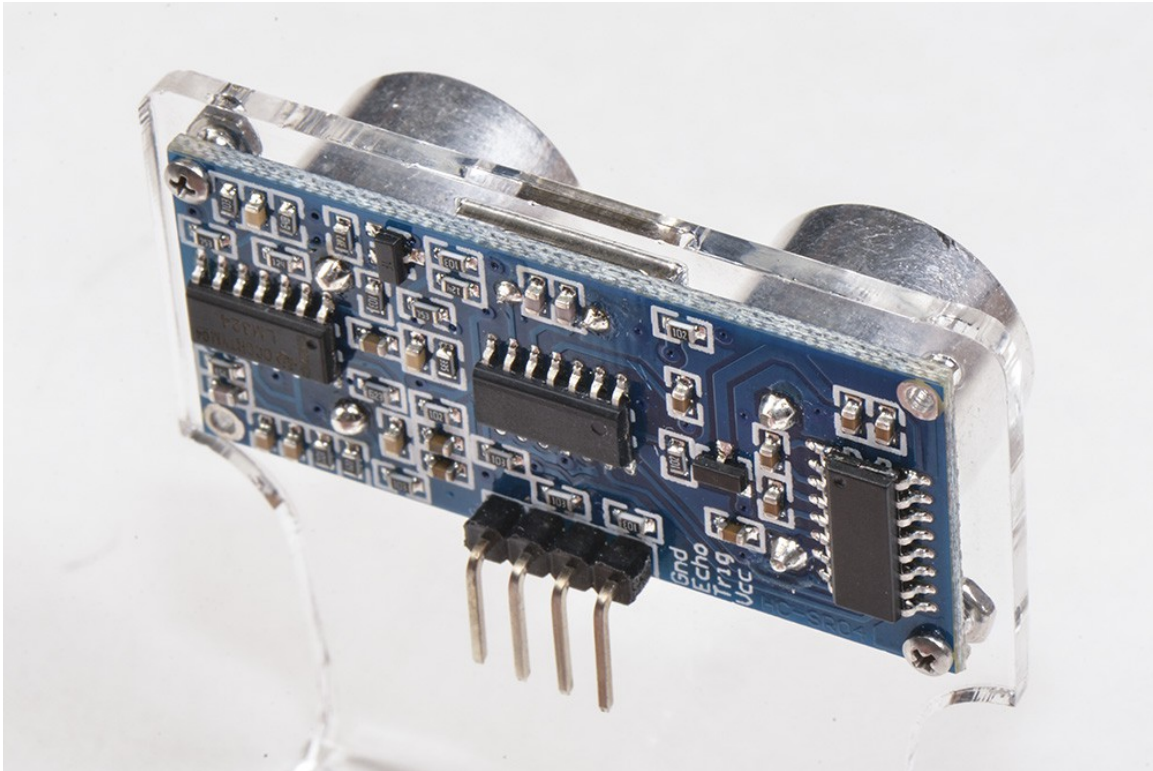


## LESSON 5

### **Programming the Distance Sensor**

The HC-SR04 ultrasonic distance sensor board is a practical component for your Pi-Bot. Once programmed, your Pi-Bot will be able to detect and determine the distance of an object in front of it.



