





障害回避車




Points of this section

学習の喜びは、あなたの車を制御する方法だけでなく、あなたの車を守る方法も知っています。
だから、あなたの車を衝突から遠ざけてください。

Learning parts:

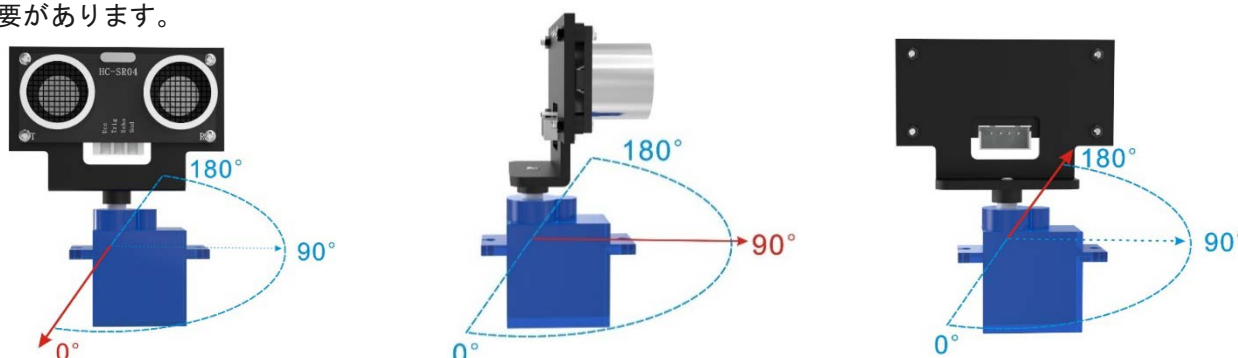
-  超音波モジュールの組み立て方を学ぶ
-  ステアリングの使い方に慣れている
-  車の回避の原則について学ぶ
-  障害物回避車を実現させるプログラムを使用する

Preparations:


-  *A car (with battery)*
-  *A USB cable*
-  超音波クレードルヘッドのスーツ

I . Connection

超音波センサーモジュールホルダーを組み立てるときは、サーボを 180 度回転できるようにサーボもデバッグする必要があります。



STEP1: UNO をコンピュータに接続し、“¥ Lesson 4 Obstacle Avoidance Car ¥ Servo_debug ¥ Servo_debug. ino” というパスに Servo_debug コードファイルを開きます。

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 }

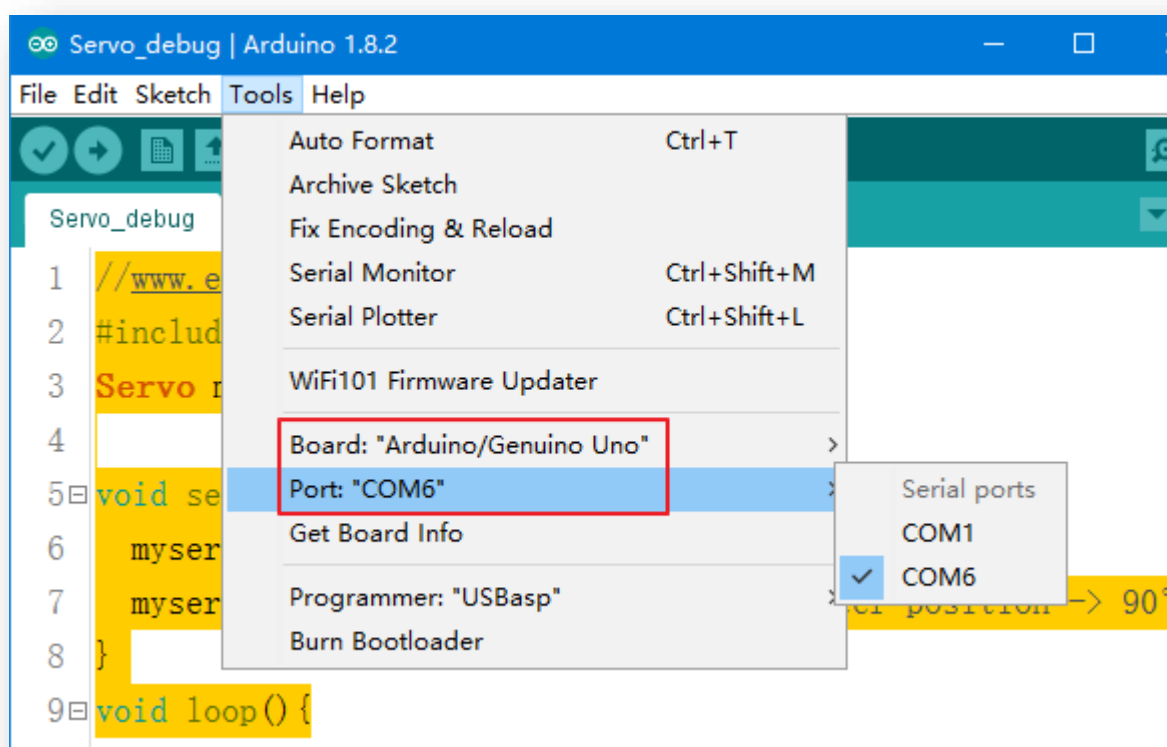
```

Servo_debug code preview:

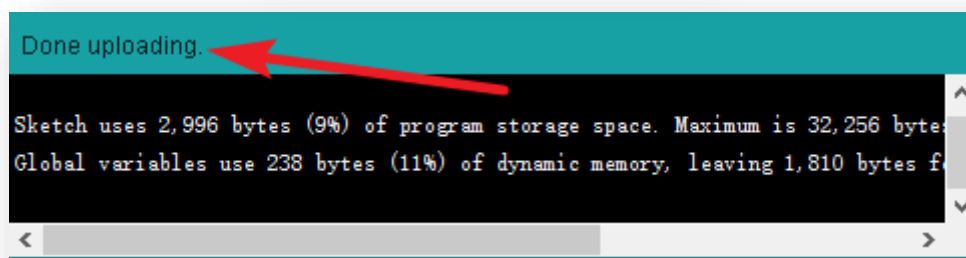
```
//www.elegoo.com
#include <Servo.h>
Servo myservo;

