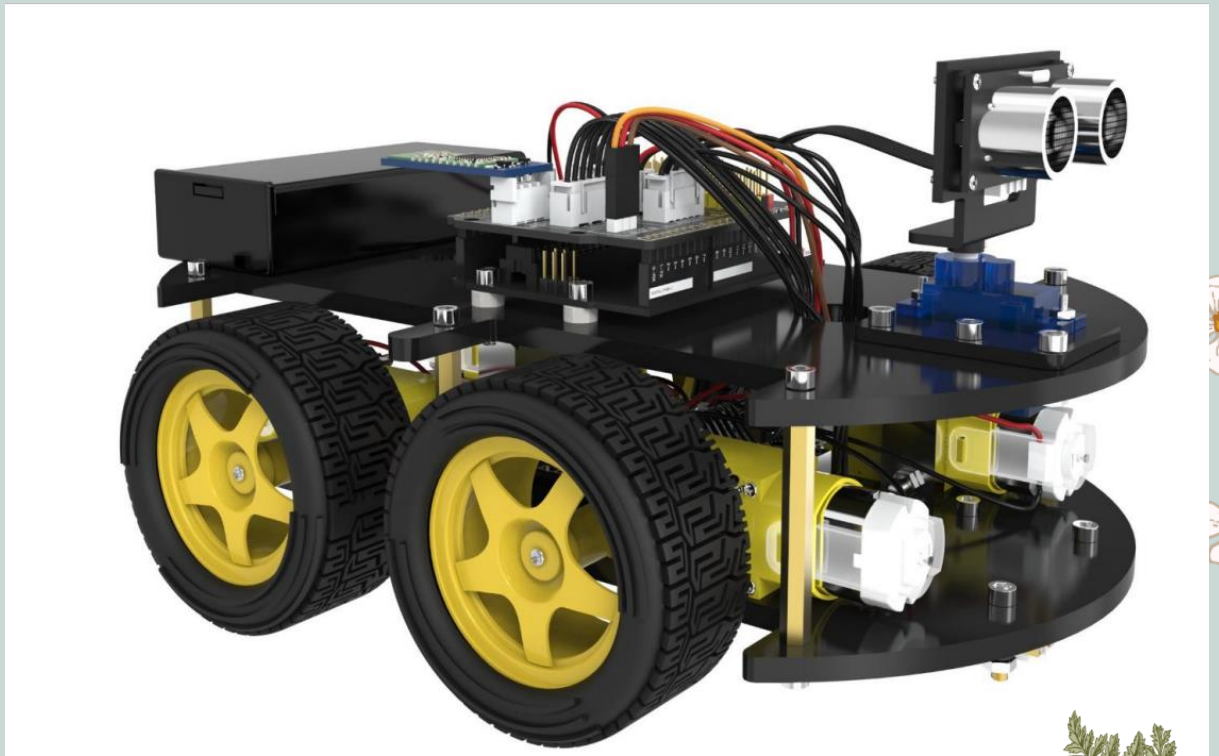
Manual

Smart Robot Car Using Arduino

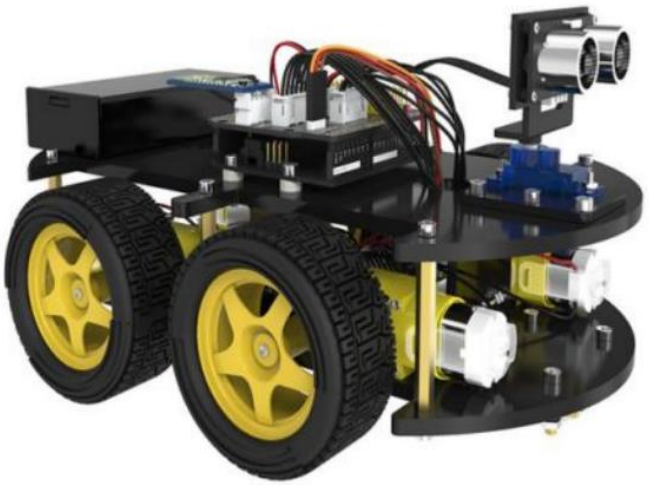


Agenda

- Introduction
- Module Test
- Bluetooth Car
- Infrared Remotecontrol Car
- Obstacle Avoidance Car
- Line Tracking Car
- Smart Car Multi function

Introduction

A smart robot car is a self-controlled automobile that functions to teach kids (12+), amateurs with electronics, and robot enthusiasts the science of programming, parts assembly, remote controlling, and robotics. It makes the world of IoT interactive and provides hands-on experience to kids and adults.



Module_ Test

Module_Test

Module_Test

i. Points of This Section

ii. *Learning Objectives*:

1. *Learn how to use Arduino IDE*
2. *Make the car move by uploading program*

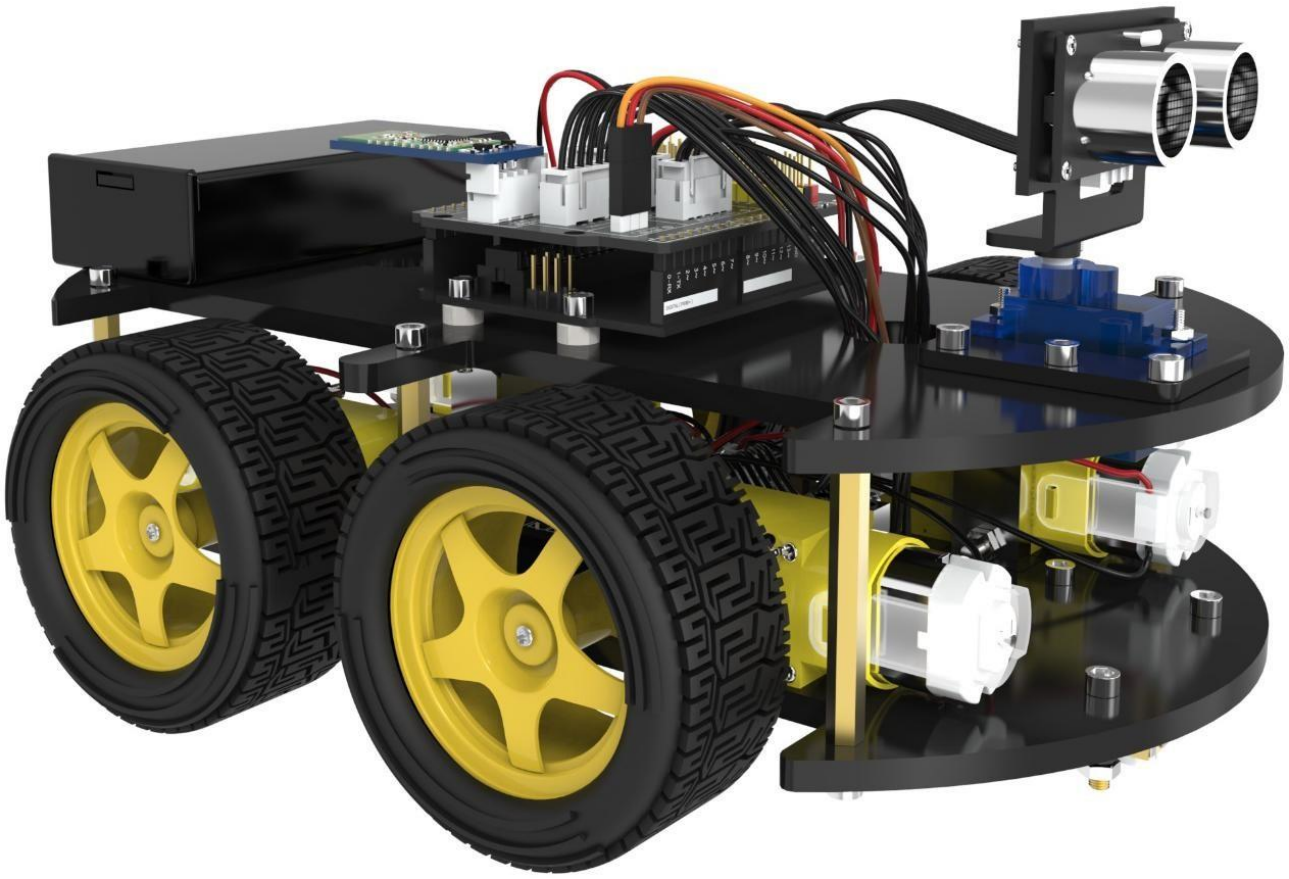
iii. Preparations:

- *One car (with a battery)*
- *One USB cable*

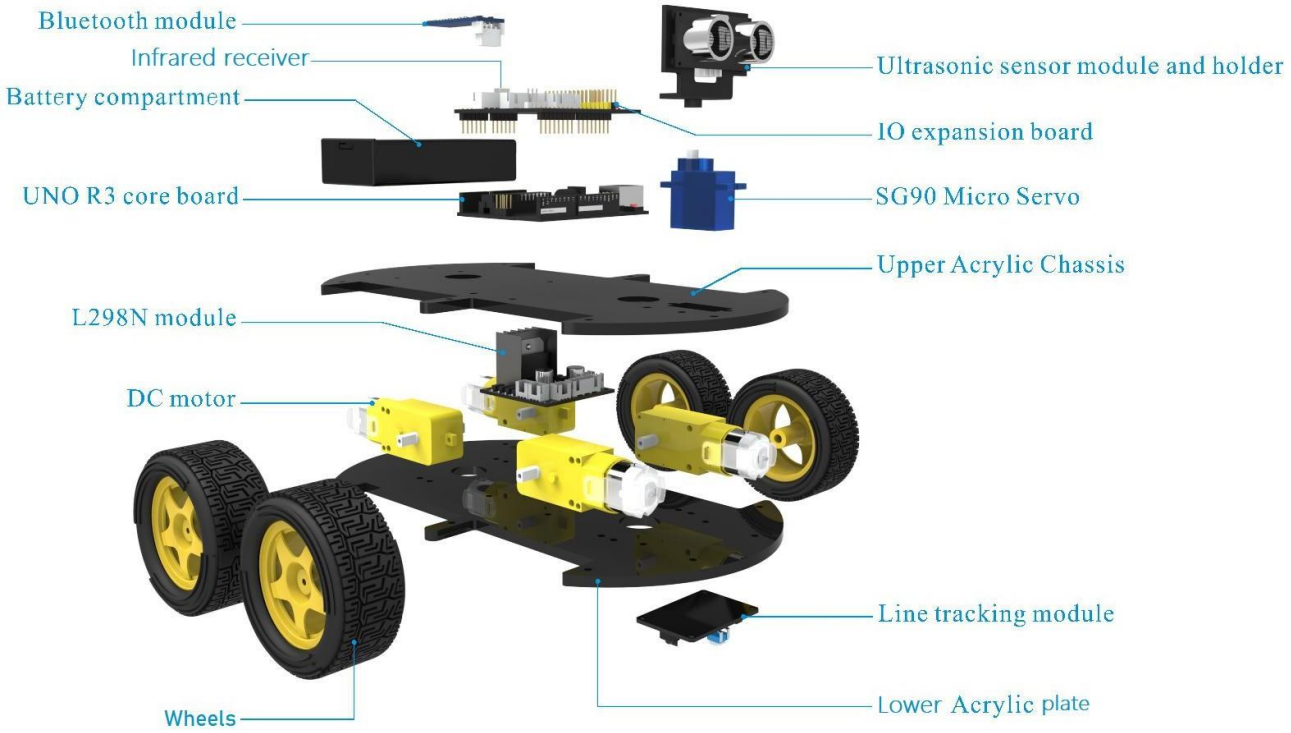
Ⅰ. Introduction of the car

This kit is an extremely flexible vehicular kit particularly designed for education, competition and entertainment purposes. The upper panel of the kit is directly compatible with 9-gram steering engine. It also carries supersonic sensor, battery and other fixed holes to facilitate installation of various sensors. This is a very funny and versatile robot that meets learning and production purposes. With it, you can implement diverse interesting ideas, such as Bluetooth and Infrared remote control, automatic avoidance of obstacles and line inspection.

Let’s introduce the small vehicle that will accompany us for a long time in the future.



Each parts of the car is as below:



Function of each part：

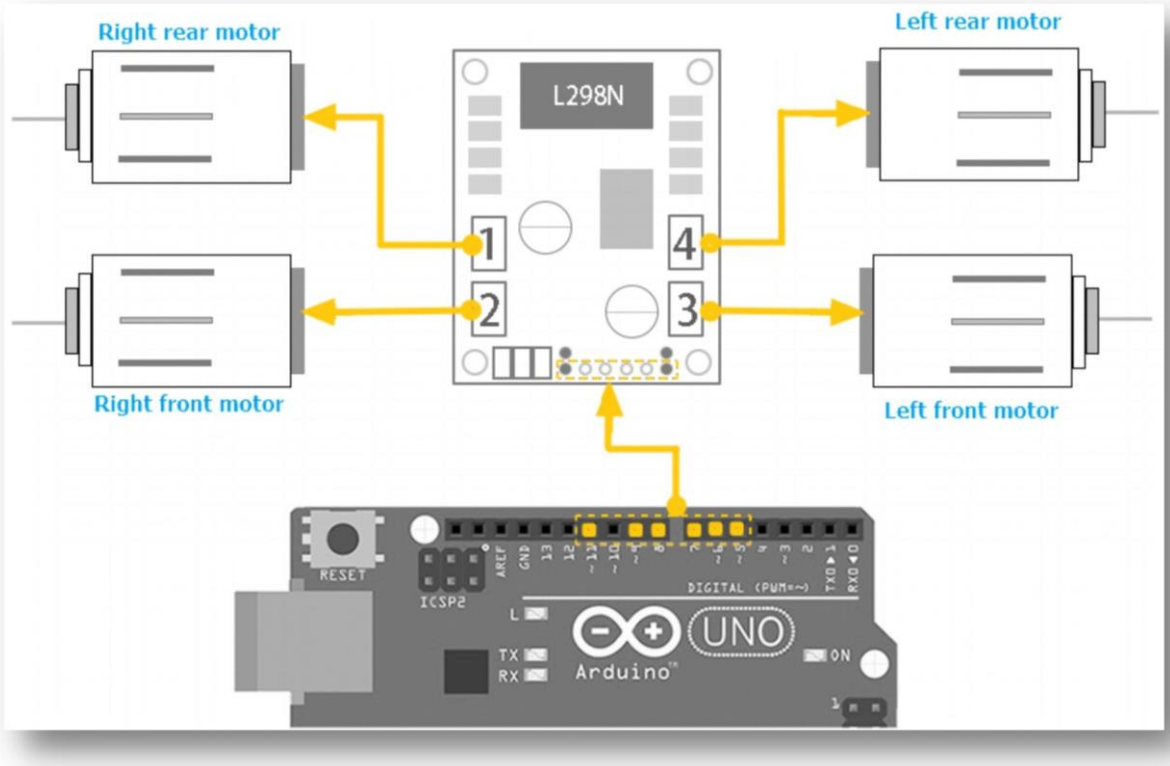
1. Battery compartment (with a switch): supply power for the vehicle
2. DC motor + wheels: drive the vehicle to move
3. Upper and under Acrylic Chassis: the frame of the car
4. L298N motor driving board: drive the motor to rotate
5. UNO R3 core board: the brain of the car, controls all the parts
6. IO expansion board: combined with the UNO, make connection become more easier
7. SG90 Micro Servo: enable the GP2Y0A21 distance sensor to rotate 180 degrees
8. Ultrasonic sensor module and holder: distance measurement and obstacle avoidance
9. Line tracking module: black and white sensor for recognition of the white and black lanes
10. Infrared receiver and remote control: provide the Infrared remote control function
11. Bluetooth module: provide the Bluetooth control function

Ⅱ. Upload program

Because we have uploaded the program to the car in Lesson 0, now, you can turn on the power switch and put the car on the ground. Then you will see the car moving.

Tips: Before turning on the power switch, check whether the battery is fully charged. If the battery is low, charge it in time. In the charging process, the charger shows a red LED indicates that the battery is not fully charged, the charger shows a blue LED indicates that it is fully charged.

Ⅲ. Description of Principles



How to use L298N motor driver board

Definition of the connection ports on L298N board have been marked above. The motors should be connected to the L298N board as shown in the picture above, and if you find the rotational direction of one of the motors is opposite, please change the connecting position of its black and red wires.

L298N GND is connected to battery box GND;

L298N VCC is connected to battery box VCC;

UNO board is also connected to battery box.

L298N 5V here cannot be connected to UNO 5V;

ENA and ENB control the speed of right and left motor separately by PWM.

IN1, IN2, IN3 and IN4: IN1 and IN2 are used to control left motor, IN3 and IN4 are used to control right motor. About the principle, please look at the sheet below: (We take left motor for example)

ENA	IN1	IN2	DC MOTOR STATUS
0	X	X	STOP
1	0	0	BRAKING
1	1	0	FORWARD
1	0	1	BACKWARD
1	1	1	BARKING

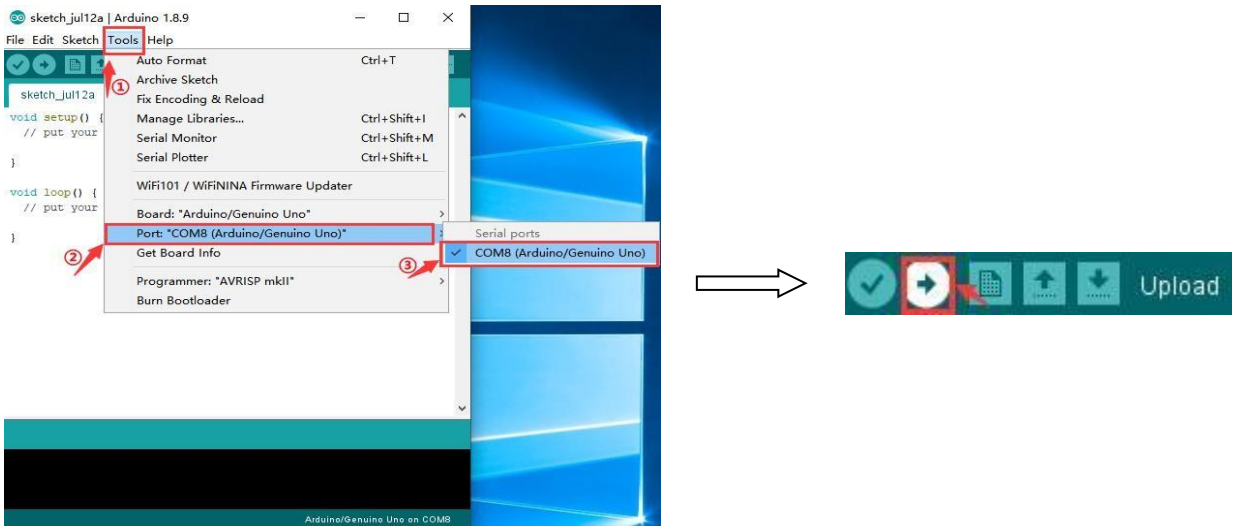
IV. Make the Car Move

TIPS:

When uploading codes, please remove the Bluetooth module from the IO expansion board (Because the serial port for uploading codes and Bluetooth communication is the same one and there will be conflicts.)

You can mount the Bluetooth module after the upload.

Please make sure your com is right before click on upload to upload the program.



The first step: drive the motor

We will try to move the motor without speed controlling. Because it is easier to write program without speed controlling.

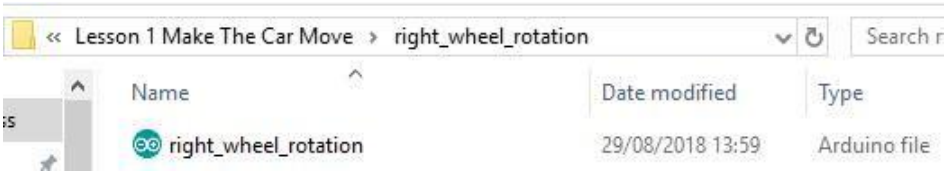
First of all, let's see the connection of the motor and the L298N board, we will use Arduino 5, 6, 7, 8, 9, 11 pins to control the car. 9 and 11 pins control the right wheel. 7 and 8 pins control the left wheel. 5 and 6 pins control ENA and ENB respectively.

So the connection is as below:

L298N	V5 expansion board
ENA	5
ENB	6
IN1	7
IN2	8
IN3	9
IN4	11

Based on the sheet given above, we first design a simple program to make the right wheel turn 0.5s in positive direction, stop 0.5s, turn 0.5s in negative direction and stop 0.5s. And the wheel will repeat the reaction.

Connect the UNO controller board to the computer, open the code file in the path "\Lesson 1 Make The Car Move\right_wheel_rotation\ right_wheel_rotation.ino". Upload the program to the UNO board.



Disconnect it from the computer, and then switch on the car's power supply. You will see that the right wheel moves as you expected.

If the car is not moving, press the reset button on the UNO board.

If the moving direction of the motor is different from the direction you set, you can change the connection of black and red lines from the motor to the L298N board.

Then, we make the left wheel rotate in the same way.

Connect the UNO controller board to the computer, open the code file in the path "Lesson 1 Make The Car Move\Left_wheel_rotation\ Left_wheel_rotation.ino". Upload the program to the UNO board.

Disconnect it from the computer, and then switch on the car's power supply. You will see that the right wheel moves as you expected.

The second step: move forward and backward

After finishing debugging the car, you can write programs to make the car move.

Below is the way how car moves:

CAR	Forward	Back	Stop
Left wheel	Forward	Back	Stop
Right wheel	Forward	Back	Stop

CAR	Turn left	Turn right	Stop
Left wheel	Back	Forward	Stop
Right wheel	Forward	Back	Stop

Next, we will write a simple program to make the car go forward 0.5s, then stop 0.5s, then back up 0.5s and then stop 0.5s.

Connect the UNO controller board to the computer, open the code file in the path "Lesson 1 Make The Car Move\forward_back\forward_back.ino". Upload the program to the UNO board.

Upload the program to the UNO board, disconnect it from the computer, and then switch on the car's power supply. You will see that the right wheel moves as you expected.

The third step: write the program

It may be difficult for you to write the whole program to make the car move automatically. So we separate the movements into different function, for example moving forward and turning left. And when we write the program in the final step, we can call the function.
Next, we begin to write programs for each movement:

The fourth step: move automatically

We start to write program to make the car move automatically: go forward 0.4s - back up 0.4s - turn left 0.4s - turn right 0.4s.

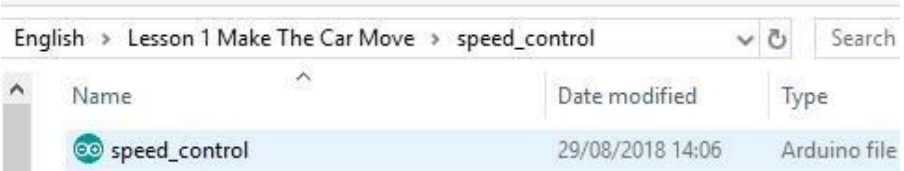
Connect the UNO controller board to the computer, open the code file in the directory "Lesson 1 Make The Car Move\AUTO_GO_\ AUTO_GO_.ino". Upload the program to the UNO board.

Disconnect it from the computer, and then switch on the car's power supply. You will see that the wheel moves as you expected.

The fifth step: speed_control

The code to achieve the function is to control the speed of the car: go forward and reduce the speed → stop 1s → running back and accelerate → stop 2s.

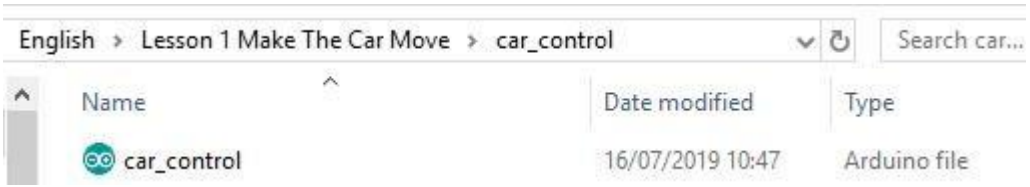
Connect the UNO controller board to the computer, open the code file in the directory "Lesson 1 Make The Car Move\speed_control\ speed_control.ino". Upload the program to the UNO board.



