Snowplow Final Report

SYSC4805-L2

Department of Systems and Computer Engineering

Carleton University

Group 7

#A040FF

Daniel Zielinski (101109481)

Advith Thoutu (101229236)

Sanya Peter (101205395)

Manasha Abhayaratne (101244907)

December 4, 2025

1 Project Charter

1.1 Team

As part of the project charter, we were asked to pick a name from the official list of colors. We discussed the choices, and our team has selected #A040FF (Violet) as our official team's name.

1.2 Objectives

The purpose of this project was to design and develop an autonomous robot capable of clearing snow, represented by lightweight wooden cubes, from a defined area marked by a black boundary. The robot needed to operate entirely within this enclosed space while avoiding both fixed and moving obstacles. Our main goal was to create a practical and reliable system suitable for an indoor test environment using simple and affordable components.

The major focus of the project was modular design, where each subsystem such as the motors, inertial measurement unit (IMU), line sensors, and obstacle detection units were individually developed, tested, and integrated into the final platform. Another objective was to maximize snow clearing performance while minimizing penalties, including going out of bounds, colliding with obstacles, or exceeding speed limits. Overall, the project aimed to demonstrate a fully functional hardware–software system that could navigate effectively, avoid obstacles, and maintain sufficient accuracy and clearance for the plow to operate as intended

1.3 System Architecture

1.3.1 Overview

The overall architecture of the autonomous snowplow robot is based on a modular sensing-and-control design centred around a microcontroller. The overall system is divided into several coordinated subsystems, each responsible for a specific set of functions:

- **Microcontroller (Central Control Unit)**
  - Collects data from all sensors
  - Runs navigation, obstacle-avoidance, and decision-making logic
  - Sends PWM and direction commands to the motor driver
- **Sensing Subsystem**
  - Line Sensors: Detect the black perimeter tape to keep the robot inside the boundary
  - Ultrasonic Sensors: Measure distance to obstacles and trigger avoidance behaviors
  - IMU: Provides orientation and drift correction to maintain stable motion

All sensor data feeds into the control loop for real-time adjustments

- **Movement Subsystem**
  - Motor Driver: Converts logic-level signals into motor power
  - Drive Motors: Execute forward, reverse, and turning motions based on control logic

    - Differential motor control allows smooth navigation, line following, and obstacle avoidance
- **Snow-Clearing Subsystem**
    - Plow: Front-mounted blade that physically pushes wooden cubes out of the arena
    - Works in coordination with the movement subsystem to maximize clearing efficiency
- **Safety and Reliability Features**
    - Emergency Stop: Immediately cuts motor outputs
    - Watchdog Timer: Resets the system if software becomes unresponsive
    - This structure ensures the robot can sense its environment, make decisions, and act reliably while clearing snow cubes autonomously.

1.3.2 System Progress

In terms of progress, the major hardware components of the architecture were designed, mounted, and fully tested. The motors were wired and successfully controlled with basic motion commands, and the line-following sensors were calibrated and tested on a mock path, where they performed very well. The ultrasonic sensors were installed and integrated into the detection logic, and they showed strong and reliable performance during obstacle-detection tests. The plow was designed, fabricated, and mounted, with successful tests showing that it could push the required wooden cubes. Overall, the robot worked well, but we did have trouble with the turning logic and getting stable readings from the IMU.

1.3.3 Prototype

This figure shows the completed prototype snowplow in its final configuration prior to demonstration. The system incorporates paired ultrasonic, infrared (IR) obstacle, and line-following sensors, along with a single IMU for orientation. A breadboard is used to interconnect the components in parallel, and power regulation is provided by a 3.3 V buck regulator. The entire system is powered by five AA batteries supplying a total input voltage of 7.5 V.

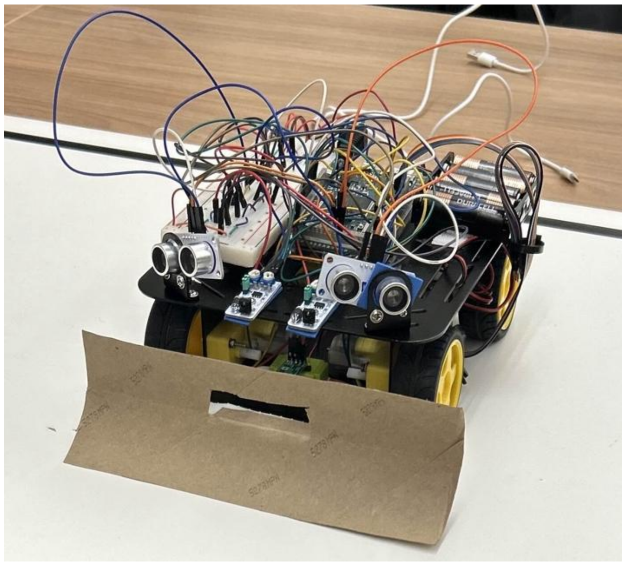
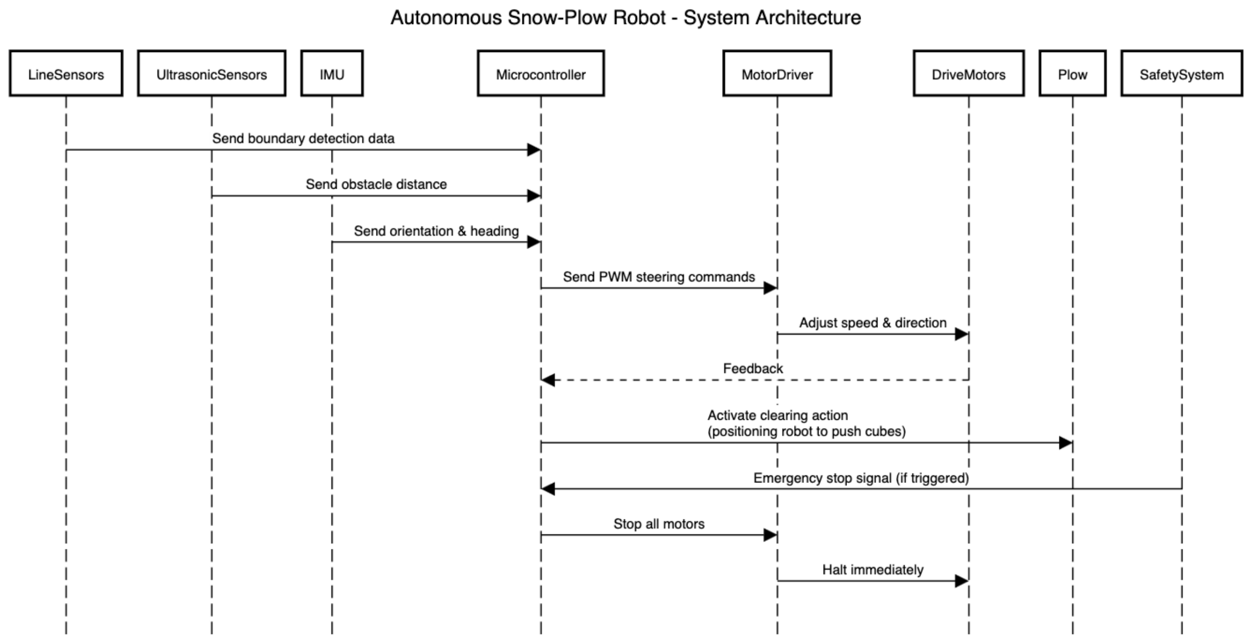


