VAAL UNIVERSITY OF TECHNOLOGY

Inspiring thought. Shaping talent.

Title: design and implement a home automation and security system

Report submitted in fulfilment of the requirements for the Diploma

Diploma in Electrical Engineering

Department of Electrical Engineering

Faculty of Engineering and Technology

LLS MONYELA

Student number: 221 659 595

Diploma in Electrical Engineering

Supervisor: Mr. D.F OJO-SERIKI

Date: 15/05/2025

Declaration

I (full name), declare that this report is my original work and that it has not been presented to any other university or institution for similar or any other degree award. It is being submitted for Engineering programming 3 (EIENP3A) to the Department of Electrical Engineering at the Vaal University of Technology, Vanderbijlpark.

LLS MONYELA 15/05/2025

……………………………………. ………………….

Signature Date

Acknowledgement

224004921	Khoza B
221529667	Ayissa Obama EC
221551220	Mejeni M
224071777	Juingubo S

Dedication

To my loving family that has been my rock of inspiration during my academic journey and Mr. Dipo for going an extra mile for us as his students, your efforts are seen and appreciated. Truly grateful and blessed.

TABLE OF CONTENTS

Acronyms & Abbreviations

ldr	Light Dependent Resistor
pir	Passive Infrared
MYSQL	My Structured Query Language
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
json	JavaScript Object Notation
Iot	Internet of things

Chapter 1: Introduction and purpose of the study

1.1 Introduction

A home automation and security system it is a network of devices and sensors that work together to provide comfort, convenience, and safety for your home and loved ones.it integrate automation and security technologies to control, monitor and protect. This technical report explores the design and implementation of such system.

1.2. Background

Technology is becoming more integrated into everyday life, with smart home systems playing a growing role in improving living standards. our system makes it possible to control lights, appliances, security systems, and environmental conditions without manual effort. These systems aim to save time, reduce energy use, and make homes safer and more comfortable. As more people seek affordable and efficient ways to improve their living conditions home automation and security systems have become not just a luxury, but a practical and accessible solution.

This project focuses on designing and building a home automation and security system using the microcontroller. The system uses multiple sensors to monitor key environmental conditions inside a home.

This study is essential because it provide a reliable, low-cost solution that combines both automation and security, two major needs in modern homes. Many existing systems in the market are expensive or require professional installation. By using open-source tools and affordable components, this project shows that a functional system can be built by students or homeowners with basic technical skills which is one of our goal is a team.

Rahman et al. (2020) developed a basic smart home system using a NodeMCU board and a mobile app. However, their design lacked environmental data logging and was limited to appliance control. Another project by Kumar and Saini (2021) used a Raspberry Pi for home automation, including temperature and motion detection. While effective, it had a higher cost and was more complex than necessary for basic automation.

Our system improves upon those designs by using a cost-effective microcontroller and adding real-time data monitoring. The system also stores historical data, which can be used to improve automation rules or track environmental changes over time.

There is a strong need for home automation and security systems that are easy to use, affordable, and work well. This project combines electronics, programming, networking, and web design to build a system that can be used in real life. The project presented in this study aims to address these needs by exploring how basic automation and monitoring functions can be achieved in a way that is both effective and affordable.

Figure 1: Home automation and security system example.

1.3 Problem statement

Many households and residential spaces struggle to keep track of indoor conditions such as temperature, lighting, motion, and humidity. The lack of a system that provides timely updates or environmental awareness limits the user’s ability to react quickly. This results in discomfort, energy waste, and reduced security in the home.

The people most affected working individuals, families or homeowners who are away from home during the day or for extended periods. These users need a way to stay informed about their home environment but often lack a practical and affordable means to do so.

This issue mainly affects indoor living environments such as student accommodations, family homes, and small apartments. These locations often lack accessible and affordable systems to monitor changes in environmental conditions in real time.

The problem exists continuously but becomes more noticeable during times of travel, work hours, night-time, or seasonal changes that affect indoor temperature, light levels, and air quality. It also becomes critical during unexpected events such as intrusions or electrical interruptions.

This problem matters because it affects both safety and comfort. Without timely feedback or environmental awareness, users may face energy loss, overheating, intrusion risks, or health discomfort. The absence of basic home monitoring limits the ability to act quickly, especially when users are not physically present.

A practical solution is to develop an automated home monitoring system that can track indoor environmental changes and present that information in a simple, accessible way to the user. This system should allow users to observe trends and make informed decisions, improving both comfort and awareness in daily living. It should be reliable and usable even for those without technical backgrounds.

1.4 Significance of the study

This study is crucial because it explores how home automation and security systems can enhance safety, convenience, and efficiency for homeowners. As smart technology advances, combine automation with security measures ensures better protection against violation, emergencies, and everyday risks.

By developing an advanced home automation and security system, my research fills gap in current solutions by incorporating real-time monitoring, and automated responses to security threats. It offers insights into the most effective technologies, such as motion, light, temperature and humidity sensors and database contributing to safer and smarter homes.

This research helps us as upcoming engineers to understand how to create efficient, cost- effective systems that improve daily and provide peace of mind for users.

1.5. Objectives

- To obtain real-time environmental data from temperature (Dallas), light (LDR), motion (PIR), and humidity (MH-series) sensors and store it in a structured MySQL database for future retrieval and analysis.
- To develop a Wi-Fi-enabled data communication process that collects sensor values via the ESP32 microcontroller and transmits them to a Python based backend for real-time display on an interactive html web interface.
- To identify hardware limitations, potential signal interference, and environmental factors that may affect the accuracy and consistency of sensor readings during live operation.
- To establish a responsive, secure, and user-friendly home automation and monitoring platform that integrates sensor data, user access control, and system alerts in one functional environment.
- To determine the responsiveness and accuracy of sensor data under varying environmental conditions such as changes in light, temperature, humidity, and motion activity.

1.6. Project plan

Step 1: The purpose of this project is to design and implement a home automation and security system that collects and monitors environmental data in real time. The system aims to improve home safety, comfort, and efficiency by tracking temperature, light intensity, motion, and humidity. The information is collected using sensors and presented to the user through a secure, user-friendly web interface. This project addresses the need for low-cost, easy-to-use smart home solutions, particularly in homes where commercial systems are unaffordable.

Step 2: To achieve the overall goal, the following specific tasks were identified

- Connect the Dallas Temperature**,** LDR (Light Dependent Resistor)**,** PIR (Passive Infrared)**,** and MH-Series Humidity sensors to the ESP32 WeMo’s D1 R32 microcontroller board.
- Write and upload Arduino ide code that reads values from all sensors.
- Test and calibrate sensors to ensure they detect temperature, light, motion, and humidity levels reliably.
- Configure the Wi-Fi module on the ESP32 to send sensor data wirelessly to a local MySQL database.
- Design a structured database schema to store and organize sensor readings efficiently for retrieval and analysis.
- Verify that the system inserts live data into the database in real time without data loss.

- Develop a web-based interface using Python, HTML, CSS, and JSON to present the collected data in a clear and accessible format. Step 3: research method

The research method supporting is online resources, including

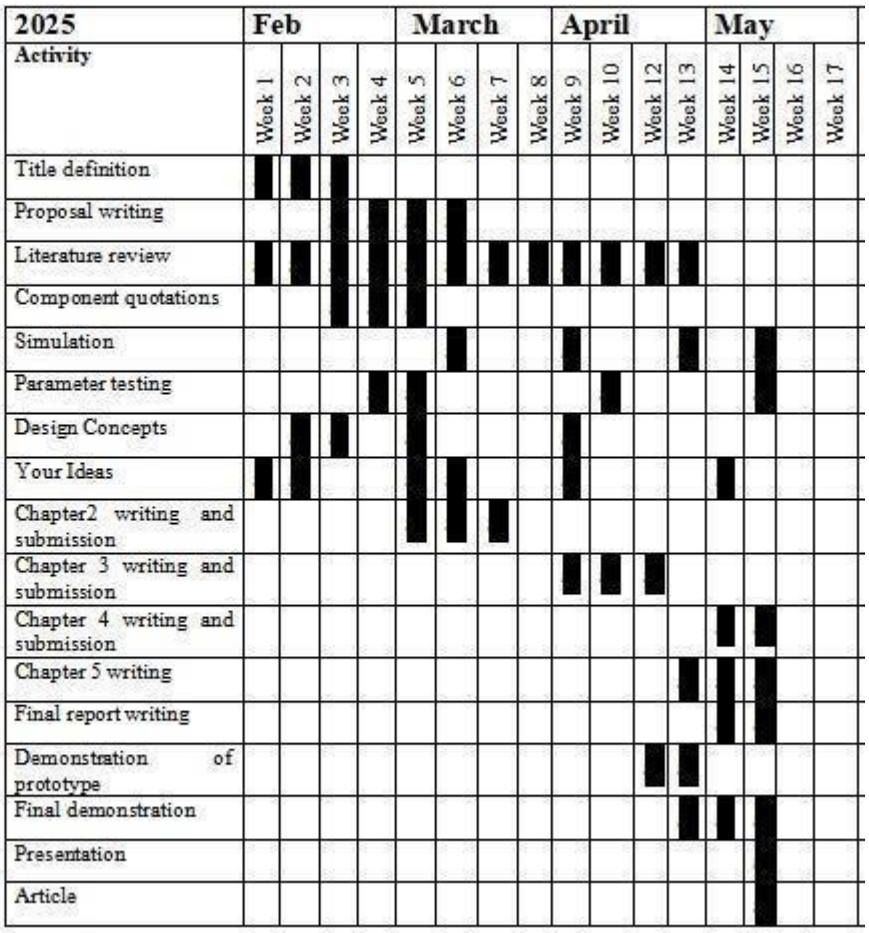
- Technical datasheets for each sensor and the ESP32 board
- Case studies on IoT and smart home development
- MySQL and Flask (Python) integration guides
- Arduino programming tutorials

1.7 budget table

Descrip tion	Cost
Esp32 microcontroller	R120
Temperature sensor	R135.00
Infra- red	R30.00
Humidity sensor	R37.00
Ldr	R2.00 X2
Breadboard	R30
Resistors x 5	R2.90 X5
Cables	R25X3
Total	**R448.4**

Table 1: budget plan for home automation and security system

1.8Time Schedule



2025	Feb				March				April				May			
Activity	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 12	Week 13	Week 14	Week 15	Week 16	Week 17
Title definition	■	■	■													
Proposal writing			■	■	■	■										
Literature review	■	■	■	■	■	■	■	■	■	■	■	■				
Component quotations			■	■	■											
Simulation						■			■			■		■		
Parameter testing				■	■					■				■		
Design Concepts		■	■		■				■							
Your Ideas	■	■			■	■			■				■			
Chapter2 writing and submission					■	■	■									
Chapter 3 writing and submission									■	■	■					
Chapter 4 writing and submission													■	■		
Chapter 5 writing												■	■	■		
Final report writing													■	■		
Demonstration of prototype											■	■				
Final demonstration												■	■	■		
Presentation														■		
Article														■		

Table 2: Project Time Schedule for home automation

1.9 Limitations

Limited sensor accuracy:
The sensors used (Dallas for temperature, MH series for humidity, and ldrs for light) are low cost components. Their accuracy is limited compared to industrial-grade sensors. This may lead to slight variations in readings, especially under extreme environmental conditions.

Security features are basic:
The use of passive infrared sensors only detects motion. It does not differentiate between types of motion (human, animal, object), which could lead to false alarms. There is also no camera or audio feature for verification. No mobile app or voice control:

The user interface is limited to a basic web page. There is no mobile app or voice assistant integration. This reduces ease of use for some users.

Power supply dependency:
The ESP32 and sensors rely on a stable power source. There is no backup power system such as battery, so if the power goes off, the system stops working. This affects reliability, especially for security functions.

1.10 Conclusion

This chapter introduced the background, purpose, and objectives of the home automation and security system project. It outlined the main problem, which is the lack of a simple and affordable system to monitor and control home conditions such as temperature, humidity, light, and motion. Key limitations that may affect the performance and expansion of the system were highlighted.

Chapter 2 will cover the research design and methodology. It will also include a description of the software and hardware tools used in the study.

Chapter 2: Research Design and Methodology

2.1 Introduction

This chapter explains how the design of the home automation and security system was developed. The goal of the project is to build a working system that can read and display real-time temperature, humidity, light levels, and detect motion using affordable components. The system also aims to improve security and comfort in the home. our planning process started by identifying all the functional requirements, such as data collection, storage, and display.

The desired outcome is to produce a fully working prototype that updates sensor values in real time, responds to motion with alerts, and controls curtain movement based on light intensity. This chapter will explain the development steps, wiring methods, and the logic used to achieve this functionality.

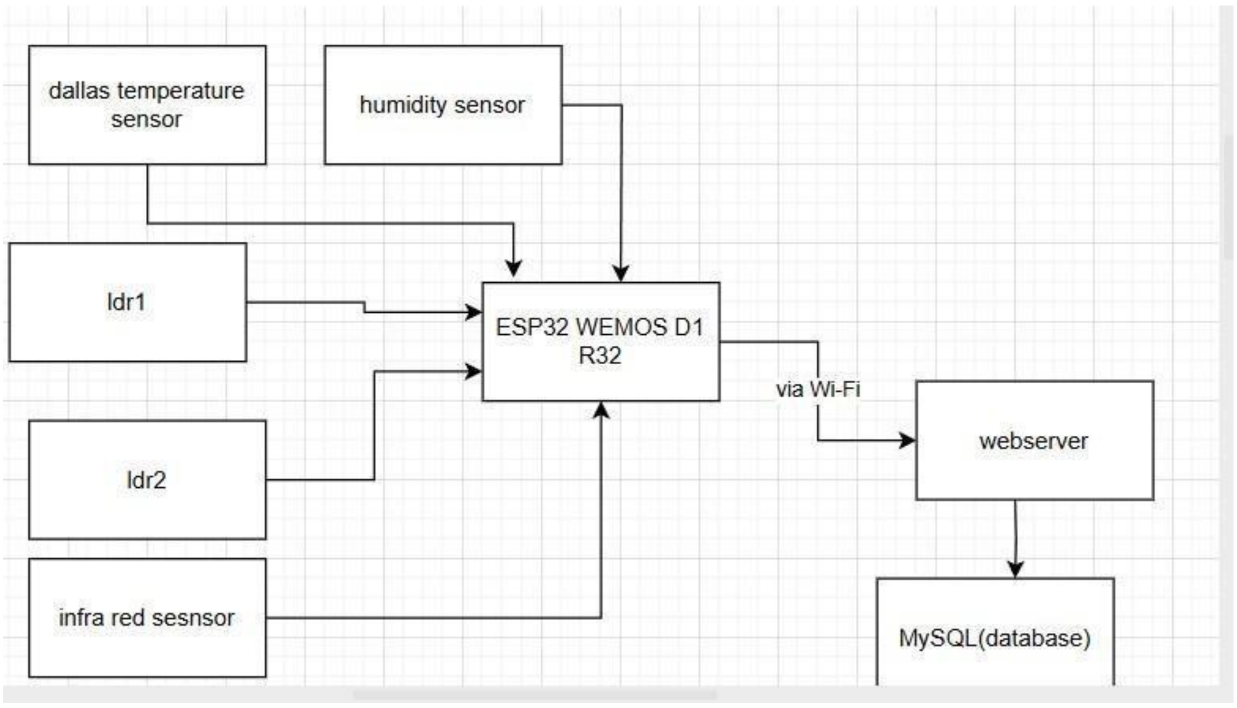


Figure 2: Block diagram of home automation and security system

2.2 Scope of work

The purpose and function of each block will be explained under this section

2.2.1 Dallas temperature sensor

This sensor sends temperature readings to the ESP32 using a digital signal. It operates over a 1Wire protocol, which allows easy connection with the microcontroller and ensures accurate measurements. It is used to measure house temperature. Easy to integrate with ESP32, making it reliable for indoor monitoring in a home, it is also water resistant.

Figure3: ds18b20 temperature sensor

2.2.2 Humidity sensor

It provides an analog or digital signal to the ESP32. The microcontroller reads this value and logs it into the system for real-time display and storage. It monitors humidity level in the house.it Offers reasonable accuracy and stability for indoor conditions, low power usage and fast response.

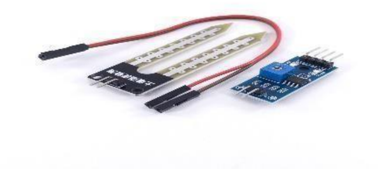


Figure4: humidity sensor

2.2.3Ldr sensor

These sensors change resistance based on light intensity. The ESP32 reads the analog values and decides whether to open or close curtains automatically, depending on the light level. It detects the amount of light in the house or near the window. it simple to use, require minimal circuitry and Ideal for basic light detection, which is enough for automated curtain or lighting control.

