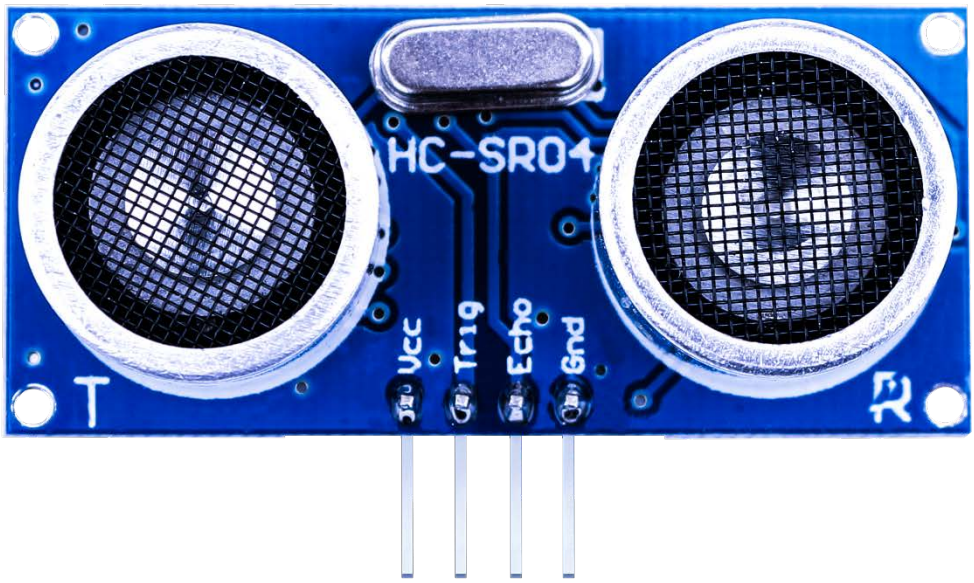
Leçon 4 – Evitement d'obstacles



Points Points clés de la leçon

La joie d'apprendre ne se limite pas seulement savoir contrôler son robot, mais également à savoir protéger votre robot. Donc apprenez à votre robot à se déplacer de manière autonome en évitant les obstacles.

Sommaire:

- Assembler le module ultrasons
- Se familiariser avec l'utilisation de la direction
- Les principes de l'évitement d'obstacles
- Utiliser le programme d'évitement d'obstacle

Matériel:

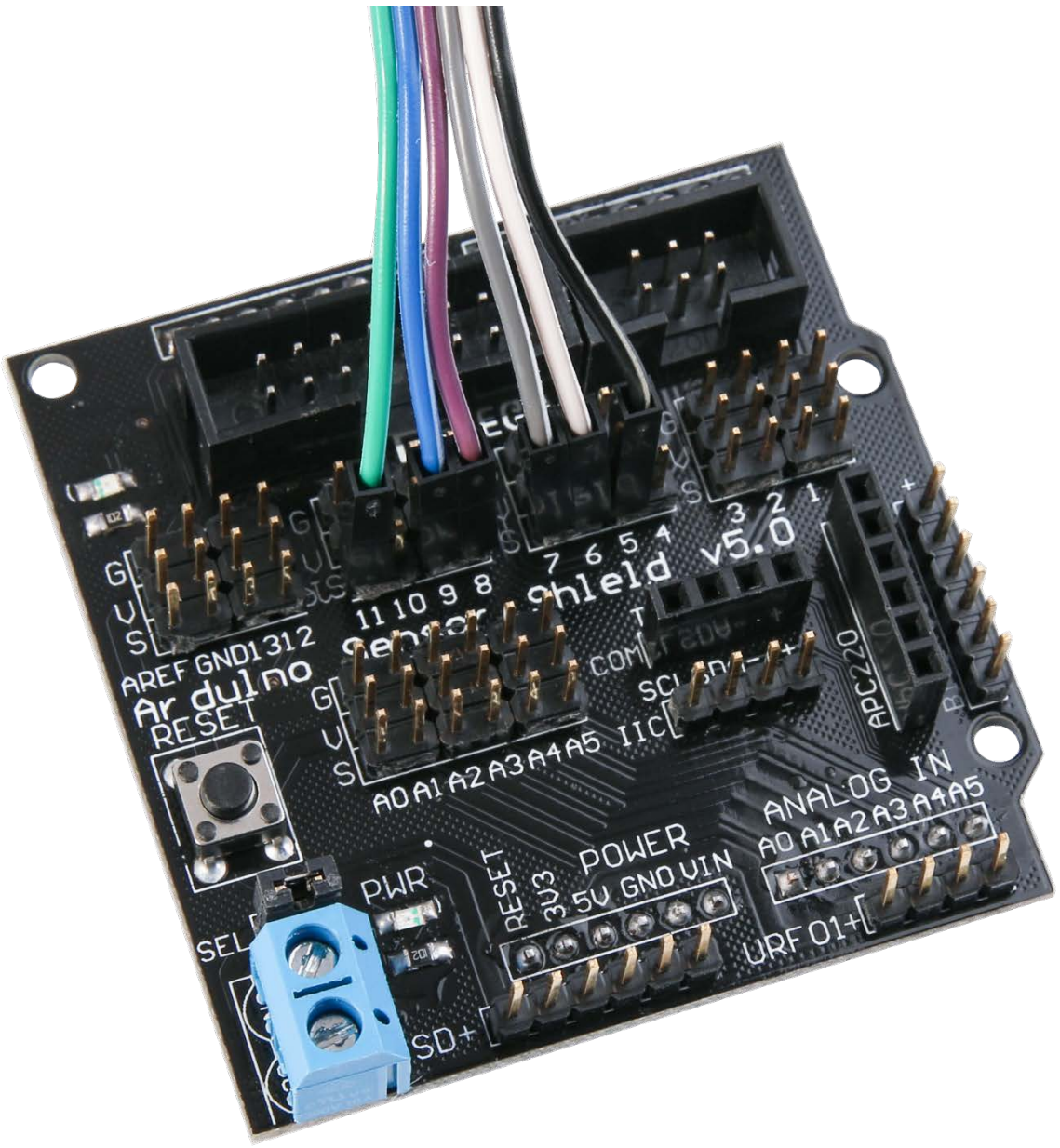
- Le robot
- Un cable USB
- Tête ultrason

Ⅰ. Connexions

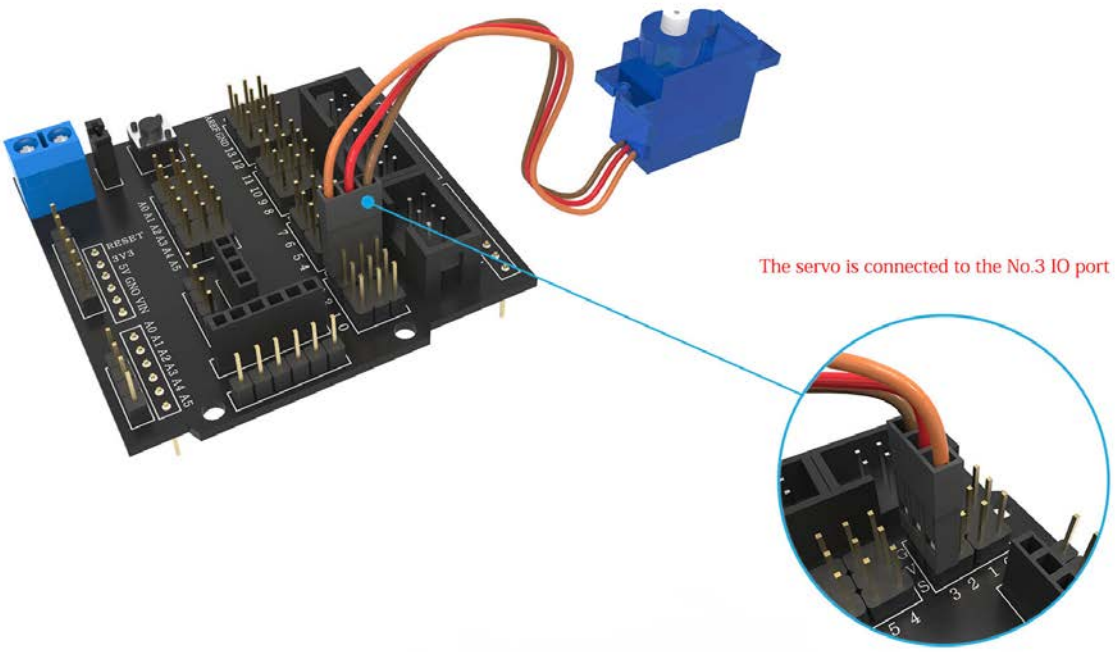
L298N

La bibliothèque du servomoteur permet de brancher jusqu'à 12 moteurs sur la plupart des cartes Arduino et jusqu'à 48 sur la carte Arduino Mega

