

# Kit de robot 4WD ZHIYI ZYC0076 ESP32 cu roata omnidirectionala mecanum

Info aici: <https://www.bitmi.ro/kit-de-robot-4wd-zhiyi-zyc0076-esp32-10005.html>

## CONTENT

Lesson 1 Install Arduino IDE .....	5
Introducing Arduino .....	5
Arduino Board .....	5
Arduino software .....	6
Install Arduino (Mac OS X) .....	14
Install Arduino (Linux) .....	14
Lesson 2 Adding "Libraries" to the Arduino IDE .....	15
How to Install Additional Libraries in Arduino IDE 1 .....	15
What are Libraries? .....	15
Method 1: Import the .zip library .....	18
Method 2: (This method requires networking) .....	19
Lesson 3 The first program code - Blink .....	22
About this lesson .....	22
Main control board .....	22
Make your own "Blink" sketches .....	23
Lesson 4 Expansion Board and Motor Driver .....	32
Course Introduction .....	32
Expansion board introduction .....	32
motor driven .....	33
code analysis .....	41
Lesson 5 Servo Motors .....	46
Course Introduction .....	46
Introduction of steering gear .....	46
Wiring diagram .....	47
code analysis .....	48

Lesson 6 Ultrasonic Ranging .....	49
Course Introduction .....	49
Ultrasonic sensor .....	49
Ultrasonic ranging is a non-contact detection method .....	50
code analysis.....	53
Lesson 7 Obstacle Avoidance Car .....	56
Course Introduction .....	56
Obstacle Avoidance Principle.....	56
code analysis.....	57
Lesson 8 Tracking Car .....	60
Course Introduction .....	60
tracking module .....	60
Wiring diagram.....	62
Three-way tracking principle.....	66
code analysis.....	69
Lesson 9 Integrated Functions and Remote Control.....	72
Course Introduction .....	72
Introduction to ESP32-CAM .....	72
code analysis.....	80
functional combination .....	90
code analysis.....	91
Mode 1 (model1_func) is free control .....	91
Mode 2 (model2_func) is the obstacle avoidance mode .....	93
Mode 3 (model3_func) for car following .....	95
Mode 4 (model4_func) for car tracking.....	97

# 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:



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.



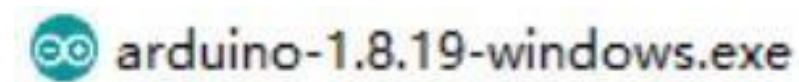
You can choose between an installer (.exe) and a Zip package. We recommend that you use the first "Windows Win7 and newer" to directly install everything you need to use the Arduino software (IDE), including drivers. With the Zip package, you need to install the driver manually. Of course, Zip files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

Click on "Windows Win7 and newer"

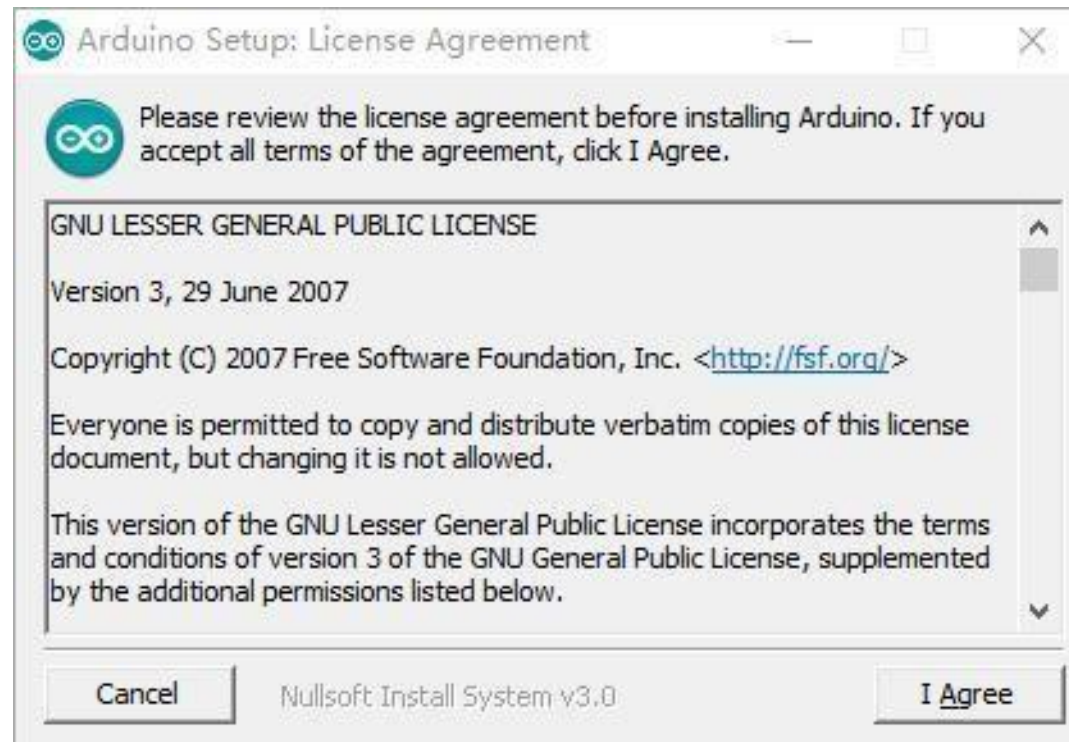


Click on "JUST DOWNLOAD".

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

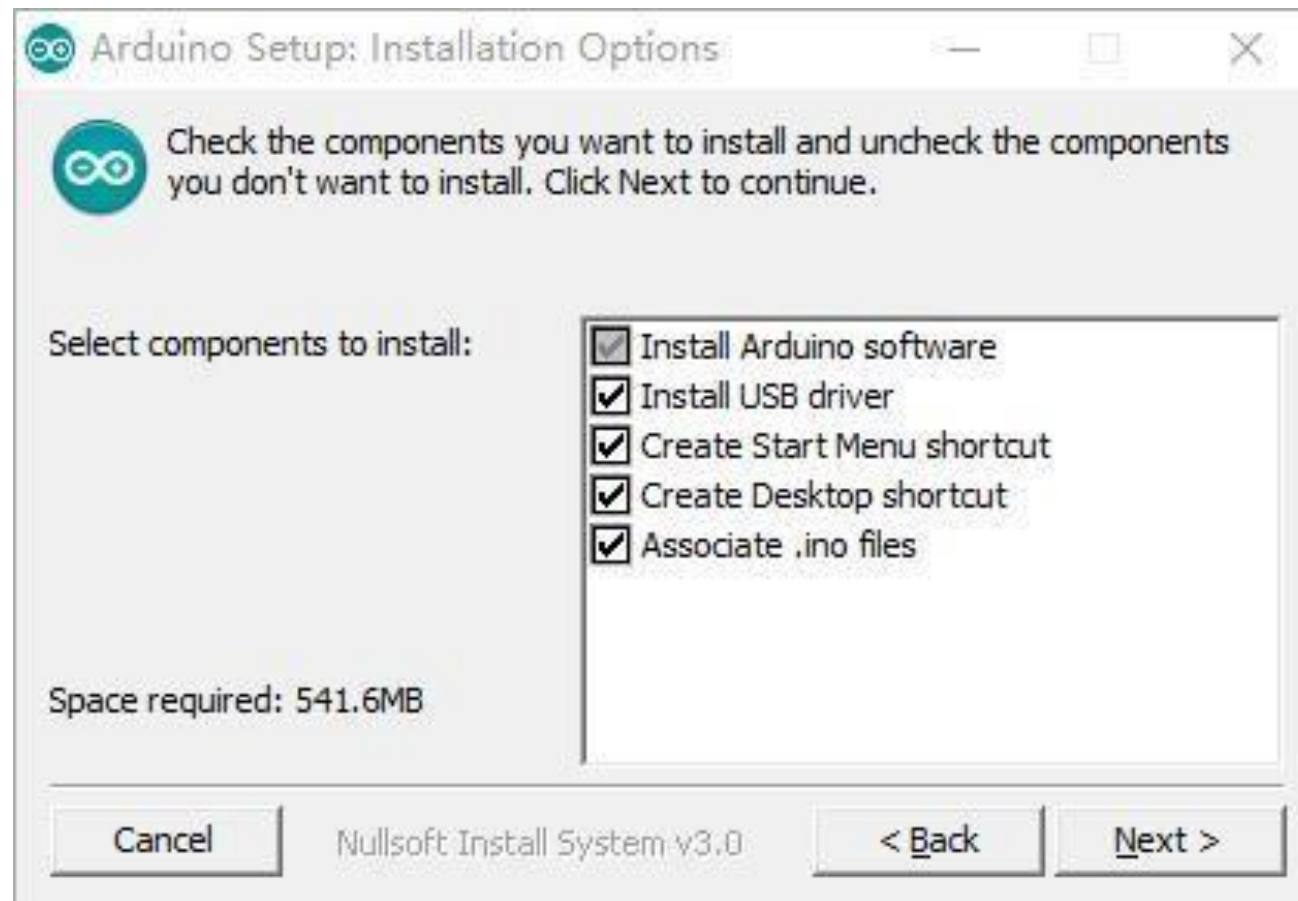


Double click to run the installer

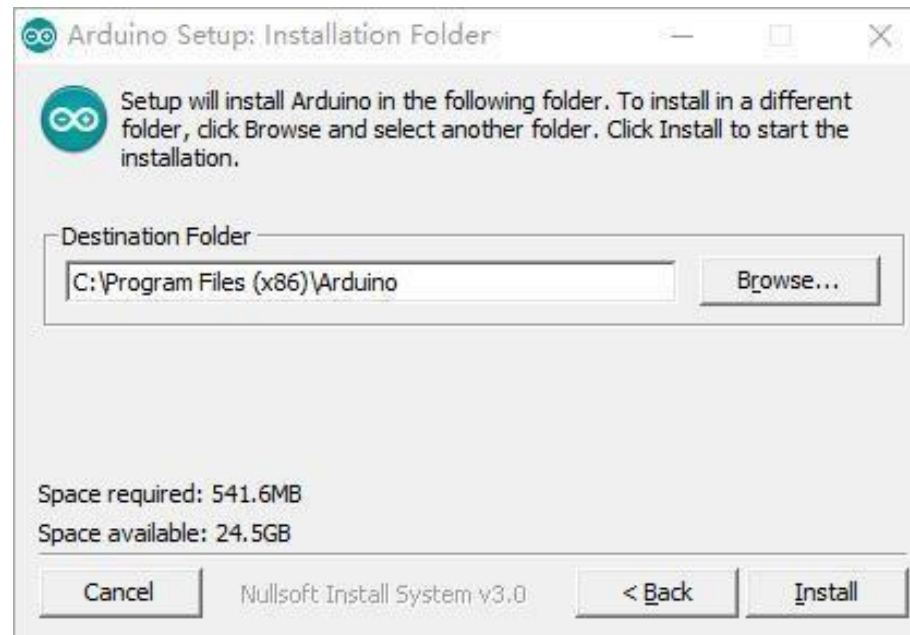


Click "I Agree" to see the following interface





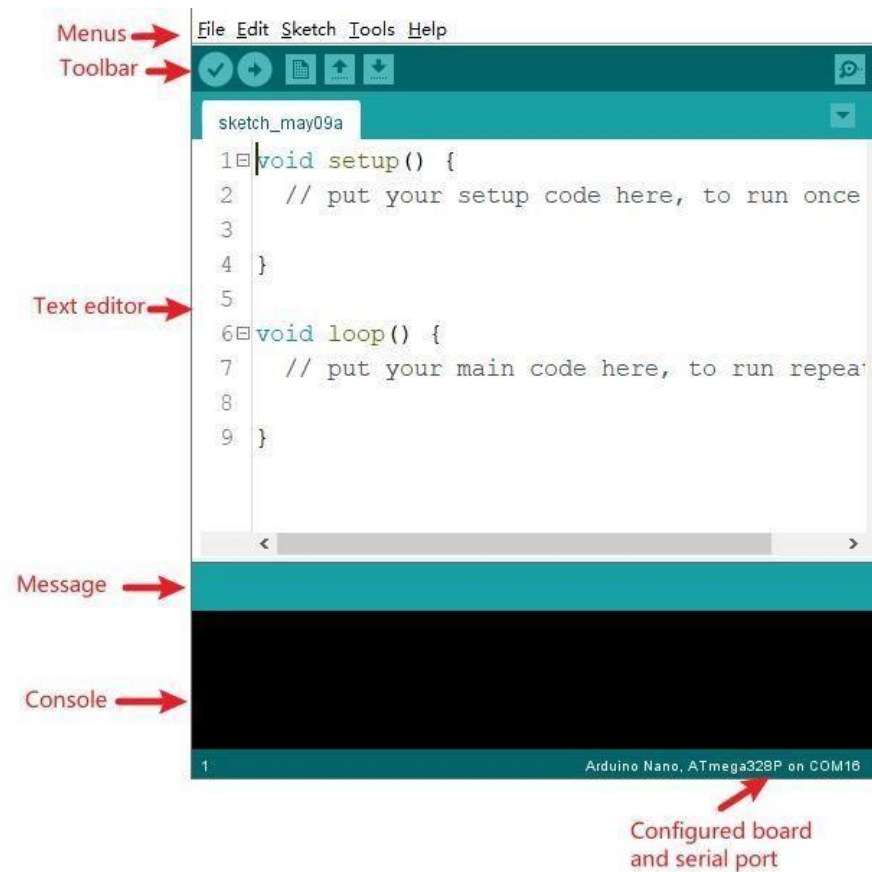
Click "Next"



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:



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 the same name folder, 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 IDE.

### **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.

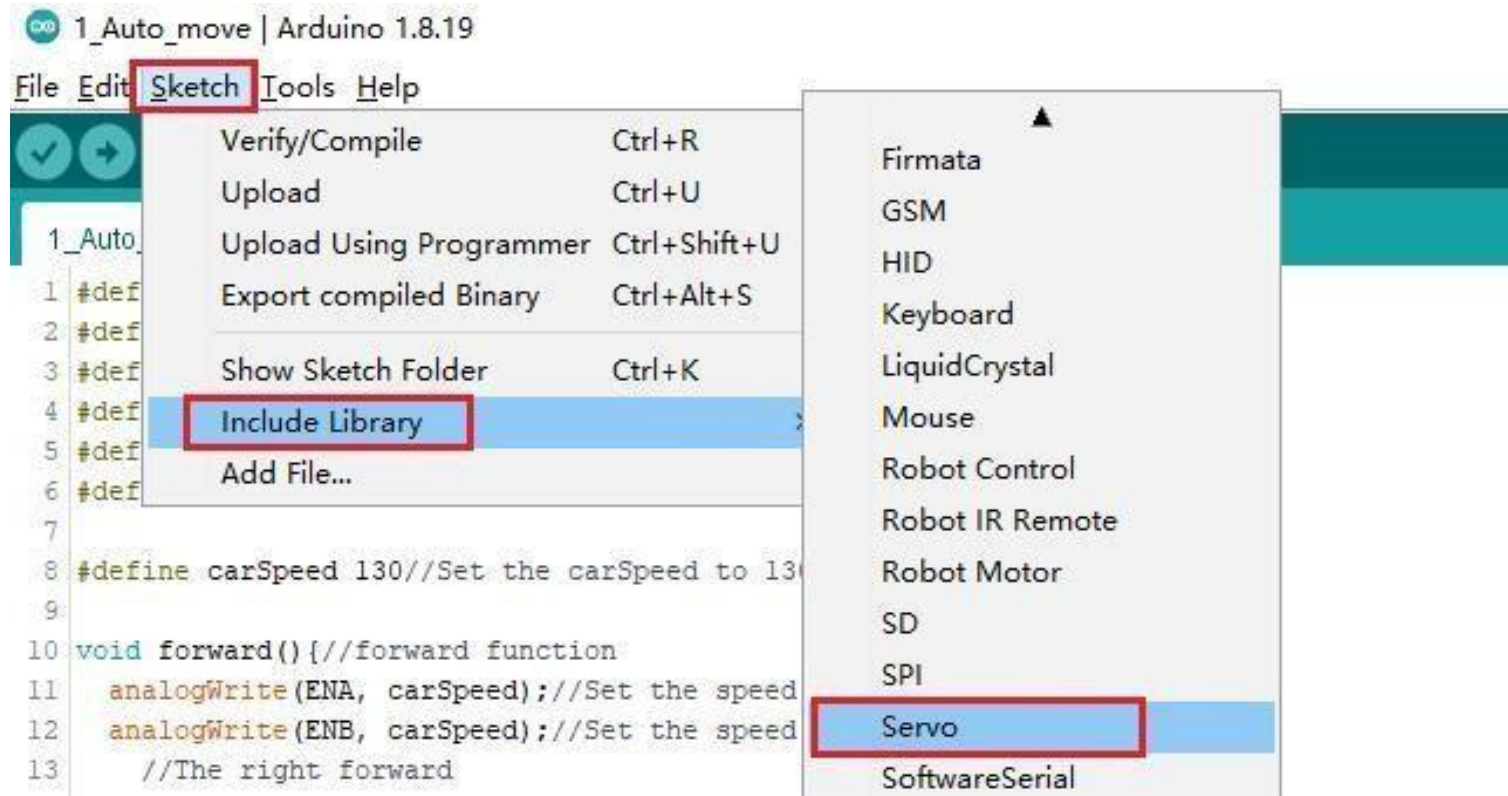
There are thousands of libraries available for download directly through the Arduino IDE, all of which you can find in the [Arduino Library Reference](#) .

