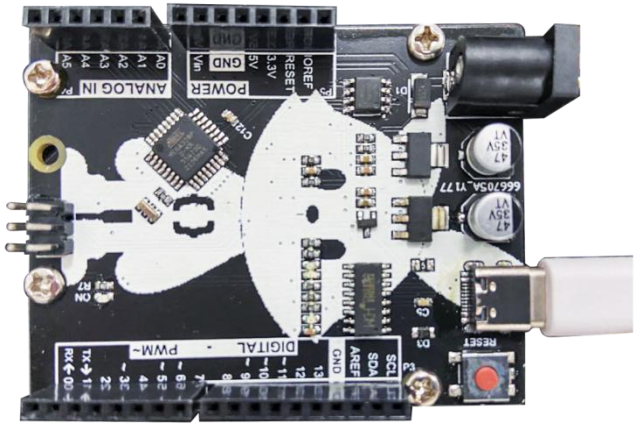
In the "Tutorial Documentation" PDF file, we have learned how to successfully realize the function of camera recording and remote control car movement through WiFi connection. This document is used as a reference document to guide you to understand the principle process of function realization.

content

1. The first program code - Blink

1.1 Description

In this section, we start from understanding the TSCINBUNY control board, make a test program, and learn the use of Arduino IDE, code upload and simple programming methods.



1.2 Start the first test program

Make your own "Blink" sketch

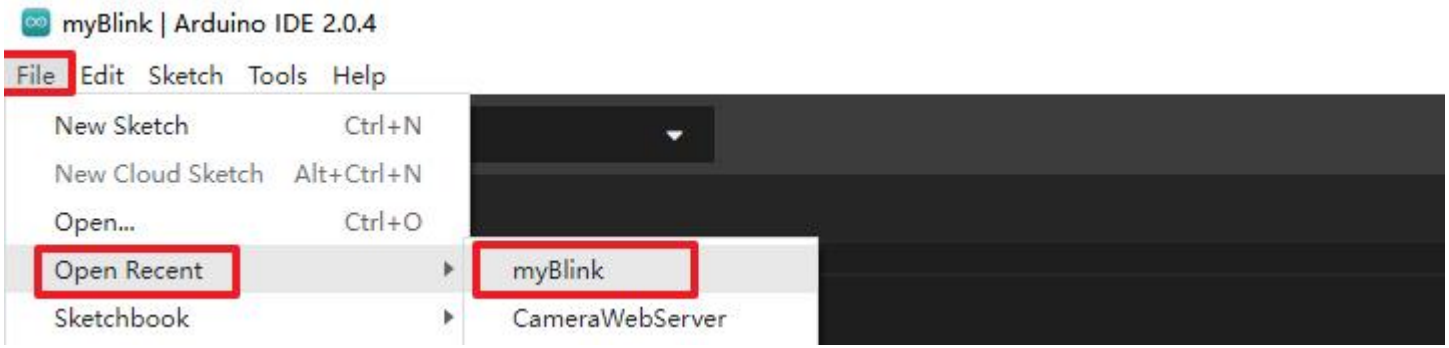
Here we'll reprogram the board with our own Blink sketch, and then change its blink rate. Now keep the board connected to the computer, set up the Arduino IDE and make sure you can find the correct serial port , and upload the program to test.

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 .

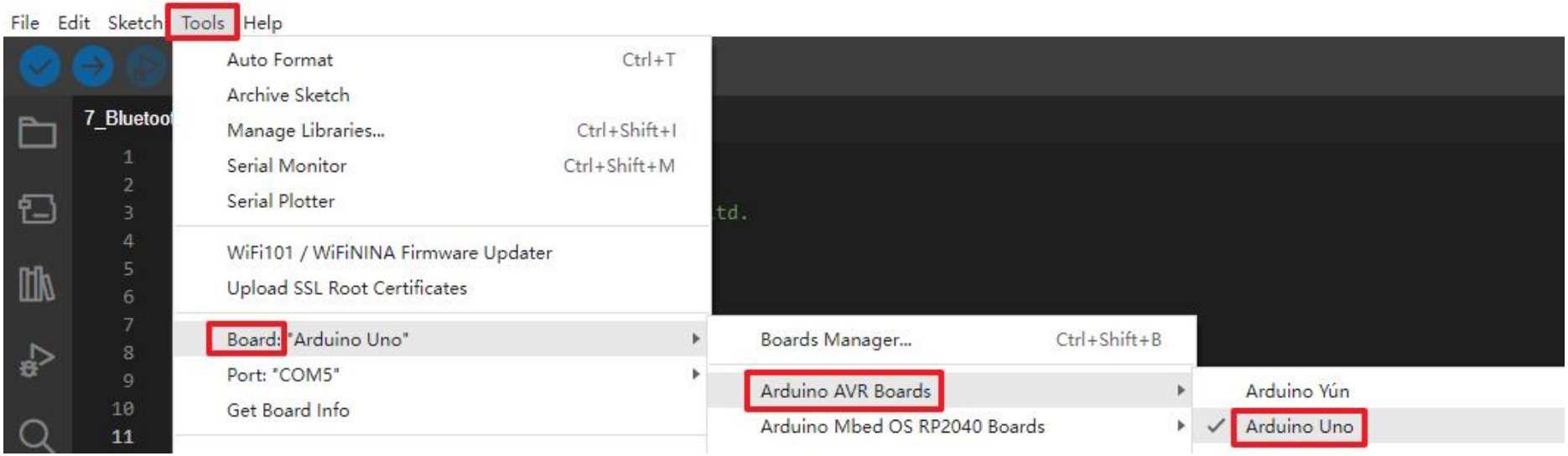
The example sketches included with the Arduino IDE are "read only". That is, you can upload them to the dashboard , but if you change them, you can't save them as the same file. So the first thing you need to do is save your own copy.

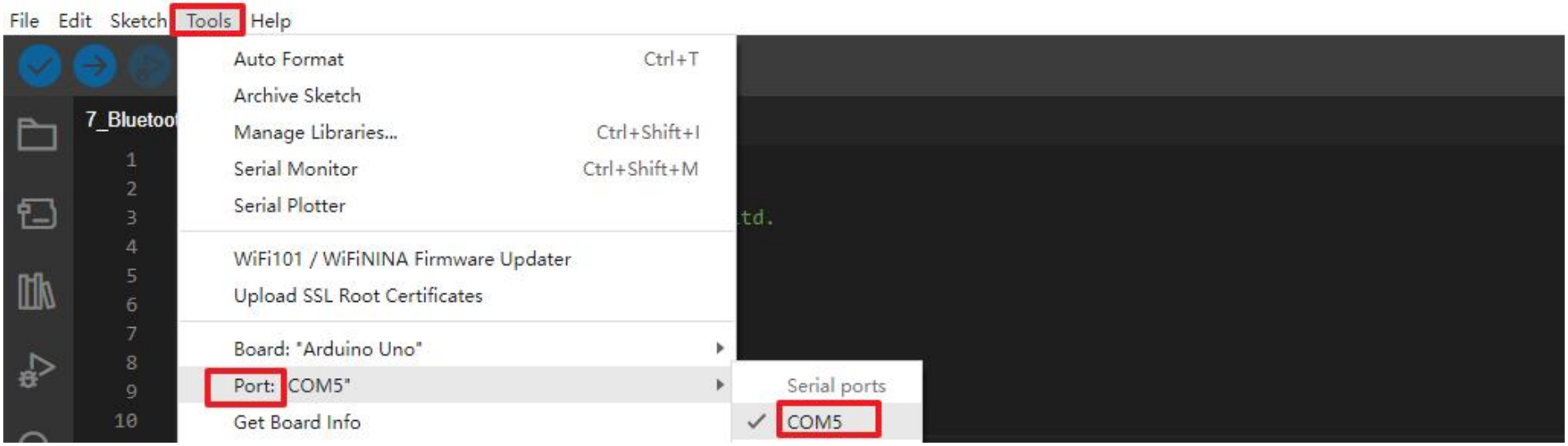
From the Arduino IDE's File menu, select "Save As.." and save the sketch as " myBlink ".

You've saved a copy of the "flicker" in your sketchbook, if you ever want to find it again, just open it using the " File > Open Recent " menu option.



