

# Preface

## About SunFounder

SunFounder is a technology company focused on Raspberry Pi and Arduino open source community development. Committed to the promotion of open source culture, we strive to bring the fun of electronics making to people all around the world and enable everyone to be a maker. Our products include learning kits, development boards, robots, sensor modules and development tools. In addition to high quality products, SunFounder also offers video tutorials to help your own project. If you have interest in open source or making something cool, welcome to join us! Visit [www.sunfounder.com](http://www.sunfounder.com) for more!

## About the Rollbot Kit

Designed based on Arduino, Rollbot is a simple and funny kit. With a cute appearance, you can dress it up with pretty "clothes" and colorful OLED stickers and there's an elaborately-designed map for it to look for treasures in a magic forest. The Rollbot will sing and dance for you. It can follow lines on a maze map by programming and you can also control it through an Android APP **Rollman**. In addition, you can play games similar to the Chinese aeroplane chess, Ludo and Monopoly with your friends!

Rollbot kit is a very interesting and suitable learning tool for Arduino and robot hobbyists. It has a simple structure and adopts PCB as the main structure. Based on the Arduino platform, we can learn programming from easy to difficult so as to control the robot. With the help of the mobile APP, we can play different games! The kit includes a Bluetooth module, an OLED module, a line-following module, and an insect-shaped PCB with beautiful LED lights, etc. With the help of this manual, you will learn to assemble the Rollbot and download the program to control it. Hope you will love this adorable bot!

You will enjoy learning how all this work. Visit our website [www.sunfounder.com](http://www.sunfounder.com) to download the related code, view the user manual on [LEARN -> Get Tutorials](#), and watch related videos under [VIDEO](#), or clone the code on our page of github.com at

[https://github.com/sunfounder/SunFounder\\_Rollbot](https://github.com/sunfounder/SunFounder_Rollbot)

You are welcome to pull requests and issue posts on our page on Github.

## Free Support



If you have any **TECHNICAL questions**, add a topic under **FORUM** section on our website and we'll reply as soon as possible.



For **NON-TECH questions** like order and shipment issues, please **send an email to [service@sunfounder.com](mailto:service@sunfounder.com)**. You're also welcomed to share your projects on FORUM.

# Contents

Introduction.....	2
1.1 Overview.....	2
Components List.....	3
2.1. Mechanical Fasteners.....	3
2.2. Electronic Components.....	4
2.3. Tools.....	7
Software Operation.....	8
3.1 Get the Code.....	8
3.1.1 Install Arduino IDE.....	8
3.1.2 Connecting Rollbot to Your Computer.....	8
3.2 Add Libraries.....	9
Assembly.....	11
4.1 Main Board + N20 Gear Motors.....	12
4.2 N20 Gear Motors + Wheels.....	13
4.3 Main Board + Universal Wheel.....	14
4.4 Main Board + Battery.....	15
4.5 Main Board + Infrared Sensor.....	17
4.6 Main Board + OLED Screen.....	17
4.7 Rollbot + Covers.....	17
4.8 DIY Covers.....	18
Gameplay.....	20
5.1 Singing.....	21
5.2 Display the Sensor Signal Intensity on OLED.....	23
5.3 Motor Testing.....	24
5.4 Line Following.....	25
5.4.1 Magic Forest.....	25
5.4.2 DIY Map.....	27
5.5 Bluetooth App Controls the Robot.....	28
5.6 Dice Game.....	31
Hardware Introduction.....	35
6.1 Overview.....	35
6.2 Main Control Module.....	35

6.3	USB Interface Module.....	37
6.4	LED Indicator Module .....	37
6.5	OLED Display Module .....	38
6.6	Bluetooth Module .....	38
	Program Explanation .....	39
7.1	Singing with Buzzer.....	39
7.2	Smiling Face with LEDs .....	42
7.3	Sensor Signal Intensity on OLED Screen .....	43
7.4	The Bluetooth Control Program .....	43
7.5	Line Following .....	46
	Appendix: FAQ .....	48
	Afterword.....	49

SunFounder

## Kind Reminder



Kids can seek help from their parents or teacher if there're difficulties.

# Introduction

## 1.1 Overview

Rollbot kit is a very interesting and suitable learning tool for Arduino and robot hobbyist. It has simple structure and runs PCB as the main structure of the robot. As it is based on the Arduino platform, we can learn programming from easy to difficult so as to achieve the control of the robot. With the help of the mobile APP, we can play different games similar to the Chinese aeroplane chess, Ludo and Monopoly with friends!

This funny mini robot is composed of an integrated PCB, two driving wheels and a universal wheel. It is charged by a rechargeable Li-ion battery and compatible with the Arduino board. Accessories include an OLED screen and an infrared sensor module. In addition, the cute insect-shaped integrated board contains the control of the N20 gear motor, the OLED screen and buzzer, as well as a Bluetooth module. This kit is also equipped with a battery charger and a simple but delicate map for you to play the game via the line following function. By downloading and installing the mobile **app Rollman**, you can play the dice game (similar to the Chinese aeroplane chess or the Monopoly). What's more interesting is that you can DIY the clothes of the Rollbot and even DIY the steps it goes on the map by programming. Use your brain and enjoy the fun!



# Components List

Note: The Rollbot wears itself in three colors: red, yellow, and black. Here we take a red rollbot as an example.

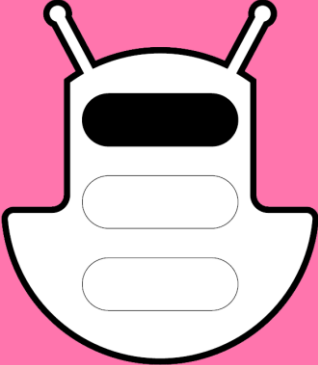
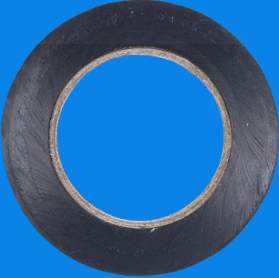

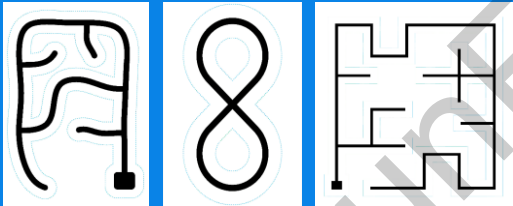


## 2.1. Mechanical Fasteners

Component	Name	Quantity
	M2*10 Screw	4
	M2 Nut	4
	M3*8 Screw	2
	M3 Nut	2
	Gasket	4

## 2.2. Electronic Components


Component	Name	Quantity
	N20 Gear Motor	2
	N20 Motor Mount	2
	Main Board	1
	OLED Screen	1
	Infrared Sensor	1

	Velcro	1
	Universal Wheel	1
	Wheel	2
	USB Type-C Cable	1
	3.7V USB Charger for Li-Po Battery	1
	Battery	1

	Test Card	1
	Black Tape	1
	Marker Pen	1
	Guide Card	3
	Double-sided Tape	1
	Cover	1

	<p>OLED Cover</p>	<p>1</p>
	<p>Map</p>	<p>1</p>

### 2.3. Tools

Accessory	Name	Quantity
	<p>Screw Driver</p>	<p>1</p>

# Software Operation

## 3.1 Get the Code

Go to our website [www.sunfounder.com](http://www.sunfounder.com), click **LEARN** -> **Get Tutorials**, under **Robot Kit** find **Rollbot**, click it and on the page click to download *Rollbot.zip*, and unzip it.

### 3.1.1 Install Arduino IDE

The code in this kit is written based on Arduino, so you need to install the IDE first. Skip it if you have done this.

Now go to the [arduino.cc](http://arduino.cc) website and click **DOWNLOAD**. On the page, check the software list on the right side under **Download the Arduino Software**.

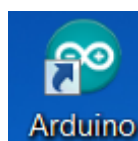


Find the one that suits your operation system and click to download. There are two versions of Arduino for Windows: **Installer** or **ZIP file**. You're recommended to download the former. Just download the package, and run the executable file to start installation. It will download the driver needed to run Arduino IDE. After downloading, follow the prompts to install.

For details of the installing steps, you can go to WIKI on our website [www.sunfounder.com/wiki](http://www.sunfounder.com/wiki).

### Install Arduino Software

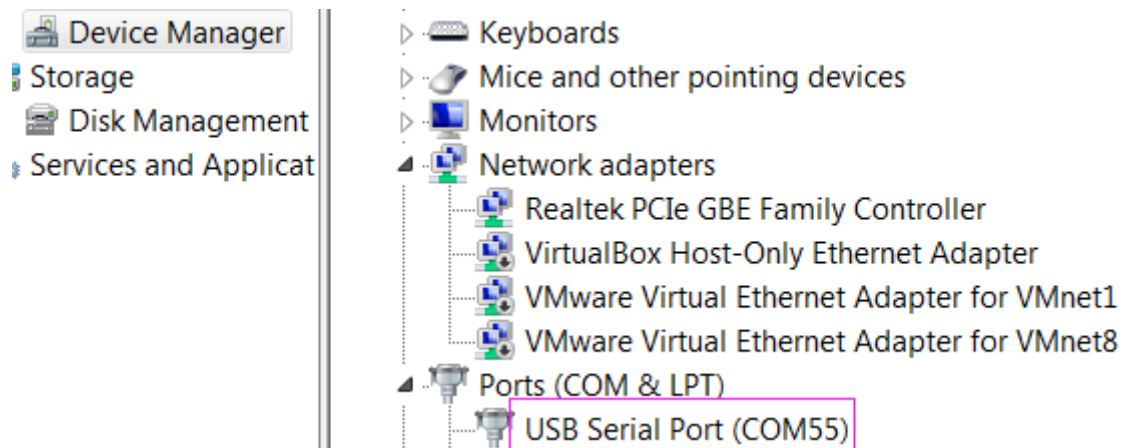
After installing, you will see the **Arduino** icon on your desk and double click to open it.



### 3.1.2 Connecting Rollbot to Your Computer

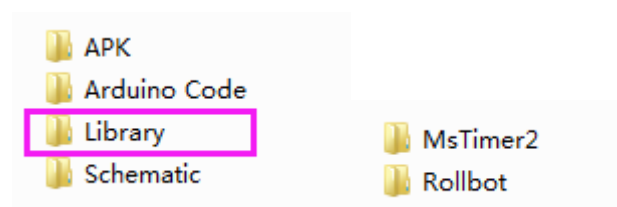
Use a USB Type-C cable (equipped in the kit) to connect the Rollbot to the computer. Once you've plugged in, the LEDs on the board will light, and the computer will install the necessary

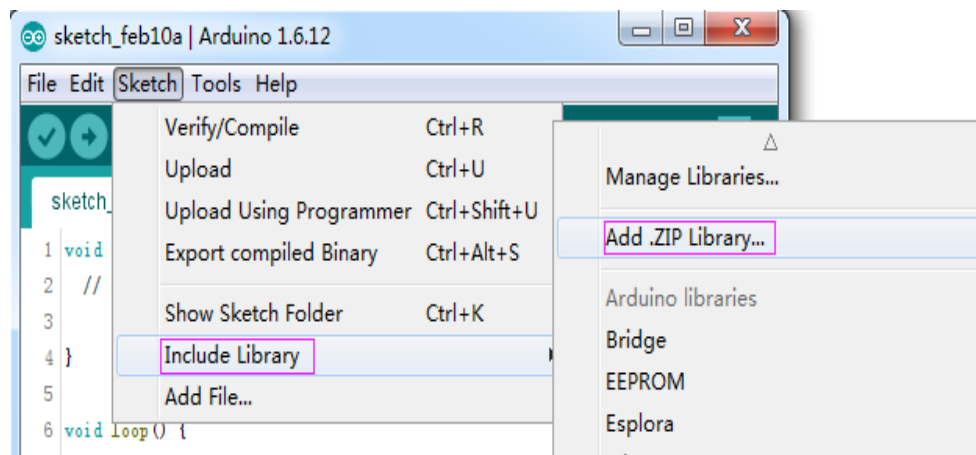
driver automatically. Windows users can view the port in the device manager: **USB Serial Port (COM55)**; your port must be different from mine.



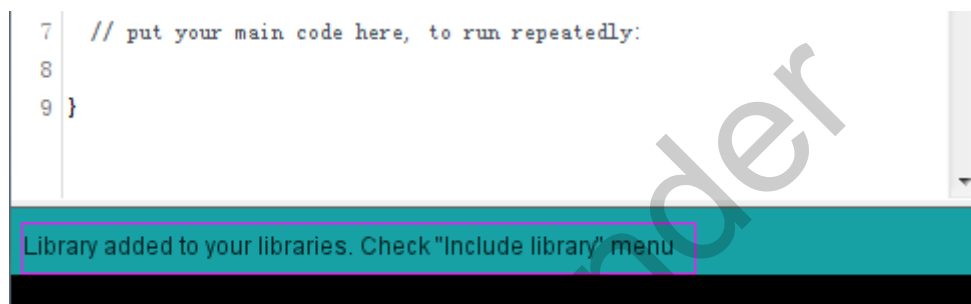
### 3.2 Add Libraries

- 1) Libraries are a collection of code that makes it easy for you to connect to a sensor, display, module, etc. In this kit, you need to add two libraries to the Arduino libraries folder: **FlexiTimer2** and **Rollbot**.
- 2) Select **Sketch -> Import Library -> Add Library**. Find the *MsTimer2* library under the *libraries* folder in the Rollbot. Click **Open**.

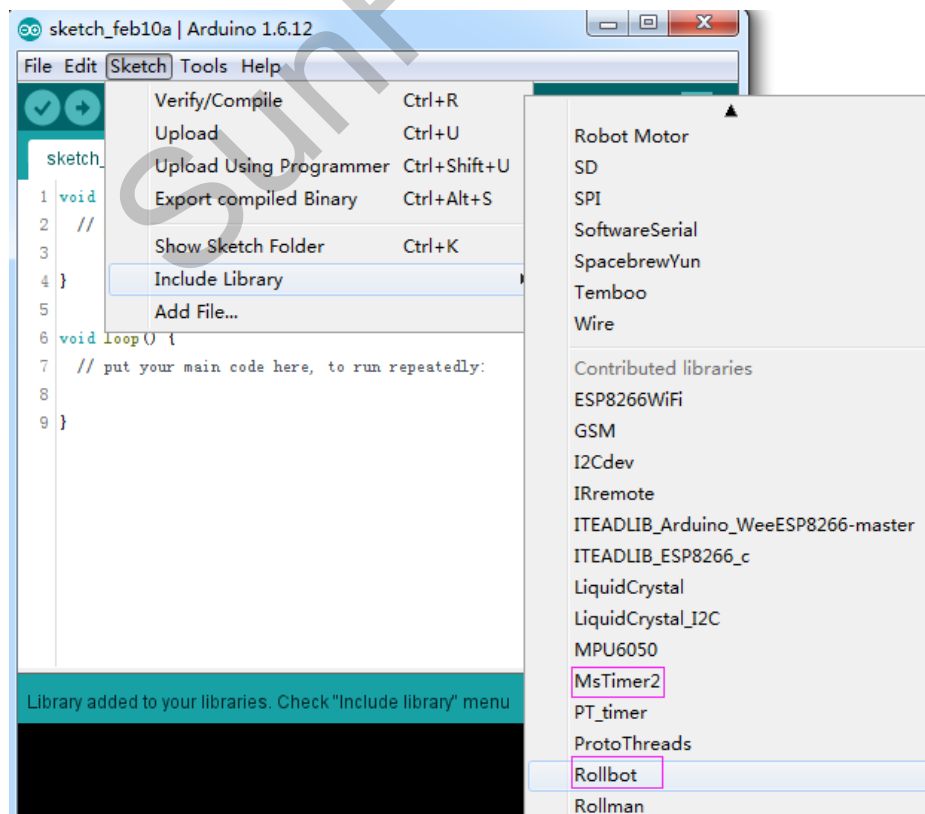




- 3) Here you should see the library has been added to your libraries. Check it by **Sketch -> Include Library**



