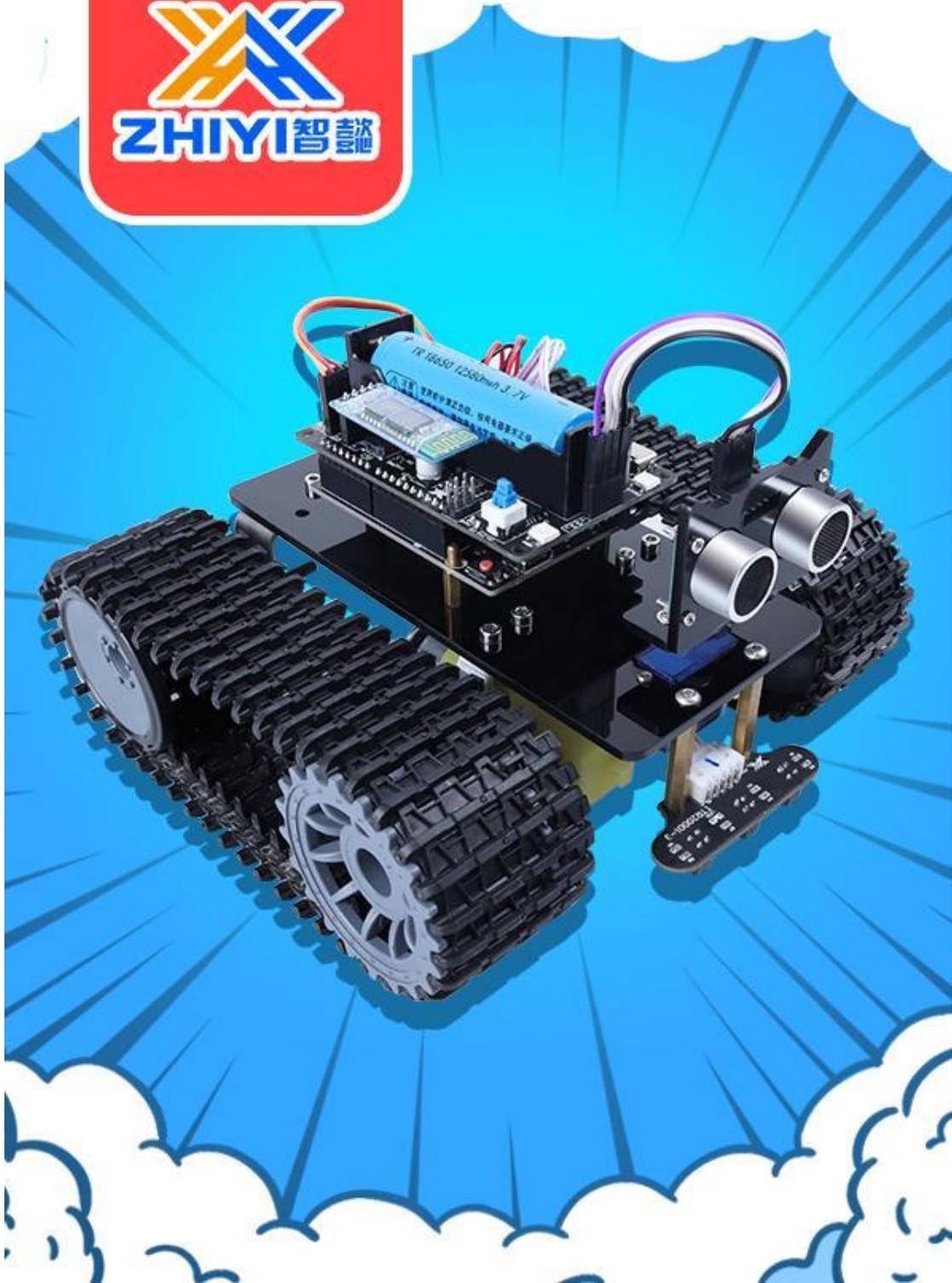




U-BOT crawler trolley V2.0
ZYC0053



Directory

1	Installing IDE.....	1
1.1	Introduction.....	1
1.2	Click Windows Installer.....	2
1.3	Click JUST DOWNLOAD.....	3
1.4	Click Install to initiate installation.....	4
1.5	Installing Arduino(Mac OS X).....	6
1.6	Installing Arduino(Linux).....	6
2	Add Libraries.....	7
2.1	Installing Additional Arduino Libraries.....	7
2.2	What are Libraries.....	7
2.3	How to Install a Library.....	7
2.4	Example:SR04.....	8
2.5	Add method two:.....	8
3	Ultrasonic transducer.....	9
3.1	Introduction.....	9
3.2	Obstacle avoidance principle.....	10
3.3	Acquisition of measurement data.....	11
4	Three-way tracking module.....	14
4.1	Introduction.....	14
4.2	Principle Introduction:.....	14

4.2.1 Photodiode.....	14
4.2 Phototriode.....	14
4.3 get data.....	15
5 buzzer.....	18
5.1 Introduction.....	18
6 Bluetooth.....	19
6.1 Introduction.....	19
6.2 Bluetooth module wiring.....	20
6.3 Write a program.....	20
6.4 Installing mobile software.....	20
6.5 The command sets the reference table.....	21
7 Infrared remote control.....	24
7.1 Principle of infrared remote control.....	24
7.2 routine is as follows:.....	25
8 servo.....	27
8.1 principle is introduced.....	27
8.2 Servo.h library.....	27
8.3 Why can't the analogWrite() statement be used?.....	28
8.4 Control the SERVO through the SERVO library.....	28
9 WS2812B.....	31
9.1 Introduction.....	31

9.2 characteristics.....	31
9.3 test program.....	32
10 code analysis.....	34
10.1 Pin configuration.....	34
10.2 Define car parameters.....	34
10.3 setup function.....	35
10.4 Motor drive.....	36
10.5 Infrared remote control.....	37
10.6 Ultrasonic sensor ranging.....	38
10.7 serial port communication.....	39
10.8 Set motor speed.....	40
10.9 Servo motor drive.....	41
10.10 print Data.....	41
10.11 Three road tracking.....	42

1 Installing IDE

1.1 Introduction

The Arduino Integrated Development Environment(IDE)is the software side of the Arduino platform.

In this lesson,you will learn how to setup your computer to use Arduino and how to set about the lessons that follow.

The Arduino software that you will use to program your Arduino is available for Windows,Mac and Linux.The installation process is different for all three platforms and unfortunately there is a certain amount of manual work to install the software.

STEP 1:Go to <https://www.arduino.cc/en/software>.

A screenshot of the Arduino IDE 1.8.13 download page. The page is split into two main sections. The left section has a light blue background and contains the Arduino logo, the title 'Arduino IDE 1.8.13', a paragraph describing the IDE as open-source software for writing code and uploading to boards, a reference to the 'Getting Started' page for installation instructions, and a 'SOURCE CODE' section mentioning GitHub and PGP-signed archives. The right section has a teal background and is titled 'DOWNLOAD OPTIONS'. It lists download links for Windows (Win 7 and newer, ZIP file), Windows app (Win 8.1 or 10, with a 'Get' button), Linux (32 bits, 64 bits, ARM 32 bits, ARM 64 bits), and Mac OS X (10.10 or newer). It also includes links for 'Release Notes' and 'Checksums (sha512)'.

 **Arduino IDE 1.8.13**

The open-source Arduino Software (IDE) makes it easy to write code and upload it to the board. This software can be used with any Arduino board.

Refer to the [Getting Started](#) page for Installation instructions.

SOURCE CODE

Active development of the Arduino software is [hosted by GitHub](#). See the instructions for [building the code](#). Latest release source code archives are available [here](#). The archives are PGP-signed so they can be verified using [this](#) gpg key.

DOWNLOAD OPTIONS

Windows Win 7 and newer
Windows ZIP file

Windows app Win 8.1 or 10 

Linux 32 bits
Linux 64 bits
Linux ARM 32 bits
Linux ARM 64 bits

Mac OS X 10.10 or newer

[Release Notes](#)
[Checksums \(sha512\)](#)

The version available at this website is usually the latest version,and the actual version may be newer than the version in the picture.

STEP2 : Download the development software that is compatible with the operating system of your computer.Take Windows as an example here.

DOWNLOAD OPTIONS

Windows Win 7 and newer
Windows ZIP file

Windows app Win 8.1 or 10 [Get](#) 

Linux 32 bits
Linux 64 bits
Linux ARM 32 bits
Linux ARM 64 bits

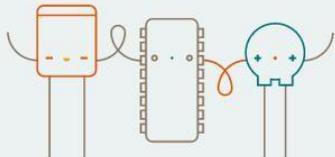
Mac OS X 10.10 or newer

[Release Notes](#)
[Checksums \(sha512\)](#)

1.2 Click Windows Installer.

Contribute to the Arduino Software

Consider supporting the Arduino Software by contributing to its development. (US tax payers, please note this contribution is not tax deductible). [Learn more on how your contribution will be used.](#)



SINCE MARCH 2015, THE ARDUINO IDE HAS BEEN DOWNLOADED **32,439,835** TIMES. (IMPRESSIVE!) NO LONGER JUST FOR ARDUINO AND GENUINO BOARDS, HUNDREDS OF COMPANIES AROUND THE WORLD ARE USING THE IDE TO PROGRAM THEIR DEVICES, INCLUDING COMPATIBLES, CLONES, AND EVEN COUNTERFEITS. HELP ACCELERATE ITS DEVELOPMENT WITH A SMALL CONTRIBUTION! REMEMBER: OPEN SOURCE IS LOVE!

