Milestone 4 - Validation Specification

Project: Propbot Autonomous Robot

Authors: Team- 50,
Amr Almoallim (AA) - 83386714,
Apoorv Garg (AG) - 39485545,
Cole Shanks (CS) - 54950860,
Johanan Agarwal (JA) - 29188166,
Sajjad Al-Kazzaz (SA) - 23401565

Affiliation: UBC Radio Science Lab

Table of Contents

1. Summary of Validation

The validation tests are classified under three subsections which are autonomy tests, hardware tests, and integration tests. The following table showcases all the tests and requirements it fulfills.

Requirement Tag	Requirements Description	Test Tag	Status
F1.1	The robot shall be able to accelerate, decelerate and stop when in remote control mode or autonomy mode	HT4.3.10, IT4.5.1	PASS
F1.2	The robot shall be able to move longitudinally when in remote control mode or autonomy mode	HT3.8.1, HT4.3.1, IT4.4.1	PASS
F1.3	The robot shall be able to execute turns	HT4.3.6, IT4.5.1	PASS
F1.4	The robot shall not collide with any object while in motion	HT4.3.9	PASS
F1.5	The robot shall be able to take a set of coordinate waypoints and derive a continuous path that connects those waypoints	AT2.3.1	PASS
F1.6	The robot shall come to a full stop when it reaches a specified waypoint	AT2.3.2	PASS
F1.7	The robot shall have enough battery power to run all core navigation components for 1.5 hours	HT3.3.1	PASS

F1.8	Robot speed shall be configurable in both autonomy and remote control mode	HT4.3.2	PASS
NF1.1	The path generated by the robot shall be the shortest valid path that is within the geofence and ODD.	AT2.3.7	PASS
NF1.2	The robot shall come to a full stop within 0.5m of the brake being triggered	HT4.3.8, HT4.3.9	PASS
NF1.3	The robot shall be able to execute locally generated control signals from the autonomy module	IT4.5.1	PASS
NF1.4	All navigation components must be supplied with power according to their specifications	HT3.4.1, HT3.4.2, HT3.5.1, HT3.5.2	PASS
NF1.5	The minimum robot speed when in motion shall be 1km/hr	HT4.3.3	PASS
NF1.6	The maximum robot speed when in motion shall be 5km/hr	HT4.3.4	PASS
NF1.7	The robot’s movement shall be fluid	HT4.3.5	PASS
F2.1	The robot shall navigate around static objects in front of it when in autonomy mode	AT2.3.3	PASS
F2.2	The robot shall come to a full stop if no suitable path is available	AT2.3.4	PASS

F2.3	The robot shall come to a full stop if an object cannot be avoided	HT4.3.9	PASS
F2.4	In the absence of obstacles, the robot shall move towards its next waypoint when it autonomy mode	AT2.3.1	PASS
F2.5	The robot shall have enough battery power to run all core decision making components (G.2) for 1.5hrs	HT3.3.1	PASS
NF2.1	The robot shall maintain a distance of at least 0.5m from surrounding objects	AT2.3.3, HT4.3.9	PASS
NF2.2	Movement decisions shall be updated at no less than 10Hz	AT2.3.4	PASS
NF2.3	The robot shall be able to take in data from an array of sensors and perform localization	HT3.9.1, HT3.11.1	PASS
NF2.4	All decision components must be supplied with power according to their specifications	HT3.4.1, HT3.4.2, HT3.5.3	PASS
F3.1	The robot shall be operable in autonomy mode and fall-back user mode	IT4.4.2	PASS
F3.2	The fall-back user should be able to manually switch the robot operating mode	IT4.4.2	PASS
F3.3	The robot shall have an onboard switch that kills power to the system	HT3.6.1	PASS

F3.4	The robot shall be enabled with a Physical E-Stop (G.9) and Remote E-Stop (G.11)	HT3.6.1, IT4.4.2	PASS
F3.5	The robot shall send the mission command center a warning whenever it switches over to remote control mode	IT4.4.3	PASS
F3.6	The robot should only operate if an RC transmitter and receiver are paired	HT4.3.12	PASS
F3.7	The robot shall have audio and/or visual warning devices to alert nearby people when moving	HT3.5.4, HT3.7.1	PASS
F3.8	The robot shall detect an incline/decline greater than 2 degrees	HT3.11.1	PASS
NF3.1	Switching between remote and autonomy mode should occur in no more than 0.5s	IT4.4.4	PASS
NF3.2	Physical E-Stop should be in an accessible place in case of emergency	HT3.6.2	PASS
F4.1	The robot shall be teleoperable with an RC transmitter up to a distance of 100m.	HT4.3.11	PASS
F4.2	The robot shall communicate with the mission command center over a cellular network	HT3.12.1	INCOMPLETE (check the test)
F4.3	The robot shall communicate real-time location to the mission command center	HT3.10.1	PASS

F4.4	The user shall be able to upload missions via Mission Command Center	AT2.2.1	PASS
F4.5	The robot shall communicate cellular signal strength data to the mission command center	HT3.12.2	INCOMPLETE (check the test)
F5.1	The robot shall collect time-stamped data at a sampling rate of 1Hz for the location	AT2.3.5, HT3.10.1, IT4.6.1	PASS
F5.2	The robot shall collect time-stamped data at a sampling rate of 1Hz for the motor speed for each motor	IT4.6.2	PASS
F5.3	The robot shall collect time-stamped data at a sampling rate of 1Hz for the command status from both the mission command center and manual control transmitter	AT2.3.6, IT4.6.3	PASS
F5.4	The robot shall collect time-stamped data at a sampling rate of 1Hz for the cellular signal strength	HT3.12.2	INCOMPLETE (check the test)
NF5.1	The time-stamped location data must be accurate within 0.5m	HT3.10.1	PASS

Table 1.1 Requirements and Associated Validation Tests

2. Autonomy Simulation Tests

2.1. Overview of Autonomy Package Simulation Tests

This section describes tests that validate the autonomy package in the Autonomy System. They are significant for our scope because they:

- Ensure that legacy functionality is still intact, even after we make changes to the autonomy package to achieve the requirements of our project.
- Make it easier to integrate the Autonomy System with the Vehicle Interface System
- Give us experience with and knowledge of the details of the autonomy system

2.2. General Setup Procedures and Resources

The legacy system [1] includes a Gazebo-based simulation package that allows the autonomy package to be validated. Gazebo is a simulator that is built for ROS projects and allows users to configure simulated sensor data and model vehicles. There are three components to the autonomy package simulation tests (figure 2.1):

Mission Command Centre (MCC): GUI which can be used by the tester to upload missions. In our current setup, it is run on the tester's personal computer.

Simulation Package: Package created through Gazebo and visualized through the RVIZ ROS package. When launched, the Gazebo simulator simulates LiDAR, IMU, Camera, and GPS data based on the environment launched with it. For our tests, we use the environment in figure 2.2. It shows a Propbot model on a grid. The grid represents all navigable locations for which the simulation data is active. Around Propbot are a number of different objects. Propbot starts in a stationary position, facing a concrete curb. Gazebo then receives wheel speeds from the Autonomy Package and applies them to the Propbot model robot in the simulation. Based on Propbot's new position in the simulated environment, published simulated data is updated. RVIZ listens on the ROS network and populates a Graphical User Interface (GUI) with all data and parameters (ex. LiDAR point clouds, global path plan, current trajectory, etc) that are being published. The GUI can then be viewed by the tester (figure 2.3).

Autonomy Package: This is the package under test. It receives simulated sensor data from the simulation package and outputs wheel speeds, as it would with real-world sensor data. It is run on the Autonomy Computer (currently an NVIDIA Jetson NX), which also runs the ROScore package (responsible for being the Centre of the ROS network and connecting nodes to topics), as it would in real-life use of Propbot.

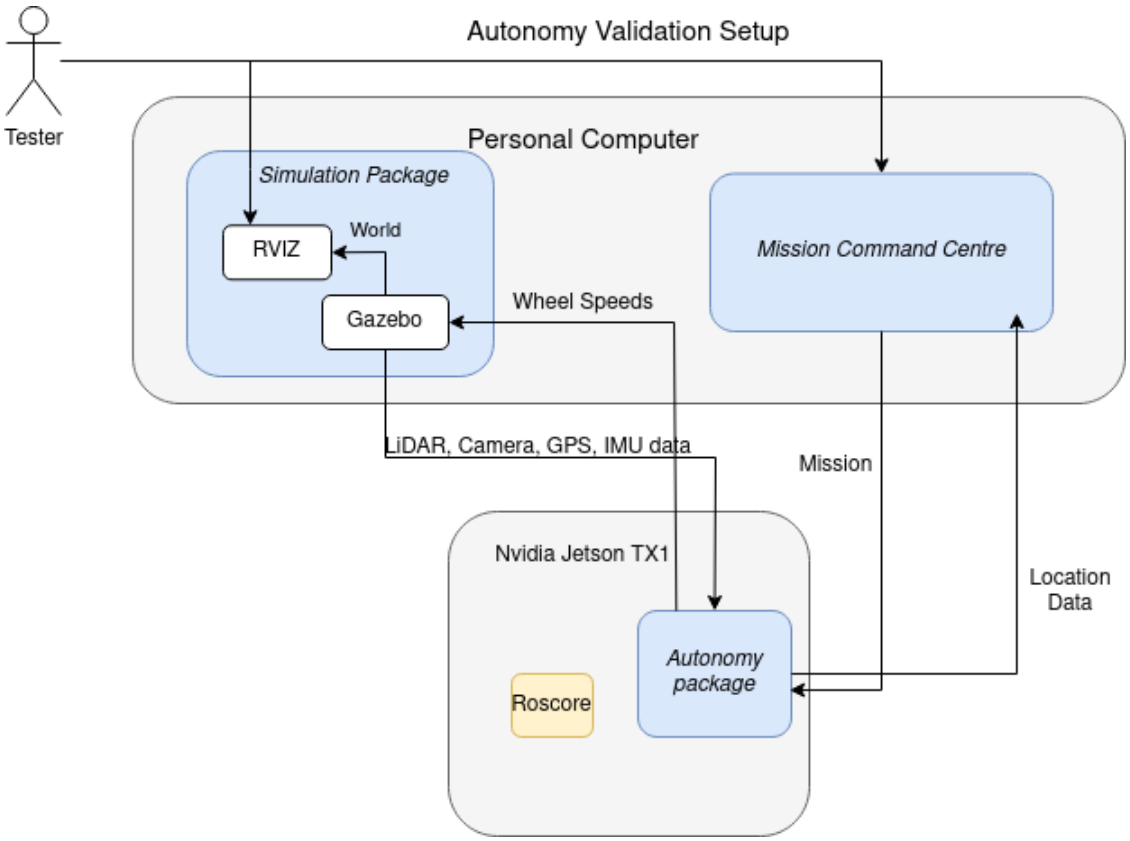

Figure 2.1: Current Autonomy Validation Setup

The general strategy used in evaluation metrics is to inspect ROS “topics” related to data under test. A ROS topic is a bus over which nodes exchange messages. A node in ROS is a process that performs a certain computation. Nodes can publish messages to a topic, or subscribe to messages from a topic. Users connected to the ROS network also have the ability to listen to these topics, allowing us to verify that the correct messages are being sent. For example, figure A.3 shows an example output of querying the topic “/velodyne/points”, which shows LiDAR sensor data.

To verify that Propbot is behaving correctly, we use the RVIZ visualizer to inspect Propbot's motion and obstacle avoidance. An example of what RVIZ looks like on a mission is shown in figure 2.3. In the figure, the model Propbot can be seen moving around an obstacle (the cement curb from figure 2.2). Propbot's target waypoint is shown through the green line, whereas the red line shows Propbot's plan to reach the target waypoint without colliding with obstacles.

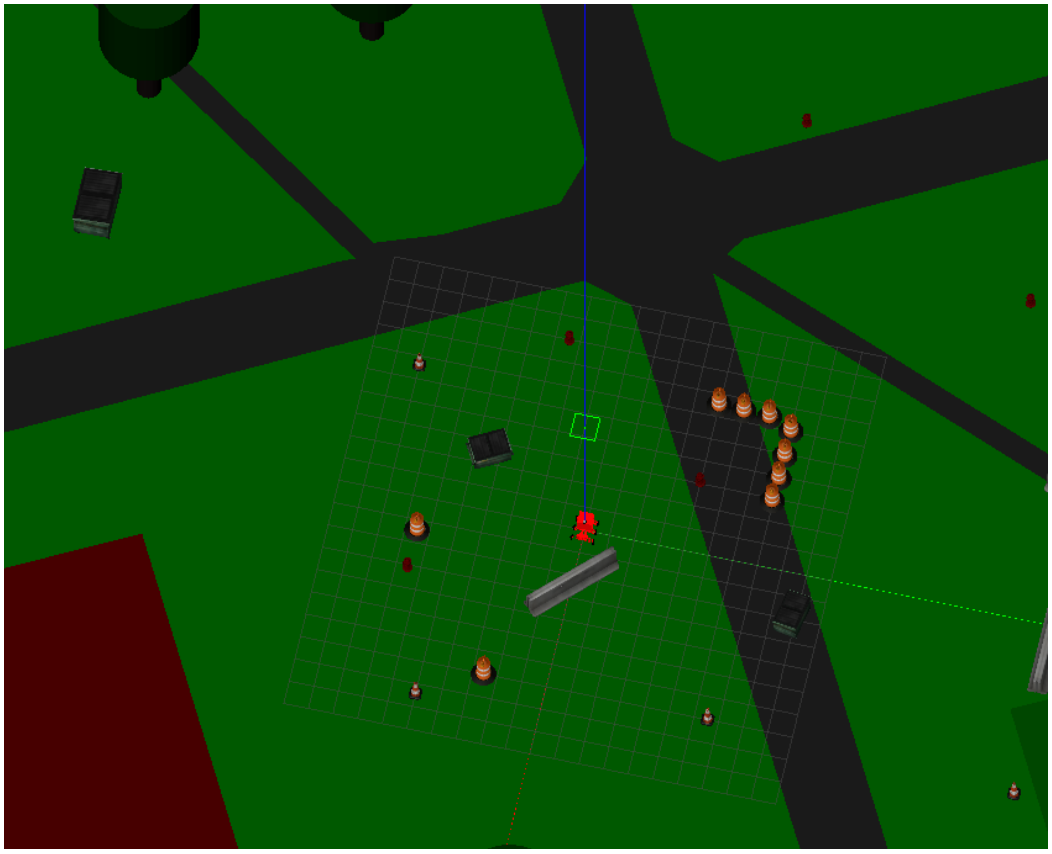

Figure 2.2: Gazebo simulation environment

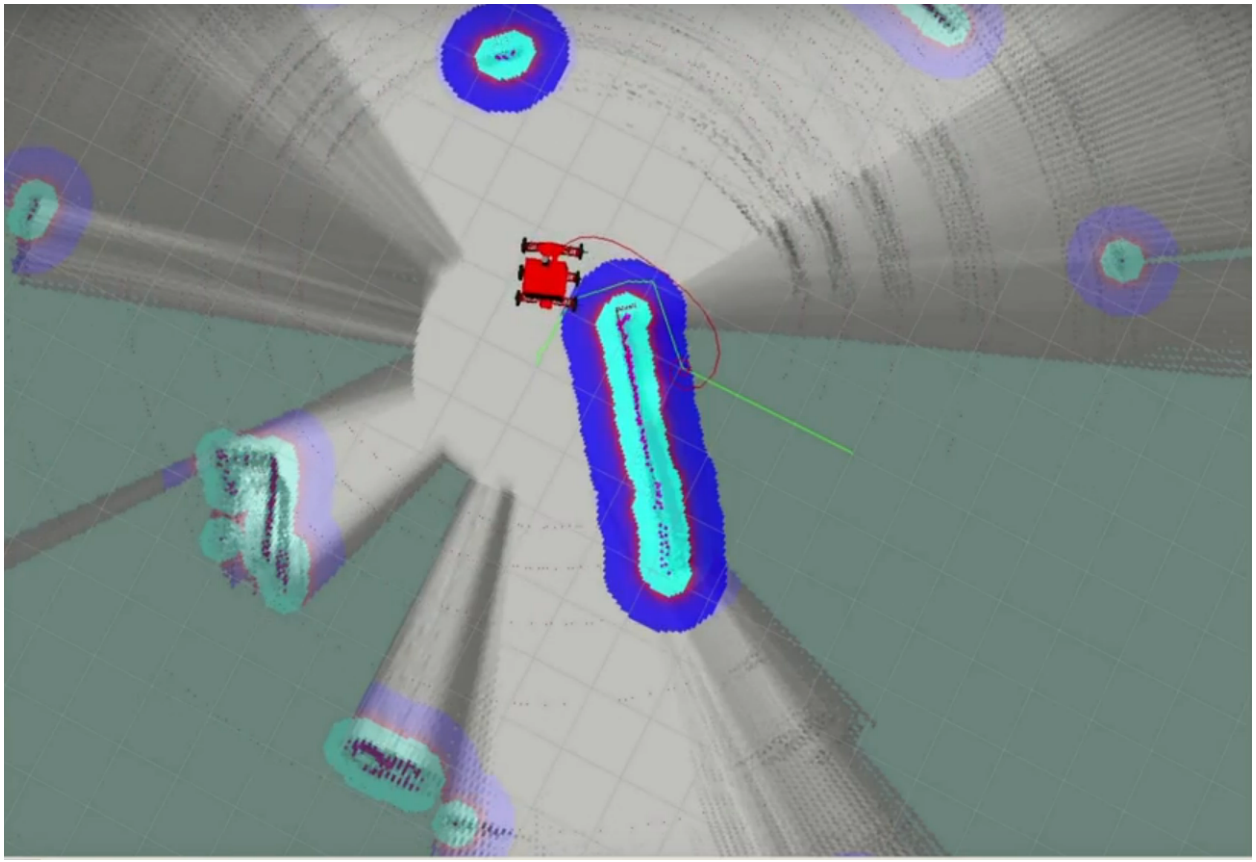

Figure 2.3: RVIZ example showing PropBot in motion around obstacles

2.3. Gazebo Simulation

2.3.1. Target

In order to validate the autonomy package and the mission command centre via simulation, we validated whether the Gazebo Simulations themselves were working. The Gazebo simulations are responsible for simulating LiDAR, GPS, IMU, and camera data.

2.3.2. Evaluation Metrics

Since this subsystem is a test tool and on its own does not account for a requirement of Propbot, we give descriptions for pass criteria based on the requirements of this subsystem.

Test Tag	Requirement Tag	Requirement Description	Pass Criteria
AT 2.1.1	NA	LiDAR data must be published	Data output in /velodyne/points topic (as in A.3) RVIZ simulation shows obstacles and point clouds
AT 2.1.2	NA	IMU data must be published	Data output in /imu/data topic shows: Angular velocity x = 0 Angular velocity y = 0 Linear Acceleration x = 0 Linear Acceleration y = 0 This is because at the start of simulation the robot is not moving
AT 2.1.3	NA	Camera data must be published	Data output in /camera/depth/points topic RVIZ simulation shows obstacle heat maps
AT 2.1.4	NA	GPS data must be published	Data output in /map/fix topic shows Latitude = 49.2676479 Longitude = -123.25107 These are the starting positions of the simulation

Table 2.1: Gazebo Simulation Evaluation Metrics

2.3.3. Procedure

The following steps demonstrate how to conduct the Gazebo Simulation test.