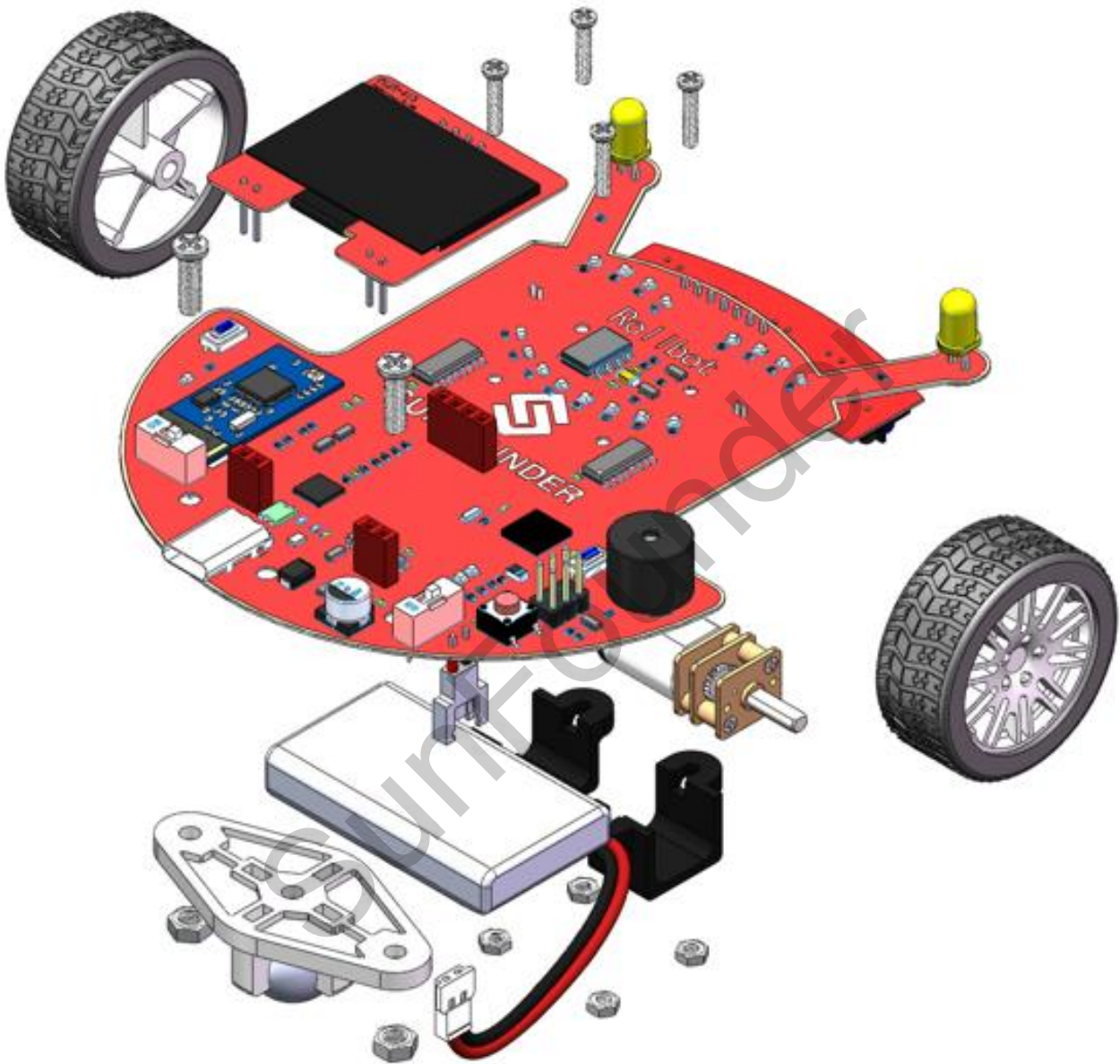
- 4) Import the Rollbot library from the *libraries* folder in the same way.
- 5) Here you should see the libraries added to your libraries. Check **Include Library** and the libraries just imported have appeared on the list.



For more information about Arduino, visit: <http://arduino.cc/> and <http://www.sunfounder.com/>.

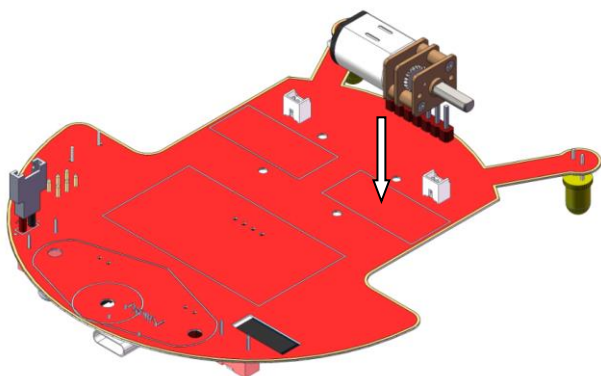
## Assembly

You can see the general assembling procedures through the exploded view. On the whole, the assembling is easy.

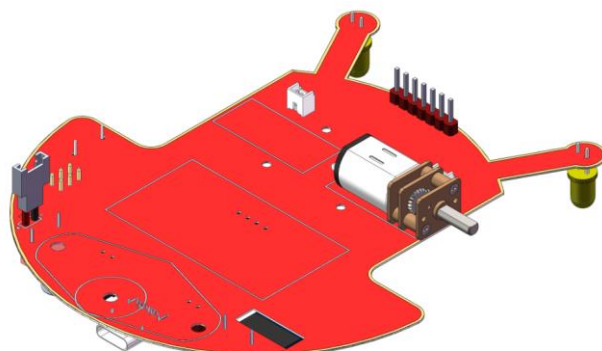


## 4.1 Main Board + N20 Gear Motors

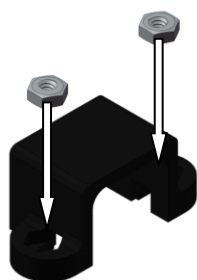
1. Align the N20 gear motor with the areas in the lines of the mainboard (the car body).



2. Put the gear motor to the corresponding area of the main board and align the outside edge with the edge of the main board.



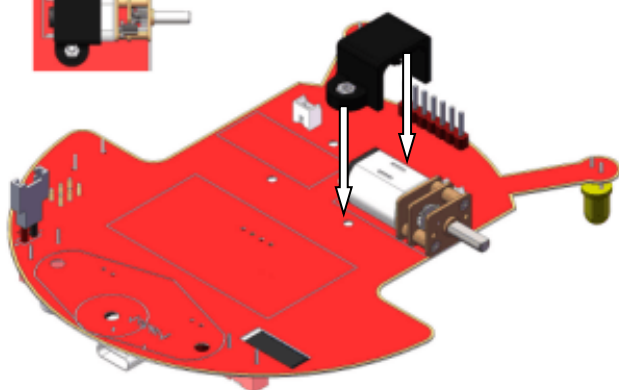
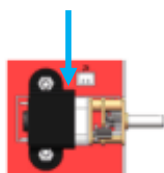
3. Align two M2 nuts with the holes of the motor mount.



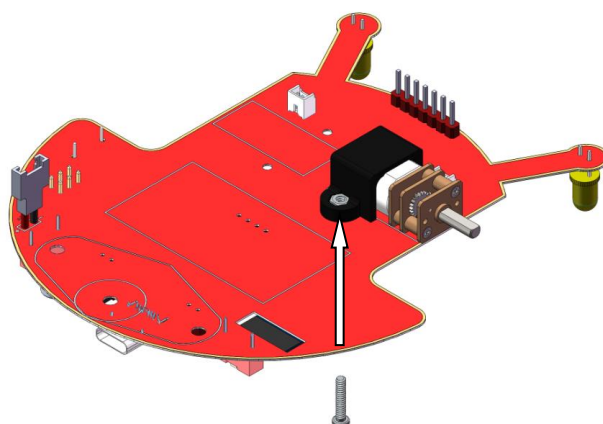
4. Place the nuts in the holes, and be careful about not falling out.



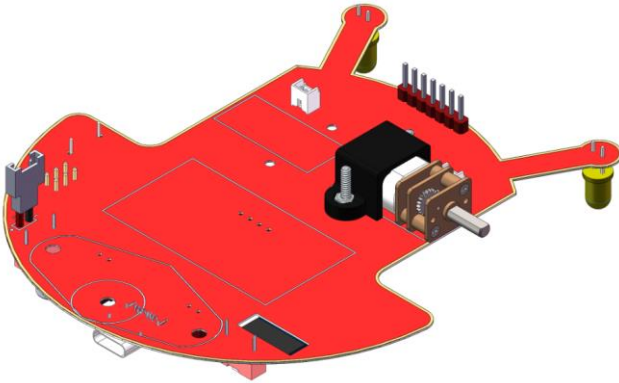
5. As shown in the figure, the bulgy parts of the motor mount should be placed towards the outside of the main board. Align the holes.



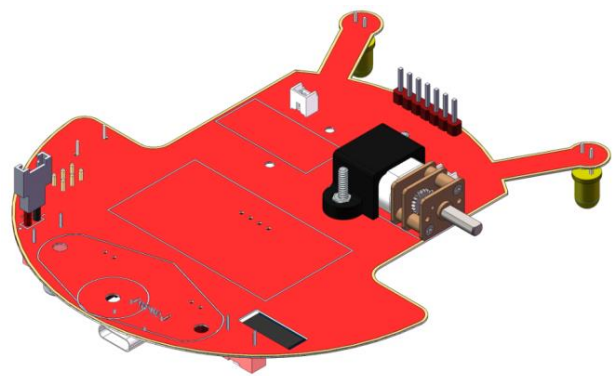
6. Put the motor mount on the board and align the holes. Then plug 2 M2\*10 screws through and fasten them lightly (not too tight for now).



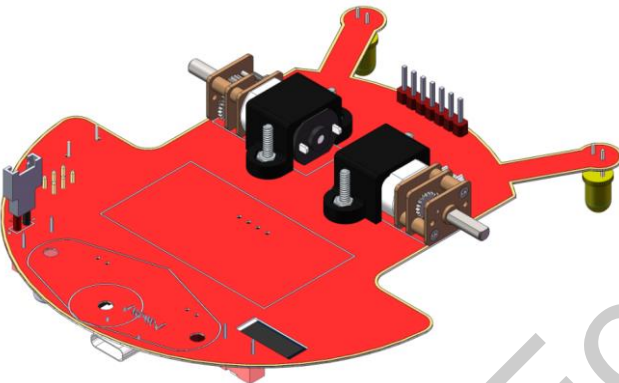
7. Similarly, plug another screw into the nut on the mount and fasten it.



8. Now back to the first screw and tighten it.



9. Fasten the other motor in the same way.

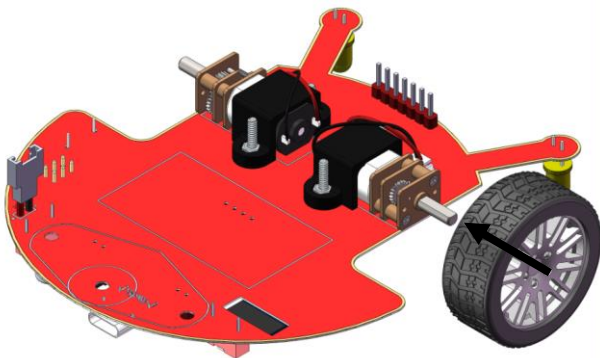


10. Connect the anti-reverse cable of the N20 motor on the left to the J1 port (just beside it) and the right one to J3 respectively.

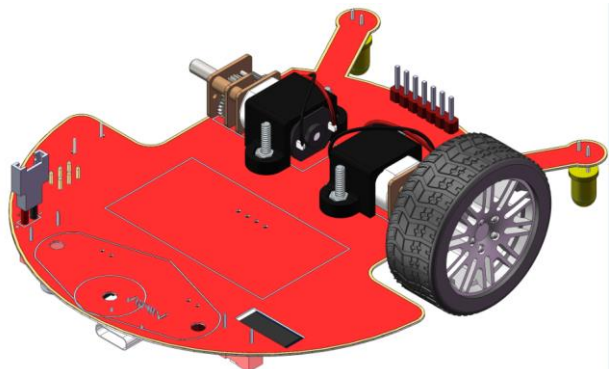


## 4.2 N20 Gear Motors + Wheels

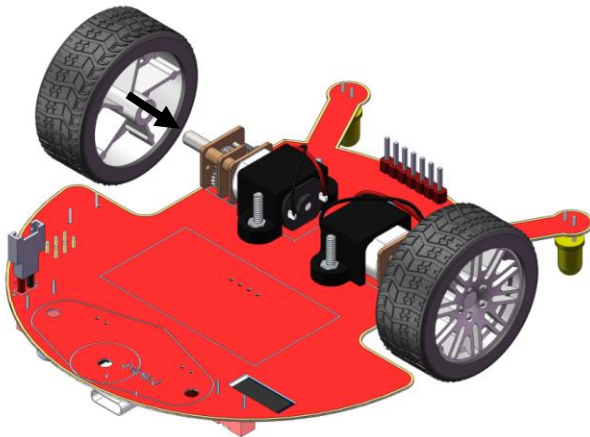
1. Insert the wheel into the shaft of the motor. Note: Press the motor on the board tightly with fingers before inserting, in case of any movement.



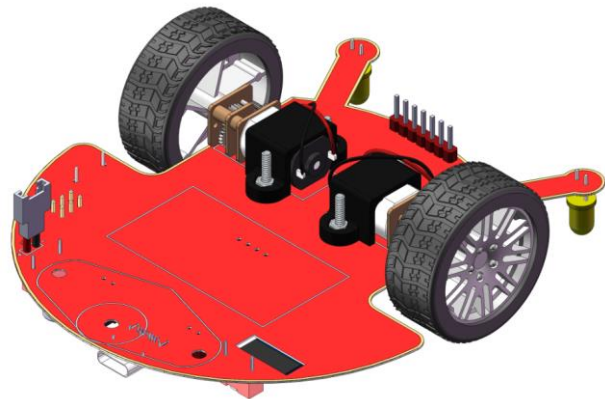
2. As shown below, don't put the wheel too close to the board edge as the wheel movement will be impeded.



3. Insert the other wheel in the same way.

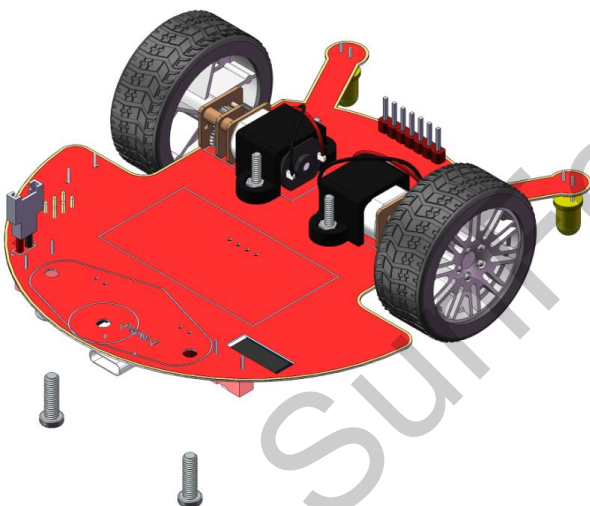


4. The completed assembly should be as shown below.

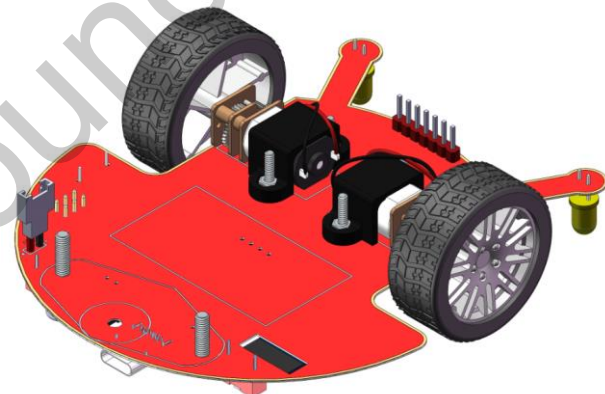


### 4.3 Main Board + Universal Wheel

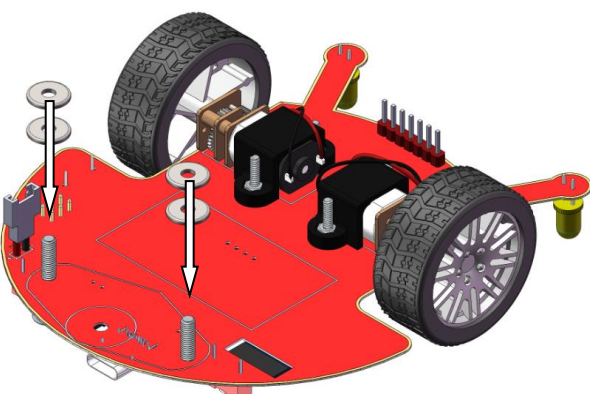
1. Align two M3\*8 screws with the holes of the main board from its upside.



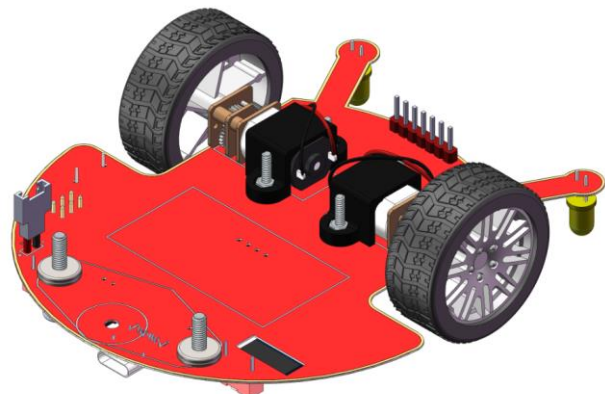
2. Insert the screws across the board and prop them with fingers.



