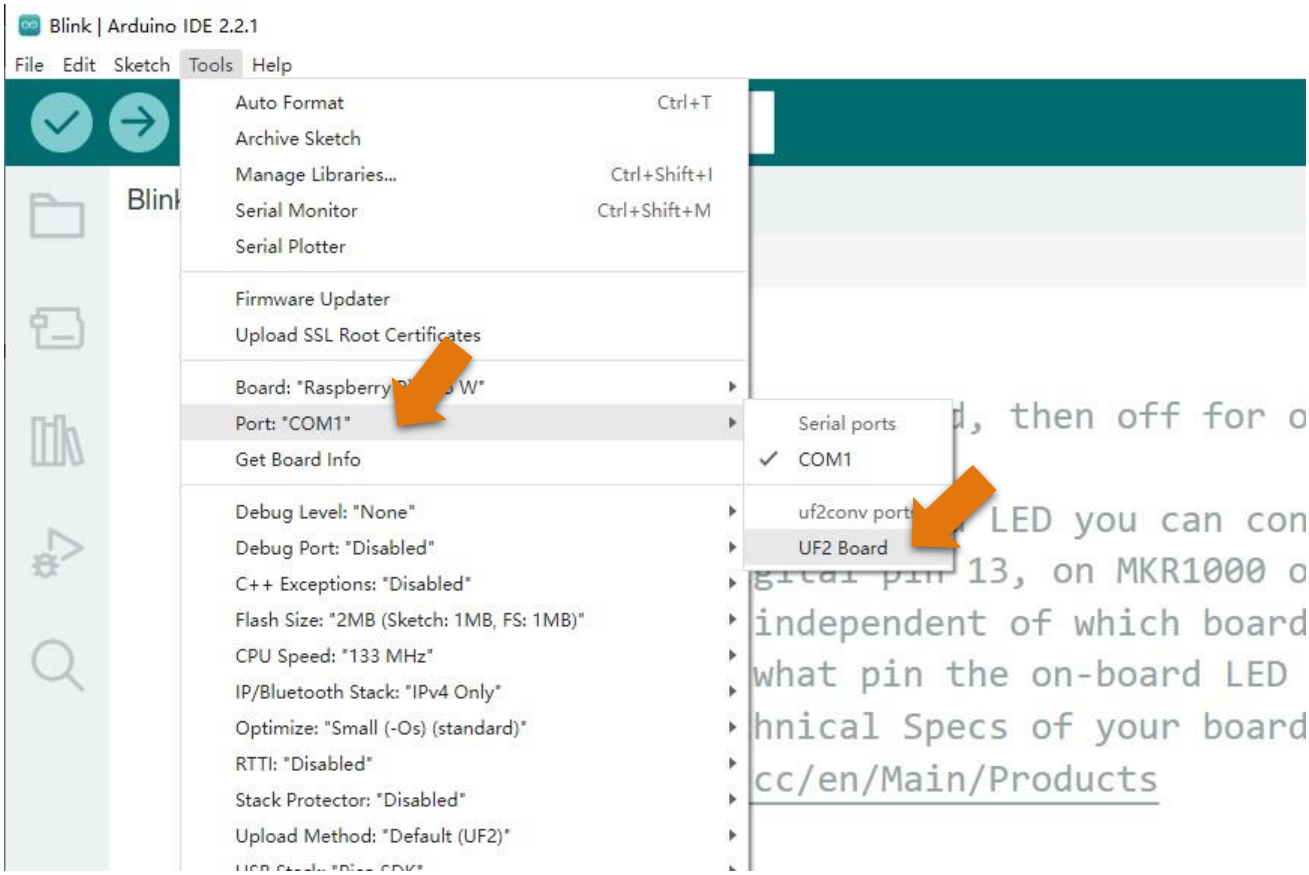
2. Open Arduino IDE. Click File>Examples>01.Basics>Blink.



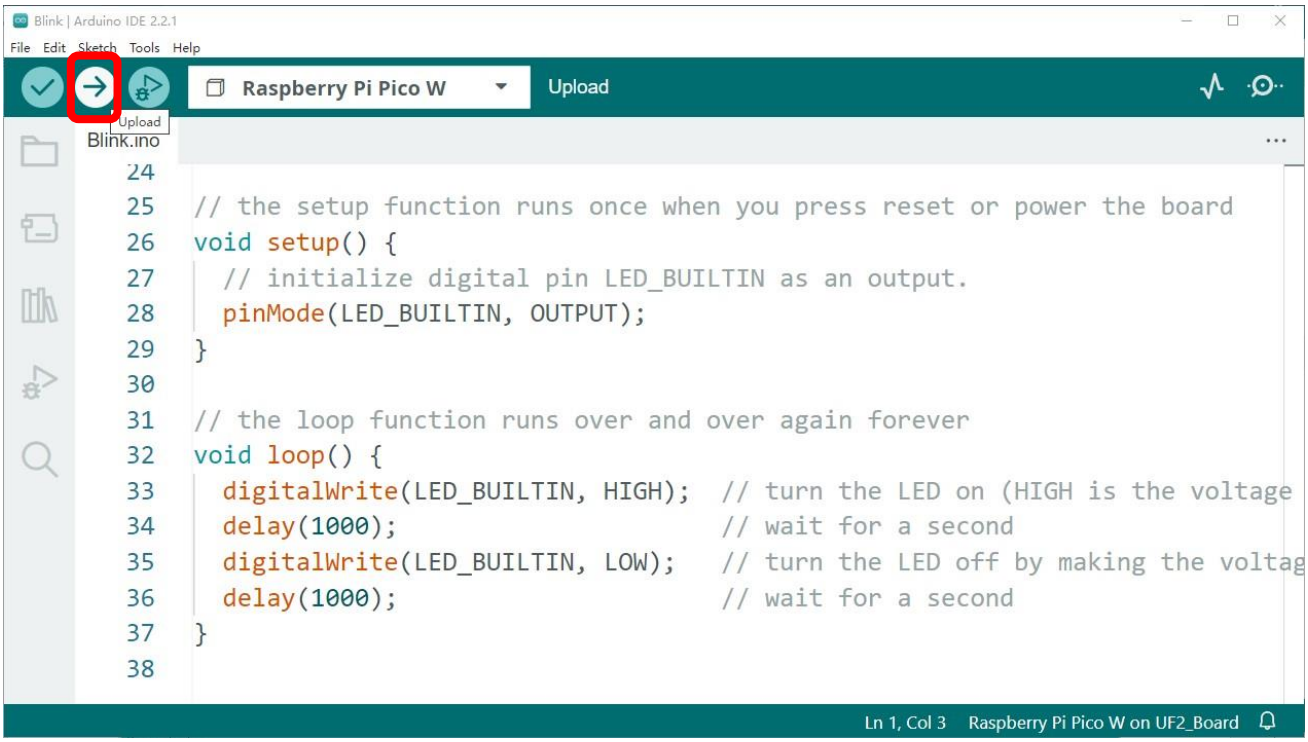
3. Click **Tools>Board>Raspberry Pi RP2040 Boards>Raspberry Pi Pico W**.



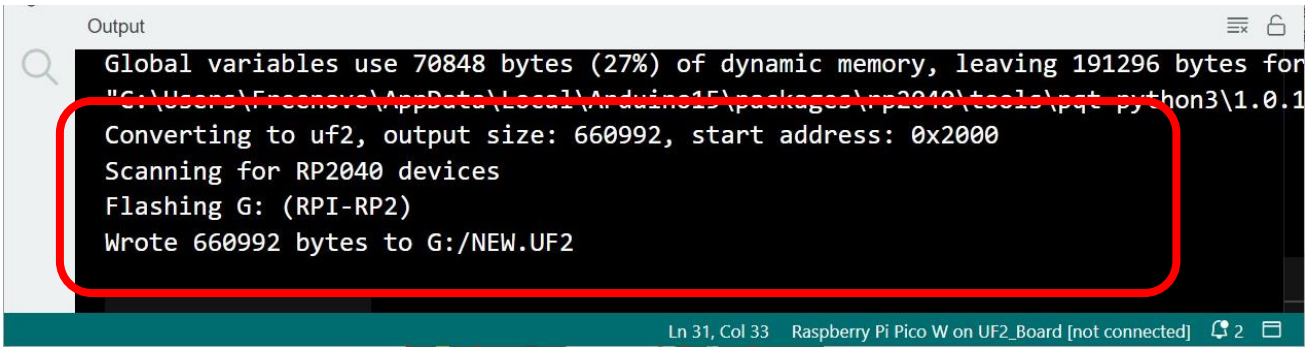
4. Click **Tools>Port>UF2 Board**.



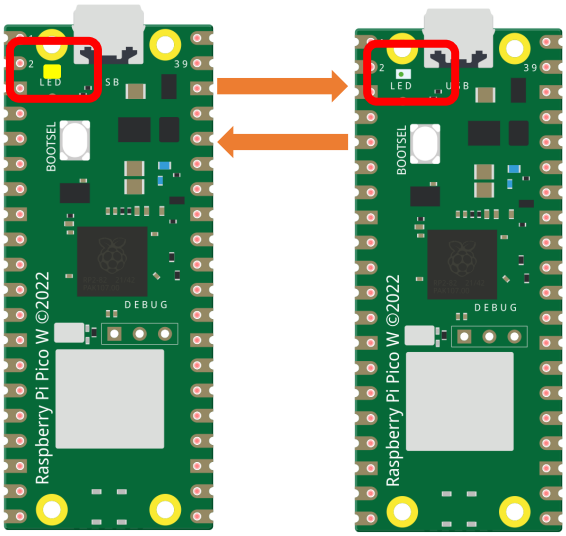
5. Upload sketch to Raspberry Pi Pico W.



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

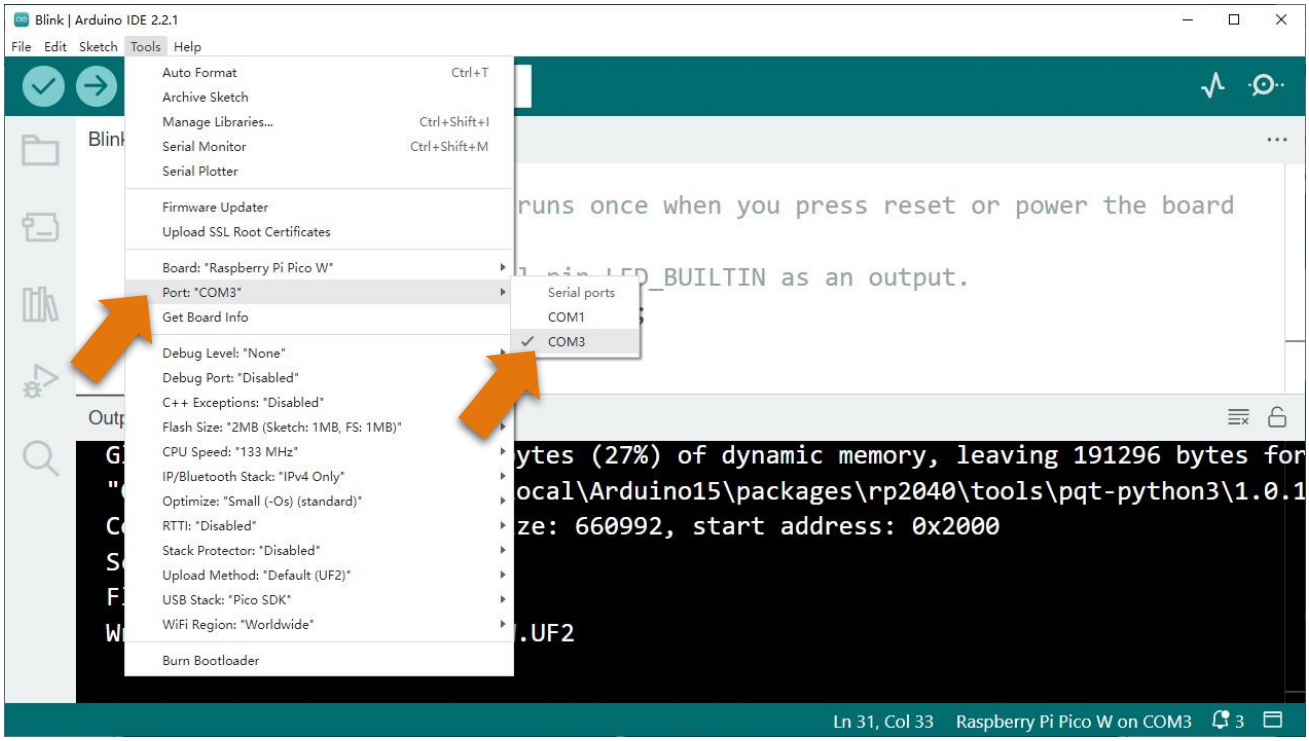


The indicator on Pico W starts to flash.



5, Click **Tools>Port>COMx (Raspberry Pi Pico W)**. X of COMx varies from different computers. Please select

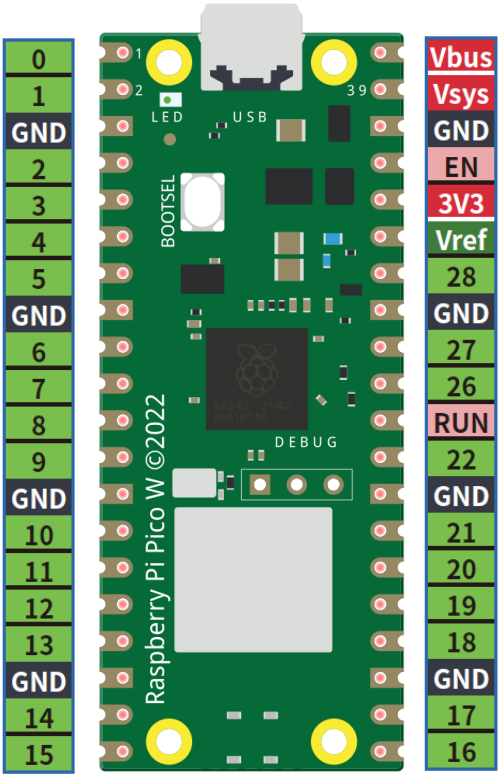
the correct one on your computer. In our case, it is COM26.



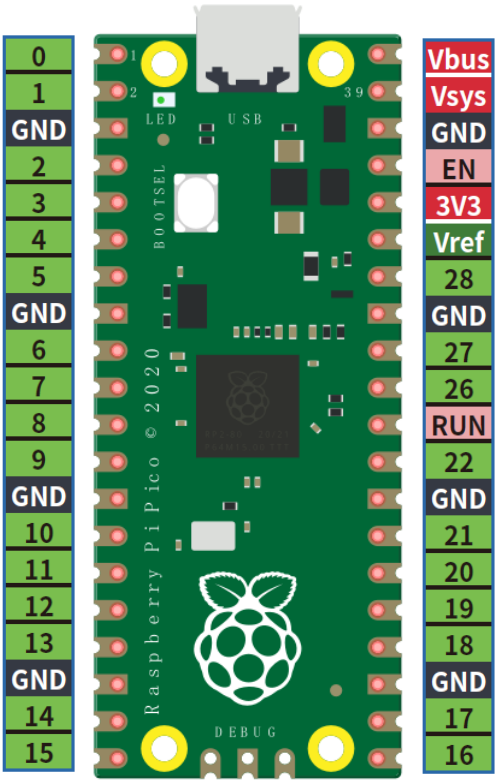
Note:

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

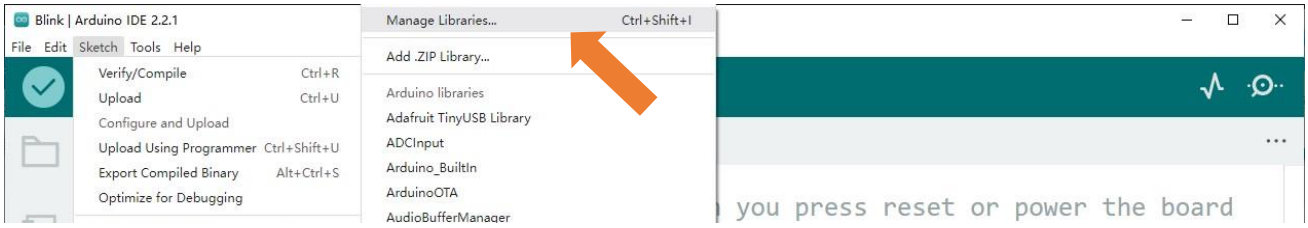
Here we take “00.0_Servo_90” in
“Freenove_Bipedal_Robot_Kit_for_Raspberry_Pi_Pico\Sketches” as an example.

How to Add Libraries

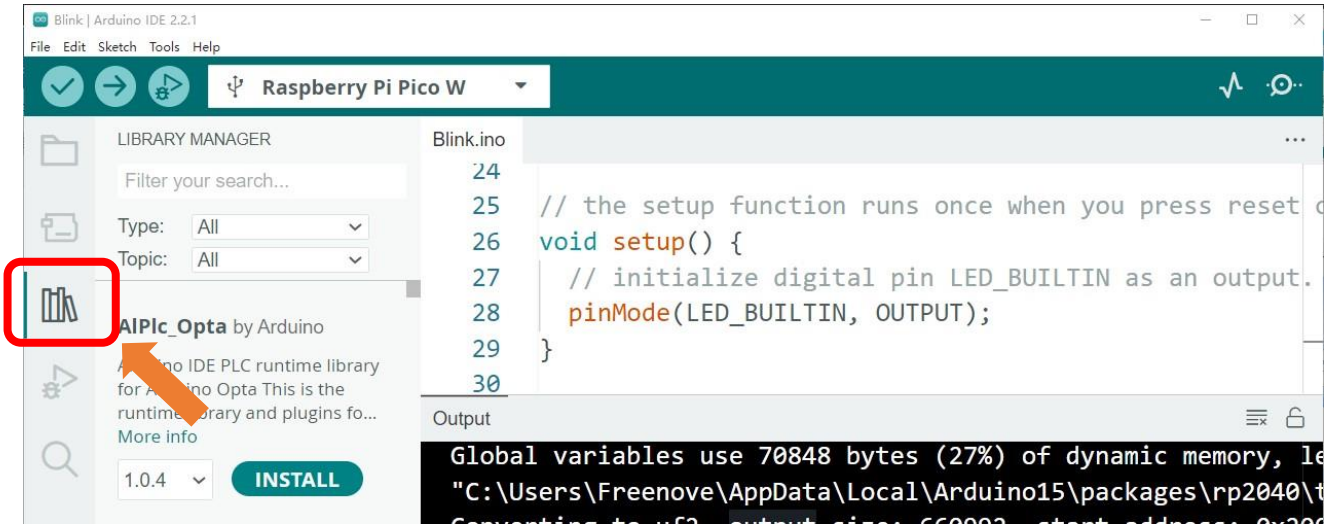
Method 2 is preferred.

Method 1

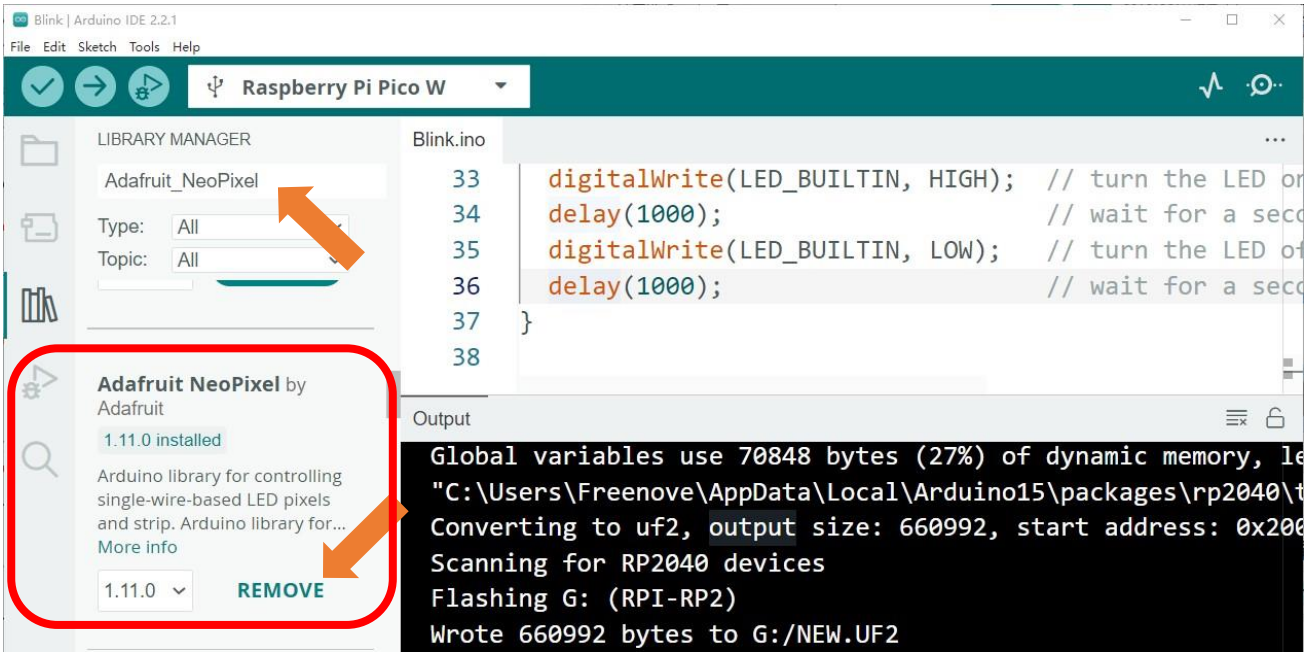
Open Arduino IDE, click Sketch on Menu bar ->Include Library -> Manage Libraries.



Or clikc the shortcut icon on the left.



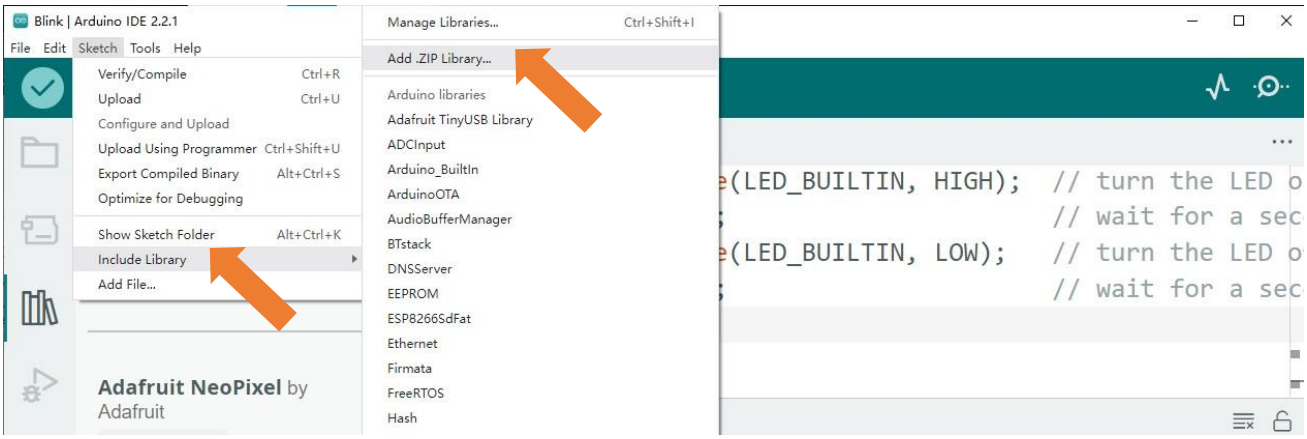
There is an input field on the right top of the pop-up window. Enter Adafruit_NeoPixel there and click to install the library boxed in the following figure.



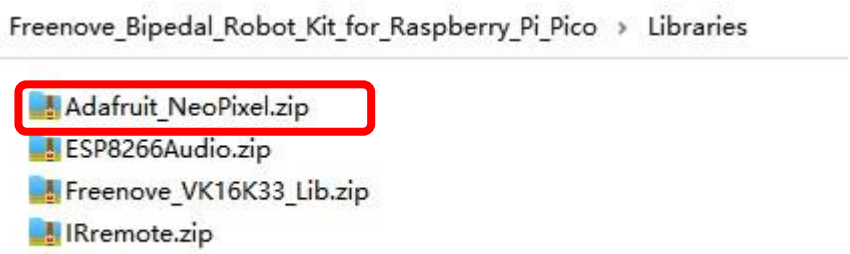
Wait for the installation to finish.

Method 2

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

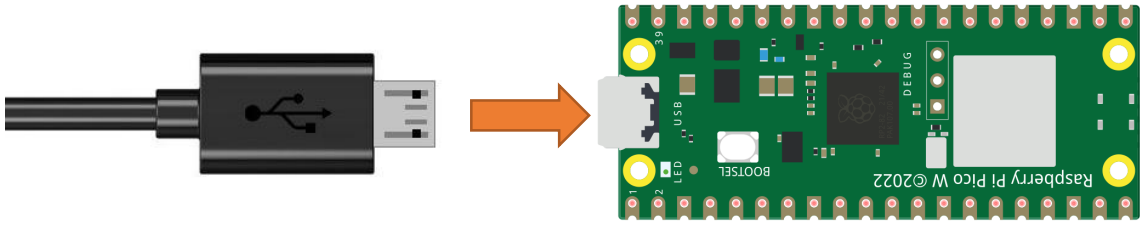


On the pop-up window, select Adafruit_NeoPixel.zip in Libraries folder under **"Freenove_Bipedal_Robot_Kit_for_Raspberry_Pi_Pico\Libraries"**, and then click Open.

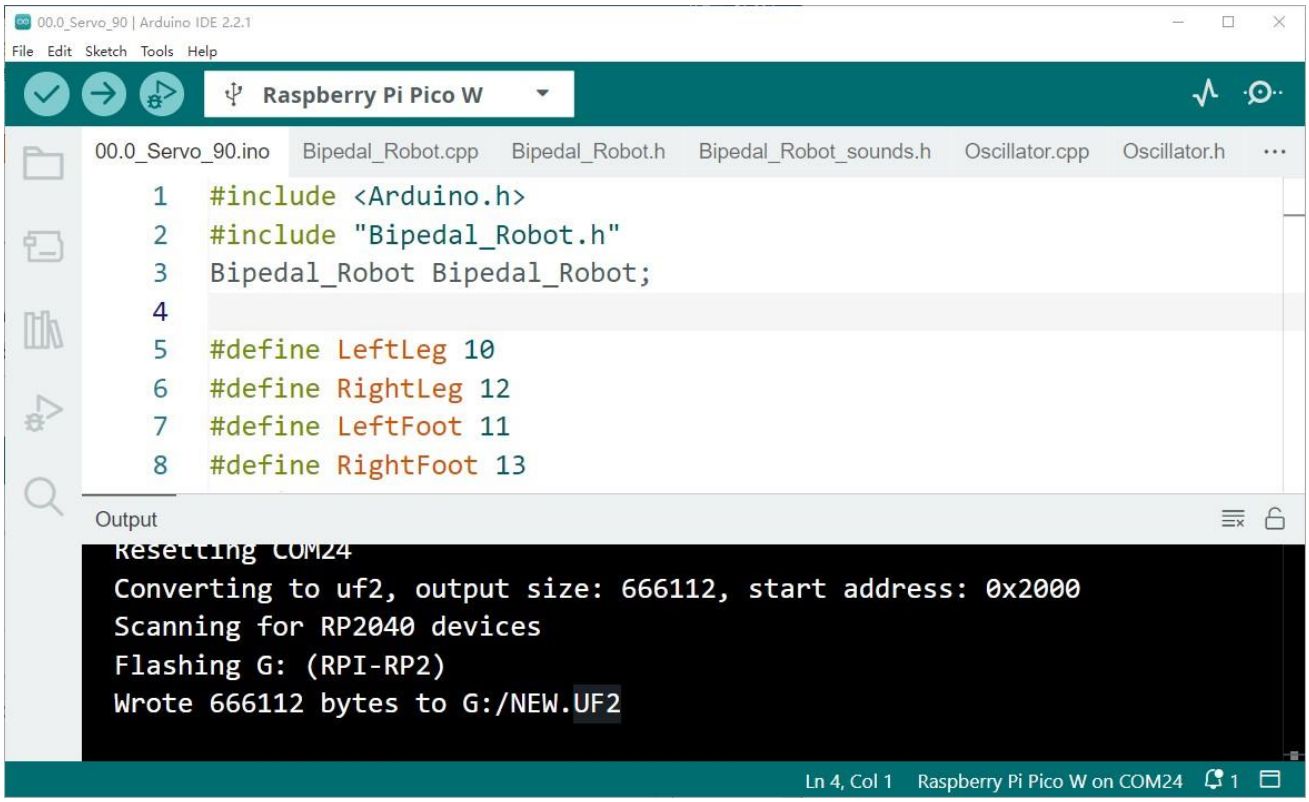


Upload the First Code

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

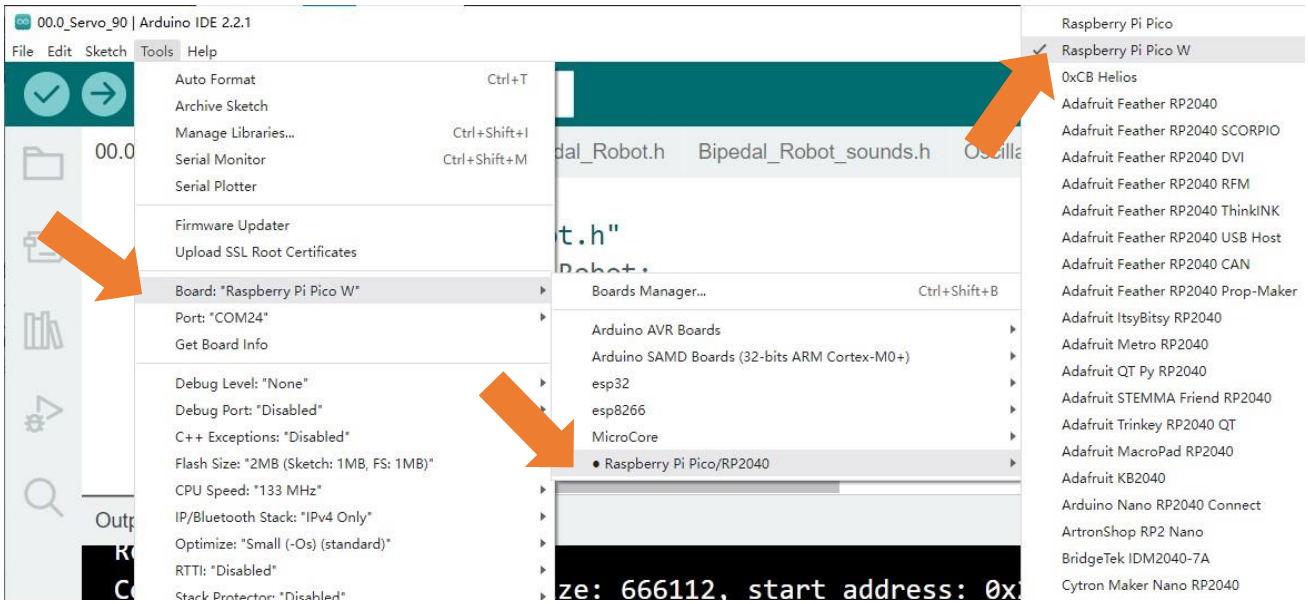


Step 2. Open "00.0_Servo_90" folder in **"Freenove_Bipedal_Robot_Kit_for_Raspberry_Pi_Pico\Sketches"**, and double-click "00.0_Servo_90.ino". The code is to rotate the servo motors to 90°.



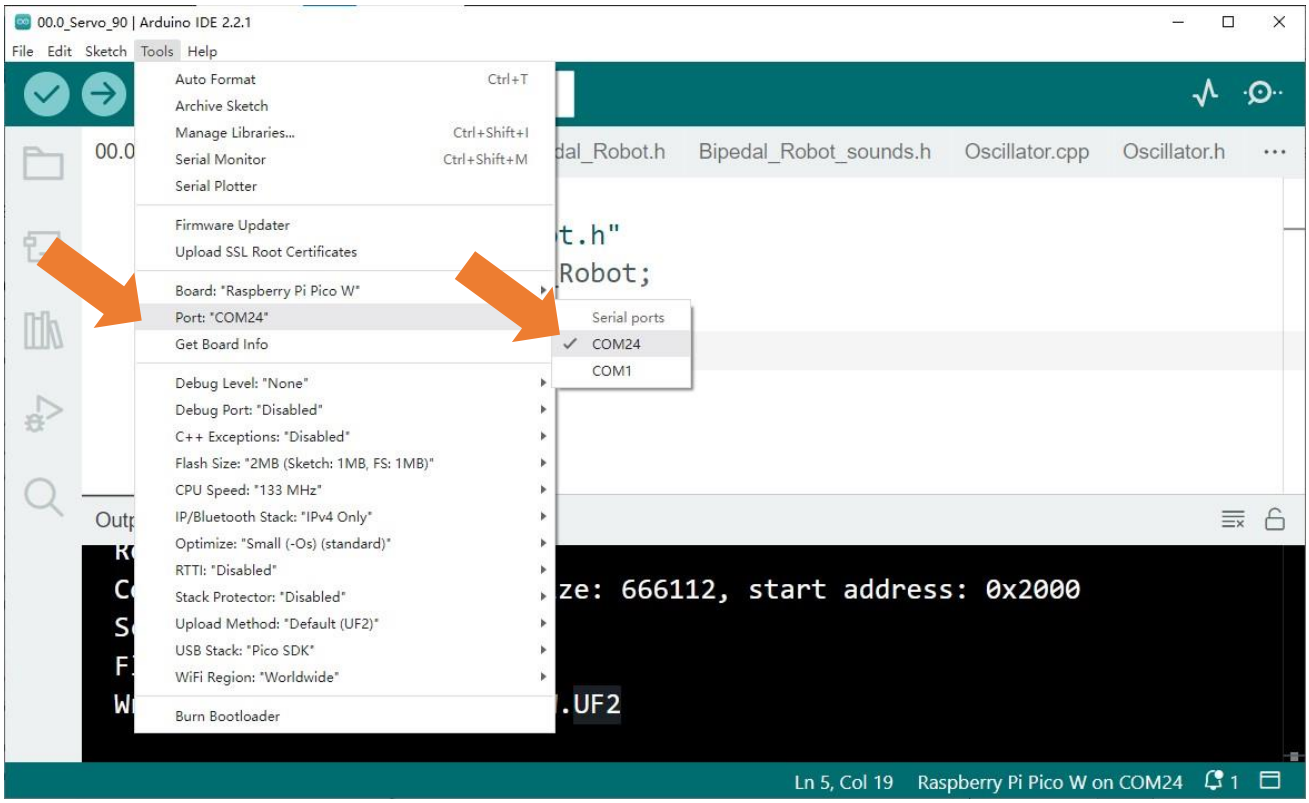
Step 3. Select development board.

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

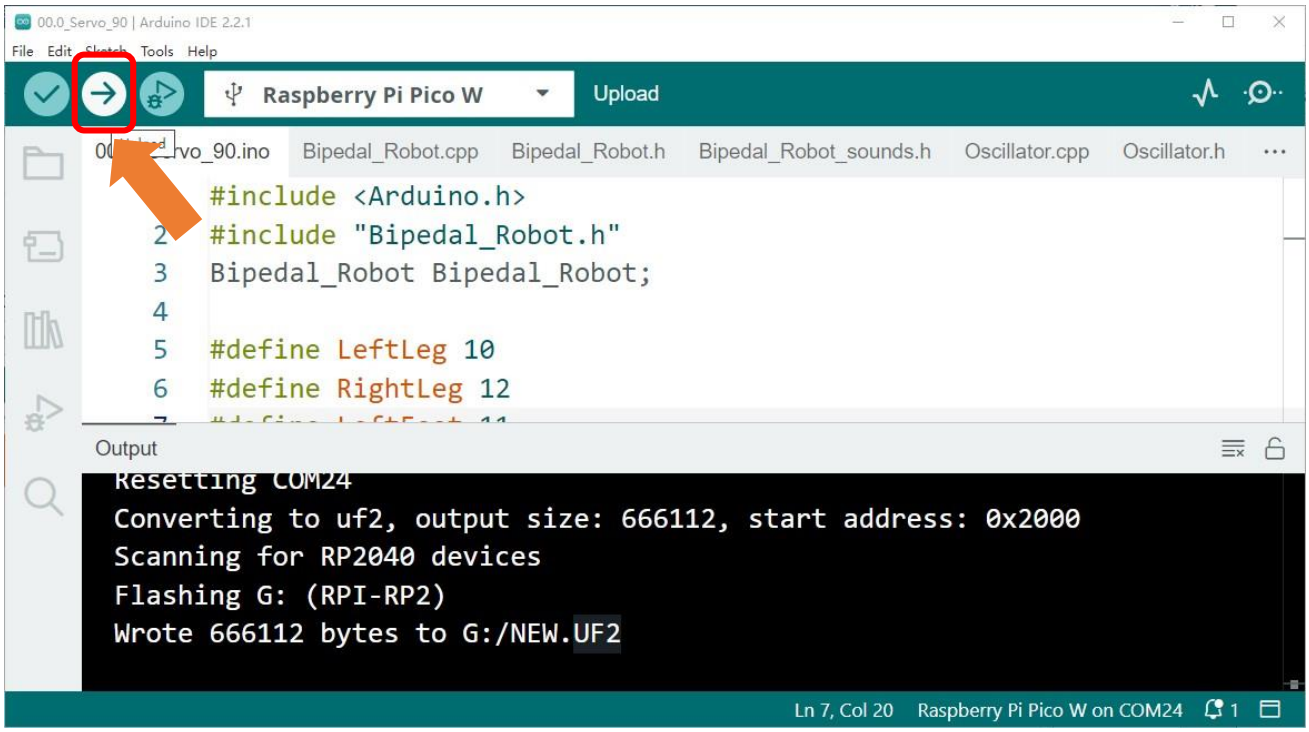


Step 4. Select serial port.

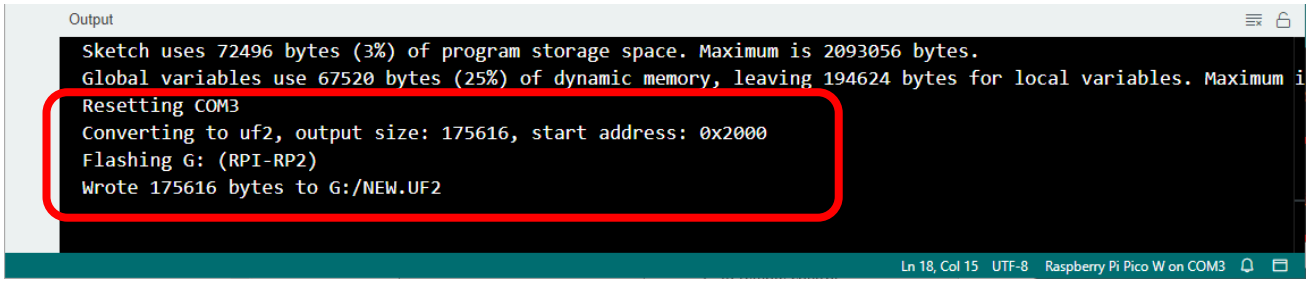
Click Tools on Menu bar, navigate your mouse to Port and select COMx on your computer. The value of COMx varies in different computers, but it will not affect the download function of Raspberry Pi Pico (W), as long as you select the correct one.



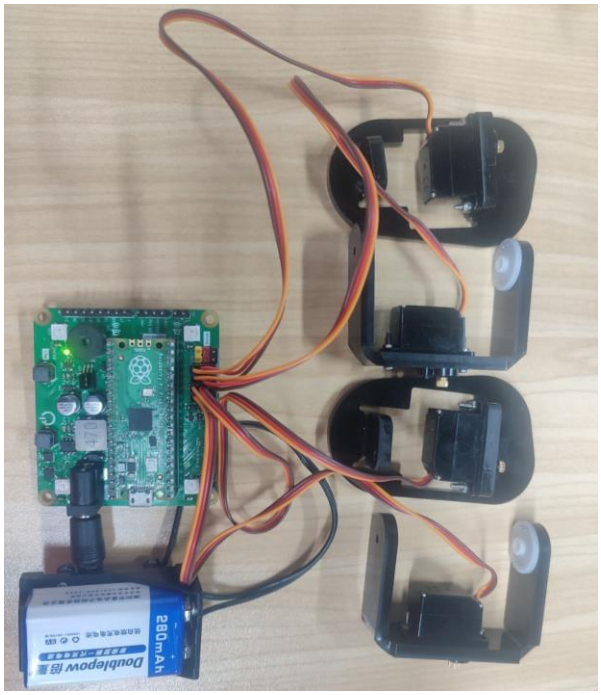
Click "Upload Using Programmer" and the program will be downloaded to Raspberry Pi Pico (W).



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



Connect the servo cables to the robot board, install the battery and press the power switch. The gears of the four servos will rotate to 90°.



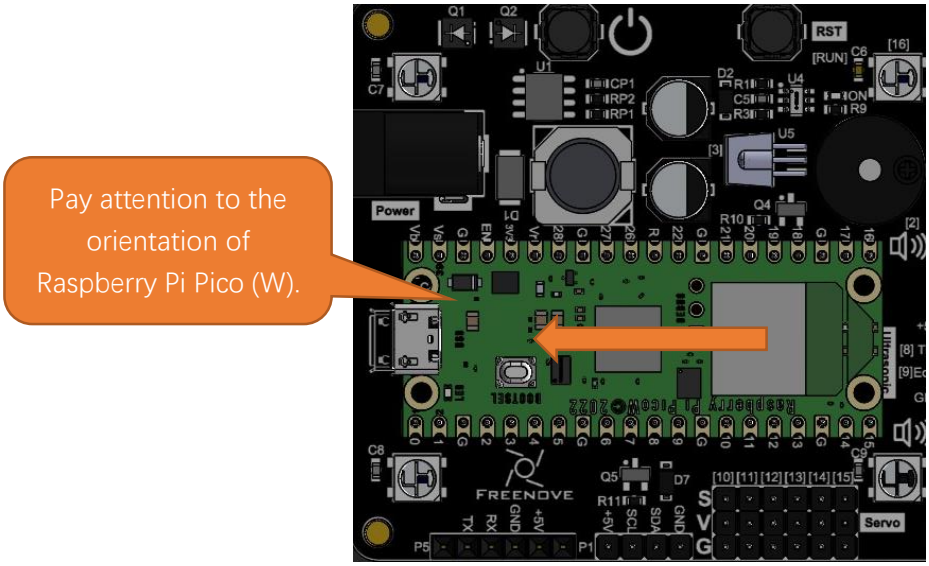
Chapter 0 Assembling Bipedal Robot

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

Assembling the Robot

Plugging in Raspberry Pi Pico (W)

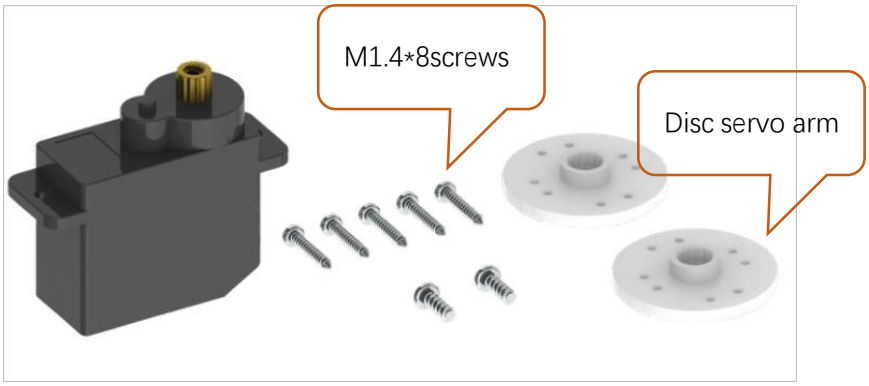
Plug Raspberry Pi Pico (W) into the shield.



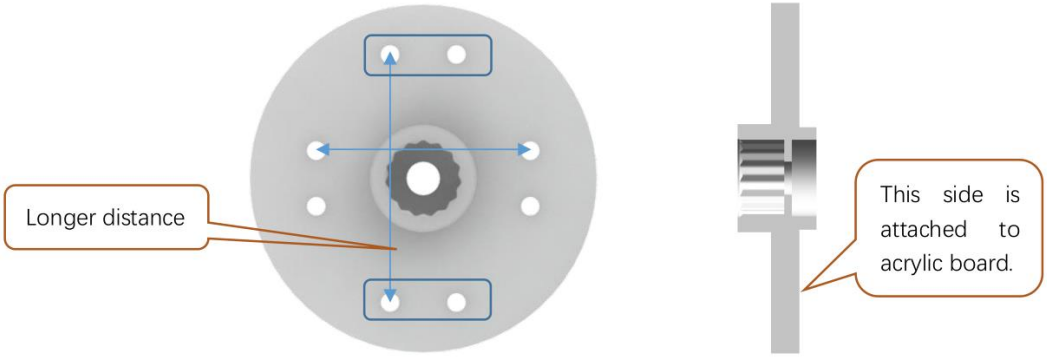
Please pay attention to the orientation of Raspberry Pi Pico (W). Do NOT reverse it; Otherwise, it may burn the Raspberry Pi Pico (W).