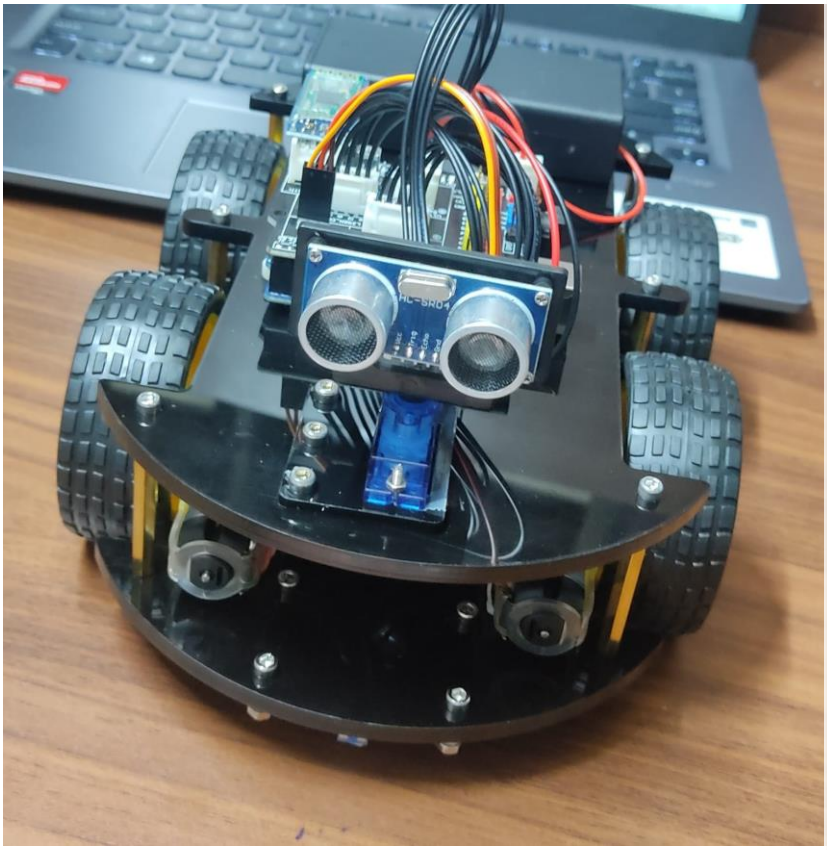
The sixth step: car_control

The code to achieve the function is to control the speed of the car and you can set the speed you like to change the value of **CAR_SPEED.**

Connect the UNO controller board to the computer, open the code file in the directory "Lesson 1 Make The Car Move\speed_control\ car_control.ino". Upload the program to the UNO board.



Bluetooth Car



Bluetooth Car

Points of this section

It is very important and so cool to control your car wirelessly in a certain space when we learn the Arduino, so in the lesson, we will teach you how to control a car by Bluetooth.

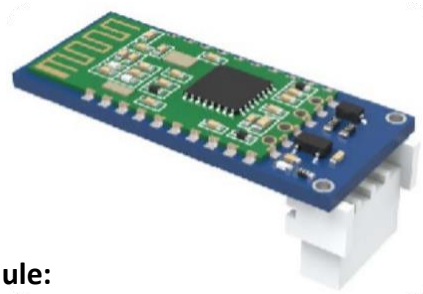
Learning Parts:

- *Learn how to use the Bluetooth module and the Bluetooth APP*
- *Learn how to control the vehicle via Bluetooth*
- *Write programs to implement this function*

Preparations:

- *A vehicle (equipped with battery)*
- *A USB cable*
- *A Bluetooth module*
- *An IPhone or tablet*

Ⅰ. Bluetooth module

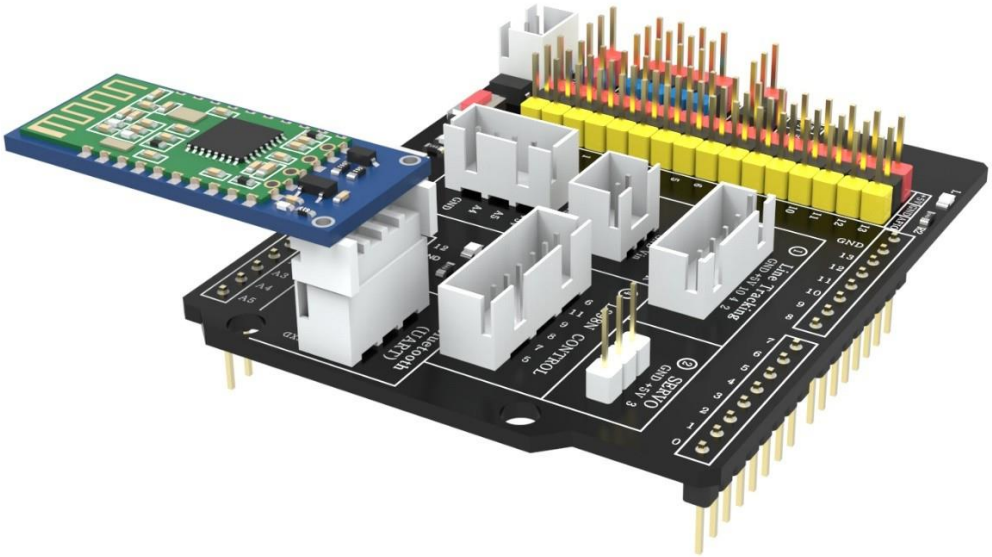


The description of Bluetooth module:

1. Adopt mainstream Bluetooth chip of TI, protocol standard of Bluetooth V4.0.
2. Analog working voltage of serial port is 3.3V.
3. Users can set baud rate 1200, 2400, 4800, 9600, 19200, 38400, 57600,115200.
4. Dimension of key module is 28mm x 15 mm x 2.35mm.
5. Working electric current: 40MA.
6. Dormancy electric current: less than 1MA.
7. Being used for GPS navigation system, hydroelectric gas reading system, industrial field mining control system.
8. Can be connected to Bluetooth laptop, computer with Bluetooth adapter, PDA, etc.

This is the schematic diagram of Bluetooth module connected to UNO controller board:

In the experiment we will connect it to UNO board via expansion board V5.



Ⅱ. Getting Started with the Bluetooth APP

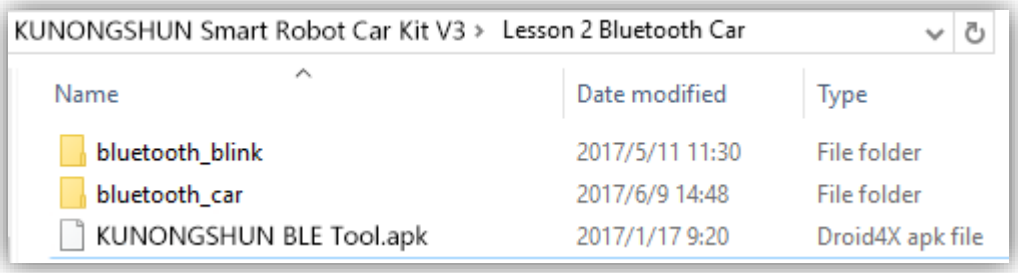
Before beginning, connect the HC-08 Bluetooth module to the expansion board and turn on the power.

STEP1: Install the application.

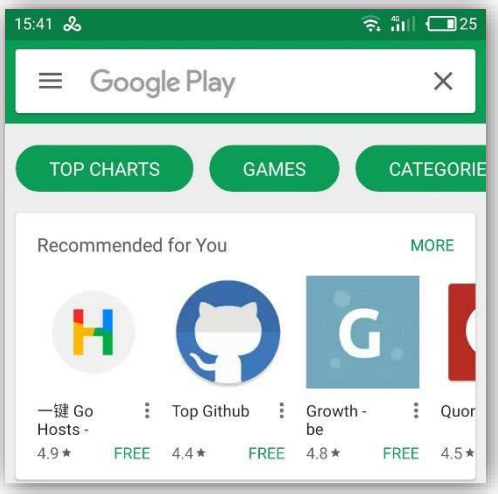
For Android

There are two ways to install the application.

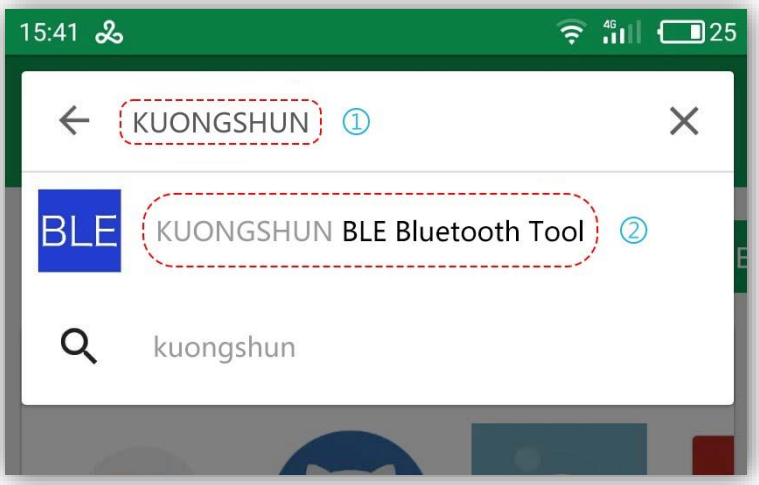
1. Copy the “KUONGSHUN BLE Tool.apk” file to the Android products and install it.



2. Search “KUONGSHUN BLE Bluetooth Tool” in Google Play Store and install it.

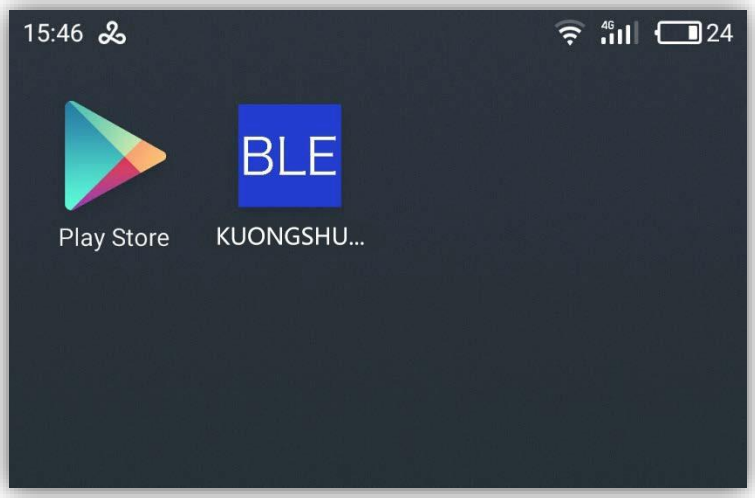


1. Launch Google Play



2. Search “KUONGSHUN BLE Bluetooth Tool”

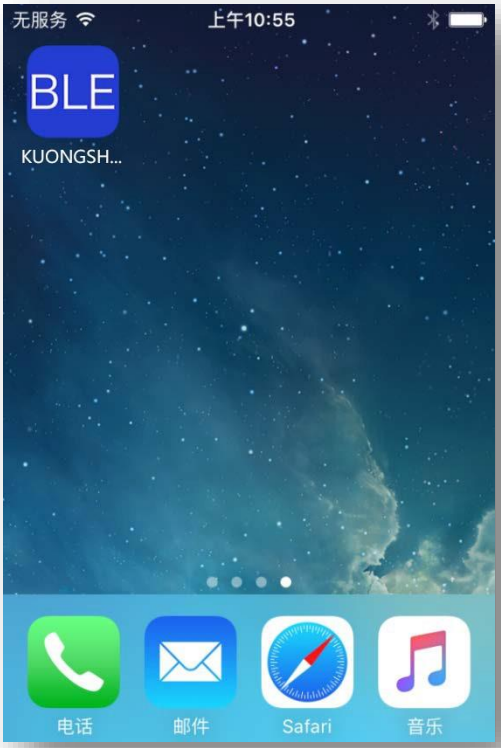
3. INSTSLL



4. Finish the installation

For iOS

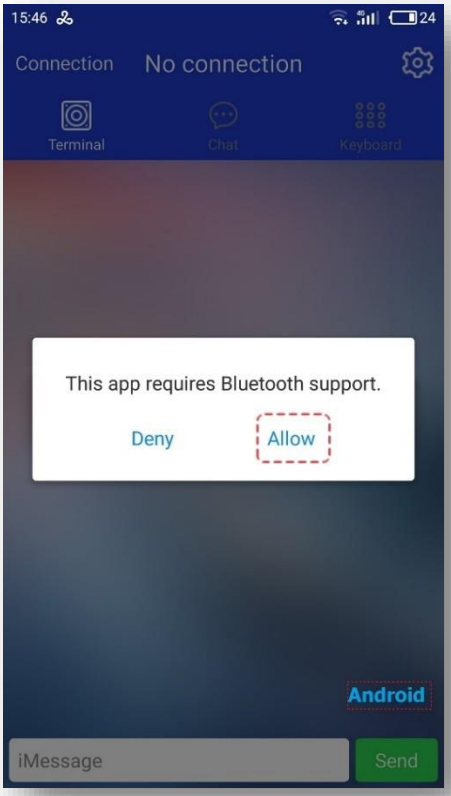
Search “KUONGSHUN BLE Tool” in Apple APP Store and install it.

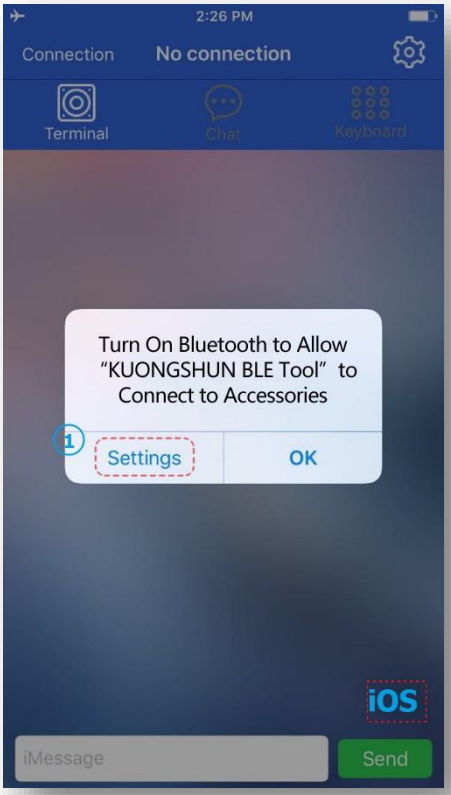


STEP2: Application Settings.

Launching the application.

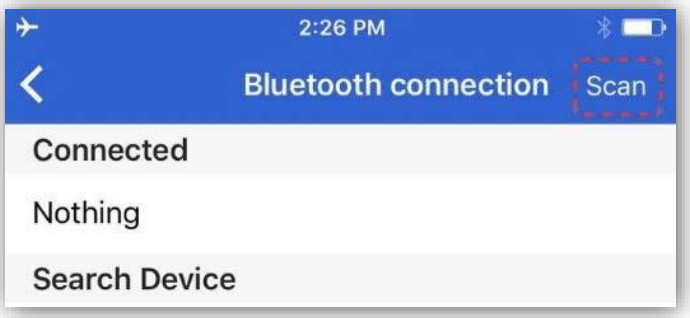
If you don't open your Bluetooth, it will show as blow and advise you to turn on Bluetooth function. For Android, click “Allow” that will open the bluetooth automatically. For iOS, click “Settings” that will jump to bluetooth setup interface, you should turn on the bluetooth and then return to the “KUONGSHUN BLE Tool” APP interface.



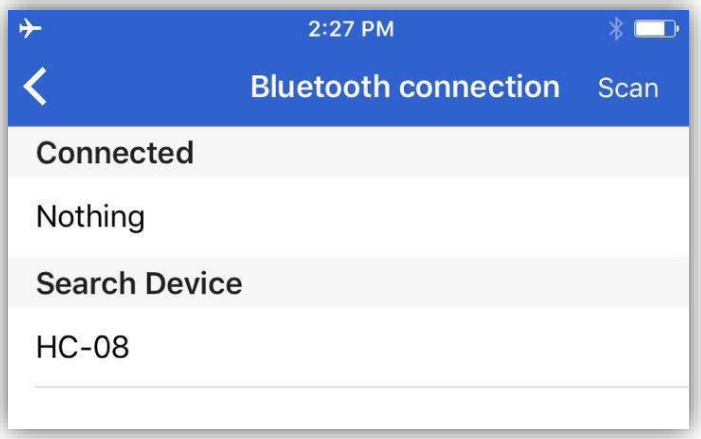


After turn on Bluetooth, the bluetooth icon appears on the APP interface. Click “Connection”.

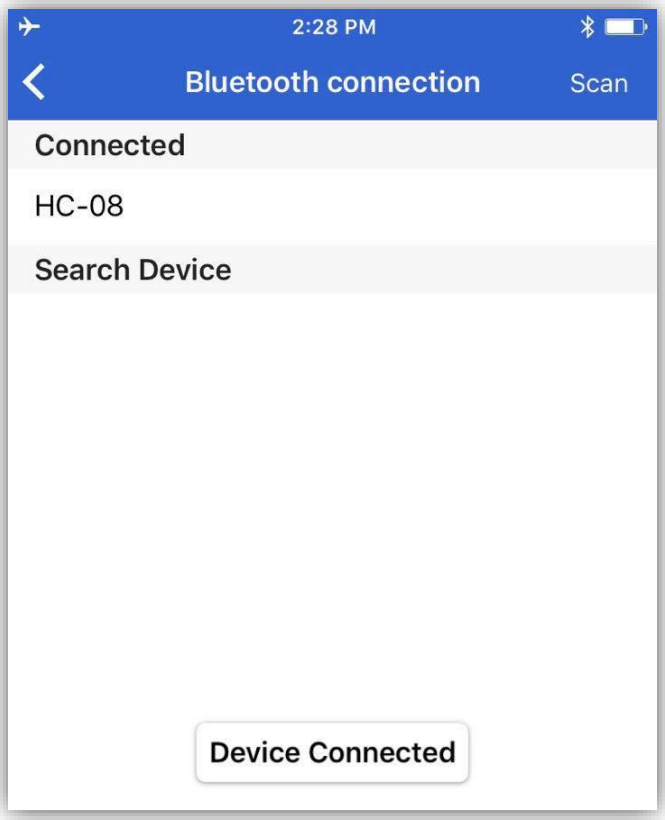
Scan the Bluetooth signal.



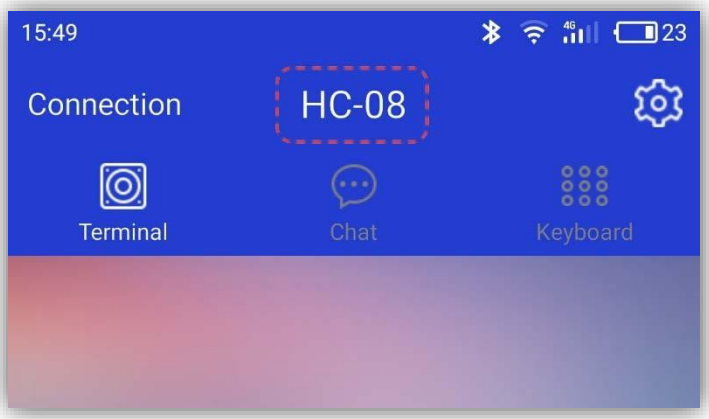
Then you phone will search Bluetooth equipment nearby. HC-08 device appear.



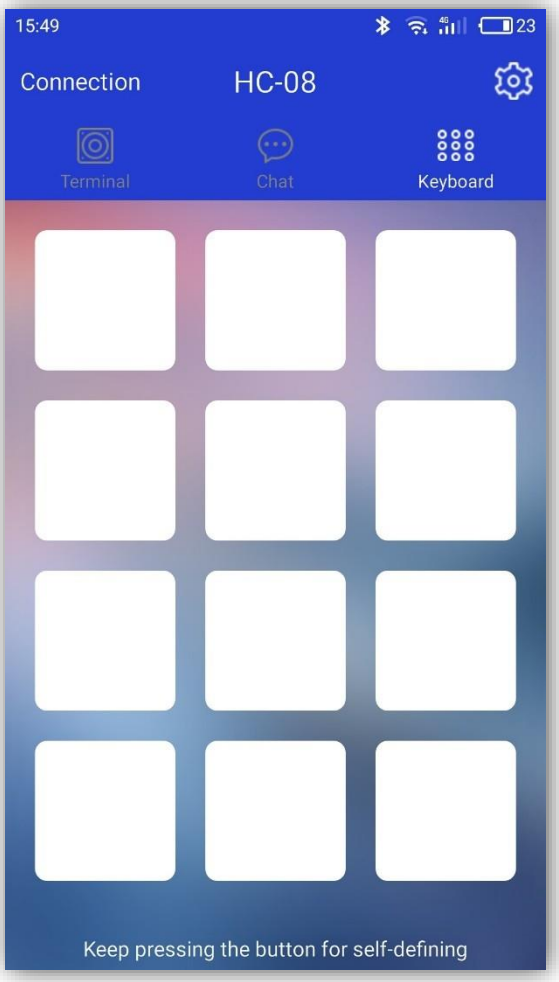
Click the Bluetooth name “HC-08”, when the connection is successful, the screen will be displayed “Device Connected”.



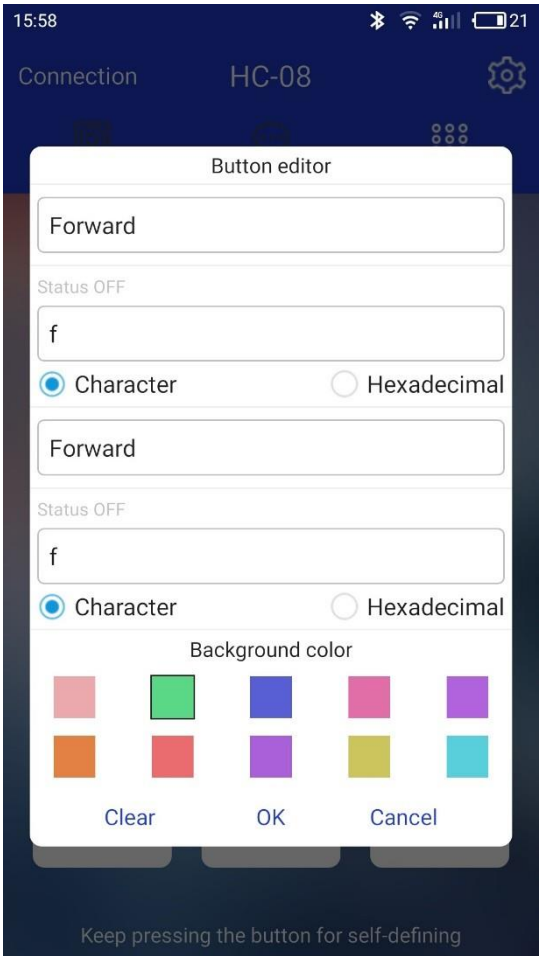
Returns the main interface of the application. After the Bluetooth connection, the Bluetooth name will be displayed on the screen.

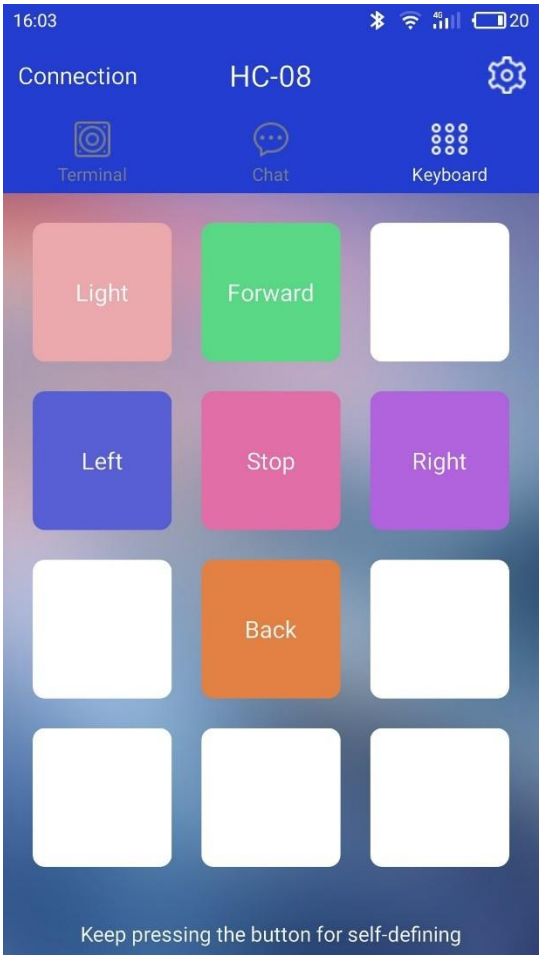


Then we can slide the screen to the right side by our finger and we can get the key pattern as below:



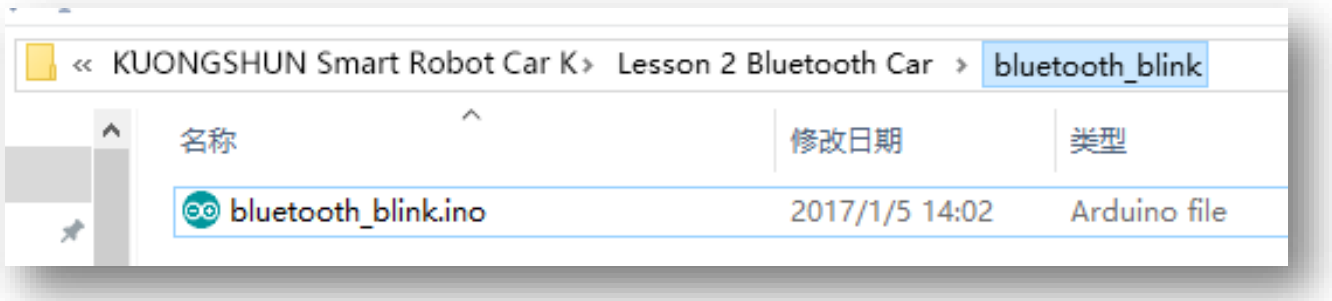
Finally, we set the definition of each button, we will take “go forward” for example, please see below, and the rest key-values are set in the same manner.


