





## *Lección4 Evitar obstaculos con el coche*

---




### ***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, evite que su coche sufra una colisión.*

#### ***Partes a aprender:***

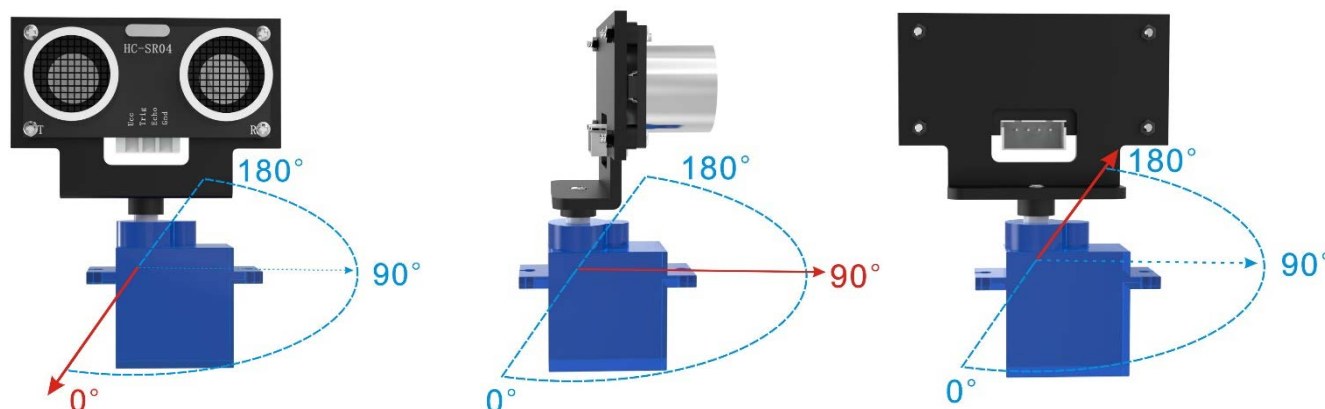
-  *Aprender a ensamblar el modulo de ultrasonidos*
-  *Familiarizarse con su manejo*
-  *Aprender sobre el principio para evitar colisiones*
-  *Usar el programa para que el vehiculo evite obstaculos*

#### ***Necesitaremos:***


-  *Un coche (con bacteria)*
-  *Un cable USB*
-  *Una carcasa de ultrasonidos*

## I . Conexión

*Cuando ensamble el soporte del módulo del sensor ultrasónico, el servo también debe ser depurado para asegurar que el servidor pueda girar 180 grados.*



**PASO1:** Conecte el UNO al ordenador y abra el archivo de código *Servo\_debug* en la ruta “\Lesson 4 Obstacle Avoidance Car\Servo\_debug\ Servo\_debug.ino”.

