Arduino UNO Car 1.0Tutorial

V1.0.2019.07.31

Lesson 2 Barrier avoidance car

The point of this section is the magic of writing code. Not only can you control it by repeatedly changing the sketch, but you can also let your car know about self-protection

So, let's learn how to keep your car away from obstacles.

1. Learning:

1. To learn how to use ultrasonic modules

2.To know the principle of the car obstacle

3.To use this program to make barrier avoidance car a reality

II. Preparations:

Smart car

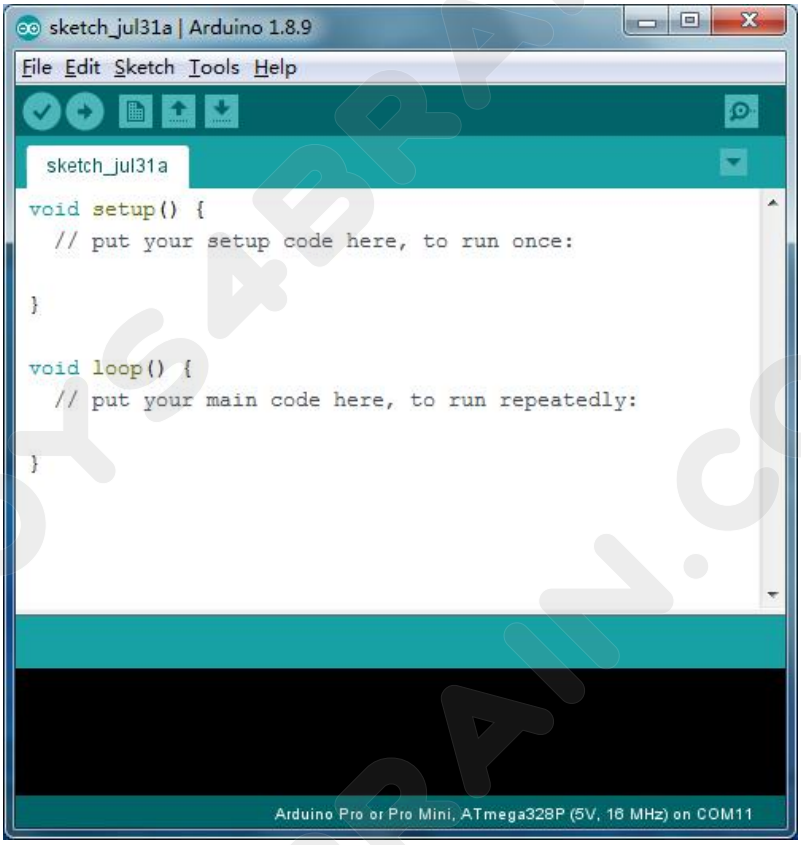
USB cable

Ultrasonic bracket kit with colorful LED

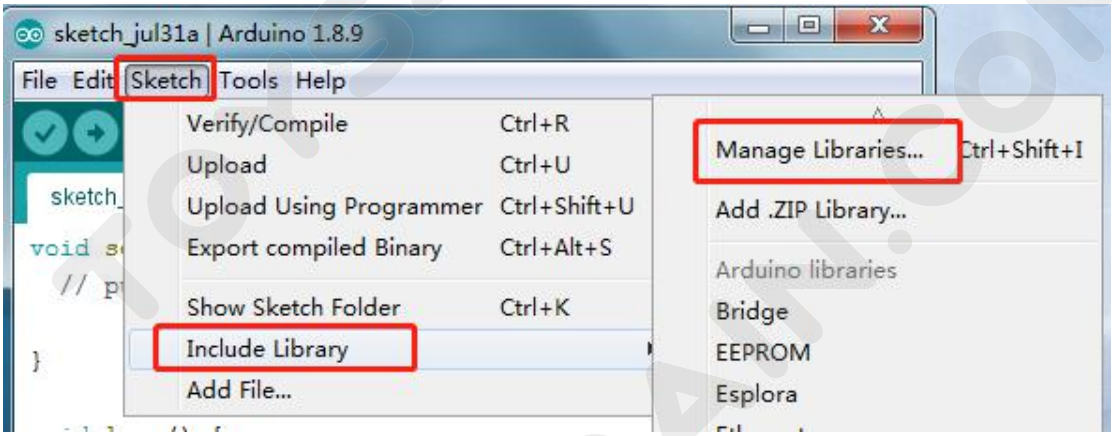
1. Upload program

The program uses library<servo.h>, So we need to install the library first .

