American International University- Bangladesh

Department of Electrical and Electronic Engineering

EEE 4103: Microprocessor and Embedded Systems Laboratory

Title: Building an Obstacle Detection System.

Introduction:

Choosing a microcontroller is how easy it is to develop products around it. Key considerations include the availability of an assembler, a debugger, a code-efficient C language compiler, an emulator, technical support, and both in-house and outside expertise. We have chosen Arduino series because a lot of community support is available online. Many free and efficient code samples from different projects are present for Arduino.

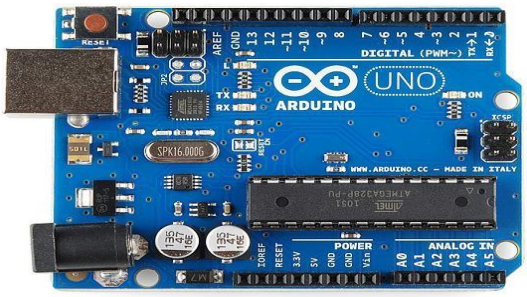
Objectives: In this lab, we will learn –

(i) How to code a simple Obstacle Detection System in Arduino IDE.
(ii) Implement a simple Obstacle Detection System in Hardware.
(iii) How to simulate the code in Proteus/Tinkercad and observe the results.

Theory and Methodology:

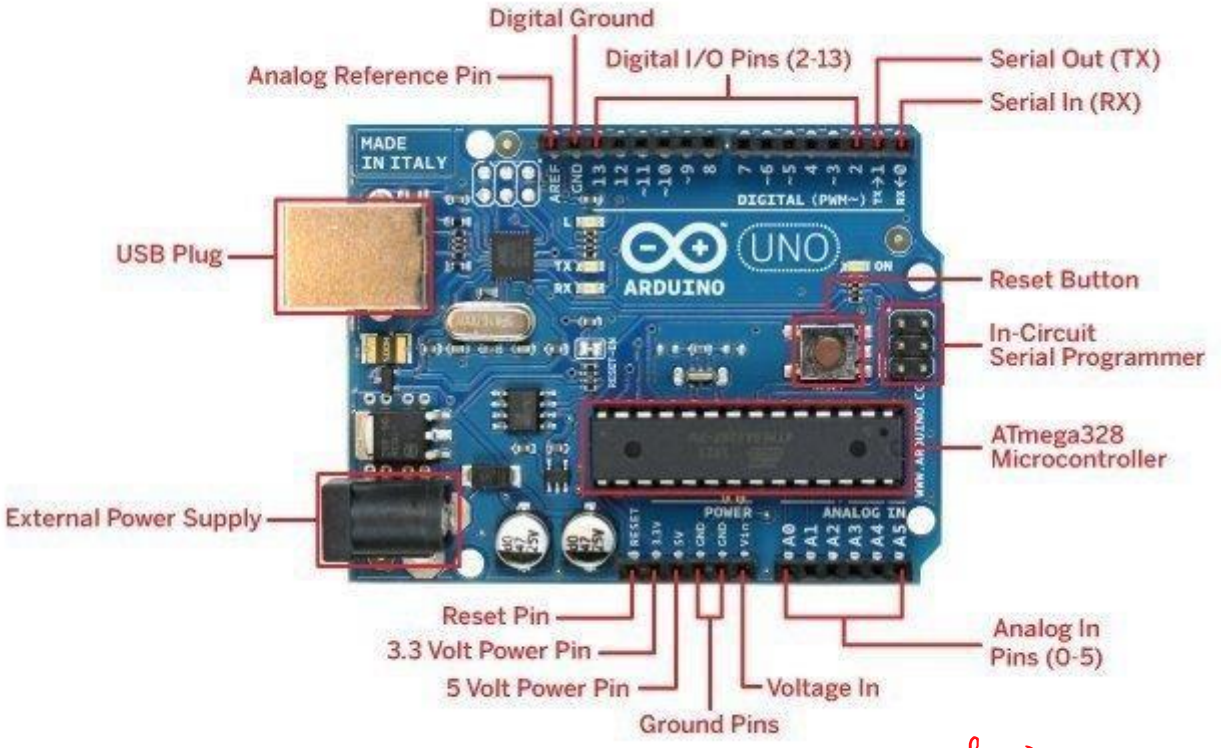
Arduino is an open-source platform used for creating interactive electronics projects. Arduino consists of both a programmable microcontroller and a piece of software, or IDE (Integrated Development Environment) that runs on your computer, used to write and upload computer code to the microcontroller board. Arduino Uno also doesn't need a hardware circuit (programmer/ burner) to load a new code into the board. We can easily load a code into the board just using a USB cable and the Arduino IDE (that uses an easier version of C++ to write a code).

