SYSC4805A
Computer Systems Design Lab

Automated Snowplow
Progress Report

Section L3 - Group 1 – Unmellow Yellow

Aryan Huq Khan, 101205385
Ibrahim Faisal, 101209598
Raneem Abouseta, 101180464
Rozba Hakam, 101190098
Ali Zaid, 101223823

Professor : Dr. Mostafa Taha

Date : 08-11-2024

Due Date : 15-11-2024

Table of Contents

1 Project Charter

Group Name: Unmellow Yellow (#ffff66)

1.1 Overall Objective

Our team, Unmellow Yellow, aims to design and develop an automated snowplow robot as part of the SYSC4805 Computer Systems Design Lab. The objective of this project is to solve real-world problems related to traditional snow removal, which is frequently labor-intensive, time-consuming, and expensive. This robot will effectively clear snow, represented by lightweight cubes, from a marked area while avoiding fixed and moving obstacles like boxes or other robots. Using sensors for obstacle detection and adaptive navigation algorithms, the robot will autonomously perform the task within the specified size, speed, and time constraints. Most current techniques rely on fixed-path algorithms, which are rigid in dynamic situations. Using adaptive algorithms and various sensor integration, our robot will improve operating efficiency, resulting in safer and more productive snow removal. Through the application of sensor integration, microcontroller programming, and modular design techniques, the project also acts as a hands-on learning experience that gets us ready for future engineering projects, including our fourth-year capstone. Our technical and project management skills are further enhanced by using GitHub to manage the project. Unmellow Yellow, our team, is dedicated to finishing the project in the allocated budget and time frame while making sure we fulfill all performance and efficiency standards. Because it automates snow removal, lowers labor costs, and improves safety during winter maintenance duties, this technology might be extremely valuable to municipalities, property owners, and maintenance companies.

1.2 Overall Deliverables

1) Milestone Deliverables

1.1) Guided Labs

- The labs consisted of testing the following components: VMA330 IR Obstacle Avoidance Sensor, Line Follower Sensor, GP2Y0A51SK0F Distance Measuring Sensor, Ultrasonic Distance Sensor, VL53L1X Time-of-Flight Distance Sensor, MinIMU-9 v5 IMU Unit, Communication between Arduino Nano Every and Due, Cytron Motor Driver, and Wheel Encoders, as well as completing the robot setup and conducting the first drive.

1.2) Project Proposal

- The project proposal will outline the development of an autonomous robot, detailing objectives, milestones, and deliverables. It will include a thorough scope with requirements, a testing plan for unit and integration testing, a schedule with activities and timelines, a cost estimate, and a responsibility assignment matrix for team roles and responsibilities [1].

1.3) Progress Report

- The progress report will provide updates to the project proposal, addressing concerns and refining the system's architecture. It will include state charts, sequence diagrams, watchdog timer implementation, and planned value analysis. Additionally, the report will demonstrate working code with function handlers, and document unit testing results that meet or exceed success criteria [2].

2) Final Deliverables

2.1) Final Presentation

- We will be having our project presentations in class showcasing the works of the snowplow and what we learnt throughout the course [3].

2.2) Final Demo

- The final demo will showcase the fully integrated and functional robot, demonstrating its ability to clear the enclosed area of obstacles and simulated snow. The demo will evaluate the robot's performance based on speed, accuracy, and adherence to task specifications, including obstacle avoidance, boundary detection, and snow clearing efficiency within the allotted time [4].

2.3) Final Report

- The final report will include an updated Project Proposal with individual contributions, control charts for key requirements, and results from system integration testing. It will also include the full working code on GitHub, with both the report and a .zip of the repository [5].

1.3 Overall Architecture

The robot is equipped with a range of sensors that work together to enable autonomous navigation within a designated testing arena. To ensure it remains within the specified boundaries, the robot uses a Line Follower Sensor mounted at the base, which detects the contrast between colors on the ground, such as black tape. This sensor identifies boundary lines, helping the robot distinguish the testing area from its surroundings.

For obstacle detection, the robot relies on multiple sensors positioned strategically around its body. Two **Infrared (IR) Sensors** are placed on the sides, allowing the robot to identify obstacles that might be approaching from the left or right. These IR sensors are essential for detecting obstacles that are not directly in front of the robot, enhancing its awareness of nearby objects and helping it avoid side collisions.

At the front, the robot has two **Ultrasonic Sensors** which measure the distance to obstacles directly in its path. These sensors emit ultrasonic waves and measure the time taken for the waves to bounce back, enabling accurate distance calculation. When an obstacle comes within a critical range, such as 15 cm, the robot's control system is triggered to adjust its path, ensuring a proactive response to objects in front of it.

Additionally, a single **Time-of-Flight (ToF) Sensor** is positioned at an angle, covering an oblique forward view. This sensor is highly accurate in measuring distances to obstacles that are slightly off-centre in the robot's forward path. Unlike the ultrasonic sensors, the ToF sensor operates by measuring the time taken for light pulses to reflect off an object, offering precise distance data. This angled placement allows the

robot to detect obstacles that might appear at an angle, ensuring smooth navigation around obstacles that aren't directly in front.

Together, these sensors enable the robot to autonomously maneuver within the arena, avoiding obstacles from multiple directions and maintaining a safe distance from the boundary, which is crucial for effective navigation and obstacle avoidance.

