Automated Home Climate Control System

IIoT Course Project Report

Team 46:
Kushagra Agrawal - 2024101110
Devshree Vyas - 2024111003
Samarth Rao - 2024111035

May 6, 2025

1 Introduction

Our project, the Automated Home Climate Control System, regulates room temperature, humidity, CO_2 levels, and occupancy in real time. The system aims to significantly improve comfort and save energy. We use multiple sensors and actuators, controlled via ESP-32 and visualized through a web dashboard.

2 Project Objectives

- Automatically adjust indoor climate based on real-time sensor data.
- Enable remote monitoring and control of all devices through a web interface from anywhere in the world.
- Reduce power consumption by operating the fan and AC only as required.

3 System Overview

- **Sensors:**
  - **DHT22** - Measures outdoor temperature and humidity. Calibrated using local temperature data available through various sources.
  - **MQ135** - Monitors CO_2 levels(in ppm) and air quality. Calibrated proportionately according to people entering and leaving the room, and other data available.
  - **Ultrasonic & Infrared Sensors** - Used for occupancy detection. Calibrated by trying various configurations to match perfect sensitivity for error-free detection.
- **Actuators:**
  - **Fan (Stepper Motor)** - Helps in cooling when temperature exceeds a certain threshold.
  - **Automatic Window (Servo Motor)** - Opens/closes based on air quality and temperature. Automatically closes when AC is on.
  - **AC (Represented via LEDs)** - Uses 3 LED's to represent the temperature at which the AC is set (19-25°C).
- **Microcontroller:** ESP32 for integrated Wi-Fi and low power consumption. (Qty: 3)

- **Communication:** MQTT protocol with Mosquitto broker for lightweight and low-latency messaging.
- **Software Stack:** Arduino IDE (C/C++), Python (backend), HTML/CSS/JS (web dashboard).