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

STEP2: Select "Tool" --> "Port" and "Board" in the Arduino IDE.



STEP3: 矢印ボタンをクリックしてコードを UNO コントローラボードにアップロードします

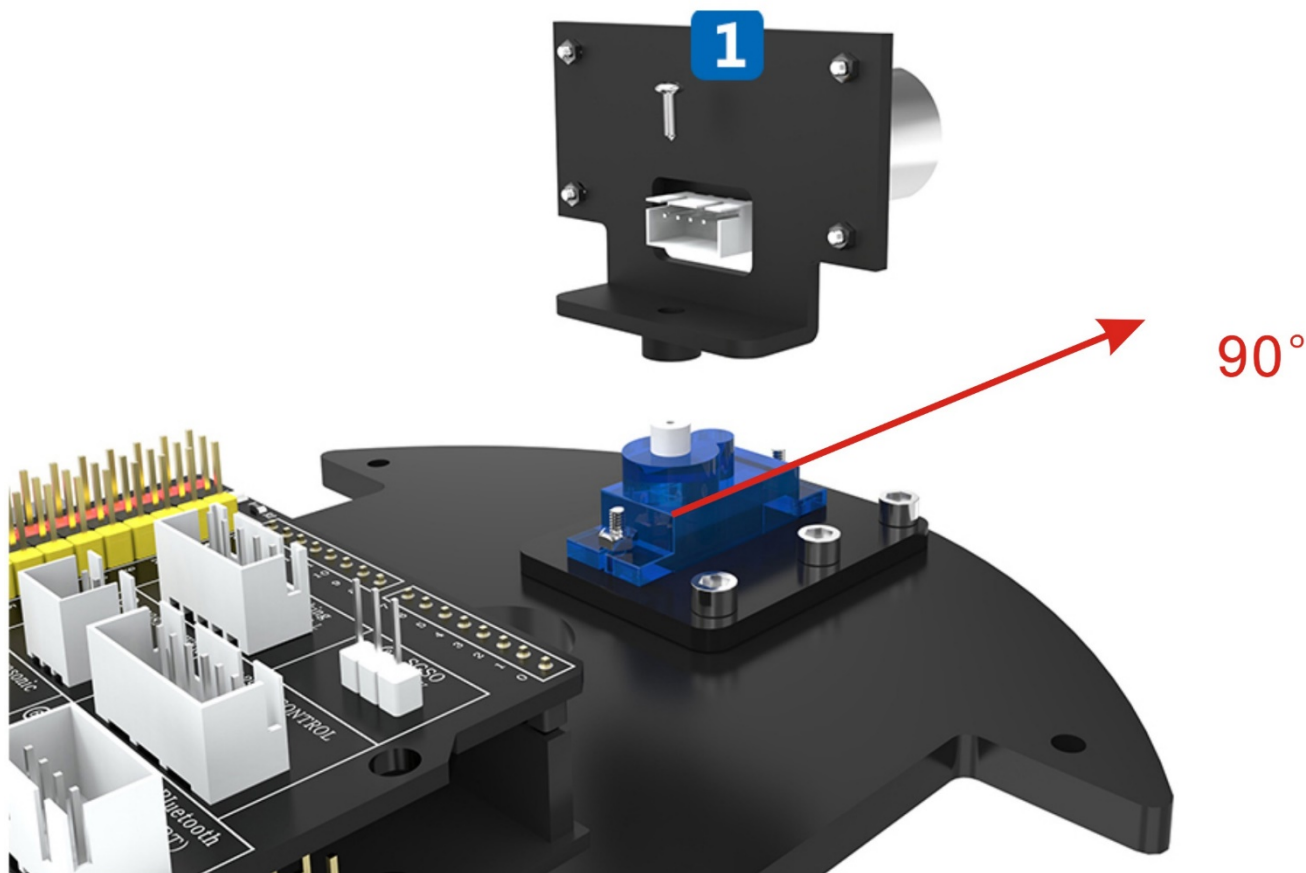


アップロードが完了すると、サーボは 90 度回転して停止します

STEP4: Assemble the ultrasonic sensor module at 90 degrees.

(超音波センサーモジュールを 90 度で組み立てます。)

マイクロサーボの各歯の角度は 15 度であり、90 度の方向の中央に取り付けると、それは左右に 15 度回転します。つまり、実際のマイクロサーボの取り付け具合は 15 度です 105 度



