**Smart Home Sensor Monitoring & Control System** **Tiloschan Karki**

**Detailed Project Report**

## 1. Introduction

This report explains the full implementation of a Smart Home Sensor Monitoring and Control System built using Python socket programming. The project simulates IoT devices communicating with a central Smart Hub through TCP and UDP protocols, similar to real-world smart home environments.

## 2. Project Objectives

The goal of this project is to show how TCP and UDP work in a real IoT-style environment.

- TCP is used for reliable device registration and control commands.
- UDP is used for fast real-time sensor streaming.
- Each IoT device acts as a client.
- The Smart Hub runs as both a TCP and UDP server simultaneously.

## 3. TCP Component (Device Registration & Control)

The TCP server listens on port 5050 and accepts multiple device connections using threading. When a device connects, it registers itself using the format:

DEVICE <DeviceName> TYPE <SensorType>

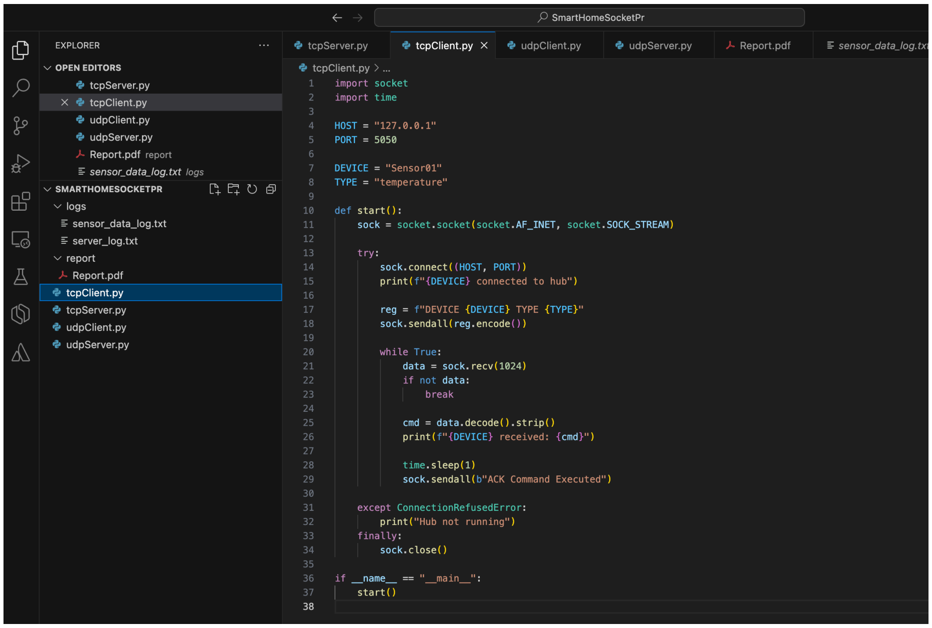
The server stores the device information and sends control commands:

- SET_INTERVAL 3

- ACTIVATE_ALARM

The client acknowledges each command with:

ACK Command Executed




### 4. UDP Component (Real-Time Sensor Data Streaming)

UDP is used for lightweight and fast transmission of sensor data. Each packet sent by the device has:

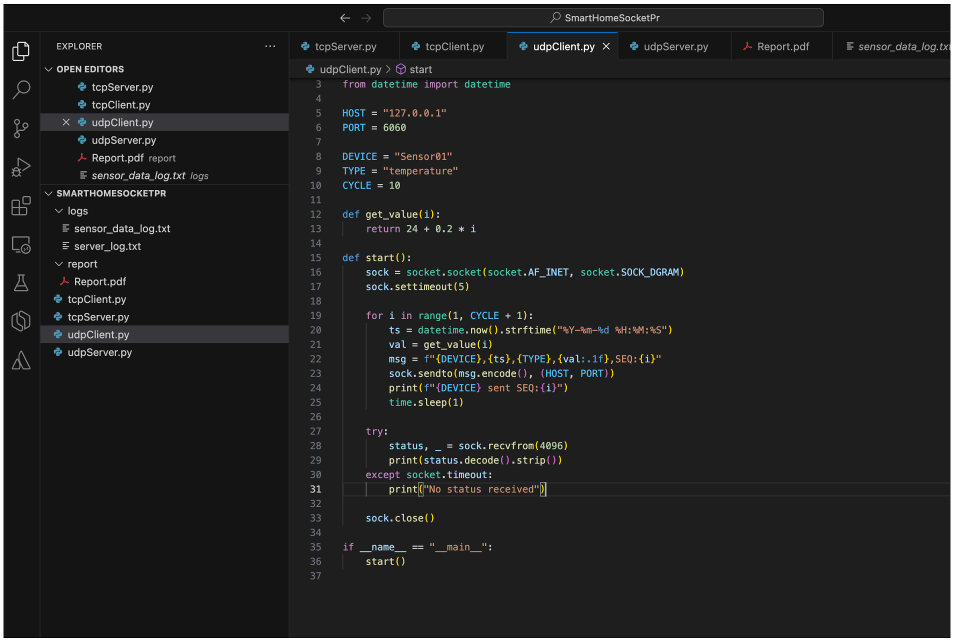
DeviceID, Timestamp, SensorType, Value, SEQ:Number

Example:

Sensor01,2025-10-22 18:20:15,temperature,24.8,SEQ:5

The server logs each packet and checks for missing packets. After receiving a full cycle of 10 packets, the server responds with:

STATUS RECEIVED 10/10 PACKETS




## 5. Logging System

The system stores logs in two separate files:

• server_log.txt – stores TCP connection, commands, and acknowledgments.

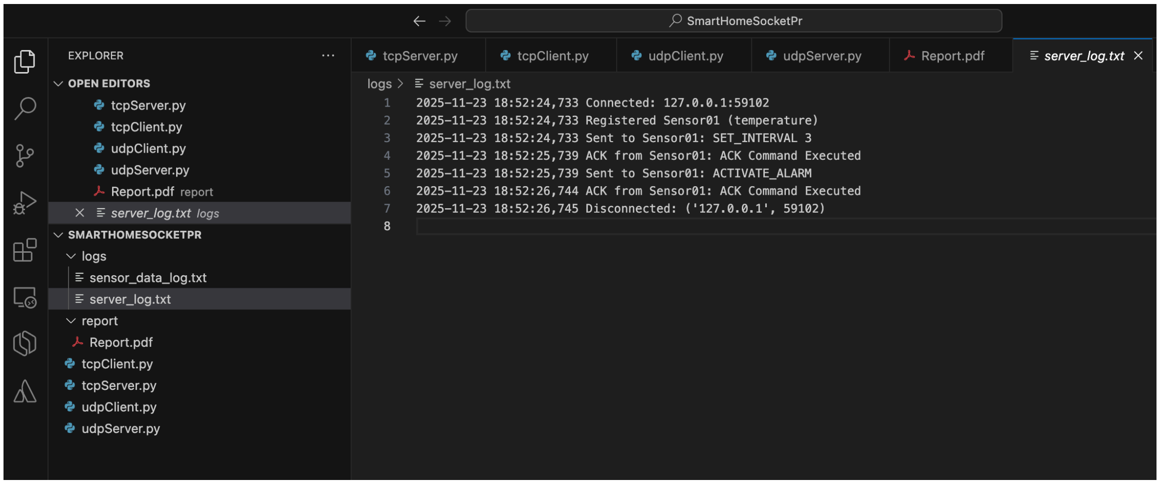
• sensor_data_log.txt – stores every UDP sensor packet received.

