

# 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](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](mailto: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

All the files, materials and instructional guides provided are released under [Creative Commons Attribution-NonCommercial-ShareAlike 3.0 Unported License](#). A copy of this license can be found in the folder containing the Tutorial and software files associated with this product.



This means you can use these resources in your own derived works, in part or completely, but **NOT for the intent or purpose of commercial use.**

Freenove brand and logo are copyright of Freenove Creative Technology Co., Ltd. and cannot be used without written permission.



Need support? ✉ [support.freenove.com](mailto:support.freenove.com)

# Contents

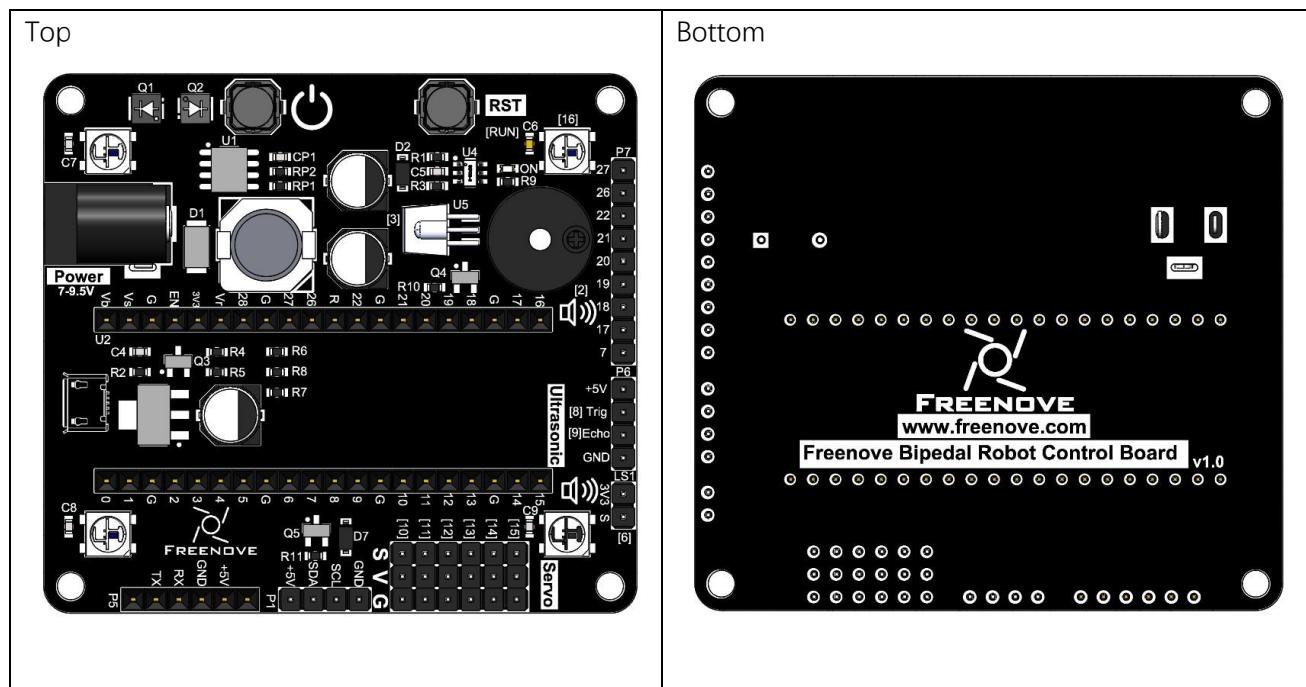
Welcome .....	1
Contents .....	1
List .....	2
Raspberry Pi Pico (W) Robot Shield.....	3
Machinery Parts .....	4
Acrylic Parts.....	5
Electronic Parts.....	5
Wires.....	6
Tools.....	7
Required but NOT Contained Parts.....	7
Preface .....	8
Raspberry Pi Pico.....	9
Raspberry Pi Pico W .....	12
Pins of the Robot.....	15
Introduction to the Bipedal Robot .....	16
Installation of Arduino IDE .....	17
Arduino Software .....	17
Environment Configuration .....	20
Additional Remarks .....	23
Uploading Arduino-compatible Firmware for Raspberry Pi Pico (W).....	24
Uploading the First Code .....	28
Chapter 0 Assembling Bipedal Robot .....	31
Assembling the Robot.....	31
Chapter 1 Module test.....	46
1.1 Servo .....	46
1.2 Buzzer .....	69
1.3 Loudspeaker .....	72
1.4 ADC Module .....	86
1.5 LED Matrix .....	90
1.6 LED .....	102
Chapter 2 Ultrasonic Obstacle Avoidance Robot.....	108
2.1 Ultrasonic Module .....	108
2.2 Obstacle Avoidance Robot .....	111
Chapter 3 Sing and Dance .....	113
3.1 Sing and dance .....	113
Chapter 4 Infrared Robot .....	119
4.1 Introduction of infrared reception function .....	119
4.2 Infrared Robot.....	124
4.3 Multi-Functional Infrared Robot.....	129
Chapter 5 Bluetooth remote control robot.....	137
5.1 Bluetooth Sending and Receiving Data.....	137

5.2 Bluetooth Remote Robot .....	147
5.3 Multifunctional Bluetooth Remote Robot.....	153
What's next?.....	169

# List

If you have any concerns, please feel free to contact us via [support@freenove.com](mailto:support@freenove.com)

## Raspberry Pi Pico (W) Robot Shield



## Machinery Parts

**M3\*8**

Screw

**x10**

Freenove

**M3\*18**

Screw

**x2**

Freenove

**M3**

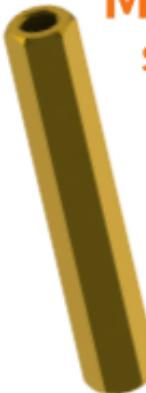
Nut

**x2**

Freenove

**M3\*35**

Standoff

**x4**

Freenove

**M2\*12**

Screw

**x14**

Freenove

**M2**

Nut

**x17**

Freenove

**M1.4\*5**self-tapping  
Screw**x10**

Freenove

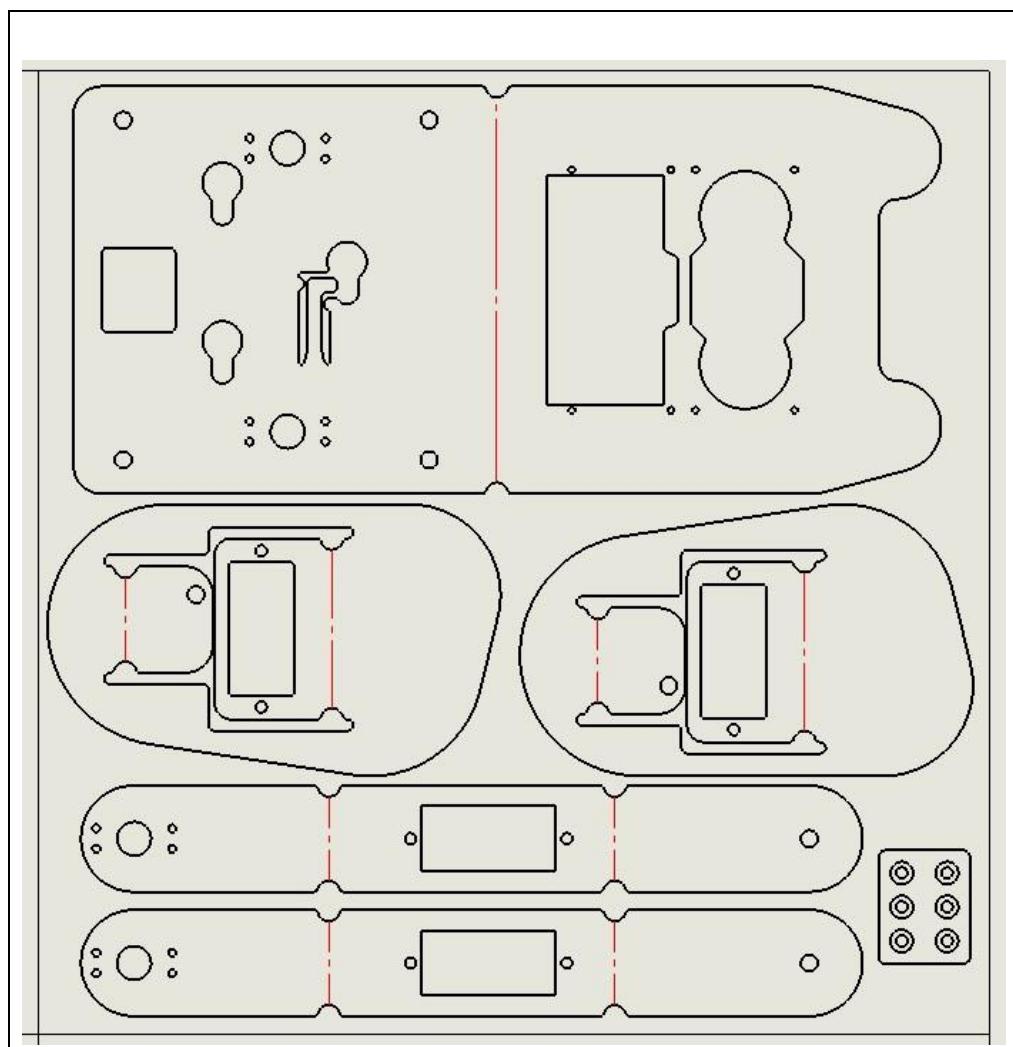
**M2\*6\*0.5**

Nut

**x4**

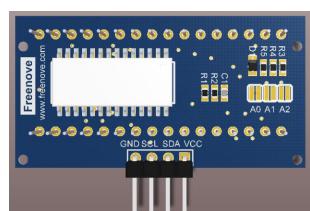
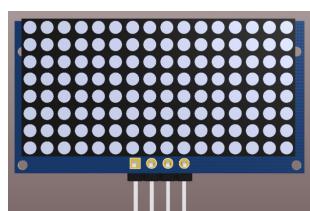
Freenove

## Acrylic Parts



## Electronic Parts

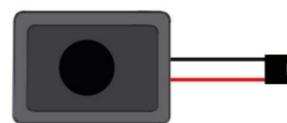
Dot Matrix Module x 1



Ultrasonic Module x 1



Speaker x 1



Servo package x 4



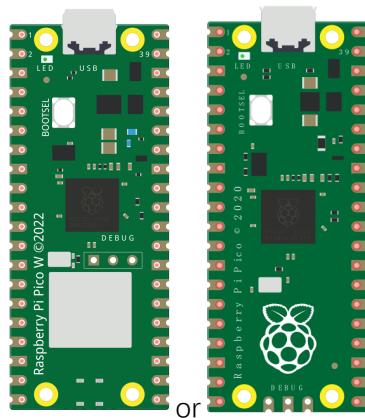
Bluetooth module x 1



Infrared emitter x 1



Raspberry Pi Pico x 1 or Raspberry Pi Pico W x 1



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

## Wires

Jumper Wire F/F(4 pin) x 2



Need support? ✉ [support.freenove.com](mailto:support.freenove.com)

## Tools

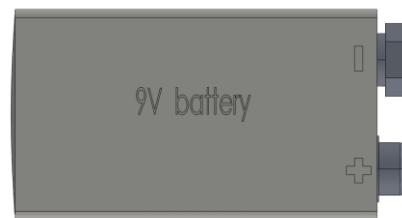
Cross screwdriver (3mm) x 1 	Cross screwdriver (2mm) x 1 	Cable Tidy x 40cm 
M3 spanner 		

## 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.

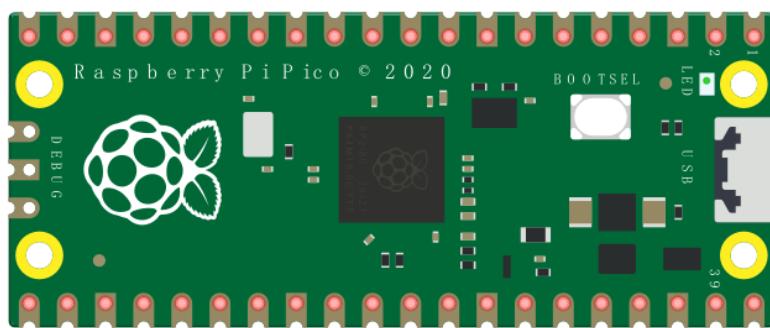
**[support@freenove.com](mailto:support@freenove.com)**

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:

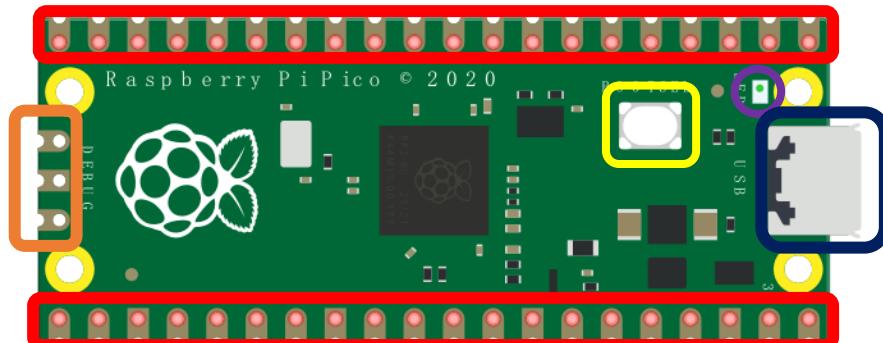
[www.freenove.com](http://www.freenove.com) or contact us to buy our kit designed for beginners: **Freenove Ultimate Kit for Raspberry Pi Pico.**

## Raspberry Pi Pico

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

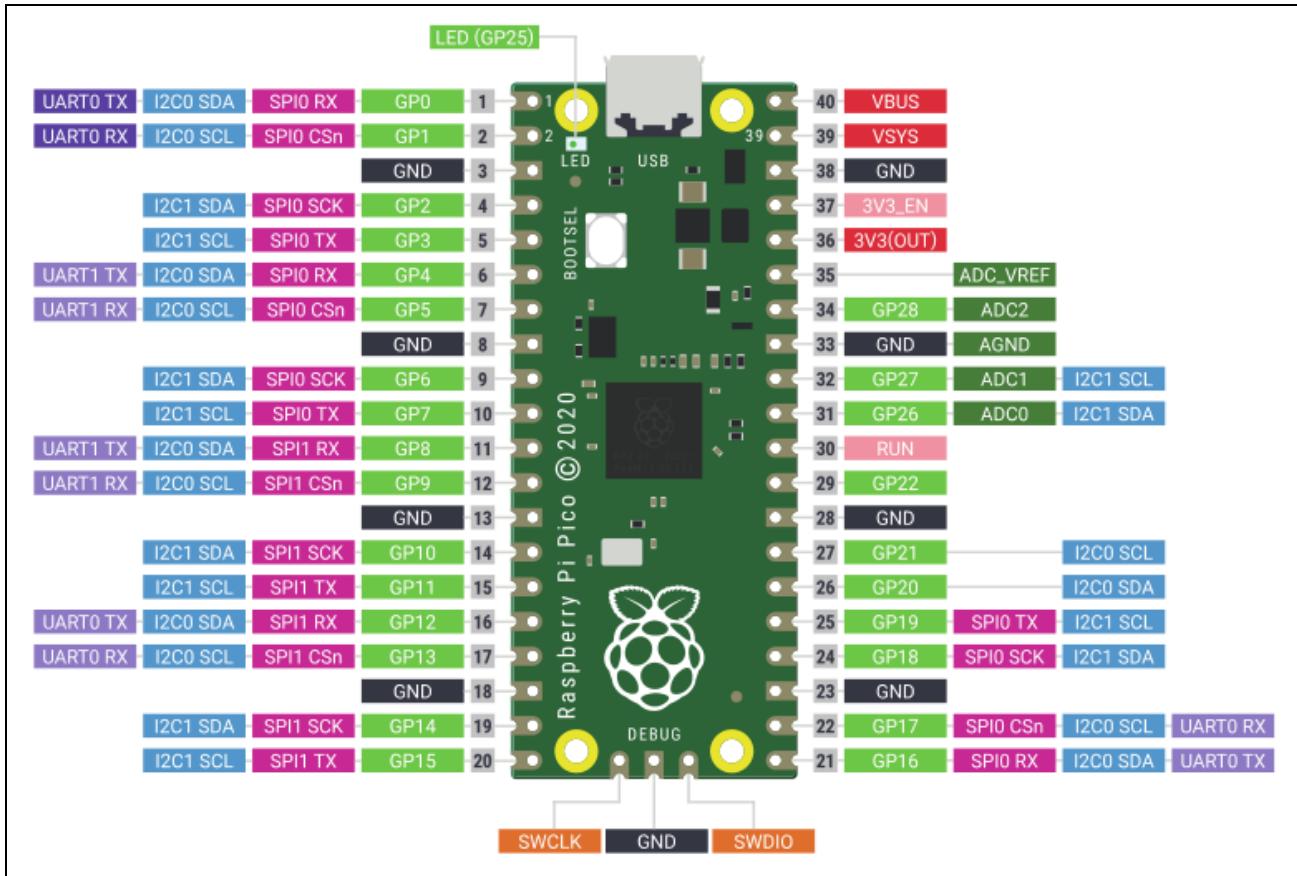


The hardware interfaces are distributed as follows:



Frame color	Description
	Pins
	BOOTSEL button
	USB port
	LED
	Debugging

Function definition of pins:



Color	Pins	Color	Pins
	GND		Power
	GPIO		ADC
	UART(default)		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

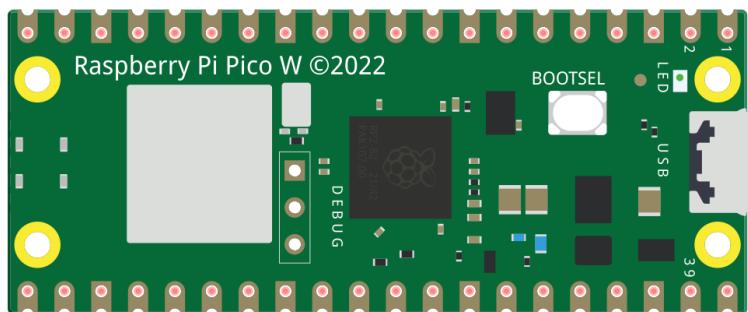
Function	Default
I2C Frequency	400000
I2C_SDA	Pin 4
I2C_SCL	Pin 5

### SPI

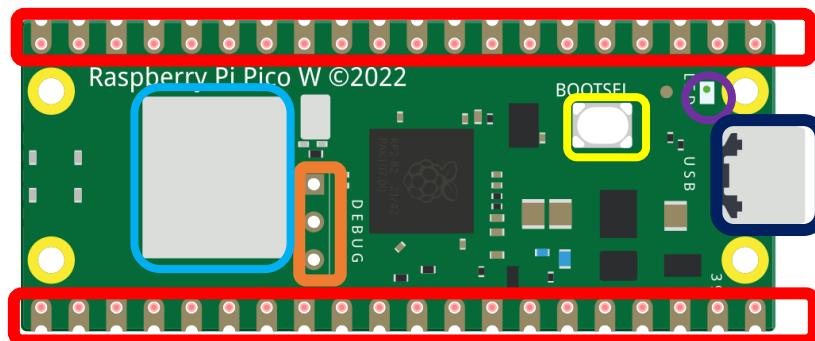
Function	Default
SPI_BAUDRATE	1000000
SPI_POLARITY	0
SPI_PHASE	0
SPI_BITS	8
SPI_FIRSTBIT	MSB
SPI_SCK	Pin 18
SPI_MOSI	Pin 19
SPI_MISO	Pin 16
SPI_SS	Pin 17

## Raspberry Pi Pico W

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

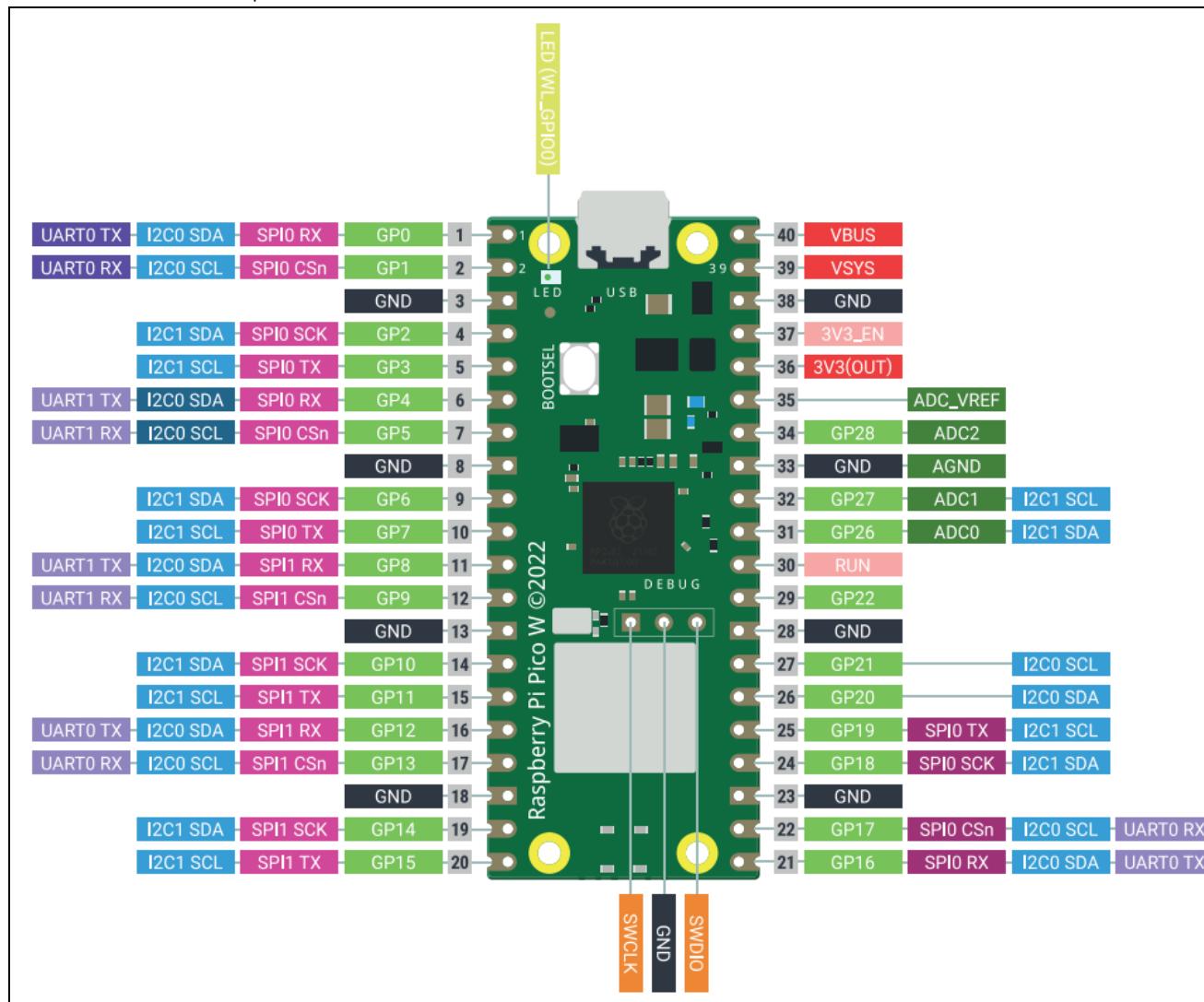


The hardware interfaces are distributed as follows:



Frame color	Description
	Pins
	BOOTSEL button
	USB port
	LED
	Debugging
	Wireless

Function definition of pins:



Color	Pins	Color	Pins
Black	GND	Red	Power
Green	GPIO	Dark Green	ADC
Purple	UART(default)	Lavender	UART
Magenta	SPI	Cyan	I2C
Pink	System Control	Orange	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

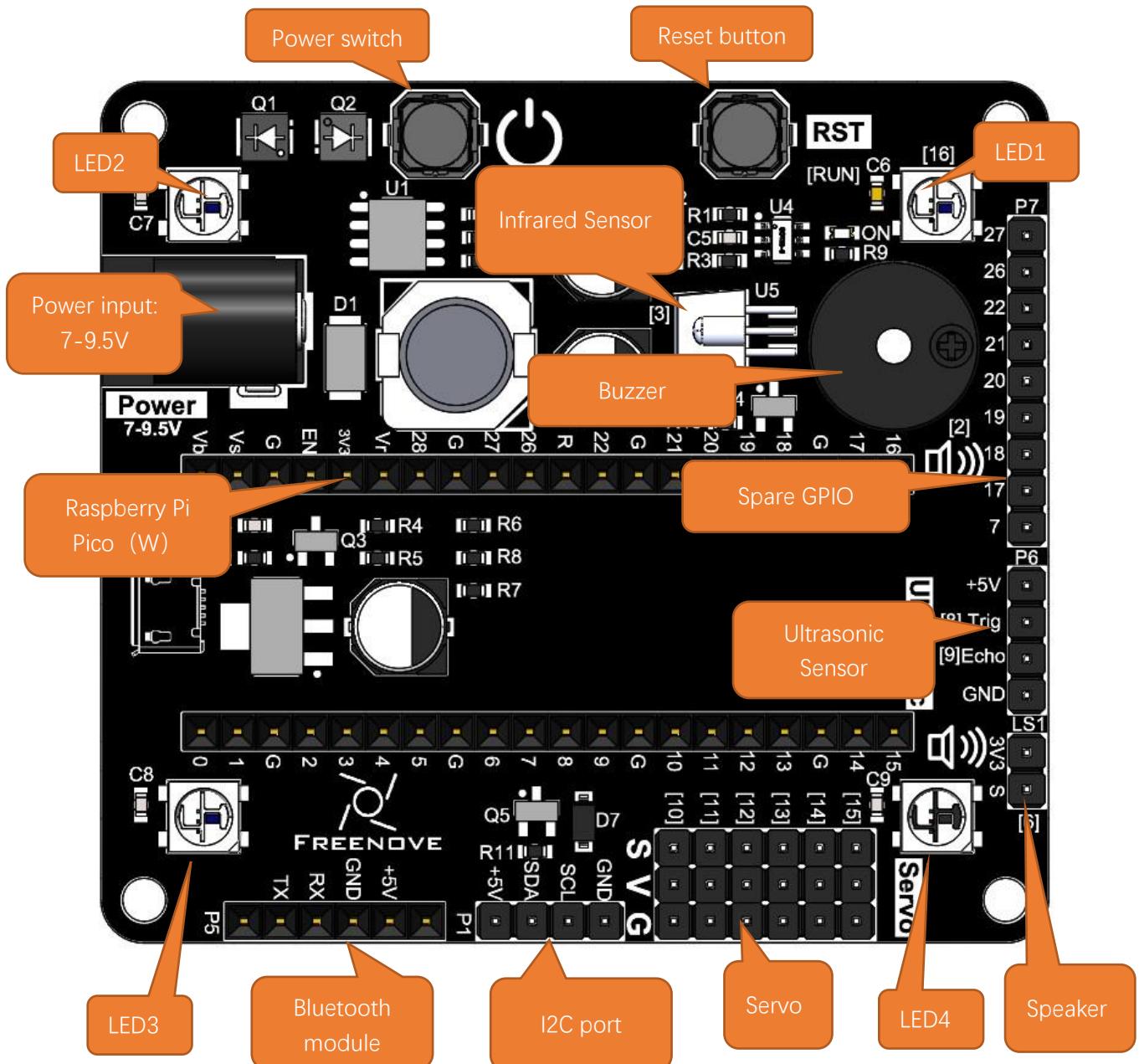
To learn what each GPIO corresponds to, please refer to the following table.

The functions of the pins are allocated as follows:

Pins of Raspberry Pi Pico W	Functions	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:



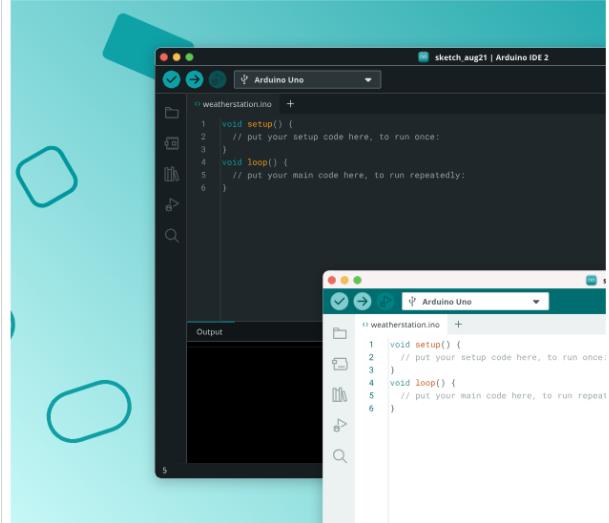
# Installation of Arduino IDE

## 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/en/software/>

### Bring Your Projects to Life with Arduino Software



The screenshot shows the Arduino IDE 2.3.6 interface. It features a central code editor window with the following sketch code:

```
weatherstation.ino
1 void setup() {
2   // put your setup code here, to run once:
3 }
4 void loop() {
5   // put your main code here, to run repeatedly:
6 }
```

Below the code editor is an 'Output' window. To the left of the code editor is a sidebar with icons for file operations like Open, Save, and Print.

**Arduino IDE 2.3.6**  
[Release notes](#)

The new major release of the Arduino IDE is faster and even more powerful! In addition to a more modern editor and a more responsive interface it features autocompletion, code navigation, and even a live debugger. For more details, check the [Arduino IDE 2.0 documentation](#).

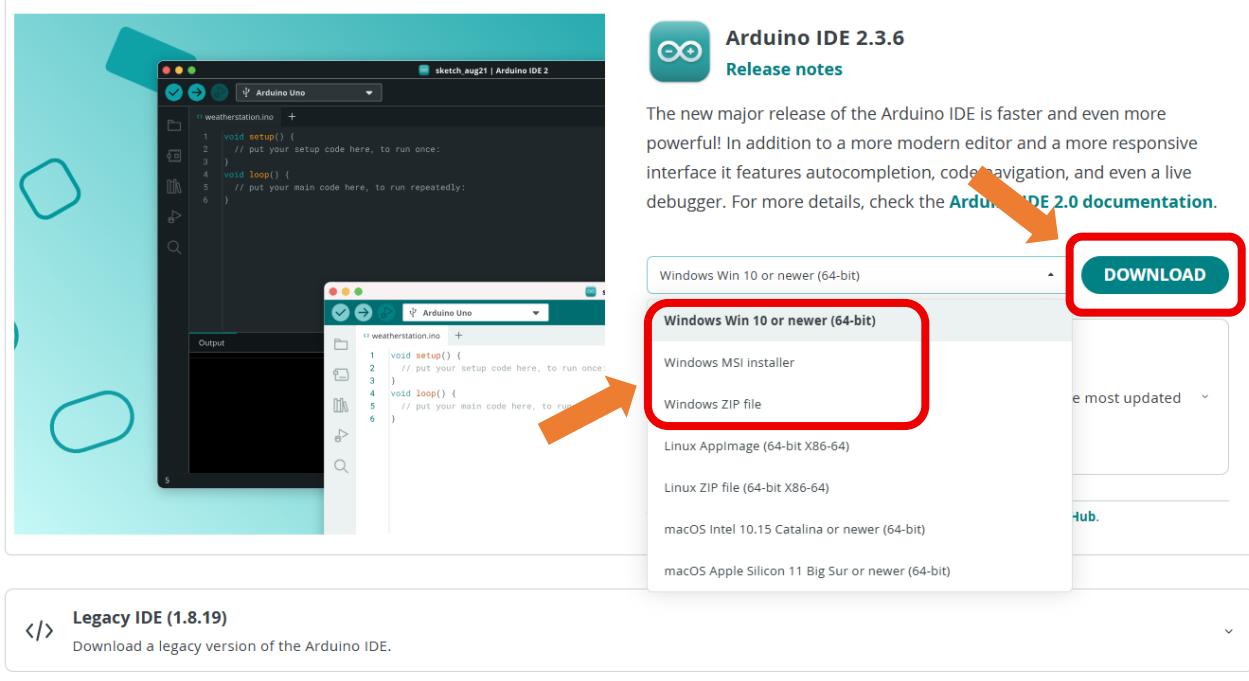
Windows Win 10 or newer (64-bit) [DOWNLOAD](#)

**Nightly Builds**  
Download a preview of the incoming release with the most updated features and bugfixes.

The Arduino IDE 2.0 is open source and its source code is hosted on [GitHub](#).

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

## Bring Your Projects to Life with Arduino Software

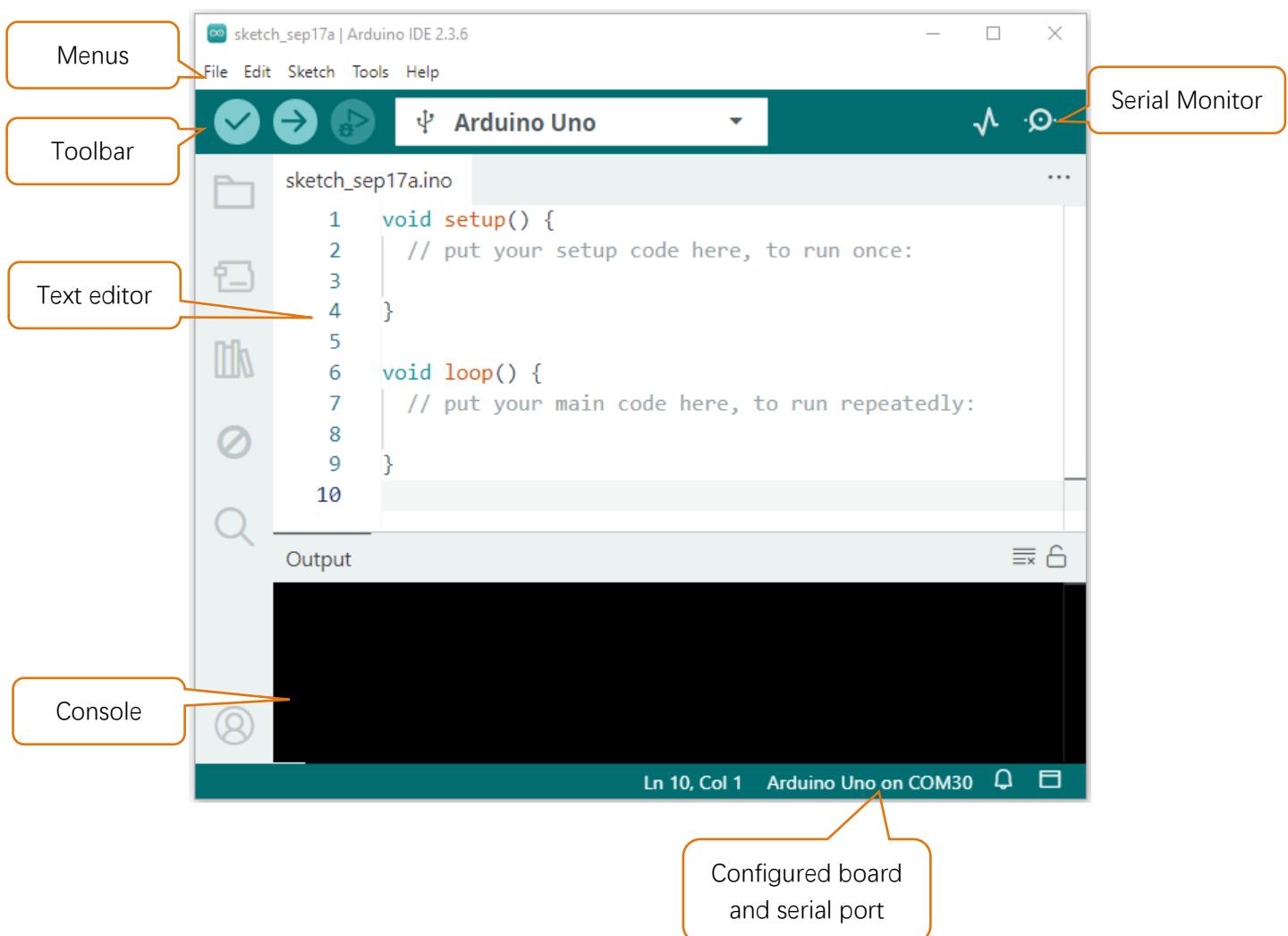


The screenshot shows the Arduino Software download page. At the top, there's a large image of the Arduino IDE interface with code for a weather station. To the right of the image, there's a summary of the new features in version 2.3.6, including a faster editor, responsive interface, and live debugger. Below this, there's a dropdown menu for selecting the operating system: "Windows Win 10 or newer (64-bit)". A red box highlights this dropdown, and an orange arrow points from it to a red-bordered button labeled "DOWNLOAD". Another red box highlights the "Windows Win 10 or newer (64-bit)" option in the dropdown menu. At the bottom left, there's a link for "Legacy IDE (1.8.19)" with the note "Download a legacy version of the Arduino IDE".

After the downloading completes, run the installer. For Windows users, there may pop up an installation dialog box of driver during the installation process. When it is popped up, please allow the installation. After installation is completed, an shortcut will be generated in the desktop.



Run it. The interface of the software is as follows:

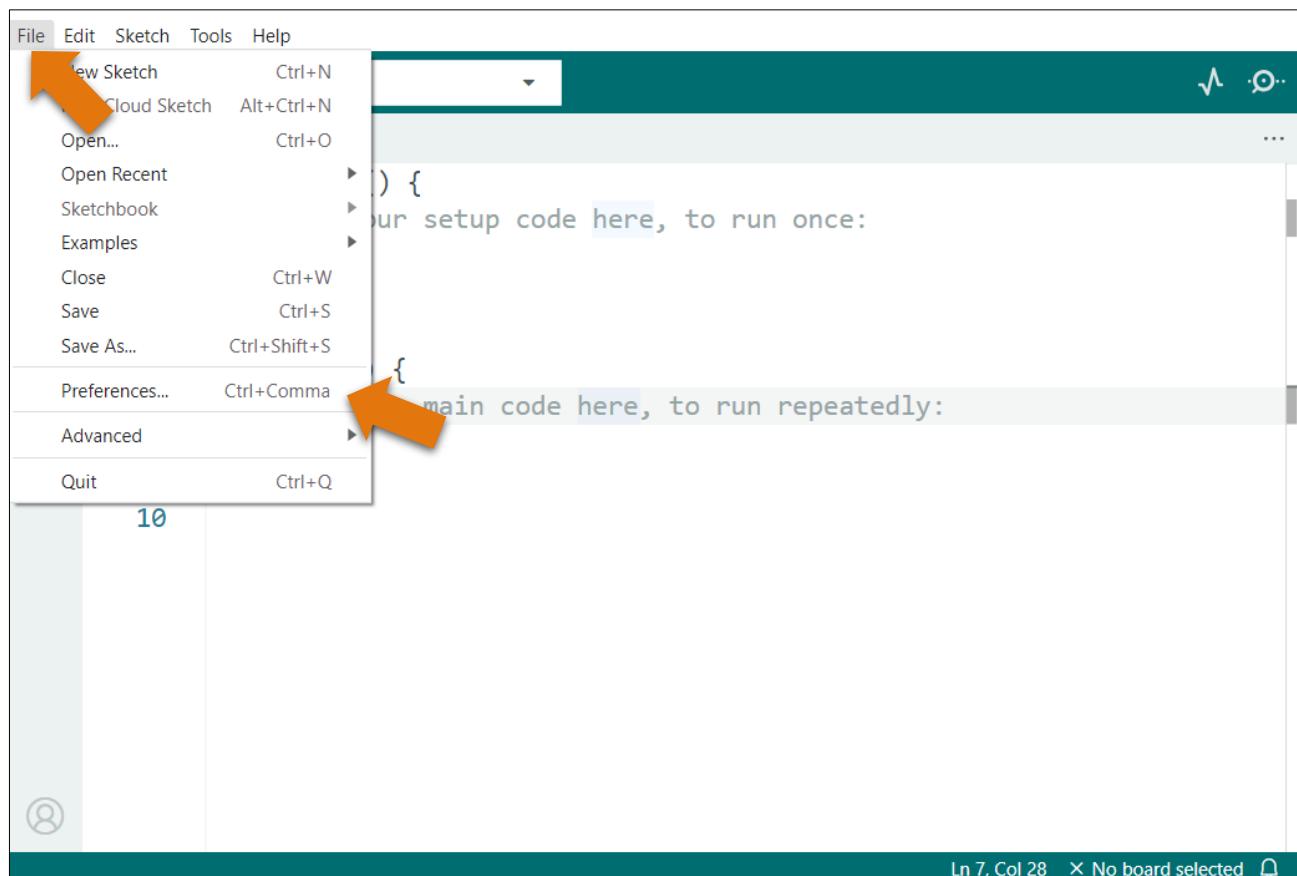


Programs written with Arduino IDE are called sketches. These sketches are written in a text editor and are saved with the file extension .ino. The editor has features for cutting/pasting and for searching/replacing text. The console displays text output by the Arduino IDE, including complete error messages 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, open the serial monitor, and access the serial plotter.

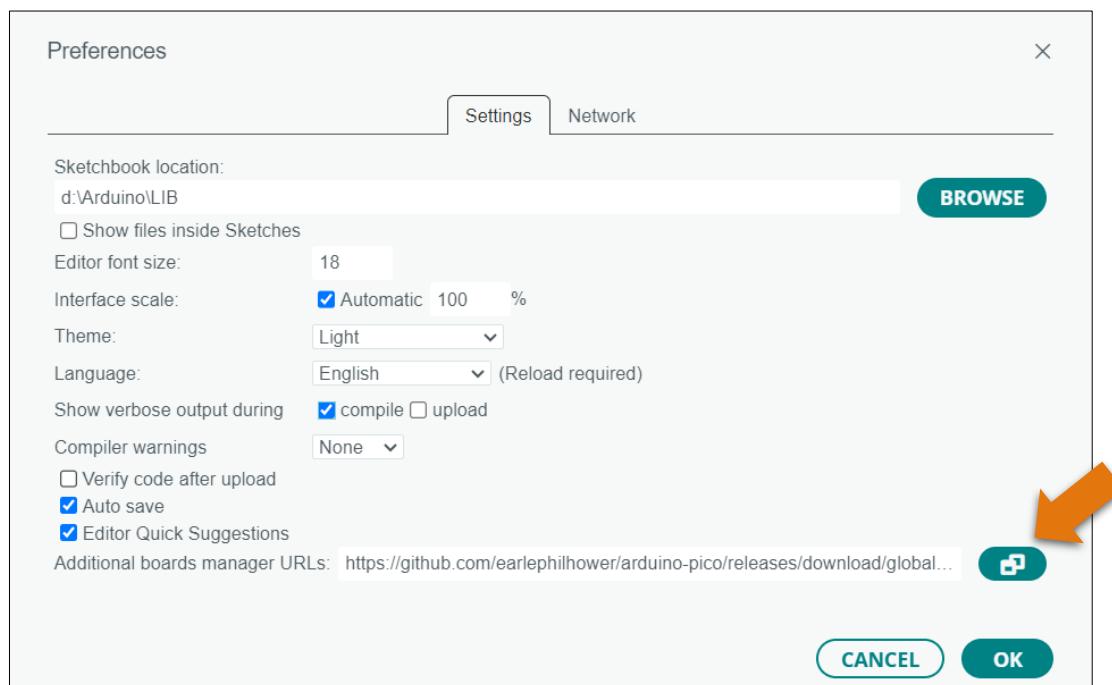
	Verify Checks your code for errors compiling it.
	Upload Compiles your code and uploads it to the configured board.
	Debug Troubleshoot code errors and monitor program running status.
	Serial Plotter Real-time plotting of serial port data charts.
	Serial Monitor Used for debugging and communication between devices and computers.

## Environment Configuration

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

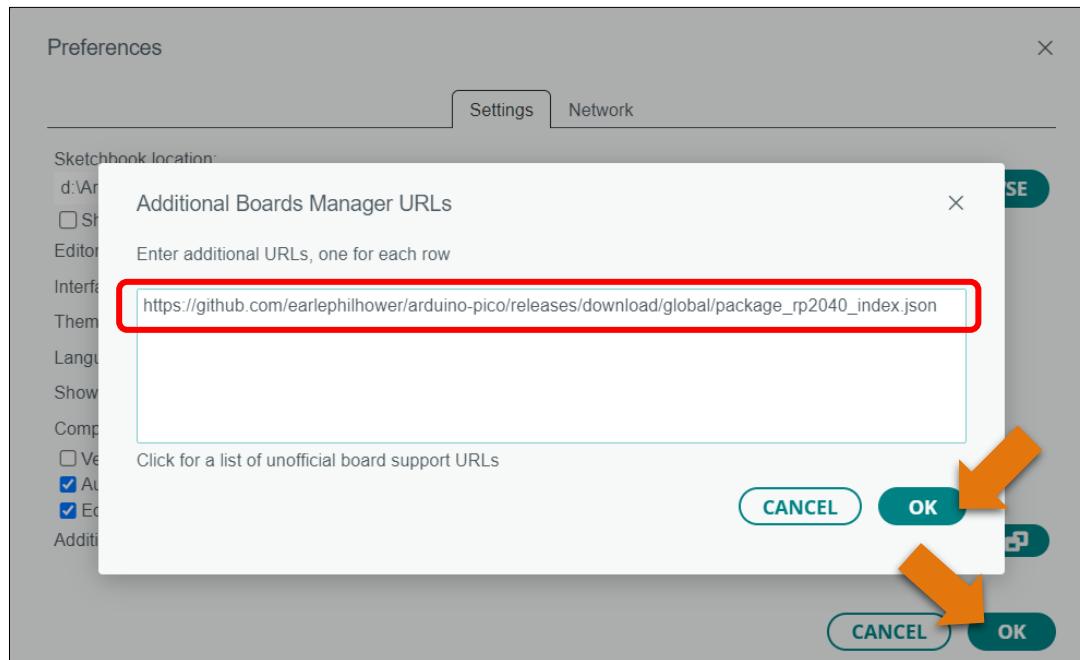


Second, click on the symbol behind "Additional Boards Manager URLs"



Third, fill in

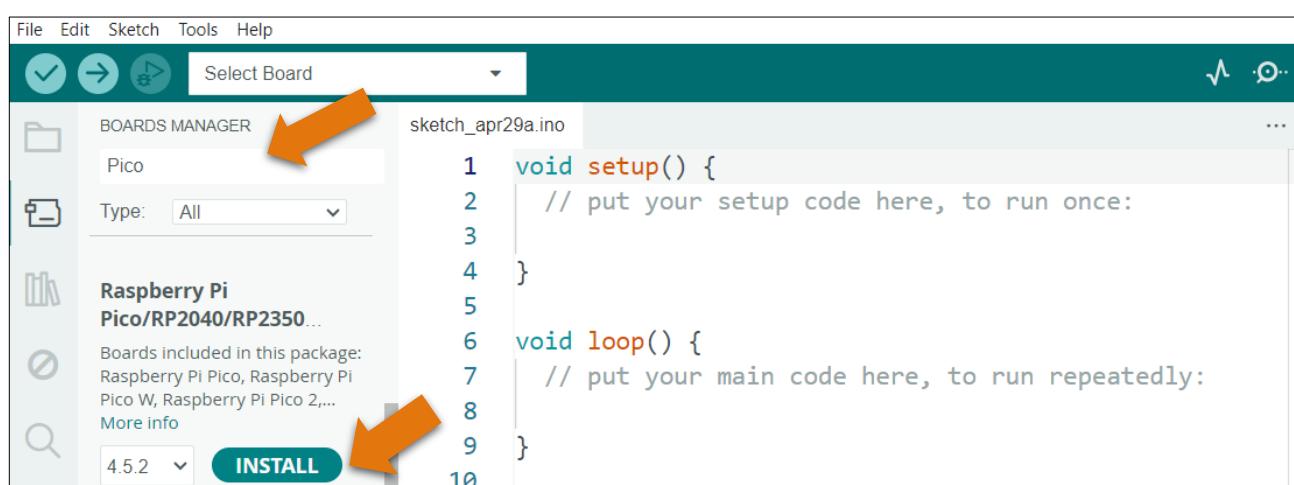
[https://github.com/earlephilhower/arduino-pico/releases/download/global/package\\_rp2040\\_index.json](https://github.com/earlephilhower/arduino-pico/releases/download/global/package_rp2040_index.json) in the new window, click OK, and click OK on the Preferences window again.



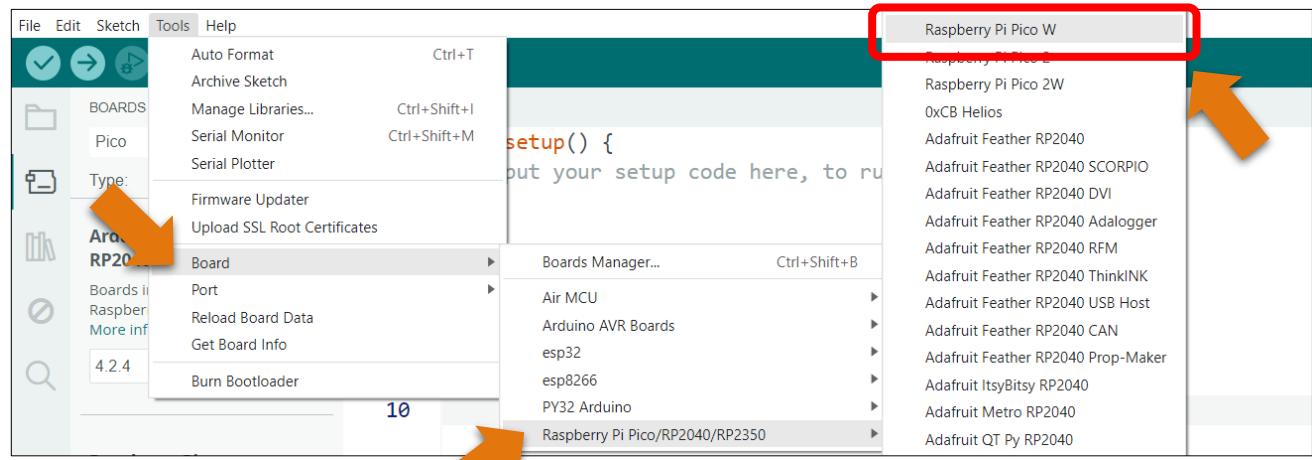
Fourth, click Boards Manager on the left.