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

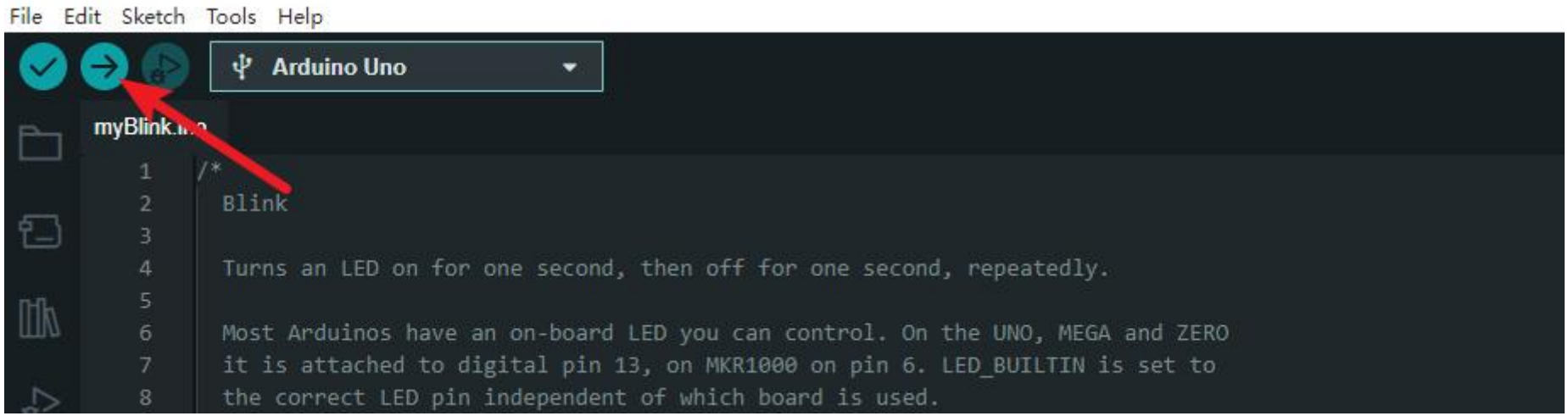

File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter
WiFi101 / WiFiNINA Firmware Updater
Upload SSL Root Certificates
Board: "Arduino Uno"
Port: "COM5"
Get Board Info
Boards Manager... Ctrl+Shift+B
Arduino AVR Boards
Arduino Mbed OS RP2040 Boards
Arduino Yún
Arduino Uno


File Edit Sketch Tools Help
Auto Format Ctrl+T
Archive Sketch
Manage Libraries... Ctrl+Shift+I
Serial Monitor Ctrl+Shift+M
Serial Plotter
WiFi101 / WiFiNINA Firmware Updater
Upload SSL Root Certificates
Board: "Arduino Uno"
Port: COM5"
Get Board Info
Serial ports
COM5

Note: The board type here is Uno and the serial port is COM5 .

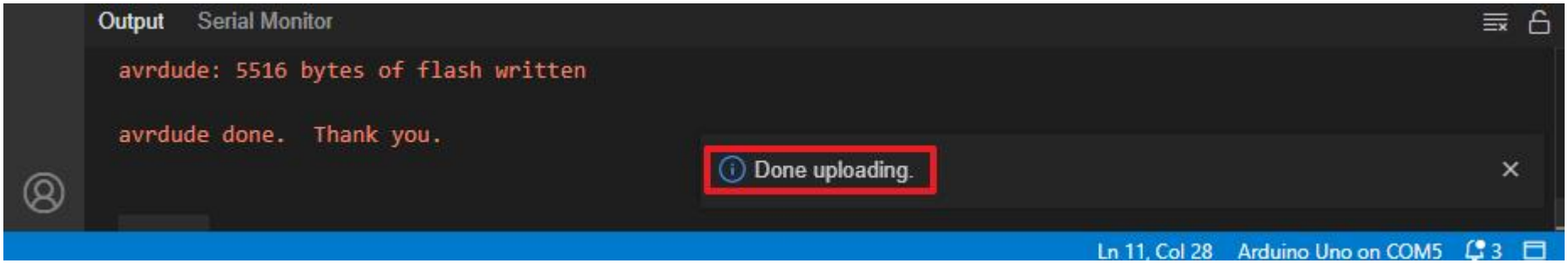
Actually everyone 's serial port display will be different, although COM 5 is selected here , it may be COM3 or COM4 on your computer. A correct COM port should be COM X (arduino X XX) standard.

After clicking the "Upload" button, the program starts to upload. At this time, the LED on the Arduino will start to blink as the sketch is transferred.



The transfer is complete and "Upload complete"

appears



"Compile Sketch.." process , you might get an error message:

This could mean that your board is not connected at all, or that the CH340 driver is not installed (if required) or that the serial port is selected incorrectly . If you encounter this situation, please check your IDE settings and motherboard connection, and ask business personnel for help after taking a screenshot .

When the upload is complete , the board LED should reboot and start blinking. Note that a large part of this sketch consists of annotations. These are not actual program instructions; rather, they explain how to make the program work. They are there for your easy readability . Everything between " /* " and " */ " at the top of the sketch is a block comment ,

which explains the purpose of the sketch.

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

The first part of 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 needs 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 powering on for the first time or after uploading a sketch .

The next step is to name the pin and set the output, here set " LED_BUILTIN " 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 it is not necessary to rename it to pin 13 for direct use.

The sketch must also have a " loop " function. Unlike the "Set" function, which only runs once, after a reset, the "Loop" function will restart immediately after finishing 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 level), then "delays for 1000 milliseconds (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 changing the parameters in " delay ()".

```
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
}
```

This delay time 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 on each delay, instead of a second. Upload the sketch again and you should see the "LED" start blinking faster .

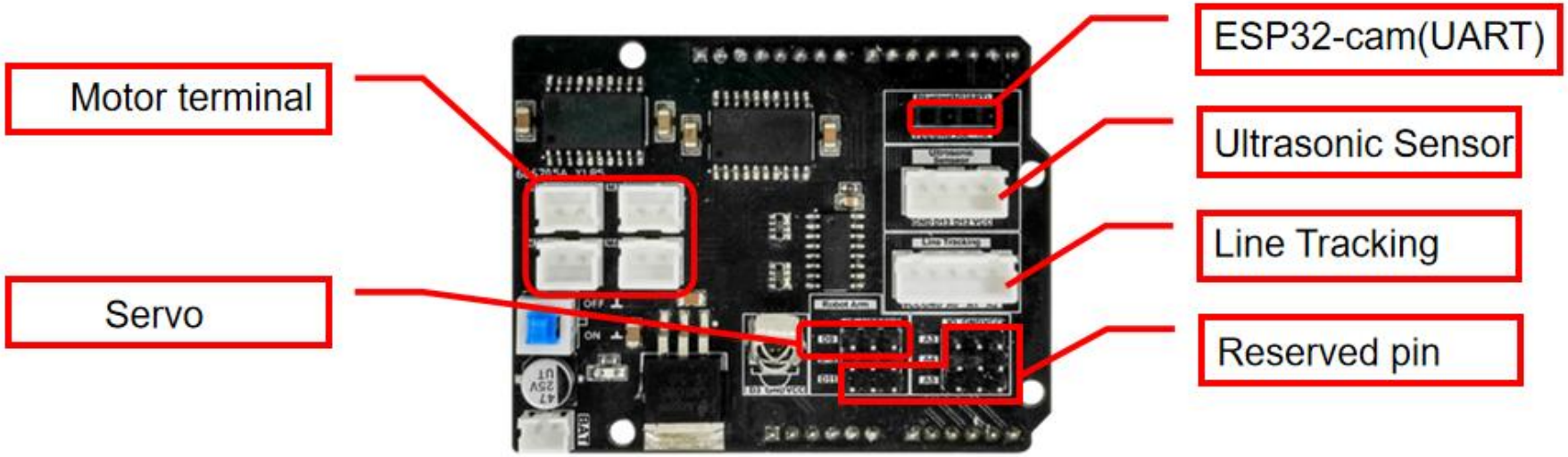
So far, you have understood and mastered the basic knowledge of Arduino programming and the basic steps of downloading the program.

2. Expansion Board and Motor Driver

2.1 Description

The content of this section is mainly to understand the knowledge related to high-performance expansion boards and motor drives

