System Test Plan

Team Brew Science

By: Reagan Stovall, Andrew Klonitsko, Andrew Gates
Version: 1.0

Revision History Block

Version 0.1: Changes made include,

- Initialization and formatting of all portions of the document

Version 1.0: Changes made include,

- Each topic and subsequent topics of the System Test Plan has been filled out
    - 1. Introduction
    - 2. Test Items
    - 3. Procedures for Testing/Approach
    - 4. Item Pass/Fail Criteria
    - 5. Testing Deliverables
    - 6. Testing Requirements
    - 7. Roles/Responsibilities
    - 8. Schedule
    - 9. Testing Risks and Mitigation
    - 10. Approvals

Version 3.0: Changes made include,

- Modified sections to finalized information where applicable.
- Updated diagrams and datasheets for final setup.

1. Introduction

This test plan covers the Brew Science project, which involves the Automation Station and Specific Gravity sensor. This test plan covers revision 1.0 of the associated System Specification. It will outline all aspects associated with testing and verification of this project as a whole.

2. Test Items

2.1. Hardware Test Items - what must be tested and how?

2.1.1. Specific Gravity Sensor/TDC1000 Hardware

- Connect through ToF Mode_0 and read single TxRx pulse.
- Test Mode_3 Temperature.
- Test ToF measurement through various solids and distances to find accurate range of use.
- Test TDC1000 temperature compensation with actual measurements of pure water to compare density measurement integrity.

2.1.2. Automation Station Hardware

- ADC Chip - The Analog to Digital Converter will be tested separately from everything else in the circuit. It will be tested with a single sensor feeding into one input of the ADC chip, and then the output will be evaluated.
- Individual Multiplexers - The multiplexers will be tested via individual test cases that will change various inputs and evaluate the outputs.
- Combinational Multiplexers - The multiplexers will also be tested in combination with other multiplexers. Once they pass the individual test another multiplexer will be added to the chain and then tested to ensure correct output based on various inputs.
- Bias Resistor Multiplexer - Once all of the other multiplexers are combined and tested the bias resistor multiplexer will be created and tested to ensure that each resistor is within 1% of desired value, and that each resistor can be chosen correctly.
- Final Sensor Testing - Once the entire circuit is built various combinations of sensor types, pin types, and resistor types will be used to ensure desired functionality.

2.1.3. Features/functionality not to be tested - it is critical to define what will NOT be tested!

- All other Modes of the TDC1000.
- Every single possible sensor configuration. The main ones will be tested but there are 8 Digital ports with 2 different voltage options, 2 different pin combinations with 16 resistor combinations, as well as 8 Analog ports with the same number of options as well. This results in 1024 different sensor configurations and it will be impossible to test all of those in depth.
- All other aspects of the Raspberry Pi 3.

2.2. Software Test Items

2.1.1. Specific Gravity Sensor/TDC1000 Software

- Connect to the TDC1000 through SPI.
- Test reading Temperature.
- Test reading Time of Flight..

2.2.2. Automation Station Software

- Test that the GUI outputs all variables and functions in predefined format.
- Test that each and every available function is logical and acts accordingly with the storyboard feature.
    - This is a user feedback that helps the users to build and understand their functions.

2.2.3. Features/functionality not to be tested

- Restricted functions that do not logically make sense. These will be neglected as the User should not have the ability to create a restricted Function.

2.3. System Test Items

2.3.1. Features/functionality to be tested

- Database Management:
    - Connecting each component software to the Database and test writing/reading in real time from GUI, Phone App, and BackEnd program.
- Start-to-Finish GUI application setup.
- Real time monitoring from Phone App.
- TDC1000 ToF/Temperature reading from BackEnd Program.

2.3.2. Features/functionality not to be tested

- Every possible sensor/motor during every possible time frame.
- High level control from phone app.

3. Procedures for Testing / Approach

3.1. Special Tools - are any required to complete testing?

- Multimeter and/or Oscilloscope
- Hydrometer
- Thermometer

3.2. Training Required - Will testing require special training?

No, the only possible exception could be how to use a multimeter or oscilloscope.

