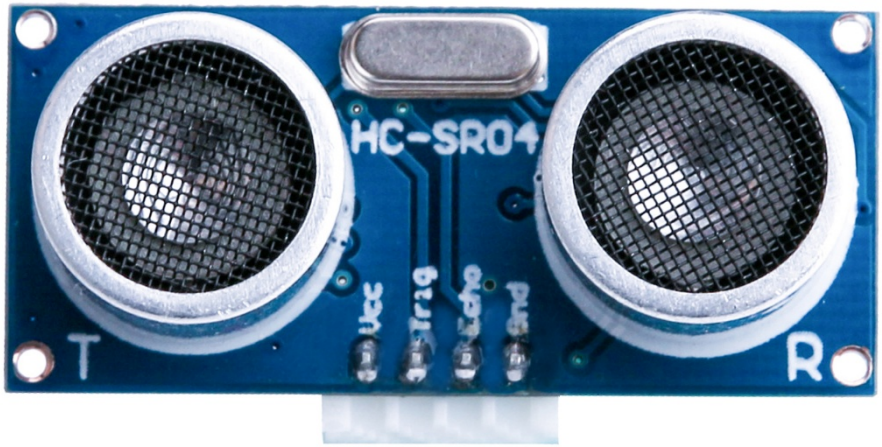
Lección 4 Coche Evita obstáculos



Puntos de esta sección

La alegría de aprender, no es sólo saber cómo controlar su coche, sino también saber cómo proteger su coche. Por lo tanto, hacer que el coche quede lejos de la colisión.

Conocimiento de las piezas:

- Aprenda cómo ensamblar el módulo ultrasónico
- Estar familiarizado con el manejo
- Aprenda sobre el principio de la evitación del coche
- Utilice el programa para hacer realidadd que el coche evite los obstáculos

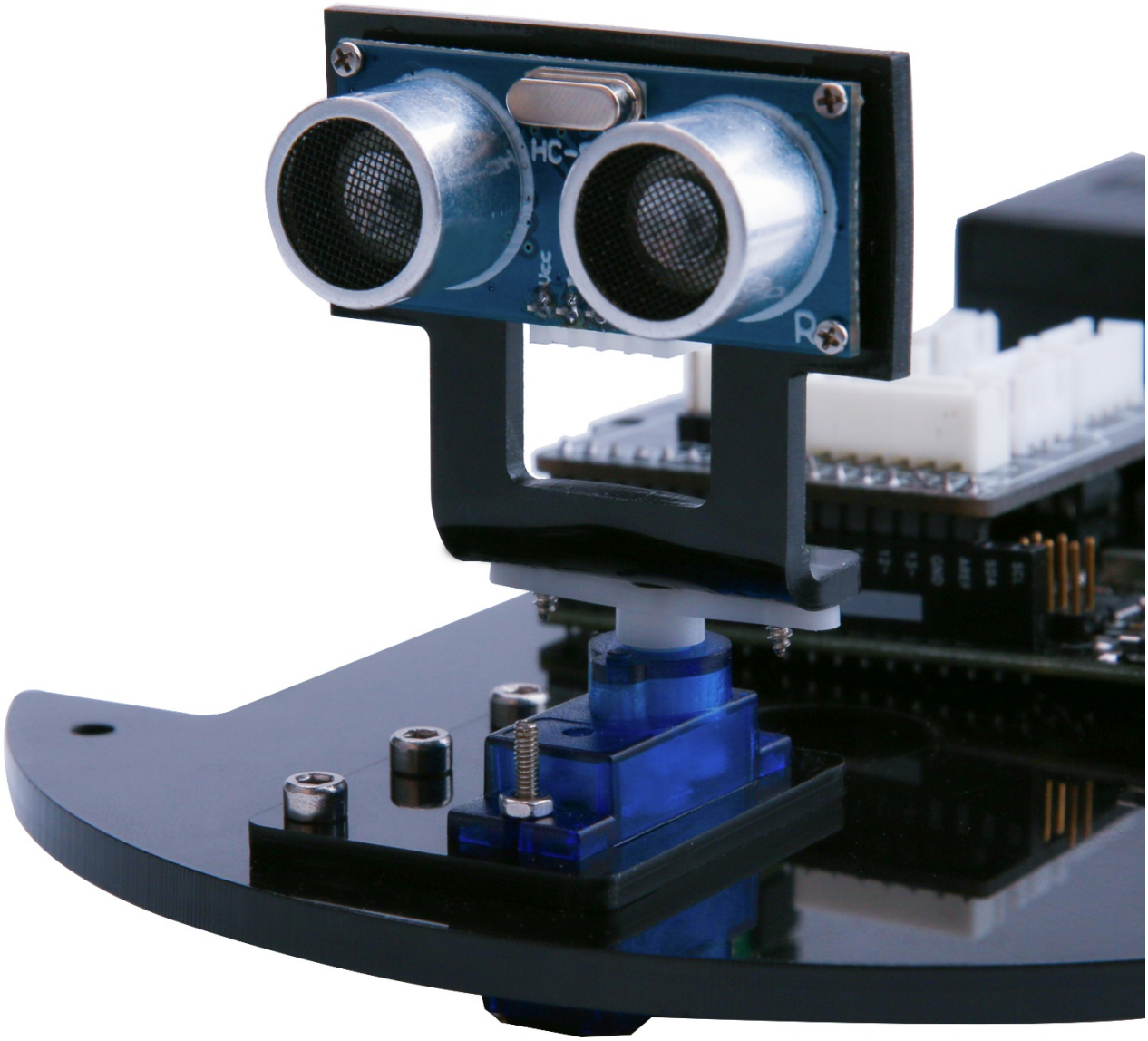
Preparación:

- Un coche (con batería)
- Un cable USB
- Una cabeza sensora ultrasónica

Ⅰ. Conexión

servo


TOWER PRO™
Micro Servo
99
SG90



Ⅱ. Subir programa

```
#include <Servo.h> //servo library
Servo myservo; // Crear objeto servo para controlar el servo
int Echo = A4;
int Trig = A5;
int in1 = 6;
```

```
int in2 = 7;
int in3 = 8;
int in4 = 9;
int ENA = 5;
int ENB = 11;
int ABS = 150;
int rightDistance = 0,leftDistance = 0,middleDistance = 0 ;
void _mForward()
{
  analogWrite(ENA,ABS);
  analogWrite(ENB,ABS);
    digitalWrite(in1,HIGH);//digital output
    digitalWrite(in2,LOW);
    digitalWrite(in3,LOW);
    digitalWrite(in4,HIGH);
  Serial.println("go forward!");
}

void _mBack()
{
  analogWrite(ENA,ABS);
  analogWrite(ENB,ABS);
    digitalWrite(in1,LOW);
    digitalWrite(in2,HIGH);
    digitalWrite(in3,HIGH);
    digitalWrite(in4,LOW);
  Serial.println("go back!");
}

void _mleft()
{
  analogWrite(ENA,ABS);
  analogWrite(ENB,ABS);
```

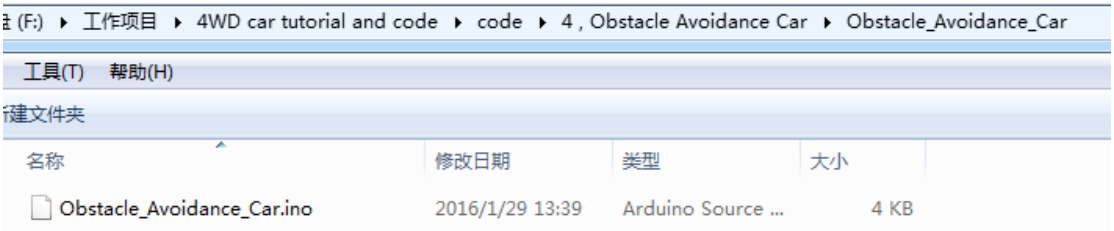
```
  digitalWrite(in1,HIGH);
  digitalWrite(in2,LOW);
  digitalWrite(in3,HIGH);
  digitalWrite(in4,LOW);
 Serial.println("go left!");
}

void _mright()
{
 analogWrite(ENA,ABS);
 analogWrite(ENB,ABS);
  digitalWrite(in1,LOW);
  digitalWrite(in2,HIGH);
  digitalWrite(in3,LOW);
  digitalWrite(in4,HIGH);
 Serial.println("go right!");
}
void _mStop()
{
  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);
  _mStop();
}

void loop()
{
    myservo.write(90);//setservo position according to scaled value
    delay(500);
    middleDistance = Distance_test();
    #ifdef send
    Serial.print("middleDistance=");
    Serial.println(middleDistance);
    #endif

    if(middleDistance<=20)
    {
      _mStop();
      delay(500);
      myservo.write(5);
```

```
        delay(1000);
        rightDistance = Distance_test();

        #ifdef send
        Serial.print("rightDistance=");
        Serial.println(rightDistance);
        #endif

        delay(500);
          myservo.write(90);
        delay(1000);
        myservo.write(180);
        delay(1000);
        leftDistance = Distance_test();

        #ifdef send
        Serial.print("leftDistance=");
        Serial.println(leftDistance);
        #endif

        delay(500);
        myservo.write(90);
        delay(1000);
        if(rightDistance>leftDistance)
        {
          _mright();
           delay(360);
         }
          else if(rightDistance<leftDistance)
         {
          _mleft();
           delay(360);
         }
```

