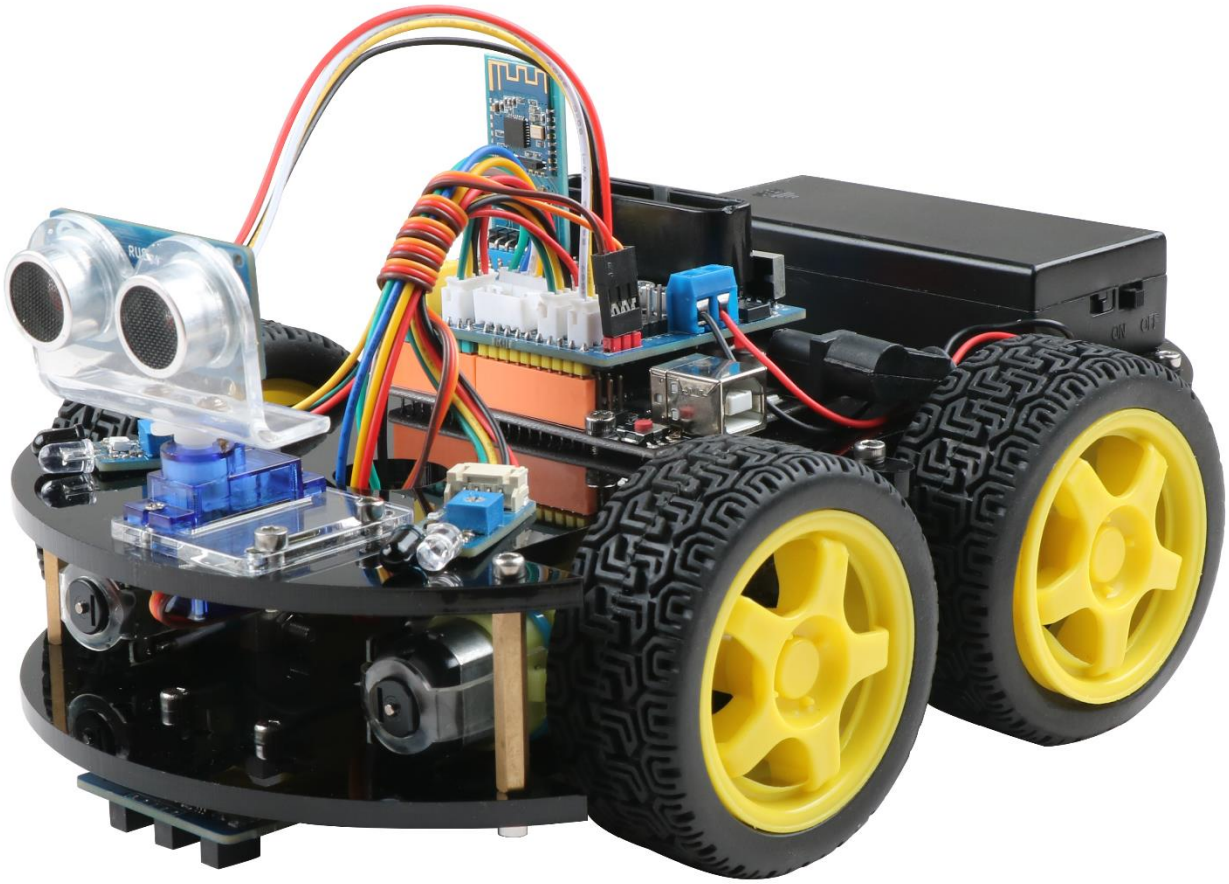
Hummer-bot3.0

Instruction Manual

V.1.3



Github : https://github.com/emakefun/hummer-bot/

Revised Version Of History

Date	Version	Description	Author
2019/2/21	V.1.0	Create	Carl. Du
2019/3/25	V.1.1	Check the file and specification folder	Abbott. Chen
2019/4/10	V.1.2	Motor installation method modification	Carl. Du
2019/5/18	V.1.3	Modify L298N drive mode, and driver	Carl.Du

Table Of Contents

Chapter1 Introduction

1.1 Writing Purpose

The purpose of this manual is to create a fast, practical and convenient development learning platform for the vast number of electronic enthusiasts and let them grasp the Arduino and its extended system design methods and design principles, as well as the corresponding hardware debugging methods.

This manual will lead you to learn every function of "Hummer-Bot" step by step and open a new "Hummer-Bot" journey for you. It is divided into two parts:

1, Preparation chapter, which mainly introduces the use of common Arduino development software and some downloading and debugging skills.

2, Experiment chapter, which contains hardware and software, the former mainly introduces the function and principle of each module; the latter mainly introduces each part of the program and leads you to understand and grasp the principle of Arduino and the car development through written examples step by step.

This manual is a specifications for "Hummer-Bot" , the file whose format is PDF which is in the CD along with our product requires the corresponding software to open. It contains detailed schematic diagrams and complete source codes for all instances, the codes won't have any mistake under our strict test. In addition, the library files used in the source codes are put into the corresponding path, you only need to see corresponding phenomenon of the car and personally experience the process of experiment by downloading the source codes to Arduino via the serial port emulator.

1.2 Product Introduction

"Hummer-Bot" is a multifunctional car based on the Arduino UNO and L298N motor. Compared with the traditional car, "Hummer-Bot" is also equipped with wireless control (Bluetooth, infrared, WIFI and so on); ultrasonic; infrared. It can trace and avoid obstacles automatically, of course, makers can also automatically control the car with wireless and make full use of each module, as well as integrate all kinds of related sensors to make the car more intelligent, which is more challenging. "Hummer-Bot" has various types of information, technical manuals, routines, etc., which can teach you step by step. Each electronic fan can use it easily to achieve their desired function.

Product Features

- Three groups of black line infrared tracing module
- Two groups of infrared obstacle avoidance module
- Ultrasonic obstacle avoidance
- Four DC motor drive
- Two 3000mZh, 3.7V rechargeable lithium battery with longer endurance
- Remaining capacity of battery real-time detection
- Infrared remote control
- Bluetooth app control
- PS2 handle control (optional)
- Support NRF24L01+ wireless control（optional）

1.3Product Device List

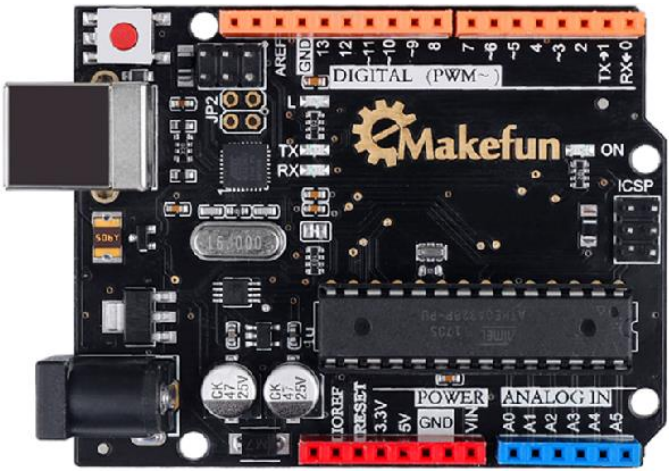
 2 x smart car Board	 1 x CD-ROM with tutorial	 4 x DC geared motor	 1 x USB Cable	 1 x IR remote control
 1 x Emakefun UNO R3 MainBoard	 1 x Arduino extension board	 1 x JDY-16 Bluetooth module	 1 x Ultrasound Module	 2 x Infrared obstacle avoidance module
 1 x L298N motor drive board	 1 x Infrared line tracking module	 1 x 18650 li-battery box	 4 x 65mm wheel core	 1 x 18650 battery charger
 1 x SG90 servo	 1 x Power indicator digital tube	1 x SG90 Servo fixed plate	1 x Ultrasonic Holder	4 x Motor aluminum alloy holder
 1 x 3.0-7.5mm Phillips screwdriver	 1 x Cross wrench	 6 x M3*30mm Dual channel copper posts	 2 x M3*5+6mm Single channel copper pillar	 12 x M3 *12mm 22 x M3 *10mm 2 x M3 *8mm Socket head cap screws
 28 x M3-Hex nuts 2 x M2-Hex nuts	 8 x M3*30MM Crosshead screws	 4 x M3*8MM 2 x M2*12MM Crosshead screws	Several Cable Tie	

Chapter2 Preparations

2.1 On Arduino UNO R3 Master Control Board and Extension Board

2.1.1 Arduino UNO R3 Master Control Board

In " Hummer-Bot ", we used the Arduino uno r3 as the main control board, which has 14 digital input/output pins (6 of which can be used as PWM output), 6 analog inputs, and a 16 MHz ceramic resonator, 1 USB connection, 1 power socket, 1 ICSP head and 1 reset button. It contains everything that supports the microcontroller; You just need to connect it to a computer via a USB cable or start with an AC-DC adapter or battery.



Technical specifications:

- Working voltage: 5V
- Input voltage: USB powered or external 7V~12V DC input
- Output voltage: 5V DC output and 3.3V DC output and external power input
- Microprocessor: ATmega328 (Chip data sheet is in the documentation)
- Bootloader： Arduino Uno
- Clock frequency: 16 MHz
- Support USB interface protocol and power supply (without external power supply)
- Support ISP download function
- Digital I/O port: 14 (4 PWM output ports)
- Analog input port: 6
- DC Current I/O Port: 40mA

- DC Current 3.3V Port: 50mA
- Flash memory: 32 KB (ATmega328) (0.5 KB for bootloader)
- SRAM ： 2 KB (ATmega328)
- EEPROM： 1 KB (ATmega328)
- Size：75x55x15mm

2.1.2 Arduino UNO R3 Extension Board Interface Diagram

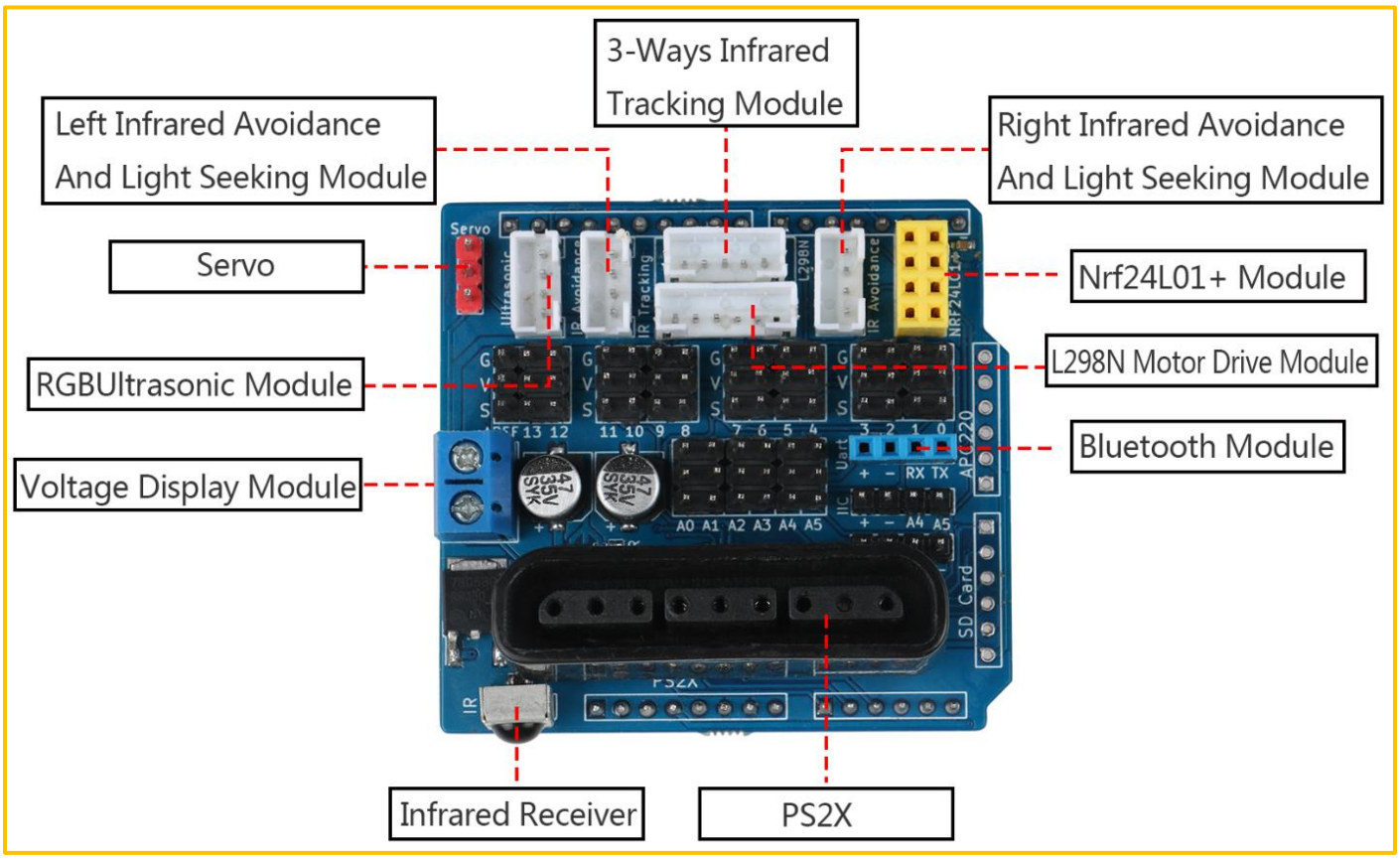


Figure 2.1.2 Expansion Board Interface Diagram

2.2 Development environment **Arduino IDE**

2.2.1 Install the IDE

AduinoIDE is an open source software and hardware tool written by open source software such as Java, Processing, and avr-gcc. It is an integrated development environment that runs on a computer. It can write and transfer programs to the board. The major feature of the IDE is cross-platform compatibility for Windows, MaxOSX, and Linux. Only a simple code base is needed, and the creators can create personalized home internet solutions through the platform, such as remote home monitoring and constant temperature control and so on.

In this tutorial, we use the version is 1.6.0, download address is : https://www.arduino.cc/en/Main/OldSoftwareReleases#previous， After opening the link, we can see the interface as shown in Figure 2.1.1. In this interface, we can see the different versions of the IDE and different operating environments. Everyone can download according to their own computer system, of course, There will be a downloaded installation package on our companion CD, but only the Windows version, because this tutorial is all running under Windows system.

1.8.1	Windows Windows Installer	MAC OS X	Linux 32 Bit Linux 64 Bit Linux ARM	Source code on Github
1.8.0	Windows Windows Installer	MAC OS X	Linux 32 Bit Linux 64 Bit Linux ARM	Source code on Github
1.6.13	Windows Windows Installer	MAC OS X	Linux 32 Bit Linux 64 Bit Linux ARM	Source code on Github
1.6.12	Windows Windows Installer	MAC OS X	Linux 32 Bit Linux 64 Bit Linux ARM	Source code on Github
1.6.11	Windows Windows Installer	MAC OS X	Linux 32 Bit Linux 64 Bit Linux ARM	Source code on Github
1.6.10	Windows Windows Installer	MAC OS X	Linux 32 Bit Linux 64 Bit Linux ARM	Source code on Github
1.6.9	Windows Windows Installer	MAC OS X	Linux 32 Bit Linux 64 Bit Linux ARM	Source code on Github
1.6.8	Windows Windows Installer	MAC OS X	Linux 32 Bit Linux 64 Bit	Source code on Github
1.6.7	Windows Windows Installer	MAC OS X	Linux 32 Bit Linux 64 Bit	Source code on Github

Figure 2.1.1 ArduinoIDE download interface

After the downloading, we will get a compressed package as shown in Figure 2.1.2. The compressed package will be decompressed. After decompression, the files in Figure 2.1.3 are extracted. The “drivers” is the driver software. When the “Arduino.exe” is installed, it will be Install the driver automatically. Because the installation of "arduino.exe" is very simple, it will not be explained here. It is recommended to exit the anti-virus software during the installation process, otherwise it may affect the installation of the IDE. After the installation is complete, click "arduino.exe" again to enter the IDE programming interface.

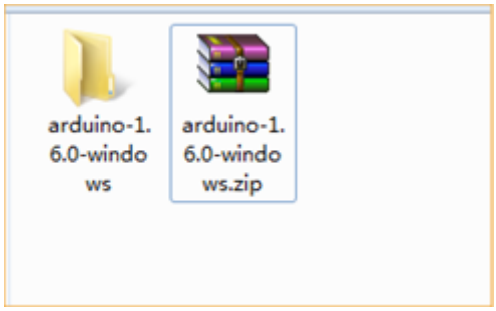


Figure 2.1.2 Arduino IDE Installation Package

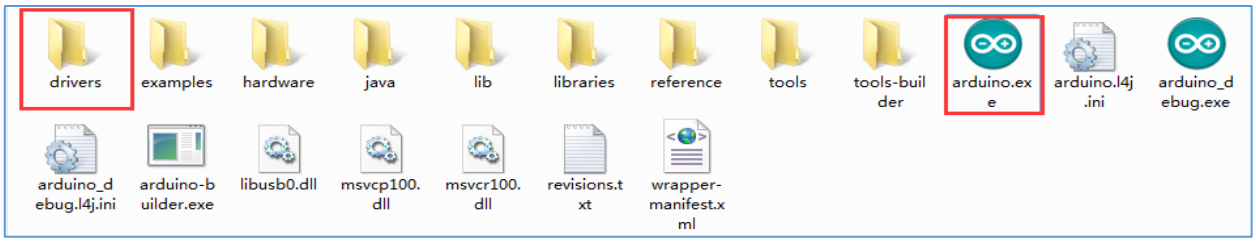


Figure 2.1.3 Extracted files

When finish the installation of the IDE，connect to the Arduino motherboard，click “My Computer” → “Properties” → “Device Manager” → “Viewing Ports (COM and LTP)”，If you can see as the Figure 2.1.4

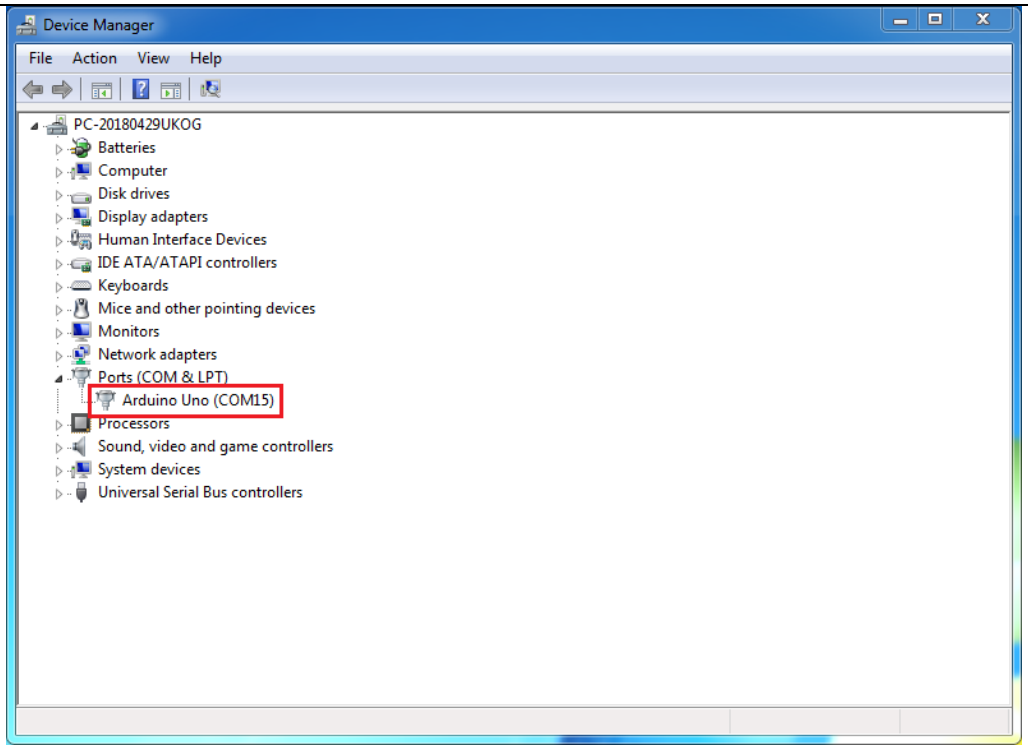


Figure 2.1.4 Driver installation success interface

that indicates the driver has been installed successfully， At this time we open the IDE, select the corresponding development board model and port in the toolbar to use normally. If you see Figure 2.1.5, it means that the computer does not recognize the development board and you need to install the driver yourself.

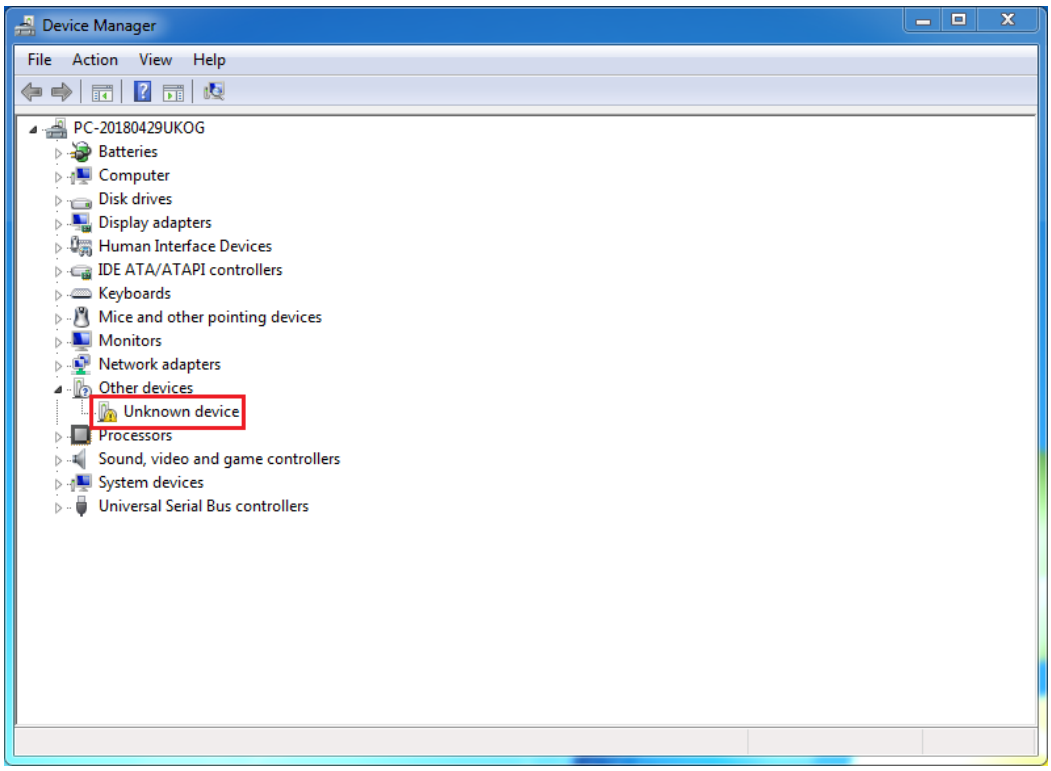


Figure 2.1.5 Driver is not successfully installed interface

Notice:

1）If you connect the controller board to the computer, the computer does not respond. Right-click "My Computer" and select Open Device Manager then find viewing port (com & lpt). If there is no com or lpt, or only an unknown device, there is a problem with the controller board or the USB cable.

2）Right-click "My Computer" and select Device Manager, find the viewing ports (COM and LPT). If there is a yellow Arduino UNO exclamation point, this means you need to install the driver yourself.

3）If you install the driver again and again, it eventually fails. Please uninstall the driver and re-install> install the driver automatically> restart the computer.

2.2.2 Install Driver

If your computer is a Windows 7 system

1）Right-click on "My Computer" and open the Device Manager, find viewing the ports (COM and LPT). At this point you will see a "USB Serial Port", right-click "USB Serial Port" and select the "Update Driver Software" option.

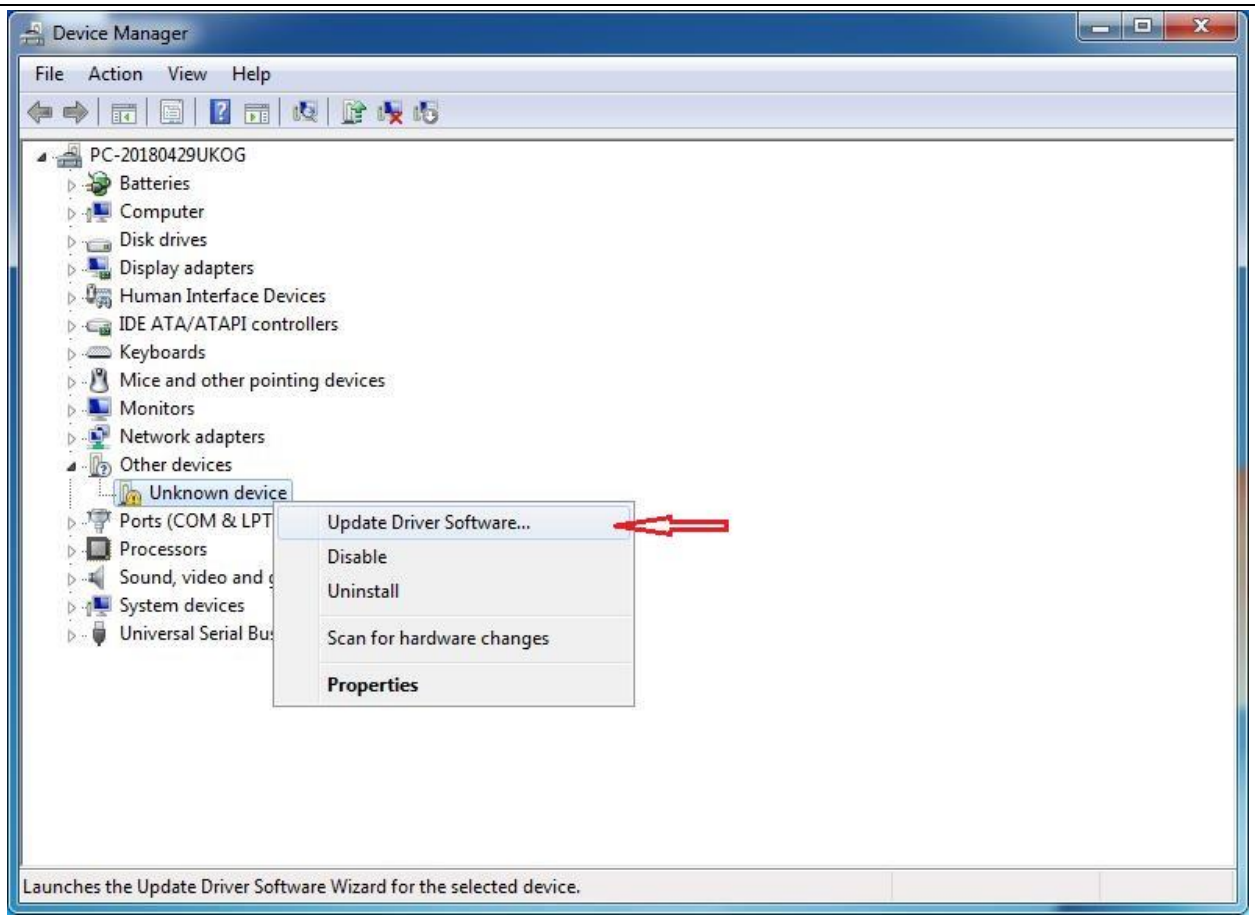


Figure 2.1.6 Updated Driver Interface

2）Next, select the "Browse my computer for driver software" option.

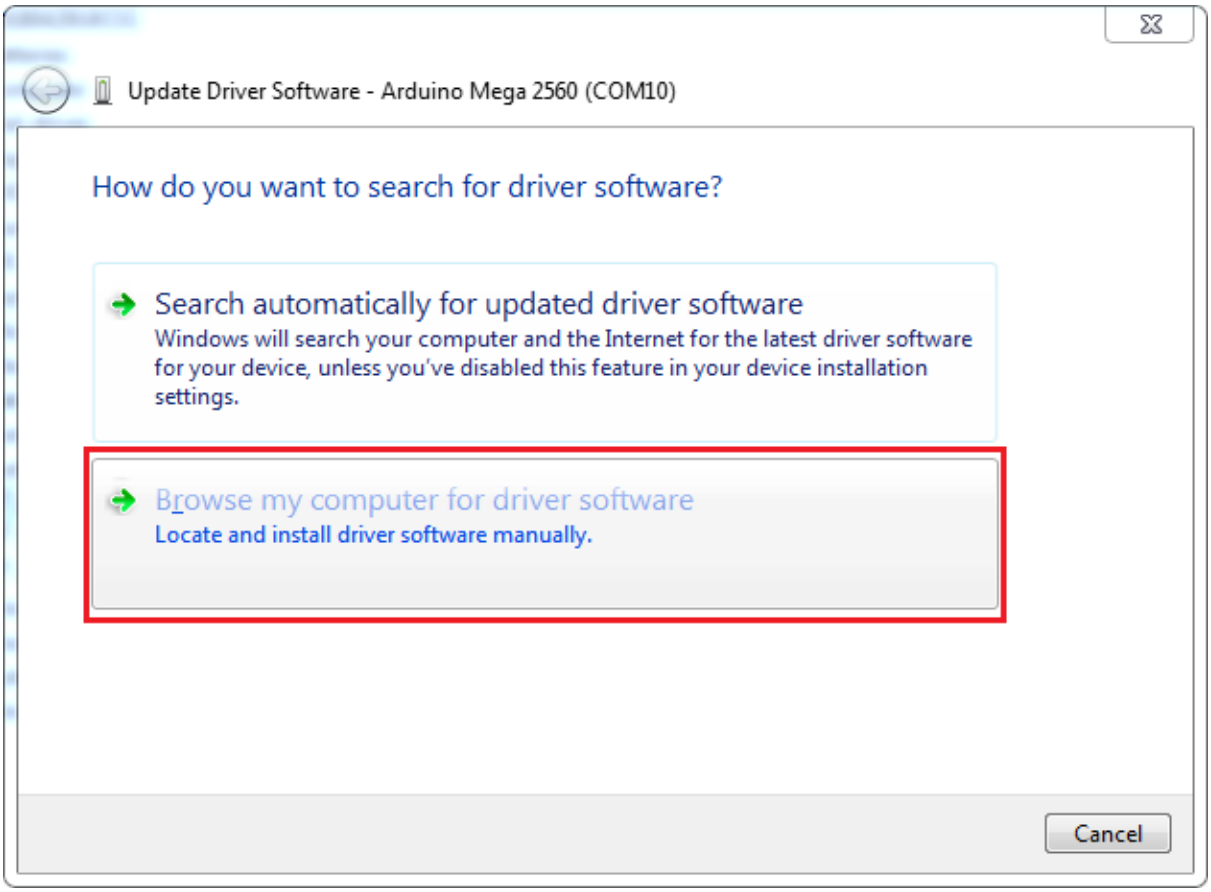


Figure 2.1.7 Driver Update Selection Screen

3）Finally select the driver file named "FTDI USB Drivers" located in the "Drivers" folder of the Arduino software download.