In this tutorial package, the ESP32 libraries are placed in the same directory as the main program, and you can open the

program and run it directly. It is important to note that you should not arbitrarily modify or move program files to prevent library files from becoming invalid.

 esp camera	2022/11/2 9:37	文件夹	
 6.1_ESP32_Car.ino	2022/11/2 12:04	INO 文件	
 app_httpd.cpp	2022/9/21 16:49	C++ 源文件	3
 camera_index.h	2022/9/21 16:49	C Header 源文件	2
 esp camera.zip	2020/8/15 17:16	WinRAR ZIP 压缩...	1,93
 inols.log	2022/11/23 14:38	文本文档	58
 inols-clangd.log	2022/11/17 14:50	文本文档	78
 inols-clangd-err.log	2022/11/17 14:50	文本文档	3,20
 inols-err.log	2022/11/23 14:38	文本文档	1,86

Arduino UNO also uses servo libraries. Newer versions of the Arduino IDE have integrated this library. You can open the Arduino IDE and find it in Sketch>Include Library .

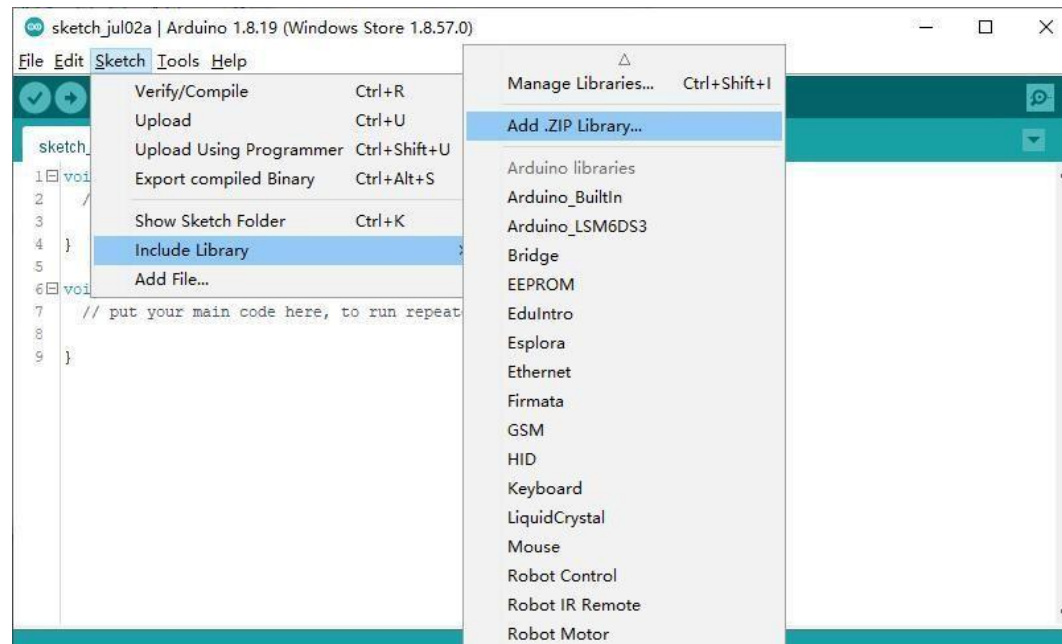


If the library is not found , follow the steps below to install the library .



## 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. 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.



The system will prompt you to select the library you want to add , navigate to the saved library on your computer as shown below The path location of the servo .zip file ( 2\_Libraries/servo.zip ) and open it .



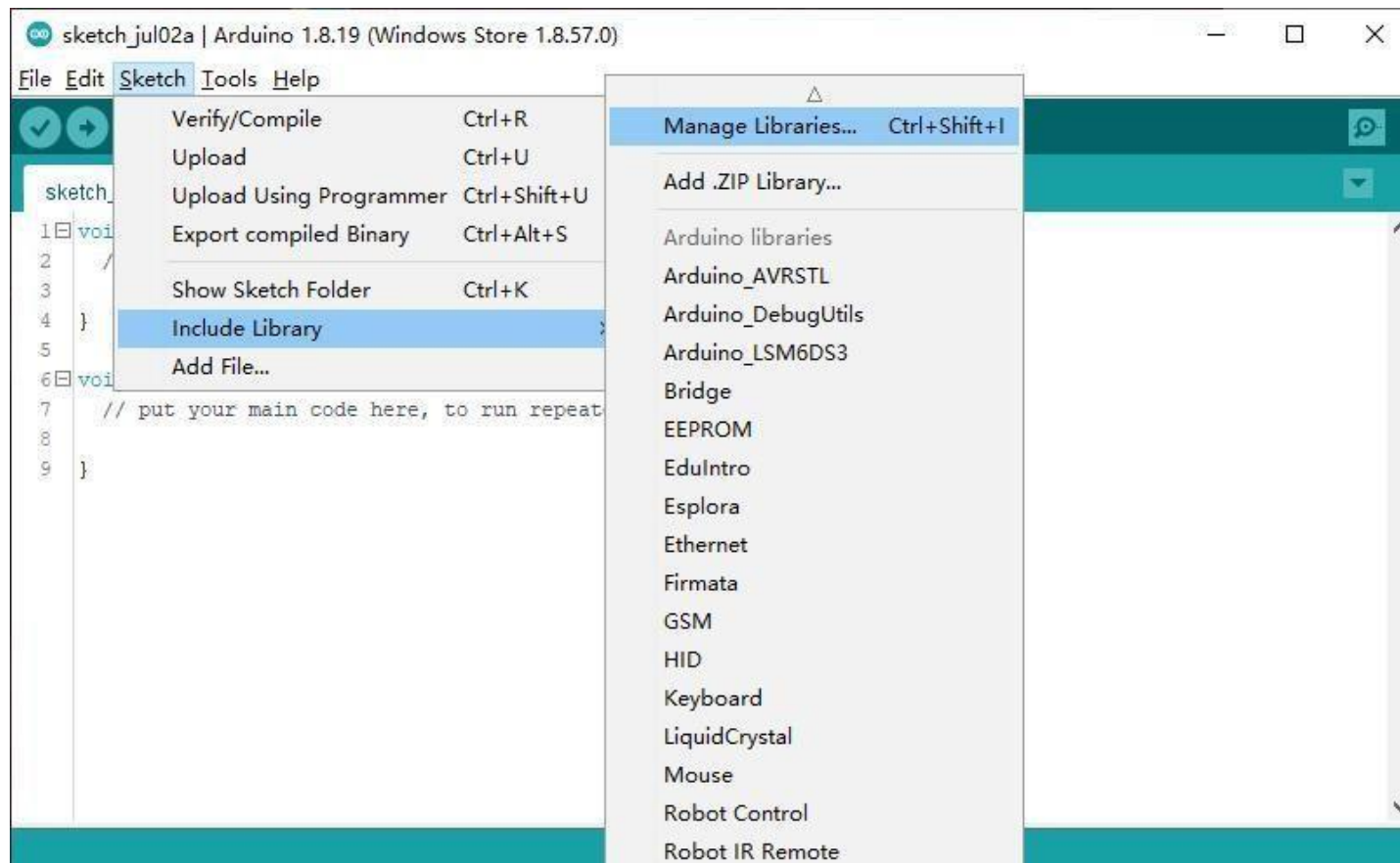
Return to the Sketch > Include Library menu. You should now see the library at the bottom of the drop-down menu , ready to use in your sketches .

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.

**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 click Include Library > Manage Libraries.



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 servo 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 .



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

Likewise, you can now find new libraries available in the Sketch > Include Library menu.

## **Lesson 3 The first program code - Blink**

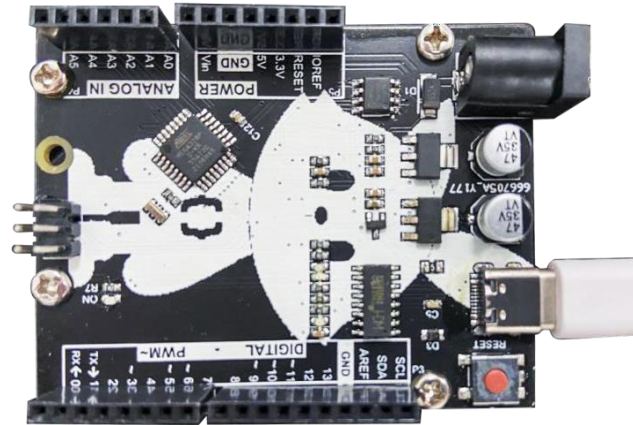
### **About this lesson:**

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 .



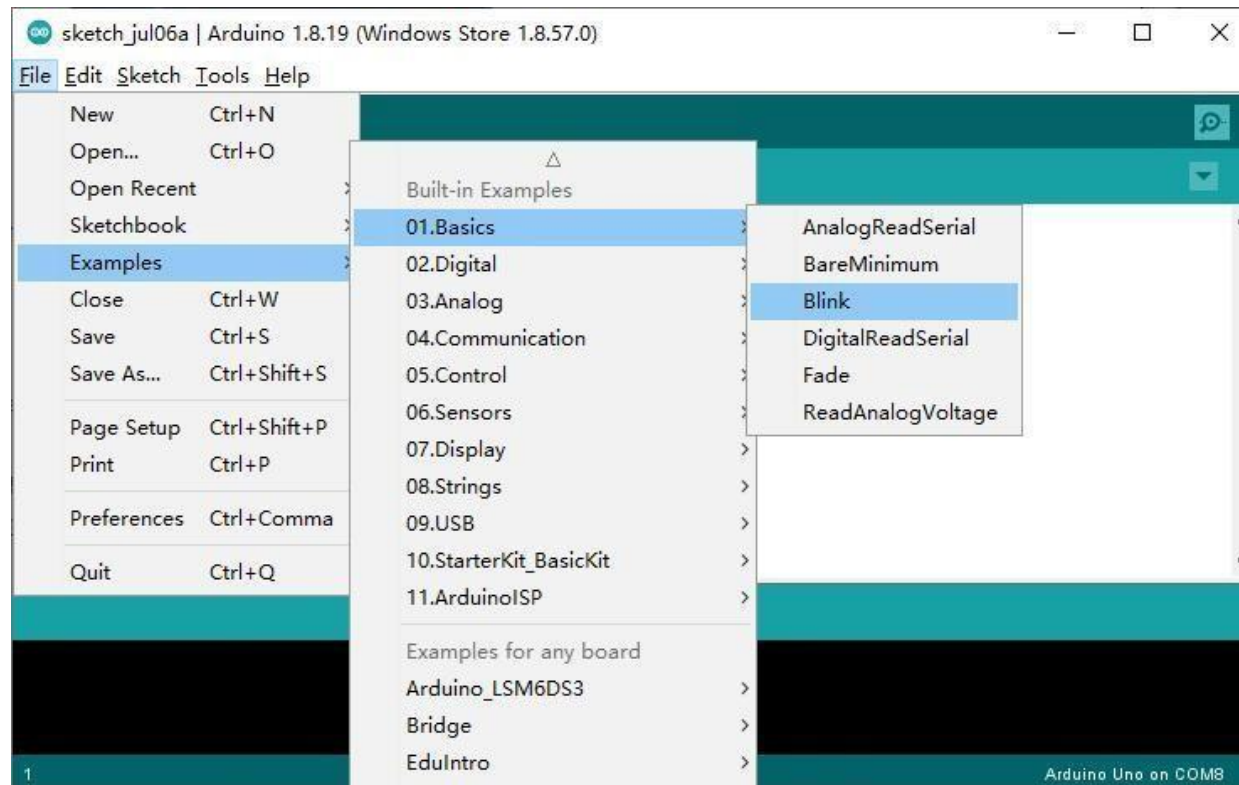
When you connect the board to the USB plug, you may find that its LEDs are already blinking because the "Blink" sketch is pre-installed on the board.

### **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

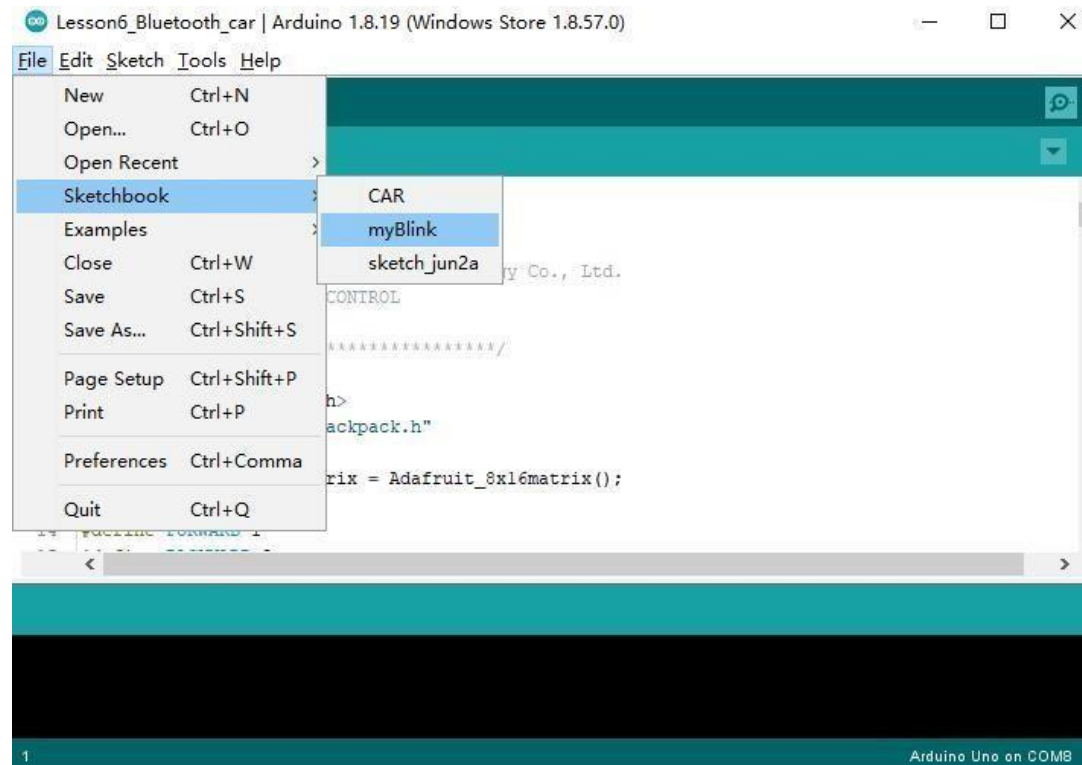
for making an "L" LED. In the IDE menu system File > Examples > 01. The " Blink " sketch you will find in Basics .



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.

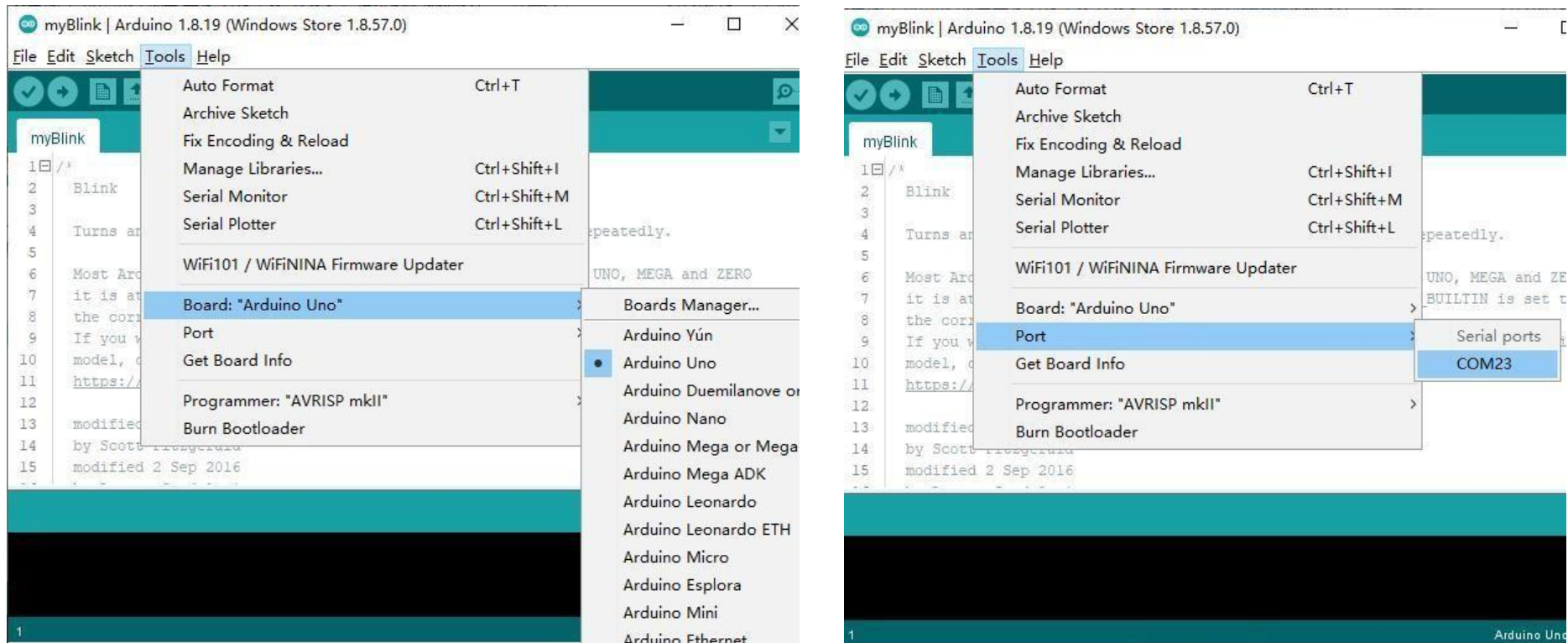
From the Arduino IDE's File menu, select "Save As.." and save the sketch as "MyBlink".

