AISC2010 – Programming and Deployment of IOT Devices

In-class Case Study 1

Instructor: Qasim Ali

Implementing and Demonstrating the IoT System

Group Name: Group B

Student Name: Govind Ram Gupta Belde

Objective

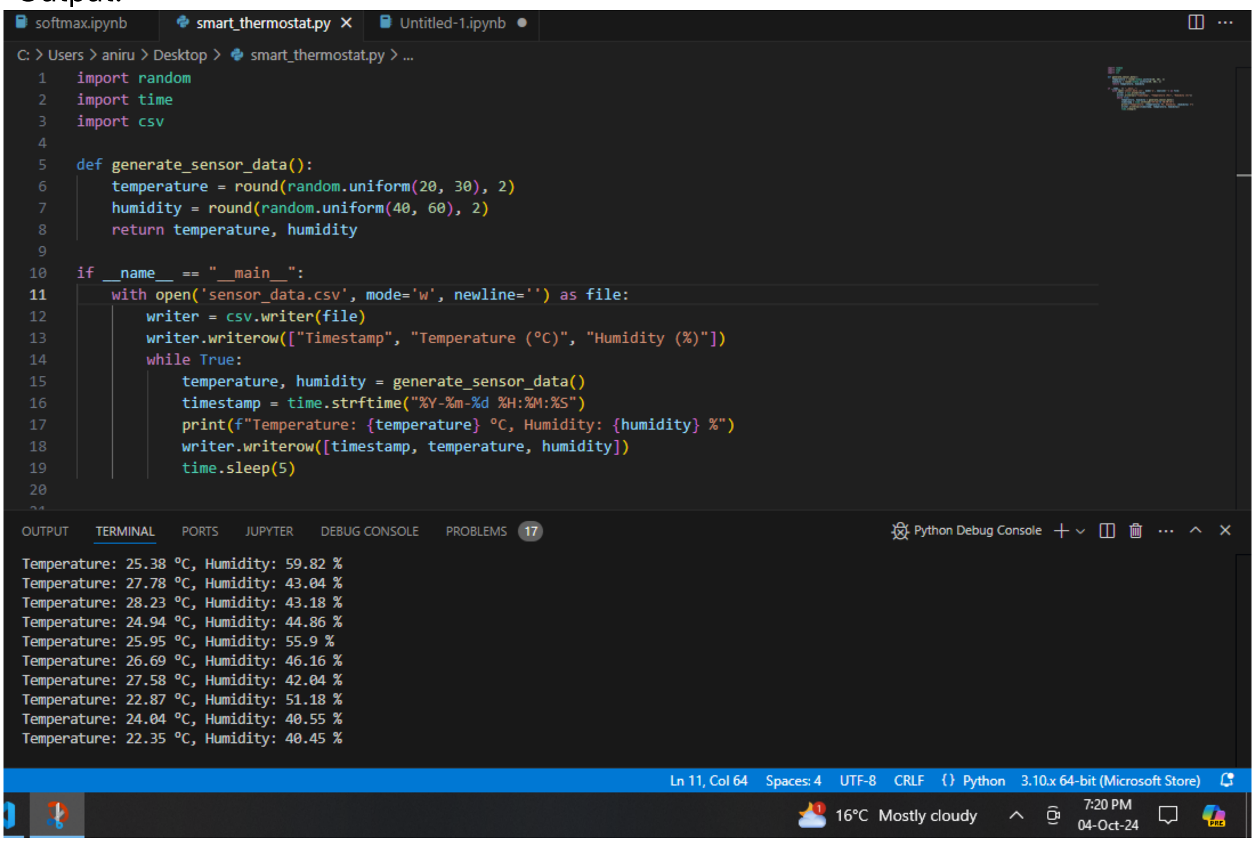
The objective of this assignment is to practically implement the designed IoT system using real or simulated IoT devices, and demonstrate data collection, processing, and visualization.

