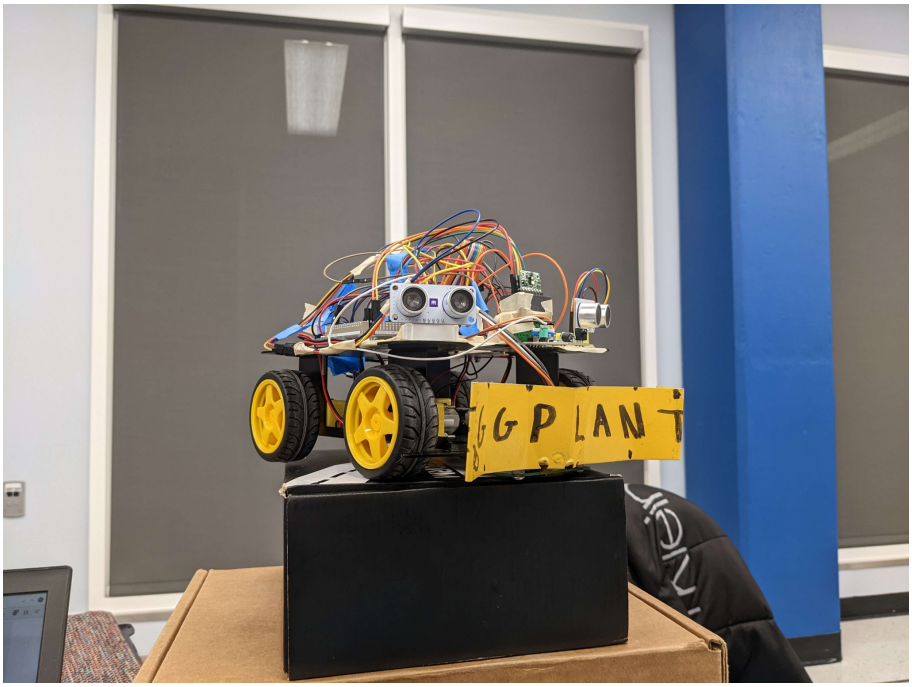
SYSC 4805
Computer Systems Design Lab

Autonomous Snowplow
Final Report



Group L1-3 Members:
Raiyan Hasan #101125753
Kenny Deng #101122713
Paul Okenne #101115693

Table of Contents

1 Project Charter

1.1 Eggplant

The team name that our group has chosen is Eggplant. We decided upon this name because eggplants are our favourite vegetable and a great source of fibre and antioxidants like Vitamin A and C.

1.2 Overall Objective

The overall objective is to design and create an autonomous robot that will clear and push the highest amount of snow out away from a defined area without hitting any obstacles or leaving the designated area simulating a real-life application. Coming from Ottawa, the winters can be extremely harsh, and getting snow cleared out at all times is a big task for anyone living in the city. Whether it is your driveway or the roads, snow must get cleared out due to the increased likelihood to cause accidents and the hazards that come with it. For this reason, Our project is extremely relevant as it solves a significant problem. This project has a defined end goal but how we reach that end goal is up to us. This project will teach us to work with various hardware components such as sensors and microcontrollers, providing valuable experience and challenging our skillset. The sensors will be there to detect obstacles which will allow the robot to navigate without hitting any obstacles while plowing snow away. Furthermore, working in a team will enable us to bring different ideas to the project, and enhance our ability to work collaboratively.

1.3 Overall Deliverables

Over the course of the project, teams are expected to provide deliverables. The upcoming sections break down our milestones and final deliverables.

1.3.1 Milestone Deliverables

The milestone deliverables are tangible non-graded deliverables that indicate our progress in the project. Table 1 details the team's project milestone deliverables.

Table 1. Project Milestone Deliverables Date Summary

Deliverable ID	**Deliverable**	**Due Date**
1	Completed robot chassis	Friday, October 21 @ 11:59 pm
2	Sensor selections finalized	Friday, October 21 @ 11:59 pm
3	Completed assembly of robot chassis with sensors and Arduino board	Friday, October 21 @ 11:59 pm
4	Computer-Aided Design (CAD) of plow	Friday, November 2 @ 11:59 pm

5	Mount constructed plow to robot chassis.	Friday, November 4 @ 11:59 pm
6	Completed obstacle and boundary detection algorithm modules	Friday, November 18 @ 11:59 pm
7	Finalized implementation of snowplow behavioural logic	Friday, November 18 @ 11:59 pm

1.3.2 Final Deliverables

Table 2 displays our final deliverables along with their respective due date.

Table 2. Project Final Deliverables Date Summary

Deliverable ID	Deliverable	Due Date
1	Project Proposal	Friday, October 14 @ 11:59 pm
2	Progress Report	Friday, November 11 @ 11:59 pm
3	Final Report	Friday, December 9 @ 11:59 pm
4	In-Lecture Presentation	Nov. 21: L1 Groups 1 to 5

The project proposal details our design approach for creating the autonomous snowplow. The progress report outlines the current project status including the working code and design architecture. The final report provides a detailed guide to the finalized design. The in-lecture presentation allows teams to share their solutions and justifications. Finally, the core deliverable is the autonomous snowplow which removes snowballs from a given area.

2 Updates to Project Proposal

The feedback provided to us regarding our previously submitted Project Proposal was as follows:

1. You have to mention one activity for each code. It should not be more than 1 activity per code.

   For example, you mentioned in your report that " The robot shall be able to detect a solid black testing arena border strip within 30 cm and not go less than 5 cm to the inside edge other border " which contains more than 1 activity in one code.

2. You have added 20 unique activities, but you added only 16 activities in your SND. However, You have to explain each activities in details with a code in SND.

3. Not all activities are mentioned in the SND. You have around 20 activities, but only 16 showing up in the SND. If you gave unique code ID to each activity, it would be easier to remember and shown them in the SND.
4. You have to mention the equation with the Graph from where you get the values of your table (Cost Baseline).

Based on the aforementioned feedback given to us, we have made the following changes to our Report.

2.1 Change: One Activity Per Code

In our proposal activities were not singular for each activity code. To fix this we have updated the activity list to only include one activity per code/id.

2.2 Update Schedule Network Diagram (SND)

In our proposal we did not include all activities from our activities list into our schedule network diagram (SND). To fix this we have updated the SND to include all listed activities. We have also attached the relevant activity code to each node in the SND.

2.3 Revise Cost Baseline Table

Originally in our proposal, we did not mention the equation with the Graph from where you get the values of our table (Cost Baseline). In this report we have revised the Cost Baseline Table and have included the aforementioned equation and values.

3 Updates to Project Progress Report

There was no additional feedback provided to us regarding our previously submitted Project Progress. In terms of the changes made between the Progress Report and the Final Report the major changes are as follows:

- Contributions of each member identified
- Updated and added additional linked contributions of each member to GitHub commit
- Control charts added
- State machine refactoring which includes new states and behaviours
- Calculated overall cost of project
- Results of testing system testing
- Results of customer testing
- Updated tables and figures with new and removed data/activities
- Added a contributions section to detail the work done by members for deliverables

4 Scope

4.1 Requirement List

Table 1 shows the project requirement list requirement and its associated code id.

Table 1: List of Requirements

Requirement Id	Requirement Description
1	The snow plow shall have an operation time that is at least 5 minutes.
2	When an object detection signal is received, the snow plow application sends a stop signal to the wheels within 1 second.
3	The snowplow shall push white plastic balls, that are 42.6 mm in diameter, outside the testing area.
4	The plow can only extend in width by a maximum of 40 mm (20 mm in each direction), and extend in length by 20 mm.
5	The maximum allowable robot speed is 30 cm/s.
6	The robot shall be able to detect a solid black testing arena border strip within 1 second.
7	The robot shall not go less than 5 cm to the inside edge of the border.

4.2 Activity List

Figure 1 below illustrates a broad general overview of the proposed design, development, and testing workflow that our group has proposed to follow during the course of this project.

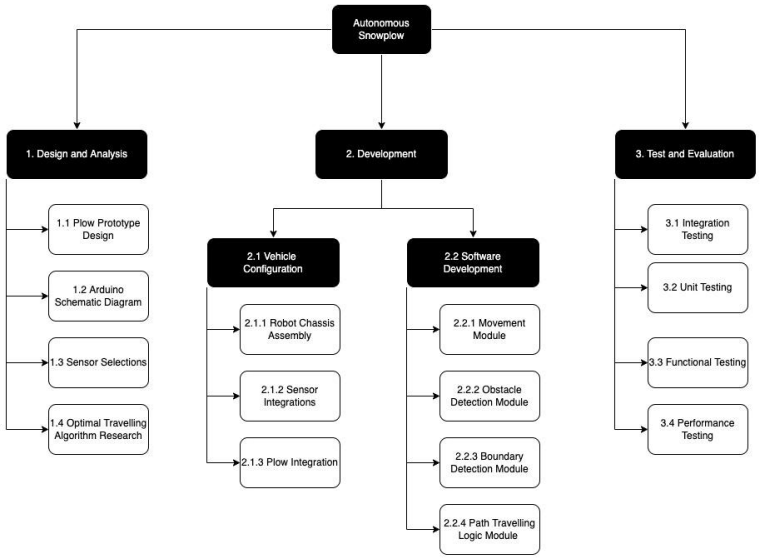


Figure 1: Waterfall-based work breakdown structure

Table 2 below provides a detailed description of each activity.