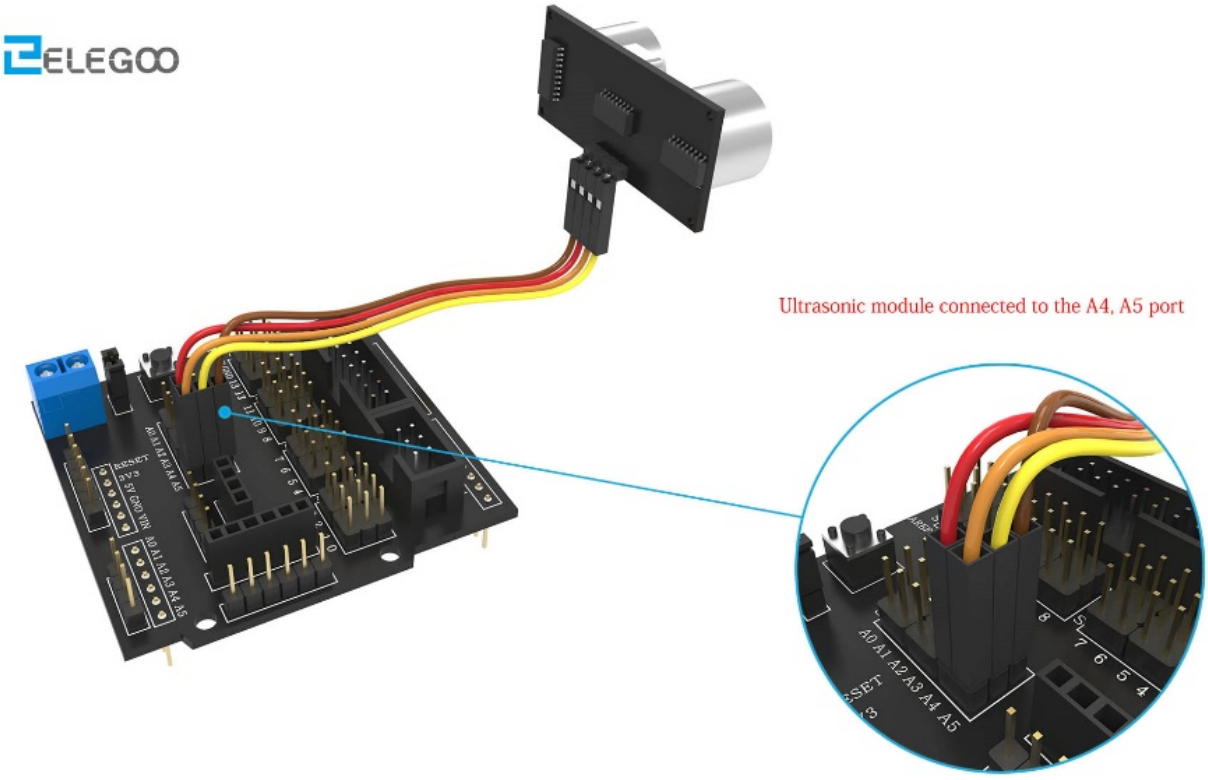
Sur les cartes différentes de l'Arduino Mega, l'utilisation de la bibliothèque désactive les fonctions "analogwrite" (PWM) sur les broches 9 et 10 même s'il y a des servomoteurs sur ces broches. Sur l'Arduino Mega, jusqu'à 12 servomoteurs peuvent être utilisés sans interférer avec les fonctions PWM. Utiliser entre 12 et 23 moteurs désactivera les broches 11 et 12. Vous aurez donc à changer les affectations des broches 10 et 11.