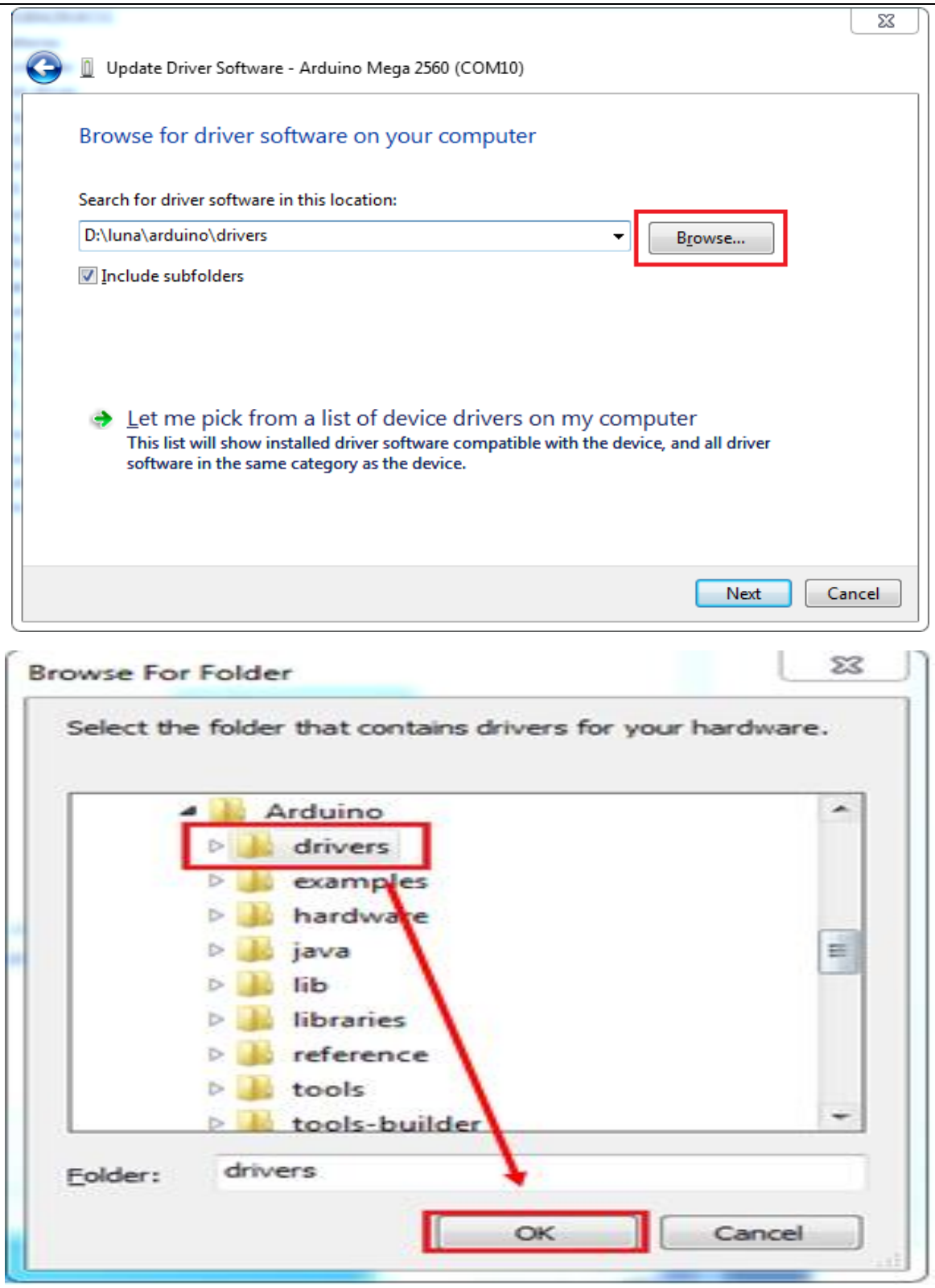


Figure 2.1.8 Driver file selection interface

4）If you have already installed, the following figure will automatically inform you that the driver was successful.

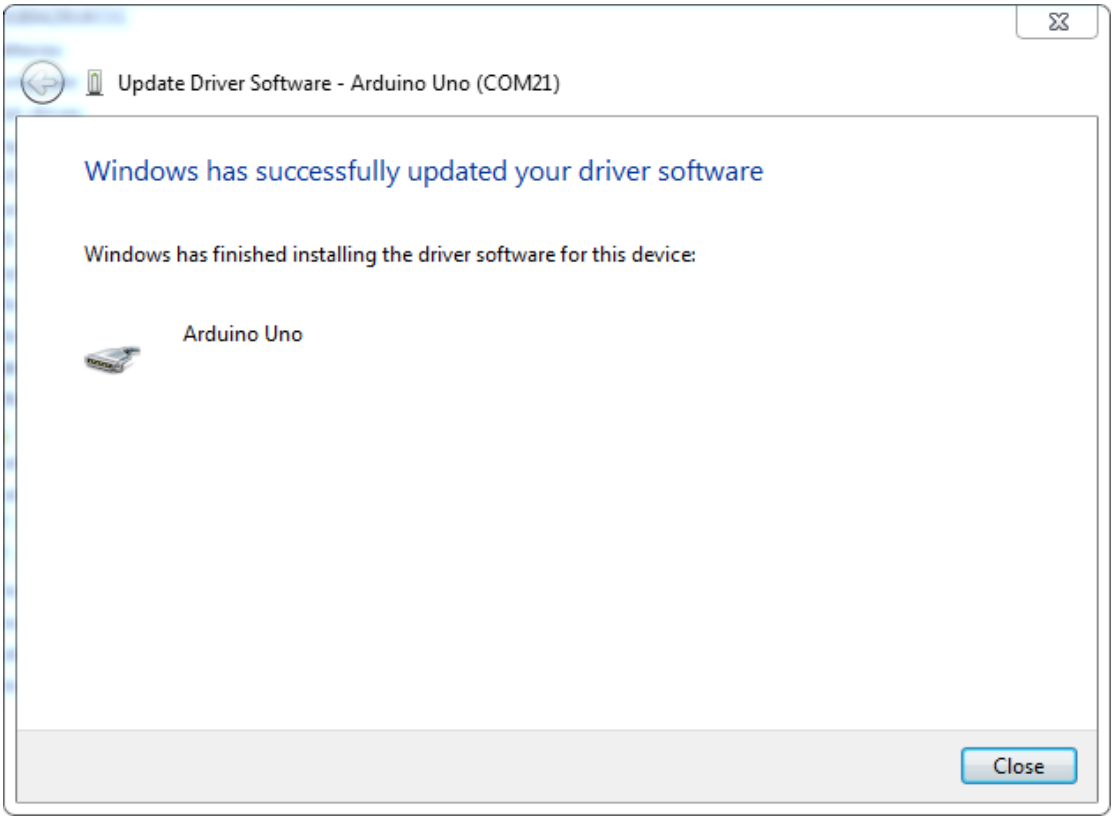


Figure 2.1.9 Driver Installation Successful Interface

At this time, we return to the "Device Manager" interface, the computer has successfully identified Arduino, as shown in the below Figure 2.1.10 .then open the Arduino compilation environment, you can open the Arduino trip.

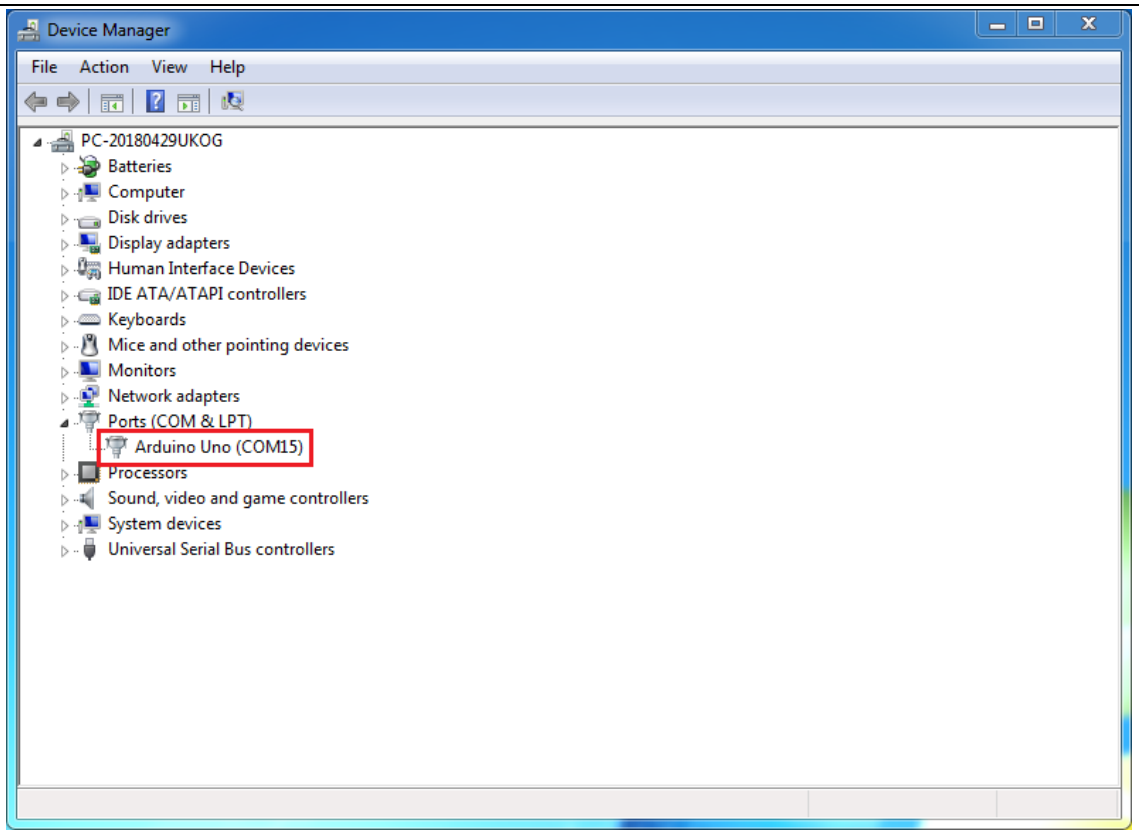


Figure 2.1.10 Driver Success Recognition Interface

Notice In Win10 system, some Arduino are connected to the computer (non-genuine chips are difficult to identify), the system will automatically download the corresponding driver, so you can not install the driver yourself, but in the Win7 system, you have to do it yourself.

In addition, we can see that the USB serial port is identified as COM15 in the above figure, but it may be different with different computer, you may be COM4, COM5, etc., but USB-SERIAL CH340, this must be the same. If you do not find the USB serial port, you may have installed it incorrectly or the system is incompatible.

2、If your computer is a Windows 8 system: Before installing the driver, you should save the files you are editing because there will be several shutdowns during the operation.

1）Press "Windows key" + "R"

2）Iput shutdown.exe / r / o / f / t 00

3）Click the "OK" button.

4）The system will reboot to the "Select an option" screen

5）Select "Troubleshooting" from the "Select an option" screen

6）Select "Advanced Options" from the "Troubleshoot" screen

7）Select “Windows startup settings screen” from “Advanced Options”

8）Click the "Restart" button

9）The system will reboot to the “Advanced Boot Options” screen

10）Select "Disable Driver Signature Enforcement"

11）Once the system is booted, you can install Arduino driver the same as Windows7

3、If your computer is a Windows XP system: The installation steps are basically the same as for Windows 7, please refer to the above Windows 7 installation steps.

2.2.3 IDE Interface Introduction

Nextly,we introduce the Arduino IDE interface, firstly enter the software directory. Then you can see the arduino.exe file and double-click to open the IDE. As shown in Figure 2.1.11.

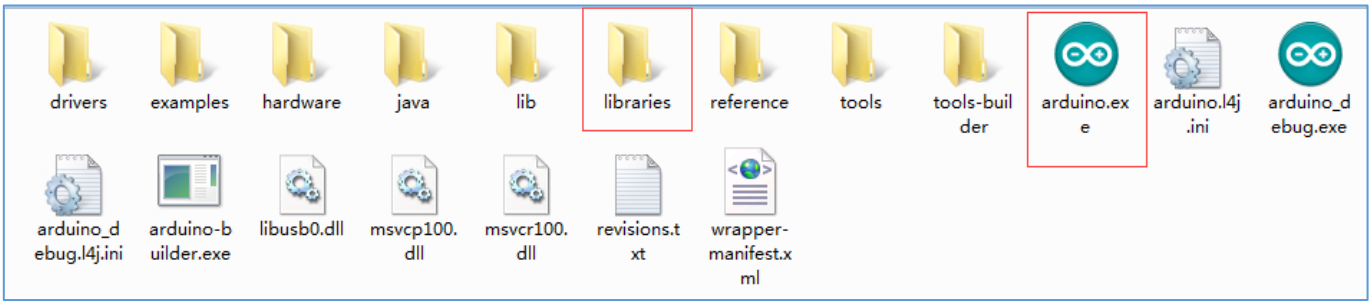


Figure 2.1.11 Software Catalog

1、The first thing you can see is the interface of the following figure. The functions of the toolbar buttons are "Compile" - "Upload" - "New Program" - "Open Program" - "Save Program" - "Serial Monitor" , as shown in Figure 2.1.12.

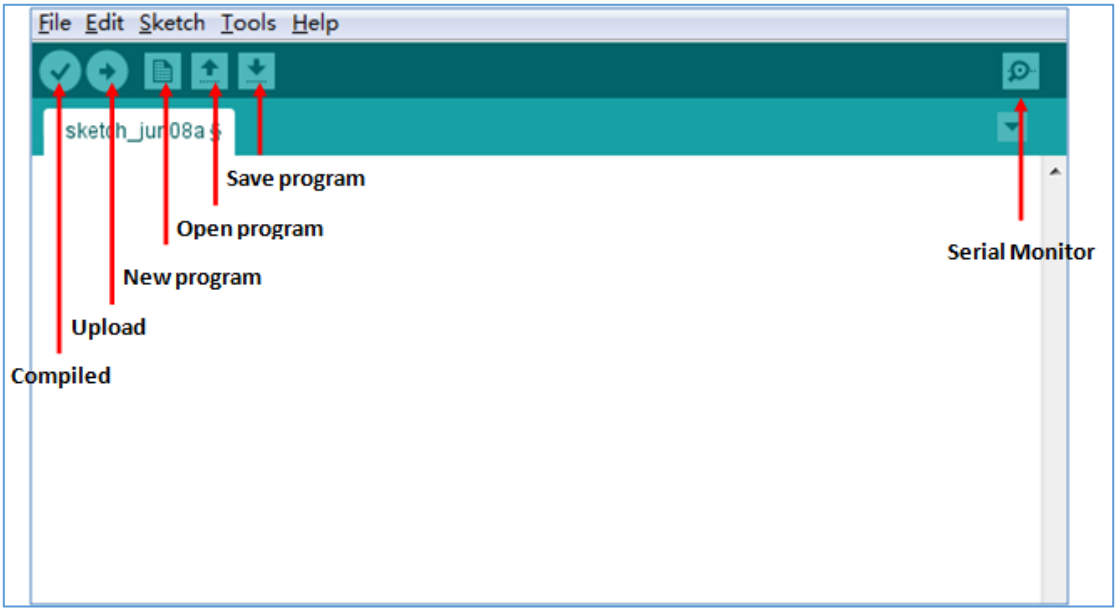


Figure 2.1.12 Arduino IDE Interface

2.There are 5 menus on the menu bar, but we mainly introduce File and Tools. Click File, the interface as shown in Figure 2.1.13 will be displayed, you can see the Examples and Preference options. The Examples are some of the Arduino's own programs, these are compiled without errors, the normal use of the program, a great help for beginners. The Preference option, It's mainly about the parameter settings, such as language, fonts and so on.

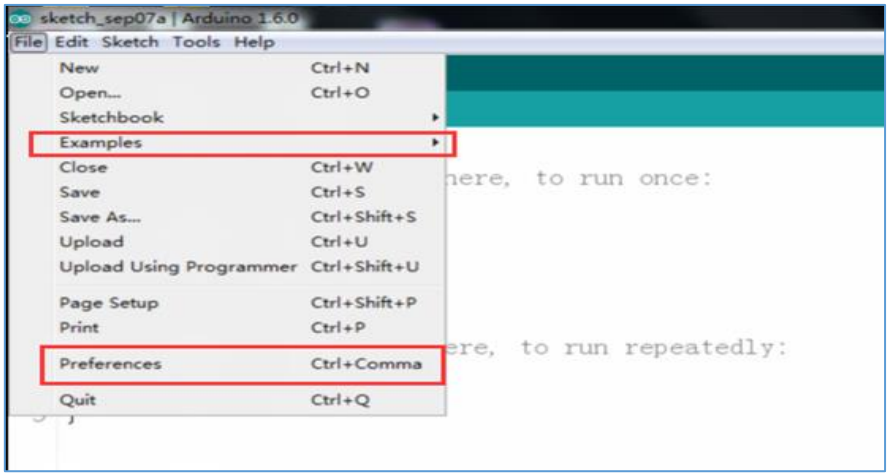


Figure 2.1.13 File Menu Bar Options

3、Click Tools, the interface shown in Figure 2.1.14 will pop up. Here we can see two options: Board and Port. In the board option, we can see the commonly used Arduino development board model, we only need to choose according to their own development board. In the Port option, the USB serial port is mainly selected, as shown in Figure 2.1.15. If you are not sure, you can check it in the "Device Manager" and select the corresponding COM port.

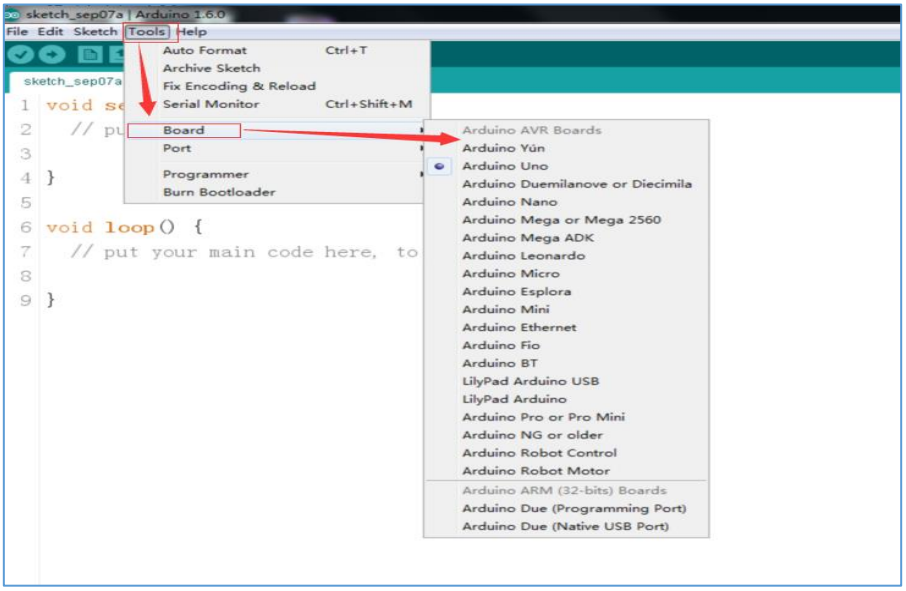


Figure 2.1.14 Tools interface

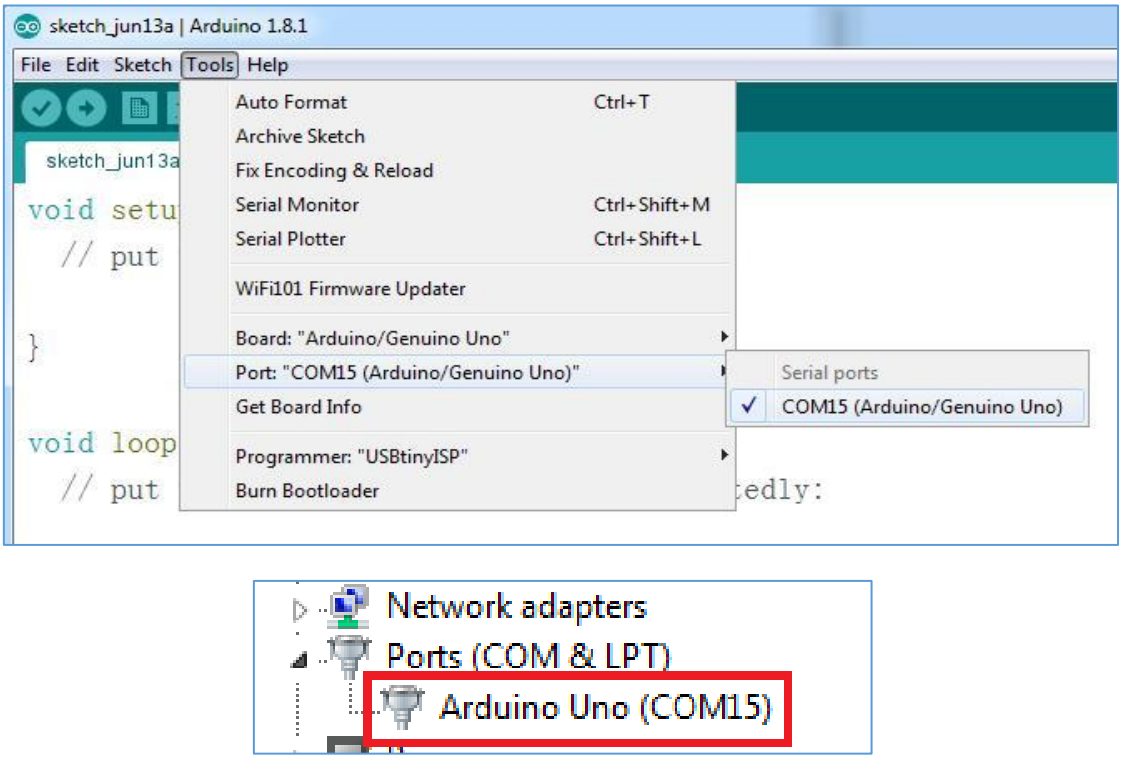


Figure 2.1.15 USB serial port selection

So far, we have basically completed all the work. The next step is actual experiments. Open any program in Examples. First compile the program. If it is compiled correctly, it can be directly downloaded to the development board and the corresponding device of the connection number. With wires, you can see the corresponding phenomenon.

Chapter3 Hummer-Bot Assembly

Cart assembly please see the assembly video.

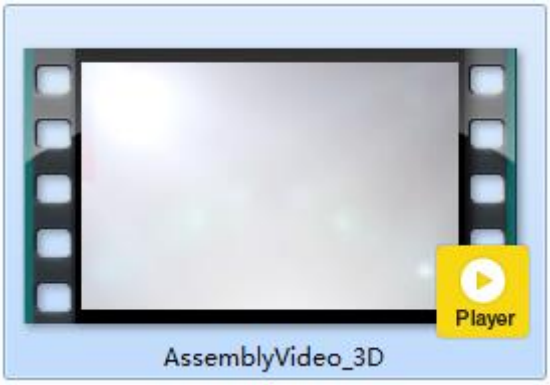


Figure 2.2.26 3D installation animation video screenshot

Chapter4 Hummer- Bot Module Experiment

4.1 Hummer Bot Module experiment

4.1.1 Motor drive principle

In the "Hummer-Bot" car, we choose the L298N as the motor driver chip for it is a high voltage and current full-bridge driver chip, the chip uses 15 pins package. It is a special motor driven integrated circuit (two H bridges) with high voltage and current full-bridge driver. And it contains 4 channel logic drive circuit, basically belongs to a kind of two-phase and four-phase special motor drive which contains two H bridges of high voltage large current. The output current is 2A, the maximum current is 4A, the maximum working voltage is 50V, which can drive the load under 46V and 2A, such as high power DC motor, stepper motor, solenoid valve and so on. The chip with two enable control terminals uses the standard logic level to control signals, allows or prohibits the device to work when the input signal is not interfered, it has a logic power input terminal which can enable the internal logic circuit to work under low voltage, and feedback the variation to the control circuit. Especially, the input can be connected directly with the MCU and easily controlled. When the DC motor is driven, the stepper motor can be directly controlled, and it can be turned forward and reversely, which only needs to change the logic level of the input. The pin arrangement is shown in Fig.3.2.1. The pin 1 and 15 can separately connect to the current sampling resistor and form the current sensing signal.

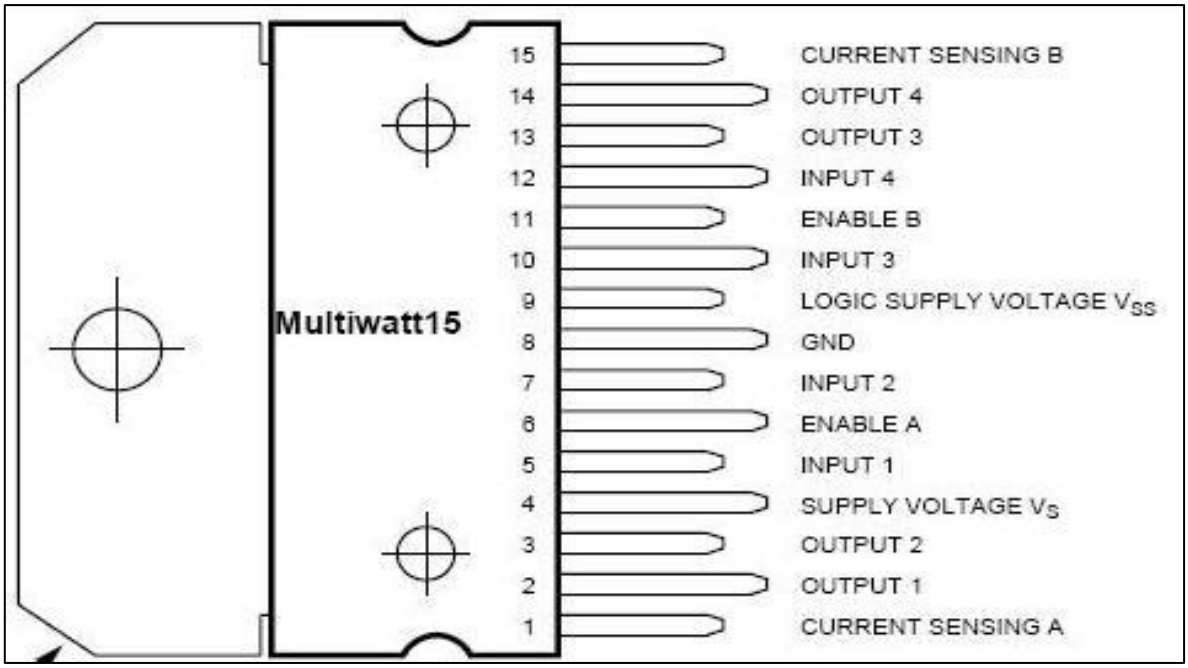


Figure .4.1.1 Arrangement of Chip Pins

L298N can drive 4 motors which are connected between OUTl, OUT2 and OUT3, OUT4. 5, 7, 10 and 12 pin are connected to input control level for controlling the positive and negative rotation of the motor, ENA, ENB are connected to control enable terminal for controlling the running and shutdown of the motor. Its characteristics:

- Signal indicator
- The speed is adjustable
- The strong anti-interference ability with photoelectric isolation
- Overvoltage and overcurrent protection
- Controlling of two motors separately
- Controlling the stepper motor
- The speed control with PWM pulse width
- Positive and negative rotation

ENA	IN1	IN2	Motor status
H	H	L	Forward
H	L	H	Reversal
H	IN2	IN1	Quick stop
L	X	X	Stop

Figure 4.1.2 Logic Function Chart

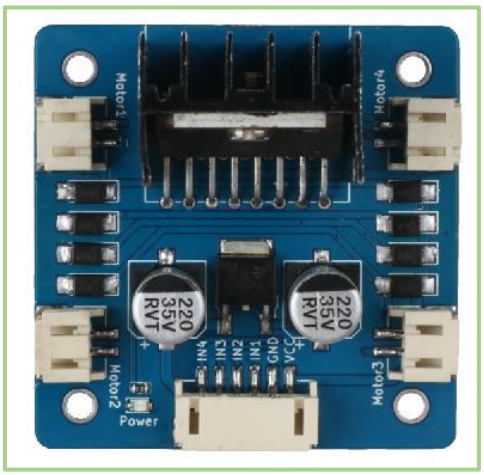


Figure 4.1.3 Module physical map

Detailed L298N chip data please refer to "hummer-bot\ Document\L298N_datasheet.pdf"

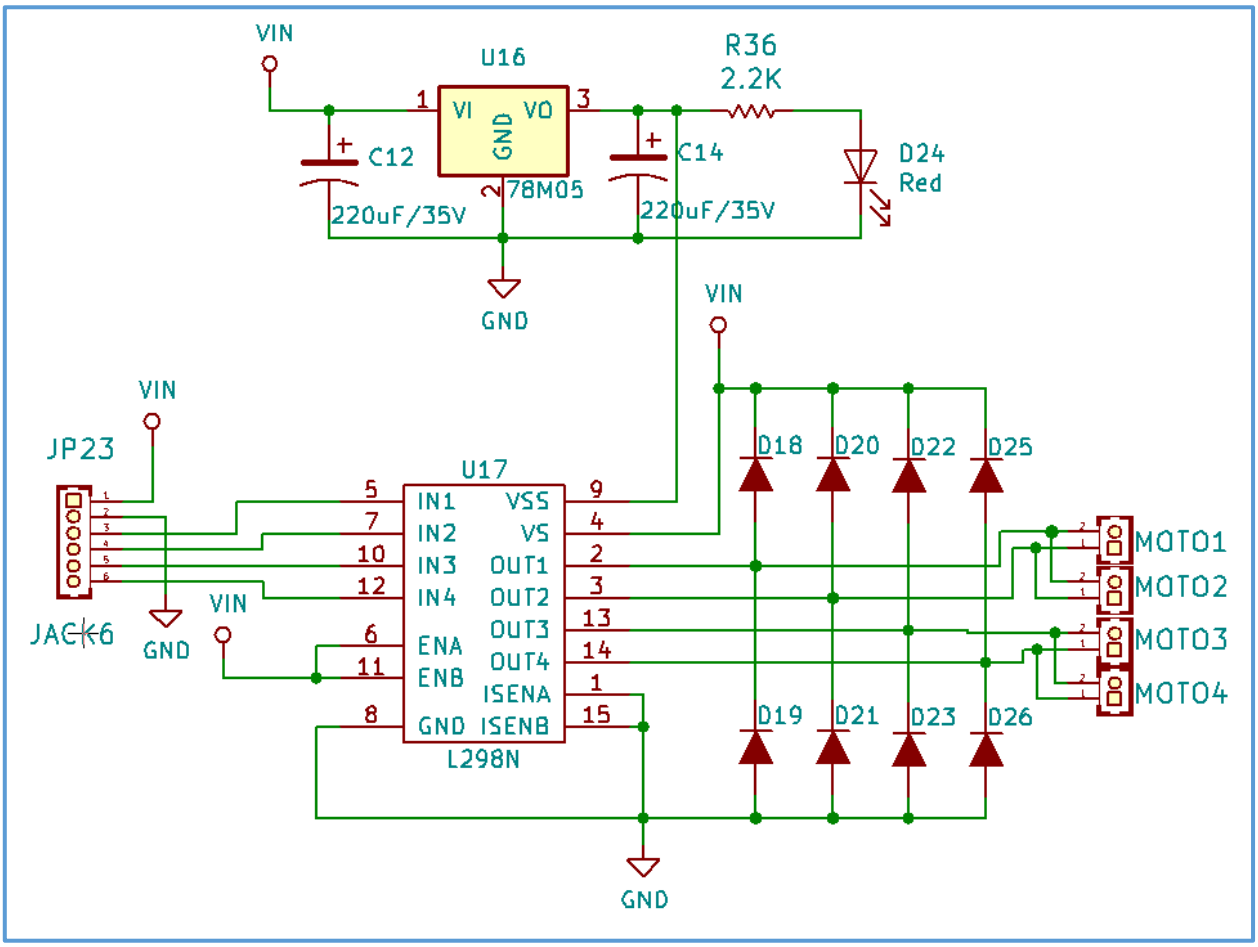


Figure 4.1.4 Schematic Diagram of Motor Drive

4.1.2 DC motor speed control principle

Four DC motors with high power L298N drive enable "Hummer-Bot" to run faster than conventional two-wheel car, the acceleration time is shorter and the structure is more stable. However, in the actual application, we need to adjust the speed of the car because of environmental or other factors, yet this does not affect the forward, backward, stop, flexible steering of the car, so we use PWM to control the speed of the motor(Note: PWM is a way to simulate the simulation output via square waves with different duty cycles.), Arduino PWM port outputs a series of square waves with fixed frequency, the power and current of the motor can be amplified after receiving the signal, thereby changing the motor's speed. The speed coordination of two motors on the right and left wheels can achieve the forward, backward, turning and other functions of the car. Figure 2.4.5 shows the sequence diagram of PWM duty cycles.

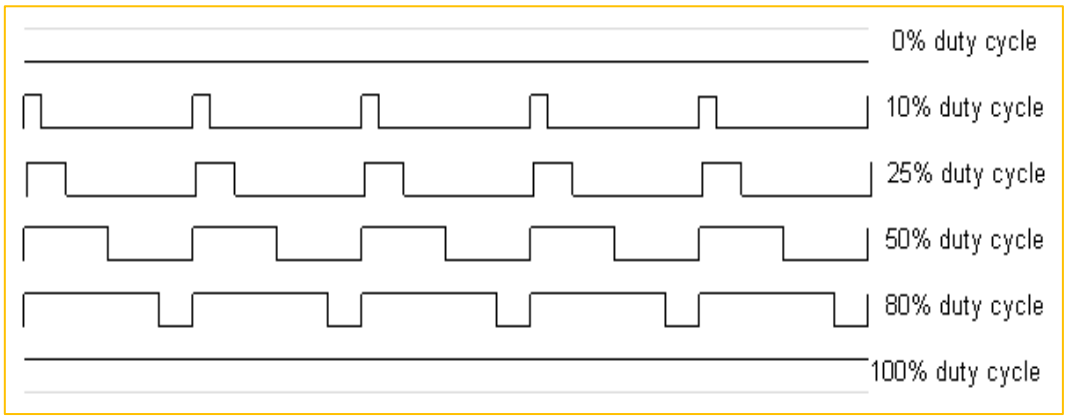


Figure 4.1.5 Sequence Diagram of PWM Duty Cycles

In Arduino, analog voltage can't be output, only 0 or 5V digital voltage value, we can use high resolution counter and the duty cycles of the square wave modulation method to encode a specific level of analog signal. The PWM signal is still digital, because at any given time, the full amplitude of DC power supply is either 5V (ON) or 0V (OFF). The voltage or current source is added to the analog load with a ON or OFF repetitive pulse sequence. When the DC power supply is added to the load, the power supply is on, otherwise the power supply is off. As long as the bandwidth is enough, any analog value can use PWM to encode. The output voltage value is calculated by the on and off time. Output voltage = (turn-on time / pulse time) * maximum voltage. Fig.2.4.6 shows the corresponding voltage to the pulse change.

