ZHIYI智懿

McKnum Wheel robotic arm Car

V2.02.2.816

About ZHIYI.Ltd

ZHIYI.Ltd is a professional company engaged in UNO , NANO A company that develops and sells control boards and various modules or sensors for Arduino . We have also designed starter kits for hobbyists of different levels to learn Arduino. We are located in Shenzhen , China, all our products meet international quality standards and are highly appreciated in various different markets around the world.

Customer service

Establishing cooperation with companies from different countries and regions , and also helping them source high-quality electronic components in China . We look forward to establishing cooperative relations with more companies in the future. If you have any comments or suggestions , please contact us by email robot@zhiyi.ltd

Content

Lesson 1 Install Arduino IDE

Introducing Arduino

Arduino is an open source electronics platform based on easy-to-use hardware and software. Suitable for anyone working on interactive projects . Generally speaking, an Arduino project is composed of hardware circuits and software codes.

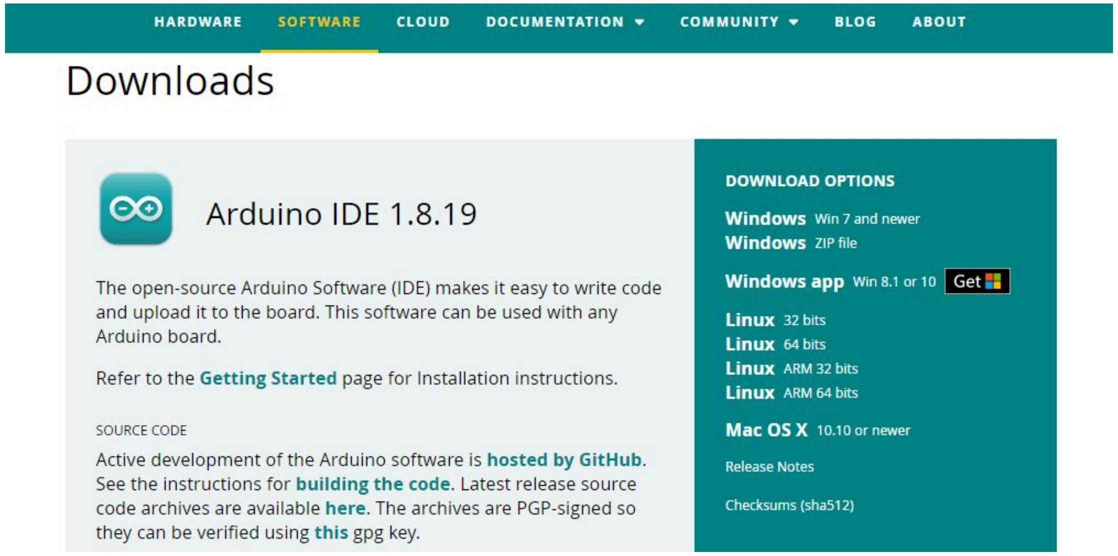
Arduino Board

Arduino Board is a circuit board that integrates a microcontroller, input and output interfaces, etc. The Arduino Board can sense the environment using sensors and receive user actions to control LEDs, motor rotation, and more. We only need to assemble the circuit and write the code for burning to make the product we want . Currently, there are many models of Arduino Boards, and the code is common between different types of boards (due to differences in hardware, some boards may not be fully compatible).

Arduino software

Arduino Integrated Development Environment (IDE) is the software side of the Arduino platform. For writing and uploading code to the Arduino Board. Follow the tutorial below to install the Arduino software (IDE) .

Step 1: Click to go to https://www.arduino.cc/en/software webpage and find the following webpage location:



Downloads

Arduino IDE 1.8.19

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

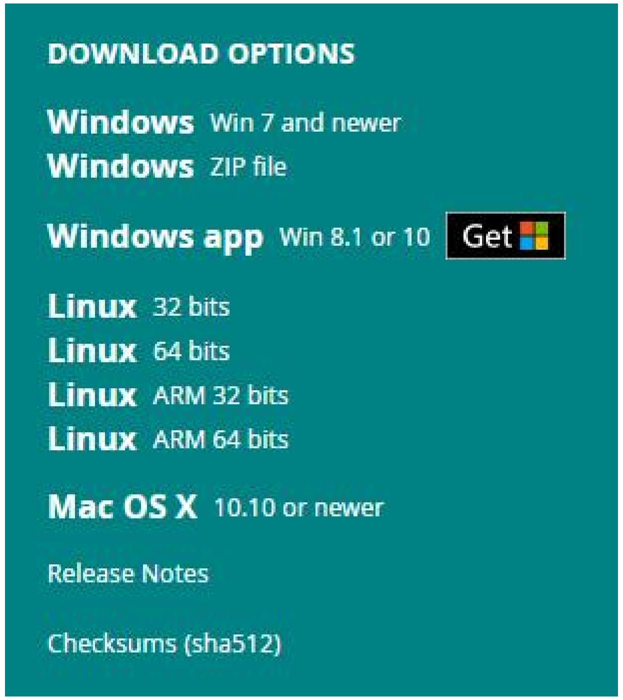
Active development of the Arduino software is **hosted by GitHub**. See the instructions for **building the code**. Latest release source code archives are available **here**. The archives are PGP-signed so they can be verified using **this** gpg key.

DOWNLOAD OPTIONS

Windows Win 7 and newer
Windows ZIP file
Windows app Win 8.1 or 10 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)

There may be a newer version on the site when you see this tutorial!

Step 2: Download the development software compatible with your computer system, here we take Windows as an example.


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

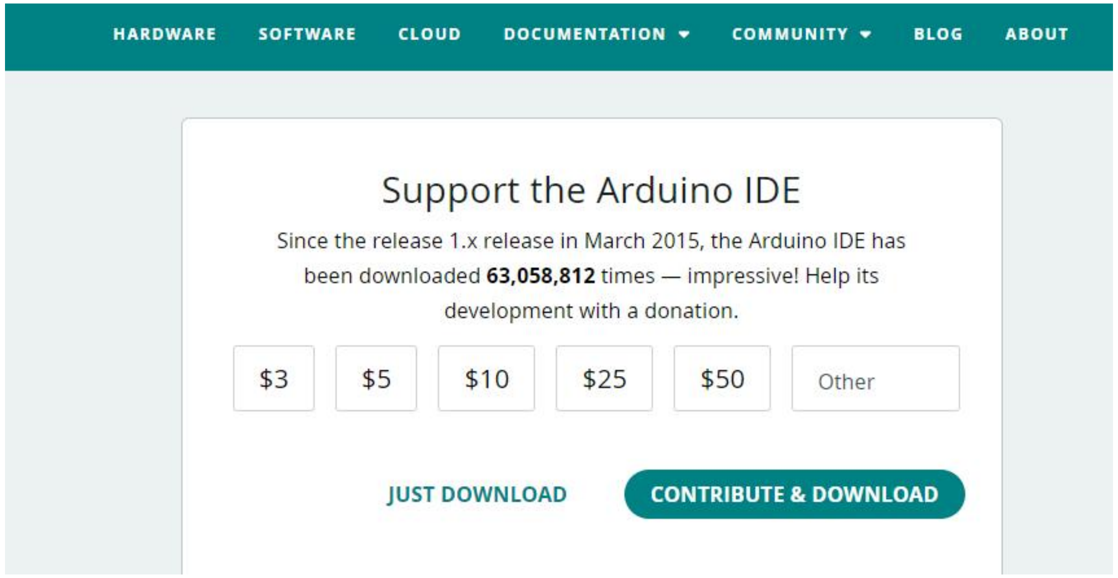
Mac OS X 10.10 or newer

Release Notes

Checksums (sha512)

You can choose between an installer (.exe) and a Zip package. We recommend that you use the first "Windows Win7 and newer" to directly install everything you need to use the Arduino software (IDE), including drivers. With the Zip package, you need to install the driver manually. Of course, Zip files are also useful if you want to create portable installations.

Click on "Windows Win7 and newer"



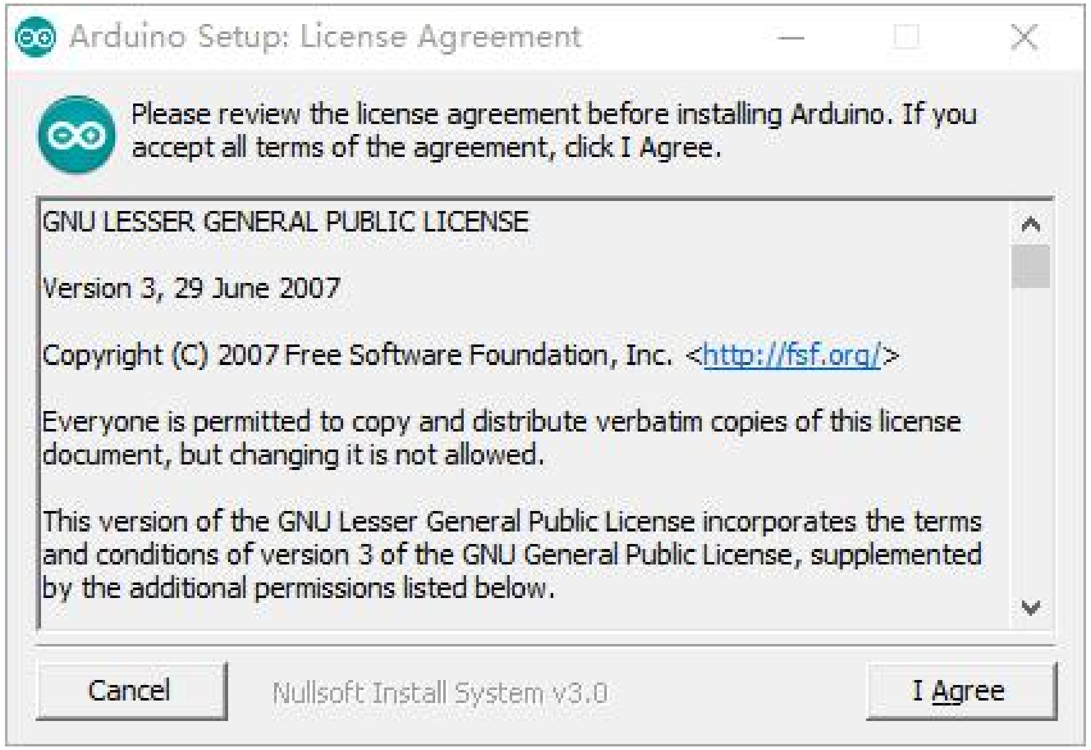
Support the Arduino IDE

Since the release 1.x release in March 2015, the Arduino IDE has been downloaded **63,058,812** times — impressive! Help its development with a donation.

Click on "JUST DOWNLOAD".

After the download is complete, the installation package file with the "exe" suffix will be obtained

Double click to run the installer



Arduino Setup: License Agreement

Please review the license agreement before installing Arduino. If you accept all terms of the agreement, click I Agree.

GNU LESSER GENERAL PUBLIC LICENSE

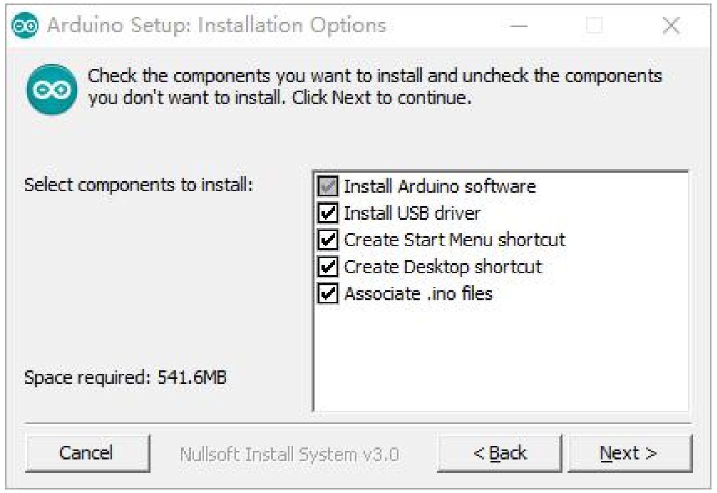
Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

This version of the GNU Lesser General Public License incorporates the terms and conditions of version 3 of the GNU General Public License, supplemented by the additional permissions listed below.

Click "I Agree" to see the following interface



Arduino Setup: Installation Options

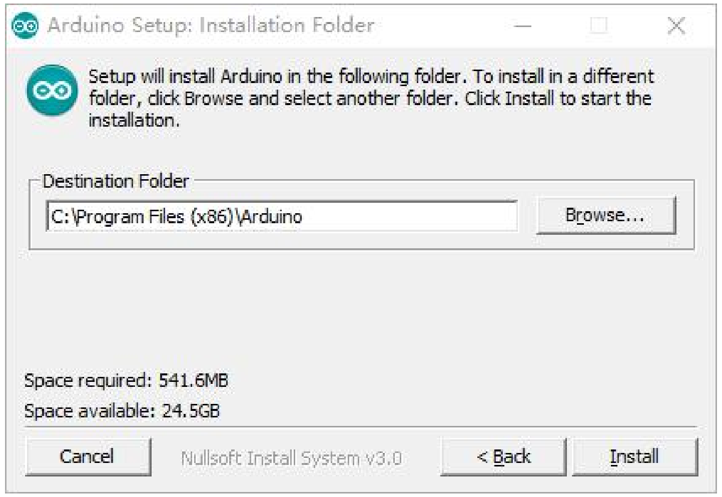
Check the components you want to install and uncheck the components you don't want to install. Click Next to continue.

Select components to install:

- ✓ Install Arduino software
- ✓ Install USB driver
- ✓ Create Start Menu shortcut
- ✓ Create Desktop shortcut
- ✓ Associate .ino files

Space required: 541.6MB

Click "Next"



Arduino Setup: Installation Folder

Setup will install Arduino in the following folder. To install in a different folder, click Browse and select another folder. Click Install to start the installation.

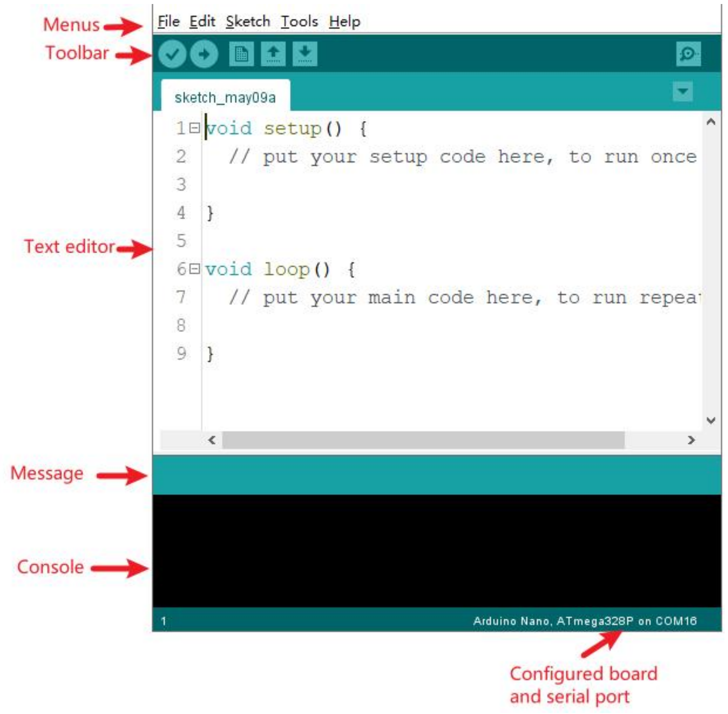
Destination Folder: C:\Program Files (x86)\Arduino

Space required: 541.6MB
Space available: 24.5GB

You can press "Browse..." to select the installation path or directly enter the directory you want. Then click "Install" to install. (For Windows users, the driver installation dialog may pop up during the installation process , when it pops up, please allow the installation)

After the installation is complete, an Arduino software shortcut will be generated on the desktop , double click to enter the Arduino software platform environment .

After the installation is complete, open the software to see the software platform interface as shown below:


Menus
Toolbar
Text editor
Message
Console
Configured board and serial port
File Edit Sketch Tools Help
sketch_may09a
Arduino Nano, ATmega328P on COM16

```
void setup() {
  // put your setup code here, to run once

}

void loop() {
  // put your main code here, to run repea

}
```

Programs written using the Arduino software (IDE) are called "Sketch". These "Sketch" are written in a text editor and saved with the file extension " .ino " .

The editor has functions for cutting , pasting, and searching and replacing text. The message area provides feedback and displays errors when saving and exporting. The console displays text output by the Arduino software (IDE), including full error messages and other information. The lower right corner of the window displays the configured boards and serial ports. Toolbar buttons allow you to verify and upload programs, create, open and save projects, and open the serial monitor. The positions of the corresponding functions in the toolbar buttons are as follows:

- Verify : Compile code to check for errors
- Upload : Compile code and upload to circuit board
- New : Create a project file
- Open : Select an item from an existing library and open it in a new window
- Save : Save your project files
- Serial Monitor : Open the serial monitor

(It is worth noting that the "ino" file must be saved in a folder with the same name as itself. If the program is not opened in a folder with the same name, it will be forced to automatically create a folder with the same name.

Install Arduino (Mac OS X)

Download and unzip the zip file, double-click Arduino.app to enter the Arduino IDE; if there is no Java runtime library in your computer, you will be asked to install it, after the installation is complete, you can run Arduino lDE.

Install Arduino (Linux)

You will have to use the make install command. If you are using Ubuntu system, it is recommended to install Arduino ID from Ubuntu Software Center

Lesson 2 Adding "Libraries" to the Arduino IDE

How to Install Additional Libraries in Arduino IDE 1

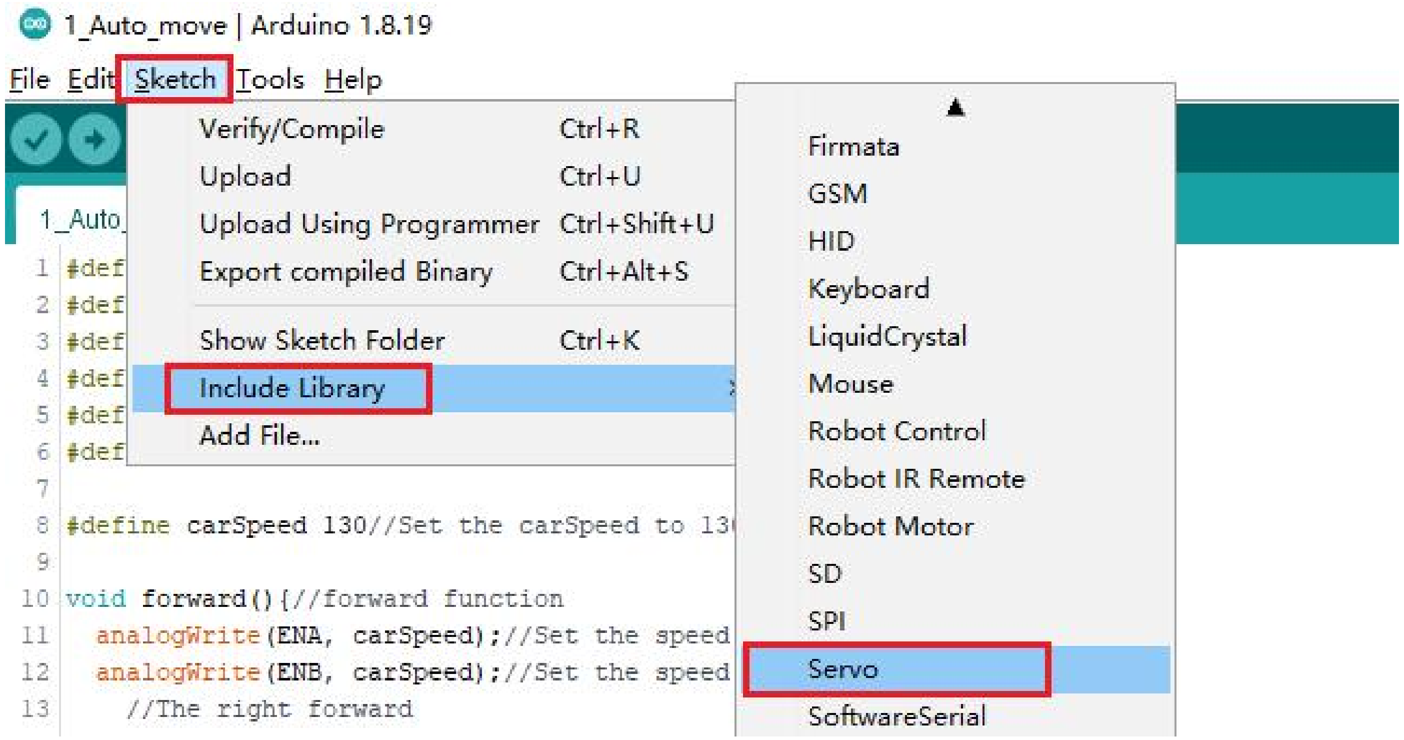
Once you are familiar with the Arduino software and use the built-in functions, you may wish to extend the functionality of the Arduino with other libraries.

What are Libraries?

A library is a set of code that allows you to easily connect to sensors, displays, modules, and more. For example, the LiquidCrystal library allows you to easily talk to character LCD displays.

There are thousands of libraries available for download directly through the Arduino IDE, all of which you can find in the Arduino Library Reference .

The library used in this tutorial is the servo motor: servo. The newer version of the Arduino IDE has integrated this library. You can open the Arduino IDE and find it in Sketch>Include Library. If the library is not available, follow the steps below to install the library That's it.

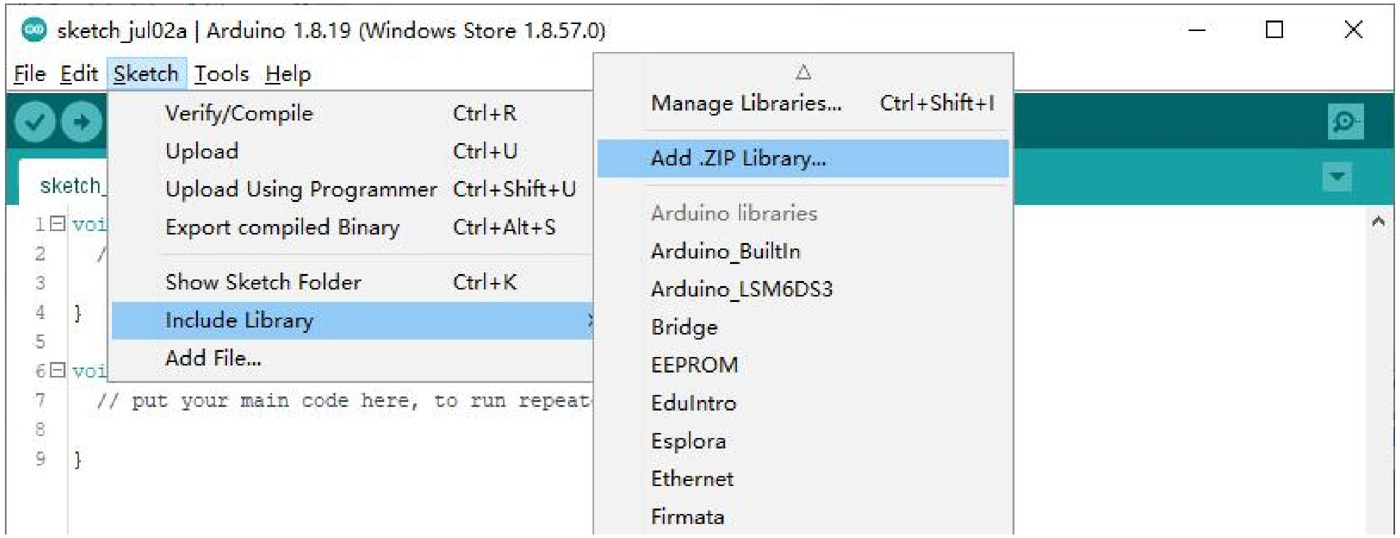


```
#def
#def
#def
#def
#def
#def

#define carSpeed 130//Set the carSpeed to 13

void forward(){//forward function
  analogWrite(ENA, carSpeed);//Set the speed
  analogWrite(ENB, carSpeed);//Set the speed
   //The right forward
```

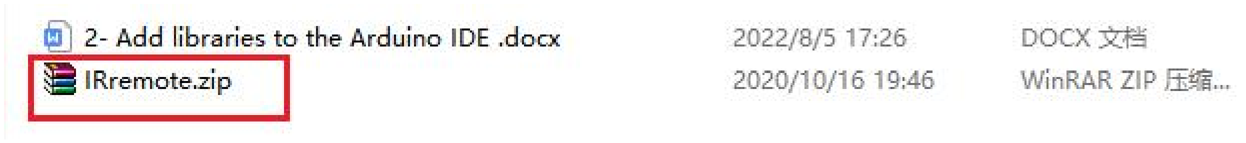
Method 1: Import the .zip library

Libraries are usually distributed as ZIP files or folders. The name of the folder is the name of the library. This folder will contain a .cpp file, a .h file, and usually a keywords.txt file, examples folders, and other files needed by the library. Starting with version 1.0.5, you can install 3rd party libraries in the IDE. Do not unzip the downloaded library, keep it as is.

In the Arduino IDE, navigate to Sketch > Include Library > Add .ZIP Library , and at the top of the drop-down list, select the "Add .ZIP Library" option.

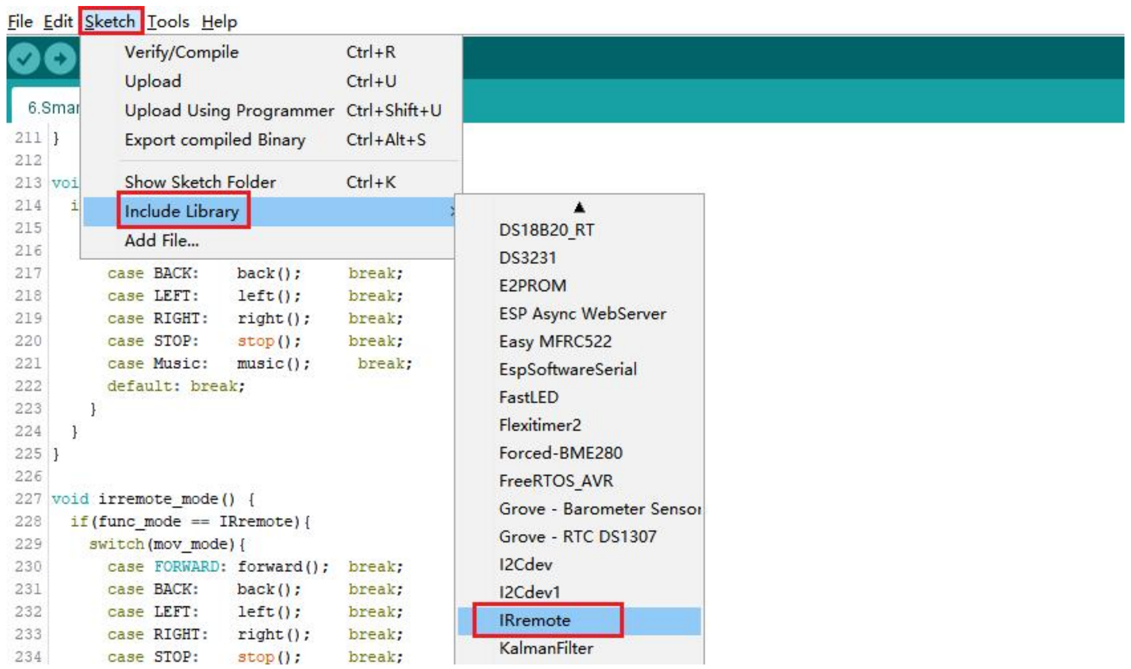

sketch_jul02a | Arduino 1.8.19 (Windows Store 1.8.57.0)
File Edit Sketch Tools Help
Verify/Compile Ctrl+R
Upload Ctrl+U
Upload Using Programmer Ctrl+Shift+U
Export compiled Binary Ctrl+Alt+S
Show Sketch Folder Ctrl+K
Include Library
Add File...
Manage Libraries... Ctrl+Shift+I
Add .ZIP Library...
Arduino libraries
Arduino_BuiltIn
Arduino_LSM6DS3
Bridge
EEPROM
EduIntro
Esplora
Ethernet
Firmata

You will be prompted to select the library you want to add , navigate to the location of the saved IRremote.zip file on your computer (4. Tutorial_Arduino\2_Libraries) and open it as shown below . (Here is an example of adding the IRremote library)



```
2- Add libraries to the Arduino IDE .docx    2022/8/5 17:26      DOCX 文档
IRremote.zip                                  2020/10/16 19:46    WinRAR ZIP 压缩...
```

Return to the Sketch > Include Library menu. You should now see Libraries at the bottom of the drop-down menu. It's ready to use in your sketches .



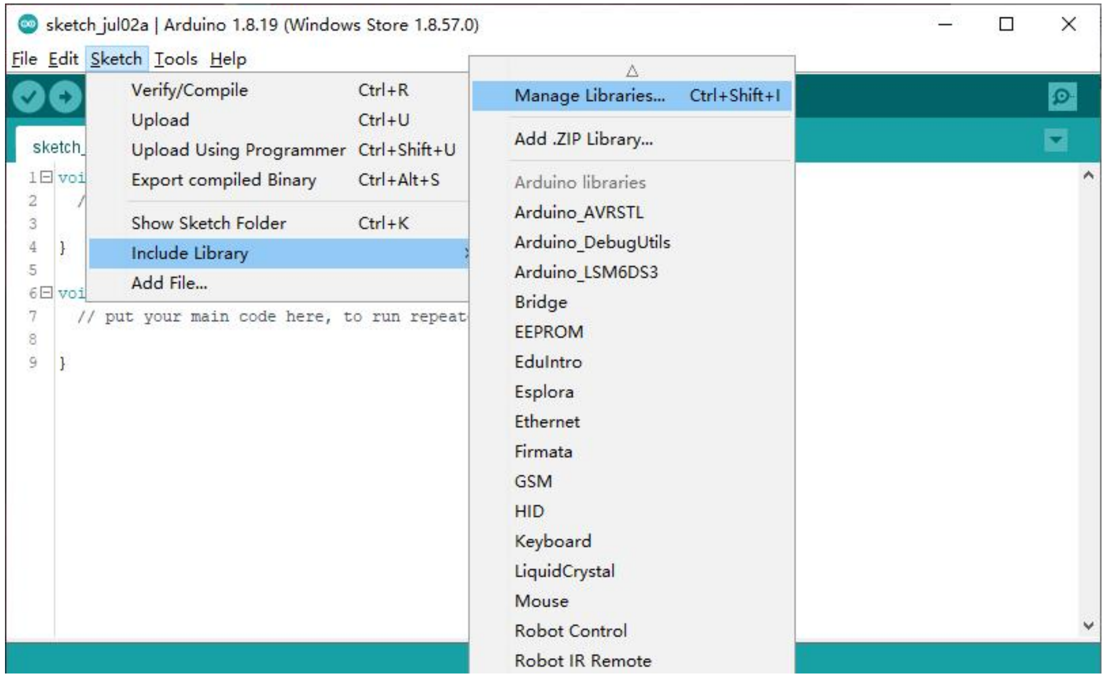
```
}

voi
  i


      case BACK:    back();     break;
      case LEFT:    left();     break;
      case RIGHT:   right();    break;
      case STOP:    stop();     break;
      case Music:   music();     break;
      default: break;
    }
  }
}

void irremote_mode() {
  if(func_mode == IRremote){
    switch(mov_mode){
      case FORWARD: forward();  break;
      case BACK:    back();     break;
      case LEFT:    left();     break;
      case RIGHT:   right();    break;
      case STOP:    stop();     break;
```

Note: This library will be available for sketches, but with older IDE versions, the library's examples will not be exposed in File > Examples until the IDE is restarted.

Please add the "MsTimer2" and "servo" libraries in the same way .