Apparatus:

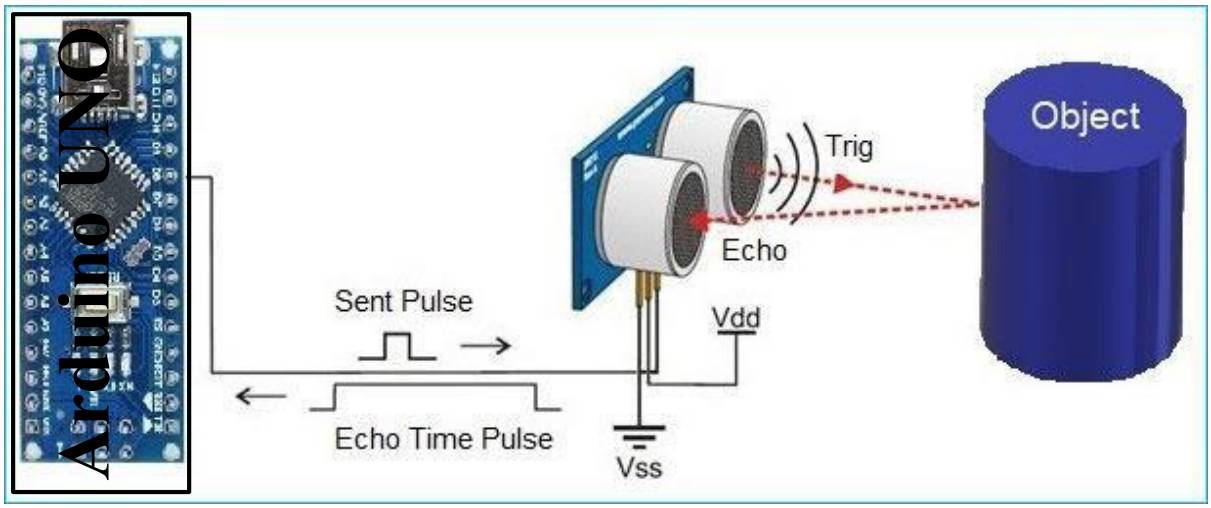
1) Arduino IDE (any version)	Software
2) Arduino Uno (R3) board	
3) Sonar Sensor (HCSR04)	

4) LED	

Overview of the Board (Arduino UNO R3):



Overview of Sonar Sensor:



Setting up the Arduino IDE:

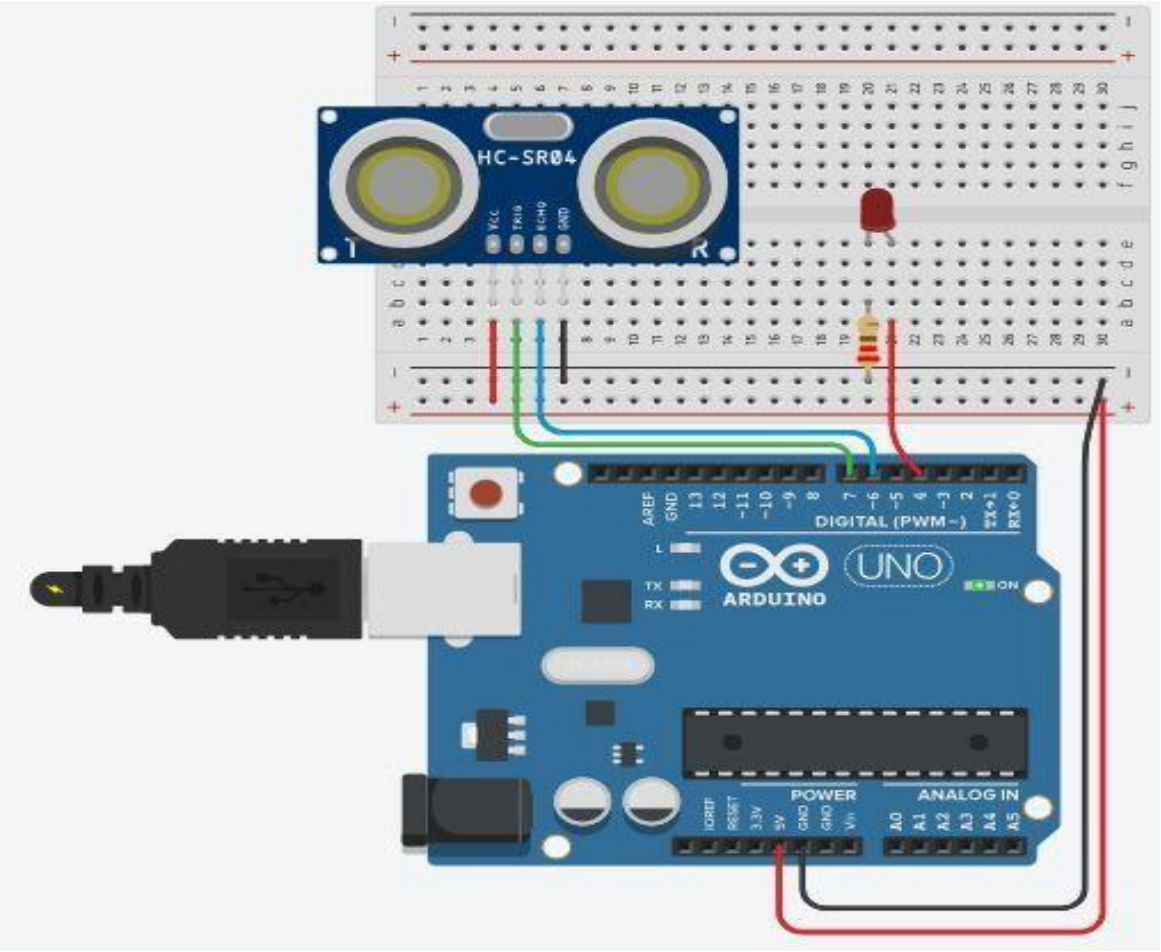
At first go to the following official Arduino IDE download page to download the latest version of Aduino IDE (before start downloading Aduino IDE, always keep in mind which operating system you are using in your PC).

Link: https://www.arduino.cc/en/Main/Software

Experimental Procedure:

Setting up the Circuit

The main task of our lab is to use a sonar sensor to detect the distance of an obstacle.



How It Works:

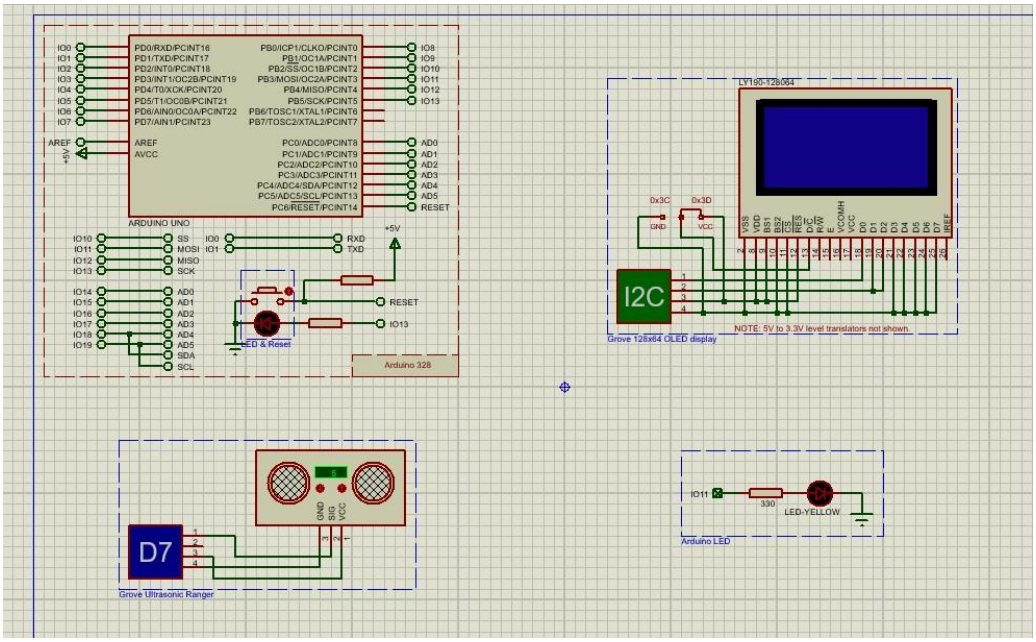
Here we are using a sonar sensor (HCS04) to detect the distance of an obstacle and will also glow an LED as soon as it detects the obstacle. HCSR04 is an Ultrasonic ranging module which consists of a transmitter, receiver and control circuit. It has four pins for VCC, GND, Trigger and Echo. You can easily interface it with Arduino boards. Using IO trigger for at least 10us high level signal, the Module automatically sends eight 40 kHz and detect whether there is a pulse signal back. The Trigger pin of the sensor is connected to digital pin 3 and the Echo pin at digital pin 2 of the Arduino Uno R3 board with connecting wire. And LED is connected to pin 13 to show that an obstacle is detected. Here pin 3 and 13 will act as output pins because trigger will be generated from Arduino and LED state (HIGH/LOW) will also be changed by the Arduino board. As the ping generated from the Arduino board travels out from the trigger and comes back to the echo, so to find the distance of the object we take half of the distance travelled. As we all know, the speed of sound is 340 m/s or 29 microseconds per centimeter or 74 microseconds per inch. So, the distance covered in centimeter by the trigger will be calculated by the following equation: -

distance_cm=microseconds / 29/ 2;
distance_in=microseconds / 74 /2;

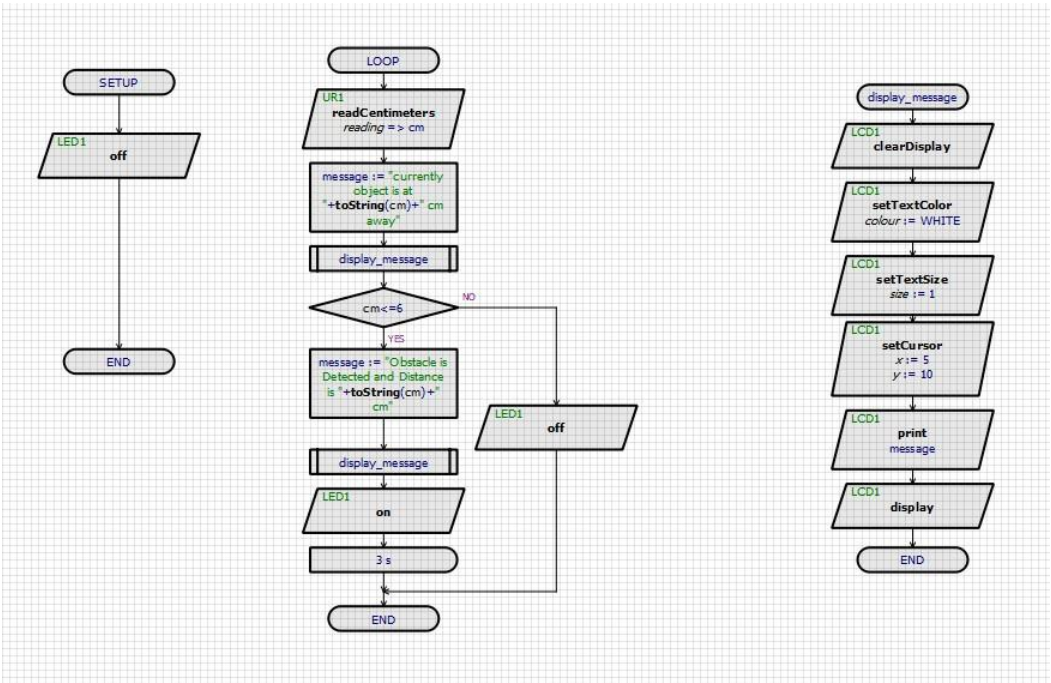
Simulating Ultrasonic Sensor with Arduino in Proteus:

Measuring distance using Ultrasonic Sensor Library with Arduino Library in Proteus.