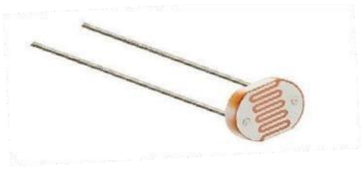


Figure5: ldr sensor

2.2.4Infra red sensor

It senses infrared radiation from a moving body. When motion is detected, it sends a digital signal to the ESP32, which can then trigger an alert or store the event for monitoring. We use it to detect human movement for security purposes.it is widely used in home automation and security system and we choose for its low power consumption ability.

Figure6: infra -red sensor

2.2.5 ESP32 WeMo's D1 R32

The ESP32 receives data from all sensors, processes the inputs, and sends the information to a MySQL database through WiFi.it is the brain of the system. it Has built-in Wi-Fi and enough GPIO pins for multiple sensors and it is Affordable, powerful, and well-supported by libraries and documentation.

Figure7: esp32

2.2.6MySQL(database)

Receives structured data from ESP32 via Wi-Fi. Acts as backend storage. Stores data from sensors for monitoring and logging. It keeps track of the data and the timestamps used to record the date.
It is easy to connect to via Python and it is a free, open-source, and widely used.

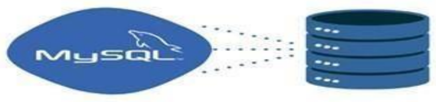


Figure8: MySQL database illustration

2.2.7 webserver

The webserver pulls live sensor values from the database then uses python to prepare the data. Html and css are used to Displays data on a web pages. Json is used to updates values dynamically and allows the user to monitor the system in real time through any web browser.

python
HTML
JS
CSS
Figure9: webserver illustration

2.3 conclusion

These tools were selected because they are reliable, easy to implement, and suitable for real-time applications like our system. Together, they make the system user-friendly and ensure that the data collected from the sensors is accessible from any browser within the network. Python handles backend logic and database communication, html creates the page layout, css improves the look and feel of the interface, and JSON ensures smooth data transfer between backend and frontend. The technologies used were each explained based on how they contribute to displaying real-time sensor data from the database.

The next chapter will focus on the hardware design of the system. It will cover the connection of all physical components, including sensors, the ESP32 microcontroller, and power supply setup. Wiring layouts, pin configurations, and safety considerations will also be explained.

Chapter 3: hardware design

3.1 introduction

This chapter focus on the hardware design and set up of the hardware used in the home automation and security system. This section aims to show how each sensor and component is connected to the ESP32 WeMo's D1 R32 microcontroller. Proper planning is crucial to ensure that the system operates reliably and all components are connected without conflict.

In this chapter I will be explaining the connection method for each sensor, as well as the communication link to the server and database. This section will also describe how the ESP32 handles input signals, processes data, and sends it over Wi-Fi, forming the base of the entire automation system.

3.2hardware design

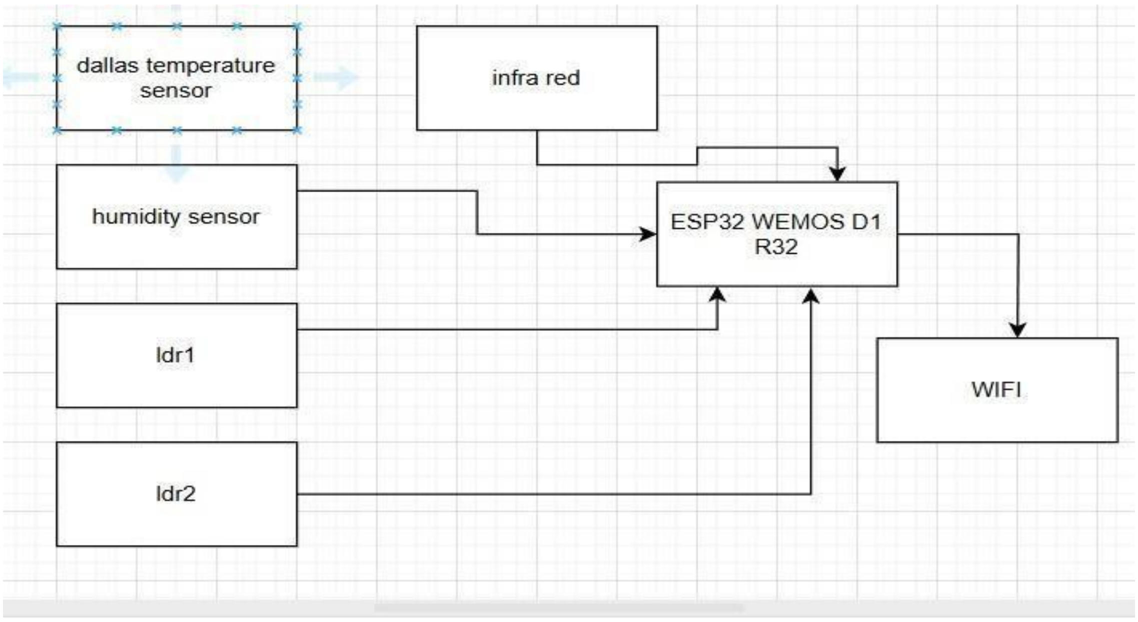


Figure10: block diagram of a hardware design

3.2.1electronic circuit design

3.2.1.1ESP32

The ESP32 WeMo's D1 R32 is a dual-core microcontroller with built-in Wi-Fi and Bluetooth. It is suitable for IoT applications due to its ability to handle multiple sensor inputs and send data wirelessly (WeMo's, 2023). This is the brain of the system. It is powered via USB (5V) and internal 3.3V regulator powers logic.it consist of both digital and analog pins. The esp32 reads all sensor data through analog and digital pins and sends the collected data via Wi-Fi to a web server using HTTP requests, the web server processes the data and stores it into a MySQL database and a web page built with Python, HTML, CSS, and JSON retrieves and displays the stored data for the user in real time. Tools used are Arduino IDE, breadboard and jumper wires for flexible testing and serial monitor for debugging.

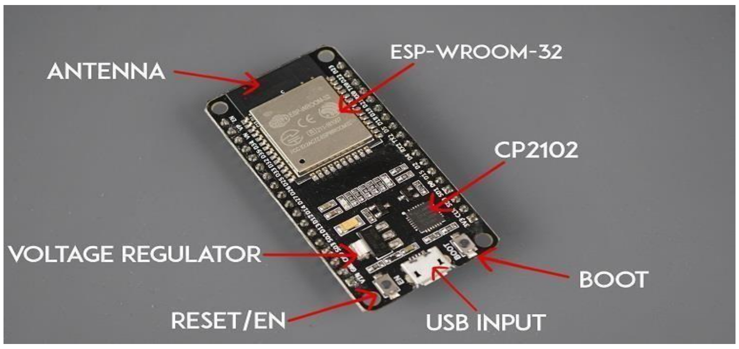
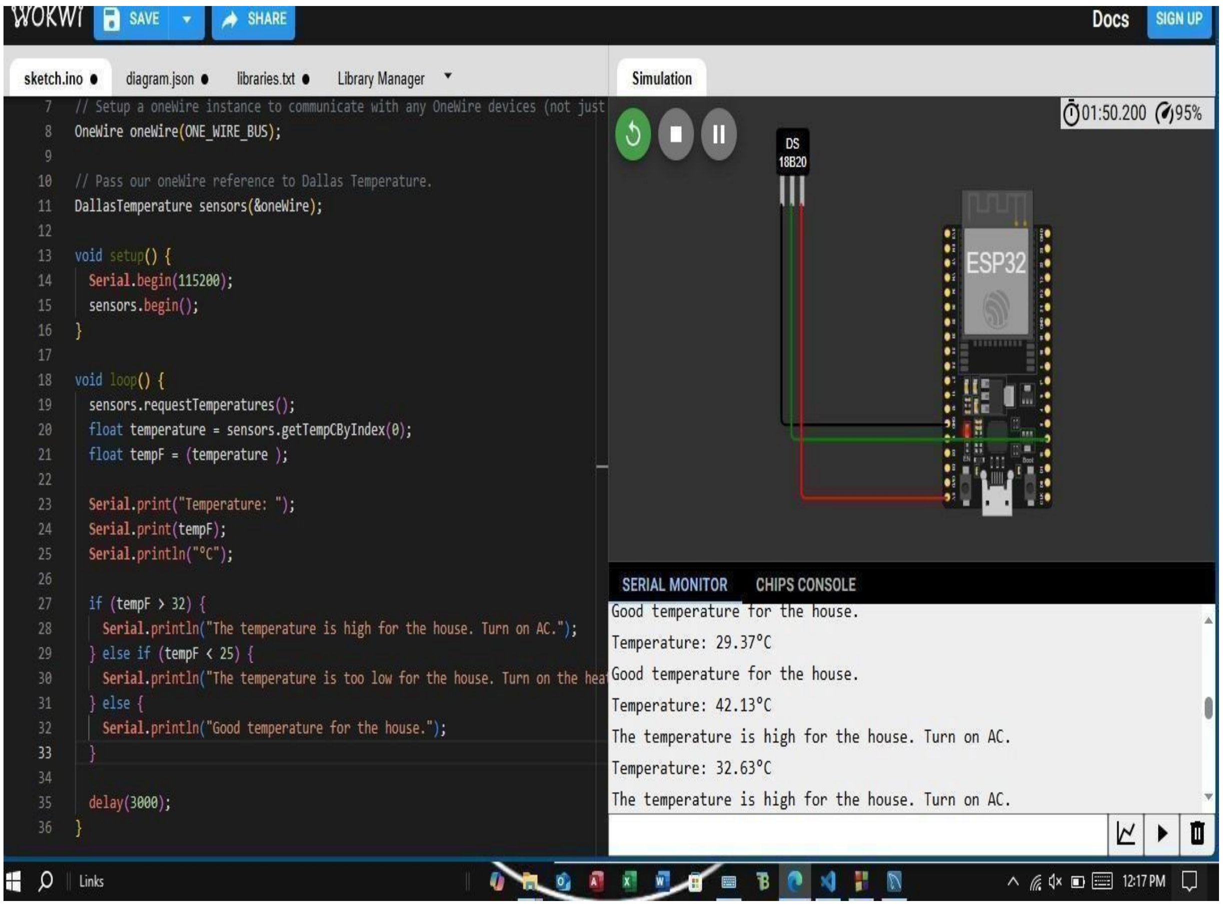


Figure11: fully labelled esp32

3.2.1.2 Ds18b20 Dallas temperature sensor

The DS18B20 temperature sensor operates using a 1-Wire digital communication protocol and requires an external 4.7kΩ pull-up resistor between the data line and power to function properly (Dallas Semiconductor, 2023). Since the resistor affect the line's rise time this means $\tau = R \times C$ where R=4.7Kohms because a balance is needed to ensure fast signal recovery without drawing too much current and C is the parasitic capacitance of the line. The method used with this sensor are Arduino ide which included installing onewire.h and dallas libraries, serial monitor to view temperature readings and Breadboard prototyping.



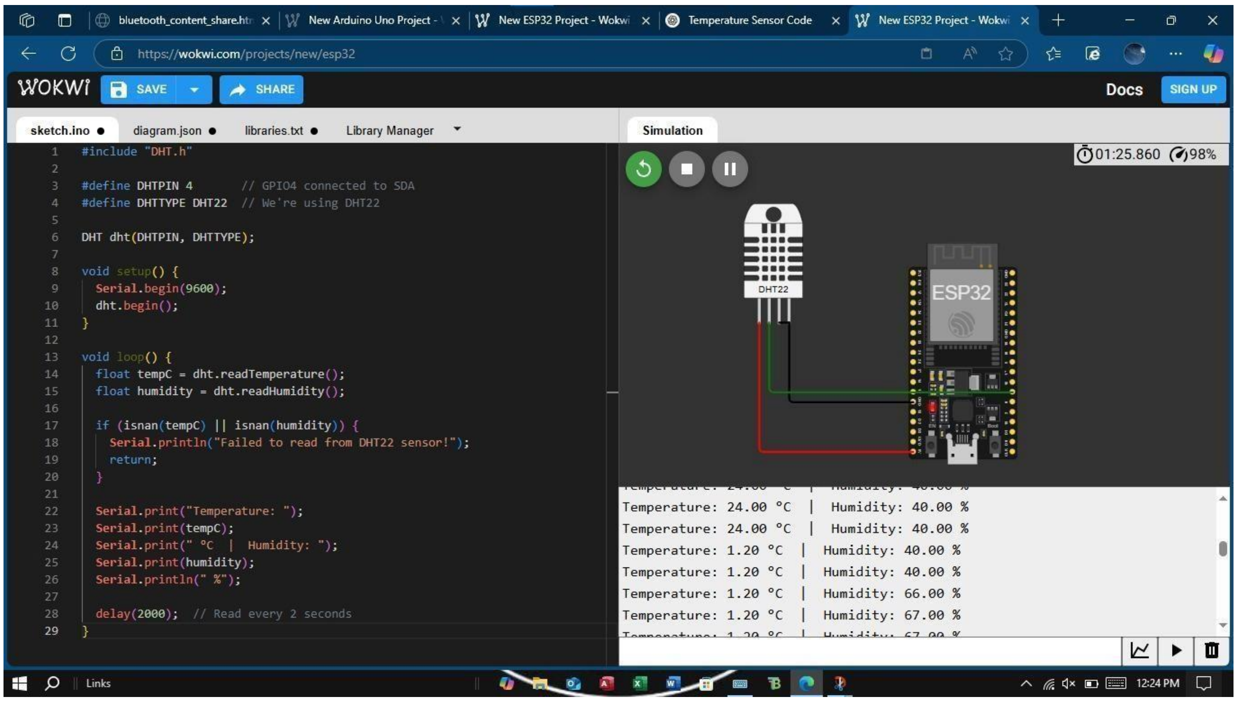
```
// Setup a oneWire instance to communicate with any OneWire devices (not just
OneWire oneWire(ONE_WIRE_BUS);

// Pass our oneWire reference to Dallas Temperature.
DallasTemperature sensors(&oneWire);

void setup() {
  Serial.begin(115200);
  sensors.begin();
}

void loop() {
  sensors.requestTemperatures();
  float temperature = sensors.getTempCByIndex(0);
  float tempF = (temperature );

  Serial.print("Temperature: ");
  Serial.print(tempF);
  Serial.println("°C");

  if (tempF > 32) {
    Serial.println("The temperature is high for the house. Turn on AC.");
  } else if (tempF < 25) {
    Serial.println("The temperature is too low for the house. Turn on the hea
  } else {
    Serial.println("Good temperature for the house.");
  }

  delay(3000);
}
```

```
Good temperature for the house.
Temperature: 29.37°C
Good temperature for the house.
Temperature: 42.13°C
The temperature is high for the house. Turn on AC.
Temperature: 32.63°C
The temperature is high for the house. Turn on AC.
```

Figure12: temperature sensor wokwi

3.2.1.3 humidity sensor

Our humidity sensor is MH series sensor, dhtt has been used since it's the only available humidity sensor in the wokwi.It is a both digital and analog sensor. Unlike digital sensors that provide pre-processed data, this sensor outputs a voltage that changes in proportion to the relative humidity in the air. The ESP32 reads this voltage through one of its analog input pins and converts it into humidity percentage. The MH-series humidity sensor provides analog output based on the surrounding moisture levels. It is powered by a 5V supply and produces a varying voltage signal that is proportional to relative humidity (SparkFun, 2022).



