SYSC4805A
Final Report
Winter 2021 - Dr. Mostafa Taha

Warehouse Transport Robot (WTR)

Team Asparagus
github.com/SYSC4805-Winter-2021/project-asparagus

Ben Bozec (101034327)
Brannon Chan (101045946)
Alan Lin (101036660)

April 9th, 2021

Table of Contents

Project Charter

Overall Objective

Producing a self navigating and optimized robotic approach for loading and unloading goods in a commercial warehouse environment.

Overall Deliverables

Large commercial warehouses similar to E-Commerce giants like Amazon.com contain hundreds of thousands of goods on shelves that will eventually be shipped. The process of moving these items from their storage locations to a designated packaging area becomes a significant challenge for humans at this scale. We propose the Warehouse Transportation Robot (WTR), a robot that will take care of this warehouse goods transporting process. In order to aid in autonomous shelving operations, the Storage Shelf Unit (SSU) robot will assist the WTR in receiving and dispensing warehouse goods. Our robot will be capable of carrying a payload and autonomously navigate itself to multiple specific locations in some arbitrary aisle-based warehouse. In order for the robot to properly navigate this warehouse, it will continuously update its routing data during operation. Beginning at the robot's home base (original initialization location), it is loaded with all routes to each destination in the warehouse. Once the robot reaches any destination, the robot will update its routing data with information that is relative to the current location. This method will keep the size of the stored routing data on the robot to a minimum, effectively reducing memory resource consumption. The robot will also feature vision sensors to implement collision avoidance for other robots, human workers or obstacles found in the warehouse. For optimal maneuverability, the robot will utilize four independently driven motors that allows for three hundred and sixty degree turns. Though the implemented WTR is relatively small, many of these robots could be deployed independently to simultaneously maximize work output as a scalable solution for larger warehouses.

Scope

List of Requirements

- Navigation Requirements
    - Robot can automatically navigate a warehouse with marker-labeled aisles and specific aisle shelf locations
    - Robot can navigate warehouse from any marker-defined starting location
    - Robot can return to a home base (or initial starting point) from any specified maker location
- Collision Detection Requirements
    - Robot can avoid imminent or detectable collisions with obstacles & humans

    - Robot can automatically make locomotion route adjustments intermittently to avoid glancing collisions against aisle walls
- Transportation Performance Requirements
    - Robot's payload can be more than one physical item(s)
    - Robot can perform precise standstill turns (within negligible turn error)
    - Robot can autonomously load and unload a payload into a shelving unit properly
    - Robot can queue up multiple jobs at a time (start next job immediately after completion of current)

Work Breakdown Structure (WBS)

Figure 1 below is a work breakdown structure diagram of the project. As seen in the diagram, we will start off with planning work tasks, then robot development, warehouse scene creation, and then to the development of navigation logic. After we have a robot built and has locomotion with basic navigation implemented, the next stage of work would be testing the whole system. After performing and passing the suite of tests, the final stage is to deliver the product with proper documentation and presentation. For each column of work in the WBS, the order of which work should be done is from top to bottom. This is done to ensure any dependencies are fulfilled ahead of time.

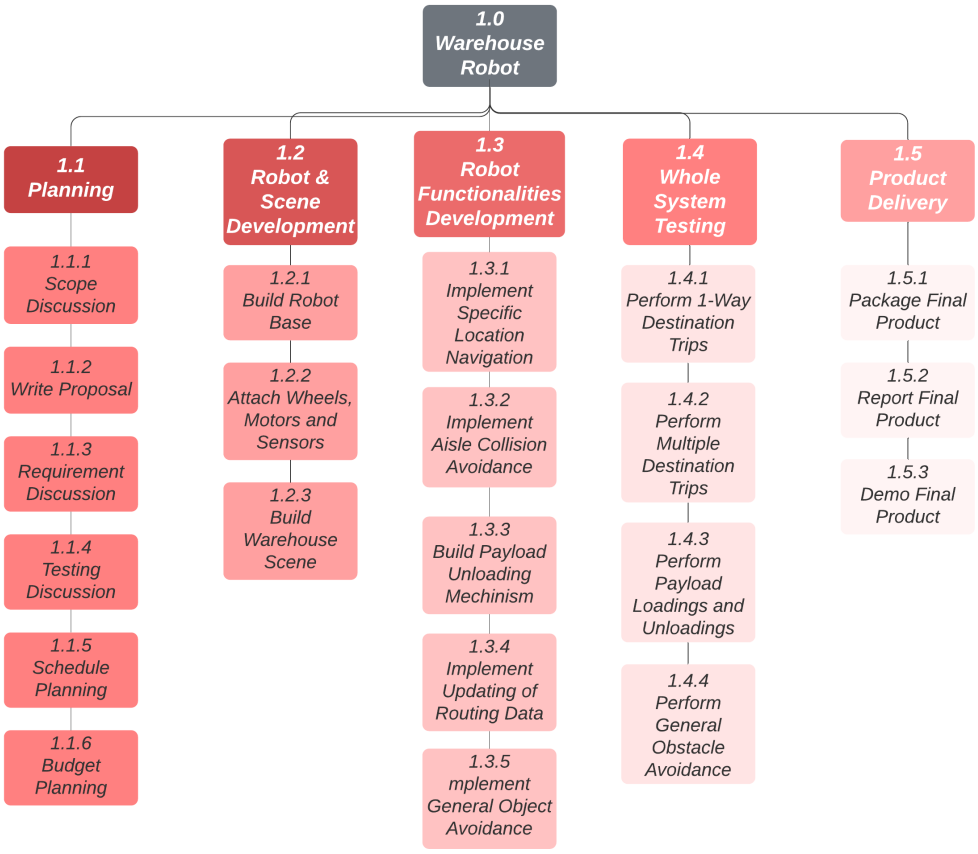

Figure 1 Work Breakdown Structure (WBS) of warehouse robot development

Testing

Table 1 below are the customer focused test cases our warehouse robot will undergo. These tests are based around the robot's functionality and the user's experience. The tests are separated into categories/features to help isolate potential issues. When a feature is completed, the test cases that are related to that category of feature should run. I.e. After building the robot, *Robot Functionality* tests should run.