Elegoo Smart Robot Car Kit V3.0 > Lesson 4 Obstacle Avoidance Car > Servo_debug			
名称	修改日期	类型	大小
 Servo_debug.ino	2017/5/10 11:33	Arduino file	1 KB

```
Servo_debug | Arduino 1.8.2
File Edit Sketch Tools Help

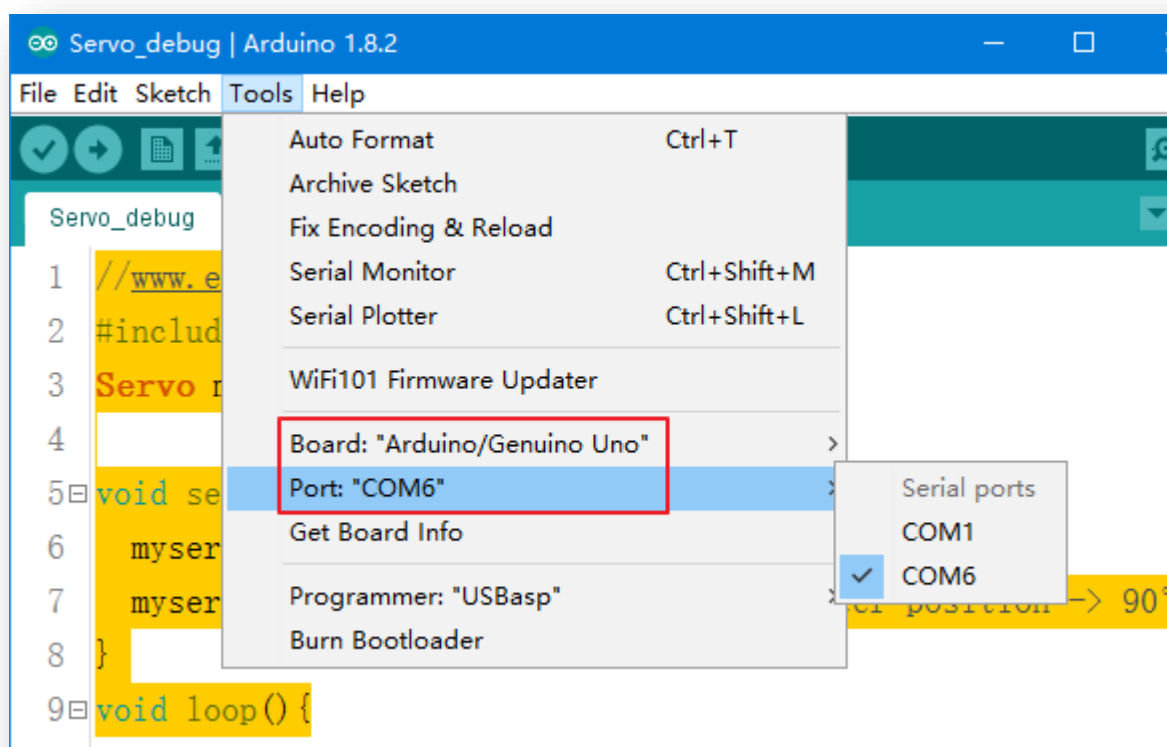
Servo_debug
1 //www.elegoo.com
2 #include <Servo.h>
3 Servo myservo;
4
5 void setup() {
6   myservo.attach(3);
7   myservo.write(90); // move servos to center position -> 90°
8 }
9 void loop() {
10
11 }
```

### Vista previa del código Servo\_debug:

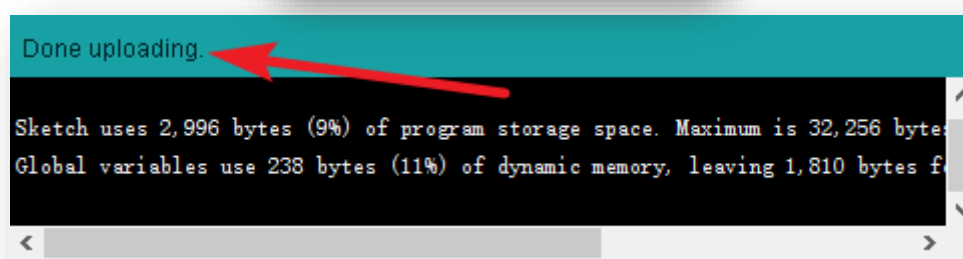
```
//www.elegoo.com
#include <Servo.h>
Servo myservo;

void setup(){
  myservo.attach(3);
  myservo.write(90); // move servos to center position -> 90°
}
void loop(){
}
```

