Validation Specification

Capstone PL-116

April 8th, 2020

Hannah Sawiuk (HS), 51196160

Divya Budihal (DB), 24512155

Yan Hong (YH), 31510150

Jack Guo (JG), 31373153

Zhaosheng Li (ZL), 43032168

Contents

Glossary

Automated Driving Systems (ADS) A system of software and hardware capable of performing DDT on a sustained basis. [1].

Autonomy-Processor-in-the-loop (APIL) testing A testing process in which simulated sensor data is fed into the autonomy algorithms. The loop is closed by the algorithms feeding information back to the simulation and having the simulation respond to the information..

Controller-Processor-in-the-loop (CPIL) testing A test methodology in which dynamic systems are integrated into the development by adding their mathematical representations..

Dedicated short-range communication (DSRC) One-way or two-way short-range to medium-range wireless communication channels specifically designed for automotive use and a corresponding set of protocols and standards..

Dynamic Driving Task (DDT) This term encompasses all of the real-time operational and tactical functions required to operate a vehicle. [2].

Dynamic Driving Task (DDT) This term encompasses all of the real-time operational and tactical functions required to operate a vehicle.

Experimenter Person who observes and directs the execution of each test trial. This role entails:

- Being knowledgeable of the operation of ADS feature under test
- Recording test conditions, test trial notes and judging test trial validity
- May operate data acquisition system and other test equipment

.

Extended Functional/Non-functional Requirement (EFR/ENFR) Given the broad scope of the PropBot initiative and the relevance of the requirements of the initiative to the narrowed scope of our project, functional and their non-functional requirement counterparts that are out of the scope of our contributions have been renamed to "extended" functional and non-functional requirements (abbr: EFR and ENFR).

National Highway Traffic Safety Administration (NHSTA) The National Highway Traffic Safety Administration is an agency of the U.S. federal government, part of the Department of Transportation.

Object and Event Detection and Response (OEDR) The action of detecting an object or event near the robot and responding accordingly. This specifically refers to the subtasks of the DDT that include monitoring the driving environment. [2].

Object and Event Detection and Response (OEDR) The action of detecting an object or event near the robot and responding accordingly. This specifically refers to the subtasks of the DDT that include monitoring the driving environment [2].

Operational Design Domain (ODD) Describes the operating domains in which the system is designed to function. ODD examples include speed range, lighting conditions, weather conditions, roadway types, etc. [2].

Operational Design Domain (ODD) Describes the operating domains in which the system is designed to function. ODD examples include speed range, lighting conditions, weather conditions, roadway types, etc. [2].

Operational Scenario The environment in which a feature is designed to function, and it is described by a set of ODDs [2].

Pedestrian Surrogate (PS) Human surrogate that is attached to a mobile base and has similar mobility and detection characteristics (visual appearance, radar and/or lidar reflectivity etc.).

Pedestrian Surrogate Operator (PSO) Person responsible for positioning and controlling the pedestrian surrogate. This role entails (1) being knowledgeable PS construction and mobility and (2) positioning and Operating surrogate for test trials.

Point Of Interest (POI) Specific location of interest. Within the context of path-planning, a POI is a waypoint that the path must include. Within the context of propagation data collection, a POI is the point where measurements are to occur.

Software-in-the-loop (SIL) testing An electronic device that measures and reports a body's specific force, angular rate, and sometimes the orientation of the body, using a combination of accelerometers, gyroscopes, and sometimes magnetometers. [3].

Subject Vehicle (SV) Vehicle under test.

Subject Vehicle Operator (SVO) Person responsible for positioning and controlling the Subject Vehicle. This role entails (1) being familiar with the activation and deactivation of ADS feature and potential failure modes and (2) disengaging the ADS feature under test and bringing the vehicle back to minimal risk state if the experiment approaches or reaches unsafe state..

Vehicle-in-the-loop (VIL) testing A test methodology that deals with in-vehicle tests. These tests require a prototype to integrate the developed system. Three approaches are commonly used are test-drives on test-tracks, test-drives on open-roads, and Vehicle-Hardware-In-the-Loop (VeHIL). [4].

Vehicle-to-everything(V2X) The passing of information from a vehicle to any entity that may affect the vehicle, and vice versa. It is a vehicular communication system that incorporates other more specific types of communication as V2I, V2N, V2V, V2P, V2D and V2G..

World Geodetic System (wgs84) The standard coordinate system referenced by Global Positioning System (GPS). [5].

1 Overview of Validation Tests

The validation tests consists of two major parts, the simulation tests in software and the hardware tests on the robot or a placeholder mini RC (Remote Control) car.

1.1 Simulation Tests

Table 1.1 is an overview of all the simulation tests with the requirements and test status.