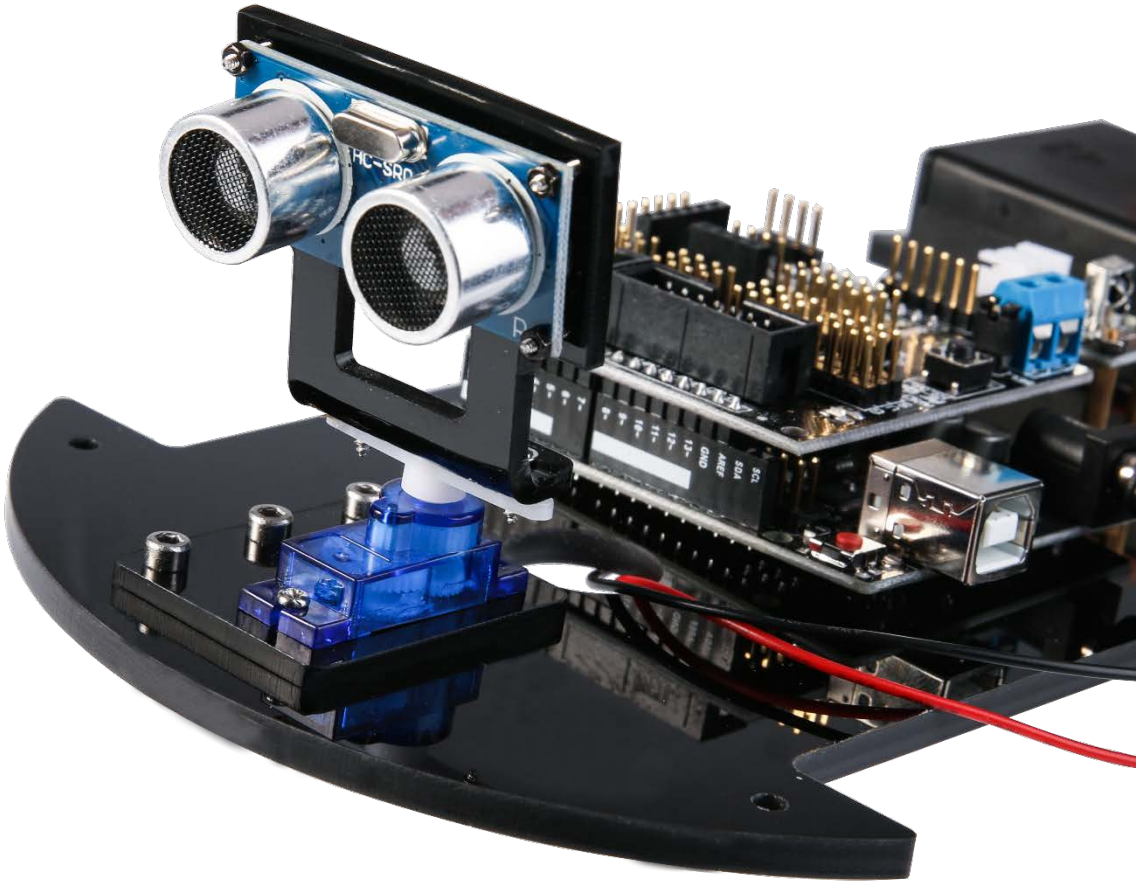

AREF GND 13 12
RESET
PWR
SEL
SD+
RESET
3V3
5V GND VIN
POWER
A0 A1 A2 A3 A4 A5
ANALOG IN
A0 A1 A2 A3 A4 A5
IIC
COM
APC220
URF 01+
v5.0

servo



Module Ultrason





Ⅱ. Transférer le code

```
#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);
```

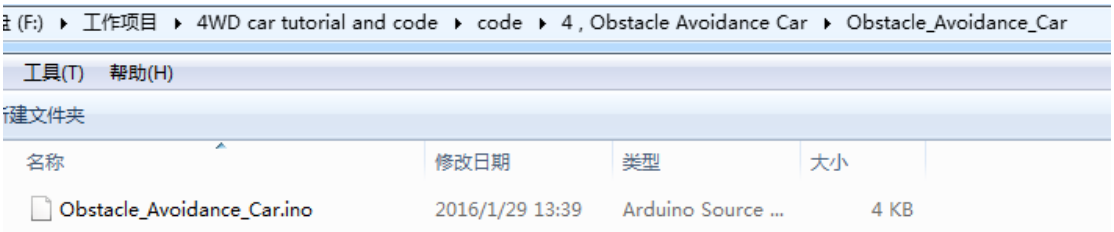
```
  digitalWrite(in2,LOW);
  digitalWrite(in3,HIGH);
  digitalWrite(in4,LOW);
 Serial.println("go left!");
}

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))
```