サーボモータに関するよくある質問

1 マイクロサーボが電源を入れるたびに反時計回りに 15 度回転するのはなぜですか？

これは SG90 マイクロサーボには正常であり、プログラムでの通常の使用には影響しません。

プログラムでコントロールしていない場合は、電源を入れる前に、手で通常の状態に戻したり、マイクロサーボに接続されているワイヤを外したりすることができます。

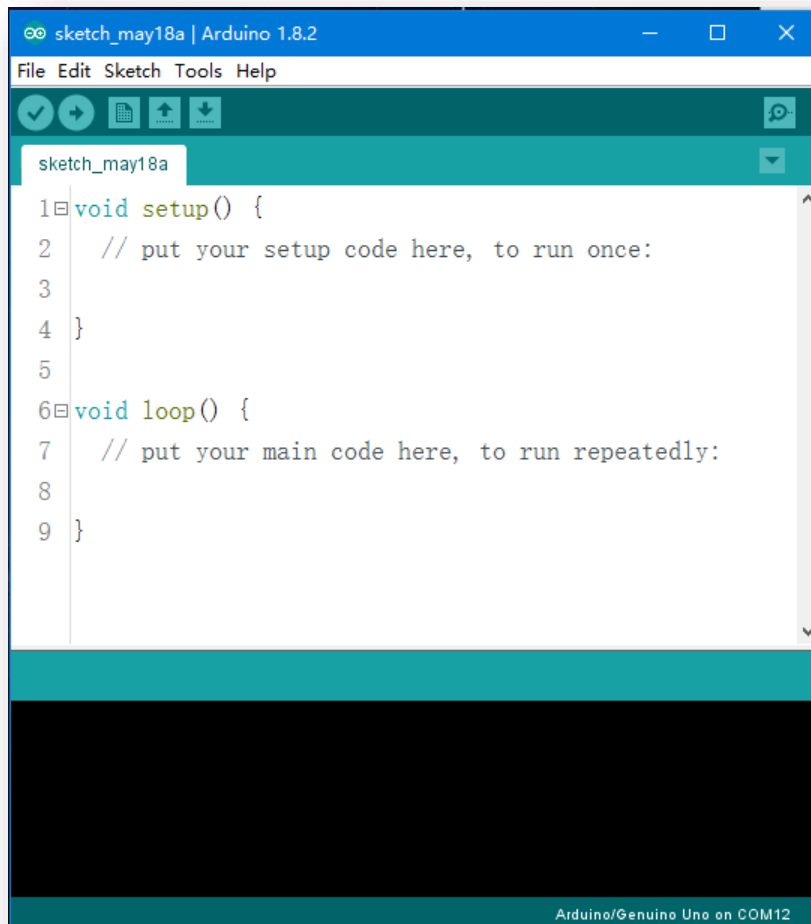
2 マイクロサーボは制御不能で回転し続けます。

0~180 の範囲の角度にマイクロサーボを指令するには、“myservo.write (angle)”を使用します。範囲を超えると、マイクロサーボはこの角度を認識せず、回転し続けます。