Figure 1: Autonomous snowplow prototype showing the mounted plow, wiring, and sensor layout

1.3.4 Sequence Diagram



Autonomous Snow-Plow Robot - System Architecture
LineSensors
UltrasonicSensors
IMU
Microcontroller
MotorDriver
DriveMotors
Plow
SafetySystem
Send boundary detection data
Send obstacle distance
Send orientation & heading
Send PWM steering commands
Adjust speed & direction
Feedback
Activate clearing action (positioning robot to push cubes)
Emergency stop signal (if triggered)
Stop all motors
Halt immediately

Figure 2: Sequence diagram describing signaling between sensors and Arduino Due

1.3.5 State Diagram

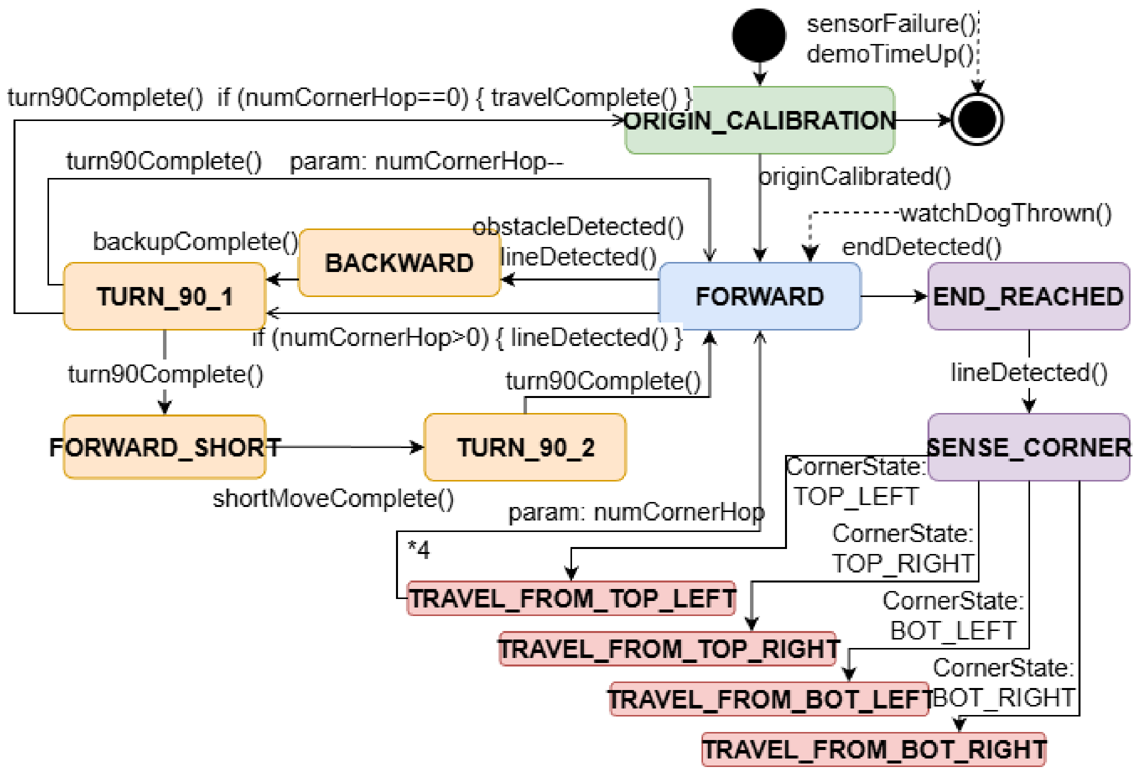


Figure 3: State diagram depicting the algorithm state changes and events

1.4 Deliverables

The project will be accomplished through the execution of a series of structured milestones, in the table below. The following tables consist of each of the deliverables, their description and the status of the deliverables.

1.4.1 Key Deliverables Table

Deliverable	Description	Completed Status
Plow and sensor design	Placement of the sensors for both line-following and to sense obstacles.	The plow has been designed and implemented. Ultrasonic and line sensors have been mounted and tested.
Motor and plow integration	Gather and integrate, with wiring, the drive motors, and the plow. Then we will review and test simple motion.	Basic motor control achieved; fine-tuning motion algorithms completed.
Line-following sensor design setup and test	Mount, calibrate, and test the line-following sensors with the drive motors to follow the line.	Line sensor tested and calibrated successfully on the mock path.
Obstacle detection setup and integration	Mount the obstacle detection sensors and integrate this into our line-following logic.	Ultrasonic sensors installed, software integration completed.
Advanced sensor integration	Integrate the IMU and a plan for how to react to dynamic obstacles, and test integration of sensor inputs for overall functionality.	IMU integration and functionality testing are completed.
System integration and pre-demo tests	Perform a full system test in the arena, get the robot to move as required, the sensors to respond and for it to handle errors as expected.	Full pre-demo testing is complete.
Demo and final evaluation	Perform the full demo, evaluate the snow clearing ability of the robot and submit final documentation.	The demo is complete.

