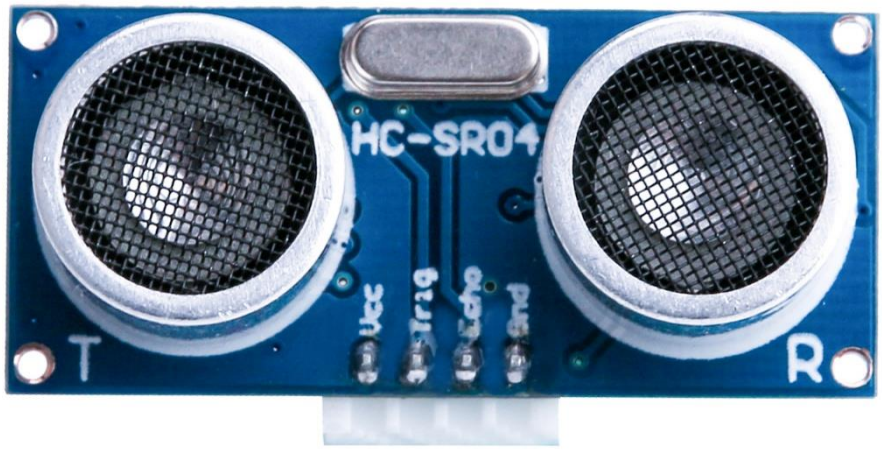
Leçon 4 – Evitement d'obstacles

(Obstacle avoidance car)



Points clés de la leçon

La joie d'apprendre ne se limite pas seulement savoir contrôler son robot.
Apprenez à permettre à votre robot de 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'obstacle
- Utiliser le programme d'évitement d'obstacle

Préparatifs:

- Le robot
- Un câble USB (fourni)

I . Connexions

Servomoteur

Un servomoteur est un moteur dont il est possible de définir de manière précise la position angulaire de la tête. Les applications sont nombreuses, notamment lorsqu'il est nécessaire d'orienter un module selon un angle donné.

Ⅱ. Déverser le code sur la carte Elegoo UNO

```
#include <Servo.h> //servo library
Servo myservo; // create servo object to control 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!");
}
```

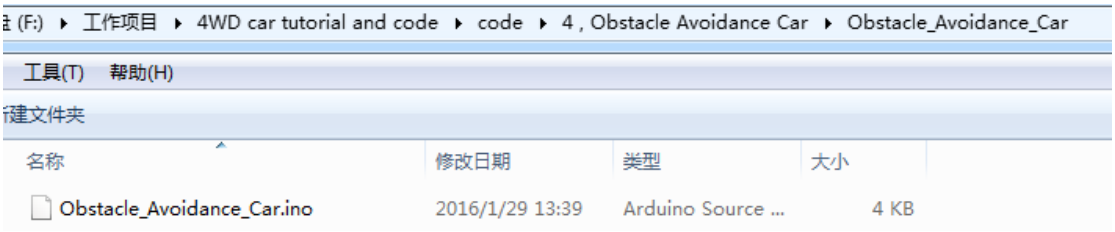
```
void _mleft()
{
  analogWrite(ENA,ABS);
  analogWrite(ENB,ABS);
    digitalWrite(in1,HIGH);
    digitalWrite(in2,LOW);
    digitalWrite(in3,HIGH);
    digitalWrite(in4,LOW);
  Serial.println("go left!");
}

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);
        }
```

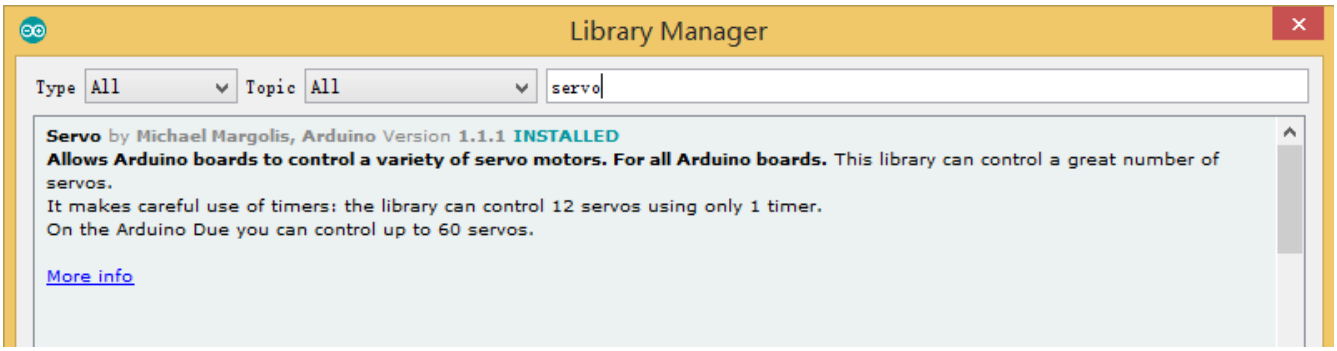
```
        else
        {
          _mForward();
        }
    }
    else
        _mForward();
}
```