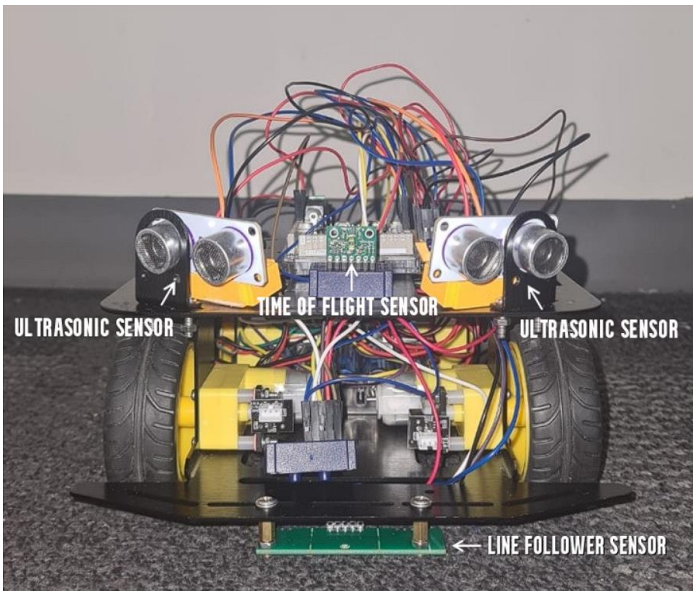

Figure 1: Front View of the Robot with Sensors

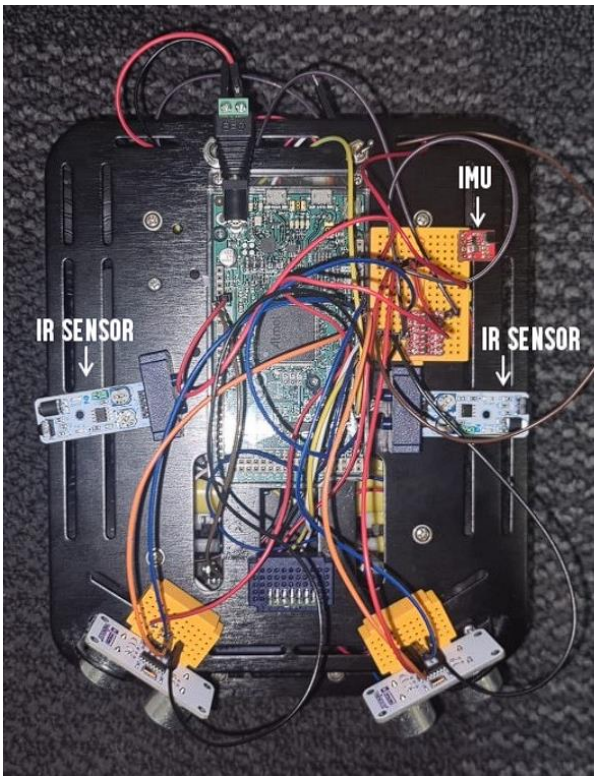

Figure 2: Top view of the Robot with the Sensors

SI.NO	ToF	LFS	LUS	RUS	LIS	RIS	Action	Motor Action
1	0	0	0	0	0	0	MF	FR ↑, BR ↑, FL ↑, BL ↑
2	0	1	0	0	0	0	TL	FR ↑, BR ↑
3	0	1	0	0	0	0	TR	FL ↑, BL ↑
4	0	1	0	0	0	0	HS => BU	- => BR ↓, BL ↓ => FR ↑, BR ↑ / FL ↑, BL ↑
5	0	0	1	0	0	0	MF	FR ↑, BR ↑, FL ↑, BL ↑
6	0	0	1	0	0	0	TR	FL ↑, BL ↑
7	0	0	0	1	0	0	MF	FR ↑, BR ↑, FL ↑, BL ↑
8	0	0	0	1	0	0	TL	FR ↑, BR ↑
9	0	0	0	0	1	0	MF	FR ↑, BR ↑, FL ↑, BL ↑
10	0	0	0	0	0	1	MF	FR ↑, BR ↑, FL ↑, BL ↑
11	1	0	0	0	0	0	HS => TL	- => FR ↑, BR ↑
12	1	0	0	0	0	0	HS =>TR	- => FL ↑, BL ↑
13	1	0	1	0	0	0	TR	FL ↑, BL ↑
14	1	0	0	1	0	0	TL	FR ↑, BR ↑
15	0	0	1	0	1	0	MF	FR ↑, BR ↑, FL ↑, BL ↑
16	0	0	0	1	0	1	MF	FR ↑, BR ↑, FL ↑, BL ↑

Table 1: Truth Table of Sensors and Corresponding Actions taken by the Motor

- ToF – Time of Flight Sensor
  LFS – Line Follower Sensor
  LUS – Left Ultrasonic Sensor
  RUS – Right Ultrasonic Sensor
  LIS – Left IR Sensor
  RIS – Right IR Sensor

- 0 – OFF
  1 – ON

- MF – Move Forward
  MB – Move Backward
  TL – Turn Left
  TR – Turn Right
  HS – Hard Stop
  BU - Move Backward (1 sec) & U-turn

- FR – Front Right Wheel
  BR – Back Right Wheel
  FL – Front Left Wheel
  BL – Back Left Wheel

In the table above we see 16 different actions, these are every possible that can cause an action on the motors, to simplify it we can reduce it to 6 scenarios which are MF – Move Forward, MB – Move Backward, TL – Turn Left, TR – Turn Right, HS – Hard Stop, BU - Move Backward (1 sec) & U-turn. These 6 scenarios are sufficient for testing purposes.