Test Category	Test Case	Expected Results
Navigation	Starting from a specific warehouse location, the robot can navigate to any known warehouse destination.	Robot navigates to the specific warehouse location.
	Starting from a specific warehouse location, the robot will navigate to the next queued destination.	Robot navigates to the specific warehouse location.
Routing	Starting from its home base, the robot has routing data to every known warehouse destination.	Debug data will show a complete and correct list of the warehouse routing data.
	Starting from a specific warehouse location, the robot has routing data to every known warehouse destination.	Debug data will show a complete and correct list of the warehouse routing data.
	When arriving at a specific warehouse location, the robot's routing data is updated.	Debug data will show a complete and correct list of the warehouse routing data that is based on the specific location as the starting point.
Collision Avoidance	Robot will maneuver around an object that it detects in only one of its two front facing collision sensors	Robot will detect the object, maneuver around it, return to the centre of the lane and continue on its course.
	Robot will stop when it detects and object in both of its front facing collision sensors and wait for the object to be removed	Robot will detect the object, come to a stop, continue to wait until the object is moved, and then continue on its course.

Payload Management	When arriving at an SSU, the WTR and SSU can properly determine when the two are docked.	WTR & SSU will successfully enter the “EMPTY” or “FULL” state.
	When departing from an SSU, the WTR and SSU undock appropriately.	WTR & SSU will successfully enter the “IDLE” state.
	When the SSU dispenses a payload, the WTR will load it successfully.	SSU enters “EMPTY” state, WTR enters “FULL” state.
	When the SSU receives a payload, the WTR will unload it successfully.	SSU enters “FULL” state, WTR enters “EMPTY” state.
	Following a loading/unloading operation, the WTR will correctly resume navigation.	WTR returns the thread of control to Navigation and is correctly lined up with floor markers.
	Prior to a loading/unloading operation, the WTR will correctly begin the docking sequence.	WTR transfers the thread of control to payload management and lines up with SSU properly.
	When performing a loading/unloading operation, the payload will not fall off, get stuck, or cause a state deadlock.	Whole system testing shows that the payload does not get stuck, fall off, or cause the WTR & SSU to become state deadlocked.
Robot Functionality	Robot can move forwards and backwards.	Robot moves forward on demand and moves backwards on demand.
	Robot can make standstill turns.	Robot stops one side of wheels and turns.
	Robot can make turns while moving.	Robot reduces speed of one side of wheels and turns.
	Robot can carry a payload.	Robot does not function differently when the payload is loaded.

Table 1 Warehouse robot test cases

Design

The link below is to our GitHub repository for this project. The main directory of the repository contains the latest file versions. The file *mainRobot.ttm* is the CoppeliaSim model WTR (seen in Figure 2) and *WarehouseSceneWithRobotV5.ttt* is the CoppeliaSim scene that contains the robot in a warehouse.

https://github.com/SYSC4805-Winter-2021/project-asparagus

Overall Architecture

Figure 2 below is the Warehouse Transport Robot (*mainRobot.ttm*) carrying some payload in the warehouse scene. As seen here, the robot's hardware components consist of four independently driven wheels, a payload conveyor belt, multiple vision and proximity sensors & a translucent payload barrier. The independently driven wheel design allows for tight, 'on the spot' maneuverability. The conveyor belt on top doubles as a payload carrier and its shelf loading/unloading mechanism. The various sensors seen below are for collision avoidance, floor marker recognition and payload detection. The payload barrier was added to prevent lost cargo after noticing the payload moves around on the conveyor belt while navigating.

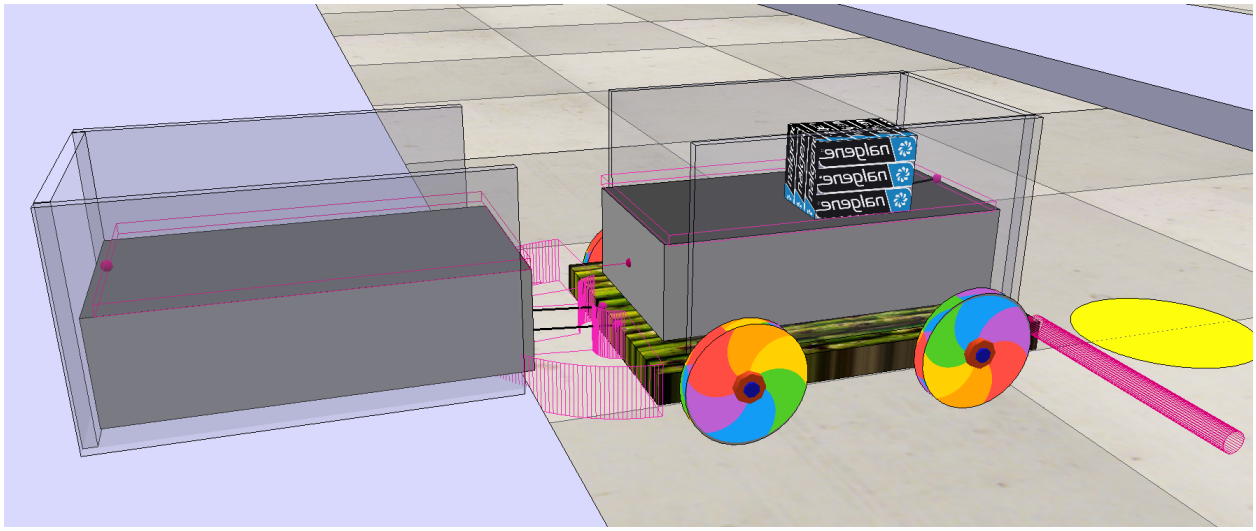

Figure 2: The Warehouse Transport Robot (WTR) with a sample cuboid payload approaching a Storage Shelf Unit (SSU).

