





Lezione 2 evitare gli ostacoli

Punti di questa sezione




La gioia di imparare, non è solo sapere come controllare la tua auto, ma anche sapere come proteggere la tua auto.

Quindi, fai in modo che la tua auto sia lontana dalle collisioni.

punti di apprendimento:

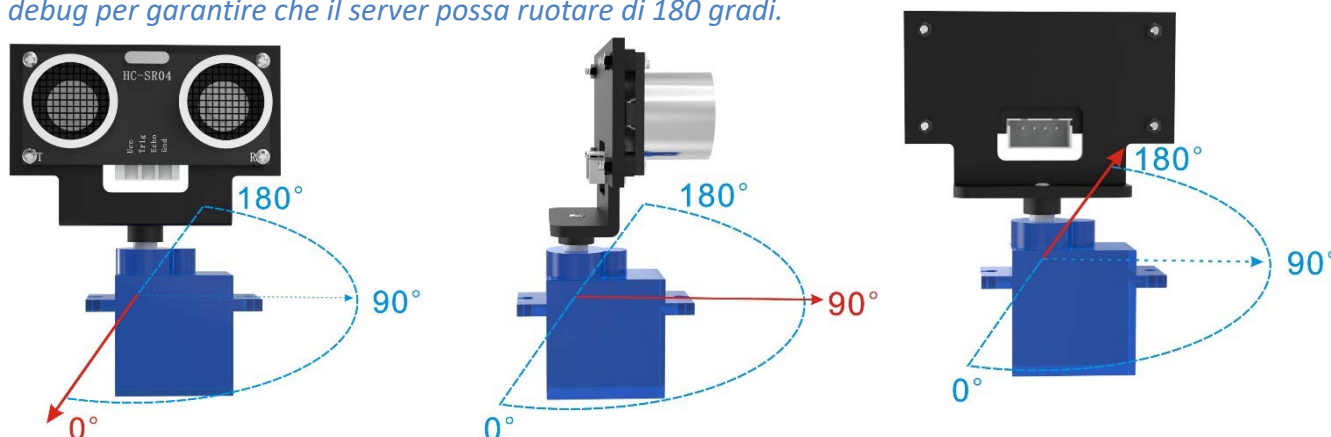
-  *Assemblare un modulo a ultrasuoni*
-  *Avere familiarità con l'utilizzo dello sterzo*
-  *I principi dell'evitare ostacoli*
-  *Usa il programma per fare in modo che l'auto eviti gli ostacoli*

Preparativi:


-  *Auto (con batteria)*
-  *Cavo usb*
-  *Un sensore a ultrasuoni*

I . Connection

Quando si monta il supporto del modulo sensore ad ultrasuoni, il servo deve anche essere sottoposto a debug per garantire che il server possa ruotare di 180 gradi.