1) Connect the simulation, vehicle autonomy and the mission command centre computer to the same local network. Do a simple ping test to verify bi-directional communication between each computer
2) On the simulation computer, run:
    a) export ROS_MASTER_URI=http://Propbot@vehicle_autonomy_computer:11311
3) On the vehicle autonomy computer, run:
    a) export ROS_MASTER_URI=http://Propbot@vehicle_autonomy_computer:11311
4) On the simulation computer, run: roslaunch Propbot_simulation ubc_student_union_sim.launch
5) On the vehicle autonomy computer, run:

    a) roslaunch Propbot_autonomy autonomy.launch
6) Ensure that the robot model appears in the simulation
7) On the simulation computer, run:
    a) rostopic echo /velodyne/points
8) From the output, check that the robot publishes its velodyne data
9) On the RVIZ tab (which should've popped up after step 2) check that the robot has built a representation of surrounding objects, based on figure 2.2. An example of a good representation is in figure 2.3. Take particular notice of the cement block in front of it and the L shaped group of cones 120 degrees from PropBot
10) If 8 and 9 pass, then AT 2.1.1 passes
11) On the simulation computer, run:
    a) rostopic echo /imu/data
12) From the output, check that the robot publishes IMU data (figure A.2) according to the evaluation criteria of AT 2.1.2. If it does, AT2.1.2 passes
13) On the simulation computer, run:
    a) rostopic echo /camera/depth/points
14) From the output, check that the robot publishes camera data. If it does, and step 10 was satisfied, AT2.1.3 passes
15) On the simulation computer, run:
    a) rostopic echo /map/fix
16) From the output, check that the robot publishes map data according to the evaluation criteria of AT2.1.4. If it does, AT2.1.4 passes

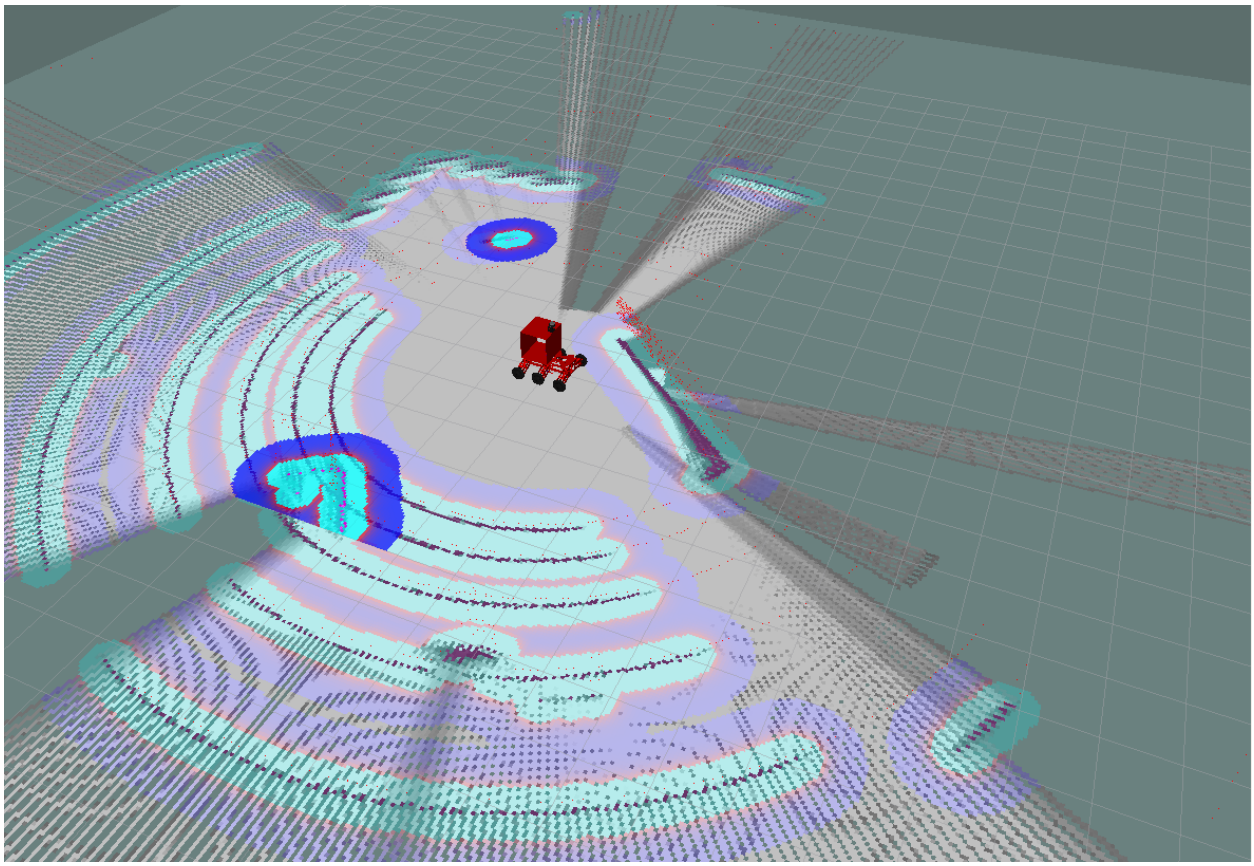

Figure 2.4: RVIZ at starting position

2.3.4. Test Results

The success of these tests (table 2.2) implies that our simulation setup is working as intended and that we can move forward with validating the Autonomy Package.

Test Tag	**Pass**	**Notes**
AT2.1.1	Yes	Velodyne data is output by the simulator and visible on the visualizer.
AT2.1.2	Yes	IMU data is output by the simulator and its values show Propbot is at initialized state.
AT2.1.3	Yes	Camera data is output by the simulator and visible
AT2.1.4	Yes	Map data is output by the simulator and corresponds to the expected starting point

Table 2.2 Gazebo Simulation Results

2.4. Mission Command Centre

2.4.1. Target

Validates mission command centre's ability to load the right plugins and upload missions.

2.4.2. Evaluation Metrics

Test Tag	Requirement Tag	Requirement Description	Pass Criteria
AT2.2.1	F4.4	The user shall be able to upload missions via the mission command Centre	"Uploaded mission" message given in mission command Centre The topic /mapviz/mission_command shows "Uploaded Mission" The topic /mapviz/mission shows details of the mission uploaded (based on the user's input to MCC)

Table 2.3: Mission Command Centre Requirement and Pass Criteria

2.4.3. Procedure

1) Connect the simulation, vehicle autonomy and the mission command centre computer to the same local network. Do a simple ping test to verify bi-directional communication between each computer
2) On the simulation computer, run: export ROS_MASTER_URI=http://Propbot@vehicle_autonomy_computer:11311
3) On the vehicle autonomy computer, run: export ROS_MASTER_URI=http://Propbot@vehicle_autonomy_computer:11311
4) On the simulation computer, run: roslaunch Propbot_simulation ubc_student_union_sim.launch
5) On the vehicle autonomy computer, run: roslaunch Propbot_autonomy autonomy.launch
6) Ensure that a mission command Centre appears, where a map of UBC loads by default
7) On the left side panel click "add" and select "plan_mission"
8) Ensure that a menu item called "plan_mission" appears (figure A.3)
9) On the map, right-click on any 6 points on the map
10) Ensure that a blue waypoint is added to the map after each click
11) On the menu "plan_mission" menu item, click "upload mission"
12) Evaluate the pass criteria for AT 2.2.1, if all evaluation criteria pass, AT 2.2.1 passes

2.4.4. Test Results

Test Tag	Pass	Notes
AT2.2.1	Yes	MCC is very responsive when adding waypoints. A comment is displayed showing a successful mission upload. The contents of the mission are also seen on the ROS network.

Table 2.4: Mission Command Centre Result

The success of this test implies that the MCC is working as intended and that we can move forward with validating the Autonomy Package.

2.5. Autonomy Package - Missions

2.5.1. Target

To test the basic ability of the autonomy package to conduct a multi-waypoint mission in a static environment, where objects need to be avoided.

2.5.2. Evaluation Metrics

Test Tag	Requirement Tag	Requirement Description	Pass Criteria
AT2.3.1	F1.5, F2.4	The robot shall be able to take a set of coordinate waypoints and derive a continuous path that connects those waypoints	Robot follows the generated path from the given waypoints
AT2.3.2	F1.6	The robot shall come to a full stop when it reaches a specified waypoint	At each waypoint, the robot stops within 2m from target location
AT2.3.3	F2.1, NF2.1	The robot shall navigate around static objects in front of it when in autonomy mode The robot shall maintain a distance of at least 0.5m from surrounding objects	Robot must complete the mission without entering an objects fence
AT2.3.4	NF2.2	Movement decisions shall be updated at no less than 10Hz	Movement data is published by the autonomy computer via the rostopic "/geometry_msgs/TwistStamped"

AT2.3.5	F5.1	The robot shall collect time-stamped data at a sampling rate of 1Hz for the location	Time-stamped position data is published to the rostopic "/navsat/fix ROS topic in wgs84 format every second
AT2.3.6	F5.3	The robot shall collect time-stamped data at a sampling rate of 1Hz for the command status the mission command center	The rostopic /cmd_vel is published at frequency greater than 1 Hz
AT2.3.7	NF1.1	The path generated by the robot shall be the shortest valid path that is within the geofence and ODD	Visual inspection of path that robot takes is the shortest path

Table 2.5 Autonomy Package Requirement and Pass Criteria

2.5.3. Procedure

The following steps demonstrate how to conduct the multiple waypoint mission test. The setup required is described in section 1.2 (Autonomy Package Simulation Tests). We run the following steps on a number of scenarios, described in table 2.6. The tests only pass if they pass in every scenario.

Scenario Tag	**Validated Requirement Tag(s)**	**Test Scenario**
S1	NF1.3, F1.1, F1.3	Upload mission with 1 waypoint at a reachable position
S2	F1.1 F1.3, F1.4, F2.1	Upload mission with 1 waypoint at a reachable position, with an obstacle between Propbot and the waypoint
S3	NF2.3, F1.6, F2.4	Upload mission with 2 waypoints at reachable positions. When Propbot reaches the first waypoint, send the “resume mission” command.
S4	F2.2	Upload mission with 1 waypoint at a location outside of map

S5	F2.2, F2.3	(Propbot surrounded by obstacles from all areas) Upload mission with 1 waypoint in the unreachable area (figure 2.5)

Table 2.6 Autonomy Package Scenarios

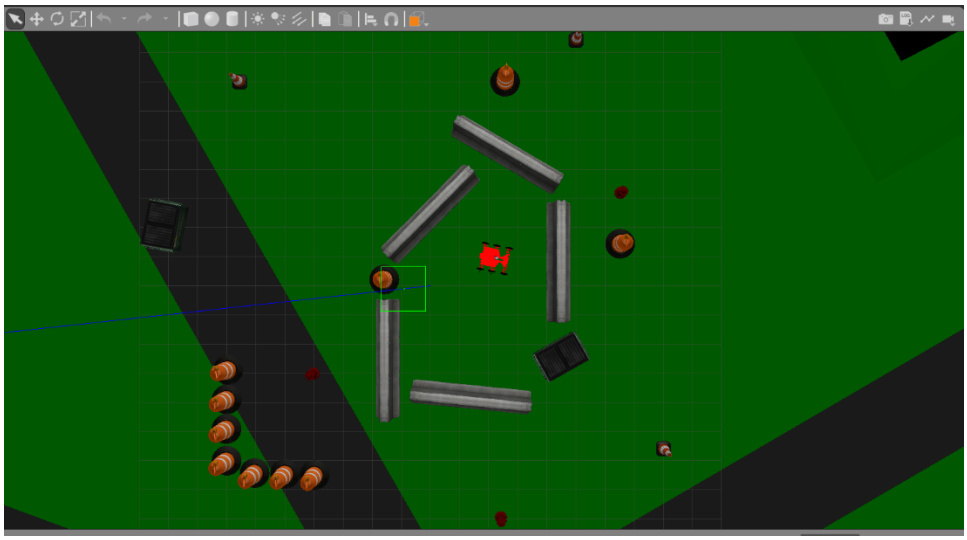

Figure 2.5: Gazebo environment for S5

1. Connect the simulation, vehicle autonomy and the mission command centre computer to the same local network. Do a simple ping test to verify bi-directional communication between each computer
2. On the simulation computer, run: export ROS_MASTER_URI=http://Propbot@vehicle_autonomy_computer:11311
3. On the vehicle autonomy computer, run: export ROS_MASTER_URI=http://Propbot@vehicle_autonomy_computer:11311
4. On the mission command centre computer, run: export ROS_MASTER_URI=http://Propbot@vehicle_autonomy_computer:11311
5. On the simulation computer, run: roslaunch Propbot_simulation ubc_student_union_sim.launch
6. On the vehicle autonomy computer, run: roslaunch Propbot_autonomy autonomy.launch
7. On the mission command centre computer, run: roslaunch Propbot_mission_gui mapviz.launch
8. Ensure that the robot model appears in the mission command centre GUI at the same geo-graphic coordinate as the robot model in the simulation
9. Setup the desired scenario on the mission command Centre GUI and click the Upload Mission button

10. Click the Start Mission button on the mission command Centre GUI
11. Conduct the test as described in the test scenario
12. On the vehicle autonomy computer, run: rostopic hz /navsat/fix
13. Check the average output frequency rate. If it is 1Hz or greater, AT2.3.5 passes
14. On the vehicle autonomy computer, run: rostopic hz /geometry_msgs/TwistStamped
15. Check the average output frequency rate. If it is 10Hz or greater, AT2.3.4 passes
16. When the robot model reaches the first mission waypoint, run: rostopic echo /navsat/fix
17. If the robot GPS coordinate is within 2m of the target coordinate, AT2.3.2 passes
18. If the path the robot took was the shortest path, LAT 2.3.7
19. End trial when:
    a. Mission is complete. (AT2.3.1, AT2.3.3 pass)
    b. Robot model collides with an obstacle

2.5.4. Test Results

The results of these tests show that the Autonomy Package is working as intended. For S4 and S5, we note that Propbot also sent warning messages to the mission command centre, informing the user that the mission has failed. For S4, the failure message was propagated as soon as the waypoint coordinates were input. For S5, Propbot took 12 minutes to determine that a suitable path was not available. In those 12 minutes, Propbot rotated around its axis five times and moved around (without collisions) to find possible openings. This fall-back behaviour is expected and can be further configured in the Autonomy Package.

Test Tag	**Pass**	**Notes**
AT2.3.1	Yes	Propbot immediately finds a path to the intended target and starts moving towards it.
AT2.3.2	Yes	A log is also put out saying the mission is complete
AT2.3.3	Yes	Robot did not enter any objects fences
AT2.3.4	Yes	Tester needs to make sure the command is issued when Propbot has set a path to go on. If the command is issued when Propbot is stationary and still trying to build a path, then the result will be smaller.
AT2.3.5	Yes	This test does not completely validate this requirement, it only checks that the data is published at 1hz, not that it is collected. For the data collection test, see Section 4.6.
AT2.3.6	Yes	This test does not completely validate this requirement, it only checks that the data is published at 1hz, not that it is collected. For the data collection test, see Section 4.6.
AT2.3.7	Yes	1. Visual inspection is not the best way to judge shortest path, however we are interested in a general notion of

		shortest path, meaning we want to ensure Propbot is not making clear and obvious miscalculations (for example, not going around an object from the nearest side, having a loop in the path..) 2. This test is invalid for S4 and S5

Table 2.7 Autonomy Package Results

3. Hardware Tests

3.1. Overview of Hardware Tests

To fully validate Propbot as a system it is necessary to first test and verify the various subsystems that make up Propbot (figure 3.1). This section details tests done to verify power delivery and capacity needed to operate the autonomy system and the vehicle interface system. Tests were also done to verify various key components related to motion in the vehicle interface system as well as sensor components in the autonomy system. Lastly, this section tests the safety and risk mitigation components intended to ensure the safety of users, the public, and Propbot itself.

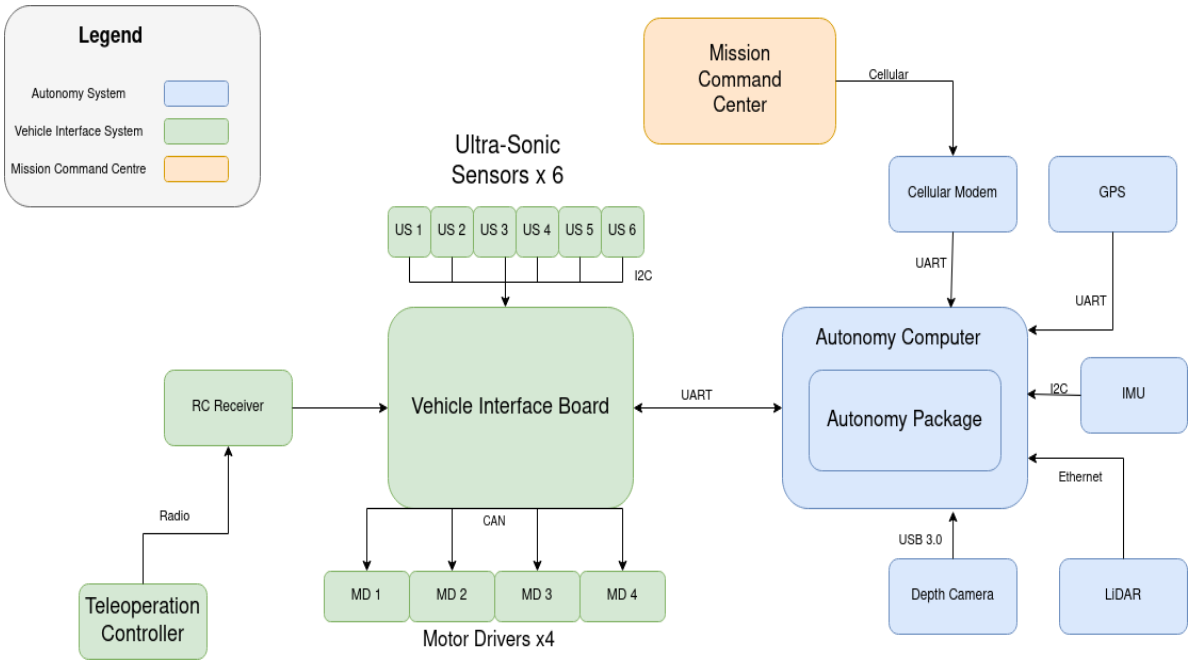

Figure 3.1 System Overview

3.2. General Setup Procedures and Resources

3.2.1. Connecting Components

Connecting components is a necessary action that is needed to test and use electrically powered items. The process can vary slightly depending on the terminals present on different power sources and power sinks. The battery used on Propbot and for associated tests is the downtube 36V 26.5Ah battery from E-bikes [4]. It is connected to the necessary components using Anderson power pole connectors, terminal blocks with screw-in terminals, and bullet connectors.

To connect the battery to the main terminal distribution block, an Anderson power pole connector is connected to two cables that are in turn screwed into the terminal block (figure 3.2).

