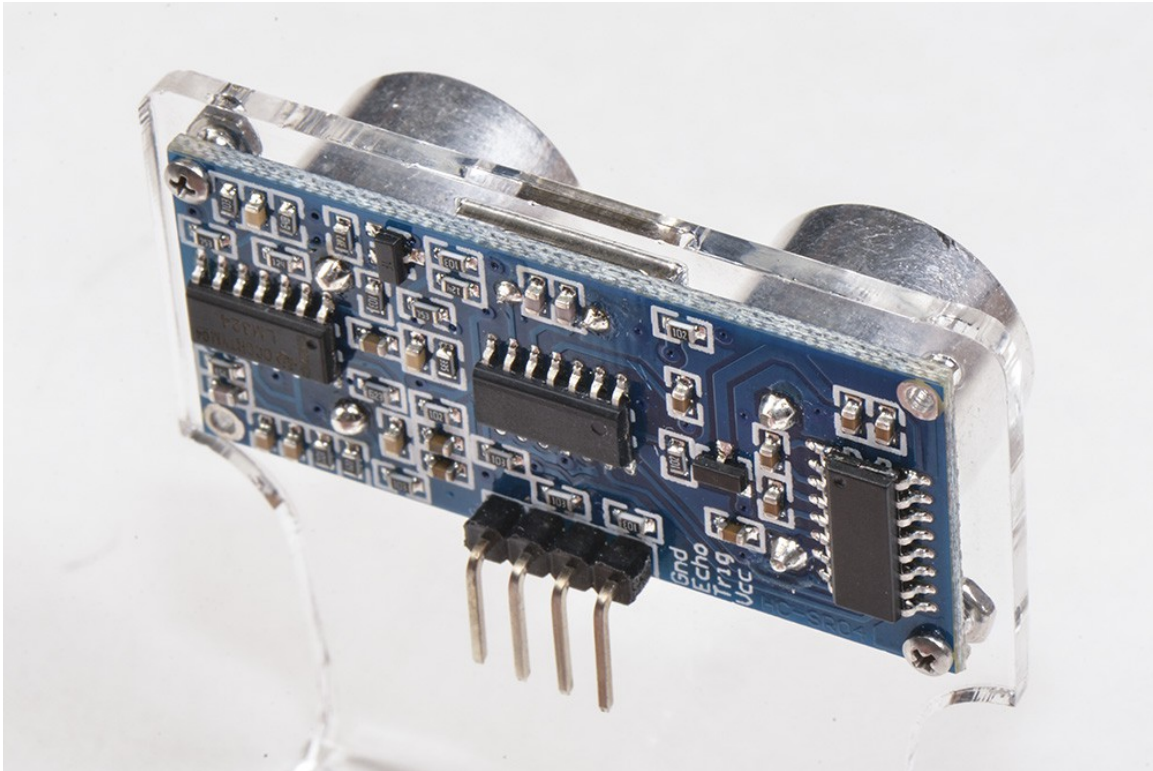


## LESSON 5

### **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 an object in front of it and determine its distance.



*Figure 5.1*

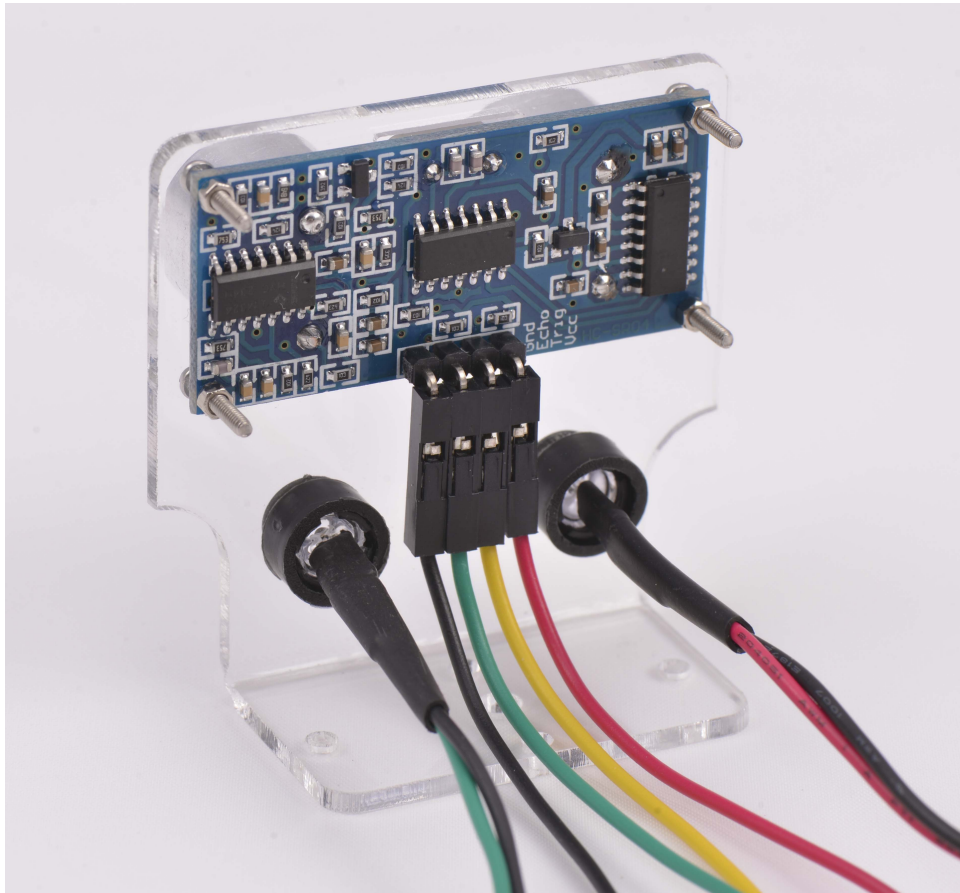
Before we begin, let's take a moment to make sure the ultrasonic sensor is correctly wired.