15:58
21
Connection
HC-08
Button editor
Forward
Status OFF
f
Character
Hexadecimal
Forward
Status OFF
f
Character
Hexadecimal
Background color
Clear
OK
Cancel
Keep pressing the button for self-defining


16:03
20
Connection
HC-08
Terminal
Chat
Keyboard
Light
Forward
Left
Stop
Right
Back
Keep pressing the button for self-defining

Ⅲ. Testing

Open the code file in the path "\kuongshun Smart Robot Car Kit V3.0\bluetooth_blink \bluetooth_blink.ino" and upload the program to the UNO board.



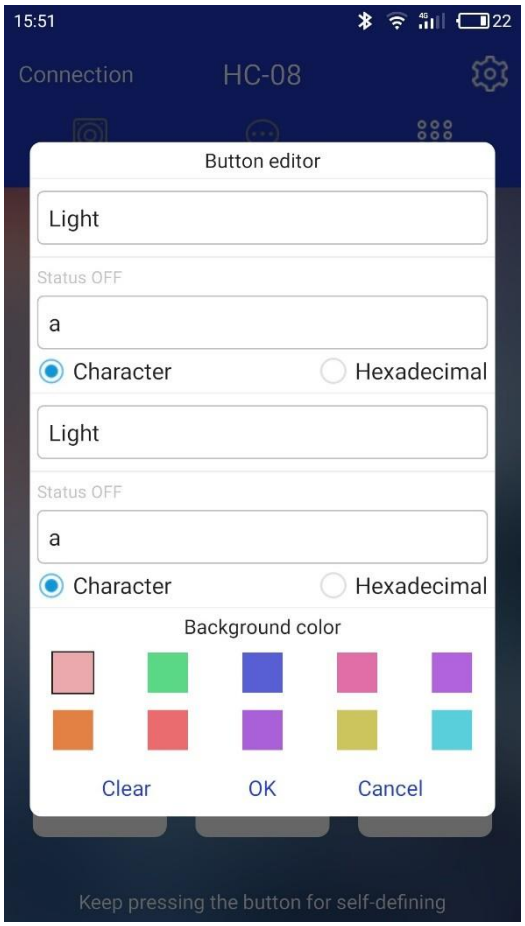
Code preview: //

```
#define LED 13    //Define 13 pin for LED
bool state = LOW; //The initial state of the function is defined as a low level
char getstr;     //Defines a function that receives the Bluetooth character

void setup() {
  pinMode(LED, OUTPUT);
  Serial.begin(9600);
}

//Control LED sub function
void stateChange() {
  state = !state;
  digitalWrite(LED, state);
}

void loop() {
  //The Bluetooth serial port to receive the data in the function
  getstr = Serial.read();
  if(getstr == 'a'){
    stateChange();
  }
}
```

Disconnect it from the computer, and then switch on the car's power supply. (TIPS: The Bluetooth module should be pulled out when you upload the program, or it will be failed to upload the program.)

Open APP

After connecting the phone to the car through Bluetooth, we set data as below:



After set-up, press this button. You will find that light on the UNO board changes with the switch.

The code

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

The purpose of this block of code is to set the baud rate of the UNO control board as 9600 and open the serial port. In this way, they can communicate with each other, because the original baud rate of the Bluetooth module is 9600.

```
getstr = Serial.read(); //The Bluetooth serial port to receive the data in the function
if(getstr == 'a'){
    stateChange();
}
```

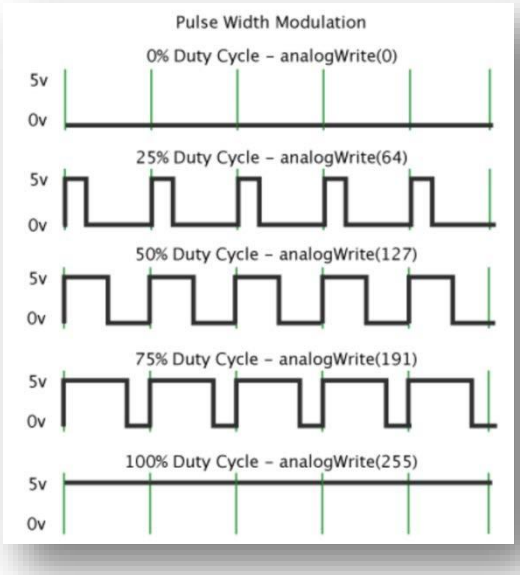
This function is executed repeatedly within the circulating function. It will first read data from the serial port and then check the data. If it meets the condition, it will execute the corresponding sub-function. For example, if it reads the letter ‘a’ from the serial port, it will execute the sub-function responsible for switching on/off the LED light.

IV. Make a Bluetooth Car

When the car turns left or right, it's not necessary to set the speed too fast. On the contrary, we need to control the speed of car. But how to control?

The answer is PWM.

PWM is the abbreviation of "Pulse Width Modulation", is called pulse modulation in short, is an effective technology to control analog circuit with digital output of microprocessor, car is used to change speed of motor by altering duty cycle of a square wave. In other words, connect and break circuit between two sides of motor constantly, is switch of holding motor work, motor will not be off when power is off because of the fast speed. So we can control speed of car if we control specific value of power on time and power off time. The speed of car will be max when circuit is holding still. The speed of car will be minimum if circuit is holding off. The speed of car will be median in half time. PWM is a technology to get analog quantity through digital method. A square wave is formed by digital control, square wave signal only have two state of on and off (That is high-low of digital pins).Simulate voltage changing from 0 to 5V by controlling specific value of duration on and off time. Occupied time of on (That is high level in academy) is called pulse width, so PWM is also called pulse width modulation. Let's learn about PWM through five square waves below.



Green vertical line above represent a period of square wave. The value written into every analogWrite(pin,value) corresponds to the percentage, the percentage is also called Duty Cycle, refer to the percentage gotten from specific value between duration high level and low level time in a period. In figure, from top to bottom, the first square wave, duty cycle is 0%, corresponding value is 0. Output circuit current is minimum, motor hold still. The longer duration time is, the bigger circuit current motor

gets, the faster the speed is. So, the final one's duty cycle is 100%, corresponding value is 255, motor rotates in full speed. 50% is medium hyponastic rotate speed, 25% is relatively slower, even can't start (The circuit current is relatively big to start motor because of static friction). PWM is mostly used to adjust light of LED and rotate speed of motor, wheel speed controlled by motor is easily be controlled. The advantage of PWM can be more reflected when you play with some Arduino cars.

```
analogWrite(pin, value)；
```

analogWrite() is used to write analog value of 0 to 255 for PWM ports. What you need to note is that, analogWrite() is only used to digital pins with function of PWM. Pins with function of PWM in UNO are only digital pins of 3，5，6，9，10，11.

Our car's speed is controlled by connecting pin5 and pin6 of ENA and ENB. The program below, have set a digital function int carSpeed = 150;

The speed is controlled in below program, so you can control the speed on your own.

```
analogWrite(ENA,    carSpeed);
analogWrite(ENB,    carSpeed);
```

After learning the basic knowledge, we will upload the program as below to the car, open the code file in the path "\kuongshun Smart Robot Car Kit V3.0\bluetooth_car\ bluetooth_car.ino" and then upload the program to the UNO control board.

Code preview:

```
#define ENA 5
#define ENB 6
#define IN1 7
#define IN2 8
#define IN3 9
#define IN4 11
#define LED 13

unsigned char carSpeed = 150;
bool state = LOW;
```

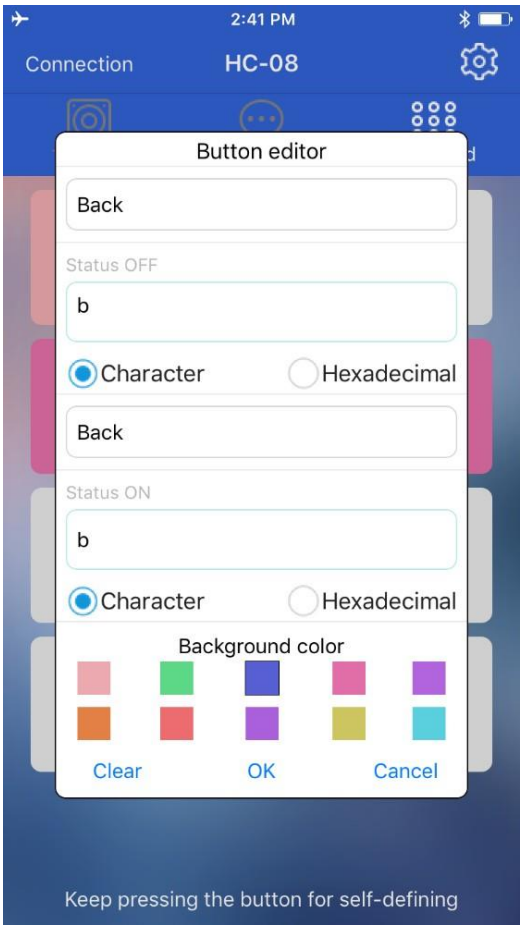
```
char getstr;

void forward(){
  digitalWrite(ENA,HIGH);
  digitalWrite(ENB,HIGH);
  digitalWrite(IN1,HIGH);
  digitalWrite(IN2,LOW);
  digitalWrite(IN3,LOW);
  digitalWrite(IN4,HIGH);
  Serial.println("Forward");
}

void back(){
  digitalWrite(ENA,HIGH);
  digitalWrite(ENB,HIGH);
  digitalWrite(IN1,LOW);
  digitalWrite(IN2,HIGH);
  digitalWrite(IN3,HIGH);
  digitalWrite(IN4,LOW);
  Serial.println("Back");
}

void left(){
  analogWrite(ENA,carSpeed);
  analogWrite(ENB,carSpeed);
  digitalWrite(IN1,LOW);
  digitalWrite(IN2,HIGH);
  digitalWrite(IN3,LOW);
  digitalWrite(IN4,HIGH);
  Serial.println("Left");
}

void right(){
  analogWrite(ENA,carSpeed);
  analogWrite(ENB,carSpeed);
  digitalWrite(IN1,HIGH);
  digitalWrite(IN2,LOW);
  digitalWrite(IN3,HIGH);
  digitalWrite(IN4,LOW);
  Serial.println("Right");
}

void stop(){
  digitalWrite(ENA,LOW);
```

```
  digitalWrite(ENB,LOW);
  Serial.println("Stop!");
}

void stateChange(){
  state = !state;
  digitalWrite(LED, state);
}

void setup() {
  Serial.begin(9600);
  pinMode(LED, OUTPUT);
  pinMode(IN1,OUTPUT);
  pinMode(IN2,OUTPUT);
  pinMode(IN3,OUTPUT);
  pinMode(IN4,OUTPUT);
  pinMode(ENA,OUTPUT);
  pinMode(ENB,OUTPUT);
  stop();
}

void loop() {
  getstr = Serial.read();
  switch(getstr){
    case 'f': forward(); break;
    case 'b': back();  break;
    case 'l': left();  break;
    case 'r': right();  break;
    case 's': stop();  break;
    case 'a': stateChange(); break;
    default:  break;
  }
}
```

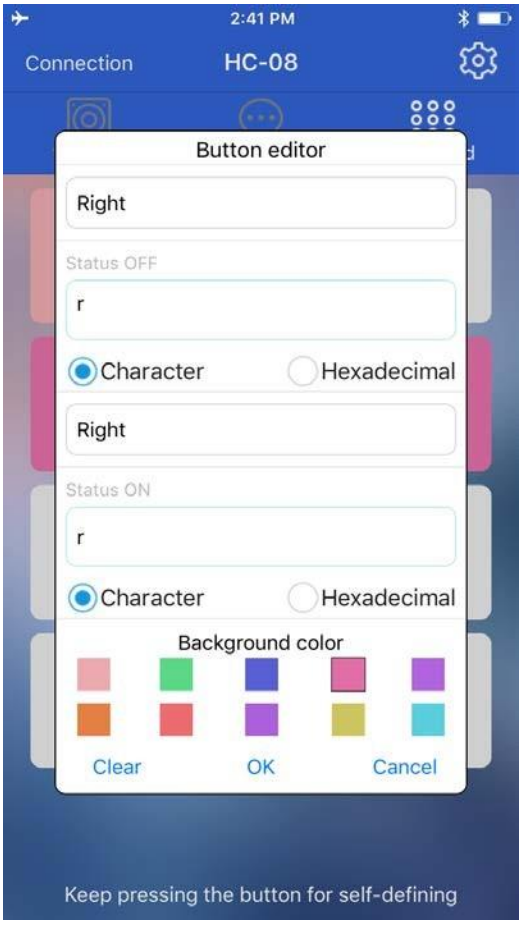
Switch on the power supply of the vehicle and put it on the ground.

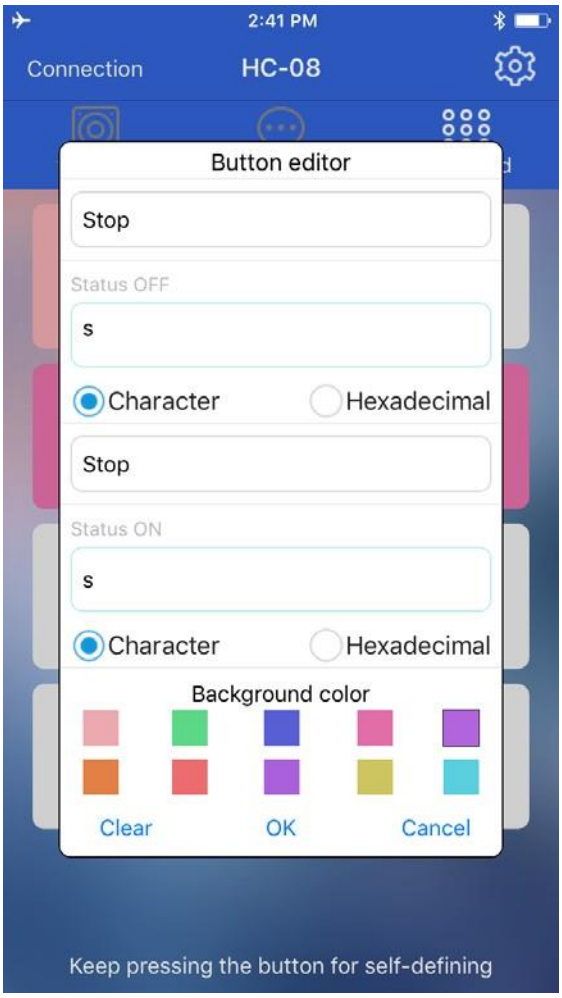
Open the mobile APP, and set up parameters as follows.

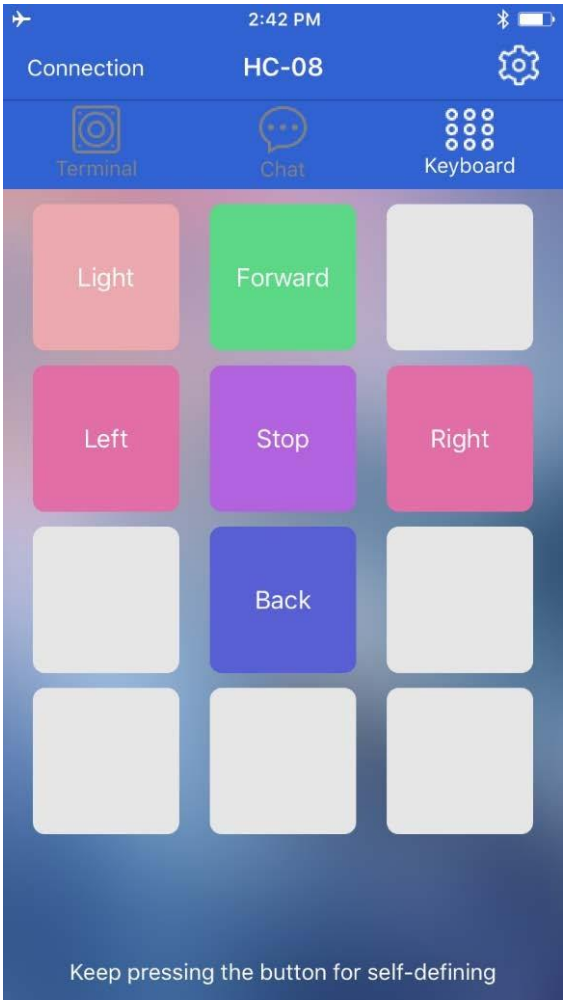

2:40 PM
Connection
HC-08
Button editor
Forward
Status OFF
f
Character
Hexadecimal
Forward
Status ON
f
Character
Hexadecimal
Background color
Clear
OK
Cancel
Keep pressing the button for self-defining


2:41 PM
Connection
HC-08
Button editor
Back
Status OFF
b
Character
Hexadecimal
Back
Status ON
b
Character
Hexadecimal
Background color
Clear
OK
Cancel
Keep pressing the button for self-defining

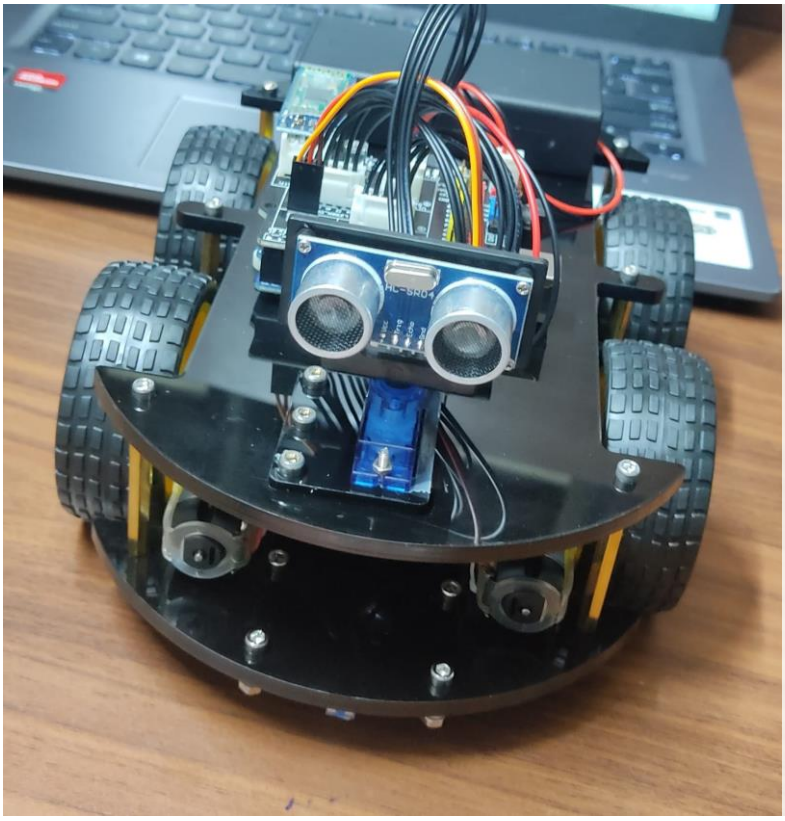

2:40 PM
Connection
HC-08
Button editor
Left
Status OFF
l
Character
Hexadecimal
Left
Status ON
l
Character
Hexadecimal
Background color
Clear
OK
Cancel
Keep pressing the button for self-defining


2:41 PM
Connection
HC-08
Button editor
Right
Status OFF
r
Character
Hexadecimal
Right
Status ON
r
Character
Hexadecimal
Background color
Clear
OK
Cancel
Keep pressing the button for self-defining





Now we can control the car by Bluetooth and play with it.



Infrared
Controlling Car

Infrared Controlling Car

Infrared Controlling Car

The points of section

Infrared remote control is a widely used method for remote control. The car has been equipped with infrared receiver and thus allows it to be controlled using the infrared remote controller.

Learning parts:

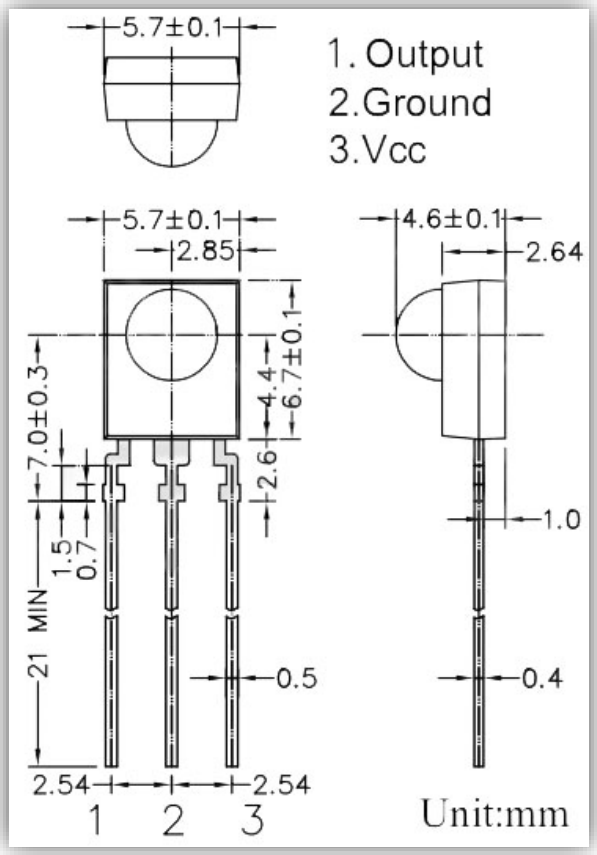
- *Understand the infrared remote controller and the receiver*
- *Understand the remote control principles*

Preparations:

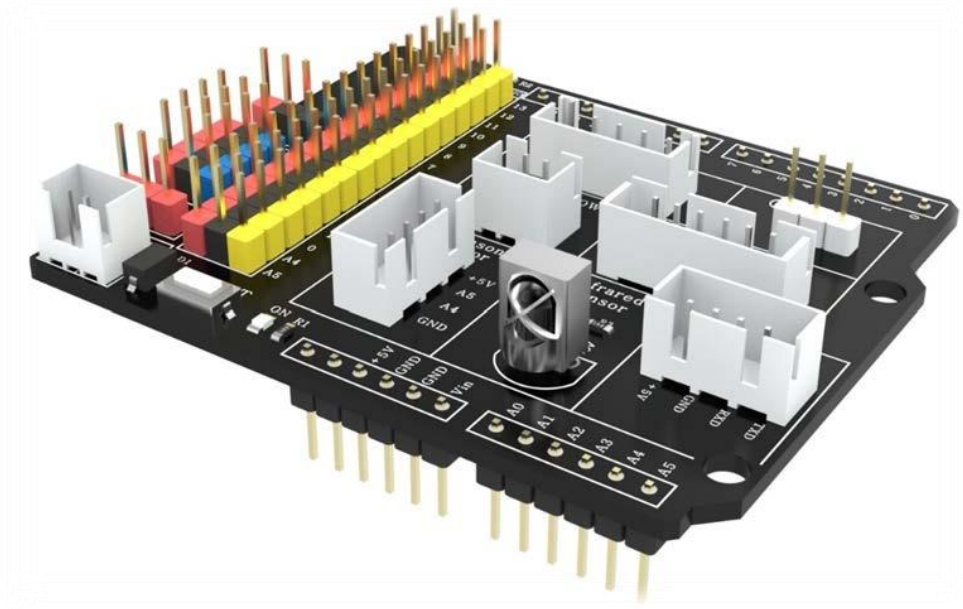
- *A car (with battery)*
- *A USB cable*
- *IR receiving module and IR remote*

Ⅰ. IR Receiving Module and IR Remote

The data of IR receiver sensor is as below:



The connection of receiver module is as below:

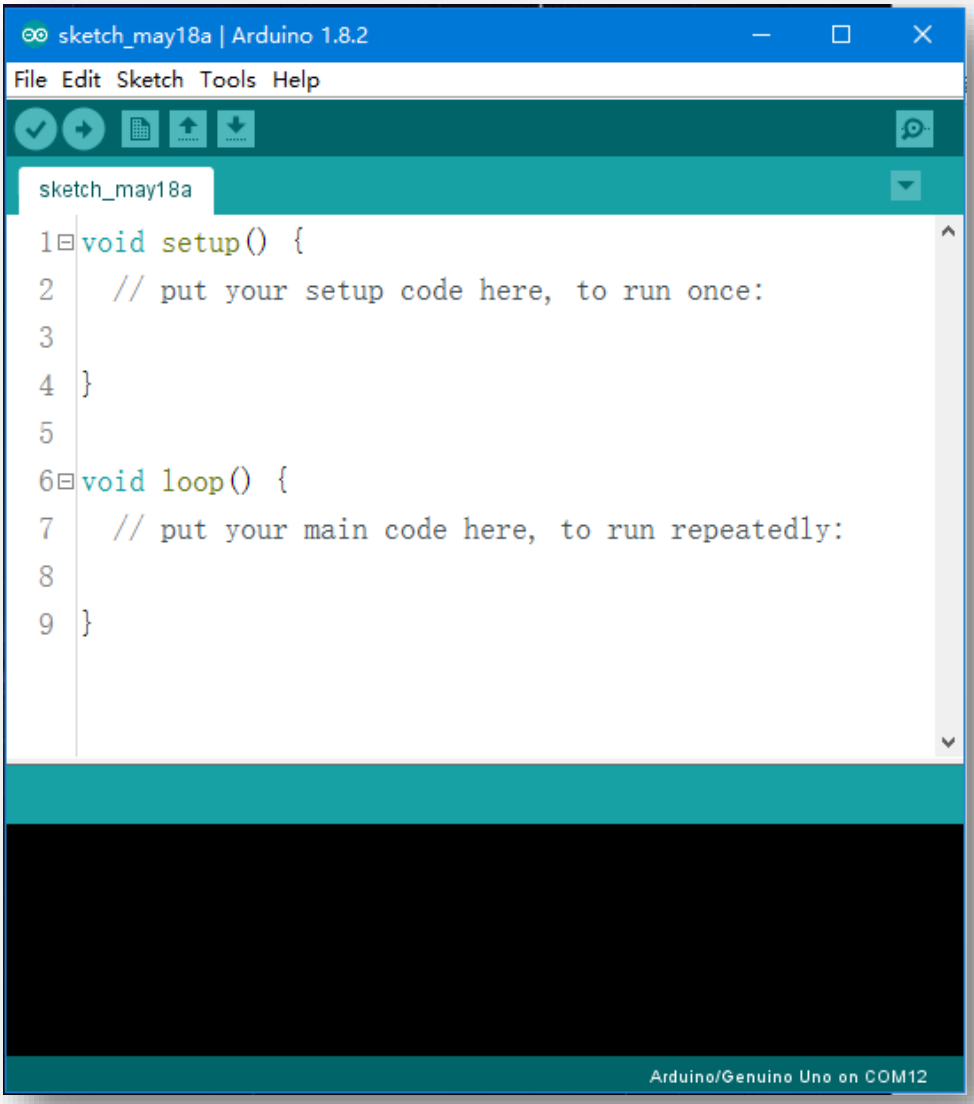


This is IR remote:

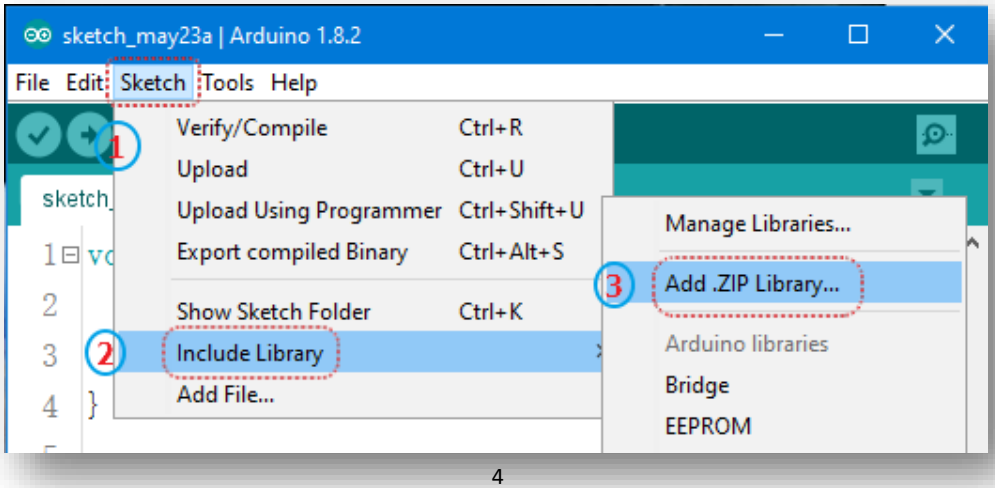
Ⅱ. Testing program

Because in this program, we need to use the library so that we need to add library file at first.

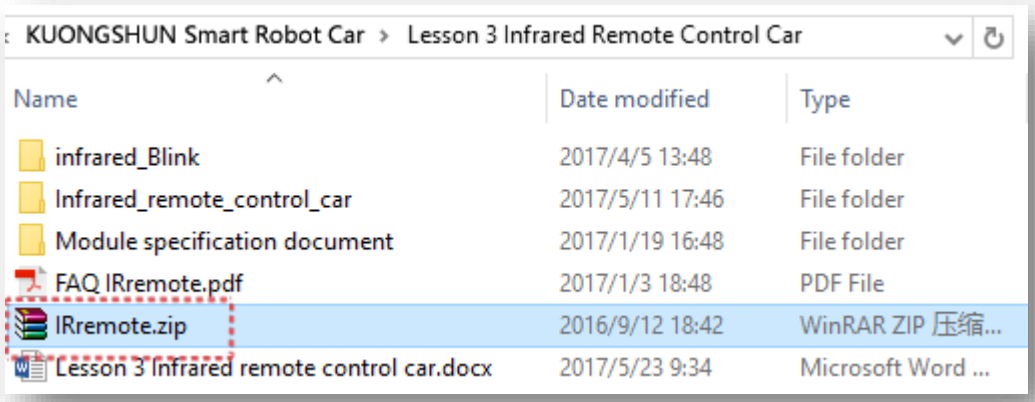
Connect the UNO to the computer and open the Arduino IDE.



Click Sketch --> Include Library --> Add .ZIP Library... --> then select the library as blow.

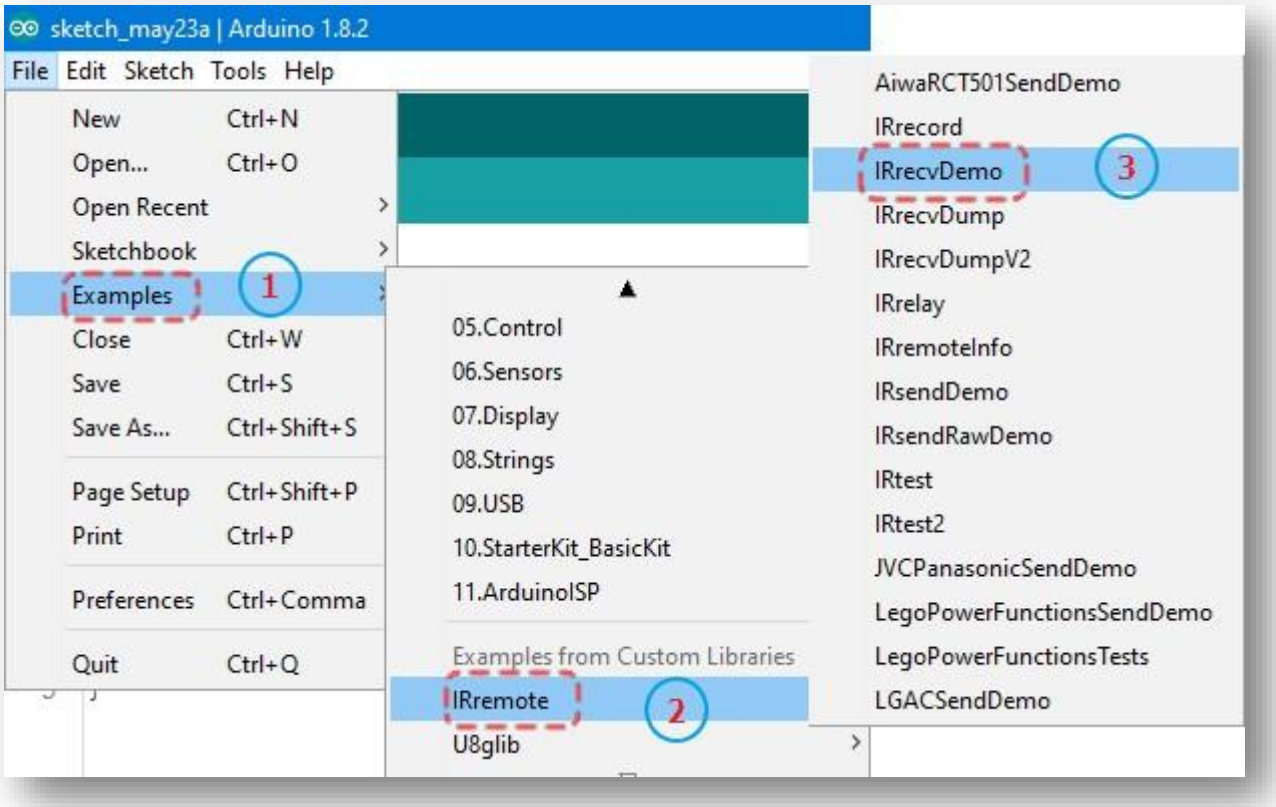


The file name of ZIP library must be IRremote.zip.

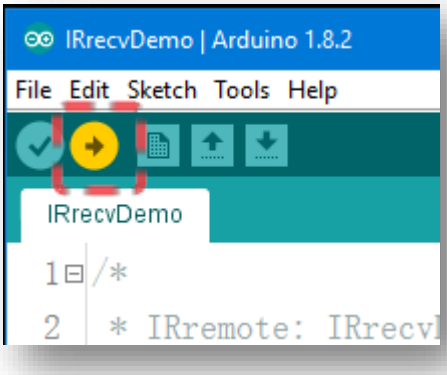


Must be compiled with this library file, which is a specially modified library file.

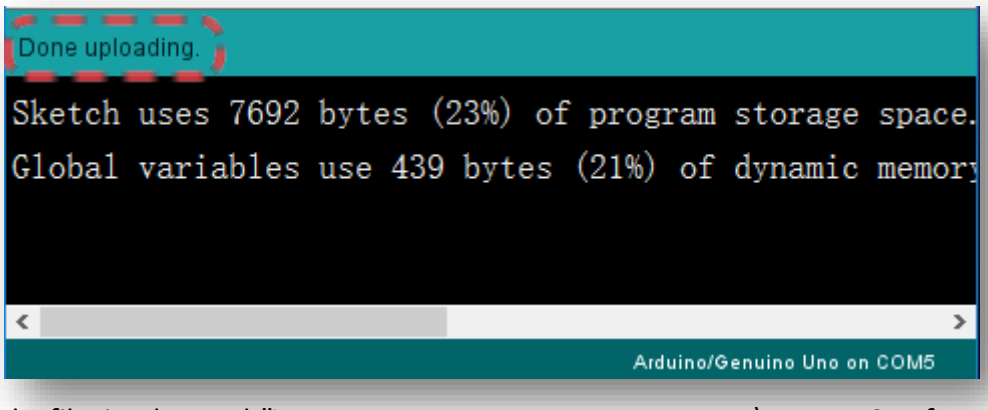
Select an IRremote example



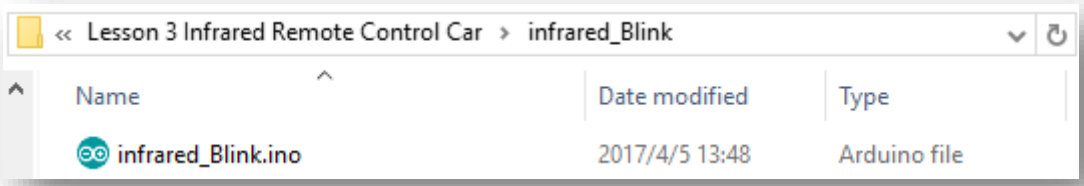
Click compile button.



Done compiling. If not, that the IRremote library was not installed successfully. Please go to add the IRremote library again.



Open the code file in the path"\KUONGSHUN Smart Robot Car Kit V3.0\Lesson 3 Infrared Remote Control Car\infrared_Blink\infrared_Blink.ino" and upload the program to the controller board.



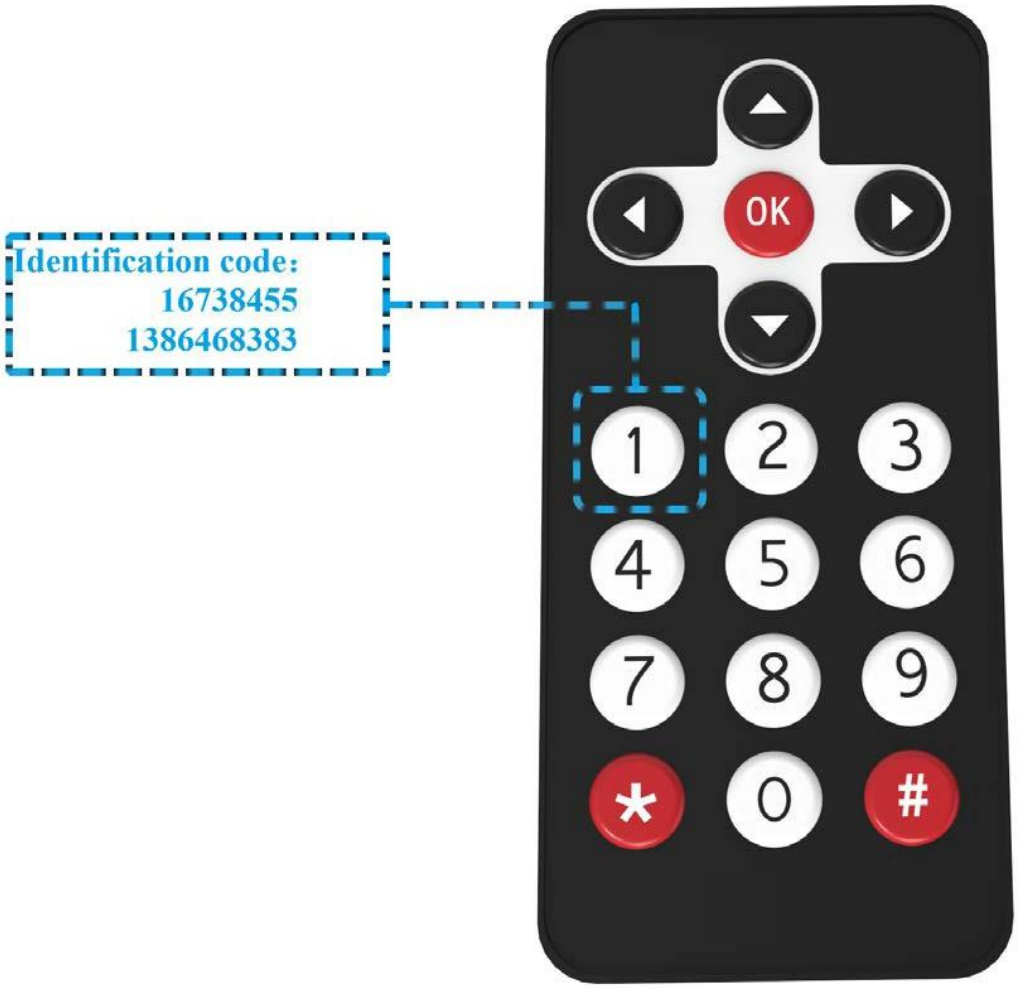
Code preview :

```
#include

<IRremote.h> #define

                          //Infrared signal receiving pin
#define LED      13       //define LED pin
#define L        16738455
#define UNKNOWN L 1386468383

bool state = LOW;          //define default input mode
unsigned long val;

IRrecv irrecv(RECV_PIN);   //initialization
decode_results results;   //Define structure type

void stateChange() {
  state = !state;
  digitalWrite(LED, state);
}

void setup() {
```

```
  pinMode(LED, OUTPUT); //initialize LED as an output
  Serial.begin(9600); // debug output at 9600 baud
  irrecv.enableIRIn();  // Start receiving
}

void loop() {
  if (irrecv.decode(&results)) {
    val = results.value;
    Serial.println(val);
    irrecv.resume();     // Receive the next value
    delay(150);
    if(val == L || val == UNKNOWN_L) {
      stateChange();
    }
  }
}
```

After disconnecting the car to the computer, you can turn on the power switch and put the car on the ground.

Press the button “1” facing toward the car, observe the car, and you will find the LED with a label ”L” on the extension board turn off.



Ⅲ. Introduction of principle

1. Working principle

The universal infrared remote controlling system consists of two parts: sending and receiving, the sending part consists of an IR remote controller, the receiving part consists of an infrared receiving tube. The signals that sent by IR remote controlling is a serial of binary pulse code. In order to be free from distraction of other infrared signals during wireless transportation, it's general to modulate it at given carrier frequency, and then launch it through infrared emitted phototransistor. Infrared receiving tube filters out other noise waves, only receives signals of given frequency and restores them to binary pulse code that is demodulation. Built-in receiving tube transform light signals that are sent from infrared light-emitting diode to weak electric signals, signals are enlarged through amplifier inside IC, and through automatic gain controlling, band-pass filtering, demodulation, wave shaping and be restored to original encoding sent by remote control, recognize the circuit by coding that is input to electric appliance through signal output pin of infrared receiving module.

2. Protocol of infrared remote controlling

The coding scheme of matched IR remote controlling is: NEC protocol.

Next, let's learn what NEC protocol is.

Features:

（1）8 address bit, 8 order bit

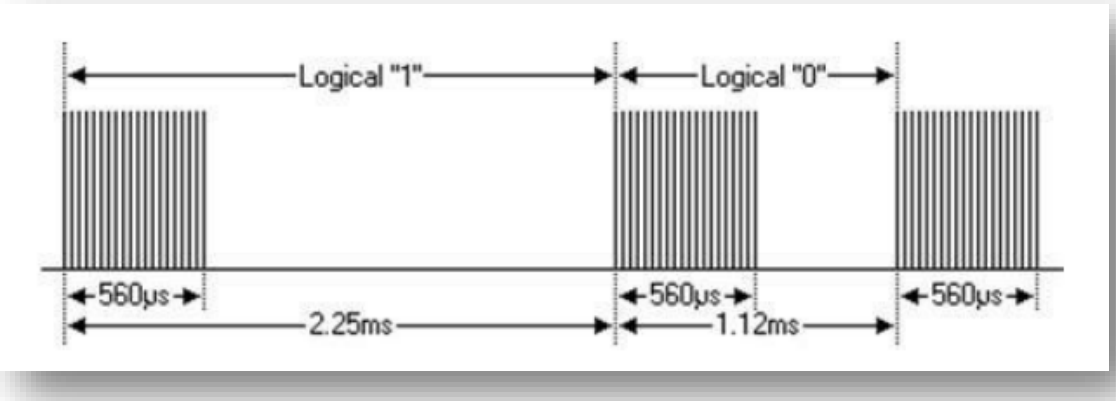
（2） Address bit and order bit are transmitted twice in order to guarantee reliability

（3） Pulse position modulation

（4） Carrier frequency is 38kHz

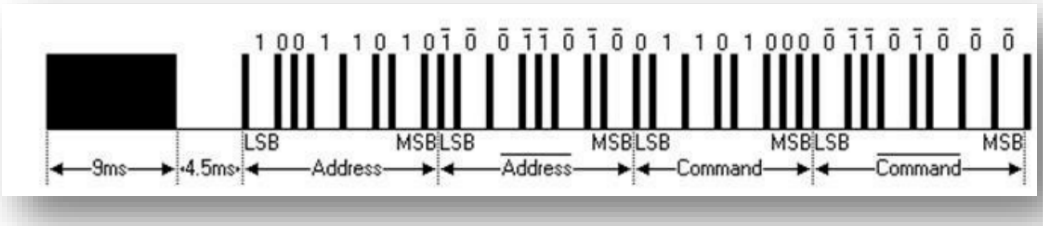
（5） Time of every bit is 1.125ms or 2.25ms

Definitions of logical 0 and 1 are as below:



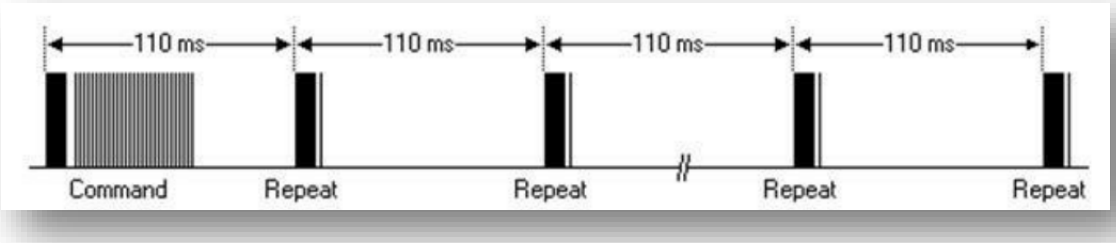
The protocol is as below:

Press instant loosen transmission pulse:



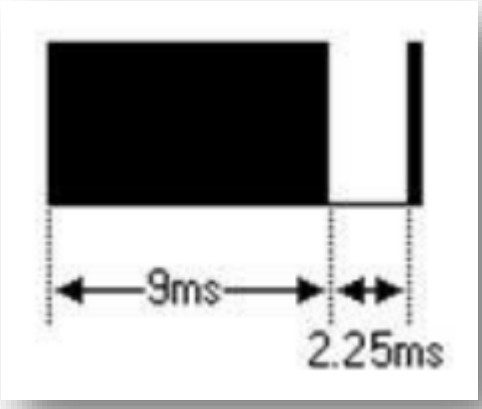
Note: This is protocol of sending LSB (least-significant bit) firstly. Transport address of the above pulse is 0x59, order is 0x16. One message starts from a high level of 9ms, the follow is a low level of 4.5ms, (Two level forms guidance code) and through address code and order code. Address and order are transmitted twice. In the second time, all bits are inverted the opposite, can be used to confirm the receiving messages to be used. Total sending time is fixed, if you are not interested in it, you can ignore reliability of invert, and can expand address and order at 16 bit! Because the fact that length repeat if every bit is opposite.

Press transmitted pulse loosened after a time.



Once a command was sent, even if the button of remote controlling is pressed. When button is still pressed, the pulse of first 110ms is different from above, duplicated code is transmitted after every 110ms. Duplicated code is made up of a high level pulse of 9ms and a low level of 2.25 and a high level of 560μs.

Repeat pulse：



Note: After impulse waveform enters into integration of sensor, owing to the fact that integration of sensor should be decoded, signal magnified and plastic, you should note the time when there are no infrared signals, its output terminal is high level, is low level when there are signals. So the level of output signal is opposite to transmitting terminal. Everybody can see receiver pulse through oscilloscope, understand program with wave form seen.