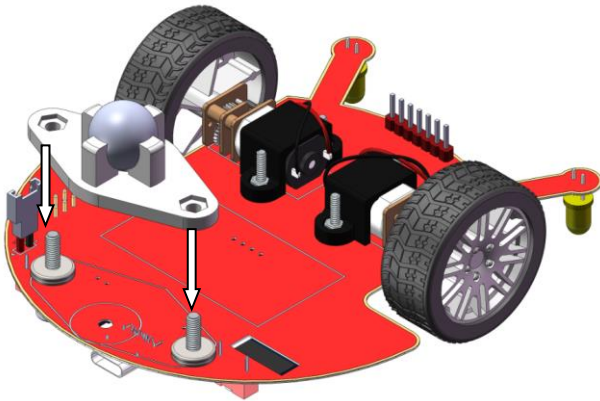
3. Put two 3x9x0.9 gaskets into each of the two screws on the other side.



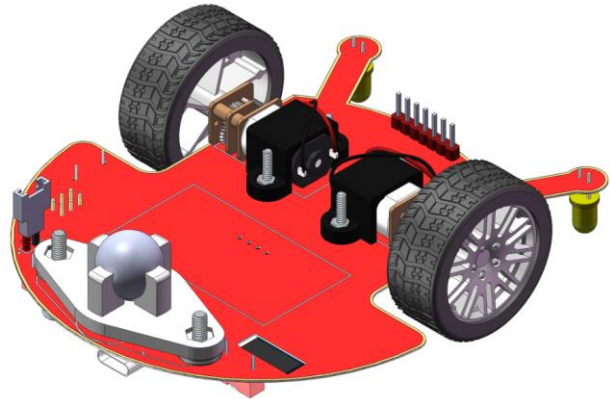
4. It would be shown as below.



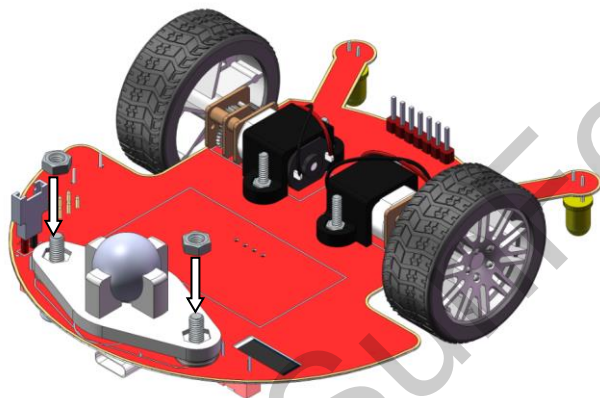
5. Align the fixed holes of the universal wheel with the screws.



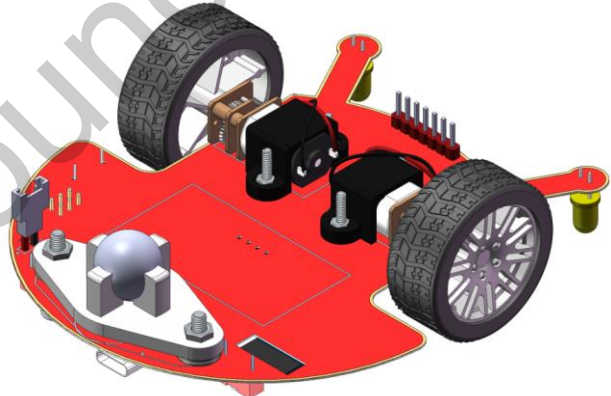
6. Put the universal wheel onto the gaskets. So the screws go through the holes of the board, gaskets, and holes of the universal wheel.



7. Put two M3 nuts into the screws. Screw a little and then let the screw sink a bit till the nuts are fixed in the hole of the universal wheel.



8. Fasten the screws with fingers pressing the nuts. Then the universal wheel is fixed.

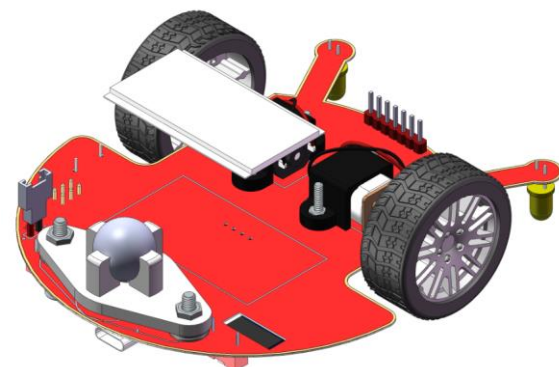


#### 4.4 Main Board + Battery

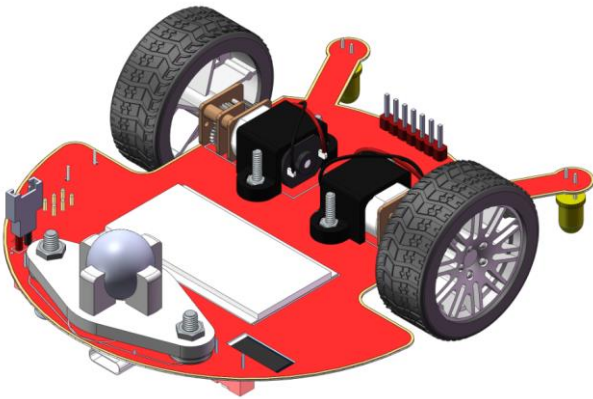
1. Tear off the transparent film of the Velcro.



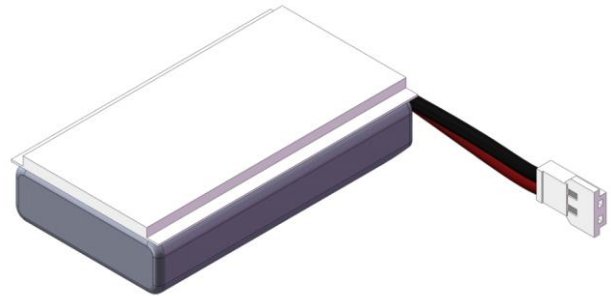
2. Align one of the Velcro with the area with lines on the main board, as shown below.



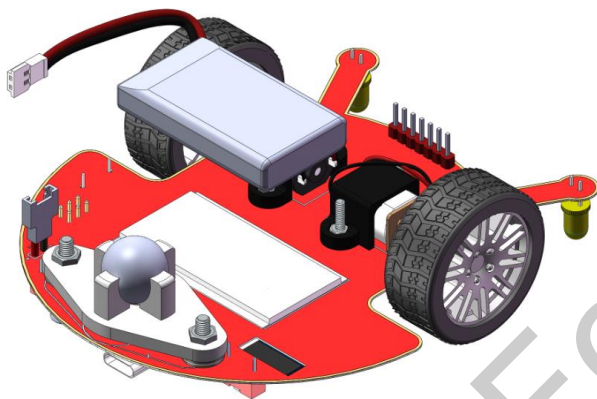
3. Paste it and press tightly.



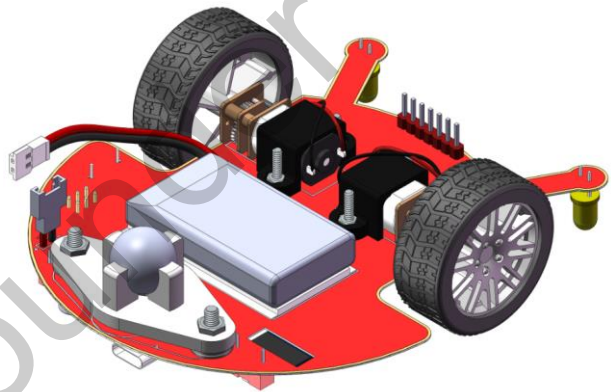
4. Paste the other Velcro on the back of the battery, as shown below.



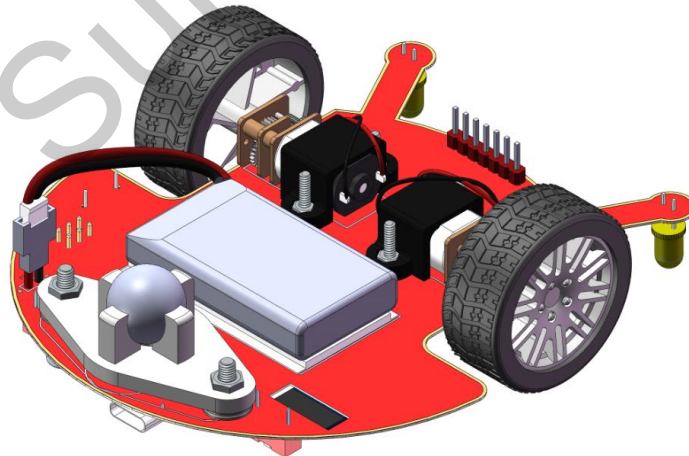
5. Align the Velcro on the battery with that on the main board.



6. Paste and press it tightly.

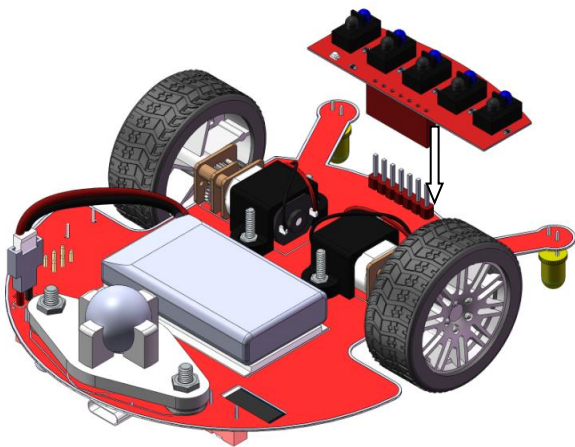


7. Connect the anti-reverse cable of the battery to the 2-pin port on the main board.

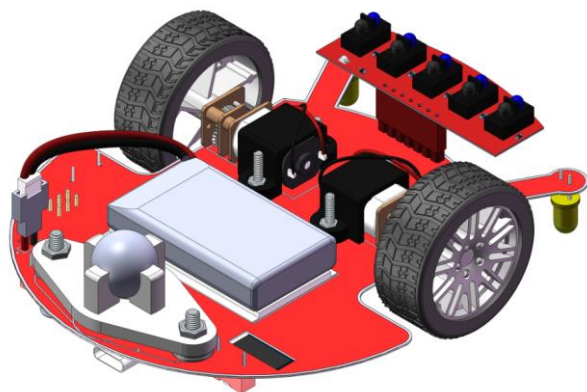


#### 4.5 Main Board + Infrared Sensor

1. Align the pins on the infrared sensor with the headers of the main board.

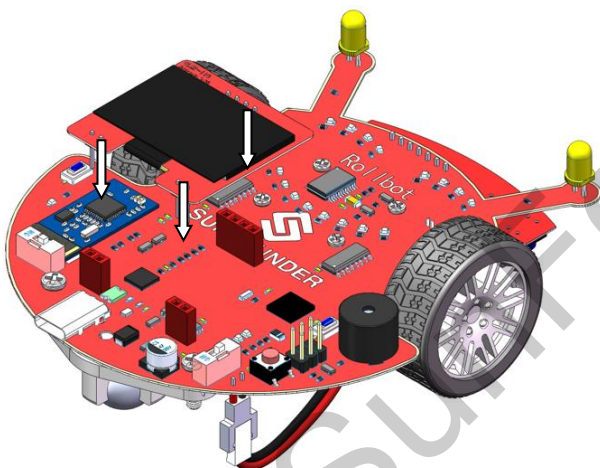


2. Insert the pins into headers and press them tightly.

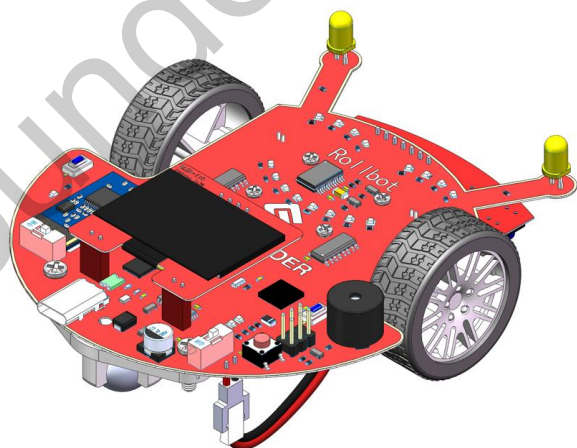


#### 4.6 Main Board + OLED Screen

1. Align the pins of the screen with the headers of the main board.



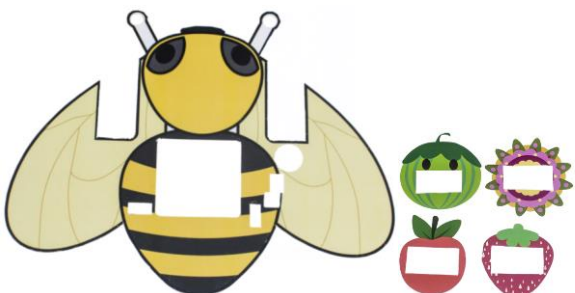
2. Insert the pins and press them tightly.



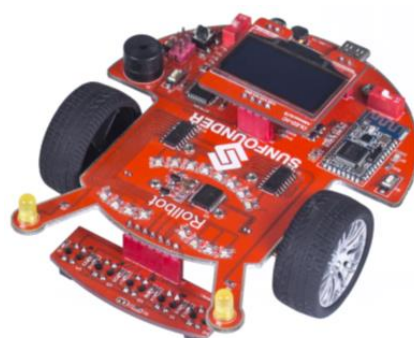
#### 4.7 Rollbot + Covers

The covers in the kit are provided for decoration. You can choose to assemble them or not. Here we will show you how to assemble.

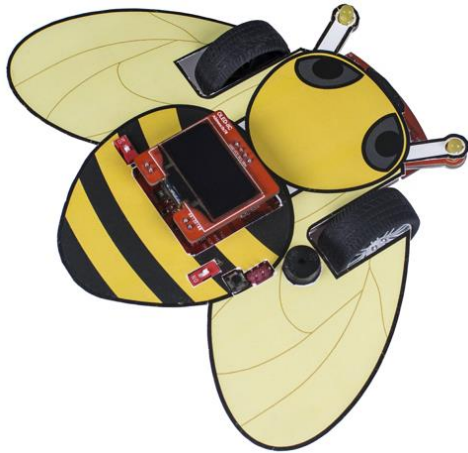
1. The bee cover and OLED screen covers provided are to make the Rollbot more adorable and harmonious on the map.



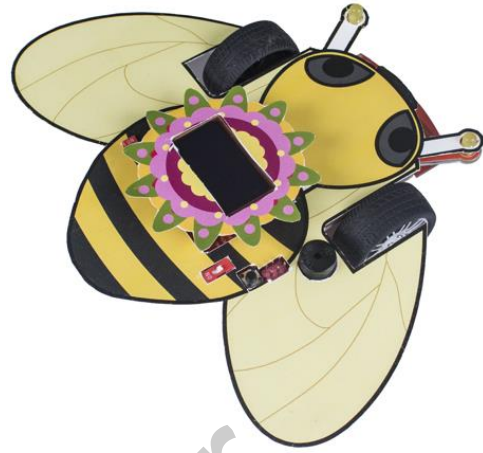
2. Stick some double-sided tape on the main board and smooth surface around the OLED.