Requirement Tag	Requirements Description	Test Tag	✓
F1.1	Robot shall drive through user-generated waypoints on campus.	ST1.1	✓
F1.2	Robot shall not collide with any obstacles along the route.	ST1.2	✓
F1.3, NF1.3	Robot shall stop at predefined points of interest.	ST1.3	✓
NF2.1	Robot must maintain a safe minimum euclidean distance of at least 1m from objects at all times	ST1.4	✓
F3.1	Robot shall record time-stamped position data.	ST1.5	✓
F3.2	Robot must record time-stamped position data at 1Hz.	ST1.6	✓
F4.4	Mission command shall communicate with the robot over a local network	ST1.7	✓
F2.1	Robot shall perform go-around maneuver when it detects the ODD static object in frontal zone	ST2.1	✓
F2.2	Robot shall stop if it cannot determine a feasible trajectory given the surrounding objects.	HT2.3	✓
NF1.1	Robot must create a path from user-generated waypoints that minimizes distance while not leaving approved operation zones.	ST2.2	✓
F4.1	Mission command centre shall receive operational status of the robot (location coordinates, and route progress) of the robot	ST3.1	✓
F4.2	Mission command centre shall have the ability to start and stop robot mission at all times.	ST3.2	✓
NF4.1	Speed commands must cease within 1 second of the assertion of "stop" on the mission command GUI	ST3.3	✓

Table 1.1: Summary of Simulation Tests

1.2 Hardware Tests

Table 1.2 shows a summary of all the hardware tests with respective requirements and the status of each test.

Requirement Tag	Requirements Description	Test Tag	✓
F5.2	The fallback ready user shall be able to command 0 speed to the robot via physical e-stop button on robot.	HT2.1	✓
F5.3	The manual control device shall allow the user to control both longitudinal sets of wheels on the robot independently.	HT2.2	✓
F5.4	The manual control device shall allow the user to send 0 speed to the robot.	HT2.3	✓
NF5.1	Any changes in the robot operation must execute within 1 second of the command assertion (estop, mode switch)	HT2.4	✓
NF1.2	Robot batteries must last for at least 1.5 hours during route navigation.	HT3.1	*
F3.1	Robot shall record time-stamped position data.	HT4.1	*
NF3.1, NF3.2	Robot must record position that is accurate to at least 0.5m.	HT4.2	*
F5.1	The fallback ready user shall be able to kill power to the system via power-off switch on robot.	HT1.1	✓

Table 1.2: Summary of Hardware Tests

* Due to constraints introduced by the COVID-19 situation, a set of the hardware tests could not completed as they require outdoor testing.

2 Simulation Tests

Given that the focus of our contributions is the autonomy package, the majority of the requirements can be validated via simulation. The follow section is an overview of all the validation tests that can be completed in simulation.

Test Setup

The following tests make use of the files `mapviz.launch`, `autonomy.launch` and `ubc_student_union_sim.launch` to run our mission command GUI, autonomy stack, and simulation stack respectively. Each file is launched on a separate computer in order to validate the bi-directional communication between the mission command center and vehicle autonomy computer, as well as to dedicate the entirety of the Jetson TX1's processing power to the execution of autonomy stack. The interactions between the three computers are illustrated in figure 2.1.

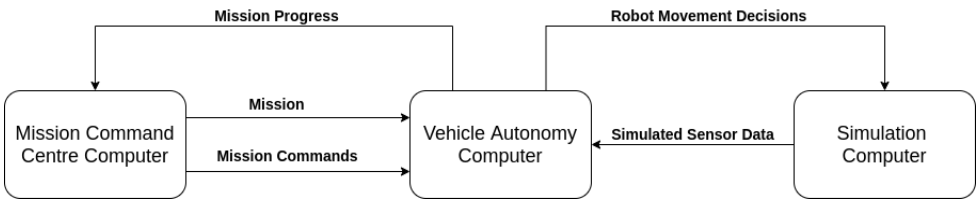
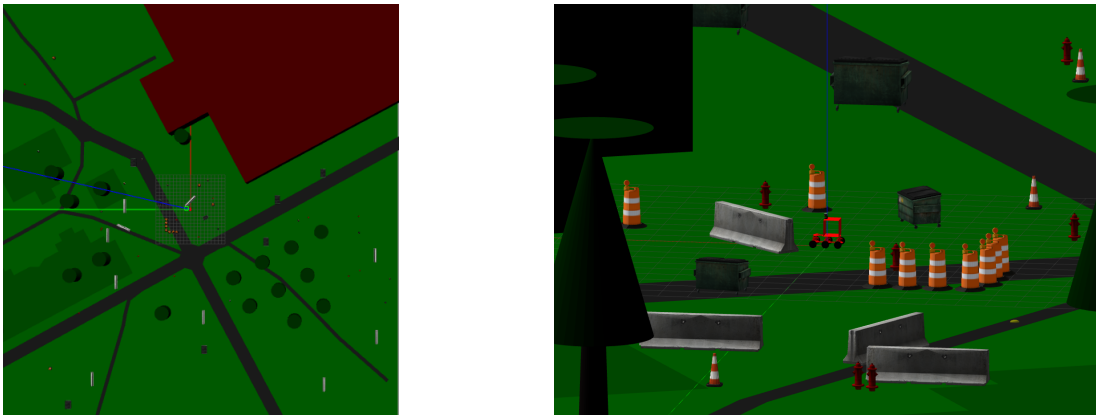


Figure 2.1: Testing setup for Simulation Validations

The simulation environment launched by `ubc_student_union_sim.launch` is a reconstruction of an area near the UBC Student Union Building in figure 2.2 bounded by a red box. The corresponding Gazebo 3-D simulation environment is seen in figure 2.3 and it contains addition static obstacles to create a more realistic and challenging environment for the robot to navigate.