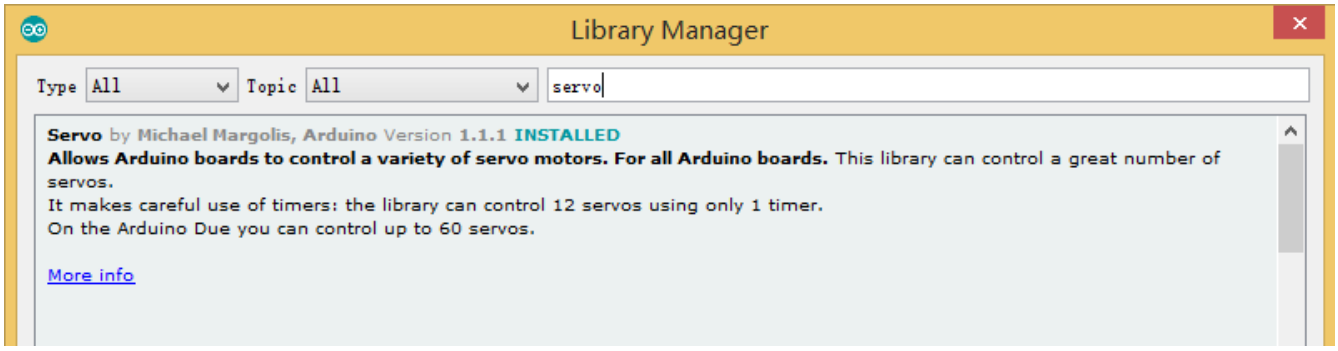
```
        {
          _mBack();
          delay(180);
        }
        else
        {
          _mForward();
        }
    }
    else
          _mForward();
}
```

Ouvrez le fichier “Obstacle_Avoidance_Car\Obstacle_Avoidance_Car.ino”



Veillez à installer en premier la bibliothèque “servo.h” qui permet de piloter le servomoteur en premier.

Allez dans le menu Sketch---Include Library---Manage Libraries



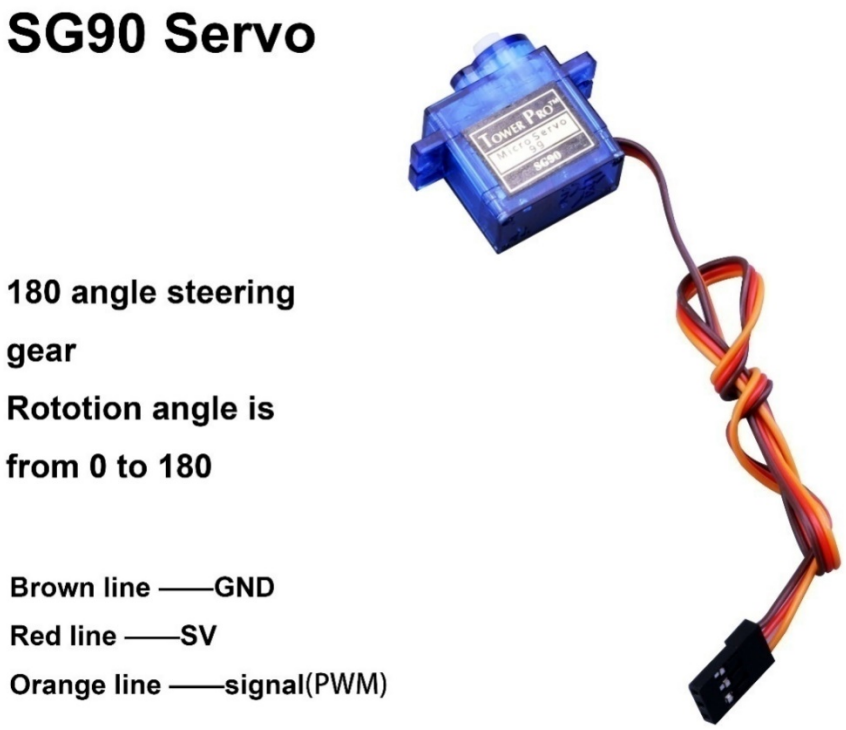
Cherchez “servo” et installez la bibliothèque

Après avoir transféré le code sur la carte UNO, déconnectez le câble USB, posez le robot au sol et mettez-le en marche.