Table 1: Specific deliverables highlighted for each milestone

Stakeholders can expect the fully assembled robot, a documented GitHub repository with modular software, comprehensive unit and integration testing reports, and a final demonstration during Lab 11.

2 Scope

2.1 Requirements [1]

2.1.1 Functional Requirements

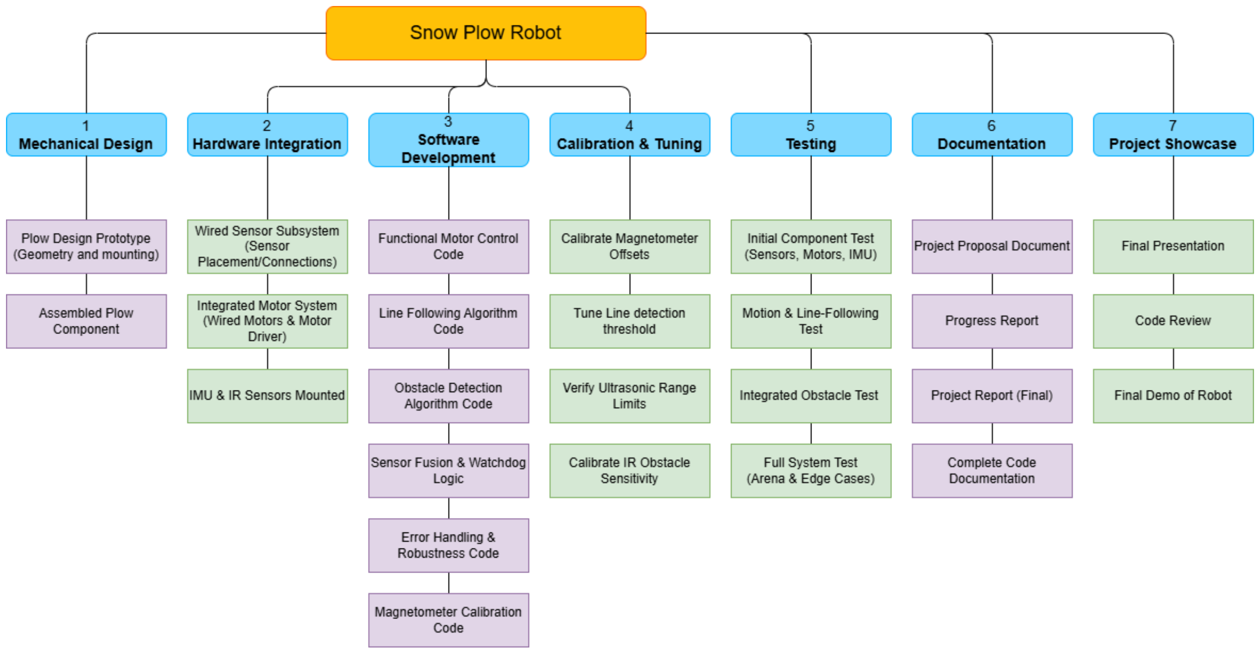
- The plow shall be capable of pushing 20 mm wooden cubes outside the perimeter.
- The robot shall not exceed 5 mins to clear snow.
- The robot shall detect and follow the black taped perimeter without crossing it
- The robot shall begin operation from a designated corner aligned with one edge of the perimeter.
- The robot shall be capable of completing the demonstration without human intervention.
- The robot shall integrate an emergency stop mechanism.

2.1.2 Non-Functional Requirements

- The robot shall not exceed the maximum size of 226 mm × 262 mm × 150 mm.
- The plow shall extend to no more than 50 mm in width and 30 mm in length.
- The robot shall not exceed a maximum speed of 30 cm/s.
- All sensor function handlers shall be modular, reportable, and uploaded to GitHub.

2.2 Detailed Deliverables

2.2.1 Work Breakdown Structure



Snow Plow Robot
1 Mechanical Design
Plow Design Prototype (Geometry and mounting)
Assembled Plow Component
2 Hardware Integration
Wired Sensor Subsystem (Sensor Placement/Connections)
Integrated Motor System (Wired Motors & Motor Driver)
IMU & IR Sensors Mounted
3 Software Development
Functional Motor Control Code
Line Following Algorithm Code
Obstacle Detection Algorithm Code
Sensor Fusion & Watchdog Logic
Error Handling & Robustness Code
Magnetometer Calibration Code
4 Calibration & Tuning
Calibrate Magnetometer Offsets
Tune Line detection threshold
Verify Ultrasonic Range Limits
Calibrate IR Obstacle Sensitivity
5 Testing
Initial Component Test (Sensors, Motors, IMU)
Motion & Line-Following Test
Integrated Obstacle Test
Full System Test (Arena & Edge Cases)
6 Documentation
Project Proposal Document
Progress Report
Project Report (Final)
Complete Code Documentation
7 Project Showcase
Final Presentation
Code Review
Final Demo of Robot

Figure 4: Work Breakdown Structure (WBS) detailing deliverable goals and subgoals

2.2.2 Control Chart

2.2.2.1 Sensor metrics

The chart below shows the metrics for the sensors used on the robot, specifically the line follower sensors and the two ultrasonic sensors**.** Their readings were averaged each week, and the deviation from the expected values was plotted. Over the development period, a clear improvement trend emerged as weekly adjustments were made to address sensor-related issues. Early on, the sensors detected objects only at very close range because their sensitivity was set too low. This was corrected by adjusting the potentiometers to increase the detection range. Additionally, inconsistent or incorrect readings were often caused by faulty or loose connecting wires, unstable voltage, and incorrect timing in the code. As these issues were identified and fixed, the sensors became significantly more stable, resulting in reduced deviation and much more reliable performance over time.

