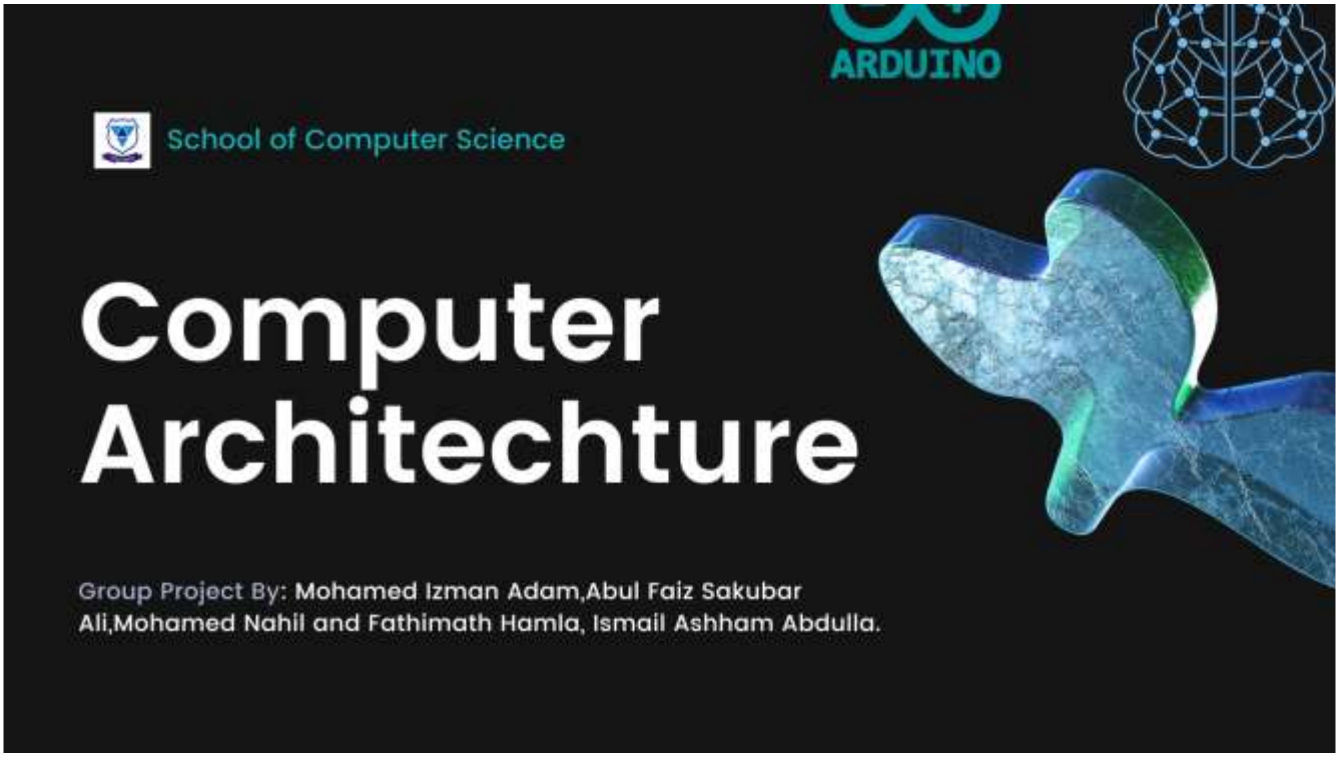


School of Computer Science

Computer Architechture

Group Project By: Mohamed Izman Adam,Abul Faiz Sakubar Ali,Mohamed Nahil and Fathimath Hamla, Ismail Ashham Abdulla.

Introduction

This is a report prepared by the members of Group C on the 3 Arduino programming tasks to be completed for the Computer Systems Architecture assignment and will mainly consist of explanations of our understandings, our methodology when approaching each task and the results of our experimentations. Our approach was to derive various conclusions through experimentation, whether it gave us the desired result or not and in doing so, these tasks have helped us gain a better understanding of Arduinos and its hardware functionality and given us a good amount of practice with programming and tinkering. From here on each section will be dedicated to a task and the components used in each task and images will be provided where required.

Task 2

The goal of this task as stated in the assignment specifications is to convert the analogue reading from an Infrared Sensor on Arduino A into a binary value same as the digital reading and display the converted analogue reading and digital reading on the Serial Monitor of Arduino B via I2C communication.

Components Used -

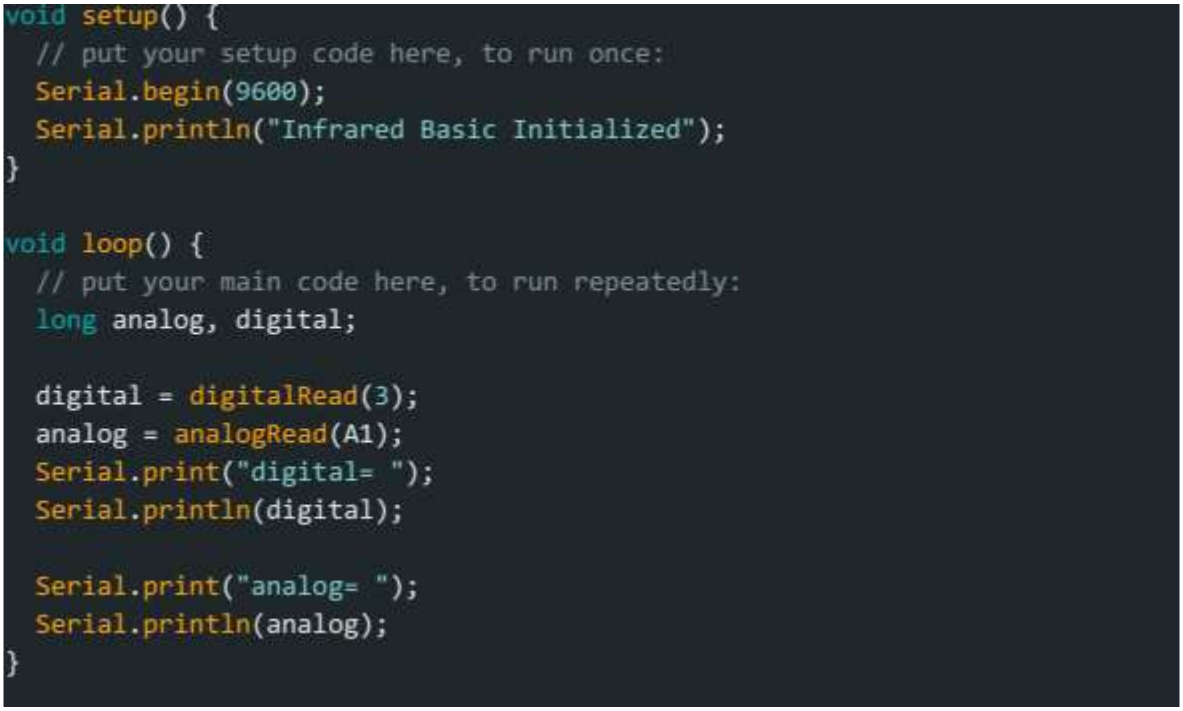
Arduino Uno Rev3 - Is a microcontroller board and is the main component that everything is based around. It is connected to a PC or Laptop running the Arduino IDE which is used to code in the behavior of the Arduino and in turn, the other connected parts. Two such boards were used for this task.

