Skills Workshop 4: Introduction to Coding

Objective

By the end of this workshop, students will be able to:

1. Write a program in the Arduino IDE that records data from a sensor
2. Extract, tabulate, and visualize the recorded data from a sensor

Background Information

Coding is the process of writing instructions for a computer to interpret, resulting in a computer program which allows the computer to execute certain tasks. In engineering disciplines, coding facilitates the interactions between software and hardware systems. Applications include consumer electronics, surgical instruments, and aerospace control surfaces. Coding is used in all professional fields. This lab will serve as an introduction to writing code for an Arduino UNO microcontroller.

Arduino Integrated Development Environment

The Arduino Integrated Development Environment (IDE) is a program that can be used to edit, compile, and upload code to a supported microcontroller.
Programs written in Arduino IDE are called sketches. A basic sketch can be broken up into three different areas: global, setup, and loop. These areas are shown in Figure 1.

- **Global:** Contains constants and imported libraries
- **Setup:** Functions that run once at the start of the program. Setup functions often are used to activate pins and sensors in the program
- **Loop:** Functions that run continuously after the Setup functions. Code in the Loop Area will continue to run while the Arduino is powered. In many professional environments, the Loop will run for the duration of the data collection period. Functions often used in most of the program to read sensors and switch pins HIGH to LOW.

Figure 1 shows a screenshot of the program's interface and how this sketch interacts with the Arduino.

- **Verify:** Checks code for errors and points those errors
- **Upload:** Verifies code and uploads it to the Arduino board
- **Console:** Shows errors found in the hardware
- **Serial Monitor:** Sends and receives messages to and from the board

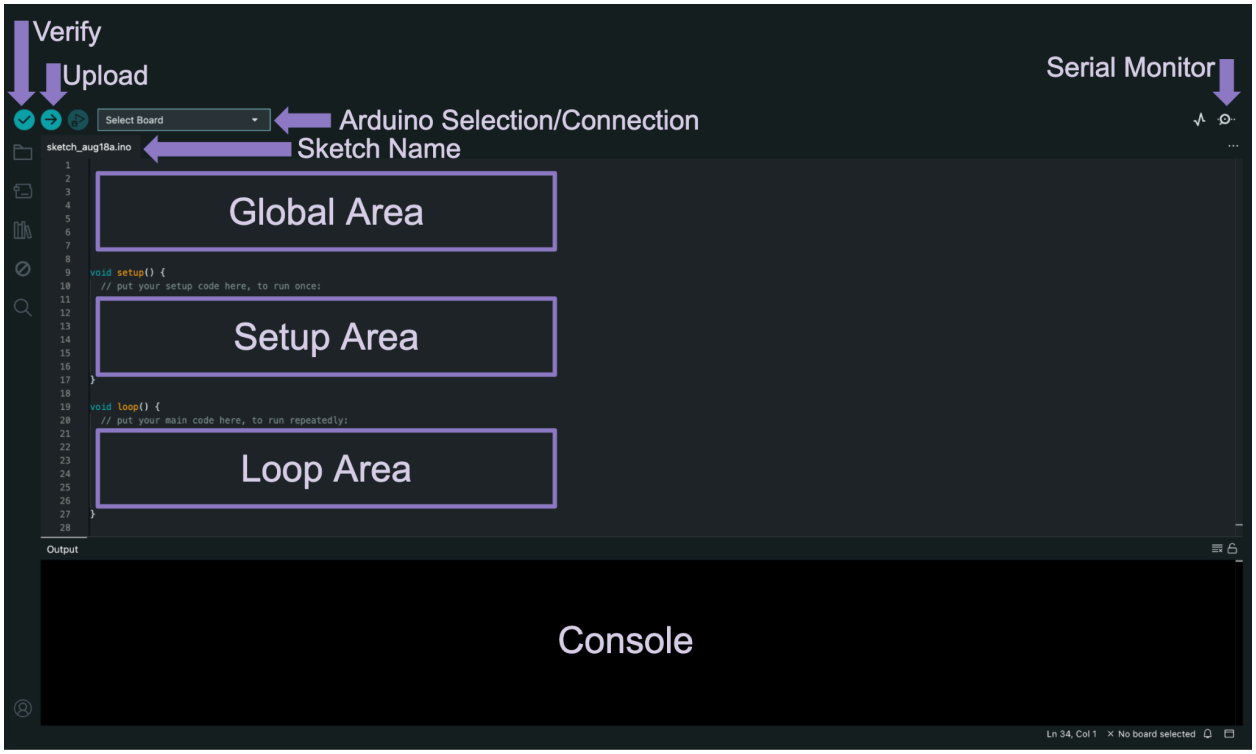

Figure 1: Arduino IDE Components

Variables, Loops, and Conditions

The Arduino programming language is based on C/C++, but it is designed to be simpler and easier to learn. The most intuitive way to think about programming is like building with LEGO blocks: certain rules must be followed and different building blocks can be used to build bigger parts.

Syntax:

- Every line must either end with a semicolon ; unless it is a conditional, loop, or function
- Comments start with a //
    - Comments are text that the program ignores
    - Used to label and explain code
    - These are essential in coding to explain your thought process to other users who may interact with the program

Datatypes:

Datatypes are the different kinds of data values that can be used, manipulated and stored using C++. Table 1 includes the most basic and widely used data types.

Table 1: Common Datatypes in Code

Datatype	What it stores	Default value	Notes
Boolean	A true value (1, TRUE, HIGH) or a false value (0, FALSE, LOW)	0, FALSE, LOW	-
int	An integer number (-5, 15, 1047, etc.)	0	Can be positive or negative
double	A decimal number (-0.5, 123.77, etc.)	0	Can be positive or negative
char	A single character ('c', 'A', '5', '?', etc.)	Indeterminate	Must be enclosed in single quotes
string	A sequence of characters ("Hello World!", "10", "157+5", etc.)	Empty ("")	Must be enclosed in double quotes

Operators

Operators perform operations on variables and constants. The results of these operations are usually stored in a variable. Table 2 includes common operators.

Table 2: Common Operators in Code

Operator	What it does	Notes
=	Assigns a value to a variable	Mathematical symbol
+	Adds two or more values	Mathematical symbol
-	Subtracts two or more values	Mathematical symbol
*	Multiplies two or more values	Mathematical symbol

/	Divides two or more values	Mathematical symbol
++	Increment by 1	Usually used in loops
--	Decrement by 1	Usually used in loops
==	Checks if two values are equal	Usually used in conditionals
!=	Checks if two values are not equal	Usually used in conditionals
> or <	Less than/greater than comparison	Usually used in conditionals
<= or >=	Less than/greater than or equal to comparison	Usually used in conditionals
&& or ||	**Boolean AND** or **Boolean OR** used to cascade multiple Boolean operations	Usually used in conditionals

Constants and Variables

Constants and variables hold data according to their datatype. They need to be given a name so they can be referenced later. Constants hold data that will not change while a program is running. Constants usually contain information like pin numbers or sensor threshold values. Variables contain data that will change while a program is running. Variables usually contain sensor values and other values that will interact mathematically within the program. Figure 2 shows an example of how to create different constants and variables.

```
// Examples of Constants

const int ledPin = 5;
const string greetingMsg = "Hello World!"

//Examples of Variables

int sensorValue = 0;
char myStringVariable = "General Engineering!";
```

Figure 2: Examples of Different Constants and Variables in Arduino IDE

Conditional Statements

Conditional statements run code enclosed by their curly brackets when a condition is met (Figure 3).

