PowerCloud Instruction Manual

version 2.0

Illusion Solutions

Stuart Andrews
12153983
Marc Antel
12026973
Mothusi Masibi
12004589
Brandon Wardley
29005150

October 11, 2016

Contents

1 Overview

dwefwef

2 Introduction

PowerCloud is an IoT project intended to supply electrical usage data to large companies and households. A PowerCloud device accurately measures the electrical consumption of rooms, offices and large machinery in order to provide analytics, predictions and reduce costs associated with electricity usage.

2.1 Purpose

This document provides unit test plans, results and a final report into one coherent document which can be used to analyse the current system status, this includes but is not limited to the test procedure, functional test cases, test results, conclusions and recommendations.

2.2 Scope

The scope of this document is structured as follows. The features that are considered for testing are listed in section 3, tests that have been identified from the requirements are discussed in detail in section 4.

Furthermore, this document outlines the test environment and the risks involved in the testing approaches that will be followed. Assumptions and dependencies of this test plan will also be mentioned.

2.3 Test Environment

- **Programming Languages:**
  - *JAVA v1.8*
  - *C++*
  - *Javascript*
  - *HTML*
  - *LESS*
- **Testing Frameworks:**
  - *JUnit*
  - *Karma & Jasmine*
- **Coding Environment:**
  - *Firmware: Atom + ParticleDev Plugin*
  - *Application Server: IntelliJ IDEA*
  - *Web Application: WebStorm*
- **Operating System:**

- *KALI Linux x64 2016*
- *Windows 10 x64*

- **Internet Browsers:**

  - *Chrome Browser for Windows x64*
  - *Chrome Browser for Debian x64*

2.4 Assumptions & Dependencies

The following dependencies are required to reproduce tests:

- **JUnit** Test Suite
- **GCC Compiler**
- **KarmaJS:** Installed globally via NPM
- **GulpJS:** Installed globally via NPM
- **Particle.io** Account

The following assumptions have been made:

- The user has constant internet access.
- The user's firewall is open to connections on port 1883, 9000, 8080, 80.
- The Application Server is hosted locally.

3 Functional Features to be tested.

The tests were implemented as follows for each of the components of the entire system. Each point represents an adequate test of the feature mentioned. **This section also outlines the item pass/fail criteria for each test.**

3.1 Firmware

Store to EEPROM using a struct.

1. Create the appropriate structure.
2. Attempt to store said structure.
3. Create an empty struct.
4. Attempt to store an empty struct.
5. Retrieve structs from EEPROM and verify that they are correct.

Store to EEPROM using raw values.

1. Declare raw values which need to be stored.
2. Attempt to store these acceptable values.

3. Retrieve data from EEPROM and verify that it is correct.

Retrieve from EEPROM.

1. Create multiple struct objects.
2. Store these struct objects to EEPROM.
3. Attempt to retrieve stored objects.
4. Verify that the objects are correct.
5. Clear Memory.
6. Attempt to retrieve objects again.

Check EEPROM for data.

1. Calculate how many struct objects can be stored in EEPROM.
2. Create enough struct objects to fill EEPROM to capacity.
3. Call appropriate function to determine if the EEPROM is full.
4. Create half the capacity of struct objects.
5. Store these struct objects to EEPROM.
6. Call appropriate function to determine if the EEPROM is full.

MQTT connect, which establishes a reliable connection to the server.

1. Provide incorrect host IP.
2. Attempt to connect.
3. Provide correct host IP.
4. Attempt to connect.

MQTT publish, which publishes JSON to the server.

1. Ensure connection to server is established.
2. Publish well-formed JSON.
3. Publish malformed JSON.
4. Ensure ACK packets are received on both occasions.

3.2 Application Server

Concurrent MQTT Client Connections.

1. Connect to server using Particle Photon.
2. Connect to server using MQTT Lens plugin.
3. Publish messages to server from photon and plugin concurrently.
4. Evaluate Firebase to ensure data is stored correctly.

Validate Data from Client.

1. Once a message has been received from a client.
2. Extract payload from message.
3. Parse JSON String into JSON object.
4. Catch exception if necessary.
5. Ensure JSON Object is accessible.

Store data to Firebase.

1. Receive a continuous stream of messages from MQTT Clients.
2. Attempt to store all messages to Firebase.
3. Verify that all messages are present.
4. Receive a continuous stream of messages with a 10 second delay.
5. Attempt to store messages to Firebase.
6. Verify that all messages are present.

3.2.1 Web Application

User Login

1. Test correct user credentials.
2. Test incorrect login credentials.
3. Verify API token exists after login.
4. Verify API token deleted after logout.

User Register

1. Test ideal user registration: Alphanumeric password 8+ characters long. Verify user email.
2. Test non-ideal user registration: Simple password should fail registration. Emails with non-FQTLDs: "temp@co" should fail registration.

Retrieve initial dashboard data

1. Verify Firebase object exists in controller.
2. Verify correct promise received from Firebase.
3. Retrieve all data from all devices.
4. Calculated averages, totals and carbon footprint values are accurate.

Retrieve list of devices

1. Verify Firebase object exists in controller.
2. Verify correct promise received from Firebase.
3. Retrieve all device meta details.

Retrieve device specific data

1. Verify Firebase object exists in controller.
2. Verify correct promise received from Firebase.
3. Retrieve all data for device based on device ID.

Retrieve data from date range

1. Verify Firebase object exists in controller.
2. Verify correct promise received from Firebase.
3. Retrieve all data for device based on device ID.

4 Other

4.1 Test Incident Report

4.1.1 Firmware

The following incidents occurred during firmware testing. Namely during: **Testing the connect and publish:**

1. Particle Photon disconnects from WiFi after concurrent publish events.
2. Particle Photon does not continue publishing after disconnecting.

Testing the EEPROM No incidents occurred with regards to testing the EEPROM.

4.1.2 Application Server

The following incidents occurred during application server testing. Namely during: **Storing to Firebase**

1. When receiving continuous messages from clients, some messages are not stored to Firebase.

5 Conclusions & Recommendations

5.1 Firmware

The current state of the Firmware is as follows.
The following tests have been implemented and assessed:

- Attempt to publish a message without establishing a reliable connection to the server.
- Attempt to overflow variables.
- Attempt store incorrect datatypes.
- Test if EEPROM clears once full.
- Check data persists after device loses power.
- Criteria which would cause the device to disconnect or reboot.

5.2 Application Server

The current state of the Application Server is as follows.
The following tests have been implemented and assessed:

- Test whether messages received from clients contain the correct data.
- Attempt to store malformed JSON to Firebase.
- Test store methods using parametrized test.
- Test *checkMonth()* with parametrized tests.
- Test *validateID()* using parametrized tests.
- Test appropriate exceptions which are included within the exceptions package.

The following tests need to be implemented and assessed:

- Load testing with 50 simultaneous connections.
- Load testing with 100 simultaneous connections.
- Load testing with 1000 simultaneous connections.

5.3 Web Application

The current state of the Web Application is as follows.
The following tests need to be implemented and assessed:

- Load testing with 50 simultaneous connections.
- Load testing with 100 simultaneous connections.
- Load testing with 1000 simultaneous connections.
- Test load time for 1000 database entries.

- Test dashboard graphs with 1000 database entries.

As a whole, the system functions as expected using expected parameters and under the correct conditions.

The next round of testing will attempt to put strain on the system through various load tests using load testing framework "Tsung".