SYSC 4805: Computer Systems Design Lab

Project Proposal

Team Spiro Disco Ball

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

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.

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.

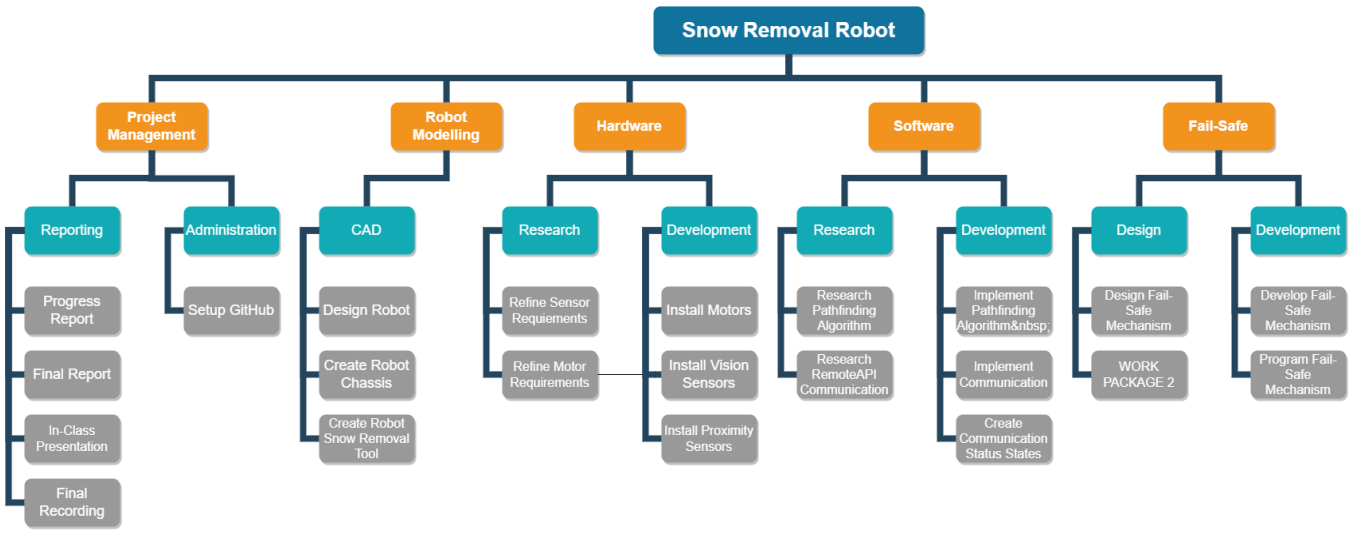


Figure 1: *A work breakdown structure of the activities set to be completed in the project*

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

2.3 Testing

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. The testing can be broken down as follows:

- *Unit Testing*
    - This will involve testing all the different components of the robot to ensure that on their own they are working properly. (ie. testing wheels and joints to ensure they have a proper range of motion)
    - This will also include testing the scripts we plan on using to ensure that they meet all necessary cases.
- *Integration Testing*
    - This will involve running the robot through our own environments to ensure that proper communication amongst parts is maintained throughout the simulation.
    - This involves activities such as moving around obstacles and collecting and removing snow from an area.
- *Environment Testing*
    - The final stage of testing will involve running the system in the training environments and ensuring all requirements with regards to time and completeness of the given task are completed successfully.

Once all the testing described above is complete, we will be able to say with certainty that our robot will be able to function correctly in any future testing environments with some level of success.

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.

