SUMMARY OF FINDINGS

This document summarizes all the findings from the test plan and execution. This document also includes certain information that might be crucial to calculate the numbers in the Test Plan document

**Host operating system**: macOS BigSur 11.1

**Docker hard disk space**:



A picture containing text, clock, meter, device

Description automatically generated



**Graphical user interface, email, website

Description automatically generated**

**Docker desktop main memory space:**

**Graphical user interface, application

Description automatically generated**

**Docker container main memory space:**

**Graphical user interface, application, table

Description automatically generated**

**Test Strategy Implementations**

**ROS core startup**

**Text

Description automatically generated**

**Ros Topics and Services list**

**Text

Description automatically generated**

**Ros Service call failure:**

**Text

Description automatically generated**

The set target command couldn’t be achieved by using these service methods as they were throwing some error. Hence, I had to alter both the hardware\_node.py and the controller\_node.py to display position and PID command values.

There were other services defined in the methods like reset, next, enable, disable, set target but I couldn’t find them in the services list as seen in the ROS services list picture above. The service methods also didn’t have .srv file in the folder structure inside the /srv/ folder.

A picture showing the tree structure of the folder is given below.

**A screenshot of a computer

Description automatically generated with medium confidence**

Hence, I had to modify the hardware\_node.py and the controller\_node.py to display the position and PID values.

**Roscore service start**

Text

Description automatically generated

**Launching hardware\_node.py**

PID command values

**Graphical user interface

Description automatically generated**

**Launching controller\_node.py**

Current position

**Graphical user interface

Description automatically generated**

**Observed Trends**

|  |  |
| --- | --- |
| **Set Point** | **Saturation value** |
| **0.1** | **0.088** |
| **0.2** | **0.176** |
| **0.3** | **0.264** |
| **0.4** | **0.352** |
| **0.5** | **0.441** |

The position setpoints seem to saturate at these values after few seconds. Every input except the last set point 0.5m seem to pass within the accepted 0.05m range. The last one failed by a small margin. I thought of manipulating the PID gains to make it work but it was beyond the scope of this project.

**Modified Python test code**

I wanted to use the standard unittests.py and nose tests module to test it in a methodical fashion generating user reports. However, services didn’t work properly and debugging took a lot of time. Hence, in the interest of time. I recorded these saturation values by trial and error, and I stopped the execution of the publisher and subscriber nodes as soon as they hit these saturation values. A python code would then feed in input values from 0.1 to 0.5 and then tell them if it passed or failed. A sample is shown below:

A picture containing graphical user interface

Description automatically generated

Given some more time, I was planning on debugging the unit tests and the nose tests module to generate proper reports as required. Also, I would have made the services to work, resulting in a clean modular and efficient code for testing.

Other than these small caveats, I think the code sufficiently addresses all the requirements of the project.