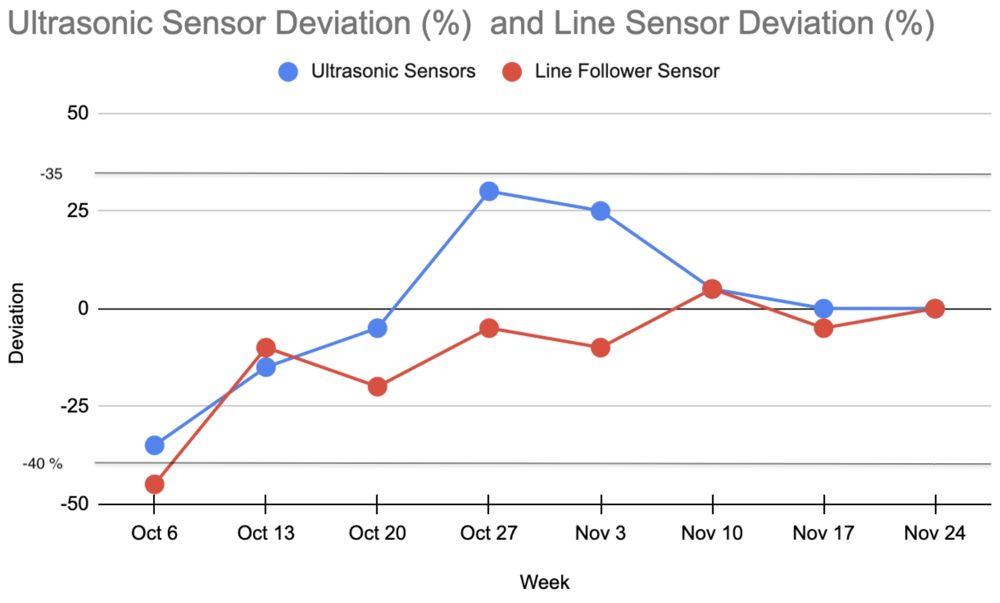


Figure 5: Control chart describing sensor deviations

2.2.2.2 Movement metrics

The following charts show the robot's start-moving delay and turning deviation that was measured weekly throughout development. Early tests showed long delays and inconsistent motor response, mostly due to low battery voltage affecting motor power. We also experienced Cytron motor driver issues where only the right motor responded, caused by a faulty rainbow cable that was later replaced. After fixing wiring, stabilizing battery power, and finalizing turning logic, the robot's responsiveness improved significantly, resulting in faster starts and stable, predictable turning performance across all test runs.

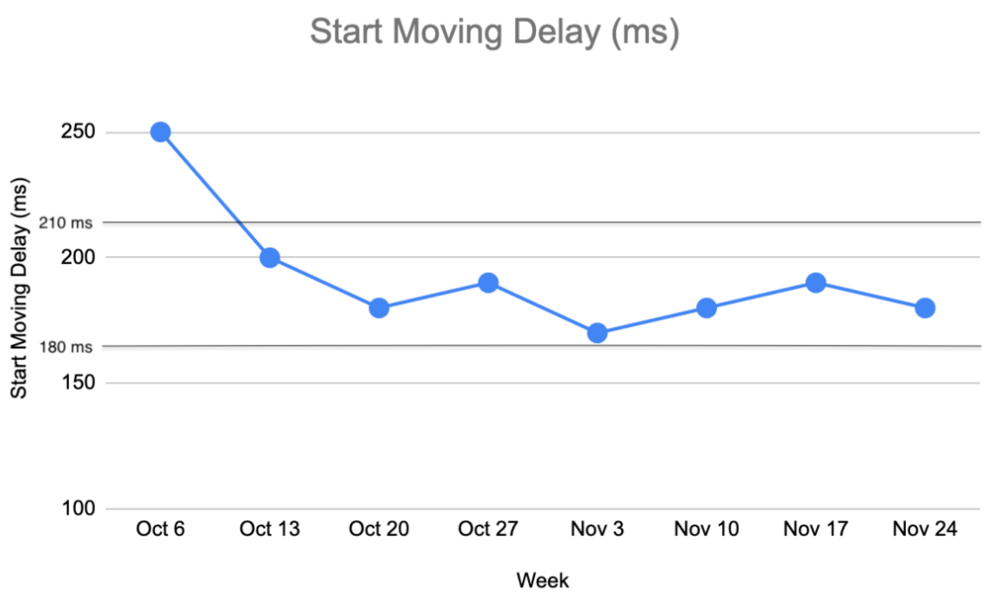


Figure 6: Control chart describing start moving delays

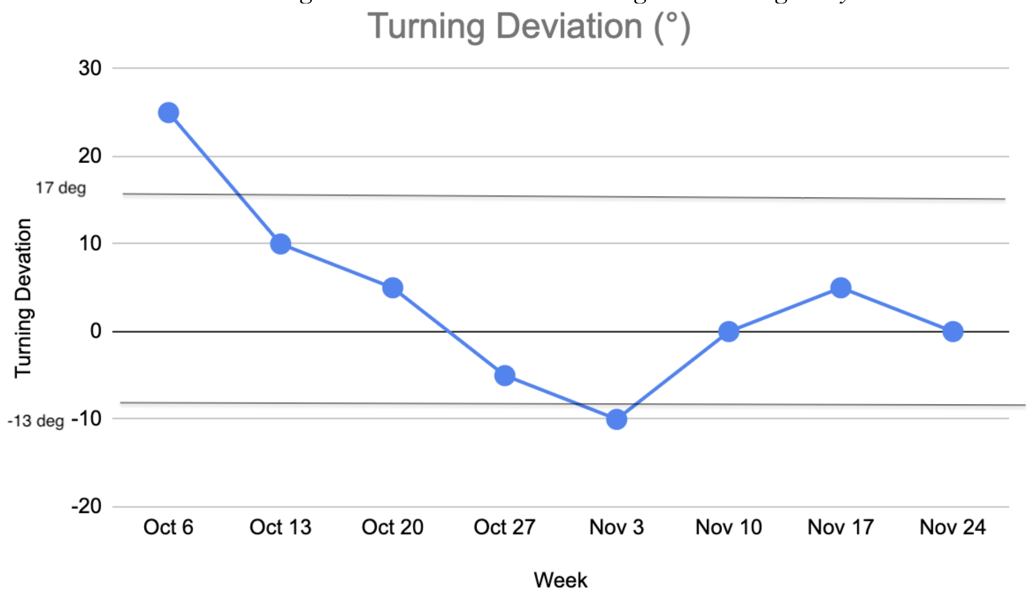


Figure 7: Control chart describing turning deviations

2.2.2.2 Snowplow metrics

The following chart shows the snow clearing performance measured weekly. After mounting the plow, early tests revealed that wooden cubes were slipping under it rather than being pushed forward. We adjusted the plow angle and height to improve ground contact, which increased the cube removal rate. As more subsystems were added, such as IMU correction, obstacle detection, and line following, the robot slowed down between tests, which temporarily reduced the number of cubes cleared. Once these modules were tuned and integrated properly, the plow operated more consistently, allowing the robot to remove cubes more effectively across the final runs.