Installing Disk Servo Arm

Take out four disk servo arms from the servo packages.

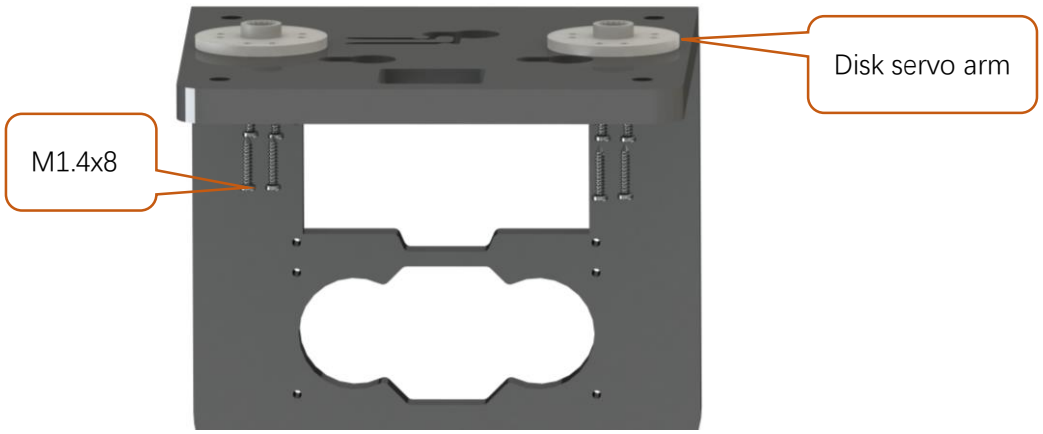


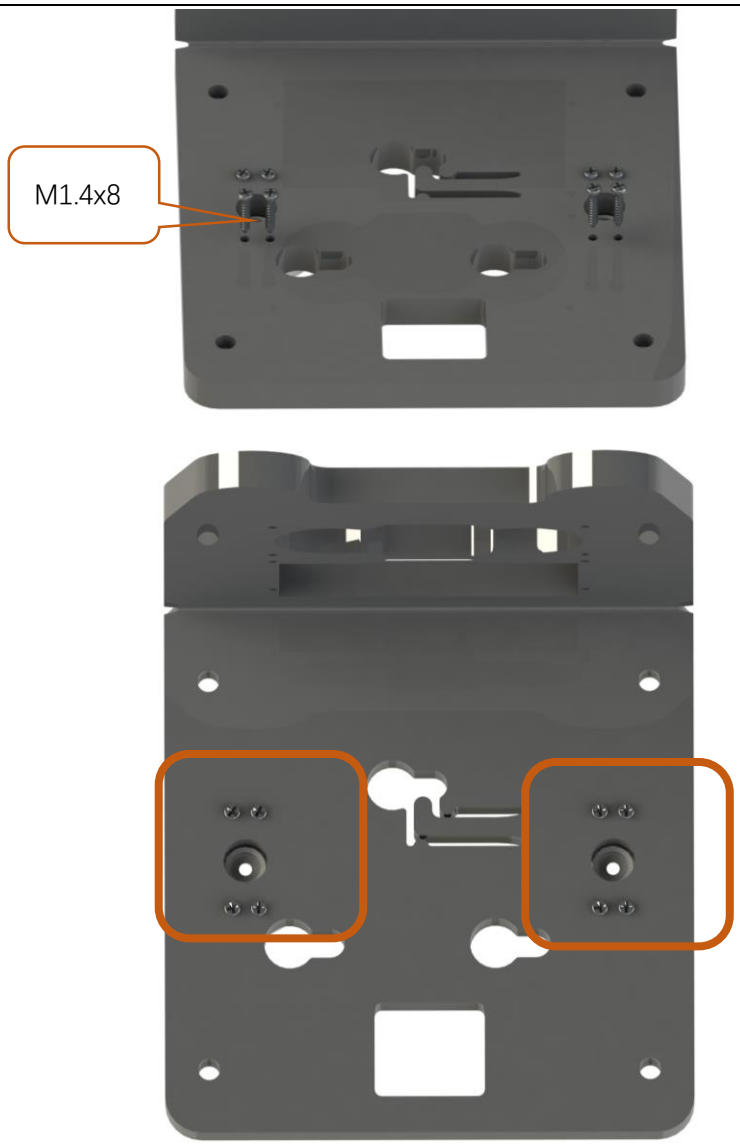
Each servo package includes five M1.4x8 screws and two black servo screws.
There are 4 pairs of opposite holes on the disk servo arm, and the distance between each pair is different.
Please use the 2 pairs of holes with **longer distance.**



Installation steps:

Step 1 The following illustrates the installation of the disc servo arms of the servos for the body. Use the M1.4x8 screws in the servo package.



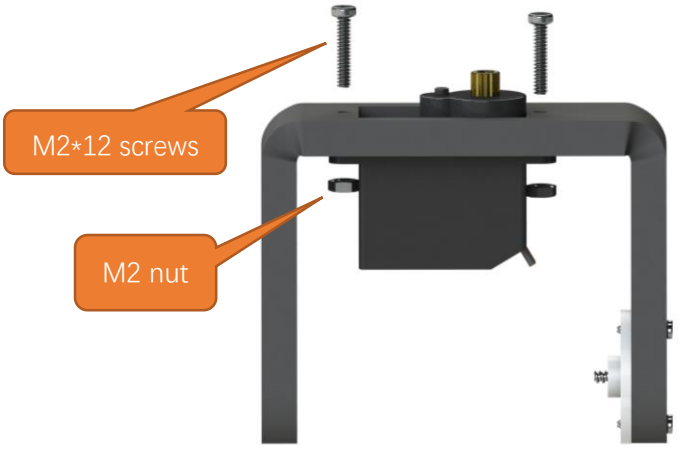


Step 2 The following illustrates the installation of the disc servo arms of the servos for the leg. Use the M1.4x8 screws in the servo package.

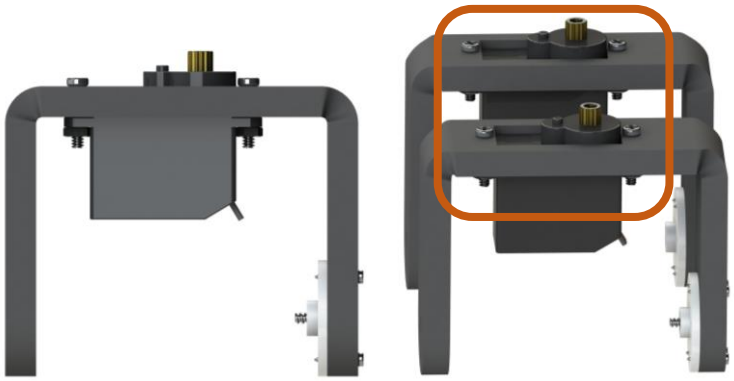
Installing Servo

Installation steps:

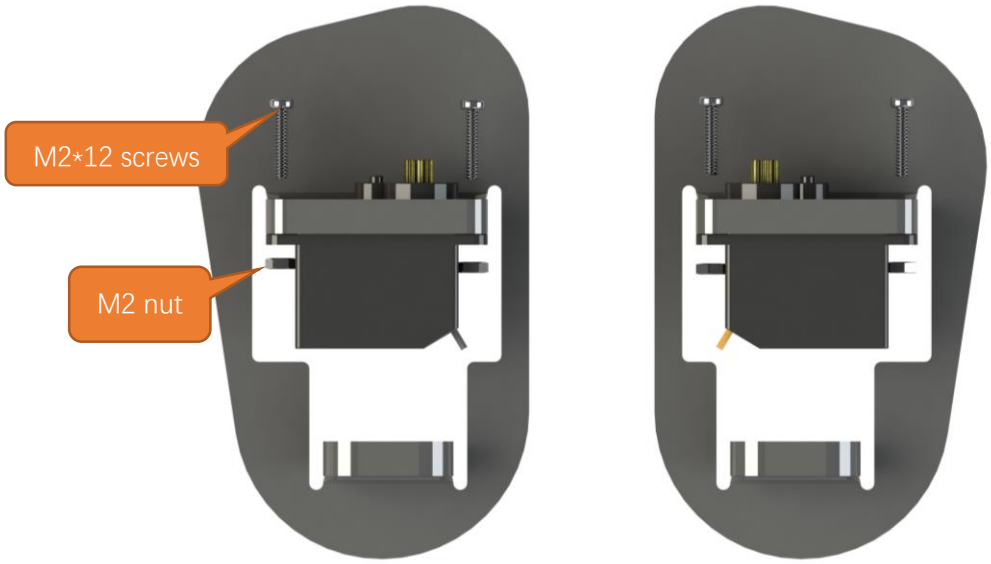
Step 1 Assemble the servos to the two legs with M2*12 screws and M2 nuts.



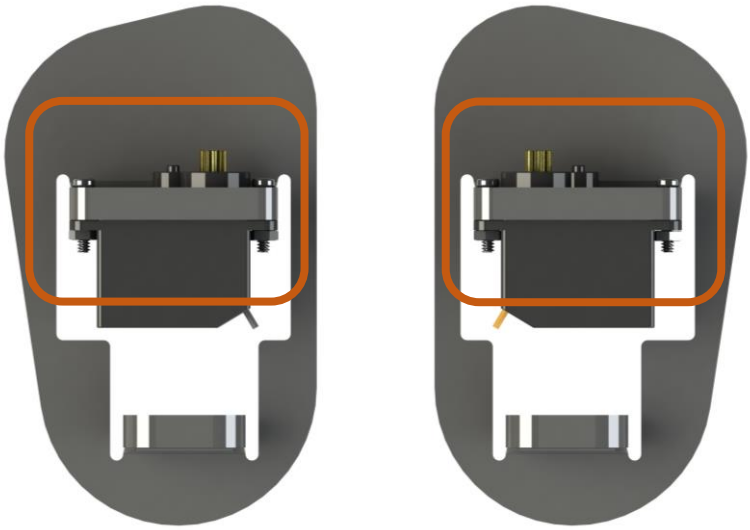
Finished:



Step 2 Assemble the servos to both feet with M2*12 screws and M2 nuts.



Finished:



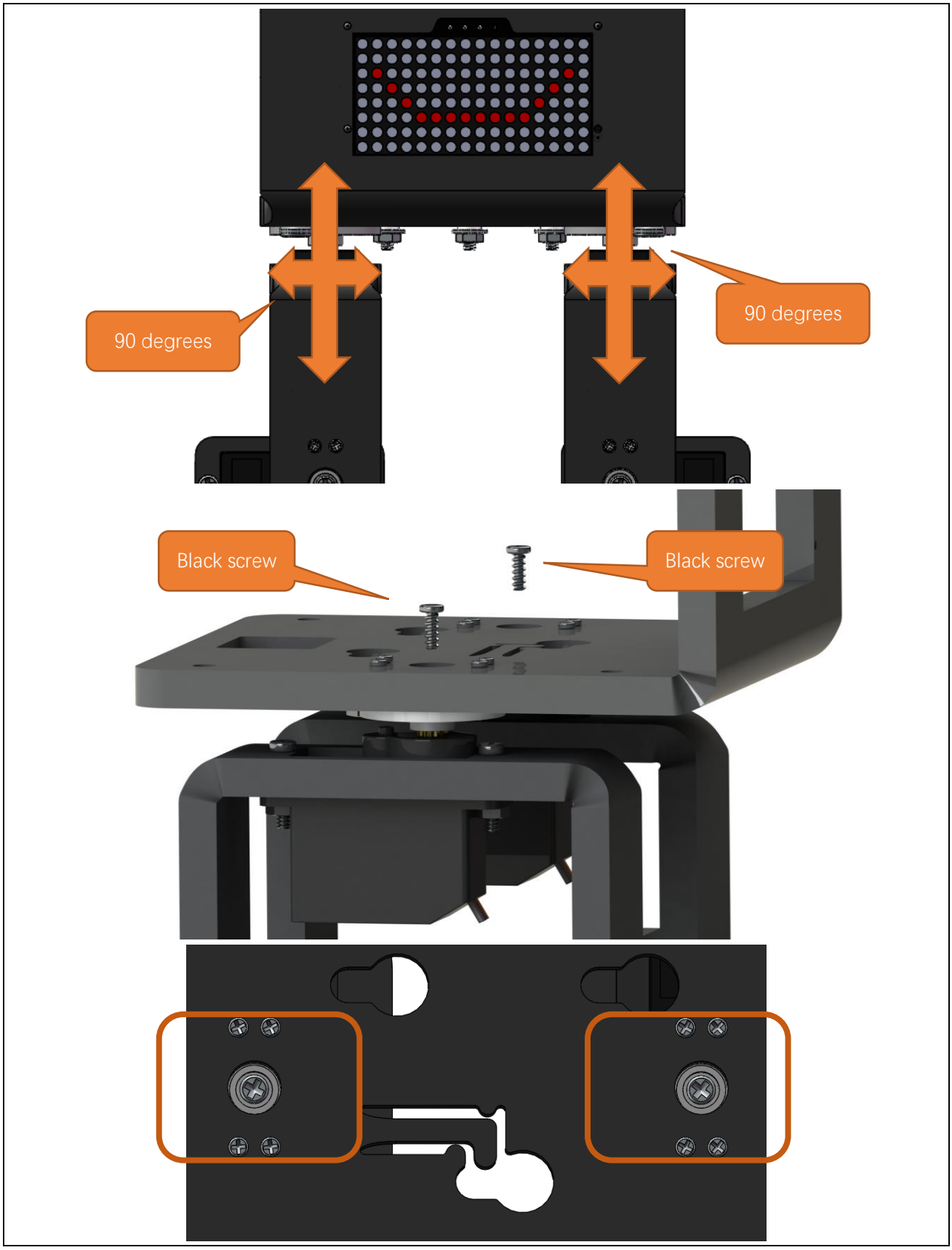
Step 3 Assemble the servos to the robot. Connect Servos 1, 2, 3, and 4 to the corresponding port on the robot board, which are GPIO10, GPIO11, GPIO12, and GPIO13 respectively. Pay attention to the colors of the cables. Connect the yellow cable to the yellow pin, red to red and brown to black. Do not misalign the cables.

Open “00.0_Servo_90” in “Freenove_Bipedal_Robot_Kit_for_Raspberry_Pi_Pico\Sketches” and double-click “00.0_Servo_90.ino”. For details, please refer to here.

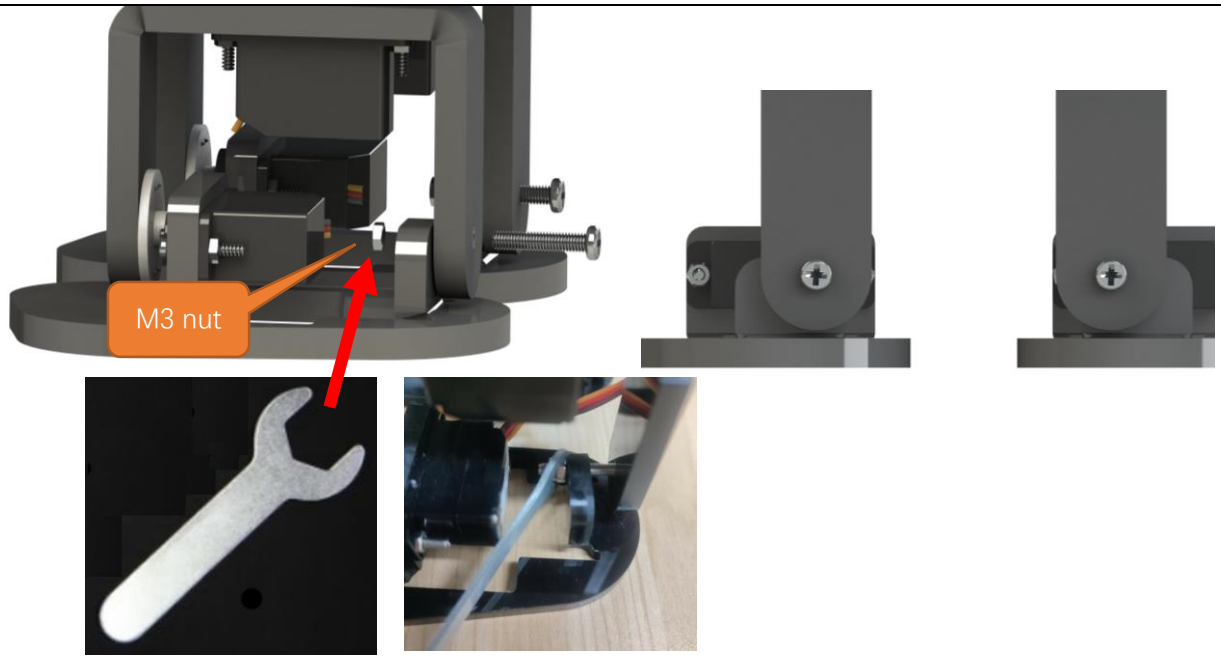
Turn ON the power switch, run the servo installation program, and the servos will keep at 90°.

Assemble the servos to the acrylic parts with the black screws in the servo packages.

Make them as close to 90° as possible. It is acceptable that the angle is in the range of 80 – 100 degrees.


90 degrees
90 degrees
Black screw
Black screw

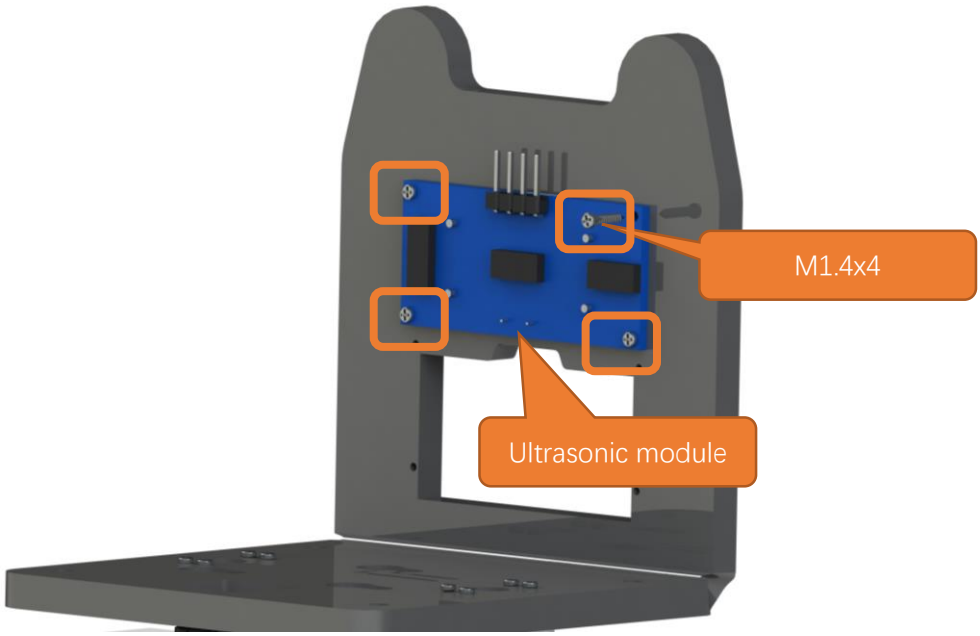
Assemble the leg support screws: Fix the M3 nut with the M3 spanner and screw the the M3 screws in. Please note that the M3 nuts here are self-locking non-slip nuts and the srews serve as supports, so you do not need to screw them too tight and some space is needed, so that the feet can move more flexible. If your robot is stuck when walking, please check if the screws are fixed too tight.



When assembling the leg supporting rivets with the spanner, you can tilt it at certain angles to facilitate assembly.

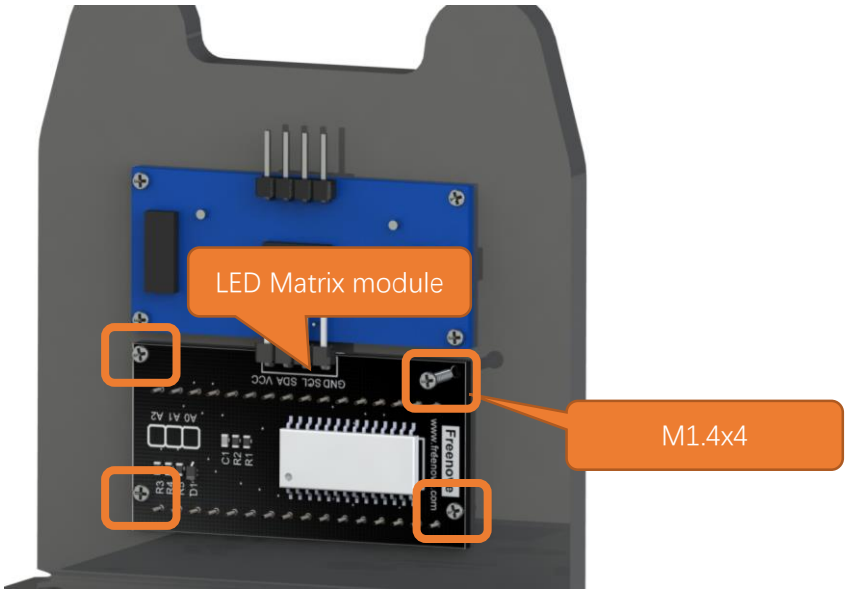
Installing Ultrasonic Module

Step 1 Install ultrasonic module.



Installing LED Matrix Module

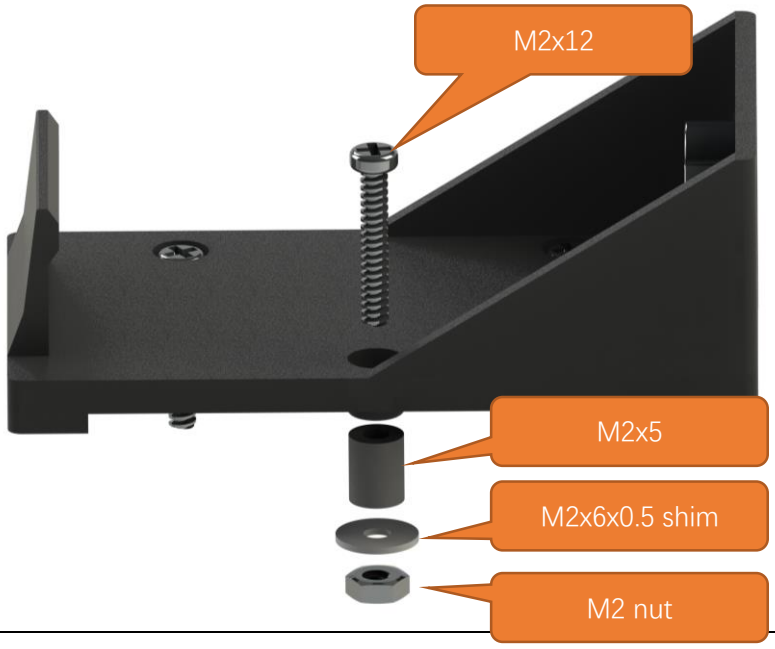
Step 1 Install LED matrix module.



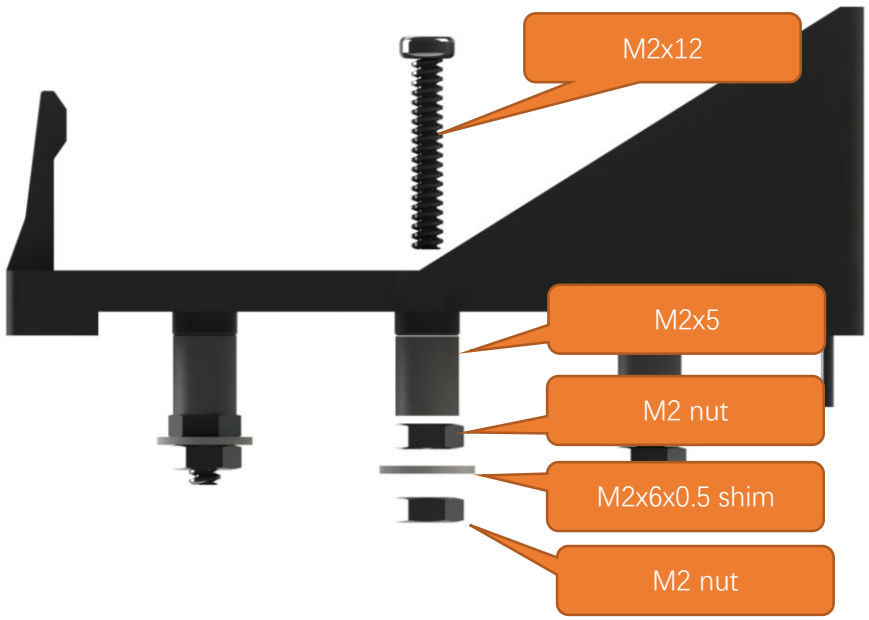
Installing 9V Batteries

This robot is powered with **9V alkaline batteries** or **9V NIMH rechargeable batteries** as power supply. Please note that 9V carbon batteries and 9V rechargeable lithium batteries **cannot** provide enough power for the robot. Therefore, it is important to pay attention to the battery model when purchasing batteries.

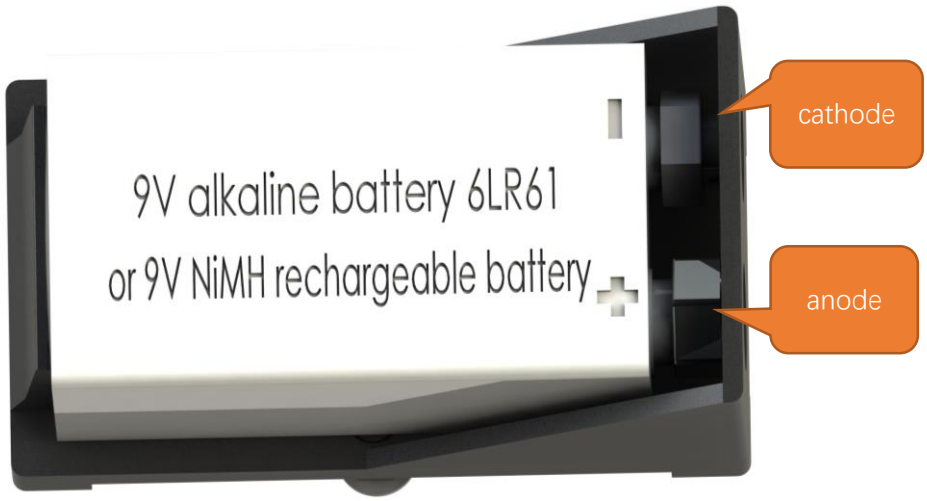
Step 1 Installed with batteries.
Correctly install the batteries.
Here we provide two installation methods:
Method 1:



Method 2:

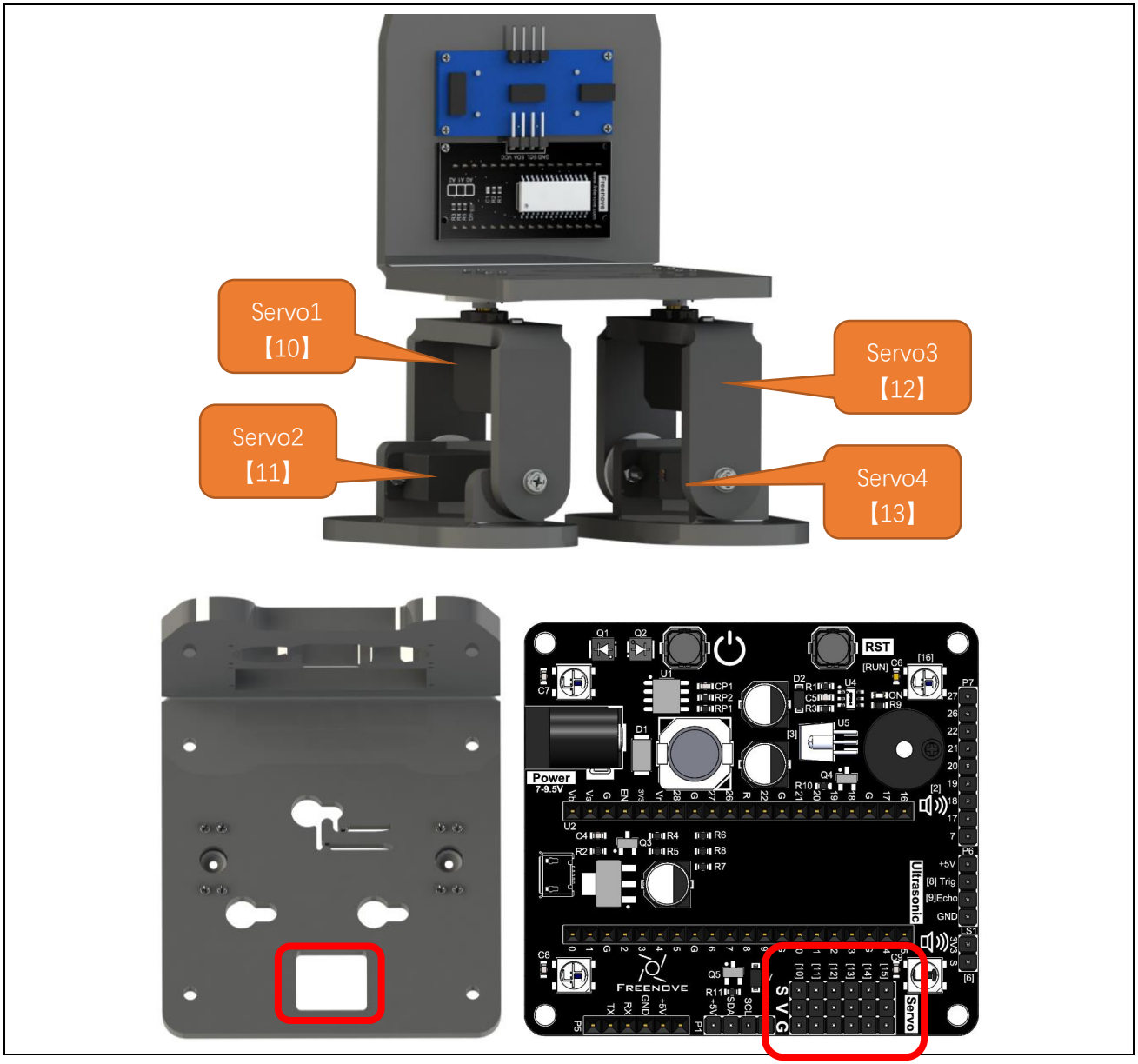


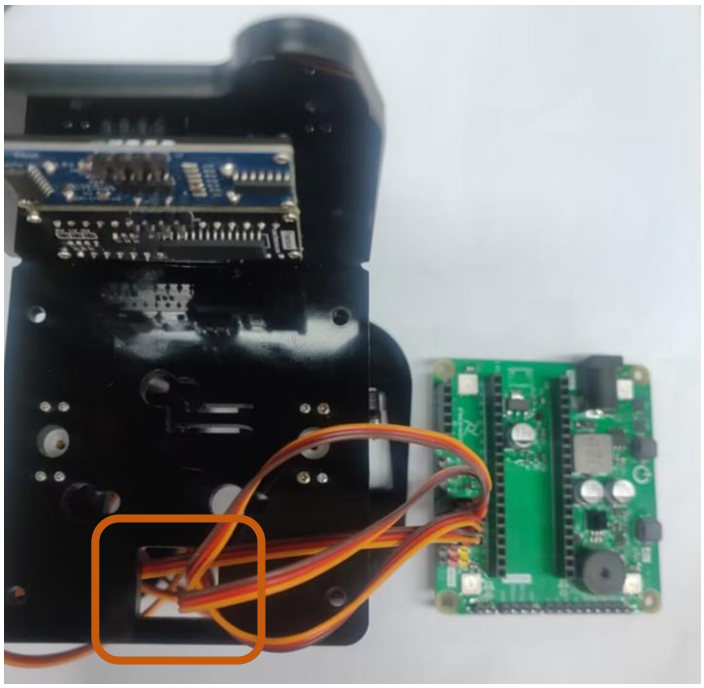
Among them, M2x5 is the acrylic element, and there are 6 M2x5 rings in the acrylic fitting. Only three are used here, and the remaining three are reserved for use.



Connecting Servo Cables

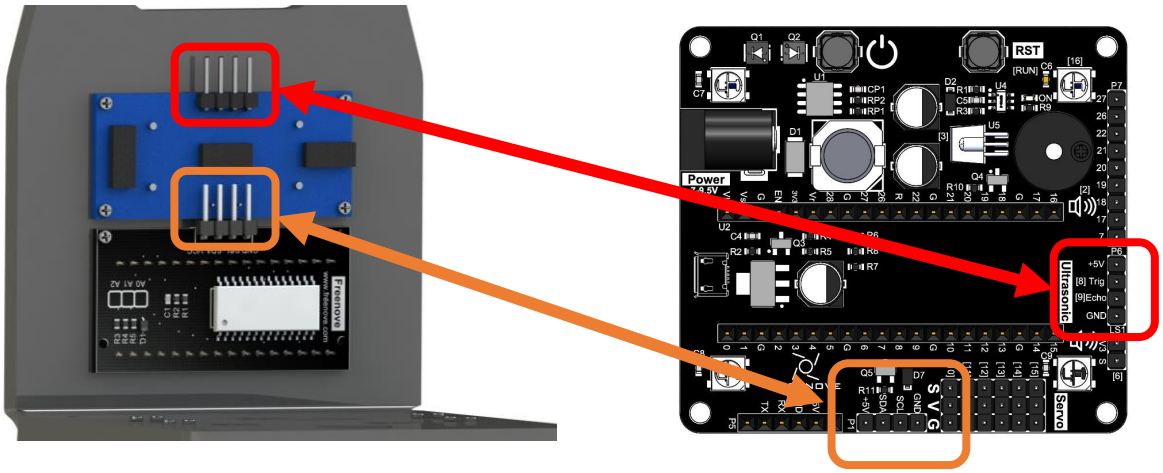
Step 1 Connect the servo cables. Thread the servo wires through the hole on the acrylic part and connect them to the servo port on the robot shield.


Servo1
【10】
Servo2
【11】
Servo3
【12】
Servo4
【13】
RST
Power
7-9.5V
Ultrasonic
Servo
FREENOVE



Connect the Ultrasonic Sensor and LED Matrix Module

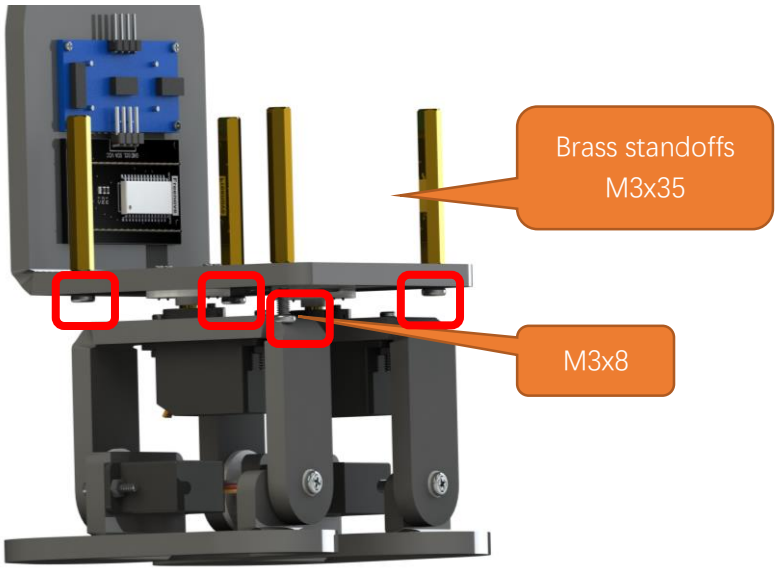
Step 1 Connect the ultrasonic sensor and LED matrix module. It is important that the pins be connected in line with the silk print on the robot shield.



Installing the Standoffs for Robot Shield

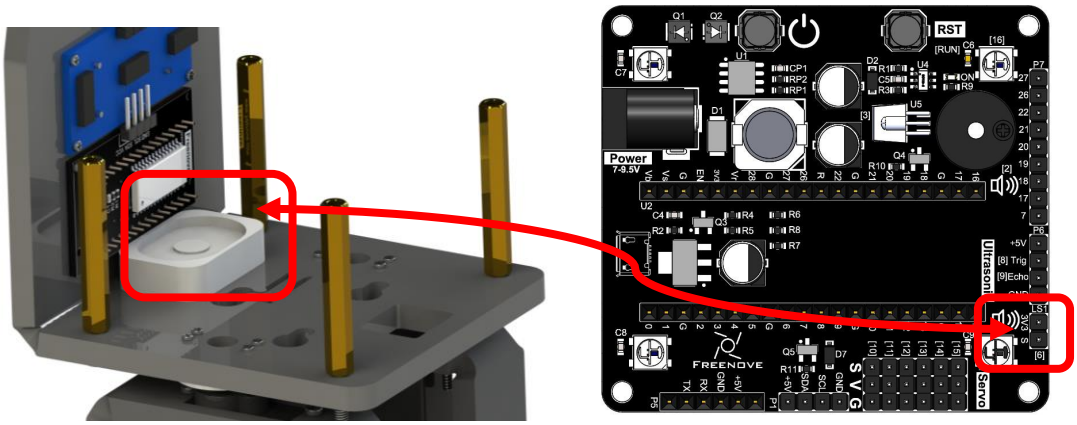
Fix the robot shield.

Step 1



Installing the Speaker

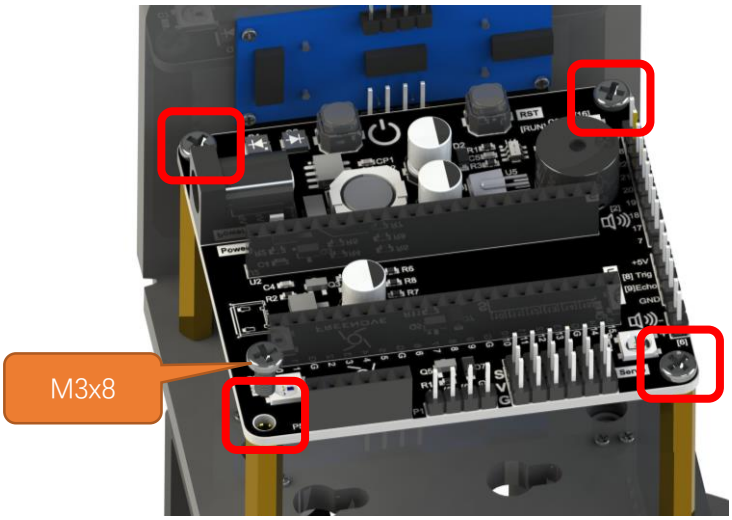
Step 1 Attach the speaker to the robot.



The red line of the speaker here is connected to the interface marked 3V3 on the control board.

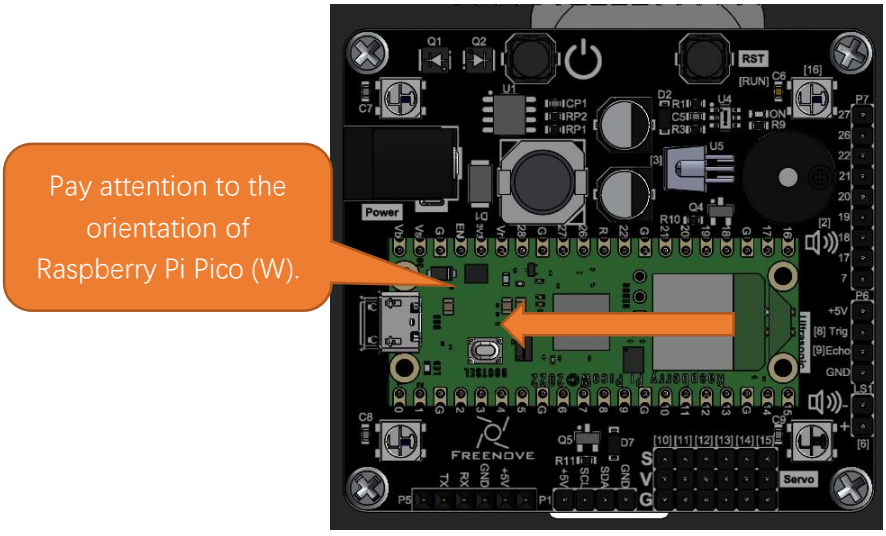
Installing Robot Shield

Step 1 Install the robot shield.



Plugging in Raspberry Pi Pico (W)

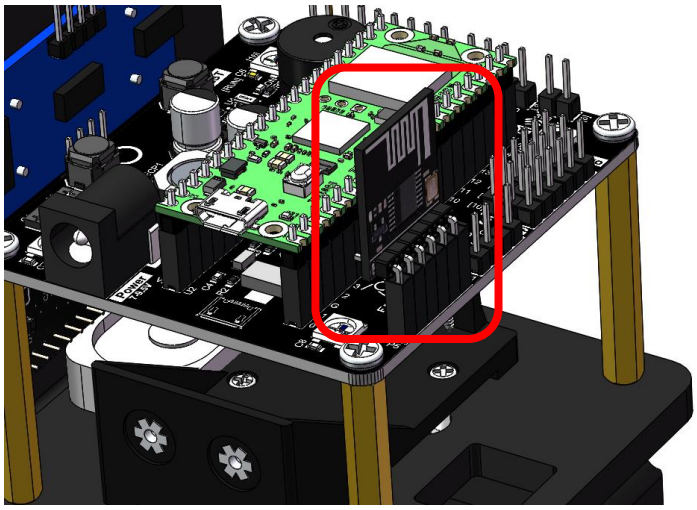
Plug Raspberry Pi Pico (W) into the shield.



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

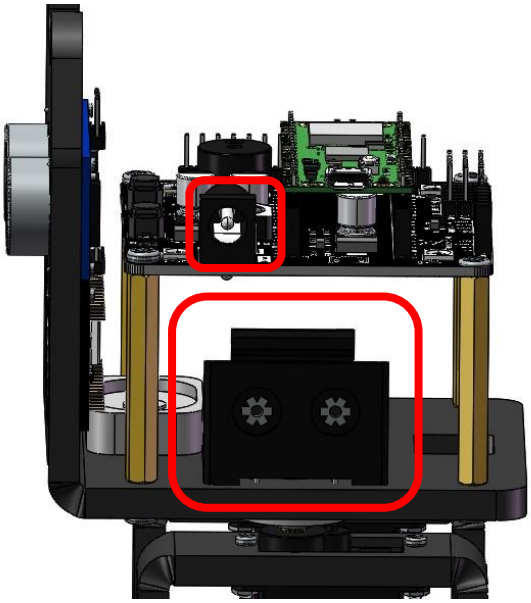
Installing Bluetooth Module

Plug the Bluetooth module to the board. Pay attention to the direction.



Installing 9V Battery Holder (with batteires installed)

Install the batteries to power the robot. Please note that the input voltage should be 7-9.5V. Although it is a 9V battery, its actual voltage can exceed this number. It is acceptable as long as the voltage is within 10V. When installing the batteris, you may press the battery down and push it to the end.

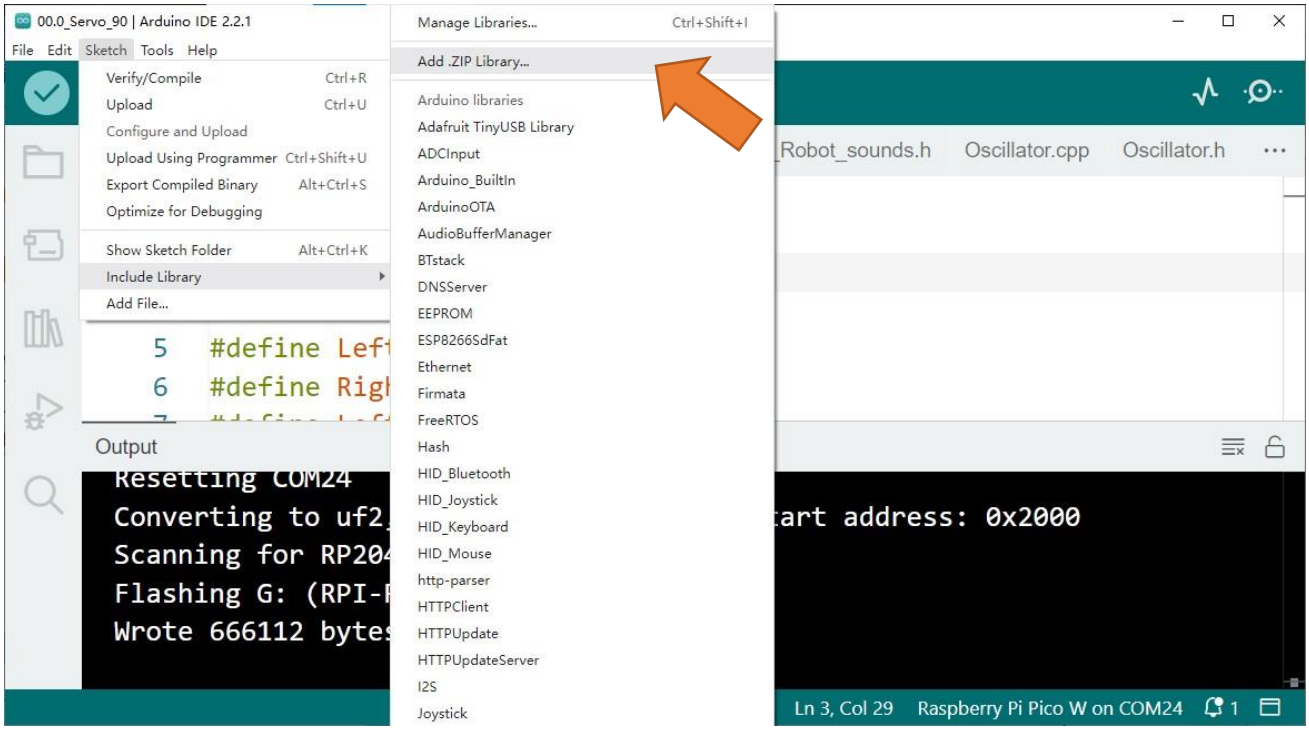


Note: When the robot is not in use, please take out the batteries to avoid draining them.
It is recommended to buy rechargeable batteries; whichi is more environment-friendly.