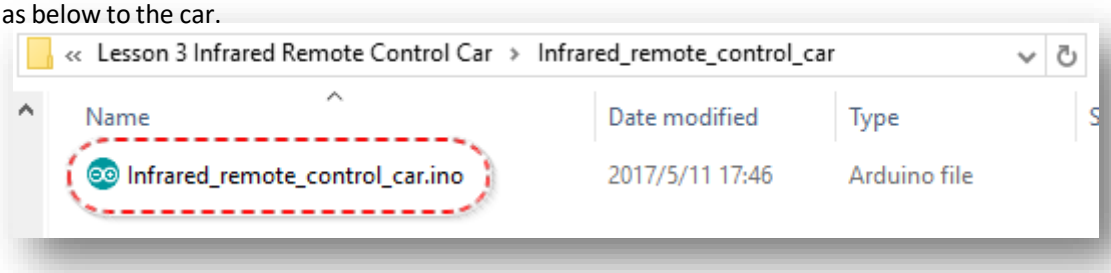
3. The idea of programming remote control car

According to the characteristic of NEC code and wave of receiving-end, this experiment divides wave of receiving-end into four parts: leading code (Pulse of 9ms and 4.5ms)、address code (including 8-bit address code and 8-bit address fetch)、
16-bit address code (including 8-bit address code and 8-bit address fetch)、16-bit order code(including 8-bit order code and 8-bit order fetch)、repeat code(be made up of pulse of 9ms、2.25ms、560us). Exploit the timer to test high level and low level of wave received, being distinguished according to the time tested: logical“01”、 logical”1”、leading pulse、repeat pulse. Leading code and address code are judged whether correct, not be stored, owing to the fact that order code of each key is different, action is carried out by order code.
During car experiment, we just need to control the car to go forward and backward ,turn left and right, and stop ,which means we would need 5 keys and the value of them are as below:

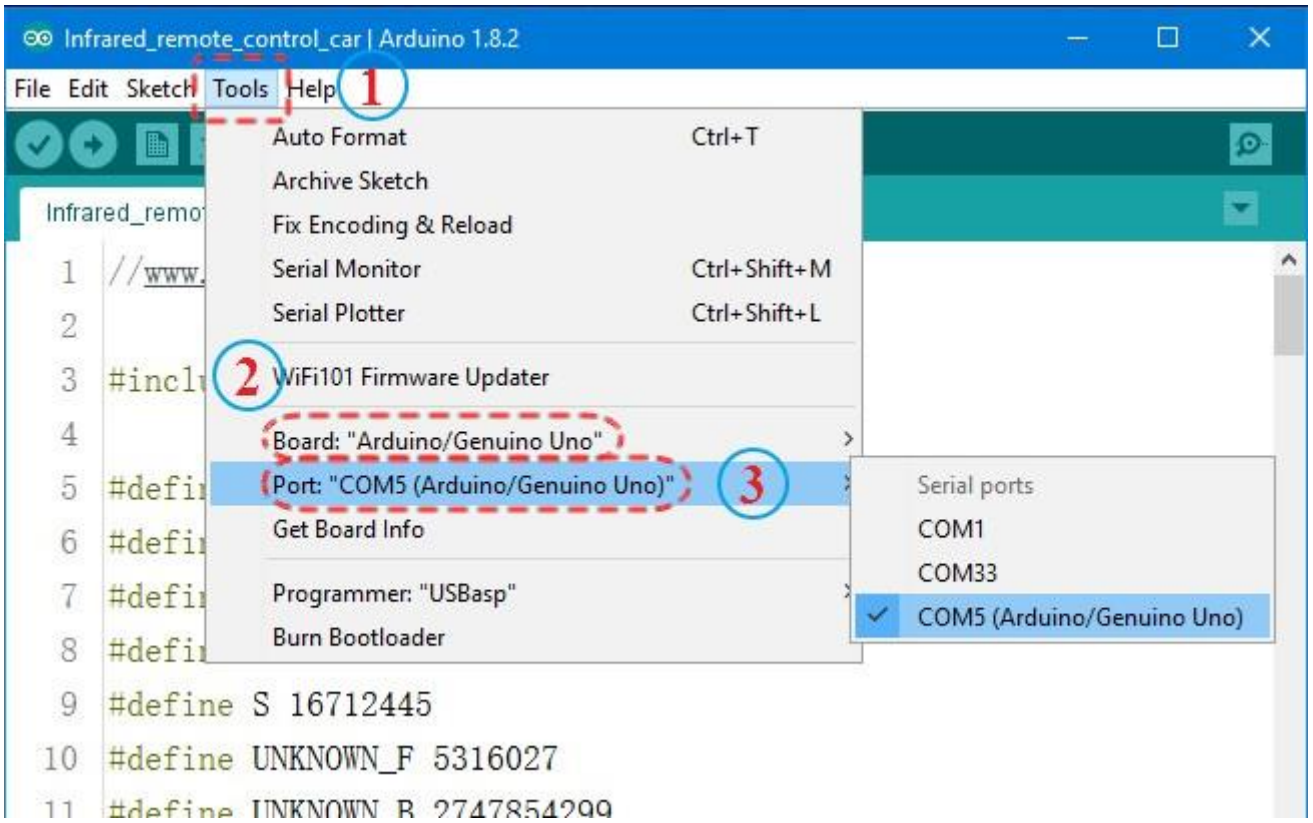
Remote control character	key value
Middle red button	16712445, 3622325019
Above triangle	16736925, 5316027
Below triangle	16754775, 2747854299
Left triangle	16720605, 1386468383
Right triangle	16761405, 553536955

IV. Make a remote controlling car

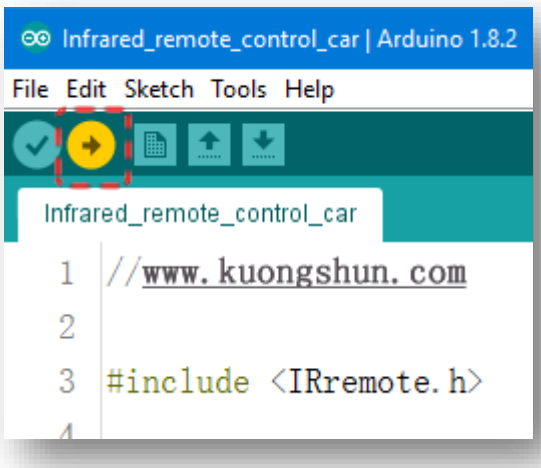
Open the code file in the path"\KUONGSHUN Smart Robot Car Kit V3.0 \Infrared_remote_control_car\ Infrared_remote_control_car.ino" and then upload the program as below to the car.



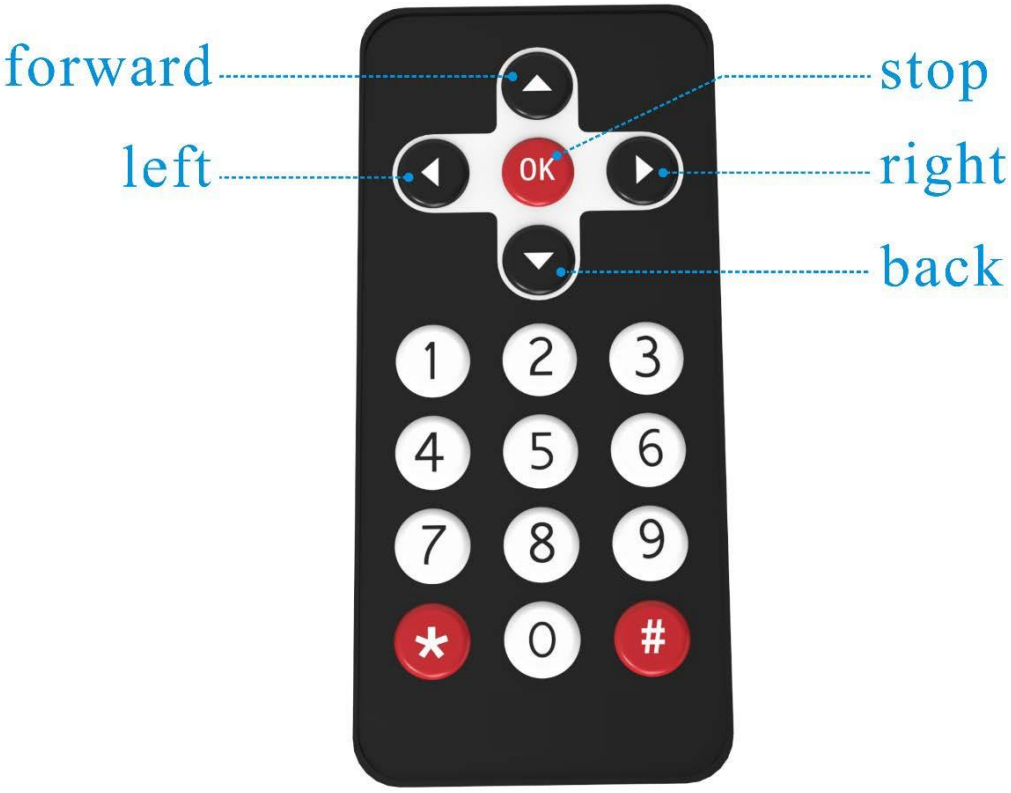
Select the Arduino Uno Board and Serial port.



Press the upload button



When done uploading, disconnecting the car to the computer. Then turn on the power switch and put the car on the ground. Press the button on the remote and you can see the car move accordingly as you command.



Voila, now you can play with the IR controlling car happily.

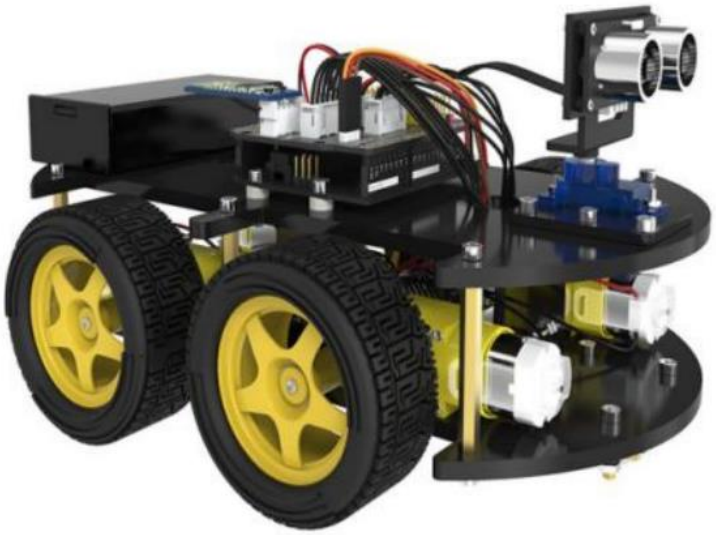
Code preview:

```
#include <IRremote.h>

#define F 16736925
#define B 16754775
#define L 16720605
#define R 16761405
#define S 16712445
#define UNKNOWN_F 5316027
#define UNKNOWN_B 2747854299
#define UNKNOWN_L 1386468383
#define UNKNOWN_R 553536955
#define UNKNOWN_S 3622325019
```

```
#define RECV_PIN  12
#define ENA 5
#define ENB 6
#define IN1 7
#define IN2 8
#define IN3 9
#define IN4 11
#define carSpeed 150

IRrecv irrecv(RECV_PIN);
decode_results results;
unsigned long val;
unsigned long preMillis;

void forward(){
  digitalWrite(ENA,HIGH);
  digitalWrite(ENB,HIGH);
  digitalWrite(IN1,HIGH);
  digitalWrite(IN2,LOW);
  digitalWrite(IN3,LOW);
  digitalWrite(IN4,HIGH);
  Serial.println("go forward!");
}
void back(){
  digitalWrite(ENA,HIGH);
  digitalWrite(ENB,HIGH);
  digitalWrite(IN1,LOW);
  digitalWrite(IN2,HIGH);
  digitalWrite(IN3,HIGH);
  digitalWrite(IN4,LOW);
  Serial.println("go back!");
}
void left(){
  analogWrite(ENA,carSpeed);
  analogWrite(ENB,carSpeed);
  digitalWrite(IN1,LOW);
  digitalWrite(IN2,HIGH);
  digitalWrite(IN3,LOW);
  digitalWrite(IN4,HIGH);
  Serial.println("go left!");
}
void right(){
  analogWrite(ENA,carSpeed);
```

```
  analogWrite(ENB,carSpeed);
  digitalWrite(IN1,HIGH);
  digitalWrite(IN2,LOW);
  digitalWrite(IN3,HIGH);
  digitalWrite(IN4,LOW);
  Serial.println("go right!");
}
void stop(){
  digitalWrite(ENA, LOW);
  digitalWrite(ENB, LOW);
  Serial.println("STOP!");
}

void setup() {
  Serial.begin(9600);
  pinMode(IN1,OUTPUT);
  pinMode(IN2,OUTPUT);
  pinMode(IN3,OUTPUT);
  pinMode(IN4,OUTPUT);
  pinMode(ENA,OUTPUT);
  pinMode(ENB,OUTPUT);
  stop();
  irrecv.enableIRIn();
}

void loop() {
  if (irrecv.decode(&results)){
    preMillis = millis();
    val = results.value;
    Serial.println(val);
    irrecv.resume();
    switch(val){
      case F:
      case UNKNOWN_F: forward(); break;
      case B:
      case UNKNOWN_B: back(); break;
      case L:
      case UNKNOWN_L: left(); break;
      case R:
      case UNKNOWN_R: right();break;
      case S:
      case UNKNOWN_S: stop(); break;
      default: break;
    }
```

```
  }
  else{
    if(millis() - preMillis > 500){
      stop();
      preMillis = millis();
    }
  }
}
```



Obstacle avoidance car

Obstacle avoidance car

Obstacle avoidance car

Obstacle avoidance car

Points of this section

The joy of learning, is not just know how to control your car, but also know how to protect your car.

So, make your car far away from collision.

Learning parts:

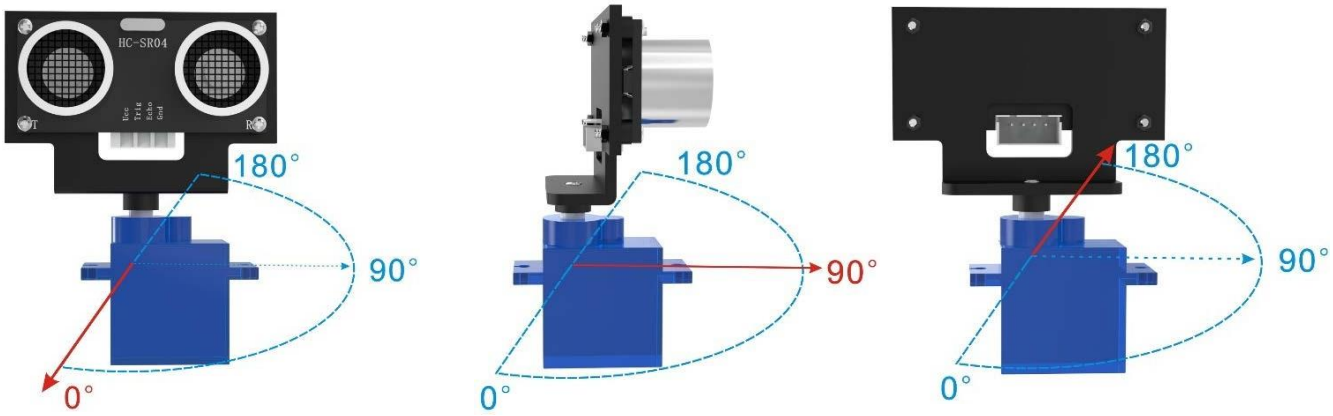
- *Learn how to assemble the ultrasonic module*
- *Be familiar with using steering*
- *Learn about the principle of car avoidance*
- *Use the program to make obstacle avoidance car come true*

Preparations:

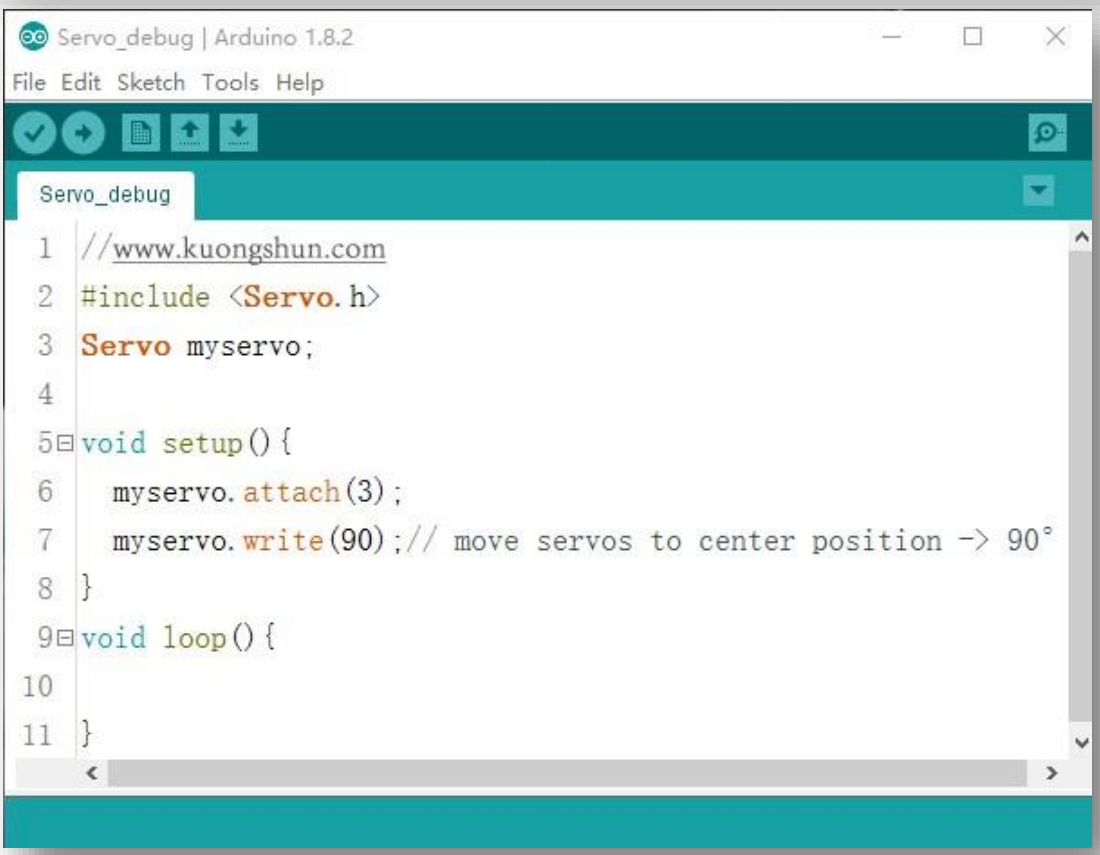
- *A car (with battery)*
- *A USB cable*
- *A suit of ultrasonic cradle head*

Ⅰ. Connection

When assemble the ultrasonic sensor module holder, the servo should also be debugged to ensure that the server can rotate 180 degrees.



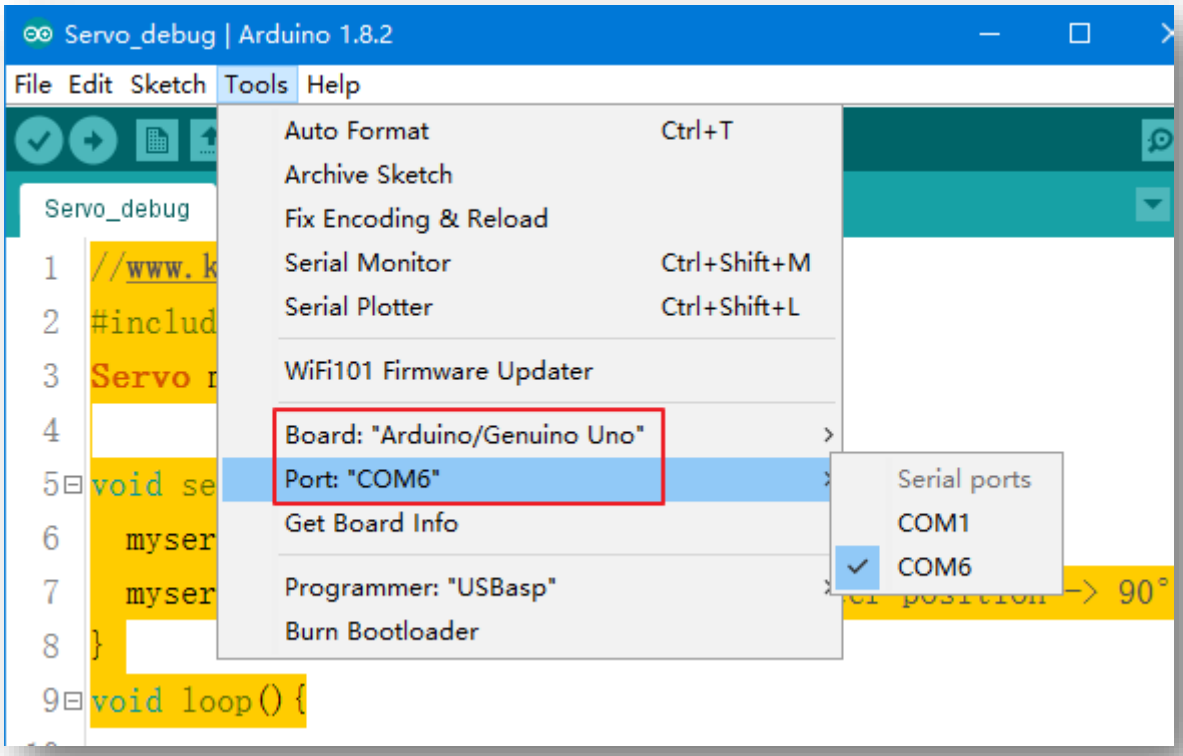
STEP1: Connect the UNO to the computer and open the Servo_debug code file in the path "\Lesson 4 Obstacle Avoidance Car\Servo_debug\ Servo_debug.ino".



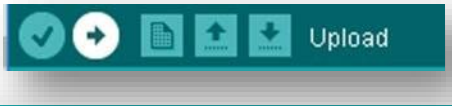
Servo_debug code preview:

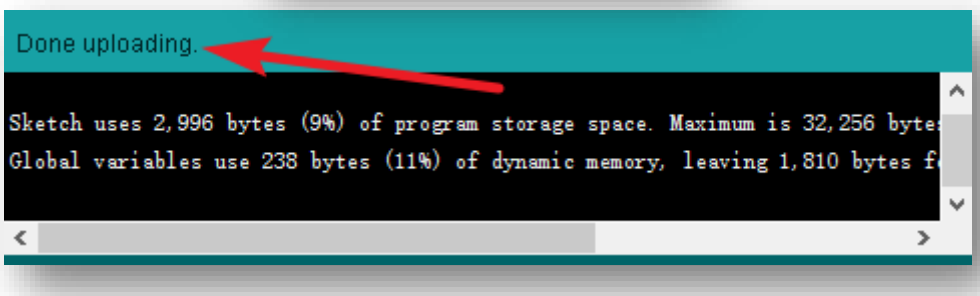
```
#include <Servo.h>
Servo myservo;

void setup(){
  myservo.attach(3);
  myservo.write(90);// move servos to center position -> 90°
}
void loop(){

}
```

STEP2: Select "Tool" --> "Port" and "Board" in the Arduino IDE.



STEP3: Click the arrows button to upload the code to the UNO controller board.

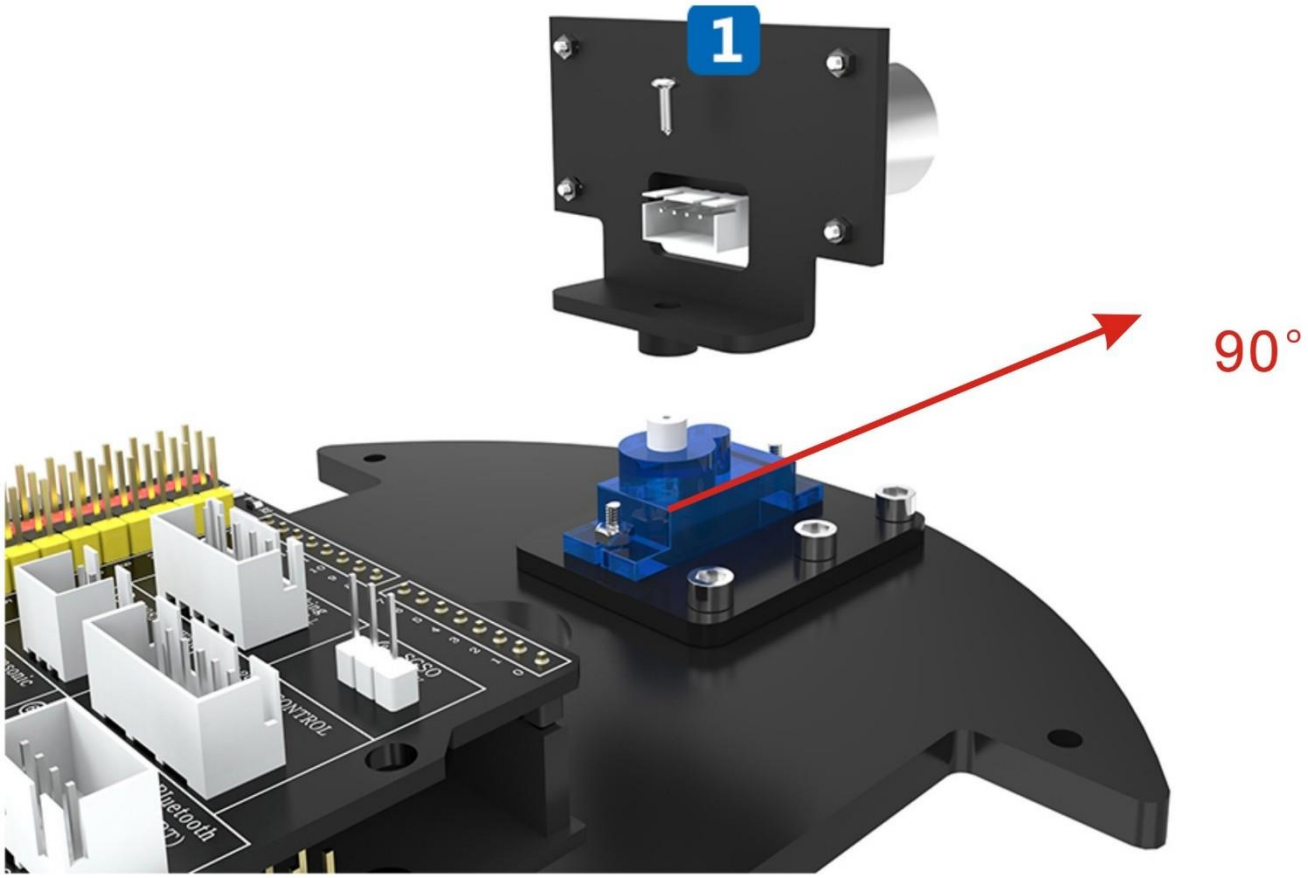




After done uploading, the servo will rotate to 90 degrees and stationary.

STEP4: Assemble the ultrasonic sensor module at 90 degrees.