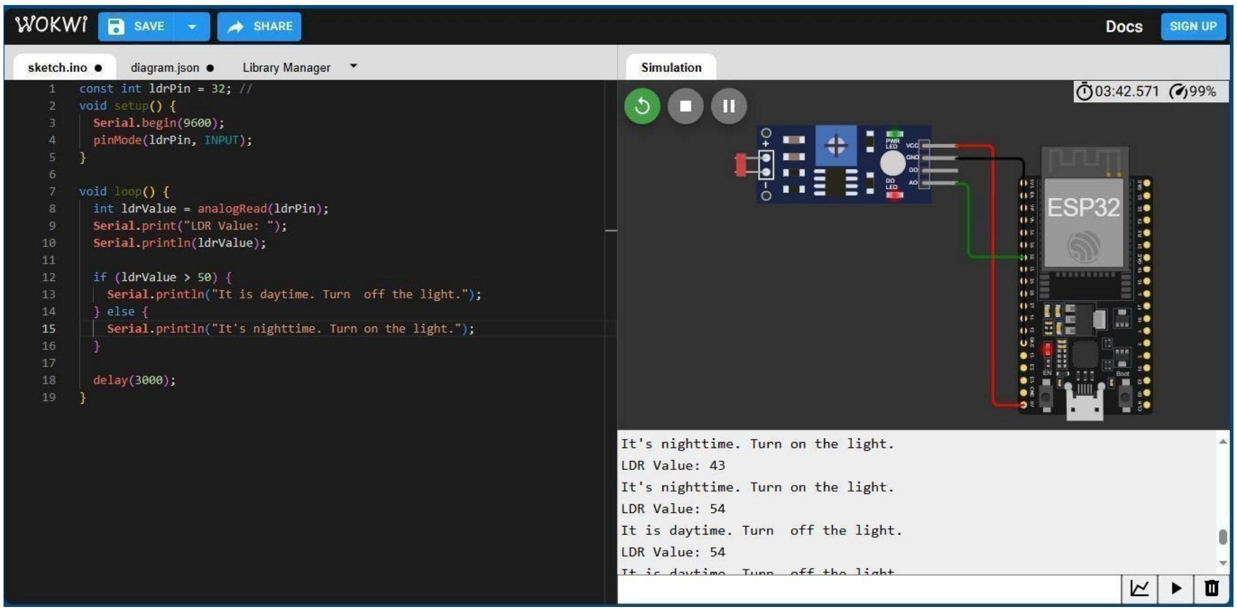
```
#include "DHT.h"

#define DHTPIN 4        // GPIO4 connected to SDA
#define DHTTYPE DHT22  // We're using DHT22

DHT dht(DHTPIN, DHTTYPE);

void setup() {
  Serial.begin(9600);
  dht.begin();
}

void loop() {
  float tempC = dht.readTemperature();
  float humidity = dht.readHumidity();

  if (isnan(tempC) || isnan(humidity)) {
    Serial.println("Failed to read from DHT22 sensor!");
    return;
  }

  Serial.print("Temperature: ");
  Serial.print(tempC);
  Serial.print(" °C  |  Humidity: ");
  Serial.print(humidity);
  Serial.println(" %");

  delay(2000);  // Read every 2 seconds
}
```

```
Temperature: 24.00 °C  |  Humidity: 40.00 %
Temperature: 24.00 °C  |  Humidity: 40.00 %
Temperature: 1.20 °C  |  Humidity: 40.00 %
Temperature: 1.20 °C  |  Humidity: 40.00 %
Temperature: 1.20 °C  |  Humidity: 66.00 %
Temperature: 1.20 °C  |  Humidity: 67.00 %
```

Figure13: humidity sensor wokwi

3.2.1.4 LDR sensor

Ldr is an analog sensor. LDRs do not provide a voltage output on their ow The LDR changes its resistance depending on the light falling on it. In this project, it was connected in a voltage divider circuit with a 10kΩ resistor to generate a readable analog voltage signal (Components101, 2023). The output voltage was calculated using the formula Vout=Vin×R2/R1+R2(ElectronicsTutorials, n.d.).



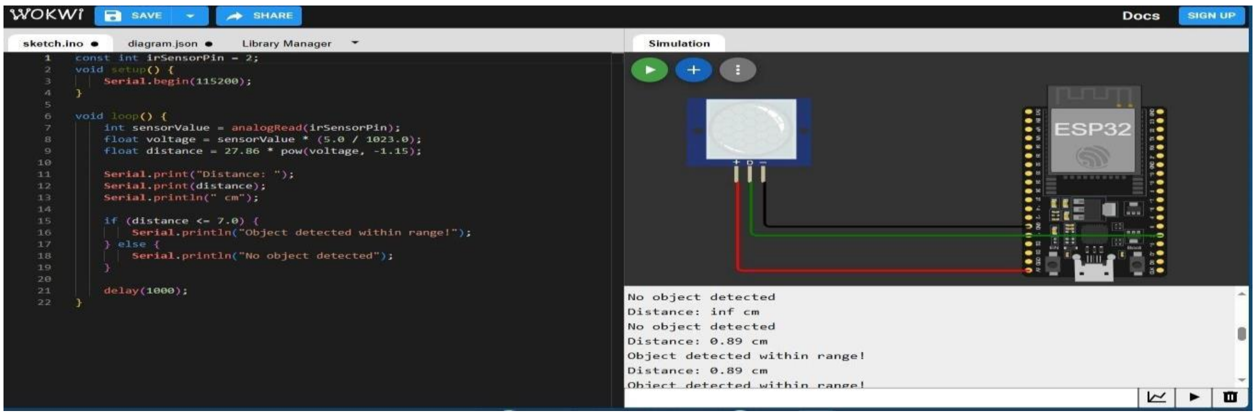
```
const int ldrPin = 32; //
void setup() {
  Serial.begin(9600);
  pinMode(ldrPin, INPUT);
}

void loop() {
  int ldrValue = analogRead(ldrPin);
  Serial.print("LDR Value: ");
  Serial.println(ldrValue);

  if (ldrValue > 50) {
    Serial.println("It is daytime. Turn  off the light.");
  } else {
    Serial.println("It's nighttime. Turn on the light.");
  }

  delay(3000);
}
```

```
It's nighttime. Turn on the light.
LDR Value: 43
It's nighttime. Turn on the light.
LDR Value: 54
It is daytime. Turn  off the light.
LDR Value: 54
It is daytime. Turn  off the light.
```

Figure14: ldr sensor wokwi

3.2.1.5 infra-red

The PIR sensor detects motion by sensing infrared radiation from humans or animals. When movement is detected, it sends a HIGH signal to the microcontroller. It operates on 5V and has a detection range of 3–5 (SparkFun, 2022). It is an analog sensor



```
const int irSensorPin = 2;
void setup() {
    Serial.begin(115200);
}

void loop() {
    int sensorValue = analogRead(irSensorPin);
    float voltage = sensorValue * (5.0 / 1023.0);
    float distance = 27.86 * pow(voltage, -1.15);

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

    if (distance <= 7.0) {
        Serial.println("Object detected within range!");
    } else {
        Serial.println("No object detected");
    }

    delay(1000);
}
```

```
No object detected
Distance: inf cm
No object detected
Distance: 0.89 cm
Object detected within range!
Distance: 0.89 cm
Object detected within range!
```

Figure15: infra-red sensor wokwi

3.3 main circuit design

The main circuit of the system combines all sensors, the microcontroller, and power distribution into one complete layout. It is designed to gather real-time environmental data, process it, and send the information to a server over a wireless network. Each sensor plays a specific role, but the circuit as a whole must operate in coordination to support the overall goal of automation and monitoring.

The sensors are divided into two types which is analog and digital. The analog sensors, such as the LDRs and humidity sensor, are connected to the ESP32's analog input pins. Their varying voltage outputs are interpreted as environmental changes. Voltage dividers are used where needed to ensure that signals stay within the microcontroller's 0 to 3.3V range.

The digital sensors, including the temperature sensor and motion detector, are connected to digital GPIO pins and communicate through digital high or low signals. All sensors share a common ground with the microcontroller, ensuring stable operation. Pull-up resistors and protection resistors are added based on the sensor requirements to avoid signal input damage**.** The circuit was first built and tested on a breadboard**.** Sensor values were monitored through the serial output during early testing stages. Adjustments were made to resistor values and wiring based on test results to improve reliability and consistency.

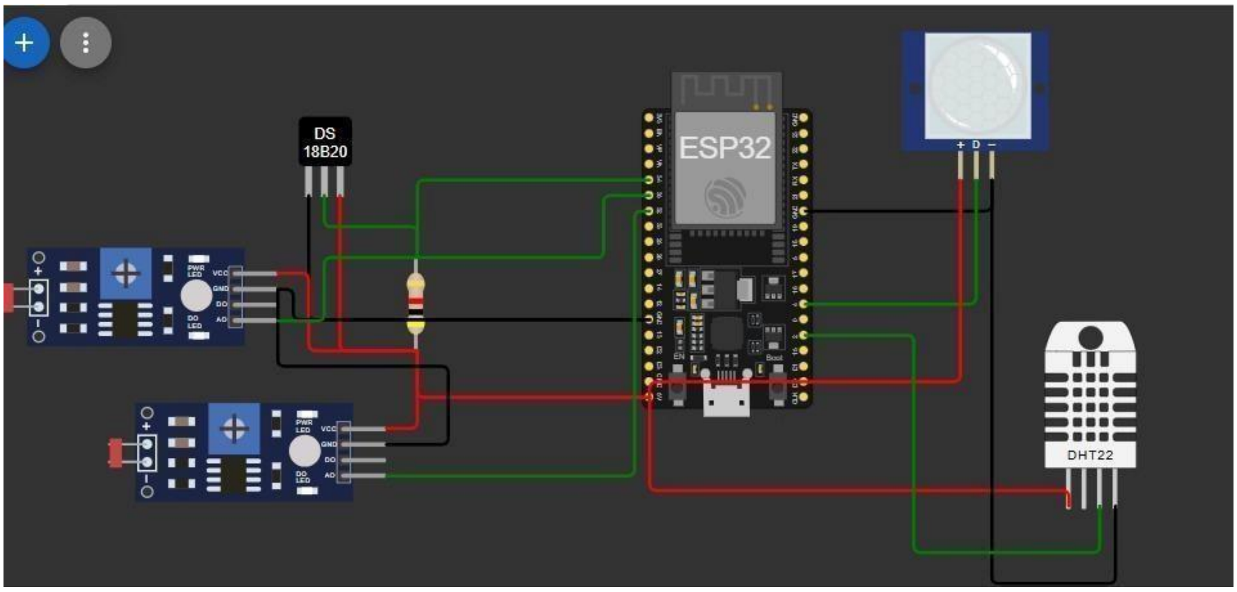


Figure16: main circuit illustration wokwi

3.4 safety precautions for hardware

While working on the hardware design and assembly of the system, several safety measures were followed to protect both the components and the individuals involved in the project which are:

- Before wiring any components, the power supply was always disconnected to avoid accidental shorts.
- The breadboard was inspected regularly to ensure no loose connections or overlapping wires could lead to incorrect readings.
- Components were never touched during live testing to avoid static discharge

3.5 wiring

The wiring process was done with careful attention to sensor voltage requirements, signal types, and safe connection practices to ensure stable and accurate system operation. All sensors were connected to the ESP32 WeMo's D1 R32**,** which served as the central control unit for reading data and transmitting it wirelessly to the web server. A common ground was used across all components to maintain consistent voltage referencing and avoid signal noise.

LDR Sensors (LDR1 and LDR2)

- Both LDRs were connected using voltage divider circuits with fixed 10kΩ resistors. One end of each LDR was connected to 3.3V, while the other end was linked to the analog input pins A0 and A1 on the ESP32. The junction between the LDR and resistor created a variable voltage signal based on light intensity, which the ESP32 could read as an analog value. This setup allowed accurate monitoring of brightness levels for light control or curtain automation.

Dallas Temperature Sensor (DS18B20)

- The temperature sensor was wired to GPIO D4 on the ESP32. It uses a 1-Wire digital protocol, which required a 4.7kΩ pull-up resistor between the data line and 3.3V to maintain a proper HIGH state during communication. The sensor's VCC pin was connected to 3.3V, and GND was connected to the ESP32's GND. This digital setup allowed for reliable single-wire communication, saving GPIO space.

Humidity Sensor (MH Series Analog)

- The MH-series humidity sensor was powered with 5V from the ESP32 and connected to an analog input pin A2. Since it provides a continuous analog voltage output proportional to humidity levels, no special interface circuitry was required. However, care was taken to ensure the output signal did not exceed the ESP32's 3.3V logic tolerance.

Passive Infrared (PIR) Sensor

- The PIR sensor had three pin, vcc, ground, and out. It was powered from the 3.3V pin, and the digital out pin was connected to GPIO D2. Its logic output is directly compatible with the ESP32's digital inputs, so no level shifting or resistors were required. The sensor was mounted with an unobstructed view to ensure reliable motion detection during testing.

Power Distribution and Grounding

The ESP32 provided both 3.3V and 5V outputs, which were used based on each sensor's requirements. All GND connections from the sensors were tied back to the ESP32's ground rail to ensure stable voltage references and prevent electrical floating signals. The entire circuit was built on a breadboard, which allowed easy rearrangement and voltage.

3.6 Breadboard design

To build and test the home automation and security system, all components were first assembled on a solderless breadboard. This method allowed for fast prototyping, easy rearrangement of components, and troubleshooting during the early stages of hardware design. The breadboard layout followed the exact wiring plan described in the previous section and matched the simulated circuit from Wokwi.

Each sensor was connected using jumper wires for power, ground, and signal pins. Resistors used in voltage divider circuits were placed directly across the power rails and sensor pins.

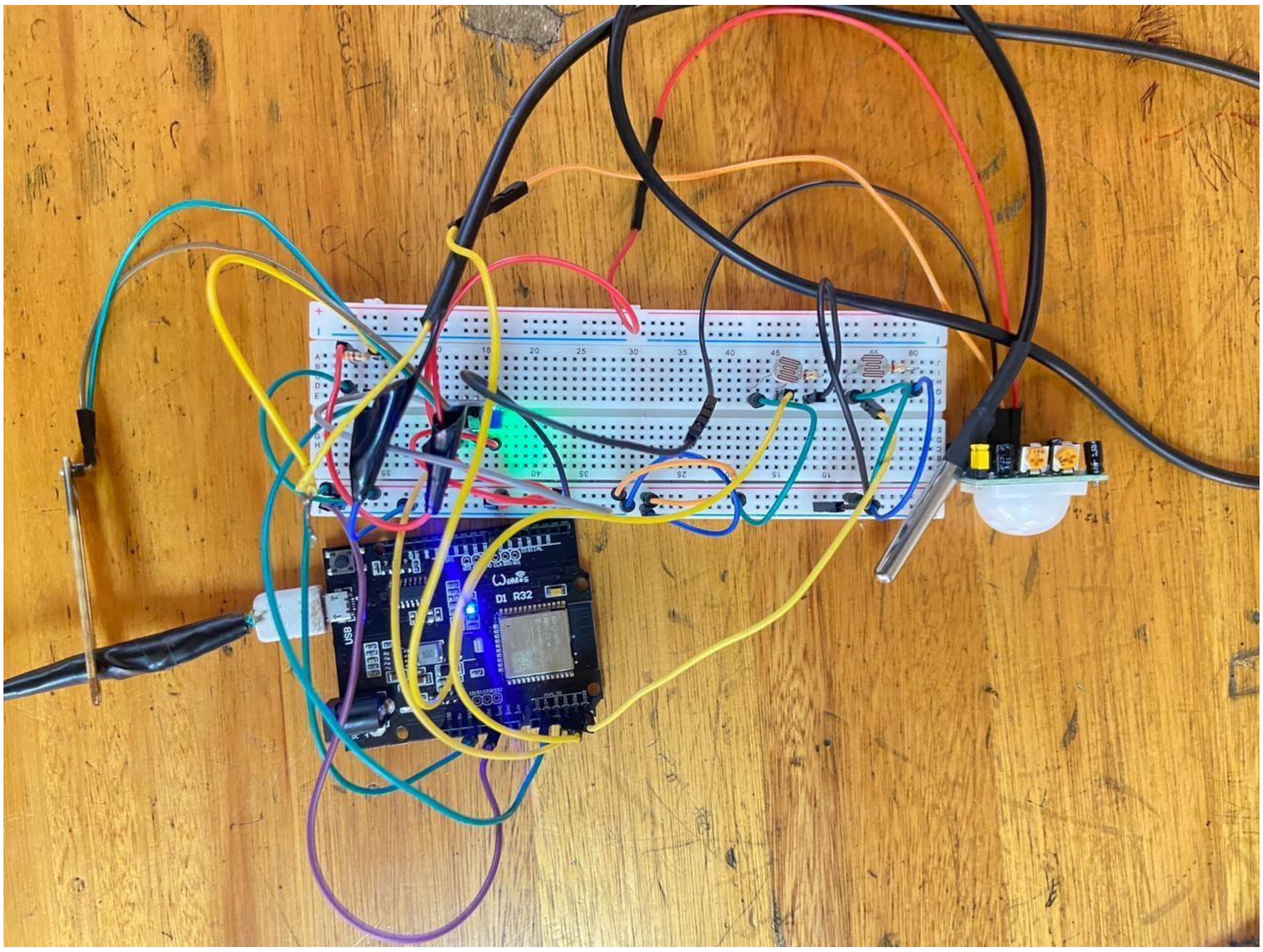


Figure17: Breadboard and sensor wiring layout

3.7 enclosure design

After confirming that the circuit worked as expected on the breadboard, the components were carefully mounted inside a custom enclosure made from cardboard for demonstration purposes. The enclosure served as a physical housing to simulate how the system might be installed in a real home. Sensor positions were chosen to allow open access to light and motion. Airflow was also considered for accurate humidity and temperature readings.

Though the enclosure was temporary, it gave a practical understanding of sensor placement, wire management, and how physical layout can affect sensor performance. For example, the PIR sensor had to be positioned with a clear field of view to detect motion effectively, while LDRs needed to face light sources without obstruction.