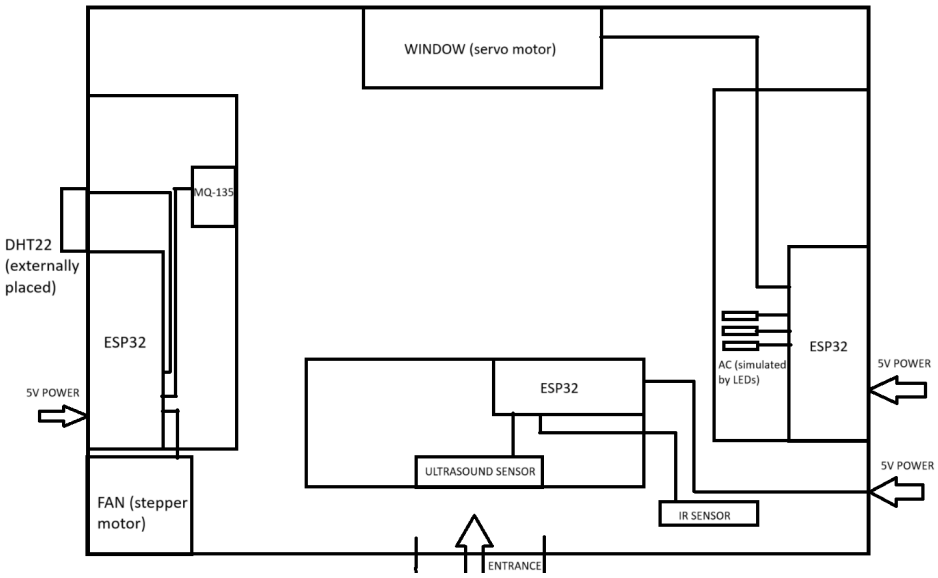

Figure 1: Schematic diagram

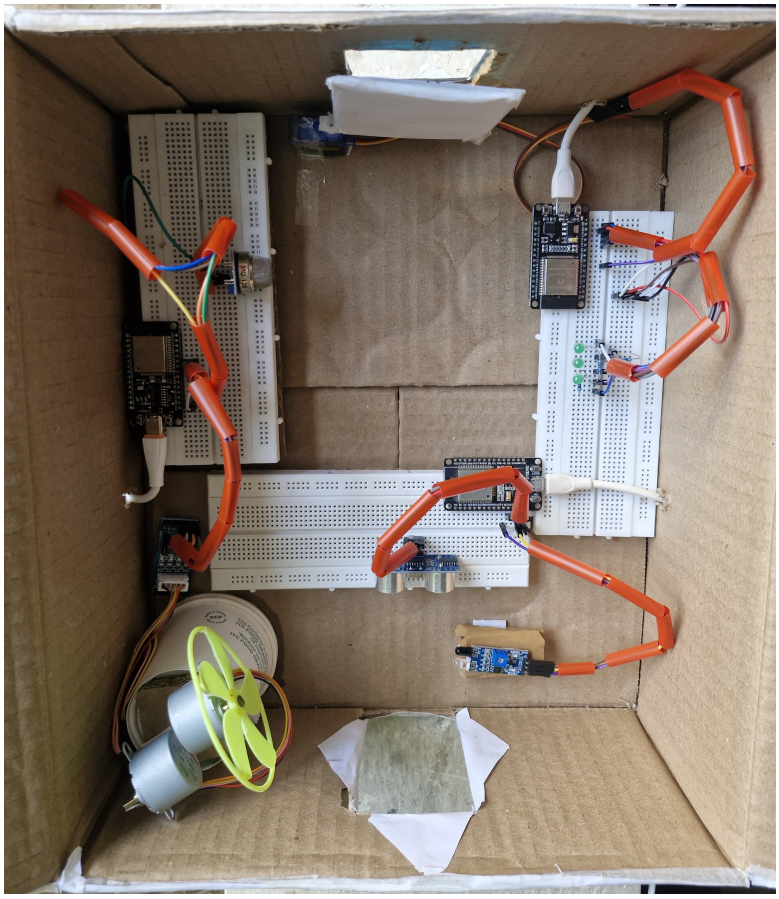

Figure 2: Hardware Setup of the Climate Control System

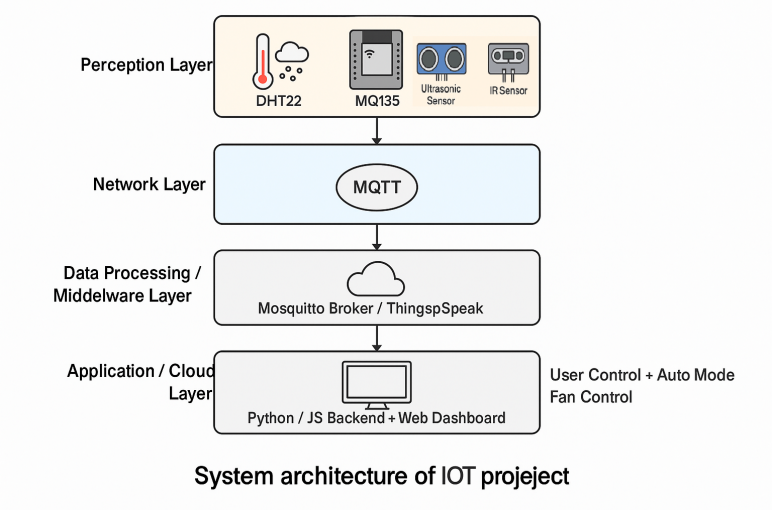

Figure 3: System Architecture of the Automated Climate Control System

4 Implementation Details

4.1 Automatic Mode

- **Code Logic:** The ESP32 continuously reads real-time temperature, humidity, CO_2 data, and occupancy count. These values are evaluated against predefined thresholds. Based on the logic, control signals are automatically issued to actuators: the fan, air conditioner, and automated window system. Here are the threshold values which we set:

  AC: The AC will be switched on if the temperature is more than 28°C and $occupancy > 0$.

  Fan: The fan will be switched on if the temperature is between 23°C and 27°C and $occupancy > 0$.

  Window: The window will be switched on if the CO_2 levels of the room is more than 130 units.

- **MQTT Communication:** When conditions are met, the ESP32 publishes actuator commands via MQTT to the appropriate topics.