STEP1: Connetti UNO al computer e apri Servo_debug code file dalla cartella “\Lesson 2 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 }
```

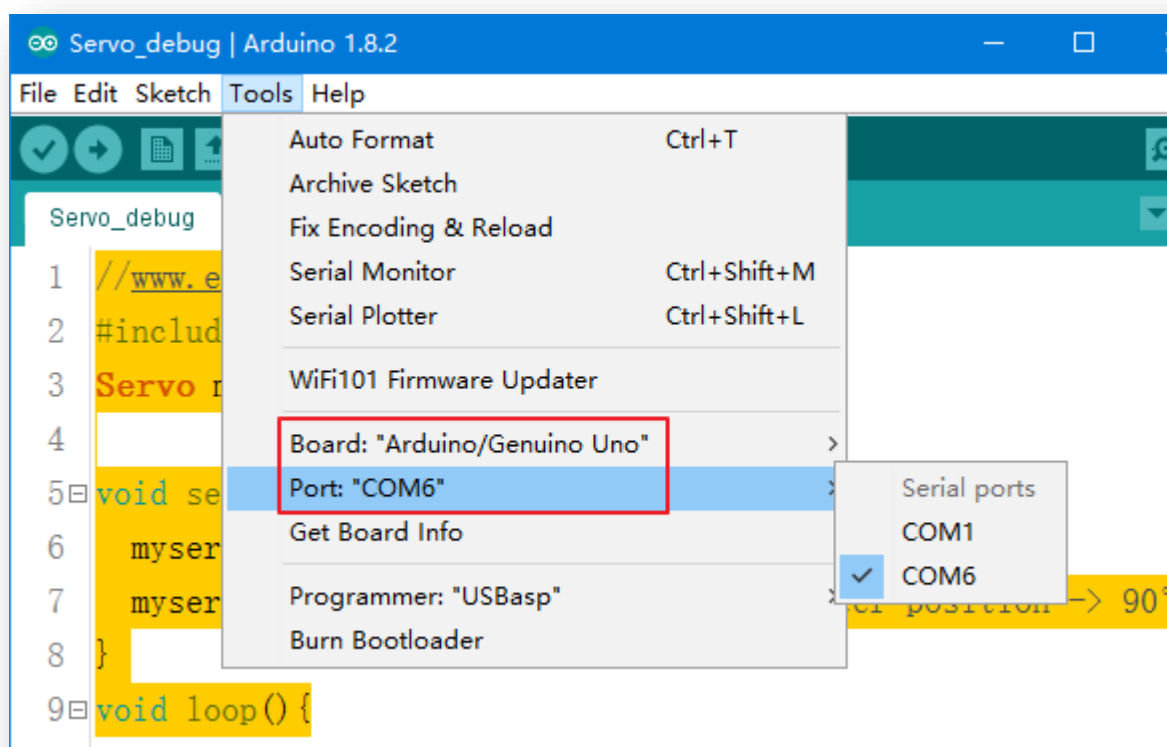
Servo_debug code preview:

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

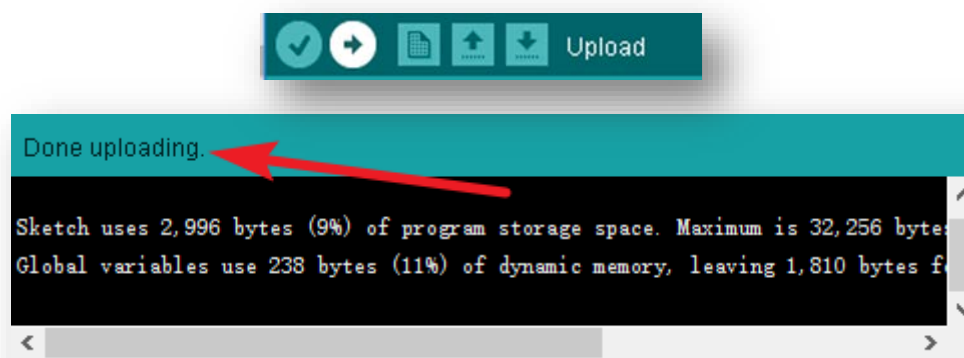
void setup(){
  myservo.attach(3);
  myservo.write(90); // move servos to center position -> 90°
}

void loop(){
}
```

STEP2: Seleziona "Tool" --> "Port" and "Board" nell' Arduino IDE.