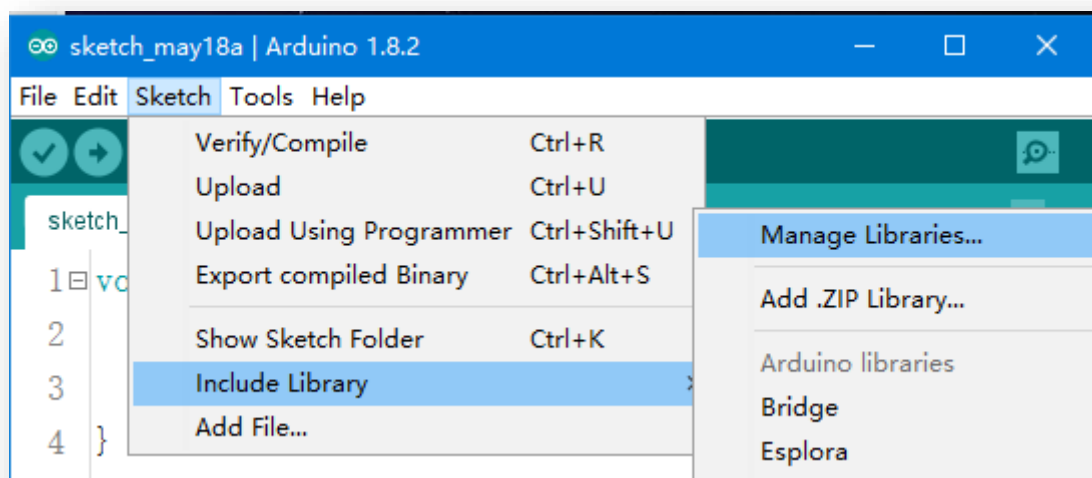
II. Upload program

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。

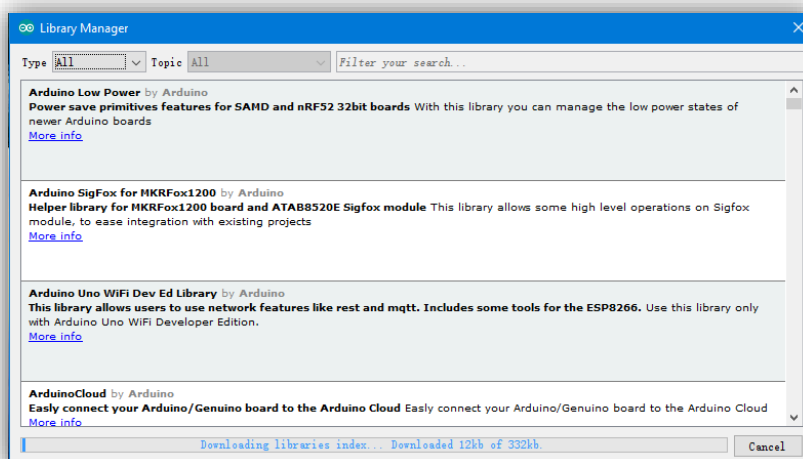
Arduino ソフトウェアを開く



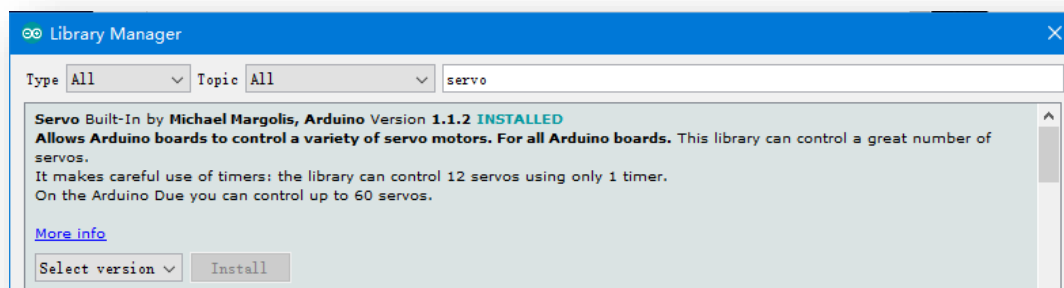
Select Sketch -> Include Library -> Manage Libraries...



