Smart Home Sensor Monitoring & Control System

By: Jonathan DeGrasse gtv9

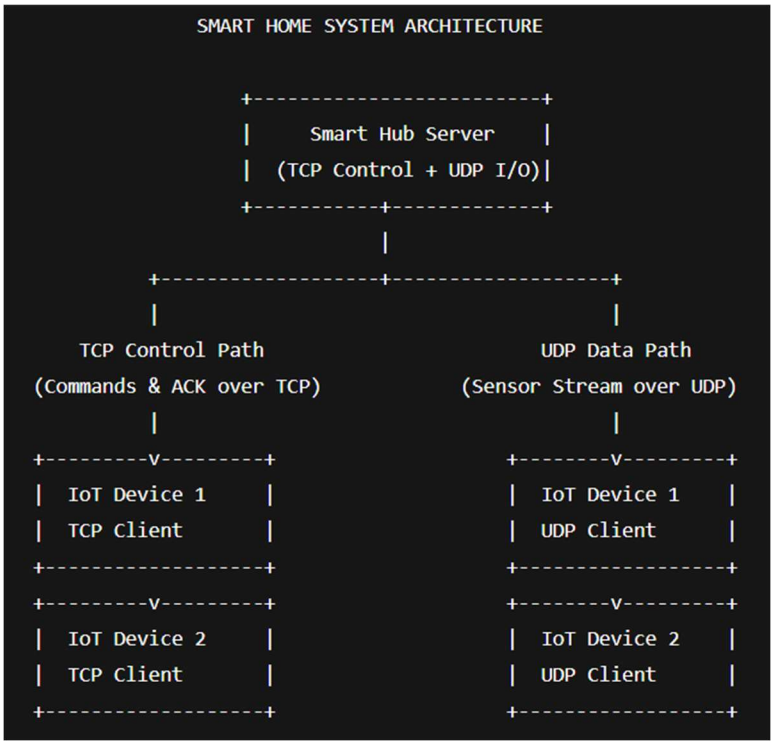
1. Introduction

This project simulates a smart home environment using Python socket programming features. It demonstrates how different devices in the home can talk to a control hub using TCP for dependable commands and UDP quick sensor updates.

