SYSC 4805: Computer Systems Design Lab

FINAL REPORT

Team Spiro Disco Ball

Group #5

Aedyn Ladd	*101068855*
David Casciano	*101069255*
Daniel Tura	*100995028*
Tadhg McDonald-Jensen	*101047069*

A. ABSTRACT

For more than six months of the year, more than 65% of Canada's landmass is blanketed with snow [1]. For some, snow is a source of fun and for others a nuisance - but for a select few, overexertion due to snow removal can be deadly. Hospitals indicate that the duration and volume of snowfall are directly correlated with an increase in hospital visits due to heat-related injuries in susceptible individuals [2]. Our team proposes a system capable of following a predetermined path and removing snow as it falls, nullifying the risk of injury and increasing the enjoyability of the winter months. To do this we have created a robot that makes use of a process called Boustrophedon Cellular Decomposition (BCD) in order to provide full area coverage. This report follows the design process for such a robot, including items such as: project management tasks, architecture, and testing.

Below is a link to our public github repository as an alternative to the code that has been submitted directly via brightspace. If any issues should arise with regards to viewing of code please don't hesitate to contact Aedyn Ladd through his cmail: aedynladd@cmail.carleton.ca

https://github.com/SYSC4805-Winter2022/sysc4805-project-group-5-l2-spiro-disco-ball

PLEASE NOTE: We have included blown-up versions of images found in this report to ensure that they are visible, these can be found in the appendices section, after the bibliography.

TABLE OF CONTENTS

1. Charter

In seven weeks, our group plans to have fully developed a system capable of navigating and removing snow cover from a predetermined path.

1.1 Overall Deliverables

Our final deliverables will include a robot capable of navigating a user-defined path while avoiding damage to obstacles found in its dynamically changing environment. This will be done using a variety of sensors and actuators available to us in the coppeliaSim simulation program and a full description of how the system was built and corresponding vendors of the used components.

The robot will interface with coppeliaSim through a script developed using Python. The python script and robot model will be delivered with detailed instructions indicating how to run the system, as well as with a final report indicating the process in which the robot was built and insights into the design.

The development of the model will be broken down into a series of 28 unique activities distributed evenly among group members that will dictate the progression of our development cycle as we move through the term. At the end of the term, our findings will be presented to our colleagues, teaching assistants, and professor in a formal presentation.

2. Scope

The system we are intending to create will be described fully in this section. This will include a set of strict requirements that we will follow while designing and building our robot, as well as details describing how we intend to accomplish the tasks at hand. Along with this, we will also include a breakdown of how the system will be tested - taking into account factors such as the provided simulation environments and unit testing for code.

2.1 Requirements

This section defines requirements set out by the project outline, as well as requirements elicited by our group with respect to how the system should function. The table below describes the various requirements our project will meet. It is important to note that seeing as these requirements are elicited from the supporting project documentation that the owner of all requirements is the professor.

The risk column is filled based on the following definition:

- Event Risks: A supplier may go out of business during the project.
- Non-Event Risks: Ambiguity in one requirement
- Emergent Risks: being unknowable unknowns

Table 1: *Elicitation of project requirements as described by the 'Project Description'*

Requirement		Rationale	Type	Priority	Risk
A1	While idle, the size of the robot will be constrained to a space of 0.5 x 0.8 x 1 meters.	Size constraints are required to ensure the robot's environment properly mimics real-life conditions. This also ensures no extraneous parts are used	Functional Requirement	High	Event
A2	While in motion the size of the robot will never exceed a space of 1 x 0.8 x 1 meter.		Functional Requirement	High	Event
B	The system will be able to remove snow from an area not exceeding 144 sq. m, within an allotted time frame of 5 minutes, so long as the volume of snow does not far exceed the height of the robot.	This requirement specifies a realistic zone in which our robot is effectively able to operate. Any increase in the area will result in a decreased performance.	Functional Requirement	High	Event

C	The velocity of the robot will be limited to a maximum speed of 2 m/s. This will be calculated as a function of the angular velocity of the joints, and the circumference of the wheels.	Speed restraints are required to ensure the robot is able to maintain a sense of realism.	Quality-in-use Requirement	High	Event
D	Maximum torque will be limited to realistic values supporting the components used in the assembly of the robot.	Torque limit is set to ensure realism in the simulated environment	Quality-in-use Requirement	Medium	Non-Event
E	The robot will be controlled in CoppeliaSim through the use of the external API via a script written in Python.	Python is a simple programming language with a solid foundation. It allows for further development if desired.	Interface Requirement	Medium	Non-Event
F	The system in question will avoid obstacles presented to itself so long as obstacles are not intentionally attempting to actively disrupt operations at a speed faster than the system is able to operate.	Robot should never cause financial or bodily harm. We also specify that any harm caused by misuse of the robot is not captured by this requirement.	Human Factor Requirements	High	Event
G	A list of feasible components as well as their specifications, cost, and availability recorded from technology vendors will be maintained throughout the course of the project	This requirement serves to ensure the realism of the robot through its development life cycle. By maintaining a list of components we can ensure that the final product can be delivered.	Quality-in-use Requirement	Medium	Emergent
H	The system will have the ability to operate in whatever environment it is placed in so long as it follows requirements specified by B.	The system has a range of flexibility with respect to where it can operate.	Non-Functional Requirement	High	Event
I	A repository with all documentation will be maintained through the project lifecycle	This is to ensure transparency in the development of our project.	Process Requirement	Medium	Emergent

The requirements mentioned will be mentioned in further sections through the use of their tagging letter found in the first column of the above table. All requirements identified here will be delivered by the end date of the project.

2.2 Work Breakdown Structure

To satisfy the requirements listed above in Section 2.1, the group has composed twenty-eight tasks to create the Snow Removal Robot. These tasks are displayed in a hierarchical manner below, categorized into Project Management, Robot Modeling Hardware, Software, and Fail-Safe.

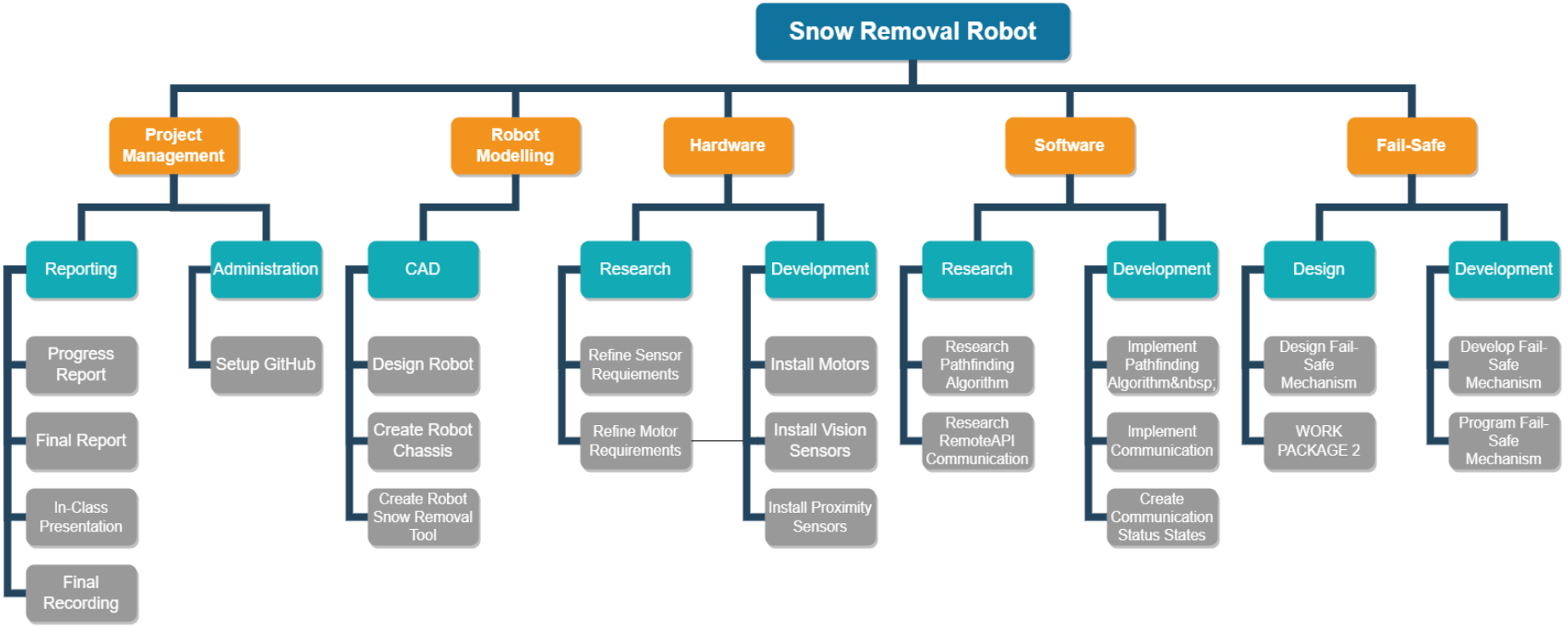


Figure 1: *A work breakdown structure of the activities set to be completed in the project*

Given Figure 1 on the previous page, Table 2 below comprises the tasks titles, descriptions, and requirements that it satisfies.

***Table 2**: A lookup table of task information*

Activity	Description	Satisfying Requirements
Draw Schematic	Create a rough draft of the robot with ideas of implementation	A1 and A2
Setup Github	Setup Github Repository, which includes the code base and Kanban boards	I
Refine Sensor Requirements	Identifying available sensors from real vendors	G and D
Refine Motor Requirements	Identifying available motors from real vendors	G, D and, C
Create Robot Body	Design robot in the coppeliaSim environment	A1, A2, and E
Research RemoteAPI Communication	Research available functions to be used through the RemoteAPI. These functions must be compatible with Python	I and E
Research Pathfinding Algorithm	Research pathfinding options to best clear the snow.	I and E
Design Fail-Safe Mode	Design a failsafe mode, which will allow the robot to fail gracefully in the case of a malfunction.	F, and H
Create Robot Snow Removal Tool	Implement a method for the robot to clear snow most effectively via CAD software.	A1, A2, and B
Install Motors	Implement physical components in CoppeliaSim and logical components in Python	G, A1, A2, C, and D
Install Vision Sensors	Implement physical components in CoppeliaSim and logical components in Python	G, A1, and A2
Install Proximity Sensors	Implement physical components in CoppeliaSim and logical components in Python	G, A1, and A2
Implement Pathfinding Algorithm	Program the algorithm in Python. This algorithm will communicate via RemoteAPI	I and E
Implement Communication via RemoteAPI	Program connection for bidirectional communication through RemoteAPI	I and E

Unit test sensors/vision	Create and perform tests to verify the correct function of the sensors	H
Unit Test Motors	Create and perform tests to verify the proper operation of the motors	H
Integration Testing	Test the complete robot with the test environment to identify and correct bugs	H

3. Schedule

The project is divided into thirty-three activities spread to ten weeks. The network diagram and Gantt chart identify activities that are float and critical (blocking). The work is fairly distributed equally to each member and we are confident that the project will be delivered to stakeholders by week ten.

3.1 Schedule Network Diagram:

The figure below shows the order and time analysis of each deliverable. Critical path analysis looks for the earliest and latest points at which tasks can begin and end. The digits on top of each box represent the est/lst and the arrows represent the steps of the tasks.

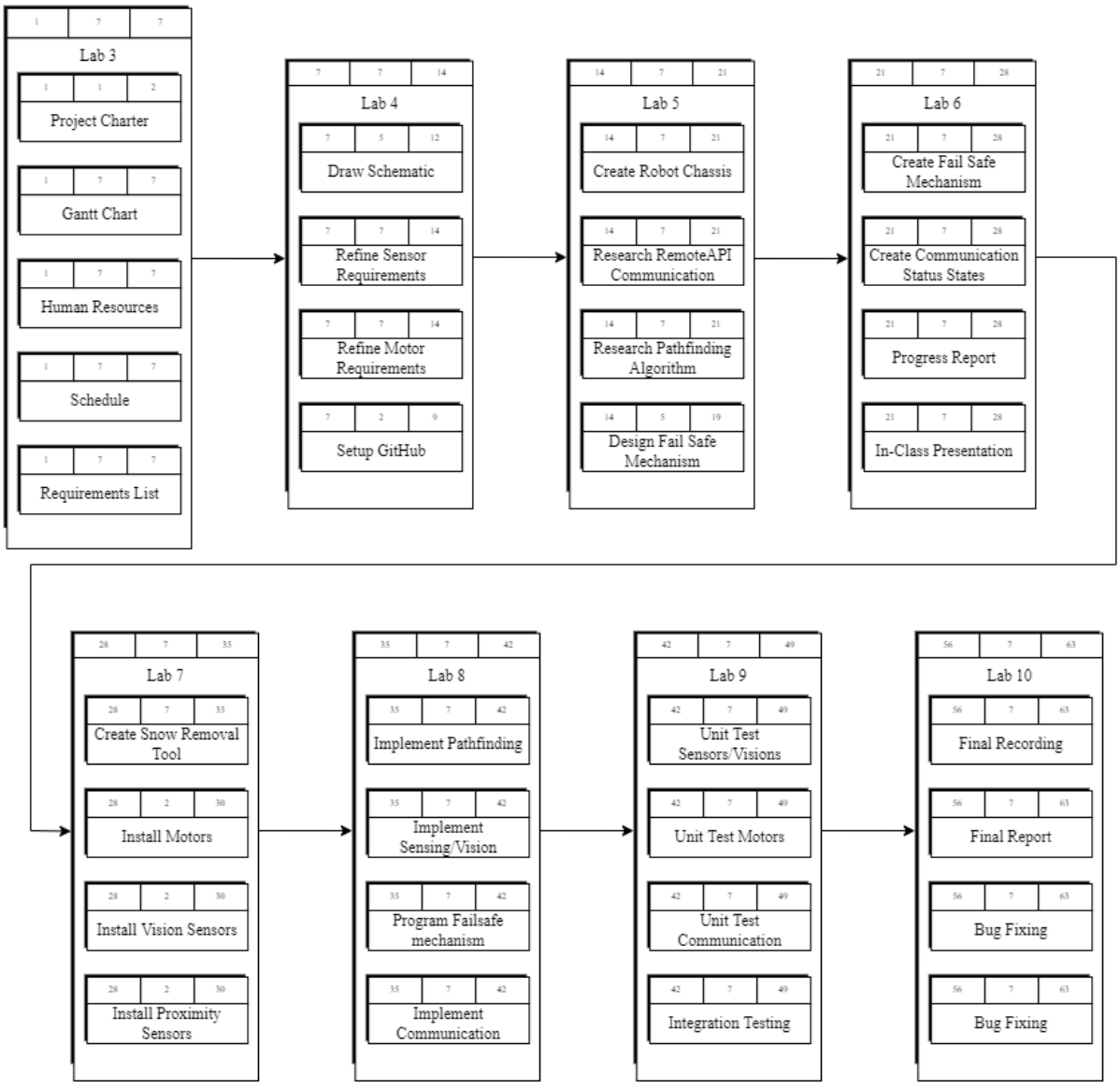
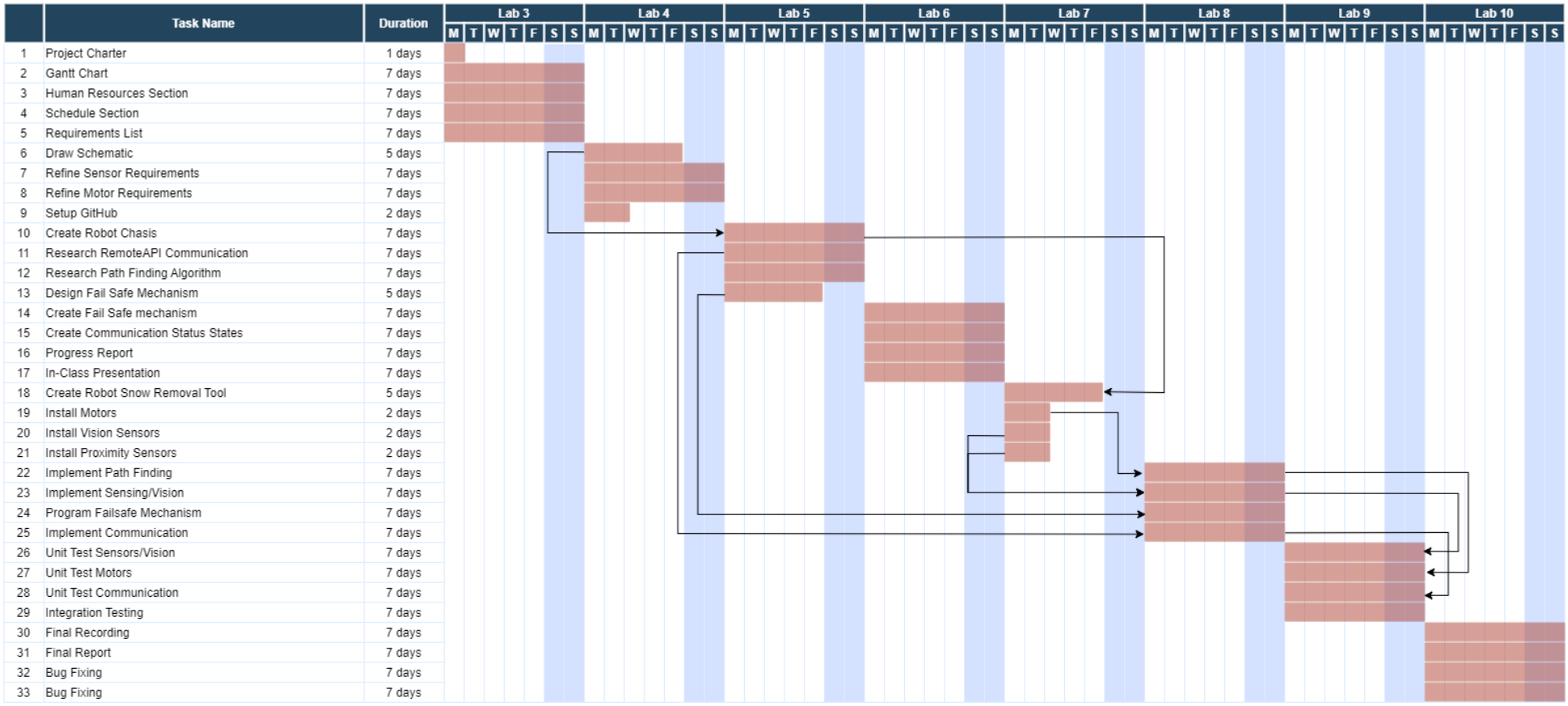


Figure 2: *Schedule diagram of the project deliverables*

3.2 Gantt Chart:

The Gantt chart is composed of three components. Activities are listed under the task name column, the number of days planned to complete the tasks are listed under the duration column, and the horizontal bar under labs is used to demonstrate tasks concurrency. In Addition, the order (prerequisites) of tasks are represented by using the arrows leaving and pointing to the horizontal bars.