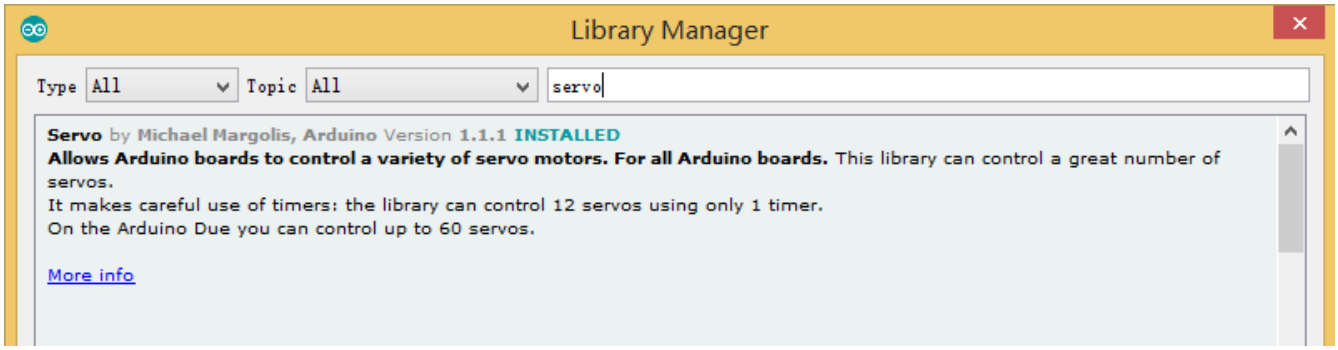
```
        else if((rightDistance<=20) | | (leftDistance<=20))
        {
          _mBack();
          delay(180);
        }
        else
        {
          _mForward();
        }
    }
    else
        _mForward();
}
```

Abre el richero Obstacle_Avoidance_Car\Obstacle_Avoidance_Car.ino



Porque el programa utiliza la biblioteca <servo.h>, así que necesitamos instalar la biblioteca al principio.

Abre la ruta---Include Library---Manage Libraries



Buscar servo y luego instalar la versión más reciente.