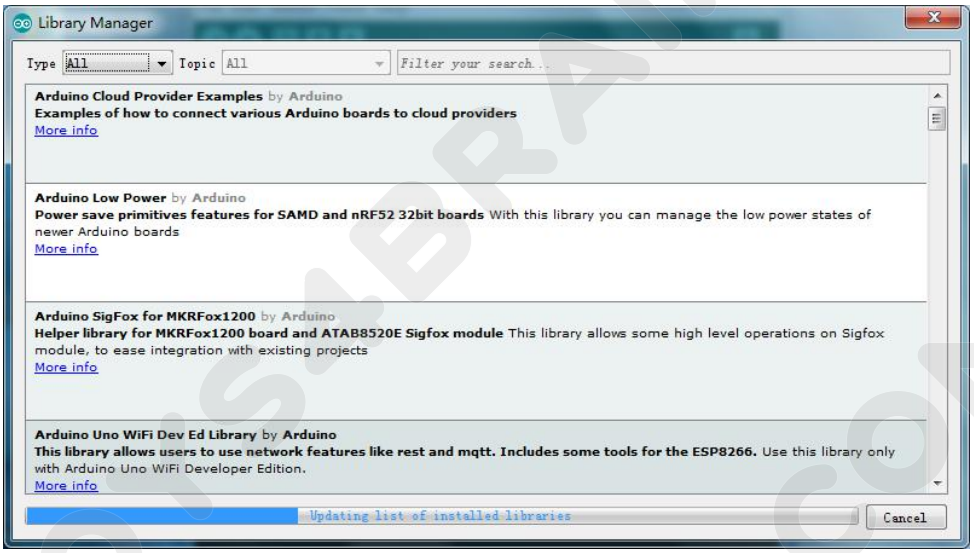
Open the Arduino compiler software



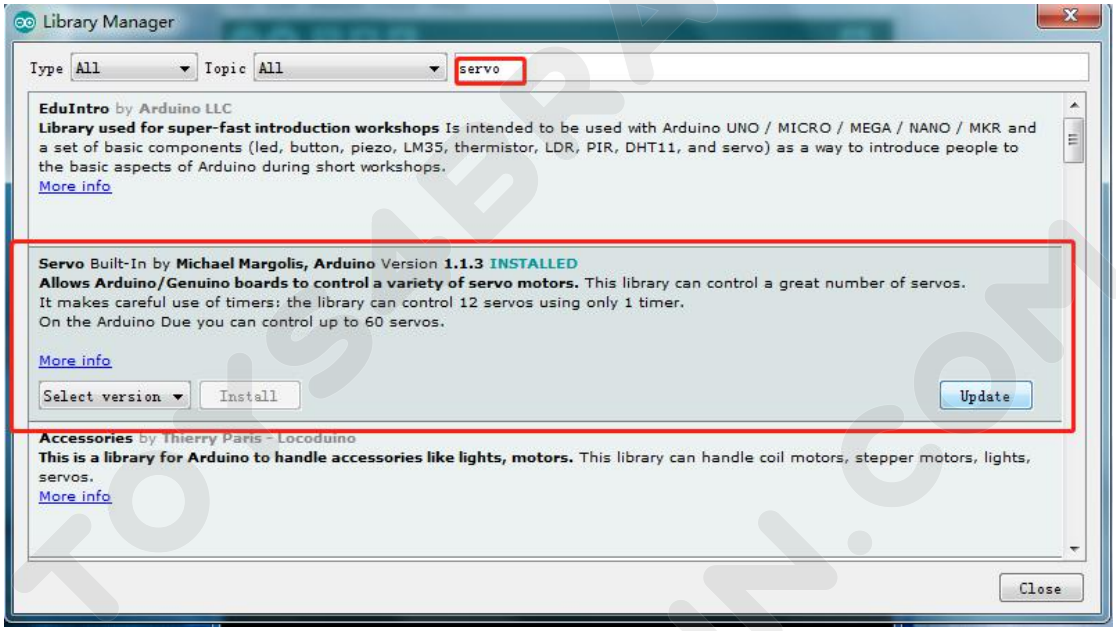
Select Sketch -> Include Library -> Manage Libraries…



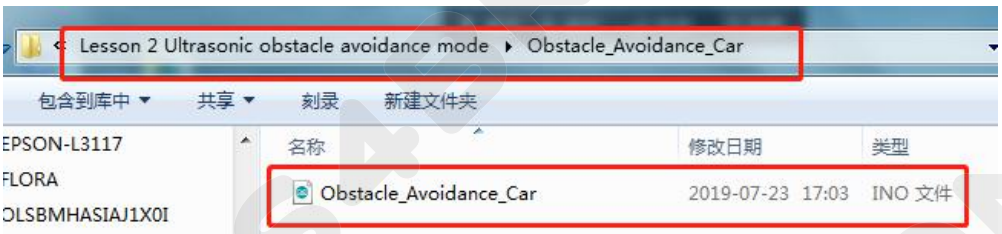
Waiting for the “Library” downloaded



Search the servo and install the latest version. The following figure shows that the server library is installed.