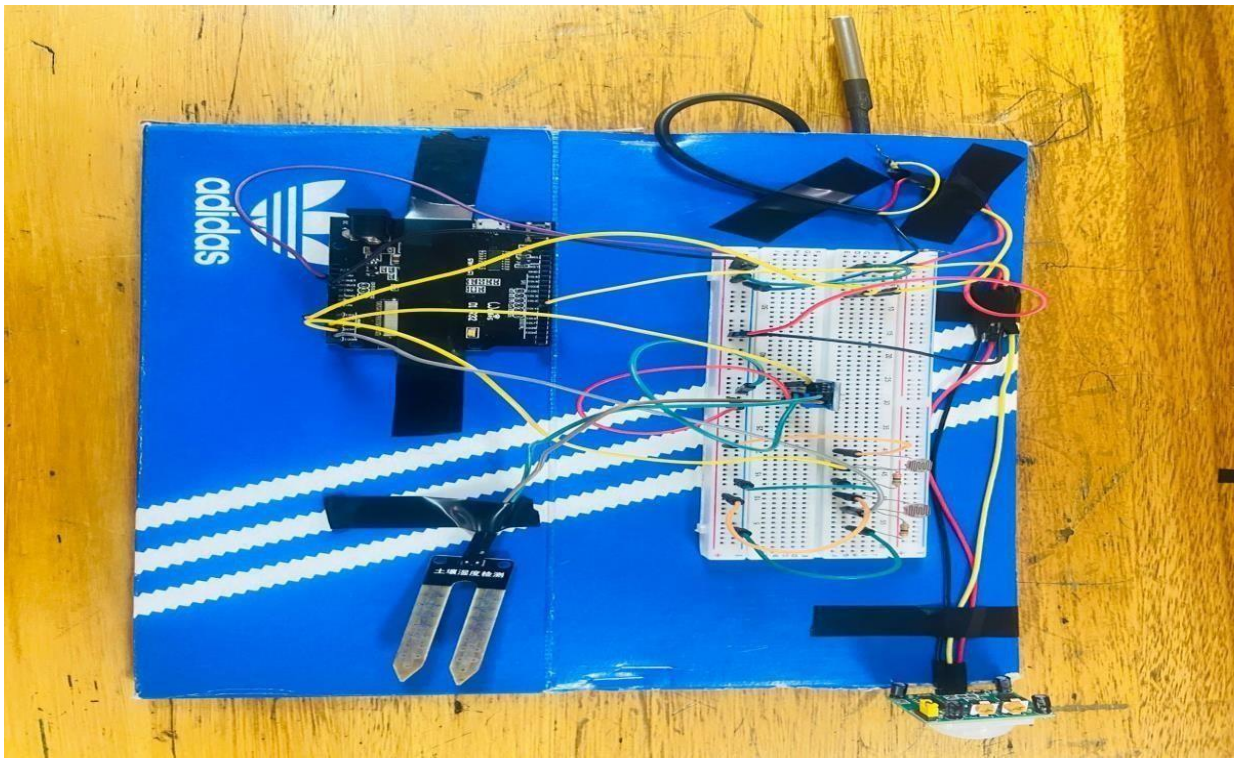


Figure 18: The prototype’s final assembly

3.8 conclusion

This chapter provided a detailed explanation of the hardware setup used to build the home automation and security system. The wiring and integration of all sensors were carried out carefully, with attention to voltage levels, signal types, and safe circuit design. Each sensor was connected to the microcontroller according to its output requirements, and supporting components such as resistors were included where needed to ensure stability and protection. This hardware setup forms the physical foundation of the entire system and ensures that data collection happens accurately and consistently.

The next chapter will focus on the software side of the project, including how sensor data is read, processed, transmitted, and displayed through the user interface.

Chapter 4: software design

4.1 INTRODUCTION

This chapter focuses on the software development required to support the hardware system described in Chapter 3. The software is responsible for collecting sensor data from the ESP32 microcontroller, processing the input values, and sending the information to a web server for storage and display. It also manages communication between the hardware and user interface, making it possible to monitor environmental changes in real time.

The software requirements for this project include reliable sensor data reading, stable wireless communication, proper formatting of the data, and a responsive web interface that displays updated information clearly. To meet these needs, embedded code was developed for the ESP32, and server-side scripts were written to handle data transfer and storage. A web-based front end was created to show the data to users in a simple and readable format.

Based on the circuit design and wiring established in Chapter 3, the software was structured to match the sensor inputs, voltage levels, and timing required for accurate data capture. The logic was tested and refined to ensure smooth operation and system stability. This chapter will explain the programming structure, data flow, and software tools used to bring the entire system together.

4.2 software design

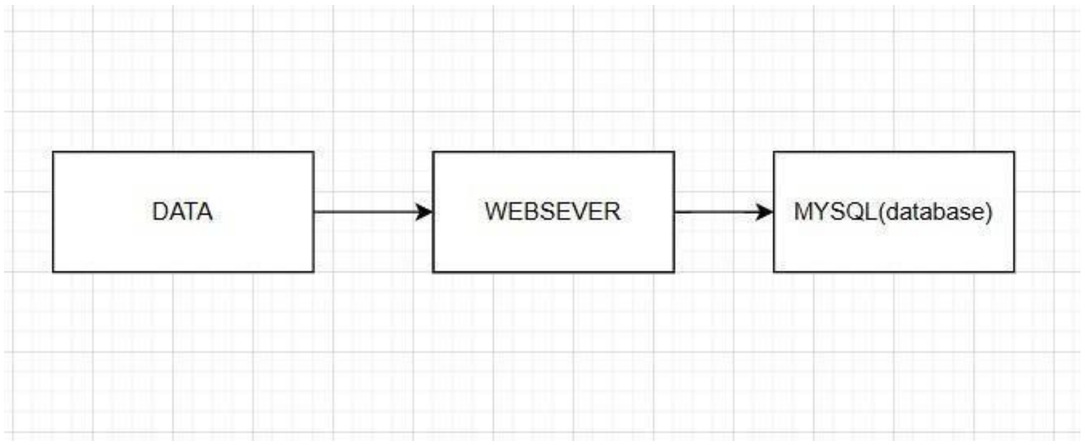


Figure19: block diagram of software

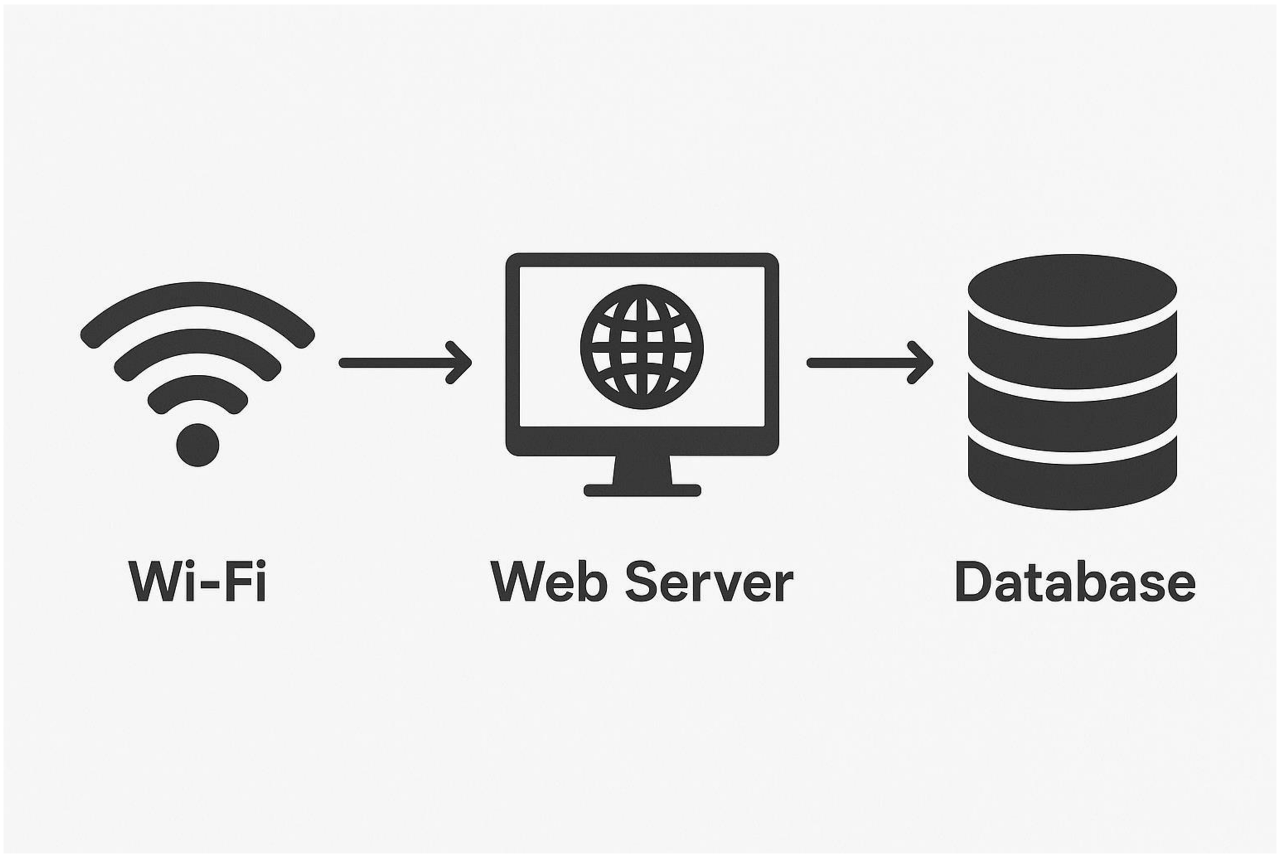


Figure20: a network diagram showing a Wi-Fi connection to a webserver which then connect to database

4.2.1 software use case diagram

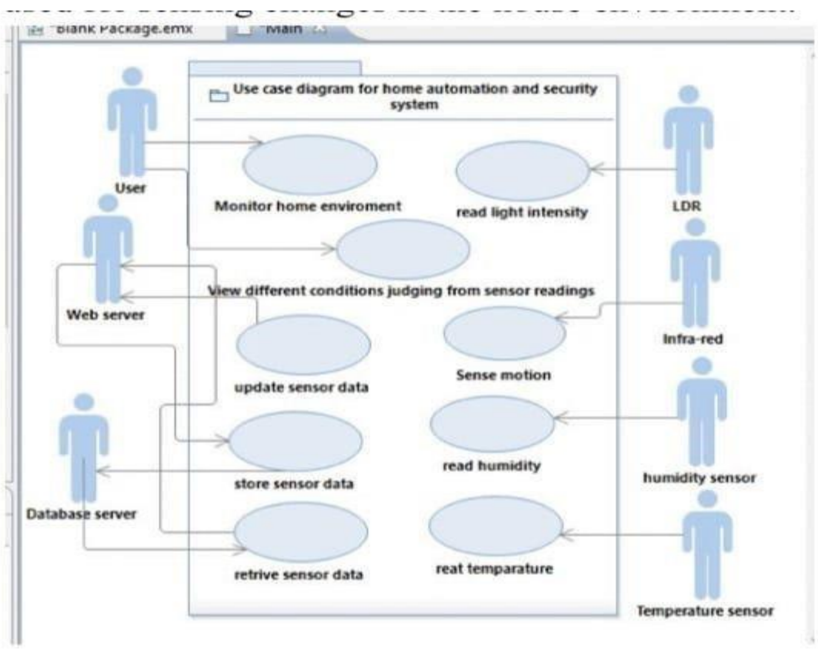


Figure21: use case diagram

4.2.2software activity diagram

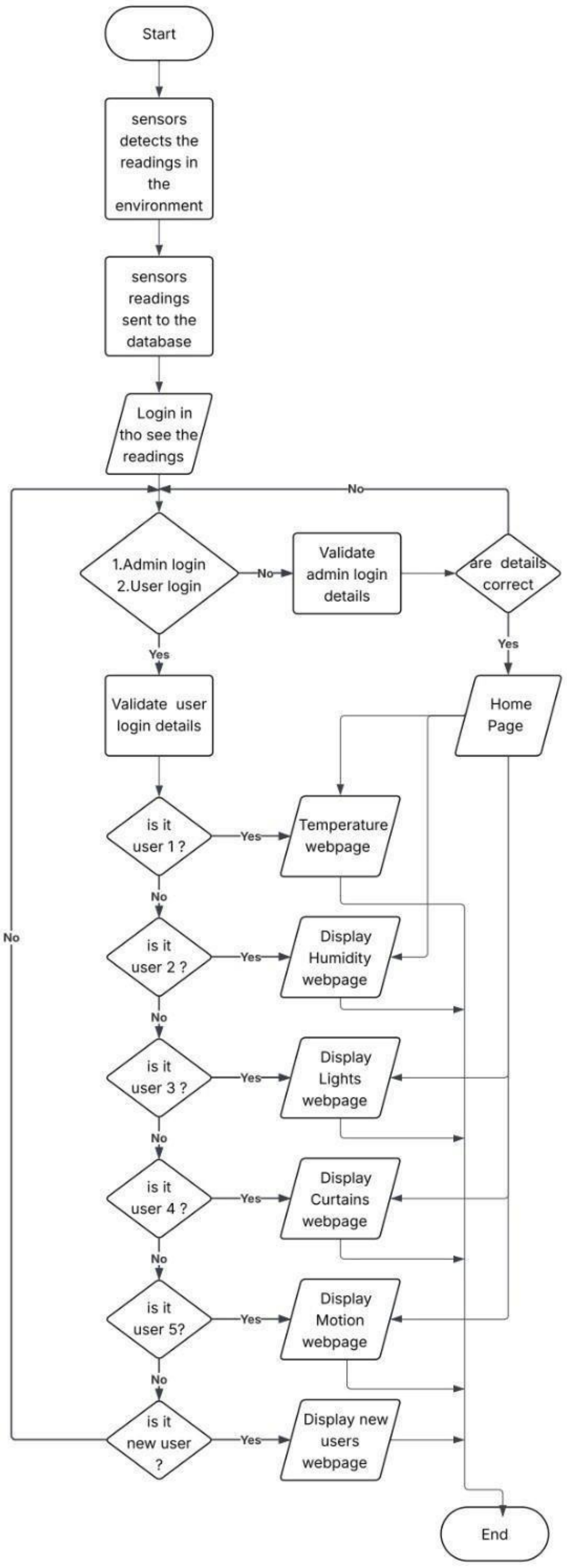


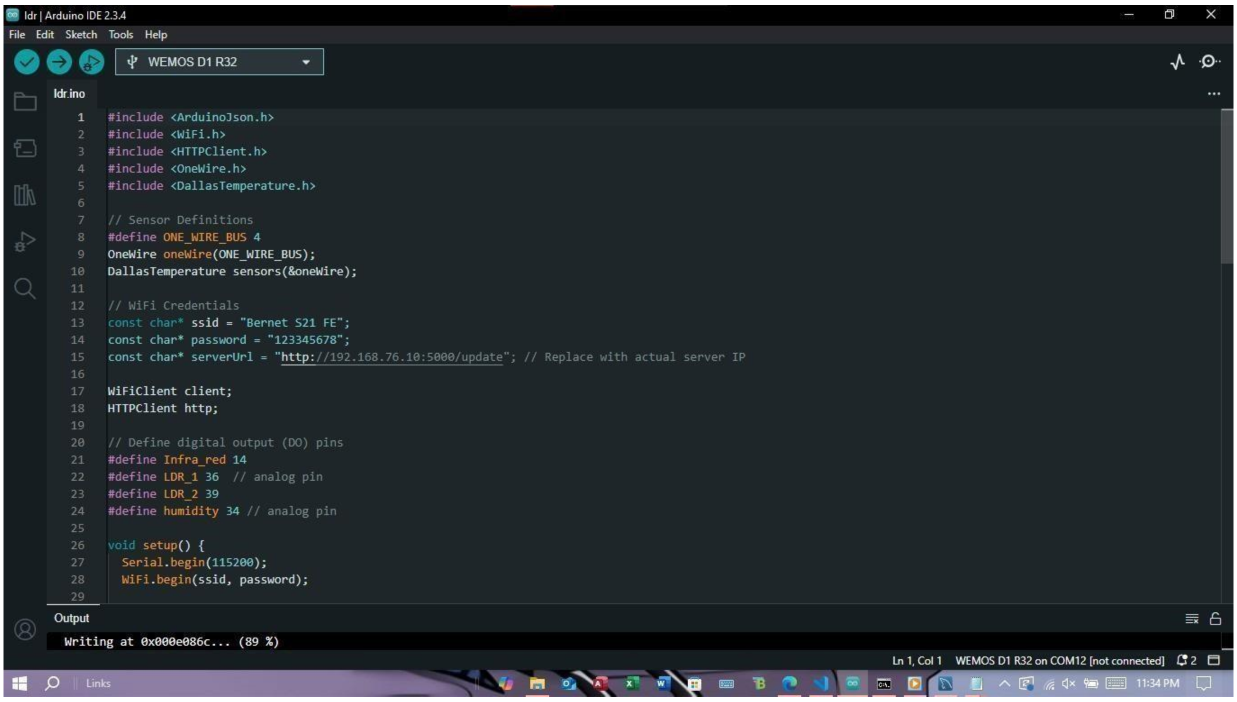
Figure22: activity diagram

4.2.3 Arduino ide code

It is uploaded to the ESP32 WeMo’s D1 R32 microcontroller, which is responsible for reading all sensor data, processing the inputs, and sending the results to a web server over Wi-Fi. The values collected using this code are stored in variables and formatted into an HTTP request that is sent to the web server.