State Chart

The state chart below (*Figure 3)* is of a single run of the robot in the warehouse. The main navigation of the robot is performed in the *Navigation* container, as this is the group of states where the robot is continuously traversing the warehouse, while looking out for obstacles.

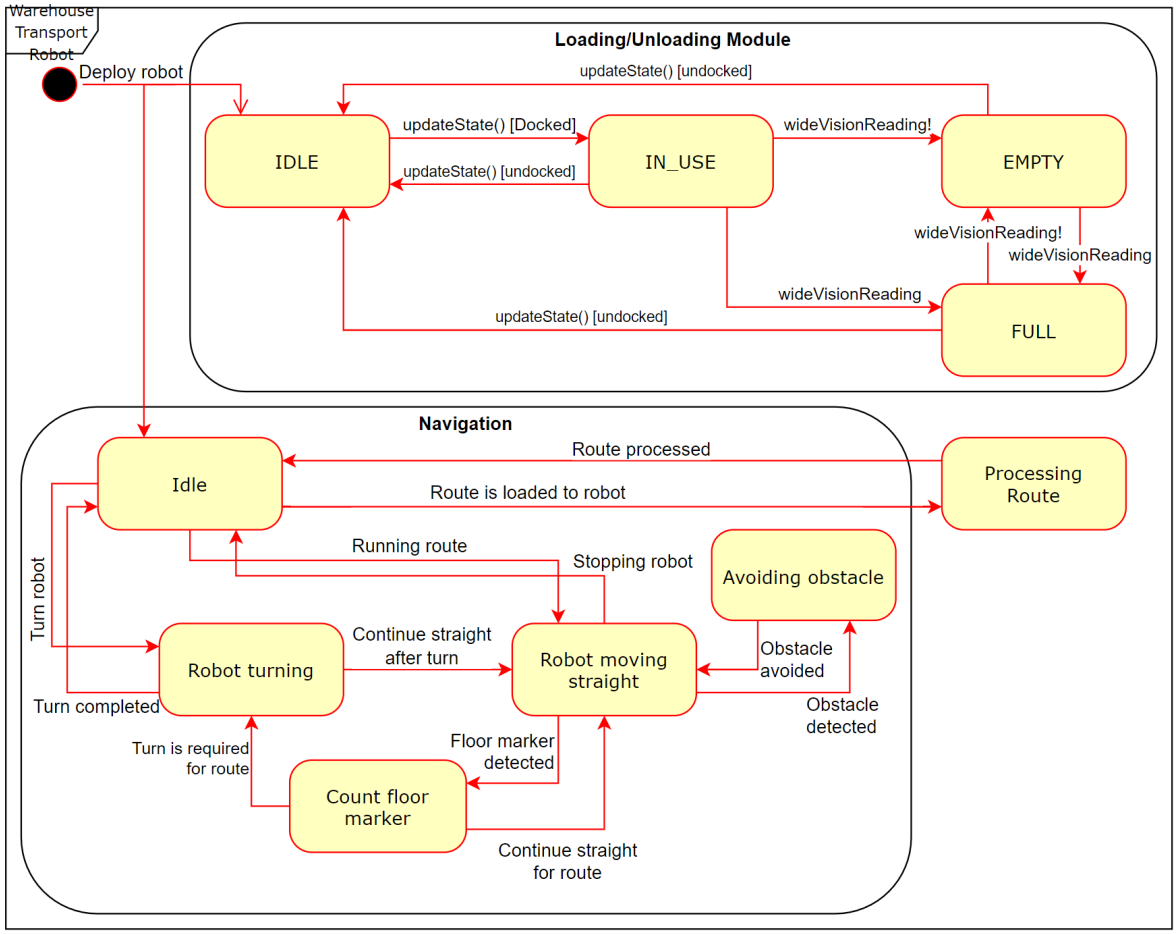

Figure 3: The Warehouse Transport Robot state chart

Sequence Diagram

The sequence diagram on the next page (*figure 4)* is of a single run of the robot in the warehouse. The main navigation of the robot is performed in the *loop* container, as this is the sequence where the robot is continuously traversing the warehouse, while looking out for obstacles.