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 PWM 2
- Echo – Green – to PWM 8
- Gnd – Black – to Gnd

Note: VCC is the board +5V power supply and the wire is always colored red. The Ground wire is always 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 Arduino UNO board. 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.

It is important that the program does not spend all of its time polling the sensor to determine the distance to an object.

To allow the arduino processor to perform more than one task such as sensing the distance using the ultrasonic and controlling the motors for line following, interrupts will be used.

Interrupt control is a technique which allows the arduino 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 Arduino to perform two tasks at the same time.

A program demonstrating the use of interrupt control for the ultrasonic sensor is shown in figure 5.3.

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

```

/*
Ultrasonic sensor test
*/
#define TrigPin 2 //will light
#define EchoPin 3 //interrupt-enabled pin

#define INTNUM 1 //interrupt pin 1 is digital pin 3 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 3
}

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
}

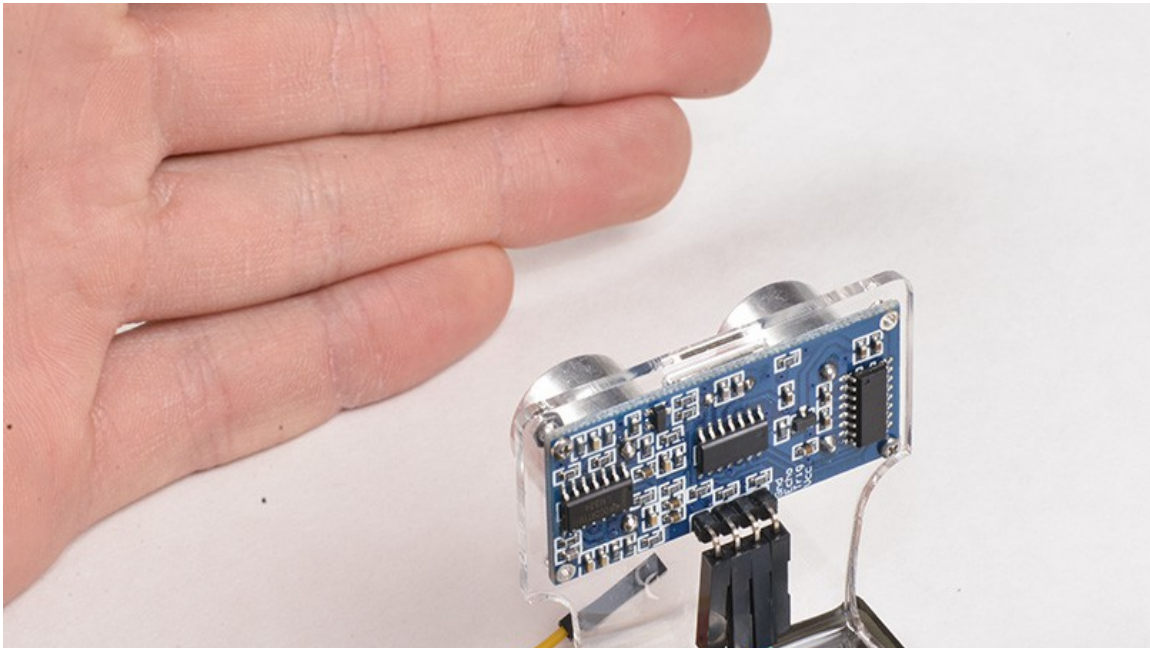
void getTime() //calling micros() here returns micros when function was called. No increment during interrupt
{
  if(isHigh == false) // pin LOW->HIGH
  {
    microsPrev = micros();
    isHigh = true;
    return;
  }
  else //pin HIGH->LOW
  {
    LedWrite(micros() - microsPrev);
    isHigh = false;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

```

*Figure 5.3: Program - sr04\_1a.ino*

Connect the GRD to Arduino ground, the trigger pin to digital pin 2, the echo pin to digital pin 3 and the VCC to +5 volts on the Arduino. By monitoring the serial monitor, the distance as seen by the sensor is displayed.

Wave your hand in front of the sensor to see it work, as shown in figure 5.4.



*Figure 5.4*

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

Connect a lead through the dropping resistor to Arduino digital pin 13. The short lead of the LED must connect to the microprocessor ground.

Using the program provided in figure 5.5, the LED will light whenever the distance to a object is less than 10cm.

This completes the basics for understanding the ultrasonic distance sensor.

```

/*
Ultrasonic sensor controlling a LED
*/
#define TrigPin 2 // trigger pin
#define EchoPin 3 //interrupt-enabled pin
#define Led 13 // led pin

#define INTNUM 1 //interrupt pin 1 is digital pin 3 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 3
}

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 5.5: Program - sr04\_1b.ino*