\$3 **\$5** **\$10** **\$25** **\$50** **OTHER**

JUST DOWNLOAD

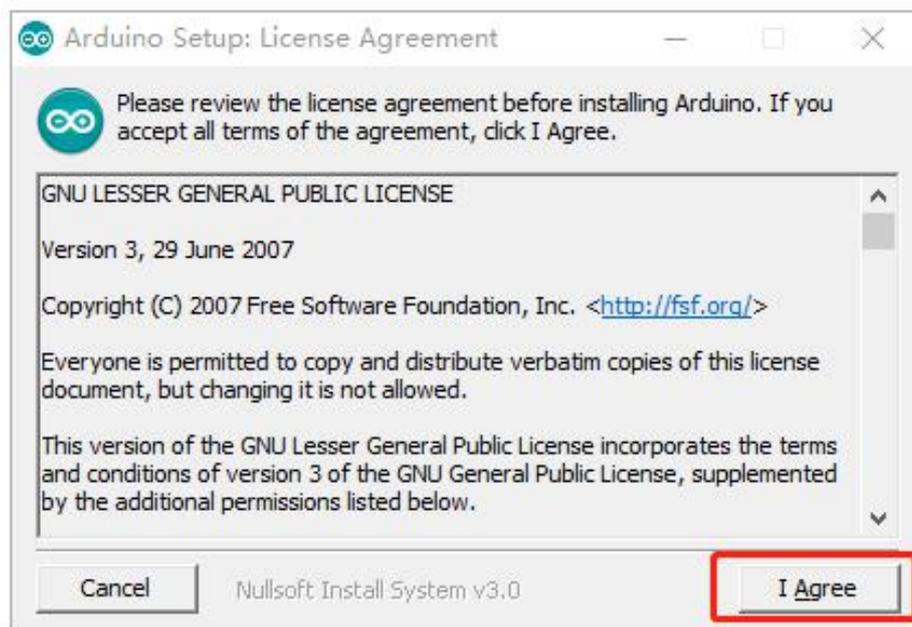
CONTRIBUTE & DOWNLOAD

1.3 Click JUST DOWNLOAD

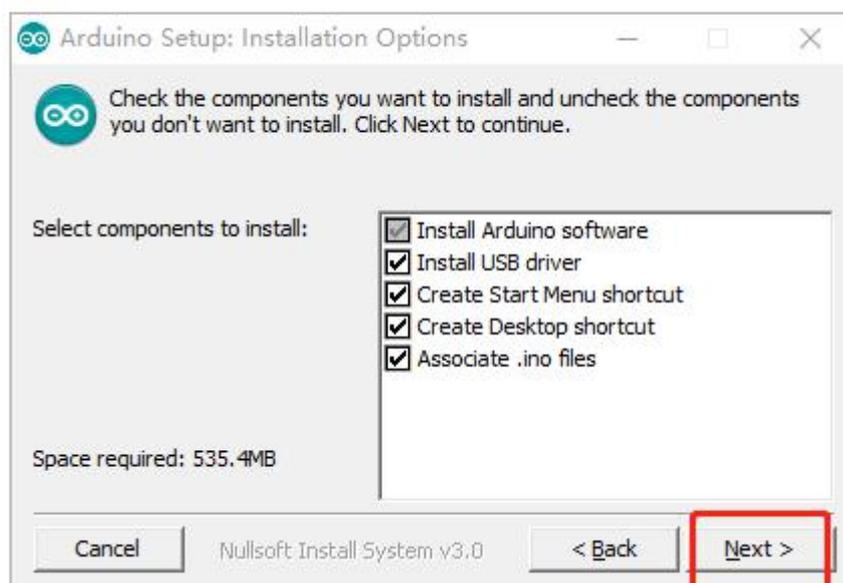
Also version 1.8.9 is available in the material we provided, and the versions of our materials are the latest versions when this course was made.

Installing Arduino(Windows)

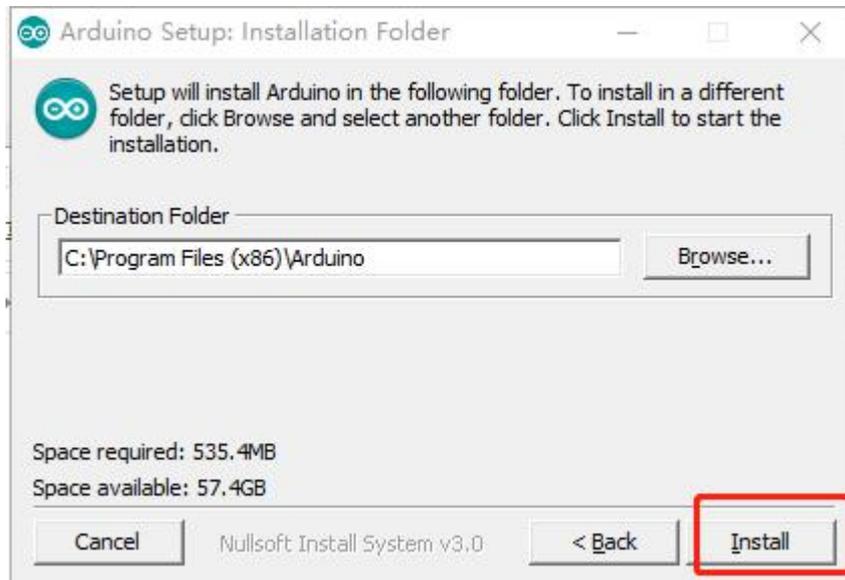
 arduino-1.8.13-windows



Click I Agree to see the following interface

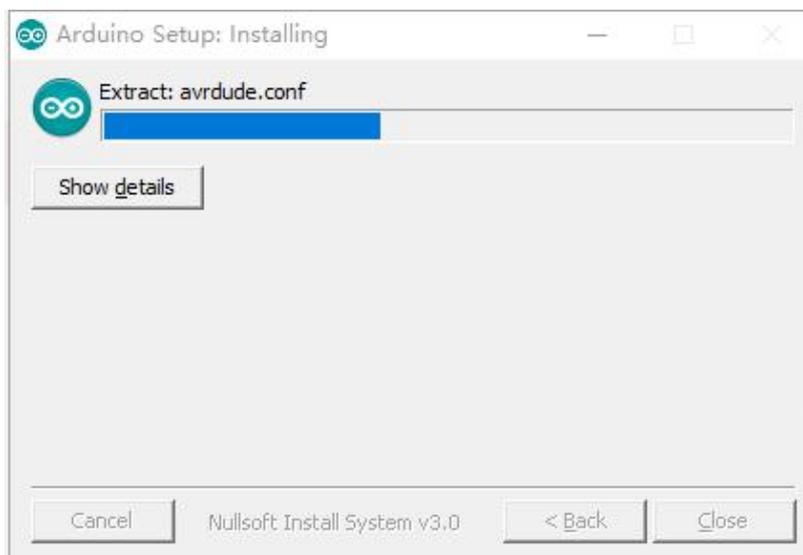


Click Next

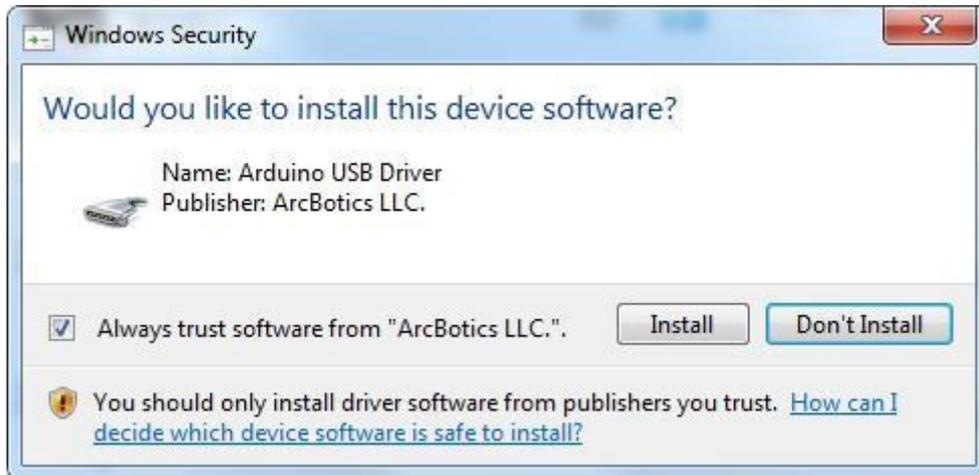


You can press Browse...to choose an installation path or directly type in the directory you want.