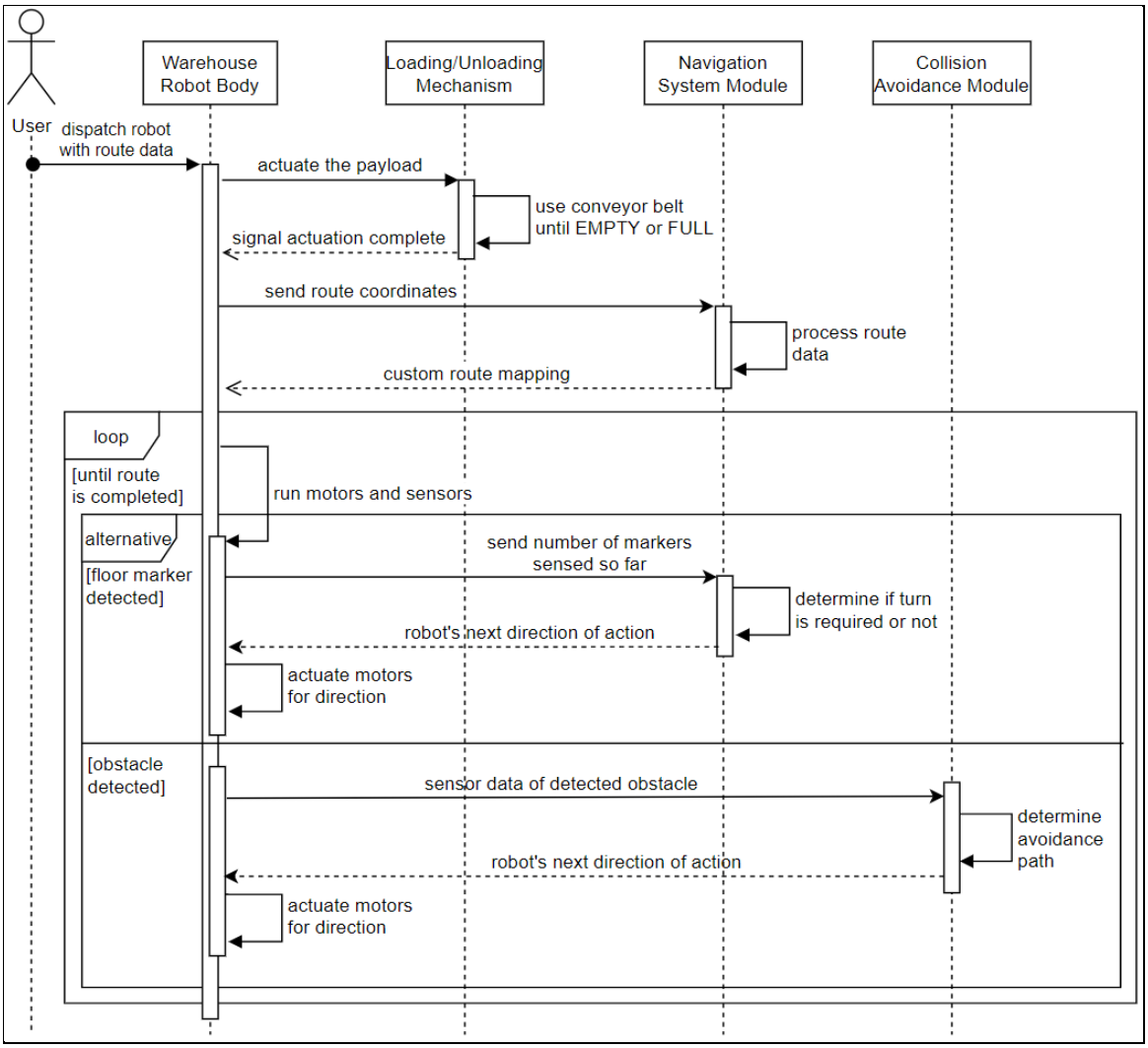

Figure 4: Warehouse Transport Robot UML sequence diagram

Event Triggered Implementation

The warehouse transport robot uses an event triggered design, as it relies heavily on moment to moment inputs from it's sensors to determine if an action is required. The robot's two key functions, navigation and collision avoidance, remain inactive as long as the associated sensors do not detect the necessary stimulus needed to initiate a change in the system's current action or state. The navigation system uses markers throughout the "warehouse" to determine its current location and decide where/when it should change directions. The collision avoidance system relies on receiving the inputs from the front facing sensors shortly following a change so that it has time to respond to obstacles in the robot's path. Since the robot does not inherently know when these stimuli will be detected, a time triggered design would limit its ability to respond in a timely manner and could lead to the system's response arriving too late or being missed entirely. With an event triggered design, the system will be able to immediately recognize that these stimuli have entered the detection range of the sensors and take the necessary response.

Control Charts

Robot Navigation

When running the WTR with no obstacles in the warehouse, the robot was able to reach the destination without issue. When an obstacle was placed in the warehouse scene, the same runs resulted in several failures. This was due to hiccups in the collision avoidance system. The robot would get stuck at the obstacle and navigation could not continue.

