

Important Information

Thank you for choosing Freenove products!

Getting Started

If you have not yet downloaded the zip file, associated with this kit, please do so now and unzip it.

https://github.com/Freenove/Freenove_Robot_Ant_Kit

Get Support and Offer Input

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

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

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

support@freenove.com

Safety and Precautions

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

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

Any concerns?  support@freenove.com



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 and micro:bit
- 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>

sale@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.



Contents

Important Information.....	1
Contents	1
Chapter 0 Software Installation and Assembly	1
01 Installation of CH340	1
02 Installation of Arduino IDE.....	10
03 Installation of Libraries	13
04 Control board.....	16
05 Component List	18
06 Assembly.....	21
07 Test	32
Chapter1 Servo Test(Importance)	34
Component Knowledge	34
Circuit	34
Sketch.....	35
Chapter2 WS2812 Module Test	40
Component Knowledge	40
Circuit	41
Sketch.....	43
Chapter3 Ultrasonic Module Test.....	45
Component Knowledge	45
Circuit	46
Sketch.....	47
Chapter4 Expression module Test.....	49
Component Knowledge	49
Circuit	51
Sketch.....	52
Chapter5 ADC and Buzzer Test.....	55
Component Knowledge	55
Circuit	57
Sketch.....	58
Chapter6 Infrared Sensor Test.....	61
Component Knowledge	61

Any concerns? ✉ support@freenove.com

Circuit.....	62
Sketch.....	63
Chapter7 Infrared Controlled Robot	66
Sketch.....	66
Chapter8 Obstacle Avoiding Robot Ant	69
Sketch.....	69
Chapter9 Bluetooth Controlled Robot Ant	71
Sketch.....	74
Chapter10 Integrated Routine.....	78
Sketch.....	78
What's next?.....	87

Chapter 0 Software Installation and Assembly

If you haven't downloaded the sources for the robot ant, you can download them via the link below:

https://github.com/Freenove/Freenove_Robot_Ant_Kit

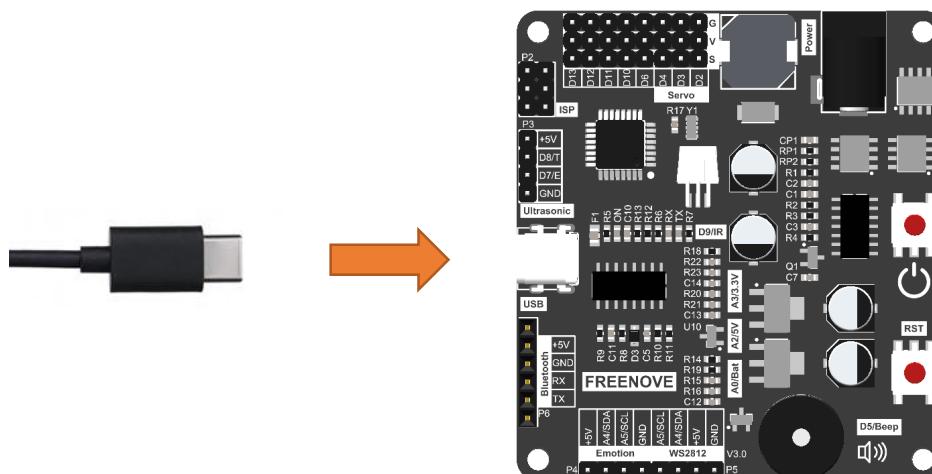
01 Installation of CH340

We upload code to the control board through CH340. Therefore, we need to install the driver for CH340 first.

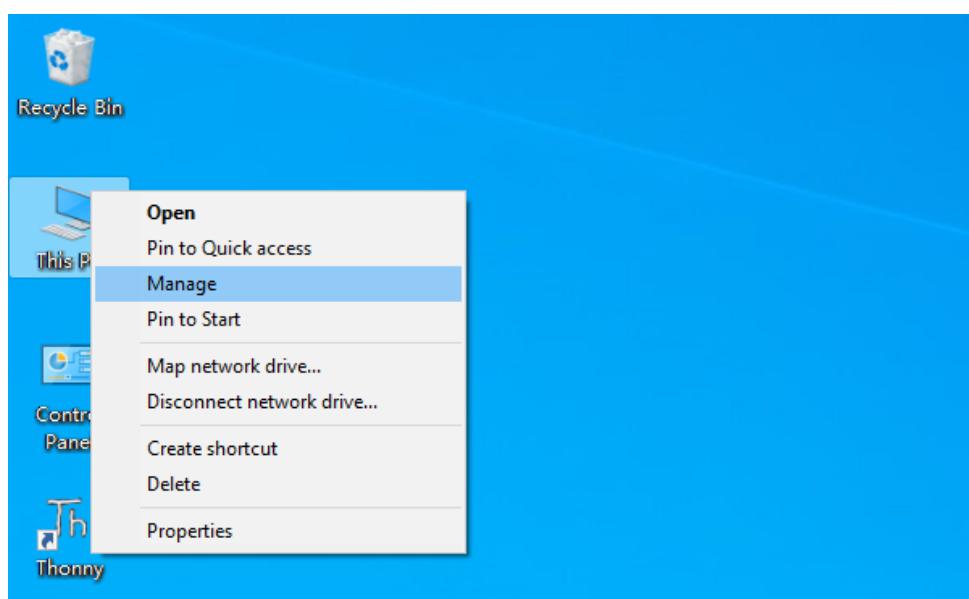
Windows

Check whether CH340 has been installed to your computer.

1. Connect the control board with a Type-C cable.



2. Back to the computer homepage, right-click on "This PC" and select "Manage".



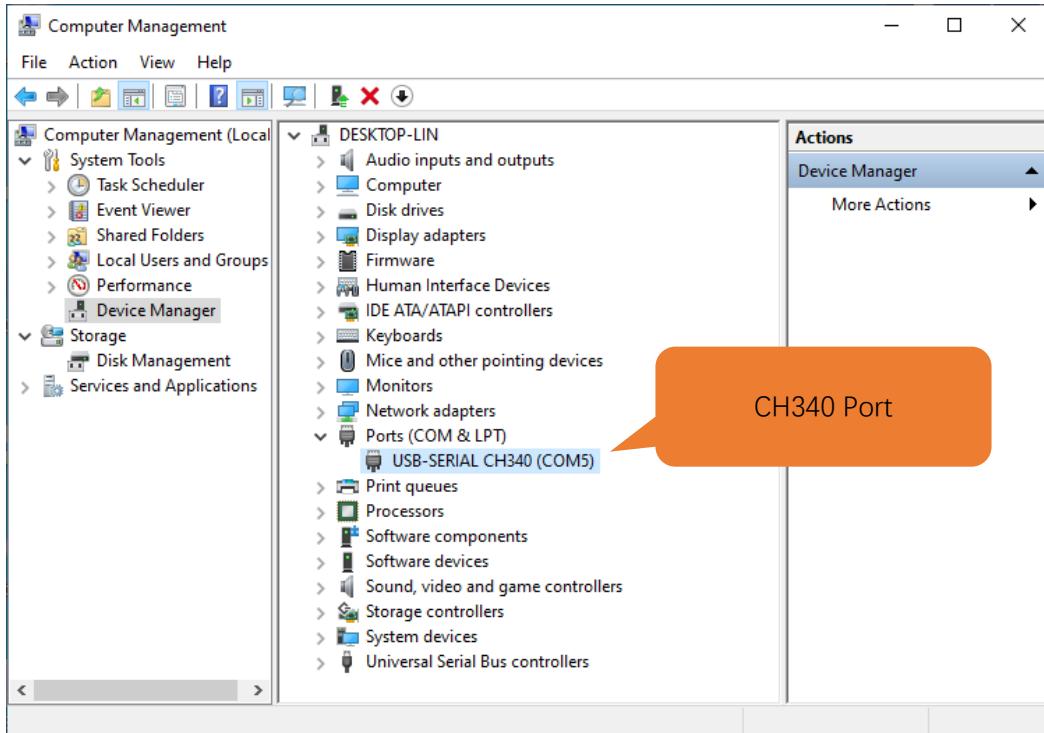
Any concerns? ✉ support@freenove.com

3. Click “Device Manage” on the pop-up window. If CH340 has been installed to your computer, you can see “USB-SERIAL CH340 (COMx)”.

Note: Only when the control board has connected with computer will “USB-SERIAL CH340 (COMx)” show up.

If you have installed CH340, you can skip to [next step](#).

If you haven't, please install it as follows.



Install CH340 Driver

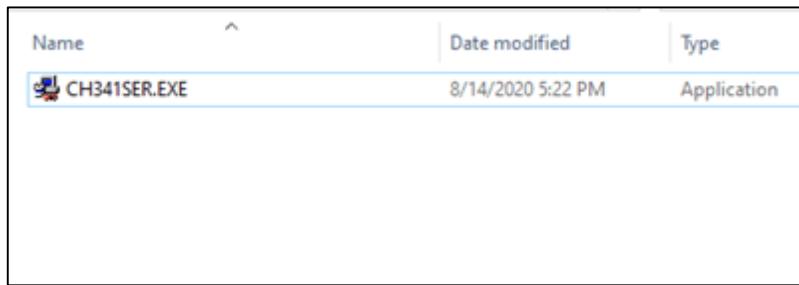
- Click <http://www.wch-ic.com/search?q=CH340&t=downloads> to go to the downloading website of CH340. Choose a proper CH340 driver based on your operating system.

file category	file content	version	upload time
Driver&Tools	Windows		
CH341SER.EXE	CH340/CH341 USB to serial port Windows driver, supports 32/64-bit Windows 10/8.1/8/7/VISTA/XP, Server 2016/2012/2008/2003, 2000/ME/98	3.5	2019-03-18
CH341SER.ZIP	CH340/CH341 USB to serial port Windows driver, includes DLL dynamic library and non-standard baud rate settings and other instructions. Supports 32/64-bit Windows 10/8.1/8/7/VISTA/XP, Server 2016/2012/2008/2003, 2000/ME/98	3.5	2019-03-05
CH341SER_ANDROID...	CH340/CH341 USB to serial port Android free drive application library, for Android OS 3.1 and above version which supports USB Host mode already, no need to load Android kernel driver, no root privileges. Contains apk, lib library file (linux Driver), App Demo Example (USB to UART Demo)	1.6	2019-04-19
CH341SER_LINUX...	CH340/CH341 USB to serial port LINUX driver	1.5	2018-03-18
CH341SER_MAC.ZI...	CH340/CH341 USB to serial port MAC OS driver	1.5	2018-07-05
Others			
PRODUCT_GUIDE.P...	Electronic selection of product selection manual, please refer to related product technical manual for more technical information.	1.4	2018-12-29
InstallNoteOn64...	Instructions for the driver after 18 years of August cannot be installed under some 64-bit WIN7 (English)	1.0	2019-01-10

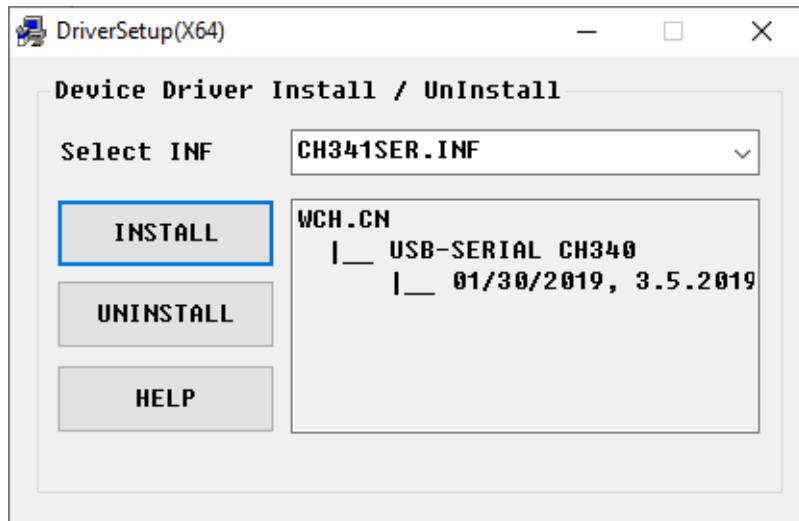
If you don't feel like downloading CH340 driver from the official website, you can also open the folder "Freenove_Robot_Ant_Kit/CH340". We've prepared it for you in advance.

Name	Date modified	Type	Size
Linux	8/14/2020 5:24 PM	File folder	
MAC	8/14/2020 5:23 PM	File folder	
Windows	8/14/2020 5:23 PM	File folder	

2. Open the folder “Freenove_Robot_Ant_Kit/CH340/Windows/”.

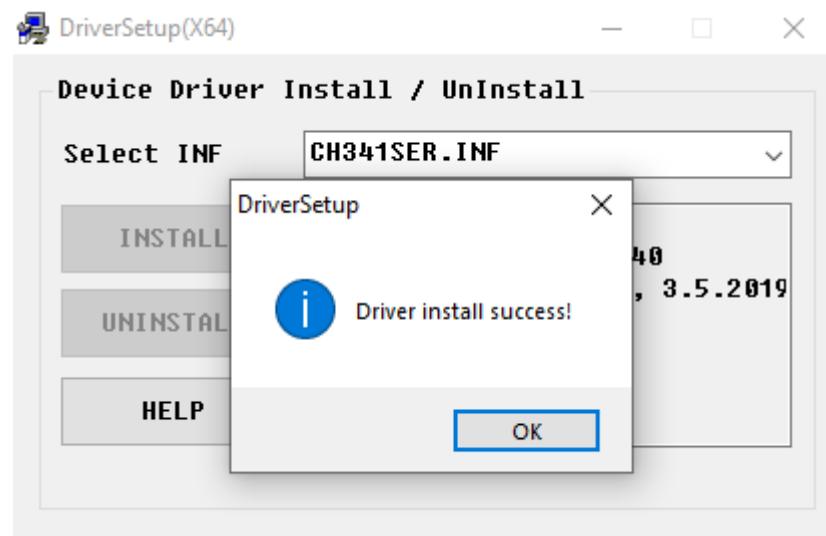


3. Double click to open “CH341SER.EXE”.

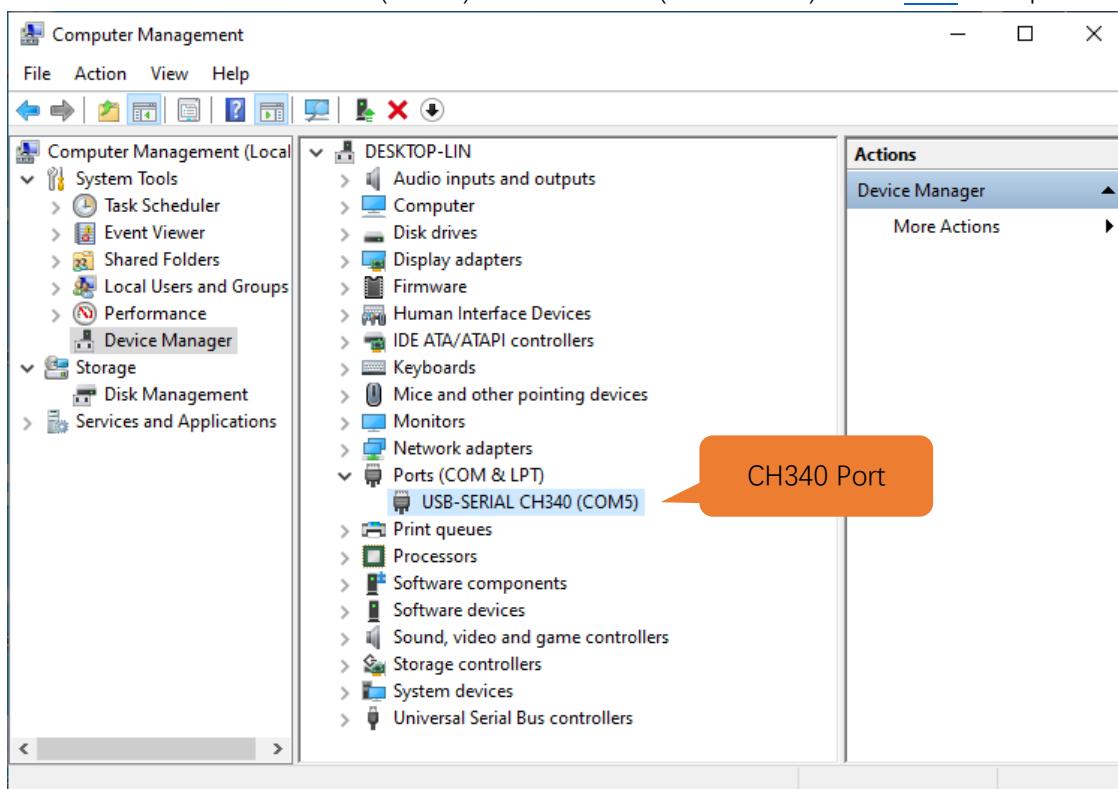


4. Click “INSTALL” and wait for the installation to finish.

5. When the following interface appears, it means the installation is successful. You can close it.



6. Connect the control board and computer with a type-C cable, when clicking "Device Manager" again, you will see "USB-SERIAL CH340 (COMx)" under "Ports (COM & LPT)". Click [here](#) to skip to next step.



7. So far, CH340 has been installed. You can close all windows.

Mac

- Click <http://www.wch-ic.com/search?q=CH340&t=downloads> to go to the downloading website of CH340. Choose a proper CH340 driver based on your operating system.

The screenshot shows a search results page for 'ch340' on the WCH website. The left sidebar has categories: All (14), Downloads (7), Products (4), Application (2), Video (1), and News (0). The main area is titled 'keyword ch340' and shows 'Downloads(7)'. It lists files categorized by file content: Driver&Tools. The files are:

- Windows**: CH341SER.EXE (Windows driver, supports 32/64-bit Windows 10/8.1/8/7/VISTA/XP, Server 2016/2012/2008/2003, 2000/ME/98) - Version 3.5, upload time 2019-03-18
- CH341SER.ZIP (CH340/CH341 USB to serial port Windows driver, includes DLL dynamic library and non-standard baud rate settings and other instructions. Supports 32/64-bit Windows 10/8.1/8/7/VISTA/XP, Server 2016/2012/2008/2003, 2000/ME/98) - Version 3.5, upload time 2019-03-05
- CH341SER_ANDROID... (CH340/CH341 USB to serial port Android free drive application library, for Android OS 3.1 and above version which supports USB Host mode already, no need to load Android kernel driver, no root privileges. Contains apk, lib library file (Java Driver), App Demo Examples, and Demo SDK) - Version 1.6, upload time 2019-04-19
- Linux**: CH341SER_LINUX... (CH340/CH341 USB to serial port LINUX driver) - Version 1.5, upload time 2018-03-18
- MAC**: CH341SER_MAC.ZIP (CH340/CH341 USB to serial port MAC OS driver) - Version 1.5, upload time 2018-07-05

 Other files listed include CH340SER_MAC.DLL and CH340SER_MAC.DRIVER.

If you don't feel like downloading CH340 driver from the official website, you can also open the folder "Freenove_Robot_Ant_Kit/CH340". We've prepared it for you in advance.

- Open "Freenove_Ant_Kit/CH340/MAC/".

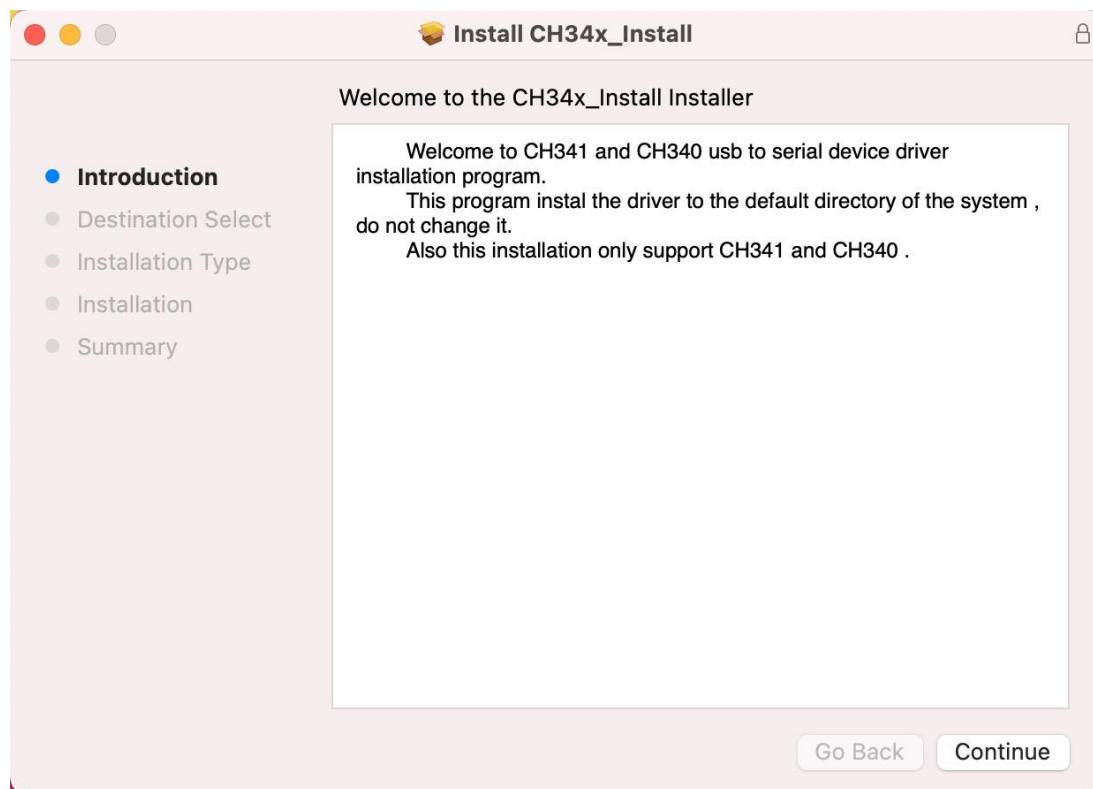
The screenshot shows a Mac Finder window titled 'MAC'. The sidebar on the left lists Favorites (AirDrop, Recents, Applications, Desktop, Documents, Downloads), iCloud (iCloud Drive), Tags, Locations, and Network. The main pane shows two files in the 'Name' column:

- CH34x_Install_V1.5.pkg
- ReadMe.pdf

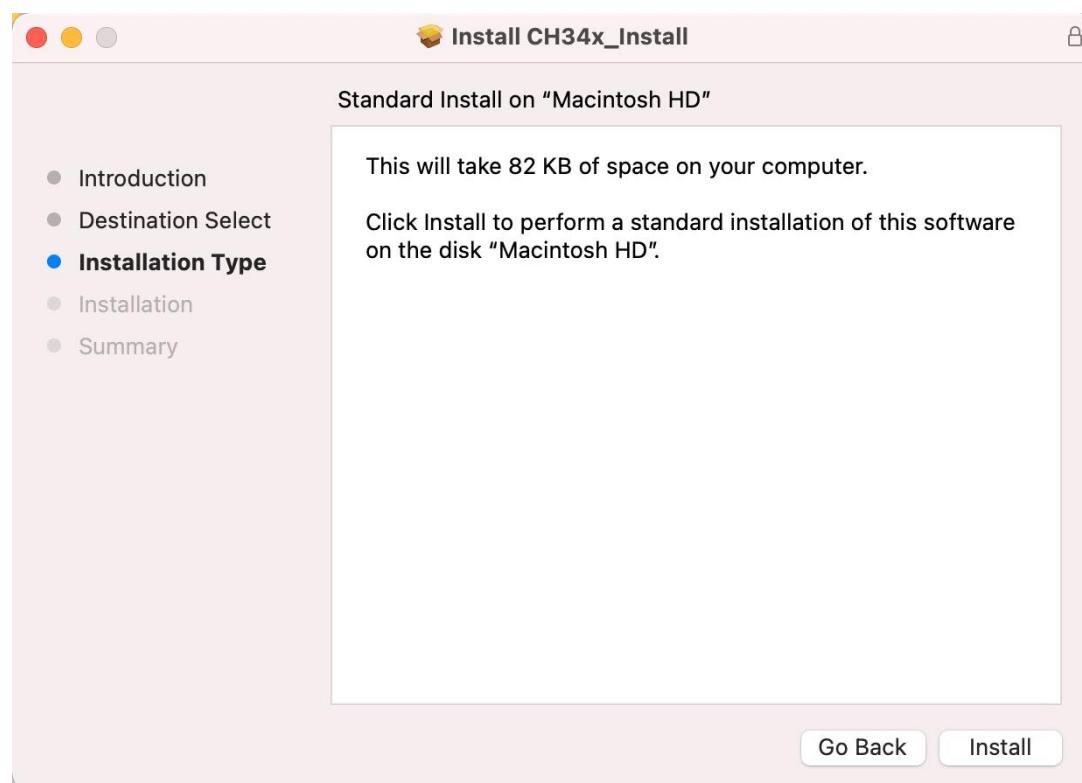
 The 'Run it' button is highlighted with an orange callout bubble. The table headers are Name, Modified, Size, and Kind.

Name	Modified	Size	Kind
CH34x_Install_V1.5.pkg	Oct 1, 2018 at 2:00 PM	26 KB	Installer package
ReadMe.pdf	Dec 8, 2016 at 4:09 PM	146 KB	Adobe...document

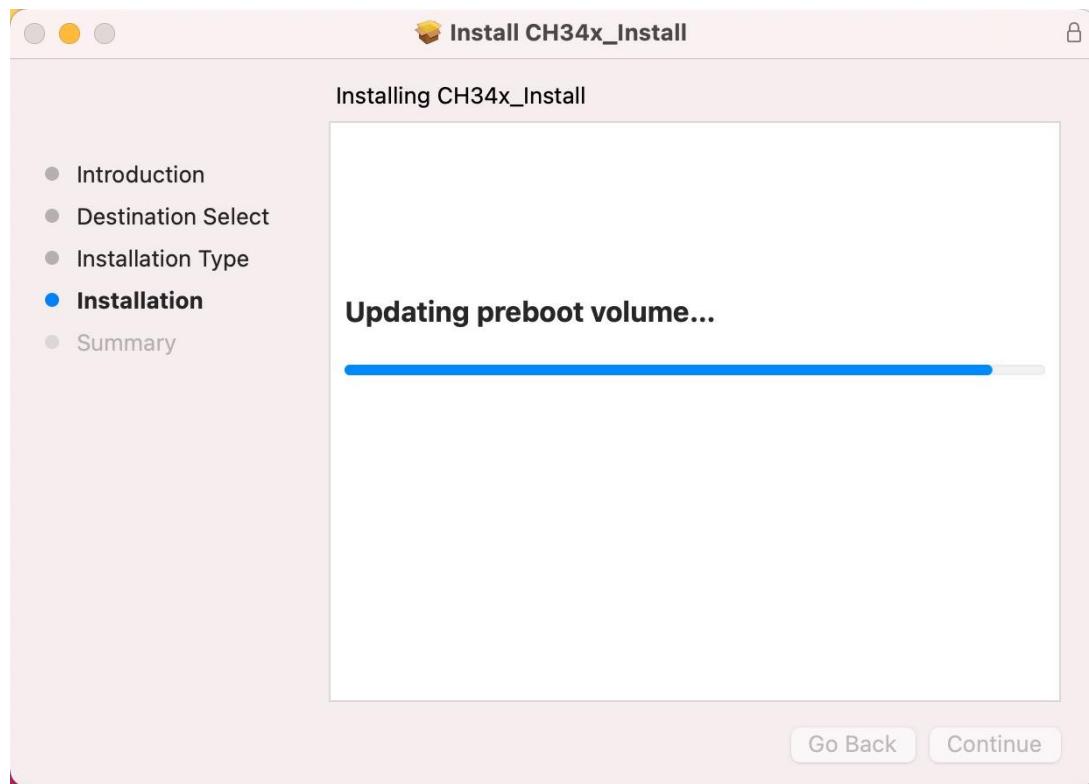
3. Click on Continue.



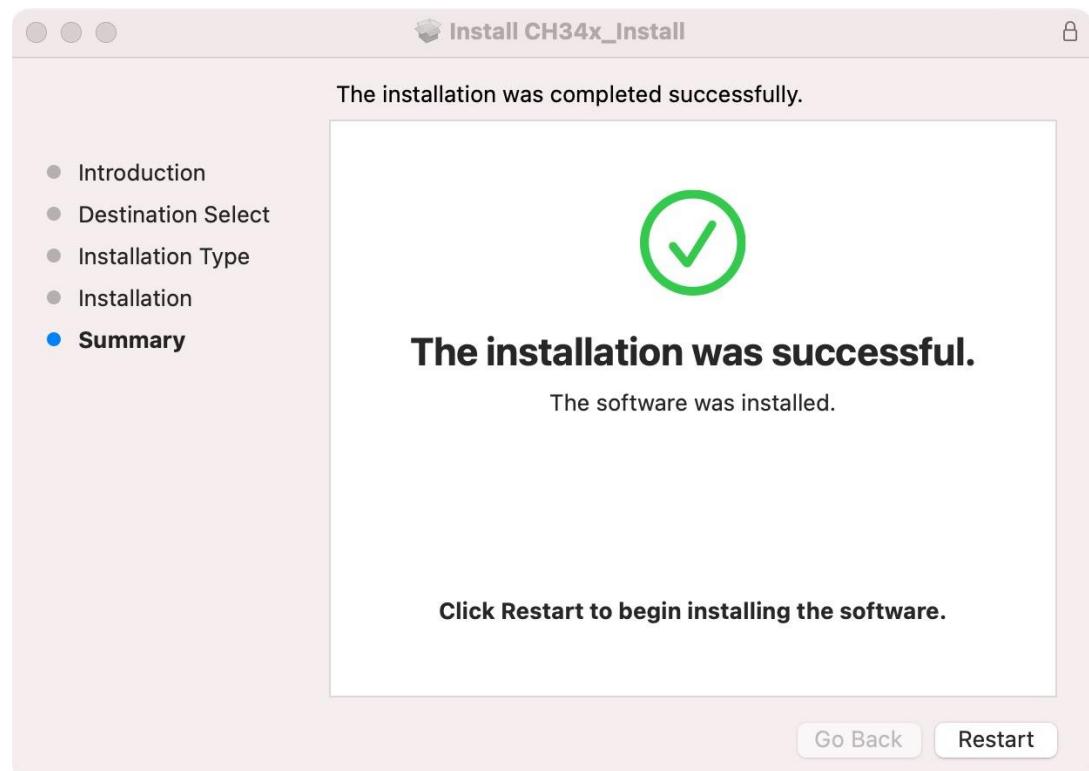
4. Click on Install.



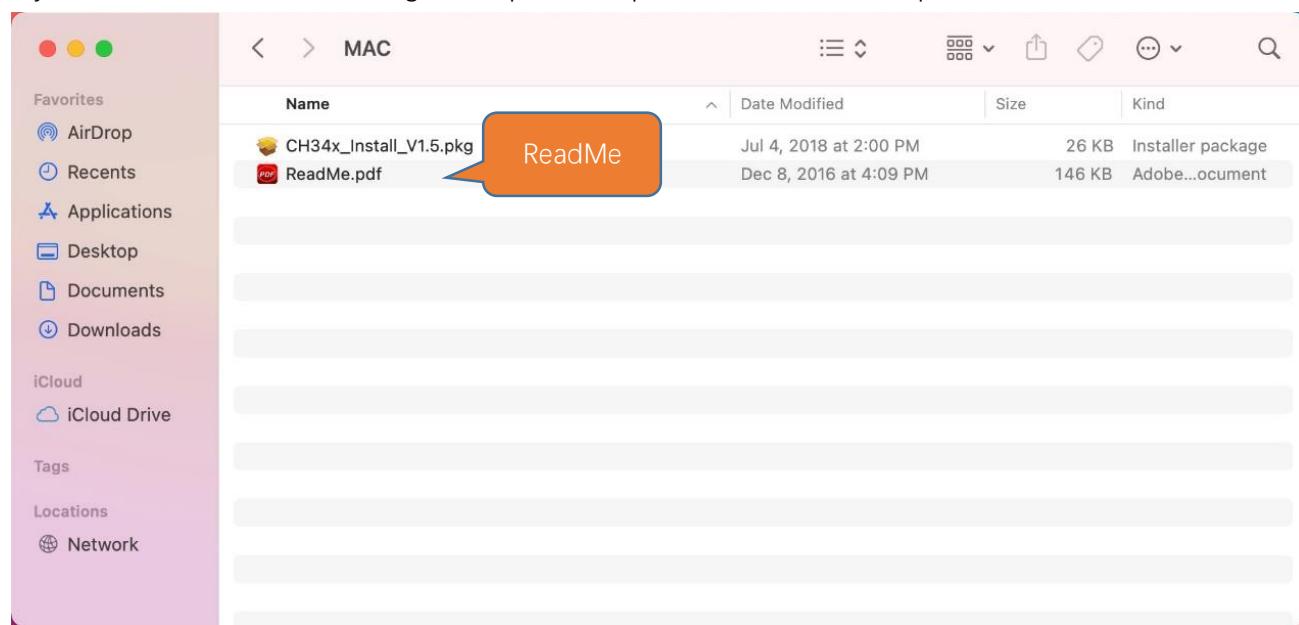
5. Wait for the installation to finish.