1.4 Click Install to initiate installation



Finally, the following interface appears, click Install to finish the installation.



Next, the following icon appears on the desktop



Double-click to enter the desired development environment



1.5 Installing Arduino(Mac OS X)

Download and Unzip the zip file,double click the Arduino.app to enter Arduino IDE;the system will ask you to install Java runtime library if you don't have it in your computer.Once the installation is complete you can run the Arduino IDE.

1.6 Installing Arduino(Linux)

You will have to use the make install command.If you are using the Ubuntu system,it is recommended to install Arduino IDE from the software center of Ubuntu.

2 Add Libraries

2.1 Installing Additional Arduino Libraries

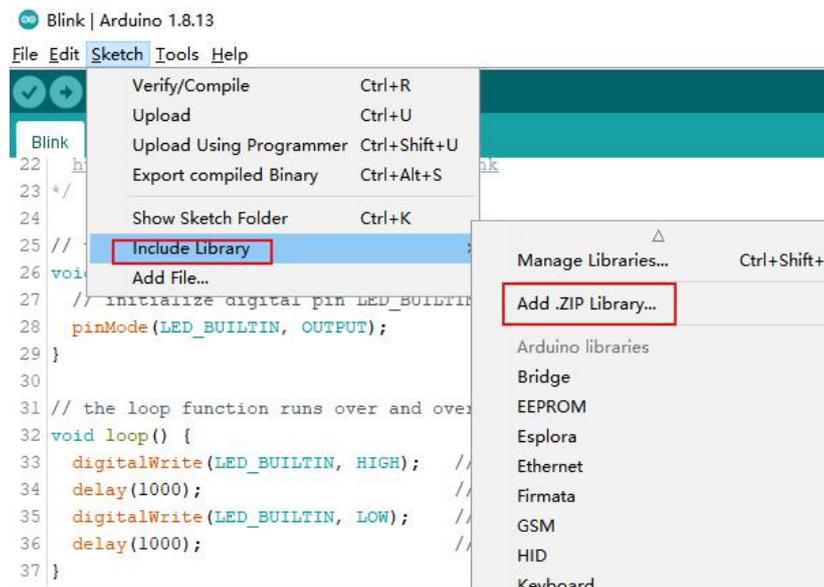
Once you are comfortable with the Arduino software and using the built-in functions, you may want to extend the ability of your Arduino with additional libraries.

2.2 What are Libraries

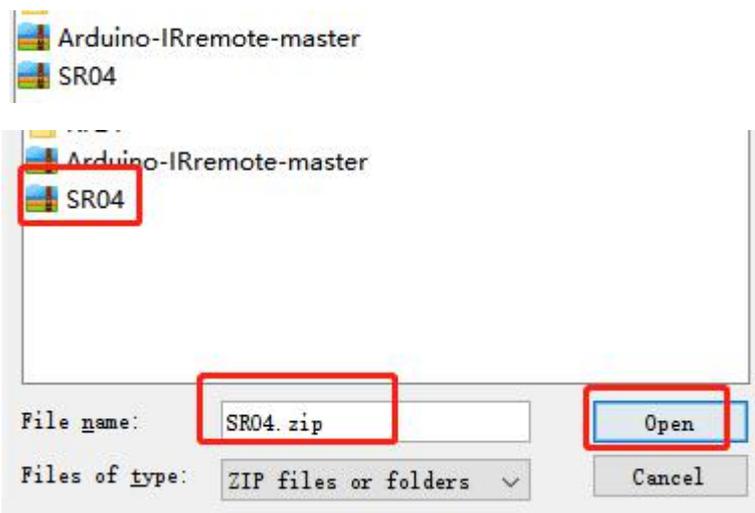
Libraries are a collection of code that makes it easy for you to connect to a sensor, display, module, etc. For example, the built-in LiquidCrystal library makes it easy to talk to character LCD displays. There are hundreds of additional libraries available on the Internet for download. The built-in libraries and some of these additional libraries are listed in the reference. To use the additional libraries, you will need to install them.

2.3 How to Install a Library

Using the Library Manager. To install a new library into your Arduino IDE you can use the Library Manager (available from IDE version 1.8.13). Open the IDE and click to the "Sketch" menu and then Include Library > Manage Libraries.



and then Include Library>Manage Libraries.



2.4 Example:SR04

Open arduino software-project-load library-add a.zip library

2.5 Add method two:

Copy the library folder to the Libraries folder in the Arduino installation directory. Restart Arduino and the added library will take effect.

3 Ultrasonic transducer

3.1 Introduction

Ultrasonic waves are produced by mechanical vibration and can travel at different speeds in different media.

Ultrasonic wave is often used to measure distance because of its strong directivity, slow energy loss and long propagation distance in the medium.

Such as rangefinder and level measuring instrument can be realized through ultrasonic.



Ultrasonic ranging is a non-contact detection method.

Compared with other methods, such as electromagnetic or optical methods, it is not affected by light, the color of the object under test, etc.

For the measured objects in the dark, dust, smoke, electromagnetic interference, toxic and other harsh environment has a certain adaptability.

Therefore, it is widely used in liquid level measurement, manipulator control, vehicle automatic navigation, object recognition and other aspects.

Especially applied to air ranging, because of the slow wave velocity in the air, the echo signal contained along the propagation direction of the structure information is easy to detect, has a very high resolution, so its accuracy is higher than other methods;

And ultrasonic sensor has the advantages of simple structure, small size, Signal processing reliability and other characteristics.

The use of ultrasonic detection is often more rapid, convenient, simple calculation, easy to achieve real-time control, and in terms of measurement accuracy

can meet the requirements of industrial practical.

There are many methods of ultrasonic ranging, the principle of the system in the ultrasonic measurement is: detect the ultrasonic from the ultrasonic transmitter, the transmission time to the receiver through the gas medium,

Multiply this time by the speed of sound in the gas to give the distance the sound travels.

Ultrasonic transmitter to a certain direction of the transmission of ultrasonic wave, at the same time the MCU began to time, ultrasonic wave in the air transmission, encountered obstacles on the way to return immediately, ultrasonic receiver received the reflected wave immediately stop timing.

According to the time T recorded by the timer, the distance (s) between the launching point and the obstacle can be calculated.

$$\text{formula: } S = \frac{VT}{2}$$

Four factors limit the maximum measurable distance of an ultrasonic system: the amplitude of the ultrasonic wave, the texture of the reflector, the Angle between the reflected and incident sound waves, and the sensitivity of the receiving transducer.

The ability of the receiving transducer to receive the acoustic pulse directly will determine the minimum measurable distance.

3.2 Obstacle avoidance principle

The trigger signal input end (TRIG) will input a high-level signal of more than 10 microseconds, and the ultrasonic transmitter will automatically send 8 40Hz square waves upon receiving the signal. At the same time, the timer will be started. When the sensor receives the echo, the timing will be stopped and the echo signal will be output.

According to the time interval, the distance can be calculated by the formula: distance =(high level time * sound velocity)/2, and the speed of sound propagation in

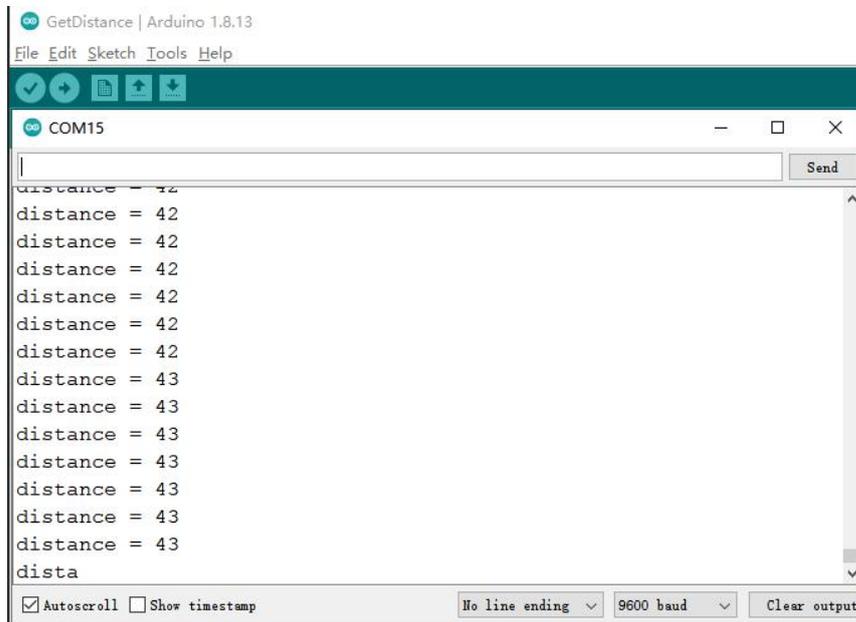
the air is 340m/s.

