SYSC 4805 - Final Report

Simulated Autonomous Snow Plow

Members:

Michael Marsland - 101042414

Edmond Chow - 100883365

Alfred Akinkoye - 101109483

Deji Sayomi - 100847393

Date:

April 12th, 2022

Table of Contents

PROJECT PROPOSAL

Project Charter

Team Name: ***Jazzberry Jam***

Objective:

To design and build an autonomous robot in CoppeliaSim to clear snow from a designated area in multiple varying test environments within a limited amount of time while avoiding collisions with obstacles.

Deliverables:

An autonomous vehicle capable of clearing snow from a designated area. The vehicle will be equipped with a unique plow to remove snow from the area as well as a number of sensors to allow the vehicle to detect various obstacles and the perimeter. The vehicle will be run using a Python API connected to CoppeliaSim.

We will develop an algorithm to help accomplish all the tasks necessary, such as; keeping track of the area previously covered by the plow as well as areas to be covered, ensure the vehicle does not collide with objects and ensure that snow is pushed outside the perimeter of the area.

To validate the snow plower and ensure the algorithm works as intended, we will devise a number of test scenarios to make certain all the deliverables meet the requirements of the project. To show the progress of our work, and inevitable changes we've made to the projects, we will also be making a progress report.

The progress report will highlight all the tasks we've accomplished from what we set out to do, and what we have left to accomplish. Through this project we will be learning the life cycle of developing a product, from proposal all the way to the final product and presentation. We will also be learning how to work in a team environment to validate each other's work, to certify the best possible out-come.

Scope

Requirements:

1. The snow plow will clear the enclosed area of at least 80% of snow orbs within 5 minutes
2. The snow plow will clean the enclosed area without impacting other environmental objects (IE. walls and trees)
3. The snow plow will not move at speeds that exceed 2 m/s
4. When in the Parking Space the dimensions of the snow plow will not exceed 0.5m by 0.8m by 1m
5. The dimensions of the snow plow will not exceed 1 m by 0.8 m by 1 m

Work Breakdown Structure:

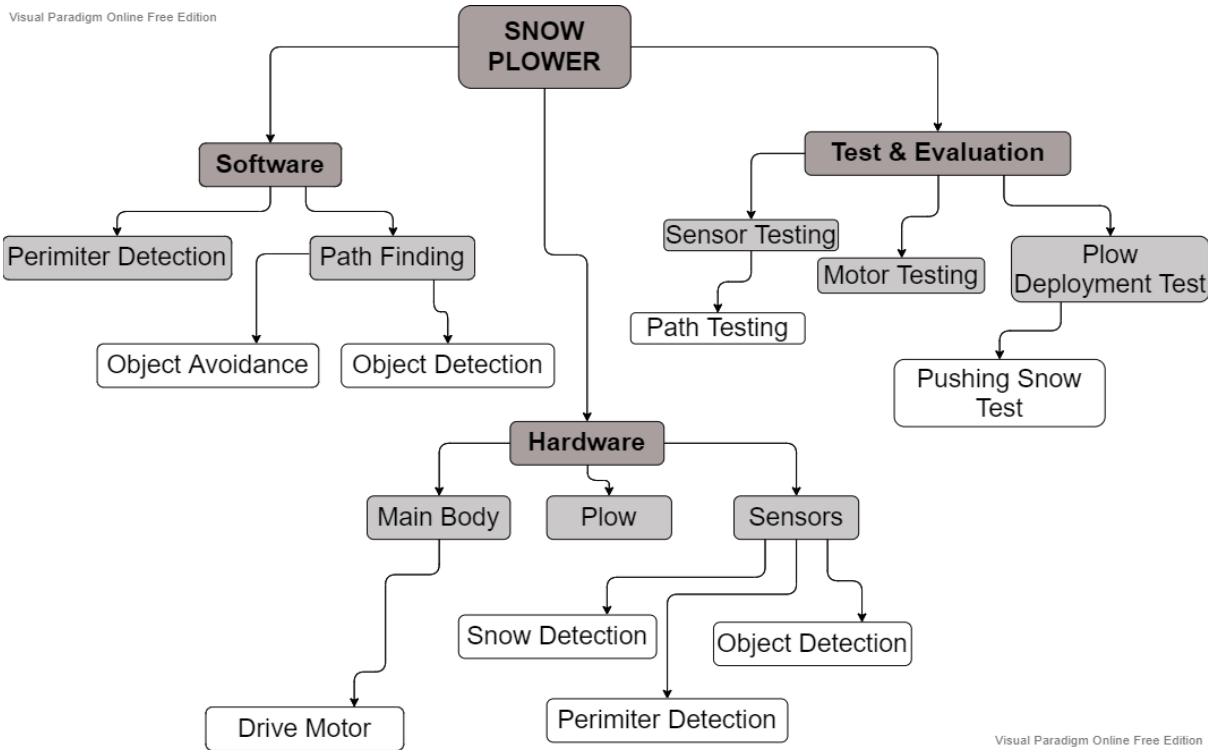

Figure 1: Work Breakdown Structure

Description of Tasks:

TASK	Description
Discuss and compose Project Proposal	Write the project proposal
Discuss and Compose Progress Report	Write the progress report
Discuss and Compose Final Report	Write the final report, create the final presentation, deploy the program for demonstration
Build Robot Body	Construct robot body including motorised wheels with locations for the plow and various sensors
Create Unfolding plow	Design a plow with motor that fits into the required space and is capable of unfolding to its full length and manoeuvring as required
Experiment with Plow Shapes	Test various plow shapes to determine which are best for pushing and manoeuvring the snow
Attach Plow to body	Connect the designed plow to the robot and ensure

	both function together
Attach Sensors	Attach the various required sensors at locations that work for their respective tasks
Plow and Motor Control	Control drive motors from Python API and track movement, build basic movement library
Program Pathfinding	The plow should evaluate a path to push snow off an empty map
Program Sensor Detection	Read values from sensors in Python through the API and detect various objects by their distinctions
Implement Python API	Connect CoppeliaSim to a Python Program through the API
Create Plowing Algorithm (Pathfinding + Avoidance)	Ensure snow detected by the robot and collected is removed from the perimeter and not left on the map
Object Avoidance	Re-evaluate a new path to push snow off the map based on the detected objects
Outline Test Requirements	Create detailed test plans for each feature based on the requirements and how the parts are constructed
Sensor Testing	Ensure that when the robots sensors pass over the black perimeter we should respond by dumping snow outside the perimeter, Ensure that when the robot's sensors first detect an object the plow should stop it's drive motors and determine how to carry on to not collide
Motor Testing	Ensure calculated movement amounts in Python are correctly translated into CoppeliaSim (1 meter movement testing)
Plow Deployment Testing	Ensure the plow fits within the 0.5x0.8x1m parking space when folded and extends to the full 1m width when deployed
Pushing Ability Testing	Ensure that when the plow is pushing a large amount of snow the motors still move the robot the correct distance when instructed
Pathing Testing	Ensure the CoppeliaSim robot's path mimics the path determined in an empty and populated map
Combined Testing	Run the entire system in the test scene given for 5 minutes and determine our score. Iterate this test with various object, and snow positions.

Testing:

Test (to be performed)	Pass Criteria	Fail Criteria
Sensor (IR)	Pass if it can detect when the vehicle crosses over the black perimeter of the stage and transmit data to the script	Fail otherwise
Sensor (Proximity)	Pass if it can detect surrounding objects and transmit data to the script	Fails if either one of the conditions above fails
Motor	Pass if the movements in the simulation is an accurate portrayal of inputted values in the script	Fail if motors aren't accurately being controlled by the script
Plow Deployment	Pass if able to open and close plow at the request of the script	Fails if not
Pushing Ability	Pass if plow is still sturdy whilst pushing snow	Fails if plow falls apart, or is unable to push the snow for any reason
Path Finding	Pass if able to move around the designated area without running into obstacles and always comes back within bounds if it exits.	Fails if either robot stays out out of bounds for more than 10 seconds, or robot collided with obstacles
Combined Operations	Ensure the entire system can effectively clear 80% of the snow within 5 minutes without colliding with objects in the environment	Fails if unable to clear the amount of designated snow, or any of the above tests fails

Schedule

Schedule Network Diagram:

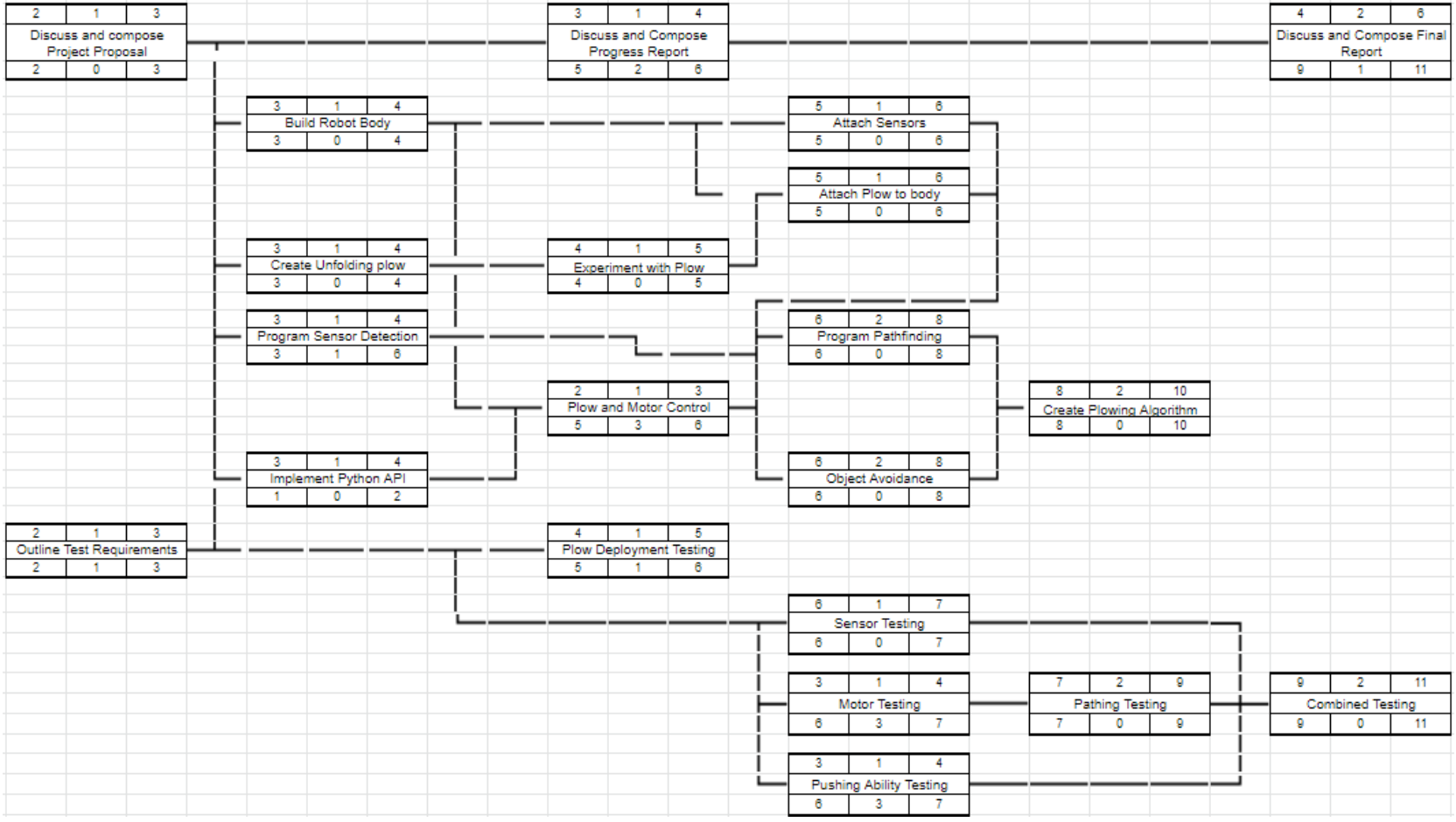

Figure 2: Schedule Network Diagram

Gantt Chart:

Key	Project Deliverables				Group Deliverables						ALL
Task Title	**Lab 3**	**Lab 4**	**Lab 5**	**RB**	**Lab 6**	**Lab 7**	**Lab 8**	**Lab 9**	**Lab 10**	**Lab 11**	**Lab 12**
Project Proposal (Feb 4th @ 11:59PM)											
Progress Report (March 1st @ 11:59PM)											
Demonstration											
General Milestones											
Completed Progress Report											
Completed Testing and Production Ready System											
Administration											
Discuss and compose Project Proposal											
Discuss and Compose Progress Report											
Construction											
Build Robot Body											
Create Unfolding plow											
Experiment with Plow Shapes											
Attach Plow to body											
Attach Sensors											
Programming											
Program Sensor Detection											
Implement Python API											
Plow and Motor Control											
Program Pathfinding											
Object Avoidance											
Create Plowing Algorithm (Pathfinding + Avoidance)											
Testing											
Outline Test Requirements											
Plow Deployment Testing											
Sensor Testing											
Motor Testing											
Pushing Ability Testing											
Pathing Testing											
Combined Testing											

Human Resources

Responsibility Assignment Matrix:

TASK	ALFRED	MICHAEL	EDMOND	Deji
Discuss and compose Project Porposal	Doing	Doing	Doing	
Discuss and Compose Progress Report	Doing	Doing	Doing	
Build Robot Body	Doing		Checking	
Create Unfolding plow	Checking		Doing	
Experiment with Plow Shapes		Checking	Doing	
Attach Plow to body		Doing	Checking	
Attach Sensors	Doing		Checking	
Plow and Motor Control		Checking	Doing	
Program Pathfinding	Doing		Checking	
Program Sensor Detection	Checking			Doing
Implement Python API	Checking	Doing		
Create Plowing Algorithm (Pathfinding + Avoidance)	Checking		Doing	Checking
Object Avoidance	Doing			Checking
Outline Test Requirments				
Sensor Testing		Doing		Checking
Motor Testing		Doing		Checking
Plow Deployment Testing			Checking	Doing
Pushing Abillity Testing		Doing	Checking	
Pathing Testing		Checking		Doing
Combined Testing	Doing	Doing	Doing	Doing

PROGRESS REPORT

Overall Architecture

The purpose of this project is to design an autonomous snow plow capable of clearing a number of simulated environments of snow within the given time constraint while avoiding various obstacles. In order to achieve this goal we have created our own CoppeliaSim model of a two wheeled vehicle with an attached plow. The CoppeliaSim Robot will be controlled by a python script running through the CoppeliaSim PythonAPI. The figure below shows an early model of the robot with the plow unfolded. It is important to note that even though only three proximity sensors are visible there are a total of twelve proximity sensors in total covering the robot's circumference as well as 4 proximity sensors mounted on the bottom of the robot. The plow on the front of the robot has two articulating arms that will be folded in and up when the robot is in the garage and then deployed once the robot leaves the garage in order to plow the snow.

The snow plow will work by working its way back and forth across the designated area in straight lines aiming to move from edge to edge pushing out the snow as it reaches each edge. In the case of the robot encountering an obstacle in its path, it will be programmed to manoeuvre around at the obstacle. In doing so, it will have a tendency to move into an area it has already cleared when passing an obstacle before moving back onto its line. This is to avoid not missing areas due to manoeuvring to avoid obstacles. The exact details of this algorithm are discussed later in the state and sequence diagram sections. Inspiration for our algorithm was devised by reading articles about similar autonomous robots pathfinding such as the Roomba[1].

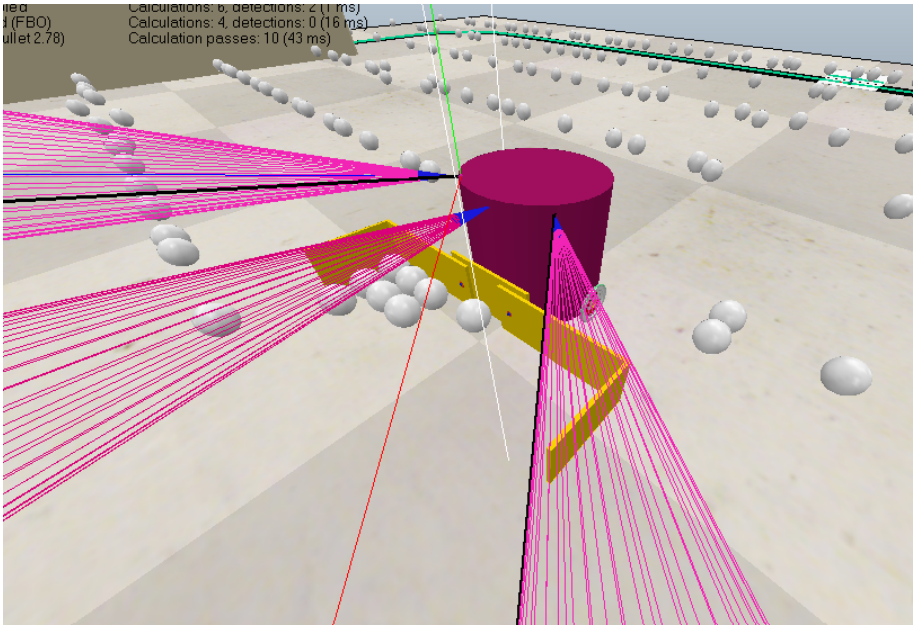

Figure 3: The Basic Plow Note: Some further changes have already been made to the design

Architecture Diagram

The following figure shows an Architecture Diagram of the system comprising the Robot running in CoppeliaSim and the Python code that controls the robot. The Python code is broken into components. The main loop runs the pathing and object avoidance algorithms

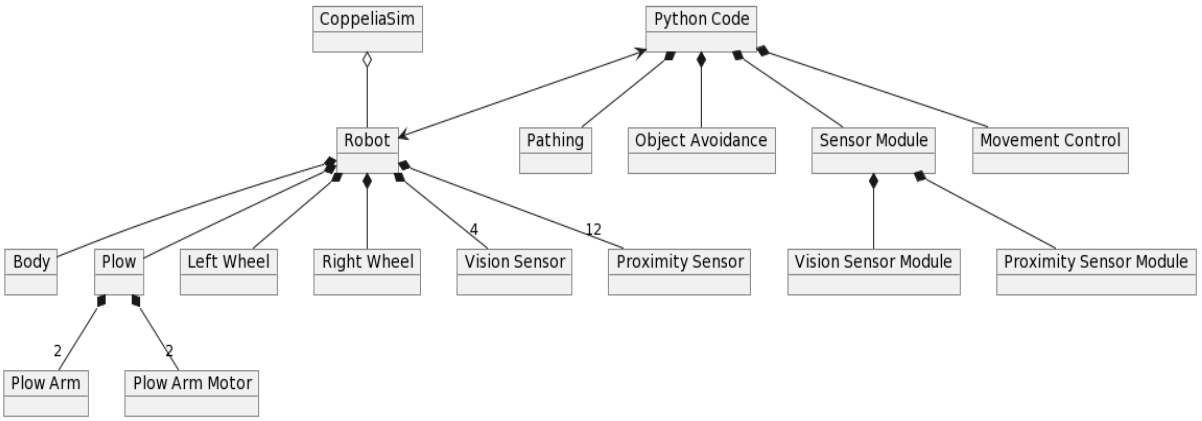

Figure 4: Overall Architecture of the System Components (Made with PlantUML[4])

The Plower itself consists of many components. There is the main body itself which is a cylinder to allow the plow to rotate on the spot. The articulating plow consists of two plow arms that are controlled by positionally controlled motors. Two independent wheels allow for the plow to drive forward, backward and rotate on the spot by controlling one wheel to drive forward while the other drives backwards. The final plow consists of one vision sensor placed under the front of the vehicle for detecting the edge of the map. We decided to use one sensor instead of four due to the time it took to receive data from coppeliaSim for four sensors. The ended up requiring 14 proximity sensors to cover the main circumference of the plow once the correct specifications were assigned to the sensors from the real world counterparts we selected.