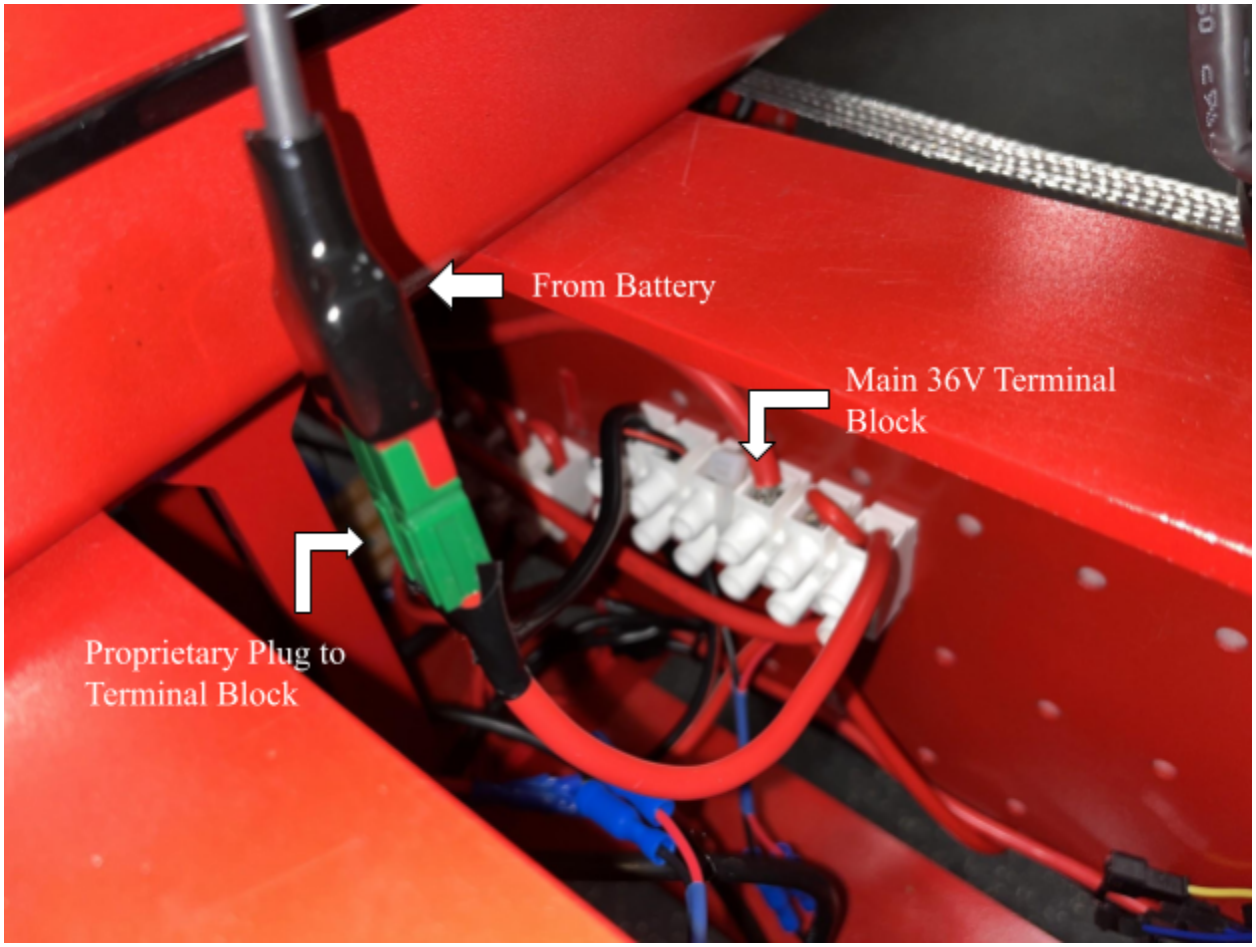

Figure 3.2 Connecting the Battery to the Main Terminal Distribution

To connect the buck converters to the terminal block two cables are used per buck converter. Each cable has a male bullet connector crimped onto one end, with the other end being screwed into the terminal block associated with its colour and voltage level (red = VCC, black = ground) (figure 3.3). Each of the input terminals on the buck converters has female bullet connectors crimped onto them for easy connect/disconnect to the male connectors (Note: care should be taken to match the cables for the right colour/voltage).

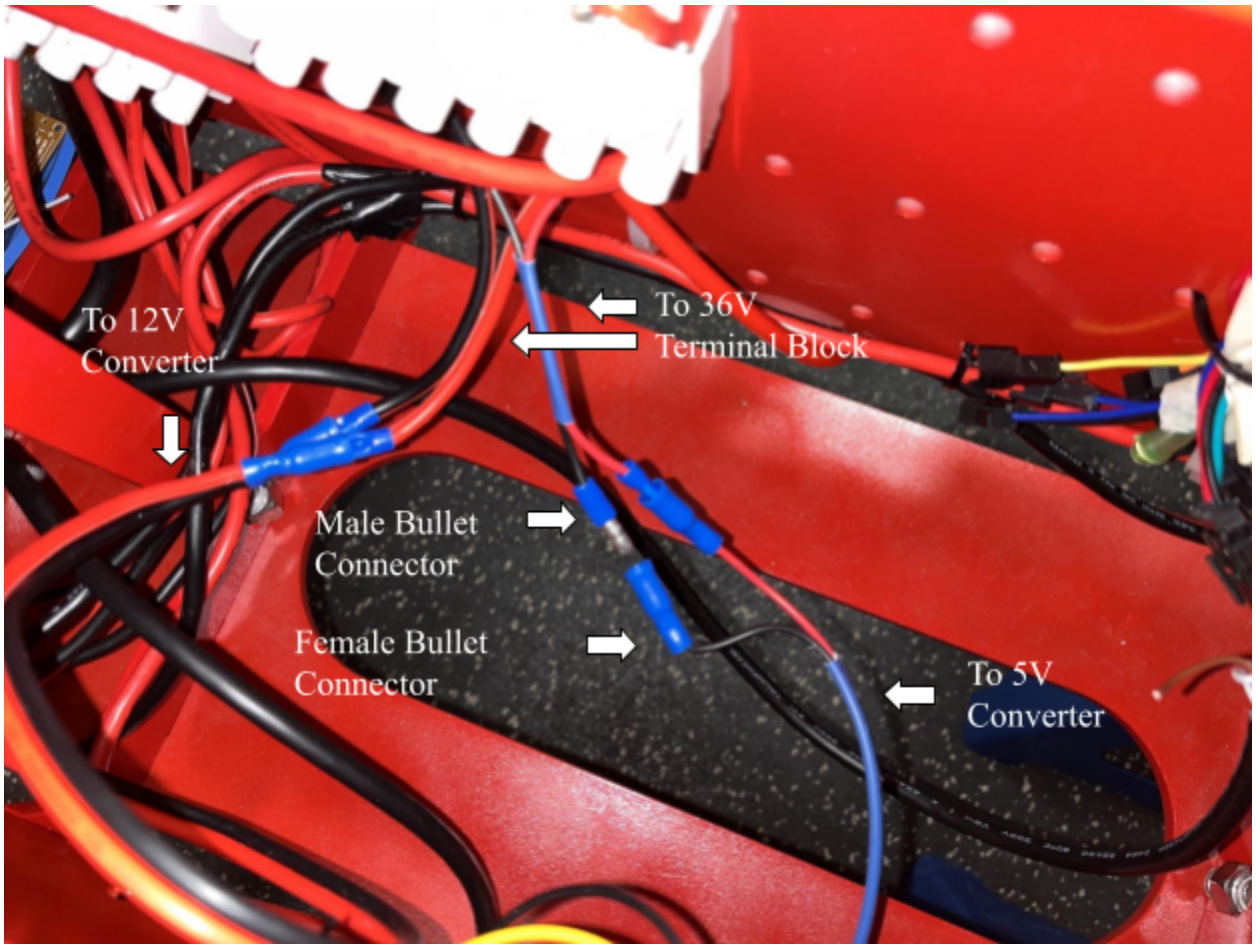

Figure 3.3 Bullet Connectors to the Converters

To connect the low voltage components to the buck converter another terminal distribution block is used. The output terminals from the buck converter are screwed into the terminal block. Similarly, the connecting cables to the Arduino, Jetson and light beacon are also screwed into the terminal block following the colour/voltage required (figure 3.4).

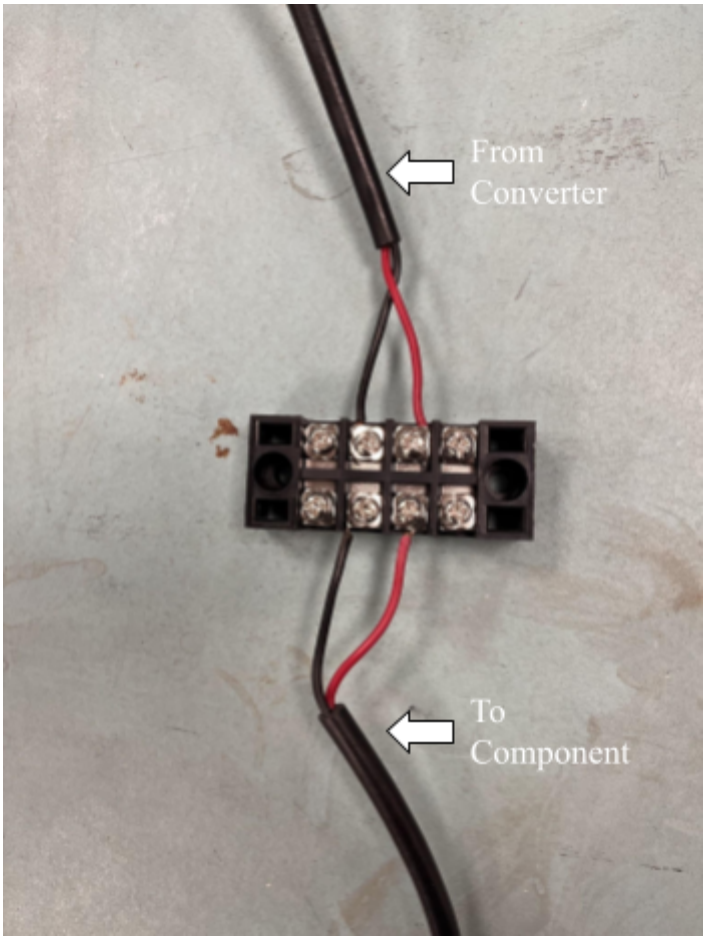

Figure 3.4 Screw in Terminal for Low-Voltage

3.2.2. Turning Components “ON” and “OFF”

Some components need to be physically turned “ON” to start working. This section is to explain which components have this behaviour and how to turn them “ON”. Unless otherwise stated in this section a component will be turned “ON” when power is supplied to it and “OFF” when no power is supplied.

The downtube battery has a power switch next to the charging port as shown in figure 3.5. When the “O” is down that means the battery is “OFF” and when the “1” is down that means the battery is “ON”.

Figure 3.5 Power Switch on Downtube Battery

In order to turn on the motor drivers, it is necessary to turn on the physical E-stop. The E-stop is in series with only the motor drivers to give users a quick way to cut power and stop motion in case of an emergency without cutting power to the rest of the system. The “ON” position is when the mushroom button is up, as shown in figure 3.6. To turn “OFF” (figure 3.7) power one must simply push down on the button. To turn the E-stop back “ON” the mushroom button must be twisted clockwise as shown by the engraving on the top of the button.

Figure 3.6 E-Stop in “ON” Position

Figure 3.7 E-Stop in “OFF” Position

3.2.3. Jacking up Propbot

Many of the tests are done when Propbot is elevated on jack stands (figure 3.8). The process of jacking Propbot requires using a hydraulic jack which requires proper training and can be a serious safety hazard if not operated correctly. The main risks include unstable loads due to improper use of/insufficient stands and pinch points when lowering the load. These risks can be mitigated by choosing a sturdy location to jack from, using sufficient stands to balance the load, and making sure to keep fingers and hands away from the jacking area while the load is being lowered.

1. Place the jack under one end of Propbot and start elevating that side
2. Put a jack stand in place
3. Place a jack stand near the section elevated by the hydraulic lift jack
4. Remove jack and repeat steps 2-3 on the other side of Propbot

Figure 3.8 Propbot Jacked Up

3.3. Battery

3.3.1. Target

This test is to confirm the downtube batteries' capacity to power the vehicle interface system and the autonomy system for the required 1.5 hours.

3.3.2. Evaluation Metrics

Test Tag	Requirement Tag	Requirement Description	Pass Criteria
HT 3.3.1	F1.7, F2.5	The robot shall have enough battery power to run all core navigation components and decision-making components for 1.5 hours	Propbot runs for 1.5 hours in a situation comparable to intended use

Table 3.1 Propbot Battery Capacity Evaluation Metrics

3.3.3. Safety Considerations

Propbot is a large and heavy robot that can seriously harm team members, pedestrians, and infrastructure. Propbot must be operated in an open and controlled environment with the necessary safety mechanisms in place and functional. This means a geofenced area that has no people with physical and remote E-stop tested and fully functional.

3.3.4. Procedure

Note, a more accurate and complete way of testing the capacity of the battery would be to use a power monitor or multimeter in series with the battery that could read and record current over an extended use of measured time and then using $dC = I\,dt$. However, since Propbot is capable of drawing in the 10s of amps and that power monitors and multimeters that can handle such currents are beyond the budget and time capabilities of the team, this alternative test was devised to verify the battery's ability to fulfill the stated requirements. Also note that although the battery capacity test is listed before the remaining component and system integration tests, it was one of the last tests done since it requires running the entire system to simulate intended use cases. It is placed before the other tests due to the other components' reliance on battery power and for organizational purposes.

1. Take Propbot to a fenced-off area that is no less than 15m by 15m that has no obstacles or pedestrians but includes an inclined plane
2. Turn on Propbot including all associated navigation and decision-making components (including Jetson and available sensors even though they will not be operating Propbot, they will be running the autonomy package to simulate autonomous operation)
3. Use the RC transmitter to operate Propbot and perform various motions including forward, backward, rotational turning and braking on both flat and inclined planes. The goal is to create a situation like real data collection scenarios.
4. Repeat this operation for a timed interval of 1.5 hours at least
5. Turn off the battery and record the built-in battery capacity meter’s level (figure 3.9)
6. Recharge the battery to full and repeat steps 1-4 a minimum of three times

Figure 3.9 Power Level Indicator on Downtube Battery

3.3.5. Test Results

The battery was successfully able to power Propbot for the minimum required 1.5 hours. The table below shows the results of three tests done. Note that this situation was not a final test for two reasons:

1. Propbot had multiple missing or differing components from the final design including Velodyne puck LiDAR, Roboteq motor controllers, self-levelling platform and all measurement equipment and devices needed by researchers
2. Propbot was designed for a multi-battery setup including three 36V 26.5Ah batteries (Design Document 7.3), and the test was run using only one of these recommended batteries

Test Tags and Number	**Pass**	**Notes**
HT3.3.1 - 1	Yes	The battery-powered all components for 2 hours and 24 minutes with the built-in capacity indicator reading 2/4 bars
HT3.3.1 - 2	Yes	The battery-powered all components for 1 hour and 46 minutes with the built-in capacity indicator reading 3/4 bars
HT3.3.1 - 3	Yes	The battery-powered all components for 3 hours and 9 minutes with the built-in capacity indicator reading 2/4 bars

Table 3.2 Propbot Battery Capacity Results

Therefore, given the results showing over 50% (2/4 bars) capacity while running Propbot for longer than the stated requirement we can confidently conclude that if the extra components were added onto Propbot the 1.5-hour requirement would be met with the single chosen battery or at most one additional battery out of the initial proposed three battery setup. This can be concluded because by far the largest source of power consumption comes from the motors (Design Document 7.3), and our tests accurately accounted for this consumption and passed.

3.4. Power Conversion

3.4.1. Target

This test is to validate that the buck converters can output a steady voltage as per their specifications and to measure the efficiency at which they perform.

3.4.2. Evaluation Metrics

Test Tag	Requirement Tag	Requirement Description	Pass Criteria
HT3.4.1	NF1.4, NF2.4	All components must be supplied with power according to their specifications	The 12V converter takes in the rated 36-48V and outputs 12V
HT3.4.2	NF1.4, NF2.4	All components must be supplied with power according to their specifications	The 5V converter takes in the rated 12-48V and outputs 5V
HT3.4.3	NA	Verify high-efficiency rating	The 12V converter has a power delivery efficiency greater than 85%
HT3.4.4	NA	Verify high-efficiency rating	The 5V converter has a power delivery efficiency greater than 85%

Table 3.3 Power Conversion Evaluation Metrics

3.4.3. Safety Considerations

Dealing with high voltage and current electricity can result in electrocution, burns, or even blinding. Safety procedures and mitigations must be followed.

3.4.4. Procedure

HT3.4.1:

1. Take the downtube battery and flip the switch to turn it on
2. Using a multimeter, read the output voltage of the battery and verify it is in the 36-48V range, then turn off the battery
3. Connect the battery's black and red cables to the converter's black and red cables labelled "input" (figure 3.10)
4. Turn on the battery and verify, using a multimeter, that the voltage being output by the converter on the yellow and black wires labelled "output" is 12V, then turn off the battery

Figure 3.10 12V Buck Converter Wiring

HT3.4.2:

1. Take the downtube battery and flip the switch to turn it on
2. Using a multimeter read the output voltage of the battery and verify it is in the 12-48V range, then turn off the battery
3. Connect the battery's black and red cables to the converter's black and red cables labelled "in" (figure 3.11)
4. Turn on the battery and verify, using a multimeter, that the voltage being output by the converter on the yellow and black wires labelled "out" is 5V, then turn off the battery

Figure 3.11 5V Buck Converter Wiring

HT3.4.3:

1. Take the downtube battery and flip the switch to turn it on.
2. Using a multimeter record the output voltage of the battery
3. Connect a device rated for 12V input to the yellow and black wires labelled "output" on the buck converter
4. Connect the battery's red cable to the red cable labelled "input" on the buck converter
5. Connect the battery's black cable to the black cable labelled "input" on the buck converter but with a multimeter in series
6. Turn on the battery and record the current being drawn
7. Turn off the battery, remove the multimeter and connect the battery straight to the input of the converter.
8. Turn on the battery and using a multimeter record the voltage across the output of the buck converter
9. Turn off the battery and connect the multimeter in series with the black wire on the multimeter labelled "output" and the Arduino
10. Turn on the battery and record the current being drawn
11. Turn off the battery and disconnect all the components

HT3.4.4:

1. Take the downtube battery and flip the switch to turn it on
2. Using a multimeter, record the output voltage of the battery
3. Connect a device rated for 5V input to the yellow and black wires labelled "out" on the buck converter
4. Connect the battery's red cable to the red cable labelled "in" on the buck converter
5. Connect the battery's black cable to the black cable labelled "input" on the buck converter but with a multimeter in series
6. Turn on the battery and record the current being drawn
7. Turn off the battery, remove the multimeter, and connect the battery straight to the input of the converter
8. Turn on the battery and using a multimeter record the voltage across the output of the buck converter
9. Turn off the battery and connect the multimeter in series with the black wire on the multimeter labelled "out" and the Arduino
10. Turn on the battery and record the current being drawn
11. Turn off the battery and disconnect all the components

3.4.5. Test Results

Test Tags	Pass	Notes
HT3.4.1	Yes	The battery output voltage was 40.1V. The converter output voltage was 12.08V.
HT3.4.2	Yes	The battery output voltage was 40.1V. The converter output voltage was 5.02V.
HT3.4.3	Yes	The battery output voltage was 40.1V and the output current was 18.01mA. The converter output voltage was 12.07V and the output current was 51.30mA. Therefore, the power efficiency is Pout/Pin = (12.07*51.3)/(40.1*18.01) = 0.8574 = 85.74%.
HT3.3.4	Yes	The battery output voltage was 40.1V and the output current was 2.6mA. The converter output voltage was 5.02V and the output current was 19.26mA. Therefore, the power efficiency is Pout/Pin = (5.02*19.26)/ (40.1*2.6) = 0.9273 = 92.73%.

Table 3.4 Power Conversion Results

3.5. Power Delivery

3.5.1. Target

Verify that all the components are receiving power according to their specifications. This includes the main navigation components: motor drivers and Arduino Mega, as well as the decision-making component: the Jetson NX. Since the Arduino Mega and the Jetson NX have on-board power delivery systems it will be assumed that components getting powered through either of those boards will receive the correct power if the related board gets the correct power.

3.5.2. Evaluation Metrics

Test Tag	Requirement Tag	Requirement Description	Pass Criteria
HT3.5.1	NF1.4	All navigation components must be supplied with power according to their specifications	The Arduino Mega is receiving between 7-12V
HT3.5.2	NF1.4	All navigation components must be supplied with power according to their specifications	The motor drivers are receiving 24-48V
HT3.5.3	NF2.4	All decision-making components must be supplied with power according to their specifications	The Jetson NX is receiving between 9-18V
HT3.5.4	F3.7	The robot shall have audio and/or visual warning devices to alert people when moving	The Agrieyes light beacon is receiving between 12-24V

Table 3.5 Power Delivery Evaluation Metrics

3.5.3. Safety Considerations

Dealing with medium voltage and current electricity can result in electrocution, burns, or even blinding. Safety procedures and mitigations must be followed.