STEP 1: Connect all components as shown in simulation figure.



STEP 2: design a flowchart for your system as follows:



Simulating the Code in Tinkercad:

Autodesk's Tinkercad is one of the most popular classroom tools for creating simple designs from scratch, quickly modifying existing designs. It's a free online 3D design program that you can use in your web browser without downloading any software. Tinkercad is extremely

intuitive and easy to use and has built-in Lessons to help you learn the ropes, making it perfect for beginners both young and old.

Go to tinkercad.com and set up a free account. At the top of your front page you can click Learn to follow walk-through Lessons that will teach you basic navigation and modeling techniques. From the Gallery you can browse models that you can Copy and Tinker to remix and modify, or just use the Search field to find models to explore. To begin creating your own 3D model, click Create New Design and start looking around; many of Tinkercad's tools and features are easy to figure out by experimenting. To learn more and level up your Tinkercad design skills, follow the reference links.

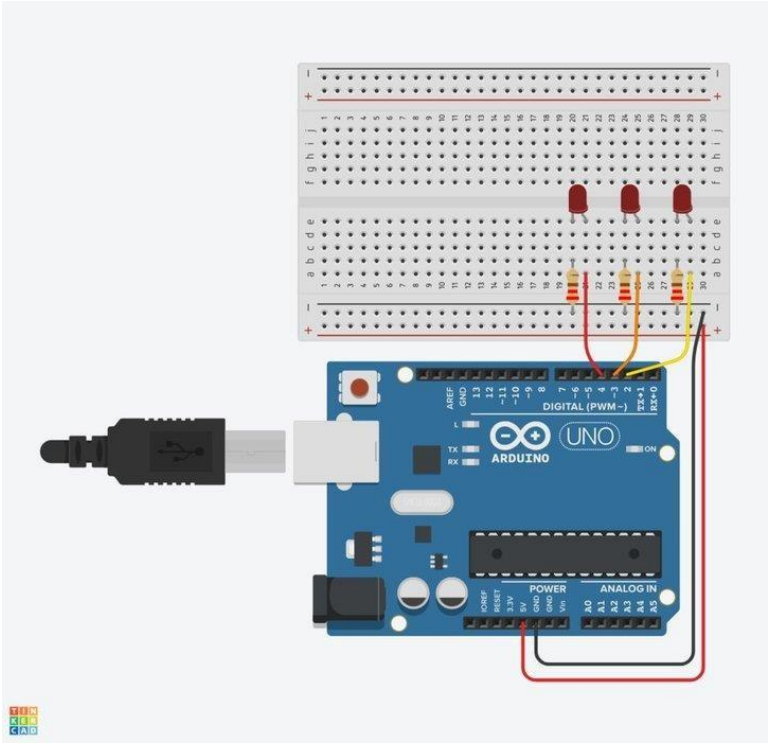
You can follow along virtually using Tinkercad Circuits. You can even view this lesson from within Tinkercad (free login required)! Explore the sample circuit and build your own right next to it. Tinkercad Circuits is a free browser-based program that lets you build and simulate circuits. It's perfect for learning, teaching, and prototyping.

Ultrasonic Distance Sensor in Arduino ~~With Tinkercad~~:

Let's measure distances with an ultrasonic rangefinder (distance sensor) and Arduino's digital input. We'll connect up a circuit using a breadboard and use some simple Arduino code to control a single LED.

Ultrasonic rangefinders use sound waves to bounce off objects in front of them, much like bats using echolocation to sense their environment. The proximity sensor sends out a signal and measures how long it takes to return. The Arduino program receives this information and calculates the distance between the sensor and object.

Step 1: Build the LED Circuit



Start by wiring up your Arduino and breadboard with power and ground next to the example circuit, then add the three red LEDs to the breadboard, as shown. These will be the "bar graph" lights for visually indicating the sensor's distance measurement.

Drag an Arduino Uno and breadboard from the components panel to the workplane, next to the existing circuit.