2. System Architecture & Flow

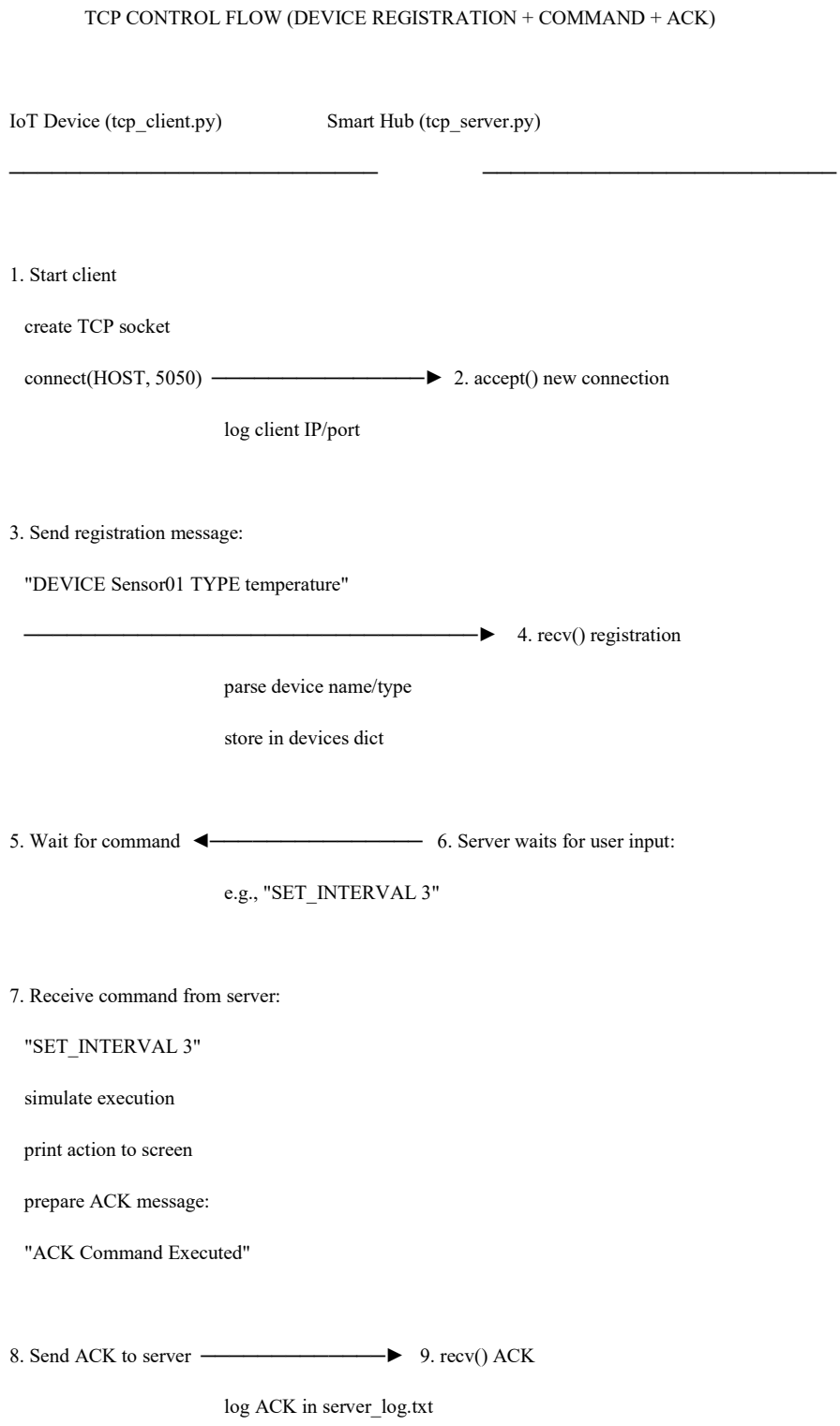
The system uses two communication paths:

TCP: For the control path. Devices register and receive commands from the Smart Hub.
UDP: Data streaming path. Devices will send sensor readings to the hub periodically.



3. TCP Communication

The TCP server accepts device connections, stores registration information, sends commands, and logs ACK responses. IoT devices register in the form: DEVICE <Name> TYPE <SensorType>.

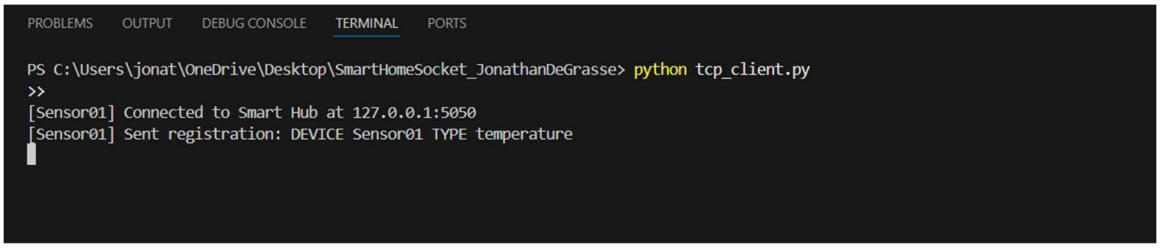


10. Loop for more commands

or close socket when server sends "quit"

TCP Server Start: Shows the start of the server and that it is running.

TCP Client Start: Shows client registration and device identification.