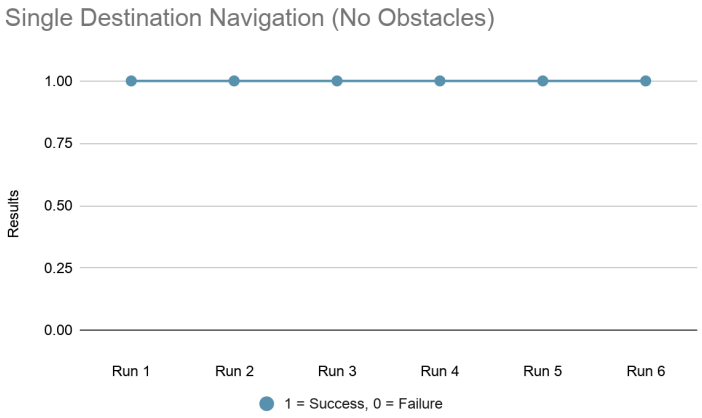

Figure 5: Single destination with no obstacles control chart

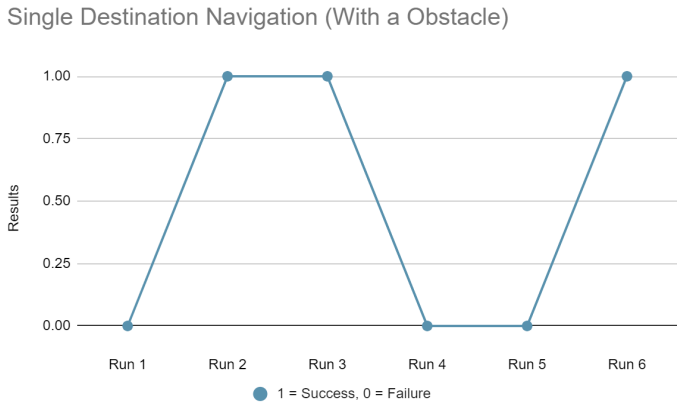

Figure 6: Single destination with obstacles control chart

Similarly to the single destination navigation runs, the WTR with no obstacles in the warehouse, was able to reach the destination. Though there were a couple of failures due to an ongoing bug in the navigation system. When obstacle(s) were placed in the warehouse scene, the same runs resulted in mostly failures. This was due to hiccups in the collision avoidance system. Particularly when the robot had to avoid two obstacles during the run, it would usually get stuck at least in one of the obstacle avoidance attempts.

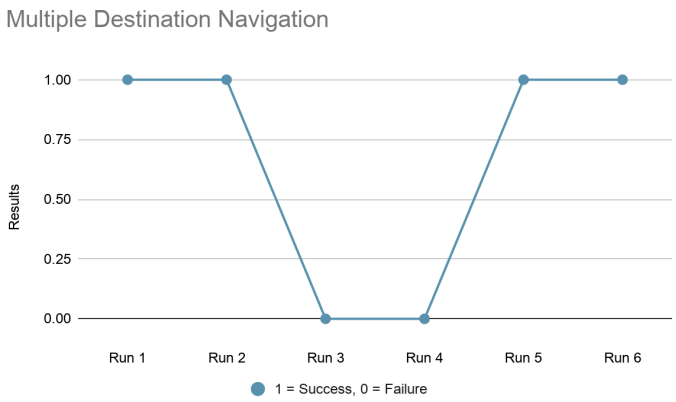

Figure 7: Multiple destination with no obstacles control chart

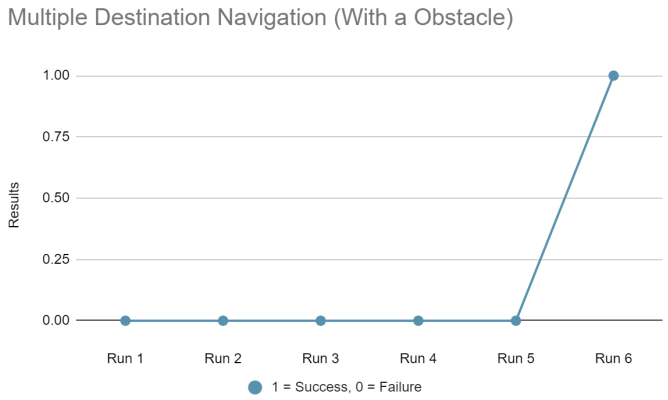

Figure 8: Multiple destination with obstacles control chart

Collision Avoidance

For the partially obstructed scenario, the system can sometimes fail when avoiding objects that do not have a flat surface. The rear sensors sometimes do not reach the object after the initial turn, causing the robot to believe that it is no longer blocked. This causes it to immediately turn back and drive directly into the obstacle. For some reason, this issue is more likely to occur when the object is detected on the right side compared to the left.

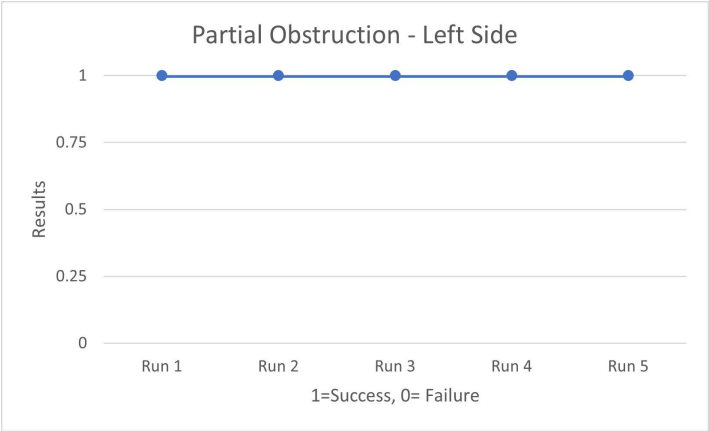

Figure 9: Partial obstruction, left side collision avoidance control chart

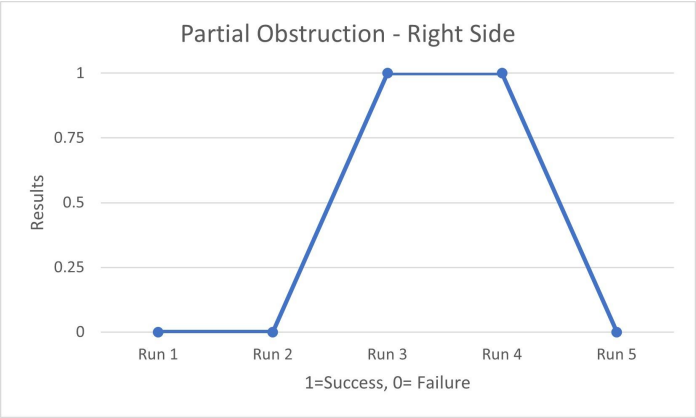

Figure 10: Partial obstruction, right side collision avoidance control chart

For full-frontal obstruction, the robot will more often than not successfully stop in front of the object and wait for the obstacle to be removed. The main cause of failure comes from objects that do not have a flat surface for the sensors to detect. In this scenario, one of the two sensors will register the object before the other which triggers the partial obstruction handling. While this does not present a problem if there is a gap that the robot can fit through, the problem occurs when the object extends to the wall. This causes the robot to crash and require a reset.