1. Implementation of IoT Devices

Device 1: Smart Thermostat

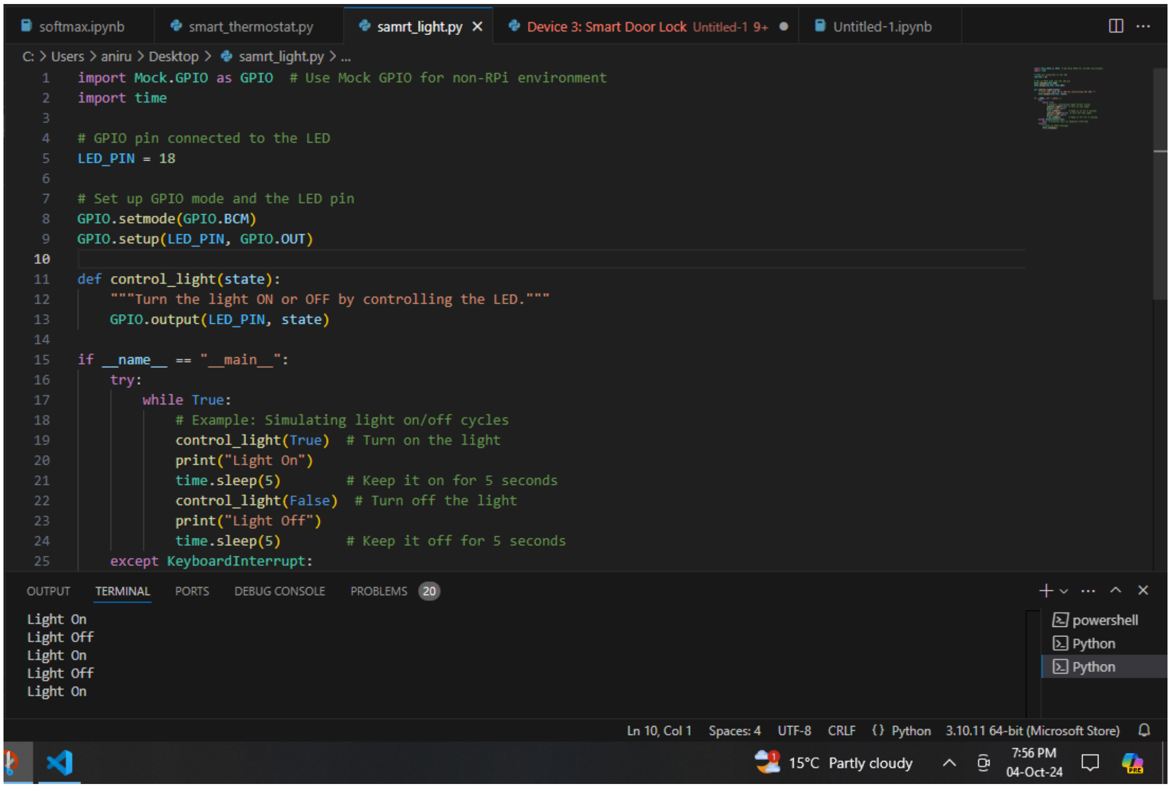
- Implementation: Create a Python script (`smart_thermostat.py`) that generates random temperature and humidity values.

Output:



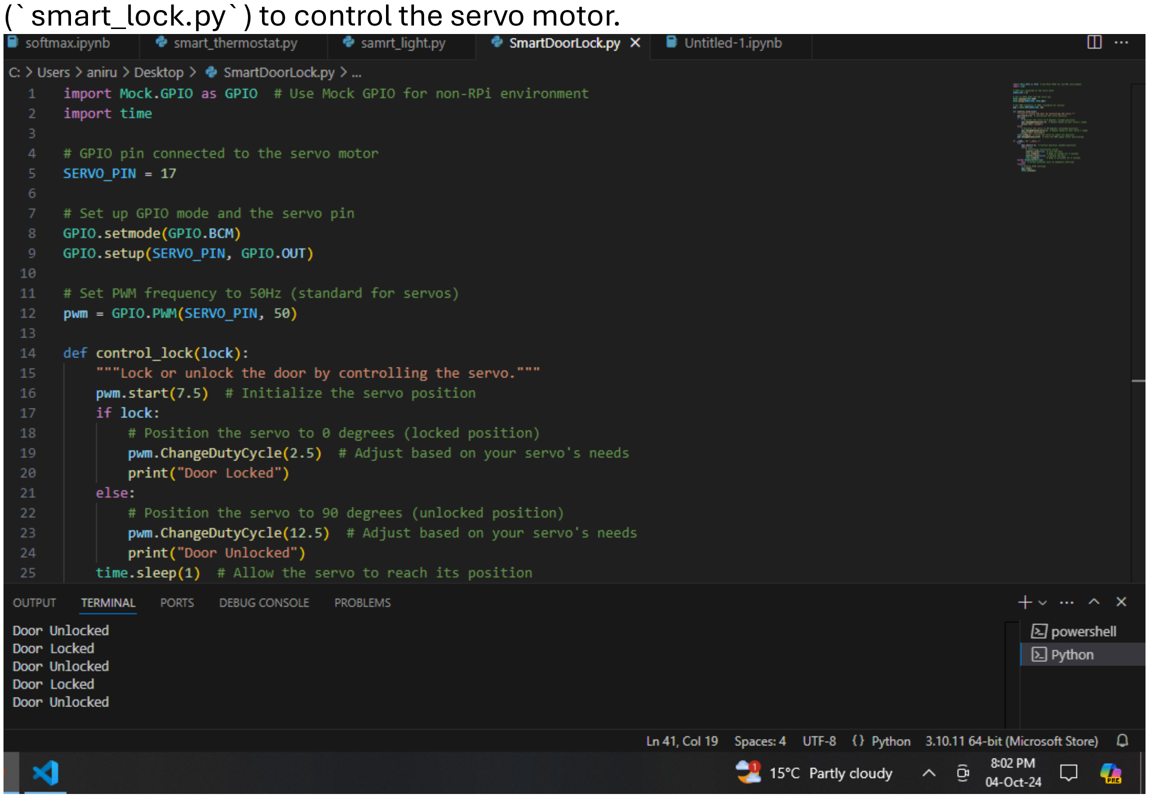
Device 2: Smart Light Bulb

- Implementation: Use a Raspberry Pi with a connected LED. Write a Python script (`smart_light.py`) to control the LED.



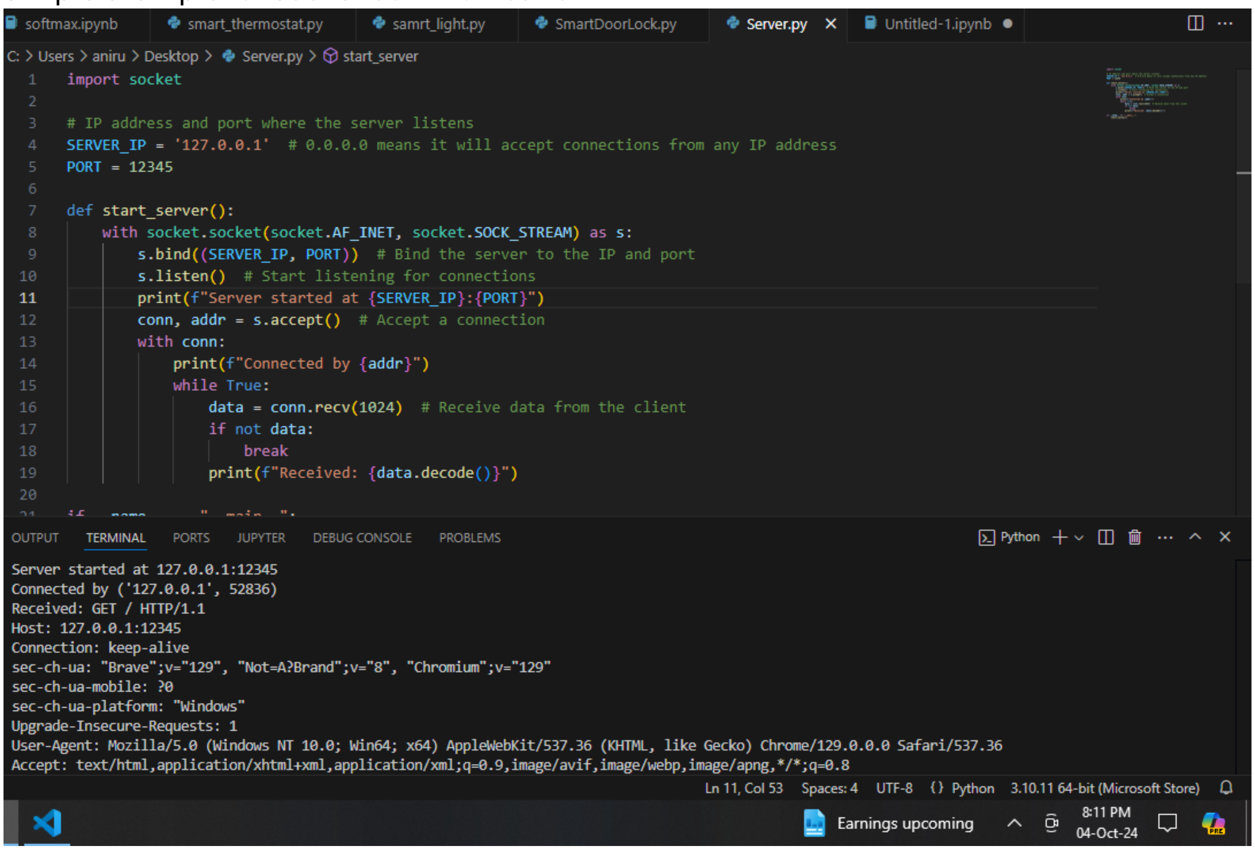
Device 3: Smart Door Lock

- Implementation: Use a Raspberry Pi with a servo motor. Write a Python script (`smart_lock.py`) to control the servo motor.