	Task Name	Duration
1	Project Charter	1 days
2	Gantt Chart	7 days
3	Human Resources Section	7 days
4	Schedule Section	7 days
5	Requirements List	7 days
6	Draw Schematic	5 days
7	Refine Sensor Requirements	7 days
8	Refine Motor Requirements	7 days
9	Setup GitHub	2 days
10	Create Robot Chasis	7 days
11	Research RemoteAPI Communication	7 days
12	Research Path Finding Algorithm	7 days
13	Design Fail Safe Mechanism	5 days
14	Create Fail Safe mechanism	7 days
15	Create Communication Status States	7 days
16	Progress Report	7 days
17	In-Class Presentation	7 days
18	Create Robot Snow Removal Tool	5 days
19	Install Motors	2 days
20	Install Vision Sensors	2 days
21	Install Proximity Sensors	2 days
22	Implement Path Finding	7 days
23	Implement Sensing/Vision	7 days
24	Program Failsafe Mechanism	7 days
25	Implement Communication	7 days
26	Unit Test Sensors/Vision	7 days
27	Unit Test Motors	7 days
28	Unit Test Communication	7 days
29	Integration Testing	7 days
30	Final Recording	7 days
31	Final Report	7 days
32	Bug Fixing	7 days
33	Bug Fixing	7 days

Figure 3: *Gantt chart of activities duration and order*

4. Human Resources:

Each member plays a different role to complete the project. For each deliverable, one member is responsible to implement the task and another member approves. The responsibility matrix table illustrates the accountability of members divided by activities.

4.1 Responsibility Matrix

Deliverables, names and responsibility assignments are components of the responsibility matrix. Deliverables are the list of tasks that are to be performed. Responsibility assignments are represented as R(responsible) and A (Approve).

***Table 3**: Responsibility Matrix of team members*

Deliverables	Daniel	Aedyn	Tadhg	David
Project Statement		R		A
Gantt Chart		A		R
Human Resources	R		A	
Schedule	A		R	
Requirements List		R		A
Draw Schematic		R	A	
Refine Sensor Requirement	R			A
Refine Motor Requirement	A		R	
Setup Github		A		R
Create Robot		R		A
Research RemoteAPI Communication	R	A		
Research Path Finding Algorithm	A		R	
Design Fail Safe			A	R
Create Communication States	R			A
Progress Report Task		A	R	
Create Fail-Safe Mechanism	A			R

In-Class Presentation		R	A	
Create Robot Snow Removal Tool		R		A
Install Motors	A			R
Install Vision Sensors	R		A	
Install Proximity Sensors		A	R	
Implement Path Finding		R	A	
Implement Sensing/Vision			R	A
Implement Failsafe	A			R
Implement communication	R		A	
Unit test Sensors/Vision	A		R	
Unit test Motors	R			A
Unit test communication		A		R
Integration Testing		R		
Final Recording	R		A	
Final Report	A		R	
Bug Fixes		R		A
Bug Fixes		A		R

4.2 Breakdown of Team Contributions

This section will serve to breakdown the contributions made by our team throughout the course of this project, topics covered in this section will include breakdowns of what section of code were written by who as per github, and breakdowns of each project deliverable that has been delivered up to and including this report, progress report, proposal, and the group presentation.

All of these details will be discussed in individual tables on the following page.

4.2.1 Code Contributions

By using Github Insights, we are able to identify contributions to our github repository throughout the term.

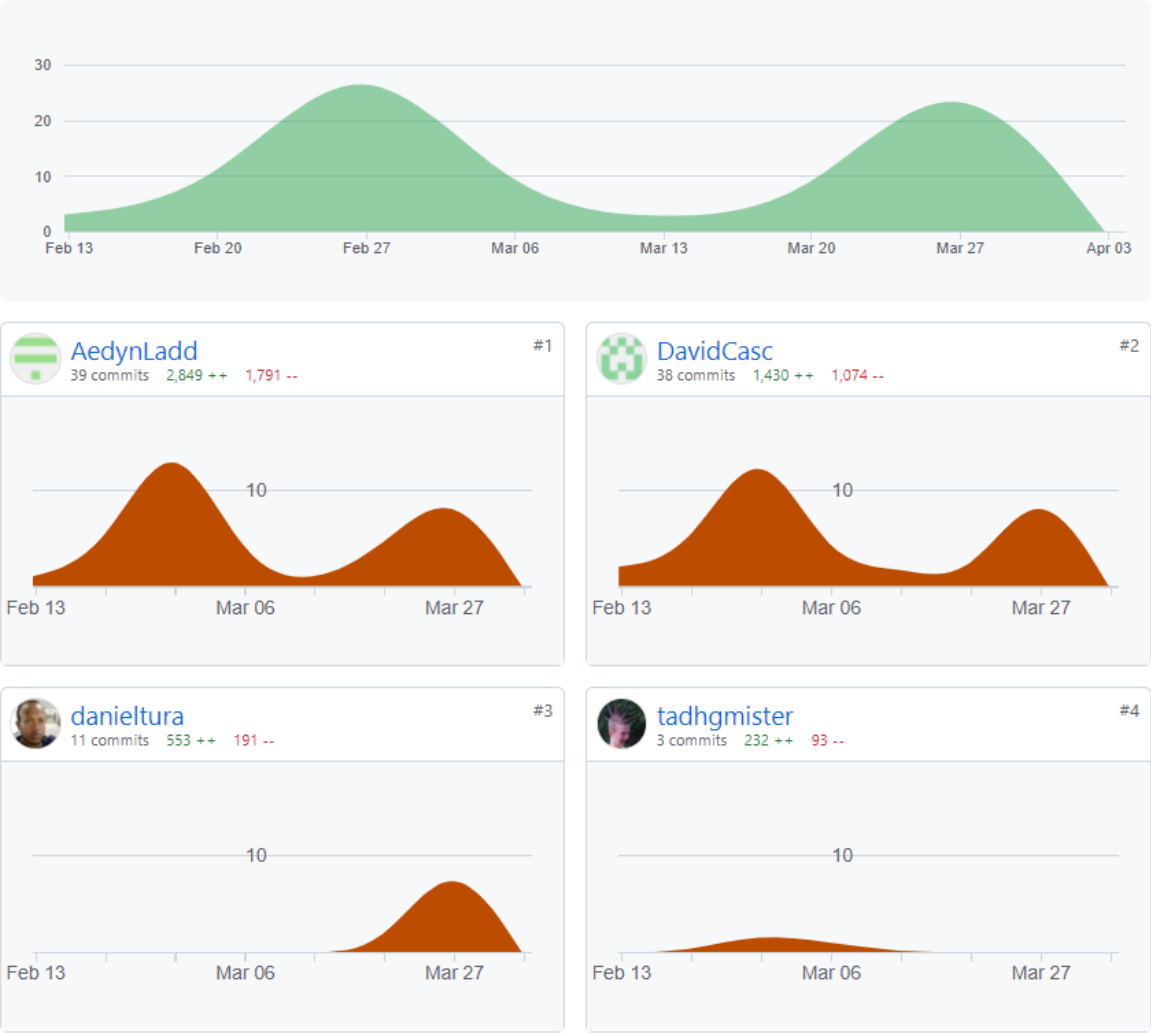


Figure 4: Github Contribution breakdown

Figure 4 above dictates the breakdown of frequency and amount of code committed by each member of our team, this includes the robot model, testing environments, documents, and all necessary code used to make our robot run.

The robot model was developed in its entirety early on in the semester by Aedyn with the advice of other group members. Later on near the end of the semester the architecture was updated to include a front bumper composed of a single proximity sensor used to detect incoming obstacles in the event that an obstacle was missed or suddenly appeared quickly.

The rest of the code contributions will be broken down in the following table.

Table 4: Code related contribution

Contribution	Purpose	Contributor
MotorControl	The motor control class provides helper functions to control the movement of wheels and positioning	Tadhg
RobotBody.lua		
RobotBody	This script contains all functions needed by the robot so that it can function when loaded in by the robot models child script. It interfaces with several other functions to provide the robot with all its functionality	David, Aedyn
Mapping	The robot makes use of a tomographic style of mapping to allow the robot to remember its environment. It makes use of the lilypad and math to identify objects, and will be discussed later on	Aedyn
Morphological	A map cleaning function makes use of a combination of erosion and dilation - a well known opening operation used to close small holes as the model detects its environment	David
GoTo	This function is used by the robot to navigate the map, it was primarily developed by Aedyn and further tuned after the fact by Tadhg	Aedyn, Tadhg
State Machine	The robot uses a list of points to navigate its environment, as it moves through each point it removes them from the list and attempts to reach the next.	Aedyn

Table 4: Code related contribution

PathFinding.lua		
Boustrophedon	This is the main algorithm used to plan paths in the back and forth boustrophedon pattern	Aedyn
Cellular Decomposition	Cellular decomposition is a technique used to decompose an area into a variety of cells that can be traversed	Aedyn
Algorithms	A series of algorithms such as DBSCAN, and harris edge detection are used by the robot to help in path planning	Aedyn and David
Testing files		
Testing files	Testing files make use of previously described functions to ensure they function exactly as expected	Daniel
Paths	Path files are used post simulation to determine what path the robot followed. This lets us determine the effectiveness of simulations	Aedyn

4.2.2 Project Proposal Contributions

The table below provides a breakdown of contributions related to the first iteration of our project, the project proposal. Proof reading is omitted from this table, but it can be assumed that all group members played a role in editing the report after the majority of sections were written. This table only displays main contributors for each section.

Table 5: Project Proposal breakdown

ID	Section	Contributor(s)
1	Charter	Aedyn (1, 1.1)
2	Scope	Aedyn (2, 2.1, 2.2, 2.3), Daniel (2.1), David (2.2)
3	Schedule	David (3, 3.1, 3.2)

Table 5: Project Proposal breakdown

4	Human Resources	David and Aedyn (4, 4.1)
5	Bibliography	Aedyn (5)

4.2.3 Project Progress Report Contributions

The table includes contributions made to the Project Progress report, this section only includes updates to proposal sections, and any newly added sections, all previously existing sections are assumed to be the same.

Table 6: Project Progress Report breakdown

ID	Section	Contributor(s)
-	Updates to Proposal	Aedyn (2.3 as per comments)
5	Architecture	Aedyn (5, 5.1)
6	State Chart	David (6)
7	Sequence Diagram	David (7) and Aedyn (7)
8	Project Budget	David and Daniel

4.2.4 Project Presentation Contributions

The table below provides the reader with a breakdown of the development of the end of semester presentation.

Table 7: Project Presentation breakdown

Slide(s)	Sections Described	Contributor
1-8	Introduction, Overview, Parts	David - Designed/Presented
9-18	Technical Components, Motor Control, Pathfinding, Morphological Processing	Aedyn - Designed (All) Tadhg - Presented (9-11) Aedyn- Presented (12-15) David - Presented (15-18)
19-23	Budget, and Conclusion	Daniel - Designed(All) David - Designed(23) Daniel - Presented(All)

4.2.5 Final Report Contributions

Lastly, here is a breakdown of the contributions for this document, the final report.

Table 8: Project Final Report breakdown

ID	Section	Contributor(s)
Updates to Progress Report		
3.3	Testing	- Moved to Section
4	Contributions	Aedyn and David
5	Architecture	David (5.2 and 5.3) Aedyn (5.4)
7	Sequence Diagrams	Aedyn (7)
Newly Added Final Report Sections		
9	Control Charts	David (9)
10	Testing and Results	Daniel (10)
11	Conclusion	Tadhg (11) and Aedyn (11.3)

5. Architecture

When designing our robot, we started with several rough ideas on what our overall design should be based on the task at hand. While brainstorming several ideas came to light: a robot consisting of several smaller entities that would all work simultaneously to plow a larger area at once, a robot with a large storage system that instead of directing plowing the snow would store it and then dump snow in a designated area, we even considered a robot with a set of omni-directional wheels allowing it to have a full 360 degrees of motion at a whim. We decided that for this specific project - simple was better - and that our robot needed three main components to effectively meet all the requirements specified in the course outline:

1) The robot needs a way to navigate from location A to location B.
2) The robot needs a set of sensors to interact with and manage the state of its environment
3) The robot needs a plow that is able to move in various directions as situations evolve over time.

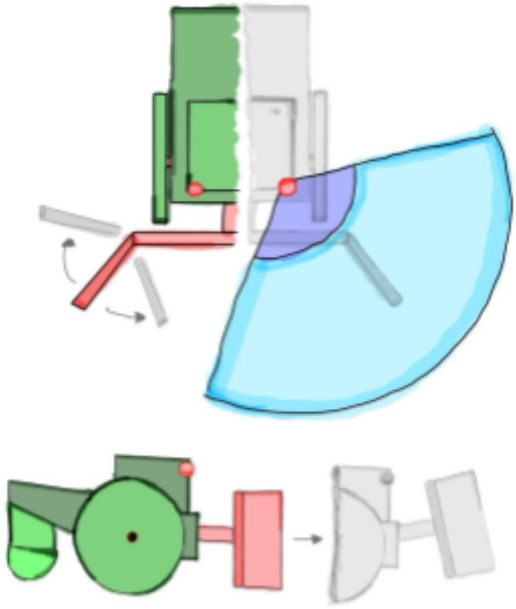


Figure 4: Initial Architectural Concept

The navigation system is a simple set of 2 wheels used to control where the robot is able to go, along with a castor wheel which acts as a third point of contact for additional stability. The robot also makes use of a series of proximity sensors allowing it to react to its environment. The proximity sensors are housed on the head of the robot and provide a large range of vision on the area directly in front of and to the left and right of the robot. Along with this we also have a series of vision sensors on the bottom of the robot that allow our system to identify when it has left (or is bordering on) the designated "clean up" zone. As identified above, most of the sensors are housed in a large box at the head of the robot. This is to provide a sense of the space required to house the components required to allow our robot to function, including the ability to add additional sensors as we require in the future if needed without worrying about the space required to do so. The head is raised above the level of the plow to ensure that the proximity sensors are able to monitor the surrounding environment without being blocked.

The proximity sensors work on a premise similar to that of tomography. They are attached to a swivel, which allows them to freely rotate around the head of the robot to produce projections of the environment around it based on the distance of objects located around it in 1 degree segments. Using

basic trigonometry we could map a projection of the robots available space into a two dimensional array as a function of the object's distance provided by the sensor, and the current angle of the sensor.

Lastly - and most important to our robot - is the mechanism used to move snow across the map, a plow. The plow we designed is fairly simple and consists of 4 main components: An arm, a body, and 2 separate wings. Using the motor connected to the main arm, the entire plow is able to move freely up and down as situations permit it to do so. The center body of the plow is fixed to this arm and will always be facing forward, to maximize the space our robot is able to operate in. The center body is flanked with left and right wings that are able to move with around 220 degrees of freedom. This allows for more snow to be moved in a larger area, as well as provide the ability to micromanage the areas in which our robot is plowing and ensure no snowballs are lost while plowing.