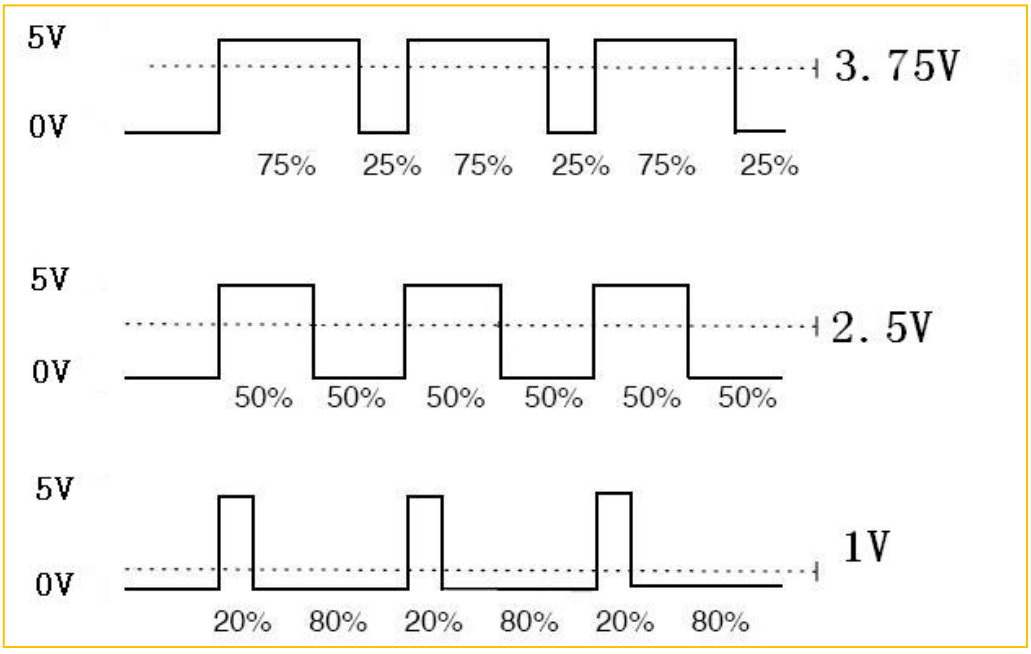


Figure 4.1.6 Relation between Pulse and Voltage

4.1.3 Motor drive board wire connection

By referring to the chip data, we will know that Arduino UNO has 6 PWM pins, namely digital interfaces 3, 5, 6, 9, 10, 11, and we select 5, 6, 9, 10 as the motor control IO, the connection is shown in Fig 4.1.7. L298N and Arduino expansion board wiring method is as follows：

L298N	Arduino UNO
VCC	VIN
GND	GND
IN1	6
IN2	10
IN3	5
IN4	9

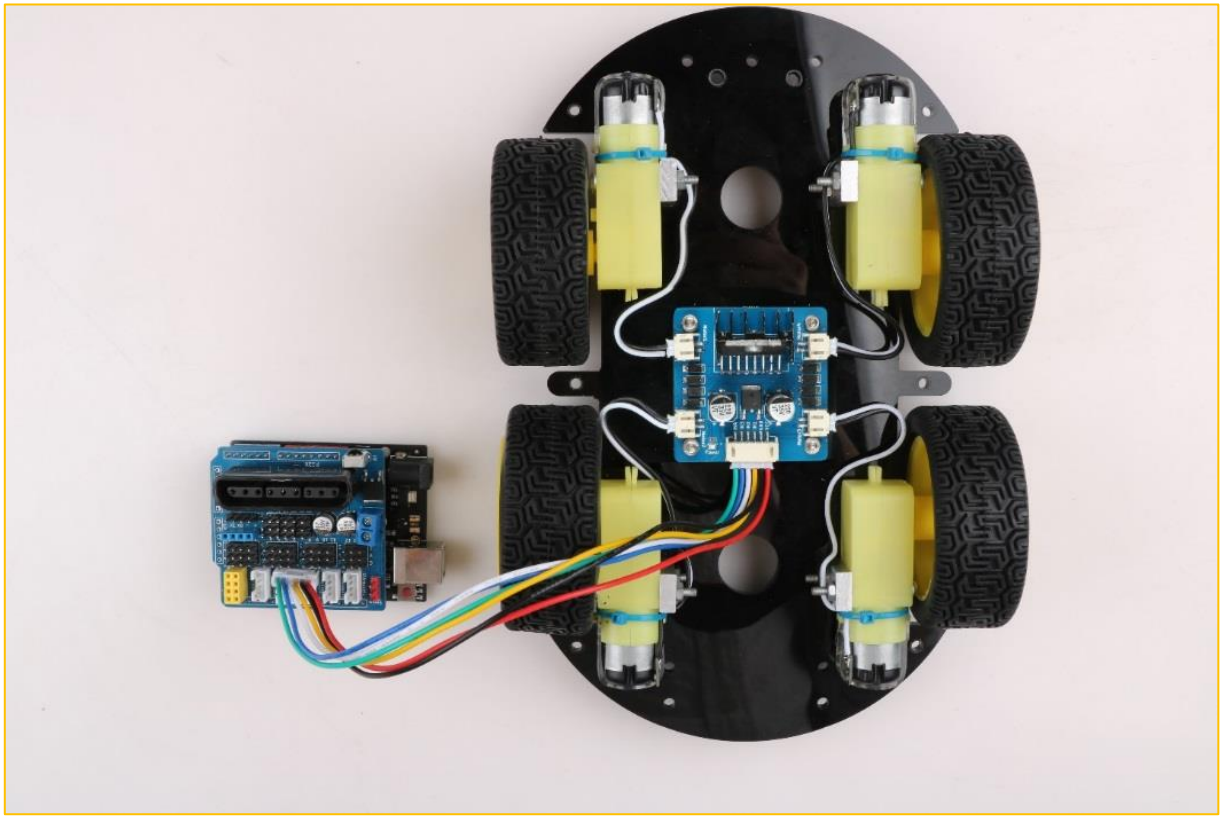


Figure 4.1.7　L298N driver board and arduino expansion board connection diagram

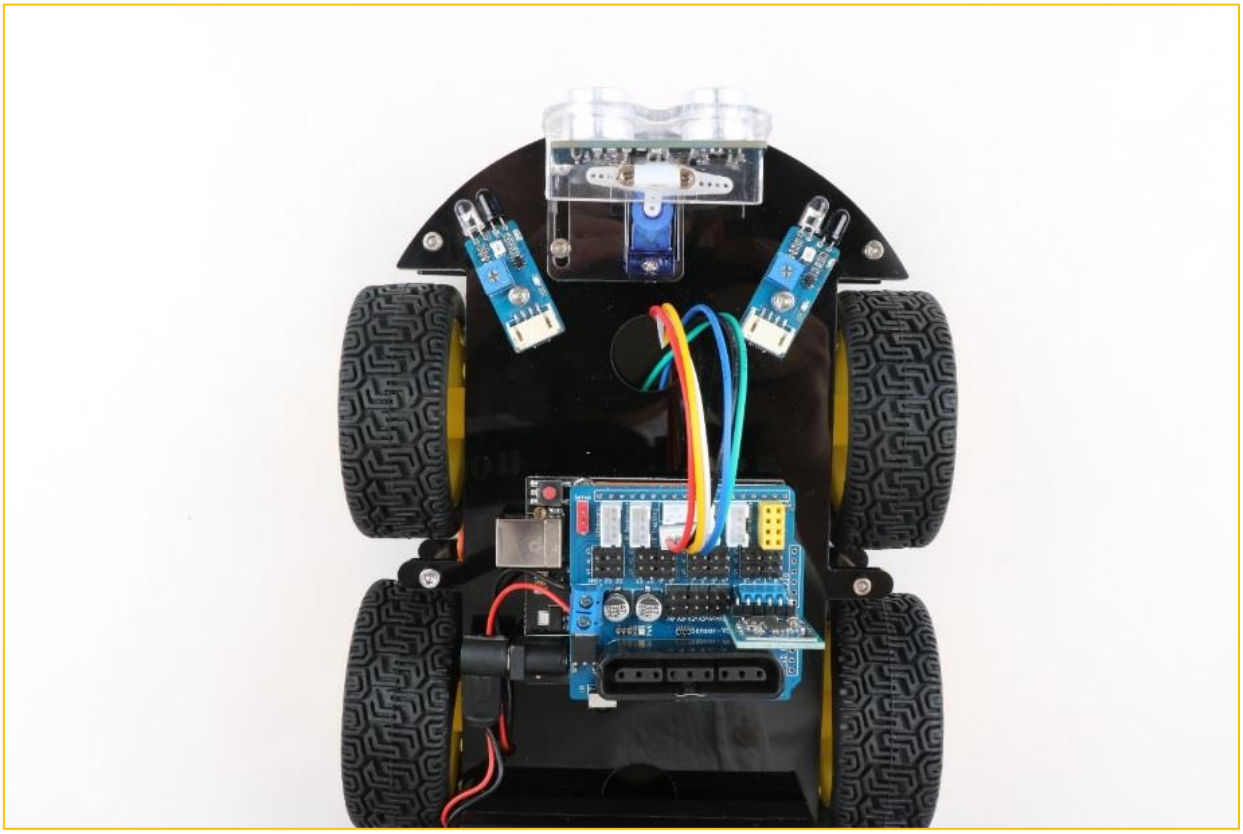


Figure 4.1.8 L298N and Arduino expansion board connection physical map

4.1.4 Motor Test Experimental Procedure

After the connection is over, we still don't know if the motor can work normally and the steering is consistent, so we need to do a simple test. The test steps are as follows:

1) Connect the Arduino UNO control board and computer via USB cable;

2) Open in the disc data "Lesson\Module_Test\Motor_Test\Motor_Test.ino"

3) Burn the motor test program to the Arduino UNO main control board

Turn on the power to observe the wheel rotation, if it appears: "Advance 5s----stop 1s----reverse 5s----stop 1s----left turn 3s----stop 1s----turn right 3s".

Note: If the motor does not change the line to check if the contact is poor, the battery is fully charged (6.5V or more), If the car is not following the procedure, please turn off the L298N driver board.

```
#define IN1_PIN 6
#define IN2_PIN 10
#define IN3_PIN 5
#define IN4_PIN 9
```

```
void setup() {
  Serial.begin(9600);
  pinMode(IN1_PIN, OUTPUT);
  digitalWrite(IN1_PIN, LOW); // When not sending PWM, we want it low
  pinMode(IN2_PIN, OUTPUT);
  digitalWrite(IN2_PIN, LOW); // When not sending PWM, we want it low
  pinMode(IN3_PIN, OUTPUT);
  digitalWrite(IN3_PIN, LOW); // When not sending PWM, we want it low
  pinMode(IN4_PIN, OUTPUT);
  digitalWrite(IN4_PIN, LOW); // When not sending PWM, we want it low
}

void loop() {
  analogWrite(IN1_PIN, 200);
  analogWrite(IN2_PIN, LOW);
  analogWrite(IN3_PIN, LOW);
  analogWrite(IN4_PIN, 200);
  delay(5000);
  //******** ******************************//forward
  analogWrite(IN1_PIN, LOW);
  analogWrite(IN2_PIN, LOW);
  analogWrite(IN3_PIN, LOW);
  analogWrite(IN4_PIN, LOW);
  delay(1000);//*********************************************//stop
  analogWrite(IN1_PIN, LOW);
  analogWrite(IN2_PIN, 200);
  analogWrite(IN3_PIN, 200);
  analogWrite(IN4_PIN, LOW);
  delay(5000);//*********************************************//back
  analogWrite(IN1_PIN, LOW);
  analogWrite(IN2_PIN, LOW);
  analogWrite(IN3_PIN, LOW);
  analogWrite(IN4_PIN, LOW);
  delay(1000);
  //******* ************************************//stop
  analogWrite(IN1_PIN, 200);
  analogWrite(IN2_PIN, LOW);
  analogWrite(IN3_PIN, 200);
```

```
  analogWrite(IN4_PIN, LOW);
  delay(3000);
  //****************************************//left
  analogWrite(IN1_PIN, LOW);
  analogWrite(IN2_PIN, LOW);
  analogWrite(IN3_PIN, LOW);
  analogWrite(IN4_PIN, LOW);
  delay(1000); //****************************************//stop
  analogWrite(IN1_PIN, LOW);
  analogWrite(IN2_PIN, 200);
  analogWrite(IN3_PIN, LOW);
  analogWrite(IN4_PIN, 200);
  delay(3000);//*** ************************************//right
}
```

By now, the car can move normally, next we will add several common sensor modules.

4.2 Infrared obstacle avoidance and finder module test experiment

4.2.1 Introduction to infrared obstacle avoidance and light seeking module

Infrared obstacle avoidance and finder module integrates infrared obstacle avoidance function and chasing function on one module , the infrared obstacle avoidance function is to emit an infrared signal through an infrared transmitting tube on the module , when the infrared signal encounters an obstacle being launched back , the infrared receiving tube receives the reflected infrared signal , thus judging that there are obstacles , To achieve the purpose of avoiding obstacles . The light-seeking function is realized by using a photoresistor on the module , When the photoresistor is illuminated by strong light , Its resistance value drops rapidly , the current passed increases, the resistance of the photoresistor rises rapidly in a dark environment, the current passed through is reduced, The main control board determines whether there is a light source.

4.2.2 Principle of infrared obstacle avoidance

The infrared obstacle avoidance and light-seeking module has a pair of infrared emitting and receiving tubes, and the transmitting tube emits infrared rays of a certain frequency. When the detecting direction encounters an obstacle (reflecting surface), the infrared light is reflected back and received by the receiving tube, and passes through the comparator. After the circuit is processed, the green indicator light will be on, and the signal output interface will output a digital signal (a low level signal). The detection effective range is 2~30cm, and the working voltage is 3.3V-5V. The detection distance of the sensor can be adjusted by the potentiometer. Because of the use of infrared rays, the anti-interference ability is very strong, and the

measurement accuracy is high when the distance is moderate. In addition, the module is easy to assemble and easy to use, and can be widely used in many situations such as robot obstacle avoidance, obstacle avoidance trolley, pipeline counting and black and white line tracking.

4.2.3 Principle of light seeking

The optical finder function of the infrared obstacle avoidance and finder module is to output the analog signal through the photodiode to judge the light intensity of the surrounding environment. When the photoresistor is illuminated by strong light, its resistance value drops rapidly, and the passing current increases. The resistance of the photoresistor rises rapidly in the dark environment, and the passing current decreases, and the main control board determines whether there is a light source. Working voltage: 0.3-10V. Since the SG-PT3528 photodiode is used, it can simulate the human eye sensitization, the peak sensitivity wavelength is 590 nm, it can resist infrared interference, and the response speed is fast and the performance is stable. In addition, the module is easy to assemble and use, and can be widely used in nightlights, lawn lamps, sun lamps, and the like.

4.2.4 Infrared obstacle avoidance and finder module parameters

- Working voltage: 5V
- Infrared detection effective distance range: 2 ~ 30cm
- Peak sensitivity wavelength: 590nm
- Output infrared obstacle avoidance signal: digital signal
- Output light signal: analog signal

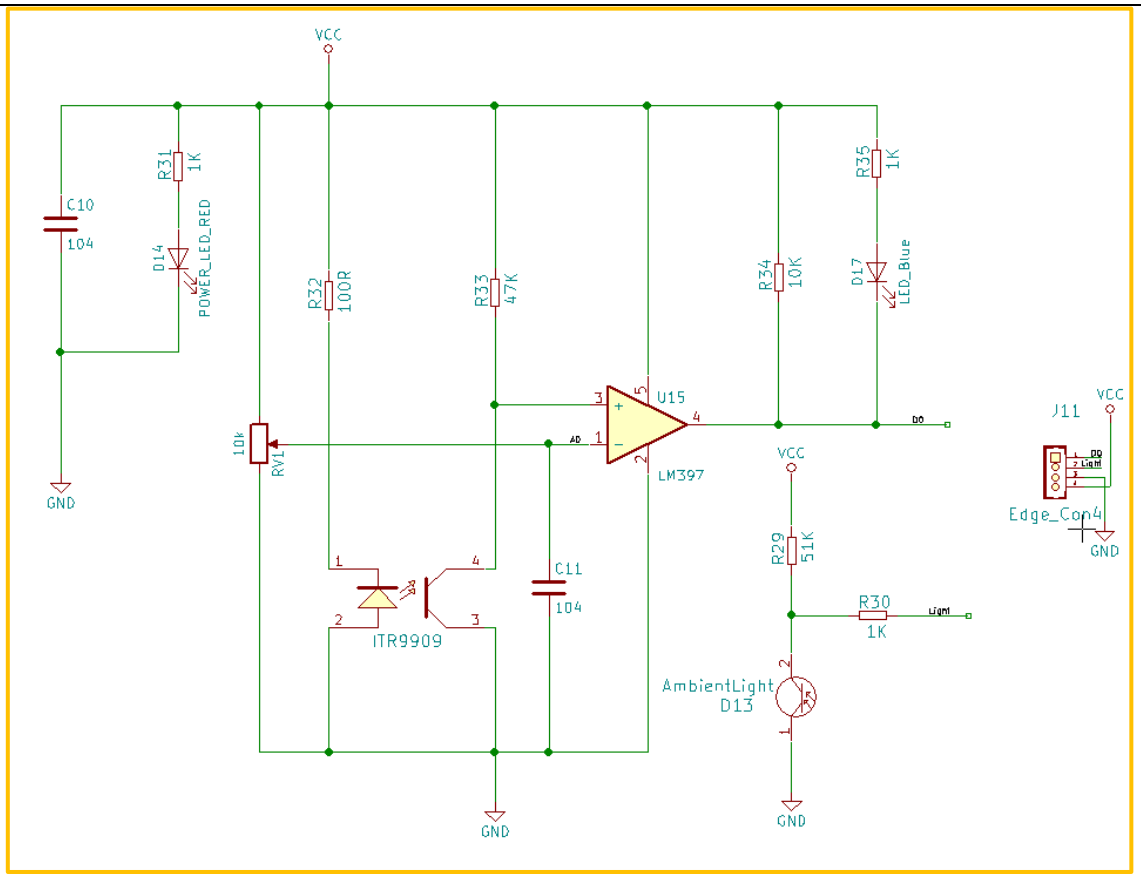


Figure 4.2.1 Schematic diagram of infrared obstacle avoidance and finder module

Note: This module can adjust the distance of infrared obstacle avoidance detection by potentiometer. The detection distance is 2~30cm. If you find that the distance measurement is not very sensitive during use, you can use the trimmer potentiometer to achieve the desired result (clockwise potentiometer, The detection distance is increased; the counterclockwise potentiometer is used to reduce the detection distance), as shown in Figure 4.2.2.

Manual adjustment as shown:

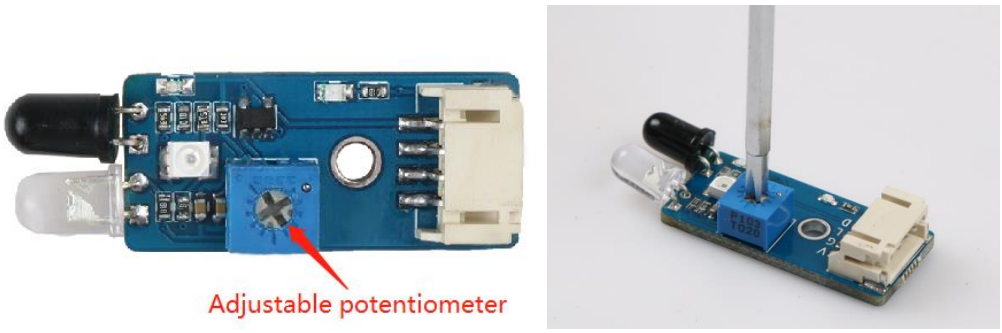


Figure 4.2.2 Schematic diagram of detection distance adjustment

4.2.5 Infrared obstacle avoidance and light seeking module wire connection

Left Infrared obstacle avoidance module	Arduino UNO	Right Infrared obstacle avoidance module	Arduino UNO
V	VCC	V	VCC
G	GND	G	GND
L	A2	L	A4
D	A3	D	A5

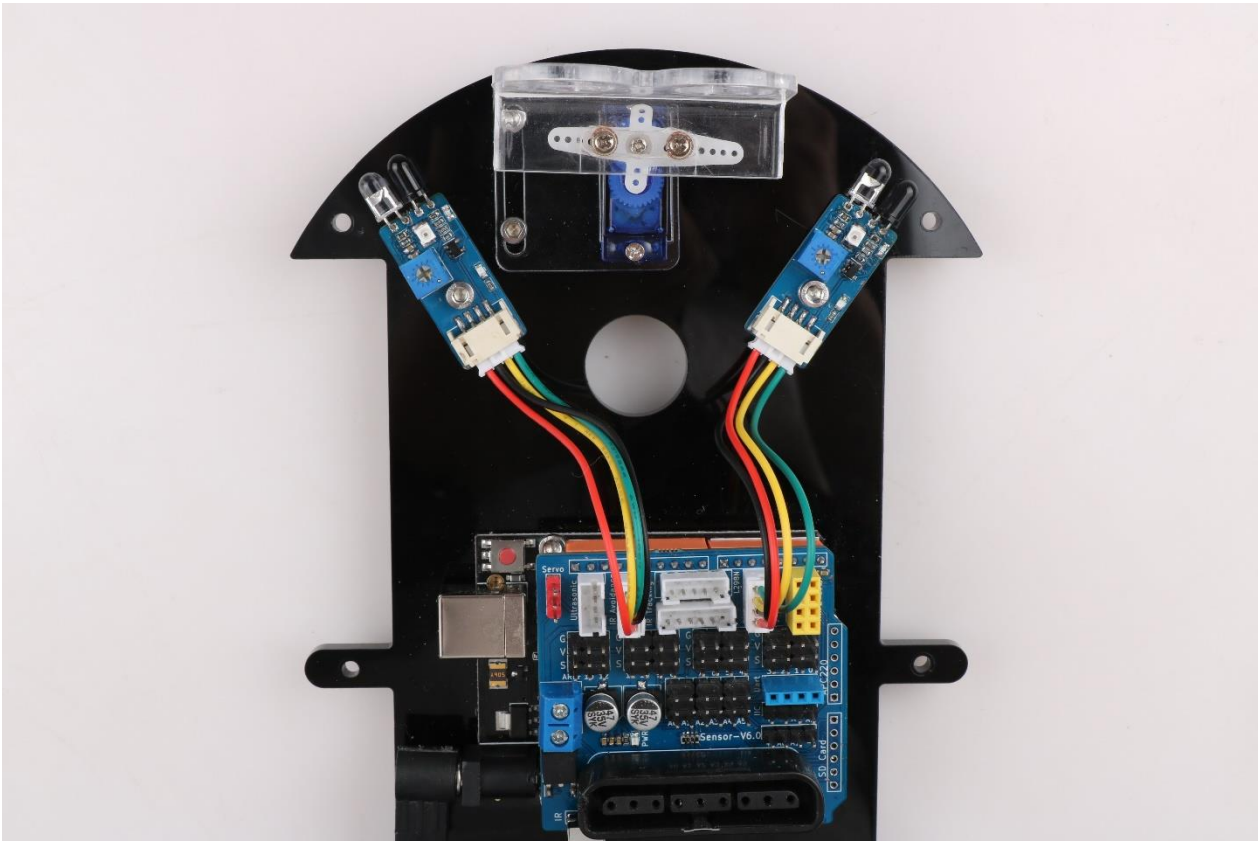


Figure 4.2.3 Schematic diagram of infrared obstacle avoidance and ray-seeking module wire connection

4.2.6 Infrared obstacle avoidance and finder module test experimental steps

1) Fix the two sensors on the trolley and connect them to the Arduino UNO R3 expansion board with wires (completed)

2) Openin the disc data "Lesson\Module_Test\InfraredAvoidanceAndLightSeekingModule_Test\ InfraredAvoidanceAndLightSeekingModule_Test.ino"

3) Burn the infrared obstacle avoidance and finder module test program to Arduino UNO R3.

4) Open the serial monitor.

5) Open the switch on the battery box. At this time, the power indicator on the module lights up. Place the obstacle 10cm in front of the infrared transmitting tube and the receiving tube, and fine tune the potentiometer until the output indicator lights up. Observed at this time. The output of the infrared obstacle avoidance signal on the serial monitor is 0. When the infrared obstacle avoidance signal output on the serial port monitor of the obstacle is removed, the infrared obstacle avoidance function is proved to be normal.

6) Use a cover to block the light near the photoresistor, and then observe that the output value of the light-seeking signal on the serial monitor is relatively large, and then remove the obstruction. It is found that the output value of the light-seeking signal on the serial monitor becomes smaller. You can also use the flashlight of the mobile phone to illuminate the photoresistor to observe the change of the output value of the light-seeking signal on the serial monitor.

```
const int LeftAvoidancePin = A3;
const int RightAvoidancePin = A5;
const int LeftLightPin = A2;
const int RightLightPin = A4;

int dl, dr, LL, LR;
void setup() {
  Serial.begin(9600);
  pinMode(LeftAvoidancePin, INPUT);
  pinMode(RightAvoidancePin, INPUT);
  pinMode(LeftLightPin, INPUT);
  pinMode(RightLightPin, INPUT);
  delay(1000);
}

void loop() {
  dl = digitalRead(LeftAvoidancePin);
  dr = digitalRead(RightAvoidancePin);
  LL = analogRead(LeftLightPin);
  LR = analogRead(RightLightPin);
  Serial.print("LeftAvoidance:");
  Serial.print(dl);
  Serial.print("   ");
  Serial.print("RightAvoidance:");
  Serial.println(dr);
```

```
Serial.print("LeftLight:");
Serial.print(LL);
Serial.print("   ");
Serial.print("RightLight:");
Serial.println(LR);
delay(1000);
}
```

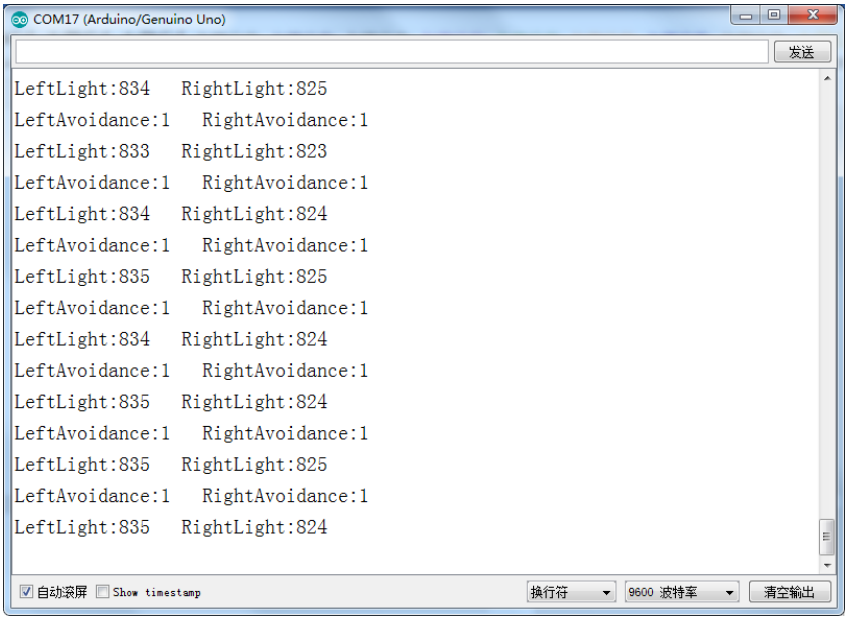


Figure 4.2.4 Schematic diagram of data when there is no obstacle or ambient light

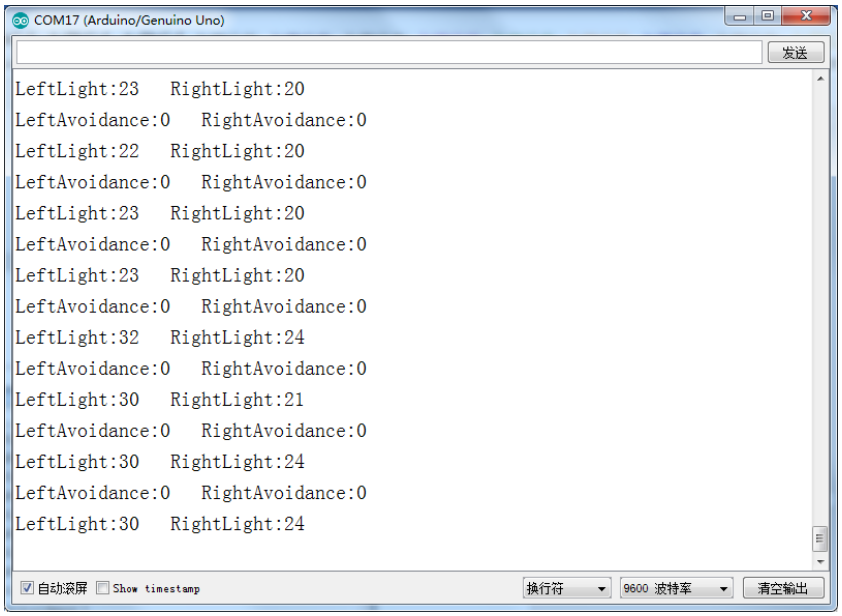


Figure 4.2.5 Schematic diagram of obstacles and strong ambient light

4.3 Hummer-Bot infrared obstacle avoidance function experiment

4.3.1 Hummer-Bot Infrared Obstacle Avoidance Software Logic

In the previous section, we have tested the infrared obstacle avoidance and finder module and learned the principle of the infrared obstacle avoidance function. In this section, we learn how to implement the Hummer-Bot infrared obstacle avoidance function. To realize the infrared obstacle avoidance function, I need to combine the infrared obstacle avoidance with the finder module and the motor. The logic judgment process is shown in Figure 4.3.1.

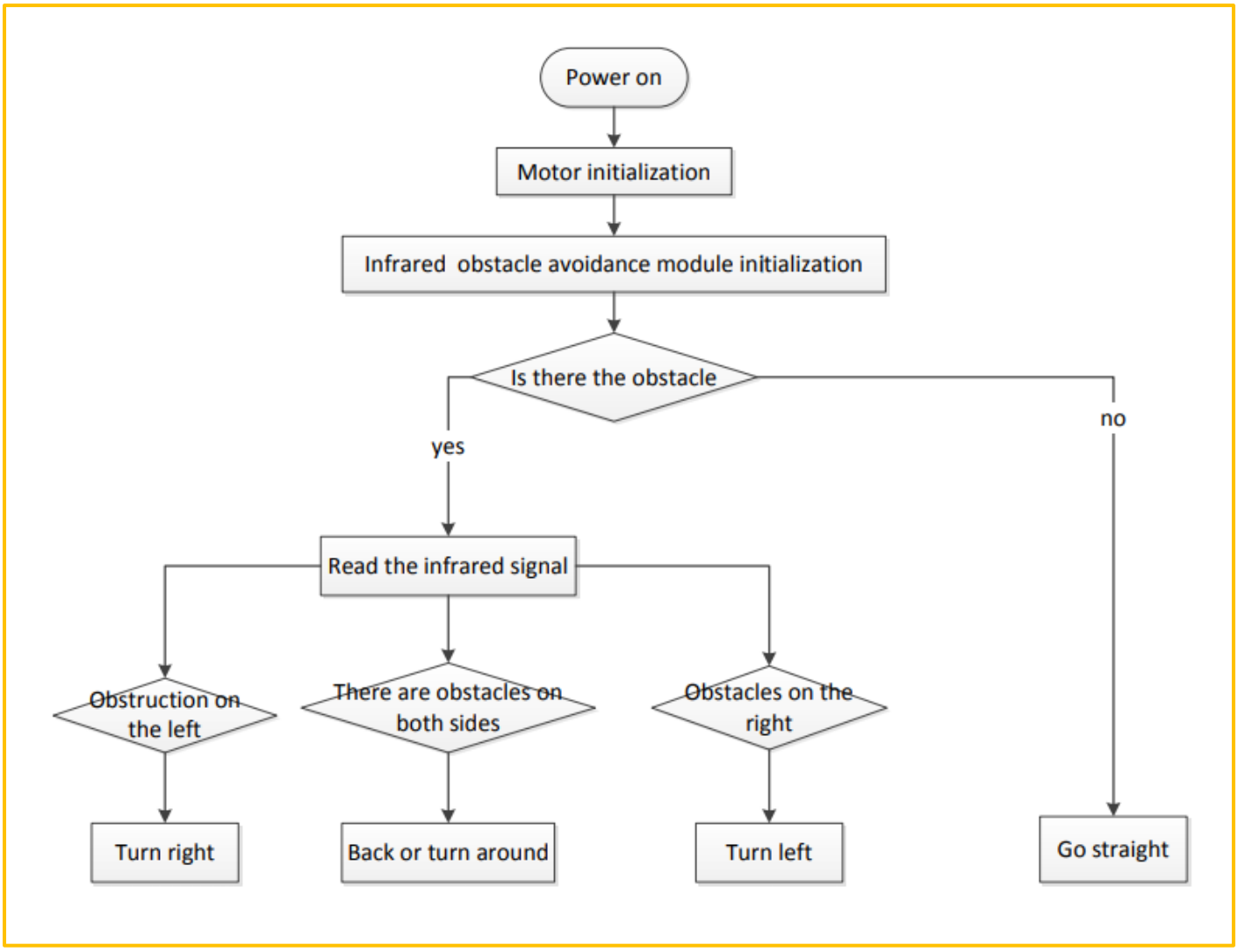


Figure 4.3.1 Infrared obstacle avoidance logic flow chart

The essence of infrared obstacle avoidance is that there are obstacles on the side to avoid obstacles, so every time you want to avoid obstacles, you must first obtain the direction of the car where the obstacle is

located, then the main control board then controls the motor to the opposite direction. The direction is turned to achieve the purpose of obstacle avoidance.

