5 Testing

Testing is an **extremely** important component of most projects, whether it involves a circuit, a process, power system, or software.

The testing plan should connect the requirements and the design to the adopted test strategy and instruments. In this overarching introduction, given an overview of the testing strategy and your team's overall testing philosophy. Emphasize any unique challenges to testing for your system/design.

In the sections below, describe specific methods for testing. You may include additional types of testing, if applicable to your design. If a particular type of testing is not applicable to your project, you must justify why you are not including it.

When writing your testing planning consider a few guidelines:

- Is our testing plan unique to our project? (It should be)
- Are you testing related to all requirements? For requirements you're not testing (e.g., cost related requirements) can you justify their exclusion?
- Is your testing plan comprehensive?
- When should you be testing? (In most cases, it's early and often, not at the end of the project)

5.1 Unit Testing

What units are being tested? How? Tools?

Three software units for testing - Leader selection, Display & UI, Sunspec Communication

To test these software units, we are going to be using multiple test cases to confirm that our software is functioning correctly.

Unit	**Unit Testing / Method**
Leader Selection	Test algorithm with thread teams before implementing on our Raspberry Pis. After moving the algorithm to Pis, confirm one leader is selected for each possible system change (addition of a device, removal of a leader/follow, one device on the network).
Display & UI	Write unit tests for display software and input software. Correct signals are sent to the Sunspec communication software.
Sunspec Communication	Test various read/write sequences to a single inverter to verify that modifications made are accurate and reliable. Do sequences for each mutable register. Run SunSpec compliance tests as provided by Sunspec alliance.

Hardware Units for testing - Network topology, Display & UI (Button input/ Display functional), Microgrid Outputs/Inputs

Unit	**Unit Testing / Method**
Network Topology	Ping devices across the network to ensure connections. Send a series of messages across the network and assert successful acknowledgement. Use basic command line pings.
Display & UI	Create software tests to probe input from individual buttons or other inputs. Confirm these are as expected. Example test: Connect button to RPi GPIO Press button, verify that correct response occurred (print message, etc.)
Inverter Configurations	Will discuss with PowerFilm contacts on how to test hardware to see if it works. Since we do not have much experience in power systems, it would be hard for us to come up with a testing environment for this hardware.

5.2 Interface Testing

What are the interfaces in your design? Discuss how the composition of two or more units (interfaces) are being tested. Tools?

GUI Hardware Testing

We will likely implement a test screen where a user can manipulate the control hardware, and see whether or not it is working properly. We are unsure how to implement testing beyond using the physical hardware and seeing if the system responds appropriately.

Raspberry Pi to Raspberry Pi Communication (TCP/IP)

Here we will have clearly laid out parameters designating our packet information. We should be able to check incoming packets, and make sure they are valid before the microcontrollers take any action with the data. We may use something like modbus as a protocol to follow for this communication.

Raspberry Pi to Sunspec Command/AXS (TCP/IP)

The AXS port is meant to receive sunspec commands, if an invalid command is created we should get notified by the AXS port, and/or we will be able to observe improper/missing configurations. We will also write sanity checks to verify that messages being sent are of the correct format and in accordance with the sunspec API. The sunspec API will most likely have some of this functionality built in.

AXS Port to Inverter/Microgrid (Modbus/Sunspec)

This connection/communication is largely proprietary and will be difficult to test. The communication here isn't necessarily defined by our team, but we will want to make sure it is working properly by verifying that our messages are being received by the microgrid. We can do this through simply

making configuration changes and reading these messages back. Modbus should send responses with codes that will verify changes that are received and let us know if it was successful (or alert us if there are issues)

5.3 Integration Testing

What are the critical integration paths in your design? Justification for criticality may come from your requirements. How will they be tested? Tools?

The critical integration paths are making sure the User Interface fits well with the rest of the codebase. While we have been assigned to program all the features of the old system into a more modern design, we have also been assigned to make a very user friendly interface. That being said, the user interface seems to be one of the project's main actors and it's important that we put a lot of attention into making sure all of the features of the interface display correctly and call the correct methods. We must also make sure changes to configuration made through our UI are enacted across all applicable components of the system. The scope of configurations we will provide has not yet been specified, but we will most likely verify the results manually across parts of the system.

5.4 System Testing

Describe system level testing strategy. What set of unit tests, interface tests, and integration tests suffice for system level testing? This should be closely tied to the requirements. Tools?

System level testing is end to end testing making sure the whole system works. So in our situation making sure that the microgrids are able to talk to each other with a simple test. We would have to write up some unit tests that check the code, where it shows that both the microgrids will be able to talk with each other. We will also need to verify that the commands being sent are in fact being received and the proper configuration changes are being enacted.

For system level testing, our project goals are to make the microgrid work in tandem with minimal user interaction. To test this requirement, we will create a simple instruction set for connecting the pallets. We will ask people without experience with the project to attempt to set them up.

5.5 Regression Testing

How are you ensuring that any new additions do not break the old functionality? What implemented critical features do you need to ensure they do not break? Is it driven by requirements? Tools?

To ensure new additions do not cause regressions, we will run tests on commits merged to the main branch making sure the application still works the same way. The most important features we will make sure are intact are the backend SunSpec Management methods and the potential TMS interface methods. These features are vital to the project and we must make sure that they are 100 percent functional. We will do this through building a Continuous Integration/ Continuous Delivery system into our GitLab.

5.6 Acceptance Testing

How will you demonstrate that the design requirements, both functional and non-functional are being met? How would you involve your client in the acceptance testing?

We will have branches corresponding to every feature that we will be adding to the project.

Alongside that, we will be using GitLab issues and milestones to keep track of our overall progress in the project. The descriptions of these issues and milestones alongside any changes made to the original plan during development will be relayed to the client during our meetings with them. We will also provide images of any major milestones or problems encountered so that the client is up to date as much as possible.

5.7 Security Testing (if applicable)

5.8 Results

What are the results of your testing? How do they ensure compliance with the requirements? Include figures and tables to explain your testing process better. A summary narrative concluding that your design is as intended is useful.

We have not conducted significant testing yet and do not have any testing results.