The overall physical architecture of our system is rather simple, to show more personality and take ownership of our work we decided to add color and personality to the design, thus creating a system that resembles - very loosely - a frog. In the preceding section we will be discussing the internal architecture and processing mechanisms that our system makes use of.

5.1 Timer Triggered Framework

Given the requirements our system is required to be reactive to the state of the environment around it, a timer triggered framework was the most appropriate fit for our design. Running on a frequency of 500Hz, every 0.002 seconds our system makes note of the environment it is in, this allows it to perform maneuvers and make adjustments as required. This was the optimal scenario for our robot to map out the region that it needs to plow, and proceed to clear it of snow. Along with this it also allows for a more constant decision making when approaching obstacles and an easier way for our robot to interact with the environment rather than react to it.

This relates to the real world as the act of driving and by extension plowing snow, is timer based. The expectation is that drivers maintain constant awareness of their environments by checking mirrors every 5-8 seconds and making small changes in anticipation of their environment, rather than in reaction to it.

5.2 Environmental Awareness and Mapping mechanism

To create environmental awareness we implemented a mapping mechanism that is relative to the starting point of the robot. To do this a Bluetooth beacon called the lilypad is paired with the robot. Utilizing Bluetooth 5's location abilities, we are able to find the location of the robot. Given the signal strength, Bluetooth receivers are able to infer the distance of the transmitting device. As well, new features added to Bluetooth 5 allow the device to have awareness of the angle relative to the device; these features are Angle of Arrival (AoA) and Angle of Departure. Through these technologies the

robot has enough awareness to plot the observed environment with trigonometric equations. To capture the obstacles within the environment, we can use the swiveling ultrasonic sensors. Using more trigonometry we could map a projection of the robot's available space into a two dimensional array as a function of the object's distance provided by the sensor, and the current angle of the sensor.

$$X\ component\ =\ sin(eye\ orientation)\ *\ distance \quad (1)$$

$$Y\ component\ =\ pitch\ *\ cos(eye\ orientation)\ *\ distance \quad (2)$$

Where our pitch angle is defined by the orientation of each eye with the lilypad - or the orientation of the lilypad with the robot body plus the currently tracked angle of the eye. The sign of our pitch value indicates the direction our eye is looking. The figure below describes an example of what the robot sees as it is moving around its environment, in the following section we will discuss how this noise is turned into solid obstacles.

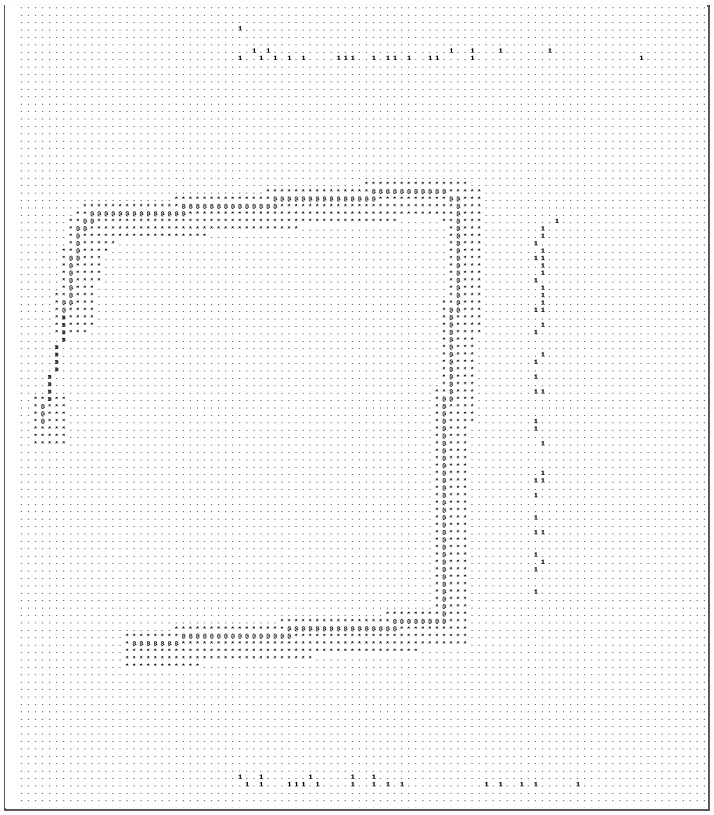


Figure 5: Object Detection Example

5.3 Morphological Processing

To remove noise from the noise from the image of the recorded environment, we utilized morphological processing. To perform a morphological change we iterate an element or kernel through the image morphing it. The two main morphological changes are dilation and erosion; dilation adds new bits while erosion removes bits. By performing a dilation followed by an erosion one can perform a closing operation. By performing a closing operation on the data points collected by the sensor we take what's known by our robot and attempt to quite literally fill in gaps of knowledge and create the edges of the obstacles for it.

5.4 Boustrophedon Cellular Decomposition

To navigate the robots environment we use a process called Boustrophedon Cellular Decomposition. The process is broken down into two main components, first the robot breaks down the map described above into cells through the use of an exact cellular decomposition. Then each cell is traversed individually in a boustrophedon motion. The result of this is a method of full area coverage being reduced to a simple graph traversal once the robot knows enough about its environment.

5.4.1 Cellular Decomposition

When the robot detects that its previous path planning was deemed incorrect, it begins the cellular decomposition process. This process is composed of 5 steps:

1) We begin by identifying the path traveled by the robot, and filling covered spaces with a '#' symbol.
2) A gradient of the map is created using a 3x3 sobel operator in both the X and Y directions. This gradient is passed to a Harris corner detection algorithm. This algorithm detects edges by identifying regions of the map with a high gradient value in both directions. Here the value of k - our Harris constant - was chosen as 0.05.

$$r = det - k(trace^2) \quad (3)$$

Where our determinant is defined by a 5x5 kernel based on both sobel operators and used to identify the eigenvalues of each region - below is the equation for the determinant.

$$det = (Sxx * Syy) - (Sxy^2) \quad (4)$$

The last step of corner detection is the comparison of the calculated R with our threshold value. All values with R > 12000 are defined as corners.

3) After regions with R are discovered, we traverse them through the use of a clustering algorithm to separate all corners from each other. The clustering algorithm we use is DBSCAN. This algorithm benefits us over other algorithms as no K value is required to identify how many

clusters we plan on creating. The hyperparameters we do use are epsilon which indicates the distance in which we look at adjacent points, and minPts which identifies if a point is a core point, or just an edge point. The DBSCAN algorithm was modified such that instead of returning a list of all points per cluster, we are given a list of all edges denoted by their centroid. This is crucial for the next step

4) After clustering through DBSCAN, we use the centroids to identify cells of our map. We do this by extending the centroids through freespace until an object is encountered. This ended up being broken down by 3 different cases of corners
    a) **Case A**: A corner is on an outward edge - pointing outwards. Here we are able to extend a line in both the + and - directions.
    b) **Case B**: A corner is only open to one direction. Here we extend the line in either a + or a - direction to create a cell boundary. But never both.
    c) **Case C**: A corner is not open in either direction. Here no extension lines are used as this corner does not provide a boundary with any other object.
5) After these lines are drawn, we run our map through a segmentation algorithm. We chose again to use a modified DBSCAN, in this case we make use of block distance as our distance heuristic hyperparameter. This is because we only care to identify edge and core points to those that are directly adjacent with one another. This algorithm returns a list of cells with minimum and maximum values per cell on the left and right indicating boundaries of the cell.

5.4.2 Boustrophedon Path Planning

Boustrophedon path planning is a process used to provide area coverage to our scene. It is defined by the traditional sweeping back and forth motion used by oxen to plow a field - this is the namesake for this algorithm. Each cell discussed in the last step when encountered is fed into the boustrophedon path planning algorithm, we then break down the Y distance into separate path segments and these are passed to the GoTo function to plow the map. The figure to the right is an example of the path planning function in action, here you can see the boustrophedon path that was created, and traveled by our robot

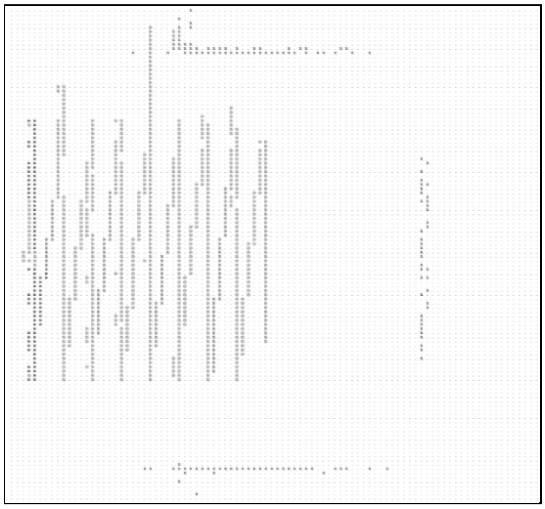


Figure 6: Boustrophedon Path Planning

6. System State Chart

The diagram depicted below provides readers with an explanation - in the simplest of terms - to how our system works as a whole, and how our design reacts and interacts with the various external stimuli. This mainly includes its ability to path find and remove snow from the specified region, as well as the avoidance of collisions with both stationary and dynamic obstacles in its environment.

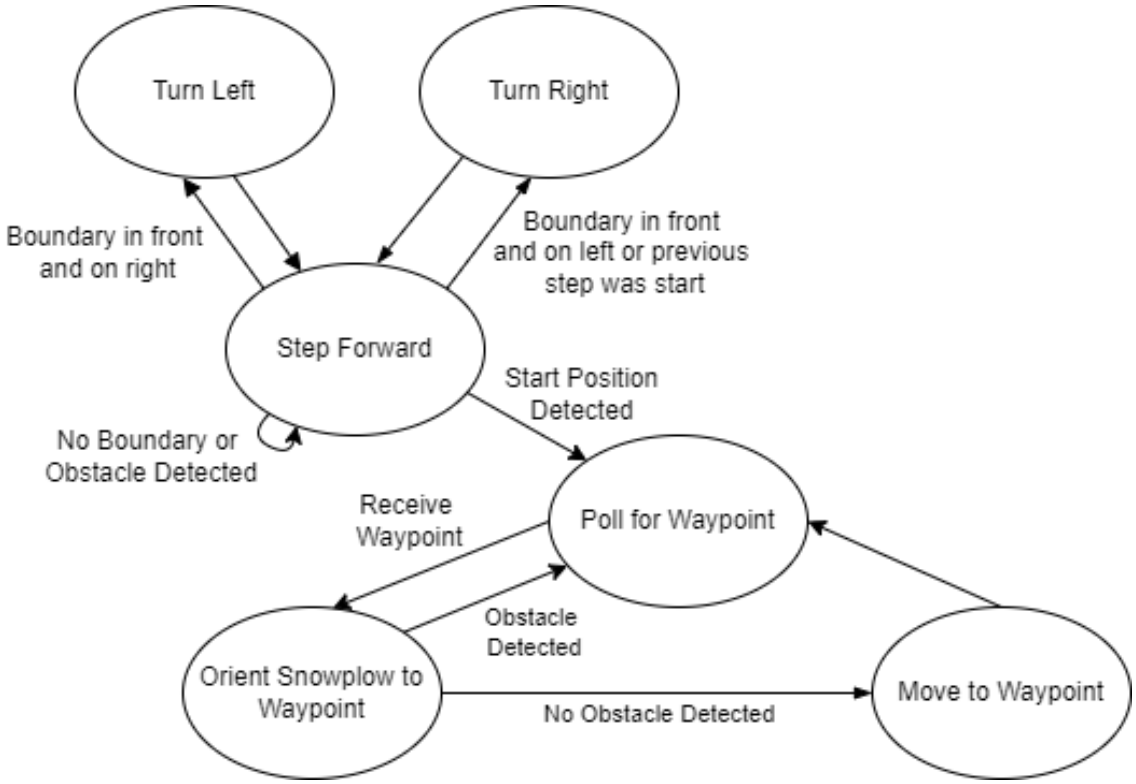


Figure 7: System State Chart Diagram

7. Sequence Diagram

On the following pages are sequence diagrams dictating the communication flow of our system, as well as how various parts will interact and communicate with each other as based on the system state chart shown in figure 5 in the preceding section. Please note that due to the size of the sequence diagrams, larger versions are in Appendix F-H for the reader's sake.

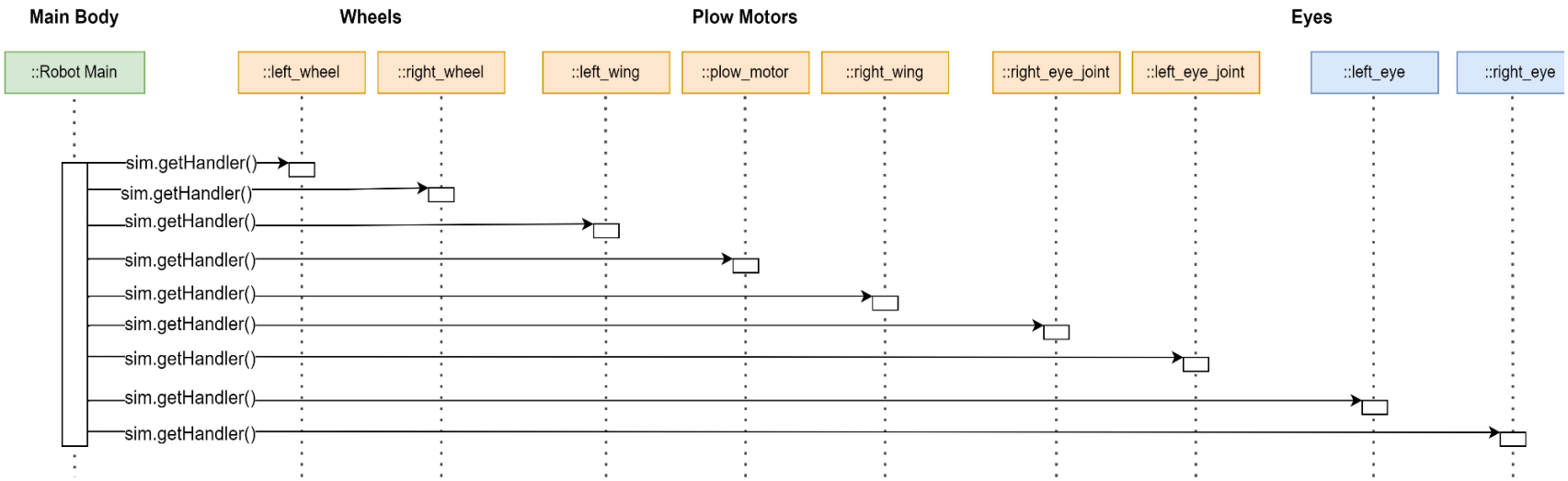


Figure 8: Initial Start Up Sequence Diagram - see appendix F for larger view

Figure 6 above dictates the initial communication setup of the system. The main body activates all main components of the robot. These include the 7 motors, 3 proximity sensors, and 3 vision sensors of our system. Please note that in this and all future diagrams, we follow the convention that the main body script is in green, all actuators are in orange, and sensors are denoted by blue.

On the next page, we will provide the reader with a sequence diagram that dictates the sequence of events that take place when the robot records its environment.

In figure 7 we see the sequence diagram used to dictate the flow required for our robot to map its environment. Continuous calls throughout the robot's operation are required to increment the eye rotation. This process could have been automated - however by having a function perform this - future iterations of the project would be able to focus the eyes on a specific obstacle or area if needed. While this is never implemented it was worth having just as a precaution.

While in rotation, if the eye sensors detect an obstacle, we return to the main environment where we record the distance, and subsequently the obstacle onto our map.

