

DT124A: Lab 6- Smart Environment Monitoring Systems

Name: Polkotuwa Walawwe Chathuranga Bandara Polkotuwa

Date:10th of June 2025

Authorship statement:

I have prepared this lab report as a result of our my original work carried out on 10th May 2025 in the Lab. This report has not been submitted previously for any academic or professional purpose**.**

I have completed these lab tasks independently, including implementation, debugging, and observation. For this tasks , I did some research on the internet for better understanding. With the help of the AVR datasheet, I prepared the assembly code and presented it to the lecturer on the same day.

Table of Contents

1. **Introduction** 4

1.1 Aim 4

1.2 Explanation of the Different Sensors 4

2. **Materials and Methods** 5

2.1 Design 5

2.2 Hardware Component List 5

2.3 Software Design 6

2.4 Optimization 8

2.5 Tests 9

3. **Results** 10

3.1 Results from the tests 11

3.2. Flowcharts or other diagrams to show how the prototype works 12

4. **Discussion** 15

5. **References** 16

6. **Appendix**: 16

1. Introduction

1.1 Aim

The aim of this project was to design and build a simple but practical embedded system that can monitor environmental conditions using multiple sensors. The system is intended to help users, such as home gardeners or greenhouse operators, understand the current state of their environment , like soil moisture and temperature and respond accordingly. It not only displays sensor data on an LCD but also takes action, such as turning on a water pump when the soil gets too dry. This project brought together the topics learned throughout the course, including sensor integration, microcontroller programming, user interaction, and system feedback.

1.2 Explanation of the Different Sensors

This system uses three types of sensors, each with a specific role:

- **DHT11 Sensor**: A digital sensor that measures both temperature and humidity. It provides two important readings in one package and is useful for understanding the general air conditions around the plants.
- **Soil Moisture Sensor**: This analog sensor detects how wet or dry the soil is. It outputs a voltage that changes based on the resistance of the soil, which allows the system to estimate moisture levels. It's a key part of the automation system that controls the relay to simulate watering when the soil is too dry.
- **LM35 Temperature Sensor**: A simple analog sensor that provides an accurate reading of ambient temperature. It's helpful for double-checking the temperature reported by the DHT11 and adds redundancy to the system.

2. Methods and Equipments

2.1 Design

The system was designed as a simple and practical smart environment monitor. The main goal was to collect real time data from different environmental sensors and present the information in a clear and organized way using an LCD screen. To make it easy for users, the system uses a single button to switch between different sensor readings. Additionally, a relay was included to simulate an automatic watering system when soil moisture levels drop too low. Two LEDs (green and red) provide visual feedback about the system status.

2.2 Hardware Component List

- **Arduino Uno (ATmega328P)**
- **DHT11 Sensor** – for temperature and humidity readings
- **Soil Moisture Sensor** – to measure how dry or wet the soil is
- **LM35 Temperature Sensor** – for precise ambient temperature monitoring
- **16x2 LCD Display (1602A)** – shows sensor values in a user-friendly format
- **Push Button** – switch between different sensor modes
- **Relay Module** – turns ON/OFF based on soil moisture, simulating a pump
- **Green and Red LEDs** – indicate system status (normal or error)
- **220Ω Resistors** – for current-limiting with LEDs
- **Breadboard and Jumper Wires** – for building the circuit
- **10kΩ Potentiometer** – used to adjust LCD contrast

2.3 Software Design

The software for the Smart Environment Monitor is written in C and compiled using Microchip Studio. Its primary goal is to manage sensor readings, handle user input, update the LCD display, and control the relay based on sensor data, all while ensuring that error conditions are gracefully handled.

Key Components and Flow

- **Initialization:**
  At startup, the code configures the microcontroller's I/O ports, setting the appropriate pins as inputs (for sensors and the push button) or outputs (for the LCD, LEDs, and relay). This initial setup also includes configuring internal pull-ups for the button and setting up the LCD in 4-bit mode, ensuring that the display is ready to show information once the system boots up.

- **Main Loop:**
  The main thing of the program is an infinite loop that continuously performs these actions:

  - **Sensor Readings:** Depending on the current operating mode, the loop calls specific functions to read data from the connected sensors. For instance:

    - A function (e.g., read_dht11()) handles the digital communication with the DHT11 sensor to obtain temperature and humidity values.
    - Another routine reads the analog output of the soil moisture sensor.
    - Yet another function retrieves ambient temperature via the LM35 sensor.