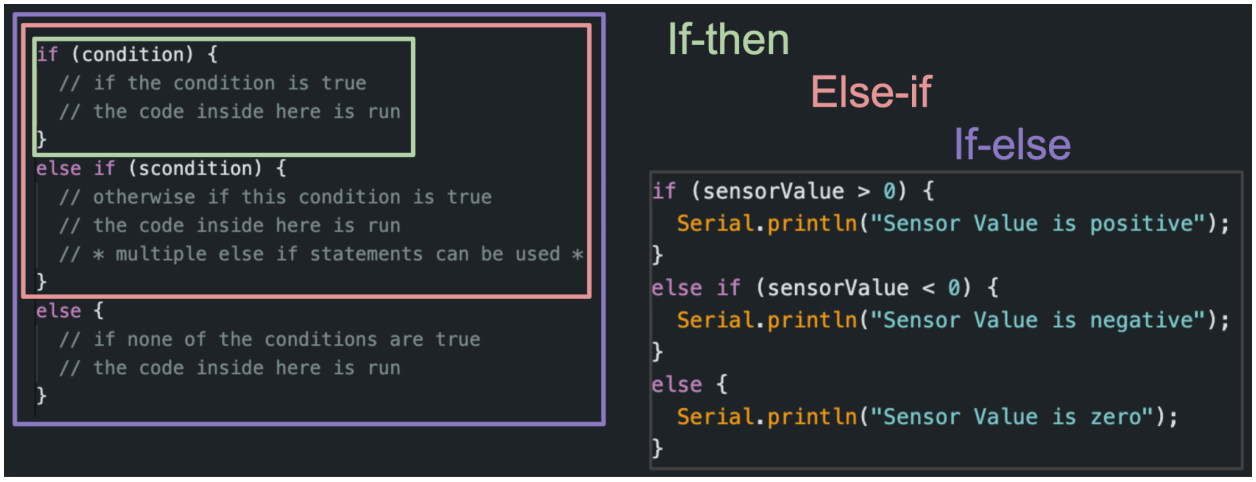

Figure 3: Example Conditional Statement in Arduino IDE

Loops

Loops are control structures that allow a set of instructions to be repeated, either a specific number of times or until a certain condition is met. They are essential for condensing repetitive tasks and reducing code redundancy. There are two primary types of loops used in this lab: **while-loops** and **for-loops**.

While-loops are used when an action needs to continue until a specific condition is no longer true. The loop keeps running as long as the condition remains true. **For-loops** are used when a task needs to be repeated a discrete number of times. Although their syntax may seem complex at first, most for-loops follow a standard structure. Inside the parentheses, the first part initializes a counting variable (typically `i` for index), the second part defines the condition under which the loop continues, and the third part updates the counting variable (usually by incrementing or decrementing it).

```
while (buttonState == HIGH) {
  delay(1000);
}

for (int i=0; i > 10; i++) {
  Serial.println(i); // This will print 0 to 9 in the serial monitor
}

for (int i=13; i > 0; i--) {
  Serial.println(i); // This will print 13 to 1 in the serial monitor
}
```

Figure 4: Example **While** and **For** Loops

Libraries

In programming, a library refers to a collection of pre-written code that serves a specified purpose, reducing the amount of code a programmer needs to write. For example, the Math library provides usable functions that can perform math operations such as calculating the square root, or trigonometry calculations). In Arduino programming, libraries are commonly used in conjunction with different components such as motors, screens, and sensors, and allows programmers to have an easier time working with them.

Commonly Used Arduino Functions

Table 3: Common Functions in Arduino Code

Function	What it does
pinMode(pin,mode)	Sets a pin as an INPUT or OUTPUT
digitalWrite(pin, value)	Sets a digital output pin to HIGH or LOW
digitalRead(pin)	Reads a digital input pin as HIGH or LOW
analogWrite(pin, value)	Sets an analog output pin to a value 0-255
analogRead(pin)	Reads an analog output pin as a value 0-255
delay(milliseconds)	Pauses the program for a certain amount of time
delayMicroseconds(microseconds)	Pauses the program for a certain amount of time (in microseconds)
Serial.begin(baud rate)	Starts serial communication at the given baud rate

Serial.println(value)	Prints the value (variable) to the Serial Monitor on a new line
pulseIn(pin, value)	Measures the length (in microseconds) of a HIGH or LOW pulse on a pin.
map(value, fromLow, fromHigh, toLow, toHigh)	Re-maps a number from one range to another

Code Flowchart

A critical preliminary step in coding is being able to map out what your program or circuit will do before you begin the actual coding process. One effective method of doing so is a code flowchart, which is a visual depiction of the building blocks of a program. A code flowchart can help outline the essential functions of a program and illustrate logical relationships between the steps of the program, such as loops, conditionals, and mathematical operations. This step is important not only to simplify the code into more manageable segments, but allows for a baseline to refer to when writing code. A code flowchart represents the actions that a program will iterate through, and should not refer to how the code itself is written.

The flowchart below (Figure 5) depicts a self watering pot with bluetooth reminders to refill water and fertilizer.