Figure 2.2: Google Earth rendering of the UBC Student Union Area (simulated environment bounded by the red box)



(a) Bird's eye view

(b) Closeup Orthographic view

Figure 2.3: Gazebo Simulation of the UBC Student Union Area

2.1 Multiple Waypoint Mission

This test validates the robot's ability to receive multi-waypoint missions and complete them in static environments while meeting some driving, obstacle avoidance and data collection requirements.

2.1.1 Evaluation Metrics

Test Tag	Requirement Tag	Requirement Description	Pass Criteria
ST 1.1.	F1.1	Robot shall drive through user-generated waypoints on campus.	Robot must follow the path generated from the defined waypoints
ST 1.2.	F1.2	Robot shall not collide with any obstacles along the route.	Robot model completes the mission without colliding with any obstacles in the simulation.
ST 1.3.	F1.3, NF1.3	Robot shall stop at predefined points of interest.	Robot stops at each mission waypoint within 2m from target location.
ST 1.4.	NF2.1	Robot must maintain a safe minimum euclidean distance of at least 1m from objects at all times.	The centre of the robot model does not traverse into the light blue regions in the Rviz simulation which indicates the areas that are within a 1m radius of any obstacle.
ST 1.5.	F3.1	Robot shall record time-stamped position data.	Time-stamped position data is published to the rostopic "/navsat/fix" ROS topic in World Geodetic System (WGS84) format.
ST 1.6.	NF3.2	Robot must record time-stamped position data at 1Hz.	Time-stamped position data is published to the rostopic "/navsat/fix ROS topic in wgs84 format every second.
ST 1.7.	F4.4	Mission command shall communicate with the robot over a local network	Communication with the mission command must be successful

Table 2.1: Multiple waypoint mission test evaluation metrics

2.1.2 Procedure

The following steps demonstrate how to conduct the multiple waypoint mission test.

1. Connect the simulation, vehicle autonomy and the mission command centre computer to the same local network. Do a simple ping test 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 geographic coordinate as the robot model in the simulation.
9. Set 3 waypoints 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. Ensure that the robot model moves on both the mission command centre GUI and in the simulation. If this is true, ST1.7 passes.
12. On the vehicle autonomy computer, run:

    `rostopic echo /navsat/fix`

13. Check that the robot publishes its location data in wgs84 in the "/navsat/fix" topic rosbag. If it does, ST1.5 passes.
14. On the vehicle autonomy computer, run:

    `rostopic hz /navsat/fix`.

15. Check the average output frequency rate. If it is 1Hz or greater, ST1.6 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, ST1.3 passes.
18. End trial when:

    - Mission is complete. (ST1.1, ST1.2, ST1.4 pass)
    - Robot model collides with an obstacle
    - The Center of the Robot model traverses into the light blue regions

2.1.3 Test Results

Test Tag	Pass	Notes
ST1.1	✓	robot was able to reach all 3 waypoints set within the "ubc$_s tudent_u nion$"map
ST1.2	✓	The center of the robot stayed outside of the "inflated" light blue regions of the obstacles
ST1.3	✓	robot stopped less than 0.5m from each target waypoint
ST1.4	✓	The center of the robot stayed outside of the "inflated" light blue regions of the obstacles
ST1.5	✓	The rostopic /navsat/fix contained gps position data in WGS84 format
ST1.6	✓	This validates our software stack's ability to publish GPS position data at 1Hz or more, however, this criteria should be re-validated by future teams with real hardware
ST1.7	✓	The mission command center computer was able to communicate with the vehicle autonomy computer to upload and command missions. This was observed in printed output in the console of the vehicle autonomy computer as well as in the 3D simulation of the simulation computer.

Table 2.2: Multiple waypoints test results

2.2 Routing Simulation

This test validates the robot's ability to detect static obstacles and respond accordingly. The robot is expected to route around most static obstacles, however if a feasible route cannot be determined the robot is expected to stop and alert the user.

2.2.1 Evaluation Metrics

Test Tag	Requirement Tag	Requirement Description	Pass Criteria
ST 2.1.	F2.1	Robot shall perform go-around maneuver when it detects the ODD static object in frontal zone.	Robot does not run into a static obstacle in front of it, but rather builds a route around it.
ST 2.2.	NF1.1	Robot must create a path from user-generated waypoints that minimizes distance while not leaving approved operation zones.	The robot's route does not contain unnecessary detours or loops.
ST 2.3.	F2.2	Robot shall stop if it cannot determine a feasible trajectory given the surrounding objects.	The robot shall stop and report back a mission failure status to the user.

Table 2.3: Routing test evaluation metrics

2.2.2 Procedure

The following procedure lists how to conduct the routing test.