  - **Mode Switching:**
    The push button is placed within the loop. A debouncing mechanism (implemented as a short delay after detecting a press) ensures that only one mode change is registered per button press. Each press cycles the system through the different sensor display modes.

  - **Display Update:**
    After reading the relevant sensor data, the code updates the 16×2 LCD display accordingly. The display is cleared and refreshed with the new information, ensuring that the user always sees the current system status.

  - **Relay Control and Error Handling:**
    The software compares the soil moisture reading against predefined thresholds. If the reading is below the lower threshold, the relay is activated (simulating the turning on of a water pump).

Code Structure

- **Function:**
  The program is structured into clearly defined functions:
  - **initialize_system()** is responsible for setting up hardware configurations.
  - **read_sensors()** (or separate functions like read_dht11(), read_soil(), read_lm35()) get the sensor reading process. This modularity allows to isolate sensor-specific code, which simplifies debugging and future modifications.
  - **update_display(mode, sensorData)** handles the logic for refreshing the LCD based on the current operating mode.
  - **debounce_button()** ensures reliable detection of user input, preventing false triggering from mechanical noise.
- **Control Flow:**
  A simple state-machine approach is used to manage the system's operating mode. Each state corresponds to a different sensor's data being displayed. The state is updated each time the button press is confirmed, and the new state is immediately reflected on the LCD.
- **Interrupts and Timing:**
  The design primarily uses polling for button presses. This helps avoid issues where sensor readings might be delayed, ensuring a smooth user experience.

2.4 Optimization

Several optimization techniques were applied to make the system efficient, responsive, and reliable while using minimal hardware resources.

1. 4-Bit LCD Communication

The LCD is operated in 4-bit mode instead of 8-bit mode. This significantly reduces the number of microcontroller pins used (from 8 to 4 for data lines), freeing up GPIOs for other components such as sensors, the relay, and user input.

2. Minimalist and Efficient ADC Usage

Sensor data from the soil moisture sensor and LM35 temperature sensor is read through the built-in ADC channels A0 and A1. The read_adc() function dynamically sets the channel and initiates the conversion. It uses bit-masking to avoid affecting other bits in the ADMUX register, making it both safe and efficient.

3. Hysteresis in Relay Control

To avoid rapid switching (chattering) of the relay when the soil moisture value fluctuates around the threshold, a hysteresis approach is used:

- Relay turns ON when moisture < 40%
- Relay turns OFF only after moisture > 45%

This small change significantly improves system stability and reduces wear on the relay or connected actuator like a pump.

4. Button Debouncing in Software

The push button is read using a custom debouncing function, which waits 20 milliseconds after detecting a press to confirm it. This helps eliminate false triggers caused by mechanical noise in the button and avoids the need for external components.

5. Efficient Mode Switching with State Cycling

The system uses a simple counter (mode = (mode + 1) % 3) to cycle through display modes. This is a compact and resource-friendly way to switch between multiple states without complex conditional logic or extra variables.

2.5 Tests

To ensure the system functioned correctly and met the lab requirements, various tests were carried out both during and after development shown in *Table 1*. These tests focused on individual components (sensors, button, LCD, relay) and the overall system behavior under both normal and faulty conditions.

Functional Tests

Test Case	Condition	Expected Result
System Startup	Power on the device	LCD shows welcome message, green LED is ON
Button Mode Switching	Press button once	Display cycles to the next sensor mode
Display All Modes	Press button repeatedly	Modes cycle: DHT11 ,Soil Moisture ,LM35
DHT11 Sensor Read	Sensor connected and valid	Temperature and humidity appear on LCD
Soil Moisture Reading	Simulate dry/wet soil	LCD shows corresponding % moisture
LM35 Temperature Read	Apply external heat using hair dryer	LCD updates with approximate ambient temperature
Relay ON Logic	Simulate dry soil (moisture < 40%)	Relay pin goes HIGH, indicating pump ON
Relay OFF Logic	Simulate wet soil (moisture > 45%)	Relay pin goes LOW, indicating pump OFF

Table 1. Types of tests carried out

3. Results

To build the Smart Environment Monitor system, I first gathered all necessary hardware components including an Arduino Uno, a 16x2 LCD display, DHT11 sensor, soil moisture sensor, LM35 temperature sensor, a relay module, push button, red and blue LEDs, resistors, a potentiometer, and jumper wires. The LCD was connected in 4-bit mode to save I/O pins, using PD0 to PD3 for data lines, PD5 for RS, and PD6 for Enable. A 10kΩ potentiometer was used to adjust the LCD contrast, connected between VCC and GND with its center pin to the LCD's V0 (pin 3).