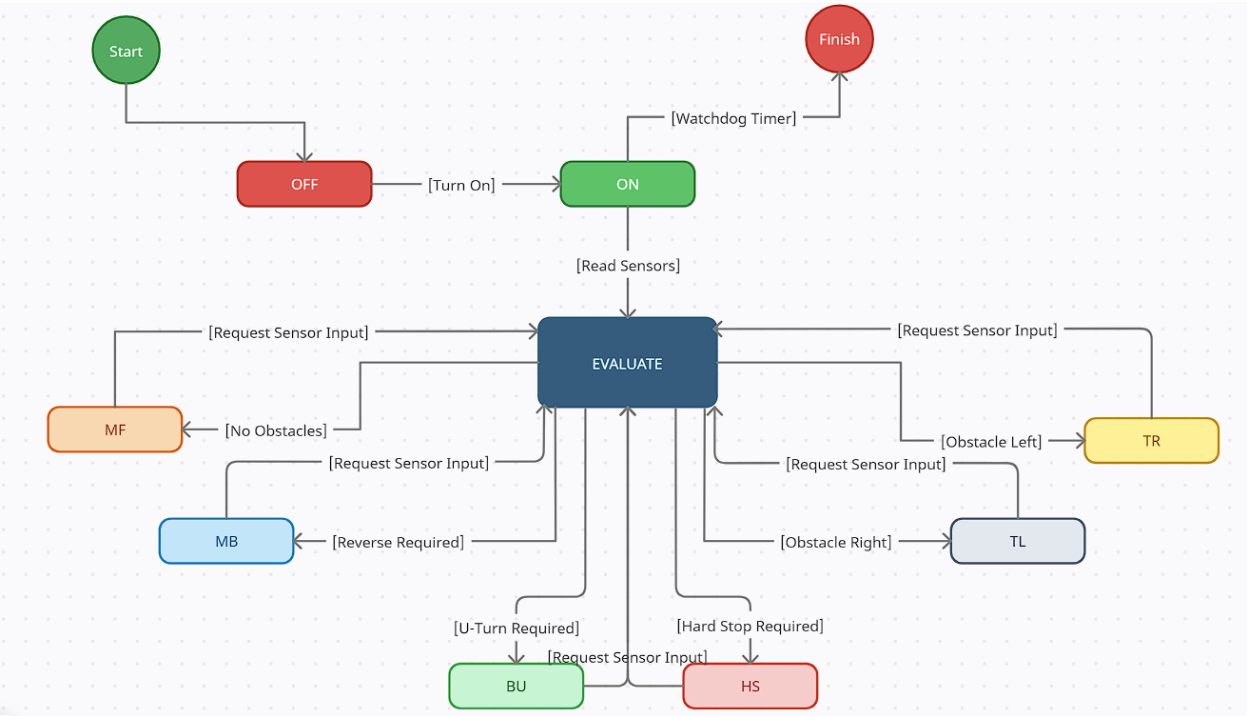

Figure 3: State Chart Diagram of the System [9]

The state diagram represents a robot's navigation system that begins in an "OFF" state and activates to "ON" upon receiving a signal. Once "ON," it enters the "EVALUATE" state, where it continuously

processes sensor inputs to decide on actions like moving forward, reversing, turning, or stopping, depending on detected obstacles. For example, if an obstacle is on the left, the robot turns right, and vice versa. A watchdog timer can transition the system to a "Finish" state for safety, ensuring the robot halts if it operates too long without progress. This setup allows the robot to navigate autonomously by making real-time decisions based on its environment.

The watchdog timer is a safety feature in the robot's code that helps prevent it from getting stuck or unresponsive. It works by setting a time limit (like 2 seconds) within which the robot's main code needs to tell the timer, that the system is working. If the robot fails to reset the timer within this period perhaps because it got stuck in a loop or encountered an error the watchdog automatically restarts the entire system. This ensures that if something goes wrong, the robot will reset itself and try to run the code from the beginning, keeping it reliable and responsive.
For now watchdog timers have been implemented within separate test codes these can be found in out GitHub Repo.

Below is the sequence diagram of how the system will interact:

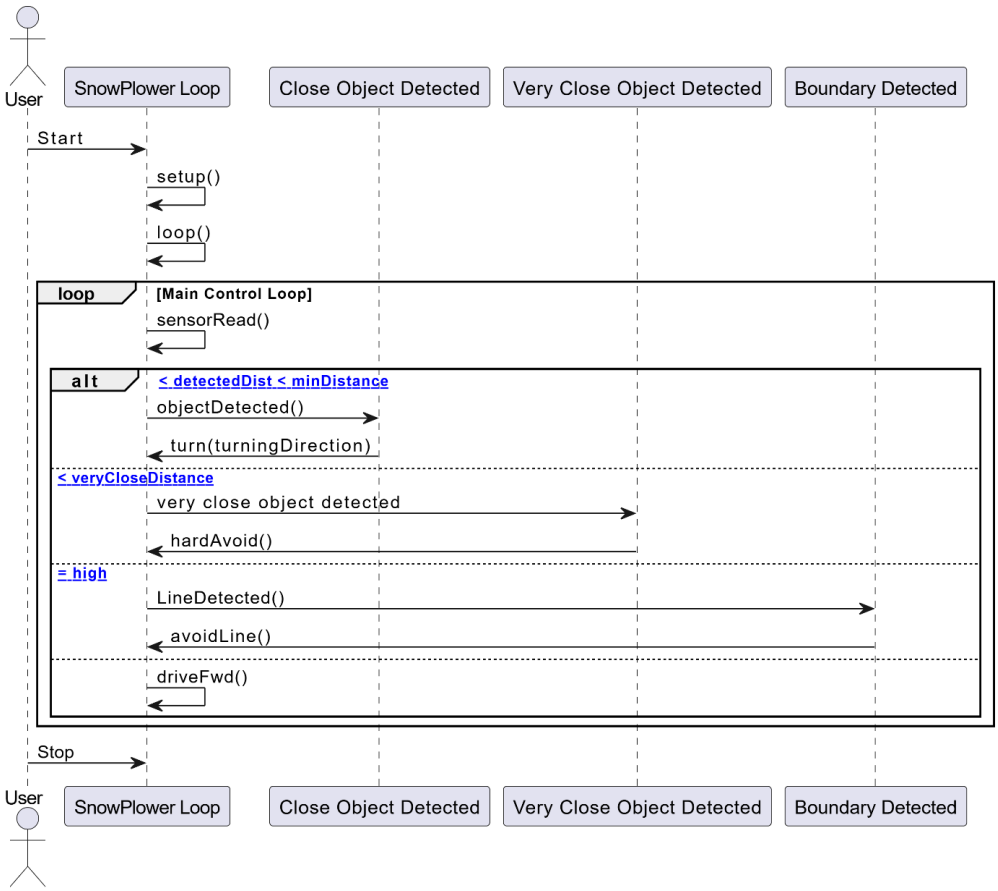

Figure 4: Sequence Diagram of the System [10]

2 Scope

2.1 Requirements

Unmellow Yellow is committed to meeting all the requirements outlined in the project description. This list has been carefully compiled to ensure that we stay within the defined scope of the project and focus on delivering the expected outcomes efficiently and effectively.

1. Functional Requirements (FR):

- FR-01: The robot must not exceed a speed of 30 cm/s while autonomously following a defined path.
- FR-02: It will detect and avoid both fixed and moving obstacles, adjusting its course to avoid collisions.
- FR-03: The robot will clear snow or balls (represented by lightweight cubes) encountered in its path within a designated arena marked by black tape.
- FR-04: If stuck in a loop or between obstacles, the robot will use a watchdog timer to reset or trace back within 8 seconds to free itself.
- FR-05: The robot will provide sufficient power for all sensors to operate continuously and offer real-time feedback on its operational status through an onboard interface.
- FR-06: The robot will plow wooden cubes from the designated area using a snow-clearing algorithm.
- FR-07: The robot will stay within the black tape boundary while navigating.

2. Non-Functional Requirements (NFR):

- NFR-01: The robot will operate efficiently within the defined area, completing tasks within a reasonable timeframe.
- NFR-02: It will minimize power consumption to extend operational runtime.
- NFR-03: The system will be designed for modularity to support easy upgrades or component replacements.
- NFR-04: The robot will maintain the specified dimensions.
- NFR-05: Independent development of sensor and motor modules will allow for easy integration.
- NFR-06: Phased integration of motor, sensor, and control modules will ensure smooth system operation.
- NFR-07: Development will be managed on GitHub, including all version control and documentation.
- NFR-08: The robot will avoid penalties related to speed, collisions, or boundary violations.

