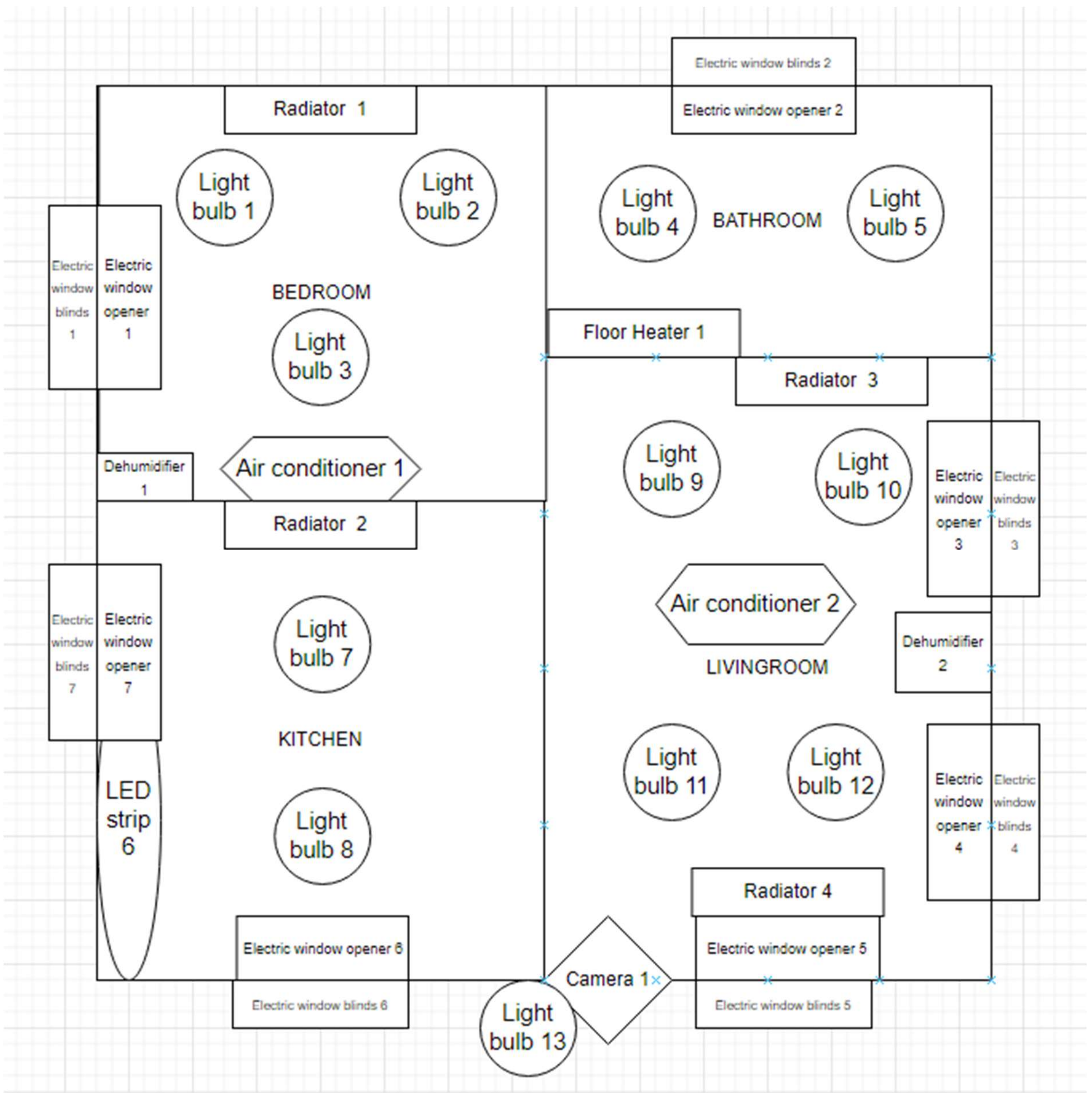
# SMART HOME AUTOMATED CONTROL AND MONITORING SYSTEM

## Introduction

The Smart Home Automated Control and Monitoring System is an IoT-based solution that enhances home automation by providing intelligent control and monitoring. It offers homeowners a seamless platform to manage various aspects of their home environment. Designed with flexibility, the system is based on a model home layout, with an example developed for a small house.



