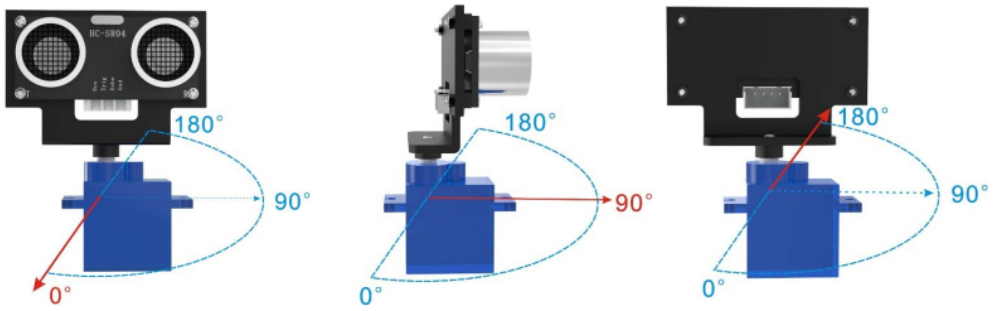
Aberystwyth Robotics Club - Elegoo Robot
Obstacle Avoidance

Introduction

In this worksheet we'll learn how to use the ultrasonic sensor to avoid obstacles.

Aligning the servo

A servo is a small motor which can measure the amount the shaft has turned. In our case we will want to know how far the servo has turned in degrees. But before we start, we need to make sure the sonar is aligned. To do this, we will set the position of the servo to a desired point, and adjust the alignment of the ultrasound sensor on top if necessary. The servo has a range from 0°to 180°. When it is at 90°, it should be pointing straight forward.



Create a new program with the code below to set the servo postion.

```
#include <Servo.h>   // library required for interacting with servo
Servo myservo;

void setup(){
  myservo.attach(3); //Specify the pin the servo is connected to
  myservo.write(90); // move servos to center position -> 90 degrees
}
void loop(){

}
```

After running the program, **if** the sonar is not pointing forward then:

1. Disconnect the cable from the back of the sonar.
2. Unscrew the screw between the sonar and the blue servo.
3. Carefully pull up the sonar mount
4. If the servo has moved at all, rerun the program above, otherwise reattach the sonar pointing straight forward.
5. Reconnect the cable to the sonar.

Reading from the sonar

The ultrasound sensor works by sending out high frequency sound ways on one side, and measuring how long before they are received on the other side. Using this, we can estimate how far away obstacles are. The side that sends out the signal is typically called the *Trigger* or trig for short, while the side that receives the signal is called the *Echo*.

Code

Remember that all Arduino programs have two parts: the setup which happens once, and is used to set up anything you need to use in the program. This is where you tell the Arduino which components are wired up to which pin, and stuff like that. There is also a loop, and the loop is repeated again and again until the Arduino is turned off (or runs out of battery power).

In the setup function here we need to tell the Arduino which pin has the Trig connection, and which pin has the Echo connection.

We are going to use the trigger pin to send a message to the sensor from the Arduino (so that is an OUTPUT pin), but we are going to use the Echo pin to read a message from the sensor (so that is an INPUT pin).

We are also going to incorporate the servo so we can 'look' in different directions with the ultrasonic sensor. The `Servo.h` library provides instructions to help control the servo, e.g. allowing us to specify an angle to turn to.

Building on your line following program from last time:

1. At the very top of your program, add the library for the servo and a variable (myservo) we can use to refer to the servo later on.

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

2. Define the pins echo, trigger and servo.

```
int Echo = A4;
int Trig = A5;
```

3. Set them to input and output in the setup function. Note the servo is written slightly differently, making use of its library.

```
pinMode(Echo, INPUT);
pinMode(Trig, OUTPUT);
myservo.attach(3);  // attach servo on pin 3 to servo object
```

4. Next, we'll add a function that will send out a brief signal from the Trigger and measure how long it takes to come back on the Echo.

```
//Ultrasonic distance measurement Sub function
int Distance_test() {
  digitalWrite(Trig, LOW);   //reset Trigger
  delayMicroseconds(2);
  digitalWrite(Trig, HIGH);  //Send signal
  delayMicroseconds(20);     //Wait 20 micro seconds
  digitalWrite(Trig, LOW);   //stop signal
  float Fdistance = pulseIn(Echo, HIGH);  //record delay on Echo
  Fdistance= Fdistance / 58; //convert range to cm
  return (int)Fdistance;
}
```

   The formula for measuring distance is as follows:

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

   Which gives, delay in microseconds (μS) / 58 = distance in centimeters

5. In the loop, you can now start to obtain readings from the ultrasonic sensor and print them in the serial monitor by adding the code below. Remember to comment out any other code you may currently have in the loop function.

```
Serial.println(Distance_test());
delay(500);
```

   Moving your hand in front of the ultrasound what sort of range can you get?