Table 2: Activity Breakdown

Activity ID	Activity	Description
1	Design and Analysis	The designing process for the system
1.1	Plow Prototype Design	Creating the most optimal design that can be used on the Robot.
1.2	Arduino Schematic Design	Arduino schematic diagram that will help in configuring and tracking all the pins which will help in connecting sensors.
1.3	Sensor Selections	Deciding which are the most effective sensors that can be used on the robot.
1.4	Optimal Travelling Algorithm Research	Researching on which algorithm is the most effective for the Robot to travel.
2.	Development	The hardware and software development of the system
2.1	Vehicle Configuration	The development of the vehicle and overall hardware system
2.1.1	Robot Chassis Assembly	Assembling the Chassis by wiring and attaching controllers, creating a neat design and professional package.

2.1.2	Sensors Integration	Connecting all the sensors required onto the Robot chassis
2.1.3	Plow Integration	Attaching the plow towards the from of the Robot chassis
2.2	Software Development	The development of software that enables the system behavior
2.2.1	Movement Module	
2.2.2	Obstacle Detection Module	Development of the Robot In which It will be detecting obstacles using the sensors attached.
2.2.3	Boundary Detection Module	Development of the Robot In which It will be detecting the boundary area using the sensors attached.
2.2.4	Path Travelling Logic Module	Implementing the logic that is behind the Robot to travel to its designated area.
3.	Testing and Evaluation	The testing and evaluation of the system
3.1	Integration Testing	Integration testing is ensuring the system of different sensors and the microcontroller works together efficiently and effectively.
3.2	Unit Testing	Unit test is the testing of software components such as modules.
3.3	Functional Testing	Functional testing is the testing of hardware. In this activity, the sensors and movement subsystems are verified
3.4	Performance Testing	Performance testing is the testing that focuses on how the system perform in given practical situations

Table 3 below breaks down each activity into further detail each with an assigned member and weekly activity.

Table 3: Weekly Activity Overview

Lab Week #:	**Member Activity**		
	Raiyan	**Kenny**	**Paul**

7	Research and justify sensor selections Complete robotic chassis assembly and connect sensors to robot chassis Sensor Functional Testing	Research and justify sensor selections Schematic arduino diagram	Research optimal travelling algorithm Movement functional testing: Test that when electrical components are configure in a way that enables movement
8	High level algorithm development of boundary detection module with dummy data Unit testing of boundary detection module	Build & mount plow to robot chassis High level algorithm development of obstacle detection module with dummy data Unit testing of obstacle detection module	Implementation of vehicle movement module Unit testing of movement module
9	Implement boundary detection module with real sensor data Integration testing of obstacle detection module	Implement obstacle detection module with real sensor data Integration testing of obstacle detection module	Implement the main code, the path-travelling algorithm Unit testing of path-travelling algorithm
10	Optimize boundary algorithm	Optimize obstacle detection algorithm	Optimize motor speed for maximum snow plowed
11	Performance Testing	Clean up smelly code	Ensure stability of system

4.3 Testing Plan

Testing is an important project phase that highlights defects that when resolved, provide confidence about the product quality. The upcoming sections will disclose our testing strategies.

4.3.1 Hardware Functional Testing

Hardware functional testing is the first testing phase that occurs in the project. The purpose of the hardware tests is to validate the functionality of hardware subsystems.

Our hardware subsystems can be separated into a sensor subsystem, a power subsystem, and a movement subsystem.

The sensor subsystem consists of a line follower and a distance sensor. Each sensor test verifies that appropriate voltage values are observed for different situations.

The movement subsystem consists of the motor driver and the wheels. The movement subsystem tests verify that the vehicle is electronically configured to enable movement.

4.3.2 Unit Testing

Unit testing is the process of testing individual components of software applications. The components in our software are modules with varying functionalities.

The plan is that after the modules are developed, unit tests are written to verify the module's functionality. Table 4 shows our high-level test plan for each of our modules.

Table 4: High-Level Module Test Plan

Test ID	**Software Module**	**Test Description**
1	Movement module	Test that the direction and enable configurations for given movement instructions are valid
2	Obstacle Detection module	Test that given different voltage values, the module identifies obstacles
3	Line Detection module	Test that given different voltage values, the module identifies boundary lines
4	Path Traveling module	Test that given line and obstacle detection inputs, the module provides appropriate instructions.

The proposal is limited to 15 pages and as result, detailed unit test case plans are constrained. Table 4 & Table 5 serves as an example of how the team will approach breaking down unit test cases. Table 5 discloses the unit test cases for the movement module.

Table 5: Detailed Movement Module Unit Testcases

Test Id	**Test Description**	**Test Setup**	**Success Criteria**	**Failure Criteria**
0	Test the vehicle behaviour when given a move *forward* instruction	Create and send the specified instruction	The application alters direction and enables configurations to move *forward*	The application does NOT provide correct configurations to move forward
1	Test the vehicle behaviour when given a move *backwards* instruction		The application alters direction and enable configurations to moves *backwards*	The application does NOT alter the directions or does NOT move backwards correctly.
2	Test the vehicle behaviour when given a *turn right* instruction		The application alters direction and enable configurations to *turn right*	The application does NOT provide correct configurations to turn right
3	Test the vehicle behaviour when given a *turn left*		The application alters direction and enable	The application does NOT provide correct

	instruction		configurations to *turn left*	configurations to turn left

The movement module unit test cases cover the application's movement requirement. The obstacle-detection module unit test cases will cover the application requirement to detect obstacles. The boundary-detection module unit test cases will cover the application requirement to detect a boundary. Finally, the path-travelling module will cover the application requirement to navigate in an area.

4.3.3 Integration Testing

Integration testing is another branch of testing that the team will perform. The integration tests focus on interactions between the software and hardware components.
Table 6 discloses the team's high-level integration test plan.

Table 6: Integration Test Plan

Test ID	**Software Component**	**Hardware Subsystem**	**Test Description**	**Success Criteria**	**Failure Criteria**
1	Movement module	Mobility hardware subsystem	Test that direction and enable configurations from the movement module translates to expected physical motions	The observed physical movement of the vehicle is correct.	The observed physical movement of the vehicle is NOT correct.
2	Obstacle-detecti o module	Sensors subsystem	Test that the module sets global obstacle-detection flags according to voltages from sensors	The observed obstacles are detected correctly	The observed obstacles are not detected correctly
3	Line-detection module	Sensors subsystem	Test that the module sets global line-detection flags according to voltages from sensors	The global line-detection flags are set according to voltages from sensors	The line detection flags are not set accordingly
4	Travelling path module	Hardware Unit as a whole	Test that for different cases, the travelling path module uses supporting modules to determine and take appropriate physical actions	The travelling path module uses supporting modules to determine and take appropriate physical actions	The taken actions by the module are unwanted behaviours

4.3.4 Performance Testing

The purpose of performance testing is to evaluate how well the system performs in real-life scenarios. With the feedback, modifications can be made to improve and enhance the system's functionalities.

The team plans on allocating time for testing the autonomous snowplow in the testing area. The autonomous snowplow will be placed in various situations within the area and its performance will highlight needed modifications. Multiple tests will be running with the same and varying arena parameters (obstacle count, snowball count, arena shape & size) and modifications to the software (movement, obstacle, line detection, travelling module) can be made to experimentally see which software implementation results in the greatest number of snowballs cleared with the lowest number of obstacle hits.

Table 7: Detailed Performance Testcases

Test ID	System	Test Environment	Test Description	Success Criteria	Failure Criteria
1	Software Implementation	Defined arena with constant obstacle number and snowballs	Test and measure the time that robot takes to clear snow from the arena. Measure obstacle hit number and number of snowballs cleared from arena.	Greater number of snowballs cleared and less obstacles hit. Robot stays in the defined arena.	Lesser number of snowballs cleared and more obstacles hit. Robot leaves the defined arena.

4.3.4 Control Charts

Figure A shows the control chart that reveals the deviations in the vehicle speed. The control chart relates to requirement 5 which states that the vehicle's maximum allowable robot speed is 30 cm/s.

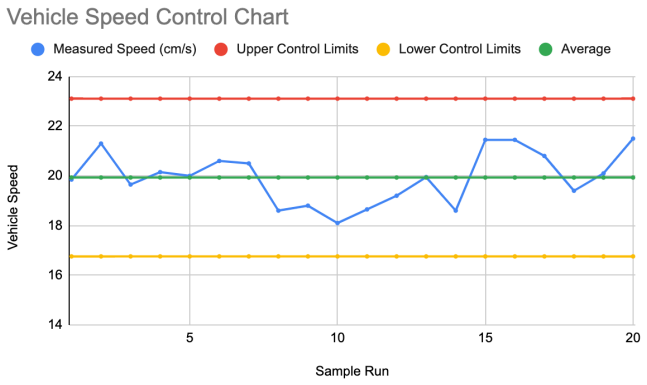


Figure A: Vehicle speed control chart, pertaining to requirement 5

20 sample runs were performed. Each run consist of measuring the time that snowplow is required to travel a meter. The speed is then calculated in cm/s. The target speed of our system was 20 cm/s. Figure A shows that on average the snowplow is roughly 20 cm/s. The observed deviations in Figure A also show that there are no outliers as deviations are within control limits. Hence, the conclusion is that the system fulfills the requirement to have a speed of less than 30 cm/s.