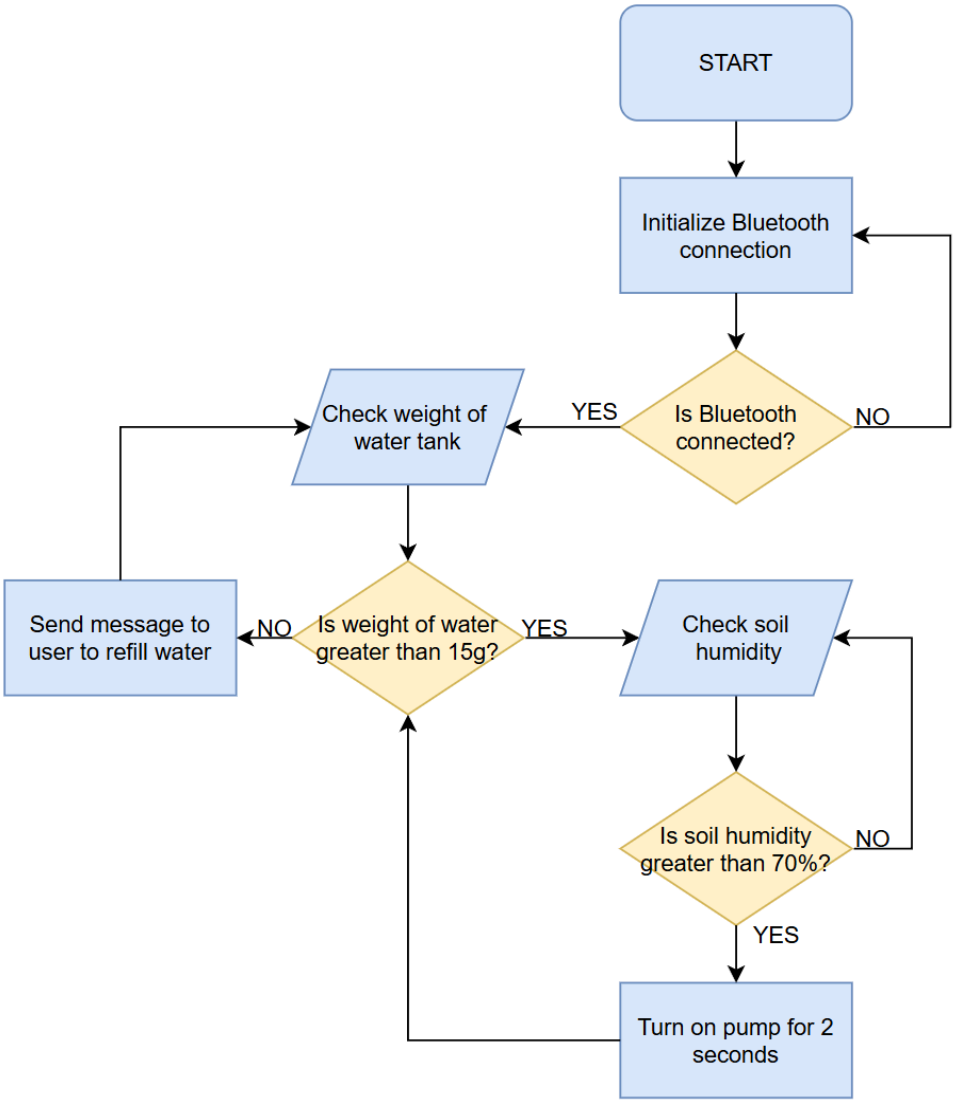

Figure 5: Example Code Flowchart

Hardware

This lab uses a set of physical components commonly found in embedded systems to simulate a real-world data collection and feedback process. Each component plays a specific role in the system and understanding how they work individually helps explain how they function together.

Ultrasonic Sensor

An ultrasonic sensor (HC-SR04) measures the distance between itself and a nearby object by sending out pulses through two ultrasonic transducers. One is a transmitter that sends out ultrasonic pulses and the other is a receiver that detects reflected waves. The pinout configuration of the sensor can be seen in Figure 6.

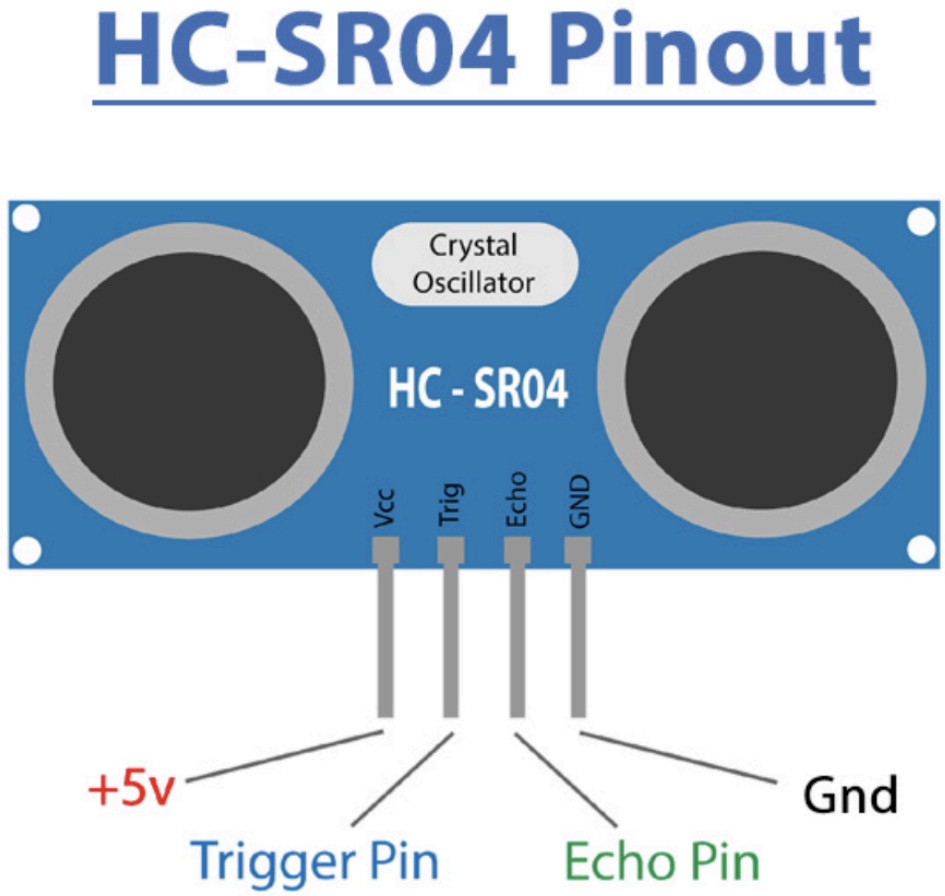

Figure 6: Pinout of the HC-SR04 courtesy of How to Mechatronics

The sensor has four pins: **VCC** and **GND** connect to **5.00 V** and **GND** and **Trig** and **Echo** go to any digital pin on the Arduino. The **Trig** pin sends out a short burst of high frequency ultrasonic pulses and when those pulses hit an object they reflect back to the **Echo** pin. The time between the trigger and echo is used to measure the distance using the speed of sound.

The following formula is used to calculate distance in centimeters:

$$Distance = (Speed(cm/\mu s) * Time(\mu s)) / 2$$

Dividing by two accounts for the time it takes for the sound to travel to an object and back. *Speed* is the speed of sound, 340.00 m/s, which is converted to 0.034 cm/µs, since the *Time,* or the duration of a pulse measured by the ultrasonic sensor, is measured in microseconds (µs).

Liquid Crystal Display (LCD)

The LCD screen is used to display real-time information about a system, in this case, the duration measured by the ultrasonic sensor. This lab uses a 16x2 character LCD screen with an I2C interface, which simplifies wiring by reducing the number of required connections to only four pins (Figure 7).

