

Smart Home Sensor Monitoring & Control System

By: Jonathan DeGrasse gtv9

1. Introduction

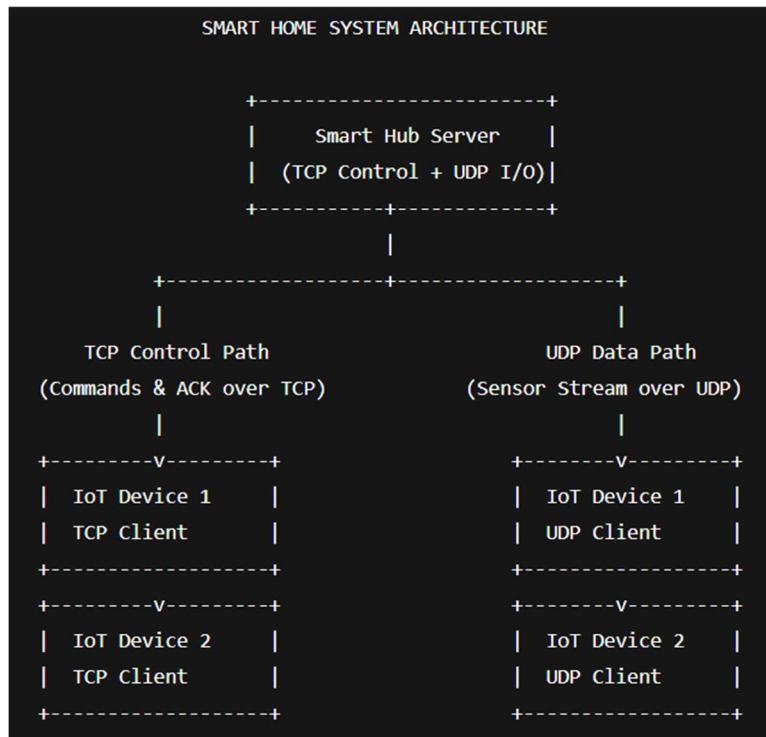
This project simulates a smart home environment using Python socket programming features. It demonstrates how different devices in the home can talk to a control hub using TCP for dependable commands and UDP quick sensor updates.

2. System Architecture & Flow

The system uses two communication paths:

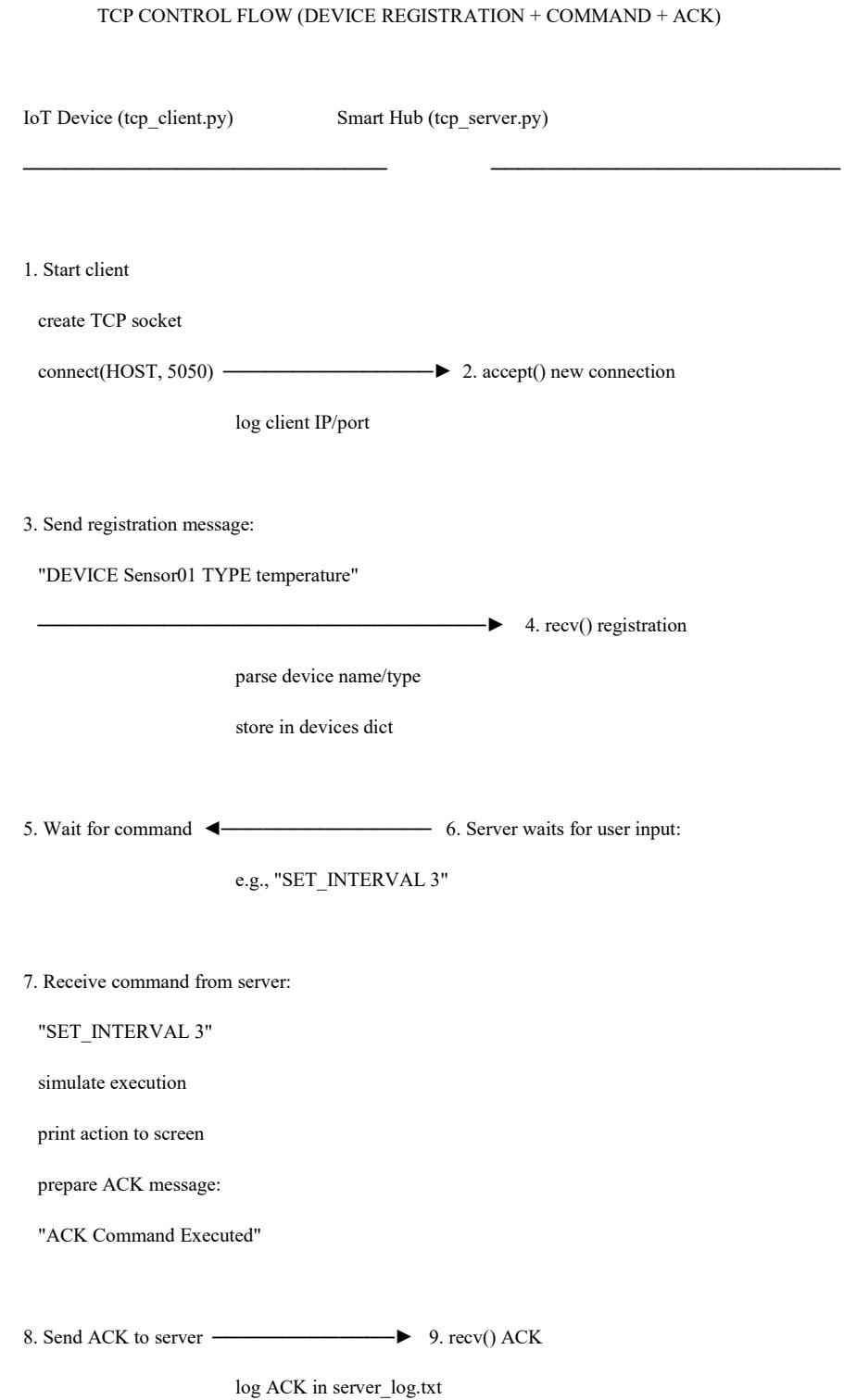
TCP: For the control path. Devices register and receive commands from the Smart Hub.