6. Restart your computer.



If you fail to install CH340 following the steps above, please refer to ReadMe.pdf.





02 Installation of Arduino IDE

We use Arduino Software (IDE) to compile and upload code to the control board.

If Arduino has been installed, you can skip to [next step](#).

Please visit <https://www.arduino.cc/en/software>

For Microsoft Windows users, please click following one.

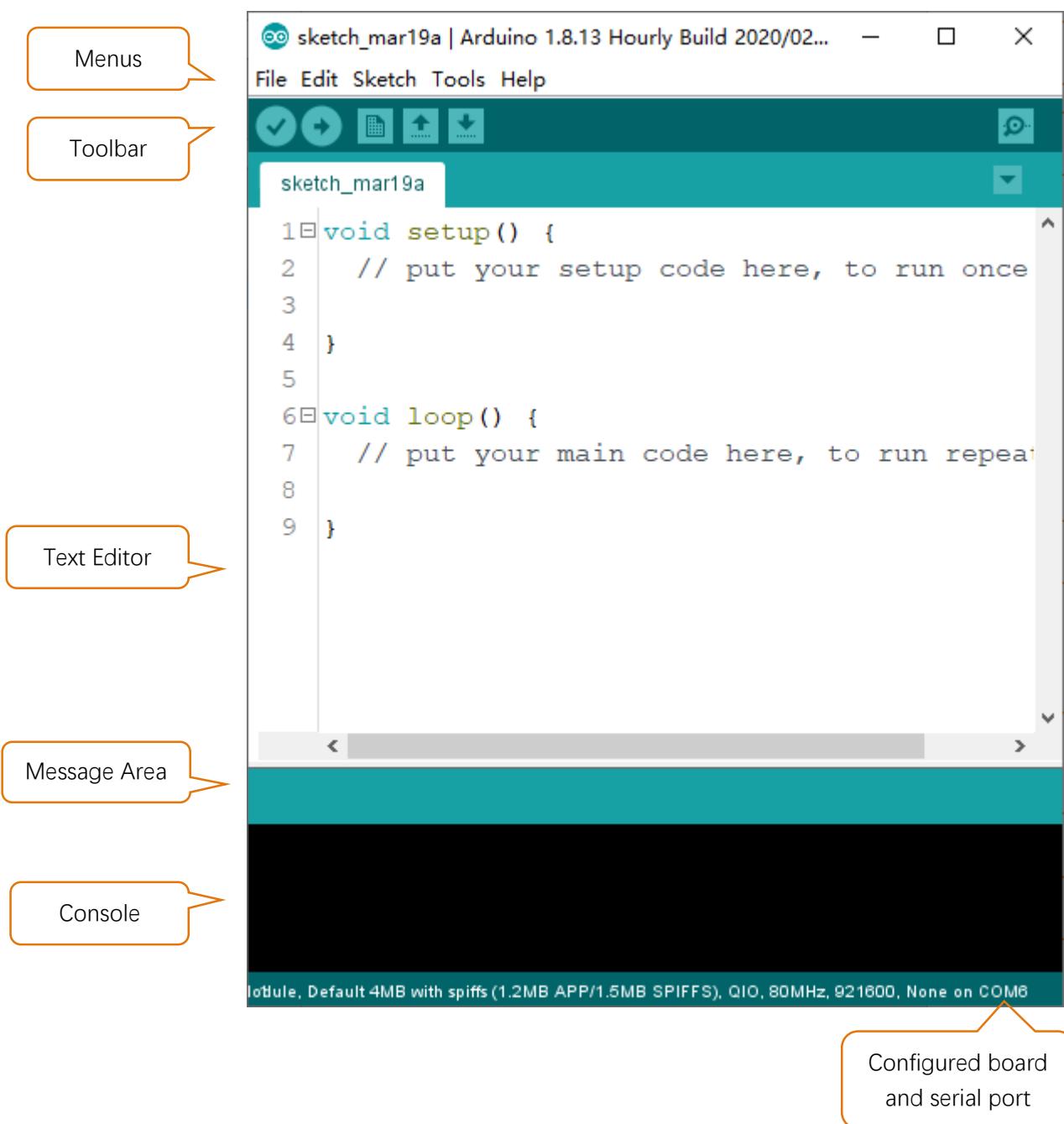
The screenshot shows the Arduino IDE 1.8.19 download page. On the left, there's a logo and a brief description of what the IDE does. Below that is a link to the 'Getting Started' page. Underneath is a 'SOURCE CODE' section with instructions about GitHub. On the right, there's a 'DOWNLOAD OPTIONS' sidebar with links for Windows (highlighted with a red arrow), Windows ZIP file, Windows app (with a 'Get' button), Linux 32 bits, Linux 64 bits, Linux ARM 32 bits, Linux ARM 64 bits, Mac OS X 10.10 or newer, Release Notes, and Checksums (sha512).

After the download completes, run the installer and complete the installation.

When finishing installation, there will be a shortcut generated on desktop and you can run it.



The interface of Arduino IDE is as follows:





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



Verify

Checks your code for errors compiling it.



Upload

Compiles your code and uploads it to the configured board.



New

Creates a new sketch.



Open

Open a new sketch.



Save

Save your sketch.



Serial Monitor

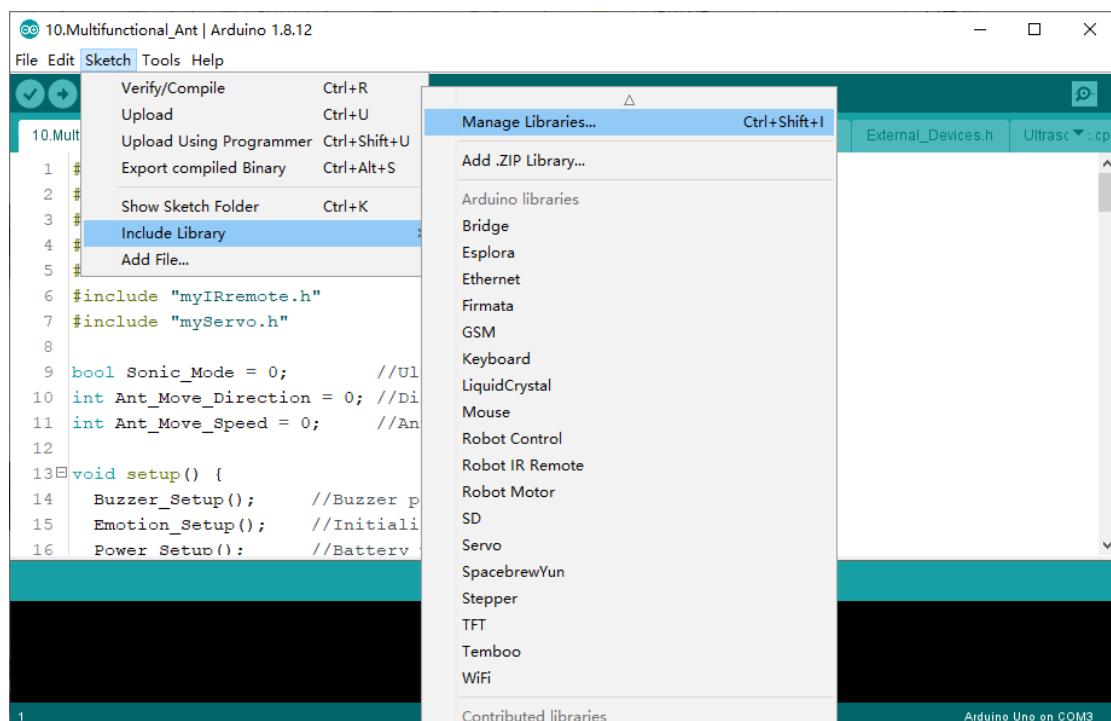
Open the serial monitor.

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

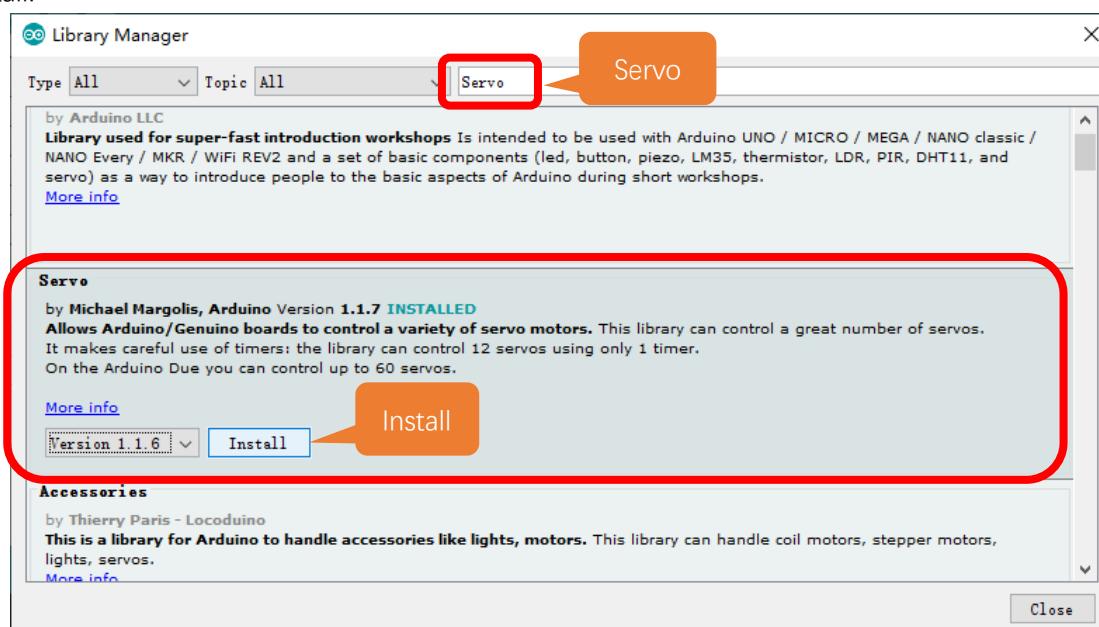
03 Installation of Libraries

In this tutorial, we use several libraries: Servo, UltrasonicSensor, Freenove_VK16K33_Lib and Freenove_WS2812_RGBLED_Controller respectively. Before running the sketches we provide, please make sure all the libraries have been installed. Otherwise, they may fail to run.

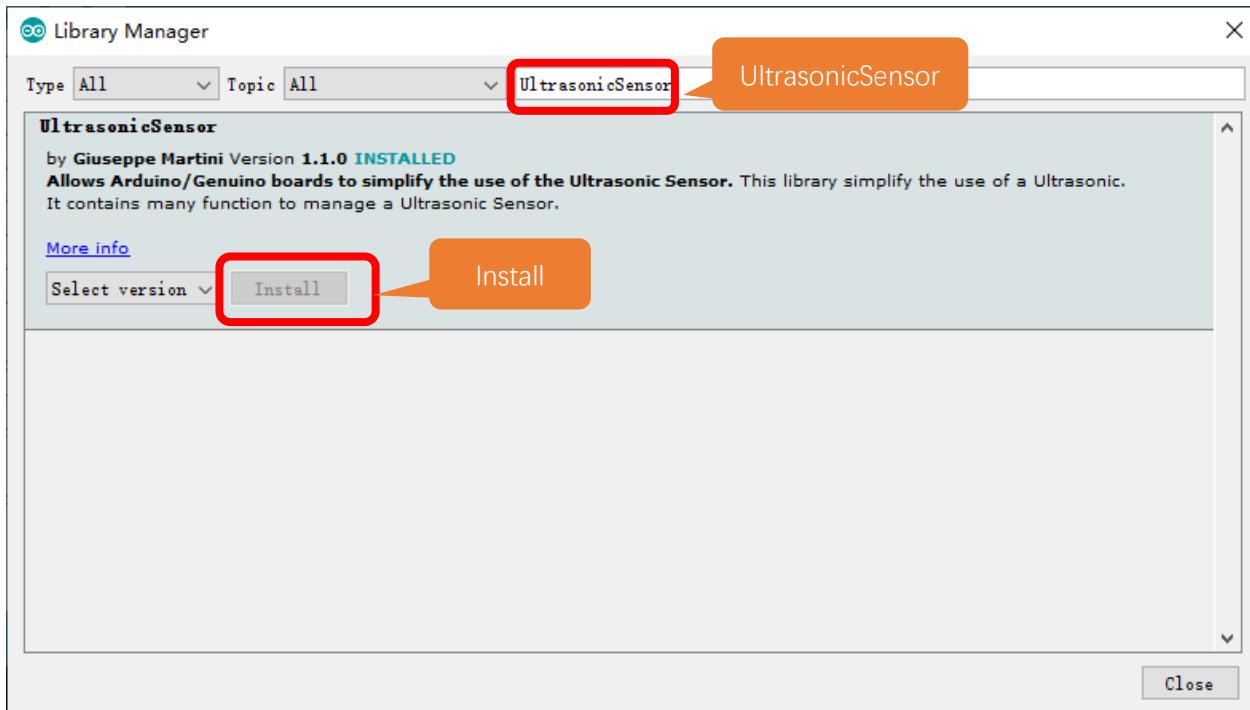
1. Open Arduino IDE, click on Sketch on Menus, select Include Library and then click on Manage Libraries...:



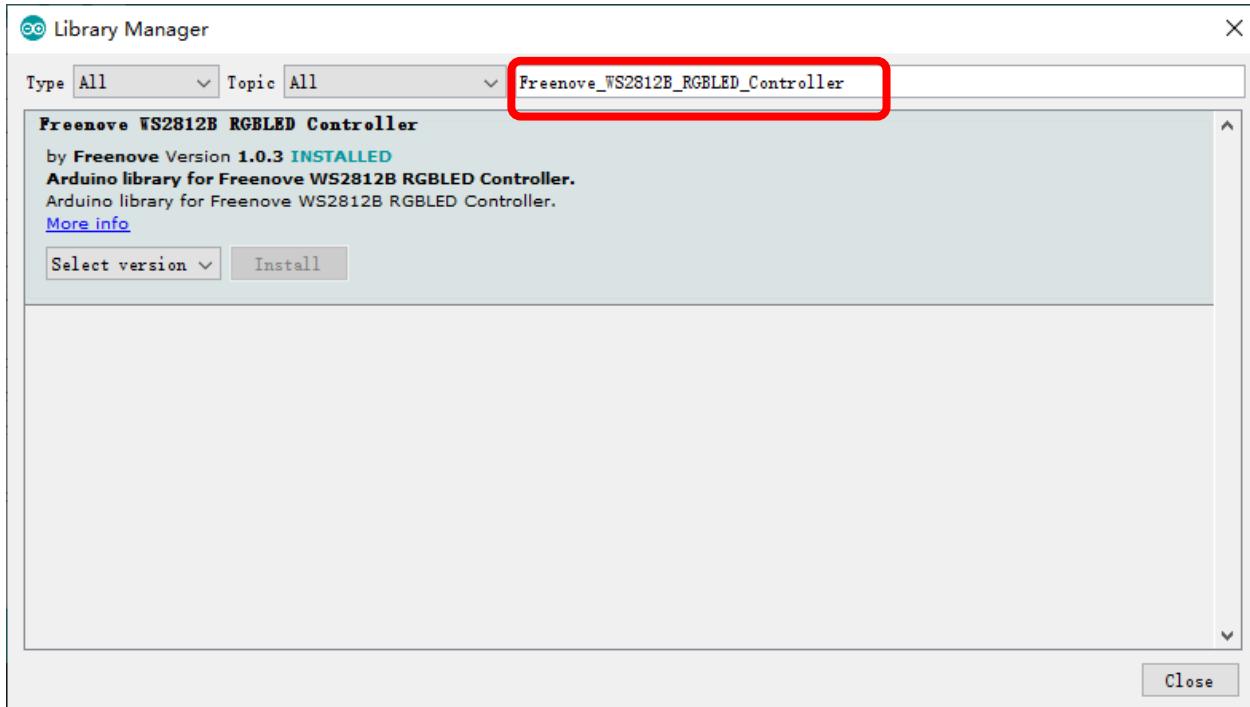
2. On the pop-up window, input **Servo** and press Enter. Select the Servo library as marked below and click to install.



3. After Servo library installs successfully, go on to input the name of the next library, that is, **UltrasonicSensor** and then install it.

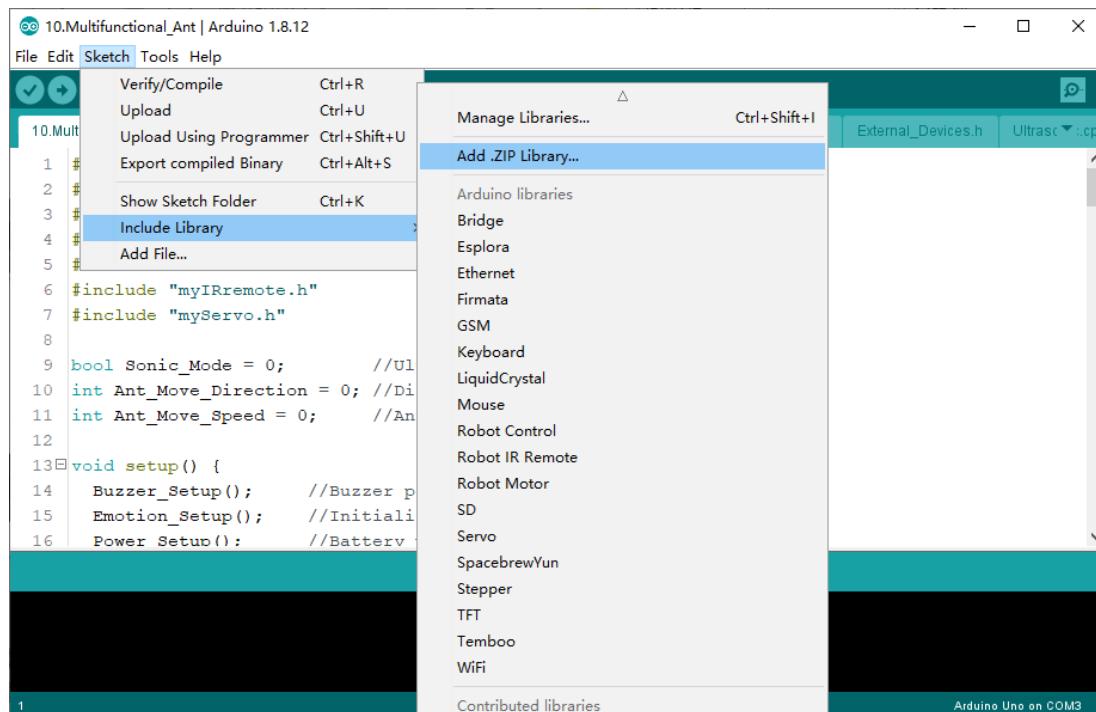


4. After finishing installation, install the next library **Freenove_WS3212B_RGBLED_Controller** in the same way.



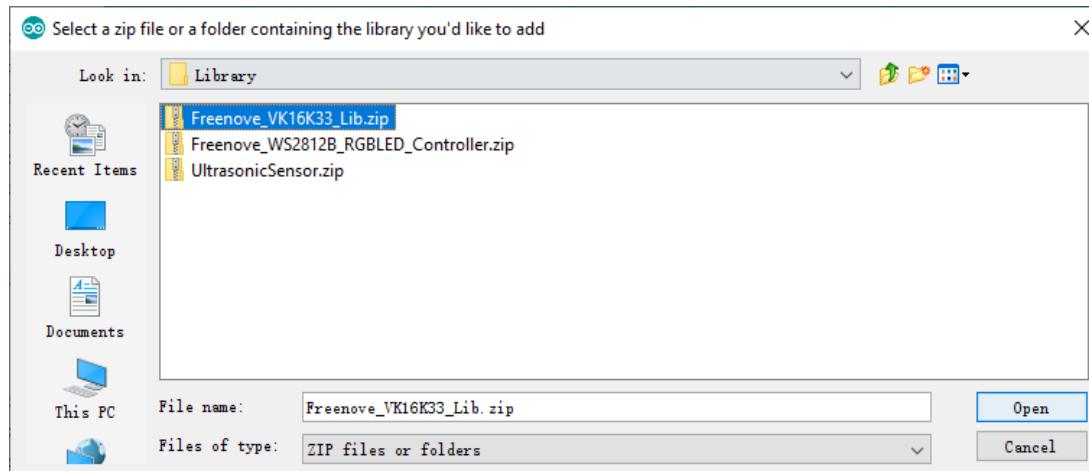
There is also another way to install libraries.

5. Click on Sketch on Menus, select Include Library, and then click on Add .ZIP Libraries….

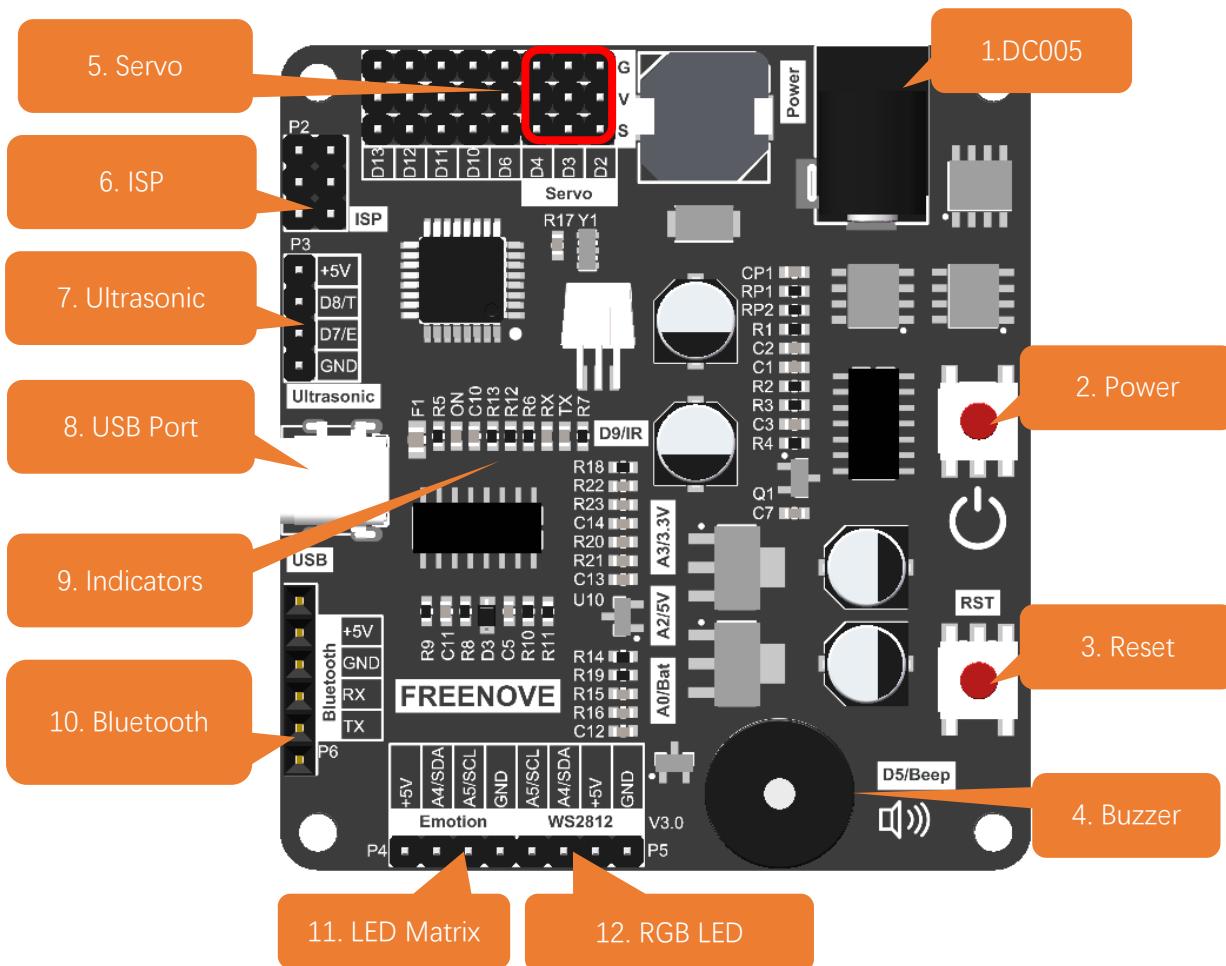


6. On the pop-up window, select the source pakage

Freenove_Robot_Ant_Kit/Library/Freenove_VK16K33_Lib.zip and click on Open.



04 Control board



1. DC005: Powered by 2 18650 batteries. Supports 7-12V power supply.
2. Power button: Uses digital chip circuits to control the power's ON and OFF. It's more durable than traditional switches.
3. Reset button: Reset the control board.
4. Passive buzzer: Controlled by D5 pin on ArduinoNo.
5. Servo: Controlled by D2, D3 and D4 on Arduino with D2 connecting to Servo1, D3 connecting to Servo2 and D4 connecting to Servo3.
6. ISP connector: Uses ISP to download programmer to download code.
7. Ultrasonic connector: If you use ultrasonic module for the ant's head, please connect it to this port. Connect D8 of the Arduino to Trig pin of ultrasonic module and D7 to Echo.
8. USB port: Connect computer and control board with a type-C cable to download programs.
9. Indicators: Used to indicates whether the board is powered and the status of the serial port.
10. Bluetooth connector: Plug the Bluetooth module HC-05 to this port. Note: The Bluetooth module should orient to the control board.
11. LED matrix connector: If you use LED matrix for the ant's head, please connect it to this port.
12. RGB LED module: the IIC port to connect to RGB LED module.

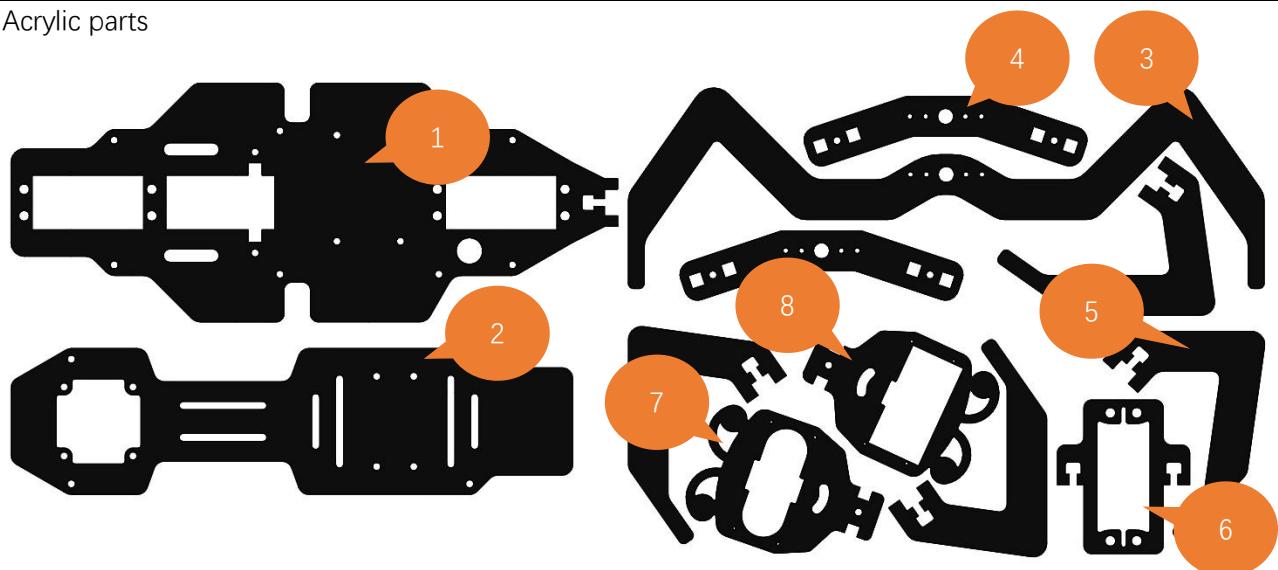
The pinout of the control board is as below:

Pins	Functions	Introduction
0	Uart-RX	Bluetooth module. Please remove it when uploading code.
1	Uart-TX	
2	Servo1	Servo motors
3	Servo2	
4	Servo3	
5	Buzzer	Buzzer
6		Extension IO
7	Ultrasonic-Echo	Ultrasonic module
8	Ultrasonic-Trig	
9	Infrared receiving pin	Infrared module
10		Extension IO
11		ISP/Extension IO
12		ISP/Extension IO
13		ISP/Extension IO
A0	Battery Voltage Pin	Detect voltage of batteries (1/4)
A1	AREF Pin	Detect voltage of AREF pin
A2	5V Voltage Pin	Detect 5V voltage (1/4)
A3	3.3V Voltage Pin	Detect 3.3V Voltage (1/4)
A4	IIC-SDA	WS2812 RGB LED module/LED Matrix (Expression module)
A5	IIC-SCL	

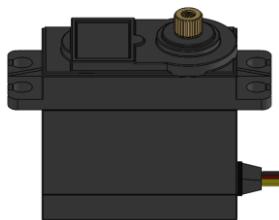
05 Component List

Before assembly, please compare the list with the kit you receive to make sure that no item is missed or damaged. Should this happens, please send emails to us: support@freenove.com, we will offer solution.

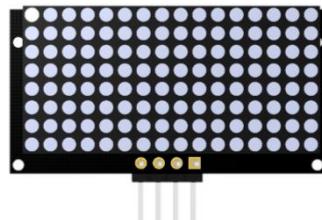
Acrylic parts



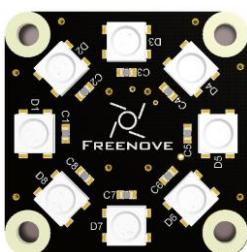
Servo *3



Expression Module *1



8-digit RGB LED Module *1



I2C RGB LED Control Module *1



Ultrasonic Module *1

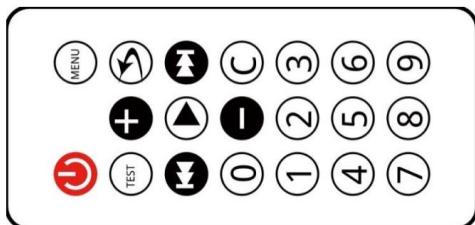
Bluetooth Module *1



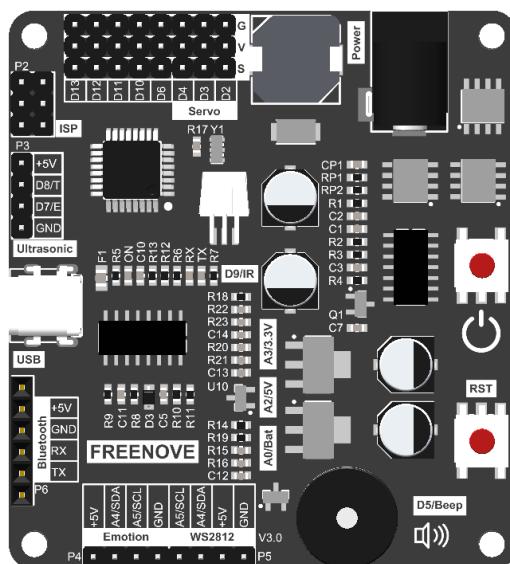
Infrared Remote Controller *1



Battery Holder *1



Control board *1



4P F-F Jumper Wires 20cm*3



3P F-F Jumper Wires 20cm*1



Tidy Cable 15cm *1



Type-C USB Cable *1



Metal cross sleeve*1



Machinery Parts



Cross Screwdriver (3mm) *1



Cross Screwdriver (2mm) *1



PVC rubber sheath *8



Required but NOT Contained Parts

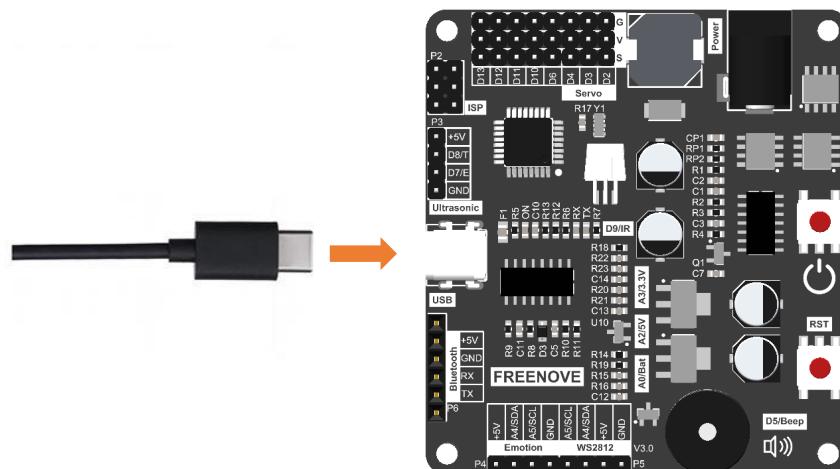
2 x 3.7V 18650 lithium **rechargeable** batteries with continuous discharge current >3A.Please refer to **AboutBattery.pdf** to purchase the batteries.