Requirement 2 states that when an object detection signal is received, the snow plow application sends a stop signal to the wheels within 1 second. In order words, the snow plow undergoes behaviour changes within a second of detecting an obstacle. Figure B shows the obstacle detection time control chart. 20 sample runs were performed.

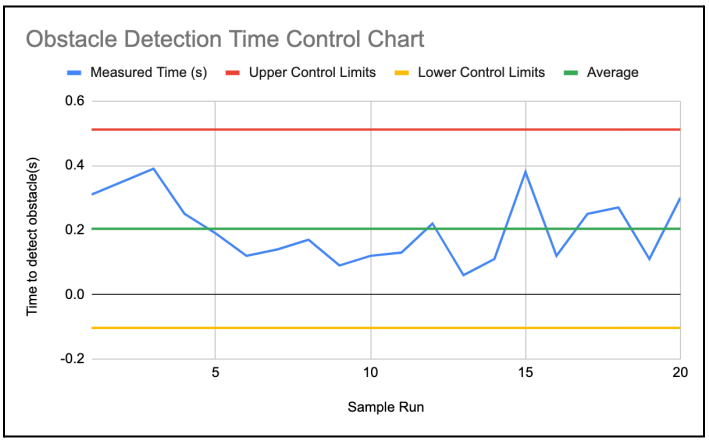


Figure B: Obstacle detection time control chart, pertaining to requirement 2

In Figure B, each sample run involved placing an object in front of the vehicle and measuring the time required for the motors to stop. The average obstacle detection time was 0.201s; the upper control limits were 0.511s; finally, the lower control limits were -0.104s. As a result, the conclusion is that requirement 2 is fulfilled as the obstacle detection time is less than a second.

Requirement 6 states that the robot shall be able to detect a solid black testing arena border strip within 1 second. Figure C shows the boundary detection time control chart. Figure C comprises 20 sample runs.

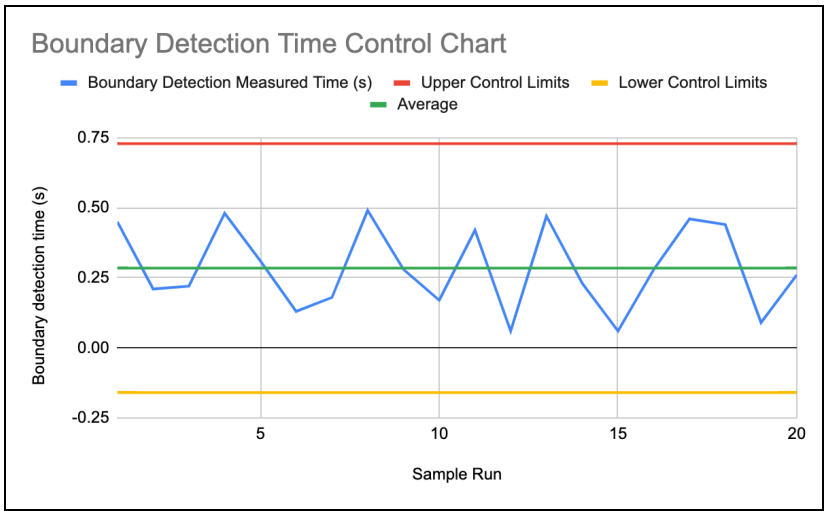


Figure C: Boundary detection time control chart, pertaining to requirement 6.

Each sample run in Figure C involved placing the snowplow on the black area and measuring the time required for the motors to stop. The average boundary detection time was 0.284s; the upper control limit was 0.728s; finally, the lower control limit was -0.159s. Thus, the conclusion is that requirement 6 is fulfilled as the time boundary detection time is less than a second.

Requirement 1 details that the system shall have an operation time that is at least five minutes. Figure D shows the operation time control chart, consisting of 10 sample runs.

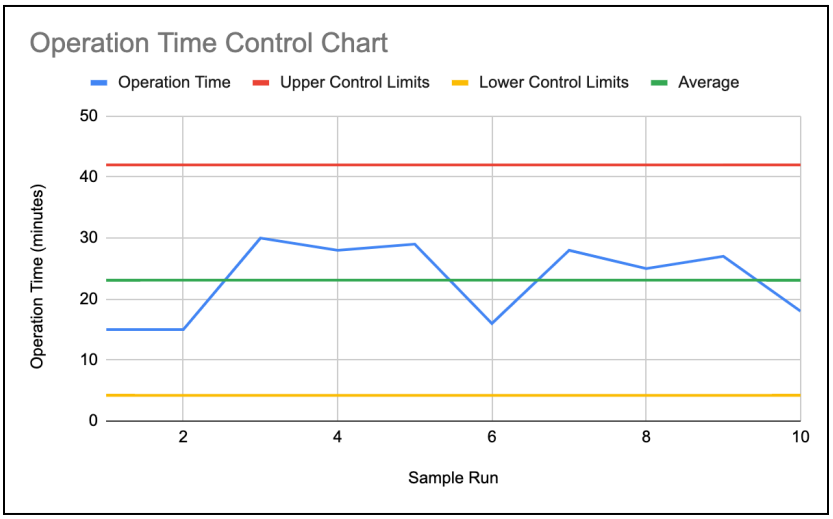


Figure D: Operation time control chart, pertaining to requirement 1

In each sample run, the team allowed the system to run until it stops or leaves the arena. It is important to note that the team adds a maximum cutoff time of 30 minutes: if a sample run operation time reaches 30 minutes, the sample run is stopped. Figure D shows that the average operation time was 23.1 minutes; the upper control limit was 42

minutes and the minimum control limit was 4.2 minutes. Hence, the conclusion is that requirement 1 is satisfied.

5 System Design

5.1 Overall System Architecture

The overall general system architecture of our robot design is very simple and straightforward, we constantly monitor the ambient environment for the boundary and obstacles using the line follower and distance sensor respectively, which feed data into the Arduino. The Arduino will then apply the logic that will be later explained in the following Section 4.2 (Overall System Statechart) and using the results obtained from the logic, update and control each motor state/behaviour/movement via the motor controller to produce a favourable outcome. An overall system architecture diagram is shown below in Figure 2.

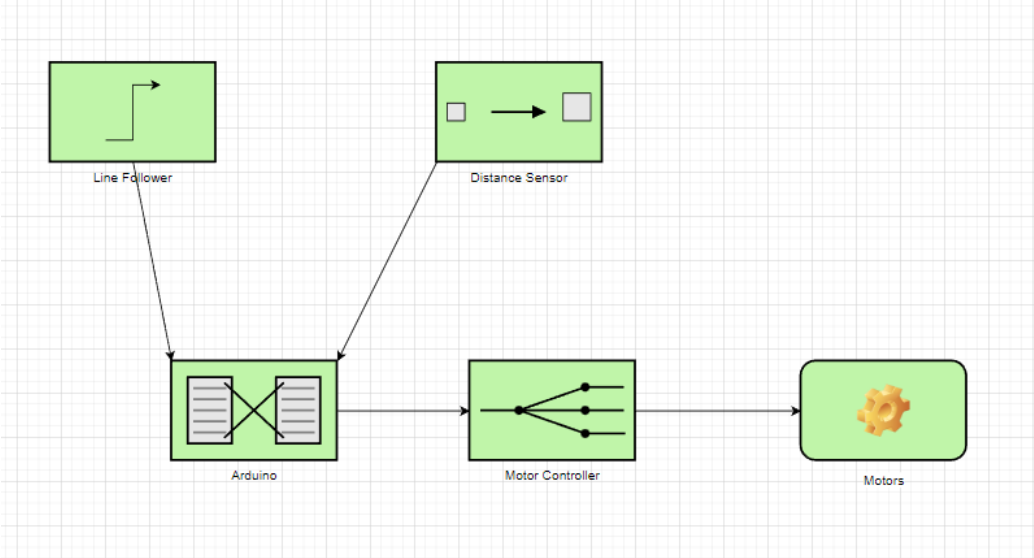


Figure 2. System Architecture Diagram

The line follower sensor used is an IR-based sensor that detects changes in high-contrast lines. Our Robot will have 2 line follower sensors to detect the boundary lines on the front left and right side of the Robot. The distance sensor used is the VL53L1X time-of-flight sensor. This sensor is laser-based and is very accurate and has a relatively high range (compared to ultrasonic, IR, etc). Our Robot will have 2 distance sensors top mounted facing forwards to detect any obstacles in front of the Robot and to also determine any clear paths for the Robot to travel.

Figure 3 shows the breakdown of the software and hardware subsystems.

