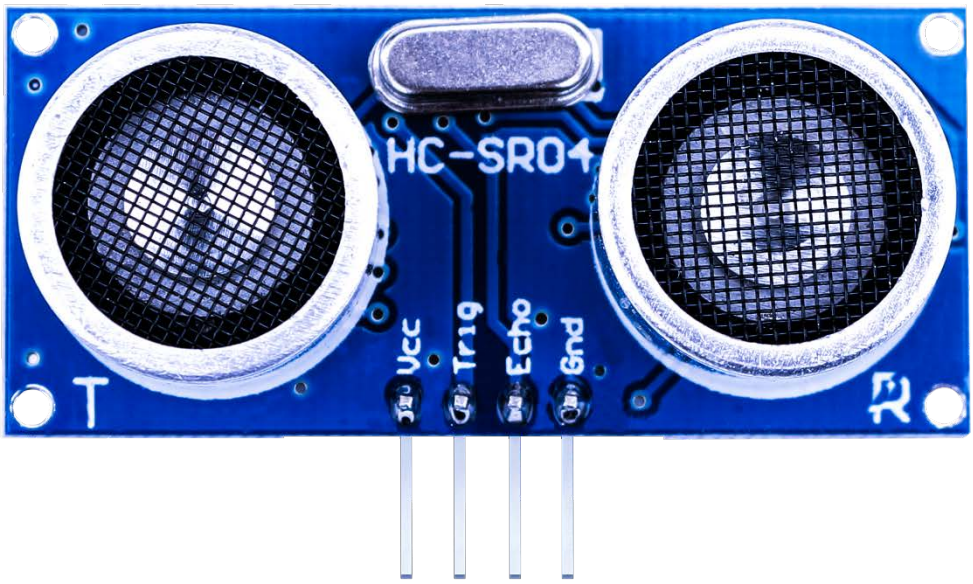
Lezione4 La macchina che evita gli ostacoli



Punti della sezione

Il divertimento dell'apprendere, non e' solo sapere come controllare la vostra macchina, ma anche sapere come proteggerla. Quindi teniamo la macchina lontano dagli urti.

Parti di apprendimento:

- Imparare ad assemblare il modulo ad ultrasuoni
- Prendere familiarita' con lo sterzo
- Imparare I principi dell'evitamento degli ostacoli
- Realizzare un programma che attui l'evitamento degli ostacoli

Componenti necessari:

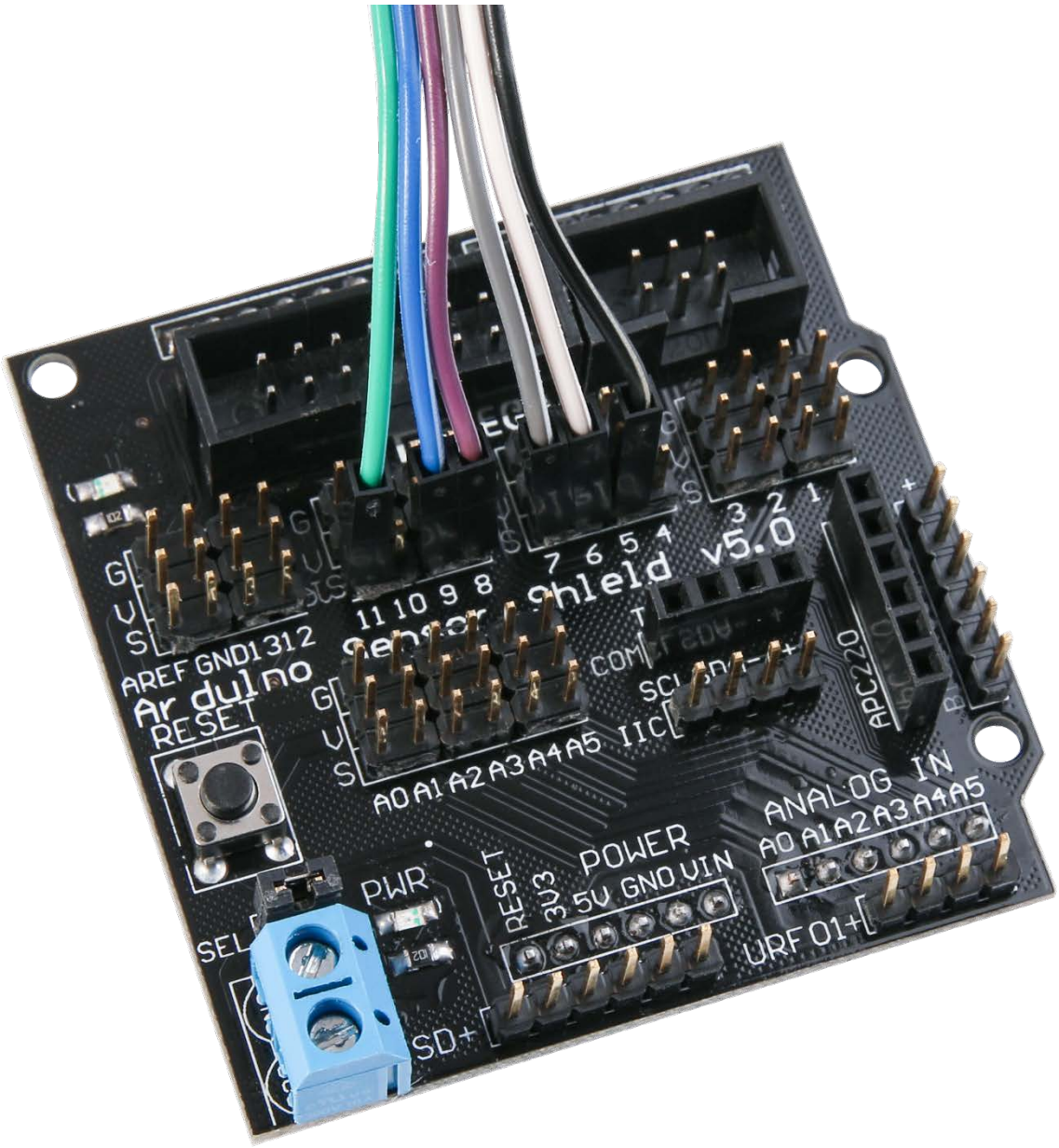
- Una macchina (con la batteria)
- Un cavo USB
- Alcuni sensori ad ultrasuoni