3. Fit the bee cover onto the car, leave bulges through the holes. Press the cover to stick it on the car.

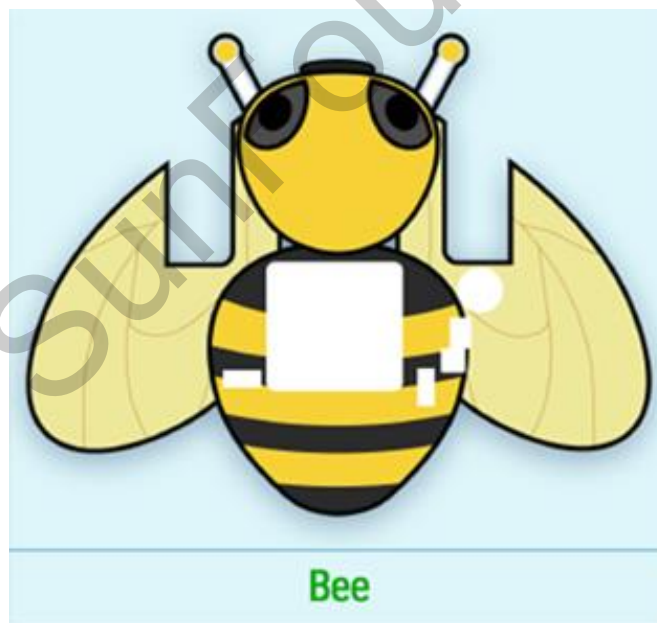


4. There are four OLED covers in the kit. Choose your favorite and stick it on.



#### 4.8 DIY Covers

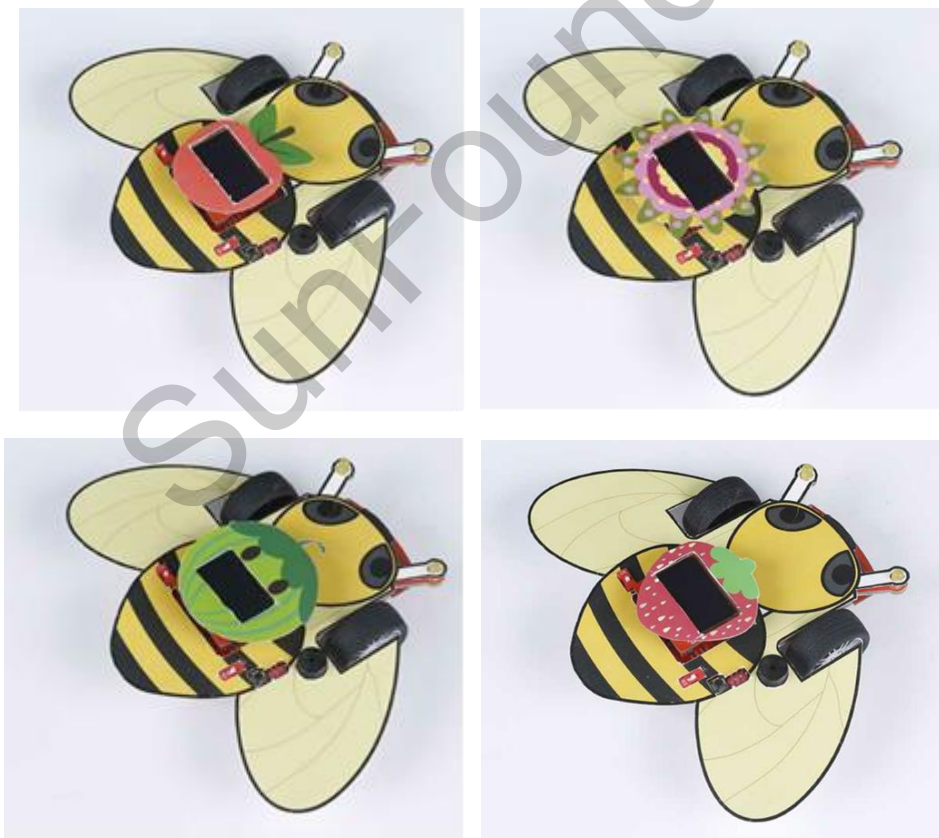
The cute bee cover is provided in the kit. You can dress your Rollbot up to make it more adorable.



You can use the double-sided tape to stick the fruit and flower covers on the OLED screen of the Rollbot. It'll just feel like the bee is carry an apple, a watermelon (hmm, seems too heavy, LOL...), or a blossoming flower. The Rollbot wearing the covers would seem most fit when running on the forest map included in the kit. Just give it a try!



Pretty cute, right?



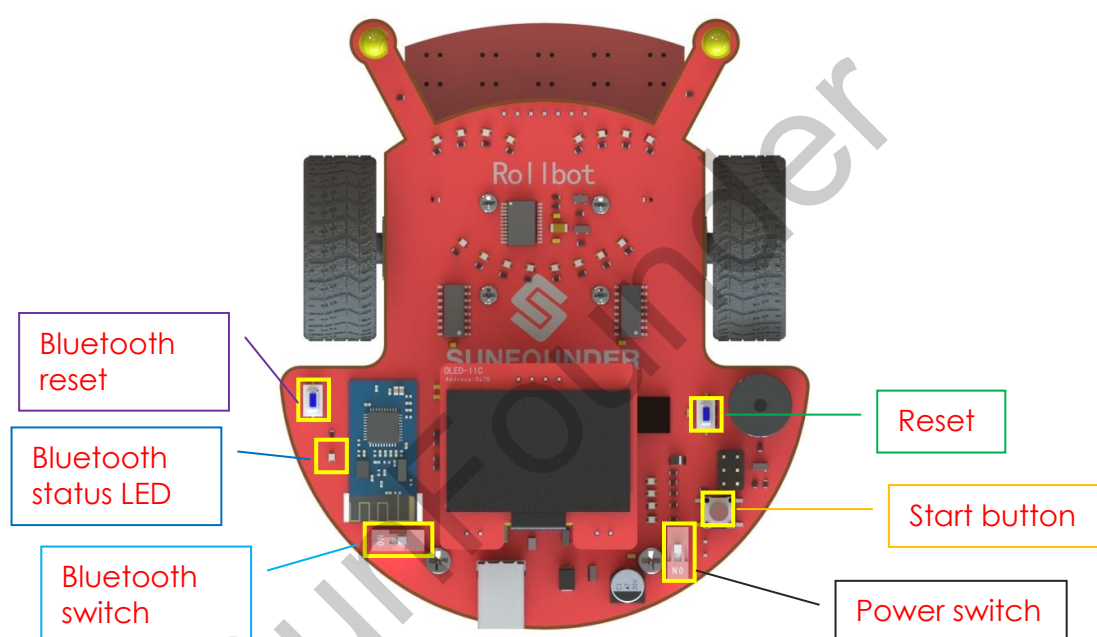
What makes it more interesting is the DIY part - You can design different "clothes" for the Rollbot and welcome to share on our website [www.sunfounder.com/](http://www.sunfounder.com/).

# Gameplay

After dressing the Rollbot with pretty covers, let's start the most exciting journey: games!

With the basic operation of the Arduino IDE and the circuit theory of the car previously, you should already have a general understanding of the robot car. Next, let's move on to the specific applications and the corresponding actions of the robot. Note that when we download the APP for the robot, we need to turn off the power of the Bluetooth module if it has been powered. Otherwise we are not able to download the APP.

In subsequent operations, you need to know some basic buttons on the main board:



**Power switch:** Switch on and off the robot; after switching on, the OLED screen and LEDs on the board will light up. When it's off, the whole robot will be shut down.

**Start button:** To start the program, especially in the line following function.

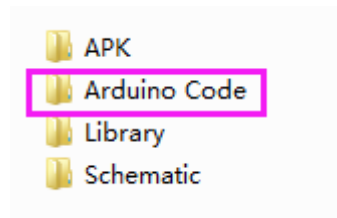
**Reset button (S1):** To reset the master board.

**Bluetooth switch:** Switch on and the Bluetooth will be enabled on the board.

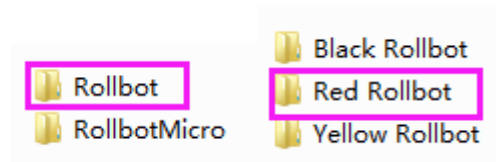
**Bluetooth status LED (D7):** When the Bluetooth is open and enabled, it'll start to blink; after it's connected to another Bluetooth device like an Android mobile, the LED will keep constant lighting.

**Bluetooth reset button (S2):** To reset the Bluetooth.

Open the *Rollbot* folder downloaded and unzipped previously in **Chapter 3 Software Operation**. You can see the following folders inside:



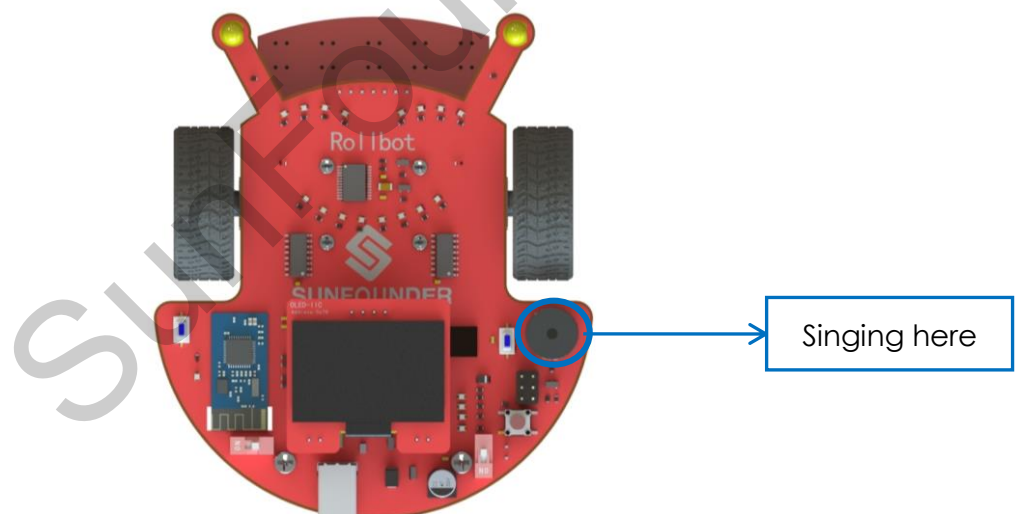
Inside *Arduino Code*, there are three folders for the specific robot in a certain color and type.



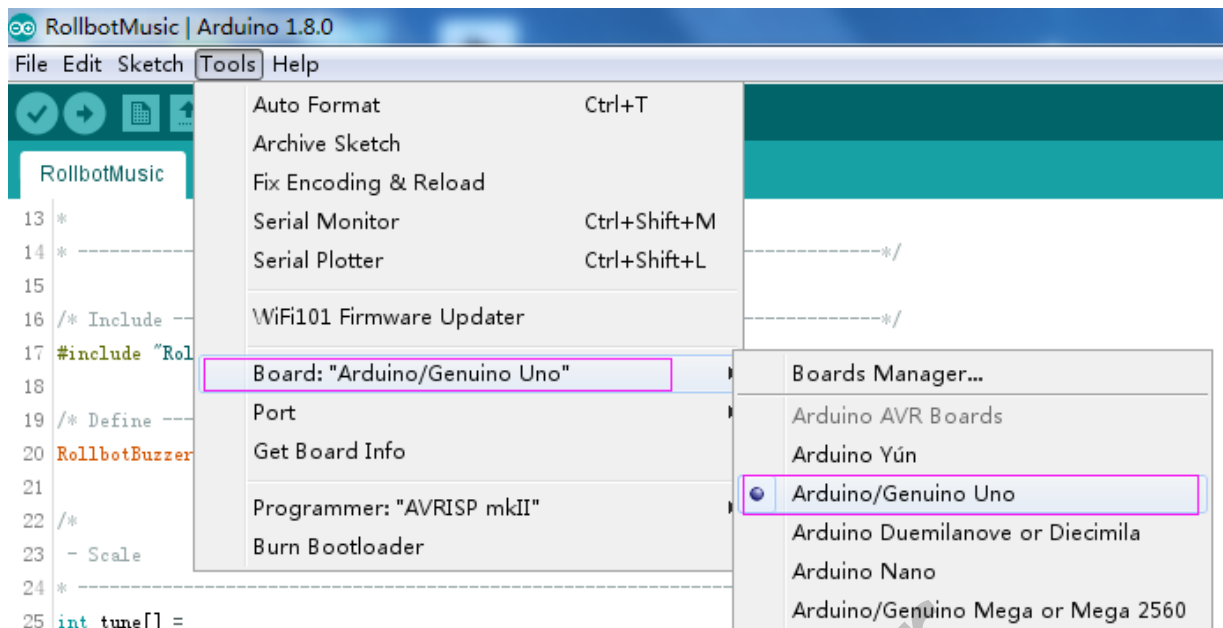
We just take the **red** one as an example in the following part. You can program your Rollbot correspondingly in the same way.

### 5.1 Singing

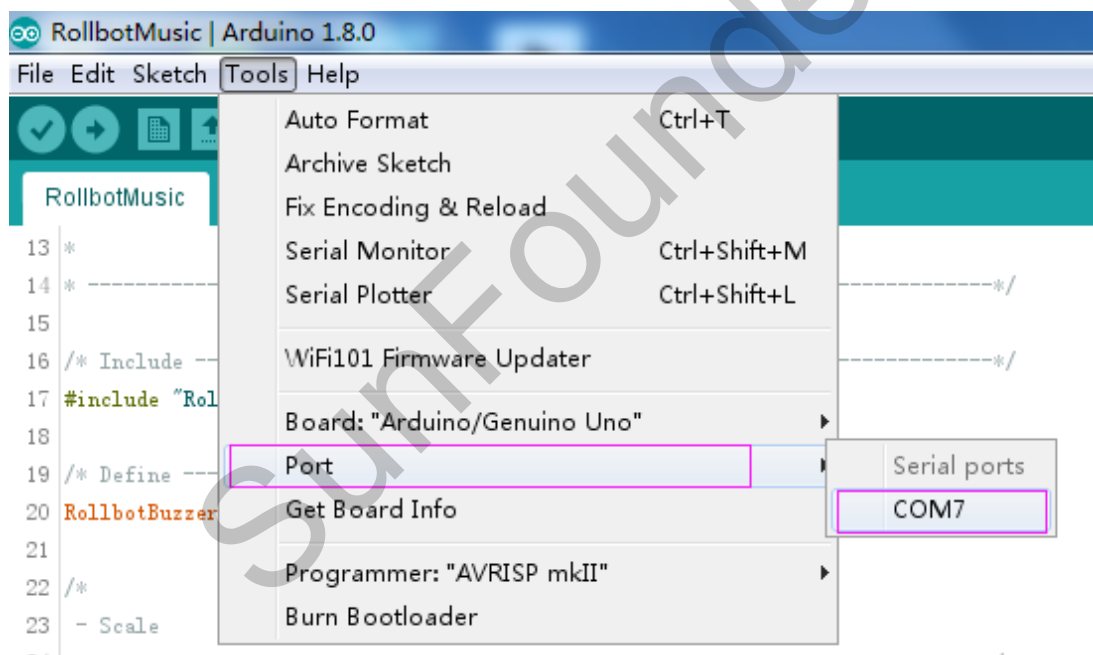
The Rollbot can sing with the buzzer onside via programming. Please see **7.1 Singing with buzzer** for detailed explanation of the code. After reading the explanations, you can modify the music as you like.



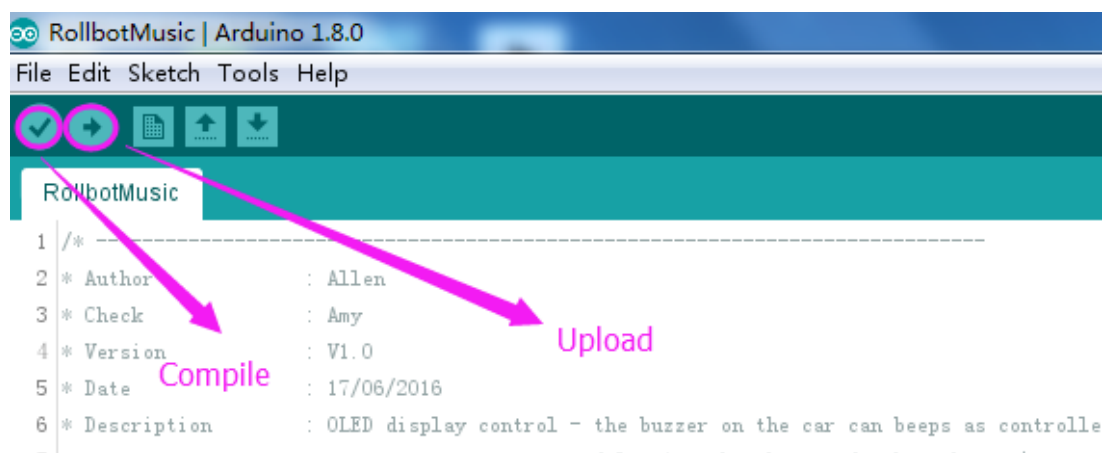
- 1) Find *RollbotMusic.ino* under *Rollbot\Arduino Code\Rollbot\Red Rollbot\RollbotMusic*, double click to open it. You can also open the file by clicking **File** -> **Open** in Arduino.
- 2) Click **Tools** -> **Board**, select **Arduino/Genuino Uno**



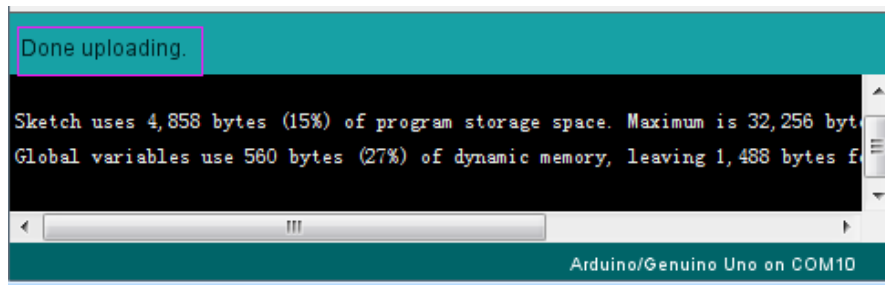
and then, click **Tools** -> **Port**, select **COM7** (depending on your actual situation).