MH Sensor Series IR Sensor - An Infrared sensor that also measures distance between itself and an object within its area. Uses Infrared Waves to detect the distance and takes in an analog reading and a digital reading, which changes depending on the analog reading.

Jumper Wires - These electric wires consisting of both type Male to Female and Male to Male, are used to connect all the parts together directly and indirectly. Connections are made to the different pins on the Arduino.

Methodology

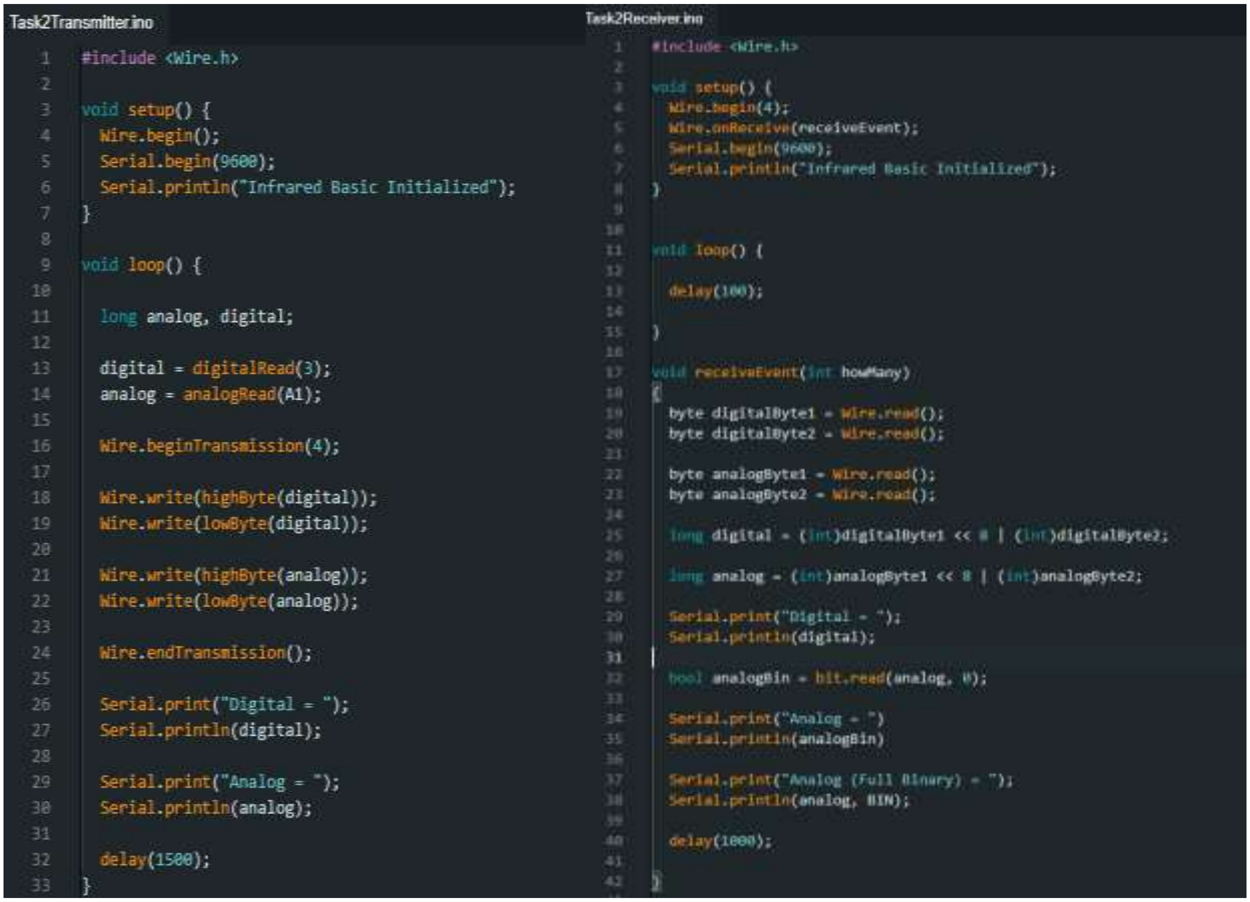
From the code our professor gave us for task 2 we found that the light would turn on and off on the IR Sensor and found that this also corresponds to a change in the digital value to equal 1 from 0 on the Serial Monitor while the analogue reading would constantly be changing and would remain within a 1 – 2000 range or so. We figured that the analogue and digital reading were related in some way so we determined the number the analogue reading was at when the light turns off. We did this by slowly and slightly moving the IR Sensor until the light turned off and determined this value to be approximately 529.



```
void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);
  Serial.println("Infrared Basic Initialized");
}

void loop() {
  // put your main code here, to run repeatedly:
  long analog, digital;

  digital = digitalRead(3);
  analog = analogRead(A1);
  Serial.print("digital= ");
  Serial.println(digital);

  Serial.print("analog= ");
  Serial.println(analog);
}
```

For communication between two Arduino we used the Wire.h example code also given to us by our professor (code is not shown here due to size). In it the receiver code initializes an Arduino sketch that communicates over the I2C protocol using the Wire library. The void setup() function configures the board as an I2C receiver with address 4 and registers a callback function receiveEvent for handling incoming data. The main loop has a delay, allowing for continuous execution. The receiveEvent function reads four bytes from the I2C bus and reconstructs two 16-bit integer values (y1 and y2) by combining pairs of bytes. It then prints these received values over the serial connection.