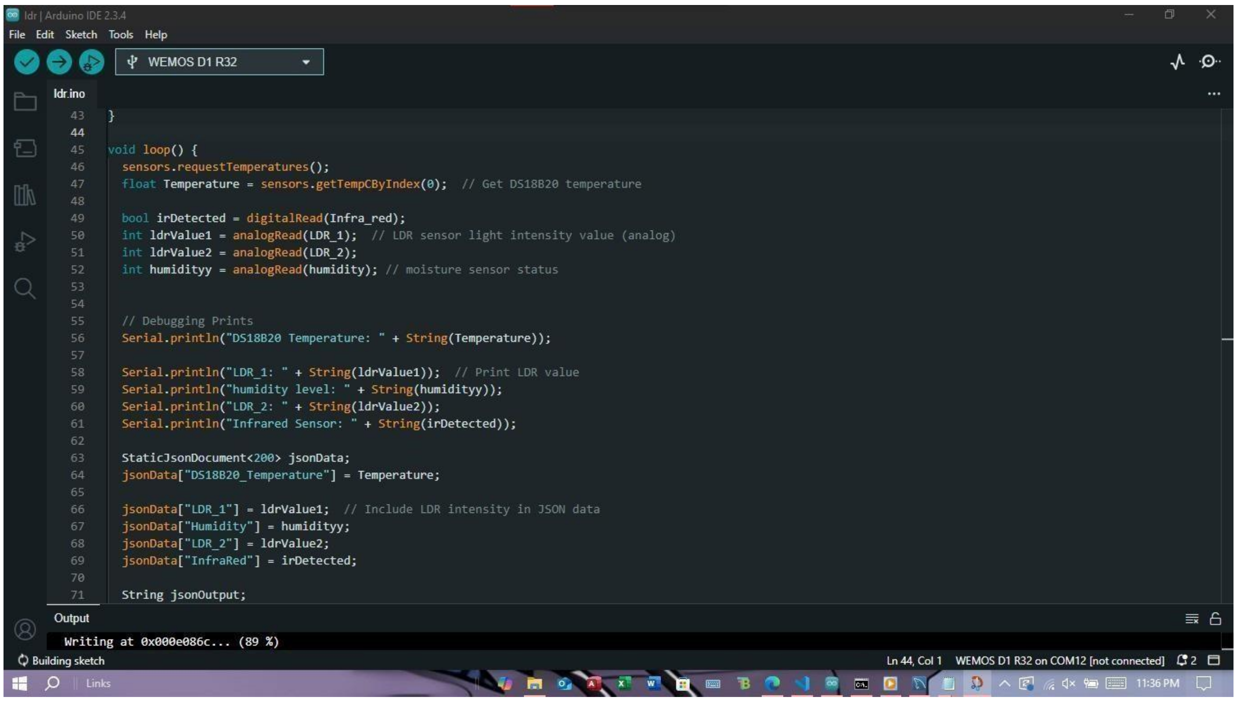
The HTTP GET or POST request contains the latest sensor readings, which the server receives and passes to the database for storage. This makes it possible for users to view the live data on the web interface.

The Arduino code also includes basic logic to handle and to ensure that the Wi-Fi connection remains active. Error handling features are included to restart the connection if Wi-Fi fails or sensor readings are not received correctly.


ldr | Arduino IDE 2.3.4
File Edit Sketch Tools Help
WEMOS D1 R32
ldr.ino
Output
Ln 1, Col 1 WEMOS D1 R32 on COM12 [not connected]
11:34 PM

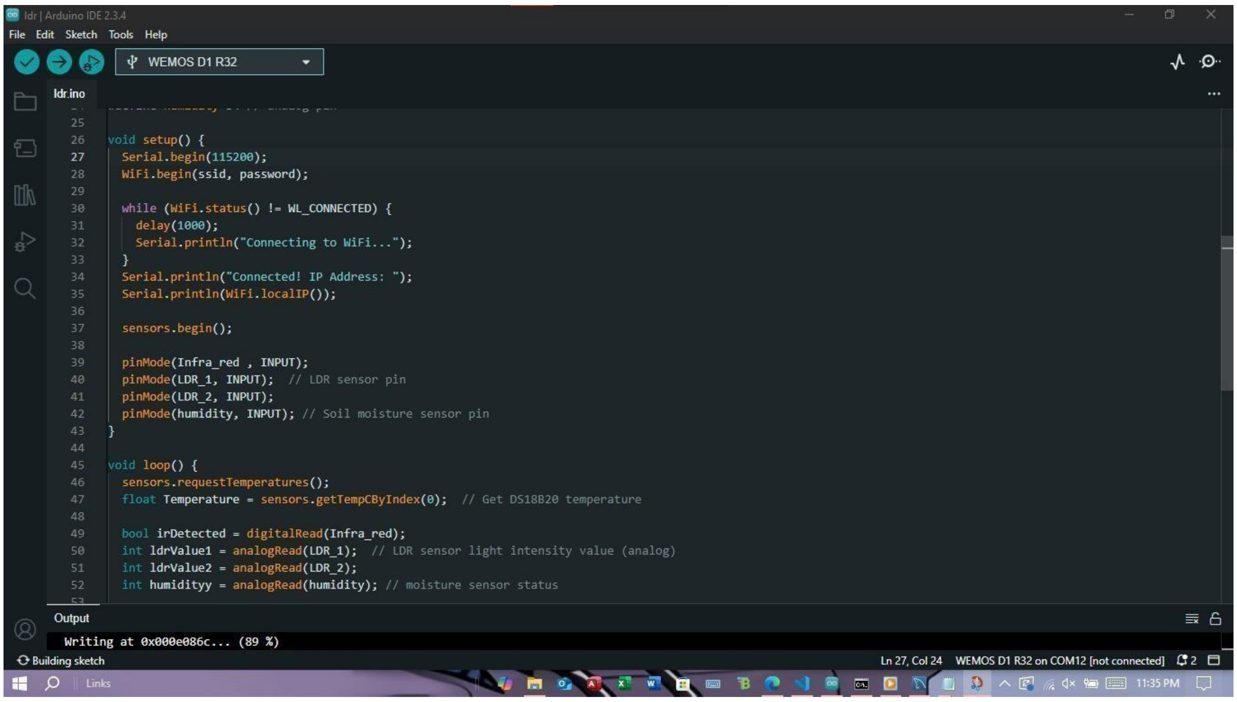
```cpp
#include <ArduinoJson.h>
#include <WiFi.h>
#include <HTTPClient.h>
#include <OneWire.h>
#include <DallasTemperature.h>

// Sensor Definitions
#define ONE_WIRE_BUS 4
OneWire oneWire(ONE_WIRE_BUS);
DallasTemperature sensors(&oneWire);

// WiFi Credentials
const char* ssid = "Bernet S21 FE";
const char* password = "123345678";
const char* serverUrl = "http://192.168.76.10:5000/update"; // Replace with actual server IP

WiFiClient client;
HTTPClient http;

// Define digital output (DO) pins
#define Infra_red 14
#define LDR_1 36  // analog pin
#define LDR_2 39
#define humidity 34 // analog pin

void setup() {
  Serial.begin(115200);
  WiFi.begin(ssid, password);

```

```
Writing at 0x000e086c... (89 %)
```


ldr | Arduino IDE 2.3.4
File Edit Sketch Tools Help
WEMOS D1 R32
ldr.ino
Output
Building sketch
Ln 44, Col 1 WEMOS D1 R32 on COM12 [not connected]
11:36 PM

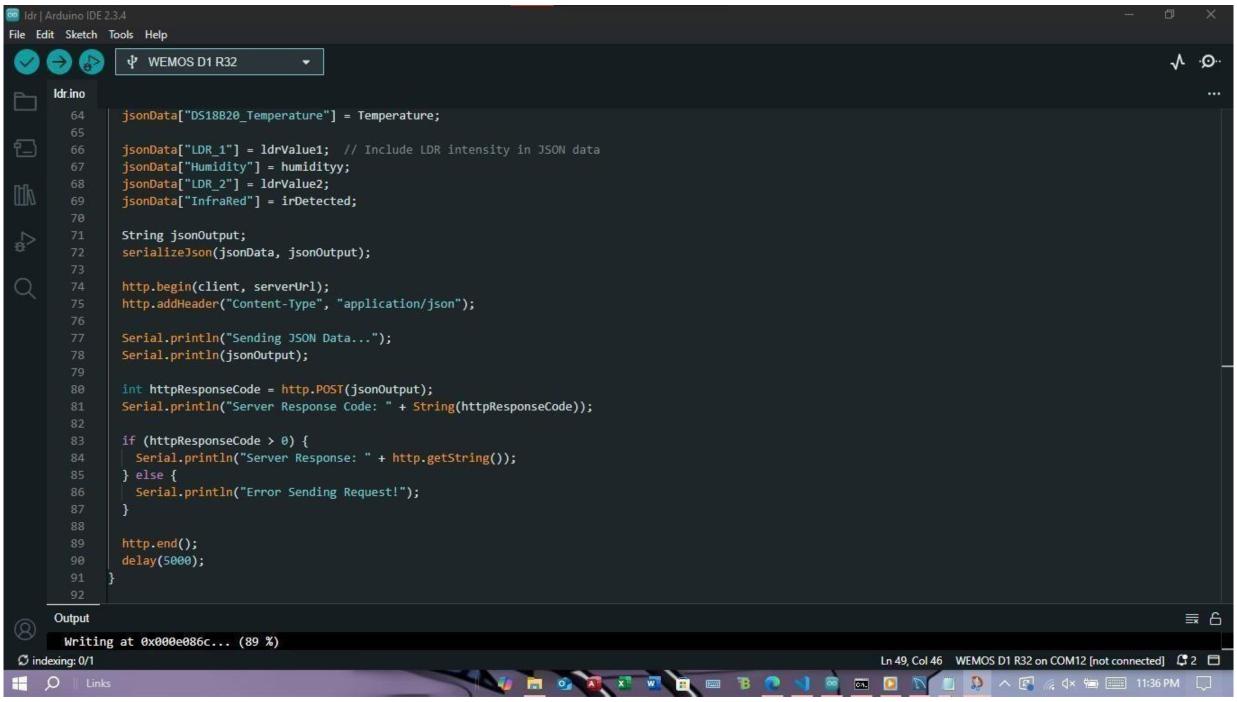
```cpp
}

void loop() {
  sensors.requestTemperatures();
  float Temperature = sensors.getTempCByIndex(0);  // Get DS18B20 temperature

  bool irDetected = digitalRead(Infra_red);
  int ldrValue1 = analogRead(LDR_1);  // LDR sensor light intensity value (analog)
  int ldrValue2 = analogRead(LDR_2);
  int humidityy = analogRead(humidity); // moisture sensor status


  // Debugging Prints
  Serial.println("DS18B20 Temperature: " + String(Temperature));

  Serial.println("LDR_1: " + String(ldrValue1));  // Print LDR value
  Serial.println("humidity level: " + String(humidityy));
  Serial.println("LDR_2: " + String(ldrValue2));
  Serial.println("Infrared Sensor: " + String(irDetected));

  StaticJsonDocument<200> jsonData;
  jsonData["DS18B20_Temperature"] = Temperature;

  jsonData["LDR_1"] = ldrValue1;  // Include LDR intensity in JSON data
  jsonData["Humidity"] = humidityy;
  jsonData["LDR_2"] = ldrValue2;
  jsonData["InfraRed"] = irDetected;

  String jsonOutput;
```

```
Writing at 0x000e086c... (89 %)
```



```
void setup() {
  Serial.begin(115200);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(1000);
    Serial.println("Connecting to WiFi...");
  }
  Serial.println("Connected! IP Address: ");
  Serial.println(WiFi.localIP());

  sensors.begin();

  pinMode(Infra_red , INPUT);
  pinMode(LDR_1, INPUT);  // LDR sensor pin
  pinMode(LDR_2, INPUT);
  pinMode(humidity, INPUT); // Soil moisture sensor pin
}

void loop() {
  sensors.requestTemperatures();
  float Temperature = sensors.getTempCByIndex(0);  // Get DS18B20 temperature

  bool irDetected = digitalRead(Infra_red);
  int ldrValue1 = analogRead(LDR_1);  // LDR sensor light intensity value (analog)
  int ldrValue2 = analogRead(LDR_2);
  int humidityy = analogRead(humidity); // moisture sensor status
```

```
Writing at 0x000e086c... (89 %)
```



```
  jsonData["DS18B20_Temperature"] = Temperature;

  jsonData["LDR_1"] = ldrValue1;  // Include LDR intensity in JSON data
  jsonData["Humidity"] = humidityy;
  jsonData["LDR_2"] = ldrValue2;
  jsonData["InfraRed"] = irDetected;

  String jsonOutput;
  serializeJson(jsonData, jsonOutput);

  http.begin(client, serverUrl);
  http.addHeader("Content-Type", "application/json");

  Serial.println("Sending JSON Data...");
  Serial.println(jsonOutput);

  int httpResponseCode = http.POST(jsonOutput);
  Serial.println("Server Response Code: " + String(httpResponseCode));

  if (httpResponseCode > 0) {
    Serial.println("Server Response: " + http.getString());
  } else {
    Serial.println("Error Sending Request!");
  }

  http.end();
  delay(5000);
}
```

```
Writing at 0x000e086c... (89 %)
```

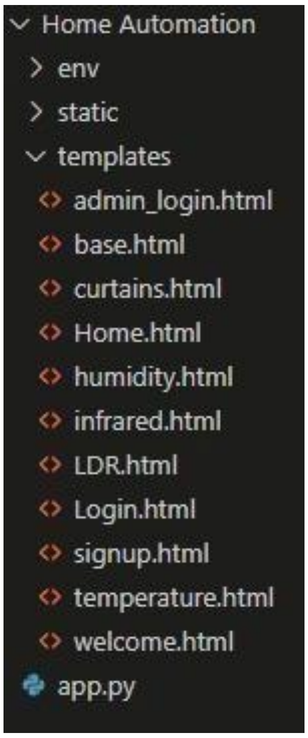
Figure23: Arduino code used to collect and transmit sensor data

4.2.4webserver

The web server acts as the central point for receiving, storing, and displaying the data sent from the ESP32 microcontroller. It forms the communication bridge between the hardware and the user interface, making it possible for users to view sensor readings in real time through a web browser. Once the ESP32 reads values from the connected sensors which are ldr, temperature, humidity and infra- red it sends this data over Wi-Fi to the web server using an HTTP request**.** This request includes all sensor readings in the URL 4.2.4.1front-end development

4.2.4.1front-end development

The front-end is the part of the system that users see and interact with in their browser. It provides a clean and accessible layout that shows real-time sensor data and basic system controls.



```
Home Automation
  env
  static
  templates
    admin_login.html
    base.html
    curtains.html
    Home.html
    humidity.html
    infrared.html
    LDR.html
    Login.html
    signup.html
    temperature.html
    welcome.html
  app.py
```

Figure24: Webserver setup with HTML and Python

The base HTML file in the project acts as the foundation layout for all the web pages in the system. It defines the main structure of the user interface and ensures that all other pages share a consistent design and layout. This approach simplifies development, improves readability, and allows for easier updates in the future.

Login.html

The login page is a key part of the system, used to control access to the sensor dashboard and prevent unauthorized users from viewing or altering data. It provides a basic authentication system that checks user credentials against stored information before granting access to the main interface.