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 2.4GHz wireless interface using Infineon CYW43439. That is to say, the hardware Raspberry Pi Pico W is almost the same as the normal Pico except for the wireless interface.

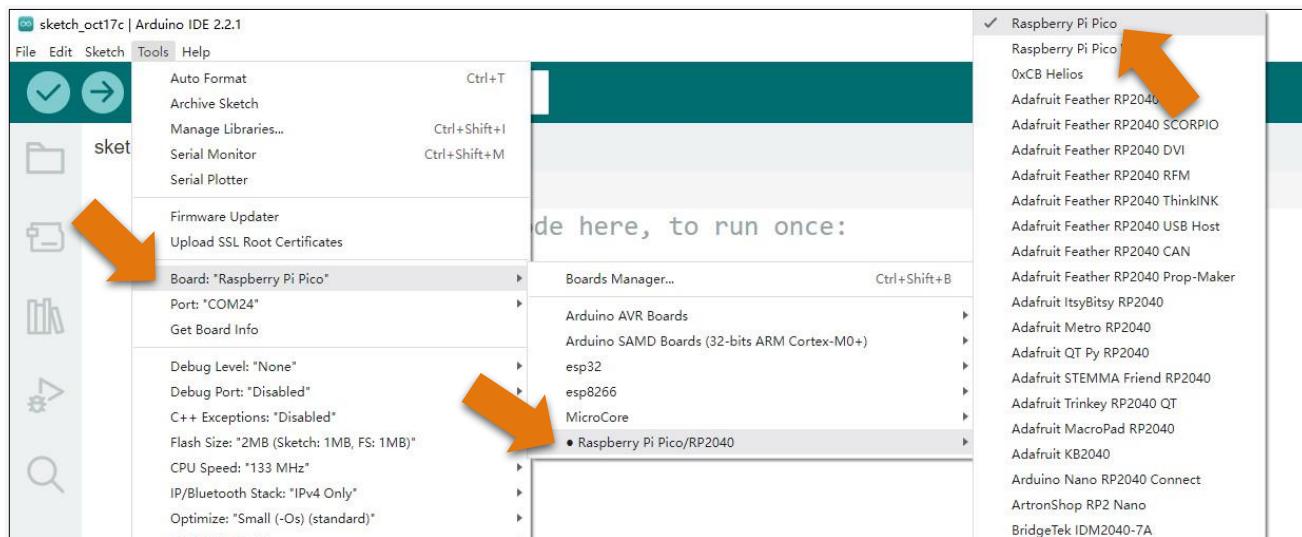
No matter which board you use, the procedure for each project is the same. You only need to select the correct board and upload the corresponding code based on the board you use.

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.

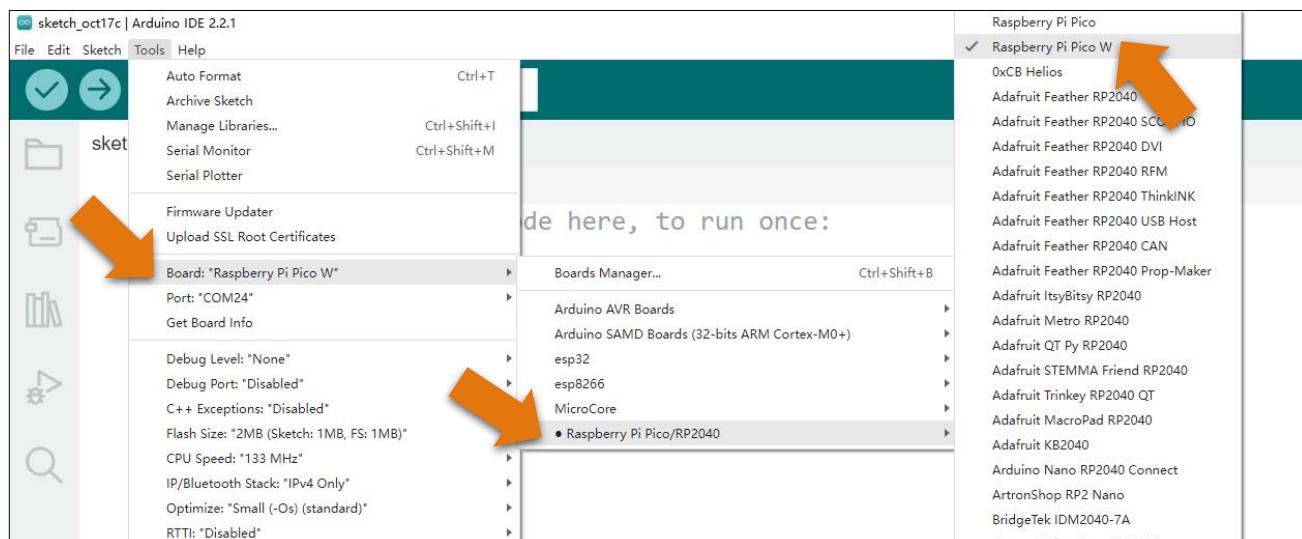
For Raspberry Pi Pico:

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



For Raspberry Pi Pico W:

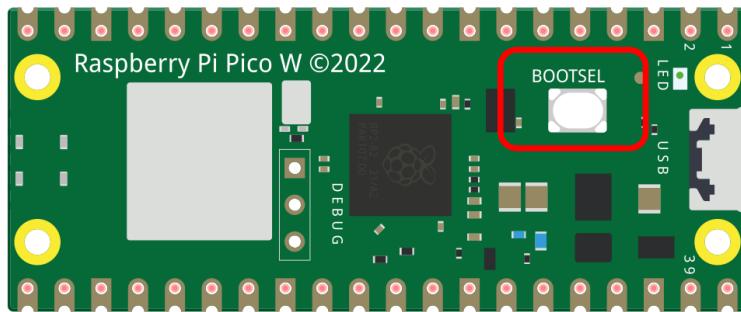
Click Tools on Menu bar, click Board, select “Raspberry Pi Pico/RP2040”, and select Raspberry Pi Pico W.



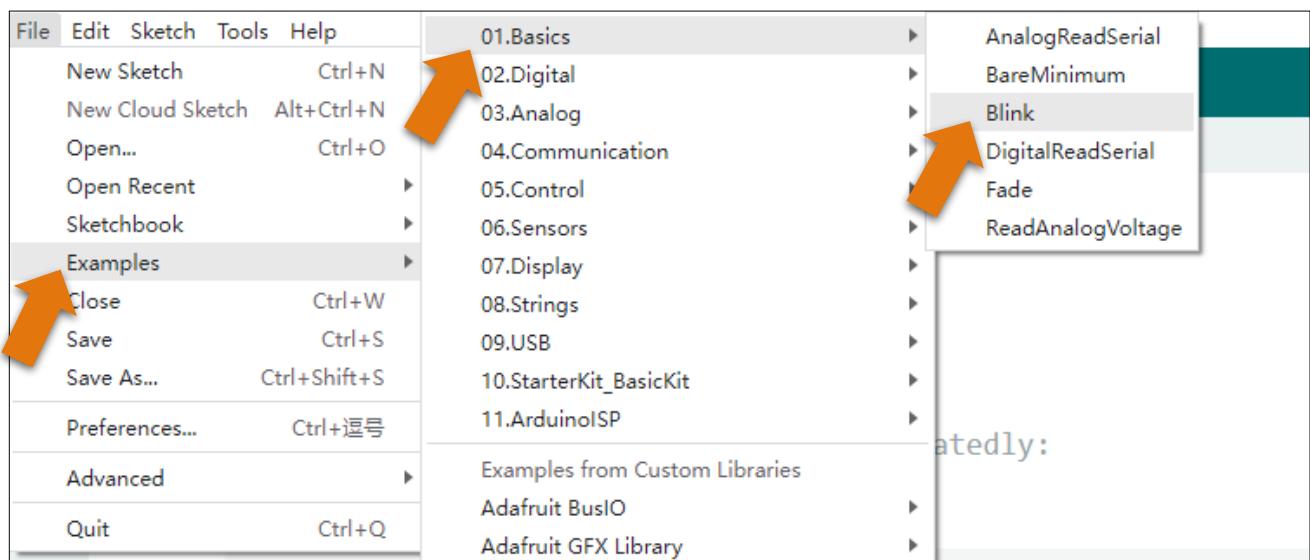
## Uploading Arduino-compatible Firmware for Raspberry Pi Pico (W)

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

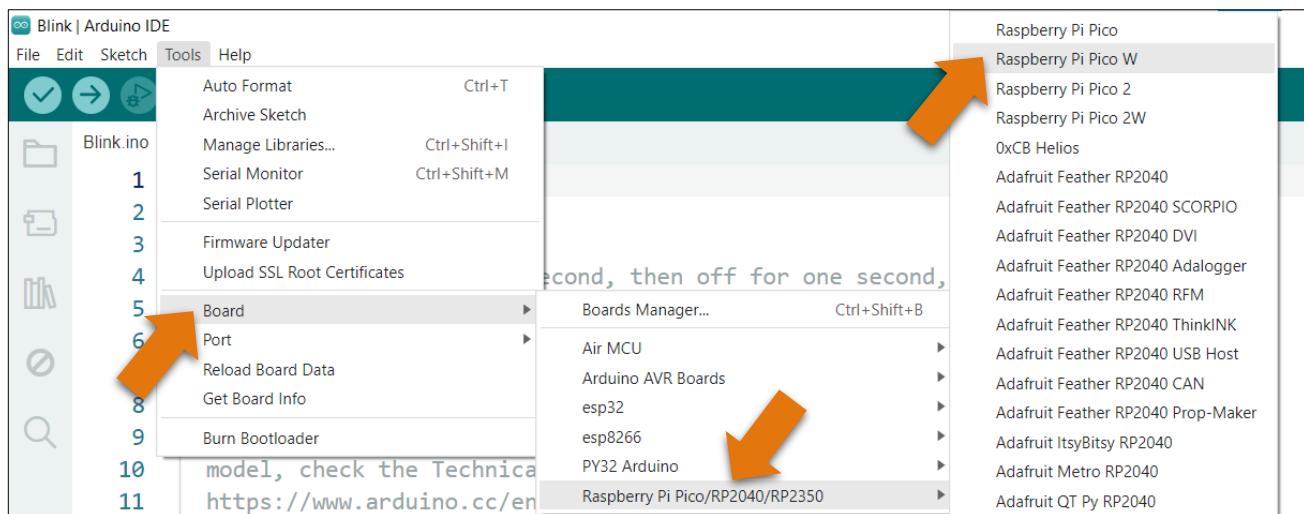
1. Disconnect Pico (W) from your computer. Press and hold the white button (BOOTSEL) on Pico (W) while connecting it to your computer. (Note: Be sure to hold the button before powering the Pico, otherwise the firmware will not download successfully.)



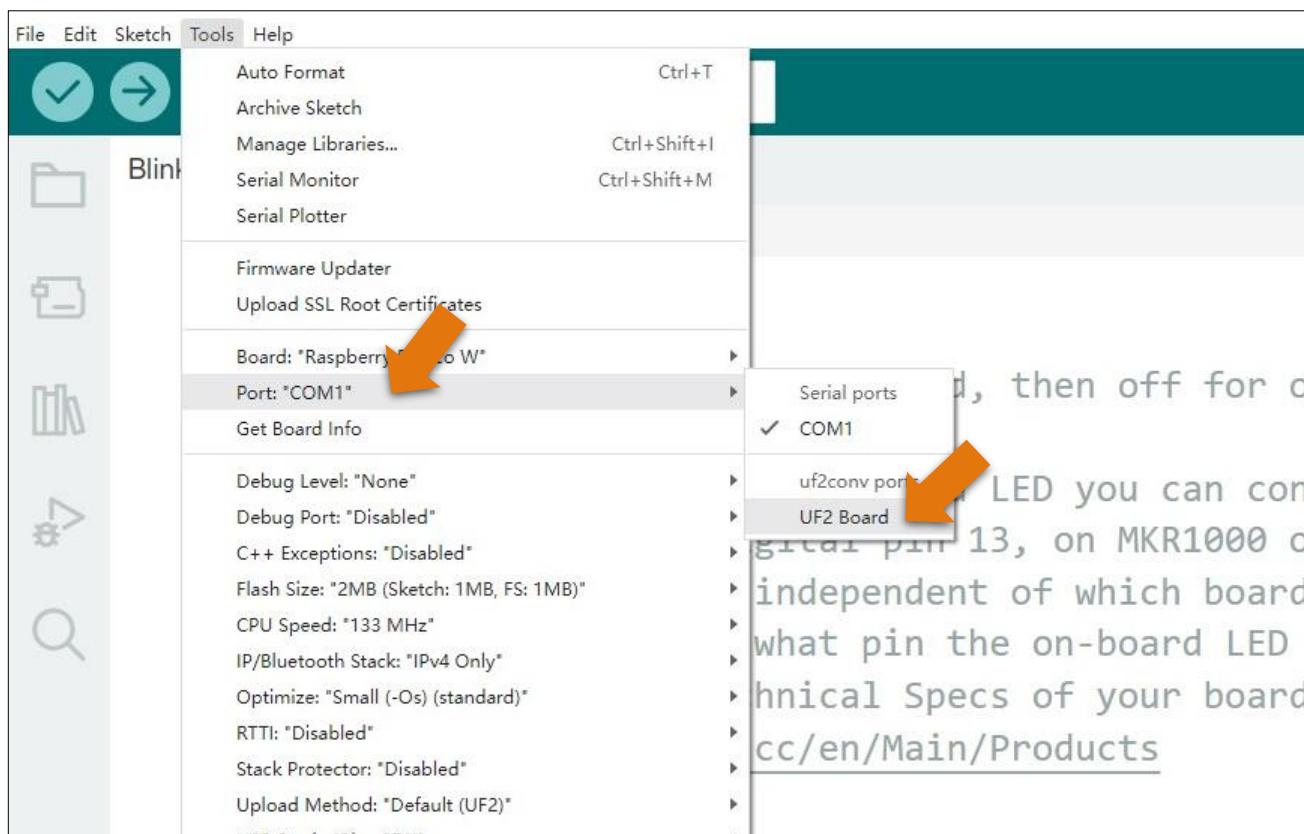
2. Open Arduino IDE. Click **File>Examples>01.Basics>Blink**.



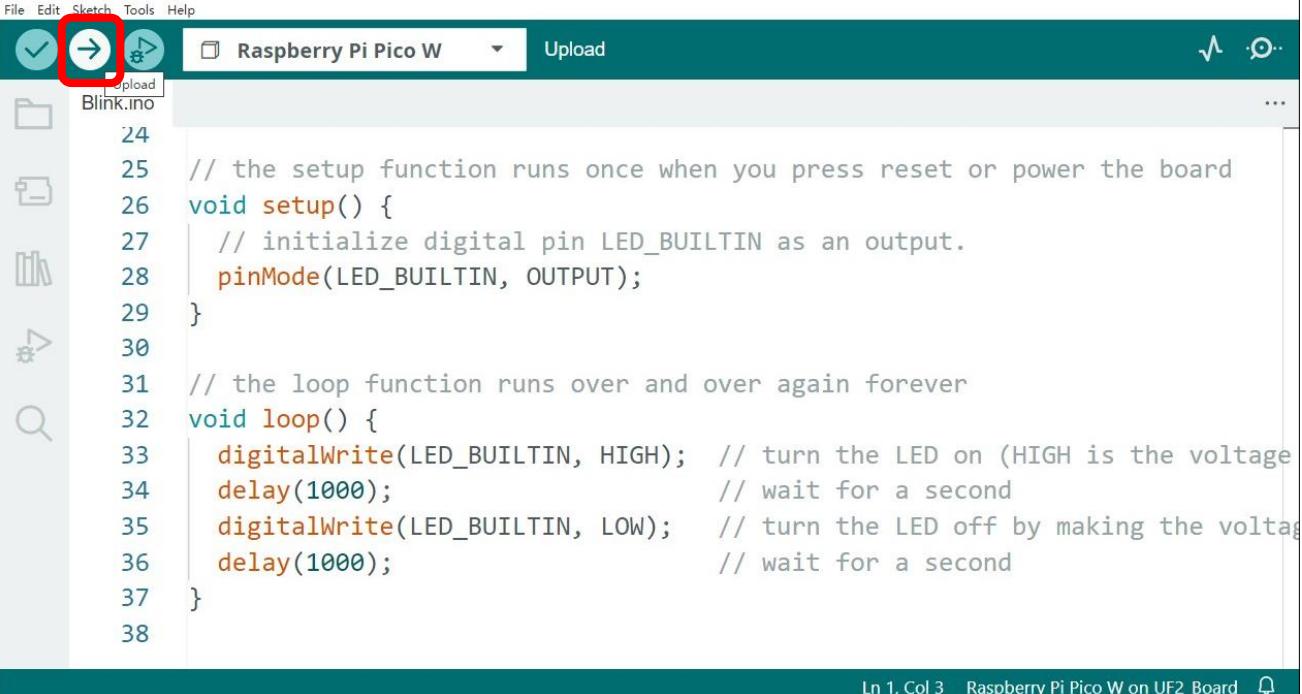
3. Click Tools>Board>Raspberry Pi RP2040 Boards>Raspberry Pi Pico W.



4. Click Tools>Port>UF2 Board.



## 5. Upload sketch to Raspberry Pi Pico W.



The screenshot shows the Arduino IDE interface. The top menu bar includes File, Edit, Sketch, Tools, Help, and a language selection dropdown set to English (US). Below the menu is a toolbar with icons for upload, refresh, and other functions. The central area displays the code for the 'Blink' sketch:

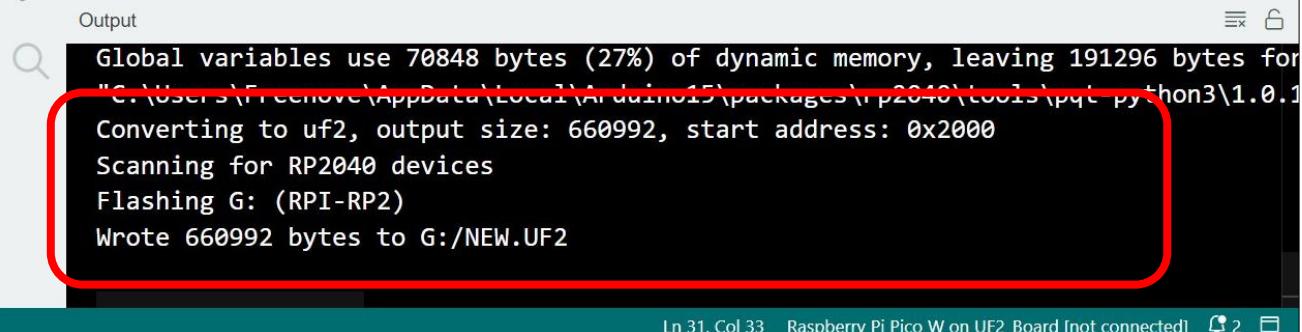
```

24
25 // the setup function runs once when you press reset or power the board
26 void setup() {
27     // initialize digital pin LED_BUILTIN as an output.
28     pinMode(LED_BUILTIN, OUTPUT);
29 }
30
31 // the loop function runs over and over again forever
32 void loop() {
33     digitalWrite(LED_BUILTIN, HIGH);    // turn the LED on (HIGH is the voltage
34     delay(1000);                      // wait for a second
35     digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage
36     delay(1000);                      // wait for a second
37 }
38

```

The status bar at the bottom indicates "Ln 1, Col 3" and "Raspberry Pi Pico W on UF2\_Board".

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

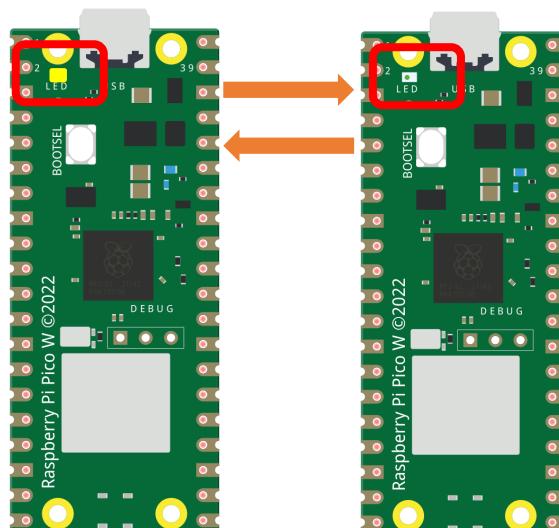


The screenshot shows the Arduino IDE's Output window. It displays the progress of the upload process:

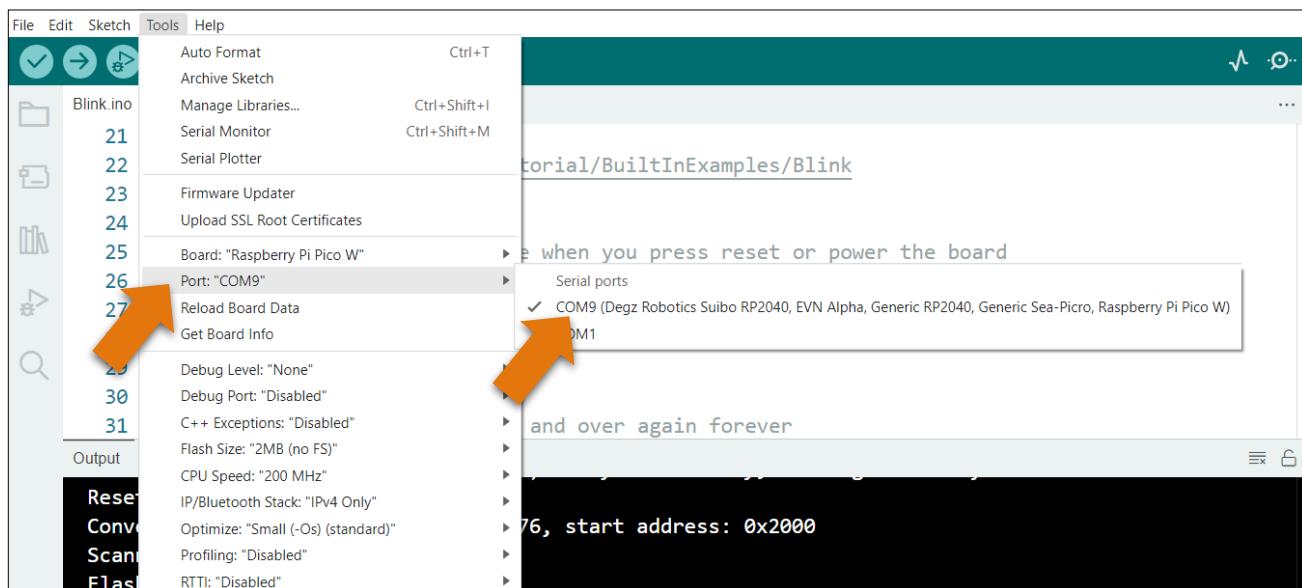
- Global variables use 70848 bytes (27%) of dynamic memory, leaving 191296 bytes for "C:\Users\Freenove\AppData\Local\Arduino15\packages\rp2040\tools\pyuf2\python3\1.0.1"
- Converting to uf2, output size: 660992, start address: 0x2000
- Scanning for RP2040 devices
- Flashing G: (RPI-RP2)
- Wrote 660992 bytes to G:/NEW.UF2

The status bar at the bottom indicates "Ln 31, Col 33" and "Raspberry Pi Pico W on UF2\_Board [not connected]".

The indicator on Pico W starts to flash.



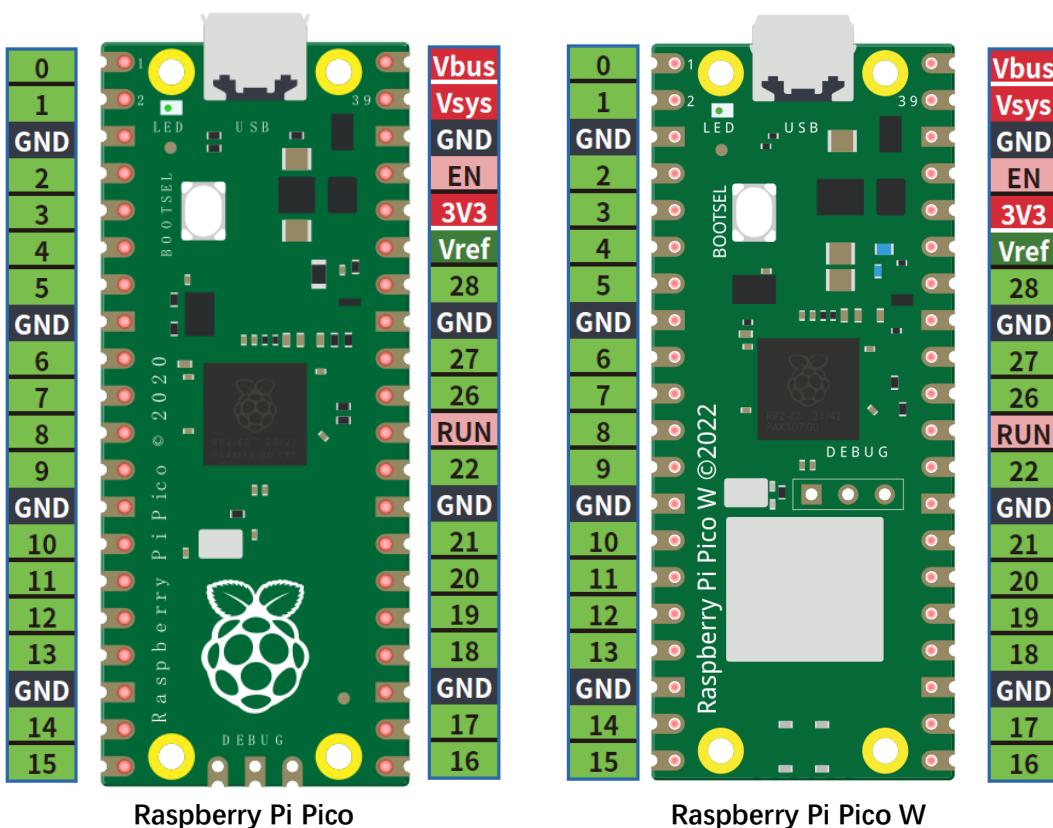
6. Click **Tools>Port>COMx (Raspberry Pi Pico W)**. X of COMx varies from different computers. Please select the correct one on your computer. In our case, it is COM9.



#### 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 been 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:

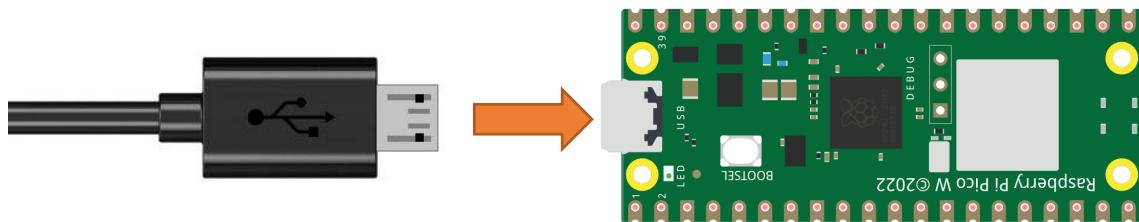


## Uploading the First Code

Here we take “00.0\_Servo\_90” in “**Freenove\_Bipedal\_Robot\_Kit\_for\_Raspberry\_Pi\_Pico\Sketches**” as an example. The code is to rotate the servo motors to 90°.

## Upload the First Code

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

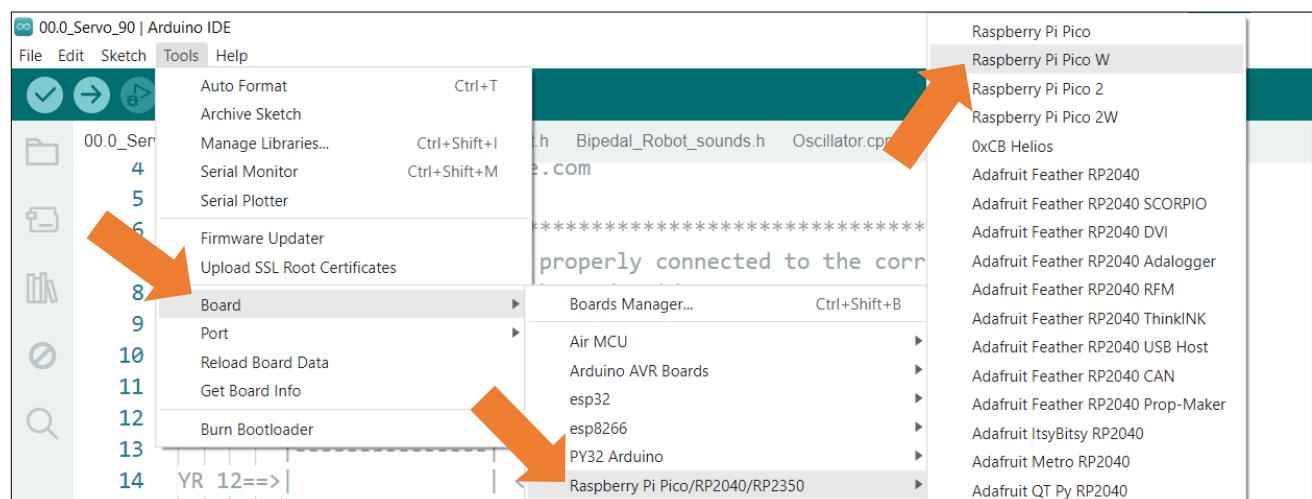


Step 2. Open “00.0\_Servo\_90” folder in “**Freenove\_Bipedal\_Robot\_Kit\_for\_Raspberry\_Pi\_Pico\Sketches**”, and double-click “00.0\_Servo\_90.ino”.

Freenove_Bipedal_Robot_Kit_for_Raspberry_Pi_Pico-main > Sketches >			
Name	Date modified	Type	Size
00.0_Servo_90	11/14/2024 10:51 AM	File folder	
01.1.1_Servo_Calibration	11/14/2024 10:51 AM	File folder	
01.1.2_Servo_Calibration_PC_Tools	11/14/2024 10:51 AM	File folder	
01.1.3_Servo_Calibration_Bluetooth	11/14/2024 10:51 AM	File folder	
01.1.4_Servo	11/14/2024 10:51 AM	File folder	
01.2_Buzzer	11/14/2024 10:51 AM	File folder	

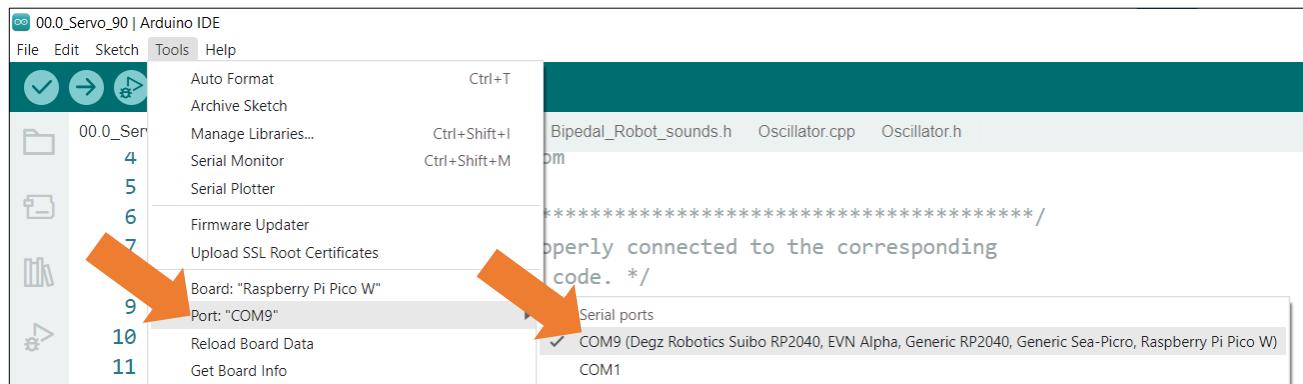
Step 3. Select development board.

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

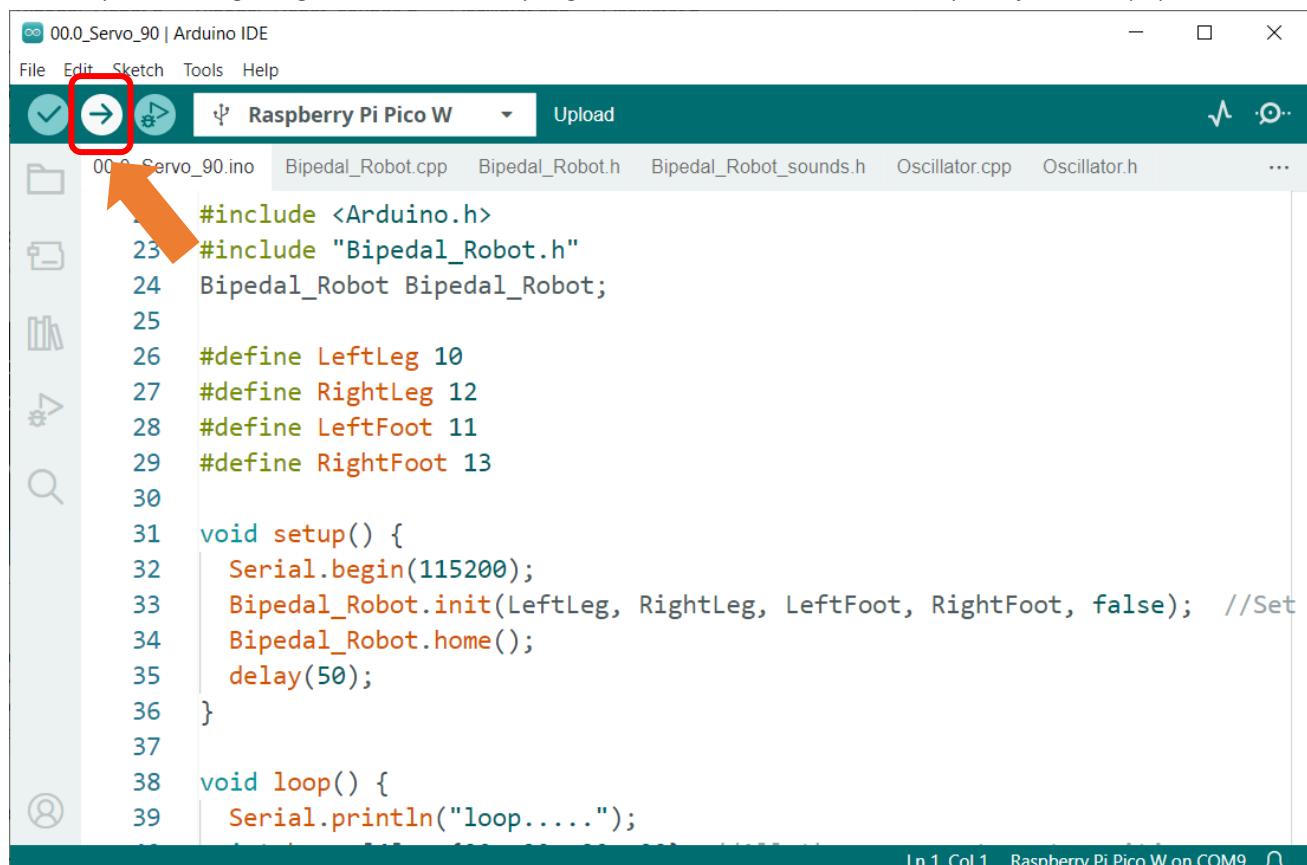


#### Step 4. Select serial port.

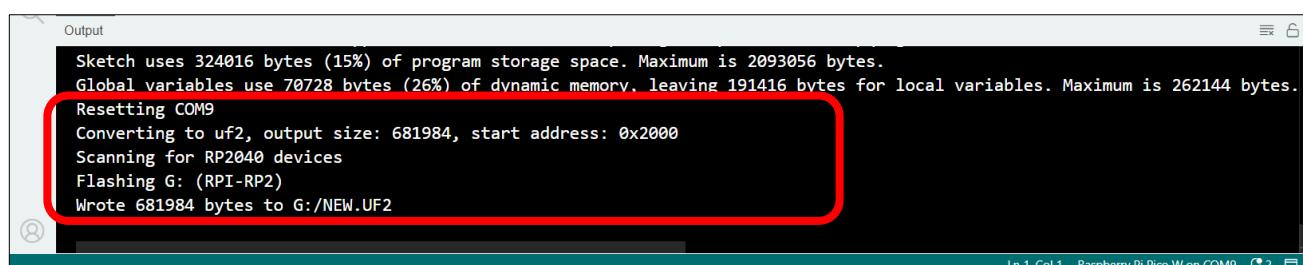
Click Tools on Menu bar, navigate your mouse to Port and select COMx on your computer. The value of COMx varies in different computers, but it will not affect the download function of Raspberry Pi Pico (W), as long as you select the correct one.



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. Align the cable to the pins, yellow cable to yellow pin, red to red, and brown to black. Incorrect wiring may damage the servos. Install the battery and press the power switch. The gears of the four servos will rotate to 90°.



# Chapter 0 Assembling Bipedal Robot

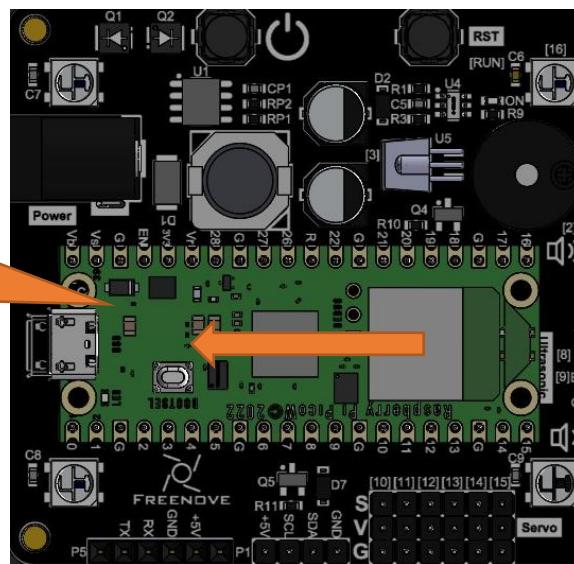
If you have any concerns, please feel free to contact us via [support@freenove.com](mailto:support@freenove.com)

## Assembling the Robot

### Plugging in Raspberry Pi Pico (W)

Plug Raspberry Pi Pico (W) into the shield.

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



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

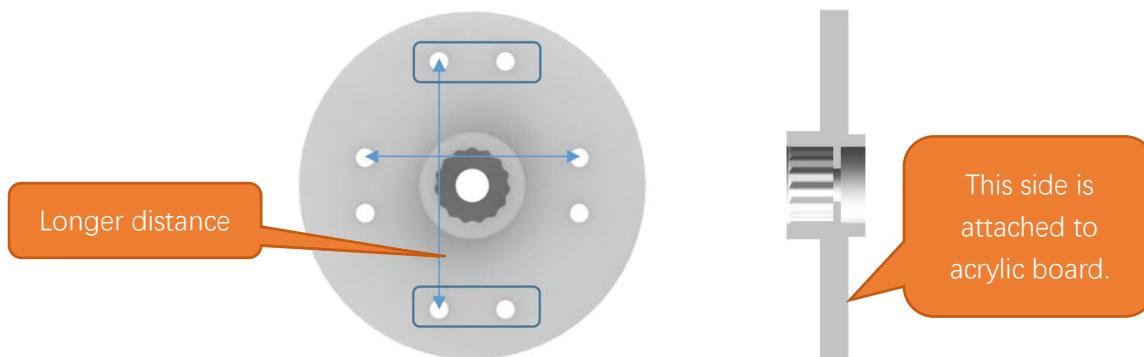
## Installing Disk Servo Arm

Take out four disk servo arms from the servo packages.



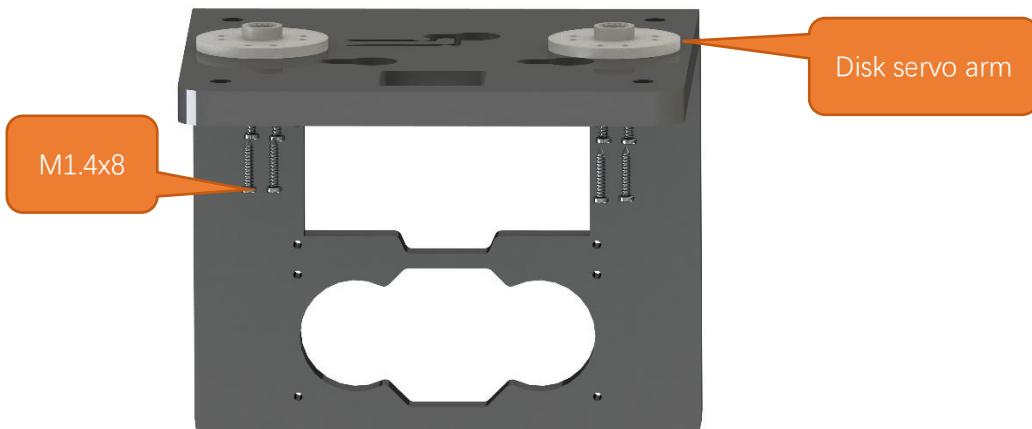
Each servo package includes five M1.4x8 screws and two black servo screws.

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.

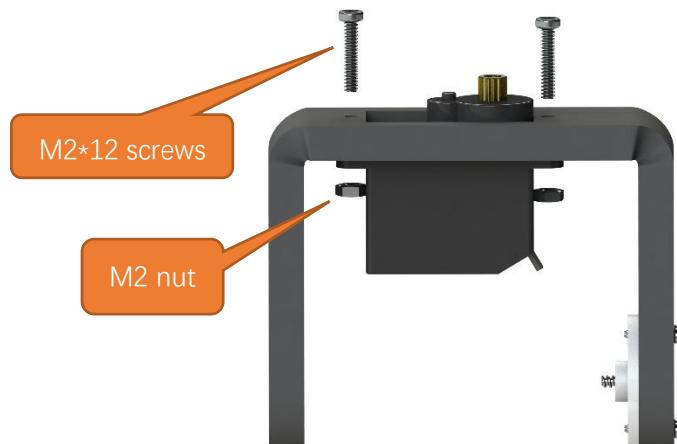




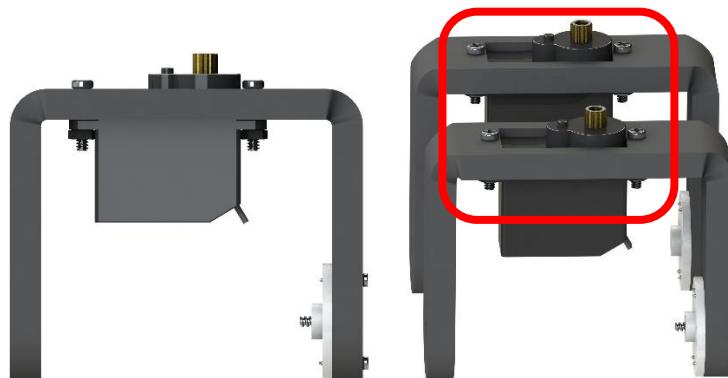
## Installing Servo

Installation steps:

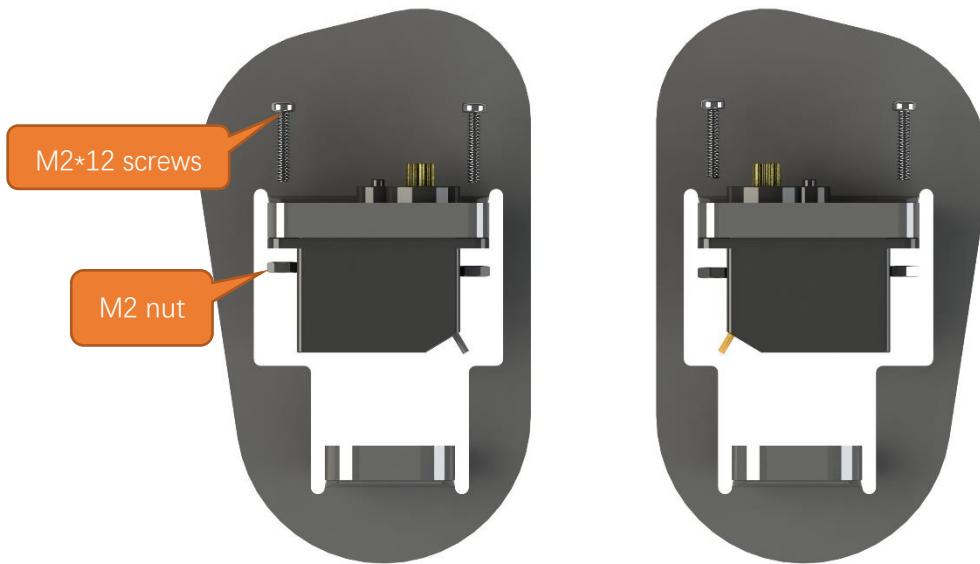
Step 1 Assemble the servos to the two legs with M2\*12 screws and M2 nuts.