- 3) Click the **Compile** button to compile the code, click **Upload** to upload the code to the board (or directly, skip **Compile**).



- 4) Wait for a moment until the following information appears at the bottom of the window, which indicates it is uploaded successfully.

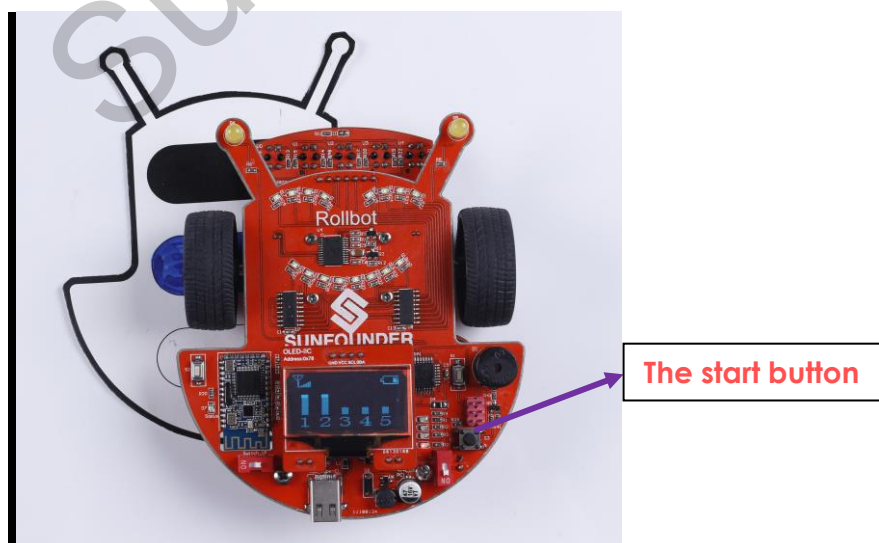


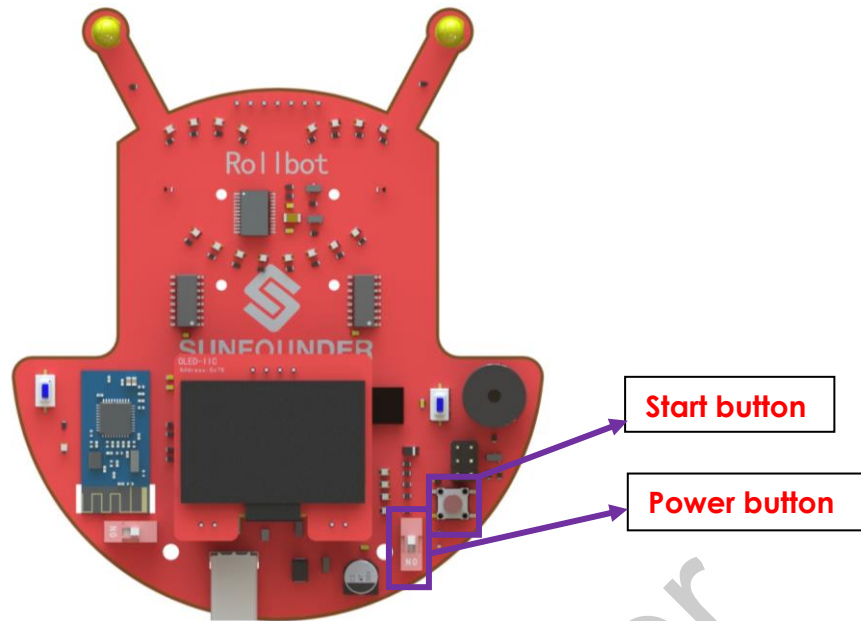
- 5) Remove the USB cable, switch on the power. The buzzer of the car will sing. Certainly, you can play the music by programming, and the specific music theory and the related code will be explained in the Chapter 7.1.

## 5.2 Display the Sensor Signal Intensity on OLED

This experiment is to display the collected sensor data on the OLED screen. Here it is just applied for simple display. Later in the line following experiment, we will use the intensity to control the direction of the car.

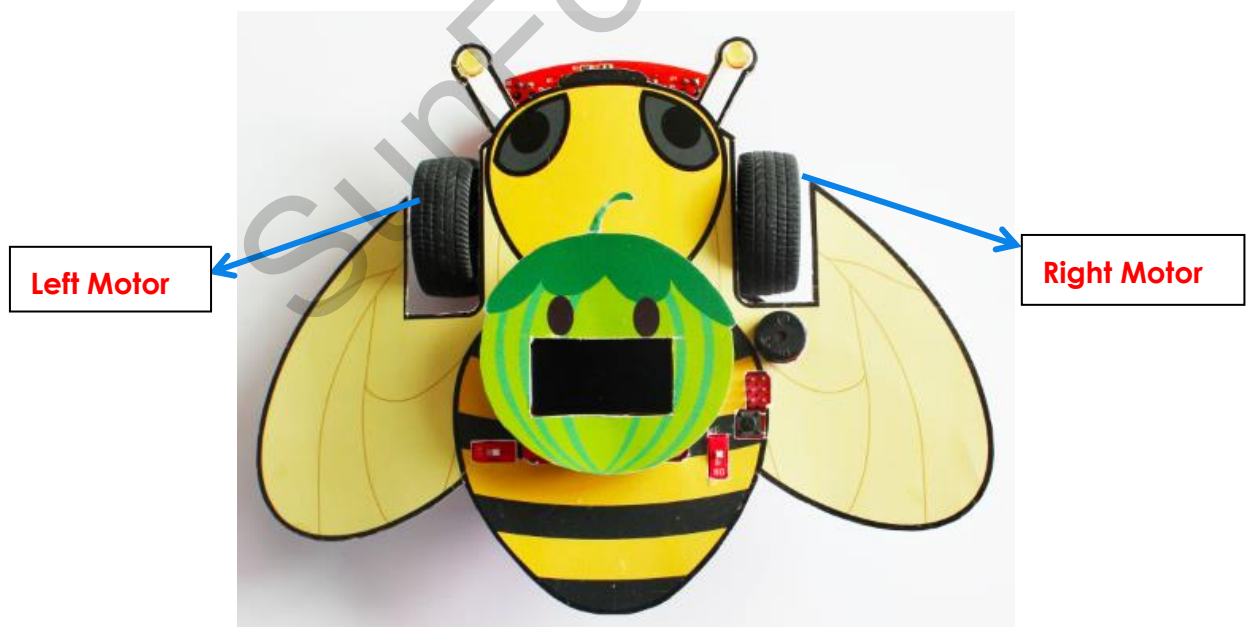
- 1) Find *RollbotOLED.ino* under *Rollbot\Arduino Code\Rollbot\Red Rollbot\RollbotOLED*, open it and upload to the Rollbot.
- 2) Remove the USB cable, switch on the power. Put the test card under the sensor module of the robot. Press the **start button** for more than 2 seconds to enter the debugging mode. You will see that on the OLED screen, there are five bars simulating the signal intensity of the 5 detectors. Now put the first and second detectors from the left over the black stripe. So the 1 and 2 bars show the strongest signal, as shown in the figure below:





### 5.3 Motor Testing

This experiment is to test whether the motor wiring is right or not. There can be 4 situations: the wiring of both motors is normal; both motors are wired inversely; the left one is normal but the right one is wired inversely; and the right one is good while the left one is wired inversely. We will introduce how to distinguish the cases then.



First, find *MotorTest.ino* under *Rollbot\Arduino Code\MotorTest* and upload to the Rollbot and observe the car.

**NOTE:** Please remember **Speed\_Dir** since it will be useful in the programs related to the motors later.

1) If the robot goes forward, it indicates all motors are wired normally. There is no need to modify the value of the variable **Speed\_Dir** in the program since it's **0** by default.

- 2) If the robot goes backward, it means that both motors are wired inversely. You need to change the variable value 0 of `Speed_Dir` into **1** to run the program normally.
- 3) If the robot rotates clockwise (always towards its right side), it indicates the left motor is wired normally while the right one is done reversely. You need to change the value 0 of `Speed_Dir` into **2**.
- 4) If the robot spins anti-clockwise (always towards its left side), it indicates the right motor is wired normally while the left one is reverse. Change the value 0 of `Speed_Dir` to **3**.

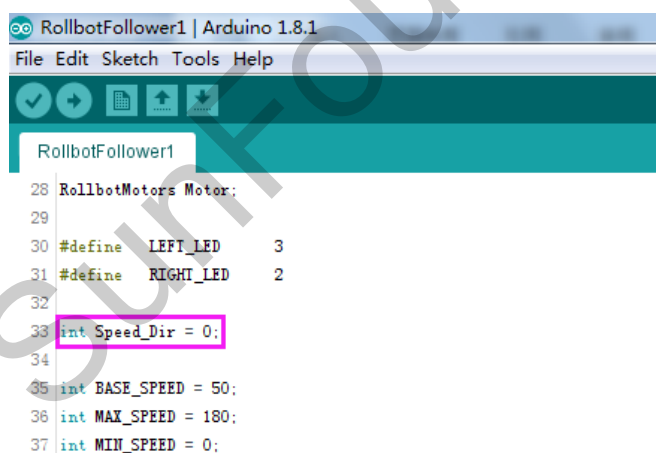
## 5.4 Line Following

This experiment is to let the robot goes forward along the black line. OLED displays the signal strength that 1, 5 sensor received. When the robot turns left, the yellow LED on the left lights. The yellow LED on the right lights when the robot turns right. No LED lights when the robot goes straight.

### 5.4.1 Magic Forest

- 1) Open `RollbotFollower1.ino` under `Rollbot\Arduino Code\Rollbot\Red Rollbot\RollbotFollower1`

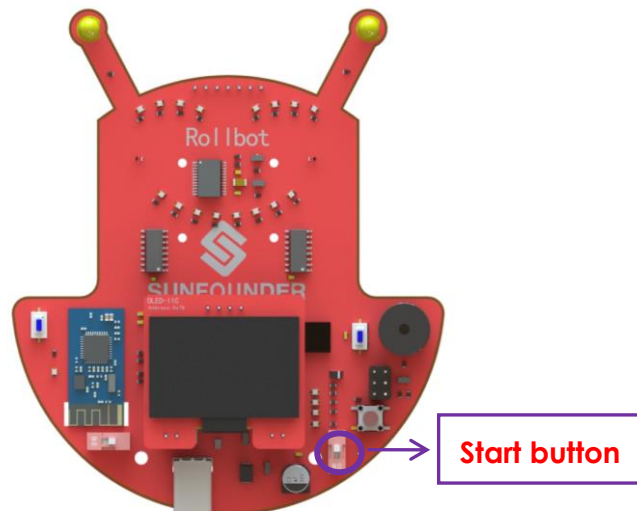
**Note:** You need to modify the value of `Speed_Dir` according to the test in **5.3**. It's 0 by default. Then upload the `RollbotFollower1.ino` to the Rollbot.



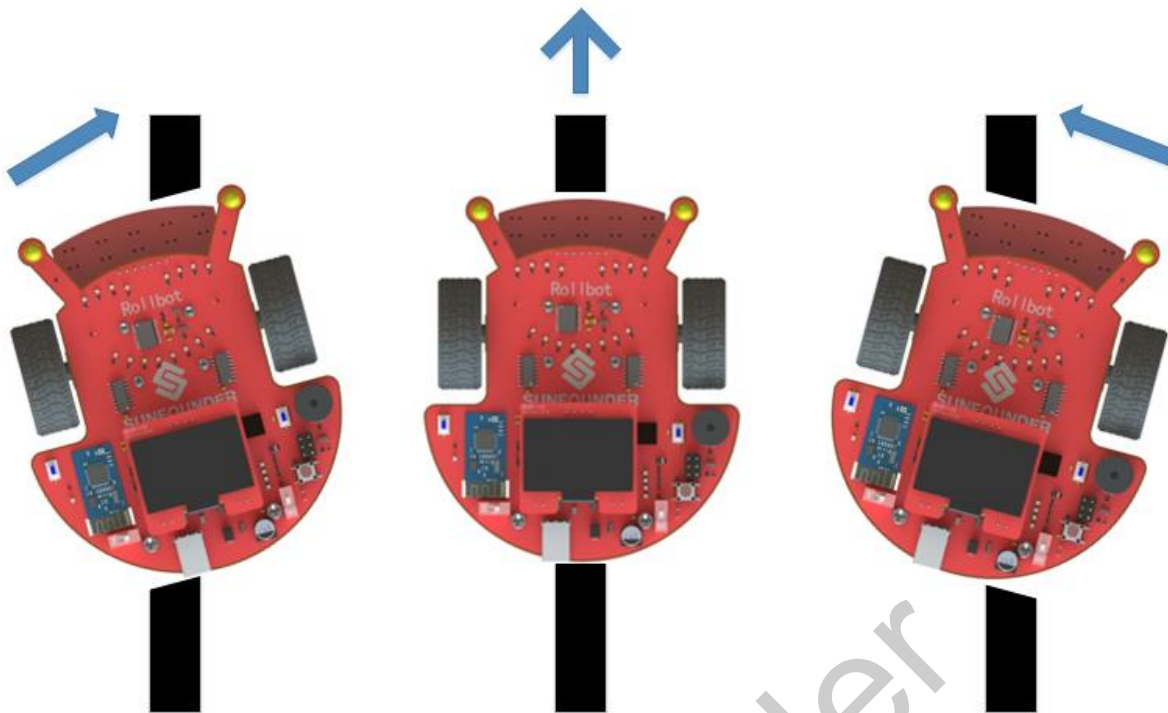
```

28 RollbotMotors Motor;
29
30 #define LEFT_LED 3
31 #define RIGHT_LED 2
32
33 int Speed_Dir = 0;
34
35 int BASE_SPEED = 50;
36 int MAX_SPEED = 180;
37 int MIN_SPEED = 0;
  
```

- 2) Take out the map and put it on a smooth surface like a table. Turn on the robot and place it with the middle detector over the black line on the map. Press the **start button** for more than 2 seconds. The robot will test the reference value of the black line automatically. Just wait patiently till it backs to the initial position after swaying a bit. This process is to test the sampling value of the black line. Then the robot will move along the line.



- 3) You will see the robot goes following the black line on the map. When the robot swings to the left relatively from the black line, the robot will turn towards the right to correct its direction and the LED on the left will keep blinking until it's back on track; when the middle detector is right over the middle of the black line, the robot goes forward straightly; when it deviates to the right, it will turn left to correct and the LED on the right will blink. See the figure below.



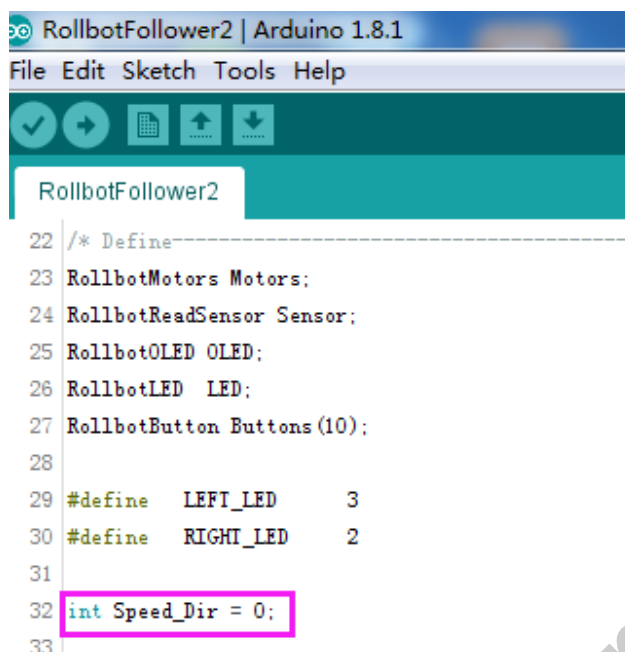
- 4) If you have mastered the line-following program, you can try modifying the PD parameters to learn more the principle of line-following. You can do this through the Android APP **Rollman** (find the Rollman.apk under Rollbot\APK and install it on the mobile). After test, the debugging process is as follows: Set the PD to 0 and increase P slowly. When the robot starts wiggling side to side, increase the D value slowly to make the motion of the robot flat. So the PD adjustment is done successfully.

#### 5.4.2 DIY Map

On a blank paper, you can draw a map you like according to instructions on the guide card. Three patterns are provided and you can also design by yourself. Then use the black tape to paste the map you designed on a flat floor or draw with the black Marker pen.