4.3.2 Hummer-Bot infrared obstacle avoidance experiment wiring diagram

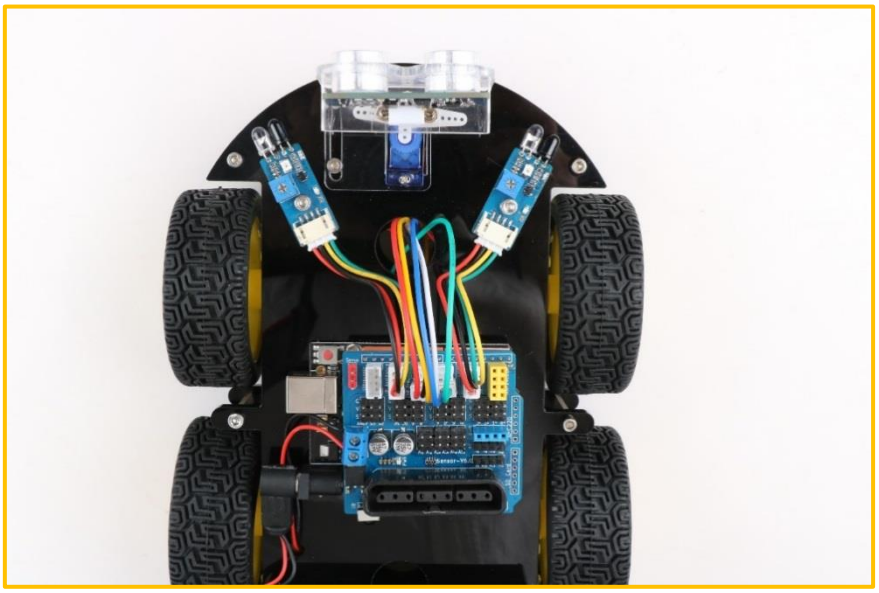


Figure 4.3.2 Hummer-Bot infrared obstacle avoidance experiment wiring diagram

4.3.3 Hummer-Bot infrared obstacle avoidance function program code

Understand the programming structure of the infrared obstacle avoidance function, we first burn an infrared obstacle avoidance program to the Hummer-Bot main control board;

1) Openin the disc data "Lesson\Advanced Experiment\InfraredAvoidance_Function\ InfraredAvoidance_Function.ino"
2) Burn the InfraredAvoidance_Function .ino program to the Arduino UNO R3 main control board;
3) Place the car on a relatively flat floor, turn on the power, and observe the progress of the car.

```
#define IN1_PIN 6   // PWMB
#define IN2_PIN 10  // DIRB  ---  right
#define IN4_PIN 9   // PWMA
#define IN3_PIN 5   // DIRA  ---  left
const int leftPin = A3;
const int rightPin = A5;
byte LeftValue,RightValue;
void setup()
{
  Serial.begin(9600);
```

```
  pinMode(leftPin, INPUT);
  pinMode(rightPin, INPUT);
  delay(1000);
}

void loop()
{
  LeftValue = digitalRead(leftPin);
  RightValue = digitalRead(rightPin);
  if (LeftValue >= 1 && RightValue >= 1)
  {
    analogWrite(IN1_PIN, 180); //the speed value of motorA is val
    analogWrite(IN2_PIN, LOW);
    analogWrite(IN3_PIN, LOW);
    analogWrite(IN4_PIN, 180); //the speed value of motorB is val
    Serial.print(LeftValue);
    Serial.print("   ");
    Serial.print(RightValue);
    Serial.print("   ");
    Serial.println("go");//*********************************************//forward
    } else if (LeftValue >= 1 && RightValue < 1) {
        analogWrite(IN1_PIN, 200);//the speed value of motorA is 200
        analogWrite(IN2_PIN, 0);
        analogWrite(IN3_PIN, 200);//the speed value of motorB is 200
        analogWrite(IN4_PIN, 0);
        Serial.print(LeftValue);
        Serial.print("   ");
        Serial.print(RightValue);
        Serial.print("   ");
        Serial.println("Turning left");
        delay(300);
        analogWrite(IN1_PIN, 0);
        analogWrite(IN2_PIN, 0);
        analogWrite(IN3_PIN, 0);
        analogWrite(IN4_PIN, 0);
        delay(1000); //*********************************************//Turning left
    } else if (LeftValue < 1 && RightValue < 1) {
        analogWrite(IN1_PIN, 0);
        analogWrite(IN2_PIN, 255); //the speed value of motorA is 255
```

```
        analogWrite(IN3_PIN, 0);
        analogWrite(IN4_PIN, 255); //the speed value of motorB is 255
        Serial.print(LeftValue);
        Serial.print("   ");
        Serial.print(RightValue);
        Serial.print("   ");
        Serial.println("Turning around");
        delay(500);
        analogWrite(IN1_PIN, 0);
        analogWrite(IN2_PIN, 0);
        analogWrite(IN3_PIN, 0);
        analogWrite(IN4_PIN, 0);
        delay(1000); //*******************************************//Turning around
    } if (LeftValue < 1 && RightValue >= 1) {
        analogWrite(IN1_PIN, 0);
        analogWrite(IN2_PIN, 200);//the speed value of motorA is 200
        analogWrite(IN3_PIN, 0);
        analogWrite(IN4_PIN, 200);//the speed value of motorB is 200
        Serial.print(LeftValue);
        Serial.print("   ");
        Serial.print(RightValue);
        Serial.print("   ");
        Serial.println("Turning right");
        delay(300);
        analogWrite(IN1_PIN, 0);
        analogWrite(IN2_PIN, 0);
        analogWrite(IN3_PIN, 0);
        analogWrite(IN4_PIN, 0);
        delay(1000); //*******************************************//Turning right
    }
}
```

In the above procedure, we have annotated some of the programs to facilitate the learning and understanding of the program. This section of the program is relatively simple. After learning, you can write according to your own ideas, giving the car more movements and postures. Of course, if you want to use it directly, we have the corresponding source program in the supporting CD, which can be used by directly downloading it to the car.

4.4 Hummer-Bot light seeking function experiment

4.4.1 Hummer-Bot light seeking function software logic

The previous chapters have explained the principle of the Hummer-Bot homing function. Next, we will practice the finder function in a specific way, similar to the infrared tracking. In order to realize the Hummer-Bot homing function, the finder module and the motor control need to be combined, so that when the car senses strong light, it will move in the direction of the light. The specific software design idea is as shown in Figure 4.4.1. Shown

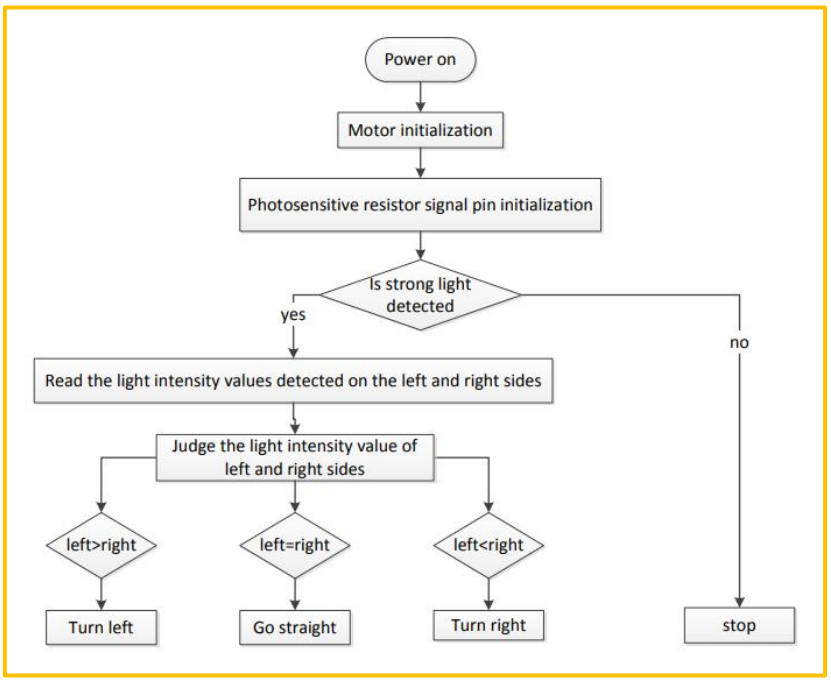


Figure 4.4.1 Finder function software design flow chart

4.4.2 Hummer-Bot light seeking function code

Understand the programming structure of the light-seeking function, we first burn a program of light-seeking function to the Hummer-Bot main control board;

1) Openin the disc data "Lesson\Advanced Experiment\ LightSeeking_Function \ LightSeeking_Function.ino"
2) Burn the LightSeeking_Function.ino program to the Arduino UNO R3 main control board;
3) Place the car on a relatively flat floor, turn on the power, and illuminate the photoresistor on the module with a flashlight to observe the progress of the car.

4) Disconnect the IR tracking module when using the homing function, because the infrared obstacle avoidance and finder module pins are multiplexed with the pins of the IR tracking module to cause interference. Therefore, please disconnect the infrared tracking module when using the infrared obstacle avoidance module and the finder module.

By observing the results of the obstacle avoidance procedure of the car, we will look at the complete program to deepen our understanding.

```
#define IN1_PIN 6   // PWMB
#define IN2_PIN 10  // DIRB  ---  right
#define IN4_PIN 9   // PWMA
#define IN3_PIN 5   // DIRA  ---  left
const int leftPin = A2;
const int rightPin = A4;
float LeftValue,RightValue;
int Angle;
float f;
// When using this program, please pull out the connection line of the infrared
tracing module, just pull one end of the expansion board.
void setup()
{
  Serial.begin(9600);
  pinMode(leftPin, INPUT);
  pinMode(rightPin, INPUT);
  delay(1000);
}

void loop()
{
  LeftValue = analogRead(leftPin) / 10;
  RightValue = analogRead(rightPin) / 10;
  if ( (LeftValue > 50) && (RightValue > 50)) {
    analogWrite(IN1_PIN, LOW); //the speed value of motorA is val
    analogWrite(IN2_PIN, LOW);
    analogWrite(IN3_PIN, LOW);
    analogWrite(IN4_PIN, LOW); //the speed value of motorB is val
  } else {
    if (LeftValue > RightValue) {
      Angle = ((float)(RightValue/LeftValue)) * 90;
    } else if (LeftValue < RightValue) {
```

```
      Angle = (90 - ((float)(LeftValue/RightValue)) * 90) + 90;
    }
    Serial.println(Angle);
    if (Angle <= 90) {
      f = (float)(Angle) / 90;
      analogWrite(IN1_PIN, (float)(200 * f)); //the speed value of motorA is val
      analogWrite(IN2_PIN, LOW);
      analogWrite(IN3_PIN, LOW);
      analogWrite(IN4_PIN, 200); //the speed value of motorB is val
    }
    if (Angle > 90) {
      f = (float)(180 - Angle) / 90;
      analogWrite(IN1_PIN, 200); //the speed value of motorA is val
      analogWrite(IN2_PIN, LOW);
      analogWrite(IN3_PIN, LOW );
      analogWrite(IN4_PIN, (float)(200 * f)); //the speed value of motorB is val
    }
  }
}
```

4.5 Infrared tracking module test experiment

4.5.1 Infrared tracking module introduction

The Hummer-Bot uses a 3-way infrared tracking module that can follow either a black line or a white line. If the floor of your experiment is black, then you should follow the white line (that is, put a white track on the floor); if the floor of your experiment is white, then you should follow the black line (ie, put a black on the floor) In other words, it is to make the track and other environments form a distinct color difference. The program in our CD-ROM data, the default is black line.

Black line tracking means that the car walks along the black line on the white floor. Because the black line and the white floor have different reflection coefficients of infrared rays, the "road" can be judged according to the strength of the received reflected infrared signal.

White line tracking means that the car walks along the white line on the black floor. Because the white line and the black floor have different reflection coefficients to the infrared rays, the "road" can be judged according to the strength of the received reflected infrared signal.

In the "Hummer-Bot" car, each tracking module uses the ITR9909 sensor, the ITR9909 infrared reflection sensor, a photoelectric sensor that combines emission and reception. It consists of an infrared light-emitting diode and an NPN infrared phototransistor. The detection reflection distance is 1mm-25mm. The sensor is specially equipped with M3 fixed mounting hole. The adjustment direction and fixing are convenient and easy to use. The 74HC14 Schmitt trigger inverter is used, the signal is clean, the waveform is good, and the driving ability is strong. It can be applied to robot obstacle avoidance, robot tracking of white line or black line, black line in white background, white line in black bottom, and is a necessary sensor for tracking robot. Pipeline metering, metering pulse data sampling, fax machine shredder paper detection and many other occasions. The small module physical map is shown in Figure 4.5.1.

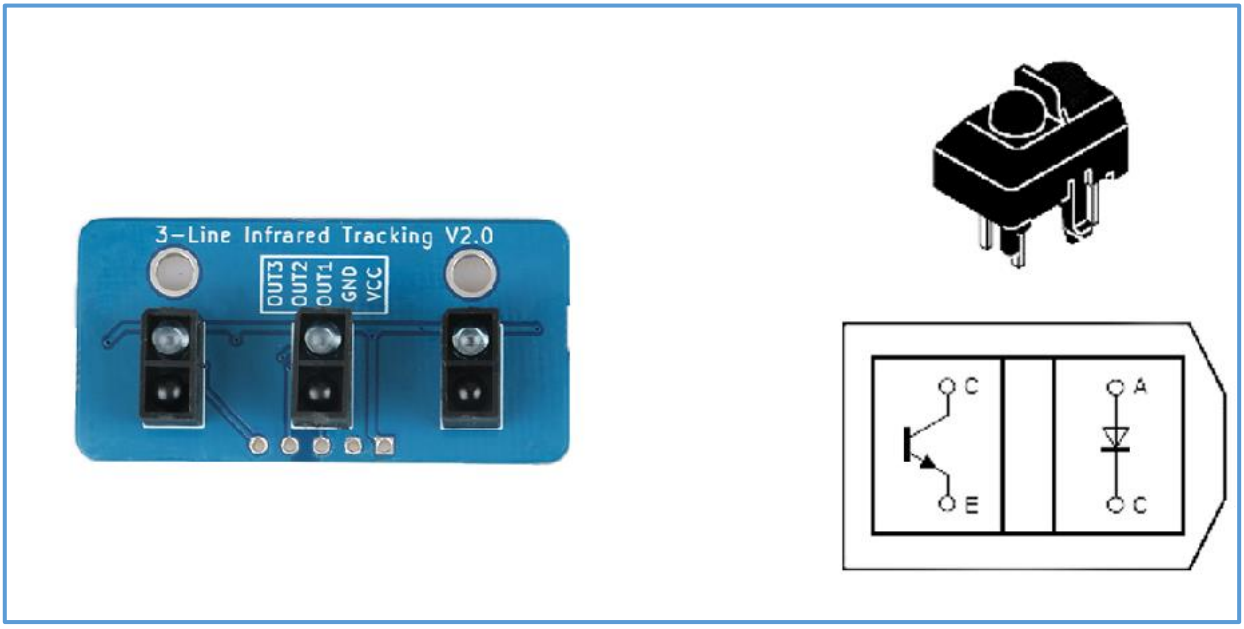


Figure 4.5.1 Infrared tracking module physical map

4.5.2 Working principle of infrared tracking module

Infrared tracking module Whether it is black or white, the method we usually take is infrared detection.

Infrared detection method, which uses infrared rays to have different reflection properties on the surface of objects of different colors, continuously emits infrared light to the ground during the driving of the car, and the infrared rays emitted are not reflected or reflected back but are not strong enough. When the time is large, the infrared receiving tube is always in the off state. At this time, the output end of the module is low, indicating that the LED is always off; when the infrared light encounters the white paper floor, diffuse reflection occurs, and the reflected light is mounted on The receiving tube on the trolley receives; and the

infrared light is reflected back and the intensity is large enough, the phototransistor is saturated, and the output of the module is at a high level, indicating that the LED is illuminated.

Hummer-Bot is based on the reflected infrared light to determine the position of the black line and the walking path of the car.

In the "Hummer-Bot" car, we use a three-way tracking module, one module has three sensors, one on each side, one in the middle, as shown in Figure 4.5.3, the tracking sensors are all in a straight line.

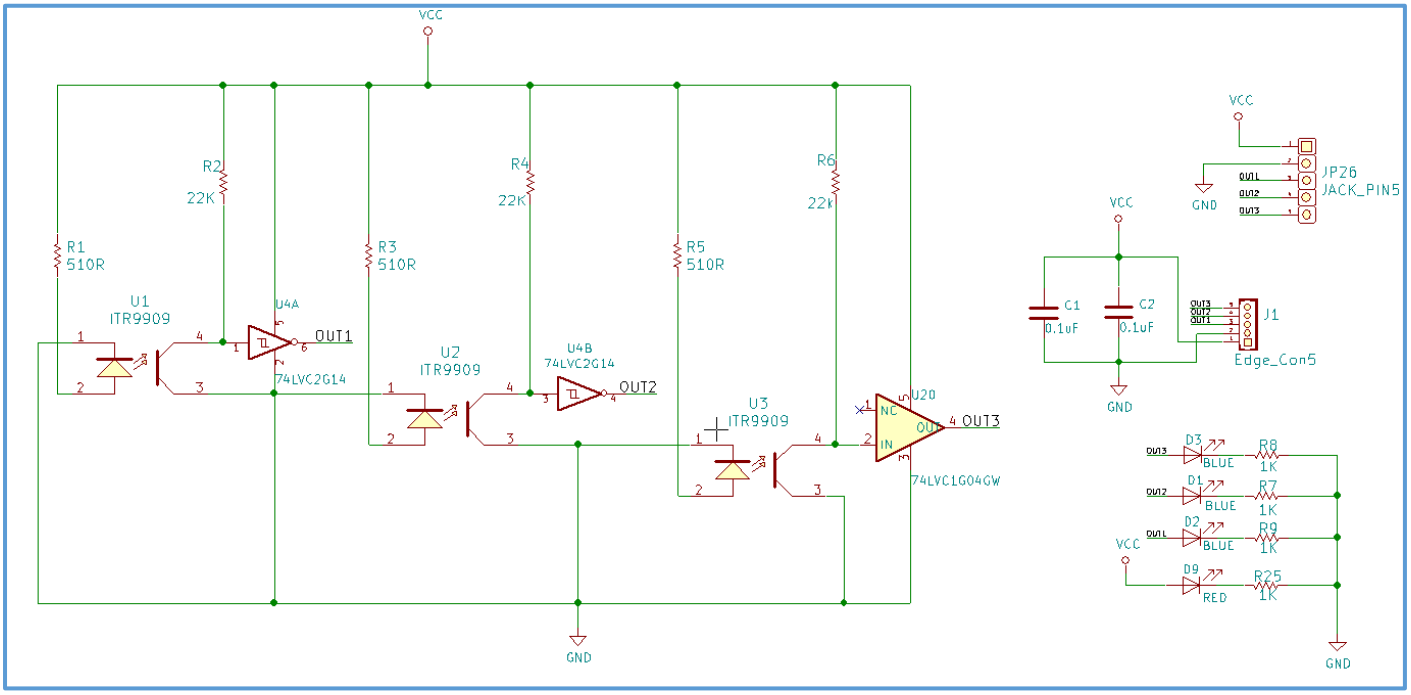


Figure 4.5.2 Schematic diagram of the tracking module

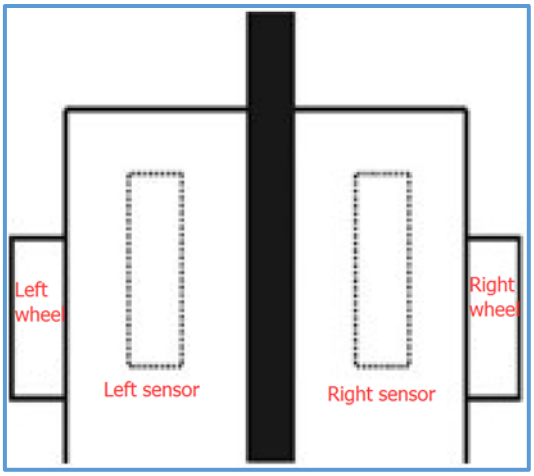


Figure 4.5.3 Schematic diagram of the tracking module installation

When the car advances, it is always between the two first-level sensors. When the car deviates from the black line: If the left sensor detects the black line, the right and middle sensors will not detect the black line,

indicating the car. Offset to the left, the cart will deflect slightly to the right, keeping the middle sensor always detecting the black line and walking along the black line; if the right sensor detects the black line, the left and middle sensors will not detect the black line , indicating that the car has been shifted to the right, then the car will be slightly deflected to the left, keeping the middle sensor always detecting the black line, and walking along the black line.

4.5.3 Infrared tracking module parameters

- Using ITR9909 infrared reflection sensor.
- Detection distance: 1mm~25mm, focus distance is 2.5mm.
- Comparator output, clean signal, good waveform, strong driving capability, over 15mA.
- Working voltage 3.3V-5V.
- Use wide voltage 74LVC1G04 and 74LVC1G4 inverters, digital switching output (0 and 1).
- Fixed bolt holes for easy installation

  For detailed parameters, please see "ITR9909.pdf"

4.5.4 Infrared tracking module wire connection

Infrared tracking module	Arduino UNO
VCC	VCC
GND	GND
OUT1	A0
OUT2	A1
OUT3	A2

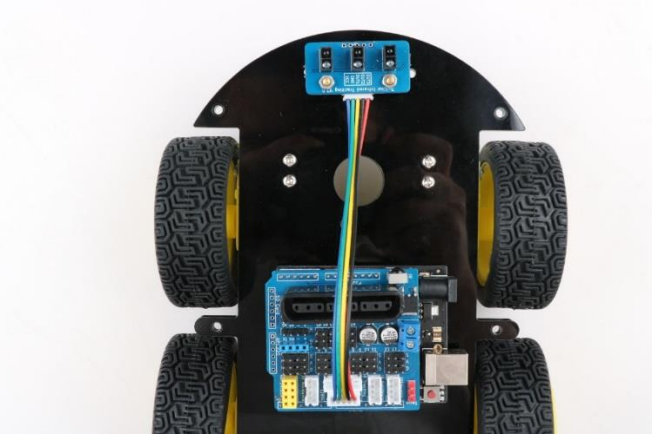


Figure 4.5.4 Schematic diagram of infrared tracking module wiring

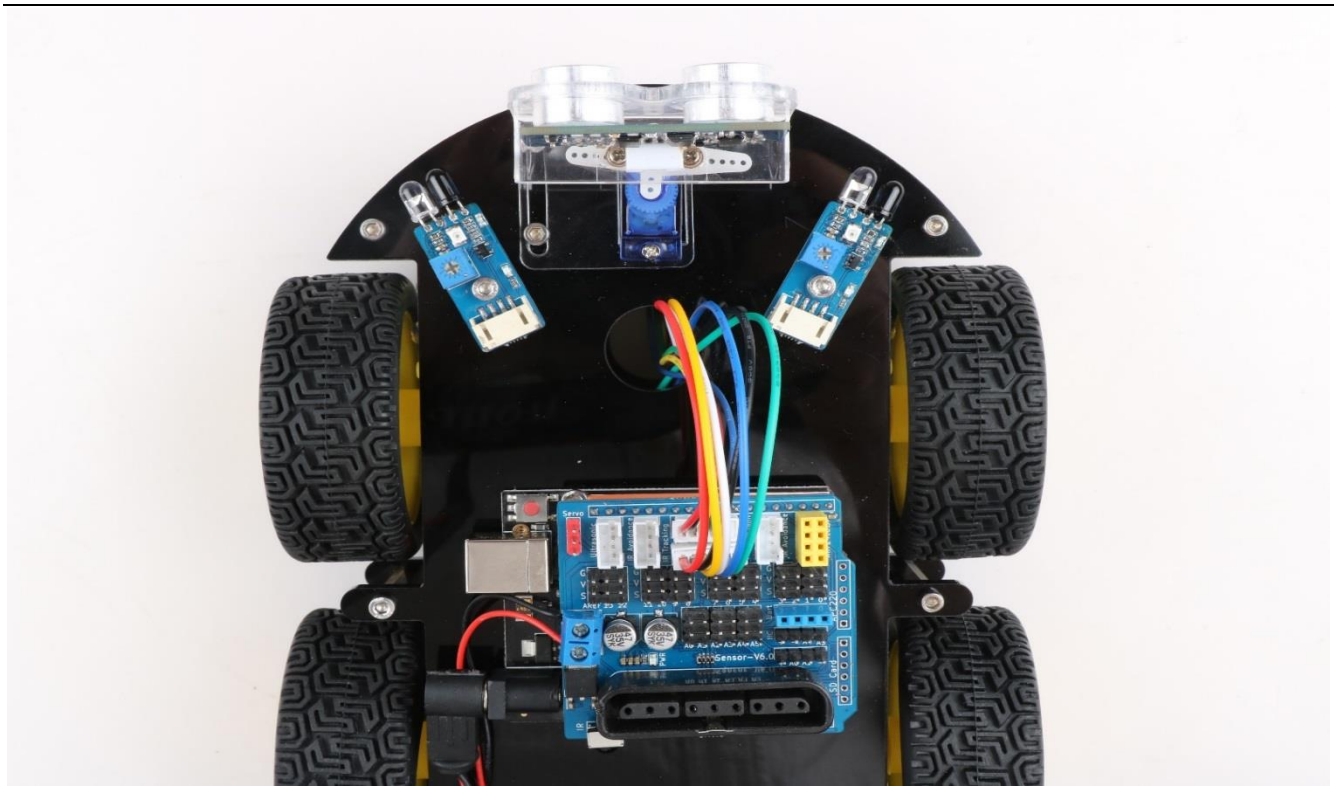


Figure 4.5.5 Infrared tracking module wiring physical map

4.5.5 Infrared tracking module test experiment steps

1) Fix the sensor on the trolley (the assembly is complete) and connect it to the Arduino with a wire.

2) On the track production, if the floor color of the experimental environment is white, then a black tape that forms a loop can be attached to the floor, and vice versa. As for the shape of the track, you can follow your own ideas. The tape has a wide bandwidth of 13 to 18 mm. In this tutorial, we are using a black line track, as shown in Figure 4.5.5.

3) Openin the disc data

"Lesson\Module_Test\InfraredTracingModule_Test\InfraredTracingModule_Test.ino"

5) Burn the infrared tracking module test program to Arduino UNO R3;
6) Open the serial monitor;
7) First block the infrared tracking module sensor with your hand. At this time, the red indicator light on the module will light up, observe the printed content on the serial monitor, and then block the infrared tracking module sensor with a black plate. At this time, the red on the module. The indicator light will not illuminate and then observe what is printed on the serial monitor.

8) If the black line is detected, the module output prints 0, if no black line is detected, the module output prints 1;

```
void setup() {
  Serial.begin(9600);
  pinMode(A0, INPUT);
  pinMode(A1, INPUT);
  pinMode(A2, INPUT);
}
void loop() {
  int left, mid, right;
  left = digitalRead(A0);
  mid = digitalRead(A1);
  right = digitalRead(A2);
  Serial.print("right:");
  Serial.print(right);
  Serial.print("   ");
  Serial.print("mid:");
  Serial.print(mid);
  Serial.print("   ");
  Serial.print("left:");
  Serial.print(left);
  Serial.println("   ");
}
```

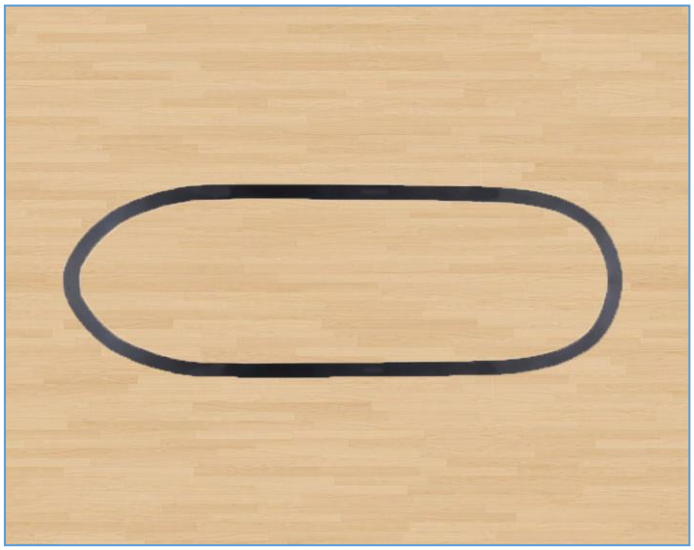


Figure 4.5.6 Schematic diagram of the black line track

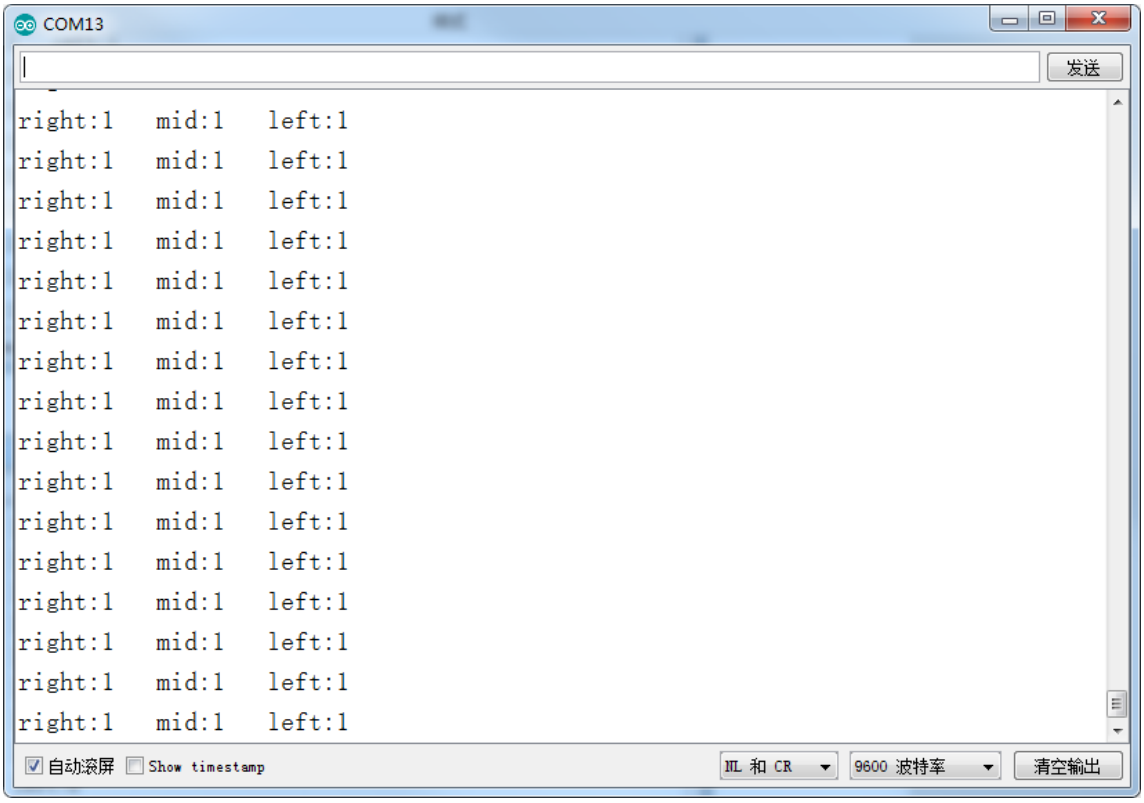


Figure 4.5.7 When the infrared tracking module does not detect the black line data

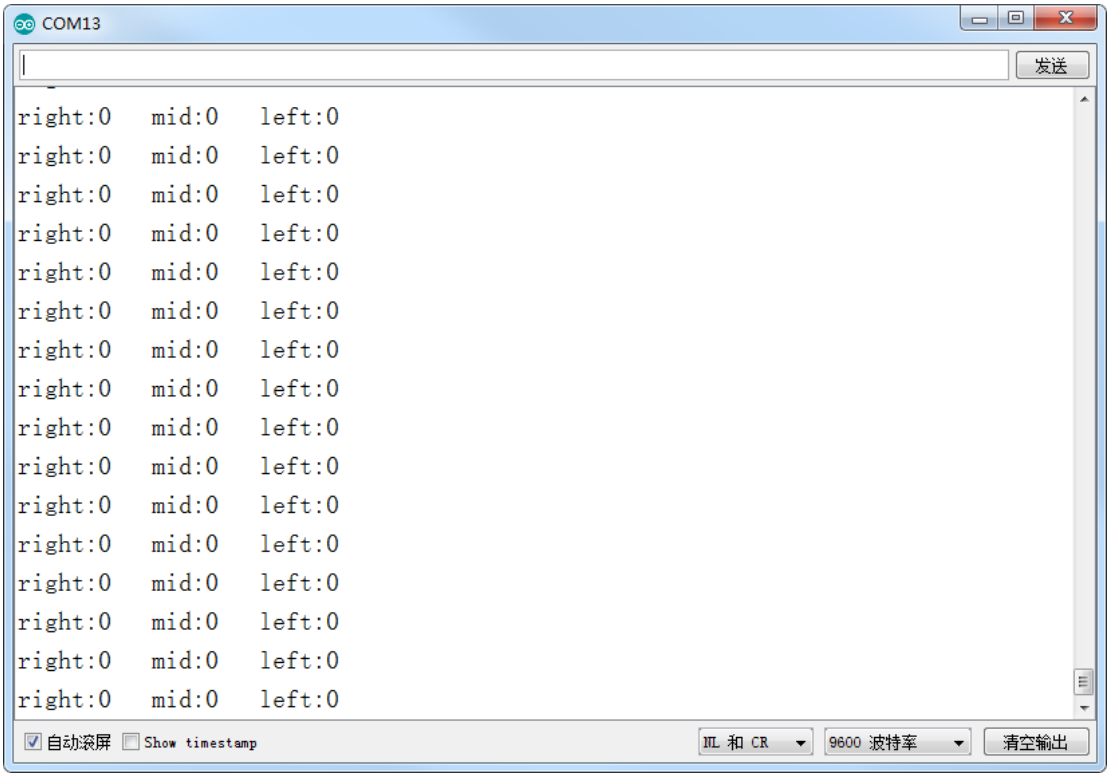


Figure 4.5.8 Data when the infrared tracking module detects a black line

From Figure 4.5.7 and Figure 4.5.8, we can see that when the infrared tracking module does not detect the black line, it outputs a high level, and when the black line is detected, it outputs a low level because the analog port is used to acquire the sensor signal. Therefore, the value printed by the serial port is also analog, that is, it is close to 1024 when it is high level and 0 when it is low level. If the output is a digital signal, the serial port prints a value of 0 or 1.