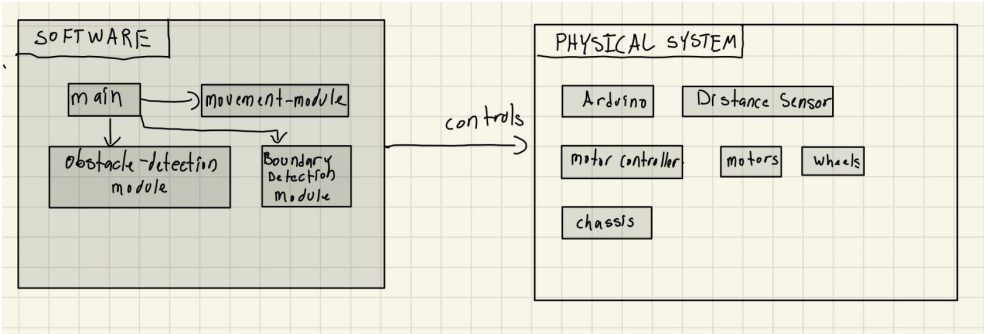


Figure 3: Software and Hardware Component Breakdown

Figure 2 focuses on the interconnection between the physical system. Figure 3 reveals details about the software subsystem. The software subsystem consists of four modules: main, movement, obstacle-detection, and boundary-detection modules.

The movement module enables vehicle movement: front, backward, right, and left. The obstacle-detection module detects when an obstacle is closer to the vehicle. Also, the boundary-detection module detects when the vehicle is at a boundary line. Finally, the main module acts as the controller and uses other modules to control the vehicle's behaviour.

5.2 Overall System Statechart

The purpose of a state chart or state machine is a diagram that details system behaviours under varying conditions. Figure 4a is the state machine diagram for our system.

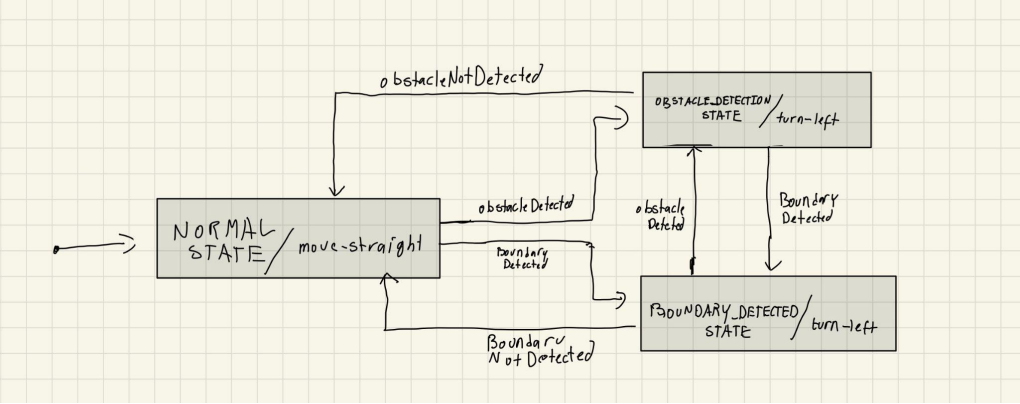


Figure 4a: State Machine Diagram

Our system has 3 states: normal, obstacle-detected, and boundary-detected state. The default state is the normal state. In the normal state, the vehicle moves straight using the movement module.

If an obstacle is detected, we enter the obstacle-detection state. In the obstacle-detection state, the vehicle turns left. If an obstacle is not detected, the state

changes to the normal state. If the boundary is detected, the state changes to the boundary-detected state.

From the normal state, if the boundary is detected the state changes to the boundary-detected state. In the boundary-detected state, the vehicle turns left. If the boundary is no longer detected, the state changes to the normal state. If an obstacle is detected, the state changes to the obstacle-detection state.

5.2.1 Updated System Statechart

Figure 4b shows the updated system statechart. The state machine was updated to improve how the system interacts with obstacles and boundaries. The changes include breaking the obstacle detection state into two states and updating the boundary detection behaviour.

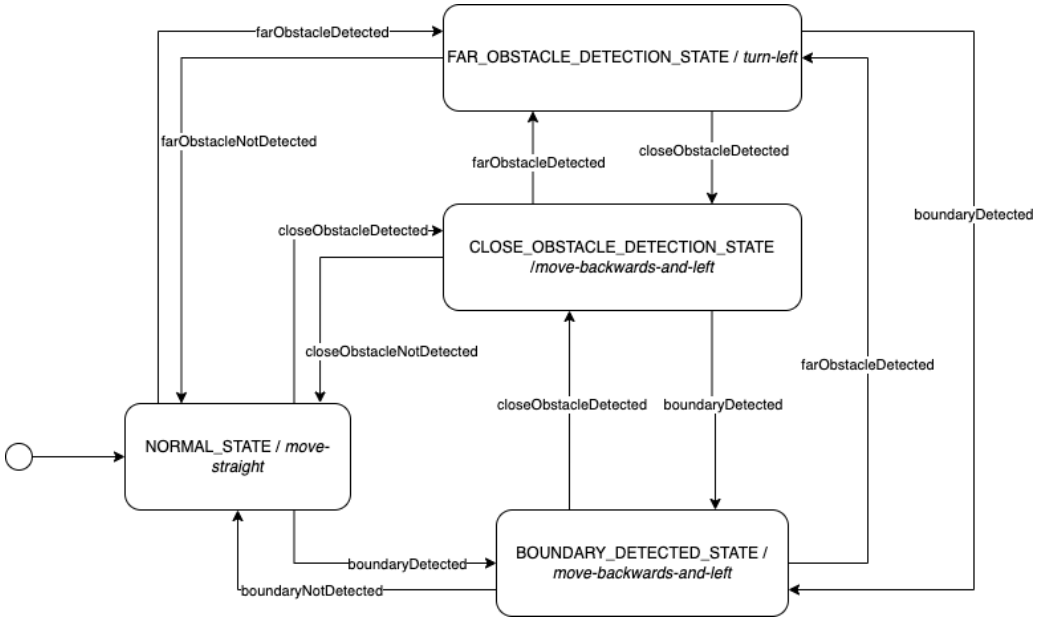


Figure 4b: The Updated System Statechart

The system starts at the normal state, moving forward. When a close obstacle is detected, the system enters the close-obstacle-detection state; when a far obstacle is detected, the system enters the far-obstacle-detection state; finally. when a boundary is detected, the system enters the boundary-detected state.

In the close-obstacle-detection state, the snowplow moves backwards while turning left. If a far obstacle is detected, the system enters the far-obstacle-detection state. If a boundary is detected, the system enters the boundary-detected state. If no close obstacle is detected, the system enters the normal state.

In the far-obstacle-detection state, the snowplow turns left. If a close obstacle is detected, the system enters the close-obstacle-detection state. If a boundary is detected, the system enters the boundary-detected-state. If no far obstacle is detected, the system enters the normal state.

In the boundary-detection state, the snowplow moves backwards while turning left. If a far obstacle is detected, the system enters the far-obstacle-detection state. If a close obstacle is detected, the system enters the close-obstacle-detection state. If no boundary is detected, the system enters the normal state.

5.3 Sequence Diagram

Figure 5 shows the sequence diagram of boundary detection.

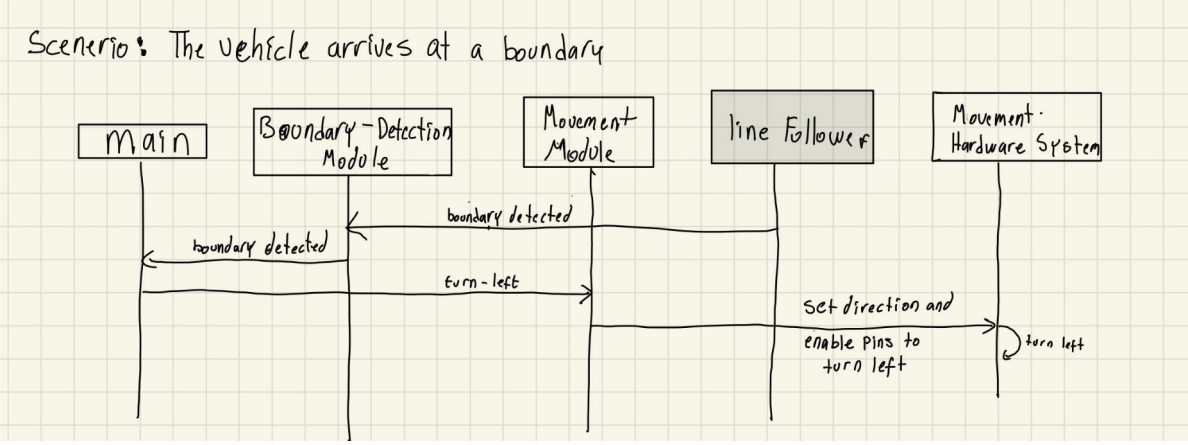


Figure 5: Sequence diagram of boundary detection

The boundary-detection module is based on an interrupt-based approach. The line follower informs the boundary-detection module that a boundary has been detected. The boundary-detection module sets the boundary-detected flag, notifying the main module. The main module sends a turn-left command to the movement module; the movement module sets the direction and enable pins to turn left in the movement hardware system. The movement hardware system comprises the motor controller, the motors, and the wheels. Based on the set pins, the vehicle moves differently.