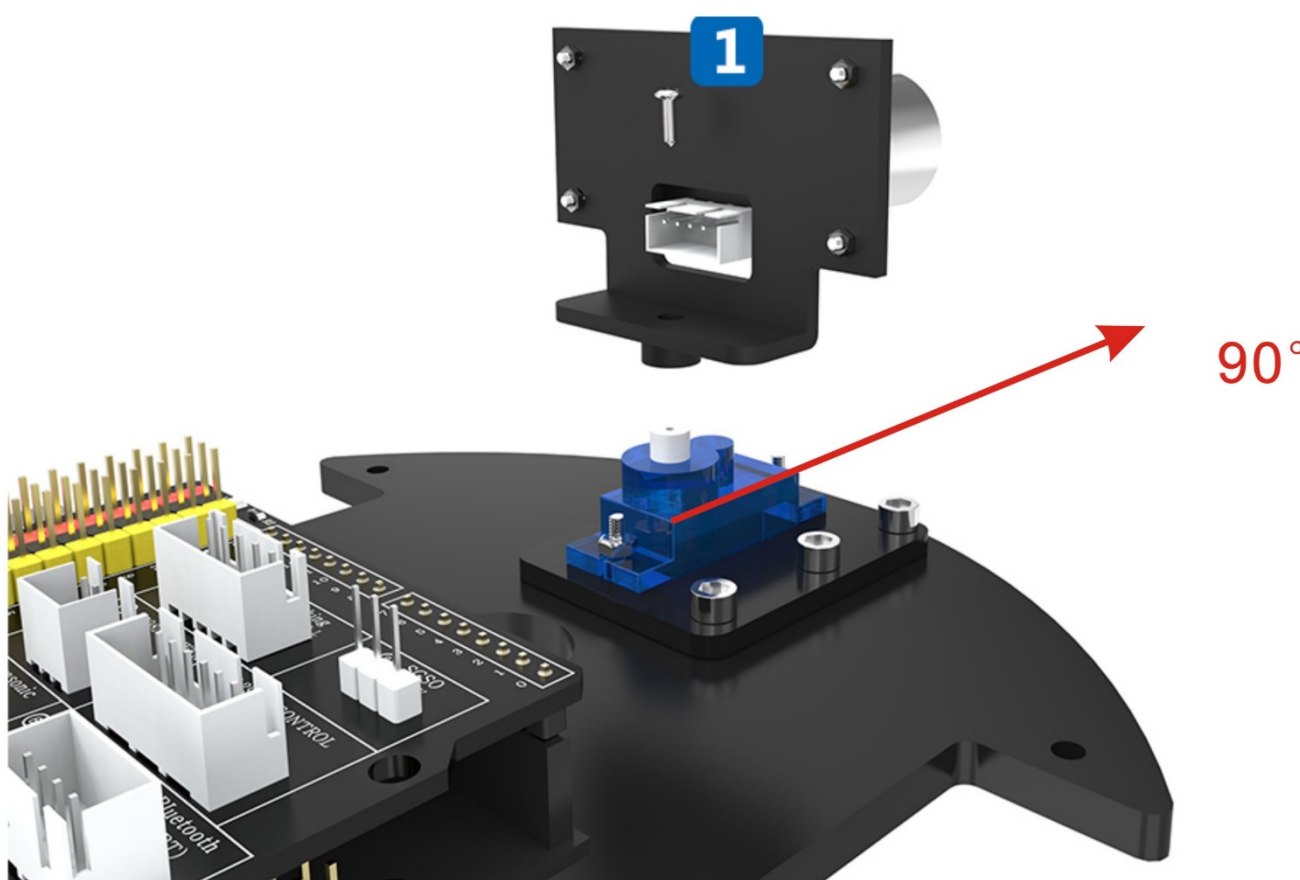
STEP3: Fare clic sul pulsante con le frecce per caricare il codice sulla scheda del controller UNO.



Dopo averlo caricato , il servo ruoterà di 90 degrees e si fermerà

STEP4: Assemblare il modulo a ultrasuoni a 90 gradi.

L'angolo di ogni dente sul micro servo è di 15 gradi e se lo installi al centro della direzione di 90 gradi, ruoterà verso sinistra o verso destra di 15 gradi, il che significa che il grado effettivo di installazione del micro servo è di 15 gradi o 105 gradi.



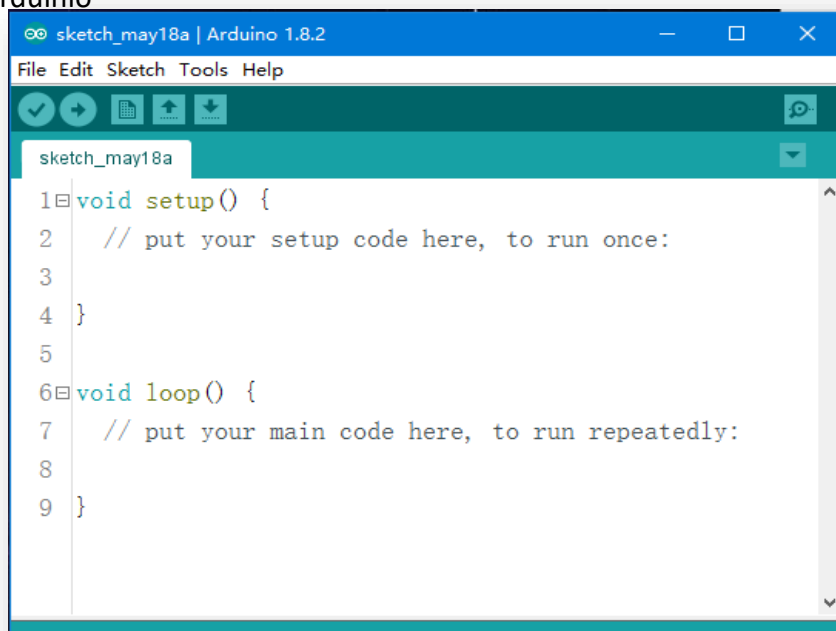
FAQ riguardo il servo motore.