## 6. Sample Logs

TCP Log Example (server_log.txt):

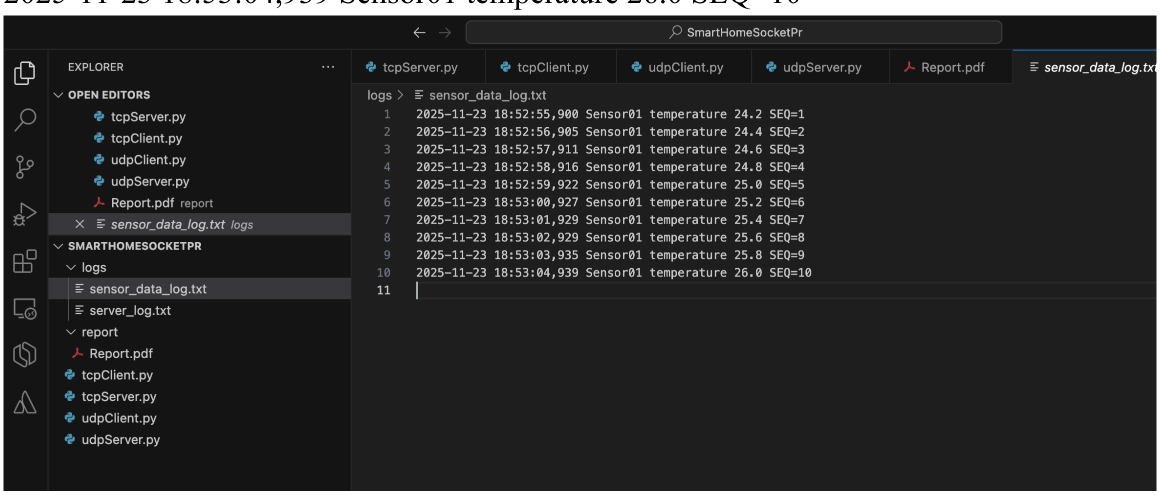
- 2025-11-23 18:52:24,733 Connected: 127.0.0.1:59102
- 2025-11-23 18:52:24,733 Registered Sensor01 (temperature)
- 2025-11-23 18:52:24,733 Sent to Sensor01: SET_INTERVAL 3
- 2025-11-23 18:52:25,739 ACK from Sensor01: ACK Command Executed
- 2025-11-23 18:52:25,739 Sent to Sensor01: ACTIVATE_ALARM
- 2025-11-23 18:52:26,744 ACK from Sensor01: ACK Command Executed

- 2025-11-23 18:52:26,745 Disconnected: ('127.0.0.1', 59102)




UDP Log Example (sensor_data_log.txt):

- 2025-11-23 18:52:55,900 Sensor01 temperature 24.2 SEQ=1
- 2025-11-23 18:52:56,905 Sensor01 temperature 24.4 SEQ=2
- 2025-11-23 18:52:57,911 Sensor01 temperature 24.6 SEQ=3
- 2025-11-23 18:52:58,916 Sensor01 temperature 24.8 SEQ=4
- 2025-11-23 18:52:59,922 Sensor01 temperature 25.0 SEQ=5
- 2025-11-23 18:53:00,927 Sensor01 temperature 25.2 SEQ=6
- 2025-11-23 18:53:01,929 Sensor01 temperature 25.4 SEQ=7
- 2025-11-23 18:53:02,929 Sensor01 temperature 25.6 SEQ=8
- 2025-11-23 18:53:03,935 Sensor01 temperature 25.8 SEQ=9
- 2025-11-23 18:53:04,939 Sensor01 temperature 26.0 SEQ=10



**7. How to Run the System**

1. Open a terminal in your project folder.
2. Start the TCP Server:
   python tcpServer.py
3. Start the TCP Client in another terminal:
   python tcpClient.py
4. For UDP, start the UDP Server:
   python udpServer.py
5. Then start the UDP Client:
   python udpClient.py

The logs will automatically appear in the log files.

**8. Conclusion**

This system successfully demonstrates how TCP and UDP can work together in a real IoT environment. TCP is used for reliable commands while UDP handles fast streaming sensor data. The project includes device registration, command execution, sequence tracking, missing packet detection, multi-threading, and logging — fulfilling all assignment requirements.