The transmitter code sets up another Arduino sketch to act as the I2C transmitter. In the setup() function, the Wire library is initialized for I2C communication, and the serial connection is started for debugging. The void loop() function contains a loop that iterates while incrementing one integer (a) and decrementing another (b) within predefined limits. During each iteration, the transmitter board initiates a transmission to the slave at address 4 using Wire.beginTransmission(). It sends the high and low bytes of both a and b using Wire.write(), and then ends the transmission with Wire.endTransmission(). The values of a and b are printed to the serial monitor for monitoring purposes. The loop introduces a delay to control the transmission rate, and after completing a series of transmissions, there's a longer delay before the process repeats. This was the basis of our own code and we came up with two approaches to the task. (Below is the first approach)



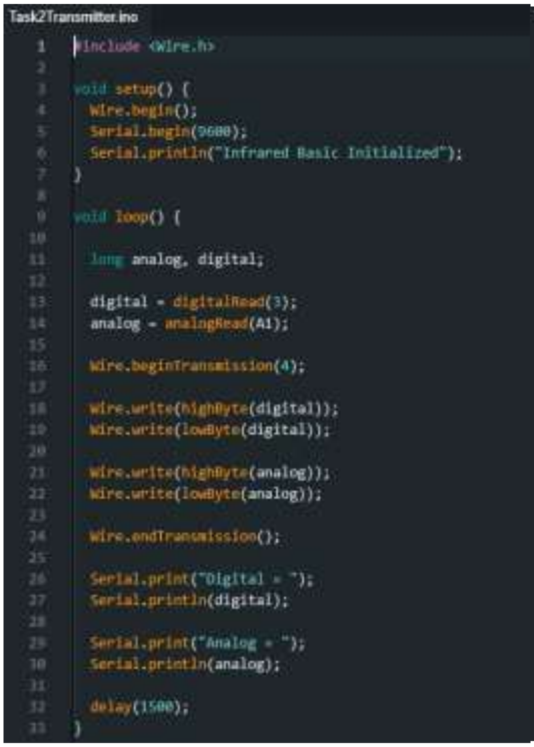
Task2Transmitter.ino

```
#include <Wire.h>

void setup() {
  Wire.begin();
  Serial.begin(9600);
  Serial.println("Infrared Basic Initialized");
}

void loop() {

  long analog, digital;

  digital = digitalRead(3);
  analog = analogRead(A1);

  Wire.beginTransmission(4);

  Wire.write(highByte(digital));
  Wire.write(lowByte(digital));

  Wire.write(highByte(analog));
  Wire.write(lowByte(analog));

  Wire.endTransmission();

  Serial.print("Digital = ");
  Serial.println(digital);

  Serial.print("Analog = ");
  Serial.println(analog);

  delay(1500);
}
```

Task2Receiver.ino

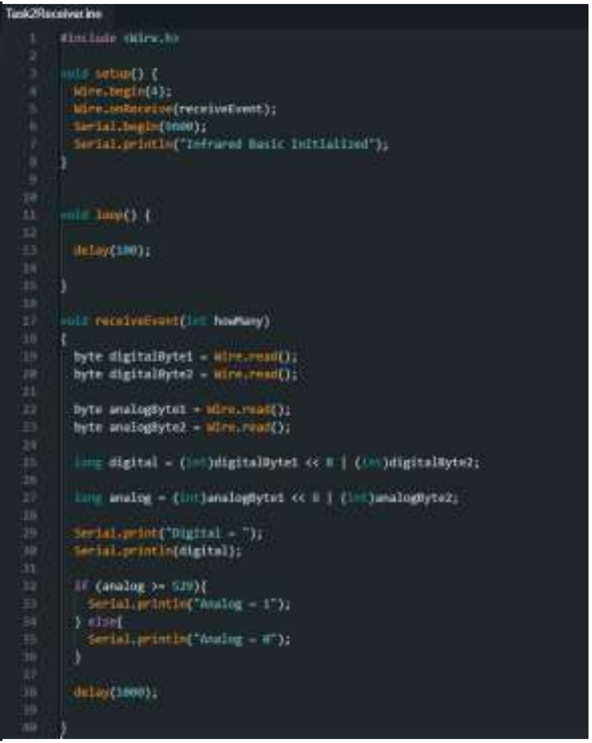
```
#include <Wire.h>

void setup() {
  Wire.begin(4);
  Wire.onReceive(receiveEvent);
  Serial.begin(9600);
  Serial.println("Infrared Basic Initialized");
}


void loop() {

  delay(100);

}

void receiveEvent(int howMany)
{
  byte digitalByte1 = Wire.read();
  byte digitalByte2 = Wire.read();

  byte analogByte1 = Wire.read();
  byte analogByte2 = Wire.read();

  long digital = (int)digitalByte1 << 8 | (int)digitalByte2;

  long analog = (int)analogByte1 << 8 | (int)analogByte2;

  Serial.print("Digital = ");
  Serial.println(digital);

  bool analogBin = bit.read(analog, 0);

  Serial.print("Analog = ")
  Serial.println(analogBin)

  Serial.print("Analog (Full Binary) = ");
  Serial.println(analog, BIN);

  delay(1000);

}
```

(Below is the second approach)



Task2Transmitter.ino

```
#include <Wire.h>

void setup() {
  Wire.begin();
  Serial.begin(9600);
  Serial.println("Infrared Basic Initialized");
}

void loop() {

  long analog, digital;

  digital = digitalRead(3);
  analog = analogRead(A1);

  Wire.beginTransmission(4);

  Wire.write(highByte(digital));
  Wire.write(lowByte(digital));

  Wire.write(highByte(analog));
  Wire.write(lowByte(analog));

  Wire.endTransmission();

  Serial.print("Digital = ");
  Serial.println(digital);

  Serial.print("Analog = ");
  Serial.println(analog);

  delay(1500);
}
```