06 Assembly

Set Servo Gear to 90°

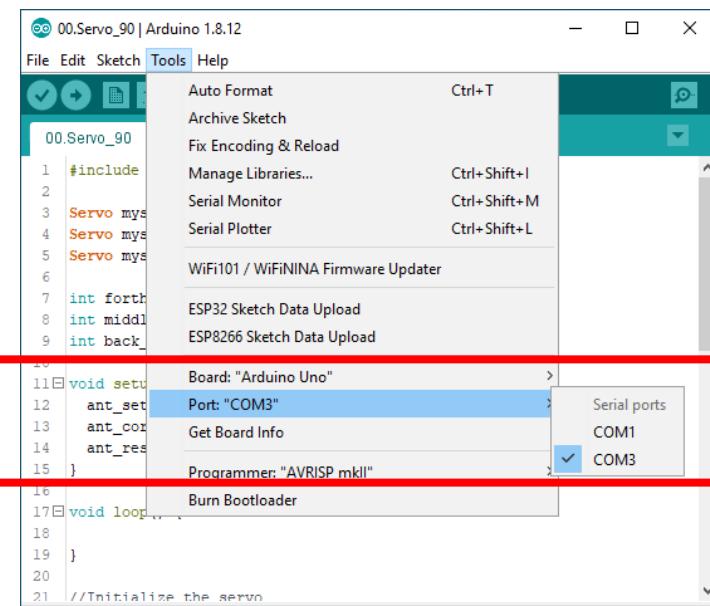
Before installing the legs, please set the servos to 90°. Only after doing this can you assemble the robot ant correctly.

Step 1: Connect the control board and computer with a type-C cable.



Step 2: Open the file 00.Servo_90.ino in **Freenove_Robot_Ant_Kit\Sketches\00.Servo_90**.

Step 3: Click on Tools and then select “Arduino Uno” for Board and port COMx for port. In our computer, the port is COM3.



Note:

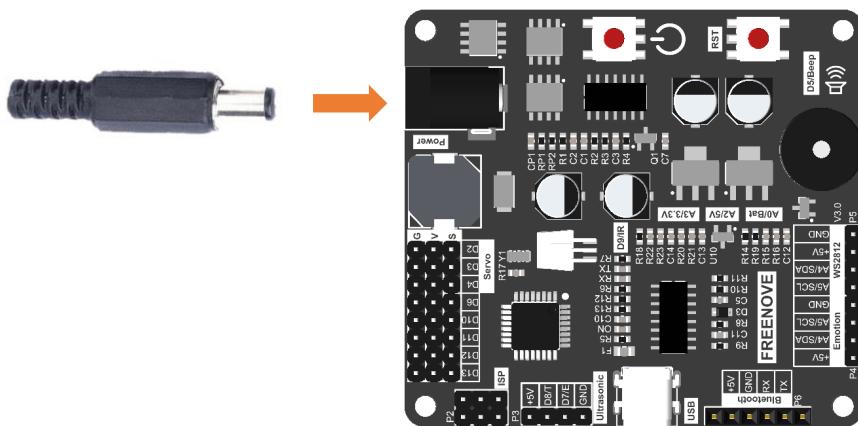
1. Generally, COM1 is not the port of development board, so please do NOT choose it.
2. The x of COMx varies among different computers. If you cannot find your port, please check whether CH340 has been installed or whether the board has connected to PC.

Any concerns? ✉ support@freenove.com

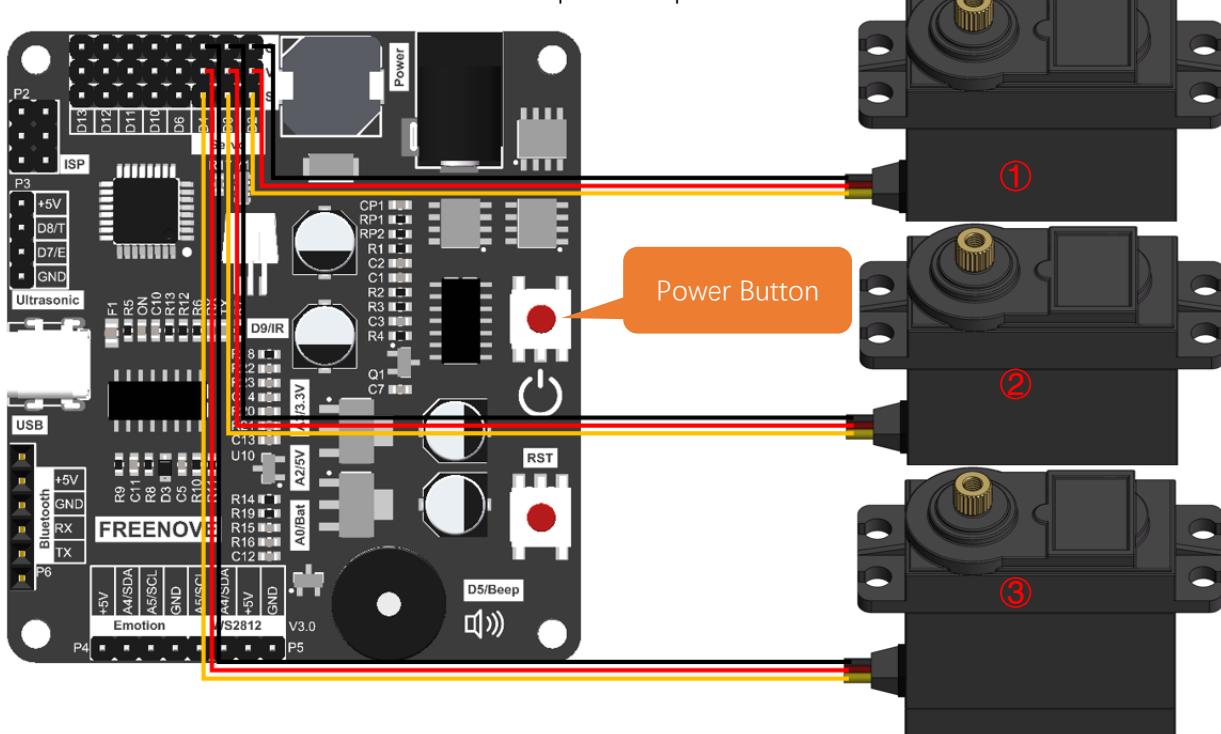
Step 4: Click to download Code (Upload Using Programmer).



Step 5. Plug the power cord from the battery holder into control board.



Step 6. Connect 3 servos to the control board and press the power button.



Step 7: When you see the power indicator on the board light up, and all the 3 servos rotate to a specific position and remain still, it means you have set the servos to the middle position. At this point, you can assemble the legs.

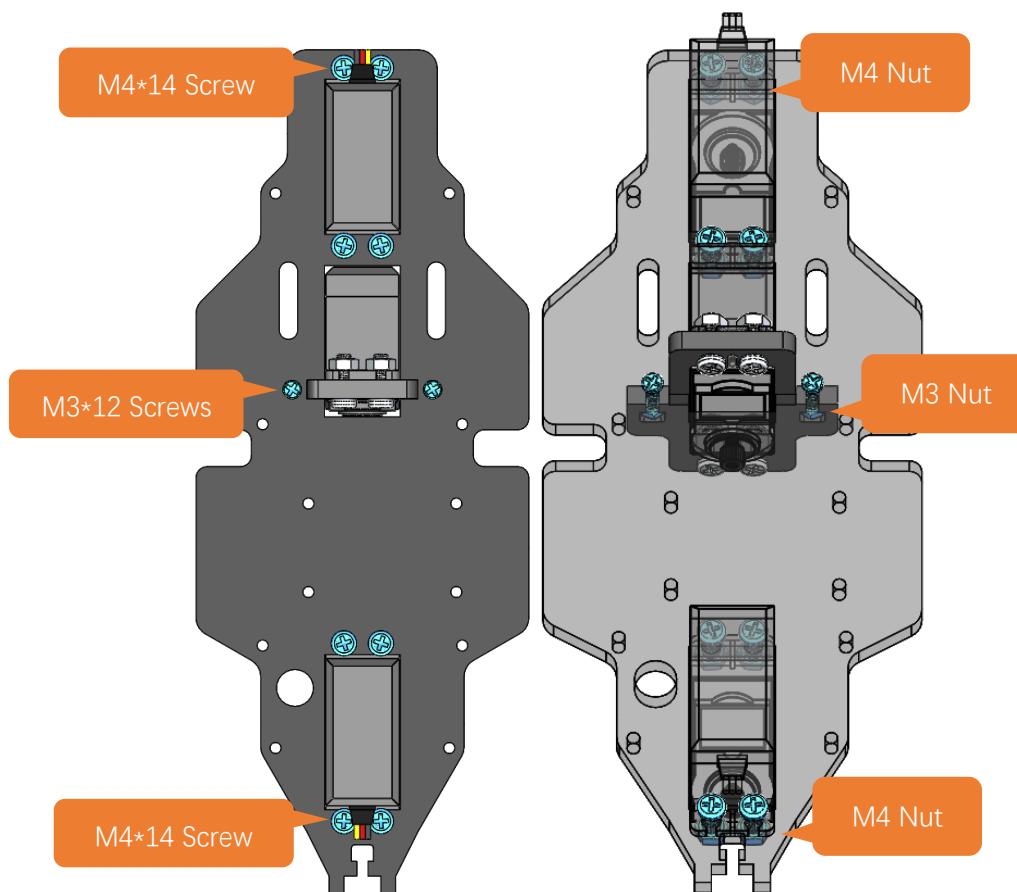
If the servos have been at that position, you won't observe anything.

Assembly of Legs

Use 4 M4*14 screws and 4 M4 nuts to fix a servo to No.6 acylic part.



Fix 3 servos to No.1 acrylic part. (8 M4*14 screws, 8 M4 nuts, 2 M3*12 screws and 2 M3 nuts)



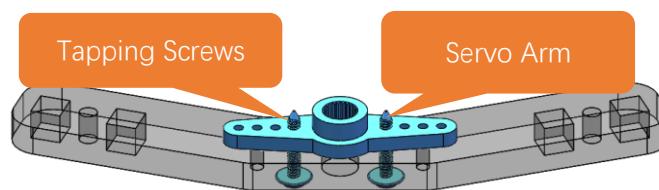
On the left is the plan view of the acrylic part.

On the right is its perspective view.

Use 4 tapping screws to fix the disc servo arms to two No.4 acrylic parts.

Note: Tapping screws, disc servo arms and servos are packed in the servo package, please don't use wrong ones.

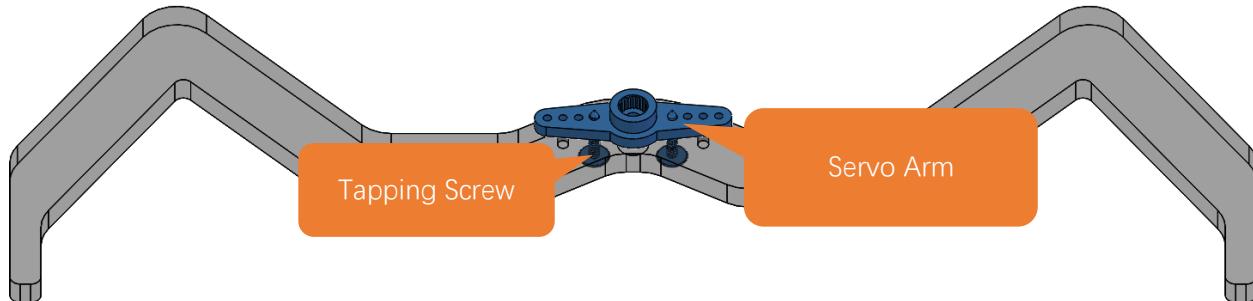
It is a bit strenuous to make the disc servo arm fix tightly, which is normal.



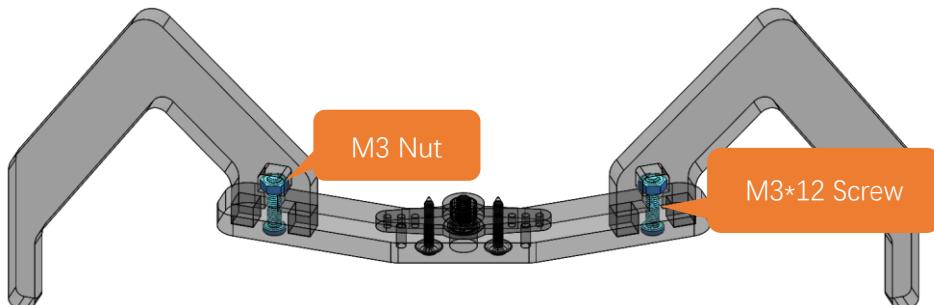
Use 2 tapping screws to fix the disc servo arms to No.3 acrylic part.

Note: Tapping screws, disc servo arms and servos are packed in the servo package, please don't use wrong ones.

It is a bit strenuous to make the disc servo arm fix tightly, which is normal.

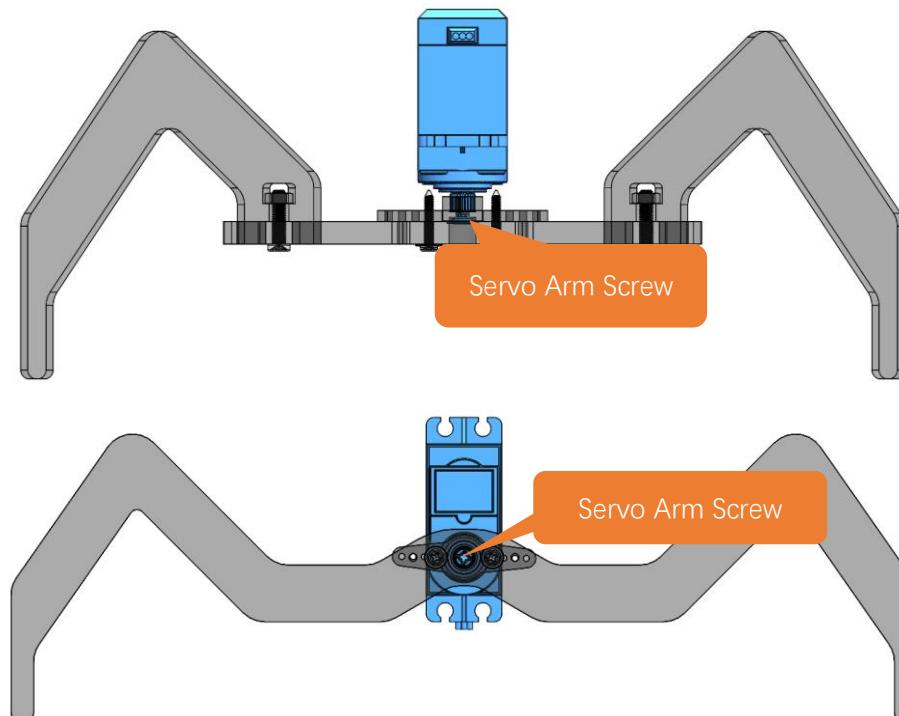


Use 4 M3*12 screws and 4 M3 nuts to fix 4 No.5 acrylic parts to 2 No.4 acrylic parts.

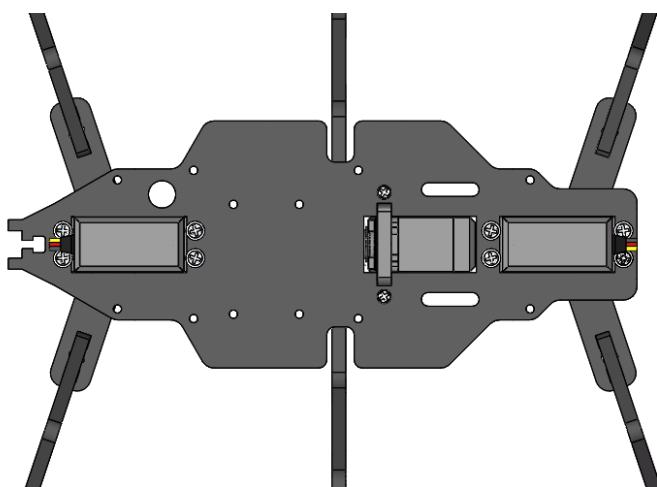


Use 3 servo arm screw to fix 3 servos to No.3 acrylic part and 2 No.4 acrylic parts respectively.

Note: servo arm screws are packed with the servos. Don't use wrong ones.

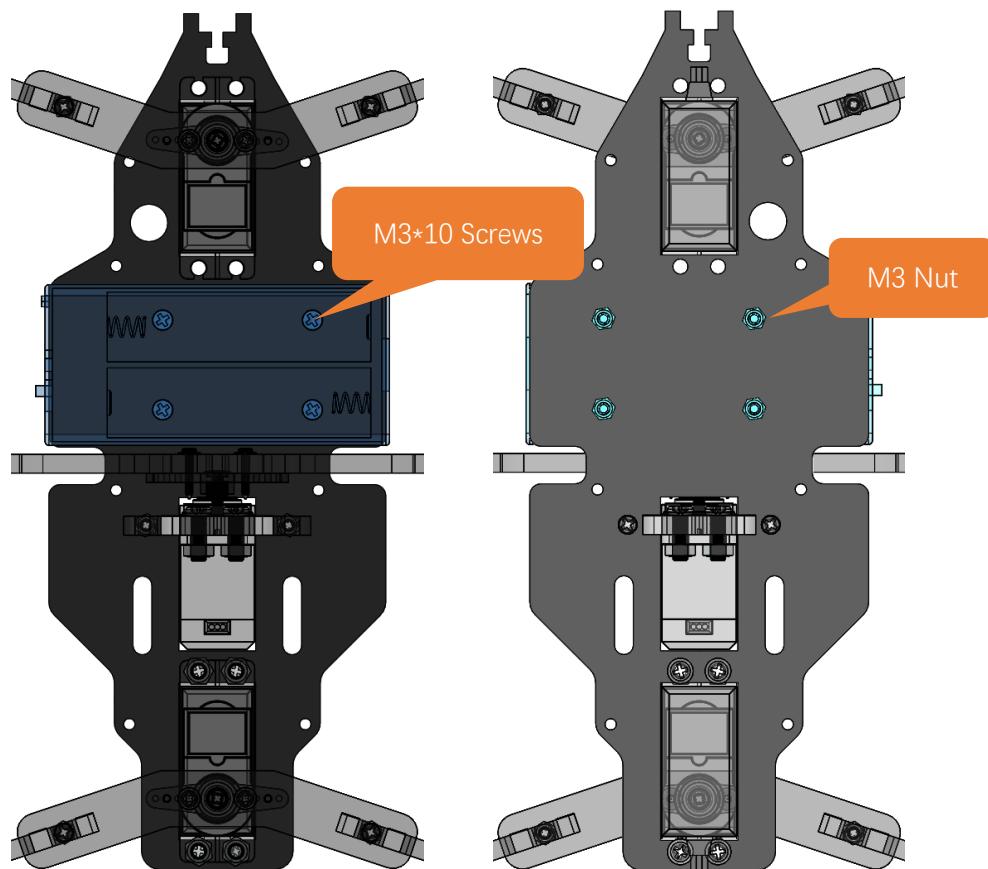


When finish assembly, from the top view, it looks as below:



Assembly of Battery Holder

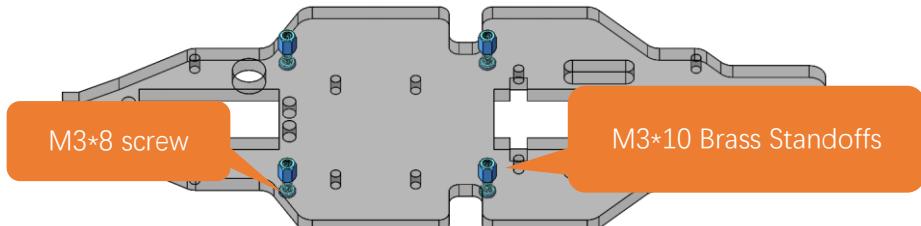
Use 4 M3*10 screws and 4 M3 nuts to fix the 18650 battery holder to the back of No.1 acrylic part.



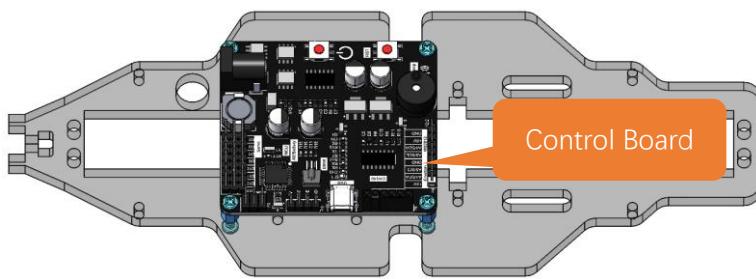
The left is the bottom view and the right is the top view

Assembly of the Control board

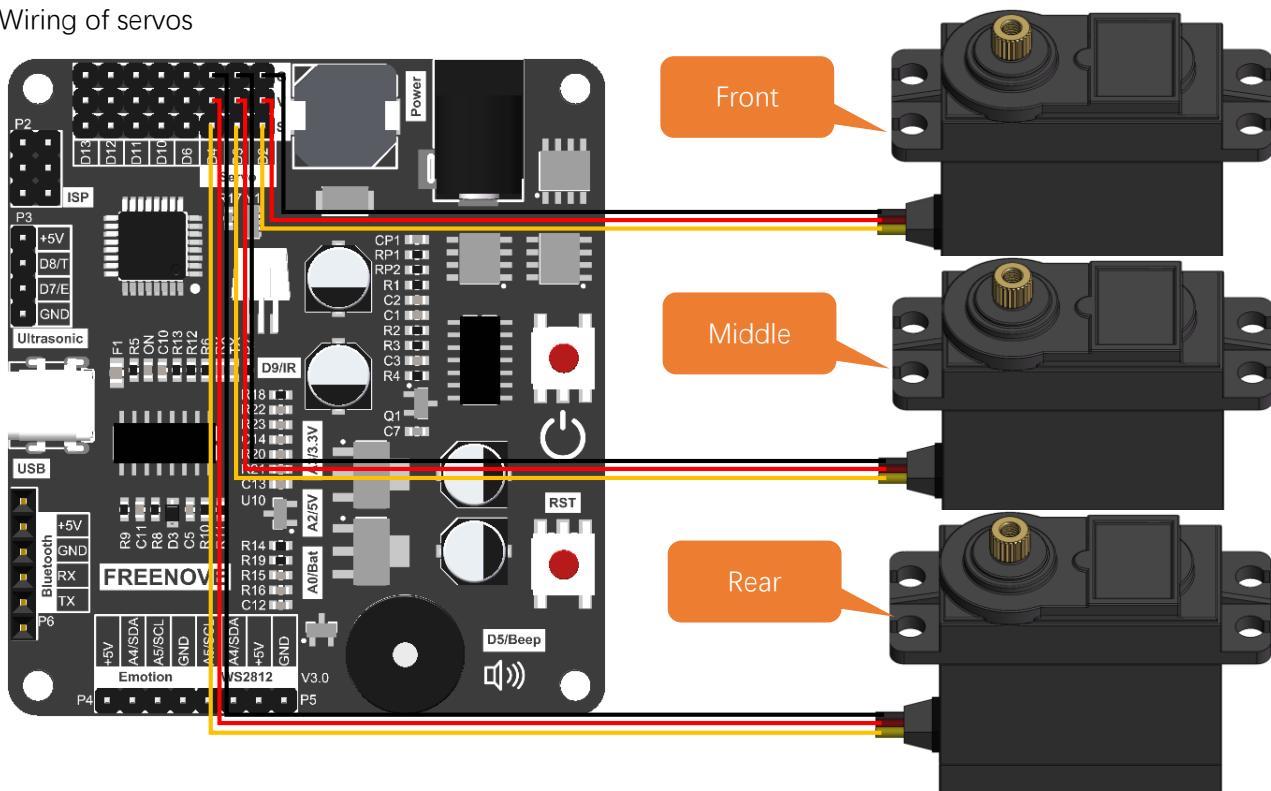
Use 4 M3*8 screws to fix 4 M3*10 brass standoffs to No.1 acrylic part. (Top view)



Use 4 M3*8 screws to fix the control board to the brass standoffs. Pay attention to the orientation of the control board. (Top view)



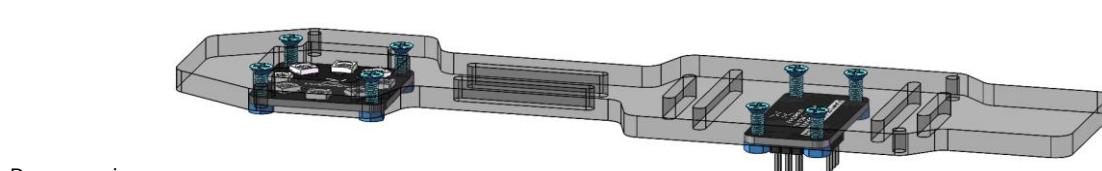
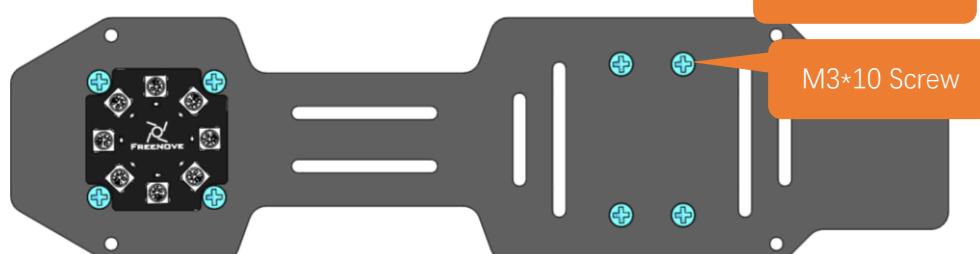
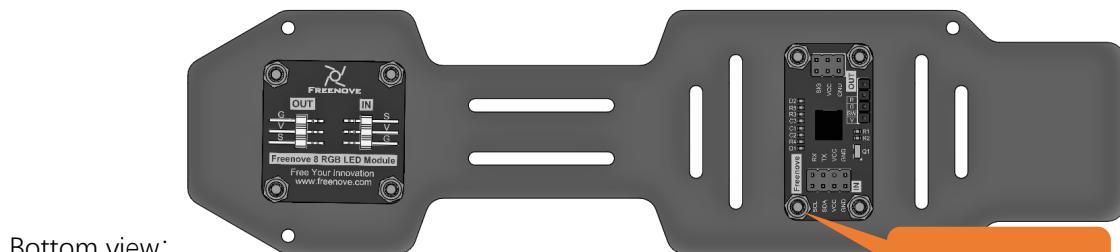
Wiring of servos



Control board (Servo)	Servo1	Servo2	Servo3
S	orange-D2	orange-D3	orange-D4
V	red-VCC	red-VCC	red-VCC
G	black-GND	black-GND	black-GND

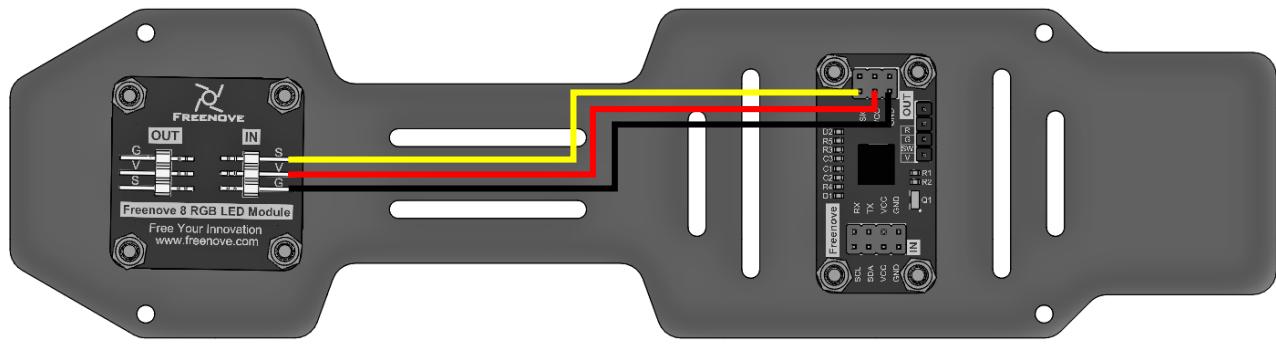
Assembly of the WS2812 LED Module

Use 8 M3*10 screws and 8 M3 nuts to fix the LED module and LED control module to No.2 acrylic parts.



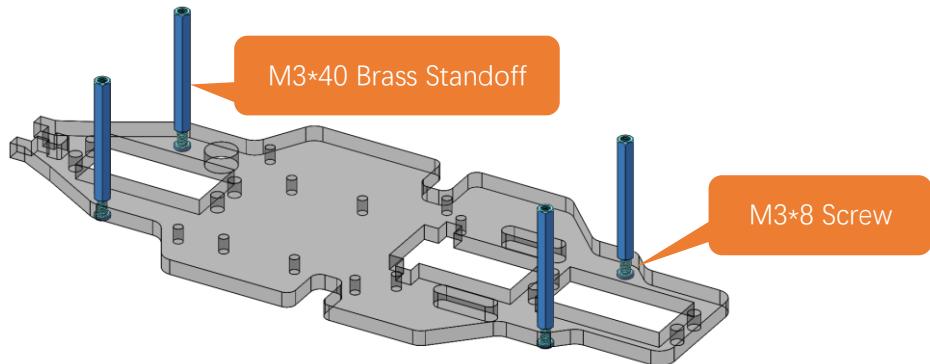
Perspective:

Wiring of RGB LED module to its control module



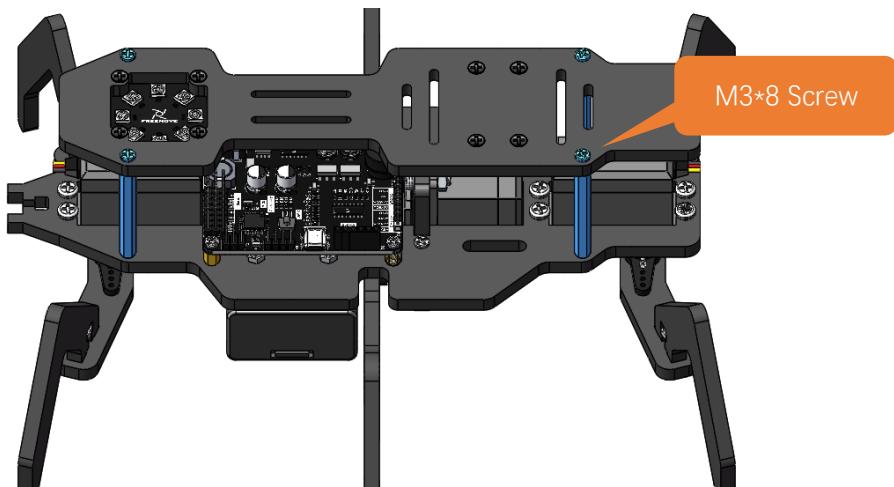
Assembly of the Body

Use 4 M3*8 screws to fix 4 M3*40 brass standoff to No.1 acrylic part. (Top view)



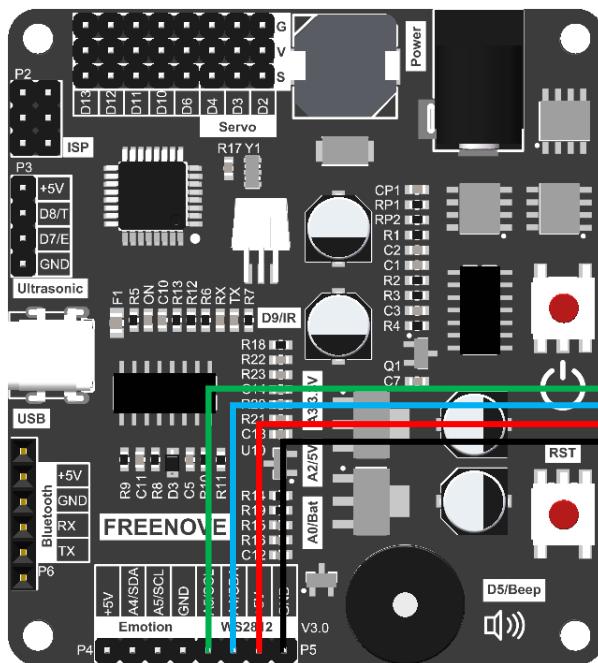
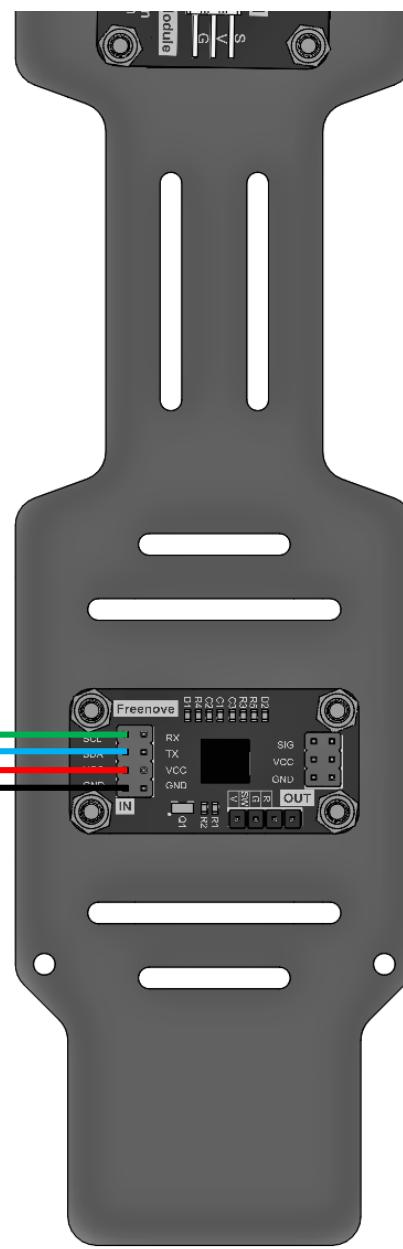
Use 4 M3*8 screws to fix No.2 acrylic part to the standoffs. (Top view)

Pay attention to the orientation of No.2 acrylic parts.



