

## *Lesson4 Obstacle avoidance car*





---

### *Points of this section*




*The joy of learning, is not just know how to control your car, but also know how to protect your car.*

*So, make your car far away from collision.*

#### **Learning parts:**

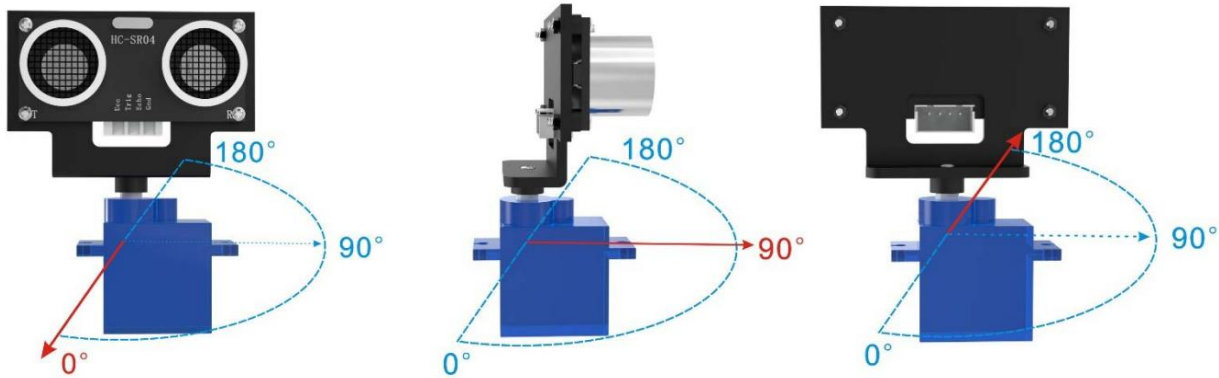
-  *Learn how to assemble the ultrasonic module*
-  *Be familiar with using steering*
-  *Learn about the principle of car avoidance*
-  *Use the program to make obstacle avoidance car come true*

#### **Preparations:**

-  *A car (with battery)*
-  *A USB cable*
-  *A suit of ultrasonic cradle head*

## I . Connection

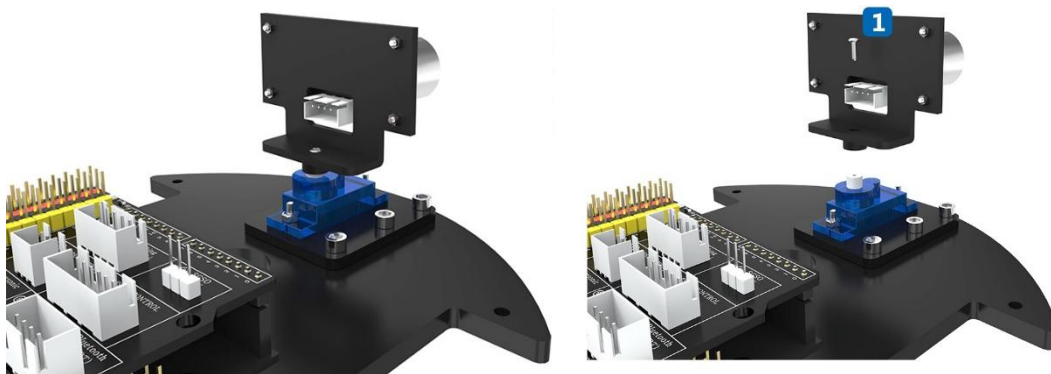
The servo should also be tuned before using the servo ultrasonic unit. You should make sure that the servo can be rotated 180 degrees and that the range is symmetrical along the center axis of the trolley.



The two general methods of debugging a servo are the mechanical method and the program method. The mechanical method is to unscrew the machine screw on the servo, swing the servo left and right to the limit, then turn it back to 90 degrees in the middle of the two limits, remove the ultrasonic structure, turn it toward the front, put it back on, and then screw back the machine screw that you just unscrewed.

This method may have errors, it is recommended to use the program method, the following are the steps to debug the servo by the program method.