However, it can easily be adapted to fit different homes and apartments. This documentation provides a comprehensive guide to the system's architecture, components, functionalities, and implementation for developers, engineers, and users.

## Functional Requirements

1. Sensor Integration - Simulate or use real sensors to collect data. The data should be published periodically with timestamps and unique IDs.
2. Communication Protocols - Use a lightweight messaging protocol for communication. Define topics for data publishing and subscribing.
3. Data Processing - Process the data to aggregate, transform, and calculate important metrics like alerts or environmental indices.
4. Data Storage - Store sensor data in a time-series format, including tags like timestamps, sensor type, and location.
5. Visualization- Create user-friendly dashboards to display real-time data, provide filtering options, and show alert indicators.
6. Alerting Mechanisms - Set customizable thresholds for alerts and notify users through various communication channels.

## Non-Functional Requirements

1. Portability - Ensure the system can be easily deployed across different environments.
2. Scalability - Support the addition of more sensors and locations without requiring major changes to the system.
3. Resilience - Ensure the system can handle failures in communication while maintaining data integrity.
4. User-Friendly Design - Provide intuitive interfaces and clear documentation to guide users.
5. Security - Include necessary security features such as authentication to protect data and communications.

## Technologies Used

The following technologies were utilized for the development of the project:

1. **Python**: A versatile, high-level programming language known for its simplicity, readability, and wide range of third-party packages.

2. **Docker**: An open-source platform that uses containerization to package applications with all dependencies, ensuring seamless operation across different environments.

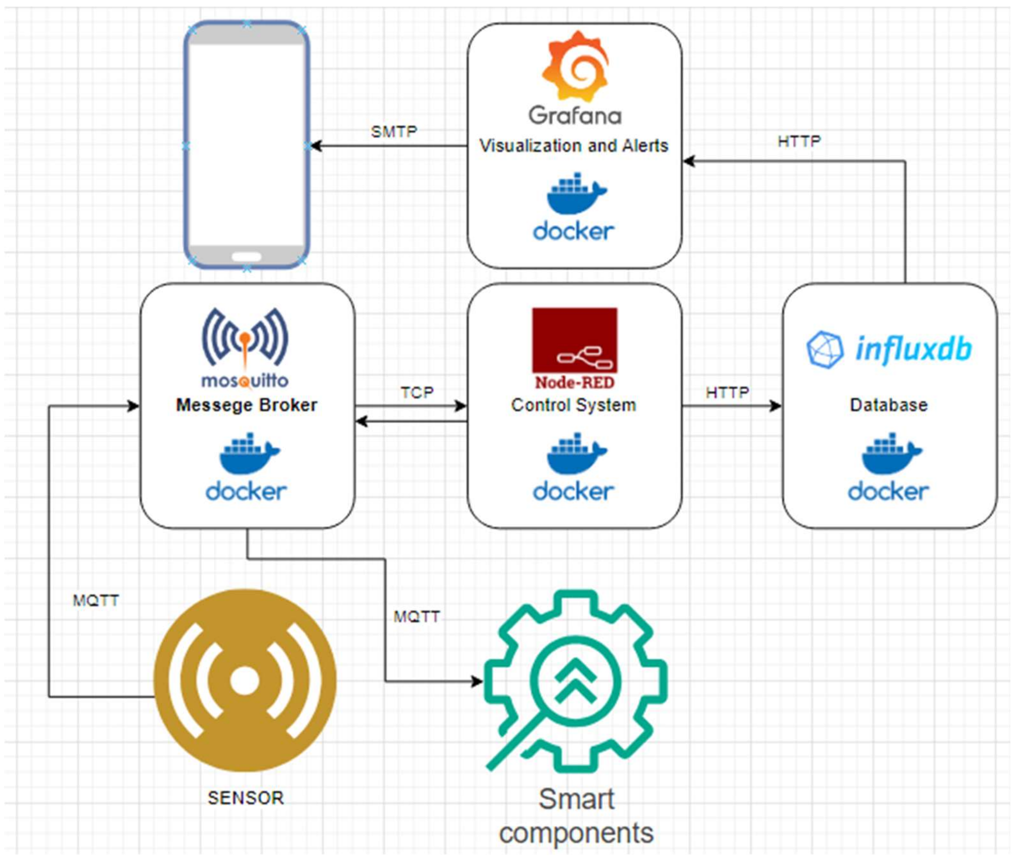
3. **Node-Red**: A browser-based programming tool that allows easy flow wiring and deployment using a wide range of nodes.