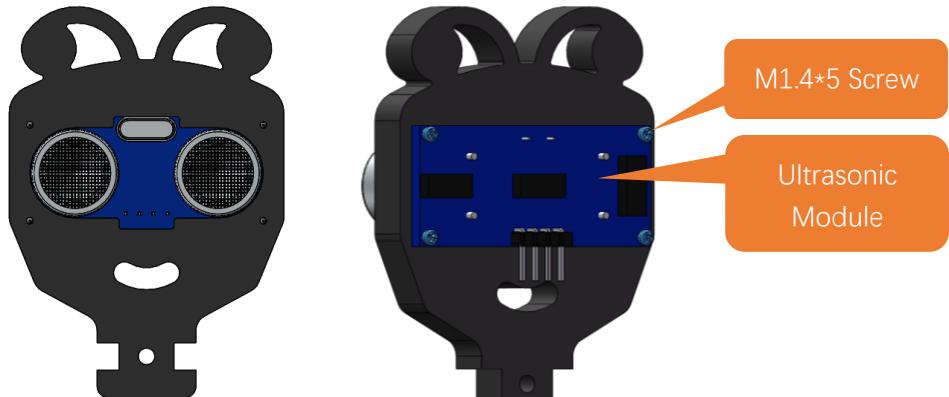
Wiring of the LED control module to control board

Control board (WS2812)	LED control module
A5/SCL	SCL
A4/SDA	SDA
+5V	VCC
GND	GND

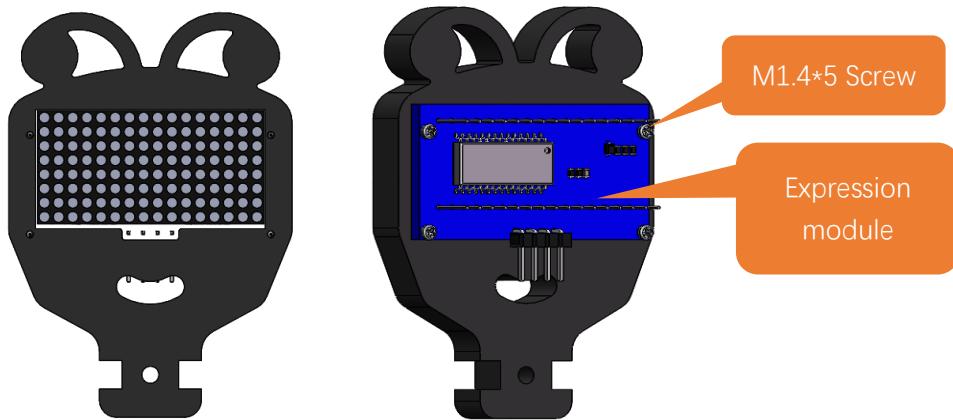


Assembly of the Head

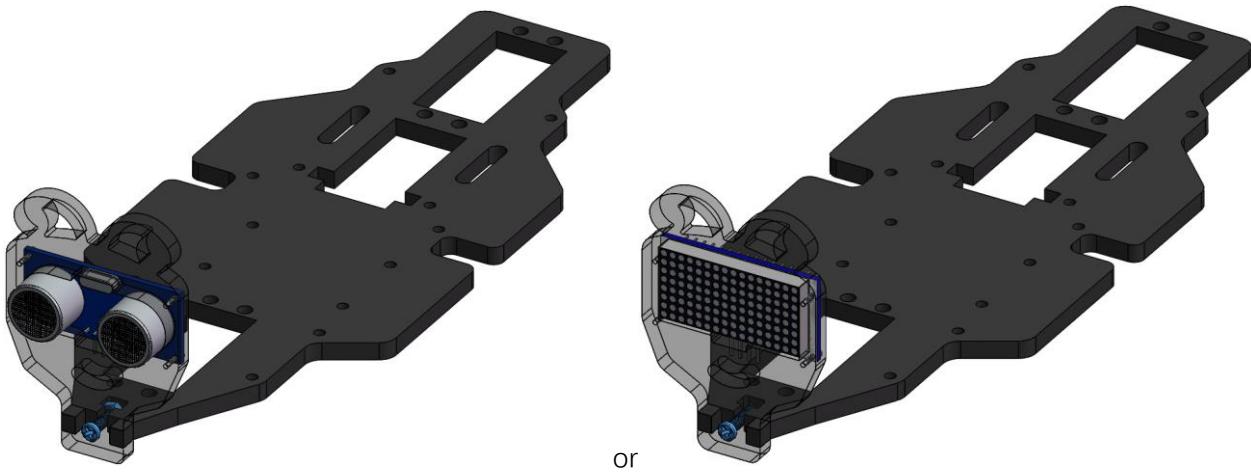
Use 4 M1.4*5 tapping screws to fix the ultrasonic module to No.7 acrylic part.



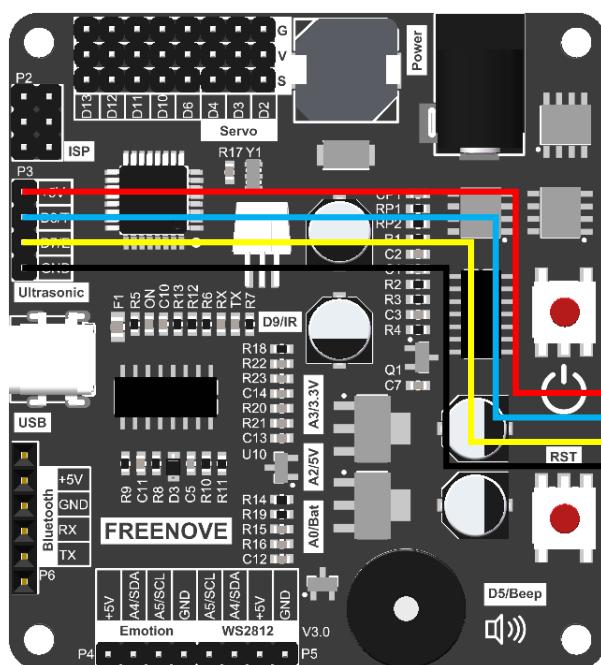
Use 4 M1.4*5 tapping screws to fix the expression module to No.8 acrylic part.



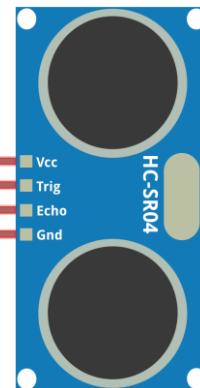
Use a M3*12 screw and a M3 nut to fix No.7 or No.8 acrylic part to No.1 acrylic part.



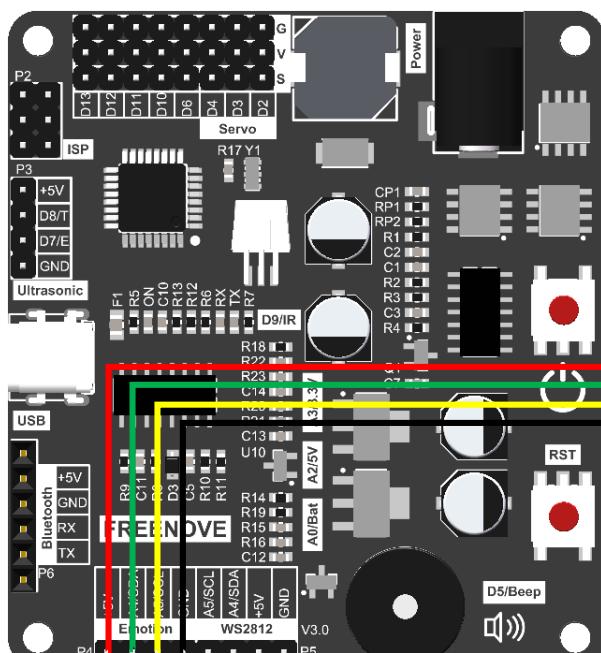
Wiring of ultrasonic module to control board



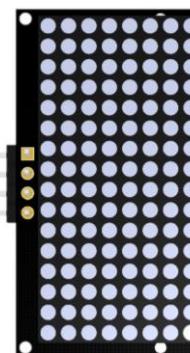
Control board (WS2812)	Ultrasonic module
+5V	Vcc
D8/T	Trig
D7/E	Echo
GND	GND



Wiring of the expression module to control board.



Control board (WS2812)	Expression module
+5V	VCC
A4/SDA	SDA
A5/SCL	SCL
GND	GND



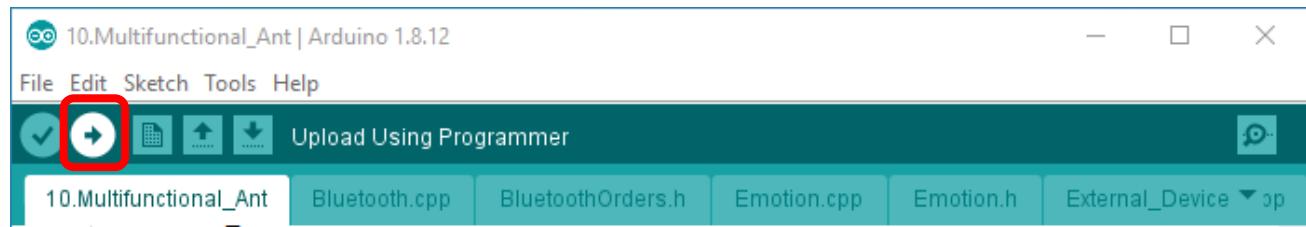
When finishing the assembly of the robot ant and make sure all the wirings are correct, you can use tidy cable to arrange the wires. If you have any questions during the assembly, please send emails to us: support@freenove.com.

Any concerns? ✉ support@freenove.com



07 Test

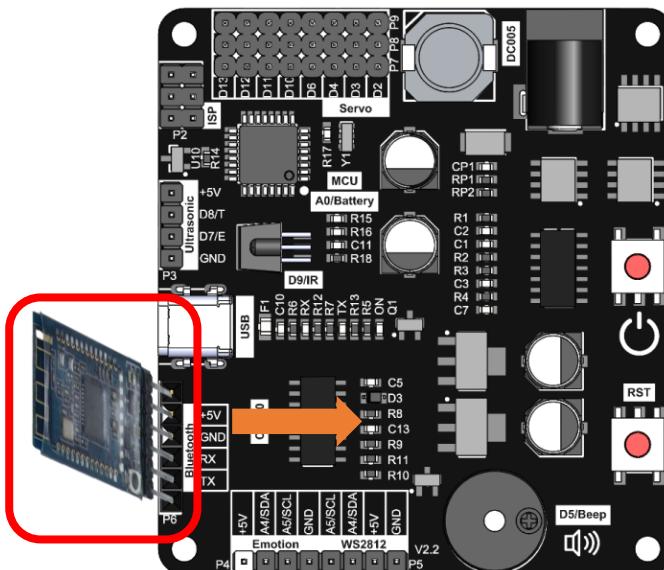
- 1, Connect the control board and computer with a type-C cable.
- 2, Open **10.Multifunctional_Ant.ino** in **Freenove_Robot_Ant_Kit\Sketches\10.Multifunctional_Ant**. Click to download the code to control board.



- 3, When the code is downloaded successfully, you can unplug the cable.

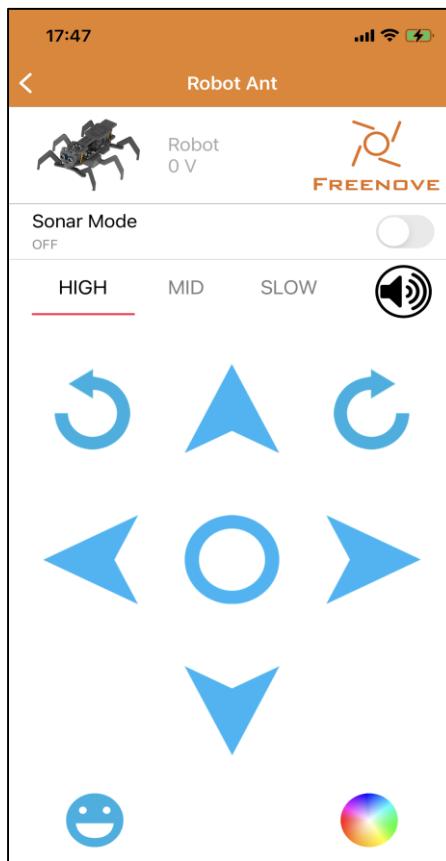
Note: Remember to remove Bluetooth module every time when you upload code; Otherwise the uploading will fail.

- 4, Plug in the Bluetooth module. Please pay attention to the orientation of the Bluetooth module.



5, Turn ON the power. When you hear a beep from buzzer, it means that the robot ant is ready. You can control it through the phone APP.

App download address: https://github.com/Freenove/Freenove_app_for_Android



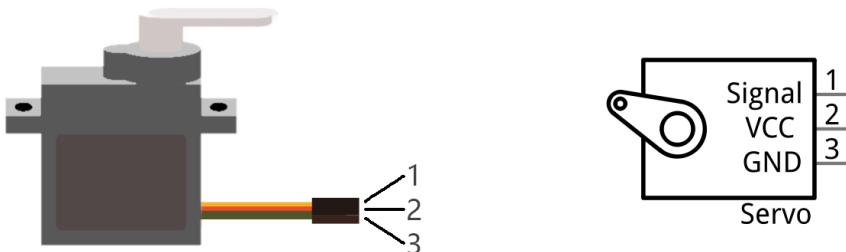
You can also control the robot with Infrared remote controller.

		Reset				Buzzer ON/OFF
		Spot Turn to Left		Forward		Spot Turn to Right
		Forward Left		Speed Adjustment		Forward Right
		Obstacle Avoidance		Backward		Obstacle Avoidance Mode OFF
		Red of RGB LED ON/OFF		Green of RGB LED ON/OFF		Blue of RGB LED ON/OFF
		Modes of WS2812		Modes of Dynamic Expressions		Random Display of Static Expressions
		Custom		Custom		Custom

Chapter1 Servo Test(Importance)

Component Knowledge

Servo is an auto-control system, consisting of DC motor, reduction gear, sensor and control circuit. Usually, it can rotate at a range of 180 degrees. Servo can output larger torque and is widely used in model airplane, robot and so on. It has three lines, including two for electric power line positive (2-VCC, red), negative (3-GND, brown), and the signal line (1-Signal, orange).



We use 50Hz PWM signal to drive the servo. The lasting time 0.5ms-2.5ms of PWM single cycle high level corresponds to the servo angle 0 degrees - 180 degrees 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, servo will rotate to the designated position.

Circuit

Schematic diagram

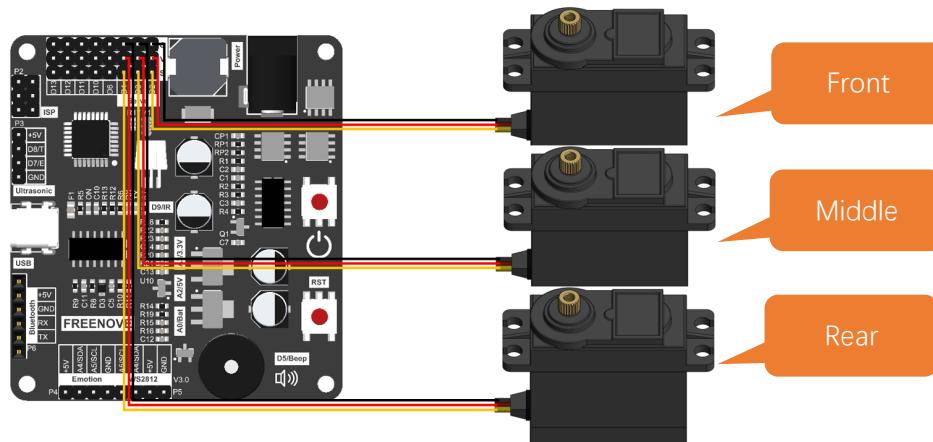
<p>Servo</p> <p>P7 P8 P9</p>	<p>Main control chip pins</p>
--	-------------------------------

If you don't feel like using D2-D4, you can change the wiring and modify code to use D6, D10-D13

Hardware connection. If you need any support, please feel free to contact us via: support@freenove.com

Any concerns? ✉ support@freenove.com

Wiring of servo to main board:



Sketch

Upload code to control board to control the robot ant.

Open the 01.1.Servo_Test.ino in **Freenove_Robot_Ant_Kit\Sketches\01.1.Servo_Test**.

If you are interested in the realization of functions in myServo.h file, you can click on myServo.cpp to check.

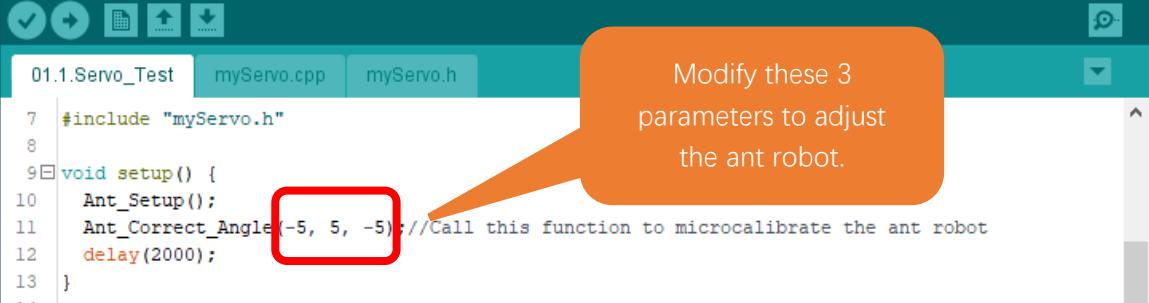
You can click .cpp to check the source code of functions for each

```

01.1.Servo_Test | Arduino 1.8.12
File Edit Sketch Tools Help
01.1.Servo_Test myServo.cpp myServo.h
7 #include "myServo.h"
8
9 void setup() {
10     Ant_Setup();
11     Ant_Correct_Angle(-5, 5, -5); //Call to correct angle
12     delay(2000);
13 }
14
15 void loop() {
16     ant_run_forth(2,7); //Forward
17     delay(1000);
18     ant_run_back(2,7); //Backwards
19     delay(1000);
20     ant_run_left(4,7); //Forward to the left
21     delay(1000);
22     ant_run_right(4,7); //Forward to the right
23     delay(1000);
24     ant_run_situ_left(2,7); //Turn left in situ
25     delay(1000);
26     ant_run_situ_right(2,7); //Turn right in situ
27     delay(1000);
28 }
```

Calibrate the joints of the ant robot

When robot ant is powered, robot legs may be not in ideal position. You can adjust the joint by modify parameters of function Ant_Correct_Angle(x1, x2, x3) to modify.



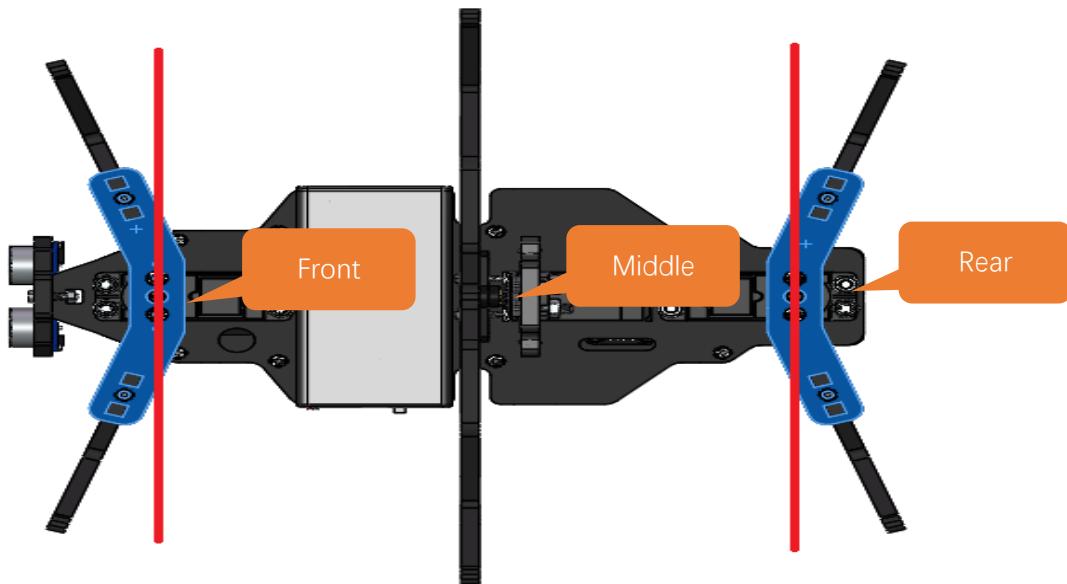
```

01.1.Servo_Test myServo.cpp myServo.h
7 #include "myServo.h"
8
9 void setup() {
10     Ant_Setup();
11     Ant_Correct_Angle(-5, 5, -5); //Call this function to microcalibrate the ant robot
12     delay(2000);
13 }

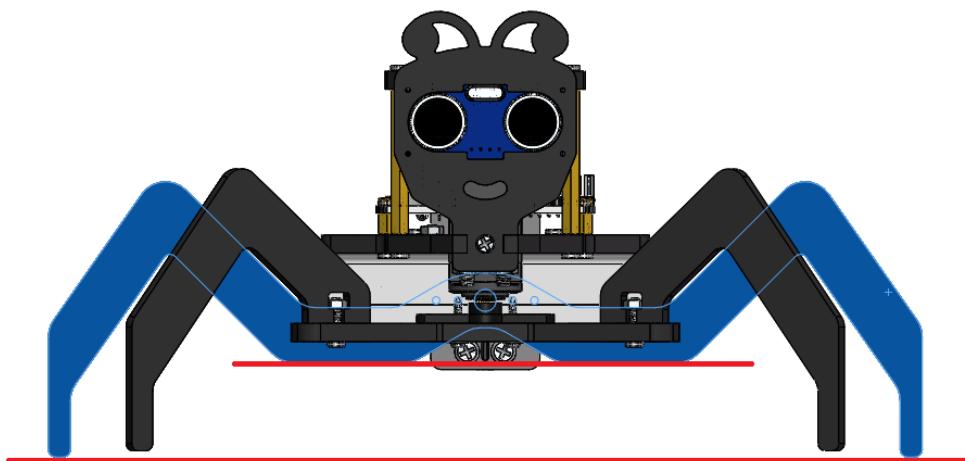
```

Modify these 3 parameters to adjust the ant robot.

Modify the first and third parameters of the Ant_Correct_Angle(x1, x2, x3) function, to make the front and rear legs of the ant robot parallel to the red line.

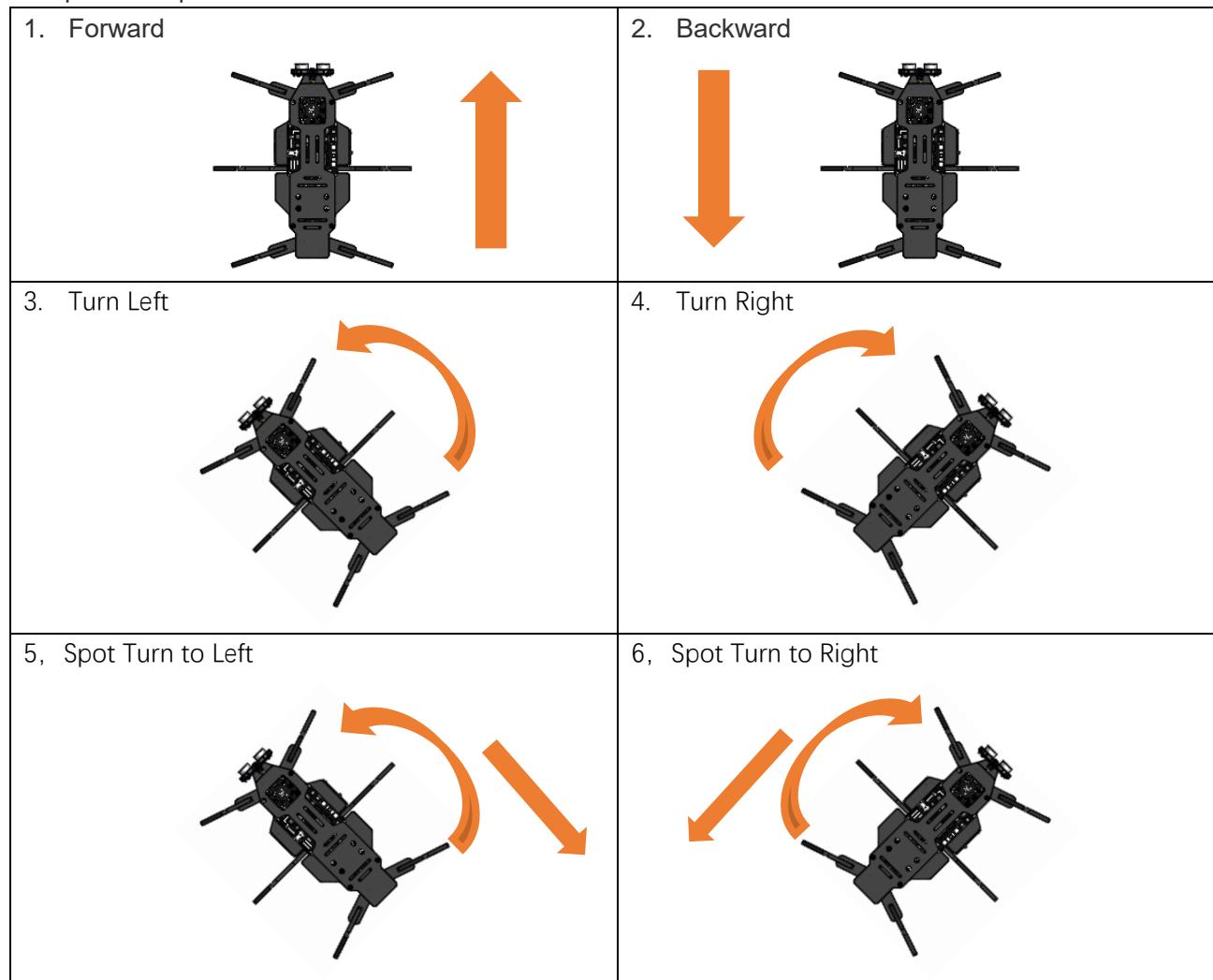


Modify the second parameter of the Ant_Correct_Angle(x1, x2, x3) function, to make the middle leg of the ant robot parallel to the red line.



The joints in later projects may also need to be adjusted.

Compile and upload code to control board. The movement of the ant will be as follows:



Code:

```

1 #include "myServo.h"
2
3 void setup() {
4     Ant_Setup();
5     delay(2000);
6 }
7
8 void loop() {
9     ant_run_forth(2,7); //Forward
10    delay(1000);
11    ant_run_back(2,7); //Backwards
12    delay(1000);
13    ant_run_left(4,7); //Forward to the left
14    delay(1000);
15    ant_run_right(4,7); //Forward to the right
16    delay(1000);
17    ant_run_situ_left(2,7); //Turn left in situ

```

Any concerns? ✉ support@freenove.com

```

18   delay(1000);
19   ant_run_situ_right(2, 7); //Turn right in situ
20   delay(1000);
21 }
```

Explanation of Code

Include the header file of library function, which makes it easier to call the program.

```
1 #include "myServo.h"
```

In the Initialization function setup(), call Ant_Setup() function to initialize the servos. Delay for 2s after initialization.

```

3 void setup() {
4     Ant_Setup();
5     delay(2000);
6 }
```

In the loop() function, the motion function of the robot is constantly called to control it to make different actions.

```

8 void loop() {
9     ant_run_forth(2, 7); //Forward
10    delay(1000);
11    ant_run_back(2, 7); //Backwards
12    delay(1000);
13    ant_run_left(4, 7); //Forward to the left
14    delay(1000);
15    ant_run_right(4, 7); //Forward to the right
16    delay(1000);
17    ant_run_situ_left(2, 7); //Turn left in situ
18    delay(1000);
19    ant_run_situ_right(2, 7); //Turn right in situ
20    delay(1000);
21 }
```

To facilitate users to learn, we have written some driving functions for servos to help our users learn how to control robot ant more quickly.

Below are the contents of myServo.h.

Reference

```
void Ant_Setup(void)
```

Ant_Setup() function is used to initialize the pins of servos so that they can control the servos. When initializing control board, please include it to initialization code; Otherwise, it won't be able to control servos.

```
void Ant_Correct_Angle(int angle1, int angle2, int angle3)
```

Ant_Correct_Angle() function is used to correct the installation error of the servos. After we install the robot ant, its legs may not be exactly at the middle, so we can use this function to fine-tune the program. When initializing control board, please include it to initialization code.

Angle1: Adjust Servo1. Recommended value: -5 to +5.

Angle2: Adjust Servo2. Recommended value: -5 to +5.