The code written to control the coppeliaSim Plower was written in python and controls the plow through the CopeeliaSim API. The main algorithm of the code runs in a continuous loop polling both the proximity sensors and vision sensors. If an edge is detected the pathing algorithm is run to return The Plower back into the area for its next strip. If an object is detected the Object avoidance algorithm code is run to ensure we can avoid the object while still maintaining our desired trajectory. Each of these algorithms have recursive elements that are able to call each other in the case that edge detection is required during object avoidance or vice versa. Finally, independent modules were made for the sensors and for movement control so that receiving information from and controlling the plow would be a trivial and reliable task while the algorithms were being written. These modules contain functions like “readVisionSensor” that returns a boolean, or “turnRight” which turns The Plower 90 degrees clockwise. This added modularity and expandability to the code.

Real World Sensors

We have related our Proximity and Vision Sensors to two real world sensors. Ultrasonic Sensors were selected for the proximity sensors and Infrared Sensors for the vision sensors [2, 3]. Links to the found products and their datasheets can be found in the references section. The ultrasonic sensors would allow us to detect objects up to 4 metres away and in a 15 degree cone. Infrared sensors would act as the vision sensors and would be capable of detecting the black outline of the area.

State Chart

Included below is a State Chart Diagram for the plowing robot. Each node describes a state of the robot in it's algorithm and each line is a state transition controlled by a certain event or producing a specific result. In plain English, the algorithm of the plow is the following: The plow will begin by exiting the garage and deploying it's plow. It will then begin by turning East (by the movement control module code, based on the current variables of the system). It will then travel in a straight line, "Run", until an object or edge is detected. At an edge the robot will dump the snow, move 1 meter over and return to the area to plow the next strip. If an object is detected in the path of the robot it will turn "south" (the direction towards the area that has already been plowed) and make its way around the object before returning "north" to the strip it was currently plowing. Maintaining the strip that the robot is currently plowing will be done by the movement control module either by encoding the wheels positions to determine how far it moves south and north or by getting the global position of the robot from coppelia sim as if GPS or heading/position sensors were being used. The system will continue to loop through these states until the 5 minutes are up.

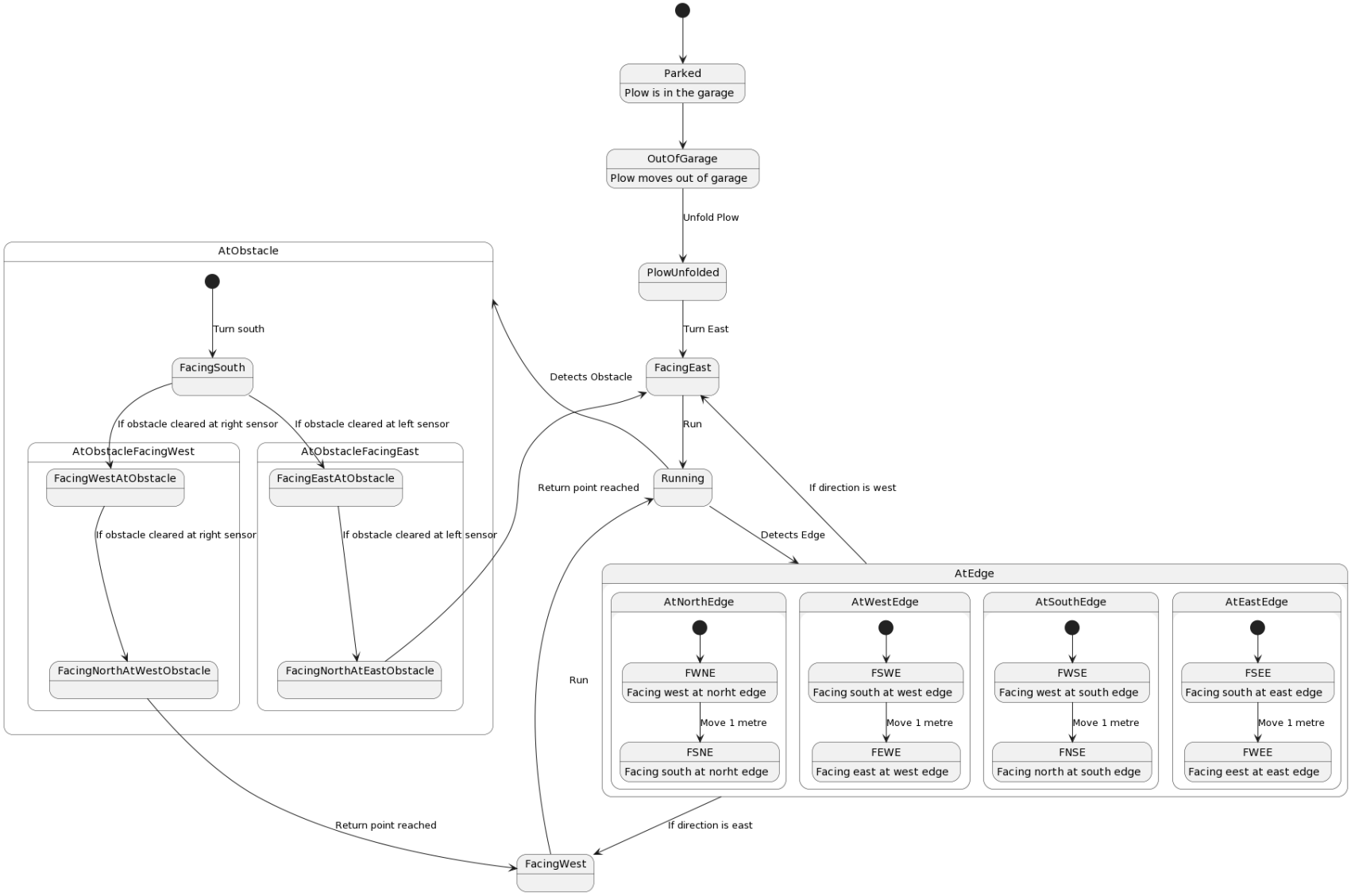

Figure 5: State Diagram of Snow Plow Robot by the Algorithm (Made with PlantUML[4]

Sequence Diagrams

The following sequence diagrams describe the behaviour of the plow in the various situations that it will encounter such as detecting the edge of the area where the snow must be released or encountering an object and how the plow will respond to these detections. The sequence diagrams read like flowcharts as there is very little message passing between components besides calls, commands, and API functions.

Sequence Diagram 1: Edge Line Detection

The plow polls the IR sensors (Vision Sensors) in a loop and listens for a detection. When the edge is detected by the back sensor of the plow, the plow will perform a sequence of moves to remove the snow from the plow and turn on to the next strip to plow. This movement sequence is described in the next sequence diagram.

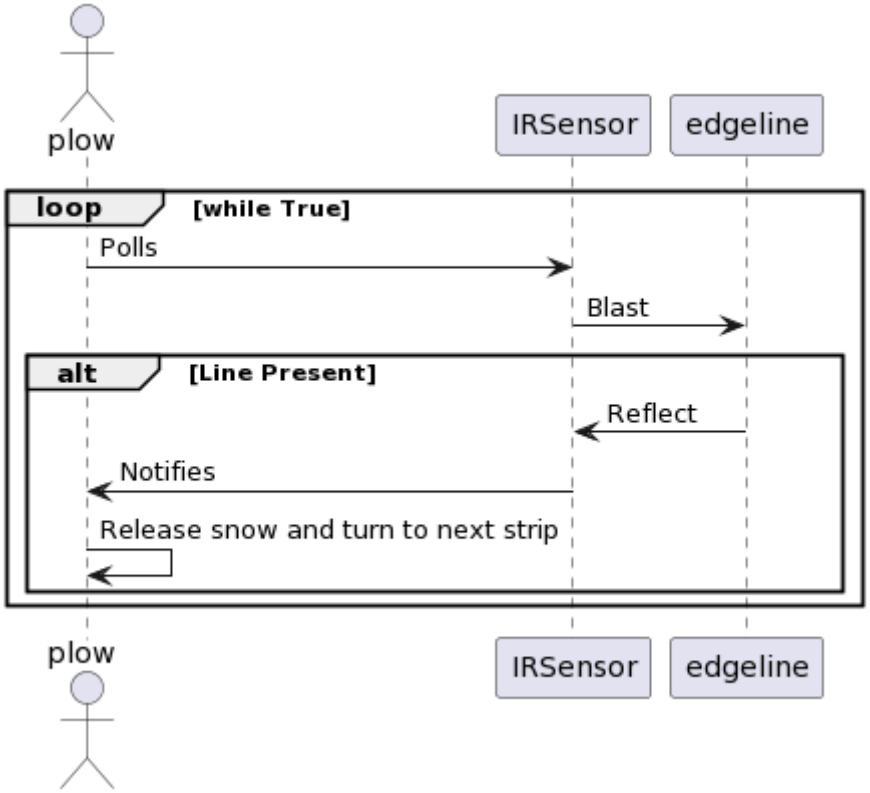

Figure 6: Sequence Diagram 1: Edge Line Detection (Made with PlantUML[4])

Sequence Diagram 2: Release snow and turn to next strip

When the plow detects that it has reached the edge of the area it will release the snow by making a quick forward motion to push the snow and then reversing. Then it will turn either to the left or right depending on the direction it was just travelling, move 1 meter over and turn another 90 degrees to prepare to enter the area and continue plowing.