Ⅰ. Connessioni

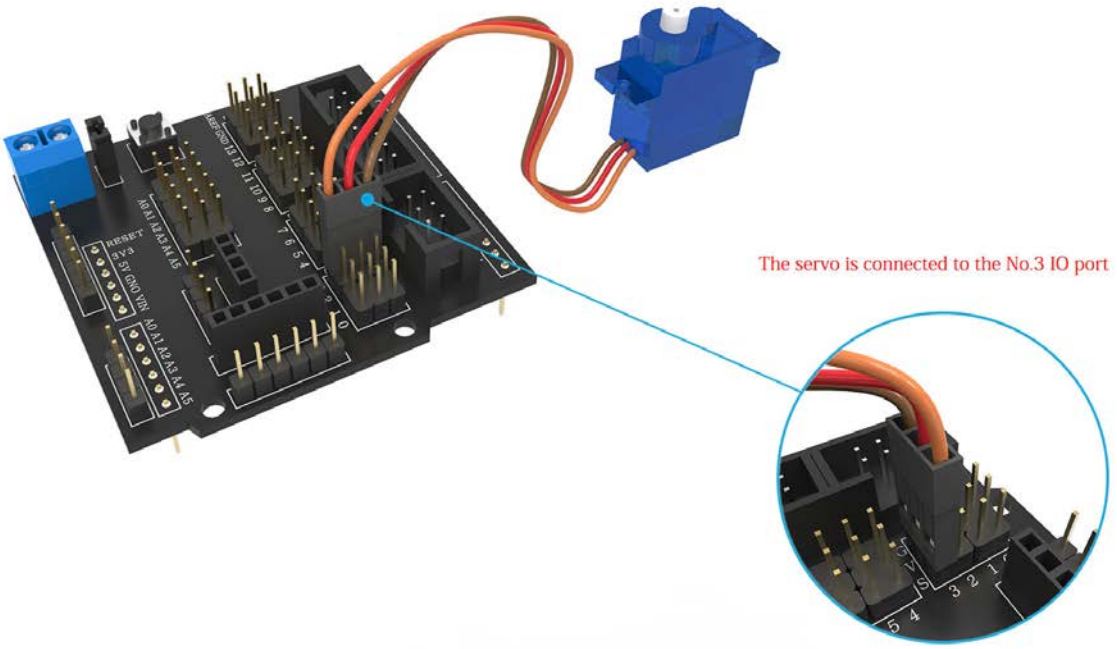
L298N

La libreria Servo supporta fino a 12 motori su molte delle chede Arduino e 48 su Arduino Mega. Su schede diverse dal Mega, l'uso della libreria disabilita la funzionalita' analogWrite() (PWM) sui pin 9 e 10, sia che ci sia o no un Servo su questi pin. Sul Mega, fino a 12 servo possono essere usati senza interferire con la funzionalita' del ; usare da 12 a 23 motori disabilitera' il PWM sui pin 11 e 12.

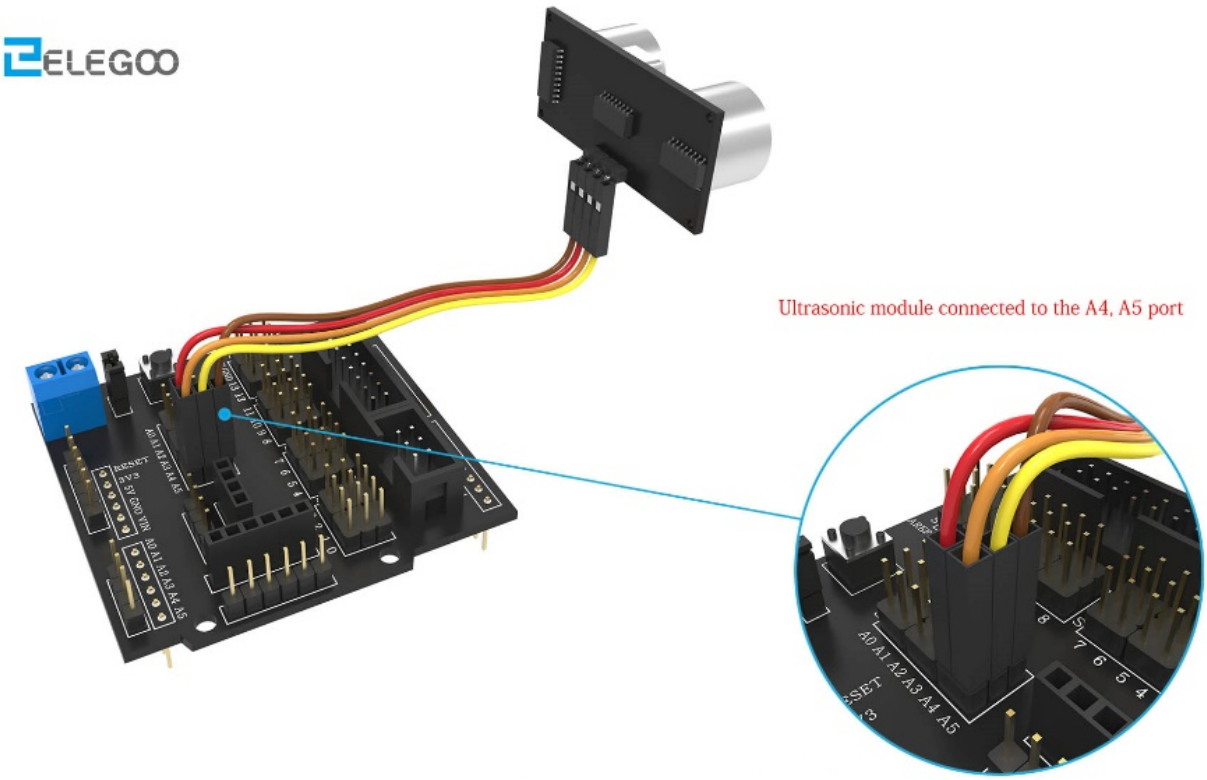
Quindi si rendera' necessario spostare l' ENA Enable dal pin digitale 10 all'11

