School of Engineering, Discipline of Mechanics and Mechatronics, pp. 1–7

Lab 3: Ultrasonic Sensors and Conditional Programming

ENGG1500
Semester 1 2020

Nicholas O'Dell, Brenton Schulz, Dylan Cuskelly.

---

1 Introduction

In this lab you will use the HC-SR04 ultrasonic sensor to measure distance and perform basic vehicle control using sensory feedback.

Prework: Read the Arduino reference pages on `if()` and `if-else` statements:

- https://www.Arduino.cc/en/reference/if
- https://www.Arduino.cc/en/reference/else

Figure 1: HC-SR04 Ultrasonic Transducer.

Task: Attach an ultrasonic sensor to the front of the vehicle using the 3D printed parts as shown in Figure 2 (see the assembly video on Blackboard for instructions).

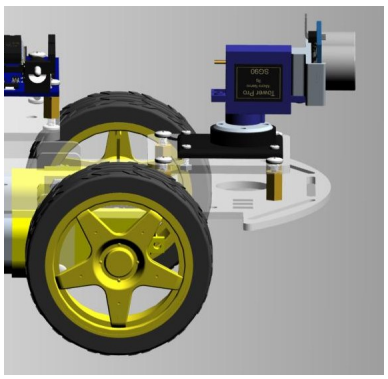

Figure 2: HC-SR04 mounted to vehicle chassis.

Task: Connect the ultrasonic sensor to your Arduino board as defined by the schematic in Figure 3 and the pin assignments in Table 1.

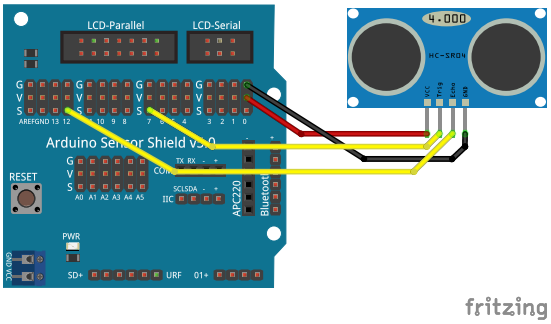

Figure 3: HC-SR04 connection schematic.

Table 1: Ultrasonic sensor pin assignment.

Arduino	Sensor
5V	VCC
GND	GND
TRIG	7
ECHO	12

2 Programming With The Ultrasonic Sensor

An ultrasonic sensor measures distance by emitting a burst of high frequency sound and measuring the delay before an echo is received. An ultrasonic transducer (speaker) emits this pulse, an ultrasonic microphone receives it, and electronics measures the delay. The conversion from a time delay to distance units is performed on a microcontroller such as your Arduino.

The sensor you have been supplied with is the HC-SR04, purchased from AliExpress/eBay. This device has two interface pins: TRIG and ECHO. A short pulse ($\approx 10\mu s$) present on the TRIG pin will cause the module to emit and ultrasonic *ping*. Once the ping has been transmitted the ECHO pin will become logic 1 (+5 V) and stay at logic 1 until an echo is received by the sensor or a timeout has occurred (no echo was reflected). By measuring the width of the echo pulse the round trip ping time can be measured and a distance calculated. This is demonstrated in Figure 4.

In your Arduino code a 10 microsecond pulse can be produced using: `digitalWrite()` and `delayMicroseconds()`. The pulse length of the ECHO signal is then measured using the advanced I/O function: `pulseIn()`, this function returns the width of pulse detected on a given pin in microseconds (For full documentation see: https://www.Arduino.cc/en/Reference/pulseIn).

