Test Plan and Results

Overall Test Plan

Due to the modular nature of my senior design project, testing will include a combination of both unit & integration tests. This is necessary to ensure proper coverage of such a wide variety of functionality, split across numerous modules/libraries. Each of the individual libraries (git submodules) will have extensive unit tests covering their respective internal functionality. Then, a set of integration tests must be implemented to verify the cross-module compatibility. Since this project has such strong dependence on Bluetooth, many of our test cases will have a mock Bluetooth endpoint to simulate the expected physical device. This allows the tests to send/receive simulated data as if it were in normal operation. These are the two key aspects to my testing strategy. By implementing a thorough combination of unit & integration tests, as well as mocking physical Bluetooth devices wherever necessary, the project should have extensive code coverage as a result.

Some Background Info

*For the sake of brevity, I've only added a subset of tests to this document. The entire templated Android architecture has a **lot** more than what's shown. I've focused on the unit tests for one of the services in the communication spec we designed & implemented. Thus, the individual test cases shown below are endpoints in our communication spec. The goal of each test is merely to verify that the proper input command (with a custom header we define in the communication spec), reaches the proper endpoint. The communication spec library/module can be thought of a giant traffic router, and the tests serve to verify that nothing is incorrectly routed. Additionally, I've added some of the Pub/Sub tests which help ensure functionality for our custom Pub/Sub, which enables cross-module communication within my senior design project.*

Test Case Descriptions

CCV1.1 **Command Characteristic Version - Manifest Version Test**
CCV1.2 This test verifies the PE base service, command characteristic, manifest version endpoint of the PE communication specification.
CCV1.3 This test creates a mock byte array with the appropriate header bytes (following the communication specification) to reach the proper endpoint.
CCV1.4 Inputs: There are no inputs for this test. However, the mock byte array message is created internally in order to test the routing path.
CCV1.5 Outputs: To validate proper endpoint routing, a globally-accessible string is modified & verified within this test.
CCV1.6 Normal
CCV1.7 Whitebox
CCV1.8 Functional
CCV1.9 Unit Test
CCV1.10 Results: The globally-accessible endpoint string was modified to the expected result, verifying the routing path.

CCV2.1 **Command Characteristic Version - CP Application Version Test**
CCV2.2 This test verifies the PE base service, command characteristic, CP application version endpoint of the PE communication specification.
CCV2.3 This test creates a mock byte array with the appropriate header bytes (following the communication specification) to reach the proper endpoint.
CCV2.4 Inputs: There are no inputs for this test. However, the mock byte array message is created internally in order to test the routing path.
CCV2.5 Outputs: To validate proper endpoint routing, a globally-accessible string is modified & verified within this test.
CCV2.6 Normal
CCV2.7 Whitebox
CCV2.8 Functional
CCV2.9 Unit Test
CCV2.10 Results: The globally-accessible endpoint string was modified to the expected result, verifying the routing path.

CCV3.1 **Command Characteristic Version - CP Bootloader Version Test**
CCV3.2 This test verifies the PE base service, command characteristic, CP bootloader version endpoint of the PE communication specification.
CCV3.3 This test creates a mock byte array with the appropriate header bytes (following the communication specification) to reach the proper endpoint.
CCV3.4 Inputs: There are no inputs for this test. However, the mock byte array message is created internally in order to test the routing path.
CCV3.5 Outputs: To validate proper endpoint routing, a globally-accessible string is modified & verified within this test.
CCV3.6 Normal

CCV3.7 Whitebox
CCV3.8 Functional
CCV3.9 Unit Test
CCV3.10 Results: The globally-accessible endpoint string was modified to the expected result, verifying the routing path.

CCV4.1 **Command Characteristic Version - BLE Version Test**
CCV4.2 This test verifies the PE base service, command characteristic, BLE version endpoint of the PE communication specification.
CCV4.3 This test creates a mock byte array with the appropriate header bytes (following the communication specification) to reach the proper endpoint.
CCV4.4 Inputs: There are no inputs for this test. However, the mock byte array message is created internally in order to test the routing path.
CCV4.5 Outputs: To validate proper endpoint routing, a globally-accessible string is modified & verified within this test.
CCV4.6 Normal
CCV4.7 Whitebox
CCV4.8 Functional
CCV4.9 Unit Test
CCV4.10 Results: The globally-accessible endpoint string was modified to the expected result, verifying the routing path.