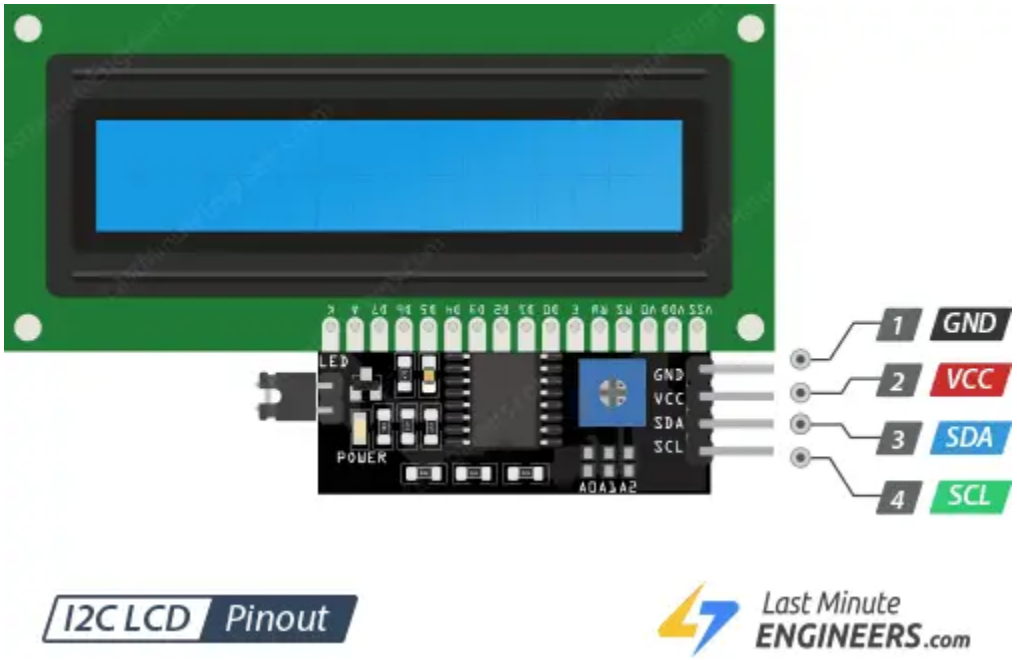

Figure 7: LCD I2C Pinout Diagram

VCC and **GND** connect to **5V** and **GND**. **SDA** and **SCL** are connected to analog pins **A4** and **A5**, and represent special designated pins on the Arduino – Serial Data and Serial Clock. The **SDA** line carries the data being transmitted, while the **SCL** line provides the timing signal that keeps both the microcontroller and the LCD synchronized during communication. These designated I2C pins must be connected correctly for the screen to function. Once connected, the screen can be initialized using the **LiquidCrystal_I2C** library.

Table 4 shows common functions provided by the library for the LCD to function properly.

Table 4: Commonly Used Functions from LCD I2C Library

Function	What it does
init()	Initializes the LCD screen. Must be called before using other functions
backlight()	Turns on the LCD backlight
noBacklight()	Turns off the LCD backlight
setCursor(col, row)	Sets the position of the cursor (column, row) for printing text. Columns start at 0
print("text")	Prints the text at the current cursor position. Accepts strings, numbers, or variables
clear()	Clears all text from the display
nodisplay()	Turns off the display without losing contents

Light Emitting Diode (LED) and Pulse Width Modulation

An LED is commonly used to provide real-time visual feedback for different systems. Since digital pins can only support two states, ON or OFF, brightness can be controlled by analog pins with Pulse Width Modulation (PWM) – a method that rapidly switches the pin on and off to simulate varying light levels. The **analogWrite()** function accepts values between 0 and 255, where 0 represents a 0% duty cycle, and 255 represents a 100% duty cycle. The **Duty Cycle** refers to the percentage of time a digital pin stays on during one complete cycle, where higher values result in the signal being ON for a greater portion of time. This can result in effects such as a brighter LED, or lower duty cycles making the LED appear dimmer. To translate distance values to brightness levels, the **map()** function is used to scale readings into a range appropriate for PWM output. The map() function supports the following format:

map(value, fromLow, fromHigh, toLow, toHigh)

Where *value* is the number to be converted, *fromLow* and *fromHigh* define the original range of values and *toLow* and *toHigh* represent the new range to which the value should be mapped.

Materials

- Arduino
- Arduino USB Connector
- LCD
- LED
- 220 Ω Resistor
- Ultrasonic Sensor
- Wires
- Target

Activity

This workshop lab simulates a quality assurance process at a hardware manufacturing company, where the objective is to evaluate the performance of an ultrasonic distance sensor. The activity involves programming a microcontroller to collect pulse duration measurements from the sensor and determining the speed of sound to compare to the known speed of sound.

The system under test consists of three main components:

- An **ultrasonic sensor** that measures the distance between the sensor and a nearby object.
- An **LED** that varies in brightness depending on the measured distance, offering visual feedback.
- An **LCD screen** that displays real-time values for both distance and LED brightness.

The procedure begins with designing the code through creating a coding flowchart, wiring the components, and writing code to initialize and control each part of the system. Once operational, the setup collects data over a fixed time period. These readings are imported into Excel via a data streamer, and analyzed to evaluate the relationship between measured speed of sound and known speed of sound.

This workshop lab introduces key concepts in coding, embedded systems, and testing, including microcontroller programming using Arduino, interfacing with digital sensors, real-time data display, and basic data analysis. It emphasizes how software can be used to interpret sensor input and generate meaningful output, as well as how logical structures in code relate to system behavior and testing protocols.

Procedure

Part 1: Liquid Crystal Display Exercise

1. Wire the circuit according to the wiring diagram below in Figure 8.

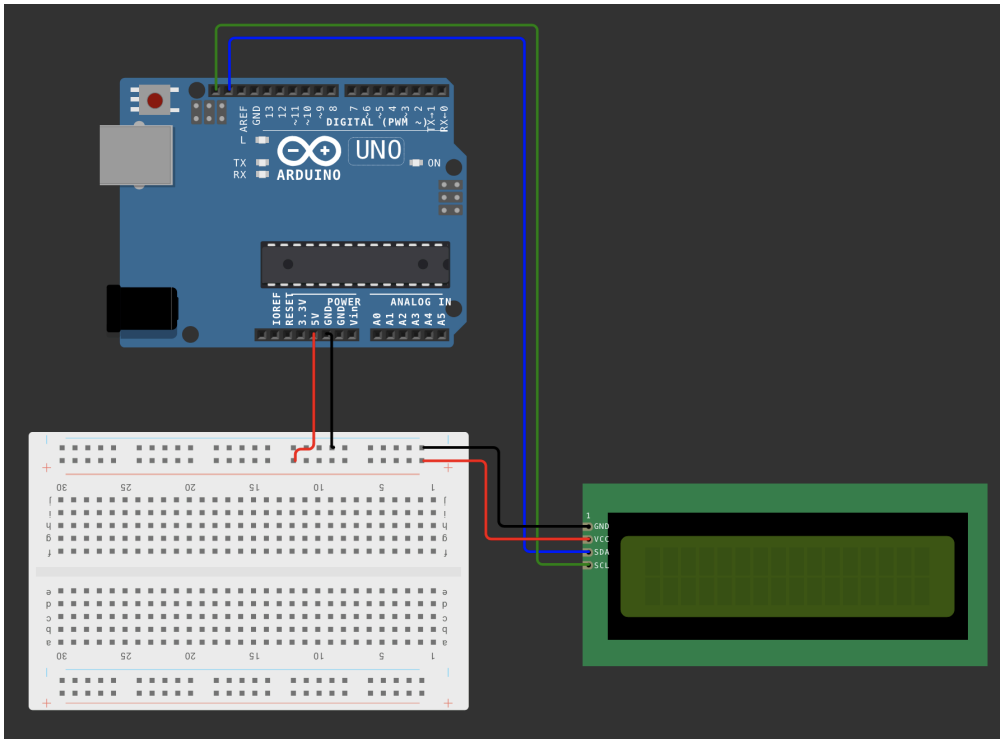

Figure 8: LCD and Arduino Circuit Diagram (https://wokwi.com/)

