



American International University- Bangladesh
Department of Electrical and Electronic Engineering
 EEE 4212: 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.

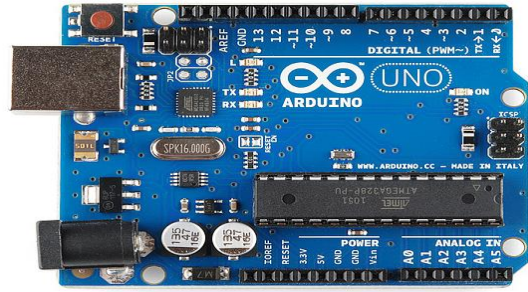


Objectives: In this lab, we will learn –

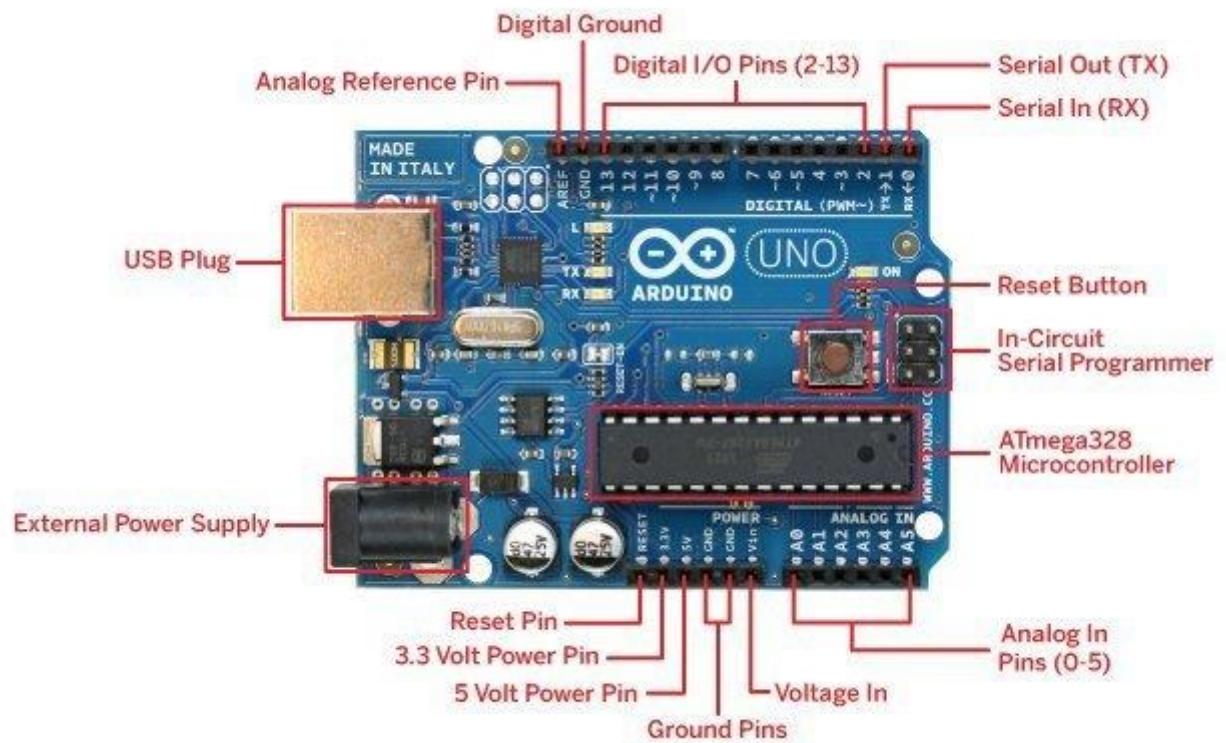
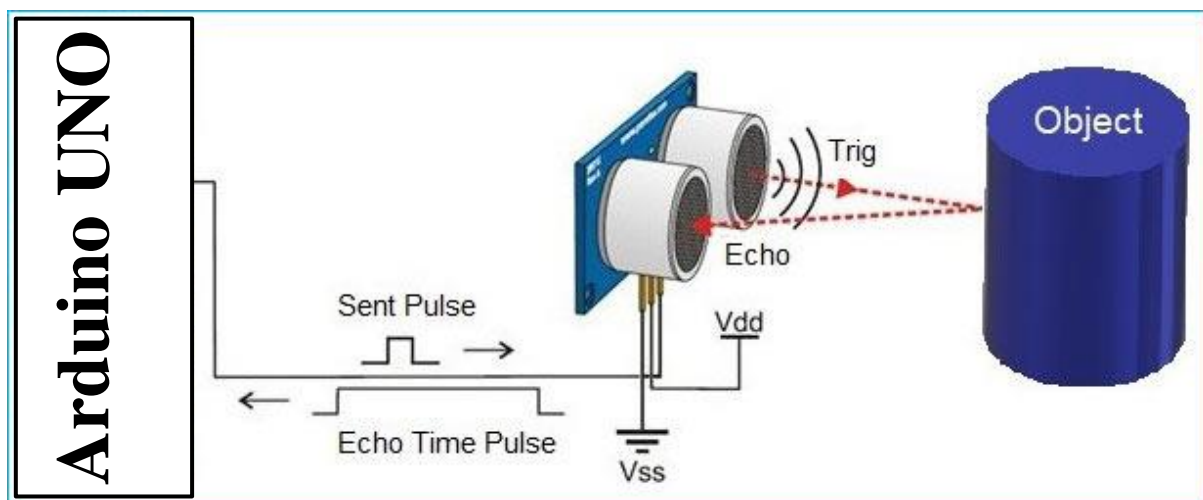
- (i) How to code a simple Obstacle Detection System in Arduino IDE.
- (ii) Implement a simple Obstacle Detection System in Hardware.
- (iii) How to simulate the code in Proteus/Tinkercad and observe the results.

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).