**PASO2:** Pinchar en "Tool" --> "Port" y "Board" en el Arduino IDE.



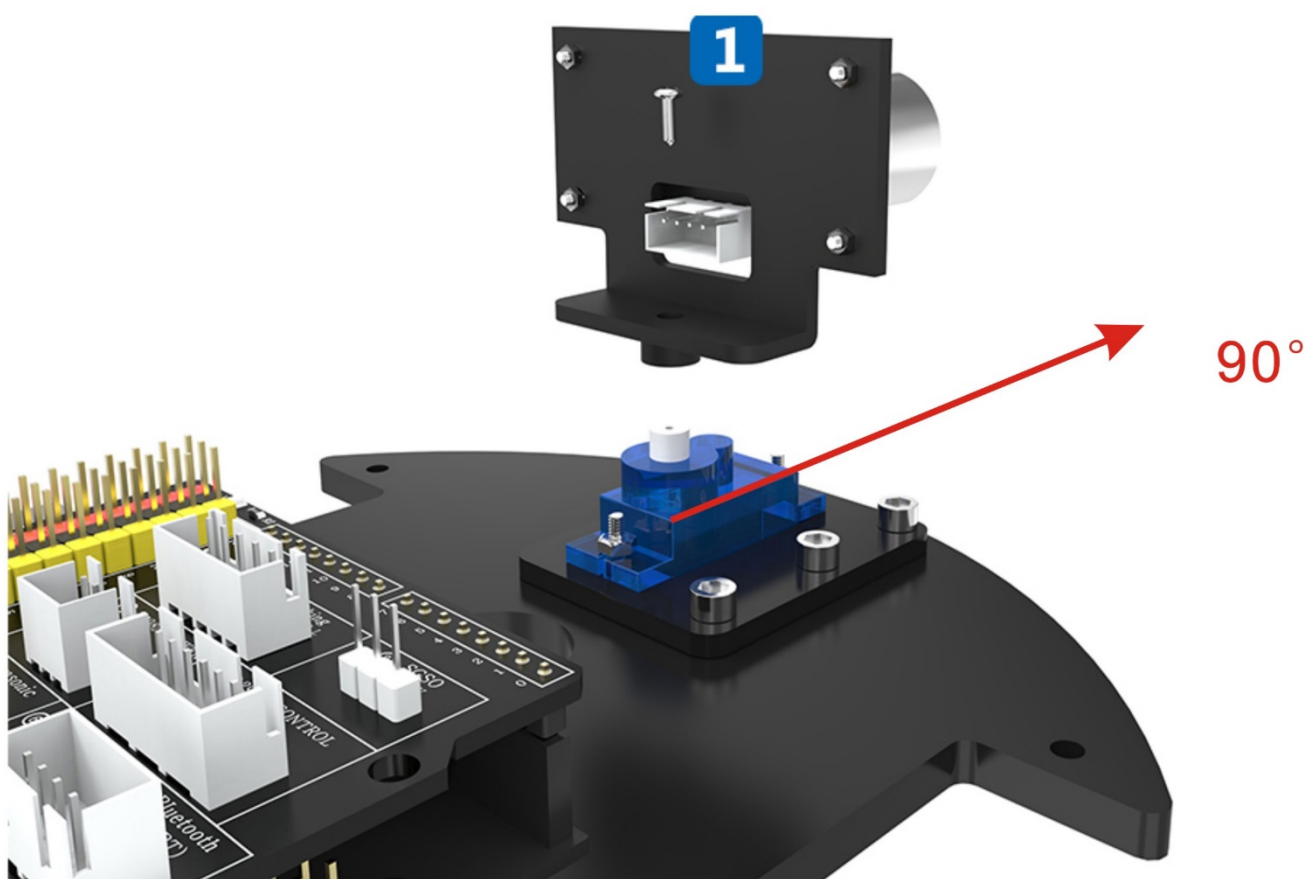
**PASO3:** Haga clic en la flecha para cargar el código en la placa de UNO.



Tras cargar el archive el servo girará 90 grados y se parará

#### ***PASO4: Ensamblar el modulo del sensor de ultrasonidos a 90 grados.***

El ángulo de cada dientes en el micro servo es de 15 grados y si lo instala en el medio de la dirección de 90 grados, girará a la izquierda o a la derecha por 15 grados, lo que significa que realmente el micro servo se está instalando a 15 o 105 grados.



#### ***FAQ sobre el servo motor.***

1 ¿Por qué el micro servo gira en sentido contrario a las agujas del reloj por 15 grados cada vez que enciendo la alimentación?

Es normal en el micro servo SG90 y no afectará al uso normal de los programas.

Si no lo ha controlado con programas, puede volver a girarlo a la normalidad con la mano o desconectar los cables conectados con el micro servo antes de encender la alimentación.