Vous constaterez que le robot se met en mouvement. Le module ultrasons mesure en continue la distance qui sépare le robot d'un éventuel obstacle placé devant.

En cas de détection, le robot stoppe sa progression, la tête tourne à droite, à gauche, puis le robot tourne afin de se mettre dans une direction où il n'y a plus d'obstacle.

Ⅲ. Principes de fonctionnement



Caractéristiques : angle de rotation de 180 °

Le servomoteur est connecté à la carte via trois fils:

Fil Brun : la masse

Fil Rouge : le 5v

Fil Orange : le signal

Normally the servo has 3 controlling line: power supply, ground and sign.

Comment fonctionne le servomoteur::

La puce de modulation du signal à l'intérieur du servo reçoit le signal de la carte controlleur ainsi qu'une tension continue. Il y a également un circuit de reference à l'intérieur du servo qui produit une tension standard. Ces deux tensions sont comparées et la difference est envoyée en sortie. La puce du moteur reçoit alors cette difference et définit la vitesse de rotation, la direction et l'angle. Lorsque les tensions sont identiques, le servo s'arrete.

Comment controller le servo:

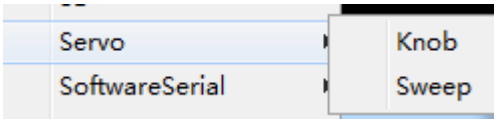
Pour controller la rotation du servo, il faut envoyer une impulsion d'une durée de 20 ms Avec une impulsion de niveau haut comprise entre 0.5 ms et 2.5 ms qui correspondra aux valeurs angulaires limites du servomoteur.

Prenons l'exemple d'un servomoteur de 180 °, voici les correspondances ci-dessous entre durées et angles :

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

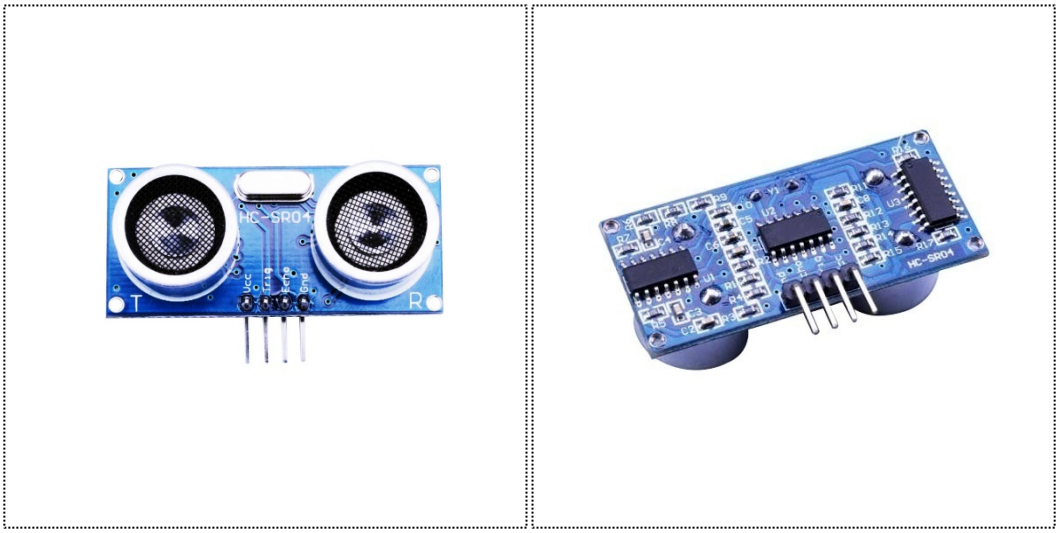
Le code:

Une bibliothèque est intégrée dans l'Arduino , c'est le fichier "Servo.h"




```
Servo myservo;  // create servo object to control servo
 myservo.attach(3);  // attach servo on pin 3 to servo object
 myservo.write(90); //set servo position according to scaled value
```

Le module ultrason.



