Maseeh College of Engineering and Computer Science

PORTLAND STATE UNIVERSITY

Test Plan

Prepared By:

Ali Saad

Jonathan Christian

Nick Long

Jiaqi Liu

Prepared For:

Joe Kiniry

Joey Dodds

Daniel Zimmerman

Revision Number: 3

Date of Revision: May 29, 2020

Date of Document Submission: June 12, 2020

FREE & FAIR

Table of Contents

I. Introduction

1. Product Overview

The SBB 2019 - PSU Edition is an altered version of the existing SBB 2019, which replicates the functionality and security of the original implementation while improving the economical value by adapting to an Arduino UNO platform.

The purpose of this "smart" ballot box (SBB) is to allow voters to either cast or spoil a ballot, securely recording the choice that is made. "Spoiling" a ballot at the SBB means that the voter is voiding the ballot that was inserted. Voters typically do so either because they realized that they made a mistake on their ballot or they changed their mind about their vote.

This document contains a summary of the SBB 2019- PSU Edition that will be tested and encompasses the necessary tests that should be performed to assess the functionality and security of this product against the sponsor's requirements.

This overview is followed by Section II, which lists the reference documents used in this project. In Section III, the test plan continues with identifying the modules, features, and type of testing to be performed. The testing schedule of all testing activities, resources, and the risks associated with the plan are described in Section IV.

2. Out of Scopes

- SBB 2019 - PSU Edition enclosure and external peripherals
- Ballot Marking Device (BMD)
- Networked communication
- Product/Prototype Production
- Network logging and DEFCON debugging

II. Reference Documents

1. Industry Standards

- Universal Serial Bus (USB)
- Arduino UNO R3
- I2C and UART TTL Communication Protocol
- ISO/IEC 18004:2015 *Information – Automatic identification and data capture techniques – QR Code barcode symbology specification*

2. Design Documents

- SBB 2019 - PSU Edition Project Description

- SBB 2019 - PSU Edition Project Proposal UPDATED 4/5/20
- SBB 2019 - PSU Edition Initial Design Specifications
- SBB 2019 - PSU Edition Project Schedule UPDATED 4/5/20

3. Test Plan Documents

- "How Can a Test Plan Software Help in IEEE 829 Standard Compliance?" *ReQtest*, 25 Nov. 2019, https://reqtest.com/testing-blog/how-to-write-a-test-plan-2/
- "The Pros & Cons of Using a Test Plan" *Reqtest*, 11 Nov. 2016, https://reqtest.com/testing-blog/using-test-plan/
- Sonali, et al. "Tips to Write Simple & Effective Test/QA Plan - [Sample Test Plan Report to Download]." *Opencodez*, 17 June 2019, www.opencodez.com/software-testing/test-planning.htm

III. Objective Statements:

The objectives of this test plan are to layout and describe the individual modules of the SBB 2019 - PSU Edition (Unit Test), test the integration between modules as described in the Design Specifications and the Project Proposal document (Integration Test), test the ease of use with a small internal group (Alpha Test) and then test with a large group of external user through actual use (Beta Testing). Two additional tests have been included, test the final prototype while comparing the operation of the previous implementation (Regression Test) and then test that all agreed-upon deliverables are provided with the final product. Developers and testers of SBB 2019 - PSU Edition will put the capstone deliverables through those tests (unit, integration, alpha, beta, regression and acceptance testing), to ensure proper operation according to the design documentation.

In the next section, we have the test cases with their unique test identifiers grouped to fulfill the following six objectives:

1. Unit Testing

2. Integration Testing

3. Alpha Testing

4. Beta Testing

5. Regression Testing

6. Acceptance Testing

IV. List of Test Cases Grouped by Objective:

1. Objective 1- Unit testing:

a. **Test case A - Motor Indicator Test #1 (Test ID: UT-MIT-1):** This test is to verify if the motor indicators on the CASCADIO board initialize and indicate proper operation in both directions. A short Arduino program will be loaded and run on the UNO where high and low signals are sent to the indicators. They will be visually inspected to determine the operation.

   Duration: This test can be run as many times as needed to visually verify both indicators on the CASCADIO light up. The sequence depends on if the ballot is taken in or returned.