1. On the simulation computer, run:

   ```
   export ROS_MASTER_URI=http://propbot@vehicle_autonomy_computer:11311
   ```

2. On the vehicle autonomy computer, run:

   ```
   export ROS_MASTER_URI=http://propbot@vehicle_autonomy_computer:11311
   ```

3. On the mission command centre 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. On the mission command centre computer, run:

   ```
   roslaunch propbot_mission_gui mapviz.launch
   ```

7. Check that the robot model appears in the mission command centre GUI at the same geographic coordinate as the robot model in the simulation.

8. Set 3 waypoints on the mission command centre GUI. The first waypoint should be in an open area that the simulated robot can reach. The next should be placed such that the robot must go around an obstacle like a building. The last waypoint should be placed at an unreachable point, such as inside of a building. Click the *Upload Mission* button.

9. Click the *start mission* button on the mission command centre GUI
10. Observe if the robot reaches the first waypoint successfully following the route. If the robot's route does not contain loops or unnecessary deviation, ST2.2 passes.
11. Observe if the robot re-routes around the building and reaches the second waypoint. If the robot does not collide with anything and successfully goes around the building, ST2.1 passes.
12. Observe if the robot stops in front of the unreachable waypoint and if the message "Robot was unable to reach waypoint number %i... Stopping Mission..." is printed in the console of the vehicle autonomy computer. If it does, ST2.3 passes.
13. End trial.

2.2.3 Test Results

Test Tag	Pass	Notes
ST2.1	✓	The robot was able to route around each obstacle which was between its current and goal position. The robot's planned trajectory could be observed as a green line when enabling "Global Plan" in Rviz, and its short term trajectory which takes into account its kinematics could be observed as a red line when enabling "Local Plan" in Rviz. Both of these planned routes did not traverse the "inflated" light blue regions each obstacle.
ST2.2	✓	The robot's "Global Plan" seems to route around each obstacle but remained direct in free space.
ST2.3	✓	The expected output was observed in the console of the vehicle autonomy computer and the robot stopped in the 3D simulation of the the simulation computer.

Table 2.4: Object detection test results

2.3 Mission Command Centre

This test validates the bi-directional communication between the mission command centre and the robot. From the mission command centre GUI, the user should be able to upload a multi-point mission to the robot, send mission commands to the robot (start, pause, resume or end the mission at anytime during the mission), and monitor the mission progress and robot location.

2.3.1 Evaluation Metrics

Test Tag	Requirement Tag	Requirement Description	Pass Criteria
ST 3.1.	F4.1	Mission command centre shall receive operational status of the robot (location coordinates, and route progress) of the robot (location coordinates, and route progress).	Mission Command Centre GUI displays the robot's traversed path as the robot executes the mission in the simulation.
ST 3.2.	F4.2	Mission command centre shall have the ability to start and stop robot mission at all times.	Robot moves in the simulation after clicking *Start Mission* or Resume Mission button on the mission command GUI; Robot stops in the simulation after clicking *Pause Mission* or *End mission* on the mission command GUI.
ST 3.3.	NF4.1	Speed commands must cease within 1 second of the assertion of the End Mission button on the mission command GUI.	Robot stops in the simulation within 1 second after clicking the *Pause Mission* or *End Mission* on the mission command GUI.

Table 2.5

2.3.2 Procedure

The following procedure shows the method to test the functionality of the mission command centre.

1. On the simulation computer, run:

   ```
   export ROS_MASTER_URI=http://propbot@vehicle_autonomy_computer:11311
   ```

2. On the vehicle autonomy computer, run:

   ```
   export ROS_MASTER_URI=http://propbot@vehicle_autonomy_computer:11311
   ```

3. On the mission command centre 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. On the mission command centre computer, run:

   ```
   roslaunch propbot_mission_gui mapviz.launch
   ```

7. Set 3 waypoints on the mission command centre GUI and click the *Upload Mission* button

8. Click the *Start Mission* button on the mission command centre GUI
9. Observe if the robot model move on both the mission command centre GUI and in the simulation.
10. Observe if the robot's traversed route marked on the mission command centre GUI. If this movement is the same, ST3.1 passes.
11. Click the Pause Mission button on the mission command GUI.
12. Observe if the robot model stops on both the mission command centre GUI and in the simulation.
13. Click the *Resume Mission* button on the mission command centre GUI.
14. Observe if the robot model resumes movement on both the mission command centre GUI and in the simulation.
15. Click *End Mission* button on the mission command centre GUI. Start a stopwatch.
16. Observe if the robot model stops on both the mission command centre GUI and in the simulation. Stop the stopwatch when the robot stops. If the robot stopped within 1 second and the message "Mission is finished" is printed in the console of the autonomy computer, ST3.3 passes.
17. Observe if the mission command centre GUI is cleared (i.e. mission waypoints, traveled route, mission route disappears). If all of these conditions are met, ST3.2 passes.
18. End trial.

2.3.3 Test Results

Test Tag	Pass	Notes
ST3.1	✓	The robot's traversed path is shown as a red line in GUI
ST3.2	✓	The robot responded accordingly to the mission commands. This was observed in the printed output of the vehicle autonomy computer as well as the 3-D simulation of the simulation computer.
ST3.3	✓	The robot responded within 1 second to the commands Pause and End Mission. This was observed in the printed output of the vehicle autonomy computer as well as the 3-D simulation of the simulation computer.

