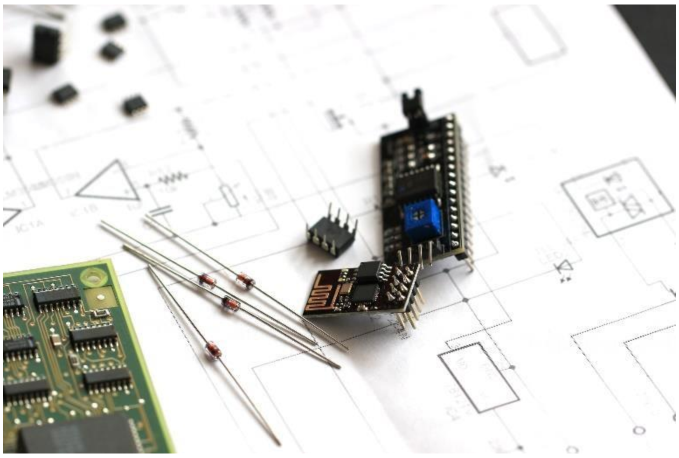
University of Hertfordshire

School of Physics, Engineering & Computer Science

5ENT2049-0901-2024 - Analogue and Mixed-Signal Design

Group Report



THIS COPY BELONGS TO:	*Group 6*

Group members

Name of members	Role/Task taking in this project
Mit Sabhadiya (Leader)	Temperature sensor, Amplifier Design and Amplification, Sensors Schematic and calibration, Filter Design, Arduino-Programming and Integrating Sensors, MATLAB-Encryptions & Decryptions.
Abdul Momin	Lighting Sensor-Designing, Board Implementation and Calibration.
Asif Ahmed	Components Selection, Humidity Sensor Design-Timing, calibration, soldering, Breadboard Implementation.
Chandan Singh	Dimension Mapping, 3D Enclosure, Review Sensors, Documentation.
Yatin Sharma	Block-Diagram, Temperature sensor, Documentation.

Abstract

This report presents the design and testing of an environmental monitoring system measuring humidity and light intensity. Using analog sensors integrated with an ATmega16U2 microcontroller on the Arduino Uno Rev 3, the system processes and displays data on an OLED screen. Error-handling strategies, including smoothing algorithms and range constraints, ensure accuracy and reliability. The system is ideal for controlled environments such as greenhouses, with potential enhancements like wireless modules for remote monitoring.

Table of Contents

LIST OF FIGURES

1. Introduction

1.1 Overview

Environmental monitoring is pivotal in various sectors, including agriculture, industrial processes, and smart home systems. Accurate real-time data on environmental parameters such as humidity and light intensity facilitates optimal decision-making, enhances efficiency, and ensures safety. This report details the design, development, and testing of an advanced environmental monitoring system that measures key environmental metrics and provides real-time data visualization and potential remote monitoring capabilities. Leveraging the versatility of the Arduino platform and Adafruit's OLED FeatherWing display, the system offers a robust solution for diverse monitoring applications.

1.2 Tools and technologies

Group 6 is developing an environmental monitoring system designed to measure humidity and light intensity. The system employs analog signal conditioning using non-inverting amplifiers to enhance the accuracy of sensor readings. Data processing is handled by an Arduino2 microcontroller, and the results are displayed on an OLED screen. Optional features include a mobile application for remote monitoring and control, utilizing wireless communication modules such as Bluetooth or Wi-Fi to facilitate seamless connectivity and accessibility

1.3 Bill of Materials

Component	Manufacturer/Make
Humidity sensor	RS PRO humidity sensor, 2 Pins
Light sensor	NS-19M51 Luna Optoelectronics
Organic LED (OLED) display	Adafruit 2900 (128x32)
Microcontroller	Arduino Uno Rev 3

1.4 Power consumption

The system is engineered for optimal energy efficiency. The Arduino2 microcontroller operates at approximately 5 volts during active use, while the OLED display requires significantly lower voltage levels. Although not implemented in the current design, the system is intended to incorporate “sleep” modes to minimize energy consumption when idle.

1.5 Block Diagram

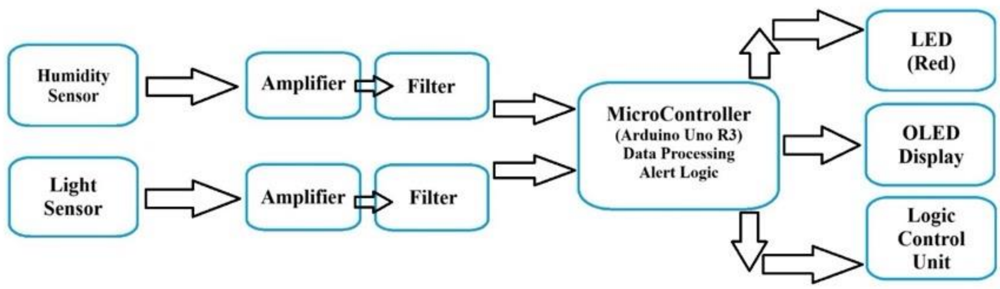


Figure 1-Project block diagram. Block diagrams use flow-chart structure to explain input, output, and intermediate stages of a design process.

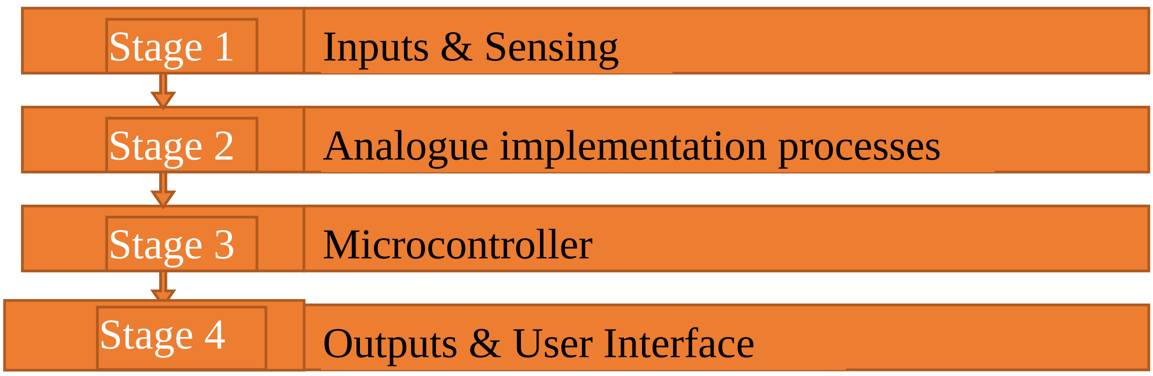


Figure 2-Simplified stages of the block diagram

1.6 Project Timeline

TASK	START DATE	TIME /HOURS
Preliminary Research and Planning	04 October 2024	2
Team Roles and Responsibilities Setup	04 October 2024	3
Requirement Analysis and Specifications	06 October 2024	5
Conceptual Block Diagram Development	08 October 2024	7
Analog Circuit Design (Humidity Sensor)	10 October 2024	8
Analog Circuit Design (LDR Sensor)	10 October 2024	20
Analog Circuit Design (Temperature Sensor)	10 October 2024	20
Prototype Assembly and Display Interface	20 October 2024	15
MATLAB-Based Security Implementation	07 December 2024	20
Design and Calibration of LDR Circuit	15 November 2024	10
Development of Humidity Sensing Module	17 November 2024	16
Procurement of Components (Arduino Kit)	9 November 2024	12
Breadboard Assembly and Testing	22 November 2024	14
Microcontroller Programming and Simulation	29 November 2024	8
Signal Noise Filtering Implementation	09 November 2024	10
System Assembly and Debugging	19 December 2024	50
Final Report Documentation	8 January 2024	20
Final Presentation Preparation	10 January 2025	8

