SYSC 4805 - Computer Systems Design Lab

Project Progress Report

Tuesday April 12, 2022

Group 2 - L1

Nirda Ali - 101045488
Cameron Chung - 101118926
Collins Mbogori - 101088609
Mario Shebib - 101035000

Table of Contents

1.0 Introduction

This report is made to fulfill the requirement for the SYSC 4805 design lab. Our team is called The Amethyst. This report covers the entire project progress over the course of the Winter 2022 semester of SYSC 4805.

2.0 Project Charter

2.1 Objective

The purpose of this project is to create an autonomous snowplow robot within simulation software. The snowplow will be created with CoppliaSim software. Its objective is to remove snow and avoid obstacles in test scenes.

2.2 Deliverables

The deliverables for this project will include a project proposal outlining the plan for the project, a progress report giving an update halfway through the semester, and a final report/presentation to demonstrate our robot simulation. The work for the project will be a CoppeliaSim model that has a robot to the specification provided, which will be able to remove snow from a path without hitting any obstacles.

3.0 Scope

3.1 Requirements

3.1.1 What The Robot Must Do

- When the simulation starts, the robot shall start at the following position (x, y) = (0, -6.25) meters.
- When the simulation starts, the robot shall be in the following orientation relative to the world (Alpha, Beta, Gamma) = (90, 0, 90) degrees.
- When the simulation is running, the robot size will be 0.5m x 0.8m x 1m.
- When the simulation is running, the robot will be at a maximum speed of 2m/s.
- The simulation will only run for a maximum of 5 minutes.
- The sensors used in the robot design will coordinate with real-life sensors that can be purchased.
- When the simulation is running, the robot will stay within the black path border.
- When a snowball is encountered, the robot will push it to the black path.
- When the proximity sensor senses a static obstacle, the robot will maneuver around it.
- When the proximity sensor senses a dynamic obstacle, the robot will maneuver around it.
- When the simulation is running, the robot will move through the map and remove snow by pushing it outside the black path
- When the vision sensors detect the black path, the robot will move back and rotate to turn around.

3.1.2 What The Robot Must Not Do

- When the simulation is running, the robot size must not exceed 1m x 0.8m x 1m.
- When the simulation is running the robot can not go faster than 2m/s.
- When the robot is at the black path it will not go past it.
- When the simulation is running, the robot will not fall off the edges of the training map.
- When a static obstacle is encountered, the robot will not pass through it.
- When a dynamic obstacle is encountered, the robot will not pass through it.

The sensors used on the robot for detecting snow or other properties through CoppeliaSim must match up to real-life sensors that can be purchased. We will be using LUA for programming the robot within the CoppeliaSim software. Both ultrasonic and infrared style proximity sensors will be utilized to help the robot safely avoid both moving and stationary obstacles. This will include a camera/vision sensor, accelerometer, force sensor and angle sensor. Real-life examples of the sensors that will be used for the robot can be found in the appendix. There are three components for controlling the movement of the robot. These include the camera/vision sensor that will be used for viewing the landscape. As well as the accelerometer that will be used to control the robot speed Additionally, there are multiple sensors that are needed, one is a force sensor to measure when the snow forces on the plow. An angle sensor for measuring the inclination of the robot's plow and adjusting its angle. With an ultrasonic proximity sensor for detecting both static and moving objects. Using these sensors, we will program the robot to remove snow from a path and avoid obstacles as outlined in the specification.

3.2 Activities

There are 4 members in this group and the project spans seven weeks. Each week, every member will work on a specific task.

The tasks involved with creating the robot are the following:

1. Configure robot body on landscape
2. Build a snowplow
3. Attach the snowplow to the robot body
4. Configure robot starting position
5. Configure robot starting angle
6. Configure snowplow to move with robot body
7. Program snowplow to move on path
8. Add and configure ultrasonic proximity sensor
9. Program robot to avoid static obstacles
10. Program robot to avoid moving obstacles
11. Add and configure camera
12. Program vision sensor to detect the path in landscape
13. Program vision sensor to detect snowballs in landscape
14. Program vision sensor to detect obstacles in landscape
15. Add and configure angle sensor to snowplow
16. Program snowplow to move based on angle detection
17. Add and configure force sensor
18. Program snowplow to adjust torque based on the force of snowballs

19. Add and configure accelerometer to robot body
20. Program robot speed range using accelerometer readings
21. Test robot starting position and angle
22. Test robot’s ability to follow path
23. Test robot’s ability to avoid static objects
24. Test robot’s ability to avoid dynamic objects
25. Test robot’s ability to move snowballs
26. Test robot on training map 1
27. Test robot on training map 2
28. Test robot on training map 3
29. Progress report
30. Final report
31. Final presentation script
32. Final presentation slides