3.5.4. Procedure

HT3.5.1:

1. Connect the 36V downtube battery to the 12V buck converter
2. Connect the barrel plug cable for the Arduino to the buck converter through the associated terminal block, but do not connect the plug to the Arduino yet
3. Turn on the battery
4. Using a multimeter verify the voltage across the barrel connector to be 12V
5. Connect the barrel plug to the Arduino and verify that it powers on
6. Turn off the battery and disconnect everything

HT3.5.2:

1. Connect the 36V downtube battery to the main distribution block
2. Connect the motor driver to the 36V motor driver distribution blocks
3. Turn on the battery
4. Using a multimeter verify the voltage across the motor driver connector to be in range
5. Turn off the battery and disconnect everything

HT3.5.3:

1. Connect the 36V downtube battery to the 12V buck converter
2. Connect the barrel plug cable for the Jetson to the buck converter through the associated terminal block, but do not connect the plug to the Jetson yet
3. Turn on the battery
4. Using a multimeter verify the voltage across the barrel connector to be 12V
5. Connect the barrel plug to the Jetson and verify that it powers on
6. Turn off the battery and disconnect everything

HT3.5.4:

1. Connect the 36V downtube battery to the 12V buck converter
2. Turn on the battery
3. Using a multimeter read the voltage output by the buck converter through the terminal block and verify that it is 12V
4. Turn off the battery

5. Screw in the power cables from the beacon into the 12V terminal block (figure 3.12)
6. Turn on the battery
7. Verify that the beacon is being powered on and flashing a bright light

Figure 3.12 Terminal Distribution Block

3.5.5. Test Results

Test Tags and Number	Pass	Notes
HT3.5.1	Yes	The Arduino Mega input voltage was 12.01V
HT3.5.2	Yes	The motor driver input voltage was 39.8V
HT3.5.3	Yes	The Jetson NX input voltage was 12.01V
HT3.5.4	Yes	The Agrieyes light beacon input voltage was 12.01V

Table 3.6 Power Delivery Results

3.6. E-stop Function and Accessibility

3.6.1. Target

This test is to validate whether the onboard E-stop switch kills power to the system and to verify that the E-stop is in an easily accessible location.

3.6.2. Evaluation Metrics

Test Tag	Requirement Tag	Requirement Description	Pass Criteria
HT 3.6.1	F3.3, F3.4	The robot shall have an onboard switch that kills power to the motor drivers	Power must be cut to all devices

HT 3.6.2	NF3.2	The E-stop should be in an easily accessible E-stop on the robot	Users should be able to easily reach the E-stop

Table 3.7 E-stop Function and Accessibility Evaluation Metrics

3.6.3. Safety Considerations

Dealing with high voltage and current electricity can result in electrocution, burns, or even blinding. Safety procedures and mitigations must be followed.

3.6.4. Procedure

HT3.6.1:

1. Connect the E-stop in series with the power line being delivered to the motor driver distribution block from the battery distribution block in the off position
2. Connect the battery and turn it on
3. Send a "forward" command on the RC transmitter and verify that no wheel spinning occurs
4. Turn the E-stop on
5. Verify that power is now being delivered by observing wheel spinning
6. Turn off E-stop and once again confirm that the motors stop spinning
7. Turn off the battery

HT3.6.2:

1. Setup Propbot in an open 10mx10m area so that it can be easily circumnavigated
2. Place the E-stop in the desired location on top of Propbot
3. Choose 8 evenly spaced points on a circle surrounding Propbot and that is 0.25m away from Propbot at the nearest point. (figure 3.13)
4. Stand on each point and attempt to trigger the E-stop button
5. Record the success and difficulty rating at each point

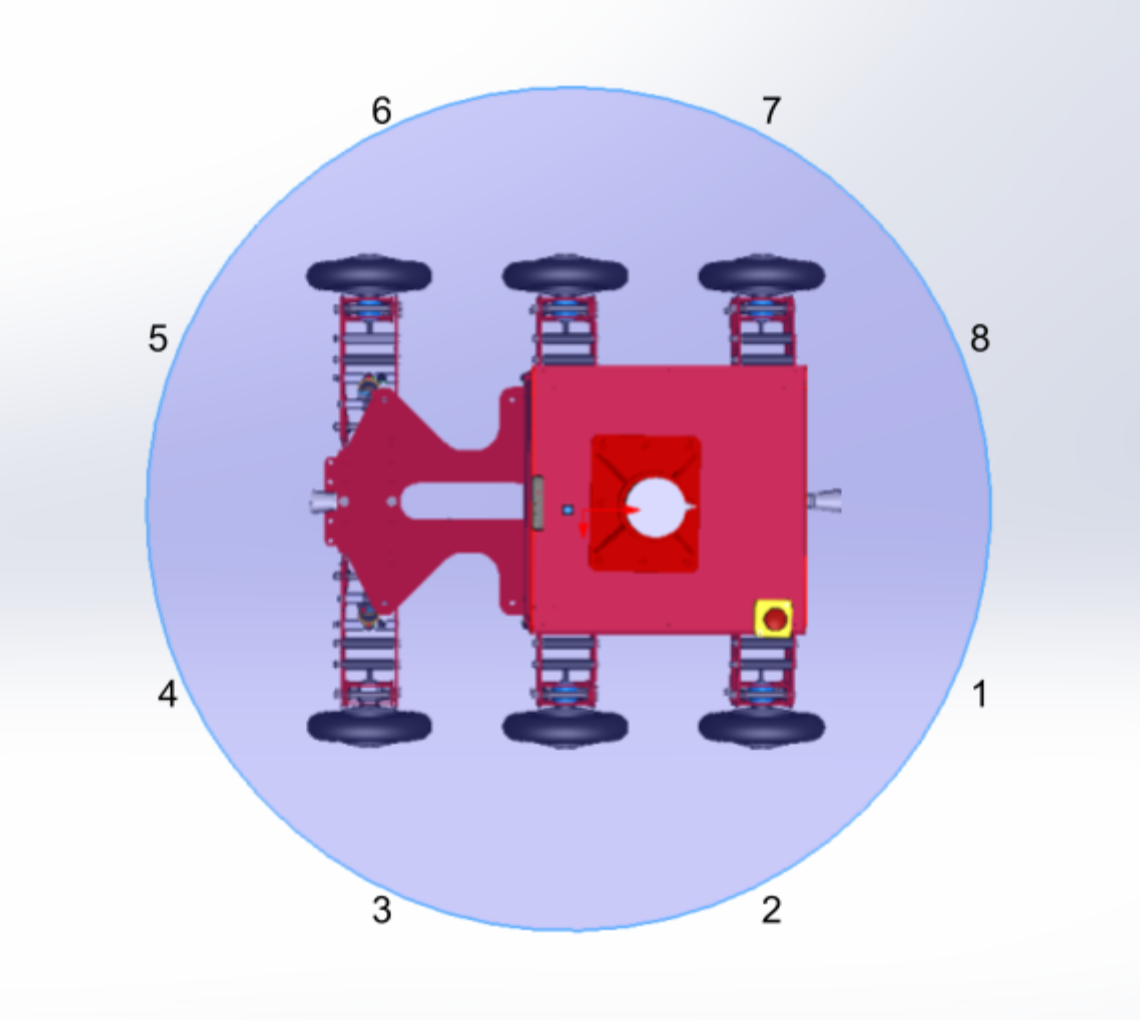

Figure 3.13 E-Stop Accessibility Test Locations

3.6.5. Test Results

Test Tag	Pass	Notes
HT 3.6.1	Yes	The E-stop could cut power to the motor drivers
HT 3.6.2 - 1	Yes	E-stop can be reached very easily
HT 3.6.2 - 2	Yes	E-stop can be reached very easily
HT 3.6.2 - 3	Yes	E-stop can be reached easily
HT 3.6.2 - 4	Yes	E-stop can be reached

HT 3.6.2 - 5	Yes	E-stop can barely be reached
HT 3.6.2 - 6	Yes	E-stop can be reached
HT 3.6.2 - 7	Yes	E-stop can be reached easily
HT 3.6.2 - 8	Yes	E-stop can be reached very easily

Table 3.8 E-stop Function and Accessibility Results

3.7. Light Beacon

3.7.1. Target

This test is to validate whether the light beacon can easily warn pedestrians of Propbot's presence.

3.7.2. Evaluation Metrics

Test Tag	Requirement Tag	Requirement Description	Pass Criteria
HT3.7.1	F3.7	The robot shall have a visual warning device to alert nearby people	The lights from the beacon are easily seen from all angles at a radial distance of 5m

Table 3.9 Light Beacon Evaluation Metrics

3.7.3. Safety Considerations

Dealing with medium voltage and current electricity can result in electrocution, burns, or even blinding. Safety procedures and mitigations must be followed.

3.7.4. Procedure

HT3.7.1:

1. Setup Propbot in an open 10m by 10m area so that it can be easily circumnavigated
2. Place the beacon in the desired location on top of Propbot
3. Choose 8 evenly spaced points on a circle surrounding Propbot that are 5m away from the centre of Propbot. (figure 3.14)
4. Power on the beacon using the steps outlined in section 3.2
5. Stand on each point and access the visibility of the flashing beacon
6. Record the success and difficulty rating at each point

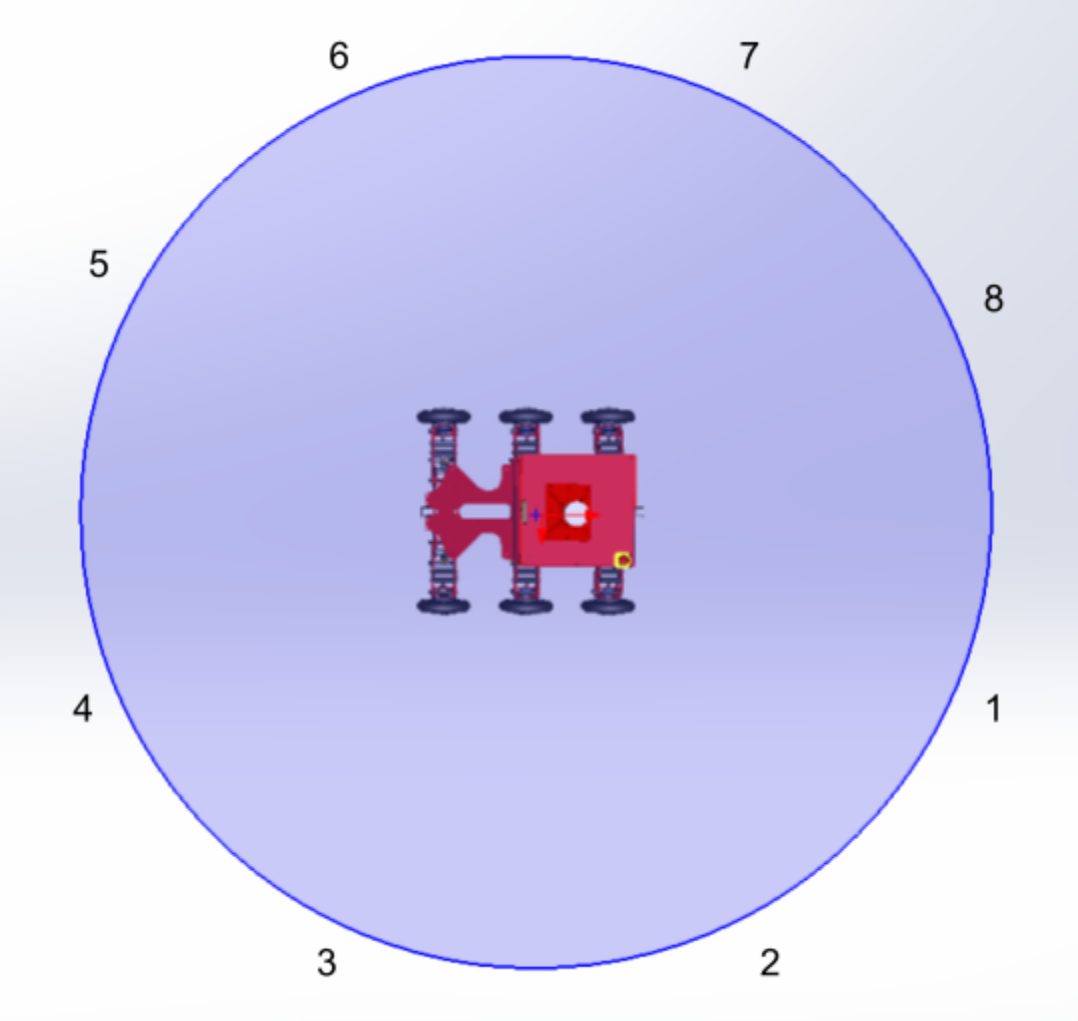

Figure 3.14 Light Beacon Visibility Test Locations

3.7.5. Test Results

Test Tag	Pass	Notes
HT 3.7.1 - 1	Yes	Beacon light can be seen very easily
HT 3.7.1 - 2	Yes	Beacon light can be seen very easily
HT 3.7.1 - 3	Yes	Beacon light can be seen very easily
HT 3.7.1 - 4	Yes	Beacon light can be seen very easily

HT 3.7.1 - 5	Yes	Beacon light can be seen very easily
HT 3.7.1 - 6	Yes	Beacon light can be seen very easily
HT 3.7.1 - 7	Yes	Beacon light can be seen very easily
HT 3.7.1 - 8	Yes	Beacon light can be seen very easily

Table 3.10 Light Beacon Results

3.8. Existing Motor Controllers and Motors

3.8.1. Target

These tests validate the functionality of the legacy motor controllers and motors in the following aspects: speed and direction control, braking, and smooth operation of wheels.

Note that the existing motor controllers were used since the Roboteq controllers could not arrive in time due to supply chain issues. For setup and procedures regarding the Roboteq controllers, see Appendix section A.3.

3.8.2. Evaluation Metrics

Test Tag	Requirement Tag	Requirement Description	Pass Criteria
HT3.8.1	F1.2	The robot shall be able to move longitudinally when in remote control mode or autonomy mode	All four motor controllers should be able to vary the motor speed, direction, and perform braking

Table 3.11 Motor Controllers and Motors Evaluation Metrics

3.8.3. Safety Considerations

In order to perform individual testing of motors, a hydraulic jack must be used to elevate Propbot. Follow safety guidelines in section 3.2.3 for procedure and safety considerations in jacking up Propbot.

Spinning motors pose the risk of catching loose clothing or long hair. This risk can be eliminated by ensuring long hair is tied back and loose clothing is not worn when testing Propbot motors while elevated.

3.8.4. Procedure

The following procedure requires Propbot to be elevated. Instructions for jacking up Propbot can be found in Section 3.2.3.

The following procedures require that the motor controllers and microcontroller be connected with digital IO and PWM pins. For motor controller pinouts, refer to figure 3.15. Note that the unlabelled connections are not relevant for Propbot control.

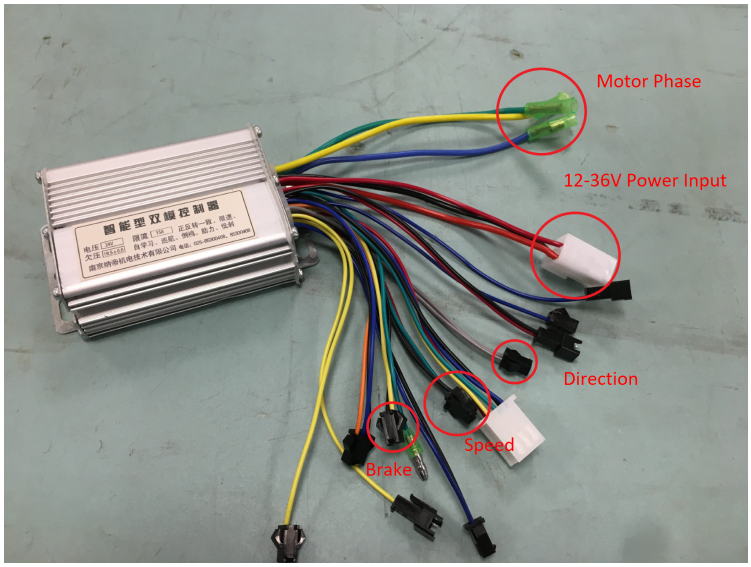

Figure 3.15

A simple Arduino-based script to test the speed, direction, and braking functionality can be found in the following motor_test.ino script: https://github.com/UBC-RSL-PropBot/Propbot_Vehicle_Interface_Firmware/tree/main/Test/motor_test . Ensure that pin mapping matches that of what is mapped to the Arduino being used to run the script.

HT3.8.1:

1. Connect pinouts from Arduino to motor controller and upload test script
2. Observe motor speed ramp up in both forwards and reverse direction followed by a braking of the wheels. This process should continue indefinitely until power from the Arduino is removed.
3. Repeat steps 1-2 for all remaining motor controllers

3.8.5. Test Results

Test Tag	Pass	Notes
HT 3.8.1	Yes	All four motor controllers and motors displayed the ability to vary the speed, direction, and perform braking

Table 3.12 Motor Controllers and Motors Results

3.9. Depth Camera

3.9.1. Target

The ability of the autonomy computer to receive raw data information from the Intel RealSense d435, using our configuration file for the camera, and its ability of the autonomy computer to visualize received image data.

3.9.2. Evaluation Metrics

Test Tag	Requirement Tag	Requirement Description	Pass Criteria
HT 3.9.1	NF2.3	The robot shall be able to take in data from an array of sensors and perform localization	ROS topics /camera/raw and /camera/points/depth must be publishing data Rviz is able to visualize received data

Table 3.13 Depth Camera Evaluation Metrics

3.9.3. Procedure

1. Plug in Intel RealSense to Autonomy Computer USB port
2. On the Autonomy Computer, run the command “roslaunch realsense2_camera rs_camera.launch”
3. Open Rviz and add topics /camera/raw and /camera/points/depth
4. Verify that resulting camera feed corresponds to what should be expected from a camera

3.9.4. Results

Test Tag	Pass	Notes
HT 3.9.1	Yes	camera data is published and can be viewed in Rviz

Table 3.14 Depth Camera Results

3.10. GPS - Global Positioning System

3.10.1. Target

The RTK GPS (U-Blox Neo-M8P) deployment is responsible for acquiring accurate and precise longitude and latitude data.

It is difficult to find highly accurate location data in an open field to compare against the GPS output data. To overcome this, we resort to finding the consistency of measurements taken at multiple points [2], while also verifying that a trace of a trip corresponds to the trip taken.

3.10.2. Evaluation Metrics

Test Tag	Requirement Tag	Requirement Description	Pass Criteria
HT 3.10.1	NF5.1, F4.3, F5.1	The time-stamped location data must be accurate within 0.5m	Ublox U-center program shows location to be corresponding to accurate Data range at one point is less than 0.5m

Table 3.15 GPS Evaluation Metrics

3.10.3. Procedure

Test Scenario: outdoor in open field

Stage 1:

1. Setup base and rover GPS by connecting receiver and UHF antenna
2. Use USB cable to connect base GPS board to laptop, rover to mobile charger
3. Open U-Center program [3]
4. Establish serial connection with both GPS through U-Center

5. After 5 minutes, verify that
    a. Rover has entered RTK mode (flashing green light)
    b. U-Center shows map with rover coordinates
6. Identify a path to traverse. The ideal path has identifiable features on common map services, like Google Maps.
7. On U-Center, start recording GPS logs
8. Walk through path in step 6, carrying rover in hand
9. At the end of the path, stop recording logs on U-Center
10. Open the created log file and select view maps to create a visualization similar to Figure 3.16
11. Verify that the trace of the trip shows a continuous path, has no noticeable noise, and accurately depicts the path that was taken

Stage 2:

12. Mark locations A,B,C,D,E,F,G,H such that they form a straight line where each location is 0.35m away from the next
13. Iteratively move the rover to each location, recording a longitude, latitude reading form U-Center
14. Repeat steps 6 and 7 five times
15. For each location, find the range of distance between samples at the same location (Figure 3.17)
16. Verify that all ranges are less than 0.5m

3.10.4. Results

Figure 3.16 shows the trace of the trip conducted for this test. The path corresponds to a 50 meter long, 1 meter-wide pathway in a public park that is noticeable on Google Maps. The dotted green represents GPS data along the trip. The dotted green is always at close proximity to the pathway, indicating that the reading is very accurate.