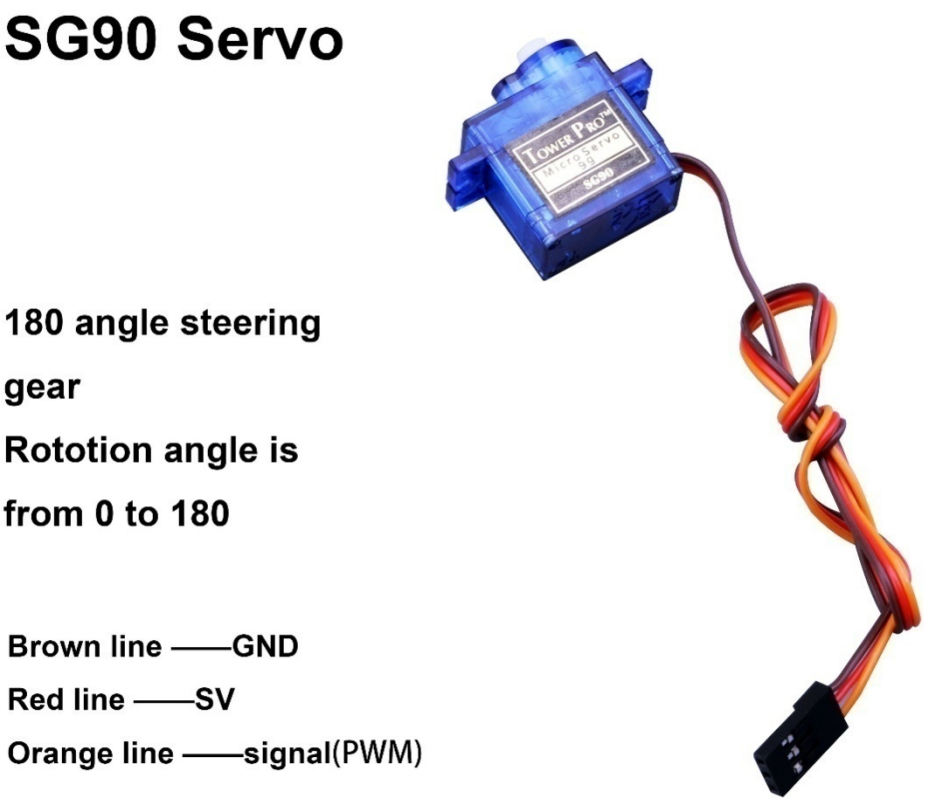
Después de cargar el programa en la placa de control UNO, desconecte el cable, ponga el vehículo en el suelo y encienda la fuente de alimentación.

Verá que el vehículo se moverá hacia adelante y la plataforma de nubes seguirá

girando para que los sensores de medición de distancia funcionen continuamente. Si hay obstáculos por delante, la plataforma de la nube se detendrá y el vehículo cambiará su dirección para evitar el obstáculo. Después de pasar por alto el obstáculo, la plataforma de nube seguirá girando de nuevo y el vehículo también se moverá.

Ⅲ. Introducción del principio

En primer lugar, vamos a aprender sobre el Servo SG90:



Clasificación: 180 equipo de dirección
Normalmente el servo tiene 3 líneas de control: alimentación, tierra y señal.
Definición de los pines servo: línea marrón - GND, línea roja - 5V, naranja - señal.

¿Cómo funciona el servo?:

El chip de modulación de señal en el servo recibe señales de la placa controladora, entonces el servo obtendrá la tensión CC básica. También hay un circuito de referencia dentro del servo que producirá un voltaje estándar. Estos dos voltajes se compararán entre sí y la diferencia será la salida. Entonces el chip del motor recibirá la diferencia y decidirá la velocidad de rotación, la dirección y el ángel. Cuando no hay diferencia entre los dos voltajes, el servo se detendrá.

Cómo controlar el servo:

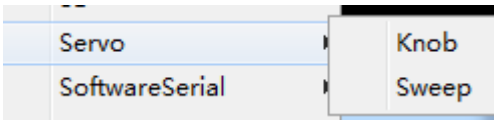
Para controlar la rotación del servo, es necesario que el pulso de tiempo sea de unos 20ms y que el ancho de pulso de alto nivel sea aproximadamente 0.5ms ~ 2.5ms, lo cual es consistente con el ángulo limitado del servo.

Tomando el servo de 180 ángulos por ejemplo, la relación de control correspondiente es la siguiente:

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

El programa:

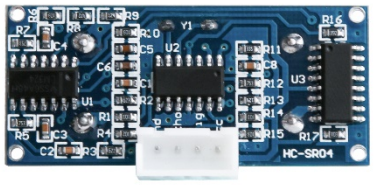
Arduino tiene un archivo de biblioteca.<Servo.h>




```
Servo myservo;  // Crear objeto servo para controlar el servo
 myservo.attach(3);  // Fije el servo en el pasador 3 al objeto servo
 myservo.write(90); // Ajuste la posición del servo según el valor escalado
```

Usted puede conducir el engranaje de dirección en 9 palabras.

A continuación, echemos un vistazo al módulo de sensor ultrasónico.



Característica del módulo: Distancia de prueba, módulo de alta precisión.

Aplicación de los productos: Robot de evitación de obstáculos, objeto de prueba de distancia, pruebas de líquidos, seguridad pública, pruebas de estacionamiento.