3.3 Testing

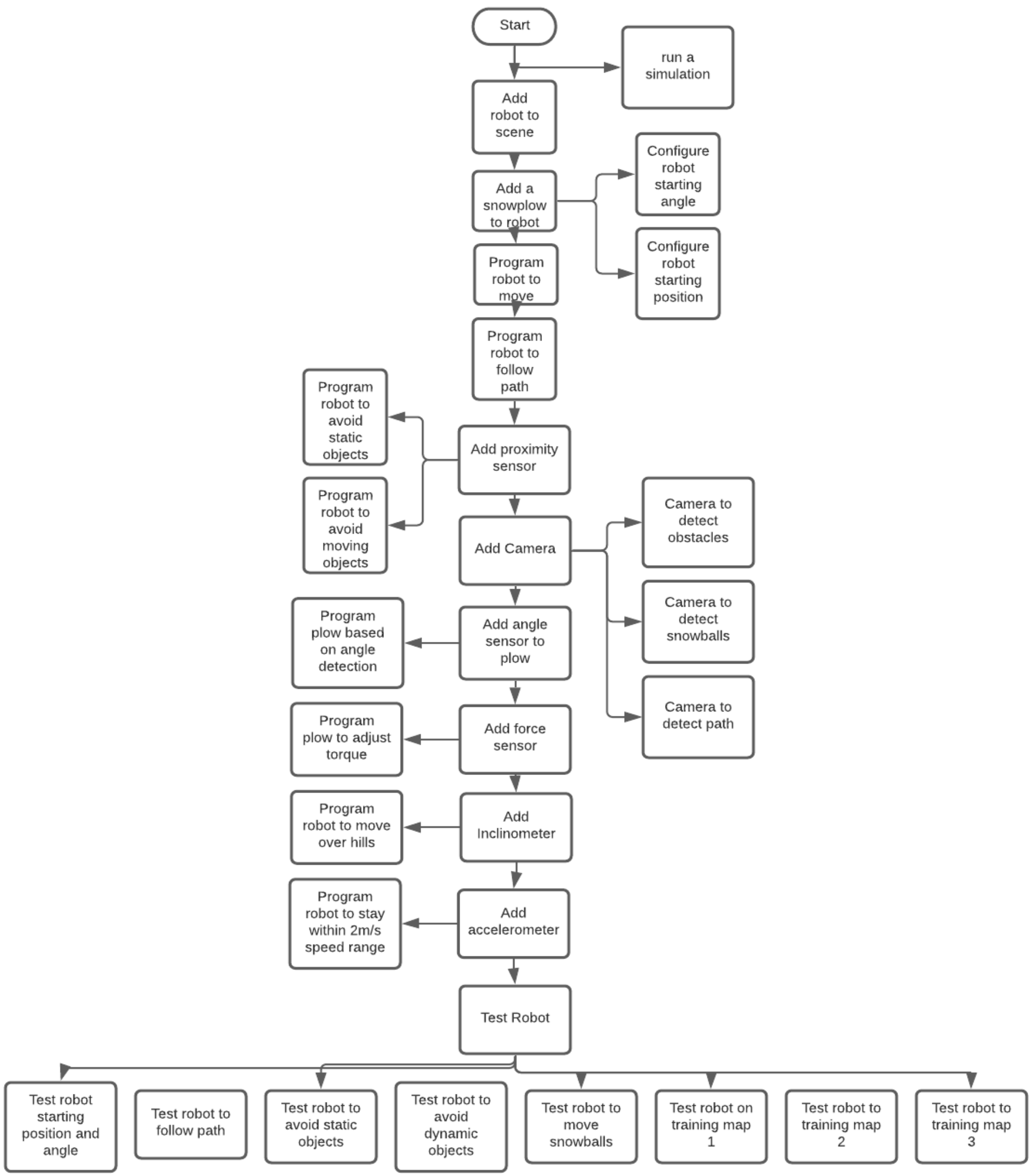
The robot was tested to ensure it meets the specification provided. The tests will include three main sections that will encompass the provided requirements. The first section of tests will ensure the robot is made to the proper specification, this includes the size, position, and orientation. The second section of tests will ensure the robot moves properly, thus following the path and avoiding obstacles. The last section of testing will analyze the robot’s ability to remove snow from the path.