- 1 Perché il micro servo ruota in senso antiorario di 15 gradi ogni volta che accendo l'alimentazione? Questo è normale al micro servo SG90 e non influenzerà il normale utilizzo con i programmi. Se non lo hai controllato con i programmi, puoi ruotarlo alla normalità con la mano o scollegare i fili collegati con il micro servo prima di accenderlo.
- 2 Il micro servo è fuori controllo e continua a ruotare. Usa "myservo.write (angle)" per comandare il micron servo al grado dell'angolo che ha un range da 0 a 180. Se supera il range, il micro servo non riconoscerà questo angolo e continuerà a ruotare.

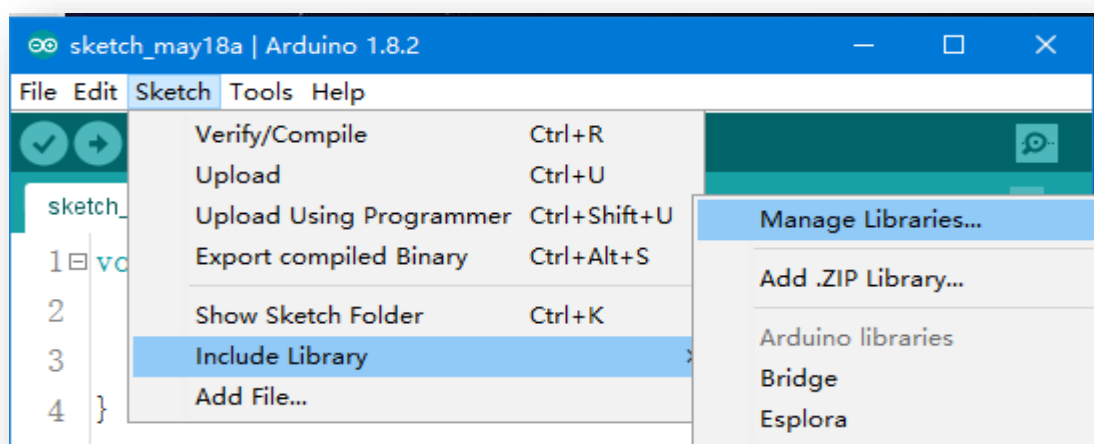
II. Caricare il programma

Siccome il programma usa le librerie <servo.h>, abbiamo bisogno di installare le librerie.