4.6 Hummer-Bot infrared tracking function experiment

In the previous chapter, we have learned the tracking principle of the infrared tracking module. In this section, we will practice the Hummer-Bot infrared tracking function. In order to make the car follow the black line, we need to combine the infrared tracking module with the motor. When the sensor of the infrared tracking module detects the black line, the main control board determines whether the position of the black line found is deviated, thereby controlling the motor to drive the four wheels to follow a certain route.

1) Openin the disc data
   "Lesson\Advanced Experiment\InfraredTracing_Function\InfraredTracing_Function.ino"
2) Burn the InfraredTracing_Function .ino program to the Arduino UNO R3 main control board;
3) Place the car on the production track, turn on the power, and observe the progress of the car;

By observing the results of the obstacle avoidance program of the car, we will look at the complete program to deepen understanding.

```
#define IN1_PIN 6   // PWMB
#define IN2_PIN 10  // DIRB  ---  right
#define IN4_PIN 9   // PWMA
#define IN3_PIN 5   // DIRA  ---  left

void setup()
{
  Serial.begin(9600);
  pinMode(A0, INPUT);
  pinMode(A1, INPUT);
  pinMode(A2, INPUT);
}

void loop()
{
  int left, mid, right;
  left = digitalRead(A0);
  mid = digitalRead(A1);
```

```
  right = digitalRead(A2);
  if ((right == 1) && (mid == 0) && (left == 1)) //******* straight *******//
  {
    analogWrite(IN1_PIN, 180);
    analogWrite(IN2_PIN, LOW);//the speed value of motorA is val
    analogWrite(IN3_PIN, LOW);//the speed value of motorB is val
    analogWrite(IN4_PIN, 180);
  }
  else if ((right == 0) && (mid == 1) && (left == 1)) //*** Left, turn right ***//
  {
    analogWrite(IN1_PIN, 20);
    analogWrite(IN2_PIN, LOW);//the speed value of motorB is val
    analogWrite(IN3_PIN, LOW);//the speed value of motorB is val
    analogWrite(IN4_PIN, 180);
  }
  else if ((right == 1) && (mid == 1) && (left == 0)) //*** Right, turn left ***//
  {
    analogWrite(IN1_PIN, 180);
    analogWrite(IN2_PIN, LOW);//the speed value of motorB is val
    analogWrite(IN3_PIN, LOW);//the speed value of motorB is val
    analogWrite(IN4_PIN, 20);
  }
  if ((right == 0) && (mid == 0) && (left == 0)) //******* stop *******//
  {
    analogWrite(IN1_PIN, LOW);
    analogWrite(IN2_PIN, LOW);
    analogWrite(IN3_PIN, LOW);
    analogWrite(IN4_PIN, LOW);
  }
}
```

4.7 Steering gear

3.2.3.1 Steering gear introduction

The steering gear is also called servo motor. It was first used to realize its steering function on the ship. Because it can continuously control its rotation angle through the program, it is widely used to realize the steering and the various joint movements of the robot. The steering gear is the control of the steering of the

trolley. The mechanism has the characteristics of small size, large torque, simple external mechanical design and high stability. Whether in hardware design or software design, the steering gear design is an important part of the car control. Generally speaking, the steering gear is mainly composed of the following The components are composed of steering wheel, reduction gear set, position feedback potentiometer, DC motor, control circuit, etc., as shown in Figure 4.7.1. Figure 4.7.2 shows the physical map of the most commonly used 9G steering gear at this stage.

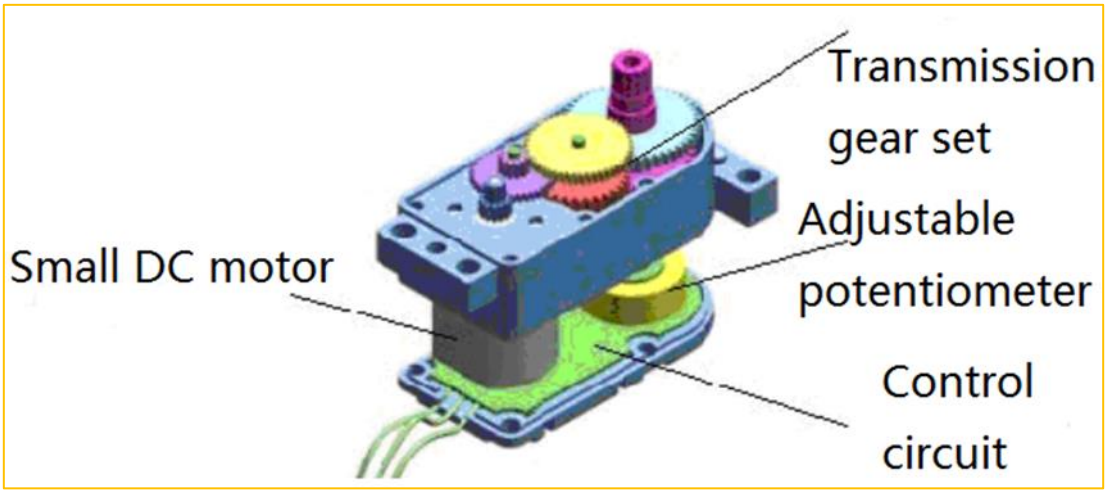


Figure 4.7.1 Schematic diagram of the steering gear

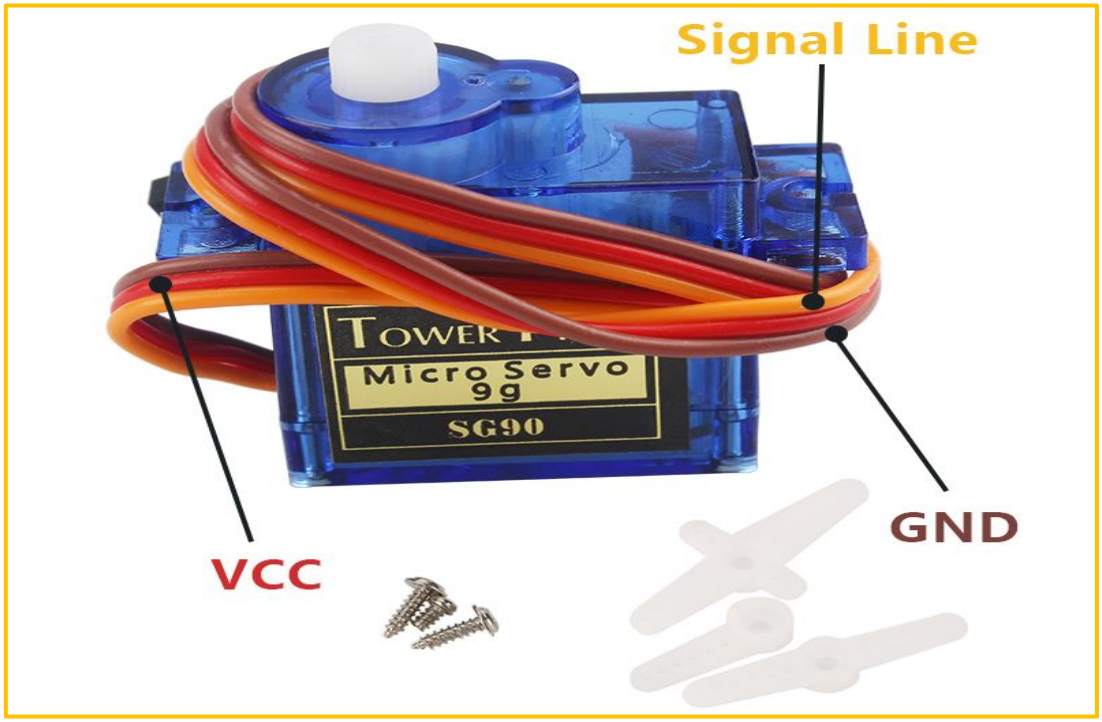


Figure 4.7.2 SG90 servo machine physical map

4.7.2 SG90 steering gear parameters

There are three input lines for the SG90 steering gear, as shown in Figure 4.7.2. The middle of the red is the power line, and the one side is the ground line. This line provides the most basic energy guarantee for the steering gear, mainly the rotation of the motor. Consumption, orange is the signal line. The power supply has two specifications, one is 4.8V, the other is 6.0V, which corresponds to different torque standards, that is, the output torque is different, and the corresponding 6.0V is larger, depending on the application conditions.

4.7.3 How the steering gear works

The control signal enters the signal modulation chip from the channel at the receiving end to obtain a DC bias voltage. It has a reference circuit inside, which generates a reference signal with a period of 20ms and a width of 1.5 milliseconds, and compares the obtained DC bias voltage with the voltage of the potentiometer to obtain a voltage difference output. Finally, the positive and negative voltage output of the voltage difference to the motor drive chip determines the forward and reverse of the motor. When the motor speed is constant, the potentiometer is rotated by the cascade reduction gear, so that when the voltage difference is 0, the motor stops rotating.

When the control circuit board receives the control signal from the signal line, the motor is controlled to rotate, and the motor drives a series of gear sets to decelerate and then drives to the output steering wheel. The output shaft of the steering gear and the position feedback potentiometer are connected. When the steering wheel rotates, the position feedback potentiometer is driven. The potentiometer will output a voltage signal to the control circuit board for feedback, and then the control circuit board determines the motor according to the position. The direction and speed of rotation to achieve the target stop. The workflow is: control signal → control circuit board → motor rotation → gear set deceleration → steering wheel rotation → position feedback potentiometer → control circuit board feedback.

The control signal period of the servo is 20MS pulse width modulation (PWM) signal, wherein the pulse width is from 0.5-2.5MS, and the corresponding steering wheel position is 0-180 degrees, which varies linearly. That is to say, give him a certain pulse width, its output shaft will maintain a certain corresponding angle, no matter how the external torque changes, until it is given a pulse signal of another width, it will change the output angle to the new The corresponding position is shown in Figure 4.7.3. There is a reference circuit inside the steering gear to generate a reference signal with a period of 20MS and a width of 1.5MS. There is a comparator that compares the applied signal with the reference signal to determine the direction and size to produce the motor's rotation signal.

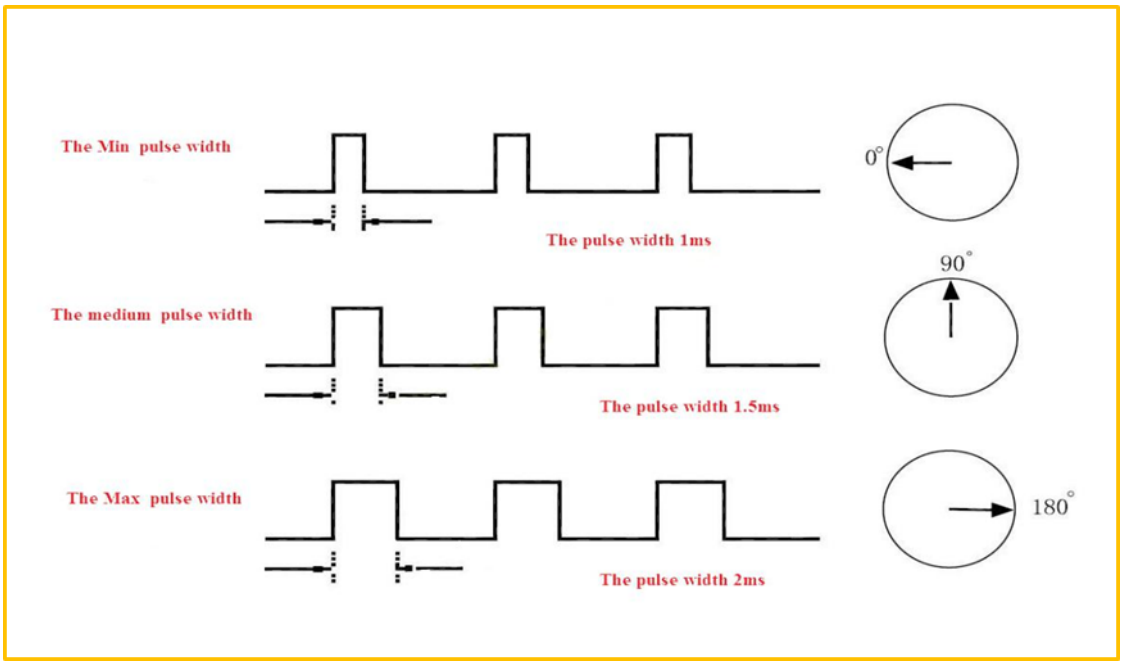


Figure 4.7.3 Relationship between the output angle of the servo and the input pulse

The test of the steering gear has been explained in Section 3.7 of Chapter 3, and will not be repeated.

4.8 RGB ultrasonic obstacle avoidance module test experiment

4.8.1 Introduction to RGB Ultrasonic Obstacle Avoidance Module

The RGB Ultrasonic Obstacle Module integrates RGB LED lights and ultrasonic ranging modules, and RGB LED lights can be used in a variety of colors.

The ultrasonic obstacle avoidance module is a device that converts other forms of energy into ultrasonic energy of a desired frequency or other forms of energy that converts ultrasonic energy into the same frequency. At present, there are two types of ultrasonic sensors commonly used, namely electroacoustic type and fluid dynamic type. Electroacoustic types mainly include: piezoelectric sensors, magnetostrictive sensors, and electrostatic sensors. The hydrodynamic type includes both gas and liquid whistle. Most of the ultrasonic sensors at this stage use piezoelectric sensors to work. Its ranging is a hot spot for ultrasound.

In the "Hummer-Bot" car, we use the ultrasonic module to measure the distance between the car and the obstacle. The module can provide 2cm-400cm non-contact distance sensing function, the ranging accuracy can be up to 3mm; The compensation corrects the ranging result and uses the GPIO communication method with the watchdog inside, which is stable and reliable. The module includes an ultrasonic transmitter, a receiver, and a control circuit. Like the distance measurement and steering of smart cars, or some projects, it is often used. The intelligent car distance measurement can find the obstacles in front in time, so that the

smart car can turn in time and avoid obstacles. Figure 4.8.1 is the physical map of the RGB ultrasonic obstacle avoidance module.

Figure 4.8.1 Physical diagram of the ultrasonic module

4.8.2 Ultrasonic obstacle avoidance module parameters

- Working voltage: 4.5V~5.5V. In particular, it is absolutely not allowed to exceed 5.5V.
- Power consumption current: minimum 1mA, maximum 20mA
- Resonant frequency: 40KHz;
- Detection range: 4 mm to 4 m. Error: 4%;
  (In particular, the detection of the closest distance is 4mm, the maximum distance is 4 meters, the data is continuously output, no setting is required.)
- Operating temperature range: 0°C to +100°C;
- Dimensions: 46mm*21mm*20mm(H);
- Fixed hole size 2*Φ2mm Pitch: 42x16mm

4.8.3 RGB ultrasonic obstacle avoidance module works

The most commonly used HS-04 ultrasound at this stage is 4Pin. After our improvement, only 3PIN can be used for communication, and we have added RGB function to the ultrasound. The method of ultrasonic ranging is echo detection, that is, the ultrasonic transmitter emits ultrasonic waves in a certain direction. At the same time as the emission time, the timer starts to count, and the ultrasonic waves propagate in the air. When the obstacle is blocked on the way, it is reflected back immediately. When the receiver receives the reflected ultrasonic wave, it stops timing immediately. The working sequence diagram is shown in Figure 4.8.2. The propagation speed of the ultrasonic wave in the air is 340 m/s. According to the time t recorded by the timer, the distance s of the emission point from the obstacle surface can be calculated, that is, s=340t/2.

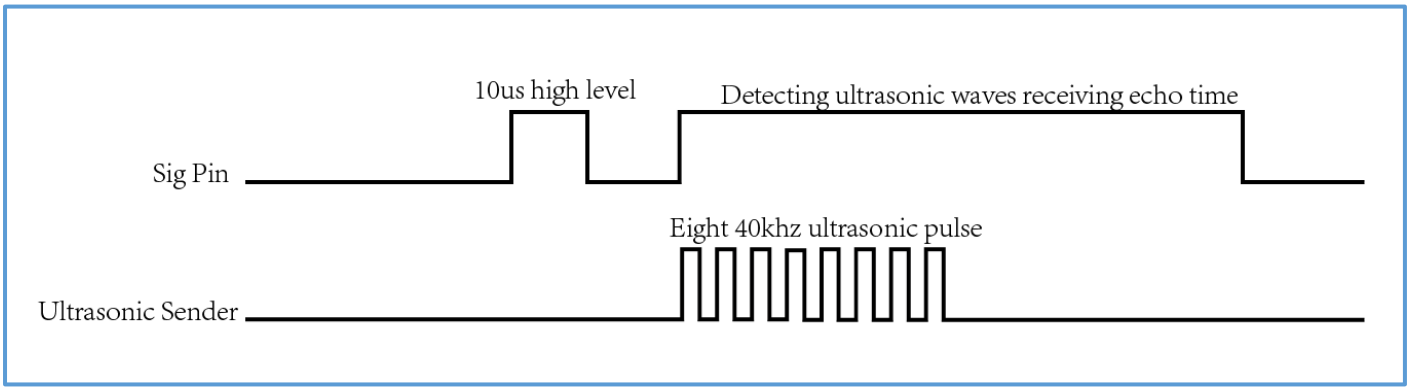


Figure 4.8.2 Ultrasonic working sequence diagram

Let's analyze this timing diagram. First, the ultrasonic ranging module is started by the trigger signal. That is to say, the host must first send a high level of at least 10us to trigger the ultrasonic module. The signal sent by the module is automatically responded by the sensor. We don't have to go. Take it. The output echo signal is what we need to pay attention to. The high level of the signal output is the time it takes for the ultrasonic wave to be sent back to reception. With a timer, you can record this time and calculate the distance. Don't forget that the result is divided by 2, because the total time is the sum of the time of transmission and reception.

Since the ultrasonic wave is also a kind of sound wave, its sound velocity V is related to temperature. In use, if the temperature of the propagation medium does not change much, it can be approximated that the ultrasonic velocity is substantially constant during the propagation. If the accuracy of the ranging is very high, the measurement results should be numerically corrected by means of temperature compensation. After the speed of sound is determined, the distance can be determined by measuring the time of the ultrasonic round trip. This is the basic principle of ultrasonic ranging. As shown in Figure 4.8.3:

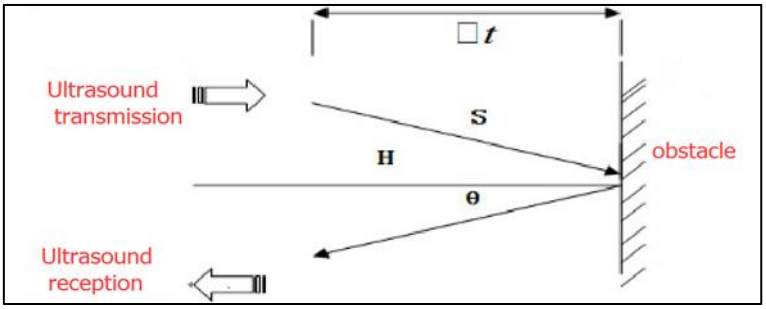


Figure 4.8.3 Principle of the distance measurement of ultrasonic waves

4.8.4 RGB Ultrasonic Obstacle Block Schematic

Ultrasound is mainly divided into two parts, one is the transmitting circuit and the other is the receiving circuit, as shown in Figure 4.8.4. The transmitting circuit is mainly composed of inverter SP232EN and ultrasonic transmitting transducer. The square wave signal of the output of 40 kHz of the Arduino port is

sent to one electrode of the ultrasonic transducer through the primary inverter, and the other is through the two-stage inverter. After being sent to the other electrode of the ultrasonic transducer, the square wave signal is applied to both ends of the ultrasonic transducer by this push form, and the emission intensity of the ultrasonic wave can be improved. The output uses two inverters in parallel to improve drive capability

The receiving circuit is composed of an ultrasonic sensor, a two-stage amplifying circuit and a phase-locked loop circuit. The reflected wave signal received by the ultrasonic sensor is very weak, and the two-stage amplifying circuit is used to amplify the signal received by the sensor. The phase-locked loop circuit sends an interrupt request to the microcontroller after receiving the signal with the frequency matching. The center frequency of the internal voltage controlled oscillator of the phase locked loop LM324 is , and the locking bandwidth is related to C15. Since the transmitted ultrasonic frequency is 40 kHz, the relevant components are adjusted to make the center frequency of the phase-locked loop 40 kHz, and only the signal of the frequency is responded, thereby avoiding interference of other frequency signals.

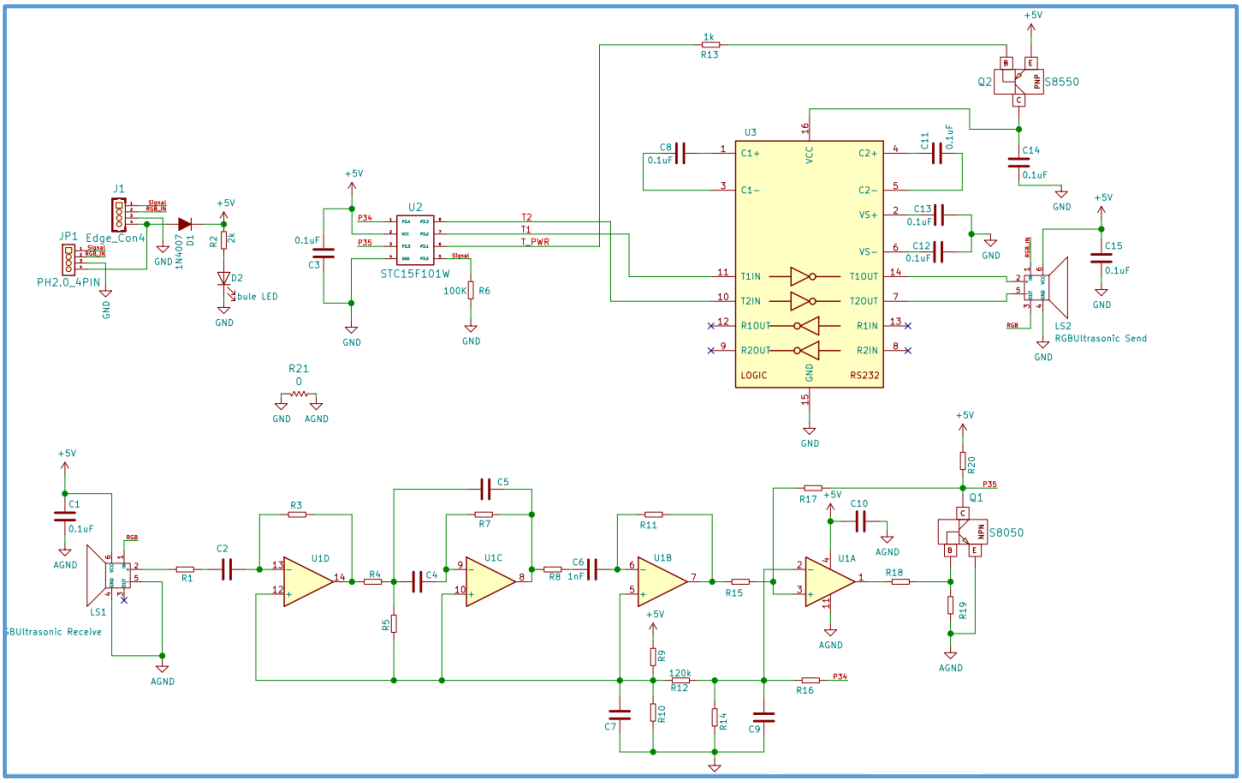


Figure 4.8.4 Schematic diagram of ultrasonic transmission and reception

When the ultrasonic sensor receives the ultrasonic signal, it is sent to the two-stage amplifier for amplification, and the amplified signal enters the phase-locked loop detection. If the frequency is 40 kHz, the low-level interrupt request signal is sent from the 1-pin to the P3.5 terminal of the single-chip microcomputer. The Arduino Uno R3 stops the timer after detecting a low level.

4.8.5 Introduction to RGB LED Lights

WS2812B is a three-channel LED drive control dedicated circuit. The chip contains intelligent digital interface, data latch signal, shaping and amplifying drive circuit. It also includes high-precision internal oscillator and 15V high-voltage programmable constant current output driver. At the same time, in order to reduce the power supply ripple, the three channels have a certain delay conduction function, which can reduce the circuit ripple installation more easily during frame refresh.

Unlike the traditional RGB, the WS2812B integrates the WS2812B LED driver control chip, which can control one LED lamp bead or multiple LED modules with only one signal line. Its main features are as follows:

- Output port withstand voltage 15V.
- Chip built-in voltage regulator, 24V power supply only requires string resistor to IC VDD pin, no external voltage regulator
- Grayscale adjustment circuit (256 levels of grayscale adjustable);
- Built-in signal shaping circuit;
- Waveform shaping and output, to ensure that the line waveform distortion will not accumulate;
- Built-in power-on reset and power-down reset circuit;
- PWM control terminal can achieve 256 levels of adjustment, scanning frequency is not lower than 400Hz/s
- Serial interface, can receive and decode data through one signal line
- Any two points transmission distance of more than 10 meters without adding any circuit
- When the refresh rate is 30 frames/second, the low-speed mode cascade is not less than 512 points, the high-speed mode is not less than 1024 points, and the data transmission speed is 800Kbps mode.

4.8.6 Introduction to WS2812B Driver RGB LED Control Mode

The WS2812B data protocol adopts the single-line return-to-zero code communication mode. After the power-on reset, the DIN terminal accepts the data transmitted from the controller. The first 24 bits of data are extracted by the first pixel and sent to the pixel. The data latches, the remaining data is shaped and amplified by the internal shaping processing circuit, and then the output is forwarded to the next cascaded pixel through the DO port, and the signal is reduced by 24 bits every time a pixel is transmitted. The pixel adopts automatic shaping and forwarding technology, so that the number of cascaded pixels is not limited by signal transmission, and only the signal transmission speed is limited.

The data latch inside the chip generates different duty cycle control signals at the OUTR, OUTG, and OUTB control terminals according to the received 24bit data. When the DIN terminal inputs the RESET signal, all the chips synchronously send the received data to each. Segment, the chip will re-accept the new data after the end of the signal, after receiving the first 24bit data, the data port is forwarded through the DO port, the original output of the OUTR, OUTG, OUTB pins is maintained before the chip receives the RESET code. No change, when receiving a low level RESET code above 50μs, the chip will output the 24-bit PWM data pulse width just received to the OUTR, OUTG, OUTB pins. The chip pins and functions are shown in Figure 4.8.5 and Table 1. Shown

Serial number	symbol	Pin name	Function description
1	D0	LED drive	Display data cascade output
2	GND	Ground	Ground, finally cascaded
3	IN	LED drive	Display data input
4	VDD	Logic power	IC power supply

Table 1 WS2812B chip pin function table

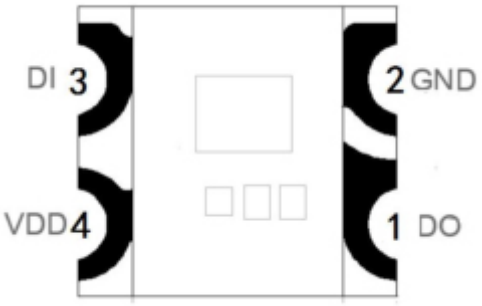


Figure 4.8.5 WS2812B pin diagram

The WS2812B low level is represented by T0, which is 0.5μs high level 2μs low level. T1 consists of a low level of 2μs and a low level of 0.5μs. Low level time above 50μs during reset

T0H	0 yards, high voltage time	0. 5μs
T1H	1 yard, high voltage time	2μs
T0L	0 yards, low voltage time	2μs
T1L	1 yards, low voltage time	0. 5μs
RES	Reset code, low voltage time	>50μs

Timing waveform diagram is as follows:

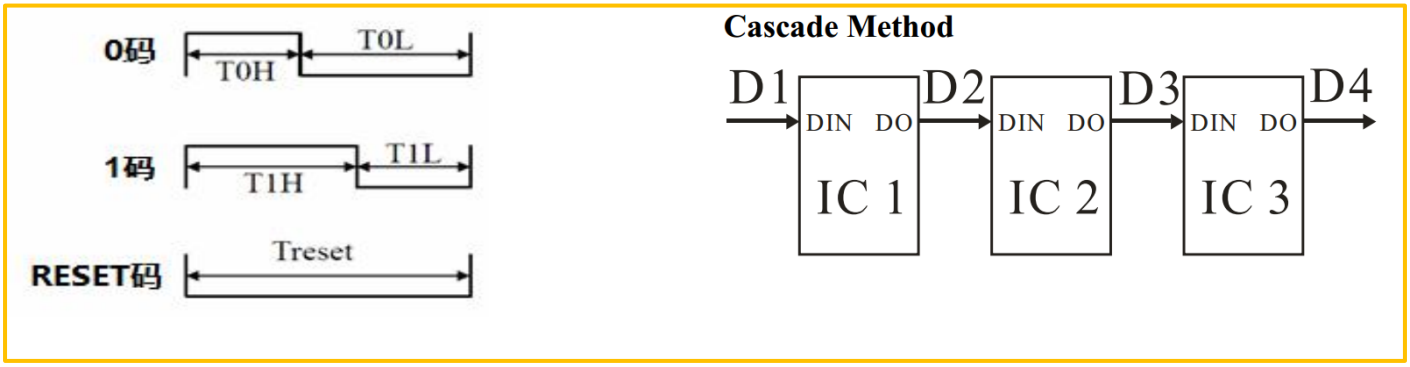


Figure 4.8.6 WS2812B Timing Diagram and Connection Method

24bit data structure:

R7	R6	R5	R4	R3	R2	R1	R0	G7	G6	G5	G4	G3	G2	G1	G0	B7	B6	B5	B4	B3	B2	B1	B0

Note: High-order first, send data in RGB order

4.8.7 Connection Diagram of RGB Ultrasonic Obstacle Avoidance Module and Arduino Expansion Board

RGB Ultrasoni	Arduino UNO
VCC	VCC
GND	GND
SIG	3
RGB	2

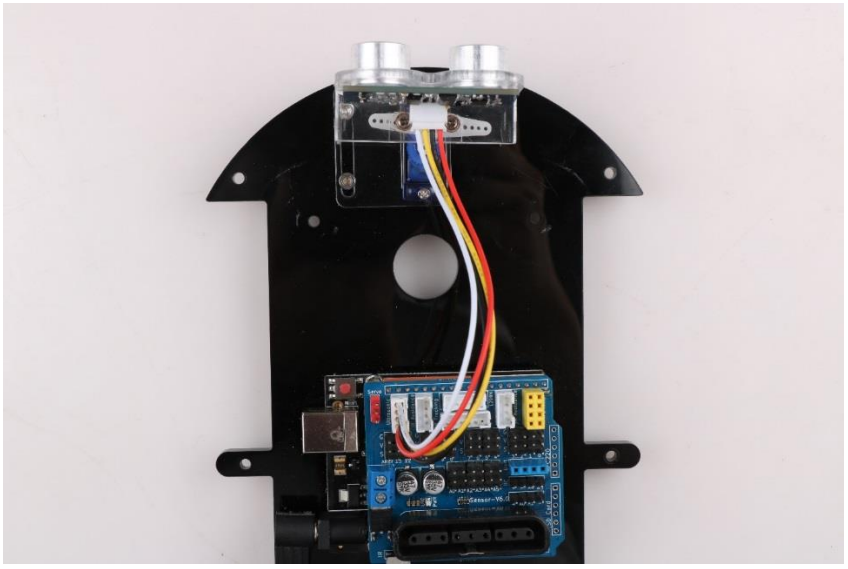


Figure 4.8.7 Schematic diagram of RGB ultrasonic obstacle avoidance module wiring

4.8.8 RGB ultrasonic obstacle avoidance module test experiment

1) Install the steering gear and ultrasonic on the trolley and connect it correctly.