Test ID	Test Question	Success Criteria	Pass/Fail Result
1	Does the robot body remain intact when the simulation is started?	Upon pressing play for the simulation the robot body components remain intact	Pass
2	Does the robot start at the correct position?	Robot’s starting potion is (x, y) = (0, -6.25)	Pass
3	Does the robot start at the correct orientation?	Robot’s starting orientation is (Alpha, Beta, Gamma) = (90, 0, 90) degrees	Pass
4	Is the robot the correct size?	Robot size is measured to be at or below these dimensions: 0.5m x 0.8m x 1m	Pass
5	Does the robot maintain the correct size during simulation?	Robot size is measured to be at or below these dimensions during simulation: 1 m x 0.8 m x 1m	Pass
6	Can the robot avoid static obstacles?	Upon reaching a static obstacle the robot is able to maneuver around it without making contact	Fail
7	Can the robot avoid dynamic obstacles?	Upon being confronted by a moving obstacle the robot is able to maneuver around it without making contact	Fail

8	Can plow design collect snow?	When the robot reaches a snowball is able to collect it in the plow	Pass
9	Can the robot move snow?	When the robot reaches a snowball it is able to push it across the scene	Pass
10	Can the robot remove snow from the scene?	When the robot comes into contact with the snow is able to push it out past the black border	Fail
11	Does the robot stay within physical boundaries?	The robot does not go past any black border	Pass
12	Does the robot run at specified speed?	Robot speed does not exceed 2m/s	Pass

4.0 Schedule

4.1 Schedule Network Diagram



4.2 Gantt Chart

Task Number	Task	Lab 4	Lab 5	Lab 6	Lab 7	Lab 8	Lab 9	Lab 10
1	*Configure robot body*							
2	*Build snowplow*							
3	*Attach snowplow to body*							
4	*Configure starting position*							
5	*Configure starting angle*							
6	*Configure snowplow to move with body*							
7	*Program snowplow to move on path*							
8	*Add and configure proximity sensor*							
9	*Program robot to avoid static obstacles*							
10	*Program to avoid moving obstacles*							
11	*Add and configure camera*							
12	*Program vision sensor to detect path*							
13	*Program camera to detect snowballs*							
14	*Program camera to detect obstacles*							
15	*Add and configure angle sensor*							
16	*Program snowplow for angle adjustability*							
17	*Add and configure force sensor*							
18	*Program snowplow to adjust torque*							
19	*Add and configure inclinometer*							
20	*Program robot to move over hills/bumps*							
21	*Add and configure accelerometer*							
22	*Program robot speed range*							
23	*Test robot starting position and angle*							
24	*Test robots ability to follow path*							
25	*Test robots ability to avoid static objects*							
26	*Test robots ability to avoid dynamic objects*							
27	*Test robots ability to move snowballs*							
28	*Test robot on training map 1*							
29	*Test robot on training map 2*							
30	*Test robot on training map 3*							
31	*Progress report*							
32	*Final report*							
33	*Final presenation script*							
34	*Final presentation slides*							

5.0 Human Resources

Responsibility Assignment Matrix

Step	Project Initiation	Member Responsible	Member Approver
Configure robot body on landscape	Design and implement the snow plow body and its wheels, ensure the snow plow can move	Nirda Cameron	Collins Mario
Build a snow plow	Design a plow that can push snow to the side of the path efficiently	Nirda	Collins Cameron Mario
Attach the snow plow to the robot body	Make sure the movable plow is attached to the robot body and doesn't fall off	Cameron Nirda	Collins Mario
Configure robot starting position	Adjust the snow plow to be on the same coordinates as specified in the project description	Cameron	Nirda Collins Mario
Configure robot starting angle	Adjust the snow plow to have the same orientation as specified in the project description	Collins	Nirda Cameron Mario
Configure plow to move with robot body	Add functionality to enable the plow to move on its axis and adjust its angle	Mario	Nirda Cameron Collins
Program robot to move on the path	Code the snow plow using Lua to follow the lined path using a vision sensor	Cameron	Nirda Collins Mario
Add ultrasonic proximity sensor and configure	Position and configure ultrasonic sensor to detect position of objects to avoid obstacles using ultrasonic sound waves	Mario	Nirda Cameron Collins
Program robot to avoid static obstacles	Code using Lua to enable the robot using an ultrasonic sensor and a proximity sensor to change its trajectory around stationary objects	Nirda	Mario Cameron Collins
Program robot to avoid moving obstacles	Code using Lua to enable the robot using an ultrasonic proximity sensor to change its trajectory and move around or stop before colliding with objects in motion	Collins	Nirda Cameron Mario