Expansion board introduction

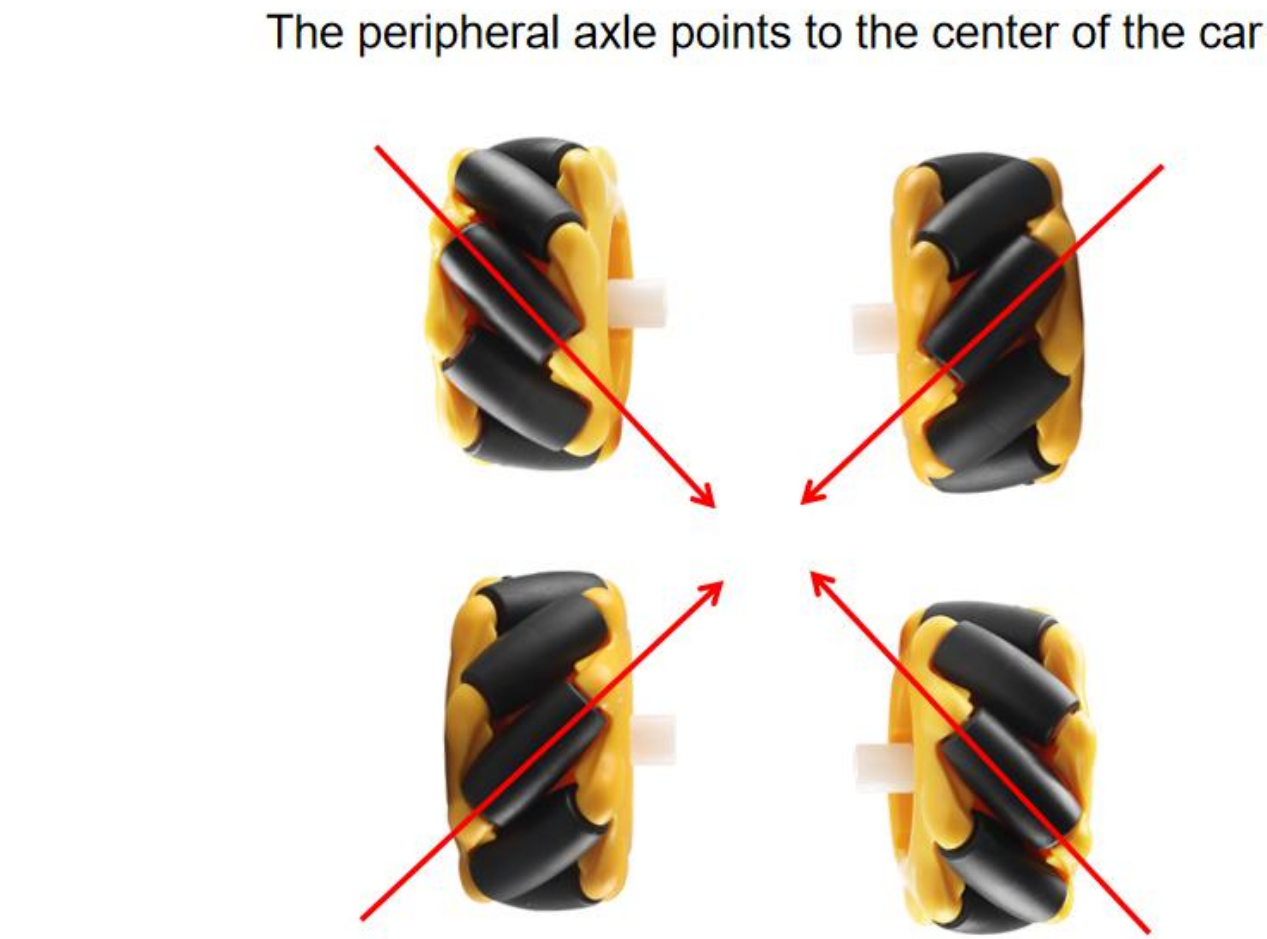


The expansion board extends the pins of the main board very well, and the motor, ultrasonic and tracking modules are all connected by terminal plug-in, which increases the firmness and reliability. The switch-controlled BAT port is reserved, and the reserved infrared receiver is stably integrated on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

Mecanum Wheel:

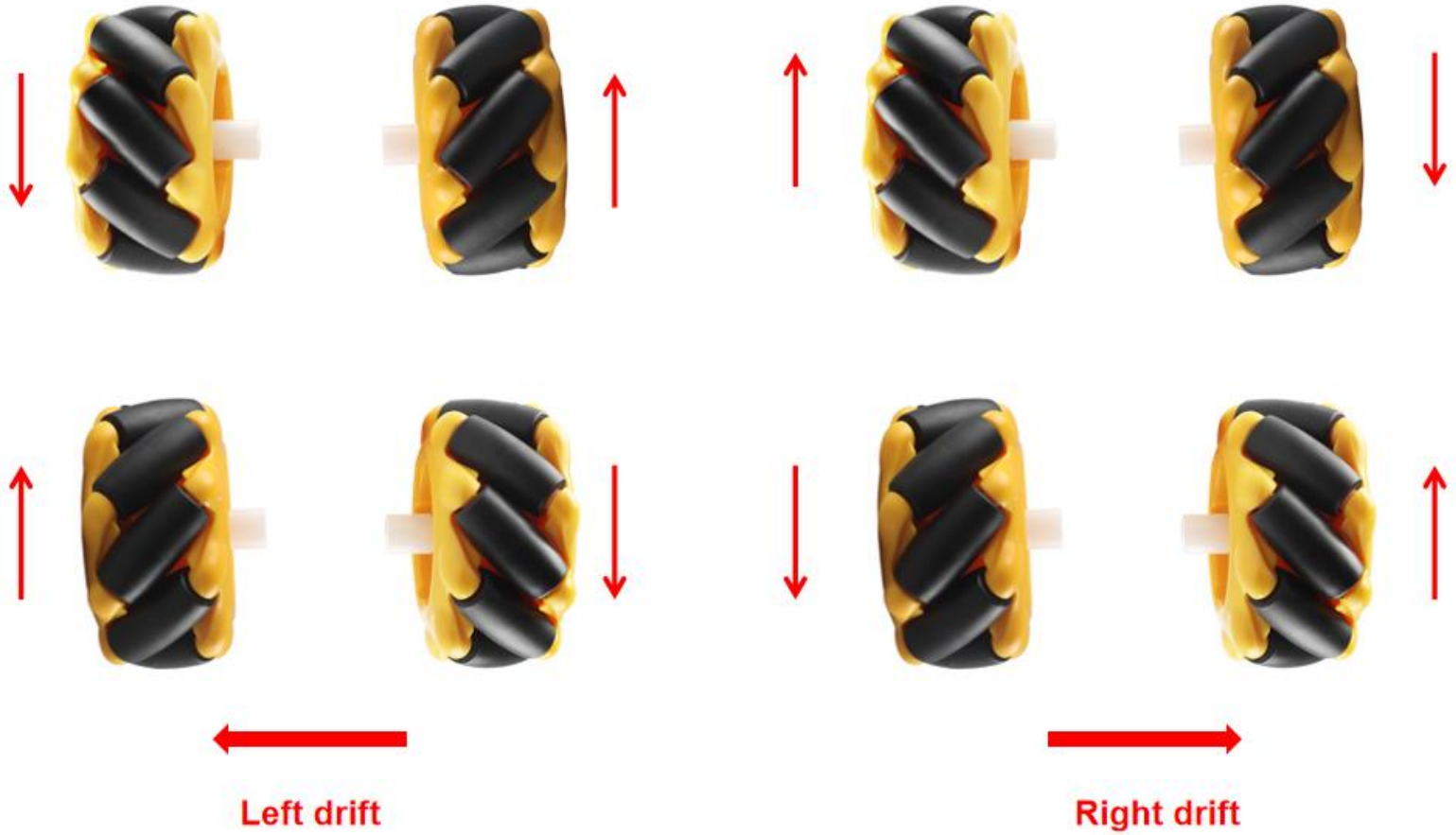
A mecanum wheel is a wheel with a peripheral axle, generally divided into two types, one with a peripheral axle that is inclined to the left, and the other with an axle that is inclined to the right. These angled peripheral axles convert part of the wheel steering force into a wheel normal force, so that the car can achieve side-to-side translational motion.

Assembly points (top view):

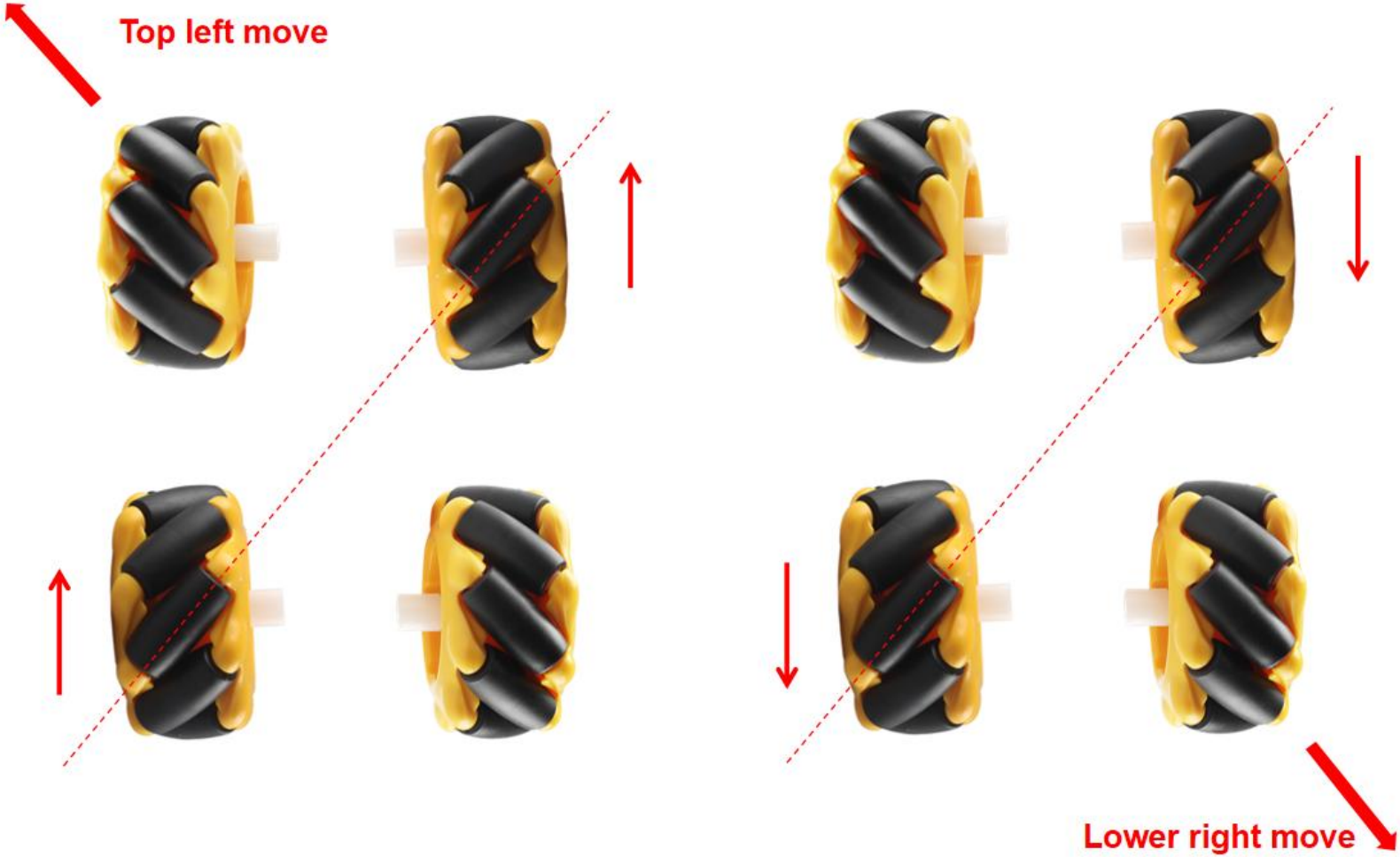


Movement principle:

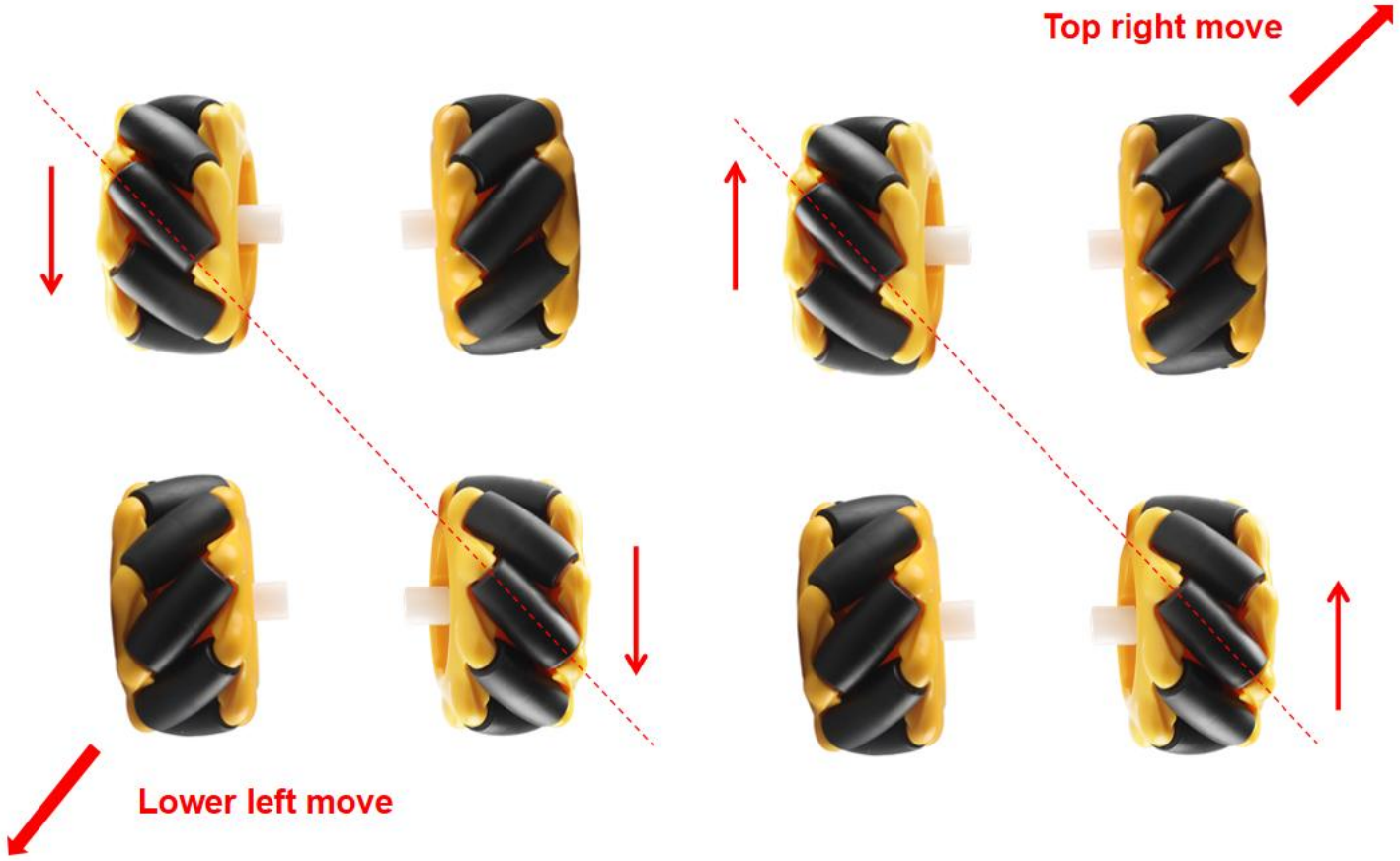
Different rotation directions of the wheel correspond to left and right translational movement:



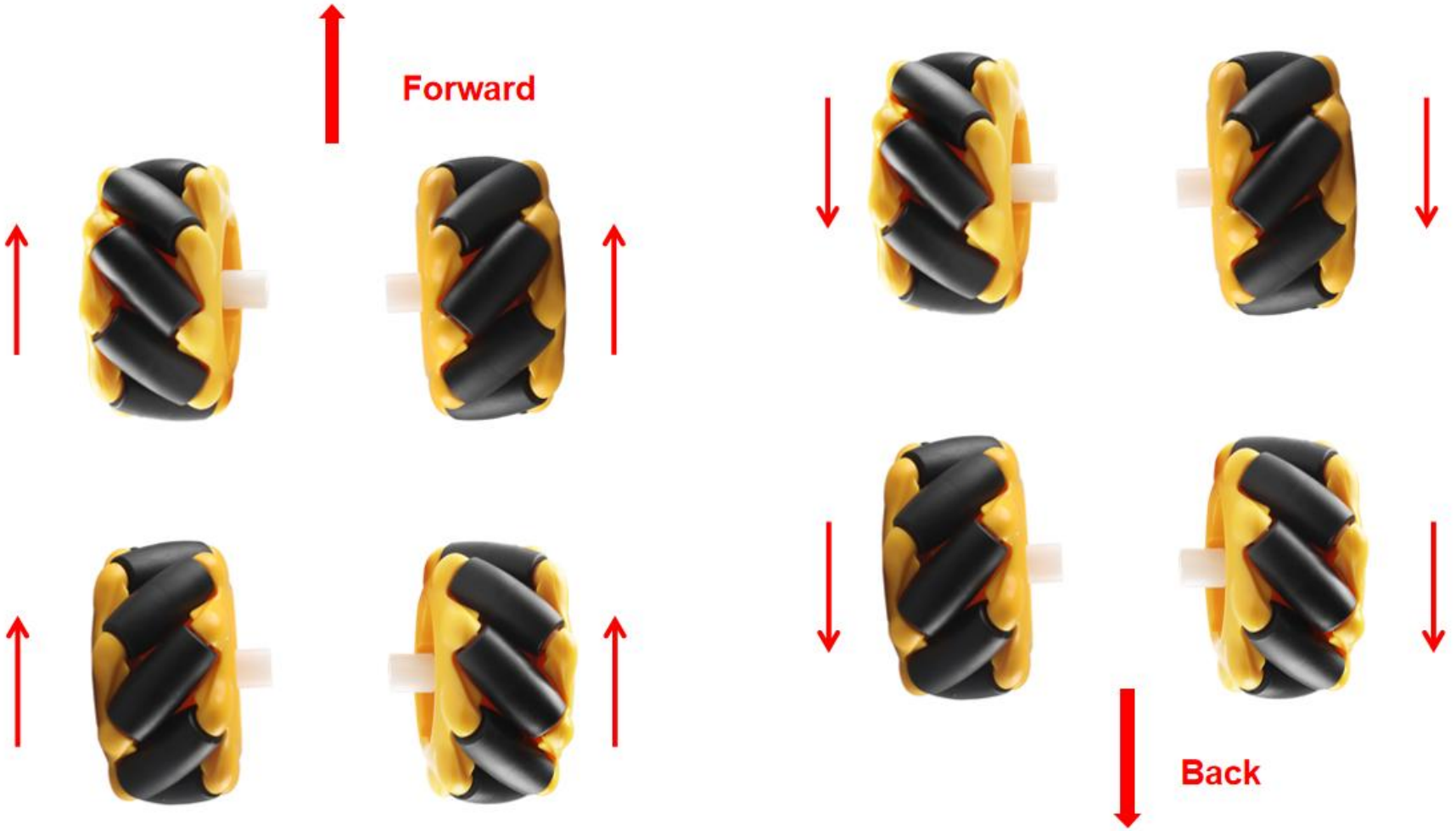
The different rotation directions of the wheel correspond to the upper left direction and the lower right direction:


Top left move
Lower right move

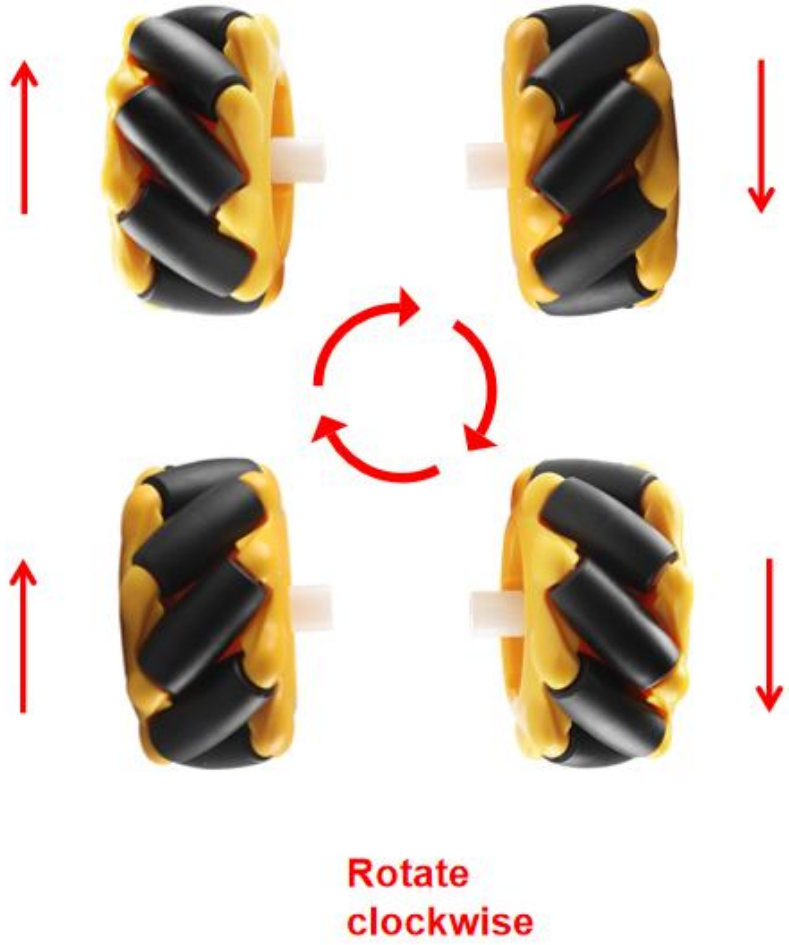
The different rotation directions of the wheel correspond to the movement in the lower left direction and the upper right direction:



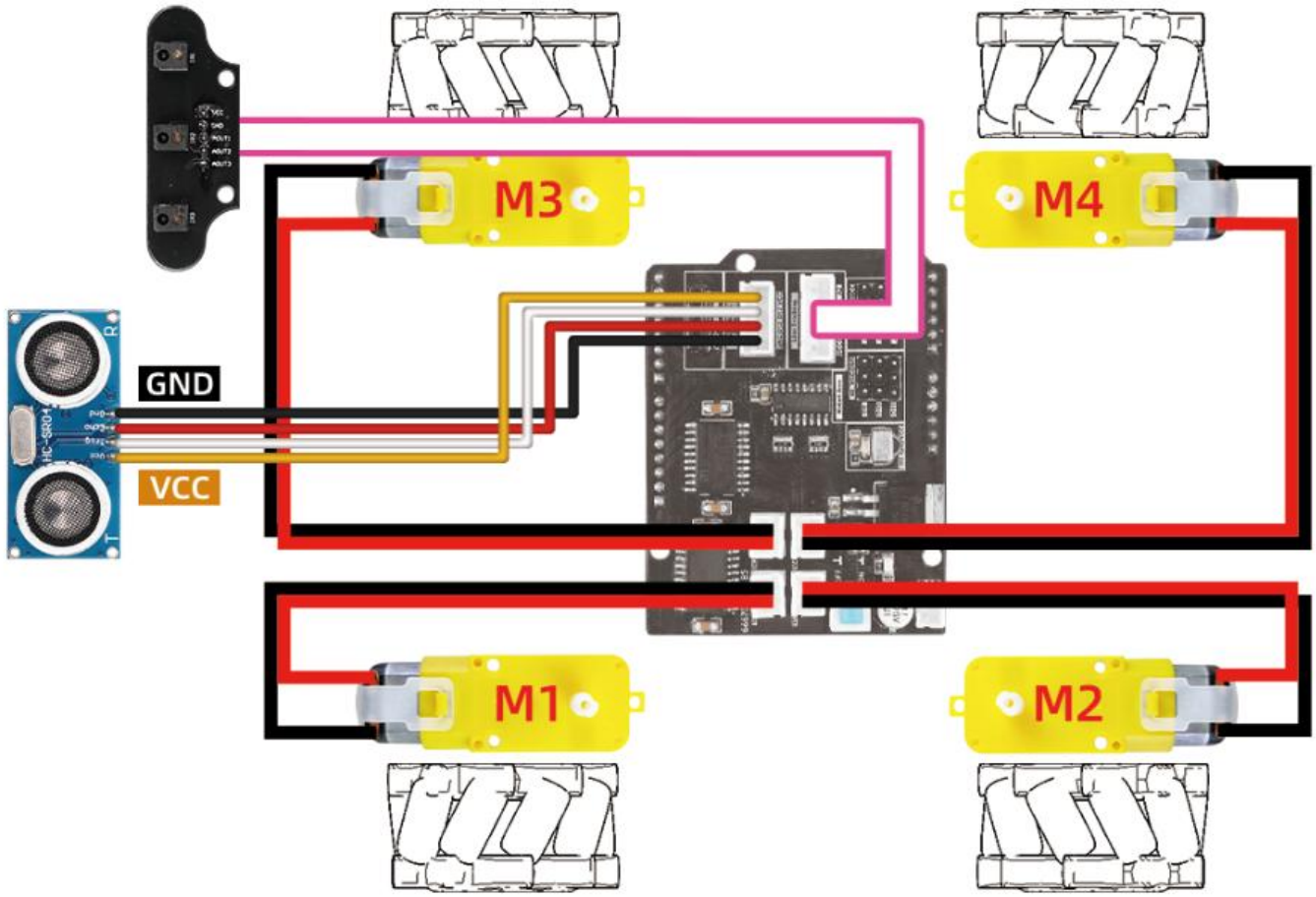
Different rotation directions of the wheels correspond to forward and backward movement:



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



Wiring diagram:



code analysis

Open the program file (path: 2_Arduino Code\1_Auto_move\1_Auto_move.ino)

(Remind again: before uploading the code, please unplug the expansion board VCC/GND/RX/TX four connecting lines, 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.

Arduino's shiftOut function mainly works on 74HC595 chip;

Control the high and low levels of each pin through decimal numbers 0 to 255 for 8-bit binary numbers;

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 number into an 8-bit

binary number to achieve the function of controlling 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
```

Sets the variable that stores 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, two values are passed in, one is the code value to control the rotation direction of the motor 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);
}
```

Methods for debugging 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 value of the variable to 1 (0000 0001 for eight-bit binary), 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 0000 0001 is the code for 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 the 8 states of the 4 motors * 2 rotation modes, positive and negative, are inferred. Summarizing the data records, if you want the car to move forward, you need to keep the four motors in the 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 controlling the same motor cannot be 1 (high level) at the same time, which will cause 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

Pass the value of the rightmost column to the variable Dir of the function shiftOut to get different motion states.

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

3. servo motor

3.1 Description

The content of this section is mainly to understand the attributes and characteristics of the steering gear, 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 mode of the steering gear in Arduino programming.

3.2 Introduction to steering gear

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

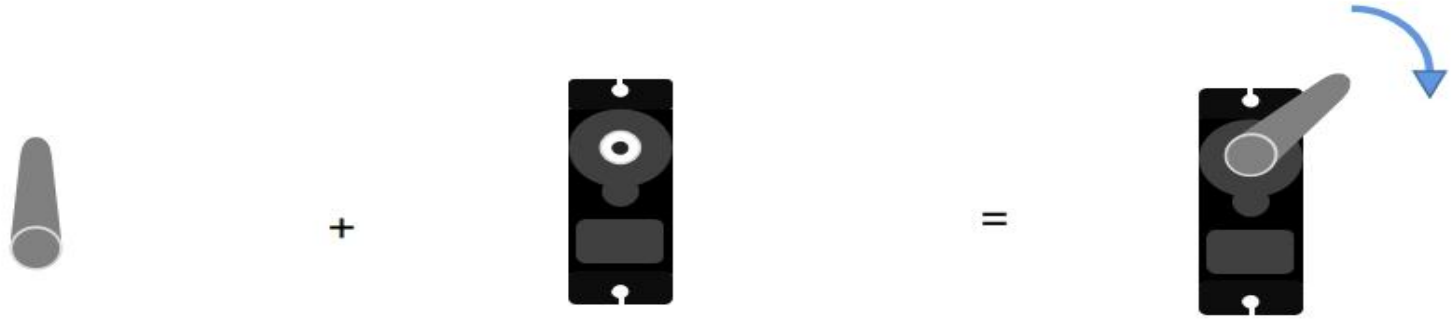
That is to say, if a certain pulse width is provided to the steering gear, its output shaft will maintain a certain corresponding angle. No matter how the external torque changes, it will not change the output angle to a new corresponding position until another pulse signal is provided to it.

There is a reference circuit inside the servo, which generates a pulse signal with a period of 20ms and a width of 1.5ms .