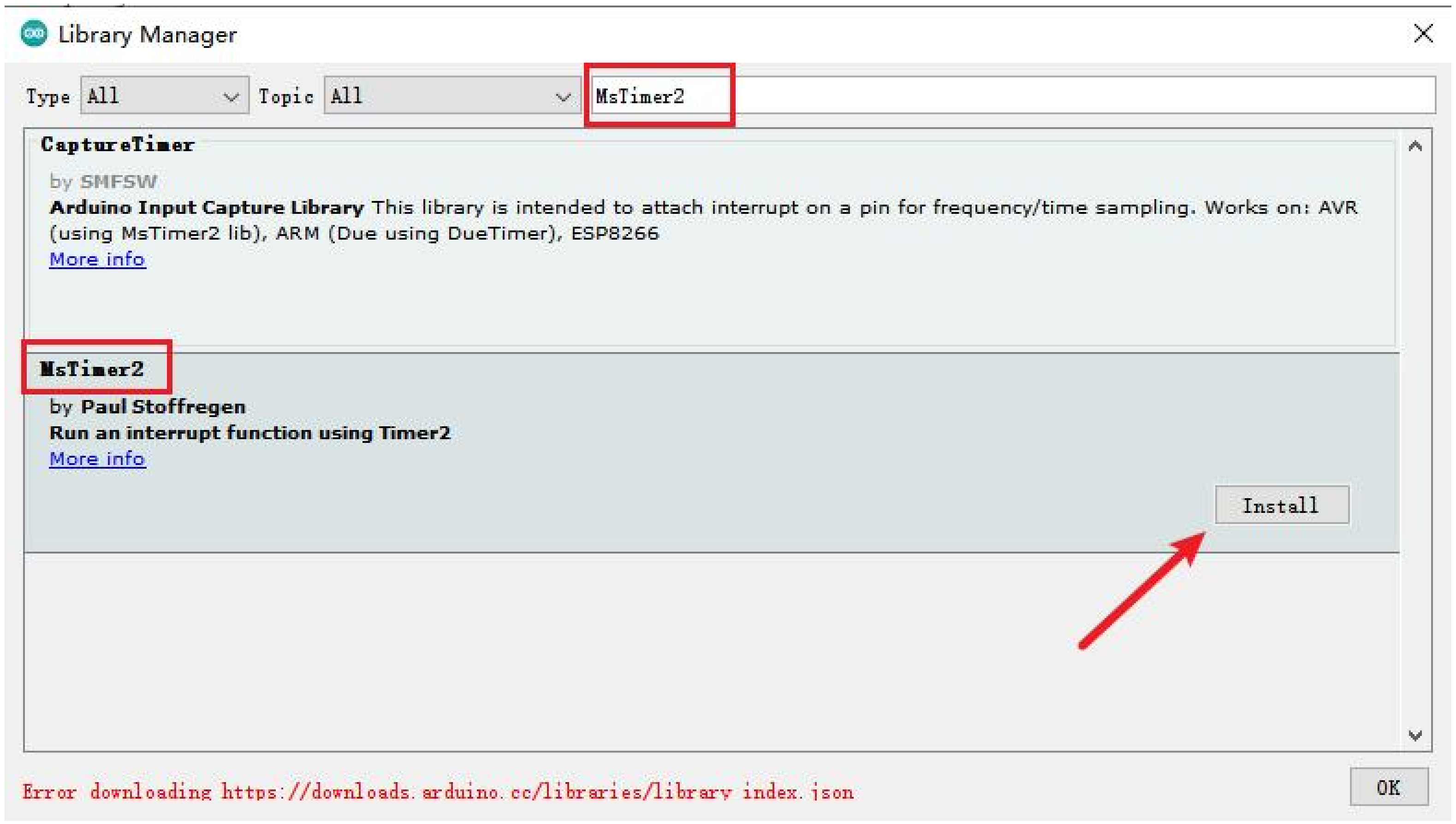
Method 2: (This method requires networking) In addition to adding the library that has been prepared, you can also use the library manager to search and download the desired library

To install new libraries into your Arduino IDE, you can use the library manager (available from IDE 1.6.2 and above). Open the IDE and click the Sketch menu, then Include Library > Manage Libraries.



```
// put your main code here, to run repeat
}
```

The library manager will open and you will see a list of libraries that are installed or ready to be installed. Here, we take the installation of the MsTimer2 library as an example, and the same is true for installing other libraries. Scroll the list to find it, then select the version of the library to install, sometimes only one version of the library is available , click Install to install it .


Library Manager

Type All Topic All MsTimer2

CaptureTimer
by SMFSW
Arduino Input Capture Library This library is intended to attach interrupt on a pin for frequency/time sampling. Works on: AVR (using MsTimer2 lib), ARM (Due using DueTimer), ESP8266
More info

MsTimer2
by **Paul Stoffregen**
Run an interrupt function using Timer2
More info

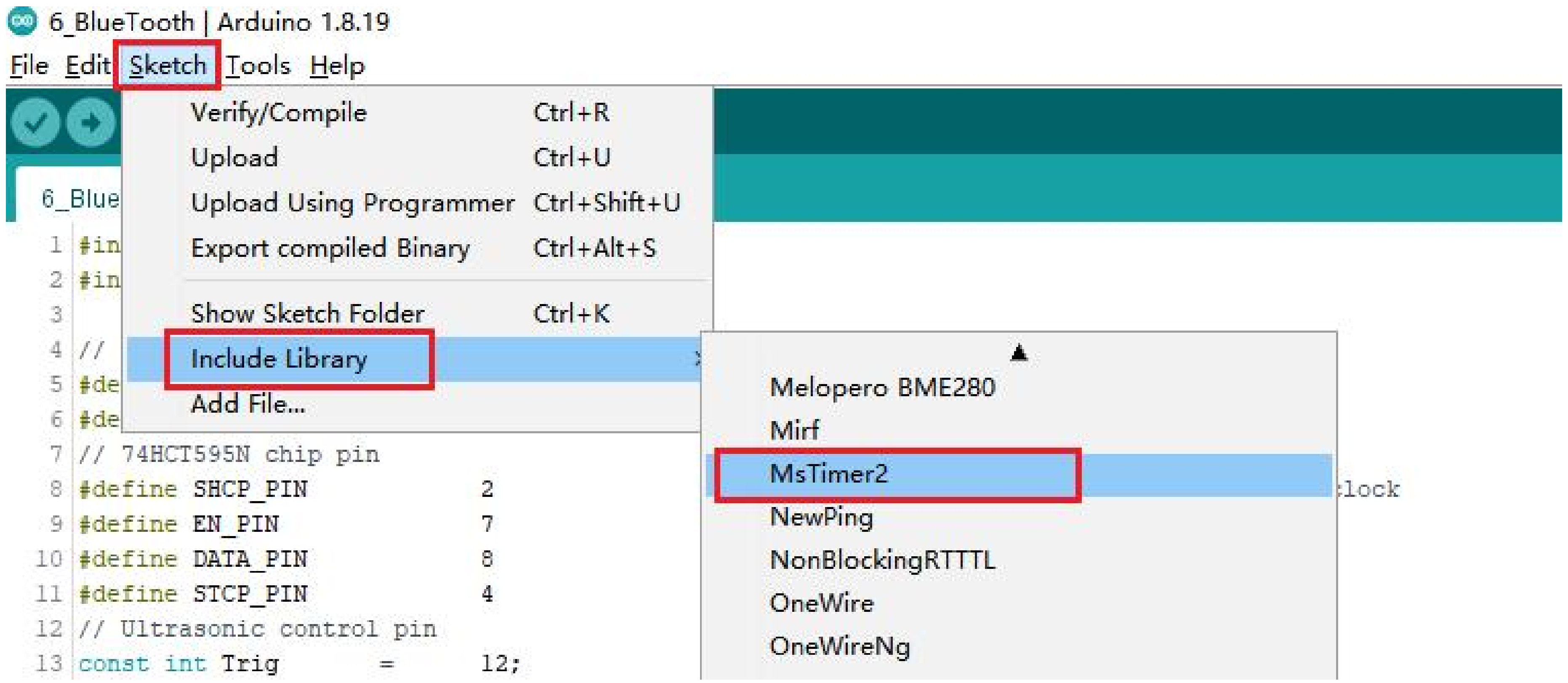
Install

Error downloading https://downloads.arduino.cc/libraries/library index.json

OK

The download may take some time, depending on your connection speed , and you can close the library manager when finished.

You can now find new libraries available in the Sketch > Include Library menu.



```
#in
#in

//
#de
#de
// 74HCT595N chip pin
#define SHCP_PIN           2                              :lock
#define EN_PIN             7
#define DATA_PIN           8
#define STCP_PIN           4
// Ultrasonic control pin
const int Trig      =      12;
```

Lesson 3 The first program code - Blink

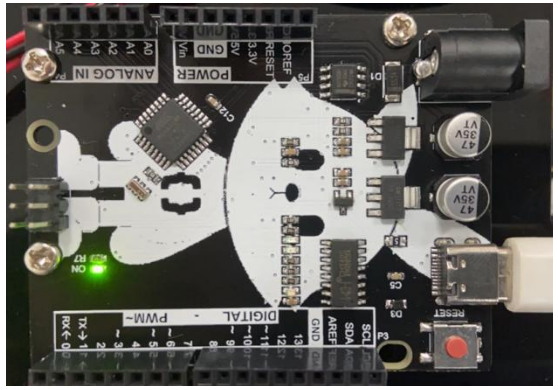
About this lesson:

In this lesson, you'll learn how to program your control board to blink the built-in LEDs, as well as learn the basic steps to download a program. Here is an example of the LUCKY SIX motherboard.

Main control board :

There are rows of connectors on both sides of the motherboard for connecting multiple electronic devices and plug-in "modules" that extend their functionality.

It also has an LED that you can control from the sketch , which is built into the motherboard .

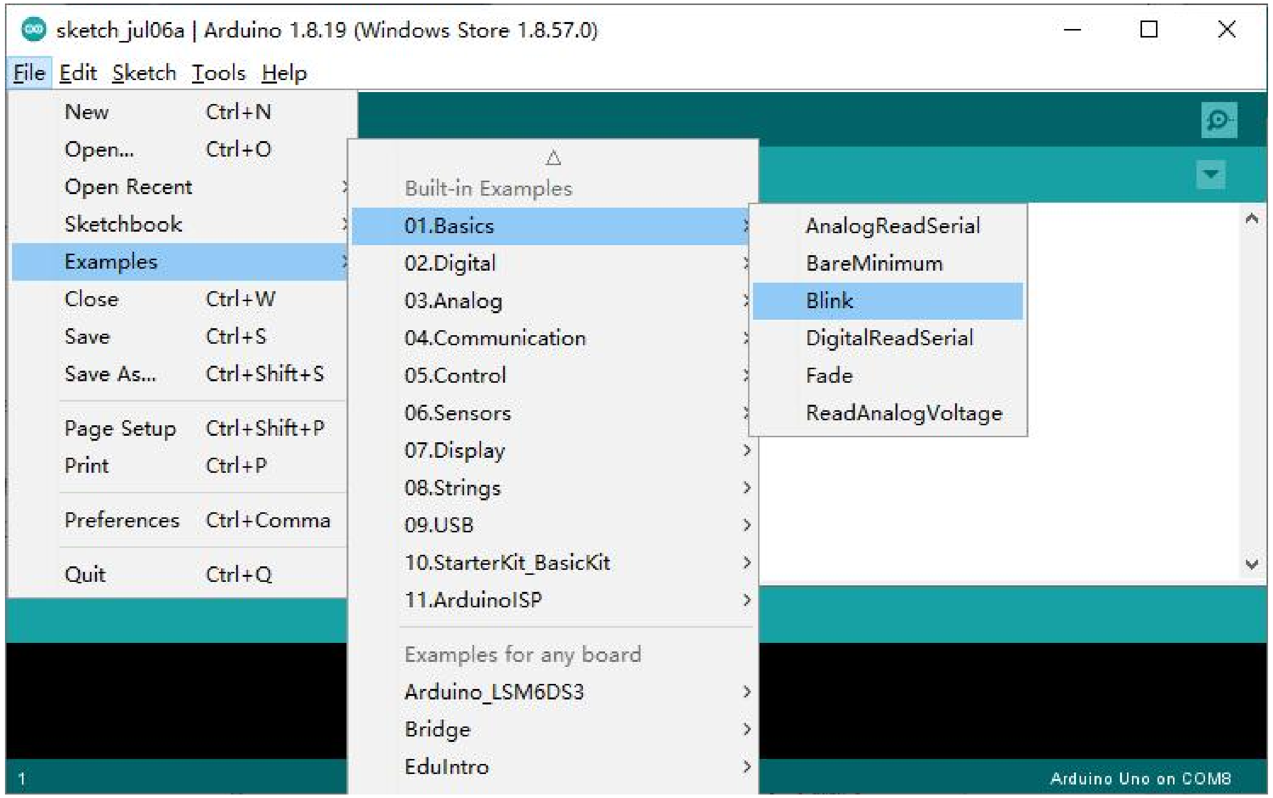


When you connect the board to the USB plug, you may find that its LEDs are already blinking because the "Blink" sketch is pre-installed on the board.

Make your own "Blink" sketches

the board with our own Blink sketch , and then change its blink rate. Connect the board to the computer, set up the Arduino IDE and make sure you can find the correct serial port , and upload the program for testing.

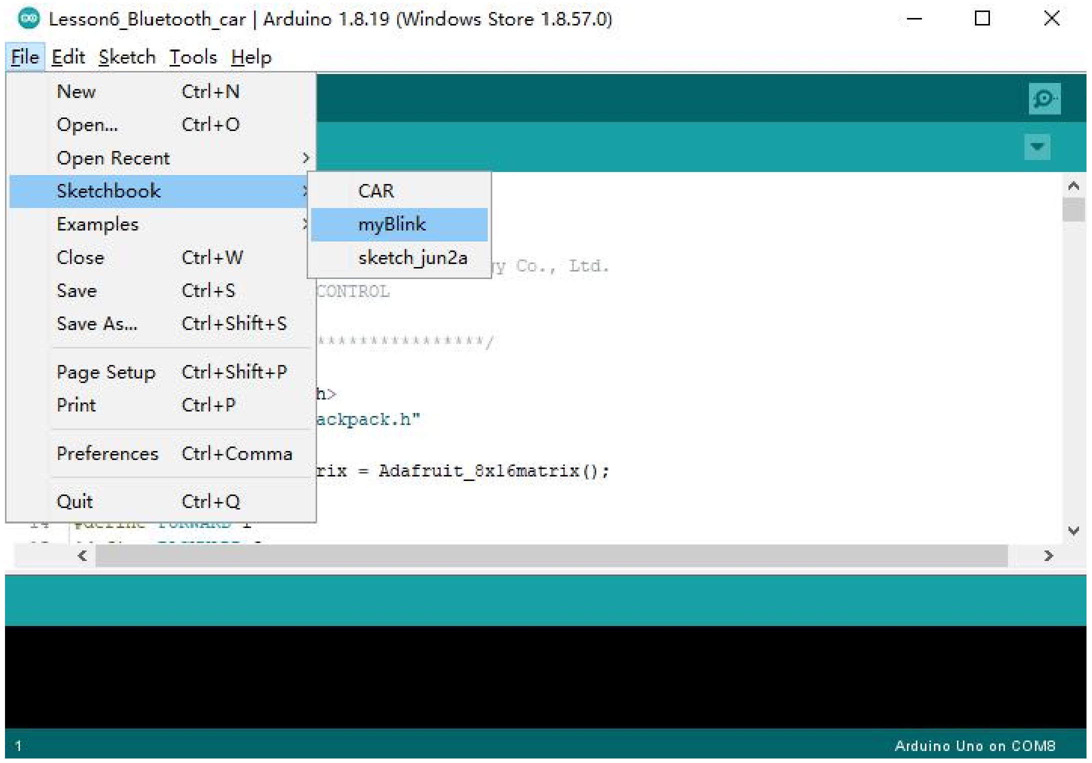
The Arduino IDE includes a number of example sketches that you can load and use , including a "Blink" example sketch for making an "L" LED. In the IDE menu system File > Examples > 01. The " Blink " sketch you will find in Basics .



The example sketches included with the Arduino IDE are "read-only". That is, you can upload them to the UNOR3 board, but if you change them, you cannot save them as the same file. Since we're changing this sketch, the first thing you need to do is save a copy of yourself.

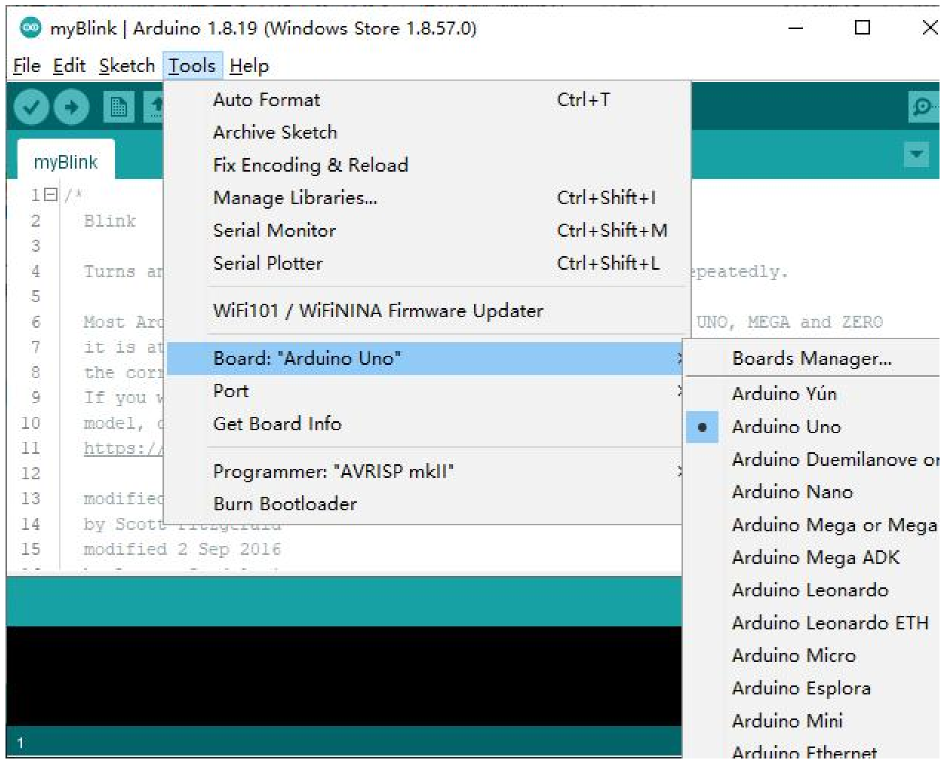
From the Arduino IDE's File menu, select "Save As.." and save the sketch as "MyBlink".

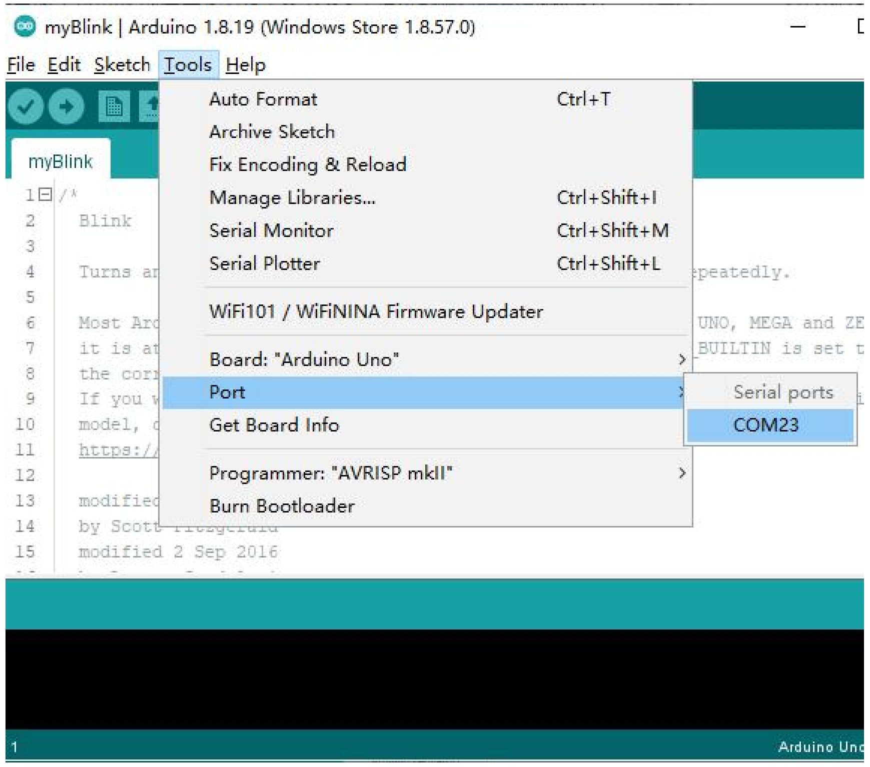
You've saved a copy of "Blink" in Sketchbook, which means that if you want to find it again, just open it with the " File > Sketchbook " menu option.



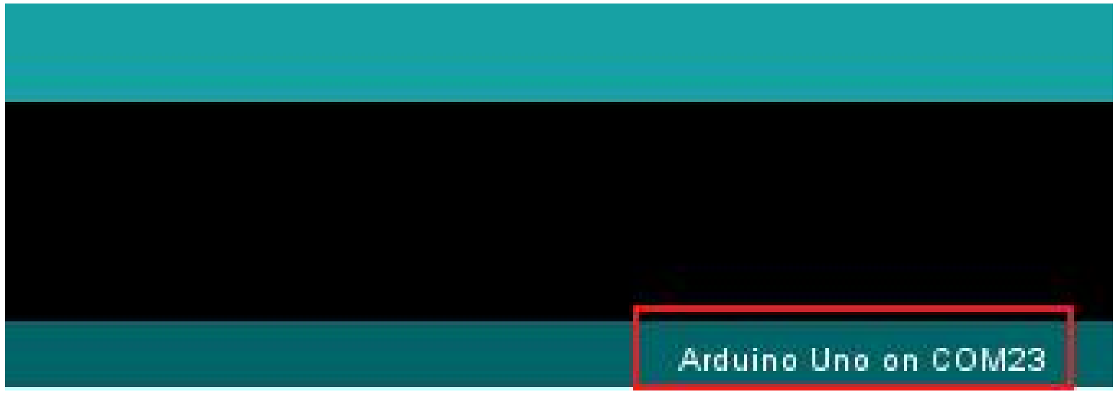
```
y Co., Ltd.
CONTROL
****************/
h>
ackpack.h"
rix = Adafruit_8x16matrix();
```

the Arduino board to the computer using a USB cable and check that the Board Type and Serial Port are set correctly.

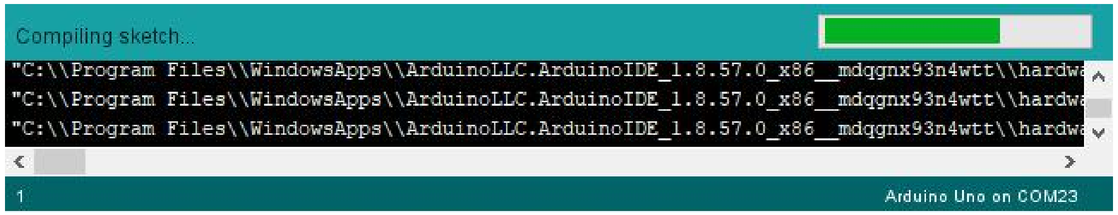



myBlink | Arduino 1.8.19 (Windows Store 1.8.57.0)
File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Fix Encoding & Reload
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter Ctrl+Shift+L
WiFi101 / WiFiNINA Firmware Updater
Board: "Arduino Uno"
Port
Get Board Info
Programmer: "AVRISP mkII"
Burn Bootloader
Serial ports
COM23

Note: Select board type as Arduino UNO and select serial port number 23. The serial port appears to be different for everyone, although COM23 is chosen here , it could be COM3 or COM4 on your computer. A correct COM port should be COMx compliant (arduino X XX) standard.



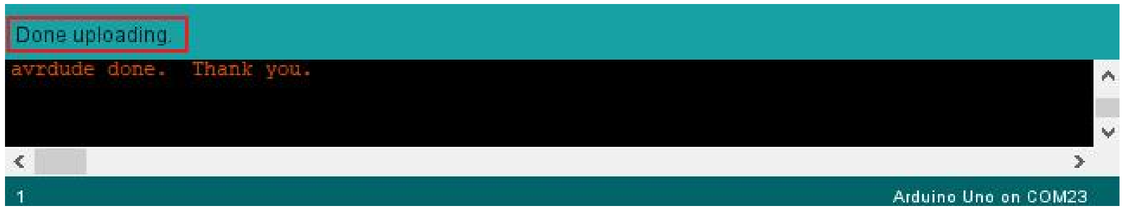
Because Bluetooth takes up the RX/TX port, pull out the Bluetooth before uploading the code and install it after uploading! After clicking the "Upload" button, the program starts uploading, and at this time, the LED on the Arduino will start blinking as the sketch is transferred.



Compiling sketch...

```
"C:\\Program Files\\WindowsApps\\ArduinoLLC.ArduinoIDE_1.8.57.0_x86__mdqgnx93n4wtt\\hardwa
"C:\\Program Files\\WindowsApps\\ArduinoLLC.ArduinoIDE_1.8.57.0_x86__mdqgnx93n4wtt\\hardwa
"C:\\Program Files\\WindowsApps\\ArduinoLLC.ArduinoIDE_1.8.57.0_x86__mdqgnx93n4wtt\\hardwa
```

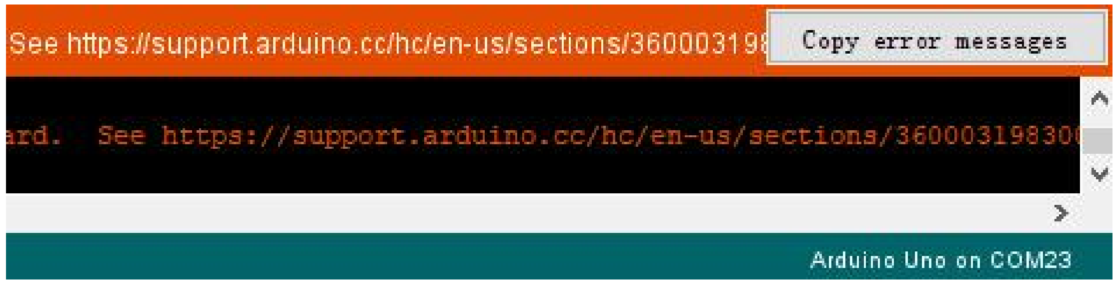
The transfer is complete, "Transfer complete" appears



Done uploading.

```
avrdude done.  Thank you.
```

During the "Compile Sketch.." process , you may receive the following error message:



See https://support.arduino.cc/hc/en-us/sections/36000319

```
ard.   See https://support.arduino.cc/hc/en-us/sections/360003198300
```

This could mean that your board isn't connected at all, or the driver isn't installed (if needed) or the wrong serial port is selected wrong . If this happens to you, please check your IDE settings and motherboard connections . After the upload is complete , the board LEDs should reboot and start blinking.

Note that a large portion of this sketch is made up of annotations. These are not actual program instructions; instead, they just explain how to make the program work. They are there for your ease of reading . Everything between " /* " and " */ " at the top of the sketch is a block comment that explains what the sketch is for.

A single-line comment starts with "//", and everything up to the end of the line is considered a comment.

The first part of the code is:

```
// the setup function runs once when you press reset or power the board
void setup () {
// initialize digital pin LED_BUILTIN as an output.
  pinMode (LED_BUILTIN, OUTPUT);
}
```

Every sketch requires a "setup" function , aka "Void" setup()" function, which is executed when the reset button is pressed. It is executed whenever the board is reset for any reason, such as first powering up or after uploading a sketch .

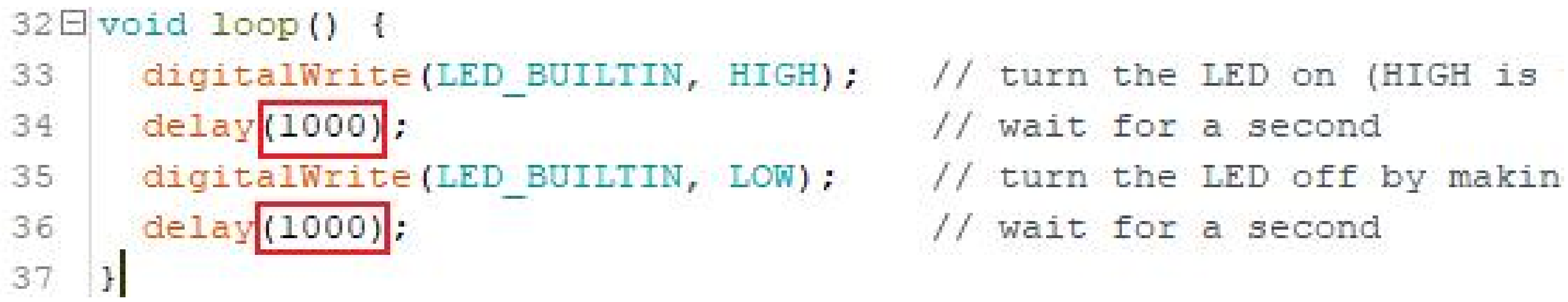
The next step is to name the pin and set the output, here " LED_BUILTIN " is set as the output port. On most Arduinos, including UNO , pin 13 is the pin corresponding to the LED, and for the convenience of programming, the program has set the LED_BUILTIN variable to this pin, so there is no need to rename it to pin 13 for direct use.

The sketch must also have a " loop " function. Unlike the "setup" function, which only runs once, after a reset, the "loop" function will restart as soon as it finishes running the command.

```
// the loop function runs over and over again forever
void loop () {
  digitalWrite (LED_BUILTIN, HIGH); // turn the LED on (HIGH is the voltage level)
  delay ( 1000 ); // wait for a second
  digitalWrite (LED_BUILTIN, LOW); // turn the LED off by making the voltage LOW
  delay ( 1000 ); // wait for a second
}
```

Inside the loop function, the command first turns on the LED pin (high), then "delays for 1000 ms (1 second), then turns off the LED pin and pauses for one second.

You are now going to make your LED blink faster. The key, as you might have guessed, is to change the parameters in " delay ()".



```
void loop() {
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is
  delay(1000);                       // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by makin
  delay(1000);                       // wait for a second
}
```

This delay is in milliseconds, so if you want the "LED" to blink twice as fast, change the value from "1000" to "500" . This will pause for half a second at each delay instead of a second. Upload the sketch again and you should see the "LED" start blinking faster .

So far, you have understood and mastered the basic Arduino programming knowledge and the basic steps of downloading programs, which lays a good foundation for learning complex projects later.

Lesson 4 Servo Motor Drive

Course Introduction

This class mainly understands the properties and characteristics of the steering gear and learns the relevant knowledge of the steering gear, and masters the debugging method and circuit connection of the steering gear, and finally realizes the working method of the steering gear in the Arduino programming.

Introduction of steering gear

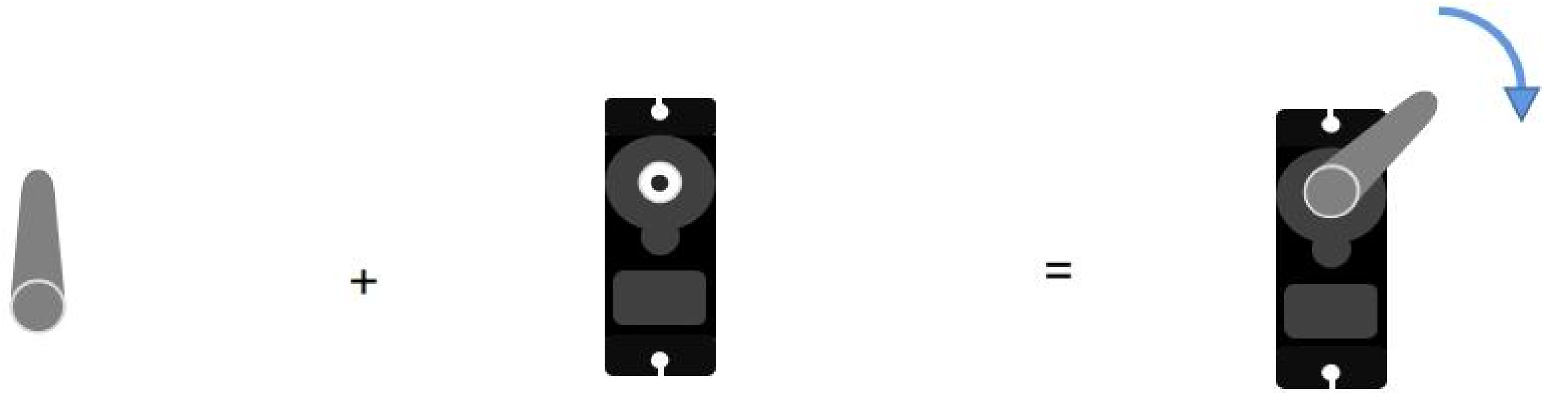
MG90S servo motor control pulse signal cycle is 20MS pulse width modulation signal (PWM), the pulse width is from 0.5ms to 2.5ms , and the corresponding steering position is from 0 to 180 degrees, which changes linearly.

That is to say, if a certain pulse width is provided to the servo, 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 supplied to it.

There is a reference circuit inside the steering gear, which generates a pulse signal with a period of 20ms and a width of 1.5ms . There is a comparator, which compares the external signal with the reference signal, determines the direction and size, and generates the motor rotation signal.