Our precision test results are shown in Figure 3.17. We conclude that variability at a single location is always less than 0.1m, indicating high precision.

Test Tag	**Pass**	**Notes**
HT 3.10.1	Yes	For calculation of distance between two GPS coordinates, we use a flat-earth approximation, which is appropriate for very small distances.

		We believe more testing needs to be done in different environments (ex near buildings, at high speeds). Its also important to find how far the base-rover communication can be. On one of our trips we were getting signals from 350m from base, but that is a lower bound.

Table 3.16 GPS Results

Figure 3.16 The tested

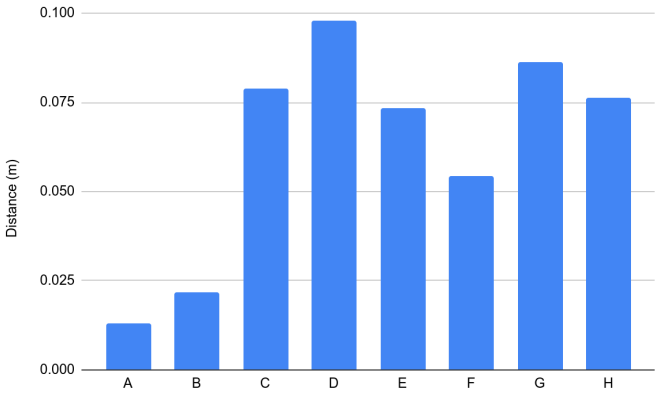

Figure 3.17 Distance between samples at same location

3.11. IMU - Inertial Measurement Unit

3.11.1. Target

The purpose of this test is to validate IMU integration with the ROS autonomy package on the Nvidia Jetson NX. The autonomy package requires orientation and acceleration data from the IMU. To validate this, we can output this data from the terminal while the ROS IMU node is running. Furthermore, we can visualize orientation and acceleration vectors using the Rviz simulation environment.

3.11.2. Evaluation Metrics

Test Tag	**Requirement Tag**	**Requirement Description**	**Pass Criteria**
HT 3.11.1	NF2.3, F3.8	The robot shall be able to take in data from an array of sensors and perform localization	The IMU should publish orientation and acceleration data from a ROS node on the autonomy computer.

		The robot shall detect an incline/decline greater than 2 degrees	

Table 3.17 IMU Evaluation Metrics

3.11.3. Safety Considerations

Ensure that the IMU is wired up correctly to avoid any electrical shorts that have the potential to cause damage to equipment and/or physical harm.

3.11.4. Procedure

The IMU used is provided by Adafruit Industries and uses the BNO055 chip. See figure 3.18.

The libraries being used to communicate with the IMU are RTIMULib and rtimulib_ros which use I2C communications and are already installed on the Jetson NX autonomy computer.

HT3.11.1:

1. Connect IMU to the Jetson NX as per figure 3.18 and figure 3.19. Ensure Vin on the IMU is powered with 5V.
2. Detect the IMU I2C bus on the Jetson NX linux shell using the command “sudo i2cdetect -r -y”
3. Create an empty text file that will be used to generate calibration data for the IMU. Edit the rtimulib_ros.launch file and change the default calibration file name to the calibration file that was just created. Don’t include the file extension.
4. Run the following command: “roslaunch rtimulib_ros <calibration_file>” to launch the ROS node where <calibration_file> is the name of the calibration file (not including the file extension) from Step 3.
5. In a new terminal run the command “rostopic echo /imu” to display the published data
6. To visualize the IMU data in RVIZ, run the command “rviz rviz”. The IMU topic should be visible and orientation and acceleration vectors should be visible.

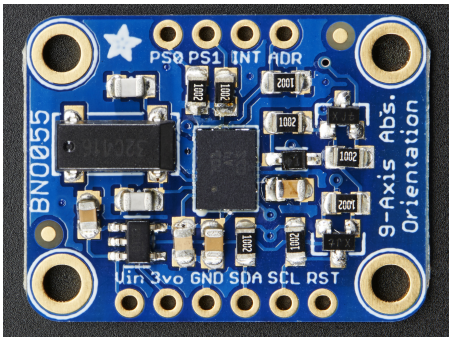

Figure 3.18 Adafruit BNO055 IMU

Sysfs GPIO	Name	Pin	Pin	Name	Sysfs GPIO
Jetson Xavier NX J12 Header					
	3.3 VDC *Power*	1	2	5.0 VDC *Power*	
	I2C1_SDA *I2C Bus 8*	3	4	5.0 VDC *Power*	
	I2C1_SCL *I2C Bus 8*	5	6	GND	

Figure 3.19 Nvidia Jetson NX J12 Pinouts 1-6

3.11.5. Test Results

Test Tag	Pass	Notes
HT 3.11.1	Yes	The IMU is able to publish orientation and acceleration data from a ROS node running on the Nvidia Jetson NX. This data was also validated using the RVIZ simulation environment as shown below. It was also validated using an onboard IMU on Propbot where the IMU orientation readings matched up with the measured incline. See figure 3.20 and 3.21

Table 3.18 IMU Results

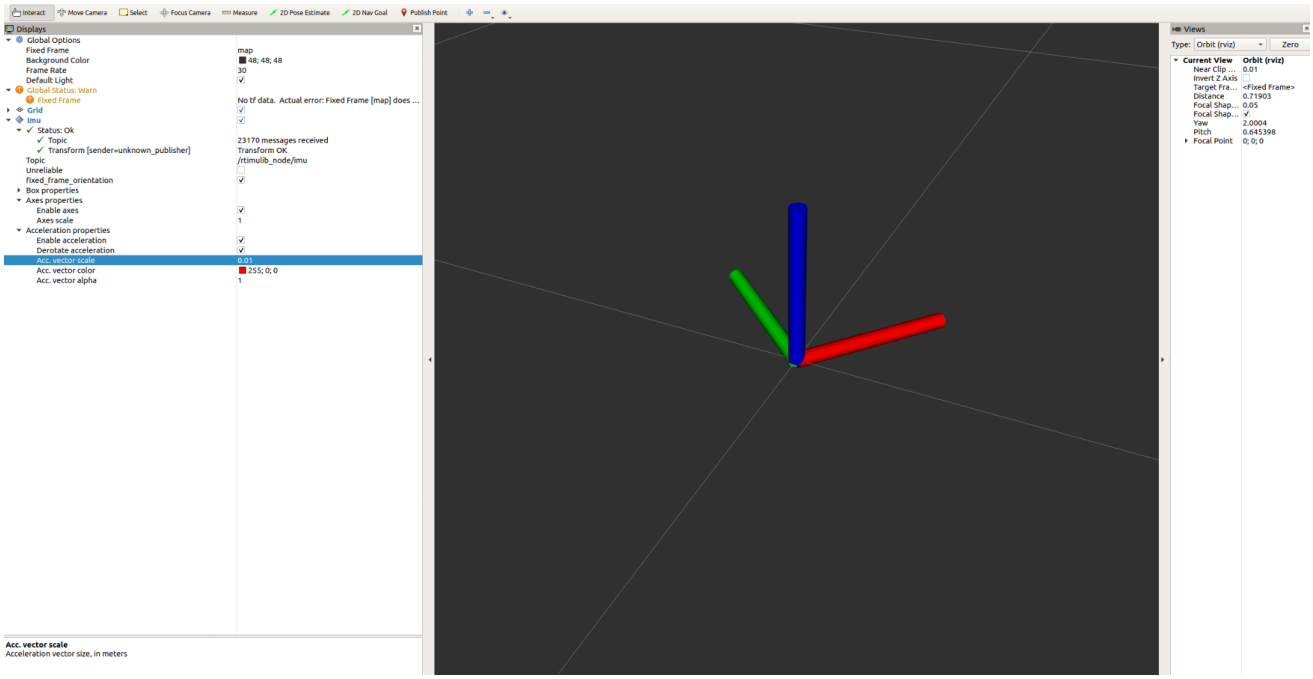

Figure 3.20 IMU Visualization in Rviz

Figure 3.21 Navigation of Propbot on an Incline

Figure 3.22 Smartphone App Showing 6-Degree Incline

3.12. WiFi Router

3.12.1. Target

This test is to validate the communication between the Autonomy Computer and the mission command center through WiFi. Although our requirements stated that Propbot should communicate over a cellular network, we elected to use WiFi, and specifically the D-Link router, because we were not able to source the cellular modem due to supply chain issues.

In the following tests, we keep the requirements as they were stated originally, and describe a test that, while run successfully using WiFi, can be run on any connection medium.

3.12.2. Evaluation Metrics

Test Tag	Requirement Tag	Requirement Description	Pass Criteria
HT 3.12.1	F4.2	The robot shall communicate with the mission command center over a cellular network	From Autonomy Computer, able to ping MCC From MCC, able to ping Autonomy Computer

HT 3.12.2	F4.5	The robot shall communicate cellular signal strength data to the mission command center	able to retrieve “Link Quality” and “Signal Level” fields from iwconfig command run on Autonomy Computer

Table 3.19 WiFi Router Evaluation Metrics

3.12.3. Procedure

1. Ensure Autonomy Computer and MCC are in the same room
2. On the computer running MCC, connect to the D-link router
3. On a terminal, run the command “ifconfig” and note the computers ip address
4. On the Autonomy Computer, connect to the D-link router
    a. if using the command line, use the command “sudo nmcli dev wifi connect propbot” and enter the router’s password
5. run the command “ifconfig” and note the Autonomy Computers’s ip address
6. On the computer running MCC, run the command “ping {Autonomy Computer IP address}”
    a. verify successful output of command, showing RTT and 0% packet loss
7. On the Autonomy Computer, run the command “ping {MCC IP address}”
    a. verify successful output of command, showing RTT and 0% packet loss
8. if both 5.a and 6.a pass, 3.12.1 passes
9. On MCC computer, run the command “ssh propbot@{Autonomy Computer IP address} iwconfig”
    a. Verify successful output of command, showing “Link Quality” and “Signal Level”

3.12.4. Results

Test Tag	**Pass**	**Notes**
HT 3.12.1	No	Passes with WiFi, not with Cellular. In lab, RTT was 14.8ms.

HT 3.12.2	No	Passes with WiFi, not with Cellular. In lab, Link Quality was 66/70, Signal Level was -44dBm

Table 3.20 WiFi Router Results

4. Integration Tests

4.1. Overview

Integration validation involved integrating our design contributions with the existing legacy system and verifying their compatibility. This involved the wiring and power management system and testing of the motor driver connections. Integration of the sonar sensors and short-range obstacle detection was also tested.

4.2. General Setup Procedures and Resources

4.2.1. Mock Autonomy Integration Setup

The underlying premise for Mock Autonomy Integration tests is that some aspects of Propbot, such as data collection and modes of operations, do not need a functioning autonomy system to be validated. This means the Autonomy System can be replaced with a “Test Client” which outputs wheel speeds as the autonomy system would.

The general setup is shown in figure 4.1. Unlike in an Autonomy Simulation setup (Section 2.1) methods, no autonomy package or simulation of raw data is needed. Instead, the entire autonomy system is mocked and reduced to a “Test Client” which propagates wheel speeds it receives from human input (Tester #1). The VIS receives wheel speeds from the “Autonomy System” and from a fallback user (Tester #2, who is using a remote controller). Observable behaviours, which can serve as evaluation metrics, are the movements of Propbot and their response to the fall-back user, and messages communicated to MCC.

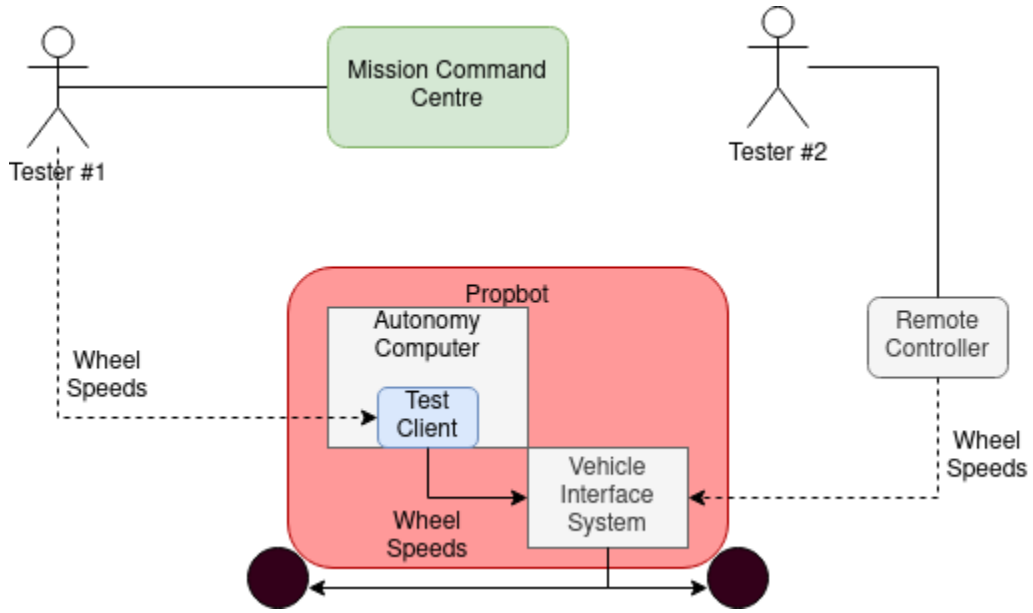

Figure 4.1 Autonomy Simulation General Setup

4.2.2. Integrated Simulation Setup

The integrated simulation setup is a combination of both Autonomy Simulation setup (Section 2.2.1) and Mock Autonomy Integration setup (Section 4.2.1). It combines the use of both simulation and real hardware to test the ability of the Propbot to deal with raw data published at a high frequency. This is useful for testing Propbot as a whole under a limited sensor suite.

The setup is represented in figure 4.2. The Gazebo simulator must, at startup, have a view of the world that matches the real, physical environment Propbot is in. The role of each component during the test is described below:

Gazebo Simulator: uses the current world to generate raw mock data, which is sent to the Autonomy System. It also receives the Velocity Commands generated by the Autonomy System and converts them to wheel speeds, which is applied to the simulated Propbot through Gazebo physics engine and "World Updater". The process of updating the world and publishing simulated data based on the latest world continues for the duration of the test.

Autonomy System: behaves normally, receiving sensor data and outputting Velocity Command based on the algorithms in the autonomy package.

Vehicle Interface System: periodically queries the Simulator for the wheel speeds the simulator is using, and applies them to real life Propbot.

Based on this architecture, a perfect (real) Propbot would show movement identical to (simulated) Propbot. From this expectation, we are able to derive evaluation criteria that can be used to validate the Integration of the Autonomy System and the Vehicle Interface System.

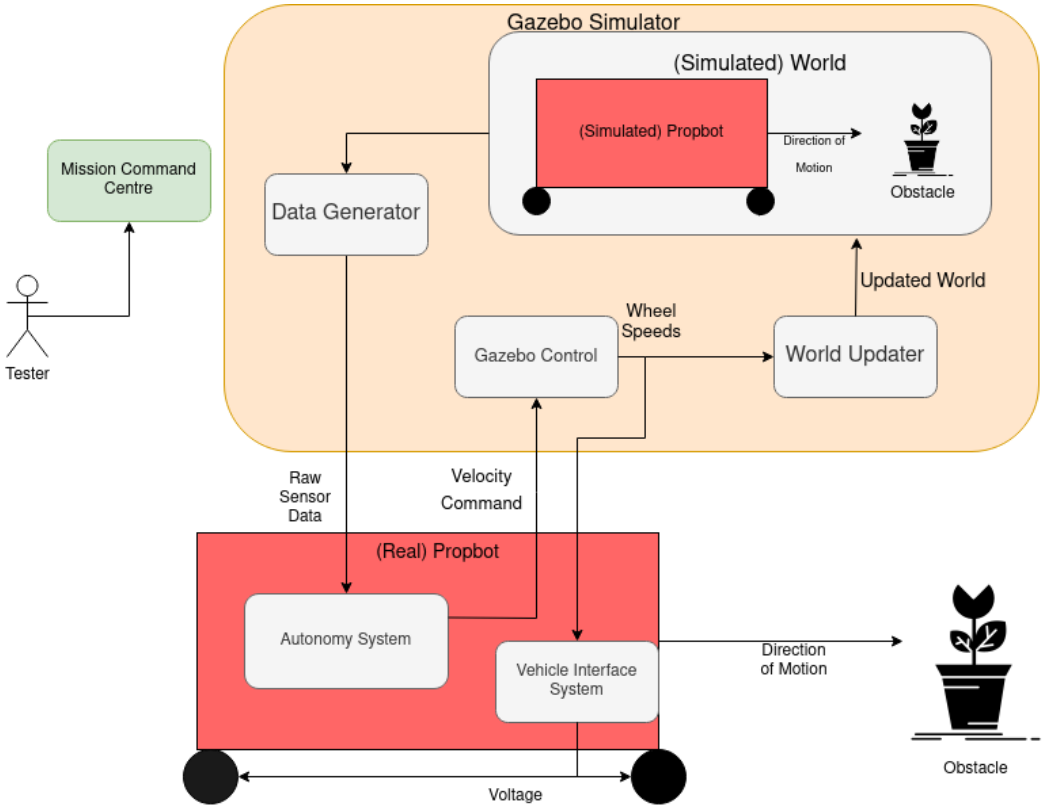

Figure 4.2 Integrated Simulation Setup

4.3. Teleoperation

4.3.1. Maximum and Minimum Speed

4.3.1. Target

The purpose of this test (figure 4.3) was to analyze Propbot's linear velocity on the ground as a function of the digital PWM signals it is receiving. Using this data, we were able to generate curves in both the forward and reverse directions and determine the relationship between these two parameters. Once the data was formatted, we were able to select PWM values that corresponded to velocities within the bounds we have set in our requirements. Namely, a minimum velocity of 1km/hr and a maximum velocity of 5km/hr when Propbot is in motion in either teleoperation or autonomy mode.

Figure 4.3 Maximum and Minimum Speed Test (Forward Direction)

4.3.1. Evaluation Metrics

Test Tag	Requirement Tag	Requirement Description	Pass Criteria
HT4.3.1	F1.2	The robot shall be able to move longitudinally when in remote control mode or autonomy mode	The robot can move longitudinally (ie: parallel to its forward orientation) when controlled via the transmitter
HT 4.3.2	F1.8	Robot max and min speed shall be configurable in both autonomy and remote control mode	The user is able to issue different PWM values corresponding to different maximum and minimum speeds as specified by NF1.5 and NF1.6. This must be possible in both forward and reverse
HT 4.3.3	NF1.5	The minimum robot speed when in motion shall be 1km/hr	The user sets a minimum PWM input for the robot. After the robot reaches this speed following the ramp-up function this speed is not less than 1km/hr in either the forward or reverse direction
HT 4.3.4	NF1.6	The maximum robot speed when in motion shall be 5km/hr	The maximum speed the robot achieves after the ramp-up in both the forward and reverse direction does not exceed 5km/hr

HT 4.3.5	NF1.7	The robot's movement shall be fluid	The robot does not exhibit "jerky" motion when moving in the forward or reverse direction

Table 4.1 Maximum and Minimum Speed Evaluation Metrics

4.3.1. Safety Considerations

Several safety measures were put in place to ensure that this test was conducted in a safe manner. One team member was placed at the entrance of the hallway nearest the computer labs and made sure no one turned the corner as this test was taking place. Another team member was placed at the other end of the hallway near the ECE lounge with the same responsibility. One team member operated Propbot and was responsible for doing so in a safe manner while also applying the remote E-Stop if necessary. The final team member was in charge of filming and also triggering the manual E-Stop should something go wrong with teleoperation. Therefore, a total of four team members were needed to ensure that this test was conducted safely. Additional precautions for operating with high voltage devices and moving Propbot's heavy frame were also followed.