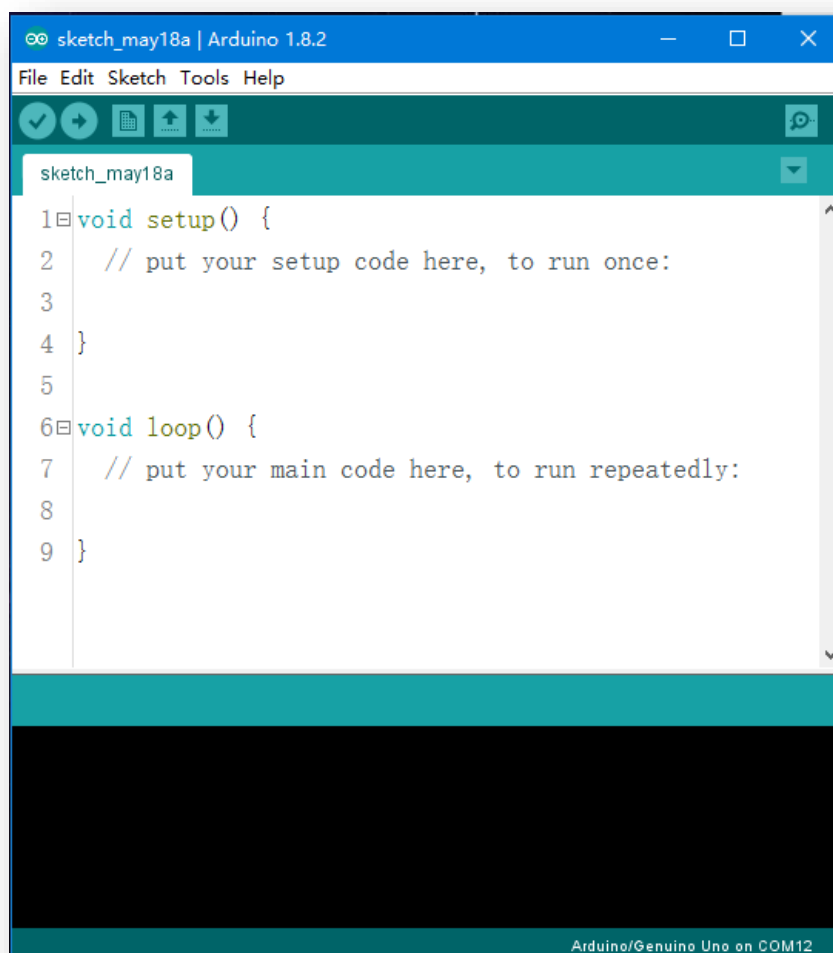
2 El microservo está fuera de control y sigue girando.

Use "myservo.write (angle)" to command the micron servo to the angle degree which has a range from 0 to 180. If it exceeds the range, the micro servo won't recognize this angle and will keep rotating.

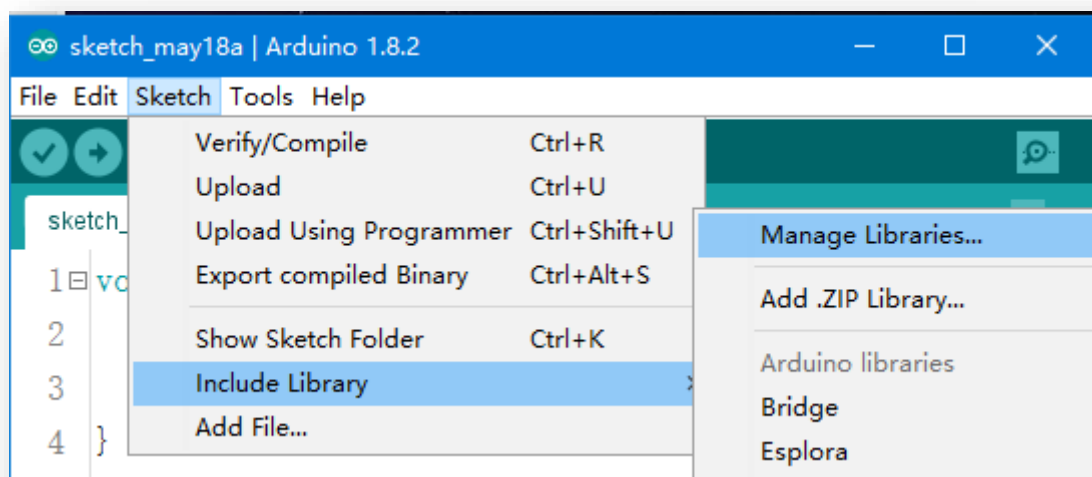
## II. Cargar el programa

Dado que el programa utiliza la librería <servo.h>, necesitaremos instalal antes de nada.