3.3. Metrics - What metrics will be collected as part of testing?

- For testing the hardware of the automation station the main metric that will be collected is the input voltages and the output voltages of the multiplexers. These should be consistent among all multiplexers and.
- Another metric of the hardware portion of the automation station to be collected is the sensor values that are output based on various resistor and voltage configurations.

3.3.1. Which level is each metric to be collected at?

- For the multiplexers these metrics will be collected at each step before a new multiplexer is added. They will be collected individually and in combination with other multiplexers.
- For the sensor readings these will be collected only after the full scale system is setup.

3.4. Configuration - How is Configuration Management to be handled?

There will be 3 configuration Templates that will demonstrate the variety of options available to the user in the three timing setups.

3.4.1. How many different configurations will be tested?

- BrewSystem: One-Shot
    - A pre built template that with all ports and functions prespecified for a specific brew application.
- Traffic Logger: Continuous
    - A pre built template that monitors a single IR sensor and logs the time that it was crossed.
- Daily Weather Station: Periodic
    - The station wakes up at a specific time each day and records the Time, Light, Humidity, and Temperature.

3.5. Regression Testing -

3.5.1. What levels of regression testing will be done and how much at each test level?

Each feature is currently being tested and worked on individually as software and hardware. The advantage of this is that the hardware components specific to the software are being tested concurrently. The only exceptions that will require Regression testing are:

- Connection and sharing with the database from all the software applications.

3.5.2. Will regression testing be based on severity of defects detected?

No, if defects are detected they will most likely be hardware related. As long as the software is outputting the correct sensor configurations then it will be on the hardware side if some defect is observed.

3.6. How will elements in the requirements and design that do not make sense or are untestable be processed? What process will you follow?

If elements show up that do not make sensor or are untestable to be processed we will follow the process of identifying which portion of the system it came from (hardware, GUI, database, and etc.). Then based on which portion of the system it came from we will try to narrow it down as best as possible to which items it could be associated with. If it is associated with something that we have already tested we will retest all portions of that corresponding system. If it is something that is untestable we will attempt to remove the occurrence of that element through other means outside of testing.

4. Item Pass/Fail Criteria

4.1. How will you define success/failure/areas of concern for hardware components, software modules, and/or system design?

- Hardware Components - Success of the hardware components will be achieved if and only if each multiplexer works with every possible input/output combination, and the output value is within 5% of the desired value. If this is achieved and the ADC chip is tested and verified then the overall hardware system will be a success, due to the scalability of the system. Failure is defined as a system where even one possible sensor configuration does not work.
- Software GUI - Success will be based on whether or not a user with no coding or hardware experience can implement a simple program. This will be based on evaluating non-technical volunteers and asking them to implement a simple project. Feedback from the volunteers will result in improvements to the GUI
- Software Phone - Success will be based on real time monitoring/slight modifications of the running process.
- Software Specific Gravity Sensor - Success will be based on measurements with a tolerance of 0.2% density change after temperature compensation with. Failure will require reevaluation on possible alternatives or supplemental hardware depending on the point of failure.

- System Design - full system design success will be determined if all aspects of the project have be met and work together accordingly. These aspects are:
    - GUI, BackEnd, Phone App, Database, and SG sensor.

4.2. How will you characterize defects? Some may be critical, others might be ok. How will you define show-stoppers versus those that may degrade system functionality, but still allow the system to work.

The main defect that could be present would be in the variation of resistor values, and input/output voltages of the IC chips. These will degrade system functionality to a small extent, while still allowing the system to work. The only issue will be that sensors may have to be scaled differently based on the readings that are produced due to the variation in voltages and resistances.

4.3. Testing Suspension Criteria - when does it make sense to just wait on testing versus continued testing?

It only makes sense to suspend testing if a critical portion of the connection between two systems is not working. If the GUI and the hardware can't communicate, or the GUI and the database can't communicate then these will need to be fixed before testing can resume.

5. Testing Deliverables

5.1. Test plan

The overall test plan to be delivered is the one described in this document, as well as portions in the System Specification document which will be delivered as well.

5.2. Test case definition

This will outline each individual test to be performed.