Caractéristiques du module : module de haute précision permettant de mesurer des distances.

Applications: robot d'évitement d'obstacles, testeur de distance, testeur de liquides, sécurité, stationnement.

Paramètres techniques :

(1)：voltage: DC---5V
(2)：courant: < 2mA
(3)：sortie: >5V
(4)：sortie: < 0V
(5)：détection angle: 15°
(6)：détection de distance: 2 à 450cm
(7)：précision: 0.2cm

Fonctionnement d'un module:

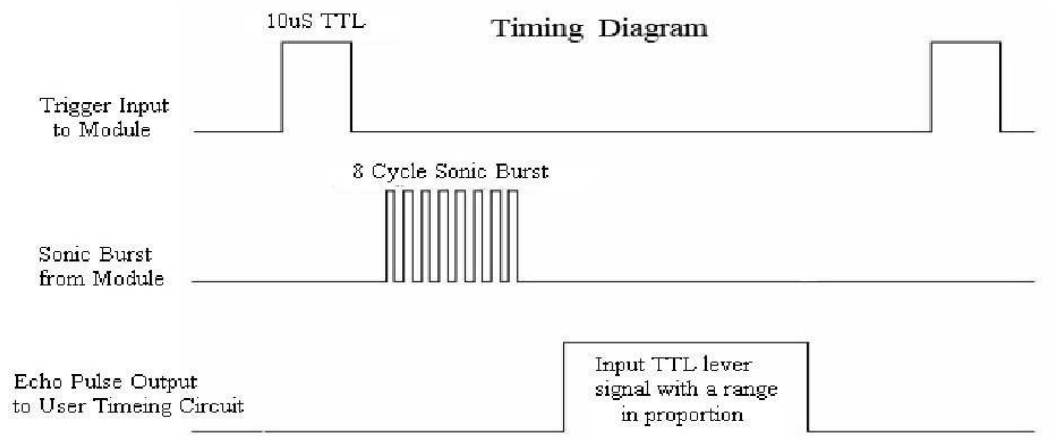
(1)Appliquer un signal de sortie sur TRIG pendant au moins 10μs une fois.
(2)Le module émet 8 vagues de 40kHz.
(3)Si un signal est reçu en retour, le module va émettre une impulsion sur ECHO.

La durée de l'impulsion correspond au temps entre émission et réception du signal ultrasons. Il est donc possible (connaissant la vitesse de son) d'en déduire la distance parcourue.