The angle of each teeth on micro servo is 15 degree and if you install it on the middle of the direction of 90 degree, it will rotate to left or right by 15 degree, which means the actual degree of installing the micro servo is 15 degree or 105 degree.



FAQ about the servo motor.

1 Why does the micro servo rotate anticlockwise by15 degree each time I turn on the power?

This is normal to SG90 micro servo and it won't affect normal use with programs.

If you didn't control it with programs you can rotate it back to normal with your hand or plug off the wires connected with micro servo before you turn on the power.

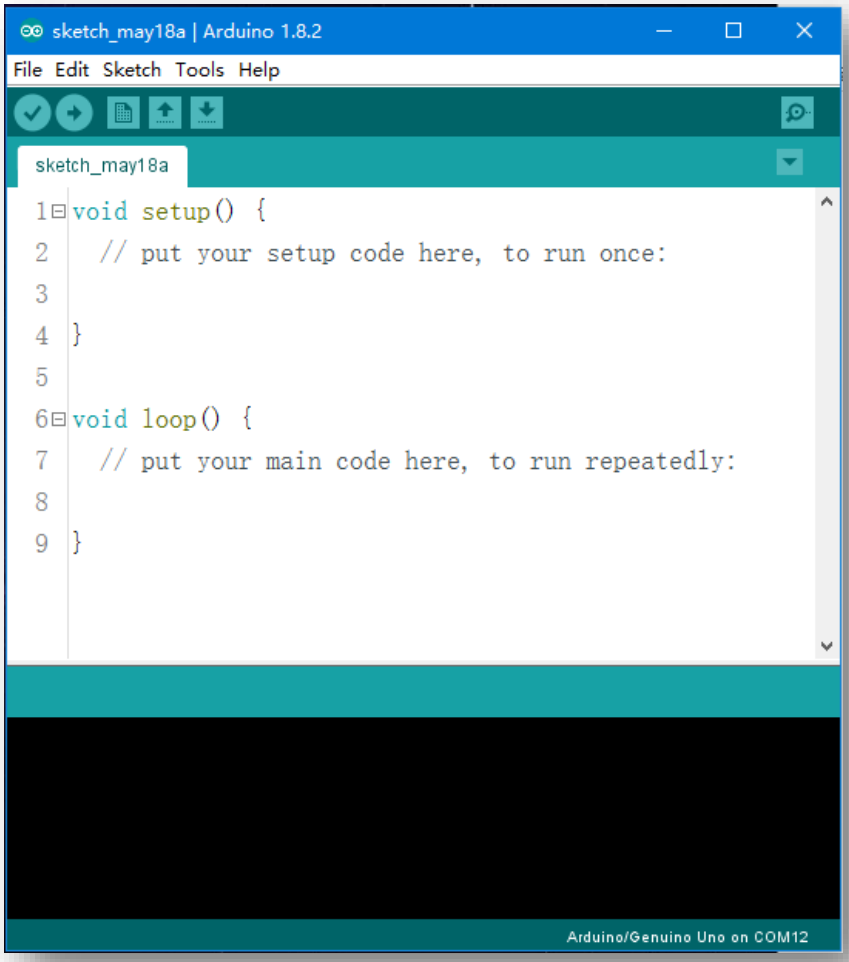
2 The micro servo is out of control and keeps rotating.

Use "myservo.write (angle)" to command the micron servo to the angle degree which has a range from 0 to 180. If it exceeds the range, the micro servo won't recognize this angle and will keep rotating.

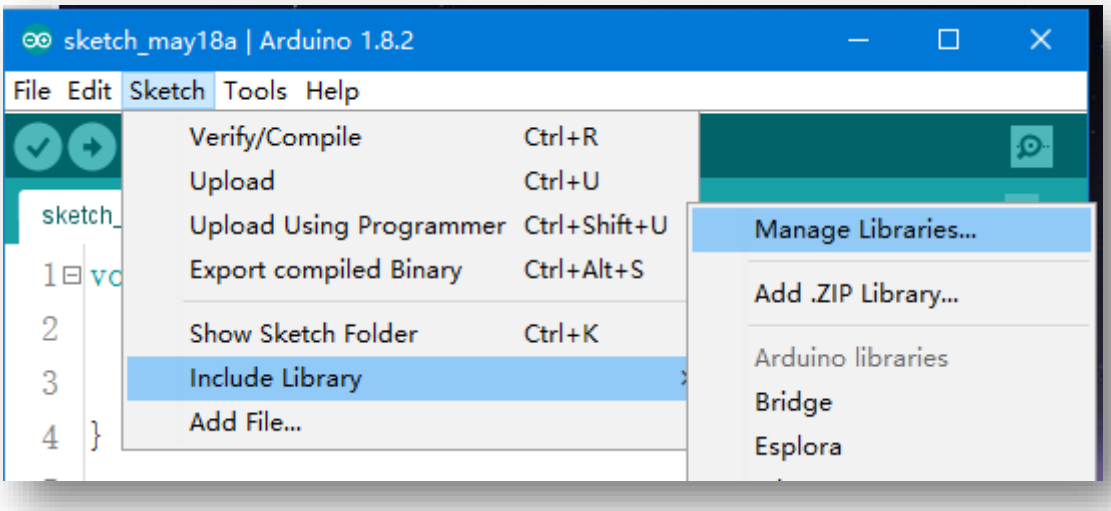
Ⅱ. Upload program

Because the program uses the library <servo.h>, so we need to install the library at first.

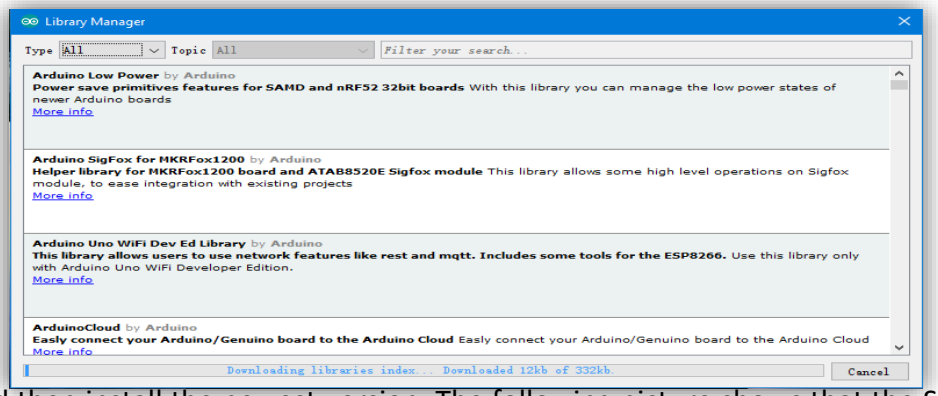
Open the Arduino software



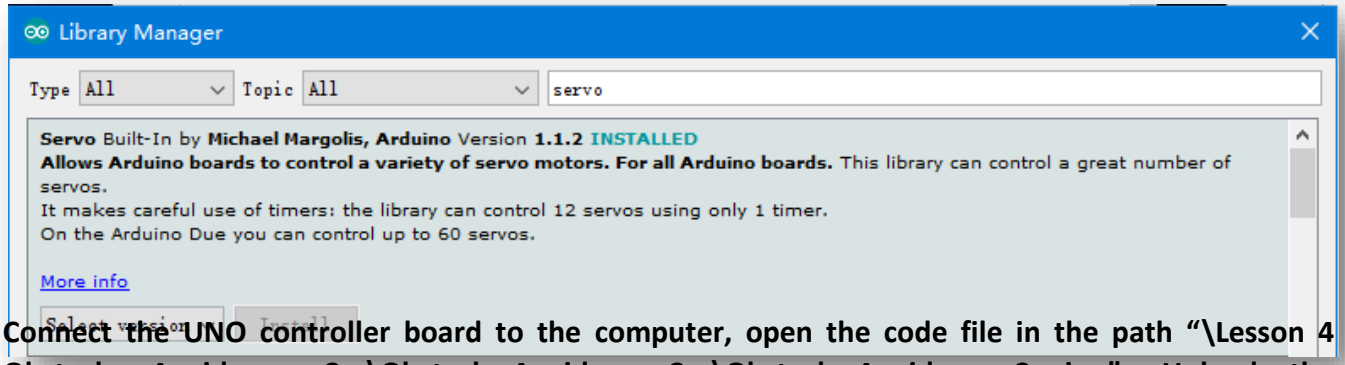
Select Sketch -> Include Library -> Manage Libraries…



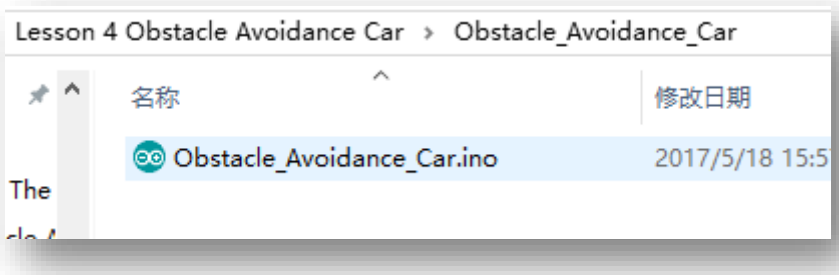
Waiting for “Downloading libraries index” to finish.



Search servo and then install the newest version. The following picture shows that the Servo library is already installed.



Connect the UNO controller board to the computer, open the code file in the path "\Lesson 4 Obstacle Avoidance Car\Obstacle_Avoidance_Car\Obstacle_Avoidance_Car.ino". Upload the program to the UNO board.



Code preview: //

```
#include <Servo.h>  //servo library
Servo myservo;     // create servo object to control servo

int Echo = A4;
int Trig = A5;

#define ENA 5
#define ENB 6
```

```
#define IN1 7
#define IN2 8
#define IN3 9
#define IN4 11
#define carSpeed 150
int rightDistance = 0, leftDistance = 0, middleDistance = 0;

void forward(){
  analogWrite(ENA, carSpeed);
  analogWrite(ENB, carSpeed);
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
  Serial.println("Forward");
}

void back() {
  analogWrite(ENA, carSpeed);
  analogWrite(ENB, carSpeed);
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
  Serial.println("Back");
}

void left() {
  analogWrite(ENA, carSpeed);
  analogWrite(ENB, carSpeed);
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
  Serial.println("Left");
}

void right() {
  analogWrite(ENA, carSpeed);
  analogWrite(ENB, carSpeed);
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
```

```
  Serial.println("Right");
}

void stop() {
  digitalWrite(ENA, LOW);
  digitalWrite(ENB, LOW);
  Serial.println("Stop!");
}

//Ultrasonic distance measurement Sub function
int Distance_test() {
  digitalWrite(Trig, LOW);
  delayMicroseconds(2);
  digitalWrite(Trig, HIGH);
  delayMicroseconds(20);
  digitalWrite(Trig, LOW);
  float Fdistance = pulseIn(Echo, HIGH);
  Fdistance= Fdistance / 58;
  return (int)Fdistance;
}

void setup() {
  myservo.attach(3);  // attach servo on pin 3 to servo object
  Serial.begin(9600);
  pinMode(Echo, INPUT);
  pinMode(Trig, OUTPUT);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
  pinMode(ENA, OUTPUT);
  pinMode(ENB, OUTPUT);
  stop();
}

void loop() {
    myservo.write(90);  //setservo position according to scaled value
    delay(500);
    middleDistance = Distance_test();

    if(middleDistance <= 20) {
      stop();
      delay(500);
      myservo.write(10);
```

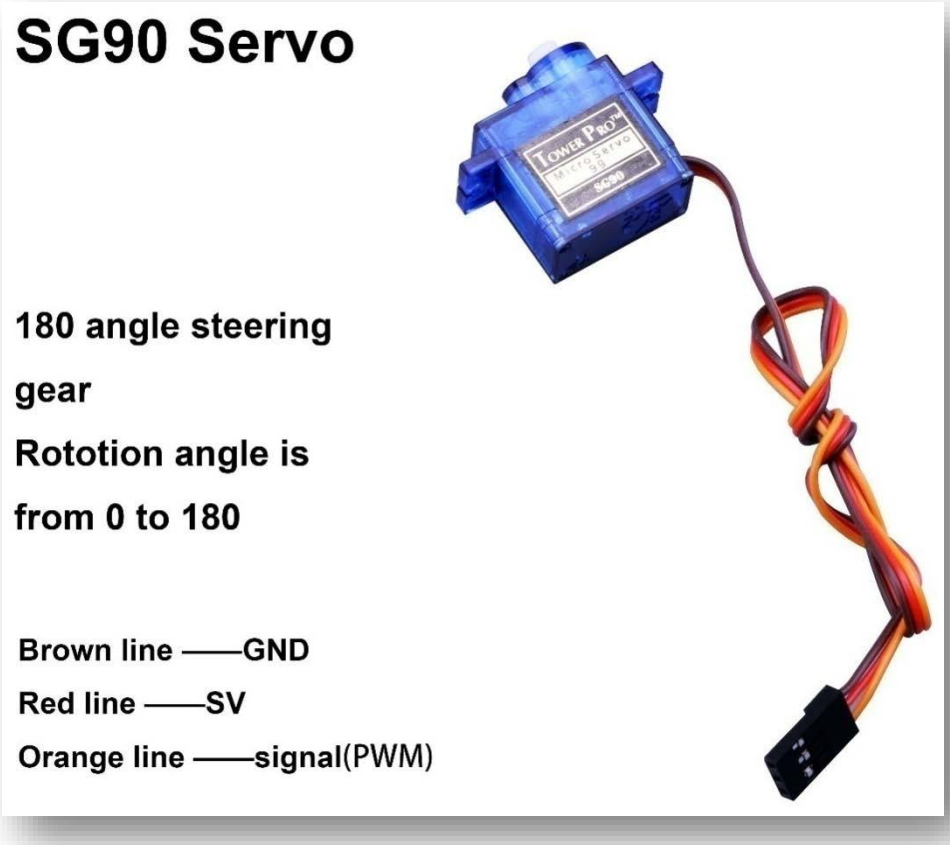
```
    delay(1000);
    rightDistance = Distance_test();

    delay(500);
    myservo.write(90);
    delay(1000);
    myservo.write(180);
    delay(1000);
    leftDistance = Distance_test();

    delay(500);
    myservo.write(90);
    delay(1000);
    if(rightDistance > leftDistance) {
      right();
      delay(360);
    }
    else if(rightDistance < leftDistance) {
      left();
      delay(360);
    }
    else if((rightDistance <= 20) || (leftDistance <= 20)) {
      back();
      delay(180);
    }
    else {
      forward();
    }
  }
  else {
     forward();
  }
}
```

After uploading the program to the UNO control board, disconnect the cable, put the vehicle on the ground and switch on the power supply.

You will see that the vehicle will move forward and the cloud platform keeps rotating to make the distance measuring sensors operate continuously. If there are obstacles ahead, the cloud platform will stop and the vehicle will change its direction to bypass the obstacle. After bypassing the obstacle, the cloud platform will keep rotating again and the vehicle will also move on.

Ⅲ. Introduction of principle

First of all, let's learn about the SG90 Servo:



Classification: 180 steering gear

Normally the servo has 3 controlling line: power supply, ground and sign.

Definition of the servo pins: brown line—— GND, red line——5V, orange ——signal.

How does servo work:

The signal modulation chip in the servo receives signals from the controller board then the servo will get the basic DC voltage. There is also a reference circuit inside the servo which will produce a standard voltage. These two voltages will compare to each other and the difference will be output. Then the motor chip will receive the difference and decide the rotational speed, direction and angel. When there is no difference between the two voltages, the servo will stop.

How to control the servo:

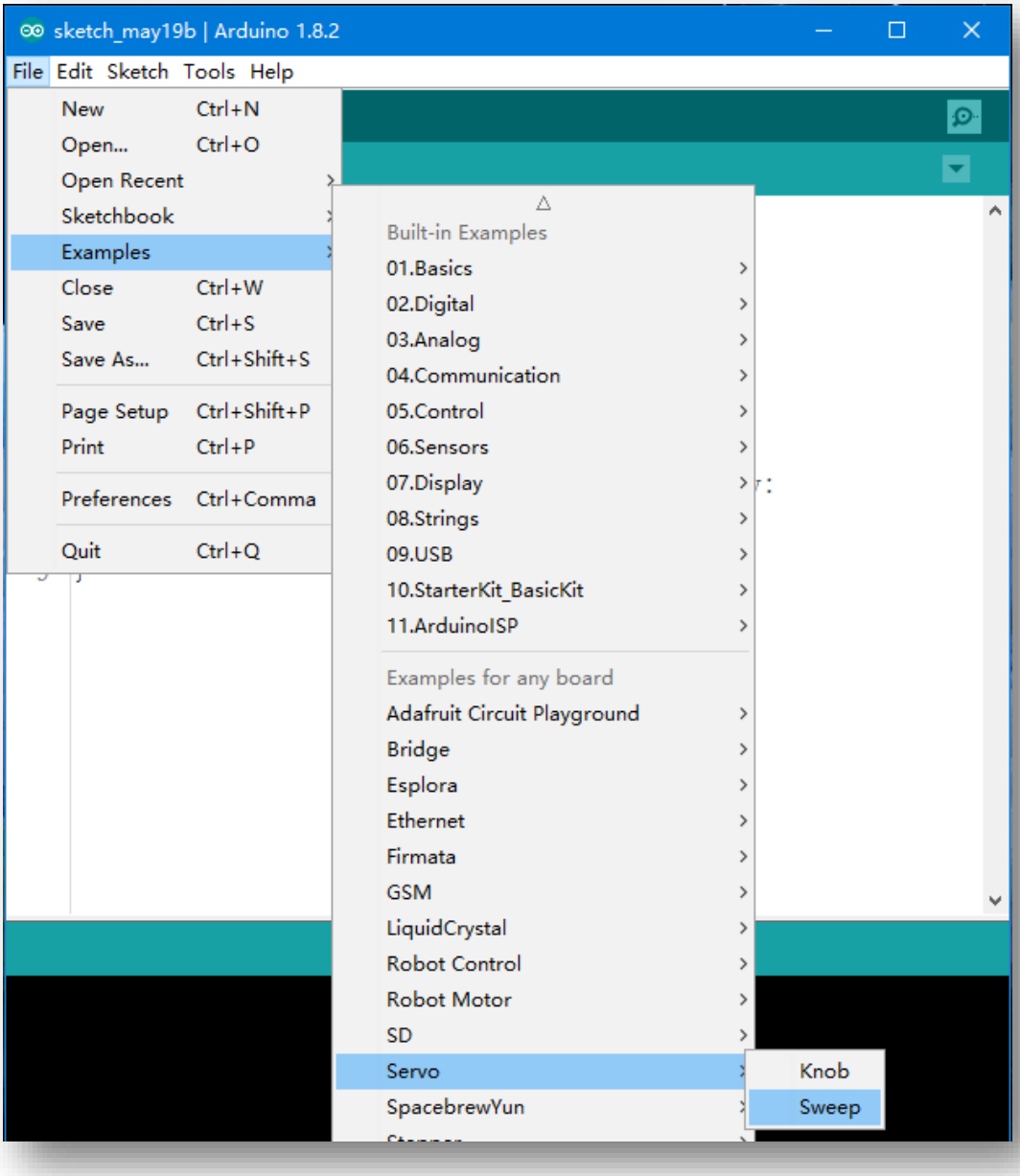
To control the servo rotation, you need to make the time pulse to be about 20ms and the high level pulse width to be about 0.5ms~2.5ms, which is consistent with the angle limited of the servo.

Taking 180 angle servo for example, corresponding control relation is as below:

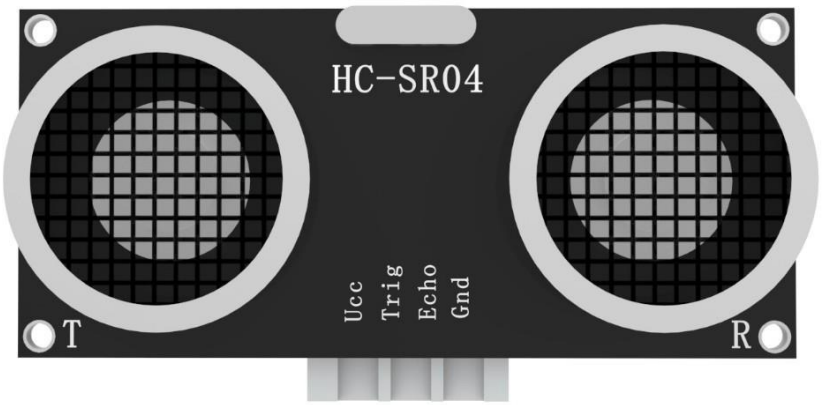
0.5ms	0 degree
1.0ms	45 degree
1.5ms	90 degree
2.0ms	135 degree
2.5ms	180 degree

The example program:

Open Arduino IDE and select "File->Examples->Servo->Sweep"



Next, let's have a look at the ultrasonic sensor module.



Feature of the module: testing distance, high precision module .

Application of the products: robot obstacle avoidance 、obje ct testing distance、liquid testing、public security 、parking lot testing.

Main technical parameters

(1)：voltage used: DC ---5V (2)：

static current: less than 2mA(3)：

level output: higher than 5V(4)：

level output: lower than 0

(5)：detection angle: not bigger than 15 degree(6)：

detecting distance: 2cm -450cm

(7)：high precision: up to 0.2cm

Method of connecting lines: VCC, trig (the end of controlling), echo (the end of receiving), GND

How does the module work:

(1) Apply IO port of TRIG to trigger ranging, give high level signal, at least 10us one time;

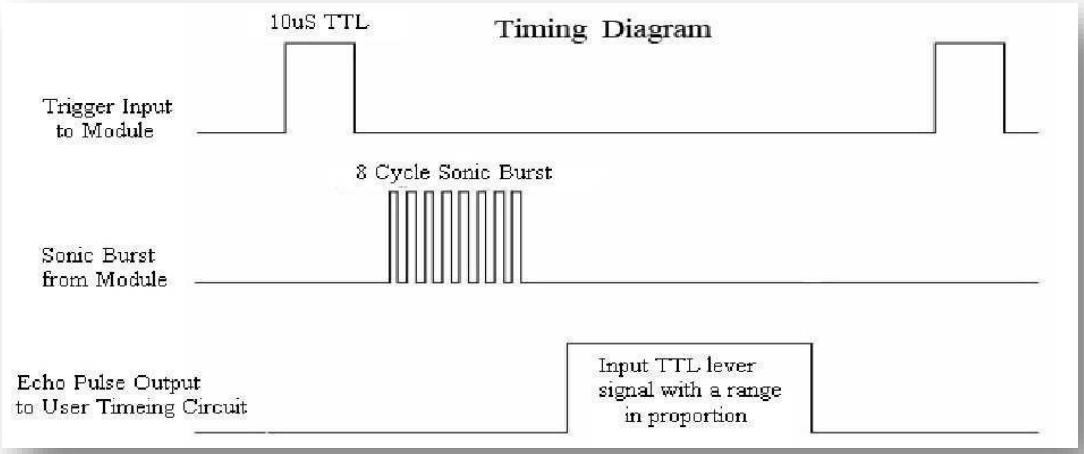
(2) The module sends 8 square waves of 40kz automatically, tests if there are signals returned automatically;

(3) If there are signals received , the module will output a high level pulse through IO port of ECHO, the duration time of high level pulse is the time between the wave sending and receiving. So the module can know the distance according to the time.

