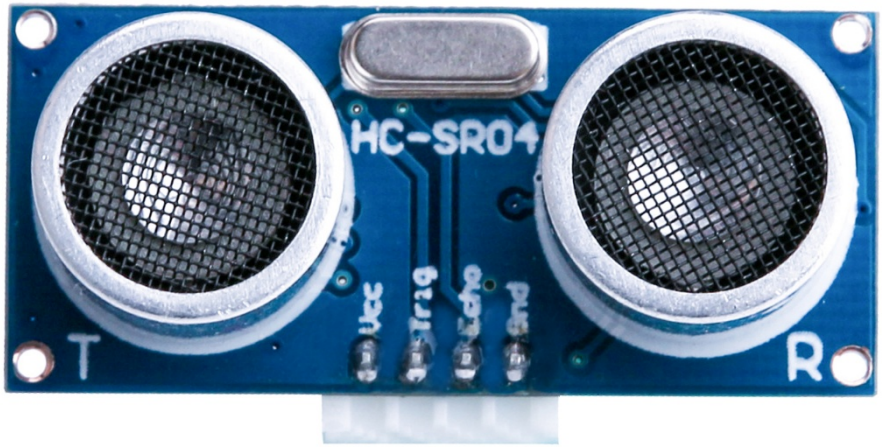
Lesson4 障害物回避車



このセクションのポイント

学習の喜びは、あなたの車を制御する方法だけでなく、あなたの車を守ることも可能です。 つまり、車は衝突を回避することができます。

学習パーツ:

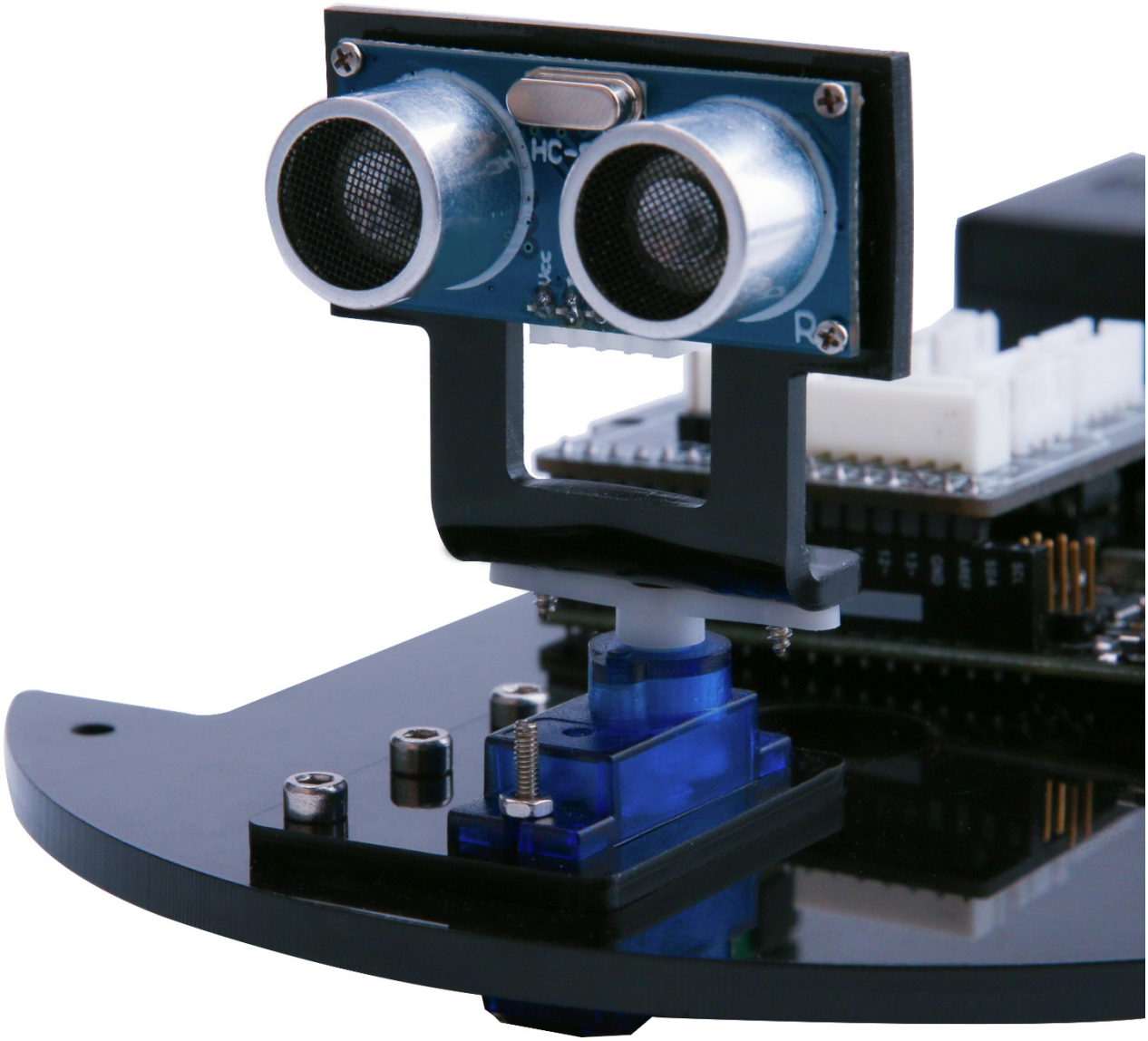
- 超音波モジュールの組み立て方を学ぶ
- ステアリングの使い方に慣れている
- 車の回避の原則について学ぶ
- 障害物回避車を実現させるプログラムを使用する