Add and configure the vision sensor to robot	Position on the snow plow to visualize the trajectory and path of the robot and the plow's functionality	Nirda	Cameron Collins Mario
Program vision sensor to detect the path in landscape	Position at the bottom of the snow plow and code in Lua to be able to detect the black coloured path and allow the robot to follow the line path	Cameron	Nirda Collins Mario
Program vision to detect snowballs in landscape	Position at the front of the snow plow and code in Lua to detect snowball objects	Mario	Nirda Cameron Collins
Program vision sensor to detect obstacles in landscape	Code using Lua to detect rendered objects that the snow plow should avoid	Mario	Nirda Cameron Collins
Add and configure angle sensor to plow	Position angle sensor on front of robot body to measure angles between the robot and other rigid objects	Collins	Nirda Cameron Mario
Program plow to move based on angle detection	Code using Lua to enable the robot to adjust plow angle in order to move snowballs in a certain direction	Collins	Nirda Cameron Mario
Add and configure force sensor	Position and configure the force sensor to the plow to receive data on the force exerted onto the robot plow	Cameron	Nirda Collins Mario
Program plow to adjust torque based on the force of snowballs	Code using Lua to adjust torque on the robot plow based on information from the force sensor	Cameron	Nirda Collins Mario
Add and configure accelerometer to robot body	Position accelerometer on robot body to be able to read the speed of the robot	Nirda	Cameron Collins Mario
Program robot to stay within 2 m/s speed range	Code using Lua to maintain a 2 m/s speed, with acceleration or deceleration	Nirda	Cameron Collins Mario
Test robot starting position and angle	Test for the robot position, and its initial angle are optimal	Nirda	Cameron
Test robot's ability to follow a path	Test the robot's ability to follow a path	Cameron	Collins

Test robot’s ability to avoid objects	Test the robot's ability to maneuver around static objects that are not snowballs	Mario	Cameron
Test robot's ability to avoid dynamic objects	Test the robot's ability to maneuver around dynamic objects that are not snowballs	Collins	Nirda
Test robot’s ability to move snowballs	Test the robot’s ability to move the snowballs away and clear the area	Mario	Cameron
Test robot on training map 1	Test the robot on a pre-made path, randomly generated snowball objects and obstacles	Mario	Collins
Test robot on training map 2	Test the robot on a pre-made path, randomly generated snowball objects and obstacles	Cameron	Nirda
Test robot on training map 3	Test the robot on a pre-made path, randomly generated snowball objects and obstacles	Nirda	Collins
Progress report	Write the progress report on how the project is proceeding	All	All
Final report	Write the final report about design decisions, testing methods, and results of the project	All	All
Final presentation script	Write a script to be followed in our presentation as we explain design, implementation, and results of the project	All	All
Final presentation slides	Create a slideshow for the audience to follow along with the presentation, displaying information	All	All

6.0 Overall Architecture

The autonomous snowplow robot is composed of three main parts. The first is a pre-made CoppeliaSim mobile robot for the body, specifically the “Pioneer p3dx”. This is used for moving the robot across the scenes. This robot body has the ability to move in a straight line.

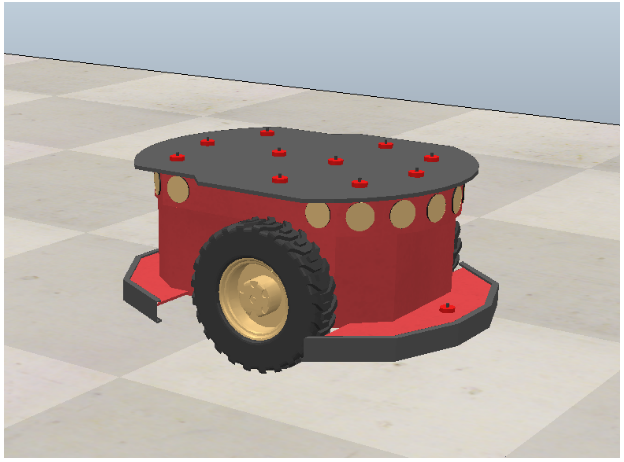


Figure 1. Pre-made CopliaSim robot body used for snow plow robot

The second component of the snowplow is the snowplow face. This is made using three primitive cuboid shapes arranged in a C pattern that resembles a box plow design (as can be seen in figure 2). This is the part that will handle the snow and clear them from the practice scenes by collecting snowballs within the concave opening.

Figure 2. Box plow design inspiration

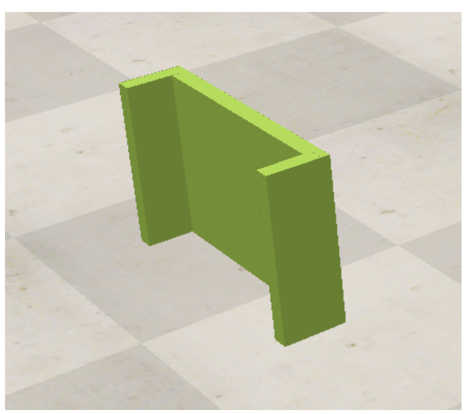


Figure 3. Snowplow face design created in CoppeliaSim

The C shape design of the snowplow will allow the robot to collect and push the snowballs to the desired locations. The snowplow face is attached to the robot using a prismatic joint, these have one DoF (degree of freedom) and are used for transnational movements between objects. For this robot design, the prismatic join is being used in passive mode. While in passive mode the joint acts as a link and is not directly controlled. This works for our purposes as the robot body needs a rigid structure to connect to the face to be able to push it as it moves through the scenes.