Controlling the servo

The servo has a range of 180°. Forward is at 90°so we can move it from 0 to 180.

Below is some code that can be added inside the loop function to move the servo all the way from 0° to 180° in 1° steps.

```
for (int pos = 0; pos <= 180; pos += 1) { // goes from 0 degrees to 180
    degrees
                    // in steps of 1 degree
  myservo.write(pos); // tell servo to go to position in variable "pos"
  delay(20); // waits 20ms for the servo to reach the position
}
for (int pos = 180; pos >= 0; pos -= 1) { // goes from 180 degrees to 0
    degrees
  myservo.write(pos); // tell servo to go to position in variable "pos"
  delay(20); // waits 20ms for the servo to reach the position
}
```

We are using a programming structure here called a 'for-loop'. This consists of a counter, a termination condition and an amount to increment by on each loop. Starting from the initial value (pos=0), the condition is checked (pos<=180), then if true, the code inside the curly brackets is executed before increasing the counter by the specified amount (pos+=1). The conidition is then checked again and the block of code with be repeated until the condition is no longer true.

We start the position of the servo at 0 degrees, if the position is less than or equal to 180 degrees, then add one degree to the position variable.

Now we have the value 1 in variable 'pos', myservo will move 1 degree and wait 20 milliseconds.

Once it has moved 1 degree and waited, the code returns to the top and adds an extra degree to the position variable and loops this over and over until the position variable reaches 180.

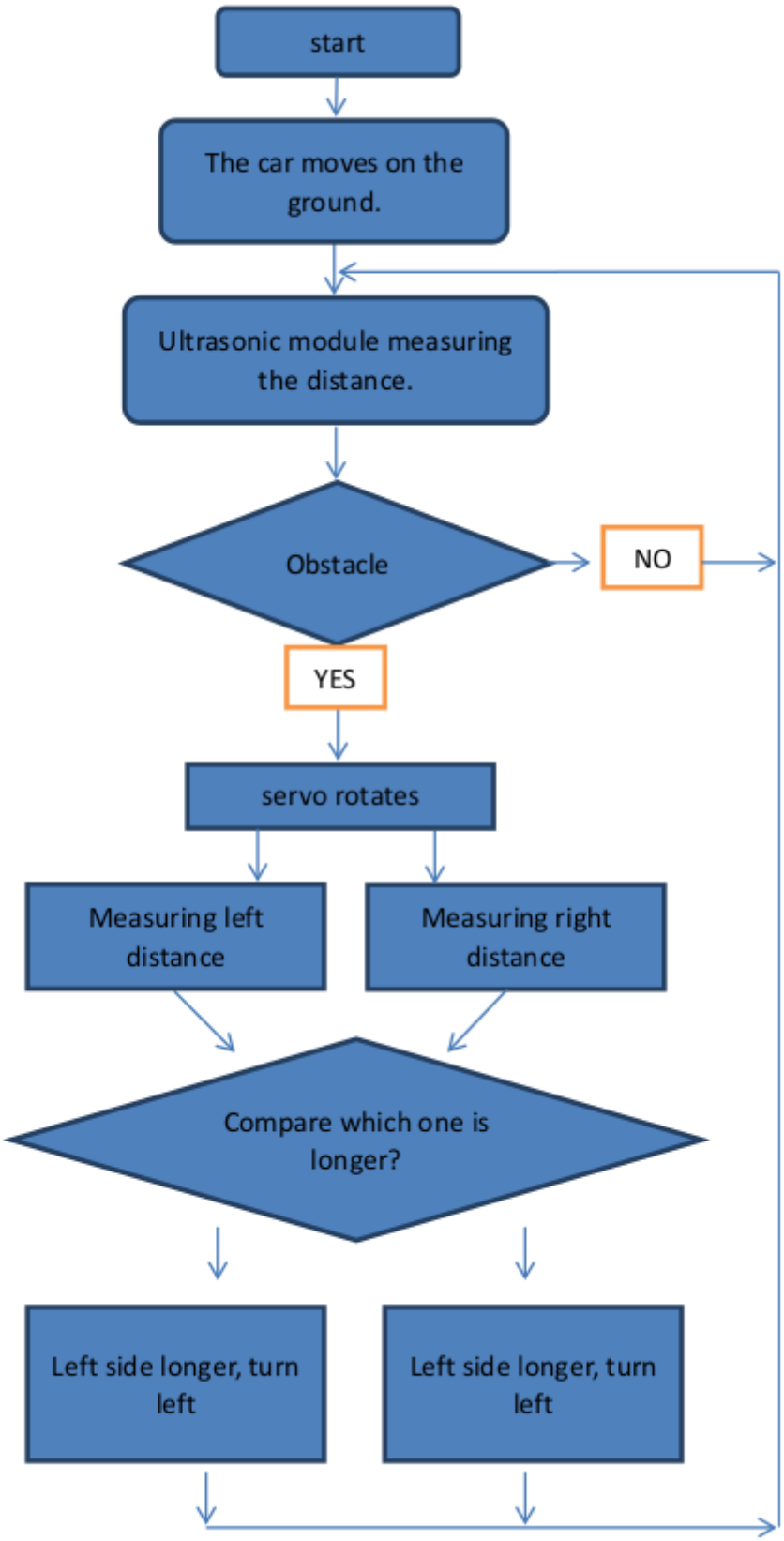
Once the servo reaches 180 degrees, it jumps outside the for loop, then moves along to the next for loop where it does exactly the same action but in reverse from 180 to 0.