4.3.1. Procedure

We used the time it took Propbot to traverse a fixed distance at a constant speed to determine the linear velocity. The distance selected was 24ft which did not exceed beyond the corridor and pose a safety concern, but allowed Propbot enough room to ramp up to its maximum velocity before entering the desired area. We measured the distance and put black tape at both the initial and final points. An iPhone video was taken as Propbot travelled this distance to determine the exact frame it entered and left the area. The frames are time-stamped so the difference in time between the two frames gives us the time it takes Propbot to travel from the initial to the final point. This gave us an accurate measure of the time it took for Propbot to travel 24ft and from that data, we were able to determine the linear velocity and then convert that to km/hr in excel. The procedure below describes how the trials were conducted in the forward direction.

1. Power on battery
2. Power on the transmitter and ensure successful pairing with receiver
3. Spotter 1 ensures that the corridor is clear
4. Spotter 2 ensures that the corridor is clear
5. Disengage E-Stop
6. Accelerate Propbot in the forward direction to the starting position by placing the left joystick in the UP direction
7. Take a video of Propbot entering and leaving the area
8. Stop Propbot, execute a 180-degree turn, and return it to the starting position
9. Engage E-Stop

10. Analyze video and enter values into excel
11. Repeat steps 3-10 for different PWM values

4.3.1. Test Results

Test Tag	Pass	Notes
HT 4.3.1	Yes	The robot is able to move longitudinally in the forward direction
HT 4.3.2	Yes	In the forward direction, the user is able to input different PWM values and achieve a range of speeds within the boundaries given by NF1.5 and NF1.6
HT 4.3.3	Yes	The minimum PWM value (90) selected from the data in the forward direction does not fall below 1km/hr
HT 4.3.4	Yes	The maximum PWM value (105) selected from the data in the forward direction does not exceed 5km/hr
HT 4.3.5	Yes	The robot does not exhibit “jerky” motion when moving in the forward direction

Table 4.2 Maximum and Minimum Forward Speed Results

Trial	PWM	Velocity (km/hr)
1	110	5.85
2	105	4.88
3	100	3.70
4	95	2.80
5	90	1.78
6	85	0.77

Table 4.3 Maximum and Minimum Forward Speed Trial Results

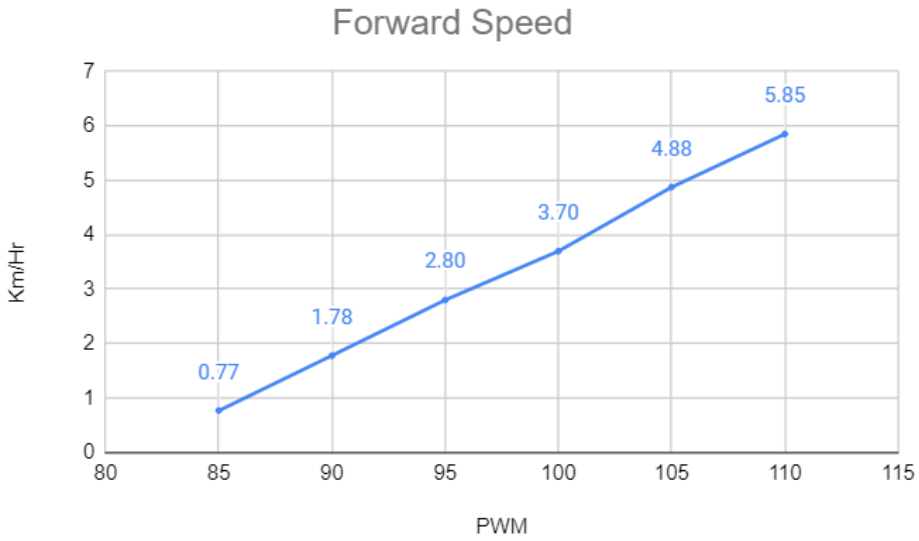

Figure 4.4 Maximum and Minimum Forward Linear Velocity vs. PWM Plot

4.3.1. Safety Considerations

The same safety precautions as in 4.3.1 Safety Considerations were followed when conducting the trials in the reverse direction.

4.3.1. Procedure

The procedure is the same as described in 4.3.1 Procedure except now the direction signal is flipped and these trials were conducted with Propbot moving in the reverse direction.

4.3.1. Test Results

Test Tag	**Pass**	**Notes**
HT 4.3.1	Yes	The robot is able to move longitudinally in the reverse direction
HT 4.3.2	Yes	In the reverse direction, the user is able to input different PWM values and achieve a range of speeds within the boundaries given by NF1.5 and NF1.6
HT 4.3.3	Yes	The minimum PWM value (85) selected from the data in the reverse direction does not fall below 1km/hr
HT 4.3.4	Yes	The maximum PWM value (105) selected from the data in the reverse direction does not exceed 5km/hr

HT 4.3.5	Yes	The robot does not exhibit “jerky” motion when moving in the reverse direction

Table 4.4 Maximum and Minimum Reverse Speed Results

Trial	PWM	Velocity (km/hr)
1	110	5.54
2	105	4.59
3	100	3.57
4	95	2.51
5	90	1.78
6	85	1.01

Table 4.5 Maximum and Minimum Reverse Speed Trial Results

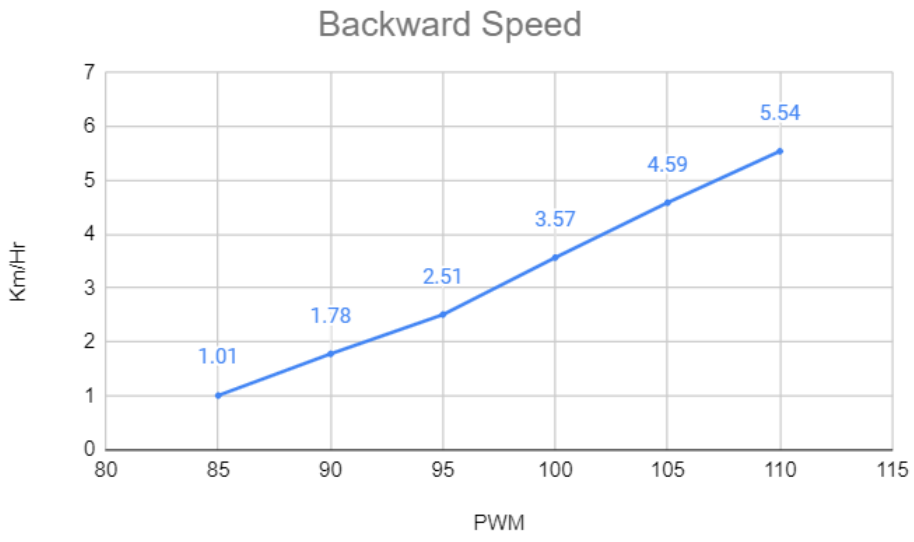

Figure 4.5 Maximum and Minimum Reverse Linear Velocity vs. PWM Plot

4.3.1. Analysis

From these tests we were able to determine that the linear velocity of Propbot on the ground rises linearly with PWM signals in both the forward (figure 4.4) and reverse directions (figure 4.5). Furthermore, by plotting and analyzing this data we were able to select PWM values corresponding to our maximum and minimum speeds as per NF1.5 and NF1.6. In the forward direction, we selected the minimum PWM to be 90 which corresponds to a linear velocity of 1.78km/hr. We selected the maximum PWM to be 105 which corresponds to a linear velocity of 4.88km/hr. Likewise in the reverse direction, we selected the minimum PWM to be 85 which corresponds to a linear velocity of 1.01km/hr. We selected the maximum PWM to be 105 which corresponds to a linear velocity of 4.59km/hr. The robot was also able to move longitudinally in both the forward and reverse directions and exhibit fluid movement thus validating F1.2 and NF1.7.

4.3.2. Turning Left and Right

4.3.2. Target

The purpose of this test (figure 4.6) was to ensure that Propbot can execute turns.

Figure 4.6 Turning Left and Right Test

4.3.2. Evaluation Metrics

Test Tag	Requirement Tag	Requirement Description	Pass Criteria
HT 4.3.6	F1.3	The robot shall be able to execute turns	Propbot is able to execute a full 360 turn clockwise and anti-clockwise
HT 4.3.7	NF1.7	The robot's movement shall be fluid	Propbot visually appears to executes turn smoothly

Table 4.6 Maximum and Minimum Speed Evaluation Metrics

4.3.2. Safety Considerations

Several safety measures were put in place to ensure that this test was conducted in a safe manner. We conducted this test in the atrium of the Life building on the second floor. We made sure that we were performing the test when no one was around Propbot and that Propbot had ample space to move forward, backward, and execute turns.

4.3.2. Procedure

We used the time it took Propbot to execute a 360 turn in each direction to determine if Propbot is able to execute turns. We used a back tape marker to indicate 0-degree orientation for Propbot. An iPhone video was taken as Propbot executed turns to determine the exact frame when it completed the 360-degree turn. The frames are time-stamped so the difference in time between the two frames gives us the time it takes Propbot to travel from the initial to the final point. This gave us an accurate measure of the time it took for Propbot to execute a 360 turn. The procedure below describes how the trials were conducted in the forward direction.

1. Power on battery
2. Power on the transmitter and ensure successful pairing with receiver
3. Spotter 1 ensures that the front of the atrium is clear
4. Spotter 2 ensures that the back of the atrium is clear
5. Disengage E-Stop
6. Execute a Left turn by holding the right joystick down.
7. Take a video of Propbot executing a full 360 turn starting from the 0-degree marker.
8. Stop Propbot by setting the right joystick back to middle position
9. Execute a Right turn by holding the right joystick up
10. Stop Propbot by setting the right joystick back to middle position
11. Analyze video and record the time values
12. Repeat steps 6 through 11 two more times

4.3.2. Test Results

Test Tag	Pass	Notes
HT 4.3.6	Yes	Propbot is able to execute turns
HT 4.3.7	Yes	The motion appears to be fluid while executing turns

Table 4.7 Turning Left and Right Results

Trial	Direction	Time taken to execute a full 360 turn
1	LEFT	8.2 seconds
2	RIGHT	8.3 seconds
3	LEFT	7.9 seconds
4	RIGHT	8.1 seconds
5	LEFT	8.2 seconds
6	RIGHT	8.3 seconds

Table 4.8 Turning Left and Right Trial Results

4.3.2. Analysis

From these tests we were able to determine that the Propbot can execute turns in either direction. We noticed that it takes Propbot 8.2 seconds to execute a 360 turn clockwise (right) and 8.1 seconds to execute a 360 turn anticlockwise (left).

4.3.3. Maximum Braking Distance

4.3.3. Target

The purpose of this test (figure 4.7) was to measure the maximum braking distance it takes for Propbot to come to a full stop from max speed.

Figure 4.7 Maximum Braking Distance Test

4.3.3. Evaluation Metrics

Test Tag	Requirement Tag	Requirement Description	Pass Criteria
HT 4.3.8	NF1.2	The robot shall come to a full stop within 0.5m of the brake being triggered	Propbot is able to come to a full stop within 0.5m

Table 4.9 Maximum Braking Distance Evaluation Metrics

4.3.3. Safety Considerations

Several safety measures were put in place to ensure that this test was conducted in a safe manner. One team member was placed at the entrance of the hallway nearest the computer labs and made sure no one turned the corner as this test was taking place. Another team member was placed at the other end of the hallway near the ECE lounge with the same responsibility. One team member operated Propbot and was responsible for doing so in a safe manner while also applying the remote E-Stop if necessary. The final team member was in charge of filming and also triggering the manual E-Stop should something go wrong with teleoperation. Therefore, a total of four team members were needed to ensure that this test was conducted safely. Additional

precautions for operating with high voltage devices and moving Propbot's heavy frame were also followed.

4.3.3. Procedure

We measure the distance it takes for Propbot to come to a full stop from its maximum speed of 5 km/h. We used a back tape marker to indicate when the brake should be triggered and used a measuring tape to measure the distance. This gave us an accurate measure of the distance it took for Propbot to come to a full stop. The procedure below describes how the trials were conducted.

1. Power on battery
2. Power on the transmitter and ensure successful pairing with receiver
3. Spotter 1 ensures that the corridor is clear
4. Spotter 2 ensures that the corridor is clear
5. Disengage E-Stop
6. Accelerate Propbot in the forward direction to the starting position by placing the left joystick in the UP position until the max speed is achieved.
7. Trigger the brakes by setting the throttle to 0 by placing the left joystick in the middle position.
8. Measure the distance from the brake trigger position marker to the position where Propbot (front wheels) came to a full stop.
9. Engage E-Stop
10. Enter values into excel
11. Repeat steps 3-10 for two more times
12. Repeat steps 3-11 in the backward direction

4.3.3. Test Results

Test Tag	**Pass**	**Notes**
HT 4.3.8	Yes	Propbot is able to come to a full stop within 0.5m

Table 4.10 Maximum Braking Distance Results

Trial	**Direction**	**Distance measured from brake trigger maker to full stop position**
1	FWD	21cm
2	FWD	24cm

3	FWD	22cm
4	BKWD	23cm
5	BKWD	25cm
6	BKWD	22cm

Table 4.11 Maximum Braking Distance Trial Results

4.3.3. Analysis

From these tests we were able to determine that the Propbot can come to a full stop within 0.5m of the brake being triggered.

4.3.4. Short Range Obstacle Detection and Avoidance

4.3.4. Target

To confirm that the robot can detect an obstacle in short-range distances (20cm to 675cm). This is to test the fail-safe obstacle detection which allows Propbot to avoid immediate collisions that would have otherwise resulted from mistakes made by the autonomy system or the RC user (figure 4.8).

4.3.4. Evaluation Metrics

Test Tag	**Requirement Tag**	**Requirement Description**	**Pass Criteria**
HT 4.3.9	NF2.1, F2.3, F1.4, NF1.2	The robot shall maintain a distance of at least 0.5m from surrounding objects and come to a full stop if the obstacle is not avoidable.	The robot came to a full stop and maintained a distance of at least 0.5m from the obstacles.

Table 4.12 Short Range Obstacle Detection and Avoidance Evaluation Metrics

4.3.4. Safety Considerations

Several safety measures were put in place to ensure that this test was conducted in a safe manner. One team member was placed at the entrance of the hallway nearest the computer labs and made sure no one turned the corner as this test was taking place. Another team member was placed at the other end of the hallway near the ECE lounge with the same responsibility. One team member operated Propbot and was responsible for doing so in a safe manner while also

applying the remote E-Stop if necessary. The final team member was in charge of filming and also triggering the manual E-Stop should something go wrong with teleoperation. Therefore, a total of four team members were needed to ensure that this test was conducted safely. Additional precautions for operating with high voltage devices and moving Propbot's heavy frame were also followed. This test involves a high risk of collision and should be performed in a closed, safe, and controlled environment.

4.3.4. Procedure

We measure the distance between the obstacle and Propbot when the Estop mode is triggered by the sonar sensors. This gave us an accurate measure of the distance between Propbot and the obstacle when it came to a full stop. The procedure below describes how the trials were conducted.

1. Place 2 traffic cones or recycling bins (obstacles), at least 3 meters away, on each end of the Propbot.
2. Ensure that the Estop button is engaged on Propbot
3. Turn on the battery of Propbot
4. Power on the transmitter and ensure successful pairing with receiver
5. Spotter 1 ensures that the corridor is clear
6. Spotter 2 ensures that the corridor is clear
7. Disengage E-Stop
8. Use the Remote Controller to manually navigate closer to the traffic cone (obstacle) in front of the Propbot at different speeds and record your observation
9. Use the Remote Controller to manually navigate closer to the traffic cone (obstacle) behind the Propbot and record your observation
10. Engage the Estop button
11. Turn off the battery of Propbot

Figure 4.8 Short Range Obstacle Detection and Avoidance Test

4.3.4. Test Results

Test Tag	Pass	Notes
HT 4.3.9	Yes	The passing of the test confirms that the sonar sensors provide fail-safe obstacle detection by activating the E-stop when 0.5m from the obstacle.

Table 4.13 Short Range Obstacle Detection and Avoidance Evaluation Results

Trial	Speed	Direction	Distance between obstacle and Propbot after the Sonar sensors detect an obstacle
1	1 kmph	FWD	52.6 cm
2	1 kmph	FWD	54.4 cm
3	1 kmph	FWD	53.9 cm
4	1 kmph	BKWD	62.1 cm
5	1 kmph	BKWD	58.7 cm

6	1 kmph	BKWD	53.1 cm
7	5 kmph	FWD	54.1 cm
8	5 kmph	FWD	55.5 cm
9	5 kmph	FWD	56.1 cm
7	5 kmph	BKWD	61.0 cm
8	5 kmph	BKWD	59.2 cm
9	5 kmph	BKWD	54.7 cm

Table 4.14 Short Range Obstacle Detection and Avoidance Trial Results

4.3.4. Analysis

From these tests, we can confirm that the Propbot is successfully stopping before hitting the obstacle and maintaining 0.5m from a nearby obstacle. We have also validated that our solution works with autonomy mode.

4.3.5. Robot Acceleration

4.3.5. Target

The purpose of this test was to characterize the acceleration of Propbot as it moves from its stopped position to its maximum speed of 5kmph.

4.3.5. Evaluation Metrics

Test Tag	**Requirement Tag**	**Requirement Description**	**Pass Criteria**
HT 4.3.10	F1.1	The robot shall be able to accelerate, decelerate, and stop when in remote control mode or autonomy mode	The goal of this test is to characterize the acceleration of Propbot as it goes from stop to its max speed.

Table 4.15 Robot Acceleration Evaluation Metrics

4.3.5. Safety Considerations

This test involves a high risk of collision and should be performed in a closed, safe, and controlled environment. Follow the same precautions as per 4.3.1 and 4.3.2.

4.3.5. Procedure

1. Setup IMU-based datalogger to record time-stamped acceleration and orientation data
2. Navigate Propbot to an unobstructed area containing a narrow stretch of at least 10m
3. Secure datalogger within Propbot chassis and start logging data
4. Use RC transmitter to accelerate Propbot to its maximum speed of 5kmph in the forward direction
5. Use RC transmitter to apply brake once Propbot reaches the end of its path
6. Repeat steps 4-5 but this time move Propbot in the backwards direction

4.3.5. Test Results

Acceleration and time data for both forward and backwards runs of Propbot were plotted on the same graph (figure 4.9). As shown in the following plot, the maximum forward acceleration, 2.3 m/s^2 occurs at around time 45 and the maximum backwards acceleration, -2.5 m/s^2 occurs at roughly time 80.