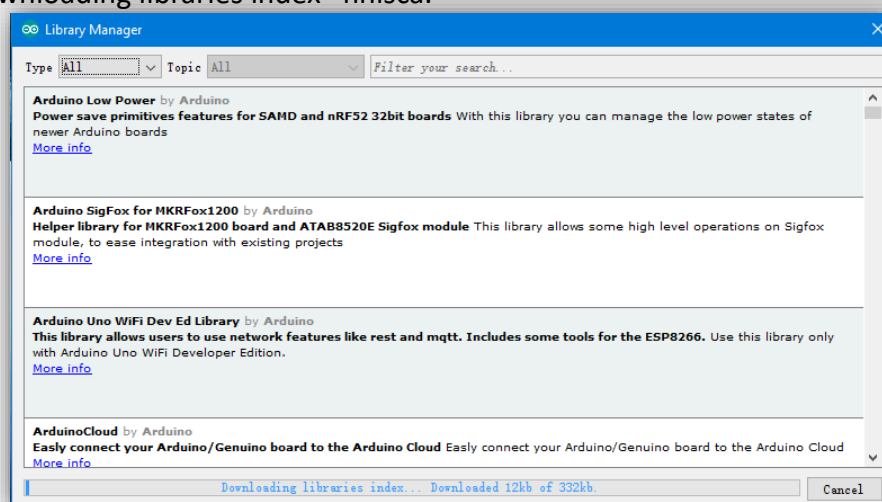
Apri il software Arduino



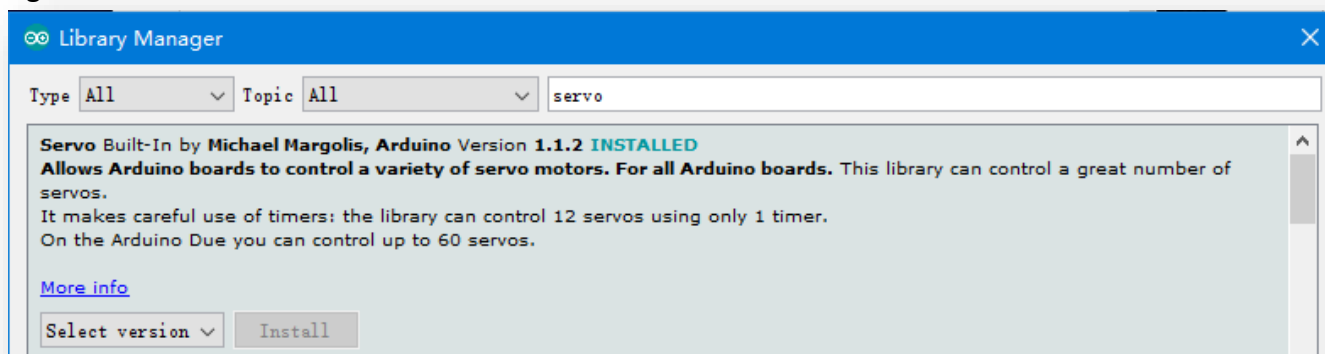
Seleziona Sketch -> Include Library -> Manage Libraries...



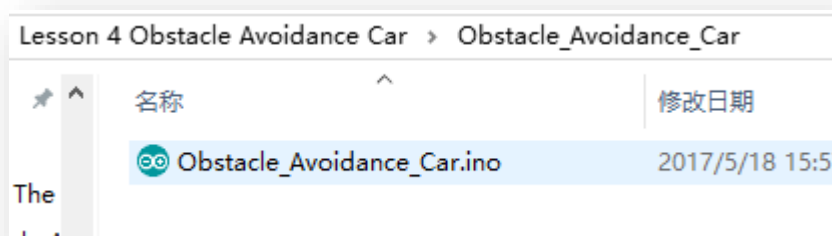
Aspetta che "Downloading libraries index" finisca.



Cerca servo e quindi installa la versione più recente. L'immagine seguente mostra che la libreria Servo è già installata.



Collegare la scheda di controllo UNO al computer, open the code file in the path “\Lesson 2 Obstacle Avoidance Car\Obstacle_Avoidance_Car\Obstacle_Avoidance_Car.ino”. Carica il programma nella scheda UNO



Anteprima del codice:

```
//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();  
}  
}
```

Dopo aver caricato il programma sulla scheda di controllo UNO, scollegare il cavo, mettere il veicolo a terra e inserire l'alimentazione.

Vedrai che il veicolo si sposterà in avanti e la piattaforma cloud continua a ruotare per far funzionare i sensori di misurazione della distanza in modo continuo. Se ci sono ostacoli avanti, la piattaforma cloud si fermerà e il veicolo cambierà direzione per superare l'ostacolo. Dopo aver aggirato l'ostacolo, la piattaforma continua a ruotare e anche il veicolo si muoverà.

III. Introduzione ai principi

Prima di tutto, conosciamo il Servo SG90:



Classificazione: 180 sterzo

Normalmente il servo ha 3 linee di controllo: alimentazione, terra e segnale.

Definizione dei pin del servo: linea marrone - GND, linea rossa - 5V, segnale arancione.

Come funziona il servo:

Il chip di modulazione del segnale nel servo riceve i segnali dalla scheda controller, quindi il servo ottiene la tensione DC di base. C'è anche un circuito di riferimento all'interno del servo che produrrà una tensione standard. Queste due tensioni si confronteranno e la differenza verrà emessa. Quindi il chip motore riceverà la differenza e deciderà la velocità di rotazione, la direzione e l'angolo. Quando non c'è differenza tra le due tensioni, il servo si fermerà.