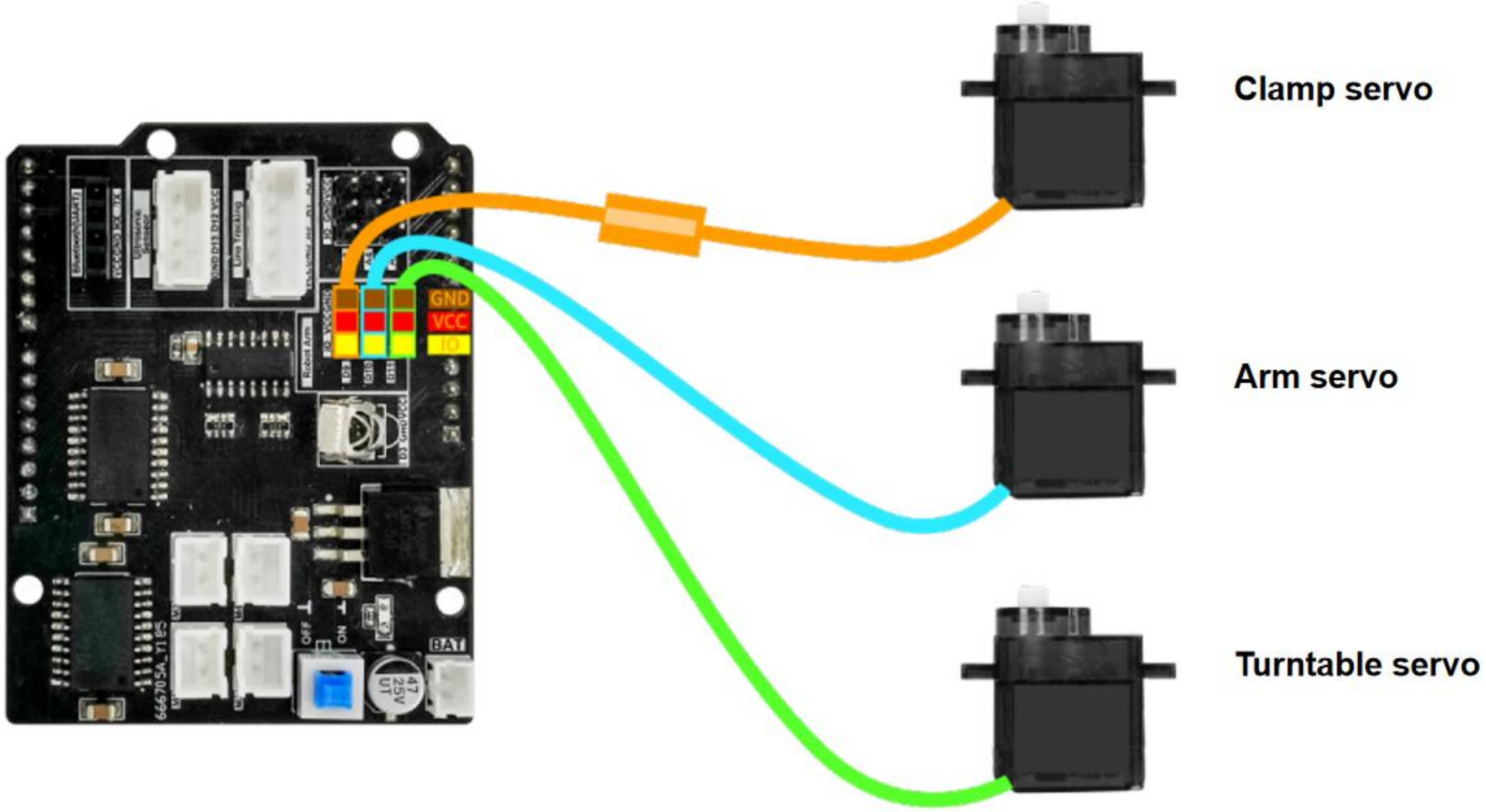
Recognize the initial 0° of the servo motor

Install the servo manipulator on the servo, and slowly turn the manipulator clockwise to the limit position, that is, the servo motor is currently at the initial 0° , and turning the manipulator counterclockwise can reach a maximum of 180° .



connection line

Servo motor wiring diagram:



Code Analysis:

Open the code file (path: 4. Tutorial_Arduino\4_Arduino Code\1.1_Servo_Angle\1.1_Servo_Angle.ino)

Import the servo library file

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

Declare the three servo motor signal ports as 9/10/11

```
#define CLAW_PIN 9
#define ARM_PIN 10
#define BASE_PIN 11
```

Define the rotation angle variable

```
int pos = 0 ;
```

Initialize setting of servo motor angle

```
void setup ()
{
    clawservo.attach ( CLAW_PIN ) ;
    armservo.attach ( ARM_PIN ) ;
    baseservo.attach ( BASE_PIN ) ;
    clawservo.write ( 135 ) ; _
    armservo.write ( 135 ) ; _
    baseservo.write ( 90 ) ; _
}
```

Loop execution, the clip servo motor reciprocates from 150° to 45°

```
    for (pos = 150 ; pos >= 45 ; pos -= 1 )
    {
        clawservo.write(pos);
        delay(15);
    }
    for (pos = 45; pos <= 150; pos += 1)
    {
        clawservo.write(pos);
        delay(15);
    }
```

Robot arm servo motor reciprocates from 150° to 0°

```
for (pos = 150 ; pos >= 0 ; pos -= 1 )
{
        armservo.write ( pos ) ;
        delay ( 15 );
}
    for (pos = 0 ; pos <= 150 ; pos += 1 )
{
        armservo.write ( pos ) ;
        delay ( 15 );
}
```

The turntable servo motor reciprocates from 180° to 0°

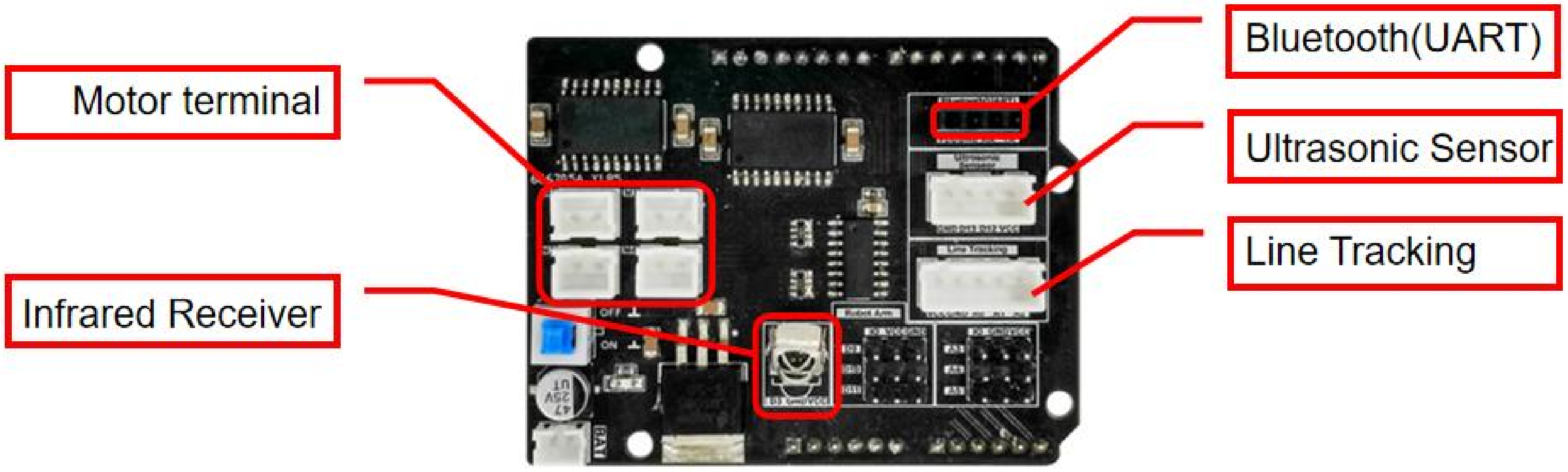
```
for (pos = 180 ; pos >= 0 ; pos -= 1 )
{
        baseservo.write ( pos ) ;
        delay ( 15 );
}
    for (pos = 0 ; pos <= 180 ; pos += 1 )
{
        baseservo.write ( pos ) ;
        delay ( 15 );
}
```

Lesson 5 Expansion Board and Motor Driver

Course Introduction

This class mainly learns about high-performance expansion boards and motor drive related knowledge

Expansion board introduction



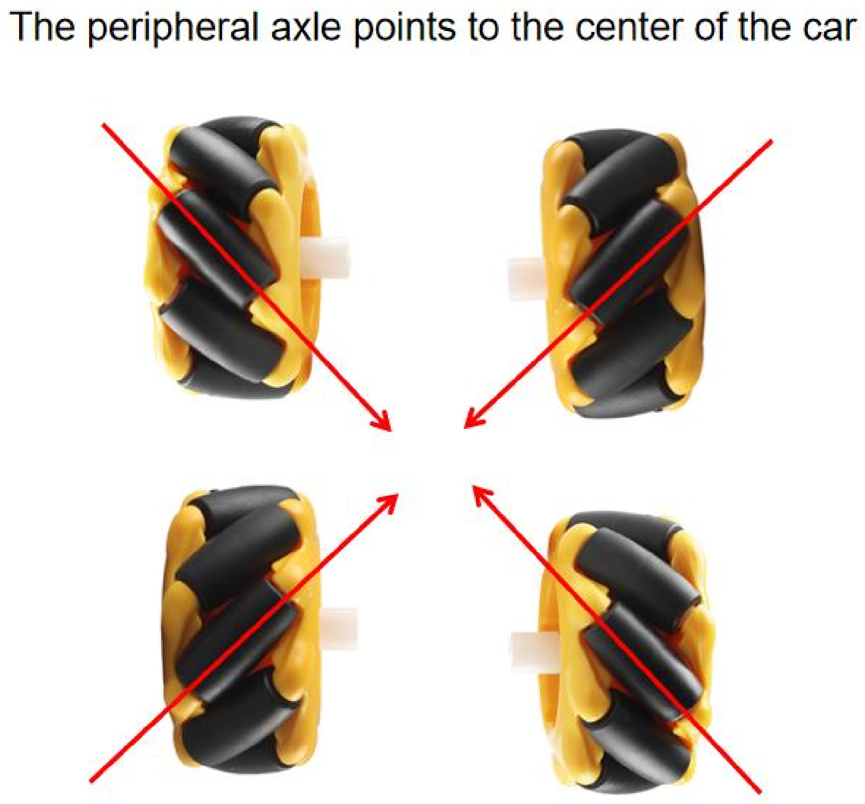
The expansion board expands the pins of the main board well, and integrates the infrared receiver stably on the board. The motor, ultrasonic and tracking modules are all connected by terminal plugging, which increases firmness and reliability. The BAT port for switch control is reserved, and the commonly used digital signal and analog signal ports have also been marked on the board, which can be used for DIY.

motor driven

Mecanum Wheel:

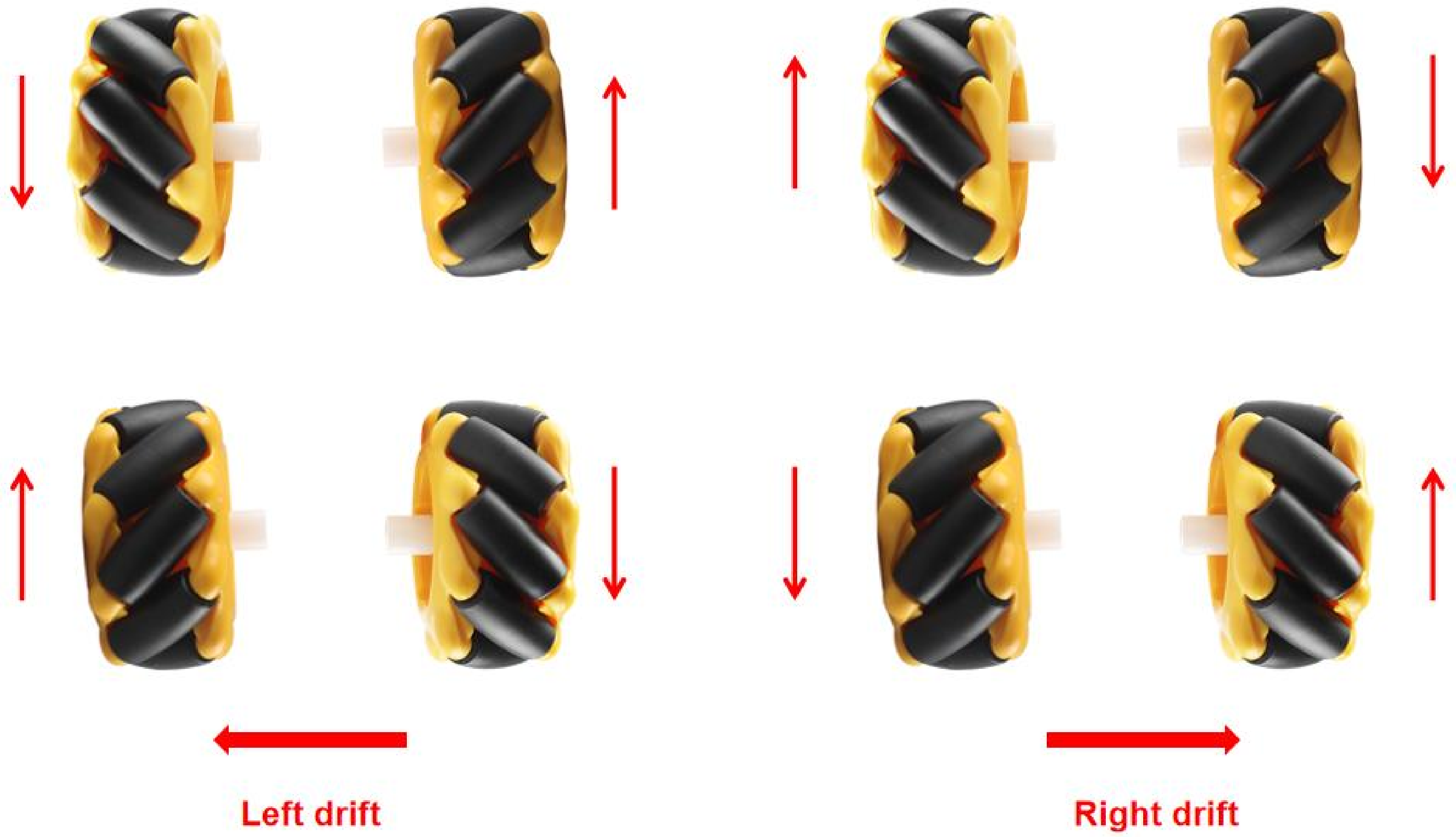
Mecanum wheel is a kind of wheel with a peripheral axle, generally divided into two types, one peripheral axle is inclined to the left, and the other is inclined to the right. These angled peripheral axles convert a portion of the wheel steering force into a wheel normal force, thus enabling the car to move in translation from side to side.

Assembly points (top view):

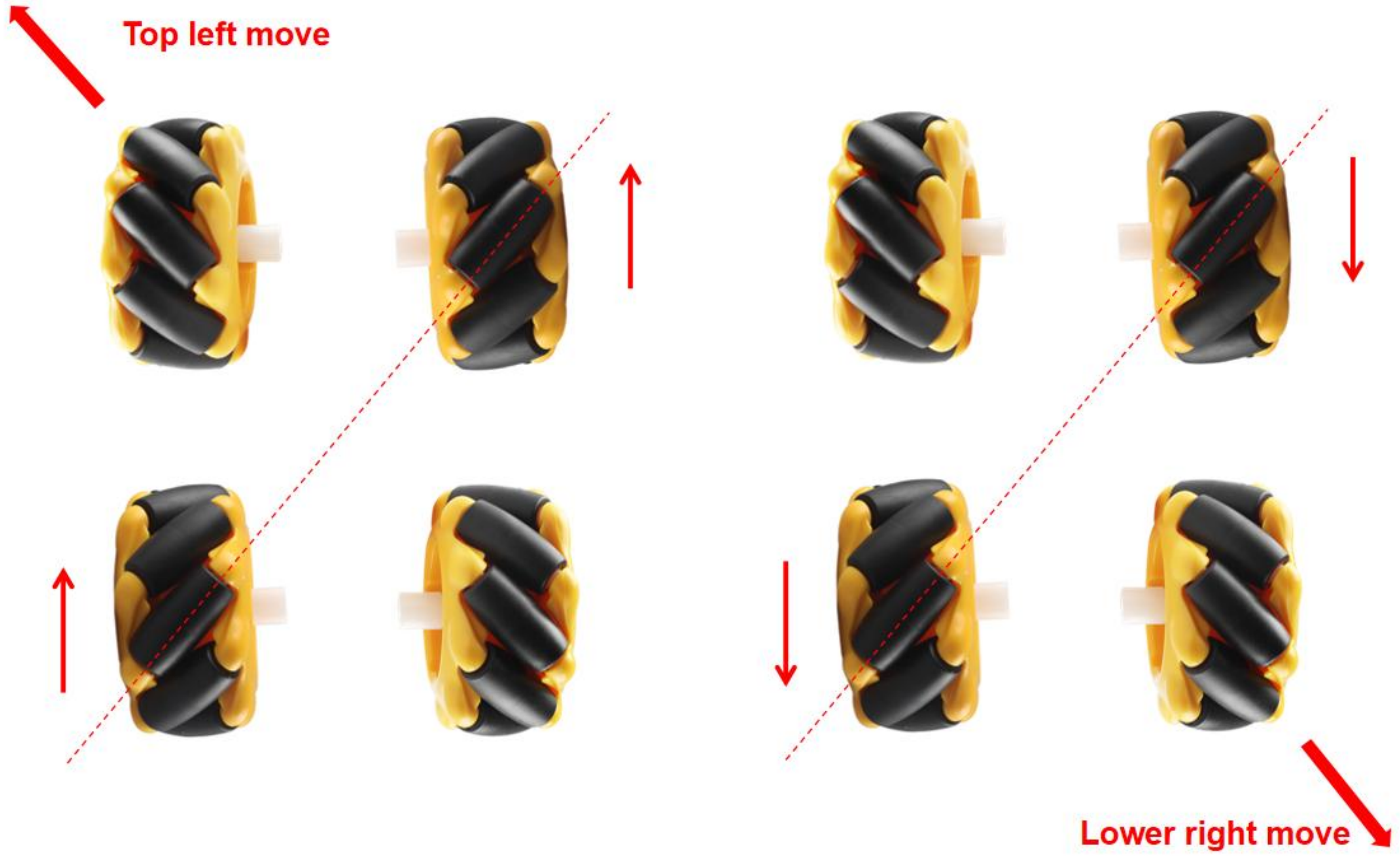


Movement principle:

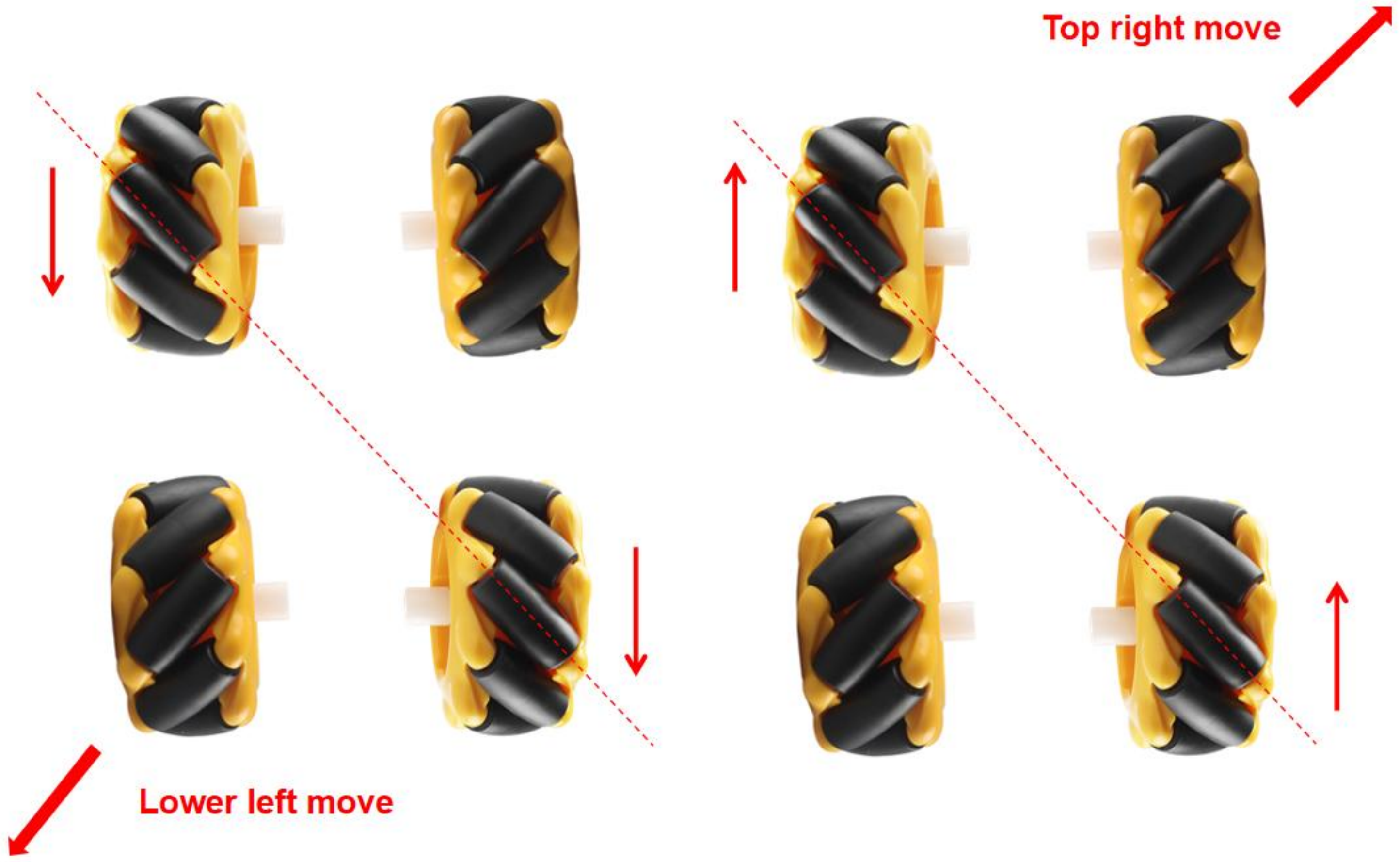
The different rotation directions of the wheel correspond to the left and right translation movement:



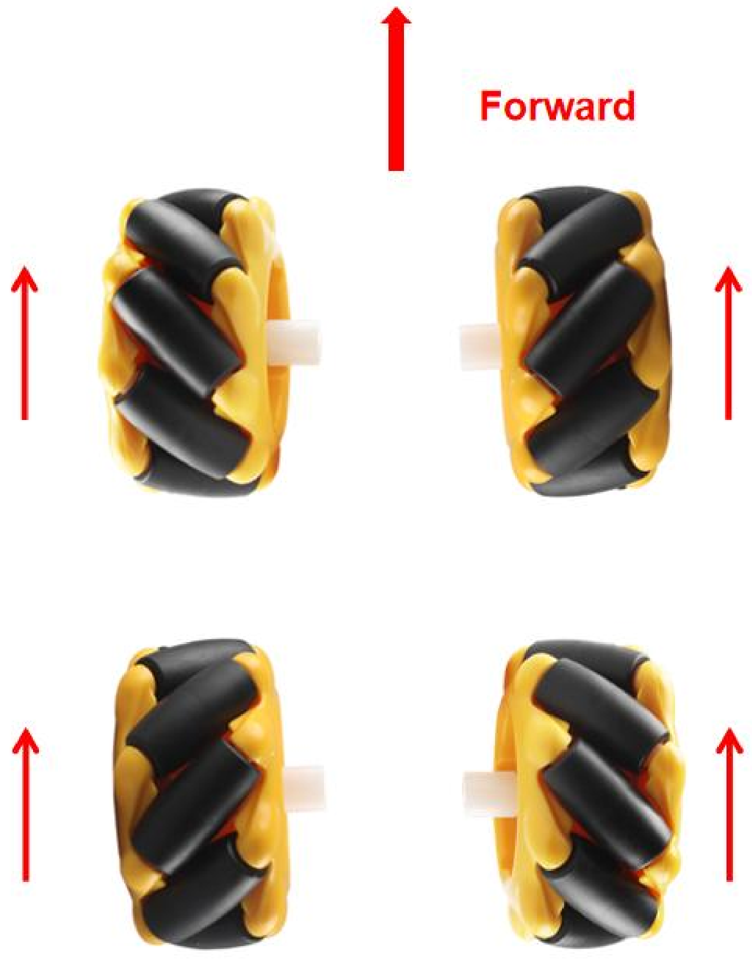
The different rotation directions of the wheels correspond to the movement in the upper left direction and the movement in the lower right direction:

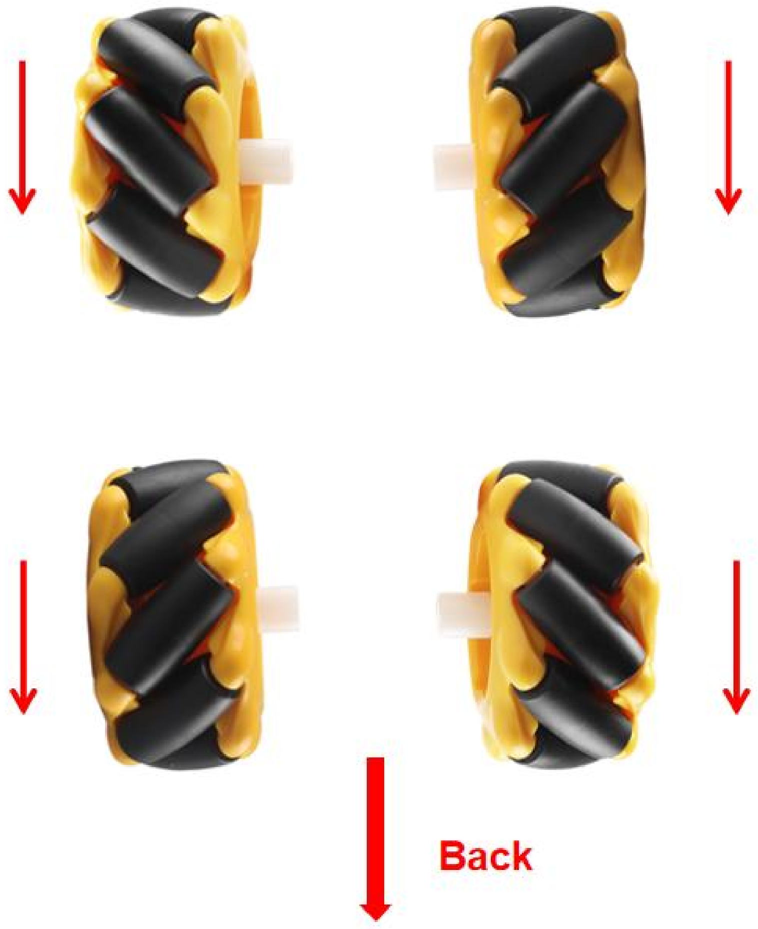


The different rotation directions of the wheels correspond to the movement in the lower left direction and the upper right direction:

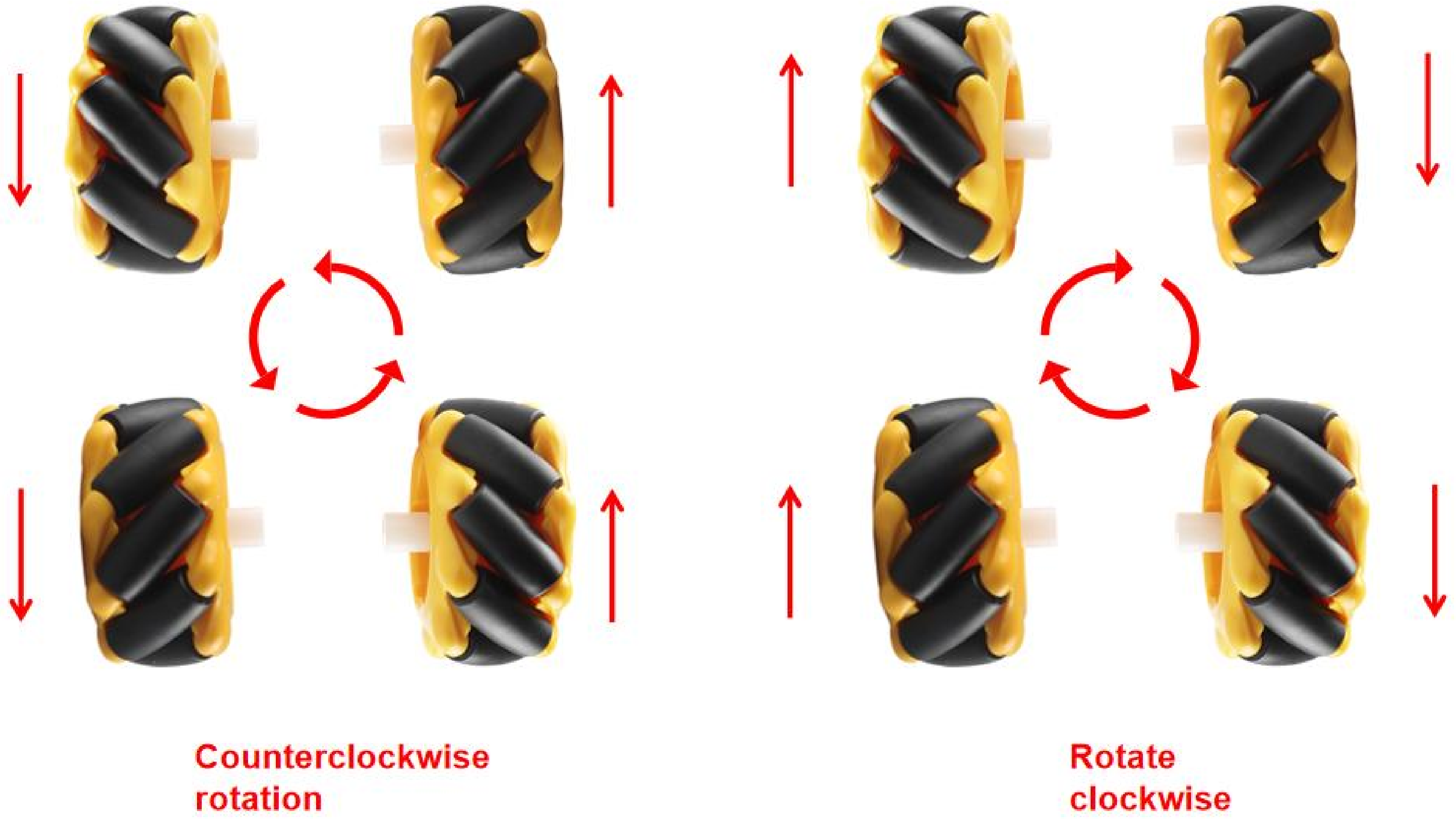


The different rotation directions of the wheels correspond to forward and backward movement:

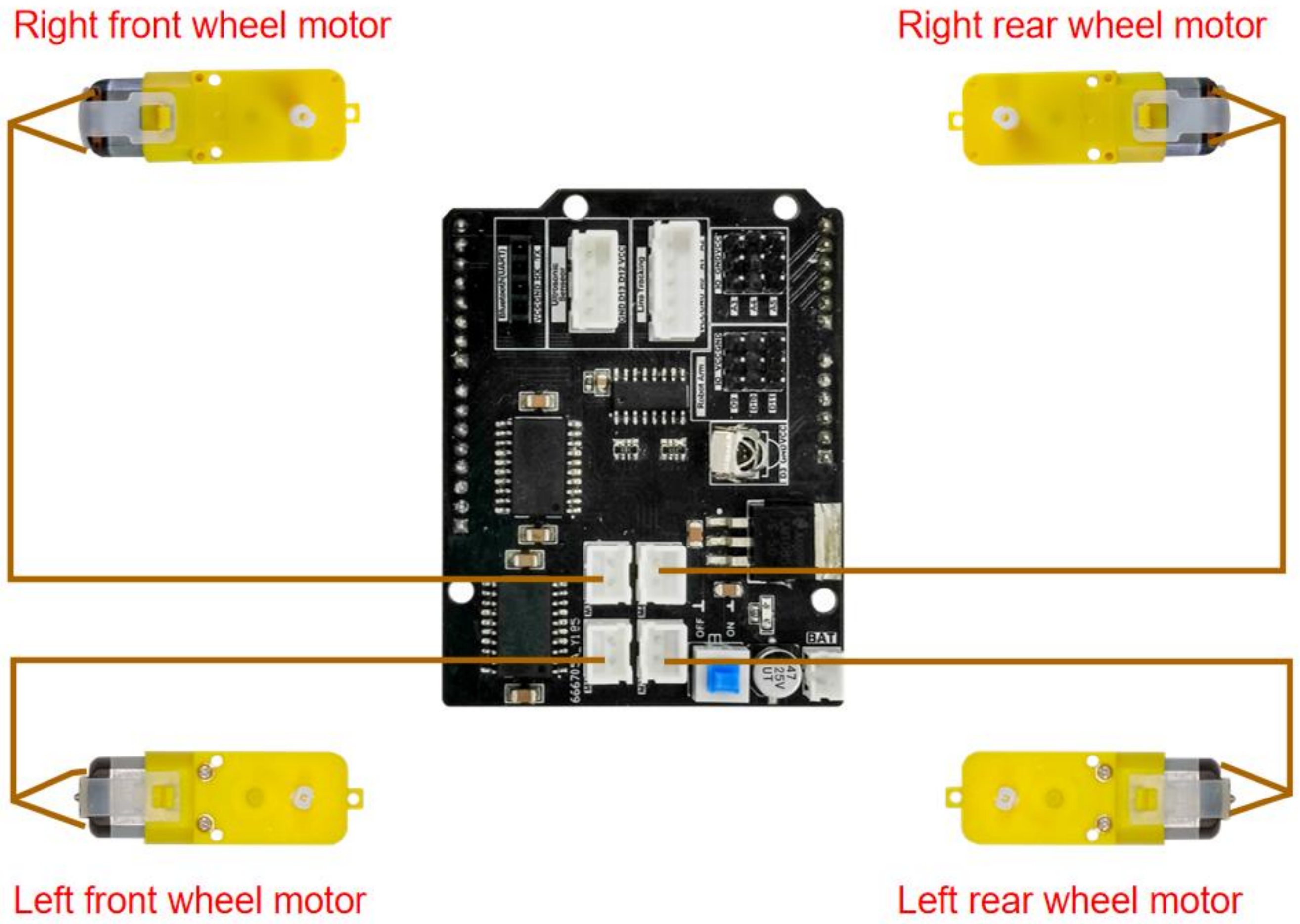




The different rotation directions of the wheels correspond to counterclockwise and clockwise rotations:



Wiring diagram:



code analysis

Open the program file (path: 4. Tutorial_Arduino\4_Arduino Code\2.1_Motor_Speed\2.1_Motor_Speed.ino)

Programming control principle:

The Pwm pin controls the power (speed) of the wheel, and then controls the rotation direction of each motor through the 74HC595 chip pin.