Connect the Arduino nano controller board to your computer and open the code file in the path:"\ Lesson 2 Ultrasonic obstacle avoidance mode \ Obstacle_Avoidance_Car \ Obstacle_Avoidance_Car.ino".
Upload the program to the Arduino nano board.



```
#include <Servo.h>    //servo library
Servo myservo;          // create servo object to control servo

int Echo = A4;
int Trig = A5;
int sound = 15;
int left_LED = 16;
int Right_LED = 17;
#define ENB 5
#define IN1 7
#define IN2 8
#define IN3 9
#define IN4 11
#define ENA 6
#define carSpeed 150
int rightDistance = 0, leftDistance = 0, middleDistance = 0;

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

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

void left() {
  analogWrite(ENA, carSpeed);
  analogWrite(ENB, carSpeed);
  digitalWrite(IN1, HIGH);
```

```
  digitalWrite(IN2, LOW);
  digitalWrite(IN3, HIGH);
  digitalWrite(IN4, LOW);
    digitalWrite(left_LED,HIGH);
  digitalWrite(Right_LED,LOW);
  Serial.println("Left");
}

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

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

//Ultrasonic distance measurement Sub function
int getDistance() {
      digitalWrite(Trig, LOW);
      delayMicroseconds(2);
      digitalWrite(Trig, HIGH);
      delayMicroseconds(10);
      digitalWrite(Trig, LOW);
      return (int)pulseIn(Echo, HIGH) / 58;
}


/***********BLINKSOUND****/
void blinks()
{
 for(int i=0;i<5;i++)
 {
  digitalWrite(left_LED,HIGH);
  digitalWrite(Right_LED,HIGH);
```

```
    digitalWrite(sound,HIGH);
    delay(100);
    digitalWrite(left_LED,LOW);
    digitalWrite(Right_LED,LOW);
    digitalWrite(sound,LOW);
    delay(100);
  Serial.println("Blink!");
}
delay(150);
    }



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);
    pinMode(left_LED,OUTPUT);
    pinMode(Right_LED,OUTPUT);
    pinMode(sound,OUTPUT);
    stop();
}

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

      if(middleDistance <= 30) {
        stop();
        delay(500);
        myservo.write(10);
        delay(1000);
        rightDistance = getDistance();

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

```
        delay(1000);
        myservo.write(180);
        delay(1000);
        leftDistance = getDistance();

        delay(500);
        myservo.write(90);
        delay(1000);
        if(rightDistance < leftDistance) {
              left();
           delay(360);
        }
        else if(rightDistance > leftDistance) {
       right();
           delay(360);
        }
        else if((rightDistance <= 30) || (leftDistance <= 30)) {