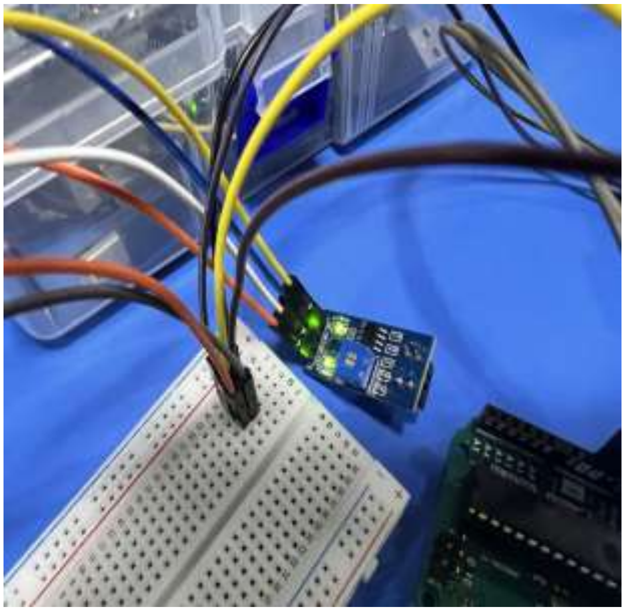


Task2Receiver.ino

```
#include <Wire.h>

void setup() {
  Wire.begin(4);
  Wire.onReceive(receiveEvent);
  Serial.begin(9600);
  Serial.println("Infrared Basic Initialized");
}


void loop() {

  delay(100);

}

void receiveEvent(int howMany)
{
  byte digitalByte1 = Wire.read();
  byte digitalByte2 = Wire.read();

  byte analogByte1 = Wire.read();
  byte analogByte2 = Wire.read();

  long digital = (int)digitalByte1 << 8 | (int)digitalByte2;

  long analog = (int)analogByte1 << 8 | (int)analogByte2;

  Serial.print("Digital = ");
  Serial.println(digital);

  if (analog >= 529){
    Serial.println("Analog = 1");
  } else{
    Serial.println("Analog = 0");
  }

  delay(1000);

}
```

Result & Conclusions

Due to limited lab time and the unavailability of the model of IR Sensor on the simulation site Tinkercad, we were unable to test the two approaches thoroughly but we believe the second approach is more valid. (Below is shown IR Sensor Setup during earlier testing phase)



Task 3

In the domain of electronics, the interaction between devices can pave the way for innovative solutions. Our project centers around the collaboration of two Arduino microcontrollers, Arduino A and Arduino B, using an ultrasonic sensor to measure distances and relay the data. This section delves into the intricacies of their interaction within the larger context of the project.

Components Used -

Arduino Uno Rev3 - Is a microcontroller board and is the main component that everything is based around. It is connected to a PC or Laptop running the Arduino IDE which is used to code in the behavior of the Arduino and in turn, the other connected parts. Two such boards were used for this task.

HC-SR04 Ultrasonic Sensor - A distance sensor that sends out and receives Ultrasonic waves via an Ultrasonic transmitter to measure distance of any object within 2cm to 400cm.

Jumper Wires - These electric wires consisting of both type Male to Female and Male to Male, are used to connect all the parts together directly and indirectly. Connections are made to the different pins on the Arduino.

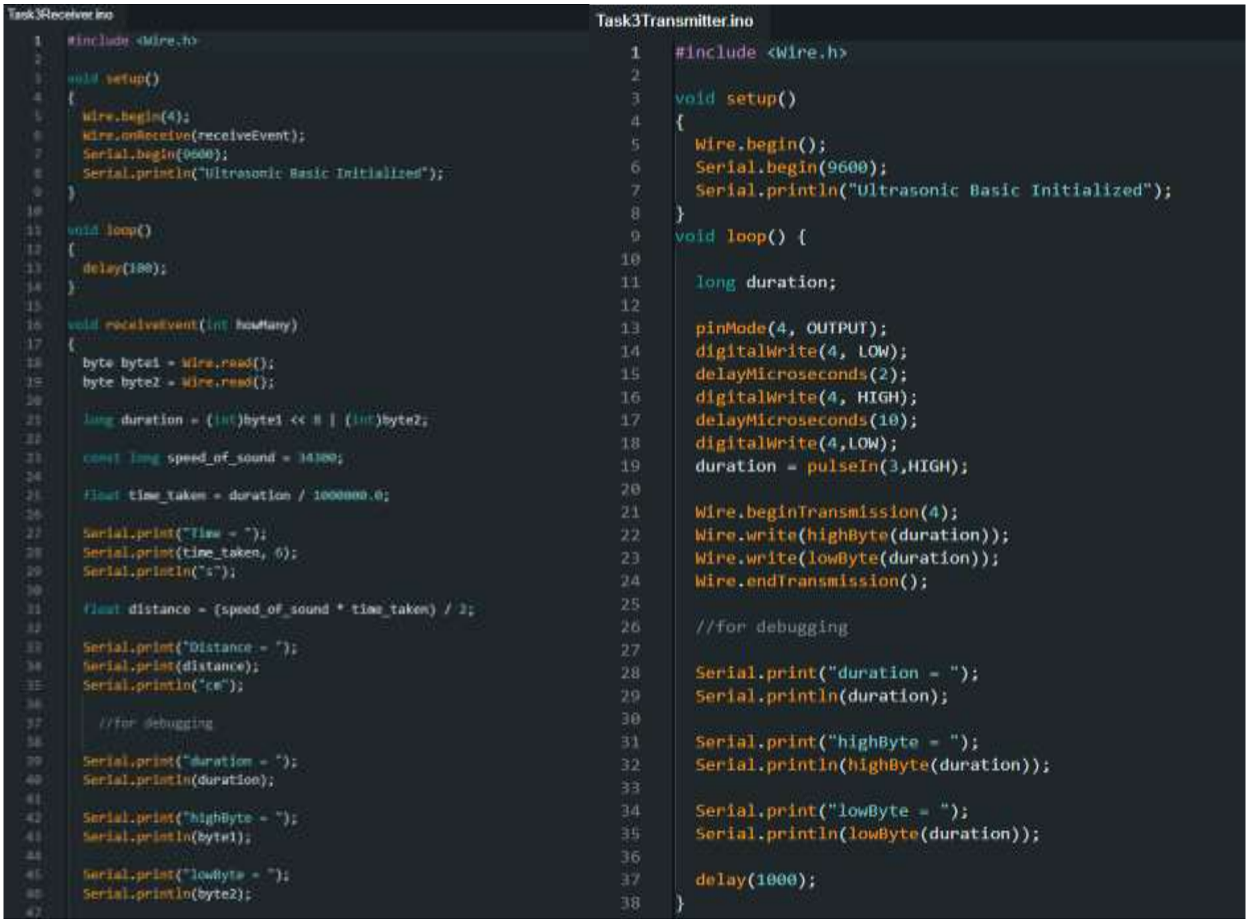
Methodology

At the heart of our project lie Arduino A and Arduino B, each playing a distinct role in the grand scheme. Arduino A acts as the sender, tasked with utilizing an ultrasonic sensor to measure distances, while Arduino B assumes the role of the receiver, responsible for interpreting and displaying the received data.