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.

Vous pouvez vérifier que la bibliothèque est bien installée:



Allez dans le menu “Croquis/Inclure une bibliothèque/Gérer les bibliothèques...”

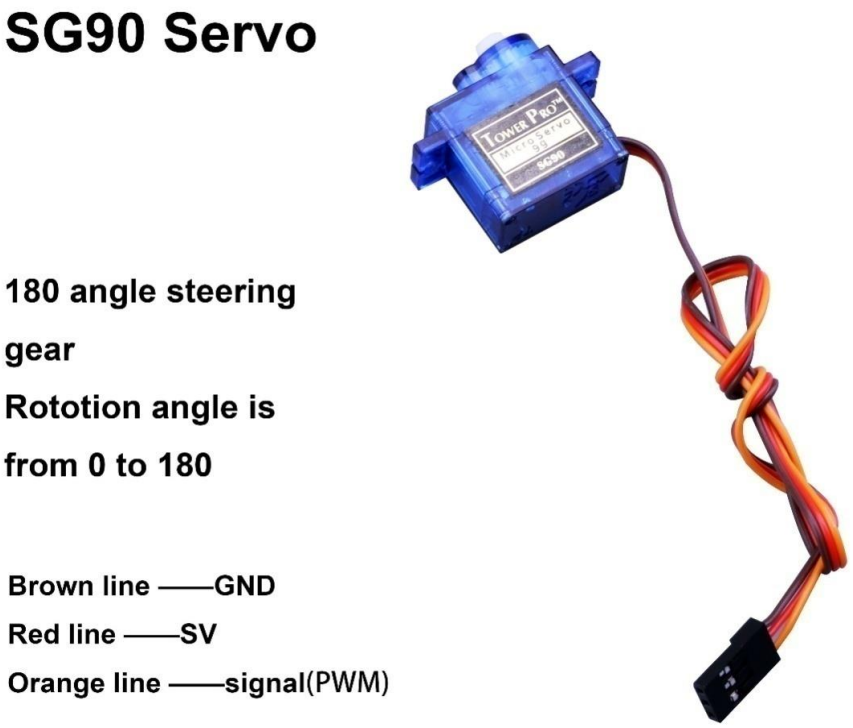
Après avoir déversé le code sur la carte Elegoo UNO, déconnectez le câble USB, posez le robot au sol et mettez-le en marche.

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

Le servomoteur SG90:



Le servomoteur est connecté à la carte via trois fils:

Fil Brun : la masse
Fil Rouge : le +
Fil Orange : le signal

Comment fonctionne un servomoteur:

Un signal de référence est généré dans le servomoteur. Il est comparé par une carte de contrôle interne au signal reçu en provenance de la carte Elegoo UNO. Cela génère un ordre de mouvement au moteur, jusqu'à ce que la différence entre les deux signaux soit nulle. Le moteur se place ainsi selon une valeur angulaire donnée.

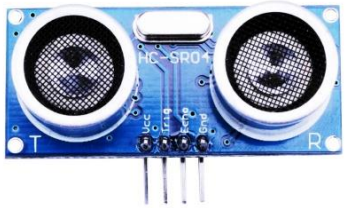
Comment contrôler le servomoteur:

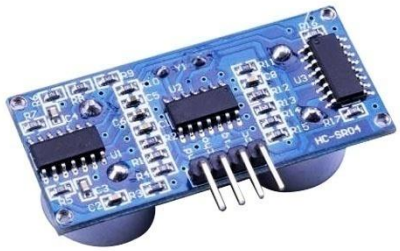
Une bibliothèque intégrée (servo.h) s'occupe de toute la gestion. Il suffit simplement de définir l'angle souhaité.

