Kristianstad University Sweden

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

In this final lab project, the goal was to combine everything we've learned in the Embedded Systems course to build a simple and functional embedded system. The idea was to create a working prototype that could read environmental conditions using multiple sensors and show the results in a way that's easy for people to understand—using an LCD display.

To keep the design user-friendly, I added a single push button that lets the user switch between different sensor views, instead of showing all data at once. This makes the system simpler to operate, especially for someone who's not technically trained. The system automatically checks the condition of the environment and controls a relay (which could turn on a water pump) when the soil gets too dry.

To make the system more reliable, I used two LEDs to indicate the status: a Gren LED lights up when everything is working normally, and a Red LED turns on if an error is detected.

Overall, this lab was about turning theoretical knowledge into a hands-on, working embedded system that could actually be useful in real life—like monitoring a garden or greenhouse. It also helped improve skills in circuit design, sensor integration, microcontroller programming, and user interface through LCDs.

2. Materials and Methods

The project was designed as a basic smart environment monitor that uses sensors to collect data from the surrounding environment and displays the information on an LCD screen. The user can switch between sensor readings using a push button, which cycles through the display modes.

The project was developed around the ATmega328P microcontroller on an Arduino Uno board. The main loop in the program constantly reads data from the sensors and updates the LCD accordingly. A relay is also controlled based on the soil moisture level, and status LEDs (green for normal, red for error) provide quick visual feedback.

Materials

Arduino Uno (ATmega328P)

DHT11 Sensor – Measures temperature and humidity

Soil Moisture Sensor – Analog sensor used to detect the moisture level in the soil.

LM35 Temperature Sensor – Analog sensor for precise temperature measurement.

16x2 LCD Display (1602A) – Used to display the current sensor values.

Push Button – Allows the user to cycle through the different sensor readings.

Relay Module – Simulates an actuator (e.g., pump) that turns on when the soil is too dry.

Red and Green LEDs – Indicate system error or normal function.

220Ω Resistors – Used for current-limiting with LEDs.

Breadboard and Jumper Wires

Potentiometer (10kΩ) – Used to adjust contrast on the LCD.

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).

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. 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.