2.2 Activities

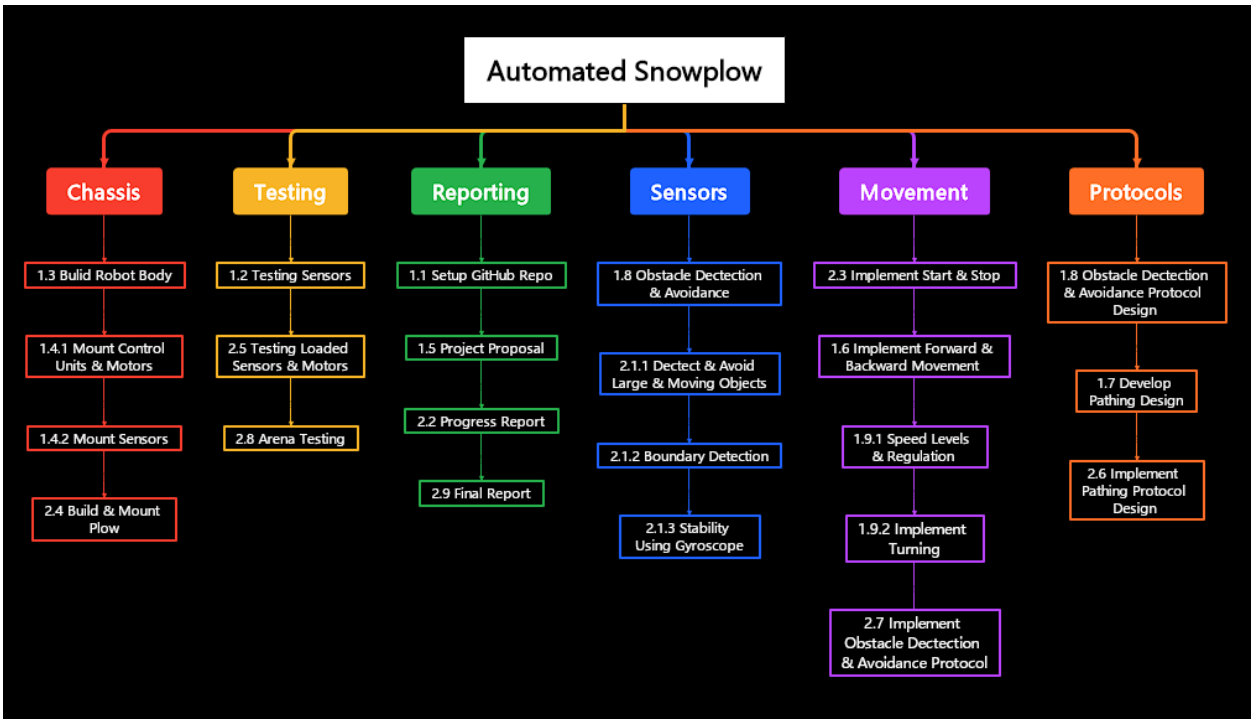

Figure 5: List of Activities shown in a WBS [7]

2.3 Testing Plans

The project includes unit, integration, and end-to-end testing phases to ensure that the robot meets all functional requirements.

1. Unit Testing

Individual tests ensure that each component (sensors and motors) functions as expected.

Sensor Testing:

- Test: Measure each sensor's response at different distances and surfaces.
- Success Criteria: If the sensor accurately detects within ±2 cm of the actual distance, it is a Pass; otherwise, it is a Fail.

Motor Testing:

- Test: Run motors at different speeds, moving forward and backward, and varying frequencies.
- Success Criteria: If the motor moves smoothly in both directions at all speeds, it is a Pass; otherwise, it is a Fail.

2. Integration Testing

Integration tests ensure that all components work together to achieve specific tasks.

Movement Tests:

- Test: Perform movements such as 90-degree turn, 180-degree turn, straight-line driving, and hard stopping.
- Success Criteria: If the robot performs each movement without errors, it is a Pass; otherwise, it is a Fail.

Obstacle Detection:

- Test: Sensors detect obstacles, and the robot reacts accordingly (stopping or changing direction).
- Success Criteria: If the robot detects the obstacle and avoids it without collision, it is a Pass; otherwise, it is a Fail.

Boundary Detection:

- Test: Detect the boundary (black tape) and ensure the robot stays within the designated area.
- Success Criteria: If the robot changes direction upon detecting the boundary, it is a Pass; otherwise, it is a Fail.

3. End-to-End Testing

End-to-end tests evaluate the robot's overall functionality and ability to complete tasks.

Obstacle Avoidance:

- Test: Ensure the robot steers to avoid obstacles during operation.
- Success Criteria: If the robot avoids all obstacles without collision, it is a Pass; otherwise, it is a Fail.

Object Steering:

- Test: Push lightweighted balls (simulated snow) out of the designated area.
- Success Criteria: If 80% of golf balls are moved out of the arena, it is a Pass; if more than 20% of golf balls remain, it is a Fail.

Boundary and Speed:

- Test: Ensure the robot stays within boundaries and operates under the speed limit of 30 cm/s.
- Success Criteria: If the robot remains within the boundary and doesn’t exceed the speed limit, it is a Pass; otherwise, it is a Fail.

2.4 Results of Unit Testing

Item	Result	Success Criteria Results
VMA330 IR Obstacle Avoidance Sensor Module	Notify the Arduino to stop when obstacles on the front of the car are detected.	The car can stop when obstacles appear from the front, the computer shows the reason for stopping (obstacles detected).
Line Follower Sensor	Inform Arduino to turn right, left or 180 degrees when the car reaches the boundary of the test arena.	When the robot car encounters the black boundary, it should notice the computer and turn the direction of travel to ensure that the car does not leave the test site
Ultrasonic Distance Sensor	When the distance of the obstacle in front is less than 15cm, send stopping signals to Arduino.	When there is an obstacle within 15cm in front of the car, stop moving forward and display the real-time distance between the car and the obstacle.
VL53L1X Sensor	When the distance between the car and the obstacles on the left or right is less than 15cm, inform Arduino to turn the direction.	When there is an obstacle less than 15cm away from the car on the left or right side, stop and change the moving direction to the side without the obstacle.
MinIMU-9 v5 IMU Unit	Notify Arduino to stop reversing after 1 second, and then proceed to turn right until no boundaries are detected.	When the car meets the boundary of the test field, it reverses for 1 second, then proceeds to turn right; when the car turns to the direction of clear road condition due to obstacles, stop rotating and continue to move forward
Wheel Encoders	Inform Arduino that the car's two front wheels rotated speed and the forward speed of robotics.	When the car speed is close to 30cm/s, a warning should be issued, and the speed should be reduced to a reasonable value
Motor Driver Board	Notify Arduino when the car needs to speed up, slow down, stop or turn.	The Arduino can change the speed of the car in real time, stop the car, and turn the car by controlling the speed difference of two rear wheels.