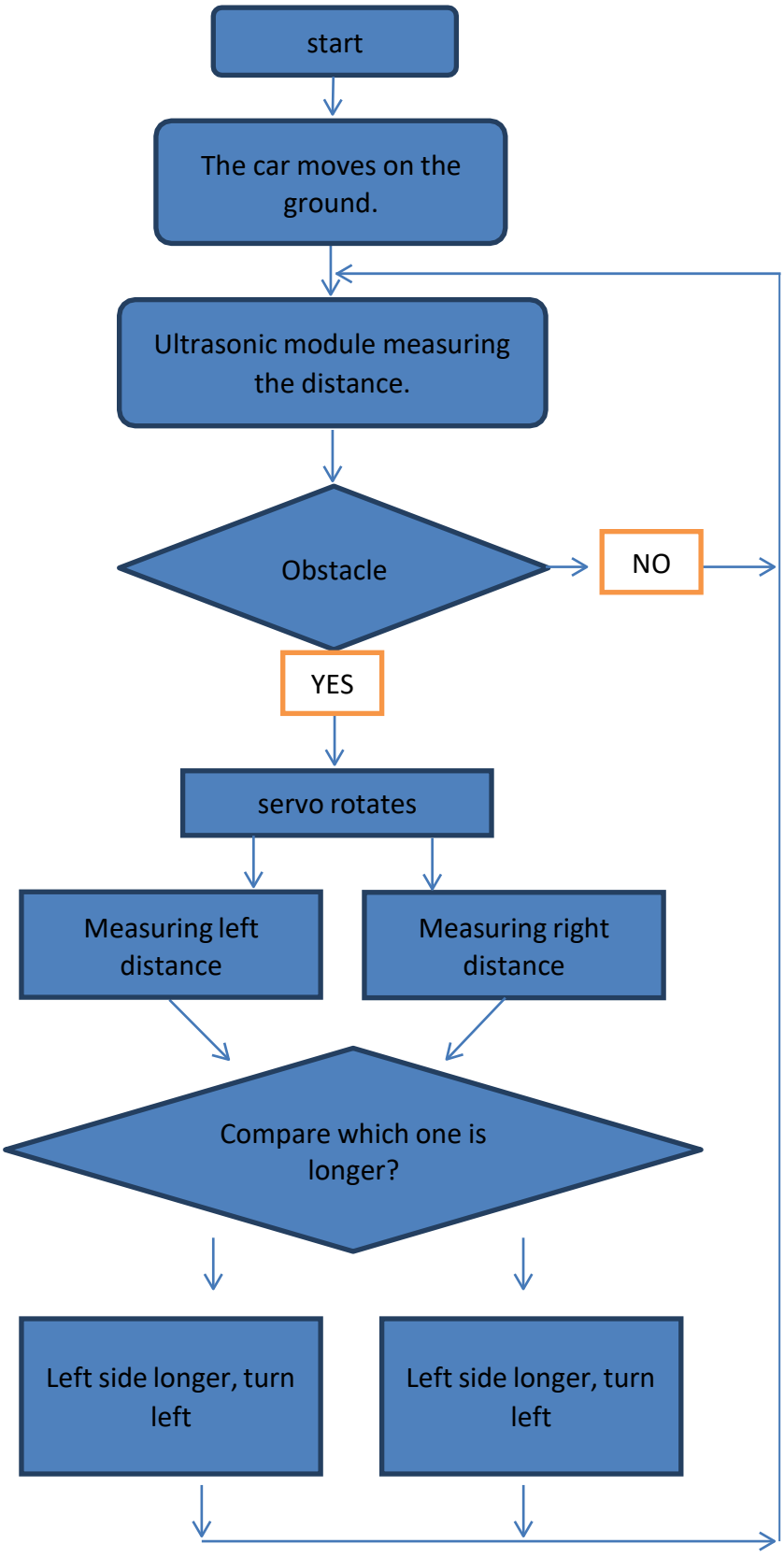
Testing distance= (high level time* velocity of sound (340M/S))/2);

Actual operation:

The Timing diagram is shown below. You only need to supply a short10uS pulse to the trigger input to start the ranging, and then the module will send out an 8 cycle burst of ultrasound at 40 kHz and raise its echo. The Echo is a distance object that is pulse width and the range in proportion .You can calculate the range through the time interval between sending trigger signal and receiving echo signal. Formula: uS / 58 = centimeters or uS / 148 =inch; or: the range = high level time * velocity (340M/S) / 2; we suggest to use over 60ms measurement cycle, in order to prevent trigger signal to the echo signal.




```
/*Ultrasonic distance measurement Sub  function*/
int Distance_test()
{
  digitalWrite(Trig, LOW);
  delayMicroseconds(2);
  digitalWrite(Trig, HIGH);
  delayMicroseconds(20);
  digitalWrite(Trig, LOW);
  float Fdistance = pulseIn(Echo, HIGH);
  Fdistance= Fdistance/58;
  return  (int)Fdistance;
}
```


start
The car moves on the ground.
Ultrasonic module measuring the distance.
Obstacle
NO
YES
servo rotates
Measuring left distance
Measuring right distance
Compare which one is longer?
Left side longer, turn left
Left side longer, turn left

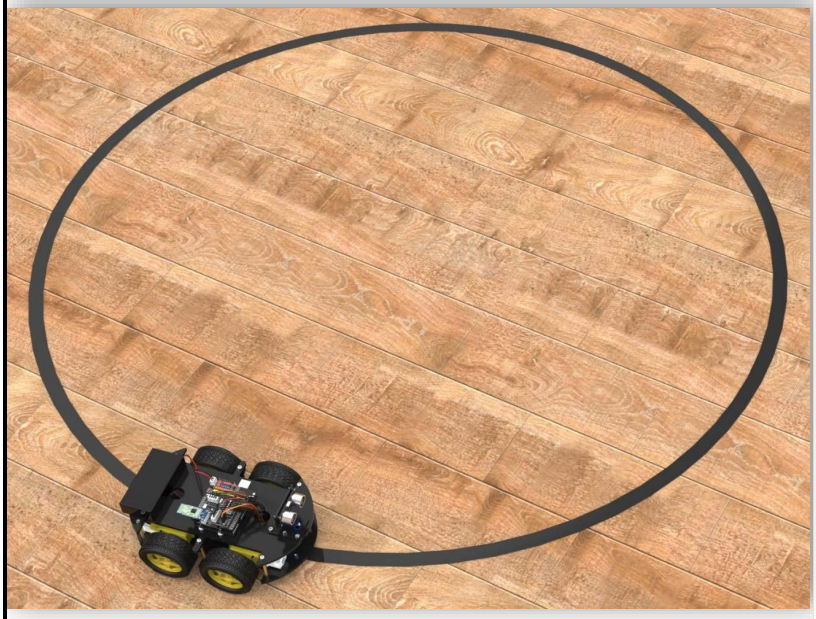
From above picture, we can see that the principle of obstacle avoidance car is very simple. The ultrasonic sensor module will detect the distance between the car and the obstacles again and again and sending the data to the controller board, then the car will stop and rotate the servo to detect the left side and right side. After compared the distance from the different side, the car turn to the side which has longer distance and move forward. Then the ultrasonic sensor module detects the distance again.

Code preview:

```
if(rightDistance > leftDistance) {
  right();
  delay(360);
}
else if(rightDistance < leftDistance) {
  left();
  delay(360);
}
else if((rightDistance <= 20) || (leftDistance <= 20)) {
  back();
  delay(180);
}
else {
  forward();
}
```

Line tracking car

Line tracking car



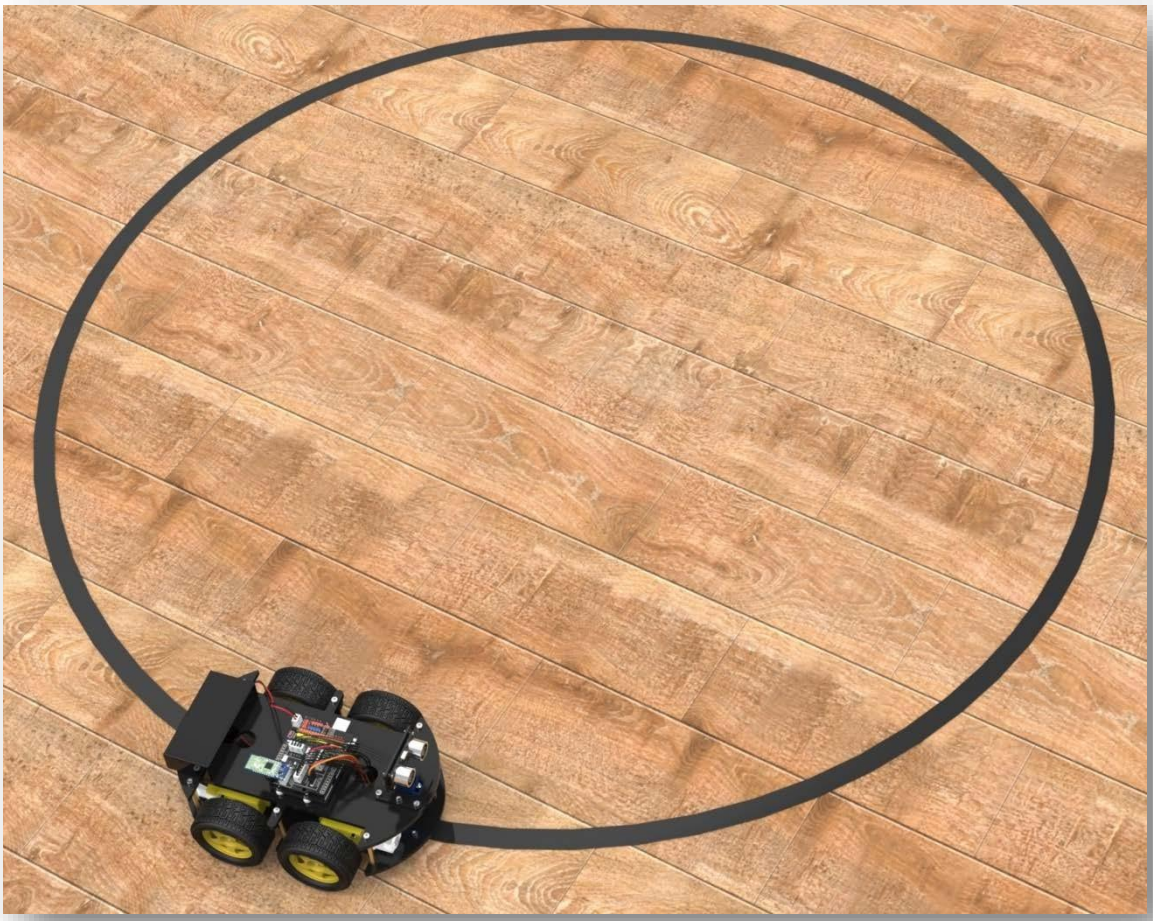
Line tracking car

1. *Points of the section*
2. *In this lesson, we will learn how to control car to move along a runway.Learning parts*：
   a. *Learn how to use the line tracking module Learn the line tracking principles*
   b. *Learn how to implement line tracking via programmingPreparations:*
   c. *A car (equipped with battery) A USB cable*
   d. *Three line tracking modulesA roll of black tape*

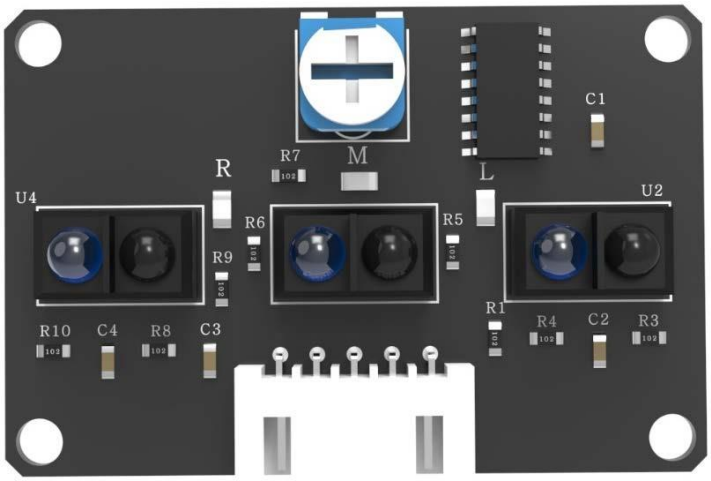
Ⅰ. Making runway

Materials: electrical adhesive tape (black tape)

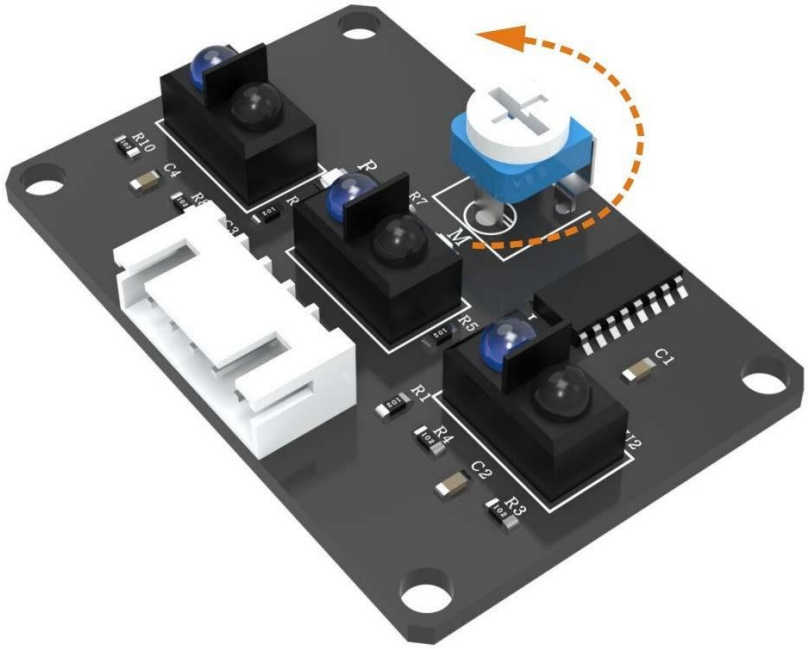
First of all, we need to make a runway on our own. We can make a circuit by pasting black tape on a suitable paper or the ground. Before pasting, you can draw a runway by pen, and then paste with electrical adhesive tape. Pay attention to make the corner as smooth as possible. Because the car will outgo of the line if the angle is too small, but if you want to make it more difficulty, you can make it small. The size of runway is generally not smaller than 40*60 cm.



Ⅱ. Connect modules and debug



The component which is pointed at is potentiometer. It can adjust the sensitivity of the line tracking module by change its resistance value.



Ⅲ. Upload program

After making runway and connecting modules, you just need to open the the code file “\Lesson 5 Line Tracking Car\Line_Tracking_Car\Line_Tracking_Car.ino” and upload the program to the UNO controller board.

Code preview:

```
//Line Tracking IO define
#define LT_R !digitalRead(10)
#define LT_M !digitalRead(4)
#define LT_L !digitalRead(2)

#define ENA 5
#define ENB 6
#define IN1 7
#define IN2 8
#define IN3 9
#define IN4 11

#define carSpeed 150

void forward(){
  analogWrite(ENA, carSpeed);
  analogWrite(ENB, carSpeed);
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
  Serial.println("go forward!");
}

void  back(){
  analogWrite(ENA, carSpeed);
  analogWrite(ENB, carSpeed);
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
  Serial.println("go back!");
```

```
}

void  left(){
  analogWrite(ENA, carSpeed);
  analogWrite(ENB, carSpeed);
  digitalWrite(IN1, LOW);
  digitalWrite(IN2, HIGH);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
  Serial.println("go left!");
}

void  right(){
  analogWrite(ENA, carSpeed);
  analogWrite(ENB, carSpeed);
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
  Serial.println("go right!");
}

void stop(){
   digitalWrite(ENA, LOW);
   digitalWrite(ENB, LOW);
   Serial.println("Stop!");
}

void setup(){
  Serial.begin(9600);
  pinMode(LT_R,INPUT);
  pinMode(LT_M,INPUT);
  pinMode(LT_L,INPUT);
}

void loop() {
  if(LT_M){
    forward();
  }
  else if(LT_R) {
    right();
    while(LT_R);
  }
  else if(LT_L) {
```

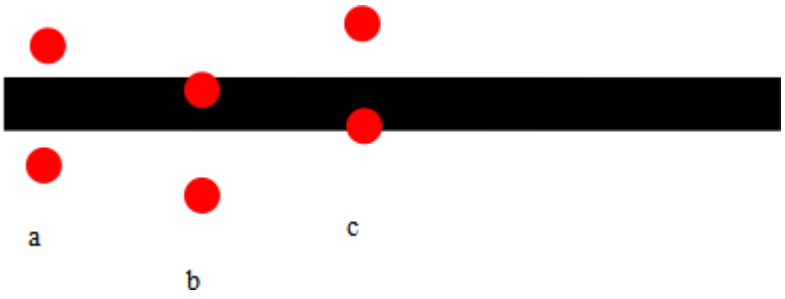
```
    left();
    while(LT_L);
  }
}
```

After disconnecting the car to the computer, you can turn on the power switch and put the car on the runway. Then the car will follow the lines. If you find that it can't move as you expected, please adjust the potentiometer on the line tracking module.

IV. Introduction of principle

Line tracking module

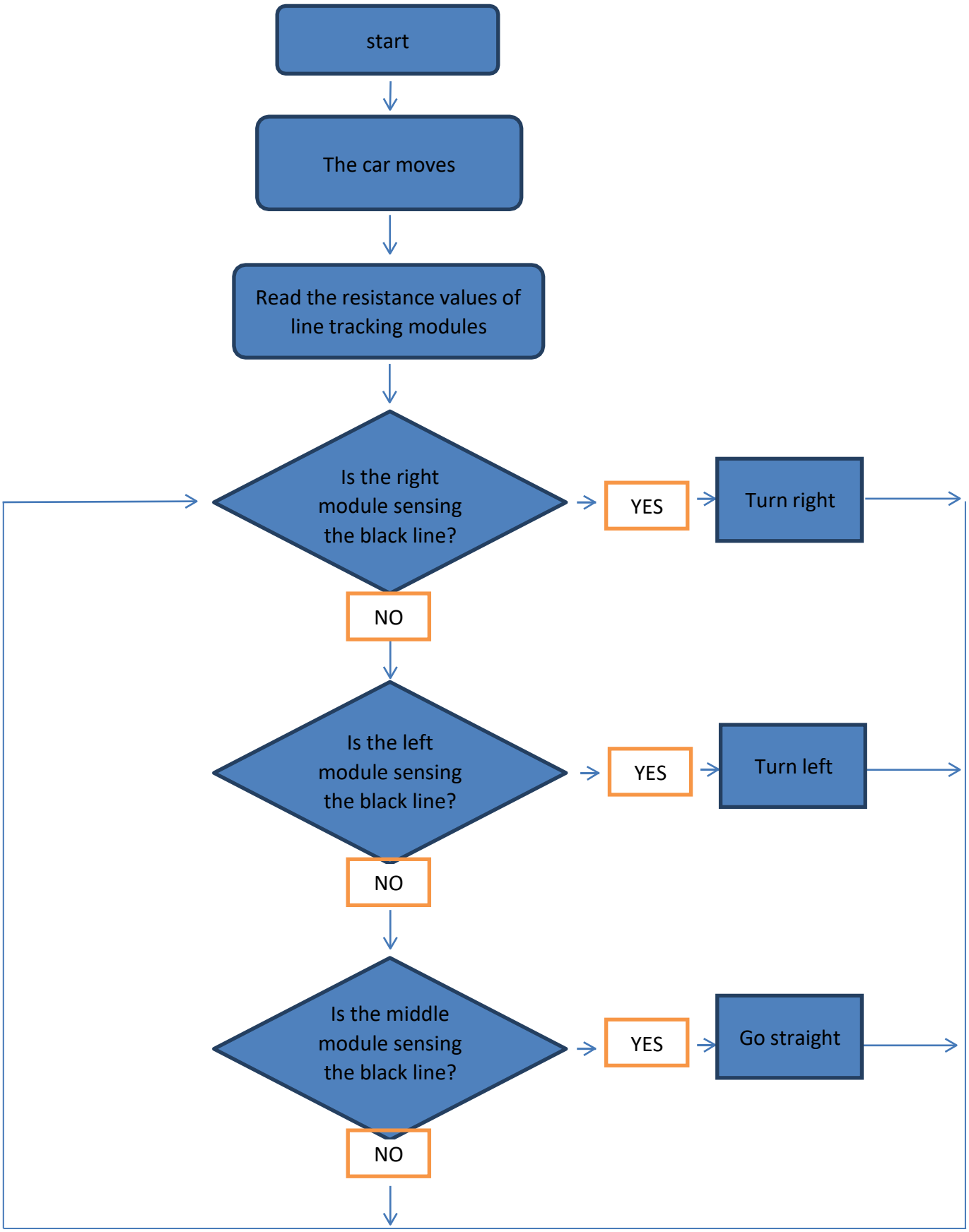
The line tracking sensors are the two components located downward and abreast before the car. The line tracking sensor consists of an infrared transmitter tube and an infrared receiver tube. The former is a LED that can transmit the infrared ray, while the latter is a photoresistor which is only responsible for receiving the infrared light. Light reflectance for the black surface is different from that for the white surface. Hence, the intensity of the reflected infrared light received by the car at the black road differs from that at the white road, and the resistance quantity also changes. According to the principle of voltage division among series resistance, motion path can be determined by inferring the color of road below the car from the voltage of the sensor.



a →T The car move along the black line. One of the line tracking module is on the left side of the line and the other one is on the right side. They can't detect the black line.

b →The car learns to move right. The module on the left side can detect the black line, then it will send signal to the controller board and the car turn left.

c →The car learns to move left. The module on the right side can detect the black line, then it will send signal to the controller board and the car turn right.


start
The car moves
Read the resistance values of line tracking modules
Is the right module sensing the black line?
YES
Turn right
NO
Is the left module sensing the black line?
YES
Turn left
NO
Is the middle module sensing the black line?
YES
Go straight
NO

From above, we can see the principle of the line tracking car. After the car starting, the line tracking module just need to sense the black line on the road surface, make corresponding action according to the program.

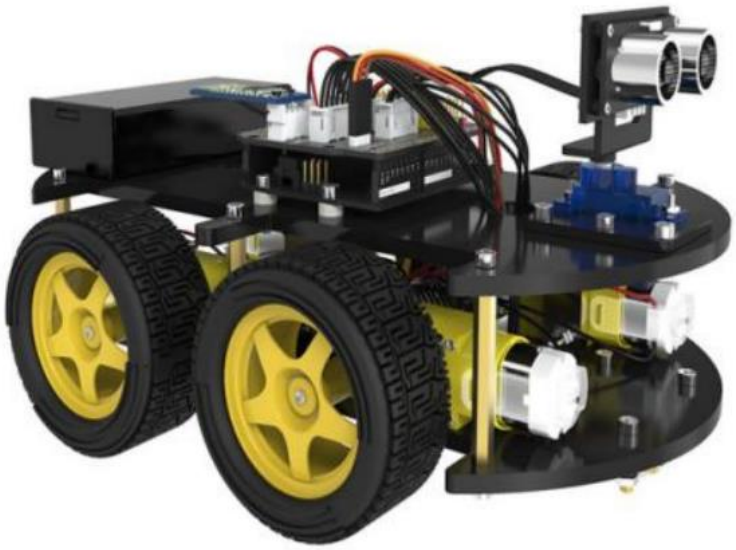
This is a simple algorithm chart of car line tracking program. There are many more complex algorithms such as PID. So after making the function of line tracking come true, you can learn more algorithms of controlling car on your own.

Small tips

(1) The bending part of the line should be as smooth as possible. If the cornering radius is too small, the car is very likely to move beyond the runway.

(2) The line tracking scene can be made with the black and white tape or the paper of any color that be distinguished from the path.

(3) In addition to line tracking, we can stretch imagination to develop other programs based on the line tracking principles, such as those that confine the car within a region regardless of its motion.



Multi-functional Smart Car

Multi-functional Smart Car

Multi-functional Smart Car

---

i. *Points of this Section*

ii. *After learning lesson 0~5, I think you have a deep understanding of the various functions of the robot car. Now, it's time for us to combine*

iii. *all functions together injecting the soul into the robot car, and achieve a more dazzling operation.*

iv. *Learning Objectives:*

a. ◆ *Learn how to combine app with the car to ensurerocker control function by using bluetooth.*

b. ◆ *Learn how to combine app with the car to ensureGraphical Programming by using bluetooth.*

v. *Preparations:*

2. *A vehicle (equipped with battery)A USB cable*

a.

3. *A Bluetooth moduleAn IPhone or tablet*

b.

◆

◆

Ⅰ. Rocker control

STEP1: Upload the program

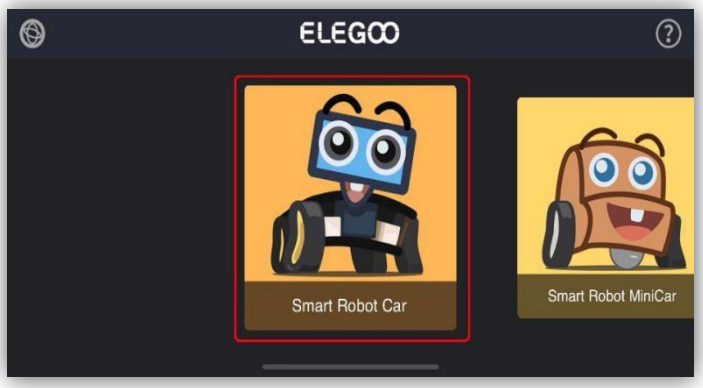
Open the code file in the path "\Elegoo Smart Robot Car Kit V3.0 Plus\Lesson 6 SmartCar_ Multi_function\Rocker_Control" and upload the program to the UNO board.

Disconnect it from the computer, and then switch on the car's power supply.

(TIPS: The Bluetooth module should be pulled out when you upload the program, or it will be failed to upload the program.)