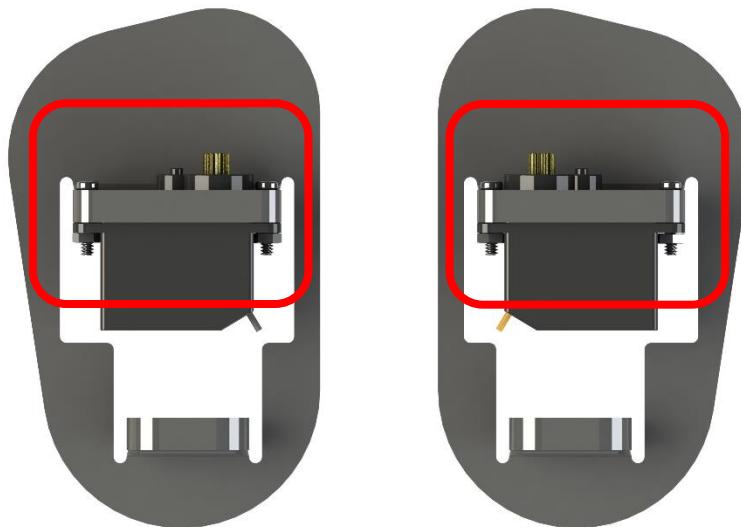
Finished:



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



Finished:



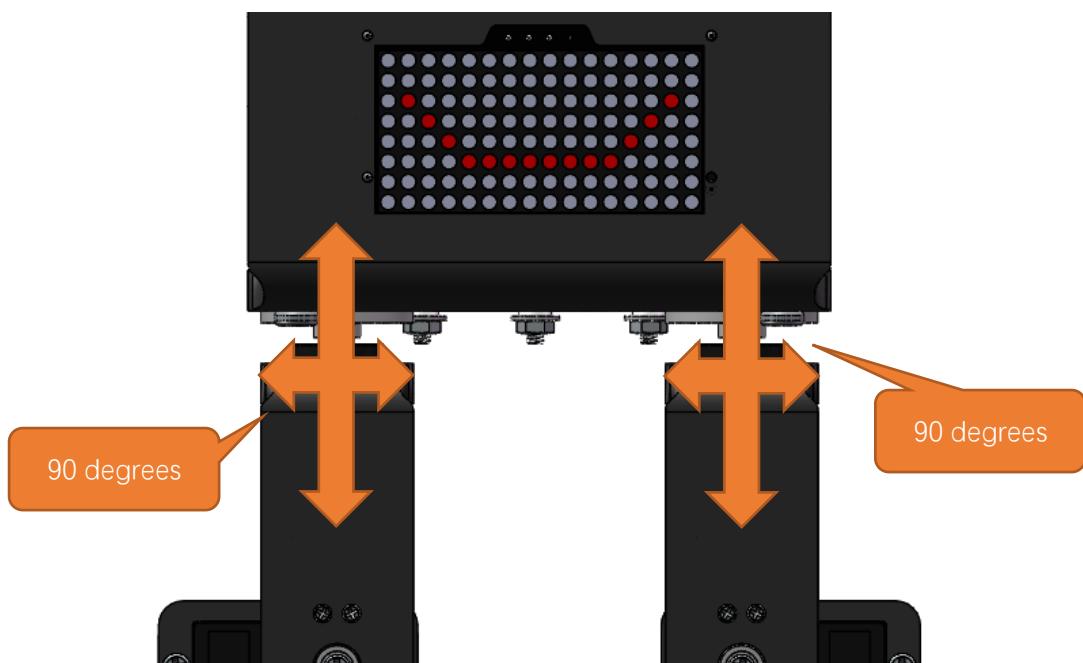
Step 3 Assemble the servos to the robot. Connect Servos 1, 2, 3, and 4 to the corresponding port on the robot board, which are GPIO10, GPIO11, GPIO12, and GPIO13 respectively. Align the cable to the pins, yellow cable to yellow pin, red to red, and brown to black. Incorrect wiring may damage the servos

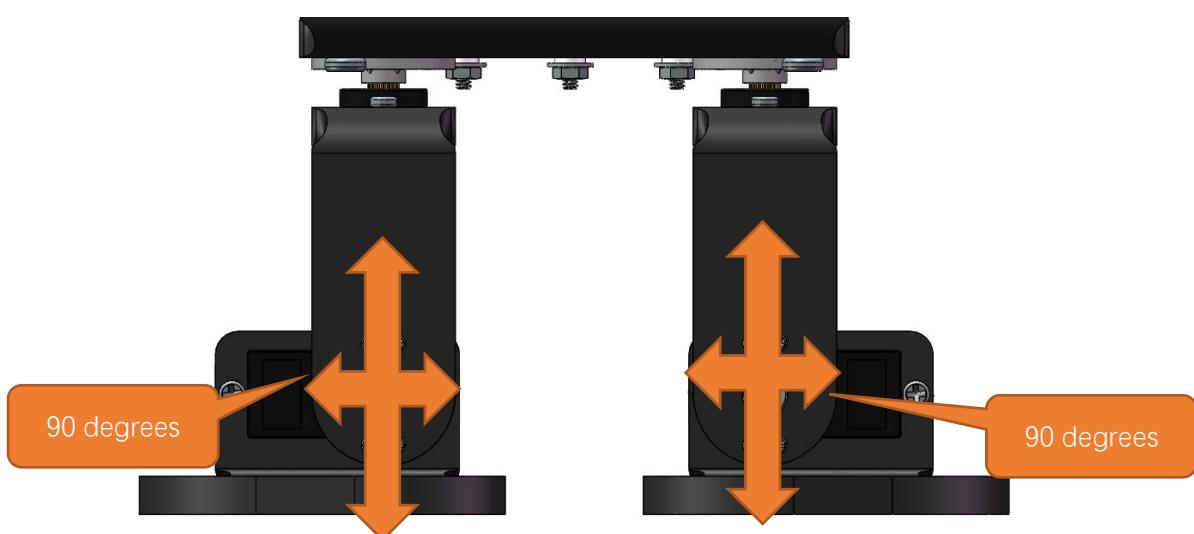
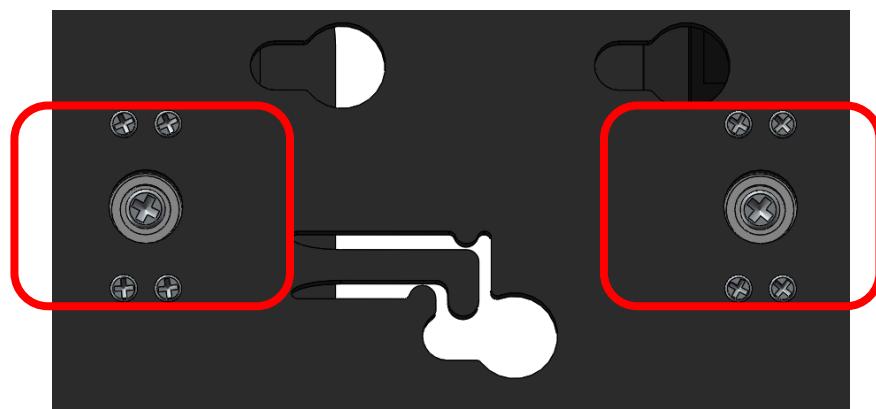
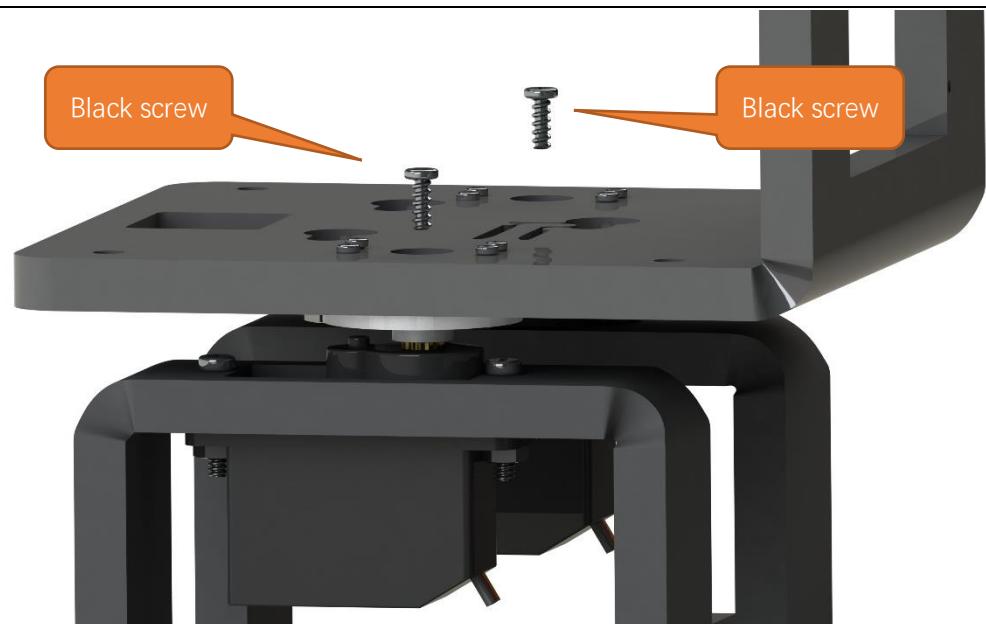
Before assembling the servos, please make sure that the servos are already set to the 90-degree position.

If you have not yet done this, please refer to [this to upload the sketch 00.0\\_Servo\\_90.ino](#).



After uploading the sketch, assemble the servos to the designated position.







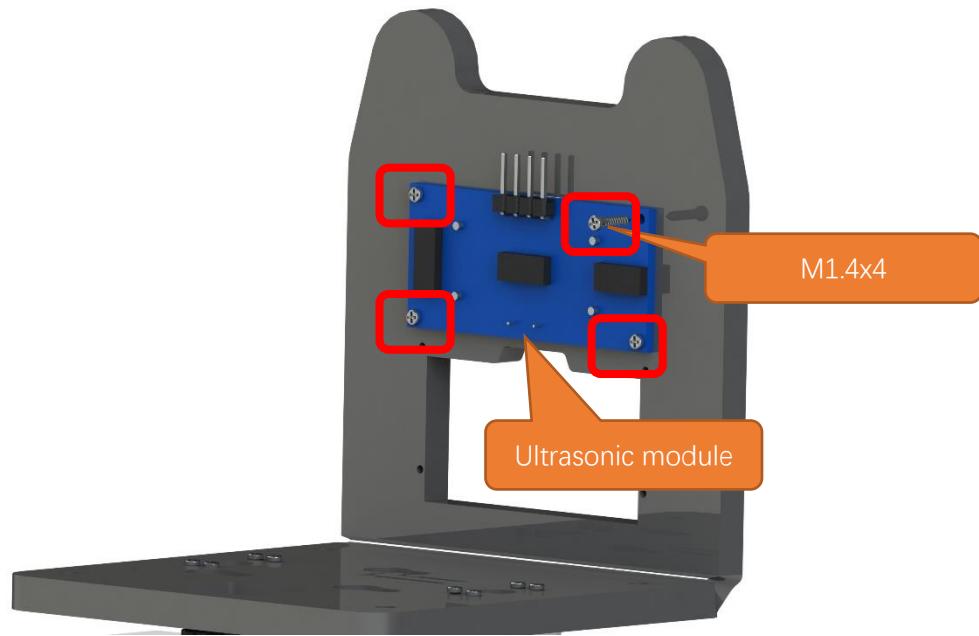
Assemble the leg support screws: Fix the M3 nut with the M3 spanner and screw the M3 screws in. Please note that the M3 nuts here are self-locking non-slip nuts and the screws serve as supports, so you do not need to screw them too tight and some space is needed, so that the feet can move more flexible.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

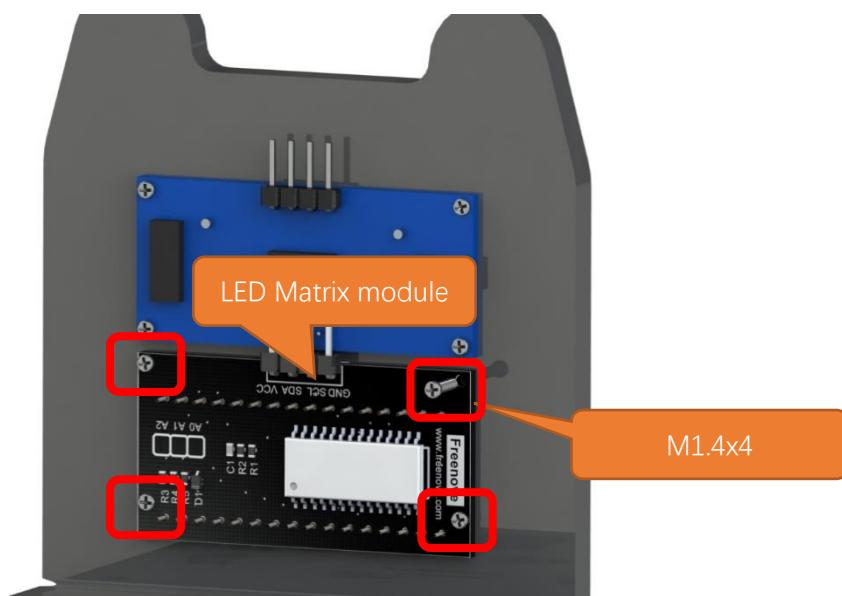
## Installing Ultrasonic Module

Step 1 Install ultrasonic module.



## Installing LED Matrix Module

Step 1 Install LED matrix module.



## Installing 9V Batteries

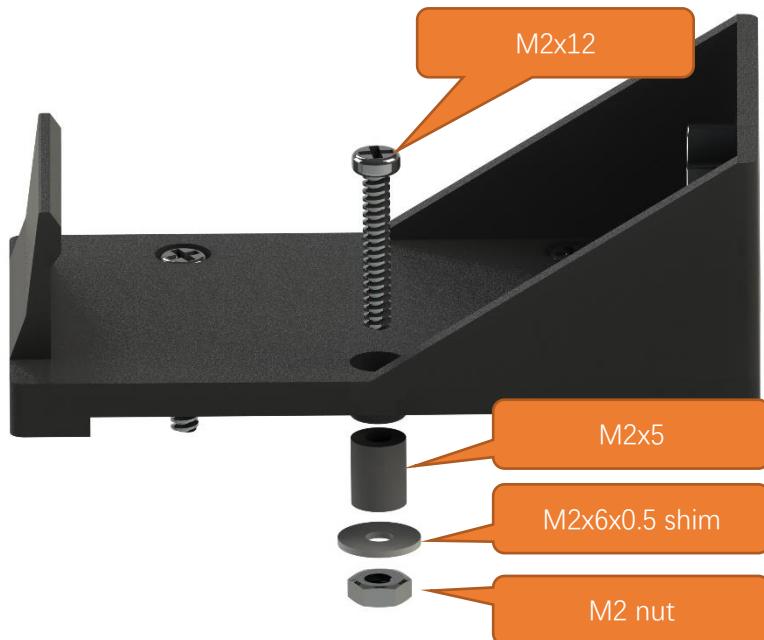
This robot is powered with **9V alkaline batteries** or **9V NIMH rechargeable batteries** as power supply. Please note that 9V carbon batteries and 9V rechargeable lithium batteries **cannot** provide enough power for the robot. Therefore, it is important to pay attention to the battery model when purchasing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

Step 1 Installed with batteries.

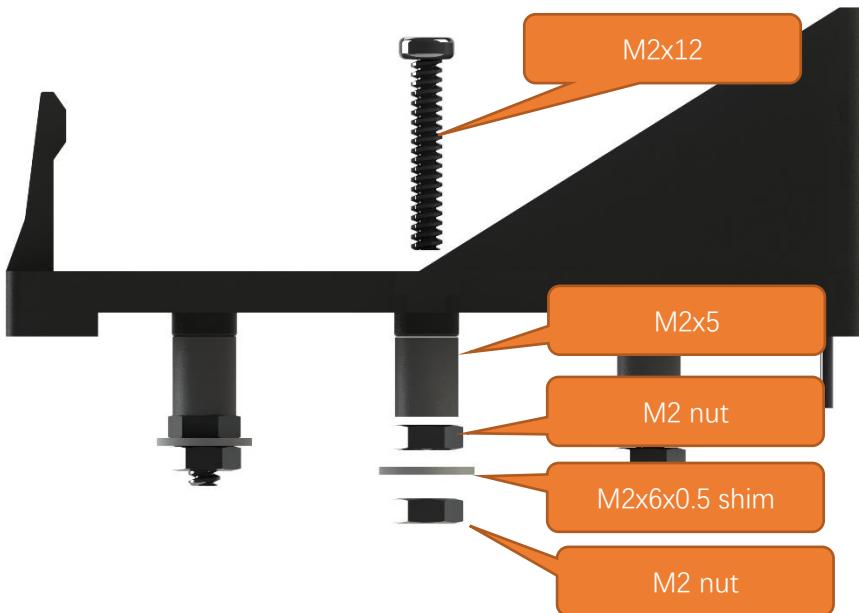
Correctly install the batteries.

Here we provide two installation methods:

Method 1:



Method 2:



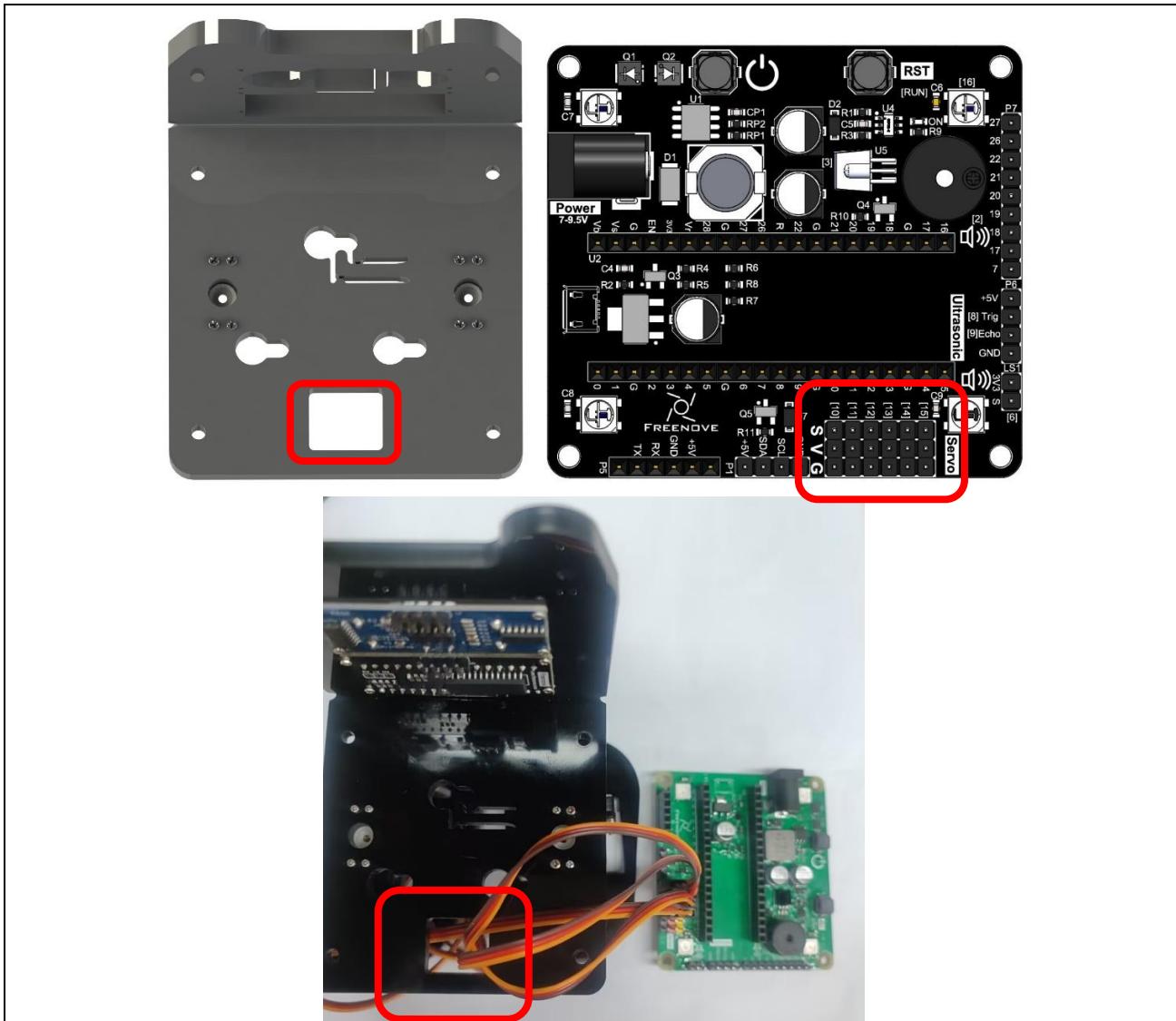
Among them, M2x5 is the acrylic element, and there are 6 M2x5 rings in the acrylic fitting. Only three are used here, and the remaining three are reserved for use.



## Connecting Servo Cables

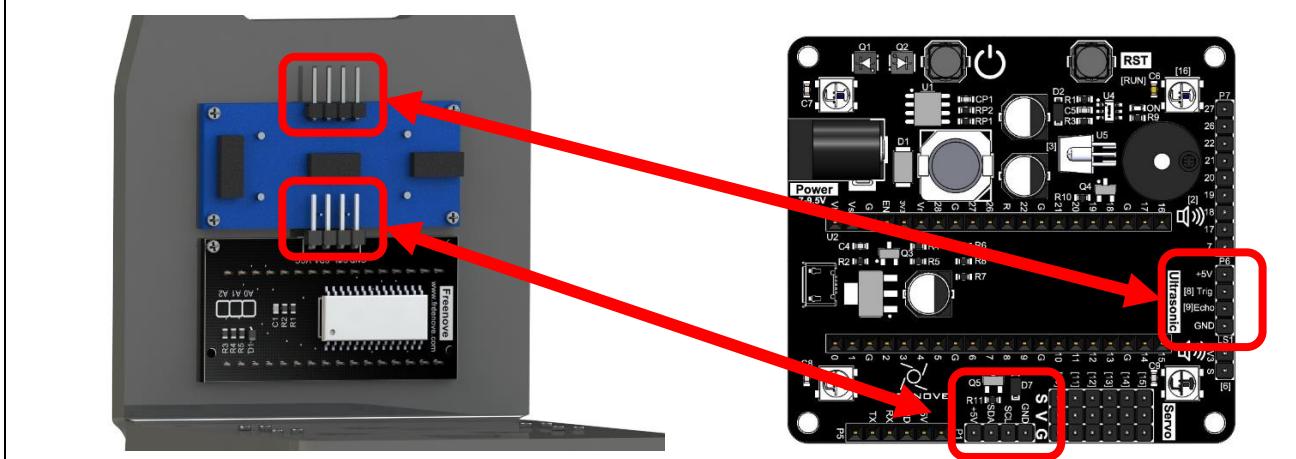
Step 1 Connect the servo cables. Thread the servo wires through the hole on the acrylic part and connect them to the servo port on the robot shield.





## Connect the Ultrasonic Sensor and LED Matrix Module

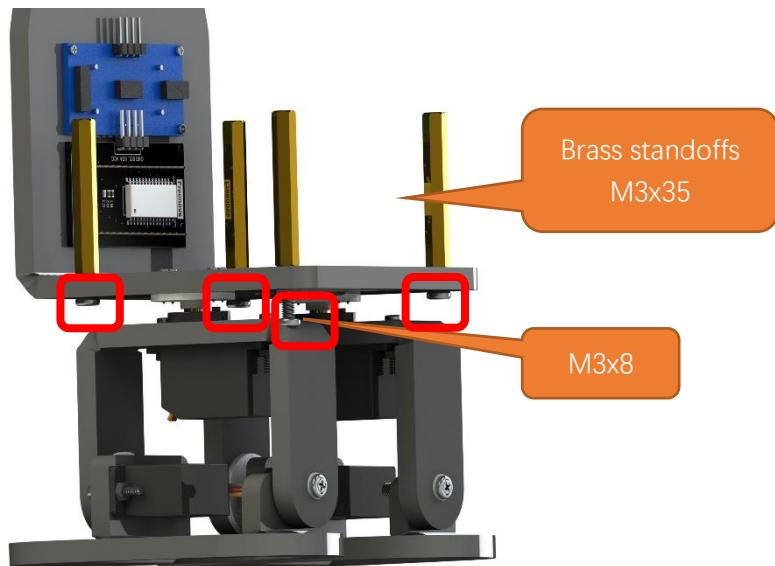
Step 1 Connect the ultrasonic sensor and LED matrix module. It is important that the pins be connected in line with the silk print on the robot shield.



Need support? ✉ [support.freenove.com](mailto:support.freenove.com)

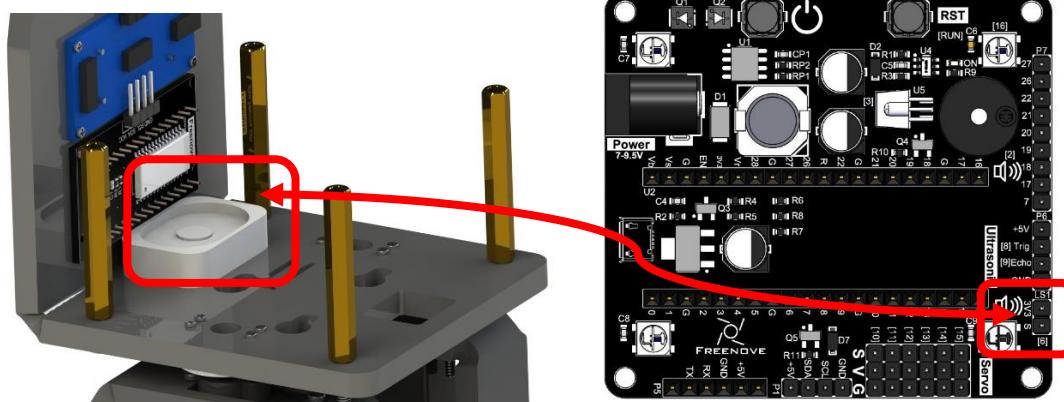
## Installing the Standoffs for Robot Shield

Step 1



## Installing the Speaker

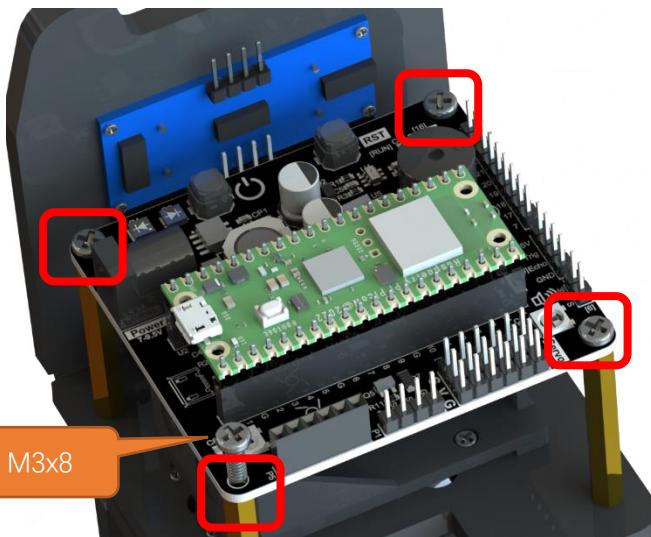
Step 1 Attach the speaker to the robot.



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

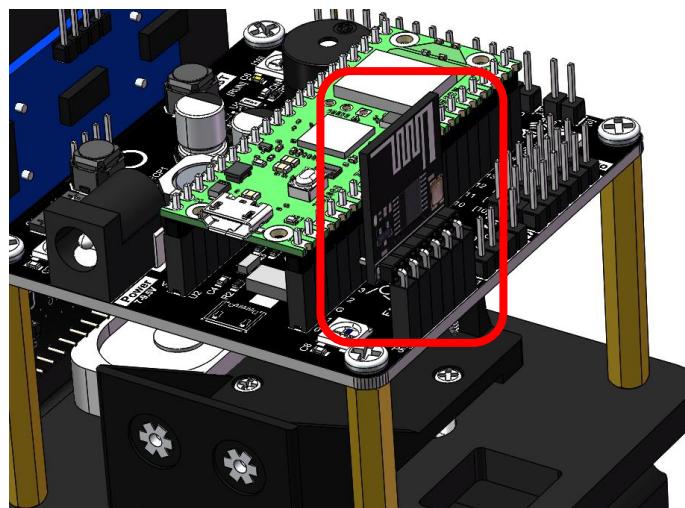
## Installing Robot Shield

Step 1 Install the robot shield.



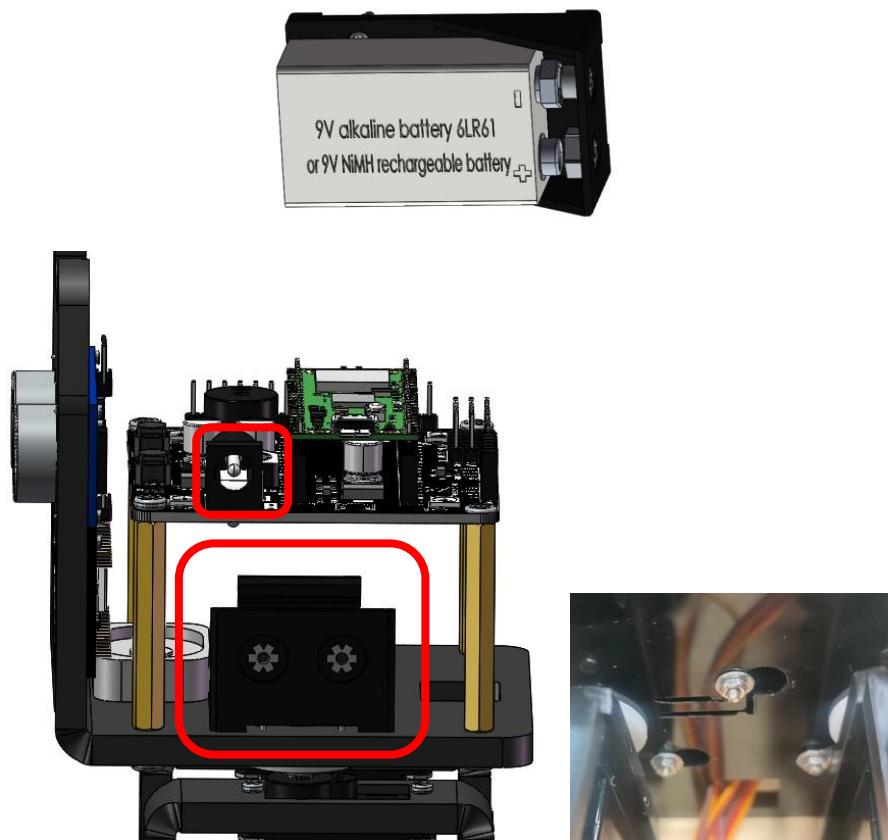
## Installing Bluetooth Module

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



## Installing 9V Battery Holder (with batteries 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 batteries, 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; which is more environment-friendly.

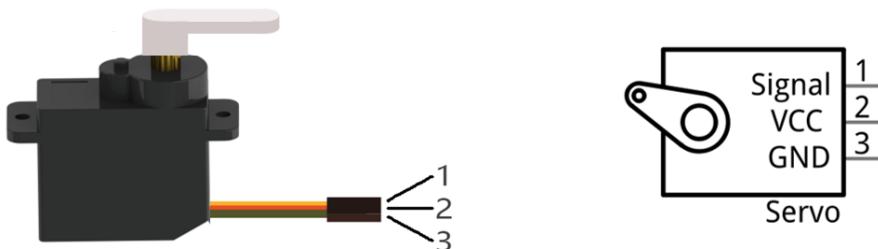
# Chapter 1 Module test

If you have any concerns, please feel free to contact us via [support@freenove.com](mailto: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.

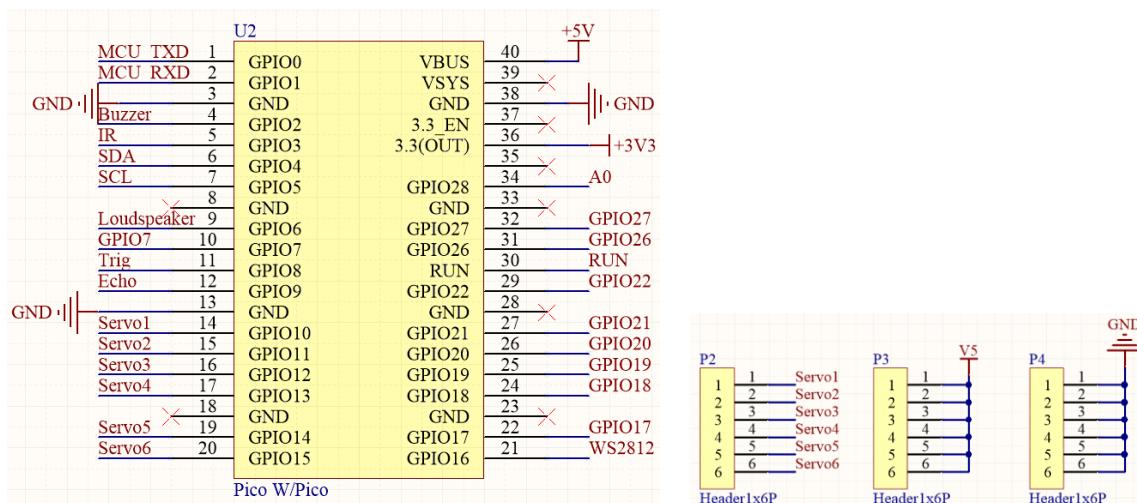


We will use a 50Hz PWM signal with a duty cycle in a certain range to drive the Servo. The lasting time of 0.5ms-2.5ms of PWM single cycle high level corresponds to the servo angle 0 degrees - 180 degree linearly. Part of the corresponding values are as follows:

High level time	Servo angle
0.5ms	0 degree
1ms	45 degree
1.5ms	90 degree
2ms	135 degree
2.5ms	180 degree

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

## Schematic



## Calibration for the Four Servos

To ensure stable movements of the robot, please calibrate the servos after assembling it. Here are three calibration methods for your reference. Please choose one at your preference.

- Method 1: Calibrating via the serial monitor on Arduino IDE. Please refer to [1.1.1\\_Servo\\_Calibration](#).
- Method 2: Calibrating with the PC tool, Refer to [1.1.2\\_Servo\\_Calibration\\_PC\\_Tools](#) for details.
- Method 3: Calibrating via Bluetooth. Refer to [1.1.3\\_Servo\\_Calibration\\_Bluetooth](#) for details,

It is recommended using Method 3 to calibrate. As it is a wireless way, it can avoid the interruption of the USB cable to the robot's movements, and the operation is more convenient and efficient.

### 1.1.1\_Servo\_Calibration

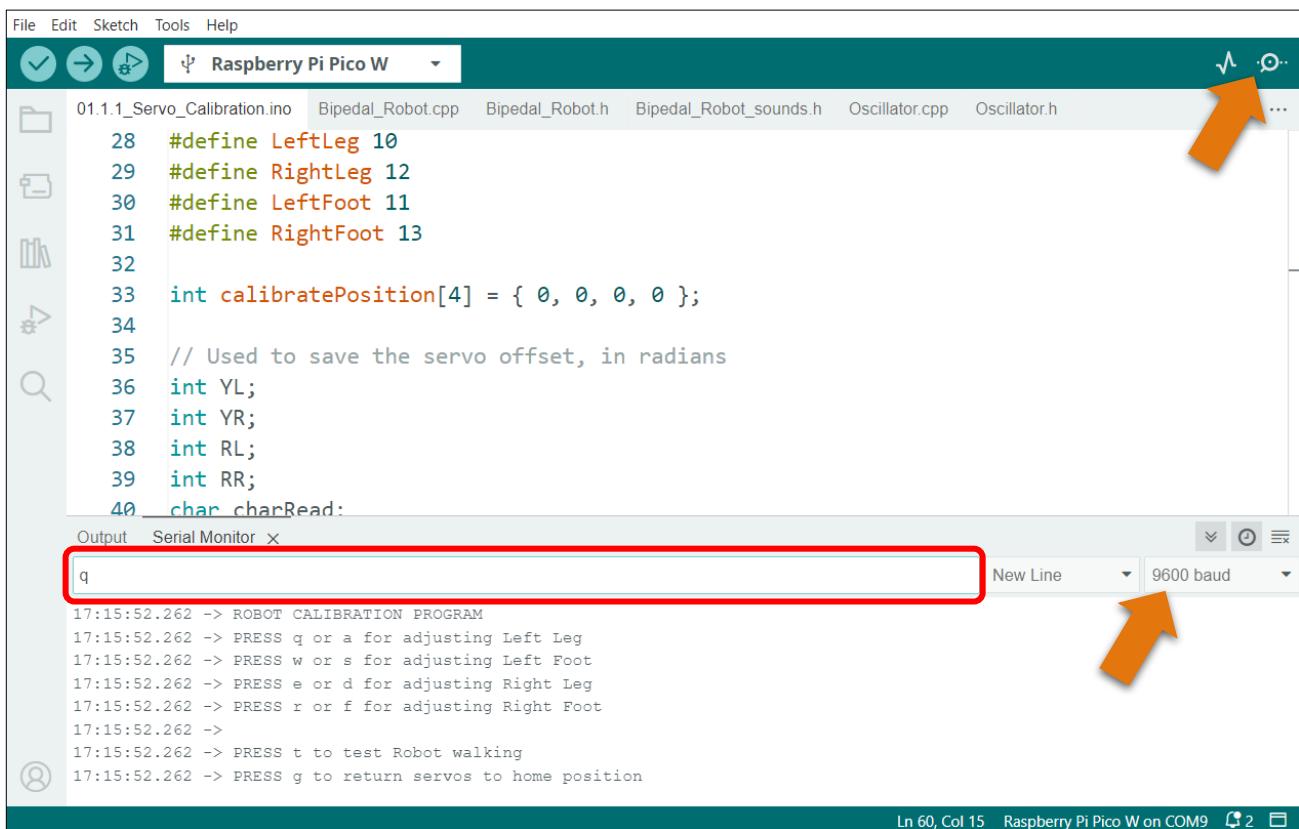
#### Sketch

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”.

Freenove_Bipedal_Robot_Kit_for_Raspberry_Pi_Pico-main > Sketches >			
Name	Date modified	Type	Size
00.0_Servo_90	11/14/2024 10:51 AM	File folder	
01.1.1_Servo_Calibration	11/14/2024 10:51 AM	File folder	
01.1.2_Servo_Calibration_PC_Tools	11/14/2024 10:51 AM	File folder	
01.1.3_Servo_Calibration_Bluetooth	11/14/2024 10:51 AM	File folder	
01.1.4_Servo	11/14/2024 10:51 AM	File folder	
01.2_Buzzer	11/14/2024 10:51 AM	File folder	

Open the Serial Monitor, set the baud rate to 9600, and input the designated values to calibrate the servos.



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>
2 #include <Wire.h>
3 #include <SoftwareSerial.h>
4 #include <EEPROM.h>
5 #include "Bipedal_Robot.h"
6 Bipedal_Robot;
7
8 #define LeftLeg 10
9 #define RightLeg 12
10#define LeftFoot 11
11#define RightFoot 13
12
13 int calibratePosition[4] = { 0, 0, 0, 0 };
14 // Used to save the servo offset, in radians
15 int YL;
16 int YR;
17 int RL;
18 int RR;
19 char charRead;
20 void setup() {
21   EEPROM.begin(512);
22   Serial.begin(9600);
23   for (int i = 0; i < 4; i++) {
24     EEPROM.write(i, calibratePosition[i]);
25   }
26   Bipedal_Robot.init(LeftLeg, RightLeg, LeftFoot, RightFoot, true); //Set the servo pins
27   YL = EEPROM.read(0);
28   if (YL > 128) YL -= 256;
29   YR = EEPROM.read(1);
30   if (YR > 128) YR -= 256;
31   RL = EEPROM.read(2);
32   if (RL > 128) RL -= 256;
33   RR = EEPROM.read(3);
34   if (RR > 128) RR -= 256;
35   calib_homePos();
36   Bipedal_Robot.saveTrimsOnEEPROM();
37   EEPROM.commit();
38   Bipedal_Robot.home();
39   Serial.println("ROBOT CALIBRATION PROGRAM");
40   Serial.println("PRESS q or a for adjusting Left Leg");
41   Serial.println("PRESS w or s for adjusting Left Foot");
42   Serial.println("PRESS e or d for adjusting Right Leg");
43   Serial.println("PRESS r or f for adjusting Right Foot");
```

```
44 Serial.println();
45 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

Included the required header file.

```

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

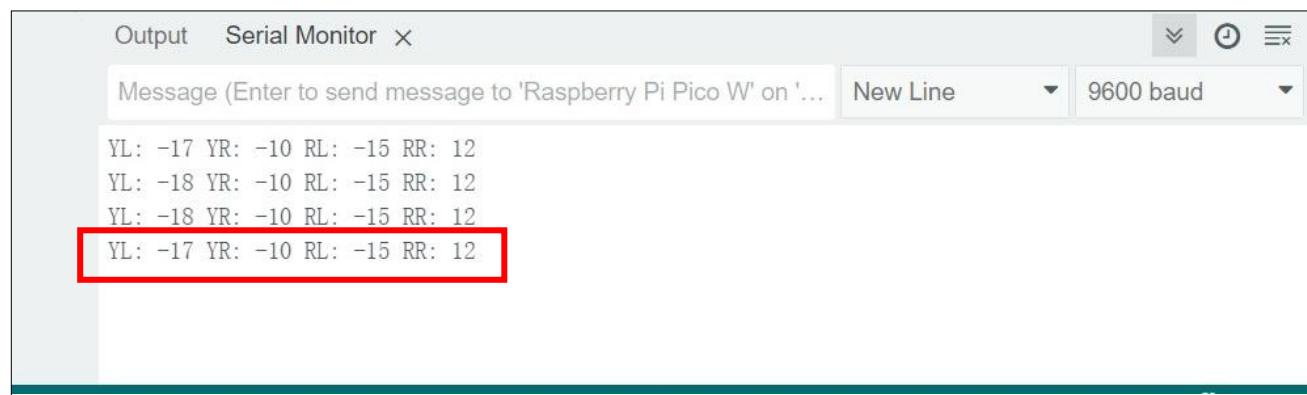
The serial monitor prints the current servo calibration value when receiving data.

```

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 }
```

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 };
```



You have now finished calibrating the servos. You can [skip](#) other calibration methods.

## 1.1.2\_Servo\_Calibration\_PC\_Tools

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 pyserial1-3.5-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
```

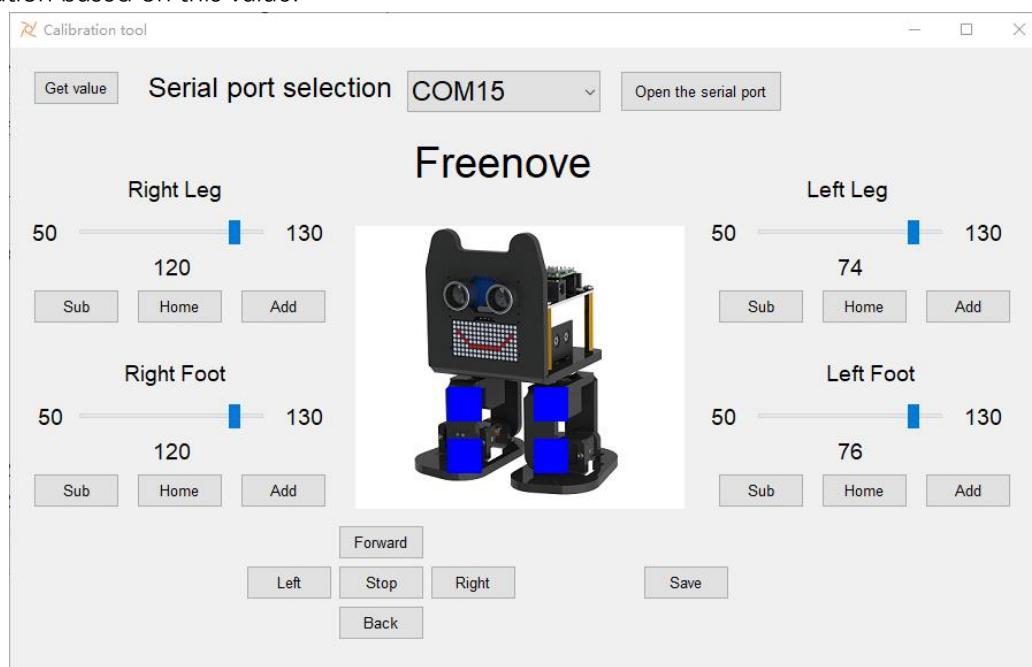
You will see the calibration tool open after the above steps.

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. Connect the Raspberry Pi Pico to your computer with the USB cable.
2. 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.)
3. 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.
4. After all the four servos are calibrated, click the movement buttons (Forward, Back, Left, Right, Stop) to see if the servos work correctly.
5. If they work as expected, click the Save button.
6. 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 rotate back to the last calibration value. You can calibration based on this value.



## Sketch

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.

Open “01.1.2\_Servo\_Calibration\_PC\_Tools” in  
**“Freenove\_Bipedal\_Robot\_Kit\_for\_Raspberry\_Pi\_Pico\Sketches”** and double-click  
“01.1.2\_Servo\_Calibration\_PC\_Tools.ino”.

Freenove_Bipedal_Robot_Kit_for_Raspberry_Pi_Pico-main > Sketches >			
Name	Date modified	Type	Size
00.0_Servo_90	11/14/2024 10:51 AM	File folder	
01.1.1_Servo_Calibration	11/14/2024 10:51 AM	File folder	
01.1.2_Servo_Calibration_PC_Tools	11/14/2024 10:51 AM	File folder	
01.1.3_Servo_Calibration_Bluetooth	11/14/2024 10:51 AM	File folder	
01.1.4_Servo	11/14/2024 10:51 AM	File folder	
01.2_Buzzer	11/14/2024 10:51 AM	File folder	

## 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
8 #define LeftLeg 10
9 #define RightLeg 12
10 #define LeftFoot 11
11 #define RightFoot 13
12
13 int calibratePosition[4] = { 0, 0, 0, 0 };
14 // Used to save the servo offset, in radians
15 int YL;
16 int YR;
17 int RL;
18 int RR;
19
20 String inputString = "";      // a String to hold incoming data
21 bool stringComplete = false; // whether the string is complete

```

```
22  
23 #define COMMANDS_COUNT_MAX 8  
24 char charRead;  
25 String inputStringBLE = "";  
26  
27 int move = 0;  
28  
29 #define ACTION_calibratio 'G' //Ant movement commands  
30 #define ACTION_flag 'F' //Expression control commands  
31 #define ACTION_MOVE 'A'  
32  
33 #define INTERVAL_CHAR '#' //The directive resolves the separator character  
34 String inputCommandArray[COMMANDS_COUNT_MAX];  
35 int paramters[COMMANDS_COUNT_MAX];  
36  
37 //Parses the Bluetooth data received by the serial port  
38 void Deal_Serial_Data(void) {  
39     String inputStringTemp = inputStringBLE;  
40     int string_length = inputStringTemp.length();  
41     for (int i = 0; i < COMMANDS_COUNT_MAX; i++) {  
42         int index = inputStringTemp.indexOf(INTERVAL_CHAR);  
43         if (index < 0) {  
44             if (string_length > 0) {  
45                 inputCommandArray[i] = inputStringTemp; //Get command  
46                 paramters[i] = inputStringTemp.toInt(); //Get parameters  
47             }  
48             break;  
49         }  
50         inputCommandArray[i] = inputStringTemp.substring(0, index);  
51         inputStringTemp = inputStringTemp.substring(index + 1);  
52         paramters[i] = inputCommandArray[i].toInt();  
53     }  
54     stringComplete = false;  
55     inputStringBLE = "";  
56 }  
57  
58 void setup() {  
59     EEPROM.begin(512);  
60     Serial.begin(9600);  
61     // for (int i = 0; i < 4; i++) {  
62     //     EEPROM.write(i, calibratePosition[i]);  
63     // }  
64     Bipedal_Robot.init(LeftLeg, RightLeg, LeftFoot, RightFoot, true); //Set the servo pins  
65     YL = EEPROM.read(0);
```

```
66 if (YL > 128) YL -= 256;
67 YR = EEPROM.read(1);
68 if (YR > 128) YR -= 256;
69 RL = EEPROM.read(2);
70 if (RL > 128) RL -= 256;
71 RR = EEPROM.read(3);
72 if (RR > 128) RR -= 256;
73 calib_homePos();
74 Bipedal_Robot.saveTrimsOnEEPROM();
75 EEPROM.commit();
76 Bipedal_Robot.home();
77 }

78

79 void loop() {
80 while (Serial.available()) {
81     inputStringBLE = Serial.readStringUntil('\n');
82     stringComplete = true;
83 }
84 if (stringComplete) { //Get the Bluetooth command
85     Deal_Serial_Data();
86     char commandChar = inputCommandArray[0].charAt(0);
87     if (commandChar == ACTION_calibratio) { //Control expression module display
88         // Serial.println("paramters[1]: " + String(paramters[1]) + " paramters[2]: " +
89         String(paramters[2]) + " paramters[3]: " + String(paramters[3]) + " paramters[4]: " +
90         String(paramters[4]) + " paramters[5]: " + String(paramters[5]));
91         YL = constrain(paramters[1] - 90, -40, 50);
92         YR = constrain(paramters[2] - 90, -40, 50);
93         RL = constrain(paramters[3] - 90, -40, 50);
94         RR = constrain(paramters[4] - 90, -40, 50);
95         Bipedal_Robot.setTrims(YL, YR, RL, RR);
96         calib_homePos();
97         if (paramters[5] = 1) {
98             Bipedal_Robot.setTrims(YL, YR, RL, RR);
99             calib_homePos();
100            Bipedal_Robot.saveTrimsOnEEPROM();
101        }
102    }
103    if (commandChar == ACTION_flag) { //Control expression module display
104        if (paramters[1] = 1) {
105            YL = EEPROM.read(0);
106            if (YL > 128) YL -= 256;
107            YR = EEPROM.read(1);
108            if (YR > 128) YR -= 256;
109            RL = EEPROM.read(2);
```

```

110     if (RL > 128) RL == 256;
111     RR = EEPROM.read(3);
112     if (RR > 128) RR == 256;
113     YL = constrain(YL + 90, 50, 140);
114     YR = constrain(YR + 90, 50, 140);
115     RL = constrain(RL + 90, 50, 140);
116     RR = constrain(RR + 90, 50, 140);
117     Serial.print("F#" + String(YL) + "#" + String(YR) + "#" + String(RL) + "#" + String(RR));
118   }
119 }
120 if (commandChar == ACTION_MOVE) { //Control expression module display
121   move = paramters[1];
122 }
123 }
124 if (move == 1) {
125   Bipedal_Robot.walk(1, 1500, 1);
126 }
127 if (move == 2) {
128   Bipedal_Robot.walk(1, 1500, -1);
129 }
130 if (move == 3) {
131   Bipedal_Robot.turn(1, 1500, 1);
132 }
133 if (move == 4) {
134   Bipedal_Robot.turn(1, 1500, -1);
135 }
136 if (move == 0) {
137   Bipedal_Robot.home();
138 }
139 }
140
141 void calib_homePos() {
142   int servoPos[4];
143   servoPos[0] = 90;
144   servoPos[1] = 90;
145   servoPos[2] = 90;
146   servoPos[3] = 90;
147   Bipedal_Robot._moveServos(500, servoPos);
148 }
```

### Code Explanation

Parses the serial port data received by the serial port.

```

38 void Deal_Serial_Data(void) {
39   String inputStringTemp = inputStringBLE;
40   int string_length = inputStringTemp.length();
```

```

41   for (int i = 0; i < COMMANDS_COUNT_MAX; i++) {
42     int index = inputStringTemp.indexOf(INTERVAL_CHAR);
43     if (index < 0) {
44       if (string_length > 0) {
45         inputCommandArray[i] = inputStringTemp; //Get command
46         paramters[i] = inputStringTemp.toInt(); //Get parameters
47       }
48       break;
49     }
50     inputCommandArray[i] = inputStringTemp.substring(0, index);
51     inputStringTemp = inputStringTemp.substring(index + 1);
52     paramters[i] = inputCommandArray[i].toInt();
53   }
54   stringComplete = false;
55   inputStringBLE = "";
56 }
```

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

	<pre> 87     if (commandChar == ACTION_calibratio) { //Control expression module display 103    if (commandChar == ACTION_flag) { //Control expression module display 120    if (commandChar == ACTION_MOVE) { //Control expression module display </pre>
--	---

You have now finished calibrating the servos. You can [skip](#) other calibration methods.