Table 2: Unit Testing Results and their Success Criteria (with Arduino)

3 Schedule

3.1 Schedule Network Diagram

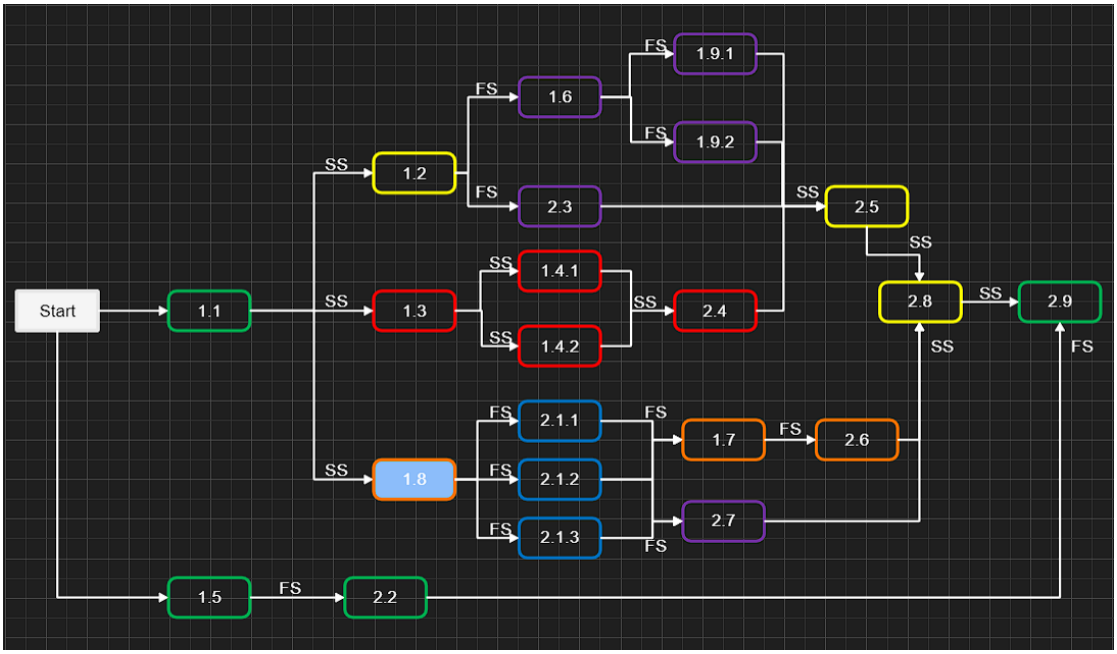

Figure 6: Schedule Network Diagram showing how each Activity is connected [7]

3.2 Gnatt Chart

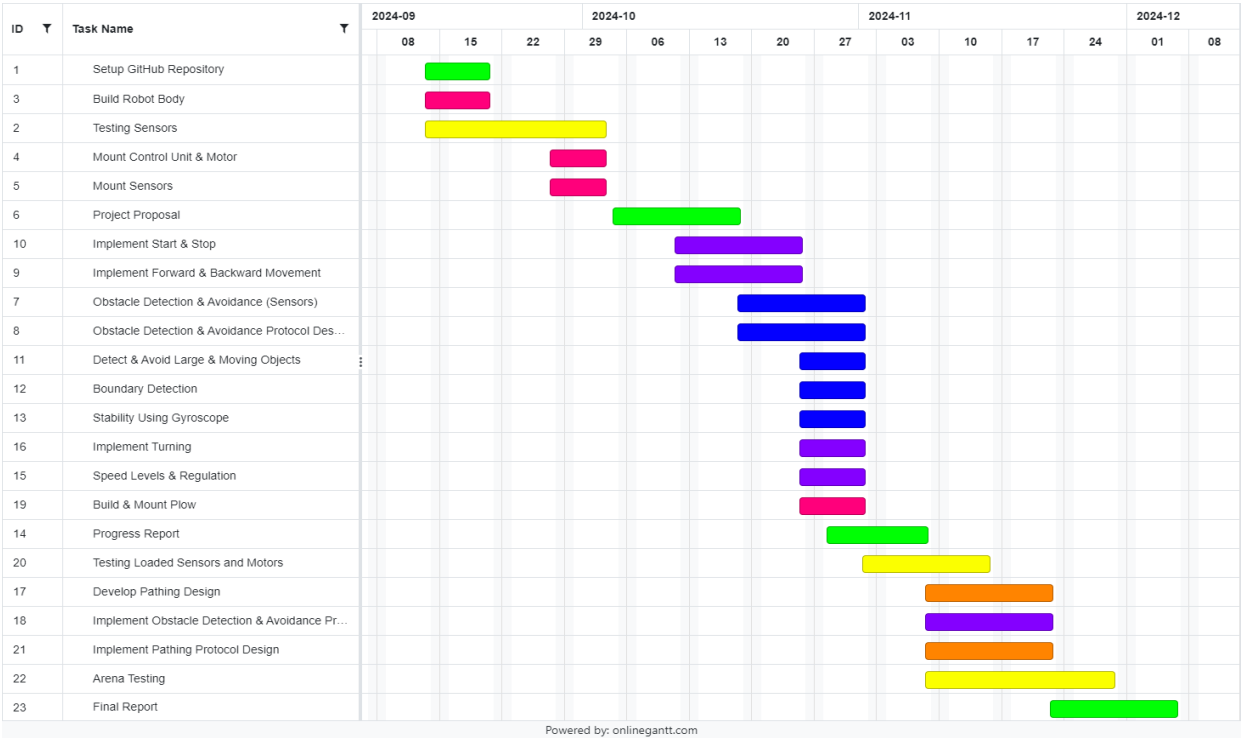

Figure 7: Gantt Chart showing time required for each activity [8]

4 Cost

4.1 Cost Breakdown

For each of the Labs that are conducted throughout the term of this project, 4 hours are assigned for each developer. There is a total of 11 Labs (3 Guided and 8 Unguided) to present a final project. In addition, each member will partake in an extra 2 hours of work per week to provide a report and any additional documentation. However, during the final 2 weeks, each developer will add an additional 2 hour per week on top of the 2 hours to ensure final testing and preparation for the final demonstration and report. Each developer/member will receive a payrate of $50/hour. Our team unmellow yellow has 5 members.