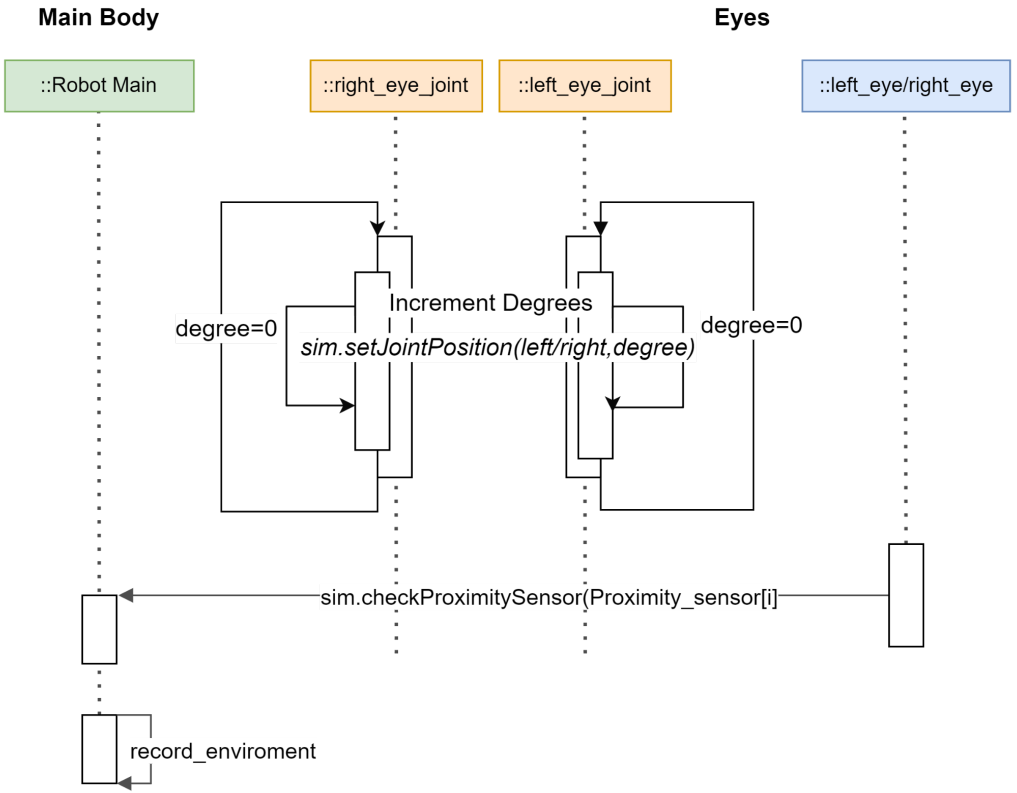


Figure 9: Sequence Diagram - Environmental Mapping

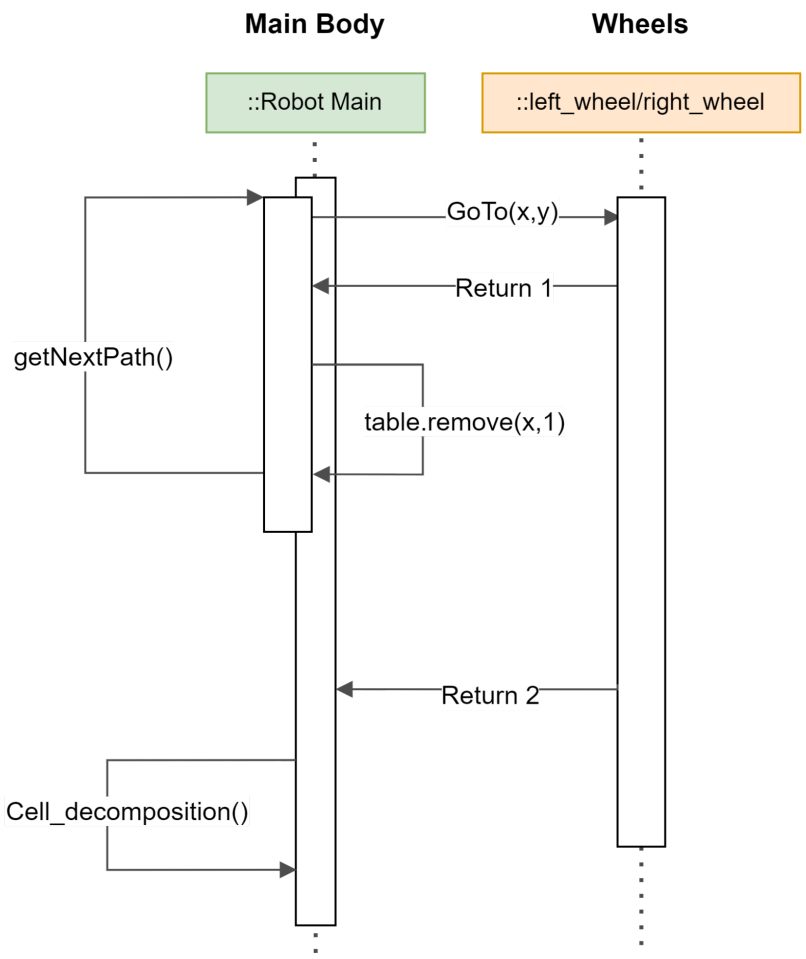


Figure 10: Path finding sequence

To the left in figure 8, we define the path finding sequence of our model. Not shown in this diagram, the sequence begins by the model running through the first iteration of the cell_decomposition() to get the first cell - which ends up being the entire map. It creates a boustrophedon, and decomposes it into all its appropriate pathing elements.

The pathing elements are gone through one by one. While the robot is moving, if suddenly it comes across a piece of environment that it for whatever reason is not able to get through as previously thought, it stops and recalculates the pathing with its newly found knowledge of obstacles present in the environment. This separate sequence is started when goTo returns 2.

If goTo returns 1, we remove from the stack our first element and get rid of it. We then select the next element from our path and set it as our path. The robot re-orients itself in the direction of the new path and travels to the end point.

Once all pathing elements are gone through for that specific cell, we identify the next best cell that we need to go through and move the contents of its area traversal to the path element being gone through.

8. Project Budget

The following section explores the budget we have laid out for our project, including the time we have taken to design the product, as well as the cost associated with acquisition of the various necessary components and their vendors.

8.1 Planned Value

A proximity sensor will be used for the detection of both moving and static obstacles. For this, the group has chose the following requirements for the sensor:

- Detect object from a distance of a meter
- Detect both nonmetallic and metallic objects
- Wide beam angle
- Cost Effective

Due to this the team is using the CUSA-R75-18-2400-TH. This ultrasonic proximity sensor has a price of $5.08 per unit and satisfies the above requirements. The sensor has a sensing distance from 20 centimeters to 18 meters. As well the sensor has a ebam angle of 75 degrees. Infrared sensors will be used for the line detection that is boundary sensing. The Sparkfun QRE1113 is a widely used infrared sensor within line following robots. The infrared sensor is $4.07 per unit. This sensor has a sensing distance of 1mm therefore will be placed at the bottom of the robot.

The motor the team had initially selected for the snow removal robot is the Sparkfun ROB-09238. This motor is widely used in mobile robots. As well this motor is $24.70 per unit. This motor was selected due to its sizing and availability. However re-examining the torque requirements this option was severely underpowered for the force needed to move the robot and push snow so the part 80807015 by Crouzet was chosen instead as it has a maximum rated torque that matches the torque limit used in coppelia sim. The ROB-09238 will be repurposed to control the angle of the ultrasonic proximity sensors.

8.2 Budget at Completion

Given the pricing of the above components the budget at completion will be

Table 9 : Budget Breakdown

Item	Price (CAD)	Quantity	Provider
Robot Body	$3500	1	Protocase
ROB-09238	$24.70	2	SparkFun
80807015	$313.10	2	Crouzet

CUSA-R75-18-2400-TH	$10.16	4	CUI Devices
QRE1113	$4.07	4	Sparkfun
nRF52833-MS88SF31	$14.00	2	Minew
Total		$4260.5	

8.3 Cost Management Chart

Each group member planned to work on the project five hours a week. The labor rate is $30 Canadian dollar per hour totaling to $600 CAD each week for ten weeks . Based on the work schedule, we are on time however we completed some of the activities less than the budgeted hours.

Table 10: Labor Cost Breakdown

Name	Labor Rate (CAD)/Hour	Hours/Week
Aedyn Ladd	$30	5
David Casciano	$30	5
Daniel Tura	$30	5
Tadhg McDonald-Jensen	$30	5

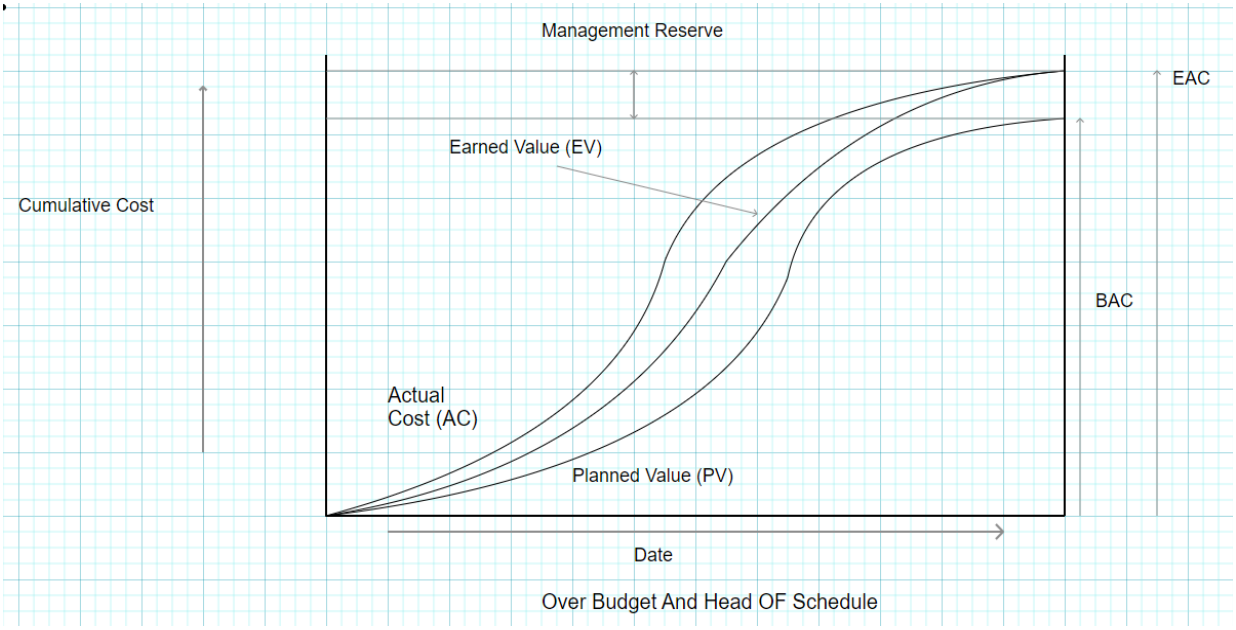


Figure 11: Cost Management Chart - see appendix I for larger view

9. Control Charts

To ensure that the requirements are consistently met, control tests were performed. Given the set of requirements outlined in Section 2.1, the team polled for key metrics.

Firstly, the team wanted to ensure that requirements A1 and A2 were met. Requirement A1 states, the size of the robot while idle will be constrained to a space of 0.5 x 0.8 x 1 meters. Five separate measurements were taken to test the dimensions, the results are shown in Figures X-X.

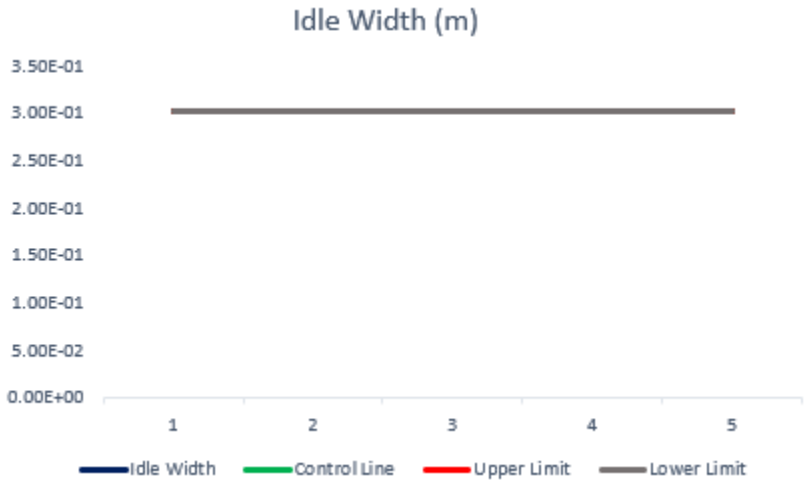


Figure 12: Control Chart for Idle Width

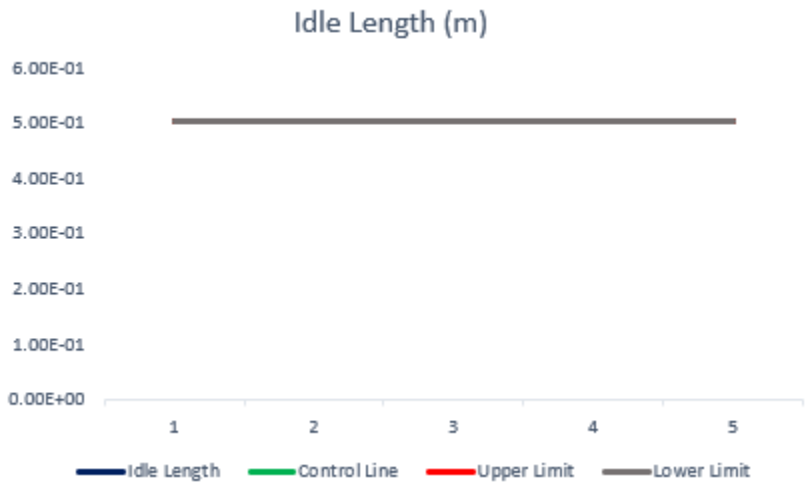


Figure 13: Control Chart for Idle Length

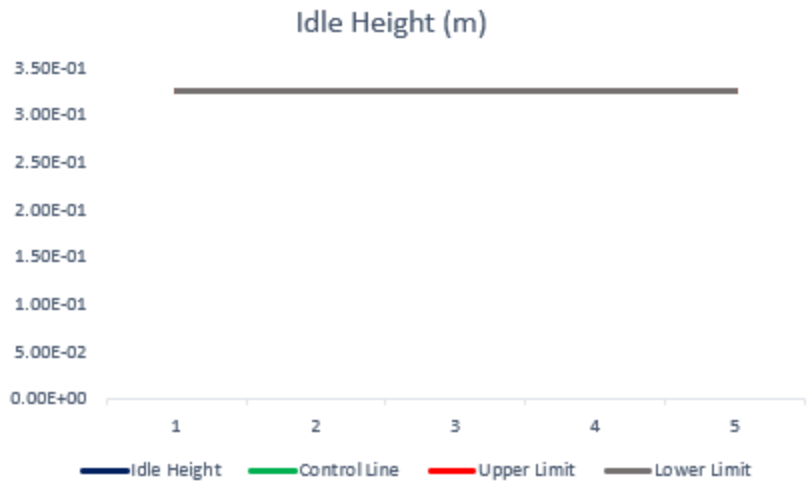


Figure 14: Control Chart for Idle Height

As seen above, the dimensions of the idle robot are static; the measurements of the robot remain at 30 x 50 x 32.5 cm. The upper and lower limits of each measurement do not exceed the requirements, therefore the control tests pass.

Secondly, Requirement A2 states that the robot should be no more than 1 x 0.8 x 1 meters. To test this the dimensions of the robot were taken during operation, while the plow was fully articulated. Again, five measurements were taken to test the dimensions, the results are shown in Figures 15.

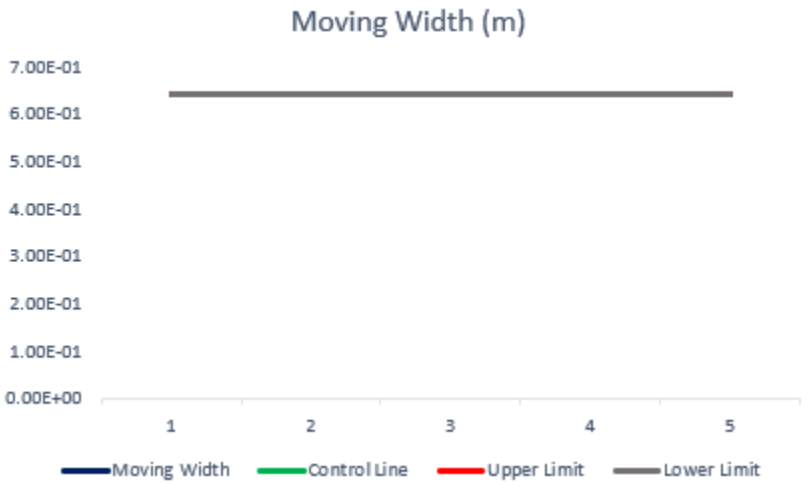


Figure 15: Control Chart for Moving Width

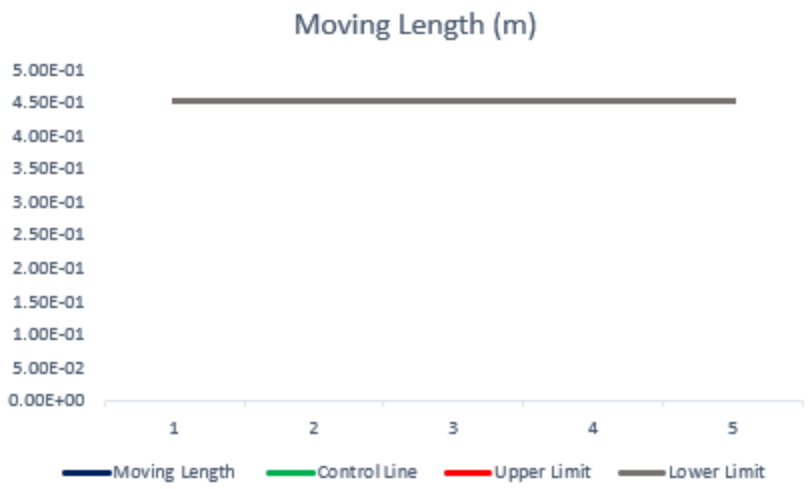


Figure 16: Control Chart for Moving Length

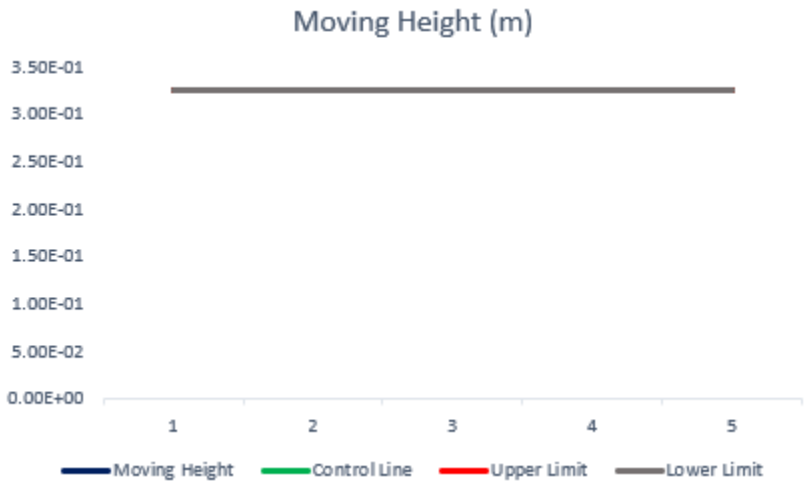


Figure 17: Control Chart for Moving Height

As seen above, the dimensions of the moving robot remain static; the measurements of the robot remain at 64 x 45 x 32.5 cm. The upper and lower limits of each measurement do not exceed the requirements, therefore the control tests pass.