The shiftOut function of Arduino mainly acts on the 74HC595 chip;

main idea:

The high and low levels of each pin are controlled by the decimal numbers 0 to 255 for the 8-bit binary number;

Instructions:

```
shiftOut (dataPin, clockPin, bitOrder, value)
```

The shiftOut function has a total of four parameters, and the first three parameters are defined and configured at the beginning, we only need to modify the value of value. At this time, the system will convert the decimal numbers into 8-bit binary numbers to control the high and low levels.

Define PWM pins and chip pins

```
// PWM control pin
#define PWM1_PIN 5
#define PWM2_PIN 6
// 74HCT595N Chip pins
#define SHCP_PIN 2 // The displacement of the clock
#define EN_PIN 7 // Can make control
#define DATA_PIN 8 // Serial data
#define STCP_PIN 4 // Memory register clock
```

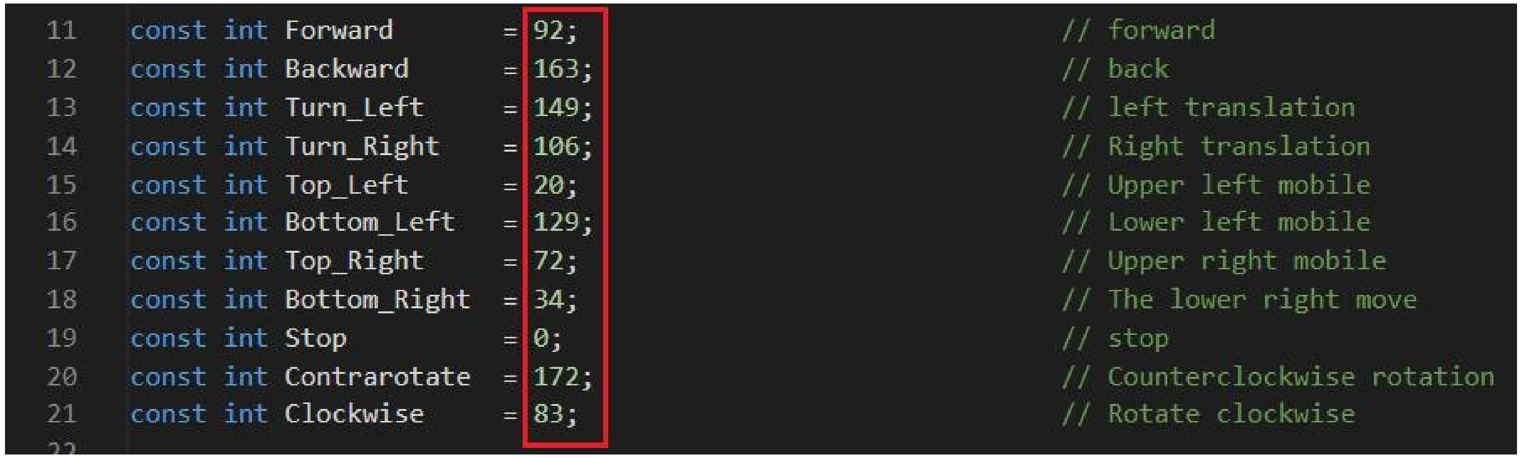
Set the variable to store the decimal encoded value

```
const int Forward = 92 ; // forward
const int Backward      = 163;                              // back
const int Turn_Left     = 149;                              // left translation
const int Turn_Right    = 106;                              // Right translation
const int Top_Left      = 20;                               // Upper left mobile
const int Bottom_Left   = 129;                              // Lower left mobile
const int Top_Right     = 72;                               // Upper right mobile
const int Bottom_Right  = 34;                               // The lower right move
const int Stop          = 0;                                // stop
const int Contrarotate = 172 ; // Counterclockwise rotation
const int Clockwise = 83 ; // Rotate clockwise
```

Motor drive function, pass in two values, one is the encoding value that controls the direction of motor rotation and the PWM power (speed) value

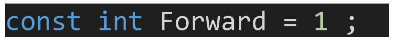
```
void Motor ( int Dir , int Speed )
{
   digitalWrite (EN_PIN, LOW);
   analogWrite (PWM1_PIN, Speed);
   analogWrite (PWM2_PIN, Speed);

   digitalWrite (STCP_PIN, LOW);
   shiftOut (DATA_PIN, SHCP_PIN, MSBFIRST, Dir);
   digitalWrite (STCP_PIN, HIGH);
}
```

Ways to debug encoded values:

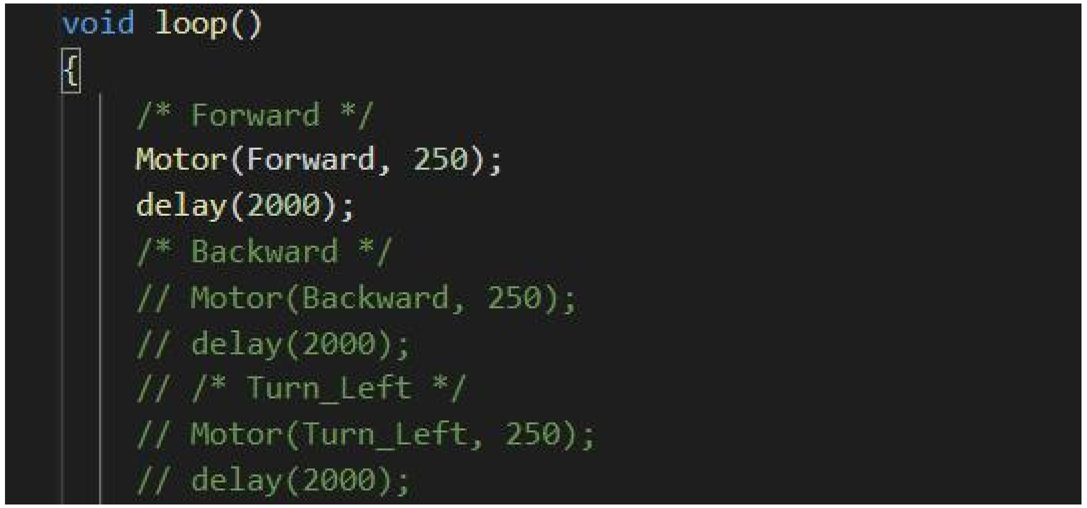


```
const int Forward       = 92;                          // forward
const int Backward      = 163;                         // back
const int Turn_Left     = 149;                         // left translation
const int Turn_Right    = 106;                         // Right translation
const int Top_Left      = 20;                          // Upper left mobile
const int Bottom_Left   = 129;                         // Lower left mobile
const int Top_Right     = 72;                          // Upper right mobile
const int Bottom_Right  = 34;                          // The lower right move
const int Stop          = 0;                           // stop
const int Contrarotate  = 172;                         // Counterclockwise rotation
const int Clockwise     = 83;                          // Rotate clockwise
```

Take the "Forward" variable as an example, set the variable value to 1 (change to 8-bit binary as 0000 0001), comment out other codes, and observe the motor rotation when calling the function.



```
const int Forward = 1 ;
```



```
void loop()
{
    /* Forward */
    Motor(Forward, 250);
    delay(2000);
    /* Backward */
    // Motor(Backward, 250);
    // delay(2000);
    // /* Turn_Left */
    // Motor(Turn_Left, 250);
    // delay(2000);
```

It can be seen that only one motor is rotating, which means that 0000 0001 is the code of the rotation direction of the motor. As shown in the figure below, when the variable is set to 2 (binary is 0000 0010), it represents another rotation state of another motor. When the variable is set to 4 (binary is 0000 0100), it represents another rotation state of another motor. In this way, the codes corresponding to 8 states of 4 motors * 2 rotation modes of forward and reverse are inferred. Summarizing the data record, if you want the car to move forward, you must keep the four motors in a forward rotation state, that is, 01011100, which is converted to 92 in decimal. Finally, all the corresponding codes of all motion states are obtained.

In particular, the potentials that control the same motor cannot be 1 (high level) at the same time, which will lead to failure.

The specific corresponding code table is as follows:

8 bit binary	1000 0000	0100 0000	0010 0000	0001 0000	0000 1000	0000 0100	0000 0010	0000 0001		
The decimal system	128	64	32	16	8	4	2	1		
State of the wheel	Upper left wheel back	Left upper wheel forward	Lower left wheel back	Lower left wheel forward	Lower right wheel forward	Upper right wheel forward	Upper right wheel back	Lower right wheel back	8 bit binary	decimal system
Forward	0	1	0	1	1	1	0	0	01011100	92
Back	1	0	1	0	0	0	1	1	10100011	163
left translation	1	0	0	1	0	1	0	1	10010101	149
Right translation	0	1	1	0	1	0	1	0	01101010	106
Upper left mobile	0	0	0	1	0	1	0	0	00010100	20
Lower left mobile	1	0	0	0	0	0	0	0	10000001	129
Upper right mobile	0	1	0	0	1	0	0	0	01001000	72
Lower right mobile	0	0	1	0	0	0	1	0	00100010	34
Counterclockwise	1	0	1	0	1	1	0	0	10101100	172
Rotate clockwise	0	1	0	1	0	0	1	1	01010011	83

The value of the rightmost column is passed into the variable Dir of the function shifOut, and different motion states can be obtained.

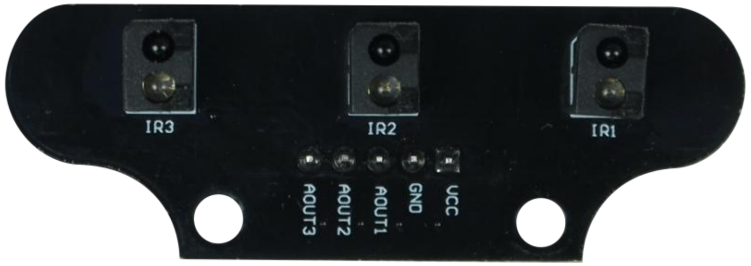
```
    shiftOut (DATA_PIN, SHCP_PIN, MSBFIRST, Dir);
```

Lesson 6　Tracking Module

Course Introduction

This course mainly learns the line-following sensor module and its application.

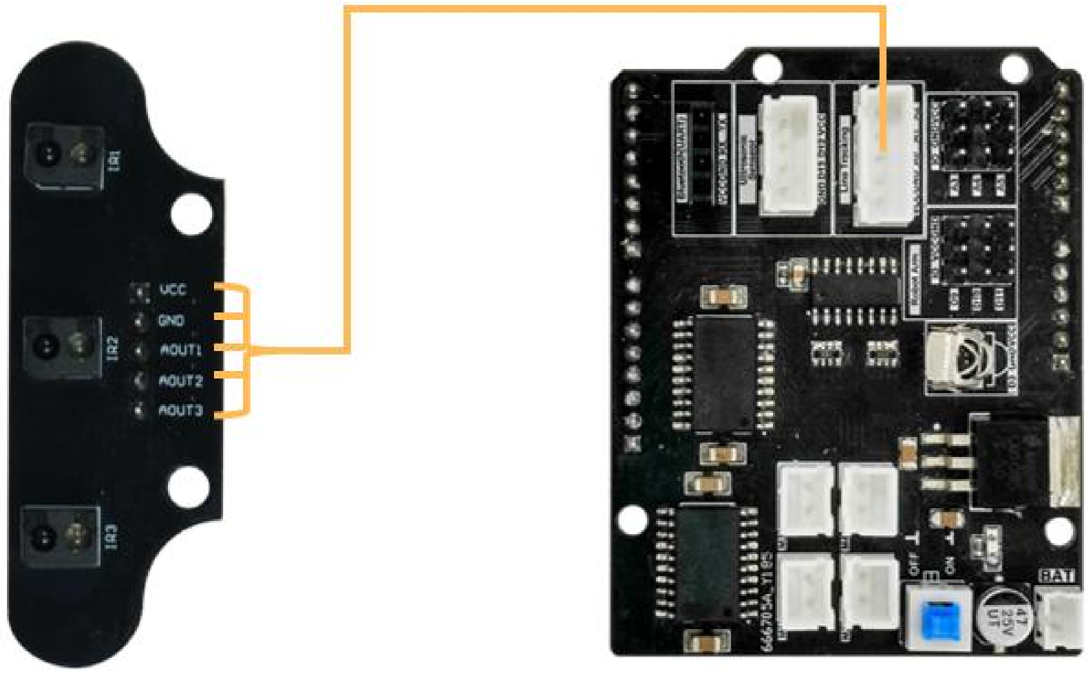
tracking module



tracking sensor, which is usually used in the manufacture of tracking smart cars. The tracker sensor adopts ITR20001/T infrared reflection sensor. The infrared emitting diode of the ITR2001/T sensor continuously emits infrared rays. When the emitted infrared rays are reflected by objects, they are received by the infrared receiver and output an analog value. The output analog value is related to object distance and object color. The position of the trace line is determined by calculating the analog values of the 3 outputs.

The tracking sensor is located at the front of the car and consists of an infrared transmitting tube and an infrared receiving tube. The former is an LED that transmits infrared light, and the latter is a photoresistor for receiving infrared light. The light reflectivity of a black surface is different from that of a white surface. Therefore, the intensity of reflected infrared light received by the car on a black road is different from that on a white road, and the motion changes. According to the principle of voltage division between series resistors, the movement path is determined by inferring the color of the route under the car from the voltage of the sensor.

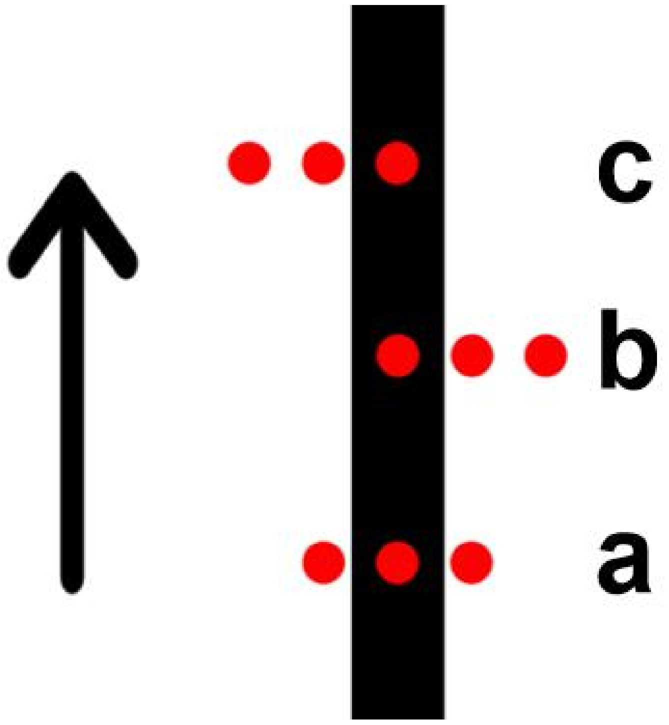
Wiring diagram:



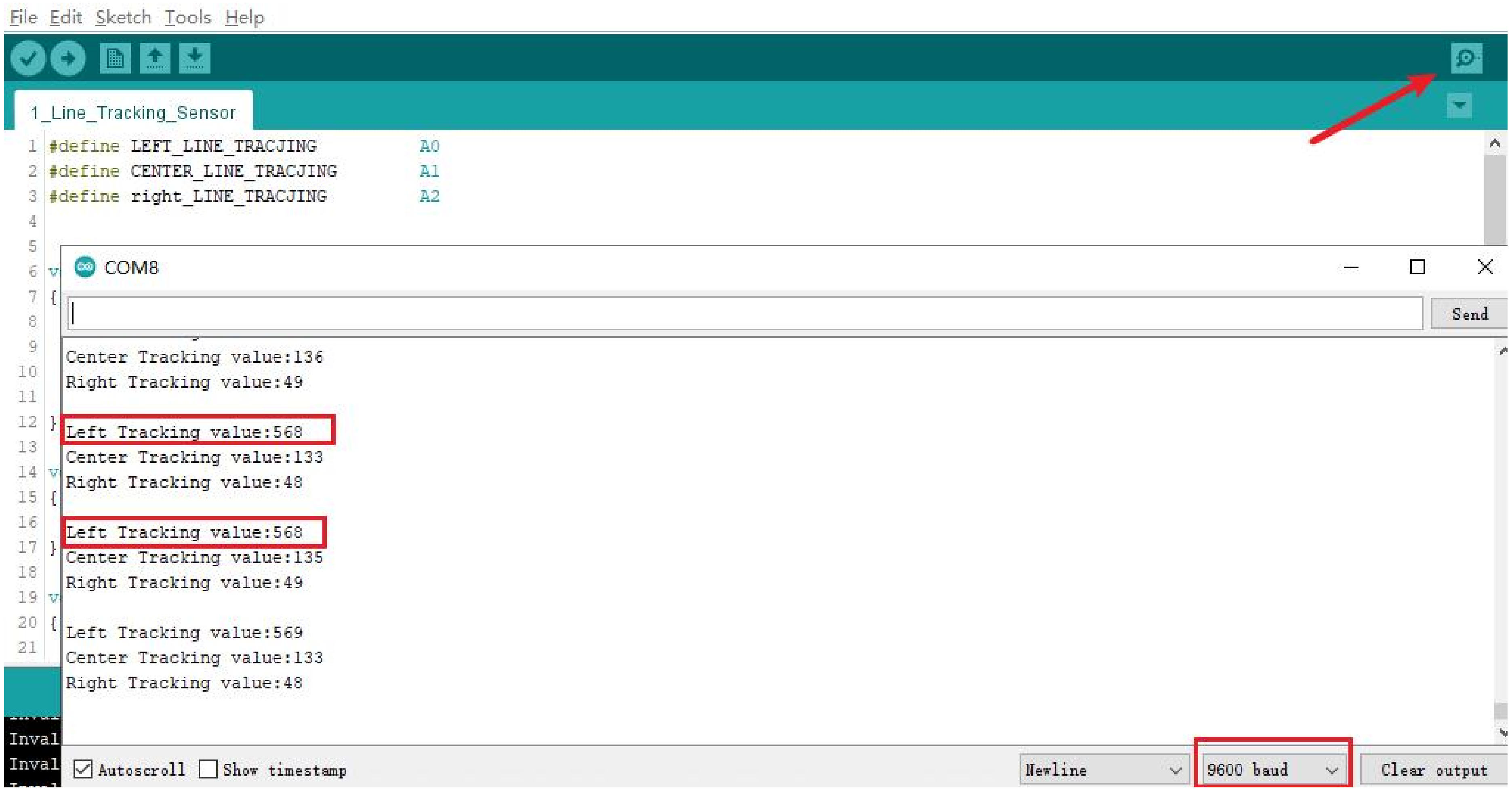
Get the measurement value of the three-way tracking sensor:

Open the code file (path: 4. Tutorial_Arduino\4_Arduino Code\3.1_Line_Tracking_Sensor\3.1_Line_Tracking_Sensor. I no)

Connect the mainboard of the car to the computer and burn the code. A black line (line width about 1.5cm) for testing is attached to the ground with black tape. Then place the car on the black line, click on the serial monitor in the upper right corner of the Arduino IDE, and observe the test value.



As shown in the figure (b) above, the sensor on the left side of the tracking module detects the black line and obtains the value:

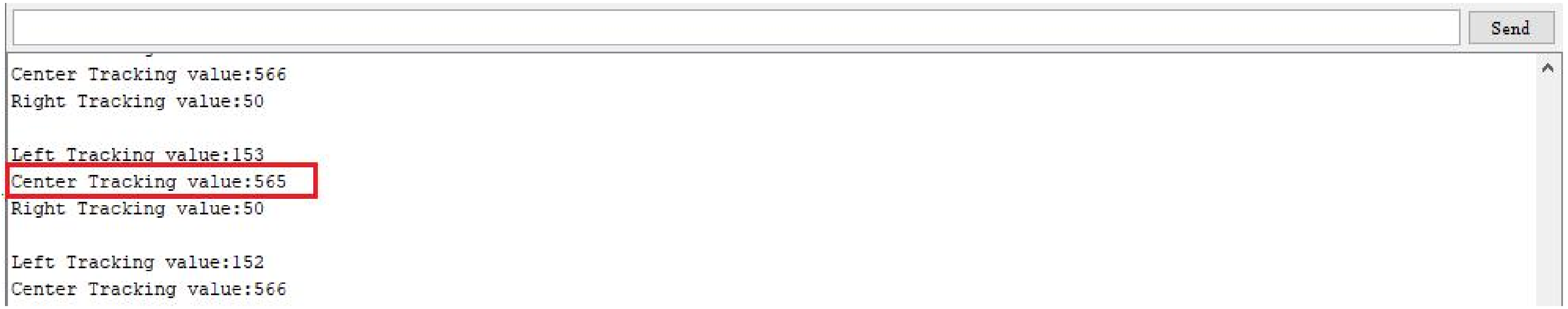


```
#define LEFT_LINE_TRACJING          A0
#define CENTER_LINE_TRACJING        A1
#define right_LINE_TRACJING         A2
```

```
Center Tracking value:136
Right Tracking value:49

Left Tracking value:568
Center Tracking value:133
Right Tracking value:48

Left Tracking value:568
Center Tracking value:135
Right Tracking value:49

Left Tracking value:569
Center Tracking value:133
Right Tracking value:48
```

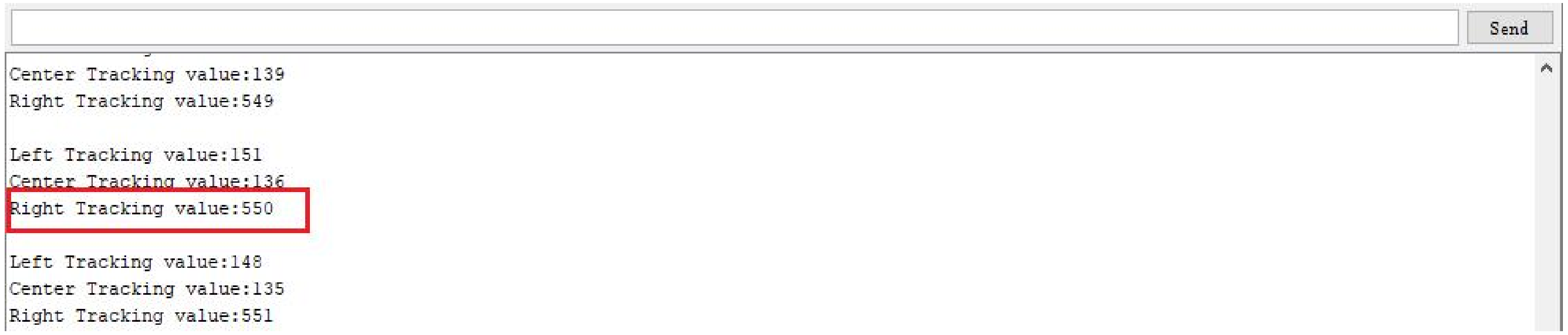
9600 baud

As shown in the figure (a) above, when the middle sensor of the tracking module detects the black line, the value is obtained:


Send

```
Center Tracking value:566
Right Tracking value:50

Left Tracking value:153
Center Tracking value:565
Right Tracking value:50

Left Tracking value:152
Center Tracking value:566
```

As shown in the figure (c) above, when the sensor on the right side of the tracking module detects the black line, the value is obtained:


Send

```
Center Tracking value:139
Right Tracking value:549

Left Tracking value:151
Center Tracking value:136
Right Tracking value:550

Left Tracking value:148
Center Tracking value:135
Right Tracking value:551
```

The result can be obtained: when the sensor detects the black line, the value is about 550~560, and when the black line is not detected, the value is less than 200. (Different materials and colors of tape and light have an impact on the test results, the tests here do not represent all results)

code analysis

Define the left, middle and right sensor pins of the tracking module as A0/A1/A2 respectively

```
#define LEFT_LINE_TRACJING A0
#define CENTER_LINE_TRACJING A1
#define right_LINE_TRACJING A2
```

Three pin ports set as inputs

```
void setup ()
{
    Serial.begin ( 9600 ) ; _
    pinMode (LEFT_LINE_TRACJING, INPUT);
    pinMode (CENTER_LINE_TRACJING, INPUT);
    pinMode (right_LINE_TRACJING, INPUT);
}
```

The analog value reads the value obtained by the tracking module, and then prints it out

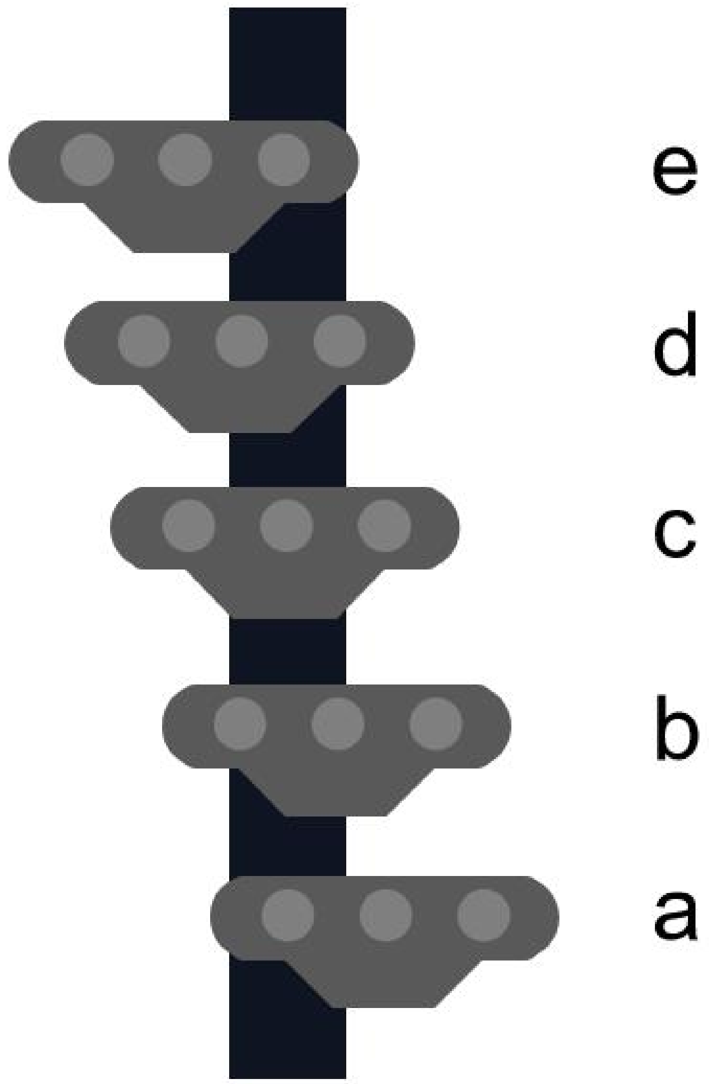
```
Left_Tra_Value = analogRead (LEFT_LINE_TRACJING);
Center_Tra_Value = analogRead (CENTER_LINE_TRACJING);
Right_Tra_Value = analogRead (right_LINE_TRACJING);
   Serial . print ( "Left Tracking value:" );
    Serial.println ( Left_Tra_Value ) ;
    Serial . print ( "Center Tracking value:" );
    Serial.println ( Center_Tra_Value ) ;
    Serial . print ( "Right Tracking value:" );
    Serial.println ( Right_Tra_Value ) ;
```

Lesson 7 Tracking Car

Course Introduction

Master the principle of line tracking and realize the function of three-way tracking trolley through Arduino programming.

tracking principle


e
d
c
b
a

a→ Only the left sensor detects the black line in the tracking module , at this time the car needs to turn to the left at a large angle ;

b→ The left and middle sensors of the tracking module detect the black line at the same time, and the car needs to turn to the left at a small angle ;

c → Only the middle sensor detects the black line in the tracking module , and the car can drive in a straight line at this time ;

d → The right and middle sensors of the tracking module detect the black line at the same time, and the car needs to turn at a small angle to the right ;

e → Only the right sensor detects the black line in the tracking module . At this time, the car needs to turn to the right at a large angle ;

Combining the above information, we can see the tracking principle of the tracking car . After the car starts, the tracking module only needs to sense the black lines on the road and make corresponding actions as needed. There are many more complex algorithms like PID. Therefore, after implementing the tracking function, you can learn more algorithms to control the car yourself.

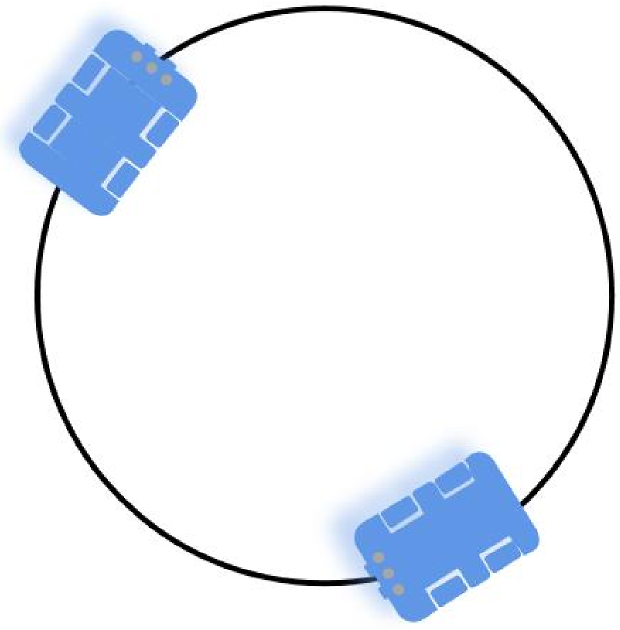
Prepare insulating tape (black tape) lines:

First, we need to make a runway ourselves. We can stick black tape on flat and clean ground. It is best to let the trajectory angle change slowly, not too much at once. Because if the angle of the turn is too large, the car may run off the track. However, if you want to make it more difficult, you can make the angle of the turn wider. The size of the runway is generally not less than 40*60 cm.

（1）Curved sections of the line should transition as smoothly as possible, otherwise there is a good chance the car will overtake the track.

（2）Line trace scenes can be made from black and white tape to design different walking paths.