3.3 Acquisition of measurement data

The measured distance is obtained through the `getDistance ()` function.

```
float GetDistance()  
{  
float distance;  
    digitalWrite(Trig, LOW);  
    delayMicroseconds(2);  
    digitalWrite(Trig, HIGH);  
    delayMicroseconds(10);  
    digitalWrite(Trig, LOW);  
    distance = pulseIn(Echo, HIGH) / 58.00;  
    return distance;  
}
```

The measurement results can be viewed in the Aduino IDE by printing the information through the serial port.

A screenshot of the Arduino IDE serial monitor window. The window title is 'COM15'. The output text shows a series of 'distance = 42' and 'distance = 43' messages, followed by a partial 'dista' message. The bottom of the window shows settings: 'Autoscroll' is checked, 'Show timestamp' is unchecked, 'No line ending' is selected, '9600 baud' is selected, and there is a 'Clear output' button.

```
distance = 42
distance = 43
dista
```

When the distance detected by ultrasonic wave is less than the set distance, the car will be judged to encounter obstacles, and then the obstacle avoidance program will be written to make the car turn backward or turn left to avoid obstacles, so as to realize the car's automatic driving around obstacles.

How to make car automatic obstacle avoidance more flexible?

Of course, the ultrasonic sensor is installed on the steering gear. Through the rotation of the steering gear, the distance between the left and the right and the middle of the car is measured. Comparing the three distances, the maximum value is obtained, and the car is driven to the maximum measured direction.

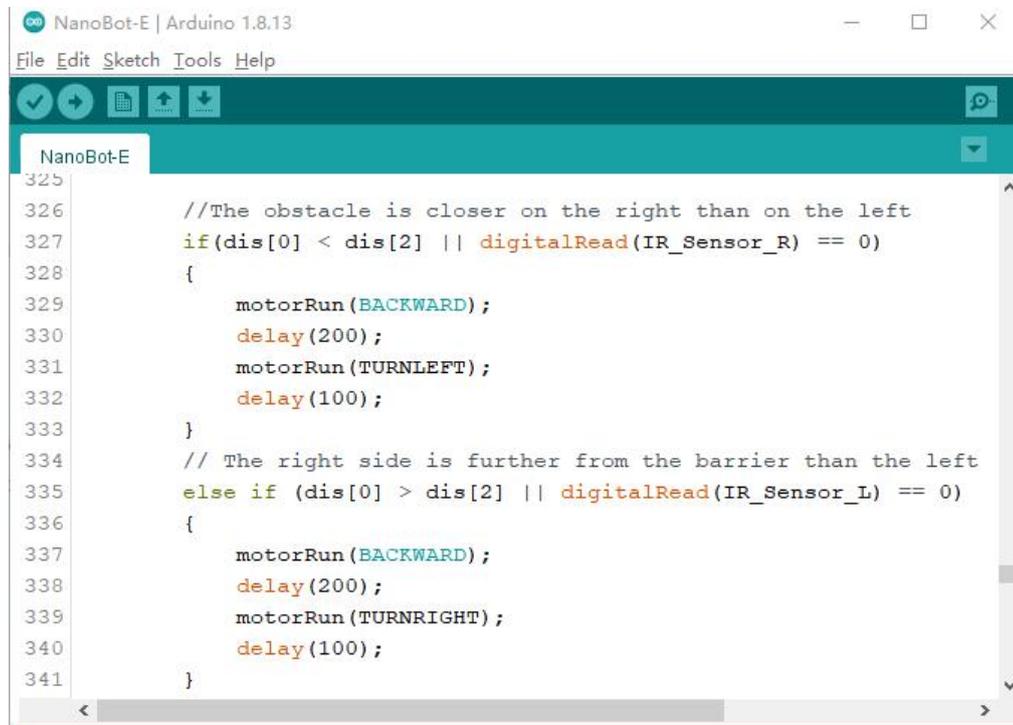
The direction of the car to avoid obstacles will be more accurate.

Car follow function using ultrasonic ranging principle, when in the set distance, the car will follow the front of the items driving, when less than the set range, it will reverse.

Trigger the left infrared obstacle avoidance sensor, the car and servo motor will turn left, trigger the right infrared obstacle avoidance sensor, the car and servo

motor will turn right, to follow.

Part of the procedure is as follows:

A screenshot of the Arduino IDE interface. The window title is 'NanoBot-E | Arduino 1.8.13'. The menu bar includes 'File', 'Edit', 'Sketch', 'Tools', and 'Help'. The toolbar shows icons for check, run, upload, and download. The code editor shows the following code:

```
325
326 //The obstacle is closer on the right than on the left
327 if(dis[0] < dis[2] || digitalRead(IR_Sensor_R) == 0)
328 {
329     motorRun(BACKWARD);
330     delay(200);
331     motorRun(TURNLEFT);
332     delay(100);
333 }
334 // The right side is further from the barrier than the left
335 else if (dis[0] > dis[2] || digitalRead(IR_Sensor_L) == 0)
336 {
337     motorRun(BACKWARD);
338     delay(200);
339     motorRun(TURNRIGHT);
340     delay(100);
341 }
```

A screenshot of the Arduino IDE interface. The window title is 'NanoBot-E'. The menu bar includes 'File', 'Edit', 'Sketch', 'Tools', and 'Help'. The toolbar shows icons for check, run, upload, and download. The code editor shows the following code:

```
397 else if(get_dis > 20 && get_dis < 40)
398 {
399     myservo.write(100);
400     motorRun(FORWARD);
401 }
402 //Backup less than 15 cm
403 else if(get_dis < 15 )
404 {
405     myservo.write(100);
406     motorRun(BACKWARD);
407 }
408 else
409 {
410     myservo.write(100);
411     motorRun(STOP);
412 }
```

4 Three-way tracking module

4.1 Introduction

Let the car walk along the black line, need infrared sensor to detect the black line of the floor, through the black line and the white floor detection, detection results are transmitted to the single chip microcomputer, data processing and comparison, and then control the direction of the motor, so that the car along the black line driving.

4.2 Principle Introduction:

4.2.1 Photodiode

Infrared emitting tube belongs to a class of photodiodes. When the positive pole of the photodiode is connected to the positive voltage, and the negative pole is connected to the negative pole of the power supply, the photodiode will emit light.

Ordinary light-emitting diodes, such as LEDs, emit visible light, which can be directly seen by human eyes, but infrared emitting tubes emit infrared light, which is invisible to human eyes. Special equipment is needed to detect infrared light.

The phone's camera can detect the infrared light from the tube and see the purplish red light.

4.2 Phototriode

Infrared receiving tube is also called photoelectric triode. Photoelectric triode is also called photosensitive triode. Its current is controlled by external illumination. It is a kind of semiconductor photoelectric device.

A photodiode is a triode in which a photodiode is connected between the base

of the audion and the collector, and the current of the photodiode is equivalent to the base current of the diode.

Because of its current amplification effect, the phototriode is much more sensitive than the photodiode and can output a large photocurrent at the collector.

Tracking the car generally using infrared detection method, using infrared light in different colors of the characteristics of the object surface with different reflection properties, in the process of car driving, infrared emission infrared tracking sensor to the ground, when met white paper floor infrared diffuse reflection occurs, the reflected light comes on car receiving tube;

If the black line is encountered, the infrared light is absorbed, and the receiving tube on the car can not receive the infrared light. The output voltage of the receiving tube will increase, and if a white or reflective object is detected, the output voltage of the receiving tube will decrease.

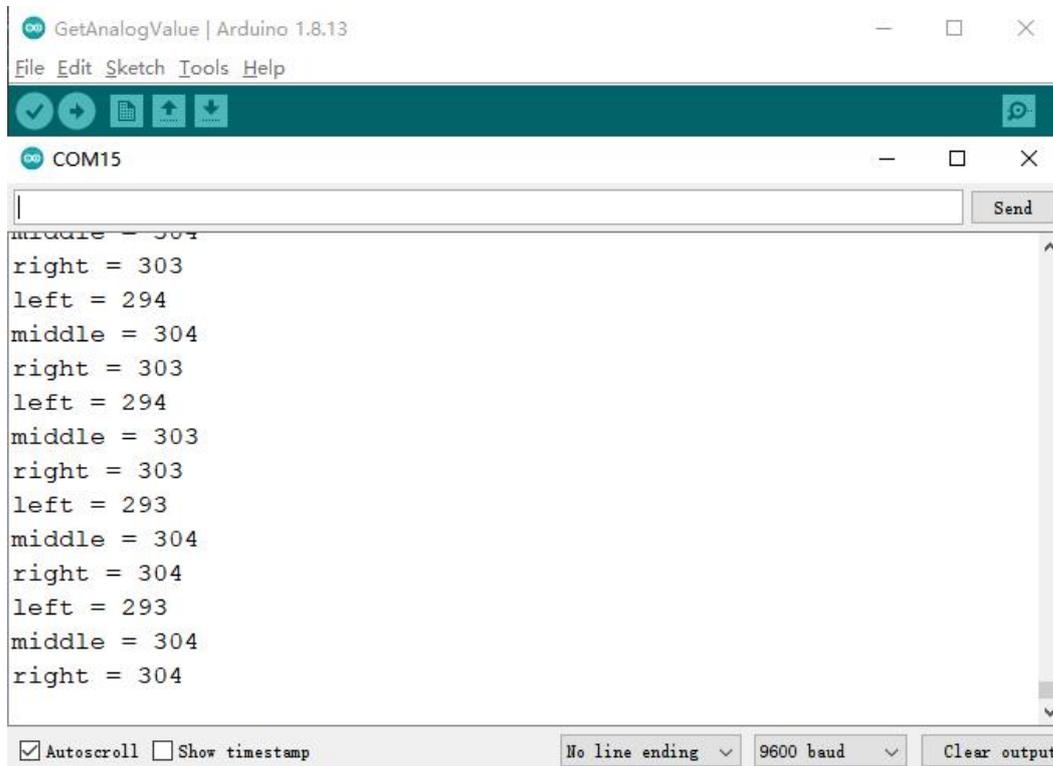
The detected data is sent to the processor for processing, so as to determine whether the infrared light reflected back is received to determine the position of the black line and the car's walking route.