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.



A screenshot of a terminal window titled "python3.11". The window has tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (which is selected), and PORTS. The terminal content shows the command "python tcp_server.py" being run, followed by the message "[Server] TCP Smart Hub listening on 127.0.0.1:5050".

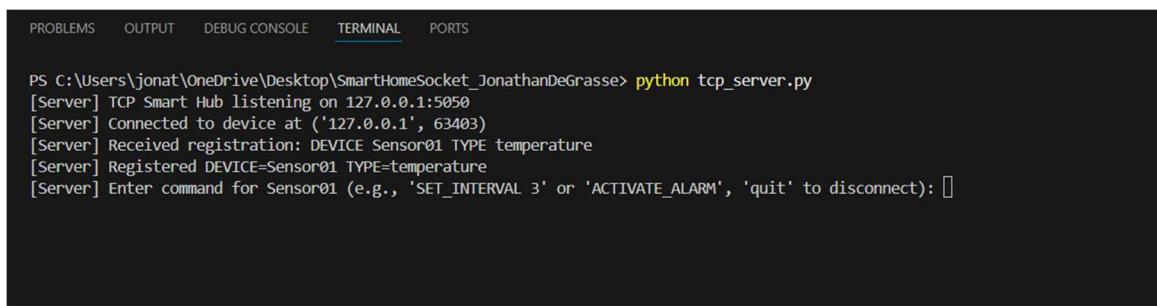
TCP Client Start: Shows client registration and device identification.



A screenshot of a terminal window titled "python3.11". The window has tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (selected), and PORTS. The terminal content shows the command "python tcp_client.py" being run, followed by the output: "[Sensor01] Connected to Smart Hub at 127.0.0.1:5050" and "[Sensor01] Sent registration: DEVICE Sensor01 TYPE temperature".