- **Broker:** A Mosquitto MQTT broker is used to handle all communication between devices, ensuring reliable and lightweight messaging.

4.2 Manual Mode

- **User Control:** In manual mode, users can override automatic controls through a secure web interface. This mode allows authenticated users to operate home appliances remotely.

- **Web Dashboard:** The dashboard, built using HTML/CSS/JS, displays live environmental metrics such as temperature, humidity, occupancy, and CO levels. It also shows the current status of all actuators (AC/Fan ON/OFF, Window Open/Closed).

- **Backend Integration:** Python scripts manage backend logic and integrate user commands with the MQTT broker, allowing remote monitoring and control of the actuators.

5 Project Repository

The full source code, hardware schematics, and documentation are available on GitHub:
github.com/kushagra1310/IIOT_PROJECT

6 Results and Data Visualization

Attached is our ThingSpeak data, visualized as a graph

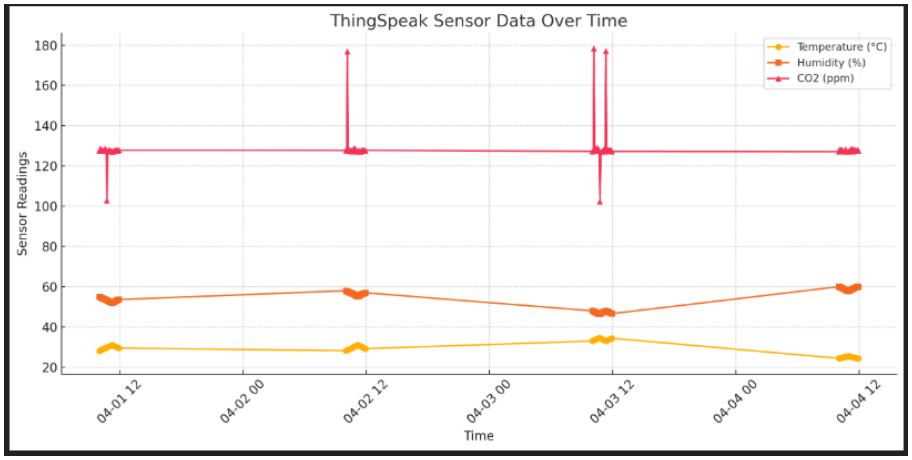

Figure 4: Indoor Temperature, Humidity and CO_2 level plot

On analysing data collected over multiple days and multiple times, we analysed and found the threshold of temperature for AC to be turned on to be best suited at 28°C, based on human feedback. Also the CO_2 level (ppm) threshold was adjusted to 130 on conducting similar tests. Tracking number of people entering and leaving the room and analysing the other data along with that allowed us to safely calculate thresholds required for the control of different actuators in auto-mode. Data collected is stored in a csv file in the datacollection directory of the github repository attached.

7 Challenges Encountered

- **Connectivity Issues:** Maintaining stable MQTT and Wi-Fi communication during prolonged operation was challenging.
- **Sensor Calibration:** Inaccuracies in DHT22 and MQ135 readings required manual calibration and filtering.
- **Power Management:** Efficient power usage, particularly for actuators, had to be considered to prevent energy waste.
- **Deploying the website:** To host the website and manage mqtt communication integrating both manual and auto modes was challenging.
- **Deugging code errors:** The code for website, being extremely long was prone to errors and fixing them while making changes to the project, and eventually making them readable and error proof was tough.
- **Data collection:** Data collection was another challenge, and it needed people entering and leaving the room.

8 Key Learnings

- Gained hands-on experience in integrating sensor networks with control systems.
- Understood the importance of stable, low-latency communication protocols in IoT.
- Learned to create and optimise control logic for IoT systems.
- Improved skills in debugging and managing concurrent sensor inputs in real time.

9 Conclusion and Future Work

- Integration of machine learning for predictive control.
- Voice assistant compatibility (e.g., Alexa, Siri).
- Solar-powered modules for renewable energy support.
- Expanding system capabilities to manage multi-room setups.

Figure 5: Thank You!