Infrared detector detection range is limited, generally the maximum should not exceed 2cm, the greater the detection distance, detection ability will become smaller.

Note that since the sun also emits infrared light, sunlight has a certain effect on the detected signal, and should be avoided during use.

4.3 get data

The `analogRead(PIN)` function is used to read the voltage of the phototriode. The UNO R3 board has a multi-channel 10-bit analog to digital converter. The `AnalogRead` function is used to obtain integer values between 0 and 1023. A 5V voltage is converted to 1023, and a 0V voltage is converted to 0. Reading accuracy: 5 volts / 1024 units, equal to about 0.049 volts (4.9 mV) per unit.



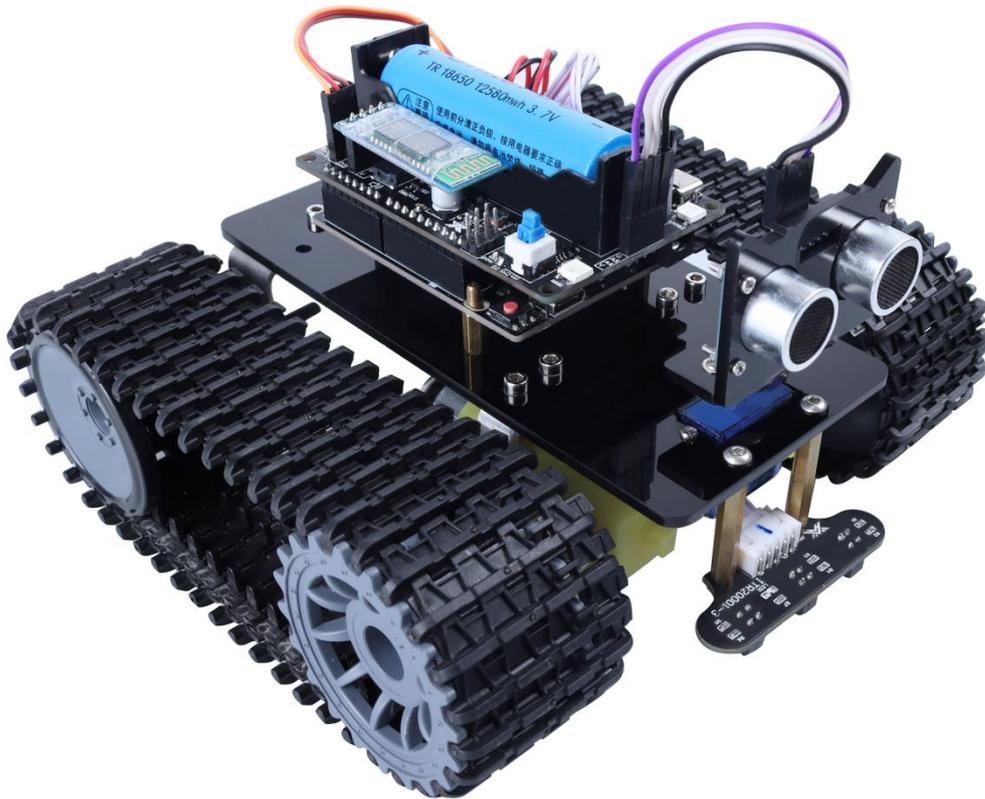
The tracking mode of the car adopts three-way tracking module. The picture is as follows:



Wiring table is as follows:

UNO	Three road tracking
5V	VCC
GND	GND
A0	AOUT1(left) ----->IR1
A1	AOUT2(middle) ----->IR2
A2	AOUT3(right) ----->IR3

Three - way tracking module mounted on the trolley.



5 buzzer

5.1 Introduction

According to the driving mode, it can be divided into active buzzer and passive buzzer.

Active and passive here do not refer to power sources, but oscillating sources. The active buzzer has an oscillating source inside it, so give the Buzz pin a low level and the buzzer will ring directly.

And the passive buzzer is not with an oscillating source, to make him ring must be given between 500Hz ~ 4.5KHz pulse frequency signal to drive it will ring.

Active buzzer driver simple pronunciation, by the level can be driven; Passive buzzer sound frequency can be controlled, and the scale and frequency have a definite corresponding relationship, so you can make the effect of "Doremifasollasi", can be used to make simple music tracks, such as birthday song, two tigers and soon.

6 Bluetooth

6.1 Introduction

The Bluetooth core module uses HC-06 from the module, and the lead interface includes VCC,GND,TXD and RXD. The LED state output port is reserved, and the MCU can judge whether the Bluetooth has been connected through the state of the foot.



LED indicates Bluetooth connection status, flashing means there is no Bluetooth connection, constantly on means Bluetooth has been connected and the port has been opened.

The input voltage is 3.6~6V, the current is about 30mA when it is not paired, and about 10mA after pairing. The input voltage is forbidden to exceed 7V!

Can be directly connected to a variety of microcontroller (51, AVR, PIC, ARM, MSP430, etc.), 5V microcontroller can also be directly connected.

When Bluetooth connection is not established, it supports setting baud rate, name and pair password through AT instruction. The parameters set are saved when power is off.Switch to pass through mode automatically after Bluetooth connection.



6.2 Bluetooth module wiring

Wiring table is as follows

UNO	HC06
VCC	5V
GND	GND
D1(TX)	RX
D0(RX)	TX

When connecting the Bluetooth module, please pay attention to whether the connection of the wire is correct and check the connection once before energizing. In addition, do not short-circuit the pins, so in order to avoid unnecessary losses, please pay attention here.

6.3 Write a program

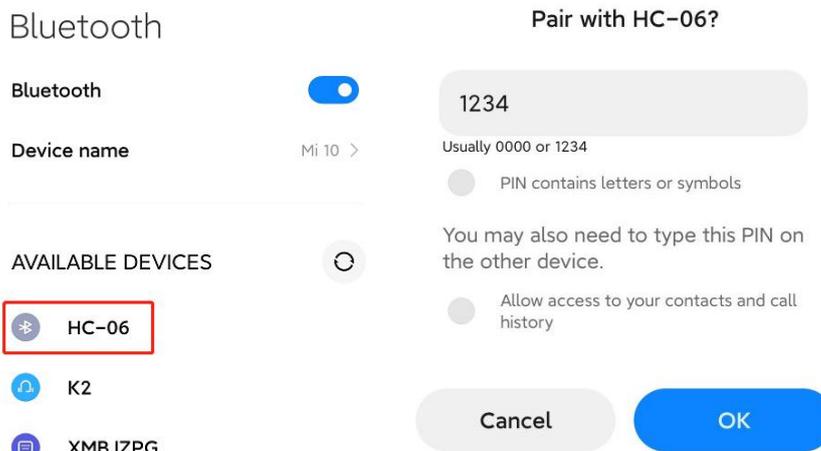
Note that you need to unplug the Bluetooth module or press the power switch of the extension board when uploading the program, otherwise it will not be burned. Why?

Because the serial port is used to burn the program, the UNO board only has this set of serial ports (TX, RX), and it is impossible to download the program after connecting the Bluetooth module.

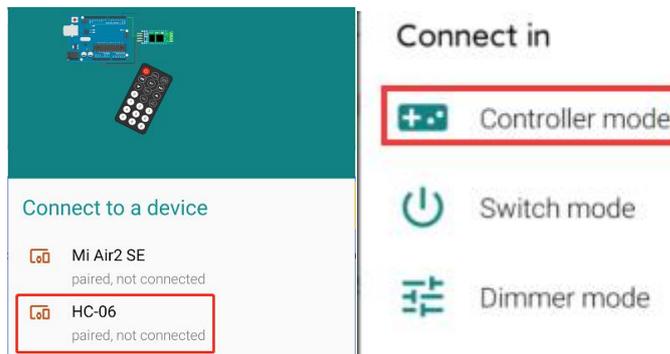
6.4 Installing mobile software

After installation on your phone, turn on Bluetooth and connect to HC-06. Then open the phone software and connect to HC-06.