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.





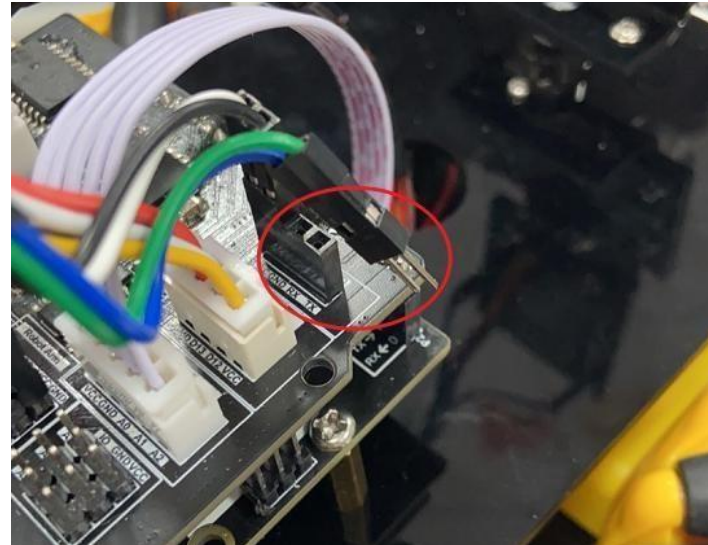
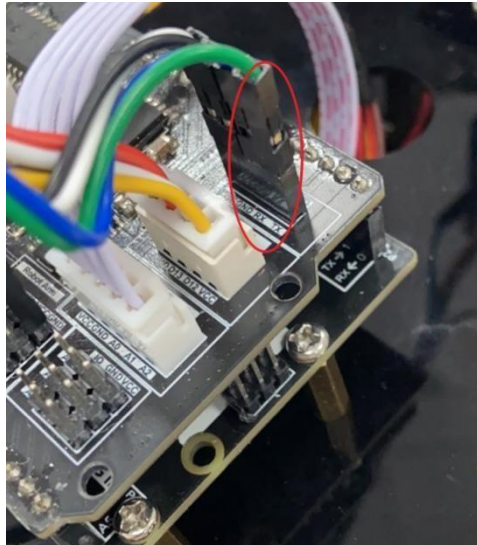
the Arduino board to the computer using a USB cable and check that the Board Type and Serial Port are set correctly.



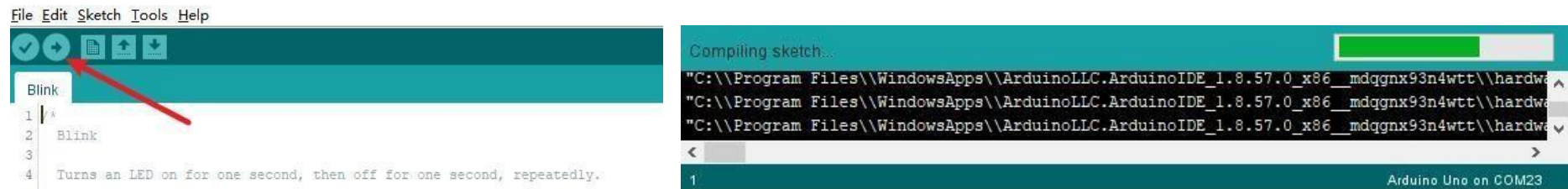
Note: The control board type here is Uno . If you are using 2560, then you will have to choose Mega 2560 as the Board Type , the other selection methods are the same. The serial port appears to be different for everyone, although COM23 is chosen here , it could be COM3 or COM4 on your computer.



Because the communication between the car expansion board and the ESP32-CAM expansion board will occupy the RX/TX port, so unplug the two serial ports of RX/TX before uploading the program.



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.



The transfer is complete, "Transfer complete" appears



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



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 ()".

```
32 void loop() {  
33     digitalWrite(LED_BUILTIN, HIGH);    // turn the LED on (HIGH is  
34     delay(1000);                        // wait for a second  
35     digitalWrite(LED_BUILTIN, LOW);     // turn the LED off by makin  
36     delay(1000);                        // wait for a second  
37 }
```

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.

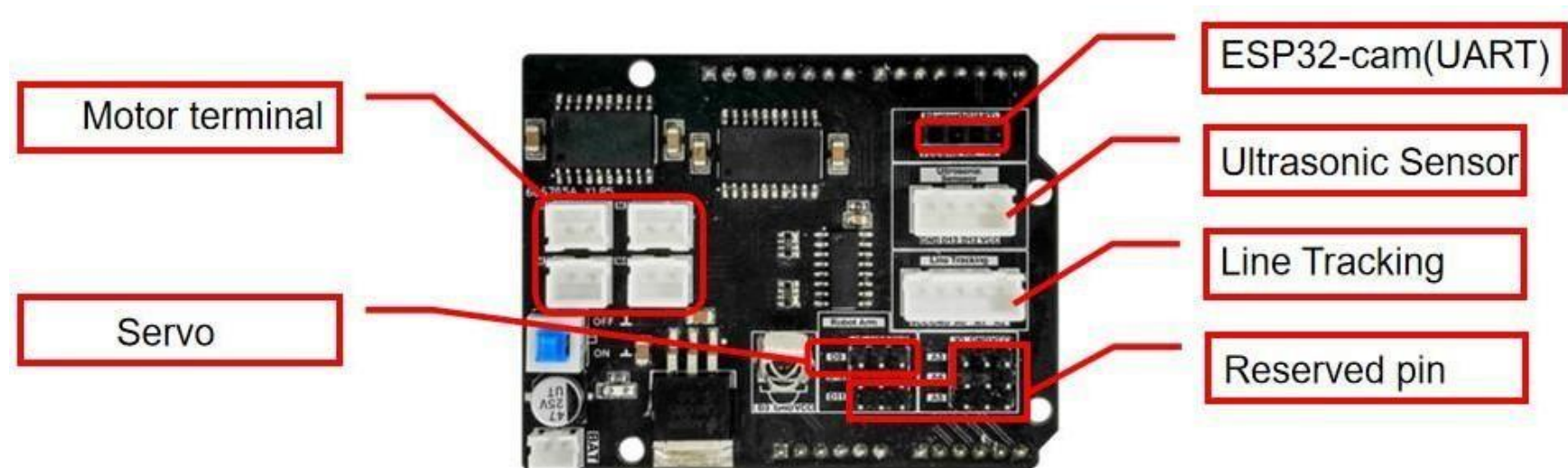


## Lesson 4 Expansion Board and Motor Driver

### Course Introduction

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

### Expansion board introduction



The expansion board extends the pins of the main board well, and the motor, ultrasonic and tracking modules are all connected by terminal plugging, which increases firmness and reliability. The switch-controlled BAT port is reserved, the reserved infrared receiver is stably integrated on the board, 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:**



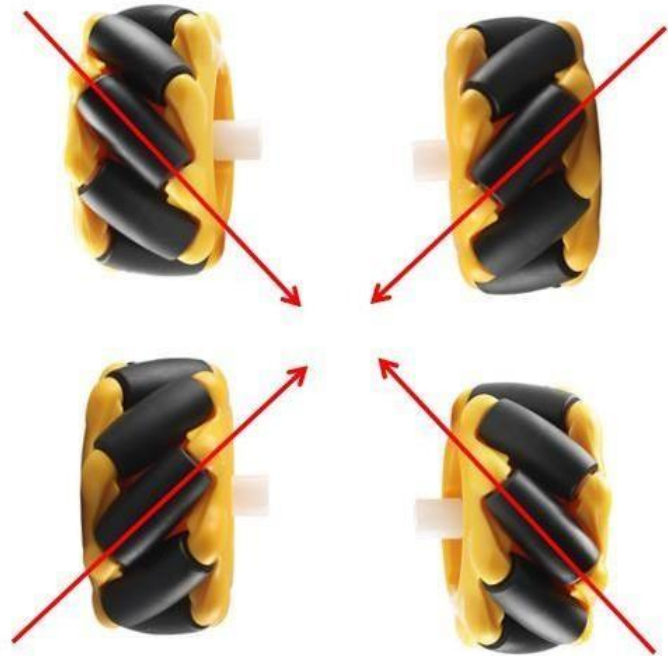
Mecanum wheel is a kind of wheel with a peripheral axle, generally divided into two types, one peripheral axle is inclined



to the left, and the other is inclined to the right. These angled peripheral axes 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.