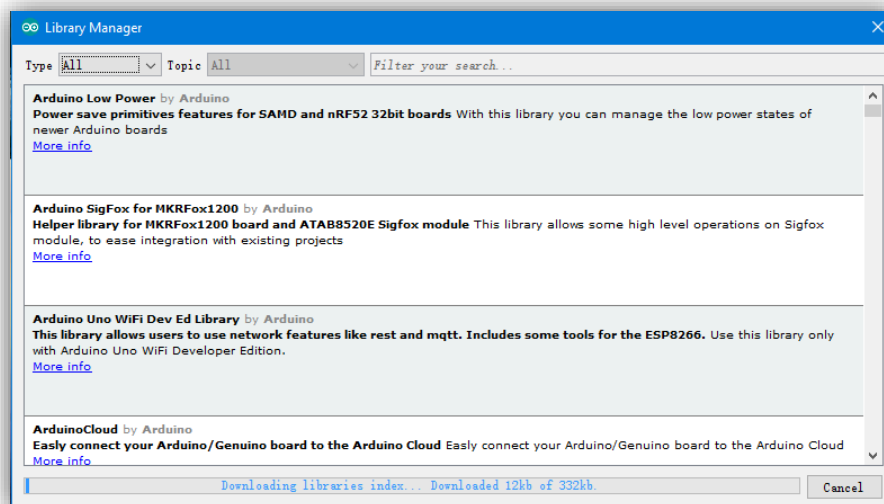
Abrimos el software de Arduino



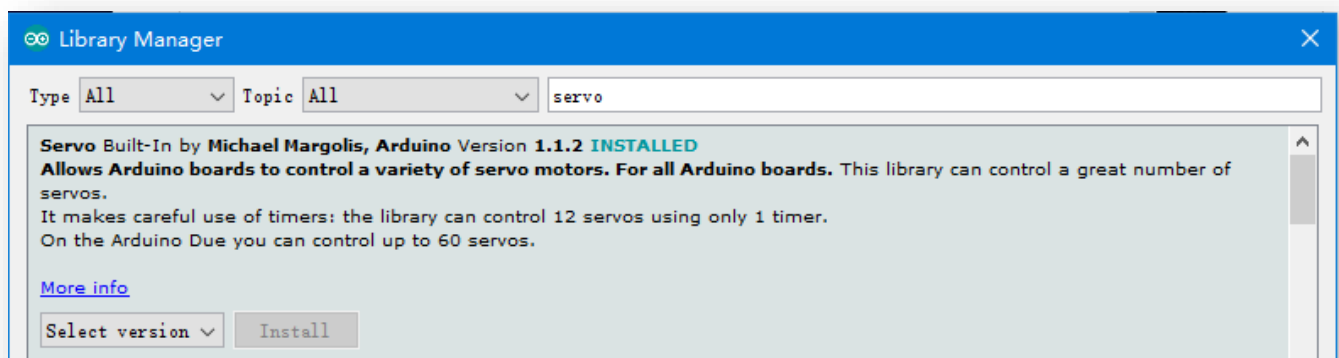
Seleccionar Sketch -> Include Library -> Manage Libraries...



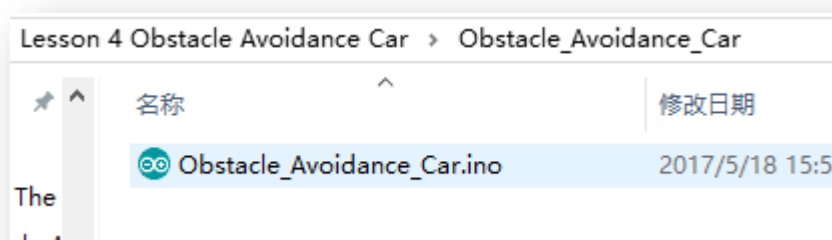
Esperar por “Downloading libraries index” para acabar.



Buscar servo y luego instalar la versión más reciente. La siguiente imagen muestra que la biblioteca Servo ya está instalada.



Conectar la placa UNO al pc y abrir el archive con el código de la ruta “\Lesson 4 Obstacle Avoidance Car\Obstacle\_Avoidance\_Car\Obstacle\_Avoidance\_Car.ino”. Cargar el programa en la placa UNO



Vista previa del código:

```
//www.elegoo.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 150
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 <= 20) {
        stop();
    }
}

```



```

delay(500);
myservo.write(10);
delay(1000);
rightDistance = Distance_test();

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

delay(500);
myservo.write(90);
delay(1000);
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();
}
}
}

```

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

### III. Introducción al principio

Antes de nada aprendamos acerca del servo SG90 :



Clasificación: 180 grados de rotación

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.