*Figure 5.1*

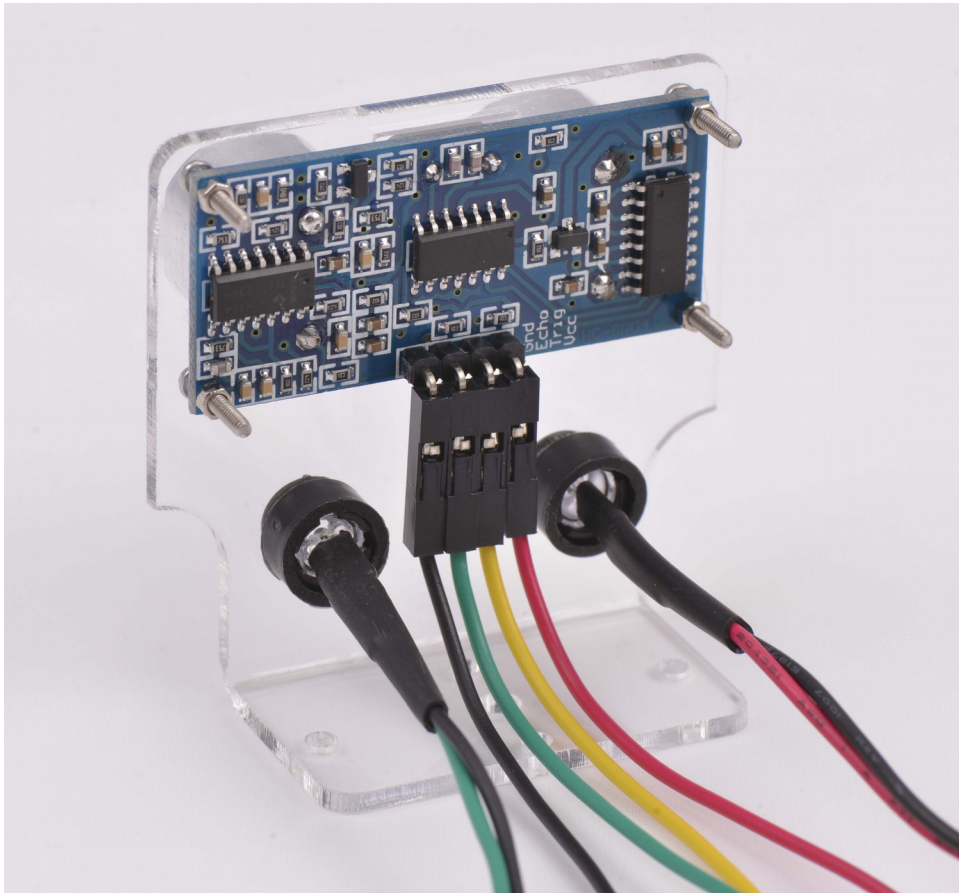
Before we begin, let's take a moment to make sure the ultrasonic sensor is correctly wired.

Four (4) 20cm wires should be connected to the rear pins on your ultrasonic sensor as follows:

- VCC – Red – to positive 5V
- Trig – Yellow – to Digital Pin 2
- Echo – Green – to Digital Pin 4
- Gnd – Black – to Gnd

Note: VCC is the board +5V power supply. Wires connected to VCC wires should be colored red. The Ground wires should be colored black. Consistency between these colors will help prevent accidental electrical damage to the components.

The assembled ultrasonic sensor and mount is pictured in figure 5.2.



*Figure 5.2*

Connect the USB cable to the STEM board microprocessor. Your distance sensor is now ready for testing!

### **Programming the Distance Sensor**

The SR04 ultrasonic detector is used to determine the distance of an object in front of your Pi-Bot. This is accomplished with the use of an ultrasonic transmitter and an ultrasonic receiver.

The sensor emits a chirp using the transmitter and listens for an echo on the receiver. The time it takes for the return echo to be detected is proportional to the distance to the object.

To demonstrate the use of the SR04 ultrasonic distance sensor, consider the program: [distance sensor test 01.ino](#), as shown in figure 5.3.

The trigger is defined on Pin 2 as an output. The echo is defined on Pin 4 as an input. The purpose of the program is to read the distance to an object and write it to the serial monitor.

Using the digital write on the trigger pin, the trigger pin is held LOW for 2 microseconds, followed by a HIGH pulse of 10 microseconds. This generates an ultrasonic pulse from the sensor. The program waits for an echo to bounce off an object in front of the sensor.

The time it takes for the echo is proportional to the distance to the object. To measure the time it takes for the echo to return, the `pulseIn` function is used. The time, which is stored in the variable `duration`, is converted to a decimal distance using the equation:  $(\text{duration}/2) / 29.1$ . Both the duration and the distance are printed using the serial print functions.

```
distance_sensor_test_01.ino

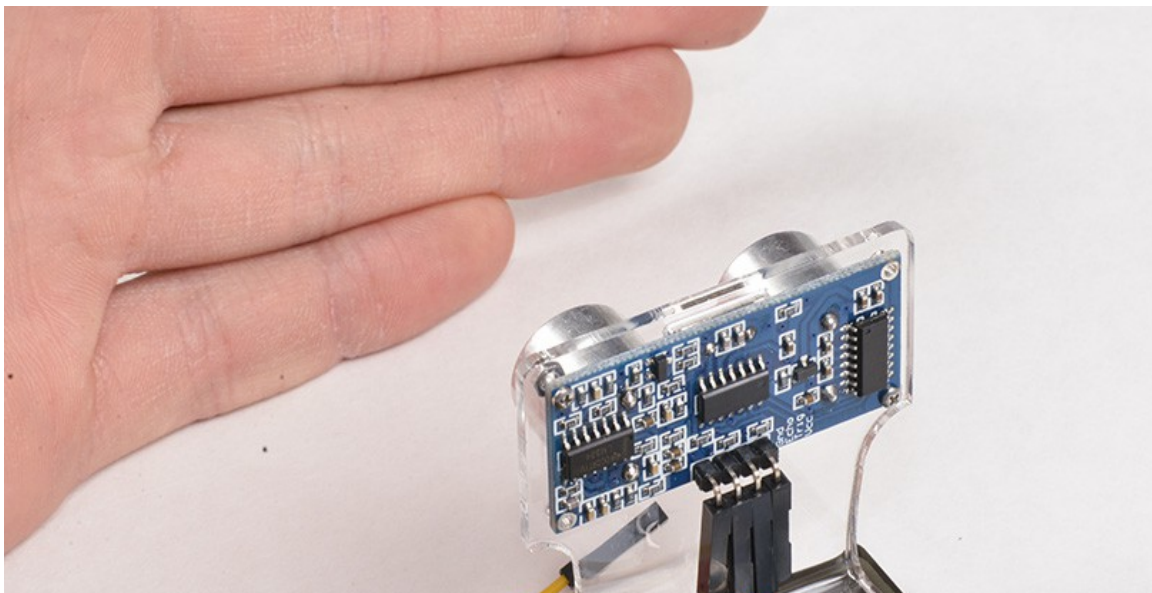
#define trigPin 2
#define echoPin 4

void setup()
{
  Serial.begin (9600);
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}

void loop()
{
  long duration;
  float distance;
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distance = (duration/2) / 29.1;
  Serial.print("Duration = ");
  Serial.print(duration);
  Serial.print("    Distance = ");
  Serial.println(distance);
  delay(50);
}
```