2) Connect the main control board of the car and the computer via USB;

3) Openin the disc data“Lesson\Module_Test\RGBUltrasonicModule_Test\RGBUltrasonicModule_Test.ino”

4) Burn the RGB ultrasonic obstacle avoidance module test program to Arduino UNO R3

5) Open the serial monitor;

6) Align the RGB ultrasonic obstacle avoidance module with a plane obstacle, and then observe the serial port monitor to see the distance between the RGB ultrasonic obstacle avoidance module and the obstacle continuously on the serial monitor (Figure 4.8.8); At the same time, the RGB LED on the RGB ultrasonic obstacle avoidance module will illuminate. This proves that the RGB ultrasonic obstacle avoidance module works normally.

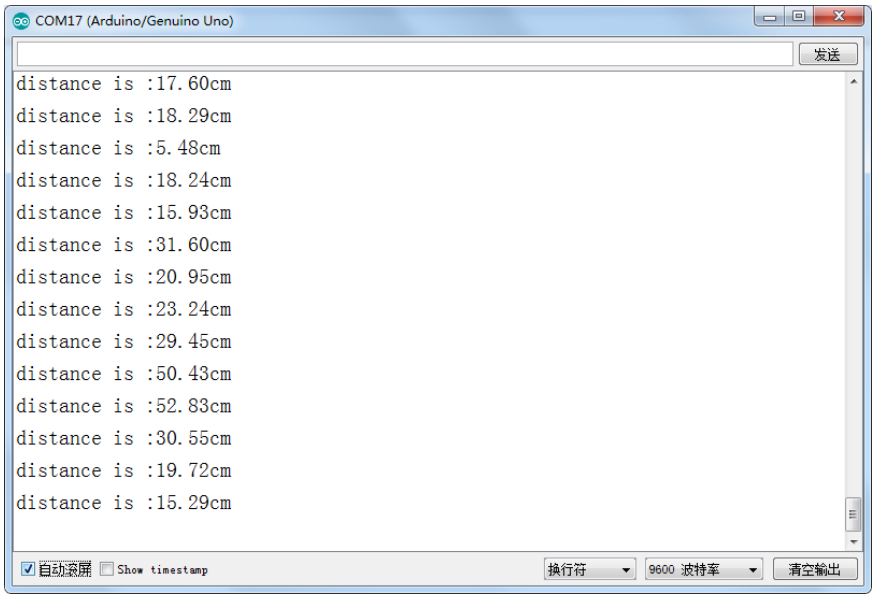


Figure 4.8.8 Serial monitor prints the distance between the RGB ultrasonic obstacle avoidance module and the obstacle

```
#include "RGBLed.h"
#define RGB_RED     0xFF0000
#define RGB_GREEN   0x00FF00
#define RGB_BLUE    0x0000FF
#define RGB_YELLOW  0xFFFF00
#define RGB_PURPLE  0xFF00FF
#define RGB_WHITE   0xFFFFFF
```

```
const int SingPin = 3;
const int RgbPin = 2;
float distance;
RGBLed mRgb(RgbPin,6);

void setup() {
  Serial.begin(9600);
  Serial.println("Ultrasonic sensor:");
  mRgb.setColor(1,RGB_RED);
  mRgb.setColor(2,RGB_GREEN);
  mRgb.setColor(3,RGB_BLUE);
  mRgb.setColor(4,RGB_YELLOW);
  mRgb.setColor(5,RGB_PURPLE);
  mRgb.setColor(6,RGB_WHITE);
  mRgb.show();
}
void loop() {
  digitalWrite(SingPin, LOW);
  delayMicroseconds(2);
  digitalWrite(SingPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(SingPin, LOW);
  pinMode(SingPin, INPUT);
  distance = pulseIn(SingPin, HIGH) / 58.00;
  Serial.print("distance is :");
  Serial.print(distance);
  Serial.print("cm");
  Serial.println();
  delay(1000);
}
```

4. 9 Hummer-Bot RGB ultrasonic obstacle avoidance function experiment

4.9.1 RGB ultrasonic obstacle avoidance function software logic

In this product, we integrate the ultrasonic module and the steering gear to make the two parts work at the same time, which greatly increases the flexibility and data validity of the car. The main work flow is: after power-on, the steering gear automatically turns to 90 degrees. The single-chip computer reads the data

value returned by the ultrasonic wave. If the value is greater than the set safety value, the trolley continues to move forward, and vice versa. At this time, the steering gear first rotates 90 degrees to the right, and the single-chip computer reads the ultrasonic wave again. The data, then, the servo is rotated 180 degrees to the left, the MCU reads the ultrasonic data, and then the servo returns to 90 degrees. The MCU compares the two detected data. If the left is greater than the right, the car turns and then drives. To the right, if the data on both the left and the right are less than the safe value, the car will turn around and the flow chart is shown in Figure 4.9.1.

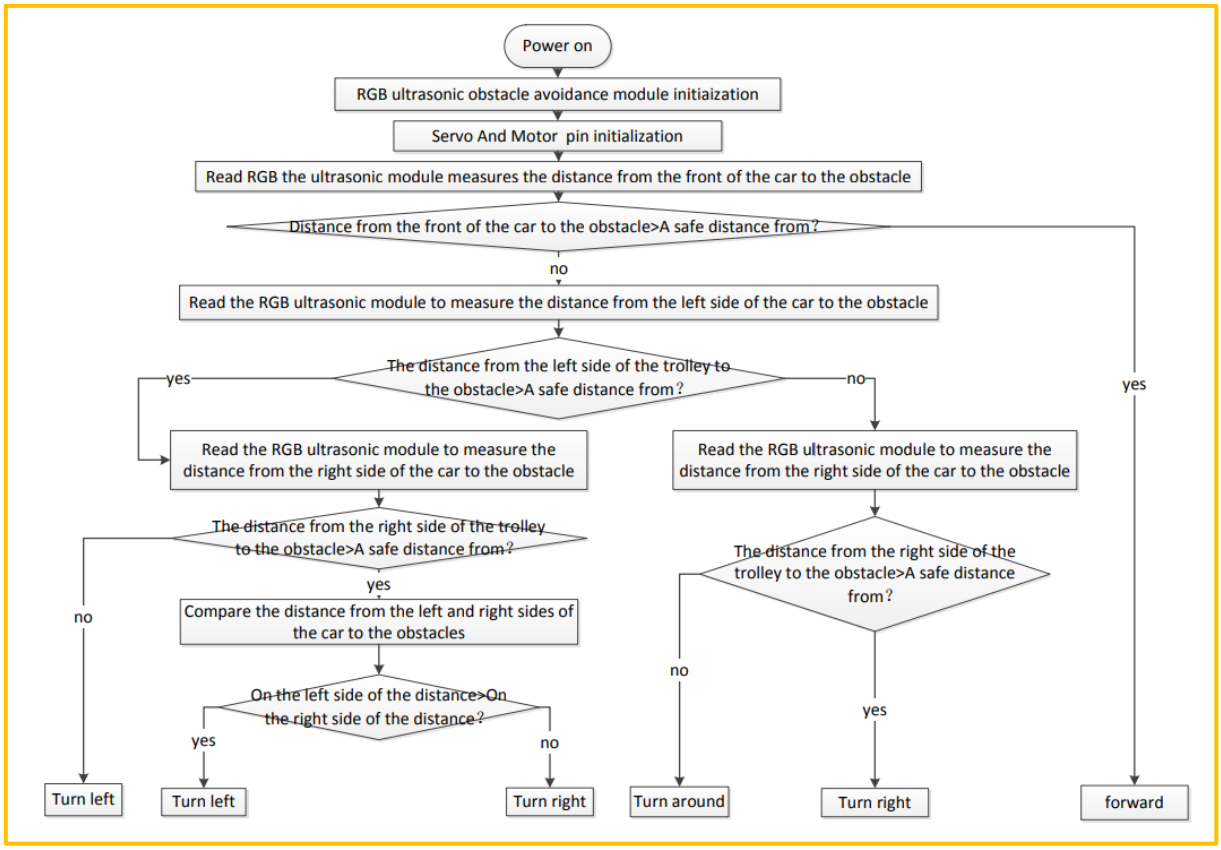


Figure 4.9.1 Ultrasonic obstacle avoidance function program execution flow chart

4.9.2 RGB ultrasonic obstacle avoidance module and steering gear wire connection

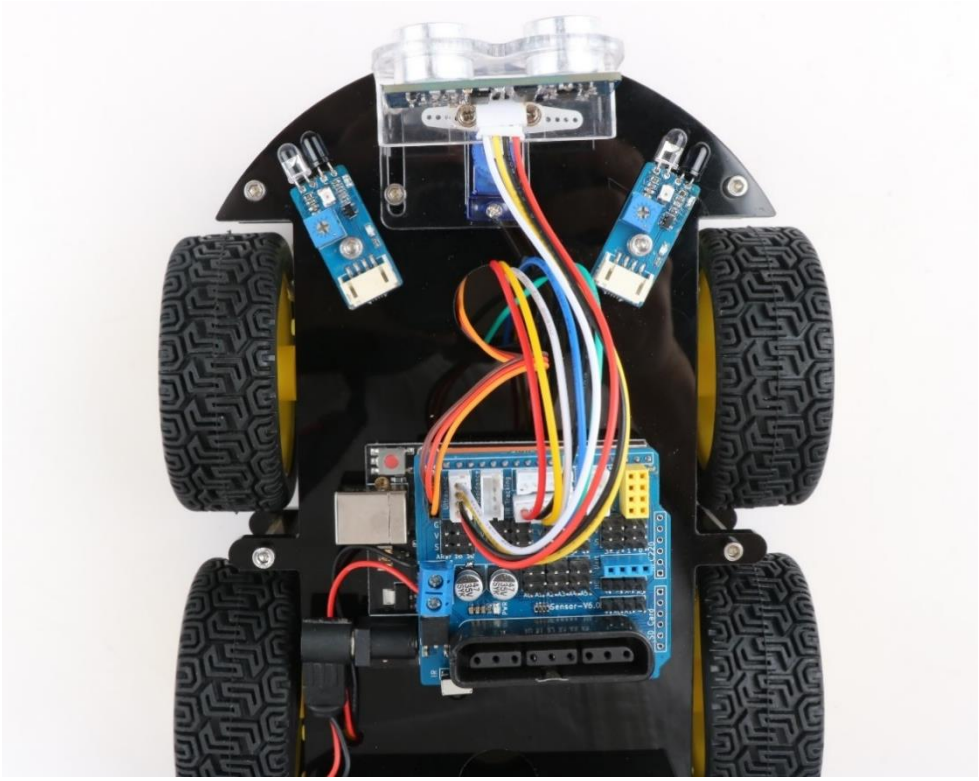


Figure 4.9.2 Wiring diagram of servo, RGB ultrasonic obstacle avoidance module and expansion board

4.9.3 Hummer-Bot RGB Ultrasonic Obstacle Avoidance Function Experimental

Procedure

1) Openin the disc data “Lesson\Advanced Experiment\UltrasonicAvoidance_Function\ UltrasonicAvoidance_Function.ino”
2) Burn the UltrasonicAvoidance_Function.ino program to the Arduino UNO R3 main control board;
3) Turn on the power and observe the progress of the car.

By observing the results of the obstacle avoidance procedure of the car, we will look at the complete program to deepen our understanding.

```
#include "RgbUltrasonic.h"
#define IN2_PIN 10 // PWMB
#define IN1_PIN 6  // DIRB  ---  right
#define IN4_PIN 9  // PWMA
#define IN3_PIN 5  // DIRA  ---  left
#define SERVO_PIN 13
#define SING_PIN 3
```

```
#define RGB_PIN 2
#define UL_LIMIT_MID 20
#define UL_LIMIT_MAX 550
RgbUltrasonic mRgbUltrasonic(SING_PIN, RGB_PIN, SERVO_PIN);/*Define ultrasonic and
servo pins*/

void setup()
{
  Serial.begin(9600);
  pinMode(IN1_PIN, OUTPUT);
  pinMode(IN2_PIN, OUTPUT);
  pinMode(IN3_PIN, OUTPUT);
  pinMode(IN4_PIN, OUTPUT);
  mRgbUltrasonic.SetServoBaseDegree(90);/*Adjust the initial angle of the steering
gear according to the steering gear error*/
  mRgbUltrasonic.SetServoDegree(90);/*Set the servo angle*/
}

void loop()
{
  uint16_t FrontDistance, LeftDistance, RightDistance;
  FrontDistance = mRgbUltrasonic.GetUltrasonicFrontDistance();/*The ultrasonic module
collects the front data*/
  delay(50);
  if ((FrontDistance > UL_LIMIT_MID) && (FrontDistance < UL_LIMIT_MAX))
  /*According to the data collected by the ultrasonic module and the infrared
obstacle avoidance module,
    it is judged whether there is an obstacle in front, and if there is no obstacle,
go straight.*/
  {
    analogWrite(IN1_PIN, 200);
    analogWrite(IN2_PIN, LOW);
    analogWrite(IN3_PIN, LOW);
    analogWrite(IN4_PIN, 200);
  }
  else if ((FrontDistance < UL_LIMIT_MID) || (FrontDistance > UL_LIMIT_MAX))
  /*According to the data collected by the ultrasonic module and the infrared
obstacle avoidance module, it is determined whether there is an obstacle in front.
For example,
```

```
   * the infrared obstacle avoidance module determines that there is no obstacle in
front, and the ultrasonic module determines that the right obstacle is an obstacle,
   * first stops the car, and uses the ultrasonic module to perform left and right.*/
  {
    analogWrite(IN1_PIN, 0);
    analogWrite(IN2_PIN, 0);
    analogWrite(IN3_PIN, 0);
    analogWrite(IN4_PIN, 0);
    RightDistance = mRgbUltrasonic.GetUltrasonicRightDistance();/*The ultrasonic
module collects the right side*/
    LeftDistance = mRgbUltrasonic.GetUltrasonicLeftDistance();/*The ultrasonic module
collects the left side*/


    delay(10);
    if ((RightDistance > UL_LIMIT_MID) && (RightDistance < UL_LIMIT_MAX) &&
(RightDistance > LeftDistance))
    /*According to the ultrasonic module to collect the data on the left and right
sides to determine whether there is an obstacle on the right side.*/
    {
      analogWrite(IN1_PIN, LOW);
      analogWrite(IN2_PIN, 200);
      analogWrite(IN3_PIN, LOW);
      analogWrite(IN4_PIN, 200);
      Serial.println("testRight");
      delay(380);
    }
    else if ((LeftDistance > UL_LIMIT_MID) && (LeftDistance < UL_LIMIT_MAX) &&
(LeftDistance > RightDistance))
   /*According to the ultrasonic module to collect the data on the left and right
sides to determine whether there is an obstacle on the left side.*/
    {
      analogWrite(IN1_PIN, 200);
      analogWrite(IN2_PIN, LOW);
      analogWrite(IN3_PIN, 200);
      analogWrite(IN4_PIN, LOW);
      Serial.println("testLeft");
      delay(380);
    }
```

```
    else if ((RightDistance < UL_LIMIT_MID) && (LeftDistance < UL_LIMIT_MID) )
    /*According to the ultrasonic module to collect the data on the left and right
sides to determine whether there are obstacles on the left and right sides*/
    {
      analogWrite(IN1_PIN, 200);
      analogWrite(IN2_PIN, 0);
      analogWrite(IN3_PIN, 200);
      analogWrite(IN4_PIN, 0);
      delay(760);
    }
  }
}
```

In the previous sections, we mainly explained the "automatic driving" of the car, and the front part is some experiments to avoid obstacles. Only the car itself is walking around, it seems to have nothing to do with us, feel and Not being involved, it is very lacking in fun, so in the next few sections, we will control the car from other aspects, so that everyone can control the car and play the track that they want, then we will The "Infrared Remote Control" section begins, followed by "Mobile Bluetooth Control" and finally "2.4G Handset Remote Control".

Openin the disc data “Lesson\Comprehensive Experiment\Hummerbot_UltrasonicFollow\Hummerbot_UltrasonicFollow”

4.9.4 RGB ultrasonic follow

In order for the car to follow the hand movement, it is necessary to determine the movement mode of the car by judging the distance value measured by the ultrasonic module. The distance value of the ultrasonic module becomes smaller, indicating that the hand is close, and the small car needs to be retracted when the trail is a certain value; when the distance value of the ultrasonic module becomes large, the hand is far away, and the trolley needs to advance when it is large. When the distance of the ultrasonic module is within a range, we can stop the car. Let's take a look at the program below.

Openin the disc data “Lesson\Comprehensive Experiment\Hummerbot_UltrasonicFollow\Hummerbot_UltrasonicFollow.ino”

```
#include "Hummerbot.h"
#include "BluetoothHandle.h"
#include "ProtocolParser.h"
```

```
#include "KeyMap.h"
#include "debug.h"

#define IN1_PIN 6   // PWMB
#define IN2_PIN 10  // DIRB  ---  right
#define IN3_PIN 5   // DIRA  ---  left
#define IN4_PIN 9   // PWMA

#define SERVO_PIN 13
#define UL_SING_PIN 3
#define UL_RGB_PIN 2

ProtocolParser *mProtocol = new ProtocolParser();
Hummerbot hbot(mProtocol, IN1_PIN, IN2_PIN, IN3_PIN, IN4_PIN);
byte Ps2xStatus, Ps2xType;

void setup()
{
  Serial.begin(9600);
  hbot.init();
  hbot.SetControlMode(E_ULTRASONIC_FOLLOW_MODE);
  hbot.SetRgbUltrasonicPin(UL_SING_PIN, UL_RGB_PIN, SERVO_PIN);
  hbot.SetSpeed(0);
  hbot.mRgbUltrasonic->SetServoBaseDegree(90);
  hbot.mRgbUltrasonic->SetServoDegree(90);
}

void UltrasonicFollow()
{
  hbot.SetSpeed(40);
  uint16_t UlFrontDistance =  hbot.GetUltrasonicValue(FRONT);
  delay(10);
  if (UlFrontDistance < 13) {
    hbot.GoBack();
  } else if (UlFrontDistance > 16) {
    hbot.GoForward();
  } else if (13 <= UlFrontDistance <=16) {
    hbot.KeepStop();
    }
```

```
}
void loop()
{
  switch (hbot.GetControlMode()) {
    case E_ULTRASONIC_FOLLOW_MODE:
      UltrasonicFollow();
      break;
    default:
      break;
  }
  switch (hbot.GetStatus()) {
    case E_FORWARD:
      hbot.SetRgbColor(E_RGB_ALL, RGB_WHITE);
      break;
    case E_LEFT:
      hbot.SetRgbColor(E_RGB_LEFT, RGB_WHITE);
      break;
    case E_RIGHT:
      hbot.SetRgbColor(E_RGB_RIGHT, RGB_WHITE);
      //   Mirage.Sing(S_OhOoh);
      break;
    case E_BACK:
      hbot.SetRgbColor(E_RGB_ALL, RGB_RED);
      break;
    case E_STOP:
      hbot.SetRgbColor(E_RGB_ALL, RGB_OFF);
      break;
    case E_SPEED_UP:
      hbot.SetRgbColor(E_RGB_ALL, hbot.GetSpeed() * 2.5);
      break;
    case E_SPEED_DOWN:
      hbot.SetRgbColor(E_RGB_ALL, hbot.GetSpeed() * 2.5);
      break;
    default:
      break;
  }
}
```

4.10 Infrared remote control test

4.10.1 Introduction to infrared remote control

The infrared wireless remote control kit consists of a Mini ultra-thin infrared remote control and a 38KHz infrared receiver. The Mini ultra-thin infrared remote control has 17 function keys and a transmission range of up to 8 meters. It is very suitable for remote control of various devices indoors. The infrared receiving module can receive the standard 38KHz modulated remote control signal. By programming the Arduino UNO R3, the decoding of the remote control signal can be realized, so that various remote control robots and interactive works can be produced. The kit is shown in Figure 4.10.1.

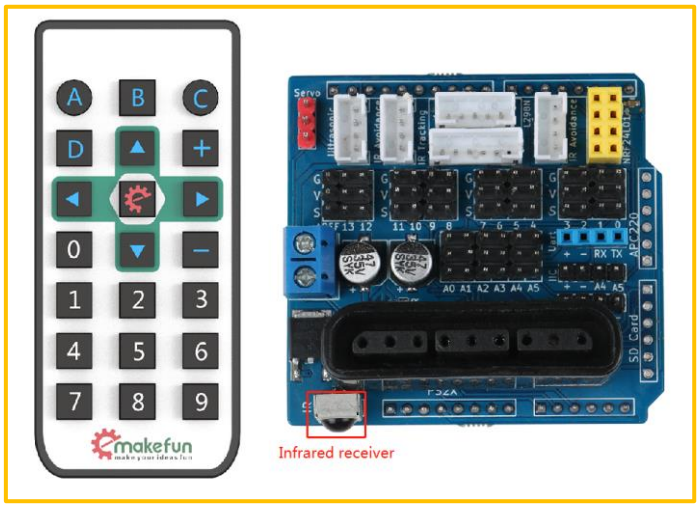


Figure 4.10.1 Infrared remote control kit physical map

The common infrared remote control system is mainly divided into three parts: modulation, transmission and reception. The infrared remote control transmitting part is mainly composed of a keyboard matrix, a remote control ASIC, an exciter and an infrared light emitting diode. The remote control ASIC is the core part of the transmitting system, and its internal part is composed of an oscillating circuit, a timing circuit, a scanning signal generator, a key input encoder, a command decoder, a user code converter, a digital modulating circuit and a buffer amplifier. It can generate the key scan pulse signal, and can decode the key code of the button, and then obtain the remote command of the key (remote control code pulse) through the remote command encoder, pulse amplitude modulation by the carrier of 38KHZ, and the remote command The modulated signal excites the infrared diode to emit an infrared remote control signal.

In the infrared receiver, a photoelectric conversion device (generally a photodiode or a phototransistor, which we use here is a PIN photodiode) converts the received infrared light command signal into a corresponding electrical signal. Since the received signal is very weak and the interference is particularly

large, in order to achieve accurate detection and conversion of the signal, in addition to the high-performance infrared photoelectric conversion device, it is also reasonable to select and design a circuit form with good performance. The most commonly used photoelectric conversion device is a photodiode. When the photosensitive surface of the photodiode PN junction is exposed to light, the semiconductor material of the PN junction absorbs light energy and converts the light energy into electrical energy. When a reverse voltage is applied to the photodiode, the reverse current in the diode will change as the incident light intensity changes. The greater the irradiance of the light, the greater the reverse current. That is to say, the reverse current of the photodiode changes with the frequency of the incident light pulse.

In the Hummer-Bot trolley, the integrated infrared receiver has three pins, including the power supply pin, grounding and signal output pins. Its circuit is shown in Figure 4.10.1. The ceramic capacitor 104 is a decoupling capacitor that filters out interference from the output signal. The 1st end is the output end of the demodulated signal, which is directly connected to the No. 2 port of the Arduino of the single chip microcomputer. When there is an infrared coded signal transmission, after being processed by the infrared connector, the output is a square wave signal after the detection and shaping, and is directly supplied to the single chip microcomputer, and the corresponding operation is performed to achieve the purpose of controlling the motor.

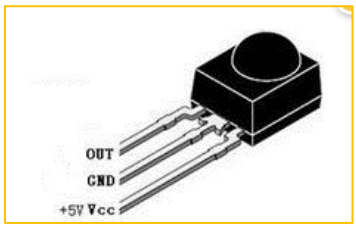


Figure 4.10.2 Infrared Receiver Pin Diagram

4.10.2 Working principle of infrared remote control

The remote control system generally consists of a remote control (transmitter) and a receiver. When you press any button on the remote control, the remote control generates a corresponding code pulse to output various infrared-based control pulse signals. The pulse is a computer command code. The infrared monitor diode monitors the infrared signal and then sends the signal to the amplifier and the limiter. The limiter controls the pulse amplitude to a certain level regardless of the distance between the infrared transmitter and the receiver. The AC signal enters the bandpass filter. The bandpass filter can pass the load wave of 30KHZ to 60KHZ, enter the comparator through the demodulation circuit and the integration circuit, and the comparator outputs high and low levels to restore the signal waveform of the transmitter. As shown in the system block diagram of 4.10.3.

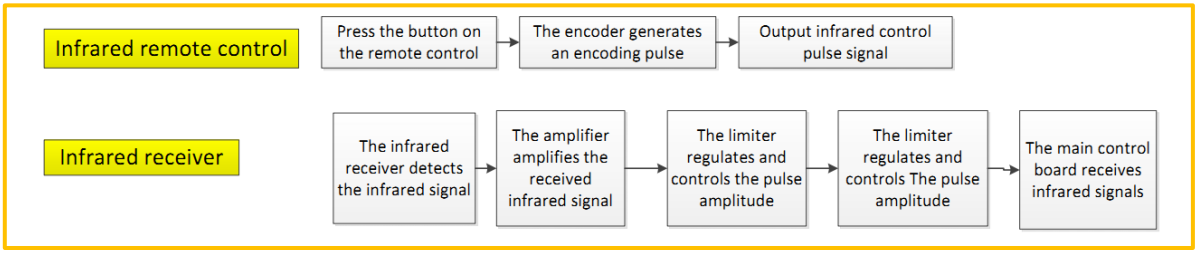


Figure 4.10.3 Infrared remote control system block diagram

4.10.3 Infrared remote control test experimental steps

1) Connect the main control board of the car and the computer via USB;

2) Openin the disc data“Lesson\Module_Test\IrkeyPressed_Test\IrkeyPressed_Test.ino”

3) Burn the IrkeyPressed_Test.ino program to Arduino UNO R3;

4) Open the serial monitor;

5) Turn on the battery switch;

6) Press the button on the infrared remote control against the car to observe the value printed on the serial port monitor (as shown in Figure 4.10.4). If the value of the button on the infrared remote control is the same as the value printed on the serial port monitor. , the infrared remote control communication is normal.

Infrared remote test program:

```
#include "IRremote.h"
IRremote ir(12);
unsigned char keycode;
char str[128];
void setup() {
    Serial.begin(9600);
    ir.begin();
}
void loop()
{
    if (keycode = ir.getCode()) {
        String key_name = ir.getKeyMap(keycode);
        sprintf(str, "Get ir code: 0x%x key name: %s \n", keycode, (char
*)key_name.c_str());
        Serial.println(str);
    } else {
      //   Serial.println("no key");
    }
    delay(110);
}
```

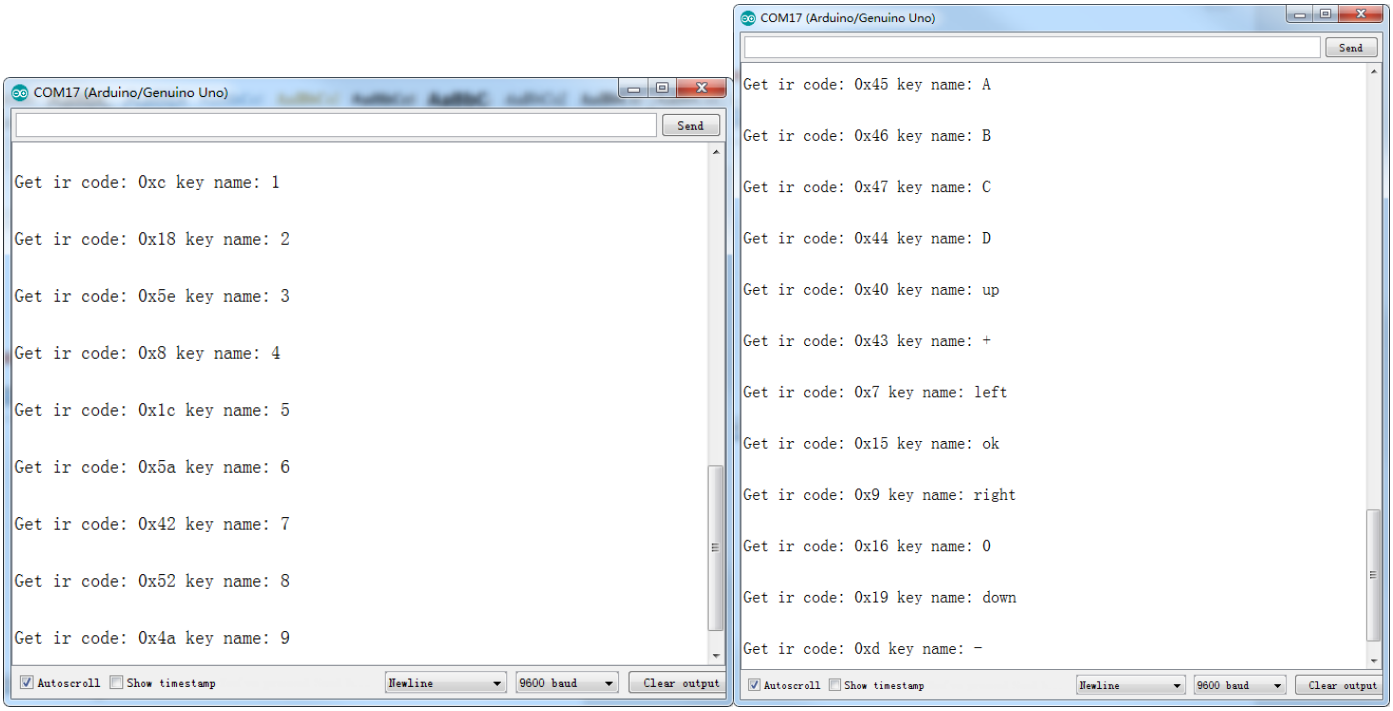


Figure 4.10.4 Remote Control Code Query

In Figure 4.10.4, we can see the two values of Ir Code “0xc” and keyname “1”, where “0x45” is the code of a button on the remote control and “1” is the name of the button function of the remote control.

```
ST_KEY_MAP irkeymap[KEY_MAX] = {
    {"A", 0x45},
    {"B", 0x46},
    {"C", 0x47},
    {"D", 0x44},
    {"up", 0x40},
    {"+", 0x43},
    {"left", 0x07},
    {"ok", 0x15},
    {"right", 0x09},
    {"0", 0x16},
    {"down", 0x19},
    {"-", 0x0d},
    {"1", 0x0c},
    {"2", 0x18},
    {"3", 0x5e},
    {"4", 0x08},
    {"5", 0x1c},
```

```
    {"6", 0x5A},
    {"7", 0x42},
    {"8", 0x52},
    {"9", 0x4A}
};
```

4.11 Hummer-Bot infrared remote control function experiment

4.11.1 Hummer-Bot infrared remote control function software logic

Infrared remote control, as the name suggests, is to remotely control the Hummer-Bot car through the remote control. How does it make Hummer-Bot "obedient" through a small remote control? This is the problem we need to explore.

To make Hummer-Bot "obeying", you first need Hummer-Bot to hear the commands of the remote control all the time, that is, both must be within a certain communication range, and secondly, Hummer-Bot can "understand". The remote control says that you can understand the meaning of each instruction. Finally, Hummer-Bot controls the limbs (four wheels) according to the instructions of the remote control to do the corresponding action. This is the basic principle of the realization of the infrared remote control function. The software logic is shown in Figure 4.11.1.