（3）In addition to line tracing, we can develop other procedural line tracing principles. For example, confining a car to a certain area and letting it move freely will be covered in a later lesson .



code analysis

Open the code file (path: 4. Tutorial_Arduino\4_Arduino

Code\3.2_Line_Tracking_Smart_Car\3.2_Line_Tracking_Smart_Car.ino)

Define a three-way tracking measurement value variable and a reference value variable "Black_Line" used to compare whether a black line is detected.

```
int Left_Tra_Value;
int Center_Tra_Value;
int Right_Tra_Value;
int Black_Line = 500 ;
```

Comparing the value of the three-way tracking sensor with the reference value Black_Line, the five situations described in the tracking principle are obtained:

Case c only the middle sensor detects the black line :

```
if (Left_Tra_Value < Black_Line && Center_Tra_Value >= Black_Line && Right_Tra_Value < Black_Line)
{
        Motor (Forward, 175 );
}
```

Case b Left and middle sensors detect black lines at the same time :

```
else if (Left_Tra_Value >= Black_Line && Center_Tra_Value >= Black_Line && Right_Tra_Value < Black_Line)
{
        Motor (Contrarotate, 165 );
}
```

Case a Only the left sensor detects the black line :

```
    else if (Left_Tra_Value >= Black_Line && Center_Tra_Value < Black_Line && Right_Tra_Value < Black_Line)
{
        Motor (Contrarotate, 190 );
}
```

Case e only the right sensor detects the black line :

```
    else if (Left_Tra_Value < Black_Line && Center_Tra_Value < Black_Line && Right_Tra_Value >= Black_Line)
{
        Motor (Clockwise, 190 );
}
```

Case d The right and middle sensors detect the black line at the same time :

```
    else if (Left_Tra_Value < Black_Line && Center_Tra_Value >= Black_Line && Right_Tra_Value >= Black_Line)
{
        Motor (Clockwise, 165 );
}
```

Plus the case where all the sensors detect the black line:

```
    else if (Left_Tra_Value >= Black_Line && Center_Tra_Value >= Black_Line && Right_Tra_Value >= Black_Line)
{
        Motor (Stop, 0 );
}
```

Execute the Motor() function with parameters

```
Motor ( int Dir , int Speed )
```

Lesson 8 Draw a dungeon for a prison car

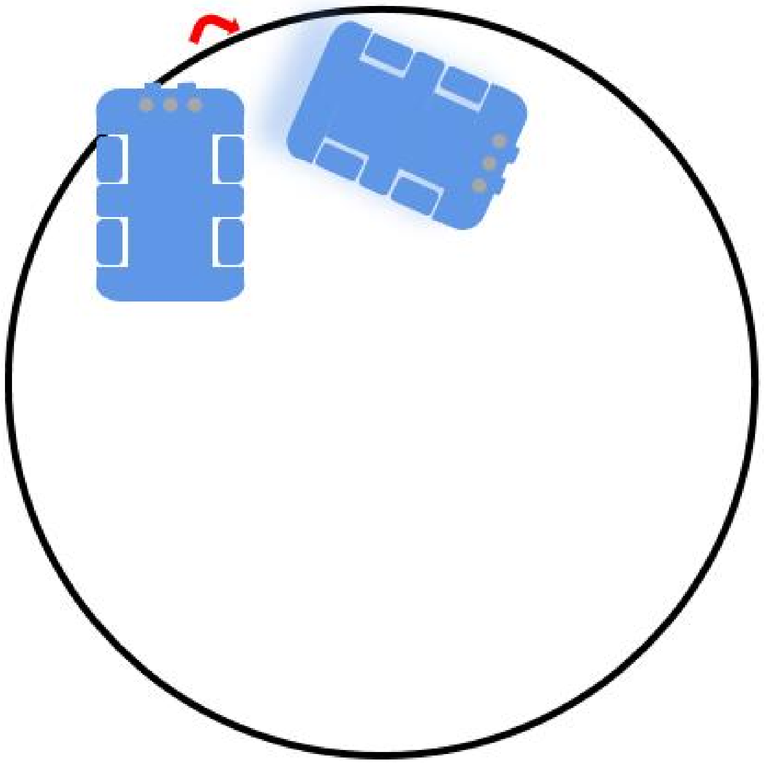
Course Introduction

This course mainly deepens the understanding of the tracking module through an interesting "drawing the ground as a prison" project, and learns to apply the tracking module more flexibly in different scenarios.

Drawing the ground as a prison: As the name suggests, it is to circle an area on the ground, and let the car robot run freely in the circle without rushing out of the area, just like staying in a dungeon.

Circle the dungeon area on the ground

Prepare black tape and stick an area with a radius of not less than 40cm on a clean and flat ground. Note that the circled area must be closed, otherwise the car will rush out of the gap.



code analysis

Open the code file (path: 4. Tutorial_Arduino\4_Arduino Code\3.3_cage\3.3_cage.ino)

Compare the value of the three-way tracking sensor with the reference value Black_Line, and get 5 cases:

None of the three sensors detected the black line, that is, they are all smaller than Black_Lin

```
    if (Left_Tra_Value < Black_Line && Center_Tra_Value < Black_Line && Right_Tra_Value < Black_Line)
{
        Motor (Forward, 150 );
}
```

Only the left sensor detects the black line, that is, the left side of the car is about to exceed the black line. At this time, you need to back up and then turn right (rotate clockwise) to adjust the direction to walk.

```
else if (Left_Tra_Value >= Black_Line && Center_Tra_Value < Black_Line && Right_Tra_Value < Black_Line)
{
        Motor (Backward, 150 );
        delay ( 200 );
        Motor (Clockwise, 160 );
        delay ( 500 );
}
```

When the left and middle sensors detect the black line, you need to back up and then turn right to adjust the direction at a larger angle to walk;

```
    else if (Left_Tra_Value >= Black_Line && Center_Tra_Value >= Black_Line && Right_Tra_Value < Black_Line)
{
        Motor (Backward, 150 );
        delay ( 200 );
        Motor (Clockwise, 160 );
        delay ( 600 );
}
```

The same is true for the sensor on the right. After detecting the black line, it will back up and then turn left (rotate counterclockwise) to adjust the direction to walk;

```

    else if (Left_Tra_Value < Black_Line && Center_Tra_Value < Black_Line && Right_Tra_Value >= Black_Line)
    {
        Motor(Backward,150);
        delay(200);
        Motor(Contrarotate,160);
        delay(500);
    }
    else if (Left_Tra_Value < Black_Line && Center_Tra_Value >= Black_Line && Right_Tra_Value >= Black_Line)
{
        Motor (Backward, 150 );
        delay ( 200 );
        Motor (Contrarotate, 160 );
        delay ( 600 );
}
```

Otherwise, when the three-way tracking module detects the black line, it rotates clockwise and adjusts the direction to walk by default.

```
    else
{
        Motor (Backward, 150 );
        delay ( 600 );
        Motor (Clockwise, 160 );
        delay ( 500 );
}
```

Lesson 9 Ultrasonic Ranging

Course Introduction

This class mainly understands the working principle of the ultrasonic module, masters the connection of the ultrasonic circuit diagram, and learns how to measure the distance of the ultrasonic module through programming.

Ultrasonic sensor

Sound waves are produced by vibrations and can travel at different speeds in different media. Ultrasound has the advantages of strong directionality, slow energy loss, and long propagation distance in the medium, and is often used for ranging. Such as distance meter, liquid level measuring instrument, etc. can be realized by ultrasonic.

Electrical parameters	HC-SR04 Ultrasonic module
Working voltage	DC-5V
Working current	15mA
Working frequency	40KHz
Maximum range	4m
Minimum range	2cm
Measuring angle	15 °
Input trigger signal	10 US TTL pulse
Output echo signal	Output TTL level signal, proportional to the range
Size	45*20*15

Ultrasonic ranging is a non-contact detection method

Especially for aerial ranging, due to the slow wave speed in the air, the echo signals contained in the propagation direction of the structural information are easy to detect and have very high resolution, so the accuracy is higher than other methods; while the ultrasonic sensor has a simple structure , small size, reliable signal processing and so on. Using ultrasonic testing is often faster, more convenient, simple in calculation, easy to realize real-time control, and can meet the requirements of industrial practicality in terms of measurement accuracy.

There are many methods of ultrasonic ranging. The principle of this system in ultrasonic measurement is to detect the transmission time of ultrasonic waves from ultrasonic transmitters to receivers through gas medium. Multiply this time by the speed of sound in the gas to get the distance the sound travels.

The ultrasonic transmitter emits ultrasonic waves in a certain direction, and the MCU starts timing at the same time. The ultrasonic waves are transmitted in the air, and they return immediately when they encounter obstacles on the way. The ultrasonic receiver receives the reflected waves and stops the timing immediately.

From the time T recorded by the timer, the distance (s) from the launch point to the obstacle can be calculated .

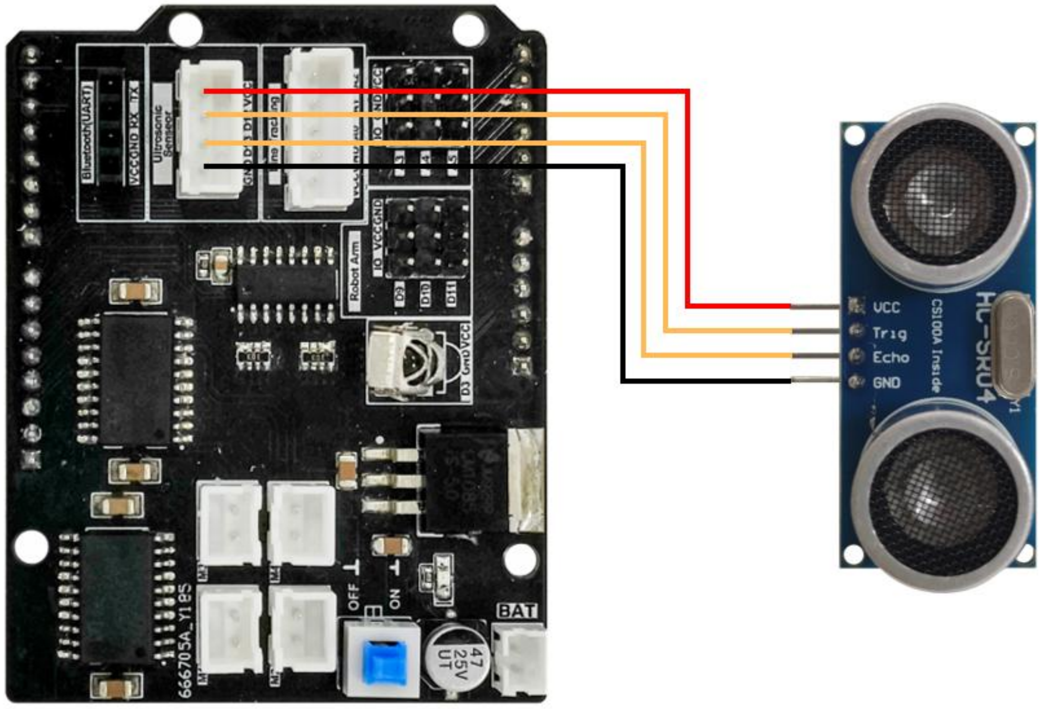
Formula: S = VT/2

Four factors limit the maximum measurable distance of an ultrasound system: the amplitude of the ultrasound, 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 directly receive the acoustic pulse will determine the minimum measurable distance. trigger signal input terminal (TRIG) will input a high-level signal of more than 10 microseconds, and the ultrasonic transmitter will automatically send 8 square waves of 40Hz after receiving the signal. At the same time, the timer will start. When the sensor receives the echo, it stops timing and outputs the echo signal.

According to the time interval, the distance can be calculated by the formula:

distance = (high level time * speed of sound) / 2 .

Wiring diagram



code analysis

Open the code file (path: 4. Tutorial_Arduino\4_Arduino Code\4.1_Ultrasonic_Sensor_Module)

Define Ultrasonic Control Pins

```
// Ultrasonic control pin
const int Trig = 12 ;
const int Echo = 13 ;
```

Get the ranging distance function SR04( )

```
float SR04 ( int Trig , int Echo )
{
    digitalWrite (Trig, LOW);
    delayMicroseconds ( 2 );
    digitalWrite (Trig, HIGH);
    delayMicroseconds ( 10 );
    digitalWrite (Trig, LOW);
    float distance = pulseIn (Echo, HIGH) / 58.00 ;
    delay ( 10 );
    return distance;
}
```

The pulseIn function is actually a function to measure the pulse width, and the default unit is us. That is to say, what pulseIn measures is the time elapsed from transmitting to receiving ultrasonic waves.

The speed of sound in dry, 20 degree Celsius air is about 343 m / s , which means that it takes 29.15 microseconds to travel 1 cm. Transmit plus receive takes double the time, so about 58.3 microseconds, take the value 58.

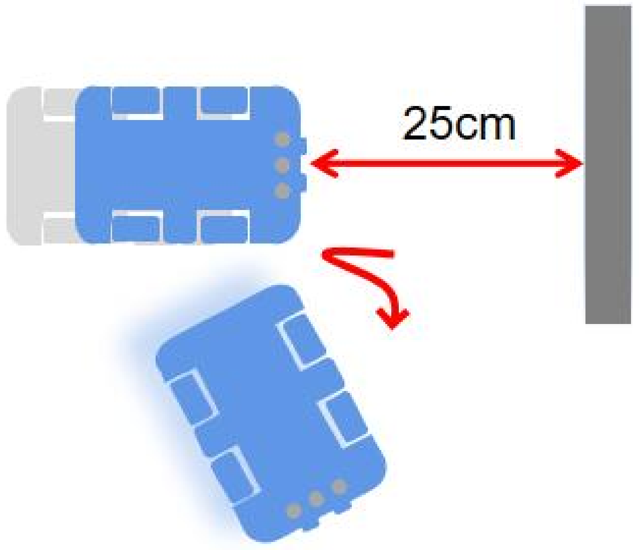
Course 10 Ultrasonic Obstacle Avoidance Vehicle

Course Introduction

This class mainly learns to consolidate the practical application of ultrasonic waves. On the basis of the previous class, learn the principle of ultrasonic obstacle avoidance and realize the obstacle avoidance function of the obstacle avoidance car through programming.

Obstacle Avoidance Principle

When the distance detected by the ultrasonic wave is less than the set distance, it is judged that the car encounters an obstacle, and then the obstacle avoidance program is triggered , so that the car backs up and then turns left or right to avoid the obstacle, so as to realize the automatic driving of the car to avoid the obstacle.



code analysis

Open the code file (path: 4. Tutorial_Arduino\4_Arduino Code\4.2_Ultrasonic_Obstacle_Avoidance_Robot_Car)

Save the ultrasonic measurement distance obtained by the SR04 function to the variable Avoidance_distance. The first "if" condition judges whether the distance is less than or equal to 25. If the distance between the car and the object in front is less than or equal to 25, then judge whether it is less than 15. If yes, execute the internal Motor function to stop the car, back up and turn clockwise, otherwise Just turn counterclockwise. If the first "if" condition is not met, keep going straight.

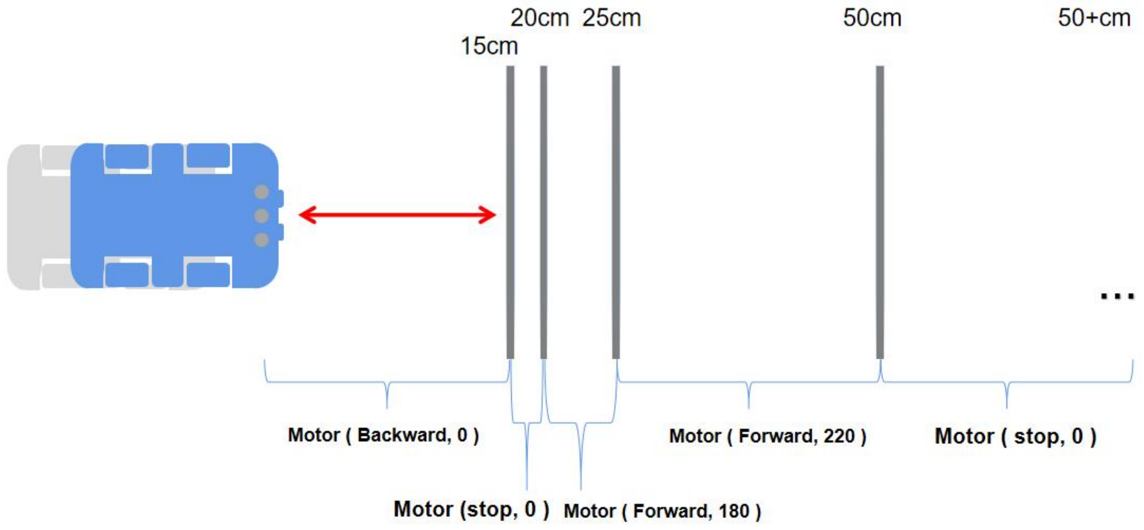
```
Avoidance_distance = SR04(Trig, Echo);
    if (Avoidance_distance <= 25)
    {
        if (Avoidance_distance <= 15)
        {
            Motor(Stop, 0);
            delay(100);
            Motor(Backward, 180);
            delay(600);
            Motor(Clockwise, 180);
            delay(200);
        }
        else{
            Motor(Stop, 0);
            delay(100);
            Motor(Backward, 180);
            delay ( 300 );
            Motor (Contrarotate, 180 );
            delay ( 600 );
    }
}
    else{
        Motor (Forward, 180 );
}
```

Lesson 11 Follow the car

Course Introduction

After learning the application of ultrasound in obstacle avoidance, this lesson will learn a follow-up system. Through the ultrasonic ranging in front of the car, it always keeps a certain distance from the object in front of it.

Follow the principle diagram



Combine the above figure with code analysis

Open the code file (path: 4. Tutorial_Arduino\4_Arduino Code\4.3_Ultrasonic_Follow)

When the ultrasonic wave in front of the car detects that the distance from the object in front is 20~25cm, the car moves forward with an analog power of 180;

```
 else if ( 20 <= Avoidance_distance && Avoidance_distance <= 25 )
{
       Motor (Forward, 180 );
}
```

When the distance between the car and the object in front is 25~50cm, the car moves forward with the analog power of 220;

```
   else if ( 25 <= Avoidance_distance && Avoidance_distance <= 50 )
{
       Motor (Forward, 220 );
}
```

When the distance between the car and the object in front is less than 15, the car backs up with 200 analog power;

```
if (Avoidance_distance < 15 )
{
        Motor (Backward, 200 );
}
```

Otherwise, the car stops when the distance between the car and the object in front exceeds 50cm or there is no object;

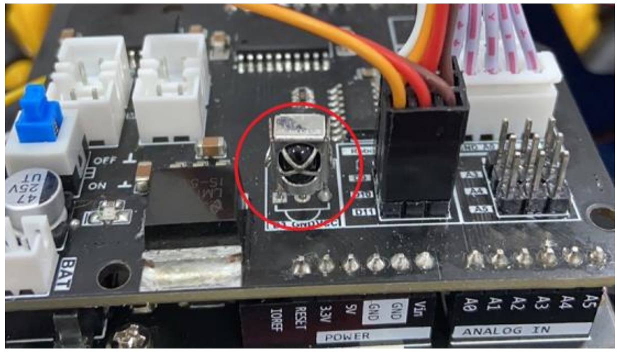
```
    else
{
        Motor (Stop, 0 );
}
```

Lesson 12 Infrared Remote Control Cars

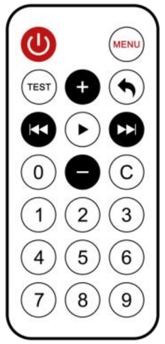
Course Introduction

Infrared remote control is a widely used remote control method. The car is already equipped with an IR receiver, thus allowing it to be controlled using an IR remote. This class mainly understands the infrared remote control and receiver , the principle of infrared remote control and the realization of infrared remote control programming. We have upgraded the function and added the operation of the robotic arm controlled by the servo motor. Let's feel the charm of controlling the robotic arm together.

Infrared receiver



Infrared remote control



Infrared remote control principle

The universal infrared remote control system consists of two parts: sending and receiving. The sending part consists of an infrared remote control, and the receiving part consists of an infrared receiving tube. The signal sent by the infrared remote control is a series of binary pulse codes. In order to be free from interference by other infrared signals during wireless transmission, it is generally necessary to modulate at a given carrier frequency, and then transmit through an infrared-emitting phototransistor. The infrared receiving tube filters out other noise waves, only receives the signal of a given frequency, and restores it to the demodulated binary pulse code. The built-in receiving tube sent by the infrared light-emitting diode converts the optical signal, and the amplifier in the IC amplifies the signal, and restores the original code sent by the remote control through automatic gain control, band-pass filtering, demodulation, and wave formation, and outputs the signal through the infrared receiving module. The pin identifies the circuit that enters the appliance.

The encoding scheme that matches the infrared remote control protocol is NEC protocol. Next, let us understand what is the NEC protocol.

(1) 8 address bits, 8 sequential bits address bits and sequential bits are transmitted twice to ensure reliability

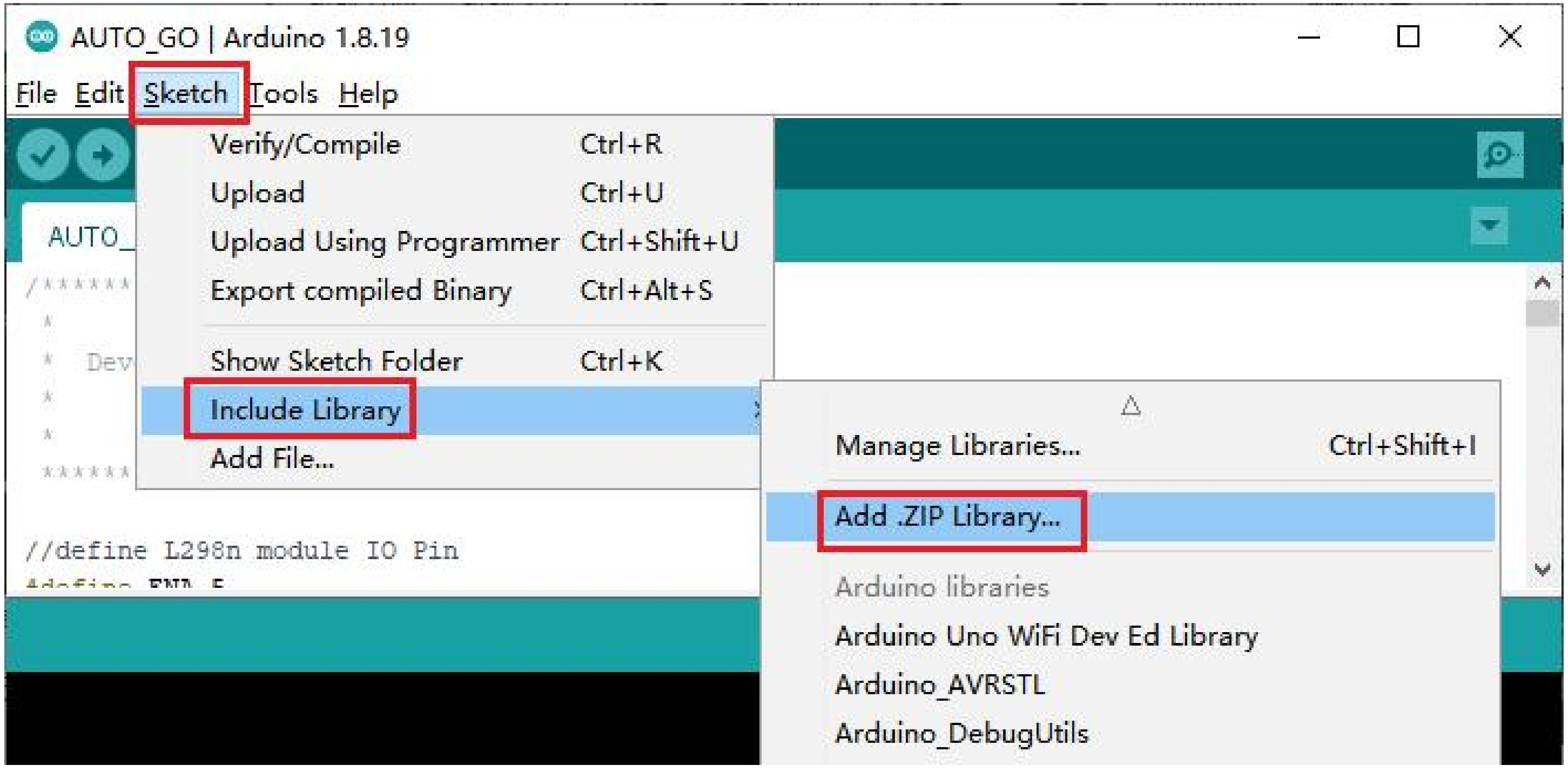
(3) Pulse position modulation

(4) The carrier frequency is 38 kHz

(5) The time per bit is 1.125 ms or 2.25 ms

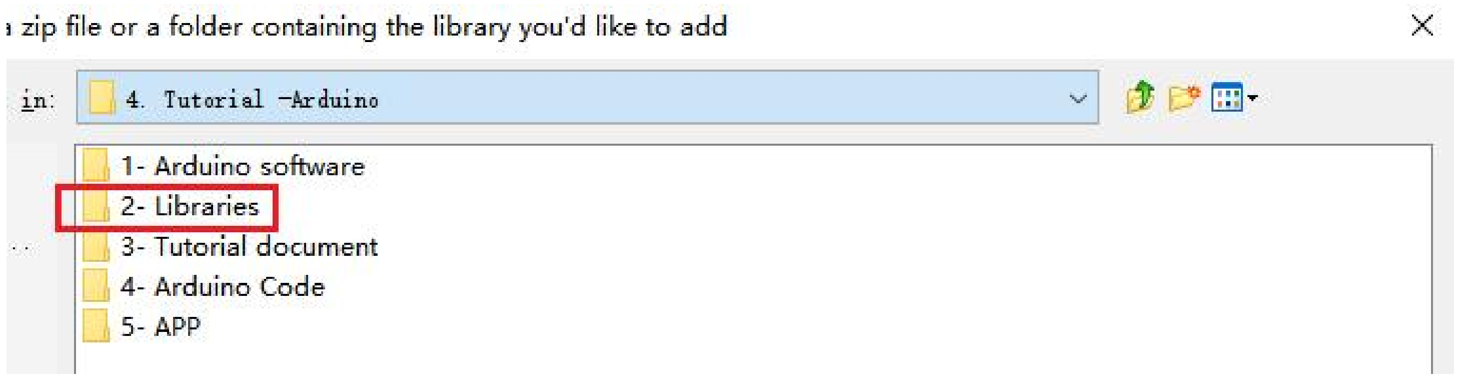
Add library files

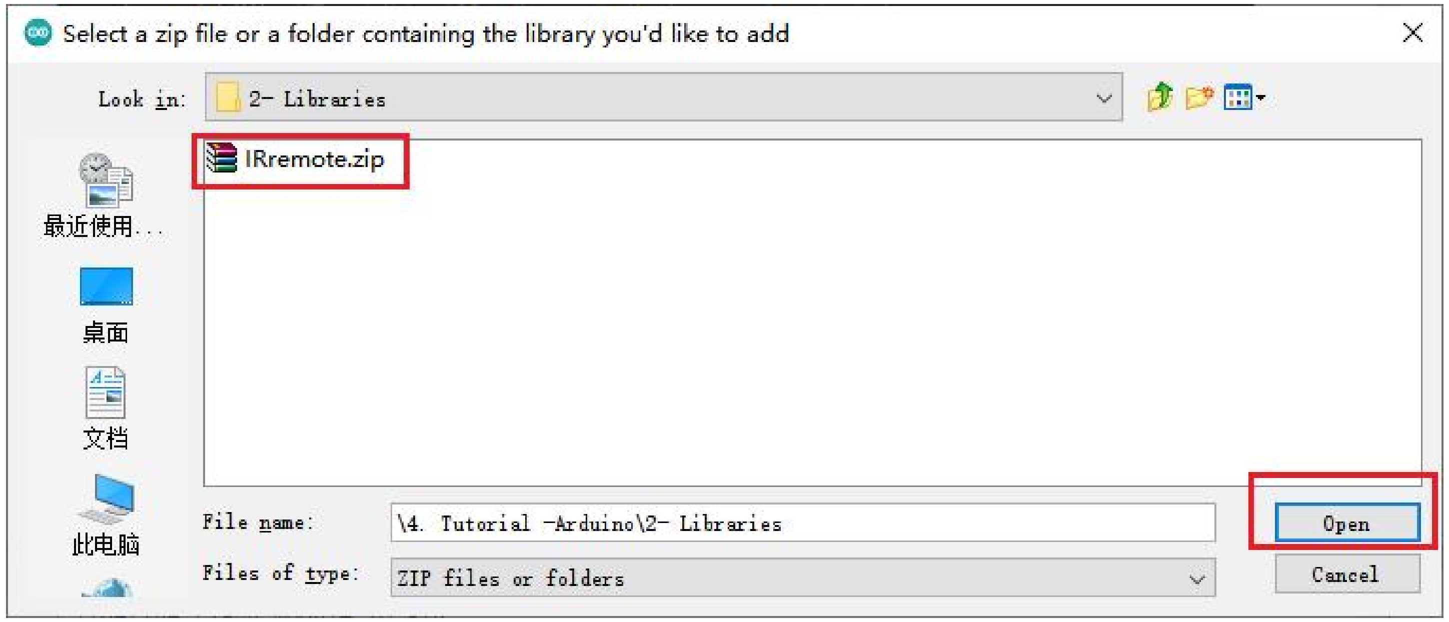
In this program, we need to use the "IR Remote Control" related library, so first we need to add the library file. Connect the control board to the computer and turn on the Arduino IDE , click Sketch >> Include Library >> Add .ZIP Library


AUTO_GO | Arduino 1.8.19
File Edit Sketch Tools Help
Verify/Compile Ctrl+R
Upload Ctrl+U
Upload Using Programmer Ctrl+Shift+U
Export compiled Binary Ctrl+Alt+S
Show Sketch Folder Ctrl+K
Include Library
Add File...
Manage Libraries... Ctrl+Shift+I
Add .ZIP Library...
Arduino libraries
Arduino Uno WiFi Dev Ed Library
Arduino_AVRSTL
Arduino_DebugUtils
AUTO_

```
//define L298n module IO Pin
```

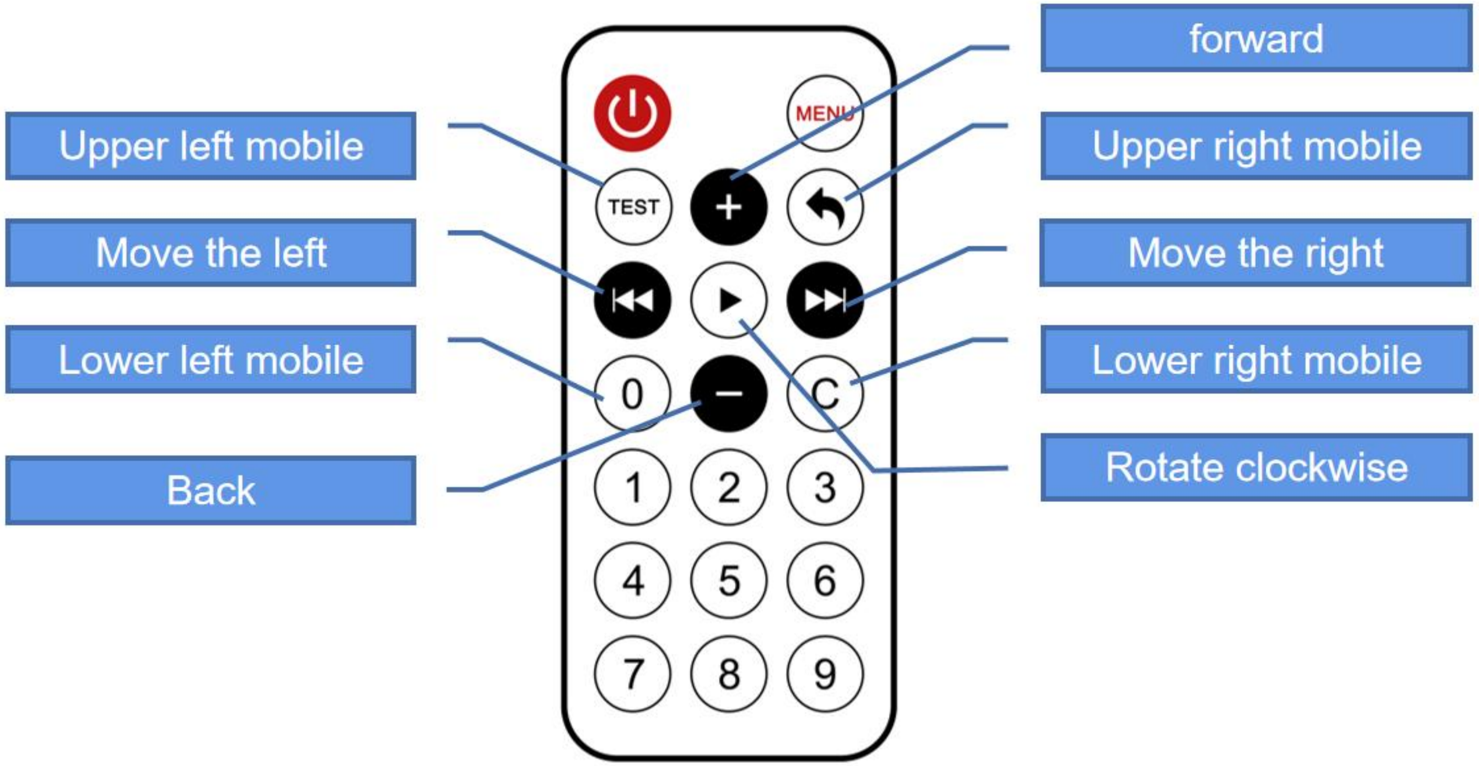
Select the " 4. Tutorial -Arduino >> 2-Libraries" folder, and then select the compressed package to open

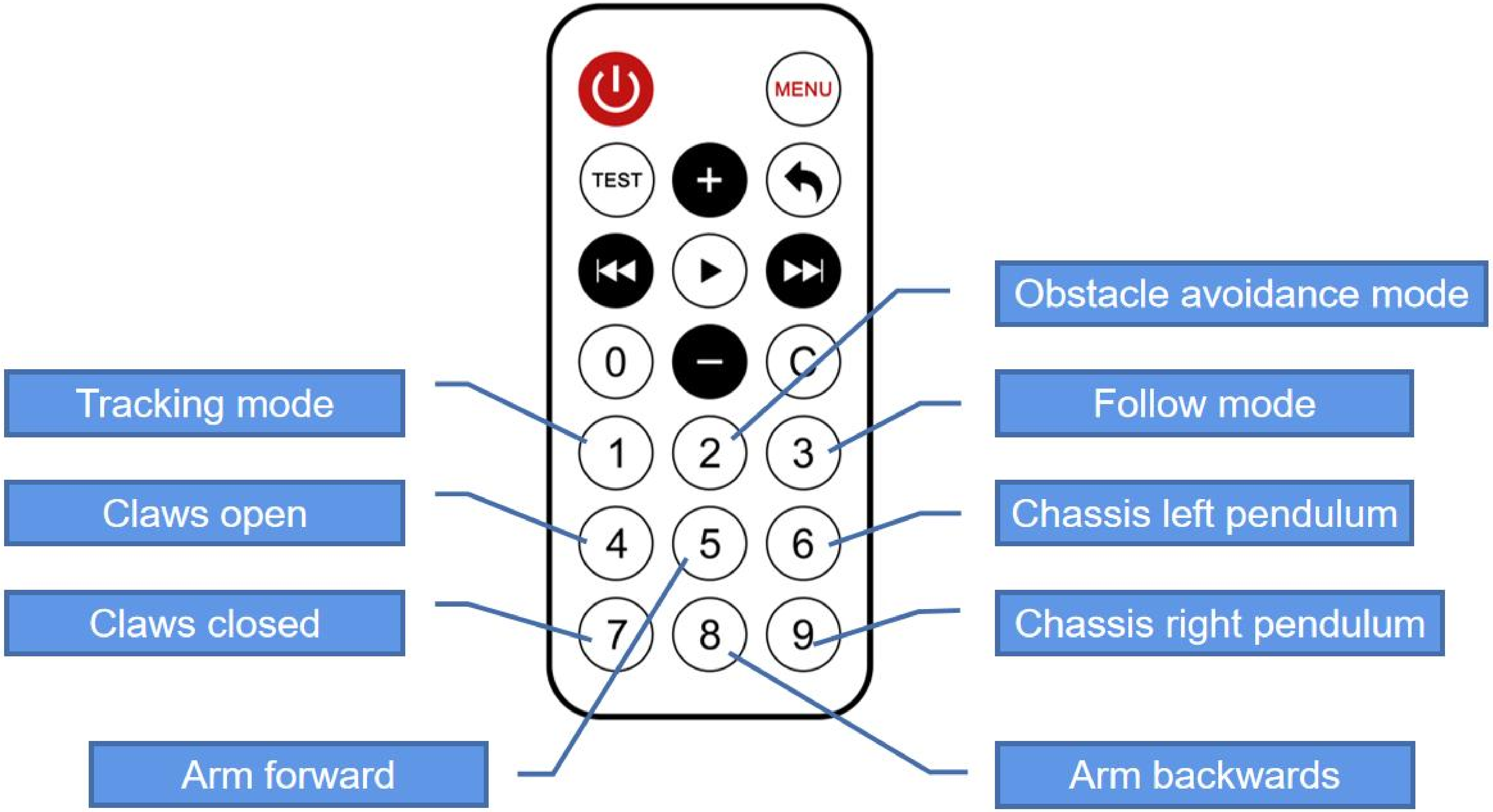



Select a zip file or a folder containing the library you'd like to add
Look in: 2- Libraries
IRremote.zip
最近使用...
桌面
文档
此电脑
File name: \4. Tutorial -Arduino\2- Libraries
Files of type: ZIP files or folders
Open
Cancel

IR remote control settings

In the car experiment, we need to control the car to move in all directions and the robotic arm action , which means that we need to remote control buttons and settings to send and receive information.