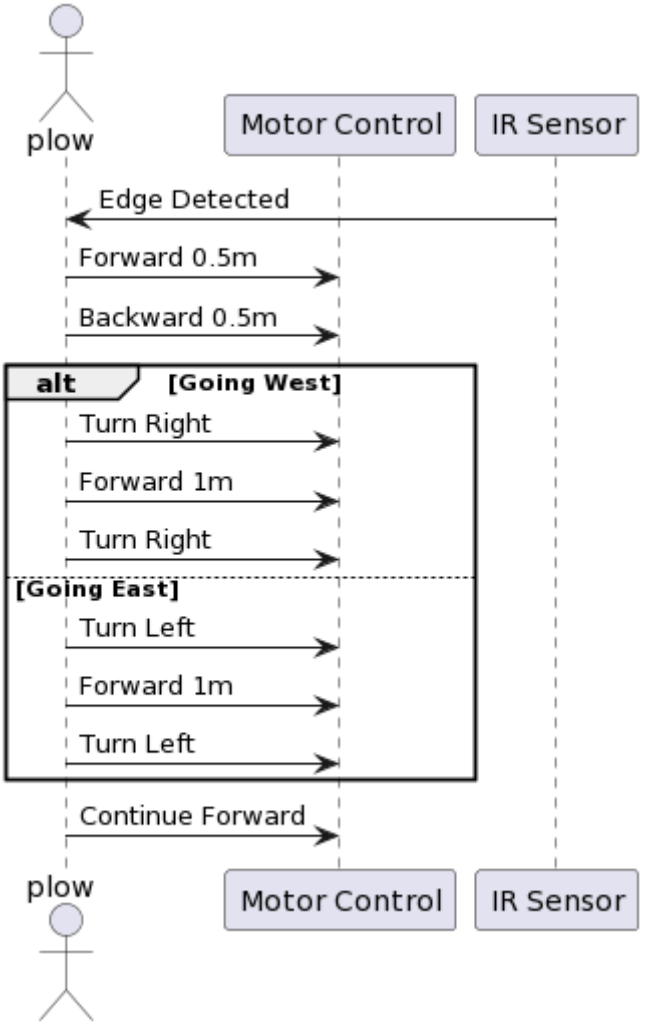

Figure 7: Sequence Diagram 2: Release snow and turn to next strip (Made with PlantUML[4])

Sequence Diagram 3: Object detected in path

The plow will periodically poll its ultrasonic sensors (proximity sensors) to check for objects. If an object is detected on one of the 5 forward facing ultrasonic sensors that the plow would hit the plow will run it's object avoidance sequence described in the next

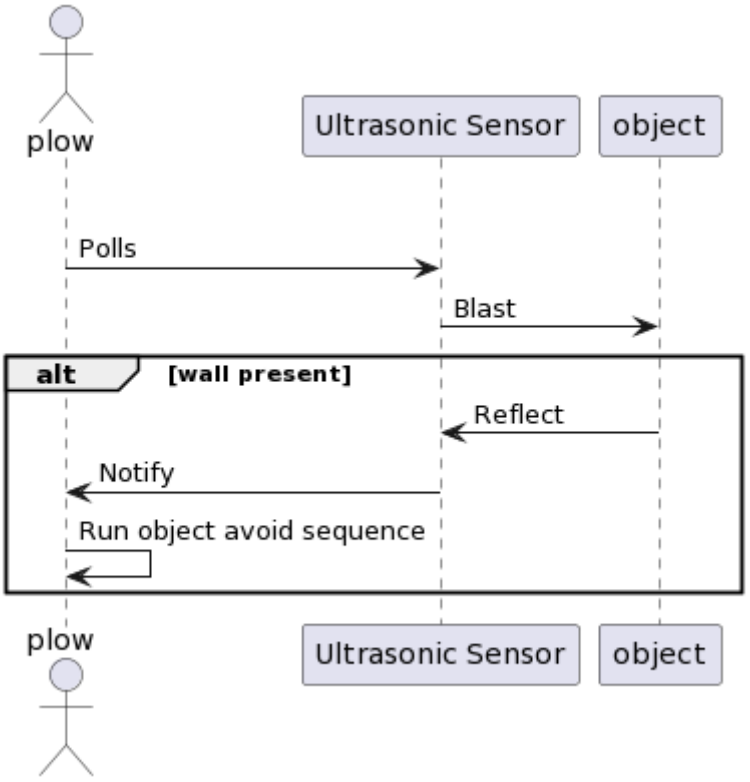

Figure 8: Sequence Diagram 3: Object detected in path (Made with PlantUML[4])

Sequence Diagram 4: Object Avoidance Sequence

When the plow has to avoid an object it will move south around the object and then go around the object and back north until it returns to the level it was on when it detected the object. For example, in the sequence diagram shown below where the plow was moving east, the plow will move south until it is “below” the object, then go east again until it is past the object, and then go north until it is back in line with the strip it was on before the object was detected.

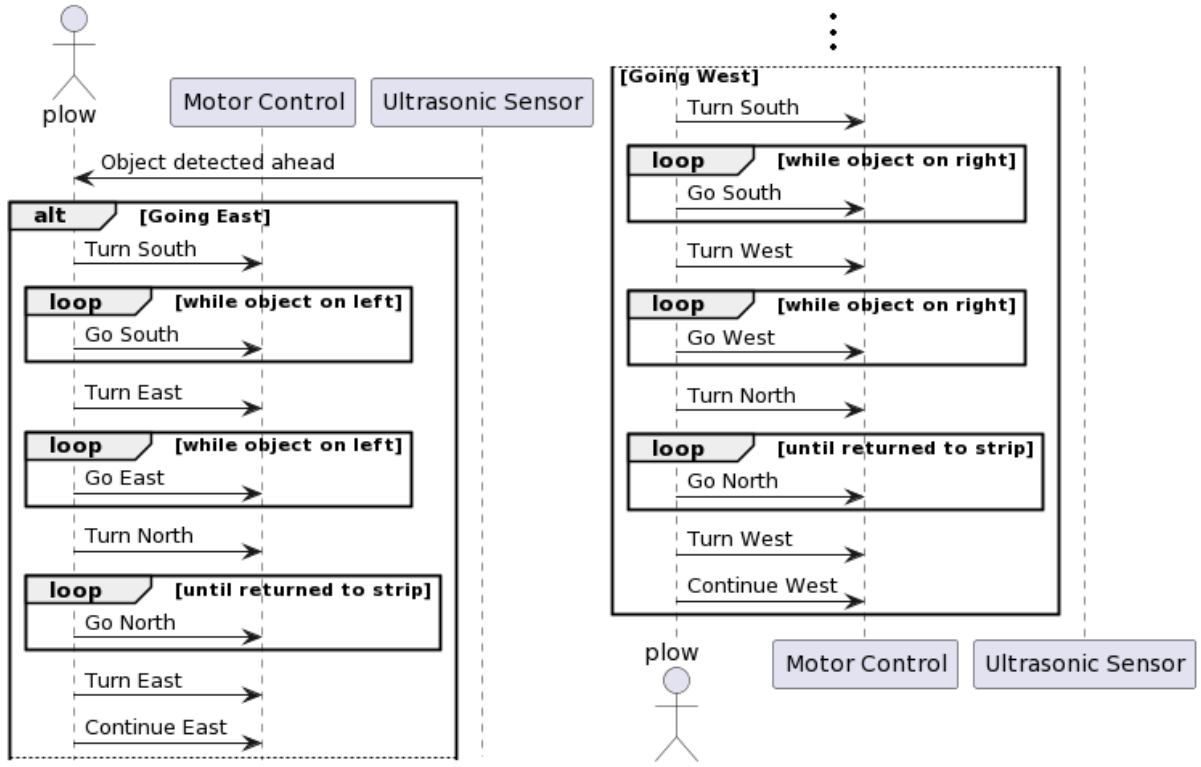

Figure 9: Sequence Diagram 4: Object Avoidance Sequence (Made with PlantUML[4])

Event vs. Time Driven

For this project, the approach chosen is a time driven one for the plowing vehicle. The main difference between a time driven approach and an event driven one is “what actions trigger events”. In the time driven approach used, a loop is used to check the status of all sensors, and said results are used to determine how the vehicle behaves in its surroundings. Whereas in an event driven scenario, the loop is not required to get status updates from the sensors, but rather the sensors will trigger an interruption when a predetermined objective is met. We determined that since we are using the PythonAPI for CoppeliaSim it would be more practical to use a time driven method. This also allows us to perform our whole algorithm in a loop of sensing, and then adjusting the movement of the robot, then sensing again and so on. SInce our code loop will run very quickly and our robot will be moving relatively slow in comparison a time driven loop is sufficient for the project.

Budget

For determining the project's budget we used the methods discussed in class and assigned salaries to ourselves as well as costs for the real world sensors we are “using” as well as a very rough estimate of what it would practically cost to build our robot body. Using the Earned Value Analysis method we developed budgets for our PV and BAC. Which can both be seen below. Based on Earned Value and Actual Cost estimations from our Gantt Chart Tasks We are slightly behind schedule on testing and therefore have earned value below our Planned Value. We also spent more time on the weekend and out of the lab period working on the project so that would make our actual cost greater than the planned value putting us over budget. The two figures below, 10 and 11, show the PV and BAC estimations respectively.