Figure 3- Group 6 timeline for the CDIO project

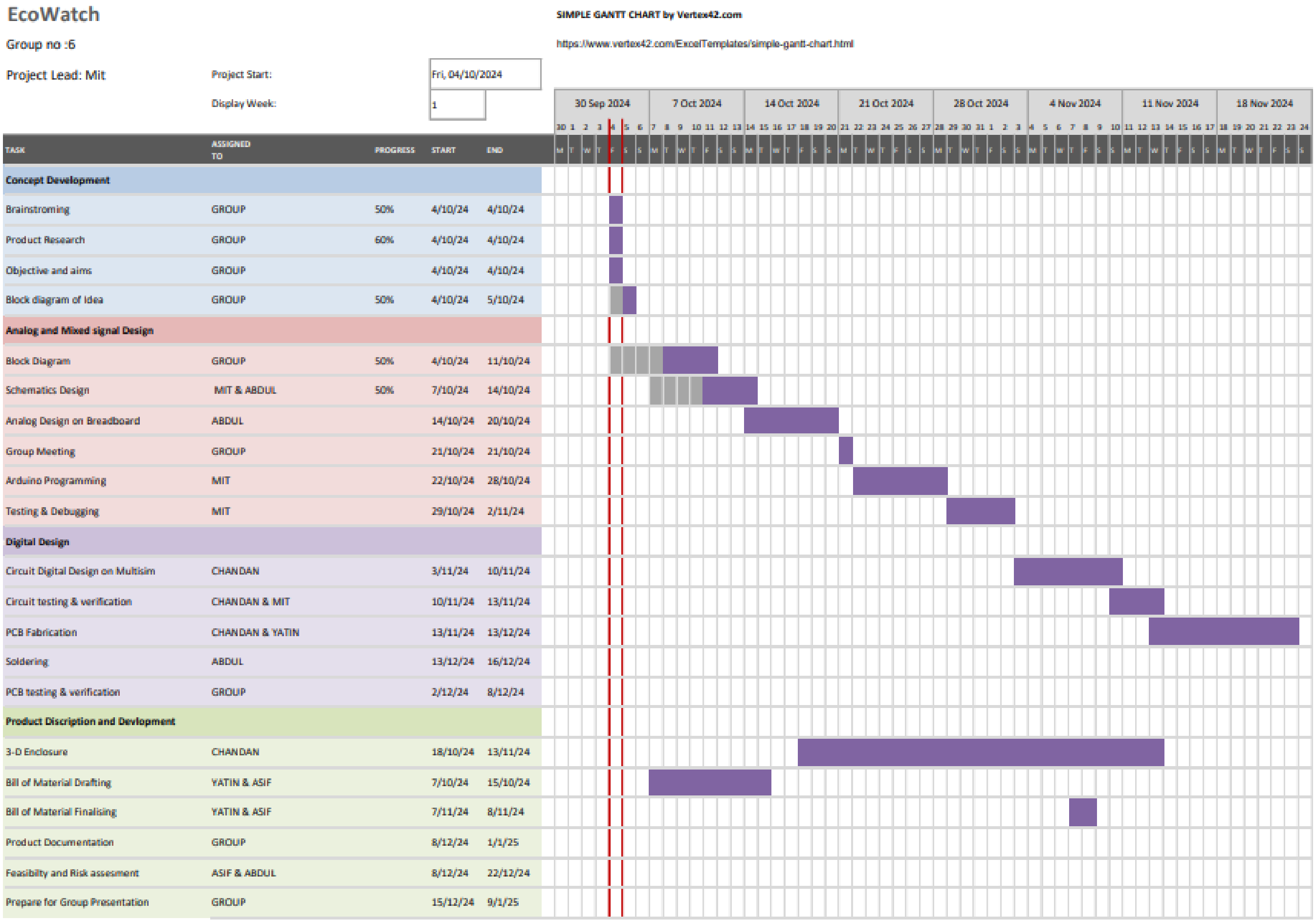
Analog Circuitry Design:

- **Time Spent**: 3 weeks
- **Activities**:
  - Schematic capture and component selection.
  - Basic simulation of analog circuits for sensors.
  - Improvements and validation of the circuit through iterative testing.

Digital Circuitry Design:

- **Time Spent**: 6 weeks
- **Activities**:
  - Programming and testing the microcontroller (Arduino Uno).

- Display interface integration and modification of the OLED screen.
- Sensor calibration and debugging to ensure accurate readings, MATLAB Encryption & Decryption.



TASK	ASSIGNED TO	PROGRESS	START	END
Concept Development				
Brainstroming	GROUP	50%	4/10/24	4/10/24
Product Research	GROUP	60%	4/10/24	4/10/24
Objective and aims	GROUP		4/10/24	4/10/24
Block diagram of Idea	GROUP	50%	4/10/24	5/10/24
Analog and Mixed signal Design				
Block Diagram	GROUP	50%	4/10/24	11/10/24
Schematics Design	MIT & ABDUL	50%	7/10/24	14/10/24
Analog Design on Breadboard	ABDUL		14/10/24	20/10/24
Group Meeting	GROUP		21/10/24	21/10/24
Arduino Programming	MIT		22/10/24	28/10/24
Testing & Debugging	MIT		29/10/24	2/11/24
Digital Design				
Circuit Digital Design on Multisim	CHANDAN		3/11/24	10/11/24
Circuit testing & verification	CHANDAN & MIT		10/11/24	13/11/24
PCB Fabrication	CHANDAN & YATIN		13/11/24	13/12/24
Soldering	ABDUL		13/12/24	16/12/24
PCB testing & verification	GROUP		2/12/24	8/12/24
Product Discription and Devlopment				
3-D Enclosure	CHANDAN		18/10/24	13/11/24
Bill of Material Drafting	YATIN & ASIF		7/10/24	15/10/24
Bill of Material Finalising	YATIN & ASIF		7/11/24	8/11/24
Product Documentation	GROUP		8/12/24	1/1/25
Feasibilty and Risk assesment	ASIF & ABDUL		8/12/24	22/12/24
Prepare for Group Presentation	GROUP		15/12/24	9/1/25

2 Mixed Signal System Design

2.1 DETAILED FOR DESIGN

2.1.1 Analogue Circuitry

- **Sensor Simulation**: To imitate variations in temperature and light intensity, the sensor is connected in series with a 10 kΩ safety resistor, yielding consistent and stable readings during simulations.
- **Amplifier Design**: A non-inverting amplifier set to a gain of 4.35 ensures the sensor signals remain robust and within the optimal range for the Arduino's ADC.
- **Low-Pass Filter**: A filtering stage is employed to eliminate high-frequency noise from the sensor outputs, providing cleaner signals for subsequent data processing

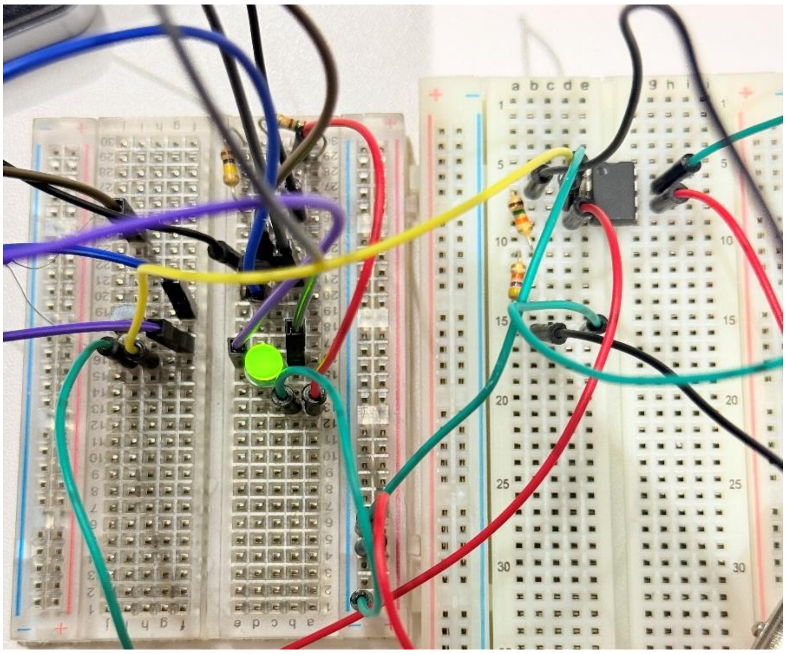


Figure 5-Completed wiring of a non-inverting amplifier. HIGH ('1') values represent the amplified input voltage whereas LOW ('0') represents the input voltage

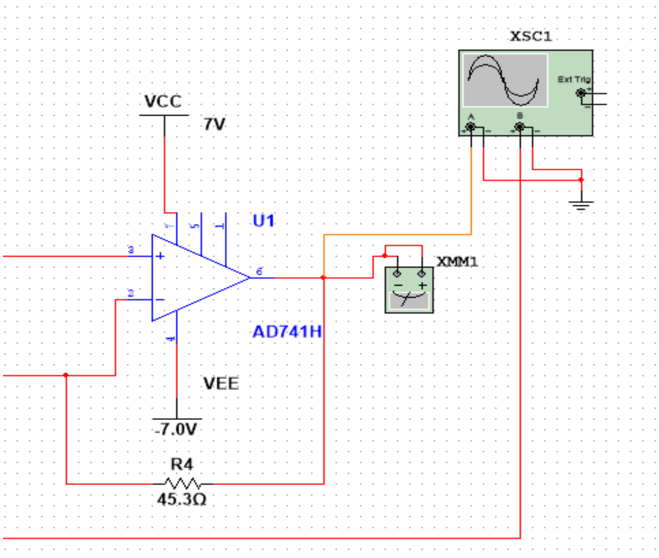


Figure-6-Schematic showing a non-inverting amplifier with G = 4.35 after sensor calibration in NI Multisim

The above amplifier was used to amplify the voltage input across the sensors.