Angle3: Adjust Servo3. Recommended value: -5 to +5.

```
void ant_run_forth(int times, int mdelayms);
void ant_run_back(int times, int mdelayms);
void ant_run_left(int times, int mdelayms);
void ant_run_right(int times, int mdelayms);
void ant_run_situ_left(int times, int mdelayms);
void ant_run_situ_right(int times, int mdelayms);
```

These functions are used to control the robot's movement. They are blocking functions that can only be ended by interruption function.

ant_run_forth(): Controls the robot to crawl forward

ant_run_back(): Controls the robot to crawl backward

ant_run_left(): Controls the robot to crawl forward left

ant_run_right(): Controls the robot to crawl forward right

ant_run_situ_left(): Controls the robot to make spot turn to left.

ant_run_situ_right(): Controls the robot to make spot turn to right.

times: Indicates the times that the robot moves

mdelayms: Indicates the speed of the robot. The bigger the value is, the slower the robot moves.

```
void ant_reset_angle(void);
bool ant_move_forth(int times, int mdelayms, int state = 1);
bool ant_move_back(int times, int mdelayms, int state = 1);
bool ant_move_left(int times, int mdelayms, int state = 1);
bool ant_move_right(int times, int mdelayms, int state = 1);
bool ant_situ_left(int times, int mdelayms, int state = 1);
bool ant_situ_right(int times, int mdelayms, int state = 1);
```

These functions are also used to control the robot's movement. But they are non-blocking functions that can only be ended at any time.

ant_move_forth(): Controls the robot to crawl forward

ant_move_back(): Controls the robot to crawl backward

ant_move_left(): Controls the robot to crawl forward left

ant_move_right(): Controls the robot to crawl forward right

ant_situ_left(): Controls the robot to make spot turn to left.

ant_situ_right(): Controls the robot to make spot turn to right.

times: Indicates the times that the robot moves

mdelayms: Indicates the speed of the robot. The bigger the value is, the slower the robot moves.

state: If state=1, then the robot keeps crawling and times is invalid. If state=0, then the robot will crawl for "times" times and then stops.

```
void Ant_Servo_Mode(int mode, int speed=1, bool state=1); //Set the movement mode of the ant
```

Ant_Servo_Mode() function is a further package of the above seven non-blocking functions.

mode: 0-ant_reset_angle(), 1-ant_move_forth(), 2-ant_move_back(), 3-ant_move_left(),
4-ant_move_right(), 5-ant_situ_left(), 6-ant_situ_right()

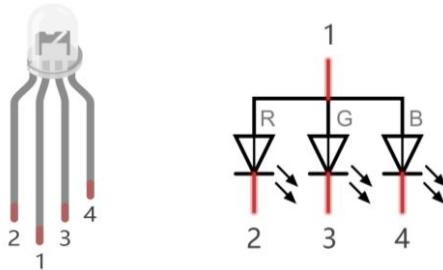
speed: 0-fast, 1-intermediate, 2-slow

state: If state=1, then the robot keeps crawling and times is invalid. If state=0, then the robot will crawl for "times" times and then stops.

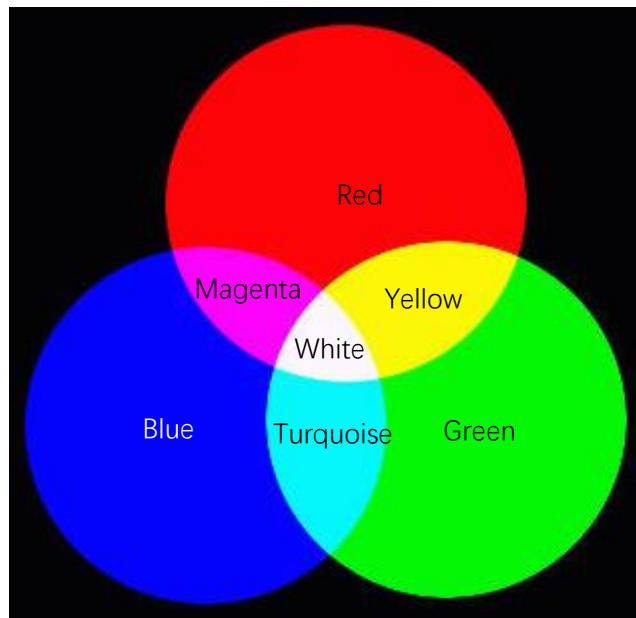
Chapter2 WS2812 Module Test

Component Knowledge

An RGB LED has 3 LEDs integrated into one LED component. It can respectively emit Red, Green and Blue light. In order to do this, it requires 4 pins (this is also how you identify it). The long pin (1) is the common which is the Anode (+) or positive lead, the other 3 are the Cathodes (-) or negative leads. A rendering of an RGB LED and its electronic symbol are shown below. We can make RGB LED emit various colors of light and brightness by controlling the 3 Cathodes (2, 3 & 4) of the RGB LED.



Red, Green, and Blue light are called 3 Primary Colors when discussing light (Note: for pigments such as paints, the 3 Primary Colors are Red, Blue and Yellow). When you combine these three Primary Colors of light with varied brightness, they can produce almost any color of visible light. Computer screens, single pixels of cell phone screens, neon lamps, etc. can all produce millions of colors due to this phenomenon.



We know from previous section that, control board controls LEDs to emit a total of 256(0-255) different brightness. So, through the combination of three different colors of LEDs, RGB LED can emit $256^3=16777216$ Colors, 16Million colors.

Circuit

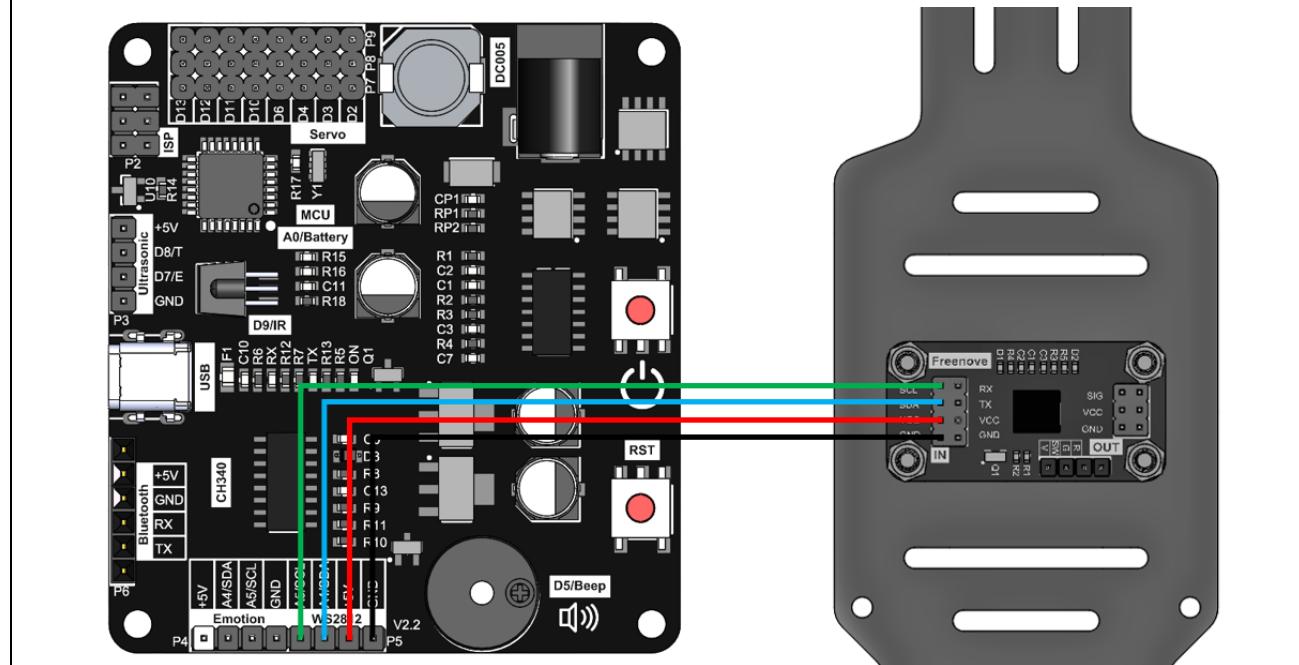
Schematic diagram

RGB LED Control Module pins		Main control chip pins	

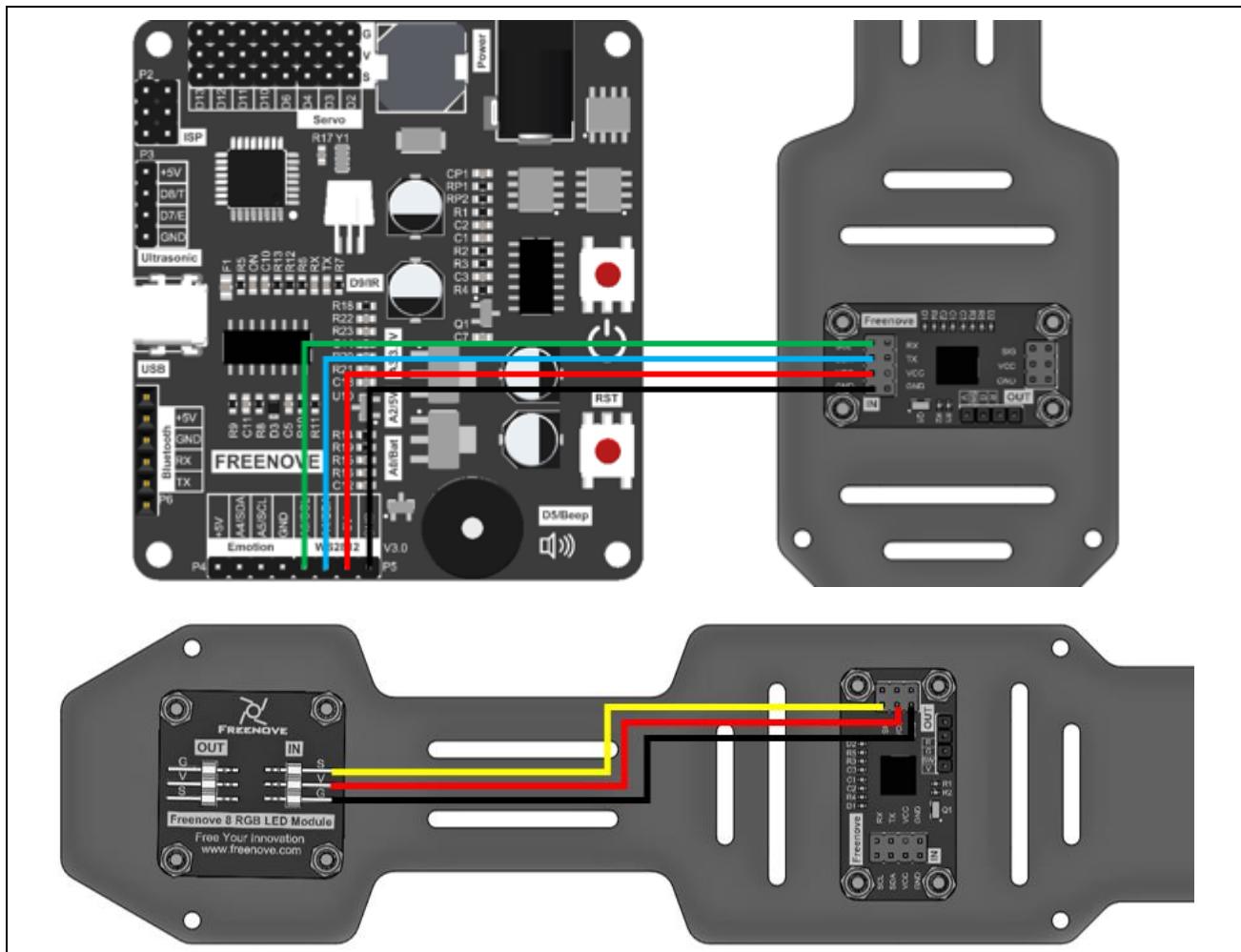
Hardware connection.

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

Wiring of RGB LED module to the control board



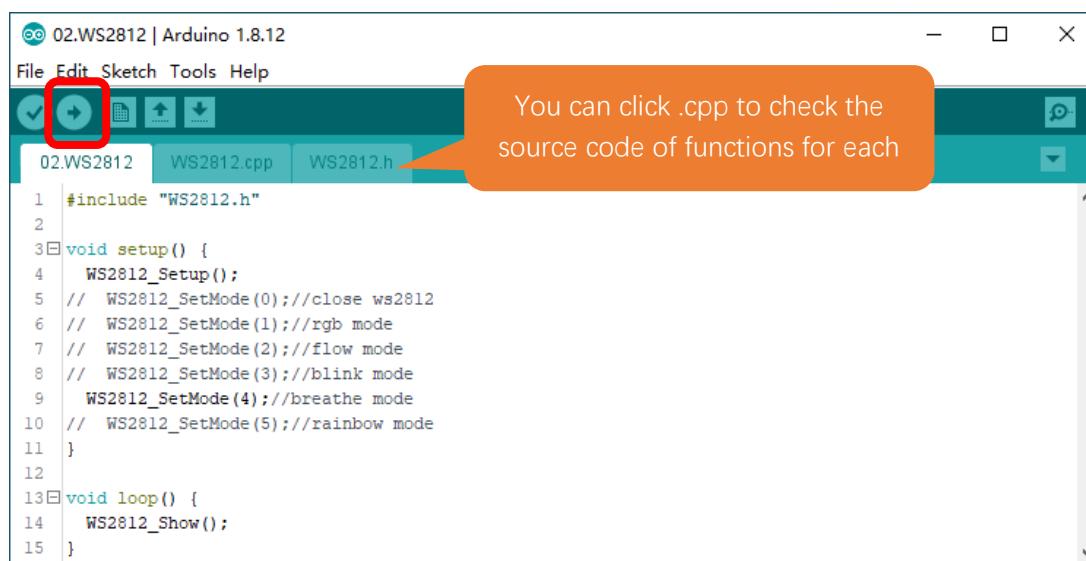
Any concerns? [✉ support@freenove.com](mailto:support@freenove.com)



Sketch

Open the 02.WS2812.ino in **Freenove_Robot_Ant_Kit\Sketches\02.WS2812**.

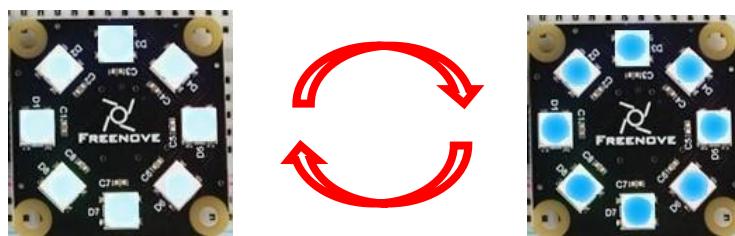
If you are interested in the realization of functions in WS2812.h file, you can click on WS2812.cpp to check.



```

02.WS2812 | Arduino 1.8.12
File Edit Sketch Tools Help
02.WS2812 WS2812.cpp WS2812.h
1 #include "WS2812.h"
2
3 void setup() {
4     WS2812_Setup();
5     // WS2812_SetMode(0); //close ws2812
6     // WS2812_SetMode(1); //rgb mode
7     // WS2812_SetMode(2); //flow mode
8     // WS2812_SetMode(3); //blink mode
9     WS2812_SetMode(4); //breath mode
10    // WS2812_SetMode(5); //rainbow mode
11 }
12
13 void loop() {
14     WS2812_Show();
15 }
```

Compile and upload code and the WS2812 LED module will gradually emit blue light and then OFF gradually. This process will repeat over and over again.



Code:

```

1 #include "WS2812.h"
2
3 void setup() {
4     WS2812_Setup();
5     // WS2812_SetMode(0); //close ws2812
6     // WS2812_SetMode(1); //rgb mode
7     // WS2812_SetMode(2); //flow mode
8     // WS2812_SetMode(3); //blink mode
9     WS2812_SetMode(4); //breath mode
10    // WS2812_SetMode(5); //rainbow mode
11 }
12 void loop() {
13     WS2812_Show();
14 }
```

Explanation of Code

Include the header file of library function, which makes it easier to call the program.

Any concerns? ✉ support@freenove.com

```
1 #include "WS2812.h"
```

WS2812_Setup() function is called to initialize the LED control module and set the default color to blue.

```
4 WS2812_Setup();
```

WS2812_SetMode() function is called to set the display mode to breathing light mode.

```
9 WS2812_SetMode(4);
```

In the main loop, WS2812_Show() is called constantly to make the module work.

```
13 WS2812_Show();
```

Reference

```
void WS2812_Setup(void)
```

WS2812_Setup() function is used to initialize RGB LED moduel. When initializing control board, please include it to the initialization code; Otherwise, it will fail to have the module display any color.

```
void WS2812_Set_Color(unsigned char color_1,unsigned char color_2,unsigned char color_3);
```

WS2812_Set_Color() function is used to set the color of RGB LED module.

color_1: Brightness of red. Range: 0-255.

color_2: Brightness of green. Range: 0-255.

color_3: Brightness of blue. Range: 0-255.

```
void WS2812_Set_LED(bool red=0,bool green=0,bool blue=0,int brightness=20);
```

WS2812_Set_LED() function is used to set the color of the LED module.

red: If red=1, emits red light; if red=0, red light OFF.

green: If green=1, emits green light; if green=0, green light OFF.

blue: If blue=1, emits blue light; if blue=0, blue light OFF.

Brightness: Set brightness of the LED. Range: 0-255.

```
void ws2812_close(void);
void ws2812_flow(void);
void ws2812_rgb(void);
void ws2812_blink(void);
void ws2812_breathe(void);
void ws2812_rainbow(void);
```

Above are 6 non-blcoking function controlling the LED moduel, which are called through WS2812_Show().

And we use WS2812_SetMode() to determine the mode and use WS2812_Set_Color() to set the display color.

ws2812_close(): Turn OFF the LED module.

ws2812_flow(): Flowing Water Light Mode

ws2812_rgb(): Light Always-ON Mode

ws2812_blink(): Blinking Light Mode

ws2812_breathe(): Breathing Light Mode

ws2812_rainbow(): Rainbow Light Mode

```
void WS2812_Show(void);
```

WS2812_Show() function is used to send command to LED control module to control the LED module to display colors.

```
void WS2812_SetMode(int mode);
```

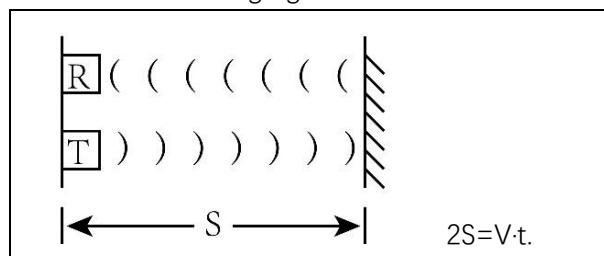
The function to set the display mode of the LED module. Based on different parameter, the LED module can run on different mode.

Mode:0-LED OFF; 1-Light Always-ON; 2-Flowing Water Light; 3- Blinking; 4- Breathing; 5- Rainbow Light

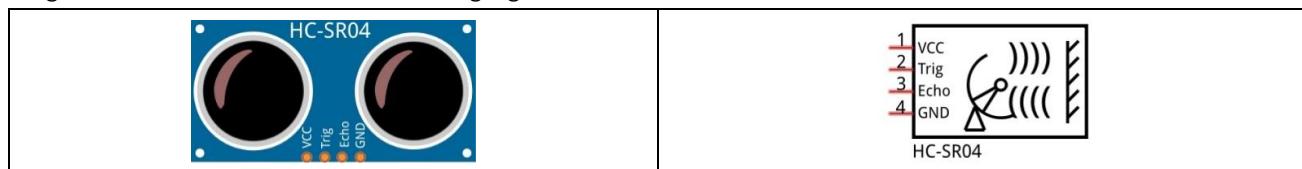
Chapter3 Ultrasonic Module Test

Component Knowledge

The ultrasonic ranging module uses the principle that ultrasonic waves will be sent back when it detects 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 HC-SR04 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 HC-SR04 ultrasonic ranging module are shown below:



Pin description:

Pin	Description
VCC	power supply pin
Trig	trigger pin
Echo	Echo pin
GND	GND

Technical specs:

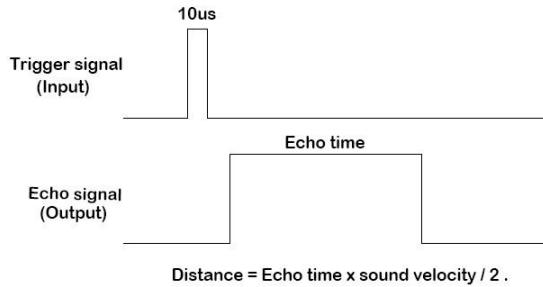
Working voltage: 5V

Working current: 12mA

Minimum measured distance: 2cm

Maximum measured distance: 200cm

Instructions for use: output a high-level pulse in Trig pin lasting for at 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, $s=vt/2$.



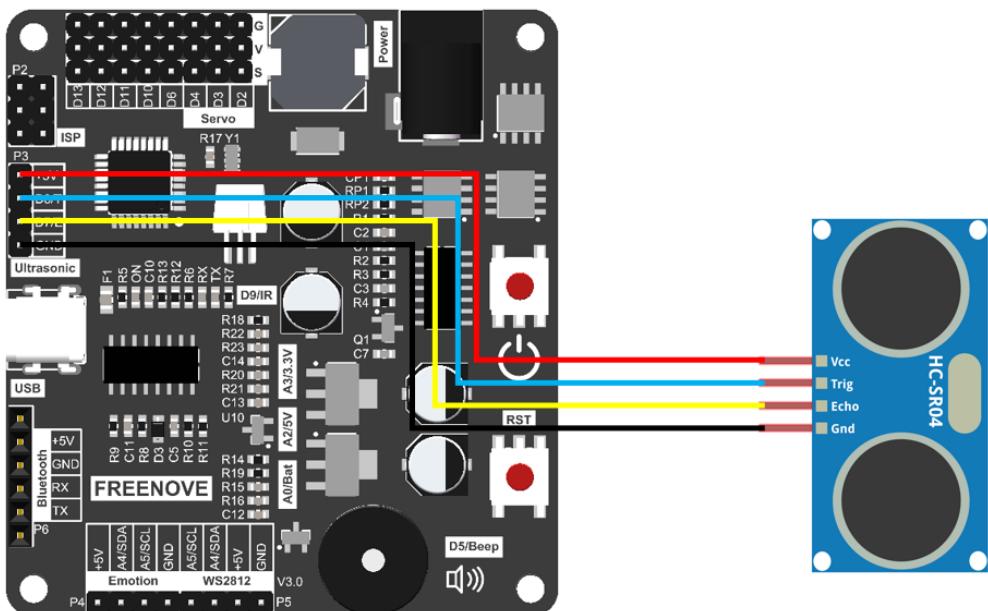
Circuit

Schematic diagram

Ultrasinnic Module Pins	Main control chip pins																																																																												
<p>P3</p> <p>+5V</p> <p>1 2 D8/Trig</p> <p>3 D7/Echo</p> <p>4 GND</p> <p>Header1x4Pin</p>	<table border="1"> <tr> <td>MCU_RXD</td> <td>30</td> <td>PC0/ADC0</td> <td>23 A0</td> </tr> <tr> <td>MCU_TXD</td> <td>31</td> <td>PC1/ADC1</td> <td>24 A1</td> </tr> <tr> <td>D2/Servo1</td> <td>32</td> <td>PC2/ADC2</td> <td>25 A2</td> </tr> <tr> <td>D3/Servo2</td> <td>1</td> <td>PC3/ADC3</td> <td>26 A3</td> </tr> <tr> <td>D4/Servo3</td> <td>2</td> <td>PC4/ADC4/SDA</td> <td>27 A4/SDA</td> </tr> <tr> <td>D5/Beep</td> <td>9</td> <td>PC5/ADC5/SCL</td> <td>28 A5/SCL</td> </tr> <tr> <td>D6</td> <td>10</td> <td></td> <td></td> </tr> <tr> <td>D7/Echo</td> <td>11</td> <td></td> <td></td> </tr> <tr> <td>D8/Trig</td> <td>12</td> <td>ADC6</td> <td>19</td> </tr> <tr> <td>D9/IR</td> <td>13</td> <td>ADC7</td> <td>22</td> </tr> <tr> <td>D10</td> <td>14</td> <td></td> <td></td> </tr> <tr> <td>D11</td> <td>15</td> <td>VCC</td> <td>4</td> </tr> <tr> <td>D12</td> <td>16</td> <td>VCC</td> <td>6</td> </tr> <tr> <td>D13</td> <td>17</td> <td>AVCC</td> <td>18</td> </tr> <tr> <td>XTAL1</td> <td>7</td> <td>RESET</td> <td>29</td> </tr> <tr> <td>XTAL2</td> <td>8</td> <td>AREF</td> <td>20</td> </tr> <tr> <td></td> <td></td> <td>GND</td> <td>3</td> </tr> <tr> <td></td> <td></td> <td>GND</td> <td>5</td> </tr> <tr> <td></td> <td></td> <td>GND</td> <td>21</td> </tr> </table> <p>ATMega328P-AU</p>	MCU_RXD	30	PC0/ADC0	23 A0	MCU_TXD	31	PC1/ADC1	24 A1	D2/Servo1	32	PC2/ADC2	25 A2	D3/Servo2	1	PC3/ADC3	26 A3	D4/Servo3	2	PC4/ADC4/SDA	27 A4/SDA	D5/Beep	9	PC5/ADC5/SCL	28 A5/SCL	D6	10			D7/Echo	11			D8/Trig	12	ADC6	19	D9/IR	13	ADC7	22	D10	14			D11	15	VCC	4	D12	16	VCC	6	D13	17	AVCC	18	XTAL1	7	RESET	29	XTAL2	8	AREF	20			GND	3			GND	5			GND	21
MCU_RXD	30	PC0/ADC0	23 A0																																																																										
MCU_TXD	31	PC1/ADC1	24 A1																																																																										
D2/Servo1	32	PC2/ADC2	25 A2																																																																										
D3/Servo2	1	PC3/ADC3	26 A3																																																																										
D4/Servo3	2	PC4/ADC4/SDA	27 A4/SDA																																																																										
D5/Beep	9	PC5/ADC5/SCL	28 A5/SCL																																																																										
D6	10																																																																												
D7/Echo	11																																																																												
D8/Trig	12	ADC6	19																																																																										
D9/IR	13	ADC7	22																																																																										
D10	14																																																																												
D11	15	VCC	4																																																																										
D12	16	VCC	6																																																																										
D13	17	AVCC	18																																																																										
XTAL1	7	RESET	29																																																																										
XTAL2	8	AREF	20																																																																										
		GND	3																																																																										
		GND	5																																																																										
		GND	21																																																																										

Hardware connection. If you need any support, please feel free to contact us via: support@freenove.com

Wiring of ultrasonic module to the control board



Any concerns? [✉ support@freenove.com](mailto:support@freenove.com)

Sketch

Open 03.UltrasonicSensor.ino in **Freenove_Robot_Ant_Kit\Sketches\03.UltrasonicSensor**.

If you are interested in the realization of functions in Ultrasonic.h file, you can click on Ultrasonic.cpp to check.

The screenshot shows the Arduino IDE interface with the title bar "03.UltrasonicSensor | Arduino 1.8.12". The menu bar includes File, Edit, Sketch, Tools, and Help. Below the menu is a toolbar with icons for back, forward, and other functions. The tabs at the top are "03.UltrasonicSensor", "Ultrasonic.cpp" (which is selected and highlighted in blue), and "Ultrasonic.h". The main code editor area contains the following C++ code:

```

1 #include "Ultrasonic.h"
2
3 void setup() {
4     Serial.begin(9600);
5     Ultrasonic_Setup(); //Ultrasonic initialization
6 }
7
8 void loop() {
9     Ultrasonic_Get_Data(); //Ultrasonic get distance value
10}

```

Below the code editor is a black terminal window displaying the message "avrduude done. Thank you.". The status bar at the bottom right shows "Arduino Uno on COM3". A callout bubble with the text "You can click .cpp to check the source code of functions for each" points to the Ultrasonic.cpp tab.

Compile and upload the code. When the program is uploaded successfully, open the serial monitor, set baud rate to 9600, and you can observe that the control board receives distance messages from the ultrasonic module every 1s and print them on the screen in centimeter

The screenshot shows the Arduino Serial Monitor window titled "COM3". The text input field is empty, and the "Send" button is visible. The main text area displays a series of distance measurements in centimeters, each preceded by a timestamp and followed by a right arrow and the distance value:

```

19:16:32.804 -> E#44
19:16:33.794 -> E#44
19:16:34.786 -> E#45
19:16:35.811 -> E#44
19:16:36.798 -> E#45
19:16:37.793 -> E#44
19:16:38.782 -> E#45
19:16:39.812 -> E#43
19:16:40.800 -> E#43
19:16:41.789 -> E#42
19:16:42.782 -> E#41
19:16:43.803 -> E#43

```

At the bottom, there are several configuration options: "Autoscroll" (checked), "Show timestamp" (checked), "Newline" (dropdown menu), "9600 baud" (highlighted with a red box), and "Clear output".

Code:

```

1 #include "Ultrasonic.h"

2

3 void setup() {
4     Serial.begin(9600);
5     Ultrasonic_Setup(); //Ultrasonic initialization
6 }

7

8 void loop() {
9     Ultrasonic_Get_Data(); //Ultrasonic get distance value
10}

```

Explanation of Code

Include the header file of library function, which makes it easier to call the program.

```
1 #include "Ultrasonic.h"
```

Initialize the serial communication function of the control board and set baud rate to 9600.

```
4 Serial.begin(9600);
```

Initialize the ultrasonic module.

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

Obtain the distance messages from ultrasonic module every 1s and print them through serial port. 每

```
9 Ultrasonic_Get_Data(); //Ultrasonic get distance value
```

Reference

```
void Ultrasonic_Setup(void);
```

Ultrasonic_Setup() function is used to initialize ultrasonic module.

```
int Ultrasonic_Get_Data(int delayms=1000);
```

By default, control board receives the distance messages from ultrasonic module every 1s and stores them to Ultrasonic_Value and print them through serial port.

Delayms: Every delayms millisecond, distance information is obtained from ultrasonic module. Range: 0-32767

```
void Ultrasonic_Bluetooth_Data(void)
```

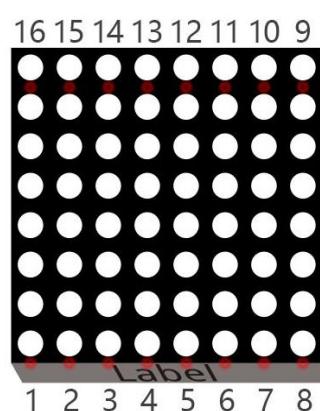
Obtain information from ultrasonic module and send it through serial port.(applied when receiving commands from Bluetooth module)

Chapter4 Expression module Test

Component Knowledge

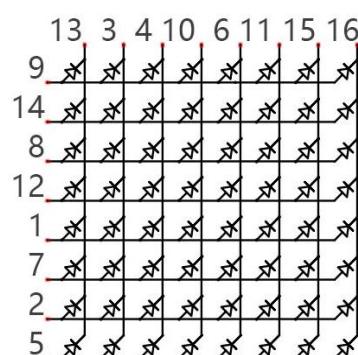
LED Matrix

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).