PV at Lab 6

AC of Project						
Labour Costs						
	Salary	# of Workers	# Hours a week	# Weeks Worked	Cost	
Worker Cost	$ 22.00	4	4	6	$2,112.00	
Overtime Contingency	$ 25.00	4	0	0	$ -	
Labour Expenditure					**$2,112.00**	
Material Expenses						
Sensors	Unit Cost	# of Units	# Contingency	Subtotal	Tax	Cost
Ultra Sonic	$ 4.50	12	0	$ 54.00	$ 7.02	$ 61.02
IR Sensors	$ 0.48	4	0	$ 1.92	$ 0.25	$ 2.17
$ spent of Sensors						*$ 63.19*
Mechanical Parts						
Wheel Motors	$ 30.00	2	0	$ 60.00	$ 7.80	$ 67.80
Plow Rotators	$ 20.00	2	0	$ 40.00	$ 5.20	$ 45.20
Body	$ 30.00	1	0	$ 30.00	$ 3.90	$ 33.90
Plow Metal	$ 50.00	1	0	$ 50.00	$ 6.50	$ 56.50
$ Spent on Mechanical						*$ 203.40*
Material Budget spent						**$ 533.18**
Spending so Far	**$ 2,645.18**					
Mangerial Reserve Spent	**$ 100.60**					
Actual Cost of project	**$ 2,745.78**					

Figure 10: Planned Value (PV) Budget to Date (Lab 6)

BAC

PV of Project						
Labour Costs						
	Salary	# of Workers	# Hours a week	# Weeks Worked	Cost	
Worker Cost	$ 22.00	4	4	10	$3,520.00	
Overtime Contingency	$ 25.00	4	3	4	$1,200.00	
Estimated Labour Costs					**$4,720.00**	
Material Expenses						
Sensors	Unit Cost	# of Units	# Contingency	Subtotal	Tax	Cost
Ultra Sonic	$ 4.50	12	4	$ 72.00	$ 9.36	$ 81.36
IR Sensors	$ 0.48	4	2	$ 2.88	$ 0.37	$ 3.25
Estimated Sensor Cost						*$ 84.61*
Mechanical Parts						
Wheel Motors	$ 30.00	2	1	$ 90.00	$ 11.70	$ 101.70
Plow Rotators	$ 20.00	2	1	$ 60.00	$ 7.80	$ 67.80
Body	$ 30.00	1	1	$ 60.00	$ 7.80	$ 67.80
Plow Metal	$ 50.00	1	0	$ 50.00	$ 6.50	$ 56.50
Estimated Mech Costs						*$ 293.80*
Estimated Material Costs						**$ 756.83**
Budget at completion	**$ 5,476.83**					
	Percent of BA	Reserve Amount				
Mangerial Reserve	10.00%	**$ 547.68**				
Planned Value of Project	**$ 6,024.51**					

Figure 11: Budget At Completion (BAC)

Github

The following is the link to our project's github repo. There we have maintained the work each member has completed in each of the labs so far as well as our projects main files, which is a compilation of our individual work into one functioning system.
https://github.com/Alfred-Akinkoye/Snow_Plower_4805
For posterity we have outlined below the work each student completed in each lab including the main work we did to complete this report.

Lab 4

- Build Plow Body (Alfred)
- Build Unfolding Plow (Edmond)
- Connect Python API (Michael)
- Determine Required Sensors and Positioning (Deji)

Lab 5

- Connect Sensors to Robot (Deji)
- Connect Sensors to PythonAPI (Alfred)
- Basic Movement Motor Control (Edmond)
- Connect Plow to Robot and Control (Michael)

Lab 6

The group as a whole is working on the progress report together. We had a meeting at the beginning of the lab period to discuss the pathfinding algorithm, and how that would be handled. Then we broke out to handle individual parts of the progress report.
Tasks:

- State Chart (Deji)
- Github, Event vs Time driven (Alfred)
- Budget (Edmond)
- Sequence Diagrams, Real World Sensor Comparisons (Michael)

References

[1] Connaughton, Niall, et al. "Roomba Algorithms and Visualization." *Code and a Glass of Wine*, 9 Mar. 2016, https://blog.niallconnaughton.com/2016/01/25/roomba-algorithms-and-visualization/

[2] "Ultrasonic Distance Sensor - HC-SR04." *SEN-15569 - SparkFun Electronics*, https://www.sparkfun.com/products/15569

[3] "0.48C$ 5% Off: IR Infrared Obstacle Avoidance Sensor Module for Arduino Smart Car Robot 3 Wire Reflective Photoelectric New: Sensor Module: Obstacle Avoidance Sensor Moduleavoidance Sensor - Aliexpress." *Aliexpress.com*, https://www.aliexpress.com/item/32321964595.html

[4] "Open-Source Tool That Uses Simple Textual Descriptions to Draw Beautiful UML Diagrams." *PlantUML.com*, https://plantuml.com

FINAL REPORT

Team Member Contributions

Over the course of the project work was completed mainly in each of the labs and often work was shared between team members. The following sections give a brief overview highlighting the main areas in which the students worked over the course of the project.

Edmond

Edmond designed the rotating plow system used to fold and unfold the plow. He also worked on developing both the edge detection algorithm that determines behaviour when the plow leaves the boundaries of the map and the object avoidance algorithm that determines the behaviour of the plow when it encounters an obstacle. He also designed the api calls used to determine the current location and the orientation of the plow which is used in navigation.

Michael

Michael worked on the implementation of the PythonAPI and the management of the python codebase. This included setting up the python code to communicate with CoppeliaSim. Michael built the framework for the project including a movementControl Module, Sensors Module, and API module. He also was mainly responsible for building the movement control module which allowed the plow to be controlled from python and move accurately in CoppeliaSim, this included movement and pushing testing as well as many other tests.

Deji

Deji worked primarily on the proximity sensors and vision sensors. He also did some work on the rotateTo method of the movement control module. With the proximity sensors, he added them to the API and wrote the methods involved with it. In initial model designs, he collaborated with Alfred to add the sensors to the main body. He also collaborated with Alfred in adding the vision sensors to the model. With the rotateTo method, in the movement control method, he wrote an algorithm that would quickly turn the robot to a desired direction without overshooting its target.

Alfred

Alfred worked on the base design for the vehicle model being used, the vision sensor functionality, and the edge control algorithm. He worked on testing the appropriate distance for the vision sensor as needed for our test maps, as well as how to use the data from the vision sensor to complement the edge control algorithm. He designed the base vehicle to be mobile, and have the ability to rotate on a spot, to allow for more delicate control on the overall machine.

New Components

GPS

We use a GPS [5] to keep track of the direction we are going, and position we are currently at. Like in our object avoidance algorithm we use our current y-coordinate position to re-align the vehicle when avoiding the obstacle, to ensure we are back on our original path.

Figure 12: The Real-World Counterpart for the GPS used to find our Plower's position in the map

Gyroscope

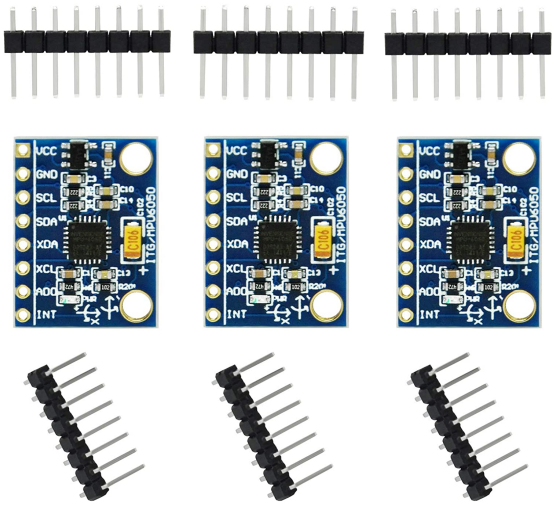

Figure 13: The Real-World Counterpart for the MPU used to find our Plower's orientation in the map

We use the gyroscope [6] to keep track of the vehicle's orientation, so as to be able to turn in any direction. This is used in both our edge detection and object avoidance algorithms. When we need to turn in a certain direction, or a certain amount, the gyroscope is useful in tracking what the orientation should be.

Budget at Completion

In the end the project was finished under budget compared to the projected value. The reason for this is how the project so far has only actually been a simulation. In the project budget given, money was left for spare parts under contingency because in real life it is recommended to have contingency parts in case some of them break but no parts were broken due to the project only being done in simulation. By the summation of the project the amount of overtime expected overshot the actual overtime worked leading to even more additional savings. The following figure shows the planned value of the project at completion.

PV of Project						
Labour Costs						
	Salary	# of Workers	# Hours a week	# Weeks Worked	Cost	
Worker Cost	$ 22.00	4	4	10	$3,520.00	
Overtime Contingency	$ 25.00	4	3	4	$1,200.00	
Estimated Labour Costs					**$4,720.00**	
Material Expenses						
Sensors	Unit Cost	# of Units	# Contingency	Subtotal	Tax	Cost
Ultra Sonic	$ 4.50	12	4	$ 72.00	$ 9.36	$ 81.36
IR Sensors	$ 0.48	4	2	$ 2.88	$ 0.37	$ 3.25
Estimated Sensor Cost						*$ 84.61*
Mechanical Parts						
Wheel Motors	$ 30.00	2	1	$ 90.00	$ 11.70	$ 101.70
Plow Rotators	$ 20.00	2	1	$ 60.00	$ 7.80	$ 67.80
Body	$ 30.00	1	1	$ 60.00	$ 7.80	$ 67.80
Plow Metal	$ 50.00	1	0	$ 50.00	$ 6.50	$ 56.50
Estimated Mech Costs						*$ 293.80*
Estimated Material Costs						**$ 378.41**
Budget at completion	**$ 5,098.41**					
	Percent of BAC	Reserve Amount				
Mangerial Reserve	10.00%	**$ 509.84**				
Planned Value of Project	**$ 5,608.26**					

Figure 14: Planned Value of the Project at completion

The figure on the next page shows the actual cost of the project. These were used to determine how under budget we ended up going and how the budget for the project progressed as a whole. Overall we are very happy with how the budget of this project turned out. Without the need for real hardware components and the complications that come with those as well as our generous budgeting for overtime we were able to come in well under budget for the project. Although out project did come in quite under the requirements the extra allotted budget would likely have to be put into mitigating the damages from our requirements agreement and if the project timeline did not have a hard cut off we would likely go very much over budget completing the project to meet all the set-out requirements.