CCV5.1 **Command Characteristic Version - BLE Bootloader Version Test**
CCV5.2 This test verifies the PE base service, command characteristic, BLE bootloader version endpoint of the PE communication specification.
CCV5.3 This test creates a mock byte array with the appropriate header bytes (following the communication specification) to reach the proper endpoint.
CCV5.4 Inputs: There are no inputs for this test. However, the mock byte array message is created internally in order to test the routing path.
CCV5.5 Outputs: To validate proper endpoint routing, a globally-accessible string is modified & verified within this test.
CCV5.6 Normal
CCV5.7 Whitebox
CCV5.8 Functional
CCV5.9 Unit Test
CCV5.10 Results: The globally-accessible endpoint string was modified to the expected result, verifying the routing path.

CCV6.1 **Command Characteristic Version - Hardware Revision ID Test**
CCV6.2 This test verifies the PE base service, command characteristic, hardware revision ID endpoint of the PE communication specification.
CCV6.3 This test creates a mock byte array with the appropriate header bytes (following the communication specification) to reach the proper endpoint.

CCV6.4 Inputs: There are no inputs for this test. However, the mock byte array message is created internally in order to test the routing path.

CCV6.5 Outputs: To validate proper endpoint routing, a globally-accessible string is modified & verified within this test.

CCV6.6 Normal

CCV6.7 Whitebox

CCV6.8 Functional

CCV6.9 Unit Test

CCV6.10 Results: The globally-accessible endpoint string was modified to the expected result, verifying the routing path.

CCV7.1 **Command Characteristic Version - Device ID Test**

CCV7.2 This test verifies the PE base service, command characteristic, device ID endpoint of the PE communication specification.

CCV7.3 This test creates a mock byte array with the appropriate header bytes (following the communication specification) to reach the proper endpoint.

CCV7.4 Inputs: There are no inputs for this test. However, the mock byte array message is created internally in order to test the routing path.

CCV7.5 Outputs: To validate proper endpoint routing, a globally-accessible string is modified & verified within this test.

CCV7.6 Normal

CCV7.7 Whitebox

CCV7.8 Functional

CCV7.9 Unit Test

CCV7.10 Results: The globally-accessible endpoint string was modified to the expected result, verifying the routing path.

CCV8.1 **Command Characteristic Version - Customer ID Test**

CCV8.2 This test verifies the PE base service, command characteristic, customer ID endpoint of the PE communication specification.

CCV8.3 This test creates a mock byte array with the appropriate header bytes (following the communication specification) to reach the proper endpoint.

CCV8.4 Inputs: There are no inputs for this test. However, the mock byte array message is created internally in order to test the routing path.

CCV8.5 Outputs: To validate proper endpoint routing, a globally-accessible string is modified & verified within this test.

CCV8.6 Normal

CCV8.7 Whitebox

CCV8.8 Functional

CCV8.9 Unit Test

CCV8.10 Results: The globally-accessible endpoint string was modified to the expected result, verifying the routing path.

CCV9.1 **Command Characteristic Version - Product ID Test**

CCV9.2 This test verifies the PE base service, command characteristic, product ID endpoint of the PE communication specification.
CCV9.3 This test creates a mock byte array with the appropriate header bytes (following the communication specification) to reach the proper endpoint.
CCV9.4 Inputs: There are no inputs for this test. However, the mock byte array message is created internally in order to test the routing path.
CCV9.5 Outputs: To validate proper endpoint routing, a globally-accessible string is modified & verified within this test.
CCV9.6 Normal
CCV9.7 Whitebox
CCV9.8 Functional
CCV9.9 Unit Test
CCV9.10 Results: The globally-accessible endpoint string was modified to the expected result, verifying the routing path.