Sl.NO	Week/s	Combined Hours	Cost
1	0 - 3 (Guided Labs)	90 (60Hrs Lab, 30Hrs Report)	$4500
2	4 - 5	60 (30Hrs Project, 30Hrs Report)	$3000
3	6	30 (25 Hrs Project, 5 Hrs Report)	$1500
4	7 (Winter Break)	10 (2Hrs Project)	$500
5	8 - 9	60 (40Hrs Project, 20Hrs Report)	$3000
6	10 - 12	120 (80Hrs Project, 40Hrs Report)	$6000
7	13	0	$0
8 (Total)	14	370Hrs	$18,500

Table 3: Cost Breakdown per week

Package Cost Estimate: Sum of the Cost for the Work Package (Lab Kit)

- Lab Kit is worth $500

Contingency Reserves: Emergency funds for causes that are known

- Estimated cost for any known issues are estimated to be around $150.

Management Reserves: Emergency funds for causes that are known

- Estimated cost for any issues that may arise unknowingly are estimated to be around $50.

Total Baseline Cost = Total Cost + Package Cost Estimate + Contingency Reserves
= $18,500 + $500 + $150
= $19,150

Project Budget Worksheet = Total Baseline Cost + Management Reserves
= $19,150 + $50
= $19,200

4.2 Cost Baseline Figure

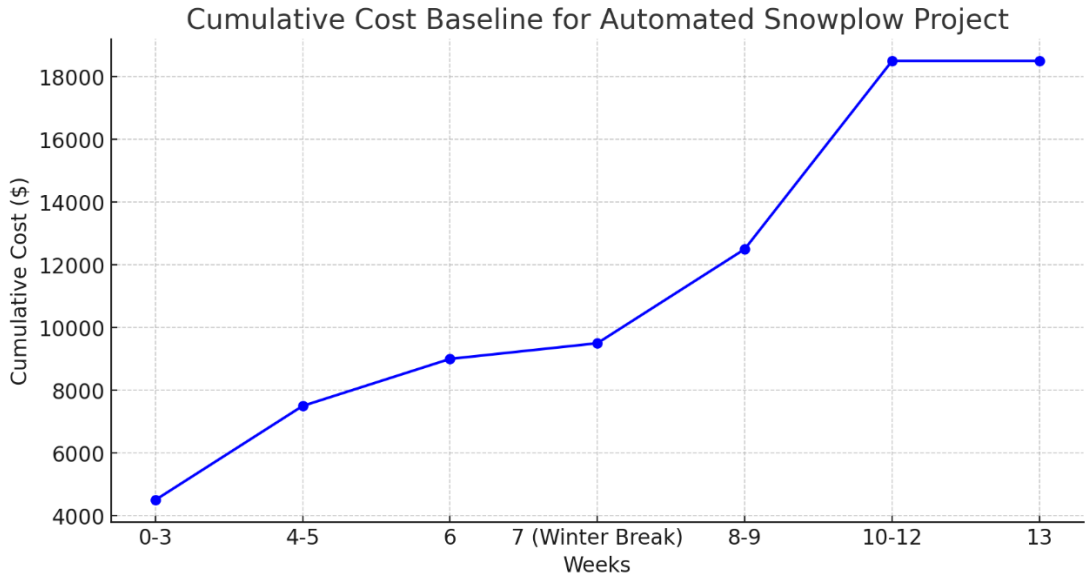

Figure 8: Cost Baseline Figure

4.3 Planned Value Analysis

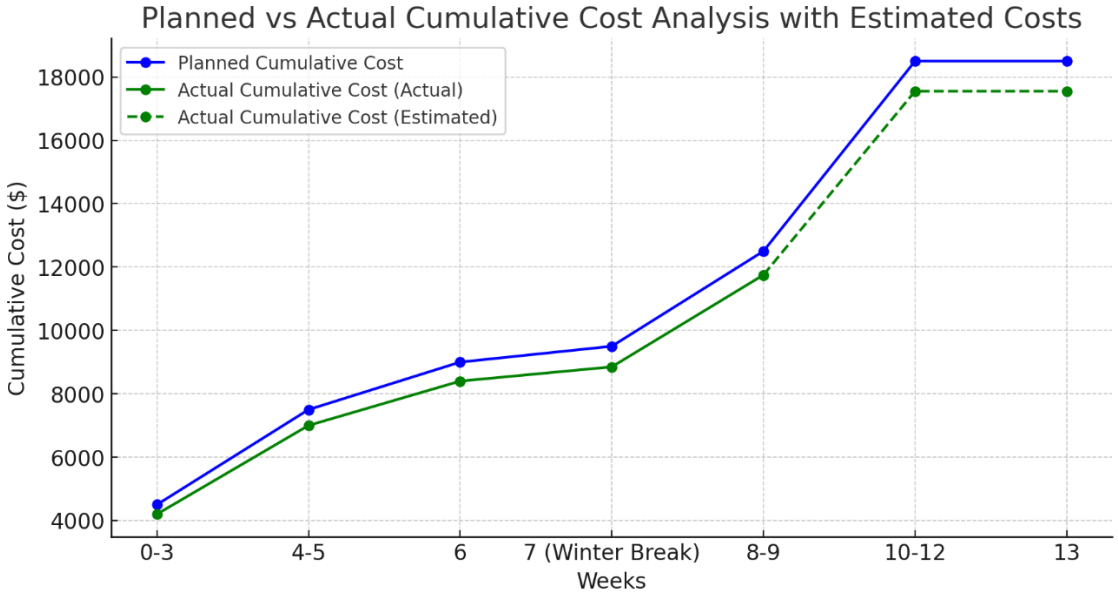

Figure 9: Planned Value Analysis Figure

Based on our analysis, the estimation indicates that we are currently behind schedule but under-budget. This is evident as our actual costs (which represent the number of hours worked) are lower than initially planned. This shortfall is due to several challenges and obstacles encountered throughout the project. If we continue at the current pace, we are likely to fall further behind based on our projected timeline. Consequently, it's crucial for us to accelerate our work in the upcoming weeks to meet our final deadlines and keep the project on track.

Week	Planned Cost ($)	Actual Cost ($)	Planned Cumulative Cost ($)	Actual Cumulative Cost (S)
0-3	4500	4200	4500	4200
4-5	3000	2800	7500	7000
6	1500	1400	9000	8400
7 (WB)	500	450	9500	8850
8-9	3000	2900	12500	11750
10-12	6000	5800	18500	17550
13	0	0	18500	17550

Table 4: Planned Value Analysis Table

5 Human Resources

Responsible - (R), Approver - (A)

Activity No.	Activity	Rozba	Aryan	Ibrahim	Raneem	Ali
1.1	Set Up GitHub Repository	R	R	A	A	A
1.2	Testing Sensors	R, A	R, A	R, A	R, A	R, A
1.3	Build Robot Body	A	A	R	A	R
1.4.1	Mount Control Units & Motors	A	A	R	R	R
1.4.2	Mount Sensors	A	A	R	R	R
1.5	Project Proposal	R, A	R, A	R, A	R, A	R, A
1.6	Implement Forward and Backward Movement	R	R	A	A	A