MENU
TEST
0
C
Obstacle avoidance mode
Tracking mode
1
2
3
Follow mode
Claws open
4
5
6
Chassis left pendulum
Claws closed
7
8
9
Chassis right pendulum
Arm forward
Arm backwards

Code Analysis:

Open the code file with Arduino IDE (path: " 4.Tutorial_Arduino\4_Arduino Code\ 5_IRID \ 5_IRID.ino ")

pull out the Bluetooth before uploading the code and install it after uploading!

Define the pins of the IR receiver

```
// Infrared receiving control pin
#define RECV_PIN       3
```

Enable infrared module

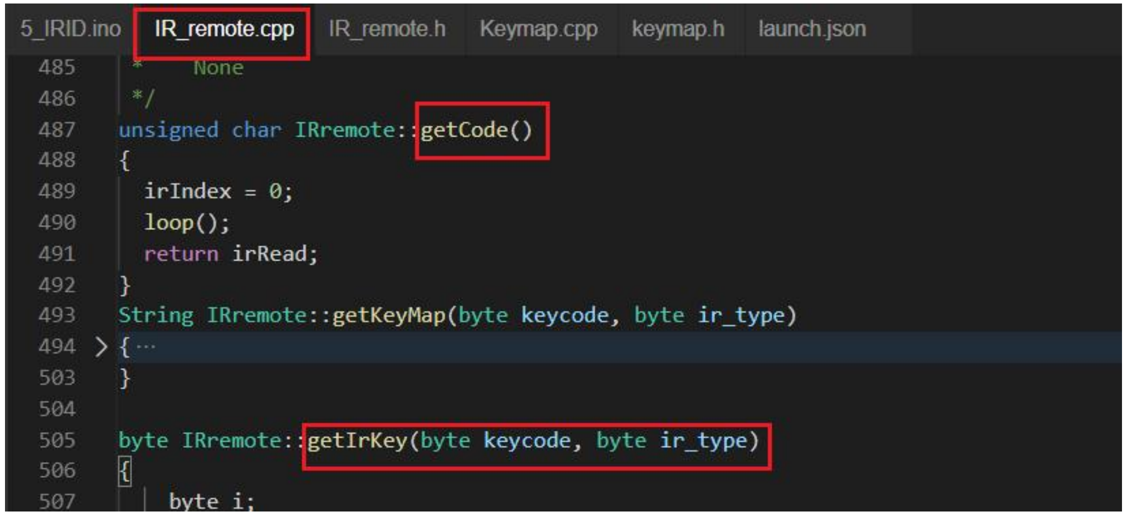
```
IRremote IR ( RECV_PIN );
```

Initialize tracking, obstacle avoidance and following modes to off

```
boolean Line_tracking_Function_flag = false ;
boolean Avoidance_Function_flag = false ;
boolean Following_Function_flag = false ;
```

Receive the information sent by the infrared remote control for analysis and judgment, and trigger the car to realize various actions, including the switching of the other two control modes. Among them, IR.getIrKey() and IR.getCode() are well encapsulated in the IR_remote.cpp file, and the corresponding data can be obtained directly by calling them.

```
switch (IR.getIrKey(IR.getCode(), IR_TYPE_EM))
   {
       case EM_IR_KEYCODE_UP:      // Forward
           Motor(Forward, 200);
           delay(200);
           break;
       case EM_IR_KEYCODE_DOWN:    // Backward
           Motor (Backward, 200 );
           delay ( 200 );
           break ;
```


5_IRID.ino IR_remote.cpp IR_remote.h Keymap.cpp keymap.h launch.json

```
 *     None
 */
unsigned char IRremote::getCode()
{
  irIndex = 0;
  loop();
  return irRead;
}
String IRremote::getKeyMap(byte keycode, byte ir_type)
{…
}

byte IRremote::getIrKey(byte keycode, byte ir_type)
{
    byte i;
```

Function corresponding to each function

car mobility

```
void Motor ( int Dir , int Speed )
```

Tracking function

```
void Line_tracking_Function ()
```

follow function

```
void Following_Function ()
```

Obstacle avoidance function

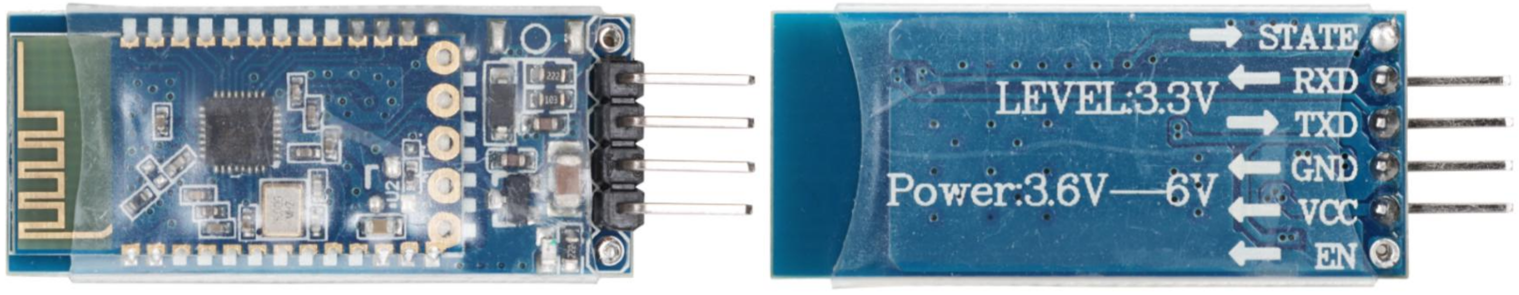
```
void Avoidance_Function ()
```

Lesson 13 Bluetooth Remote Control

Course Introduction

Bluetooth remote control is a very convenient and efficient control method. The car is already equipped with a bluetooth module , so it can be controlled via a bluetooth APP . This class mainly understands the principle of bluetooth remote control, can wirelessly control your car in a specific space and learn the programming and implementation ideas of bluetooth remote control.

BT06 Bluetooth



Bluetooth is a wireless technology standard used to exchange data at close range between different devices using short-wave ultra-high frequency radio waves in the radio frequency band (2.400 to 2.485 GHz) in various fields such as industry, science and medicine. The car is equipped with a BT06 Bluetooth, and a Bluetooth APP for remote control is prepared in the data package: btcontroller.apk. BT06 bluetooth has 4 pins, it can only work if connected correctly, otherwise it will damage the bluetooth. Note that one side of the Bluetooth has been marked, and the pin placement is as follows:

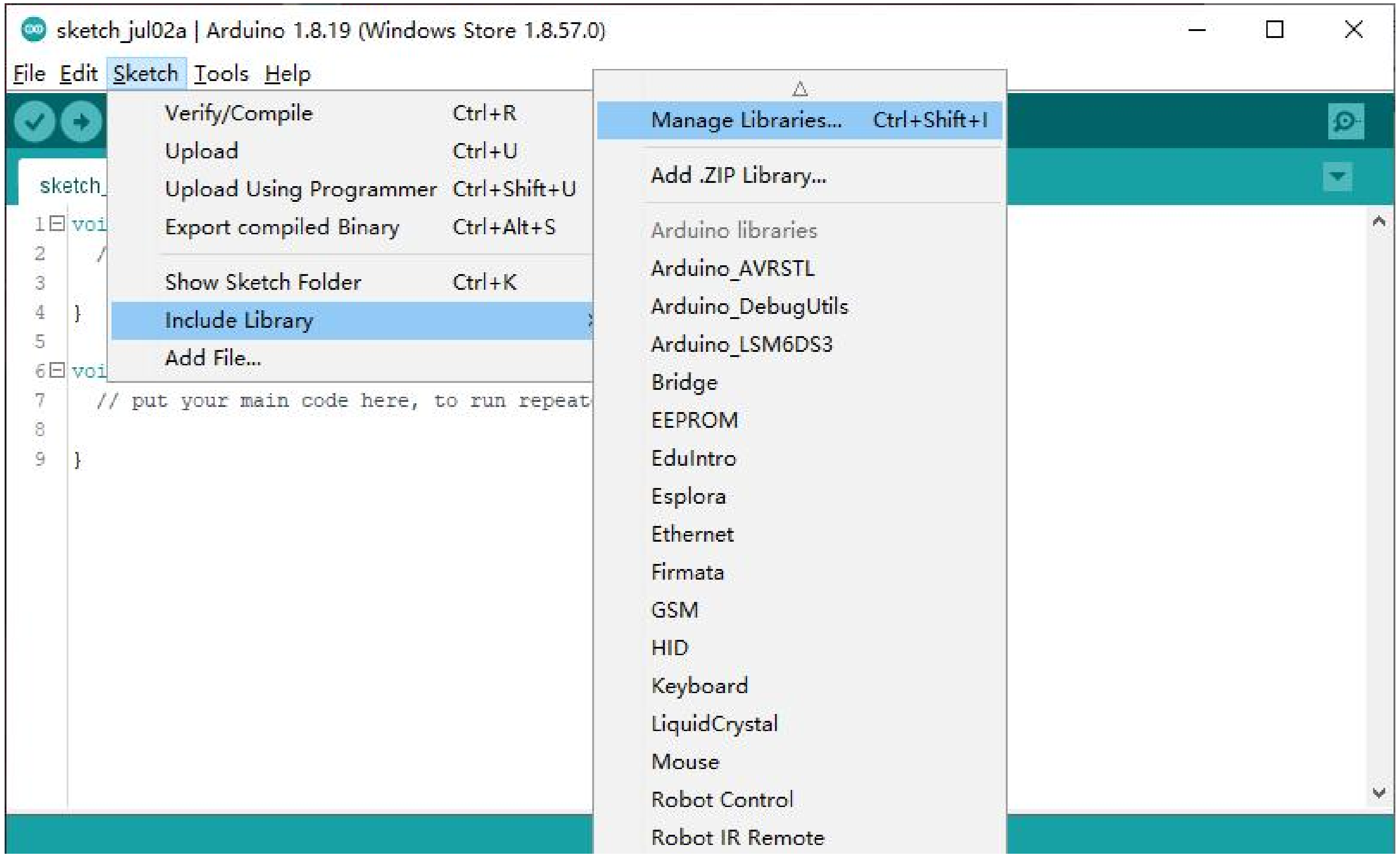
The bluetooth module	Expansion board
RXD	D13
TXD	D12
GND	GND
VCC	VCC

Because Bluetooth takes up the RX/TX port, pull out the Bluetooth before uploading the code and install it after uploading!

Add library files

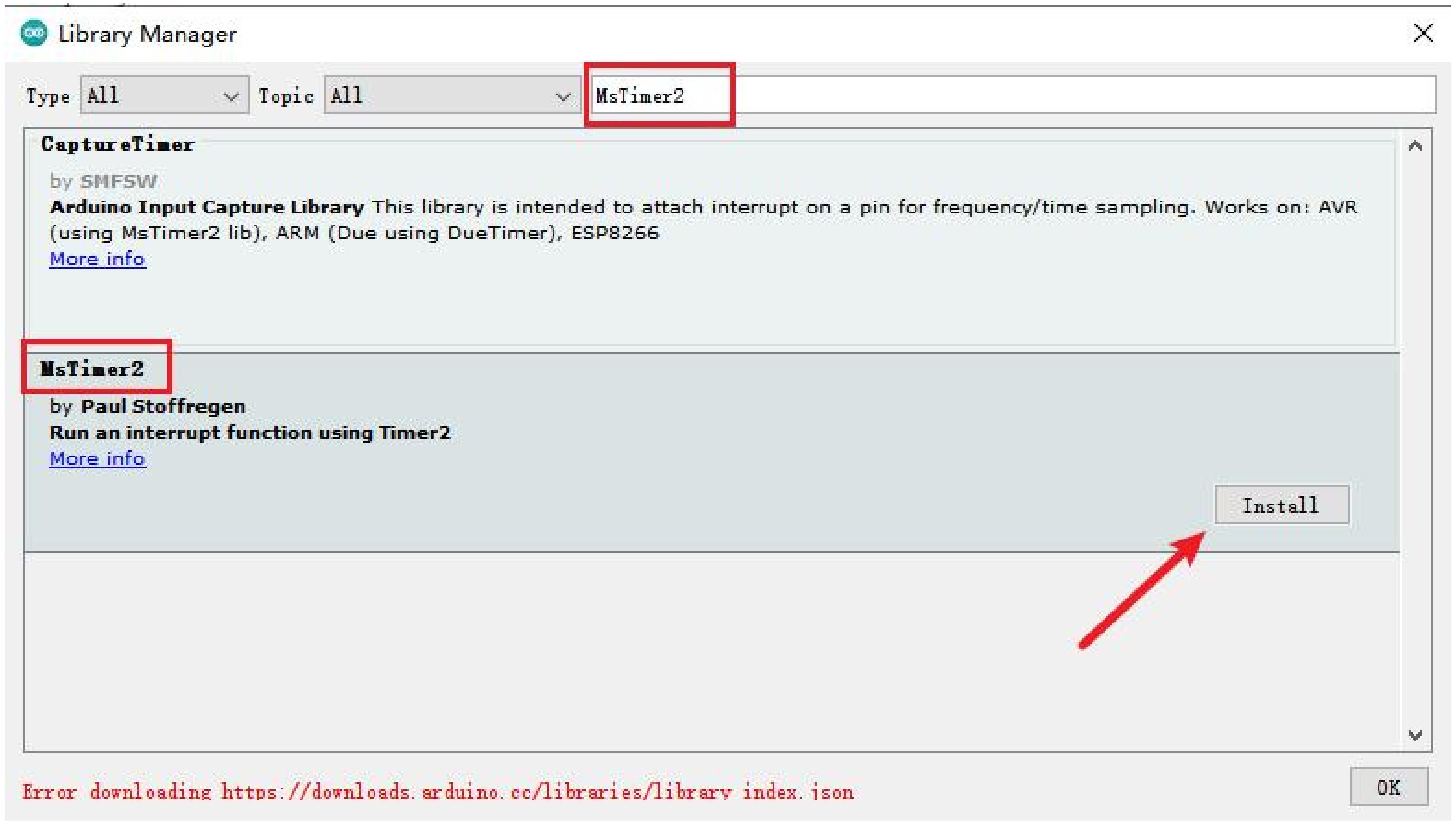
Make sure the "MsTimer2" library file is installed before uploading the code (if it has been installed in the previous lesson, please ignore this step), as follows:

In the last lesson we learned to add libraries that are already prepared locally , and now there is a way to search and download the desired library by using the library manager . Open the IDE and click the Sketch menu, then Include Library > Manage Libraries.



```
voi
  /

}

voi
  // put your main code here, to run repeat

}
```

The library manager will open and you will see a list of libraries that are installed or ready to be installed. Here, we take the installation of the MsTimer2 library as an example, and the same is true for installing other libraries. Scroll the list to find it, then select the version of the library to install, sometimes only one version of the library is available , click Install to install it .


Library Manager

Type All Topic All MsTimer2

CaptureTimer
by SMFSW
Arduino Input Capture Library This library is intended to attach interrupt on a pin for frequency/time sampling. Works on: AVR (using MsTimer2 lib), ARM (Due using DueTimer), ESP8266
More info

MsTimer2
by **Paul Stoffregen**
Run an interrupt function using Timer2
More info

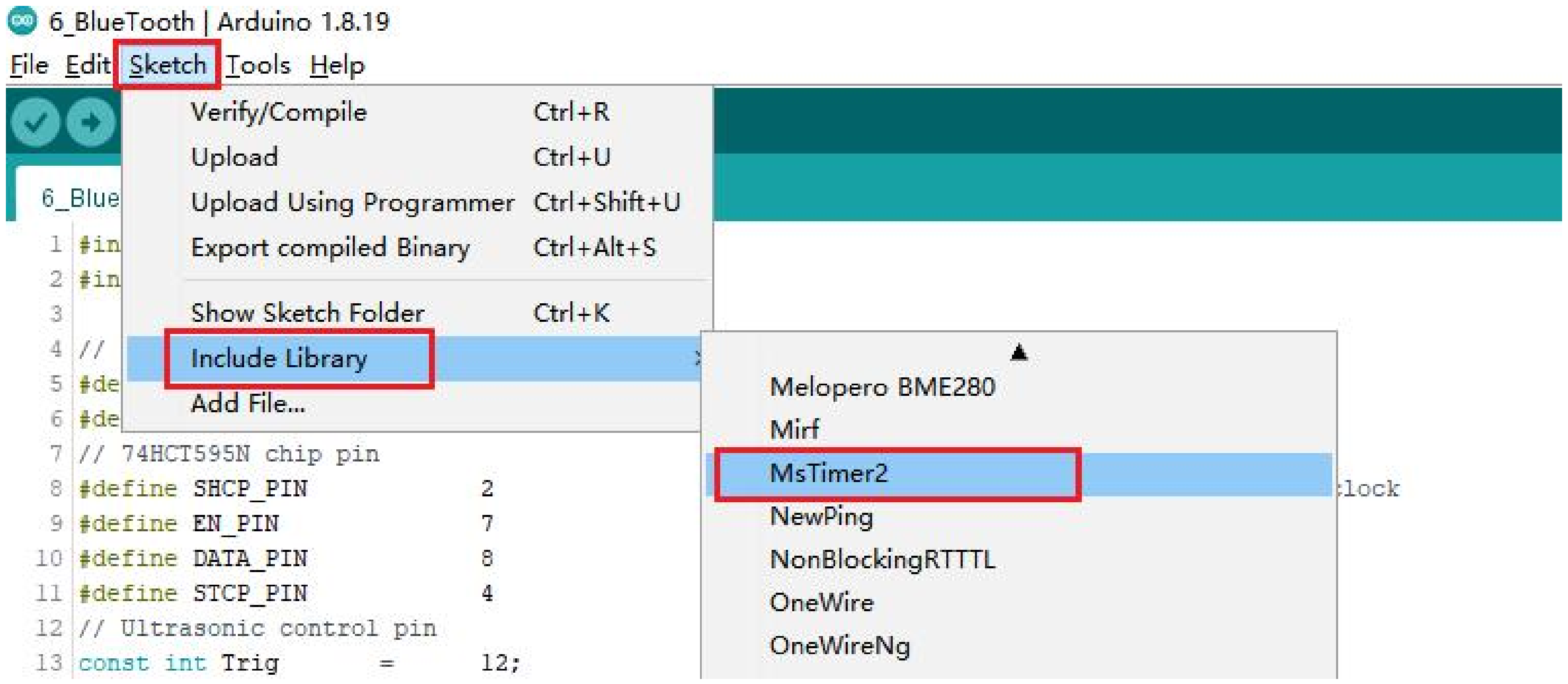
Install

Error downloading https://downloads.arduino.cc/libraries/library_index.json

OK

The download may take some time, depending on your connection speed , and you can close the library manager when finished.

You can now find new libraries available in the Sketch > Include Library menu.


6_BlueTooth | Arduino 1.8.19
File Edit Sketch Tools Help
Verify/Compile Ctrl+R
Upload Ctrl+U
Upload Using Programmer Ctrl+Shift+U
Export compiled Binary Ctrl+Alt+S
Show Sketch Folder Ctrl+K
Include Library
Add File...
Melopero BME280
Mirf
MsTimer2
NewPing
NonBlockingRTTTL
OneWire
OneWireNg

```
#in
#in

//
#de
#de
// 74HCT595N chip pin
#define SHCP_PIN          2
#define EN_PIN            7
#define DATA_PIN          8
#define STCP_PIN          4
// Ultrasonic control pin
const int Trig      =     12;
```

code analysis

Open the code file (path: 4. Tutorial_Arduino\4_Arduino Code\6_BlueTooth\6_BlueTooth.ino)

In the previous course, we have defined components such as motors and sensors. In this Bluetooth remote control program, we mainly analyze Bluetooth remote control and communication.

Define and assign bluetooth to send and receive packets

```
byte TX_package [ 5 ] = { 0xA5 , 0 , 0 , 0 , 0x5A };
byte RX_package [ 10 ] = { 0 };
```

Define the variables of the ultrasonic module and the tracking module, the initial angle of the servo motor, define the memory action and the boolean value of each automatic mode

```
int UT_distance = 0;
int Serialcount = 0;
int base_degrees = 90;
int arm_degrees = 90;
int claw_degrees = 90;
```

```
boolean menory_action_flag;
boolean Line_tracking_Function_flag = false;
boolean Avoidance_Function_flag = false;
boolean Following_Function_flag = false;
boolean Jail_Function_flag = false;
```

Define the action position and action step number of the three servo motors of the robotic arm

```
int actions_count = 0 ;
int auto_count;
int claw_read_degress [ 20 ] = { 0 , 0 , 0 };
int arm_read_degress [ 20 ] = { 0 , 0 , 0 };
int base_read_degress [ 20 ] = { 0 , 0 , 0 };
```

Define free remote control mode and intelligent automation mode

```

typedef struct
{
byte mode1; // Bit0: free mode;Bit1: tracking mode;Bit2: Obstacle avoidance mode;Bit3: Follow mode;
// Bit4: Dungeon Mode;Bit5: Save button;Bit6: Automatic button;Bit7: empty
byte mode2; // Bit0: fluke;Bit1: closed claw;Bit2: clockwise rotation;Bit3: reverse;
```

```
    char x_axis = 0;        // Store variables on the X axis
    char y_axis = 0;        // Store the variables on the Y axis
    byte C_speed = 127;     // Speed of storage cart
    char x_Base = 0;        // Store the steering gear on the X axis
    char y_Arm = 0;         // Store the steering gear on the Y-axis
}rx_buffer;

rx_buffer RX_buffer;
```

Initialize Timer2Isr() and setup()

```
void Timer2Isr ()
{
    sei ();
UT_distance = SR04 ( Trig, Echo);
}
void setup ()
{
    pinMode (SHCP_PIN, OUTPUT);
    pinMode (EN_PIN, OUTPUT);
```

Repeat function

```
void loop ()
```

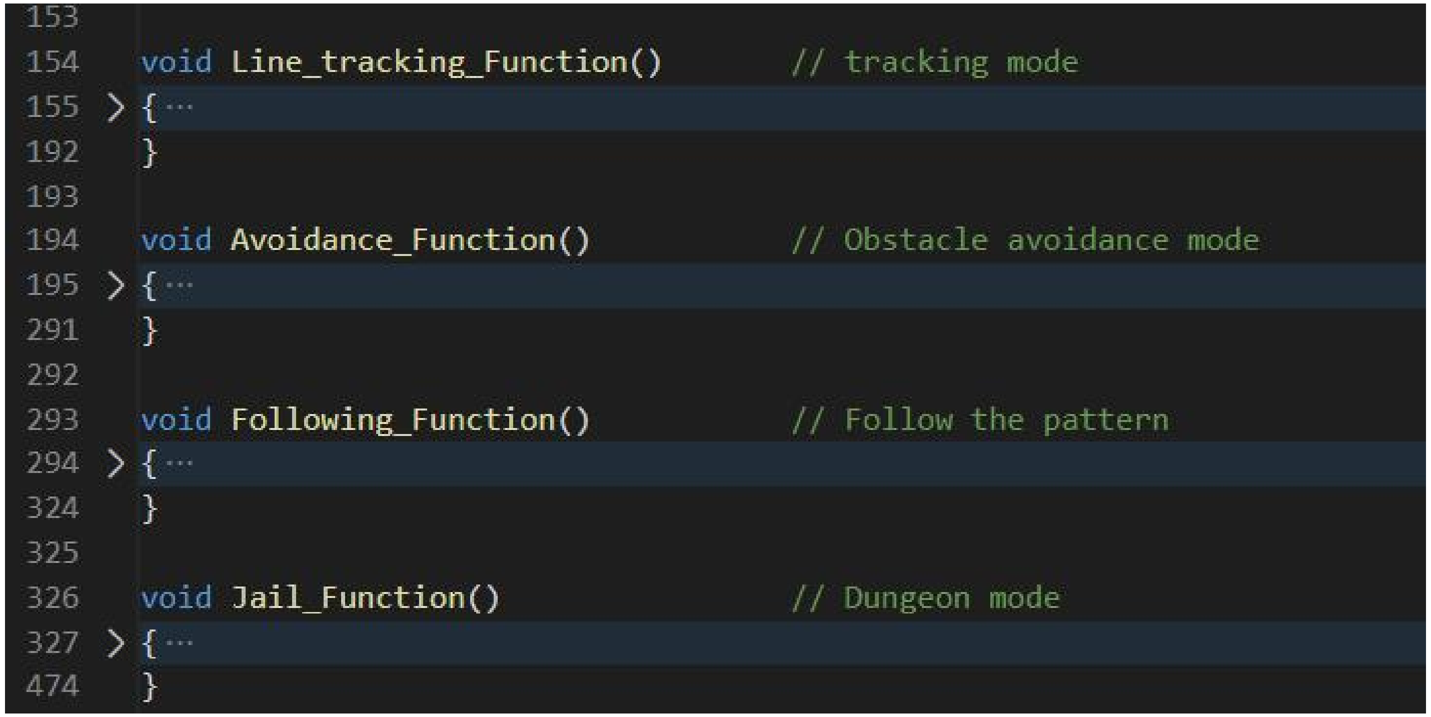
Communication between Bluetooth remote control APP and smart car

```
    TX_Information (UT_distance, auto_count); // Send ultrasonic data
    RX_Information (); // Receiving Bluetooth data
```

After the received remote control commands are analyzed, different function modes are executed, such as tracking, obstacle avoidance, dungeon, etc.

```
switch ( RX_buffer.mode1 ) _ _
```

4 different functional modes



```
void Line_tracking_Function()       // tracking mode
{…
}

void Avoidance_Function()           // Obstacle avoidance mode
{…
}

void Following_Function()           // Follow the pattern
{…
}

void Jail_Function()                // Dungeon mode
{…
}
```

The action memory function function auto_doit( ) initializes the variable actions_count of the recording step to zero, and then starts recording the steps of the automation program

```
void auto_doit () // Automatic mode
{
    if ( 0 != auto_count )
{
menory_action_flag = true ;
}
actions_count = 0 ;
```

Free control mode function, according to the information sent by APP and saved in RX_buffer to control movement in all directions and each function button to control the action of the manipulator

```
void free_mode () // free mode
{
    if(RX_buffer.x_axis >= -30 && RX_buffer.x_axis <= 30 && RX_buffer.y_axis >= 30)     //Forward
    {
        Motor(Forward, RX_buffer.C_speed);
        delay(5);
    }
```

Read the saved action step records, the number of records is less than or equal to 19

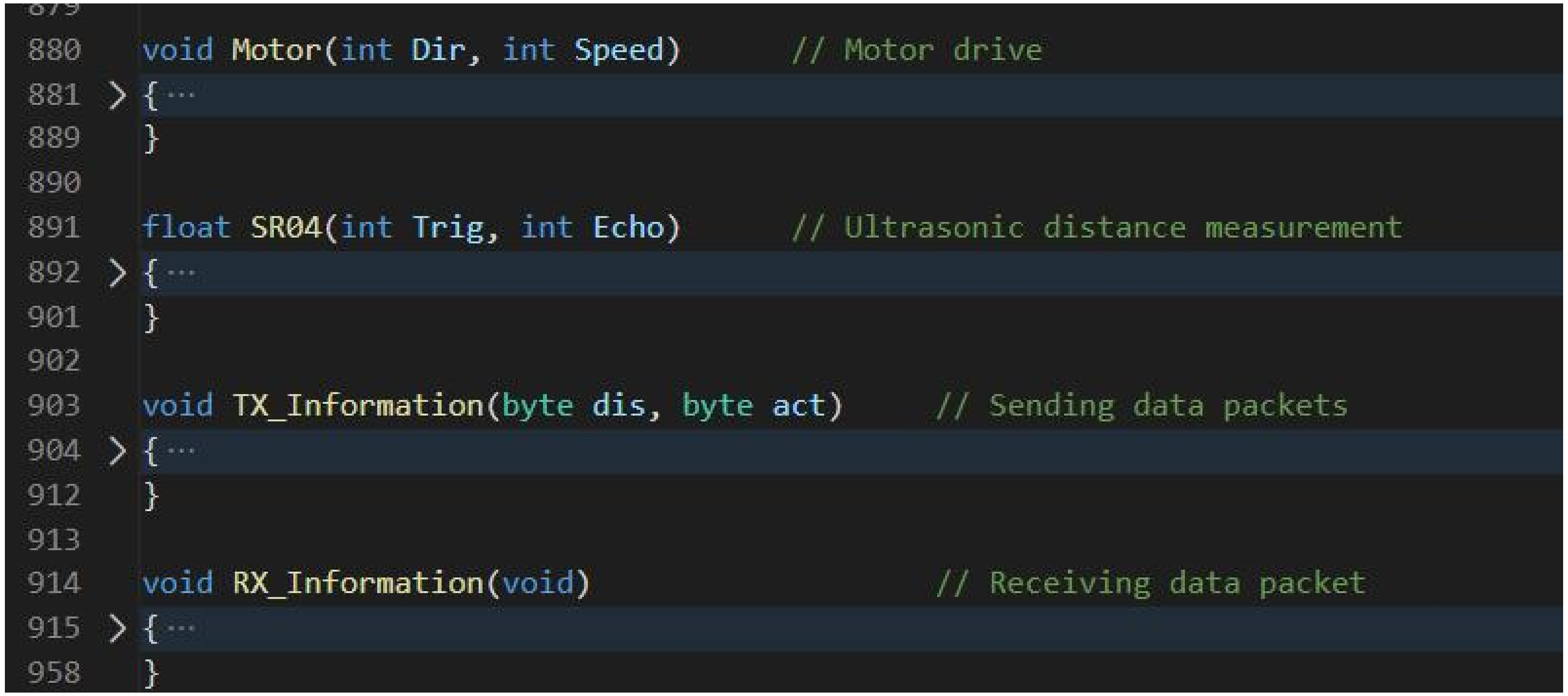
```
void read_degress ()
{
    if (actions_count <= 19 )
{
        claw_read_degress [( int )((actions_count + 1 ) - 1 )] = clawservo .read ( ) ;
        delay ( 50 );
        RX_Information ();
```