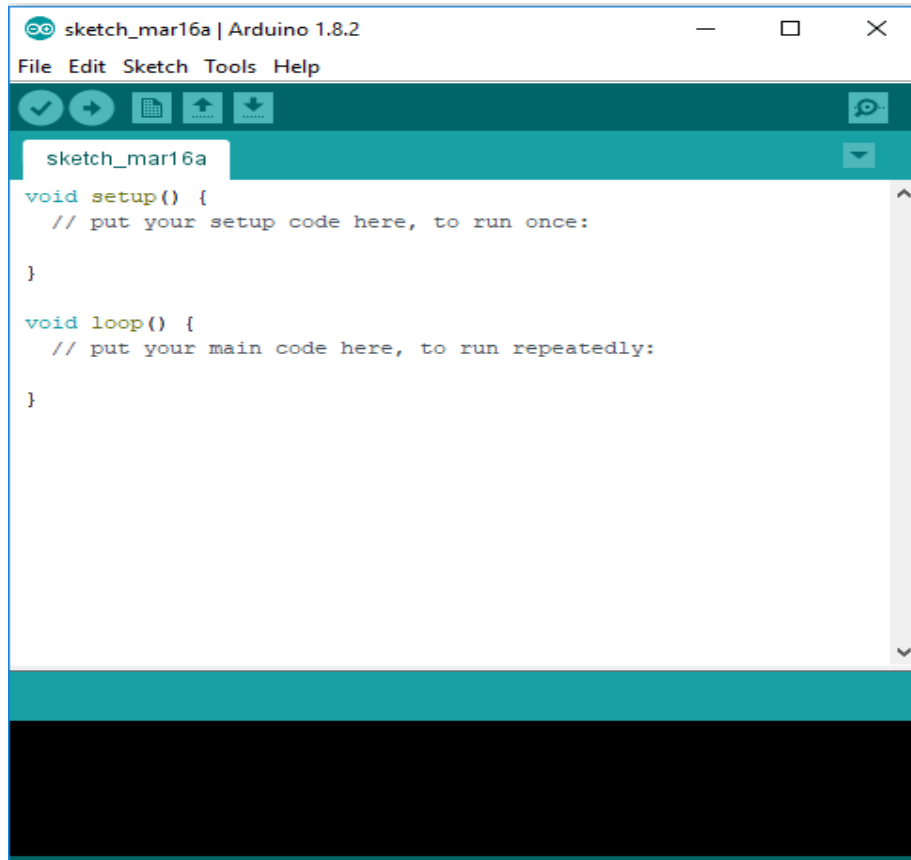
Link: <https://www.arduino.cc/en/Main/Software>

Experimental Procedure:

You have all you need to start your Lab work.

Using Arduino IDE to Write Code:

1. Open the Arduino Uno IDE 1.8.2 and a blank sketch will open. The following window will come up on your PC: -



2. Now write the following code in a blank sketch.

```
const int echoPin = 6; // Echo Pin of Ultrasonic Sensor
const int trigPin = 7; // Trigger Pin of Ultrasonic Sensor
const int LED = 4; // LED at Pin 4

void setup()
{
  Serial.begin(9600); // Starting Serial Communication
  pinMode(trigPin, OUTPUT); // initialising pin 7 as output
  pinMode(echoPin, INPUT); // initialising pin 6 as input
  pinMode(LED, OUTPUT); // initialising pin 4 as output
}

void loop()
{
  long duration, inches, cm;

  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
```

```
digitalWrite(trigPin, HIGH);
delayMicroseconds(10);
```

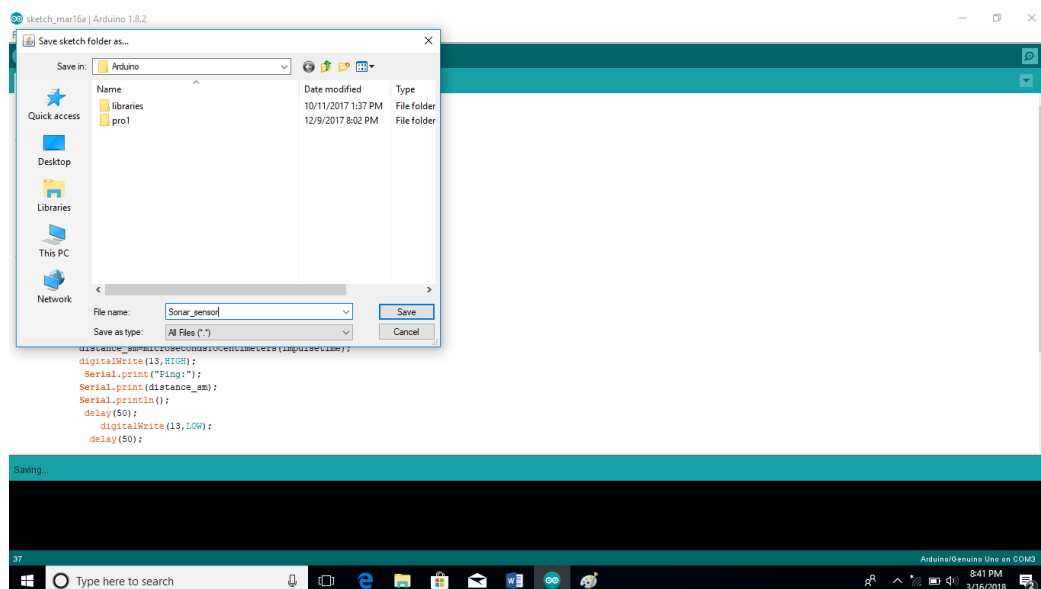
```
digitalWrite(trigPin, LOW);
```

```
duration = pulseIn(echoPin, HIGH); // using pulsin function to determine total time
inches = microsecondsToInches(duration); // calling method
cm = microsecondsToCentimeters(duration); // calling method
if(cm<10)
{ Serial.print(inches);
  Serial.print("in, ");
  Serial.print(cm);
  Serial.print("cm");
  Serial.println();
  digitalWrite(LED, HIGH);
}
else
{ digitalWrite(LED, LOW);
}
delay(100);
}
```

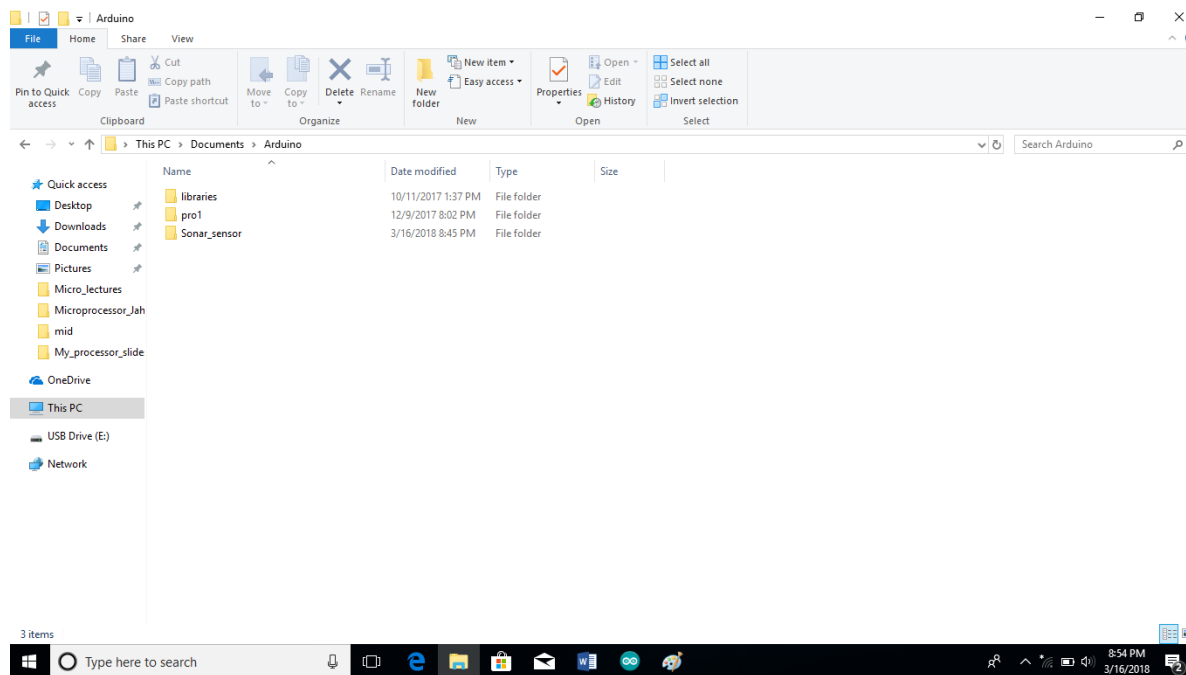
```
long microsecondsToInches(long microseconds) // method to covert microsec to inches
{
  return microseconds / 74 / 2;
}
```

```
long microsecondsToCentimeters(long microseconds) // method to covert microsec to cm
{
  return microseconds / 29 / 2;
}
```