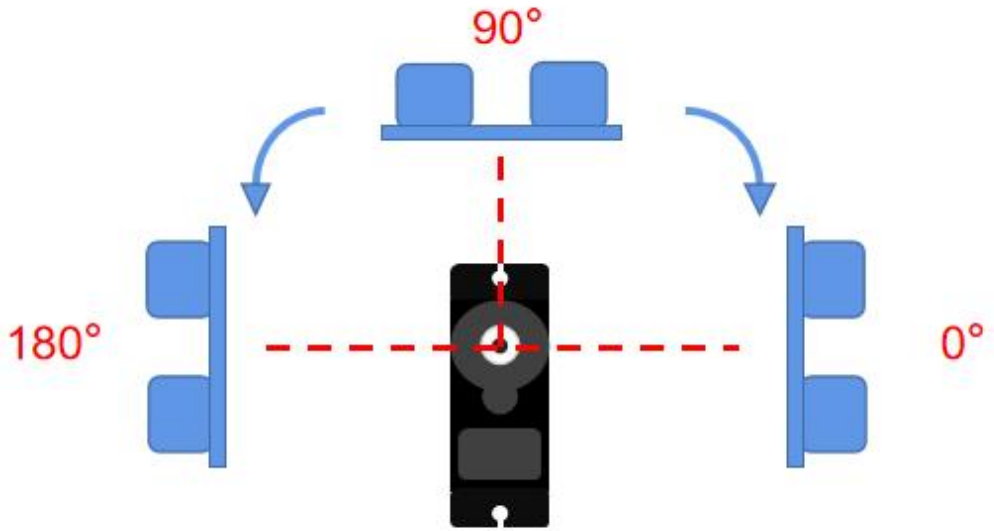
There is a comparator that compares the external signal with the reference signal to determine the direction and magnitude, thereby generating a motor rotation signal. (Note: It is forbidden to artificially twist the servo motor after power on, otherwise it will be easily damaged!)

3.3 Understanding the initial 0° of the servo motor

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

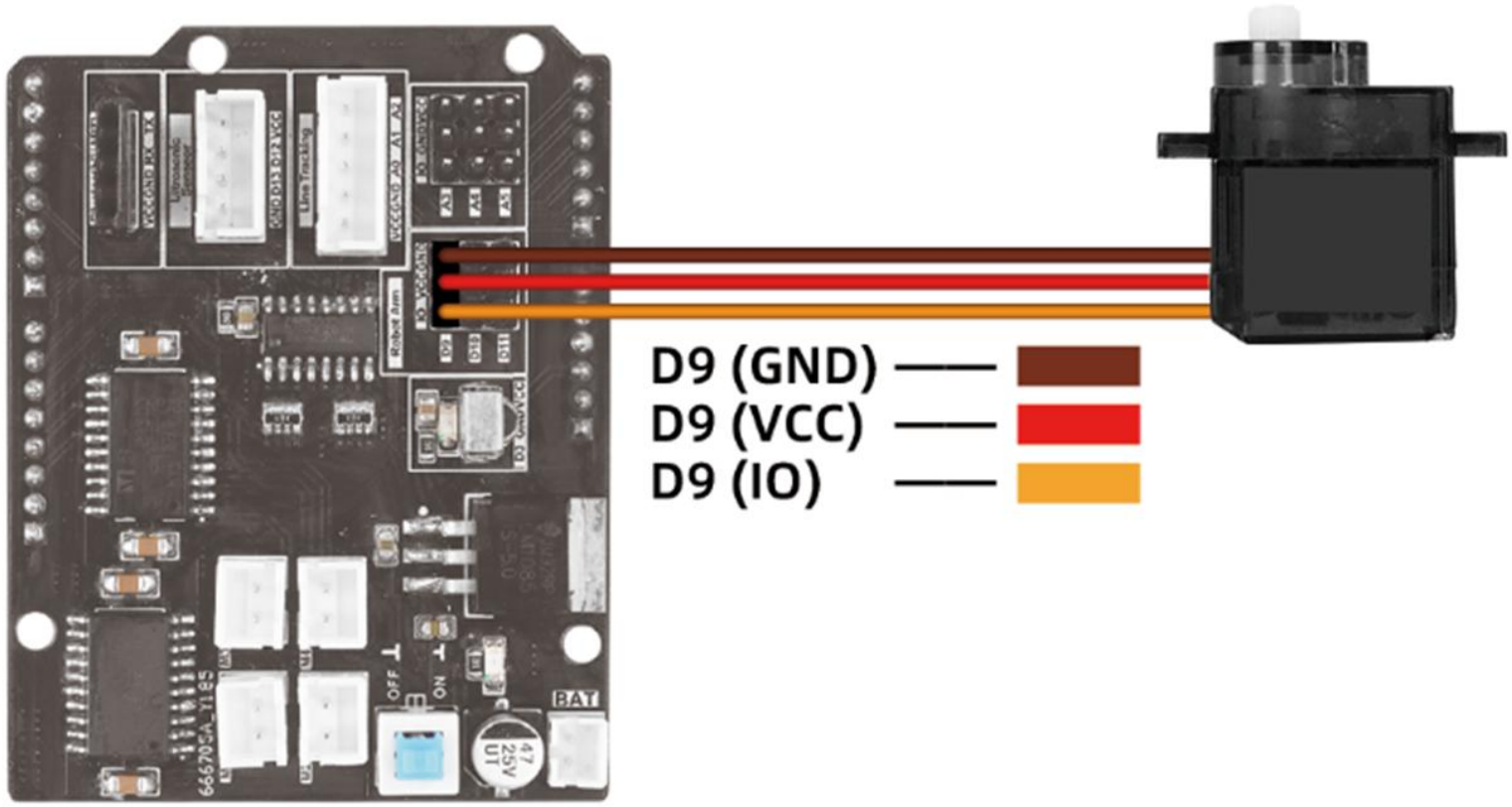


3.4 Servo motor installation method: top view



As shown in the picture above (left), the front of the car faces upwards at 90° . When installing the ultrasonic wave, first make sure that the initial position is on the right . 180° position , slowly turn the central shaft of the servo motor clockwise to the limit position. To ensure that the rotation range of the ultrasonic wave (0~180°) is right , front, and left , the ultrasonic wave should be installed facing the direction of 0°, and then turned 90° counterclockwise to be the front.

3.5 Wiring Diagram



3.6 Code Analysis

Open the code file (path: 2_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 _
```

Define the rotation angle variable

```
int pos = 0 ;
```

Initialize and set the servo motor angle

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

Cyclic execution, the servo motor reciprocates from 10° to 170°

```
void loop ()
{
    for (pos = 170 ; pos >= 10 ; pos -= 1 )
{
```

```
        myservo.write (pos ) ;
        delay ( 15 );
}
    for (pos = 10 ; pos <= 170 ; pos += 1 )
{
        myservo.write (pos ) ;
        delay ( 15 );
}
}
```

4. Ultrasonic ranging

4.1 Description

The content of this section mainly understands the working principle of the ultrasonic module, masters the connection of the ultrasonic circuit diagram, and learns the method of distance measurement of the ultrasonic module through programming.

4.2 Ultrasonic sensor

Sound waves are generated 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 distance measurement. Such as range finder, 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 pulses
Output echo signal	Output TTL level signal, proportional to the range
size	45*20*15

4.3 Ultrasonic ranging is a non-contact detection method

Especially for air distance measurement, due to the slow wave velocity in the air, the echo signal contained along the propagation direction of structural information is easy to detect and has very high resolution, so its accuracy is higher than other methods; while the ultrasonic sensor has a simple structure , Small size, reliable signal processing and so on. The use of ultrasonic detection is often faster, more convenient, simpler to calculate, easier 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 the system in ultrasonic measurement is to detect the transmission time of ultrasonic waves from the ultrasonic transmitter through the gas medium to the receiver. Multiplying this time by the speed of sound in the gas gives 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 emitted in the air, and return immediately when encountering obstacles on the way, and the ultrasonic receiver stops timing immediately after receiving the reflected waves.

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 waves, the texture of the reflector, the angle between the reflected and incident sound waves, and the sensitivity of the receiving transducer. The ability of the receiving transducer to receive the sound pulse directly will determine the minimum measurable distance.

The trigger signal input terminal (TRIG) will input a high - level signal of more than 10 microseconds, and the ultrasonic transmitter will automatically send eight 40Hz square waves 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 .

4.4 Code Analysis

Open the code file (path: 2_Arduino Code\3_Ultrasonic\3_Ultrasonic.ino)

Get distance function checkdistance( )；(Sometimes it takes battery power to measure the value accurately!)

```
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 the transmission to the reception of the ultrasonic wave.

The speed of sound in dry air at 20 degrees Celsius is about 343 m / s , which means that it takes 29.15 microseconds to travel 1 centimeter. It takes double the time to send and receive, so it is about 58.3 microseconds, and the value is 58.