*** Checkpoint: TA must approve the circuit before moving on to Step 2 ***

2. Plug the Arduino into the monitor using the USB connector provided. The LCD should light up briefly once connected.
3. Launch the Arduino IDE software and start a new sketch. **Important:** for safety purposes, make sure to disconnect the Arduino from any power source whenever wiring new components to the Arduino
4. Download the **LiquidCrystal_I2C** zip folder **(Do not extract the file)**. Click **Sketch**, **Include Library**, and **Add .ZIP Library** (Figure 9). Select the LiquidCrystal_I2C .ZIP file and accept.

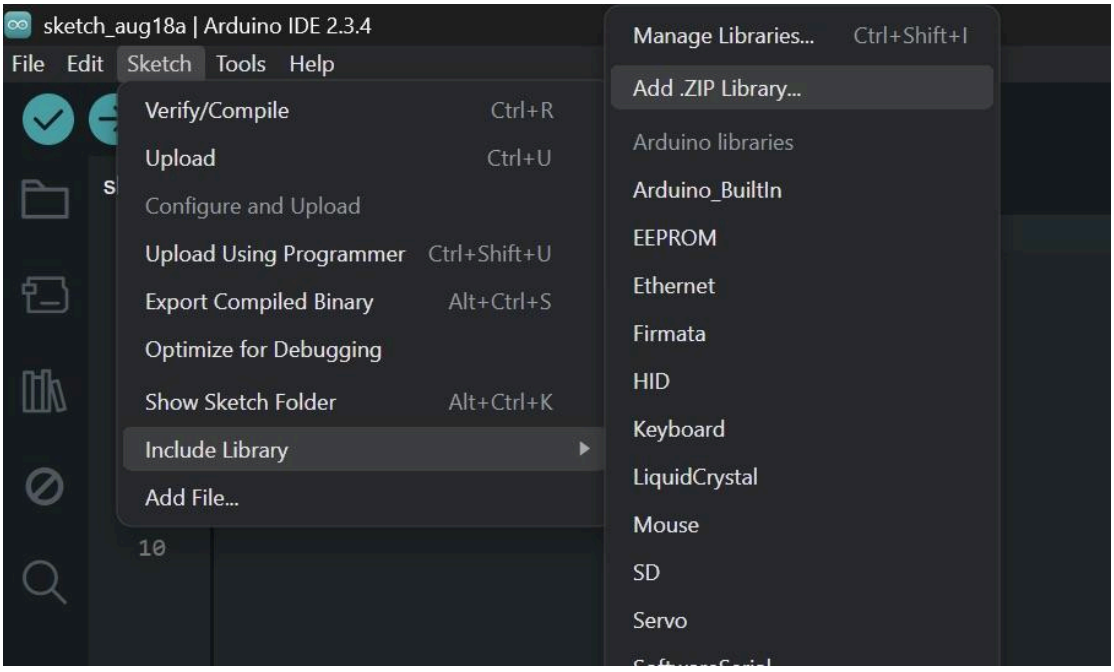

Figure 9: Click Path for Adding .ZIP Library in Arduino

5. Navigate to the **Global Area** of the arduino sketch.
    a. Import the **LiquidCrystal_I2C** library.
    b. Declare the LiquidCrystal_I2C object as **lcd**. Important: The lcd object requires 3 parameters to create successfully: the address of the lcd (in this case 0x27), and the numbers of **Columns** and **Rows** on the screen itself. In this lab, a **16 x 2** display is used.
6. Navigate to the **Setup Area** of the arduino sketch. This is also referred to as the **setup()** function.
    a. Call the LiquidCrystal_I2C function to initialize the display.
    b. Call the LiquidCrystal_I2C function to turn on the backlight of the display. There will not be any immediate results from calling these functions.
7. Navigate to the **Loop Area** of the arduino sketch. This is also referred to as the **loop()** function.
    a. Call the LiquidCrystal_I2C function that sets the cursor to the top left of the LCD. This function accepts two integers, **Column** and **Row**.
    b. Call the LiquidCrystal_I2C function that prints “Hello World” to the LCD starting at the coordinate the cursor was set to.
8. Select the Arduino Uno as the board for the sketch, seen in Figure 10 below.

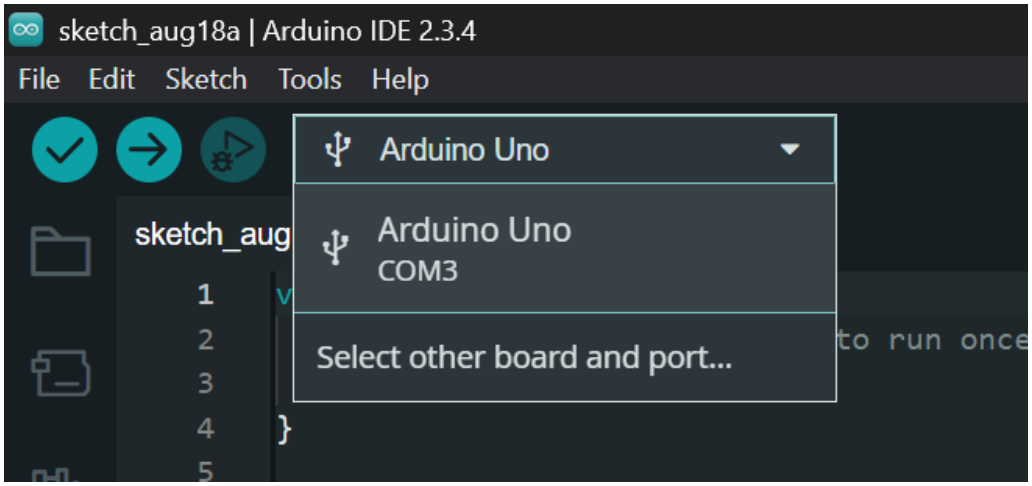

Figure 10: Arduino Selection/Connection Menu

9. Select the COM port that your Arduino is plugged into (Figure 11). **Note:** If the COM port is listed as PORT 1, plug the Arduino into another port and repeat this step.