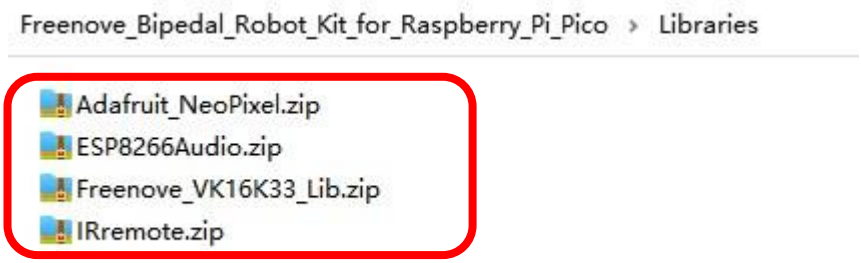
How to Play

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_Bipedal_Robot_Kit_for_Raspberry_Pi_Pico\Libraries**, select every Library, click **Open**, and repeat this process until you have installed all six Libraries into the Arduino.



These libraies are used for this robot. Before uploading code, please make sure they have beed added.

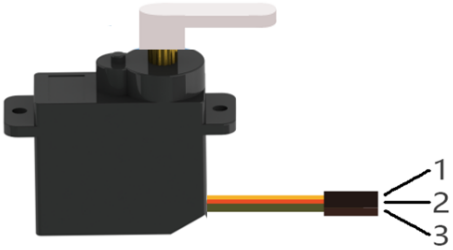
Chapter 1 Module test

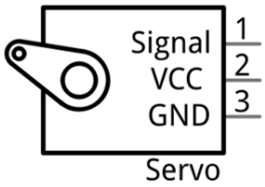
If you have any concerns, please feel free to contact us via support@freenove.com

1.1 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.



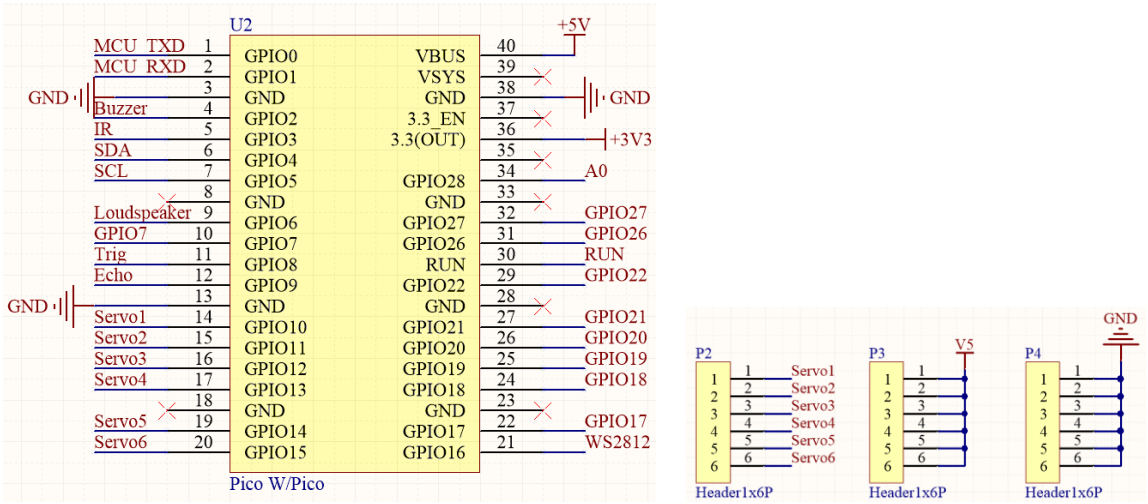


We will use a 50Hz PWM signal with a duty cycle in a certain range to drive the Servo. The lasting time of 0.5ms-2.5ms of PWM single cycle high level corresponds to the servo angle 0 degrees - 180 degree linearly. Part of the corresponding values are as follows:

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



Clibration for the Four Servos

Here are three ways to calibrate the servos.
The first two methods require connecting the pico board to your computer with an USB cable, while the third one is achieved via Bluetooth, which is a wireless way. You can chose either method, as you prefer.

Nevertheless, the third method is more recommended, as the USB cable may affect the movement of the robot. It is more convenient and quicker to use the third method for calibration.

Method 1:
Calibrate through serial communication.

Upload sketch 01.1.1_Servo_Calibration to the Raspberry Pi Pico. After uploading successfully, click the serial monitor on Arduino IDE. Press the corresponding keys, on the keyboard to calibrate each servo.
The specific calibration command characters are as follows:

Servo interface	The servo was increased by 1°	The servo is reduced by 1°
LeftLeg [10]	q	a
RightLeg [12]	e	d
LeftFoot [11]	w	s
RightFoot [13]	r	f

Enter o to save the calibrate data to the pico board.

Method 2:
This method also uses serial communication to calibrate the servos. There are two ways to achieve this.

1. Directly run the executable file to calibrate. You only need to double click the file to run it.
2. Enter commands to run the calibration tool. If you choose this, you need the following preparation:
   2.1 Install python on your computer.
   2.2 Install pyseiral library by running the command pip3 install pyserial.

```
C:\Users\Freenove>pip3 install pyserial
Defaulting to user installation because normal site-packages is not writeable
Collecting pyserial
  Downloading pyserial-3.5-py2.py3-none-any.whl.metadata (1.6 kB)
Using cached pyserial-3.5-py2.py3-none-any.whl (90 kB)
Installing collected packages: pyserial
Successfully installed pyserial-3.5

C:\Users\Freenove>
```

   2.3 Enter the command to enter the directory where the file locates, and then run python calibration.py to run the calibration tool.

```
C:\Users\Freenove>cd C:\Users\Freenove\Desktop\Freenove_Bipedal_Robot_Kit_for_Raspberry_Pi_Pico\Calibration_tool\Code

C:\Users\Freenove\Desktop\Freenove_Bipedal_Robot_Kit_for_Raspberry_Pi_Pico\Calibration_tool\Code>python calibration.py
```

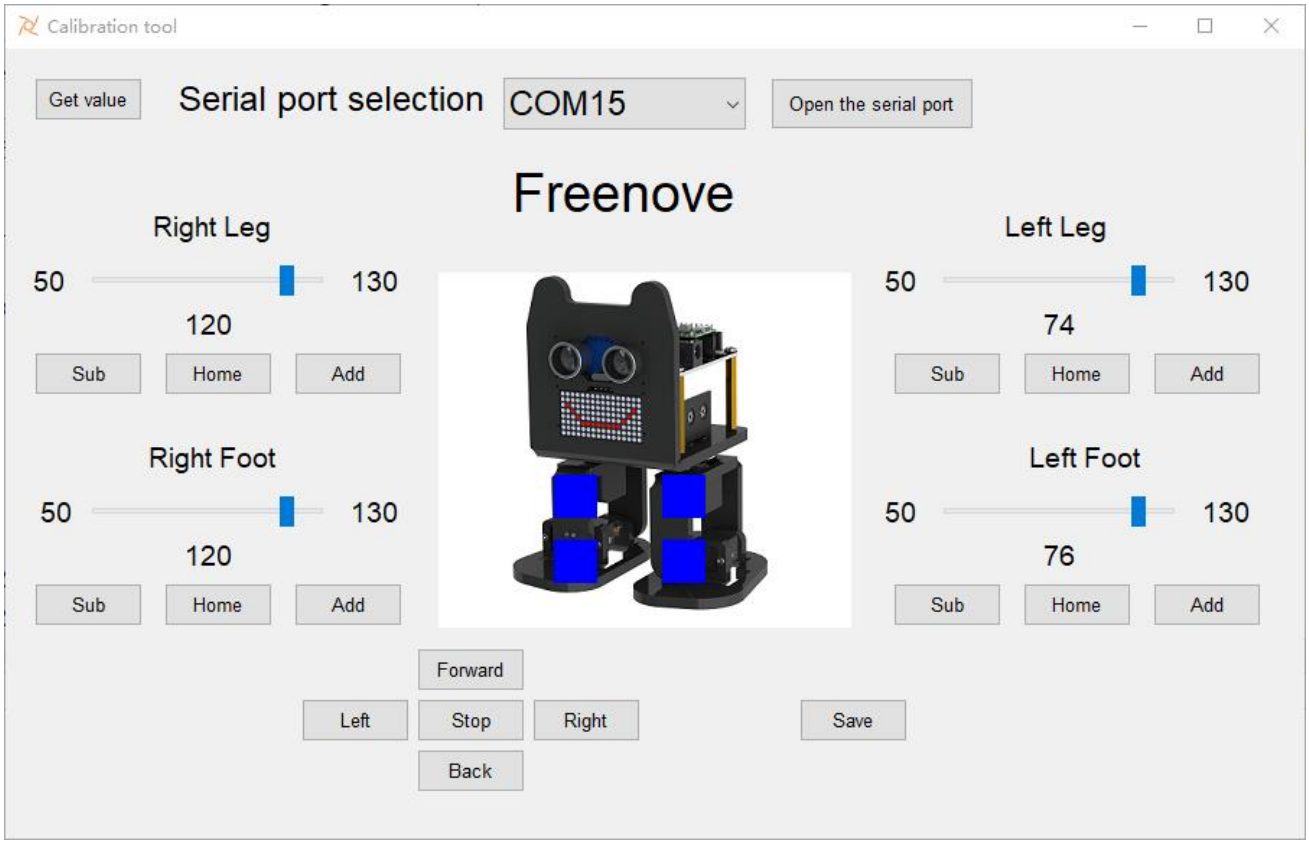
You will see the calibration tool open after the above steps.

Before using the calibration tool, you need to upload sketch 01.1.2_Servo_Calibration_PC_Tools to the pico board.

The following shows you how to calibrate the servos with the tool.

1. Select the serial port. (Make sure that the port is not occupied by your computer, and that the serial debugger tool on Arduino IDE is close, otherwise the connection will fail.)
2. Click the servo to calibrate. The four blue boxes represent the four servos. When one is selected, it will turn red. Calibrate the four servos one by one.
3. After all the four servos are calibrated, click the movement buttons (Forward, Back, Left, Right, Stop) to see if the servos work correctly.
4. If they work as expected, click the Save button.
5. If the servos does not work correctly in future operation, please recalibrate them.

Note: When you open the calibration tool next time, you can click the Get value button to get the servo angle data saved on the pico board. In this way, the servos can roate back to the last calibration value. You can calibration based on this value.



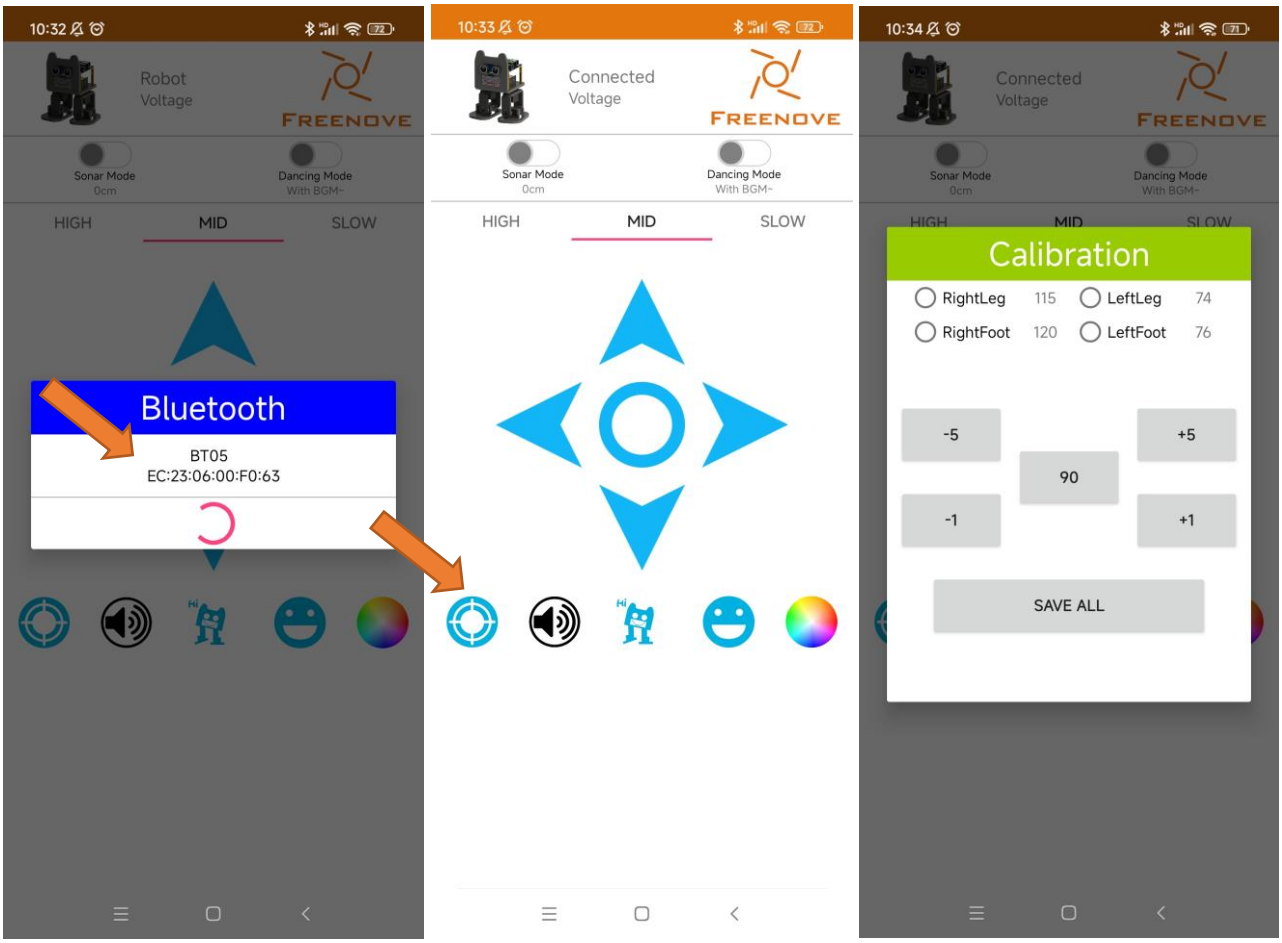
The above two methods require connecting the pico board to computer with an usb cable.

Method 3 (wireless):
Calibrate the servo angles via Bluetooth wireless connection.

1. Upload the corresponding code to pico (see the sketch section for detail). Open Freenove app on your phone and connect to the Bluetooth named “BT05”.
2. After connecting successfully, tap the calibration icon on the app. At this point, it will get the angle data of the four servos.
3. Select each servo to calibrate.
4. Tap the SAVE ALL button after the four servos are calibrated, and close the calibration window.
5. Test the robot’s movement by tapping the movement buttons on the homepage. Check whether the servos move in line with the button tapped.
6. Similarly, if the servos work abnormally, you can recalibrate them.

Notes:

1. Please make sure the Bluetooth module is correctly connected to avoid burning it.
2. Please take out the batteries when the robot is put idle to avoid draining them.

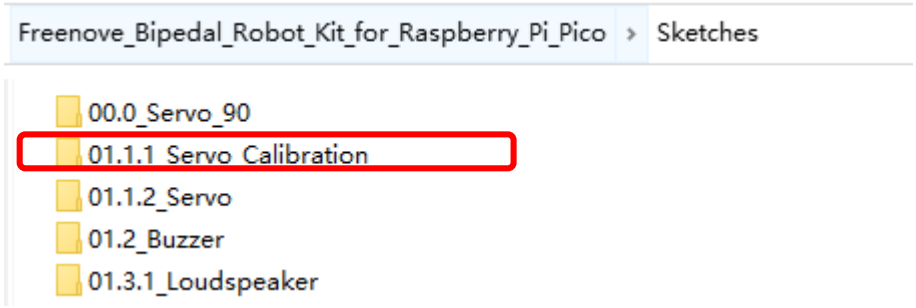

10:32
Robot
Voltage
FREENOVE
Sonar Mode
0cm
Dancing Mode
With BGM~
HIGH
MID
SLOW
Bluetooth
BT05
EC:23:06:00:F0:63
10:33
Connected
Voltage
FREENOVE
Sonar Mode
0cm
Dancing Mode
With BGM~
HIGH
MID
SLOW
10:34
Connected
Voltage
FREENOVE
Sonar Mode
0cm
Dancing Mode
With BGM~
Calibration
RightLeg 115
LeftLeg 74
RightFoot 120
LeftFoot 76
-5
+5
90
-1
+1
SAVE ALL

Sketch

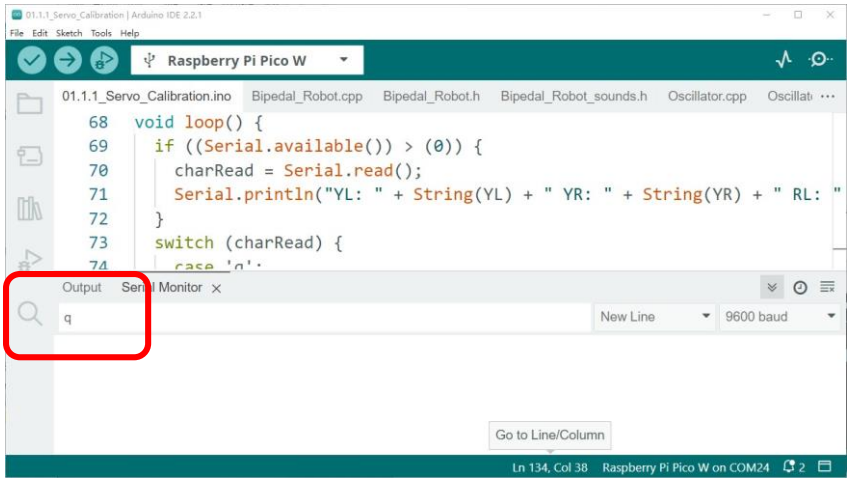
01.1.1_Servo_Calibration

Upload the sketch to Raspberry Pi Pico (W). This sketch is for servo calibration. You can adjust the robot's status via keyboard input, until the robot can stand up normally. After calibration, you can send the character **t** for the robot to move forward. Pay attention to the robot's movement, if it does not walk well, you can adjust each servo again until the robot's movement is normal. After the calibration finishes, you need to modify the code according to the calibration data and upload the sketch again. Refer to the sketch for more details.

Open "01.1.1_Servo_Calibration" in **"Freenove_Bipedal_Robot_Kit_for_Raspberry_Pi_Pico\Sketches"** and double-click "01.1.1_Servo_Calibration.ino".



As indicated below, a character is sent via the serial monitor to calibrate the robot.



The specific calibration command characters are as follows:

Servo interface	The servo was increased by 1°	The servo is reduced by 1°
LeftLeg [10]	q	a
RightLeg [12]	e	d
LeftFoot [11]	w	s
RightFoot [13]	r	f

Press O to save the current servo angle data to the pico board.

Code

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

```
#include <Wire.h>
#include <SoftwareSerial.h>
#include <EEPROM.h>
#include "Bipedal_Robot.h"
Bipedal_Robot Bipedal_Robot;

#define LeftLeg 10
#define RightLeg 12
#define LeftFoot 11
#define RightFoot 13

int calibratePosition[4] = { -17, -10, -15, 12 };
// Used to save the servo offset, in radians
int YL;
int YR;
int RL;
int RR;
char charRead;
void setup() {
  EEPROM.begin(512);
  Serial.begin(9600);
  for (int i = 0; i < 4; i++) {
    EEPROM.write(i, calibratePosition[i]);
  }
  Bipedal_Robot.init(LeftLeg, RightLeg, LeftFoot, RightFoot, true);  //Set the servo pins
  YL = EEPROM.read(0);
  if (YL > 128) YL -= 256;
  YR = EEPROM.read(1);
  if (YR > 128) YR -= 256;
  RL = EEPROM.read(2);
  if (RL > 128) RL -= 256;
  RR = EEPROM.read(3);
  if (RR > 128) RR -= 256;
  calib_homePos();
  Bipedal_Robot.saveTrimsOnEEPROM();
  EEPROM.commit();
  Bipedal_Robot.home();
  Serial.println("ROBOT CALIBRATION PROGRAM");
  Serial.println("PRESS q or a for adjusting Left Leg");
  Serial.println("PRESS w or s for adjusting Left Foot");
  Serial.println("PRESS e or d for adjusting Right Leg");
  Serial.println("PRESS r or f for adjusting Right Foot");
  Serial.println();
  Serial.println("PRESS t to test Otto walking");
```

```
46    Serial.println("PRESS g to return servos to home position");
47  }
48
49  void loop() {
50    if ((Serial.available()) > (0)) {
51      charRead = Serial.read();
52      Serial.println("YL: " + String(YL) + " YR: " + String(YR) + " RL: " + String(RL) + " RR: "
53  + String(RR));
54    }
55    switch (charRead) {
56      case 'q':
57        YL++;
58        Bipedal_Robot.setTrims(YL, YR, RL, RR);
59        calib_homePos();
60        Bipedal_Robot.saveTrimsOnEEPROM();
61        break;
62      case 'a':
63        YL--;
64        Bipedal_Robot.setTrims(YL, YR, RL, RR);
65        calib_homePos();
66        Bipedal_Robot.saveTrimsOnEEPROM();
67        break;
68      case 'w':
69        RL++;
70        Bipedal_Robot.setTrims(YL, YR, RL, RR);
71        calib_homePos();
72        Bipedal_Robot.saveTrimsOnEEPROM();
73        break;
74      case 's':
75        RL--;
76        Bipedal_Robot.setTrims(YL, YR, RL, RR);
77        calib_homePos();
78        Bipedal_Robot.saveTrimsOnEEPROM();
79        break;
80      case 'e':
81        YR++;
82        Bipedal_Robot.setTrims(YL, YR, RL, RR);
83        calib_homePos();
84        Bipedal_Robot.saveTrimsOnEEPROM();
85        break;
86      case 'd':
87        YR--;
88        Bipedal_Robot.setTrims(YL, YR, RL, RR);
89        calib_homePos();
```

```
90          Bipedal_Robot.saveTrimsOnEEPROM();
91          break;
92        case 'r':
93          RR++;
94          Bipedal_Robot.setTrims(YL, YR, RL, RR);
95          calib_homePos();
96          Bipedal_Robot.saveTrimsOnEEPROM();
97          break;
98        case 'f':
99          RR--;
100         Bipedal_Robot.setTrims(YL, YR, RL, RR);
101         calib_homePos();
102         Bipedal_Robot.saveTrimsOnEEPROM();
103         break;
104       case 't':
105         Bipedal_Robot.walk(1, 2000, 1);
106         Bipedal_Robot.home();
107         break;
108       case 'g':
109         Bipedal_Robot.home();
110         EEPROM.commit();
111         break;
112       case '2':
113         Bipedal_Robot.walk(5, 2000, -1);
114         break;
115       case '3':
116         Bipedal_Robot.turn(5, 2000, 1);
117         break;
118       case '4':
119         Bipedal_Robot.turn(5, 2000, -1);
120         break;
121       default:
122         break;
123     }
124 }
125
126 void calib_homePos() {
127   int servoPos[4];
128   servoPos[0] = 90;
129   servoPos[1] = 90;
130   servoPos[2] = 90;
131   servoPos[3] = 90;
132   Bipedal_Robot._moveServos(500, servoPos);
133 }
```

Code Explanation

Include the header files for controlling the robot. Each time before the servos being controlled, the header files need to be included first.

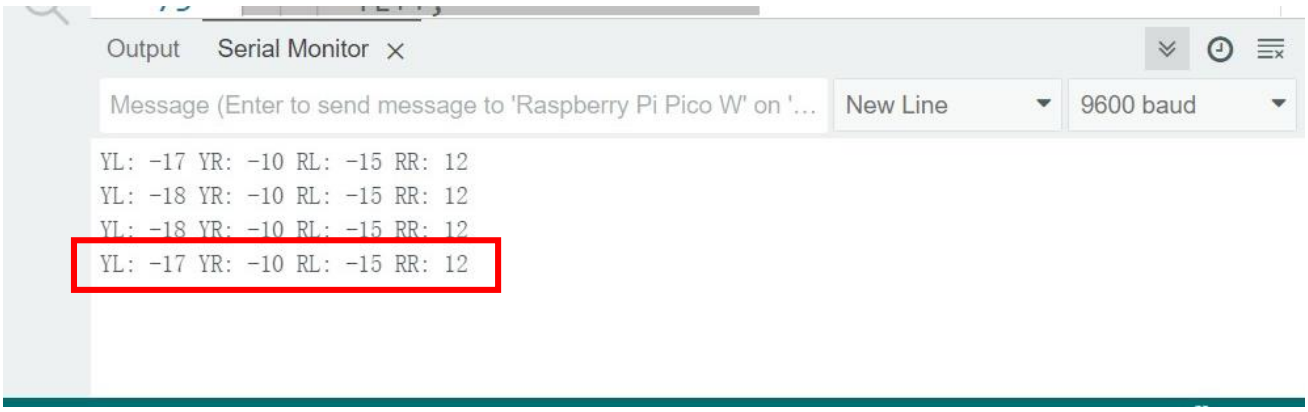
```
#include <Arduino.h>
#include <Wire.h>
#include <SoftwareSerial.h>
#include <EEPROM.h>
#include "Bipedal_Robot.h"
```

The serial monitor prints the current servo calibration value when receiving data.

```
  if ((Serial.available()) > (0)) {
    charRead = Serial.read();
    Serial.println("YL: " + String(YL) + " YR: " + String(YR) + " RL: " + String(RL) + " RR: "
+ String(RR));
  }
```

After finish calibration, it will record the final calibration data. Please modify the code according to the final data and upload the code again.

```
int calibratePosition[4] = { -17, -10, -15, 12 };
```



01.1.2_Servo_Calibration_PC_Tools

Upload this sketch to the Raspberry Pi Pico (W). This sketch corresponds to the PC Servo Calibration Tool. You can calibrate the servo angle by clicking the corresponding servo button. After the calibration is completed, click the Save button. When testing the robot's movement, if it does not move well, you can readjust each servo until the robot moves normally.

Code

```
1  #include <Arduino.h>
2  #include <Wire.h>
3  #include <SoftwareSerial.h>
4  #include <EEPROM.h>
5  #include "Bipedal_Robot.h"
6  Bipedal_Robot Bipedal_Robot;
7
```

```
#define LeftLeg 10
#define RightLeg 12
#define LeftFoot 11
#define RightFoot 13


int calibratePosition[4] = { 0, 0, 0, 0 };
// Used to save the servo offset, in radians
int YL;
int YR;
int RL;
int RR;

String inputString = "";      // a String to hold incoming data
bool stringComplete = false;  // whether the string is complete

#define COMMANDS_COUNT_MAX 8
char charRead;
String inputStringBLE = "";

int move = 0;

#define ACTION_calibratio 'G'  //Ant movement commands
#define ACTION_flag 'F'        //Expression control commands
#define ACTION_MOVE 'A'

#define INTERVAL_CHAR '#'       //The directive resolves the separator
character
String inputCommandArray[COMMANDS_COUNT_MAX];
int paramters[COMMANDS_COUNT_MAX];


//Parses the serial port data received by the serial port
void Deal_Serial_Data(void) {
  String inputStringTemp = inputStringBLE;
  int string_length = inputStringTemp.length();
  for (int i = 0; i < COMMANDS_COUNT_MAX; i++) {
    int index = inputStringTemp.indexOf(INTERVAL_CHAR);
    if (index < 0) {
      if (string_length > 0) {
        inputCommandArray[i] = inputStringTemp;  //Get command
        paramters[i] = inputStringTemp.toInt();  //Get parameters
      }
      break;
```

```
52      }
53      inputCommandArray[i] = inputStringTemp.substring(0, index);
54      inputStringTemp = inputStringTemp.substring(index + 1);
55      paramters[i] = inputCommandArray[i].toInt();
56    }
57    stringComplete = false;
58    inputStringBLE = "";
59  }
60
61
62  void setup() {
63    EEPROM.begin(512);
64    Serial.begin(9600);
65    // for (int i = 0; i < 4; i++) {
66    //   EEPROM.write(i, calibratePosition[i]);
67    // }
68    Bipedal_Robot.init(LeftLeg, RightLeg, LeftFoot, RightFoot, true);  //Set
69  the servo pins
70    YL = EEPROM.read(0);
71    if (YL > 128) YL -= 256;
72    YR = EEPROM.read(1);
73    if (YR > 128) YR -= 256;
74    RL = EEPROM.read(2);
75    if (RL > 128) RL -= 256;
76    RR = EEPROM.read(3);
77    if (RR > 128) RR -= 256;
78    calib_homePos();
79    Bipedal_Robot.saveTrimsOnEEPROM();
80    EEPROM.commit();
81    Bipedal_Robot.home();
82  }
83
84  void loop() {
85    while (Serial.available()) {
86      inputStringBLE = Serial.readStringUntil('\n');
87      stringComplete = true;
88    }
89    if (stringComplete) {  //Get the serial port command
90      Deal_Serial_Data();
91      char commandChar = inputCommandArray[0].charAt(0);
92      if (commandChar == ACTION_calibratio) {  //Control expression module
93  display
94        // Serial.println("paramters[1]: " + String(paramters[1]) + "
95  paramters[2]: " + String(paramters[2]) + " paramters[3]: " +
```

```
96  String(paramters[3]) + " paramters[4]: " + String(paramters[4]) + "
97  paramters[5]: " + String(paramters[5]));
98        YL = paramters[1] - 90;
99        YR = paramters[2] - 90;
100       RL = paramters[3] - 90;
101       RR = paramters[4] - 90;
102       Bipedal_Robot.setTrims(YL, YR, RL, RR);
103       calib_homePos();
104       if (paramters[5] = 1) {
105         Bipedal_Robot.setTrims(YL, YR, RL, RR);
106         calib_homePos();
107         Bipedal_Robot.saveTrimsOnEEPROM();
108       }
109     }
110     if (commandChar == ACTION_flag) {  //Control expression module display
111       if (paramters[1] = 1) {
112         YL = EEPROM.read(0);
113         if (YL > 128) YL -= 256;
114         YR = EEPROM.read(1);
115         if (YR > 128) YR -= 256;
116         RL = EEPROM.read(2);
117         if (RL > 128) RL -= 256;
118         RR = EEPROM.read(3);
119         if (RR > 128) RR -= 256;
120         YL = YL + 90;
121         YR = YR + 90;
122         RL = RL + 90;
123         RR = RR + 90;
124
125 Serial.print("F#"+String(YL)+"#"+String(YR)+"#"+String(RL)+"#"+String(RR));
126       }
127     }
128 if (commandChar == ACTION_MOVE) {  //Control expression module display
129     move = paramters[1];
130     }
131   }
132   if (move == 1) {
133       Bipedal_Robot.walk(1, 1500, 1);
134     }
135     if (move == 2) {
136       Bipedal_Robot.walk(1, 1500, -1);
137     }
138     if (move == 3) {
139       Bipedal_Robot.turn(1, 1500, 1);
```

```
    }
    if (move == 4) {
      Bipedal_Robot.turn(1, 1500, -1);
    }
    if (move == 0) {
      Bipedal_Robot.home();
    }
}

void calib_homePos() {
  int servoPos[4];
  servoPos[0] = 90;
  servoPos[1] = 90;
  servoPos[2] = 90;
  servoPos[3] = 90;
  Bipedal_Robot._moveServos(500, servoPos);
  Bipedal_Robot.detachServos();
}
```

Parses the serial port data received by the serial port.

```
void Deal_Serial_Data(void) {
  String inputStringTemp = inputStringBLE;
  int string_length = inputStringTemp.length();
  for (int i = 0; i < COMMANDS_COUNT_MAX; i++) {
    int index = inputStringTemp.indexOf(INTERVAL_CHAR);
    if (index < 0) {
      if (string_length > 0) {
        inputCommandArray[i] = inputStringTemp;  //Get command
        paramters[i] = inputStringTemp.toInt();  //Get parameters
      }
      break;
    }
    inputCommandArray[i] = inputStringTemp.substring(0, index);
    inputStringTemp = inputStringTemp.substring(index + 1);
    paramters[i] = inputCommandArray[i].toInt();
  }
  stringComplete = false;
  inputStringBLE = "";
}
```

Execute the corresponding task according to the received serial port command.

```
    if (commandChar == ACTION_calibratio) {  //Control expression module
display
    if (commandChar == ACTION_flag) {  //Control expression module display
if (commandChar == ACTION_MOVE) {  //Control expression module display
```

01.1.3_Servo_Calibration_Bluetooth

Upload this sketch to the Raspberry Pi Pico (W). This sketch corresponds to the way of calibration via phone app. You can calibrate the servo angle by clicking the corresponding servo button. After the calibration is completed, click the Save button. When testing the robot's movement, if it does not move well, you can readjust each servo until the robot moves normally.

Code

```
#include <Arduino.h>
#include <Wire.h>
#include <SoftwareSerial.h>
#include <EEPROM.h>
#include "SerialCommand.h"
#include "Bipedal_Robot.h"

#define LeftLeg 10
#define RightLeg 12
#define LeftFoot 11
#define RightFoot 13

int calibratePosition[4] = { 0, 0, 0, 0 };
// Used to save the servo offset, in radians
int YL;
int YR;
int RL;
int RR;

String inputString = "";      // a String to hold incoming data
bool stringComplete = false;  // whether the string is complete

#define COMMANDS_COUNT_MAX 8
char charRead;
String inputStringBLE = "";

#define ACTION_calibratio 'G'  //Ant movement commands
#define ACTION_flag 'F'        //Expression control commands
#define ACTION_MOVE 'A'

#define INTERVAL_CHAR '#'  //The directive resolves the separator
character
String inputCommandArray[COMMANDS_COUNT_MAX];
int paramters[COMMANDS_COUNT_MAX];
int move = 0;
int movespeed = 1500;
```

```
37
38  SoftwareSerial BTserial = SoftwareSerial(1, 0);  // RX pin to 0 and TX pin
39  to 1 on the board
40  SerialCommand SCmd(BTserial);
41  Bipedal_Robot Bipedal_Robot;
42
43  void receiveStop() {
44    delay(10);
45    Serial.flush();
46  }
47
48  //Parses the Bluetooth data received by the serial port
49  void Deal_Serial_Data(void) {
50    String inputStringTemp = inputStringBLE;
51    int string_length = inputStringTemp.length();
52    for (int i = 0; i < COMMANDS_COUNT_MAX; i++) {
53      int index = inputStringTemp.indexOf(INTERVAL_CHAR);
54      if (index < 0) {
55        if (string_length > 0) {
56          inputCommandArray[i] = inputStringTemp;  //Get command
57          paramters[i] = inputStringTemp.toInt();  //Get parameters
58        }
59        break;
60      }
61      inputCommandArray[i] = inputStringTemp.substring(0, index);
62      inputStringTemp = inputStringTemp.substring(index + 1);
63      paramters[i] = inputCommandArray[i].toInt();
64    }
65    stringComplete = false;
66    inputStringBLE = "";
67  }
68
69  void setup() {
70    EEPROM.begin(512);
71    Serial.begin(115200);
72    BTserial.begin(115200);
73    delay(2000);
74    BTserial.println("AT+UART?\r\n");
75    delay(200);
76    BTserial.println("AT+UART=115200\r\n");
77    delay(200);
78    BTserial.println("AT+NAME=BT05\r\n");  //Set the radio called Robot
79    delay(200);
80
```

```
81    SCmd.addDefaultHandler(receiveStop);
82    inputString.reserve(200);
83    Serial.println("Set the name of the Bluetooth module to BT05.");
84    Serial.println("Set the Bluetooth baud rate to 115200.");
85
86    // for (int i = 0; i < 4; i++) {
87    //   EEPROM.write(i, calibratePosition[i]);
88    // }
89    Bipedal_Robot.init(LeftLeg, RightLeg, LeftFoot, RightFoot, true);  //Set
90  the servo pins
91    YL = EEPROM.read(0);
92    if (YL > 128) YL -= 256;
93    YR = EEPROM.read(1);
94    if (YR > 128) YR -= 256;
95    RL = EEPROM.read(2);
96    if (RL > 128) RL -= 256;
97    RR = EEPROM.read(3);
98    if (RR > 128) RR -= 256;
99    calib_homePos();
100   Bipedal_Robot.saveTrimsOnEEPROM();
101   EEPROM.commit();
102   Bipedal_Robot.home();
103   Serial.println("start...");
104 }
105
106 void readservo(int parameter1, int parameter2, int parameter3, int
107 parameter4) {
108   parameter1 = EEPROM.read(0);
109   parameter2 = EEPROM.read(1);
110   parameter3 = EEPROM.read(2);
111   parameter4 = EEPROM.read(3);
112 }
113
114 void loop() {
115   if (BTserial.available()) {
116     inputStringBLE = BTserial.readStringUntil('\n');
117     Serial.println(inputStringBLE);
118     stringComplete = true;
119   }
120   if (stringComplete) {  //Get the Bluetooth command
121     Deal_Serial_Data();
122     char commandChar = inputCommandArray[0].charAt(0);
123     if (commandChar == ACTION_calibratio) {  //Control expression module
124 display
```

```
      Serial.println("paramters[1]: " + String(paramters[1]) + "
paramters[2]: " + String(paramters[2]) + " paramters[3]: " +
String(paramters[3]) + " paramters[4]: " + String(paramters[4]) + "
paramters[5]: " + String(paramters[5]));
      YL = paramters[1] - 90;
      YR = paramters[2] - 90;
      RL = paramters[3] - 90;
      RR = paramters[4] - 90;
      Bipedal_Robot.setTrims(YL, YR, RL, RR);
      calib_homePos();
      if (paramters[5] = 1) {
        Bipedal_Robot.setTrims(YL, YR, RL, RR);
        calib_homePos();
        Bipedal_Robot.saveTrimsOnEEPROM();
      }
    }
    if (commandChar == ACTION_flag) {  //Control expression module display
      if (paramters[1] = 1) {
        YL = EEPROM.read(0);
        if (YL > 128) YL -= 256;
        YR = EEPROM.read(1);
        if (YR > 128) YR -= 256;
        RL = EEPROM.read(2);
        if (RL > 128) RL -= 256;
        RR = EEPROM.read(3);
        if (RR > 128) RR -= 256;
        YL = YL + 90;
        YR = YR + 90;
        RL = RL + 90;
        RR = RR + 90;
        Serial.println("YL :" + String(YL) + " YR : " + String(YR) + "
RL : " + String(RL) + " RR : " + String(RR));
        BTserial.println("F#" + String(YL) + "#" + String(YR) + "#" +
String(RL) + "#" + String(RR));
      }
    }
    if (commandChar == ACTION_MOVE) {  //Control expression module display
      Serial.println("paramters[1] :" + String(paramters[1]));
      Serial.println("paramters[2] :" + String(paramters[2]));
      move = paramters[1];
      movespeed = paramters[2];
      if (movespeed == 0) {
        movespeed = 1000;
      } else if (movespeed == 1) {
```

```
169           movespeed = 1500;
170         } else if (movespeed == 2) {
171           movespeed = 2000;
172         } else
173           movespeed = 1500;
174       }
175     }
176     if (move == 1) {
177       Serial.println("move :" + String(move));
178       Bipedal_Robot.walk(1, movespeed, 1);
179     }
180     if (move == 2) {
181       Serial.println("move :" + String(move));
182       Bipedal_Robot.walk(1, movespeed, -1);
183     }
184     if (move == 3) {
185       Serial.println("move :" + String(move));
186       Bipedal_Robot.turn(1, movespeed, 1);
187     }
188     if (move == 4) {
189       Serial.println("move :" + String(move));
190       Bipedal_Robot.turn(1, movespeed, -1);
191     }
192     if (move == 0) {
193       Bipedal_Robot.home();
194     }
195 }
196
197 void calib_homePos() {
198   int servoPos[4];
199   servoPos[0] = 90;
200   servoPos[1] = 90;
201   servoPos[2] = 90;
202   servoPos[3] = 90;
203   Bipedal_Robot._moveServos(10, servoPos);
204   Bipedal_Robot.detachServos();
205 }
```

Initialize the Bluetooth module, set the baud rate of the Bluetooth module and the Bluetooth name.

```
    delay(2000);
    BTserial.println("AT+UART?\r\n");
    delay(200);
    BTserial.println("AT+UART=115200\r\n");
    delay(200);
```

```
  BTserial.println("AT+NAME=BT05\r\n");  //Set the radio called Robot
  delay(200);
```

Similar to serial communication, according to the received Bluetooth data, the corresponding function is executed.

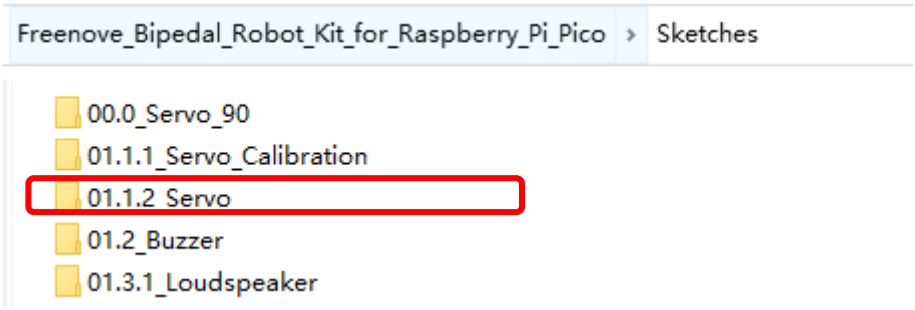
```
  if (BTserial.available()) {
    inputStringBLE = BTserial.readStringUntil('\n');
    Serial.println(inputStringBLE);
    stringComplete = true;
  }
  if (stringComplete) {  //Get the Bluetooth command
    Deal_Serial_Data();
    char commandChar = inputCommandArray[0].charAt(0);
    if (commandChar == ACTION_calibratio) {  //Control expression module
display
      Serial.println("paramters[1]: " + String(paramters[1]) + "
paramters[2]: " + String(paramters[2]) + " paramters[3]: " +
String(paramters[3]) + " paramters[4]: " + String(paramters[4]) + "
paramters[5]: " + String(paramters[5]));
      YL = paramters[1] - 90;
      YR = paramters[2] - 90;
      RL = paramters[3] - 90;
      RR = paramters[4] - 90;
      Bipedal_Robot.setTrims(YL, YR, RL, RR);
      calib_homePos();
      if (paramters[5] = 1) {
        Bipedal_Robot.setTrims(YL, YR, RL, RR);
        calib_homePos();
        Bipedal_Robot.saveTrimsOnEEPROM();
      }
    }
    if (commandChar == ACTION_flag) {  //Control expression module display
      if (paramters[1] = 1) {
        YL = EEPROM.read(0);
        if (YL > 128) YL -= 256;
        YR = EEPROM.read(1);
        if (YR > 128) YR -= 256;
        RL = EEPROM.read(2);
        if (RL > 128) RL -= 256;
        RR = EEPROM.read(3);
        if (RR > 128) RR -= 256;
        YL = YL + 90;
        YR = YR + 90;
        RL = RL + 90;
        RR = RR + 90;
```

```
        Serial.println("YL :" + String(YL) + " YR : " + String(YR) + "
RL : " + String(RL) + " RR : " + String(RR));
        BTserial.println("F#" + String(YL) + "#" + String(YR) + "#" +
String(RL) + "#" + String(RR));
      }
    }
    if (commandChar == ACTION_MOVE) {  //Control expression module display
      Serial.println("paramters[1] :" + String(paramters[1]));
      Serial.println("paramters[2] :" + String(paramters[2]));
      move = paramters[1];
      movespeed = paramters[2];
      if (movespeed == 0) {
        movespeed = 1000;
      } else if (movespeed == 1) {
        movespeed = 1500;
      } else if (movespeed == 2) {
        movespeed = 2000;
      } else
        movespeed = 1500;
    }
  }
```

Sketch

If the robot has passed the tests of moving forward, backward, left, and right in the calibration section, you can skip this test.

Open "01.1.4_Servo" in **"Freenove_Bipedal_Robot_Kit_for_Raspberry_Pi_Pico\Sketches"** and double-click "01.1.4_Servo.ino".



Please check whether the calibration is completed. If the calibration operation is not performed, please carry out the corresponding calibration steps and upload the code.
Upload the code to Raspberry Pi Pico (W). After the code uploads successfully, the robot will walk forward, backward, turn left and turn right, which repeats in a cycle.