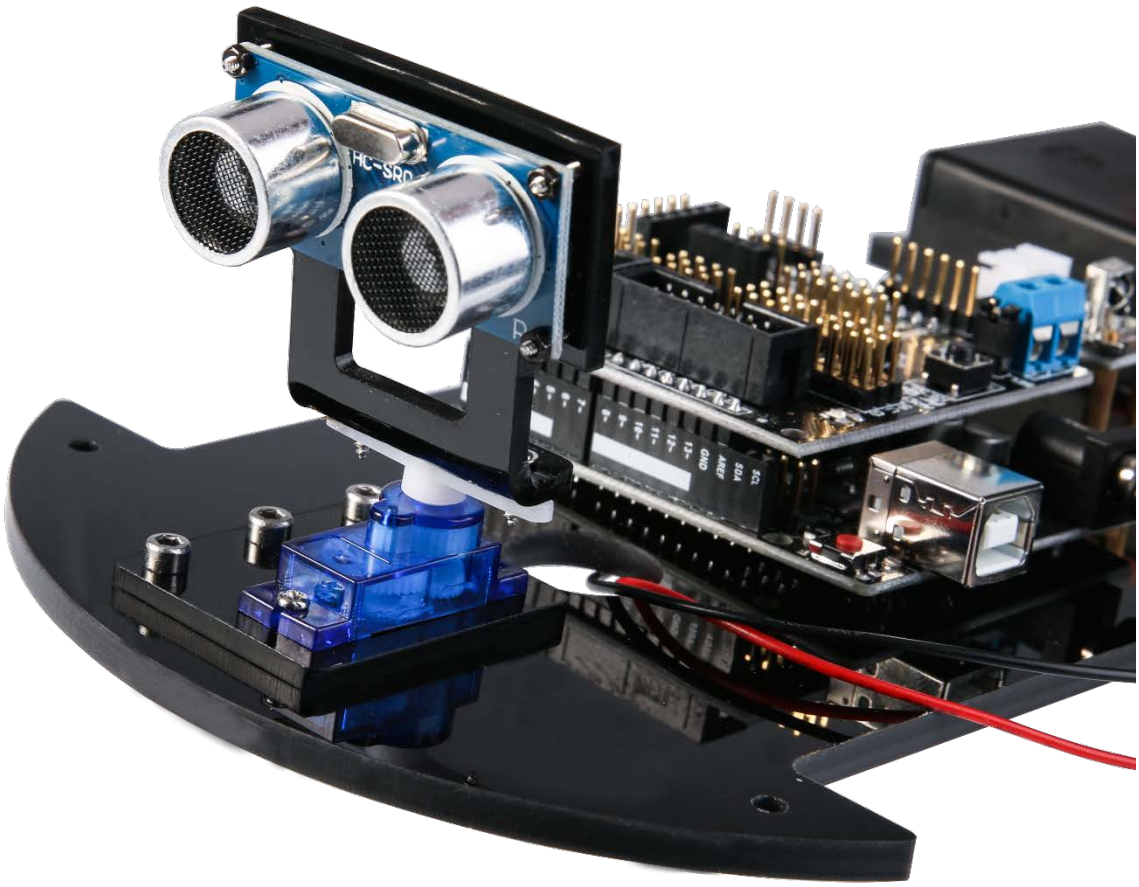


servomeccanismo



Modulo ad ultrasuoni





Ⅱ. Fate l'upload del seguente programma

```
#include <Servo.h> //servo library
Servo myservo; // create servo object to control servo
int Echo = A4;
int Trig = A5;
int in1 = 9;
int in2 = 8;
int in3 = 7;
```

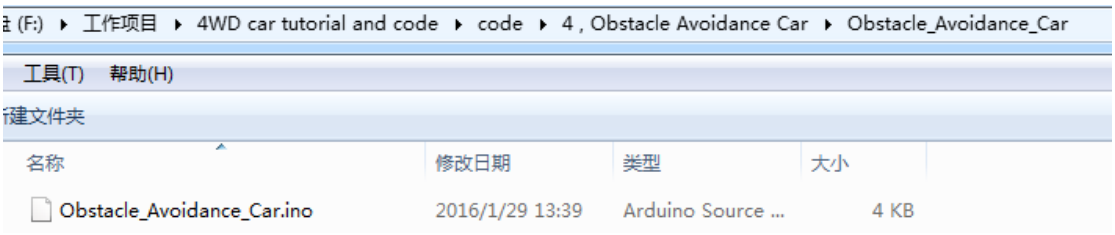
```
int in4 = 6;
int ENA = 5;
int ENB = 11;
int ABS = 130;
int rightDistance = 0,leftDistance = 0,middleDistance = 0 ;
void _mForward()
{
  analogWrite(ENA,ABS);
  analogWrite(ENB,ABS);
    digitalWrite(in1,HIGH);//digital output
    digitalWrite(in2,LOW);
    digitalWrite(in3,LOW);
    digitalWrite(in4,HIGH);
  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);
```