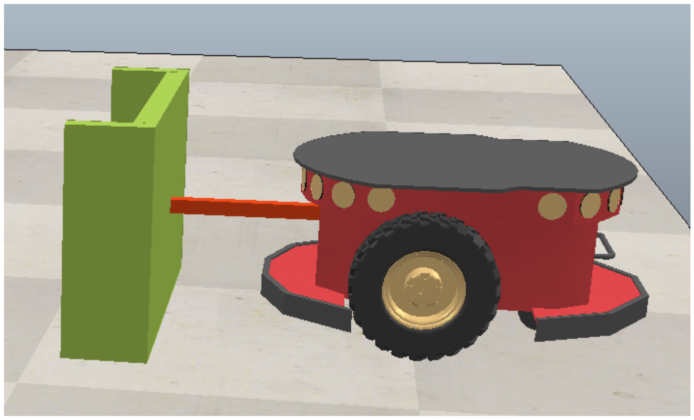


Figure 4. Snowplow attached to robot body using prismatic joint in CoppeliaSim

The final part of the autonomous snowplow architecture is the sensors. The robot will be fitted with sensors to aid its movement within the scenes and to collect/remove snowballs. Proximity sensors will be used on the front of the snowplow and the sides of the robot body to help the robot detect and avoid obstacles. The proximity sensor's data will be used in conjunction with vision sensors as they are better at detecting color, light and structures. The vision sensor will also be used for snowball detection.

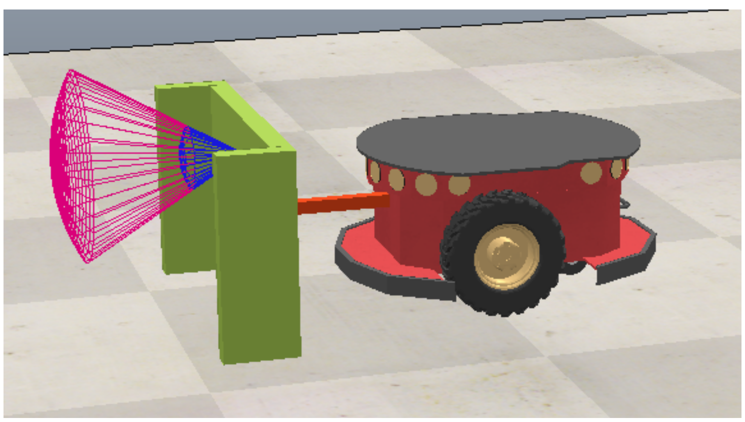


Figure 5. Snowplow with proximity sensor to detect obstacles

The other sensors to be added include the accelerometer which will be used to monitor and control the speed of the robot to meet the specifications. All the programming of the sensors is done using the LUA programming language within CoppeilaSim.

6.1 Event-Triggered Design

The autonomous snowplow robot will use an event-triggered design to communicate with its environment. The proximity and vision sensors will send interrupts to the robot when it comes close to objects or detects snowballs so it can be maneuvered in the environment.