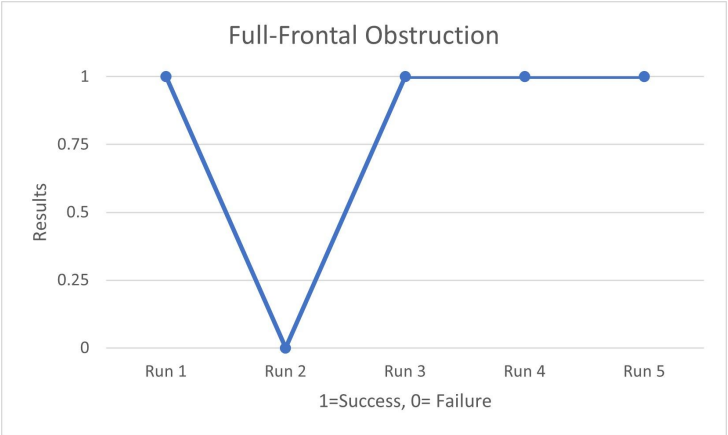

Figure 11: Full obstruction collision avoidance control chart

Payload Management

To determine reliability and consistency of the Payload Management subsystem, the WTR and SSU were subjected to multiple ‘regular’ use case tests for at least six test runs each in operation with the rest of the system; namely the Navigation logic. Due to inconsistencies in the CoppeliaSim simulation environment, the Collision Avoidance system has been proven to fail consistently on Brannon’s computer, and therefore will not be tested alongside Payload Management.

In our first test, the WTR will be required to successfully load a single payload consisting of two cuboid items (Figure 12, as seen in the Appendix) beginning at (0, 0) and unload at destination (2,0). Based on test observations, it seems that a timing issue on the SSU can cause some payloads to be inadequately pushed off or pulled onto the WTR, dropping a payload item on the floor at the origin/destination. Additionally, incorrect alignment with the SSU due to systematic marker detection error in the Navigation system can cause payloads to hit the translucent item guard on the SSU/WTR and fall off, or miss the correct shelf marker and SSU entirely. As a result, this test has a pass rate of ⅝ or 62.5%.

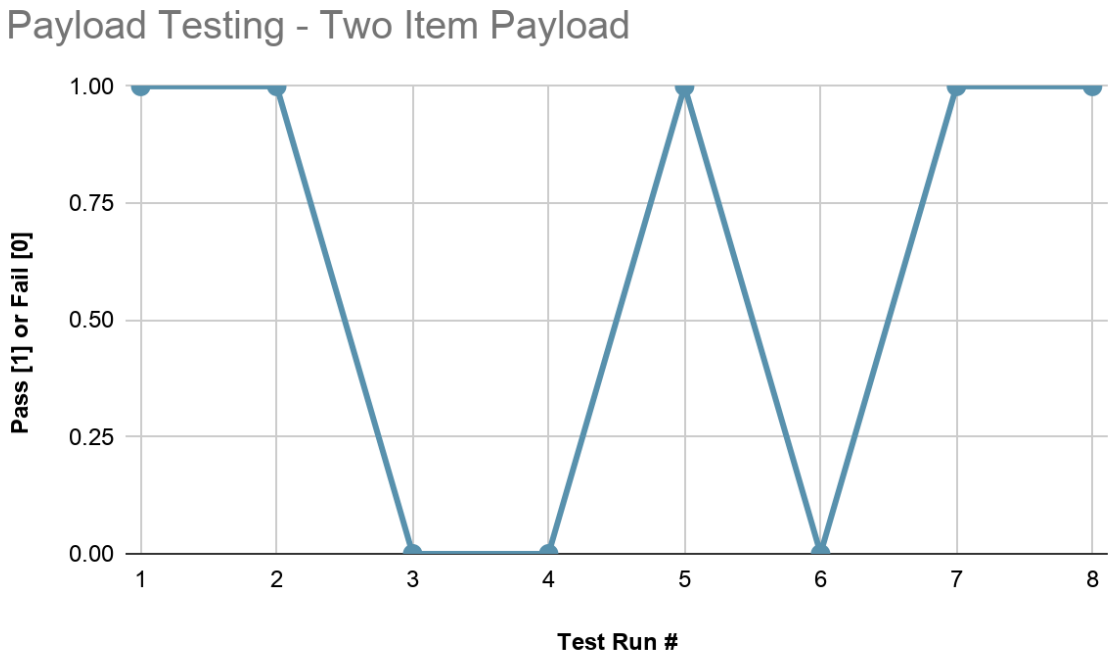

Figure 13: Control Chart for WTR carrying two items as its payload over 8 total test runs.

Next, we examine the ability for the WTR to load and unload payloads at more than one pair of origins and destinations, essentially testing multiple shelf loading/unloading capability. This test also relies heavily on the Navigation subsystem, as marker detection errors would mean completely missing the next shelf. As such, this test will require the WTR to deliver an initial payload consisting of one cuboid item to its origin (0, 0), travel to point (0,1) and receive a payload (one cuboid), which will be delivered to an empty SSU at the destination (2, 2). Based on the test observations, the systematic Navigation marker errors present in the previous test has been multiplied threefold, making the distance deviated from the centre of waypoint markers

to increase; causing whole markers to be missed entirely, counted twice, and even causing the WTR to get stuck on the SSU due to a bad docking angle. Though most test runs were able to successfully load the payload from the second point (0, 1), the navigation issues mentioned above caused the WTR to miss its' last destination almost every time, resulting in a pass rate of ⅙, or just 16.67%.