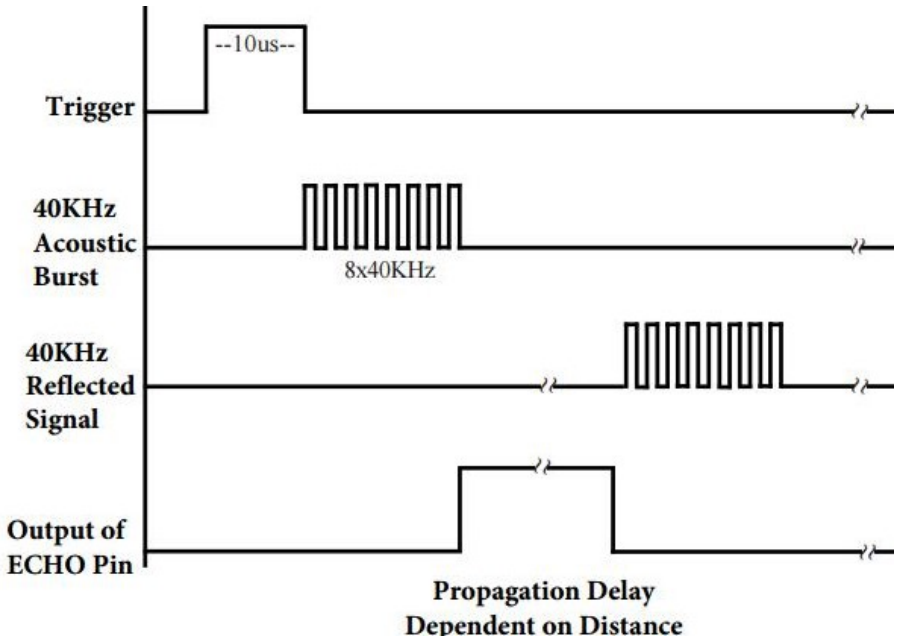

Figure 4: HC-SR04 timing diagram

Task: Retrieve the following code shown Listing 1 from Blackboard and upload it to your Arduino:

Listing 1: Measuring distance with HC-SR04.

```
// These #define statements make the code more readable.
// Instead of a pin number "7" or "12" we can write "ECHO" or "TRIG"
#define ECHO 12
#define TRIG 7

void setup() {
  pinMode(ECHO,INPUT); //Initialise pin 12 as an input
  pinMode(TRIG,OUTPUT); //Initialise pin 7 as an output
  Serial.begin(9600); //begin serial communication
}

void loop() {
  unsigned int distance_mm = 0; //This variable will hold the distance
  distance_mm = sonar_mm(); //call the function sonar_mm and store the result in
       //...distance_mm
  Serial.print("Distance="); //print the result to the serial monitor
  Serial.println(distance_mm);
}

unsigned int sonar_mm(void){
  long duration = 0;
  const float speed_sound = 340.29;// m/s, "const" makes the compiler able to optimise
       //... the program where this variable is used, cool!
  // Read in a distance from the ultrasonic distance sensor:
  // The ultrasonic burst is triggered by a HIGH pulse of 10 microseconds.
  digitalWrite(TRIG, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG, LOW);
  //read length of time pulse
  duration = pulseIn(ECHO, HIGH); //This function measures a pulsewidth and returns
       //...the width in microseconds
  // convert the time into a distance
  // the code "(unsigned int)" turns the result of the distance calculation
  // into an integer instead of a floating point (decimal or fractional) number.
  return (unsigned int)(0.5 * duration * 1e-6 * speed_sound * 1e3);
  //"unsigned" ensures we are returning an unsigned number, remember that there is no
       //...such thing as negative distance.
}
```

Open the serial monitor, **(Ctrl+Shift+M)**, and check that the distance is being printed to the console.

3 Conditional Programming with Ultrasonic Sensor

Task: The code extract in Listing 2 uses an `if()` statement to control when the distance is printed. Merge this extract with your code so that the distance is only printed when it is less than 200 mm.

Listing 2: `if()` statement introduction.

```
void loop() {
  unsigned int mm = 0; //declaring distance

  mm = sonar_mm();//call the sonar function

  if(mm < 200) //the code within this if statement only gets executed if this
       //...statement is true.
  {
    Serial.print("Distance="); //i.e. the distance will only print to the serial
         //...monitor if the distance is less than 200mm
    Serial.println(mm);
  }
  delay(100);
}
```

Task: Modify your code, using an `if()` statement, so that the motors move forward whenever the measured distance is greater than 100 mm. To perform this task you can use either of the following pseudo code algorithms:

Listing 3: Pseudo code algorithm 1

```
BEGIN
  mm = sonar_mm()

  IF mm <= 100
    Stop motors
  ENDIF

  IF mm > 100
    Move forward
  ENDIF
END
```

Listing 4: Pseudo code algorithm 2

```
BEGIN
  mm = sonar_mm()

  IF mm <= 100
    Stop motors
  ELSE
    Move forward
  ENDIF
END
```

The 2nd algorithm (Listing 4) uses an `else-if` flow control mechanism. The Arduino syntax for this is shown in Listing 5. This is very useful, often you will want to do one thing if a certain condition is true and something else if the opposite is true, i.e., if the condition is false. The conditional statement `if-else` allows you to do this without checking the condition twice. For example if a digital pin reads `HIGH`, there is no need to also check if it is `LOW`, or if you have determined that a distance is greater than or equal to 100 mm there is then no need to also check if it is less.