4. **Mosquitto**: An open-source MQTT message broker that enables communication between IoT devices by supporting publish/subscribe messaging.

5. **InfluxDB**: An open-source time-series database designed for high write and query loads, ideal for storing and analyzing IoT sensor data.

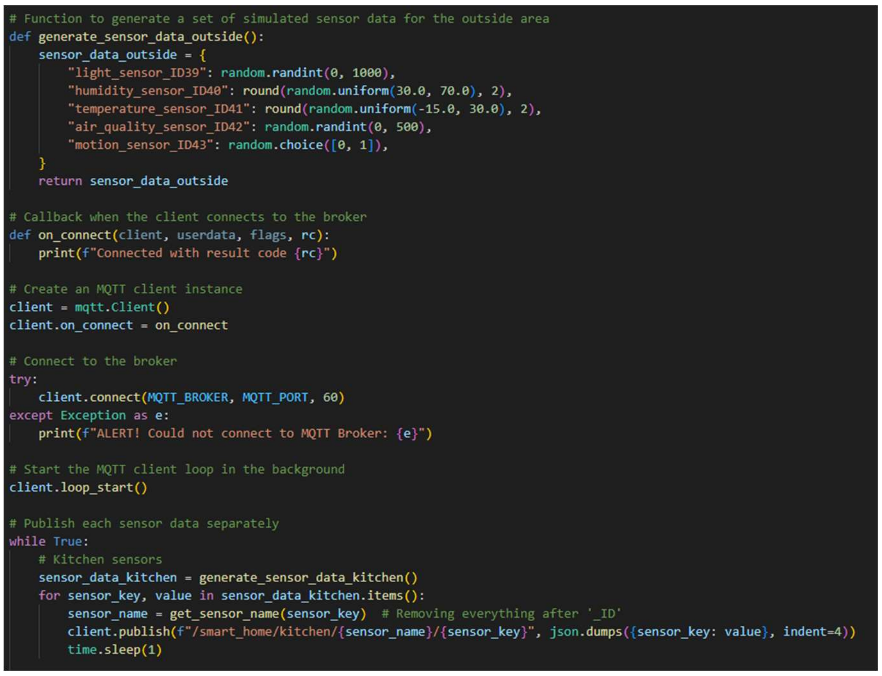
6. **Grafana**: An open-source web application for interactive data visualization, providing charts and alerts when connected to time-series databases like InfluxDB.