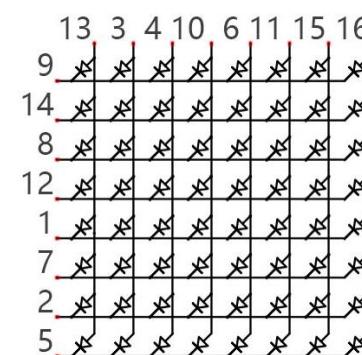


In order to facilitate the operation and reduce the number of ports required to drive this component, the positive poles of the LEDs in each row and negative poles of the LEDs in each column are respectively connected together inside the LED matrix module, which is called a common anode. There is another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

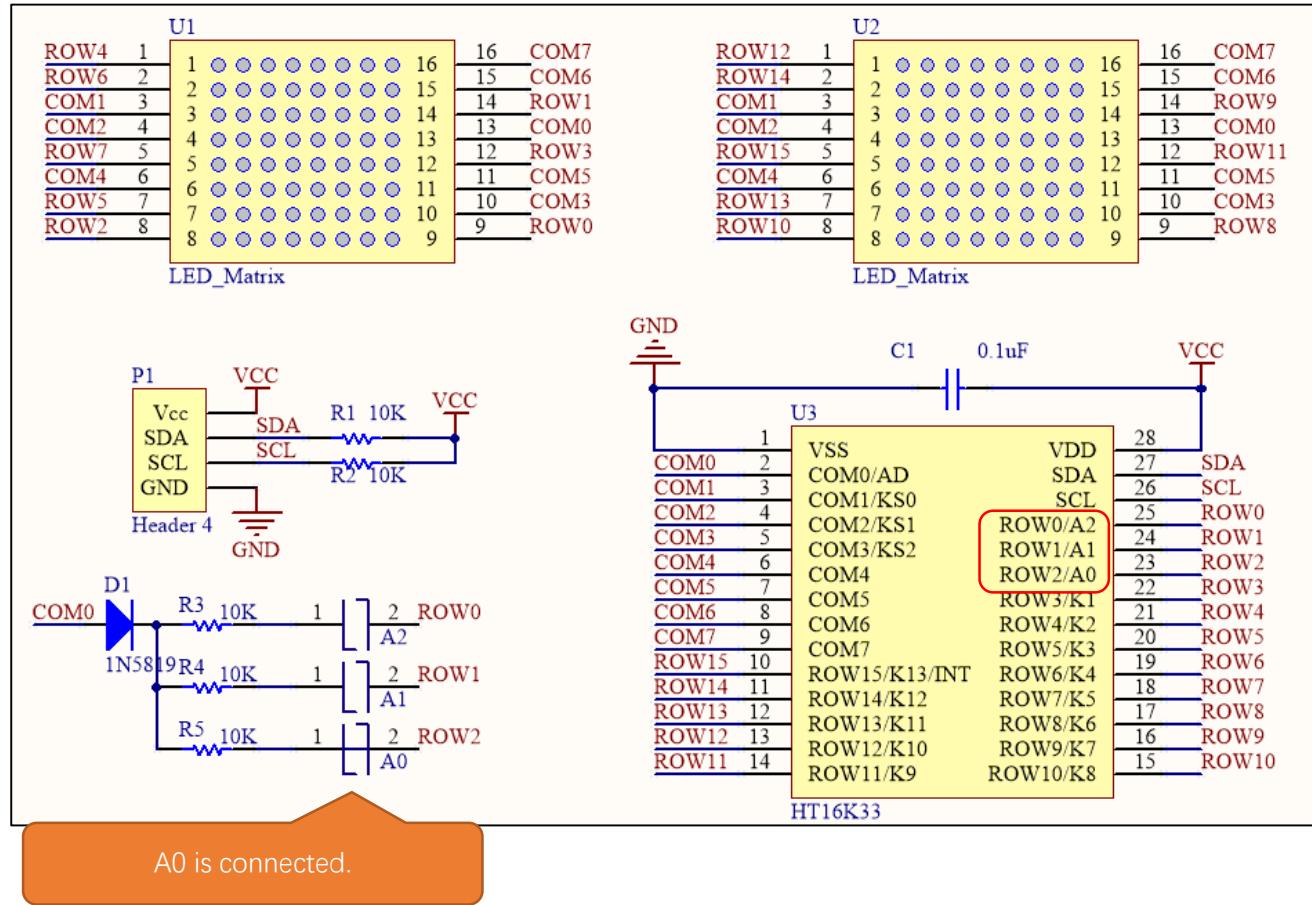


Connection mode of common cathode



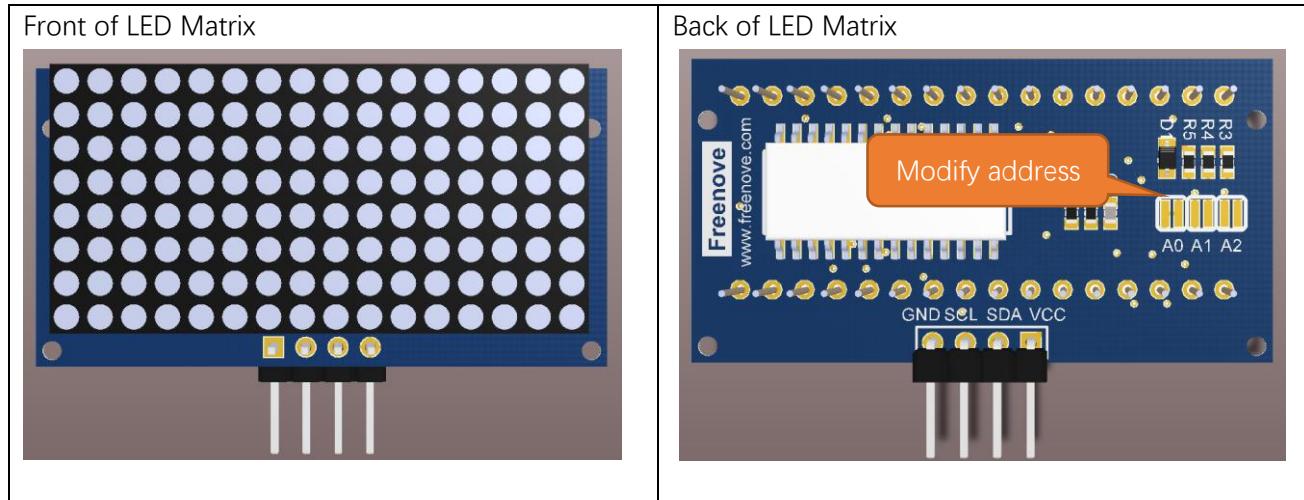
Schematic

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

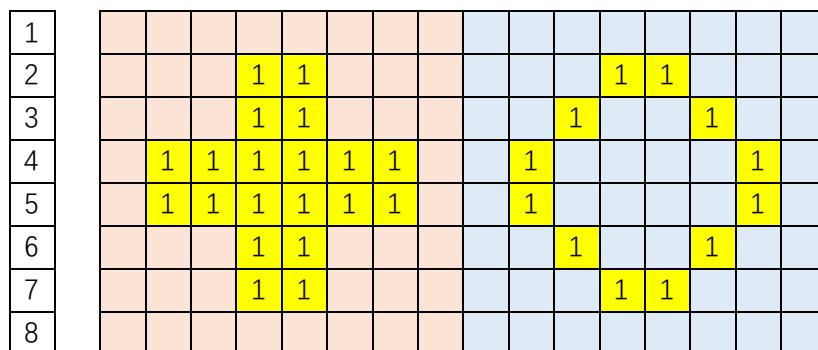


A0 is connected.

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.



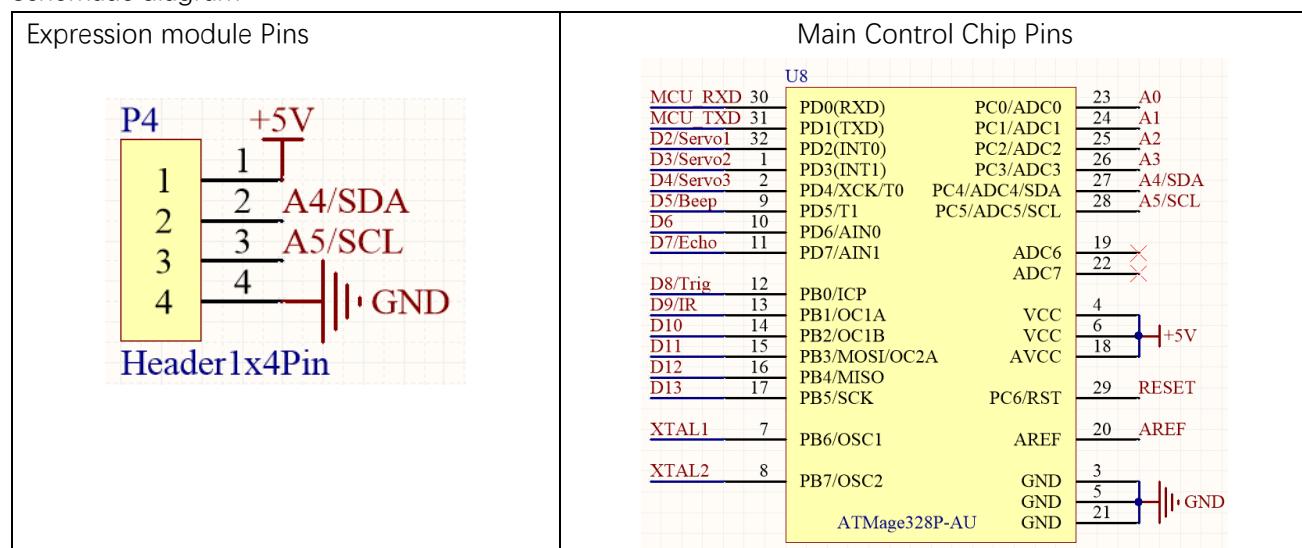
Below, the table on the left corresponds to the "+" above, and the table on the right corresponds to the "o" above.

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

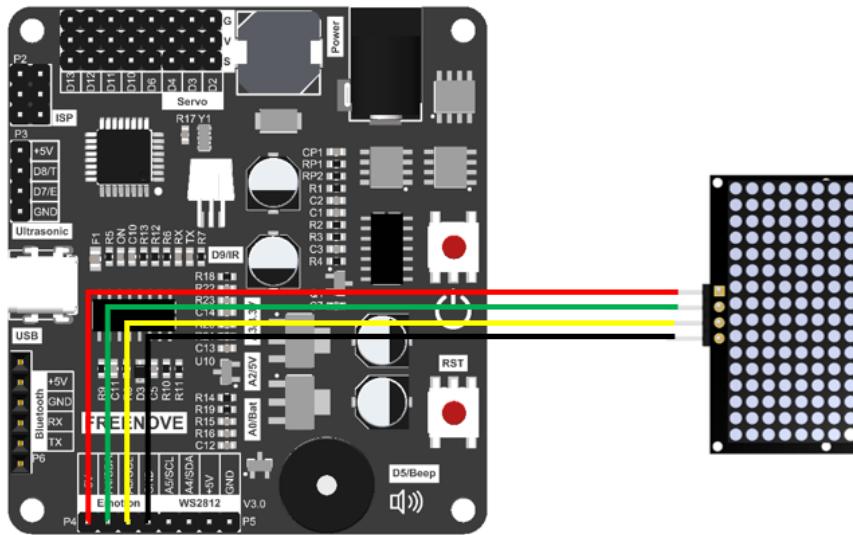
Circuit

Schematic diagram



Hardware connection. If you need any support, please feel free to contact us via: support@freenove.com

Wiring of expression module to control board



Sketch

Open 04.Emotion.ino in **Freenove_Robot_Ant_Kit\Sketches\04.Emotion**.

If you are interested in the realization of functions in Emotion.h file, you can click on Emotion.cpp to check.

You can click .cpp to check the source code of functions for each

```

04.Emotion | Arduino 1.8.12
File Edit Sketch Tools Help
04.Emotion Emotion.cpp Emotion.h
1 #include "Emotion.h"
2
3 void setup() {
4     Emotion_Setup(); //Initializes the Led Matrix
5     /*
6         * Set the emotion show mode: 0-Display off,
7         * 1-Turn the eyes,2-cry eyes,3-smile,4-right-wheel,
8         * 5-left-wheel,6-blink,7-rand static emotion,8-set static emotion
9         */
10    Emotion_SetMode (2);
11 }
12
13 void loop() {
14     Emotion_Show(); //Display
15 }
```

Done uploading.
avrdude done. Thank you.

10 Arduino Uno on COM3

Any concerns? [✉ support@freenove.com](mailto:support@freenove.com)

Compile and upload the code, you will see the expression module dropping tears.



Code:

```

1 #include "Emotion.h"
2 void setup() {
3     Emotion_Setup();    //Initializes the Led Matrix
4     /*
5      * Set the emotion show mode: 0-Display off,
6      * 1-Turn the eyes, 2-cry eyes, 3-smile, 4-right-wheel,
7      * 5-left-wheel, 6-blink, 7-rand static emotion, 8-set static emotion
8      */
9     Emotion_SetMode(2);
10 }
11 void loop() {
12     Emotion_Show();    //Display
13 }
```

Explanation of Code

Include the header file of library function, which makes it easier to call the program.

```
1 #include "Emotion.h"
```

Emotion_Setup() function is used to initialize expression module.

```
3 Emotion_Setup();    //Initializes the Led Matrix
```

Emotion_SetMode() is called to setup exression module to display expressions.

```
9 Emotion_SetMode(2);
```

In main loop, Emotion_Show() is constantly called to get expression module work.

```
12 Emotion_Show();
```

Reference

```
void Emotion_Setup(int address = 0x71);
```

Emotion_Setup() function is used to initialize expression module. When initializing expression module, include it to initialization code; Otherwise, the module will fail to display expressions

```
void Emotion_Show(void);
```

Emotion_Show() function is used to send data to expression module so that the control board can control it to display expressions.

```
void Emotion_SetMode(int mode, int static_emotion = 0);
```

Set expression mode.

Mode: 0- Expression module OFF; 1-Eyeballs rotating; 2-Dropping tears; 3- smiling; 4-Clockwise Windmill; 5-Counterclockwise Windmill; 6-Blinking; 7-Random static expressions; 8- Designated static expression, needs to be combined with the use of static_emotion

static_emotion: parameter of static expressions, Range: 0-20. Only available when mode=8.

```
void clearEmtions(void);  
void eyesRotate(int delay_ms);  
void eyesBlink(int delay_ms);  
void eyesSmile(int delay_ms);  
void eyesCry(int delay_ms);  
void wheel(int mode, int delay_ms);  
void staticEmtions(int emotion);
```

These are 7 non-blocking function for expressions display which are called through Emotion_Show(). And we use Emotion_SetMode() function to select which mode to display.

clearEmtions(void): Clear the displaying content on the module

eyesRotate(int delay_ms): Rotate eyeballs. delay_ms refers to the speed the eyeballs rotates, the larger the number, the slower the speed.

eyesBlink(int delay_ms): Eyes blinking

eyesSmile(int delay_ms): Happy

eyesCry(int delay_ms): Sad

wheel(int mode, int delay_ms): Windmill rotating. If mode=1, rotate to left; If mode=2, rotate to right.

staticEmtions(int emotion): The function for static expressions displaying. The range of emotion is 0-20.

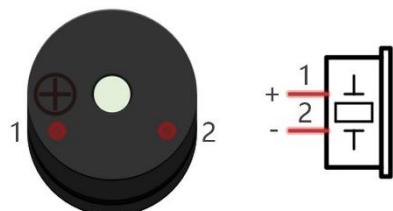
Chapter5 ADC and Buzzer Test

Component Knowledge

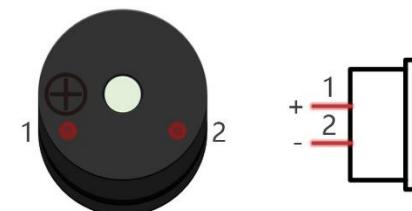
Buzzer

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

Active buzzer



Passive buzzer



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

Next, we will use an active buzzer to make a doorbell and a passive buzzer to make an alarm.

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



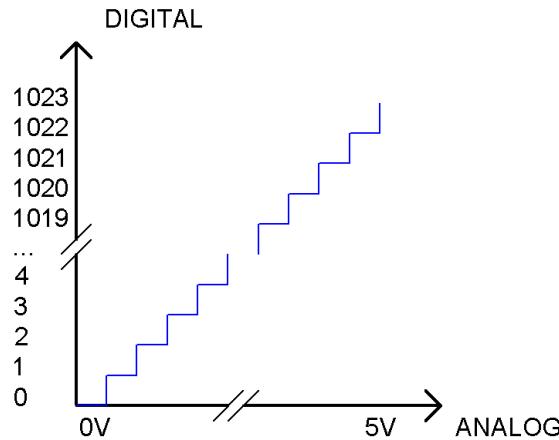
Passive buzzer



ADC

An 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 module is 10 bits, that means the resolution is $2^{10}=1024$, so that its range (at 5V) will be divided equally into 1024 parts.

Any analog value can be mapped to one digital value using the resolution of the converter. So the more bits the ADC has, the denser the partition of analog will be and the greater the precision of the resulting conversion.



Subsection 1: the analog in rang of 0V-5/1024V corresponds to digital 0;

Subsection 2: the analog in rang of 5 /1024V-2*5/1024V corresponds to digital 1;

The following analog signal will be divided accordingly.

$$ADC\ Value = \frac{\text{Analog\ Voltage}}{5.0} * 1023$$

Calculation of External Referenced Voltage

To get more accurate battery level, we use external referenced pins.

For more information of external reference, please visit:

<https://www.arduino.cc/reference/en/language/functions/analog-io/analogreference/>

Notes and Warnings

After changing the analog reference, the first few readings from `analogRead()` may not be accurate.

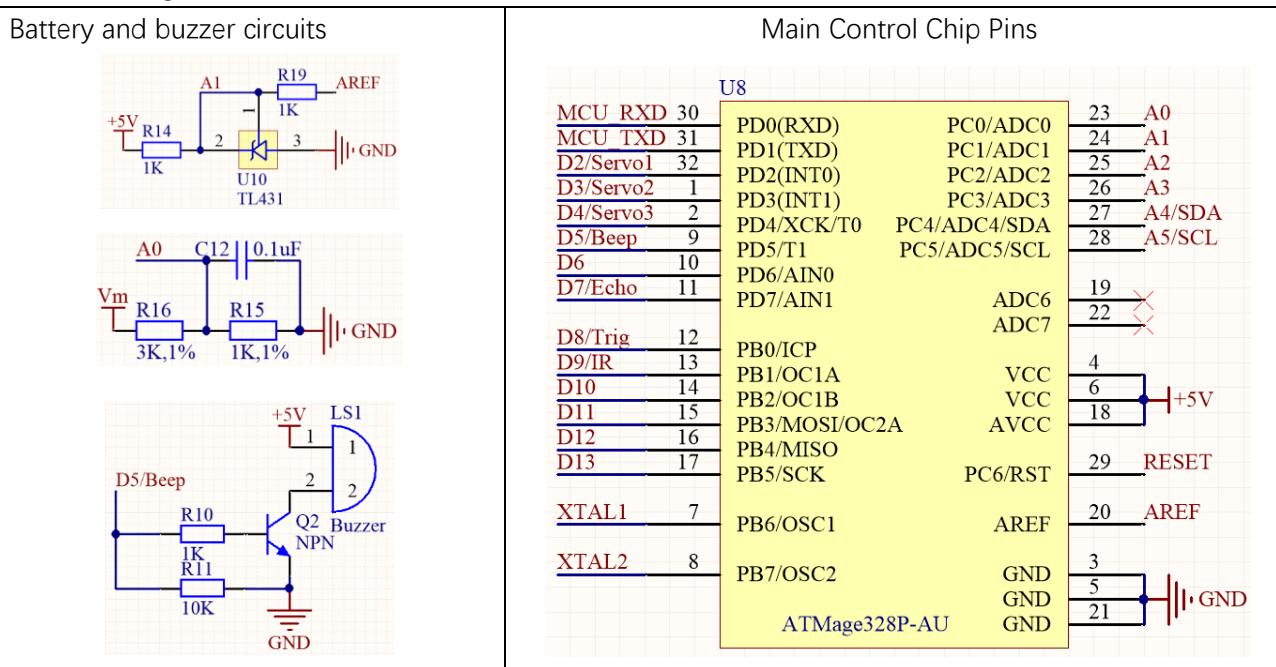
Don't use anything less than 0V or more than 5V for external reference voltage on the AREF pin! If you're using an external reference on the AREF pin, you must set the analog reference to EXTERNAL before calling `analogRead()`.

Otherwise, you will short together the active reference voltage (internally generated) and the AREF pin, possibly damaging the microcontroller on your Arduino board.

Alternatively, you can connect the external reference voltage to the AREF pin through a 5K resistor, allowing you to switch between external and internal reference voltages. Note that the resistor will alter the voltage that gets used as the reference because there is an internal 32K resistor on the AREF pin. The two act as a voltage divider, so, for example, 2.5V applied through the resistor will yield $2.5 * 32 / (32 + 5) = \sim 2.2V$ at the AREF pin.

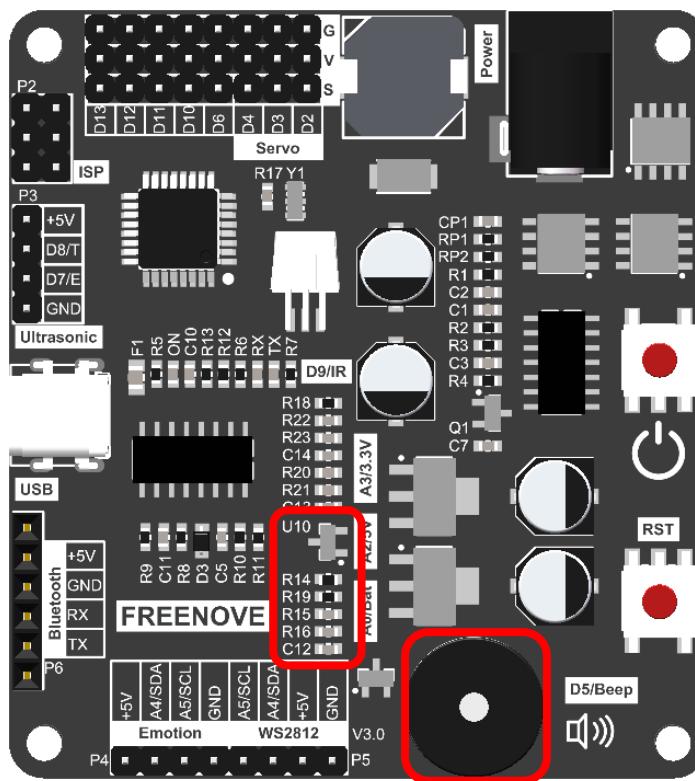
Circuit

Schematic diagram



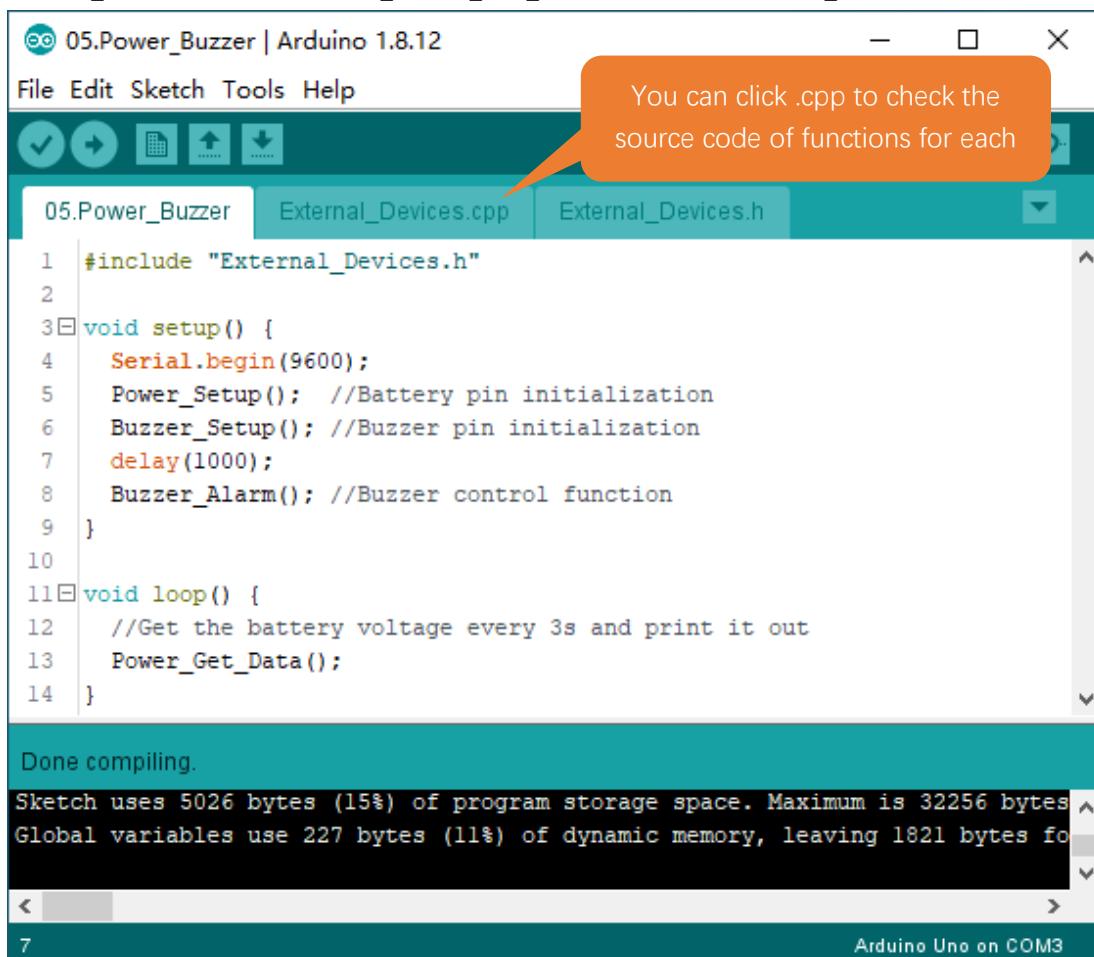
Hardware connection. If you need any support, please feel free to contact us via: support@freenove.com

The area where the circuits lie



Sketch

Open 05.Power_Buzzer.ino in Freenove_Robot_Ant_Kit\Sketches\05.Power_Buzzer.



You can click .cpp to check the source code of functions for each

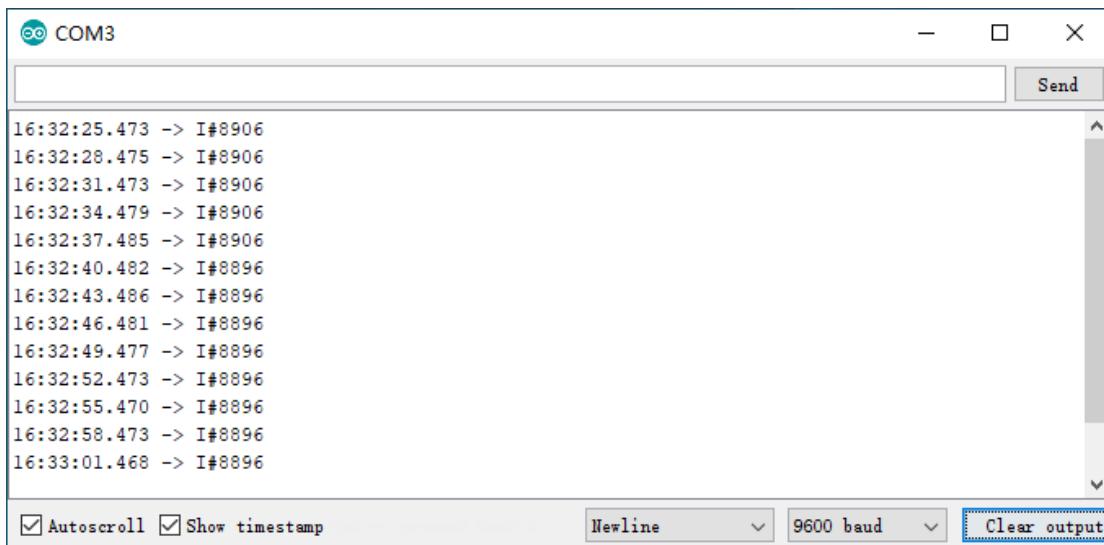
```

 05.Power_Buzzer | Arduino 1.8.12
File Edit Sketch Tools Help
5.0.0 External_Devices.cpp External_Devices.h
1 #include "External_Devices.h"
2
3 void setup() {
4     Serial.begin(9600);
5     Power_Setup(); //Battery pin initialization
6     Buzzer_Setup(); //Buzzer pin initialization
7     delay(1000);
8     Buzzer_Alarm(); //Buzzer control function
9 }
10
11 void loop() {
12     //Get the battery voltage every 3s and print it out
13     Power_Get_Data();
14 }
```

Done compiling.
Sketch uses 5026 bytes (15%) of program storage space. Maximum is 32256 bytes
Global variables use 227 bytes (11%) of dynamic memory, leaving 1821 bytes fo

Arduino Uno on COM3

Compile and upload code, open serial monitor, ser baud rate to 9600 and you will observe the main board obtains the voltage of batteries and print it through serial port after multiplying the number by 1000.



```

16:32:25.473 -> I#8906
16:32:28.475 -> I#8906
16:32:31.473 -> I#8906
16:32:34.479 -> I#8906
16:32:37.485 -> I#8906
16:32:40.482 -> I#8896
16:32:43.486 -> I#8896
16:32:46.481 -> I#8896
16:32:49.477 -> I#8896
16:32:52.473 -> I#8896
16:32:55.470 -> I#8896
16:32:58.473 -> I#8896
16:33:01.468 -> I#8896
```

Autoscroll Show timestamp Newline 9600 baud Clear output

Code:

```

1 #include "External_Devices.h"
2 void setup() {
3     Serial.begin(9600);
4     Power_Setup(); //Battery pin initialization
5     Buzzer_Setup(); //Buzzer pin initialization
6     delay(1000);
7     Buzzer_Alarm(); //Buzzer control function
8 }
9 void loop() {
10    //Get the battery voltage every 3s and print it out
11    Power_Get_Data();
12 }
```

Explanation of Code

Include the header file of library function, which makes it easier to call the program.

```
1 #include "External_Devices.h"
```

Power_Setup() is called to initialize ADC of control board and calculate external referenced voltage.

```
4 Power_Setup(); //Battery pin initialization
```

Buzzer_Setup() is called to initialize D5 of control board so that it can control buzzer to emit sound.

```
5 Buzzer_Setup(); //Buzzer pin initialization
```

Buzzer_Alarm() function is used to drive the buzzer. Each time when this function is called, the buzzer will sound at a frequency of 1k and last for 0.1s.

```
7 Buzzer_Alarm(); //Buzzer control function
```

In main loop, Power_Get_Data() is constantly called to have ADC obtain battery level data every 3s and print through serial port.

```
11 Power_Get_Data(); //Get the battery voltage every 3s and print it out
```

Reference

void Power_Setup(void);

Every time when this function is called, control board will switch referenced voltage pin mode to default mode and uses A1 pin to read ADC value and then get the external voltage of referenced voltage pin by calculating. And finally, it will set referenced voltage pin mode to external mode.

void Power_Correct(float value);

The function to set external voltage of referenced voltage pin. If you think that the external voltage value coefficient calculating automatically by control board is not accurate enough, you can use this function to reset the value to get better battery voltage feedback.

void Power_Bluetooth_Data(void);

The function for Bluetooth to get battery voltage, using with Bluetooth mode. When Bluetooth module receive checking commands, the function will be called to get battery voltage once and send it to Bluetooth module.

void Power_Get_Data(void);

The function to obtain battery voltage every 3s.

```
void Buzzer_Setup(void);
```

The function to initialize the pin that controls buzzer.

Before using buzzer, the function should be called to initialize the pin; otherwise it won't be able to drive the buzzer.

```
void Buzzer_Alarm(int frequency=1000, int msdelay=100, int times=1);
```

Function for buzzer to emit sound.

frequency: the frequency of buzzer, 1K Hz by default.

msdelay: the duration of buzzer sounding, 100ms by default.

times: times of buzzer sounding, one time by default

Chapter6 Infrared Sensor Test

Component Knowledge

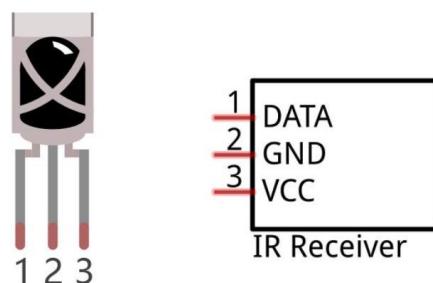
Infrared Remote

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.



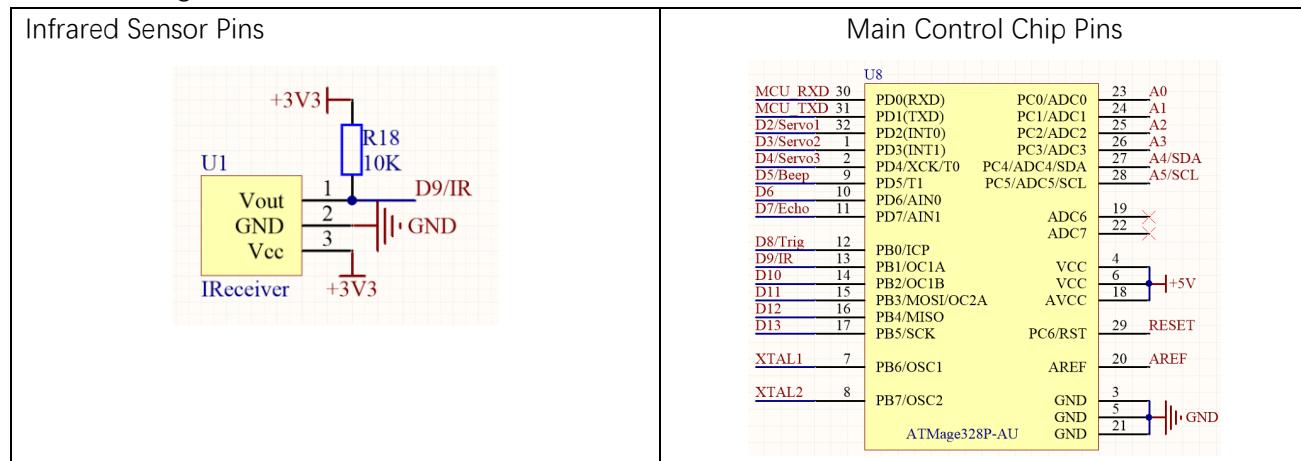
When using IR remote controller, it will send the value of the pressed key to the receiving circuit. We can program the control board so it can print the receiving key value through serial monitor.