### 1.1.3\_Servo\_Calibration\_Bluetooth

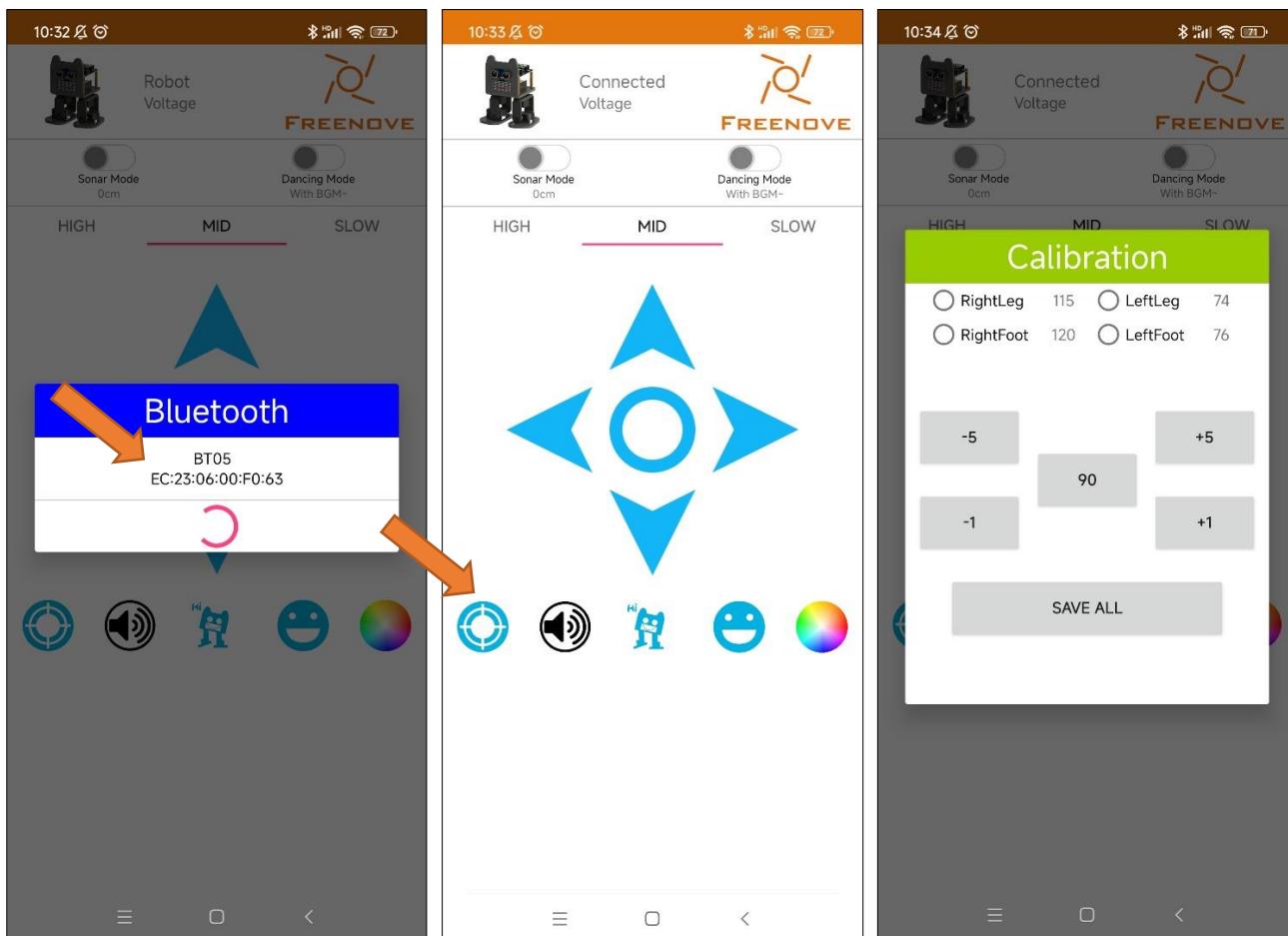
If you have not yet installed our APP, please do so first. You can download the app [here](#).

This method 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.



## Sketch

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.

Open "01.1.3\_Servo\_Calibration\_Bluetooth" in

"**Freenove\_Bipedal\_Robot\_Kit\_for\_Raspberry\_Pi\_Pico\Sketches**" and double-click  
"01.1.3\_Servo\_Calibration\_Bluetooth.ino".

Freenove_Bipedal_Robot_Kit_for_Raspberry_Pi_Pico-main > Sketches >			
Name	Date modified	Type	Size
00.0_Servo_90	11/14/2024 10:51 AM	File folder	
01.1.1_Servo_Calibration	11/14/2024 10:51 AM	File folder	
01.1.2_Servo_Calibration_PC_Tools	11/14/2024 10:51 AM	File folder	
01.1.3_Servo_Calibration_Bluetooth	11/14/2024 10:51 AM	File folder	
01.1.4_Servo	11/14/2024 10:51 AM	File folder	
01.2_Buzzer	11/14/2024 10:51 AM	File folder	

## Code

```

1 #include <Arduino.h>
2 #include <Wire.h>
3 #include <SoftwareSerial.h>
4 #include <EEPROM.h>
5 #include "SerialCommand.h"
6 #include "Bipodal_Robot.h"
7
8 #define LeftLeg 10
9 #define RightLeg 12
10 #define LeftFoot 11
11 #define RightFoot 13
12
13 int calibratePosition[4] = { 0, 0, 0, 0 };
14 // Used to save the servo offset, in radians
15 int YL;
16 int YR;
17 int RL;
18 int RR;
19
20 String inputString = ""; // a String to hold incoming data
21 bool stringComplete = false; // whether the string is complete
22 int baudRate[] = {2400, 9600, 19200, 115200};
23 int baudRateCount = sizeof(baudRate) / sizeof(baudRate[0]);
24
25 #define COMMANDS_COUNT_MAX 5
26 char charRead;
27 String inputStringBLE = "";
28
29 #define ACTION_calibratio 'G' //Ant movement commands
30 #define ACTION_flag 'F' //Expression control commands
31 #define ACTION_MOVE 'A'

```

```
32
33 #define INTERVAL_CHAR '#' //The directive resolves the separator character
34 String inputCommandArray[COMMANDS_COUNT_MAX];
35 int paramters[COMMANDS_COUNT_MAX];
36 int move = 0;
37 int movespeed = 1500;
38
39 SoftwareSerial BTserial = SoftwareSerial(1, 0); // RX pin to 0 and TX pin 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     Serial.println(inputStringTemp);
53     for (int i = 0; i < COMMANDS_COUNT_MAX; i++) {
54         int index = inputStringTemp.indexOf(INTERVAL_CHAR);
55         if (index < 0) {
56             if (19 > string_length > 0) {
57                 inputCommandArray[i] = inputStringTemp; //Get command
58                 paramters[i] = inputStringTemp.toInt(); //Get parameters
59             }
60             break;
61         }
62         inputCommandArray[i] = inputStringTemp.substring(0, index);
63         inputStringTemp = inputStringTemp.substring(index + 1);
64         paramters[i] = inputCommandArray[i].toInt();
65     }
66     stringComplete = false;
67     inputStringBLE = "";
68 }
69
70 void setup() {
71     EEPROM.begin(512);
72     Serial.begin(9600);
73     delay(2000);
74     // Loop through each baud rate and send commands
75     for(int i = 0; i < baudRateCount; i++) {
```

```
76    BTserial.begin(baudRate[i]);  
77    delay(200);  
78    BTserial.println("AT+NAME=BT05");//Set the radio called Robot  
79    delay(200);  
80    BTserial.println("AT+ROLE=0");  
81    delay(200);  
82    BTserial.println("AT+UART=2");//1=2400 2=9600 3=19200 4=115200  
83    delay(200);  
84  
85    //SoftwareSerial does not support dynamic adjustment of baud rate, so close the serial  
86    BTserial.end();  
87 }  
88  
89 BTserial.begin(9600);  
90 delay(1500);  
91  
92 inputString.reserve(200);  
93 SCmd.addDefaultHandler(receiveStop);  
94 Serial.println("Set the name of the Bluetooth module to BT05.");  
95 Serial.println("Set the Bluetooth baud rate to 9600.");  
96  
97 // for (int i = 0; i < 4; i++) {  
98 //   EEPROM.write(i, calibratePosition[i]);  
99 // }  
100 Bipedal_Robot.init(LeftLeg, RightLeg, LeftFoot, RightFoot, true); //Set the servo pins  
101 YL = EEPROM.read(0);  
102 if (YL > 128) YL = 256;  
103 YR = EEPROM.read(1);  
104 if (YR > 128) YR = 256;  
105 RL = EEPROM.read(2);  
106 if (RL > 128) RL = 256;  
107 RR = EEPROM.read(3);  
108 if (RR > 128) RR = 256;  
109 calib_homePos();  
110 Bipedal_Robot.saveTrimsOnEEPROM();  
111 EEPROM.commit();  
112 Bipedal_Robot.home();  
113 Serial.println("start...");  
114 }  
115  
116 void readservo(int parameter1, int parameter2, int parameter3, int parameter4) {  
117   parameter1 = EEPROM.read(0);  
118   parameter2 = EEPROM.read(1);  
119   parameter3 = EEPROM.read(2);
```

```
120     parameter4 = EEPROM.read(3);
121 }
122
123 void loop() {
124     if (BTserial.available()) {
125         inputStringBLE = BTserial.readStringUntil('\n');
126         stringComplete = true;
127     }
128     if (stringComplete) { //Get the Bluetooth command
129         Deal_Serial_Data();
130         char commandChar = inputCommandArray[0].charAt(0);
131         if (commandChar == ACTION_calibratio) { //Control expression module display
132             YL = constrain(paramters[1] - 90, -40, 50);
133             YR = constrain(paramters[2] - 90, -40, 50);
134             RL = constrain(paramters[3] - 90, -40, 50);
135             RR = constrain(paramters[4] - 90, -40, 50);
136             Serial.println("F#" + String(YL) + "#" + String(YR) + "#" + String(RL) + "#" +
137             String(RR));
138             Bipedal_Robot.setTrims(YL, YR, RL, RR);
139             calib_homePos();
140             if (paramters[5] == 1) {
141                 Bipedal_Robot.setTrims(YL, YR, RL, RR);
142                 calib_homePos();
143                 Bipedal_Robot.saveTrimsOnEEPROM();
144             }
145         }
146         if (commandChar == ACTION_flag) { //Control expression module display
147             if (paramters[1] == 1) {
148                 YL = EEPROM.read(0);
149                 if (YL > 128) YL -= 256;
150                 YR = EEPROM.read(1);
151                 if (YR > 128) YR -= 256;
152                 RL = EEPROM.read(2);
153                 if (RL > 128) RL -= 256;
154                 RR = EEPROM.read(3);
155                 if (RR > 128) RR -= 256;
156                 YL = constrain(YL + 90, 50, 140);
157                 YR = constrain(YR + 90, 50, 140);
158                 RL = constrain(RL + 90, 50, 140);
159                 RR = constrain(RR + 90, 50, 140);
160                 Serial.println("F#" + String(YL) + "#" + String(YR) + "#" + String(RL) + "#" +
161                 String(RR));
162                 BTserial.println("F#" + String(YL) + "#" + String(YR) + "#" + String(RL) + "#" +
163                 String(RR));
```

```

164     }
165   }
166   if (commandChar == ACTION_MOVE) { //Control expression module display
167     move = paramters[1];
168     movespeed = paramters[2];
169     if (movespeed == 0) {
170       movespeed = 1000;
171     } else if (movespeed == 1) {
172       movespeed = 1500;
173     } else if (movespeed == 2) {
174       movespeed = 2000;
175     } else
176       movespeed = 1500;
177   }
178 }
179 if (move == 1) {
180   Bipedal_Robot.walk(1, movespeed, 1);
181 }
182 if (move == 2) {
183   Bipedal_Robot.walk(1, movespeed, -1);
184 }
185 if (move == 3) {
186   Bipedal_Robot.turn(1, movespeed, 1);
187 }
188 if (move == 4) {
189   Bipedal_Robot.turn(1, movespeed, -1);
190 }
191 if (move == 0) {
192   Bipedal_Robot.home();
193 }
194 }

195 void calib_homePos() {
196   int servoPos[4];
197   servoPos[0] = 90;
198   servoPos[1] = 90;
199   servoPos[2] = 90;
200   servoPos[3] = 90;
201   Bipedal_Robot._moveServos(100, servoPos);
202 }
203 }
```

### Code Explanation

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

73	delay(2000);
----	--------------

```

74 // Loop through each baud rate and send commands
75 for(int i = 0; i < baudRateCount; i++) {
76     BTserial.begin(baudRate[i]);
77     delay(200);
78     BTserial.println("AT+NAME=BT05");//Set the radio called Robot
79     delay(200);
80     BTserial.println("AT+ROLE=0");
81     delay(200);
82     BTserial.println("AT+UART=2");//1=2400 2=9600 3=19200 4=115200
83     delay(200);
84
85     //SoftwareSerial does not support dynamic adjustment of baud rate, so close the serial
86     BTserial.end();
87 }
88
89 BTserial.begin(9600);
90 delay(1500);

```

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

```

124 if (BTserial.available()) {
125     inputStringBLE = BTserial.readStringUntil('\n');
126     stringComplete = true;
127 }
128 if (stringComplete) { //Get the Bluetooth command
129     Deal_Serial_Data();
130     char commandChar = inputCommandArray[0].charAt(0);
131     if (commandChar == ACTION_calibratio) { //Control expression module display
132         YL = constrain(paramters[1] - 90, -40, 50);
133         YR = constrain(paramters[2] - 90, -40, 50);
134         RL = constrain(paramters[3] - 90, -40, 50);
135         RR = constrain(paramters[4] - 90, -40, 50);
136         Serial.println("F#" + String(YL) + "#" + String(YR) + "#" + String(RL) + "#" +
137 String(RR));
138         Bipedal_Robot.setTrims(YL, YR, RL, RR);
139         calib_homePos();
140         if (paramters[5] == 1) {
141             Bipedal_Robot.setTrims(YL, YR, RL, RR);
142             calib_homePos();
143             Bipedal_Robot.saveTrimsOnEEPROM();
144         }
145     }
146     if (commandChar == ACTION_flag) { //Control expression module display
147         if (paramters[1] == 1) {
148             YL = EEPROM.read(0);

```

```

149     if (YL > 128) YL -= 256;
150     YR = EEPROM.read(1);
151     if (YR > 128) YR -= 256;
152     RL = EEPROM.read(2);
153     if (RL > 128) RL -= 256;
154     RR = EEPROM.read(3);
155     if (RR > 128) RR -= 256;
156     YL = constrain(YL + 90, 50, 140);
157     YR = constrain(YR + 90, 50, 140);
158     RL = constrain(RL + 90, 50, 140);
159     RR = constrain(RR + 90, 50, 140);
160     Serial.println("F#" + String(YL) + "#" + String(YR) + "#" + String(RL) + "#" +
161     String(RR));
162     BTserial.println("F#" + String(YL) + "#" + String(YR) + "#" + String(RL) + "#" +
163     String(RR));
164   }
165 }
166 if (commandChar == ACTION_MOVE) { //Control expression module display
167   move = paramters[1];
168   movespeed = paramters[2];
169   if (movespeed == 0) {
170     movespeed = 1000;
171   } else if (movespeed == 1) {
172     movespeed = 1500;
173   } else if (movespeed == 2) {
174     movespeed = 2000;
175   } else
176     movespeed = 1500;
177 }
178 }
```

## 1.1.4\_Servo

### 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”.

Freenove_Bipedal_Robot_Kit_for_Raspberry_Pi_Pico-main > Sketches >			
Name	Date modified	Type	Size
00.0_Servo_90	11/14/2024 10:51 AM	File folder	
01.1.1_Servo_Calibration	11/14/2024 10:51 AM	File folder	
01.1.2_Servo_Calibration_PC_Tools	11/14/2024 10:51 AM	File folder	
01.1.3_Servo_Calibration_Bluetooth	11/14/2024 10:51 AM	File folder	
01.1.4_Servo	11/14/2024 10:51 AM	File folder	
01.2_Buzzer	11/14/2024 10:51 AM	File folder	

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

```
1 #include <Arduino.h>
2 #include <EEPROM.h>
3 #include "Bipedal_Robot.h"
4 Bipedal_Robot Bipedal_Robot;
5
6 #define LeftLeg 10
7 #define RightLeg 12
8 #define LeftFoot 11
9 #define RightFoot 13
10
11 // int calibratePosition[4] = { -17, -10, -15, 12 };
12
13 void setup() {
14     Serial.begin(115200);
15     EEPROM.begin(512);
16     Bipedal_Robot.init(LeftLeg, RightLeg, LeftFoot, RightFoot, true); //Set the servo pins
17     // Bipedal_Robot.setTrims(calibratePosition[0], calibratePosition[1], calibratePosition[2],
18     calibratePosition[3]);
18     calib_homePos();
19     Bipedal_Robot.saveTrimsOnEEPROM();
20     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 the main loop, the robot moves forward and backward, turn left, and turn right repeatedly.

```

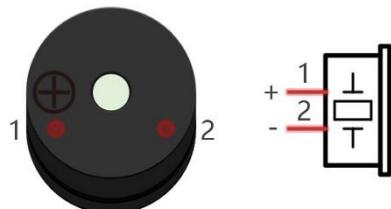
25 Serial.println("loop.....");
26 Bipedal_Robot.walk(2, 2000, 1);
27 Bipedal_Robot.home();
28 delay(1000);
29 Bipedal_Robot.walk(2, 2000, -1);
30 Bipedal_Robot.home();
31 delay(1000);
32 Bipedal_Robot.turn(2, 2000, 1);
33 Bipedal_Robot.home();
34 delay(1000);
35 Bipedal_Robot.turn(2, 2000, -1);
36 Bipedal_Robot.home();
37 delay(1000);
```

## 1.2 Buzzer

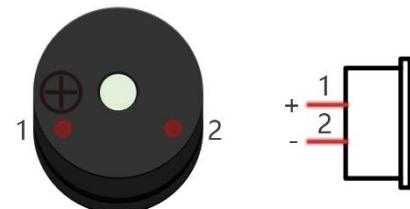
### Buzzer

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 2 kHz, which means the passive buzzer is loudest when its resonant frequency is 2 kHz.

#### 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



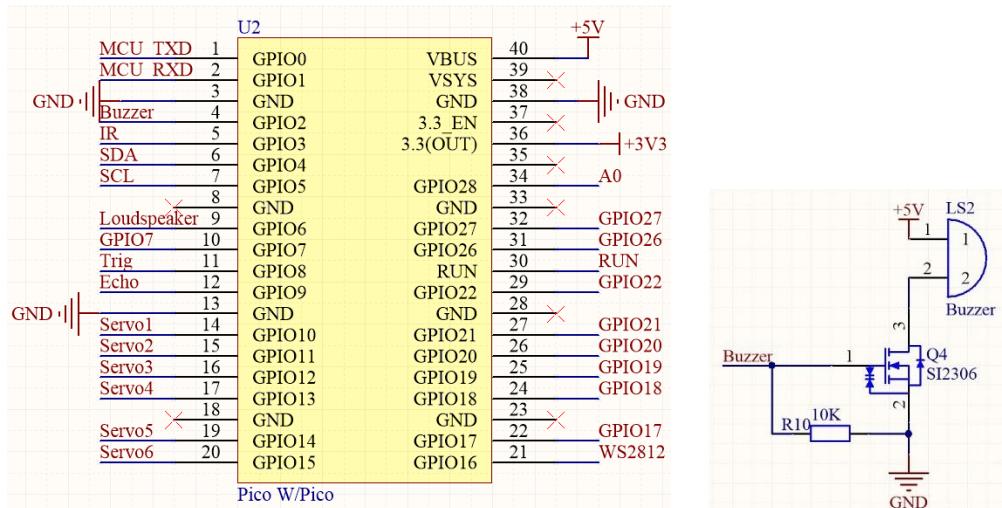
Passive buzzer



The buzzer used in this robot is a passive buzzer that can make sounds with different frequency.

## Schematic

As we can see, the buzzer is controlled by GPIO2 of Raspberry Pi Pico W. When the buzzer receives PWM signal, NMOS will be activated to make the buzzer sound. When the buzzer receives no signal, it will be controlled at low level by R10 and NMOS 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”.

Freenove_Bipedal_Robot_Kit_for_Raspberry_Pi_Pico-main > Sketches			
Name	Date modified	Type	Size
01.1.4_Servo	11/14/2024 10:51 AM	File folder	
01.2_Buzzer	11/14/2024 10:51 AM	File folder	
01.3.1_Loudspeaker	11/14/2024 10:51 AM	File folder	
01.3.2_Music	11/14/2024 10:51 AM	File folder	
01.4_Battery_level	11/14/2024 10:51 AM	File folder	
01.5_Matrix	11/14/2024 10:51 AM	File folder	
01.6_WS2812	11/14/2024 10:51 AM	File folder	

## Code

```

1 #include "Freenove_Robot_For_Pico_W.h"
2
3 void setup() {
4     Buzzer_Setup();      //Buzzer initialization function
5     Buzzer_Alert(4, 3); //Control the buzzer to sound 3 times, 4 sounds each time

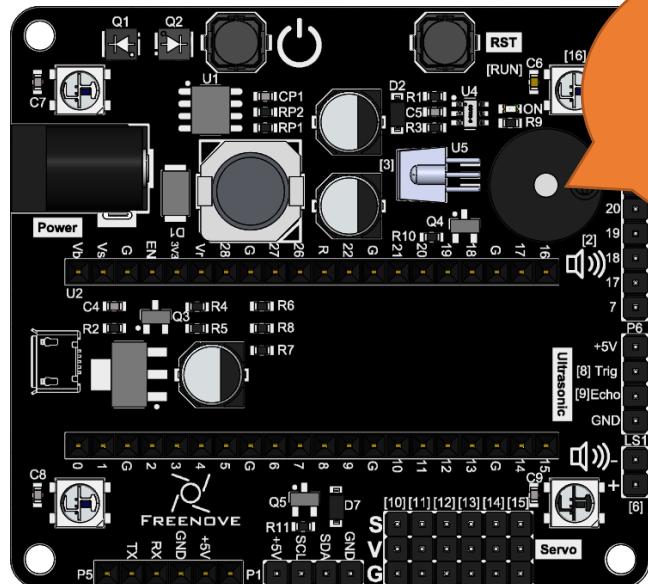
```

```

6 }
7
8 void loop() {
9     delay(1000);
10}

```

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

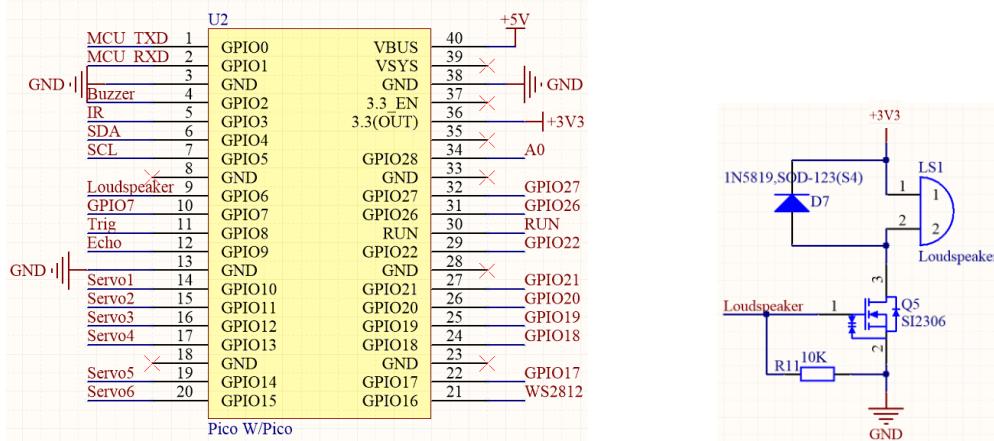
### Loudspeaker

Besides the buzzer, we still have a loud speaker 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 converter module to spare GPIO pins for extended use.

In this project, we use the speaker to play a piece of music. If you just use the speaker 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).



### 1.3.1\_Loudspeaker

#### Sketch

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”.

Freenove_Bipedal_Robot_Kit_for_Raspberry_Pi_Pico-main > Sketches			
Name	Date modified	Type	Size
01.1.4_Servo	11/14/2024 10:51 AM	File folder	
01.2_Buzzer	11/14/2024 10:51 AM	File folder	
01.3.1_Loudspeaker	11/14/2024 10:51 AM	File folder	
01.3.2_Music	11/14/2024 10:51 AM	File folder	
01.4_Battery_level	11/14/2024 10:51 AM	File folder	
01.5_Matrix	11/14/2024 10:51 AM	File folder	
01.6_WS2812	11/14/2024 10:51 AM	File folder	

#### Code

```
1 #include "Freenove_Robot_For_Pico_W.h"
2
3 void setup() {
4     Buzzer_Setup();      //Buzzer initialization function
5     Buzzer_Alert(4, 3); //Control the buzzer to sound 3 times, 4 sounds each time
6 }
7
8 void loop() {
9     delay(1000);
10}
```

#### Code Explanation

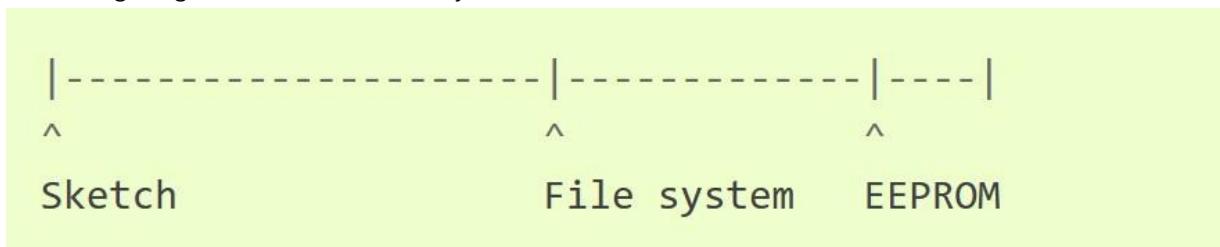
This code is the same as the buzzer code, except for the different GPIO used. In the buzzer example, we use GPIO 2 and in this one, we use GPIO 6.

```
#define PIN_BUZZER 6
```

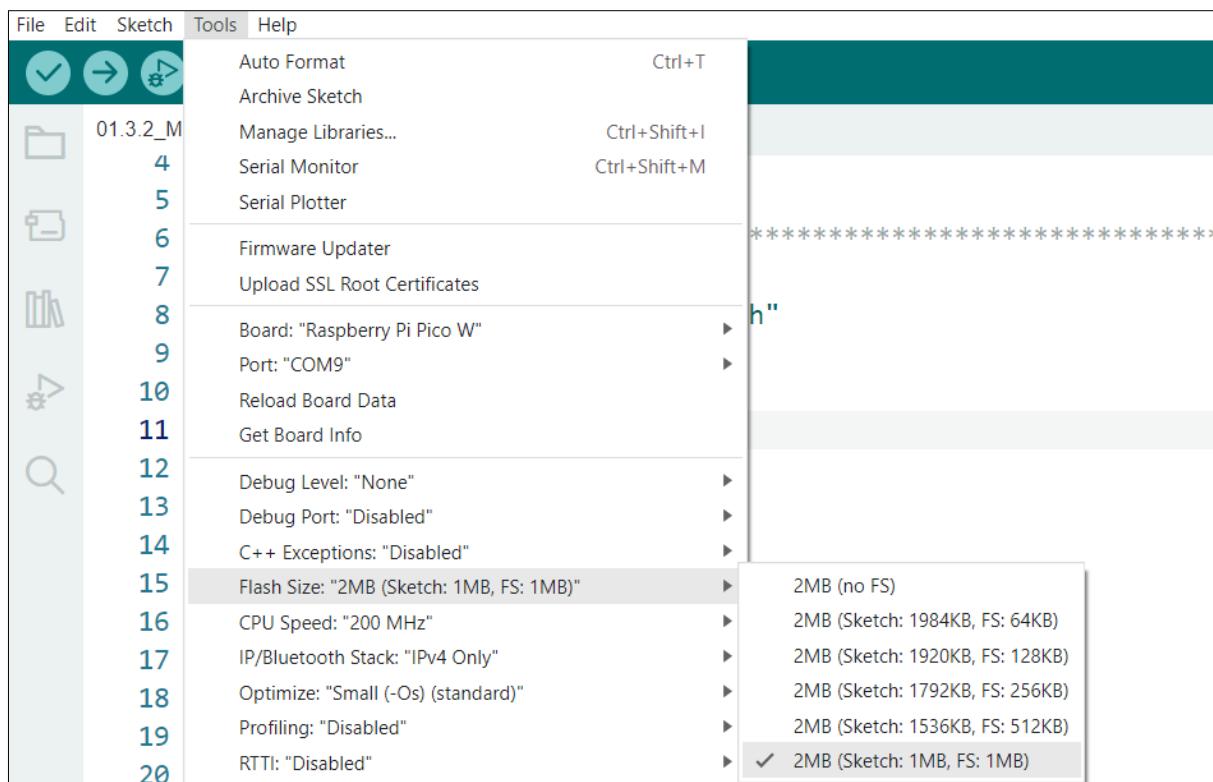
### 1.3.2\_Music

In this section, we will use GPIO6 of the Raspberry Pi Pico to play audio. After the code upload successfully, it will 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) file system, with an API that's compatible with the onboard flash file system.

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



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 storage, 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.

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”.

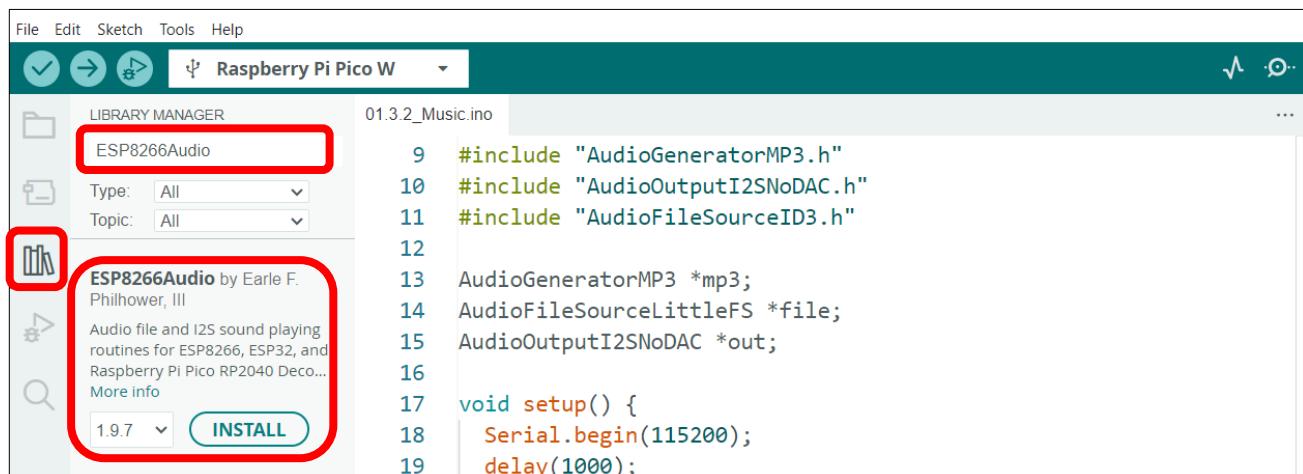
Freenove_Bipedal_Robot_Kit_for_Raspberry_Pi_Pico-main > Sketches				
Name	Date modified	Type	Size	
01.1.4_Servo	11/14/2024 10:51 AM	File folder		
01.2_Buzzer	11/14/2024 10:51 AM	File folder		
01.3.1_Loudspeaker	11/14/2024 10:51 AM	File folder		
01.3.2_Music	11/14/2024 10:51 AM	File folder		
01.4_Battery_level	11/14/2024 10:51 AM	File folder		
01.5_Matrix	11/14/2024 10:51 AM	File folder		
01.6_WS2812	11/14/2024 10:51 AM	File folder		

## Install ESP8266Audio library

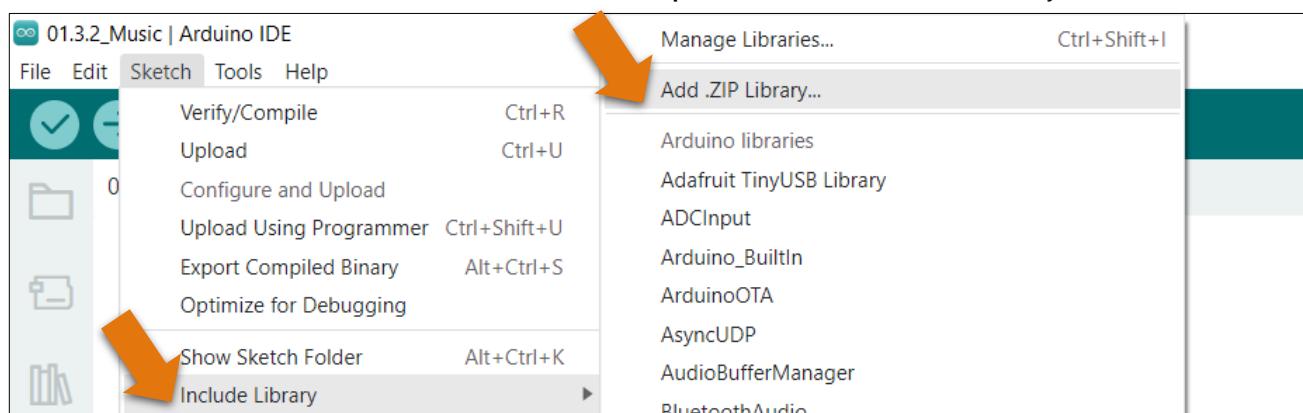
In this project, we use a third-party library named **ESP8266Audio**. Please install it first.

Open Arduino IDE, click **Library Manager** on the left, and search “**ESP8266Audio**” to install.

It is recommended to use version **1.9.7** of the ESP8266Audio library to avoid compatibility issues.



The second way, open Arduino IDE, click Sketch→Include Library→Add .ZIP Library. In the pop-up window, find the file named “./Libraries/**ESP8266Audio-V1.9.7.Zip**” which locates in this directory, and click OPEN.



Name	Date modified	Size
Adafruit_NeoPixel-V1.12.5.zip	9/6/2025 9:02 AM	87 KB
Adafruit-GFX_V1.12.1.zip	9/6/2025 8:58 AM	349 KB
<b>ESP8266Audio-V1.9.7.zip</b>	9/6/2025 9:02 AM	7,648 KB
Freenove.VK16K33.Lib-V1.0.0.zip	9/6/2025 9:02 AM	14 KB
IRremote_V4.4.1_20241029.zip	9/6/2025 9:02 AM	1,350 KB

## 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-1.5.4.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.

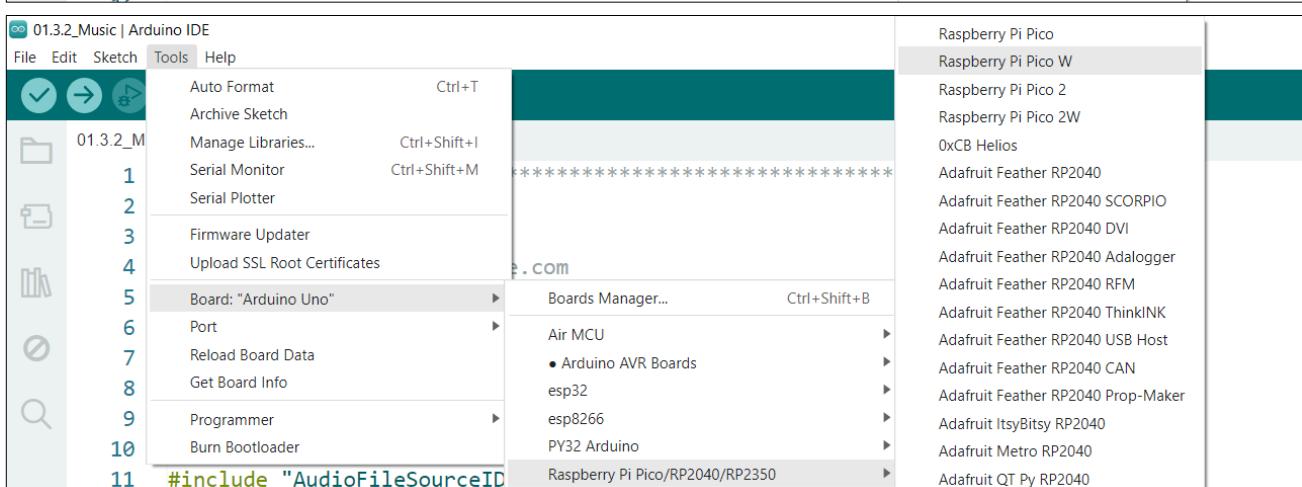
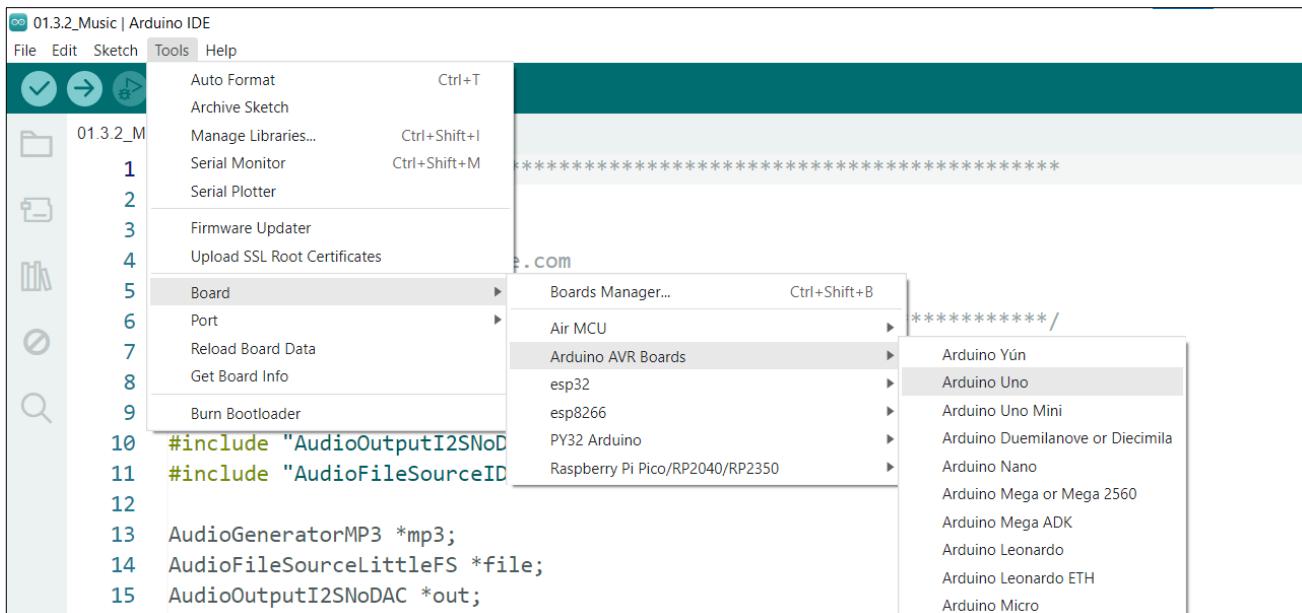
After copying the file, restart Arduino IDE.

Name	Date modified	Type
PicoLittleFS	11/14/2024 10:51 AM	File folder
<b>arduino-littlefs-upload-1.5.4.vsix</b>	4/30/2025 8:56 AM	VSIX File

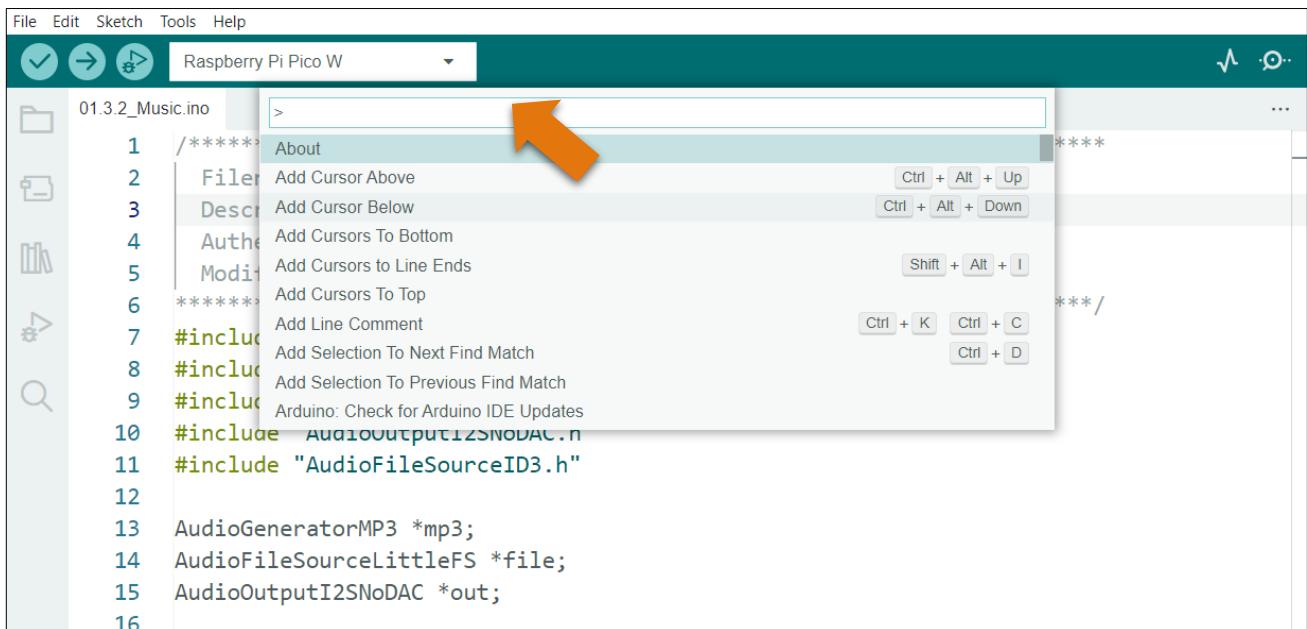
Name	Date modified	Type	Size
deployedPlugins	4/3/2025 8:54 AM	File folder	
globalStorage	4/3/2025 8:54 AM	File folder	
localization-cache	4/30/2025 8:45 AM	File folder	
logs	4/30/2025 9:02 AM	File folder	
<b>plugins</b>	4/30/2025 8:58 AM	File folder	
plugin-storage	1/25/2025 9:17 AM	File folder	
workspace-storage	4/30/2025 8:45 AM	File folder	
arduino-cli.yaml	1/25/2025 4:31 PM	Yaml 源文件	1 KB
pluggable-monitor-settings.json	4/29/2025 6:08 PM	JSON 源文件	14 KB
recent-sketches.json	4/30/2025 9:02 AM	JSON 源文件	2 KB
recentworkspace.json	4/30/2025 9:02 AM	JSON 源文件	6 KB
settings.json	4/30/2025 9:02 AM	JSON 源文件	1 KB

C:\Users\freenove-kf03\arduinoIDE\plugins			
Name	Date modified	Type	Size
arduino-littlefs-upload-1.5.4.vsix	4/30/2025 8:56 AM	VSIX File	1,057 KB

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.

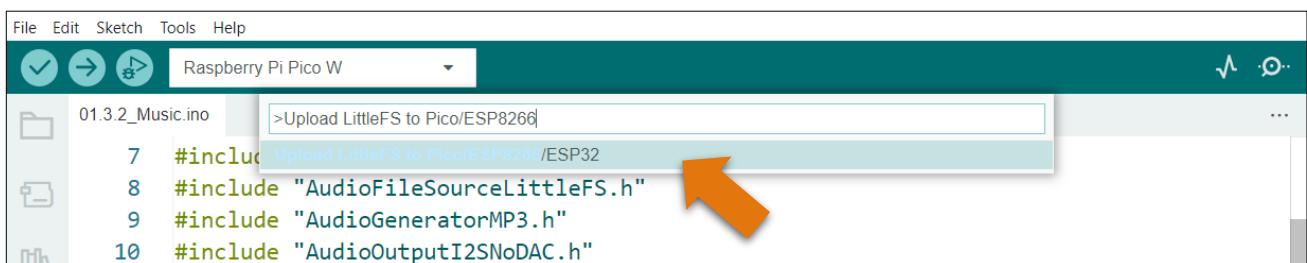


Here is how to use the PicoLittleFS tool, press [Ctrl]+[Shift]+[P] simultaneously, enter **Upload LittleFS to Pico/ESP8266** on the input field.



The screenshot shows the Arduino IDE interface with the title bar "Raspberry Pi Pico W". A context menu is open over a portion of the code editor, with the "About" option highlighted. An orange arrow points from the top right towards the "About" option. The code editor displays the following sketch:

```
File Edit Sketch Tools Help
Raspberry Pi Pico W
01.3.2_Music.ino
1 // ****
2 File
3 Descr
4 Autho
5 Modi
6 ****
7 #includ
8 #includ
9 #includ
10 #include "AudioOutputI2SNoDAC.h"
11 #include "AudioFileSourceID3.h"
12
13 AudioGeneratorMP3 *mp3;
14 AudioFileSourceLittleFS *file;
15 AudioOutputI2SNoDAC *out;
16
```



The screenshot shows the Arduino IDE interface with the title bar "Raspberry Pi Pico W". A context menu is open over a portion of the code editor, with the "Upload LittleFS to Pico/ESP8266" option highlighted. An orange arrow points from the top right towards the "Upload LittleFS to Pico/ESP8266" option. The code editor displays the following sketch:

```
File Edit Sketch Tools Help
Raspberry Pi Pico W
01.3.2_Music.ino
1 >Upload LittleFS to Pico/ESP8266
2 #includ
3 #include "AudioFileSourceLittleFS.h"
4 #include "AudioGeneratorMP3.h"
5
6
7 #include "AudioOutputI2SNoDAC.h"
```

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

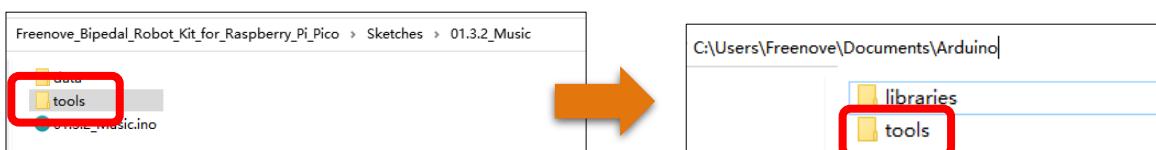
First, open the Arduino IDE, and then click File in Menus and select Preferences.



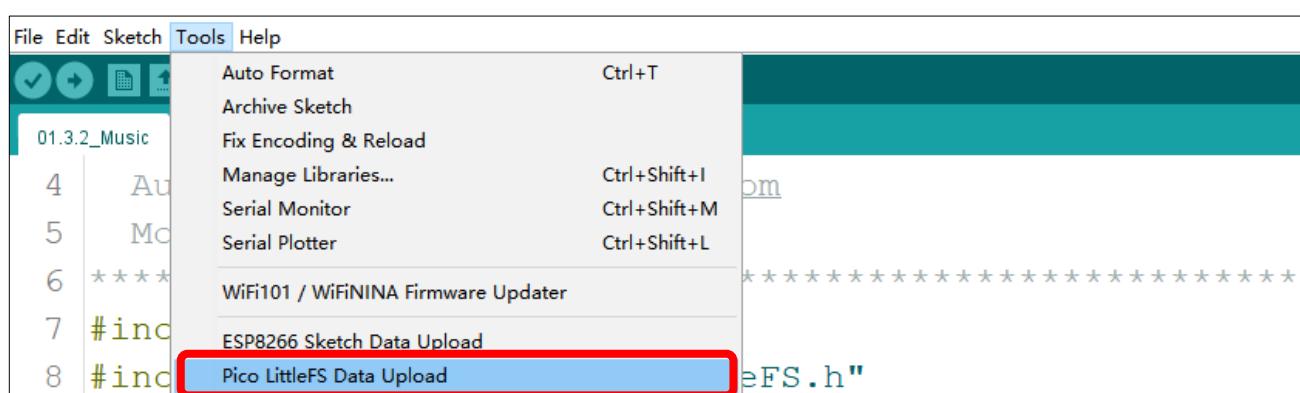
Find the Arduino IDE environment directory location.



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



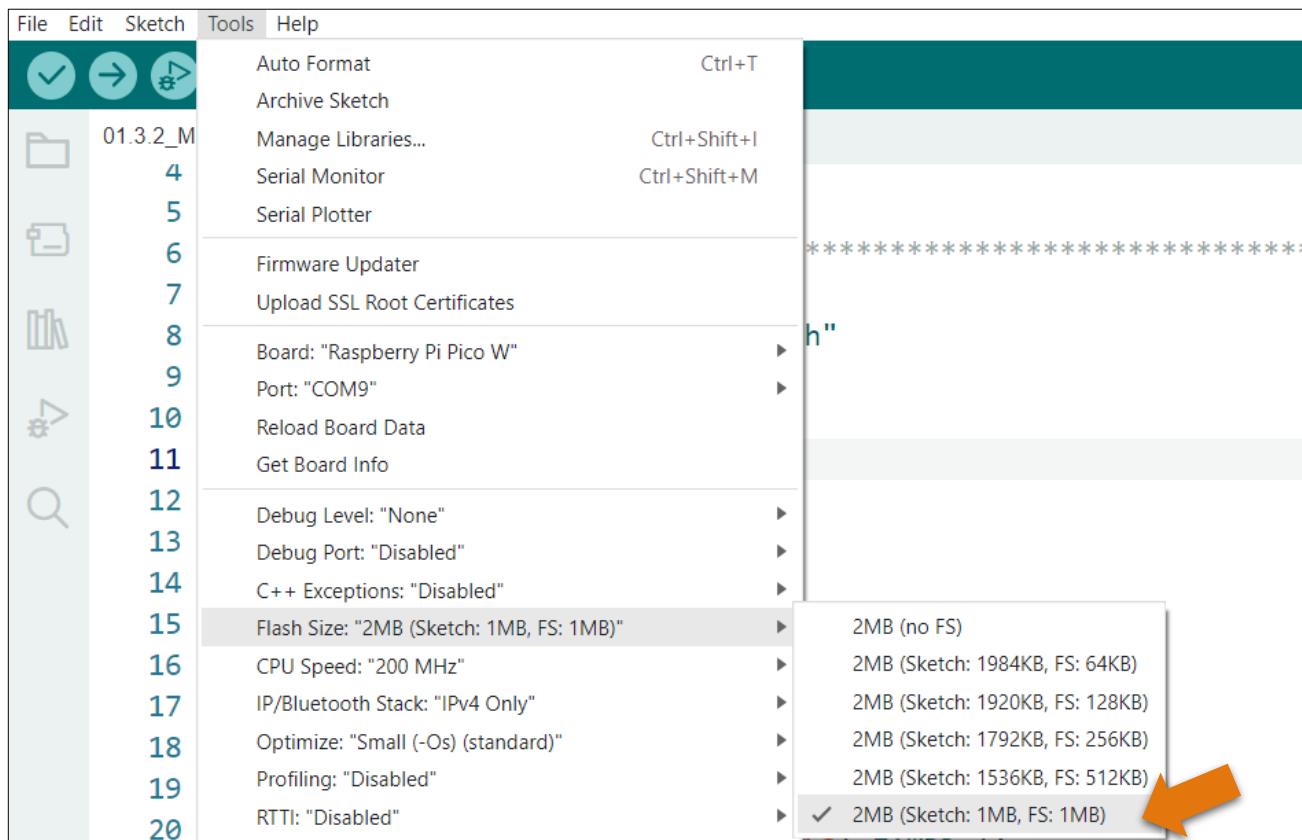
Finally, restart the Arduino IED. After restarting, you can see that the plug-in already exists in the interface.



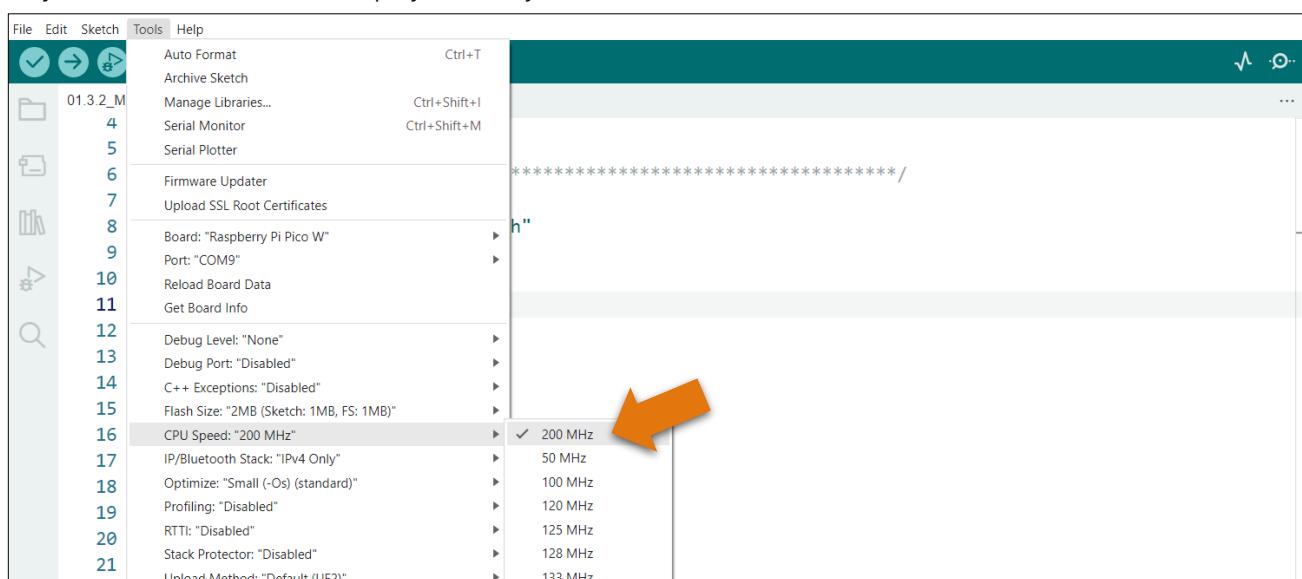
## Upload music

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

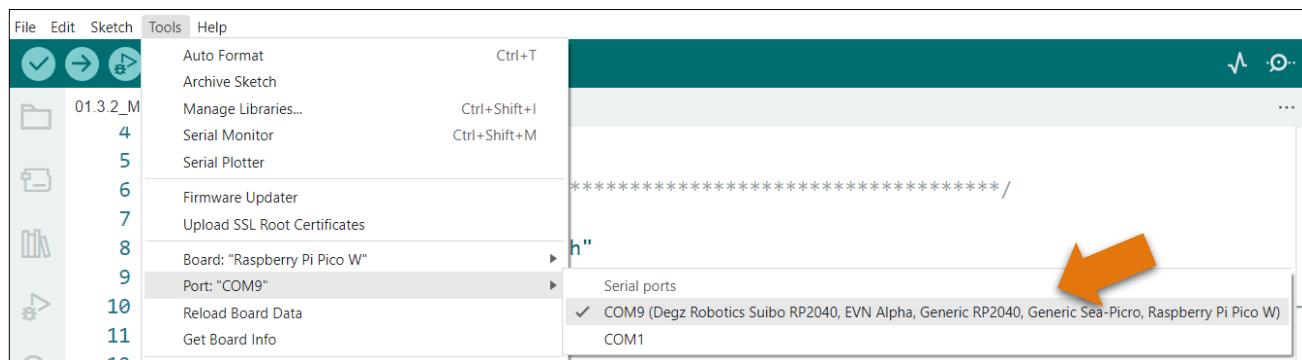
Open Arduino, select Tools from the menu bar, select Flash Size, and allocate 1MB of Flash space to store codes and 1MB to store audio files.



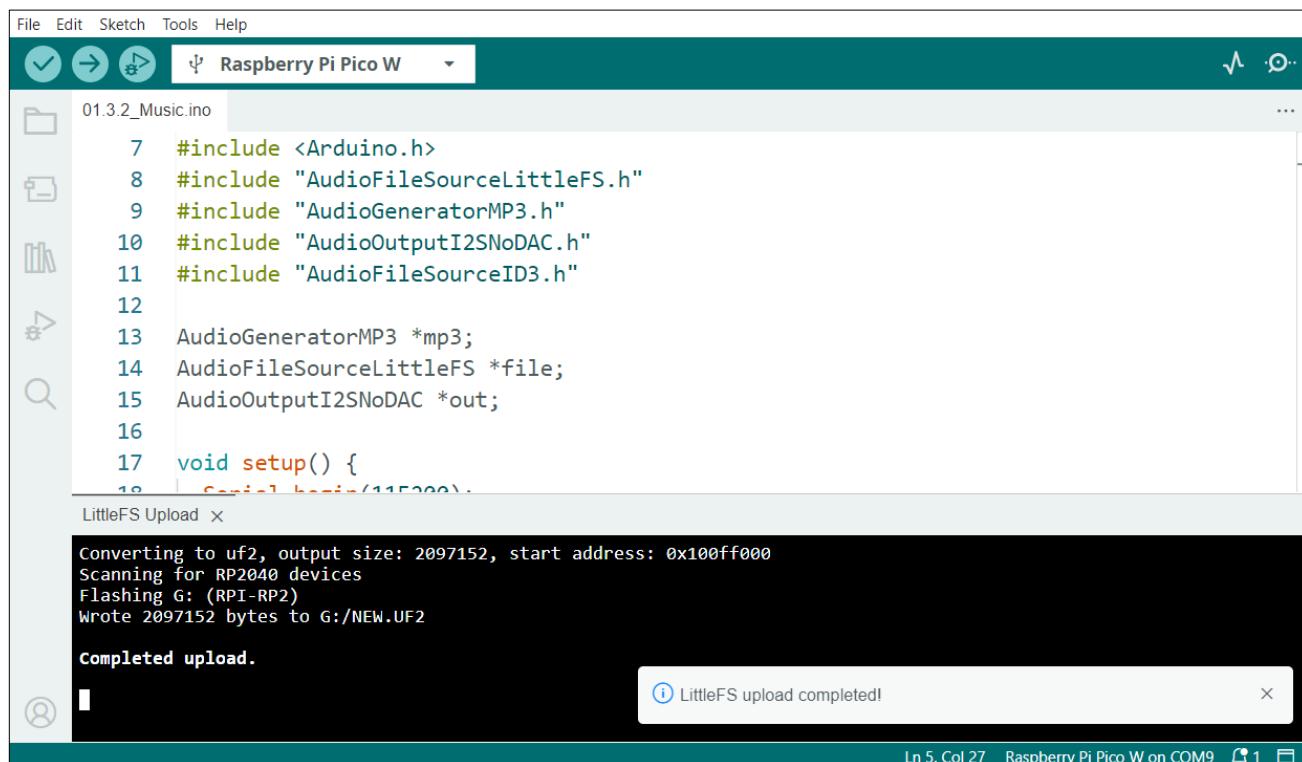
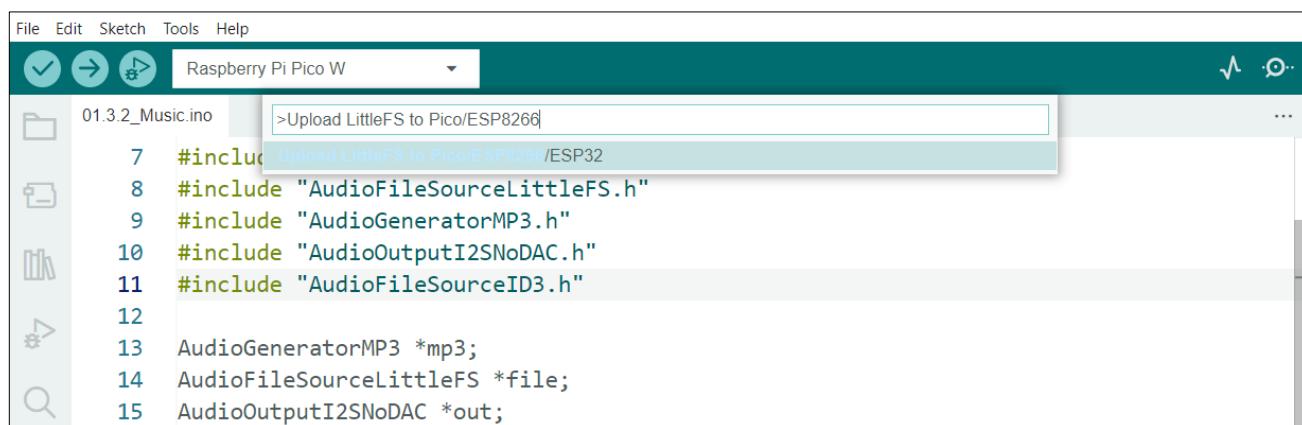
Ensure that the CPU speed is greater than 133MHz. If the frequency is too low, the audio decoding speed may be so slow that the audio playback may not be continuous.



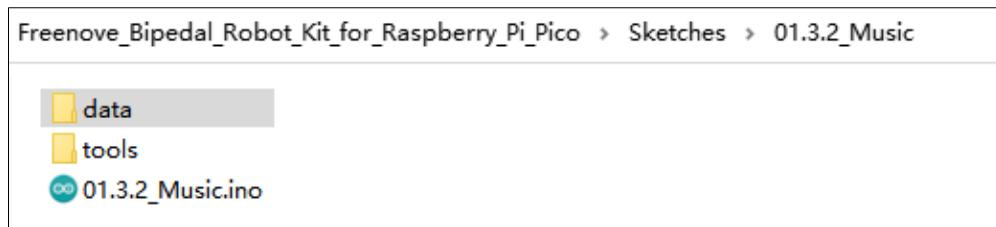
Select the correct port.



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.



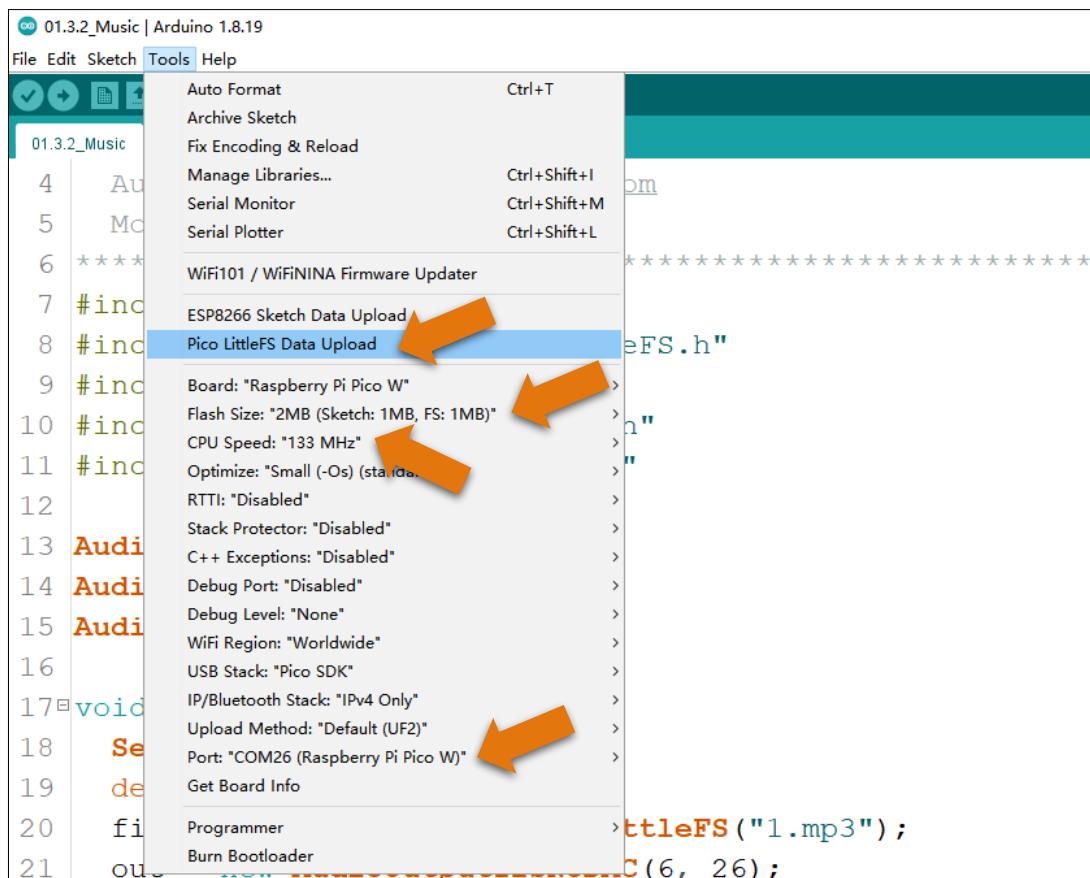
Check the code file and audio file. We create a folder named data under the same level directory of the code file, and place the audio file directly in this folder.



- 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:

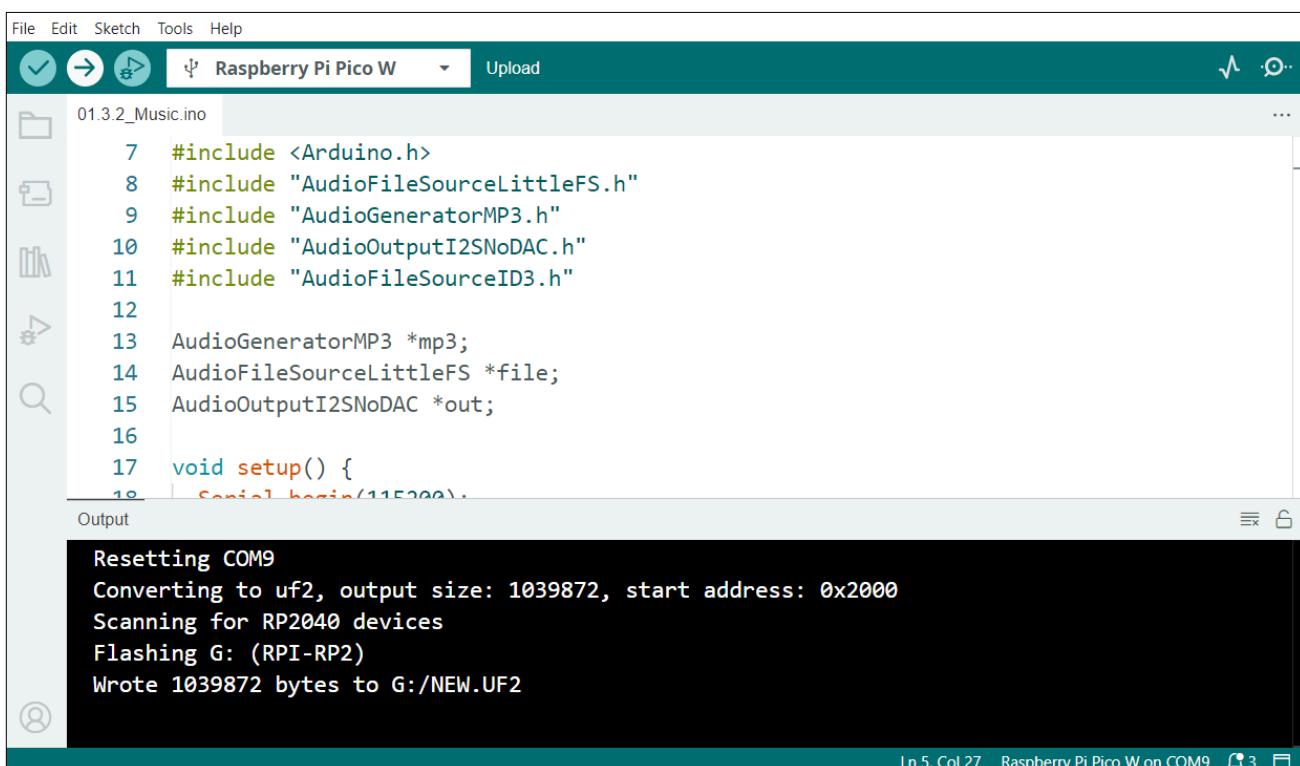


```
LittleFS Image Uploaded
[LittleFS] size      : 1024KB
[LittleFS] page      : 256
[LittleFS] block     : 4096
/l.mp3
/Hello.mp3
[LittleFS] upload    : C:\Users\Freenove\AppData\Local\Temp\arduino_build_527188/01.3.2_Music.mk
[LittleFS] address   : 269479936
[LittleFS] swerrial  : COM26
[LittleFS] python    : C:\Users\Freenove\AppData\Local\Arduino15\packages\rp2040\tools\pqt-python\2.0.0\python.exe
[LittleFS] uploader   : C:\Users\Freenove\AppData\Local\Arduino15\packages\rp2040\hardware\rp2040\0.1.3.2\cores\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
```

## Sketch

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>
2 #include "AudioFileSourceLittleFS.h"
3 #include "AudioGeneratorMP3.h"
4 #include "AudioOutputI2SNoDAC.h"
5 #include "AudioFileSourceID3.h"
```

```

6
7     AudioGeneratorMP3 *mp3;
8     AudioFileSourceLittleFS *file;
9     AudioOutputI2SNoDAC *out;
10
11    void setup() {
12        Serial.begin(115200);
13        delay(1000);
14        file = new AudioFileSourceLittleFS("1.mp3");
15        out = new AudioOutputI2SNoDAC(6);
16        out->SetGain(3); //Volume Setup
17        mp3 = new AudioGeneratorMP3();
18        mp3->begin(file, out);
19    }
20    int a = 0;
21
22    void loop() {
23        Serial.printf("loop1.. \n");
24        if (mp3->isRunning()) {
25            if (!mp3->loop()) {
26                mp3->stop();
27                Serial.printf("Hello done\n");
28                delete file;
29                delete mp3;
30                mp3 = new AudioGeneratorMP3();
31
32            }
33        } else {
34            Serial.printf("MP3 done\n");
35            pinMode(6, OUTPUT);
36            digitalWrite(6, LOW);
37        }
38    }

```

### Code Explanation

Set the audio file to be played.

14	file = new AudioFileSourceLittleFS("1.mp3");
----	--

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

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

Check whether the audio is being played.

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

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

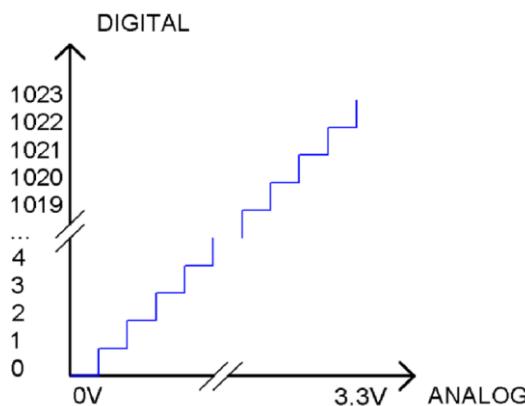
25	if (!mp3->loop()) {
26	mp3->stop();

```
27     Serial.printf("Hello done\n");
28     delete file;
29     delete mp3;
30     mp3 = new AudioGeneratorMP3();
31
32 }
```

## 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, which means the resolution is  $2^{10}=1024$ , and it represents a range (at 3.3V) will be divided equally to 1024 parts. The range 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 range of 0V---3.3/1023 V corresponds to digital 0;

Subsection 2: the analog in range of 3.3/1023 V---2\*3.3 /1023V corresponds to digital 1;

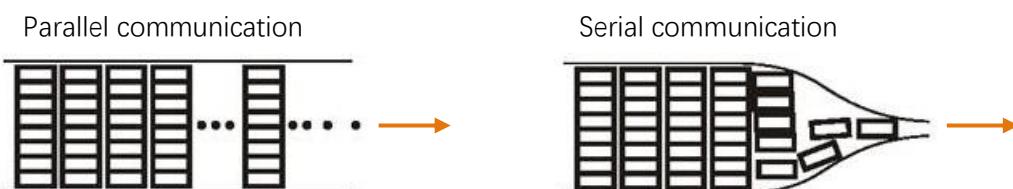
The following analog will be divided accordingly.

The conversion formula is as follows:

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

### Serial Communication

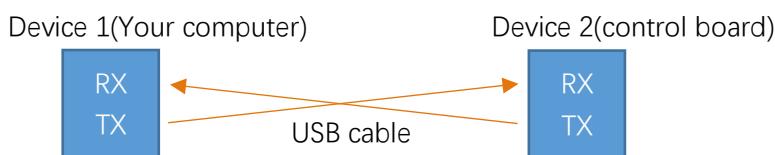
Serial communication uses one data cable to transfer data one bit by another in turn. Parallel communication means that the data is transmitted simultaneously on multiple cables. Serial communication takes only a few cables to exchange information between systems, which is especially suitable for computers 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.



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

[Need support? ✉ support.freenove.com](mailto:support.freenove.com)

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

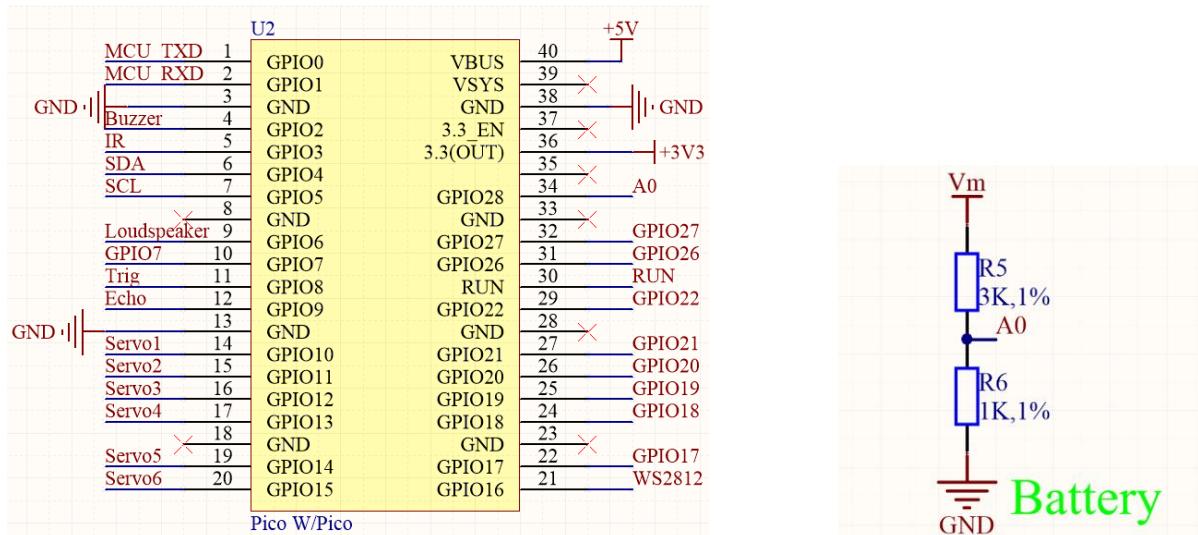


For serial communication, the **baud rate in both sides must be the same**. The baud rates commonly used are 9600 and 115200.

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 circuit, 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”.

Freenove_Bipedal_Robot_Kit_for_Raspberry_Pi_Pico-main > Sketches			
Name	Date modified	Type	Size
01.1.4_Servo	11/14/2024 10:51 AM	File folder	
01.2_Buzzer	11/14/2024 10:51 AM	File folder	
01.3.1_Loudspeaker	11/14/2024 10:51 AM	File folder	
01.3.2_Music	11/14/2024 10:51 AM	File folder	
01.4_Battery_level	11/14/2024 10:51 AM	File folder	
01.5_Matrix	11/14/2024 10:51 AM	File folder	
01.6_WS2812	11/14/2024 10:51 AM	File folder	

## Code

```

1 #include "Freenove_Robot_For_Pico_W.h"
2
3 void setup() {
4     Serial.begin(115200);           //Set the Serial Baud rate
5 }
6
7 void loop() {
8     Serial.print("Battery ADC : ");
9     Serial.println(Get_Battery_Voltage_ADC()); //Gets the battery ADC value
10    Serial.print("Battery Voltage : ");
11    Serial.print(Get_Battery_Voltage());        //Get the battery voltage value
12    Serial.println("V");
13    delay(300);
14 }
```

## Code Explanation

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

4	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 batteries and return it.

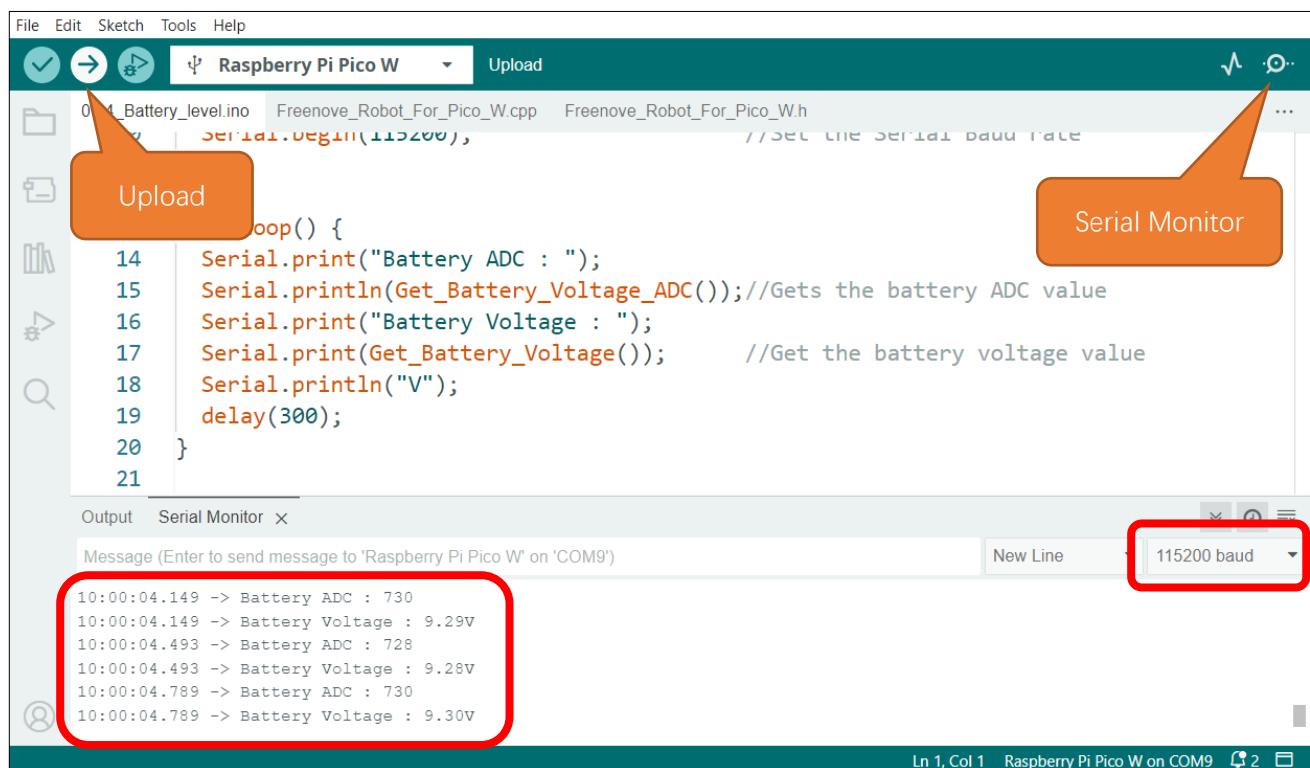
float Get_Battery_Voltage(void); //Get the battery voltage value
--

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

Need support? ✉ [support.freenove.com](mailto:support.freenove.com)

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

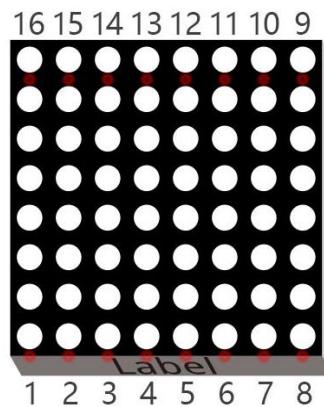
Click "Upload" to upload the code to Pico (W). After uploading successfully, click Serial Monitor, and set baud rate to 115200.



## 1.5 LED Matrix

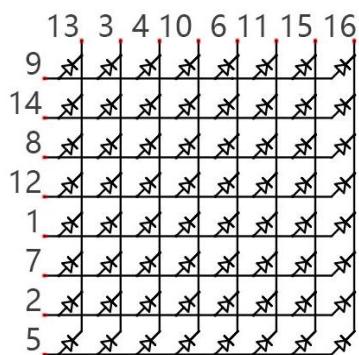
### LED Matrix

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).

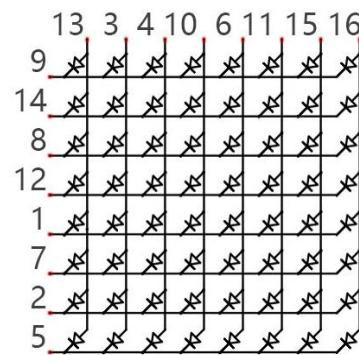


In order to facilitate the operation and reduce the number of ports required to drive this component, the positive poles of the LEDs in each row and negative poles of the LEDs in each column are respectively connected together inside the LED matrix module, which is called a common anode. There is another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

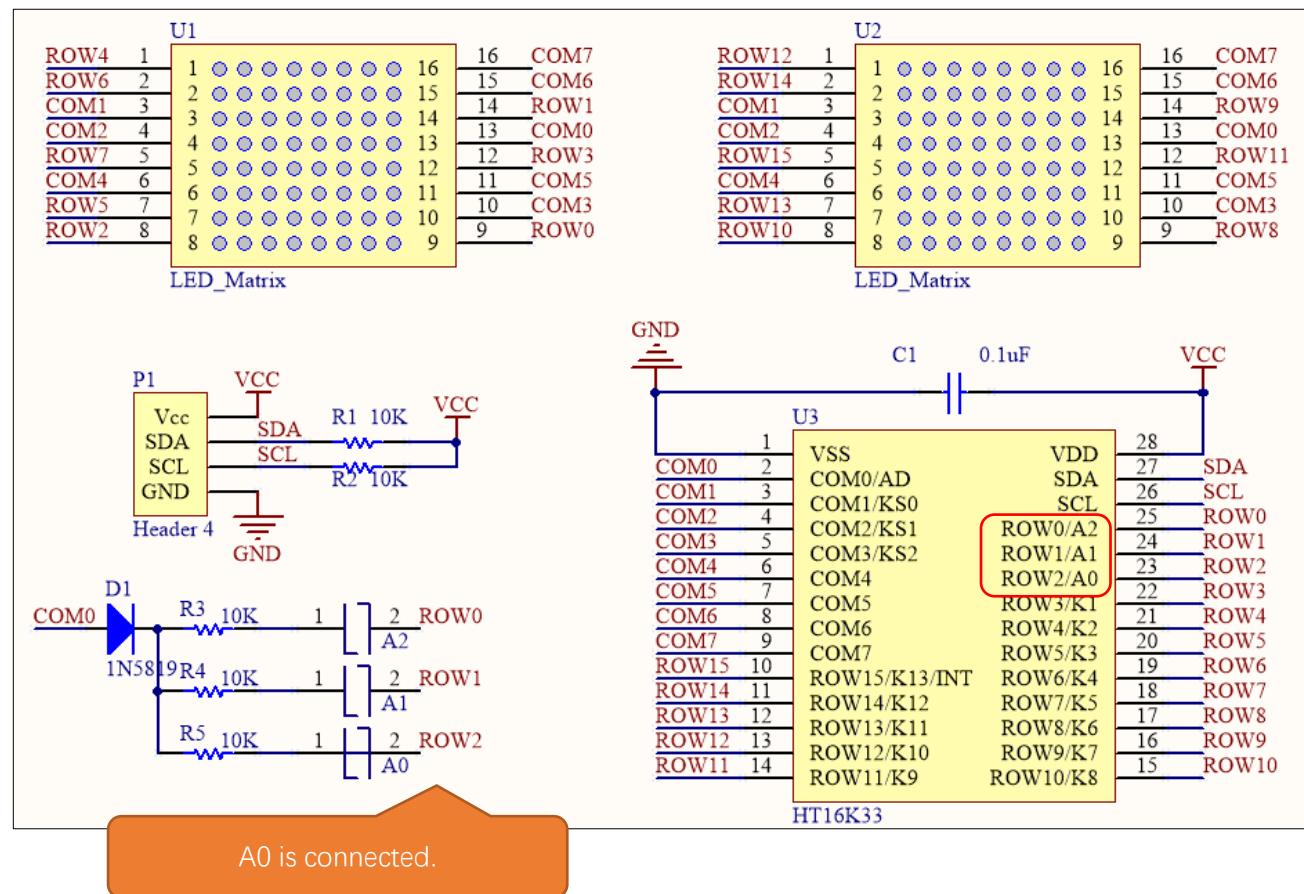


Connection mode of common cathode

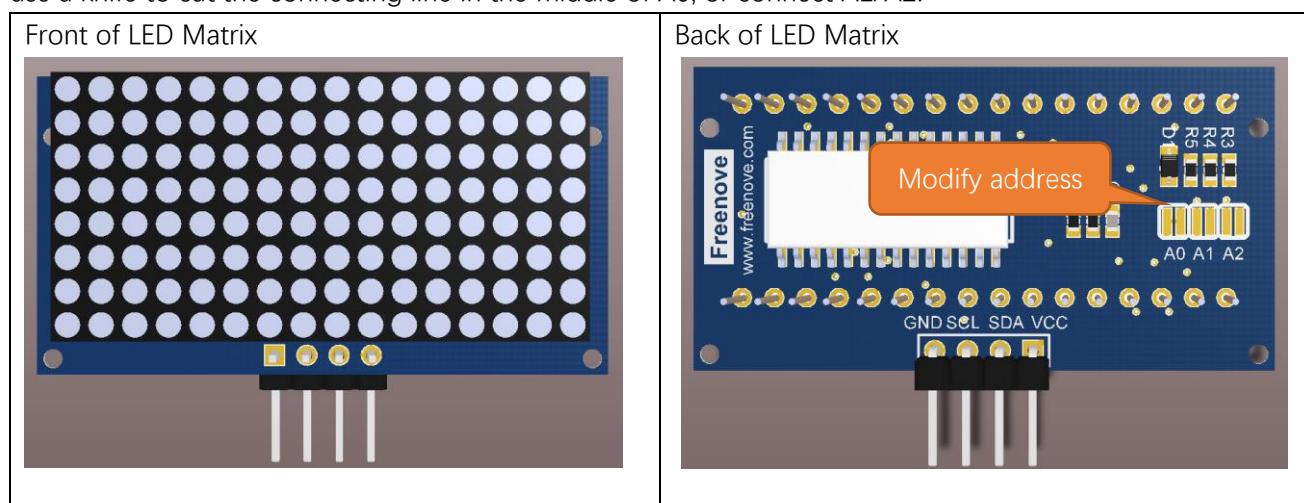


## Schematic

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.



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 represent the OFF LED.

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

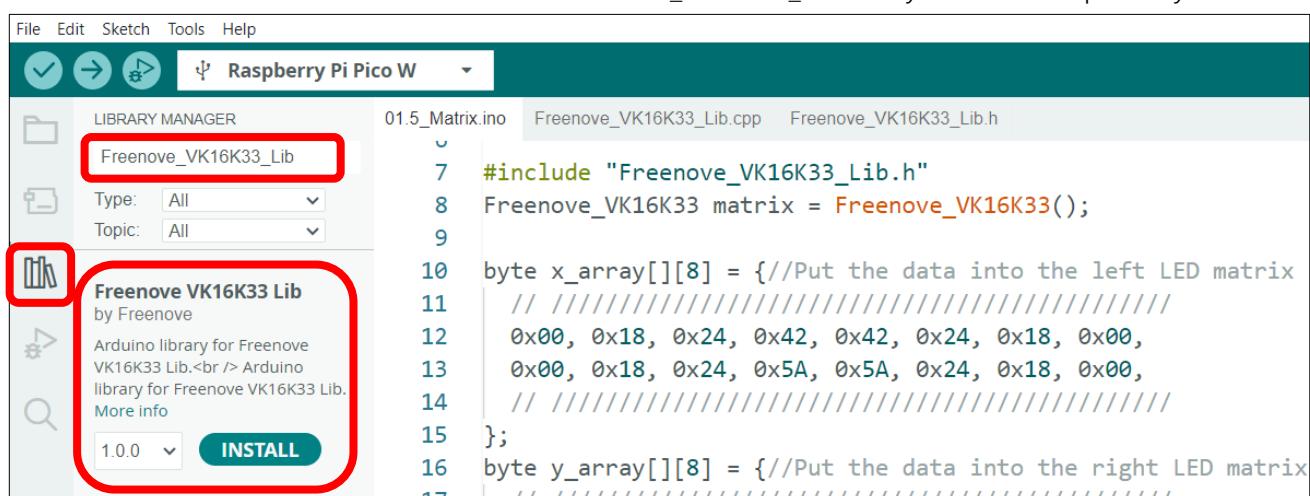
## Install Freenove\_VK16K33\_Lib Library

In this project, we use a third-party library named **Freenove\_VK16K33\_Lib** and **Adafruit-GFX-Library**.

Please install it first.

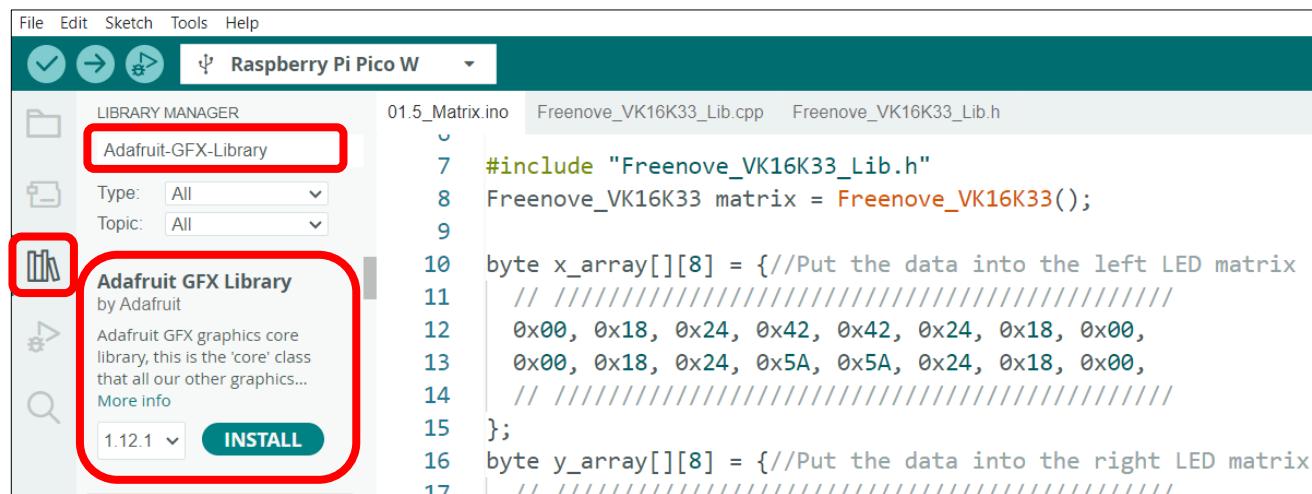
Open Arduino IDE, click **Library Manage** on the left, and search “**Freenove\_VK16K33\_Lib**” and “**Adafruit-GFX-Library**” to install.

It is recommended to use version **1.0.0** of the Freenove VK16K33 Lib library to avoid compatibility issues.



Need support?  support.freenove.com

It is recommended to use version **1.12.1** of the Adafruit-GFX-Library to avoid compatibility issues.



The second way, open Arduino IDE, click Sketch → Include Library → Add .ZIP Library. In the pop-up window, find the file named “./Libraries/ **Freenove\_VK16K33\_Lib-V1.0.0.Zip**” and “./Libraries/ **Adafruit-GFX-Library-V1.12.1.Zip**” which locates in this directory, and click OPEN.

The screenshot shows the Arduino IDE Sketch menu with the 'Include Library' option highlighted. An orange arrow points from the 'Sketch' menu to the 'Include Library' option. Another orange arrow points from the 'Include Library' option to the 'Add .ZIP Library...' item in the submenu. Below the menu, a file browser window shows the contents of the 'Libraries' folder. Several ZIP files are listed, with three specific ones highlighted with red boxes: 'Adafruit-GFX\_V1.12.1.zip', 'Freenove\_VK16K33\_Lib-V1.0.0.zip', and 'IRremote\_V4.4.1\_20241029.zip'. The 'Adafruit\_NeoPixel-V1.12.5.zip' file is also visible.

Name	Date modified	Size
Adafruit_NeoPixel-V1.12.5.zip	9/6/2025 9:02 AM	87 KB
<b>Adafruit-GFX_V1.12.1.zip</b>	9/6/2025 8:58 AM	349 KB
ESP8266Audio-V1.9.7.zip	9/6/2025 9:02 AM	7,648 KB
<b>Freenove_VK16K33_Lib-V1.0.0.zip</b>	9/6/2025 9:02 AM	14 KB
IRremote_V4.4.1_20241029.zip	9/6/2025 9:02 AM	1,350 KB

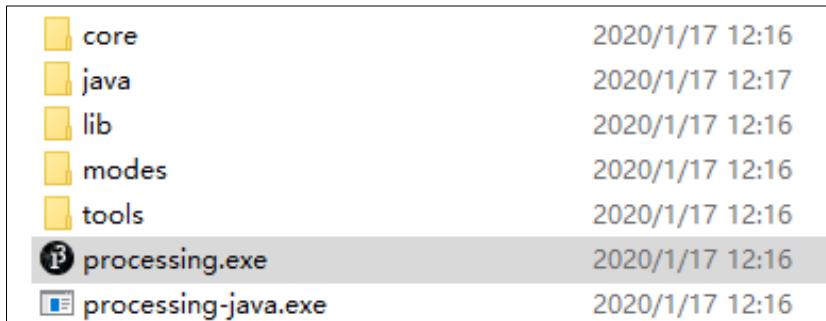
## Install Processing

In this tutorial, we use Processing to build a simple LED Matrix platform. If you prefer to use the default expression, you can skip to the [next step](#).

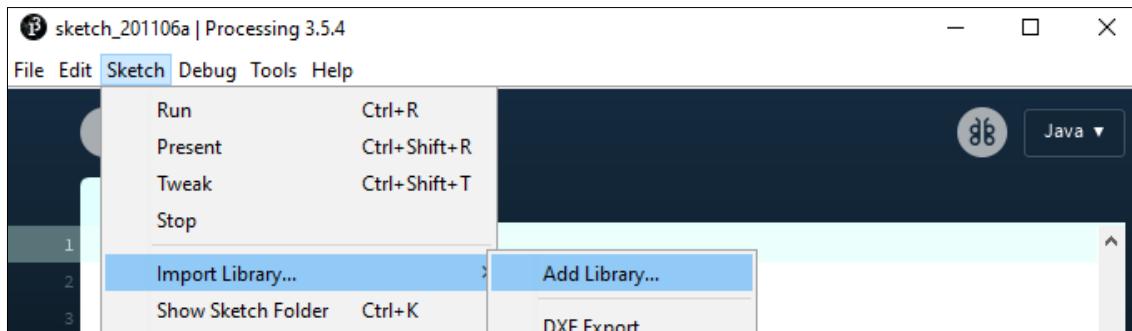
If you have not installed Processing, you can download it from the official website: <https://processing.org/>. Download an appropriate version to download corresponding to your PC system.

The screenshot shows the official Processing Foundation website. At the top, there's a navigation bar with links for "Processing", "p5.js", "Processing.py", "Processing for Android", "Processing for Pi", and "Processing Foundation". Below the navigation is a large banner with the word "Processing" and a complex geometric background. To the right of the banner is a search bar. On the left side of the main content area, there's a sidebar with links for "Cover", "Download", "Donate", "Exhibition", "Reference", "Libraries", "Tools", "Environment", "Tutorials", and "Examples". Below the sidebar is a link to "» Github". The main content area features a large "P" logo with a geometric pattern. To the right of the logo, it says "3.5.4 (17 January 2020)". Below that are links for "Windows 64-bit", "Windows 32-bit", "Linux 64-bit", and "Mac OS X". At the bottom of the main content area, there's a link to "Read about the changes in 3.0. The list of revisions covers the differences".

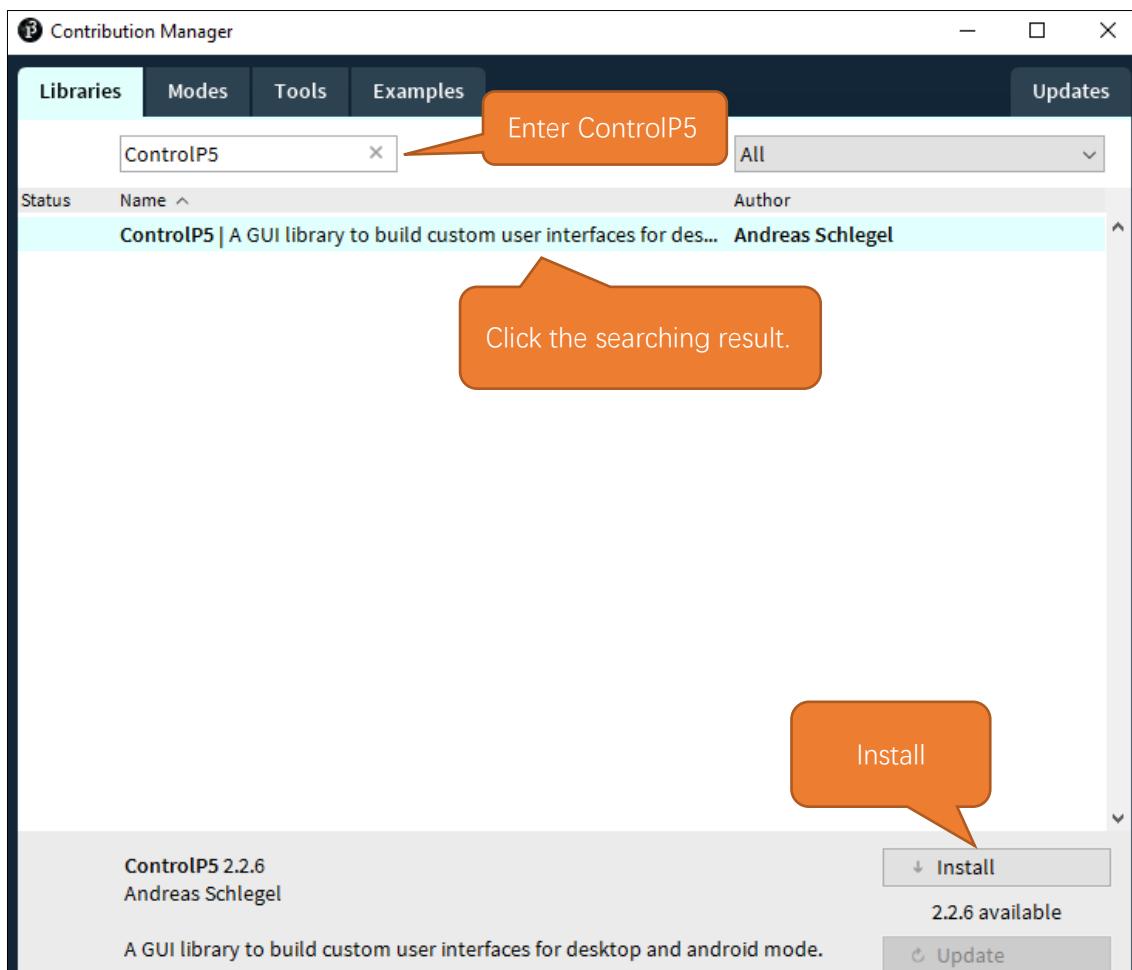
Unzip the downloaded file to your computer. Click "processing.exe" as the figure below to run this software.



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”.