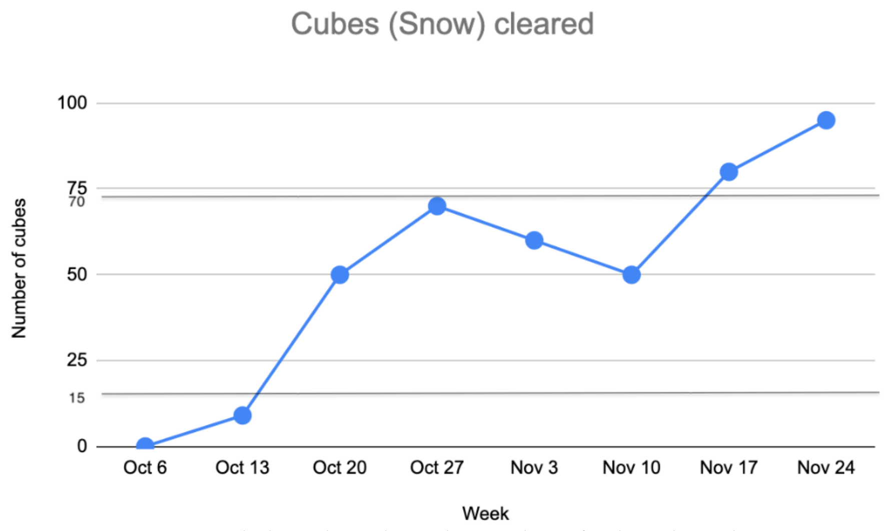


Figure 8: Control chart describing the number of cubes cleared

2.2.3 Watchdog Timer

As part of improving the reliability and safety of our snow-clearing robot, we integrated a watchdog timer (WDT) into our control system. Since the robot must operate autonomously within a closed black-tape boundary while avoiding both fixed and moving obstacles, we needed to ensure the robot never becomes unresponsive during the 5-minute run. A watchdog timer protects against software freezes that could cause the robot to hit obstacles, exceed speed limits, or drift out of bounds.

2.2.3.1 How We Use the Watchdog Timer

We configured the watchdog so that it monitors the main control loop. The timeout is set slightly longer than our expected loop cycle, allowing the watchdog to verify that the robot is still processing sensor data and updating motor commands. Under normal conditions, each loop cycle "feeds" the watchdog, confirming that the system is healthy. If something goes wrong such as a sensor communication hang, a function taking too long, unexpected CPU load, or a logic error the system will fail to feed the watchdog in time. When this happens, the watchdog times out and triggers a controlled recovery response.

2.2.3.2 How Our Watchdog Logic Works

The watchdog tracks the last time the robot was confirmed to be running normally. If too much time passes without this update, it assumes the control loop has frozen. When triggered, the robot immediately stops its motors and returns to a safe state, preventing uncontrolled motion. To verify that the watchdog worked, we intentionally caused the control loop to stall by delaying one of the sensor functions. In this test, the watchdog correctly detected the freeze, stopped the motors, and recovered the system. After the recovery step, the robot resumed normal operation. This confirmed that the watchdog behaved as expected and improved the safety and reliability of our final design.

2.3 Testing Plan

The scope of this project includes both functional and non-functional requirements, a detailed deliverable diagram, and a comprehensive testing plan covering unit and integration testing.

2.3.1 Unit Testing

2.3.1.1 Motor Driver

- **Test:** Apply varying PWM duty cycles and loads.
- **Expected Result:** Output remains a stable PWM signal.
- **Pass Criteria:** Signal varies by ≤2% from set value.
- **Status:** Completed and Passed Criteria
- **Actual Result:** Tested with 100%, 50%, 25%, 0% PWM duty cycles and the signal deviates less than 2%

2.3.1.2 Motor Commands

- **Test:** Send commands (forward, backward, stop, variable speeds) to the motors.
- **Expected Result:** Motors respond correctly within 100ms
- **Pass Criteria:** Commands produce correct motor behavior consistently.
- **Status:** Completed and Passed Criteria
- **Actual Result:** The motor responds correctly to all direction and speed commands with less than 100ms delay

2.3.1.3 IMU

- **Test:** Rotate robot through known angles and record readings.
- **Expected Result:** IMU reports orientation accurately.
- **Pass Criteria:** Readings within ±10° of actual angles.
- **Status:** Completed and Passed Criteria
- **Actual Result:** With the IMU calibrated, the reading is off by ~8° which is within acceptable range

2.3.1.4 Line Sensors

- **Test:** Move robot over a test course with black tape patterns.
- **Expected Result:** Sensors detect boundary reliably.
- **Pass Criteria:** ≥95% correct detection rate.
- **Status:** Completed and Passed Criteria
- **Actual Result:** The test measures the voltage output by each sensor; we observed a clear drop in voltage measured when a darker color is detected compared to the other sensors in same range to ground over white color.

2.3.1.5 Obstacle Detection Sensor

- **Test:** Place objects at distances from 10–20 cm.
- **Expected Result:** Sensor detects objects.

- **Pass Criteria:** Detection occurs reliably within 10–15 cm.
- **Status:** Completed and Passed Criteria
- **Actual Result:** The distance sensor range has been calibrated to detect at a range of around ~5 cm, the IR Obstacle sensor could produce accurate reading with around ~0.5 cm deviation when compared to the actual measured using a ruler

2.3.1.6 Emergency Stop

- **Test:** Trigger stop during full-speed motion.
- **Expected Result:** Motors stop immediately.
- **Pass Criteria:** Motors halt within 0.5 seconds.
- **Status:** Completed and Passed Criteria
- **Actual Result:** The robot came to complete stop within 0.2 seconds of triggering the emergency stop at full speed

2.3.1.7 Result Log

- **Test:** Record all sensor readings and motor commands during operation.
- **Expected Result:** Log holds timestamped motor commands, IMU readings, line sensor states, obstacle events, and plow actions.
- **Pass Criteria:** Logs are complete, accurate, and readable; no missing entries for >1 second.
- **Status:** Completed and Passed Criteria
- **Actual Result:** All motor, sensor reading and states are timestamped and logged