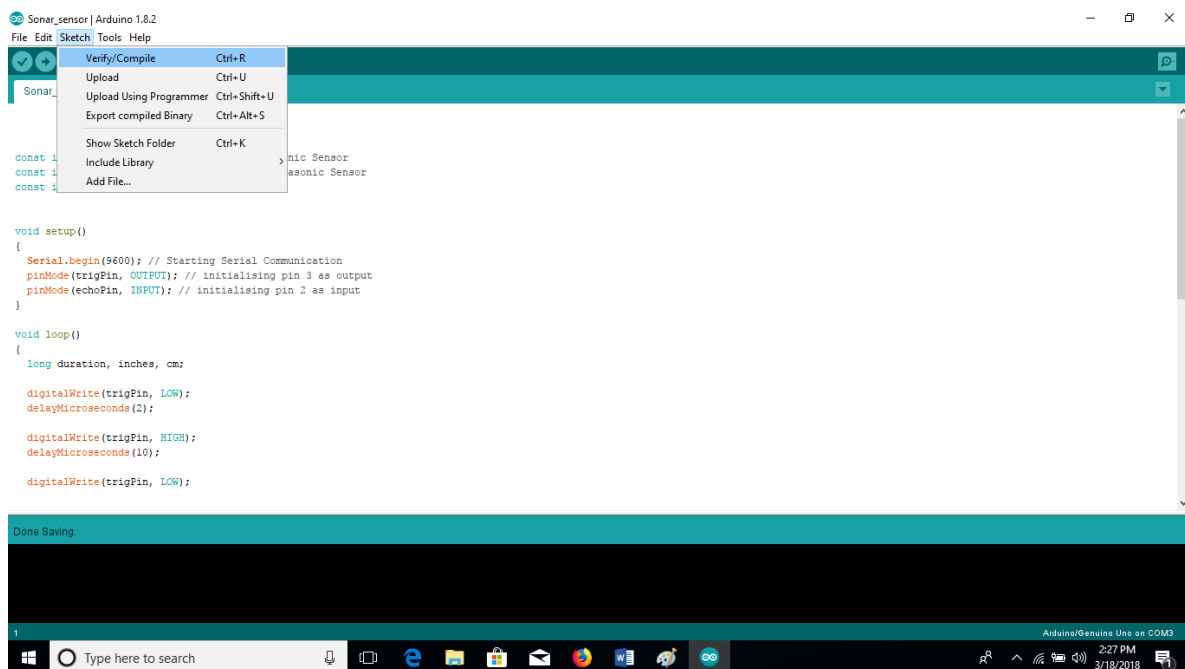
3. After the writing the code you have to save the sketch, go to File->Save As->give a File name(Sonar_sensor)-> Select Save.



N.B. After saving your code a sketch file with the desired file name will be stored in a folder with same file name.

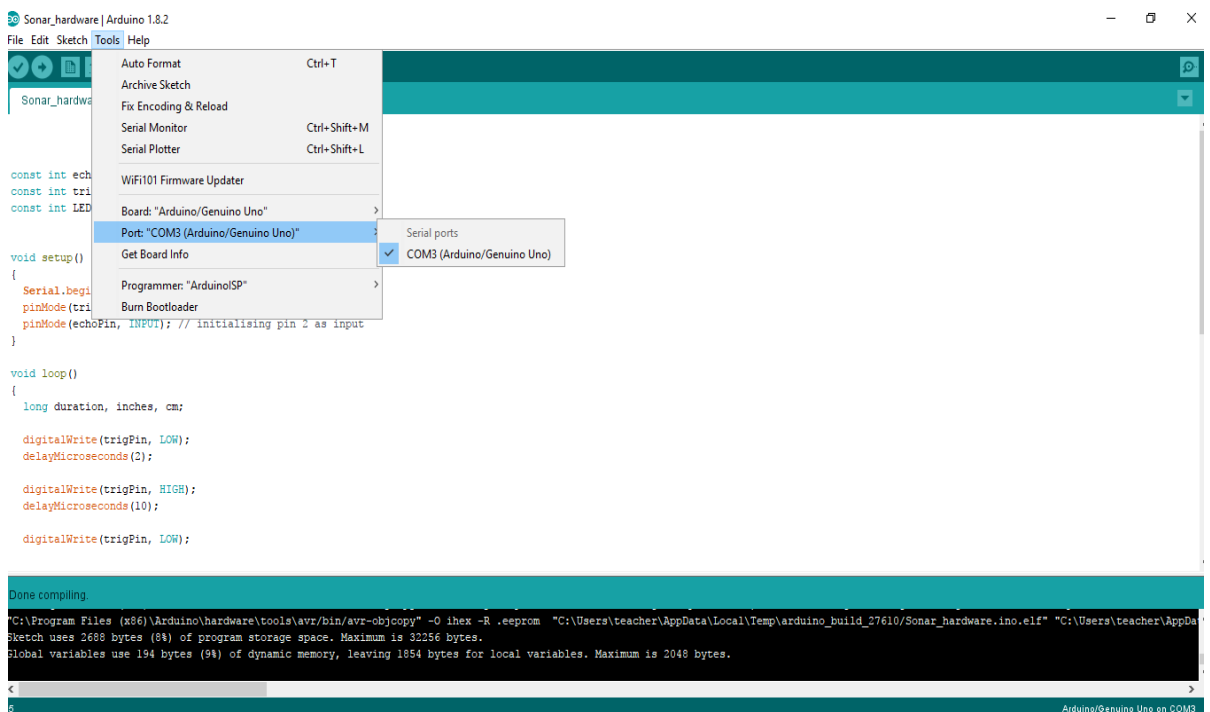
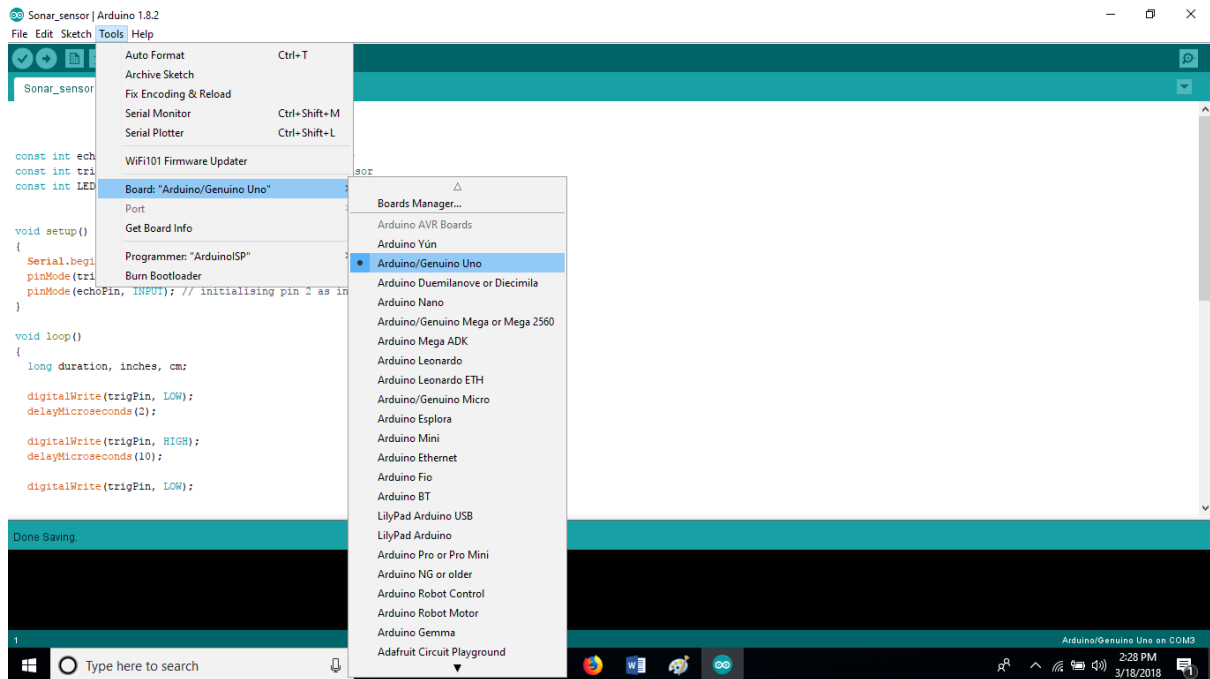