## System Architecture



## System Functionality

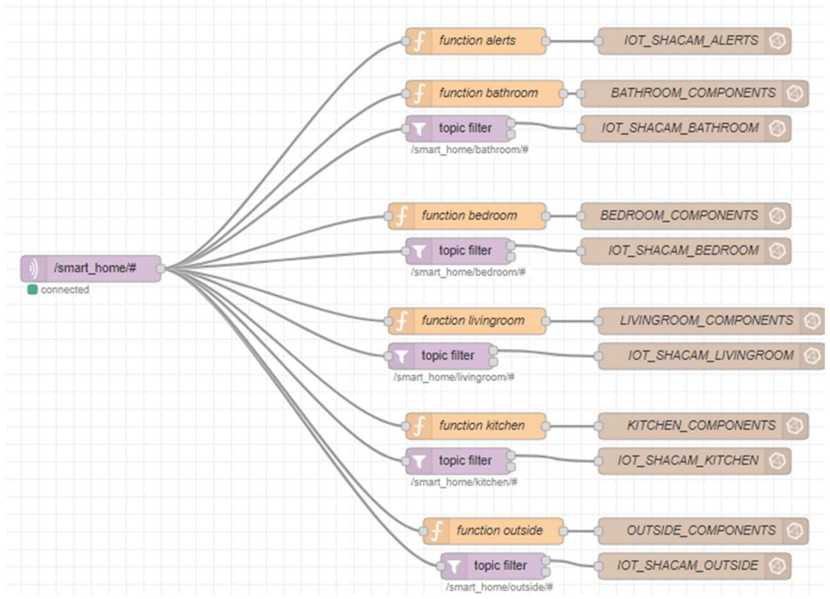
1. **Sensor Data Generation**
   Using Python, we generate data that simulates real sensor readings and convert this data into JSON files.



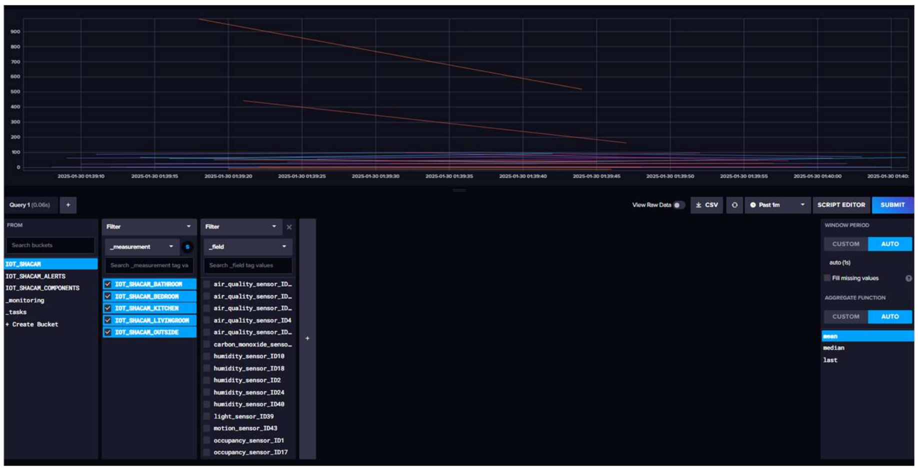
```python
# Function to generate a set of simulated sensor data for the outside area
def generate_sensor_data_outside():
    sensor_data_outside = {
        "light_sensor_ID39": random.randint(0, 1000),
        "humidity_sensor_ID40": round(random.uniform(30.0, 70.0), 2),
        "temperature_sensor_ID41": round(random.uniform(-15.0, 30.0), 2),
        "air_quality_sensor_ID42": random.randint(0, 500),
        "motion_sensor_ID43": random.choice([0, 1]),
    }
    return sensor_data_outside

# Callback when the client connects to the broker
def on_connect(client, userdata, flags, rc):
    print(f"Connected with result code {rc}")

# Create an MQTT client instance
client = mqtt.Client()
client.on_connect = on_connect

# Connect to the broker
try:
    client.connect(MQTT_BROKER, MQTT_PORT, 60)
except Exception as e:
    print(f"ALERT! Could not connect to MQTT Broker: {e}")

# Start the MQTT client loop in the background
client.loop_start()

# Publish each sensor data separately
while True:
    # Kitchen sensors
    sensor_data_kitchen = generate_sensor_data_kitchen()
    for sensor_key, value in sensor_data_kitchen.items():
        sensor_name = get_sensor_name(sensor_key)  # Removing everything after '_ID'
        client.publish(f"/smart_home/kitchen/{sensor_name}/{sensor_key}", json.dumps({sensor_key: value}, indent=4))
        time.sleep(1)
```

2. **Data Communication**
   Using the MQTT protocol with the Paho library and the Mosquitto MQTT broker, we establish communication between the sensor simulator and the Node-Red platform.