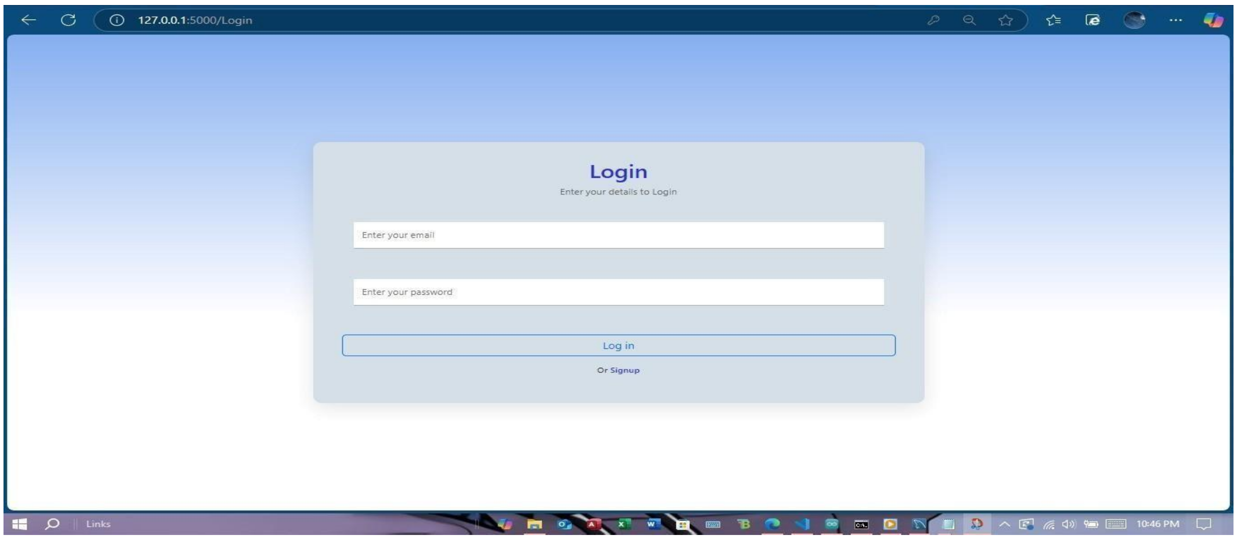


Figure25:login page

Signup.html

New users can create an account by submitting their name, email, password, and account type. The information is stored in the MySQL database if it passes validation.

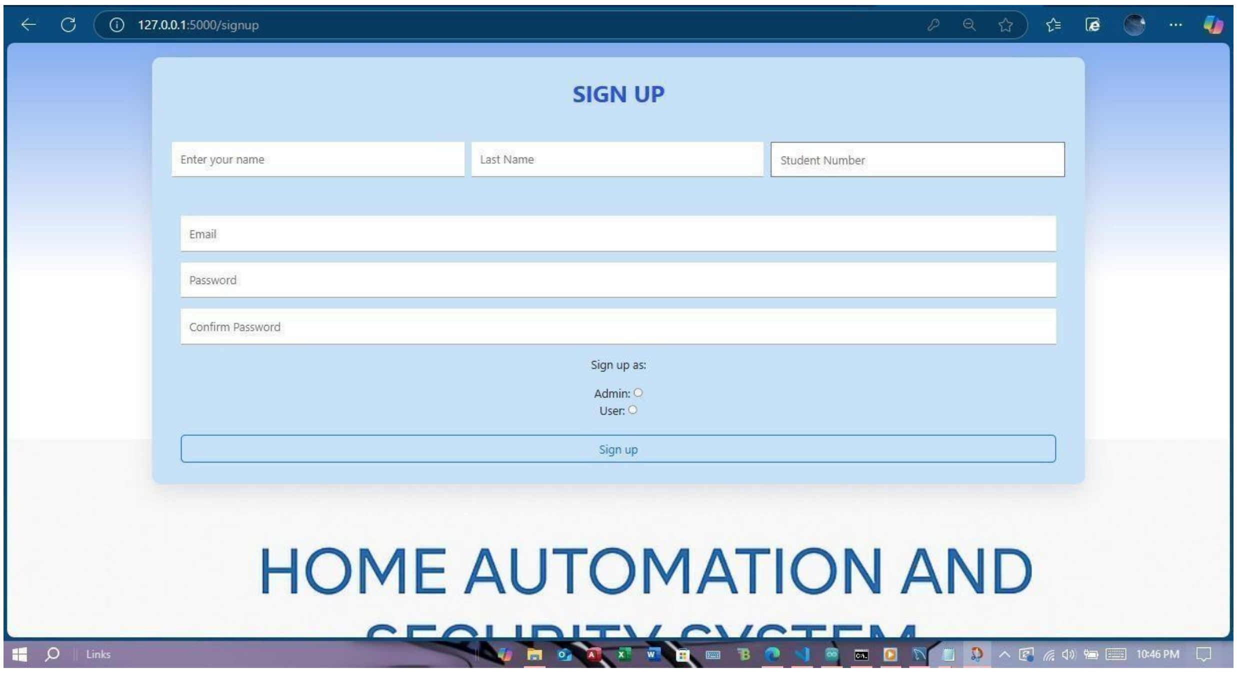


Figure26: signup page

Home html(welcome)

This page is the first web page that users see when they access the system. It acts as the landing page for the home automation and security interface, providing users with an introduction and navigation to other parts of the system, such as the login or sign-up pages.

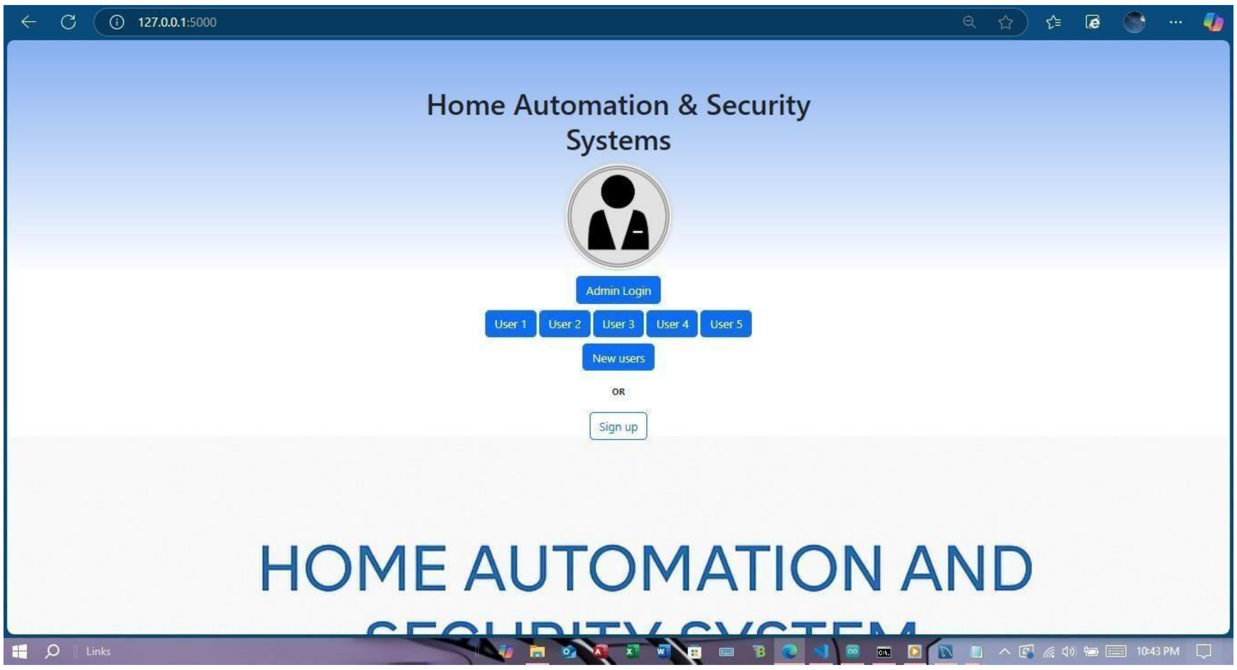


Figure27: welcome page

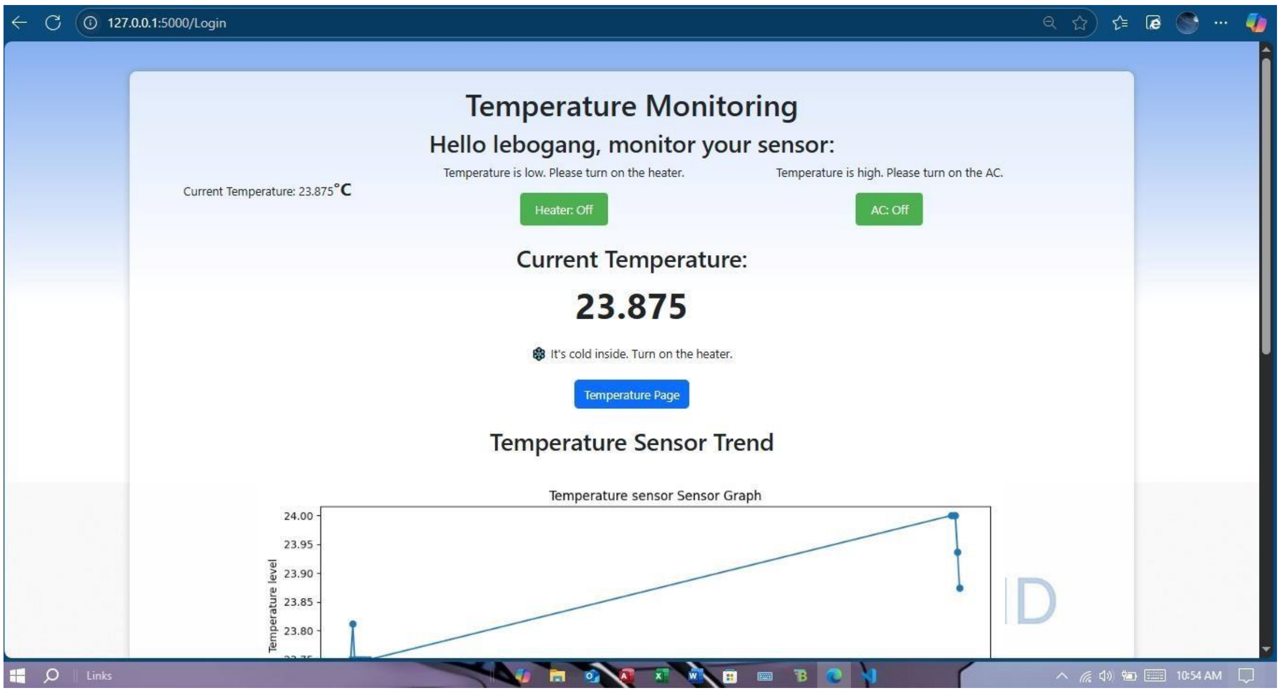
Sensors.htmls

These are the html pages for all the users. Show each sensors' current reading:

Temperature html page as an example of how html pages of each user look like:

This page is part of the user interface designed to monitor real-time temperature readings. It is developed using html, css, and flask (Python) and is displayed once the user has logged in successfully. The main purpose of this page is to give the user live updates on the current room temperature and provide control suggestions based on those readings.

The interface greets the user by name "Hello lebogang" and displays the latest temperature value in both a text format (23.875°C) and a graph format underneath. The graph shows how temperature trends over time, which helps in identifying whether the room is getting warmer or cooler. The system uses real-time data pulled from a MySQL database, processed through Python, and rendered on this page. It also provides contextual feedback, such as "It's cold inside. Turn on the heater."/" it's warm inside, turn the ac".


127.0.0.1:5000/Login

Temperature Monitoring

Hello lebogang, monitor your sensor:

Current Temperature: 23.875°C

Temperature is low. Please turn on the heater.

Temperature is high. Please turn on the AC.

Current Temperature:

23.875

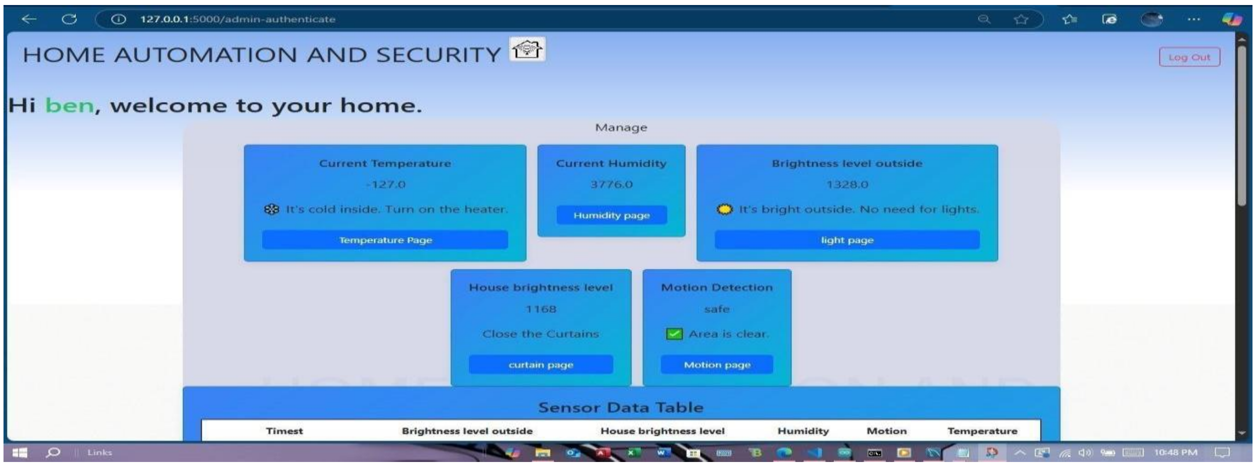
It's cold inside. Turn on the heater.

Temperature Sensor Trend

Figure28: temperature.html

Admin_login.html

This page is displayed only to authenticated users who have successfully logged in with admin credentials. It plays a key role in managing system access and monitoring sensor data securely. The admin page is structured using HTML and CSS for layout and styling. It pulls live sensor data from the server and displays it in a clean, readable format. The page layout typically includes a navigation bar, welcome header, and a live sensor table or graph panel. This allows the admin to easily track environmental conditions in real time from any browser on the local network. From a design perspective, this page also ensures security and access control. Admins can monitor sensor activity, identify motion triggers, or verify system status, while regular users do not have access to these controls. This improves system integrity and aligns with the initial project objective of providing controlled access.



127.0.0.1:5000/admin-authenticate

HOME AUTOMATION AND SECURITY

Log Out

Hi ben, welcome to your home.

Manage

Current Temperature
-127.0
It's cold inside. Turn on the heater.
Temperature Page

Current Humidity
3776.0
Humidity page

Brightness level outside
1328.0
It's bright outside. No need for lights.
light page

House brightness level
1168
Close the Curtains
curtain page

Motion Detection
safe
✓ Area is clear.
Motion page

Sensor Data Table

Timest	Brightness level outside	House brightness level	Humidity	Motion	Temperature

Figure29: Admin’s HTML dashboard

4.2.4.2backend

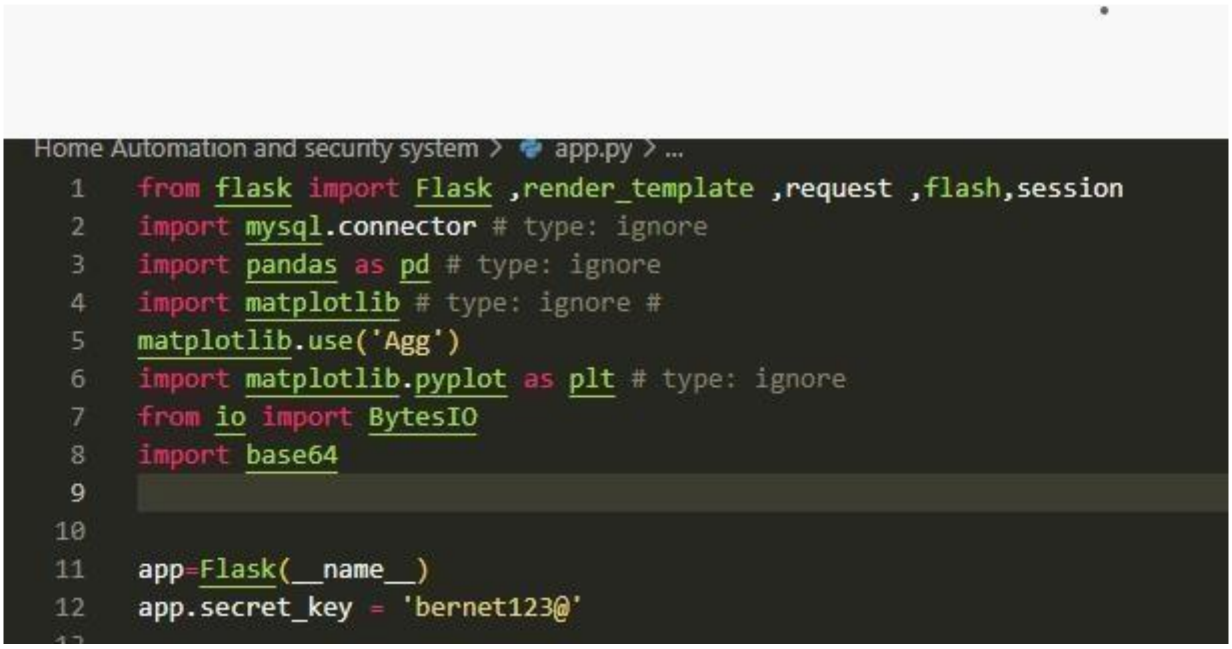
The backend of this project plays a critical role in processing and managing data collected from the sensors. The ESP32 reads values from the temperature, humidity, motion, and light sensors and transmits this data via Wi-Fi to a Flask web server built in Python. The data is sent using an HTTP POST request, where each reading is passed in JSON format**.**

Once the request is received, the Python script running on the Flask server parses the JSON payload and extracts the sensor values. These values are then inserted into the MySQL database using the mysql.connector library. This ensures that every new reading is logged and stored with its respective timestamp.