   Conditions: Arduino UNO must be plugged into a power source (USB or barrel jack) and loaded with an appropriate test program that will send high and low signals to the PaperIn/PaperOut LEDs on the CASCADIO board.

b. **Test case A1 - Motor Test #1 (Test ID: UT-MT-1):** This test is to verify if the motor connected to the CASCADIO board initializes and rotates in the proper direction. Both clockwise and counterclockwise should be achieved. A short Arduino program will be loaded and run on the UNO where high and low signals are sent to the motor outputs. This will be visually and audibly checked to determine proper operation.

   Duration: This test can be run as many times as needed to visually/audibly verify the motor rotates in both directions depending on signal inputs.

   Conditions: The 12V power supply must be connected to the CASCADIO board and the external motor needs to be connected to the dual-pin PMOD connector labeled *motor*. Arduino UNO must be plugged into a power source (USB or barrel jack) and loaded with an appropriate test program that will send high and low signals to the corresponding pins on the shield. Jumpers need to connect D6 & D7 on the microcontroller to pins 10 & 9 respectively on the CASCADIO.

c. **Test case B - Barcode Scanner Test #1 (Test ID: UT-BS-1):** This test is to verify if the externally connected Waveshare Barcode Scanner is being supplied with 5V and beeps upon a recognized barcode that is compatible. A short Arduino program will be loaded and run on the UNO where the barcode data can be observed through the serial monitor within the Arduino IDE.

   Duration: This test can be run as many times as needed to visually verify the scanner produces a white and red light as well as an audible beep upon the scanning of a recognized barcode.

   Conditions: The 12V power supply must be connected to the CASCADIO board and the barcode reader needs to be connected to the quad PMOD connector labeled *scanner*. Arduino UNO must be plugged into a power source (USB or barrel jack) and loaded with an appropriate test program that will recognize and react when a barcode is captured.

Jumpers need to connect D0 & D1 on the microcontroller to pins 8 & 7 respectively on the CASCADIO.

d. **Test case C - Paper Sensor Test #1 (Test ID: UT-PS-1):** This test is to verify if the sensor buttons on the CASCADIO board function properly and activate the corresponding LED's. A short Arduino program will be loaded and run on the UNO where the signals can be monitored through the serial monitor within the Arduino IDE.

   Duration: This test can be run as many times as needed to visually verify the paper sensor is activated. This can be observed by focusing on the LED5 & LED6 on the shield, which activates after the corresponding button is pressed to simulate the sensor.

   Conditions: Arduino UNO must be plugged into a power source (USB or barrel jack) and loaded with an appropriate test program that will recognize and react when the SENSE0 & SENSE1 buttons are pressed. Jumpers need to connect A2 & A3 on the microcontroller to pins 16 & 15 respectively on the CASCADIO.

e. **Test case D - Cast/Spoil Button Test #1 (Test ID: UT-CSB-1):** This test is to verify if the cast/spoil buttons on the CASCADIO board light up and function properly. A short Arduino program will be loaded and run on the UNO where the different button pushes can be observed through the serial monitor within the Arduino IDE.

   Duration: This test can be run as many times as needed to visually verify the cast and spoil buttons function. This can be observed by focusing on the button LEDs when power is supplied.

   Conditions: Arduino UNO must be plugged into a power source (USB or barrel jack) and loaded with an appropriate test program that will recognize and react when the cast and spoil buttons are pressed. Jumpers need to connect A2 & A3 on the microcontroller to pins 16 & 15 respectively on the CASCADIO.

f. **Test case E - LCD Screen Test #1 (Test ID: UT-LS-1):** This test is to verify if the LCD screen on the CASCADIO board is being supplied with 3V and characters are properly displayed on the screen. A short Arduino program will be loaded and run on the UNO where the LCD screen can be observed while data gets sent through the serial monitor within the Arduino IDE.

   Duration: This test can be run as many times as needed to visually verify the LCD screen backlight is on and that readable characters are displaying on the screen. This can be observed by focusing on the LCD screen affixed to the CASCADIO board when 3.3V power is supplied.

   Conditions: Arduino UNO must be plugged into a power source (USB or barrel jack). The Arduino does not need to be loaded with a test file to check if the LCD backlight activates and is receiving power. However, when determining if values are displaying properly, the Arduino must be loaded with an appropriate test program that will provide a test message. Refer to the serial monitor while this test is occurring as the values will be

displayed there as well. Jumpers need to connect A4 & A5 on the microcontroller to pins 13 & 14 respectively on the CASCADIO.

g. **Test case F - Real-Time Clock Test #1 (Test ID: UT-RTC-1):** This test is to verify if the RTC pulls the correct date and time from the initializing computer and counts in real-time. A short Arduino program will be loaded and run on the UNO where the RTC will be set and then it can be observed through the serial monitor within the Arduino IDE.