Execute the previously recorded action function auto_do( ), initialize the variable actions_count of the recorded step to zero,

and then repeat the automated program steps

```
void auto_do ()
{
    if ( 0 != auto_count )
{
menory_action_flag = true ;
}
actions_count = 0 ;
```

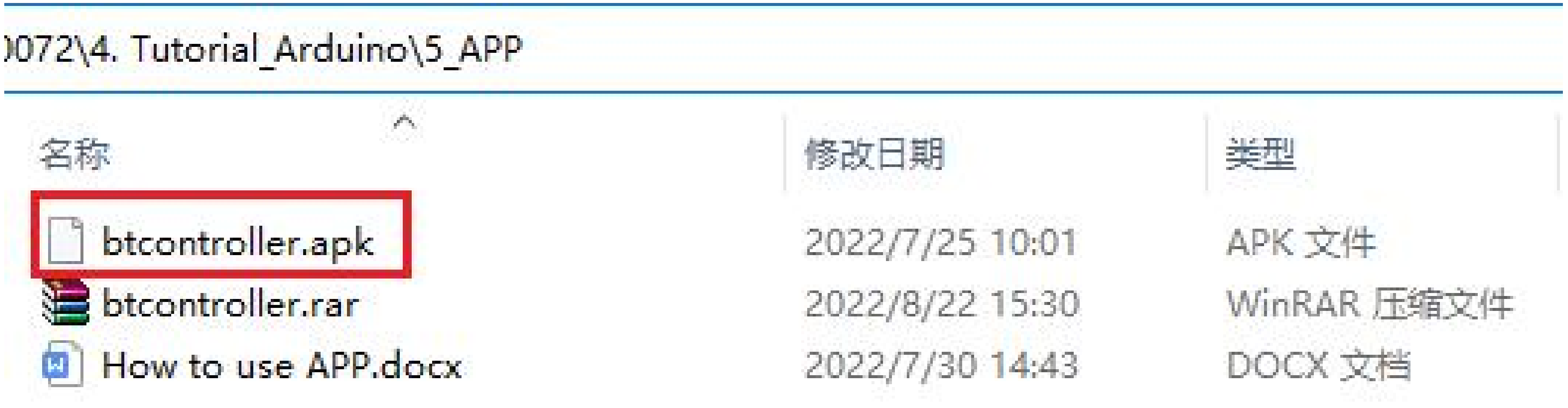
Like the infrared remote control in the previous section, it also requires a motor drive, an ultrasonic module and a Bluetooth send and receive function



```
void Motor(int Dir, int Speed)      // Motor drive
{···
}

float SR04(int Trig, int Echo)      // Ultrasonic distance measurement
{···
}

void TX_Information(byte dis, byte act)     // Sending data packets
{···
}

void RX_Information(void)                   // Receiving data packet
{···
}
```

Install and set up the app

Open the directory 4. Tutorial_Arduino\5_APP and install " btcontroller .apk " to the Android phone

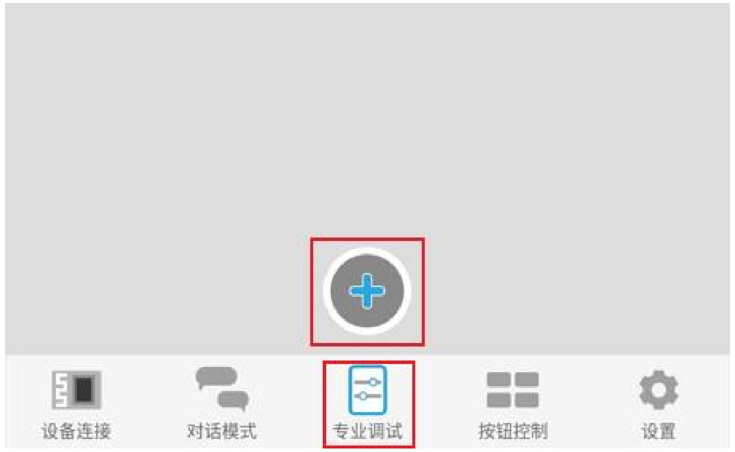


)072\4. Tutorial_Arduino\5_APP

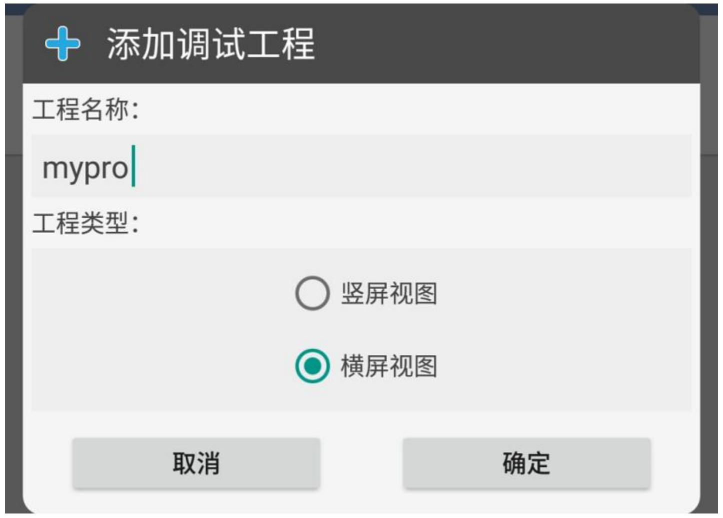
名称	修改日期	类型
btcontroller.apk	2022/7/25 10:01	APK 文件
btcontroller.rar	2022/8/22 15:30	WinRAR 压缩文件
How to use APP.docx	2022/7/30 14:43	DOCX 文档

Next , we use an Android phone to demonstrate how to control the ZHIYI intelligent robotic arm car through this application:

Enter the professional debugging interface, click the add button "+"



You need to fill in the project name of the project name.


添加调试工程

工程名称：

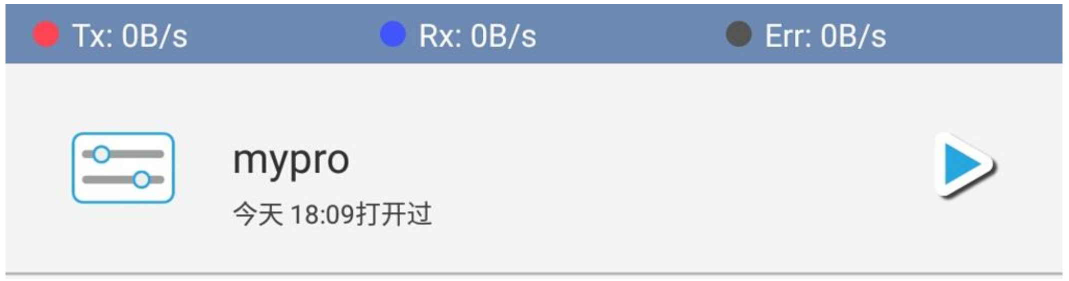
mypro

工程类型：

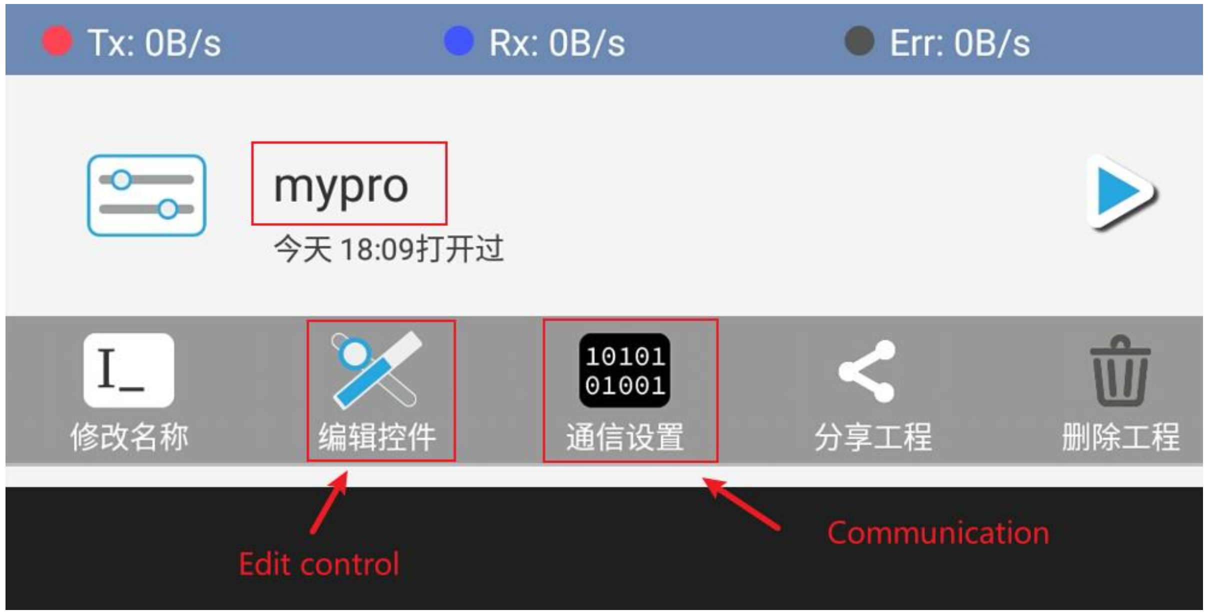
竖屏视图

[✓ 横屏视图]

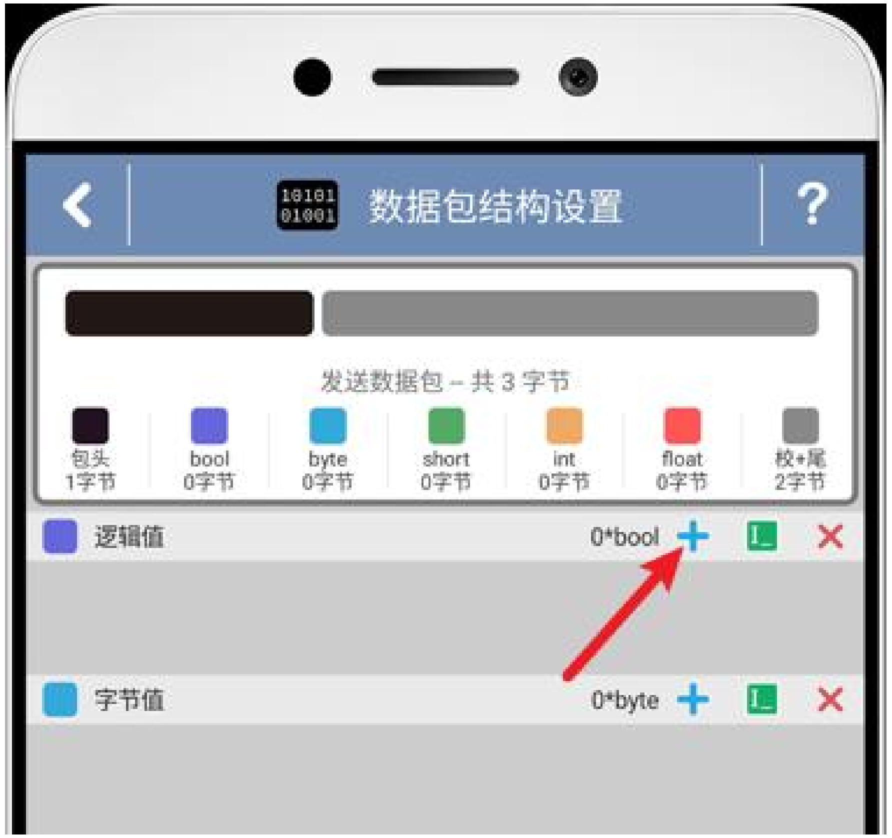
Click OK to see the built project

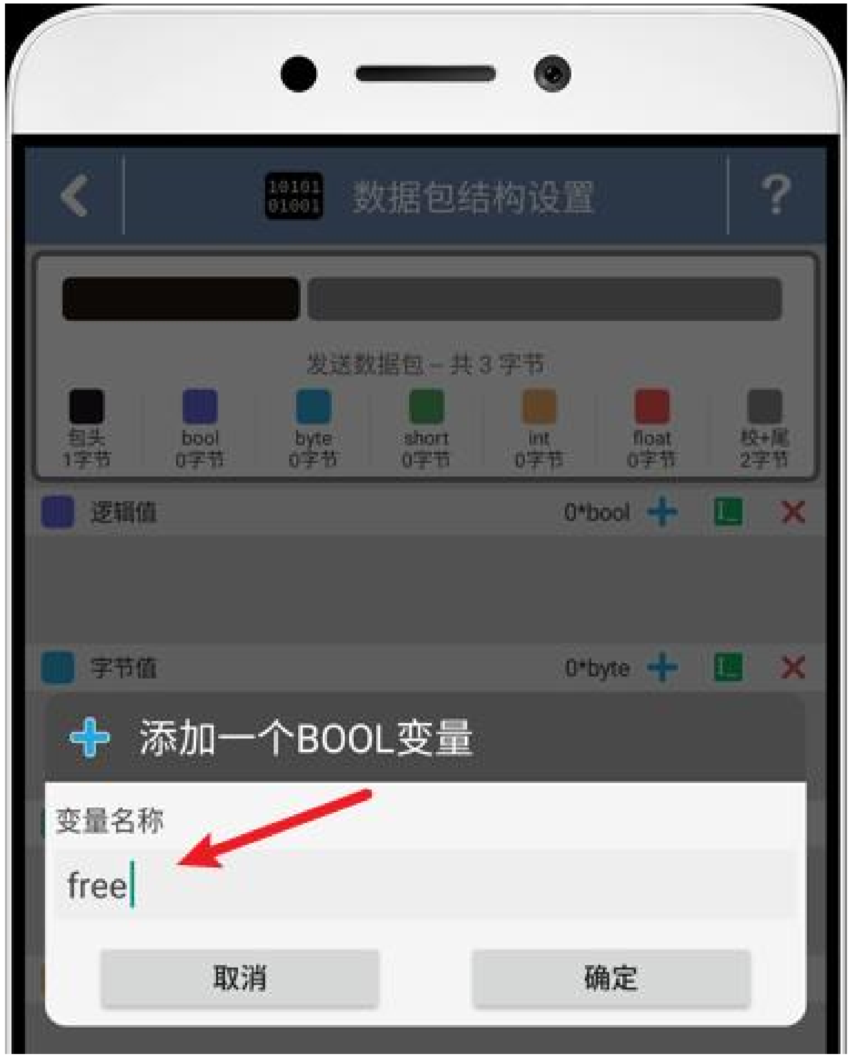


Click on the project name and the option to modify the project will appear



First configure the communication settings , click the "+" sign, add a boolean value , and enter the boolean value name.

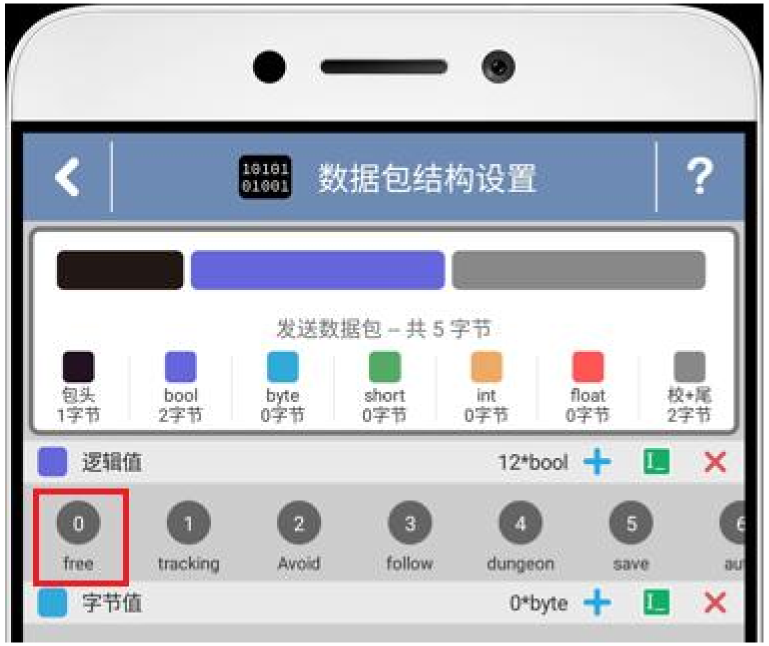



10101
01001
数据包结构设置
?
发送数据包 – 共 3 字节
包头
1字节
bool
0字节
byte
0字节
short
0字节
int
0字节
float
0字节
校+尾
2字节
逻辑值
0*bool
字节值
0*byte
添加一个BOOL变量
取消
确定

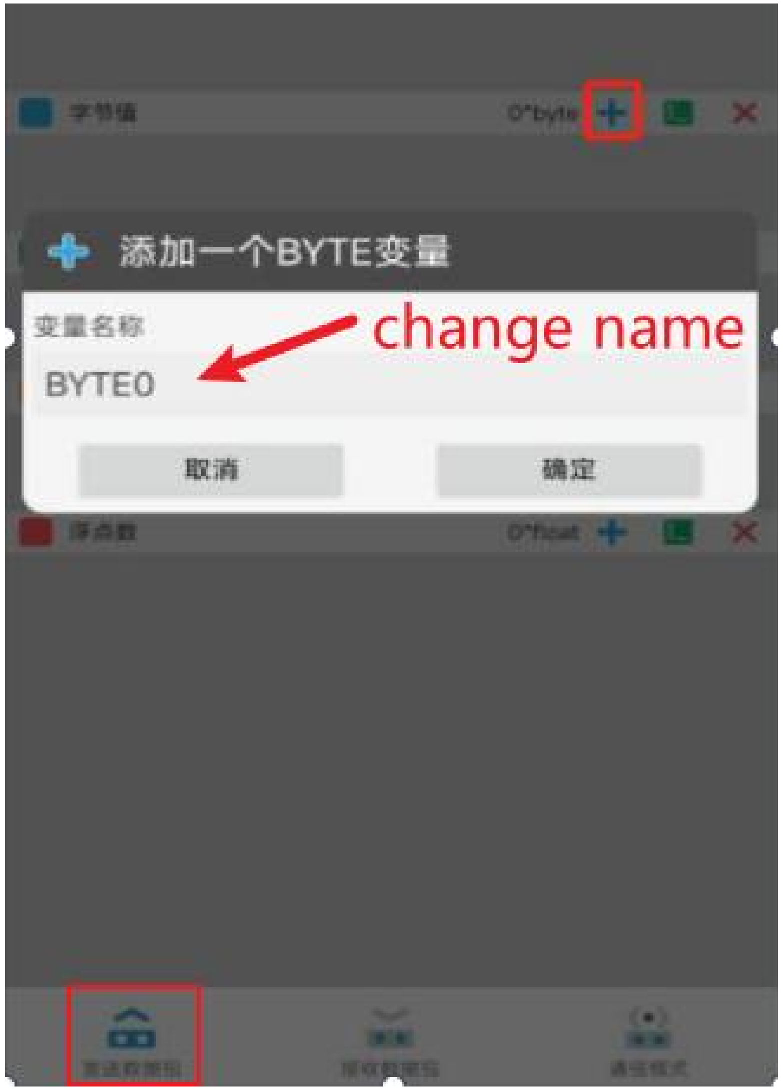
变量名称

free

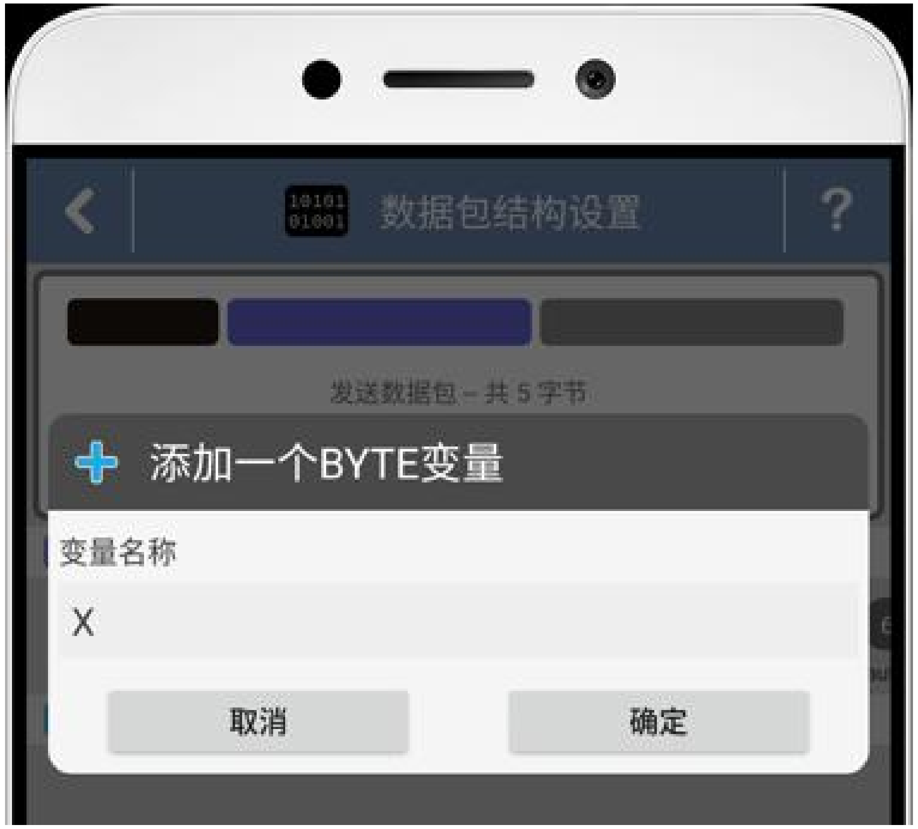
You can see that the created Boolean values will be arranged and displayed in this column, and 11 Boolean value variables will be created in the same way. Tracking, Avoid, follow, dungeon, save, auto, empty, claws_open, claws_closed, count-clockwise, clockwise



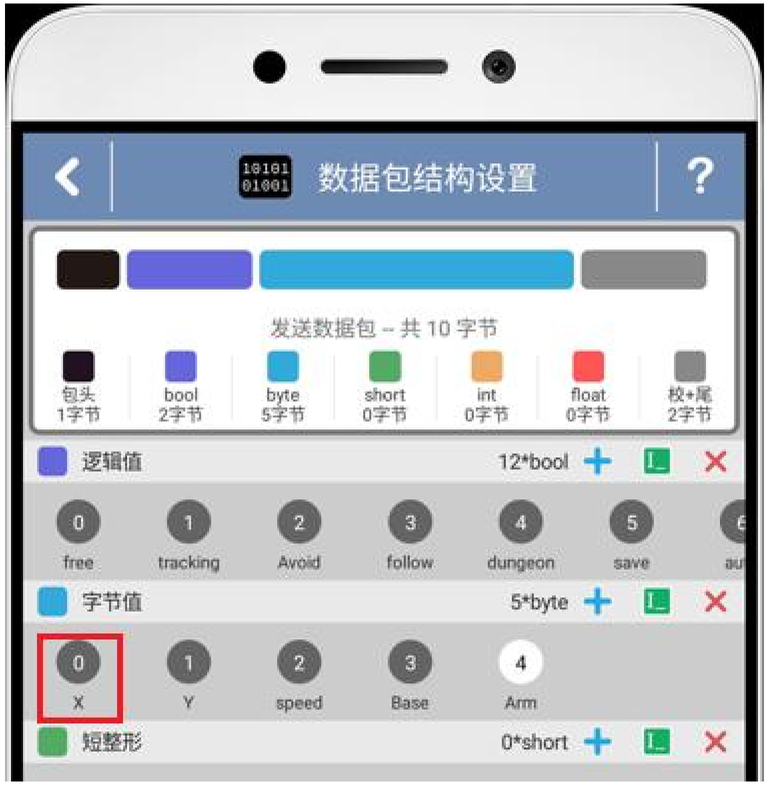
Click the "+" sign in the byte value column, add the byte value "byte" , and enter the byte value name


字节值
0*byte
添加一个BYTE变量
变量名称
change name
BYTE0
取消
确定
浮点数
0*float

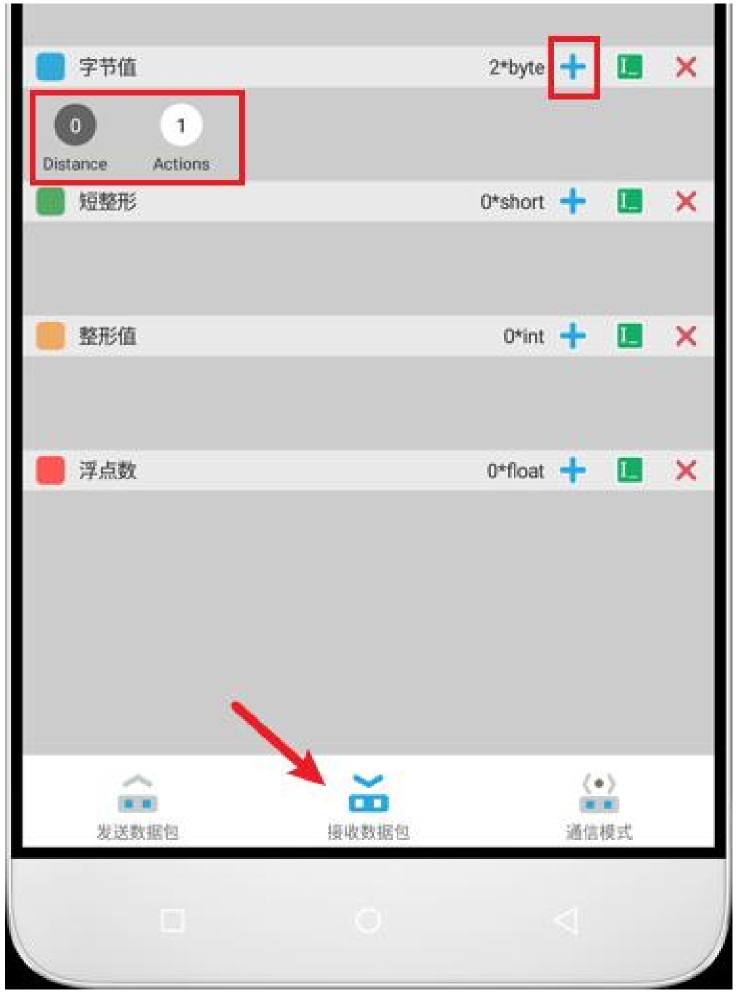
Take setting a variable X that controls the direction of left and right movement as an example, change the name to X



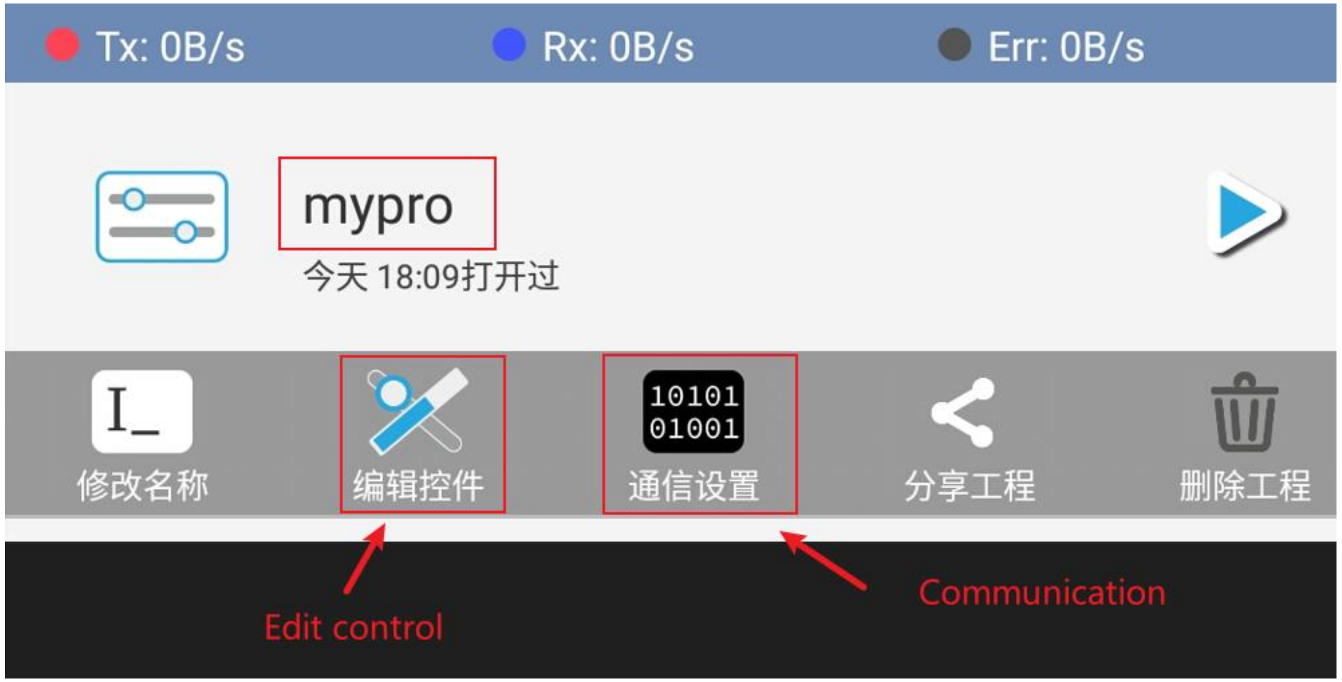
You can see that the created byte values will be arranged and displayed in this column, and use the same method to create 4 byte value variables. respectively Y, speed, Base, Arm



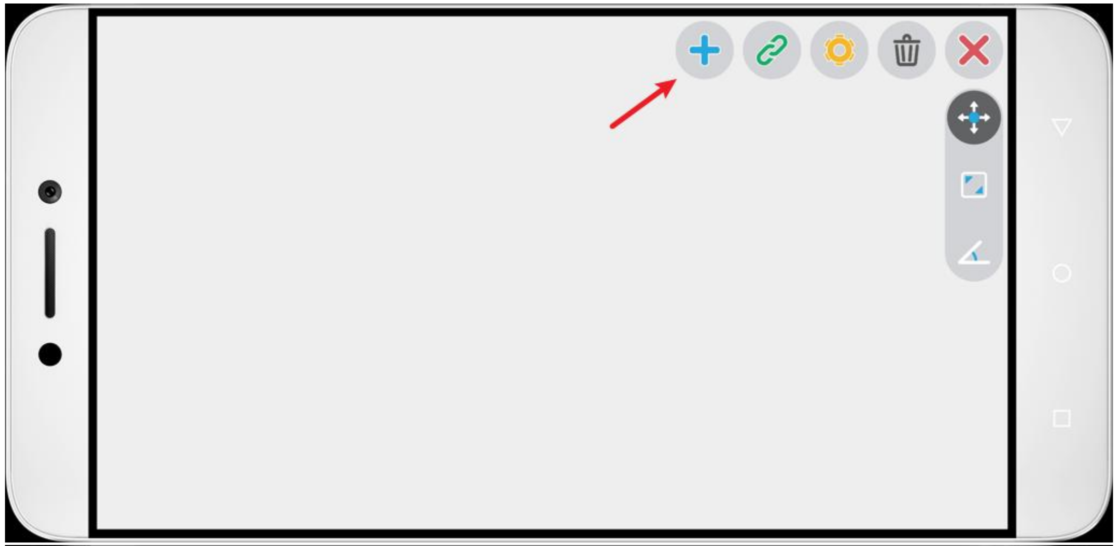
Click the second option at the bottom to create two received byte values, namely: Distance, Actions

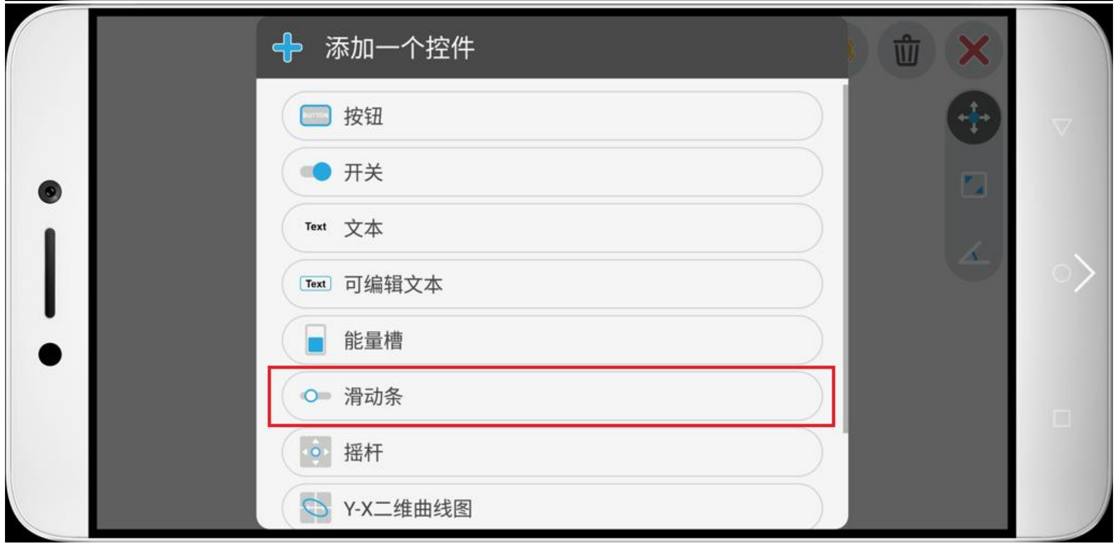


After all the above are added, return to the layout " Edit Controls " to add controls .