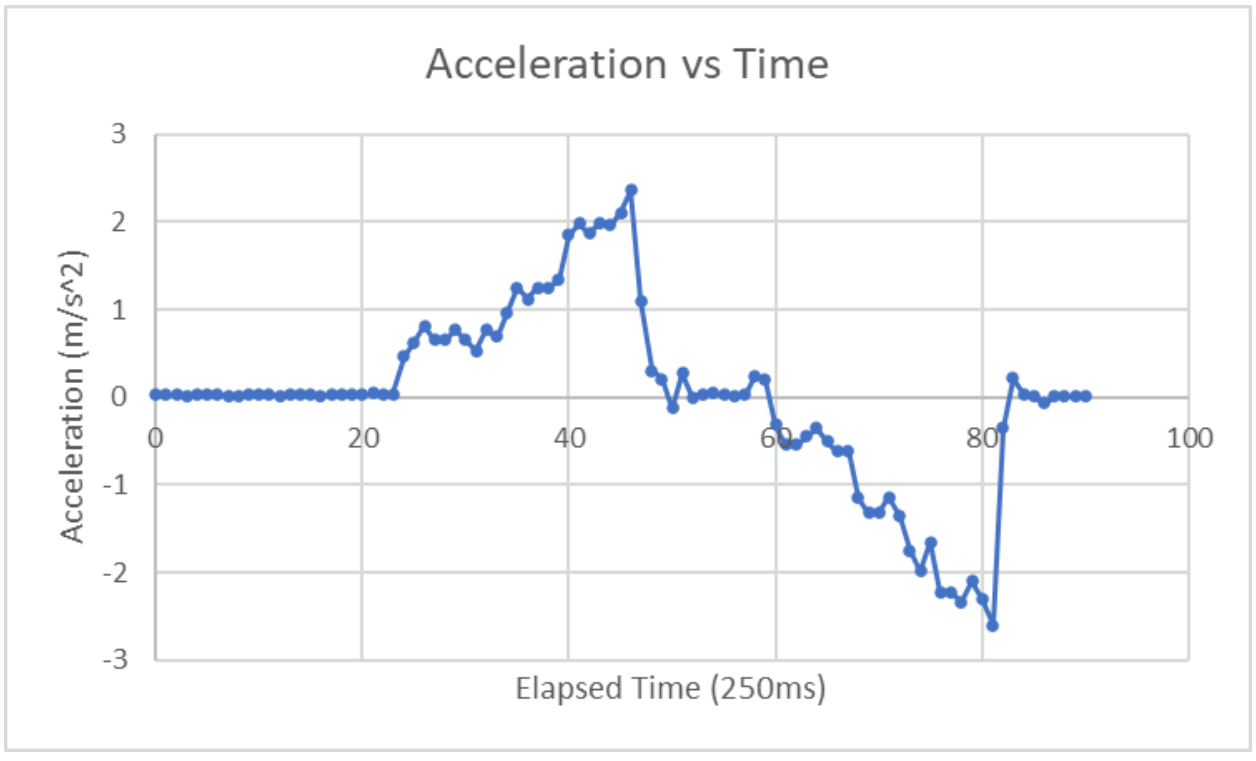

Figure 4.9 Forwards and Backwards Acceleration vs Time

Test Tag	Pass	Notes
HT 4.3.10	Yes	The goal of this test is to characterize the acceleration of Propbot as it goes from stop to its max speed.

Table 4.16 Robot Acceleration Results

This test shows us that the maximum acceleration of Propbot in both directions is roughly 2.5 m/s^2 when operated at its maximum speed of 5kmph. Performing further characterization of acceleration and velocity over a course of the larger distance of Propbot may be useful for the future team to consider. Location and velocity data can be collected from the onboard GPS.

4.3.6. Maximum Transmitter Distance

4.3.6. Target

To verify that the robot can be communicated with and controlled via the transmitter up to a distance of 100m. This test is important to determine the maximum distance at which teleoperation is viable.

4.3.6. Evaluation Metrics

Test Tag	Requirement Tag	Requirement Description	Pass Criteria
HT 4.3.11	F4.1	The robot shall be teleoperable with an RC transmitter up to a distance of 100m	The transmitter and receiver remain paired up to 100m. The transmitter can send and execute commands up to this distance

Table 4.17 Maximum Transmitter Distance Evaluation Metrics

4.3.6. Safety Considerations

When operating Propbot outside on campus (figure 4.10) we follow safety measures to ensure that no person is in the near vicinity of Propbot while it is being operated. This involves having one person operating the robot and at least 2 others being present to make sure no one gets too close. If someone is close walking by, the remote E-Stop is triggered, and someone goes beside the robot to trigger the manual E-Stop if necessary.

4.3.6. Procedure

1. Ensure sure that the manual E-Stop is engaged

2. Power on the battery
3. Power on the transmitter and verify successful pairing with receiver
4. Disengage manual E-Stop
5. Have one person stand beside Propbot
6. Have one person walk away from the robot in increments of 20m and check the transmitter connection to receiver
7. Execute small movement to ensure connection
8. Repeat Steps 6-7 until Transmitter/Receiver pairing is lost

Figure 4.10 Maximum Transmitter Distance Test

4.3.6. Test Results

Test Tag	Pass	Notes
HT 4.3.11	Yes	Transmitter connection is lost at 300m This is the maximum distance at which the robot can be controlled in teleoperation mode

Table 4.18 Maximum Transmitter Distance Evaluation Results

4.3.6. Analysis

From this analysis it was discovered that the maximum distance for teleoperation is 300m. Distances exceeding this point result in the transmitter losing its connection to the receiver and therefore no longer being able to execute commands. For safe operation of the robot we have limited the operation distance to 100m and Propbot being clearly visible to the operator. Therefore this requirement is satisfied by our design choice of transmitter and receiver.

4.3.7. Transmitter Fail-safe

4.3.7. Target

To verify that the robot is stopped fully and turned off in the event that the transmitter is turned off, loses power, or loses pairing to the receiver (figure 4.11)

4.3.7. Evaluation Metrics

Test Tag	Requirement Tag	Requirement Description	Pass Criteria
HT 4.3.12	F3.6	The robot should only operate if an RC transmitter and receiver are paired	When the transmitter loses connection to the receiver or is powered off, the robot will halt all movements and enter E-Stop mode

Table 4.19 Maximum Transmitter Distance Evaluation Metrics

4.3.7. Safety Considerations

General safety precautions surrounding the high voltage equipment on Propbot and the heavy frame of the robot were observed. In addition, because these tests were conducted in the hallway we had spotters on either side of the corridor to ensure that it was clear of people.

4.3.7. Procedure

1. Ensure that manual E-Stop is engaged
2. Power on battery
3. Power on the transmitter and verify successful pairing with receiver
4. Disengage E-Stop
5. Execute Right hand turn
6. Turn off transmitter
7. Check if Propbot stops and enters E-Stop mode
8. Repeat for all other functions (ie: Go forward, Go backward, Left hands turn)

Figure 4.11 Transmitter Fail-safe

4.3.7. Results

Test Tag	Pass	Notes
HT 4.3.12	Yes	Propbot ceases all motion and enters E-Stop mode when transmitter/receiver pairing is lost or the transmitter is powered off

Table 4.20 Maximum Transmitter Distance Evaluation Results

4.3.7. Analysis

This validation verified that in the event that the transmitter loses pairing or power the robot will come to a complete halt. This provides reassurance and added safety awareness in the situation where the transmitter loses power.

4.4. Modes of Operation

4.4.1. Target

The ability to move and respond to obstacles in RC and Autonomy Mode, as well as Propbot's ability to transition between RC, Autonomy, and E-Stop mode using a Remote Controller (figure 4.12).

Figure 4.12 Modes of Operation

4.4.2. Evaluation Metrics

Test Tag	Requirement Tag	Requirement Description	Pass Criteria
IT 4.4.1	F1.2	The robot shall be able to move when in remote control mode or autonomy mode	Robot moves based on input from RC/autonomy
IT 4.4.2	F3.1, F3.2, F3.4	The robot shall be operable in autonomy mode and fall-back user mode. The fall-back user should be able to manually switch the robot operating mode	Transition between Autonomy Mode, RC mode, and E-Stop mode must occur when the fall-back user commands it
IT 4.4.3	F3.5	The robot shall send MCC a warning whenever it switches modes	Warning observed on MCC when Propbot switches modes
IT 4.4.4	NF3.1	Switching between modes should occur in no more than 0.5s	Switching between RC and autonomy mode should occur in no more than 0.5s

Table 4.21 Modes of Operation Evaluation Metrics

4.4.3. Safety Considerations

This test involves a high risk of collision and should be performed in a closed, safe, and controlled environment. It also involves dealing with high voltage and current electricity which can result in electrocution, burns, or even blinding. All safety procedures and mitigations outlined in the safety document must be followed.

4.4.4. Procedure

The setup is based on the mock autonomy setup, described in section 4.2.1. To complete the setup, the following steps can be followed:

1. Jack up Propbot, following the instructions in section 3.2.3
2. Ensure sure that the manual E-Stop is engaged
3. Power on the battery
4. Power on the transmitter and verify successful pairing with receiver
5. Disengage manual E-Stop
6. Connect the simulation, vehicle autonomy, and the mission command centre computer to the same local network. Do a simple ping test to verify bi-directional communication between each computer

7. On the vehicle autonomy computer, run: export ROS_MASTER_URI=http://Propbot@vehicle_autonomy_computer:11311
8. On the mission command centre computer, run: export ROS_MASTER_URI=http://Propbot@vehicle_autonomy_computer:11311
9. On the vehicle autonomy computer, launch the Test Client by running: roslaunch Propbot_tools mock_autonomy.launch
10. On the mission command centre computer, run: roslaunch Propbot_mission_gui mapviz.launch
11. Ensure that the logging console appears either as panel or a tab on the MCC window
12. Through the RC, put Propbot in RC mode
13. On the Test Client, enter the wheel speeds “105 105” to set Propbot to the highest speed when in Autonomy Mode

To run the test, conduct the following steps for RC and Autonomy mode:

1. Apply forward motion to Propbot in the given mode
    a. If in RC mode, use the remote controller
    b. If in Autonomy mode, input wheel speeds through the Test Client terminal in the form “x y”, where “x” is the desired left wheel speed, and “y” is the desired right wheel speed
2. Verify that Propbot’s wheels moved in response to 9 (IT 4.4.1)
3. Move object to within 0.5m of Propbot
4. Verify that
    a. Propbot stopped (IT 4.4.2)
    b. MCC displayed a warning about moving to E-Stop state (IT 4.4.3)
5. Through RC, return to the starting mode and Apply steps 9 and 10
6. Through RC, transition to the opposite mode (I.e if in RC, transition to Autonomy, and vice versa)
7. Verify that a warning was displayed on MCC (IT 4.4.3)
8. Apply steps 9 and 10 again
9. Through RC, return to starting mode
10. Apply smallest possible (1 km/s) forward motion to Propbot
11. Start Timer while transitioning to E-Stop mode
12. When Propbot comes full stop, stop timer
13. Verify that the time it took Propbot to come to a full stop, which represents an upper bound on the time it takes to transition between modes, is less than 0.5s (AT 4.4.4)

4.4.5. Results

Test Tag	Pass	Notes
IT 4.4.1	Yes	From a tester's perspective, it is easier to use a combination of very large speeds for autonomy mode, and small speeds for RC mode. This makes it easy to tell which input Propbot is propagating to the wheels.
IT 4.4.2	Yes	Change in mode can be observed through speed of wheels. Like in previous test, high speed = autonomy mode, small speed = RC mode, no speed = e-stop.
IT 4.4.3	Yes	Warnings appear in red colour
IT 4.4.4	Yes	Robot came to an immediate stop, no timer was necessary

Table 4.22 Modes of Operation Results

4.5. Robot Control in Autonomy Mode

4.5.1. Target

Section 4.4.2 showed Propbot's ability to receive and apply commands received from a mock autonomy package. Because the commands are generated by a user, they are not representative of the complexity of the motion Propbot will encounter on a real mission. Nor do they resemble the frequency at which the autonomy system outputs those commands.

In this test, we aim to show Propbot's ability to handle commands as they are output by the autonomy system. We are limited by the fact that the current state of Propbot is not equipped with LiDAR sensor to support the SLAM module of the autonomy package, nor is it equipped with Hall sensors to aid in the control module of the autonomy package. To overcome these obstacles, we use the integrated simulations setup, described in section 4.2.2, to derive control signals from Gazebo simulations and apply them to a real-life Propbot.

4.5.2. Evaluation Criteria

Test Tag	Requirement Tag	Requirement Description	Pass Criteria
IT 4.5.1	F1.1, F1.2, NF1.3	The robot shall be able to move when in autonomy mode The robot shall be able to move longitudinally when in autonomy mode The robot shall be able to execute locally generated control signals from the autonomy module	The robot is able to execute a mission, in the physical world, using sensor data obtained through simulation, without any collisions.

Table 4.23 Robot Control in Autonomy Mode Evaluation Criteria

4.5.3. Safety Considerations

This test involves a high risk of collision and should be performed in a closed, safe, and controlled environment. It also involves dealing with high voltage and current electricity which can result in electrocution, burns, or even blinding. All safety procedures and mitigations outlined in the safety document must be followed.

4.5.4. Procedure

To complete the Integrated Simulations setup described in Section 4.2.2, conduct the following steps:

1. Ensure you are testing in a 5m x 10m empty space
2. Ensure sure that the manual E-Stop is engaged
3. Power on the battery
4. Power on the transmitter and verify successful pairing with receiver
5. Disengage manual E-Stop
6. Move Propbot such that it is facing an obstacle (ex: chair, garbage can) five meters away from it
7. Connect the simulation, vehicle autonomy and the mission command center computer to the same local network. Do a simple ping test to verify bi-directional communication between each computer
8. On the simulation computer, run:
    a. export ROS_MASTER_URI=http://Propbot@vehicle_autonomy_computer:11311
9. On the vehicle autonomy computer, run:
    a. export ROS_MASTER_URI=http://Propbot@vehicle_autonomy_computer:11311

10. On the simulation computer, run:
    a. roslaunch **Propbot_simulation straigh_path_w_object_sim.launch**
11. On the vehicle autonomy computer, run:
    a. roslaunch Propbot_autonomy autonomy.launch
12. Ensure that the robot model appears in the simulation.
13. Transition to Autonomy Mode

To begin the test:

1. On MCC, set a waypoint at the exact longitude/latitude coordinates (49.513648, 48.12364)
2. This corresponds to a position that is exactly 8 meters away from Propbot's initial position in the simulation world
3. On MCC, upload the mission
4. On MCC, start the mission
5. Stop the test once one of the following happens
    a. Propbot collides into the object (IT 4.4.1 fails)
    b. Propbot leaves 5m x 10m space (IT 4.4.1 fails)
    c. Propbot is not within 0.5m of target destination (8m ahead of starting position) and MCC shows mission complete (IT 4.4.1 fails)
    d. Propbot is within 0.5m of of target destination (8m ahead of starting position) and MCC shows mission complete (IT 4.4.1 passes)

4.5.5. Test Results

Test Tag	**Pass**	**Notes**
IT 4.5.1	Yes	We find that while we are able to control the robot through the autonomy system, we are not able to execute complex actions (sharp turns, multiple turns). Having feedback from the robot's movement (either through wheel encoders, or Hall sensors like on Robotec's controllers) would greatly help.

Table 4.24 Robot Control in Autonomy Mode Results

4.6. Data Collection

4.6.1. Target

These tests are concerned with Propbot's ability to collect useful data that might aid in development, debugging or research.

4.6.2. Evaluation Metrics

Test Tag	Requirement Tag	Requirement Description	Pass Criteria
IT 4.6.1	F5.1	The robot shall collect time-stamped location data at a sampling rate of 1Hz	Time-stamped location data can be viewed from the file after test completion
IT 4.6.2	F5.2	The robot shall collect time-stamped data at a sampling rate of 1Hz for the motor speed for each motor	Time-stamped motor speed data can be viewed from file after test completion
IT 4.6.3	F5.3	The robot shall collect time-stamped data at a sampling rate of 1Hz for the command status from both the mission command center and manual control transmitter	Time-stamped command status and manual control transmitter data can be viewed from the file after test completion

Table 4.25 Data Collection Evaluation Metrics

4.6.3. Procedure

The procedure is split into two stages. Stage 1 validates IT 4.6.1 and 4.6.2, whereas stage 2 verifies IT 4.6.3

Stage 1:

The setup is based on the mocked autonomy setup, described in section 4.2.1. To complete the setup, the following steps can be followed:

1. Ensure sure that the manual E-Stop is engaged
2. Power on the battery
3. Power on the transmitter and verify successful pairing with receiver
4. Disengage manual E-Stop
5. Connect the simulation, vehicle autonomy, and the mission command centre computer to the same local network. Do a simple ping test to verify bi-directional communication between each computer
6. On the vehicle autonomy computer, navigate to the src/vehicle_autonomy/propbot_autonomy/launch/autonomy.launch file

7. **Set the variable “save_data” to true**
8. On the vehicle autonomy computer, run: export ROS_MASTER_URI=http://Propbot@vehicle_autonomy_computer:11311
9. On the vehicle autonomy computer, launch the Test Client by running: roslaunch Propbot_tools mock_autonomy.launch
10. On the mission command centre computer, run: roslaunch Propbot_mission_gui mapviz.launch

To run the test, conduct the following steps :

11. Switch to Autonomy Mode
12. Apply forward motion to Propbot in the given mode
    a. This is done by inputting wheel speeds through the Test Client terminal in the form “x y”, where “x” is the desired left wheel speed, and “y” is the desired right wheel speed
13. End running programs on autonomy computer and MCC computer
14. On Autonomy computer, navigate to ~/.ros directory
15. Find file with latest timestamp
16. Run the command “rosbag play filename.bag”
    a. this will publish the data that was captured
17. Run the command “rostopic /gps/fix”
    a. verify that GPS data is being published from the file (IT 4.6.1)
18. Run the command “rosotpic /left_wheel” and “rostopic /right_wheel”
    a. verify that motor speeds are being published from the file (IT 4.6.2)

Stage 2:

1. On the vehicle autonomy computer, navigate to the src/vehicle_autonomy/propbot_autonomy/launch/autonomy.launch file
2. **Set the variable “save_data” to true**
3. Run any simulated test from Section 2.2 using any scenario
4. At the end of the simulated test, on MCC computer, navigate to ~/.ros directory
5. Find file with latest timestamp
6. Run the command “rosbag play filename.bag”
    a. this will publish the data that was captured
7. Run the command “rostopic /mapviz/mission_command”
    a. verify that mission command data is being published from the file (IT 4.6.3)

4.6.4. Results

Test Tag	Pass	Notes

IT 4.6.1	Yes	The data is stored in a non-readable format so that it can be immediately published by rosbag. To convert the bag file into an excel sheet, the command “rostopic echo -b file.bag”
IT 4.6.2	Yes	The data is stored in a non-readable format so that it can be immediately published by rosbag. To convert the bag file into an excel sheet, the command “rostopic echo -b file.bag”
IT 4.6.3	Yes	The data is stored in a non-readable format so that it can be immediately published by rosbag. To convert the bag file into an excel sheet, the command “rostopic echo -b file.bag”

Table 4.26 Data Collection Results

5. References

1. UBC-RSL-PropBot, “UBC-RSL-PropBot/PropBot: Software and firmware stacks for the PropBot Autonomous Robot,” *GitHub*. [Online]. Available: https://github.com/UBC-RSL-PropBot/PropBot. [Accessed: 04-Apr-2022].
2. “Chapter 3 - GPS data collection ... - virginia tech.” [Online]. Available: https://vtechworks.lib.vt.edu/bitstream/handle/10919/33746/Chapter_3.pdf. [Accessed: 04-Apr-2022].
3. “Center,” *u*, 01-Apr-2022. [Online]. Available: https://www.u-blox.com/en/product/u-center. [Accessed: 03-Apr-2022].
4. “36V 26.5ah downtube battery,” *ebikes.ca*. [Online]. Available: https://ebikes.ca/36v-26-5ah-downtube-battery.html?___store=canadian&___from_store=international. [Accessed: 04-Apr-2022].
5. R. Hanrahan, “Testing a power supply – stability (part 3),” EDN, 26-Mar-2020. [Online] Available: https://www.edn.com/testing-a-power-supply-stability-part-3/. [Accessed: 14-Feb-2022].
6. “Software - roboteq,” Roboteq Software. [Online]. Available:https://www.roboteq.com/all-products/software. [Accessed: 14-Feb-2022].
7. “Brushless DC Motor Controllers: SBL1360A - roboteq.” [Online]. Available: https://www.roboteq.com/products/products-brushless-dc-motor-controllers/sbl1360-277-detail. [Accessed: 14-Feb-2022].