For data display, the Flask backend queries the MySQL database to retrieve the latest sensor values. These are sent to the frontend and dynamically rendered using HTML and CSS, providing the user with a real-time view of environmental conditions. This setup enables smooth interaction between the microcontroller, backend logic, and web interface. The script responsible for database insertion and retrieval can be found in the backend code file titled app.py.

```
pip install Flask
```

Figure30: command for installing flask


```
Home Automation and security system > app.py > ...
from flask import Flask ,render_template ,request ,flash,session
import mysql.connector # type: ignore
import pandas as pd # type: ignore
import matplotlib # type: ignore #
matplotlib.use('Agg')
import matplotlib.pyplot as plt # type: ignore
from io import BytesIO
import base64


app=Flask(__name__)
app.secret_key = 'bernet123@'
```
Figure31: Python–MySQL connection setup with initiation of flask

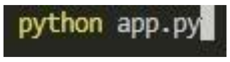

```
python app.py
```

Figure32: command for running our python file

4.2.6 MYSQL(database)

The database plays a critical role as the central storage unit for all the environmental data collected by the ESP32 microcontroller. It stores real-time sensor readings such as temperature, humidity, motion detection, and light intensity along with the exact timestamps of when each value was recorded. This allows the system to keep a complete history of the conditions inside the home.

The database not only stores this information but also ensures that the data is well organized by creating structured tables for each sensor or data type. This structure makes it easy to retrieve specific records when needed, such as viewing trends over time or triggering automation rules based on past data.

Before the data is displayed on the web interface, it is first retrieved or queried from the database. This step ensures that only accurate, up-to-date readings are shown to the user. The database improves the reliability and functionality of the system by acting as a bridge between the microcontroller and the web interface. It supports monitoring, decision-making, and potential future features such as generating alerts or performance analysis.

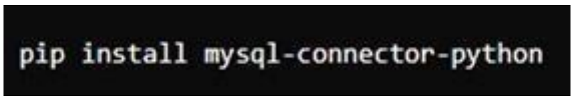

```
pip install mysql-connector-python
```

Figure33: database name

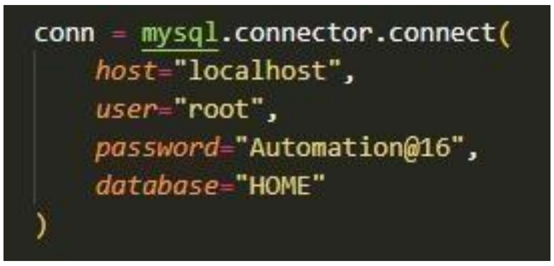
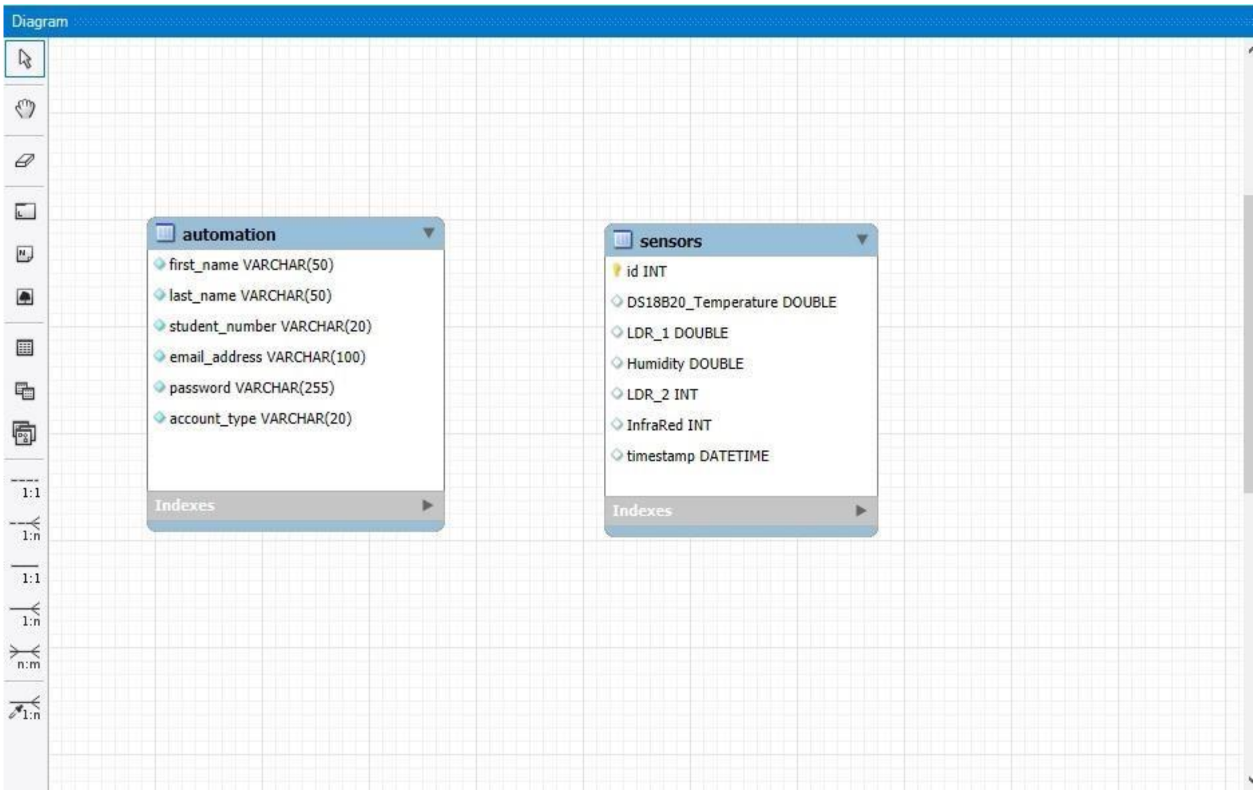
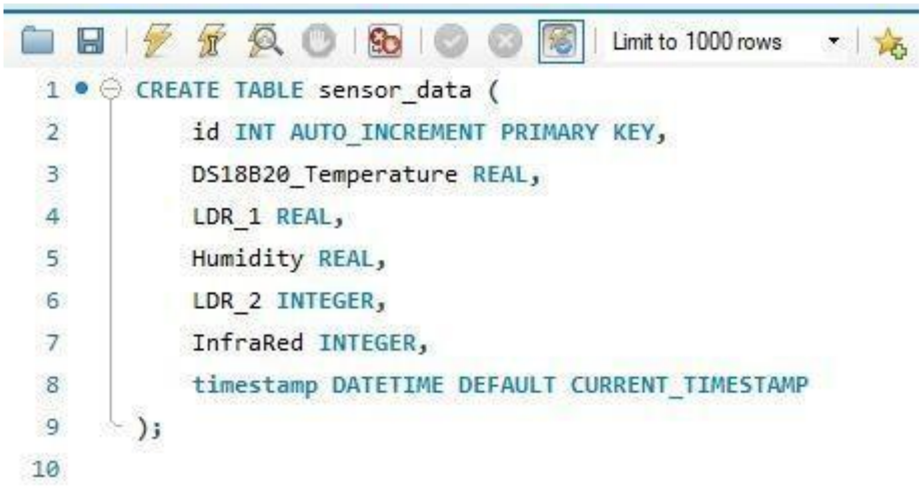

```
conn = mysql.connector.connect(
    host="localhost",
    user="root",
    password="Automation@16",
    database="HOME"
)
```

Figure34: shows how the database is connected python. The created database is called ‘’home’’



```
automation
first_name VARCHAR(50)
last_name VARCHAR(50)
student_number VARCHAR(20)
email_address VARCHAR(100)
password VARCHAR(255)
account_type VARCHAR(20)
```

```
sensors
id INT
DS18B20_Temperature DOUBLE
LDR_1 DOUBLE
Humidity DOUBLE
LDR_2 INT
InfraRed INT
timestamp DATETIME
```



```
CREATE TABLE sensor_data (
        id INT AUTO_INCREMENT PRIMARY KEY,
        DS18B20_Temperature REAL,
        LDR_1 REAL,
        Humidity REAL,
        LDR_2 INTEGER,
        InfraRed INTEGER,
        timestamp DATETIME DEFAULT CURRENT_TIMESTAMP
);
```

Figure35: these are tables created

Primary key: ensures data integrity and prevents duplicate sensor readings from having the same id.
Not null: Guarantees the integrity of the data by confirming that each user has a password and ID. And stops data that is not complete from being added to the table.
Foreign key: Requires that a value in the foreign key column match a value in the main key column of the referenced table to ensure consistency in table relationships.

4.3 Conclusion

The software development in this project successfully enabled communication between the ESP32 microcontroller, the database, and the web interface. The Arduino ide was used to write and upload code that collected data from all connected sensors, processed it, and transmitted it over Wi-Fi. On the server side, PHP and Python scripts handled data reception, while MySQL provided structured storage for easy access and analysis.

The front-end interface, built with html and css, allowed users to view live sensor data in a simple and readable format. The use of json enabled smooth data updates, improving the user experience. Basic login and signup features were also added to control access and protect data.

Despite challenges such as occasional Wi-Fi disconnection and the need for accurate data formatting, the software components worked as expected after debugging and code optimization. Overall, the software formed a stable and efficient bridge between the hardware and the user, fulfilling the system’s automation and monitoring goals.

The next chapter will focus on the results and discussion, where the performance of the entire system will be evaluated, including how well it handled real-world testing conditions and whether it met the intended objectives.

Chapter 5: Results and Discussion

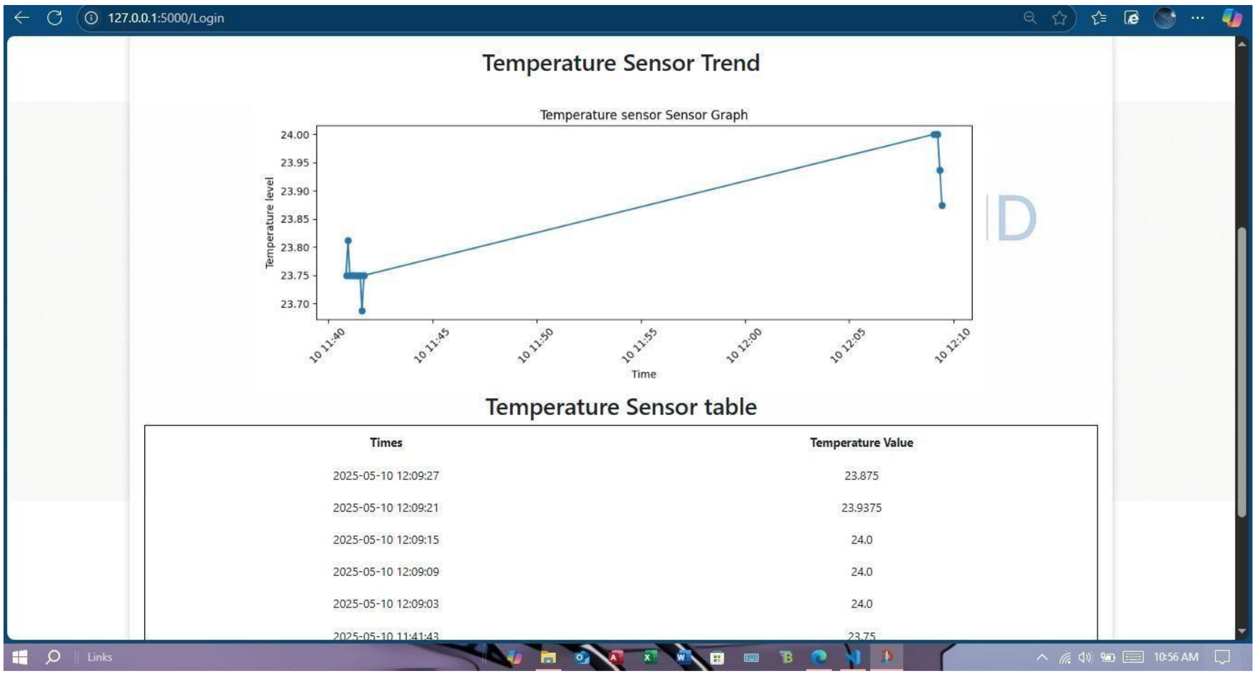
5.1 Introduction

This chapter presents the outcomes from testing the home automation and security system. Each component was tested both separately and as part of the complete system. The performance of each sensors was compared against actual conditions to check accuracy and responsiveness. The discussion focuses on how well the system met its intended goals and what could be improved for better performance.

5.2 Sensor Test Results

5.2.1Temperature sensor graph

The graph shows temperature values recorded in real time by the Dallas DS18B20 sensor. During testing, room temperature increased slightly due to human activity and sunlight. The sensor responded as expected, showing values in the 24–29°C range. This confirms that the sensor was properly adjusted and responded accurately to environmental changes. It was consistent with room thermometer readings taken for comparison, confirming good performance.



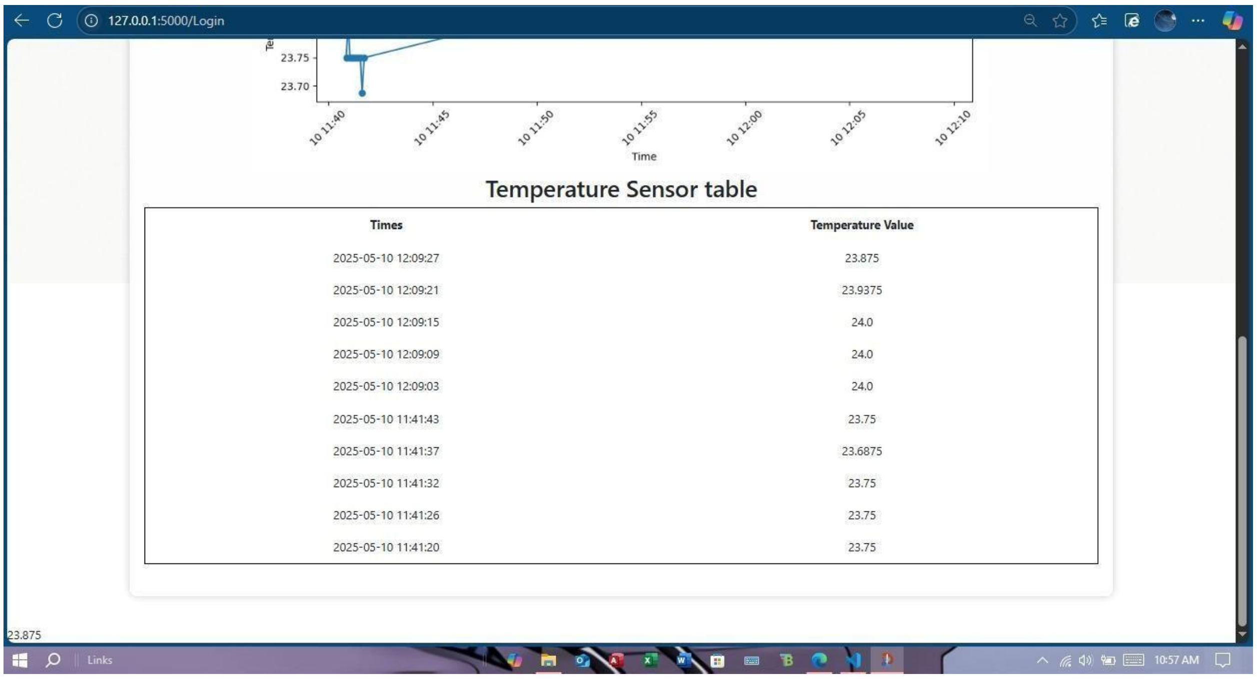
Temperature Sensor table

Times	Temperature Value
2025-05-10 12:09:27	23.875
2025-05-10 12:09:21	23.9375
2025-05-10 12:09:15	24.0
2025-05-10 12:09:09	24.0
2025-05-10 12:09:03	24.0
2025-05-10 11:41:43	23.75

Figure36: temperature sensor graph

5.2.2temperature sensor table

The table shows recorded temperatures and their timestamps. All values matched the expected range and changed smoothly over time. This shows that the sensor was working correctly and that the communication between the ESP32 and the server was stable. The data was successfully stored in the database and retrieved for display.