3. **Data Processing**
   On the Node-Red platform, we filter the received data and perform functions that include threshold violations calculations. Then, we send the processed data to the InfluxDB database.



4. **Data Storage**
   The processed data is then stored in the InfluxDB database, which retains the data for further visualization.



Query 1 (0.06s)

FROM: IOT_SHACAM

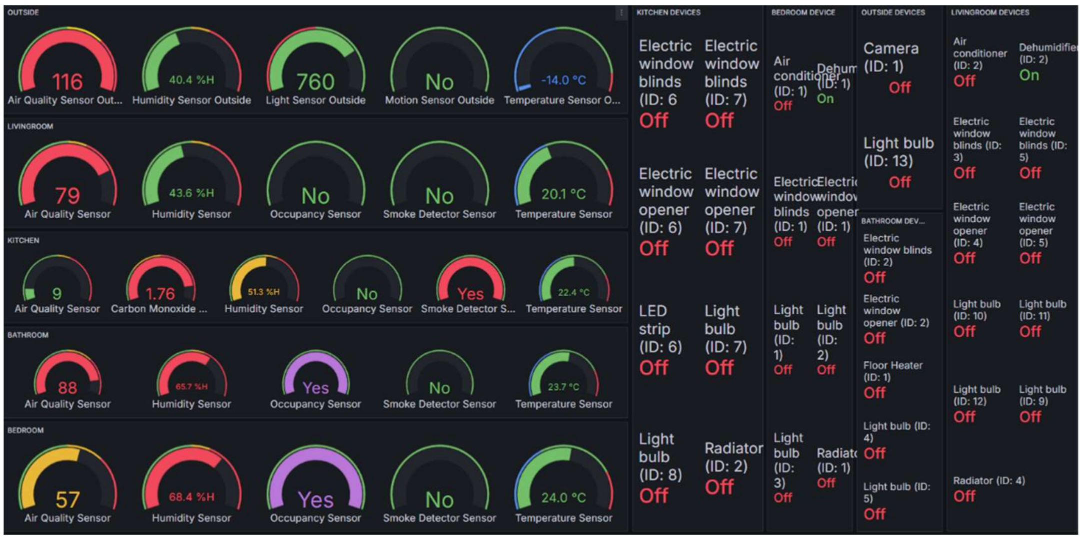
Filter: _measurement — IOT_SHACAM_BATHROOM ✓, IOT_SHACAM_BEDROOM ✓, IOT_SHACAM_KITCHEN ✓, IOT_SHACAM_LIVINGROOM ✓, IOT_SHACAM_OUTSIDE ✓