- 1) Open `RollbotFollower2.ino` under `Rollbot\Arduino Code\Rollbot\Red Rollbot\RollbotFollower2`.

**Note:** You need to modify the value **Speed\_Dir** according to the test in **5.3**. It's 0 by default. Then upload the `RollbotFollower2.ino` to the Rollbot.



```

22  /* Define-----
23  RollbotMotors Motors;
24  RollbotReadSensor Sensor;
25  RollbotOLED OLED;
26  RollbotLED LED;
27  RollbotButton Buttons(10);
28
29  #define LEFT_LED 3
30  #define RIGHT_LED 2
31
32  int Speed_Dir = 0;
33

```

- 2) Put the completed map on a smooth surface like a table. Turn on the robot and place it with the middle detector over the black line on the map. Press the start button for more than 2 seconds. The robot will test the reference value of the black line automatically. Just wait patiently till it backs to the initial position after swaying a bit. This process is to test the sampling value of the black line. Then the robot will move along the line.

You can try more possibilities such as using a white chalk to draw maps on a dark smooth surface like the floor, set a gift at the end, and then wait for the Rollbot to discover the mysteries. The program for using chalk to draw the map is not provided in our data package. You can try to change the `SignalValue` into `Sensor.Read_WhiteFlag()` to explore.

## 5.5 Bluetooth App Controls the Robot

This experiment aims to use an Android APP to control the Rollbot. You can also play the dice game (similar to Ludo and Monopoly) with the APP (the map is **NOT** provided in the kit and you have to purchase it).

### Bluetooth connection

- 1) Turn off the blue tooth switch, open `RollbotAPP.ino` under `Rollbot\Arduino Code\Rollbot\Red Rollbot\RollbotAPP`.

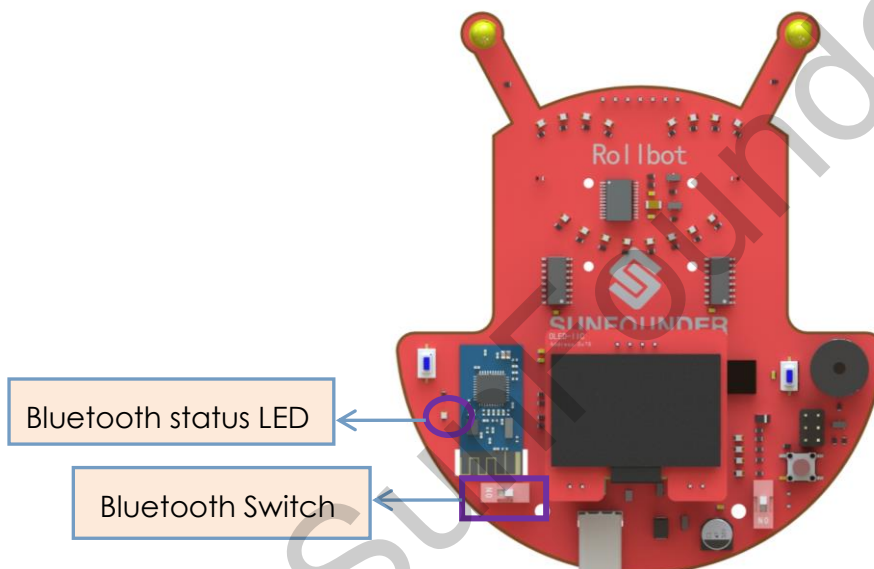
**Note:** You need to modify the value `Speed_Dir` according to the test in **5.3**. It's 0 by default. Then upload the `RollbotFollower2.ino` to the Rollbot.

```
RollbotAPP | Arduino 1.8.1
File Edit Sketch Tools Help

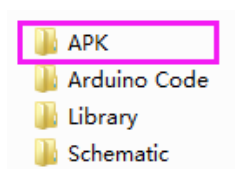
RollbotAPP


13 #include "OLEDData.h"
14
15 /* Define -----
16 RollbotMotors Motors;
17 RollbotLED LED;
18 RollbotOLED OLED;
19 RollbotButton Buttons(10);
20
21 int Speed_Dir = 0;
22
```

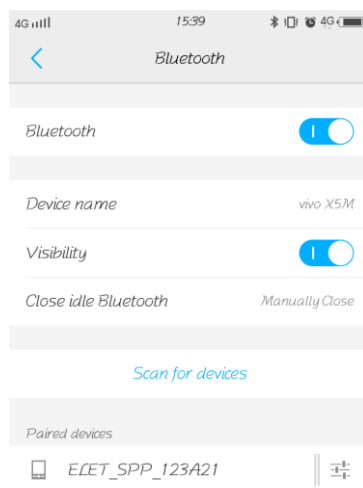
- 2) Turn on the power switch of the robot and the Bluetooth switch. When you turn on the latter, the Bluetooth Status LED will blink.



- 3) Install the Rollman app on a mobile phone (for **Android** only). Find the package *rollman.apk* in APK in the Rollbot folder.



- 4) After installing, you will see the icon , tap to open it. And it will redirect to the Bluetooth interface of the mobile, as shown below:



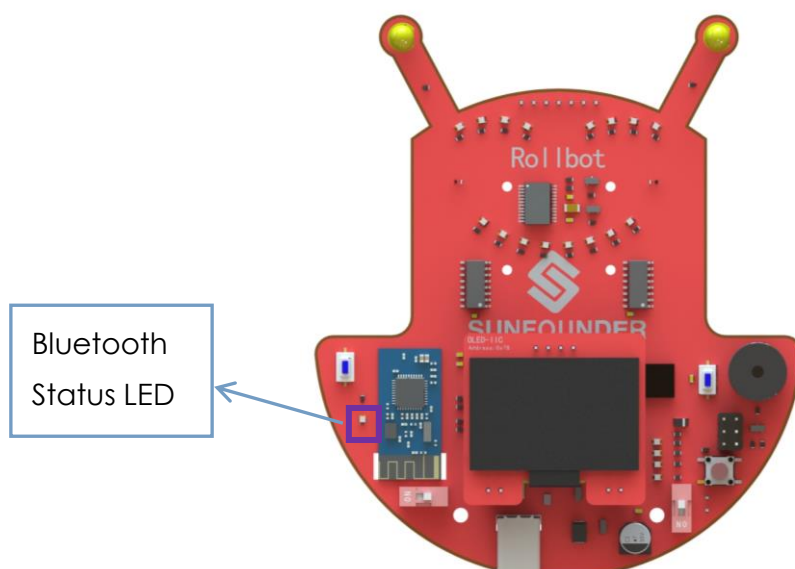
5) Now open the Bluetooth of the mobile, hit **Scan for devices** to find **Rollbot XXXXXX** to pair.

**Note:** If the pairing fails, you cannot use the APP. Try again then.

6) After the pairing is done, back to the Rollman page. Hit **Select Device** and **Rollbot XXXXXX**, then tap **Connect**.



7) If the robot has been connected to the mobile by Bluetooth successfully, the icon **Connect** in will become **Connected**. The Bluetooth Status LED in the Rollbot will keep lighting constantly instead of blinking.

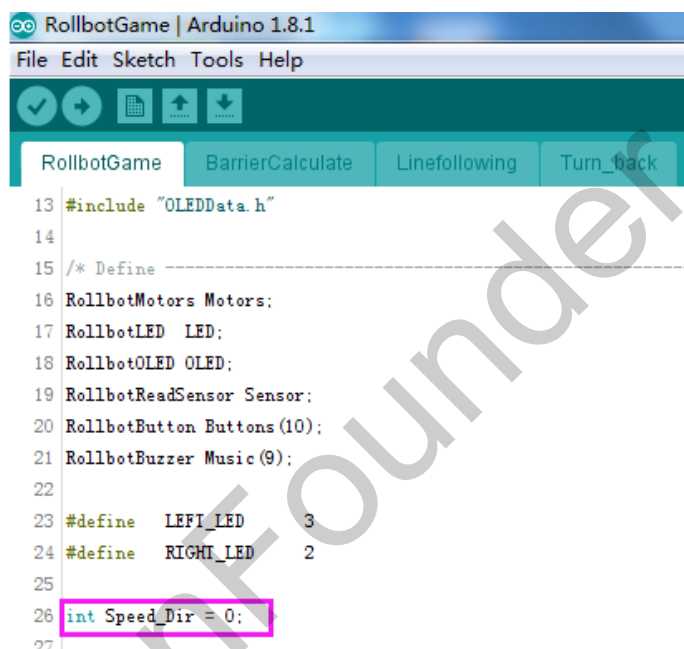


## Home page - simple control of the robot

Press the **start** button on the robot, let it in the start state, and then tap the 4 buttons on the page to control the direction of the robot.

### 5.6 Dice Game

- 1) Please refer to the previous part section **6.5** to install the app **Rollman**.
- 2) Turn off the Bluetooth switch, open *RollbotGame.ino* under *Rollbot\Arduino Code\Rollbot\Red Rollbot\RollbotGame*. You need to modify the value **Speed\_Dir** according to the test in 5.3. It's 0 by default. Then upload the *RollbotGame.ino* to the Rollbot. Turn on the power switch and the Bluetooth switch of the robot. You can see the status LED blink.



```

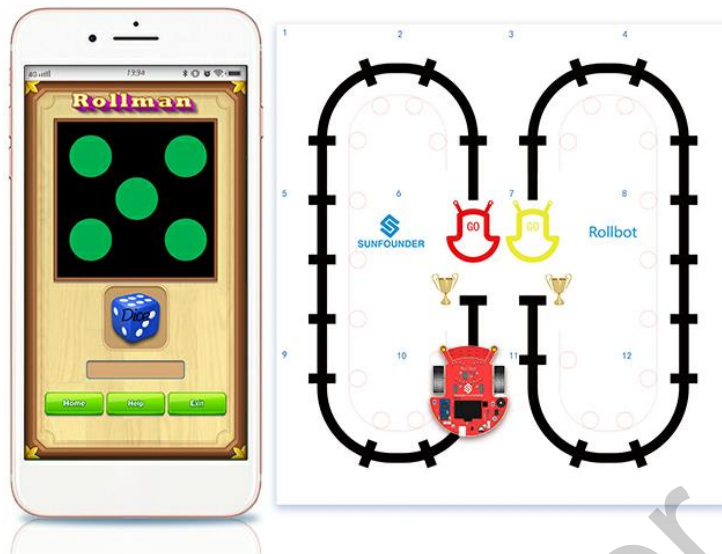
RollbotGame | Arduino 1.8.1
File Edit Sketch Tools Help
RollbotGame BarrierCalculate Linefollowing Turn_back
13 #include "OLEDData.h"
14
15 /* Define -----
16 RollbotMotors Motors;
17 RollbotLED LED;
18 RollbotOLED OLED;
19 RollbotReadSensor Sensor;
20 RollbotButton Buttons(10);
21 RollbotBuzzer Music(9);
22
23 #define LEFT_LED 3
24 #define RIGHT_LED 2
25
26 int Speed_Dir = 0;
27

```

- 3) Connect the Rollbot with your phone (refer to **6.5** for the pairing). Tap **Dice** in the lower right corner and enter the game. Press the **start button** in the robot for about 2 seconds to set it in the start state. Hit **Dice** on the phone, and the figure will roll among 1-6 and stop randomly.



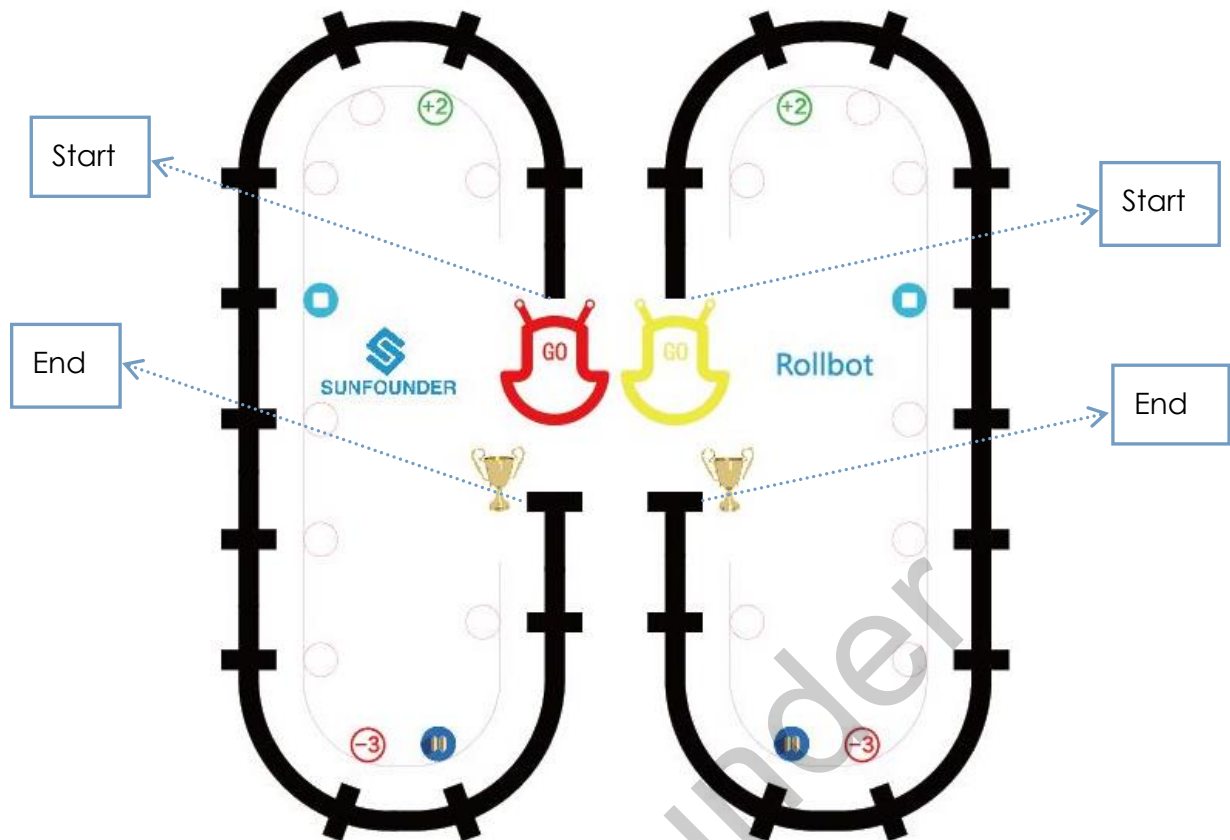
## CONTROL BY APP






The robot will go for the corresponding steps after receiving the data from the APP via the Bluetooth module on the robot. The steps are counted by the points on the map. You can change the rules of the game by pasting the provided round cards (+2, -3, Restart, and Pause; or even self-making other ones) onto circles beside the points on the map and modify the related code.

This game is suitable for two people. Go play with your friends!





The specific rules are like below (for the map above, so as the default program):

- 1) 12 points (steps) in total.
- 2) Paste the  card as a bonus at the second point, so the robot goes forward for another 2 steps without rolling the dice. Set the value of `Forward` as **2** in the program.
- 3) Paste the restart  card to the fifth point, and the robot will return back to the start and turn around again to get ready for the next start. The value of `Restart` is **5**.
- 4) Paste the step  card to the ninth point, the robot goes backward for 3 steps and turns around. The value of `Getback` is **9**.

### Tips:

- 1) The rules above apply only to the code we provide. But you can change them and adapt the card position on the map. **Remember to modify the parameters for each card in the program and set the cards at the corresponding point on the map.**
- 2) The robot should stop exactly at the twelfth point to end the game, otherwise it will go backward based on the number you roll in the app.
- 3) You'd better not paste the card for turning around (like the resume and -3) at a turning on the map for good performance.

The parameters like `Forward`, `Restart`, and `Getback` are for the corresponding +2, restart, and -3 movements in the program. You can change their values for your own rules.

```

RollbotGame$ BarrierCalculate Linefollowing Turn_back
16 RollbotMotors Motors;
17 RollbotLED LED;
18 RollbotOLED OLED;
19 RollbotReadSensor Sensor;
20 RollbotButton Buttons(10);
21 RollbotBuzzer Music(9);
22
23 #define LEFT_LED 3
24 #define RIGHT_LED 2
25
26 #define Forward 2
27 #define Restart 5
28 #define Getback 9
29
30 int Speed_Dir = 0;
31

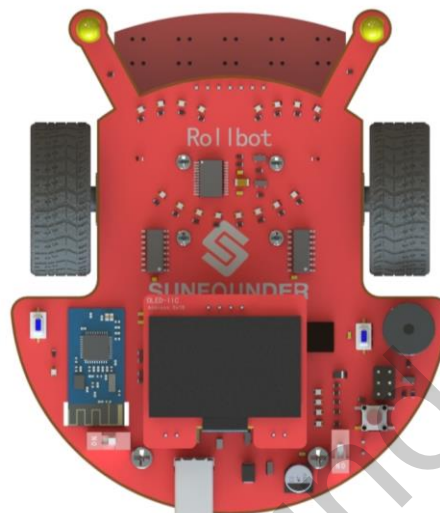
```

We also offer the program of maze but here no more explanation would be given about it. You can draw a maze map according to the cue card.