   Duration: This test can be run as many times as needed to visually verify that the RTC is set correctly. The goal is to visually verify the date and time is current in the LCD screen and serial monitor.

   Conditions: Arduino UNO must be plugged into a power source (USB or barrel jack) and the backup button cell is installed. Be sure to run the Button Cell BAckup Test before continuing on here. The Arduino does not need to be connected to the CASCADIO board to run this test, but it would be a good idea. Arduino must be loaded with an appropriate test program that will set and read the RTC. Be sure to have the correct RTC library associated otherwise it may not initialize. Refer to the serial monitor while this test is occurring as the values will be displayed there. Jumpers need to connect A4 & A5 on the microcontroller to pins 13 & 14 respectively on the CASCADIO.

h. **Test case G - SD Card Test #1 (Test ID: UT-SD-1):** This test is to verify if the SD card is recognized and can save logging data in a CSV file format. A short Arduino program will be loaded and run on the UNO where SD logging data can be observed through the serial monitor within the Arduino IDE. Once this test is run the data should be available on the SD card. Be sure to verify this before moving on.

   Duration: This test can be run as many times as needed to verify that the logging feature is operating correctly and saving the button presses accordingly.The goal is to visually verify the button presses are being shown on the serial monitor.

   Conditions: Arduino UNO must be plugged into a power source (USB or barrel jack) and the SD card must be formatted to either FAT16 or FAT32 file systems for proper recognition. Next, jumpers need to connect D10, D11, D12 & D13 on the microcontroller to pins 1, 2, 3 & 4 respectively on the CASCADIO. This requires the cast and spoil buttons to be connected to the system as they provide the data that will be saved on the card. Refer to the serial monitor while this test is occurring as the values will be displayed there.

i. **Test case H - Button Cell Backup Test #1 (Test ID: UT-BAT-1):** This test is to verify if the 3V lithium button cell is inserted and providing proper voltage to the RTC. A multimeter will be used to determine voltage level and ensure RTC continues to be powered while unplugged.

   Duration: This test can be run as many times as needed to verify that the button cell is 3.3V before installation. The goal is to visually verify the battery is not dead, it gets installed properly and that the power is transferred to the RTC.

Conditions:

j. Test case I - Arduino UNO Test #1 (Test ID: UT-AU-1): This test is to verify if the Arduino UNO functions properly and programs can be loaded onto the microcontroller. Test programs will be uploaded and observe the changes as well as test the output pins.

Duration: This test can be run as many times as needed to verify that

Conditions: Arduino UNO must be plugged into a power source (USB or barrel jack)

k. Test case J - Makefile Test #1 (Test ID: UT-MF-1): This test is to verify that the makefile compiles successfully.

Duration:

Conditions:

l. **Test case K - CASCADIO Board Test #1 (Test ID: UT-CB-1):** This test is to verify that the board header pins are connected properly. A continuity test will determine success.

Duration:

Conditions:

m. **Test case L - FreeRTOS #1 (Test ID: UT-FR-1):** This test is to verify that the FreeRTOS software compiles and counts when loaded on the Arduino UNO.

Duration:

Conditions:

n. **Test case M - SBB Software Test #1 (Test ID: UT-SBB-1):** This test is to verify that the SBB functionality software compiles and can be loaded onto the Arduino UNO properly.

Duration:

Conditions:

o. **Test case M - Memory Test #1 (Test ID: UT-MT-1):** This test will verify the size of the necessary software files and compare against Arduino UNO memory allocation and space.

Duration:

Conditions:

p. **Test case N - Ballot QR Code Test #1 (Test ID: IT-QR-1):** This test is to verify the quality and orientation of the QR code to ensure proper recognition from the scanner.

Duration:

Conditions:

2. Objective 2 - Integration testing:

a. **Test case A - Arduino FreeRTOS Test #1 (Test ID: IT-AF-1):** Tester loads the Arduino Uno with a compatible version of FreeRTOS and observes LED timing sequence.

   Duration:

   Conditions:

b. **Test case B - Arduino SBB Test #1 (Test ID: IT-AS-1):** Tester has Arduino properly loaded with FreeRTOS. Then the tester compiles the SBB functionality software onto the existing program.