7.0 State Chart of the Overall System

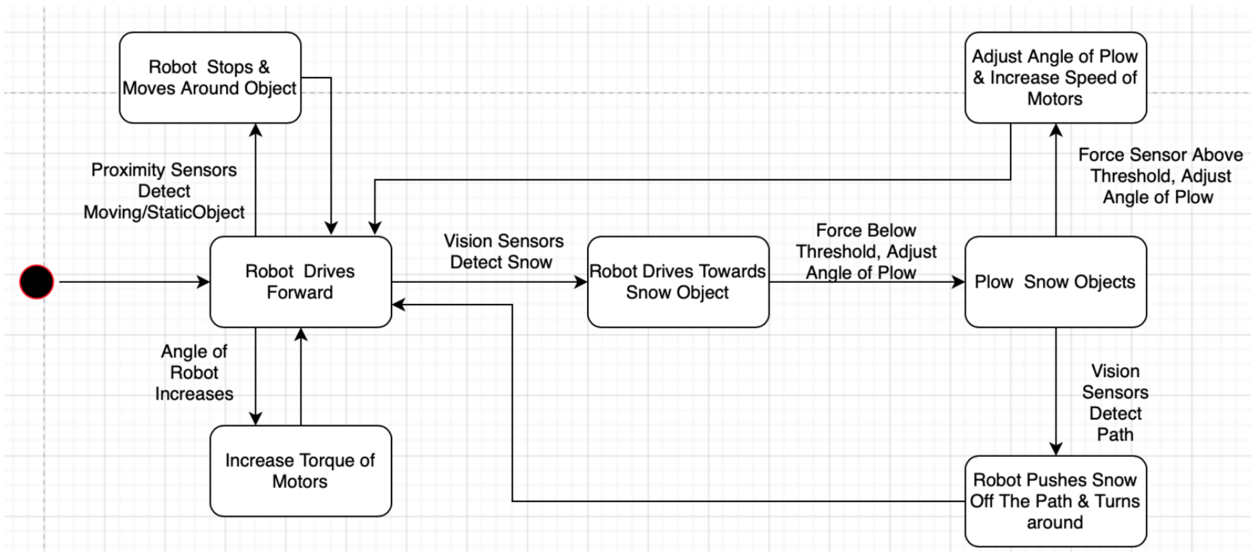


Figure 4. Illustrates the state chart of the system

The state diagram in figure 4 above illustrates the states that the snow plow robot goes through as its sensors react to the environment of the test maps. As each sensor takes a new reading, an action is performed and the robot is in the same state performing each action, up until another sensor reading causes the robot to be in another state. When the plow first starts in the simulation it is in a move forward state. In the case the plow detects an obstacle it moves to the avoidance state where it moves around the object, then continues forward. When the robot detects the snowballs it proceeds towards them to push them off the map. The robot knows when to stop the plow when it detects the black path and that triggers it to move to the next state.

8.0 Sequence Diagram

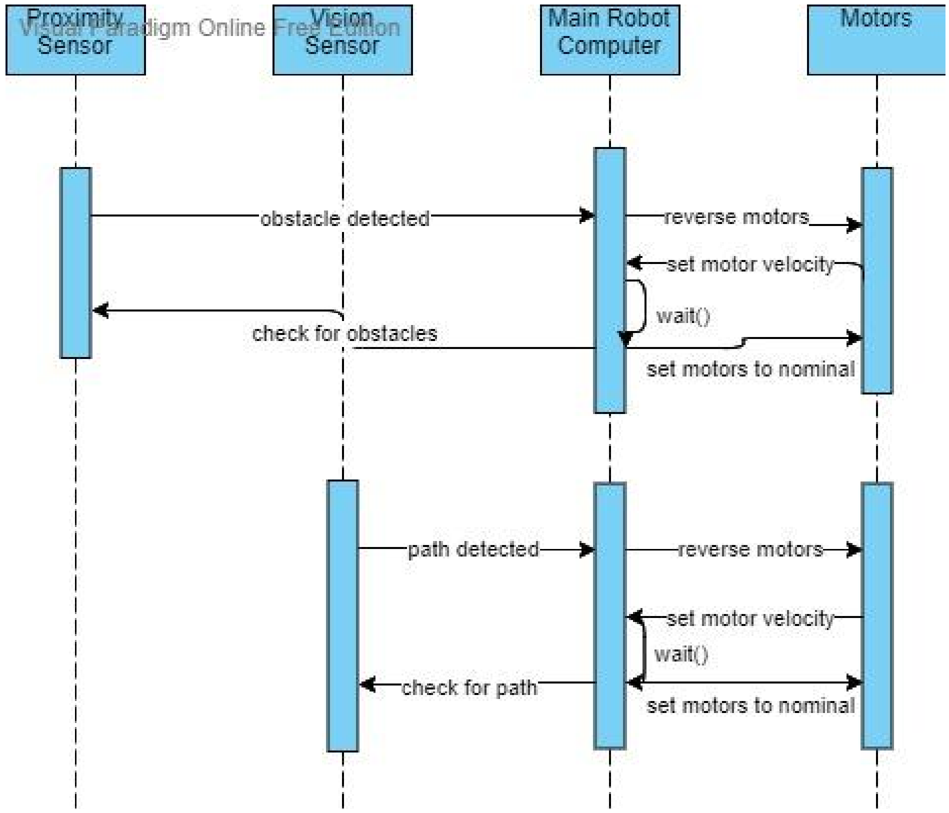


Figure 5. Illustrates, the sequence diagram of the system

This sequence diagram shows the full functionality of the program in the robot. In the top block, when an obstacle is detected, this proximity sensor information is sent to the main robot computer. When the computer algorithm receives this information, it will adjust the velocity and force of the motors accordingly. Once the object is no longer detected in the proximity sensor, the main robot computer will set the motors of the robot to nominal function. In the bottom block, when the path is detected the vision sensor information is sent to the main robot computer. The computer will receive the information from the vision sensors and adjust the velocity and force of the motors accordingly. Once the path is no longer detected from the vision sensors, the main robot computer will set the motors to nominal function.