**Step 1: Unscrew screw 1 in the picture below and remove the ultrasonic module (without unplugging the wires)**



**Step 2: Since we need to use libraries in this program, we first need to add the library files. (If the servo library was added in the previous lesson, you can skip this step and go directly to step 3)**

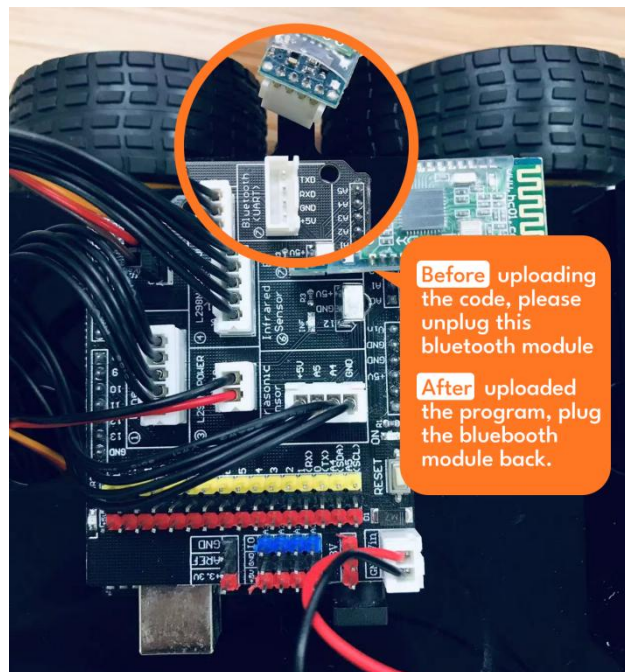
Find this folder Servo in the following path and copy it



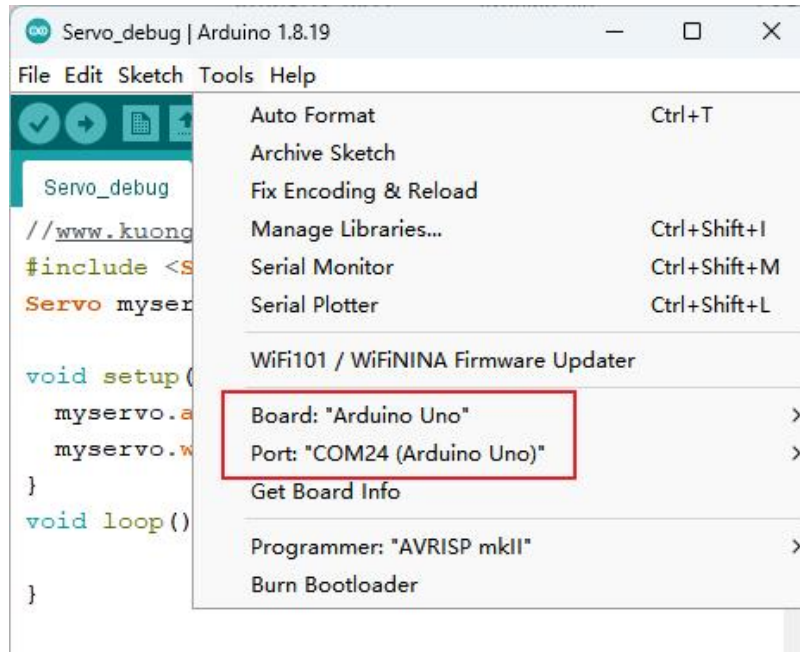
Find the following path and paste the Servo folder you just copied into this path



**Step 3: Connect the cart to the computer with a USB cable, remove the Bluetooth module first, find the path to the Servo\_debug program for the obstacle avoidance cart, open the program and brush it in, and then connect the Bluetooth module back after the upload is successful.**



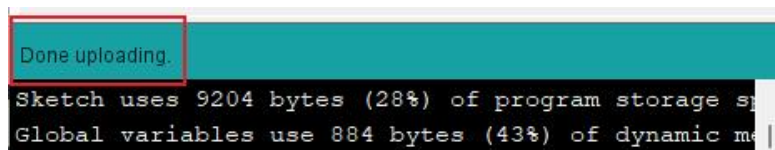
Verify that the development board and ports are Arduino UNO