#### Como 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 ángulo. Cuando no hay diferencia entre los dos voltajes, el servo se detendrá.

#### Como controlar el servo:

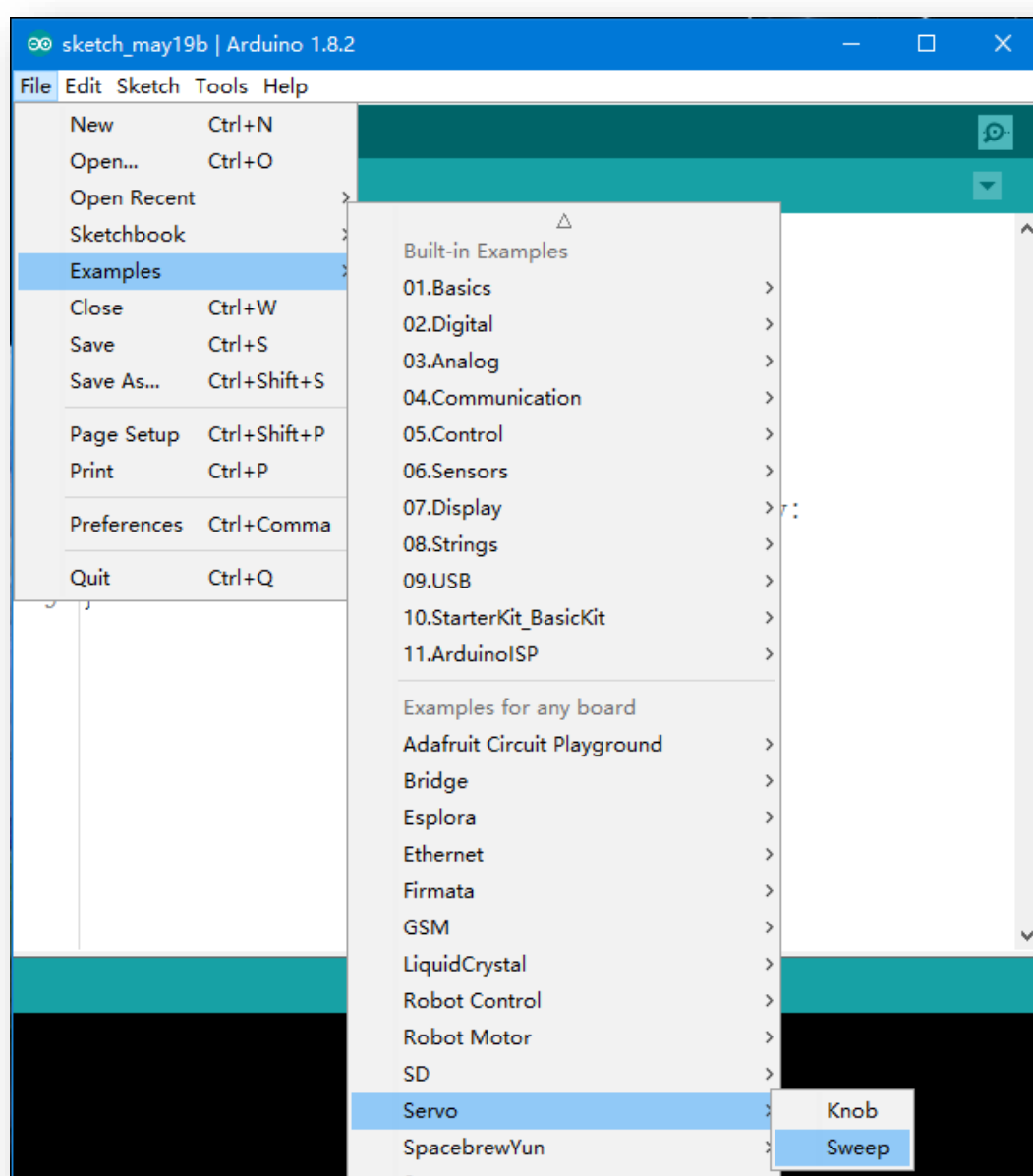
Para controlar la rotación del servo, es necesario que el pulso de tiempo sea de unos 20 ms y el ancho de pulso de nivel alto sea de aproximadamente 0,5 ms ~ 2,5 ms, lo cual es consistente con el ángulo limitado del servo.