CCV10.1 **Command Characteristic Version - Product Revision Test**
CCV10.2 This test verifies the PE base service, command characteristic, product revision endpoint of the PE communication specification.
CCV10.3 This test creates a mock byte array with the appropriate header bytes (following the communication specification) to reach the proper endpoint.
CCV10.4 Inputs: There are no inputs for this test. However, the mock byte array message is created internally in order to test the routing path.
CCV10.5 Outputs: To validate proper endpoint routing, a globally-accessible string is modified & verified within this test.
CCV10.6 Normal
CCV10.7 Whitebox
CCV10.8 Functional
CCV10.9 Unit Test
CCV10.10 Results: The globally-accessible endpoint string was modified to the expected result, verifying the routing path.

CCV11.1 **Command Characteristic Version - Device Part Number Test**
CCV11.2 This test verifies the PE base service, command characteristic, device part number endpoint of the PE communication specification.
CCV11.3 This test creates a mock byte array with the appropriate header bytes (following the communication specification) to reach the proper endpoint.
CCV11.4 Inputs: There are no inputs for this test. However, the mock byte array message is created internally in order to test the routing path.
CCV11.5 Outputs: To validate proper endpoint routing, a globally-accessible string is modified & verified within this test.
CCV11.6 Normal
CCV11.7 Whitebox
CCV11.8 Functional
CCV11.9 Unit Test

CCV11.10 Results: The globally-accessible endpoint string was modified to the expected result, verifying the routing path.

CCV12.1 **Command Characteristic Version - Software Part Number Test**
CCV12.2 This test verifies the PE base service, command characteristic, software part number endpoint of the PE communication specification.
CCV12.3 This test creates a mock byte array with the appropriate header bytes (following the communication specification) to reach the proper endpoint.
CCV12.4 Inputs: There are no inputs for this test. However, the mock byte array message is created internally in order to test the routing path.
CCV12.5 Outputs: To validate proper endpoint routing, a globally-accessible string is modified & verified within this test.
CCV12.6 Normal
CCV12.7 Whitebox
CCV12.8 Functional
CCV12.9 Unit Test
CCV12.10 Results: The globally-accessible endpoint string was modified to the expected result, verifying the routing path.

CCV13.1 **Command Characteristic Version - Configuration Test**
CCV13.2 This test verifies the PE base service, command characteristic, configuration endpoint of the PE communication specification.
CCV13.3 This test creates a mock byte array with the appropriate header bytes (following the communication specification) to reach the proper endpoint.
CCV13.4 Inputs: There are no inputs for this test. However, the mock byte array message is created internally in order to test the routing path.
CCV13.5 Outputs: To validate proper endpoint routing, a globally-accessible string is modified & verified within this test.
CCV13.6 Normal
CCV13.7 Whitebox
CCV13.8 Functional
CCV13.9 Unit Test
CCV13.10 Results: The globally-accessible endpoint string was modified to the expected result, verifying the routing path.

CCV14.1 **Command Characteristic Version - Product Serial Number Test**
CCV14.2 This test verifies the PE base service, command characteristic, product serial number endpoint of the PE communication specification.
CCV14.3 This test creates a mock byte array with the appropriate header bytes (following the communication specification) to reach the proper endpoint.
CCV14.4 Inputs: There are no inputs for this test. However, the mock byte array message is created internally in order to test the routing path.
CCV14.5 Outputs: To validate proper endpoint routing, a globally-accessible string is modified & verified within this test.
CCV14.6 Normal

CCV14.7 Whitebox
CCV14.8 Functional
CCV14.9 Unit Test
CCV14.10 Results: The globally-accessible endpoint string was modified to the expected result, verifying the routing path.

CCV15.1 **Command Characteristic Version - Customer Defined Part Number Test**
CCV15.2 This test verifies the PE base service, command characteristic, customer defined part number endpoint of the PE communication specification.
CCV15.3 This test creates a mock byte array with the appropriate header bytes (following the communication specification) to reach the proper endpoint.
CCV15.4 Inputs: There are no inputs for this test. However, the mock byte array message is created internally in order to test the routing path.
CCV15.5 Outputs: To validate proper endpoint routing, a globally-accessible string is modified & verified within this test.
CCV15.6 Normal
CCV15.7 Whitebox
CCV15.8 Functional
CCV15.9 Unit Test
CCV15.10 Results: The globally-accessible endpoint string was modified to the expected result, verifying the routing path.