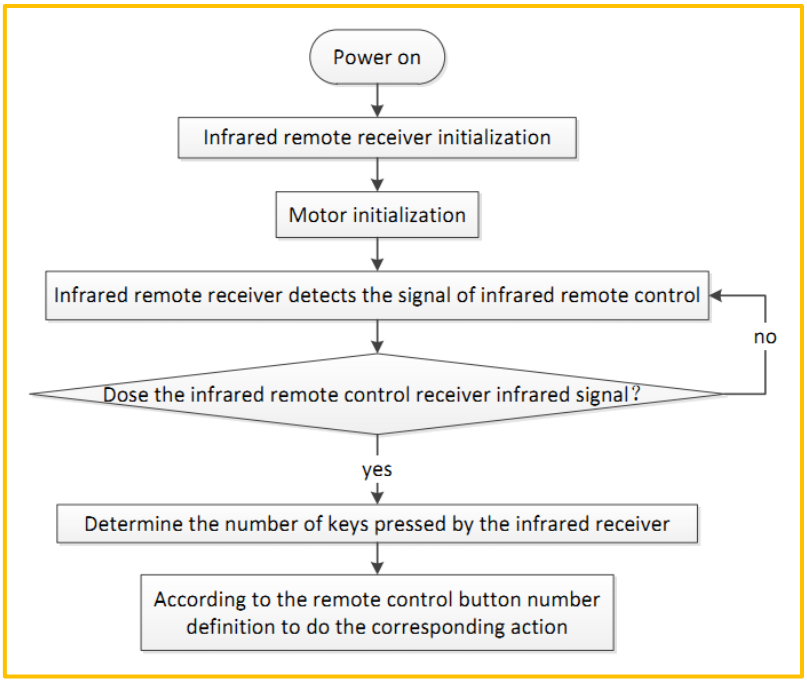


Figure 4.11.1 Flow chart of the infrared remote control software logic

4.11.2 Hummer-Bot RGB ultrasonic obstacle avoidance function experimental steps

1) Openin the disc data "Lesson\Advanced Experiment\IR_Control_Function\IR_Control_Function.ino"

2) Burn the IR_Control_Function .ino program to the Arduino UNO R3 main control board;

3) Turn on the power and press the function button defined on the remote control (as shown in Figure 4.11.2) to observe the progress of the car.

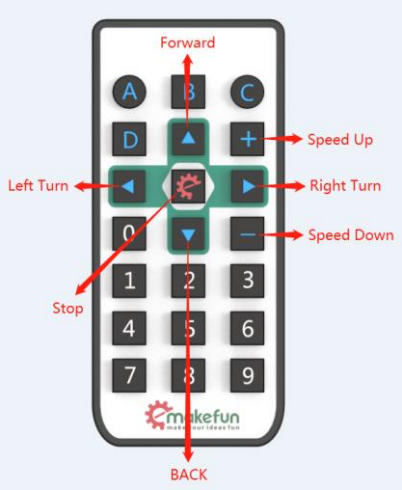


Figure 4.11.2 Infrared remote control button function definition

By observing the results of the obstacle avoidance procedure of the car, we will look at the complete program to deepen our understanding.

```
#include "IRremote.h"

#define IN1_PIN 6   // PWMB
#define IN2_PIN 10  // DIRB  ---  right
#define IN4_PIN 9   // PWMA
#define IN3_PIN 5   // DIRA  ---  left

int RECV_PIN = 12;//Define the infrared receiver pin to 12

long expedite1 = 0x43;
long expedite2 = 0x0d;
long up = 0x40;
long down = 0x19;
long stop = 0x15;
long left = 0x07;
```

```
long right = 0x09;
static int val = 160;
IRremote irrecv(RECV_PIN);

void setup() {
  Serial.begin(9600);
  irrecv.begin();
}

void loop() {
  byte irKeyCode;
  if (irKeyCode = irrecv.getCode())
  {
    if (irKeyCode == up) {
      analogWrite(IN1_PIN, val);//the speed value of motorA is val
      analogWrite(IN2_PIN, LOW);
      analogWrite(IN3_PIN, LOW);
      analogWrite(IN4_PIN, val); //the speed value of motorA is val
    }
    else if (irKeyCode == expedite1) {
      val += 20;
      if (val >= 240)
      {
        val = 255;
      }
    }
    else if (irKeyCode == expedite2) {
      val -= 20;
      if (val <= 20)
      {
        val = 0;
      }
    }
    else if (irKeyCode == stop) {
      analogWrite(IN1_PIN, LOW);
      analogWrite(IN2_PIN, LOW);
      analogWrite(IN3_PIN, LOW);
      analogWrite(IN4_PIN, LOW);
    }
```

```
    else if (irKeyCode == left) {
      analogWrite(IN1_PIN, val);
      analogWrite(IN2_PIN, LOW); //the speed value of motorA is val
      analogWrite(IN3_PIN, val);
      analogWrite(IN4_PIN, LOW); //the speed value of motorA is val
    }
    else if (irKeyCode == right) {
      analogWrite(IN1_PIN, LOW);//the speed value of motorA is val
      analogWrite(IN2_PIN, val);
      analogWrite(IN3_PIN, LOW);//the speed value of motorA is val
      analogWrite(IN4_PIN, val);
    }
    else if (irKeyCode == down) {
      analogWrite(IN1_PIN, LOW);
      analogWrite(IN2_PIN, val); //the speed value of motorA is val
      analogWrite(IN3_PIN, val); //the speed value of motorA is val
      analogWrite(IN4_PIN, LOW);
    }
  } else {
    analogWrite(IN1_PIN, LOW);
    analogWrite(IN2_PIN, LOW); //the speed value of motorA is 0
    analogWrite(IN3_PIN, LOW);
    analogWrite(IN4_PIN, LOW); //the speed value of motorB is 0
  }
}
```

The above program is a reference routine for the infrared remote control function.

4.12 JDY-16 Bluetooth Module Test Experiment

4.12.1 Introduction to Bluetooth Module

The Bluetooth module used by Hummer-Bot to support the mobile phone Bluetooth APP remote control function is JDY-16 BLE (Bluetooth Low Energy).

JDY-16 Bluetooth transparent transmission module is based on Bluetooth 4.2 protocol standard, working frequency band is 2.4GHZ range, modulation mode is GFSK, maximum transmission power is 0db, maximum transmission distance is 80 meters, adopts imported original chip design, supports users to modify by AT command The device name, service UUID, transmit power, pairing password and other instructions

are convenient and quick to use. For module information, see JDY-16 Bluetooth Module\JDY-16 Bluetooth 4.2 Module (JDY-16-V1.9). The physical diagram of the Bluetooth module is shown in Figure 4.12.1:

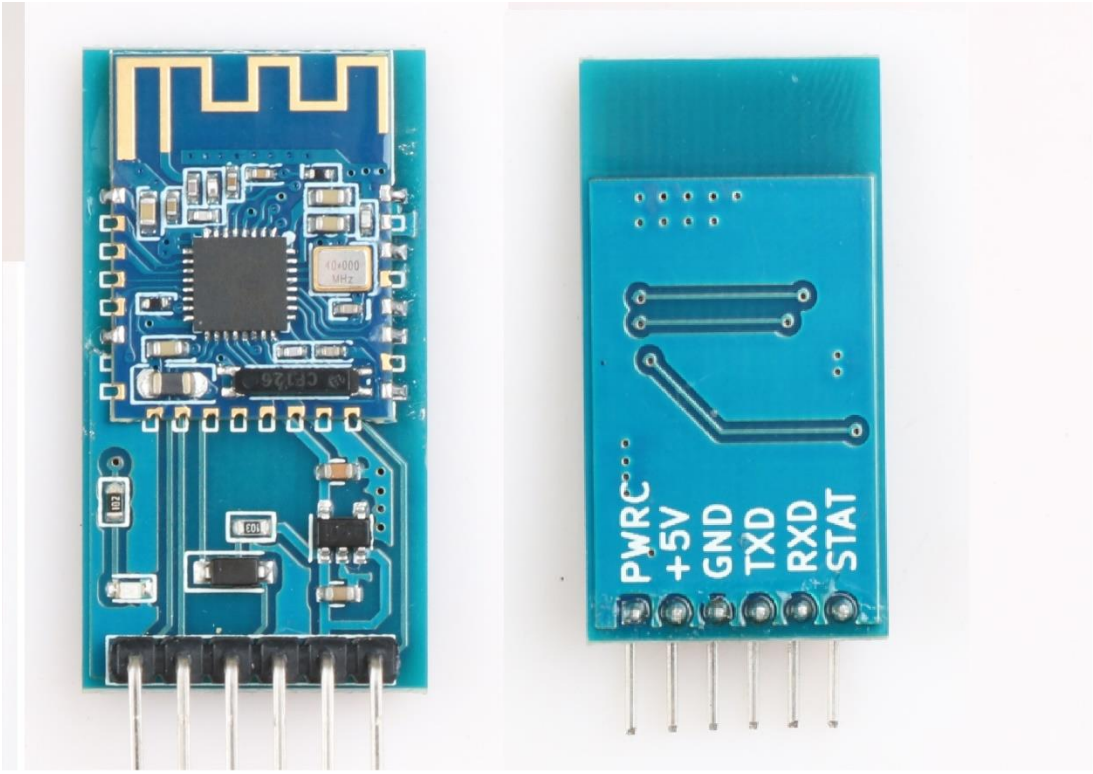


Figure 4.12.1 JDY-16 Bluetooth Module Physical Map

4.12.2 JDY-16 Bluetooth Module Parameters

1) BLE high-speed transparent transmission supports 8K byte rate communication.

2) Send and receive data without byte limit, support 115200 baud rate to continuously send and receive data.

3) Supports 3 working modes (please refer to the description of AT + STARTEN command function).

4) Support the Bluetooth module to be inserted into the expansion board and download the arduino program.

5) Support WeChat Airsync, WeChat applet and APP communication.

6) Support 4-channel IO port control.

7) Support high precision RTC clock.

8) Support PWM function (can be controlled by UART, IIC, APP, etc.).

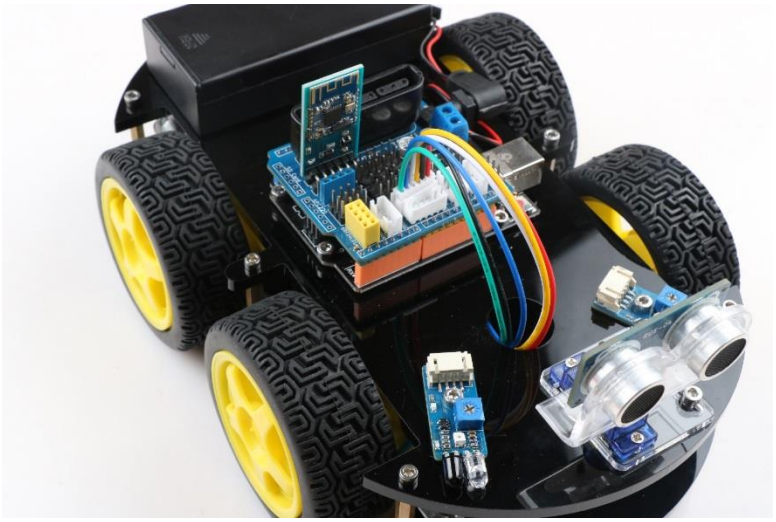
9) Support UART and IIC communication mode, the default is UART communication.

10) iBeacon mode (supports WeChat shake protocol and Apple iBeacon protocol).

11) Host transparent transmission mode (data transmission between application modules, host and slave communication).

4.12.3 Bluetooth module test experiment steps

1) Open the supporting materials, find the folder JDY-16 Bluetooth module\Bluetooth test APP installation package and open it. Install the test software "BLETestToos.apk" in the directory to the mobile phone (currently only supported on Android mobile phones)
2) Openin the disc data"Lesson\Module_Test\BleModule_Test\BleModule_Test.ino"
3) Burn the BleModule_Test.ino program to the Arduino UNO R3; align the serial port pins of the Bluetooth module with the mother socket on the Arduino-UNO r3 main board and insert it; connect the Bluetooth module to the figure shown in Figure 4.12.2 The method is as follows: the VCC port of the JDY-16 Bluetooth module is connected to the positive pole of the 5V DC power supply, the GND port is connected to the negative pole of the power supply, the RXD port is connected to the TXD port of the Arduino expansion board, and the TXD port is connected to the RXD port of the Arduino expansion board.



JDY-16	Arduino UNO
VCC	VCC
GND	GND
TXD	RXD
RXD	TXD

Figure 4.12.2 Bluetooth connection location diagram

4) Turn on the power and test that the blue indicator on the Bluetooth module starts to flash;
5) Open the test app, as shown in Figure 4.12.3. Find the corresponding Bluetooth name (JD-16) and click to connect. As shown in Figure 4.12.4, there will be three options for testing different functions, because here we only test whether Bluetooth can be normal. Send and receive data, so we choose SK Service, click to enter Figure 4.12.5.

Figure 4.12.3

Figure 4.12.4

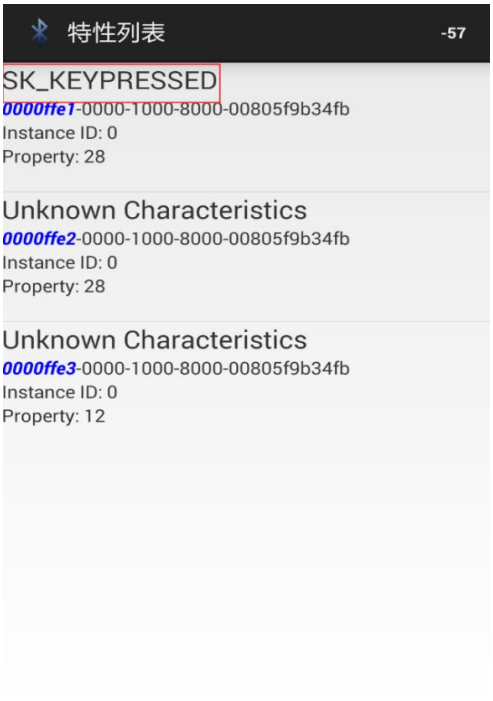


Figure 4.12.5

Figure 4.12.6

4、In Figure 4.12.6, there are three options, we choose "SK-KEYPRESSED", click on Figure 4.12.6, we can see that there is a "write" button, click to enter The interface shown in Figure 4.12.7, in Figure 4.12.7, we click on the “red box” to enter the data you want to send. After the input is completed, click “Send” to send the data, as shown in Figure 4.12.8.

Figure 4.12.7

Figure 4.12.8

5、After clicking and sending, we can see that the content sent by the mobile phone is printed on the serial monitor, as shown in Figure 4.12.9, indicating that the Bluetooth module can send data normally. Of course, in order to test the accuracy, You can test more times and try to test in different environments.

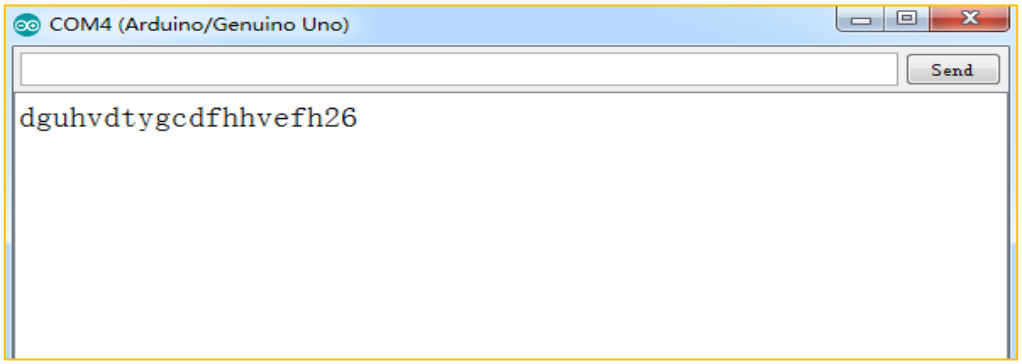


Figure 4.12.9

6、As shown in Figure 4.12.11, we can input the content you want to send on the serial monitor. After clicking “Send”, the data can be sent to the mobile APP via Bluetooth, as shown in Figure 4.12.12.

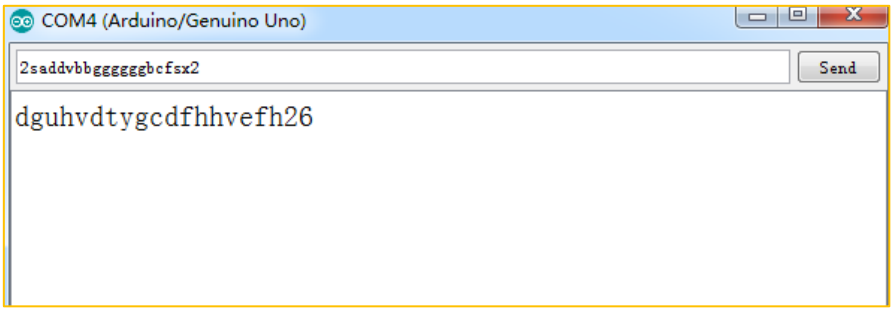


Figure 4.12.10

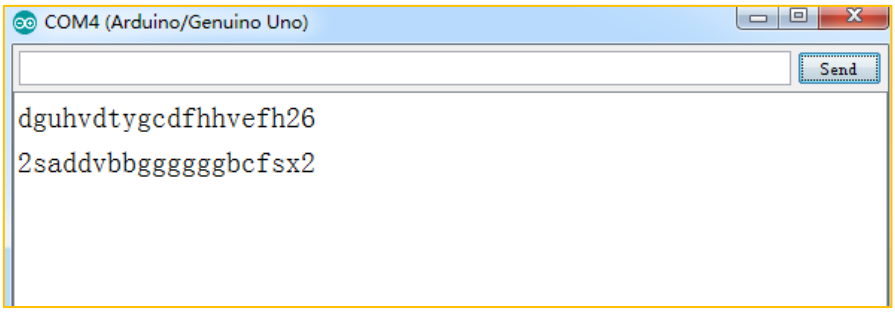


Figure 4.12.11

Figure 4.12.12

In the above test process, the PC and Android can send and receive data normally, indicating that the Bluetooth module communicates normally and achieve the desired effect. Then it can be used as a bridge between the "Hummer-Bot" and the APP, and control the "Hummer- Bot" to achieve a variety of desired features.

4.13 Mobile Bluetooth Remote Control Function Experiment

4.13.1 Bluetooth Data Protocol

Using the Bluetooth remote control car, the Android App is used to send commands to the Arduino UNO serial port via Bluetooth to control the forward, reverse, and speed of the motor. Since it involves wireless communication, one of the indispensable issues is the communication problem between the two devices. But there is no common "language" between them, so we need to design a communication protocol to ensure data interaction between Android APP and Arduino UNO.

The main process is: the Android APP sends the control command and packages the command into the corresponding data packet, and then sends it to the Bluetooth module (JDY-16). After receiving the data, the JDY-16 Bluetooth module transmits the data to the Arduino UNO main control board through the serial port. Then, the Arduino UNO main control board parses the data and then performs the corresponding action. The data format sent by the upper computer (Android APP) is as follows, mainly consisting of 8 fields:

Protocol Header	Data Length	Device Type	Device Address	Function Code	Control Data	Check Sum	Protocol End Code

- "protocol header" represents the mark of the beginning of a frame of valid data, fixed at 0xAA.
- "Data Length" - The effective data length except the data start and end code.
- “Device Type”--Indicates the model number of the smart car being controlled.
- “Device Address”--The address number of the controlled smart car (some scenes control multiple smart cars, the default is 0x01).
- “Function code”----Data control function type (see function code introduction for details)
- "Data" ------ function control values, such as speed, angle (length is not fixed)
- "Checksum"---- is the sum of all the values of the header and the end of the packet, only the last 16 bits are reserved, and the high-order is sent before
- "Procedure tail" -- is the end of packet transmission, fixed to 0x55.

In the above 8 fields, in the program we use a structure to represent

```
typedef struct
{
    unsigned char start_code;     // 8bit 0xAA
    unsigned char len;
    unsigned char type;
    unsigned char addr;
```

```
    unsigned short int function;   // 16 bit
    unsigned char *data;           // n bit
    unsigned short int sum;        // check sum
    unsigned char end_code;        // 8bit 0x55
}ST_protocol;
```

currently support is as follows: `typedef enum`

```
{
    E_BATTERY = 1,
    E_LED = 2,
    E_BUZZER = 3,
    E_INFO = 4,
    E_ROBOT_CONTROL_DIRECTION = 5,
    E_ROBOT_CONTROL_SPEED = 6,
    E_TEMPERATURE = 7,
    E_INFRARED_TRACKING = 8,
    E_ULTRASONIC = 9,
    E_INFRARED_REMOTE = 10,
    E_INFRARED_AVOIDANCE = 11,
    E_CONTROL_MODE = 12,
    E_BUTTON = 13,
    E_LED_MAXTRIX = 14,
    E_CMD_LINE = 15,
    E_VERSION = 16,
    E_UPGRADE = 17,
    E_PHOTORESISTOR = 18,
    E_CONTOROL_CODE_MAX,
} E CONTOROL FUNC;
```

For example: a complete data packet can be such a "AA 07 01 01 06 50 00 5F 55", where:

- "AA" ---------- packet header
- "07" ---------- The transmitted data length is 7 bytes.
- "06" ---------- is the transmission function code. Through the above enumeration, we know that 06 refers to the transmission "speed".
- "50 (or 0050)"-- is the control data, where 0x50 is hexadecimal, converted to binary to 80, function code transmission of 06, then the data here is the speed value, that is, the speed is 80.
- “005F”---------- is the checksum, ie 0x07+0x01+0x01+0x06+0x50=0x5F.

- "55" ---------- is the end of the protocol, indicating the end of the data transfer.

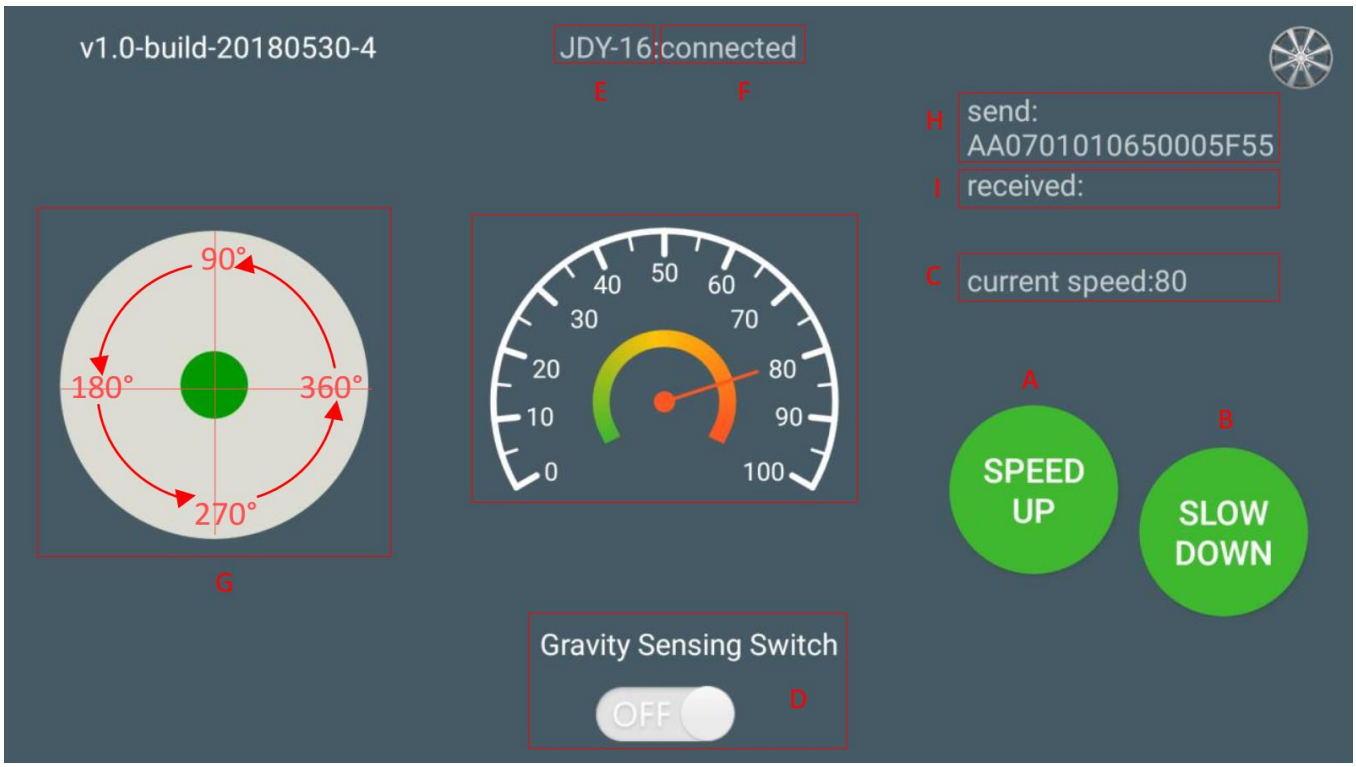


Figure 4.13.1 Android interface

In Figure 4.13.1 above:

- “A, B” part is the acceleration/deceleration button.
- “C” section includes the dashboard and digital display area, and the two sections are synchronized. Indicates the current speed, ie the speed after acceleration and deceleration,
- “D” part is a gravity remote sensing switch that can be switched to the gravity remote sensing mode.
- "E" portion indicates the name of the Bluetooth that is currently connected.
- “F” part indicates the Bluetooth connection status. If Bluetooth is not connected, it is displayed here. Show "disconnected".
- “G” part is a manual rocker, and the joystick can be rotated to allow the car to rotate.
- “I” part is the data return area, such as the current driving state and speed of the car.
- "H" part is the transmitted data packet, for example, the data is "AA 01 01 06 23 00 2B 55"

At this time the speed is 35 (23 is hexadecimal data, converted to decimal is the current speed of 35).

If the data sent is “AA 01 01 05 00 5B 00 62 55”, the car is moving forward (05 is the direction control command, 005B is converted to binary number 91, we can know by Figure4.13.1, also when 91° Belong to the forward).

4.13.2 Mobile phone Bluetooth remote control function software logic

To control the car through the mobile phone Bluetooth, it is necessary to establish a connection between the mobile phone APP and the Bluetooth module, and then the mobile APP sends a control command to the Bluetooth module. After receiving the APP control command, the Bluetooth module sends the control command to the Arduino UNO main control board through the serial port, Arduino. After the UNO main control board parses the command, it controls the car to perform the corresponding action. The distance process is shown in Figure 4.13.2.

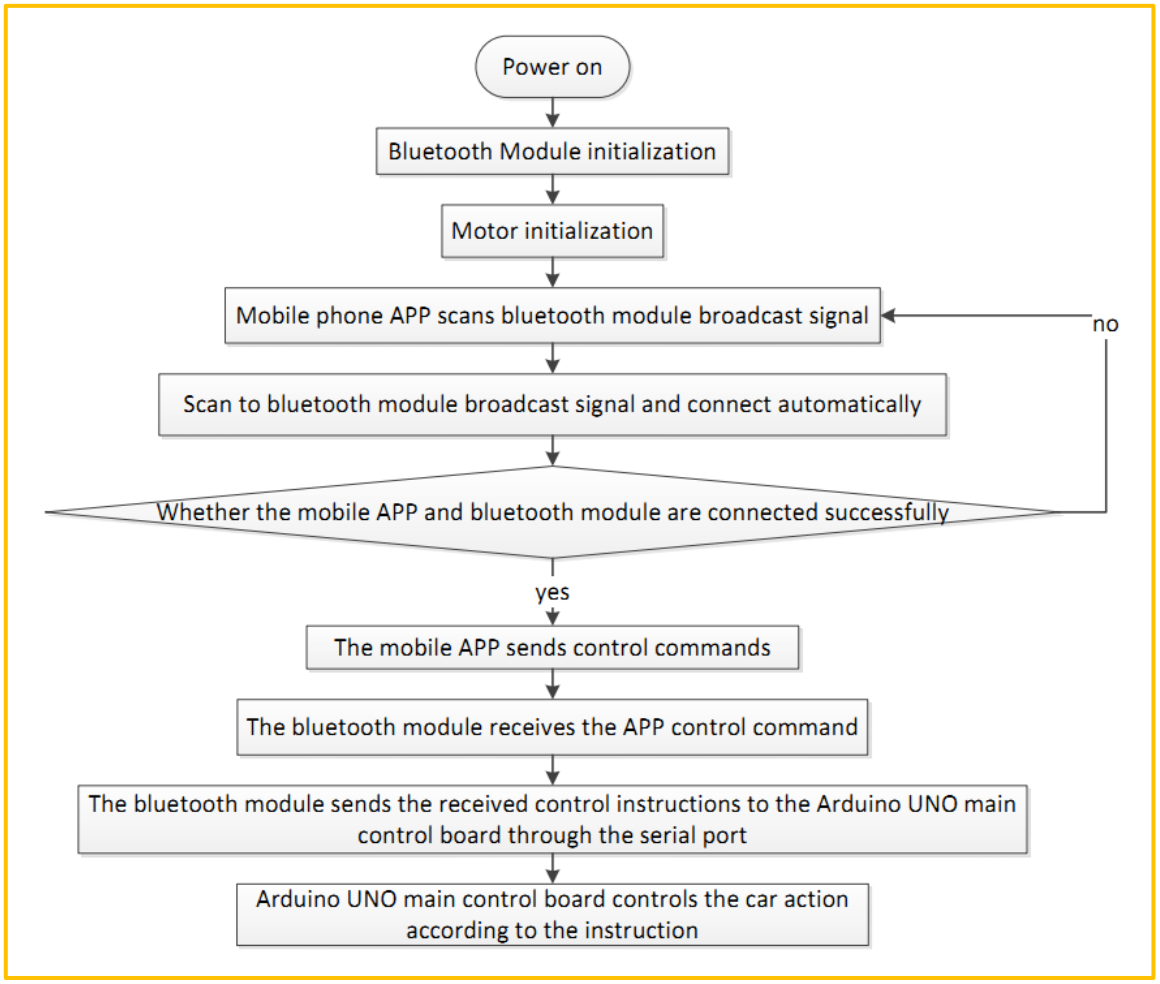


Figure 4.13.2 Bluetooth remote control function software logic flow chart

4.13.3 Mobile phone Bluetooth remote control function experiment

1) Prepare an Android phone

2) In the CD-ROM data, copy the.apk file in the APP installation package folder to the phone;

3) Install the APP to the Android phone;

4) Openin the disc data“Lesson\Comprehensive Experiment\Hummerbot_Bluetooth\Hummerbot_Bluetooth.ino”

5) Burn the Hummerbot_Bluetooth.ino program to the Arduino UNO R3 main control board;

6) Turn on the power switch;

7) Turn on the Bluetooth in the phone settings and start the APP. After the connection is successful, you can see that the Bluetooth module indicator is blinking and long. If you have downloaded the program to the development board before, it will be directly You can use your mobile phone to control the car, as shown in Figure 4.13.3.