   Duration:

   Conditions:

c. **Test case C - Ballot Scan Test #1 (Test ID: IT-BS-1):** Tester has the SBB2019 - PSU Edition powered on and in standby. The tester inserts a ballot and observes if the paper sensor is activated on the enclosure. To simulate this test using the CASCADIO board press the sensor 1 button and ensure Arduino is receiving input and corresponding LED illuminates.

   Duration:

   Conditions:

d. **Test case D - Ballot Scan Test #2 (Test ID: IT-BS-2):** Tester has the SBB2019 - PSU Edition powered on and in standby. The tester inserts a ballot and observes if the intake motor activates and pulls the ballot in. To simulate this test using the CASCADIO board, be sure an external motor is connected to the motor PMOD connector and is receiving a 12V supply.

   Duration:

   Conditions:

e. **Test case E - Ballot Scan Test #3 (Test ID: IT-BS-3):** Tester has the SBB2019 - PSU Edition powered on and in standby. The tester inserts a ballot and observes if the barcode scanner recognizes the affixed QR code and beeps to indicate successful reading. To simulate this test using the CASCADIO board, be sure the Waveshare barcode scanner is connected to the scanner PMOD connector and is receiving 5V rectified from the 12V source.

   Duration:

   Conditions:

f. **Test case F - Ballot Scan Test #4 (Test ID: IT-BS-4):** Tester has the SBB2019 - PSU Edition powered on and in standby. The tester inserts a ballot and follows through the series of progressions and observes if the ballot is deposited or returned based on the cast/spoil option selected. To simulate this test using the CASCADIO board, be sure the motor activates for a spoil selection or keeps the ballot for a cast selection.

   Duration:

   Conditions:

3. Objective 3 - Alpha testing:

a. **Test case A - Internal Group Test #1 (Test ID: AT-IG-1):** Our group will test the overall basic functionality of the SBB 2019 - PSU Edition and gather user feedback to address ease of use issues. This includes completing the entire ballot submission process which includes ballot insertion, LCD instruction, scanner operation, and cast/spoil functions in series. Modifications will be made based on observations and experiences with the SBB.

   Duration:

   Conditions:

b. **Test case B - Internal Group Test #2 (Test ID: AT-IG-2):** Our group will test the advanced functionality of the SBB 2019 - PSU Edition and gather user feedback to address ease of use issues. This includes analyzing the data being transferred between each module which includes Arduino signal outputs, logging, and CASCADIO board thoroughly. Modifications will be made based on observations and experiences with the SBB.

   Duration:

   Conditions:

4. Objective 4 - Beta testing:

a. **Test case A - External Group Test #1 (Test ID: BT-EG-1):** Another group will test the overall basic functionality of the SBB 2019 - PSU Edition and gather user feedback to address ease of use issues. This includes completing the entire ballot submission process which includes ballot insertion, LCD instruction, scanner operation, and cast/spoil functions in series. Modifications will be made based on observations and experiences with the SBB.

   Duration:

   Conditions:

b. **Test case B - External Group Test #2 (Test ID: BT-EG-2):** Another group will test the advanced functionality of the SBB 2019 - PSU Edition and gather user feedback to address ease of use issues. This includes analyzing the data being transferred between

each module which includes Arduino signal outputs, logging, and CASCADIO board thoroughly. Modifications will be made based on observations and experiences with the SBB.

Duration:

Conditions:

5. Objective 5 - Regression testing:

a. **Test case A - Previous Version Test #1 (Test ID: RT-PV-1):** This test verifies that all testing procedures completed by Free & Fair on the SBB12019 are run on the SBB2019 - PSU Edition to determine identical functionality. This also ensures that unintended changes in behavior or performance were introduced.

   Duration:

   Conditions:

6. Objective 6 - Acceptance testing:

a. **Test case A - Deliverables Test #1 (Test ID: AT-D-1):** This test verifies that all aspects of the agreed-upon deliverables are present in the SBB2019 - PSU Edition.

   - We will have the same functionality as the original SBB2019 (except network logging and DEFCON debugging) as described in the Github BVS2019 project with the deployment version of our choosing. However, physical housing is excluded.
   - The Arduino Uno will interface with the peripherals identically like the FPGA.
   - The system will be easy to replicate and program with a COTS microcontroller.
   - The cost will be below $500 to allow others to replicate our work.
   - The overall design will allow for easy exploration, debugging, modification, and experimentation for the future development of the Smart Ballot Box concept. This System is not a final product but apart from a more widespread effort to improve voting system technology.