5.3. Logs, testing reports, etc.

Logs from testing of each portion of the system will be produced along with what the desired functionality is, and what the observed functionality is. These will be concatenated into a testing report for each system.

5.4. Issue/Defect tracking

All issues and defects will be logged, along with whether they were fixed, mitigated, or unresolved.

6. Testing Requirements

6.1. Define what is needed for a team to conduct the tests outlined within this test plan.

Part Name	**Part ID**	**Quantity**
Raspberry Pi	Raspberry Pi 3	1
Analog Front End	TDC1000	1
2-to-1 Multiplexer	CD4066BE	2
8-to-1 Multiplexer	CD4051BE	2
16-to-1 Multiplexer	BOB-09056	1
Analog to Digital Converter	MCP3008	1
H-Driver	L293DD	1
12 bit ADC	ADS1015	1
SMD Resistors 100 - 1M	SMD 0603	20
.1 f Capacitors	SMD 0603	6
GPIO Expander	NXP PCA9555A	1

6.2. Define how test results will be provided to the project as well as testing feedback provided to engineering development team.

Each team will have a list of items to be tested, along with desired output/functionality of these items. The results from testing will be separated by team and recorded alongside the

desired results, with variance if possible. These lists will be provided to the engineering development team to determine if the testing was a success or failure.

7. Roles/Responsibilities

7.1. Who is in charge of the outlined testing?

Each staff member is required for testing an individual portion of the system (GUI, hardware, database/phone). While the entire staff will test the combination of these aspects.

7.2. What are the staffing requirements for carrying out the specified tests?

2 members are required with 3 being optimal.

8. Schedule

8.1. When will testing start?

Testing will start as soon as multiplexers are received. With individual multiplexers being the initial item to test, and the full interconnected system being the last item to test.

8.2. How many phases of testing?

3. The first phase is the testing of the individual hardware components. The second phase is the testing of the GUI as a single entity, the database/phone portion as a single entity, and the hardware as a single entity. The last phase is the testing of these three portions of the system combined. Testing of the Specific Gravity Sensor will occur concurrently through all phases.

8.3. Will testing be distributed across the project or will testing be done repeatedly at specific system development milestones?

For hardware testing will be distributed across the project. Whereas the GUI and database/phone will be tested and specific milestones.

9. Testing Risks and Mitigation

9.1. Identify risks to project (in terms of the schedule defined) based on testing requirements.

- The main risk to the project is the individual components that will be used in the system as a whole. If these are malfunctioning this might slow down development schedule if new components have to be obtained.
- Another risk could be due to a malfunction between portions of the system. If two aspects of the system aren't communicating correctly then both have to be evaluated and cannot be tested further until it is resolved. Which results in other aspects being unable to be tested as well.
- A risk might also occur in human error. This could be due to one of the members not testing as rigorously as the other members. Or possibly one of the members leaving the project entirely.

9.2. How will you notify the project of problems encountered during testing?

We are currently build our application with combination of a waterfall method with a firm outline of an iterative method. Problems that present themselves in the main software application of the project, the GUI, Backend, Phone app, and Database will be dealt with as they present themselves. The Hardware components are less predictable and coupled with ongoing testing as well as regressive testing, any new problems will evaluated by the application manager and if the manager does not feel that they can fix the problem on their own, they should bring the problem to the attention of the rest of the group. From there a solution or revision will be made.

9.3. How might risks be mitigated? Add people? Add resources? How many? What type of resources?

The main way to mitigate risks is to add people. Adding resources doesn't mitigate risks at all in a project like this where a minimum number of resources are required. If more people are added then there testing can be more rigorous. The optimal amount of people to add would be 3, so that each portion of the system can have 2 people testing it, verifying with each other or testing different portions of their system concurrently.

9.4. Are there capital costs associated with the test plan?

There are little capital costs associated with the test plan besides the cost of hiring more people. Discounting the cost of the items to produce the system, the only other items that will need to be purchased are a multimeter, thermometer and hydrometer.

10. Approvals

10.1. Who must approve the master test plan?

The entire group as a whole should approve of the master test plan.

10.2. Who must approve the unit test test plan?

The members who were not assigned to that portion of testing should be the ones to approve the unit test plan of that aspect of the system.