4. Now you need to verify/compile your code to find out and correct the errors, go to Sketch->Verify/Compile.

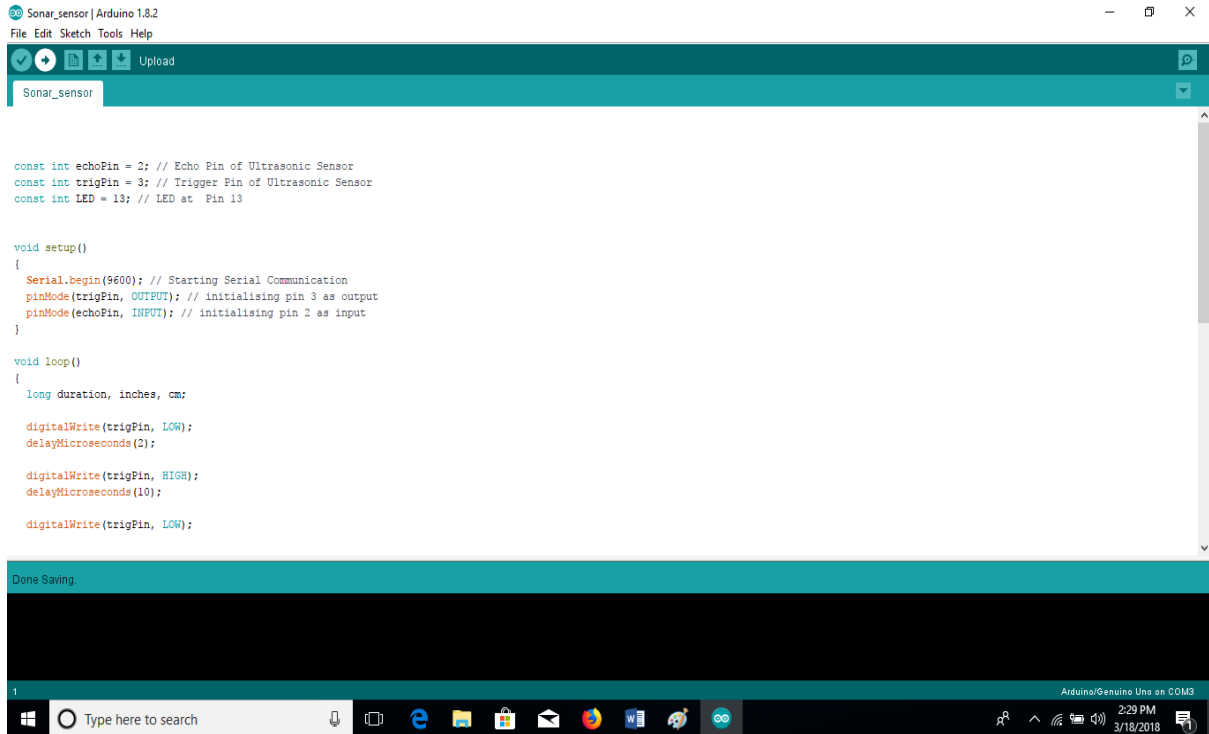


5. After compiling is done you need to upload your code into the Arduino Uno board. To upload the code connect your Arduino Uno R3 board to your PC with USB cable. Before uploading the code select the board type and port at your Arduino IDE, go to

- Tools-> Board:"Arduino/Genuino Uno" -> Arduino/Genuino Uno.
- Tools->Ports-> COMx



After you have selected the board and port select the upload option at the Arduino IDE (see the following figure) to upload the code.