Code

```
#include <Arduino.h>
#include <EEPROM.h>
#include "Bipedal_Robot.h"
Bipedal_Robot Bipedal_Robot;

#define LeftLeg 10
#define RightLeg 12
#define LeftFoot 11
#define RightFoot 13

int calibratePosition[4] = { -17, -10, -15, 12 };

void setup() {
  Serial.begin(115200);
  EEPROM.begin(512);
  Bipedal_Robot.init(LeftLeg, RightLeg, LeftFoot, RightFoot, true);  //Set the servo pins
  // Bipedal_Robot.setTrims(calibratePosition[0], calibratePosition[1], calibratePosition[2], calibratePosition[3]);
  calib_homePos();
  Bipedal_Robot.saveTrimsOnEEPROM();
  EEPROM.commit();
```

```
21    Bipedal_Robot.home();
22    delay(50);
23  }
24  void loop() {
25    Serial.println("loop.....");
26    Bipedal_Robot.walk(2,2000,1);
27    Bipedal_Robot.home();
28    delay(1000);
29    Bipedal_Robot.walk(2,2000,-1);
30    Bipedal_Robot.home();
31    delay(1000);
32    Bipedal_Robot.turn(2,2000,1);
33    Bipedal_Robot.home();
34    delay(1000);
35    Bipedal_Robot.turn(2,2000,-1);
36    Bipedal_Robot.home();
37    delay(1000);
38  }
39  void calib_homePos() {
40    int servoPos[4];
41    servoPos[0] = 90;
42    servoPos[1] = 90;
43    servoPos[2] = 90;
44    servoPos[3] = 90;
45    Bipedal_Robot._moveServos(500, servoPos);
46    Bipedal_Robot.detachServos();
47  }
```

Code Explanation

In tha main loop, the robot moves forward and backword, turn left, and turn right repeatedly.

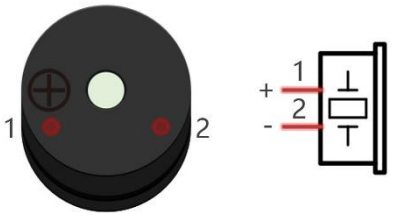
```
    Serial.println("loop.....");
    Bipedal_Robot.walk(2,2000,1);
    Bipedal_Robot.home();
    delay(1000);
    Bipedal_Robot.walk(2,2000,-1);
    Bipedal_Robot.home();
    delay(1000);
    Bipedal_Robot.turn(2,2000,1);
    Bipedal_Robot.home();
    delay(1000);
    Bipedal_Robot.turn(2,2000,-1);
    Bipedal_Robot.home();
    delay(1000);
```

1.2 Buzzer

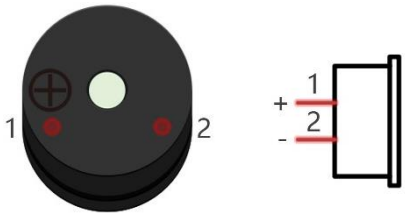
Buzzer

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.

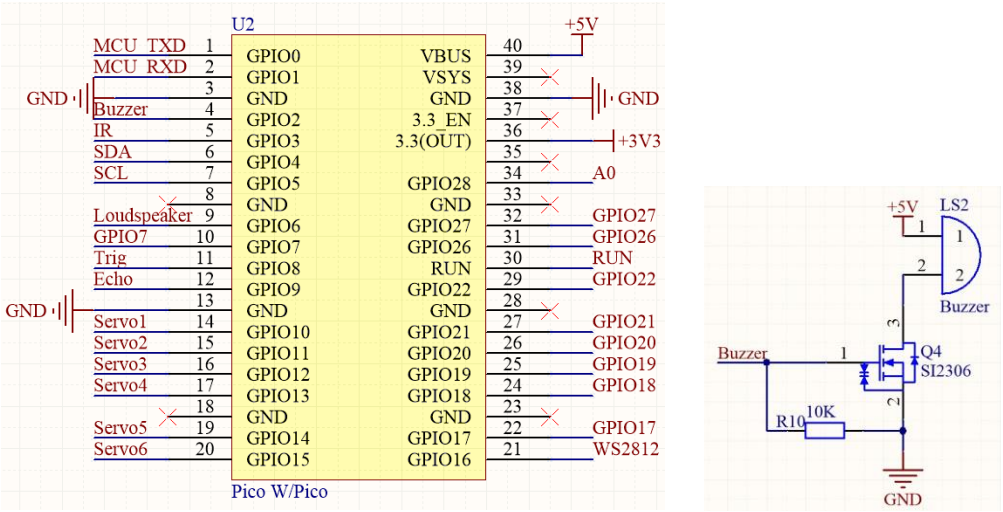
Active buzzer

Passive buzzer

The buzzer used in this robot 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.



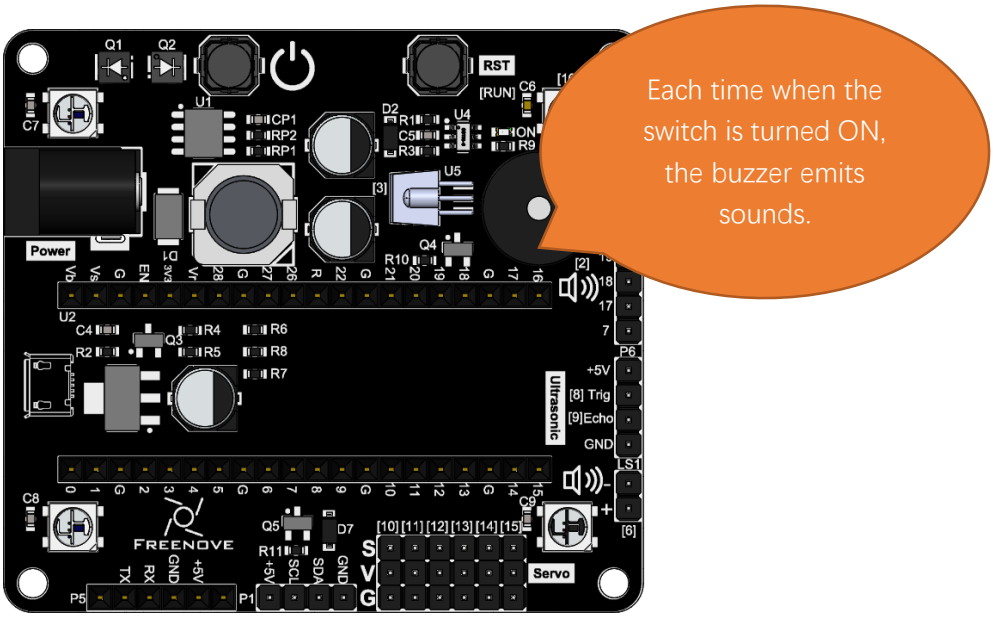
Sketch

In this section, we will test the buzzer to make it sound like an alarm.
Open "01.2_Buzzer" folder in the **"Freenove_Bipedal_Robot_Kit_for_Raspberry_Pi_Pico\Sketches"**, and then double-click "01.2_Buzzer.ino".

Code

```
#include "Freenove_Robot_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
```

1.3 Loudspeaker

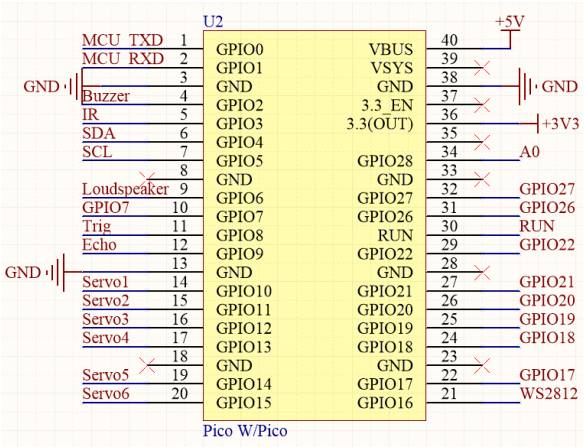
Loudspeaker

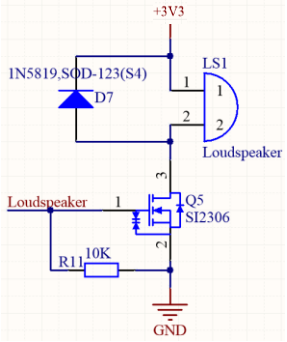
Besides the buzzer, we still have a loud speker as sound device, which is directly controlled by the Raspberry Pi Pico. However, please note that its sound quality is not so good. If you want better sound quality, you can connect audio conveter module to spare GPIO pins for extended use.

In this project, we use the speaker to play a piect of music. If you just use the speker to make simple sounds, you can refer to the example of the buzzer and modify the pins accordingly.

Schematic

As can be seen, the robot plays audio via GPIO6 of Raspberry Pi Pico (W).

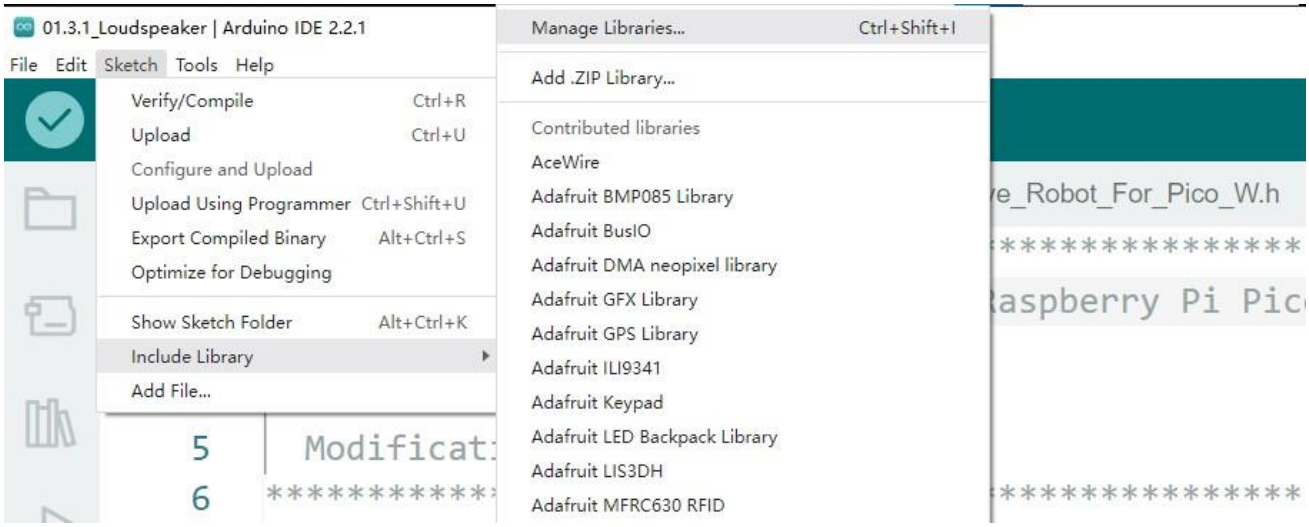




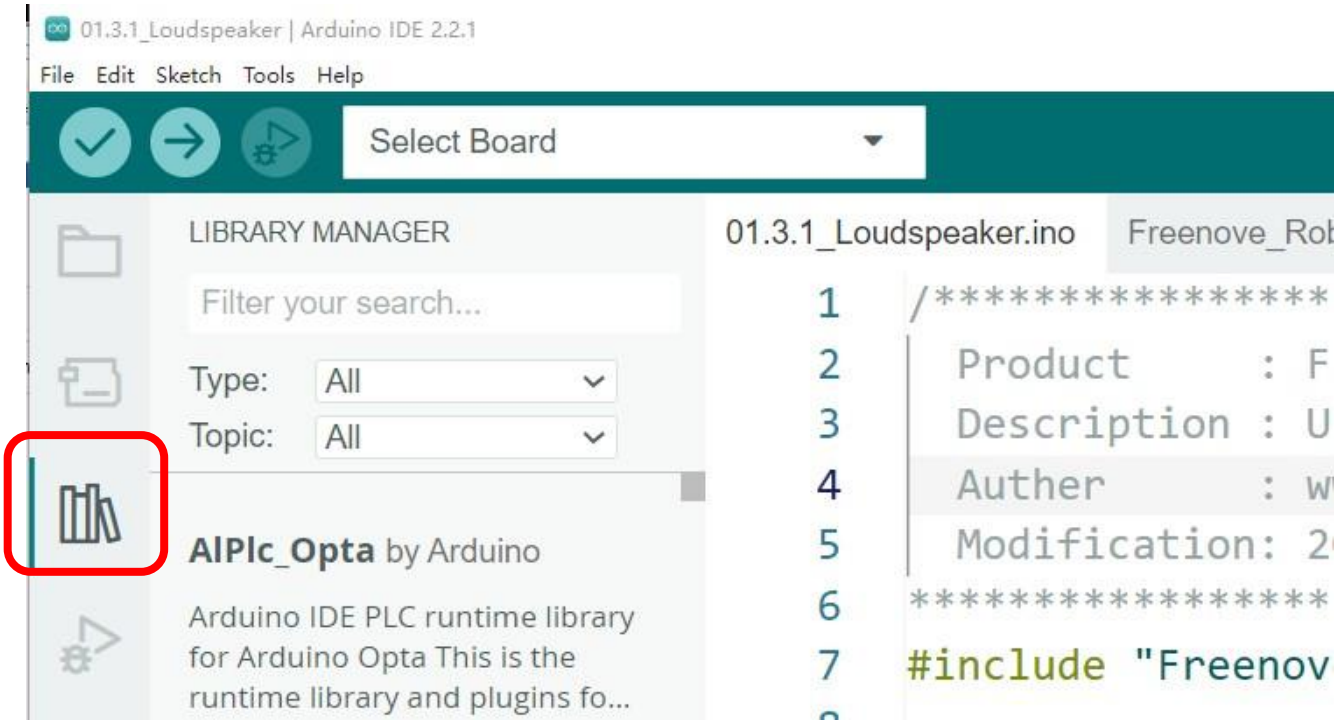
Add libraries

Method 1

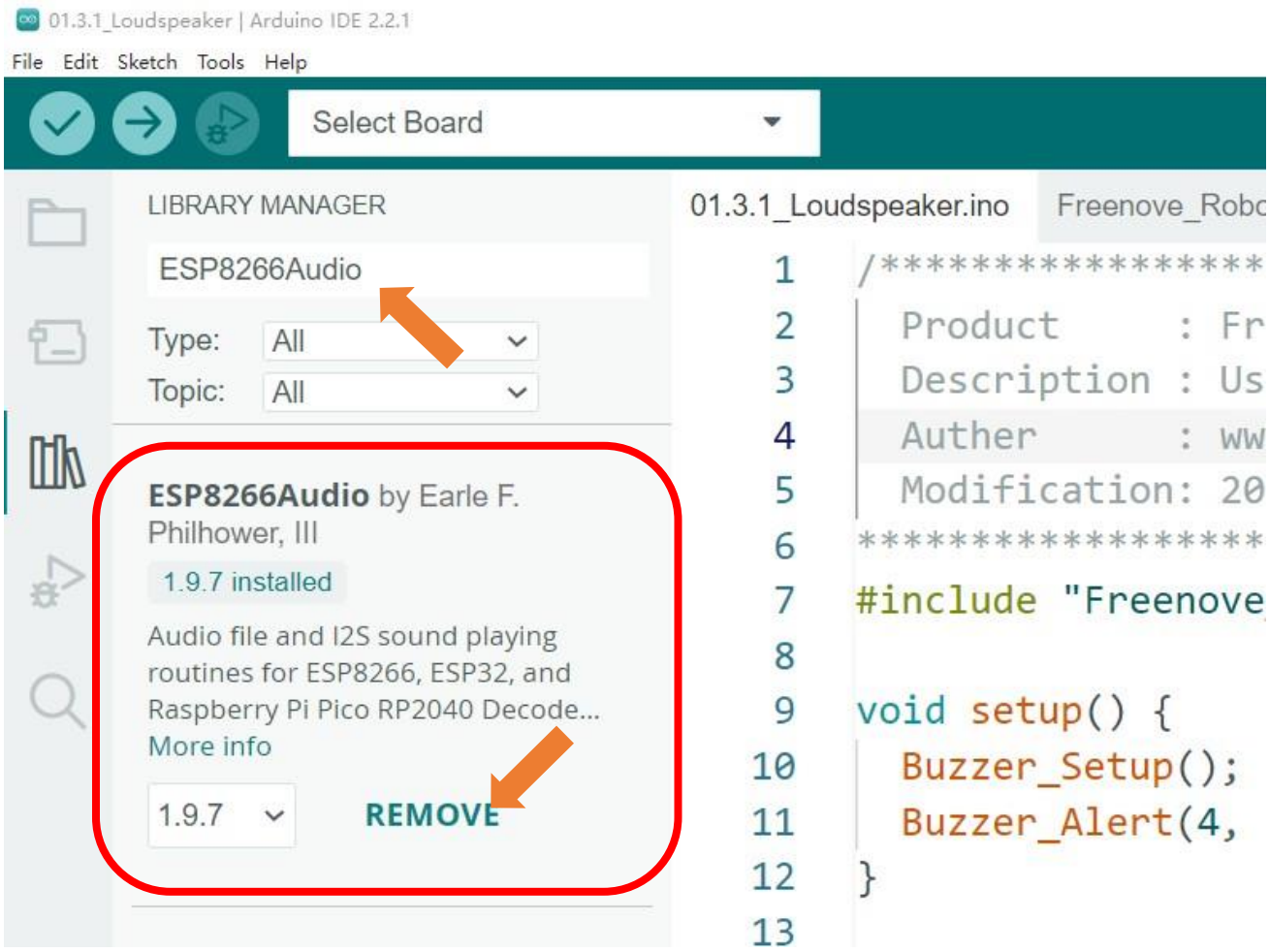
Open Arduino IDE, click Sketch on Menu bar ->Include Library -> Manage Libraries.



Or click the shortcut icon on the left.



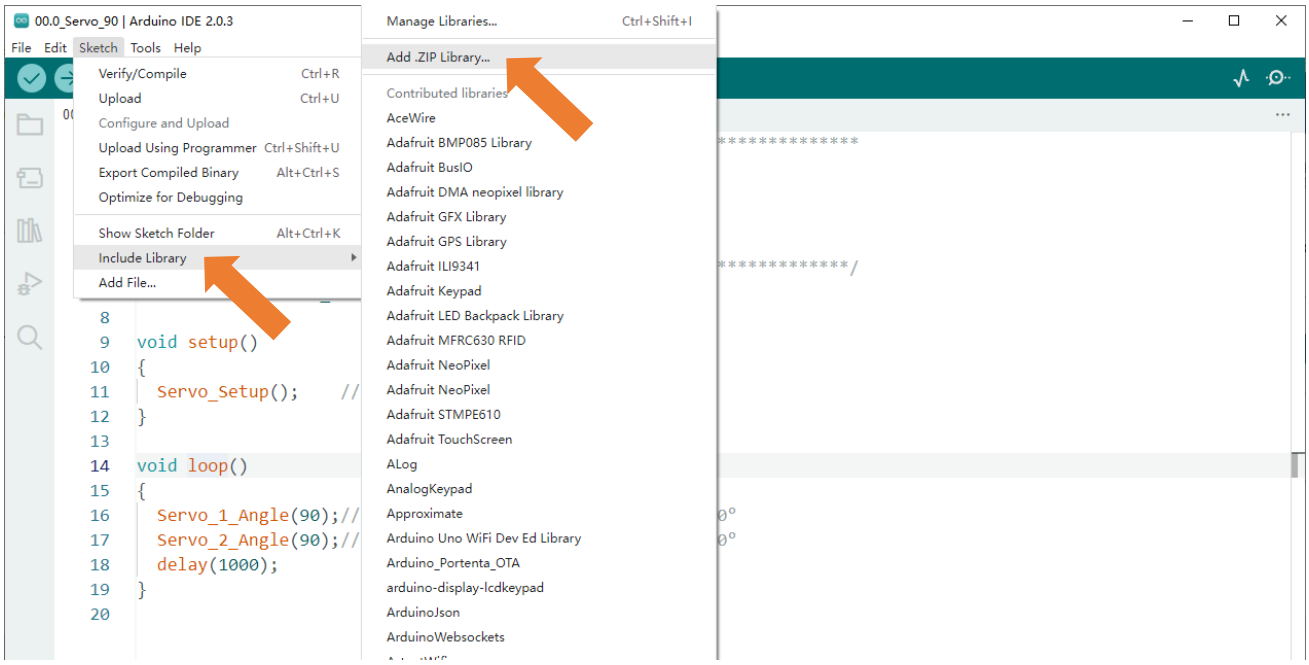
There is an input field on the right top of the pop-up window. Enter ESP8266Audio there and click to install the library boxed in the following figure.



Wait for the installation to finish.

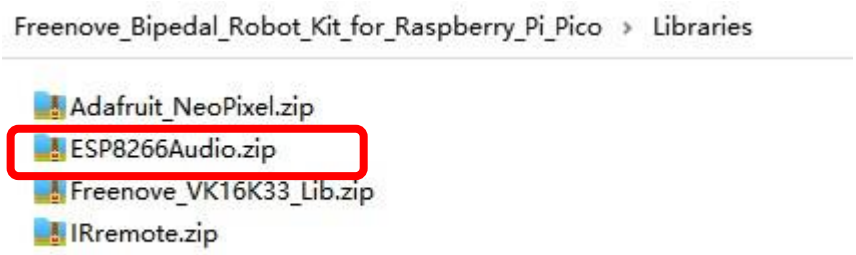
Method 2

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



On the pop-up window, select ESP8266Audio.zip in Libraries folder under

"Freenove_Bipedal_Robot_Kit_for_Raspberry_Pi_Pico\Libraries", and then click Open.



Sketch

01.3.1_Loudspeaker

In this section, we use GPIO6 of the Raspberry Pi Pico (W). This example is the same as the buzzer example, so the result is the same.

Open "01.3.1_Loudspeaker" folder in **"Freenove_Bipedal_Robot_Kit_for_Raspberry_Pi_Pico\Sketches"** and then double-click "01.3.1_Loudspeaker.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);
}
```

Code Explanation

This code is the same as the buzzer code, except for the different GPIO used. In the buzzer example, we use GPIO 2 and in this one, we use GPIO 6.

```
  #define PIN_BUZZER 6
```

01.3.2_Music

In this section, we will use GPIO6t of the Raspberry Pi Pico to play audio. After the code upload successfully, it will play a piece of music.

Open "01.3.2_Music" folder in **"Freenove_Bipedal_Robot_Kit_for_Raspberry_Pi_Pico\Sketches"** and then double-click "01.3.2_Music.ino".

Install the PicoLittleFS tool

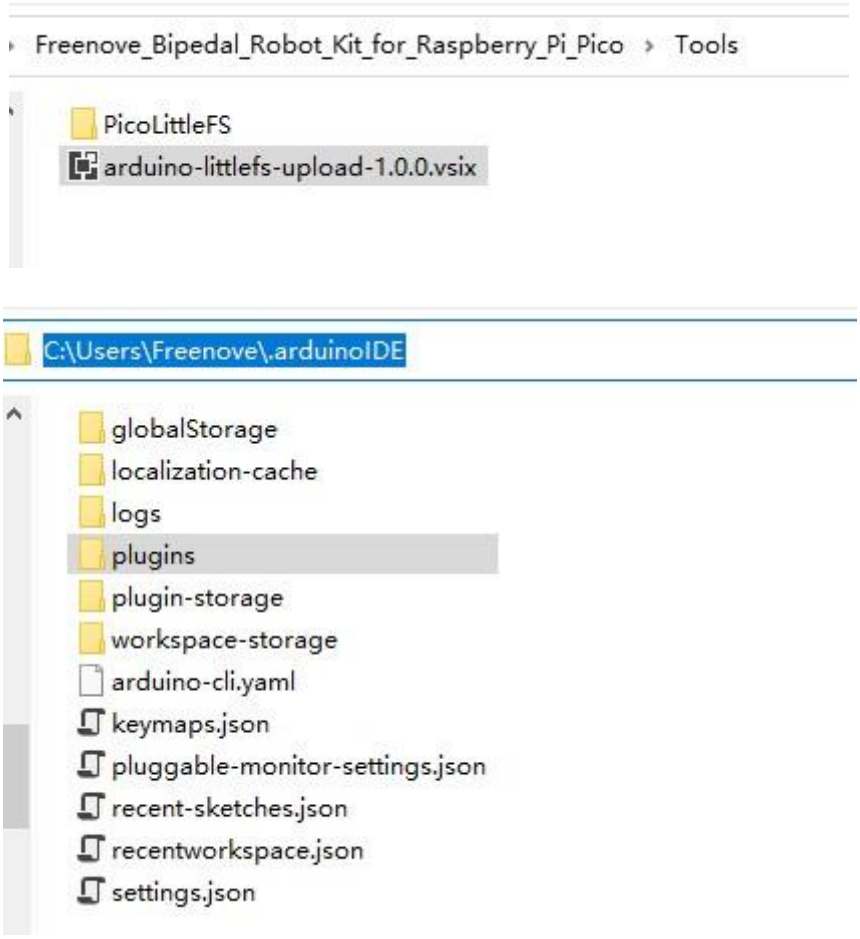
The latest Arduino IDE has supported LittleFS plugins.

The steps to install PicoLittleFS is as follows:

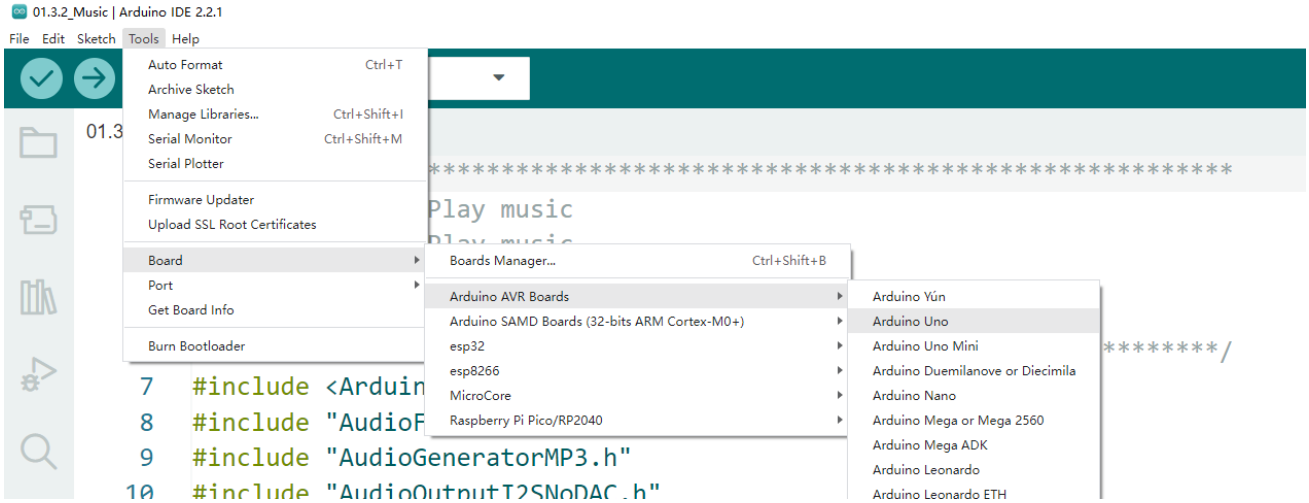
Copy the file **arduino-littlefs-upload-1.0.0.vsix** under the directory **"Freenove_Bipedal_Robot_Kit_for_Raspberry_Pi_Pico\Tools"** to the computer's directory **~/.arduinoIDE/plugins/**

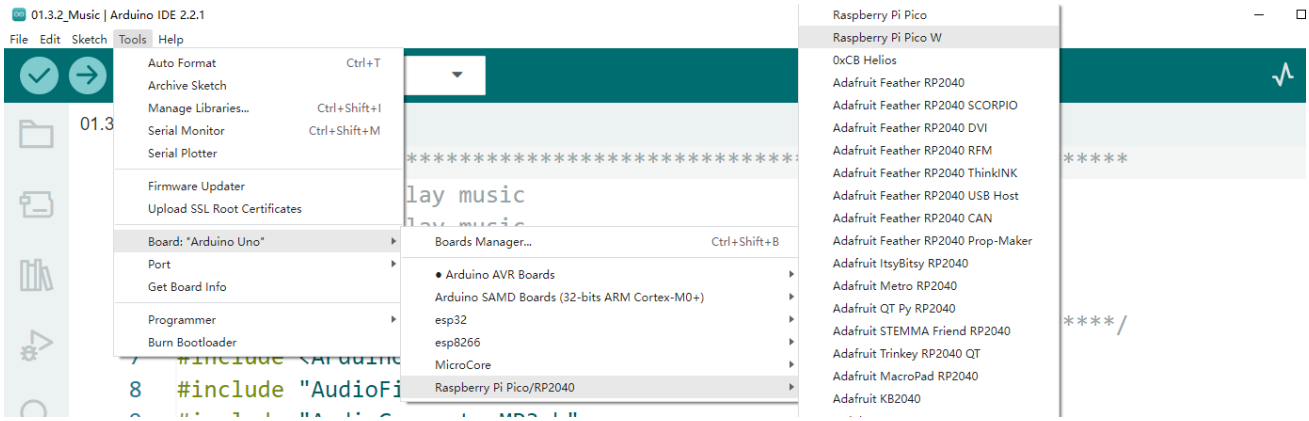
If there is no a folder name plugins in your computer, please create the folder before copying the file.

After copying the file, restart Arduino IDE.

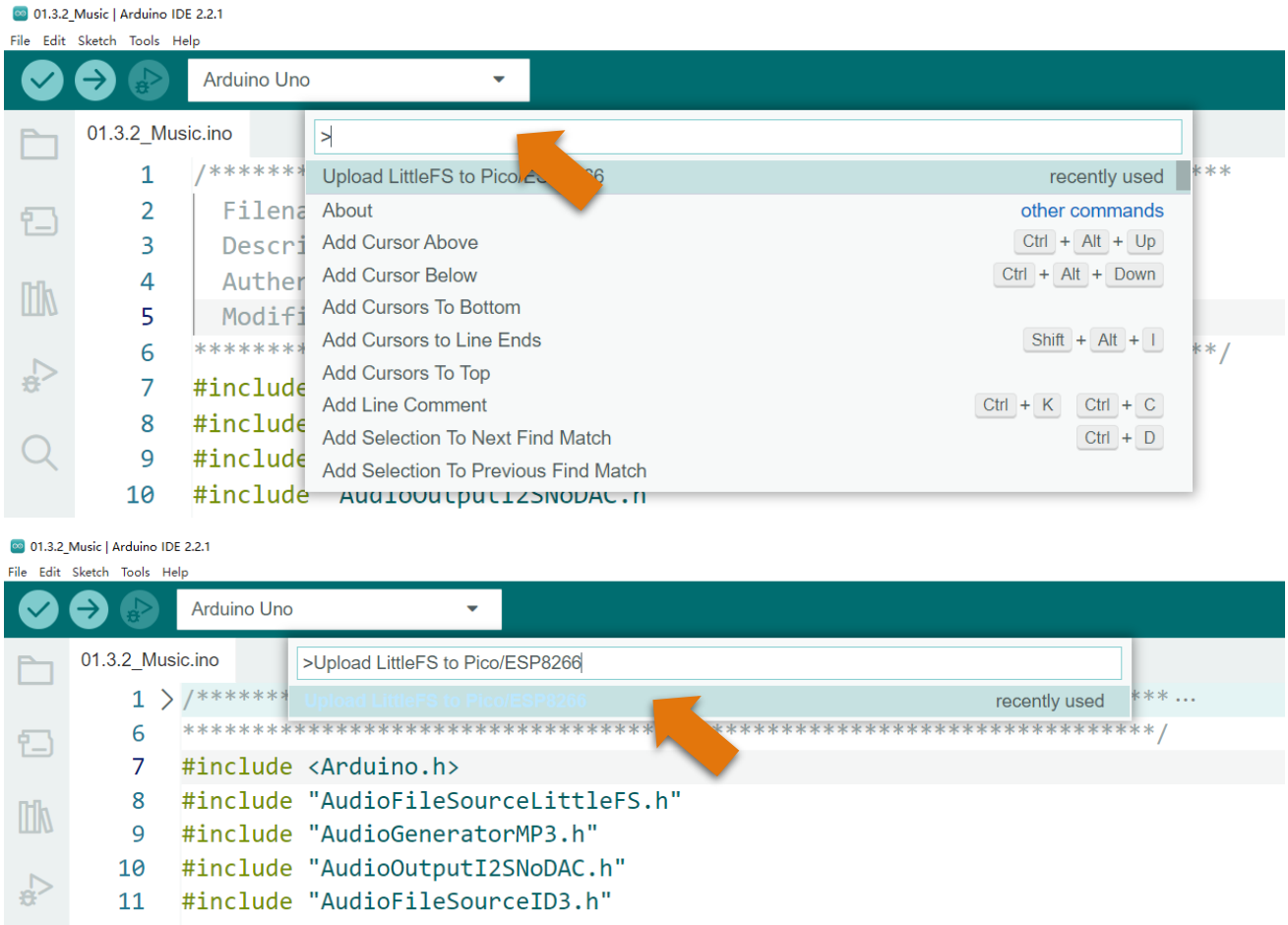


The sketch opened with the start of the Arduino IDE may be corrupted, which can lead to code uploading failure. To address this issue, you can change the board selection (switching to any board), and then select back the pico board. Alternatively, you can close the opened sketch and reopen it.





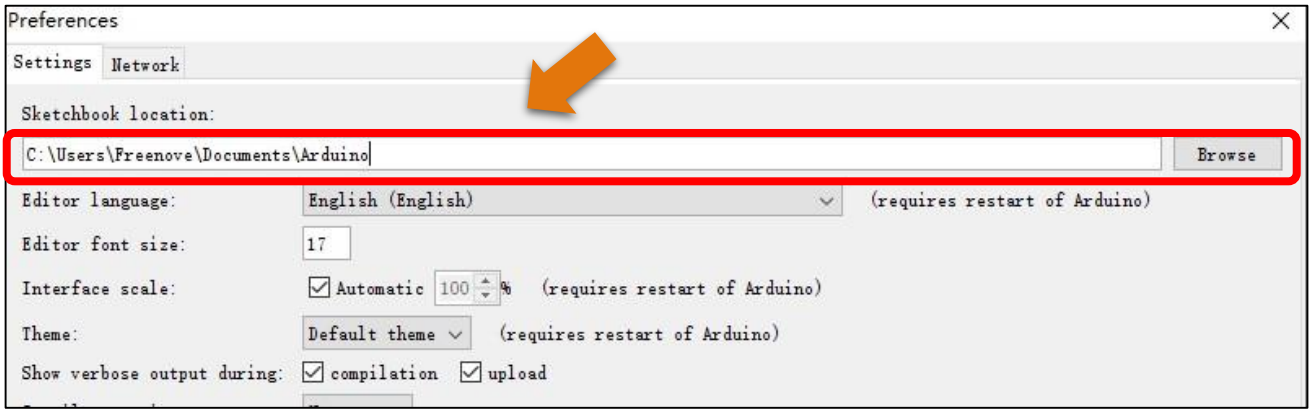
Here is how to use the PicoLittleFS tool, press [Ctrl]+[Shift]+[P] simultaneously, enter ***Upload LittleFS to Pico/ESP8266*** on the input field.



If your Arduino IDE is an old version one, like Arduino 1.x.x, please refer to the following steps to install the PicoLittleIFS tool.

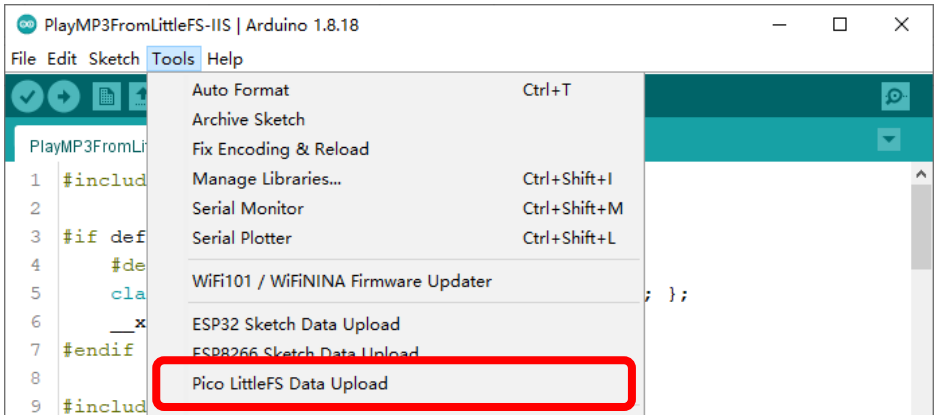
First, open the Arduino IDE, and then click File in Menus and select Preferences.

Find the Arduino IDE environment directory location.



Copy the tools folder in the code folder to your Sketchbook location.

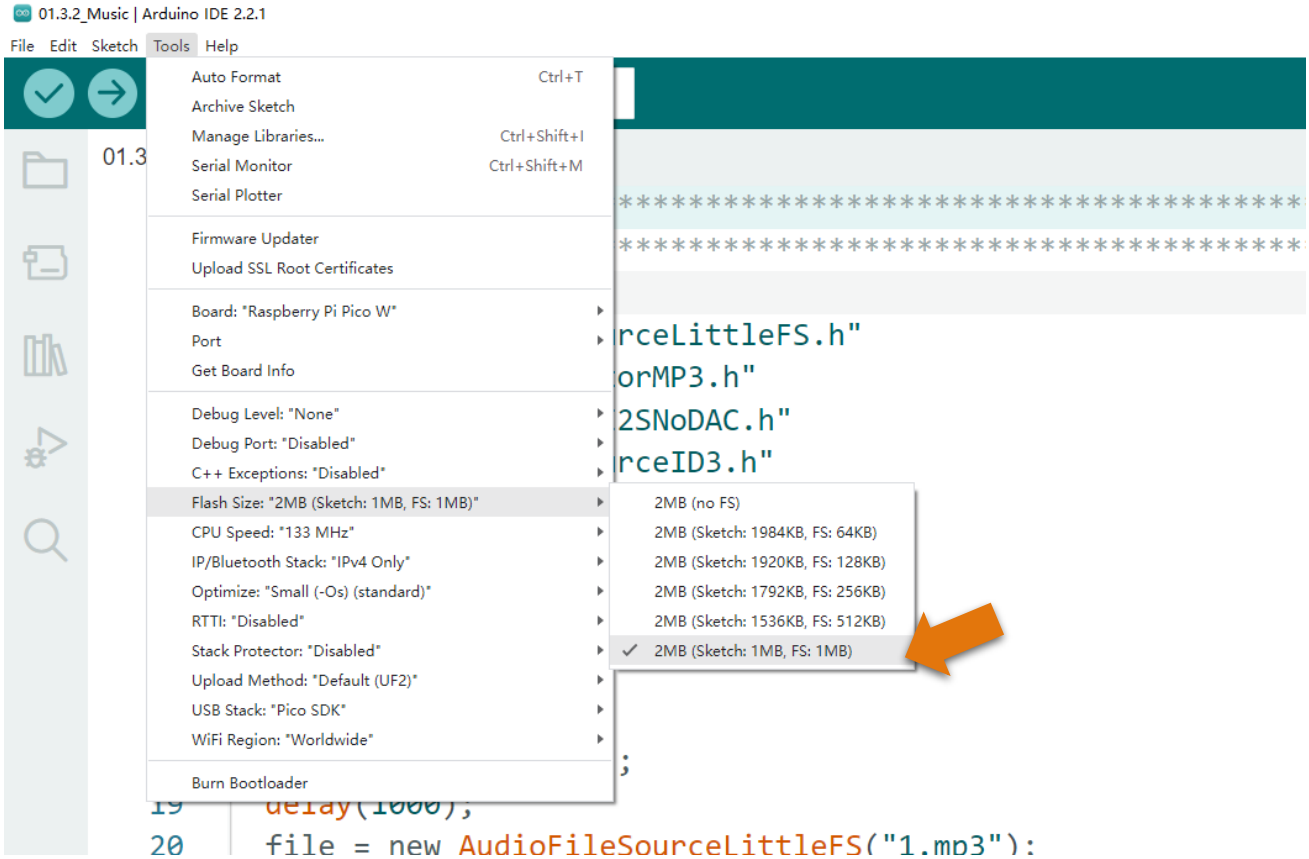
Finally, restart the Arduino IED. After restarting, you can see that the plug-in already exists in the interface.



Upload music

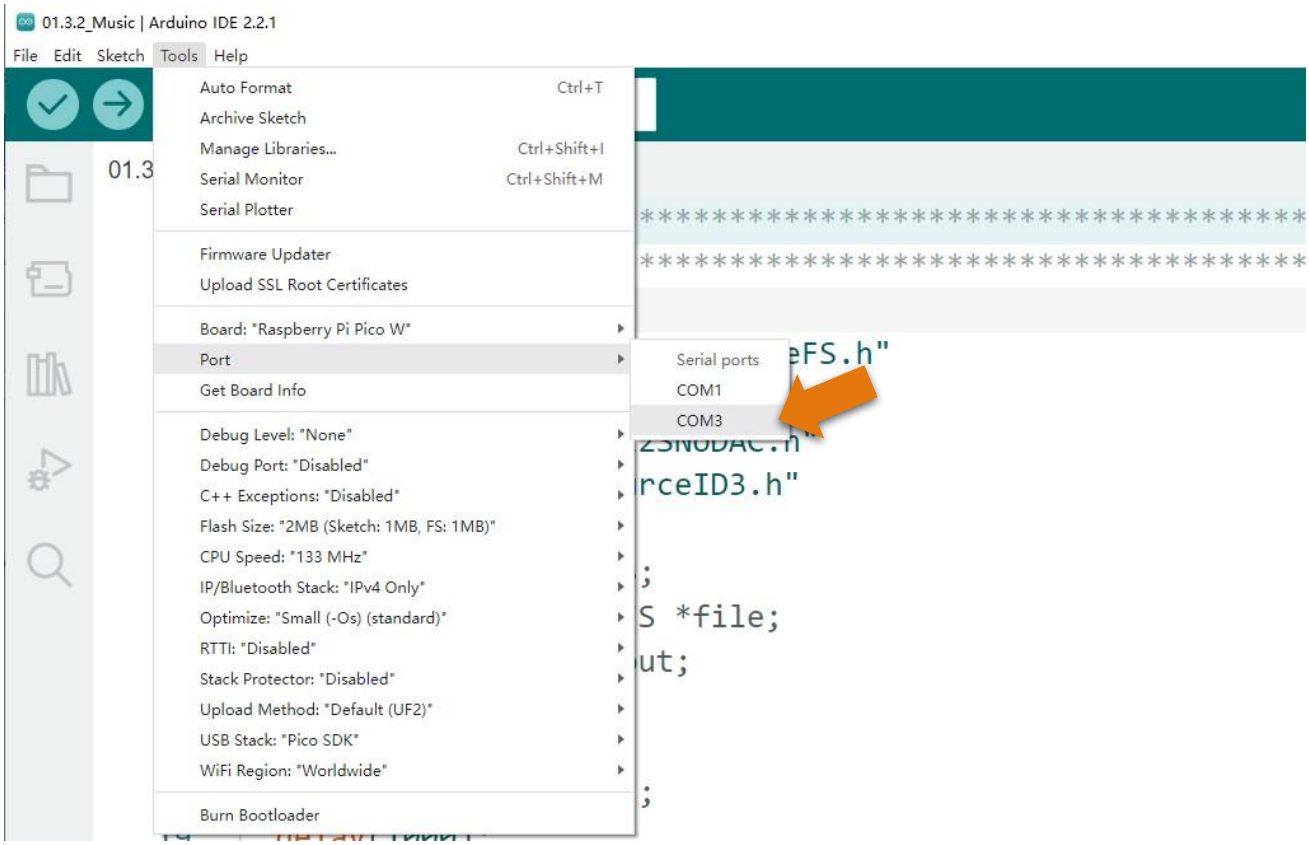
Pico of Raspberry Pie has 2M Flash space. Generally, Arduino mode allocates it to the code area. Therefore, before starting, we need to modify the configuration of Flash Size.

Open Arduino, select Tools from the menu bar, select Flash Size, and allocate 1MB of Flash space to store codes and 1MB to store audio files.

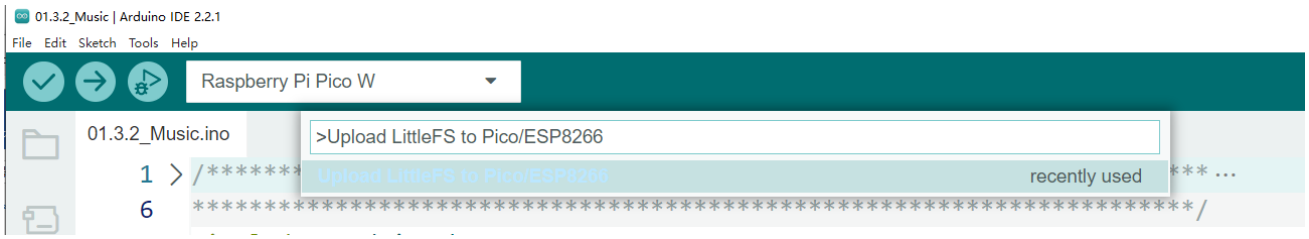


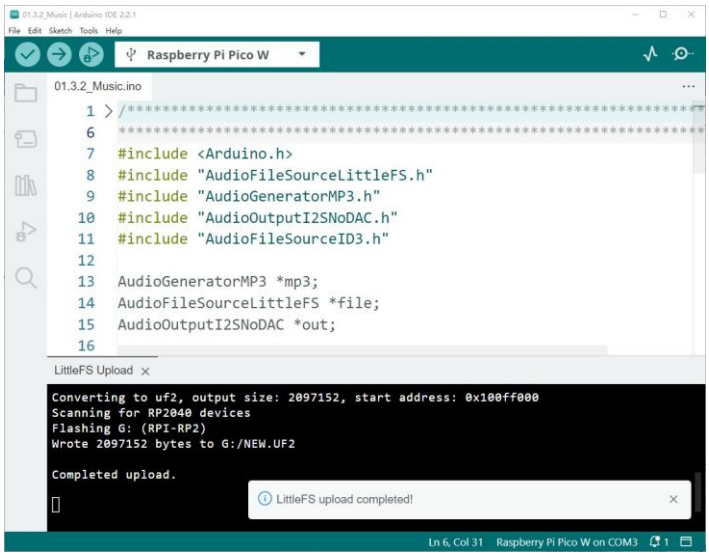
Make sure the CPU Speed of Raspberry Pie is 133Mhz. If the frequency is too low, the audio decoding speed may be too slow and the audio playback may not be continuous.

Select the correct port.

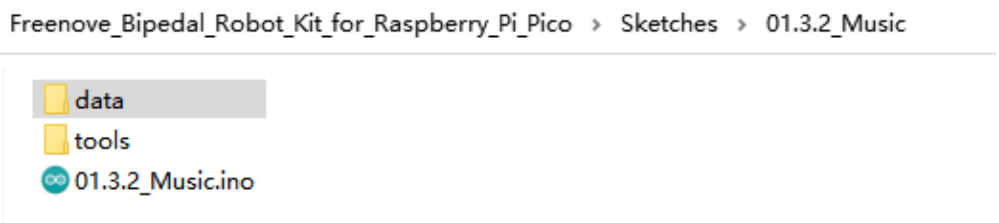


Press [Ctrl]+[Shift]+[P] simultaneously. Enter ***Upload LittleFS to Pico/ESP8266*** on the input field, click Upload LittleFS to Pico/ESP8266, and wait for it to finish.



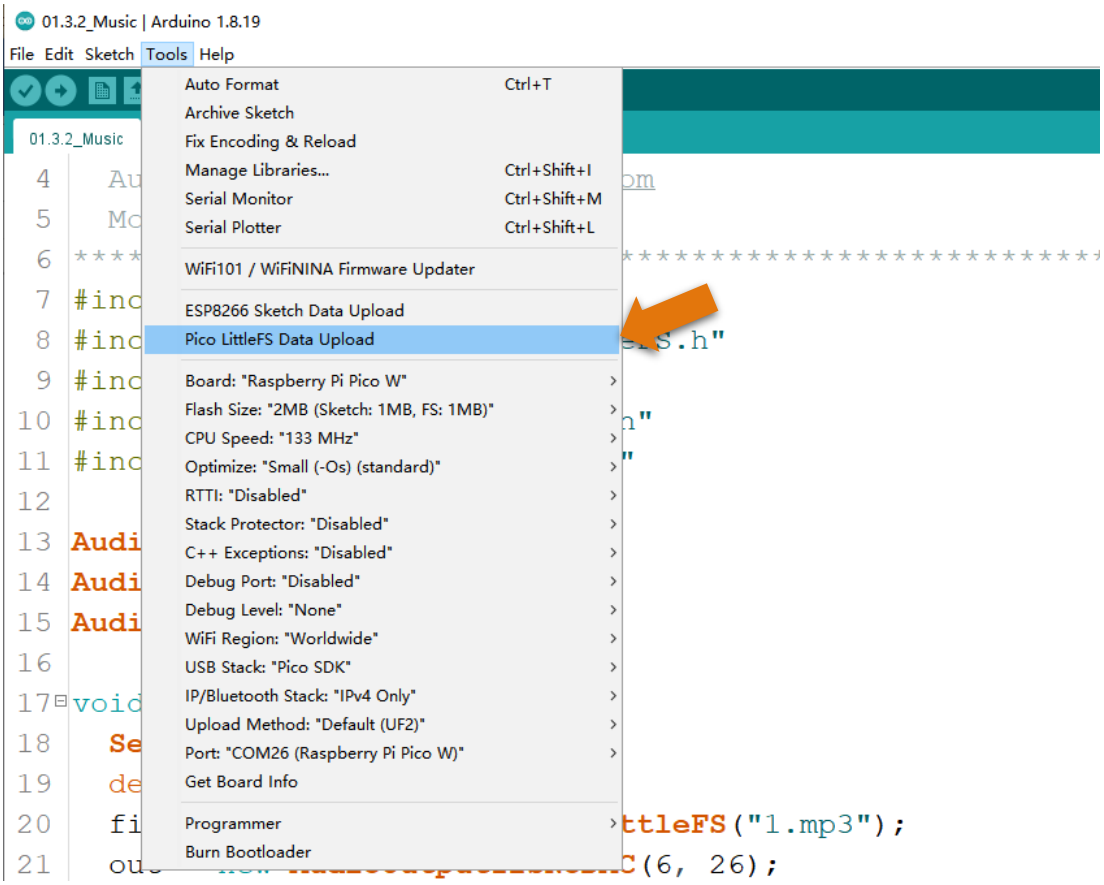


Check the code file and audio file. We create a folder named data under the same level directory of the code file, and place the audio file directly in this folder.



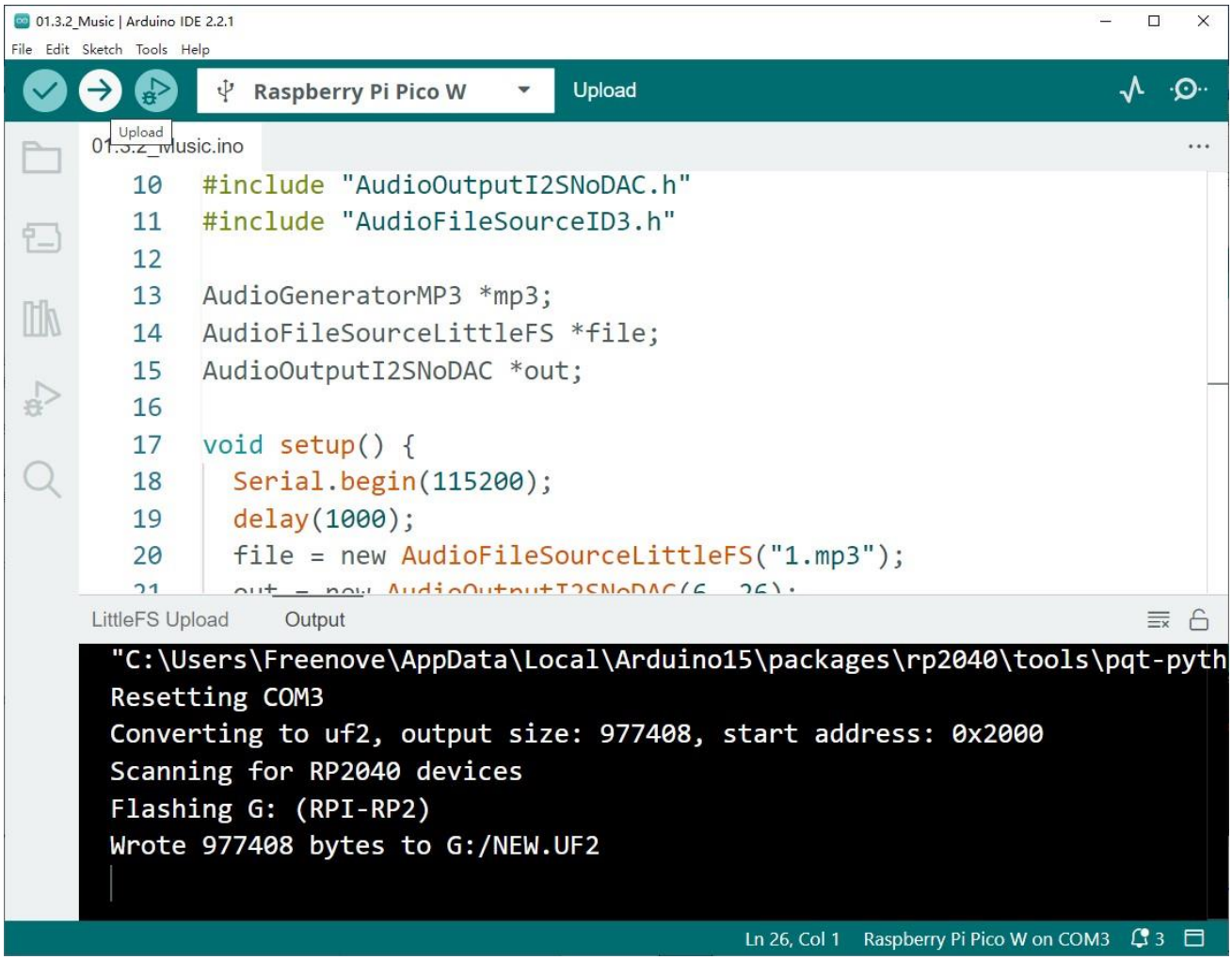
Note: 1. The name of the data folder cannot be changed, otherwise the plug-in cannot be used to upload audio files to Pico.
2. The number of audio files in the data folder is unlimited, but the total size cannot exceed 1MB. If the file upload fails, please check whether the data folder size exceeds the range.

If your Arduino IDE is an old version one, like Arduino 1.x.x, please upload the music as below:
Click Arduino IDE Tools and click following content:



```
LittleFS Image Uploaded
[LittleFS] size    : 1024KB
[LittleFS] page    : 256
[LittleFS] block   : 4096
/1.mp3
/Hello.mp3
[LittleFS] upload   : C:\Users\Freenove\AppData\Local\Temp\arduino_build_527188/01.3.2_Music.mk
[LittleFS] address  : 269479936
[LittleFS] swerial  : COM26
[LittleFS] python   : C:\Users\Freenove\AppData\Local\Arduino15\packages\rp2040\tools\pqt-pytho
[LittleFS] uploader : C:\Users\Freenove\AppData\Local\Arduino15\packages\rp2040\hardware\rp2040

Resetting COM26
Converting to uf2, output size: 2097152, start address: 0x100ff000
Scanning for RP2040 devices
Flashing G: (RPI-RP2)
Wrote 2097152 bytes to G:/NEW.UF2
```

After the music uploads successfully, click the upload button to upload the sketch to Pico (W). The compilation of this sketch takes longer time, please wait with patience.



The effect of this example is that the speaker plays a piece of audio once. After the code finishes uploading, the speaker will play the audio file you uploaded.

Code

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

```
#include "AudioFileSourceLittleFS.h"
#include "AudioGeneratorMP3.h"
#include "AudioOutputI2SNoDAC.h"
#include "AudioFileSourceID3.h"

AudioGeneratorMP3 *mp3;
AudioFileSourceLittleFS *file;
AudioOutputI2SNoDAC *out;

void setup() {
  Serial.begin(115200);
  delay(1000);
  file = new AudioFileSourceLittleFS("1.mp3");
  out = new AudioOutputI2SNoDAC(6);
  out->SetGain(3);  //Volume Setup
  mp3 = new AudioGeneratorMP3();
  mp3->begin(file, out);
}
int a = 0;

void loop() {
  Serial.printf("loop1..\n");
  if (mp3->isRunning()) {
    if (!mp3->loop()) {
      mp3->stop();
      Serial.printf("Hello done\n");
      delete file;
      delete mp3;
      mp3 = new AudioGeneratorMP3();

    }
  } else {
    Serial.printf("MP3 done\n");
    pinMode(6, OUTPUT);
    digitalWrite(6, LOW);
  }
}
```

Code Explanation

Set the audio file to be played.

```
file = new AudioFileSourceLittleFS("1.mp3");
```

Set the volume of the audio at the range of 0.0-4.0.

```
out->SetGain(3);  //Volume Setup
```

Check whether the audio is being played.

```
if (mp3->isRunning()) {}
```

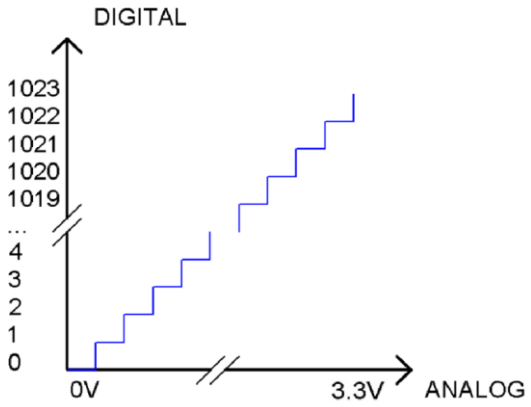
Check whether the audio has finished playing. If so, clear the audio file.

```
  if (!mp3->loop()) {
    mp3->stop();
    Serial.printf("Hello done\n");
    delete file;
    delete mp3;
    mp3 = new AudioGeneratorMP3();

  }
```

1.4 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.



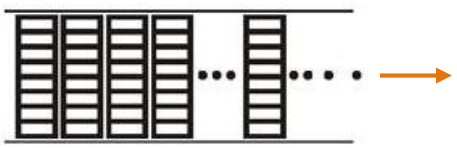
Subsection 1: the analog in rang of 0V---3.3/1023 V corresponds to digital 0;
Subsection 2: the analog in rang of 3.3/1023 V---2*3.3 /1023V corresponds to digital 1;
The following analog will be divided accordingly.
The conversion formula is as follows:

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

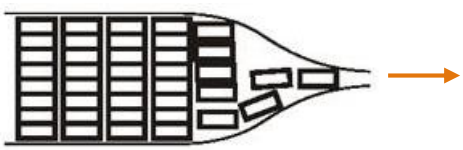
Serial Communication

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

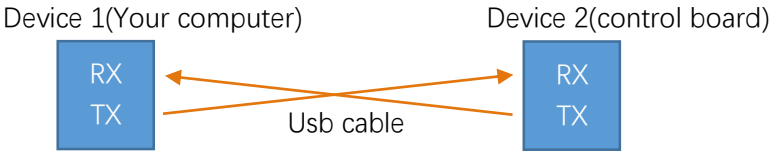


Serial communication



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:

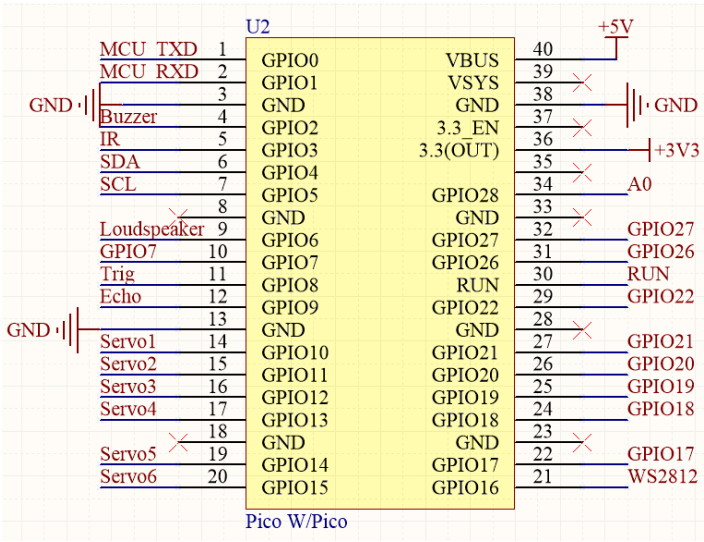


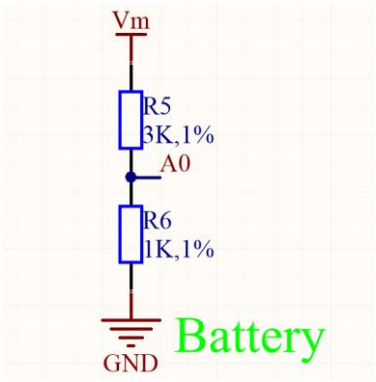
For serial communication, the **baud rate in both sides must be the same**. The baud rates commonly used are 9600 and 115200.

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 robot reads the voltage of the batteries through GPIO28 of Raspberry Pi Pico (W).

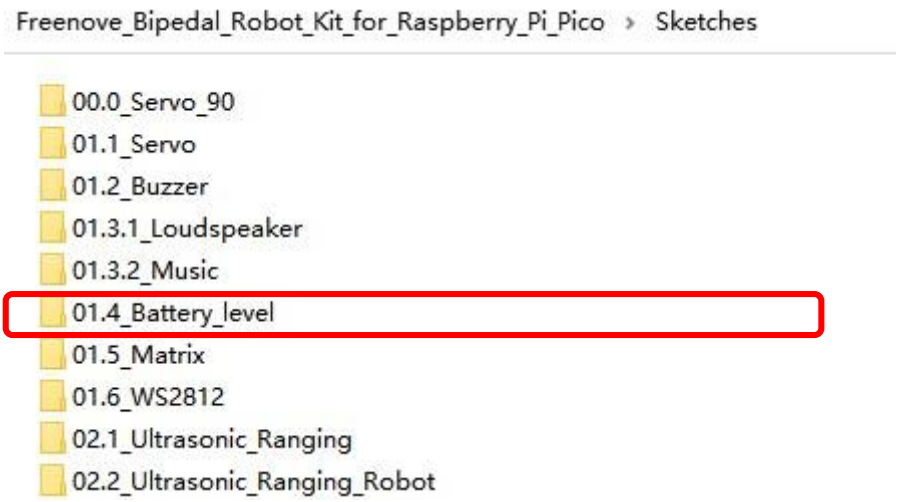




The voltage acquisition range of GPIO28 on the Raspberry Pi Pico W is 0-3.3V. However, the robot is powered with a 9V battery, and the voltage can reach 9.5V when fully charged, far exceeding the acquisition range of the pico (W). Therefore, a voltage divider circuit consisted of R3 and R4 is designed here. After passing the cicuit, the voltage of A0 is about one quarter of the battery voltage. Say the battery voltage is 9.5V, the voltage of A0 is 9.5/4 = 2.375V, which is within the voltage acquisition range of GPIO28.

Sketch

In this section, we will use GPIO28 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_Bipedal_Robot_Kit_for_Raspberry_Pi_Pico\Sketches"** and then double-click "01.4_Battery_level.ino".



Code

```
#include "Freenove_Robot_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 GPIO28 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
```

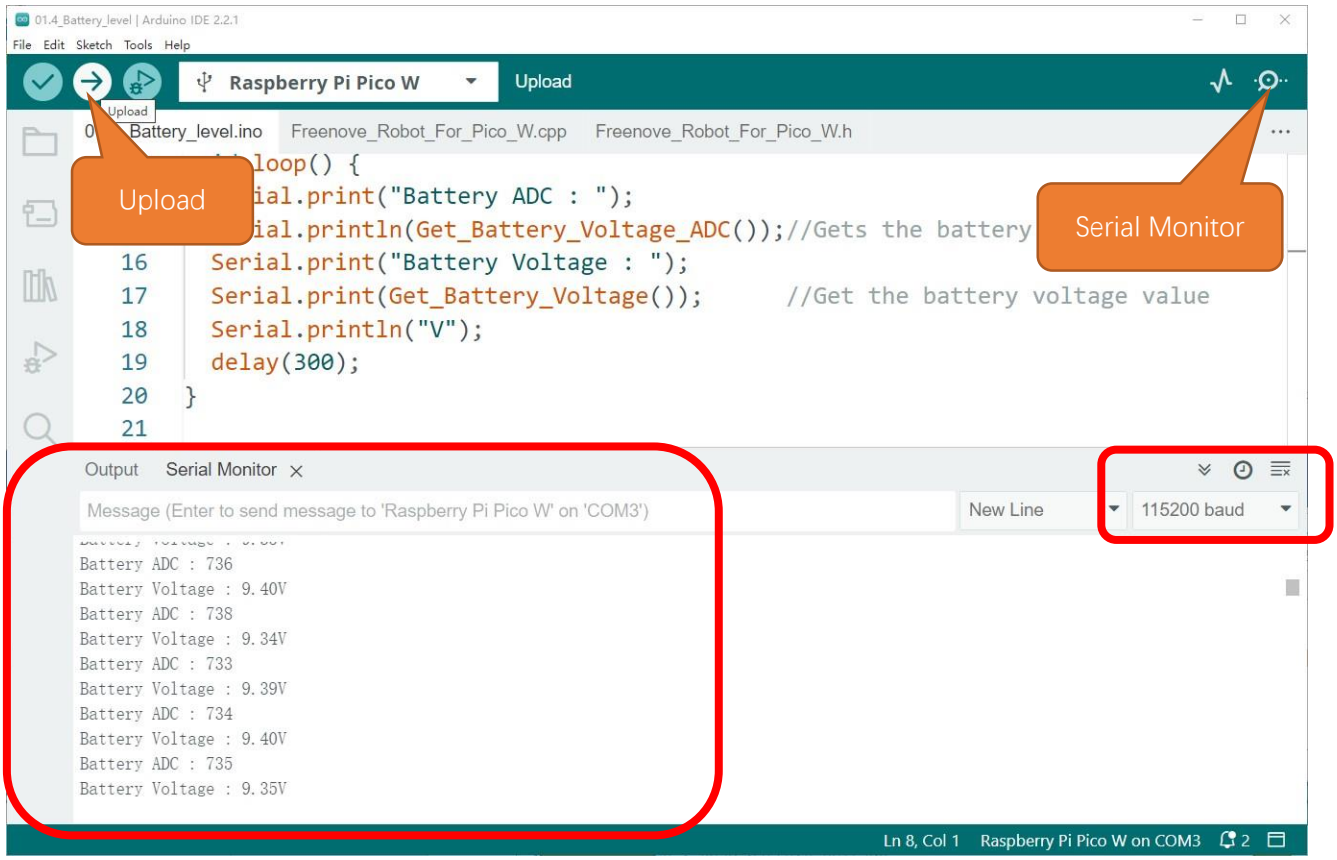
Calculate the voltage of batteris and return it.

```
  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
```

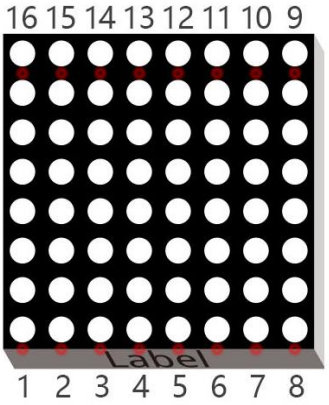
Click “Upload” to upload the code to Pico (W). After uploading successfully, click Serial Monitor,Set baud rate to 115200.



1.5 LED Matrix

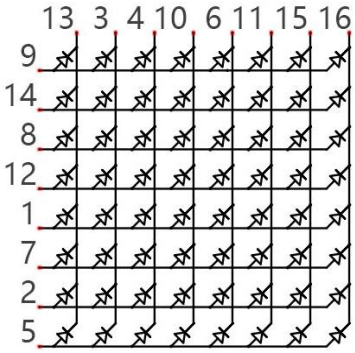
LED Matrix

An 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).

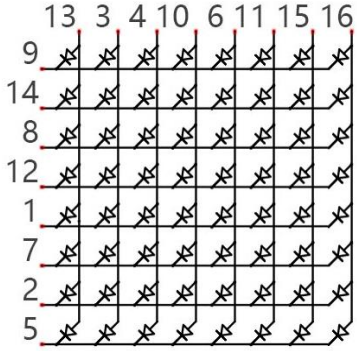


In order to facilitate the operation and reduce the number of ports required to drive this component, the positive poles of the LEDs in each row and negative poles of the LEDs in each column are respectively connected together inside the LED matrix module, which is called a common anode. There is another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

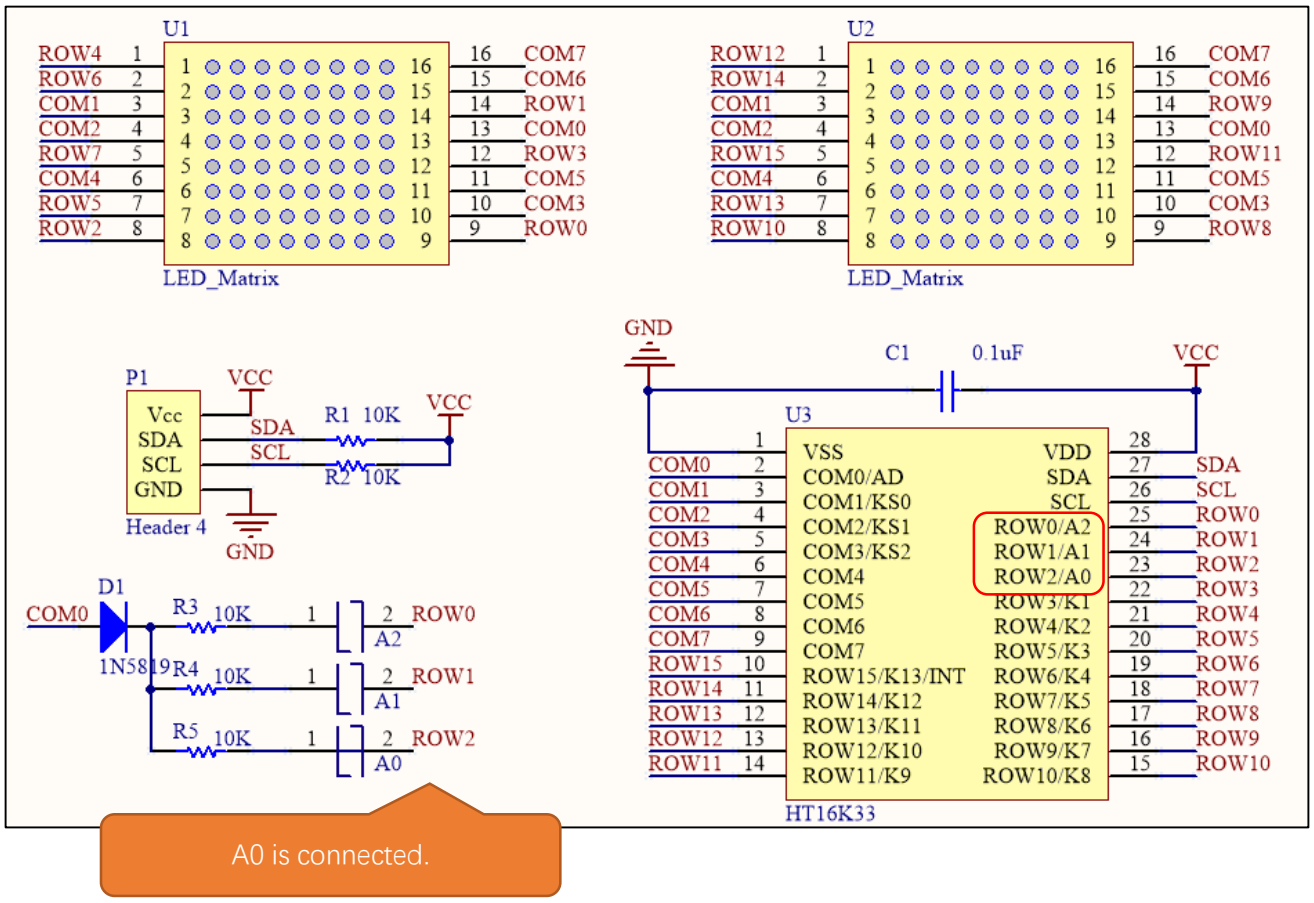


Connection mode of common cathode

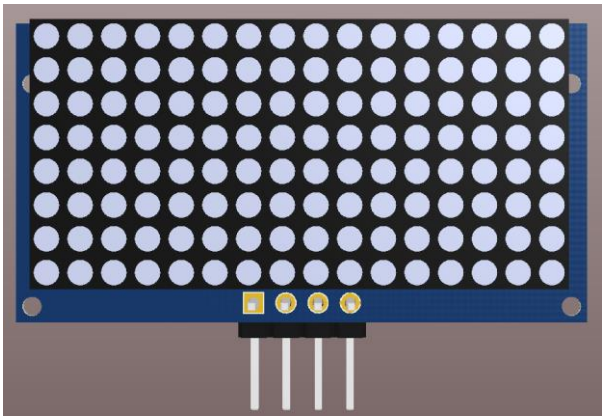
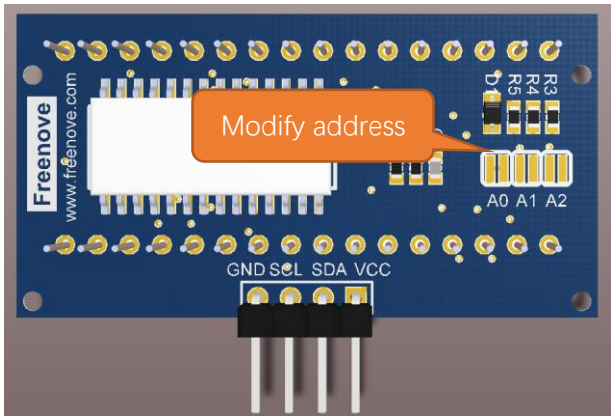


Schematic

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

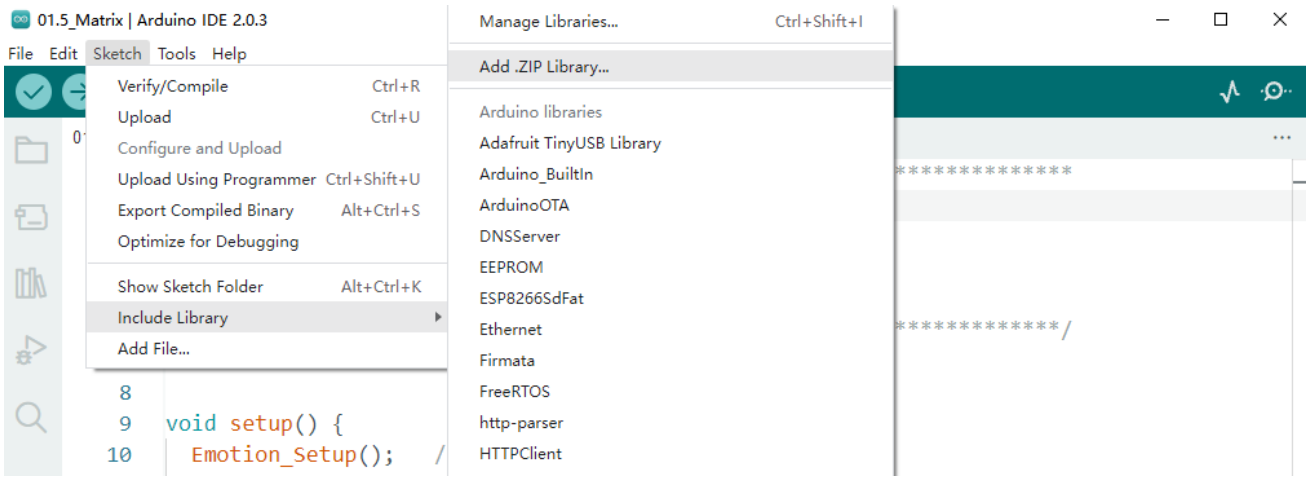
Row	Binary	Hexadecimal
1	0000 0000	0x00
2	0001 1000	0x18
3	0010 0100	0x24
4	0100 0010	0x42
5	0100 0010	0x42
6	0010 0100	0x24
7	0001 1000	0x18
8	0000 0000	0x00

Sketch

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

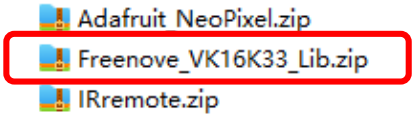
Install Freenove_VK16K33_Lib Library

Click Sketch and select Add .ZIP Library in Include.



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

"Freenove_Bipedal_Robot_Kit_for_Raspberry_Pi_Pico".



Install Processing

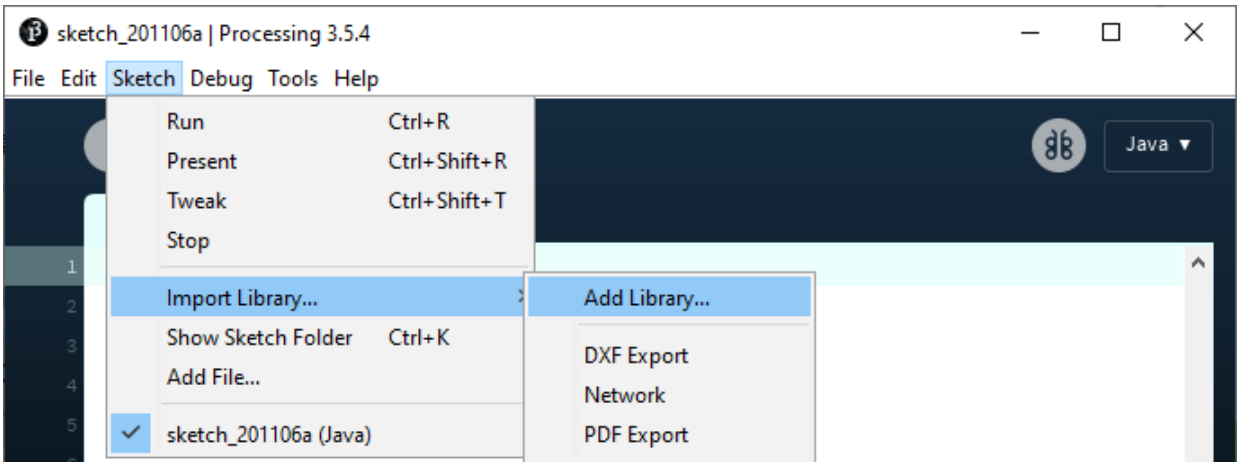
In this tutorial, we use Processing to build a simple LED Matrix platform.

If you have not installed Processing, you can download it from the official website: https://processing.org/.

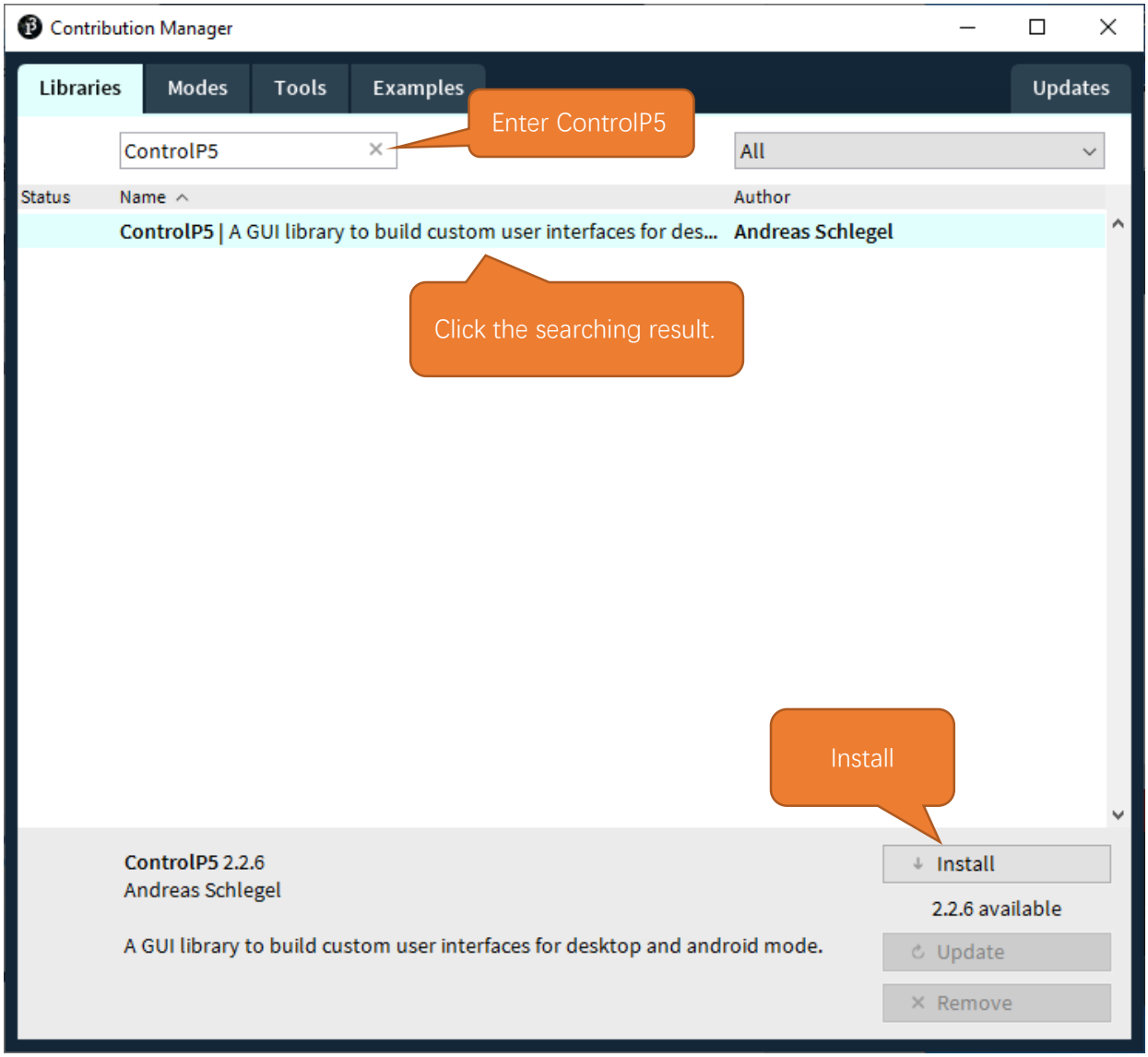
Download an appropriate version to download corresponding to your PC system.

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...”.

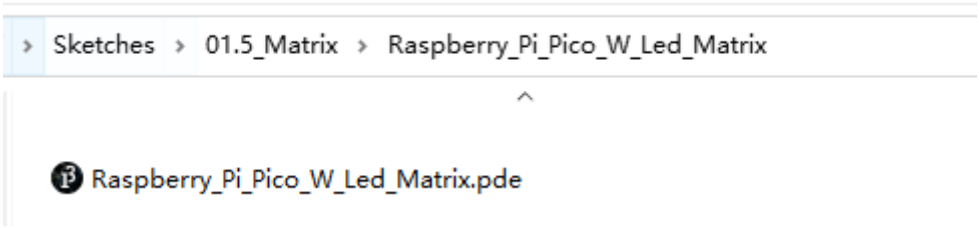


Enter “ControlP5” in the input field of the pop-up window. Click the searching result and then click “install”

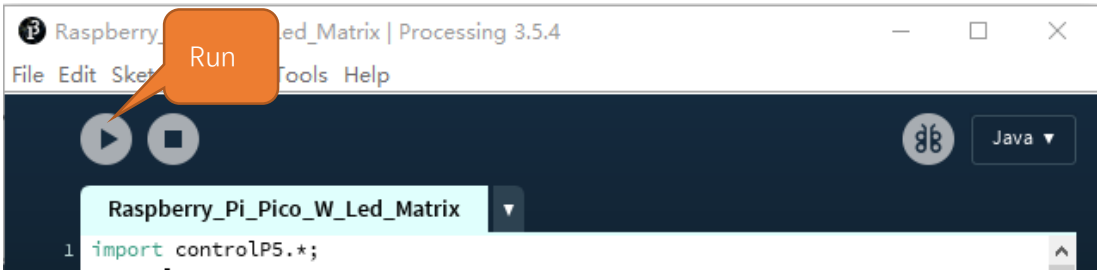


When the installation finishes, restart Processing.

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

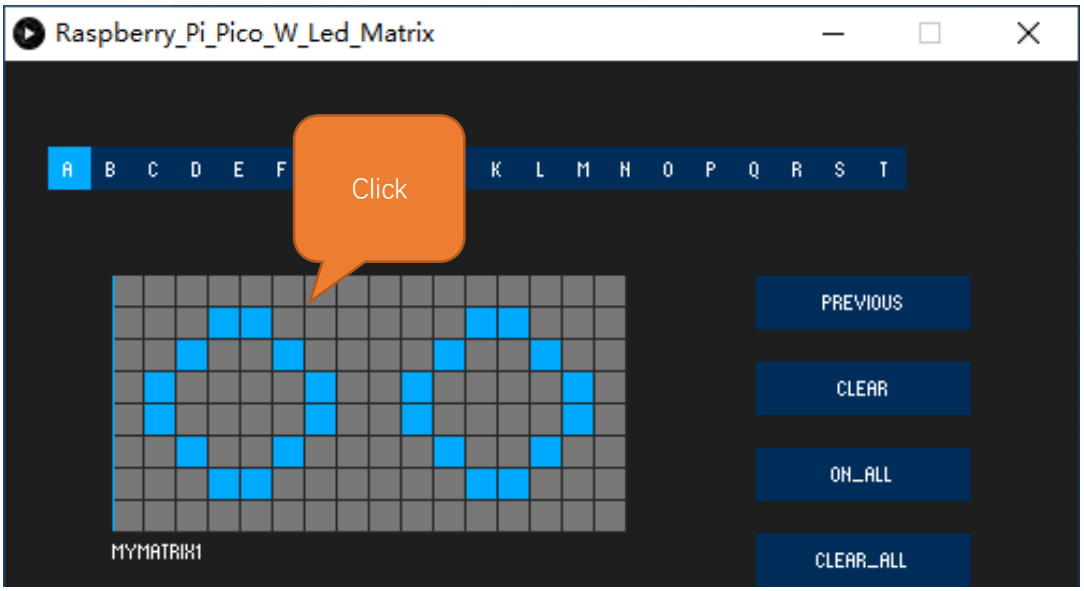


Click "Run"

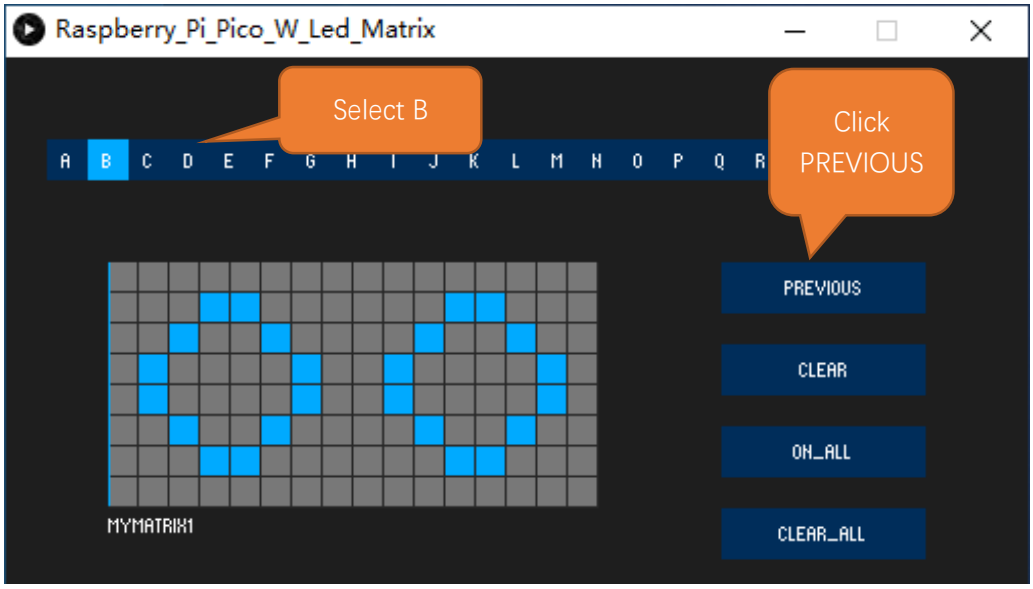


There are 20 pages from A to T. Select Page A.

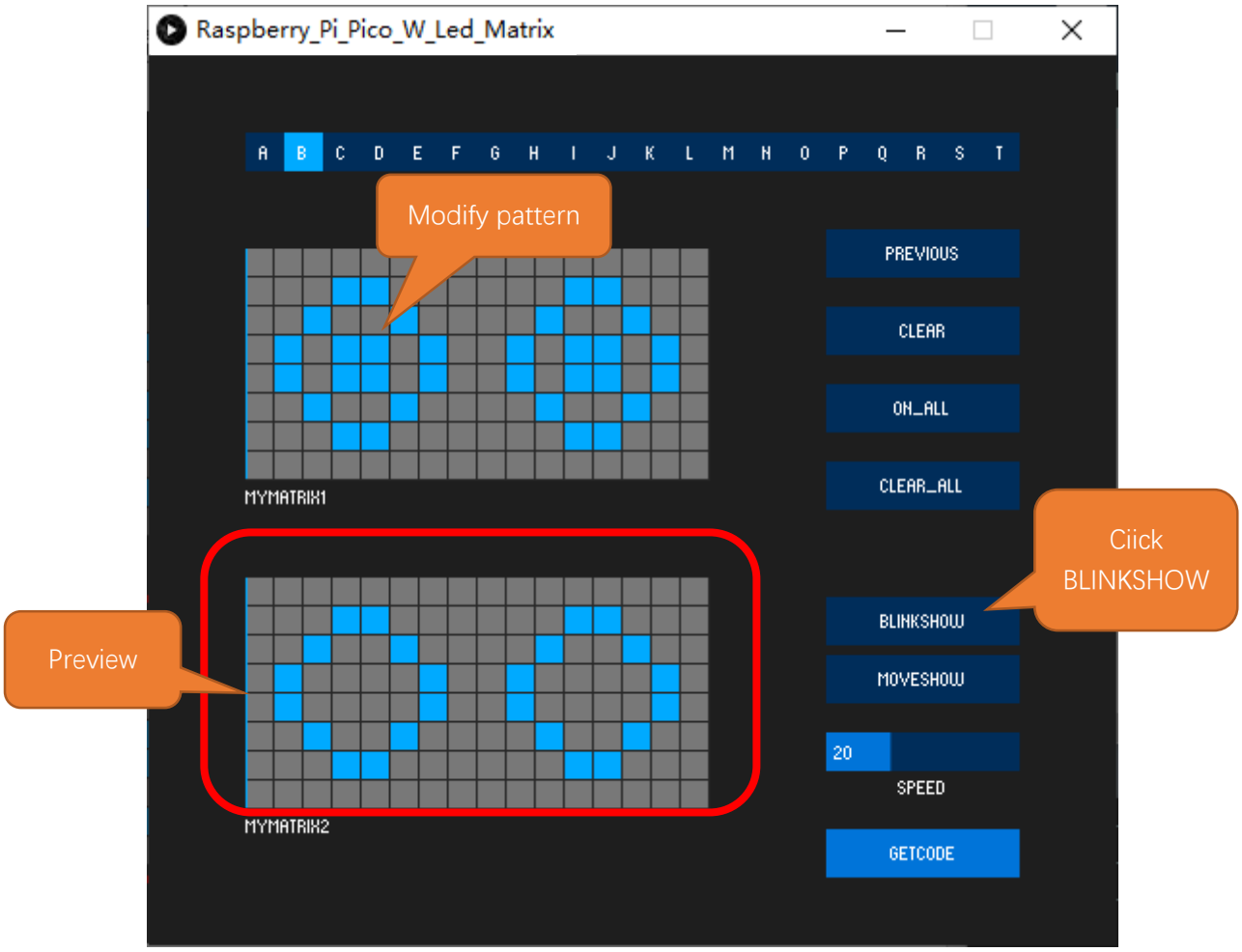
You can design your own pattern by clicking on the squares.



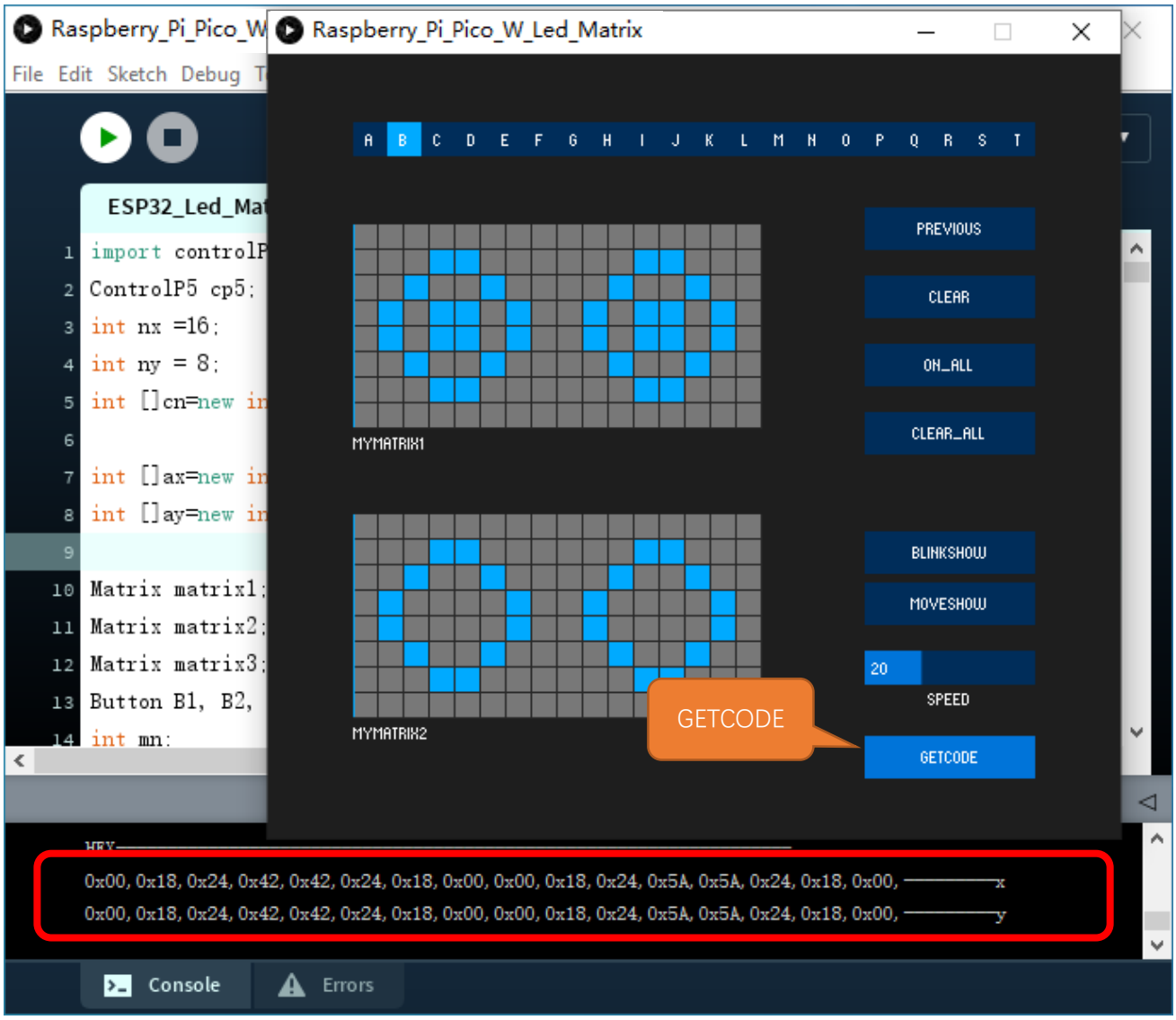
Next select Page B and click PREVIOUS, which copies the previous pattern to B.



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_Bipedal_Robot_Kit_for_Raspberry_Pi_Pico\Sketches"** and double click "01.5_Matrix.ino"

Code

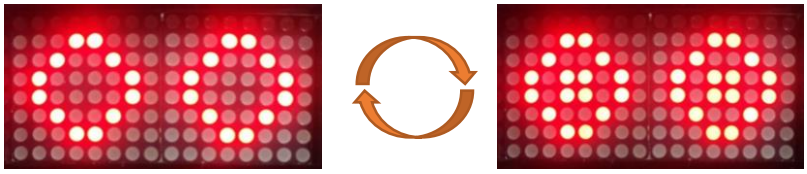
```
#include "Freenove_VK16K33_Lib.h"