As shown in *figure 2* , the DHT11 sensor was connected with its data pin to PD2, and the soil moisture sensor's analog output was connected to analog pin A0. The LM35 temperature sensor output was connected to analog pin A1. A push button was connected to PD3 with internal pull-up enabled in software, so the button reads HIGH by default and LOW when pressed. For output control, a relay module was connected to PD4, set up as active HIGH to turn ON when the soil is too dry. Two LEDs were used to indicate system status: the blue LED (connected to PD6) lights up during normal operation, and the red LED (connected to PD5) turns on when a sensor error is detected—such as an invalid checksum from the DHT11 sensor.

After wiring the components on a breadboard, I wrote and compiled the code using Microchip Studio. The code was uploaded to the Arduino Uno using an AVR ISP programmer. Upon powering the system, the LCD displays a welcome message and then begins showing sensor data *(shown in figure 3 & figure 4)*. Pressing the button cycles through the three display modes: DHT11 (temperature and humidity), soil moisture percentage, and LM35 temperature. The relay turns ON automatically if soil moisture drops below 40%, and turns OFF again when it exceeds 45%. The system was tested by interacting with each sensor individually—checking how the display responds, verifying relay activation, and observing the error LED behavior when disconnecting a sensor or simulating an invalid reading. The results of the tests were shown in *Table 2*.

3.1 Results from the tests

Test Case	Condition	Expected Result	Outcome
System Startup	Power on the device	LCD shows welcome message, green LED is ON	Pass
Button Mode Switching	Press button once	Display cycles to the next sensor mode	Pass
Display All Modes	Press button repeatedly	Modes cycle: DHT11 ,Soil Moisture ,LM35	Pass
DHT11 Sensor Read	Sensor connected and valid	Temperature and humidity appear on LCD	Pass
Soil Moisture Reading	Simulate dry/wet soil	LCD shows corresponding % moisture	Pass
LM35 Temperature Read	Apply external heat using hair dryer	LCD updates with approximate ambient temperature	Pass
Relay ON Logic	Simulate dry soil (moisture < 40%)	Relay pin goes HIGH, indicating pump ON	Pass
Relay OFF Logic	Simulate wet soil (moisture > 45%)	Relay pin goes LOW, indicating pump OFF	Pass

