Assignment 8

End-to-End IoT System Report

CECS 327

Group 47 - Sophia Thomas & Peter Khim

12/8/24

System Architecture & Approach

Our project integrates an end-to-end IoT system by combining a TCP client-server architecture, MongoDB database, and IoT sensor data via metadata provided by Dataniz. The main components of the system are as follows:

Server Layer

Purpose: Process user queries, retrieves and analyzes IoT data, and sends response back to the client.
Implementation:

- Binary search tree used for efficient data storage and retrieval
- Converts moisture readings and ensures results are presented in imperial units

Client Layer

Purpose: provide an interface for users to send queries and view responses.
Implementation:

- Accepts user queries and validates them against a predefined list
- Reject invalid queries
- Send valid queries to the server via TCP

Database Layer

Purpose: Stores metadata and real-time data for IoT devices

Data generated by Dataniz is based on configuration of virtual IoT devices and sensors. Metadata from Dataniz enhanced query handling by identifying devices dynamically and associating relevant sensor data.

Queries

1. What is the average moisture inside my kitchen fridge in the past three hours?
2. What is the average water consumption per cycle in my smart dishwasher?
3. Which device consumed more electricity among my three IoT devices?

IoT Sensors & Data Findings

Moisture Sensor:

- **Data Type**: Measures humidity inside refrigerators.
- **Unit**: RH% (Relative Humidity).
- **Precision**: ±0.1% RH.
- **Time Zone**: All timestamps normalized to PST for query responses.

Water Flow Sensor:

- **Data Type**: Tracks water consumption per cycle for dishwashers.

- **Unit**: Gallons per cycle.
- **Precision**: ±0.01 gallons.

Ammeter:

- **Data Type**: Monitors electricity consumption.
- **Unit**: kWh.
- **Precision**: ±0.05 kWh.

Dataniz Metadata

Utilized Dataniz metadata to map the data to the IoT device that it is connected to by name of that device. This allows a user to select the specific device they want data on and allows us to create queries that target the correct device amongst the two smart fridges and single dishwasher. Through utilizing Metadata, we created the function get_device_names in the databaseQuery.py file to identify a device by name.

Implementation

The implementation of the IoT system integrates a TCP client-server architecture, a MongoDB database, and IoT sensor data to process user queries efficiently. The client layer validates queries and communicates with the server, providing a user-friendly interface. The server layer handles query processing, retrieves data from the database, performs calculations, and converts units (e.g., Relative Humidity, gallons, and kilowatt-hours). The database layer stores IoT metadata and real-time sensor readings, leveraging hierarchical relationships to associate devices with their sensors dynamically. The system's modular design ensures scalability, efficient data handling using a binary search tree, and seamless interaction between all components. From our research, gallons per cycle was impossible to calculate due to the fact that we do not know how long a cycle is, therefore we substituted by measuring the water consumption sensors by gallons per minute.

Challenges

Initially attempted to query the database and store the learned information in separate lists rather than use a binary search tree to store the data from the MongoDB database. This approach led to a more complex codebase. Utilizing a binary tree allowed for simpler query functions.

Our Dataniz Feedback

Dataniz framework was valuable in the development of our IoT system. But it does require some improvements. The current metadata requires manual updates when device configuration changes. It does not account for changes of existing devices, so including logs for these configurations could provide insight into usage and performance and save a lot of time rebuilding the entire dataniz structure from scratch.