According to Requirement C, the robot should not travel at a pace faster than 2 m/s. To test that the requirement is met consistently, the team polled the velocity of the robot in five scenarios and took the maximum recorded velocity. The maximum velocys are shown in Figure 18.

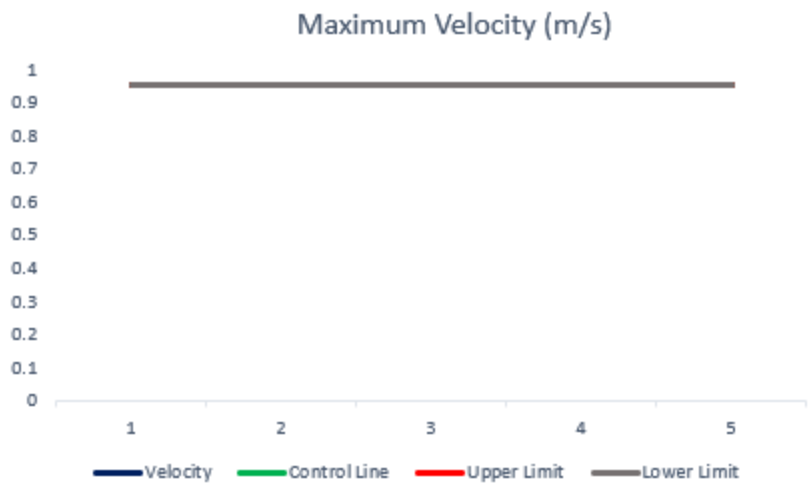


Figure 18: Control Chart for Maximum Velocity

Given the observations above, the upper and lower limits of the maximum velocities do not exceed 1 m/s which is well below the two meters per second measurement. Due to this the robot passes the maximum velocities control tests.

Requirement D states that the maximum torque should not exceed the torque specified by the datasheet of the sourced parts. According to the sourced motors datasheet, the motor has a maximum torque of two newton meters. To test that the motor does not exert more torque than the maximum specified, we polled the torque throughout five scenarios and took the maximum observed. The results are displayed in Figure 19.

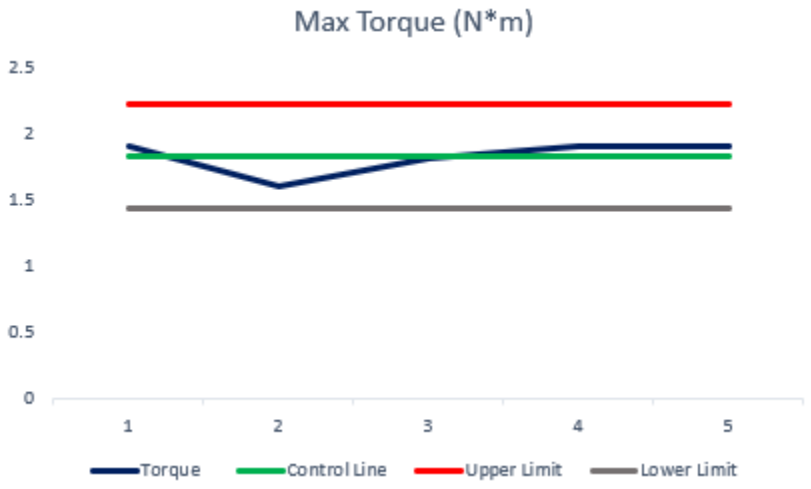


Figure 19: Control Chart for Maximum Torque

Given that the control line does not go above two newton meters of torque, the motors pass the control tests.

Lastly, Requirement OF states that the robot should be able to avoid obstacles. To test this the robot was tested in different environments and the success rate of the collision avoidance was recorded. The results are displayed in Figure 20.

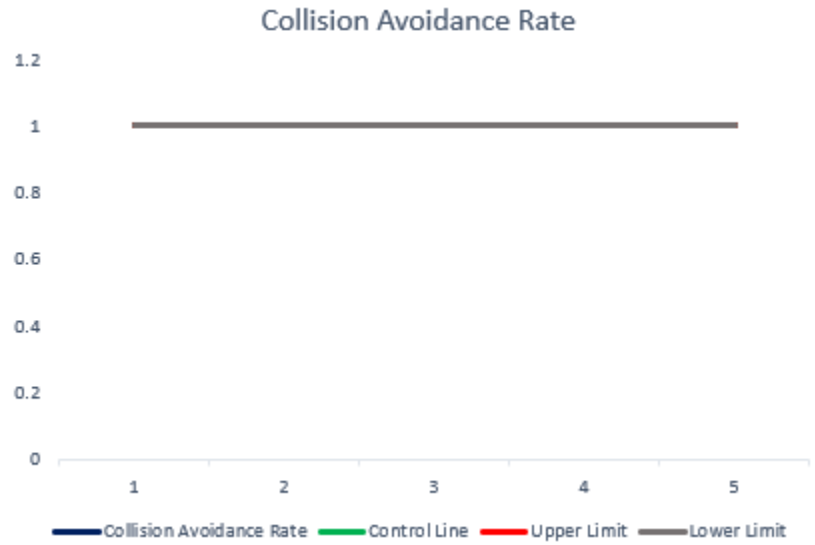


Figure 20: Control Chart for Collision Avoidance Rate

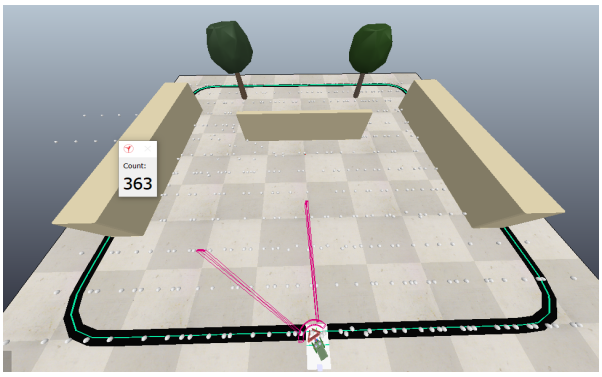
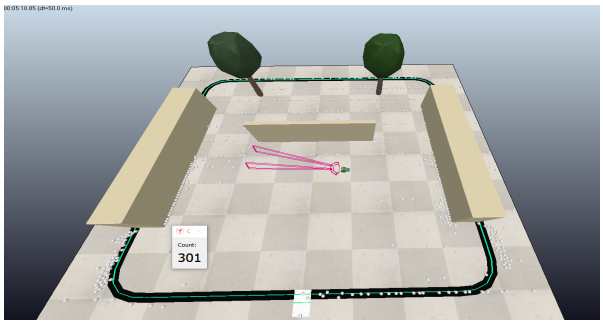
The collision avoidance rate is shown as a ratio; as the lower limit did not go below 1 or 100 percent, the robot passed the control tests.

10. Testing and Results

In order to ensure that the robot functions correctly and meets all the requirements listed in section 2.1 of this document, we will run a variety of tests on the system. If there are any failures on insufficiencies, they will be noted in the observations. Below are the results of testing on all 4 maps simulated for 5 minutes.

10.1 Map 1 Testing and Observations

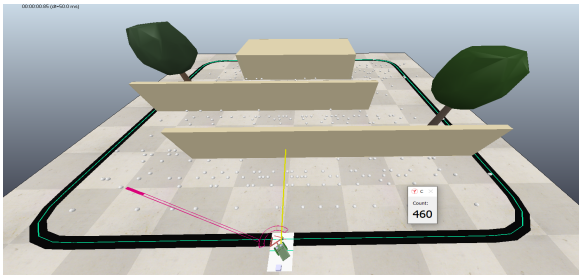
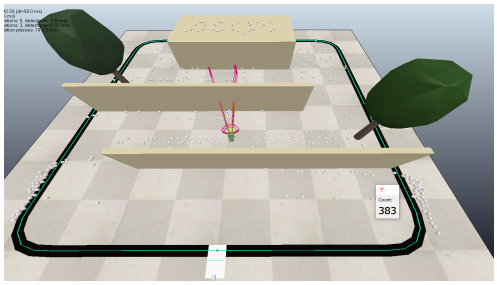
Test Map 1 has 363 snowballs at the beginning of the simulation. After five minutes of the simulation run the robot cleaned 17.07% of snowballs reducing the total count of snowballs to 301. The robot moved more than 95% of the snowballs up to the beginning of the two walls on the left and right of the map. However, once the robot gets into the middle of the two walls the snow is pushed against the two walls as supposed to be outside the map.

Map: Map 1	
Start	Five minutes elapsed
 Figure 21: Map 1 at start position	 Figure 22: Map 1 After 5 minutes
Results: After five minutes elapsed 62 pieces of snow were removed from the boundaries. That is 17 percent of the snow.	
Observations: Two observations were made when running the simulation. The first observation is that there is not an effective way of clearing snow when there is a wall parallel to the boundaries on the y axis. Secondly, the robot struggles to clear the whole area because it travels at 1 m/s.	

10.2 Map 2 Testing and Observations

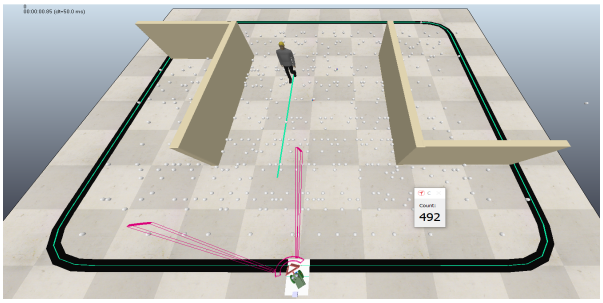
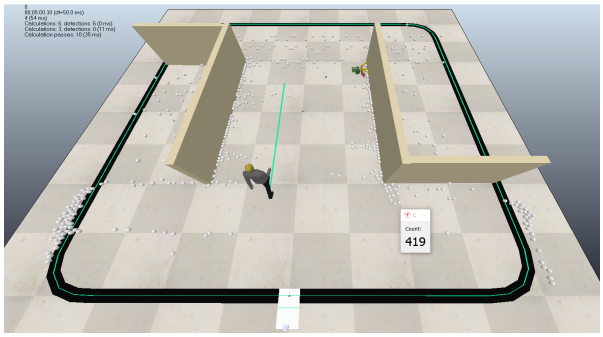
Test Map 2 has 460 snowballs at the beginning of the simulation. After five minutes of a simulation run the robot cleaned 17.25% of snow balls reducing the total count of snow balls to 383.

Our implementation of the Boustrophedon algorithm did well cleaning the snow up to the first horizontal wall in front of the lily pad(base). After getting into the middle of the two walls which we see in the figure above, the robot started moving up and down (vertically).

Map: Map 2	
Start	Five minutes elapsed
 Figure 23: Map 2 Starting Snapshot	 Figure 24: Map 2 After 5 minutes
Results: After five minutes elapsed 77 pieces of snow were removed from the boundaries. That is 17 percent of the snow.	
Observations: Our implementation of the Boustrophedon algorithm did well cleaning the snow up to the first horizontal wall in front of the lily pad(base). After getting into the middle of the two walls which we see in the figure above, the robot started moving up and down (vertically).	

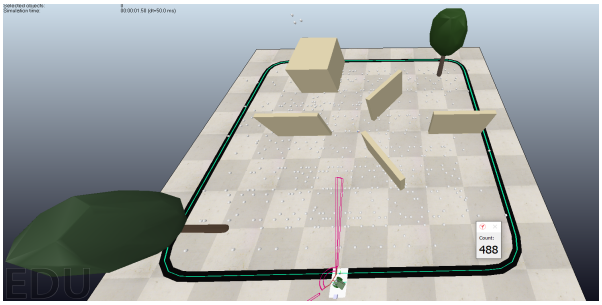
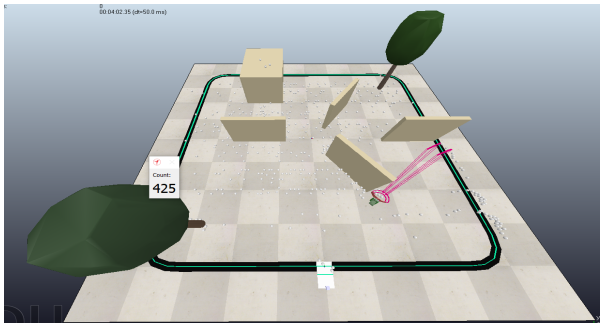
10.3 Map 3 Testing and Observations

Testing Map 3 started with a total of 492 snowballs, after five minutes there were 419 snowballs remaining. The robot removed 15 percent of the snow while traversing the majority of the map height-wise. All test results are added to the appendix if results below are not clear enough.

Map: Map 3	
Start	Five minutes elapsed
 Figure 25: Map 3 at start position	 Figure 26: Map 3 After 5 minutes
Results: After five minutes elapsed 73 pieces of snow were removed from the boundaries. That is 15 percent of the snow.	
Observations: As seen in Map 1 also, the vertical walls cause the robot not to clear the snow out of the boundaries. The Boustrophedon algorithm did not path around the moving person, therefore we had to rely on the object avoidance sensors.	

10.4 Map 4 Testing and Observations

Testing Map 4 initially had 488 pieces of snow within the boundary, after five minutes 415 pieces remained. The robot cleared 13 percent of the snow while traversing roughly a quarter of the map.

Map: Map 4	
Start	Five minutes elapsed
 Figure 27: Map 4 at start position	 Figure 28: Map 4 After 5 minutes

Results: After five minutes elapsed 63 pieces of snow were removed from the boundaries. That is 13 percent of the snow.
Observations: The robot effectively navigated the diagonal walls but left behind many particles of snow. The Boustrophedon algorithm effectively breaks the maps into cells which helps navigate through much of the map.

10.5 Testing conclusions

The path finding algorithm does not perform well in structures that are placed along the right/left side of the track. Since it takes a polygonal pattern moving the snowball to a side, the wall will block the robot from taking the snow out of the map boundary. The robot cleaned a total of 17.07% of snowballs in five minutes.

11. Final Thoughts

At the end of this project we have designed a robot with a nominal cost of $4260.5 capable of clearing on average 15.5% of an 144m^2 area of snow within the specified 5 minute time period. The software developed is very effective at subdividing irregular shaped areas and clearing them methodically, and while it does not identify moving obstacles particularly well that could be improved with more development time and was an identified limitation from the beginning of development.

11.1 Development process

We have managed to mostly stay on track with planned value although the specific task breakdown did slightly shift as the project went on. As well our original budget for the robot body was largely underestimated so the cost of the final product was much higher than planned.

11.2 Challenges faced

There were several technical hurdles that were overcome, all group members were unfamiliar with the Lua programming language at the beginning of the project and a decent amount of effort was spent learning the language and figuring out how to set up coppelia sim in a way to use lua scripts that could be easily version controlled.

The robot can comfortably move at 2m/s as in the original specification, at higher speeds the inertia of the robot when stopping or changing directions causes issues with balance. If a higher speed was desired in the future the software would likely need to implement more gradual acceleration when changing directions.

11.3 Future Development

The percentage of snow cleared is rather abysmal. 15% is nowhere near what we intended, however it is worth pointing out two key features as to why this is.

Firstly, the model we developed is primarily focused on area coverage. And when observing our results it will be noted that the robot does in fact cover a large area of the map within the allotted time. The reasoning behind only 15% of snow being removed is that in covering the area, the robot pushes a large amount of snow towards obstacles and leaves them there. This does in fact achieve the removal of snow from a majority of the area as would be expected from normal shoveling by a human, however results in calculations do not take this into consideration.