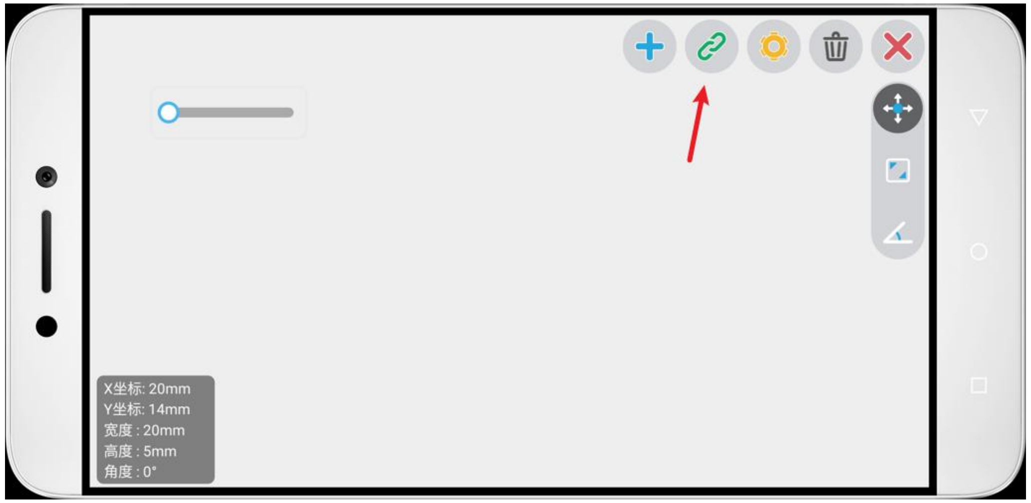


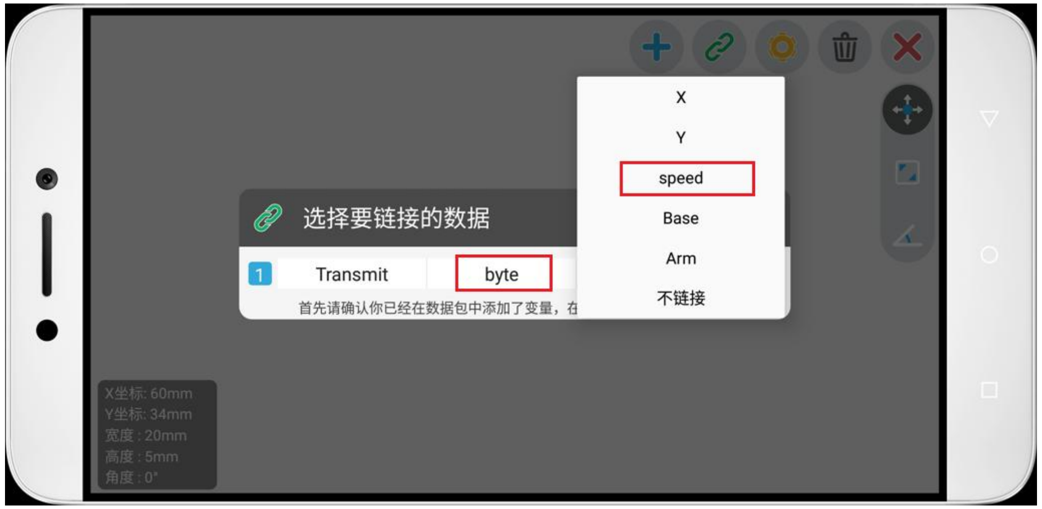
Click the "+" sign in the upper right corner to create a new control, taking the slider control for speed control as an example;



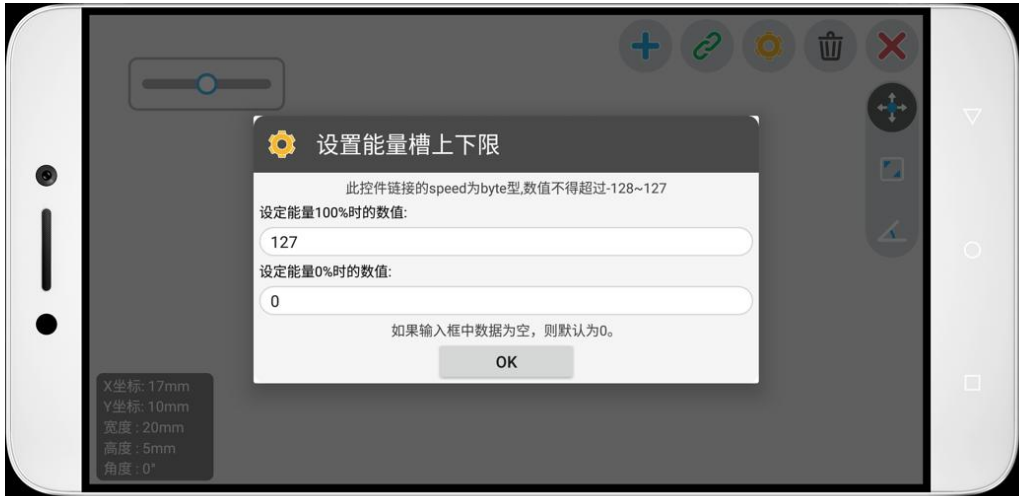

添加一个控件
按钮
开关
Text 文本
Text 可编辑文本
能量槽
滑动条
摇杆
Y-X二维曲线图

Select the data type to be connected and the variable value just set, byte and speed





Then set the upper and lower limits, click OK to complete;



设置能量槽上下限

此控件链接的speed为byte型,数值不得超过-128~127

设定能量100%时的数值:

127

设定能量0%时的数值:

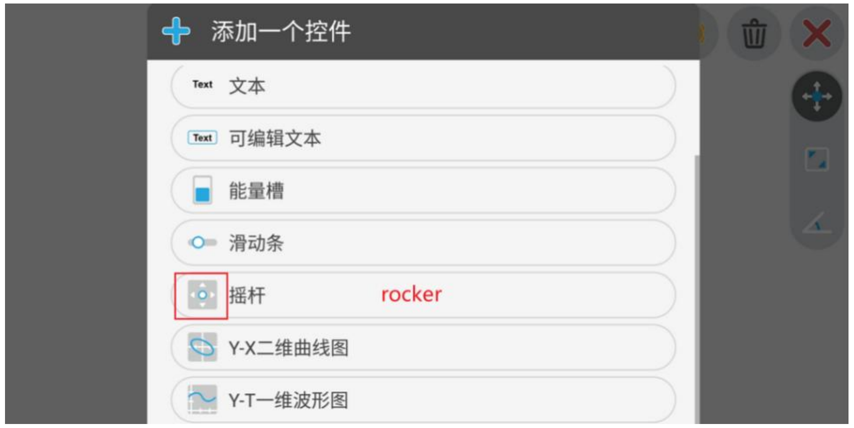
0

如果输入框中数据为空，则默认为0。

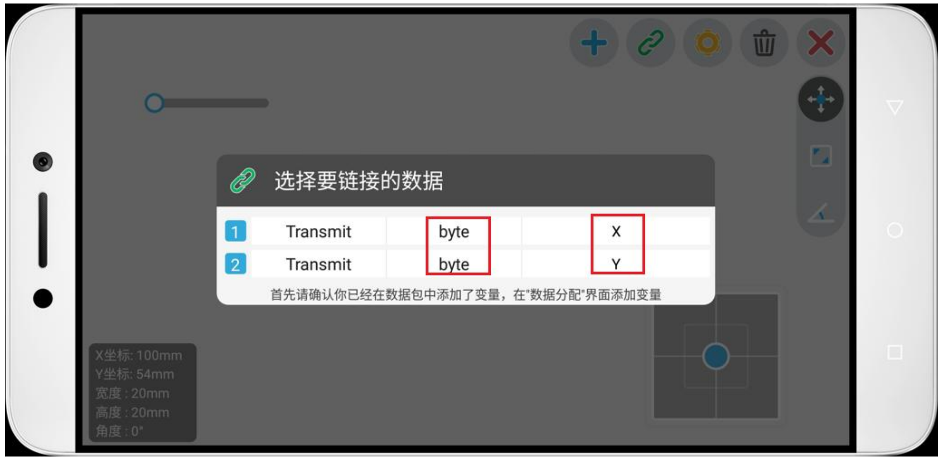
OK

X坐标: 17mm
Y坐标: 10mm
宽度 : 20mm
高度 : 5mm
角度 : 0°

Add a joystick control to control the direction of the car:



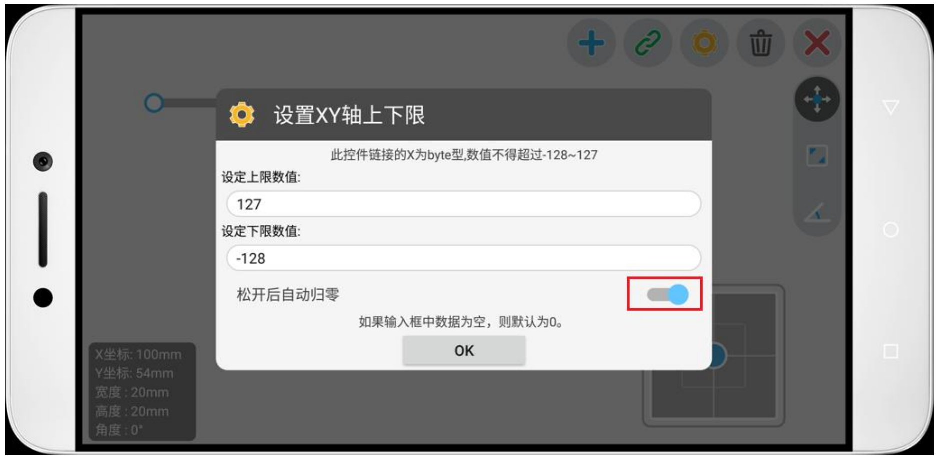
Select the connection data type byte and the variable name X/Y , then check Release Auto Reset :


选择要链接的数据

1	Transmit	byte	X
2	Transmit	byte	Y

首先请确认你已经在数据包中添加了变量，在"数据分配"界面添加变量

X坐标: 100mm
Y坐标: 54mm
宽度 : 20mm
高度 : 20mm
角度 : 0°


设置XY轴上下限

此控件链接的X为byte型,数值不得超过-128~127

设定上限数值: 127

设定下限数值: -128

松开后自动归零 [✓]

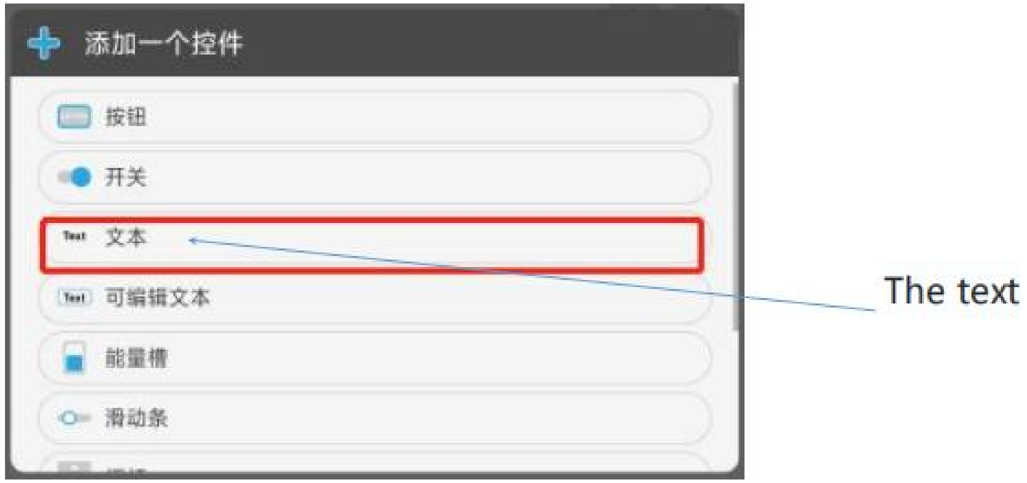
如果输入框中数据为空，则默认为0。

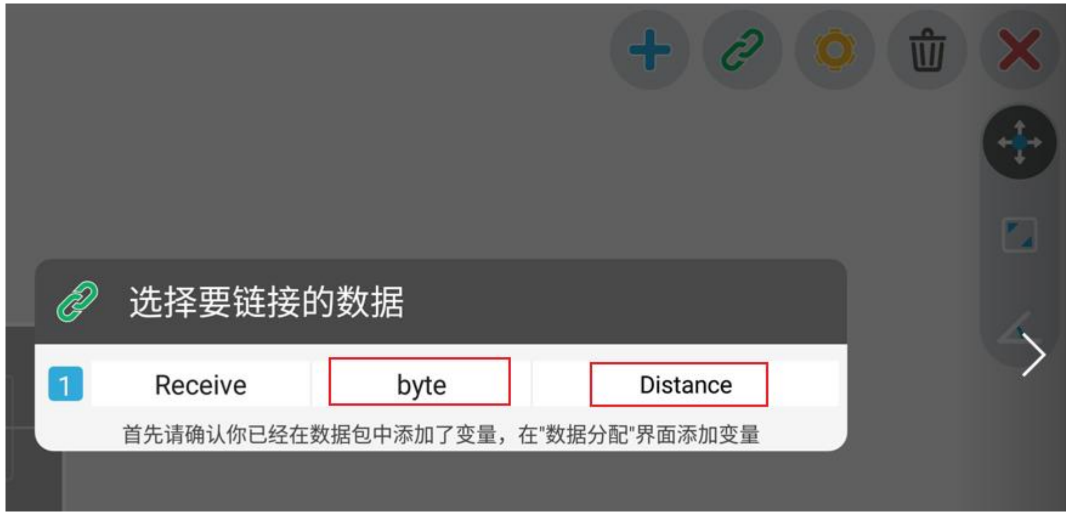
OK

X坐标: 100mm
Y坐标: 54mm
宽度 : 20mm
高度 : 20mm
角度 : 0°

Add another same joystick control to control the opening and closing of the paw, the variables are Base and Arm

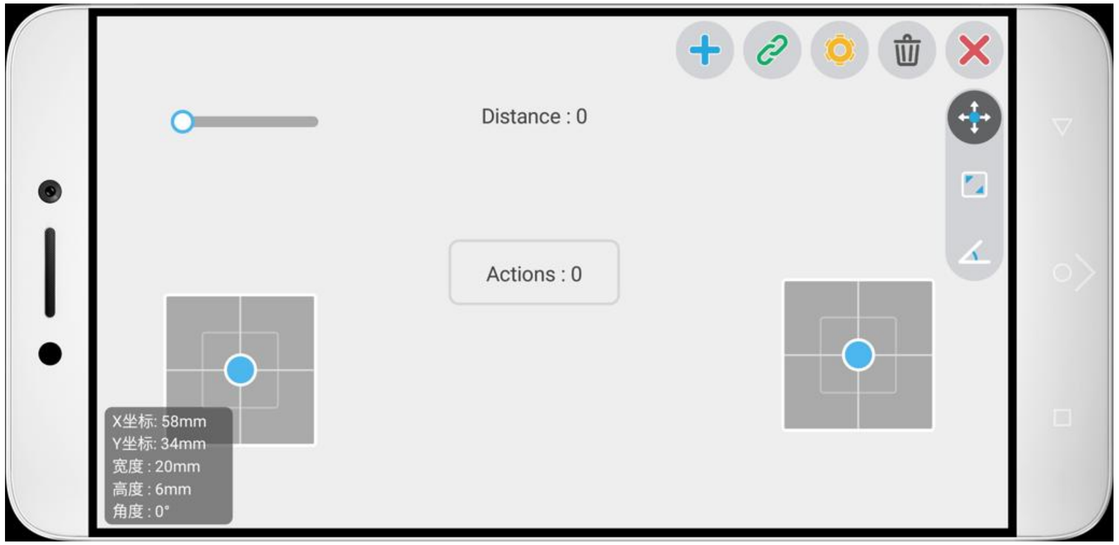
Add a text control to display the ultrasonic distance :



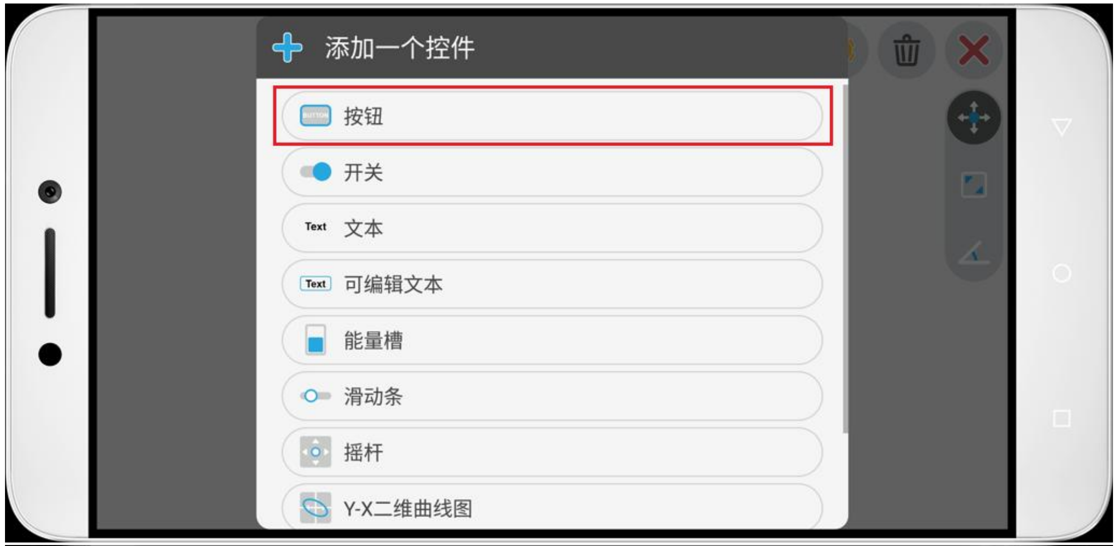


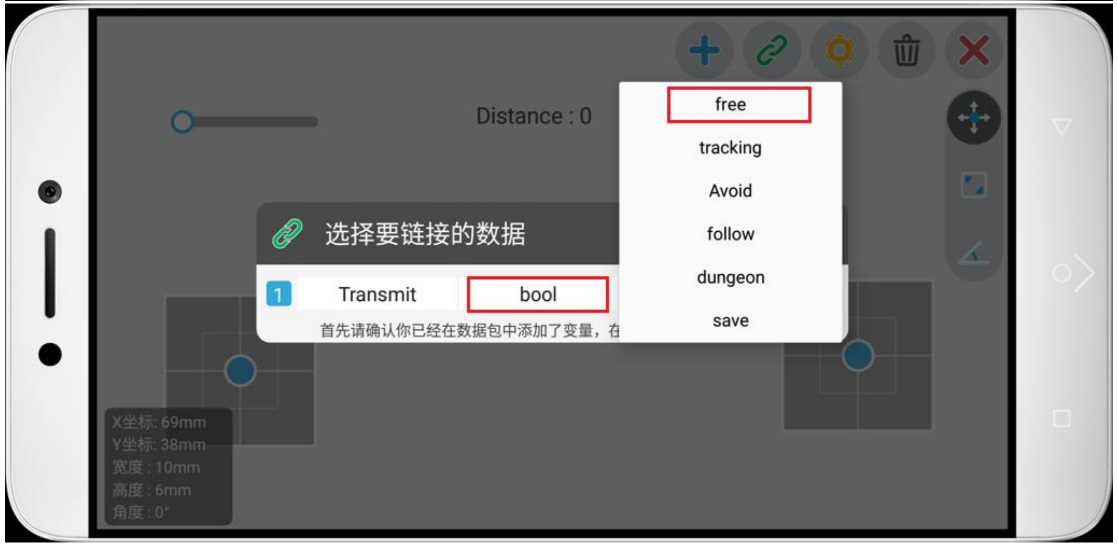
首先请确认你已经在数据包中添加了变量，在"数据分配"界面添加变量

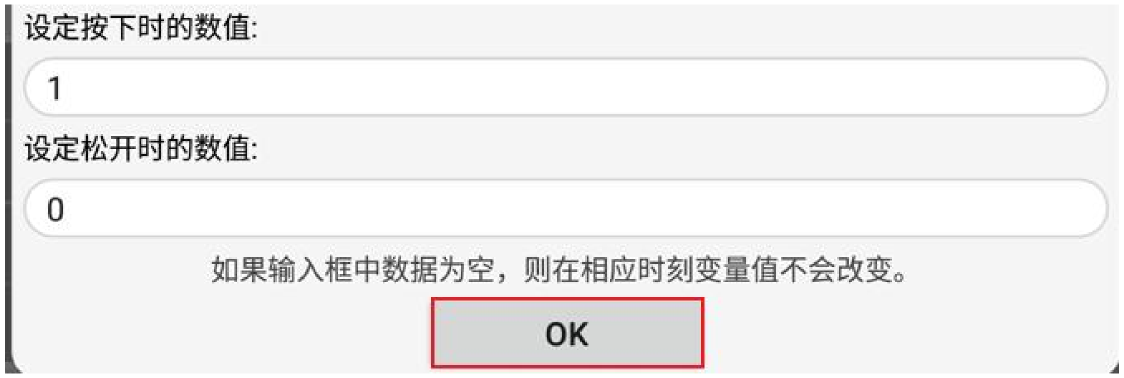
Add Actions in the same way;



Next, add a control button and click the green button in the upper right corner to link the data:


添加一个控件
按钮
开关
Text 文本
Text 可编辑文本
能量槽
滑动条
摇杆
Y-X二维曲线图


Distance : 0
free
tracking
Avoid
follow
dungeon
save
选择要链接的数据
1 Transmit
bool
首先请确认你已经在数据包中添加了变量，在
X坐标: 69mm
Y坐标: 38mm
宽度 : 10mm
高度 : 6mm
角度 : 0°



设定按下时的数值:

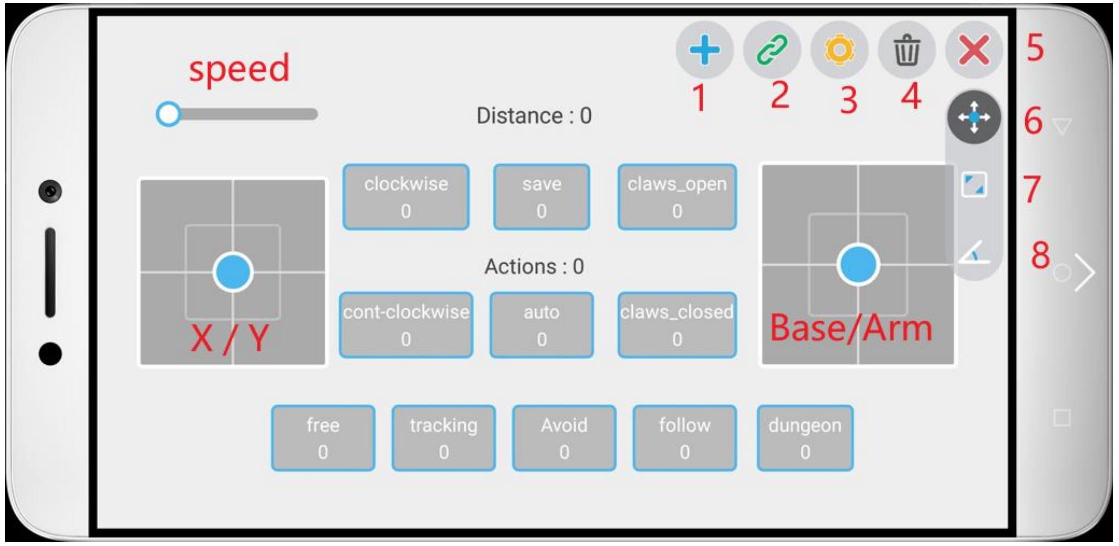
1

设定松开时的数值:

0

如果输入框中数据为空，则在相应时刻变量值不会改变。

Add 11 more controls in the same way : tracking , Avoid, follow, dungeon, save, auto, empty, claws_open, claws_closed, count-clockwise, clockwise. Please drag and place these controls in your own way.



The functions of each icon button in the upper right corner of the screen:

1 Add various controls,

2 Connection variables

3 means setting control parameters

4 is to delete the control

5 Exit the current control layout

6 Mobile controls

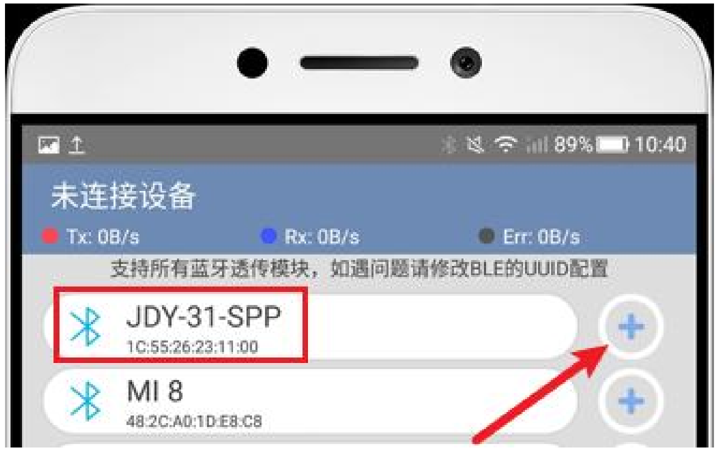
7 Zoom in and zoom out controls

8 control rotation angle

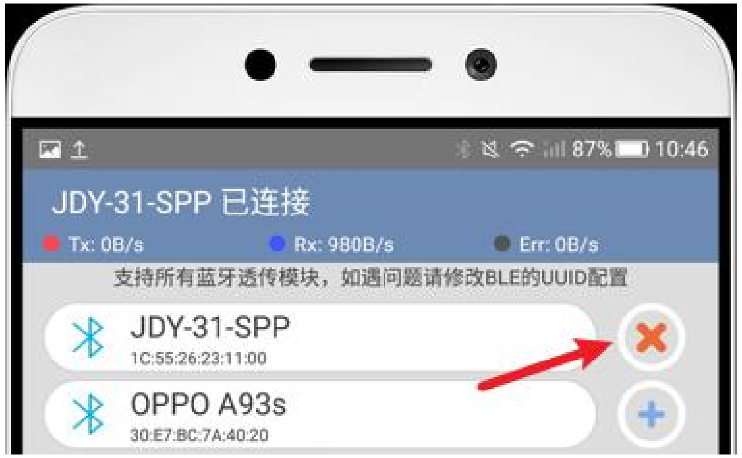
When the controls are all adjusted, start connecting to Bluetooth, go to Device Connections and click Search.

Find " JDY-31-SPP " and click "Add Device". If a password is required, enter the password as 1234 or 0000. (If you find that connecting to Bluetooth is slow in future use, first "forget" the Bluetooth in the phone settings. Then search for and connect to Bluetooth in the APP.)

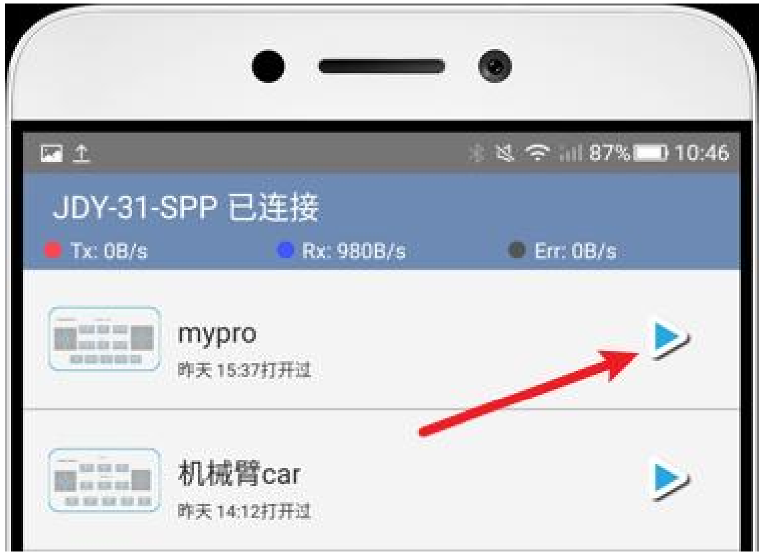
Click "+" again



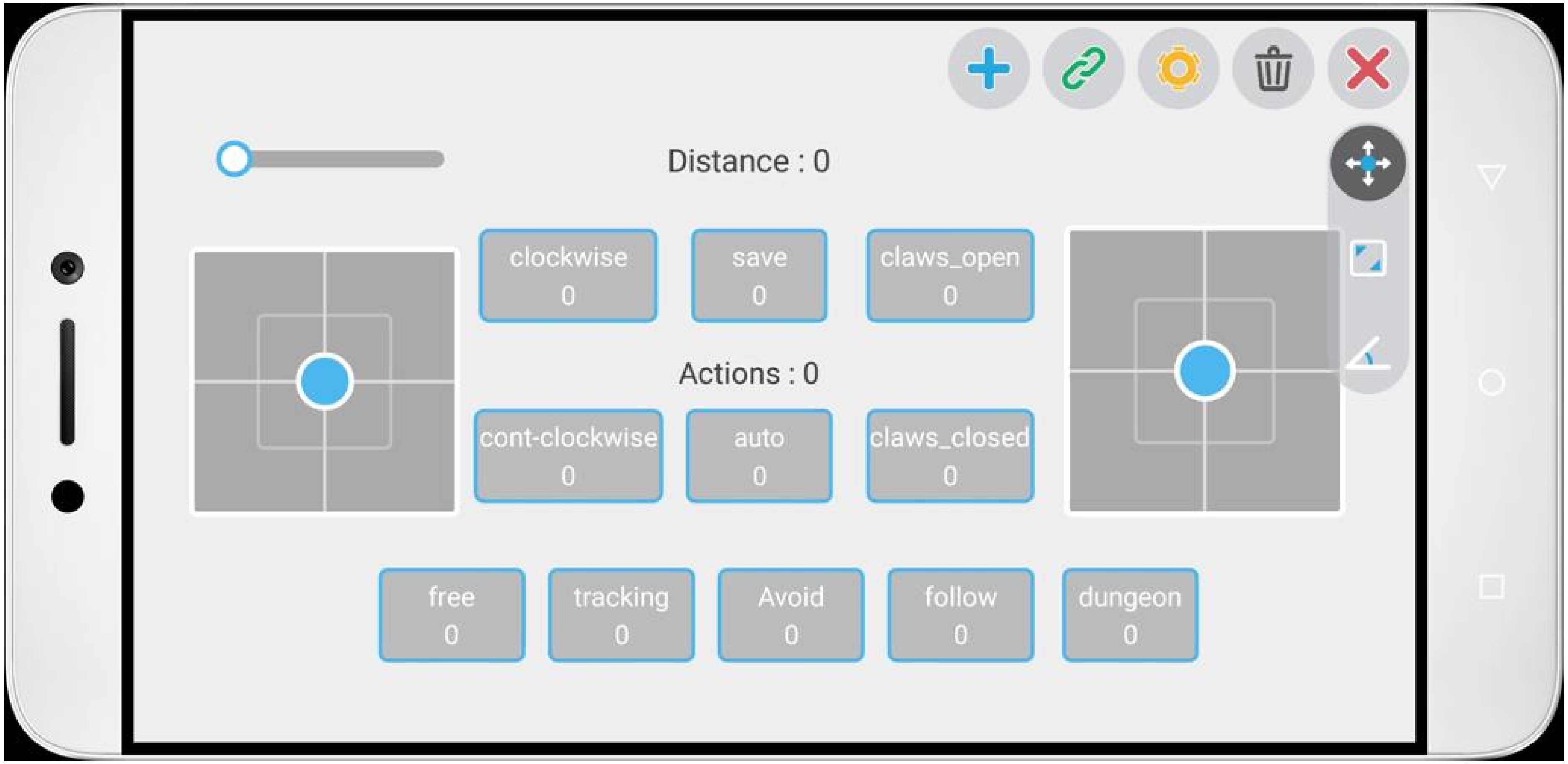
When a red "x" appears, it means the Bluetooth connection is successful :



After the connection is successful, click Start and operate the car :



The final operation interface is as follows



Notice:

In free mode, the ultrasonic detection distance can be displayed in real time, and other modes are not displayed in real time;

Other modes are automatic execution, and if you want to stop, you need to switch back to free mode;

During operation, pay attention to observe that the servo motor cannot be left in the unfinished state for a long time to prevent heat generation and damage;

Memory function operation: (every time a servo motor is remotely controlled, one action must be saved, and a maximum of 19 steps can be recorded)

The initial robot arm state is remotely controlled to state 1, and the save button is pressed to record 1;

In the remote control from state 1 to state 2, press the save button to record 2;

Remote control from state 2 to state 3, and then press the save button to record 3;

In this way, to the end action, it should be noted that the end action should be consistent with the initial action, so that a continuous and consistent action can be maintained when starting the automation execution.