AC of Project						
Labour Costs						
	Salary	# of Workers	# Hours a week	# Weeks Worked	Cost	
Worker Cost	$ 22.00	4	4	10	$3,520.00	
Overtime Contingency	$ 25.00	4	4	2	$ 800.00	
Labour Expenditure					**$4,320.00**	
Material Expenses						
Sensors	Unit Cost	# of Units	# Contingency	Subtotal	Tax	Cost
Ultra Sonic	$ 4.50	12	2	$ 63.00	$ 8.19	$ 71.19
IR Sensors	$ 0.48	4	0	$ 1.92	$ 0.25	$ 2.17
GPS	$ 58.00	1	0	$ 58.00	$ 7.54	$ 65.54
Gyroscope	$ 20.58	1	0	$ 20.58	$ 2.68	$ 23.26
$ spent of Sensors						*$ 162.16*
Mechanical Parts						
Wheel Motors	$ 30.00	2	0	$ 60.00	$ 7.80	$ 67.80
Plow Rotators	$ 20.00	2	0	$ 40.00	$ 5.20	$ 45.20
Body	$ 30.00	1	0	$ 30.00	$ 3.90	$ 33.90
Plow Metal	$ 50.00	1	0	$ 50.00	$ 6.50	$ 56.50
$ Spent on Mechanical						*$ 203.40*
Material Budget spent						**$ 365.56**
Spending so Far	**$ 4,685.56**					
Mangerial Reserve Spent	**$ 200.12**					
Actual Cost of project	**$ 4,885.68**					

Figure 15: Actual Cost of the Project (Budget at Completion)

Control Charts

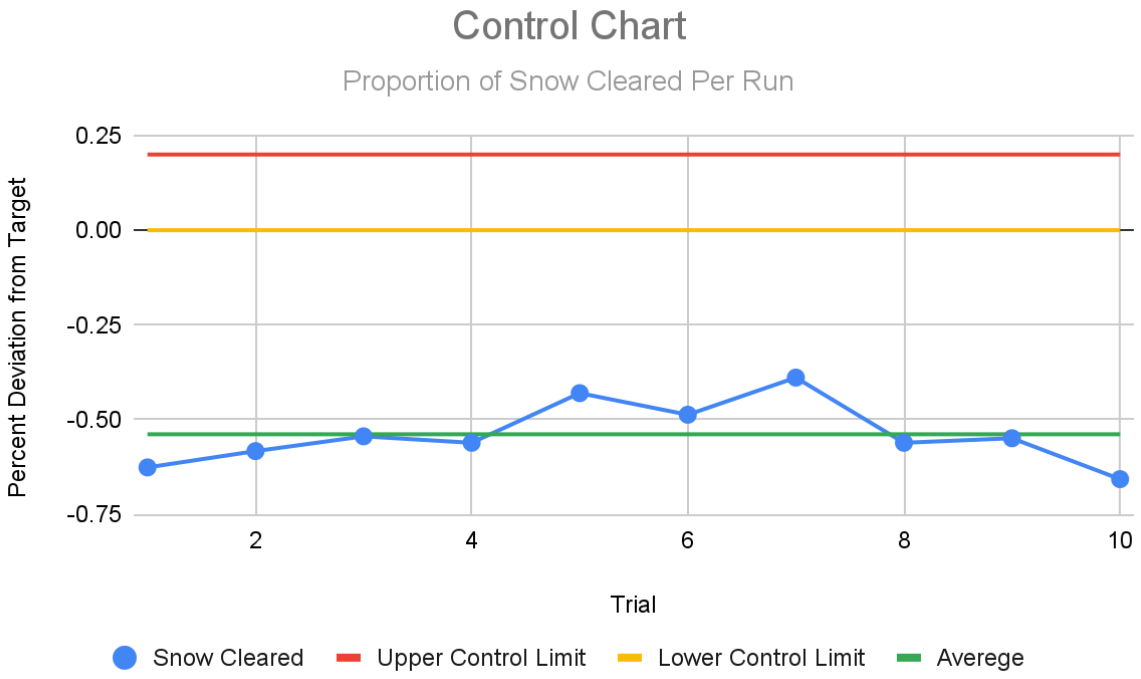

Figure 16: Proportion of Snow Cleared in Each Trial Run

Depicted in Figure 14 is the control chart for the proportion of snow cleared in each trial run. On the x-axis, the trial run number is displayed. Displayed on the y-axis is the percent deviation from the target. The target was to clear a minimum of 80% of total snow volume in each trial run. For this reason, the lower control limit (LCL) was set at the target. This is indicated by a yellow line at 0.00 on the vertical scale.

The upper control limit (UCL) is set at 100% snow removal. Since this is 20% points above 80%, the UCL is set at 20% deviation from target. This is indicated by a red line at 0.20 on the vertical scale.

An average of 26.1% of snow was removed during the tests. This fell outside the control window by an average of -53.9% from the target. This is indicated by the green line at -0.54 on the vertical scale.

There was initial difficulty in designing an algorithm to clear snow efficiently from each map. Thus, a maximum of 5% snow removal was achieved in the initial stages. The turn speed and plower acceleration was found to be too slow. The turn speed was increased but frequently overshot its target, causing the robot to spin many times before stopping at its target direction. Additionally, the edge detection algorithm proved to be a challenge, as the robot would not detect the edge. This caused it to continue driving and fall off of the map.

To remedy this, a more robust edge detection algorithm was developed. On top of this, an advanced turn algorithm was developed and further improved to quickly spin the robot and stop at its target direction without overshooting. This allowed the robot to improve

from a 5% snow removal to a 26.1% snow removal. Given one or two more weeks to refine the algorithm, developers believe it could be improved to hit its target at 80% or more.

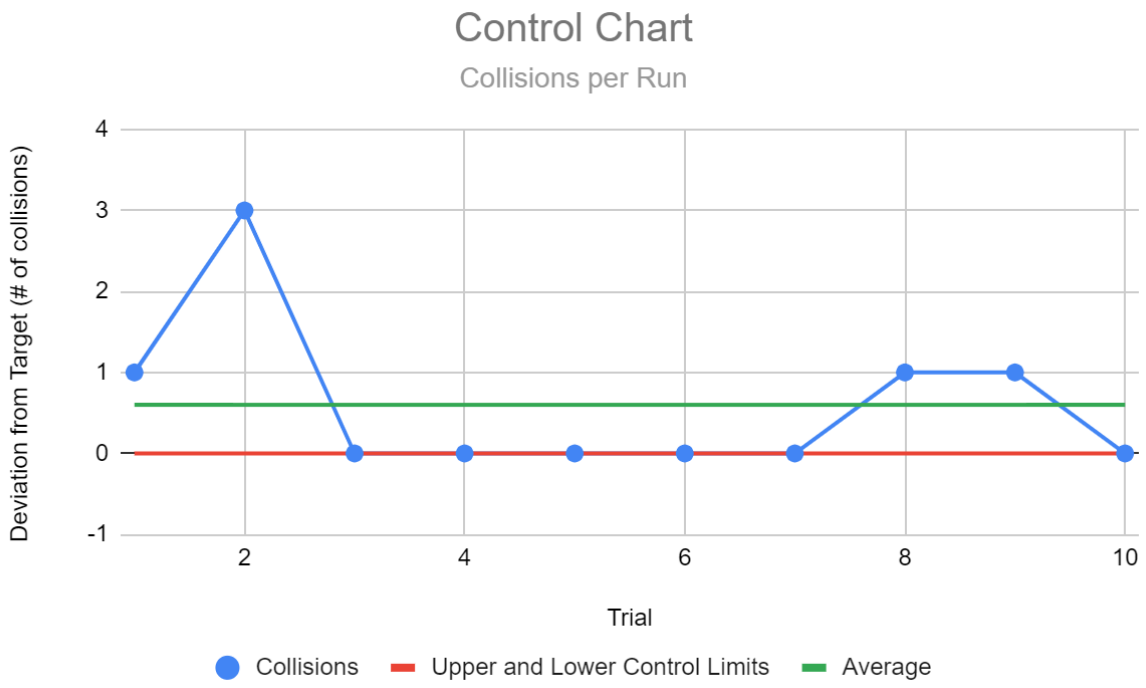

Figure 17: Collisions in Each Trial Run

Depicted in Figure 15 is the control chart for the amount of collisions suffered in each trial run. A hard limit was set at 0 collisions for the trial runs. Therefore, both the UCL and LCL were set at the target of 0 collisions.

The robot initially collided with objects on every prior run. Once the final object detection algorithm was developed, this improved to an average of 0.6 collisions per trial run. This is indicated by a green line at 0.6 on the vertical scale.

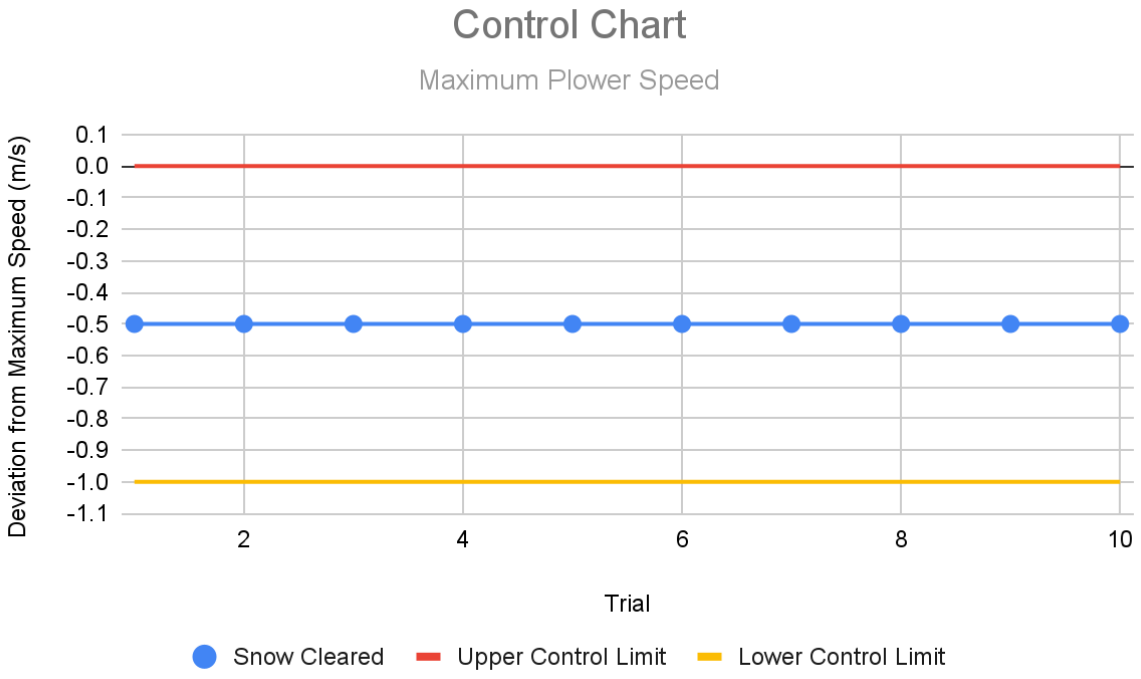

Figure 18: Maximum Plower Speed in Each Trial Run

Depicted in Figure 15 is the control chart for the maximum plower speed. The speed was not to exceed 2 m/s. The UCL was therefore set at the target of 2 m/s. This is indicated by the red line at 0.0 on the vertical scale. The LCL was set at 1m/s below the target. This is indicated by the yellow line at -1.0 on the vertical scale.