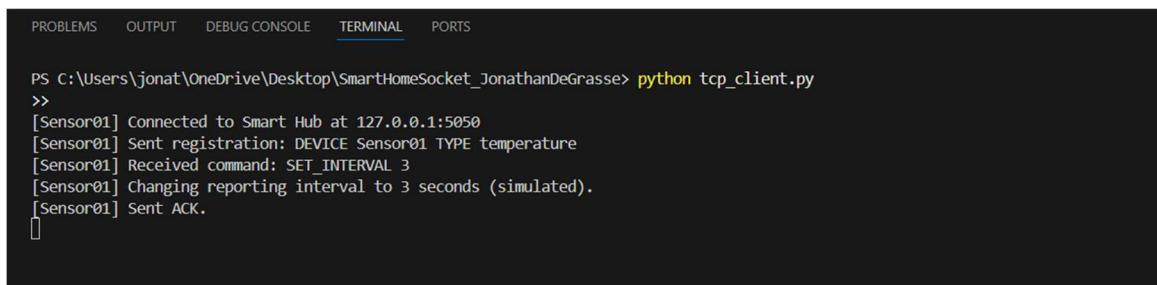
Command + ACK Server / Client

SET_INTERVAL 3 Server



A screenshot of a terminal window titled "python3.11". The window has tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (selected), and PORTS. The terminal content shows the command "python tcp_server.py" being run, followed by messages indicating device registration and a command prompt: "[Server] Received registration: DEVICE Sensor01 TYPE temperature", "[Server] Registered DEVICE=Sensor01 TYPE=temperature", and "[Server] Enter command for Sensor01 (e.g., 'SET_INTERVAL 3' or 'ACTIVATE_ALARM', 'quit' to disconnect):".

SET_INTERVAL 3 Client



A screenshot of a terminal window titled "python3.11". The window has tabs for PROBLEMS, OUTPUT, DEBUG CONSOLE, TERMINAL (selected), and PORTS. The terminal content shows the command "python tcp_client.py" being run, followed by messages indicating connection, registration, command receipt, and an ACK response: "[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.", and an empty line.

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.

UDP SENSOR DATA FLOW (PACKETS + STATUS)

IoT Device (udp_client.py) Smart Hub (udp_server.py)

1. Start client

create UDP socket

2. For seq = 1..10:

build packet:

"Sensor01,<timestamp>,temperature,<value>,SEQ:<n>"

sendto(packet, (HOST, 6060)) —————► 3. recvfrom()

parse packet fields

log "DATA ..." line

store in device_state[device_id]["packets"][seq]

4. After sending SEQ:10

wait for server response (STATUS)

set timeout (e.g., 5 sec)

5. Once 10 packets for a device:

check which SEQ numbers received

build status message:

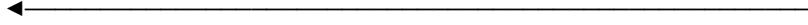
- if none missing:

"STATUS RECEIVED 10/10 PACKETS"

- else:

"STATUS RECEIVED x/10 PACKETS; MISSING: [..]"

sendto(status_msg, client_addr)



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

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS  python3.11 + ⌂ ⌂ ... | ⌂ X  
PS C:\Users\jonat\OneDrive\Desktop\SmartHomeSocket_JonathanDeGrasse> python udp_server.py  
>>  
[UDP Server] Smart Hub Data Collector listening on 127.0.0.1:6060
```

UDP Client

```
PS C:\Users\jonat\OneDrive\Desktop\SmartHomeSocket_JonathanDeGrasse> python udp_client.py
>>
[Sensor01] UDP client sending to Smart Hub at 127.0.0.1:6060
[Sensor01] Sending packet SEQ:1 - temperature=27.2
[Sensor01] Sending packet SEQ:2 - temperature=26.9
[Sensor01] Sending packet SEQ:3 - temperature=28.7
[Sensor01] Sending packet SEQ:4 - temperature=20.1
[Sensor01] Sending packet SEQ:5 - temperature=27.3
[Sensor01] Sending packet SEQ:6 - temperature=26.5
[Sensor01] Sending packet SEQ:7 - temperature=25.0
[Sensor01] Sending packet SEQ:8 - temperature=29.1
[Sensor01] Sending packet SEQ:9 - temperature=23.5
[Sensor01] Sending packet SEQ:10 - temperature=23.0
[Sensor01] STATUS RECEIVED 10/10 PACKETS
[Sensor01] Cycle complete. Starting a new cycle...

[Sensor01] Sending packet SEQ:1 - temperature=28.5
[Sensor01] Sending packet SEQ:2 - temperature=23.9
[Sensor01] Sending packet SEQ:3 - temperature=25.2
[Sensor01] Sending packet SEQ:4 - temperature=28.4
[Sensor01] Sending packet SEQ:5 - temperature=21.8
[Sensor01] Sending packet SEQ:6 - temperature=25.7
[Sensor01] Sending packet SEQ:7 - temperature=20.5
[Sensor01] Sending packet SEQ:8 - temperature=21.2
[Sensor01] Sending packet SEQ:9 - temperature=29.8
[Sensor01] Sending packet SEQ:10 - temperature=26.9
[Sensor01] STATUS RECEIVED 10/10 PACKETS
[Sensor01] Cycle complete. Starting a new cycle...
```