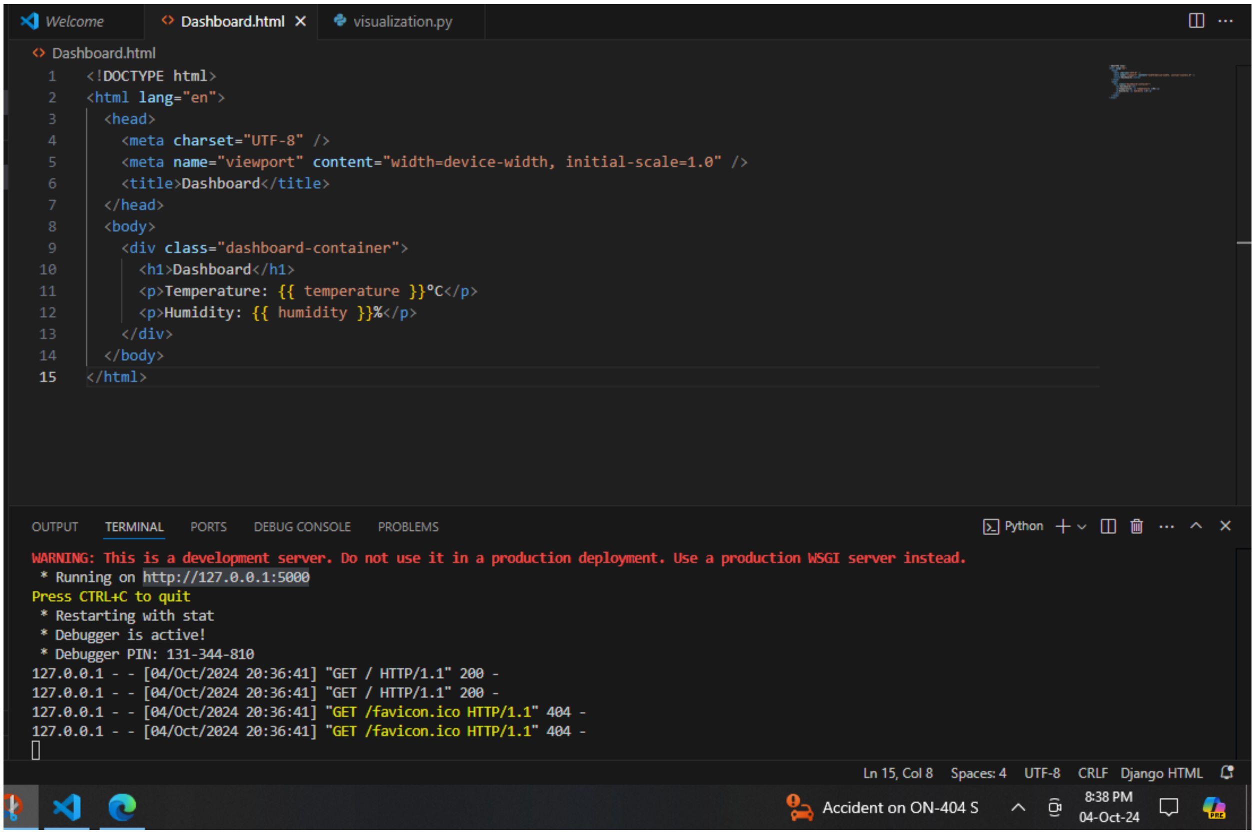
2. Network Connectivity

- Setup: Use Python libraries like `socket` to implement network connectivity. Here's a simple example for socket communication:



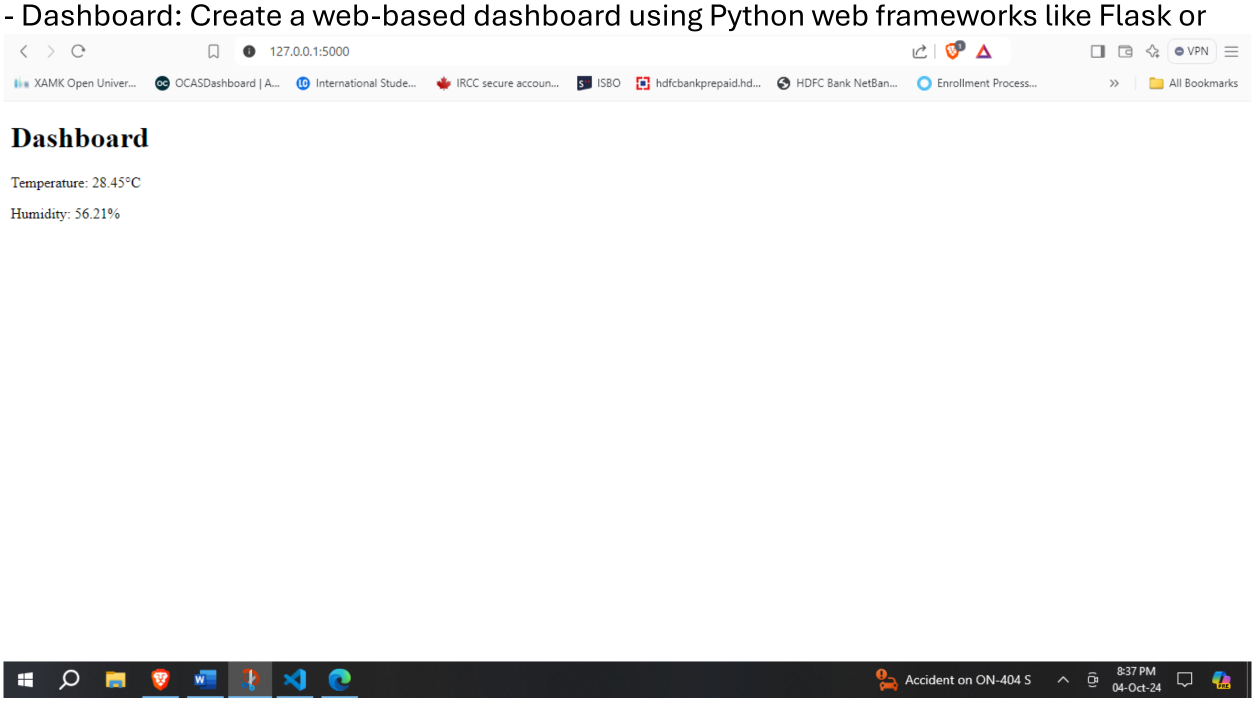
3. Data Collection and Processing

- Data Collection: Modify the device scripts to send data over MQTT using Python libraries like `paho-mqtt`.
- Data Processing: Use Python libraries like `pandas` or `numpy` to process the collected data locally or on a cloud platform.



4. Data Visualization

- Dashboard: Create a web-based dashboard using Python web frameworks like Flask or



5. Documentation and Demonstration

1. Creating a Web-Based Dashboard with Flask

We built a simple web-based dashboard using **Flask** to display dynamic temperature and humidity values. Flask was used to create a web server, and an HTML file was used as a template to render the data dynamically.

- We created a project structure with a templates folder to store HTML files, and a Python file (app.py) to handle the web server and routing.
- The Flask app uses the random.uniform() function to generate random temperature and humidity values, and it uses render_template() to pass the values to the HTML file.
- In the templates folder, we created dashboard.html to display the data using **Jinja2** syntax ({{ temperature }}, {{ humidity }}).
- We ran the Flask server locally, which allowed us to view the dashboard at http://127.0.0.1:5000/.

2. Setting Up MQTT Communication with paho-mqtt

We extended the project by introducing **MQTT** for device communication using the **paho-mqtt** Python library. This setup allows the devices to send sensor data (temperature and humidity) to a broker.

- We created a client that publishes sensor data to an MQTT broker (e.g., HiveMQ).
- The device sends data to a specific topic (iot/smart_thermostat) at regular intervals.
- Another script subscribes to the MQTT topic and collects the data for further processing.

3. Data Processing with Pandas and NumPy

We used **Pandas** and **NumPy** to process the data collected via MQTT.

- The collected sensor data is saved to a CSV file using Pandas.
- We used Pandas to generate basic statistics for temperature and humidity and visualized the data with **matplotlib**.