Equipped with Bluetooth serial port assistant for remote control of the car;First, turn on Bluetooth and search for Bluetooth devices to find HC-06 connection. The default connection pairing password is 1234.



Open the Bluetooth Assistant and find the name of the Bluetooth module HC-05 that has just been configured. After successful connection, a remote control main interface will appear. Select the Settings button in the upper right corner to set control commands.



6.5 The command sets the reference table

command	function
F	forward
B	back
L	Turn left
R	Turn right
S	stop

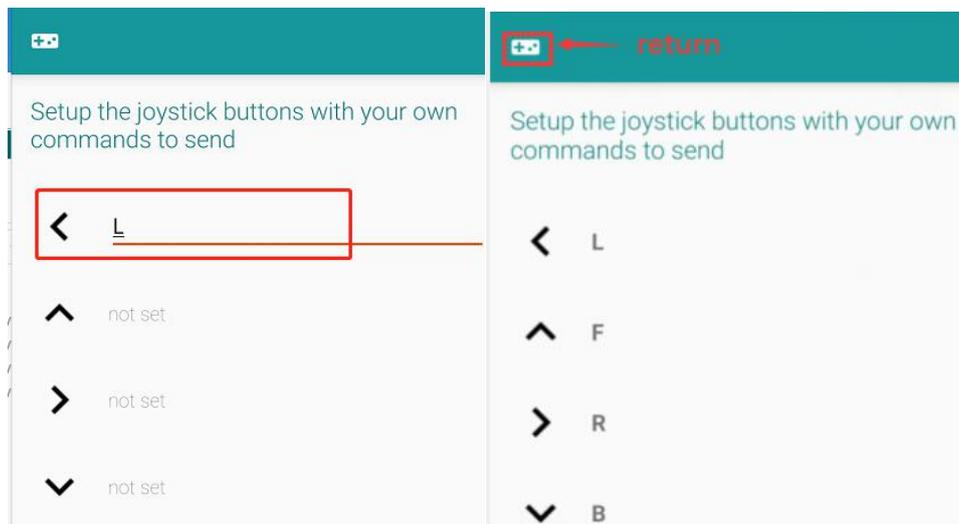
A	Obstacle avoidance
---	--------------------

Command Settings, such as L for moving car to the left.

Control command configuration completed select the icon in the upper left corner to return to the control interface.



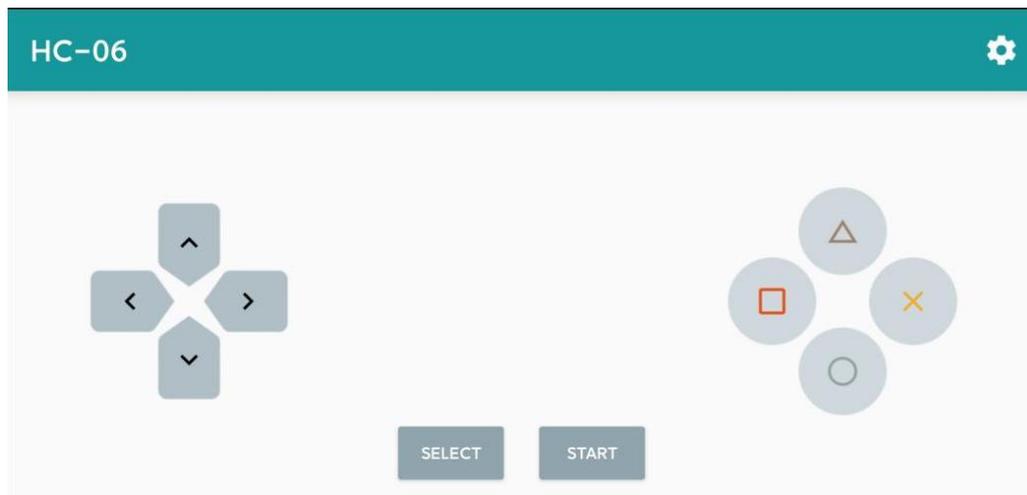
Command setting, such as L for the car to move to the left, after the input of the command, you need to press the Enter key to take effect. Control command configuration completed select the icon in the upper left corner to return to the control interface.



If the commands of L,F,R,B and so on configured just now are consistent with those in the program, the car can be successfully remote controlled, otherwise the remote control fails;

```
50 void bluetooth()
51 {
52     char serialData;
53     static char cmd;
54
55     if (Serial.available() > 0)
56     {
57         serialData = Serial.read();
58         Serial.println(serialData);
59
60         SetSeep(serialData);
61
62         if ('F' == serialData )
63         {
64             motor (forward, MotSpeed1, MotSpeed2);
65         }
66
67         else if ('B' == serialData )
68         {
69             motor (back, MotSpeed1, MotSpeed2);
70         }
71     }
```

Press the forward button and the car will move forward, release it and still move forward. Press the stop button to stop! Control car other directions are the same operation;



7 Infrared remote control

7.1 Principle of infrared remote control

Infrared remote control components: infrared emission and infrared reception.

Infrared transmission and reception of the signal, the binary pulse code, high and low level in accordance with a certain time law transformation to transmit the corresponding information.

In order to avoid interference from other signals in the wireless transmission process, the signal is usually modulated on a specific carrier frequency (38K infrared carrier signal) and emitted through an infrared emitting diode, while the infrared receiver will demodulate the signal and restore it to binary pulse code for processing.

The infrared receiver head has three pins, as shown in the figure above, which are VOUT, GND and VCC from left to right.

The 38K infrared carrier signal emitted by the infrared remote controller is encoded by the coding chip in the remote controller.

When the key of the remote control is pressed, the remote control sends out an infrared carrier signal, and the infrared receiver receives the signal. The program decodes the carrier signal and determines which key is pressed through the difference of data encoding.

The program needs to install the IRRemote library. When the infrared remote controller is pressed, the key codes of different keys can be seen in the serial monitor. The key code values are recorded and the required key code values are selected for conditional control, so the remote control car function can be realized.

7.2 routine is as follows:

```
/*
 * IRrecvDemo
 * Infrared control, receiving infrared commands to control motor movement
 */
#include <IRremote.h>
IRrecv irrecv(RECV_PIN);
decode_results results;
void setup()
{
    Serial.begin(9600);
    irrecv.enableIRIn(); // Start the receiver

    pinMode(5, OUTPUT);
    pinMode(6, OUTPUT);
    digitalWrite(5, LOW);
    digitalWrite(6, LOW);
}

void loop()
{
    if (irrecv.decode(&results))
    {
        Serial.print("value = ");
        Serial.println(results.value, HEX);
        if (results.value == 0xFF02FD) //Positive motor rotation value
        {
            motor3(255,0);
        }
    }
}
```

```
    }  
    else if (results.value == 0xFF9867) //Motor reversal value  
    {  
        motor3(0,255);  
    }  
    else if (results.value == 0xFFA857) //Motor stop value  
    {  
        motor3(0,0);  
    }  
    irrecv.resume(); // Receive the next value  
}  
}  
void motor3(int speed1, int speed2)  
{  
    analogWrite(5,speed1);  
    analogWrite(6,speed2);  
}
```

8 servo

8.1 principle is introduced

Servo motor control pulse signal cycle is 20MS pulse width modulated signal (PWM), pulse width from 0.5ms to 2.5ms, the corresponding steering position from 0 to 180 degrees, linear change.

That is to say, if the steering gear is provided with a certain pulse width, its output shaft will maintain a certain corresponding Angle. No matter how the external torque changes, it will not change the output Angle to a new corresponding position until another pulse signal is provided to it.

There is a reference circuit inside the steering gear, generating the pulse signal with a period of 20MS and a width of 1.5ms. There is a comparator, comparing the external signal with the reference signal, and judging the direction and size, so as to generate the motor rotation signal.

The steering gear is a position servo drive with a rotation range of no more than 180 degrees.

8.2 Servo.h library

With the Servo library, you can control the Servo with Arduino. The Servo library allows most Arduino development boards (such as the Arduino Uno) to control up to 12 servos simultaneously.

Please note that using the Servo library may affect the PWM functionality of some pins on the Arduino development board. When you use the Servo library, the PWM function on pins 9 and 10 of the development board is not available. In other words, if your Arduino program uses the Servo library, the pins 9 and 10 cannot be controlled using the `analogWrite()` statement, regardless of whether the Servo is attached to the pins.

8.3 Why can't the analogWrite() statement be used?

The Servo library supports up to 12 motors on most Arduino boards and 48 on the Arduino Mega. On boards other than the Mega, use of the library disables analogWrite() (PWM) functionality on pins 9 and 10, whether or not there is a Servo on those pins. On the Mega, up to 12 servos can be used without interfering with PWM functionality; use of 12 to 23 motors will disable PWM on pins 11 and 12.

8.4 Control the SERVO through the SERVO library

The steering gear generally has three leads, which are the power line, the ground line and the signal line.