Command + ACK Server / Client

SET_INTERVAL 3 Server

SET_INTERVAL 3 Client

ACTIVATE_ALARM Server

```
[Server] TCP Smart Hub listening on 127.0.0.1:5050
[Server] Connected to device at ('127.0.0.1', 63403)
[Server] Received registration: DEVICE Sensor01 TYPE temperature
[Server] Registered DEVICE=Sensor01 TYPE=temperature
[Server] Enter command for Sensor01 (e.g., 'SET_INTERVAL 3' or 'ACTIVATE_ALARM', 'quit' to disconnect): SET_INTERVAL 3
[Server] Sent command to Sensor01: SET_INTERVAL 3
[Server] Received ACK from Sensor01: ACK Command Executed
[Server] Enter command for Sensor01 (e.g., 'SET_INTERVAL 3' or 'ACTIVATE_ALARM', 'quit' to disconnect): ACTIVATE_ALARM
[Server] Sent command to Sensor01: ACTIVATE_ALARM
[Server] Received ACK from Sensor01: ACK Command Executed
[Server] Enter command for Sensor01 (e.g., 'SET_INTERVAL 3' or 'ACTIVATE_ALARM', 'quit' to disconnect):
```

ACTIVATE_ALARM Client

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS C:\Users\jonat\OneDrive\Desktop\SmartHomeSocket_JonathanDeGrasse> python tcp_client.py
>>
[Sensor01] Connected to Smart Hub at 127.0.0.1:5050
[Sensor01] Sent registration: DEVICE Sensor01 TYPE temperature
[Sensor01] Received command: SET_INTERVAL 3
[Sensor01] Changing reporting interval to 3 seconds (simulated).
[Sensor01] Sent ACK.
[Sensor01] Received command: ACTIVATE_ALARM
[Sensor01] Alarm activated (simulated).
[Sensor01] Sent ACK.
```

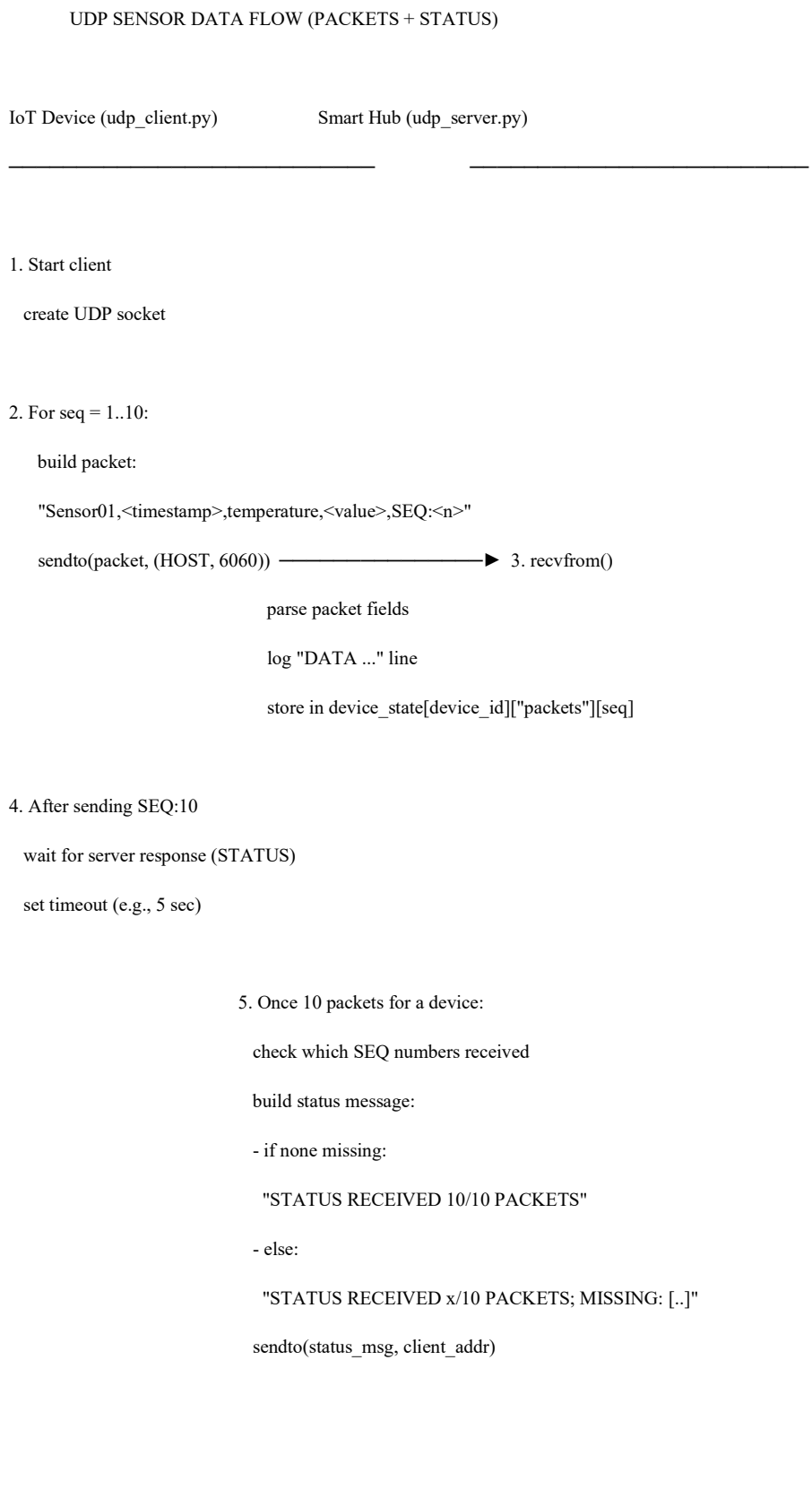
QUIT Server / Client

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

[Server] Registered DEVICE=Sensor01 TYPE=temperature