8. “Can networking manual? - génération robots.” [Online]. Available: https://www.generationrobots.com/media/roboteq/can-networking-manual.pdf. [Accessed: 14-Feb-2022].
9. B. Zuo, “Can-bus shield v2.0,” seeedstudio. [Online]. Available: https://wiki.seeedstudio.com/CAN-BUS_Shield_V2.0/. [Accessed: 14-Feb-2022].

A. Appendix

A.1. Gazebo Simulation Data

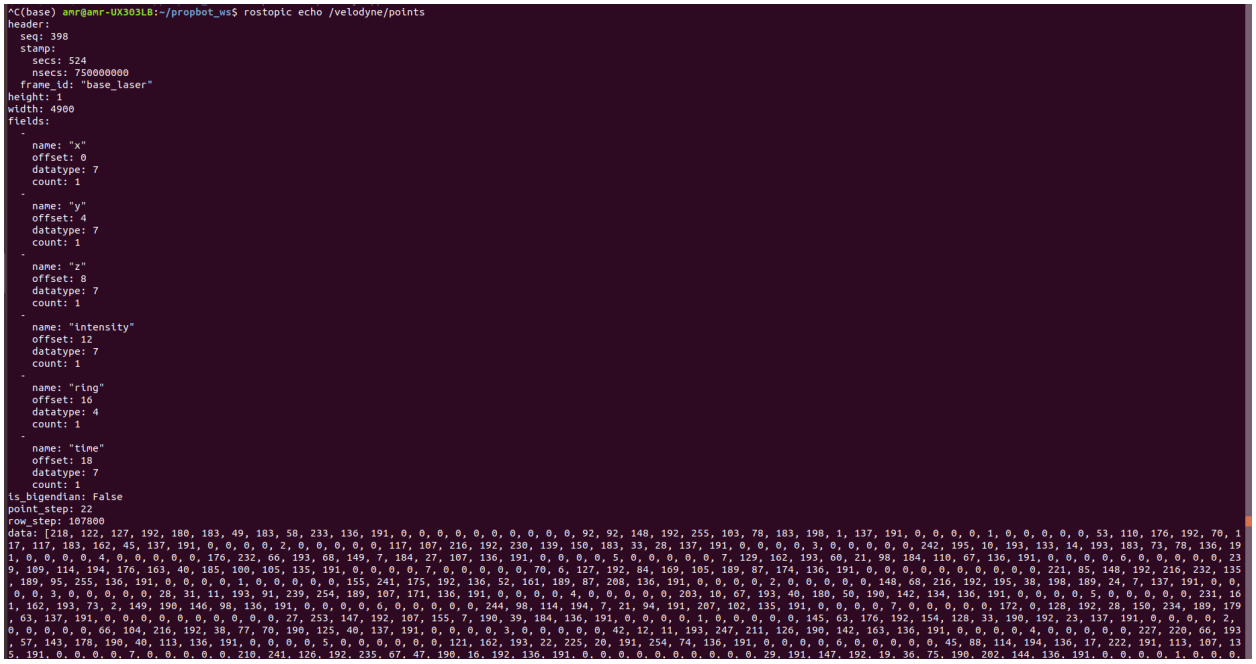

Figure A.1 Gazebo Simulation LiDAR Data

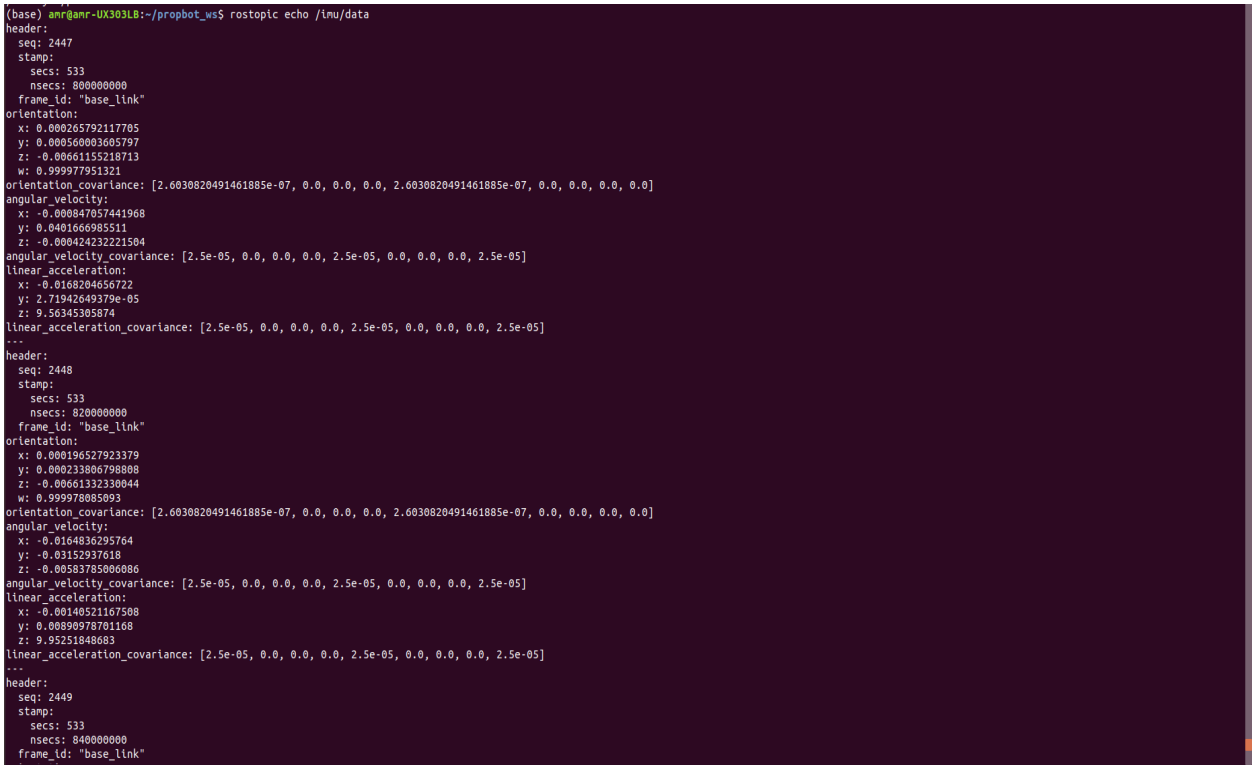

Figure A.2 Gazebo Simulation IMU Data

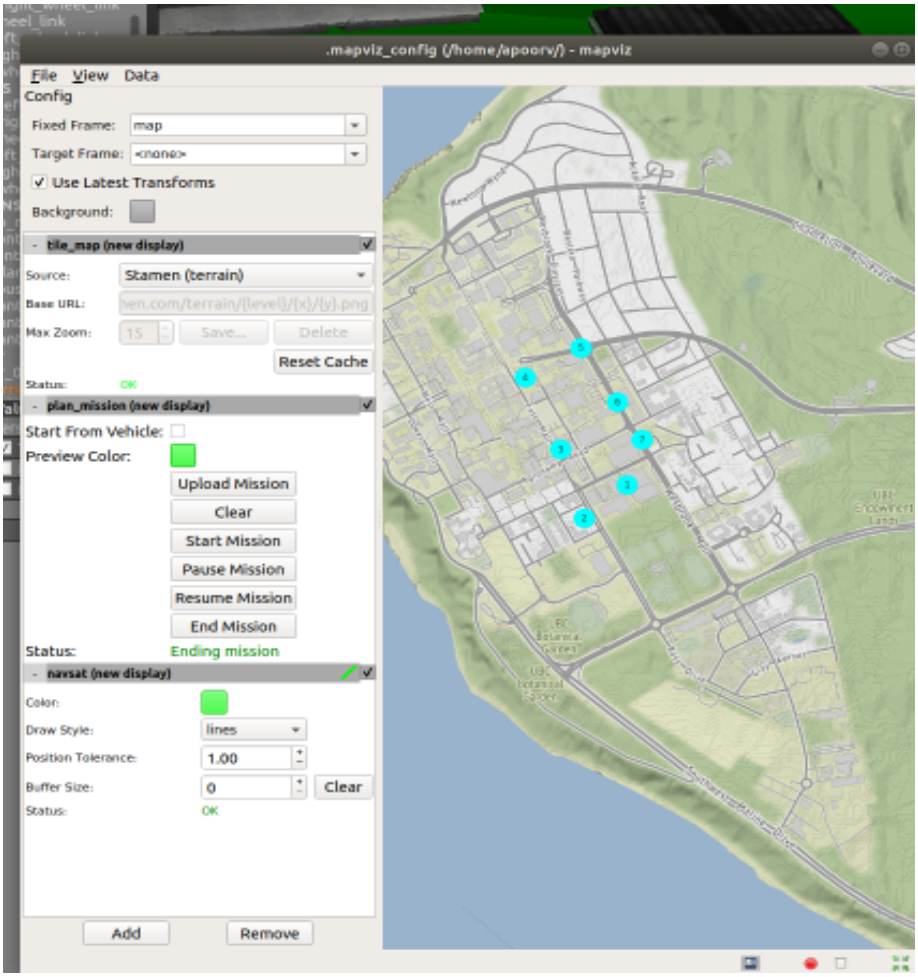

Figure A.3 Mission Command Centre GUI

A.2. Roboteq SBL1360A Motor Controllers

A.2.1. Introduction

This section provides instructions for setting up the Roboteq SBL1360A motor controller including power interfacing components and hardware necessary for creating a CAN network to communicate with the vehicle interface board (Arduino Mega 2560). It also provides test procedures for software configuration with the Roborun software [6].

A.2.2. Safety Considerations

All safety protocols relating to Propbot jacking mentioned in Section 3.1 are to be followed in this section as well. An additional safety consideration is to ensure that Roboteq controllers are wired correctly and contain all the various safety components as per the datasheet recommendations. This includes inline fuses, E-stop, flyback diodes, and precharge resistors. The main battery leads should be fitted to 12AWG to interface properly with the screw terminal connectors on the Roboteq controller. This is to prevent the potential for stray wires to touch and cause an electrical short by retrofitting the larger 10AWG battery cables to fit into the smaller Roboteq terminal. It is crucial that these safety requirements are properly followed due to the high voltage of the batteries that are used to power Propbot and its potential to cause serious personal harm if used improperly.

A.2.3. Hardware Interface Setup

This procedure outlines the hardware setup of the Roboteq controllers. For more details refer to the Roboteq SBLxx Datasheet [7].

1. Connect E-Stop, inline fuse, precharge resistor, and flyback diode as per figure A.4
2. Connect motor phase cables to screw terminal connections labelled U, V, and W as per figure A.5
3. Connect hall sensors to JST PH connector making sure that hall sensors match with the appropriate phase as per figure A.5
4. Connect the battery to Ground and positive to Vmot, ensuring that Ground is first connected

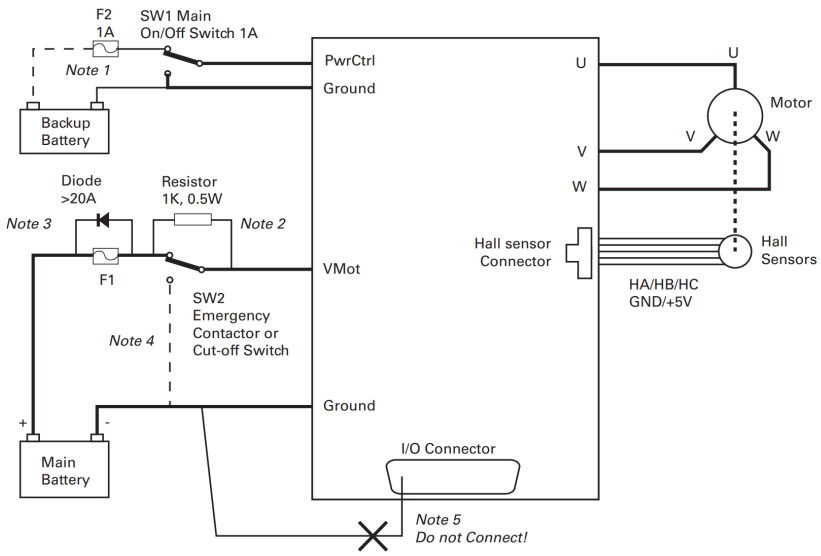

Figure A.4. Roboteq SBL1360A Wiring

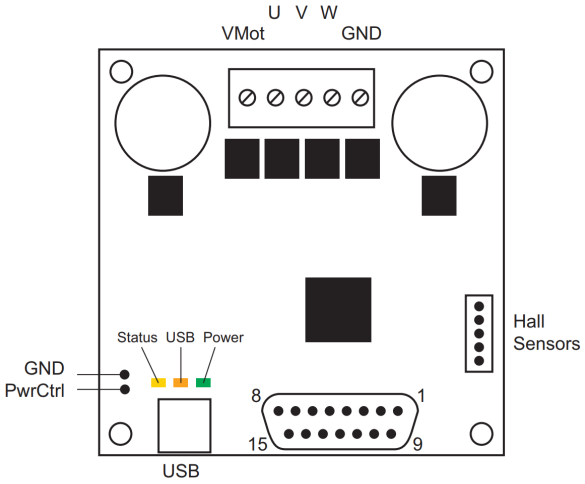

Figure A.5. Roboteq SBL1360A Connections

A.2.4. CAN Network Setup

This procedure outlines the steps for setting up the physical CAN network. Refer to Roboteq CAN datasheet for more details [8].

1. Connect DB-15 breakout to DB-15 connector
2. Connect both 120Ω terminating resistors across each end of parallel wires shown in figure A.6
3. Connect microcontroller to CANH (Pin 7 on DB-15 breakout - see figure A.7) and CANL (Pin 6 on DB-15 - see figure A.7)

4. Connect CAN analyzer cable to CANH and CANL nodes and computer

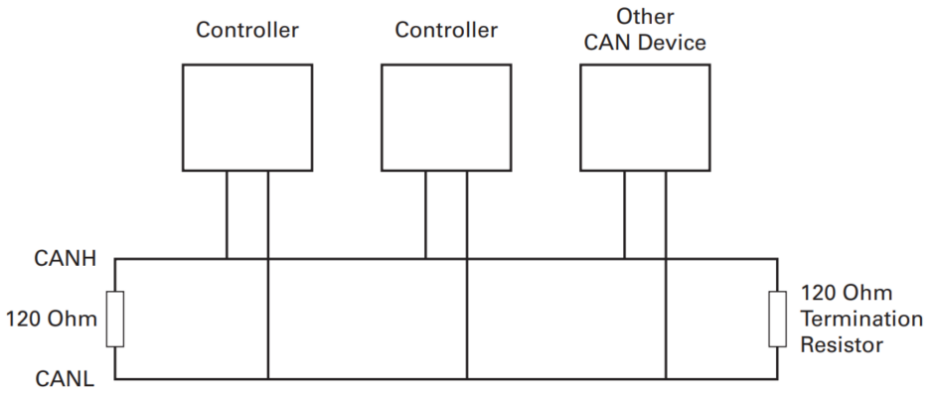

Figure A.6. CAN network

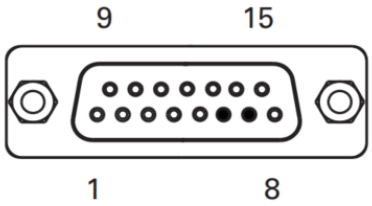

Figure A.7. Roboteq SBL1360A DB15 Pinout

A.2.5. CAN Shield Setup

This procedure outlines the steps for connecting the Seeed Studio V2.0 CAN shield with the vehicle interface computer. Refer to the datasheet for more detail on the CAN shield [9].

1. Insert CAN shield into a breadboard of appropriate size
2. Use jumper cables to connect Vcc, Gnd, CANH, and CANL pins to the Arduino Mega according to CAN Shield pinout in figure A.7
3. Connect DB9 breakout to DB9 connector
4. Wire up Arduino Mega with DB9 breakout according to figure A.8

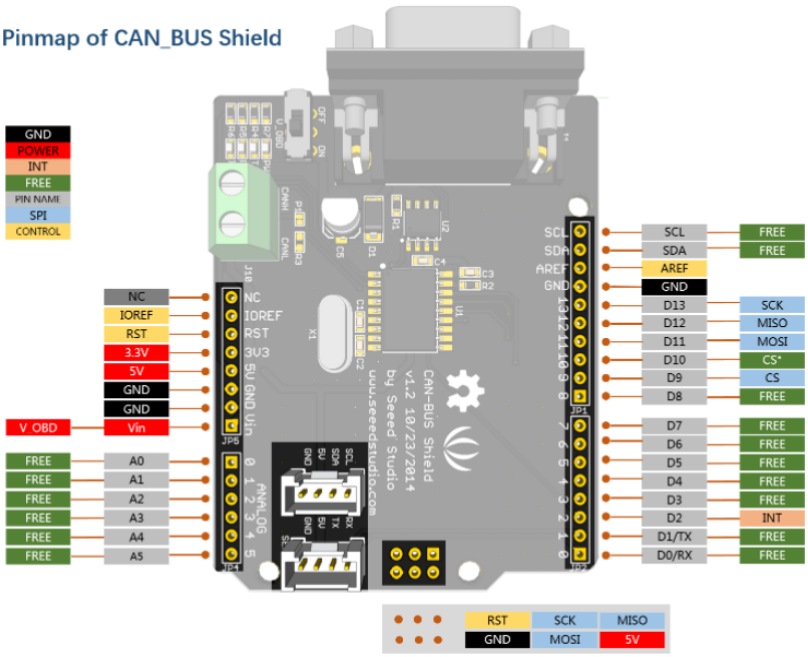

Figure A.7. Seeed Studio CAN Shield Pinout

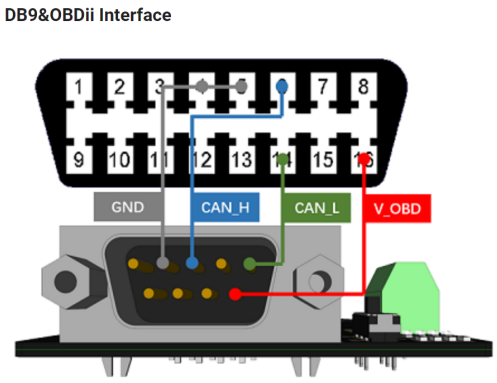

Figure A.8. Seeed Studio CAN Shield DB9 Connector Pinout

A.2.6. CAN Network Basic Validation

Below is a simple test to validate that the CAN network is sending and receiving data using a CAN network analyzer.

1. Ensure controller is supplied with power and is connected to the motor and hall sensors
2. Connect vehicle interface computer and motor controller to a CAN network

3. Send a speed command from the vehicle interface computer to the motor controller
4. Use a CAN network analyzer to record CAN activity on the bus

A.2.7. Motor Characterization and PID Tuning

The following procedure outlines the steps needed to perform the motor characterization and PID tuning automatically using Roborun software.

1. Ensure controller is supplied with power and is connected to the motor and hall sensors
2. Connect the controller to a computer installed with Roborun software using TTL-RS232 USB adapter
3. Launch Roborun application
4. Perform auto PID tuning of motor controllers from Roborun software