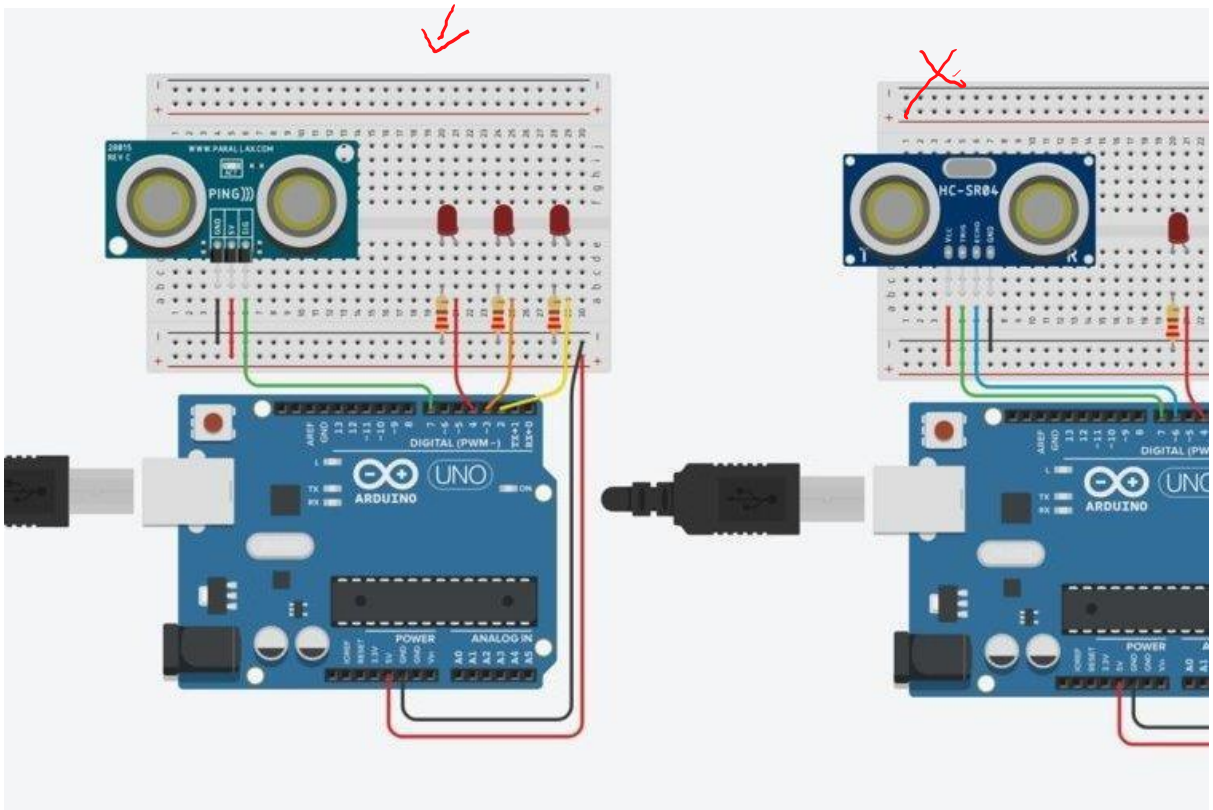
Connect the 5 volt and ground pins on the Arduino to the power (+) and ground (-) rails on the breadboard with wires. You can change the wire colors if you want to! Either use the inspector dropdown or the number keys on your keyboard.

Drag three LEDs on the breadboard in row E, spaced 2 breadboard sockets apart. You can change the LED color using the inspector that pops up when you click on each one.

Use a 220 Ohm resistor to connect each LED's cathode (left leg) to the ground rail (black) of the breadboard. You can change a resistor's value by highlighting it and using the dropdown menu.

Connect the LED anodes (right legs) to digital pins 4, 3, and 2 on the Arduino. The LED anode (+) is the terminal that current flows into. This will connect to the digital output pins on the Arduino. The cathode (-) is the terminal that current flows from. This will connect to the ground rail.