CCV16.1 **Command Characteristic Version - BLE UUID Test**
CCV16.2 This test verifies the PE base service, command characteristic, BLE UUID endpoint of the PE communication specification.
CCV16.3 This test creates a mock byte array with the appropriate header bytes (following the communication specification) to reach the proper endpoint.
CCV16.4 Inputs: There are no inputs for this test. However, the mock byte array message is created internally in order to test the routing path.
CCV16.5 Outputs: To validate proper endpoint routing, a globally-accessible string is modified & verified within this test.
CCV16.6 Normal
CCV16.7 Whitebox
CCV16.8 Functional
CCV16.9 Unit Test
CCV16.10 Results: The globally-accessible endpoint string was modified to the expected result, verifying the routing path.

CCV17.1 **Command Characteristic Version - CP UUID Test**
CCV17.2 This test verifies the PE base service, command characteristic, CP UUID endpoint of the PE communication specification.
CCV17.3 This test creates a mock byte array with the appropriate header bytes (following the communication specification) to reach the proper endpoint.

CCV17.4 Inputs: There are no inputs for this test. However, the mock byte array message is created internally in order to test the routing path.
CCV17.5 Outputs: To validate proper endpoint routing, a globally-accessible string is modified & verified within this test.
CCV17.6 Normal
CCV17.7 Whitebox
CCV17.8 Functional
CCV17.9 Unit Test
CCV17.10 Results: The globally-accessible endpoint string was modified to the expected result, verifying the routing path.

CCV18.1 **Command Characteristic Version - BLE Company ID Test**
CCV18.2 This test verifies the PE base service, command characteristic, BLE company ID endpoint of the PE communication specification.
CCV18.3 This test creates a mock byte array with the appropriate header bytes (following the communication specification) to reach the proper endpoint.
CCV18.4 Inputs: There are no inputs for this test. However, the mock byte array message is created internally in order to test the routing path.
CCV18.5 Outputs: To validate proper endpoint routing, a globally-accessible string is modified & verified within this test.
CCV18.6 Normal
CCV18.7 Whitebox
CCV18.8 Functional
CCV18.9 Unit Test
CCV18.10 Results: The globally-accessible endpoint string was modified to the expected result, verifying the routing path.

CCV19.1 **Command Characteristic Version - Product Default Name Test**
CCV19.2 This test verifies the PE base service, command characteristic, product default name endpoint of the PE communication specification.
CCV19.3 This test creates a mock byte array with the appropriate header bytes (following the communication specification) to reach the proper endpoint.
CCV19.4 Inputs: There are no inputs for this test. However, the mock byte array message is created internally in order to test the routing path.
CCV19.5 Outputs: To validate proper endpoint routing, a globally-accessible string is modified & verified within this test.
CCV19.6 Normal
CCV19.7 Whitebox
CCV19.8 Functional
CCV19.9 Unit Test
CCV19.10 Results: The globally-accessible endpoint string was modified to the expected result, verifying the routing path.

PS1.1 **Pub/Sub Tests - Subscribe Test**

PS1.2 This test verifies that subscription to topics via the Pub/Sub functions as expected. The PE Pub/Sub enables cross-module communication throughout this project.

PS1.3 This test creates a mock subscriber & topic, registers (or subscribes), and then verifies that the new subscriber has indeed been added to the list of Pub/Sub subscribers.

PS1.4 Inputs: There are no inputs for this test. However, the mock subscriber & topic are created internally in order to properly subscribe.

PS1.5 Outputs: To validate successful subscriptions, the number of registered subscribers is polled & compared.

PS1.6 Normal

PS1.7 Whitebox

PS1.8 Functional

PS1.9 Integration Test

PS1.10 Results: The number of registered Pub/Sub subscribers successfully incremented, verifying the subscription mechanism.