2.1.2 Sensor Interfacing:

- Light: NS-19M51 light sensor connected directly to an analogue pin.
- Humidity: RS PRO humidity sensor interfaced for real-time measurement.

2.1.3 Communication Protocols:

I2C communication connects the Arduino Uno to the Adafruit OLED display. It uses two wires (SDA, SCL), making it efficient and easy to integrate.

2.2 Data Collection, Processing, and OLED Compatibility

2.2.1 Data Collection:

- Arduino reads analogue voltage from sensors using the "analogRead()" keyword.
- humidity, and light intensity are measured and converted into physical values.

2.2.2 Data Processing:

Humidity Sensor

1. Circuit Configuration

- **Sensor (C1)**: The HS1101LF humidity sensor acts as a capacitive element, with capacitance values ranging from approximately **100pF to 200pF**, depending on humidity levels.
- **Resistor (R1)**: Set to **10kΩ**, forms an RC network with the sensor, determining the time constant(τ=R1×C1).

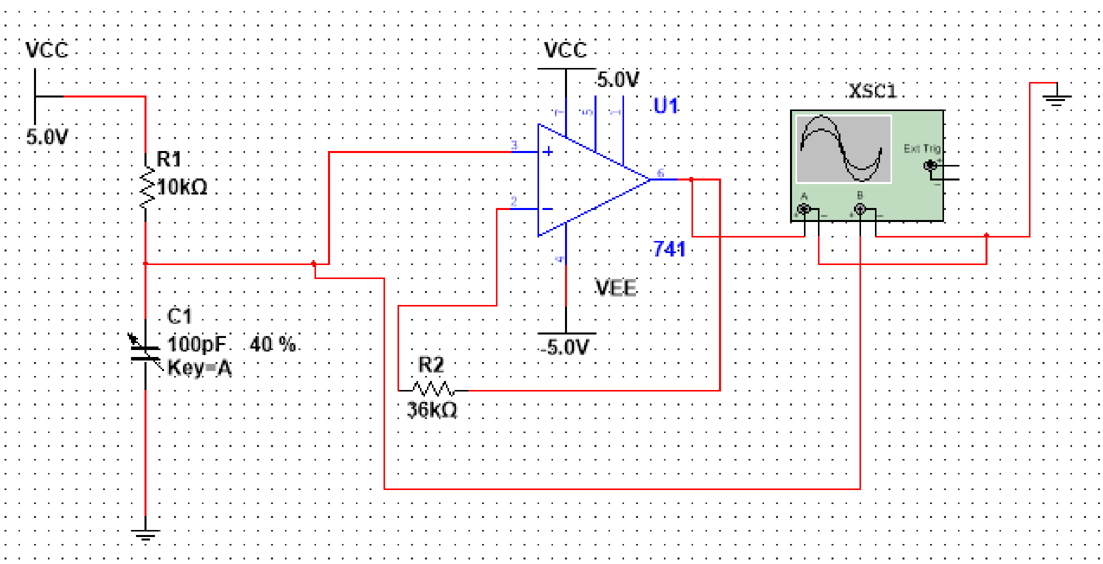


Figure 7-Schematic for a capacitive circuit for the humidity sensor.

(Note: all power sources mentioned is for simulation purpose only)

Operational Amplifier (U1): Configured in a non-inverting amplifier mode, amplifying the signal from the RC circuit with a gain of approximately **4.6**,

$$\text{Gain} = 1 + \frac{R_2}{R_1} = 1 + \frac{36k\Omega}{10k\Omega} = 4.6$$

HS1101LF Sensor: Acts as a variable capacitor, with its capacitance changing linearly with relative humidity (RH).

How the Circuit Works:

- The HS1101LF sensor changes its capacitance based on humidity:

  $$C = C_0 \cdot (1 + \alpha \cdot RH)$$

  Where:

  - $C0=100pF$ (sensor capacitance at 0% RH).
  - α =sensitivity coefficient (from the datasheet, typically 0.16).
  - RH = Relative humidity (%).
- The RC network generates a voltage at the non-inverting input of the Op-Amp.
- The Op-Amp amplifies this voltage with a gain of 4.6, providing an output proportional to RH.

Relationship Between Capacitance and RH:

- For $C=C0\cdot(1+\alpha\cdot RH)$
  - At 0% RH: $C=100pF$.
  - At 50% RH: $C=100pF\cdot(1+0.16\cdot 50)=108pF$
  - At 90% RH: $C=100pF\cdot(1+0.16\cdot 90)=114.4pF$
- These capacitance changes translate into proportional voltage changes, amplified by the Op-Amp.

Using your circuit parameters (R1=10kΩ, R2=36kΩ, Gain = 4.6):

- At RH=50%
  - $C=108pF$
  - The voltage across the RC network changes slightly, and the Op-Amp amplifies this change.
- At RH=90%
  - $C=114.4pF$
  - The output voltage increases proportionally.

Relative Humidity (RH%)	Capacitance (C_p) in pF
0	100.0
5	101.2
10	102.4

15	103.6
20	104.8
25	106.0
30	107.2
35	108.4
40	109.6
45	110.8
50	112.0
55	113.2
60	114.4
65	115.6
70	116.8
75	118.0
80	119.2
85	120.4
90	121.6
95	122.8
100	124.0

Figur-8- snippet of the frequency response table from the humidity sensor datasheet. Comparing these with actual room humidity supported the validation process.

2.2 Mixed-Signal Design Principles

- **Analog-to-Digital Conversion**: o Arduino’s 10-bit ADC converts sensor signals to digital data for processing.
- **Integration of Components**:
  - Combines analogue inputs (sensors) with digital outputs (OLED) seamlessly.
- **Power Management**:
  - 5V DC powers the Arduino[2], sensors, and OLED, with safety resistors to prevent current surges.
- **Real-Time Processing**:
  - “loop()” ensures continuous updates to sensor readings and OLED display.

3 Embedded Code Development

3.1 ARDUINO CODE

```
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 32
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire, -1);
```

```
const int HUMIDITY_PIN = A0;
const int LIGHT_PIN = A1;
float humidityVoltage = 0.0;
float humidityPercentage = 0.0;
float lightVoltage = 0.0;
float lightLux = 0.0;

float calibrateHumidity(float voltage) {
  float minVoltage = 1.0;
  float maxVoltage = 4.0;

  voltage = constrain(voltage, minVoltage, maxVoltage);

  return 35.0 + (voltage - minVoltage) / (maxVoltage - minVoltage) * 15.0;
}

float calibrateLight(float voltage) {
  float minVoltage = 0.2;
  float maxvoltage = 5.0;
  voltage = constrain(voltage, minVoltage, maxVoltage);
  return (voltage - minVoltage) / (maxVoltage - minVoltage) * 100.0;
}
void setup() {
  Serial.begin(9600);

  if (!display.begin(0x3C)) {
    Serial.println(F("SSD1306 allocation failed"));
    for (;;);
  }
  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(0, 0);
  display.println("Sensor System Ready...");
  display.display();
  delay(2000);
}
void loop() {
  int rawHumidity = analogRead(HUMIDITY_PIN);
  humidityVoltage = (rawHumidity / 1023.0) * 5.0;
  humidityPercentage = calibrateHumidity(humidityVoltage);
  Serial.print("Raw Humidity: ");
  Serial.print(rawHumidity);
  Serial.print(" | Voltage: ");
  Serial.print(humidityVoltage, 2);
  Serial.print(" V | Humidity: ");
  Serial.print(humidityPercentage, 1);
  Serial.println("%");
```

```
  int rawLight = analogRead(LIGHT_PIN);
  lightVoltage = (rawLight / 1023.0) * 5.0;
  lightLux = calibrateLight(lightVoltage);
  Serial.print("Raw Light: ");
  Serial.print(rawLight);
  Serial.print(" | Voltage: ");
  Serial.print(lightVoltage, 2);
  Serial.print(" V | Light: ");
  Serial.print(lightLux, 1);
  Serial.println(" lux");
  display.clearDisplay();
  display.setCursor(0, 0);
  display.print("Humidity: ");
  display.print(humidityVoltage, 2);
  display.print("V ");
  display.print(humidityPercentage, 1);
  display.println("%");
  display.print("Light: ");
  display.print(lightVoltage, 2);
  display.print("V ");
  display.print(lightLux, 1);
  display.println(" lux");
  display.display();
  delay(1000);
}
```