Table 2.6: Power-off switch test results

3 Hardware Tests

Although the majority of the autonomy features can be validated in simulation, a number of safety features along with some autonomy features can be validated in real-world scenarios. The following is an overview of the validation tests involving hardware.

3.1 Power-Off Switch

This test is to validate whether the power-off switch kills all the power to the system without any leakage current. This is important in terms of both safety as well as power efficiency.

3.1.1 Evaluation Metrics

Test Tag	Requirement Tag	Requirement Description	Pass Criteria
HT 1.1.	F5.1	The fallback ready user shall be able to kill power to the system via power-off switch on robot.	Power must be cut to all devices. No leakage current.

Table 3.1: Power-off switch test evaluation metrics

3.1.2 Procedure

The following procedure is about how to test the power-off switch.

1. Measure the battery voltage via probe port and mark down the voltage.
2. Confirm that the power-off switch is in the "OFF" position.
3. With the key in the "OFF" position, connect the battery bank to the robot. Then turn the key to the "ON" position to allow current flow.
4. Using a multimeter, check the inputs of each of the DC/DC converters to confirm that no power is being delivered (confirms no shorts and that the switch can appropriately block current flow).
5. Switch the power-off switch to the "ON" position.
6. Using a multimeter, check the inputs of the DC/DC converters and ensure that the voltage matches that of the battery voltage measured in the first step.
7. Ensure that the power-off switch is in the "OFF" position. Then, confirm that no voltage appears at the input of the DC/DC converters.

3.1.3 Test Results

Test Tag	Pass	Notes
HT1.1	✓	No leakage current was detected

Table 3.2: Power-off switch test results

3.2 Manual Remote Control

These tests are to validate the functionality of the manual remote control. All the tests below can be performed on either the robot or a placeholder mini RC car. Using the RC controller, the user should be able to switch control modes from manual to autonomy, stop the robot's movement asynchronously, and independently control the longitudinal sets of wheels.

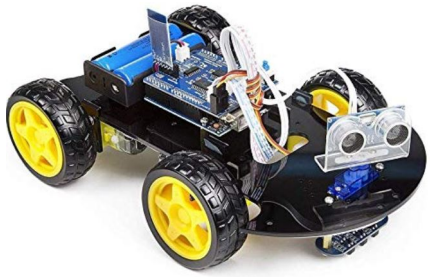


Figure 3.1: Placeholder mini RC car for testing [6]

3.2.1 Evaluation Metrics

Test Tag	Requirement Tag	Requirement Description	Pass Criteria
HT 2.1.	F5.2	The fallback ready user shall be able to command 0 speed to the robot via physical e-stop button on robot.	Both an indication of 0 speed (e.g. PWM signal) and the slowing and halting of wheel motion should occur.
HT 2.2.	F5.3	The manual control device shall allow the user to control both longitudinal sets of wheels on the robot independently.	The left and right joysticks on the remote controller must only command their corresponding wheel side (forward, backward, and coast).
HT 2.3.	F5.4	The manual control device shall allow the user to send 0 speed to the robot.	Both an indication of 0 speed (e.g. PWM signal) and the slowing and halting of wheel motion should occur.
HT 2.4.	NF5.1	Any changes in the robot operation must execute within 1 second of the command assertion (e-stop, mode switch).	Asserted actions must occur within 1 second of the assertion.
HT 2.5.	F5.5	The manual control device shall allow the user to switch between manual and autonomous driving modes.	The wheels of the RC car stops spinning when the it is switched to manual mode.

Table 3.3: Remote Control test evaluation metrics

3.2.2 Procedure

The following procedure aims to test the 0 speed command from the manual remote control, which corresponds to HT2.1, HT2.3, and HT2.4:

1. Setup an oscilloscope to measure the PWM signals to the motor drivers.
2. Via serial interface or RC, set the wheel speeds to a non-zero value.
3. After several wheel rotations, start a timer assert the 0 speed command by the method under test.
4. Once confirmed on the oscilloscope that the duty cycles are all set to 0, stop the timer.
5. Confirm that the execution is below the specified threshold of 1 second.
6. End trial when: all wheels stop rotating.

The following steps aim to test whether users can use the RC to control the two sets of wheels independently, which corresponds to HT2.2:

1. Setup the system for RC control.
2. Use one side of the joysticks to send forward speed commands, backward speed commands, and neutral speed commands (centered). Confirm that the joystick's corresponding longitudinal wheel set responds and that the other set remains stationary.
3. Repeat for the reverse side.
4. End trial when: both sides have been tested.

The following steps aim to test whether the mode switch is working properly, which corresponds to HT2.5:

1. Setup the system for RC control.
2. Prop mini RC car up on a wooden so its wheels are not in contact with a surface.
3. Connect autonomy computer with t.he RC car vehicle interface.
4. Toggle the autonomy/manual switch on the manual control device to enable autonomous mode.
5. On the autonomy computer begin HIL test by running:

   ```
   roslaunch propbot_demos propbot_mission.launch
   ```

6. Observe that the wheels of the RC car turning to indicate that it is executing a mission in autonomy mode.
7. Toggle the autonomy/manual switch on the manual control device to enable manual mode.
8. Observe that the wheels of the RC car stop to indicate that is is in manual mode. If this is true 2.5 passes.
9. End trial.

