4 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 adopting test strategy and instruments. In this overarching introduction, given an overview of the testing strategy. Emphasize any unique challenges to testing for your system/design.

4.1 Unit Testing

What units are being tested? How? Tools?

Embedded - We could create some basic test scripts that run through the different range of motor values, and print/log the sensor values as a result. This is not automated as it requires us to physically handle the robot.

Python environment - Render the neural network in the virtual environment and visually inspect that it is behaving as expected. This is the best way to ensure that it is working properly. Python comes with various rendering tools for this purpose.

Neural network agent - We can unit test this by creating a test neural network and applying it in the CPP agent. It should output random actions which would demonstrate that the CPP code is working. Note this does not test the NN, that is done in Python.

4.2 Interface Testing

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

The microcontroller of our robot will have to efficiently send signals to each of the servos and pull movement data from the IMU to make the robot walk in a coordinated manner. So, we will have to test the ability to send/receive signals which can be accomplished in real time using a DMM or oscilloscope. The I2C bus our robot relies on can be tested using an Arduino or a simple I2C testing board (ex. Bus Pirate).

Since the training of the NN is accomplished virtually, the results will be loaded onto the embedded system after training. This could pose a problem since the training environment may not be perfectly reflective of the actual environment the robot is used in. Therefore, we will have to test how well the robot performs in the real environment and make adjustments to the virtual environment accordingly.

4.3 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 integration path between the Raspberry Pi and the microcontroller on the robot will be critical. If the Raspberry Pi is unable to communicate with the robot, our robot will not move at all. We can test this by having control scripts that perform set actions on the robot like moving each leg to certain positions. If we run the script on the Raspberry Pi and the robot moves accordingly, we will know the two units are properly communicating with one another.

There is also a link between our virtually trained model and the Raspberry Pi. The Raspberry Pi must be able to deploy the virtual model in order to tell the robot how to act. The model will be trained on a separate computer. This means the Raspberry Pi must be able to download a saved version of the model and interface with it. This can be tested by comparing the output actions of the model on both the Raspberry Pi and the computer the model was originally trained on. We can have a list of sample input input data, and both models should output the same actions.

4.4 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?

Every time we make a change, we will make assumptions about which systems may be affected and run the corresponding tests from above. Additionally, we will occasionally run all tests to catch unexpected issues.To get into the tests that suffice for system level testing includes pre-train and post-train tests. The main objective of the pre-train test is to identify errors/issues in advance so that we can avoid a wasted training job. On the other hand, the post-train test could be used to interrogate the logic learned while we are training the model and provide us with a behavioral report of the model.

4.5 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 do not break? Is it driven by requirements? Tools?

To ensure that our new functionality additions do not break the old functionality, we will need to run tests after each new addition has been applied. Making multiple applications of new features without testing can cause a lot of headaches trying to figure which feature broke the functionality. Critical features that we need to make sure do not break are the movement of the legs of the robot and the ability to stabilize the robot so it does not fall over. The regression testing would be driven by the requirements of the walking robot.

4.6 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?

For our project, we are conducting a proof of concept of a ML walking robot. In the end, our goal is to demonstrate to Dr. Rover, our client, a successful NN model and a physical robot that can learn how to walk based on said model. These are both items that can be physically demonstrated to Dr. Rover. Another goal of our project is to propose a way to implement machine learning into an undergraduate course. Therefore, it would be beneficial to lay out a development process that could be followed along in a classroom setting. For example, we could present modules that mirror the development process that we followed to reach our final product. These modules will rely on each stage of the development process being completed accurately, emphasizing the importance of other types of testing for our ML robot. We will need our development process and implemented system to be reliable and maintainable in order to be utilized from semester to semester. We should also ensure that our robot performs to the expectations of our client. For example, is the robot able to travel a far enough distance based on the functional requirements we have defined (our target is traveling 3ft within 20 secs).