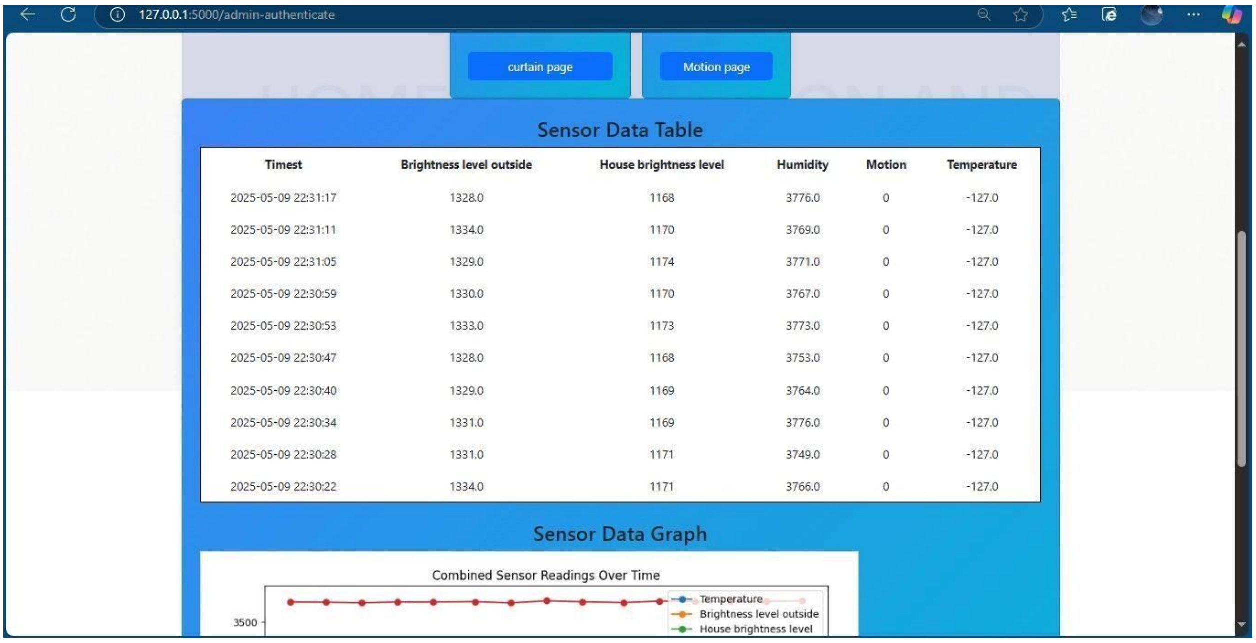
Temperature Sensor table

Times	Temperature Value
2025-05-10 12:09:27	23.875
2025-05-10 12:09:21	23.9375
2025-05-10 12:09:15	24.0
2025-05-10 12:09:09	24.0
2025-05-10 12:09:03	24.0
2025-05-10 11:41:43	23.75
2025-05-10 11:41:37	23.6875
2025-05-10 11:41:32	23.75
2025-05-10 11:41:26	23.75
2025-05-10 11:41:20	23.75

Figure37: temperature sensor table

5.2.3 admin sensor table

This table displays the latest readings from all sensors, visible to the admin. It includes values for temperature, humidity, light, and motion status. The interface made it easy to check system status. Data was updated in real time, showing that both frontend and backend were connected correctly and that users could rely on this view for live monitoring.

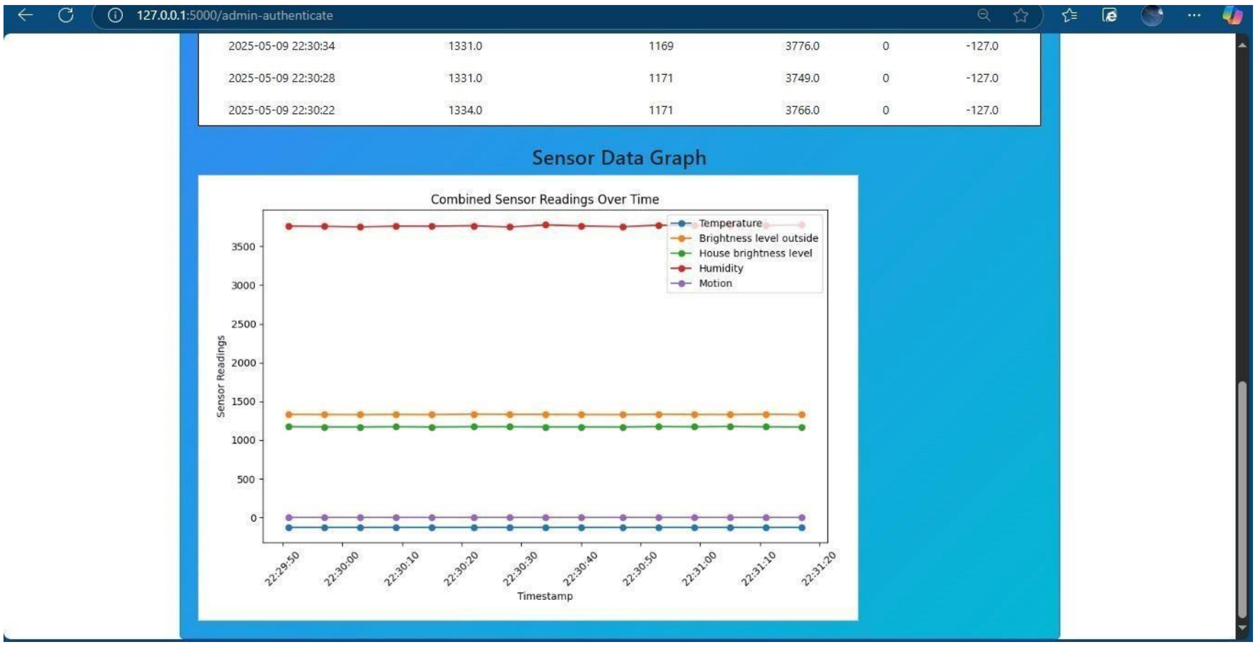


Timest	Brightness level outside	House brightness level	Humidity	Motion	Temperature
2025-05-09 22:31:17	1328.0	1168	3776.0	0	-127.0
2025-05-09 22:31:11	1334.0	1170	3769.0	0	-127.0
2025-05-09 22:31:05	1329.0	1174	3771.0	0	-127.0
2025-05-09 22:30:59	1330.0	1170	3767.0	0	-127.0
2025-05-09 22:30:53	1333.0	1173	3773.0	0	-127.0
2025-05-09 22:30:47	1328.0	1168	3753.0	0	-127.0
2025-05-09 22:30:40	1329.0	1169	3764.0	0	-127.0
2025-05-09 22:30:34	1331.0	1169	3776.0	0	-127.0
2025-05-09 22:30:28	1331.0	1171	3749.0	0	-127.0
2025-05-09 22:30:22	1334.0	1171	3766.0	0	-127.0

Figure38: admin sensor table

5.2.4 admin sensor graph

The graph provides a clear visual summary of recent sensor readings, specifically light intensity and temperature over time. It allows the administrator to monitor environmental changes throughout the day and recognize patterns such as increased heat during midday or reduced lighting in the evening. This combination of real-time and historical data supports better decision-making and demonstrates that the system reliably logs, stores, and displays sensor information. The successful presentation of this data also confirms that backend-to-frontend communication between the database and web interface is functioning correctly.


127.0.0.1:5000/admin-authenticate

2025-05-09 22:30:34	1331.0	1169	3776.0	0	-127.0
2025-05-09 22:30:28	1331.0	1171	3749.0	0	-127.0
2025-05-09 22:30:22	1334.0	1171	3766.0	0	-127.0

Sensor Data Graph

Figure39: admin graph

5.3 discussion of sensors behavior

- LDR**:** Testing with different light levels (by covering and uncovering the sensor) showed that LDR readings dropped in darkness and rose in sunlight. This change was smooth and stable, which shows the voltage divider was working well. The data helped in deciding when to open or close curtains automatically.
- Infrared: Manual walking tests were performed in front of the sensor. The output went HIGH when motion was detected and low when idle. The sensor responded quickly and accurately within a 2-3m range. This confirms it is suitable for indoor use, especially near doors or windows.
- Humidity Sensor: The sensor showed increasing values when water vapor was introduced nearby. The readings returned to normal after the source was removed. Although the sensor was more sensitive to changes than accurate in absolute values, it worked well for detecting general moisture trends.

The entire system responded correctly under different conditions. The ESP32 read all sensor values and updated the database in real time. The frontend showed the live data without delay, and the admin page remained functional during testing. All objectives including real-time monitoring, login access, and data displays were successfully achieved. As shown in Figure 18, the final prototype was assembled on a breadboard using all selected sensors and the ESP32. The layout followed the tested simulation setup and confirmed full functionality during live trials.

Chapter 6: Conclusions and Recommendations

6.1Conclusions

The home automation and security system project was successful in achieving its main objectives which is monitoring environmental data in real time, storing it in a database, and displaying it on a web interface. It met its core goals of detecting temperature, light, humidity, and motion while also allowing user access control through login and sign-up functionality. Despite limited resources, the system functioned as intended and produced reliable data when tested under different conditions.

Challenges and Solutions:

- ➢ Unstable LDR and humidity readings were observed during early testing. This was resolved by adjusting resistor values in the voltage divider circuits and double-checking all wiring connections on the breadboard.
- ➢ Wi-Fi instability on the ESP32 caused random disconnections, which interrupted data logging. This was addressed by adding reconnection logic to the Arduino code to automatically restore the connection when it failed.

One of the main limitations of the system was the absence of a backup power supply. Since the system relies on continuous power, it shuts down during outages, which reduces its reliability as a standalone security solution.

The overall outcome demonstrated that it is possible to build a low-cost and reliable home monitoring system using open-source tools and affordable components. The project also showed how different engineering fields which include electronics, embedded systems, networking, and web development can be combined to address practical, real-world challenges.

This project gave me hands-on experience in combining electronic hardware with software tools to solve real-world problems. I learned how important it is to carefully plan circuit design, write efficient code, and troubleshoot unexpected behavior during live testing. It was surprising to see how small adjustments such as resistor values or code timing—could significantly impact overall system performance. Overall, the experience helped me understand the real-world challenges of IoT system development and gave me the confidence to approach future engineering tasks with a structured and problem-solving mindset.

6.2 Recommendations / future work

If I were to repeat this project, I would make several improvements to enhance accuracy, reliability, and user experience by:

- ➢ Using higher-quality sensors with better resolution would improve data accuracy. For example, sensors with built-in digital compensation could reduce errors caused by temperature shifts or humidity changes. This would make the system more reliable in real-world conditions.

- recommend integrating a rechargeable backup battery or a small solar-powered module to keep the system running during power outages. This is especially important for a security system that must stay online at all times.
- On the software side, the web interface could be upgraded using frameworks that make it more interactive, mobile-friendly, and easier to use. This would improve the experience for users, especially on smartphones and tablets.
- For better alerts and user engagement, SMS notifications can be added using SMTP integration. This would notify the user immediately in case of motion detection or extreme environmental changes.
- future versions of the system could connect to the cloud so that users can monitor and control their home from anywhere using the internet. This can be done using platforms like Firebase. It would also allow data to be stored for longer periods and make it easier to track changes over time.

In future we can focus on expanding the system’s flexibility, reach, and intelligence. This includes exploring cloud integration for remote access, adding machine learning for predictive responses, and extending sensor coverage to detect other environmental risks such as gas leaks or air quality changes. These additions would make the system smarter, more secure, and adaptable to a wider range of real-world applications.

6.3System Evaluation

The project successfully met its key objectives as outlined in Chapter 1. Real-time sensor data was obtained and stored in a MySQL database, fulfilling the first objective. Data transmission from the ESP32 microcontroller to the backend was achieved using Wi-Fi and Python, addressing the second objective. A responsive and secure web interface was developed to display the sensor readings, meeting the third objective. During testing, sensor accuracy was evaluated under different environmental conditions, and hardware limitations were identified and resolved, satisfying the final objective. The successful completion of all project goals demonstrates the effectiveness of the proposed solution and validates its real-world application potential.

References

Afribary, 2023. How to write significance of the study in a project research paper. [Online] Available at: https://afribary.com/knowledge/how-to-write-significance-of-the-study-inaproject/ [Accessed 12 Feb 2023].

Blog, L.S., n.d. Research Methods. [Online] Available at: https://lovestats.wordpress.com/dman/ [Accessed 4 Oct 2018].

CircuitSpecialists, 2023. 13 Best Arduino Projects. [Online]
Available at: https://www.circuitspecialists.com/blog/best-arduino-projects/ [Accessed 2 Feb 2023].

Editage Insights, 2016. How do I find reliable scientific literature on the Internet? [Online] Available at: https://www.editage.com/insights/how-do-i-find-reliable-scientificliteratureon-theinternet [Accessed 28 July 2018].

Editage Insights, 2017. What is the best way of stating the background of a study? [Online] Available at: https://www.editage.com/insights/what-is-the-best-way-of-statingthebackgroundof-a-study
[Accessed 3 August 2018].

Dallas Semiconductor, 2023. DS18B20 Datasheet. [Online] Available at: https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf

SparkFun, 2022. DHT11 Humidity Sensor Datasheet. [Online] Available at: https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT11.pdf

SparkFun, 2022. PIR Motion Sensor Datasheet. [Online] Available at: https://cdn.sparkfun.com/datasheets/Sensors/Proximity/SE-10.pdf

Components101, 2023. LDR - Light Dependent Resistor. [Online] Available at: https://components101.com/sensors/ldr-light-dependent-resistor

WeMos, 2023. ESP32 D1 R32 Documentation. [Online] Available at: https://docs.wemos.cc/en/latest/d1/d1_mini_pro.html

Editage Insights, 2019. How to write research objectives? [Online]
Available at: https://www.editage.com/insights/how-to-write-research-objectives-in-point [Accessed 2 Feb 2023].

Editage Insights, 2019. What are the limitations of a study and how to write them? [Online] Available at: https://www.editage.com/insights/what-are-limitations-in-a-study [Accessed 2 Feb 2023].

LLS Monyela, 2025. Design and Implementation of a Home Automation and Security System. Project report submitted for the Diploma in Electrical Engineering, Vaal University of Technology.

MasterClass, 2023. Problem Statement Examples. [Online]
Available at: https://www.masterclass.com/articles/problem-statement [Accessed 11 Feb 2023].

MOBOTIX, 2022. Products. [Online]
Available at: https://www.mobotix.com/en/products [Accessed 31 Jan 2023].

MOBOTIX, 2022. Thermal Technology. [Online]
Available at: https://www.mobotix.com/en/thermal-technology [Accessed 31 Jan 2023].

MOBOTIX, 2022. Interface Boxes and Sensors. [Online]
Available at: https://www.mobotix.com/en/products/system-components/interface-boxessensors [Accessed 31 Jan 2023].

National Library of Medicine, 2012. The Development of Remote Monitoring and Guideline System for Safer Client Care in the Community. [Online]
Available at: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3799144/ [Accessed 5 Feb 2023].

ResearchGate, 2020. Arduino Mega Board Pin-out. [Online]
Available at: https://www.researchgate.net/figure/Arduino-Mega-board-pin-outArduinoprovides-adevelopment-environment-based-on-open_fig3_344596556 [Accessed 12 Feb 2023].

YourDictionary, 2023. Problem Statement Examples. [Online]
Available at: https://examples.yourdictionary.com/problem-statement-examples.html [Accessed 3 Feb 2023].

Appendix

Appendix A: Data Sheets of Key Components

DS18B20 Temperature Sensor

Digital temperature sensor with 1-Wire interface

Operating voltage: 3.0V to 5.5V

Accuracy: ±0.5°C from -10°C to +85°C

Source: https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf

MH-Series Humidity Sensor (DHT11 equivalent)

- Operating voltage: 3.3V to 5V
- Humidity range: 20–90% RH (±5% accuracy)
- Source: https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT11.pdf

2. LDR (Light Dependent Resistor)
   - Resistance decreases with increased light intensity
   - Used in voltage divider circuit with 10kΩ resistor
   - Source: https://components101.com/sensors/ldr-light-dependent-resistor
3. Passive Infrared (PIR) Motion Sensor
   - Operating voltage: 5V
   - Output: Digital HIGH (motion detected), LOW (no motion)
   - Source: https://cdn.sparkfun.com/datasheets/Sensors/Proximity/SE-10.pdf
4. ESP32 Wemos D1 R32 Board
   - Dual-core 32-bit processor with built-in Wi-Fi
   - Multiple GPIO pins, analog and digital
   - Source: https://docs.wemos.cc/en/latest/d1/d1_mini_pro.html