Principales parámetros técnicos

(1): voltaje utilizado: DC --- 5V

(2): corriente estática: menos de 2mA

(3): nivel de salida: superior a 5V

(4): nivel de salida: inferior a 0

(5): ángulo de detección: no más grande que 15 grados

(6): detección de la distancia: los 2cm-450cm

(7): alta precisión: hasta 0.2cm

Método de conexión de las líneas: VCC, trig (el fin del control), echo (el final de la recepción), GND

Cómo funciona el módulo:

(1) Aplique el puerto IO de TRIG para activar el rango, dar señal de nivel alto, al menos 10us una vez;

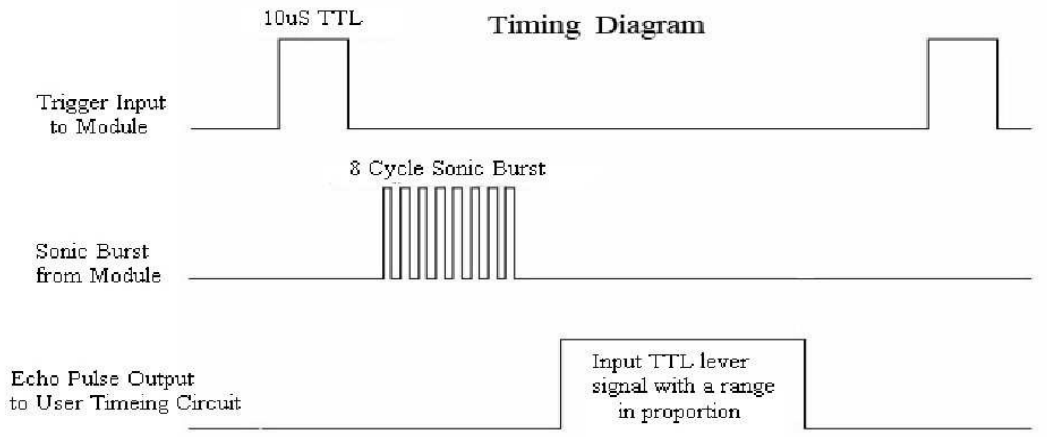
(2) El módulo envía 8 ondas cuadradas de 40kz automáticamente, prueba si hay señales devueltas automáticamente;

(3) Si hay señales recibidas, el módulo emitirá un pulso de alto nivel a través del puerto IO de ECHO, el tiempo de duración del impulso de alto nivel es el tiempo entre el envío y la recepción de la onda. Así el módulo puede conocer la distancia según el tiempo.