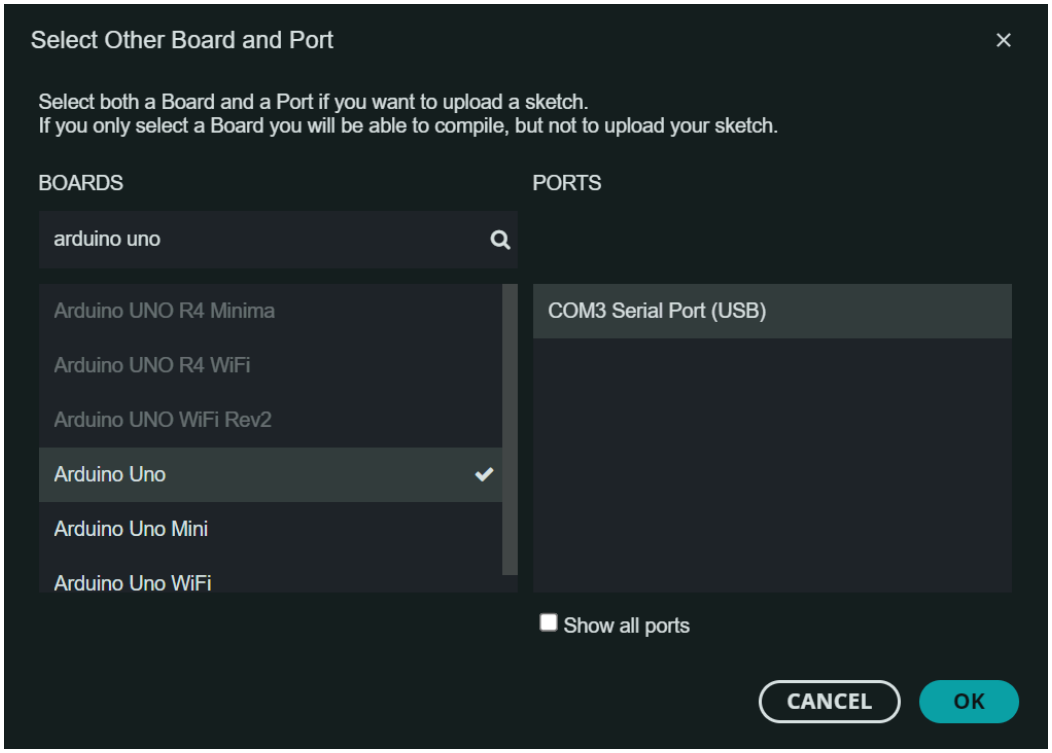

Figure 11: Arduino Other Board and Port Menu

10. Navigate to the top left corner of the Arduino interface and select **Verify**. This will compile the code and catch any syntax errors.

11. **Upload** the sketch to the Arduino. This will automatically run the code, printing “Hello World” on the LCD.

*** Checkpoint: TA must verify the results before moving on to Part 2 ***

Part 2: Creating A Code Flowchart

1. Log into your LucidChart account, which should have been created after the lecture. Note: This software is useful for flowcharts and beneficial for the SLDP.
2. Refer to the example provided, and construct a code flowchart that outlines that reflects part 3 of the lab. Refer to the overview of the lab for details.

*** Checkpoint: TA must approve the flowchart before moving on to Part 3 ***

Part 3: Integrate the LED, Ultrasonic Sensor, and Serial Monitor

1. Disconnect the power cable from the Arduino.
2. Wire the circuit according to the wiring diagram below in Figure 12.
    a. Connect the Ultrasonic Sensor to the corresponding digital pins. Connect **ECHO** to pin **8** and **Trig** to pin **9**. **Note:** If you connect the TRIG and ECHO to different pins, you will need to modify the code accordingly.

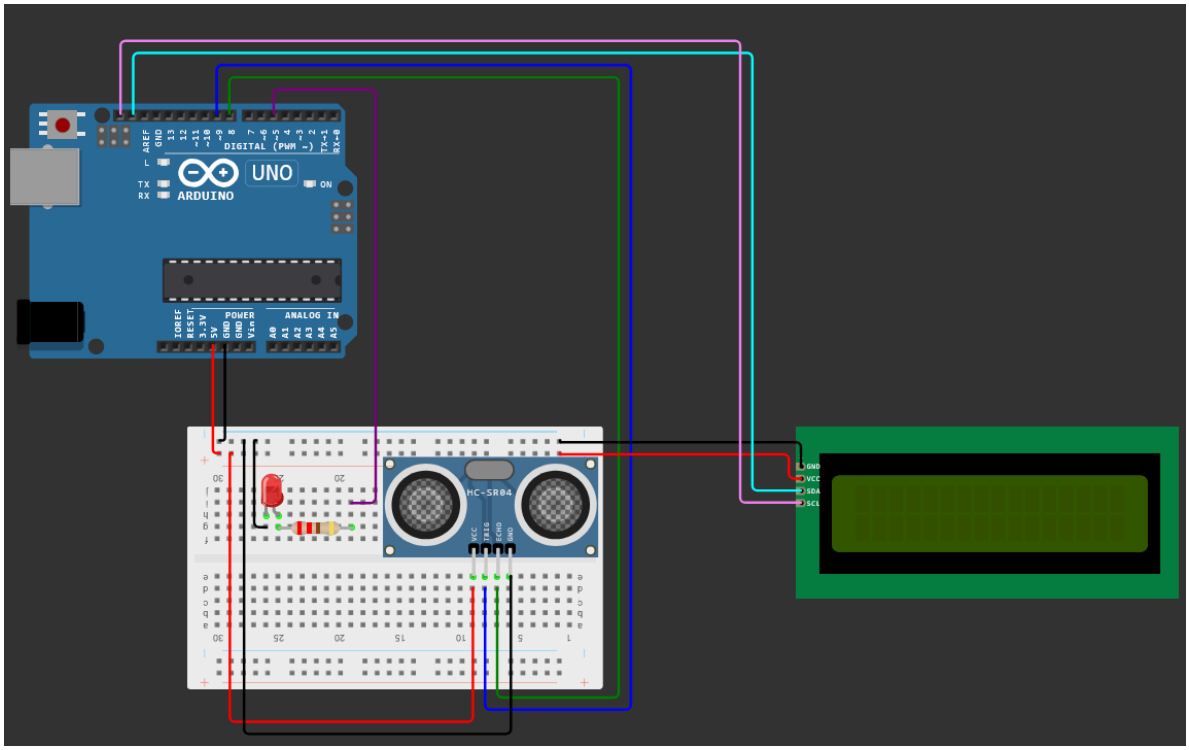

Figure 12: Full Circuit Diagram (https://wokwi.com/)

b. Make sure to not block the emitter and receiver of the ultrasonic sensor.
Note: Connect wires behind the emitter and receiver as in Figure 13.