PS2.1 **Pub/Sub Tests - Subscribe Multiple Test**

PS2.2 This test verifies that subscription to multiple topics via the Pub/Sub functions as expected. The PE Pub/Sub enables cross-module communication throughout this project.

PS2.3 This test creates multiple mock subscribers & topics, registers (or subscribes), and then verifies that the new subscribers have indeed been added to the list of Pub/Sub subscribers.

PS2.4 Inputs: There are no inputs for this test. However, the mock subscribers & topics are created internally in order to properly subscribe.

PS2.5 Outputs: To validate successful subscriptions, the number of registered subscribers is polled & compared.

PS2.6 Normal

PS2.7 Whitebox

PS2.8 Functional

PS2.9 Integration Test

PS2.10 Results: The number of registered Pub/Sub subscribers successfully incremented, verifying the subscription mechanism.

PS3.1 **Pub/Sub Tests - Unsubscribe Test**

PS3.2 This test verifies that unsubscribing from topics via the Pub/Sub functions as expected. The PE Pub/Sub enables cross-module communication throughout this project.

PS3.3 This test creates a mock subscriber & topic, registers (or subscribes), then unsubscribes from the same topic, and then verifies that the list of Pub/Sub subscribers is empty.

PS3.4 Inputs: There are no inputs for this test. However, the mock subscriber & topics are created internally in order to properly subscribe & unsubscribe.

PS3.5 Outputs: To validate unsubscribing succeeded, the number of registered subscribers is polled & compared.

PS3.6 Normal

PS3.7 Whitebox

PS3.8 Functional

PS3.9 Integration Test

PS3.10 Results: The number of registered Pub/Sub subscribers successfully returned as 0, verifying the unsubscribing mechanism.

PS4.1 **Pub/Sub Tests - Subscribe Duplicates Test**

PS4.2 This test verifies that subscription to multiple, duplicate topics via the Pub/Sub functions (or rather, rejects) as expected. The PE Pub/Sub enables cross-module communication throughout this project.

PS4.3 This test creates multiple mock subscribers with the same topics, attempts to register (or subscribe), and then verifies that only one instance of the new subscriber has been added to the list of Pub/Sub subscribers.

PS4.4 Inputs: There are no inputs for this test. However, the mock subscribers & topics are created internally in order to properly subscribe.

PS4.5 Outputs: To validate successful subscriptions, the number of registered subscribers is polled & compared.

PS4.6 Abnormal

PS4.7 Whitebox

PS4.8 Functional

PS4.9 Integration Test

PS4.10 Results: The number of registered Pub/Sub subscribers successfully returned 1, verifying the subscription mechanism does not duplicate subscriptions to the same topics.

Test Case Matrix

	Normal/ Abnormal	Blackbox/ Whitebox	Functional/ Performance	Unit/ Integration
CCV1	Normal	Whitebox	Functional	Unit
CCV2	Normal	Whitebox	Functional	Unit
CCV3	Normal	Whitebox	Functional	Unit
CCV4	Normal	Whitebox	Functional	Unit
CCV5	Normal	Whitebox	Functional	Unit
CCV6	Normal	Whitebox	Functional	Unit
CCV7	Normal	Whitebox	Functional	Unit
CCV8	Normal	Whitebox	Functional	Unit
CCV9	Normal	Whitebox	Functional	Unit
CCV10	Normal	Whitebox	Functional	Unit
CCV11	Normal	Whitebox	Functional	Unit
CCV12	Normal	Whitebox	Functional	Unit
CCV13	Normal	Whitebox	Functional	Unit
CCV14	Normal	Whitebox	Functional	Unit
CCV15	Normal	Whitebox	Functional	Unit
CCV16	Normal	Whitebox	Functional	Unit
CCV17	Normal	Whitebox	Functional	Unit
CCV18	Normal	Whitebox	Functional	Unit
CCV19	Normal	Whitebox	Functional	Unit

PS1	Normal	Whitebox	Functional	Integration
PS2	Normal	Whitebox	Functional	Integration
PS3	Normal	Whitebox	Functional	Integration
PS4	Abormal	Whitebox	Functional	Integration