The followings are the key values of each key

ICON	KEY Value	ICON	KEY Value
	FFA25D		FFB04F
	FFE21D		FF30CF
	FF22DD		FF18E7
	FF02FD		FF7A85
	FFC23D		FF10EF
	FFE01F		FF38C7
	FFA857		FF5AA5
	FF906F		FF42BD
	FF6897		FF4AB5
	FF9867		FF52AD

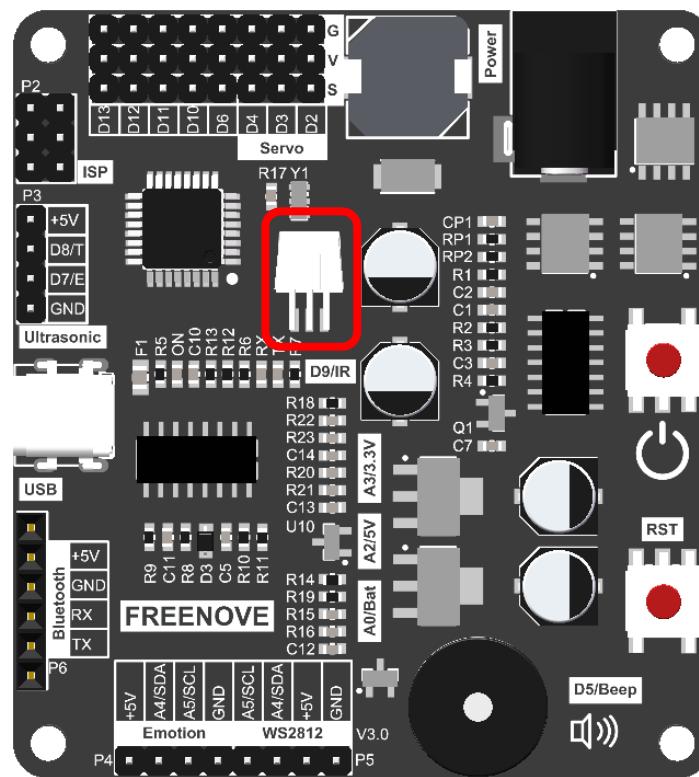
Circuit

Schematic diagram



Hardware connection. If you need any support, please feel free to contact us via: support@freenove.com

The location of the circuit



Sketch

Download the code to control board to control the robot ant to move.

Open 06.IR.ino in **Freenove_Robot_Ant_Kit\Sketches\06.IR**.

You can click .cpp to check the source code of functions for each

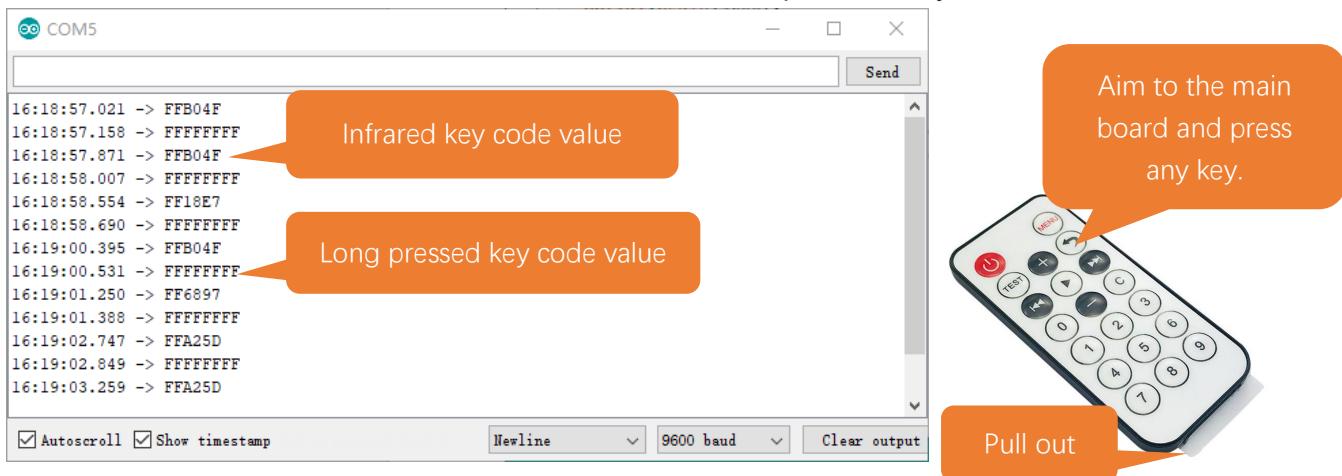
```

06.IR | Arduino 1.8.12
File Edit Sketch Tools Help
06.IR mylRremote.cpp mylRremote.h
1 #include "myIRremote.h"
2
3 void setup() {
4   Serial.begin(9600);
5   IRremote_Setup(); //Initialize the infrared receiving pin
6 }
7
8 void loop() {
9   IRremote_Read(); //Read the infrared key value
10  if(IRremote_flag==1) //If the infrared key value is received, IRremote_flag=1
11  {
12    IRremote_flag = 0;
13    Serial.println(IRremote_value, HEX);
14  }
15  else if (IRremote_flag == 2) //If you press the button long enough, IRremote_flag=2
16  {
17    IRremote_flag = 0;
18    Serial.println(IRremote_value, HEX);
19  }
20 }

```

Any concerns? ✉ support@freenove.com

Compile and upload code. When the code is uploaded successfully, open serial monitor and set baud rate to 9600. Aim the IR remote controller to control board and press the keys.



Code:

```

1 #include "myIRremote.h"
2
3 void setup() {
4     Serial.begin(9600);
5     IRremote_Setup(); //Initialize the infrared receiving pin
6 }
7
8 void loop() {
9     IRremote_Read(); //Read the infrared key value
10    if (IRremote_flag==1) { //If the infrared key value is received, IRremote_flag=1
11        IRremote_flag = 0;
12        Serial.println(IRremote_value, HEX);
13    }
14    else if (IRremote_flag == 2) { //If you press the button long enough, IRremote_flag=2
15        IRremote_flag = 0;
16        Serial.println(IRremote_value, HEX);
17    }
18 }
```

Explanation of Code

Include the header file of library function, which makes it easier to call the program.

```
1 #include "WS2812.h"
```

IRremote_Setup() function is called to initialize infrared receiving pins.

```
5     IRremote_Setup(); //Initialize the infrared receiving pin
```

IRremote_Read() function is called to read the value of a pressed key and store it to variable IRremote_value\

```
9     IRremote_Read();
```

When infrared receiver receives key value from the controller, IRremote_flag will be set to 1. If users long press the key, IRremote_flag will be set to 2. When receiving key value, print it through serial port.

```
10 if (IRremote_flag==1) { //If the infrared key value is received, IRremote_flag=1
11     IRremote_flag = 0;
12     Serial.println(IRremote_value, HEX);
13 }
14 else if (IRremote_flag == 2) { //If you press the button long enough, IRremote_flag=2
15     IRremote_flag = 0;
16     Serial.println(IRremote_value, HEX);
17 }
```

Reference

void IRremote_Setup(void);

IRremote_Setup function is used to initialize infrared receiving pins. When initializing control board, please include it to initialization code.

void IRremote_Read(void);

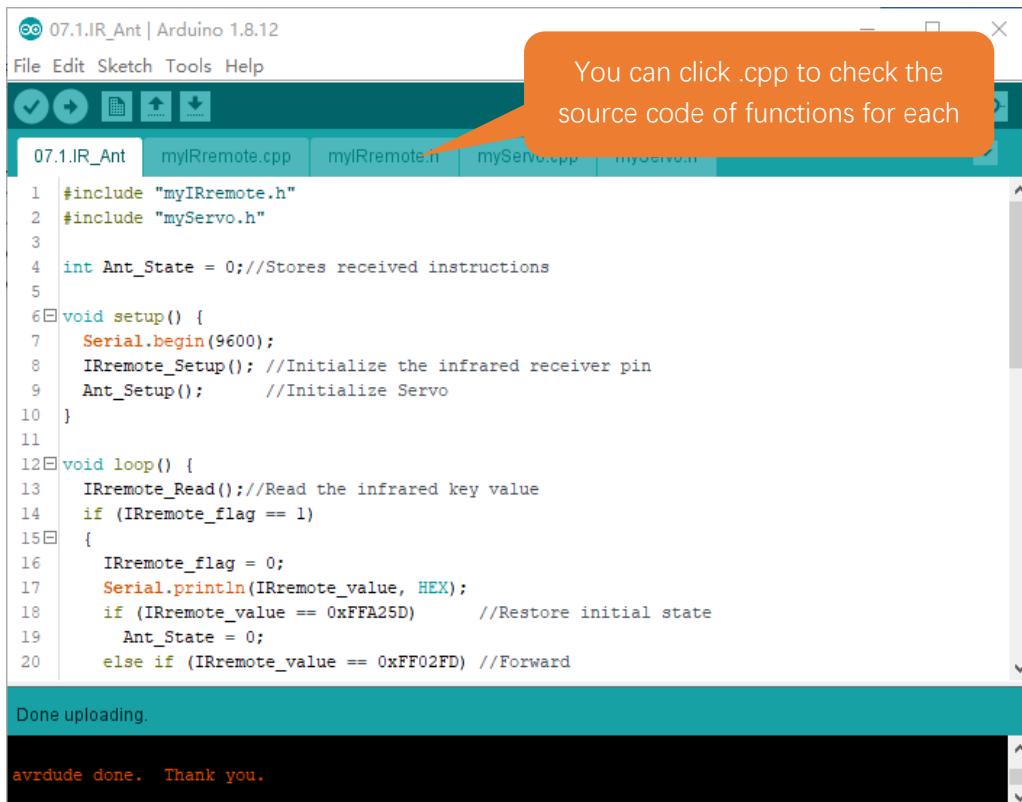
The function to read infrared key value. When reading values, it will store them to IRremote_value and set IRremote_flag to 1. When users long press the infrared remote controller, it will store the receiving value to IRremote_value and set IRremote_flag to 2.

Chapter7 Infrared Controlled Robot

In this chapter, we will learn how to control the robot ant to crawl through the infrared remote controller.

Sketch

Open 07.1.IR_Ant.ino in **Freenove_Robot_Ant_Kit\Sketches\07.1.IR_Ant**.



```

07.1.IR_Ant | Arduino 1.8.12
File Edit Sketch Tools Help
07.1.IR_Ant myIRremote.cpp myIRremote.h myServo.cpp myServo.h
1 #include "myIRremote.h"
2 #include "myServo.h"
3
4 int Ant_State = 0; //Stores received instructions
5
6 void setup() {
7   Serial.begin(9600);
8   IRremote_Setup(); //Initialize the infrared receiver pin
9   Ant_Setup(); //Initialize Servo
10 }
11
12 void loop() {
13   IRremote_Read(); //Read the infrared key value
14   if (IRremote_flag == 1)
15   {
16     IRremote_flag = 0;
17     Serial.println(IRremote_value, HEX);
18     if (IRremote_value == 0xFFA25D) //Restore initial state
19       Ant_State = 0;
20     else if (IRremote_value == 0xFF02FD) //Forward
21     else if (IRremote_value == 0xFF01FD) //Backward
22     else if (IRremote_value == 0xFF00FD) //Left
23     else if (IRremote_value == 0xFF00FF) //Right
24     else if (IRremote_value == 0xFF01FF) //Spots Left
25     else if (IRremote_value == 0xFF02FF) //Spots Right
26   }
}

```

Done uploading.
avrduke done. Thank you.

Compile and upload code. Wait for the code to finish uploading and aim the controller to the robot to control.

	FFA25D	Reset to initial action
	FF02FD	Forward
	FF9867	Backward
	FFE01F	Forward Left
	FF906F	Forward Right
	FF22DD	Spot Turn to Left
	FFC23D	Spot Turn to Right

Code:

```
1 #include "myIRremote.h"
2 #include "myServo.h"
3
4 int Ant_State = 0;//Stores received instructions
5
6 void setup() {
7     Serial.begin(9600);
8     IRremote_Setup(); //Initialize the infrared receiver pin
9     Ant_Setup();      //Initialize Servo
10 }
11
12 void loop() {
13     IRremote_Read(); //Read the infrared key value
14     if (IRremote_flag == 1) {
15         IRremote_flag = 0;
16         Serial.println(IRremote_value, HEX);
17         if (IRremote_value == 0xFFA25D)      //Restore initial state
18             Ant_State = 0;
19         else if (IRremote_value == 0xFF02FD) //Forward
20             Ant_State = 1;
21         else if (IRremote_value == 0xFF9867) //Backwards
22             Ant_State = 2;
23         else if (IRremote_value == 0xFFE01F) //Forward to the left
24             Ant_State = 3;
25         else if (IRremote_value == 0xFF906F) //Forward to the right
26             Ant_State = 4;
27         else if (IRremote_value == 0xFF22DD) //Turn left in situ
28             Ant_State = 5;
29         else if (IRremote_value == 0xFFC23D) //Turn right in situ
30             Ant_State = 6;
31     }
32     else if (IRremote_flag == 2) {
33         IRremote_flag = 0;
34         Serial.println(IRremote_value, HEX);
35     }
36     else{
37         if (Ant_State == 0)//Restore initial state
38             ant_reset_angle();
39         if (Ant_State == 1)//Forward
40             ant_move_forth(2, 5);
41         else if (Ant_State == 2)//Backwards
42             ant_move_back(2, 5);
43         else if (Ant_State == 3)//Forward to the left
```

Any concerns? ✉ support@freenove.com

```

44     ant_move_left(4, 7);
45 else if (Ant_State == 4)//Forward to the right
46     ant_move_right(4, 7);
47 else if (Ant_State == 5)//Turn left in situ
48     ant_situ_left(2, 7);
49 else if (Ant_State == 6)//Turn right in situ
50     ant_situ_right(2, 7);
51 }
52 }
```

Explanation of Code

Include the header file of library function, which makes it easier to call the program.

```

1 #include "myIRremote.h"
2 #include "myServo.h"
```

IRremote_Setup() function to initialize infrared receiving pins. Ant_Setup() function is called to initialize Servo.

```

8 IRremote_Setup(); //Initialize the infrared receiver pin
9 Ant_Setup();      //Initialize Servo
```

When infrared receiver receives key value from the controller, IRremote_flag will be set to 1. If users long press the key, IRremote_flag will be set to 2. When receiving key value, print it through serial port..

```

13 IRremote_Read();//Read the infrared key value
14 //When receiving a specific infrared key code value, it is converted into the corresponding
15 motion instruction
16 if (IRremote_flag == 1) {
17 ...
18 }
19 //When a long press command is received, the information is printed to the serial port.
20 else if (IRremote_flag == 2) {
21 ...
22 }
23 else{//According to the movement command, control the ant robot to crawl
24 ...
25 }
26 }
```

Chapter8 Obstacle Avoiding Robot Ant

In this chapter, we will learn how to make the robot ant avoid obstacles when crawling.

Sketch

Open 08.1.Ultrasonic_Ant.ino in **Freenove_Robot_Ant_Kit\Sketches\08.1.Ultrasonic_Ant**.



```
08.1.Ultrasonic_Ant | Arduino 1.8.12
File Edit Sketch Tools Help
08.1.Ultrasonic_Ant Ultrasonic.cpp Ultrasonic.h myServo.cpp myServo.h
1 #include "Ultrasonic.h"
2 #include "myServo.h"
3
4 void setup() {
5     Ultrasonic_Setup();
6     Ant_Setup();
7     delay(2000);
8 }
9
10 void loop() {
11     Ultrasonic_Get_Data(100);
12     if (Ultrasonic_Value >= 40)
13         ant_move_forth(2, 5); //Forward      //Forward
14     else if (Ultrasonic_Value >= 20 && Ultrasonic_Value < 40)
15         ant_move_left(2, 5);        //Forward to the left
16 //    ant_move_right(2, 5);       //Forward to the right
17     else if (Ultrasonic_Value >= 10 && Ultrasonic_Value < 20)
18         ant_situ_left(2, 5);    //Turn left in situ
19 //    ant_situ_right(2, 5);   //Turn right in situ
20     else
21         ant_move_back(2, 5);      //Backwards
22 }
```

You can click .cpp to check the source code of functions for each

Done compiling.

Sketch uses 6090 bytes (18%) of program storage space. Maximum is 32256 bytes.
Global variables use 265 bytes (12%) of dynamic memory, leaving 1783 bytes for local variables.

1 Arduino Uno on COM5

Compile and upload the code. When the code is uploaded successfully, put the robot on the floor and turn ON the switch. After 2 seconds, the ant will be in obstacle avoidance mode.

Note: Please be careful when taking it back, a moving ant may pinch your hands.

Code:

```

1 #include "Ultrasonic.h"
2 #include "myServo.h"
3 void setup() {
4     Ultrasonic_Setup();
5     Ant_Setup();
6     delay(2000);
7 }
8 void loop() {
9     Ultrasonic_Get_Data(100);
10    if (Ultrasonic_Value >= 40)
11        ant_run_forth(2, 5);      //Forward
12    else if (Ultrasonic_Value >= 20 && Ultrasonic_Value < 40)
13        ant_run_left(2, 5);      //Forward to the left
14    // ant_run_right(2, 5);      //Forward to the right
15    else if (Ultrasonic_Value >= 10 && Ultrasonic_Value < 20)
16        ant_run_situ_left(2, 5); //Turn left in situ
17    // ant_run_situ_right(2, 5); //Turn right in situ
18    else
19        ant_run_back(2, 5);      //Backwards
20 }
```

Explanation of Code

Include the header file of library function, which makes it easier to call the program.

```

1 #include "Ultrasonic.h"
2 #include "myServo.h"
```

Ultrasonic_Setup() function is called to initialize ultrasonic module. Ant_Setup() function is called to initialize servo.

```

4     Ultrasonic_Setup();
5     Ant_Setup();
```

Ultrasonic_Get_Data() is called to obtain distance information every 100 milliseconds and control the robot to make different actions based on the data received. When there is no obstacle 40cm in front of the ant robot, it crawls forward. When the ant robot detects an obstacle 20-40cm ahead, it crawls to the left and ahead to avoid the obstacle in advance. When the ant robot detects an obstacle 10-20cm ahead, which is a relatively close distance, therefore it will make a spot turn to left. When the obstacle is within 10cm, it will move backward to have enough space for the next movement.

```

9     Ultrasonic_Get_Data(100);
10    if (Ultrasonic_Value >= 40)
11        ant_run_forth(2, 5);      //Forward
12    else if (Ultrasonic_Value >= 20 && Ultrasonic_Value < 40)
13        ant_run_left(2, 5);      //Forward to the left
14    else if (Ultrasonic_Value >= 10 && Ultrasonic_Value < 20)
15        ant_run_situ_left(2, 5); //Turn left in situ
17    else
18        ant_run_back(2, 5);      //Backwards
```

Chapter9 Bluetooth Controlled Robot Ant

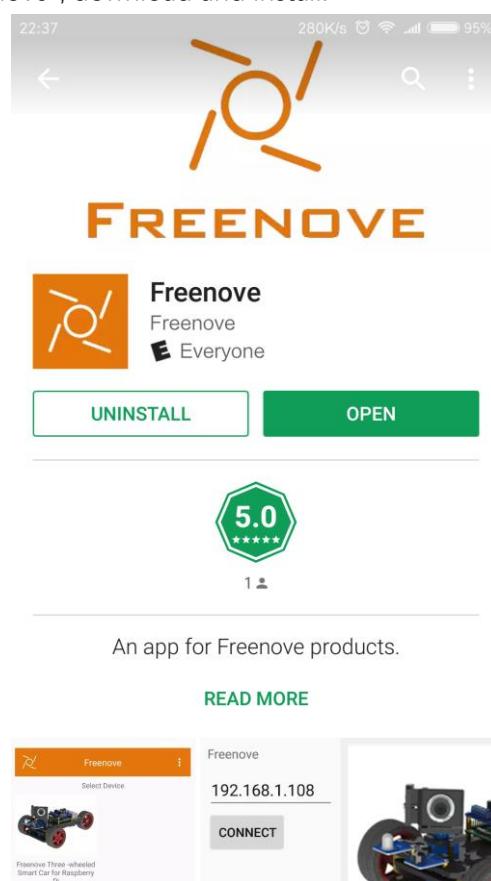
In this chapter, we will learn to use Bluetooth to control the robot.

Install Freenove APP

There are three ways to install our APP, you can choose any one you prefer.

Method 1

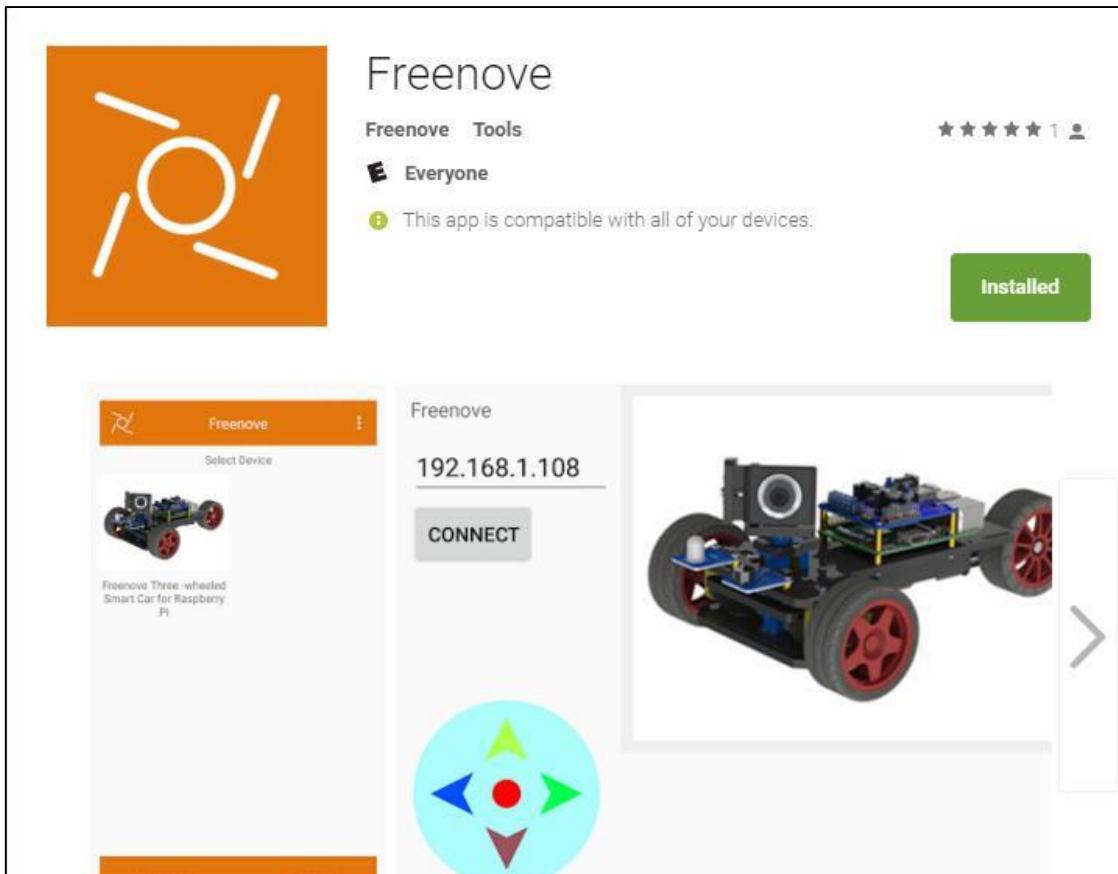
Use Google play to search “Freenove”, download and install.





Method 2

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



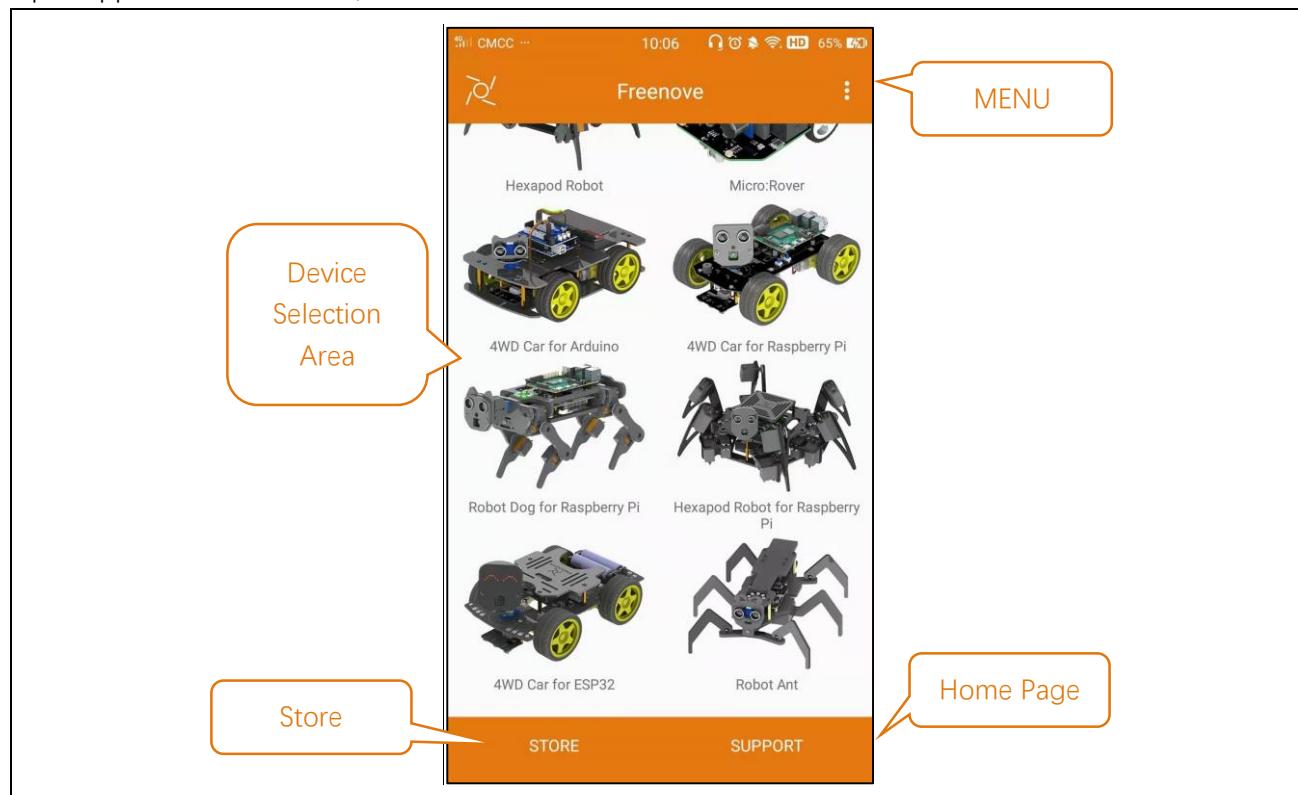
Method 3

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

The screenshot shows a GitHub repository page for 'Freenove / Freenove_app_for_Android'. At the top, there is a navigation bar with links for 'Code', 'Issues 0', 'Pull requests 0', 'Projects 0', 'Wiki', 'Pulse', 'Graphs', and 'Settings'. Below the navigation bar, there is a message 'Apply to Freenove products.' and a large orange callout bubble with the text 'Click here to download.' pointing to the 'Clone or download' button. The repository statistics show '1 commit', '1 branch', and '0 releases'. The 'Branch: master' dropdown and 'New pull request' button are also visible. The repository history shows a single commit by 'SuhaylZhao' from 'First Publish.' 3 minutes ago. The commit hash is '0323fc5'. The file list includes 'Readme.txt' and 'freenove.apk', both of which were 'First Publish.' 3 minutes ago.

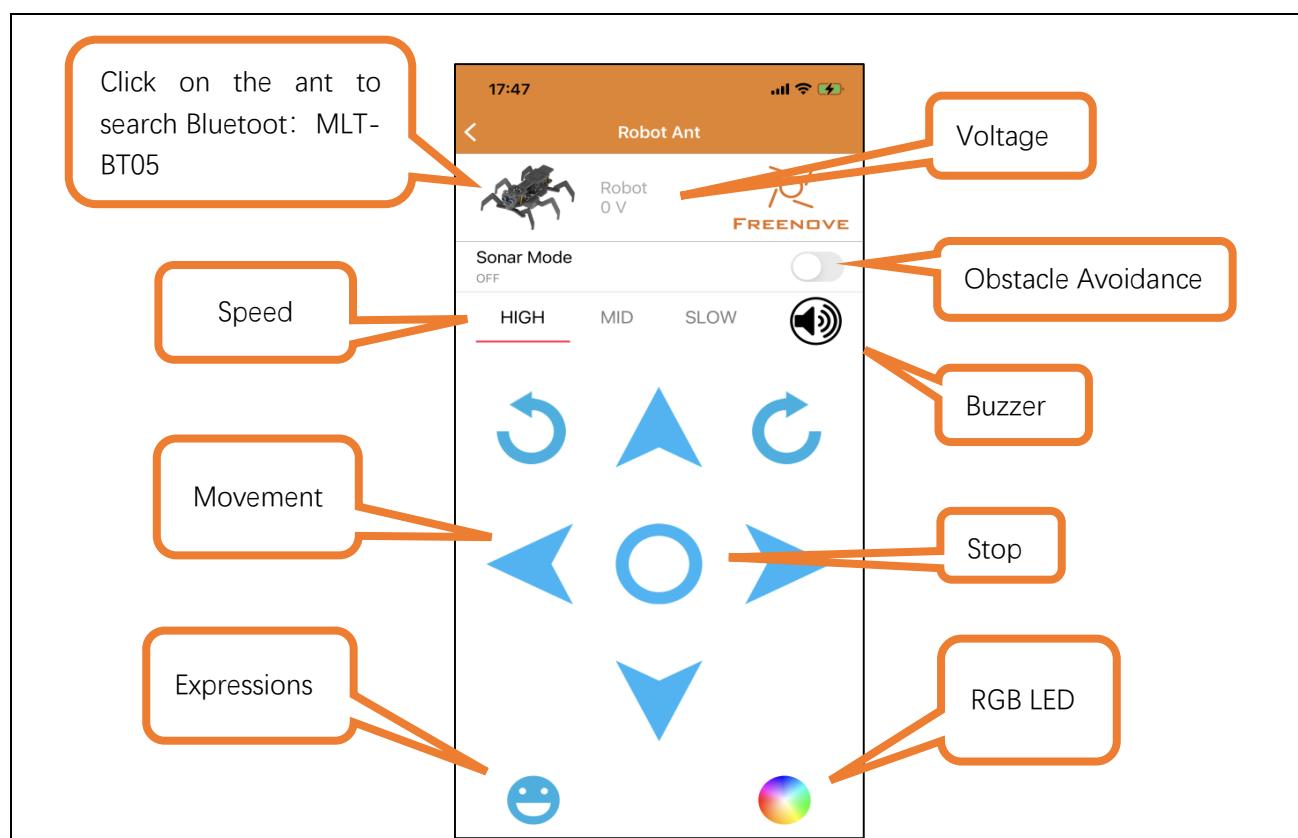
Menu

Open application “Freenove”, as shown below:



Introduction of the APP

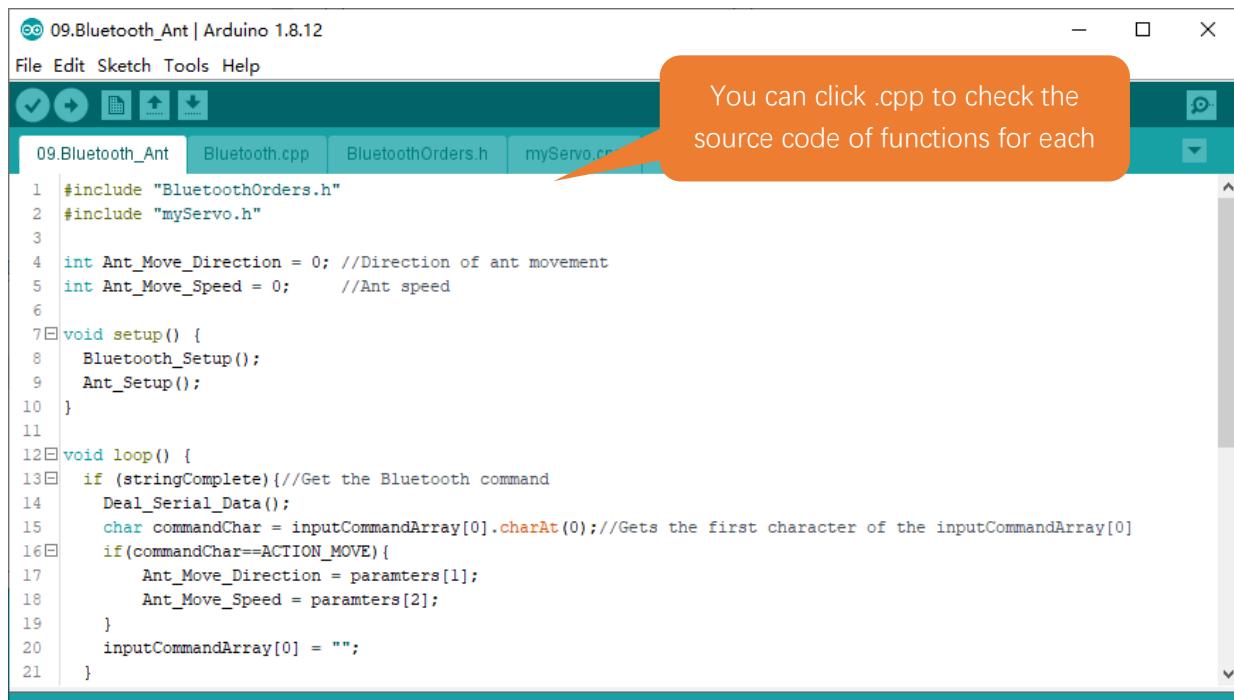
In this chapter, we use Freenove 4WD Car for ESP32, so it is necessary to understand the interface of this mode.