Voici le code présent dans les programmes qui utilisent un servomoteur :

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

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 du module

(1)Appliquer un signal de sortie sur TRIG pendant au moins 10μs une fois.

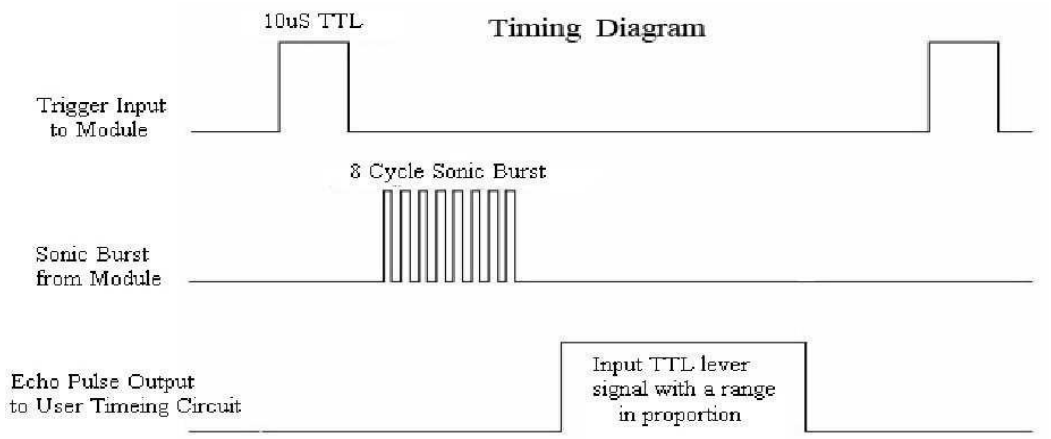
(2)Le module émet 8 vagues de 40kHz.

(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.

Comme la distance correspond à l'aller ET au retour de l'ultrason entre le module et l'objet détecté, il faut appliquer une division par deux pour avoir la distance module/objet détecté.

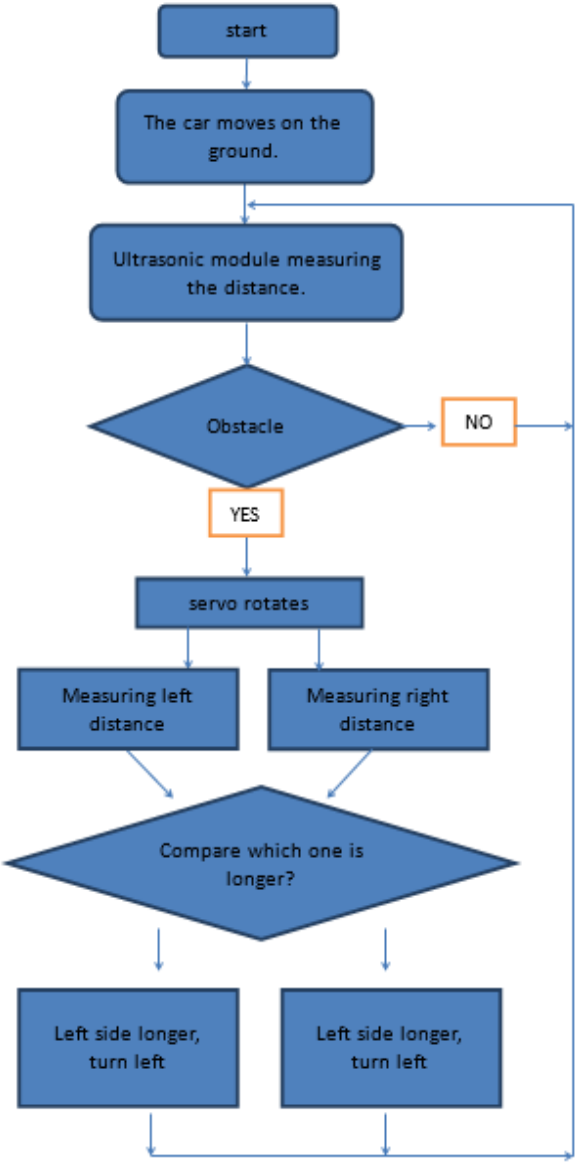
Testing distance= (high level time* velocity of sound (340M/S))/2);




```
/*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;
```

}

Diagramme de fonctionnement du code:



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();
}
```