NILE UNIVERSITY

Smart Home & Security System

An ECE241 Project Report

By

Ahmed Nour - 231002375

Date
31-05-2025

Table of Contents

ABSTRACT

This project presents the design and implementation of a smart home and security system using ESP32 microcontrollers. The system monitors environmental data such as temperature, humidity, gas (butane) concentration, and light intensity, while also offering a live camera stream and user-authenticated lock system. It integrates sensors, actuators, and communication modules to create an interactive and responsive environment. The collected data is displayed on a local web interface, allowing real-time monitoring and control. The solution demonstrates how embedded systems, IoT, and web development can be combined to provide cost-effective smart home automation with enhanced security.

Keywords: Smart Home, ESP32, IoT, DHT Sensor, Butane Sensor, LDR, WiFi Camera, Authentication, Python, HTML Dashboard

SECTION I: Introduction

The demand for intelligent and connected home environments has increased significantly in recent years. As urbanization and technological adoption grow, the need for homes that are not only livable but also responsive to environmental and security challenges becomes more apparent. Smart homes aim to provide a blend of comfort, convenience, safety, and energy efficiency by automating tasks such as monitoring air quality, lighting, and access control.

With the evolution of microcontrollers and wireless communication technologies, it has become feasible to develop integrated systems that collect, transmit, and visualize sensor data in real-time. This project implements a smart home and security system using ESP32 microcontrollers, which serve as the backbone for data acquisition and device control. The system is equipped with various sensors to monitor key environmental parameters such as temperature, humidity, light levels, and butane concentration. Additionally, it includes a surveillance camera for real-time video streaming and a user authentication system based on password input to control access to restricted areas.

Key motivations for this project include:

- Real-time sensing and alerting environmental conditions (e.g., fire hazards, poor air quality).
- Secure and user-friendly access control using a keypad-password mechanism.
- Live video monitoring through a Wi-Fi-enabled ESP32 camera module.
- A simple web-based user interface for visualizing captured sensor data.

Our goal is to develop a modular, scalable, and cost-effective smart home system that combines hardware reliability with software flexibility. The system is designed to offer accurate sensor readings, security features, and an intuitive interface suitable for household deployment. It also serves as a demonstration of how embedded systems and IoT concepts can be combined in real-world applications to improve everyday living environments.

SECTION II: System Architecture & Design

The system is structured into two major subsystems: one focused on environmental monitoring and surveillance, and the other on secure access control.

The first subsystem comprises an ESP32 microcontroller connected to three sensors: a DHT sensor to measure temperature and humidity, an MQ-2 sensor to detect gas and butane concentration, and an LDR to monitor light intensity. Additionally, it includes a WiFi camera module that provides live video monitoring. All these components are powered by a regulated power supply board to ensure stable operation.

The second subsystem is responsible for authentication and lock control. It also utilizes an ESP32 microcontroller connected to a 4x4 matrix (chars and nums) keypad and a servo motor, which functions as a locking mechanism. Users are required to enter predefined usernames and passwords using the keypad. When the input matches one of the stored combinations, the servo motor activates to unlock the system. This subsystem is powered by a separate power supply to isolate control logic and prevent interruptions.

Data communication across components is divided into two channels. The environmental sensor readings are transmitted via Bluetooth from the monitoring ESP32 to the server. Meanwhile, the camera feed is transmitted over Wi-Fi and accessed through an IP stream. The Python script processes all incoming data and displays it on a local HTMLbased dashboard. This interface updates live sensor readings approximately every 30 seconds, allowing real-time visualization and interaction.

The authentication logic is simple yet effective. Three users (A, B, and C) are predefined with their respective username-password combinations: (a, 123), (b, 456), and (c, 789). The system continuously monitors the keypad for input. Upon successful verification of credentials, the ESP32 triggers the servo motor to unlock the mechanism. Incorrect entries are ignored or flagged for security review.