3.2.3 Test Results

Test Tag	Pass	Notes
HT2.1		Not completed due to project's time constraint
HT2.2	✓	Longitudinal wheel sets responded independently.
HT2.3	✓	Wheels successfully entered a locked "coast" mode after engaging the remote e-stop.
HT2.4	✓	Wheels stop spinning immediately, after triggering the e-stop or the mode switch.
HT2.5	✓	Wheels stop spinning immediately and the RC car smoothly switch to manual mode, after pushing up the mode switch.

Table 3.4: Remote Control test results

3.3 Lithium Ion Battery

This test requires a lot of set up. It will not be completed within or project's time frame, and it has been included for future reference. But we still make a power budget table and include that table in the *Design Document*. The power budget in the *Design Document* is based on the requirement that the robot needs to be operating for 1.5 hours consistently. Further, due to the constraints arising from the COVID-19 situation, we were unable to access equipment and test outside.

3.3.1 Evaluation Metrics

Test Tag	Requirement Tag	Requirement Description	Pass Criteria
HT 3.1.	NF1.2	Robot batteries must last for at least 1.5 hours during route navigation.	Battery bank must supply power to the system for 1.5 hours of fully engaged autonomy (all modules and sensors used).

Table 3.5: Robot battery test evaluation metrics

3.3.2 Procedure

The following procedure is about how to test the duration of the robot battery.

1. Setup Propbot for autonomous driving: autonomy computer, sensor suite, vehicle interface, etc.
2. Attach a battery monitor to the bank to relay the voltage readings to a secondary device which logs the data to an external storage device.

3. Add on additional weighted items to the equipment rack to act as a placeholder for the measurement equipment.
4. Drive the robot a varying speeds for up to 30 minutes using the manual controller.
5. End trial when: the 30 minutes are up or the batteries die. Alternatively, the robot could be driven around for 1.5 hours.
6. In post-processing, use the collected voltage readings to characterize the robot's power consumption and relate to the battery capacity to determine if 1.5 hours of operation can feasibility be achieved with the current battery bank.

3.3.3 Test Results

Test Tag	Pass	Notes
HT3.1		Not completed due to project's time constraint

Table 3.6: Robot battery test results

3.4 Data Collection

For the purposes of this test, either the actual robot outfitted with all of the appropriate sensors can be used, or a "cart" that bears all of the sensors (with the static transformations appropriately configured on the vehicle autonomy computer). This cart is a surrogate test bed for the robot.

3.4.1 Evaluation Metrics

Test Tag	Requirement Tag	Requirement Description	Pass Criteria
HT 4.1.	F3.1	Robot shall record time-stamped position data.	Data must include UTC time stamps with no drift.
HT 4.2.	NF3.1, NF3.2	Robot must record position that is accurate to at least 0.5m.	Recorded data from the collection test recorded position data at 1Hz that is timestamped and accurate to 0.5m. This is calculated by comparing the recorded position data and the known coordinates.

Table 3.7: Data collection test evaluation metrics

3.4.2 Procedure

The following steps demonstrate how to conduct the data collection test.

1. The autonomy computer, sensor suite and GPS must be mounted to either the robot or on a cart.
2. Find a marked location (one that has exact GPS coordinates). Google Maps is accurate to 6 decimal points which translates to 0.11m accuracy.

3. Bring the cart/robot to the known location. Once at the location, an external device should be used to mark the start and end time of the test.
4. Run SLAM on the autonomy computer and being data collection.
5. Move away from the location at 0.5m increments (use a measuring tape) and collect data at each position for at least 10 seconds.
6. At each increment, also record the start and end time and coordinates manually using an external device (e.g. phone or online timer).
7. In post-processing:
    - Validate that the time-stamped position data was recorded once per second by checking the data point count against the recording length.
    - Validate that the time stamps match those recorded with an external device.
    - Validate that the GPS coordinates match the coordinates of the ground truth location.

3.4.3 Test Results

Test Tag	Pass	Notes
HT4.1		Not completed due to COVID-19 constraints
HT4.2		Not completed due to COVID-19 constraints

Table 3.8: Data collection test results

Appendix A Hardware and Sensor Validation

Each component is validated prior to integration. The following sections describe the hardware and sensor validation processes.

A.1 Wheel Motors

The current motors are untraceable, and as such, the various characteristics needed to accurately control the robot should be determined To determine the speed constant of the motors, the duty cycle of the PWM drive signals is set and the rotational speed is measured using a tachometer.

Procedure

1. Set the control signal duty cycle.
2. Measure the rotational speed with a tachometer.
3. Repeat step 1 and 2 for multiple duty cycles.
4. Determine an average value for the speed constant.