Listing 5: `if else` example

```
if(condition) {
    // Executed if condition is non-zero (true)
} else {
    // Executed if condition is zero (false)
}
```

Task: Modify your code to make the vehicle move forward while the distance is more than 120 mm and backwards if the distance becomes less than 60 mm. This will cause the vehicle to park at 120 mm from an object and reverse if something moves in front of it.

Extension: The distance measured by an ultrasonic sensor can be *noisy*[1] and change radically even when the sensor is stationary (you may have experienced this today in the lab). If you are using data from the ultrasonic sensor to perform accurate operations you may need to consider a method of *smoothing* this data. The following Arduino tutorial contains some useful information: https://www.arduino.cc/en/tutorial/smoothing.

4 Required Task for Next Week

Next week (week 4) all disciplines have a task to complete in their respective workshops. We will place your robot in a square enclosure, shown in Figure 5, at an arbitrary starting location. The enclosure is surrounded by walls but is otherwise free from obstacles. The goals are:

1. Don't hit the walls.
2. Get to the end zone, marked by the black square.
3. Remain stationary once reaching the end.

This goal is achievable with the content from this lab and last week's lab (week 2).

Tip
Look at the **Project Reference Manual** on Blackboard for information about how to develop a list of program goals (like the ones listed above) into a working program by using tools such as UML diagrams and pseudo code.

[1] This noise is Gaussian which means that the mean, median and mode are all the same value and the profile around this value is a Normal distribution or bell curve.

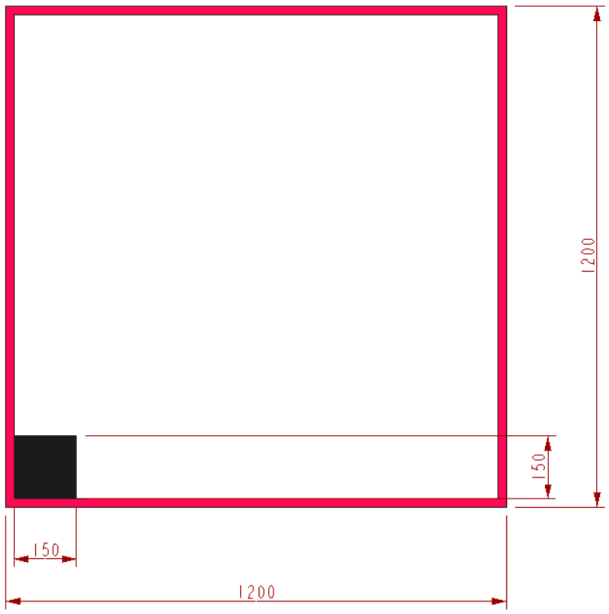

Figure 5: Robot Arena; all walls are 200 mm high.

Hint
Recall from last weeks lab when we used the line sensor data to make decisions on stopping the robot.

5 Recommendations

Here are some suggestions what to work on next:

- Take the UML diagrams and pseudo code you developed from last week's recommendations[2] and develop them into C/Arduino code ready to implement and test on your project ASAP. At this stage you have everything you need to get individual tasks functioning on your project, we will help you tie them together in the next two labs.
- Think about how you can use the ultrasonic sensor for your project, specifically how you might use a combination of different sensor data to determine which part of the track you have entered and hence what control action to take.
- Using the ultrasonic sensor can be slow compared to light based sensors because the speed of sound is significantly slower than light. The `pulseIn()` function (https://www.Arduino.cc/en/Reference/pulseIn) allows for a time-out to be specified. This prevents the function from waiting for longer than required. E.g., if you know that you never need to read a distance greater than 1 m, determine the largest time the function needs to wait (remember to double it). Simply call it with a third argument: `duration = pulseIn(pin, value, timeout);`

[2] https://i.pinimg.com/564x/36/03/62/360362fcc234564d683cd330d0aca016--my-student-loans-a-student.jpg