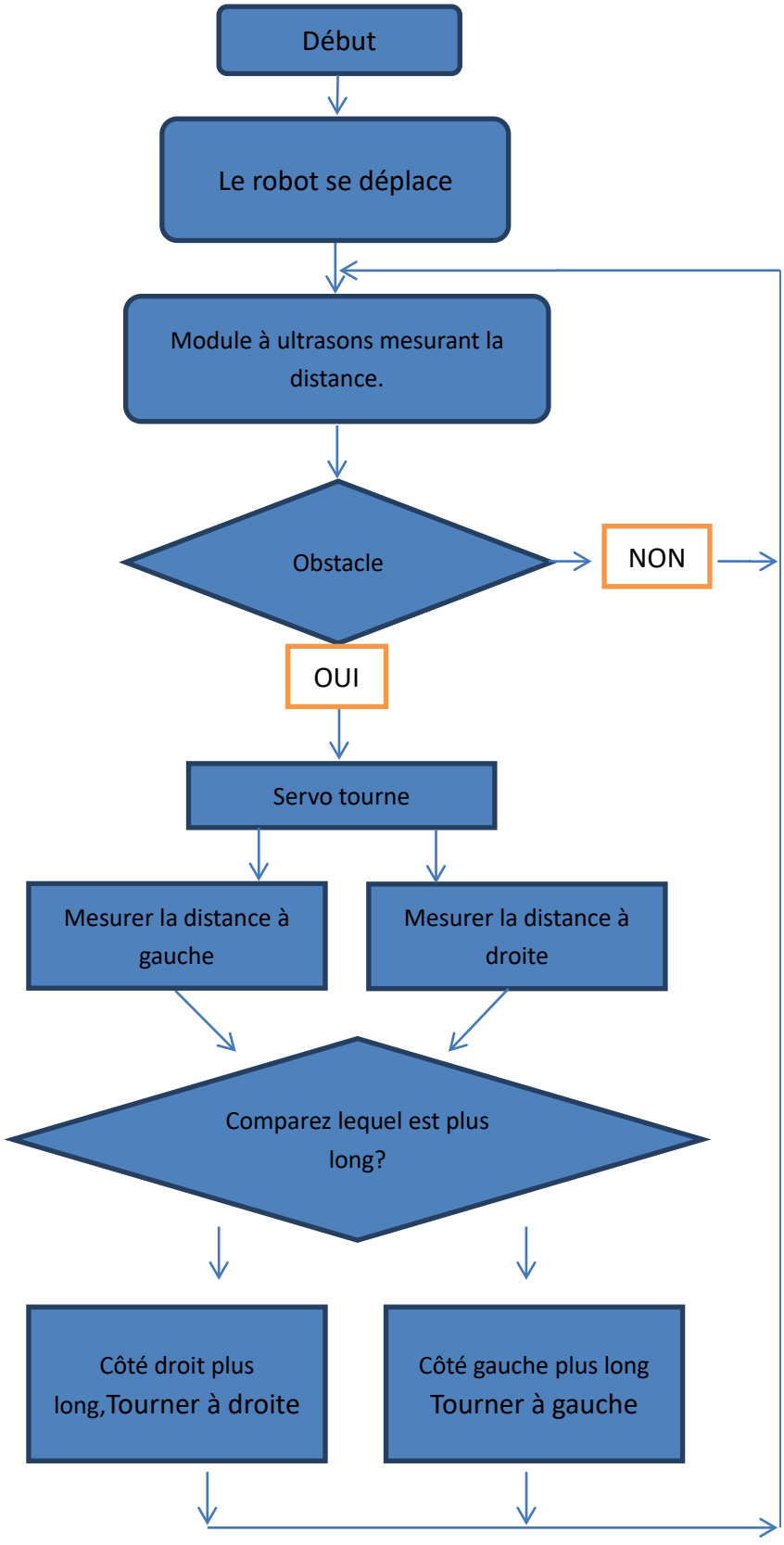
Distance parcourue = (temps d'impulsion * vitesse du son (340 m/s))/2)

Diagramme de fonctionnement:

Le diagramme du temps est le suivant. Il suffit de fournir une courte impulsion de 10µs au Trigger pour démarrer le cycle. Le module va alors envoyer 8 cycle d'ultrasons et mesurera leurs echos. L'ECHO est la distance à l'objet qui est proportionnelle à la largeur d'impulsion et à l'echelle. Vous pouvez calculer l'echelle avec l'inteervalle de emps entre le signal envoyé au trigger et le signal reçu en echo. Nous vous conseillons d'utiliser des cycles de mesures supérieurs à 60 µs pour qu'il n'y ait pas de confusion entre les signaux envoyé au trigger et la réponse de l'echo.




```
/*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;
}
```


Début
Le robot se déplace
Module à ultrasons mesurant la distance.
Obstacle
NON
OUI
Servo tourne
Mesurer la distance à gauche
Mesurer la distance à droite
Comparez lequel est plus long?
Côté droit plus long,Tourner à droite
Côté gauche plus long Tourner à gauche

En conclusion, nous voyons que le principe d'évitement d'obstacle est assez simple.
Le module ultrasons détecte en permanence la distance entre le robot et un éventuel obstacle, envoyant les données à la carte Elegoo UNO.
Lorsqu'un objet est détecté, le robot stoppe sont déplacement.
Le servomoteur oriente le module ultrason tour à tour à droite et à gauche, une mesure étant déclenchée à chaque fois.
Après avoir comparé les distances mesurées, le code décide si le robot doit tourner à droite, à gauche ou reculer et le robot reprend son parcours et ainsi de suite.

```
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();
}
```