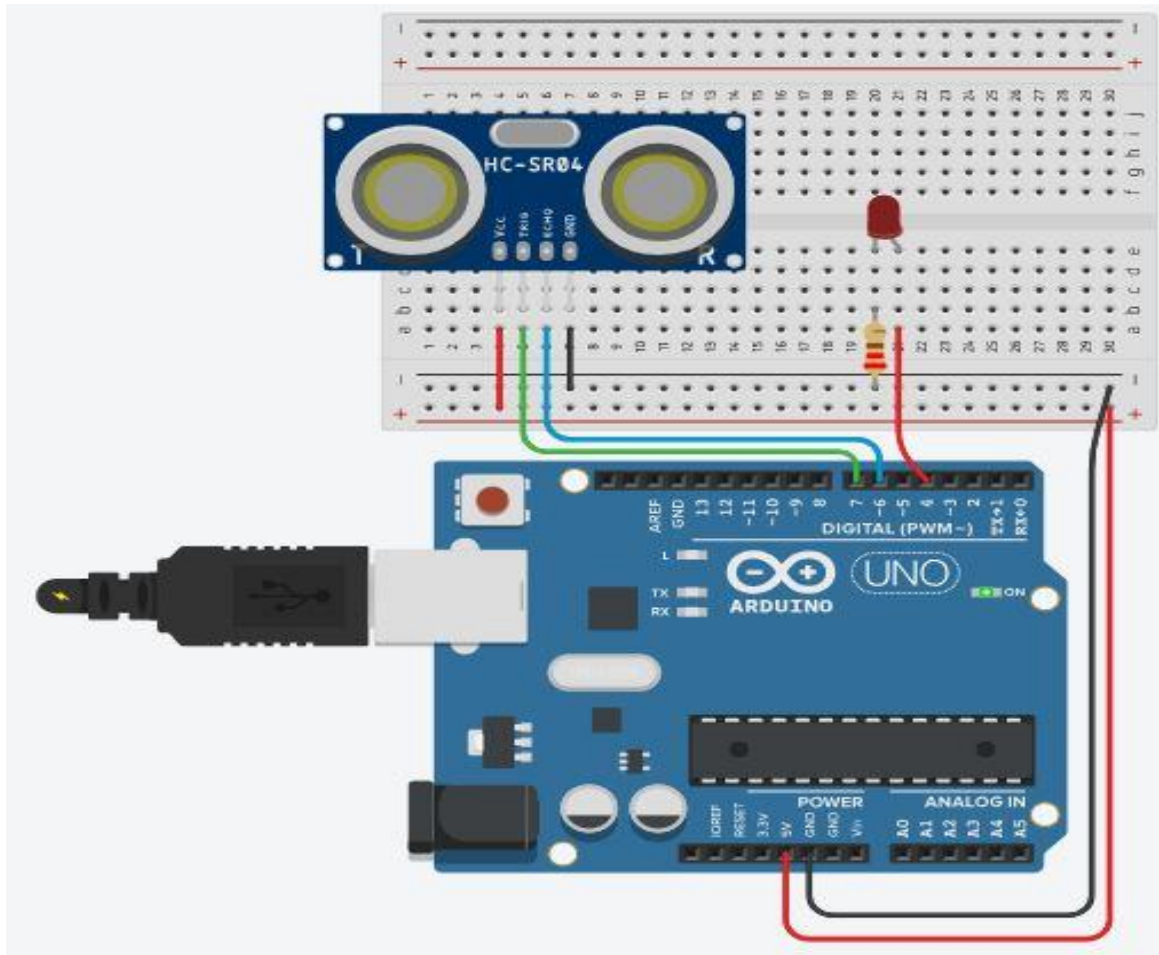
Familiarization with the Arduino Commands:

In this section,

1. We will learn about some common Arduino commands that will help write code.
2. This section also focuses on the standard Library functions associated with the IDE.
 - a) ******pinMode(X, INPUT) or pinMode(X,OUTPUT) ******
this command will configure any pin at the Arduino board as either input/output.
 - b) ******digitalWrite(X, LOW) or digitalWrite(X, HIGH) ******
this command will provide a HIGH/LOW value to any digital output pin at the Arduino board.
 - c) ****** delayMicroseconds(X) ******
this command will provide a delay of “X” microseconds after executing a line of command.
 - d) ****** Serial.print("Text") ******
this command will print a text at the serial monitor of the Arduino IDE

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 (HCSR04) 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 10 μ s 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: -

$$\begin{aligned}\text{distance_cm} &= \text{microseconds} / 29 / 2; \\ \text{distance_in} &= \text{microseconds} / 74 / 2;\end{aligned}$$

If the distance goes below 10 centimeters than the LED will glow, and the distance covered will be shown in the “**Serial Monitor**” of the Arduino IDE.

Simulating the Code in Proteus**Adding Arduino and Ultrasonic Sensor Library to Proteus:**

The Ultrasonic Sensor also called “Ping Sensor” is used to detect the objects in front and measure the distance between sensor and object.

STEP 1: First, download Arduino and Ultrasonic Sensor Library from the reference link.

STEP 2: For Ultrasonic Sensor Library, you will obtain these three files:

- (A) 1.UltrasonicTEP.LIB
- 2.UltrasonicTEP.IDX
- (B) 3.UltraSonicTEP.HEX

For Arduino Library, you will obtain these two files:

- (A) 1. ARDUINO.LIB
- 2. ARDUINO.IDX

STEP 3: Cut .LIB & .IDX files of both Ultrasonic sensor and Arduino and paste them in:

Go to “C:\Program Files (x86)\Labcenter Electronics\Proteus 7 Professional\LIBRARY”

Cut .HEX file of Ultrasonic sensor and paste it in:

Go to “C:\Program Files (x86)\Labcenter Electronics\Proteus 7 Professional\BIN”

STEP 4: Open proteus and click on “pick from libraries”

STEP 5: Search for “ultrasonic”, you will see ultrasonic sensor. Select it and then place it anywhere on the front panel!

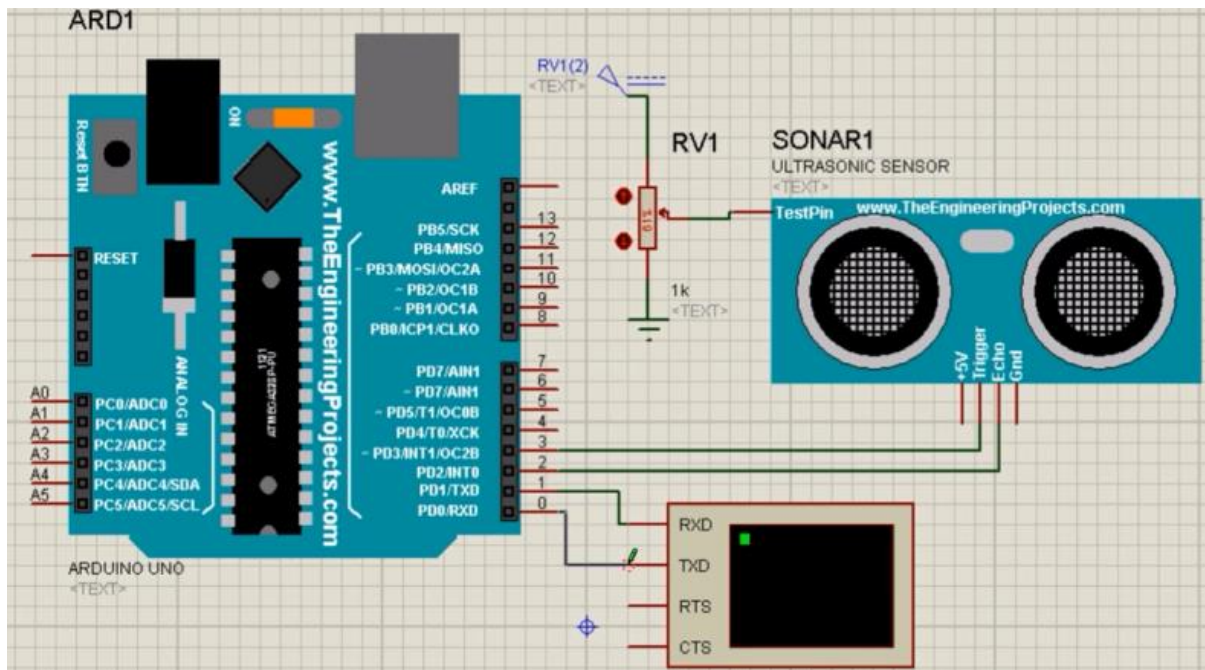
Search for “Arduino”, you will see 6 Arduino boards in result bar. Select Board of your choice and start simulating!