Tomando el servo de 180 grados, 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

### Un programa de ejemplo

Abrir Arduino IDE y seleccionar “File->Examples->Servo->Sweep”



Ahora echemos un vistazo al sensor de ultrasonidos



**Características del módulo:** Testear distancia, módulo de gran precisión

**Aplicaciones del producto:** Evitación de obstaculos, testeo de distancia a objetos, testeo de liquidos, seguridad publica, pruebas de aparcamiento.

#### Principales datos técnicos

- (1): Voltage usado: DC---5V
- (2): Corriente: less than 2mA
- (3): Nivel de salida: Superior a 5V
- (4): Nivel de salida: Inferior a 0
- (5): Angulo de detección: No superior a 15 grados
- (6): Distancia de testeo: 2cm-450cm
- (7): Alta precisión: Superior a 0.2cm

Metodo de conexión de cables: VCC trig (el fin del control), echo (el final de la recepción), GND

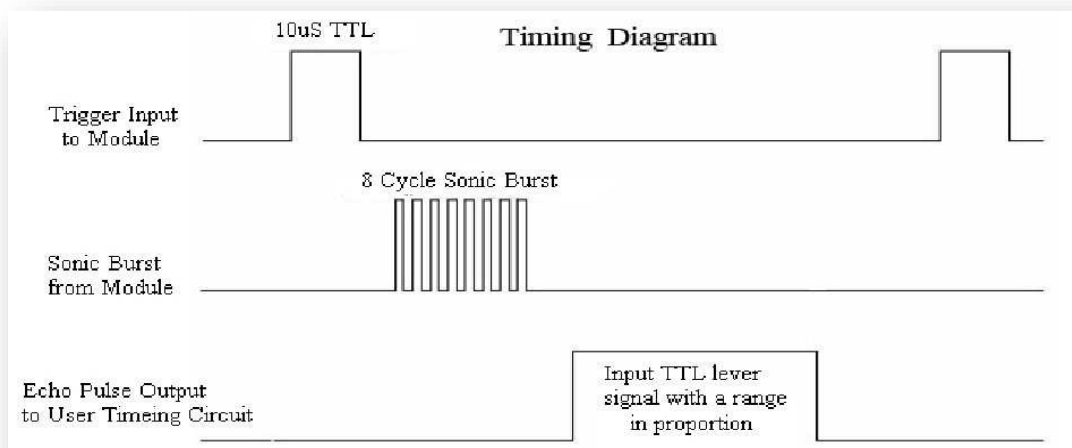
#### Cómo funciona el modulo:

- (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 40KHz automáticamente, prueba si hay señales devueltas automáticamente;
- (3) Si hay señales recibidas, el módulo emitirá un impulso de alto nivel a través del puerto IO de ECHO, el tiempo de duración del pulso 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.