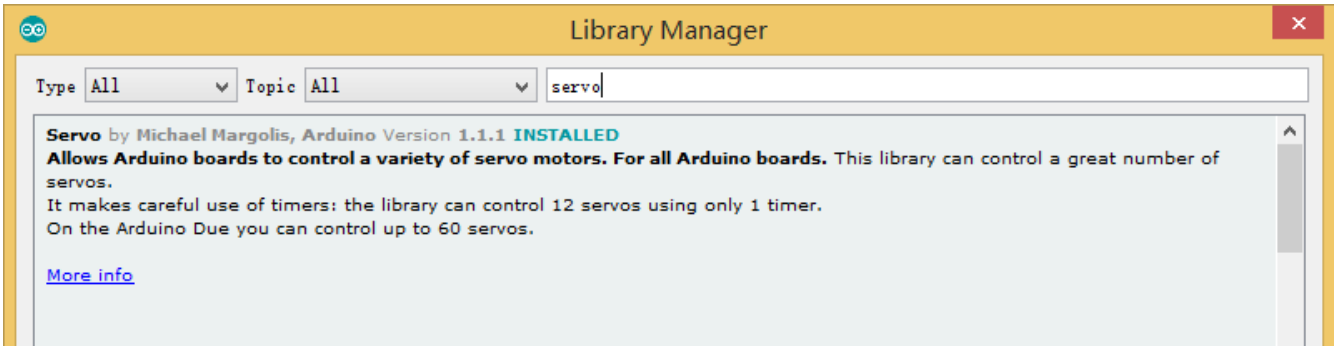
```
        myservo.write(180);
        delay(1000);
        leftDistance = Distance_test();

        #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();
}
```

Aprite il file Obstacle_Avoidance_Car\Obstacle_Avoidance_Car.ino



Dato che il programma usa la libreria <servo.h>, sara' necessario installarla.

Aprite Sketch---Include Library---Manage Libraries



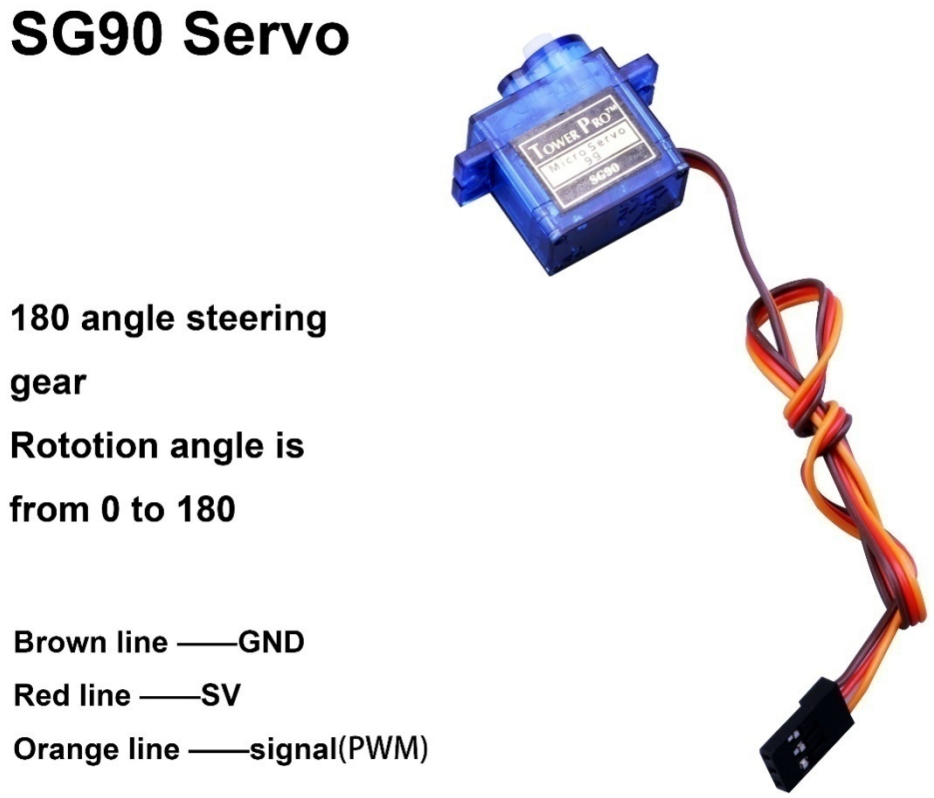
Cercate la libreria servo , quindi installate la versione piu' recente.