STEP 6: Now you are good to go! You can Start simulating Ultrasonic Sensor right away.

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: Download this “UltraSonic ” program to your Arduino IDE:

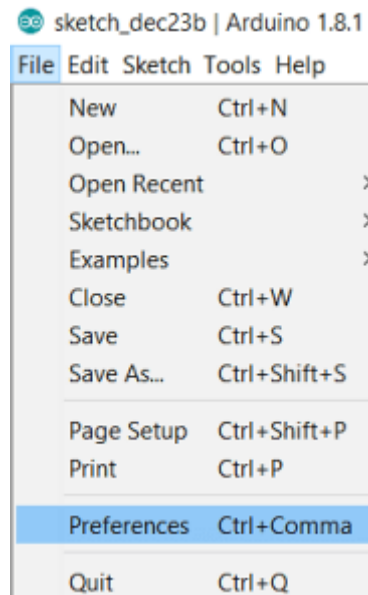
```
const int echoPin = 2; // Echo Pin of Ultrasonic Sensor
const int pingPin = 3; // Trigger Pin of Ultrasonic Sensor
void setup()
{
  Serial.begin(9600); // Starting Serial Communication
  pinMode(pingPin, OUTPUT); // initialising pin 3 as output
  pinMode(echoPin, INPUT); // initialising pin 2 as input
}
void loop()
{
  long duration, inches, cm;
  digitalWrite(pingPin, LOW);
  delayMicroseconds(2);
  digitalWrite(pingPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(pingPin, LOW);
  duration = pulseIn(echoPin, HIGH); // using pulsin function to determine total time
  inches = microsecondsToInches(duration); // calling method
  cm = microsecondsToCentimeters(duration); // calling method
  Serial.print("in, ");
  Serial.print(cm);
  Serial.print("cm");
  Serial.println();
  delay(100);
}
long microsecondsToInches(long microseconds) // method to covert microsec to inches
```

```

{
return microseconds / 74 / 2;
}
long microsecondsToCentimeters(long microseconds) // method to covert microsec to cm
{
return microseconds / 29 / 2;
}

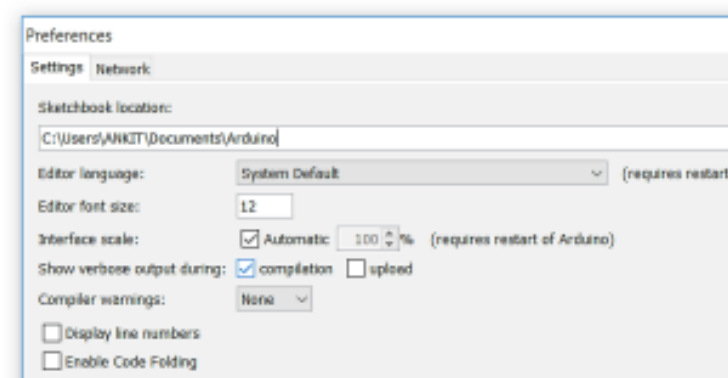
```

STEP 3: Go to preferences:



STEP 4: Check “Compilation” Check Box

This is an important step; otherwise HEX file won't be generated.



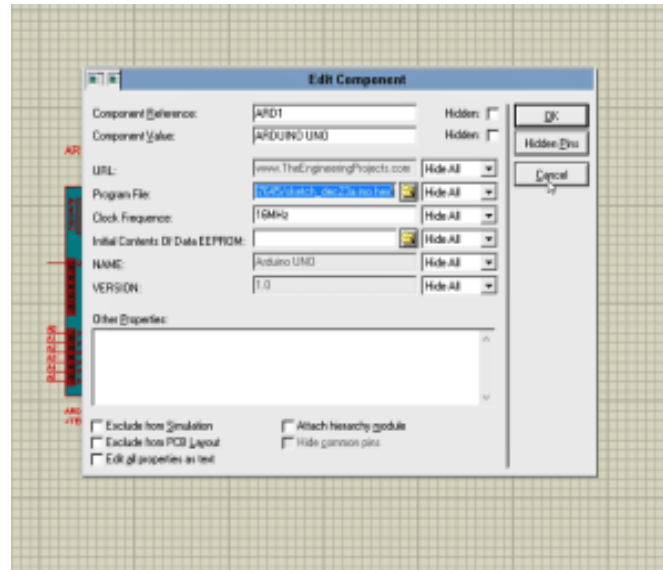
STEP 5: Click on verify (compile)



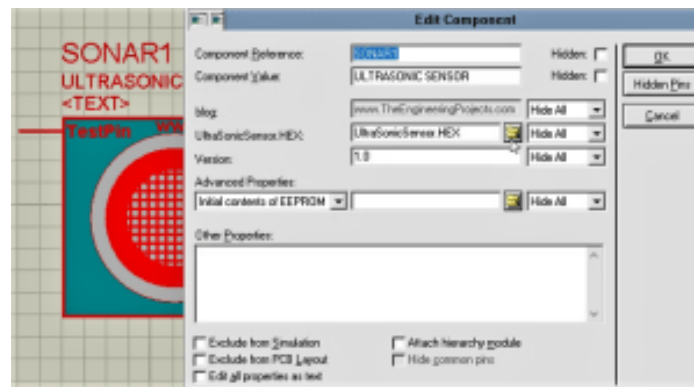
STEP 6: After compilation is complete copy HEX file path from the bottom corner:

```
Done compiling
"C:\Program Files (x86)\Arduino\hardware\tools\avr\bin\avr-objcopy" -O ihex -R .eeprom "C:\Users\AMRIT\AppData\Local\Temp\arduino_build_821076\sketch_dec23b.ino.celf" "C:\Users\AMRIT\AppData\Local\Temp\arduino_build_821076\sketch_dec23b.ino.hex"
Sketch uses 2404 bytes (54% of program storage space. Maximum is 32256 bytes.
Global variables use 206 bytes (104% of dynamic memory, leaving 1842 bytes for local variables. Maximum is 2048 bytes.
```