The power cord is usually red. You can use the 5V pins of the Arduino development board to power the steering gear. The ground wire is usually brown or black, and is usually connected to the ground pin of the Arduino development board.

Signal lines are usually orange, yellow or white. The signal wire is connected to the digital 3-pin of the Arduino development board.

Control example procedures are as follows:

```
#include <Servo.h>

Servo myservo; // create servo object to control a servo
// twelve servo objects can be created on most boards
int pos = 0; // variable to store the servo position

void setup() {
  myservo.attach(3); // attaches the servo on pin 9 to the servo object
}

void loop() {
  for (pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180 degrees
```

```
// in steps of 1 degree
myservo.write(pos);    // tell servo to go to position in variable 'pos'
delay(15);             // waits 15ms for the servo to reach the position
}
for (pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0 degrees
  myservo.write(pos);    // tell servo to go to position in variable 'pos'
  delay(15);             // waits 15ms for the servo to reach the position
}
}
```

If the Servo is used with the number 9 and number 10 pins, you need to customize one of the Servo's drive functions.

Using the following custom routine, we can resolve the situation where the number pins 9 and 10 are invalidated using the analogWrite() statement.

```
#define servoPin  3
int pos = 0;     // variable to store the servo position
void setup()
{
  Serial.begin(9600);
  pinMode(servoPin,OUTPUT);
}
void loop()
{
  for (pos = 0; pos <= 180; pos += 1)
  { // goes from 0 degrees to 180 degrees
    // in steps of 1 degree
    servo(pos);    // tell servo to go to position in variable 'pos'
    delay(15);     // waits 15ms for the servo to reach the position
  }
}
```

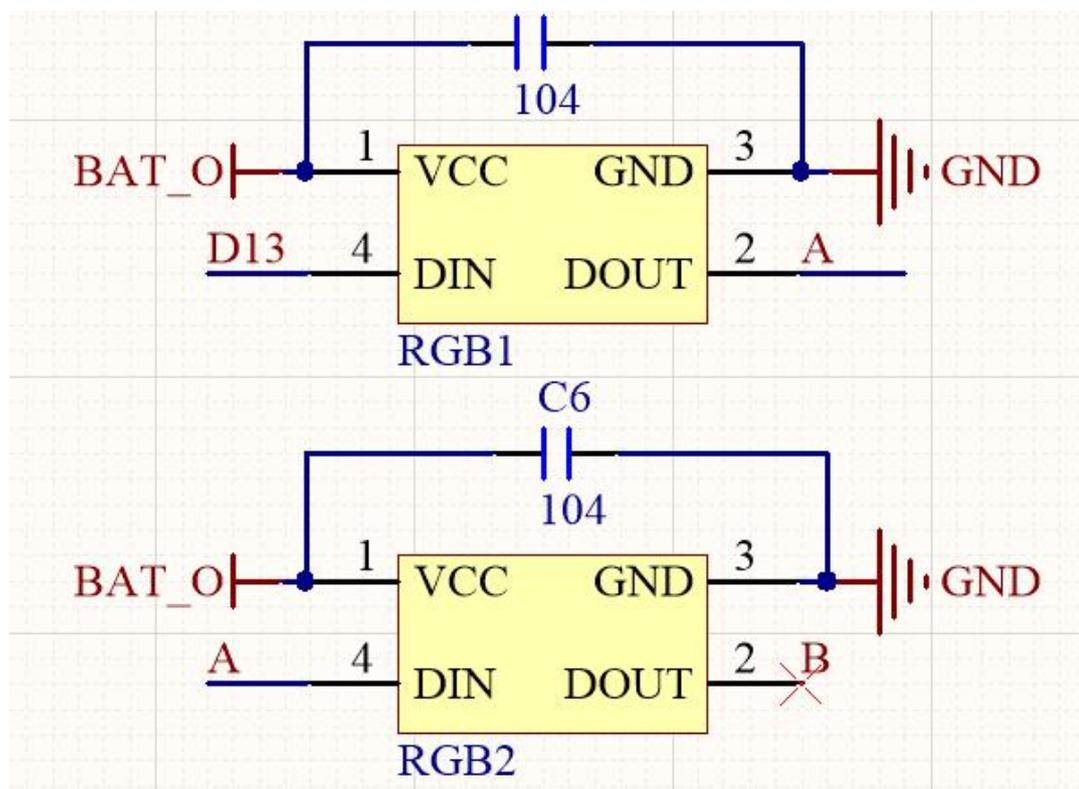
```
for (pos = 180; pos >= 0; pos -= 1)
{ // goes from 180 degrees to 0 degrees
  servo(pos); // tell servo to go to position in variable 'pos'
  delay(15); // waits 15ms for the servo to reach the position
}
}
```

9 WS2812B

9.1 Introduction

WS2812B is a chip with a built-in LED driver. One IO port can control multiple LEDs, brightness adjustment, color adjustment and other functions.

2 RGB are integrated on the extension board, which is used as the R indicator light of the car driving direction. The control pin of this RGB lamp is connected to the number 13 pin of the UNO board.



9.2 characteristics

1. The control circuit and RGB chip are integrated in a 5050 package component, forming a complete external control pixel point.
2. Built-in signal shaping circuit, any pixel after receiving the signal through the waveform

3. Type output, ensure that the line waveform distortion will not accumulate.
4. Built-in power on reset and power off reset circuit.
5. The three primary colors of each pixel can achieve 256-level brightness display, complete the full true color display of 16777216 colors, and the scanning frequency is not less than 400Hz/s.
6. Serial level connection port, through a signal line to complete the data reception and decoding.
7. Any two-point transmission distance does not need to increase any circuit when it is less than 5 meters.
8. When the refresh rate is 30 frames/SEC, the cascade number of low speed mode is not less than 512 points, and high speed mode is not less than 1024 points. Data transmission speed up to 800Kbps.
9. The color of the light is highly consistent and cost-effective.

9.3 test program

```
#include <Adafruit_NeoPixel.h>

// Which pin on the Arduino is connected to the NeoPixels?
#define PIN 13 // On Trinket or Gemma, suggest changing this to 1
// How many NeoPixels are attached to the Arduino?
#define NUMPIXELS 2 // Popular NeoPixel ring size
// When setting up the NeoPixel library, we tell it how many pixels,
// and which pin to use to send signals. Note that for older NeoPixel
// strips you might need to change the third parameter -- see the
// strandtest example for more information on possible values.
Adafruit_NeoPixel pixels(NUMPIXELS, PIN, NEO_GRB + NEO_KHZ800);

#define DELAYVAL 500 // Time (in milliseconds) to pause between pixels
```

```
void setup()
{
    Serial.begin(9600);
    pixels.begin(); // INITIALIZE NeoPixel strip object (REQUIRED)
    Serial.println("Initialization completed!");
}

void loop()
{
    pixels.clear(); // Set all pixel colors to 'off'
    // The first NeoPixel in a strand is #0, second is 1, all the way up
    // to the count of pixels minus one.
    for(int i=0; i<NUMPIXELS; i++) { // For each pixel...
        // pixels.Color() takes RGB values, from 0,0,0 up to 255,255,255
        // Here we're using a moderately bright green color:
        pixels.setPixelColor(i, pixels.Color(0, 150, 0));
        pixels.show(); // Send the updated pixel colors to the hardware.
        delay(DELAYVAL); // Pause before next pass through loop
    }
}
```

10 code analysis

This section describes the programming.

10.1 Pin configuration.

```
// Define ultrasonic sensor pins
#define Trig A4
#define Echo A5

//Three-way tracking module pin definition
#define LeftPin    A0
#define MiddlePin  A1
#define RightPin   A2

#define servoPin   3
#define BEEP       11

// Which pin on the Arduino is connected to the NeoPixels?
#define PIN 13 // On Trinket or Gemma, suggest changing this to 1
```

10.2 Define car parameters

```
//Define infrared remote coding function
#define FORWARD    0xFF02FD
#define BACKWARD   0xFF9867
#define TURNLEFT   0xFFE01F
#define TURNRIGHT  0xFF906F
#define STOP       0xFFA857
#define Buzzer     0xFF7A85
#define Avo_Obs    0xFF18E7
#define TRACK      0xFF30CF
|

//Define the direction of car movement
#define stop       0
#define forward   1
#define back      2
#define left      3
#define right     4
```

Define the PWM speed regulation value. The default speed regulation value is within the range of 0~255. In practice, the motor will not rotate if it is lower than 100, which is related to battery voltage and motor.