Any concerns? [✉ support@freenove.com](mailto:support@freenove.com)

Sketch

Open 09.Bluetooth_Ant.ino in **Freenove_Robot_Ant_Kit\Sketches\09.Bluetooth_Ant**.



```

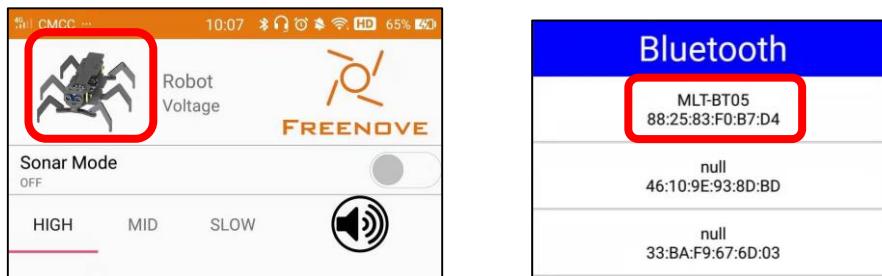
09.Bluetooth_Ant | Arduino 1.8.12
File Edit Sketch Tools Help
09.Bluetooth_Ant Bluetooth.cpp BluetoothOrders.h myServo.cpp
1 #include "BluetoothOrders.h"
2 #include "myServo.h"
3
4 int Ant_Move_Direction = 0; //Direction of ant movement
5 int Ant_Move_Speed = 0; //Ant speed
6
7 void setup() {
8   Bluetooth_Setup();
9   Ant_Setup();
10 }
11
12 void loop() {
13   if (stringComplete){//Get the Bluetooth command
14     Deal_Serial_Data();
15     char commandChar = inputCommandArray[0];//Gets the first character of the inputCommandArray[0]
16     if(commandChar==ACTION_MOVE){
17       Ant_Move_Direction = paramters[1];
18       Ant_Move_Speed = paramters[2];
19     }
20     inputCommandArray[0] = "";
21   }
}

```

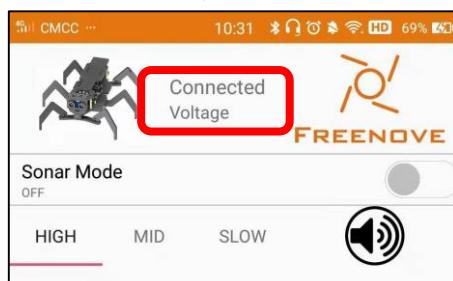
Before uploading code, please remove the Bluetooth module. Because the Bluetooth module uses the same interface as the program download function, they cannot be used simultaneously.

Compile and upload code to control board. When the code is uploaded successfully, replug in the Bluetooth module and turn ON the power. Open the mobile APP and select the robot ant.

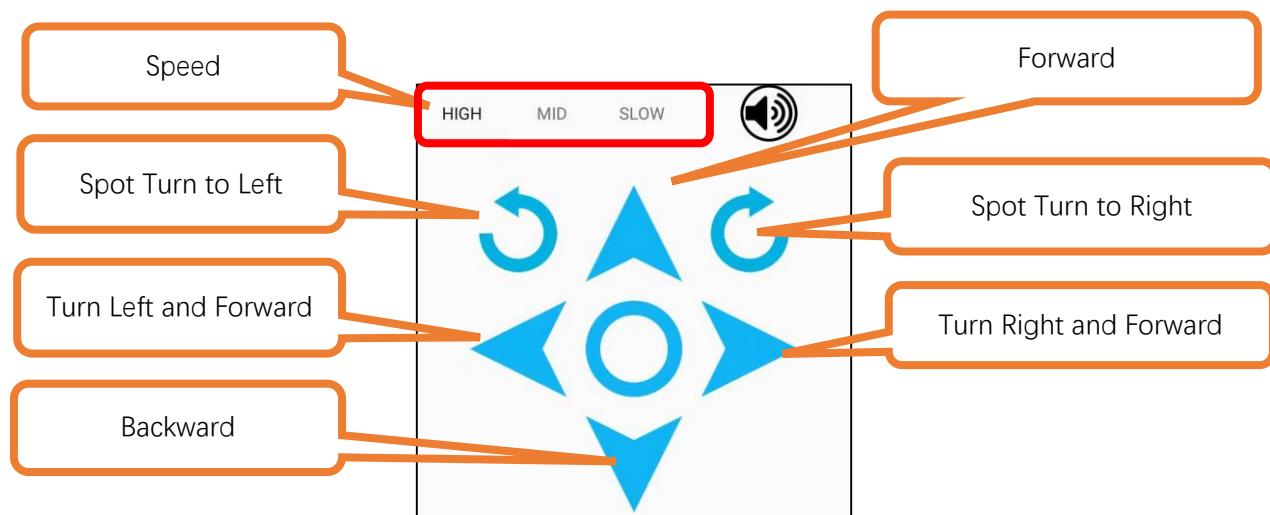
Click on the picture of the robot ant and select MLT-BT05 to connect to Bluetooth module in the pop-up window.



When it connects to Bluetooth module successfully, it will show as below:



At this point, you can use your phone to control the robot ant to move.



Code:

```

1 #include "BluetoothOrders.h"
2 #include "myServo.h"
3
4 int Ant_Move_Direction = 0; //Direction of ant movement
5 int Ant_Move_Speed = 0; //Ant speed
6
7 void setup() {
8     Bluetooth_Setup();
9     Ant_Setup();
10 }
11
12 void loop() {
13     if (stringComplete){//Get the Bluetooth command
14         Deal_Serial_Data();
15         char commandChar = inputCommandArray[0].charAt(0); //Gets the first character of the
16         inputCommandArray[0]
17         if(commandChar==ACTION_MOVE){
18             Ant_Move_Direction = paramters[1];
19             Ant_Move_Speed = paramters[2];
20         }
21         inputCommandArray[0] = "";
22     } else{
23         if (Ant_Move_Direction == 0)//Restore initial state
24             ant_reset_angle();
25         else if (Ant_Move_Direction == 1)//Forward
26             ant_move_forth(2, 5+2*Ant_Move_Speed);
27         else if (Ant_Move_Direction == 2)//Backwards
28             ant_move_back(2, 5+2*Ant_Move_Speed);
29         else if (Ant_Move_Direction == 3)//Forward to the left
30             ant_move_left(4, 5+2*Ant_Move_Speed);
31     }
32 }
```

Any concerns? ✉ support@freenove.com

```

31     else if (Ant_Move_Direction == 4)//Forward to the right
32         ant_move_right(4, 5+2*Ant_Move_Speed);
33     else if (Ant_Move_Direction == 5)//Turn left in situ
34         ant_situ_left(2, 5+2*Ant_Move_Speed);
35     else if (Ant_Move_Direction == 6)//Turn right in situ
36         ant_situ_right(2, 5+2*Ant_Move_Speed);
37     }
38 }
```

Explanation of Code

Include the header file of library function, which makes it easier to call the program.

```

1 #include "BluetoothOrders.h"
2 #include "myServo.h"
```

Bluetooth_Setup() is called to initialize serial port and set baud rate to 9600. Ant_Setup() is called to initialize Servo.

```

8 Bluetooth_Setup();
9 Ant_Setup();
```

Define two variables: one is to store the moving direction of the robot ant and the other to store the speed level.

```

4 int Ant_Move_Direction = 0; //Direction of ant movement
5 int Ant_Move_Speed = 0; //Ant speed
```

When Bluetooth module receives data from phone, the serial port will interrupt the function receiving data and store the variable to inputStringBLE, and set variable stingComplete to true at the same time.

When the main program detects that stringComplete is true, it will call Deal_Serial_Data() to analyze the receiving data and save the commands to array inputCommandArray and store the parameter to array paramters.

When the control board receives commands to make the ant move, it will store the data of the ant's moving direction and speed to Ant_Move_Direction and Ant_Move_Speed.

```

13 if (stringComplete){//Get the Bluetooth command
14     Deal_Serial_Data();
15     char commandChar = inputCommandArray[0].charAt(0); //Gets the first character of the
16     inputCommandArray[0]
17     if(commandChar==ACTION_MOVE) {
18         Ant_Move_Direction = paramters[1];
19         Ant_Move_Speed = paramters[2];
20     }
21     inputCommandArray[0] = "";
```

If the Bluetooth module doesn't receive any data, the robot ant will perform the instructions that has been received before new instructions arrive .

```

22 else{
23     if (Ant_Move_Direction == 0)//Restore initial state
24         ant_reset_angle();
25     else if (Ant_Move_Direction == 1)//Forward
26         ant_move_forth(2, 5+2*Ant_Move_Speed);
```

```

27     else if (Ant_Move_Direction == 2)//Backwards
28         ant_move_back(2, 5+2*Ant_Move_Speed);
29     else if (Ant_Move_Direction == 3)//Forward to the left
30         ant_move_left(4, 5+2*Ant_Move_Speed);
31     else if (Ant_Move_Direction == 4)//Forward to the right
32         ant_move_right(4, 5+2*Ant_Move_Speed);
33     else if (Ant_Move_Direction == 5)//Turn left in situ
34         ant_situ_left(2, 5+2*Ant_Move_Speed);
35     else if (Ant_Move_Direction == 6)//Turn right in situ
36         ant_situ_right(2, 5+2*Ant_Move_Speed);
37 }
```

Commands of Bluetooth Module

#define ACTION_MOVE	'A'	//Ant movement commands
#define ACTION_EMOTION	'B'	//Expression control commands
#define ACTION_RGB	'C'	//WS2812 control commands
#define ACTION_BUZZER	'D'	//Buzzer control commands
#define ACTION_ULTRASONIC	'E'	//Ultrasonic control commands
#define ACTION_CAR_MODE	'H'	//Ultrasonic Obstacle Avoidance Command
#define ACTION_GET_VOLTAGE	'I'	//Battery Power Query
#define INTERVAL_CHAR	'#'	//The directive resolves the separator character

Reference

`void Bluetooth_Setup(void);`

The Bluetooth module applies serial communication with a default baud rate of 9600. Every time this function is called to initialize serial port so that it can communicate with Bluetooth module.

`void Deal_Serial_Data(void);`

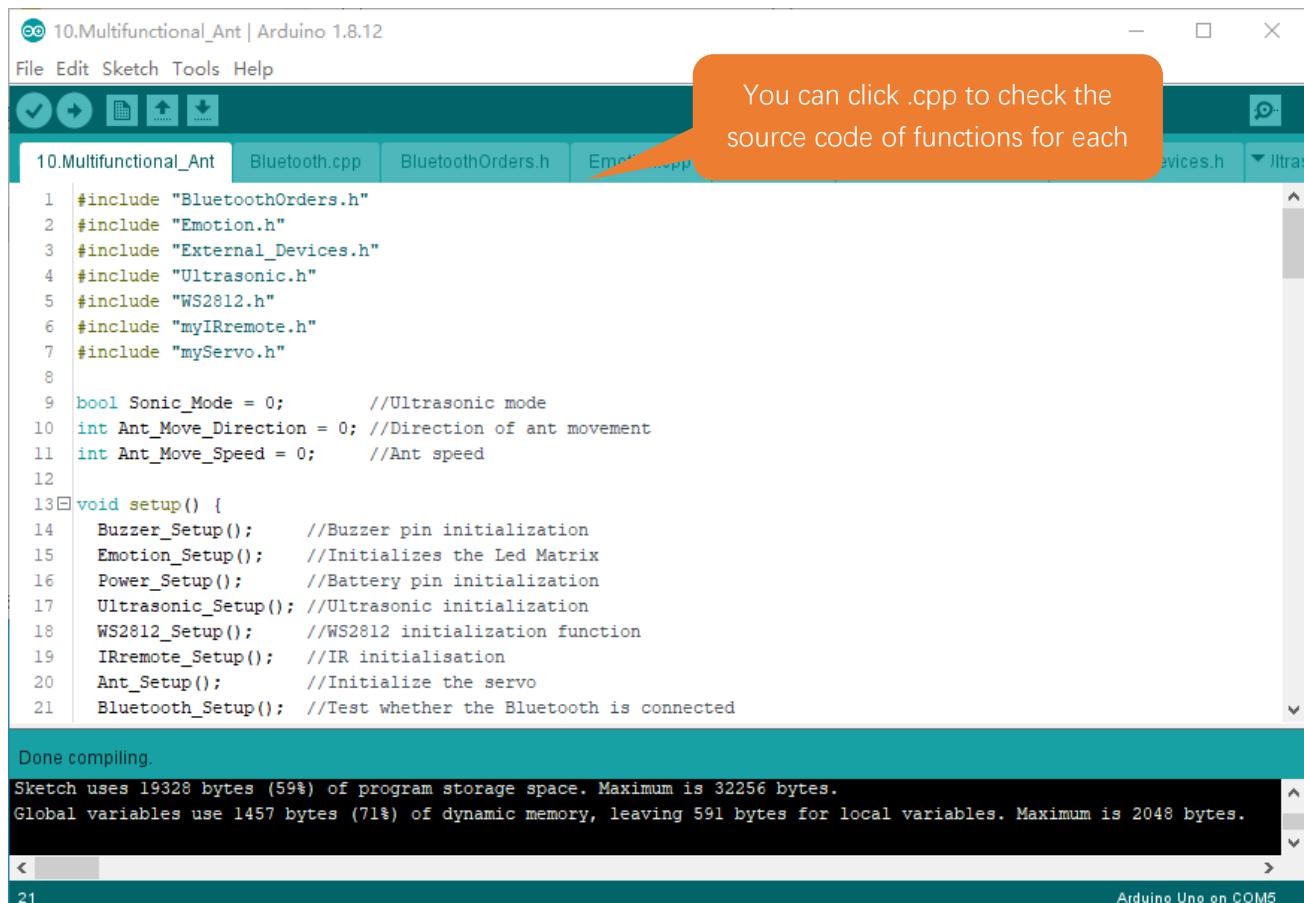
The parse function of serial port data. Every time when this function is called, the program will stop receiving data and analyze the content received, and then store the parsed instruction in the array InputCommandArray and the parsed instruction parameters in the array paramters.。

Chapter10 Integrated Routine

In this chapter, we will integrate all the previous features to make a robot feature with RGB LED, expressions, buzzer, battery level query and modes of infrared remote control, Bluetooth remote control, and obstacle avoidance.

Sketch

Open 10.Multifunctional_Ant.ino in **Freenove_Robot_Ant_Kit\Sketches\10.Multifunctional_Ant**.



You can click .cpp to check the source code of functions for each

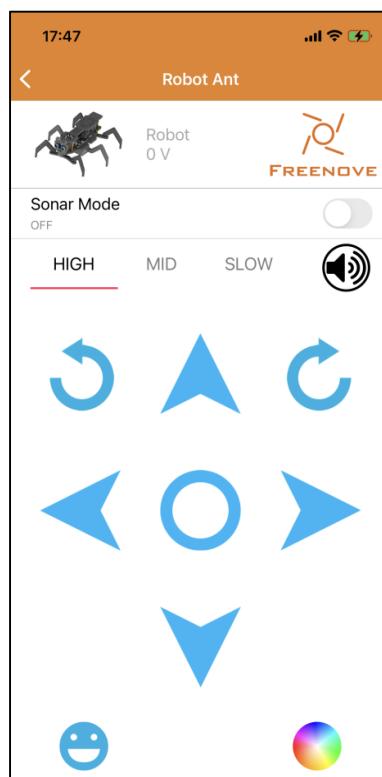
```
1 #include "BluetoothOrders.h"
2 #include "Emotion.h"
3 #include "External_Devices.h"
4 #include "Ultrasonic.h"
5 #include "WS2812.h"
6 #include "myIRremote.h"
7 #include "myServo.h"
8
9 bool Sonic_Mode = 0;           //Ultrasonic mode
10 int Ant_Move_Direction = 0;   //Direction of ant movement
11 int Ant_Move_Speed = 0;       //Ant speed
12
13 void setup() {
14     Buzzer_Setup();           //Buzzer pin initialization
15     Emotion_Setup();          //Initializes the Led Matrix
16     Power_Setup();            //Battery pin initialization
17     Ultrasonic_Setup();       //Ultrasonic initialization
18     WS2812_Setup();          //WS2812 initialization function
19     IRremote_Setup();         //IR initialisation
20     Ant_Setup();              //Initialize the servo
21     Bluetooth_Setup();        //Test whether the Bluetooth is connected
}
Done compiling.
Sketch uses 19328 bytes (59%) of program storage space. Maximum is 32256 bytes.
Global variables use 1457 bytes (71%) of dynamic memory, leaving 591 bytes for local variables. Maximum is 2048 bytes.
21
```

Before uploading code, please remove the Bluetooth module. After compile and upload code to control board, replug in the Bluetooth module and then turn ON power.



	Reset				Buzzer ON/OFF
	Turn Left		Forward		Spot Turn to Right
	Forward Left		Speed Adjustment		Forward Right
	Obstacle Avoidance		Backward		Obstacle Avoidance Mode OFF
	Red of RGB LED ON/OFF		Green of RGB LED ON/OFF		Blue of RGB LED ON/OFF
	Modes of WS2812		Modes of Dynamic Expressions		Random Display of Static Expressions
	Custom		Custom		Custom

You can also use phone APP to control the robot ant.



Code:

```

1 #include "BluetoothOrders.h"
2 #include "Emotion.h"
3 #include "External_Devices.h"
4 #include "Ultrasonic.h"
5 #include "WS2812.h"

```

Any concerns? ✉ support@freenove.com

```
6 #include "myIRremote.h"
7 #include "myServo.h"
8
9 bool Sonic_Mode = 0;           //Ultrasonic mode
10 int Ant_Move_Direction = 0; //Direction of ant movement
11 int Ant_Move_Speed = 0;      //Ant speed
12
13 void setup() {
14     Buzzer_Setup();          //Buzzer pin initialization
15     Emotion_Setup();         //Initializes the Led Matrix
16     Power_Setup();           //Battery pin initialization
17     Ultrasonic_Setup();     //Ultrasonic initialization
18     WS2812_Setup();         //WS2812 initialization function
19     IRremote_Setup();        //IR initialisation
20     Ant_Setup();             //Initialize the servo
21     Bluetooth_Setup();       //Test whether the Bluetooth is connected
22 }
23
24 void loop() {
25     if (Sonic_Mode == 1){//Ultrasonic obstacle avoidance mode
26         //Get the distance value every 1s and store it in the Ultrasonic_Value
27         Ultrasonic_Get_Data();
28         if (Ultrasonic_Value >= 40)
29             Ant_Move_Direction = 1;//Forward
30         else if (Ultrasonic_Value >= 20 && Ultrasonic_Value < 40)
31             Ant_Move_Direction = 3;//Forward to the left
32         else if (Ultrasonic_Value >= 10 && Ultrasonic_Value < 20)
33             Ant_Move_Direction = 5;//Turn right in situ
34         else
35             Ant_Move_Direction = 2;//Backwards
36     }
37     if (IRremote_flag != 0){//Get the infrared command
38         IRremote_flag = 0;
39         switch (IRremote_value){
40             case 0xFFA25D://Restore initial state
41                 Ant_Move_Direction = 0;
42                 break;
43             case 0xFF02FD://Forward
44                 Ant_Move_Direction = 1;
45                 break;
46             case 0xFF9867://Backwards
47                 Ant_Move_Direction = 2;
48                 break;
49             case 0xFFE01F://Forward to the left
```

```
50     Ant_Move_Direction = 3;
51     break;
52 case 0xFF906F://Forward to the right
53     Ant_Move_Direction = 4;
54     break;
55 case 0xFF22DD://Turn left in situ
56     Ant_Move_Direction = 5;
57     break;
58 case 0xFFC23D://Turn right in situ
59     Ant_Move_Direction = 6;
60     break;
61 case 0xFFE21D://Buzzer 100ms
62     Buzzer_Alarm();
63     break;
64 case 0xFFA857://Move speed control
65     Ant_Move_Speed--;
66     if (Ant_Move_Speed < 0)
67         Ant_Move_Speed = 2;
68     break;
69 case 0xFF6897://Ultrasonic mode on
70     Sonic_Mode = 1;
71     Buzzer_Alarm();
72     break;
73 case 0xFFB04F://Ultrasonic mode off
74     Sonic_Mode = 0;
75     Ant_Move_Direction = 0;
76     Buzzer_Alarm();
77     break;
78 case 0xFF30CF://WS2812 RED On/Off
79     WS2812_Set_LED(1, 0, 0);
80     break;
81 case 0xFF18E7://WS2812 GREEN On/Off
82     WS2812_Set_LED(0, 1, 0);
83     break;
84 case 0xFF7A85://WS2812 BLUE On/Off
85     WS2812_Set_LED(0, 0, 1);
86     break;
87 case 0xFF10EF://WS2812 set mode
88     ws2812_task_mode++;
89     if (ws2812_task_mode > 5)
90         ws2812_task_mode = 0;
91     break;
92 case 0xFF38C7://Display animated emoticons
93     emotion_count = 0;
```

```
94         emotion_task_mode++;
95         if (emotion_task_mode > 6)
96             emotion_task_mode = 0;
97         break;
98     case 0xFF5AA5://Display static emotions
99         Emotion_SetMode(7);
100        break;
101    default:
102        break;
103    }
104 }
105 if (stringComplete) {//Get the Bluetooth command
106     Deal_Serial_Data();
107     char commandChar = inputCommandArray[0].charAt(0);
108     if (commandChar == ACTION_MOVE) {//Control ant movement
109         if (Sonic_Mode != 0) {
110             Sonic_Mode = 0;
111             Buzzer_Alarm();
112         }
113         Ant_Move_Direction = paramters[1];
114         Ant_Move_Speed = paramters[2];
115     }
116     if (commandChar == ACTION_EMOTION) {//Control expression module display
117         if (paramters[1] <= 7)
118             Emotion_SetMode(paramters[1]);
119         else if (paramters[1] > 7)
120             Emotion_SetMode(paramters[1], paramters[2]);
121     }
122     if (commandChar == ACTION_RGB) {//Control WS2812 display
123         WS2812_SetMode(paramters[1]);
124         WS2812_Set_Color(paramters[2], paramters[3], paramters[4]);
125     }
126     if (commandChar == ACTION_BUZZER) //Control the buzzer sound
127         Buzzer_Alert(paramters[1]);
128     if (commandChar == ACTION_ULTRASONIC) //Get ultrasonic distance value
129         Ultrasonic_Bluetooth_Data();
130     if (commandChar == ACTION_CAR_MODE) {//Ultrasonic mode
131         int cmd_mode = constrain(paramters[1], 0, 1);
132         Sonic_Mode = cmd_mode;
133         if (Sonic_Mode == 0)
134             Ant_Move_Direction = 0;
135         Buzzer_Alarm();
136     }
137     if (commandChar == ACTION_GET_VOLTAGE) //Get battery voltage
```

```

138     Power_Bluetooth_Data();
139     inputCommandArray[0] = "";
140 }
141 else{//Execute code without blocking
142     IRremote_Read();//Non - blocking access to infrared commands
143     Ant_Servo_Mode(Ant_Move_Direction, Ant_Move_Speed);//Non-blocking ant motion function
144     Emotion_Show();//Non-blocking expression module display function
145     WS2812_Show();//Non-blocking color light display function
146 }
147 }
```

Explanation of Code

Include the header file of library function, which makes it easier to call the program.

```

1 #include "BluetoothOrders.h"
2 #include "Emotion.h"
3 #include "External_Devices.h"
4 #include "Ultrasonic.h"
5 #include "WS2812.h"
6 #include "myIRremote.h"
7 #include "myServo.h"
```

The initialization function of each module is called to initialize every feature of the robot ant.

```

14 Buzzer_Setup(); //Buzzer pin initialization
15 Emotion_Setup(); //Initializes the Led Matrix
16 Power_Setup(); //Battery pin initialization
17 Ultrasonic_Setup(); //Ultrasonic initialization
18 WS2812_Setup(); //WS2812 initialization function
19 IRremote_Setup(); //IR initialisation
20 Ant_Setup(); //Initialize the servo
21 Bluetooth_Setup(); //Test whether the Bluetooth is connected
```

Variable Sonic_Mode is used to record whether the obstacle avoidance mode is ON, while the other two variables are used to record the robot's moving direction and speed.

```

9 bool Sonic_Mode = 0; //Ultrasonic mode
10 int Ant_Move_Direction = 0; //Direction of ant movement
11 int Ant_Move_Speed = 0; //Ant speed
```

When the obstacle avoidance mode is ON, Ultrasonic_Get_Data() is called to obtain ultrasonic data every 1s and determine the robot's moving direction based on the data.

```

25 if (Sonic_Mode == 1){//Ultrasonic obstacle avoidance mode
26     //Get the distance value every 1s and store it in the Ultrasonic_Value
27     Ultrasonic_Get_Data();
28     if (Ultrasonic_Value >= 40)
29         Ant_Move_Direction = 1;//Forward
30     else if (Ultrasonic_Value >= 20 && Ultrasonic_Value < 40)
31         Ant_Move_Direction = 3;//Forward to the left
32     else if (Ultrasonic_Value >= 10 && Ultrasonic_Value < 20)
33         Ant_Move_Direction = 5;//Turn right in situ
```

```

34     else
35         Ant_Move_Direction = 2;//Backwards
36     }

```

When the robot ant receives instructions from infrared remote controller, it will make corresponding reactions.

```

37     if (IRremote_flag != 0) {//Get the infrared command
38         IRremote_flag = 0;
39         switch (IRremote_value) {
40             case 0xFFA25D://Restore initial state
41                 Ant_Move_Direction = 0;
42                 break;
43             case 0xFF02FD://Forward
44                 Ant_Move_Direction = 1;
45                 break;
46             case 0xFF9867://Backwards
47                 Ant_Move_Direction = 2;
48                 break;
49             case 0xFFE01F://Forward to the left
50                 Ant_Move_Direction = 3;
51                 break;
52             case 0xFF906F://Forward to the right
53                 Ant_Move_Direction = 4;
54                 break;
55             case 0xFF22DD://Turn left in situ
56                 Ant_Move_Direction = 5;
57                 break;
58             case 0xFFC23D://Turn right in situ
59                 Ant_Move_Direction = 6;
60                 break;
61             case 0xFFE21D://Buzzer 100ms
62                 Buzzer_Alarm();
63                 break;
64             case 0xFFA857://Move speed control
65                 Ant_Move_Speed--;
66                 if (Ant_Move_Speed < 0)
67                     Ant_Move_Speed = 2;
68                 break;
69             case 0xFF6897://Ultrasonic mode on
70                 Sonic_Mode = 1;
71                 Buzzer_Alarm();
72                 break;
73             case 0xFFB04F://Ultrasonic mode off
74                 Sonic_Mode = 0;
75                 Ant_Move_Direction = 0;
76                 Buzzer_Alarm();

```

```

77     break;
78     case 0xFF30CF://WS2812 RED On/Off
79         WS2812_Set_LED(1, 0, 0);
80         break;
81     case 0xFF18E7://WS2812 GREEN On/Off
82         WS2812_Set_LED(0, 1, 0);
83         break;
84     case 0xFF7A85://WS2812 BLUE On/Off
85         WS2812_Set_LED(0, 0, 1);
86         break;
87     case 0xFF10EF://WS2812 set mode
88         ws2812_task_mode++;
89         if (ws2812_task_mode > 5)
90             ws2812_task_mode = 0;
91         break;
92     case 0xFF38C7://Display animated emoticons
93         emotion_count = 0;
94         emotion_task_mode++;
95         if (emotion_task_mode > 6)
96             emotion_task_mode = 0;
97         break;
98     case 0xFF5AA5://Display static emotions
99         Emotion_SetMode(7);
100        break;
101    default:
102        break;
103    }
104}

```

When the robot ant receives instructions from Bluetooth module, it will make corresponding reactions.

```

105 if (stringComplete) {//Get the Bluetooth command
106     Deal_Serial_Data();
107     char commandChar = inputCommandArray[0].charAt(0);
108     if (commandChar == ACTION_MOVE){//Control ant movement
109         if (Sonic_Mode != 0){
110             Sonic_Mode = 0;
111             Buzzer_Alarm();
112         }
113         Ant_Move_Direction = paramters[1];
114         Ant_Move_Speed = paramters[2];
115     }
116     if (commandChar == ACTION_EMOTION){//Control expression module display
117         if (paramters[1] <= 7)
118             Emotion_SetMode(paramters[1]);

```

```

119     else if (paramters[1] > 7)
120         Emotion_SetMode(paramters[1], paramters[2]);
121     }
122     if (commandChar == ACTION_RGB) { //Control WS2812 display
123         WS2812_SetMode(paramters[1]);
124         WS2812_Set_Color(paramters[2], paramters[3], paramters[4]);
125     }
126     if (commandChar == ACTION_BUZZER) //Control the buzzer sound
127         Buzzer_Alert(paramters[1]);
128     if (commandChar == ACTION_ULTRASONIC) //Get ultrasonic distance value
129         Ultrasonic_Bluetooth_Data();
130     if (commandChar == ACTION_CAR_MODE) { //Ultrasonic mode
131         int cmd_mode = constrain(paramters[1], 0, 1);
132         Sonic_Mode = cmd_mode;
133         if (Sonic_Mode == 0)
134             Ant_Move_Direction = 0;
135         Buzzer_Alarm();
136     }
137     if (commandChar == ACTION_GET_VOLTAGE) //Get battery voltage
138         Power_Bluetooth_Data();
139     inputCommandArray[0] = "";
140 }
```

The program adapts non-blocking functions to constantly detect whether there is any command from infrared controller or Bluetooth module, and then control the robot dog to crawl, emit lights and display expressions accordingly.

```

141 else { //Execute code without blocking
142     IRremote_Read(); //Non-blocking access to infrared commands
143     Ant_Servo_Mode(Ant_Move_Direction, Ant_Move_Speed); //Non-blocking ant motion function
144     Emotion_Show(); //Non-blocking expression module display function
145     WS2812_Show(); //Non-blocking color light display function
146 }
```

What's next?

Thanks for your reading.

This tutorial is all over here. If you find any mistakes, omissions or you have other ideas and questions about contents of this tutorial or the kit and etc, please feel free to contact us: support@freenove.com
We will check and correct it as soon as possible.

If you want to learn more about Arduino, Raspberry Pi, smart cars, robots and other interesting products in science and technology, please continue to focus on our website. We will continue to launch cost-effective, innovative and exciting products.

<http://www.freenove.com/>

Thank you again for choosing Freenove products.

Any concerns? [✉ support@freenove.com](mailto:support@freenove.com)