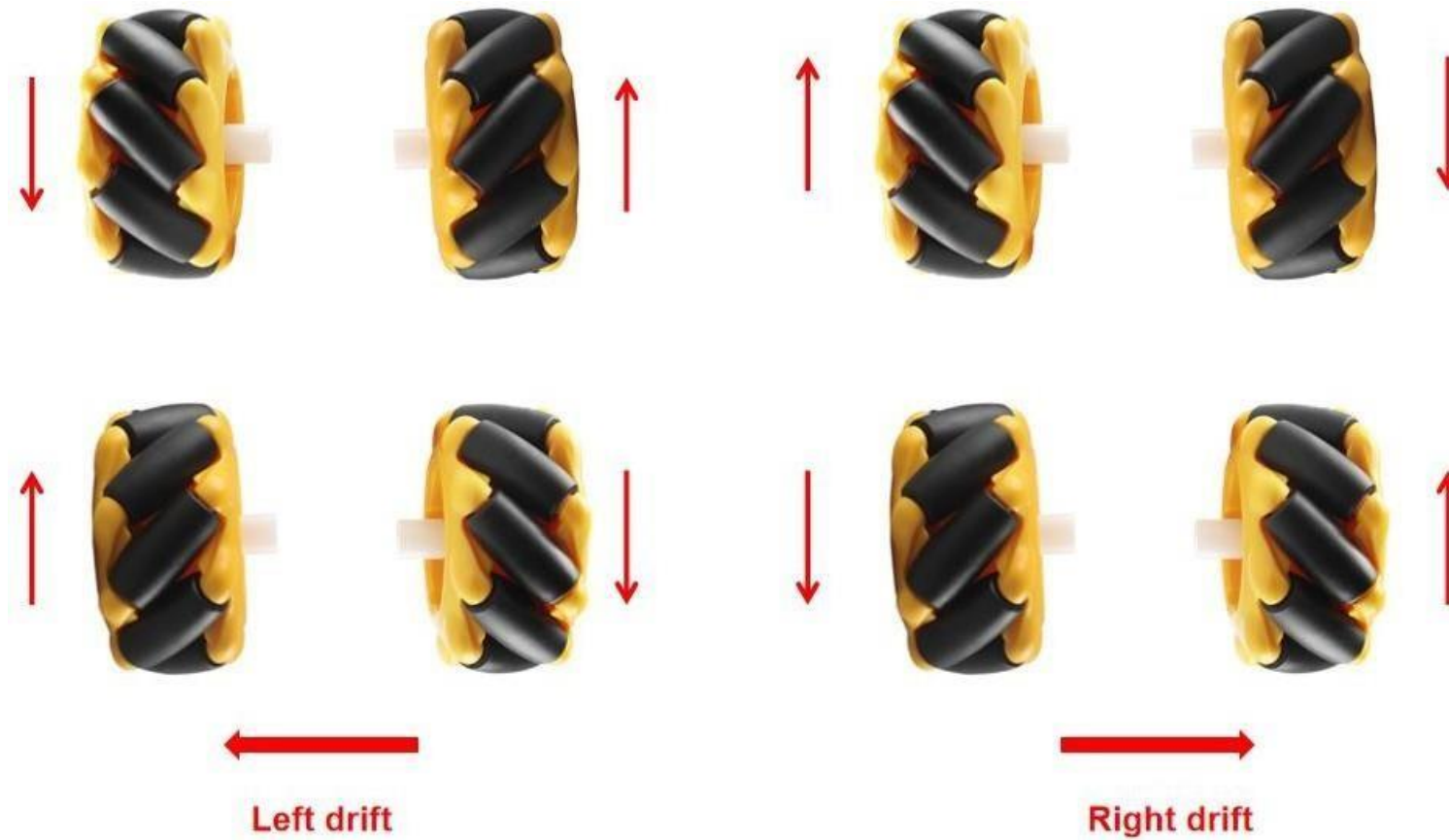
**Assembly points (top view):**

The peripheral axle points to the center of the car

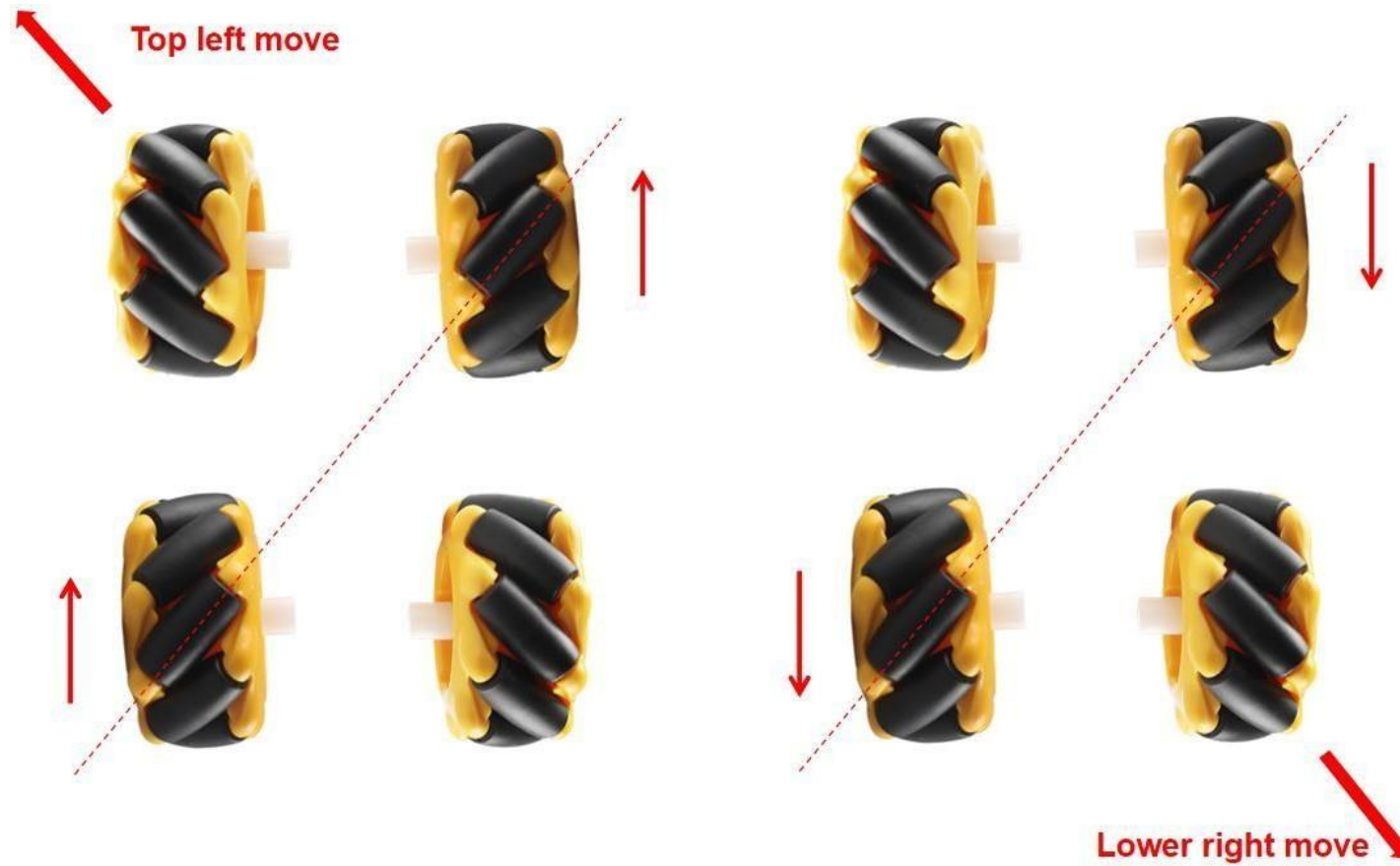


**Movement principle:**

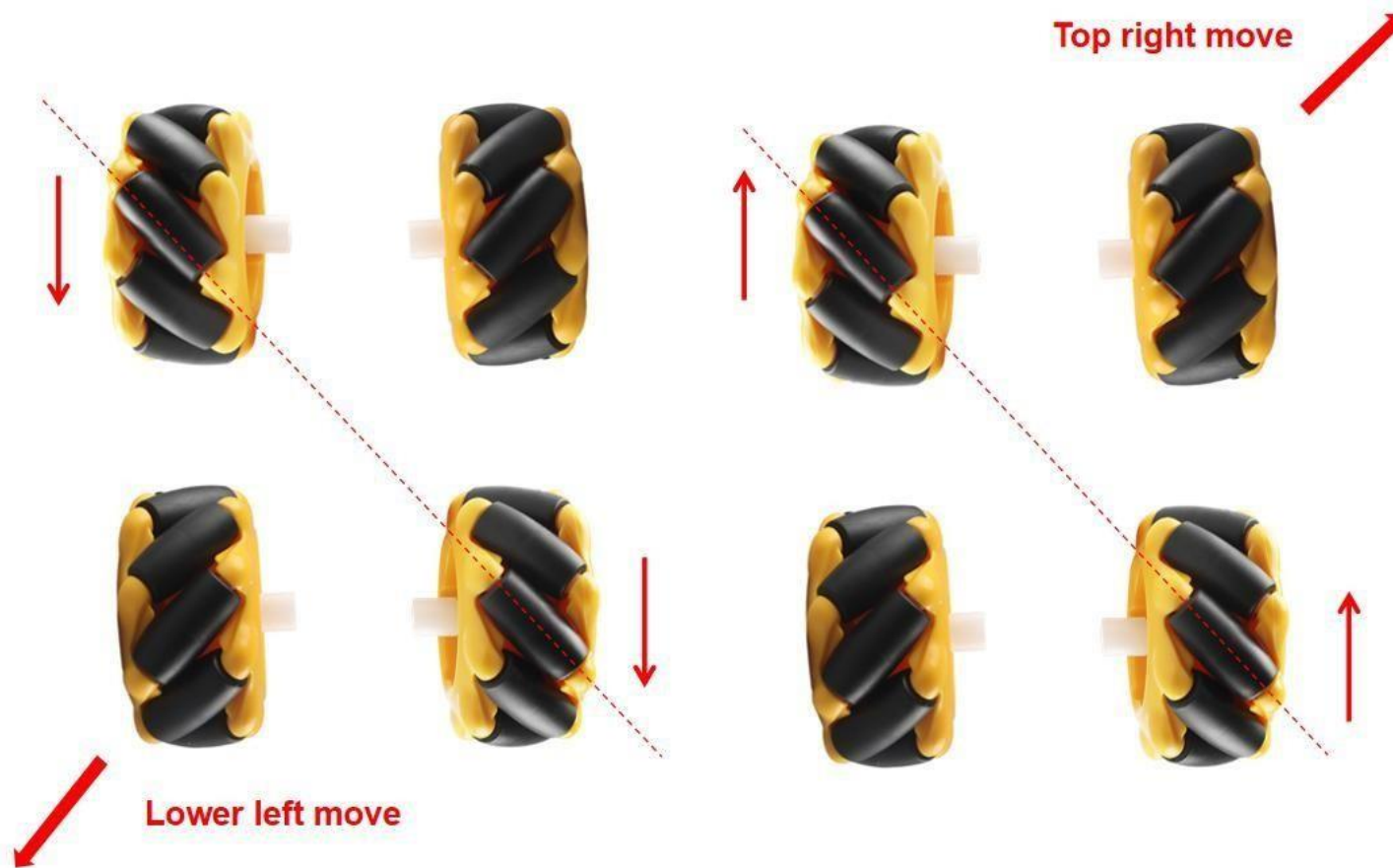
The different rotation directions of the wheel correspond to the left and right translation movement:



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



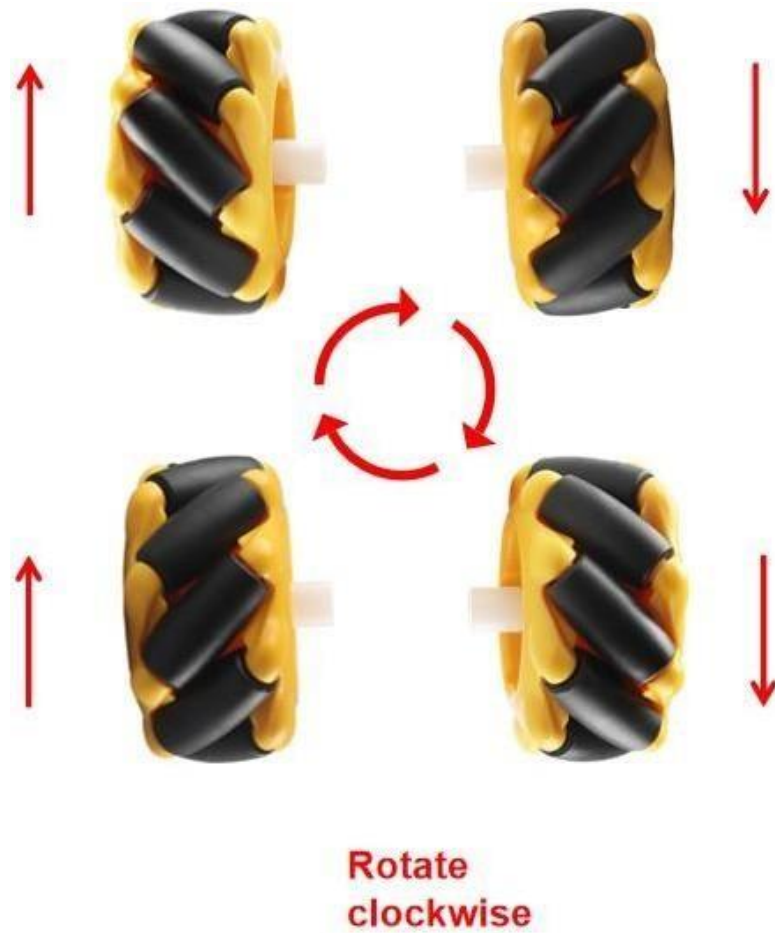
The different rotation directions of the wheels correspond to the movement in the lower left direction and the upper right direction:



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

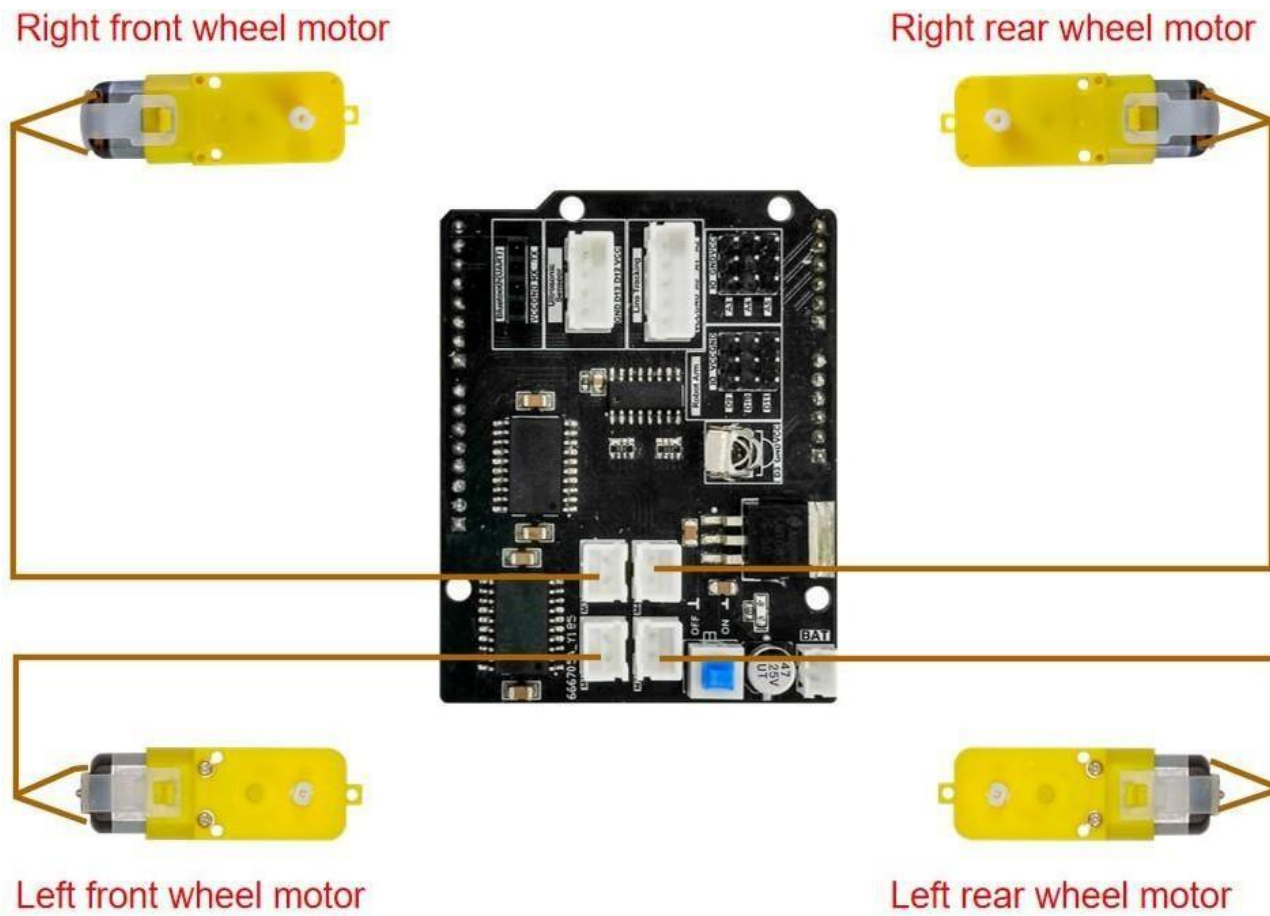


The different rotation directions of the wheels correspond to counterclockwise and clockwise rotations:





Wiring diagram:



## code analysis

Open the program file (path: 4\_Arduino Code\1\_Auto\_move\1\_Auto\_move.ino)( **Reminder: Please unplug the RX/TX cables of the expansion board before uploading the code, otherwise the upload will not be successful !**)

### 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;

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:

```

11  const int Forward      = 92;           // forward
12  const int Backward     = 163;          // back
13  const int Turn_Left    = 149;          // left translation
14  const int Turn_Right   = 106;          // Right translation
15  const int Top_Left     = 20;           // Upper left mobile
16  const int Bottom_Left  = 129;          // Lower left mobile
17  const int Top_Right    = 72;           // Upper right mobile
18  const int Bottom_Right = 34;           // The lower right move
19  const int Stop         = 0;            // stop
20  const int Contrarotate = 172;          // Counterclockwise rotation
21  const int Clockwise     = 83;          // Rotate clockwise
22

```

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 5 Servo Motors

## Course Introduction

Understand the properties and characteristics of the steering gear and learn the relevant knowledge of the steering gear, and master the debugging method and circuit connection of the steering gear, and finally experience the working method of the steering gear in the Arduino programming.

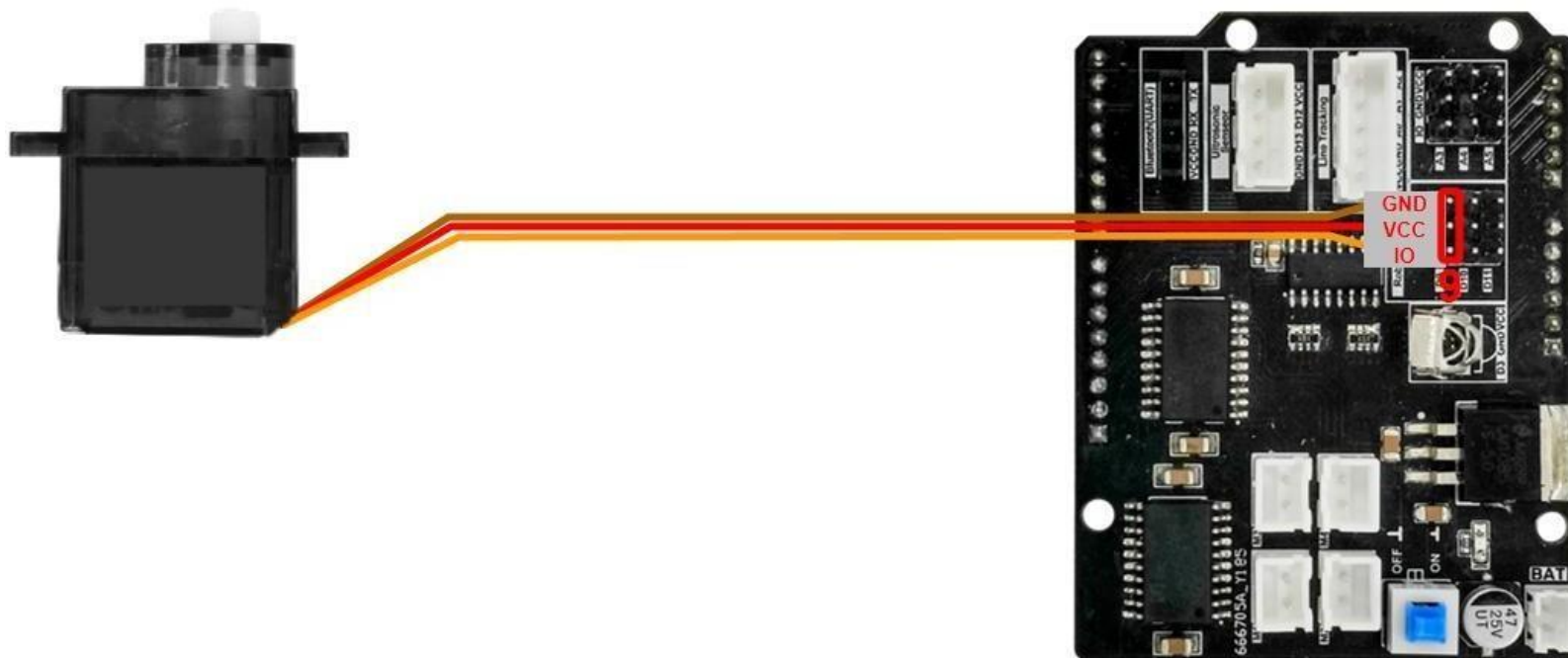
## Introduction of steering gear



This servo motor can control the direction and speed and give a PWM signal to the 360-degree steering gear. In 20msPWM square wave, the maximum speed at 0.5ms is left turn (reverse), the maximum speed at 1.5ms is stop rotation,

and the maximum speed at 2.5ms is right turn (forward). The steering gear will turn at a specific speed, similar to an electric motor. But unlike the motor, 360 steering gear is closed-loop control, speed control is stable. 90 stop, 10 or 170 for positive and negative rotation value the closer to 90, the slower the rotation speed.