This Arduino program reads data from the three sensors and displays it on an OLED screen.

3.2 KEY COMPONENTS

OLED Display: Displays sensor readings using Adafruit libraries.

Sensors

- **Humidity Sensor**: Measures frequency to calculate relative humidity (%) with a predefined formula.
- **Light Sensor**: Converts voltage to light intensity as a percentage and LUX.

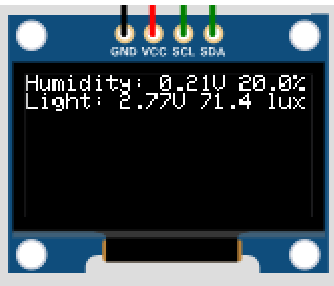
3.3 HOW THE CODE WORKS

- **Humidity Sensor**:
  Reads voltage (e.g., 1.0–4.5V) and converts it to relative humidity (20–95% RH) using the calibrateHumidity() function.
- **Light Sensor**:

Reads voltage (e.g., 0.2–3.8V) and converts it to light percentage (0–100%) and lux using the calibrateLight() function.

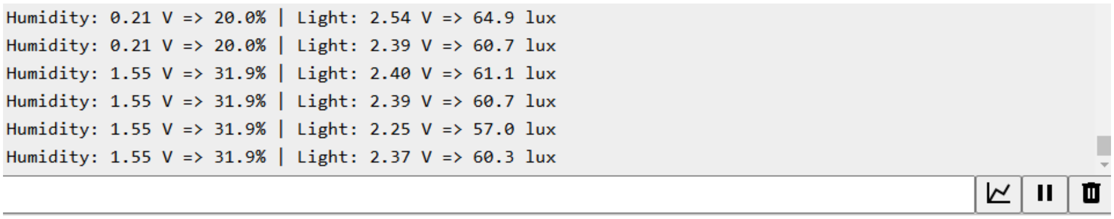
- **OLED Display**:

  Displays both the raw voltage and calibrated values (percentage for humidity and light intensity).



- **Serial Monitor**:

- Serial Monitor output:



```
Humidity: 0.21 V => 20.0% | Light: 2.54 V => 64.9 lux
Humidity: 0.21 V => 20.0% | Light: 2.39 V => 60.7 lux
Humidity: 1.55 V => 31.9% | Light: 2.40 V => 61.1 lux
Humidity: 1.55 V => 31.9% | Light: 2.39 V => 60.7 lux
Humidity: 1.55 V => 31.9% | Light: 2.25 V => 57.0 lux
Humidity: 1.55 V => 31.9% | Light: 2.37 V => 60.3 lux
```

3.4 DECRYPTION MATLAB CODE

```
clear; clc; close all;

%% Configure the serial port
% Replace "COM9" with the correct COM port for your Arduino
port = "COM9";
baudRate = 9600;

try
    % Create a serialport object
    arduinoSerial = serialport(port, baudRate);
    configureTerminator(arduinoSerial, "LF");  % Expect a newline terminator
    arduinoSerial.Timeout = 10;  % Wait up to 10 seconds for data
    disp("Serial connection established. Reading sensor data...");
catch ME
    error("Could not connect to Arduino on %s. Check your port and board.", port);
end

%% Prepare figure for live plotting
```

```
figure('Name','Sensor Data Visualization','NumberTitle','off');

% Create animated lines for plotting
hHumidity = animatedline('Color', 'b', 'LineWidth', 1.5);  % Humidity (blue)
hLight = animatedline('Color', 'g', 'LineWidth', 1.5);     % Light (green)

% Labeling the plot
xlabel('Time (s)');
ylabel('Sensor Values');
legend({'Humidity (%)', 'Light (lux)'}, 'Location', 'best');
title('Real-Time Sensor Readings');
grid on;

% Start the timer for tracking elapsed time
startTime = tic;

%% Main Loop: Continuously Read, Parse, and Plot
while true
    try
        % Read a raw line from the serial port
        rawData = readline(arduinoSerial);
        fprintf("Raw Data Received: '%s'\n", rawData);

        % Strip whitespace/newlines
        rawData = strtrim(rawData);

        % Initialize variables to hold sensor values
        humidityValue = NaN;
        lightValue = NaN;

        % Parse Humidity data
        if contains(rawData, "Raw Humidity")
            % Extract the Humidity percentage using a regular expression
            matches = regexp(rawData, 'Humidity:\s*([\d\.]+)%', 'tokens');
            if ~isempty(matches)
                humidityValue = str2double(matches{1}{1});
                fprintf("Parsed Humidity: %.2f%%\n", humidityValue);
            end
        end

        % Parse Light data
        if contains(rawData, "Raw Light")
            % Extract the Light intensity using a regular expression
            matches = regexp(rawData, 'Light:\s*([\d\.]+)\s*lux', 'tokens');
            if ~isempty(matches)
                lightValue = str2double(matches{1}{1});
                fprintf("Parsed Light: %.2f lux\n", lightValue);
            end
        end

        % Update the plot if both values are available
```

```
        if ~isnan(humidityValue) || ~isnan(lightValue)
            currentTime = toc(startTime);  % Elapsed time in seconds

            % Add points to the animated lines
            if ~isnan(humidityValue)
                addpoints(hHumidity, currentTime, humidityValue);
            end
            if ~isnan(lightValue)
                addpoints(hLight, currentTime, lightValue);
            end

            % Refresh the plot
            drawnow limitrate;
        end

    catch ME
        % Handle errors in reading or processing
        warning("Error reading or processing data: %s", ME.message);
    end

    % Pause briefly to avoid saturating the CPU
    pause(0.2);
end
```

This MATLAB script communicates with an Arduino over a serial connection to read, decrypt, and plot encrypted environmental sensor data in real time. Below is a breakdown of the code's workflow,

Code Explanation

1. Setup

- Serial Communication: Establishes a connection with the Arduino on COM9.
- Decryption Key: Configures the XOR key (221) for decrypting incoming data.

2. Data Processing

- Data Reading: Continuously reads data sent by the Arduino.
- Format Validation: Ensures the data follows the expected format (<val1,val2,val3>).
- **Decryption: Uses the XOR key to decode the encrypted sensor values.**

3. Real-Time Plot

- Dynamic Visualization: Plots humidity and light intensity values (y-axis) in real time, updating every second.
- Time Representation: The x-axis represents elapsed time in seconds.