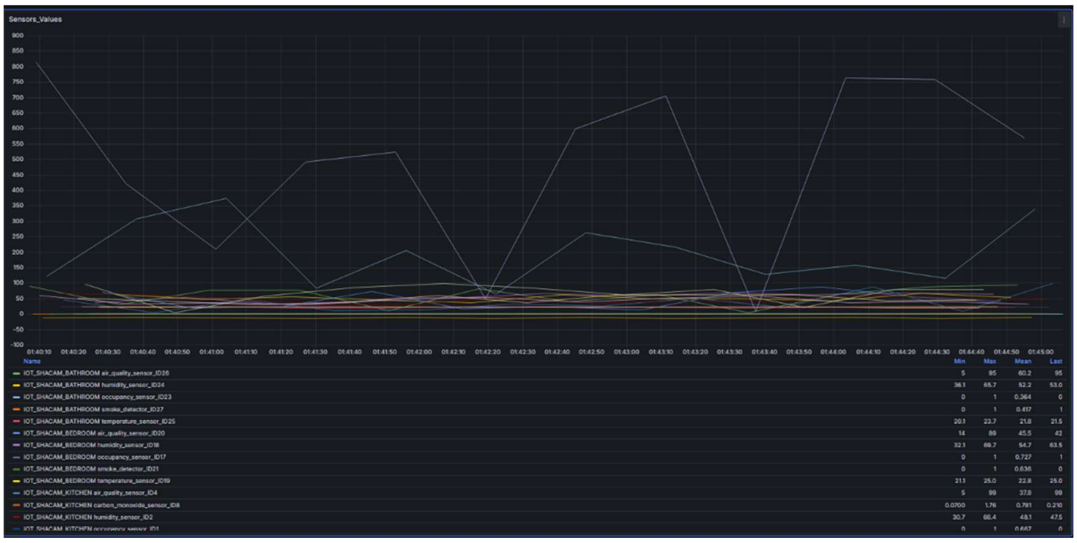
Filter: _field

WINDOW PERIOD: AUTO — auto (1s)

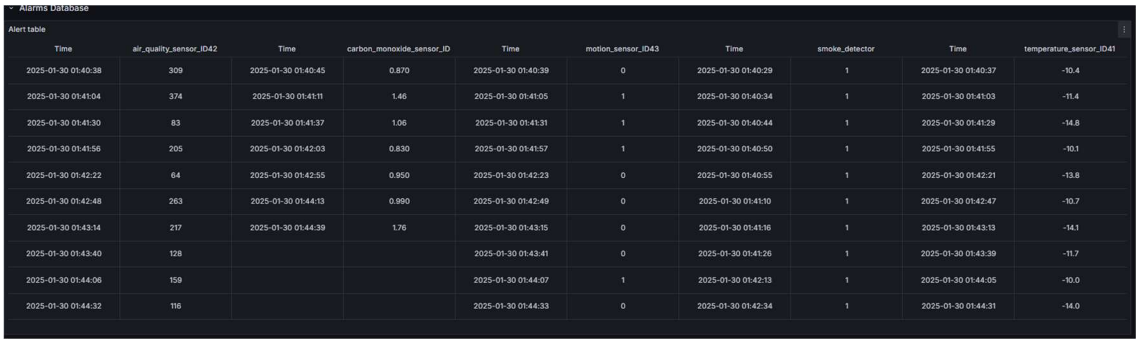
AGGREGATE FUNCTION: AUTO — mean

5. **Data Analysis and Visualization**
   The final data is presented in a user-friendly interface in Grafana, displayed as charts and indicators for easy analysis and monitoring.





| Name | Min | Max | Mean | Last |
|---|---|---|---|---|
| IOT_SHACAM_BATHROOM air_quality_sensor_ID26 | 5 | 95 | 60.2 | 95 |
| IOT_SHACAM_BATHROOM humidity_sensor_ID24 | 36.1 | 65.7 | 52.2 | 53.0 |
| IOT_SHACAM_BATHROOM occupancy_sensor_ID23 | 0 | 1 | 0.364 | 0 |
| IOT_SHACAM_BATHROOM smoke_detector_ID27 | 0 | 1 | 0.417 | 1 |
| IOT_SHACAM_BATHROOM temperature_sensor_ID25 | 20.1 | 23.7 | 21.8 | 21.5 |
| IOT_SHACAM_BEDROOM air_quality_sensor_ID20 | 14 | 89 | 45.5 | 42 |
| IOT_SHACAM_BEDROOM humidity_sensor_ID18 | 32.1 | 69.7 | 54.7 | 63.5 |
| IOT_SHACAM_BEDROOM occupancy_sensor_ID17 | 0 | 1 | 0.727 | 1 |
| IOT_SHACAM_BEDROOM smoke_detector_ID21 | 0 | 1 | 0.636 | 0 |
| IOT_SHACAM_BEDROOM temperature_sensor_ID19 | 21.1 | 25.0 | 22.8 | 25.0 |
| IOT_SHACAM_KITCHEN air_quality_sensor_ID4 | 5 | 99 | 37.8 | 99 |
| IOT_SHACAM_KITCHEN carbon_monoxide_sensor_ID8 | 0.0700 | 1.76 | 0.781 | 0.210 |
| IOT_SHACAM_KITCHEN humidity_sensor_ID2 | 30.7 | 66.4 | 48.1 | 47.5 |
| IOT_SHACAM_KITCHEN occupancy_sensor_ID1 | 0 | 1 | 0.667 | 0 |