The seamless interaction between the two Arduinos hinges upon a carefully structured communication protocol. Arduino A employs the Wire.h library to send the distance data to Arduino B via the I2C communication protocol. This protocol facilitates a smooth exchange of information, as Arduino A encapsulates the distance data into bytes using the Wire.write() function. Arduino B, equipped with the Wire library, listens for the incoming data, and upon reception, processes it to extract the distance measurements.

Arduino A's journey initiates with the ultrasonic sensor, an instrumental component in distance measurement. By emitting sound waves and gauging their return time, Arduino A computes the time taken for the waves to travel and return. This duration, measured in microseconds, forms the basis for subsequent calculations. Additionally, Arduino A employs the speed of sound (343,000 cm/s) to convert the duration into distance, ensuring an accurate portrayal of the object's proximity.

Arduino B's role comes to fruition as it awaits the incoming data from Arduino A. With the aid of the Wire library, Arduino B seamlessly receives the bytes of information. Upon successful reception, Arduino B employs bitwise operations to reconstruct the original data, thereby unveiling the distance measurements. These measurements are subsequently displayed on the serial monitor, providing a tangible representation of the object's distance from the ultrasonic sensor.



Task3Receiver.ino

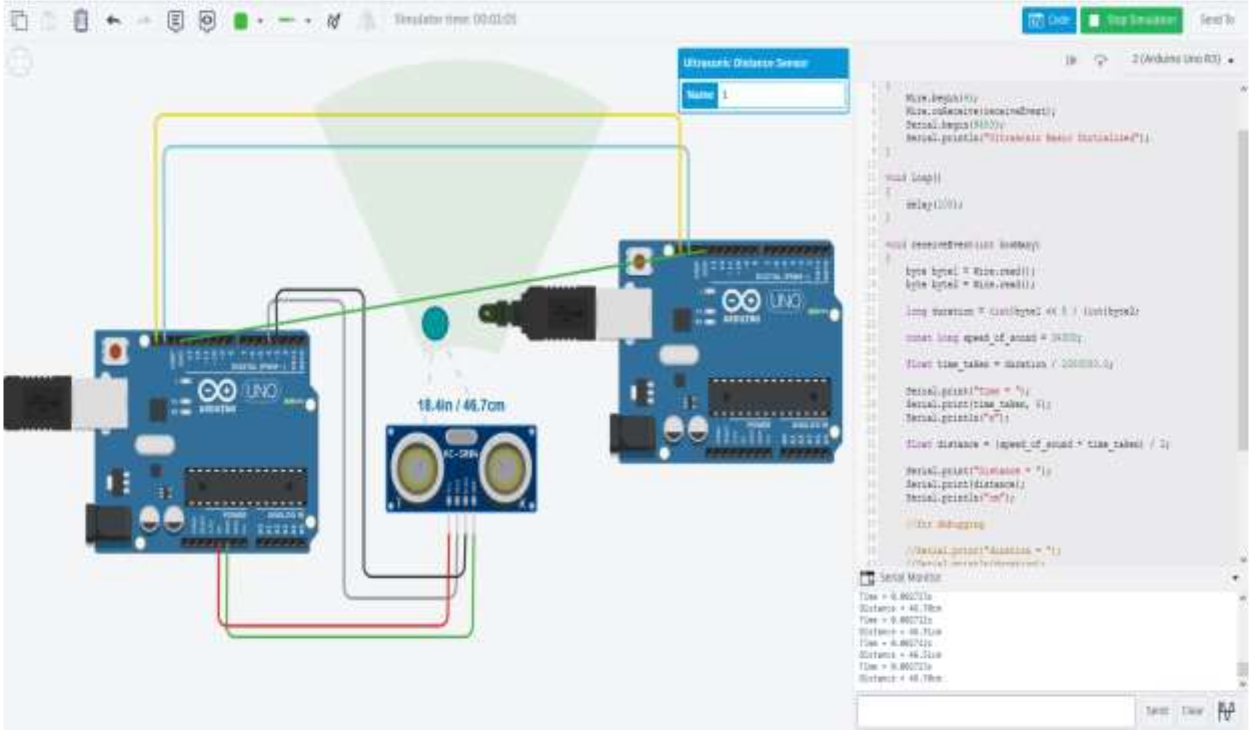
```
#include <Wire.h>

void setup()
{
  Wire.begin(4);
  Wire.onReceive(receiveEvent);
  Serial.begin(9600);
  Serial.println("Ultrasonic Basic Initialized");
}

void loop()
{
  delay(100);
}

void receiveEvent(int howMany)
{
  byte byte1 = Wire.read();
  byte byte2 = Wire.read();

  long duration = (int)byte1 << 8 | (int)byte2;

  const long speed_of_sound = 34300;

  float time_taken = duration / 1000000.0;

  Serial.print("Time = ");
  Serial.print(time_taken, 6);
  Serial.println("s");

  float distance = (speed_of_sound * time_taken) / 2;

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

    //for debugging

  Serial.print("duration = ");
  Serial.println(duration);

  Serial.print("highByte = ");
  Serial.println(byte1);

  Serial.print("lowByte = ");
  Serial.println(byte2);
```

Task3Transmitter.ino

```
#include <Wire.h>

void setup()
{
  Wire.begin();
  Serial.begin(9600);
  Serial.println("Ultrasonic Basic Initialized");
}
void loop() {

  long duration;

  pinMode(4, OUTPUT);
  digitalWrite(4, LOW);
  delayMicroseconds(2);
  digitalWrite(4, HIGH);
  delayMicroseconds(10);
  digitalWrite(4,LOW);
  duration = pulseIn(3,HIGH);

  Wire.beginTransmission(4);
  Wire.write(highByte(duration));
  Wire.write(lowByte(duration));
  Wire.endTransmission();

  //for debugging

  Serial.print("duration = ");
  Serial.println(duration);

  Serial.print("highByte = ");
  Serial.println(highByte(duration));

  Serial.print("lowByte = ");
  Serial.println(lowByte(duration));

  delay(1000);
}
```