2.3.2 Integration Testing

Stage 1: Motor + IMU Integration

- **Test:** Drive robot in a straight line using IMU feedback.
- **Expected Result:** Robot maintains straight path.
- **Pass Criteria:** Deviation <10 cm over test distance.
- **Status:** Completed and Passed Criteria
- **Actual Result:** The robot had a deviation of ~4 cm while going straight for 1m

Stage 2: Add Line Sensors

- **Test:** Navigate closed-loop boundary course using motors + IMU + line sensors.
- **Expected Result:** Robot follows black tape boundary accurately.
- **Pass Criteria:** Robot stays on track ≥95% of the course.
- **Status:** Completed and Passed Criteria
- **Actual Result:** The robot stayed within bounds and turned around when detected lines

Stage 3: Add Obstacle Detection

- **Test:** Introduce obstacles along path.
- **Expected Result:** Robot avoids collisions while continuing motion.
- **Pass Criteria:** ≥90% successful avoidance without stopping unnecessarily.

- **Status:** Completed and Passed Criteria
- **Actual Result:** The robot avoided obstacle 10 out of 10 trials

Stage 4: Add Plow System

- **Test:** Clear cubes in robot's path while moving.
- **Expected Result:** Robot removes cubes efficiently without tipping or losing control.
- **Pass Criteria:** Clears ≥90% of cubes in a single run; robot remains stable.
- **Status:** Completed and Passed Criteria
- **Actual Result:** The plow picked up most of the snow cubes missing 2 cubes in a test area with 20 cubes

Stage 5: Add Watchdog Timer

- **Test:** Intentionally stall the robot's control loop to trigger the watchdog timer reset.
- **Expected Result:** Watchdog detects the fault condition, automatically resets the system.
- **Pass Criteria:** System reliably resets on each induced stall and resumes normal operation without manual intervention.
- **Status:** Completed and Passed Criteria
- **Actual Result:** WDT reliably detected stall and reset state back to origin calibration (start state) and stopped motors

2.3.3 System Testing

The system implements a robust mowing algorithm that leverages real-time proprioceptive sensing to estimate its current state, direction, and position. The following sequences needed to be tested to ensure states were sufficiently defined:

Sequence 1: Turning around

The robot performs controlled turns at the end of each lane or when redirecting its path. The IMU tracks the rotation angle, and the motors execute a differential turn until the target heading is reached. Testing this sequence ensured the IMU readings were stable, the motors responded predictably, and the FSM advanced only when the turn was truly complete. Since turning happens frequently, consistency here was essential to prevent drift and maintain an accurate clearing pattern.

Sequence 2: Encountering end boundary

When the robot reaches the perimeter tape, the line sensors detect the boundary and trigger a state transition. The robot slows down, aligns briefly to confirm the boundary is real, and then prepares for the next maneuver. Testing focused on verifying consistent detection and ensuring the robot didn't overshoot the arena. Reliable boundary recognition is crucial to maintaining safe operation and preserving the intended mowing pattern.

Sequence 3: Corner sensing

Corner detection occurs when the robot is following a line with one line sensor while the second sensor simultaneously detects the boundary. This combination indicates that the robot has reached a known geometric endpoint. Testing this sequence ensured that corner recognition was repeatable and not triggered by partial boundary readings. Accurate corner sensing is key for resetting orientation and keeping the robot's coverage systematic.

Sequence 4: Using boundary to find next starting point

After clearing through the arena, the robot follows the boundary line to navigate toward the next starting corner. This sequence depends on stable line sensor readings and controlled motor output to trace the perimeter smoothly. Testing ensured the robot could reliably follow the boundary without drifting inside or outside the arena. Consistent boundary following allows the robot to reposition itself correctly for the next cleaning pass.

Sequence 5: Orient at starting point

Once the robot reaches the next corner, it must reorient itself to begin moving to the next corner or to face the correct direction to start the next cleaning pass. Using IMU feedback and short corrective motor actions, the robot aligns to the required heading before transitioning to forward motion. Testing verified that the robot could achieve a stable orientation without oscillation or drift. Proper alignment at each starting point is essential to keep the mowing pattern straight and repeatable.

2.4 GitHub Repository

https://github.com/SYSC4805/project-l2-g2/tree/main

2.5 Lessons Learned/Raised Concerns

2.5.1 Cytron Board Erratic Behaviour

- The Cytron motor driver exhibited inconsistent behaviour, with only the front-right and back-left motors responding.
- Testing confirmed that the board and all motors were fully functional.
- The root cause was a faulty rainbow cable interrupting the connection to the motor driver.
- Bypassing the cable restored full motor operation during testing.
- The rainbow cable was replaced to ensure stable and reliable performance.

2.5.2 IR Obstacle Sensor Low Sensitivity

- The IR sensor detected objects only when extremely close, indicating low sensitivity.
- Adjusting the potentiometer increased detection range to a functional and practical distance.

2.5.3 IMU Prone to Drift

- Relying solely on the gyroscope proved insufficient; successful operation required combining gyroscope and magnetometer data from the IMU.
- Wheel encoders could potentially improve localization accuracy, but further analysis revealed that they are prone to inaccuracy on slippery terrain, suggesting treads may be more appropriate than wheels for reliable navigation.

2.5.4 Low Battery Malfunctions

- Early in development, several system malfunctions were traced to low battery levels.

- Charging the batteries significantly improved overall performance and system stability.
- Powering the Arduino via USB was observed to have a positive impact on battery life.

2.5.5 Power Consumption

- Late in development, it became evident that supplying the Arduino Due to a 7.5V source is inefficient, as the board requires a regulated 5V and the on-board linear regulator dissipates the excess voltage as heat.
- Unregulated voltage results in unnecessary power dissipation, underscoring the importance of using a regulated 5 V supply.
- Employing a buck voltage regulator would significantly improve power efficiency, extend battery life, and reduce thermal stress on the Arduino Due.

2.5.6 Battery Critical Voltage Indicator

- A multimeter-based voltage monitoring code was developed on the Arduino Nano, but incorporating a battery replacement alert was not considered.
- Implementing a critical voltage indicator could provide real-time feedback to the user, mitigating the risk of unforeseen system malfunctions due to low battery levels.