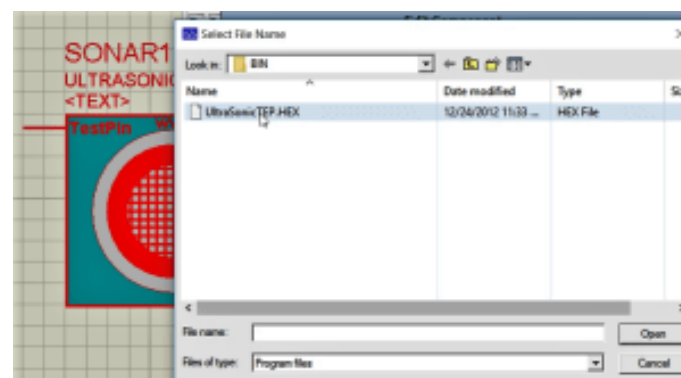
STEP 7: Open Proteus and double click on Arduino. Paste the file path and click ok



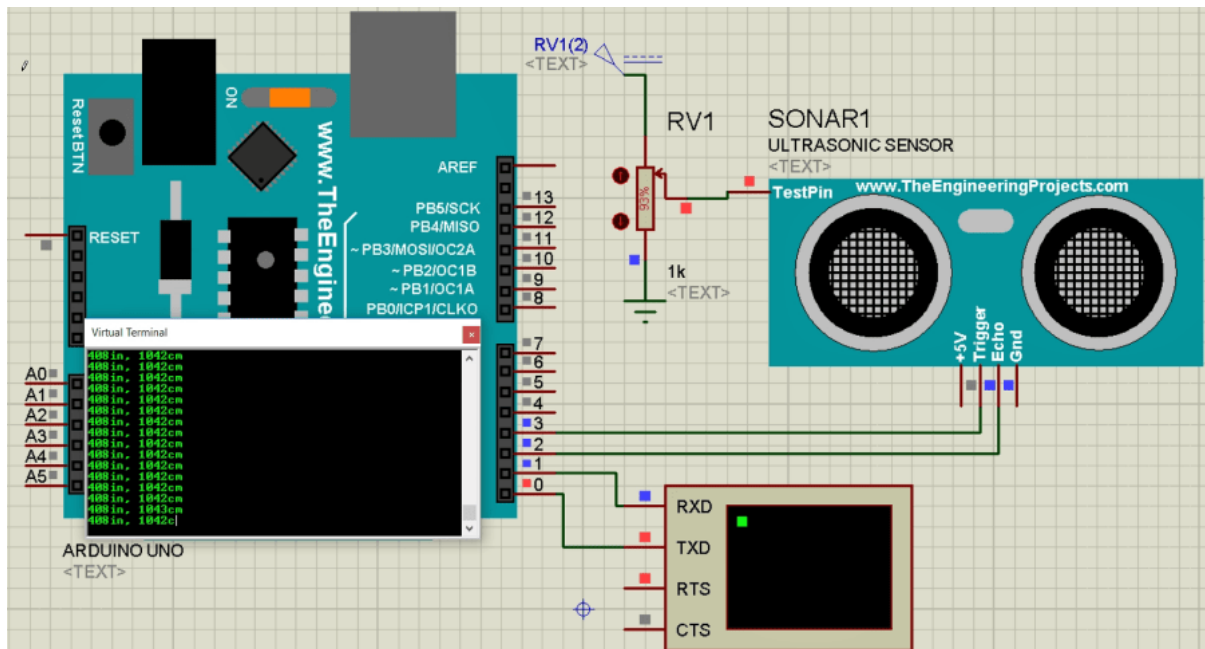
STEP 8: Double-click on sensor and then click on Ultrasonicsensor.HEX as shown in figure.



STEP 9: Select “UltraSonicTEP.HEX” file from BIN and hit enter as shown in figure.



STEP 10: Run the simulation! Change potentiometer's wiper terminal position to test the distance value on virtual terminal.



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.

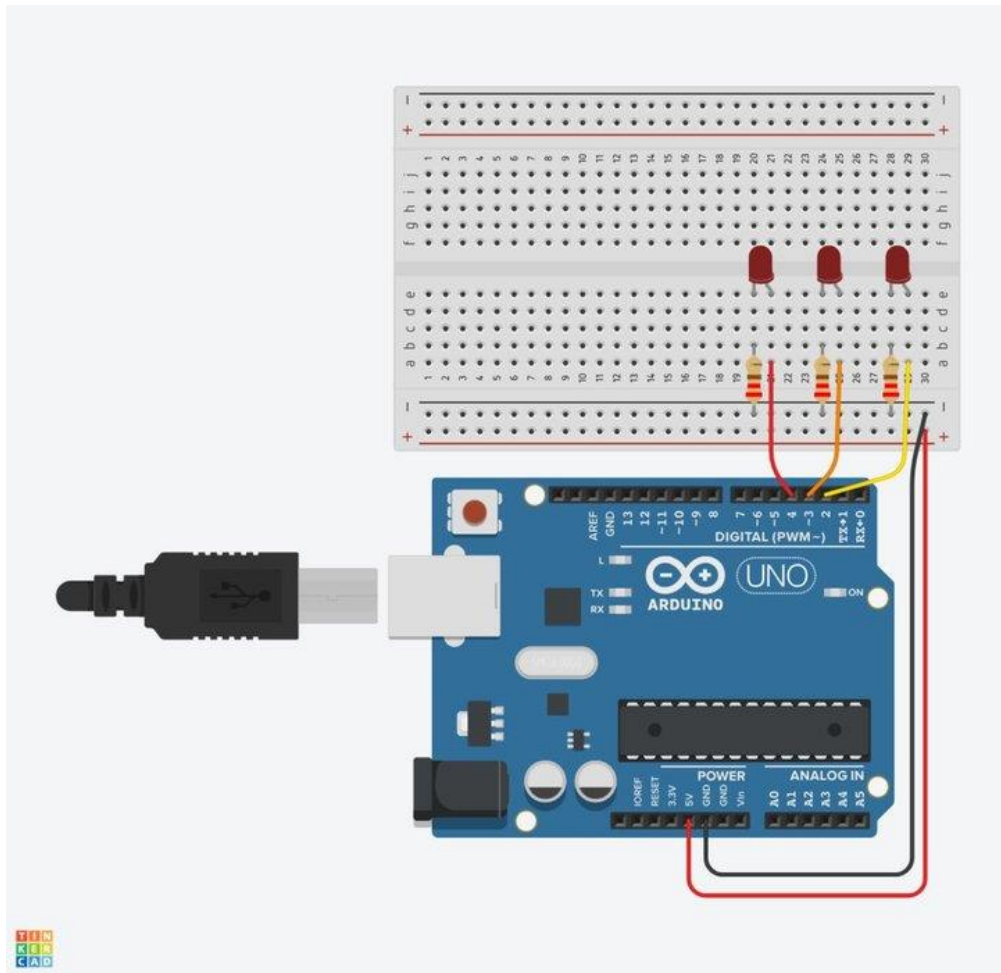
Go to tinkercad.com and set up a free account. At the top of your front page you can click Learn to follow walk-through Lessons that will teach you basic navigation and modeling techniques. From the Gallery you can browse models that you can Copy and Tinker to remix and modify, or just use the Search 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.