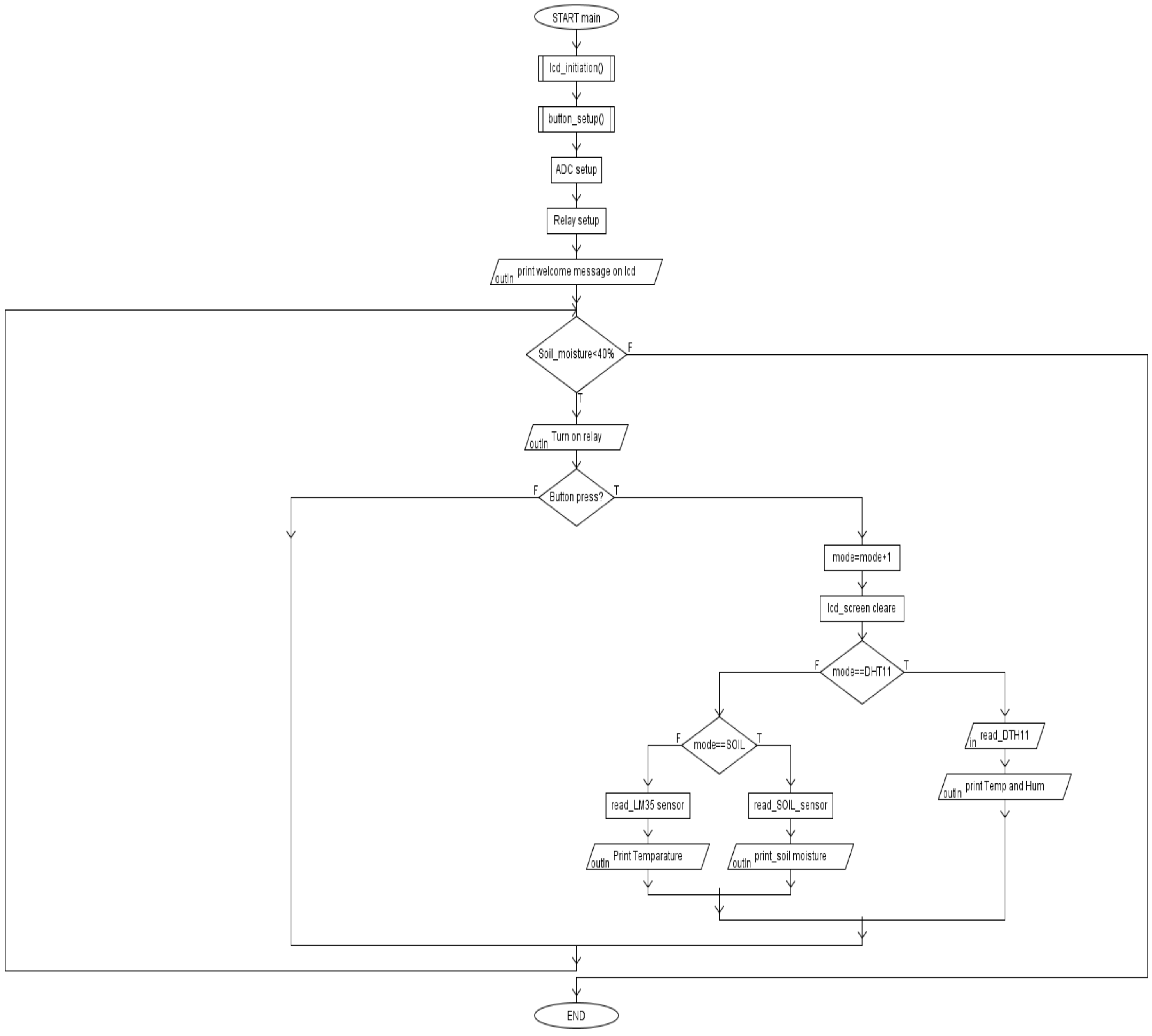

Figure 1 : Flowchart of the main program

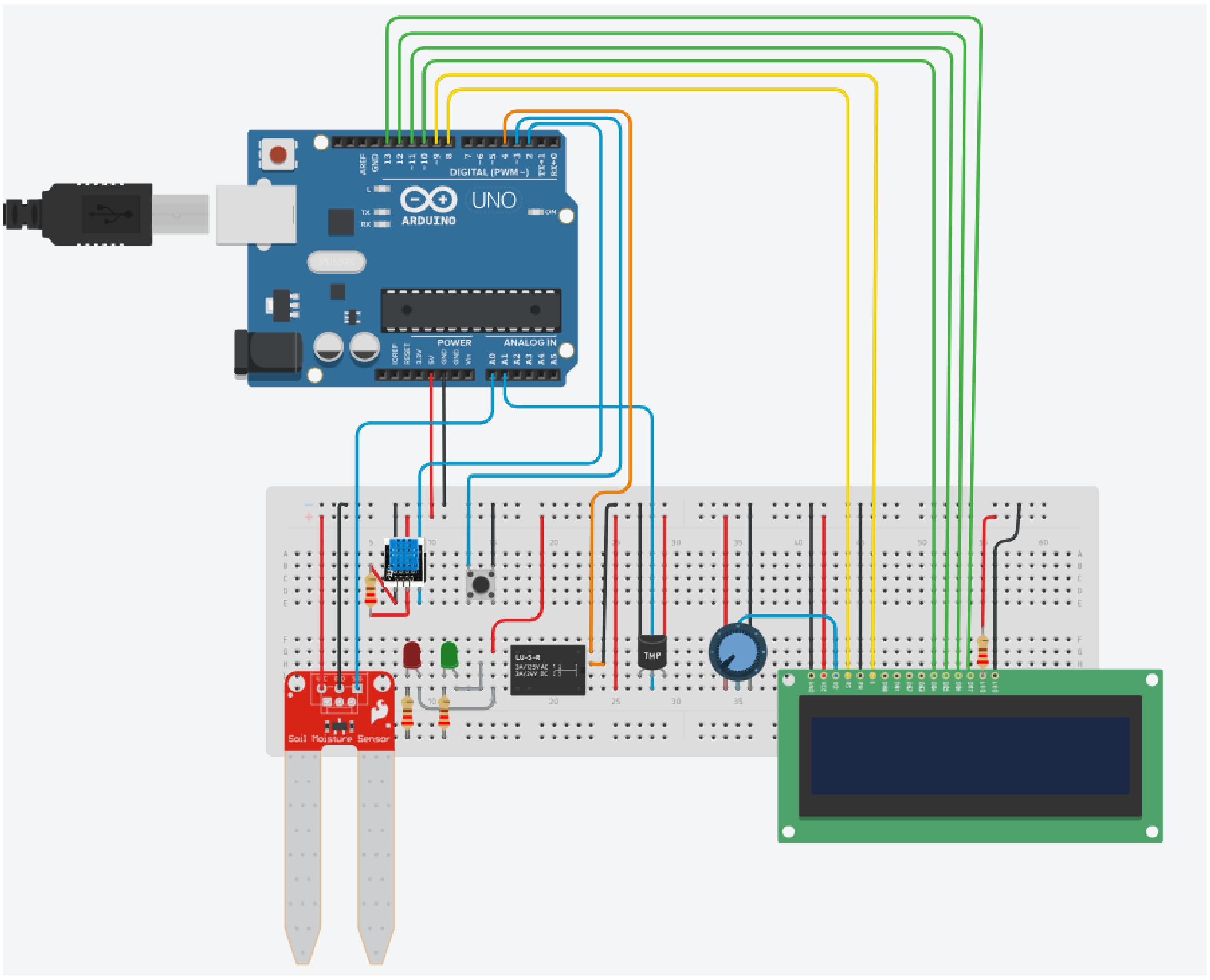

Figure 2: Circuit physical arrangement

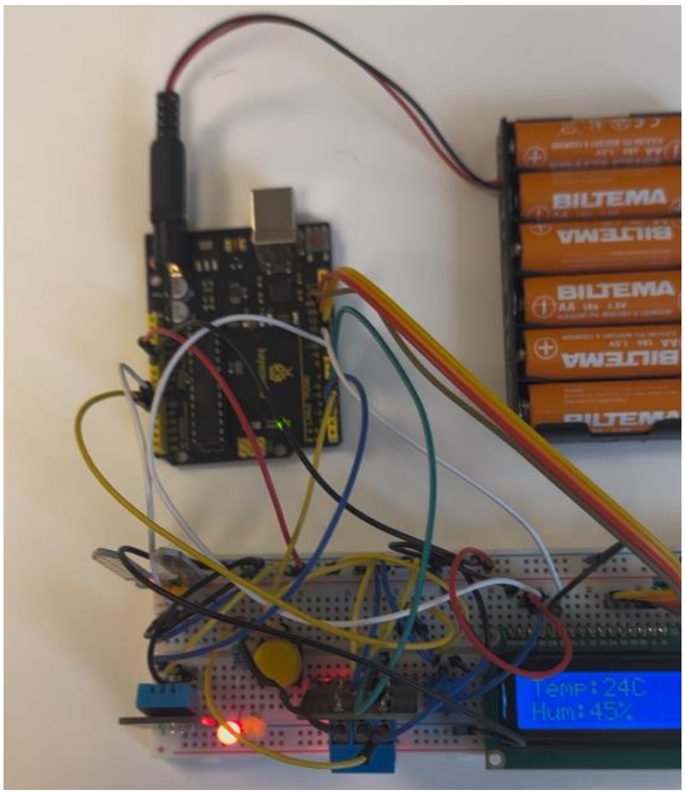