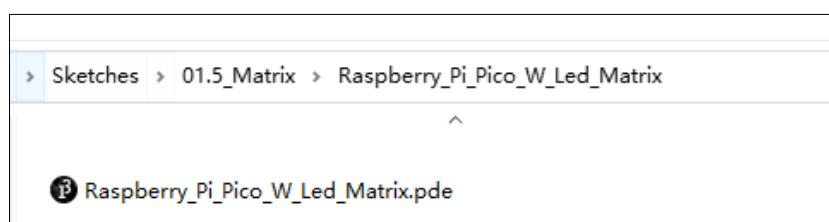


When the installation finishes, restart Processing.

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.



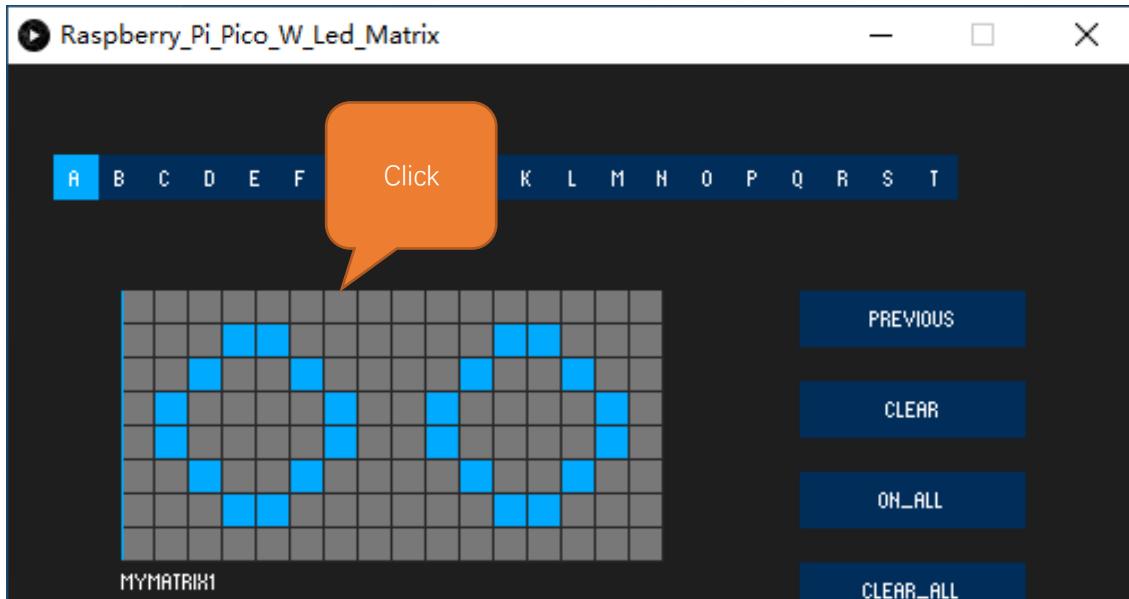
Click “Run”



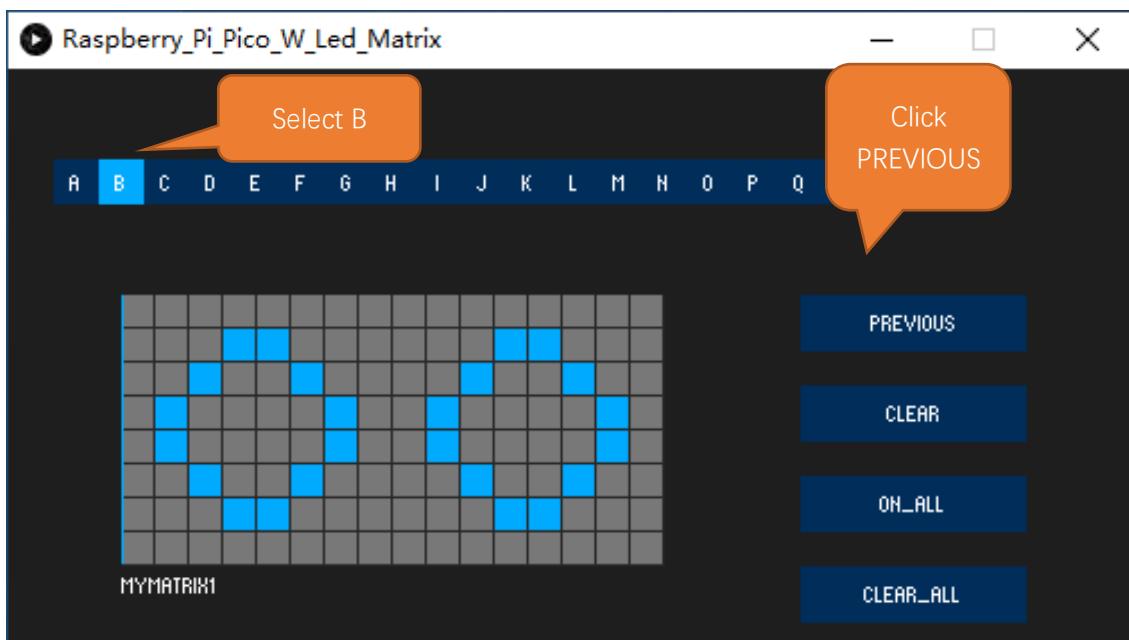
There are 20 pages from A to T. Select Page A.



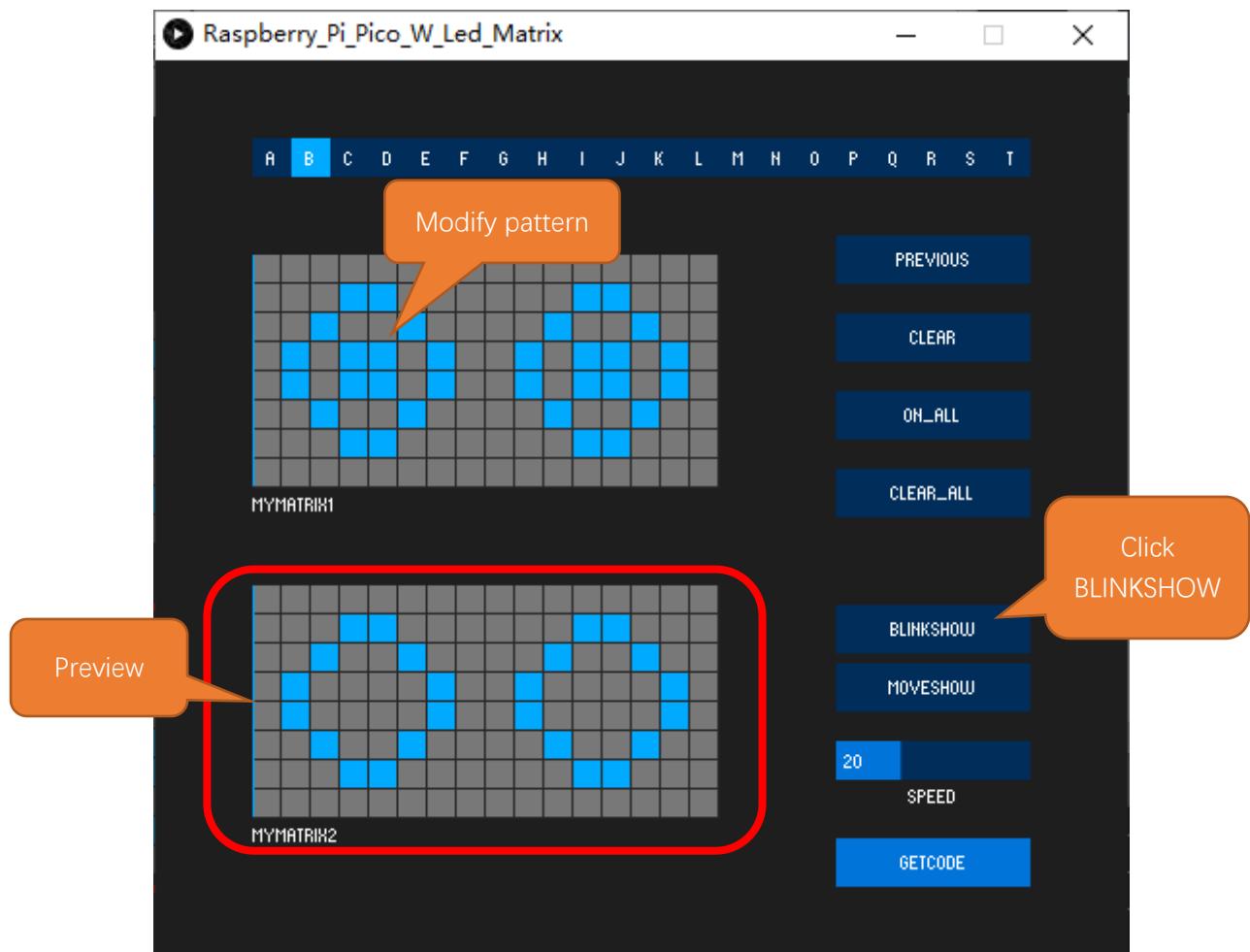
You can design your own pattern by clicking on the squares.



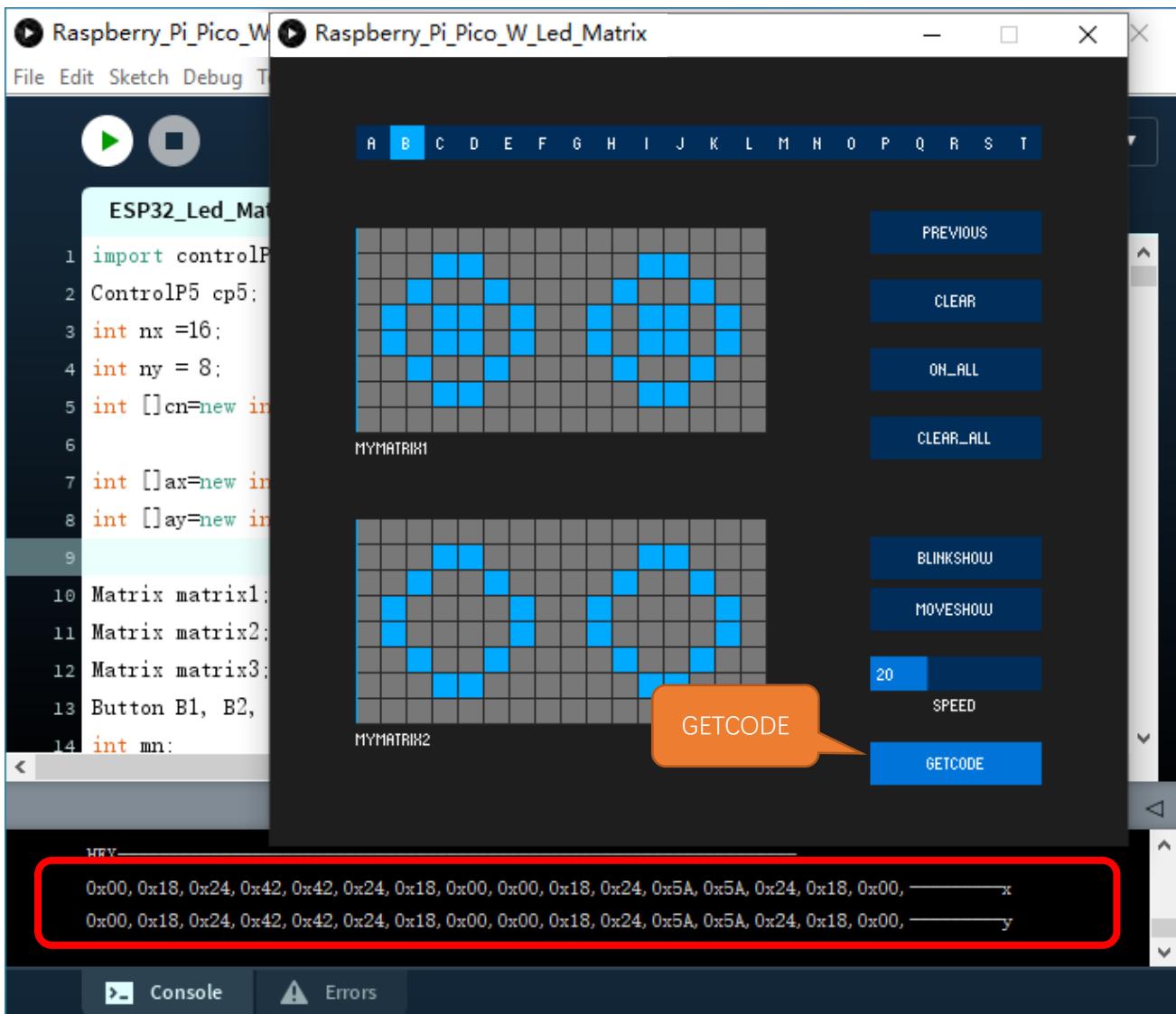
Next select Page B and click PREVIOUS, which copies the previous pattern to B.



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



Chick 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".

## Sketch

In this chapter, we will drive the LED matrix module via the I<sup>2</sup>C bus of the Raspberry Pi Pico (W).

Open the folder “01.5\_Matrix” in the “Freenove\_Bipedal\_Robot\_Kit\_for\_Raspberry\_Pi\_Pico\Sketches” and double click “01.5\_Matrix.ino”

Freenove_Bipedal_Robot_Kit_for_Raspberry_Pi_Pico-main > Sketches			
Name	Date modified	Type	Size
01.1.4_Servo	11/14/2024 10:51 AM	File folder	
01.2_Buzzer	11/14/2024 10:51 AM	File folder	
01.3.1_Loudspeaker	11/14/2024 10:51 AM	File folder	
01.3.2_Music	11/14/2024 10:51 AM	File folder	
01.4_Battery_level	11/14/2024 10:51 AM	File folder	
01.5_Matrix	11/14/2024 10:51 AM	File folder	
01.6_WS2812	11/14/2024 10:51 AM	File folder	

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

```

File Edit Sketch Tools Help
Raspberry Pi Pico W Upload
01.5_Matrix.ino Freenove_VK16K33_Lib.cpp Freenove_VK16K33_Lib.h ...
01.5_Matrix.ino
#include "Freenove_VK16K33_Lib.h"
8 Freenove_VK16K33 matrix = Freenove_VK16K33();
9
10 byte x_array[][][8] = {//Put the data into the left LED matrix
11 // /////////////////////////////////
12 0x00, 0x18, 0x24, 0x42, 0x42, 0x24, 0x18, 0x00,
13 0x00, 0x18, 0x24, 0x5A, 0x5A, 0x24, 0x18, 0x00,
14 // /////////////////////////////////
15 };
16 byte y_array[][][8] = {//Put the data into the right LED matrix
17 // /////////////////////////////////
18 0x00, 0x18, 0x24, 0x42, 0x42, 0x24, 0x18, 0x00,
19 0x00, 0x18, 0x24, 0x5A, 0x5A, 0x24, 0x18, 0x00,
20 // /////////////////////////////////
21 };
22
23 void setup()
24 {
25   matrix.init(0x71);
26   matrix.FlipVertical(); //Flips the vertical orientation of the matrices
  
```

Replace the x array on the left

Replace the y array on the right

## Code

```

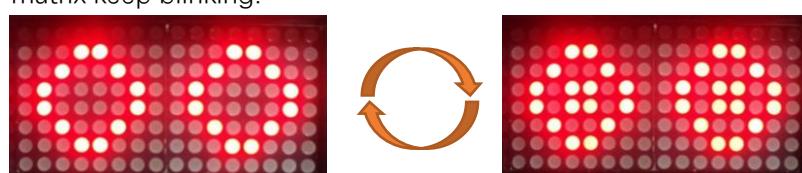
1 #include "Freenove_VK16K33_Lib.h"
2 Freenove_VK16K33 matrix = Freenove_VK16K33();
3
4 byte x_array[][][8] = {//Put the data into the left LED matrix
5 // /////////////////////////////////
  
```

```

6   0x00, 0x18, 0x24, 0x42, 0x42, 0x24, 0x18, 0x00,
7   0x00, 0x18, 0x24, 0x5A, 0x5A, 0x24, 0x18, 0x00,
8   // /////////////////////////////////
9 };
10 byte y_array[][] = { //Put the data into the right LED matrix
11   // ///////////////////////////////
12   0x00, 0x18, 0x24, 0x42, 0x42, 0x24, 0x18, 0x00,
13   0x00, 0x18, 0x24, 0x5A, 0x5A, 0x24, 0x18, 0x00,
14   // ///////////////////////////////
15 };
16
17 void setup()
18 {
19   matrix.init(0x71);
20   matrix.flipVertical(); //Flips the vertical orientation of the matrices.
21   matrix.flipHorizontal(); //Flips the horizontal orientation of the matrices.
22   matrix.setBrightness(5);
23   matrix.setBlink(VK16K33_BLINK_OFF);
24 }
25
26 void loop()
27 {
28   showArray(500);
29 }
30
31 void showArray(int delay_ms)
32 {
33   int count = sizeof(x_array) / sizeof(x_array[0]);
34   for (int i = 0; i < count; i++)
35   {
36     matrix.showStaticArray(x_array[i], y_array[i]);
37     delay(delay_ms);
38   }
39 }

```

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 a Freenove\_VK16K33 object and name it matrix.

```
2 Freenove_VK16K33 matrix = Freenove_VK16K33();
```

Define the I<sup>2</sup>C address and I<sup>2</sup>C 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 blink.

```
2 Freenove_VK16K33 matrix = Freenove_VK16K33();  
...  
19     ...  
20     matrix.init(0x71);  
21     matrix.flipVertical(); //Flips the vertical orientation of the matrices.  
22     matrix.flipHorizontal(); //Flips the horizontal orientation of the matrices.  
23     matrix.setBrightness(5);  
      matrix.setBlink(VK16K33_BLINK_OFF);
```

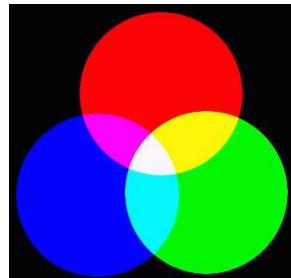
Define count to calculate the number of one-dimensional arrays contained in the two-dimensional x\_array, and use the “for” loop to call the showStaticArray() function to continuously display the content of LED matrix.

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

## 1.6 LED

### LED

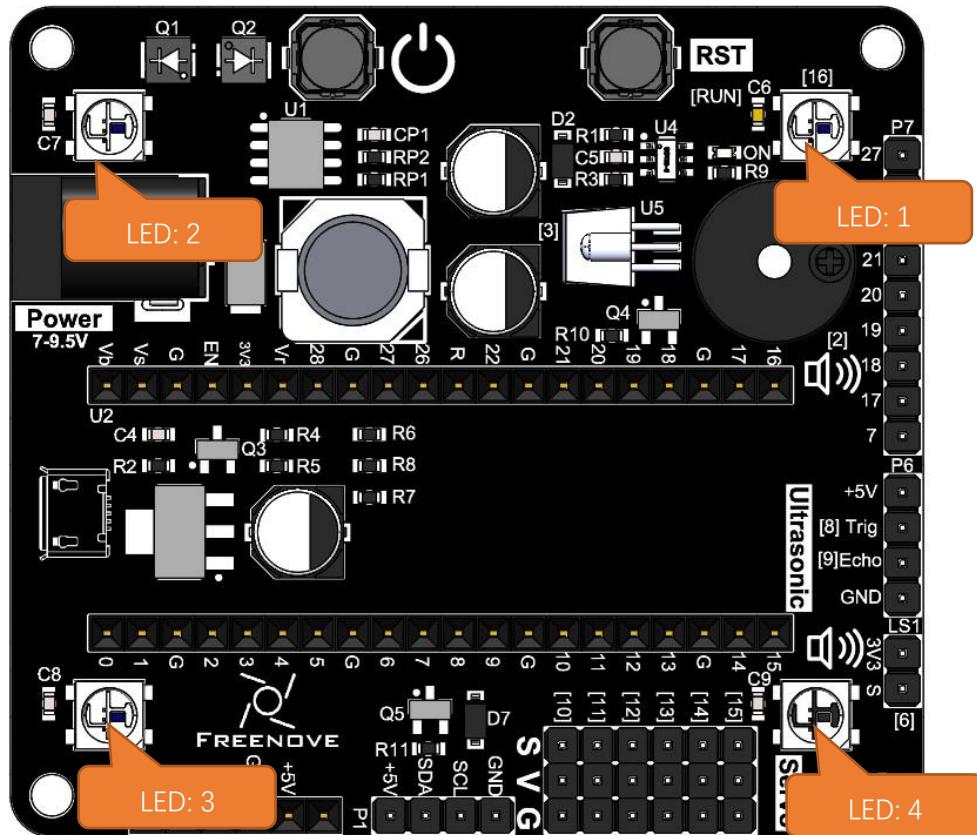
Red, green, and blue are called the three primary colors. When you combine these three primary colors of different brightness, it can produce almost all kinds of visible light.



RGB

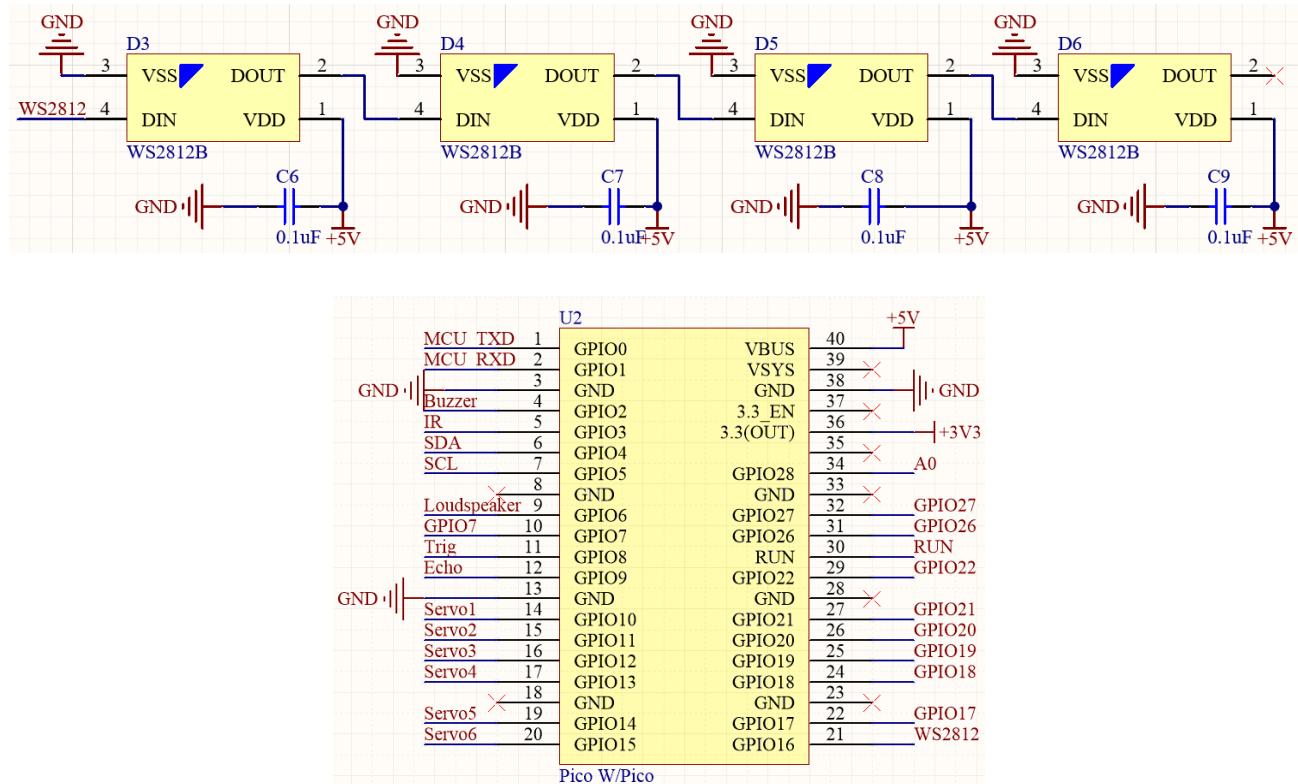
The main control board of the robot integrates 4 LEDs, each with an independent control pin and support for cascading expansion.

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. Control with the GPIO16 of the Raspberry Pi Pico (W).

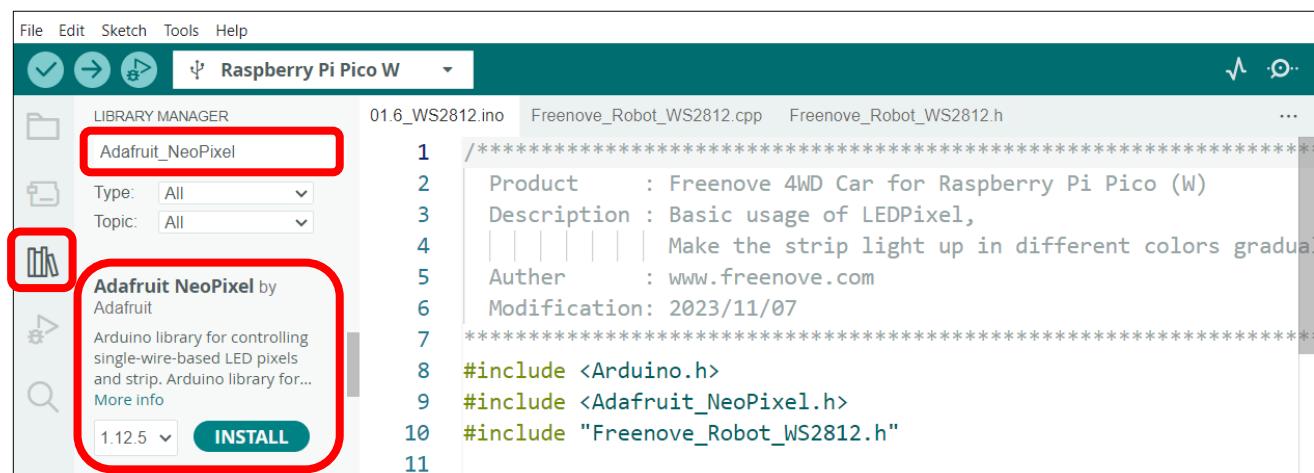


## Install Adafruit\_NeoPixel library

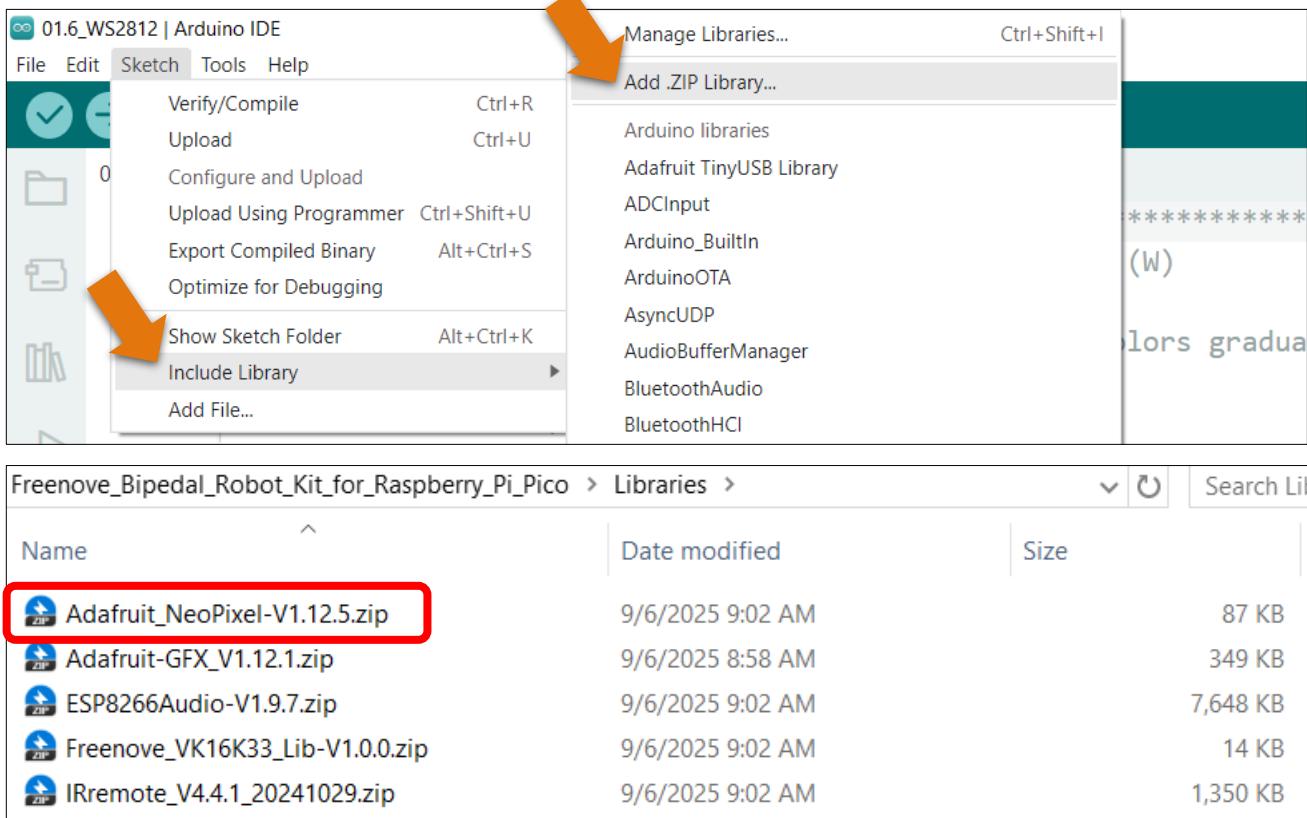
In this project, we use a third-party library named **Adafruit\_NeoPixel**. Please install it first.

Open Arduino IDE, click **Library Manager** on the left, and search “**Adafruit\_NeoPixel**” to install.

It is recommended to use version **1.12.5** of the Adafruit\_NeoPixel library to avoid compatibility issues.



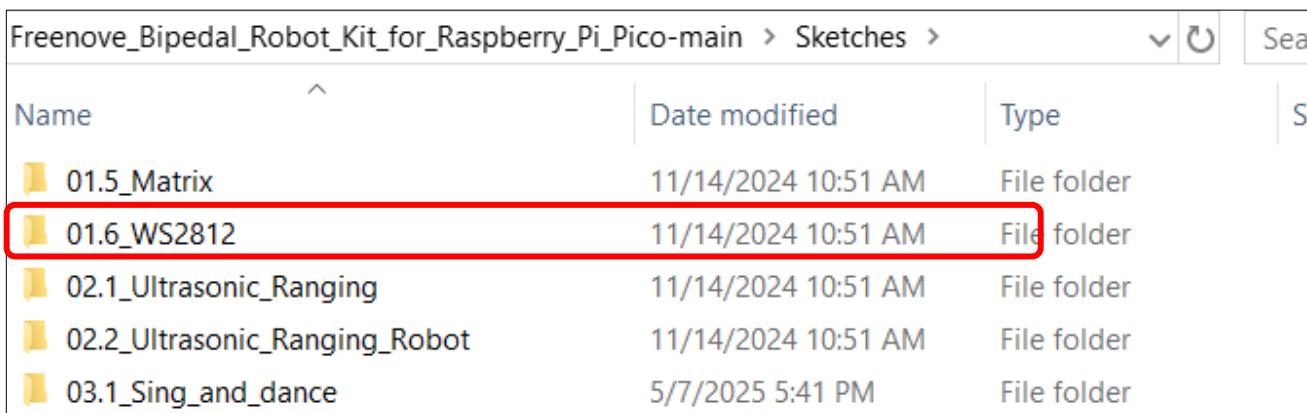
The second way, open Arduino IDE, click Sketch → Include Library → Add .ZIP Library. In the pop-up window, find the file named “./Libraries/ **Adafruit\_NeoPixel-V1.12.5.Zip**” which locates in this directory, and click OPEN.



## Sketch

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

```
1 #include <Arduino.h>
2 #include <Adafruit_NeoPixel.h>
3 #include "Freenove_Robot_WS2812.h"
4
5 void setup() {
6     WS2812_Setup();      //WS2812 initialization
7 }
8
9 void loop() {
10    WS2812_Show(5);
11 }
```

## Another part of the code

```
1 #define WS2812_PIN 16      //Control pins for Pico W
2 #define LEDS_COUNT 4
3 Adafruit_NeoPixel ws2812_strip(LEDS_COUNT, WS2812_PIN, NEO_GRB + NEO_KHZ800);
4 //WS2812 initialization function
5 void WS2812_Setup(void)
6 {
7     ws2812_strip.begin();
8     ws2812_strip.setBrightness(20);
9     ws2812_close();
10    WS2812_Set_Color_1(4095, 0, 0, 100);
11    WS2812_Set_Color_2(4095, 0, 0, 0);
12    ws2812_strip_time_now = millis();
13
14 }
15 //WS2812 non-blocking display function
16 void WS2812_Show(int mode)
17 {
18     switch (mode)
19     {
20         case 0://Close the WS2812
21             ws2812_close();
22             break;
23         case 1:
24             ws2812_rgb();
25             break;
26         case 2:
27             ws2812_following();
28             break;
29         case 3:
```

```

30     ws2812_blink();
31     break;
32     case 4:
33     ws2812_breathe();
34     break;
35     case 5:
36     ws2812_rainbow();
37     break;
38     default:
39     break;
40   }
41 }
42 int rainbow_count = 0;
43 //WS2812 rainbow display
44 void ws2812_rainbow(void)
45 {
46   ws2812_strip.setBrightness(20);
47   ws2812_strip_time_next = millis();
48   if (ws2812_strip_time_next - ws2812_strip_time_now > 5)
49   {
50     ws2812_strip_time_now = ws2812_strip_time_next;
51     rainbow_count++;
52     for (int i = 0; i < LEDS_COUNT; i++)
53       ws2812_strip.setPixelColor(LEDS_COUNT-1 - i, Wheel((i * 256 / LEDS_COUNT +
rainbow_count) & 255));
54     ws2812_strip.show();
55     if (rainbow_count > 255)
56       rainbow_count = 0;
57   }
58 }
```

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.

```

1 #define WS2812_PIN 16      //Control pins for Pico W
2 #define LEDS_COUNT 4
3 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.

```

7 ws2812_strip.begin();
8 ws2812_strip.setBrightness(20);
```

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

Need support? [✉ support.freenove.com](mailto:support.freenove.com)

```
53 ws2812_strip.setPixelColor(LEDS_COUNT-1 - i, Wheel((i * 256 / LEDS_COUNT + rainbow_count) &  
54 255));
```

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

```
| 10 WS2812_Show(5);
```

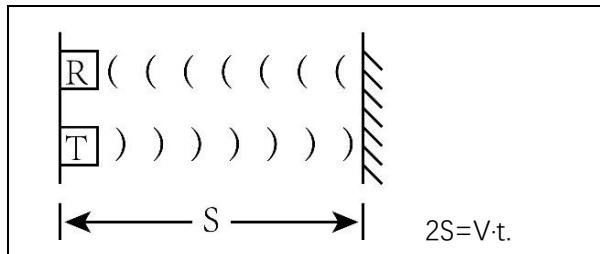
# Chapter 2 Ultrasonic Obstacle Avoidance Robot

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

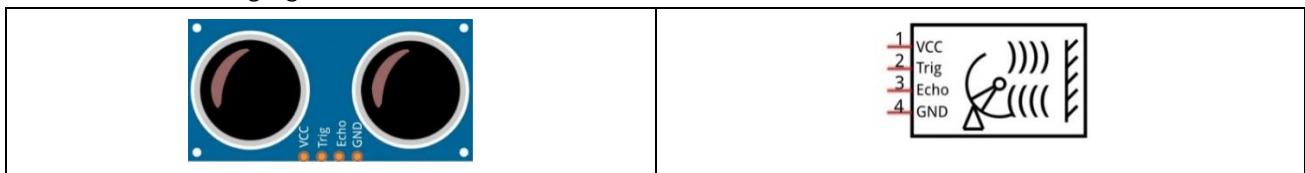
## 2.1 Ultrasonic Module

### 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  $v=340\text{m/s}$ , 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:

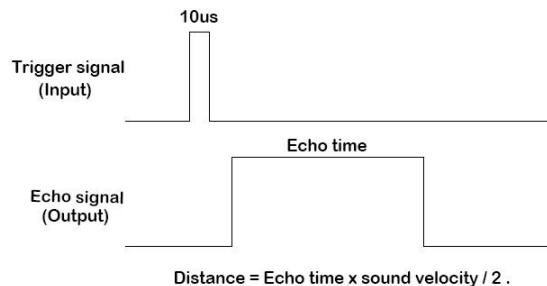
Working voltage: 5V

Working current: 12mA

Minimum measured distance: 2cm

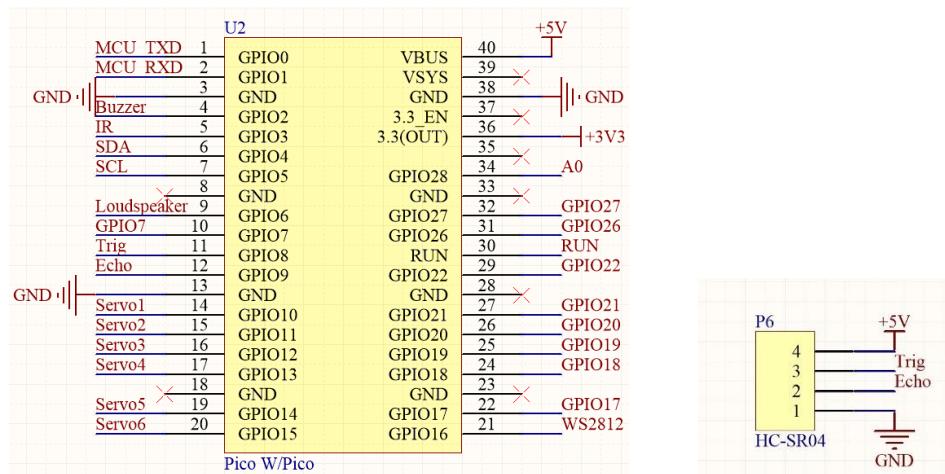
Maximum measured distance: 200cm

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"**  
double click "02.1\_Ultrasonic\_Ranging.ino".

Freenove_Bipedal_Robot_Kit_for_Raspberry_Pi_Pico-main > Sketches >		
Name	Date modified	Type
01.5_Matrix	11/14/2024 10:51 AM	File folder
01.6_WS2812	11/14/2024 10:51 AM	File folder
02.1_Ultrasonic_Ranging	11/14/2024 10:51 AM	File folder
02.2_Ultrasonic_Ranging_Robot	11/14/2024 10:51 AM	File folder

## Code

```

1 #include <Arduino.h>
2 #include "Freenove_Robot_For_Pico_W.h"
3
4 void setup() {
5     Serial.begin(115200); //Open the serial port and set the baud rate to 115200
6     Ultrasonic_Setup(); //Ultrasonic module initialization
7     delay(500); //Wait for the servo to arrive at the specified location
8 }
9
10 void loop() {
11     Serial.print("Distance: " + String(Get_Sonar()) + "\n"); //Print ultrasonic distance
12     delay(500);
13 }
```

## Code Explanation

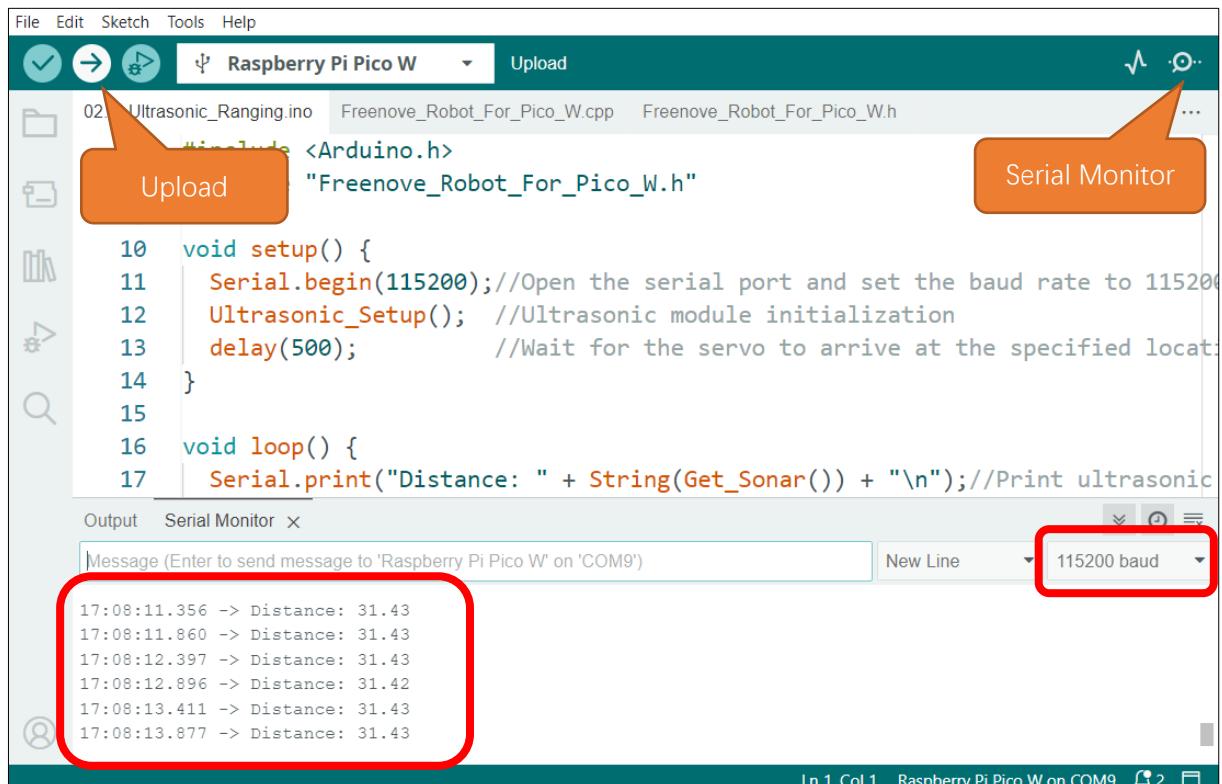
Initialize ultrasonic module.

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

Retrieve the distance between the ultrasonic module and obstacles, returning real-time data in centimeters, and print it in the serial port every 0.5 seconds.

```
11     Serial.print("Distance: " + String(Get_Sonar()) + "\n"); //Print ultrasonic distance
12     delay(500);
```

Click “Upload” to upload the code to Raspberry Pi Pico (W). After uploading successfully, open Serial Monitor and set the baud rate as 115200.



Need support? [✉ support.freenove.com](mailto:support.freenove.com)

## 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”.

Freenove_Bipedal_Robot_Kit_for_Raspberry_Pi_Pico-main > Sketches >		
Name	Date modified	Type
01.5_Matrix	11/14/2024 10:51 AM	File folder
01.6_WS2812	11/14/2024 10:51 AM	File folder
02.1_Ultrasonic_Ranging	11/14/2024 10:51 AM	File folder
02.2_Ultrasonic_Ranging_Robot	11/14/2024 10:51 AM	File folder
03.1_Sing_and_dance	5/7/2025 5:41 PM	File folder

### Code

```
1 #include <Arduino.h>
2 #include "Freenove_Robot_For_Pico_W.h"
3 #include "Bipodal_Robot.h"
4 Bipodal_Robot Bipodal_Robot;
5
6 #define LeftLeg 10
7 #define RightLeg 12
8 #define LeftFoot 11
9 #define RightFoot 13
10
11
12 void setup() {
13     Serial.begin(115200);
14     Ultrasonic_Setup();           //Initialize the ultrasonic module
15     Bipodal_Robot.init(LeftLeg, RightLeg, LeftFoot, RightFoot, true); //Set the servo pins
16     Bipodal_Robot.home();
17     delay(50);
18 }
19 void loop()
20 {
```

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

## 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.

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

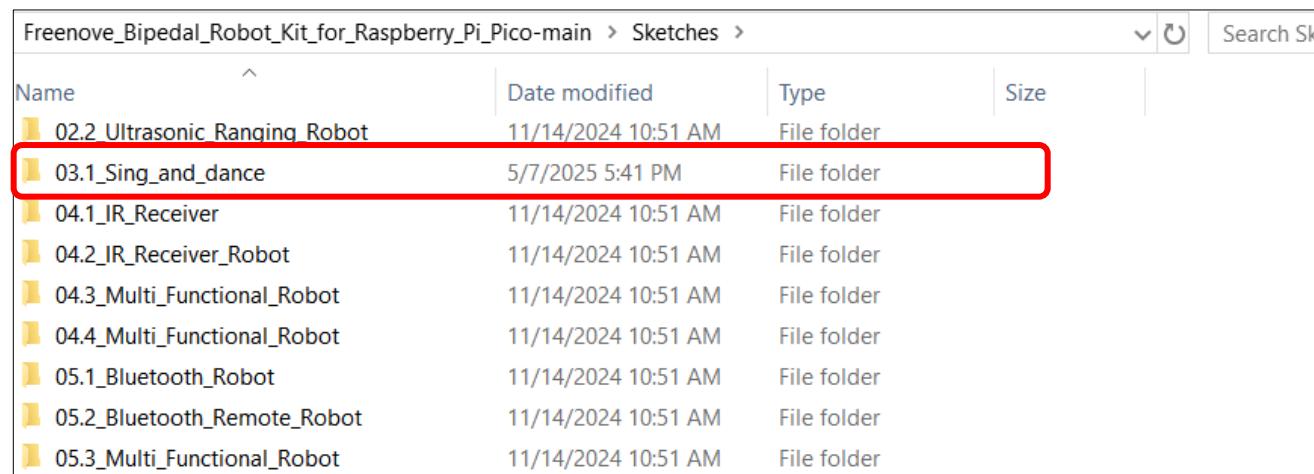
# Chapter 3 Sing and Dance

## 3.1 Sing and dance

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

Open the folder “03.1\_Sing\_and\_dance” in

“**Freenove\_Bipedal\_Robot\_Kit\_for\_Raspberry\_Pi\_Pico\Sketches**” and double click “03.1\_Sing\_and\_dance.ino”.



Freenove_Bipedal_Robot_Kit_for_Raspberry_Pi_Pico-main > Sketches >			
Name	Date modified	Type	Size
02.2_Ultrasonic_Ranging_Robot	11/14/2024 10:51 AM	File folder	
03.1_Sing_and_dance	5/7/2025 5:41 PM	File folder	
04.1_IR_Receiver	11/14/2024 10:51 AM	File folder	
04.2_IR_Receiver_Robot	11/14/2024 10:51 AM	File folder	
04.3_Multi_Functional_Robot	11/14/2024 10:51 AM	File folder	
04.4_Multi_Functional_Robot	11/14/2024 10:51 AM	File folder	
05.1_Bluetooth_Robot	11/14/2024 10:51 AM	File folder	
05.2_Bluetooth_Remote_Robot	11/14/2024 10:51 AM	File folder	
05.3_Multi_Functional_Robot	11/14/2024 10:51 AM	File folder	

## Upload music

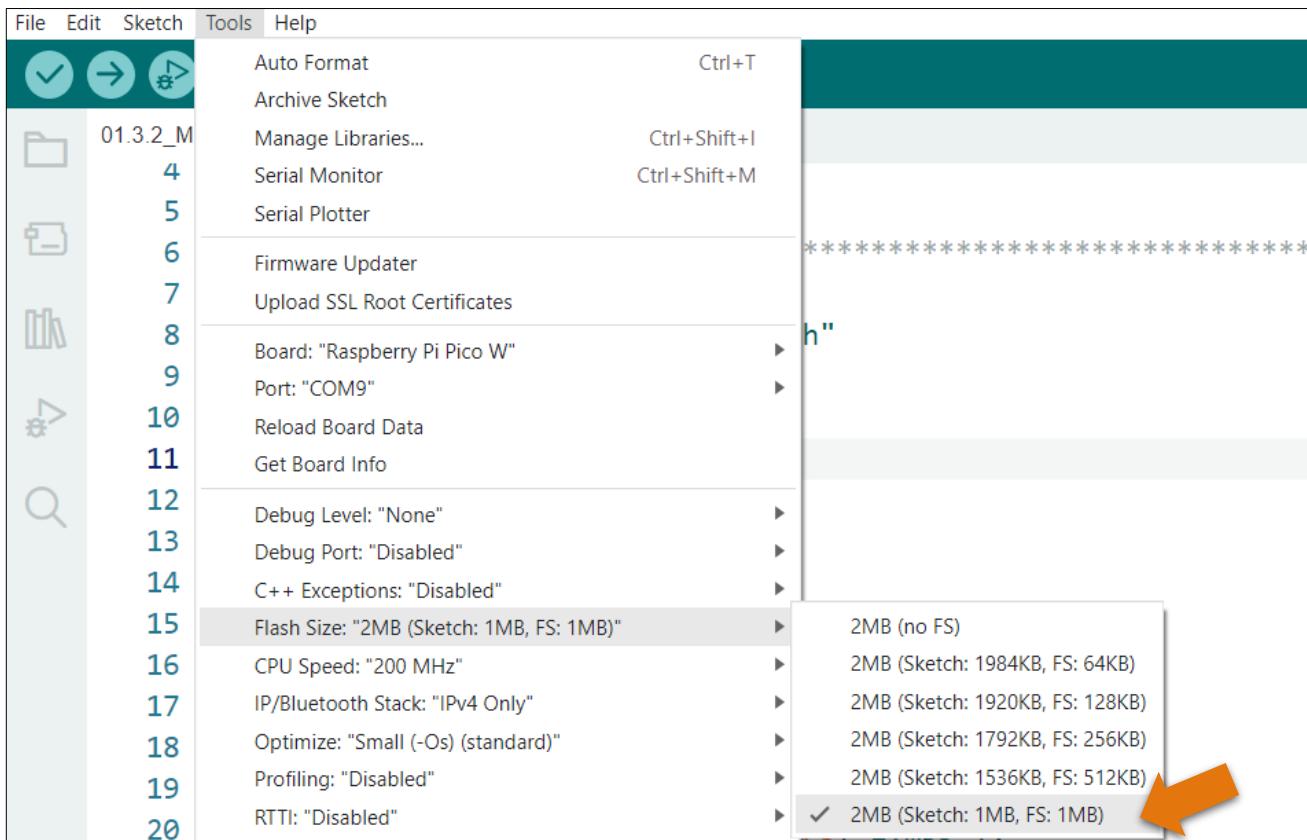
Before uploading the sketch, ensure that the following two libraries have been installed. If not, please do so first.

For ESP8266Audio library refer to [here](#) to install.

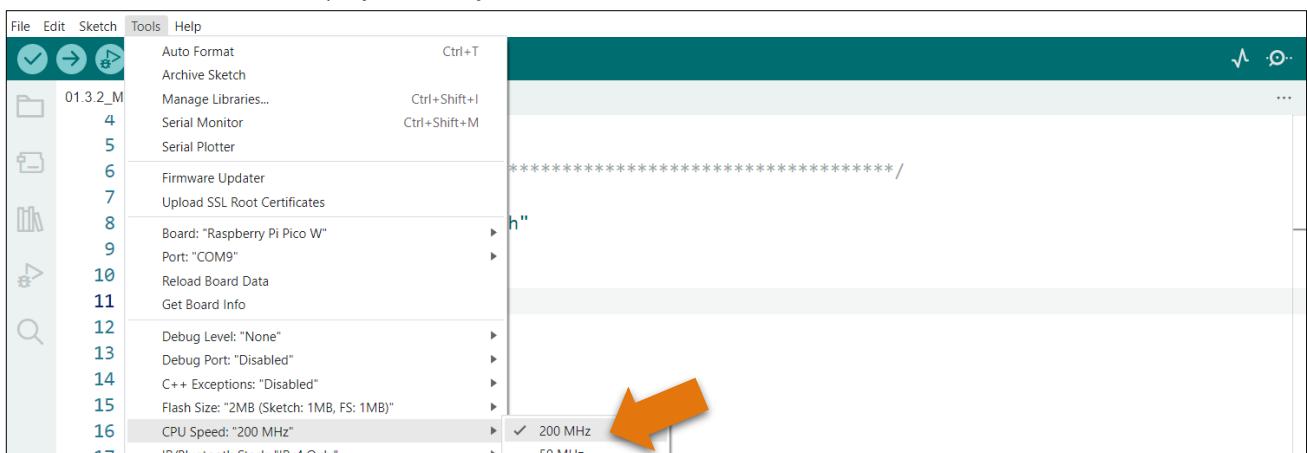
For PicoLittleFS tool library, refer to [this](#).

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

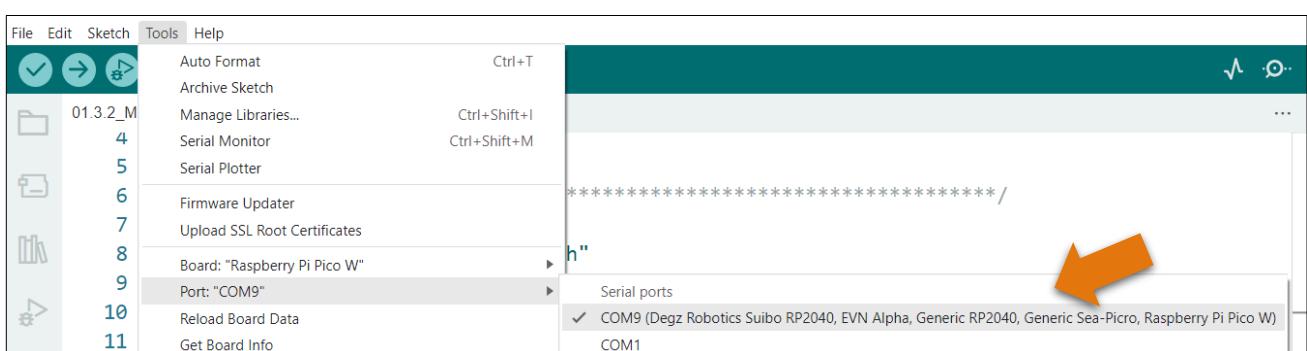
Open Arduino, select Tools from the menu bar, select Flash Size, and allocate 1MB of Flash space to store codes and 1MB to store audio files.



Ensure that the CPU speed is beyond 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.