Click the Upload button to upload the program

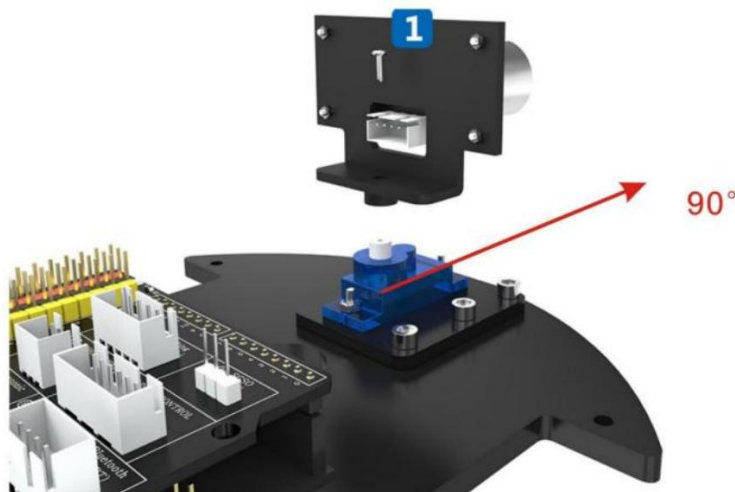


When the upload is successful, the servo will rotate to 90 degrees and come to a standstill.



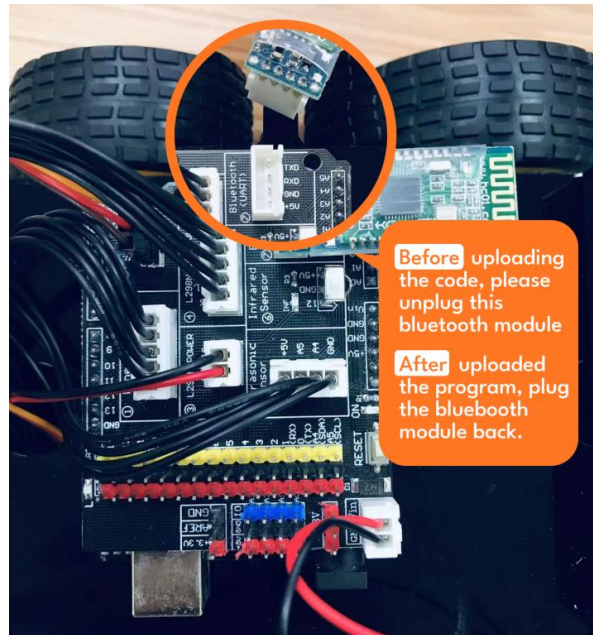
**Step 4: Put the ultrasonic alignment front back on, screw on the #1 machine wire screw in the picture and connect the Bluetooth module back on.**

The angle of each tooth of the servo is about 15 degrees, if it is installed in the center 90 degree direction, it will want to deviate 7.5 degrees to the left or right, this does not affect the normal use of the servo.



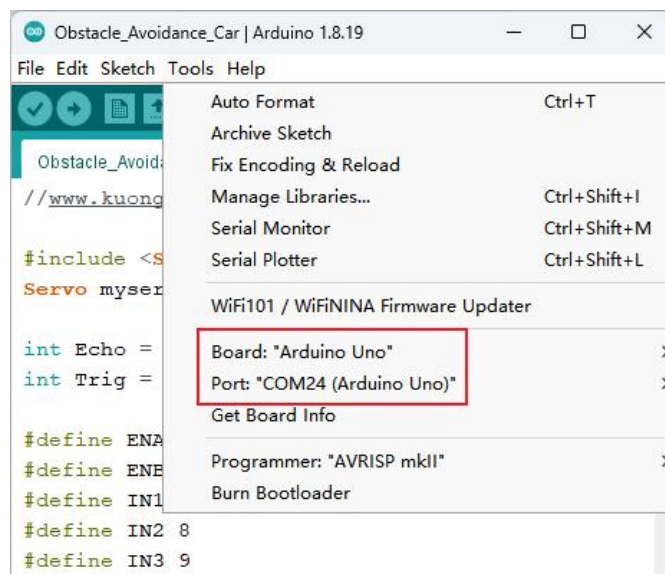
## II. Upload program

The same brush program steps as the test program, use the USB cable to connect the car to the computer, first remove the Bluetooth module, find the path to the Obstacle Avoidance Car Obstacle\_Avoidance\_Car program, open the program and brush it in, and then connect the Bluetooth module back after the upload is successful.

