Christian Bae and Fabian Ochoa
CECS 327

TCP- endTOend System Report

1. Summary of System Architecture and Approach

The system architecture is built on a client-server model for querying and processing data from IoT devices. The components include:

- Client Application: A Python-based client where users can select predefined queries (e.g., average moisture, water consumption, or energy usage).
- Server: A Python server connected to a MongoDB database that processes queries and returns results.
- Database: MongoDB stores IoT data collected from devices such as refrigerators, dishwashers, and other appliances.
- IoT Devices: Sensors provide data such as moisture levels, water consumption, and energy usage. The data is periodically uploaded to the database for real-time analytics.

The server processes queries by executing MongoDB aggregation pipelines, ensuring that data is filtered, grouped, and summarized effectively. Results are returned to the client application for display.

2. Research Findings on IoT Sensors and Data

The IoT system incorporates sensors that measure:

- Moisture Levels: Collected using a moisture meter, typically reported as Relative Humidity (RH%).
- Water Consumption: Measured in liters (L) per dishwasher cycle using a flow meter.
- Electricity Consumption: Measured in kilowatt-hours (kWh) using ammeters for energy usage by devices.

Findings from research on IoT data:
Accurate sensor calibration is critical for reliable data.
Timestamp-based filtering ensures insights are generated for recent activity (e.g., past 3 hours).
MongoDB is an efficient database for IoT applications due to its flexible schema and query capabilities.

3. Use of Dataniz Metadata

The integration of Dataniz metadata was evaluated for enhancing data organization and analysis:

- Use Case: Metadata could improve tagging, searching, and categorization of IoT data.
- Findings: While Dataniz metadata can be useful for systems requiring extensive tagging or categorization, the current implementation primarily relies on the inherent MongoDB schema and manual data structure.

- Reason for Limited Applicability: The existing IoT system did not require additional metadata due to the straightforward structure of queries and device-specific attributes (e.g., board_name, sensor fields).

4. Algorithms, Calculations, and Unit Conversions
    - Moisture Levels: Aggregated over the past 3 hours using MongoDB's $avg operator on the Moisture Meter - Moisture meter field.
    - Water Consumption: Average per dishwasher cycle calculated using $avg on the water consumption sensor field.
    - Electricity Consumption: Total usage grouped by board_name and summed using $sum on the Ammeter field.
    - Unit Conversions: Applied MongoDB's $toDouble for fields stored as strings to ensure numeric aggregation.
    -

5. Challenges Faced and Solutions
Challenge: Data Structure Mismatch
- Some fields were stored as strings instead of numeric types, causing aggregation queries to fail.
- Solution: Used MongoDB's $toDouble operator to convert fields during query execution.

Challenge: Time-based Filtering
- Filtering data based on timestamps required converting timestamps into MongoDB-compatible formats.
- Solution: Used Python’s datetime library to generate accurate time ranges for queries.
- Challenge: Understanding Device-specific Units

Lack of clear documentation on sensor units led to confusion during implementation.
Solution: Assumed standard units (RH% for moisture, liters for water, kWh for electricity) and included placeholder outputs for further validation.

6. Feedback for Dataniz.com

Strengths:
Dataniz provides a structured approach to managing and categorizing metadata, which can be helpful for complex data systems.
Integration capabilities could enhance systems requiring advanced tagging or querying.

Areas for Improvement:
Documentation on integrating Dataniz metadata into IoT systems could be more detailed.
Examples specific to IoT and time-series data management would be valuable for developers.
In the current implementation, Dataniz metadata was not directly applied due to the simplicity of the system's structure. However, it may be revisited as the system scales.

This report highlights the system's architecture, data processing methods, and lessons learned during implementation. Further iterations could incorporate advanced metadata and additional features for real-time analytics.