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.flipVertical();//Flips the vertical orientation of the matrices.
  matrix.flipHorizontal();//Flips the horizontal orientation of the matrices.
  matrix.setBrightness(15);
  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);
  }
}
```

Replace the x array on the left

Replace the y array on the right

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 the IIC address and IIC pin of the HT16K33 chip. Call the init() function to initialize it. Set the flip direction of the LED matrix on both vertical and horizontal directions. Call the setBlink function to stop the the blink.

```
3   Freenove_VK16K33 matrix = Freenove_VK16K33();
…   ...
21    matrix.init(0x71);
22    matrix.flipVertical();//Flips the vertical orientation of the matrices.
23    matrix.flipHorizontal();//Flips the horizontal orientation of the matrices.
24    matrix.setBrightness(15);
25    matrix.setBlink(VK16K33_BLINK_OFF);
```

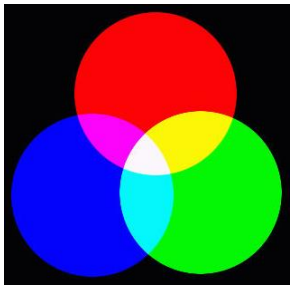
Define count to calculate the number of one-dimensional arrays contained in the two-dimensional x_array, and use the for loop to call the showStaticArray() function to continuously display the content of LED matrix.

```
35    int count = sizeof(x_array) / sizeof(x_array[0]);
36    for (int i = 0; i < count; i++)
37    {
38      matrix.showStaticArray(x_array[i], y_array[i]);
39      delay(delay_ms);
40    }
```

1.6 LED

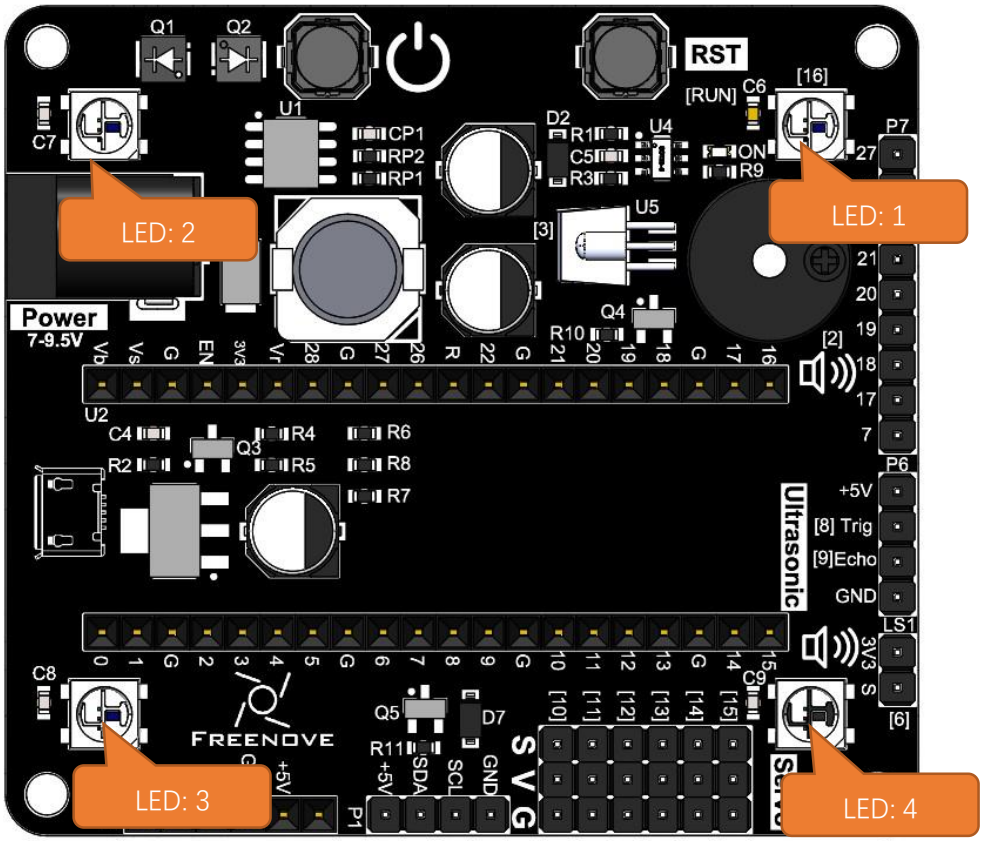
LED

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.



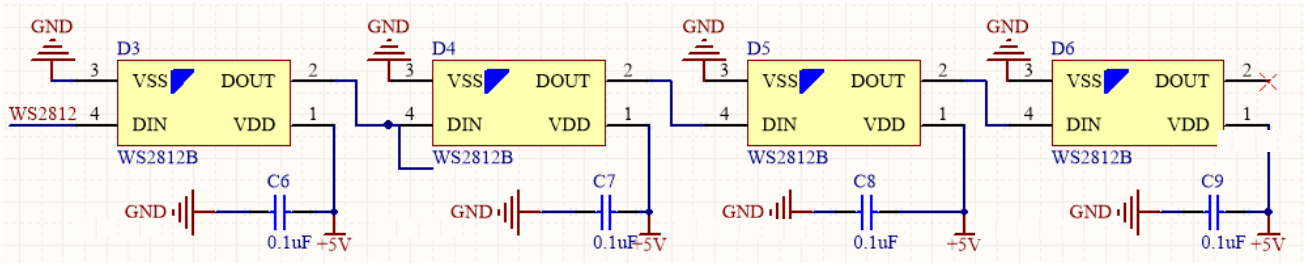
RGB

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

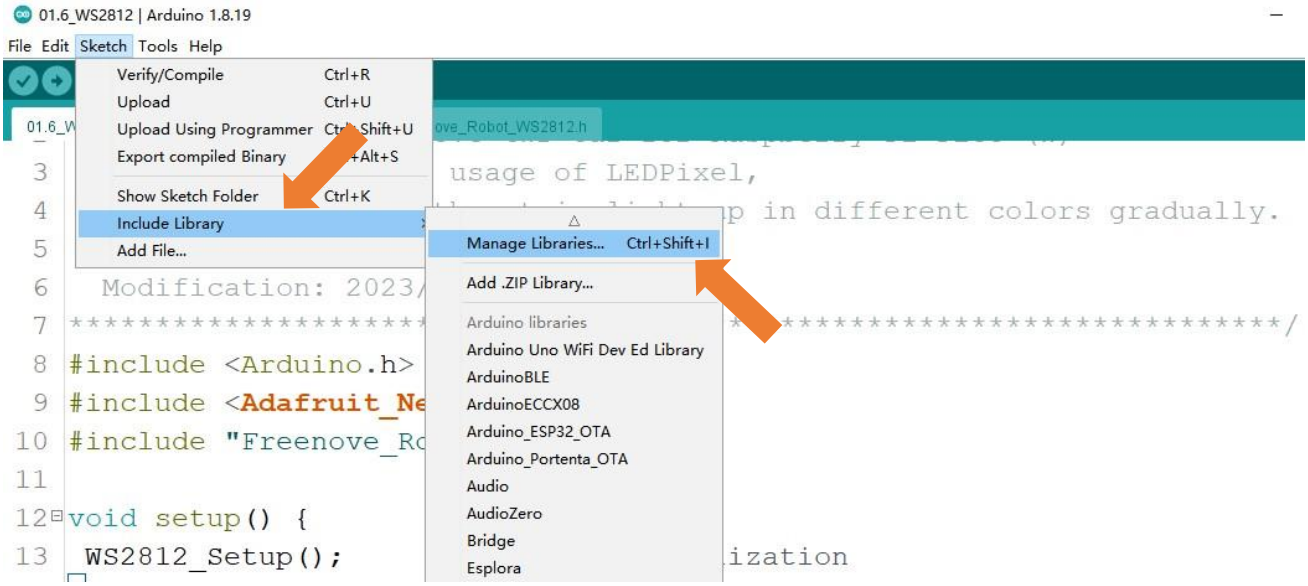
As shown below, the DOUT of each LED is connected with DIN of the next LED, and the 4 LED can be controlled to emit colorful colors by inputting control signals through LED.



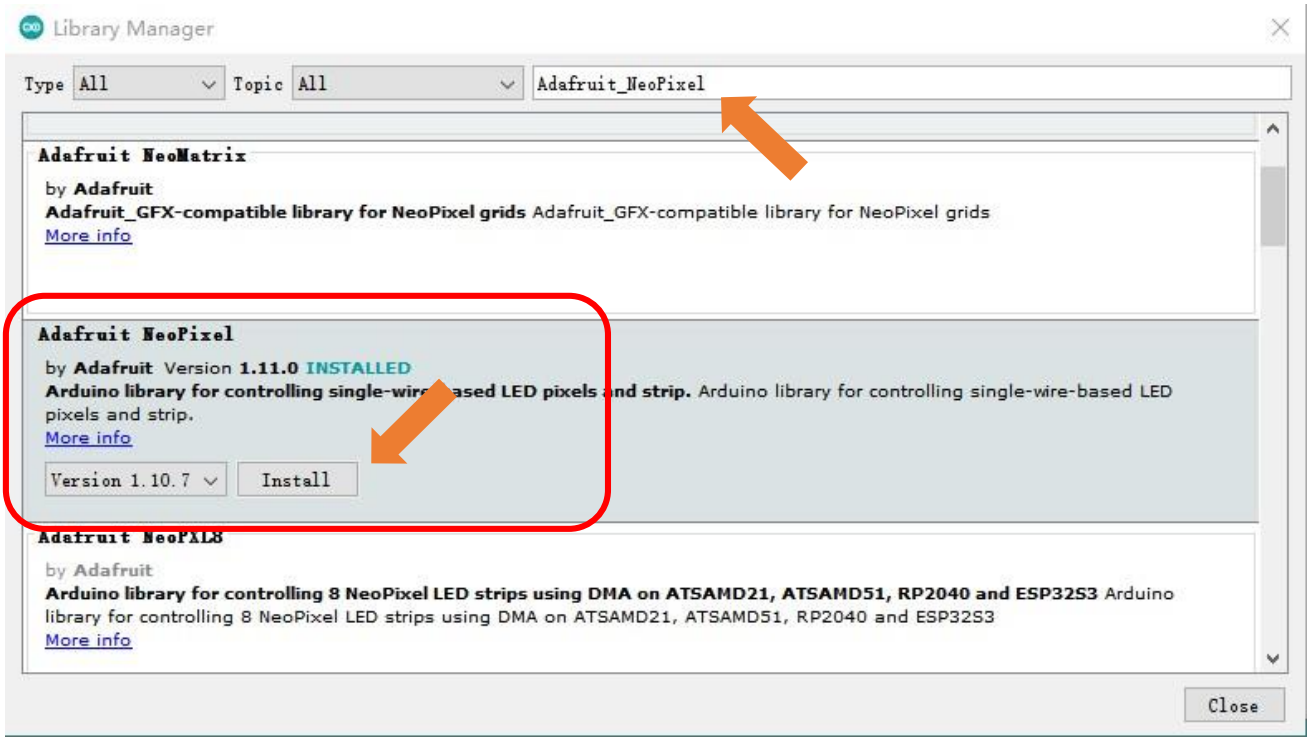
Sketch

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_Bipedal_Robot_Kit_for_Raspberry_Pi_Pico\Sketches"** and then double click "01.6_WS2812.ino".

Code

```
#include <Arduino.h>
#include <Adafruit_NeoPixel.h>
#include "Freenove_Robot_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();
```

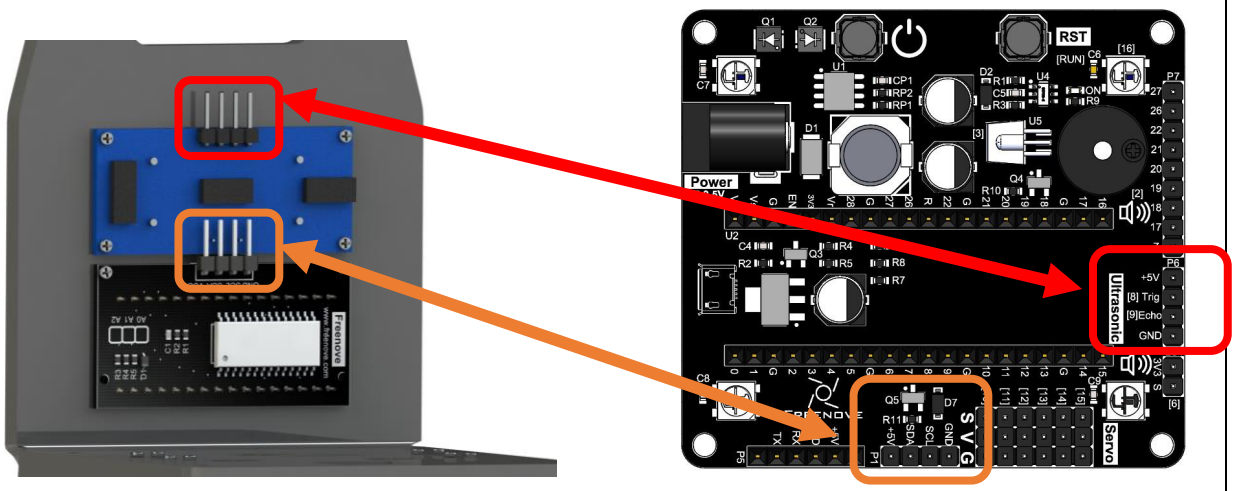
Chapter 2 Ultrasonic Obstacle Avoidance Robot

In this chapter, the robot will achieve the function of ultrasonic ostabcle avoidance.

2.1 Ultrasonic Module

Replace the ultrasonic module.

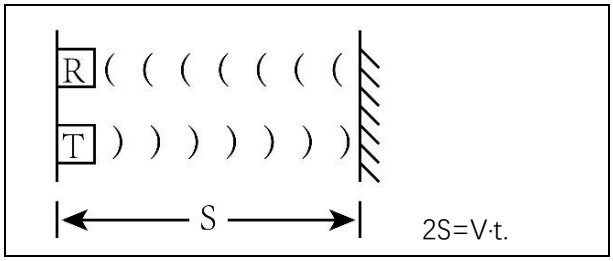
Step 1 Connect the ultrasonic module and the LED matrix module. It is important that the modules be connected in line with the silkprint of the pins.



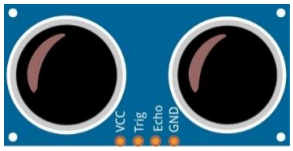
Module	Gnd	Echo	Trig	Vcc
Robot	GND	ECHO	TRIG	VCC
	GND	8	9	VCC

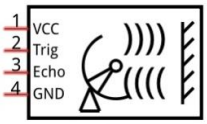
Ultrasonic Module

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:

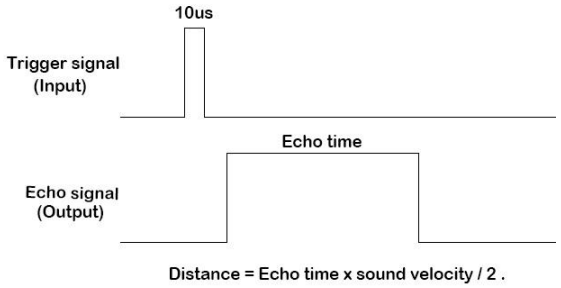
Pin	Description
VCC	Power supply pin
Trig	Trigger pin
Echo	Echo pin
GND	GND

Technical specs:

Working voltage: 5V Working current: 12mA

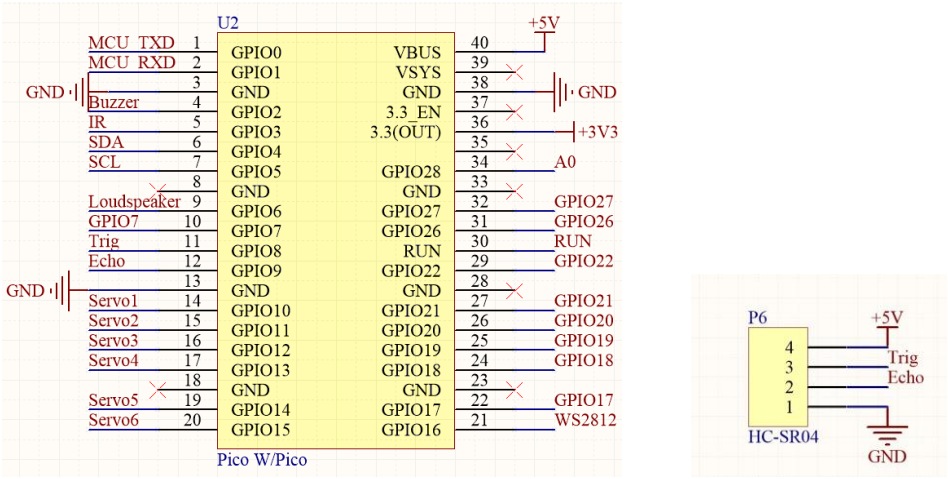
Minimum measured distance: 2cm Maximum measured distance: 200cm

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 GPIO8 and GPIO9 to control the Trig and Echo pins of the ultrasonic module.



Sketch

Turn ON the power switch of the robot, after the ultrasonic sensor finishes initialization, obtain and print the ultrasonic data on the serial monitor.

Open the folder “02.1_Ultrasonic_Ranging” in

“Freenove_Bipedal_Robot_Kit_for_Raspberry_Pi_Pico\Sketches”

and double click “02.1_Ultrasonic_Ranging.ino”

Code

```
#include <Arduino.h>
#include "Freenove_Robot_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
  delay(500);          //Wait for the servo to arrive at the specified location
}

void loop() {
  Serial.print("Distance: " + String(Get_Sonar()) + "\n");//Print ultrasonic distance
  delay(500);
}
```

Code Explanation:

Initialize ultrasonic module.

```
  Ultrasonic_Setup();  //Ultrasonic module initialization
```

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

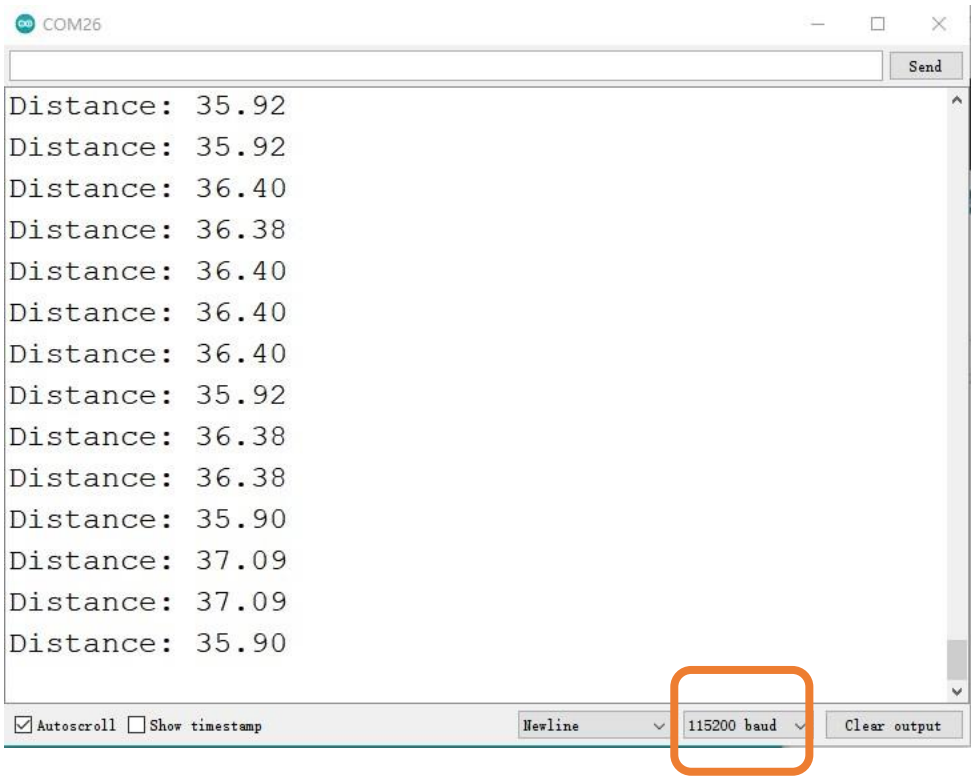
```
  Get_Sonar()
```

Obtain and print the ultrasonic data via the serial port, which is updated every 0.5 seconds.

```
  Serial.print("Distance: " + String(Get_Sonar()) + "\n");//Print ultrasonic distance
  delay(500);  Servo_1_Angle(30);   //Turn servo 1 to 30 degrees
```

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

Set the baud rate as 115200.



2.2 Obstacle Avoidance Robot

After the robot activates, the ultrasonic module collects data between it and the obstacles in the front. Then make judgements according to the data and control the robot to avoid the obstacles.

Sketch

Open the folder "02.2_Ultrasonic_Ranging_Robot"in **"Freenove_Bipedal_Robot_Kit_for_Raspberry_Pi_Pico\Sketches"** and double click "02.2_Ultrasonic_Ranging_Robot.ino".

Code

```
#include <Arduino.h>
#include "Freenove_Robot_For_Pico_W.h"
#include "Bipedal_Robot.h"
Bipedal_Robot Bipedal_Robot;

#define LeftLeg 10
#define RightLeg 12
#define LeftFoot 11
#define RightFoot 13
#define Buzzer 6


void setup() {
  Serial.begin(115200);
  Ultrasonic_Setup();           //Initialize the ultrasonic module
  Bipedal_Robot.init(LeftLeg, RightLeg, LeftFoot, RightFoot, true); //Set the servo pins
  Bipedal_Robot.home();
  delay(50);
}
void loop()
{
  if (Get_Sonar() <= 20)
  {
    Bipedal_Robot.sing(S_surprise);
    Bipedal_Robot.walk(2, 1000, -1); // BACKWARD x2
    Bipedal_Robot.turn(3, 1000, 1); // LEFT x3
  }
  Bipedal_Robot.walk(1, 1200, 1); // FORWARD x1
}

```

Code Explanation:

Check whether there is any obstacle in front of the robot with the ultrasonic wave. When the distance between the robot and the obstacles is smaller than the set number, the robot move backwards and turn left to avoid the obstacle.

```
21  void loop()
22  {
23    if (Get_Sonar() <= 20)
24    {
25      Bipedal_Robot.sing(S_surprise);
26      Bipedal_Robot.walk(2, 1000, -1); // BACKWARD x2
27      Bipedal_Robot.turn(3, 1000, 1); // LEFT x3
28    }
29    Bipedal_Robot.walk(1, 1200, 1); // FORWARD x1
30  }
```

Chapter 3 Sing and Dance

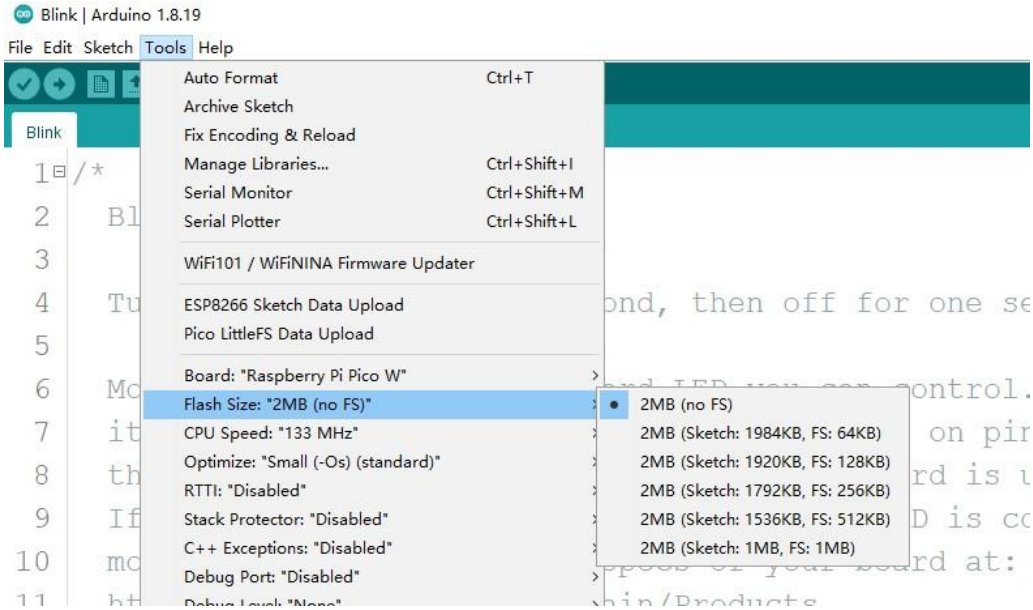
3.1 Playing Music

In this chapter, we use the speaker to play a piece of music. Please note that to play the music, it requires the use of file system. The raspberry pi pico (W) has built-in 264KB SRAM and 2MB of onboard flash, and the Arduino-Pico core supports using some of the onboard flash as a file system, useful for storing configuration data, output strings, logging, and more. It also supports using SD cards as another (FAT32) filesystem, with an API that's compatible with the onboard flash file system.

The following diagram shows the flash layout used in Arduino-Pico:

```
|--------------------|-------------|----|
^                    ^             ^
Sketch               File system   EEPROM
```

The storage of the raspberry pi pico (W) file system can be configured via the Arduino IDE menu bar, ranging from 0KB to 1MB.



Due to the limitation of the storge, the file uploaded to pico (W) should not exceed 1024Kb (1M). Therefore, we need to restrict the file scale; otherwise, it may fail to upload.

Upload the file to LittleFS file system.

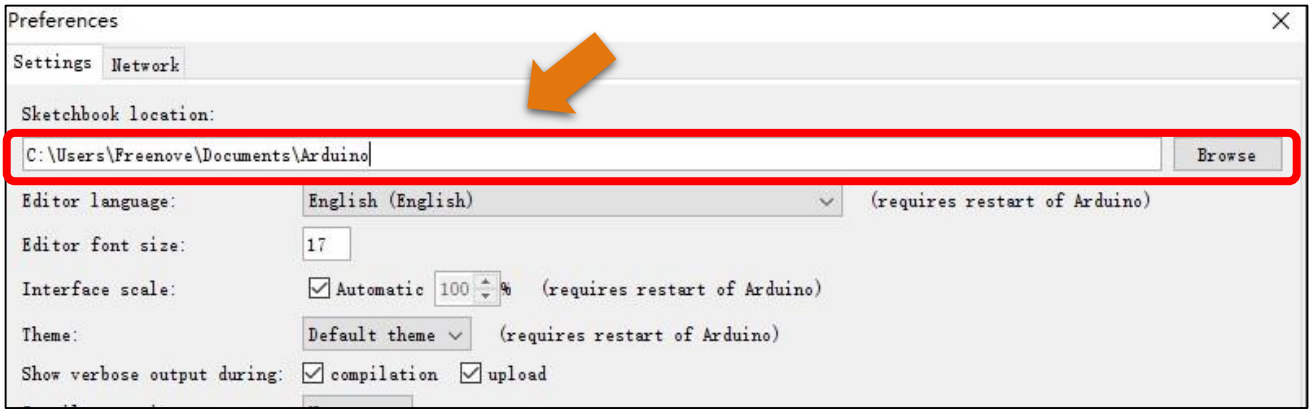
LittleFS is an onboard file system that sets aside some program flash memory for use as a file system, without requiring any external hardware.

It is also a tool integrated into the Arduino IDE, adding an item to the Tools menu for uploading the contents of a sketch data directory to a new LittleFS flash file system.

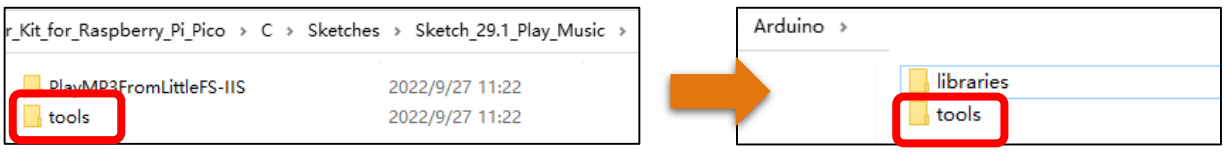
Installing PicoLittleFS Tool

The steps to install PicoLittleFS tools are as follows:
First, open the Arduino IDE, and then click File in Menus and select Preferences.

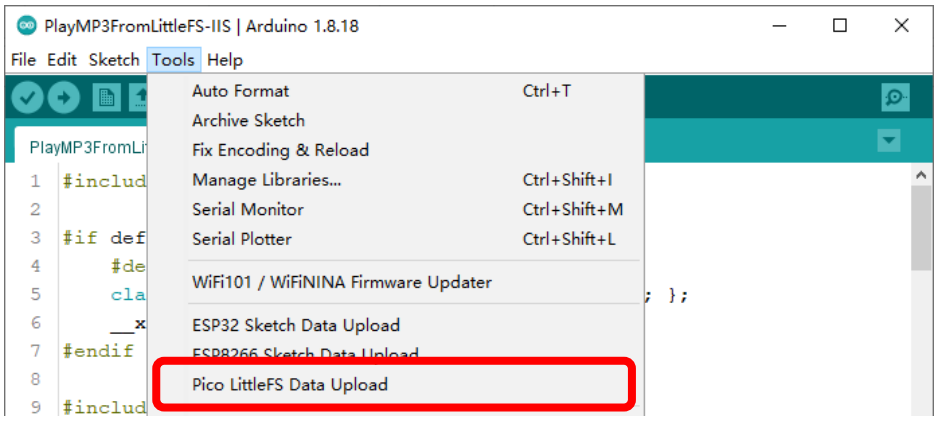
Find the Arduino IDE environment directory location.



Copy the tools folder in the code folder to your Sketchbook location.



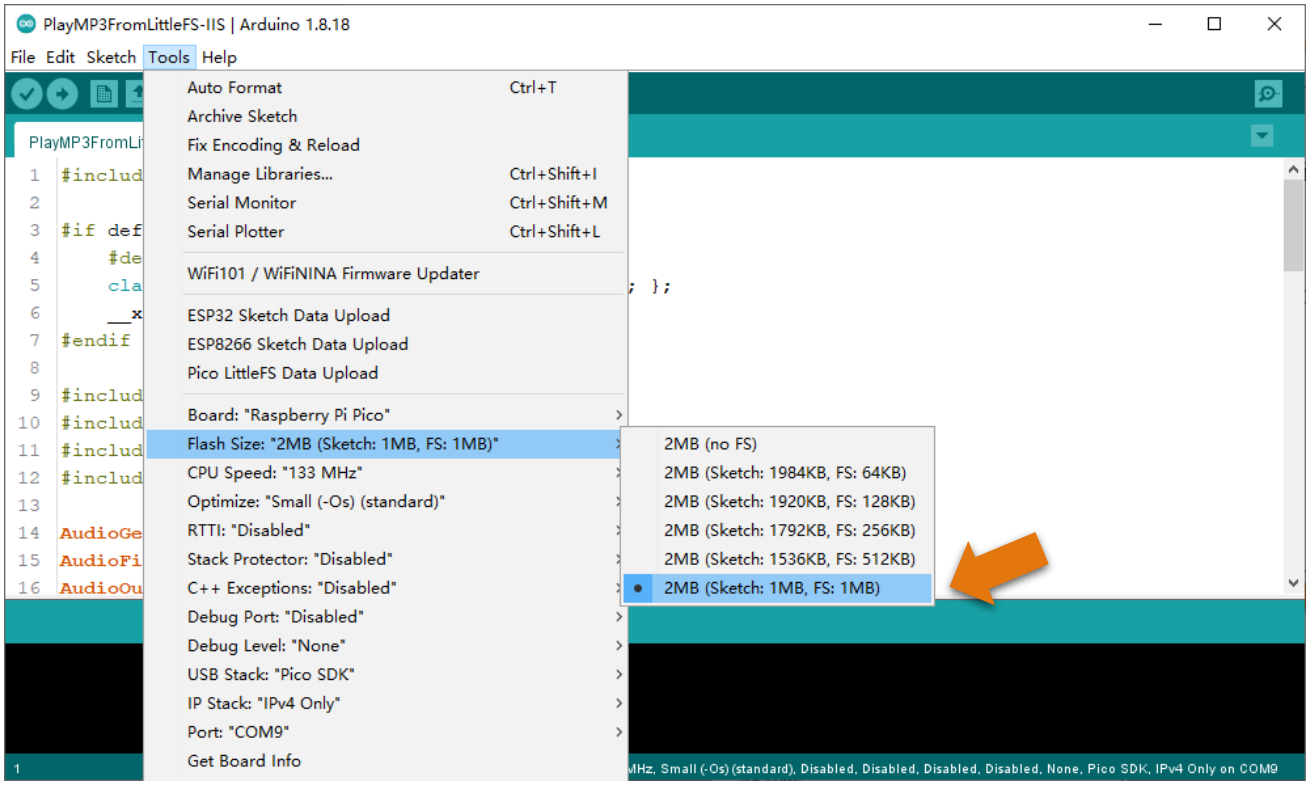
Finally, restart the Arduino IDE. After restarting, you can see that the plug-in already exists in the interface.


PlayMP3FromLittleFS-IIS | Arduino 1.8.18
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
ESP32 Sketch Data Upload
Pico LittleFS Data Upload

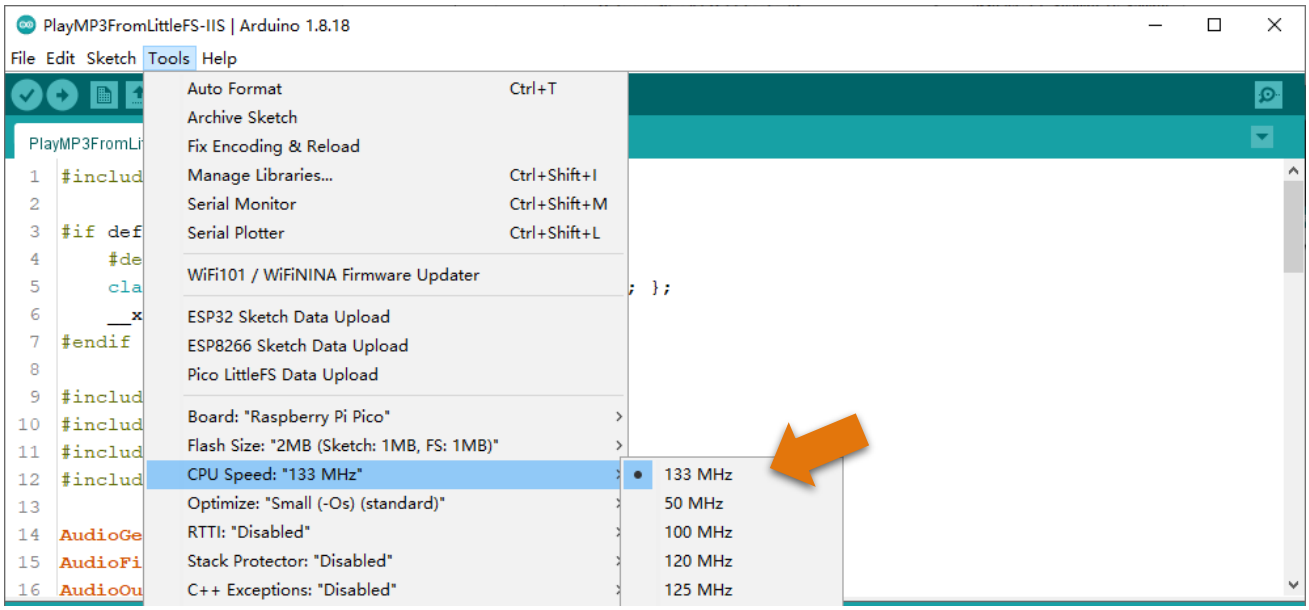
Upload music

Pico of Raspberry Pie has 2M Flash space. Generally, Arduino mode allocates it to the code area. Therefore, before starting, we need to modify the configuration of Flash Size.

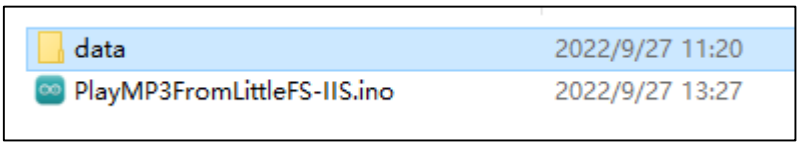
Open Arduino, select Tools from the menu bar, select Flash Size, and allocate 1MB of Flash space to store codes and 1MB to store audio files.



Make sure the CPU Speed of Raspberry Pi pico is 133Mhz. If the frequency is too low, the audio decoding speed may be too slow and the audio playback may not be continuous.



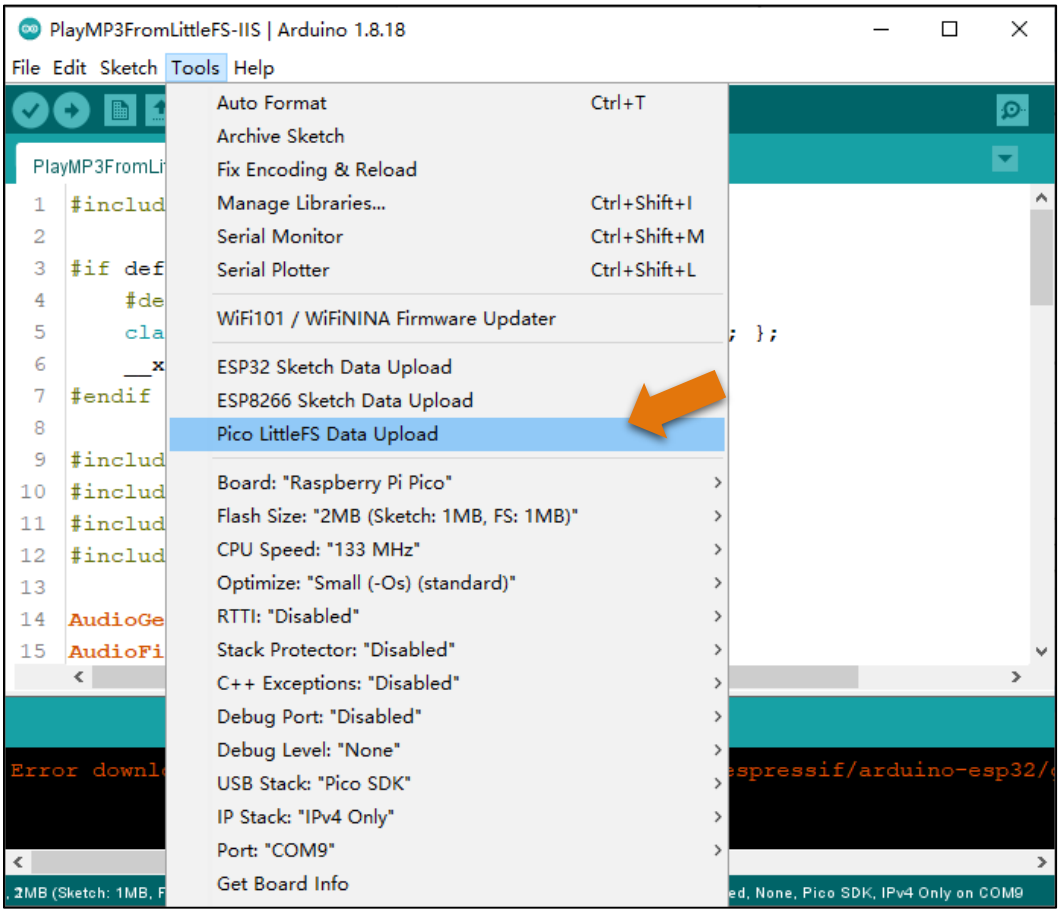
Check the code file and audio file. We create a folder named data under the same level directory of the code file, and place the audio file directly in this folder.

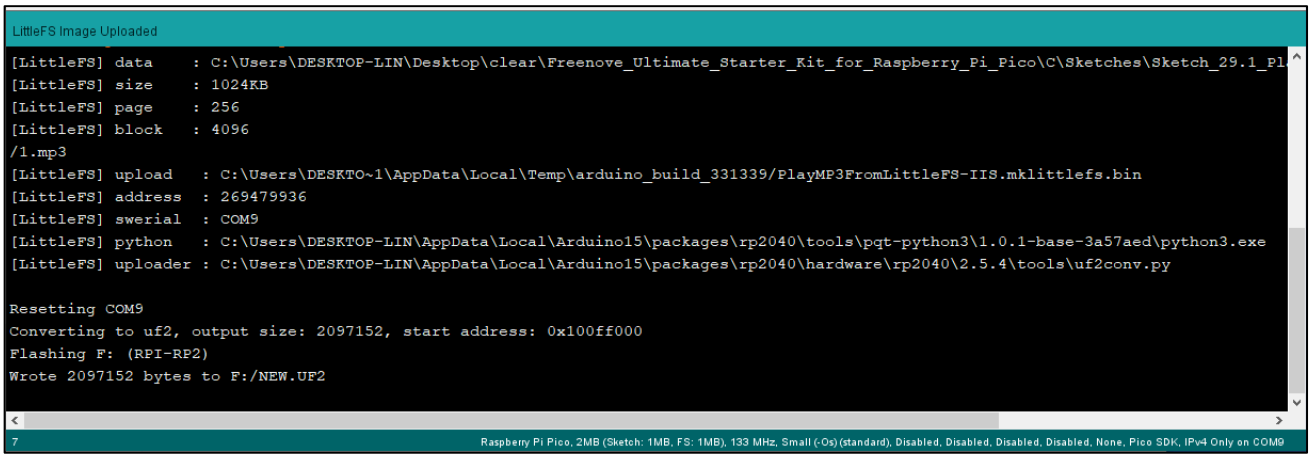


Note: 1. The name of the data folder cannot be changed, otherwise the plug-in cannot be used to upload audio files to Pico.

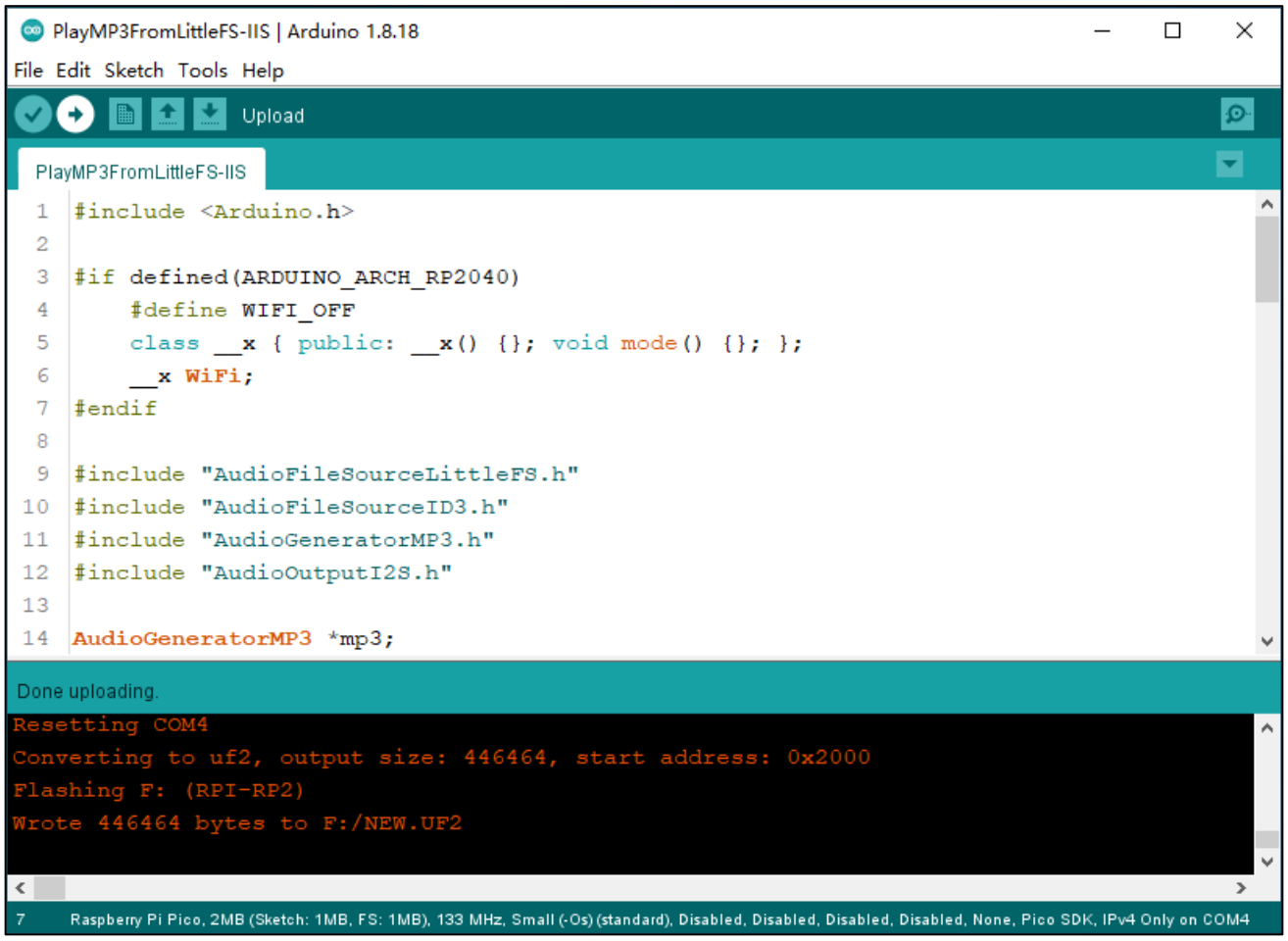
2. The number of audio files in the data folder is unlimited, but the total size cannot exceed 1MB. If the file upload fails, please check whether the data folder size exceeds the range.

Click Arduino IDE Tools and click following content:





After uploading the music, click the upload button to upload the code to Pico.



After the code upload is completed, the speaker will play the audio file you uploaded.

Sketch

Open the folder "03.1_Music" in **"Freenove_Bipedal_Robot_Kit_for_Raspberry_Pi_Pico\Sketches"** and double click "03.1_Music.ino.ino"

The sketch is the same as 01.3.2_Music.ino. Now, let's upload the code to Pico (W) to play the music.

Code