Step 2: Add Proximity Sensor



Proximity sensors come in multiple flavors. Here in Tinkercad Circuits, you can choose between a three-pin sensor or a four-pin sensor. In general, ultrasonic rangefinders have one pin that connects to ground, another that connects to 5 volts, a third for sending a signal, and a

fourth for receiving a signal. The 'send' and 'receive' pins are combined into one pin on the three-pin flavor.

In the circuits editor, find the ultrasonic rangefinder in the components drawer. To find the four-pin sensor, view "All" in the components panel (using the dropdown menu).

Place the sensor on the breadboard to the left of the LEDs in row E, as shown in the figure.

Wire up the sensor so the 5V pin connects to the 5V voltage rail, the GND pin connects to the ground rail, the SIG or TRIG pin to Arduino pin 7, and, if using the four-pin flavor, the ECHO pin connects to Arduino pin 6.

Step 3: Ultrasonic Rangefinder Arduino Code Explained
When the code editor is open, you can click the dropdown menu on the left and select "Blocks + Text" to reveal the Arduino code generated by the code blocks. Follow along as we explore the code in more detail.

```
int distanceThreshold = 0;
int cm = 0; int
inches = 0;
```

Before the setup(), we create variables to store the target distance threshold, as well as the sensor value in centimeters (cm) and inches. They're called int because they are integers, or any whole number.

```
long readUltrasonicDistance(int triggerPin, int echoPin)
{
  pinMode(triggerPin, OUTPUT);  // Clear the trigger
  digitalWrite(triggerPin, LOW);
delayMicroseconds(2);
  // Sets the trigger pin to HIGH state for 10
microseconds   digitalWrite(triggerPin, HIGH);
delayMicroseconds(10);   digitalWrite(triggerPin, LOW);
pinMode(echoPin, INPUT);
  // Reads the echo pin, and returns the sound wave travel time in microseconds
return pulseIn(echoPin, HIGH); }
```

The next section is a special bit of code for reading the ultrasonic distance sensor. It's called a function. So far you are familiar with setup() and loop(), but in this sketch, the function read UltrasonicDistance() is used to describe the sensor code and keep it separate from the main body of the program. The function definition starts with what type of data the function will return or send back to the main program. In this case the function returns a long, which is a decimal point number with many digits. Next is the name of the function, which is up to you. Then in parentheses are the arguments the function takes. int triggerPin, int echoPin are the variable declarations for your sensor's connection pins. The pin numbers will be specified when you call the function in the main program loop(). Inside the function, these local variables are used to reference the information you passed to it from the main loop (or from another function). The function itself sends a signal through the triggerPin and reports back the time it takes to get the signal back over echoPin.

```
void setup()
{
  Serial.begin(9600);
  pinMode(2, OUTPUT);
pinMode(3, OUTPUT);   pinMode(4,
OUTPUT);
}
```

Inside the setup, pins are configured using the pinMode() function. The serial monitor connection is established with Serial.begin. Pins 2, 3, and 4 are configured as outputs to control the LEDs.

```
void loop()
{
  // set threshold distance to activate
LEDs   distanceThreshold = 350;  // measure
the ping time in cm
  cm = 0.01723 * readUltrasonicDistance(7, 6);
```

In the main loop, distanceThreshold is set to its target 350cm.

```
// convert to inches by dividing by 2.54
 inches = (cm / 2.54);
Serial.print(cm);
  Serial.print("cm, ");
  Serial.print(inches);
  Serial.println("in");
```

To convert centimeters to inches, divide by 2.54. Printing to the serial monitor helps you observe the distance change more granularly than the LED states show alone.

```
if (cm > distanceThreshold) {
```

```
    digitalWrite(2, LOW);
digitalWrite(3, LOW);
digitalWrite(4, LOW);
  }
  if (cm <= distanceThreshold && cm > distanceThreshold - 100) {
digitalWrite(2, HIGH);     digitalWrite(3, LOW);
digitalWrite(4, LOW);
  }
  if (cm <= distanceThreshold - 100 && cm > distanceThreshold - 250) {
digitalWrite(2, HIGH);     digitalWrite(3, HIGH);     digitalWrite(4,
LOW);
  }
  if (cm <= distanceThreshold - 250 && cm > distanceThreshold - 350) {
digitalWrite(2, HIGH);     digitalWrite(3, HIGH);     digitalWrite(4,
HIGH);
  }   if (cm <= distanceThreshold -
350) {     digitalWrite(2, HIGH);
digitalWrite(3, HIGH);
digitalWrite(4, HIGH);
  }



  delay(100); // Wait for 100 millisecond(s) }
```