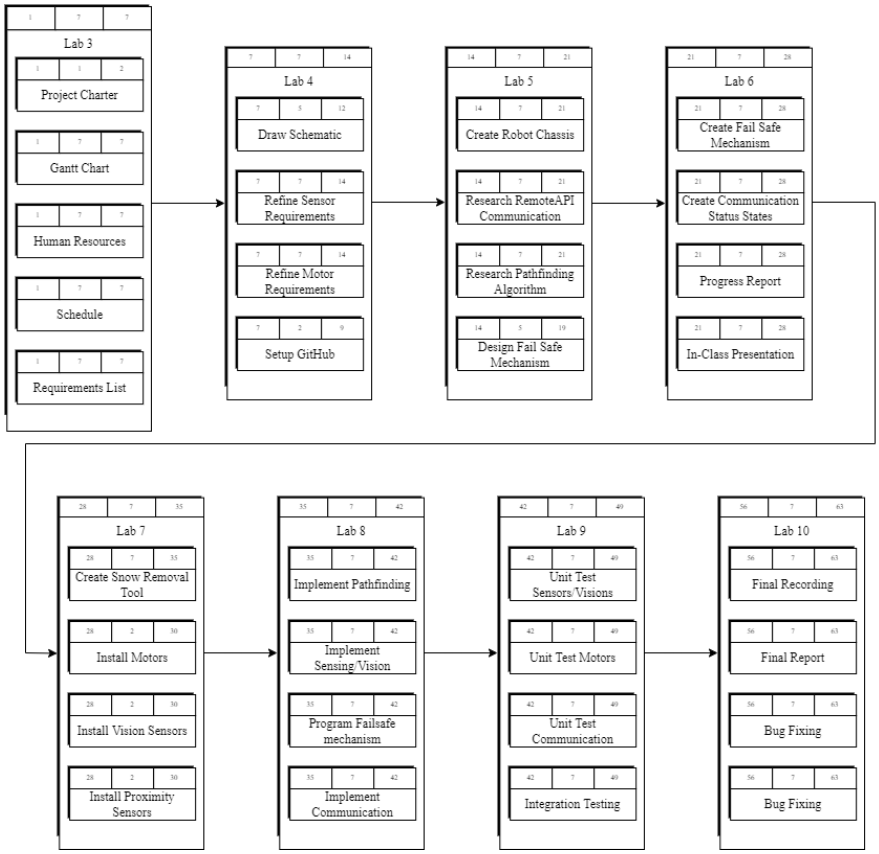
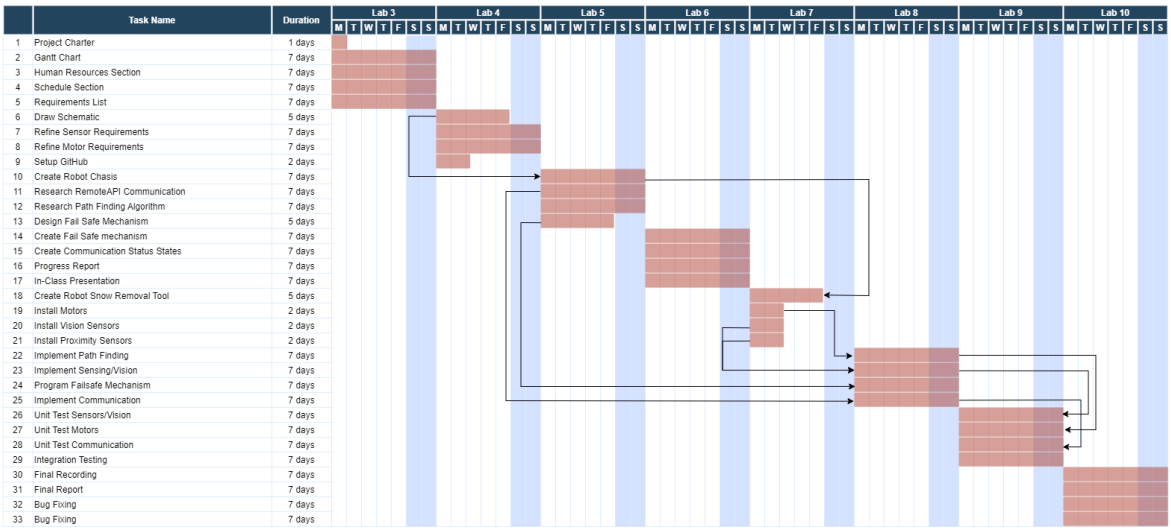


Figure 2: *Schedule diagram of the project deliverables*

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 Resource:

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

5. 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.