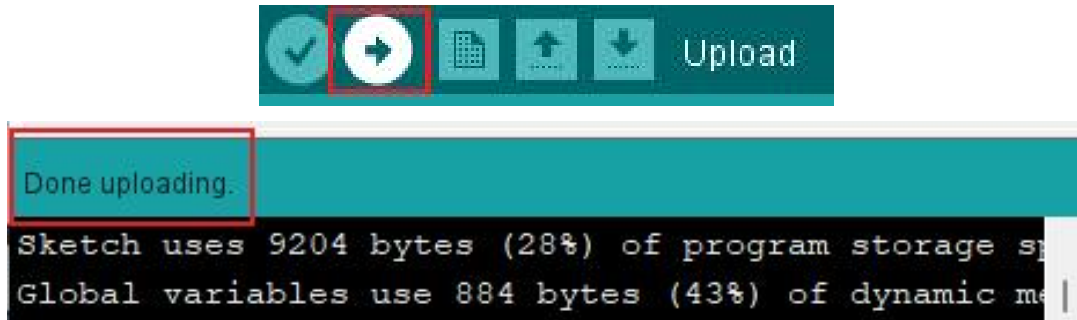


« Lesson 4 Obstacle Avoidance Car » Obstacle\_Avoidance\_Car

名称	修改日期	类型	大小
Obstacle_Avoidance_Car.ino	2023/11/2 15:56	Arduino file	4 KB







After uploading the program to the UNO control board, please connect the Bluetooth module or put it away. And disconnect the data cable, put the car on the ground and turn on the power. Identify you will see the car moving forward. If there is an obstacle ahead, the car will stop and turn the head, and let the ultrasonic distance sensor detect the distance between the two sides. The car will change direction and go in the clear direction. After the obstacle is bypassed, the ranging sensor continues to detect and the vehicle continues to move forward.

#### Code preview:

*//www.kuongshun.com*

```
#include <Servo.h> //servo library
Servo myservo;      // create servo object to control servo
int Echo = A4;
int Trig = A5;
#define ENA 5
#define ENB 6
#define IN1 7
#define IN2 8
#define IN3 9
#define IN4 11
#define carSpeed 200
int rightDistance = 0, leftDistance = 0, middleDistance = 0;

void forward(){
  analogWrite(ENA, carSpeed);
  analogWrite(ENB, carSpeed);
  digitalWrite(IN1, HIGH);
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, LOW);
  digitalWrite(IN4, HIGH);
  Serial.println("Forward");
}
```

```
void back() {
    analogWrite(ENA, carSpeed);
    analogWrite(ENB, carSpeed);
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, HIGH);
    digitalWrite(IN3, HIGH);
    digitalWrite(IN4, LOW);
    Serial.println("Back");
}

void left() {
    analogWrite(ENA, carSpeed);
    analogWrite(ENB, carSpeed);
    digitalWrite(IN1, LOW);
    digitalWrite(IN2, HIGH);
    digitalWrite(IN3, LOW);
    digitalWrite(IN4, HIGH);
    Serial.println("Left");
}

void right() {
    analogWrite(ENA, carSpeed);
    analogWrite(ENB, carSpeed);
    digitalWrite(IN1, HIGH);
    digitalWrite(IN2, LOW);
    digitalWrite(IN3, HIGH);
    digitalWrite(IN4, LOW);
    Serial.println("Right");
}

void stop() {
    digitalWrite(ENA, LOW);
    digitalWrite(ENB, LOW);
    Serial.println("Stop!");
}

//Ultrasonic distance measurement Sub function
int Distance_test() {
    digitalWrite(Trig, LOW);
    delayMicroseconds(2);
    digitalWrite(Trig, HIGH);
    delayMicroseconds(20);
    digitalWrite(Trig, LOW);
```

```
float Fdistance = pulseIn(Echo, HIGH);
Fdistance= Fdistance / 58;
return (int)Fdistance;
}

void setup() {
  myservo.attach(3); // attach servo on pin 3 to servo object
  Serial.begin(9600);
  pinMode(Echo, INPUT);
  pinMode(Trig, OUTPUT);
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  pinMode(IN3, OUTPUT);
  pinMode(IN4, OUTPUT);
  pinMode(ENA, OUTPUT);
  pinMode(ENB, OUTPUT);
  stop();
}

void loop() {
  myservo.write(90); //set servo position according to scaled value
  delay(500);
  middleDistance = Distance_test();

  if(middleDistance <= 25) {
    stop();
    delay(500);
    myservo.write(10);
    delay(500);
    rightDistance = Distance_test();

    delay(500);
    myservo.write(90);
    delay(500);
    myservo.write(170);
    delay(500);
    leftDistance = Distance_test();

    delay(500);
    myservo.write(90);
    delay(500);
    if(rightDistance > leftDistance) {
      stop();
      delay(100);
    }
  }
}
```



```
    back();
    delay(200);
    right();
    delay(300);
}
else if(rightDistance < leftDistance) {
    stop();
    delay(100);
    back();
    delay(200);
    left();
    delay(300);
}
else if((rightDistance <= 25) || (leftDistance <= 25)) {
    back();
    delay(200);
}
else {
    forward();
}
}
else {
    forward();
}
}
```