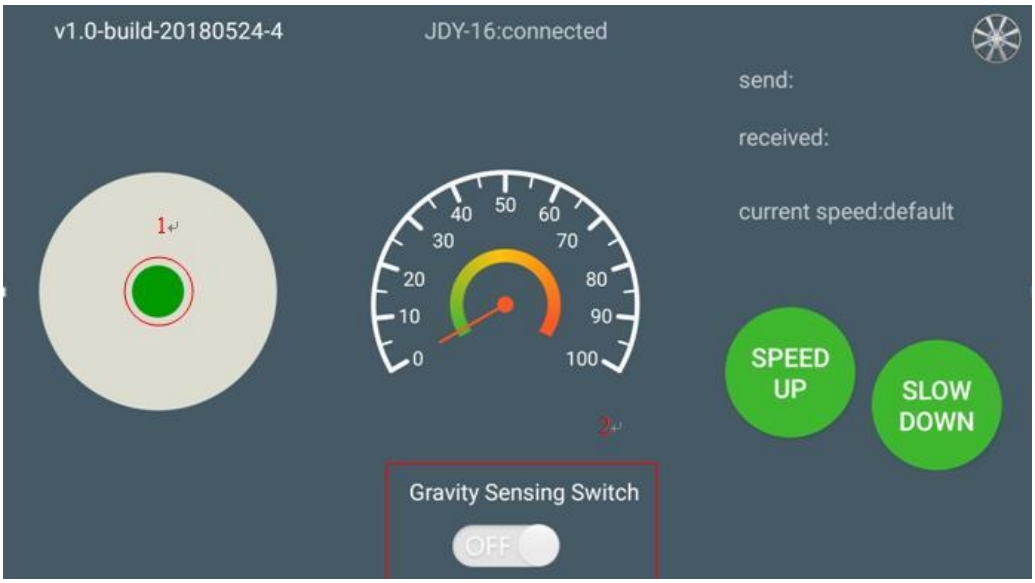


Figure 4.13.3 APP bluetooth control interface

In Figure 4.13.3, we can see the identifier "1" and the identifier "2". When the Bluetooth connection is successful, slide the green dot at the "1" in any direction, and the trolley will move in the corresponding direction. Open the gravity sensor switch at the "2" mark, and the APP switches to the gravity sensing mode. The mobile phone can control the direction of the car movement.

```
#include "Hummerbot.h"
#include "BluetoothHandle.h"
#include "ProtocolParser.h"
#include "KeyMap.h"
#include "debug.h"
#define IN1_PIN 6   // PWMB
#define IN2_PIN 10  // DIRB  ---  right
#define IN3_PIN 5   // DIRA  ---  left
#define IN4_PIN 9   // PWMA
```

```
      case E_ROBOT_CONTROL_DIRECTION:
        hbot.Drive(mProtocol->GetRobotDegree());
        break;
      case E_ROBOT_CONTROL_SPEED:
        hbot.SetSpeed(mProtocol->GetRobotSpeed());
        break ;
      case E_CONTROL_MODE:
        //Serial.println(mProtocol->GetControlMode());
        hbot.SetControlMode(mProtocol->GetControlMode());
        break;
      case E_LED:
        hbot.SetRgbColor(E_RGB_ALL, mProtocol->GetRgbValue());
        break;
      case E_VERSION:
        hbot.SendVersionPackage();
        break;
    }
  }
}
void loop()
{
  static byte mode;
  static bool recv_flag;
  mProtocol->RecevData();
  if (recv_flag = mProtocol->ParserPackage()) {
    if (mProtocol->GetRobotControlFun() == E_CONTROL_MODE) {
      hbot.SetControlMode(mProtocol->GetControlMode());
      return;
    }
  }
  switch (hbot.GetControlMode()) {
    case E_BLUETOOTH_CONTROL:
      HandleBluetoothRemote(recv_flag);
      DEBUG_LOG(DEBUG_LEVEL_INFO, "E_BLUETOOTH_CONTROL \n");
      break;
    default:
      break;
  }
```

```
switch (hbot.GetStatus()) {
    case E_FORWARD:
      hbot.SetRgbColor(E_RGB_ALL, RGB_WHITE);
      break;
    case E_LEFT:
      hbot.SetRgbColor(E_RGB_LEFT, RGB_WHITE);
      break;
    case E_RIGHT:
      hbot.SetRgbColor(E_RGB_RIGHT, RGB_WHITE);
      //   Mirage.Sing(S_OhOoh);
      break;
    case E_BACK:
      hbot.SetRgbColor(E_RGB_ALL, RGB_RED);
      break;
    case E_STOP:
      hbot.SetRgbColor(E_RGB_ALL, RGB_OFF);
      break;
    case E_SPEED_UP:
      hbot.SetRgbColor(E_RGB_ALL, hbot.GetSpeed() * 2.5);
      break;
    case E_SPEED_DOWN:
      hbot.SetRgbColor(E_RGB_ALL, hbot.GetSpeed() * 2.5);
      break;
    default:
      break;
  }
}
```

4.14 PS2 handle test experiment (optional)

4.14.1 PS2 Handle Kit Introduction

The PS2 handle is the remote control handle of the Sony game console. Sony's series of game consoles are very popular all over the world. I don't know when someone got the idea of the PS2 controller and cracked the communication protocol, so that the handle can be connected to other devices for remote control, such as remote control of the familiar four-wheelers and robots. The outstanding feature is that this handle is now very cost-effective. The buttons are rich and easy to extend to other applications. Figure 4.14.1 is a commonly used PS2 wireless controller.

Figure 4.14.1 PS2 wireless controller physical map

The PS2 handle consists of two parts: the handle and the receiver. The handle needs two sections of No.7 1.5V power supply. The power supply of the receiver and the Arduino UNO use the same power supply. The power supply range is 3~5V, which cannot be reversed, and cannot exceed voltage and overvoltage. And reverse connection, will cause the receiver to burn out. There is a power switch on the handle, ON ON/OFF OFF, the handle switch is turned ON. When the receiver is not searched, the light on the handle will flash continuously. In a certain period of time, the receiver has not been searched yet. The handle will enter standby mode and the light on the handle will be extinguished. At this time, press the “START” button to wake up the handle.

The receiver is connected to the Arduino and powered by the Arduino. In the unpaired condition, the green light flashes. The handle is opened, the receiver is powered, and the handle and receiver are

automatically paired. At this time, the light is always on and the handle is successfully paired. Press the button "MODE" (the handle batch is different, the above logo may be "ANALOG", but it will not affect the use), you can select "red light mode", "green light mode".

Some users report that the handle and receiver are not properly paired! Most problems are that the receiver is not wired correctly or there is a problem with the program.

Solution: The receiver is only connected to the power supply (the power cable must be connected correctly), and no data cable or clock cable is connected. In general, the handle can be successfully paired. After the pairing is successful, the light is always on, indicating that the handle is good. Then check if the wiring is correct and there is a problem with the program migration.

PS2	Arduino UNO
VCC	VCC
GND	GND
CLK	11
CMD	7
CS	8
DAT	4

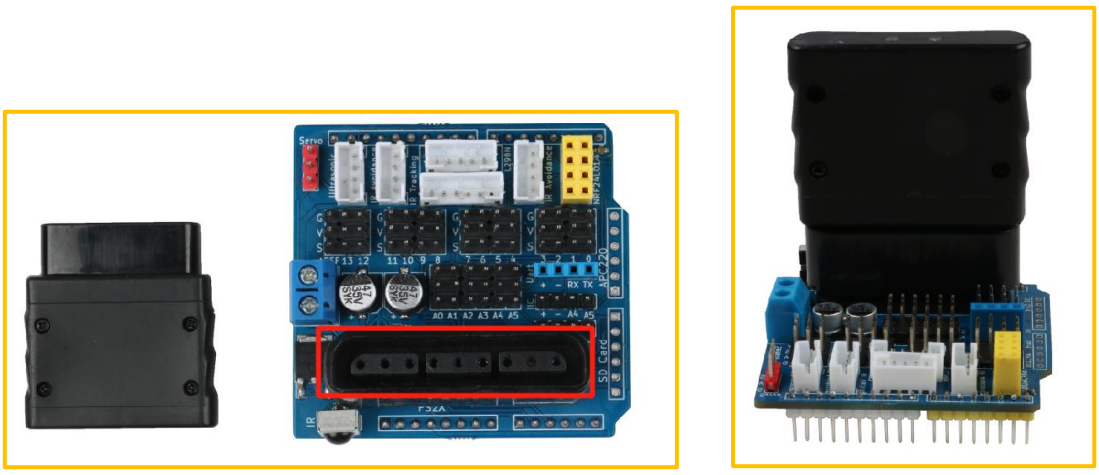


Figure 4.14.2 PS2 Remote Receiver Module

At the end of the receiver, there are a total of nine interfaces, each of which functions as shown in the following table:

1	2	3	4	5	6	7	8	9
DI/DAT	DO/CMD	NC	GND	VDD	CS/SEL	CLK	NC	ACK

Note: The appearance of the receiver will be different when the batch is different. One has a red light on the power supply and one has no power light on it, but the method is the same, and the pin definition is the same.

- DI/DAT: Signal flow, from the handle to the host, this signal is an 8-bit serial data that is synchronously transmitted on the falling edge of the clock. The reading of the signal is done during the course of the clock from high to low.
- DO/CMD: Signal flow direction, from the host to the handle. This signal is opposite to DI. The signal is an 8-bit serial data that is synchronously transmitted on the falling edge of the clock.
- NC: empty port;
- GND: power ground;
- VDD: Receiver working power supply, power supply range 3~5V;
- CS/SEL: Used to provide the handle trigger signal. During communication, at a low level;
- CLK: clock signal, issued by the host to keep data synchronized;
- NC: empty port;
- ACK: The response signal from the handle to the host. This signal goes low during the last cycle of each 8-bit data transmission and CS remains low. If the CS signal does not go low, the PS host will try another peripheral for about 60 microseconds. The ACK port is not used during programming.

4.14.2 PS2 Handle Test Experimental Procedure

1) For the wiring is simple, our expansion board has already boarded the PS2 adapter board

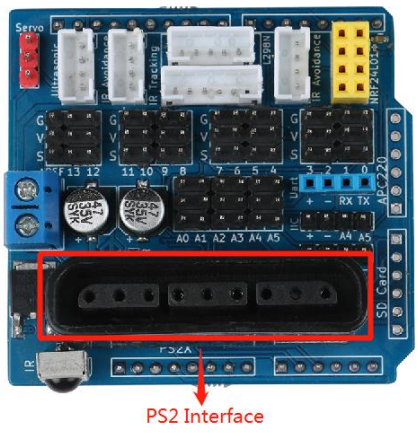


Figure 4.14.3 PS2 handle connection diagram

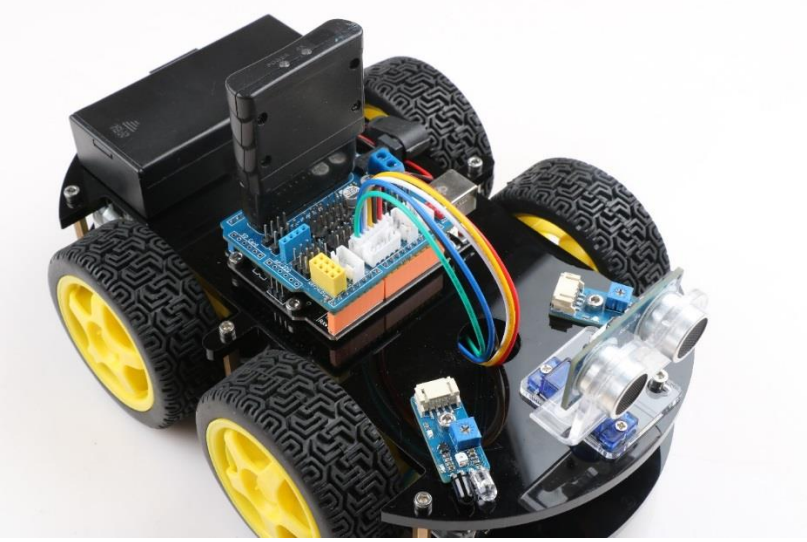


Figure 4.14.4 PS2 handle connection physical map

2) Openin the disc data“Lesson\Module_Test\PS2X_Test\ PS2X_Test.ino”

3) Download the PS2X_Test.ino program to the Arduino development board;

4) Turn on the power switch;

5) Open the Arduino serial monitor;

6) Turn on the PS2 remote control. If the receiving head is connected to the remote control (or the pairing is successful), the indicator light on the receiving head is long, otherwise, the LED light flashes continuously.

7) Press any button on the remote control to see the corresponding data on the “Serial Monitor”, as shown in Figure 4.14.5.

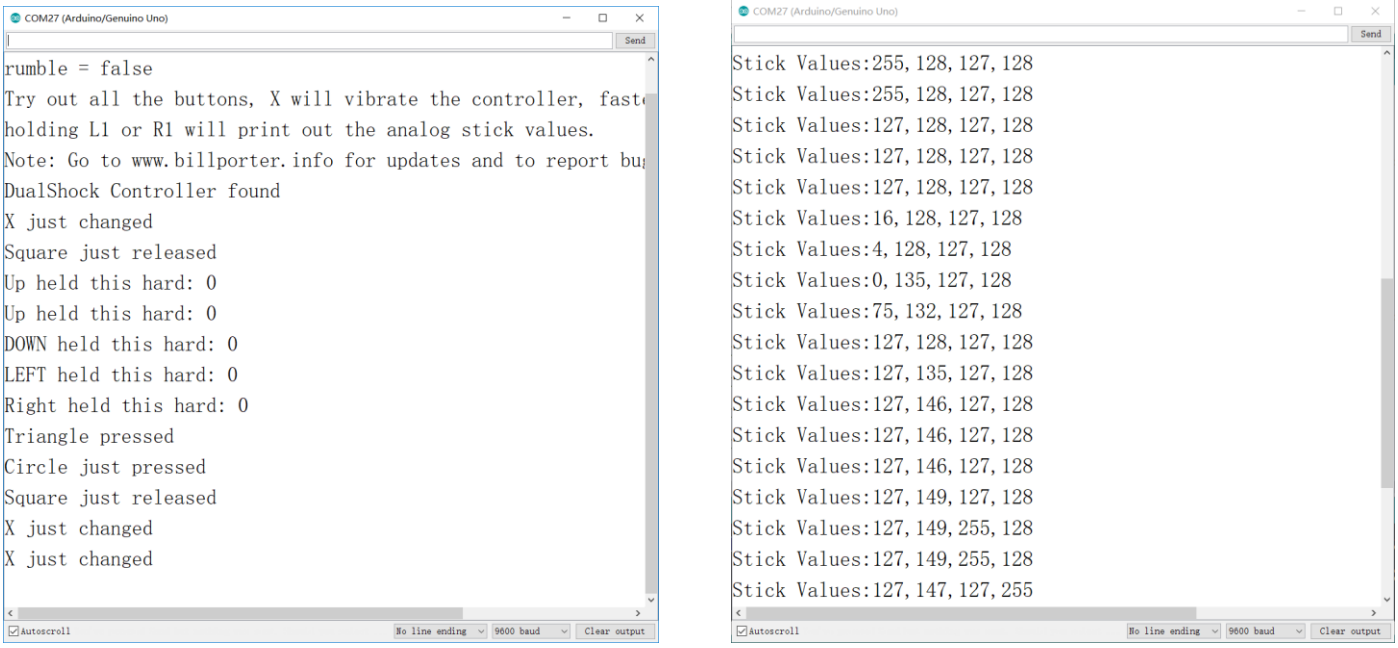


Figure 4.14.5 “Serial Port Monitor” Data Display

4.15 Hummer-Bot PS2 Handle Remote Control Experiment

4.15.1 Hummer-BotPS2 controller remote control function software logic

PS2 controller remote control, as the name suggests is to achieve the purpose of remote control Hummer-Bot car through the PS2 handle, then through a small PS2 handle, how to make Hummer-Bot "obedient"? This is the problem we need to explore.

To make Hummer-Bot "obeying", you first need Hummer-Bot to hear the instructions of the PS2 controller at all times, that is, the two must be connected within a certain communication range, and secondly, Hummer-Bot can " Understand the meaning of the PS2 controller, that is, to understand the meaning of each instruction of the PS2 controller. Finally, Hummer-Bot controls the limbs (four wheels) according to the instructions of the PS2 controller to do the corresponding action. This is the basic principle of the remote control function of the PS2 controller. The software logic is shown in Figure 4.15.1.

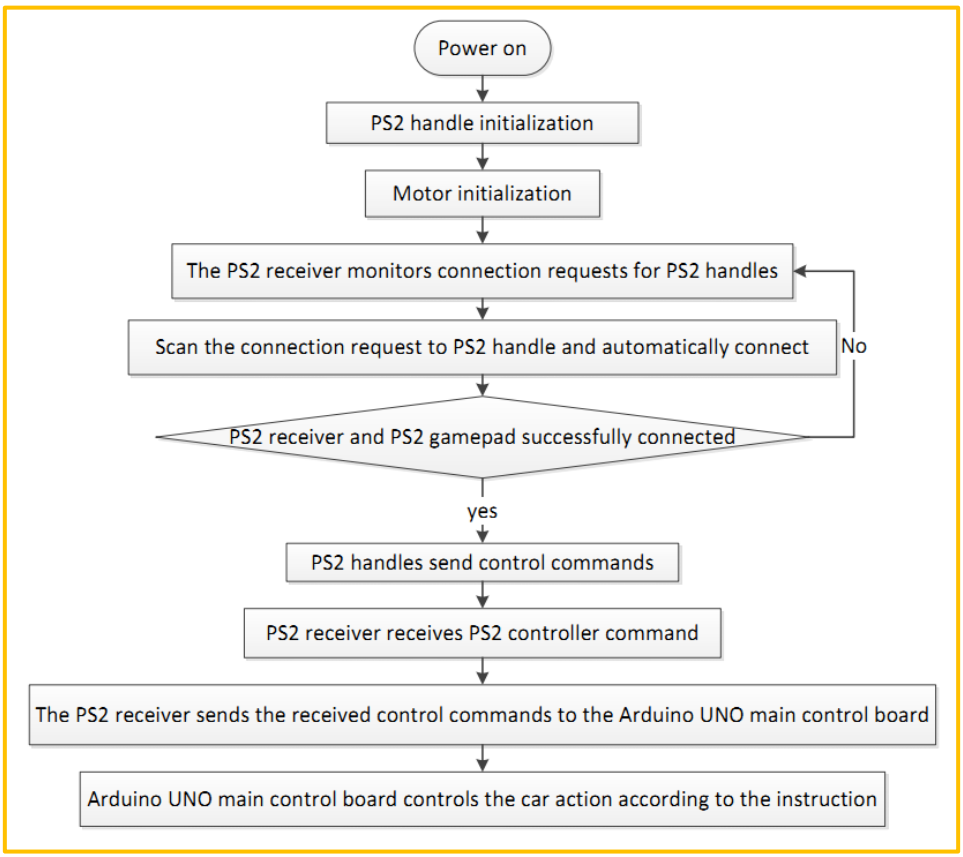


Figure 4.15.1 PS2 remote control software logic flow chart

4.15.2 Hummer-Bot PS2 Handle Remote Control Experimental Procedure

1) Openin the disc data“Lesson\Comprehensive Experiment\Hummerbot_PS2\Hummerbot_PS2.ino”

2) Burn the IR_Control__Function .ino program to the Arduino UNO R3 main control board;

3) Turn on the power and press the function button defined on the remote control of the PS2 handle (as shown in Figure 4.15.2) to observe the progress of the car.

Figure 4.15.2 PS2 handle function button diagram

- Mark UP: move forward
- Mark DOWN: move backward
- Mark LEFT: turn left
- Mark RIGHT: right
- Mark A: speed up
- Mark B: Left spin
- Mark C: slow down
- Mark D: Right spin
- Mark 3: Right joystick (mark 5) control key, only when you press R1, the Right joystick will work.
- Mark 4: Left joystick (mark 6) control key, only when you press L1, the Left joystick will work.
- Joystick left: joystick remotes control direction
- Joystick Right：joystick remotes control speed

PS2 controller remote control experiment program code:

```
#include "Hummerbot.h"
```

```
#include "BluetoothHandle.h"
#include "ProtocolParser.h"
#include "KeyMap.h"
#include "debug.h"

#define IN1_PIN 6   // PWMB
#define IN2_PIN 10  // DIRB  ---  right
#define IN3_PIN 5   // DIRA  ---  left
#define IN4_PIN 9   // PWMA

#define SERVO_PIN 13
#define UL_SING_PIN 3
#define UL_RGB_PIN 2
#define PS2X_CLK 11
#define PS2X_CMD 7
#define PS2X_CS  8
#define PS2X_DAT 4

ProtocolParser *mProtocol = new ProtocolParser();
Hummerbot hbot(mProtocol, IN1_PIN, IN2_PIN, IN3_PIN, IN4_PIN);
byte Ps2xStatus, Ps2xType;

void setup()
{
  Serial.begin(9600);
  hbot.init();
  hbot.SetControlMode(E_PS2_REMOTE_CONTROL);
  hbot.SetRgbUltrasonicPin(UL_SING_PIN, UL_RGB_PIN, SERVO_PIN);
  hbot.SetPs2xPin(PS2X_CLK, PS2X_CMD, PS2X_CS, PS2X_DAT);
  hbot.SetSpeed(0);
  Ps2xType = hbot.mPs2x->readType();
}

void HandlePS2()
{
  static int vibrate = 0;
  byte PSS_X = 0, PSS_Y = 0;
  hbot.mPs2x->read_gamepad(false, vibrate); // read controller and set large motor to
spin at 'vibrate' speed
```

```
  if (hbot.mPs2x->ButtonDataByte()) {
    if (hbot.mPs2x->Button(PSB_PAD_UP)) {     //will be TRUE as long as button is
pressed
      hbot.GoForward();
    }
    if (hbot.mPs2x->Button(PSB_PAD_RIGHT)) {
      hbot.Drive(20);
    }
    if (hbot.mPs2x->Button(PSB_PAD_LEFT)) {
      hbot.Drive(160);
    }
    if (hbot.mPs2x->Button(PSB_PAD_DOWN)) {
      hbot.GoBack();
    }
    vibrate = hbot.mPs2x->Analog(PSAB_CROSS);  //this will set the large motor vibrate
speed based on how hard you press the blue (X) button
    if (hbot.mPs2x->Button(PSB_CROSS)) {            //will be TRUE if button was JUST
pressed OR released
      hbot.SpeedDown(5);
    }
    if (hbot.mPs2x->Button(PSB_TRIANGLE)) {
      hbot.SpeedUp(5);
    }
    if (hbot.mPs2x->Button(PSB_SQUARE)) {
      hbot.TurnLeft();
    }
    if (hbot.mPs2x->Button(PSB_CIRCLE)) {
      hbot.TurnRight();
    }
    if (hbot.mPs2x->Button(PSB_L1) || hbot.mPs2x->Button(PSB_R1)) {
      uint16_t RightValue = hbot.mPs2x->RightHart();
      uint16_t LeftValue = hbot.mPs2x->LeftHart();
      if (RightValue != CENTER) {
        if ((RightValue > 0) && (RightValue < 180 )) {
          hbot.SpeedUp(2);
        } else {
          hbot.SpeedDown(2);
        }
      }
```

```
      if (LeftValue != CENTER) {
        hbot.Drive(LeftValue);
      } else {
        hbot.KeepStop();
      }
    }
  }
  else {
    hbot.KeepStop();
  }
  delay(50);
}

void loop()
{
  switch (hbot.GetControlMode()) {
    case E_PS2_REMOTE_CONTROL:
      while (Ps2xStatus != 0) { //skip loop if no controller found
        delay(500);
        Ps2xStatus = hbot.ResetPs2xPin();
        Ps2xType = hbot.mPs2x->readType();
        DEBUG_LOG(DEBUG_LEVEL_INFO, "E_PS2_REMOTE_CONTROL \n");
      }
      if (Ps2xType != 2) {
        HandlePS2();
      }
      break;
    default:
      break;
  }
  switch (hbot.GetStatus()) {
    case E_FORWARD:
      hbot.SetRgbColor(E_RGB_ALL, RGB_WHITE);
      break;
    case E_LEFT:
      hbot.SetRgbColor(E_RGB_LEFT, RGB_WHITE);
      break;
    case E_RIGHT:
      hbot.SetRgbColor(E_RGB_RIGHT, RGB_WHITE);
```

```
      //   Mirage.Sing(S_OhOoh);
      break;
    case E_BACK:
      hbot.SetRgbColor(E_RGB_ALL, RGB_RED);
      break;
    case E_STOP:
      hbot.SetRgbColor(E_RGB_ALL, RGB_OFF);
      break;
    case E_SPEED_UP:
      hbot.SetRgbColor(E_RGB_ALL, hbot.GetSpeed() * 2.5);
      break;
    case E_SPEED_DOWN:
      hbot.SetRgbColor(E_RGB_ALL, hbot.GetSpeed() * 2.5);
      break;
    default:
      break;
  }
}
```

4.16 Hummer-Bot full function comprehensive experiment

In the previous chapter, we introduced how each module of the Hummer-Bot is tested and uses each module to control the car. In this section we will introduce how to combine all the functions of Hummer-Bot. To achieve all the functions of the car we have to be based on Bluetooth control. Use the Bluetooth APP to switch between various control modes.

Specific steps are as follows:

1) Prepare an Android phone
2) In the CD-ROM data, copy the apk file in the APP installation package folder to the phone;
3) Install the APP to the Android phone;
4) Open "Courseware Code\All Functions\Hummer-bot3.0_AllFunction\Hummer-bot3.0_AllFunction.ino"
5) Burn the Hummer-bot3.0_AllFunction.ino program to the Arduino UNO R3 main control board;
6) Turn on the power switch;
7) Turn on the Bluetooth in the phone settings and start the APP. After selecting the Hummer-Bot connection, you can see that the Bluetooth module indicator is blinking and changing.
8) Enter the APP selection interface, as shown in Figure 4.16.1; you can choose the control method you want;

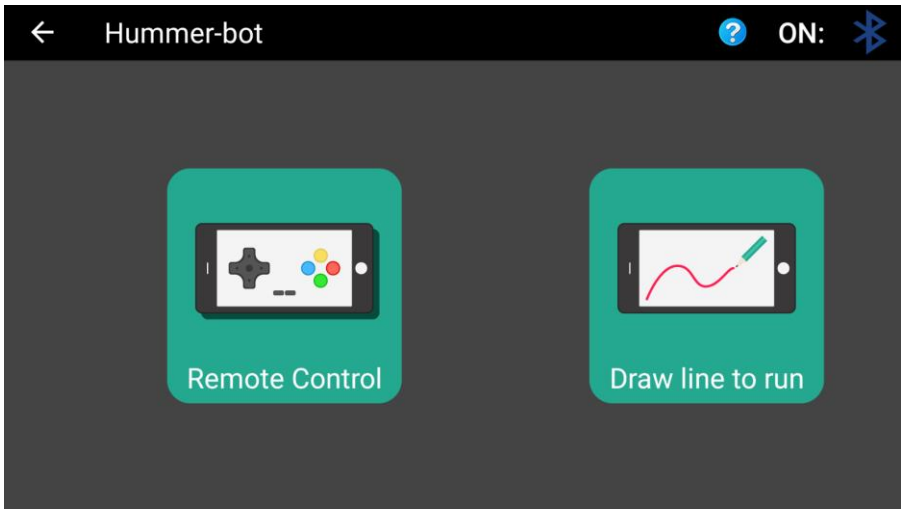


Figure 4.16.1 APP control mode selection interface 1

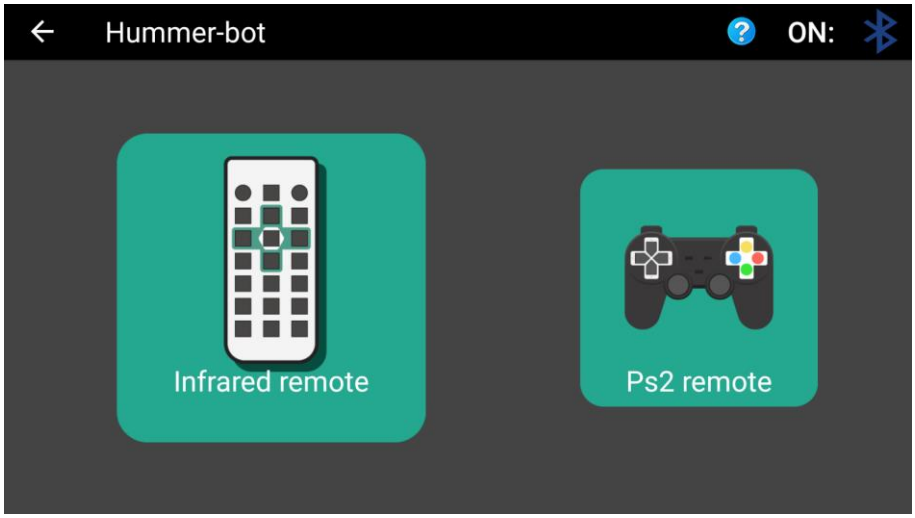


Figure 4.16.2 APP control mode selection interface 2

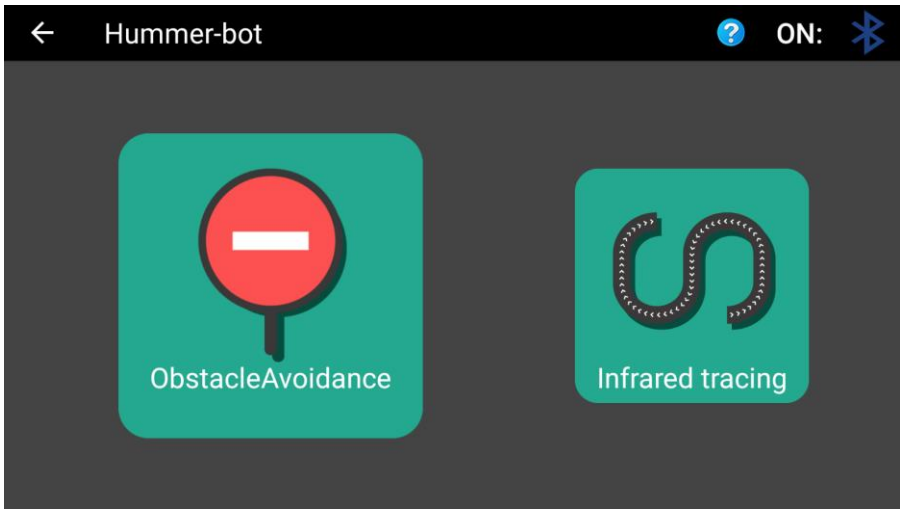


Figure 4.16.3 APP control mode selection interface 3

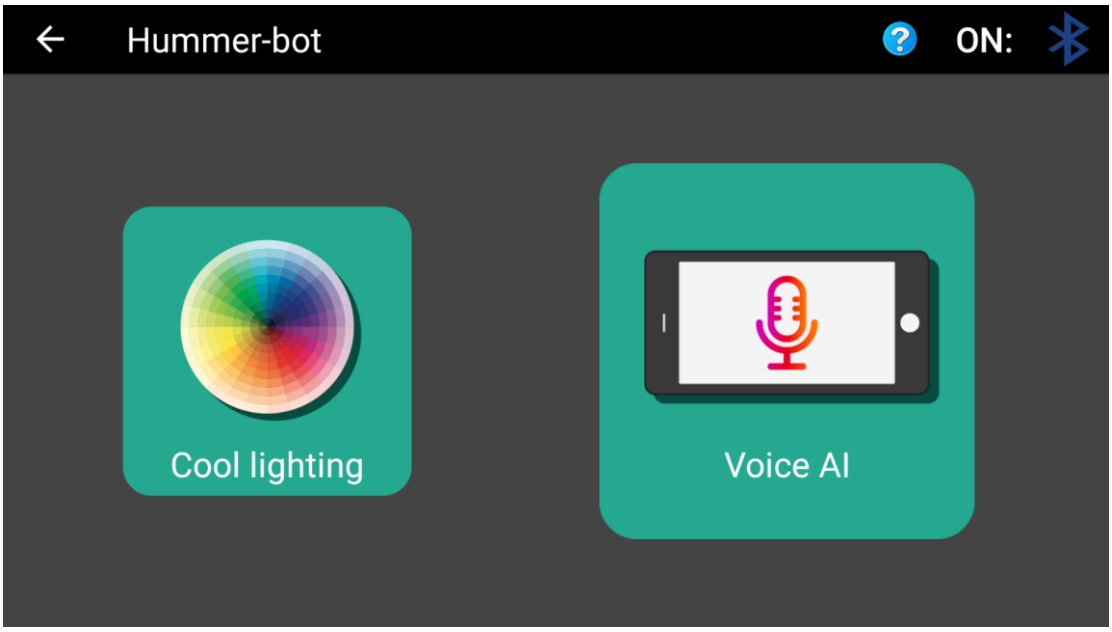


Figure 4.16.4 APP control mode selection interface 4

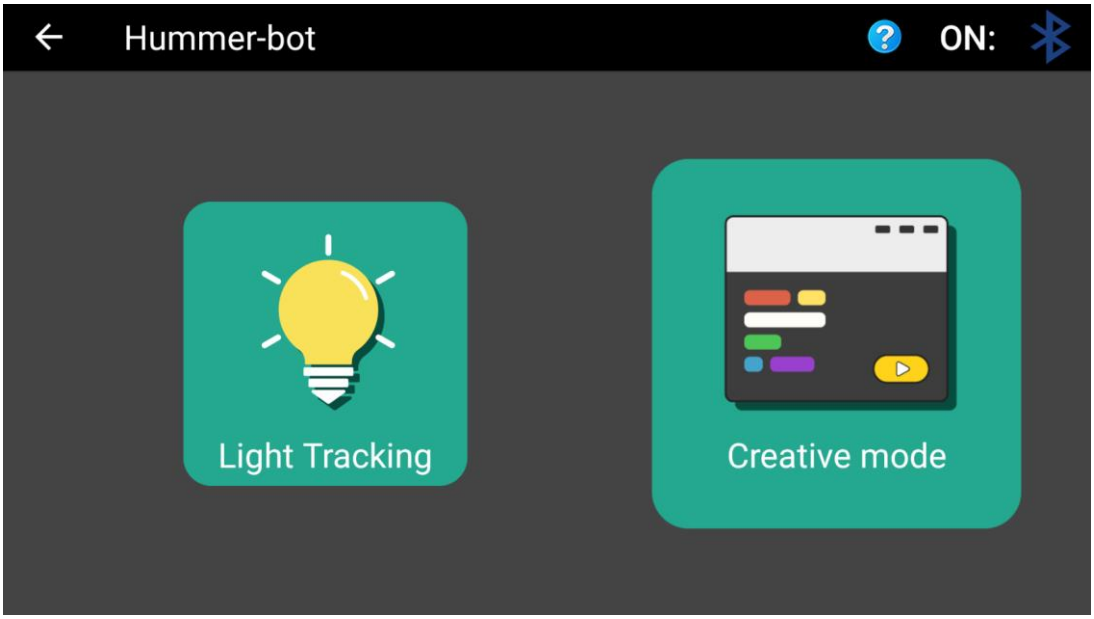


Figure 4.16.5 APP control mode selection interface 5

Remark：

In voice mode, you can say forward, left, right, back and stop to the phone microphone.The car will move by voice control