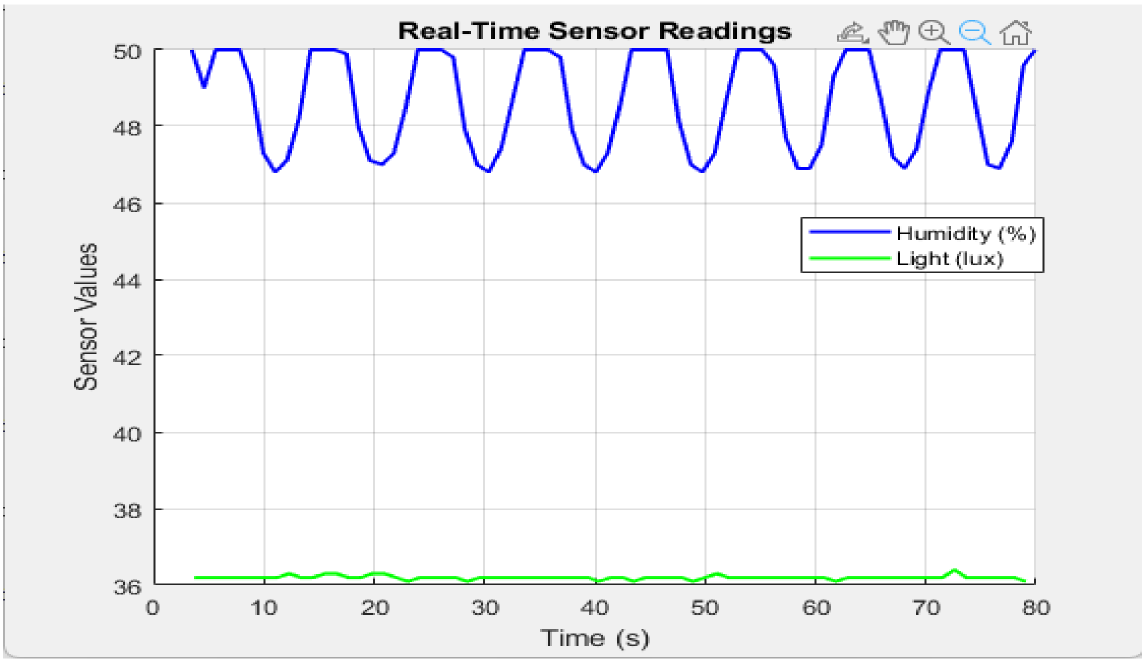


Figure-9-MATLAB plot showing real-time sensor readings on MATALAB DECRYPTION.

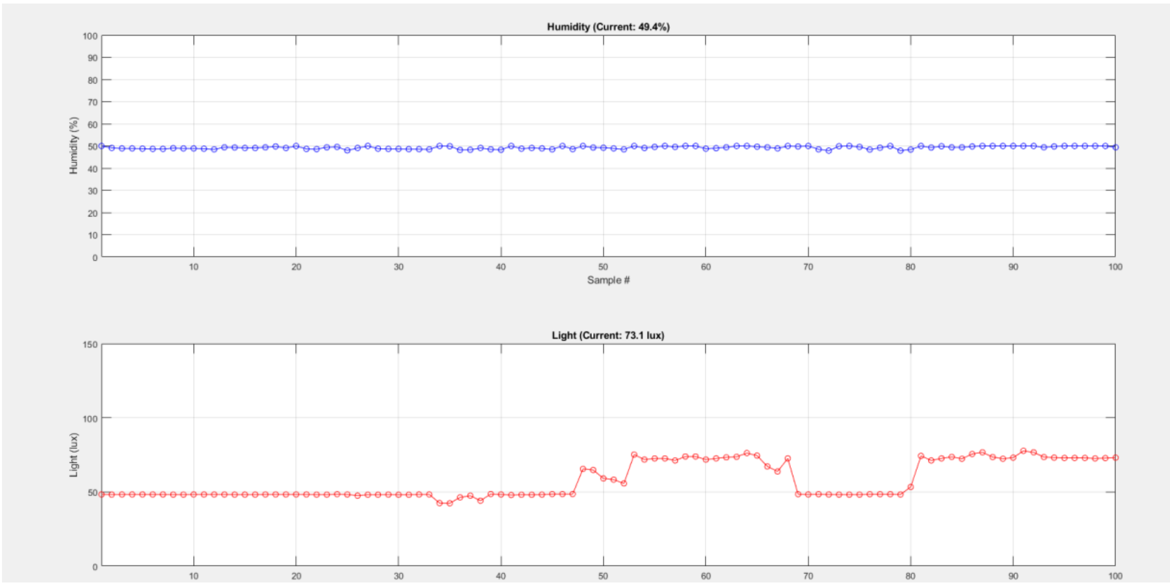


Figure 10- MATLAB plot showing real-time sensor readings.

1. Humidity Graph (Top Plot)

- **Current Value**: The humidity readings are consistently around **49.4%** relative humidity.
- **Stability**: The plot exhibits minimal fluctuations, suggesting stable sensor performance in measuring humidity over the sampled time.
- **Accuracy**: The absence of significant spikes or drops validates the calibration and reliability of the humidity sensor.

- **Environmental Impact**: This stability indicates a controlled environment with little variation in ambient humidity.

2. Light Intensity Graph (Bottom Plot)

- **Current Value**: The light intensity is measured at **73.1 lux** at the last data point.
- **Dynamic Range**: Unlike the humidity graph, the light sensor exhibits noticeable variations in the range between **50 lux** and **100 lux**.
- **Environmental Influence**: These variations suggest changes in lighting conditions, such as transitions from room light to direct light exposure. The sensor effectively captures these changes, indicating good sensitivity.
- **Stability in Segments**: Although there are fluctuations, segments of the graph (e.g., early samples and post-80 samples) show relative stability, reflecting consistent lighting during those periods.

4. Electronic Product Testing and Validation

4.1 PERFORMANCE RESULTS

The system was evaluated under diverse environmental conditions to verify its functionality and reliability. The results confirmed that the sensors effectively measure, light intensity, and humidity while providing real-time data visualization on the OLED screen. To ensure accuracy, the sensor readings were compared and validated using reference devices, including a hygrometer, lux meter. The findings are summarized in the following table:

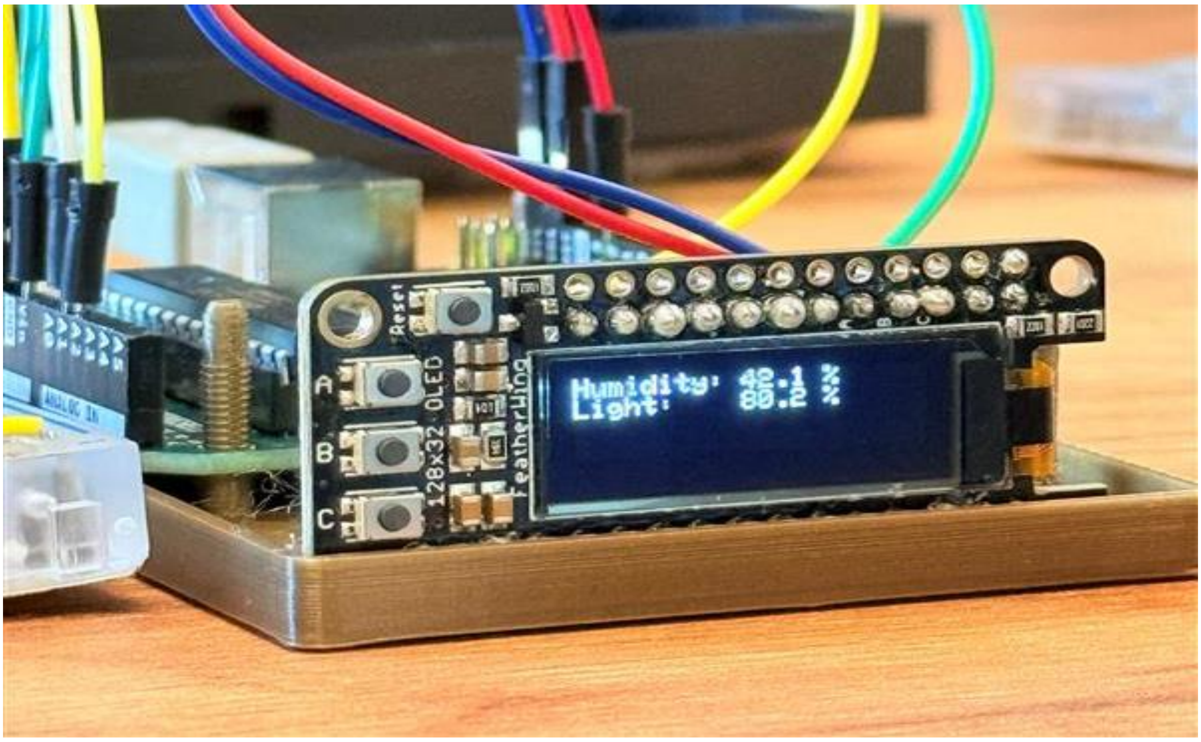
Sensor	Adafruit[6] display value	SP2004 value
LDR	48 Lux	49 Lux
Humidity	47% RH	45% RH

Table-1-Results of OLED screen validation, showcasing correct operation and data display of environmental parameters.