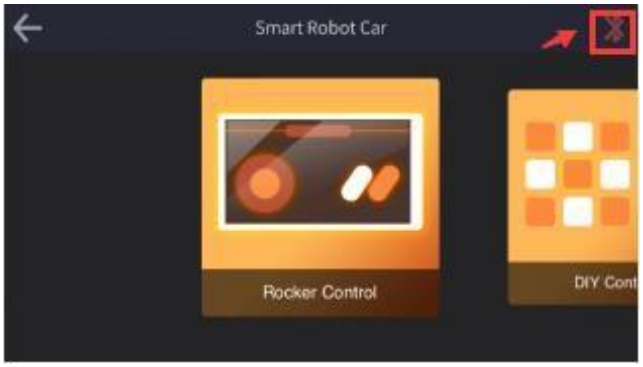
STEP2: Open the"Elegoo BLE Tool" App.

STEP3: Select the "Smart Robot Car".



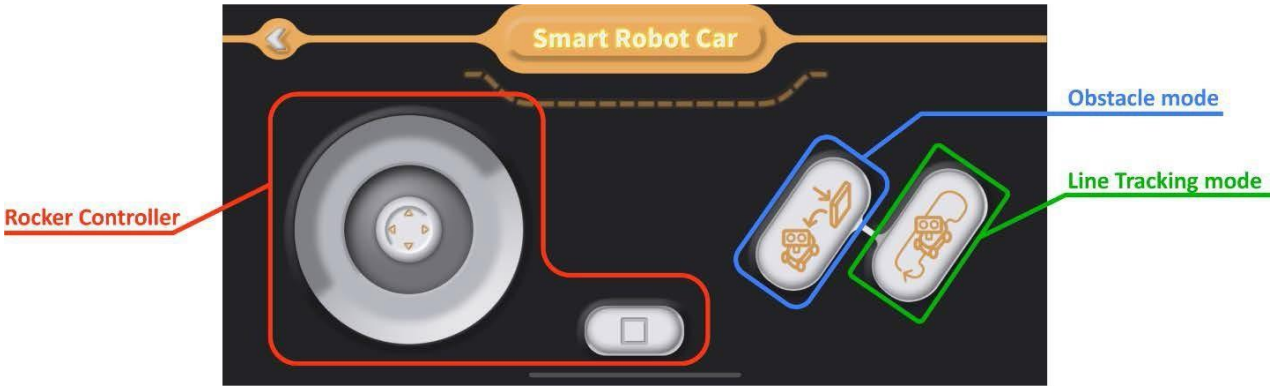
STEP4: Connect Bluetooth

Click Smart Robot Car to enter the control page. Then tap the “  **” icon to enter the Bluetooth searching interface. Refer to Lesson 2 for details.**



STEP5: Please Click "Rocker Control".

STEP6: Introduction of Interface Function



The main functions in the Rocker Control panel are divided into three parts:

Rocker controller: You can freely control the movement of the Smart Car, press the square button to stop the car.

Obstacle mode: The car will turn into the obstacle avoidance mode, which is the same as the function in Lesson4.

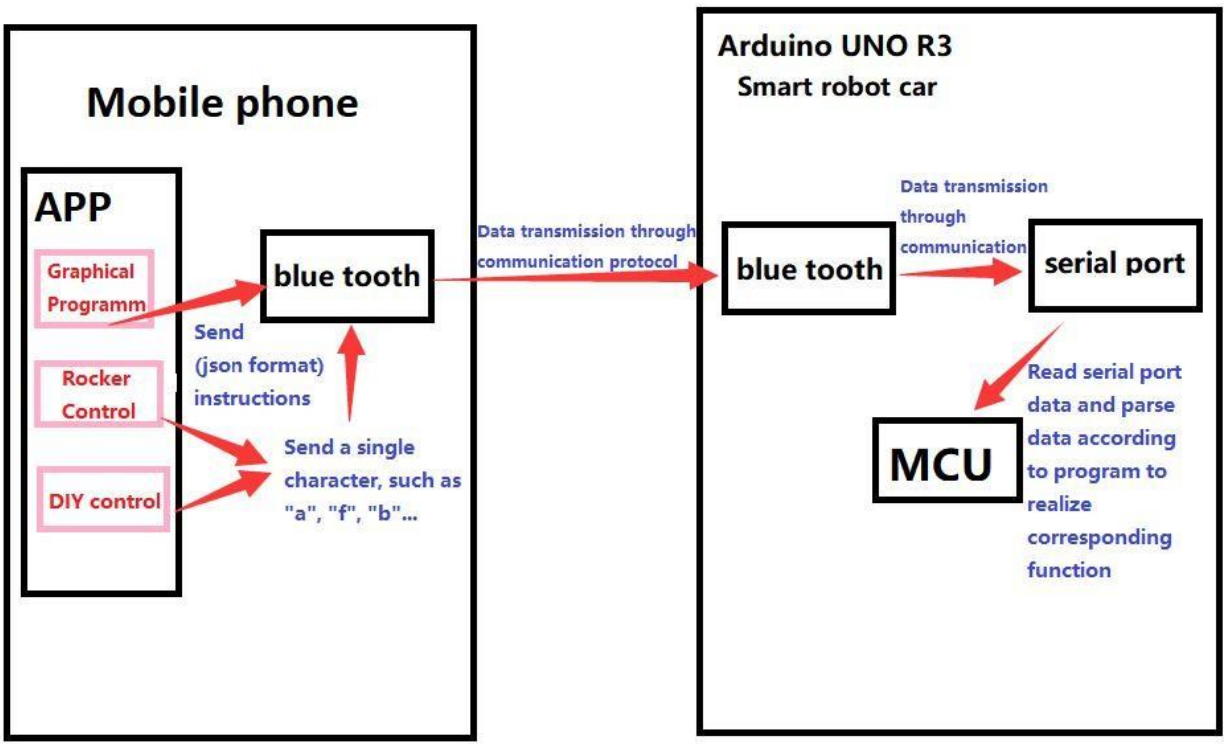
Line tracking mode: The car will turn into the line tracking mode, which is the same as the function in Lesson 5.

Ⅱ. Graphical Programming

Principle:

Every function of graphical programming in APP is realized by sending a string (Json data format) to Bluetooth on Robot Car Development Board through Bluetooth on mobile phone. Then Bluetooth on Robot Car Development Board sends the string to serial port. In the program, the string is removed by reading the data received by serial port. The key content of the implementation function in the fixed format is parsed out, and then the corresponding function is realized if the strings of the implementation function are identical.

The string (Json data format) is roughly formatted as follows: {"N": 2, "D1": 1}



Robotic cart instruction based on graphical programming (Json format)

"Car 3.0 + instruction V2" details are as follows:

Ultrasound module

command	{"N":21, "D1":parameter 1 }
function	Check if an obstacle is detected
return	{false} ： No obstacles were detected {true} ： Obstacles detected {Ultrasound numerical value}
Description of parameters	1： Query whether obstacles are detected 2： Query the Value of the Ultrasound Sensor

Tracing module

command	{"N":22,"D1":0 }
function	Query Trace Sensor for Black Line Detection
return	{false} ： No black line detected {true}： Black line detected
Description of parameters	parameter D1 0 ： left tracking sensor 1 ： Intermediate tracking sensor 2 ： Right tracking sensor

Sport mode

command	{"N":1,"D1":parameter 1,"D2":parameter 2,"D3":parameter 3 }
function	Sets the direction and speed of motor motion
return	{ok}
Description of parameters	parameter 1 （select the corresponding motor） 0 ： All motors 1 ： left front motor 2 ： Right front motor 3 ： left rear motor 4 ： Right rear motor parameter 2 （selected direction of motor rotation） 0 ： stop 1 ： Forward 2 ： Reverse 3 ： no processing parameter 3 (the selected motor speed value)

	Speed value range: 0~255 parameter 4 Duration of motor rotation User input value, 0-20 seconds
command	{“N”:4,”D1”:parameter 1,”D2”:parameter 2,”T”:parameter 4}
function	Sets the direction and speed of motor motion
return	{ok}
Description of parameters	parameter 1 （selected direction of motor rotation） 1：turn left 2：Turn right 3：Advance 4：Back parameter 2 （the selected motor speed value） Speed value range:0~255 parameter 4 Duration of motor rotation User input value, 0-20 seconds

command	{“N”:40,”D1”:parameter 1,”D2”:parameter 2 }
function	Sets the direction and speed of motor motion
return	{ok}
Description of parameters	parameter 1 （selected direction of motor rotation） 1：turn left 2：Turn right 3：Advance 4：Back parameter 2 （the selected motor speed value） Speed value range: 0~255

Clear mode

command	{“N”:5 }
function	clears all functions being executed
return	{ok}
Description of parameters	

Remote switching mode command

command	{“N”:3,”D1”:parameter 1}
function	switch car mode
return	
Description of parameters	parameter 1 1 ： Tracking mode 2 ： obstacle avoidance mode

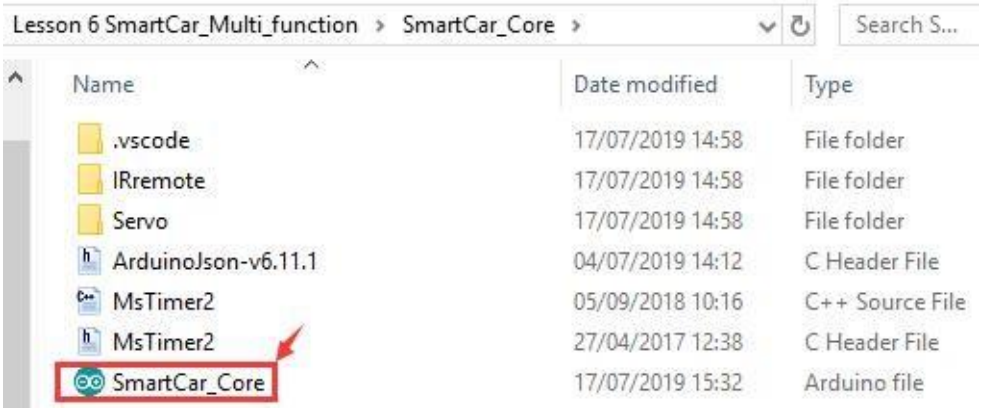
Rocker move command

command	{“N”:2,”D1”:parameter 1}
function	The car moves in a certain direction. The default maximum speed.
return	{ok}
Description of parameters	parameter 1 1：turn left 2：Turn right 3：Advance 4：Back 5：Stop

Ⅲ. Specific Operation:

STEP1: Upload the program

Open the code file in the path “\Elegoo Smart Robot Car Kit V3.0 Plus\Lesson 6 SmartCar_Multi_function\SmartCar_Core” and upload the program to the UNO board.

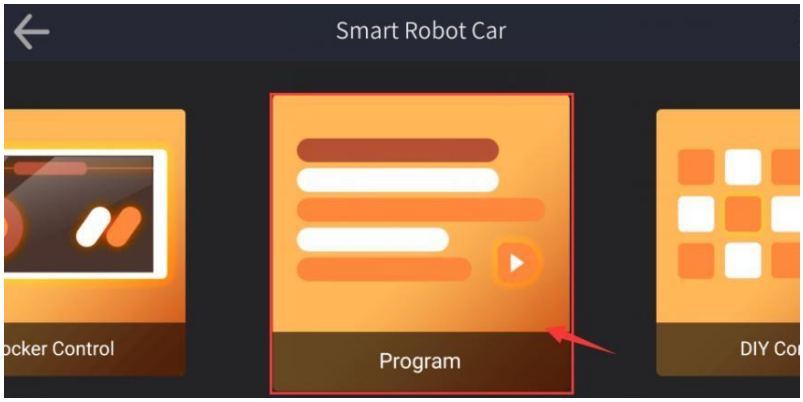


Disconnect it from the computer, and then switch on the car’s power supply.

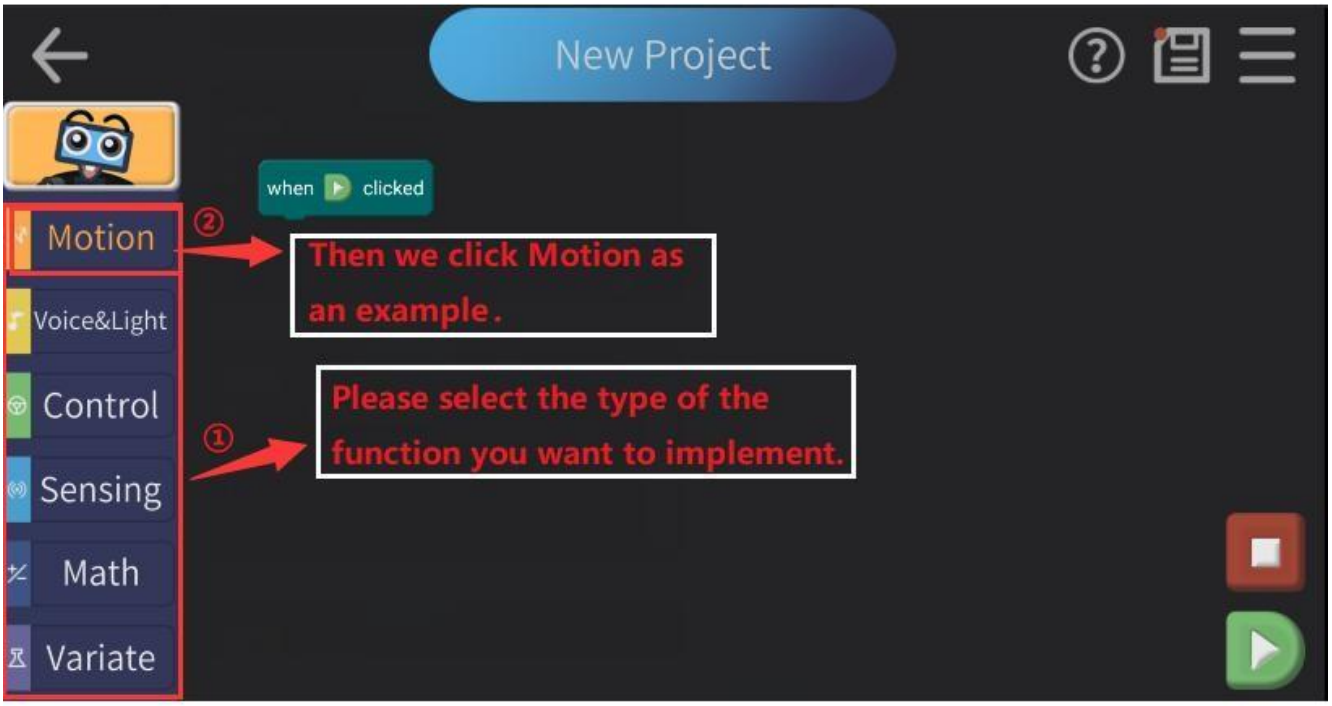
(TIPS: The Bluetooth module should be pulled out when you upload the program, or it will be failed to upload the program.)

STEP2,3,4: The same as the previous chapter.

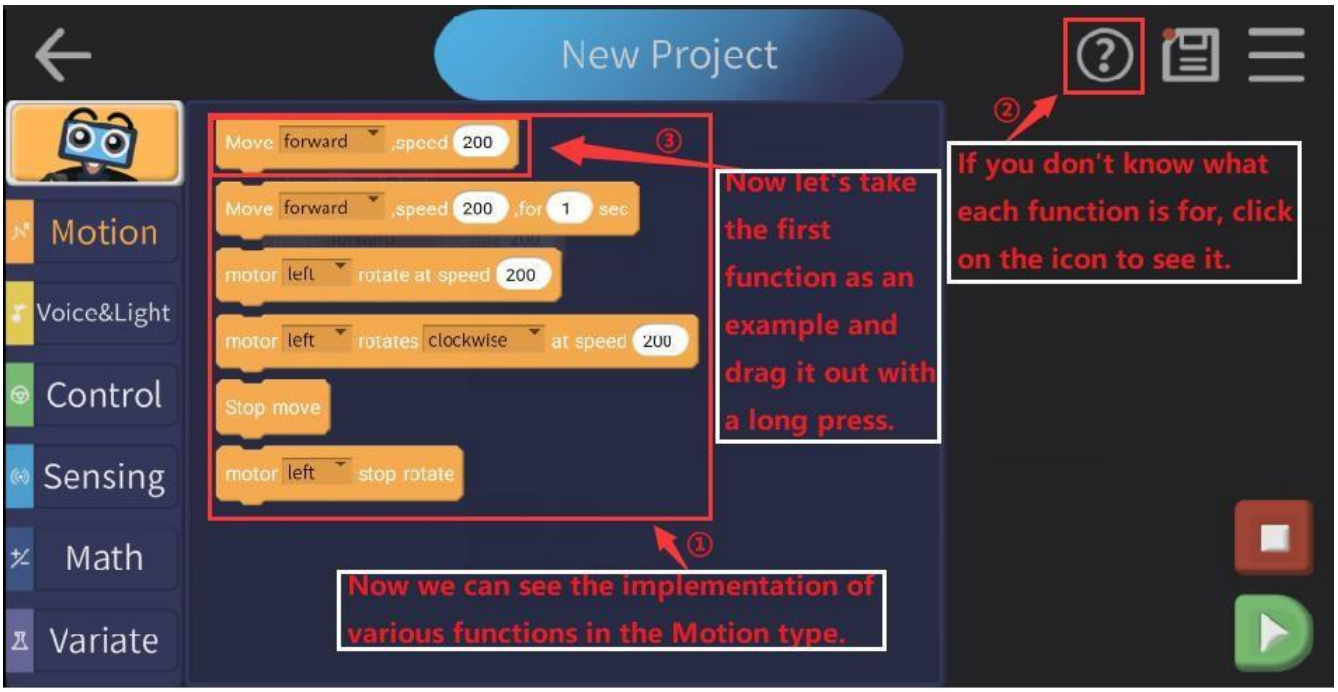
STEP5: Please click "Program".



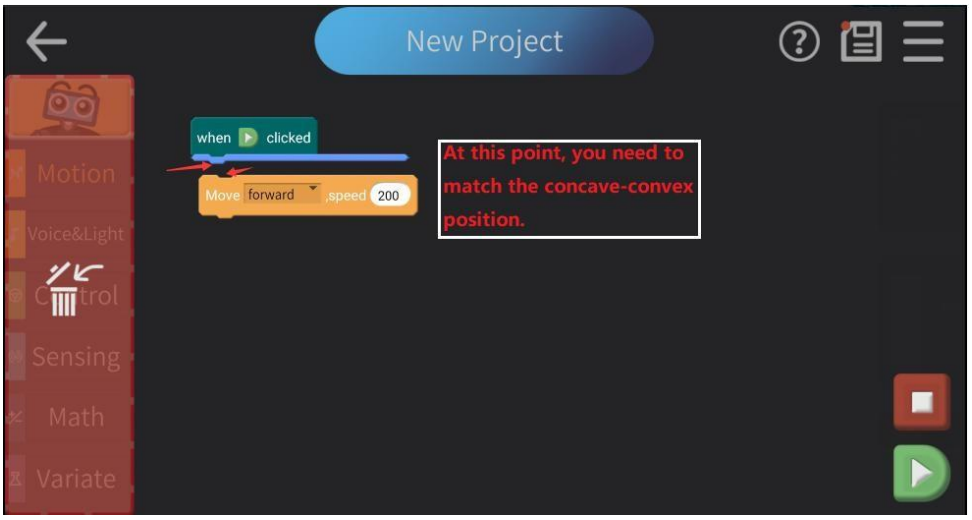
STEP6:

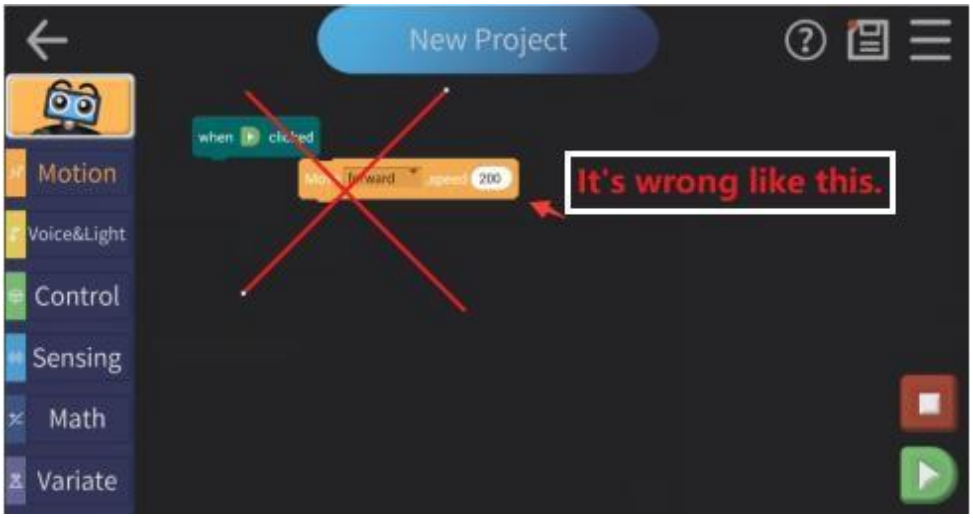

New Project
when clicked
Motion
Voice&Light
Control
Sensing
Math
Variate
②
Then we click Motion as an example.
①
Please select the type of the function you want to implement.

STEP7:


New Project
Move forward ,speed 200
Move forward ,speed 200 ,for 1 sec
motor left rotate at speed 200
motor left rotates clockwise at speed 200
Stop move
motor left stop rotate
Motion
Voice&Light
Control
Sensing
Math
Variate
③
Now let's take the first function as an example and drag it out with a long press.
②
If you don't know what each function is for, click on the icon to see it.
①
Now we can see the implementation of various functions in the Motion type.

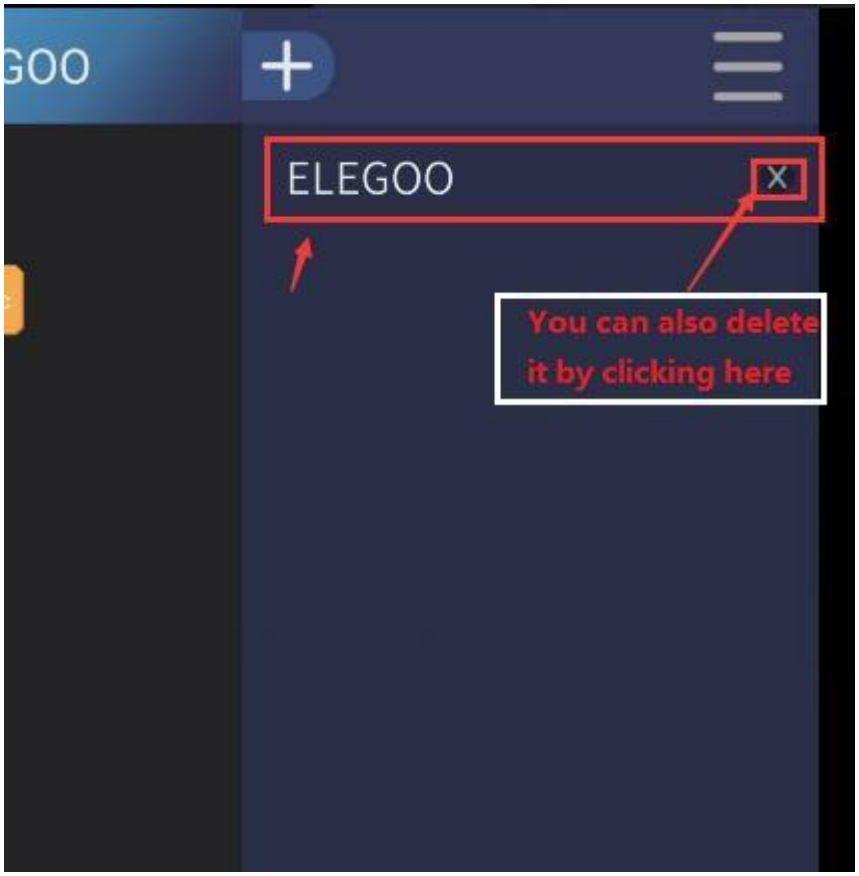
STEP8:


New Project
when clicked
Move forward ,speed 200
At this point, you need to match the concave-convex position.
Motion
Voice&Light
Control
Sensing
Math
Variate


New Project
when clicked
Move forward ,speed 200
It's wrong like this.
Motion
Voice&Light
Control
Sensing
Math
Variate


New Project
④
Then we can give our programming content a project name. Let me take ELEGOO as an example.
⑤
Finally, click the icon to save it
⑥
We can click on this icon to see the project we created.
when clicked
Move forward ,speed 200
motor left rotates clockwise at speed 200
① it is the right way
③
You can also click the red button to stop execution.
②
now you can click the green button,and you will see the car see the car execute your instructions in sequence.
Motion
Voice&Light
Control
Sensing
Math
Variate

STEP9: Finally, we can see the "ELEGOO".



REFERENCE

- https://www.scribd.com/books
- https://www.scribd.com/document/449237250/Hummer-Bot-Instruction-Manual
- https://create.arduino.cc/projecthub/samanfern/bluetooth-controlled-car-d5d9ca

Thank you

1. 20BES7010
2. 20BCD7133
3. 20BCD7115
4. 20BCE7379
5. 20BCR7104
6. 20BCD7096