Result and Conclusions

The harmony between code and hardware is the backbone of this interaction. Arduino A's code orchestrates the measurements, data encapsulation, and transmission processes, ensuring that the ultrasonic sensor's data finds its way to Arduino B. Meanwhile, Arduino B's code acts as the interpreter, unraveling the received data and translating it into a visual representation on the serial monitor.



```
{
    Wire.begin(9);
    Wire.onReceive(receiveEvent);
    Serial.begin(9600);
    Serial.println("Ultrasonic Sensor Initialized");
}

void loop()
{
    delay(100);
}

void receiveEvent(int howMany)
{
    byte byte1 = Wire.read();
    byte byte2 = Wire.read();

    long duration = (int)byte1 << 8 | (int)byte2;

    const long speed_of_sound = 34300;

    float time_taken = duration / 1000000.0;

    Serial.print("Time = ");
    Serial.print(time_taken, 6);
    Serial.println("s");

    float distance = (speed_of_sound * time_taken) / 2;

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

    //For debugging

    //Serial.print("duration = ");
    //Serial.println(duration);
```

```
Time = 0.002721s
Distance = 46.70cm
Time = 0.002721s
Distance = 46.71cm
Time = 0.002741s
Distance = 46.31cm
Time = 0.002721s
Distance = 46.70cm
```

Bonus Task

The goal of this task as stated in the assignment specifications is to display the results gotten from the previous two tasks on a single Arduino connected to an LCD. For this task we use the setups for both the Ultrasonic Sensor and the IR Sensor but instead of displaying values on the Serial Monitor we are to display them on a Liquid Crystal Display and I2C communication takes place between the LCD and Arduino, so a slight modification to the code is needed but to read the values we use the same code from the previous two setups.

Components Used -

Arduino Uno Rev3 - Is a microcontroller board and is the main component that everything is based around. It is connected to a PC or Laptop running the Arduino IDE which is used to code in the behavior of the Arduino and in turn, the other connected parts.

PCF8574-based I2C Liquid Crystal Display - This is the main new component we will be using and it is an LCD consisting of 2 rows of 16 individual columns of pixels which can be used to display any type of character. The characters themselves will be of a white color with the background being a bright blue/purple. The LCD from what we observed has a similar look and feel to what is seen on calculators, although a bit bulkier.

Breadboard - This is also a new component. Since the Arduino only has a single 5v Pin and Ground Pin, a Breadboard is used to circuit power from the pins to all the different components and acts as somewhat of a central junction point for the Arduino.

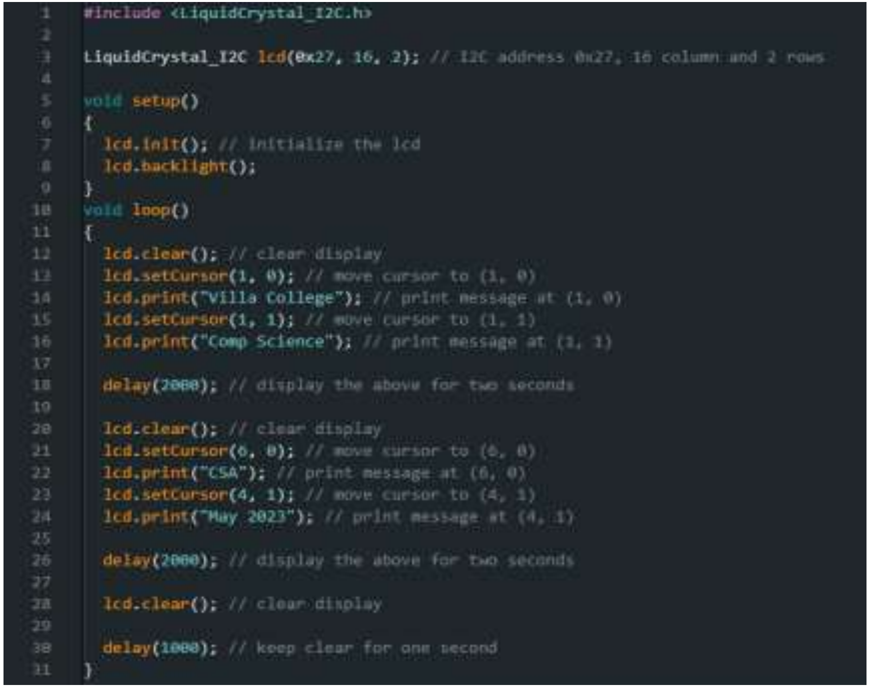
HC-SR04 Ultrasonic Sensor - A distance sensor that sends out and receives Ultrasonic waves via an Ultrasonic transmitter to measure distance of any object within 2cm to 400cm.

MH Sensor Series IR Sensor - An Infrared sensor that also measures distance between itself and an object within its area. Uses Infrared Waves to detect the distance and takes in an analog reading and a digital reading, which changes depending on the analog reading.

Jumper Wires - These electric wires consisting of both type Male to Female and Male to Male, are used to connect all the parts together directly and indirectly. Direct connection involves connecting to the pins of the Arduino and Indirect connection involves connecting to adjacent and corresponding positions in a breadboard.

Methodology

At first, we tried to understand how exactly the LCD was working and to do that we tested this piece of code given by our professor with only it connected to the Arduino.

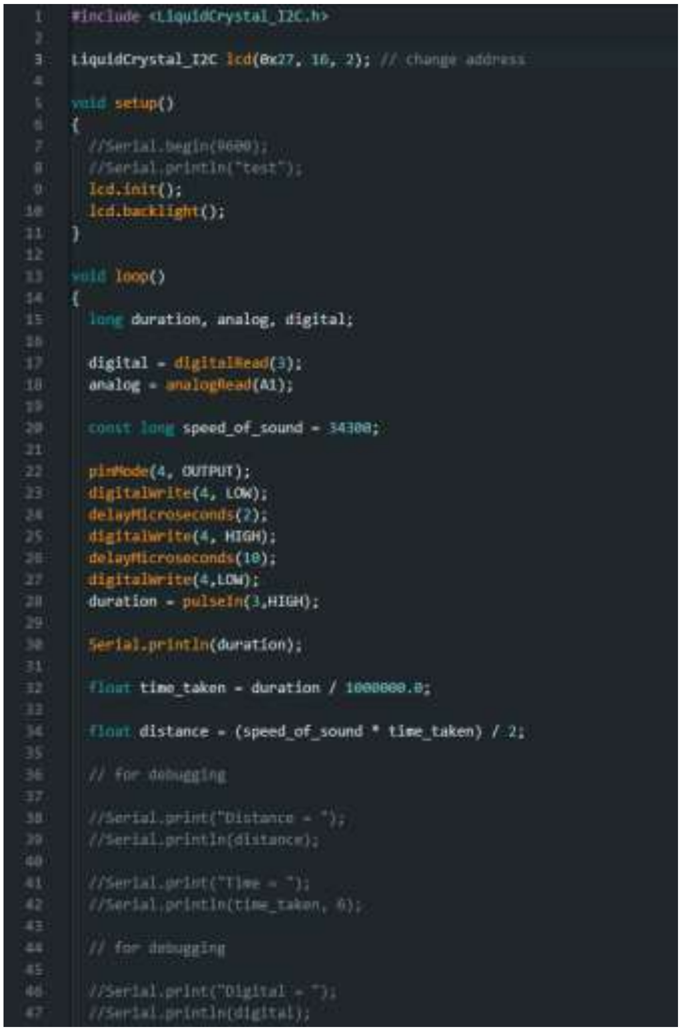


```
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 16, 2); // I2C address 0x27, 16 column and 2 rows

void setup()
{
  lcd.init(); // initialize the lcd
  lcd.backlight();
}
void loop()
{
  lcd.clear(); // clear display
  lcd.setCursor(1, 0); // move cursor to (1, 0)
  lcd.print("Villa College"); // print message at (1, 0)
  lcd.setCursor(1, 1); // move cursor to (1, 1)
  lcd.print("Comp Science"); // print message at (1, 1)

  delay(2000); // display the above for two seconds

  lcd.clear(); // clear display
  lcd.setCursor(6, 0); // move cursor to (6, 0)
  lcd.print("CSA"); // print message at (6, 0)
  lcd.setCursor(4, 1); // move cursor to (4, 1)
  lcd.print("May 2023"); // print message at (4, 1)

  delay(2000); // display the above for two seconds

  lcd.clear(); // clear display

  delay(1000); // keep clear for one second
}
```