*Figure 5.3: Program – distance\_sensor\_test\_01.ino*

Open the serial monitor and wave your hand in front of the sensor to see it work, as shown in figure 5.4.



*Figure 5.4*

As another example of using the line sensor, consider the program: distance sensor test 02.ino, as shown in figure 5.5.

```
// distance_sensor_test_02.ino

#define trigPin 2
#define echoPin 4
#define left_led 9
#define right_led 10

void setup()
{
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  pinMode(left_led, OUTPUT);
  pinMode(right_led, OUTPUT);
}

void loop()
{
  long duration;
  float distance;
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  distance = (duration/2) / 29.1;
  if( distance < 8)
  {
    digitalWrite(left_led, HIGH);
    digitalWrite(right_led, LOW);
  }
  else if(distance >20)
  {
    digitalWrite(left_led, LOW);
    digitalWrite(right_led, HIGH);
  }
  else
  {
    digitalWrite(left_led, LOW);
    digitalWrite(right_led, LOW);
  }
  delay(50);
}
```

*Figure 5.5: Program – distance\_sensor\_test\_02.ino*

The program shown in figure 5.5 is similar to the previous program, with the serial print function removed and replaced with conditional tests.

The left LED is connected to Pin 9 and the right LED connected to Pin 10. If the distance to the sensor is less than 8cm, the left LED is turned **ON**. If the distance is greater than 20cm, the right LED is turned **ON**. If the object is between 8cm and 20cm from the sensor, both LEDs are turned **OFF**.

Again, test the program, by waving your hand in front of the sensor. Try to hold your hand in a position where both sensors are turned **OFF**.

## EXERCISE 1

1. Modify the program so that the left LED is normally **ON** and turns **OFF** when the distance is less than 10cm, and the right LED is also normally **ON** and turns **OFF** when the distance is greater than 12cm.
2. Try to hold your hand in front of the sensor so that both LEDs are turned **ON**.

This exercise is similar to what you will use for distance-based sensor control.

### **Advanced Programming of the Distance Sensor**

While the previous technique demonstrates the use of the ultrasonic sensor and can be used in many of your applications, it uses a lot of microprocessor time.

To allow your STEM board microprocessor to perform more than one task, such as sensing the distance using the ultrasonic and controlling the motors for line following, interrupts can be used.

Interrupt control is a technique which allows the microprocessor to temporarily suspends what is doing (such as controlling the motors) to service a change on an input pin (such as a timed ultrasonic echo response). This effectively allows the microprocessor to perform two tasks at the same time. Program: distance sensor advanced 01.ino, as shown in figure 5.6, demonstrates the use of interrupt control for the ultrasonic sensor.