Come controllare il servo:

Per controllare la rotazione del servo, è necessario fare in modo che l'impulso del tempo sia di circa 20ms e la larghezza dell'impulso di alto livello sia di circa 0,5ms ~

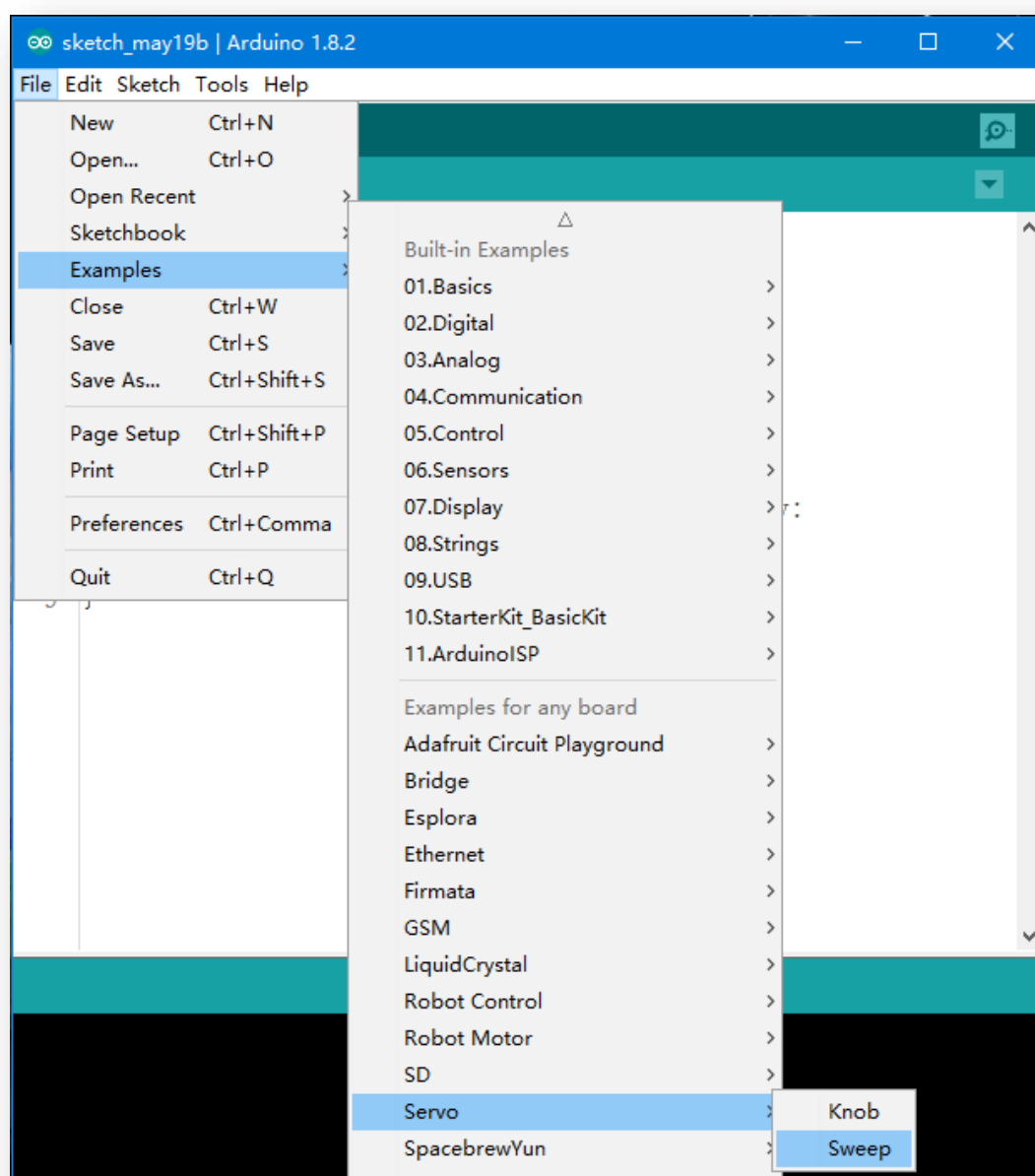
2,5ms, che è coerente con l'angolo limitato del servo.

Prendendo 180 servo angolari per esempio, la relazione di controllo corrispondente è la seguente:

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

Programma di esempio :

Apri Arduino IDE e seleziona “File->Examples->Servo->Sweep”



Quindi, diamo un'occhiata al modulo sensore ad ultrasuoni.



Caratteristica del modulo: distanza di prova, modulo di alta precisione.

Applicazione dei prodotti: prevenzione degli ostacoli del robot, distanza di prova degli oggetti, test dei liquidi, sicurezza pubblica, test dei parcheggi.