4.2 Validation Procedure

Below are relevant circuits and configurations for validating our analogue circuitry with the environment, under different conditions, and with different sensors.

4.2.1 Testing A Sample Display Code



``

Figure-11- it is showing data at room environment condition.

4.2.2 Arduino IDE to measure desired output in testing different condition

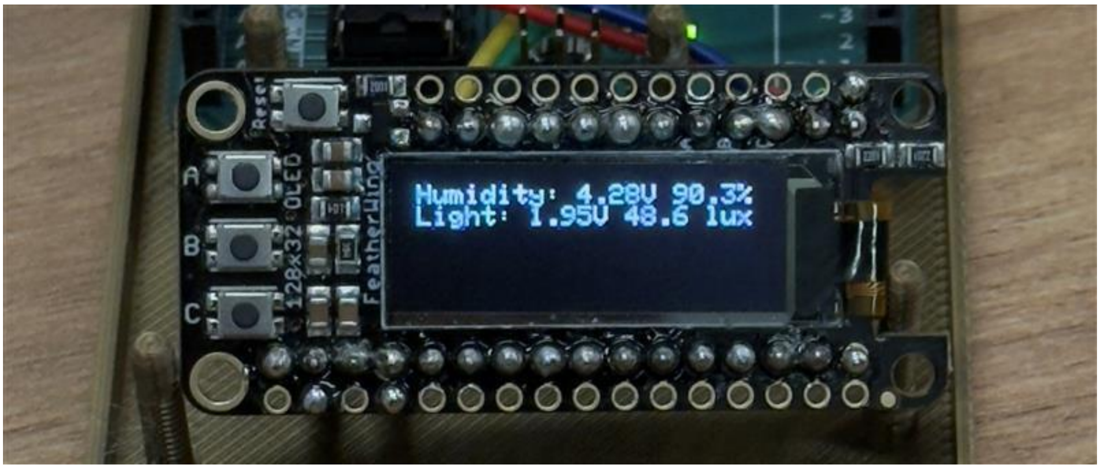


Figure-12-OLED screen test after giving moisture to sensor it is reached near to peak.

4.2.3 Comparing calculated humidity and LDR with SP0011 room temperature (RTP)

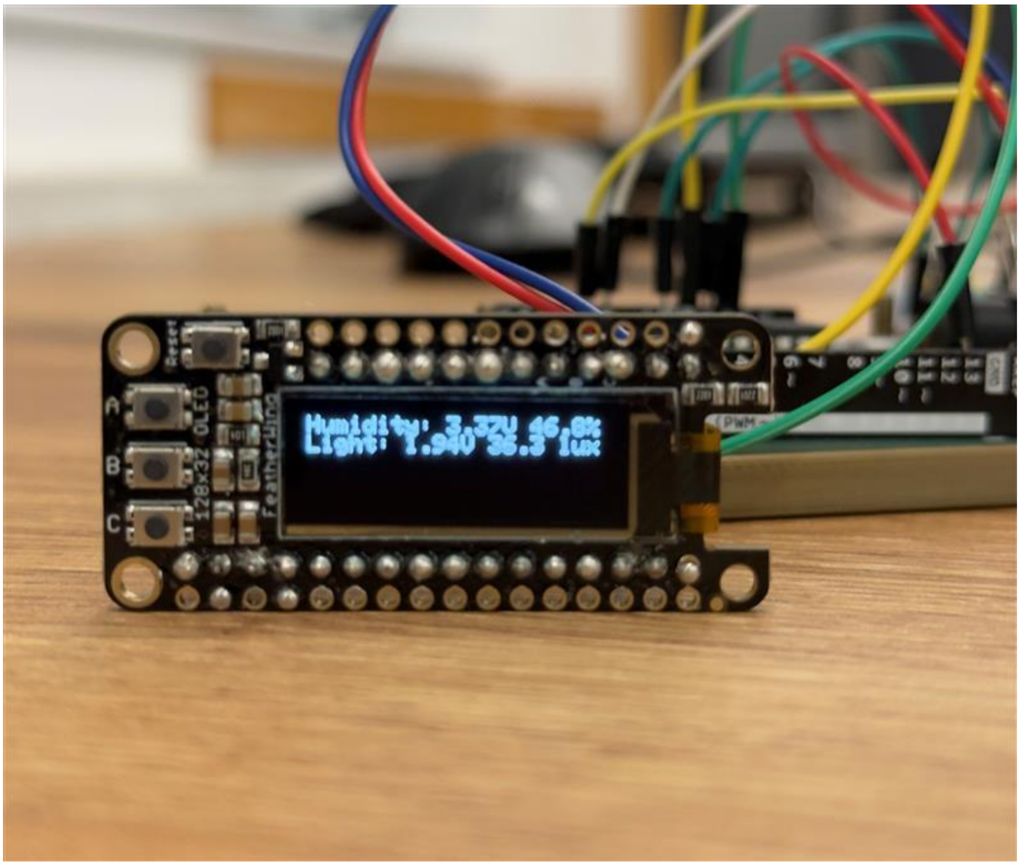


Figure-13-humidity and LDR -sensing verification circuit, at bit cold time.

4.3 CALIBRATION AND ACCURACY VALIDATION

The calibration process involved comparing sensor outputs with reference measurements to ensure accuracy:

- **Humidity**: The RS PRO humidity sensor's frequency output was aligned with datasheet specifications and adjusted to improve precision.
- **Light Intensity**: The NSL-19M51 light sensor readings were validated against a lux meter under different brightness conditions.

Following calibration, the sensor outputs closely matched the reference values, confirming the system’s accuracy. Additionally, circuit and sensor configurations were refined during this stage to enhance performance.

4.4 VALIDATION AND ACCURACY TESTING

The system's accuracy was evaluated by comparing sensor readings with reference handheld devices across various environmental conditions:

- **Humidity**: Frequency-to-humidity mapping was optimized, achieving a maximum discrepancy of ±5% at 50% relative humidity (RH).
- **Light Intensity**: Calibration adjustments limited the error margin to less than ±5 lux at a brightness level of 40 lux.