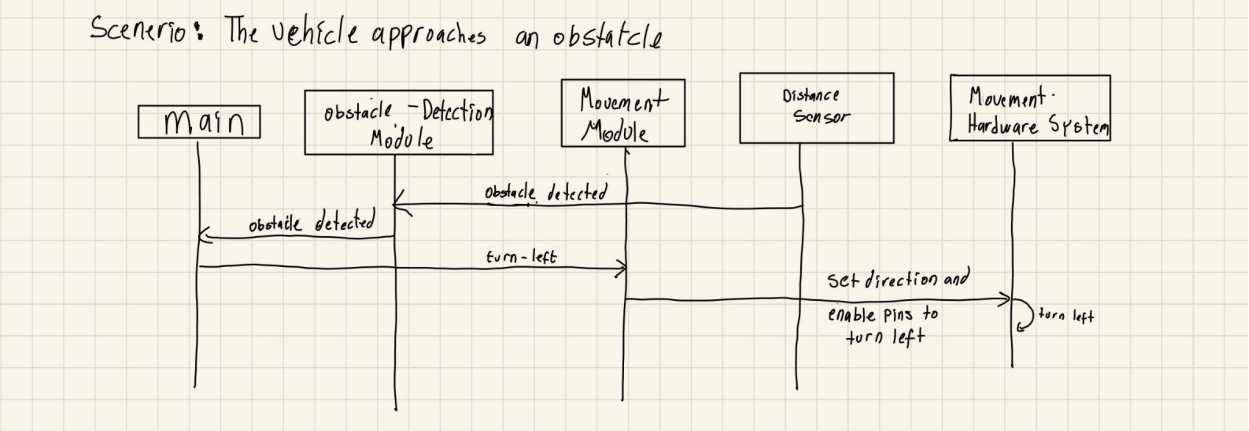


Figure 6: Sequence diagram of obstacle detection

The obstacle-detection module is also based on an interrupt-based approach. The distance sensor informs the obstacle-detection module that an obstacle has been approached. The obstacle-detection module sets the obstacle-detected flag, notifying the main module. The main module sends a turn-left command to the movement module; the movement module sets the direction and enable pins to turn left in the movement hardware system. The movement hardware system turns lefts.

5.4 Watchdog Timer

Watchdog timers are used in order to reset the main program, without having to reset manually every time. We have implemented the Watchdog timer in the main code file. We added the enable function in the setup(). At line 15, we enable the watchdog and set the time-out period to 100 ms. The main loop is where the timer gets reset using the wachDogReset function at line 20.

```
3   void watchdogSetup(){}
4
5   int currentState = NORMAL_STATE;
6   extern bool isObstacleDetected;
7   extern bool isBoundaryDetected;
8
9   void setup() {
10    setupBoundaryModule();
11    setupMovementModule();
12    setUpObstacleModule();
13
14    Serial.begin(9600);
15    watchdogEnable(100);
16  }
17
18  void loop() {
19    // Reset the watchdog timer
20    watchdogReset();
21
```

```
21
22    // Update the state
23    updateState();
24
25    Serial.println(currentState);
26
27    switch (currentState) {
28      case (NORMAL_STATE):
29        moveForward();
30        break;
31
32      case (OBSTACLE_DETECTED_STATE):
33        turnLeft();
34        break;
35
36
37      case (BOUNDARY_DETECTED_STATE):
38        turnLeft();
39        break;
40    }
41  }
```

Figure 7: Watchdog timer implementation

At the beginning of each loop, we reset the watchdog timer. The watchdog timer applies to all non-configuration functions because the watchdog is reset in the loop. If any function malfunctions, the watchdog timeout is reached and the loop function is reset.

6 Schedule

Schedules are a critical component of any project. The schedule will gauge our overall development progress and aid in tracking if we are on time to deliver the project deliverables to the stakeholders. The upcoming subsections provide insight into how the team plans to complete and deliver the project within the given timeframe.

6.1 Schedule Network Diagram

Figure 8 is the team's schedule network diagram, which shows the logical relationships between activities and timing constraints.

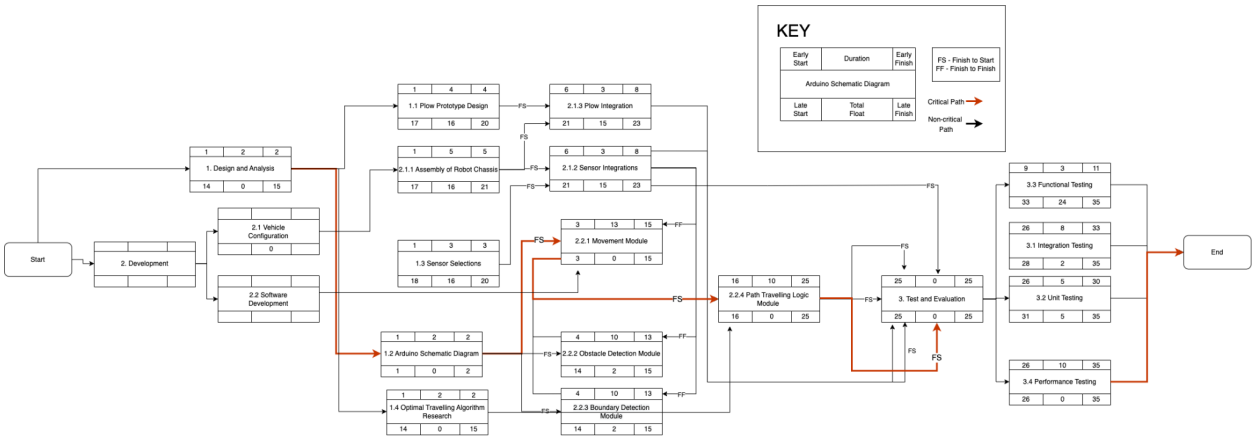


Figure 8: Schedule Network Diagram

The orange line reveals the critical path of activities in the project. The activities in the critical path are the following:

- Arduino schematic diagram
- Movement module
- Path travelling module
- Performance testing

Extra focus will be placed on completing critical activities, ensuring the project is completed and delivered in a timely fashion.

6.2 Gantt Chart

Figure 9 is the team's project Gantt chart that shows how we plan to allocate time for each activity.

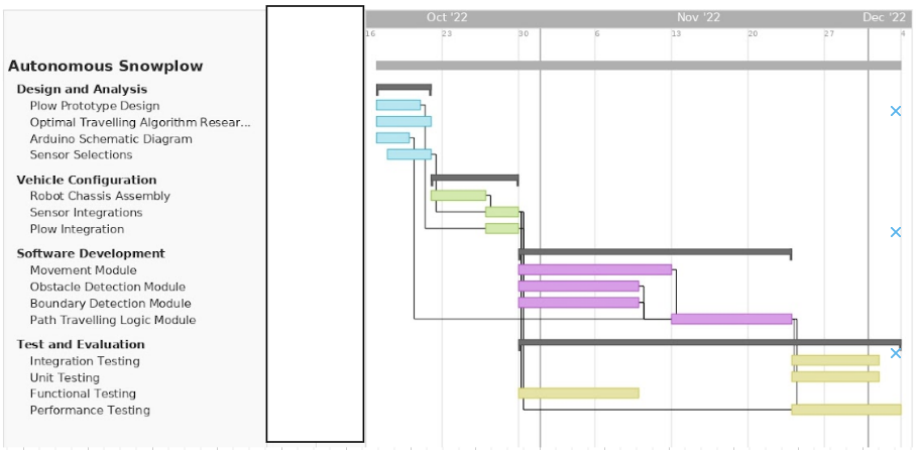


Figure 9: Project Gantt Chart

The team plan begins on October 17, 2022, and is projected to be completed by December 3, 2022.

7 Cost

7.1 Proposed Cost Baseline Figure

Table 8: Developer Cost Baseline Figure

Activity	No of developers	Total Hours per Developer	Completion Time Span (Days)	Total Cost ($)
Plow prototype Design	2	4	3	$400
Plow Integration	2	4	2	$400
Robot chassis	1	5	4	$250
Sensor Integrations	1	6	2	$300
Sensor Selections	2	4	2	$400
Movement Module	2	10	12	$1000
Obstacle/Boundary detection	2	8	11	$800
Researching Algorithms	2	5	2	$500
Path travelling logic	2	7	9	$700
Testing	3	10	10	$1500
Planned Value:				**$6250**
Management Reserve (20% of Total Cost)				**$1250**

Project Budget		**$7500**

7.2 Updated Planned Value Analysis Figure

Below is an updated planned value analysis figure with a calculated value based on completed and uncompleted activities.

PV (Planned Value) = Authorized Budget (All Activities to Date) = Plow prototype Design + Plow Integration + Robot chassis + Sensor Integrations + Sensor Selections + Movement Module + Obstacle/Boundary detection + Researching Algorithms + Path travelling logic + Testing = $6250
EV (Earned Value) = Authorized Budget (Actually Completed Activities) = Plow prototype Design + Robot chassis + Sensor Integrations + Sensor Selections + Movement Module + Obstacle/Boundary detection + Researching Algorithms + Path travelling logic = $4350
AC (Actual Cost) = Actual Budget (Completed Activities) = Plow prototype Design + Robot chassis + Sensor Integrations + Sensor Selections + Movement Module + Obstacle/Boundary detection + Researching Algorithms + Path travelling logic = $100 + $250 + $250 + $150 + $900 + $1000 + $1000 + $300 + $500 = $4450
BAC (Budget At Completion) = Sum(PV) = $6250
EAC (Estimate At Completion) = AC + PV(Remaining Activities) = $4450 + Plow Integration + Testing = $4450 + $400 + $1500 = $6350

Currently we have a budget authorized for $6250 worth of activities but have only completed $4350 worth of activities thus far. We are currently $100 over our $4350 authorized budget. We have an EAC of $100 over our initial proposed PV but that is well within our Project Budget including the management reserve of $7500.