Under normal circumstances, the speed should not be set too low to avoid the motor does not rotate;

```
//Set motor default speed
int MotSpeed1 = 200;
int MotSpeed2 = 200;
```

10.3 setup function

void setup() function will be executed only once during the program running, and the serial port baud rate can be configured only once. The baud rate is set to 9600, which is the same as the baud rate of Bluetooth module and serial port monitor, so that data can be transmitted to and from each other.

```
void setup()
{
    Serial.begin(9600);
    irrecv.enableIRIn();
    // The initial Angle of the steering gear
    servo(50);
    //Set the pin mode
    for (int i = 4; i <= 12; i++)
    {
        pinMode(i, OUTPUT);
        digitalWrite(i, LOW);
    }
    pinMode(LeftPin, INPUT);
    pinMode(MiddlePin, INPUT);
    pinMode(RightPin, INPUT);
    pinMode(Trig, OUTPUT);
    pinMode(Echo, INPUT);
    Init();
    pixels.begin(); // INITIALIZE NeoPixel strip object (REQUIRED)
    //beep();
    Serial.println("Initialization completed!");
}
```

10.4 Motor drive

The void motor1() and void motor2() functions are used to drive the first and second DC motors. Since the digital pins used have no PWM function, they are called only with the pin state set to high or low.

```
void motor1(bool PinState1, bool PinState2)
{
    digitalWrite(12, PinState1);
    digitalWrite(4, PinState2);
}

void motor2(bool PinState1, bool PinState2)
{
    digitalWrite(7, PinState1);
    digitalWrite(8, PinState2);
}
```

The void motor3() and void motor4() functions are used to drive the third and fourth DC motors. The motor can be speed-adjusted due to the PWM function of the digital pins used. The speed range is 0 to 255.

```
void motor3(int speed1, int speed2)
{
    analogWrite(5, speed1);
    analogWrite(6, speed2);
}

void motor4(int speed1, int speed2)
{
    analogWrite(9, speed1);
    analogWrite(10, speed2);
}
```

The void motor() function, which encapsulates the motor3() and motor4() functions, can control the motor3 and motor4 at the same time, and the driving direction of the car is set by the DIR parameter, and the speed is set from 0 to 255.

At the same time also includes RGB color setting function function, according to the direction of the car to change the RGB light color.

```
/*
M3 M4 motor drive
M3 -----> speed1
M4 -----> speed2
Speed adjustment range: 0~255
*/
void motor(int dir ,int speed1, int speed2)
{
    switch (dir)
    {
        case stop:
            motor3(0, 0);
            motor4(0, 0);
            pixels.clear(); // Set all pixel colors to 'off'
            pixels.setPixelColor(0, pixels.Color(0, 0, 0));
            pixels.setPixelColor(1, pixels.Color(0, 0, 0));
            pixels.show(); // Send the updated pixel colors to the hardware.
            break;

        case forward:
            motor3(speed1, 0);
```

10.5 Infrared remote control

IR is the function of the infrared remote control, which is used to obtain the coding value of the infrared remote control. The coding of each key of the remote control is unique, and the key function of the remote control can be customized.

Use the process control statement switch statement to select a custom function.

```
void IR()
{
  //Check whether infrared remote control signal is received
  if (irrecv.decode(&results) )
  {
    // The received value is stored in res_val
    unsigned long res_val = results.value;

    Serial.print("res_val=");
    // Output instruction information
    Serial.println(res_val,HEX);

    irrecv.resume();// Wait for the next reception

    switch(res_val)
    {
      case FORWARD:
        motor(forward,MotSpeed1,MotSpeed2);
        break;
    }
  }
}
```

10.6 Ultrasonic sensor ranging

The `getDistance ()` function is used to measure the distance, the driving function of the ultrasonic sensor.

Using IO trigger ranging, to give at least 10uS high level signal;

The module automatically sends 8 square waves of 40KHz to automatically detect whether there is a signal returned;

A signal is sent back, and a high level is output through IO. The duration of the high level is the time from transmission to return of the ultrasonic wave.

Distance =(high level time * sound velocity (340m/s))/2;

When the pin "TRIG" of the ultrasonic ranging sensor is high, it sends out ultrasonic waves. In order to ensure the sound wave of 10μs, the pin of TRIG is required to be pulled down before sending and delay for a short period of time. The code of the ranging function is as follows:

```
/*
Function: obtain ultrasonic sensor ranging data
Parameters: Trig, Echo
Parameter description: sensor connected to the motherboard pin port A4,A5
Trig -----> pin A4
Echo -----> pin A5
*/
float GetDistance()
{
    float distance;

    digitalWrite(Trig, LOW);
    delayMicroseconds(2);
    digitalWrite(Trig, HIGH);
    delayMicroseconds(10);
    digitalWrite(Trig, LOW);

    distance = pulseIn(Echo, HIGH) / 58.00;

    return distance;
}
```

In the automatic obstacle avoidance mode, the avoidance distance was modified by the parameters of avoidance(int set_dis) function. For example, 30cm was set, and the parameters of the function were changed: voidance(30).

10.7 serial port communication

Bluetooth () function as the function of Bluetooth remote control car, calls a key function Serial. Available (),

The serial.available () function returns the number of characters currently left in the Serial port buffer.

In this program, use this function to determine whether the Serial port buffer is data, when Serial.available() ;

0 indicates that the Serial port has received data and can read it: Serial.read() refers to taking out a Byte data from the buffer of the Serial port and reading it. For example, in the Bluetooth Assistant APP, press the forward button, then the Serial.read() function can read the sent data and judge it as "F" to make the car move forward.

```
void bluetooth()
{
    char serialData;
    static char cmd;

    if(Serial.available() > 0)
    {
        serialData = Serial.read();
        Serial.println(serialData);

        SetSeep(serialData);

        if ('F' == serialData )
        {
            motor(forward, MotSpeed1, MotSpeed2);
        }

        else if('B' == serialData )
        {
            motor(back, MotSpeed1, MotSpeed2);
        }
    }
}
```

10.8 Set motor speed

If you are adjusting the speed of the motor, use the SetSeep() function.

```
void SetSeep(char val)
{
    if(val == '+')
    {
        if( MotSpeed1 < 255)
        {
            MotSpeed1 += 10;
            MotSpeed2 = MotSpeed1;

            if(MotSpeed1 > 255) //Prevent exceeding 255
            {
                MotSpeed1 = 255;
                MotSpeed2 = 255;
            }
        }
        Serial.print("speed = ");
        Serial.println(MotSpeed1);
    }
    if (val == '-')
```

10.9 Servo motor drive

Customizing servo drive function `servo()`. Since digital pins 9 and 10 use the same timer as the steering gear store, using the steering gear store to drive the steering gear while also using pins 9 and 10 to use the `analogWrite()` function will fail. So we need to define a custom servo drive function to solve this problem.

```
//Customize a steering gear function
void servo(int angle)
{
    //Sends 5 pulses
    for(int i=0;i<5;i++)
    {
        //Convert the Angle to a pulse width of 500-2480
        int pulsewidth = (angle * 11) + 500;
        //Turn steering gear interface level to high
        digitalWrite(servoPin, HIGH);
        //The number of microseconds of delay pulse width value
        delayMicroseconds(pulsewidth);
        //Turn steering gear interface level to low
        digitalWrite(servoPin, LOW);
        delayMicroseconds(20000 - pulsewidth);
    }
}
```

10.10 print Data

The `print Data ()` information printing function, which prints every 1000 milliseconds.

The delay function is not used here, because the delay function will make the operation of the whole system become stalled, which has a great impact on the real-time task. For example, the tracking of the car is a real-time task, and it needs to detect the black line on the ground at all times, otherwise the car will deviate from the driving route.

```
void PrintData()
{
    static unsigned long PrintTime = 0;

    //Print information every second
    if (currentMillis - PrintTime >= 1000)
    {
        PrintTime = currentMillis;
        Serial.print("left = ");
        Serial.println(analogRead(LeftPin));
        Serial.print("middle = ");
        Serial.println(analogRead(RightPin));
        Serial.print("right = ");
        Serial.println(analogRead(RightPin));
        MeasurVoltage();
    }
}
```

To use millis() for timing and delay, you need to record and store the time of the start of the operation, and then periodically check to see if the defined time has elapsed. Stores the current time in a variable.

```
void loop()
{
    currentMillis = millis();

    IR();
    bluetooth();

    PrintData();

    // pixels.Color() takes RGB values, from
    // Here we're using a moderately bright (
```

10.11 Three road tracking

The principle of tracking is based on the return voltage on the pin corresponding to the tracking module, and the voltage value is read by the AnalogRead function.

When the black line is encountered, the voltage value increases while the white voltage value decreases, so as to determine whether the black line is encountered.

According to which position the module touches the black line, the motor makes the corresponding execution action.

```
void patrol()
{
    if(analogRead(LeftPin) > 100 && analogRead(MiddlePin) < 500 &&
        analogRead(RightPin) < 500)
    {
        motor(left,180,180);
    }
    else if(analogRead(LeftPin) < 400 && analogRead(MiddlePin) > 100 &&
        analogRead(RightPin) < 500)
    {
        motor(forward,160,160);
    }
    else if(analogRead(LeftPin) < 400 && analogRead(MiddlePin) < 400 &&
        analogRead(RightPin) > 100)
    {
        motor(right,180,180);
    }
    else motor(stop,0,0);
}
```

The principle of each part of the car, the program has been introduced, then begin to write procedures, let our car run!