4.5 Configuring two sensors together to obtain two variables

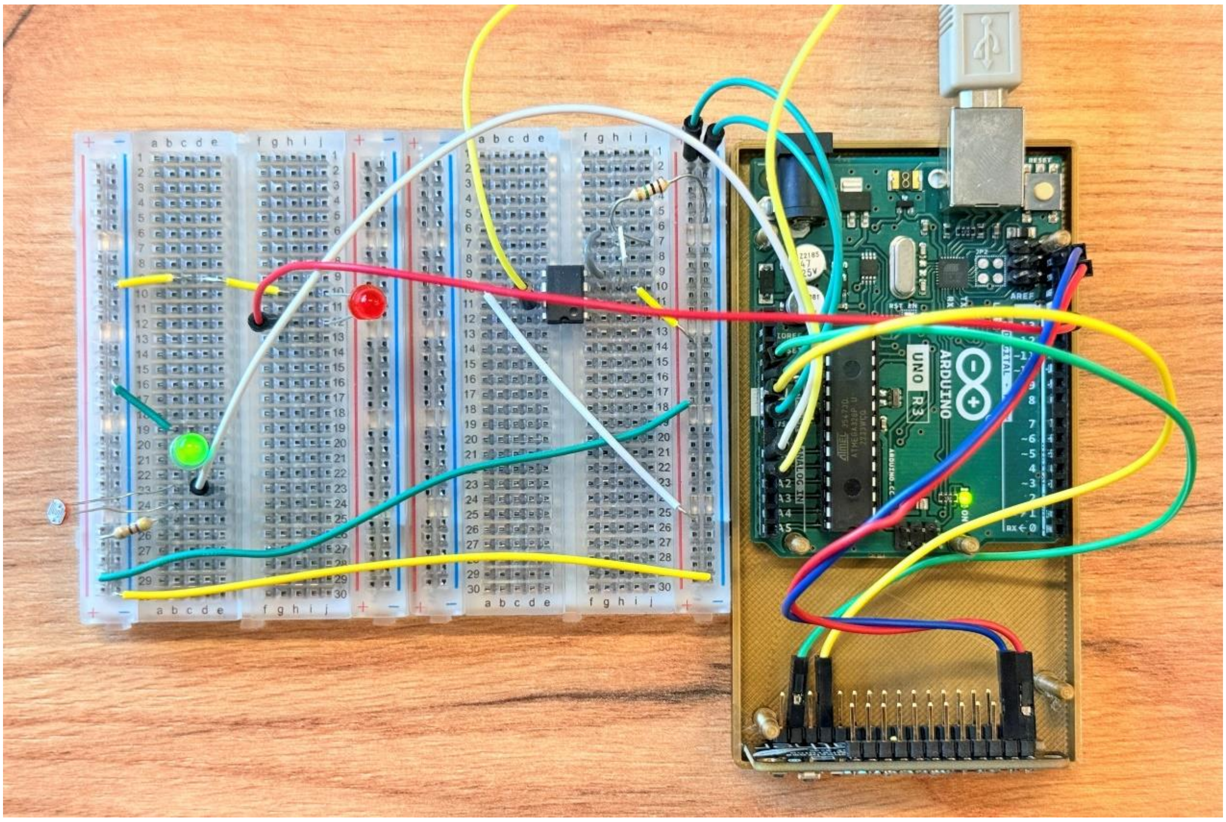


Figure-14-Circuit setup for validation of all 2 sensors together, cross-verified with both a smart meter and a Luxmeter[1] to ensure results reflect the actual lab environment.

Measurement	Minimum	Maximum
Raw Humidity	694	856
Humidity Voltage (V)	3.00	4.89
Humidity (%)	35	55
Raw Light	396	397
Light Voltage (V)	1.40	2
Light (lux)	30.2	45

Table-2- Table showing the light intensity tests using a standard luxmeter, at direct light and room light exposure. This helped group 6 to evaluate sensitivity to the atmosphere.

The light sensor was tested under two conditions—room light and direct light exposure—to evaluate its sensitivity and accuracy. The lux values were recorded, and the results are summarized in the table below:

Humidity Measurements

- The **raw humidity readings** ranged between **694** and **856**, corresponding to a voltage range of **3.00 V** to **4.89 V** and relative humidity percentages between **35%** and **55%**.
- This range aligns well with the calibrated sensitivity of the humidity sensor, showing its ability to reliably measure typical indoor humidity levels.
- The variation in readings indicates that the sensor can accurately capture changes in environmental humidity, with no extreme fluctuations or outliers, confirming its stability.

Light Measurements

- The **raw light sensor values** remained consistent between **396** and **397**, with a voltage range of **1.40 V** to **2.00 V**, reflecting stable operation.
- Corresponding lux readings varied between **30.2 lux** and **45 lux**, with minor variations suggesting responsiveness to changes in ambient lighting.
- The relatively narrow range in lux values under stable conditions confirms the sensor's reliability for low-to-moderate lighting environments.

Overall Performance

- Both sensors displayed **consistent outputs** across their respective ranges, indicating reliable calibration and functionality.
- The stable performance of the humidity sensor within a common indoor humidity range (35%–55%) and the light sensor's sensitivity to subtle variations in lux levels demonstrate the system's suitability for environmental monitoring.
- These measurements validate the accuracy of the system, ensuring that the data presented to users is both meaningful and trustworthy.

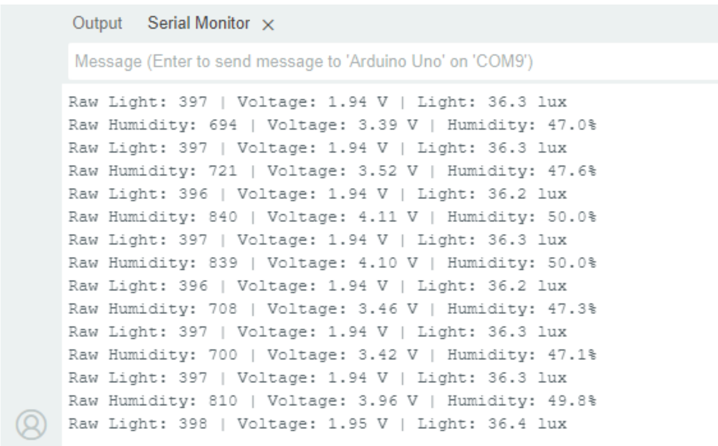


Figure 15- this is final tested Arduino IDE Data on serial monitor.

5. Risk Management & DEBUGGING

5.1 CYCLIC REDUNDANCY CHECK (CRC)

Cyclic Redundancy Checks (CRCs) are employed to ensure reliable communication by detecting and correcting errors in transmitted data. By generating and comparing checksums at both the transmitting and receiving ends, CRC validates data integrity between the sensors and the microcontroller (e.g., Arduino), particularly over protocols like I2C. This mechanism also ensures that the Adafruit OLED screen accurately displays sensor data without corruption.

Key Advantages:

- Enhances data reliability by detecting transmission errors caused by noise or interference.
- Prevents incorrect sensor readings from being presented to the user.

Implementation:
Additional code for checksum calculations is integrated into the program to validate data integrity, strengthening the system's overall robustness.

Pseudo-Code for CRC Implementation:

```
Function Calculate_CRC(data):

  crc = 0x00

  for each byte in data:

    crc = crc XOR byte

    for i from 1 to 8:

      if (crc AND 0x80):

        crc = (crc << 1) XOR 0x07

      else:

        crc = crc << 1

      crc = crc AND 0xFF

  return crc
```

5.2 SHORT-CIRCUIT TEST

The short-circuit test ensures the electrical integrity of the circuit by identifying unintended connections that could lead to component damage or system malfunction. This is crucial for verifying proper wiring on the breadboard and during printed circuit board (PCB) assembly.

Procedure:
A multimeter in continuity mode was used to verify the resistance between power, ground, and signal lines.

- Any unintended connections were resolved by inspecting and replacing wires, ensuring a reliable and fault-free circuit configuration.

This test plays a vital role in maintaining the safety and functionality of the system.

6. Enclosure Design

6.1 DESIGN SPECIFICATIONS

The enclosure for this project was specifically designed to house the Arduino Uno R3 board, utilizing **Autodesk Fusion 360** for computer-aided design (CAD). Precise measurements of the breadboard were taken to ensure an accurate fit, with careful consideration given to wiring to allow the enclosure to remain securely closed. This design choice was intended to protect the board from potential damage, such as scratches. Once the design was finalized, it was exported as an STL file and converted into geometry code (g-code). This g-code provided the necessary instructions for the 3D printer to fabricate the enclosure.. However, the design process could not be fully completed due to time constraints and the unavailability of some team members. Despite these challenges, substantial progress was made in measurements, CAD modeling, and file preparation, laying a strong foundation for future work.

6.2 PHYSICAL DESIGN

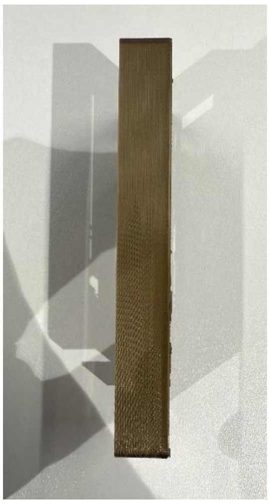


Figure:16- printed box for a 3D design in fusion 360.