Secondly, a feature that has not yet been implemented, but would greatly benefit our model if given extra time would be a memory. By this we mean that after the robot completes a full coverage of the area in however long it takes, if we were to record this map we would subsequently be recording exactly where obstacles are and even the pathing that was taken. This could then be optimized thus leading to

the second, and all subsequent clearings of maps being almost perfect in their pathing - something which could not be proven with randomized models.

In considering the results of our model, it is important to both consider the timeframe in which our model was developed as being rather short, the expectations being rather large, and our analysis of the problem at hand and use of engineering principles to guide our design - being rather thorough and well thought out.

12. Bibliography

Auger, Nathalie, et al. “Association between Quantity and Duration of Snowfall and Risk of Myocardial Infarction.” CMAJ : Canadian Medical Association Journal = Journal De L'Association Medicale Canadienne, Joule Inc., 13 Feb. 2017, https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5305403/.

Canada, Environment and Climate Change. “Government of Canada.” Canada.ca, / Gouvernement Du Canada, 30 Sept. 2020, https://www.canada.ca/en/environment-climate-change/services/environmental-indicators/snow-cover.html.

Tyagi, D. (2020, April 7). Introduction to harris corner detector. Medium. Retrieved April 11, 2022, from https://medium.com/data-breach/introduction-to-harris-corner-detector-32a88850b3f6

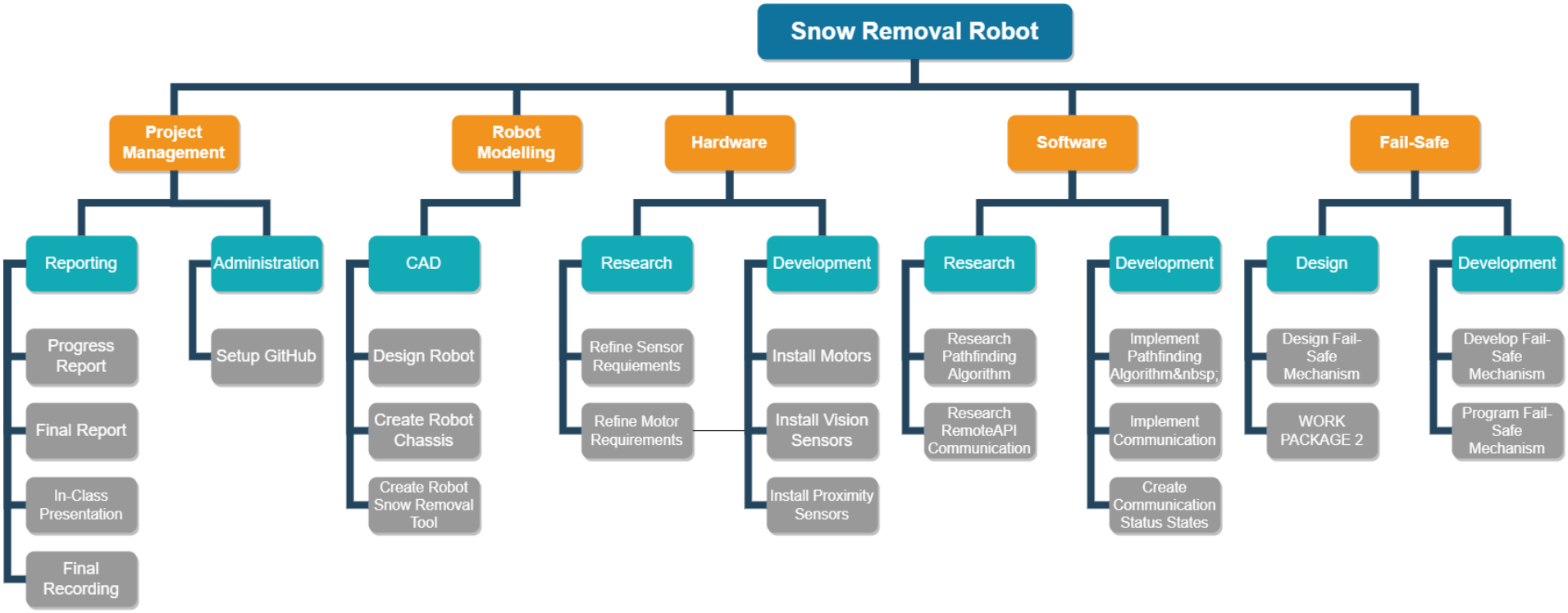
DBSCAN clustering algorithm in machine learning. KDnuggets. (n.d.). Retrieved April 11, 2022, from https://www.kdnuggets.com/2020/04/dbscan-clustering-algorithm-machine-learning.html

Maxim Likhachev , Dave Ferguson , Geoff Gordon† , Anthony Stentz , and Sebastian Thrun. (n.d.). Anytime dynamic A*: An anytime, replanning algorithm. Retrieved April 11, 2022, from https://www.cs.cmu.edu/~ggordon/likhachev-etal.anytime-dstar.pdf

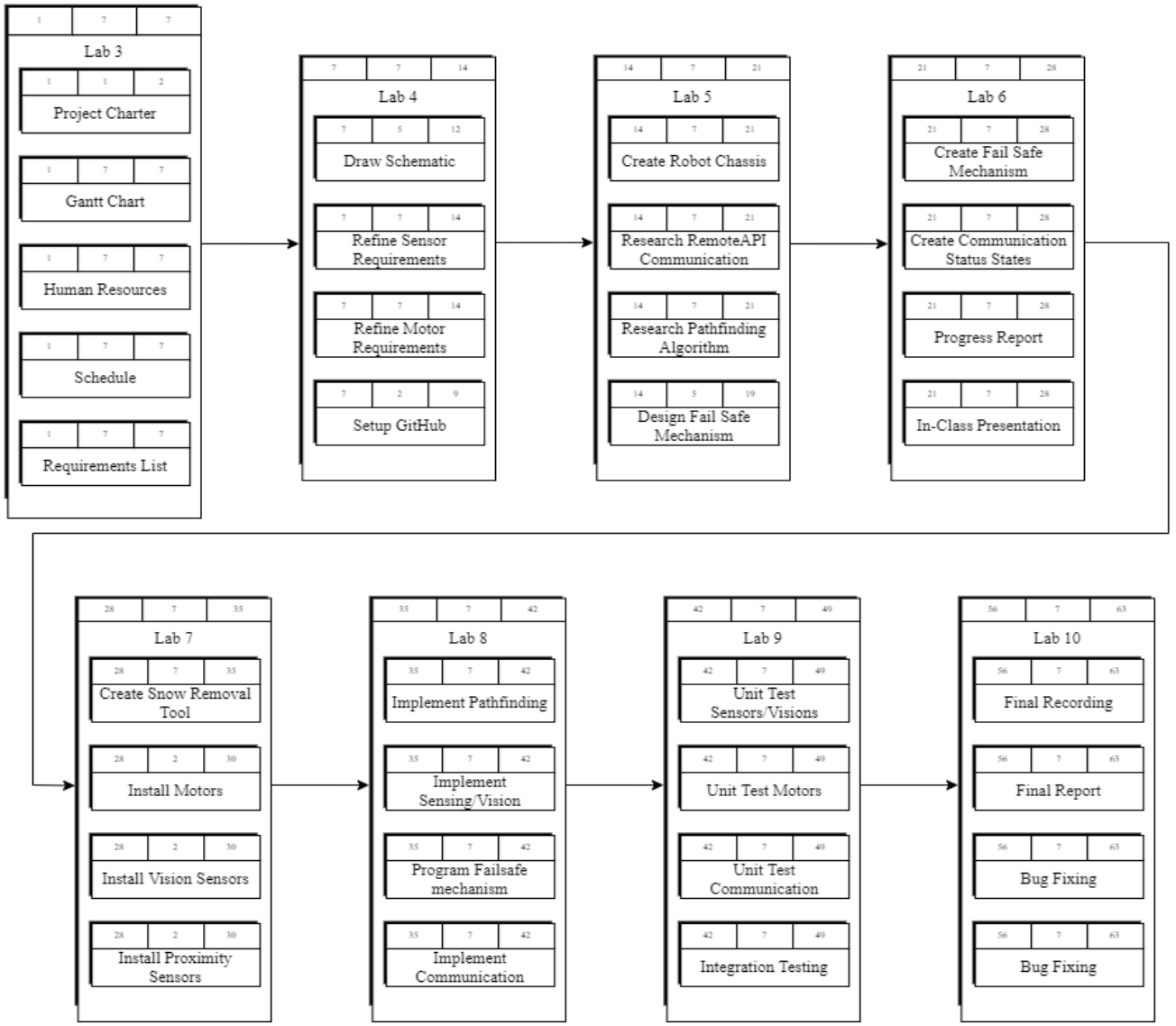
Howie Choset. (n.d.). Robotic Motion Planning: Cell decompositions. Retrieved April 11, 2022, from https://cs.cmu.edu/~motionplanning/lecture/Chap6-CellDecomp_howie.pdf