2.5.7 Loss of Progress

- The team did not consistently commit code changes to GitHub or maintain local backup versions during a critical phase.
- This resulted in loss of IMU functionality, drift correction, and state functionality further complicated by the presence of an unseen invalid Unicode character within a codebase exceeding 700 lines, which made resolution impractical without reverting to an earlier version.

3 Schedule

The project schedule includes an activity table, Schedule Network and Gantt chart diagram outlining all major tasks and their timelines.

3.1 Activity List Table

The following activity table consists of each milestone and its corresponding activities.

Milestones	**Sections**	**Activities**
1	Plow Design	Prototype plow geometry and validate size constraints Design and implement plow mounting
	Sensor Placement	Mount IR and ultrasonic sensors Document sensor connections
	Initial Component Testing	Verify individual sensor functionality Execute motor commands, log responses
2	Motor Integration	Wire and connect motors to motor driver Develop and test basic motor control code
	Plow Assembly	Test plow functioning
	Motion Testing	Execute basic movement (90°, 180° turns) Observe and document IMU behavior

3	Wire Configuration	Configure and document all sensor-to-Arduino connections Develop and test power distribution strategy
	Calibration & Tuning	Tune detection thresholds Verify repeatability
	Line-Following Integration	Integrate line-following sensor input with motor control Test line detection and line-following
	Ultrasonic Sensor Integration	Mount ultrasonic sensors in final position Verify ultrasonic sensor communication
4	Time-of-Flight Integration	Removed due to design changes
	Obstacle Detection Code	Develop obstacle detection logic Test detection performance with obstacles
	Algorithm Definition	Define high-level navigation and obstacle-avoidance logic
	Integrated Testing	Integrate obstacle and line detection Test combined functionality
5	IMU Integration	Connect IMU to the Arduino Verify IMU communication code
	Test IMU	Calibrate IMU sensors Validate orientation and heading estimation
	IR Sensor Integration	Mount IR sensors in final position Integrate IR into obstacle detection logic
	Sensor Fusion & Obstacle Detection	Combine IMU, IR, ultrasonic sensor data Test system response to dynamic obstacles, turning behavior
	Watch Dog Timer	Design and implement watchdog timer functionality
6	Algorithm Fortification	Refine logic for improved robustness and reliability
	Full Arena Setup	Construct full test arena Evaluate behavior under edge-case and failure scenarios
	Portable Arena Setup	Develop portable testing environment for continued iteration
	Incremental Algorithm Testing	Perform staged testing of individual behaviors Validate transitions between algorithm states
	Full System Testing	Execute end-to-end system tests with all components active
	Complete Comments	Complete in-code documentation
	Error Handling	Implement fault detection and recovery logic Test system response to error conditions
7	Final Demo	Demonstrate full system operation

