5 Testing

5.1 UNIT TESTING

What units are being tested? How? Tools?

Units:

- **Software Defined Radios (SDR) Software Modules:** Testing of signal processing and modulation algorithms/scripts.
- **Universal Software Radio Peripheral (USRP) Hardware Interface:** Testing for communication between software and USRP hardware.
- **OpenAirInterface (OAI) Configurations:** Validation of custom OAI configurations in controlled environments.
- **Lab Experiment Scripts:** Testing the lab experiment scripts being walked through for accuracy and reliability.
- **GitHub/ReadTheDocs:** Testing the documentation updates pushed to our public GitHub repository trigger rebuilds on ReadTheDocs.

Tools:
GNU Radio, UHD tools, OAI Software Suite, Network Analysis Tools(wireshark), PyTest/Manual Testing of scripts, GitHub webhooks, and ReadTheDocs build framework.

5.2 INTERFACE TESTING

Software-to-Hardware Interface:
Composition: This interface comprises the interaction between the SDR software modules (such as GNU Radio or custom Python scripts) and the USRP hardware.
Testing: Verify that commands sent from the software are correctly executed by the USRP hardware, and that the data received by the hardware is accurately captured and processed by the software.
Tools: UHD Utilities for direct interaction with USRP, automated scripts for functional testing.

Software-to-Software Interface:
Composition: Encompasses the communication and data flow between different software components, such as between GNU Radio and custom scripts or between different modules within OpenAirInterface.
Testing: Ensure data transmission and command execution between software modules, verifying data integrity and correct functionality.
Tools: Custom test scripts, Python for automation, GNU Radio for internal module testing.

User Interface to Backend System Interface:
Composition: Involves the interaction between the graphical user interface (GUI) or command-line interface and the backend software and hardware systems.
Testing: Check that user inputs via the GUI or command-line are accurately translated into actions by the software, and that system status is correctly reflected in the user interface.
Tools: Manual Testing or potentially the use of GUI testing tools like Selenium.

5.3 Integration Testing

SDR Software and USRP Hardware Integration:
Justification: The core functionality of the educational tool depends on effective communication between the SDR software and USRP hardware, and this integration is crucial for accurately conducting experiments and demonstrations in SDR technology.
Testing Method: Tests will involve executing specific signal processing tasks using the SDR software and verifying the correct signal transmission and reception by the USRP hardware.
Tools: GNU Radio for software simulation, UHD Utilities for interacting with USRP hardware, and manual testing or potentially custom scripts for automating testing.

OpenAirInterface (OAI) Integration with USRP:
Justification: For advanced labs involving 5G simulations, the integration between OAI and USRP devices is vital. This allows for realistic demonstrations of 5G network functionalities in the field, within an educational setting.
Testing Method: Setting up 5G network scenarios using OAI and USRP, and testing various network functions such as data transmission, handovers, and signal processing.
Tools: OAI software suite for network setup, network analysis tools like Wireshark for monitoring and troubleshooting.

Integration of Lab Experiment Scripts with SDR/OAI Systems:
Justification: The scripts guide students through the lab experiments and need to interact flawlessly with the SDR and OAI systems to ensure an effective learning experience.
Testing Method: Conducting lab sessions using these scripts, observing their interaction with the SDR and OAI systems, and collecting feedback on their effectiveness and ease of use.
Tools: Python or Bash scripts for controlling SDR and OAI operations, feedback collection tools for assessing user experience.

Documentation Hosting on ReadTheDocs Integrated with GitHub:
Justification: The integration between GitHub and ReadTheDocs is crucial for maintaining up-to-date, accessible, and version-controlled documentation. It's vital for ensuring that students and educators have the latest information and guides for the SDR and OAI based labs and experiments, designed by leveraging ARA's resources.
Testing Method: Verification of the sphinx html build for the document on a localhost/python server, and regular verified updates of documentation pushed to the GitHub repository. Verification of integration between GitHub and ReadTheDocs via the GitHub webhook, to host the documentation on ReadTheDocs.

5.4 System Testing

GROUP 1: SDRs

1. Chris
2. Josh
3. Varun

GROUP 2: OAI

1. Jared
2. Lukas
3. Zach

A complete System Test would have to be performed with Alpha and Beta testing, because what we are creating is a website with learning materials. Given the groups above, these tests would proceed as follows:

Alpha:

- **Group 1 checks Group 2, Group 2 checks Group 1**
- **Make improvements**

Beta:

- **Conduct usability study with 2 uneducated, non-teammates**
- **Make improvements**

5.5 Regression Testing

During the implementation of changes, Alpha testing will be a constant process until everything is implemented.
After the implementation of changes and the next "best" version of the teaching materials has been achieved, impromptu Beta testing will occur to verify the functionality of the labs being as understandable as possible.

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?

Similar to the Regression Testing approach, we will use the people tested in the Beta testing phase and our client/faulty advisor to verify if the labs and wiki pages work as intended. Once it passes Beta testing, our deliverable will be considered "Accepted".

5.7 Security Testing (if applicable)

N/A

5.8 Results

-Overall, for our project, we are limited on what kind of testing methods we can use because we are creating a website with learning materials. We can test how the website is created and how it works and the materials inside the website to make sure the contents that we preview make sense and works using Alpha and Beta Testing. Thus the results of these type of tests are the people that perform alpha and beta tests making sense of the lab and that they are functioning correctly. The website tests are working as intended as well.

- First: we would do **Alpha** testing which would involve users within the project to perform the labs and look through the website to ensure they are making sense of the material. And performing the labs to make sure that they are receiving the correct outcomes and values. As well as proof-reading the website for grammar issues as well.
- Second: we would perform **Beta** testing which would entail users that are not in our senior design group and have no knowledge of 5G. These users would be people that have no knowledge of our deliverable and go through the website to ensure that it makes sense to a new user and that they have the abilities to produce the results that are expected with the lab, and verifying that the instructions are valid and to spot any anomalies.