## Wiring diagram



## code analysis

Open the code file (path: 4\_Arduino\_Code\2\_servo\_Angle\2\_servo\_Angle.ino)

Import the servo library file

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

Declare the servo motor signal port as 9

```
#define servo_PIN 9
```

Initialize setting of servo motor angle

```
void setup ()  
{  
  myservo . attach (servo_PIN);  
  myservo.write ( 90 );  
}
```

Loop execution, the servo motor to the left and right reciprocating rotation

```
void loop(){  
  myservo.write(45);  
  delay(15);  
  myservo.write(135);  
  delay(15);  
}
```

# Lesson 6 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 their 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:  $S = VT/2$**

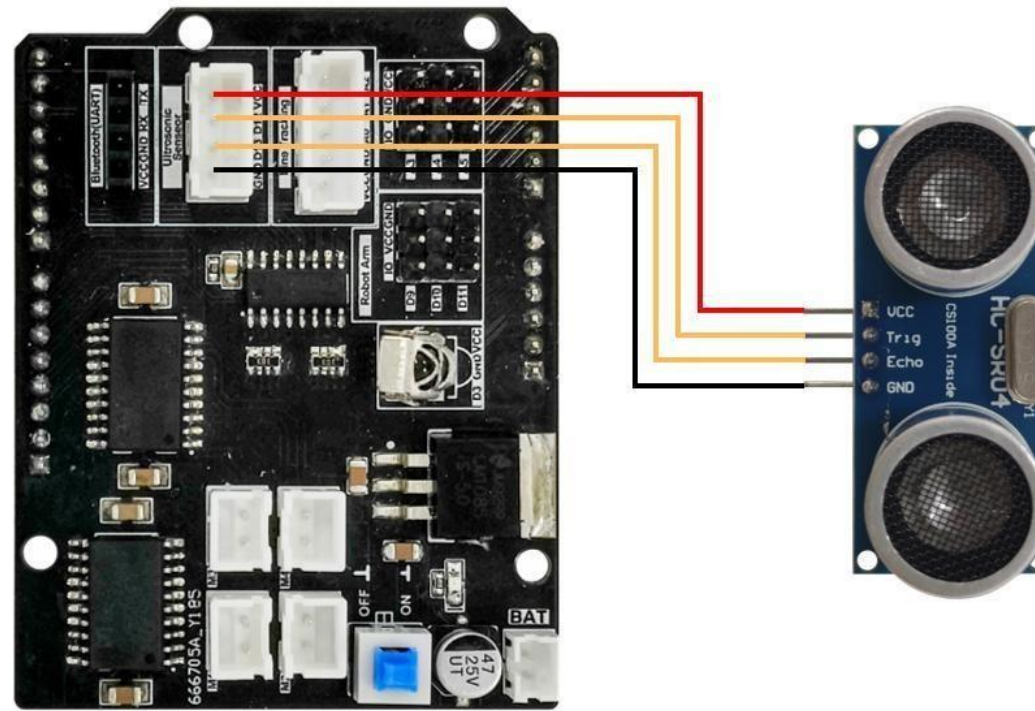
Four factors limit the maximum measurable distance of an ultrasound system: the amplitude of the ultrasound, the texture of the reflector, the angle between the reflected and incident sound waves, and the sensitivity of the receiving transducer. The ability of the receiving transducer to directly receive 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:

$$\text{distance} = (\text{high level time} * \text{speed of sound}) / 2 .$$

## Wiring diagram



## code analysis

Open the code file (path: 4\_Arduino Code\3\_Ultrasonic\3\_Ultrasonic.ino)

Get the ranging distance function checkdistance( )

```
float checkdistance ()  
{  
    digitalWrite ( 12 , LOW);  
    delayMicroseconds ( 2 );  
    digitalWrite ( 12 , HIGH);  
    delayMicroseconds ( 10 );  
    digitalWrite ( 12 , LOW);  
    float distance = pulseIn ( 13 , 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.

Print ultrasonic measurements to serial monitor

```
void Ultrasonic_Sensor_Module ()  
{  
    int Distance = 0 ;  
    Distance = checkdistance ();  
    Serial . print ( "Distance:" );  
    Serial.print ( Distance ) ;  
    Serial.println ( "CM " );  
    delay ( 100 );  
}
```

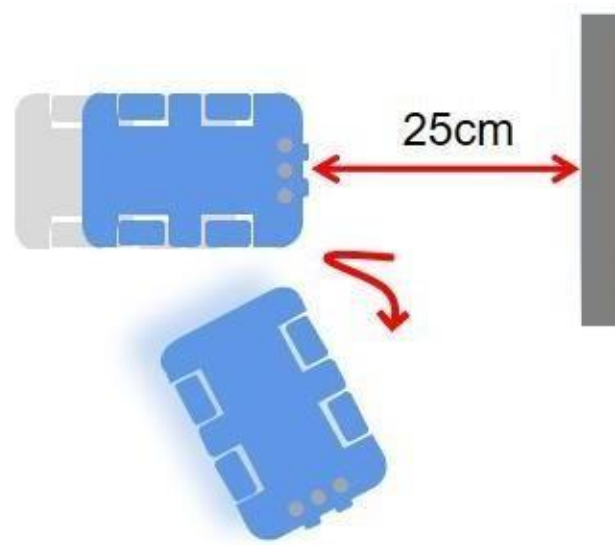
# Lesson 7 Obstacle Avoidance Car

## Course Introduction

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

## Obstacle Avoidance Principle

When the distance detected by ultrasonic is less than the set distance, the car is judged to encounter obstacles, and then the obstacle avoidance program is triggered to make the car back and turn left or right to avoid obstacles, so as to realize the automatic driving of the car to avoid obstacles.



## code analysis

Open the code file (path: 4\_Arduino\_Code\4\_Obstacle\_Avoidance\4\_Obstacle\_Avoidance.ino)

Define the ultrasonic control pins

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

Obtain 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;
}
```

Save the ultrasonic measurement distance obtained by the SR04 function into 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, let the car stop, back and then 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 8 Tracking Car

## Course Introduction

In this lesson we will learn how to use the line tracking module to learn the principle of line tracking and how to program a car to implement line tracking .

### tracking module

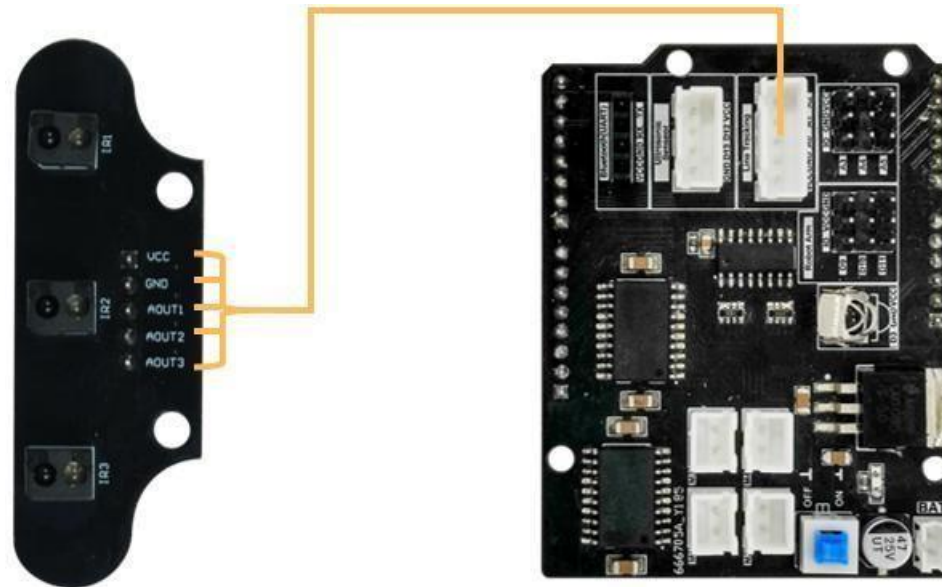


tracking sensor, which is usually used in the manufacture of tracking smart cars. The tracker sensor adopts ITR20001/T infrared reflection sensor. The infrared emitting diode of the ITR2001/T sensor continuously emits infrared rays. When the emitted infrared rays are reflected by objects, they are received 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:

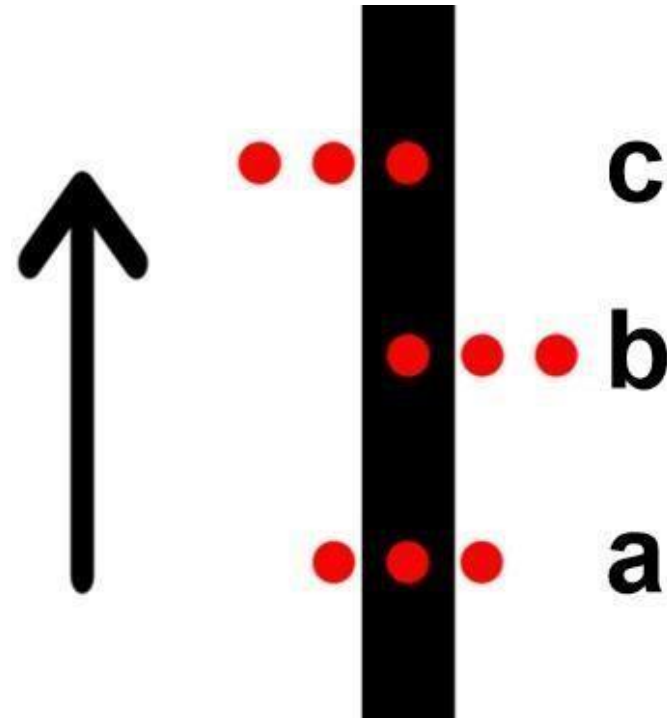


## Get the measurement value of the three-way tracking sensor:

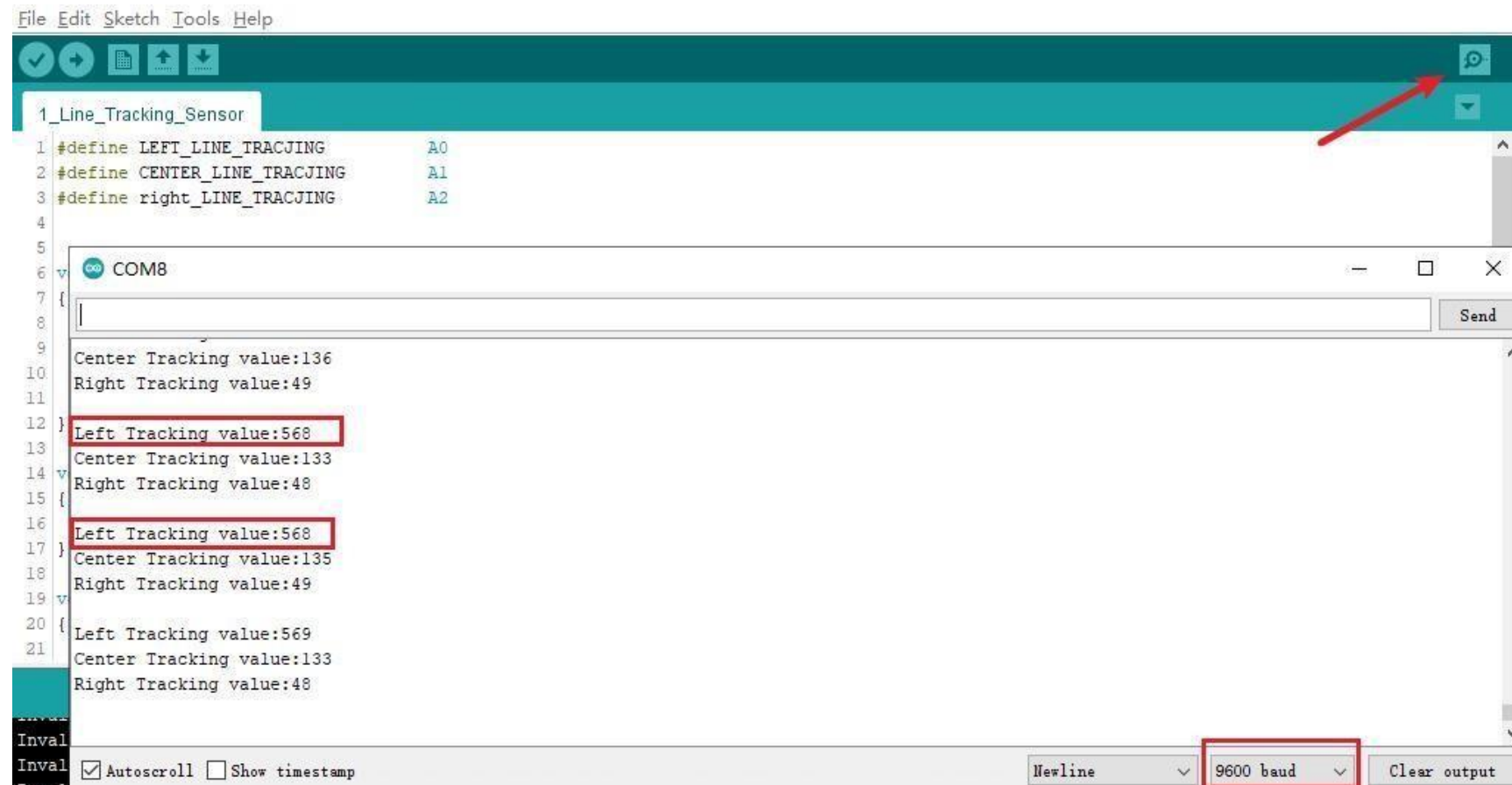
Open the code file (path: 4. Tutorial\_Arduino\4\_Arduino Code\3.1\_Line\_Tracking\_Sensor\3.1\_Line\_Tracking\_Sensor.ino )

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:



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

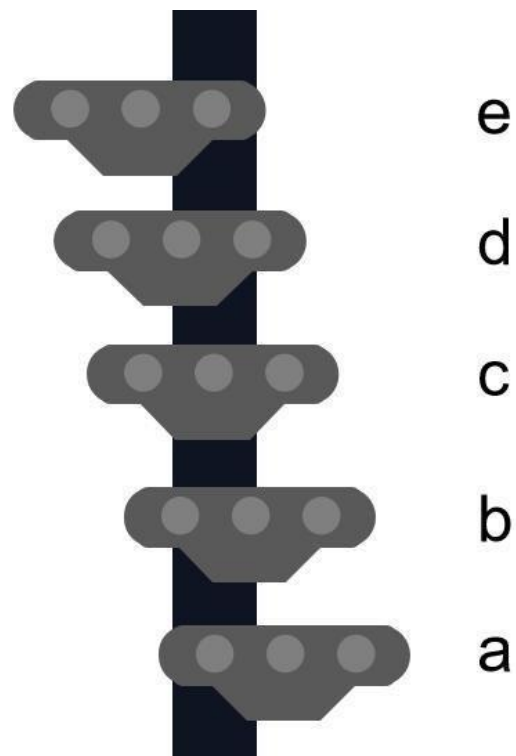
Center Tracking value:139  
Right Tracking value:549  
  
Left Tracking value:151  
Center Tracking value:136  
Right Tracking value:550  
  
Left Tracking value:148  
Center Tracking value:135  
Right Tracking value:551

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, color tapes and ambient light have an impact on the test results, the tests here do not represent all results)

### Three-way tracking principle



- 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 ;

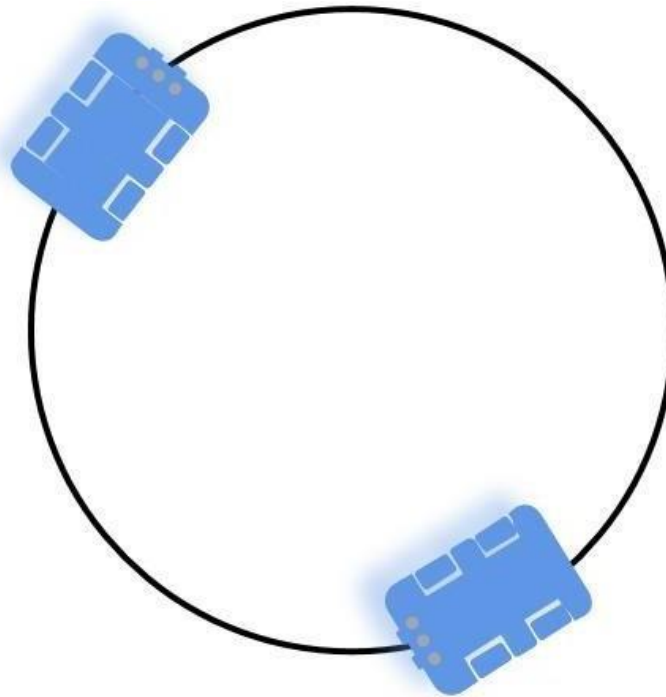
Combining the above information, we can see the tracking principle of the tracking car . After the car starts, the tracking module only needs to sense the black lines on the road and make corresponding actions as

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) for wiring**

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.



