Project Report

Project 4 ECE 528

A20561414 Saman Chouhan

## Acknowledgement

I acknowledge all works including figures, codes, and writings belong tome and/or persons who are referenced. I understand if any similarity int he code, comments, customized program behavior, report writings, and/or figures are found, both the helper (original work) and the requestor (duplicated/modified work) will be called for academic disciplinary action

Samman Chouhan

04/6/2025

# 1. Introduction

This project focuses on managing a set of IoT plug devices through a Spring Boot application. The application communicates with an MQTT broker to retrieve and control the state and power readings of plugs. The project also includes a comprehensive suite of unit tests to ensure that each feature works correctly and to maintain high code quality with at least 90% unit test coverage for the modified and added classes.

# 2. Project Overview

The project is built around the following core components:

- **Backend Application:** Implemented using Spring Boot.
- **MQTT Communication:** An MQTT controller (provided as part of the grading package) is used to interact with the broker.
- **REST API Endpoints:** Exposed via a resource class to enable HTTP-based interaction.
- **Unit Testing:** Comprehensive test cases were added to verify both the business logic and the API endpoints.

## 2.1 Classes Added/Modified

- **App.java / App (1).java:**
  These files set up the Spring Boot application. They configure the MQTT controller as a Spring Bean and create an instance of the `PlugsModel` class, which encapsulates the business logic.
- **Plug.java:**
  This class models the IoT plug. It encapsulates key attributes such as the plug's name, state, and power. It also provides getter methods to retrieve these values.
- **PlugsModel.java:**
  This is the primary class that contains the logic for:
  - Fetching individual plug details (`getPlug` method), including handling cases where state information might be missing or improperly formatted.
  - Aggregating all plug states (`getAllPlugs` method).
  - Triggering actions on plugs (`performAction` method).

  The class interacts with the MQTT controller to retrieve plug state and power values. Notably, the implementation handles data conversion (e.g., rounding float values for power) and fallback mechanisms (returning "unknown" for null states).

- **PlugsResource.java:**
  This REST controller exposes HTTP endpoints for:
  - Retrieving the state of an individual plug (`/api/plugs/{name}`).
  - Listing all plugs (`/api/plugs`).
  - Controlling a plug by sending an action (`/api/plugs/{name}?action=...`).

- **Unit Test Files:**
  The project includes multiple test classes under `src/test/java`:
  - **PlugsModelTests.java:**
    Contains detailed unit tests for the `PlugsModel` class. A stub MQTT controller is defined within the tests to simulate interactions with the MQTT layer. The tests cover:
    - Valid and invalid data scenarios.
    - Rounding behavior (e.g., converting "99.9" to 99).
    - Handling of null or missing plug information.
    - Ensuring that the action-publishing mechanism works as expected.
  - **PlugsResourceTests.java:**
    Contains unit tests for the `PlugsResource` REST endpoints. These tests verify that the resource correctly maps HTTP requests to the underlying business logic.
  - **PlugTests.java:**
    Verifies that the `Plug` class getters function correctly.

# 3. Unit Test Design

## 3.1 Test Case Design Strategy

The unit tests were designed to ensure that every aspect of the business logic and REST interface is correctly implemented. The approach included:

- **Stub Implementations:**
  A stub version of the `MqttController` was created (inside `PlugsModelTests.java`) to isolate the business logic from external dependencies. This allows for controlled testing of scenarios such as:
  - Valid power and state values.
  - Incorrect or missing power values (e.g., testing fallback behavior).
  - Verifying that actions are correctly tracked.
- **Edge Cases:**
  The tests check for:
  - Rounding of power values.
  - Handling of invalid power strings that cannot be parsed as a number.
  - Case sensitivity in plug names, ensuring that only the correct plug details are returned.
- **Comprehensive Coverage:**
  The tests ensure that each method in the `PlugsModel` and `PlugsResource` classes is covered by at least one test case. With at least 10 new unit test cases added in total, the tests have been structured to cover both typical and atypical scenarios.

## 3.2 Example Test Cases

Some of the key test cases include:

- **Valid Data Retrieval:**
  Confirming that `getPlug` returns the correct name, state, and power when valid data is provided.
- **Fallback for Missing Data:**
  Testing that a plug with no provided state defaults to "unknown" and that power defaults to 0 when invalid data is encountered.
- **Action Tracking:**
  Ensuring that when an action is performed (e.g., "toggle"), the correct plug name and action are recorded.
- **REST Endpoints:**
  Verifying that the GET and control endpoints in `PlugsResource` return or process data correctly.

Coverage Test Report :

iot_ece448 > ece448.iot_hub  Source Files  Sessions

## ece448.iot_hub

| Element | Missed Instructions | Cov. | Missed Branches | Cov. | Missed | Cxty | Missed | Lines | Missed | Methods |
|---|---|---|---|---|---|---|---|---|---|---|
| Main | | 0% | | 0% | 5 | 5 | 19 | 19 | 4 | 4 |
| App | | 0% | | n/a | 3 | 3 | 8 | 8 | 3 | 3 |
| HubConfig | | 0% | | n/a | 5 | 5 | 10 | 10 | 5 | 5 |
| PlugsModel | | 100% | | 100% | 0 | 7 | 0 | 17 | 0 | 4 |
| PlugsResource | | 100% | | n/a | 0 | 4 | 0 | 7 | 0 | 4 |
| Plug | | 100% | | n/a | 0 | 4 | 0 | 8 | 0 | 4 |
| Total | 164 of 279 | 41% | 2 of 8 | 75% | 13 | 28 | 37 | 69 | 12 | 24 |