9.0 Control Chart

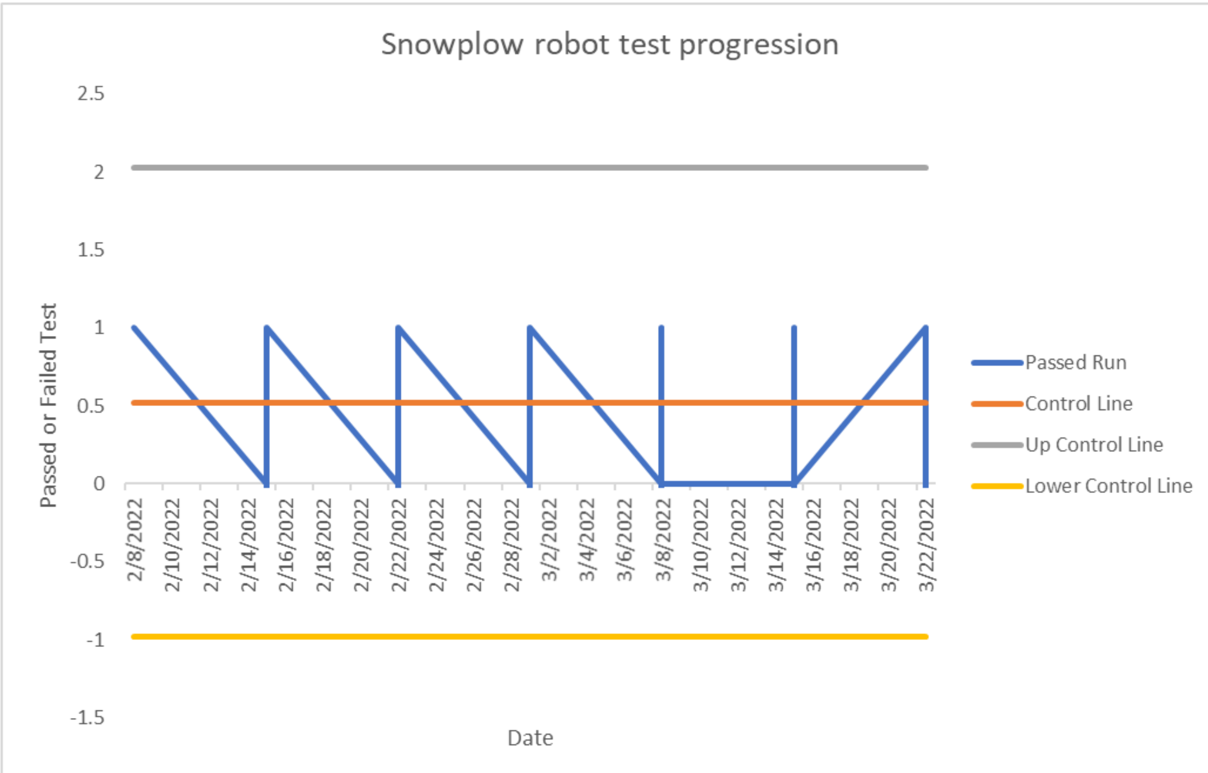


Figure 6. Control Chart

This control chart shows the progression of our project with respect to tests run for each requirement over the duration of the project time. We were able to make good progress with initially passing tests and some tests that passed after several attempts and adjustments. We were not able to implement the inclinometer and accelerometer into our project, hence the large dip from 3/8/2022 to 3/14/2022. With the control line, we averaged out to complete most of our requirements by the end of the project duration.

9.0 Project Budget

9.1 Planned Value

Cost	Value
Work per hour	25$
People in group	4
Hours of work a week	3.5
Current weeks of the project completed	12

Total Planned Value	$4200.00 from 25*4*12*3.5

9.2 Budget at Completion

Cost	**Value**
Work per hour	25$
People in group	4
Hours of work a week	3.5
Weeks left of the project	0
Cost of work remaining	0
3 Vision Sensors	$192.00 USD 3*64.00
1 angle sensor	$3.07 USD
1 force sensor	$151.64 USD
1 proximity sensor	$157.00 USD
1 ultrasonic proximity sensor	$ 141.00 USD
Total cost of sensors in USD	$644.71 USD =192.00+151.64+157.00+141.00+3.07
Total cost of sensors in CAD	$821.11 CAD based off of 1 USD = 1.2736 CAD
Budget at Completion of the Project	$5021.11 from CAD=$821.91+$4200.00

10.0 Results of Training Maps

Training Map Number	Result
Training_Map_1	Robot has 2 snowballs caught in between the plow and the main robot and due to implementation being incomplete the robot does not turn away from the wall in front of it. However, it does not have a collision with the obstacle as the snowballs instead are pushed against the wall.
Map2	Robot picks up 5 snowballs and collides with the wall, however the collision isn’t continuous as

	eventually the snowballs block the plow.
Map3	The robot collides with the moving obstacle twice as it doesn't turn away from initially sensing the human and then the human comes from the backside to hit the robot. The second time the robot is turned around and the proximity sensors code cannot individually control the motor speeds. The robot cannot avoid the obstacle and keeps moving and gets kicked by the moving obstacle. The plow is able to lower the snowball count to 486 and avoids going out of bounds before once again nearly colliding with the wall, albeit in such a way that the robot got stuck between almost colliding with the wall and the outer perimeter with a final snowball count of 481.
Map3	The robot moves at a slight angle visually colliding with the top left obstacle due to the proximity sensor's implementation issues though looking closely the snowballs once again prevent a true collision from occurring. Before the near collision occurred 9 snowballs were picked up by the plow.