Print ultrasonic measurements to the serial monitor

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

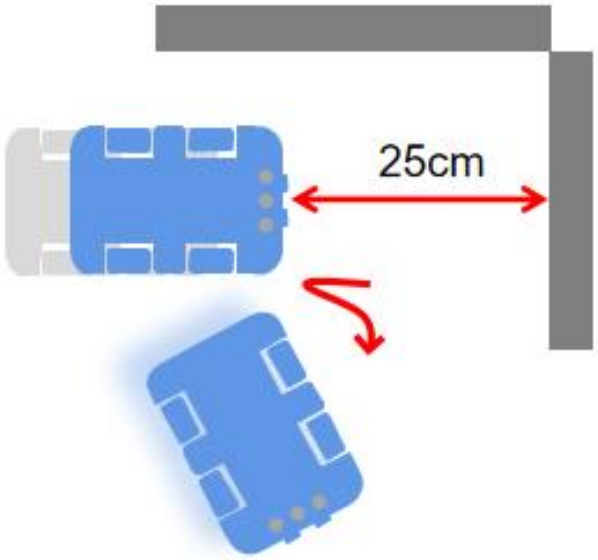
5. Obstacle avoidance car

5.1 Description

This lesson mainly studies and consolidates the practical application of ultrasonic and servo motors. On the basis of the previous lesson, continue to learn the principle of ultrasonic obstacle avoidance and the obstacle avoidance function of the obstacle avoidance car through programming.

5.2 Principle of obstacle avoidance

When the distance detected by the ultrasonic wave is less than the set distance, it is judged that the car has encountered an obstacle, and then the obstacle avoidance program is triggered . The car stops and then the servo motor drives the ultrasonic wave to turn right to obtain the right obstacle distance, and then turn left to obtain the left obstacle distance and save them separately. into the variables "rightDistance" and "leftDistance". Then compare the distance, which side has the greater distance, that is, turn to which side if you are less hindered, so as to realize automatic driving to avoid obstacles .



5.3 Code Analysis

Open the code file (path: 2_Arduino_Code\4_Obstacle_Avoidance\4_Obstacle_Avoidance.ino)

Save the ultrasonic measurement distance obtained by the SR04 function into the variable middleDistance.

```
middleDistance = SR04 (Trig, Echo);
```

The first "if" condition judges that if the distance is less than or equal to 25, the car stops, and then the ultrasonic measurement of the right and left distances is saved to the "rightDistance" and "leftDistance" variables.

```
    if (middleDistance <= 25 )
{
        Motor (Stop, 0 );
        delay ( 500 );
        myservo . write ( 10 );
        delay ( 500 );
        rightDistance = SR04(Trig, Echo);//SR04();

        delay(500);
        myservo.write(90);
        delay(500);
        myservo.write(170);
        delay(500);
        leftDistance = SR04(Trig, Echo);//SR04();

        delay ( 500 );
        myservo . write ( 90 );
        delay ( 500 );
```

Comparing the distance between the left and right sides, which side has the greater distance, that is, the side that is less hindered, backs up and then turns,

```
            if (rightDistance > leftDistance){
              Motor (Stop, 0 );
              delay ( 100 );
              Motor(Backward, 180);
              delay(1000);
              Motor(Clockwise, 250);
              delay(600);
          }
          else if(rightDistance < leftDistance) {
              Motor(Stop, 0);
              delay(100);
              Motor(Backward, 180);
              delay(1000);
              Motor(Contrarotate, 250);
              delay(600);
          }
          else if((rightDistance < 20) || (leftDistance < 20)){
              Motor (Backward, 180 );
              delay ( 1000 );
              Motor (Contrarotate, 250 );
              delay ( 600 );
}
```

6. Tracking car

6.1 Description

In this section , we will learn how to use the line tracking module to learn the principle of line tracking and learn how to control the car through programming to achieve line tracking .

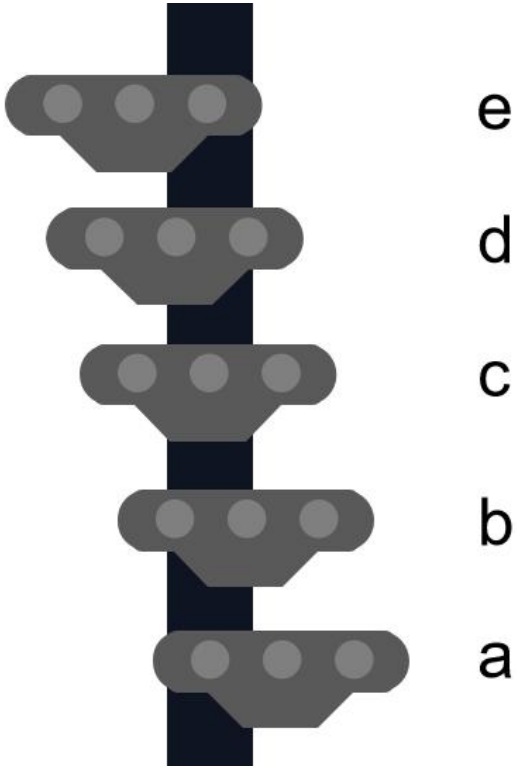
6.2 Tracking module

Tracking sensor is a kind of infrared tracking sensor, which is usually used to manufacture tracking smart cars. The tracker sensor adopts ITR20001/T infrared reflective sensor. The infrared emitting diode of the ITR2001/T sensor continuously emits infrared rays. When the emitted infrared rays are reflected by the object, 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 judged by calculating the analog value of the 3 outputs.

The tracking sensor is located in the front of the car and consists of an infrared emitting tube and an infrared receiving tube. The former is an LED that can transmit infrared light, and the latter is a photoresistor for receiving infrared light. A black surface has a different light reflectivity than a white surface. As a result, a car on a black road receives a different intensity of reflected infrared light than it does on a white road, and the motion changes. The path of motion is determined by inferring the color of the route underneath the car from the voltage of the sensor, based on the principle of voltage division between series resistors .

6.3 Three-way tracking principle



a→ Only the left sensor of the tracking module detects the black line, 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 straight at this time ;

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

e → Only the right sensor of the tracking module detects the black line, 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 is started, the tracking module only needs to sense the black lines on the road and make corresponding actions as needed. There are also many more complex algorithms, such as PID. Therefore, after implementing the tracking function, you can learn more algorithms to control the car yourself.

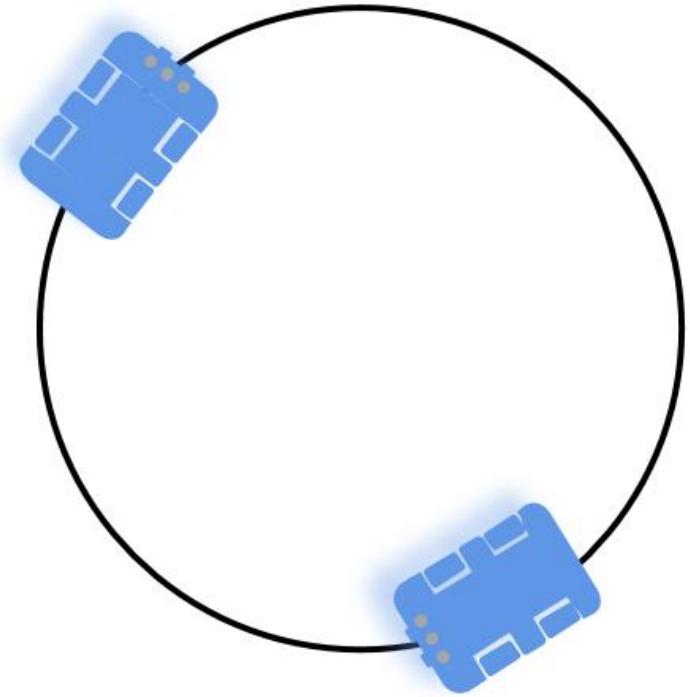
6.4 Prepare insulating tape (black tape) lines

First, we need to make a runway ourselves. We can paste black tape on the flat and clean ground. It's 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 parts of the line should transition as smoothly as possible, otherwise there is a good chance the car will overtake the track.

（2）Line tracing scenes can be made from black and white tape, designed with different walking paths.

（3）In addition to line tracing, we can develop other program line tracing principles.



6.5 Code Analysis

Open the code file (path: 2_Arduino_Code\5_Tracking\5_Tracking.ino)

Define the three-way tracking measurement value variable and the 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 = 35 0 ;
```

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 The 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 );
}
```

In 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 );
}
```

Situation d Right 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 (Clockwise, 165 );
}
```

Plus the case where all sensors detect black lines:

```
    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 ) ;
```

7. Comprehensive function and remote control

7.1 Description

The content of this section mainly understands the ESP32-CAM with camera connected to WiFi to realize real-time media stream shooting and learns to combine with ZHIYI main control board to complete the remote control of multi-function mode.

7.2 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-Fi BTSoC 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

Support OV2640 and OV7670 cameras, built-in flash memory

Support WiFI upload image

Support TF card

Support multiple sleep modes

Embedded Lwip and FreeRTOS

Support STA/AP/STA+AP operation mode

Support smart configuration/AirKiss technology

Support serial 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.

7.3 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 ESP32 core files in Arduino IDE :

In this example we use an Arduino IDE to program the ESP32-Cam board. Therefore, you will need to install the ArduinoIDE and ESP32 add-ons. If you haven't installed the ESP32 plugin in your computer, please follow this tutorial to install it.