Below is Figure 10 which outlines the value of each activity and its cost with respect to each other.

Cost Baseline = $6250
Project Budget = Cost Baseline + management reserve
= $7500

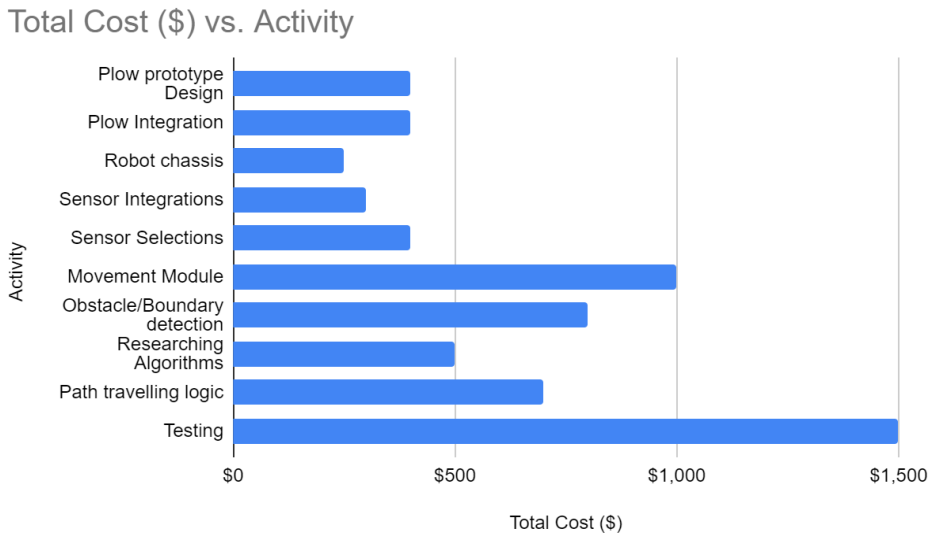


Figure 10. Value Analysis - Cost vs Activity Chart

7.3 Final Value Analysis Figure

Table 8: Developer Final Cost Baseline Figure

Activity	No of developers	Total Hours per Developer	Completion Time Span (Days)	Total Cost ($)
Plow prototype Design	2	4	2	$200
Plow Integration	2	4	2	$400
Robot chassis	1	5	4	$250
Sensor Integrations	1	6	2	$300
Sensor Selections	2	4	2	$400
Movement Module	2	16	10	$1000
Obstacle/Boundary detection	2	18	11	$800
Researching Algorithms	2	6	2	$500
Path travelling logic	2	14	9	$700
Testing	3	10	8	$1200
Planned Value:				**$5750**
Management Reserve (20% of Total Cost)				**$1150**

Project Budget		$6900

PV (Planned Value) = Authorized Budget (All Activities to Date) = Plow prototype Design + Plow Integration + Robot chassis + Sensor Integrations + Sensor Selections + Movement Module + Obstacle/Boundary detection + Researching Algorithms + Path travelling logic + Testing = $5750

EV (Earned Value) = Authorized Budget (Actually Completed Activities) = Plow prototype Design + Robot chassis + Sensor Integrations + Sensor Selections + Movement Module + Obstacle/Boundary detection + Researching Algorithms + Path travelling logic = $4550

AC (Actual Cost) = Actual Budget (Completed Activities) = Plow prototype Design + Robot chassis + Sensor Integrations + Sensor Selections + Movement Module + Obstacle/Boundary detection + Researching Algorithms + Path travelling logic = $200 + $250 + $300 + $400 + $1000 + $800 + $500 + $700 = $4150

BAC (Budget At Completion) = Sum(PV) = $5750

EAC (Estimate At Completion) = AC + PV(Remaining Activities) = $4150 + Plow Integration + Testing = $4150 + $200 + $1200 = $5550

The below Figure 11 outlines the value of each activity and its cost with respect to each other.

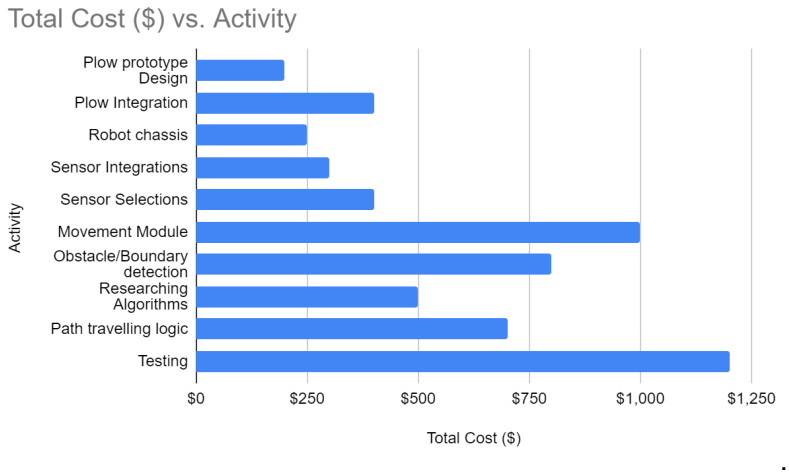


Figure 11. Value Analysis - Cost vs Activity Chart

The Overall budget assigned was $7250 from our progress report and the final figure we got after the completion of the project was $6900 which is within the budget. Based on the breakdown of calculations, you can see the PV went down due to some activities finishing sooner than expected. On the other hand, the value of EV increased slightly due to activity time increasing for testing.

8 Testing Results

8.1 System Testing Results

We created all possible scenarios that the Robot may encounter inside the area. As development progressed and testing was done we noticed that we were missing test cases and they were subsequently added.

Due to the high number of test cases, we have provided a few sample test cases below and have attached our entire testing test cases document in Appendix 13.1. All testing was completed in a self-built testing arena. This testing arena can be seen below in Figure 12.

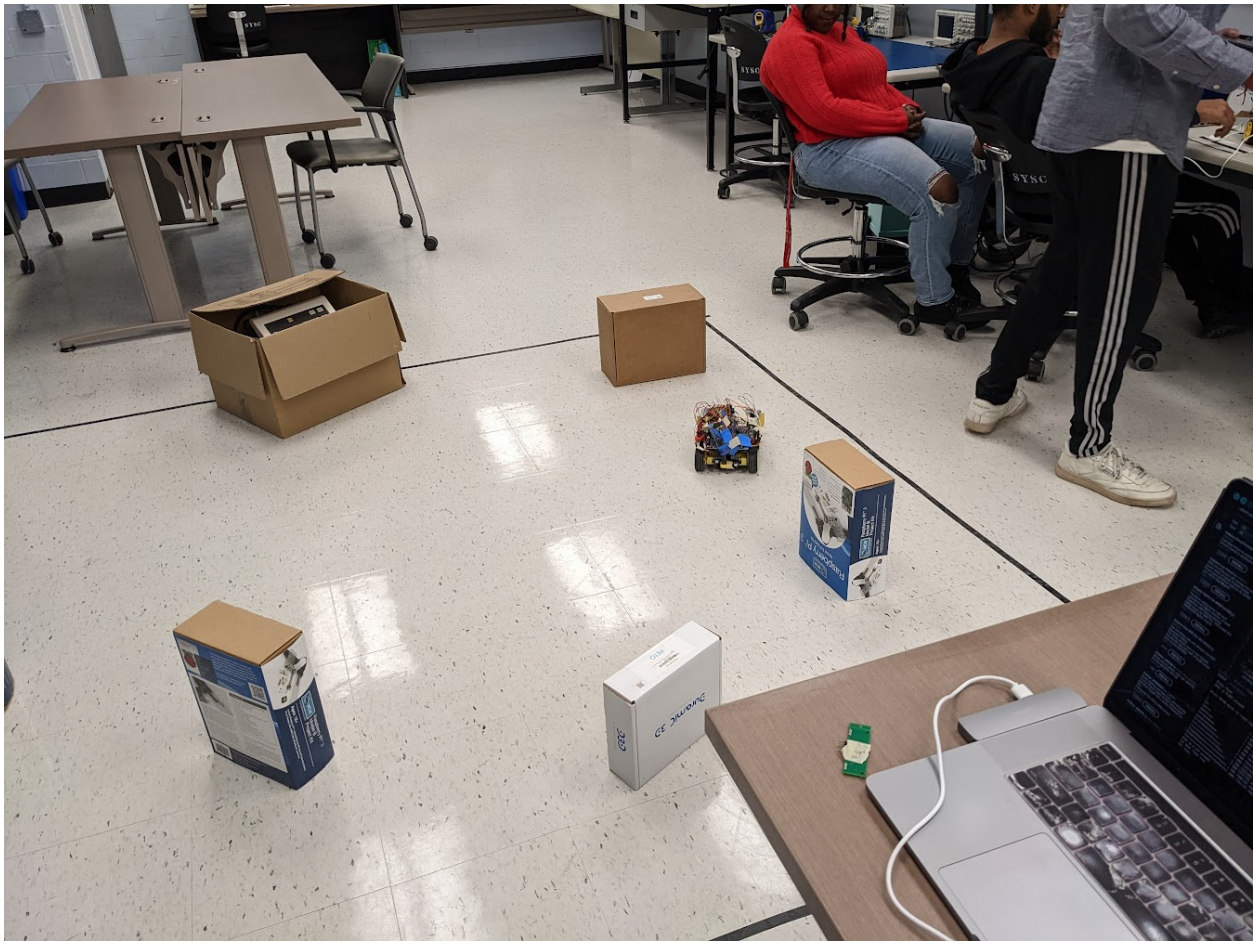
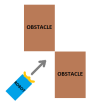
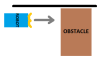


Figure 12. Self-Built Robot Testing Arena

Test ID	Diagram	Description	Pass Condition	Pass Rate
0		Robot approaching the boundary directly perpendicular to the line	Robot detects the boundary and turn left	7/7 (100%)
8		Robot approaching the obstacle directly perpendicular	Robot detects the obstacle and turns left	7/7 (100%)
17		Obstacles placed perpendicular to each other. Robot approaches the intersection at an angle	Robot detects the two obstacles and takes a left and another one right after	7/7 (100%)
26		Obstacle placed with one side on the boundary line. Robot approaches the obstacle being up straight and the boundary line being on the left-hand side parallel	Robot detects the obstacle head on, takes a left and immediately detects the boundary and takes sharp left turns until it is safe to go	7/7 (100%)

4 of the 46 test cases are displayed above with a Test ID (to track test cases), a diagram and description to visualize the test, along with a pass condition and pass rate of each test case. The entirety of the testing cases will be found in Appendix 13.1.