```
#include <Arduino.h>
#include "AudioFileSourceLittleFS.h"
#include "AudioGeneratorMP3.h"
#include "AudioOutputI2SNoDAC.h"
#include "AudioFileSourceID3.h"

AudioGeneratorMP3 *mp3;
AudioFileSourceLittleFS *file;
AudioOutputI2SNoDAC *out;

void setup() {
  Serial.begin(115200);
  delay(1000);
  file = new AudioFileSourceLittleFS("1.mp3");
  out = new AudioOutputI2SNoDAC(6);
  out->SetGain(3);  //Volume Setup
  mp3 = new AudioGeneratorMP3();
  mp3->begin(file, out);
}
int a = 0;

void loop() {
  Serial.printf("loop1..\n");
  if (mp3->isRunning()) {
    if (!mp3->loop()) {
      mp3->stop();
      Serial.printf("Hello done\n");
      delete file;
      delete mp3;
      mp3 = new AudioGeneratorMP3();

    }
  } else {
    Serial.printf("MP3 done\n");
    pinMode(6, OUTPUT);
    digitalWrite(6, LOW);
  }
```

```
}
```

3.2 Sing and dance

In this chapter, we will have the robot sing while dancing.

Sketch

Upload the code and you will get a robot that is singing while dancing.

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

Part of code

```
#include <Arduino.h>
#include <EEPROM.h>
#include "Bipedal_Robot.h"
#include "AudioFileSourceLittleFS.h"
#include "AudioGeneratorMP3.h"
#include "AudioOutputI2SNoDAC.h"
#include "AudioFileSourceID3.h"
#include "AudioOutputI2S.h"

Bipedal_Robot Bipedal_Robot;
AudioGeneratorMP3 *mp3;
AudioFileSourceLittleFS *file;
AudioOutputI2SNoDAC *out;
AudioFileSourceID3 *id3;

#define LeftLeg 10
#define RightLeg 12
#define LeftFoot 11
#define RightFoot 13
#define Buzzer 6

int calibratePosition[4] = { -17, -10, -15, 12 };

void setup() {
  Serial.begin(115200);
  EEPROM.begin(512);                                          //Initialize the
ultrasonic module
  Bipedal_Robot.init(LeftLeg, RightLeg, LeftFoot, RightFoot, true);  //Set the servo pins
```

```
29    //Bipedal_Robot.setTrims(calibratePosition[0], calibratePosition[1], calibratePosition[2],
      calibratePosition[3]);
30    calib_homePos();
      Bipedal_Robot.saveTrimsOnEEPROM();
31    EEPROM.commit();
32    Bipedal_Robot.home();
33    delay(50);
34  }
35  void loop() {
36    dance();
37  }
38
39  void setup1() {
40    playmusic(1, 2);
41    pinMode(6, OUTPUT);
42    digitalWrite(6, LOW);
43  }
44  void loop1() {
45    palymusic(1,4);
46  }
47
48
```

Code Explanation:

The Raspberry Pi pico (W) is equipped with a dual-core ARM Cortex M0+ processor. Core 0 is used to run the robot's dancing movements, and core 1 is for music playback. The two cores are independent and do not affect each other when running code.

```
  void loop() {
    dance();
  }

  void loop1() {
    palymusic(1,4);
  }
```

Chapter 4 Infrared Robot

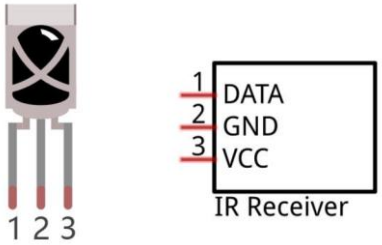
4.1 Introduction of infrared reception function

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

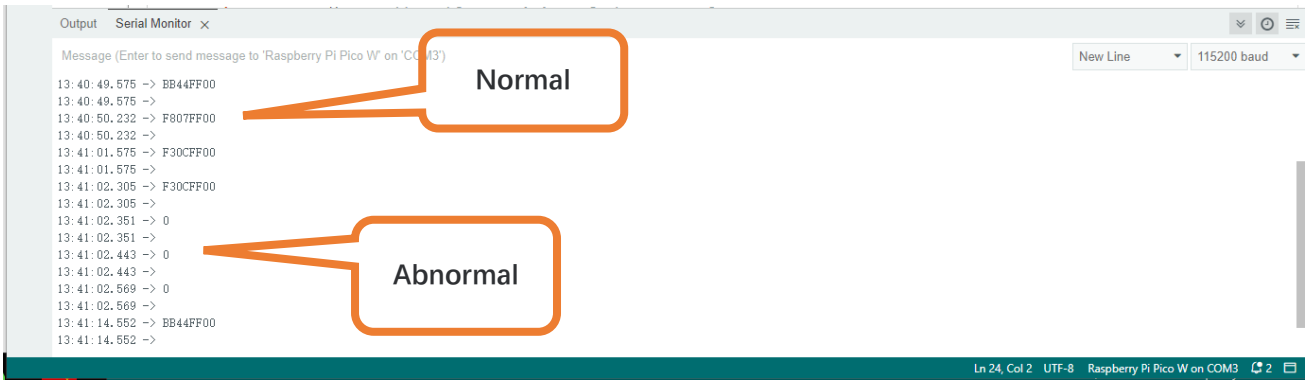
Each time when you press the infrared remote control, the car will print the received infrared coding value through serial port.

Open the folder “04.1_IR_Receiver”in **“Freenove_Robot_Kit_for_Raspberry_Pi_Pico\Ordinary_wheels\Sketches”** and double click “04.1_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 and set the baud rate to 115200. Press any keys on the IR remote and their corresponding values 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    }
```

4.2 Infrared Robot

On the basis of the previous section, we further controls the robot via the infrared remote controller. Press the black buttons on the remote, the robot will move forward, move backward, turn left and turn right accordingly. Press other buttons will stop the robot.

Sketch

Open the folder "04.2_IR_Receiver_Robot" in the **"Freenove_Bipedal_Robot_Kit_for_Raspberry_Pi_Pico\Sketches"** and double click "04.2_IR_Receiver_Robot.ino".

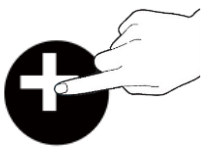
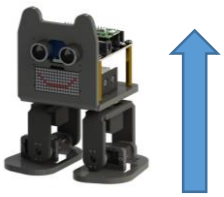
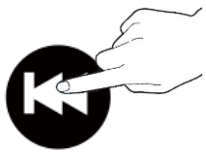
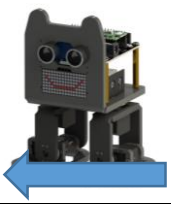
Code

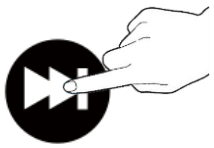
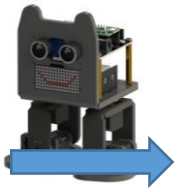
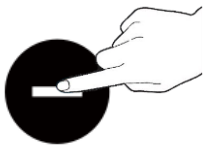
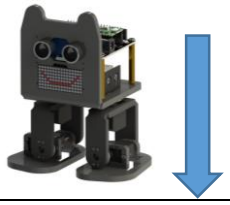
```
#include <Arduino.h>
#include <IRremote.hpp>
#include <EEPROM.h>
#include "Bipedal_Robot.h"

#define LeftLeg 10
#define RightLeg 12
#define LeftFoot 11
#define RightFoot 13

#define IR_Pin 3
#define ENABLE_LED_FEEDBACK true
#define DISABLE_LED_FEEDBACK false

// #define USE_NO_SEND_PWM

int move_flag = 0;

Bipedal_Robot Bipedal_Robot;
int calibratePosition[4] = { -17, -10, -15, 12 };

void calib_homePos() {
  int servoPos[4];
  servoPos[0] = 90;
  servoPos[1] = 90;
  servoPos[2] = 90;
  servoPos[3] = 90;
  Bipedal_Robot._moveServos(500, servoPos);
  Bipedal_Robot.detachServos();
}
```

```
31
32  void setup() {
33    Serial.begin(115200);
34    EEPROM.begin(512);
35    Bipedal_Robot.init(LeftLeg, RightLeg, LeftFoot, RightFoot, true);  //Set the servo pins
36    //Bipedal_Robot.setTrims(calibratePosition[0], calibratePosition[1], calibratePosition[2],
37  calibratePosition[3]);
38    calib_homePos();
39    Bipedal_Robot.saveTrimsOnEEPROM();
40    EEPROM.commit();
41    Bipedal_Robot.home();
42    IrReceiver.begin(IR_Pin, DISABLE_LED_FEEDBACK);  // Start the receiver
43  }
44
45  void loop() {
46    if (IrReceiver.decode()) {
47      unsigned long value = IrReceiver.decodedIRData.decodedRawData;
48      handleControl(value);         // Handle the commands from remote control
49      Serial.println(value, HEX);  // Print "old" raw data
50      Serial.println();
51      IrReceiver.resume();  // Enable receiving of the next value
52    }
53    Move(move_flag);
54  }
55
56  void handleControl(unsigned long value) {
57    // Handle the commands
58    switch (value) {
59      case 0xBF40FF00:  // Receive the number '+'
60        move_flag = 1;
61        break;
62      case 0xE619FF00:  // Receive the number '-'
63        move_flag = 2;
64        break;
65      case 0xF807FF00:  // Receive the number '|<<'
66        move_flag = 3;
67        break;
68      case 0xF609FF00:  // Receive the number '>>|'
69        move_flag = 4;
70        break;
71      case 0xEA15FF00:  // Receive the number '▶'
72        move_flag = 0;
73        break;
74      default:
```

```
        move_flag = 0;
        break;
    }
}
void Move(int data) {
    // Handle the commands
    switch (data) {
      case 1:  // Receive the number '+'
        Bipedal_Robot.walk(2, 1500, 1);
        break;
      case 2:  // Receive the number '-'
        Bipedal_Robot.walk(2, 1500, -1);
        break;
      case 3:  // Receive the number '|<<'
        Bipedal_Robot.turn(2, 1500, 1);
        break;
      case 4:  // Receive the number '>>|'
        Bipedal_Robot.turn(2, 1500, -1);
        break;
      case 0:  // Receive the number '▶'
        Bipedal_Robot.home();
        break;
      default:
        move_flag = 0;
        Bipedal_Robot.home();
        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.

```
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);
}
```

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 0xE916FF00:  // Receive the number '0'
     ...
    default:        // Control the car to stop moving
     ...
  }
}
```

4.3 Multi-Functional Infrared Robot

Now we integrated other functions to the infrared remote, so that most functions of the robot can be controlled via the infrared remote.

Sketch

Open the folder "05.3_Multi_Functional_Robot" in the **"Freenove_Bipedal_Robot_Kit_for_Raspberry_Pi_Pico\Sketches"** and double click "04.3_Multi_Functional_Robot.ino".

Part of code

```
#include <Arduino.h>
#include <Adafruit_NeoPixel.h>
#include <IRremote.hpp>
#include <EEPROM.h>
#include "Freenove_Robot_For_Pico_W.h"
#include "Freenove_Robot_Emotion.h"
#include "Freenove_VK16K33_Lib.h"
#include "Freenove_Robot_WS2812.h"
#include "Bipedal_Robot.h"

#define LeftLeg 10
#define RightLeg 12
#define LeftFoot 11
#define RightFoot 13

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

int move_flag = 0;
extern int move_flag;

Bipedal_Robot Bipedal_Robot;
int calibratePosition[4] = { -17, -10, -15, 12 };
int emotion_flag = 0;
int ws2812_flag = 0;

void calib_homePos() {
  int servoPos[4];
  servoPos[0] = 90;
  servoPos[1] = 90;
```

```
  servoPos[2] = 90;
  servoPos[3] = 90;
  Bipedal_Robot._moveServos(500, servoPos);
  Bipedal_Robot.detachServos();
}

void setup() {
  Serial.begin(115200);  //Turn on the serial port monitor and set the baud rate to 115200
  WS2812_Setup();        //WS2812 initialization
  Buzzer_Setup();        //Initialize the buzzer
  Ultrasonic_Setup();
  EEPROM.begin(512);
  Bipedal_Robot.init(LeftLeg, RightLeg, LeftFoot, RightFoot, true);  //Set the servo pins
  //Bipedal_Robot.setTrims(calibratePosition[0], calibratePosition[1], calibratePosition[2],
calibratePosition[3]);
  calib_homePos();
  Bipedal_Robot.saveTrimsOnEEPROM();
  EEPROM.commit();
  Bipedal_Robot.home();
  IrReceiver.begin(IR_Pin, DISABLE_LED_FEEDBACK);  // Start the receiver
  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_Detection();
  Emotion_Show(emotion_task_mode);  //Led matrix display function
  WS2812_Show(ws2812_task_mode);    //Car color lights display function
}

void setup1() {
}
void loop1() {
  Move(move_flag);
}

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

```
76    switch (value) {
77      case 0xBF40FF00:  // Receive the number '+'
78        move_flag = 1;
79        break;
80      case 0xE619FF00:  // Receive the number '-'
81        move_flag = 2;
82        break;
83      case 0xF807FF00:  // Receive the number '|<<'
84        move_flag = 3;
85        break;
86      case 0xF609FF00:  // Receive the number '>>|'
87        move_flag = 4;
88        break;
89      case 0xEA15FF00:  // Receive the number '▶'
90        move_flag = 0;
91        break;
92      case 0xE916FF00:  // Receive the number '0'
93        break;
94      case 0xF30CFF00:  // Receive the number '1'
95        break;
96      case 0xF708FF00:  // Receive the number '4'
97        break;
98      case 0xF20DFF00:  // Receive the number 'C'
99        break;
100     case 0xA15EFF00:  // Receive the number '3'
101       move_flag = 5;
102       break;
103     case 0xA55AFF00:  // Receive the number '6'
104       move_flag = 6;
105       break;
106     case 0xB54AFF00:  // Receive the number '9'
107       break;
108     case 0xBB44FF00:  // Receive the number 'TEST'
109       Buzzer_Variable(2000, 100, 1);
110       break;
111     case 0xE718FF00:  // Receive the number '2'
112       emotion_flag = emotion_flag + 1;
113       if (emotion_flag > 7) {
114         emotion_flag = 0;
115       }
116       Emotion_SetMode(emotion_flag);  //Display
117       break;
118     case 0xE31CFF00:  // Receive the number '5'
119       emotion_flag = 0;
```

```
120         Emotion_SetMode(emotion_flag);
121         break;
122       case 0xBD42FF00:  // Receive the number '7'
123         ws2812_flag = ws2812_flag + 1;
124         if (ws2812_flag >= 6) {
125           ws2812_flag = 0;
126         }
127         WS2812_SetMode(ws2812_flag);
128         break;
129       case 0xAD52FF00:  // Receive the number '8'
130         ws2812_flag = 0;
131         WS2812_SetMode(ws2812_flag);
132         break;
133       case 0xFFFFFFFF:  // Remain unchanged
134         break;
135       default:
136         break;
137     }
138 }
139 
140 void Move(int data) {
141   // Handle the commands
142   switch (data) {
143     case 1:  // Receive the number '+'
144       Bipedal_Robot.walk(1, 1500, 1);
145       break;
146     case 2:  // Receive the number '-'
147       Bipedal_Robot.walk(1, 1500, -1);
148       break;
149     case 3:  // Receive the number '|<<'
150       Bipedal_Robot.turn(1, 1500, 1);
151       break;
152     case 4:  // Receive the number '>>|'
153       Bipedal_Robot.turn(1, 1500, -1);
154       break;
155     case 5:  // Receive the number '>>|'
156       Ultrasonic_Avoid();
157       break;
158     case 6:  // Receive the number '>>|'
159       dance();
160       break;
161     case 0:  // Receive the number '▶'
162       Bipedal_Robot.home();
163       break;
```

```
164       default:
165         move_flag = 0;
166         Bipedal_Robot.home();
167         break;
168     }
169   }
170
171   void dance() {
172   }
```

After the code uploads successfully, 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 robot	3	A15EFF00	Ultrasonic obstacle avoidance mode
TEST	BB44FF00	Control the buzzer	6	A55AFF00	Dancing Mode

Code Explanation:

Add the header file for the robot.

```
#include <Arduino.h>
#include <Adafruit_NeoPixel.h>
#include <IRremote.hpp>
#include <EEPROM.h>
#include "Freenove_Robot_For_Pico_W.h"
#include "Freenove_Robot_Emotion.h"
#include "Freenove_VK16K33_Lib.h"
#include "Freenove_Robot_WS2812.h"
#include "Bipedal_Robot.h"
```

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'
      ...
    case 0xA55AFF00:  // Receive the number '6'
      ...
    case 0xB54AFF00:  // Receive the number '9'
      ...
    case 0xBB44FF00:  // Receive the number 'TEST'
      ...
    case 0xE718FF00:  // Receive the number '2'
      ...
```

```
    case 0xE31CFF00:  // Receive the number '5'
      ...
    case 0xBD42FF00:  // Receive the number '7'
      ...
    case 0xAD52FF00:  // Receive the number '8'
      ...
    case 0xFFFFFFFF:  // Remain unchanged
      break;
    default:
      break;
  }
}
```

Sketch 04.4_Multi_Functional_Robot.ino is almost the same as 04.3_Multi_Functional_Robot.ino, except that it is added with the music play function. You can find the function of each key in the table below.

ICON	KEY Value	Function	ICON	KEY Value	Function
	BF40FF00	Move forward		E718FF00	Change the expression display mode
	F807FF00	Turn left		E31CFF00	Turn off emoticons
	F609FF00	Turn light		BD42FF00	Change the display mode of the WS2812
	E619FF00	Move back		AD52FF00	Turn off WS2812 display
	EA15FF00	Stop the robot		A15EFF00	Ultrasonic obstacle avoidance mode
	BB44FF00	Control the buzzer		A55AFF00	Dancing Mode
	E916FF00	Playing Music 1		F30CFF00	Playing Music 2
	F708FF00	Stop playing			

Chapter 5 Bluetooth remote control robot

This section requires an Android or iPhone device with Bluetooth to control the robot.
If you have any concerns, please feel free to contact us via support@freenove.com

5.1 Bluetooth Sending and Receiving Data

Bluetooth Module

The Bluetooth module is a serial port transparent transmission module that supports Bluetooth Low Energy (BLE).

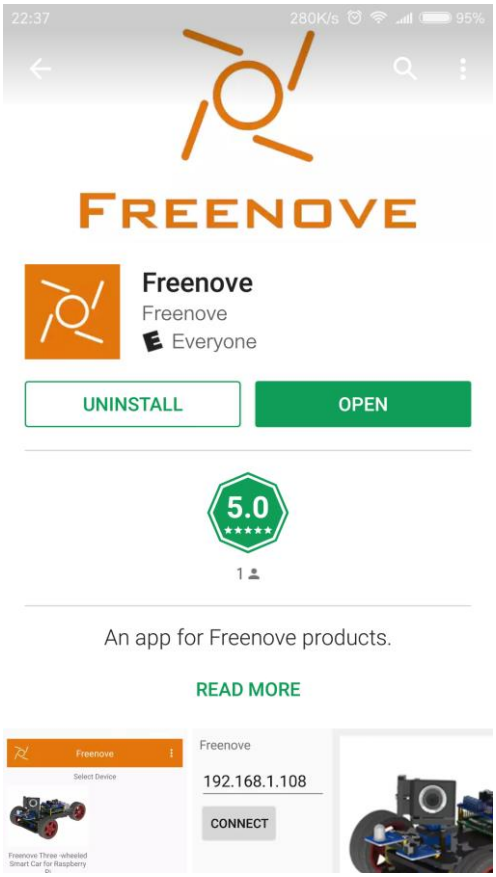
It has such advantages as small size, high performance, high-cost performance, low power consumption, and strong platform compatibility. The Bluetooth module we use is as shown below:

Install Freenove app

There are three ways to install our app.

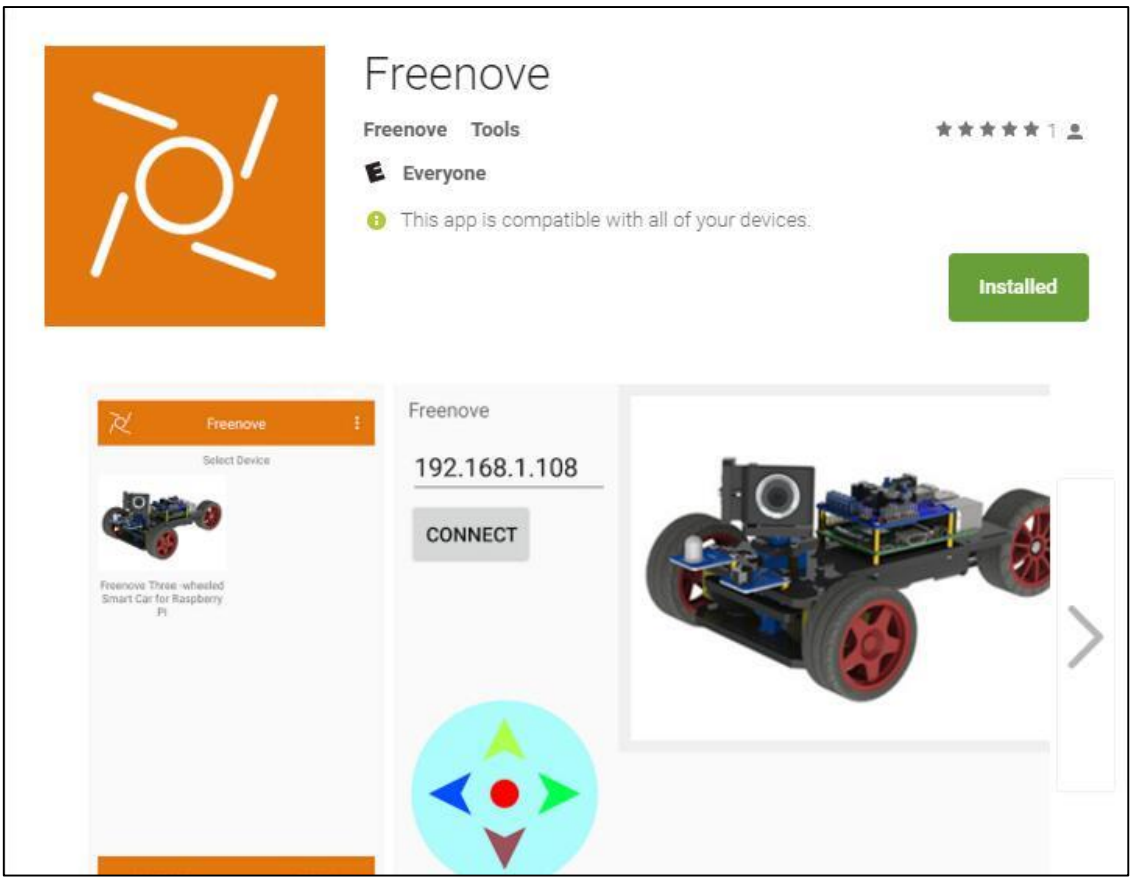
Method 1

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


22:37
280K/s
95%
FREENOVE
Freenove
Freenove
Everyone
UNINSTALL
OPEN
5.0
1
An app for Freenove products.
READ MORE
Freenove
Select Device
Freenove
192.168.1.108
CONNECT

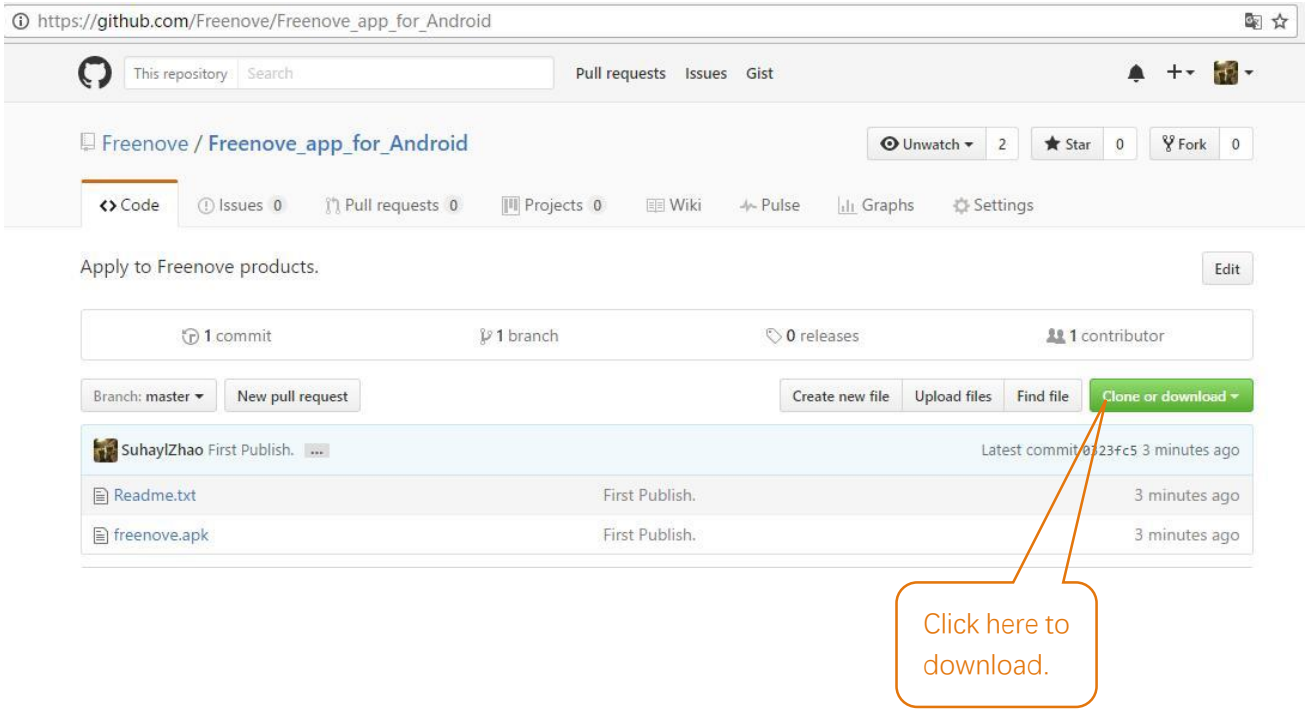
Method 2

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.



Instructions for Using the Bluetooth Module

The Bluetooth module communicates with the Raspberry Pi Pico (W) through serial port, and the program is also uploaded to Pico (W) via the serial port. **Therefore, when uploading code to pico (W), if it fails, please remove the Bluetooth module and upload again.**

When the Bluetooth module is not connected by other device, it can be configured using the AT command. Once connected, the Bluetooth module acts as a data pipe and cannot be configured.

By default, the Bluetooth module has a baud rate of 115200, no parity, 8 data bits, and 1 stop bit. Bluetooth name "BT05", role mode is slave mode.

Set name of bluetooth

If you have multiple Bluetooth modules with the same name around you, you will be confused when you connect. Which one is the Bluetooth module I want to connect to?

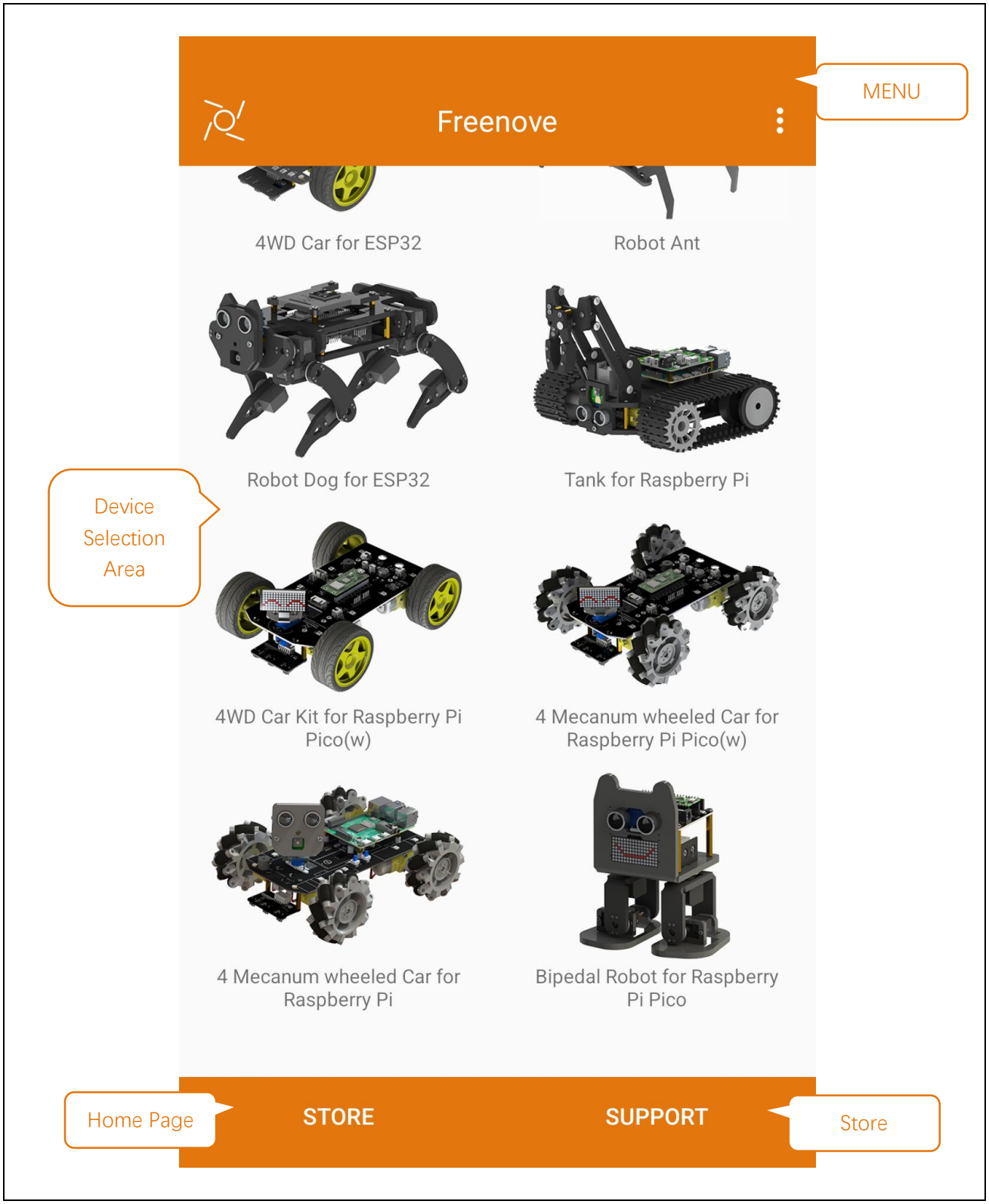
In the next project, we will introduce how to use the AT command to modify the name of the Bluetooth module and the master-slave role in the program.

For AT commands and more information about the Bluetooth module, refer to the documentation in the package for Datasheets/BT05-Instruction.pdf

This project uses the AT command to make some settings for the Bluetooth module, then uses the Bluetooth module to receive data from the app and print the data on the serial monitor.

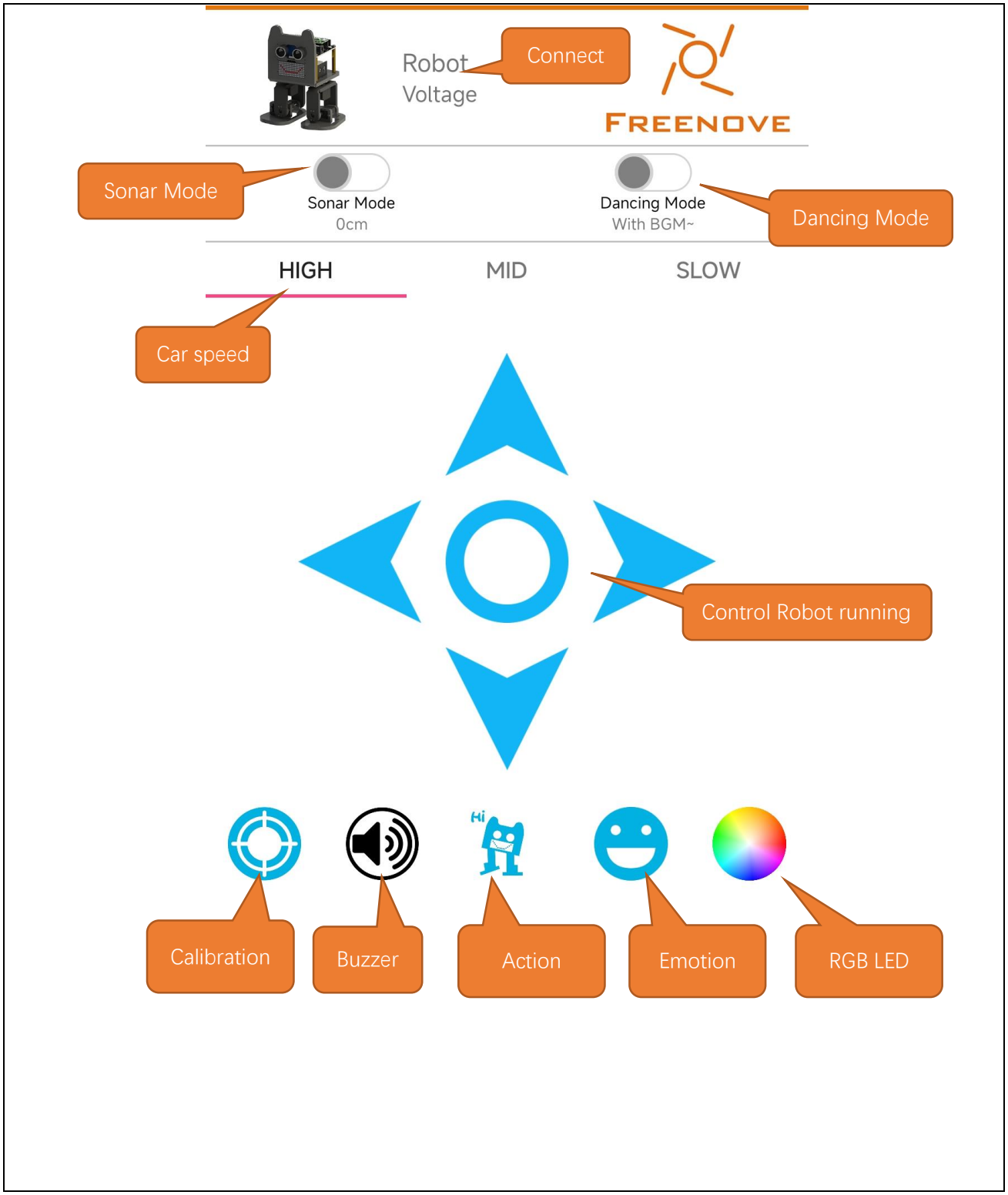
Menu

Open application “Freenove”, as shown below:



Introduction to the APP

In this chapter, we use the Freenove Bipedal robot for Raspberry Pi Pico (W) in the app, so let's learn its interface first, as shown below:



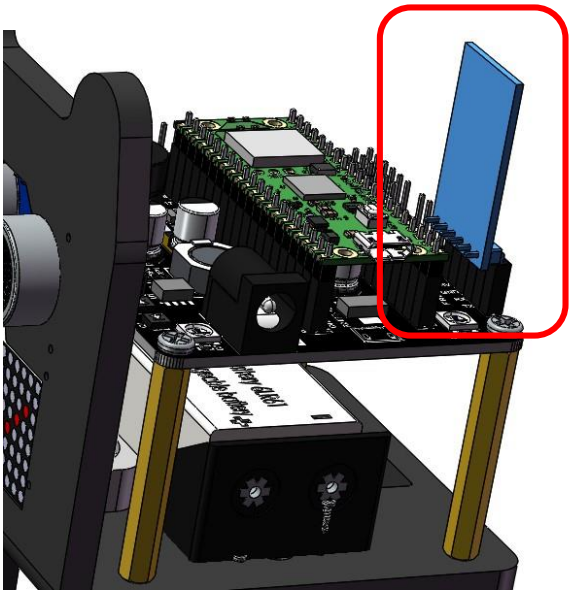
Sketch

Open "05.1_Bluetooth_Robot" in **"Freenove_Bipedal_Robot_Kit_for_Raspberry_Pi_Pico\Sketches"** and double click "05.1_Bluetooth_Robot.ino".

Note: if the code fails to upload, please remove the Bluetooth and upload it again.

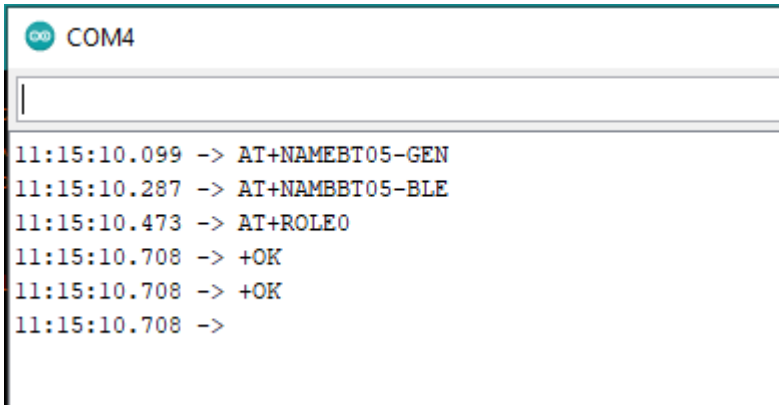
After the code is successfully uploaded, follow the steps below.