   Duration:

   Conditions:

V. Selective Test Cases in Details (Matrix):

1. Full Operation Test #1 (Test ID: FT-FO-1)

Test Writer: Jonathan Christian., EE					
Test Case Name:		Full Operation Test #1	**Test ID#**		Test ID: FT-FO-1
Description:		This test is to check if all peripherals are communicating with the AVR microcontroller (Arduino UNO) and the basic functionality matches that of the SBB 2019 implementation. This can be performed using the CASCADIO board or the full prototype setup enclosure.	**Type:**		☑Black Box ☐White Box
Tester's Information					
Name of Tester		Jiaqi Lui, EE	**Date**		5/14/2020
Hardware Ver:		2.0	**Time**		3:00 PM
Setup:		Have the microcontroller connected to the CASCADIO or breakout board. Also, ensure peripheral components are properly attracted to the PMOD connections on the board.			
Step	**Action**	**Expected Results**	**Pass**	**Fail**	**Comment**
1	Turn on PC with Ubuntu OS capable of running AVRdude	The machine boots up and displays the default desktop screen.			
2	Plugin the 12V power supply to the wall and the USB A-B cable	The CASCADIO board sends power to the			

	into a computer. Then connect the 12V to the CASCADIO board.	peripheral sensors and they light up. The Arduino receives power and displays illuminated LEDs.			

3	Open AVRdude (ver ###) and Run the following:
	`git clone git@github.com:jonathancpdx/Capstone-BallotBox.git`
	
	`cd Capstone-BallotBox/AVRDUDE`
	
	`sudo apt-get install vim avrdude avrdude-doc avr-gcc avr-libc`
	
	`chmod 755 compile upload`
		Clones GitHub repository containing saved software files and install necessary libraries for proper operation.			

4	Take the A-B USB cable and connect the Arduino UNO (microcontroller) to the PC. Select the correct board within AVRdude by going to **Devices** -> **USB**, check the box next to your desired device.	The Arduino UNO powers on and AVRdude recognizes the UNO is connected to the Virtual Machine.			

5	Run the commands within AVRdude to compile and load the correct programs on the microcontroller.
	`sudo ./compile`
	`sudo ./upload`
		The UNO accepts the upload and activates the LCD screen on the CASCADIO board. SBB functionality is up and running.			

6	Insert a valid ballot or press the intake paper sensor button on the CASCADIO board so the motor rotates.	The ballot is pulled into the SBB2019 - PSU Edition or the paper feed indicators light up on the CASCADIO board. The barcode scanner scans the ballot QR code.			
7	Observe the instructions displayed	Ballot will be deposited			

	on the LCD screen. Press cast/spoil buttons depending on desired action.	inside the box if cast or returned if spoiled.			
8	Repeat the steps 1-7 for further testing of operations.	SBB2019 - PSU Edition should yield the same results for steps 1-7 after each ballot insert.			
Overall Result					

2. **Full Operation Test #2 (Test ID: FT-FO-2)**

Test Writer: Jonathan Christian., EE				
Test Case Name:	Full Operation Test #2	**Test ID#**		Test ID: FT-FO-2
Description:	This test is to check if all peripherals are communicating with the AVR microcontroller (Arduino UNO) and the advanced functionality matches that of the SBB 2019 implementation. This can be performed using the CASCADIO board or the full prototype setup enclosure.	**Type:**		☐Black Box ☑White Box
Tester's Information				
Name of Tester	Jiaqi Lui, EE	**Date**		5/21/2020
Hardware Ver:	2.0	**Time**		9:00 AM
Setup	Have the microcontroller connected to the CASCADIO or breakout board. Also, ensure peripheral components are properly attracted to the PMOD connections on the			

		board.			
Step	**Action**	**Expected Results**	**Pass**	**Fail**	**Comment**
1	Turn on PC with Ubuntu OS capable of running AVRdude	The machine boots up and displays the default desktop screen.			
2	Plug in the 12V (+ tip) power supply into a 120VAC outlet and the USB A-B cable into the computer. Then connect the 12V plug into the CASCADIO board.	The CASCADIO board sends power to the peripheral sensors and they light up. The Arduino receives power and displays illuminated LEDs.			