[Server] Enter command for Sensor01 (e.g., 'SET_INTERVAL 3' or 'ACTIVATE_ALARM', 'quit' to disconnect): SET_INTERVAL 3
[Server] Sent command to Sensor01: SET_INTERVAL 3
[Server] Received ACK from Sensor01: ACK Command Executed
[Server] Enter command for Sensor01 (e.g., 'SET_INTERVAL 3' or 'ACTIVATE_ALARM', 'quit' to disconnect): ACTIVATE_ALARM
[Server] Sent command to Sensor01: ACTIVATE_ALARM
[Server] Received ACK from Sensor01: ACK Command Executed
[Server] Enter command for Sensor01 (e.g., 'SET_INTERVAL 3' or 'ACTIVATE_ALARM', 'quit' to disconnect): quit
[Server] Closing connection with Sensor01
[Server] Connection to ('127.0.0.1', 63403) closed.
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

[Sensor01] Connected to Smart Hub at 127.0.0.1:5050
[Sensor01] Sent registration: DEVICE Sensor01 TYPE temperature
[Sensor01] Received command: SET_INTERVAL 3
[Sensor01] Changing reporting interval to 3 seconds (simulated).
[Sensor01] Sent ACK.
[Sensor01] Received command: ACTIVATE_ALARM
[Sensor01] Alarm activated (simulated).
[Sensor01] Sent ACK.
[Sensor01] Server closed connection.
[Sensor01] Closing connection.
PS C:\Users\jonat\OneDrive\Desktop\SmartHomeSocket_JonathanDeGrasse>
```

4. UDP Sensor Data Streaming

The UDP client sends timestamped sensor readings with the sequence numbers displayed in the terminal. The UDP server reconstructs the packet order, detects missing packets, and sends STATUS updates back to the client in real time.



◄——

6. Client recvfrom() STATUS

   - if received: print STATUS to screen

   - if timeout: print "No status received (timeout)."

7. Sleep, then start next cycle and repeat.

UDP Server

UDP Client