In each trial run, the maximum speed was 1.5 m/s. This is 0.5 m/s below the target and within the control limits. This is indicated by the blue line at -0.5 on the vertical scale.

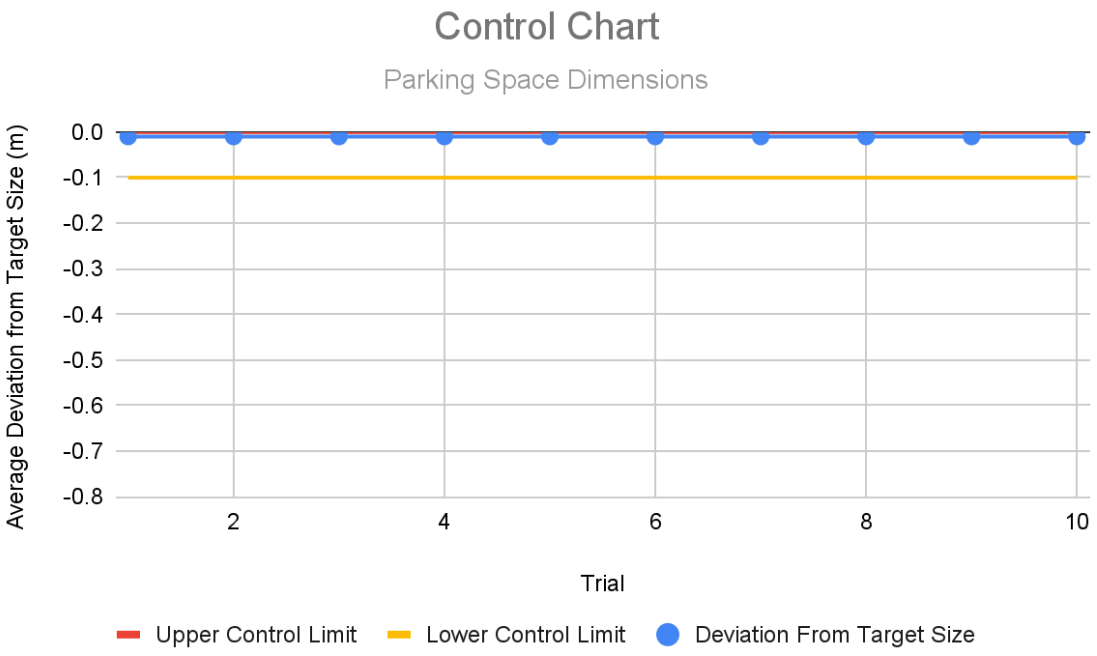

Figure 19: Deviation from Target Parking Space Dimensions

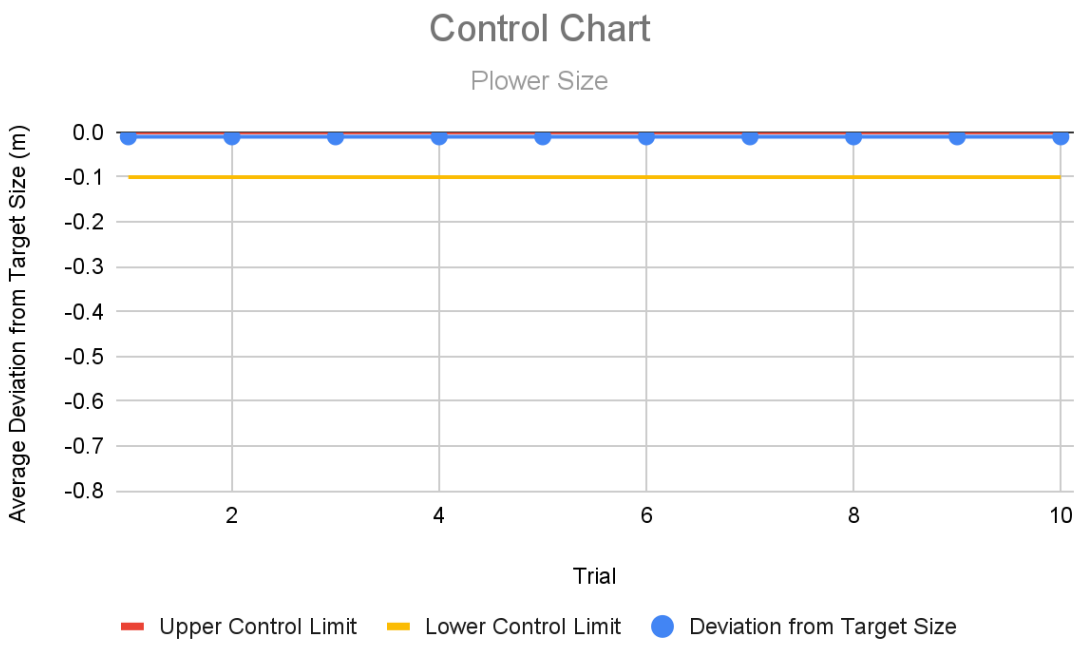

Figure 20: Deviation from Target Plower Size

Depicted in Figures 15 and 16 are the control charts for the parking space dimensions and plower size dimensions respectively. In both cases, the UCLs are set at their target dimensions. The LCL is set at a deviation of 0.1 m below the target for each dimension. In each trial run, both the parking space dimensions and plower size dimensions remain constant at the target. This is indicated by the blue line at 0.0 on the vertical scale in both charts.

Results of Training Maps

The completed Plower was run on each of the Training Maps and Test Maps provided to collect metrics and results on our finished product. The following four examples describe the successes and failures of the plower on each of the training maps provided. The data collected from these runs and additional runs on the test maps is included in the Control Charts above.

In addition to the four examples shown below the results of the three test maps were the following:
Test_Map_1: 109 pieces cleared (23%), 0 Collisions
Test_Map_2: 182 pieces cleared (37%), 0 Collisions
Test_Map_3: 154 pieces cleared (31%), 0 Collisions

Training Map 1:

Ironically, the plower performed worst on Training_Map_1. Unfortunately this map does not track the amount of snow pushed out of the area but from the figure below of the training map you can see only a few initial passes were completed before the plower fell off the map. This was due to a bug in the object avoidance algorithm when crossing the area. Our object avoidance algorithm took priority and we were able to avoid all collisions in this run but the bug in our algorithm did cause our plower to fall off the map after only a few passes. This is a rare occurrence but this example is included to display the failures of our project rather than just the successes.

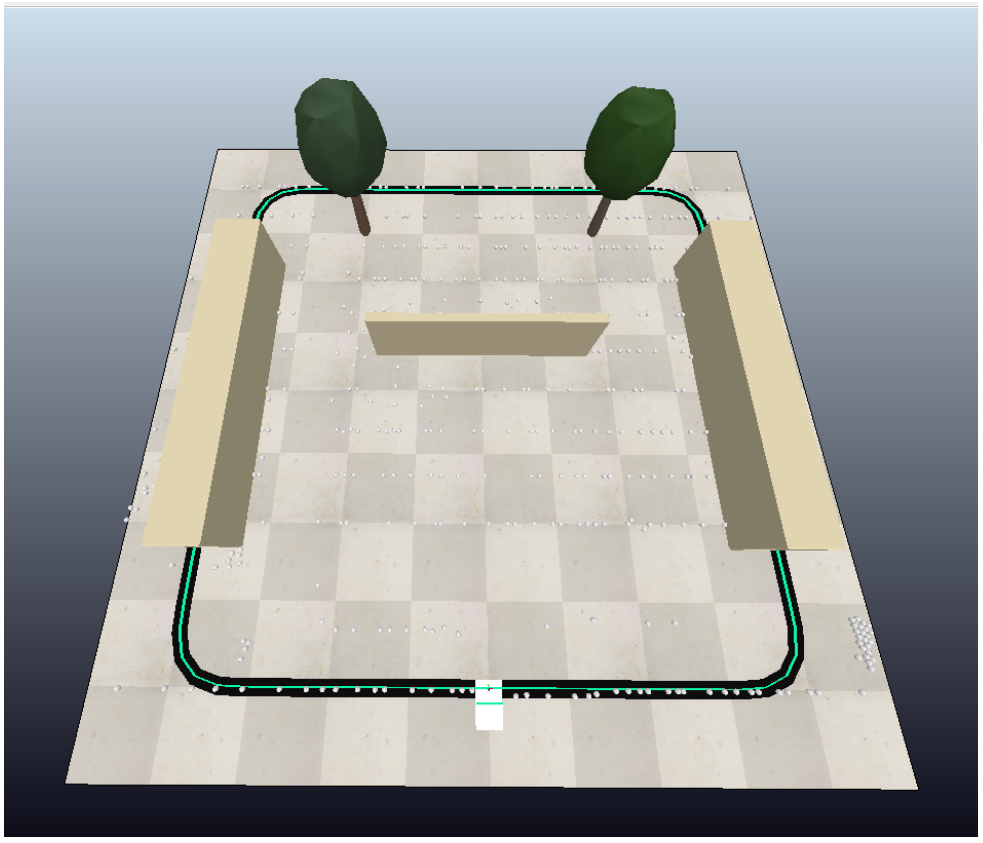

Figure 21: Result of Training_Map_1 (Example of an Erroneous Run)

Training Map 2:

Training_Map_2 provided the best result we were able to achieve on any of the maps. Clearing 40% of the snow (185 pieces) without colliding with any objects. Further runs of training map two provided varying results but the figure below gives a good indication of the very best performance our plower can achieve within the time constraints.