## code analysis

Open the code file (path: 4\_Arduino\_Code\5\_Tracking\5\_Tracking.ino)

Define a three-way tracking measurement value variable and a reference value variable "Black\_Line" (intermediate value)

used to compare whether a black line is detected.

```
int Left_Tra_Value;
int Center_Tra_Value;
int Right_Tra_Value;
int Black_Line = 350 ;
```

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 9 Integrated Functions and Remote Control

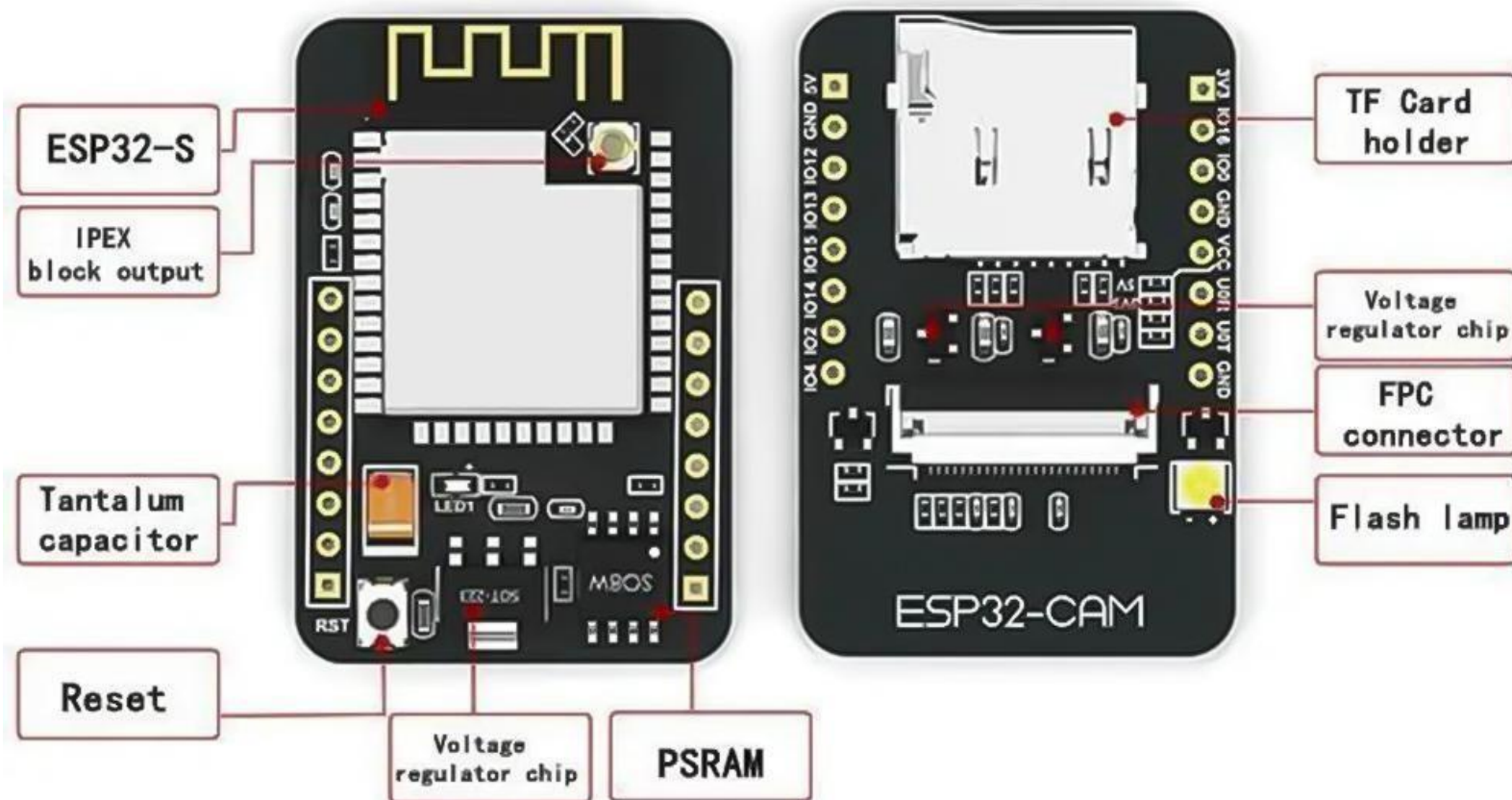
## Course Introduction

This class mainly learns about ESP32-CAM with camera connected to WiFi to realize real-time media streaming shooting and learning to complete the remote control of multi-function mode with ZHIYI main control board.

## Introduction to ESP32-CAM



The ESP32-CAM is a very small camera module with an ESP32-S chip. In addition to the OV2640 camera there are GPIOs for connecting peripherals . The corresponding function list is as follows:





Features the smallest 802.11b / g/n Wi-FiBTSoC module

The clock speed is 160MHz and the total computing power is up to 600 DMIPS

Built-in 520KB SRAM, external 4MPSRAM

Support UART/SPI/I2C/PWM/ADC/DAC

Supports OV2640 and OV7670 cameras with built-in flash memory

Support WiFi to upload images

Support TF card

Supports multiple sleep modes

Embedded Lwip and FreeRTOS

Support STA/AP/STA+AP operation mode

Support smart configuration/AirKiss technology

Support serial port local and remote firmware upgrade (FOTA)

**ESP32-Cam Pinout :** There are three GND pins and two power pins: 3.3V or 5V. GPIO 1 and GPIO 3 are serial pins. You need these pins to upload code to the board. Also, GPIO 0 plays an important role as it determines whether the ESP32 is in blink mode or not. When GPIO 0 is connected to GND, the ESP32 is in blink mode.

### **Video Streaming Server Configuration:**

Follow the steps below to build a video streaming web server using ESP32-Cam that you can access on your local network.

Install the ESP32 core files in the Arduino IDE :

In this example we use Arduino IDE to program the ESP32-Cam board. So you need to install ArduinoIDE and ESP32 add-ons. If you don't already have the ESP32 plugin installed in your computer, follow this tutorial to install it.

Here we will download the core files for the ESP32 board, you need internet to perform this step

Copy this link : [https://dl.espressif.com/dl/package\\_esp32\\_index.json](https://dl.espressif.com/dl/package_esp32_index.json)

Open the Arduino IDE and click File > Preferences

DIY\_Surveillance\_camera\_with\_ESP32-Camera | Arduino 1.8.19

File Edit Sketch Tools Help

New Ctrl+N

Open... Ctrl+O

Open Recent > SP32-Camera

Sketchbook >

Examples >

Close Ctrl+W

Save Ctrl+S

Save As... Ctrl+Shift+S

Page Setup Ctrl+Shift+P

Print Ctrl+P

Preferences Ctrl+Comma

Quit Ctrl+Q

le brownout problems

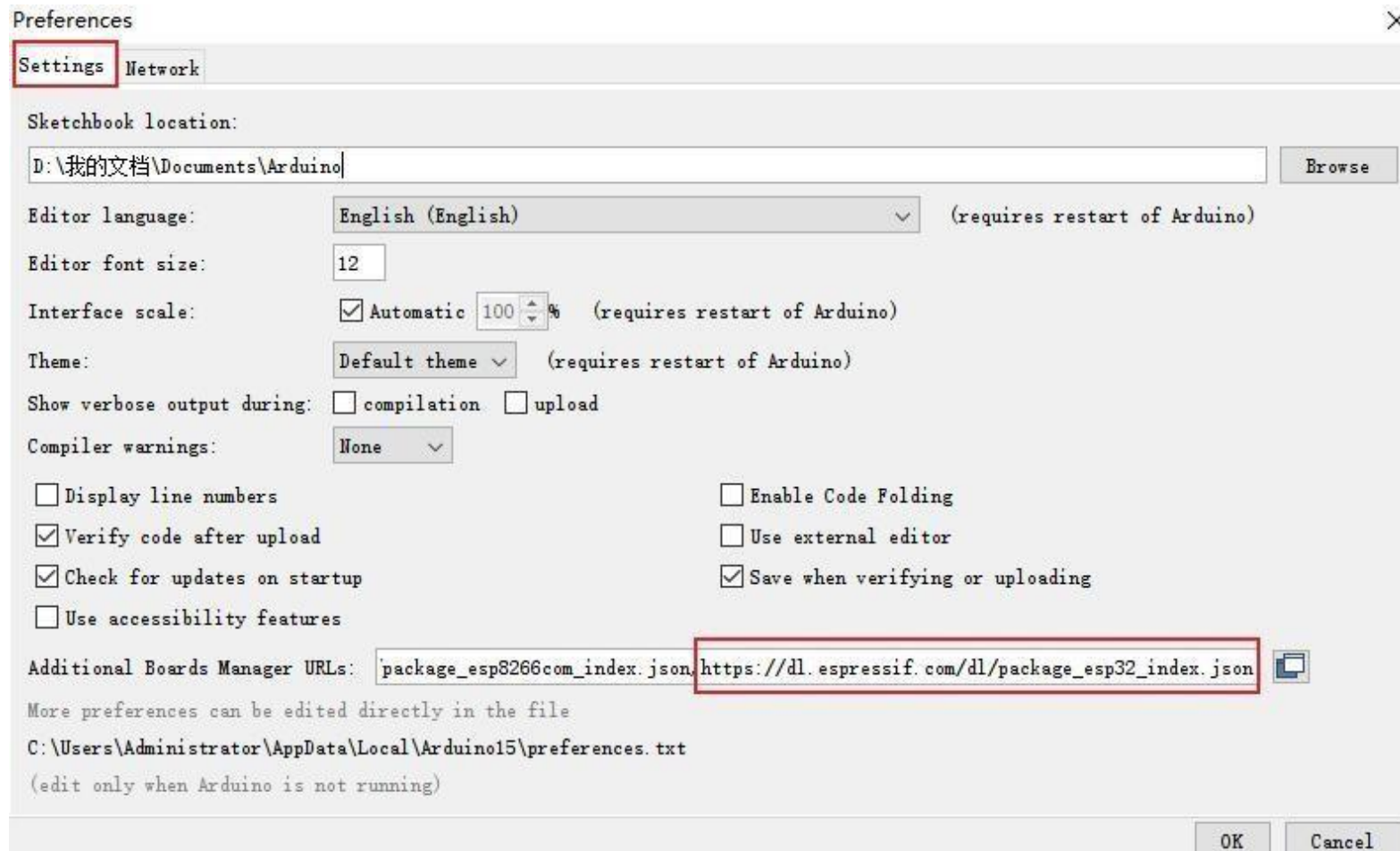
' //disable brownout problems

credentials

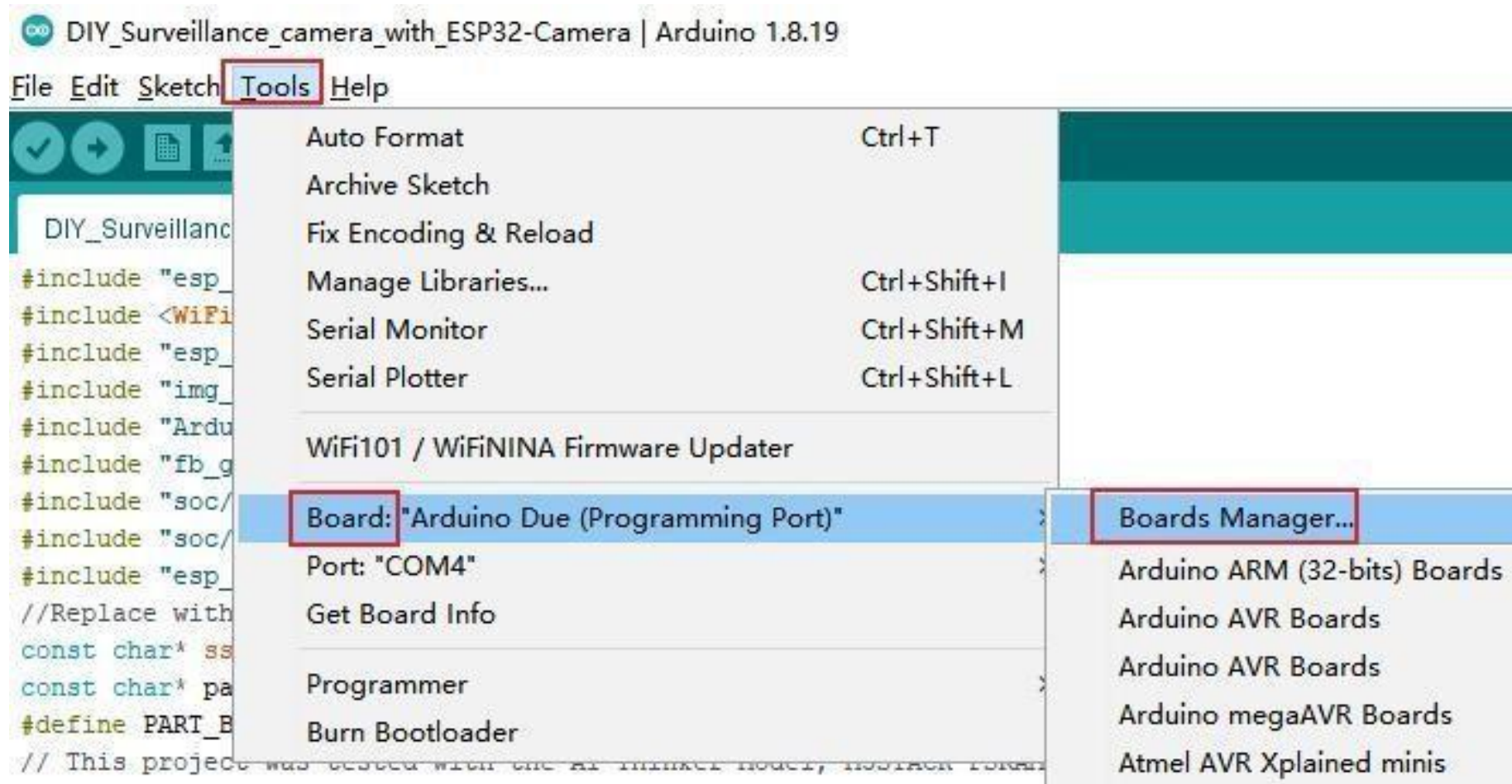
an198910";

57890000000000000987654321"

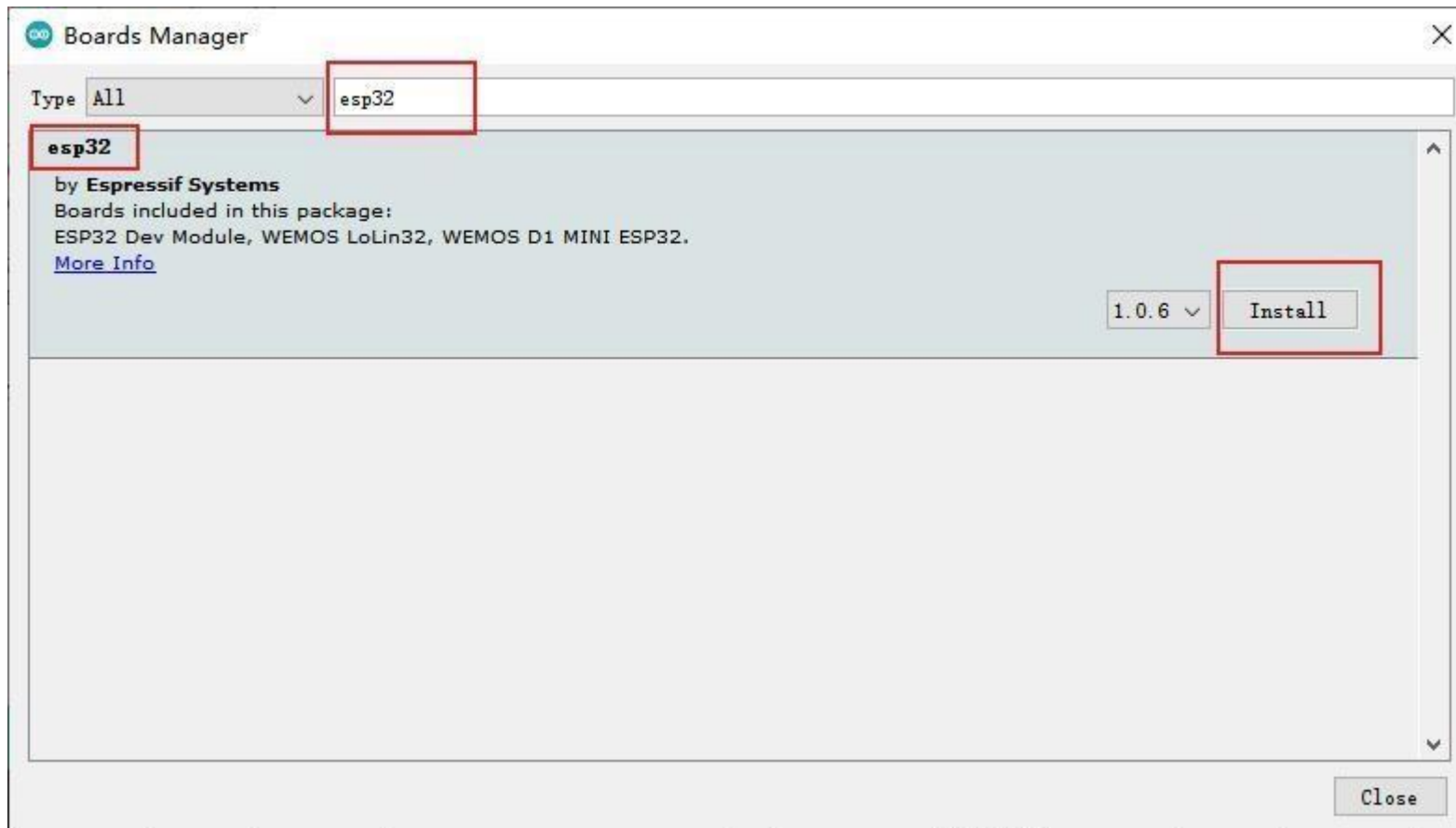
Paste the provided link as shown. If you already have some links, separate them with commas. Press "OK".



As shown below, click Tools > Board > Board Manager



esp 32" in the search bar



Select the latest version and click Install. The download will take some time, depending on your internet speed.