### III. Introduction of principle

First of all, let's learn about the SG90 Servo:

#### SG90 Servo

180 angle steering  
gear

Rotation angle is  
from 0 to 180

Brown line —GND  
Red line —SV  
Orange line —signal(PWM)



Classification: 180 steering gear

Normally the servo has 3 controlling line: power supply, ground and sign.

Definition of the servo pins: brown line——GND, red line——5V, orange——signal.

### How does servo work:

The signal modulation chip in the servo receives signals from the controller board then the servo will get the basic DC voltage. There is also a reference circuit inside the servo which will produce a standard voltage. These two voltages will compare to each other and the difference will be output. Then the motor chip will receive the difference and decide the rotational speed, direction and angel. When there is no difference between the two voltages, the servo will stop.

### How to control the servo:

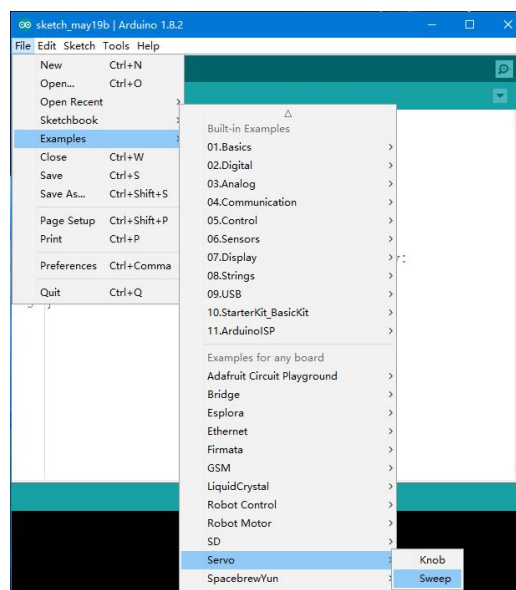
To control the servo rotation, you need to make the time pulse to be about 20ms and the high level pulse width to be about 0.5ms~2.5ms, which is consistent with the angle limited of the servo.

Taking 180 angle servo for example, corresponding control relation is as below:

0.5ms	0 degree
1.0ms	45 degree
1.5ms	90 degree
2.0ms	135 degree
2.5ms	180 degree

### The example program:

Open Arduino IDE and select “File->Examples->Servo->Sweep”



**Next, let's have a look at the ultrasonic sensor module.**



**Feature of the module:** testing distance, high precision module.

**Application of the products:** robot obstacle avoidance、 object testing distance、 liquid testing、 public security、 parking lot testing.