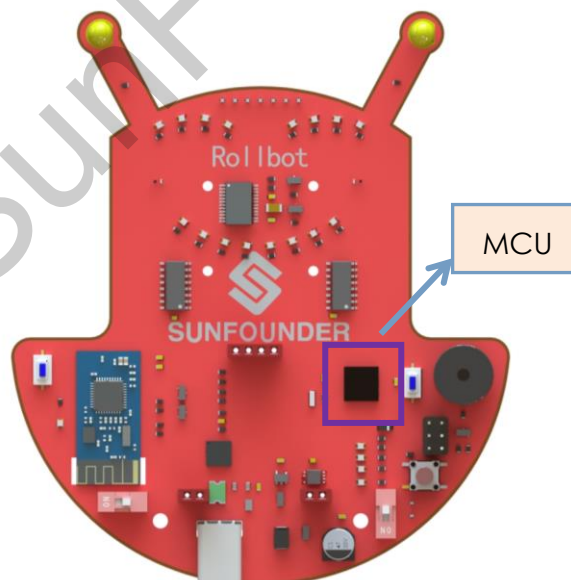
# Hardware Introduction

## 6.1 Overview

The schematic diagram is mainly shown for several modules: main control, boost/step-up convertor, power selector, USB interface, motor driver, buzzer, LED indicator, signal collection, OLED display, and Bluetooth.



## 6.2 Main Control Module



**Pin A0, A1, A2, A3, & A7:** Connect the infrared sensor module. Apply PID algorithm to achieve the line following function of the robot by distinguishing the black and white surfaces.

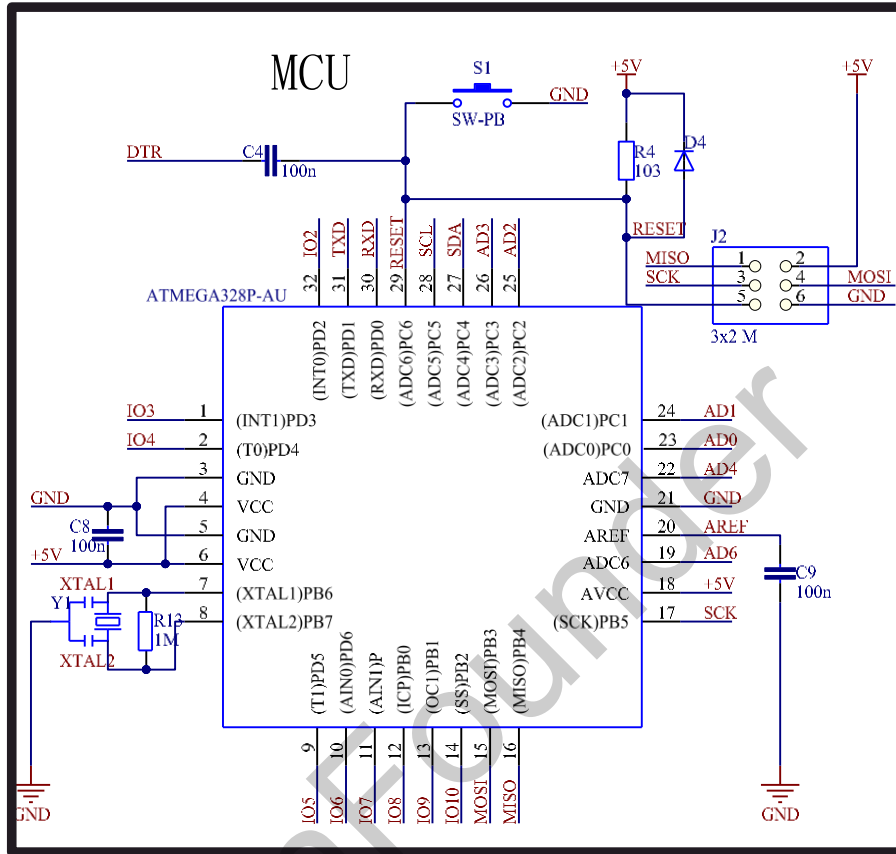
**Pin SCL & SDA:** Simulates the intensity of black lines by the sensor and display it on the OLED screen, thus enabling debugging with the OLED screen.

**Pin D4, D7, D5, & D6:** Pin D4 and D5 for the control of the left motor's direction and speed, when Pin D7 and D6 for the motor on the right.

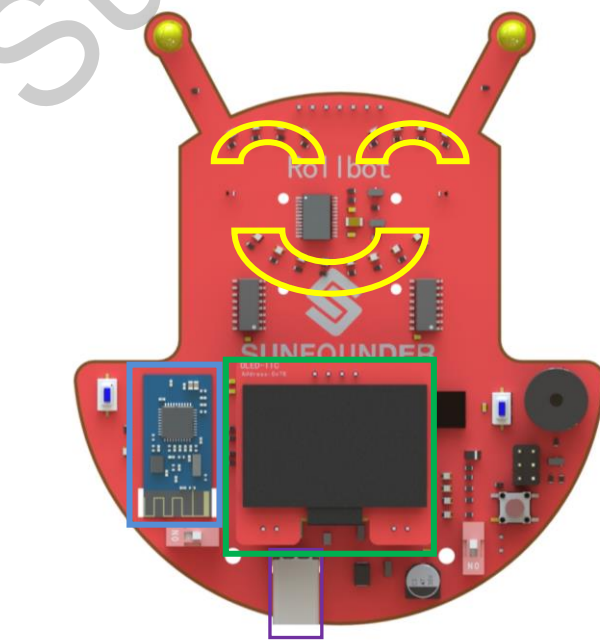
**Pin A6:** Read its AD value so as to tell the battery power in a real-time manner.

**Pin D9:** Output certain frequencies to control the buzzer beeping to play some music.

**Pin D10:** Read the high and low level of the pin to judge whether the button has been pressed or not, so as to control the start of the robot.

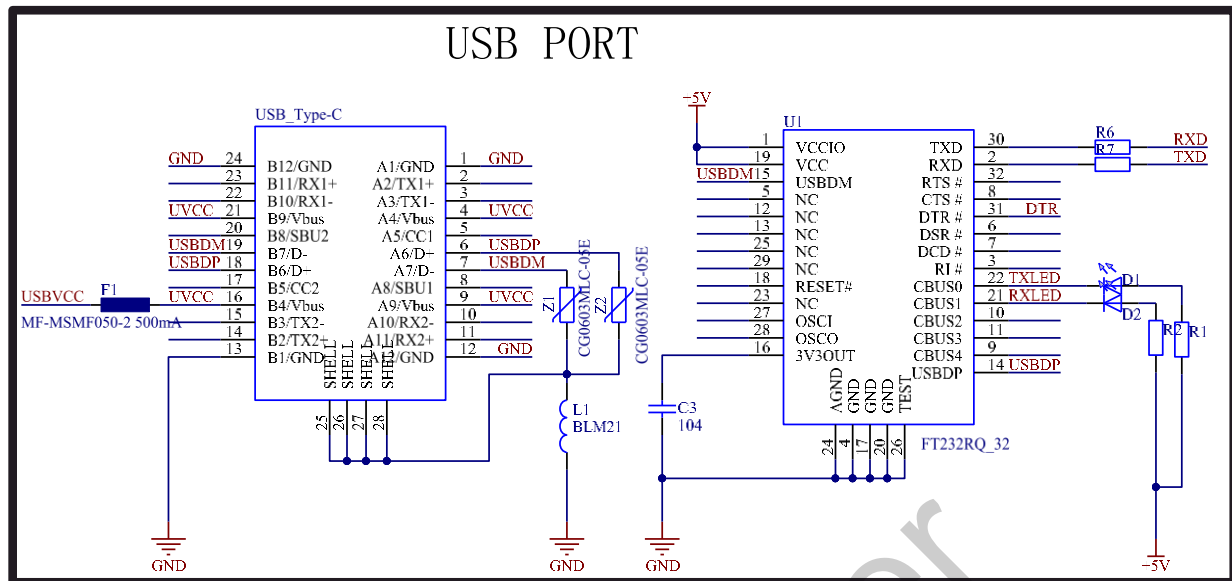


## Other Controls

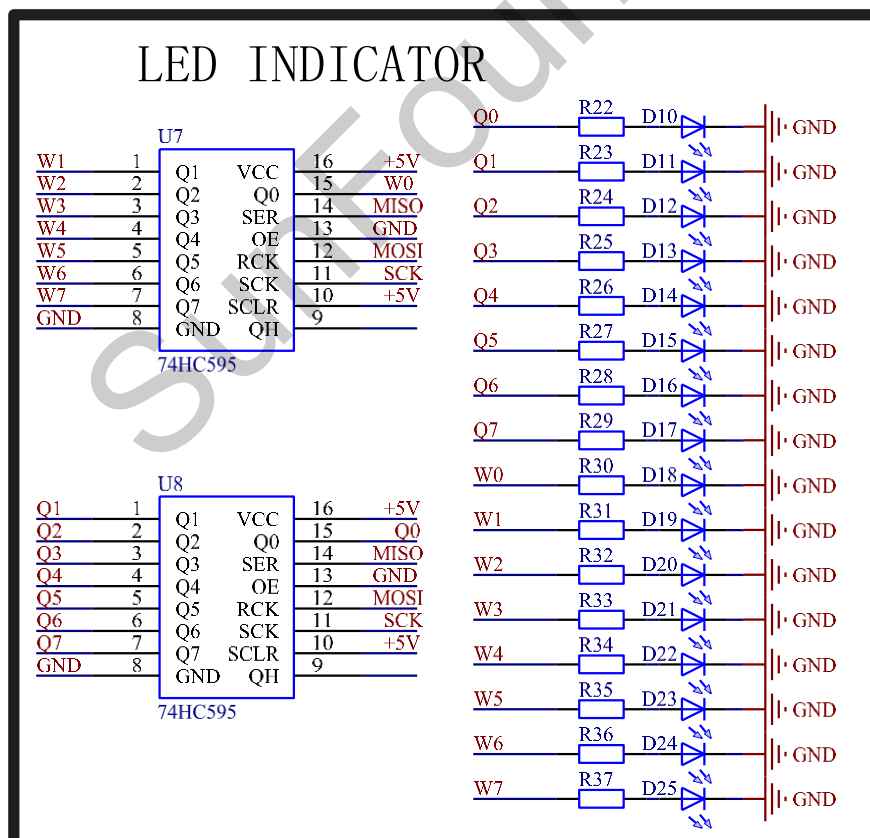


These are the major control of the Rollbot. Please check the circuit diagrams below by module names in different colors.

## 6.3 USB Interface Module



## 6.4 LED Indicator Module



38

# Program Explanation

Through the explanation previously, you should already have a general understanding of the robot. Next we will come to the most important part - theory and program. Most of the programs provided are compiled as libraries and you just need to call them in the Arduino IDE. So in the following let's focus on how to compile them.

## 7.1 Singing with Buzzer

The function of singing with buzzer is to drive the active buzzer to beep by compiling the tunes and the notes so as to play the music. So, what are the procedures?

First, let's come to some simple musical knowledge. We can compile the code after knowing how the music is played.

**1. Play a single note.** A piece of music is composed of several notes. A note corresponds to a frequency. We all know that as long as the Arduino outputs a frequency to the buzzer, there will be a corresponding beep. Here is a table for the mapping between note and tune:

Tune \ Note	1	2	3	4	5	6	7
A	221	248	278	294	330	371	416
B	248	278	294	330	371	416	467
C	131	147	165	175	196	221	248
D	147	165	175	196	221	248	278
E	165	175	196	221	248	278	312
F	175	196	221	234	262	294	330
G	196	221	234	262	294	330	371

Tones \ Notes	1	2	3	4	5	6	7
A	441	495	556	589	661	742	833
B	495	556	624	661	742	833	935
C	262	294	330	350	393	441	495
D	294	330	350	393	441	495	556
E	33	350	393	441	495	556	624
F	350	393	441	495	556	624	661
G	393	441	495	556	624	661	742

Tones \ Notes	1	2	3	4	5	6	7
A	882	990	1112	1178	1322	1484	1665
B	990	1112	1178	1322	1484	1665	1869
C	525	589	661	700	786	882	990

<b>D</b>	589	661	700	786	882	990	1112
<b>E</b>	661	700	786	882	990	1112	1248
<b>F</b>	700	786	882	935	1049	1178	1322
<b>G</b>	786	882	990	1049	1178	1322	1484

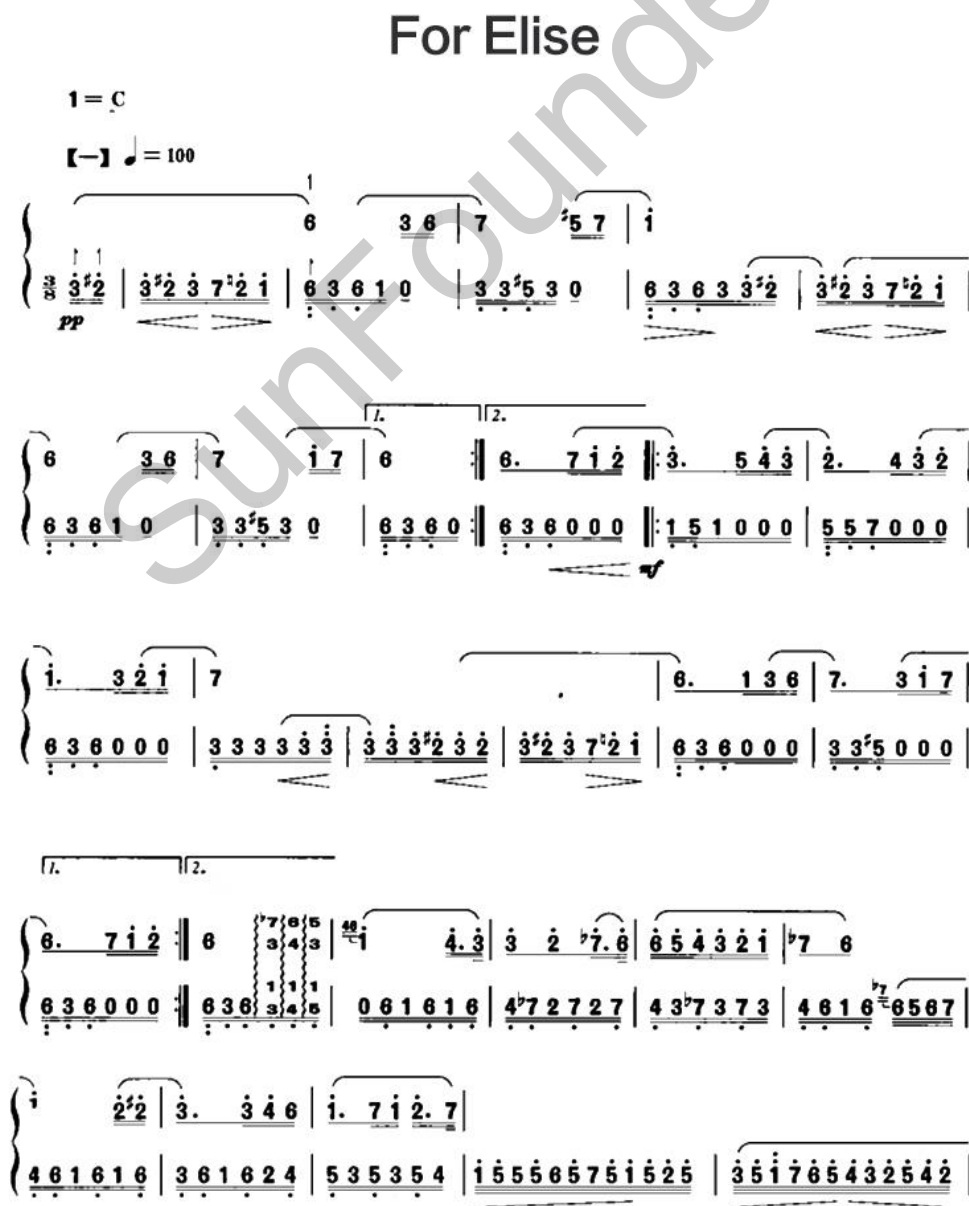
2. Adjust the time span of a note playing. After knowing how the music is being played, next we are going to control how long a single note lasts - each note should play for a certain time and a beautiful piece of music can be thus formed, otherwise just a note after another would be boring enough. How to determine the unit time for each note to be played?