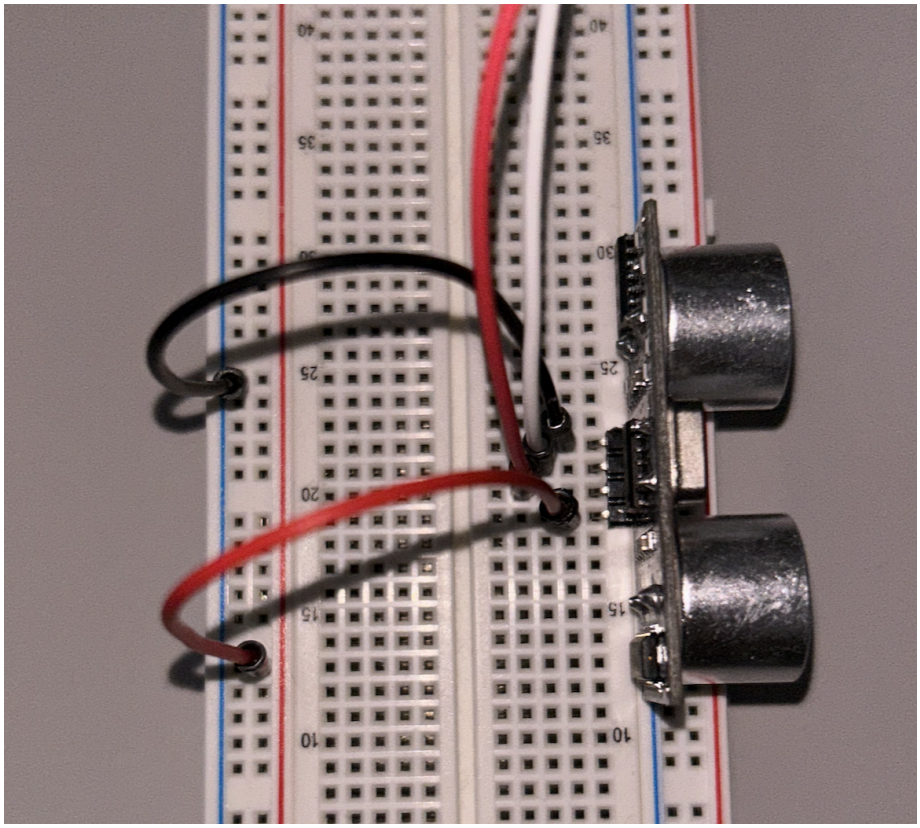

Figure 13: Proper Placement and Connection of Ultrasonic Sensor

c. Connect the **anode** of the LED to pin **5** and the **cathode** of the LED to GND. **Note:** Refer to Figure 14 to determine the cathode and anode of the LED.

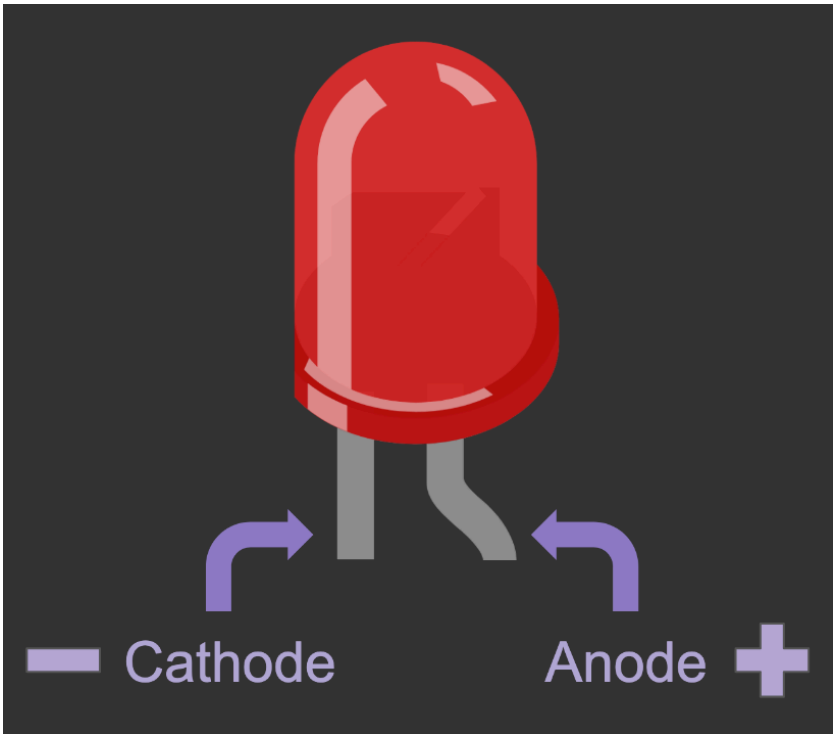

Figure 14: LED **Cathode** and **Anode**

*** Checkpoint: TA must approve the circuit before moving on to Step 3 ***

3. In the **Global** Area:
    a. Declare three constant integers. These represent the **TRIG (trigger)** and **ECHO** of the Ultrasonic Sensor and the LED. These variables must be set to the pins they are connected to.
    b. Declare a float that represents the **duration** value that will be measured by the Ultrasonic Sensor.
    c. Declare one integer that represents the **brightness** of the LED.
4. In the **Setup** Area:
    a. Set the trigger and led variable as **output** pins, and the echo as an **input** pin.
    b. Call the function to turn on the serial monitor with a baud rate of 9600.
5. In the **Loop** Area:
    a. Delete the previous code for the LCD in the loop area.
    b. Ultrasonic Sensor
        i. Turn off the trigger signal for two microseconds, then send out a signal for ten microseconds. Turn off the trigger signal again.

ii. Set the variable that represents the **duration** of the signal to receive a **pulse input** from the sensor.

c. LED Brightness Mapping
   i. Set the integer variable that represents brightness to call the function that maps the **duration** to the **duty cycle** of the LED. Note: the further the object, the dimmer the LED should be. **Note:** Map a duration of 0 → 3000 to an LED brightness of 255 → 0.
   ii. Call the function to send an analog signal to the LED that uses the brightness variable above.

d. Printing Data
   i. Call the function(s) to print **duration** on the LCD.
   ii. Call the function to print **duration** on the Serial monitor. **Note:** The data must print in the format displayed in Figure 15 for proper data collection.
   iii. Set a delay of 500 microseconds.

```
Duration
1366.00
1374.00
1343.00
1374.00
1349.00
1344.00
1374.00
1400.00
1395.00
```

Figure 15: Serial Monitor Data Format

Part 4: Enabling the Data Streamer

1. Download the Student Data Sheet as an Excel file and go to File > Options > Add-ins. Options is the last tab.
2. Under Manage, select COM Add-ins click Go.
3. Select 'Microsoft Data Streamer for Excel' add-in and then click OK. Other add-ins should be unselected.
4. Click 'Connect a Device' under the Data Streamer > Data Sources and select 'Arduino Uno (COM#)'.
5. Navigate to the 'Settings' tab at the bottom of the screen. Change parameters to the following:

a. Data interval (ms) = 1000 ms (unchanged)
b. Data rows = 100

Part 5: Data Collection

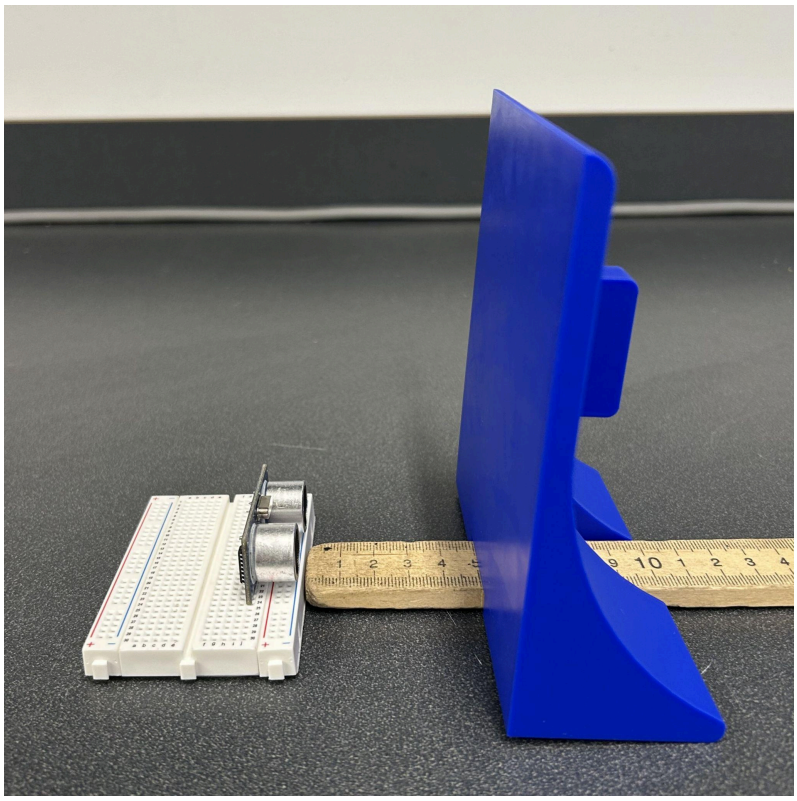

Figure 16: Ultrasonic Sensor, Meterstick, and Target Setup

1. Align the Ultrasonic Sensor with the meter stick, placing it at the 0 cm mark on the lab workstation. **Note:** Ensure the sensor is perpendicular to the length of the meter stick and faces straight ahead (Figure 16).

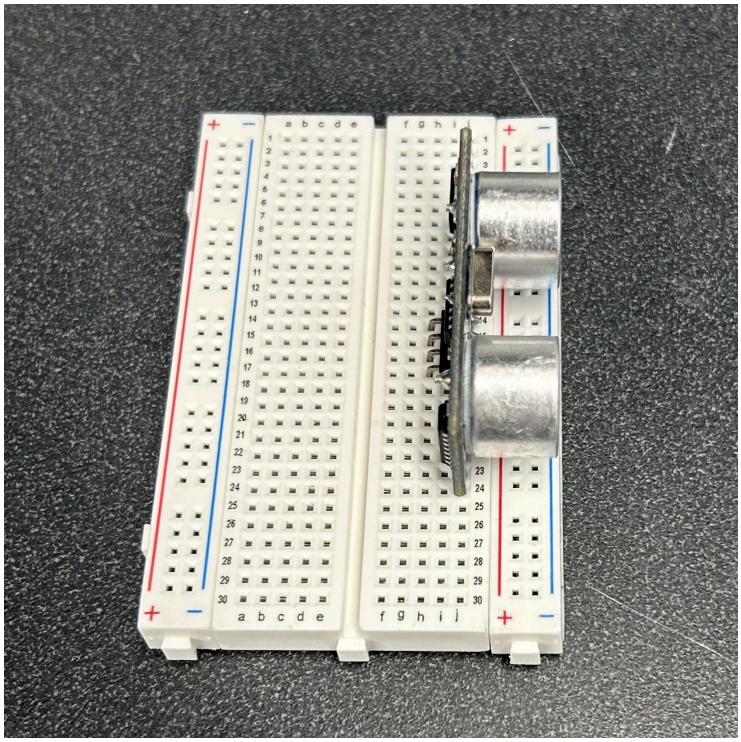

Figure 17: Ultrasonic Sensor Placement on Breadboard

2. Position the flat target by placing it 5 cm away from the sensor along the meter stick. **Note:** Ensure the ultrasonic sensor is in line with the edge of the breadboard for correct readings (Figure 17).
3. Check the LCD screen to verify that **duration** is printing and updating.
4. Begin recording data (Press **Start Data**).
    a. Make sure the target is perpendicular to the lab bench and directly in front of the sensor (Figure 16).
5. Wait for the reading to stabilize on the LCD screen..
6. Record the duration values obtained from the ultrasonic sensor for **2 minutes (or until the 100 data points are collected)**.
7. Press **Stop Data**. Copy the duration and time values to the corresponding excel data tab.
8. Repeat the above steps for each 5 cm interval up to 50 cm.

Part 6: Data Analysis

1. Complete the Student Data Sheet and associated graphs.

Assignment

Skills Workshop 4 Technical Memo

Prepare a memo of no more than 750 words, written from the perspective of an employee addressing a supervisor at an electronic component manufacturing company. The employee is testing the accuracy of an ultrasonic component that the manufacturer intends to sell. This memo will communicate the results of the test completed. This memo should include any illustrations (graphics, figures, images) that will aid in the supervisor's understanding of the results. Here is the writing professor's rubric and TA's rubric for the memo and a sample memo aligned with this assignment.

Use the subject line and outline below to complete this assignment:

Subject: Progress on Coding for Ultrasonic Sensor Testing

In the body, include:

- A list of who completed the work and the date it was completed
- A brief description of the experimental procedure, including:
    - A description of assembled circuit, including a labeled circuit diagram
    - An explanation of the code, including a code flow chart and screenshot of commented code
- A presentation of the data obtained from the experiment, through illustrations such as tables and graphs, and accompanying descriptions of each illustration
- An analysis of the data obtained, including:
    - A discussion of the accuracy and precision of the results. Note: the standard for the ultrasonic sensor is the speed of sound at 343.00 m/s in air at 20.00 °C or 1,125.00 ft/s in air at 68.00 °F.
    - A discussion of any errors or discrepancies in the results
    - Suggestions for improvements and follow-up experiments

Skills Workshop 4 Presentation

This assignment is due at 11:59 PM the night before Recitation 5.

In teams, create a slide deck in PowerPoint based on this skills workshop. This presentation should follow the guidelines included in the How to Give a Technical Presentation slide deck for guidelines on presentation style and formatting. Teams will present this slide deck during Recitation 5; all presentations must be no more than 5 minutes long. Your recitation instructors will give you feedback based on this rubric.

The presentation should include:

- The objective of the experiment
- A brief explanation of the motivation behind the experiment
- A brief walkthrough of any relevant scientific concepts, theories, or vocabulary needed to understand the data collected
- A description of the experimental procedure, including:
    - A description of assembled circuit, including a labeled circuit diagram
    - An explanation of the code, including a code flow chart
- A presentation of the data obtained from the experiment, through illustrations such as tables and graphs, and accompanying descriptions of each illustration
- An analysis of the data obtained, including:
    - A discussion of the accuracy and precision of the results. Note: the standard for the ultrasonic sensor is the speed of sound at 343.00 m/s in air at 20.00 °C or 1,125.00 ft/s in air at 68.00 °F.
    - A discussion of any errors or discrepancies in the results
    - Suggestions for improvements and follow-up experiments