Dopo aver effettuato l'upload sulla scheda di controllo UNO , disconnettete il cavo, poggiate il veicolo a terra e accendetelo.

Vedrete che il veicolo procedera' e I sensori di distanza continueranno a ruotare, misurando la distanza continuamente. Se troveranno degli ostacoli, la piattaforma rotante si fermera' e il veicolo cambiera' direzione per non urtare l'ostacolo. Dopo aver evitato l'ostacolo, la piattaforma con I sensori riprendera' a ruotare e il veicolo riprendera' la sua marcia.

Ⅲ. Introduzione ai principi

Prima di tutto , impariamo a conoscere il Servomeccanismo SG90 :



Classificazione: 180 ingranaggio rotante

Normalmente il servomeccanismo ha 3 linee di controllo: alimentazione, massa e segnale.

Definizione dei pin del servomeccanismo: filo marrone——GND, filo rosso——5V, arancione——signal.

Come funziona il servomeccanismo:

Il chip di modulazione del segnale nel servomeccanismo riceve segnali dalla scheda controller , quindi il servomeccanismo acquisira' il voltaggio DC di base. C'e' anche un circuito di riferimento interno al servomeccanismo che produce un voltaggio standard. Questi due voltaggi vengono comparati l'uno con l'altro e la differenza sara' l'output. Quindi il chip del motore ricevera' questa differenza e stabilira' la velocita' di rotazione, la direzione e l'angolo. Quando non esiste differenza tra I due voltaggi, il servomeccanismo si fermera'.

Come controllare il Servomeccanismo:

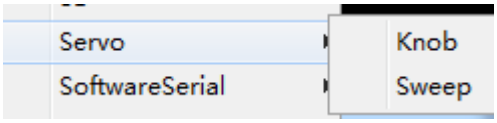
Per controllare la rotazione del servomeccanismo, e' necessario creare un impulso di circa 20ms e l'ampiezza dell'impulso di alto livello circa di 0.5ms~2.5ms, che e' coerente con l'angolo limitato del servomeccanismo.

Prendiamo un angolo del servomeccanismo di 180 come esempio , la corrispondenza sara':

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

Il programma:

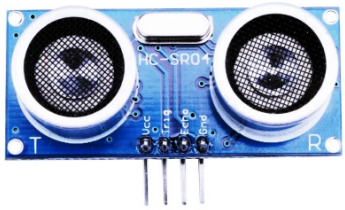
Arduino ha il file di libreria.<Servo.h>

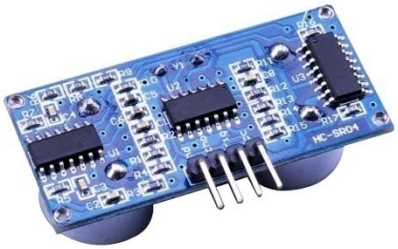



```
Servo myservo;  // crea l'oggetto servo per controllare il servomeccanismo
 myservo.attach(3);  // collega il servomeccanismo sul pin 3 all'oggetto servo
 myservo.write(90); //imposta la posizione del servomeccanismo in accordo con il valore scalare
```

E' possibile comandare l'ingrannaggio sterzante con 9 parole.

Ora, diamo uno sguardo la modulo del sensore ad ultrasuoni.





Caratteristiche del modulo: testare la distanza, modulo ad alta precisione.

Applicazione del prodotto: evitamento degli ostacoli con l robot、testare la distanza

degli oggetti、testare l liquidi、pubblica sicurezza、testare l posti di parcheggio.