11.0 Conclusion

By the end of Winter 2022 term project cycle The Amethyst team had made substantial progress towards the final goal. At this point in time, the robot body design and configuration is complete. The robot body was added and configured to the scene and the snowplow was designed and added to the scene. Then the robot body and plow were connected through joints and configured so that the robot remains intact and moves with the plow when the simulation starts. The robot was programmed according to the specification to start at the required position and orientation relative to the environment when run in testing maps. Sensor functionality was also added with proximity, ultrasonic and vision sensors for path detection to keep the robot within the borders of the testing maps. The sensors were also added for viewing obstacles and snowballs. For maintaining the robot speed to meet the requirement specification, an accelerometer was added for monitoring and maintaining the robot body.

Future recommendations and improvements for the robot would include programming an algorithm for traversing the testing maps similar to the lawn mower model of traversing an area of grass. While the accelerometer was included, it could be further developed, and the same goes for the proximity sensors and object detection.

Overall, The Amethyst team made good progress over the course of the semester in terms of meeting all deliverables on time and completing the robot to a near-finished state. The robot as it presently stands meets 9 out of 12 testing requirements originally specified in the proposal of the project. Thus the project can be deemed successful overall.

12.0 Contributions

The table below lists the team members and their respective contributions to the final report.

Section of Report	Individual Contributor
Addressing feedback from progress report	All
1.0 Introduction	Nirda
2.0 Project Charter	Nirda
3.1 Requirements	Nirda
3.2 Activities	All
3.3 Testing	Nirda, Mario
4.1 Schedule Network Diagram	Nirda
4.2 Gantt Chart	Nirda
5.0 Human Resource Matrix	All
6.0 Overall Architecture	Nirda
6.1 Event-Triggered Design	Nirda
7.0 State Chart Diagram	Collins
8.0 Sequence Diagram	Cameron
9.1 Control chart	Cameron
9.1 Planned Value	Mario
9.2 Budget At Completion	Mario
10.0 Results of Training Maps	Mario
11.0 Conclusion	Nirda

The table below outlines the different team members and their contributions to the various project components.

Project Component	Individual Contributor
Creation of plow design, programming starting position and orientation, static obstacle detection, presentation slides and script, and report contributions throughout the semester.	Nirda
Implementing plow, programming path detection, testing path detection, sequence diagram, control chart, and report contributions throughout the semester.	Cameron
Proximity sensor programming, design recommendations for plow and robot design, testing matrix and budget creation.	Mario
Initial design for robot and proximity sensor. Programming the dynamic object detection and testing proximity sensor functionality. Responsibility assignment matrix step descriptions.	Collins

Appendix A

Sensors equivalent to ones used on the robot in CoppeliaSim software.

“Accelerometer Sensor - MXC6655XA,” *DigiKey*. [Online]. Available: https://www.digikey.ca/en/products/detail/memsic-inc/MXC6655XA/12171923. [Accessed: 04-Feb-2022].

“Camera - TR-Evo-15m-I2C,” *DigiKey*. [Online]. Available: https://www.digikey.ca/en/products/detail/terabee-sas/TR-EVO-15M-I2C/13913787. [Accessed: 04-Feb-2022].

“Force Sensor - FSAGPDXX1.5LC5B5,” *DigiKey*. [Online]. Available: https://www.digikey.ca/en/products/detail/honeywell-sensing-and-productivity-solutions/FSAGPDXX1-5LC5B5/10129190. [Accessed: 04-Feb-2022].

“Angle Sensor - AS5600-Asom,” *DigiKey*. [Online]. Available: https://www.digikey.ca/en/products/detail/ams/AS5600-ASOM/4914332. [Accessed: 04-Feb-2022].

“Proximity Sensor - No5-Q08-AN7,” *DigiKey*. [Online]. Available: https://www.digikey.ca/en/products/detail/banner-engineering-corporation/NO5-Q08-AN7/10652681. [Accessed: 04-Feb-2022].

“Ultrasonic type Proximity Sensor - QS18UPAEQ8,” *DigiKey*. [Online]. Available: https://www.digikey.ca/en/products/detail/banner-engineering-corporation/qs18upaeq8/12717233. [Accessed: 04-Feb-2022].

Appendix B

Link to Github page.

https://github.com/MarioShebib/SYSC4805Robot