```

File Edit Sketch Tools Help
Raspberry Pi Pico W
03.1_Sing_and_dance.ino >Upload LittleFS to Pico/ESP8266
 99 void setUploadLittleFS() {
100   Serial.begin(115200);
101   EEPROM.begin(512); //Initialize the
102   Bipedal_Robot.init(LeftLeg, RightLeg, LeftFoot, RightFoot, true); //Set the servo pins
103   //Bipedal_Robot.setTrims(calibratePosition[0], calibratePosition[1], calibratePosition[2]);
104   calib_homePos();
105   Bipedal_Robot.saveTrimsOnEEPROM();
106   EEPROM.commit();
107   Bipedal_Robot.home();
108   delay(50);
109 }
110 void loop() {

```

```

File Edit Sketch Tools Help
Raspberry Pi Pico W
03.1_Sing_and_dance.ino Bipedal_Robot.cpp Bipedal_Robot.h Bipedal_Robot_sounds.h Oscillator.cpp Oscillator.h
 99 void setup() {
100   Serial.begin(115200);
101   EEPROM.begin(512); //Initialize the
102   Bipedal_Robot.init(LeftLeg, RightLeg, LeftFoot, RightFoot, true); //Set the servo pins
103   //Bipedal_Robot.setTrims(calibratePosition[0], calibratePosition[1], calibratePosition[2]);
104   calib_homePos();
105   Bipedal_Robot.saveTrimsOnEEPROM();
106   EEPROM.commit();
107   Bipedal_Robot.home();
108   delay(50);
109 }
110 ...

```

LittleFS Upload x

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

LittleFS upload completed!

Ln 1, Col 1 Raspberry Pi Pico W on COM9

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.

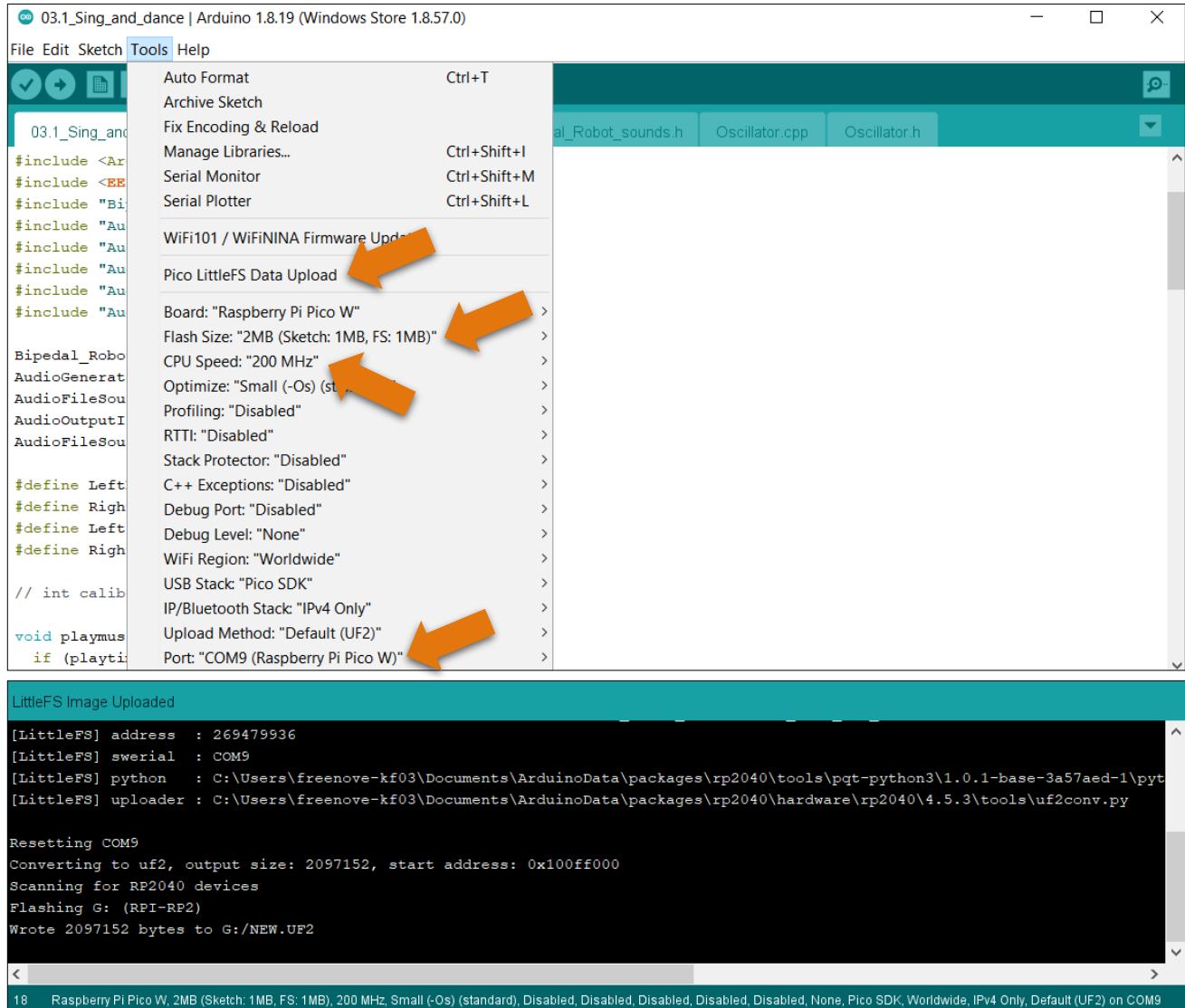
Name	Date modified	Type	Size
data	11/14/2024 10:51 AM	File folder	
tools	4/30/2025 9:11 AM	File folder	
03.1_Sing_and_dance.ino	11/14/2024 10:51 AM	INO File	

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.

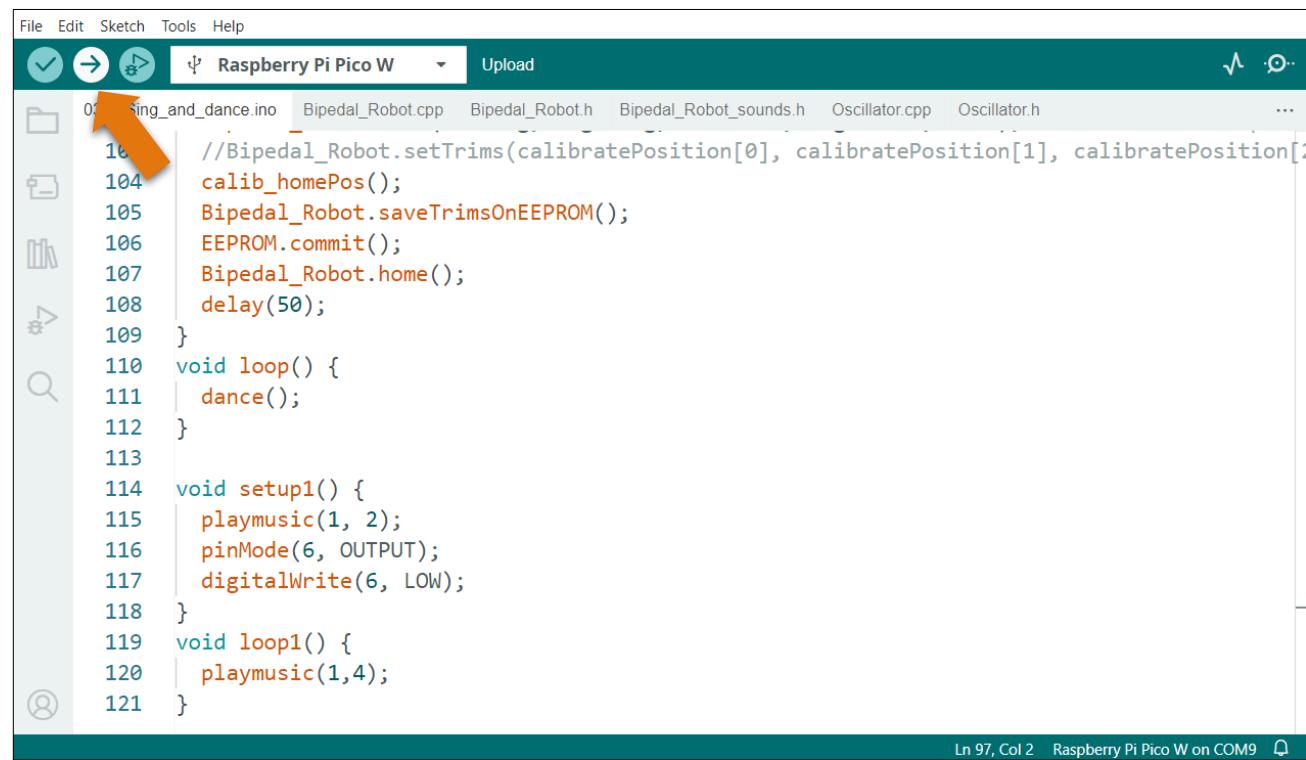
If your Arduino IDE is an old version one, like Arduino 1.x.x, please upload the music as below:

Click Arduino IDE Tools and click following content:



## Sketch

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



```

File Edit Sketch Tools Help
Raspberry Pi Pico W Upload
Sing_and_dance.ino Bipedal_Robot.cpp Bipedal_Robot.h Bipedal_Robot_sounds.h Oscillator.cpp Oscillator.h ...
1 //Bipedal_Robot.setTrims(calibratePosition[0], calibratePosition[1], calibratePosition[2]
104 calib_homePos();
105 Bipedal_Robot.saveTrimsOnEEPROM();
106 EEPROM.commit();
107 Bipedal_Robot.home();
108 delay(50);
109 }
110 void loop() {
111   dance();
112 }
113
114 void setup1() {
115   playmusic(1, 2);
116   pinMode(6, OUTPUT);
117   digitalWrite(6, LOW);
118 }
119 void loop1() {
120   playmusic(1,4);
121 }

```

Ln 97, Col 2 Raspberry Pi Pico W on COM9

### Part of code

```

1 #include <Arduino.h>
2 #include <EEPROM.h>
3 #include "Bipedal_Robot.h"
4 #include "AudioFileSourceLittleFS.h"
5 #include "AudioGeneratorMP3.h"
6 #include "AudioOutputI2SNoDAC.h"
7 #include "AudioFileSourceID3.h"
8 #include "AudioOutputI2S.h"
9
10 Bipedal_Robot Bipedal_Robot;
11 AudioGeneratorMP3 *mp3;
12 AudioFileSourceLittleFS *file;
13 AudioOutputI2SNoDAC *out;
14 AudioFileSourceID3 *id3;
15
16 #define LeftLeg 10
17 #define RightLeg 12
18 #define LeftFoot 11
19 #define RightFoot 13

```

```

20 #define Buzzer 6
21
22 // int calibratePosition[4] = { -17, -10, -15, 12 };
23
24 void setup() {
25     Serial.begin(115200);
26     EEPROM.begin(512); //Initialize the ultrasonic module
27     Bipedal_Robot.init(LeftLeg, RightLeg, LeftFoot, RightFoot, true); //Set the servo pins
28     //Bipedal_Robot.setTrims(calibratePosition[0], calibratePosition[1], calibratePosition[2],
29     calibratePosition[3]);
30     calib_homePos();
31     Bipedal_Robot.saveTrimsOnEEPROM();
32     EEPROM.commit();
33     Bipedal_Robot.home();
34     delay(50);
35 }
36
37
38 void setup1() {
39     playmusic(1, 2);
40     pinMode(6, OUTPUT);
41     digitalWrite(6, LOW);
42 }
43 void loop1() {
44     playmusic(1, 4);
45 }
```

## 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.

```

34 void loop() {
35     dance();
36 }
...
43 void loop1() {
44     playmusic(1, 4);
45 }
```

# Chapter 4 Infrared Robot

## 4.1 Introduction of infrared reception function

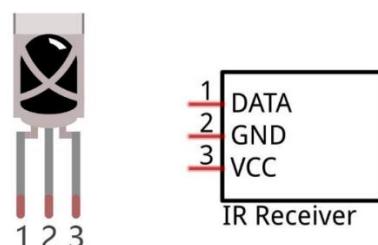
### Infrared Remote

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.

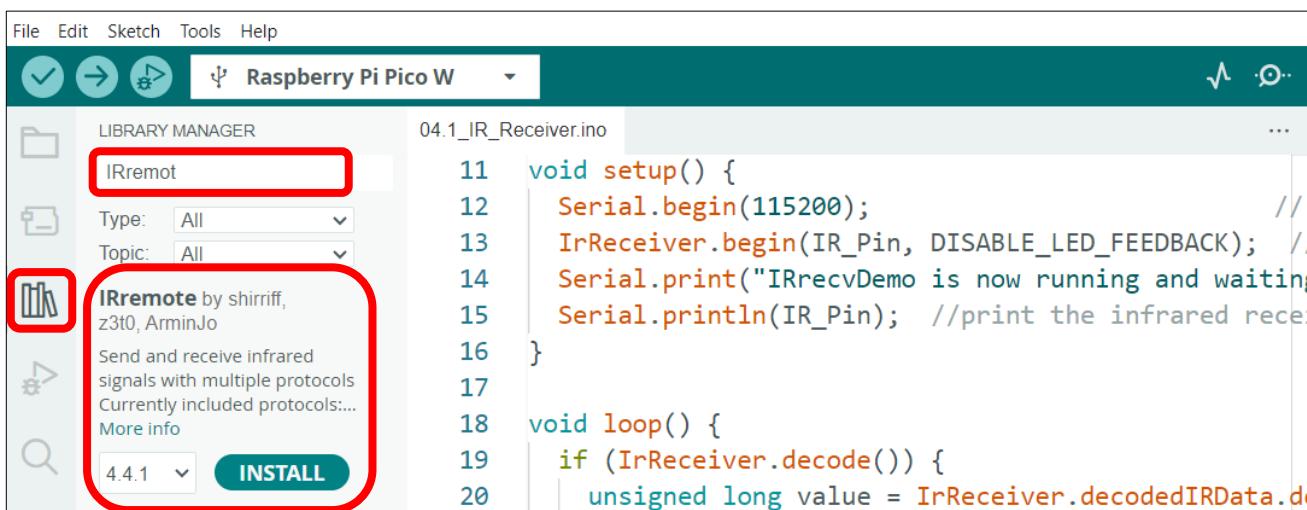
ICON	KEY Value	ICON	KEY Value
	BA45FF00		F20DFF00
	B847FF00		F30cff00
	BB44FF00		E718FF00
	BF40FF00		A15EFF00
	BC43FF00		F708FF00
	F807FF00		E31CFF00
	EA15FF00		A55AFF00
	F609FF00		BD42FF00
	E916FF00		AD52FF00
	E619FF00		B54AFF00

## Install IRremote library

In this project, we use a third-party library named **IRremote**. Please install it first.

Open Arduino IDE, click **Library Manager** on the left, and search “**IRremote**” to install.

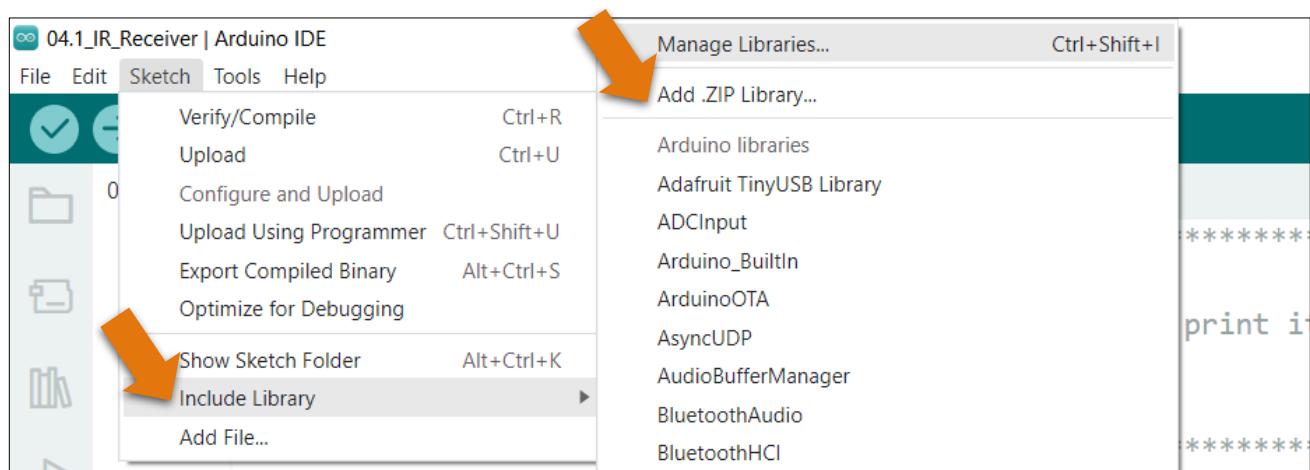
It is recommended to use version **4.4.1** of the IRremote library to avoid compatibility issues.



The second way, open Arduino IDE, click Sketch → Include Library → Add .ZIP Library. In the pop-up window, find the file named “./Libraries/ **IRremote\_V4.4.1\_20241029.Zip**” which locates in this directory, and click

Need support? support.freenove.com

OPEN.



Name	Date modified	Size
Adafruit_NeoPixel-V1.12.5.zip	9/6/2025 9:02 AM	87 KB
Adafruit-GFX_V1.12.1.zip	9/6/2025 8:58 AM	349 KB
ESP8266Audio-V1.9.7.zip	9/6/2025 9:02 AM	7,648 KB
Freenove_VK16K33.Lib-V1.0.0.zip	9/6/2025 9:02 AM	14 KB
<b>IRremote_V4.4.1_20241029.zip</b>	9/6/2025 9:02 AM	1,350 KB

## Sketch

Each time when you press the infrared remote control, the robot 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".

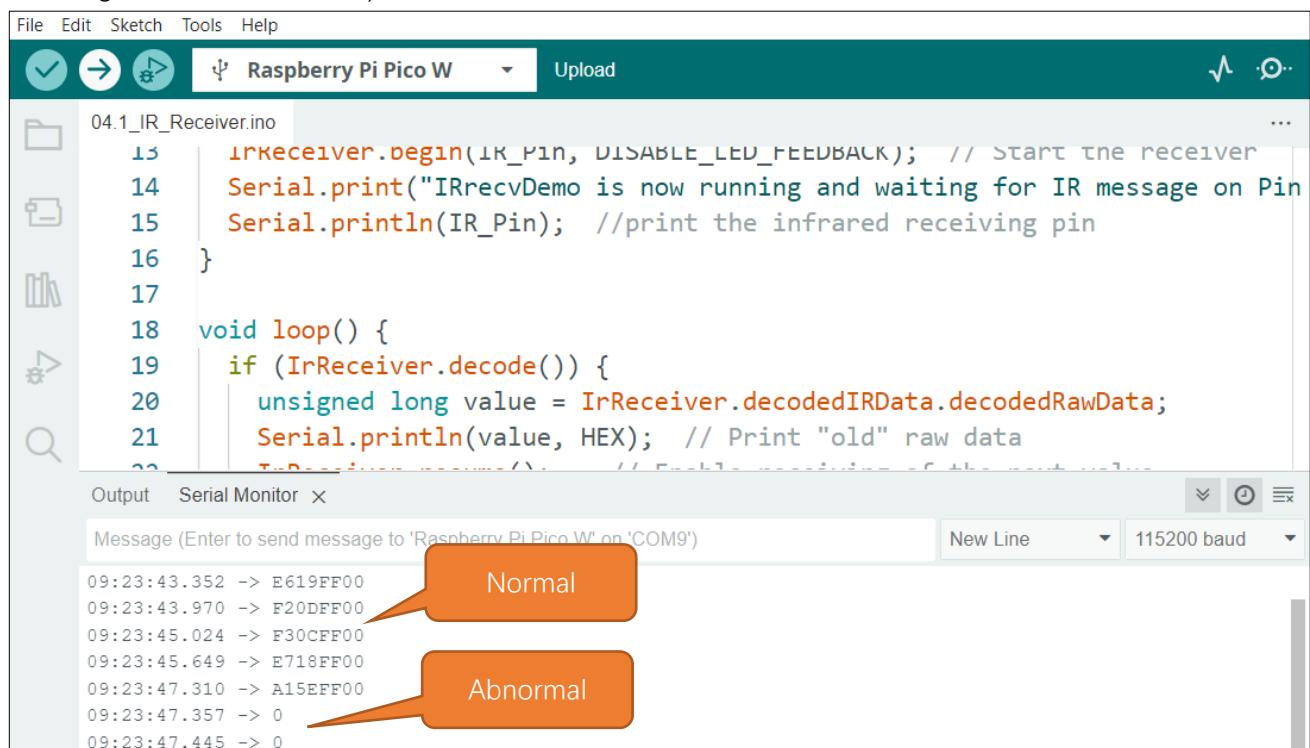
Name	Date modified	Type	Size
02.2_Ultrasonic_Ranging_Robot	11/14/2024 10:51 AM	File folder	
03.1_Sing_and_dance	5/7/2025 5:41 PM	File folder	
<b>04.1_IR_Receiver</b>	11/14/2024 10:51 AM	File folder	
04.2_IR_Receiver_Robot	11/14/2024 10:51 AM	File folder	
04.3_Multi_Functional_Robot	11/14/2024 10:51 AM	File folder	
04.4_Multi_Functional_Robot	11/14/2024 10:51 AM	File folder	
05.1_Bluetooth_Robot	11/14/2024 10:51 AM	File folder	

## Code

```

1 #include <IRremote.hpp>
2 #define IR_Pin 3 // Infrared receiving pin
3 #define ENABLE_LED_FEEDBACK true
4 #define DISABLE_LED_FEEDBACK false
5 void setup() {
6     Serial.begin(115200); // Initialize the serial port and set the
7     baud rate to 115200
8     IrReceiver.begin(IR_Pin, DISABLE_LED_FEEDBACK); // Start the receiver
9     Serial.print("IRrecvDemo is now running and waiting for IR message on Pin ");
10    Serial.println(IR_Pin); //print the infrared receiving pin
11 }
12
13 void loop() {
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     }
19 }
```

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.

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

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

```
2 #define IR_Pin 3 // Infrared receiving pin  
...  
8 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”.

Freenove_Bipedal_Robot_Kit_for_Raspberry_Pi_Pico-main > Sketches >			
Name	Date modified	Type	Size
02.2_Ultrasonic_Ranging_Robot	11/14/2024 10:51 AM	File folder	
03.1_Sing_and_dance	5/7/2025 5:41 PM	File folder	
04.1_IR_Receiver	11/14/2024 10:51 AM	File folder	
04.2_IR_Receiver_Robot	11/14/2024 10:51 AM	File folder	
04.3_Multi_Functional_Robot	11/14/2024 10:51 AM	File folder	
04.4_Multi_Functional_Robot	11/14/2024 10:51 AM	File folder	
05.1_Bluetooth_Robot	11/14/2024 10:51 AM	File folder	
05.2_Bluetooth_Remote_Robot	11/14/2024 10:51 AM	File folder	
05.3_Multi_Functional_Robot	11/14/2024 10:51 AM	File folder	

### Code

```

1 #include <Arduino.h>
2 #include <IRremote.hpp>
3 #include <EEPROM.h>
4 #include "Bipedal_Robot.h"
5
6 #define LeftLeg 10
7 #define RightLeg 12
8 #define LeftFoot 11
9 #define RightFoot 13
10
11 #define IR_Pin 3
12 #define ENABLE_LED_FEEDBACK true
13 #define DISABLE_LED_FEEDBACK false
14
15 // #define USE_NO_SEND_PWM
16
17 int move_flag = 0;

```

```
18
19 Bipedal_Robot Bipedal_Robot;
20 int calibratePosition[4] = { -17, -10, -15, 12 };
21
22 void calib_homePos() {
23     int servoPos[4];
24     servoPos[0] = 90;
25     servoPos[1] = 90;
26     servoPos[2] = 90;
27     servoPos[3] = 90;
28     Bipedal_Robot._moveServos(500, servoPos);
29     Bipedal_Robot.detachServos();
30 }
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:
75         move_flag = 0;
76         break;
77     }
78 }
79 void Move(int data) {
80     // Handle the commands
81     switch (data) {
82         case 1: // Receive the number '+'
83             Bipedal_Robot.walk(2, 1500, 1);
84             break;
85         case 2: // Receive the number '-'
86             Bipedal_Robot.walk(2, 1500, -1);
87             break;
88         case 3: // Receive the number '|<<|
89             Bipedal_Robot.turn(2, 1500, 1);
90             break;
91         case 4: // Receive the number '|>>|'
92             Bipedal_Robot.turn(2, 1500, -1);
93             break;
94         case 0: // Receive the number '>'|
95             Bipedal_Robot.home();
96             break;
97         default:
98             move_flag = 0;
99             Bipedal_Robot.home();
100            break;
101        }
102    }
```

Compile and upload the code to Raspberry Pi Pico (W).

		Move forward
		Turn left
		Turn right
		Move back

### Code Explanation:

Variable **IrReceiver.DecodedIRData.DecodedRawData** holds the infrared remote control encoding information; call handlecontrol 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.

```

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 }
```

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

```

56 void handleControl(unsigned long value) {
57     // Handle the commands
58     switch (value) {
59         case 0xBF40FF00: // Receive the number '+'
60             ...
61     }
62 }
```

```
62     case 0xE619FF00: // Receive the number '-'
...
65     ...
66     case 0xF807FF00: // Receive the number '|<<|'
...
68     ...
69     case 0xF609FF00: // Receive the number '|>>|'
...
71     ...
72     case 0xEA15FF00: // Receive the number '▶'
...
74     default:
...
75     ...
76   }
77 }
```

## 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 “04.3\_Multi\_Functional\_Robot” in the “**Freenove\_Bipedal\_Robot\_Kit\_for\_Raspberry\_Pi\_Pico\Sketches**” and double click “04.3\_Multi\_Functional\_Robot.ino”.

Freenove_Bipedal_Robot_Kit_for_Raspberry_Pi_Pico-main > Sketches >			
Name	Date modified	Type	Size
02.2_Ultrasonic_Ranging_Robot	11/14/2024 10:51 AM	File folder	
03.1_Sing_and_dance	5/7/2025 5:41 PM	File folder	
04.1_IR_Receiver	11/14/2024 10:51 AM	File folder	
04.2_IR_Receiver_Robot	11/14/2024 10:51 AM	File folder	
04.3_Multi_Functional_Robot	11/14/2024 10:51 AM	File folder	
04.4_Multi_Functional_Robot	11/14/2024 10:51 AM	File folder	
05.1_Bluetooth_Robot	11/14/2024 10:51 AM	File folder	
05.2_Bluetooth_Remote_Robot	11/14/2024 10:51 AM	File folder	
05.3_Multi_Functional_Robot	11/14/2024 10:51 AM	File folder	

### Code

```
1 #include <Arduino.h>
2 #include <Adafruit_NeoPixel.h>
3 #include <IRremote.hpp>
4 #include <EEPROM.h>
5 #include "Freenove_Robot_For_Pico_W.h"
6 #include "Freenove_Robot_Emotion.h"
7 #include "Freenove_VK16K33_Lib.h"
8 #include "Freenove_Robot_WS2812.h"
9 #include "Bipodal_Robot.h"
10
11 #define LeftLeg 10
12 #define RightLeg 12
13 #define LeftFoot 11
14 #define RightFoot 13
15
16 #define IR_Pin 3 // Infrared receiving pin
17 #define ENABLE_LED_FEEDBACK true
18 #define DISABLE_LED_FEEDBACK false
19
```

```

20 int move_flag = 0;
21 extern int move_flag;
22
23 Bipedal_Robot Bipedal_Robot;
24 int calibratePosition[4] = { -17, -10, -15, 12 };
25 int emotion_flag = 0;
26 int ws2812_flag = 0;
27
28 void calib_homePos() {
29     int servoPos[4];
30     servoPos[0] = 90;
31     servoPos[1] = 90;
32     servoPos[2] = 90;
33     servoPos[3] = 90;
34     Bipedal_Robot._moveServos(500, servoPos);
35     Bipedal_Robot.detachServos();
36 }
37
38 void setup() {
39     Serial.begin(115200); //Turn on the serial port monitor and set the baud rate to 115200
40     WS2812_Setup(); //WS2812 initialization
41     Buzzer_Setup(); //Initialize the buzzer
42     Ultrasonic_Setup();
43     EEPROM.begin(512);
44     Bipedal_Robot.init(LeftLeg, RightLeg, LeftFoot, RightFoot, true); //Set the servo pins
45     //Bipedal_Robot.setTrims(calibratePosition[0], calibratePosition[1], calibratePosition[2],
46     calibratePosition[3]);
47     calib_homePos();
48     Bipedal_Robot.saveTrimsOnEEPROM();
49     EEPROM.commit();
50     Bipedal_Robot.home();
51     IrReceiver.begin(IR_Pin, DISABLE_LED_FEEDBACK); // Start the receiver
52     delay(100);
53 }
54
55 void loop() {
56     if (IrReceiver.decode()) {
57         unsigned long value = IrReceiver.decodedIRData.decodedRawData;
58         handleControl(value); // Handle the commands from remote control
59         Serial.println(value, HEX); // Print "old" raw data
60         Serial.println();
61         IrReceiver.resume(); // Enable receiving of the next value
62     }
63     Emotion_Detection();

```

```
64 Emotion_Show(emotion_task_mode); //Led matrix display function
65 WS2812_Show(ws2812_task_mode); //Car color lights display function
66 Move(move_flag);
67 }
68
69 void handleControl(unsigned long value) {
70 // Handle the commands
71 switch (value) {
72 case 0xBF40FF00: // Receive the number '+'
73     move_flag = 1;
74     break;
75 case 0xE619FF00: // Receive the number '-'
76     move_flag = 2;
77     break;
78 case 0xF807FF00: // Receive the number '|<>|'
79     move_flag = 3;
80     break;
81 case 0xF609FF00: // Receive the number '>>|'
82     move_flag = 4;
83     break;
84 case 0xEA15FF00: // Receive the number '▶'
85     move_flag = 0;
86     break;
87 case 0xE916FF00: // Receive the number '0'
88     break;
89 case 0xF30CFF00: // Receive the number '1'
90     break;
91 case 0xF708FF00: // Receive the number '4'
92     break;
93 case 0xF20DFF00: // Receive the number 'C'
94     break;
95 case 0xA15EFF00: // Receive the number '3'
96     move_flag = 5;
97     break;
98 case 0xA55AFF00: // Receive the number '6'
99     move_flag = 6;
100    break;
101   case 0xB54AFF00: // Receive the number '9'
102     break;
103   case 0xBB44FF00: // Receive the number 'TEST'
104     Buzzer_Variable(2000, 100, 1);
105     break;
106   case 0xE718FF00: // Receive the number '2'
107     emotion_flag = emotion_flag + 1;
```

```
108     if (emotion_flag > 7) {
109         emotion_flag = 0;
110     }
111     Emotion_SetMode(emotion_flag); //Display
112     break;
113 case 0xE31CFF00: // Receive the number '5'
114     emotion_flag = 0;
115     Emotion_SetMode(emotion_flag);
116     break;
117 case 0xBD42FF00: // Receive the number '7'
118     ws2812_flag = ws2812_flag + 1;
119     if (ws2812_flag >= 6) {
120         ws2812_flag = 0;
121     }
122     WS2812_SetMode(ws2812_flag);
123     break;
124 case 0xAD52FF00: // Receive the number '8'
125     ws2812_flag = 0;
126     WS2812_SetMode(ws2812_flag);
127     break;
128 case 0xFFFFFFFF: // Remain unchanged
129     break;
130 default:
131     break;
132 }
133 }
134
135 void Move(int data) {
136     // Handle the commands
137     switch (data) {
138         case 1: // Receive the number '+'
139             Bipedal_Robot.walk(2, 1500, 1);
140             move_flag = 0;
141             break;
142         case 2: // Receive the number '-'
143             Bipedal_Robot.walk(2, 1500, -1);
144             move_flag = 0;
145             break;
146         case 3: // Receive the number '|<<'
147             Bipedal_Robot.turn(2, 1500, 1);
148             move_flag = 0;
149             break;
150         case 4: // Receive the number '|>>'
151             Bipedal_Robot.turn(2, 1500, -1);
```

```
152     move_flag = 0;
153     break;
154 case 5: // Receive the number '3'
155     Ultrasonic_Avoid();
156     break;
157 case 6: // Receive the number '6'
158     dance();
159     move_flag = 0;
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     Bipedal_Robot.jitter(1, 1000, 40);
173     Bipedal_Robot.home();
174     Bipedal_Robot.moonwalker(1, 1200, 30, 1);
175     Bipedal_Robot.home();
176     Bipedal_RobotascendingTurn(1, 1000, 50);
177     Bipedal_Robot.home();
178     Bipedal_Robot.tiptoeSwing(1, 1000, 30);
179     Bipedal_Robot.home();
180     Bipedal_Robot.flapping(1, 1000, 40, 1);
181     Bipedal_Robot.home();
182     Bipedal_Robot.crusaito(1, 3000, 40, 1);
183     Bipedal_Robot.home();
184     Bipedal_Robot.shakeLeg(1, 1000, 1);
185     Bipedal_Robot.home();
186 }
```

After the code uploads successfully, turn on the power of the robot and use the infrared remote to control the robot 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		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

### Code Explanation:

Add the header file for the robot.

```

1 #include <Arduino.h>
2 #include <Adafruit_NeoPixel.h>
3 #include <IRremote.hpp>
4 #include <EEPROM.h>
5 #include "Freenove_Robot_For_Pico_W.h"
6 #include "Freenove_Robot_Emotion.h"
7 #include "Freenove_VK16K33_Lib.h"
8 #include "Freenove_Robot_WS2812.h"
9 #include "Bipedal_Robot.h"
```

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

```
74 void handleControl(unsigned long value) {
```

Need support? [support.freenove.com](mailto:support.freenove.com)

```
75 // Handle the commands
76 switch (value) {
77     case 0xBF40FF00: // Receive the number '+'
78     ...
79     ...
80     case 0xE619FF00: // Receive the number '-'
81     ...
82     ...
83     case 0xF807FF00: // Receive the number '|<>|'
84     ...
85     ...
86     case 0xF609FF00: // Receive the number '>>|'
87     ...
88     ...
89     case 0xEA15FF00: // Receive the number '▶'
90     ...
91     ...
92     case 0xE916FF00: // Receive the number '0'
93     ...
94     ...
95     case 0xF30CFF00: // Receive the number '1'
96     ...
97     ...
98     case 0xF708FF00: // Receive the number '4'
99     ...
100    ...
101    case 0xF20DFF00: // Receive the number 'C'
102    ...
103    case 0xA15EFF00: // Receive the number '3'
104    ...
105    ...
106    case 0xA55AFF00: // Receive the number '6'
107    ...
108    ...
109    case 0xB54AFF00: // Receive the number '9'
110    ...
111    ...
112    case 0xBB44FF00: // Receive the number 'TEST'
113    ...
114    ...
115    case 0xE718FF00: // Receive the number '2'
116    ...
117    ...
118    case 0xE31cff00: // Receive the number '5'
119    ...
120    ...
121    case 0xBD42FF00: // Receive the number '7'
122    ...
123    ...
124    case 0xAD52FF00: // Receive the number '8'
125    ...
126    ...
127    case 0xFFFFFFFF: // Remain unchanged
128    ...
129    ...
130    default:
131    ...
132    ...
133    }
134    }
135 }
```

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

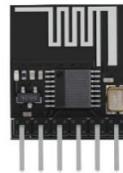
This section requires an Android or iPhone device with Bluetooth to control the robot.  
If you have any concerns, please feel free to contact us via [support@freenove.com](mailto:support@freenove.com)

## 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:



### 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 9600, 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?

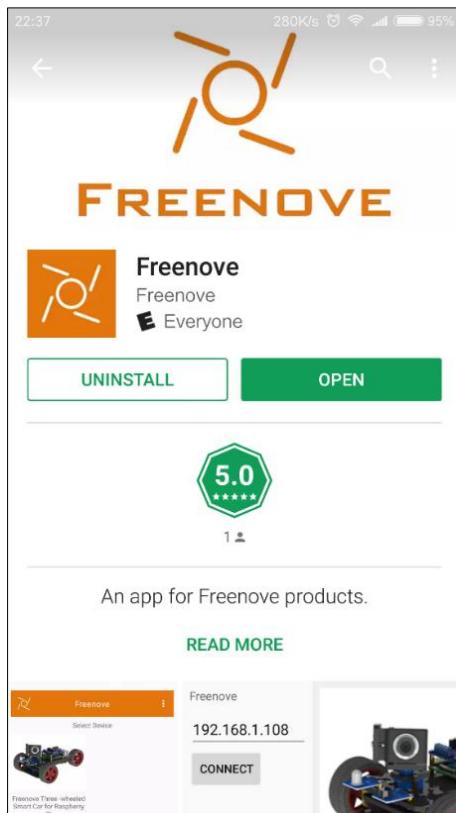
Use the command "AT+NAME?" to check the Bluetooth name or "AT+NAME=X" to set a new name (replace X with your desired name).

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

## Install Freenove app

There are three ways to install our app.

Method 1: Use Google play to search “Freenove”, download and install.



Method 2: Visit <https://play.google.com/store/apps/details?id=com.freenove.suhayl.Freenove>, and click install.

Freenove

10,000+ Downloads 3+ years old

Install share Add to Wishlist

Use this app on all your devices

Application Support

Freenove

10,000+ Downloads 3+ years old

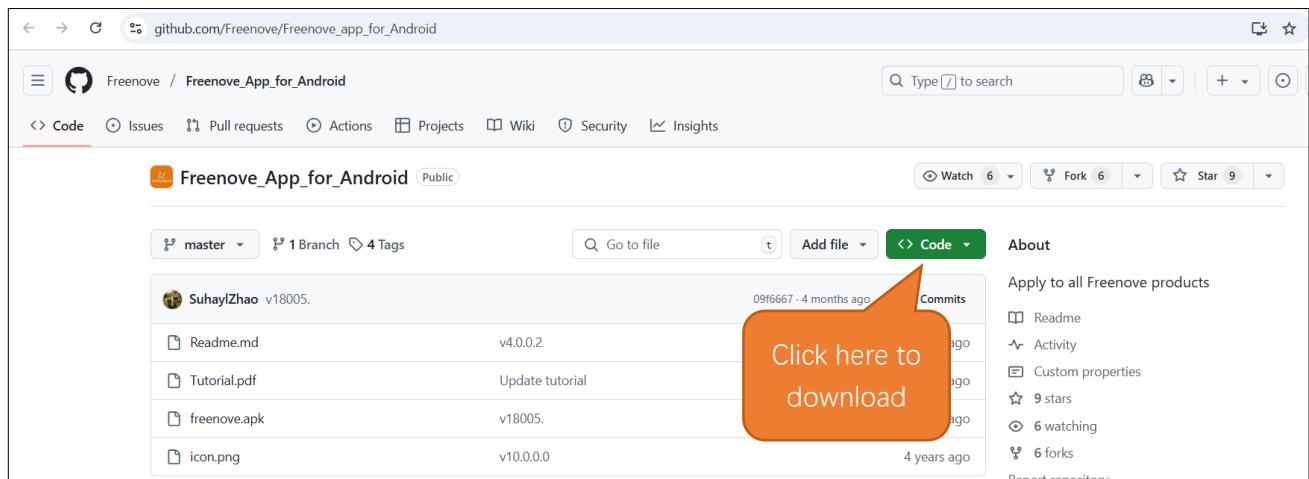
Install share Add to Wishlist

Use this app on all your devices

Application Support

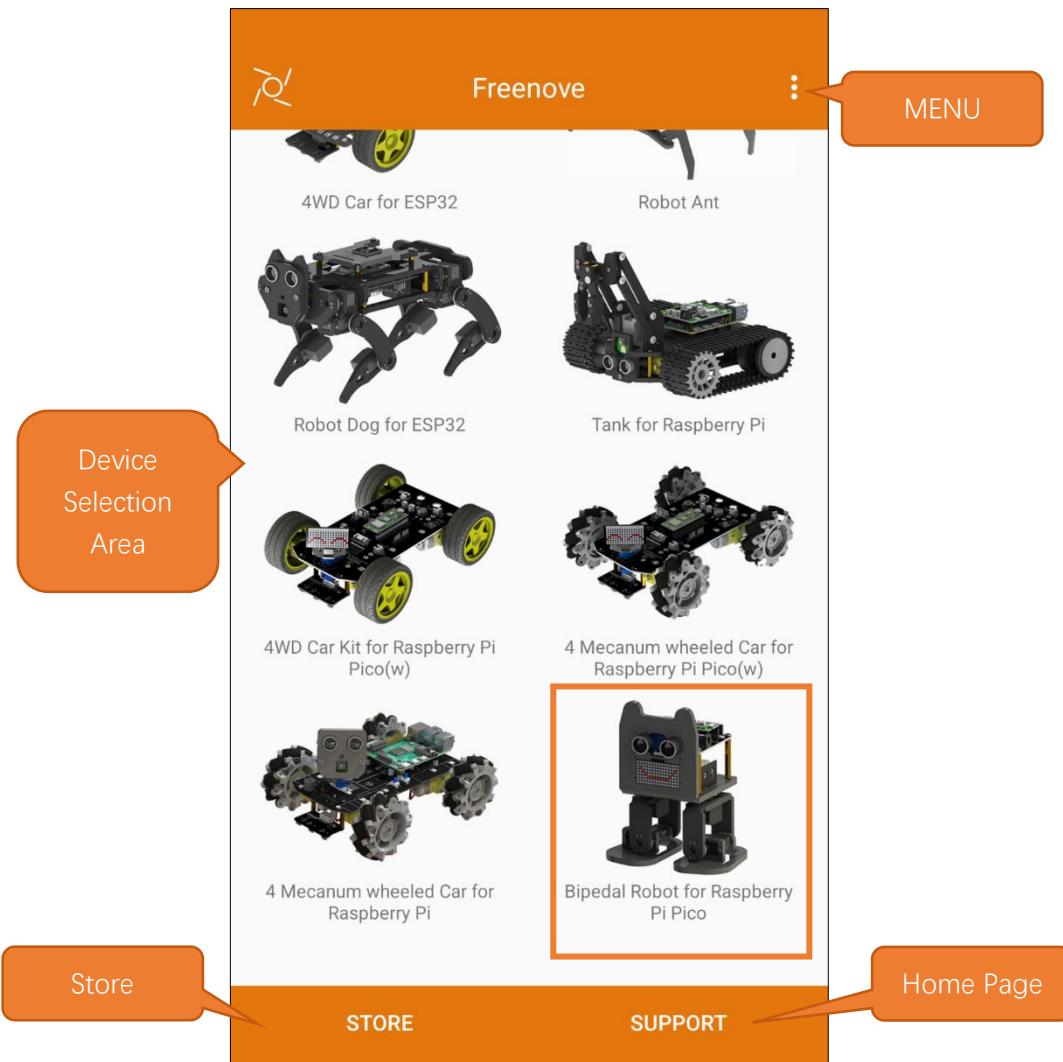
Need support? ✉ [support.freenove.com](mailto:support.freenove.com)

Method 3: Visit [https://github.com/Freenove/Freenove\\_app\\_for\\_Android](https://github.com/Freenove/Freenove_app_for_Android), download the files in this library, and install freenove.apk to your Android phone manually.