           back();
           blinks();
         delay(180);
        }
        else {
           forward();
        }
    }
    else {
          forward();
    }
}
```

After uploading the program to the Arduino Nano control board, disconnect the cable, put the car on it and turn on the power supply. You will see the car moving forward and the cloud platform continuing to rotate, so that the car moving distance can measure sensor's continuous running. If there is an obstacle in front, the cloud platform will stop and the vehicle will change direction and bypass the barrier. After bypassing the obstacle, the cloud platform will continue to rotate and the car will continue to move forward.

FAQ about the servo motor.

The angle of each tooth on the micro servo is 15 degrees. If you install it in the middle and as 90 degrees, rotate 15 degrees to the left or right, that is to say, the actual degree of micro servo is 75 degrees or 105 degrees.

1. Why does the micro-servo rotate 15 degrees counter-clockwise every time the power is turned on?

This is normal for SG90 micro servo, it won't affect the normal use of the program. If you don't use a program to control it, you can rotate it back to its normal state by hand and turn off the wire connected to the micro-servo before turning on the power supply.

2. The micro servo is out of control and keeps rotating.

Using the "myservo.write (angle)" command micron servo angle range from 0 to 180. If out of range, the micro servo will not be able to recognize the angle and will continue to rotate.

2、 Introduction of principle：

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

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.

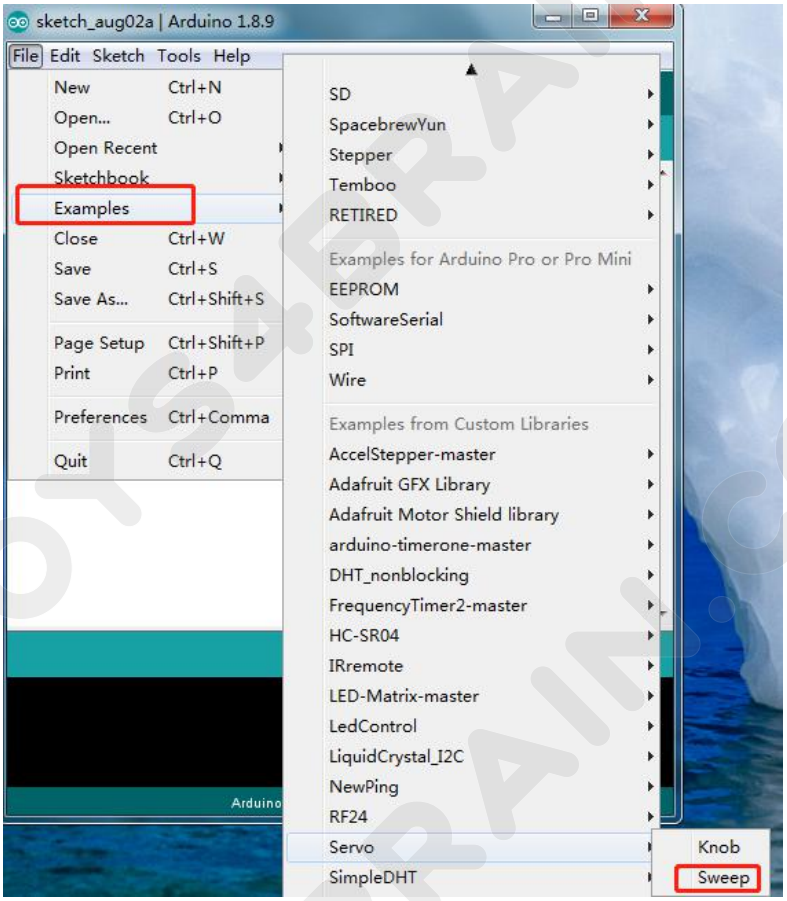
How does servo work:
The signal modulation chip in the servo receives signals from the controller board then
the servo will get the basic DC voltage. There is also a reference circuit inside the servo
which will produce a standard voltage. These two voltages will compare to each other and the difference 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:

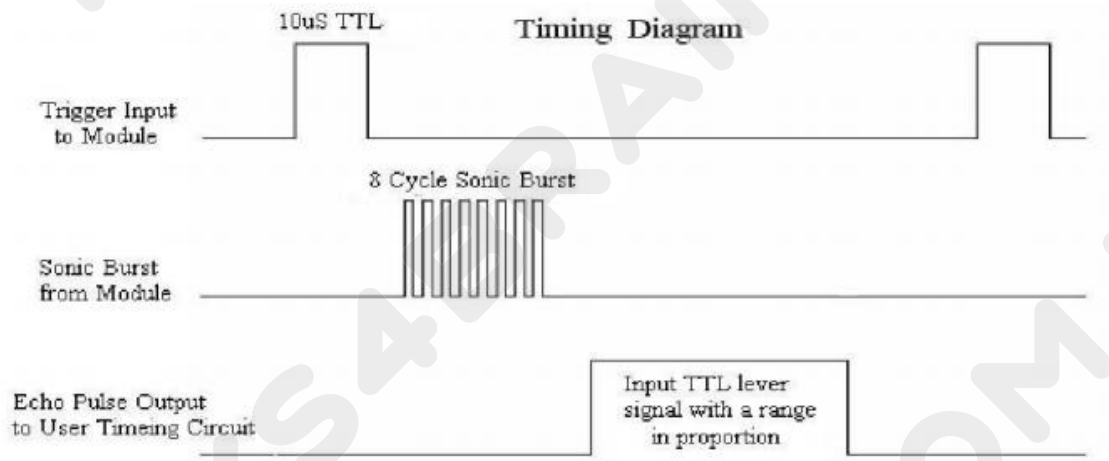
Open Arduino IDE and select ‘‘File->Examples->Servo->Sweep’’



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 40kz 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 short10uS 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: uS / 58 = centimeters or uS / 148 =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 getDistance() {
digitalWrite(Trig, LOW);
delayMicroseconds(2);
digitalWrite(Trig, HIGH);
delayMicroseconds(10);
digitalWrite(Trig, LOW);
return (int)pulseIn(Echo, HIGH) / 58;
}
```

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) {
right();
delay(360);
}
else if(rightDistance < leftDistance) {
left();
delay(360);
}
else if((rightDistance <= 20) || (leftDistance <= 20)) {
```

```
back();
delay(180);
}
else {
forward();
}
}
else {
forward();
}
```