As we all know, the rhythm can be one beat, half beat,  $\frac{1}{4}$  beat and  $\frac{1}{8}$  beat. We set the time of one beat as 1 in code, half beat as 0.5,  $\frac{1}{4}$  as 0.25, and  $\frac{1}{8}$  as 0.125. So we can give each note one such beat and then the music is generated.

Let's see how to translate the numbered musical notation to the corresponding frequency and beat. Just take **For Elise** as an example:

## For Elise

1 = C  
 [—] ♩ = 100



Let's check the details in the music score. Look at **1=C** on the top left corner in the figure above. It means it uses the key of C. Check the C line (in red and bolded) in the frequency table previously. In the first note, there is a dot above 3, so it corresponds to **661** and its corresponding variable is **NOTE\_CH3**. The time is half beat, which is **0.5** in the array **duration[]**. As for the second note, there is another dot above 2 and it is **589**. The time is half beat equaling to **0.5**. As for the definition of scale, we've already defined it in the underlying library so you can use it directly. **NOTE\_A1** means **DO** in the key of **A**, **Note\_A2** for **RE**, **NOTE\_AH1** for the treble **DO**, and **NOTE\_AD1** for the bass **DO** in key A.....

Note that there should be no sound for **0** in the score. And the beat should be **1**.

Every "-" added behind a note, it means +1 of the beat (1+1).

The dot behind a note means a half beat - for example, the beat of 3• is **1+0.5**.

**An easy way for rhythm:** If one single note is not underlined, it means one beat (1); one underline = half beat (0.5); two underlines = 1/4 beat (0.25); one "—" (dash) = duplicate beat of the former note (+1).

**For pitch:** Find the value of the note in the table previously based on the pitch (dot at the top or bottom of the number).

Now let's check the code for the **For Elise** in detail. The following two arrays are the tune (**tune[]**) and frequency (**[duration]**) of each note.

```
int tune[] =
{
    NOTE_CH3,NOTE_CH2,NOTE_CH3,NOTE_CH2,NOTE_CH3,NOTE_C7,NOTE_CH2,NOTE_CH1,NOTE_C6,
    NOTE_C1,NOTE_C3,NOTE_C6,NOTE_C7,
    NOTE_C3,NOTE_C5,NOTE_C7,NOTE_CH1,
    NOTE_C3,NOTE_CH3,NOTE_CH2,NOTE_CH3,NOTE_CH2,NOTE_CH3,NOTE_C7,NOTE_CH2,NOTE_CH1,NOTE_C6,
    NOTE_C1,NOTE_C3,NOTE_C6,NOTE_C7,
    NOTE_C3,NOTE_CH1,NOTE_C7,NOTE_C6,
};
float duration[] =
{
    0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,1,
    0.5,0.5,0.5,1,
    0.5,0.5,0.5,1,
    0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,1,
    0.5,0.5,0.5,1,
    0.5,0.5,0.5,1,
};
int tune[] =
{
    NOTE_CH3,NOTE_CH2,NOTE_CH3,NOTE_CH2,NOTE_CH3,NOTE_C7,NOTE_CH2,NOTE_CH1,NOTE_C6,
    NOTE_C1,NOTE_C3,NOTE_C6,NOTE_C7,
    NOTE_C3,NOTE_C5,NOTE_C7,NOTE_CH1,
    NOTE_C3,NOTE_CH3,NOTE_CH2,NOTE_CH3,NOTE_CH2,NOTE_CH3,NOTE_C7,NOTE_CH2,NOTE_CH1,NOTE_C6,
    NOTE_C1,NOTE_C3,NOTE_C6,NOTE_C7,
```

```
NOTE_C3,NOTE_CH1,NOTE_C7,NOTE_C6,
};
float duration[]=
{
    0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,1,
    0.5,0.5,0.5,1,
    0.5,0.5,0.5,1,
    0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,1,
    0.5,0.5,0.5,1,
    0.5,0.5,0.5,1,
};
```

What's more, the most important step is how to write the compiled tunes and voice frequencies into the buzzer and make it beep accordingly by high and low level and delay.

```
RollbotBuzzer::RollbotBuzzer(int pin)
{
    pinMode(pin,OUTPUT);//Set the pin to buzzer as output
    tonePin=pin;
}
void RollbotBuzzer::MiniBuzzer(int *tune,float *duration,int Length)
{
    for(int x=0;x<Length;x++)//the number of notation cycling
    {
        tone(tonePin,tune[x]);//The function plays the arrayys (each notation) in the tune[] list
in turn
        delay(400*duration[x]);//duration is the time each notation lasts.
//400 means the longer time, the slower tempo, and vice versa
        noTone(tonePin);//Exit the current notation and move on to the next
    }
}
```

## 7.2 Smiling Face with LEDs

The smiling face of the robot is displayed with 16 yellow LEDs. They are controlled directly by the 74HC595 chip. In the code, change the parameter in the output function Dataout() to 0xff, so the LEDs forming a smiling face will light up. Change it to 0x00, and the LEDs will go out.

```
*****Output of data control*****
void RollbotLED::DataOut(int val)
{
    for (int i = 0;i < 8;i++)
    {
        if(val&0x80)
            digitalWrite(dataPin,1);
        else
            digitalWrite(dataPin,0);
        val<<=1;
        digitalWrite(clockPin, HIGH);
        delayMicroseconds(10);
        digitalWrite(clockPin, LOW);
    }
}
```

```
digitalWrite(latchPin,LOW);
delayMicroseconds(10);
digitalWrite(latchPin,HIGH);
}
```

### 7.3 Sensor Signal Intensity on OLED Screen

The OLED screen can display characters, letters, and patterns. It is used in the robot for your better interaction with the robot. How to use it? Here IIC is used in hardware. First, initialize the OLED screen. For use, two functions are indispensable: `WriteCommand()` and `WriteData()`.

```
/******OLED command writing******/
void RollbotOLED::WriteCommand(unsigned int ins)
{
    Wire.beginTransmission(0x78 >> 1); //0x78 >> 1
    Wire.write(0x00); //0x00
    Wire.write(ins);
    Wire.endTransmission();
}
/******OLED data writing******/
void RollbotOLED::WriteData(unsigned int dat)
{
    Wire.beginTransmission(0x78 >> 1); //0x78 >> 1
    Wire.write(0x40); //0x40
    Wire.write(dat);
    Wire.endTransmission();
}
```

Next, the function for positions on the OLED screen: `SetPos()`. With this function, you can locate the specific point of the OLED display.

```
void RollbotOLED::IIC_SetPos(unsigned int x, unsigned int y)
{
    WriteCommand(0xb0 + y);
    WriteCommand(((x & 0xf0) >> 4) | 0x10); //0x10
    WriteCommand((x & 0x0f) | 0x00); //0x01
}
```

Apart from the three basic functions, you can use more for display on the OLED screen. The current code is adequate for displaying sensor signal intensity easily. For more information, you can refer to the program `RollbotOLED.CPP` in the library.

### 7.4 The Bluetooth Control Program

The Bluetooth control is to control the Rollbot with the app on the mobile phone. You can control the robot to go forward and backward, and turn left or right, as well as go for related steps in the dice game.

First, about the motor-drive. The variables `MOTOR_L_DIR`, `MOTOR_L_PWM`, `MOTOR_R_DIR`, and `MOTOR_R_PWM` are to control the direction and speed of the motor rotation. You can change their value for each movement of the Rollbot - the specific parameter for speed, and positive and negative value for rotation direction. The variable `speed_dir` represents the initial status of the

motor, which is aforementioned in **5.3**, **5.4**, and **5.5**.

```

/*****Motor drive function*****/
void RollbotMotors:: MotorSetSpeed(int speed_dir, int speed_left, int speed_right)
{
    switch(speed_dir)
    {
        //speed_dir = 0 means both motors work well and spin forward
        case 0:
        {
            if (speed_left > 0)
            {
                digitalWrite(MOTOR_L_DIR, LOW);
                analogWrite(MOTOR_L_PWM, speed_left);
            }
            else
            {
                digitalWrite(MOTOR_L_DIR, HIGH);
                analogWrite(MOTOR_L_PWM, (-1)*speed_left);
            }
            if (speed_right > 0)
            {
                digitalWrite(MOTOR_R_DIR, HIGH);
                analogWrite(MOTOR_R_PWM, speed_right);
            }
            else
            {
                digitalWrite(MOTOR_R_DIR, LOW);
                analogWrite(MOTOR_R_PWM, (-1)*speed_right);
            }
        }break;
        //speed_dir = 1 means both motors' rotation are reversed and spin backward
        case 1:
        {
            if (speed_left > 0)
            {
                digitalWrite(MOTOR_L_DIR, HIGH);
                analogWrite(MOTOR_L_PWM, speed_left);
            }
            else
            {
                digitalWrite(MOTOR_L_DIR, LOW);
                analogWrite(MOTOR_L_PWM, (-1)*speed_left);
            }
            if (speed_right > 0)
            {
                digitalWrite(MOTOR_R_DIR, LOW);
                analogWrite(MOTOR_R_PWM, speed_right);
            }
            else

```

```

    {
        digitalWrite(MOTOR_R_DIR, HIGH);
        analogWrite(MOTOR_R_PWM, (-1)*speed_right);
    }
}break;
//speed_dir = 2 means the left motor works well and the right one spins reversely
case 2:
{
    if (speed_left > 0)
    {
        digitalWrite(MOTOR_L_DIR, LOW);
        analogWrite(MOTOR_L_PWM, speed_left);
    }
    else
    {
        digitalWrite(MOTOR_L_DIR, HIGH);
        analogWrite(MOTOR_L_PWM, (-1)*speed_left);
    }
    if (speed_right > 0)
    {
        digitalWrite(MOTOR_R_DIR, LOW);
        analogWrite(MOTOR_R_PWM, speed_right);
    }
    else
    {
        digitalWrite(MOTOR_R_DIR, HIGH);
        analogWrite(MOTOR_R_PWM, (-1)*speed_right);
    }
}break;
//speed_dir = 3 means the right motor works well and the left one spins reversely
case 3:
{
    if (speed_left > 0)
    {
        digitalWrite(MOTOR_L_DIR, HIGH);
        analogWrite(MOTOR_L_PWM, speed_left);
    }
    else
    {
        digitalWrite(MOTOR_L_DIR, LOW);
        analogWrite(MOTOR_L_PWM, (-1)*speed_left);
    }
    if (speed_right > 0)
    {
        digitalWrite(MOTOR_R_DIR, HIGH);
        analogWrite(MOTOR_R_PWM, speed_right);
    }
    else
    {

```

```

        digitalWrite(MOTOR_R_DIR, LOW);
        analogWrite(MOTOR_R_PWM, (-1)*speed_right);
    }
    }break;
}
}

```

Next, how to read and save the data collected and transferred via Bluetooth:

```

while (Serial.available())
{
    ReceiveByte = Serial.read();
}

```

Last, judge the data and thus control the robot to act accordingly. Thus, you can control the robot by the app on your mobile.

## 7.5 Line Following

The line following function is the most important function of the robot. It distinguishes the black and white surfaces through the infrared sensor module and applies the PD algorithm to realize line following. It collects and saves the AD value of the 5 pins: A0, A1, A2, A3, and A7.

```

/*****Read data of the sensors*****/
void RollbotReadSensor::Read_Data()
{
    data[0] = analogRead(A0);
    data[1] = analogRead(A1);
    data[2] = analogRead(A2);
    data[3] = analogRead(A3);
    data[4] = analogRead(A7);
    OLED_Flag[0] = map(data[0], 50, 500, 3, 6);
    OLED_Flag[1] = map(data[1], 50, 500, 3, 6);
    OLED_Flag[2] = map(data[2], 50, 500, 3, 6);
    OLED_Flag[3] = map(data[3], 50, 500, 3, 6);
    OLED_Flag[4] = map(data[4], 50, 500, 3, 6);
}

```

Compare the collected AD value with the given value and generate a result which indicates the position of the robot to the black line, as shown below.

```

int RollbotReadSensor::Read_BlackFlag()
{
    while(1)
    {
        Read_Data();
        if ((data[0] < threshold) && (data[1] > threshold) && (data[2] > threshold) && (data[3] > threshold) && (data[4] > threshold)) return -4;
        else if ((data[0] < threshold) && (data[1] < threshold) && (data[2] > threshold) && (data[3] > threshold) && (data[4] > threshold)) return -3;
        else if ((data[0] > threshold) && (data[1] < threshold) && (data[2] > threshold) && (data[3] > threshold) && (data[4] > threshold)) return -2;
    }
}

```

```

    else if ((data[0] > threshold) && (data[1] < threshold) && (data[2] < threshold) && (data[3] >
threshold) && (data[4] > threshold)) return -1;
    else if ((data[0] > threshold) && (data[1] > threshold) && (data[2] < threshold) && (data[3] >
threshold) && (data[4] > threshold)) return 0;
    else if ((data[0] > threshold) && (data[1] > threshold) && (data[2] < threshold) && (data[3]
< threshold) && (data[4] > threshold)) return 1;
    else if ((data[0] > threshold) && (data[1] > threshold) && (data[2] > threshold) && (data[3]
< threshold) && (data[4] > threshold)) return 2;
    else if ((data[0] > threshold) && (data[1] > threshold) && (data[2] > threshold) && (data[3]
< threshold) && (data[4] < threshold)) return 3;
    else if ((data[0] > threshold) && (data[1] > threshold) && (data[2] > threshold) && (data[3] >
threshold) && (data[4] < threshold)) return 4;
    else if ((data[0] < threshold) && (data[1] < threshold) && (data[2] < threshold) && (data[3]
< threshold)) return 5;
    else if ((data[1] < threshold) && (data[2] < threshold) && (data[3] < threshold) && (data[4]
< threshold)) return 5;
    else if ((data[0] > threshold) && (data[1] > threshold) && (data[2] > threshold) && (data[3] >
threshold) && (data[4] > threshold)) return 6;
}
}

```

Finally, we just need to read the position of the robot and adjust the related parameters via the PD formula below, thus realizing the function of following lines.

$$\text{MotorSpeed} = P * \text{SignalValue} + D * (\text{SignalValue} - \text{LastError})$$

SignalValue is the gap between the position of Rollbot detected and that of the black line. LastError is the error value accumulated till the last time. SignalValue - LastError means the rate of change of the relative position.

```

Motorpeed = int(P * SignalValue + D * (SignalValue - LastError));
LastError = 0;
LastError = SignalValue;
Speed_L = BASE_SPEED + Motorpeed;
Speed_R = BASE_SPEED - Motorpeed;
if (Speed_R > MAX_SPEED ) Speed_R = MAX_SPEED;
if (Speed_L > MAX_SPEED ) Speed_L = MAX_SPEED;
if (Speed_R < MIN_SPEED) Speed_R = MIN_SPEED;
if (Speed_L < MIN_SPEED) Speed_L = MIN_SPEED;
Motor.Motordrive(Speed_Dir, Speed_L, Speed_R);

```

## Appendix: FAQ

**Q1.** In the line following, I've uploaded the program to the board and then unplug the cable, but the Rollbot does not move.

**A:** 1) Check whether you've switched on the power of the robot.

2) Check whether you've pressed the start button.

3) Pressing the start button for enough long time.

4) Check whether the IR sensor module is installed well.

**Q2.** The Rollbot runs on the map randomly and not following the black line.

**A:** 1) Check whether you've changed the value of `Speed_Dir` right before uploading the program. Please refer to **5.3 Motor Testing**.

2) If the robot keeps spinning itself, check whether the wire of one motor is loose.

**Q3.** Why can I control the car in the app on my phone?

**A:** 1) Check whether the Bluetooth on your mobile is enabled.

2) You need to tap **Select devices** and choose the device (Bluetooth name of the robot) correctly and hit **Connect**.

3) Check whether the Bluetooth switch on the Rollbot is turned on (see the **Gameplay** chapter).

**Q4.** I drew a map myself but the Rollbot just cannot follow the line on it.

**A:** 1) Check whether the line is wide enough. Please follow the instructions on the guide card.

2) Use black Marker pens for good performance.

## Afterword

Congrats! You've completed all the assembly and debugging of the Rollbot and should be playing with it happily now. The code part may be a little bit difficult to understand and welcome to ask technical questions under FORUM on our website [www.sunfounder.com](http://www.sunfounder.com).

To be honest, our program is just for reference and it may not be so perfect. You can improve the program by yourself if abilities permit and welcome to share with us!

Enjoy your time with the **ROLLBOT**!



## Copyright Notice

All contents including but not limited to texts, images, and code in this manual are owned by the SunFounder Company. You should only use it for personal study, investigation, enjoyment, or other non-commercial or nonprofit purposes, under the related regulations and copyrights laws, without infringing the legal rights of the author and relevant right holders. For any individual or organization that uses these for commercial profit without permission, the Company reserves the right to take legal action.