準備:

- 車（バッテリー付き）
- USB ケーブル
- 超音波クレードルヘッドのスーツ

Ⅰ. 接続

サーボ



Ⅱ．プログラムをアップロードする

```
#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);
```

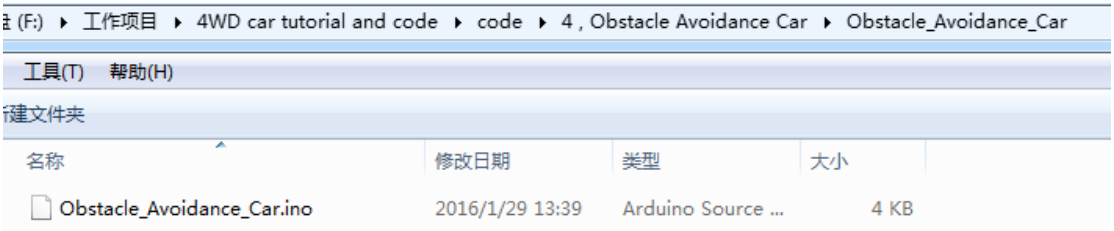
```
  digitalWrite(in1,HIGH);
  digitalWrite(in2,LOW);
  digitalWrite(in3,HIGH);
  digitalWrite(in4,LOW);
 Serial.println("go left!");
}

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);
         }
```

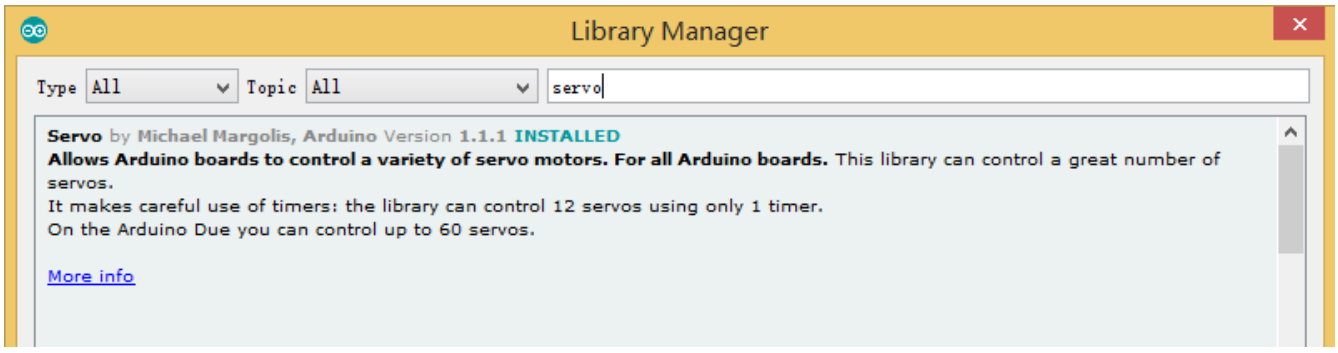
```
        else if((rightDistance<=20)||(leftDistance<=20))
        {
          _mBack();
          delay(180);
        }
        else
        {
          _mForward();
        }
    }
    else
        _mForward();
}
```

ファイル Obstacle_Avoidance_Car ￥ Obstacle_Avoidance_Car.ino を開きます。



プログラムはライブラリ<servo.h>を使用するので、最初にライブラリをインストールする必要があります。

Open the Sketch---Include Library---Manage Libraries



サーボを検索し、最新のバージョンをインストールします。

プログラムを UNO コントロールボードにアップロードした後、ケーブルを外し、車両を地上に置き、電源をオンにします。

車両が前方に移動し、クラウドプラットフォームが回転し続け、距離測定センサーが連続的に動作することがわかります。 先に障害物があると、クラウドプラットフォームが停止し、障害物を回避する方向に車

両が進行方向を変えます。 障害物を回避した後、クラウドプラットフォームは再び回転を続け、車両も移動します。

Ⅲ. 原理の導入

まず、SG90 サーボについて学びましょう:

分類：180 ステアリングギア
通常、サーボには 3 つの制御線があります：電源、グランド、および符号。
サーボピンの定義：ブラウンライン - GND、赤ライン - 5V、オレンジ - 信号。
サーボの仕組み:
サーボの信号変調チップはコントローラボードから信号を受信し、サーボは基準 DC 電圧を得ます。 また、サーボ内部には基準電圧を生成する基準回路があります。 これらの 2 つの電圧

は互いに比較され、その差が出力されます。 その後、モーターチップは差を受け取り、回転速度、方向、および角度を決定する。 2 つの電圧に差がない場合、サーボは停止します。

サーボの制御方法：

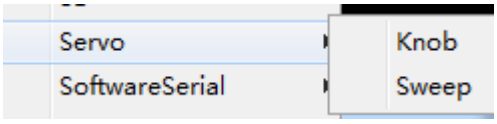
サーボの回転を制御するには、時間パルスを約 20ms、高レベルパルス幅を約 0.5ms?2.5ms とする必要があります。これはサーボの角度制限と一致しています。

例えば 180° の角度サーボを取ると、対応する制御関係は以下のようになる:

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

The program:

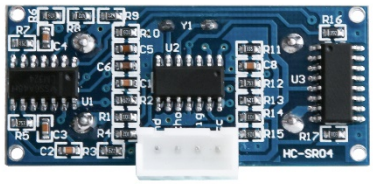
Arduino にはライブラリファイルがあります<Servo.h>




```
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
```