**Main technical parameters**

- (1): voltage used: DC---5V
- (2): static current: less than 2mA
- (3): level output: higher than 5V
- (4): level output: lower than 0
- (5): detection angle: not bigger than 15 degree
- (6): detecting distance: 2cm-450cm
- (7): high precision: up to 0.2cm

Method of connecting lines: VCC, trig (the end of controlling), echo (the end of receiving), GND

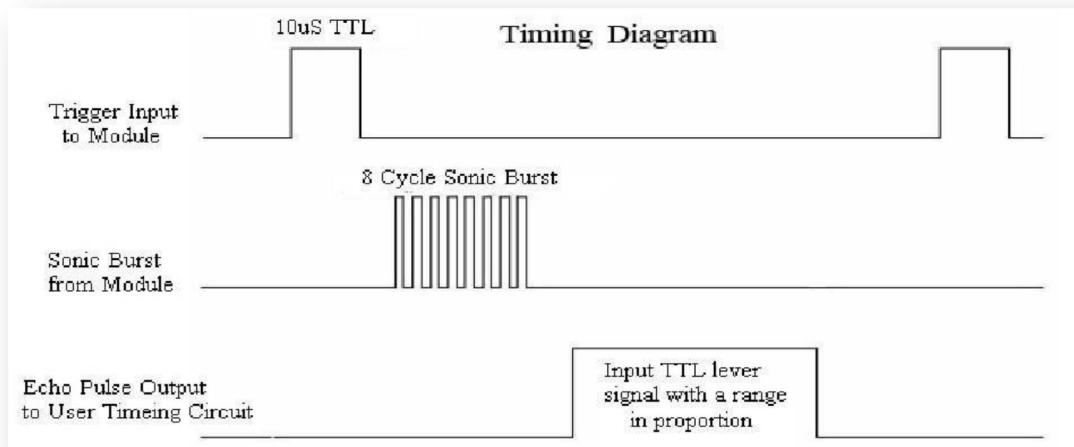
**How does the module work:**

- (1)Apply IO port of TRIG to trigger ranging, give high level signal, at least 10us one time;
- (2)The module sends 8 square waves of 40kHz automatically, tests if there are signals returned automatically;
- (3)If there are signals received, the module will output a high level pulse through IO port of ECHO, the duration time of high level pulse is the time between the wave sending and receiving. So the module can know the distance according to the time.

Testing distance= (high level time\* velocity of sound (340M/S))/2);