**Distancia de testeo= (Tiempo nivel alto \* Velocidad del sonido (340M/S))/2;**

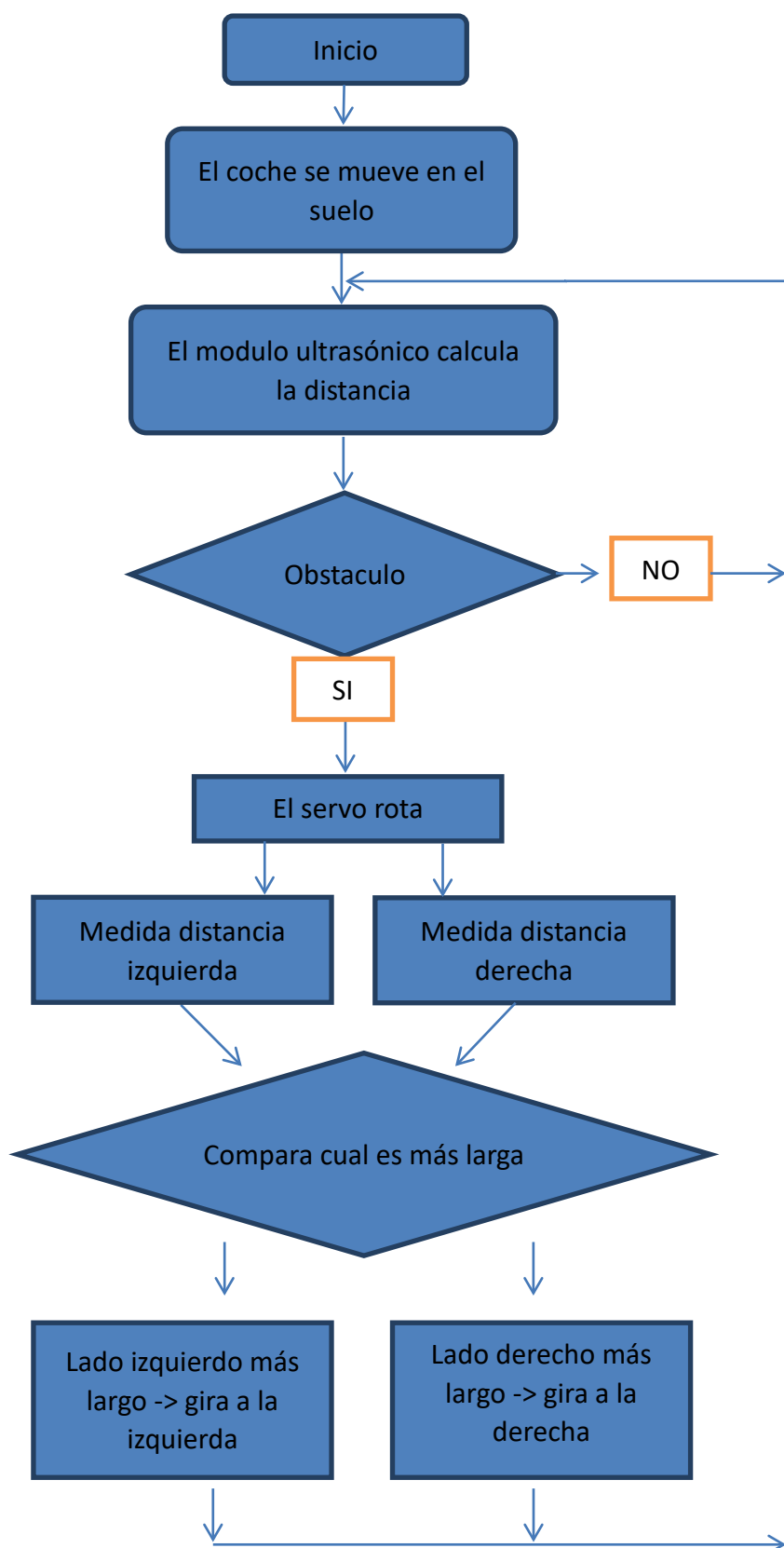
### Funcionamiento real:

El diagrama de sincronización se muestra a continuación. Sólo necesita que se le suministre un pulso corto a la entrada del disparador para iniciar el rango y luego el módulo enviará una ráfaga de 8 ciclos de ultrasonido a 40 kHz y elevará su eco. El Eco es un objeto de distancia que es proporcional al ancho de pulso y el rango. Se puede calcular el rango a través del intervalo de tiempo desde que se envía la señal de disparo y se recibe la señal de eco. Fórmula:  $\mu\text{S} / 58 = \text{centímetros}$  o  $\mu\text{S} / 148 = \text{pulgada}$ ; o: el rango = tiempo de alto nivel \* velocidad (340M / S) / 2; sugerimos utilizar un ciclo de medición de más de 60ms, con el fin de prevenir 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;
}
```



En la imagen de arriba, podemos ver que el principio de la evitación de obstáculo en el coche es muy simple. El módulo del sensor ultrasónico detectará la distancia entre el coche y los obstáculos una y otra vez y enviará los datos a la placa del controlador, luego el automóvil se detendrá y girará el servo para detectar el lado izquierdo y el lado derecho. Después de comparar la distancia desde cada lado, el coche gira hacia el lado que tiene una mayor distancia y avanza. Entonces el módulo del sensor ultrasónico detectará de nuevo la distancia.

### Vista previa del código:

```
if(rightDistance > leftDistance) {
    right();
    delay(360);
}
else if(rightDistance < leftDistance) {
    left();
    delay(360);
}
else if((rightDistance <= 20) || (leftDistance <= 20)) {
    back();
    delay(180);
}
else {
    forward();
}
```