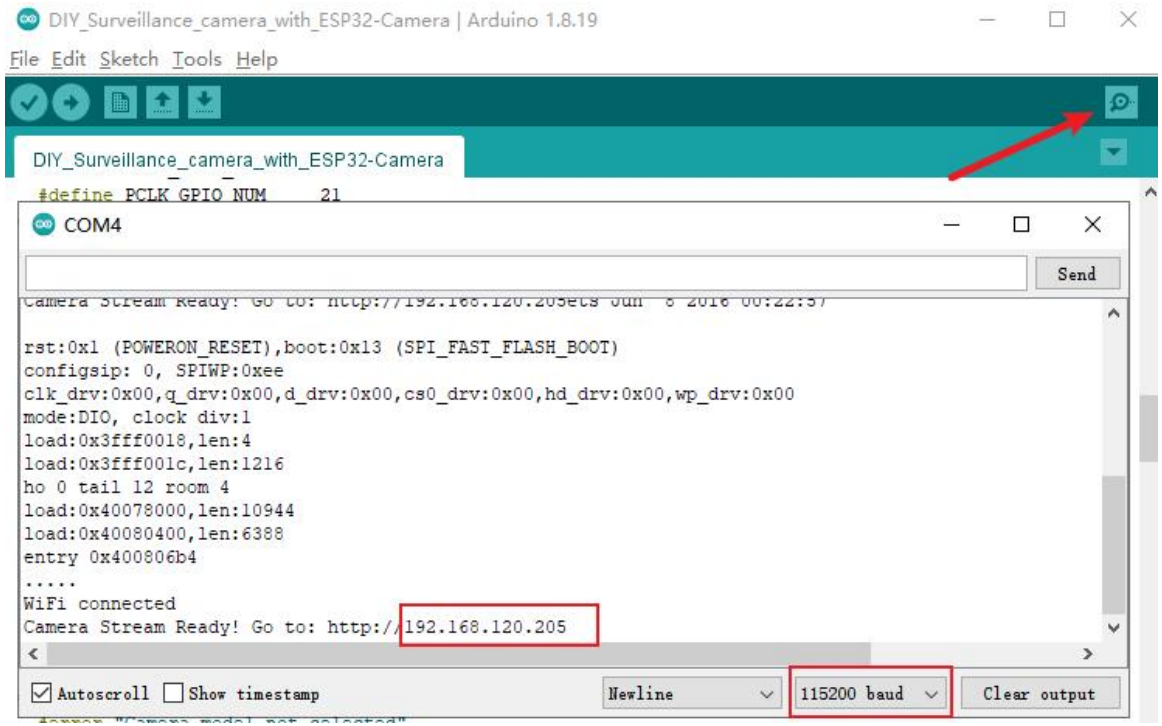
Here we will download the core file for ESP32 board, you need internet to do this step

Copy this link : https://dl.espressif.com/dl/package_esp32_index.json

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

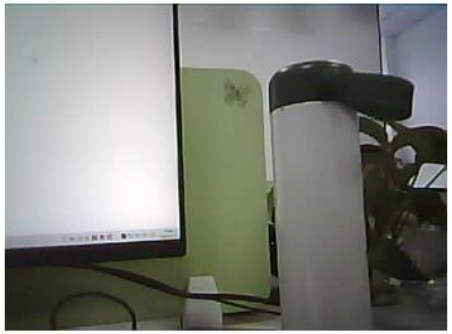
Once the code is successfully uploaded, the ESP32-CAM will reset.

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



Video streaming server:

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



LeftUp	Forward	RightUp
Left	Clockwise	Right
LeftDown	Backward	RightDown

model1	model2
model3	model4
MotorLeft	MotorRight

Light ON	Stop	Light OFF

7.4 Functional Combination

We have learned different function modes in the previous course, now we need to integrate 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, but we still can't operate the car to make corresponding actions, and we need to write a program for the car's main control board.

Open the code file (path: 2_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, click the upload button to upload the code (unplug the four wires in advance, and then plug them in again after the upload is complete)

The code file needs to receive the operation instructions transmitted by ESP32-CAM to execute the corresponding actions, so there will be no immediate response after uploading the code and powering on, and the previous set of code (6.1_ESP32_Car.Ino) needs to be run synchronously and connected to the Effective operation only after WiFi.

7.5 Code Analysis

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

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

Initially set the 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, 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 is saved to middleDistance

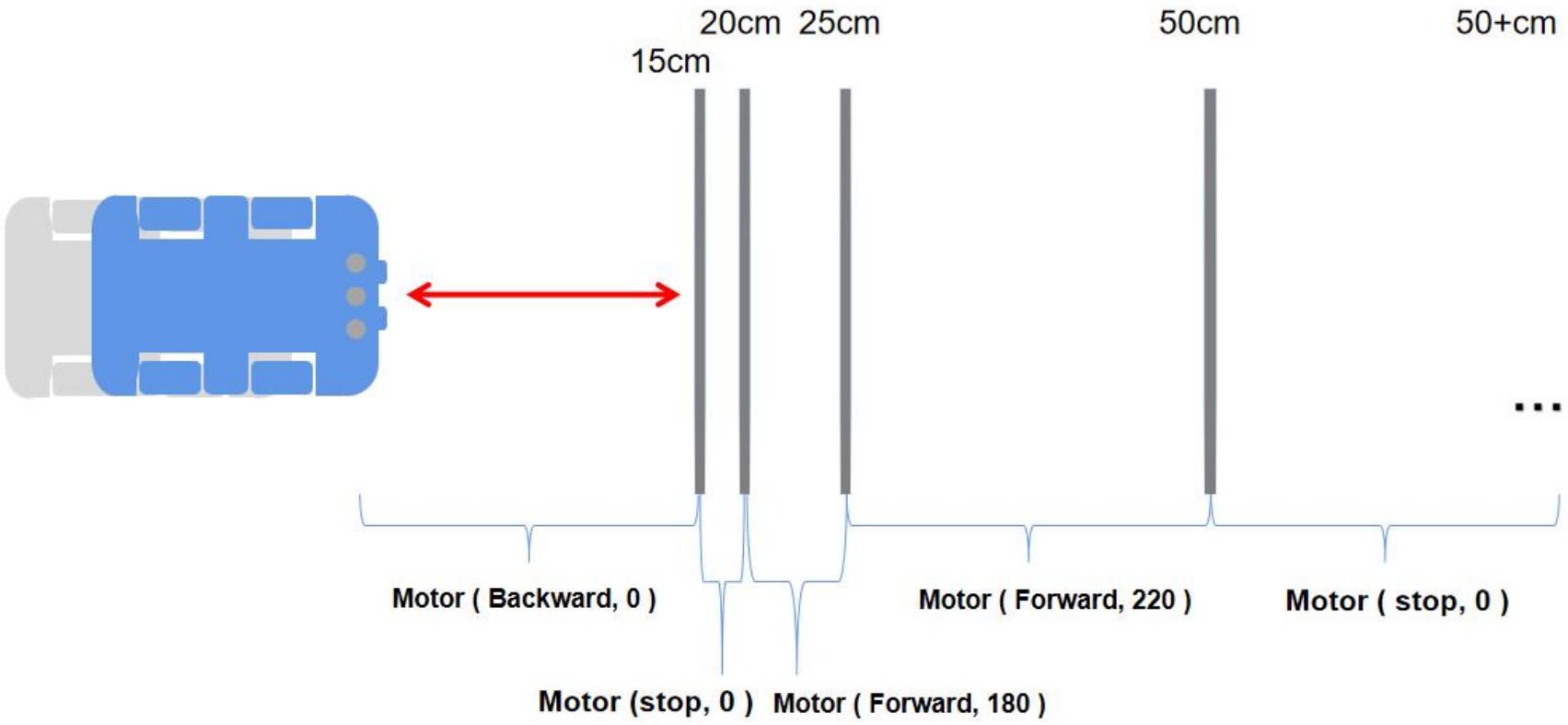
```
UT_distance = SR04 (Trig_PIN, Echo_PIN);
    Serial.println (UT_distance ) ;
middleDistance = UT_distance;
```

Because the connection between the motherboards is always maintaining 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 judge whether to switch the control mode: if(model_var != 1), part of the code is as follows:

```
if (middleDistance <= 25)
 {
      Motor(Stop, 0);
      for(int i = 0;i < 500;i++){
        delay(1);
        RXpack_func();
        if(model_var != 1)
          return ;
      }
      MOTORservo.write(10);
      for(int i = 0;i < 300;i++){
        delay(1);
        RXpack_func();
        if(model_var != 1)
          return ;
      }
```

Model 3 (model3_func) for the car to follow

Also obtain the distance obtained by the ultrasonic measurement in front, and make corresponding movements according to different distances, such as stop, follow slowly, follow quickly or retreat. 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 );
}
}
```

Model 4 (model4_func) is car tracking

Different moving speeds are set here. You can adjust the walking speed when tracking by modifying the speed parameters in the motor to achieve the desired effect. Some codes are 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 wave 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   }
```

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 mode
        break ;
    case 1 :
        model2_func (); // OA (obstacle avoidance) model
        break ;
    case 2 :
        model3_func (); // follow the model
        break ;
    case 3 :
        model4_func (); // tracking mode
        break ;
}

}
```