8.2 Customer Testing Results

Our Lab session is on Thursday afternoon. The timestamp on the recording for our team's testing for Arena A is 00:17:45 to 00:24:05 and for Arena B 01:27:05 to 01:32:45.

Arena [A] Result:	Arena [B] Result:
0.5 Obstacles Hit 0 Boundary Crossed 46 Balls Cleared	0.5 Obstacles Hit 0 Boundary Crossed 40 Balls Cleared

The result of the testing in Arena A was better than Arena B in regards to balls cleared due to a few factors; primarily in the change in obstacle placement in the testing arena. The obstacle

placement is a significant factor in the amount of balls that we clear due to the design and logic of the Robot movement. In Arena A the Robot was able to travel and clear a greater area (while traveling in a closed loop) as compared to Arena B where the closed loop travel area was much smaller.

9 Human Resources

9.1 Responsibility Assignment Matrix

Table 9 Key: R = Responsible, A = Approver, C = Consulted, I = Informed

Table 9: Responsibility Assignment Matrix

Activity	Raiyan	Kenny	Paul
Plow prototype design	I	C	R
Arduino schematic Diagram	I	R	C
Sensor Selections	R	C	I
Optimal Travelling Algorithm Research	I	C	R
Robot chassis Assembly	R	I	C
Sensor Integrations	R	C	I
Plow Integrations	I	R	C
Movement Module	C	I	R
Obstacle/ Boundary Detection Module	C	R	C
Path Travelling Logic module	I	C	R
Integration testing	C	R	I
Unit Testing	R	R	R
Functional Testing	I	C	R
Performance Testing	R	I	C

10 Working Code

The link to our GitHub repository with our working code for the Autonomous Snowplow Robot is: https://github.com/SYSC4805-Fall2022/sysc4805_term_project-eggplant_l1g3.

Table 10: Completed Activities Working Code Responsibility Table

Activity	Person Responsible	Commit Link
Movement functional testing: Test that when electrical components are configure in a way that enables movement	Paul	https://github.com/SYSC4805-Fall2022/sysc4805_term_project-eggplant_l1g3/commit/8f8ff7fef23f4ab31def6a14e428a6545ca11e7d
High level algorithm development of obstacle detection module with dummy data	Kenny	https://github.com/SYSC4805-Fall2022/sysc4805_term_project-eggplant_l1g3/commit/6e966798efa16aa09ccb59d4bc13a3608394e4a7
High level algorithm development of boundary detection module with dummy data	Raiyan	https://github.com/SYSC4805-Fall2022/sysc4805_term_project-eggplant_l1g3/commit/54976cb0bd0a409ff574b76eb70597276a86d565#diff-d98ecc778775a7a370f80886e96843968ba5061db83f0b96670d9823a75e1f8a
Implement the main code, the path-travelling algorithm	Paul	https://github.com/SYSC4805-Fall2022/sysc4805_term_project-eggplant_l1g3/commit/ce462873ecc76d4418d038de4a11ca2d0a06cd56#diff-2fb3b29b182cf800b53a777146e8db10f2a1dc557d89967a4d75dba77c7c3111
Implement obstacle detection module with real sensor data	Kenny	https://github.com/SYSC4805-Fall2022/sysc4805_term_project-eggplant_l1g3/commit/6fe25d5d8e18b71f2b209bee2111492b6474ceda
Implement boundary detection module with real sensor data	Raiyan	https://github.com/SYSC4805-Fall2022/sysc4805_term_project-eggplant_l1g3/commit/54976cb0bd0a409ff574b76eb70597276a86d565
Implementation of vehicle movement module	Paul	https://github.com/SYSC4805-Fall2022/sysc4805_term_project-eggplant_l1g3/commit/ce462873ecc76d4418d038de4a11ca2d0a06cd56
Implementation of vehicle obstacle detection module	Kenny	https://github.com/SYSC4805-Fall2022/sysc4805_term_project-eggplant_l1g3/commit/00bdde2fa628e6b8a1520540ee2d2ff22633f063
Implementation of vehicle boundary detection module	Raiyan	https://github.com/SYSC4805-Fall2022/sysc4805_term_project-eggplant_l1g3/commit/54976cb0bd0a409ff574b76eb70597276a86d565#diff-d98ecc778775a7a370f80886e96843968ba5061db83f0b96670d9823a75e1f8a

11. Contributions

Table 12 shows the contributions of each team member for deliverables. The major deliverables are the proposal, progress report, final report, presentation, and autonomous snowplow.

Table 12: Team Member Contributions

Deliverable	Group Member	Contributions
Proposal	Kenny	- Added an overall objective section - Added the tequirement list - Added a test plan Section - Worked on the activity breakdown
	Raiyan	- Added the cost section with the cost baseline - Added the human resource section - Worked on the overall objective - Worked on the test plans section
	Paul	- Added the deliverables section - Added the work breakdown structure - Added the activity breakdown - Added the weekly activity breakdown - Added the test plan section - Added the schedule section, including the schedule network diagram and gantt chart
Progress Report	Kenny	- Updated the requirement list - Revised the cost baseline table - Added the working code section - Added the physical architecture diagram
	Raiyan	- Added the total cost vs activity diagram - Worked on the watchdog timer section - Updated the progress report after the correction of the proposal
	Paul	- Update the schedule network diagram - Update the planned value analysis - Worked on the watchdog timer section - Added the sofware architecture diagram, system state machine and sequence diagrams
Final Report	Kenny	- Added the testing results section -
	Raiyan	- Worked on the test cases section for the obstacle/boundary detection modules - Worked on the conclusion - Cost baseline figures and table, diagram
	Paul	- Added the control chains - Updated the state machine - Added the team contribution section
In Lecture Presentation	Kenny	- Worked on the background slide - Worked on the plow design slides - Worked on sensor justification slides

	Raiyan	- Worked on obstacle detection modelling - Worked on boundary detection modelling
	Paul	- Worked on the project objective slide - Worked on sensor justification sildes - Worked on system arrangement slides - Worked on software overview slides: architecture and state machine
Autonomous Snowplow	Kenny	- Assembled the physical snowplow - Worked on the obstacle detection module - Worked as team to incorporate module and test functionality - Designed the plow
	Raiyan	- Worked on the boundary detection module - Worked as team to incorporate module and test functionality
	Paul	- Worked on the control module - Worked on the movement module - Worked as team to incorporate module and test functionality - Added comments to project

12 Conclusion

Overall, the project went very well. Robot implementation is stable and we met our project requirements of not hitting obstacles and not crossing boundaries. The project was completed on time, as shown in our Gantt chart. We ran into issues over the course of this project such as hardware where the sensors needed to be fixed or software issues such as modifying our algorithms. The project was completed within the budget that was estimated. Due to the issues we ran into, our hours spent on each activity increased for each developer. However, we completed it on time in the estimated days that we predicted. The project overall cost us $5750 which was less than our prediction of $6250.