APPENDICES

APPENDIX A: Work Breakdown Structure

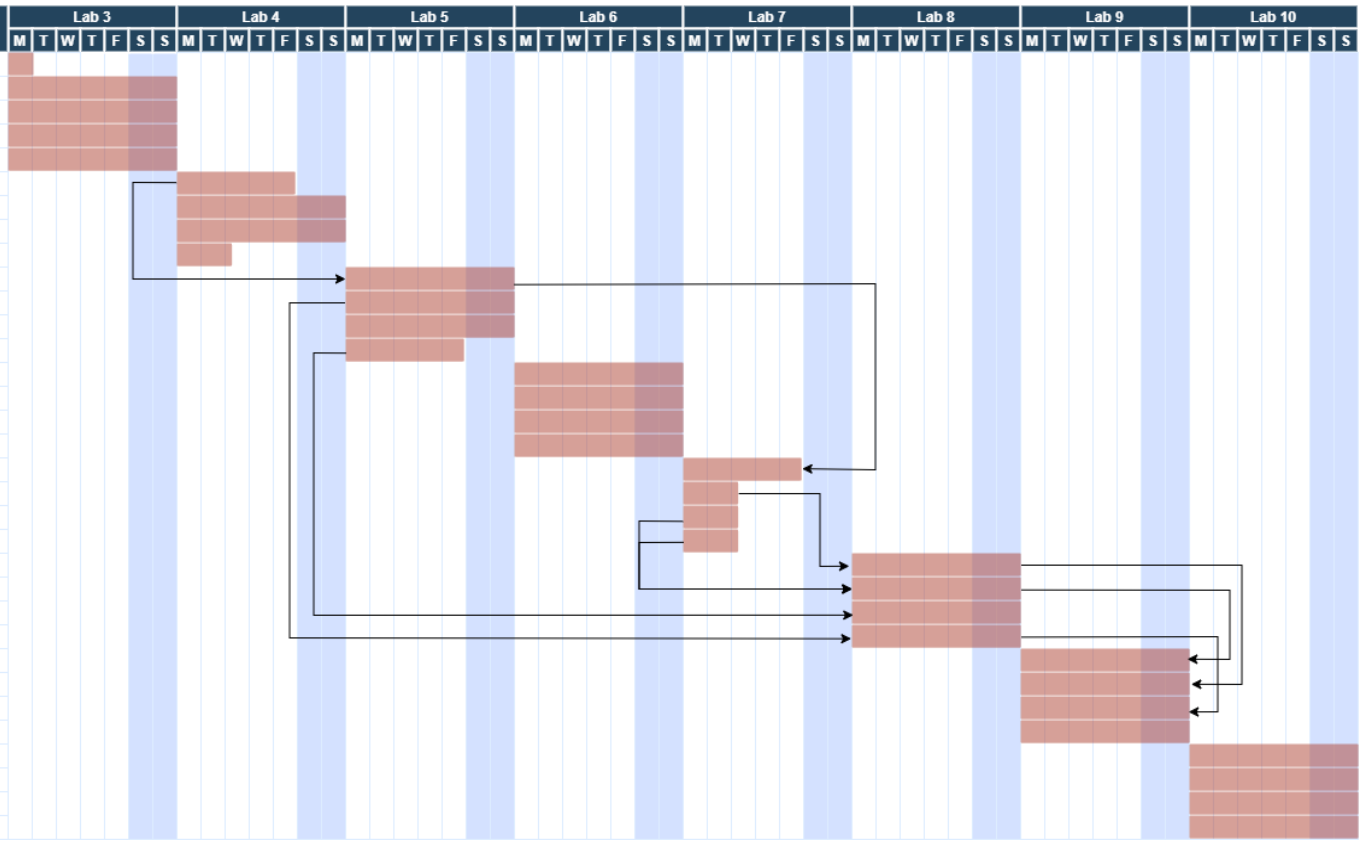


APPENDIX B: Schedule Diagram

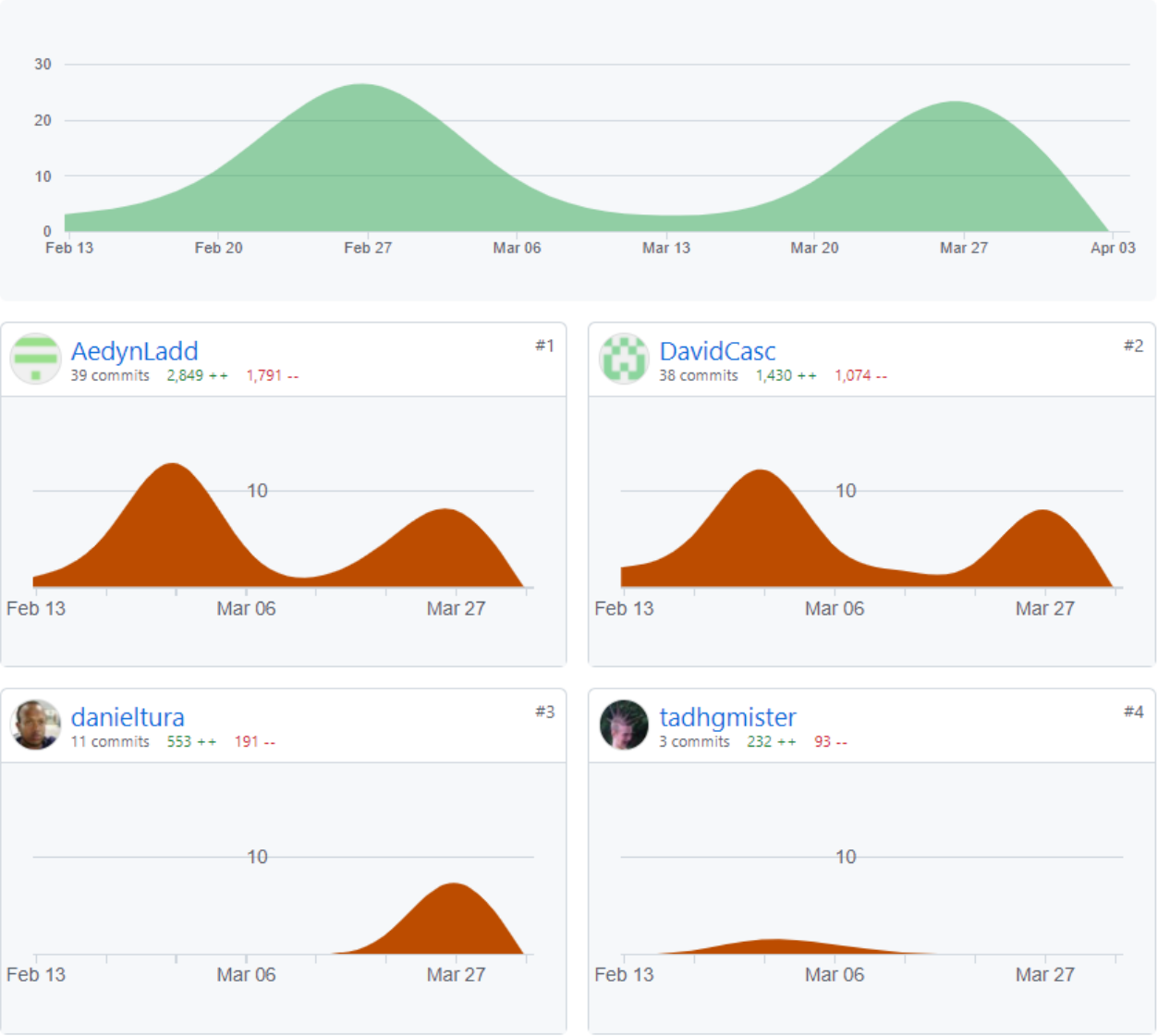


APPENDIX C: Gantt Chart

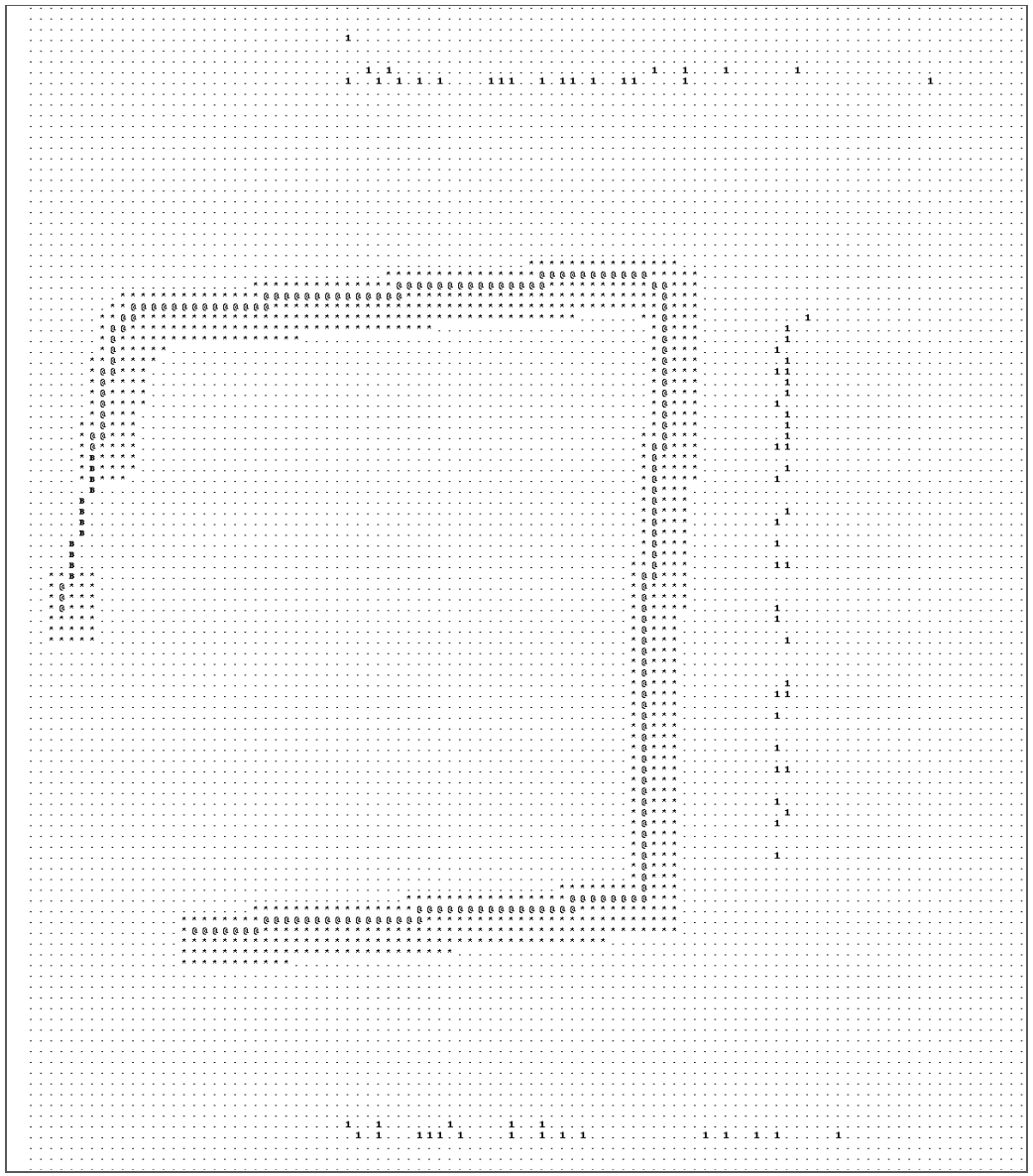
	Task Name	Duration
1	Project Charter	1 days
2	Gantt Chart	7 days
3	Human Resources Section	7 days
4	Schedule Section	7 days
5	Requirements List	7 days
6	Draw Schematic	5 days
7	Refine Sensor Requirements	7 days
8	Refine Motor Requirements	7 days
9	Setup GitHub	2 days
10	Create Robot Chasis	7 days
11	Research RemoteAPI Communication	7 days
12	Research Path Finding Algorithm	7 days
13	Design Fail Safe Mechanism	5 days
14	Create Fail Safe mechanism	7 days
15	Create Communication Status States	7 days
16	Progress Report	7 days
17	In-Class Presentation	7 days
18	Create Robot Snow Removal Tool	5 days
19	Install Motors	2 days
20	Install Vision Sensors	2 days
21	Install Proximity Sensors	2 days
22	Implement Path Finding	7 days
23	Implement Sensing/Vision	7 days
24	Program Failsafe Mechanism	7 days
25	Implement Communication	7 days
26	Unit Test Sensors/Vision	7 days
27	Unit Test Motors	7 days
28	Unit Test Communication	7 days
29	Integration Testing	7 days
30	Final Recording	7 days
31	Final Report	7 days
32	Bug Fixing	7 days
33	Bug Fixing	7 days



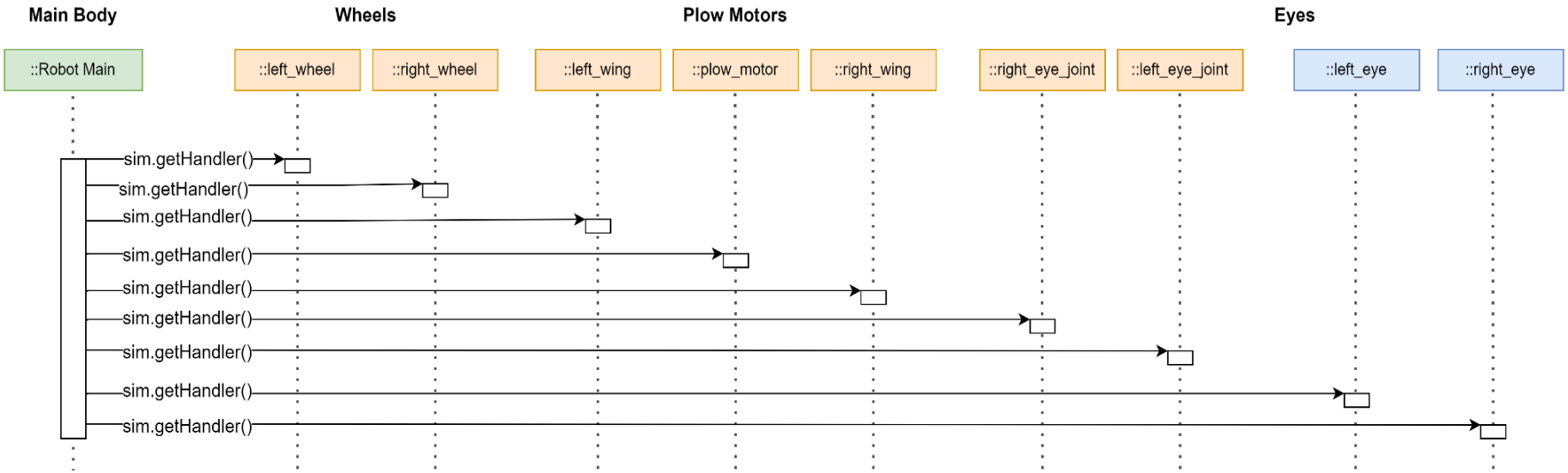
APPENDIX D: Code Contribution



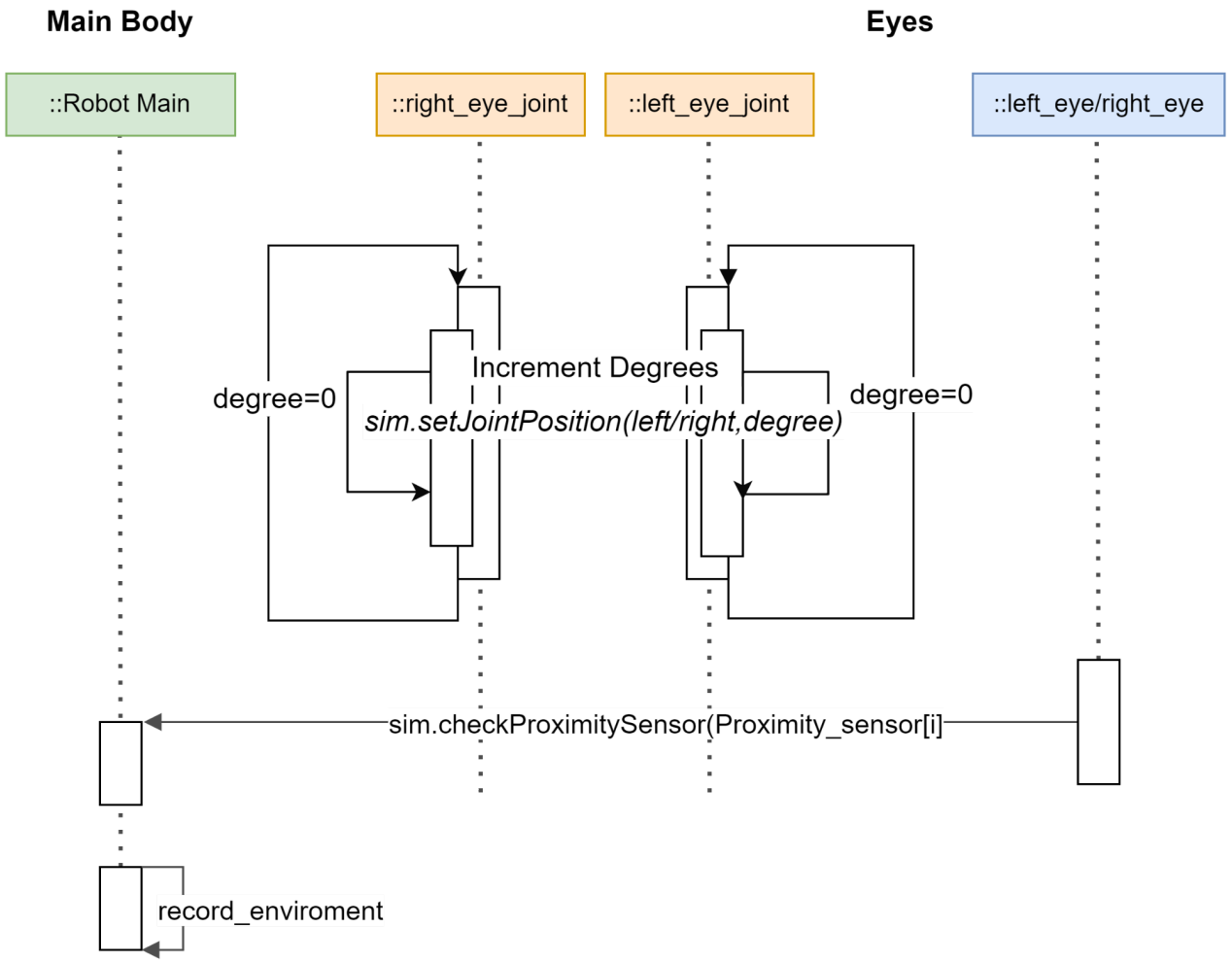
APPENDIX E: Object Detection Map



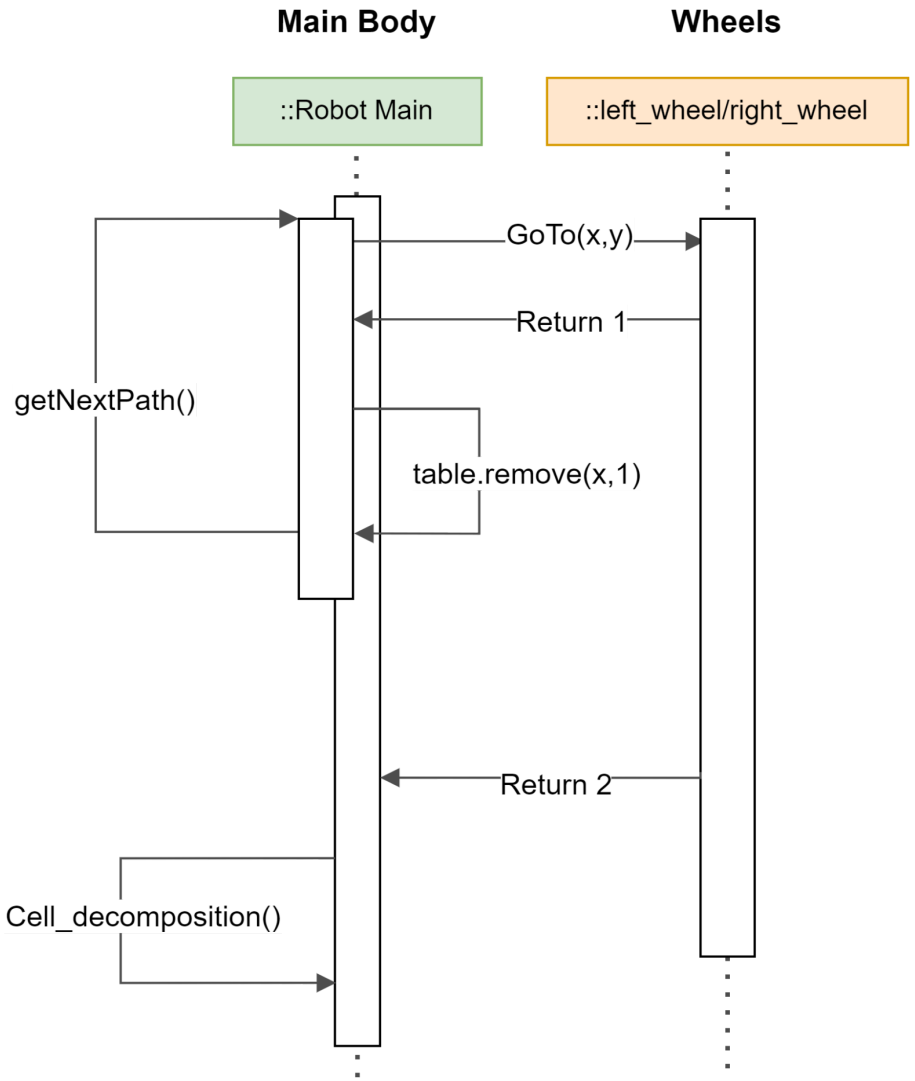
APPENDIX F: Initial Start Up Sequence Diagram



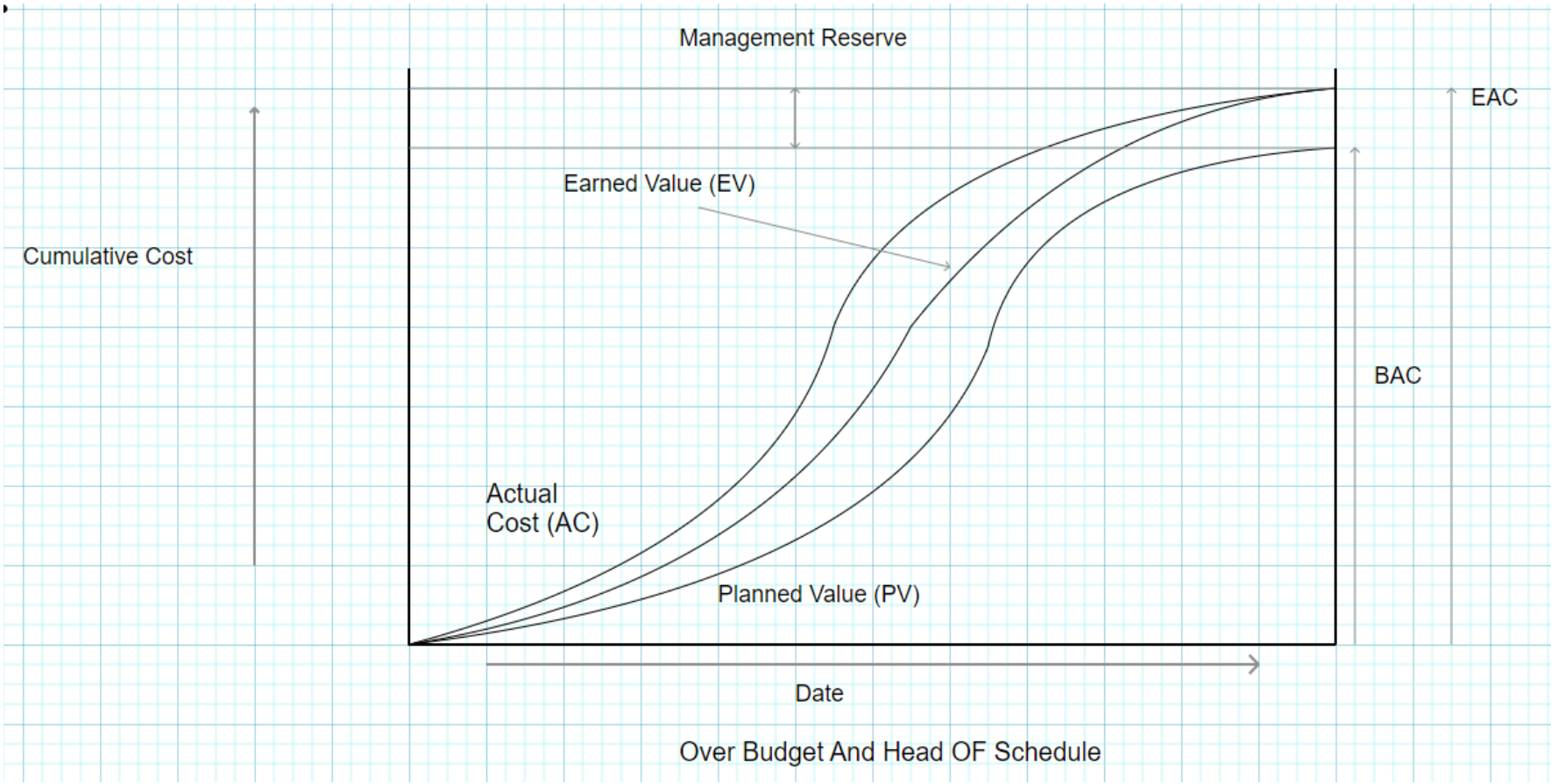
APPENDIX G: Environment Sensing Sequence Diagram