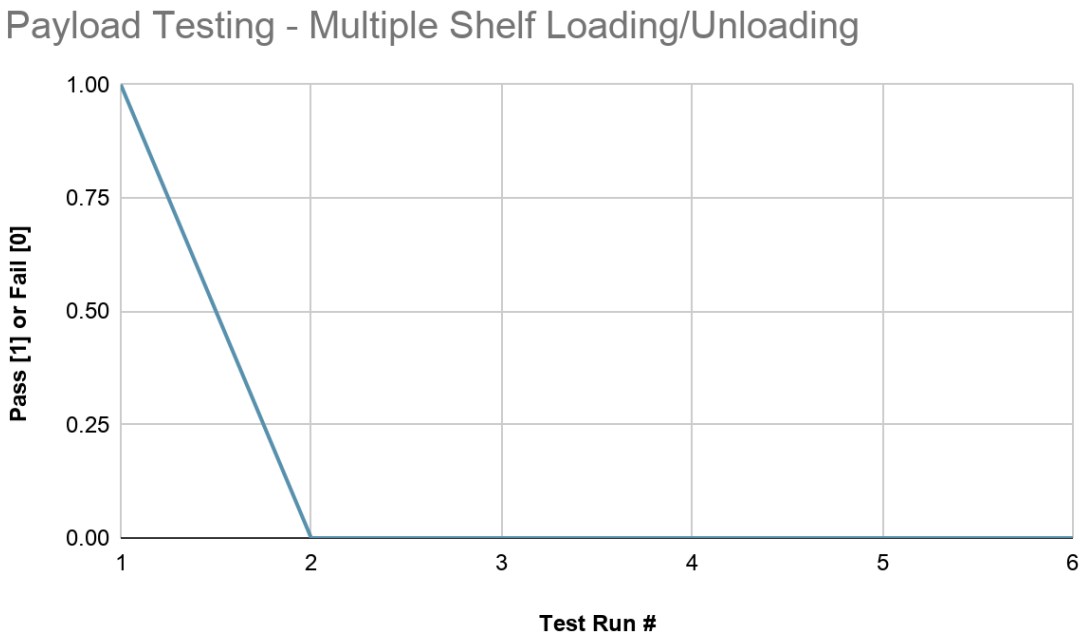

Figure 14: Control Chart for WTR performing an unload-load-unload payload operation at multiple shelves over a total of 6 test runs.

Schedule

Schedule Network Diagram

The Schedule Network Diagram (*Figure 15*, as shown below) contains the specific activities that make up the major deliverables for this project from start to finish. You may notice that some activities can be worked on in parallel with other project activities, but some activities require adjacent (or previous) activities to be completed before the activity in question may finish. Since there were no complications with project development flow since the progress report, no changes have been made to the Schedule Network Diagram.

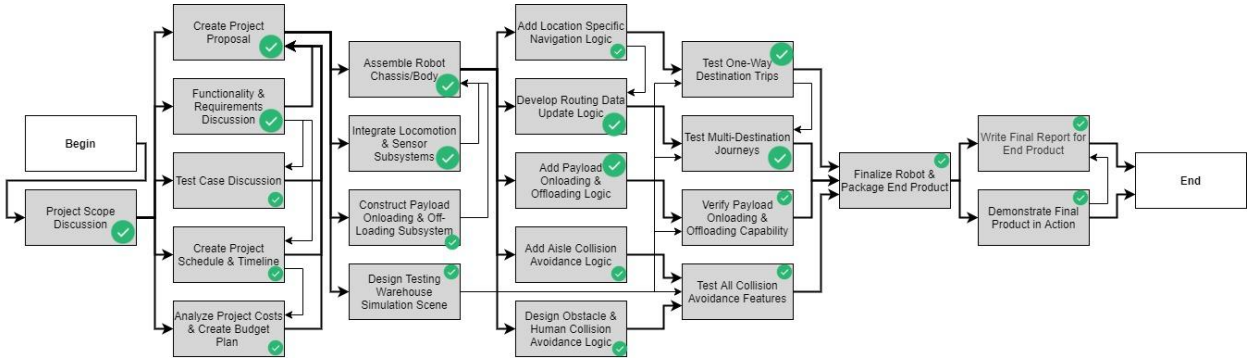

Figure 15 Schedule network diagram of warehouse robot, original image can be found in Appendix

Gantt Chart

Figure 16 below shows the current schedule for the implementation of the Warehouse Robot's different components. It began with building the body of the robot and loading mechanism, including the setting of the joints and sensors, and designing the warehouse scene that the robot navigated through. In the next lab, we used the time to ensure that the robot was capable of smooth movement and the sensors were providing the feedback that we needed. The next following two weeks were spent implementing the navigation, which allowed it to move from point to point within the "warehouse" and avoid objects on the robot's route. The fifth week was then focused on the mechanism that loads and unloads the payload from the robot. During the final week, we performed the final stage of testing and fixed any outstanding issues.

Figure 16: A Gantt Chart depicting the 6 week schedule for completion of the warehouse robot project

Human Resources

Responsibility Assignment Matrix

All the work that relates to planning and final delivery will be performed as a group.