This distance program demonstrates the use of interrupt control to periodically measure the distance to objects in front of the sensor.

```

// distance_sensor_advanced_01.ino

#define TrigPin 2 //will light
#define EchoPin 4 //interrupt-enabLed pin

#define INTNUM 1 //interrupt pin 1 is digital pin 4 on the Arduino
#define PULSE 10 //microseconds
#define CYCLETIME 50 //milliseconds - this is the time to wait until the sensor is asked to find
    // the distance again. #times/second = 1000/CYCLETIME = 20 in this case
#define Close 10 // Distance in cm to turn on the Led => Led is on if distance is less than 10 cm

void LedWrite(int), trigPulse(), getTime();
int millisNow, millisPrev = 0;
int microsPrev;

boolean isHigh = false;

void setup() {
    Serial.begin (9600);

    pinMode(TrigPin, OUTPUT);
    pinMode(EchoPin, INPUT);

    attachInterrupt(INTNUM, getTime, CHANGE); // Calls getTime when an change occurs on pin 4
}

void loop()
{
    trigPulse();
    // robot drive code will be added here later
}

void trigPulse()
{
    if( (millisNow = millis()) - millisPrev >= CYCLETIME) //sufficient cycle time
    {
        digitalWrite(TrigPin, HIGH);
        delayMicroseconds(PULSE);
        digitalWrite(TrigPin, LOW);
        millisPrev = millisNow; //reset clock
    }
    return;
}

void LedWrite(int dTime)
{
    noInterrupts(); //lets not get interrupted while we are processing the first interrupt
    int distance = dTime/58; // distance in cm

    Serial.print(distance);
    Serial.println(" cm");
    interrupts(); //turn interrupts back on
}

void getTime() //calling micros() here returns micros when function was called. No increment during interrupt
{
    if(isHigh == false) // pin LOW->HIGH
    {
        microsPrev = micros();
        isHigh = true;
        return;
    }
    else //pin HIGH->LOW
    {
        LedWrite(micros() - microsPrev);
        isHigh = false;
        microsPrev = micros();
        return;
    }
    return;
}

```

*Figure 5.6: Program – distance\_sensor\_advanced\_01.ino*

Connect the GRD to your microprocessor's ground, the trigger pin to digital pin 2, the echo pin to digital pin 4, and the VCC to +5 volts on the microprocessor. By monitoring the serial monitor, the distance as seen by the sensor is displayed.

Now, let's use the ultrasonic sensor to control an LED.

Connect an LED through the dropping resistor to digital pin 9. The short lead of the LED must connect to the microprocessor ground.

Using program: [distance\\_sensor\\_advanced\\_02.ino](#), as shown in figure 5.7, the LED will light whenever the distance to a object is less than 10cm.

This completes the basics for understanding the ultrasonic distance sensor.

```

// distance_sensor_advanced_02.ino

#define TrigPin 2 // trigger pin
#define EchoPin 4 //interrupt-enabled pin
#define Led 9 // led pin

#define INTNUM 1 //interrupt pin 1 is digital pin 4 on the Arduino
#define PULSE 10 //microseconds
#define CYCLETIME 50 //milliseconds - this is the time to wait until the sensor is asked to find
// the distance again. #times/second = 1000/CYCLETIME = 20 in this case
#define Close 10 // Distance in cm to turn on the Led => Led is on if distance is less than 10 cm

void LedWrite(int), trigPulse(), getTime();
int millisNow, millisPrev = 0;
int microsPrev;

boolean isHigh = false;

void setup() {
    Serial.begin (9600);

    pinMode(TrigPin, OUTPUT);
    pinMode(EchoPin, INPUT);
    pinMode(Led, OUTPUT);

    attachInterrupt(INTNUM, getTime, CHANGE); // Calls getTime when an change occurs on pin 4
}

void loop()
{
    trigPulse();
    // robot drive code will be added here later
}

void trigPulse()
{
    if( (millisNow = millis()) - millisPrev >= CYCLETIME) //sufficient cycle time
    {
        digitalWrite(TrigPin, HIGH);
        delayMicroseconds(PULSE);
        digitalWrite(TrigPin, LOW);
        millisPrev = millisNow; //reset clock
    }
    return;
}

void LedWrite(int dTime)
{
    noInterrupts(); //lets not get interrupted while we are processing the first interrupt
    int distance = dTime/58; // distance in cm

    Serial.print(distance);
    Serial.println(" cm");

    if (distance < Close)
    {
        digitalWrite(Led,HIGH);
    }
    else
    {
        digitalWrite(Led,LOW);
    }
    interrupts(); //turn interrupts back on
}

void getTime() //calling micros() here returns micros when function was called. No increment during interrupt
{
    if(isHigh == false) // pin LOW->HIGH
    {
        microsPrev = micros();
        isHigh = true;
        return;
    }
    else //pin HIGH->LOW
    {
        LedWrite(micros() - microsPrev);
        isHigh = false;
        microsPrev = micros();
        return;
    }
    return;
}

```

*Figure 5.7: Program – distance\_sensor\_advanced\_02.ino*