## code analysis

Open the code file (path: 4\_Arduino\_Code\6.1\_ESP32\_Car\6.1\_ESP32\_Car.Ino)

_Arduino_Code\6.1_ESP32_Car				
名称	修改日期	类型	大	
esp camera	2022/9/8 18:34	文件夹		
6.1_ESP32_Car.ino	2022/8/29 17:27	INO 文件		
app_httpd.cpp	2022/8/29 17:03	CppSourceFile		
camera_index.h	2020/12/15 16:48	CHeaderFile		
esp camera.zip	2020/8/15 17:16	WinRAR ZIP 压缩...		

Note: The location of the code file contains the required library files such as ".cpp" / ".h", please do not modify or move any files without authorization!

You can see that there are three files in the Arduino IDE, select the first main program file 6.1\_ESP32\_Car



Before uploading the code, you need to make some modifications . Modify the network credentials account and password variables to your own WiFi name and password !

```
//Replace with your network credentials
const char * ssid = "XXX" ; //"XXX" is the name of your WiFi
const char * password = "XXXXXX" ; //Your WiFi password
#define PART_BOUNDARY "1234567890000000000000987654321"
```

Then, make sure to select the correct camera module. Our correct choice is the AI-THINKER model.

Therefore, please comment all other models and uncomment the appropriate models as follows:

```
// This project was tested with the AI Thinker Model, M5STACK PSRAM Model and M5STACK WITHOUT PSRAM
#define CAMERA_MODEL_AI_THINKER
```

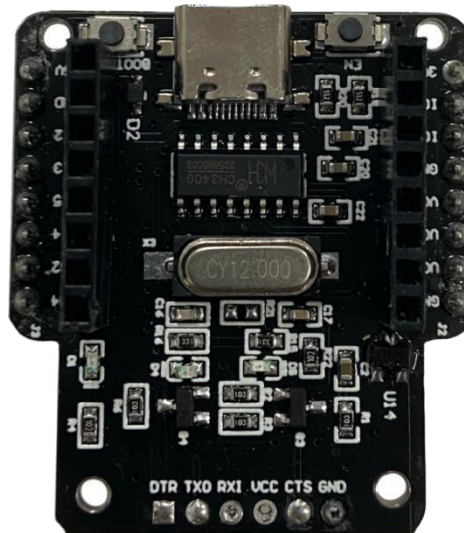


```
//#define CAMERA_MODEL_M5STACK_PSRAM  
//#define CAMERA_MODEL_M5STACK_WITHOUT_PSRAM
```

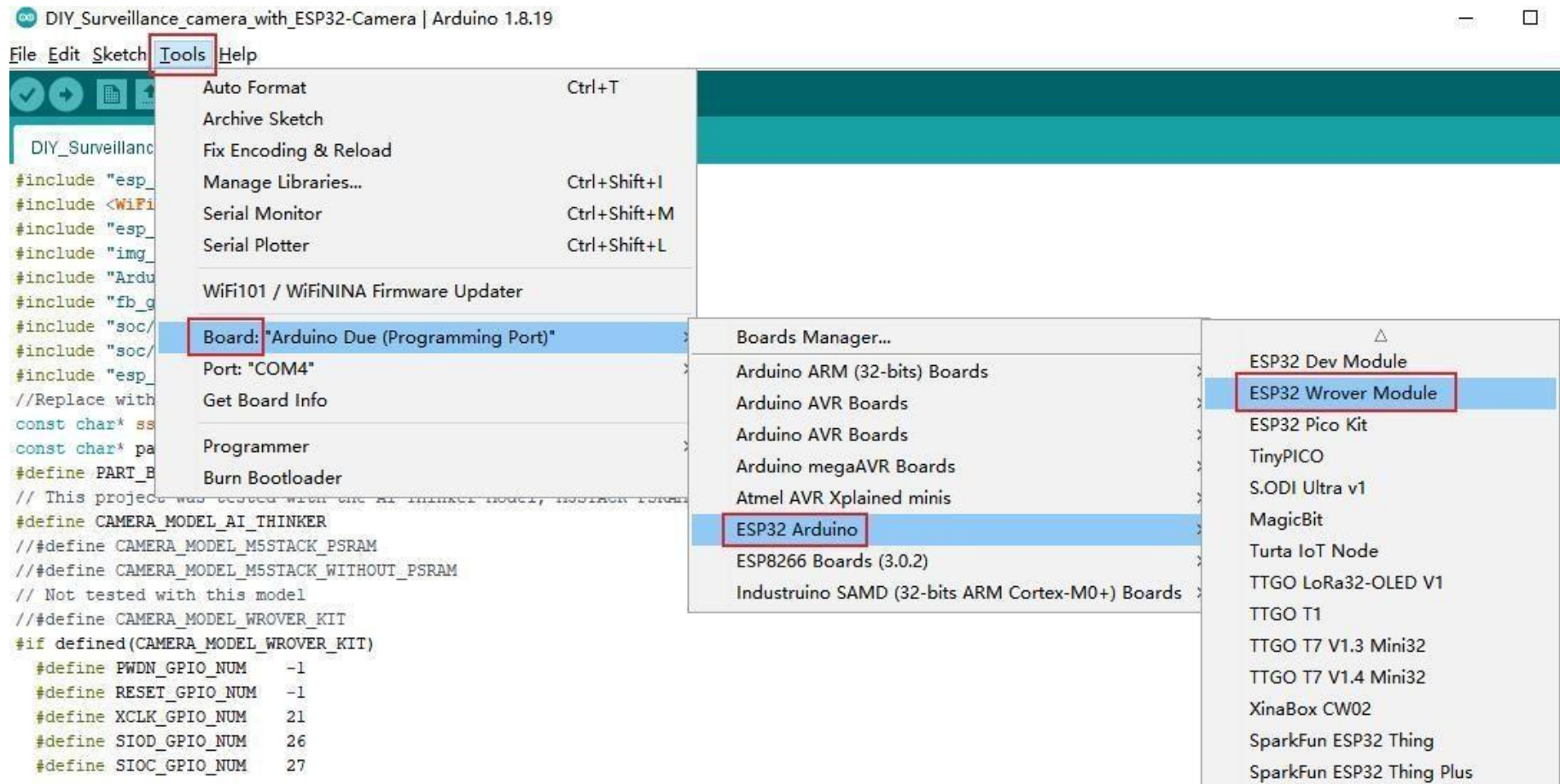
Now, the code is ready to be uploaded to the ESP32-Cam module.

### ESP32-CAM code upload:

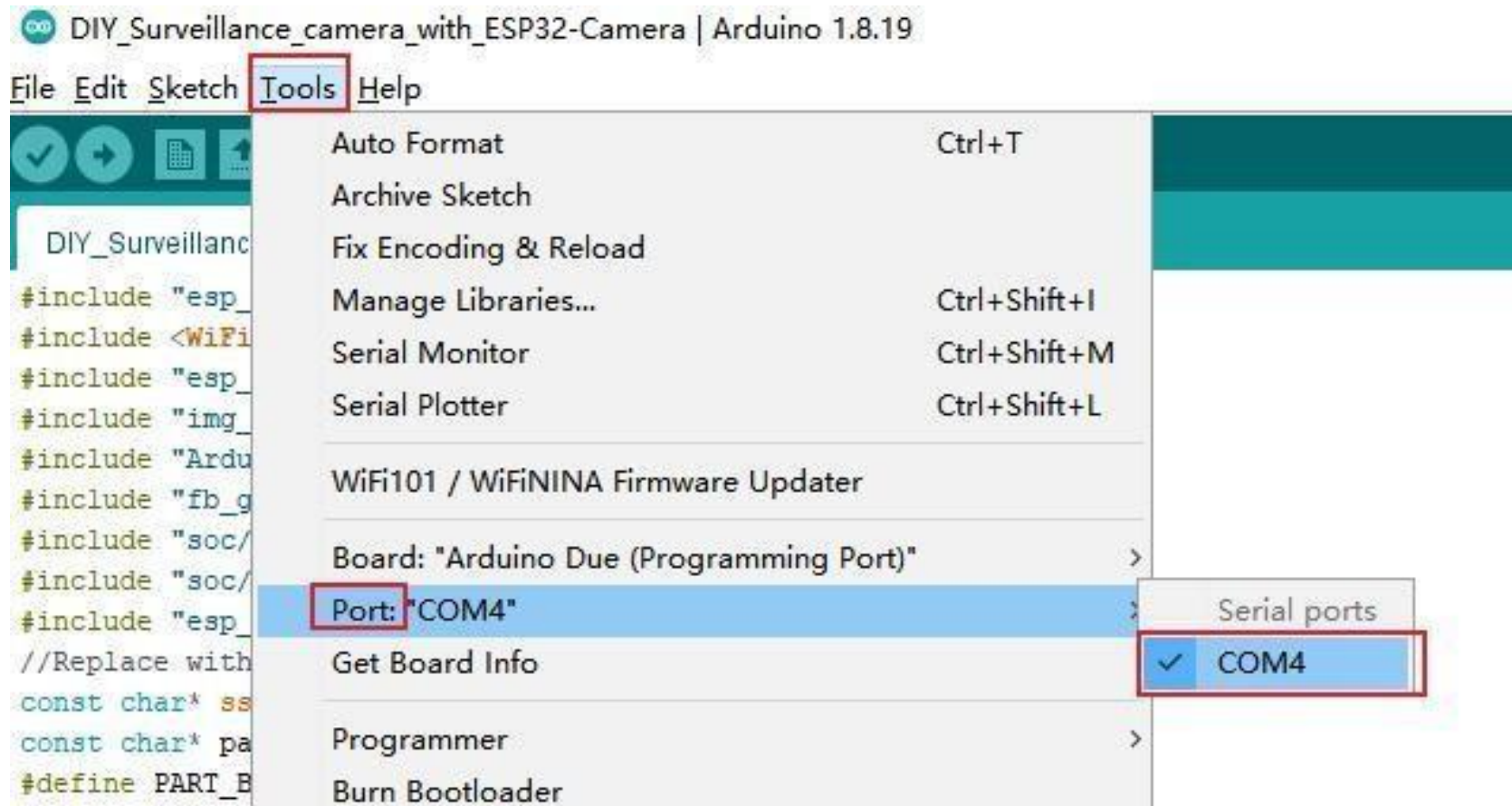
Connect the ESP32-CAM development board to the computer through the type-C port of the expansion board . The expansion board is shown in the following figure:



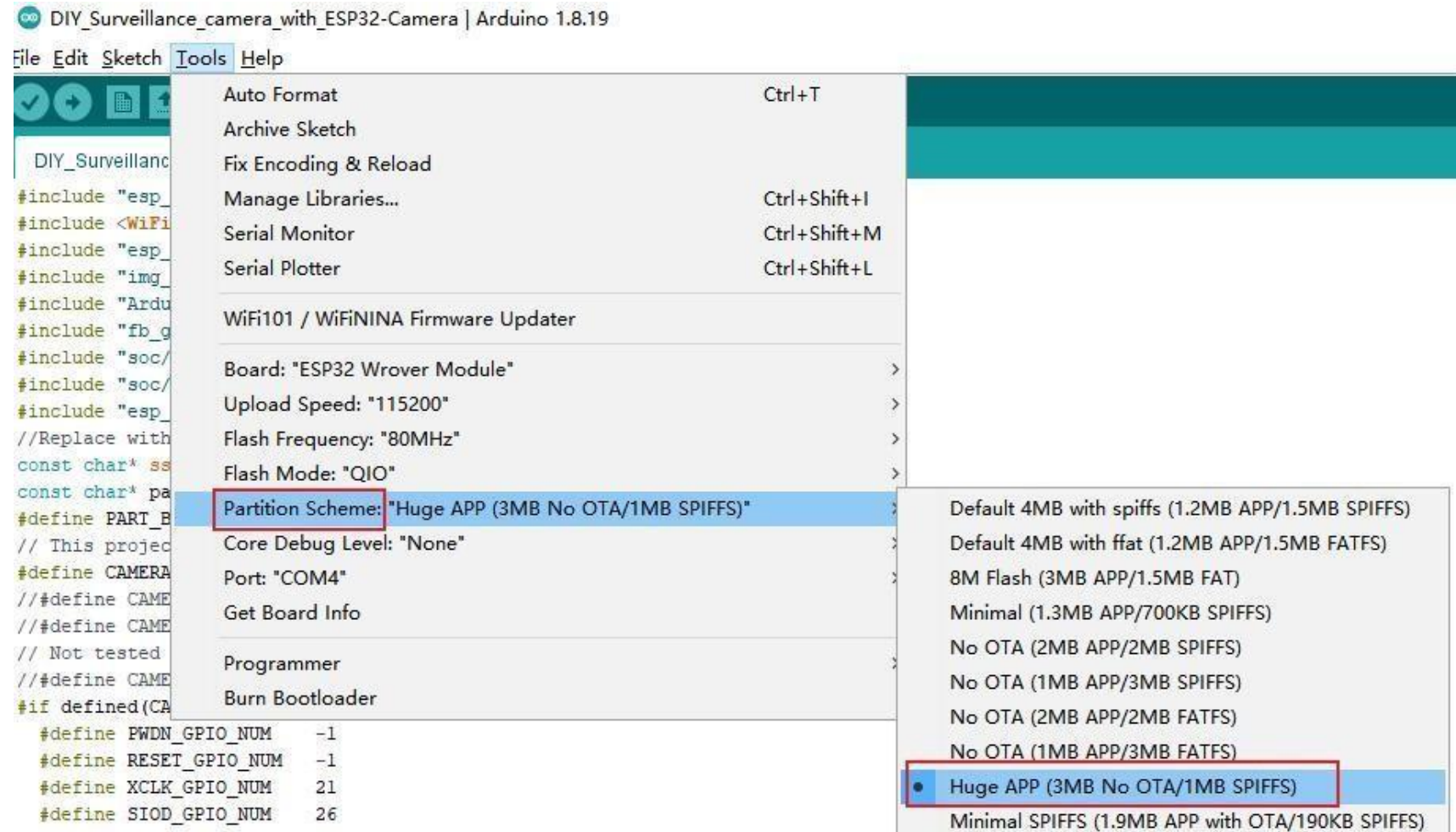
Connect ESP32-CAM to computer, go to Tools > Board > ESP32 Arduino in Arduino IDE , select ESP32 Wrover Module



Go to Tools > Port and select the COM port to which the ESP32 is connected (the COM number that everyone can choose is not necessarily the same, you may be COM3 or others)



In Tools > Partition Scheme , select " Huge APP (3MB or OTA)"

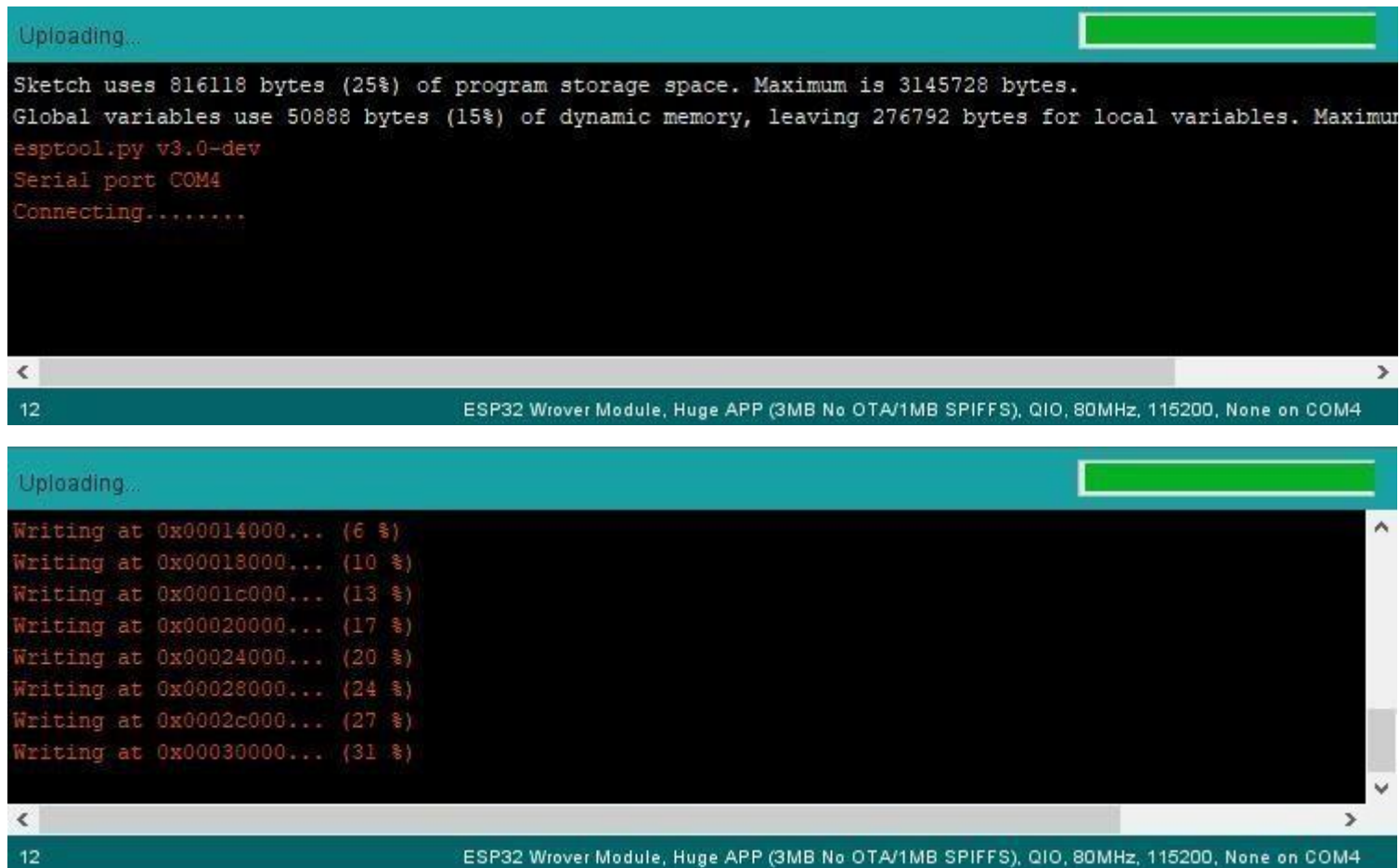


Click the "Upload" button to burn the code ( pull out the TX/RX lines in advance )