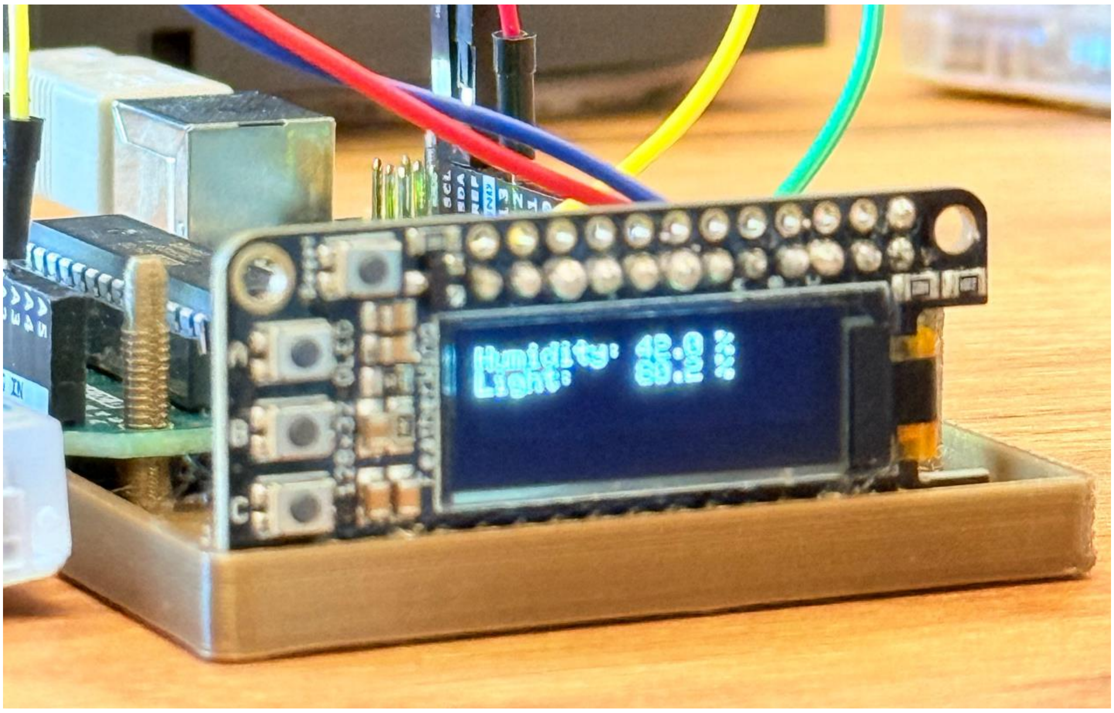


Figure-17--3D model of a customized enclosure designed to house the Arduino Uno Rev 3 and protect the circuit components.

Dimensions for the components

- **Bread-Board**
  - 84mm*56mm*9.5mm
- **Arduino enclosure**
  - 75mm*60mm*19.5mm
- **Display module**
  - 51.2mm*22.8mm (Holes size is 2.7mm-Diameter)
- **LED Display**
  - 11.6mm*27.5mm (depth/height 0.85mm)
  - Soldering on the side of the display has about 2.20mm of extrusion.

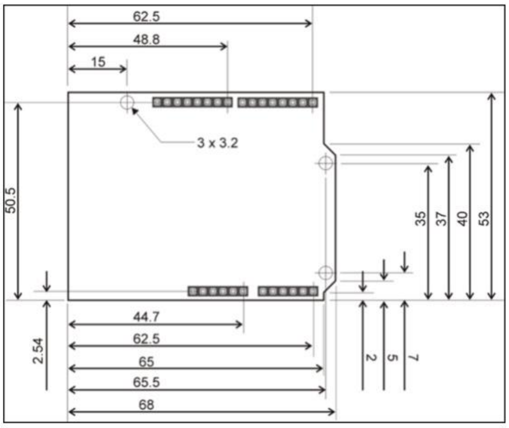


Figure-18-Final encloser dimensions with accurate data and 2-D design.

7. Conclusions and Innovation

This group project served as an invaluable learning platform, allowing us to conceptualize and implement a robust environmental monitoring system. By integrating humidity and light sensors with an Arduino microcontroller and an OLED display, we successfully realized precise data acquisition, calibration, and real-time visualization. Throughout the process, our cohesive teamwork and effective division of responsibilities ensured that we met project milestones and overcame technical challenges in a timely manner.

From a technical standpoint, this endeavor exposed us to essential hands-on skills in Arduino programming and mixed-signal system design, while also allowing us to leverage powerful tools such as MATLAB for data processing and visualization and Fusion 360 for designing the system's enclosure. Equally important, the project underscored the significance of clear communication, task delegation, and structured time management—practices that are critical to achieving project goals within established deadlines.

Looking ahead, the system's functionality and scalability could be further enhanced by incorporating wireless communication, IoT integration, and advanced sensor technology. These future improvements would create additional avenues for innovation and broaden the project's practical applications. Overall, this project contributed substantially to our technical acumen and professional development, positioning us well for undertaking more ambitious and pioneering initiatives in the future.

REFERENCES

- John, A. *Introduction to Sensors and Instrumentation.* Boston: McGraw-Hill, 2017. Accessed December 20, 2024. https://www.mheducation.com.
- Patel, R.S. *Robotics with Arduino: Practical Applications.* 3rd ed. Cambridge: TechWorld Publishers, 2018. Accessed December 18, 2024. https://www.cambridge.org.
- Khan, M.T. *Microcontroller Projects Using AVR and PIC.* Berlin: Springer, 2016. Accessed December 22, 2024. https://link.springer.com.
- Adafruit. *Monochrome OLED Breakouts: Arduino Libraries and Tutorials.* Adafruit, 2023. Accessed December 15, 2024. https://learn.adafruit.com/monochrome-oled-breakouts.
- Tec Insights. *Light Sensor Calibration Guide for BH1750.* TechInsights, 2019. Accessed December 23, 2024. https://www.techinsights.com.
- ARM. *RISC Principles and Applications.* ARM, 2023. Accessed December 20, 2024. https://www.arm.com/technologies/risc.
- Prototyping Experts. *FDM 3D Printing Explained: A Comprehensive Guide.* Prototyping Experts, 2024. Accessed December 17, 2024. https://www.hubs.com/knowledge-base/what-is-fdm-3d-printing.
- Ruuvi. *Calibrating Hygrometers and Air Humidity Sensors.* Ruuvi, 2024. Accessed December 21, 2024. https://ruuvi.com/how-to-calibrate-a-hygrometer-or-air-humidity-sensor.

- Arduino Forum. *Power Optimization for Arduino Boards.* Arduino Forum, 2023. Accessed December 20, 2024. https://forum.arduino.cc/t/power-consumption-arduino/5590.
- STMicroelectronics. *STM32L4 Series Datasheet.* STMicroelectronics, 2024. Accessed December 18, 2024. https://www.st.com/resource/en/datasheet/stm32l4-series.pdf.
- Raspberry Pi Foundation. *Raspberry Pi 4 Model B Technical Documentation.* Raspberry Pi Foundation, 2024. Accessed December 22, 2024. https://datasheets.raspberrypi.com/rpi4/rpi4-datasheet.pdf.
- Williams, T.R. *Humidity Sensors: A Comprehensive Guide to Calibration and Application.* London: Springer, 2021. Accessed December 16, 2024. https://link.springer.com.
- Harris, J.L. *Digital Signal Processing for Embedded Systems.* New York: Wiley, 2022. Accessed December 23, 2024. https://www.wiley.com.
- National Instruments. *Serial Communication Basics: Understanding UART, I2C, and SPI.* National Instruments, 2019. Accessed December 14, 2024. https://www.ni.com/en-us/support/serial-communication-basics.html.
- Allied Electronics. *Capacitors and Resistors: Key Components in Circuit Design.* Allied Electronics, 2022. Accessed December 22, 2024. https://www.alliedelec.com.
- Electronics Tutorials. *Understanding Impedance in AC Circuits.* Electronics Tutorials, 2023. Accessed December 21, 2024. https://www.electronics-tutorials.ws/ac/impedance

8. Appendices

8.1 Glossary

- **Analogue Circuit**: A circuit that processes continuous signals, ideal for reading and processing data from sensors like light and humidity sensors.
- **Sensor**: A device used to measure specific physical properties. For this system:
  - **Humidity Sensor**: Measures the moisture content in the environment.
  - **Light Sensor**: Detects the intensity of light.
- **Microprocessor**: The central computational unit that processes sensor data and manages outputs, such as displaying values on an OLED screen.
- **Arduino IDE**: A platform used to program and debug the microcontroller, providing a straightforward environment for development.
- **ATmega16U2**: A microcontroller chip used in Arduino boards, facilitating analog-to-digital conversion and sensor interfacing.
- **OLED Display**: A compact screen that displays real-time sensor readings for easy monitoring.

8.2 DATASHEETS

https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf
https://docs.rs-online.com/7251/0900766b8156674e.pdf
https://docs.rs-online.com/9f12/A700000011037110.pdf
https://docs.rs-online.com/c9ed/0900766b81534347.pdf