Next, using the equation shown below, the torque constant can be determined. Knowing the torque constant will allow us to better model the system's acceleration capabilities, which is necessary to have insight into the robot's braking ability.

$$K_T \left[\frac{lb-in}{amp}\right] = \frac{K_e}{0.000684} \left[\frac{volts_{(line-to-line)}}{rpm}\right] \qquad [7] \qquad (1)$$

A.2 Lidar

There are many example programs testing basic imaging to object detection available for camera hardware. A simple test to ensure the camera is working correctly is to run example software and ensure that the performance matches marketed results.

A.3 Ultrasonic Sensors

The ultrasonic sensors are validated by recording date while pointing the sensor at commonplace items such as people, cars, bicycles, and walls. The tests are performed at various angles to measure performance of the sensor in abnormal situations. A close range test is also performed to validate that the sensor has reasonable outputs or discernible outputs when outside its operating range.

For software development and validation, we use Github and the Continuous Integration features included with it. Our workflow is similar to the standard Gitflow, but with multiple tiers of development branches. These development branches are based on the level of testing performed on the code. As mentioned in the simulation overview we will perform testing with different levels of simulation so the branches/stages will be named like so:

Appendix B Software Development and Validation Process

1. **pull request/testing**: Changes in a pull request. Automated build and tests run.

2. **develop**: Branch to merge to for introducing new features, any feature branches branch off of here.
3. **sil-testing**: Software in loop testing with software simulation.
4. **pil-testing**: Processor in loop with simulated sensors and simulated vehicle.
5. **vil-testing**: Vehicle in loop testing with simulated sensors, simulated vehicle, and all real motor and processor hardware.
6. **drive-testing**: Software in full (closed track) driving tests.
7. **master**: The final production code in which passes all testing to ship to client.

Any new features are developed by first branching off of develop. A pull request is then made for this branch and needs to be reviewed by two team members before merging. The code is then merged down through the branches providing it passes the tests in that stage.

Automated testing in the testing, sil-testing, and cpil-testing stages is used. For basic build and unit tests, tools like Travis CI or CircleCI are used to perform builds and run tests whenever new pull requests are made. SIL and CPIL testing are more complex as they depend on hardware or drivers like Nvidia's CUDA, which not all computers are guaranteed to have. To solve this issue we have a CI server on a desktop computer that has an Nvidia GPU. This CI server works the same way as Travis/Circle CI; it will build and run automated tests whenever new commits are pushed to it.

The APIL, VIL, and close track driving tests are performed manually due to their dependence on the actual sensors and robot. Another possibility is installing a CI server on the Jetson Nano and perform automated testing on it. Testing on the Jetson Nano is more realistic as it has similar processing capabilities to the Jetson TX-1. This helps catch performance regressions which may not be noticed on a desktop GPU which usually has many times the power of the Jetson.