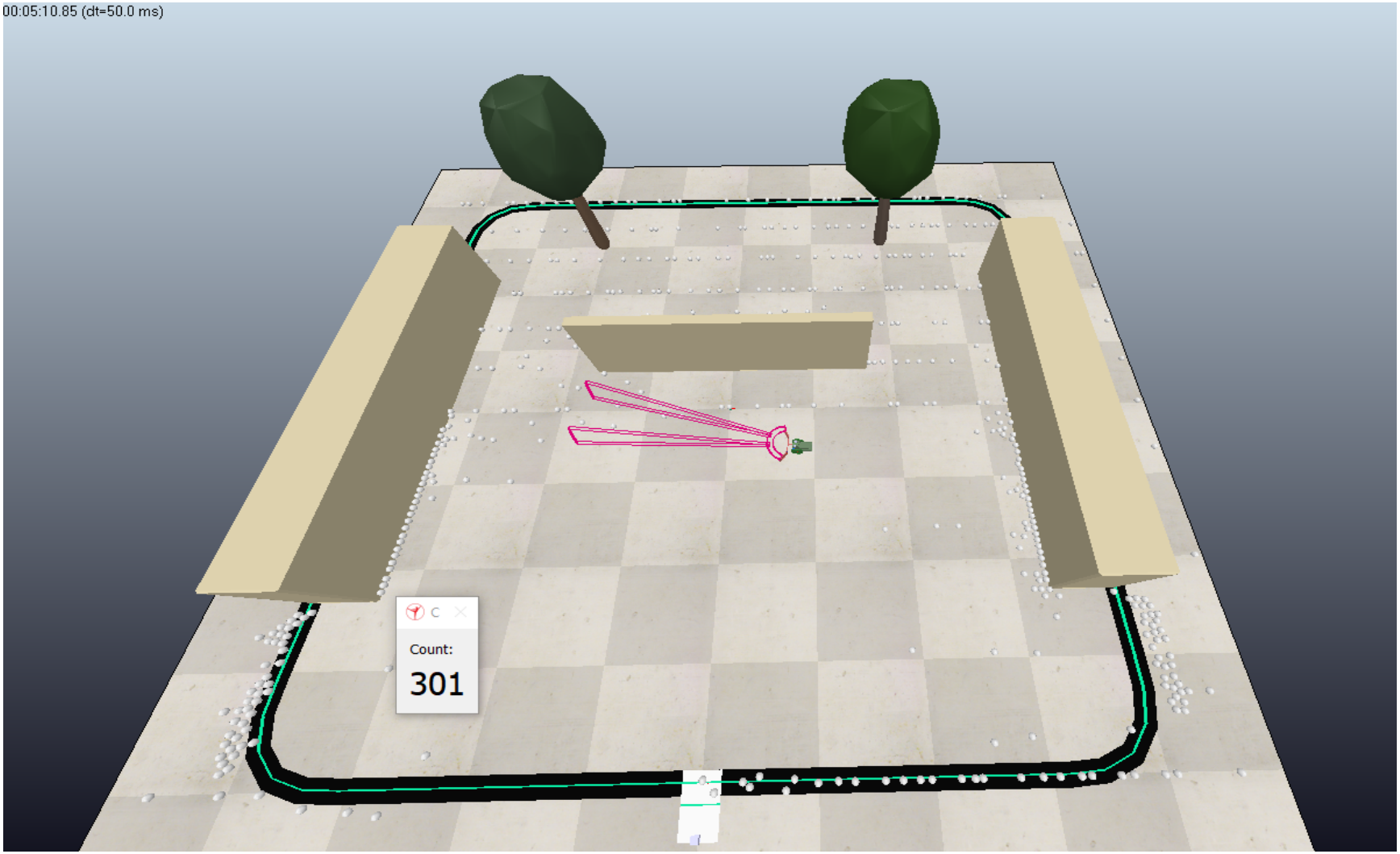
APPENDIX H: Path Finding State Machine - Sequence diagram

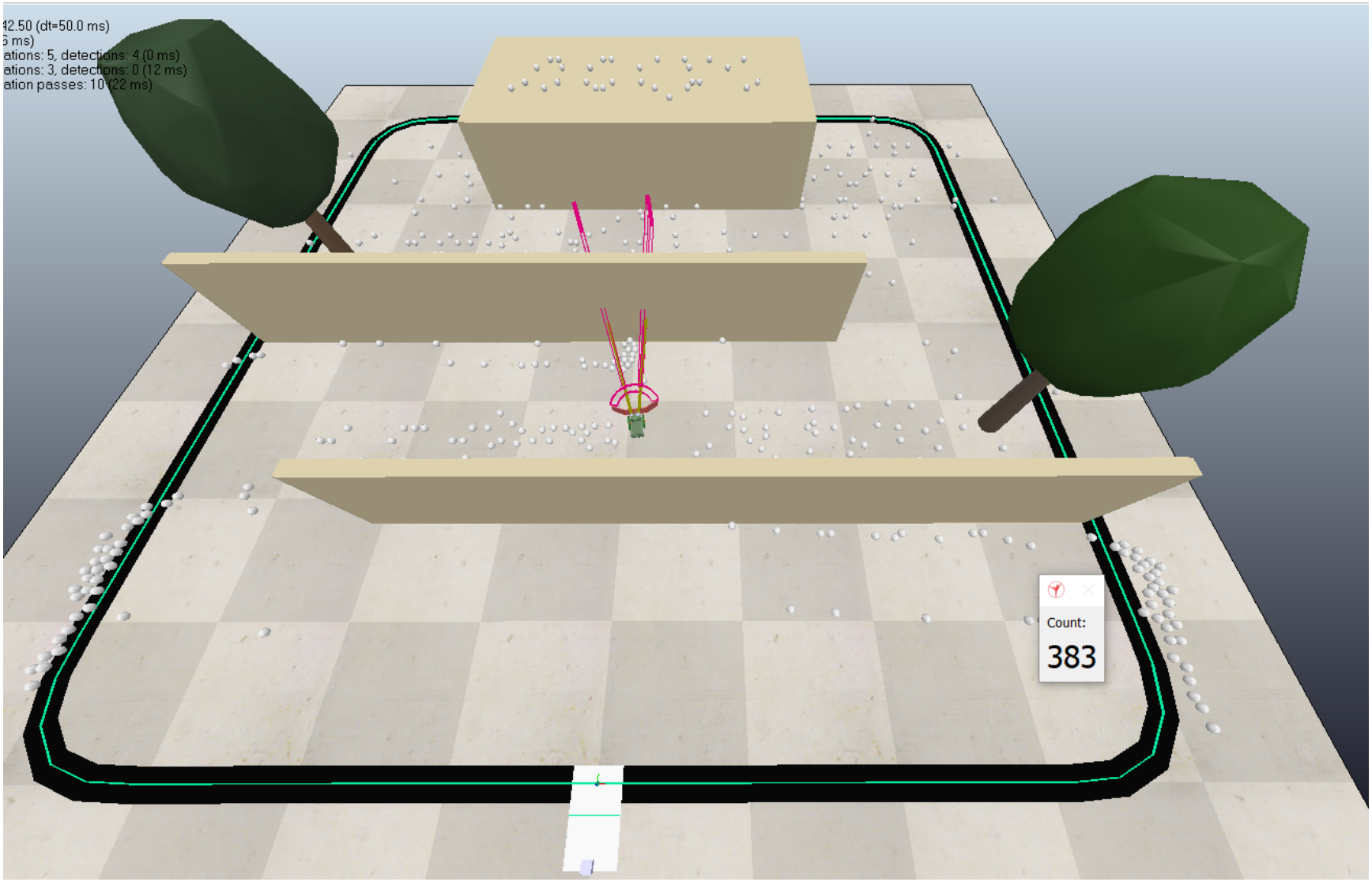


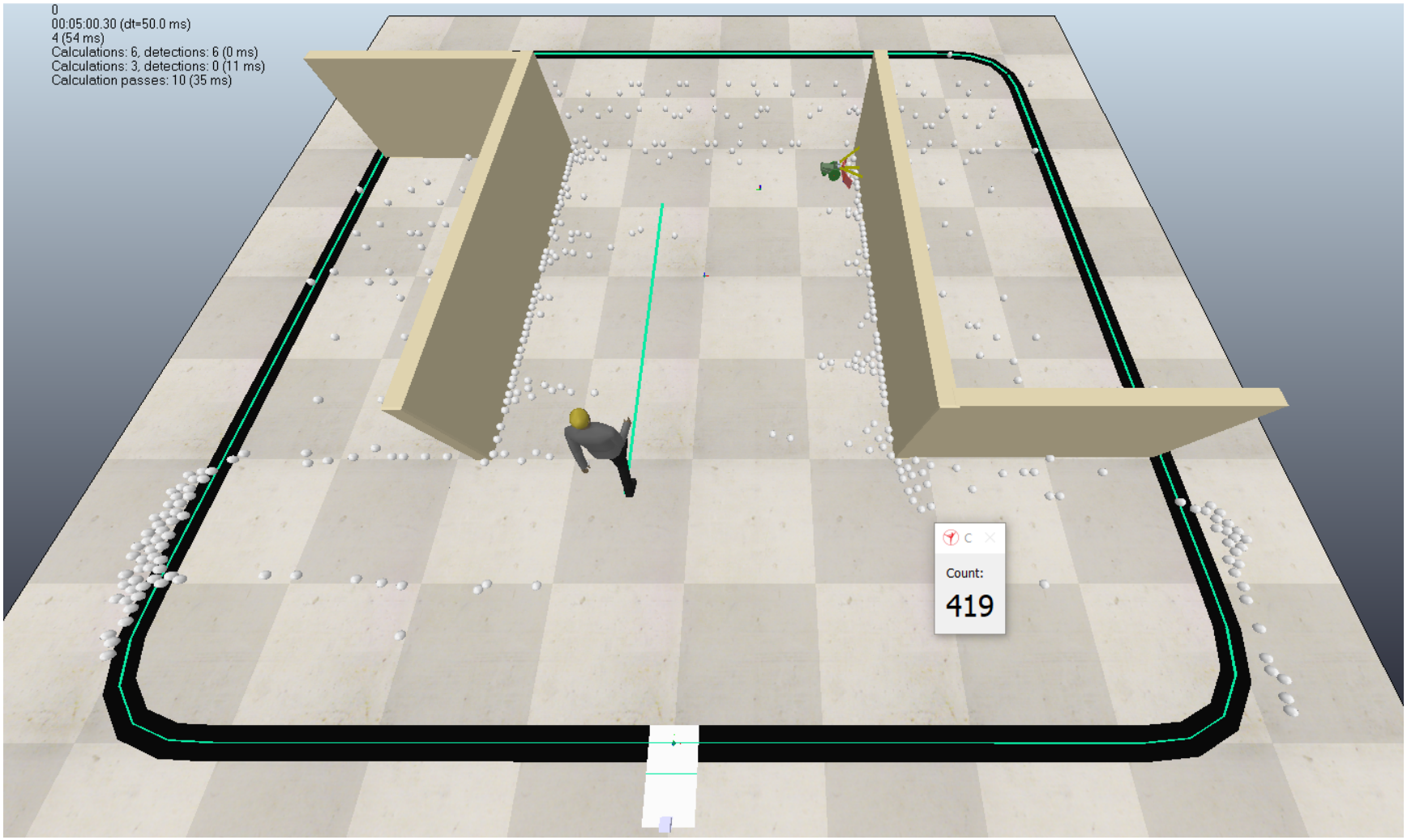
APPENDIX I: Cost Management Chart

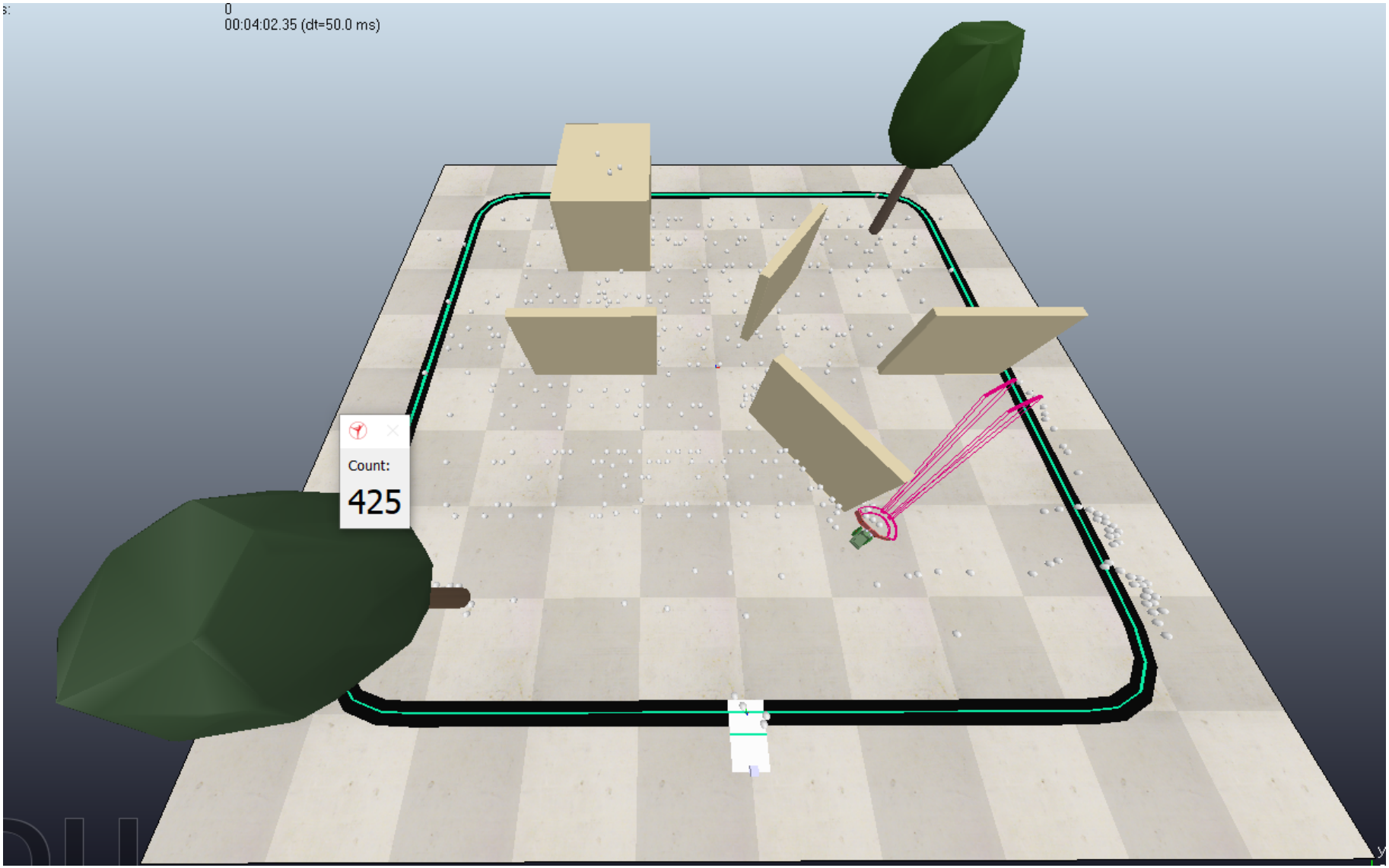


APPENDIX H: Test Results After Five Minutes Of Simulations




42.50 (dt=50.0 ms)
5 ms)
ations: 5, detections: 4 (0 ms)
ations: 3, detections: 0 (12 ms)
ation passes: 10 (22 ms)
Count:
383


0
00:05:00.30 (dt=50.0 ms)
4 (54 ms)
Calculations: 6, detections: 6 (0 ms)
Calculations: 3, detections: 0 (11 ms)
Calculation passes: 10 (35 ms)
C
Count:
419


0
00:04:02.35 (dt=50.0 ms)
Count:
425