Figure 3: Initial arrangement when power up

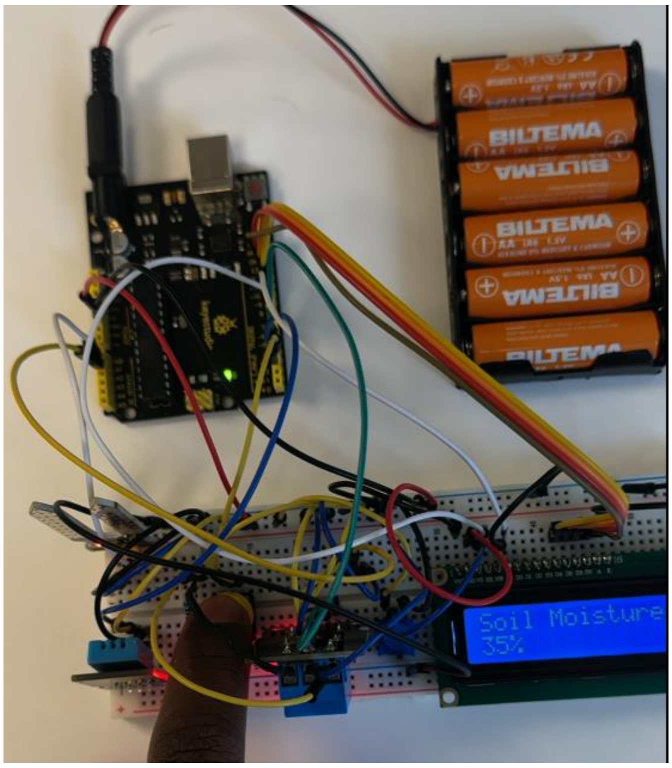

Figure 4: Soil moisture display mode

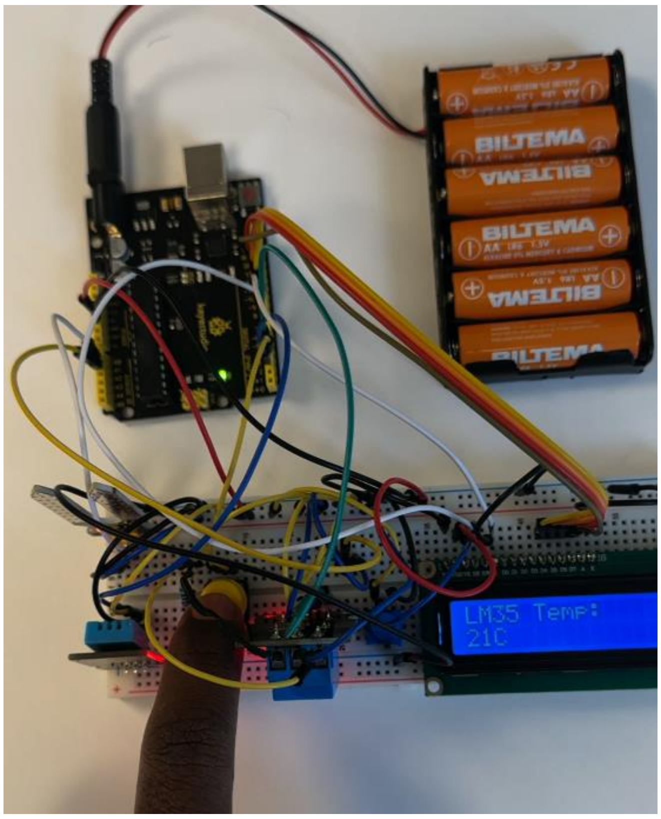

Figure 5: Temperature display mode

2.2 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.

3. 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

4. Appendix:

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

3.C-code explanation video