1. Plug the Bluetooth module to thecar as shown below. **Don't reverse it. The wrong connection may damage your hardware.**

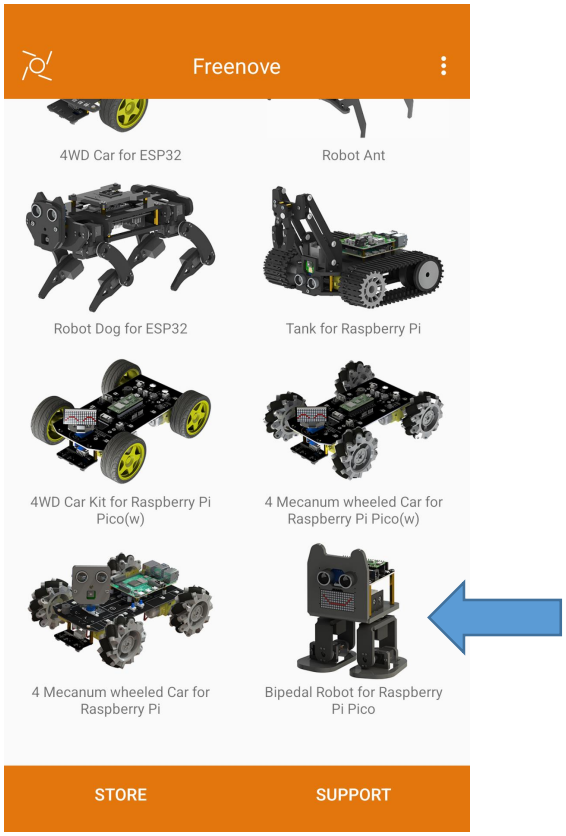


2. Open Arduino Serial Monitor, and reset the control board.

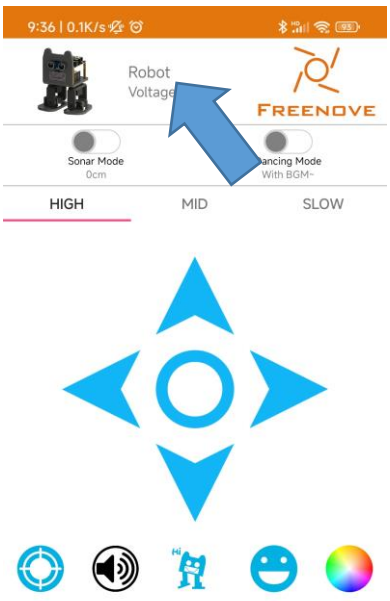
As shown in the figure below, the Bluetooth name is set to "BT05" and the Bluetooth mode is slave mode (ROLE=0).



Next, open Freenove APP, tap e Freenove Bipedal Robot for Raspberry Pi Pico.

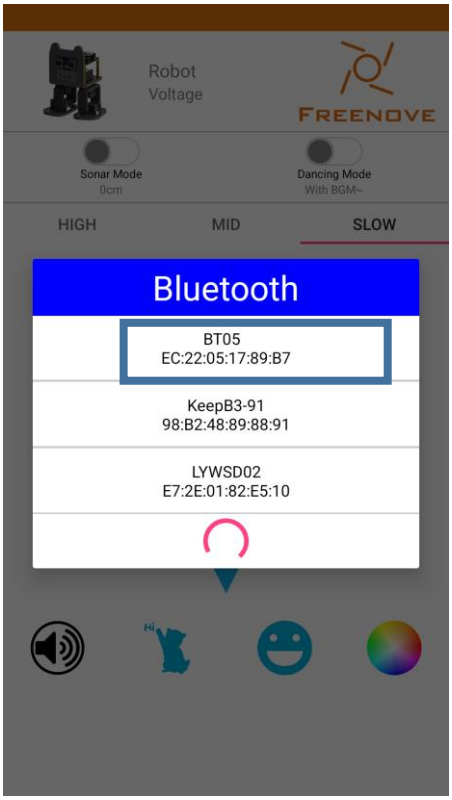


Click the following icon.

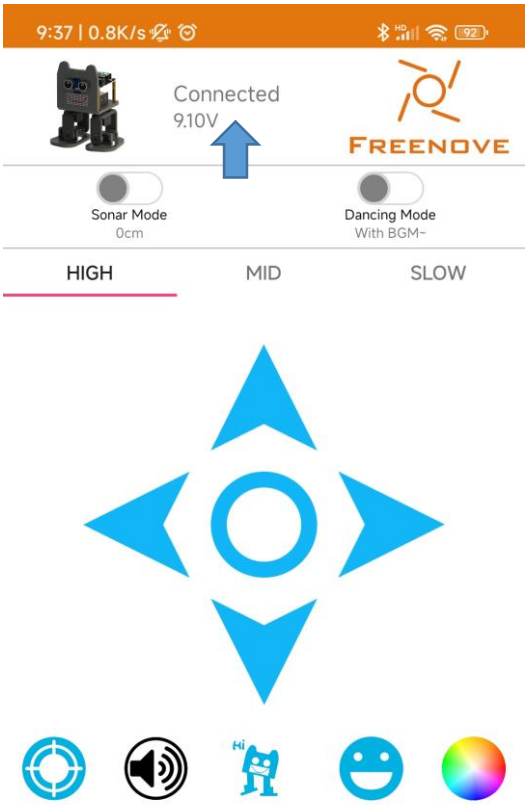


If the connection successes, you can see “Connected”.

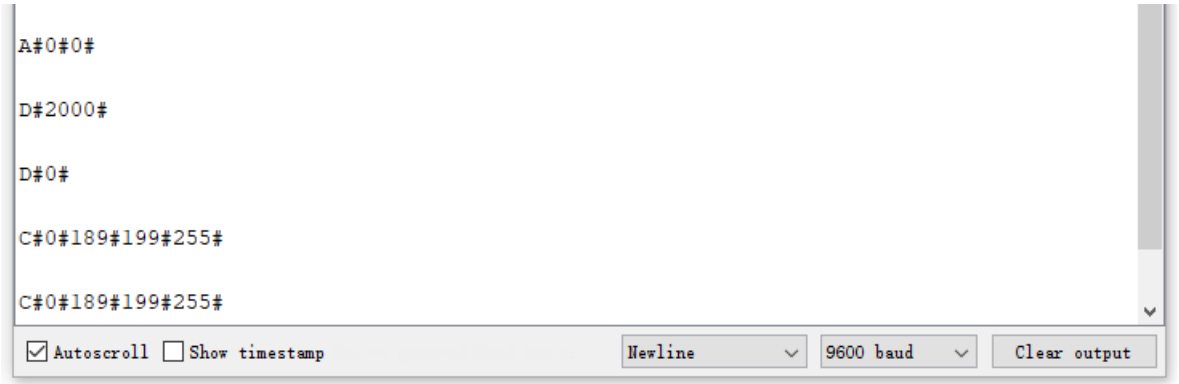
Click BT05.



Operate the app, and observe the monitor.



The monitor will show contents as below:



Bluetooth data- robot action

The command format for communication between app and robot is A#xxx#xxx#...xxx#, where # is a separator, the first character A represents the action command, it can be other characters, such as B, C, D…The xxx represents the parameters of the action command. And different commands carry different parameters. The command list is as below:

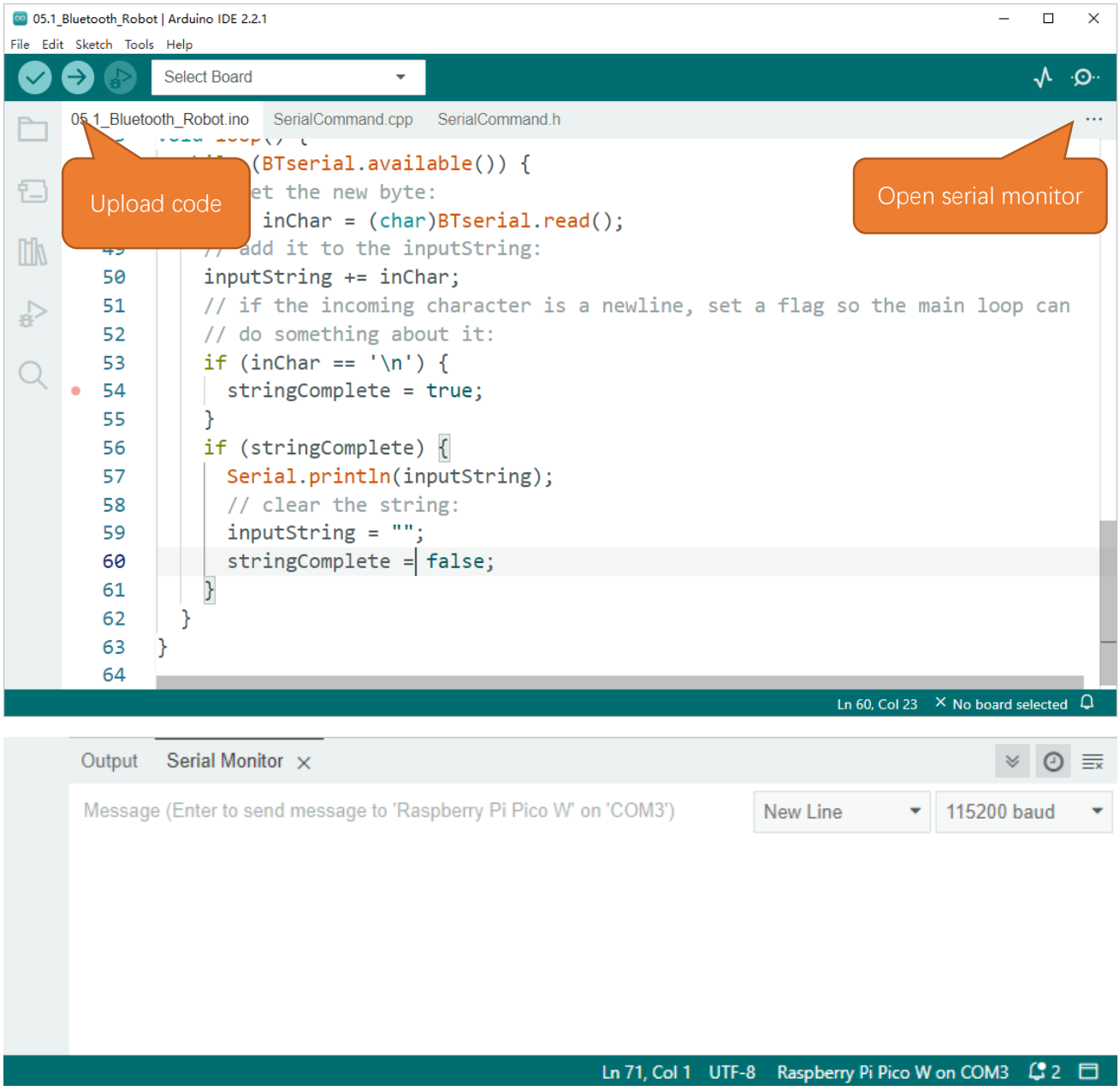
Action command	Description	Command character	Number of parameters (app send/receive)	Format example (app send/receive)
MOVE	Under the MOVE mode, parameter 1 represents the movement mode, and parameter 2 indicates the moving speed.	A	2	A#100#100#
VOLTAGE	Get the batteries voltage, parameter is the voltage value, unit mV	P	0/1	P# /P#4100#
LED_RGB	Control RGBLED, parameters respectively are serial number of LED mode, red value, green value, blue value.	C	4	C#2#100#150#200#
BUZZER	Control buzzer, and parameter is the frequency.	D	1	D#2000#

Code

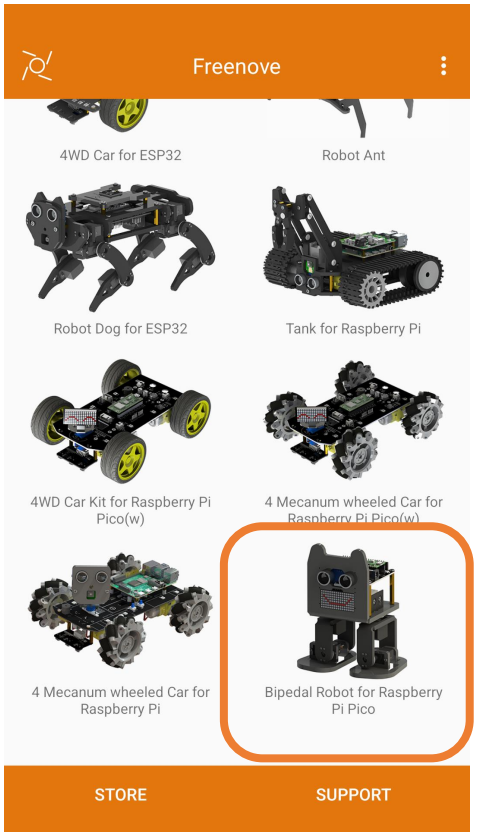
```
#include <Arduino.h>
#include <SoftwareSerial.h>  // Reference library

#include "SerialCommand.h"
SoftwareSerial BTserial = SoftwareSerial(1, 0);  // RX pin to 0 and TX pin to 1 on the board
SerialCommand SCmd(BTserial);

String inputString = "";    // a String to hold incoming data
bool stringComplete = false;// whether the string is complete

void receiveStop() {
  delay(10);
  Serial.flush();
}

void setup() {
  Serial.begin(115200);
  BTserial.begin(115200);
  BTserial.println("AT+UART?\r\n");
```

```
  delay(200);
  BTserial.println("AT+UART=115200\r\n");
  delay(200);
  BTserial.println("AT+NAME=BT05\r\n");//Set the radio called Robot
  delay(200);
  // BTserial.println(" "AT+ROLE=2\r\n" ");     //Set Bluetooth to slave mode.
  // BTserial.println("AT+RESET\r\n");     //Reset (restart)

  SCmd.addDefaultHandler(receiveStop);
  inputString.reserve(200);
  Serial.println("Set the name of the Bluetooth module to BT05.");
  Serial.println("Set the Bluetooth baud rate to 115200.");
  // Serial.println("Set Bluetooth to slave mode.");
}


void loop() {
  while (BTserial.available()) {
    // get the new byte:
    char inChar = (char)BTserial.read();
    // add it to the inputString:
    inputString += inChar;
    // if the incoming character is a newline, set a flag so the main loop can
    // do something about it:
    if (inChar == '\n') {
      stringComplete = true;
    }
    if (stringComplete) {
      Serial.println(inputString);
      // clear the string:
      inputString = "";
      stringComplete = false;
    }
  }
}
```

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



Open Freenove app, and tap the bipedal robot for Raspberry Pi Pico (W).



Code Explanation:

Configure Bluetooth, set the Bluetooth name, baud rate and other parameters.

```
Serial.begin(115200);
BTserial.begin(115200);
BTserial.println("AT+UART?\r\n");
delay(200);
BTserial.println("AT+UART=115200\r\n");
delay(200);
BTserial.println("AT+NAME=BT05\r\n");//Set the radio called Robot
delay(200);
```

After the Bluetooth module connects successfully, print the data it receives.

```
while (BTserial.available()) {
  // get the new byte:
  char inChar = (char)BTserial.read();
  // add it to the inputString:
  inputString += inChar;
  // if the incoming character is a newline, set a flag so the main loop can
  // do something about it:
  if (inChar == '\n') {
    stringComplete = true;
  }
  if (stringComplete) {
    Serial.println(inputString);
```

```
        // clear the string:
        inputString = "";
        stringComplete = false;
    }
}
```

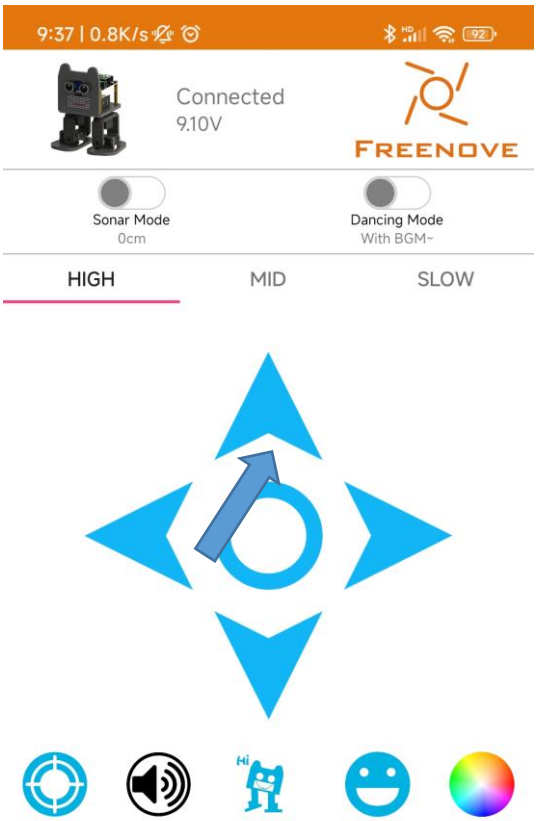
5.2 Bluetooth Remote Robot

In the previous section, we have learned how to receive the data of the app via Bluetooth, and the meanings of the data format. Next, let's use the data to control the robot to make some basic movements, like moving forward and backward, turning left and right, etc.

Sketch

Open the folder "05.2_Bluetooth_Remote_Robot" in Freenove_Bipedal_Robot_Kit_for_Raspberry_Pi_Pico\Sketches and double click "05.2_Bluetooth_Remote_Robot.ino".

As mentioned above, connect the APP to the ribot via Bluetooth, and then control the robot by taping or sliding the opration panel on the APP.



Code

```
#include <Arduino.h>
#include <SoftwareSerial.h>
#include <EEPROM.h>
#include "Freenove_Robot_For_Pico_W.h"

#include "SerialCommand.h"
SoftwareSerial BTserial = SoftwareSerial(1, 0);  // RX pin to 1 and TX pin
to 0 on the board
SerialCommand SCmd(BTserial);
#include "Bipedal_Robot.h"
Bipedal_Robot Bipedal_Robot;

#define LeftLeg 10
#define RightLeg 12
#define LeftFoot 11
#define RightFoot 13

#define Buzzer 21

int YL;
int YR;
int RL;
int RR;

int T = 1000;
int moveId = 0;
bool isServoResting = true;

extern int moveId;
extern bool isServoResting;

int moveSize = 15;

void receiveStop() {
  sendAck();
  sendFinalAck();
}

void recieveBuzzer() {
  sendAck();
  bool error = false;
  int frec;
  int duration;
```

```
  char *arg;
  arg = SCmd.next();
  if (arg != NULL) frec = atoi(arg);
  else error = true;
  Serial.println("\nfrec is:");
  Serial.println(frec);
  tone(20, frec);
  sendFinalAck();
}

void receiveMovement() {
  sendAck();
  Serial.print("move loop...");
  if (Bipedal_Robot.getRestState() == true)
Bipedal_Robot.setRestState(false);
  isServoResting = Bipedal_Robot.getRestState();
  char *arg;
  arg = SCmd.next();
  if (arg != NULL) {
    moveId = atoi(arg);
    Serial.print("moveId:");
    Serial.print(moveId);
  } else {
    moveId = 0;
  }
  arg = SCmd.next();
  if (arg != NULL) {
    T = atoi(arg);
    if (T == 0) {
      T = 1000;
    } else if (T == 1) {
      T = 1500;
    } else if (T == 2) {
      T = 2000;
    }
  } else T = 2000;
  Serial.print("T is:");
  Serial.print(T);
  arg = SCmd.next();
  if (arg != NULL) moveSize = atoi(arg);
  else moveSize = 15;
}
void move(int moveId) {
  // bool manualMode = false;
```

```
88    switch (moveId) {
89      case 0:
90        Bipedal_Robot.home();
91        // Serial.print("moveId 0 home");
92        break;
93      case 1:
94        Bipedal_Robot.walk(1, T, 1);
95        break;
96      case 2:
97        Bipedal_Robot.walk(1, T, -1);
98        break;
99      case 3:
100       Bipedal_Robot.turn(1, T, 1);
101       break;
102     case 4:
103       Bipedal_Robot.turn(1, T, -1);
104       break;
105     default:
106       break;
107   }
108 }
109
110 void sendAck() {
111   delay(10);
112   Serial.flush();
113 }
114 void sendFinalAck() {
115   delay(10);
116   Serial.flush();
117 }
118
119 void setup() {
120   Serial.begin(9600);
121   BTserial.begin(9600);
122   //Initialize some sensors
123   Buzzer_Setup();  //Initialize the buzzer
124   EEPROM.begin(512);
125   Bipedal_Robot.init(LeftLeg, RightLeg, LeftFoot, RightFoot, true);  //Set
126 the servo pins
127
128   YL = EEPROM.read(0);
129   if (YL > 128) YL -= 256;
130   YR = EEPROM.read(1);
131   if (YR > 128) YR -= 256;
```

```
132   RL = EEPROM.read(2);
133   if (RL > 128) RL -= 256;
134   RR = EEPROM.read(3);
135   if (RR > 128) RR -= 256;
136   Bipedal_Robot.setTrims(YL, YR, RL, RR);
137   calib_homePos();
138   Bipedal_Robot.saveTrimsOnEEPROM();
139   Bipedal_Robot.home();
140   delay(100);
141   EEPROM.commit();
142   SCmd.addCommand("S", receiveStop);
143   SCmd.addCommand("D", recieveBuzzer);
144   SCmd.addCommand("A", receiveMovement);
145   SCmd.addDefaultHandler(receiveStop);
146   Bipedal_Robot.home();
147 }
148
149 void loop() {
150   SCmd.readSerial();
151 }
152
153 void calib_homePos() {
154   int servoPos[4];
155   servoPos[0] = 90;
156   servoPos[1] = 90;
157   servoPos[2] = 90;
158   servoPos[3] = 90;
159   Bipedal_Robot._moveServos(500, servoPos);
160   Bipedal_Robot.detachServos();
161 }
162
163 void setup1() {
164 }
165 void loop1() {
166   move(moveId);
167 }
```

Code Explanation:

Add the header files to drive the robot.

```
#include <Arduino.h>
#include <SoftwareSerial.h>
#include <EEPROM.h>
#include "Freenove_Robot_For_Pico_W.h"

#include "SerialCommand.h"
```

Upon receiving movement commands, the robot makes corresponfing actions.

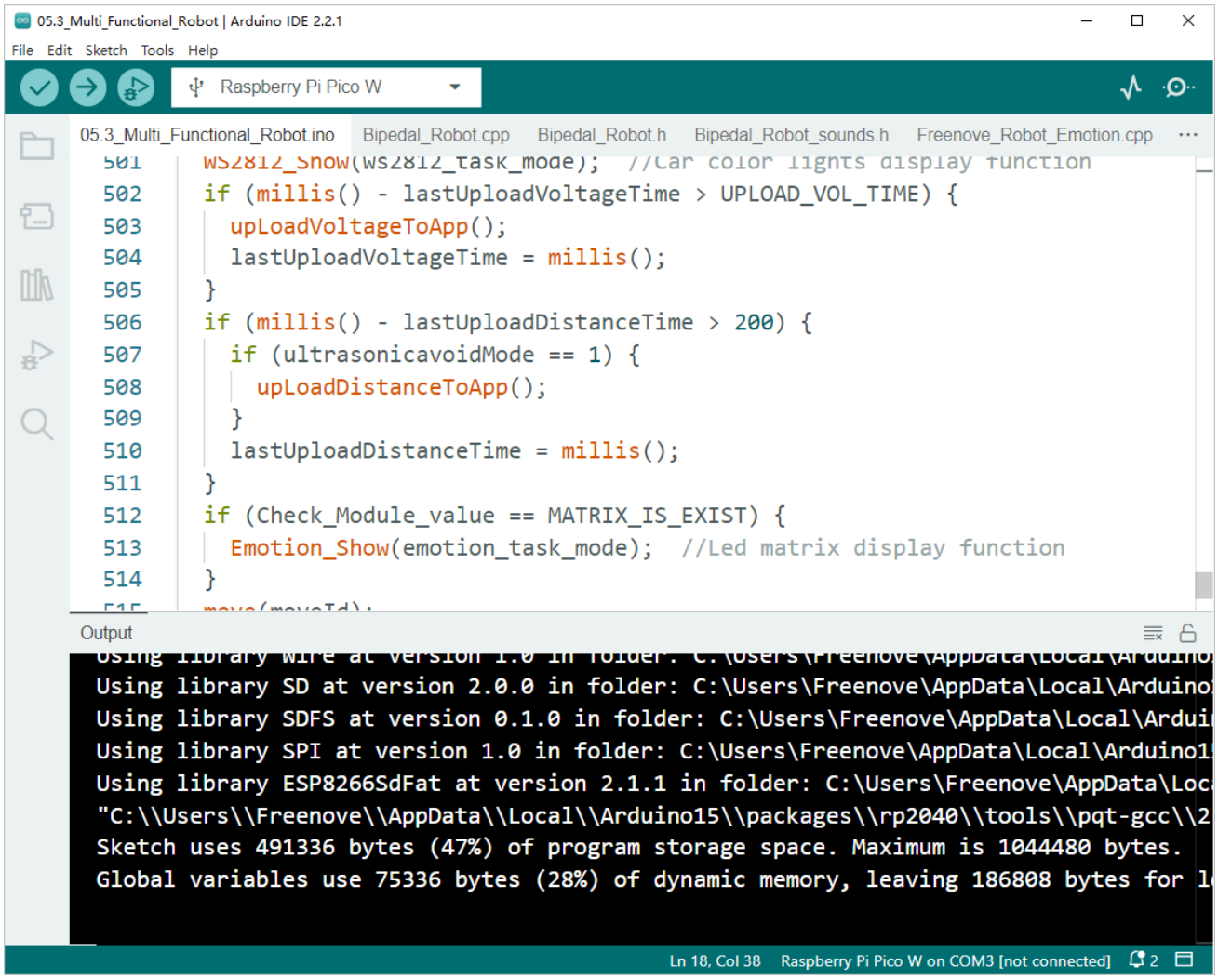
```
void move(int moveId) {
  // bool manualMode = false;
  switch (moveId) {
    case 0:
      Bipedal_Robot.home();
      // Serial.print("moveId 0 home");
      break;
    case 1:
      Bipedal_Robot.walk(1, T, 1);
      break;
    case 2:
      Bipedal_Robot.walk(1, T, -1);
      break;
    case 3:
      Bipedal_Robot.turn(1, T, 1);
      break;
    case 4:
      Bipedal_Robot.turn(1, T, -1);
      break;
    default:
      break;
  }
}
```

5.3 Multifunctional Bluetooth Remote Robot

In this section, we add more functions to the robot based on the previous section, making the robot more interested. The main functions include automatic obstacle avoidance, audio playing, LED displaying, and expressions on LED dot matrix.

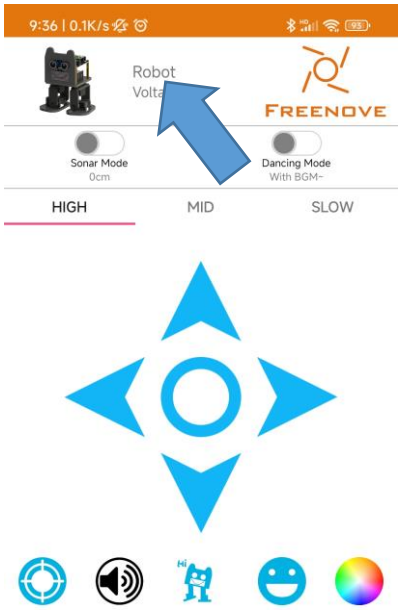
Upload Code and Running

Connect the robot to your computer with the USB cable.
As this section involves the audio playing function, please upload the audio data to the Raspberry Pi Pico (W) before uploading sketch. You can find the detailed steps here.
After uploading the audio file, you can upload sketch 05.3_Multi_Functional_Robot to the robot.
If the code fails to upload, please remove the Bluetooth module and upload again. Reconnect the Bluetooth module after the code uploads successfully.

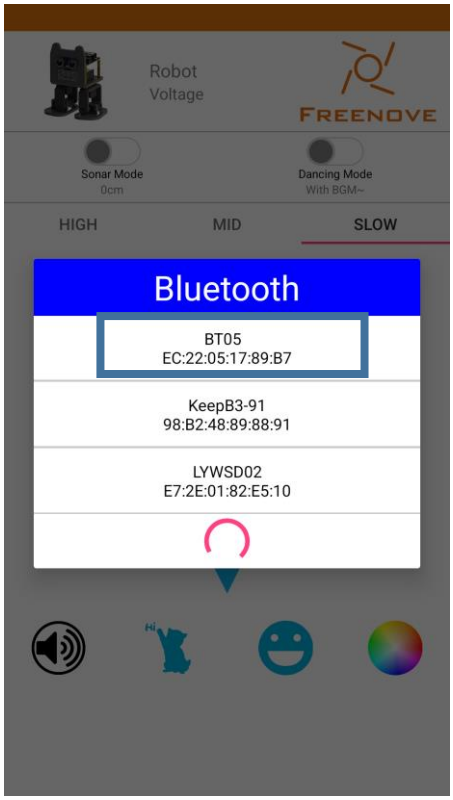


After the code is successfully uploaded, unplug the USB cable and plug the Bluetooth module to the robot. Turn on the robot power switch.

Click following icon.



Click BT05



Connect the APP with the robot through Bluetooth. Tap or swipe the operation panels on the app to control the robot's movements. The color palette of the lower right corner can be clicked to control the display mode and color of the LED. Among them,
Mode 0 is a flowing rainbow,
Mode 1 is a flowing water LED with color changing.
Mode 2 is a color-adjustable Blink.
Mode 3 displays the selected color of the current color picker for all LEDs.

Code

On the basis of the previous sketch, this one is added with more interesting functions.
05.3_Multi_Functional_Robot

```
#include <Arduino.h>
#include <EEPROM.h>
#include "Bipedal_Robot.h"
#include "AudioFileSourceLittleFS.h"
#include "AudioGeneratorMP3.h"
#include "AudioOutputI2SNoDAC.h"
#include "AudioFileSourceID3.h"
```

```
#include "AudioOutputI2S.h"
#include <SoftwareSerial.h>
#include "SerialCommand.h"
#include "Freenove_Robot_For_Pico_W.h"
#include "Freenove_Robot_Emotion.h"
#include "Freenove_Robot_WS2812.h"
#include "Freenove_VK16K33_Lib.h"

SoftwareSerial BTserial = SoftwareSerial(1, 0);  // RX pin to 0 and TX pin to 1 on the board
SerialCommand SCmd(BTserial);

Bipedal_Robot Bipedal_Robot;
AudioGeneratorMP3 *mp3;
AudioFileSourceLittleFS *file;
AudioOutputI2SNoDAC *out;
AudioFileSourceID3 *id3;

#define LeftLegPin 10
#define RightLegPin 12
#define LeftFootPin 11
#define RightFootPin 13
#define BuzzerPin 20

int motionFlag = 0;
int ultrasonicavoidMode = 0;
int playstartingmusic = 0;
int musicmode = 0;
int playsong = 0;
extern int playsong;  //Set the proportional coefficient
int T = 1000;
int moveId = 0;
int dance_steps = 0;
int Ultrasonic_Avoid_steps = 0;
bool isServoResting = true;
extern int moveId;
extern bool isServoResting;
int moveSize = 15;
#define ACTION_GET_VOLTAGE 'P'
#define ACTION_GET_DISTANCE 'E'
#define INTERVAL_CHAR '#'
unsigned long lastUploadVoltageTime;
unsigned long lastUploadDistanceTime;
#define UPLOAD_VOL_TIME 2000
```

```
void upLoadVoltageToApp() {
  float voltage = 0;
  voltage = Get_Battery_Voltage();
  BTserial.print("P#");
  BTserial.println(int(voltage * 1000));
}

void upLoadDistanceToApp() {
  float Distance = 0;
  Distance = Get_Sonar();
  BTserial.print("E#");
  BTserial.println(Distance);
}

void receiveStop() {
  sendAck();
  sendFinalAck();
}

void recieveBuzzer() {
  sendAck();
  moveId = 0;
  playsong = 0;
  bool error = false;
  int frec;
  char *arg;
  arg = SCmd.next();
  if (arg != NULL) frec = atoi(arg);
  else error = true;
  tone(2, frec);
  sendFinalAck();
}

void receiveAvoid() {
  sendAck();
  ws2812_task_mode = 0;
  emotion_task_mode = 0;
  if (Bipedal_Robot.getRestState() == true) Bipedal_Robot.setRestState(false);
  isServoResting = Bipedal_Robot.getRestState();
  char *arg;
  arg = SCmd.next();
  if (arg != NULL) {
    ultrasonicavoidMode = atoi(arg);
    playsong = 0;
```

```
    if (ultrasonicavoidMode == 1) {
      moveId = 21;
      Ultrasonic_Avoid_steps = 0;
      playsong = 0;
    }
    if (ultrasonicavoidMode == 2) {
      dance_steps = 0;
      moveId = 22;
      playsong = 4;
    }
    if (ultrasonicavoidMode == 0) {
      moveId = 0;
      playsong = 0;
    }
  }
  sendFinalAck();
}

void receiveEmotion() {
  sendAck();
  Emotion_Setup();
  moveId = 0;
  playsong = 0;
  char *arg;
  char *endstr;
  arg = SCmd.next();
  if (arg != NULL) {
    String stringOne = String(arg);  // converting a constant char into a String
    emotion_task_mode = stringOne.toInt();
    Emotion_SetMode(emotion_task_mode);
  } else {
  }
  sendFinalAck();
}

void receiveLED() {
  sendAck();
  moveId = 0;
  playsong = 0;
  char *arg;
  unsigned char paramters[3];
  char *endstr;
  arg = SCmd.next();
  if (arg != NULL) {
```

```
    String stringOne = String(arg);  // converting a constant char into a String
    ws2812_task_mode = stringOne.toInt();
  } else {
  }
  arg = SCmd.next();
  String stringOne1 = String(arg);
  paramters[0] = stringOne1.toInt();

  arg = SCmd.next();
  String stringOne2 = String(arg);
  paramters[1] = stringOne2.toInt();

  arg = SCmd.next();
  String stringOne3 = String(arg);
  paramters[2] = stringOne3.toInt();

  WS2812_Set_Color_1(15, paramters[0], paramters[1], paramters[2]);
  sendFinalAck();
}

void songmusic() {
  if (playsong > 0) {
    playmusic(1, playsong);
    if (playsong != 4) {
      playsong = 0;
    }
  } else {
    pinMode(6, OUTPUT);
    digitalWrite(6, LOW);
  }
}
void receiveMusic() {
  int music_song;
  sendAck();
  ws2812_task_mode = 0;
  emotion_task_mode = 0;
  if (Bipedal_Robot.getRestState() == true) Bipedal_Robot.setRestState(false);
  isServoResting = Bipedal_Robot.getRestState();
  char *arg;
  arg = SCmd.next();
  if (arg != NULL) {
    music_song = atoi(arg);
    playsong = music_song;
    playstartingmusic = music_song;
```

```
    clearEmtions();
    if (playsong == 1) {
      moveId = 18;
    } else if (playsong == 2) {
      moveId = 19;
    } else if (playsong == 3) {
      moveId = 20;
    } else if (playsong == 0) {
      moveId = 0;
    }
  } else {
    moveId = 0;
  }
  T = 2200;
}

void playmusic(int playtimes, int song) {
  if (playtimes--) {
    if (song == 1) {
      file = new AudioFileSourceLittleFS("Hello.mp3");
    } else if (song == 2) {
      file = new AudioFileSourceLittleFS("Nicetomeetyou.mp3");
    } else if (song == 3) {
      file = new AudioFileSourceLittleFS("goodbye.mp3");
    } else if (song == 4) {
      file = new AudioFileSourceLittleFS("1.mp3");
    } else if (song == 5) {
      file = new AudioFileSourceLittleFS("onfoot.mp3");
    } else if (song == 6) {
      file = new AudioFileSourceLittleFS("hello_cn.mp3");
    } else if (song == 0) {
      file = new AudioFileSourceLittleFS("split.mp3");
    }
    out = new AudioOutputI2SNoDAC(6);
    out->SetGain(4.0);  //Volume Setup (0-4.0)
    mp3 = new AudioGeneratorMP3();
    mp3->begin(file, out);
    if (mp3->isRunning() && song > 0) {
      if (!mp3->loop() && song > 0) {
        delete file;
        delete mp3;
        out->flush();
        out->stop();
      }
```

```
    } else {
      delete file;
      delete mp3;
      out->flush();
      out->stop();
      pinMode(6, OUTPUT);
      digitalWrite(6, LOW);
    }
  }
}

//Ultrasonic robot
void Ultrasonic_Avoid() {
  if (Get_Sonar() <= 15) {
    Serial.println(Get_Sonar());
    Bipedal_Robot.walk(2, 1500, -1);
    Bipedal_Robot.turn(3, 1500, 1);
  }
  Bipedal_Robot.walk(1, 2000, 1);
}

int avoidanceflag =0;

void avoidance(){
    if (ultrasonicavoidMode == 1) {
      Serial.println("ultrasonicavoidMode... ");
      Serial.println(ultrasonicavoidMode);
      Ultrasonic_Avoid_steps = Ultrasonic_Avoid_steps + 1;
      switch (Ultrasonic_Avoid_steps) {
        case 0:
          Ultrasonic_Avoid_steps = 0;
          break;
        case 1:
          Serial.println("steps1");
          Bipedal_Robot.walk(1, 1500, -1);
          break;
        case 2:
          Serial.println("steps2");
          Bipedal_Robot.walk(1, 1500, -1);  // BACKWARD x2
          break;
        case 3:
          Serial.println("steps3");
          Bipedal_Robot.turn(1, 1500, 1);
          break;
```

```
        case 4:
          Serial.println("steps4");
          Bipedal_Robot.turn(1, 1500, 1);
          break;
        case 5:
          Serial.println("steps5");
          Bipedal_Robot.turn(1, 1500, 1);  // LEFT x3
          avoidanceflag =0;
          break;
      }
      if (Ultrasonic_Avoid_steps > 6) {
        Ultrasonic_Avoid_steps = 0;
        avoidanceflag =0;
      }
      if (ultrasonicavoidMode != 1){
        Ultrasonic_Avoid_steps = 0;
      }
    }
}


void dance() {
  dance_steps = dance_steps + 1;
  staticEmtions(19);
  if (moveId > 0) {
    switch (dance_steps) {
      case 0:
        dance_steps = 0;
        break;
      case 1:
        Bipedal_Robot.jitter(1, 750, 20);
        break;
      case 2:
        Bipedal_Robot.jitter(1, 750, 20);
        break;
      case 3:
        Bipedal_Robot.crusaito(1, 800, 30, 1);
        break;
      case 4:
        Bipedal_Robot.crusaito(1, 800, 30, -1);
        break;
      case 5:
        Bipedal_Robot.crusaito(1, 800, 30, 1);
        delay(300);
```

```
        break;
      case 6:
        Bipedal_Robot.walk(1, 1500, -1);
        break;
      case 7:
        Bipedal_Robot.walk(1, 1000, 1);
        break;
      case 8:
        Bipedal_Robot.walk(1, 1000, 1);
        break;
      case 9:
        Bipedal_Robot.moonwalker(1, 600, 30, 1);
        break;
      case 10:
        Bipedal_Robot.moonwalker(1, 600, 30, -1);
        break;
      case 11:
        Bipedal_Robot.moonwalker(1, 600, 30, 1);
        break;
      case 12:
        Bipedal_Robot.moonwalker(1, 600, 30, -1);
        break;
      case 13:
        Bipedal_Robot.walk(1, 1500, -1);
        break;
    }
    if (dance_steps > 14) {
      dance_steps = 0;
    }
  } else {
    dance_steps = 0;
  }
}

void receiveMovement() {
  sendAck();
  ws2812_task_mode = 0;
  emotion_task_mode = 0;
  if (Bipedal_Robot.getRestState() == true) Bipedal_Robot.setRestState(false);
  isServoResting = Bipedal_Robot.getRestState();
  char *arg;
  arg = SCmd.next();
  if (arg != NULL) {
    moveId = atoi(arg);
```

```
    playsong = 0;
  } else {
    moveId = 0;
  }
  arg = SCmd.next();
  if (arg != NULL) {
    T = atoi(arg);
    if (T == 0) {
      T = 1500;
    } else if (T == 1) {
      T = 2000;
    } else if (T == 2) {
      T = 2500;
    }
  } else T = 2500;
  arg = SCmd.next();
  if (arg != NULL) moveSize = atoi(arg);
  else moveSize = 15;
}

void move(int movemode) {
  switch (movemode) {
    case 0:
      Bipedal_Robot.home();
      break;
    case 1:
      Bipedal_Robot.walk(1, T, 1);
      Serial.print(T);
      break;
    case 2:
      Bipedal_Robot.walk(1, T, -1);
      break;
    case 3:
      Bipedal_Robot.turn(1, T, 1);
      break;
    case 4:
      Bipedal_Robot.turn(1, T, -1);
      break;
    case 5:
      Bipedal_Robot.updown(1, 2000, 30);
      break;
    case 18:
      staticEmtions(21);
      Bipedal_Robot.swing(1, 1100, 50);  //1071
```

```
      Serial.print("music1");
      moveId = 0;
      clearEmtions();
      break;
    case 19:
      staticEmtions(22);
      Bipedal_Robot.moonwalker(1, 1550, 50, 1);  //1541
      Serial.print("music2");
      moveId = 0;
      clearEmtions();
      break;
    case 20:
      staticEmtions(23);
      Bipedal_Robot.jump(1, 1350);  //1332
      moveId = 0;
      clearEmtions();
      Serial.print("music3");
      break;
    case 21:
      Ultrasonic_Avoid();
      break;
    case 22:
      dance();
      break;
    default:
      break;
  }
}

void sendAck() {
  delay(5);
  Serial.flush();
}
void sendFinalAck() {
  delay(5);
  Serial.flush();
}

void calib_homePos() {
  int servoPos[4];
  servoPos[0] = 90;
  servoPos[1] = 90;
  servoPos[2] = 90;
  servoPos[3] = 90;
```

```
  Bipedal_Robot._moveServos(500, servoPos);
  Bipedal_Robot.detachServos();
}
void setup() {
  Serial.begin(9600);
  BTserial.begin(9600);
  EEPROM.begin(512);                                                    //Initialize
the ultrasonic module
  Bipedal_Robot.init(LeftLegPin, RightLegPin, LeftFootPin, RightFootPin, true);  //Set the
servo pins
  Bipedal_Robot.Buzzer_init(BuzzerPin);

  calib_homePos();
  Bipedal_Robot.saveTrimsOnEEPROM();
  EEPROM.commit();
  Bipedal_Robot.home();
  delay(50);
  Buzzer_Setup();  //Initialize the buzzer
  WS2812_Setup();  //WS2812 initialization
  Emotion_Setup();
  Ultrasonic_Setup();

  SCmd.addCommand("S", receiveStop);
  SCmd.addCommand("C", receiveLED);
  SCmd.addCommand("D", recieveBuzzer);
  SCmd.addCommand("A", receiveMovement);
  SCmd.addCommand("B", receiveEmotion);
  SCmd.addCommand("H", receiveAvoid);
  SCmd.addCommand("M", receiveMusic);
  SCmd.addDefaultHandler(receiveStop);
}

void loop() {
  SCmd.readSerial();
  Emotion_Detection();
  WS2812_Show(ws2812_task_mode);  //Car color lights display function
  if (millis() - lastUploadVoltageTime > UPLOAD_VOL_TIME) {
    upLoadVoltageToApp();
    lastUploadVoltageTime = millis();
  }
  if (millis() - lastUploadDistanceTime > 200) {
    if (ultrasonicavoidMode == 1) {
      upLoadDistanceToApp();
    }
```

```
    lastUploadDistanceTime = millis();
  }
  if (Check_Module_value == MATRIX_IS_EXIST) {
    Emotion_Show(emotion_task_mode);  //Led matrix display function
  }
  move(moveId);
}

void setup1() {
  playmusic(1, 2);
  pinMode(6, OUTPUT);
  digitalWrite(6, LOW);
}
void loop1() {
  songmusic();
}
```

Same as in the previous project, when the corresponding command is received by Bluetooth, the corresponding function is executed.

```
void loop() {
  SCmd.readSerial();
  Emotion_Detection();
  WS2812_Show(ws2812_task_mode);  //Car color lights display function
  if (millis() - lastUploadVoltageTime > UPLOAD_VOL_TIME) {
    upLoadVoltageToApp();
    lastUploadVoltageTime = millis();
  }
  if (millis() - lastUploadDistanceTime > 200) {
    if (ultrasonicavoidMode == 1) {
      upLoadDistanceToApp();
    }
    lastUploadDistanceTime = millis();
  }
  if (Check_Module_value == MATRIX_IS_EXIST) {
    Emotion_Show(emotion_task_mode);  //Led matrix display function
  }
  move(moveId);
}
```

The decription of the commands and their functions are as shwon in the table below:

Commands	Description
A#parm1#parm2	Controls the robot's movements.

B#parm1	Expressions control command
C#parm1#parm2#parm3#parm4	RGB LEDs control command
D#parm1	Buzzer control command
Send battery data every 25s	Send the battery level to the APP. Format: P#Battery voltage
M#parm1	Audio playing command
H#parm1	Ultrasonic obstacle avoidanve mode or dance mode
Send distance data under the obstacle avoidance mode	Send the ultrasonic data to the APP. Format: E# ultrasonic data

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.