ステアリングギアは 9 ワードで駆動できます。

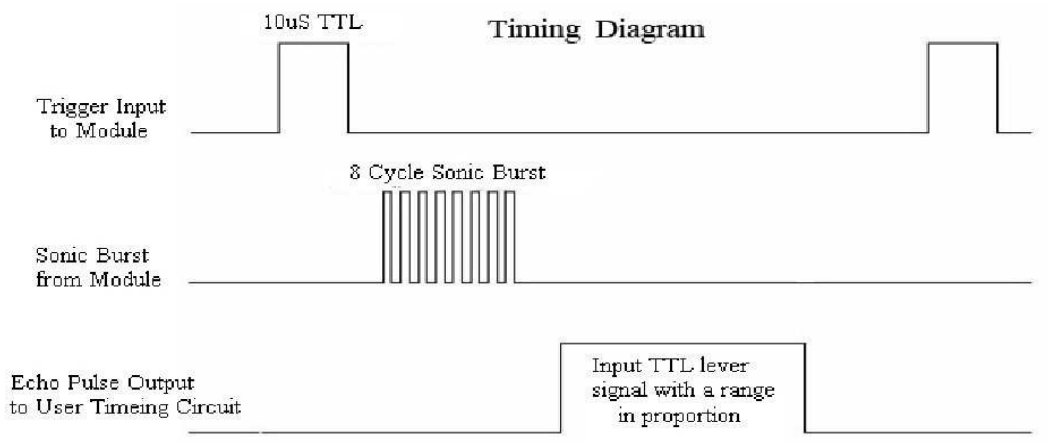
次に、超音波センサーモジュールを見てみましょう。



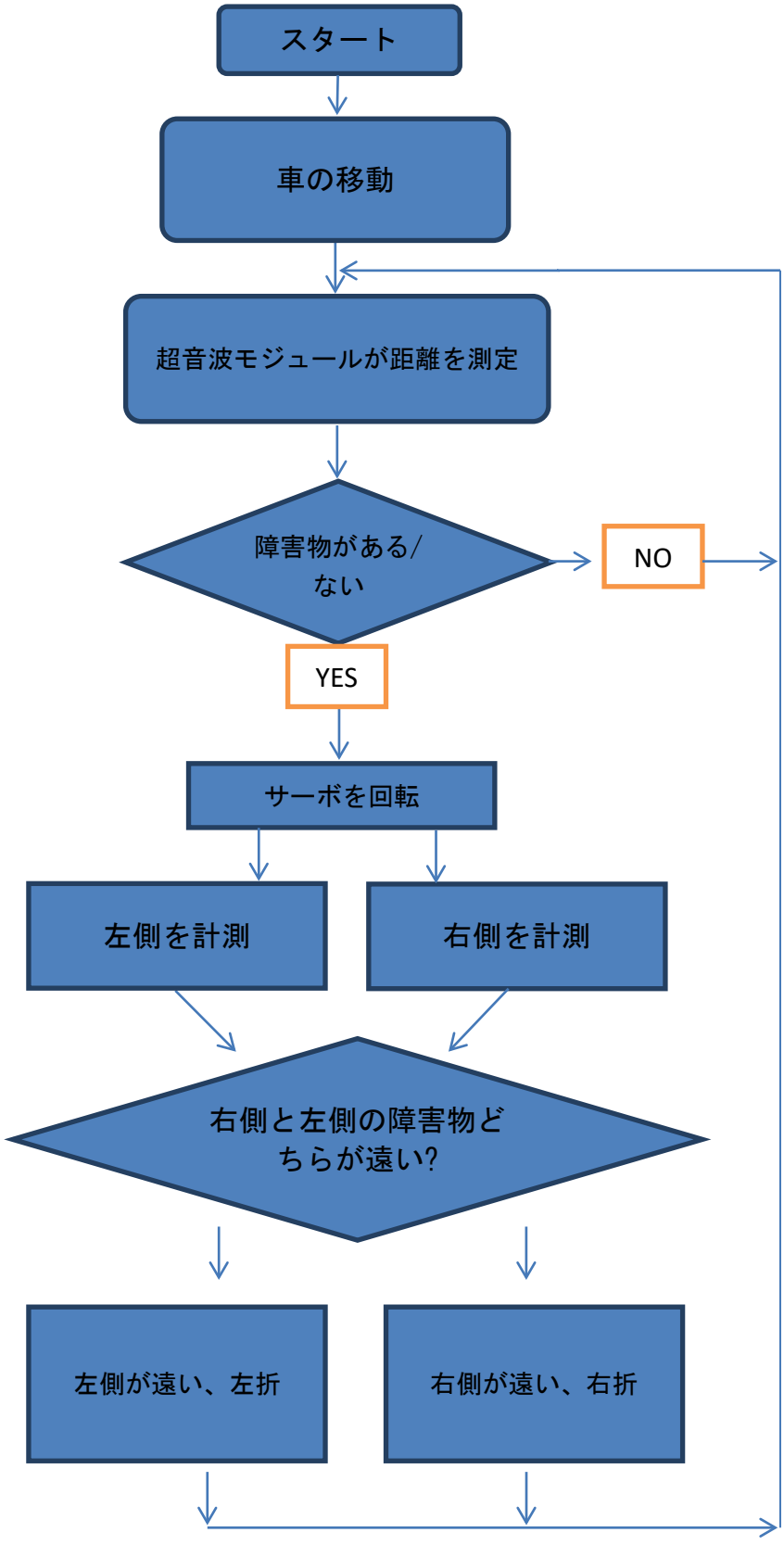
モジュールの特徴：テスト距離、高精度モジュール。
製品のアプリケーション：ロボットの障害物の回避、オブジェクトのテスト距離、液体テスト、公の安全、駐車場のテスト。
主な技術パラメータ
（1）：使用電圧：DC --- 5V
（2）：静的電流：2mA 未満
（3）：レベル出力：5V 以上
（4）：レベル出力：0 より小さい
（5）：検出角：15 度以下
（6）：検出距離：2cm~450cm
（7）：高精度：最大 0.2 センチメートル
接続方法：VCC、trig（制御終了）、エコー（受信終了）、GND
モジュールの機能方式：
（1）TRIG の IO ポートをトリガー・レンジに適用し、少なくとも 10us の高レベル信号を与えます。
（2）モジュールは 40kz の 8 つの方形波を自動的に送信し、信号が自動的に返ってくるかどうかをテストします。
（3）信号が受信された場合、モジュールは ECHO の IO ポートからハイレベルパルスを出力します。ハイレベルパルスの持続時間は送受信間の時間です。したがって、モジュールは時間に応じて距離を知ることができます。

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

実際の操作：
以下にタイミング図を示します。 トリガー入力に short10uS パルスを供給して測距を開始するだけで、モジュールは 40kHz で 8 サイクルの超音波バーストを送信し、エコーを上げます。エコーは、パルス幅と距離範囲の距離オブジェクトです。トリガー信号を送信してからエコー信号を受信するまでの時間間隔で、範囲を計算できます。 数式：uS / 58 =センチメートルまたは uS / 148 =インチ; または：範囲=高レベル時間*速度（340M / S）/ 2; エコー信号へのトリガ信号を防止するために、60ms 以上の測定サイクルを使用することを推奨します。




```
/*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;
}
```


スタート
車の移動
超音波モジュールが距離を測定
障害物がある/
ない
NO
YES
サーボを回転
左側を計測
右側を計測
右側と左側の障害物どちらが遠い?
左側が遠い、左折
右側が遠い、右折

上の図から、障害回避車の原理は非常に簡単であることがわかります。 超音波センサモジュールは、車と障害物との距離を何度も検出してコントローラボードに送信し、サーボを停止して回転させて左側と右側を検出します。 別の側からの距離を比較した後、車は障害物が遠くにある側に回り、前方に移動します。 次に、超音波センサモジュールが再び距離を検出します。

```
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();
}
```