WBS Code	Title	Ben Bozec	Brannon Chan	Alan Lin

1.1	**Planning**			
1.1.1	Scope Discussion	R, A	R, A	R, A
1.1.2	Write Proposal	R, A	R, A	R, A
1.1.3	Requirement Discussion	R, A	R, A	R, A
1.1.4	Testing Discussion	R, A	R, A	R, A
1.1.5	Schedule Planning	R, A	R, A	R, A
1.1.6	Budget Planning	R, A	R, A	R, A
1.2	**Robot Development**			
1.2.1	Build Robot Base	A		R
1.2.2	Attach Wheels, Motors, and Sensors	A		R
1.2.3	Build Payload Unloading Mechanism		R	A
1.3	**Navigation Logic Development**			
1.3.1	Implement Aisle Collision Avoidance	R	A	
1.3.2	Implement Specific Location Navigation	A		R
1.3.3	Implement Updating of Routing Data		R	A
1.3.4	Implement General Object Avoidance	R	A	
1.3.5	Design Warehouse Scene	R		A
1.4	**Whole System Testing**			
1.4.1	Perform 1-Way Destination Trips		A	R
1.4.2	Perform Multiple Destination Trips	A	R	
1.4.3	Perform Payload Loading and Unloadings		R	A
1.4.4	Perform General Obstacle Avoidance	R	A	

1.5	**Product Delivery**			
1.5.1	Package Final Product	R, A	R, A	R, A
1.5.2	Report Final Product	R, A	R, A	R, A
1.5.3	Demo Final Product	R, A	R, A	R, A

Legend:
R = Responsible
A = Approver

Conclusion

The final iteration of the WTR is able to perform all of the necessary tasks that we set out in the initial proposal. It is capable of transporting an object from one point in the warehouse to another, while avoiding obstacles enroute, and successfully deposit the payload at its destination. There are still some areas that we would prefer to be more polished. The navigation has difficulty with certain routes, the collision avoidance will occasionally attempt to return to the centre of the lane before it is completely past the object, and we initially planned for the robot to handle multiple deliveries simultaneously. Nonetheless, we were able to implement the core features and are content with the final product.

Appendix

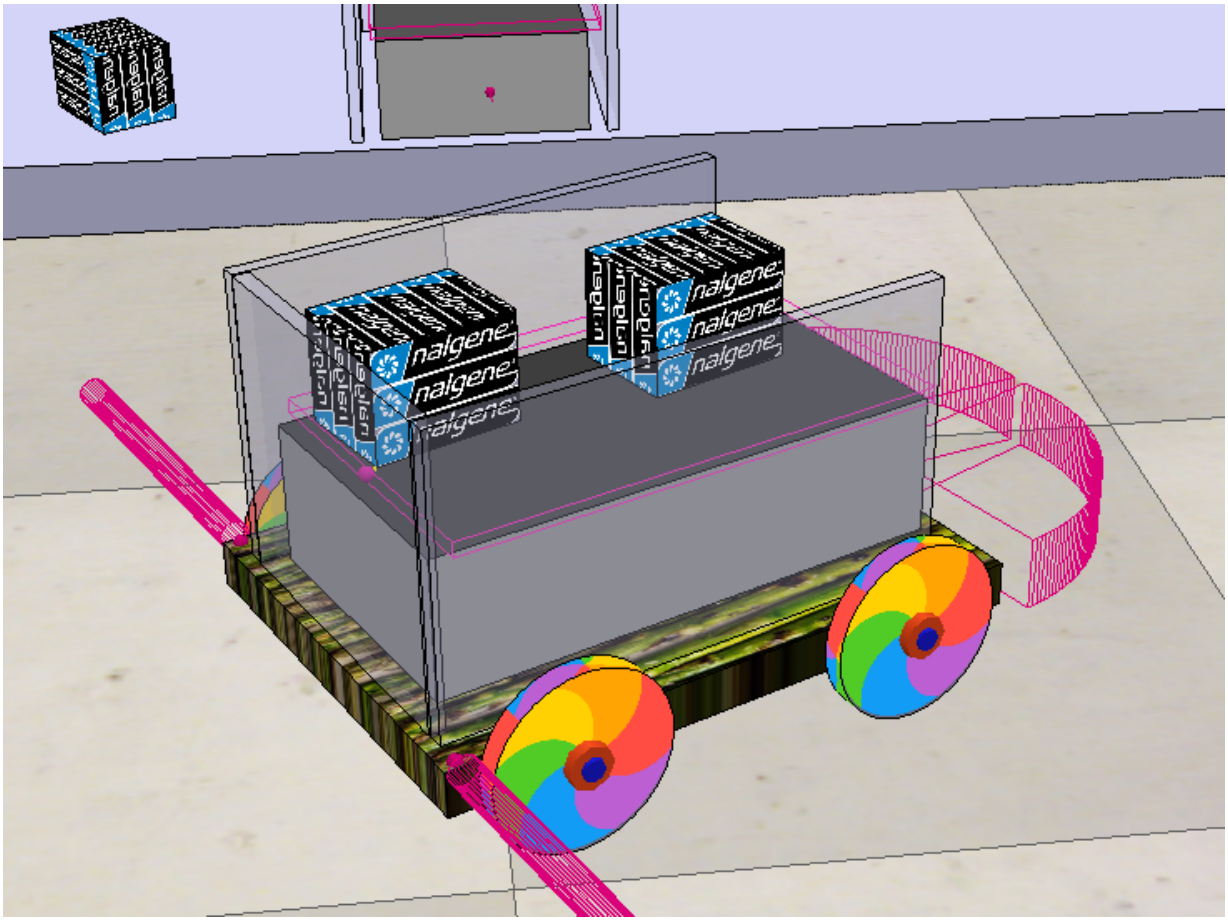

Figure 12: The WTR carrying a payload of two items for the first control chart test.

Video

https://drive.google.com/file/d/1W_pRx5ZrVIeyvG8t4b21tXYYTFivKH3G/view?usp=sharing

URL for the Schedule Network Diagram in original resolution:
https://drive.google.com/file/d/1x40fuJlfhEif7O2jjmFC_9ur0INq3fC8/view?usp=sharing