Table 2. Result of the tests carried out

3.2. Flowcharts or other diagrams to show how the prototype works

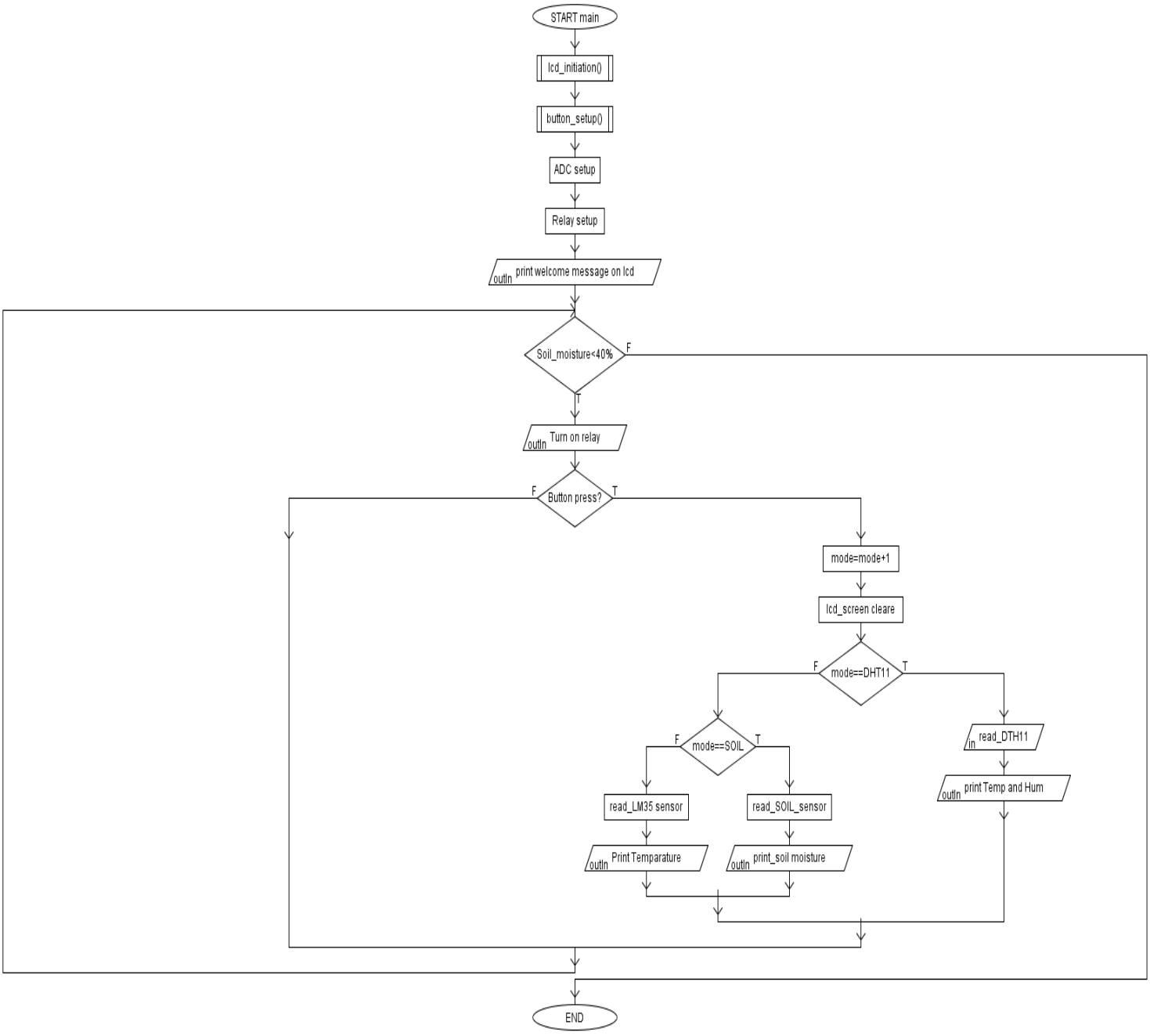

Figure 1 : Flowchart of the main program

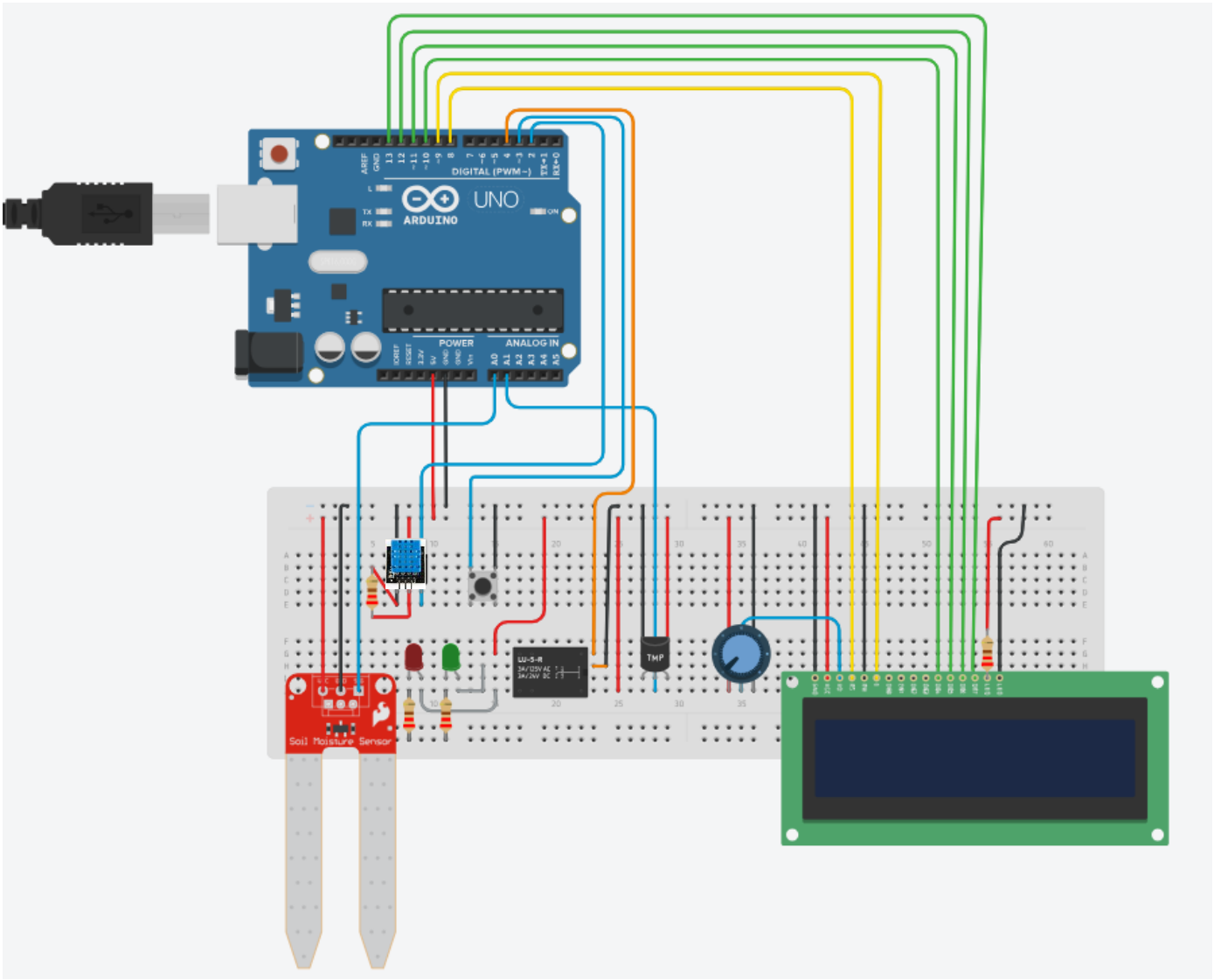

Figure 2: Circuit physical arrangement

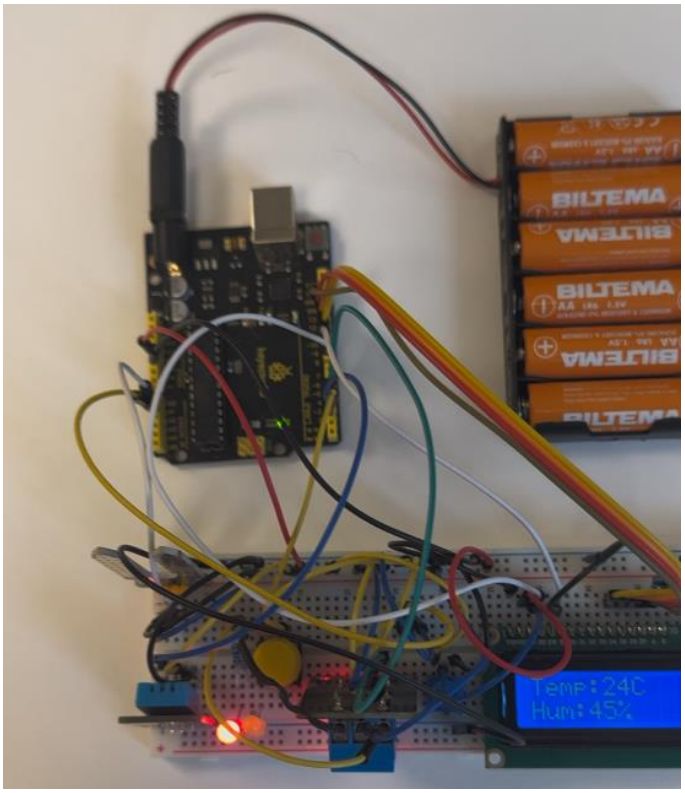

Figure 3: Initial arrangement when power up

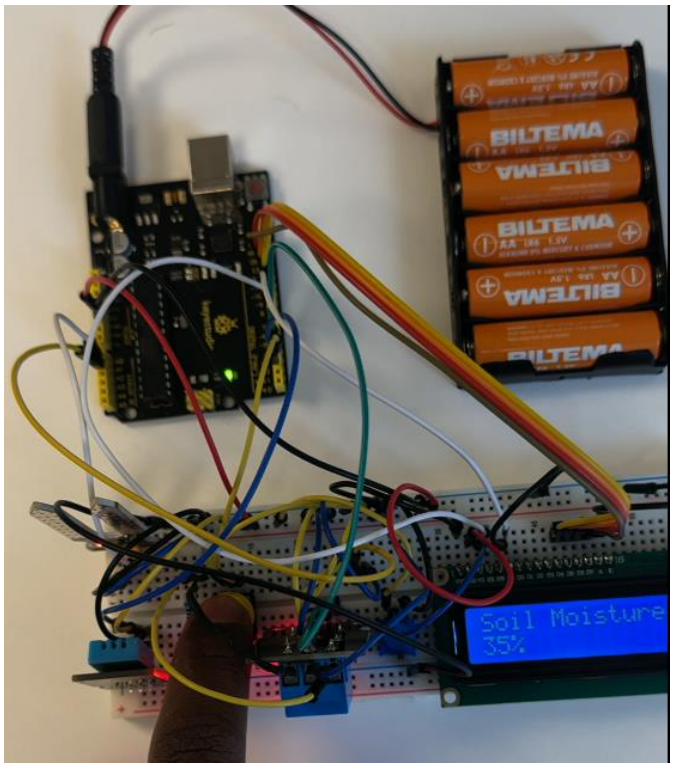

Figure 4: Soil moisture display mode

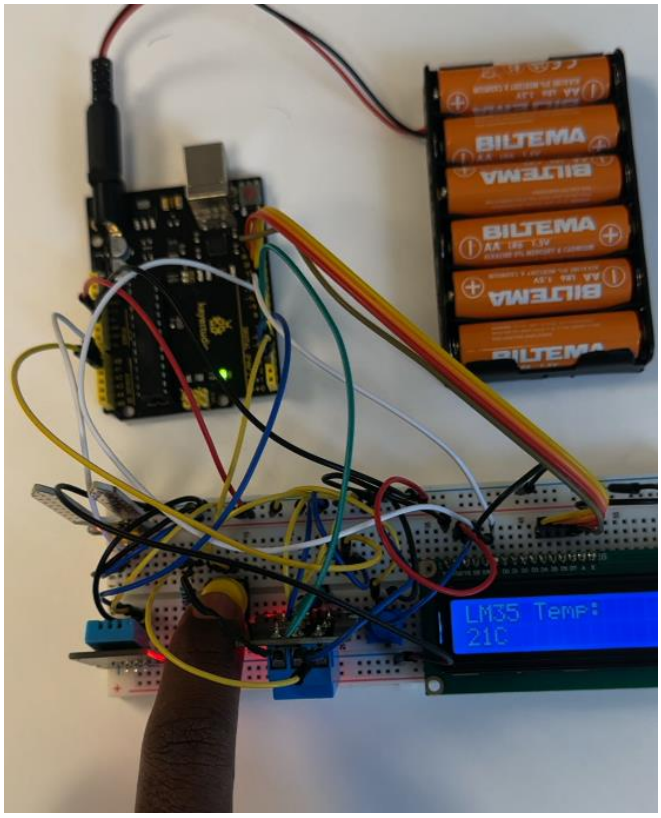

Figure 5: Temperature display mode

4. Discussion

Working on this project was a great way to bring together everything I've learned throughout the course. It wasn't just about connecting components or writing code, it was about designing something that actually works in the real world, where users expect things to be simple, reliable, and informative.

One of the most interesting challenges was figuring out how to switch between different sensor readings with just one button. It it required careful planning in both the code and the hardware to make the user experience feel smooth. Using the LCD in 4-bit mode helped save pins, and controlling it manually without Arduino libraries gave me a deeper understanding of how the display actually works at the register level.

Another key part of the system was the relay, which simulates turning on something like a water pump when the soil is too dry.

The error detection, especially for the DHT11 sensor, taught me a lot about reliability in embedded systems. If a sensor fails or gives bad data, it's important for the system to recognize it and alert the user. That's why I used checksum validation and two LEDs green to show everything is okay, and red to show there's a problem. It might seem small, but it's a big step towards building systems that can be trusted in real-life situations.

5. References

1. Lecture notes on Canvas
2. Lab discussions
3. Tinkercad (circuit connections) https://www.tinkercad.com/ ,
4. AlgoBuild085 (Flow Chart)

5. ATmega328P 8-bit AVR Microcontroller with 32K Bytes In-System Programmable Flash DATASHEET [Internet]. Available from: https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers ATmega328P_Datasheet.pdf

6. Appendix:

1. User Manual – Uploaded as separate file
2. C-code – Uploaded separately

3.C-code explanation video