Note: You may find the wires get in the way as it turns, or start to pull out the connector to the sensor. If this happens reduce the range of rotation, e.g. 30° to 150°.

Avoiding obstacles

We now have all the components to drive around whilst avoiding obstacles. The principle is to drive around until an obstacle is detected. When something gets in the way, we should then use the servo to look left and right and decide which way to turn before continuing.

Before turning the page, think about what needs to be done in each loop. Discuss this with your partner and plan it out on paper. The following can be considered as hints for both the planning and implementing.


start
The car moves on the ground.
Ultrasonic module measuring the distance.
Obstacle
NO
YES
servo rotates
Measuring left distance
Measuring right distance
Compare which one is longer?
Left side longer, turn left
Left side longer, turn left

1. In the loop, lets start by adding some variables to record the distances:

```
int rightDistance = 0, leftDistance = 0, middleDistance = 0;
```

2. Next set the servo to point forward, wait for the movement to complete and then use the distance test function defined above to get a value for the middle distance.

```
myservo.write(90);
delay(100);
middleDistance = Distance_test();
```

3. We will now use the if-then-else structure from the line-following to change the behaviour based on the middle distance.

```
if(middleDistance <= 20) {
  ?????
}
else {
  ?????
}
```

4. If there is an obstacle detected, then the first thing we should probably do is stop moving. If there isn't an obstacle in the way, we can carry on driving forward.

```
if(middleDistance <= 20) {
  setStop();
  delay(500);
  ?????
}
else {
  setForward();
}
```

5. In order to decide which direction to turn, we can first turn the servo to obtain readings to the left and right. We'll use set positions of the servo, e.g. 20 and 160. Below is the code for the right, fill in the code for getting the reading to the left as well. Remember to recentre the servo after taking the two tests.

```
if(middleDistance <= 20) {
  setStop();
  delay(500);
  myservo.write(20);
  delay(1000);
  rightDistance = Distance_test();
  delay(500);
  ?????
}
```

6. Now that we have the readings, we need to compare them and make a decision about which way to turn the robot. This will use another if-then-else structure. You can adjust the delays so that your robot turns approximately 90°.

```
if(middleDistance <= 20) {
  ...//Code to get the left and right readings
  if(rightDistance > leftDistance)
  {
```

```
    setRightForward();
    delay(360);
  }
  else if(leftDistance > rightDistance)
  {
    ?????
  }

}
```

7. There is one final condition we should consider. If the readings to the left and right both indicate obstacles, we may well be in a deadend. In this case, we need to back out or turn 180° to continue. The condition for checking this is given below, and should be checked before turning left or right, note the addition of the 'else' to the following if-statement. You should fill in what movement you want it to make.

```
  if((rightDistance<=20) && (leftDistance <= 20)){
    ?????
  }
  else if(rightDistance > leftDistance){
    ...

}
```