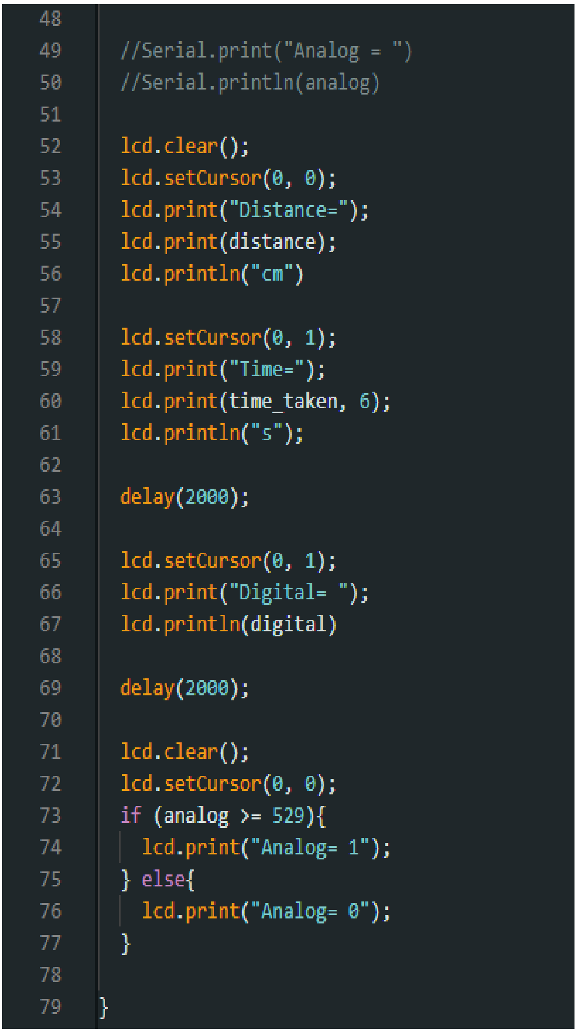
In the above code we can see that the I2C LCD library (there are 2 LCD libraries available, and we use the I2C one) is imported first and throughout the code we will be using the functions provided by this library. To use these functions a lcd object is created via the lcd function and the address of the LCD, given in hexadecimal base-16 format and the columns and rows of the display are passed in as arguments to create it.

In void setup(), the lcd object is initialized for the Arduino, and in void loop() the code functions to output to the LCD are run repeatedly. We use lcd.setCursor(x, y) to set the position of our output on the LCD, where x denotes the column and y denotes the row on the display, following zero-based numbering. The lcd.print() function is then used to print out the output, a string specified inside quotation marks or a variable containing a data value at that position.

After figuring out the LCD code we were able to surmise that it functioned in a similar way to how we were outputting to the Serial Monitor so all we had to do was implement the code we had used in the previous two tasks and make some slight adjustments.



```
#include <LiquidCrystal_I2C.h>

LiquidCrystal_I2C lcd(0x27, 16, 2); // change address

void setup()
{
  //Serial.begin(9600);
  //Serial.println("test");
  lcd.init();
  lcd.backlight();
}

void loop()
{
  long duration, analog, digital;

  digital = digitalRead(3);
  analog = analogRead(A1);

  const long speed_of_sound = 34300;

  pinMode(4, OUTPUT);
  digitalWrite(4, LOW);
  delayMicroseconds(2);
  digitalWrite(4, HIGH);
  delayMicroseconds(10);
  digitalWrite(4,LOW);
  duration = pulseIn(3,HIGH);

  Serial.println(duration);

  float time_taken = duration / 1000000.0;

  float distance = (speed_of_sound * time_taken) / 2;

  // for debugging

  //Serial.print("Distance = ");
  //Serial.println(distance);

  //Serial.print("Time = ");
  //Serial.println(time_taken, 6);

  // for debugging

  //Serial.print("Digital = ");
  //Serial.println(digital);
```