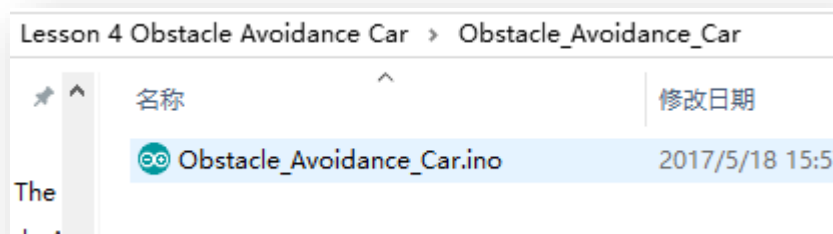
「ライブラリのインデックスをダウンロード中」が完了するのを待つ。



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。 次の図は、Servo ライブラリがすでにインストールされていることを示しています。



UNO コントローラボードをコンピュータに接続し、パス “¥ Lesson 4 Obstacle Avoidance Car ¥ Obstacle_Avoidance_Car ¥ Obstacle_Avoidance_Car.ino” にコードファイルを開きます。プログラムを UNO ボードにアップロードします。



Code preview:

```
//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();
}
}
}

```

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、雲台が停止し、障害物を回避する方向に車両が変わります。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.原則の導入

まず、SG90 サーボについて学びましょう：

SG90 Servo

**180 angle steering
gear**

**Rototion angle is
from 0 to 180**

Brown line —GND

Red line —SV

Orange line —signal(PWM)



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, red line—5V, orange—signal.

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、サーボは基本 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、および天使を決定する。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：

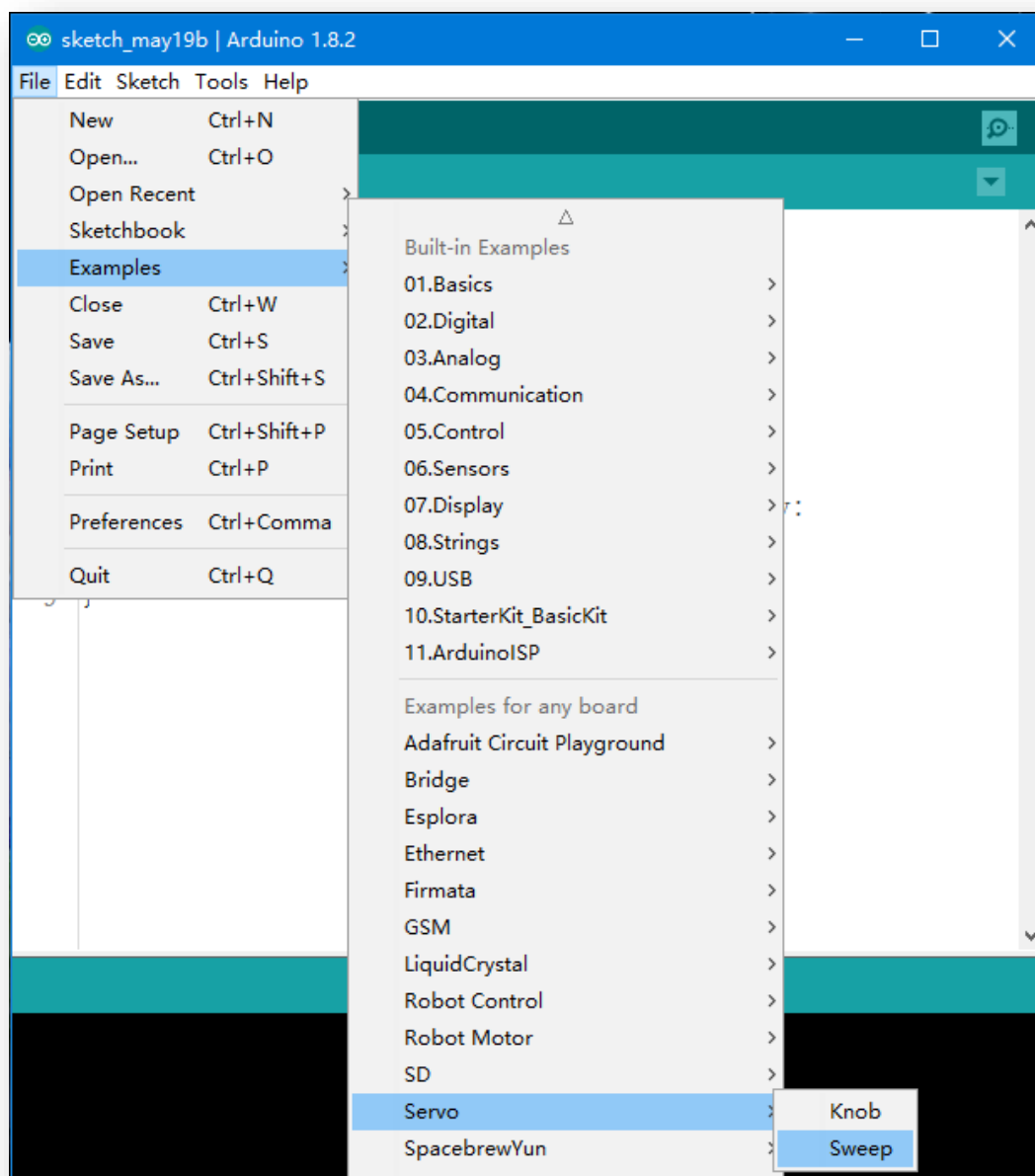
サーボの回転を制御するには、時間パルスを約 20ms、高レベルのパルス幅を約 0.5ms~2.5ms にする必要があります。これはサーボの角度制限と一致しています。