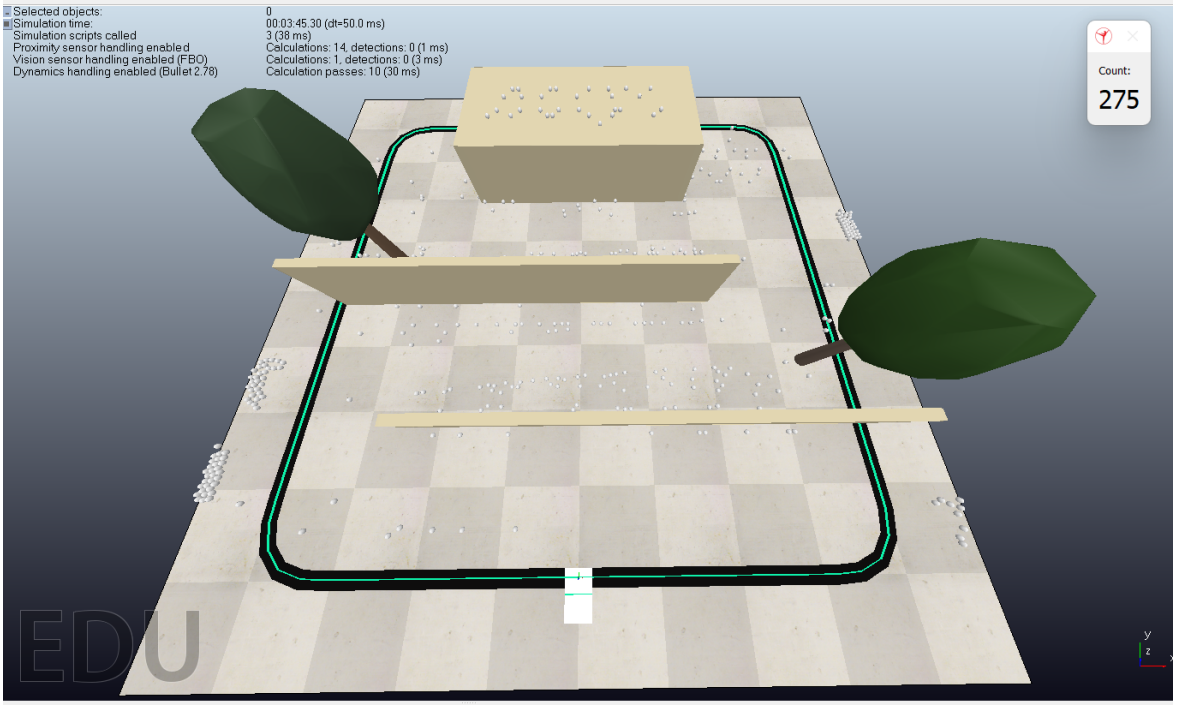

Figure 22: Result of Training_Map_2. (Example of an Optimal Run)

Training Map 3:

On Training_Map_3 we were able to clear 119 pieces of snow (~24%) however this map was the most susceptible to collisions. Mostly this man resulted in 1 or no collisions but one run involved 3 collisions in total, with the plow being hit by the moving pedestrian twice and eventually being flipped over by the pedestrian. In a real world situation we would hope that pedestrians would not intentionally attack our plow and the plow wouldn't have to actively get out of their way.

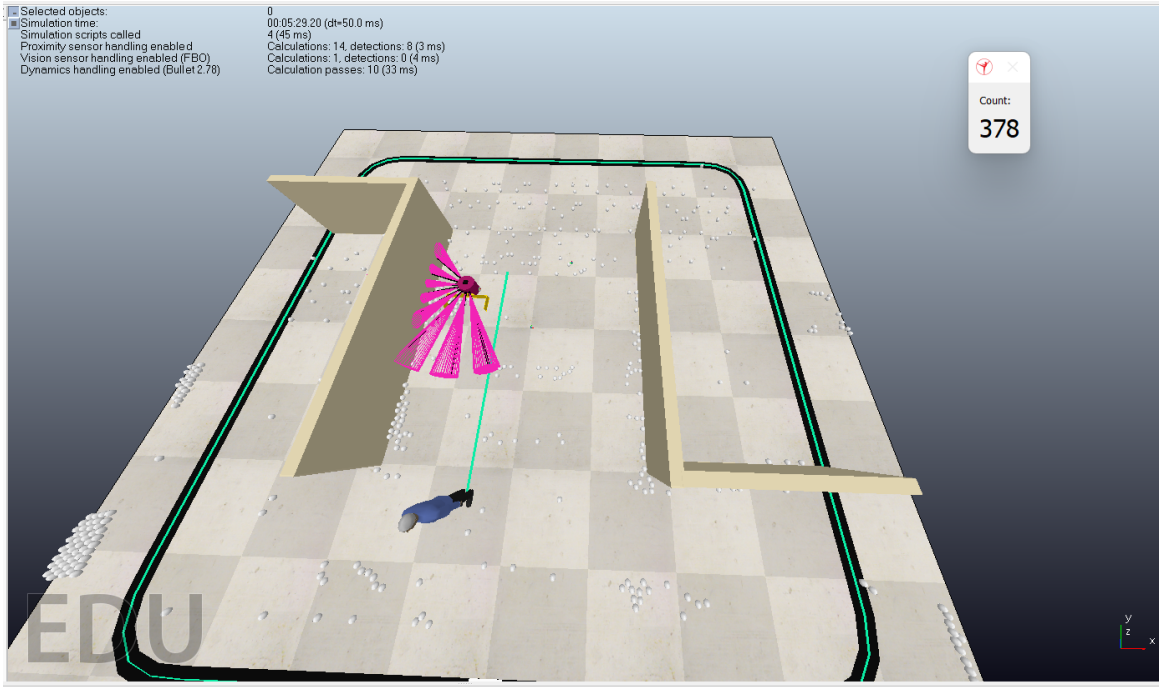

Figure 23: Result of Training_Map_3 (Final State of the Map after the run)

Training Map 4:

Training_Map_4 was likely the most technically challenging map due to the diagonal walls but our plow was able to do relatively well thanks to its recursive object avoidance algorithm. On the run shown in the figure below the plow was able to clear 125 pieces of snow (~26%) without any collisions. Our plow did end up falling off of this map due to complications in the asynchronous timing between coppeliaSim and our python code but the run but the majority of the time had already passed before this issue occurred.

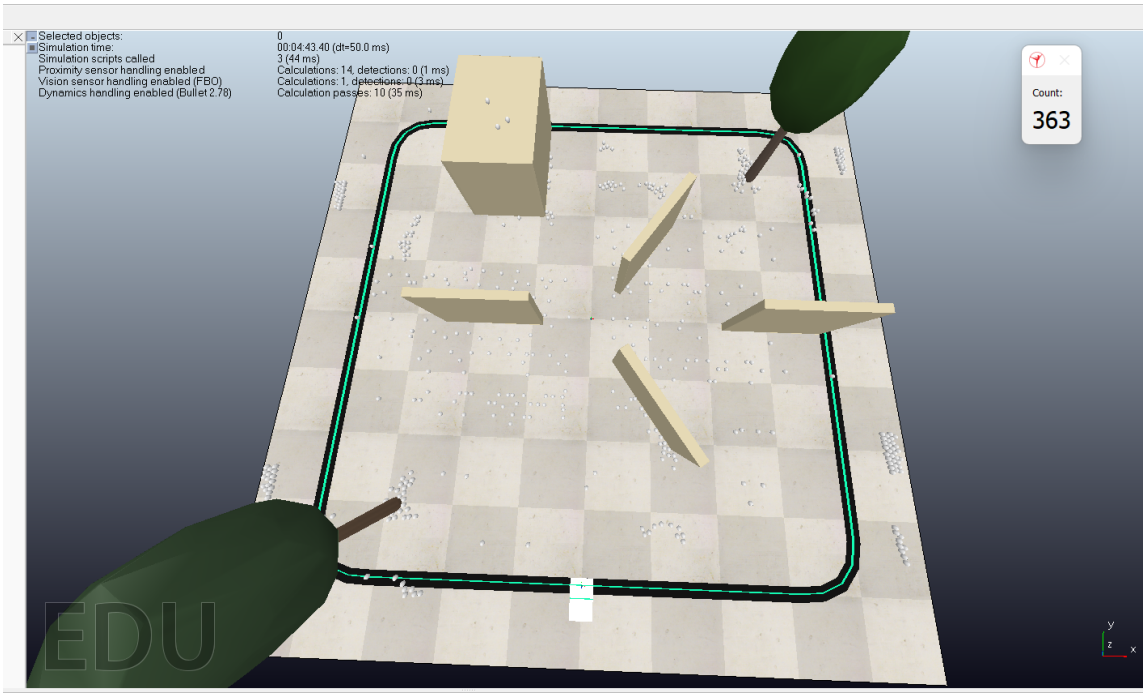

Figure 24: Result of Training_Map_4 (Final State of the Map after the run)

Completed Github:

Our Github (https://github.com/Alfred-Akinkoye/Snow_Plower_4805) has been cleaned up to only contain the required code for the project on the main branch. Our lab work and other files can be found on the “additional-work” branch. This report has also been uploaded to Github and the entire repo has been zipped and submitted on brightspace. Additional information about the project and instructions for running the project can be found in the Github README.

References

[5] GPS-15210 SparkFun, Mouser, https://www.mouser.ca/ProductDetail/SparkFun/GPS-15210?qs=Zz7%252BYVVL6bGf8coET7CrKg%3D%3D&mgh=1&gclid=Cj0KCQjwl7qSBhD-ARIsACvV1X0z_FFDV1g8XGKlzZkdBJQLZs64M2LXR-KBVIPCMVBWMaAkfppCP-EaAkISEALw_wcB

[6] Gikfun GY-521 MPU-6050 3 Axis Accelerometer Gyroscope Module 6 DOF 6-axis Accelerometer Gyroscope Sensor Module for Arduino DIY(Pack of 3) EK1091x3C, https://www.amazon.ca/Gikfun-MPU-6050-Accelerometer-Gyroscope-EK1091x3C/dp/B07JPK26X2/ref=sr_1_5?keywords=Gyroscope+Sensor&qid=1649279429&sr=8-5\