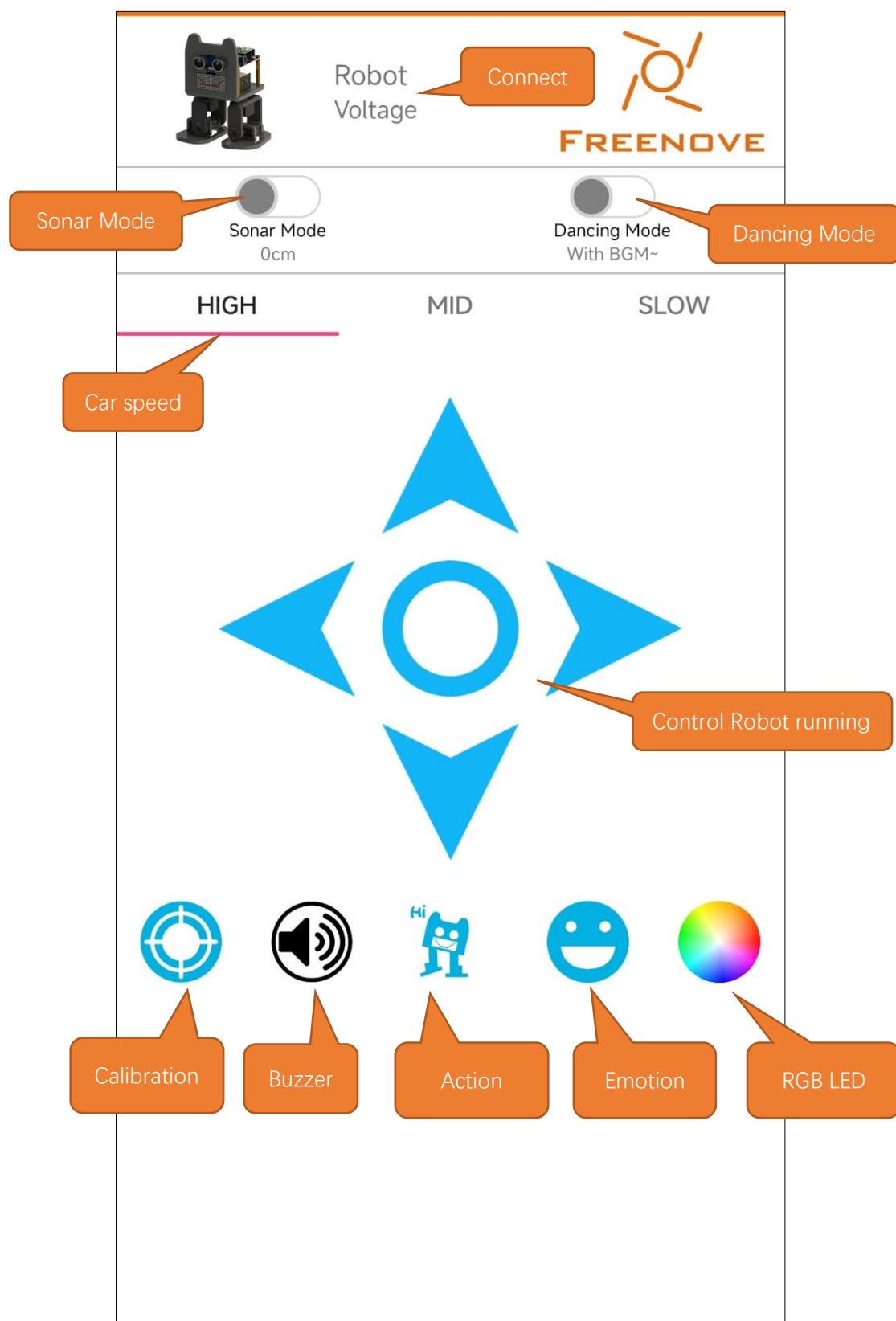
## Introduction to the APP

In this chapter, we use the Freenove APP to control the robot. Open the APP on your phone, select “Bipedal Robot for Raspberry Pi Pico”.



Need support? [✉ support.freenove.com](mailto:support.freenove.com)

The interface is as shown below. Let's explore its features.



## Sketch

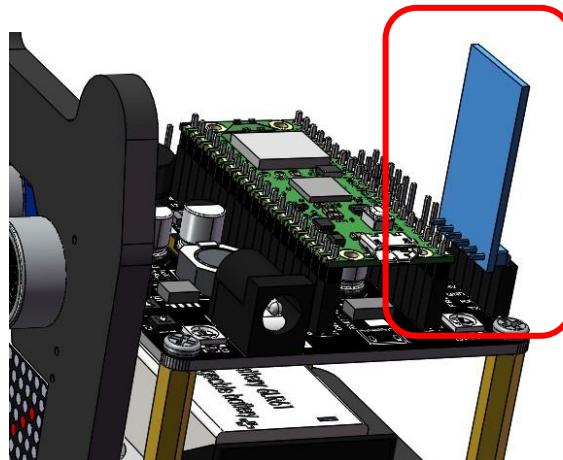
Open the folder “05.1\_Bluetooth\_Robot” in the “**Freenove\_Bipedal\_Robot\_Kit\_for\_Raspberry\_Pi\_Pico\Sketches**” and double click “05.1\_Bluetooth\_Robot.ino”.

Freenove_Bipedal_Robot_Kit_for_Raspberry_Pi_Pico-main > Sketches >			
Name	Date modified	Type	Size
02.2_Ultrasonic_Ranging_Robot	11/14/2024 10:51 AM	File folder	
03.1_Sing_and_dance	5/7/2025 5:41 PM	File folder	
04.1_IR_Receiver	11/14/2024 10:51 AM	File folder	
04.2_IR_Receiver_Robot	11/14/2024 10:51 AM	File folder	
04.3_Multi_Functional_Robot	11/14/2024 10:51 AM	File folder	
04.4_Multi_Functional_Robot	11/14/2024 10:51 AM	File folder	
05.1_Bluetooth_Robot	11/14/2024 10:51 AM	File folder	
05.2_Bluetooth_Remote_Robot	11/14/2024 10:51 AM	File folder	
05.3_Multi_Functional_Robot	11/14/2024 10:51 AM	File folder	

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 the robot as shown below. **Don't reverse it. The wrong connection may damage your hardware.**



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

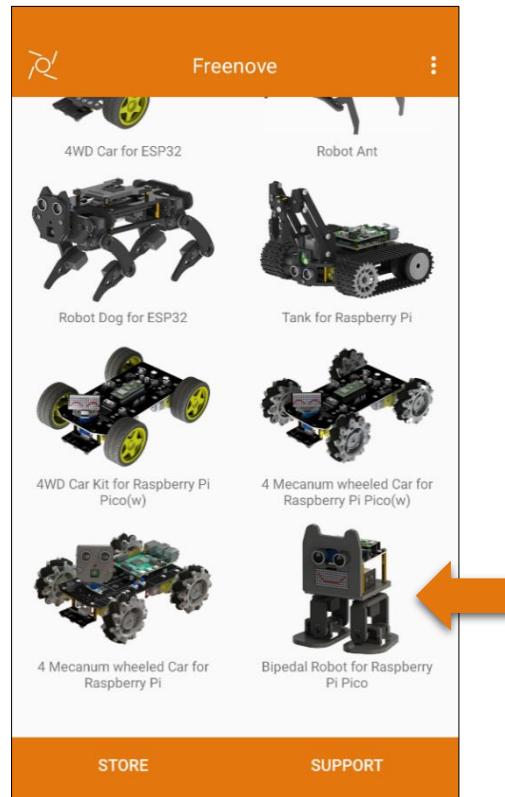
As shown in the figure below, the Bluetooth name is set to “BT05”, and set the baud rate to 9600.

```

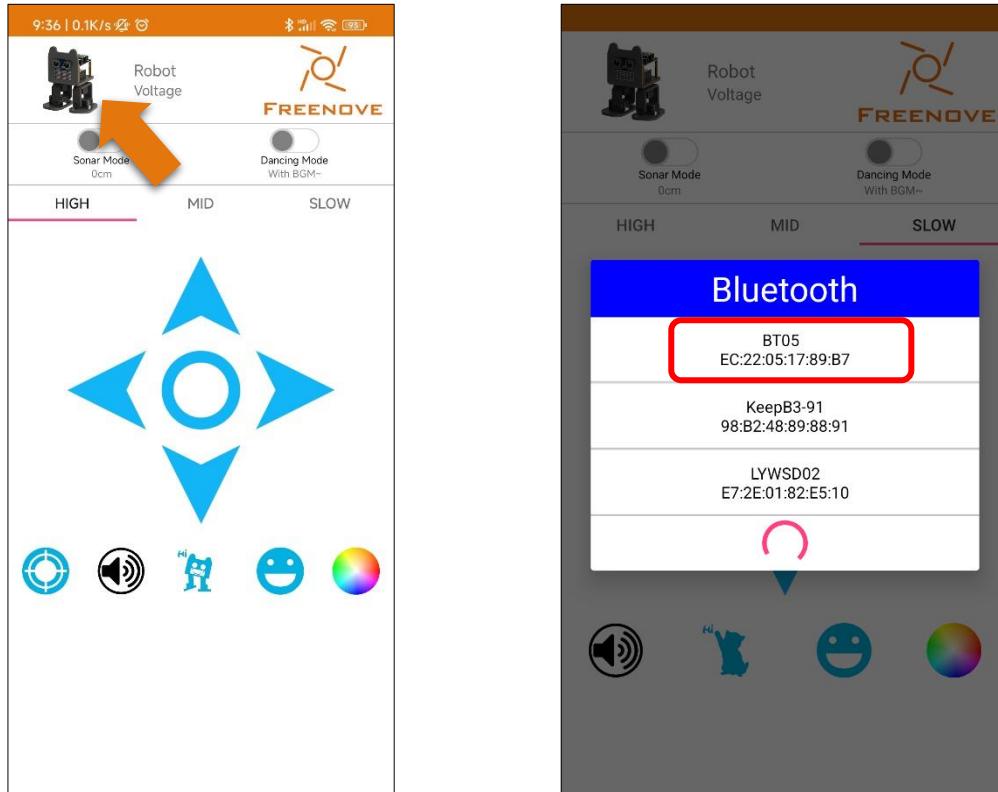
Output  Serial Monitor X
Message (Enter to send message to 'Raspberry Pi Pico W' on 'COM9')
New Line  9600 baud
13:45:12.372 -> Set the name of the Bluetooth module to BT05.
13:45:12.406 -> Set the Bluetooth baud rate to 9600.
Ln 49, Col 54  Raspberry Pi Pico W on COM9  ⌂  ⌚  ⌚

```

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



Click the following icon and select BT05.



If the connection successes, you can see "Connected". Operate the app, and observe the serial monitor.



The serial monitor will show contents as below:

A screenshot of a terminal window titled "Output Serial Monitor". The window has tabs for "Output" and "Serial Monitor". The "Serial Monitor" tab is active. It shows a message input field with placeholder text "Message (Enter to send message to 'Raspberry Pi Pico W' on 'COM9')". To the right are buttons for "New Line", "9600 baud", and a dropdown menu. The main pane displays a log of messages sent from the device to the terminal. The log entries are:  
13:47:45.857 -> C#4#255#196#1#  
13:47:45.857 ->  
13:47:49.449 -> M#1  
13:47:49.449 ->  
13:47:50.493 -> M#2  
13:47:50.540 ->  
13:47:52.222 -> D#2000#  
13:47:52.222 ->  
13:47:52.269 -> D#0#  
13:47:52.269 ->  
At the bottom of the terminal window, there is a status bar with the text "Ln 49, Col 54 Raspberry Pi Pico W on COM9" and some icons.

## 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

```

1 #include <Arduino.h>
2 #include <SoftwareSerial.h> // Reference library
3
4 #include "SerialCommand.h"
5 SoftwareSerial BTserial = SoftwareSerial(1, 0); // RX pin to 0 and TX pin to 1 on the board
6 SerialCommand SCmd(BTserial);
7
8 String inputString = ""; // a String to hold incoming data
9 bool stringComplete = false; // whether the string is complete
10 int baudRate[] = {2400, 9600, 19200, 115200};
11 int baudRateCount = sizeof(baudRate) / sizeof(baudRate[0]);
12
13 void receiveStop() {
14     delay(10);
15     Serial.flush();
16 }
17
18 void setup() {
19     Serial.begin(9600);
20     delay(2000);

```

```
21 // Loop through each baud rate and send commands
22 for(int i = 0; i < baudRateCount; i++) {
23     BTserial.begin(baudRate[i]);
24     delay(200);
25     BTserial.println("AT+NAME=BT05");//Set the radio called Robot
26     delay(200);
27     BTserial.println("AT+ROLE=0");
28     delay(200);
29     BTserial.println("AT+UART=2");//1=2400 2=9600 3=19200 4=115200
30     delay(200);
31
32     //SoftwareSerial does not support dynamic adjustment of baud rate, so close the serial
33     BTserial.end();
34 }
35
36 BTserial.begin(9600);
37 delay(2000);
38
39 SCmd.addDefaultHandler(receiveStop);
40 inputString.reserve(200);
41 Serial.println("Set the name of the Bluetooth module to BT05.");
42 Serial.println("Set the Bluetooth baud rate to 9600.");
43 // Serial.println("Set Bluetooth to slave mode.");
44 }
45
46 void loop() {
47     while (BTserial.available()) {
48         // get the new byte:
49         char inChar = (char)BTserial.read();
50         // add it to the inputString:
51         inputString += inChar;
52         // if the incoming character is a newline, set a flag so the main loop can
53         // do something about it:
54         if (inChar == '\n') {
55             stringComplete = true;
56         }
57         if (stringComplete) {
58             Serial.println(inputString);
59             // clear the string:
60             inputString = "";
61             stringComplete = false;
62         }
63     }
64 }
```

## Code Explanation

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

```

21 // Loop through each baud rate and send commands
22 for(int i = 0; i < baudRateCount; i++) {
23     BTserial.begin(baudRate[i]);
24     delay(200);
25     BTserial.println("AT+NAME=BT05");//Set the radio called Robot
26     delay(200);
27     BTserial.println("AT+ROLE=0");
28     delay(200);
29     BTserial.println("AT+UART=2");//1=2400 2=9600 3=19200 4=115200
30     delay(200);
31
32     //SoftwareSerial does not support dynamic adjustment of baud rate, so close the serial
33     BTserial.end();
34 }
35
36 BTserial.begin(9600);
37 delay(2000);
38
39 SCmd.addDefaultHandler(receiveStop);
40 inputString.reserve(200);
41 Serial.println("Set the name of the Bluetooth module to BT05.");
42 Serial.println("Set the Bluetooth baud rate to 9600.");
43 // Serial.println("Set Bluetooth to slave mode.");

```

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

```

47 while (BTserial.available()) {
48     // get the new byte:
49     char inChar = (char)BTserial.read();
50     // add it to the inputString:
51     inputString += inChar;
52     // if the incoming character is a newline, set a flag so the main loop can
53     // do something about it:
54     if (inChar == '\n') {
55         stringComplete = true;
56     }
57     if (stringComplete) {
58         Serial.println(inputString);
59         // clear the string:
60         inputString = "";
61         stringComplete = false;
62     }
63 }

```

## 5.2 Bluetooth Remote Robot

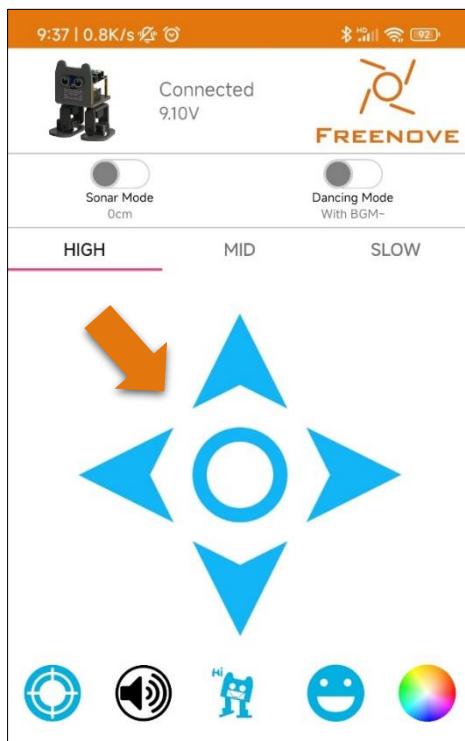
In the previous section, we have learned how to receive the data of the app via Bluetooth, and the meanings of the data format. Next, let's use the data to control the robot to make some basic movements, like moving forward and backward, turning left and right, etc.

### Sketch

Open the folder “05.2\_Bluetooth\_Remote\_Robot” in the “**Freenove\_Bipedal\_Robot\_Kit\_for\_Raspberry\_Pi\_Pico\Sketches**” and double click “05.2\_Bluetooth\_Remote\_Robot.ino”.

Freenove_Bipedal_Robot_Kit_for_Raspberry_Pi_Pico-main > Sketches >			
Name	Date modified	Type	Size
02.2_Ultrasonic_Ranging_Robot	11/14/2024 10:51 AM	File folder	
03.1_Sing_and_dance	5/7/2025 5:41 PM	File folder	
04.1_IR_Receiver	11/14/2024 10:51 AM	File folder	
04.2_IR_Receiver_Robot	11/14/2024 10:51 AM	File folder	
04.3_Multi_Functional_Robot	11/14/2024 10:51 AM	File folder	
04.4_Multi_Functional_Robot	11/14/2024 10:51 AM	File folder	
05.1_Bluetooth_Robot	11/14/2024 10:51 AM	File folder	
05.2_Bluetooth_Remote_Robot	11/14/2024 10:51 AM	File folder	
05.3_Multi_Functional_Robot	11/14/2024 10:51 AM	File folder	

Connect the App to the Bluetooth of the robot, and tap the directional keys to control its movements.



## Code

```
1 #include <Arduino.h>
2 #include <SoftwareSerial.h>
3 #include <EEPROM.h>
4 #include "Freenove_Robot_For_Pico_W.h"
5
6 #include "SerialCommand.h"
7 SoftwareSerial BTserial = SoftwareSerial(1, 0); // RX pin to 1 and TX pin to 0 on the board
8 SerialCommand SCmd(BTserial);
9 #include "Bipedal_Robot.h"
10 Bipedal_Robot Bipedal_Robot;
11
12 #define LeftLeg 10
13 #define RightLeg 12
14 #define LeftFoot 11
15 #define RightFoot 13
16
17 int YL;
18 int YR;
19 int RL;
20 int RR;
21
22 int T = 1000;
23 int moveId = 0;
24 bool isServoResting = true;
25
26 extern int moveId;
27 extern bool isServoResting;
28
29 int moveSize = 15;
30
31 String inputString = ""; // a String to hold incoming data
32 bool stringComplete = false; // whether the string is complete
33 int baudRate[] = {2400, 9600, 19200, 115200};
34 int baudRateCount = sizeof(baudRate) / sizeof(baudRate[0]);
35
36 void receiveStop() {
37     sendAck();
38     sendFinalAck();
39 }
40
41 void receiveBuzzer() {
42     sendAck();
```

```
43     bool error = false;
44     int freq;
45     int duration;
46     char *arg;
47     arg = SCmd.next();
48     if (arg != NULL) freq = atoi(arg);
49     else error = true;
50     Serial.println("\nfreq is:");
51     Serial.println(freq);
52     Buzzer_Variable(freq, 100, 1);
53     sendFinalAck();
54 }
55
56 void receiveMovement() {
57     sendAck();
58     Serial.print("move loop... ");
59     if (Bipedal_Robot.getRestState() == true) Bipedal_Robot.setRestState(false);
60     isServoResting = Bipedal_Robot.getRestState();
61     char *arg;
62     arg = SCmd.next();
63     if (arg != NULL) {
64         moveId = atoi(arg);
65         Serial.print("moveId:");
66         Serial.print(moveId);
67     } else {
68         moveId = 0;
69     }
70     arg = SCmd.next();
71     if (arg != NULL) {
72         T = atoi(arg);
73         if (T == 0) {
74             T = 1000;
75         } else if (T == 1) {
76             T = 1500;
77         } else if (T == 2) {
78             T = 2000;
79         }
80     } else T = 2000;
81     Serial.print("T is:");
82     Serial.print(T);
83     arg = SCmd.next();
84     if (arg != NULL) moveSize = atoi(arg);
85     else moveSize = 15;
86 }
```

```
87 void move(int moveId) {
88     // bool manualMode = false;
89     switch (moveId) {
90         case 0:
91             Bipedal_Robot.home();
92             // Serial.print("moveId 0 home");
93             break;
94         case 1:
95             Bipedal_Robot.walk(1, T, 1);
96             break;
97         case 2:
98             Bipedal_Robot.walk(1, T, -1);
99             break;
100        case 3:
101            Bipedal_Robot.turn(1, T, 1);
102            break;
103        case 4:
104            Bipedal_Robot.turn(1, T, -1);
105            break;
106        default:
107            break;
108    }
109 }
110
111 void sendAck() {
112     delay(10);
113     Serial.flush();
114 }
115 void sendFinalAck() {
116     delay(10);
117     Serial.flush();
118 }
119
120 void setup() {
121     Serial.begin(9600);
122     delay(2000);
123     // Loop through each baud rate and send commands
124     for(int i = 0; i < baudRateCount; i++) {
125         BTserial.begin(baudRate[i]);
126         delay(200);
127         BTserial.println("AT+NAME=BT05");//Set the radio called Robot
128         delay(200);
129         BTserial.println("AT+ROLE=0");
130         delay(200);
```

```
131 BTserial.println("AT+UART=2");//1=2400 2=9600 3=19200 4=115200
132 delay(200);
133
134 //SoftwareSerial does not support dynamic adjustment of baud rate, so close the serial
135 BTserial.end();
136 }
137
138 BTserial.begin(9600);
139 delay(1500);
140 Buzzer_Setup(); //Initialize the buzzer
141 EEPROM.begin(512);
142 Bipedal_Robot.init(LeftLeg, RightLeg, LeftFoot, RightFoot, true); //Set the servo pins and
143 Buzzer pin
144
145 YL = EEPROM.read(0);
146 if (YL > 128) YL -= 256;
147 YR = EEPROM.read(1);
148 if (YR > 128) YR -= 256;
149 RL = EEPROM.read(2);
150 if (RL > 128) RL -= 256;
151 RR = EEPROM.read(3);
152 if (RR > 128) RR -= 256;
153 Bipedal_Robot.setTrims(YL, YR, RL, RR);
154 calib_homePos();
155 Bipedal_Robot.saveTrimsOnEEPROM();
156 Bipedal_Robot.home();
157 delay(100);
158 EEPROM.commit();
159 SCmd.addCommand("S", receiveStop);
160 SCmd.addCommand("D", recieveBuzzer);
161 SCmd.addCommand("A", receiveMovement);
162
163 inputString.reserve(200);
164 SCmd.addDefaultHandler(receiveStop);
165 Serial.println("Set the name of the Bluetooth module to BT05.");
166 Serial.println("Set the Bluetooth baud rate to 9600.");
167 Bipedal_Robot.home();
168 }
169
170 void loop() {
171 SCmd.readSerial();
172 move(moveId);
173 }
```

```

175 void calib_homePos() {
176     int servoPos[4];
177     servoPos[0] = 90;
178     servoPos[1] = 90;
179     servoPos[2] = 90;
180     servoPos[3] = 90;
181     Bipedal_Robot._moveServos(500, servoPos);
182     Bipedal_Robot.detachServos();
183 }
```

## Code Explanation

Add the header files to drive the robot.

```

1 #include <Arduino.h>
2 #include <SoftwareSerial.h>
3 #include <EEPROM.h>
4 #include "Freenove_Robot_For_Pico_W.h"
5
6 #include "SerialCommand.h"
```

Upon receiving movement commands, the robot makes corresponding actions.

```

87 void move(int moveId) {
88     // bool manualMode = false;
89     switch (moveId) {
90         case 0:
91             Bipedal_Robot.home();
92             // Serial.print("moveId 0 home");
93             break;
94         case 1:
95             Bipedal_Robot.walk(1, T, 1);
96             break;
97         case 2:
98             Bipedal_Robot.walk(1, T, -1);
99             break;
100        case 3:
101            Bipedal_Robot.turn(1, T, 1);
102            break;
103        case 4:
104            Bipedal_Robot.turn(1, T, -1);
105            break;
106        default:
107            break;
108    }
109 }
```

## 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.

### Sketch

Open the folder “05.3\_Multi\_Functional\_Robot” in the “**Freenove\_Bipedal\_Robot\_Kit\_for\_Raspberry\_Pi\_Pico\Sketches**” and double click “05.3\_Multi\_Functional\_Robot.ino”.

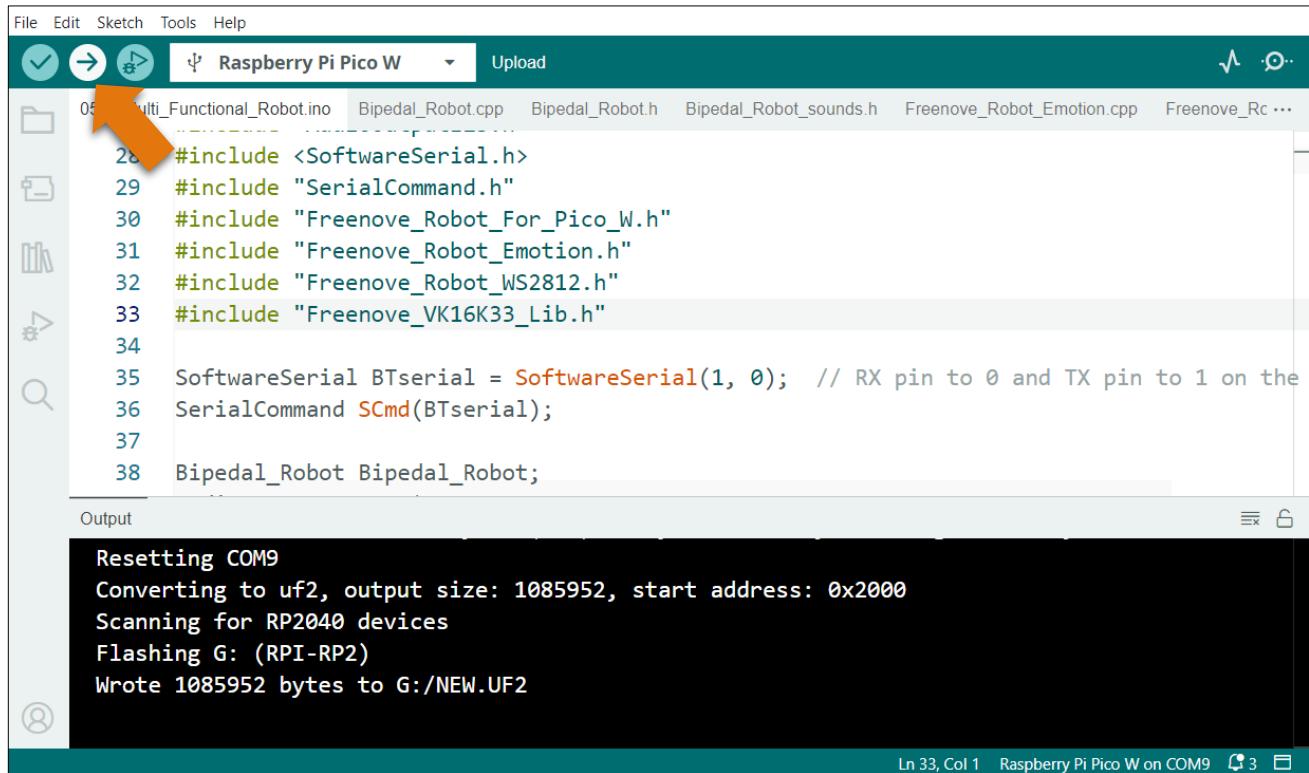
Freenove_Bipedal_Robot_Kit_for_Raspberry_Pi_Pico-main > Sketches >			
Name	Date modified	Type	Size
02.2_Ultrasonic_Ranging_Robot	11/14/2024 10:51 AM	File folder	
03.1_Sing_and_dance	5/7/2025 5:41 PM	File folder	
04.1_IR_Receiver	11/14/2024 10:51 AM	File folder	
04.2_IR_Receiver_Robot	11/14/2024 10:51 AM	File folder	
04.3_Multi_Functional_Robot	11/14/2024 10:51 AM	File folder	
04.4_Multi_Functional_Robot	11/14/2024 10:51 AM	File folder	
05.1_Bluetooth_Robot	11/14/2024 10:51 AM	File folder	
05.2_Bluetooth_Remote_Robot	11/14/2024 10:51 AM	File folder	
05.3_Multi_Functional_Robot	11/14/2024 10:51 AM	File folder	

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



The screenshot shows the Arduino IDE interface. The top menu bar includes File, Edit, Sketch, Tools, Help, and a dropdown for "Raspberry Pi Pico W". Below the menu is a toolbar with icons for file operations. The main area displays a code editor with the following content:

```

File Edit Sketch Tools Help
Raspberry Pi Pico W Upload
05_Multi_Functional_Robot.ino Bipedal_Robot.cpp Bipedal_Robot.h Bipedal_Robot_sounds.h Freenove_Robot_Emotion.cpp Freenove_Rc ...
28 #include <SoftwareSerial.h>
29 #include "SerialCommand.h"
30 #include "Freenove_Robot_For_Pico_W.h"
31 #include "Freenove_Robot_Emotion.h"
32 #include "Freenove_Robot_WS2812.h"
33 #include "Freenove_VK16K33_Lib.h"
34
35 SoftwareSerial BTserial = SoftwareSerial(1, 0); // RX pin to 0 and TX pin to 1 on the
36 SerialCommand SCmd(BTserial);
37
38 Bipedal_Robot Bipedal_Robot;

```

The output window below shows the upload progress:

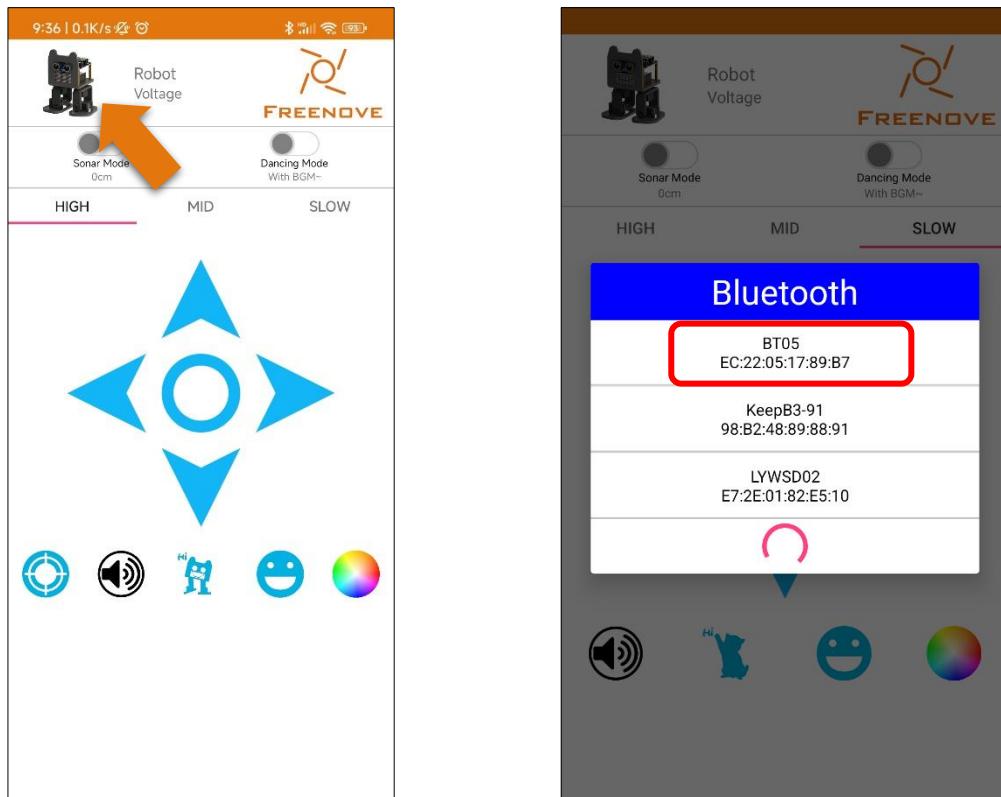
```

Resetting COM9
Converting to uf2, output size: 1085952, start address: 0x2000
Scanning for RP2040 devices
Flashing G: (RPI-RP2)
Wrote 1085952 bytes to G:/NEW.UF2

```

At the bottom right, it says "Ln 33, Col 1 Raspberry Pi Pico W on COM9".

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 the following icon and select 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 turns off all LED displays.

Mode 1 displays the selected color of the current color picker for all LEDs.

Mode 2 is a flowing water LED with color changing.

Mode 3 is a color-adjustable Blink.

Mode 4 is a breathing light effect with adjustable color.

Mode 5 is a flowing rainbow.

## Code

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

```
1 #include <Arduino.h>
2 #include <EEPROM.h>
3 #include "Bipedal_Robot.h"
4 #include "AudioFileSourceLittleFS.h"
5 #include "AudioGeneratorMP3.h"
6 #include "AudioOutputI2SNoDAC.h"
7 #include "AudioFileSourceID3.h"
8 #include "AudioOutputI2S.h"
9 #include <SoftwareSerial.h>
10 #include "SerialCommand.h"
11 #include "Freenove_Robot_For_Pico_W.h"
12 #include "Freenove_Robot_Emotion.h"
13 #include "Freenove_Robot_WS2812.h"
14 #include "Freenove_VK16K33_Lib.h"
15
16 SoftwareSerial BTserial = SoftwareSerial(1, 0); // RX pin to 0 and TX pin to 1 on the board
17 SerialCommand SCmd(BTserial);
18
19 Bipedal_Robot Bipedal_Robot;
20 AudioGeneratorMP3 *mp3;
21 AudioFileSourceLittleFS *file;
22 AudioOutputI2SNoDAC *out;
23 AudioFileSourceID3 *id3;
24
25 #define LeftLegPin 10
26 #define RightLegPin 12
27 #define LeftFootPin 11
28 #define RightFootPin 13
29 #define BuzzerPin 20
30
31 int motionFlag = 0;
32 int ultrasonicavoidMode = 0;
```

```
33 int playstartingmusic = 0;
34 int musicmode = 0;
35 int playsong = 0;
36 extern int playsong; //Set the proportional coefficient
37 int T = 1000;
38 int moveId = 0;
39 int dance_steps = 0;
40 int Ultrasonic_Avoid_steps = 0;
41 bool isServoResting = true;
42 extern int moveId;
43 extern bool isServoResting;
44 int moveSize = 15;
45
46 String inputString = ""; // a String to hold incoming data
47 bool stringComplete = false; // whether the string is complete
48 int baudRate[] = {2400, 9600, 19200, 115200};
49 int baudRateCount = sizeof(baudRate) / sizeof(baudRate[0]);
50
51 #define ACTION_GET_VOLTAGE 'P'
52 #define ACTION_GET_DISTANCE 'E'
53 #define INTERVAL_CHAR '#'
54 unsigned long lastUploadVoltageTime;
55 unsigned long lastUploadDistanceTime;
56 #define UPLOAD_VOL_TIME 2000
57
58 int Resting = 0;
59 float Distance = 0;
60
61 // int calibratePosition[4] = { -17, -10, -15, 12 };
62 // int YL;
63 // int YR;
64 // int RL;
65 // int RR;
66
67 void upLoadVoltageToApp() {
68     float voltage = 0;
69     voltage = Get_Battery_Voltage();
70     BTserial.print("P#");
71     BTserial.println(int(voltage * 1000));
72 }
73
74 void upLoadDistanceToApp() {
75     Distance = Get_Sonar();
76     BTserial.print("E#");
```

```
77     BTserial.println(Distance);
78 }
79
80 void receiveStop() {
81     sendAck();
82     sendFinalAck();
83 }
84
85 void receiveBuzzer() {
86     sendAck();
87     moveId = 0;
88     playsong = 0;
89     bool error = false;
90     int freq;
91     char *arg;
92     arg = SCmd.next();
93     if (arg != NULL) freq = atoi(arg);
94     else error = true;
95     tone(2, freq);
96     sendFinalAck();
97 }
98
99 void receiveAvoid() {
100    sendAck();
101    ws2812_task_mode = 0;
102    emotion_task_mode = 0;
103    if (Bipedal_Robot.getRestState() == true) Bipedal_Robot.setRestState(false);
104    isServoResting = Bipedal_Robot.getRestState();
105    char *arg;
106    arg = SCmd.next();
107    if (arg != NULL) {
108        ultrasonicavoidMode = atoi(arg);
109        playsong = 0;
110        if (ultrasonicavoidMode == 1) {
111            moveId = 21;
112            Ultrasonic_Avoid_steps = 0;
113            playsong = 0;
114        }
115        if (ultrasonicavoidMode == 2) {
116            dance_steps = 0;
117            moveId = 22;
118            playsong = 4;
119        }
120        if (ultrasonicavoidMode == 0) {
```

```
121     moveId = 0;
122     Resting = 1;
123     playsong = 0;
124 }
125 }
126 sendFinalAck();
127 }
128
129 void receiveEmotion() {
130     sendAck();
131     Emotion_Setup();
132     moveId = 0;
133     playsong = 0;
134     char *arg;
135     char *endstr;
136     arg = SCmd.next();
137     if (arg != NULL) {
138         String stringOne = String(arg); // converting a constant char into a String
139         emotion_task_mode = stringOne.toInt();
140         Emotion_SetMode(emotion_task_mode); //显示静态表情
141     } else {
142     }
143     sendFinalAck();
144 }
145
146 void receiveLED() {
147     sendAck();
148     moveId = 0;
149     playsong = 0;
150     char *arg;
151     unsigned char paramters[3];
152     char *endstr;
153     arg = SCmd.next();
154     if (arg != NULL) {
155         String stringOne = String(arg); // converting a constant char into a String
156         ws2812_task_mode = stringOne.toInt();
157     } else {
158     }
159     arg = SCmd.next();
160     String stringOne1 = String(arg);
161     paramters[0] = stringOne1.toInt();
162
163     arg = SCmd.next();
164     String stringOne2 = String(arg);
```

```
165 paramters[1] = stringOne2.toInt();
166
167 arg = SCmd.next();
168 String stringOne3 = String(arg);
169 paramters[2] = stringOne3.toInt();
170
171 WS2812_Set_Color_1(15, paramters[0], paramters[1], paramters[2]);
172 sendFinalAck();
173 }
174
175 void songmusic() {
176 if (playsong > 0) {
177 playmusic(1, playsong);
178 if (playsong != 4) {
179 playsong = 0;
180 }
181 } else {
182 pinMode(6, OUTPUT);
183 digitalWrite(6, LOW);
184 }
185 }
186 void receiveMusic() {
187 int music_song;
188 sendAck();
189 ws2812_task_mode = 0;
190 emotion_task_mode = 0;
191 if (Bipedal_Robot.getRestState() == true) Bipedal_Robot.setRestState(false);
192 isServoResting = Bipedal_Robot.getRestState();
193 char *arg;
194 arg = SCmd.next();
195 if (arg != NULL) {
196 music_song = atoi(arg);
197 playsong = music_song;
198 playstartingmusic = music_song;
199 clearEmtions();
200 if (playsong == 1) {
201 moveId = 18; //play music Hello.mp3
202 } else if (playsong == 2) {
203 moveId = 19; //play music Nicetomeetyou.mp3
204 } else if (playsong == 3) {
205 moveId = 20; //play music goodbye.mp3
206 Serial.print("yundong");
207 } else if (playsong == 0) {
208 moveId = 0;
```

```
209     }
210     Serial.print("moveId:");
211     Serial.print(moveId);
212 } else {
213     moveId = 0;
214     Resting = 1;
215 }
216 }

217

218 void playmusic(int playtimes, int song) {
219 if (playtimes--) {
220     if (song == 1) {
221         file = new AudioFileSourceLittleFS("Hello.mp3");
222     } else if (song == 2) {
223         file = new AudioFileSourceLittleFS("Nicetomeetyou.mp3");
224     } else if (song == 3) {
225         file = new AudioFileSourceLittleFS("goodbye.mp3");
226     } else if (song == 4) {
227         file = new AudioFileSourceLittleFS("1.mp3");
228     } else if (song == 5) {
229         file = new AudioFileSourceLittleFS("onfoot.mp3");
230     } else if (song == 6) {
231         file = new AudioFileSourceLittleFS("hello_cn.mp3");
232     } else if (song == 0) {
233         file = new AudioFileSourceLittleFS("split.mp3");
234     }
235     out = new AudioOutputI2SNoDAC(6);
236     out->SetGain(4.0); //Volume Setup (0-4.0)
237     mp3 = new AudioGeneratorMP3();
238     mp3->begin(file, out);
239     if (mp3->isRunning() && song > 0) {
240         if (!mp3->loop() && song > 0) {
241             delete file;
242             delete mp3;
243             out->flush();
244             out->stop();
245         }
246     } else {
247         delete file;
248         delete mp3;
249         out->flush();
250         out->stop();
251         pinMode(6, OUTPUT);
252         digitalWrite(6, LOW);
```

```
253     }
254   }
255 }
256
257 //Ultrasonic robot
258 void Ultrasonic_Avoid() {
259   if (Distance <= 15) {
260     Bipedal_Robot.walk(2, 1000, -1); // BACKWARD x2
261     Bipedal_Robot.turn(3, 1000, -1); // LEFT x3
262   }
263   Bipedal_Robot.walk(1, 1500, 1); // FORWARD x1
264 }
265
266 void dance() {
267   dance_steps = dance_steps + 1;
268   staticEmtions(21);
269   if (moveId > 0) {
270     switch (dance_steps) {
271       case 0:
272         dance_steps = 0;
273         break;
274       case 1:
275         Bipedal_Robot.jitter(1, 750, 20);
276         break;
277       case 2:
278         Bipedal_Robot.jitter(1, 750, 20);
279         break;
280       case 3:
281         Bipedal_Robot.crusaito(1, 800, 30, 1);
282         break;
283       case 4:
284         Bipedal_Robot.crusaito(1, 800, 30, -1);
285         break;
286       case 5:
287         Bipedal_Robot.crusaito(1, 800, 30, 1);
288         Bipedal_Robot.home();
289         delay(300);
290         break;
291       case 6:
292         Bipedal_Robot.walk(1, 1500, -1);
293         break;
294       case 7:
295         Bipedal_Robot.walk(1, 1000, 1);
296         break;
```

```
297     case 8:
298         Bipedal_Robot.walk(1, 1000, 1);
299         Bipedal_Robot.home();
300         break;
301     case 9:
302         Bipedal_Robot.moonwalker(1, 600, 30, 1);
303         break;
304     case 10:
305         Bipedal_Robot.moonwalker(1, 600, 30, -1);
306         break;
307     case 11:
308         Bipedal_Robot.moonwalker(1, 600, 30, 1);
309         break;
310     case 12:
311         Bipedal_Robot.moonwalker(1, 600, 30, -1);
312         Bipedal_Robot.home();
313         delay(100);
314         break;
315     case 13:
316         Bipedal_Robot.walk(1, 1500, -1);
317         Bipedal_Robot.home();
318         delay(100);
319         break;
320     }
321     if (dance_steps > 14) {
322         dance_steps = 0;
323     }
324 } else {
325     Resting = 1;
326     moveId = 0;
327     dance_steps = 0;
328 }
329 }

330

331 void receiveMovement() {
332     sendAck();
333     ws2812_task_mode = 0;
334     emotion_task_mode = 0;
335     if (Bipedal_Robot.getRestState() == true) Bipedal_Robot.setRestState(false);
336     isServoResting = Bipedal_Robot.getRestState();
337     char *arg;
338     arg = SCmd.next();
339     if (arg != NULL) {
340         moveId = atoi(arg);
```

```
341     if (moveId == 0) {
342         Resting = 1;
343     }
344 } else {
345     moveId = 0;
346 }
347 arg = SCmd.next();
348 if (arg != NULL) {
349     T = atoi(arg);
350     if (T == 0) {
351         T = 1000;
352     } else if (T == 1) {
353         T = 1500;
354     } else if (T == 2) {
355         T = 2000;
356     }
357 } else T = 2000;
358 Serial.print("T is:");
359 Serial.print(T);
360 arg = SCmd.next();
361 if (arg != NULL) moveSize = atoi(arg);
362 else moveSize = 15;
363 }

364

365 void move(int movemode) {
366 // bool manualMode = false;
367 switch (movemode) {
368     case 0:
369         Bipedal_Robot.home();
370         if (Resting == 1) {
371             Bipedal_Robot.setRestState(false);
372             Bipedal_Robot.home();
373             Resting = 0;
374         }
375         break;
376     case 1:
377         staticEmtions(21);
378         Bipedal_Robot.walk(1, T, 1);
379         break;
380     case 2:
381         staticEmtions(21);
382         Bipedal_Robot.walk(1, T, -1);
383         break;
384     case 3:
```

```
385     staticEmtions(21);
386     Bipedal_Robot.turn(1, T, 1);
387     break;
388 case 4:
389     staticEmtions(21);
390     Bipedal_Robot.turn(1, T, -1);
391     break;
392 case 5:
393     Bipedal_Robot.updown(1, 2000, 30);
394     break;
395 case 6: Bipedal_Robot.moonwalker(1, T, moveSize, 1); break;
396 case 7: Bipedal_Robot.moonwalker(1, T, moveSize, -1); break;
397 case 8: Bipedal_Robot.swing(1, T, moveSize); break;
398 case 9: Bipedal_Robot.crusaito(1, T, moveSize, 1); break;
399 case 10: Bipedal_Robot.crusaito(1, T, moveSize, -1); break;
400 case 11: Bipedal_Robot.jump(1, T); break;
401 case 12: Bipedal_Robot.flapping(1, T, moveSize, 1); break;
402 case 13: Bipedal_Robot.flapping(1, T, moveSize, -1); break;
403 case 14: Bipedal_Robot.tiptoeSwing(1, T, moveSize); break;
404 case 15: Bipedal_Robot.bend(1, T, 1); break;
405 case 16: Bipedal_Robot.bend(1, T, -1); break;
406 case 17: Bipedal_Robot.shakeLeg(1, T, 1); break;
407 case 18:
408     staticEmtions(21);
409     Bipedal_Robot.swing(1, 1100, 50);
410     Bipedal_Robot.home();
411     Resting = 1;
412     moveId = 0;
413     clearEmtions();
414     break;
415 case 19:
416     staticEmtions(22);
417     Bipedal_Robot.moonwalker(1, 1550, 50, 1); //1541
418     Bipedal_Robot.home();
419     Resting = 1;
420     moveId = 0;
421     clearEmtions();
422     break;
423 case 20:
424     staticEmtions(23);
425     Bipedal_RobotascendingTurn(2, 700, 12); //1332
426     clearEmtions();
427     delay(100);
428     Bipedal_Robot.home();
```

```
429     Resting = 1;
430     moveId = 0;
431     break;
432 case 21:
433     Ultrasonic_Avoid();
434     break;
435 case 22:
436     dance();
437     break;
438 default:
439     break;
440 }
441 }
442
443 void sendAck() {
444     delay(5);
445     Serial.flush();
446 }
447 void sendFinalAck() {
448     delay(5);
449     Serial.flush();
450 }
451
452 void calib_homePos() {
453     int servoPos[4];
454     servoPos[0] = 90;
455     servoPos[1] = 90;
456     servoPos[2] = 90;
457     servoPos[3] = 90;
458     Bipedal_Robot._moveServos(500, servoPos);
459     Bipedal_Robot.detachServos();
460 }
461
462 void setup() {
463     Serial.begin(9600);
464     delay(2000);
465     // Loop through each baud rate and send commands
466     for(int i = 0; i < baudRateCount; i++) {
467         BTserial.begin(baudRate[i]);
468         delay(200);
469         BTserial.println("AT+NAME=BT05");//Set the radio called Robot
470         delay(200);
471         BTserial.println("AT+ROLE=0");
472         delay(200);
```

```
473 BTserial.println("AT+UART=2");//1=2400 2=9600 3=19200 4=115200
474 delay(200);
475
476 //SoftwareSerial does not support dynamic adjustment of baud rate, so close the serial
477 BTserial.end();
478 }
479
480 BTserial.begin(9600);
481 delay(1500);
482 EEPROM.begin(512); //Initialize
483 the ultrasonic module
484 Bipedal_Robot.init(LeftLegPin, RightLegPin, LeftFootPin, RightFootPin, true); //Set the
485 servo pins
486 Bipedal_Robot.Buzzer_init(BuzzerPin);
487
488 Bipedal_Robot.saveTrimsOnEEPROM();
489 Bipedal_Robot.home();
490 delay(50);
491 Buzzer_Setup(); //Initialize the buzzer
492 WS2812_Setup(); //WS2812 initialization
493 Emotion_Setup();
494 Ultrasonic_Setup();
495
496 SCmd.addCommand("S", receiveStop);
497 SCmd.addCommand("C", receiveLED);
498 SCmd.addCommand("D", recieveBuzzer);
499 SCmd.addCommand("A", receiveMovement);
500 SCmd.addCommand("B", receiveEmotion);
501 SCmd.addCommand("H", receiveAvoid);
502 SCmd.addCommand("M", receiveMusic);
503
504 inputString.reserve(200);
505 SCmd.addHandler(receiveStop);
506 Serial.println("Set the name of the Bluetooth module to BT05.");
507 Serial.println("Set the Bluetooth baud rate to 9600.");
508 }
509
510 void loop() {
511 SCmd.readSerial();
512 Emotion_Detection();
513 WS2812_Show(ws2812_task_mode); //Car color lights display function
514 if (millis() - lastUploadVoltageTime > UPLOAD_VOL_TIME) {
515 upLoadVoltageToApp();
516 lastUploadVoltageTime = millis();
```

```

517 }
518 if (millis() - lastUploadDistanceTime > 200) {
519   if (ultrasonicavoidMode == 1) {
520     upLoadDistanceToApp();
521   }
522   lastUploadDistanceTime = millis();
523 }
524 if (Check_Module_value == MATRIX_IS_EXIST) {
525   Emotion_Show(emotion_task_mode); //Led matrix display function
526 }
527 move(moveId);
528 }

529
530 void setup1() {
531   delay(2000);
532   playmusic(1, 2);
533   pinMode(6, OUTPUT);
534   digitalWrite(6, LOW);
535 }
536 void loop1() {
537   songmusic();
538 }

```

## Code Explanation

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

```

510 void loop() {
511   SCmd.readSerial();
512   Emotion_Detection();
513   WS2812_Show(ws2812_task_mode); //Car color lights display function
514   if (millis() - lastUploadVoltageTime > UPLOAD_VOL_TIME) {
515     upLoadVoltageToApp();
516     lastUploadVoltageTime = millis();
517   }
518   if (millis() - lastUploadDistanceTime > 200) {
519     if (ultrasonicavoidMode == 1) {
520       upLoadDistanceToApp();
521     }
522     lastUploadDistanceTime = millis();
523   }
524   if (Check_Module_value == MATRIX_IS_EXIST) {
525     Emotion_Show(emotion_task_mode); //Led matrix display function
526   }
527   move(moveId);

```

528

}

## Bluetooth command

The description of the commands and their functions are as shown 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
M#parm1	Audio playing command
H#parm1	Ultrasonic obstacle avoidance mode or dance mode
Send battery data every 25s	Send the battery level to the APP. Format: P#Battery voltage
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.