Alarms Database

Alert table

| Time | air_quality_sensor_ID42 | Time | carbon_monoxide_sensor_ID | Time | motion_sensor_ID43 | Time | smoke_detector | Time | temperature_sensor_ID41 |
|---|---|---|---|---|---|---|---|---|---|
| 2025-01-30 01:40:38 | 309 | 2025-01-30 01:40:45 | 0.870 | 2025-01-30 01:40:39 | 0 | 2025-01-30 01:40:29 | 1 | 2025-01-30 01:40:37 | -10.4 |
| 2025-01-30 01:41:04 | 374 | 2025-01-30 01:41:11 | 1.46 | 2025-01-30 01:41:05 | 1 | 2025-01-30 01:40:34 | 1 | 2025-01-30 01:41:03 | -11.4 |
| 2025-01-30 01:41:30 | 83 | 2025-01-30 01:41:37 | 1.06 | 2025-01-30 01:41:31 | 1 | 2025-01-30 01:40:44 | 1 | 2025-01-30 01:41:29 | -14.8 |
| 2025-01-30 01:41:56 | 205 | 2025-01-30 01:42:03 | 0.830 | 2025-01-30 01:41:57 | 1 | 2025-01-30 01:40:50 | 1 | 2025-01-30 01:41:55 | -10.1 |
| 2025-01-30 01:42:22 | 64 | 2025-01-30 01:42:55 | 0.950 | 2025-01-30 01:42:23 | 0 | 2025-01-30 01:40:55 | 1 | 2025-01-30 01:42:21 | -13.8 |
| 2025-01-30 01:42:48 | 263 | 2025-01-30 01:44:13 | 0.990 | 2025-01-30 01:42:49 | 0 | 2025-01-30 01:41:10 | 1 | 2025-01-30 01:42:47 | -10.7 |
| 2025-01-30 01:43:14 | 217 | 2025-01-30 01:44:39 | 1.76 | 2025-01-30 01:43:15 | 0 | 2025-01-30 01:41:16 | 1 | 2025-01-30 01:43:13 | -14.1 |
| 2025-01-30 01:43:40 | 128 |  |  | 2025-01-30 01:43:41 | 0 | 2025-01-30 01:41:26 | 1 | 2025-01-30 01:43:39 | -11.7 |
| 2025-01-30 01:44:06 | 159 |  |  | 2025-01-30 01:44:07 | 1 | 2025-01-30 01:42:13 | 1 | 2025-01-30 01:44:05 | -10.0 |
| 2025-01-30 01:44:32 | 116 |  |  | 2025-01-30 01:44:33 | 0 | 2025-01-30 01:42:34 | 1 | 2025-01-30 01:44:31 | -14.0 |

## Conclusion

In conclusion, the Smart Home Automated Control and Monitoring System represents an advanced IoT-based solution for home automation and management. This project offers a comprehensive platform for monitoring and controlling various aspects of the home environment, such as temperature, lighting, security, and more. By continuously monitoring critical parameters, the system ensures the comfort and safety of the residents. The ability to customize settings and thresholds for different devices enhances the system's flexibility, allowing it to cater to a wide range of home environments. Real-time monitoring and data processing also enable quick responses to any changes, ensuring the optimal functioning of the smart home. This project demonstrates the transformative potential of IoT in modernizing home management, making it more efficient, secure, and user-friendly, while paving the way for smarter and more responsive living experiences.