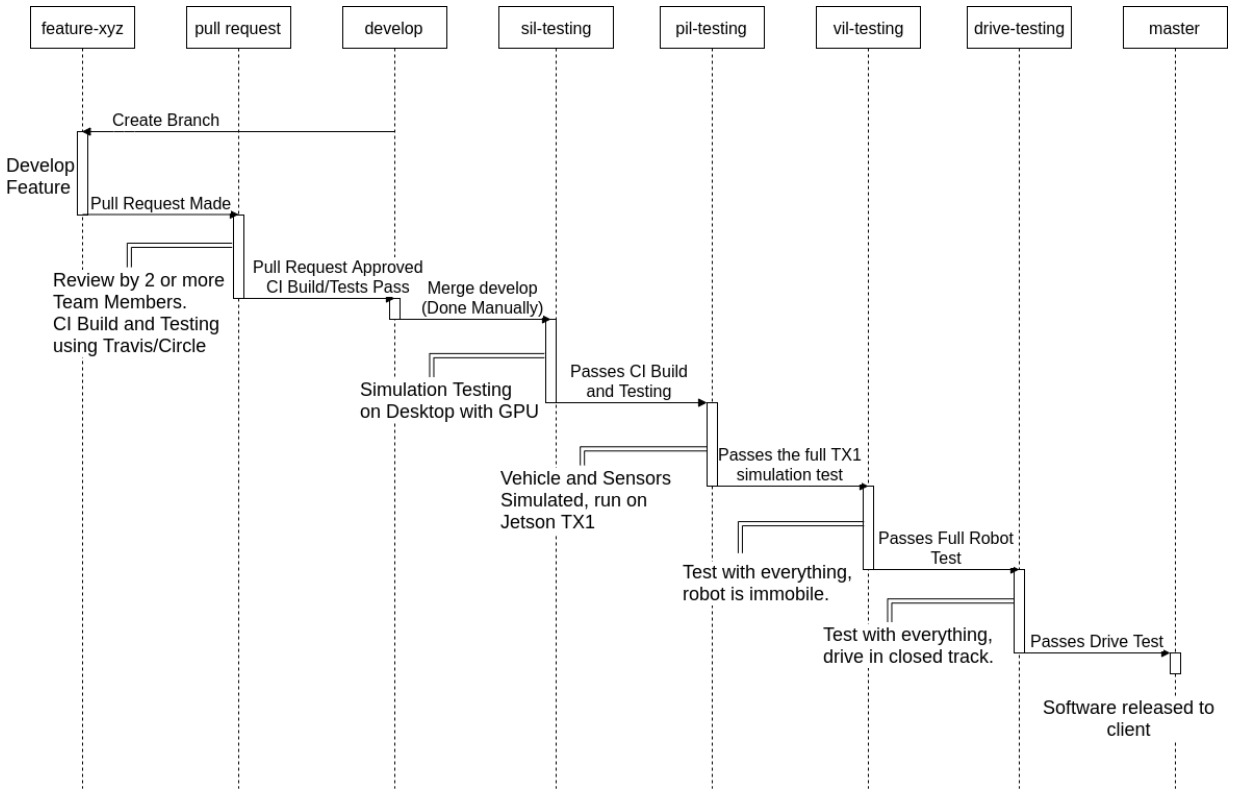


Figure B.1: Branching and loop testing strategy

Appendix C Simulation Selection

C.1 Selection Process

As mentioned before, one of the software validation methods is testing in simulation with mock sensor data. The main requirements for the simulation software is to have simulation for pedestrians and bikers on sidewalks. This is important as PropBot mainly drives on sidewalks meaning it has to avoid obstacles on a narrow pathway. In contrast, self driving cars or cars in general do not normally need to avoid oncoming obstacles. Here are the generated requirements for choosing a software simulator:

Related Tag(s)	Parameter	Value
O3.2, O3.3, O3.5 - Avoid obstacles on sidewalks	Simulation shall simulate pedestrians on sidewalks	The robot mostly drives on sidewalks, so this is the main test case we need to validate.
O3.6, O3.7, O3.8 - Avoidance of static obstacles	Simulation shall be reasonably photo-realistic	The robot needs to recognize common buildings and obstacles to effectively avoid them. There is a need to validate correctness of computer vision algorithms in simulation.
-	Simulation shall provide mocks which are similar to our chosen sensor suite	There is a need to validate ingestion and processing of sensor data in simulation.In addition, decision making based on sensor data needs to be validated in simulation.
-	Simulator shall support ROS or Generic C++ Software.	Simulation should not only test algorithms but also code robustness against different situations. In addition, it is difficult to maintain both a MATLAB/other language model alongside the actual target code.
C1.3 - Robot operates in UBC	Simulator has a University Map or allows custom maps.	Traffic and obstacles in simulation should represent the real world testing location of the Robot.

Table C.1: Software in loop simulation validation requirements

C.2 Comparison of Options

The two main simulators being considered are CARLA and Airsim. The pros and cons of each simulator are compared here:

	CARLA	Airsim
Pros	Free and Open source	
	Supports simulation of pedestrian and car traffic	
	Supports ROS and a variety of simulated sensors	
	Linux and Docker Support (Good for CI Testing)	
	Can Generate Custom Maps	Photo-realistic environments
Cons	Graphics are less detailed/not completely representative of the real world	High system requirements. Minimum system specs are a GPU with 4 GB of VRAM. Microsoft says their typical dev environment consists of 32 GB of RAM and a Nvidia Titan X.
	Fairly beta software. May be buggy or have incomplete features. CARLA may be a bit more mature.	

Table C.2: Comparison of simulators

C.3 Conclusion

Both options have good support for simulation testing of the robot. Airsim's photo-realism is a desirable feature for testing the computer vision algorithms. This may be enough justification to try Airsim first since camera is either the most or second most important sensor in the robot. The main concern with Airsim is finding a suitable map and being able to run the simulator given its high system requirements. However, due to the fact that there are 3D meshes of university campus' available for purchase online, Airsim is suitable choice with these meshes integrated.

References

[1] T. Canada, "Testing highly automated vehicles in canada," Apr 2019. [Online]. Available: https://www.tc.gc.ca/en/services/road/safety-standards-vehicles-tires-child-car-seats/testing-highly-automated-vehicles-canada.html

[2] *A Framework for Automated Driving System Testable Cases and Scenarios.* [Online]. Available: www.nhtsa.gov/sites/nhtsa.dot.gov/files/documents/13882-automateddrivingsystems_092618_v1a_tag.pdf

[3] "Software-in-the-loop testing applications," 2019. [Online]. Available: https://www.add2.co.uk/applications/sil/

[4] R. Rossi, C. Galko, H. Narasimman, and X. Savatier, *Vehicle Hardware-In-the-Loop System for ADAS Virtual Testing.* [Online]. Available: https://www.riverpublishers.com/pdf/ebook/chapter/RP_9788793519138C11.pdf

[5] "World geodetic system (wgs84)." [Online]. Available: https://gisgeography.com/wgs84-world-geodetic-system/

[6] "Uctronics smart bluetooth robot car kit - uno r3 for arduino, line tracking, ultrasonic sensor, hc-05 bluetooth, l239d motor shield, ir remote control, mobile app - charger included." [Online]. Available: https://www.amazon.ca/gp/product/B07VT31RWY/ref=ppx_yo_dt_b_asin_title_o00_s00?ie=UTF8&psc=1

[7] "Measuring motor parameters." [Online]. Available: http://support.ctc-control.com/customer/elearning/younkin/motorParameters.pdf