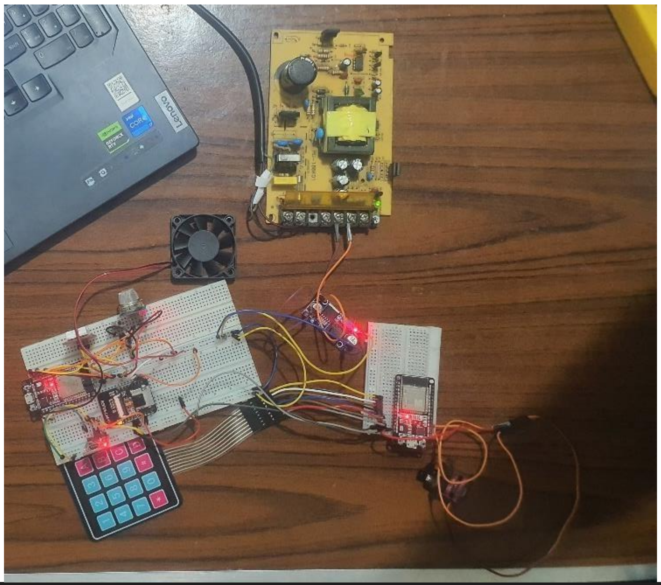
The user interface presents several key components to facilitate usability and data representation. It includes a real-time display of sensor values, graphs that visualize historical trends for temperature, humidity, gas levels, and light intensity, and a dedicated camera stream page. Additionally, an information tab is included to present group member details and project context.

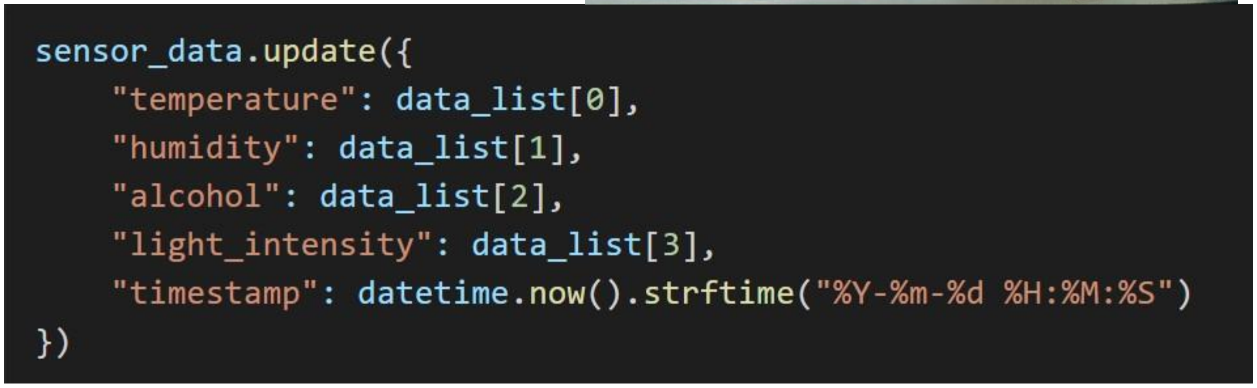
SECTION III: Implementation

The implementation phase of the project involved developing and integrating multiple hardware and software components to function together as a unified smart home system. Each module was programmed and tested independently before final integration.

The first ESP32 microcontroller was used to create the sensor node. This node connected to three primary sensors: a DHT11 sensor for measuring temperature and humidity, an MQ-2 sensor for detecting butane gas concentrations, and an LDR for monitoring ambient light intensity. The data collected by these sensors was read through the ESP32's analog and digital pins. Once acquired, the sensor values were formatted into a CSV-style string, and transmitted via Bluetooth to the Flask-based Python server.

On the server side, the Python script (app.py) served multiple critical functions. It handled Bluetooth serial communication using the PySerial library, continuously reading incoming data from the ESP32. It also parsed and stored this data in a CSV file for later reference and analysis. The script created a live data endpoint (/data) using Flask and hosted a local web interface accessible on port 5000. This interface was kept up to date via a background thread that updated sensor readings in real-time. A relevant code snippet demonstrates the way the server updates the sensor dictionary with new data and timestamps:





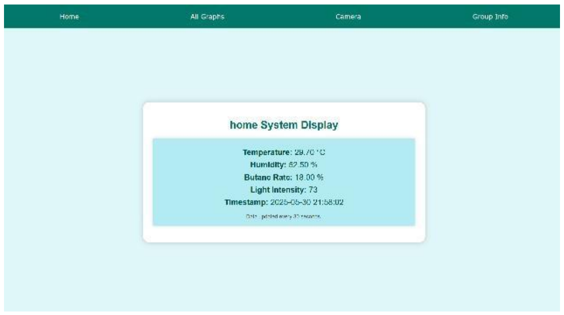
```
sensor_data.update({
    "temperature": data_list[0],
    "humidity": data_list[1],
    "alcohol": data_list[2],
    "light_intensity": data_list[3],
    "timestamp": datetime.now().strftime("%Y-%m-%d %H:%M:%S")
})
```

The camera node consisted of an ESP32 equipped with an OV2640 camera module. This unit was configured to operate on a fixed IP address via Wi-Fi and continuously stream video. The video feed was embedded into the local web dashboard, allowing users to monitor the environment visually. Users could access and toggle the live stream from the main interface.