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

サンプルプログラム：

Arduino IDE を開いて選択“File->Examples->Servo->Sweep”



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。

製品の応用： ロボット障害物回避、物体試験距離、液体試験、公安、駐車場テスト。

主な技術パラメータ

- (1) : 使用電圧： DC---5V
- (2) : 静電流： 2mA 未満
- (3) : レベル出力： 5V より高い
- (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 0.2cm

接続方法： VCC、trig（制御終了）、エコー（受信終了）、GND

モジュールはどのように機能するか：

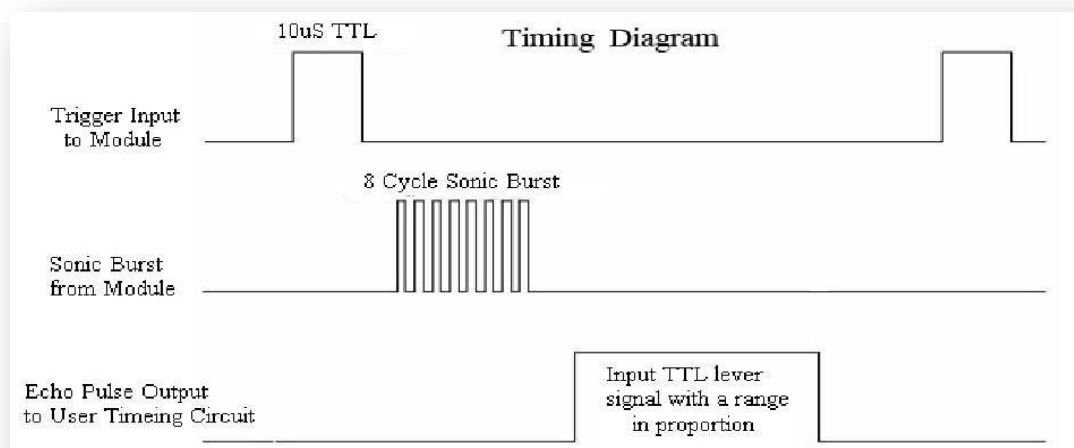
- (1) TRIG の IO ポートをトリガー・レンジに適用し、高レベルの信号、少なくとも $10\mu s$ を一度与えます。
- (2) モジュールは 40kHz の 8 つの方形波を自動的に送信し、信号が自動的に返されるかどうかをテストします。
- (3) 信号が受信されると、モジュールは ECHO の IO ポートを介してハイレベルのパルスを出し、ハイレベルのパルスの持続時間は送受信間の時間です。したがって、モジュールは時間に応じて距離を知ることができます。

$$\text{Testing distance} = (\text{high level time} * \text{velocity of sound (340M/S)}) / 2;$$

Actual operation:

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

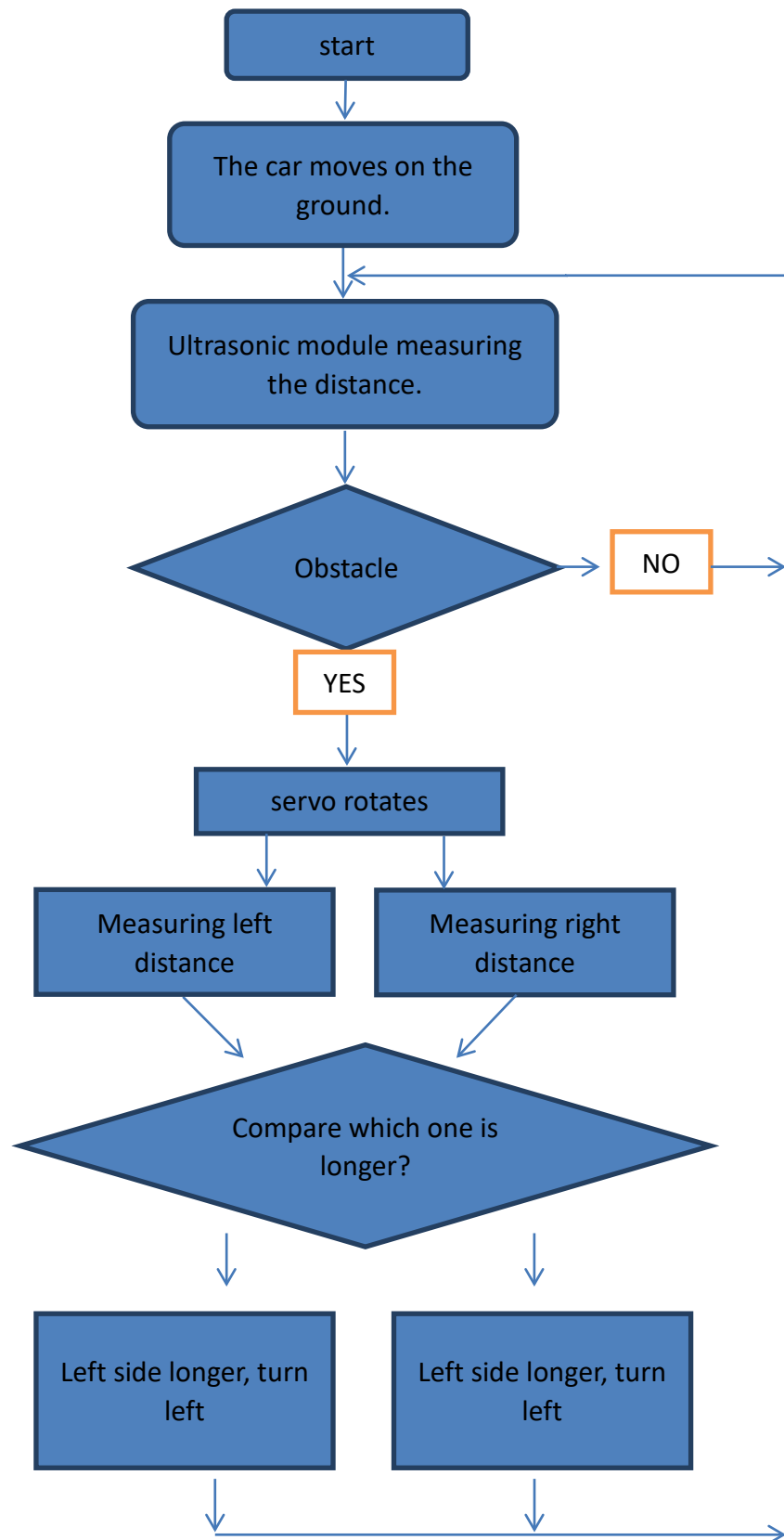
ジュールは 40kHz で 8 サイクルの超音波バーストを送信し、エコーを上げます。 エコーは、パルス幅と距離範囲の距離オブジェクトです。トリガー信号を送信してからエコー信号を受信するまでの時間間隔で、範囲を計算できます。 数式： $\mu\text{S} / 58 = \text{センチメートル}$ または $\mu\text{S} / 148 = \text{インチ}$ ；または：範囲=高レベル時間*速度（340M / S） / 2；エコー信号へのトリガ信号を防止するために、60ms 以上の測定サイクルを使用することを推奨します。



/*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;
}
```



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する。

Code preview:

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