13 Appendix

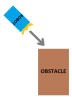
13.1 System Test Cases

Boundary test cases

Test ID	Diagram	Description	Pass Condition	Pass Rate

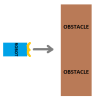
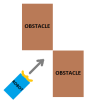
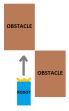
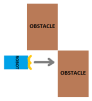
0		Robot approaching the boundary directly perpendicular to the line	Robot detects the boundary and turn left	7/7 (100%)
1		Robot approaching the boundary line at an angle from the left side	Robot detects boundary and turns right	7/7 (100%)
2		Robot approaching the boundary line at an angle from the right side	Robot detects the boundary and turns left	7/7 (100%)
3		Robot travelling parallel to the boundary line towards the right	Robot continues to travel along the line parallel	7/7 (100%)
4		Robot travelling parallel to the boundary line towards the left	Robot continues to travel along the line parallel	7/7 (100%)
5		Robot approaching the perpendicular intersection of the boundary line at an angle	Robot detects the line and turns left	7/7 (100%)
6		Robot approaching the perpendicular intersection of the boundary line at a	Robot detects the line and turns right	7/7 (100%)
7		Robot approaches the perpendicular intersection alongside the boundary line	Robot detects the line and turns left	7/7 (100%)

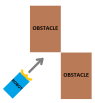
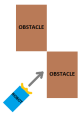
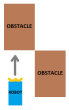
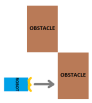
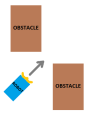
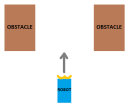
Single Obstacle test cases

Test ID	Diagram	Description	Pass Condition	Pass Rate
8		Robot approaching the obstacle directly perpendicular	Robot detects the obstacle and turns left	7/7 (100%)
9		Robot approaches the obstacle perpendicular to it's edge (left edge)	Robot detects the edge of the obstacle and turns left	6/7 (86%)
10		Robot approaches the obstacle perpendicular to it's edge (right edge)	Robot detects the edge of the obstacle and turns left	6/7 (86%)
11		Robot approaches the obstacle at a 90 degree angle	Robot detects the edge and proceeds to turn left	7/7 (100%)
12		Robot approaches the obstacle at an angle	Robot detects the edge and turns left	7/7 (100%)
13		Robot approaches the obstacle at an angle	Robot detects the edge and turns left	7/7 (100%)
14		Robot travels parallel to the obstacle on the right hand side	Robot continues to go straight untill further detection occurs	7/7 (100%)

15		Robot travels parallel to the obstacle on the left hand side	Robot continues to go straight untill further detection occurs	7/7 (100%)

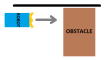
Multiple Obstacles test cases

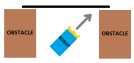
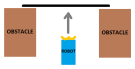
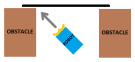
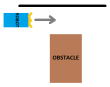
Test ID	Diagram	Description	Pass Condition	Pass Rate
16		Robot approaches two obstacles placed right next to eachother	Robot turns left after detecting obstacles	7/7 (100%)
17		Obstacles placed perpendicular to eachother. Robot approaches the intersection at an angle	Robot detects the two obstacles and takes a left and another one right after	7/7 (100%)
18		Obstacles placed perpendicular to each other. Robot approaches the obstacle head on straight with the second obstacle being on the right.	Robot detects the obstacle in front of it and turns left.	7/7 (100%)
19		Obstacles placed perpendicular to each other. Robot approaches the obstacle head on straight with the second obstacle being on the left.	Robot detects the obstacle in front of it and turns left right after it reverses to continue its pathway	6/7 (86%)

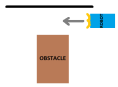
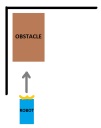
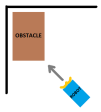
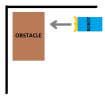
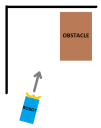
20		Obstacles placed perpendicular to each other. Robot approaches the obstacle at an angle.	Robot detects obstacle, reverses and turns left	7/7 (100%)
21		Obstacles placed perpendicular to each other. Robot approaches the obstacle at an angle.	Robot detects obstacle and turns left and a left again to continue the pathway	7/7 (100%)
22		Obstacles placed perpendicular to each other. Robot approaches the obstacle head on near the edge.	Robot detects obstacle and turns left to continue pathway	7/7 (100%)
23		Obstacles placed perpendicular to each other. Robot approaches the obstacle head on near the edge.	Robot detects obstacle and turns left and again a left to continue pathway	6/7 (86%)
24		Obstacles placed perpendicular to each other with an opening in the middle. Robot approaches the obstacles opening.	Robot detects the obstacles on both sides and if the opening is wide enough, it will proceed, else left or reverse	6/7 (86%)
25		Obstacles placed next to each other with an opening in the middle. Robot approaches the opening between the obstacles	Robot detects the area around and passes through the gap straight	7/7 (100%)

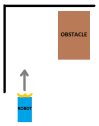
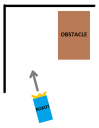
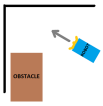
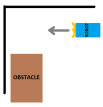
Obstacle and Boundary test cases

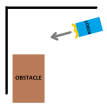
Test ID	Diagram	Description	Pass Condition	Pass Rate

26		Obstacle placed with one side on the boundary line. Robot approaches the obstacle being up straight and the boundary line being on the left hand side parallel	Robot detects the obstacle head on, takes a left and immediately detects the boundary and takes sharp left turns until its safe to go	7/7 (100%)
27		Obstacle placed with one side on the boundary line. Robot approaches the obstacle at the intersection between the obstacle and the boundary line	Robot detects both the boundary and the obstacle and turns left, if the turn becomes too tight, reverses and takes a left	7/7 (100%)
28		Obstacle placed with one side on the boundary line. Robot approaches the boundary line with the obstacle placed on the right	Robot detects the boundary line and takes a left turn and continues its path	7/7 (100%)
29		Obstacle placed with one side on the boundary line. Robot approaches the boundary line with the obstacle placed on the left	The Robot detects the line and the boundary. It reverses until there is no obstacle remaining to take a left turn	7/7 (100%)
30		Obstacle placed with one side on the boundary line. Robot approaches the obstacle at the intersection between the obstacle and the boundary line at an angle	Robot detects the obstacle and the boundary, proceeds to take a left	7/7 (100%)
31		Obstacle placed with one side on the boundary line. Robot approaches the obstacle being up	Robot detects the obstacle in the front, and takes a left turn continuing to follow it's pathway	7/7 (100%)

		straight and the boundary line being on the right hand side parallel		
32		The two obstacles are placed next to each other at the edge of the boundary line. The obstacles contain an opening in the middle. The robot approaches the gap at an angle with the obstacle and boundary in front.	The robot would detect the obstacle and boundary, since it is a tight space to turn left, it will reverse and turn left	6/7 (86%)
33		The two obstacles are placed next to each other at the edge of the boundary line. The obstacles contain an opening in the middle. The robot travels between the gap and comes in contact with the boundary first.	The robot would detect the boundary line and since there are obstacles on the side, it will reverse and take a left turn avoiding obstacles.	7/7 (100%)
34		The two obstacles are placed next to each other at the edge of the boundary line. The obstacles contain an opening in the middle. The robot approaches the gap at an angle with the obstacle and boundary in front.	The robot would detect the obstacle and boundary, since it is a tight space to turn left, it will reverse and turn left	7/7 (100%)
35		There is a obstacle placed parallel to the boundary line with an opening existing between the obstacle and the boundary. The Robot approaches parallel to the boundary line being on it's left side.	The robot detects there is an obstacle and and a boundary line on both sides, so it continues going straight until further detection	7/7 (100%)

36		There is a obstacle placed parallel to the boundary line with an opening existing between the obstacle and the boundary. The Robot approaches parallel to the boundary line being on it's right side.	The robot detects there is an obstacle and and a boundary line on both sides, so it continues going straight until further detection	6/7 (86%)
37		There is an obstacle placed at an intersection between two boundary lines, creating a 90 degree angle. The robot approaches the obstacle with the boundary line being adjacent towards the left.	Robot detects the obstacle head on, takes a left and immediately detects the boundary and takes sharp left turns until it is safe to go	7/7 (100%)
38		There is an obstacle placed at an intersection between two boundary lines, creating a 90 degree angle. The robot approaches the edge of the obstacle	The robot detects the obstacle edge and proceeds to turn left and continue	7/7 (100%)
39		There is an obstacle placed at an intersection between two boundary lines, creating a 90 degree angle. The robot approaches the obstacle with the boundary line being adjacent towards the right.	The robot detects the obstacle in front of it, and proceeds to take a left and continue it's pathway	7/7 (100%)
40		There is an obstacle placed near the on the boundary line, and on the left hand side there is a intersection of the two boundary lines.	The robot detects the obstacle, and turns left. A short instance after it will face the boundary, so it turns left again to clear out	6/7 (86%)

		The robot approaches the obstacle at an angle	of the zone.	
41		There is an obstacle placed near the on the boundary line, and on the left hand side there is a intersection of the two boundary lines. The robot approaches the obstacle head on, with the boundary being on the left side.	The Robot detect the boundary line first, and turns left. An instant later, it would turn left again since there is another line in front of it.	7/7 (100%)
42		There is an obstacle placed near the on the boundary line, and on the right hand side there is a intersection of the two boundary lines. The robot approaches the obstacle at an angle	The Robot detect the boundary line first, and turns left. An instant later, it would turn left again since there is another line in front of it.	7/7 (100%)
43		There is an obstacle placed near the boundary line, and on the front there is an intersection of the two boundary lines. The robot approaches the boundary at an angle	The Robot detects the boundary line first, and turns left. An instant later, it would turn left again since there is a boundary in front of it.	7/7 (100%)
44		There is an obstacle placed near the boundary line, and on the right hand side there is an intersection of the two boundary lines. The robot approaches the boundary line head on, with the obstacle being on the left side.	The Robot detects the boundary line first, and turns left. An instant later, it would turn left again since there is an obstacle in front of it.	7/7 (100%)

45		There is an obstacle placed near the boundary line, and on the right hand side there is an intersection of the two boundary lines. The robot approaches the obstacle at an angle	The Robot detects the boundary line first, and turns left. An instant later, it would turn left again since there is an obstacle in front of it.	7/7 (100%)