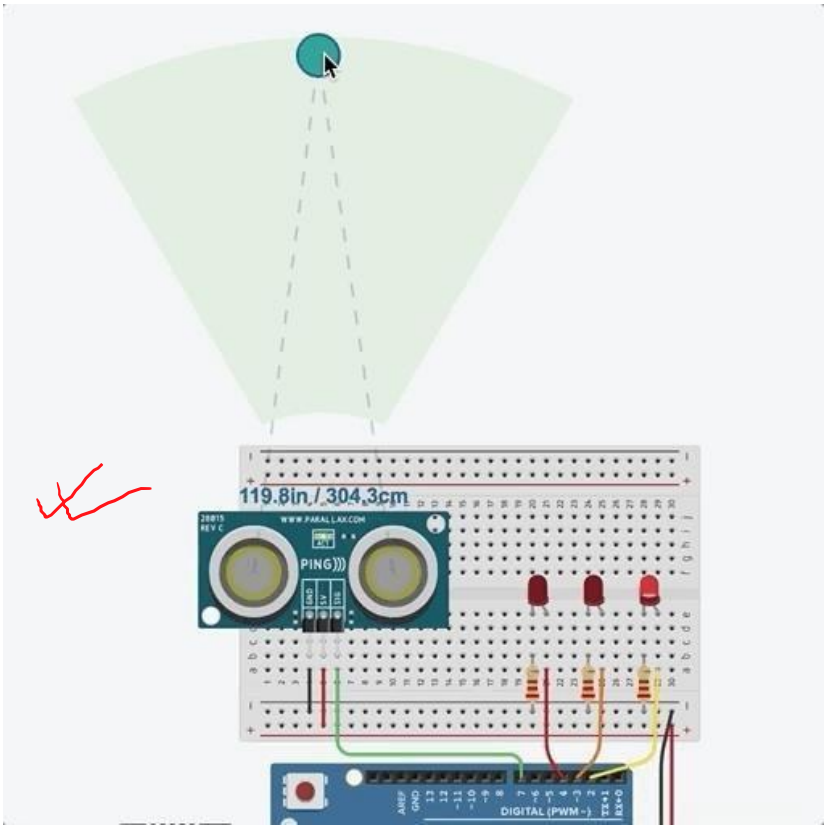
The loop's six if statements evaluate for different ranges of distance between 0 and 350cm, lighting up more LEDs the closer the object.

If you want to see a more obvious change in bar graph lights, you can change the distanceThreshold variable and/or the range that you are looking at by changing the arguments in the if() statements. This is called calibration.

Step-4: Run the Simulation → can use given diagram below

Run the simulation and click on the proximity sensor. This will activate a highlighted area in front of the sensor with a circle "object" inside. You may need to resize the view if the circle is off screen.

Click and drag the "object" circle closer and further away, noticing the changing distance values on screen. More LEDs will light up the closer you get to the sensor.



Congratulations! You have learned to detect distance using an ultrasonic sensor. You also learned about standalone functions in this lesson and used and the serial monitor to track changes inside your Arduino. You could expand this project by making it a proximity alarm by adding a piezo buzzer that turns on when all three LEDs are lit up (closest distance). Consider swapping the distance sensor for a temperature sensor. Or add motors to create a robot with obstacle detection!

Questions for Report Writing:

Include all codes with explanations into lab report by following the writing template mentioned in appendix A of Laboratory Sheet Experiment 6.

Reference(s):

[1] Arduino IDE, https://www.arduino.cc/en/Main/Software accessed on May 3, 2019. [2] Arduino and Proteus Library, https://etechnophiles.com/add-simulate-ultrasonic-sensorproteus-2018-edition/ accessed on May 3, 2019.
[3] Ultrasonic Distance Sensor in Arduino With Tinkercad https://www.instructables.com/id/Ultrasonic-Distance-Sensor-Arduino-Tinkercad/ accessed on May 3, 2019.