3	Open AVRdude (ver ###) and Run the following:
	`git clone git@github.com:jonathancpdx/Capstone-BallotBox.git`
	
	`cd Capstone-BallotBox/AVRDUDE`
	
	`sudo apt-get install vim avrdude avrdude-doc avr-gcc avr-libc`
	
	`chmod 755 compile upload`
		Clone's GitHub repository containing saved software files creates new directory file and installs necessary libraries for proper operation.			

4	Take the A-B USB cable and connect the Arduino UNO (microcontroller) to the PC. Select the correct board within AVRdude by going to **Devices** -> **USB**, check the box next to your desired device.	The Arduino UNO powers on. The ON LED illuminates and the Tx LED starts to blink. AVRdude recognizes the UNO is connected to the Virtual Machine.			
5	Run the commands within AVRdude to compile and load the correct programs on the microcontroller. Start with FreeRTOS and then place the SBB set after. `sudo ./compile`	The UNO accepts the upload and activates the LCD screen on the CASCADIO board. SBB functionality is up and running.			

	`sudo./upload`				
6	Insert a ballot (contains a QR code in the right-hand corner) or press the intake paper sensor button on the CASCADIO board so the motor rotates.	The ballot is pulled into the SBB2019 - PSU Edition because a 1 from the paper sensor turns the motor. Next, the barcode scanner scans the ballot QR code and the microcontroller verifies several parameters before continuing. Such are time, correct QR info, and valid session. If those are passed the LCD will display a prompt. While using the CASCADIO board, the paper feed indicators light up.			
7	Observe the instructions displayed on the LCD screen. Press cast/spoil buttons depending on the desired action.	Ballot will be deposited inside the box if cast or returned if spoiled.			
8	Repeat steps 1-7 for further testing of operations.	SBB2019 - PSU Edition should yield the same results for steps 1-7 after each ballot insert.			
Overall Result					

IV. Test Scheduling, Resources, and Associated Risks

1. Test Scheduling

Test Each Module Separately on CASCADIO Board	4/1/2020
Test FreeRTOS in Development Environment	4/21/2020
Test SBB Software in Development Environment	4/3/2020
Test FreeRTOS Functionality on Arduino UNO	5/15/2020

Test SSB Functionality using CASCADIO Board	5/19/2020
Test Full Operation of the Smart Ballot Box Prototype	5/22/2020
Test SBB2019 - PSU Edition using Outside Testers	5/27/2020

2. Resources

The resources required for testing are:

- Digital Multimeter (EXTECH instrument)
- Windows 10 Computer with Arduino IDE (version 1.8.12) installed
- Linux Ubuntu Computer with AVRdude installed
- 12V Power Supply
- Assorted clips and probes (alligator clips, banana clips, test leads)
- Arduino test files for unit testing
- CASCADIO board for hardware integration testing
- Arduino UNO Rev3 board and corresponding USB cable
- WaveShare Barcode Scanner
- Ballot marking Device or other source of QR code generation
- Documentation and datasheets for all components (ATmega328P) for pinout diagrams
- Two Team Members focused primarily on Software Development (Ali & Nick)
    - Education level: Undergraduate
    - Requires skills and knowledge related to:
        - Arduino and embedded systems programming
        - AVRdude or other embedded device programming environment
        - Debugging with an IDE
        - Github
        - C Programming
- Two Team Members focused primarily on Hardware Development (Jonathan & Jiaqi)
    - Education level: Undergraduate
    - Requires skills and knowledge related to:
        - Schematics and Diagrams
        - EagleCAD or other modeling/circuit layout software
        - Oscilloscope and multimeter debugging experience
        - Arduino IDE
        - Microcontroller Chipset
        - Soldering and device assembly process
- Sources of funding
- Sponsor mentoring
- Work space for prototype testing and tools

3. Associated Risks

The possible risks of the testing phase include:

- Damaging one of our CASCADIO boards or Arduino Uno

- Losing critical files needed for operation
- Not having replacement parts
- Bugs and glitches that are difficult to resolve, slowing down progress
- Not having proper facilities to perform all testing
- Not completing the tests on time
- Power or internet outage that interfere with testing and communication
- Difficulty integrating the software and hardware components of the SBB 2019 - PSU Edition
- Team members are unable to be present for testing due to external factors
- Missing test equipment