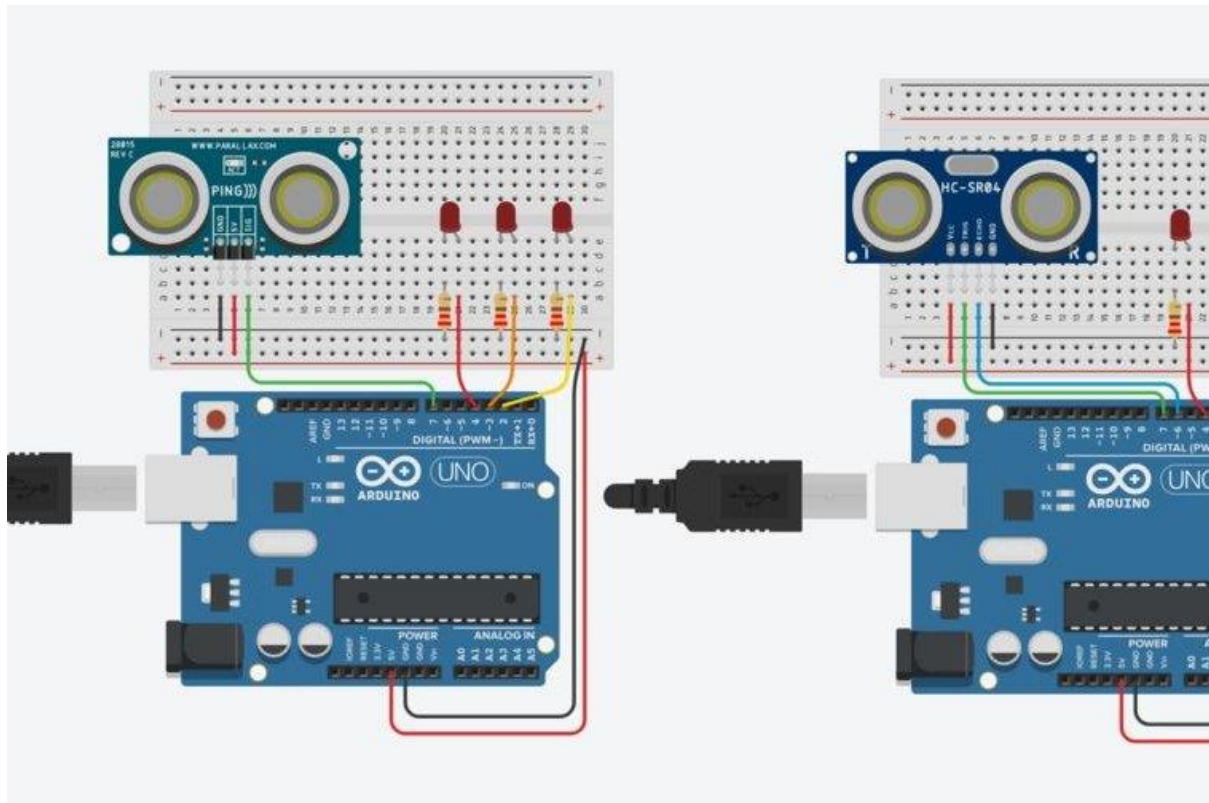
Drag an Arduino Uno and breadboard from the components panel to the workplane, next to the existing circuit.

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 `readUltrasonicDistance()` 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)
}
```

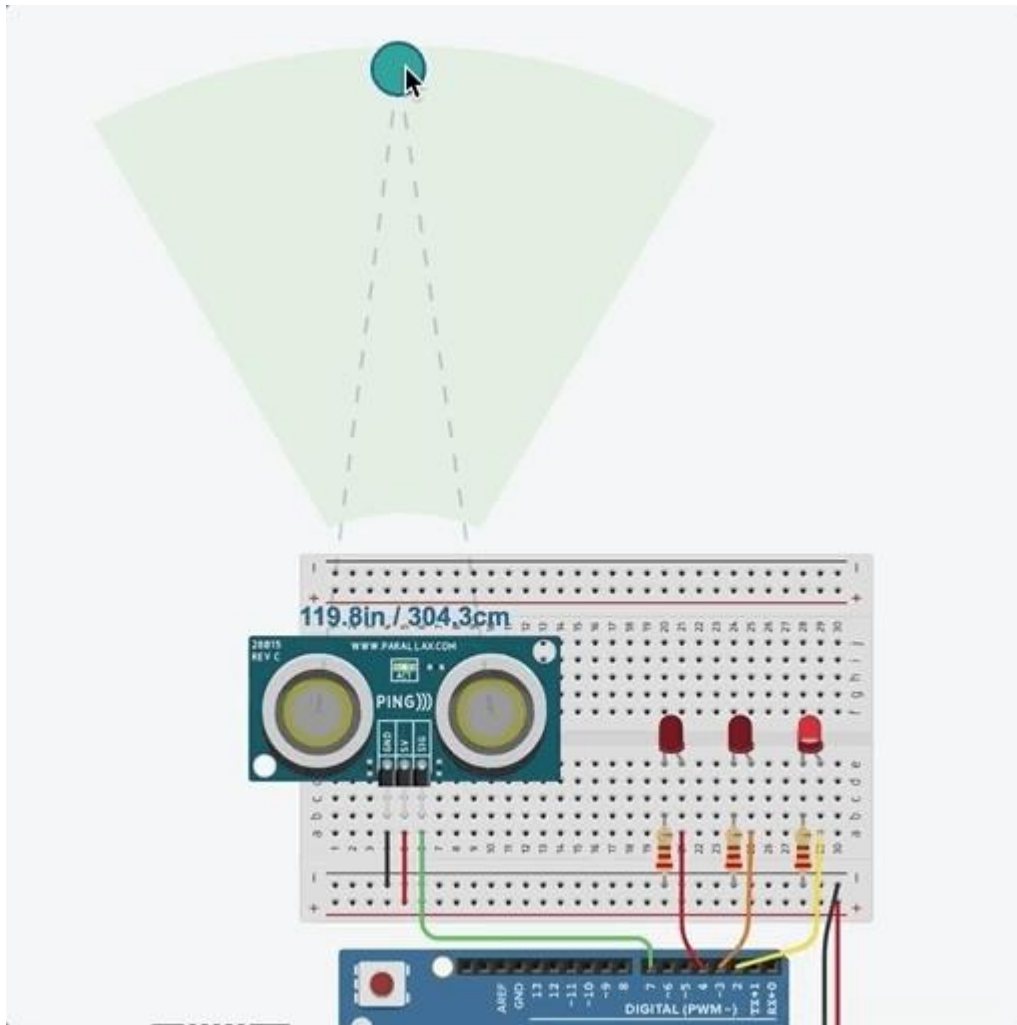
The loop's six if statements evaluate for different ranges of distance between 0 and 350cm, lighting up more LEDs the closer the object.

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

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 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 1.

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-sensor-proteus-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.