In parallel, the authentication and access control module was implemented using another ESP32 connected to a 4x4 matrix keypad and a servo motor. This subsystem processed user input and compared it against three hardcoded username-password combinations. When a valid combination was entered—such as "a" with "123"—the ESP32 generated a PWM signal to rotate the servo motor, effectively unlocking the system. This action could also be logged for additional auditing or feedback in future versions.

SECTION IV: Results & Discussion

The system was evaluated based on its responsiveness, accuracy, and overall stability. During testing, the sensor node delivered real-time environmental readings including a temperature of 29.70 °C, humidity at 52.50%, butane gas concentration at 18.00%, and light intensity measured at 73 units. These values were monitored both live and over time through graphical representations on the dashboard.

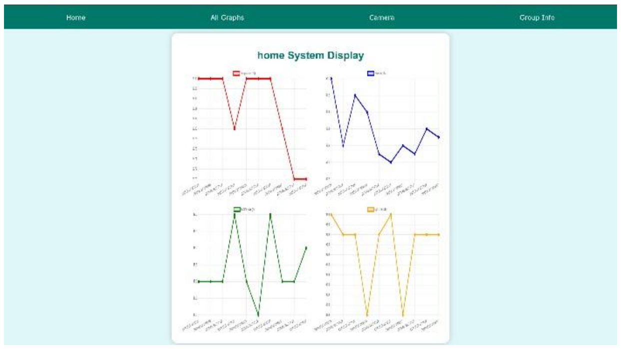


home System Display

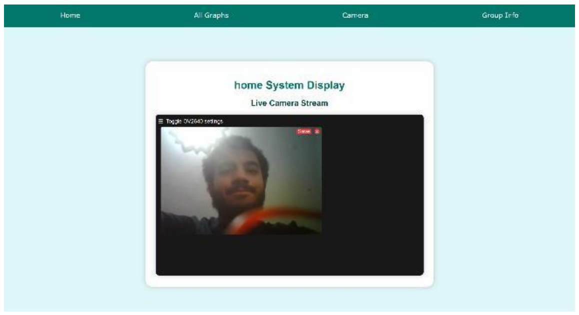
Temperature: 29.70 °C
Humidity: 52.50 %
Butane Rate: 18.00 %
Light Intensity: 73
Timestamp: 2025-05-30 21:58:02

Data updated every 30 seconds.

The authentication module was rigorously tested with all three predefined user-password combinations. Each valid combination successfully triggered the servo motor to unlock the system, while incorrect entries were effectively rejected. This validated the robustness of the password verification logic.



The live video stream provided by the ESP32-CAM was accessible via a local IP address. It offered clear visuals with minimal delay, even under varying lighting conditions, thus validating the surveillance aspect of the system.



However, the development process was not without its challenges. Data inconsistencies occasionally occurred due to incomplete serial transmissions, requiring additional error-checking routines. Bluetooth communication faced minor instability, particularly when reconnecting after disconnections. Finally, building a cross-platform HTML interface involved adjustments to ensure compatibility across browsers and screen sizes, especially when integrating Bootstrap elements.

SECTION V: Conclusion

The smart home and security system developed in this project successfully integrates environmental sensing, access authentication, and video surveillance into a unified and practical embedded system application. The project demonstrated the potential of ESP32 microcontrollers in creating affordable and functional IoT-based home automation systems.

Future enhancements to this project could include migrating data to cloud-based storage for remote access, implementing alert notifications for dangerous gas levels or unauthorized access, and expanding the authentication system to include biometric or face recognition methods. Integration with popular home automation ecosystems like Google Home or Amazon Alexa could also provide greater user convenience. Furthermore, adding motion detection to the camera module would enhance the surveillance component.

REFERENCES

1. ESP32 Technical Reference Manual
2. DHT11 Datasheet
3. MQ-2 Gas Sensor Datasheet
4. Flask Documentation: https://flask.palletsprojects.com/
5. PySerial Documentation: here
6. OV2640 Camera Module Setup Guide

Appendix

The models and code for the Security and Smart Home system can be accessed via the following link:

https://drive.google.com/drive/folders/1iLy0hhGCsV5aN7Efjmyp3YFj8fuVtCs?usp=sharing