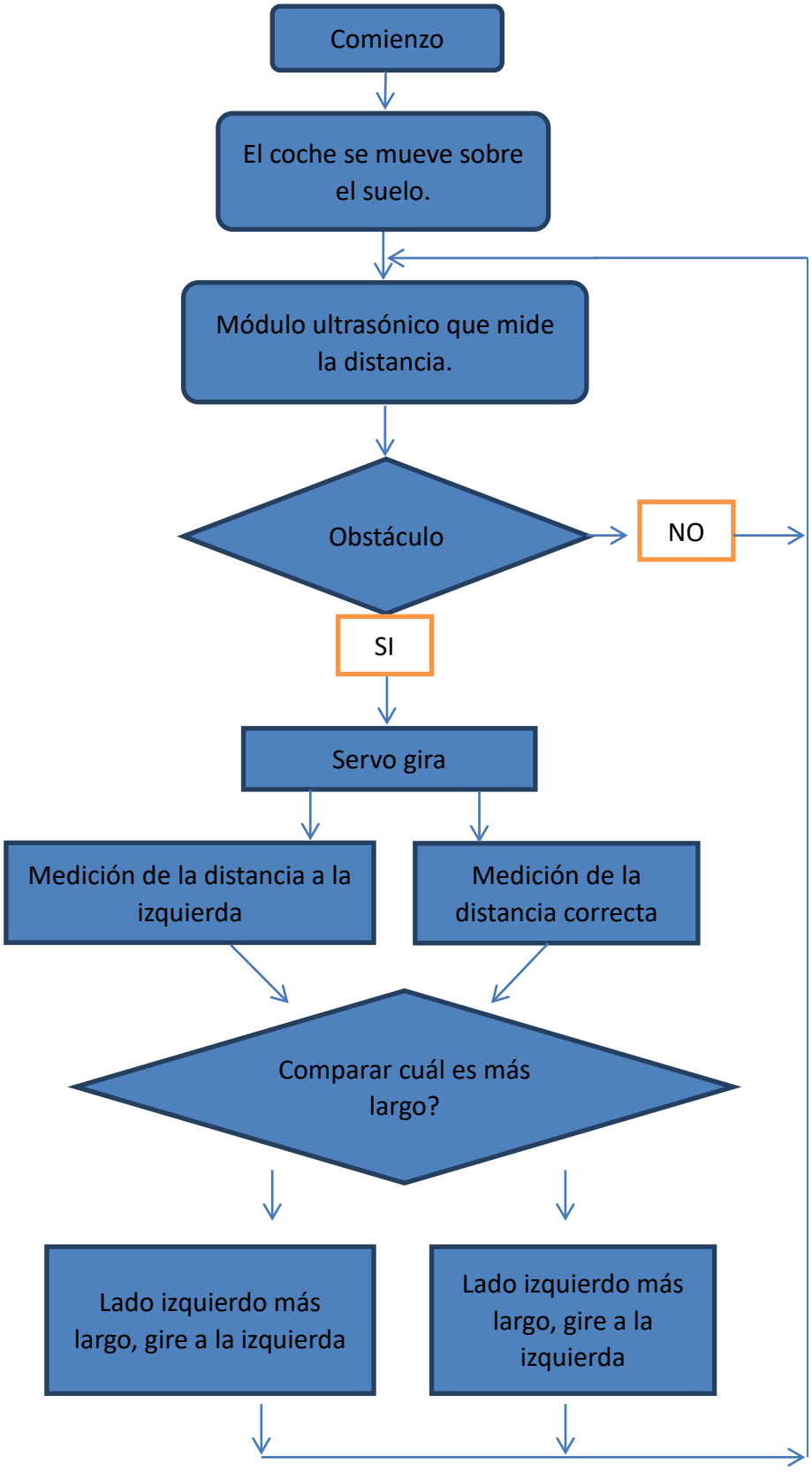
Testing distance= (high level time* velocity of sound (340M/S))/2);

Funcionamiento real:

El diagrama de sincronización se muestra a continuación. Sólo necesita suministrar un pulso corto a la entrada de disparo para iniciar el rango, y luego el módulo enviará una ráfaga de 8 ciclos de ultrasonido a 40 kHz y elevará su eco. El Echo es un objeto de distancia que es el ancho de pulso y el rango en proporción. Puede calcular el rango a través del intervalo de tiempo entre la señal de disparo de envío y la señal de eco de recepción. Fórmula: uS / 58 = centímetros o uS / 148 = pulgada; O: el rango = tiempo de alto nivel * velocidad (340M / S) / 2; Sugerimos utilizar más de 60ms de ciclo de medición, con el fin de evitar la señal de disparo a la señal de eco.




```
/*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;
}
```


Comienzo
El coche se mueve sobre el suelo.
Módulo ultrasónico que mide la distancia.
Obstáculo
NO
SI
Servo gira
Medición de la distancia a la izquierda
Medición de la distancia correcta
Comparar cuál es más largo?
Lado izquierdo más largo, gire a la izquierda
Lado izquierdo más largo, gire a la izquierda

De la imagen de arriba, podemos ver que el principio del coche de la evitación del obstáculo es muy simple. El módulo de sensor ultrasónico detectará la distancia entre el automóvil y los obstáculos una y otra vez y enviará los datos a la tarjeta de control, entonces el coche se detendrá y girará el servo para detectar el lado izquierdo y el lado derecho. Después de comparar la distancia desde el lado diferente, el coche gira hacia el lado que tiene mayor distancia y avanzar. Entonces el módulo del sensor ultrasónico detecta de nuevo la distancia.

```
if(rightDistance>leftDistance)
        {
          _mright();
          delay(360);
         }
         else if(rightDistance<leftDistance)
         {
          _mleft();
          delay(360);
         }
         else if((rightDistance<=20) | | (leftDistance<=20))
         {
          _mBack();
          delay(180);
         }
         else
         {
          _mForward();
         }
     }
     else
          _mForward();
}
```