Created with JaCoCo 0.8.11.202310140853

Test Cases Report:

## Package ece448.iot_hub

all > ece448.iot_hub

| 13 | 0 | 0 | 0.036s |
|---|---|---|---|
| tests | failures | ignored | duration |

100% successful

### Classes

| Class | Tests | Failures | Ignored | Duration |
|---|---|---|---|---|
| PlugTests | 1 | 0 | 0 | 0s |
| PlugsModelTests | 8 | 0 | 0 | 0.035s |
| PlugsResourceTests | 4 | 0 | 0 | 0.001s |

Generated by Gradle 6.9.3 at Apr 6, 2025, 1:23:22 PM

Interal Cases GradeP4:

```
OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS    COMMENTS

****************************
2025-04-06 13:28:14,083 INFO MqttCmd 1743964083223/grade_p4/iot_ece448/action/a/off:
2025-04-06 13:28:14,083 INFO MqttCmd 1743964083223/grade_p4/iot_ece448/action/a/off:
2025-04-06 13:28:14,083 WARN Unknown plug: a
2025-04-06 13:28:14,083 INFO MqttCmd 1743964083223/grade_p4/iot_ece448/action/b/toggle:
2025-04-06 13:28:14,084 WARN Unknown plug: b
2025-04-06 13:28:14,084 INFO MqttCmd 1743964083223/grade_p4/iot_ece448/action/b/toggle:
2025-04-06 13:28:14,084 INFO JHTTP: /127.0.0.1:49705 GET /c?action=off HTTP/1.1
2025-04-06 13:28:14,084 INFO HTTPCmd /c: {action=off}
2025-04-06 13:28:14,086 INFO JHTTP: /127.0.0.1:49706 GET /d?action=toggle HTTP/1.1
2025-04-06 13:28:14,086 INFO HTTPCmd /d: {action=toggle}
2025-04-06 13:28:14,088 INFO MqttCmd 1743964083223/grade_p4/iot_ece448/action/e/toggle:
2025-04-06 13:28:14,088 WARN Unknown plug: e
2025-04-06 13:28:14,088 INFO MqttCmd 1743964083223/grade_p4/iot_ece448/action/e/toggle:
2025-04-06 13:28:14,089 INFO MqttCmd 1743964083223/grade_p4/iot_ece448/action/f/off:
2025-04-06 13:28:14,089 WARN Unknown plug: f
2025-04-06 13:28:14,089 INFO MqttCmd 1743964083223/grade_p4/iot_ece448/action/f/off:
2025-04-06 13:28:14,089 INFO JHTTP: /127.0.0.1:49707 GET /g?action=off HTTP/1.1
2025-04-06 13:28:14,089 INFO HTTPCmd /g: {action=off}
2025-04-06 13:28:15,119 INFO JHTTP: /127.0.0.1:49708 GET /a HTTP/1.1
2025-04-06 13:28:15,119 INFO HTTPCmd /a: {}
2025-04-06 13:28:15,121 INFO JHTTP: /127.0.0.1:49709 GET /b HTTP/1.1
2025-04-06 13:28:15,121 INFO HTTPCmd /b: {}
2025-04-06 13:28:15,122 INFO JHTTP: /127.0.0.1:49710 GET /c HTTP/1.1
2025-04-06 13:28:15,122 INFO HTTPCmd /c: {}
2025-04-06 13:28:15,123 INFO JHTTP: /127.0.0.1:49711 GET /d HTTP/1.1
2025-04-06 13:28:15,123 INFO HTTPCmd /d: {}
2025-04-06 13:28:15,125 INFO JHTTP: /127.0.0.1:49712 GET /e HTTP/1.1
2025-04-06 13:28:15,125 INFO HTTPCmd /e: {}
2025-04-06 13:28:15,126 INFO JHTTP: /127.0.0.1:49713 GET /f HTTP/1.1
2025-04-06 13:28:15,126 INFO HTTPCmd /f: {}
2025-04-06 13:28:15,127 INFO JHTTP: /127.0.0.1:49714 GET /g HTTP/1.1
2025-04-06 13:28:15,127 INFO HTTPCmd /g: {}
******** testCase09 of GradeP4: pass
2025-04-06 13:28:15,127 INFO testCase09 of GradeP4: success
****************************

Local Grading Test Result: 10/10 cases passed
*******************************************************************************
* Make sure you "git add ." and "git commit -am "your comments" and "git push" *
* to fully upload all of your codes to the Endeavour Git Repo.                 *
* Otherwise, your grading result may differ from your local result vs. server  *
*******************************************************************************
2025-04-06 13:28:15,127 INFO Closing org.springframework.boot.context.embedded.AnnotationConfigEm
[Sun Apr 06 13:28:04 CDT 2025]; root of context hierarchy
2025-04-06 13:28:15,129 INFO Unregistering JMX-exposed beans on shutdown
2025-04-06 13:28:15,130 INFO MqttCtl testee/iot_hub: disconnected
2025-04-06 13:28:15,155 INFO JHTTP: disconnnected Socket closed
2025-04-06 13:28:15,155 INFO JHTTP: disconnnected Socket closed
2025-04-06 13:28:15,156 INFO MqttCtl grader/iot_hub: disconnected
<==========--> 88% EXECUTING [1m 19s]
> :grade_p4
```