You can see the code burning situation at the bottom of the window





```
Uploading...  
Sketch uses 816118 bytes (25%) of program storage space. Maximum is 3145728 bytes.  
Global variables use 50888 bytes (15%) of dynamic memory, leaving 276792 bytes for local variables. Maximum  
esptool.py v3.0-dev  
Serial port COM4  
Connecting.....  
  
12 ESP32 Wrover Module, Huge APP (3MB No OTA/1MB SPIFFS), QIO, 80MHz, 115200, None on COM4
```

```
Uploading...  
Writing at 0x00014000... (6 %)  
Writing at 0x00018000... (10 %)  
Writing at 0x0001c000... (13 %)  
Writing at 0x00020000... (17 %)  
Writing at 0x00024000... (20 %)  
Writing at 0x00028000... (24 %)  
Writing at 0x0002c000... (27 %)  
Writing at 0x00030000... (31 %)
```

```
12 ESP32 Wrover Module, Huge APP (3MB No OTA/1MB SPIFFS), QIO, 80MHz, 115200, None on COM4
```

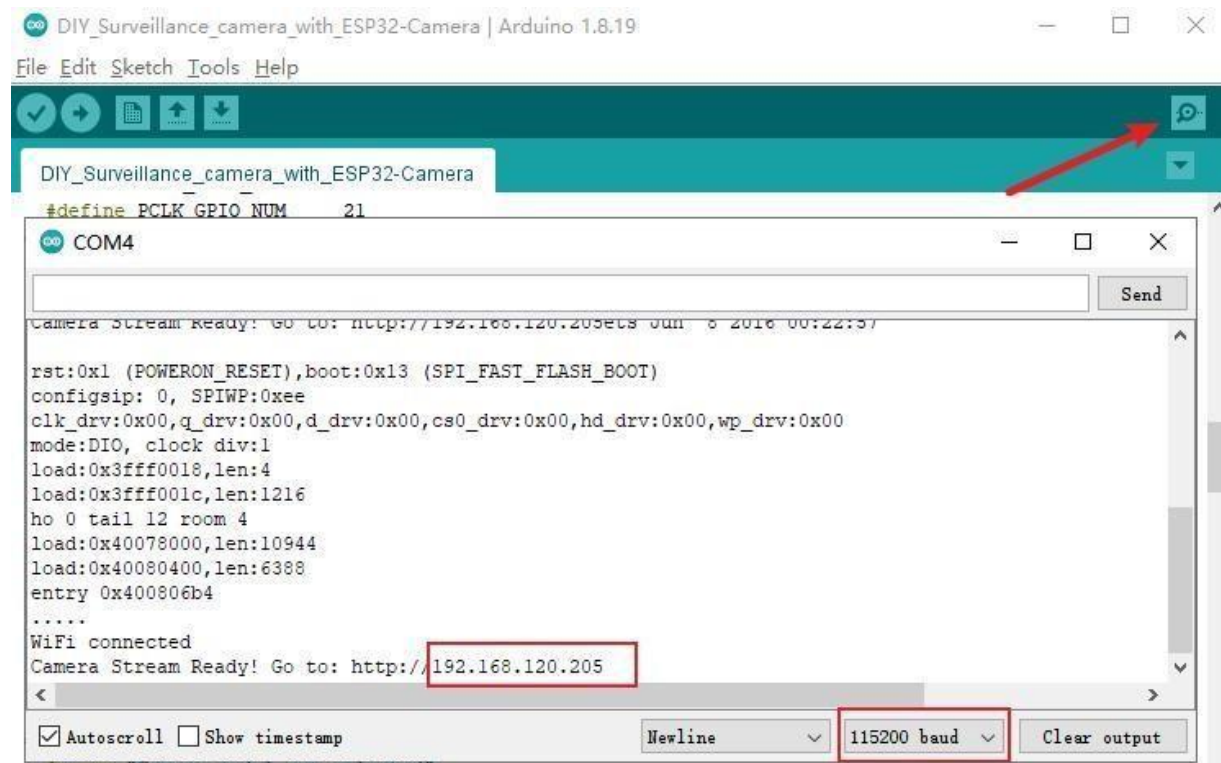
After a few seconds, the code should be successfully uploaded to your ESP32-CAM board .

Once the code is successfully uploaded, the ESP32-CAM will reset.

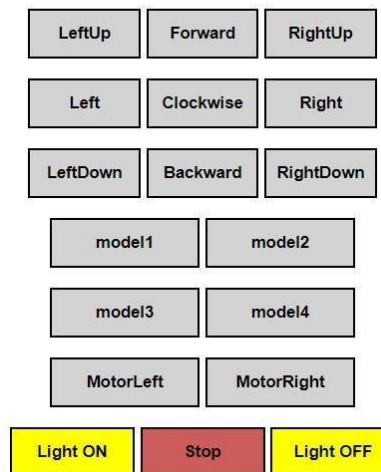
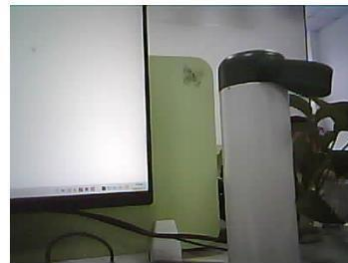
serial monitor in the upper right corner at the baud rate of 115200 and press the reset button on the ESP32-CAM expansion board . When the ESP32-CAM is successfully connected to the WiFi, you can see the http address. Everyone gets the same address. Here it is "192.168.120.205", remember it.



**Videostreaming server:**



Now you can access the camera streaming server on your local network. Connect the device to your own WiFi (same local area network ), open your browser, enter the IP address of ESP32-CAM "192.168.120.205", and click OK to load the video streaming and control page. (Please check that the WiFi signal is stable enough)





## functional combination

We have learned different functional modes in the previous course, now we need to combine them and combine them with the WIFI function of ESP32-CAM for remote control. Although we have realized the camera shooting and the button control display screen, we can't operate the car to make the corresponding action, and we also need to write a program for the car's main control board.

Open the code file (path: 4\_Arduino\_Code\6.2\_Arduino\_UNO\6.2\_Arduino\_UNO.ino)

Connect the mainboard to the computer, select the mainboard type as Arduino uno, then select the COM serial port number, and click the upload button to upload the code (unplug the TX/RX wires in advance, and then plug them back in after the upload is complete)

The code file needs to receive the operation instructions transmitted by ESP32-CAM to execute the corresponding action, so there will be no response immediately after uploading the code and turning on the power. You also need to run the previous set of code (6.1\_ESP32\_Car. It can only work effectively after WiFi.

## code analysis

Define the received data packet array, define the control mode parameters and initialize the angle of the servo motor, etc.

```
byte RX_package [ 3 ] = { 0 };  
uint16_t angle = 90 ;  
byte order = Stop;  
char model_var = 0 ;  
int UT_distance = 0 ;
```

Initialize setting delay and baud rate to prevent data loss

```
Serial.setTimeout ( 10 ) ;  
Serial.begin ( 115200 ) ;
```

## Mode 1 (model1\_func) is free control

Control the movement of the car according to the received signal, Motor() is the movement of the car, and motorleft and motorright are the rotation of the servo motor respectively.

```
void model1_func(byte orders)  
{  
    switch (orders)
```

```
{
case Stop:
    Motor(Stop, 0);
    break;
case Forward:
    Motor(Forward, 250);
    break;
case Backward:
    Motor(Backward, 250);
    break;
case Turn_Left:
    Motor(Turn_Left, 250);
    break;
case Turn_Right:
    Motor(Turn_Right, 250);
    break;
case Top_Left:
    Motor(Top_Left, 250);
    break;
case Top_Right:
    Motor(Top_Right, 250);
    break;
case Bottom_Left:
    Motor(Bottom_Left, 250);
    break;
case Bottom_Right:
```

```

    Motor(Bottom_Right, 250);
    break;
case Clockwise:
    Motor(Clockwise, 250);
    break;
case MotorLeft:
    motorleft();
    break;
case MotorRight:
    motorright();
    break ;
default :
// Serial.println(".");
order = 0 ;
    Motor (Stop, 0 );
    break ;
}
}

```

**Mode 2 (model2\_func) is the obstacle avoidance mode.**

Ultrasonic measurement of the front distance and save it to UT\_distance

```

UT_distance = SR04 ( Trig_PIN, Echo_PIN);
Serial.println ( UT_distance );

```

Because the connection between the motherboards always maintains data communication this time, the delay() waiting function cannot be simply used, which will cause the data to be interrupted during the waiting period. To this end, modify the original obstacle avoidance program to a " for " loop, keep receiving data in the middle: RXpack\_func(), and determine whether to switch the control mode: if(model\_var != 1), part of the code is as follows:

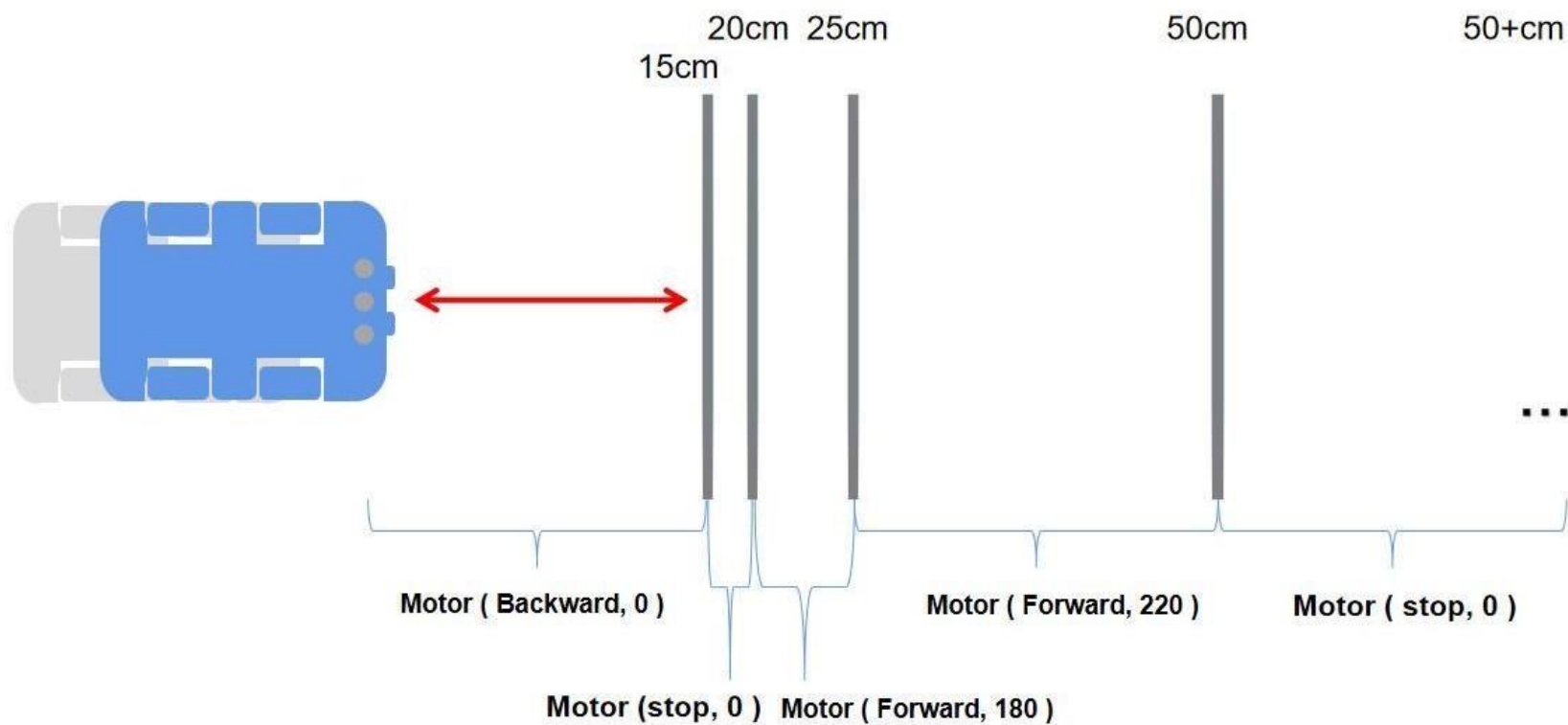
```
if (UT_distance <= 25)
{
    if (UT_distance <= 15){
        Motor(Stop, 0);
        MOTORservo.write(90);
        Serial.print("UT_distance: ");
        Serial.println(UT_distance);

        for(int i = 0; i < 300; i++){
            delay(1);
            RXpack_func();
            if(model_var != 1)
                return ;
        }
    }
}
```

.....

### Mode 3 (model3\_func) for car following

Also obtain the distance measured by the ultrasonic wave in front, and make corresponding movements according to the distance, such as stop, slow follow, fast follow or back. The schematic diagram is as follows:



```
void model3_func()    // follow model
{
    MOTORservo.write(90);
    UT_distance = SR04(Trig_PIN, Echo_PIN);
    Serial.println(UT_distance);
    if (UT_distance < 15)
    {
        Motor(Backward, 200);
    }
    else if (15 <= UT_distance && UT_distance <= 20)
    {
        Motor(Stop, 0);
    }
    else if (20 <= UT_distance && UT_distance <= 25)
    {
        Motor(Forward, 180);
    }
    else if ( 25 <= UT_distance && UT_distance <= 50 )
    {
        Motor (Forward, 220 );
    }
    else
    {
        Motor (Stop, 0 );
    }
}
```

## Mode 4 (model4\_func) for car tracking

Different moving speeds are set here. You can adjust the walking speed during tracking by modifying the speed parameters in the motor to achieve the desired effect. Part of the code is as follows:

```
void model4_func()    // tracking model
{
    MOTORservo.write(90);
    Left_Tra_Value = analogRead(LEFT_LINE_TRACJING);
    Center_Tra_Value = analogRead(CENTER_LINE_TRACJING);
    Right_Tra_Value = analogRead(right_LINE_TRACJING);
    if (Left_Tra_Value < Black_Line && Center_Tra_Value >= Black_Line && Right_Tra_Value < Black_Line)
    {
        Motor (Forward, 250 );
    }
    .....
}
```

The encapsulation functions of servo motor, motor, ultrasonic and receiving data are as follows



```

339 void motorleft() //servo
340 > { ...
345 }
346 void motorright() //servo
347 > { ...
352 }
353
354 void Motor(int Dir, int Speed) // motor drive
355 > { ...
363 }
364
365 float SR04(int Trig, int Echo) // ultrasonic measured distance
366 > { ...
376 }
377
378 void RXpack_func() //Receive data
379 > { ...
461 }

```

The motorleft() function controls the left turn of the camera servo motor, and the value in MOTORservo.write() controls the speed. When the range is from 100 to 170, it represents the left turn, and here it is 120. When the value ranges from 10 to 80,

it indicates a right turn. The smaller the value, the faster the speed. The larger the value, the slower the speed. When the value is 90, the servo motor stops. The larger the value of the delay() function, the longer the execution time.

```
void motorleft() //servo
{
    MOTORservo.write(120);
    delay(10);
    MOTORservo.write(90);
    delay(50);
}
```

Loop function, execute data receiving function and switch between different modes

```
void loop ()
{
    RXpack_func ();
    switch (model_var)
    {
        case 0 :
            model1_func (order); //Free control model
            break;
        case 1:
            model2_func();      // OA (Obstacle avoidance) model
            break;
    }
}
```

```
case 2:
    model3_func();    // follow model
    break;
case 3:
    model4_func();    // Tracking model
    break;
}
}
```