1.7	Develop Pathing Design	R	A	A	R	R
1.8	Obstacle Detection & Avoidance & Protocol Design (Counts as 2)	A	R	R	A	A
1.9.1	Speed Levels & Regulation	R	A	A	R	A
1.9.2	Implement Turning	A	R	R	A	R
2.1.1	Detect & Avoid Large & Moving Objects	A	A	R	A	R
2.1.2	Boundary Detection	R	A	A	R	R
2.1.3	Stability Using Gyroscope	A	R	A	R	A
2.2	Progress Report	R, A	R, A	R, A	R, A	R, A
2.3	Implement Start & Stop	A	A	R	R	R
2.4	Build & Mount Plow	A	A	R	A	R
2.5	Testing Loaded Sensors & Motors	R	R	A	R	A
2.6	Implement Pathing Protocol Design	A	R	A	R	R
2.7	Implement Obstacle Detection & Avoidance Protocol	A	R	A	A	R

2.8	Arena Testing	R, A	R, A	R, A	R, A	R, A
2.8	Final Report	R, A	R, A	R, A	R, A	R, A

Table 5: Responsibility Matrix

This is the testing arena where the robot moves the snow and avoid the obstacles shown in the image below:

Figure 10: Image of Final Demo Arena

6 GitHub

Below is the link to our GitHub Project Repository:

https://github.com/SYSC4805/project-l3-g1

7 Concerns Faced

Our team, Unmellow Yellow, has identified a few concerns:

1) The current battery pack is too weak and struggles to supply adequate voltage across the different subsystems. To address this, we may need to purchase an additional battery, a battery pack, or a power bank.

2) We've noticed that several components and small items are missing, and sometimes the TAs don't have replacements available.

Apart from these it is a fun project to be a part of.

8 References

[1] – https://brightspace.carleton.ca/d2l/le/content/292069/viewContent/3933858/View (Project Proposal)

[2] - https://brightspace.carleton.ca/d2l/le/content/292069/viewContent/3933860/View (Progress Report)

[3] - https://brightspace.carleton.ca/d2l/le/content/292069/viewContent/3933864/View (Final Presentation)

[4] - https://brightspace.carleton.ca/d2l/le/content/292069/viewContent/3933868/View (Final Demo)

[5] - https://brightspace.carleton.ca/d2l/le/content/292069/viewContent/3933862/View (Final Report)

[6] - https://brightspace.carleton.ca/d2l/le/content/292069/viewContent/3933855/View (Project Description)

[7] - Edraw.AI, https://www.edraw.ai/app/aipower

[8] - "Free online gantt chart software," Free Online Gantt Chart Software, https://onlinegantt.com/#/gantt

[9] - "State diagram maker: State Machine Diagram Tool," Creately, https://creately.com/lp/state-machine-diagram-tool/

[10] - "Draw.io - free flowchart maker and diagrams online," Flowchart Maker & Online Diagram Software, https://app.diagrams.net/