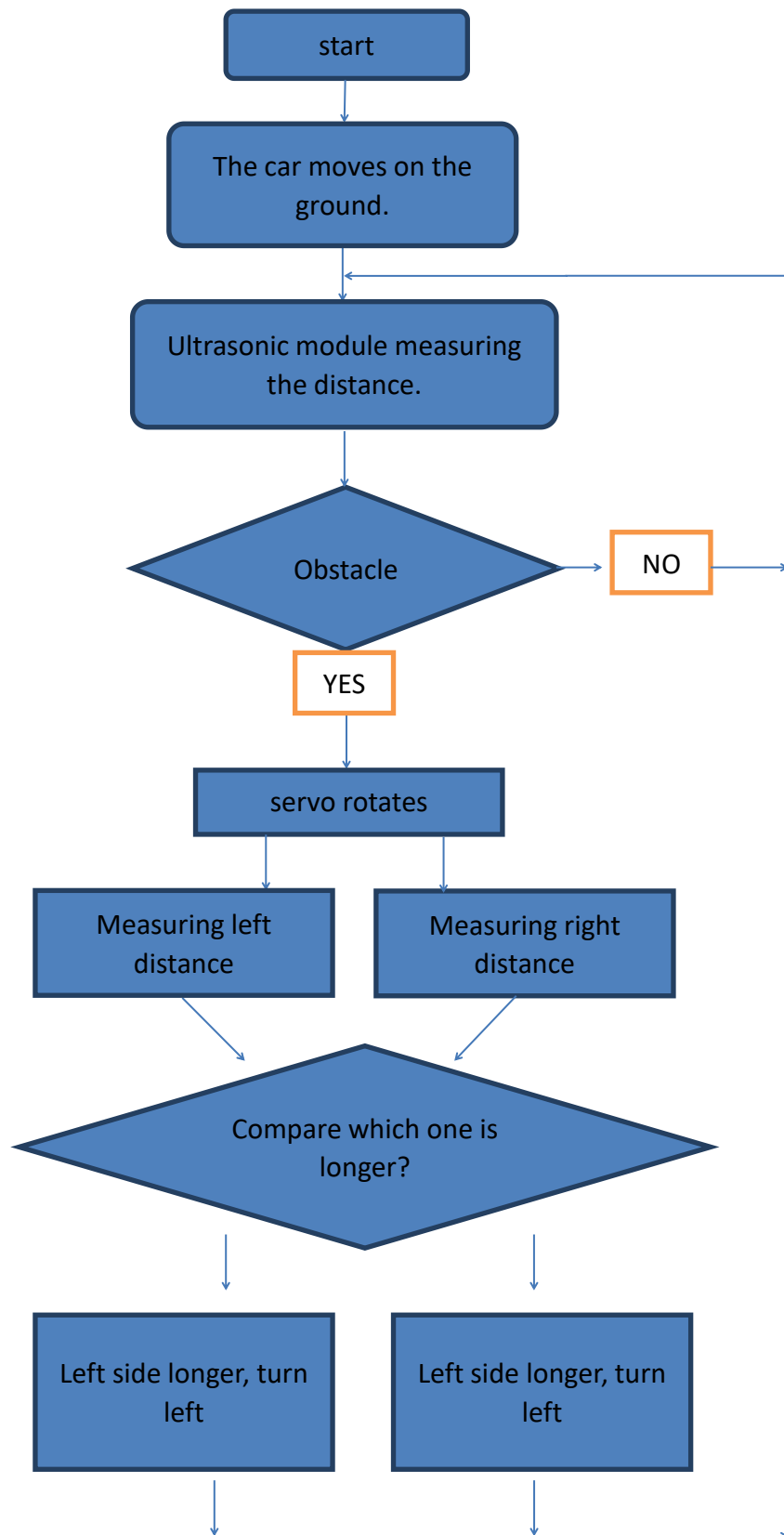
**Actual operation:**

The Timing diagram is shown below. You only need to supply a short 10uS pulse to the trigger input to start the ranging, and then the module will send out an 8 cycle burst of ultrasound at 40 kHz and raise its echo. The Echo is a distance object that is pulse width and the range in proportion. You can calculate the range through the time interval between sending trigger signal and receiving echo signal. Formula:  $\mu\text{S} / 58 = \text{centimeters}$  or  $\mu\text{S} / 148 = \text{inch}$ ; or: the range = high level time \* velocity (340M/S) / 2; we suggest to use over 60ms measurement cycle, in order to prevent trigger signal to the echo signal.



```
//Ultrasonic distance measurement Sub function
```

```
int Distance_test() {
    digitalWrite(Trig, LOW);
    delayMicroseconds(2);
    digitalWrite(Trig, HIGH);
    delayMicroseconds(20);
    digitalWrite(Trig, LOW);
    float Fdistance = pulseIn(Echo, HIGH);
    Fdistance= Fdistance / 58;
    return (int)Fdistance;
}
```



From above picture, we can see that the principle of obstacle avoidance car is very simple. The ultrasonic sensor module will detect the distance between the car and the obstacles again and again and sending the data to the controller board, then the car will stop and rotate the servo to detect the left side and right side. After compared the distance from the different side, the car turn to the side which has longer distance and move forward. Then the ultrasonic sensor module detects the distance again.

**Code preview:**

```
if(rightDistance > leftDistance) {
    stop();
    delay(100);
    back();
    delay(200);
    right();
    delay(300);
}
else if(rightDistance < leftDistance) {
    stop();
    delay(100);
    back();
    delay(200);
    left();
    delay(300);
}
else if((rightDistance <= 25) || (leftDistance <= 25)) {
    back();
    delay(200);
}
else {
    forward();
}
```