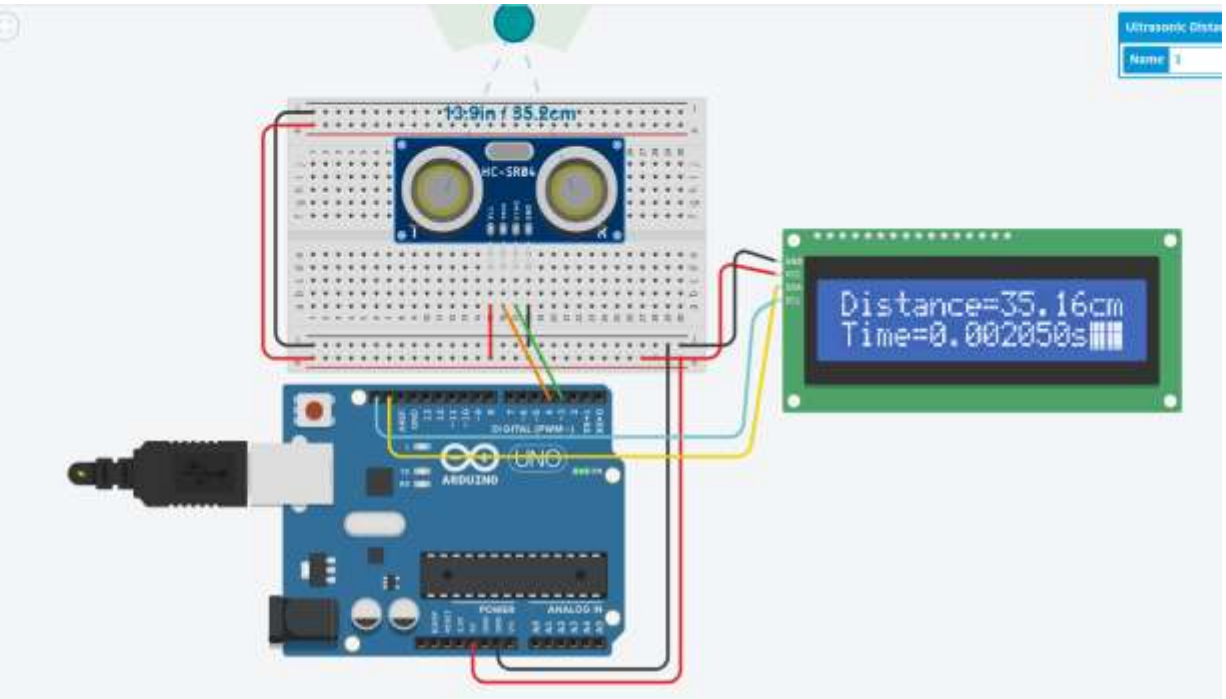


```

  //Serial.print("Analog = ")
  //Serial.println(analog)

  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print("Distance=");
  lcd.print(distance);
  lcd.println("cm")

  lcd.setCursor(0, 1);
  lcd.print("Time=");
  lcd.print(time_taken, 6);
  lcd.println("s");

  delay(2000);

  lcd.setCursor(0, 1);
  lcd.print("Digital= ");
  lcd.println(digital)

  delay(2000);

  lcd.clear();
  lcd.setCursor(0, 0);
  if (analog >= 529){
    lcd.print("Analog= 1");
  } else{
    lcd.print("Analog= 0");
  }

}
```

The code from Task 3 stays much the same but the spacing for the output is decreased here to accommodate for the 16x2 constraints of the LCD screen and display values from the Ultrasonic Sensor (especially considering the varying sizes of the numbers) neatly on the screen. After about 2 seconds the digital and analog values from the IR Sensor are shown. We ended up using our second approach from Task 2 here as we think it is more reliable in theory even though it is a far more straightforward and less elegant solution.

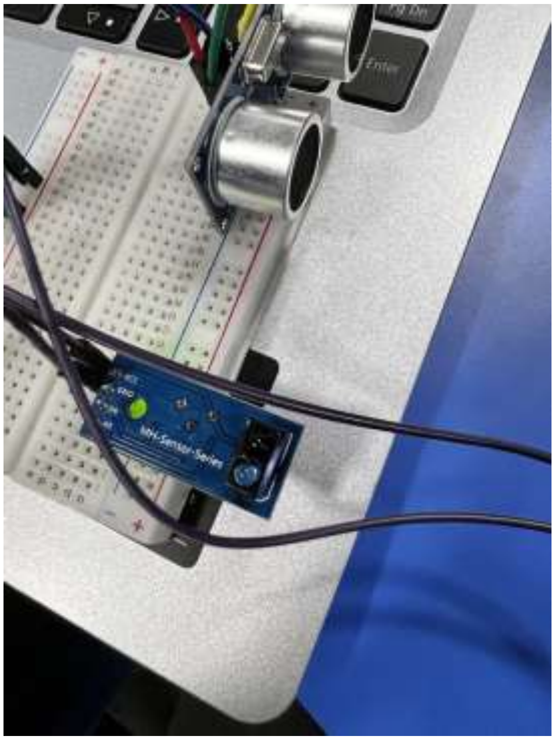
Result and Conclusions

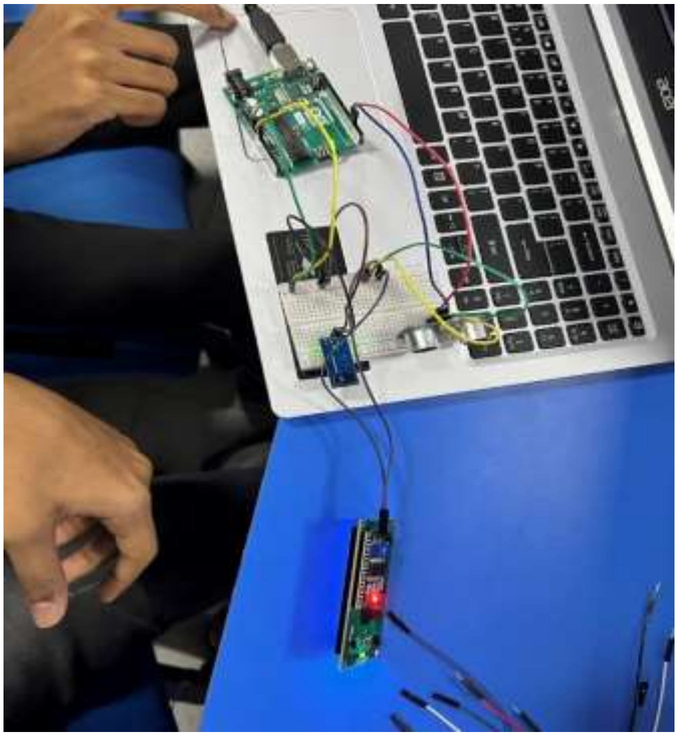
We successfully tested the code snippets of the individual components with the Arduino but due to limited lab time we were unable to thoroughly test the code. So most of the time spent testing was on the simulation site, Tinkercad.



```
Distance=35.16cm
Time=0.002050s
```

We were able to get the Ultrasonic Sensor part of the code working seamlessly but as the MH Sensor Series type IR Sensor is not available on Tinkercad, we could not test the IR sensor part of the code. However, we believe the above shown code would have worked for the most part with some tweaks. (Below are shown incomplete setups we used during the testing phase).





References

Arduino Forum - https://forum.arduino.cc/

Tinkercad - https://www.tinkercad.com/

Michael Schoeffler - https://mschoeffler.com/

Sparkfun Education - https://www.sparkfuneducation.com/index.html