Parametri tecnici principali

- (1): tensione utilizzata: DC --- 5V
- (2): corrente statica: inferiore a 2 mA
- (3): livello di uscita: superiore a 5V
- (4): livello di uscita: inferiore a 0
- (5): angolo di rilevamento: non superiore a 15 gradi
- (6): distanza di rilevamento: 2cm-450cm
- (7): alta precisione: fino a 0,2 cm

Metodo di collegamento delle linee: VCC, trig (la fine del controllo), echo (la fine della ricezione), GND

Come funziona il modulo:

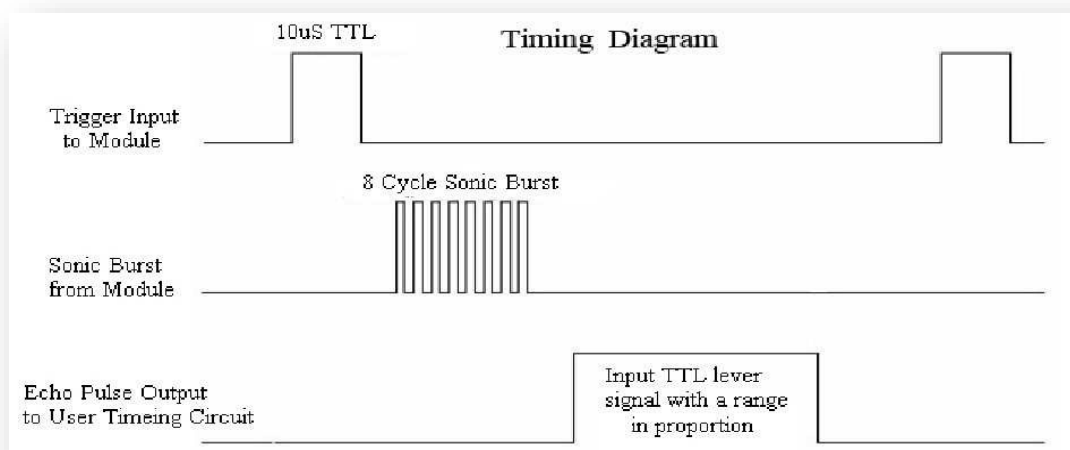
- (1) Applicare la porta IO di TRIG per attivare il range, fornire un segnale di alto livello, almeno 10us una volta;
- (2) Il modulo invia automaticamente 8 onde quadrate di 40kHz, verifica se ci sono segnali restituiti automaticamente;
- (3) Se ci sono segnali ricevuti, il modulo emetterà un impulso di alto livello attraverso la porta IO di ECHO, il tempo di durata dell'impulso di alto livello è il tempo tra l'invio e la ricezione delle onde. Quindi il modulo può conoscere la distanza in base al tempo.

Distanza di prova = (tempo di alto livello * velocità del suono (340M / S)) / 2);

Operazione reale:

Il diagramma dei tempi è mostrato sotto. È sufficiente fornire un impulso short10uS all'ingresso trigger per avviare l'intervallo, quindi il modulo invierà un ciclo di ultrasuoni a 8 cicli a 40 kHz e aumenterà l'eco. L'eco è un oggetto a distanza che è l'ampiezza dell'impulso e l'intervallo in proporzione. È possibile calcolare l'intervallo nell'intervallo

di tempo tra l'invio del segnale di trigger e la ricezione del segnale di eco. Formula: $\mu\text{s} / 58 = \text{centimetri}$ o $\mu\text{s} / 148 = \text{pollici}$; oppure: l'intervallo = tempo di livello alto * velocità $(340\text{M} / \text{S}) / 2$; suggeriamo di utilizzare un ciclo di misurazione di oltre 60 ms, al fine di prevenire il segnale di trigger sul segnale di eco.