Principali parametri tecnici

(1)：voltaggio usato: DC---5V

(2)：corrente statica: meno di 2mA

(3)：livello di output: piu' alto di 5V

(4)：livello di output: piu' basso di 0

(5)：angolo riconosciuto: non piu' grande di 15 gradi

(6)：distanza riconosciuta: 2cm-450cm

(7)：alta precisione: fino a 0.2cm

Metodo di connessione dei cavi: VCC, trig (la fine del controllo), echo (la fine della ricezione), GND

Come funziona il modulo:

(1)Per far partire la misurazione va applicato un segnale di livello alto alla porta di IO del TRIG , almeno una volta, per minimo 10us ;

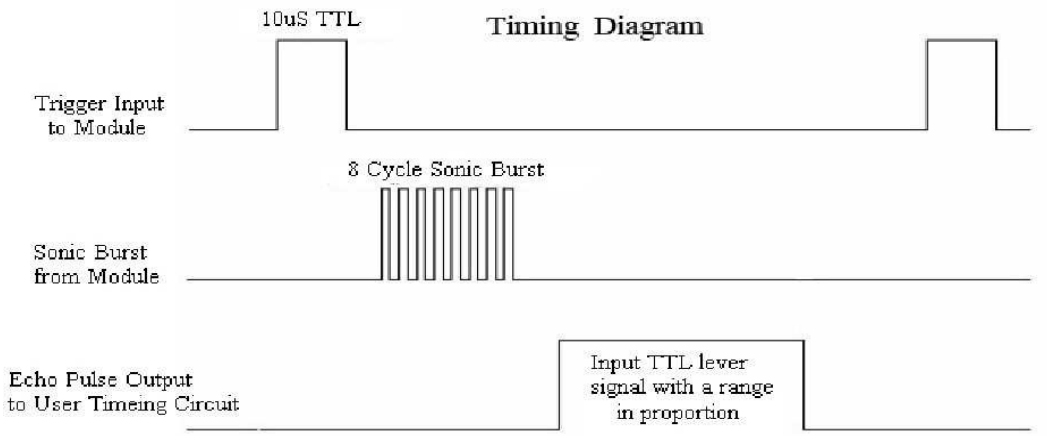
(2)Il modulo invia 8 onde quadre a 40kz automaticamente, e testa automaticamente se ci sono dei segnali di ritorno;

(3)se vengono ricevuti dei segnali, il modulo emettera' un impulso di livello alto attraverso la porta di IO dell'ECHO, la durata in tempo dell'impulso alto e' il tempo tra l'invio e la ricezione dell'onda. In questo modo il modulo puo' conoscere la distanza in base al tempo.

Calcolo della distanza= (tempo di livello alto* velocita' del suono (340M/S))/2);

Operazioni nell'istante:

Il diagramma dei tempi e' quello mostrato in basso. E' necessario solo fornire un impulso breve di 10uS sull'input per far partire la misurazione, il modulo emettera' un ciclo di otto scariche di ultrasuoni a 40 kHz e captera' l'eco. L'eco e' la distanza dell'oggetto che corrisponde all'ampiezza dell'impulso che e' la distanza in proporzione .Potete calcolare la distanza attraverso l'intervallo temporale tra l'invio del segnale di partenza e la ricezione dell'eco.La formula: uS / 58 = centimetri o uS / 148 =inch; o: la distanza = tempo di livello alto * velocita' (340M/S) / 2; suggeriamo di usare cicli di misurazione maggiori di 60ms , in modo da evitare di sovrapporre il segnale di partenza con l'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;
}
```

Dalla figura sopra, possiamo vedere che il principio su cui si basa l'evitamento degli ostacoli della macchina e' molto semplice. Il modulo sensore ad ultrasuoni rilevera' la distanza tra la macchina e l'ostacolo continuamente, inviando I dati alla scheda controller,quindi la macchina si fermera' e ruotera' il servomeccanismo rilevare le distanze sul lato destro e su quello sinistro. Dopo aver comparato le distanze dei diversi lati, la macchina girera' verso il lato che ha una distanza maggiore e iniziera' ad avanzare. Quindi il modulo del sensore ad ultrasuoni misurera' la distanza nuovamente.

```
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();
}
```