/*Ultrasonic distance measurement Sub function*/

int Distance_test()

{

digitalWrite(Trig, LOW);

delayMicroseconds(2);

digitalWrite(Trig, HIGH);

delayMicroseconds(20);

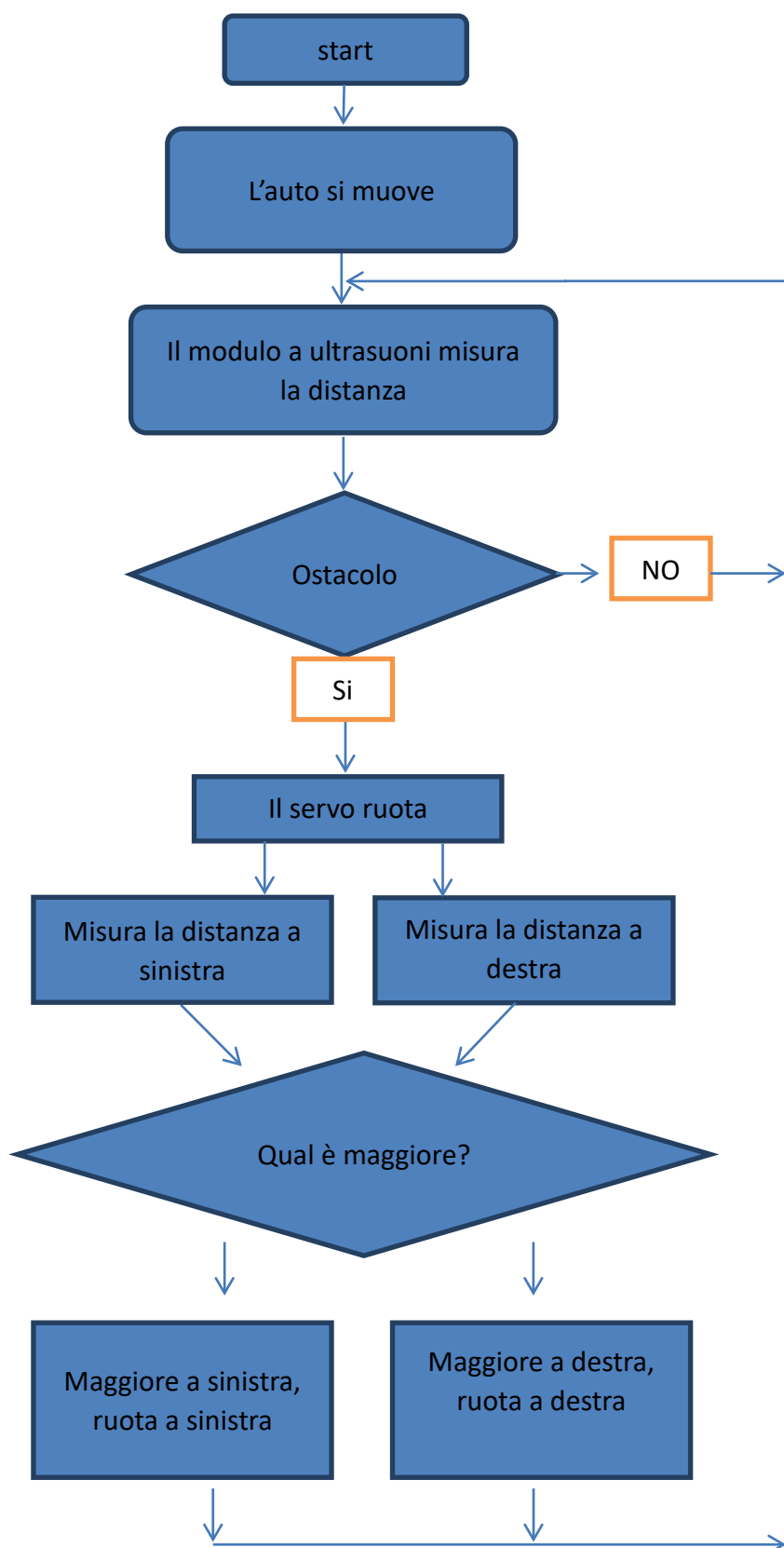
digitalWrite(Trig, LOW);

float Fdistance = pulseIn(Echo, HIGH);

Fdistance= Fdistance/58;

return (int)Fdistance;

}



Dall'immagine sopra, possiamo vedere che il principio dell'ostacolo per evitare gli ostacoli è molto semplice. Il modulo sensore ad ultrasuoni rileverà la distanza tra l'auto e gli ostacoli ancora e ancora e invierà i dati alla scheda controller, quindi l'auto si fermerà e ruoterà il servo per rilevare il lato sinistro e il lato destro. Dopo aver confrontato la distanza dal lato diverso, l'auto gira sul lato che ha una distanza maggiore e va avanti. Quindi il modulo sensore ad ultrasuoni rileva nuovamente la distanza.

Anteprima del codice:

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