Table 2: List of project activities ordered by milestone number

3.2 Schedule Network Diagram

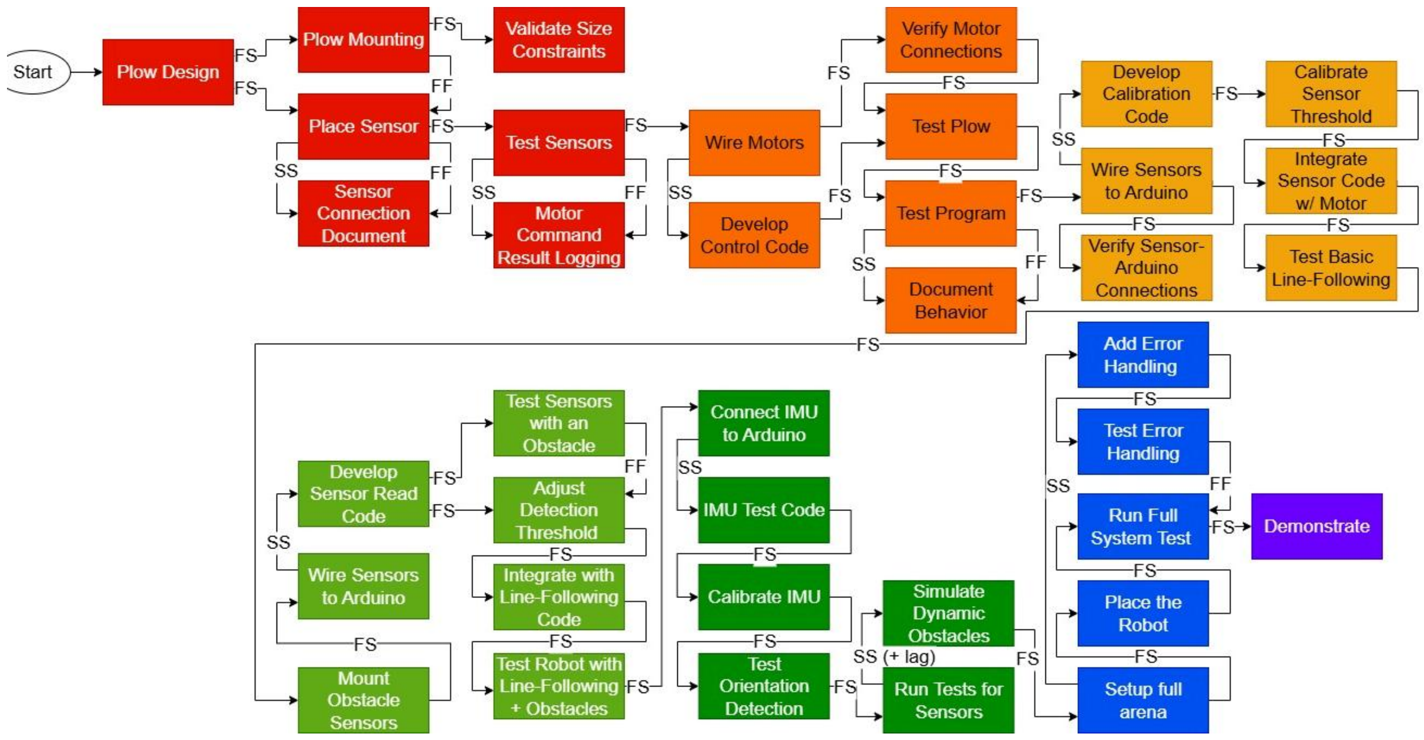


Figure 9: Timeline of project activities ordered by dependence

3.3 Gantt Chart

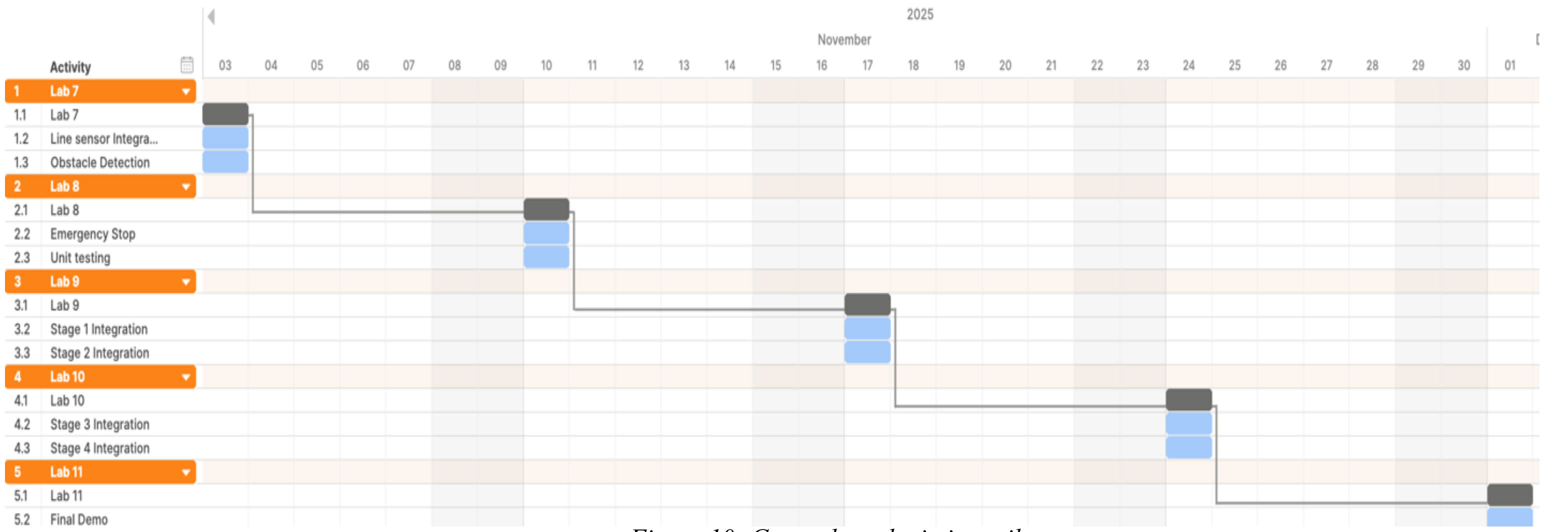


Figure 10: Gantt chart depicting milestones

4 Cost

This section presents the baseline cost estimation and an update on the project's current budget status.

4.1 Baseline Figure [1]

- 4 members with 6 hours each week for 5 labs(lab6-10):
- Cost per worker/week = $50/hour x 4 hours = $200
- Total cost of team = $200 x 4 members = $800
- Preliminary cost = $800 x 5 labs = $4000
- Total estimated cost = $50/hour x 4 hours x 4 members x 5 labs + $500 = $4500

4.2 Updated Planned Value [1]

- Total Component Expenses = $500
- Total Weeks of the Project = 5
- Weekly Expenses = $100
- Earned Value = 5/5 weeks = 100%
  = $4500
- Planned Value = $4500
- Planned Value = Earned Value
  - The project has completed all planned work, and the actual costs match the baseline budget. Cost variance is zero, indicating that the project is on track financially.

4.2.1 Updated Planned Value Table

Category	Cost to Date	Remaining Budget	Progress
Labor	= $800/week × 5 weeks = $4000	= $4000 - $4000 = $0	On Track
Component Expenses	= $100/week × 5 weeks = $500	= $500 - $500 = $0	On Track – No damages, replacements or additional components.
Total	= $4000 + $500 = $4500	= $0	

Table 3: Updated Planned Value (PV)

5 Human Resources

5.1 Contribution Matrix Table

The following table lists all the sections each team member contributed to during the project proposal.

Section	Subsection	Subsubsection	Team Member	Approval
Project Charter	Team	-	Daniel	Sanya
	Objectives	-	Sanya, Daniel	Sanya
	System Architecture	Overview	Manasha, Sanya	Daniel
		System Progress	Manasha	Sanya
		Sequence Diagram	Sanya	Advith
		State Chart	Daniel	Sanya
	Deliverables	-	Sanya	Manasha
Scope	Requirements	-	Daniel	Advith
	Detailed Deliverables	Work Breakdown Structure	Advith	Daniel
		Control Chart	Sanya	Daniel
		Watchdog Timer	Manasha	Advith
	Testing Plan	Unit Testing	Advith	Sanya
		Integration Testing	Advith	Manasha
		System Testing	Manasha, Daniel	Daniel
	GitHub Repository	-	Advith, Sanya	Manasha
	Raised concerns	-	Daniel	Sanya
Schedule	Activity List	-	Manasha, Daniel	Manasha
	Scheduled Network Diagram	-	Daniel	Manasha
	Gantt Chart	-	Sanya	Daniel
Cost	Baseline figure	-	Daniel, Manasha	Advith
	Current Status of the budget	-	Manasha	Sanya
Human Resources	Contribution Matrix	-	Daniel	Manasha
	Responsibility Assignment Matrix	-	Daniel	Manasha

Table 4: Project proposal responsibilities and approvers ordered by section

5.2 Responsibility Assignment Matrix Table

The following table lists all project activities, the team members responsible for completing them, and those who approved of each task.

Activity	Responsible	Approval
Plow Design	Daniel	Sanya
Sensor Placement	Daniel	Manasha
Initial Component Testing	Advith	Daniel
Motor Integration	Manasha	Advith
Plow Assembly	Daniel	Sanya
Motion Testing	Sanya	Manasha
Wire Configuration	Daniel	Manasha
Calibration & Tuning	Advith	Daniel
Line-Following Integration	Advith, Manasha	Daniel

Ultrasonic Sensor Integration	Sanya	Daniel
Time of Flight Integration (depreciated)	Sanya	Daniel
Obstacle Detection Code	Manasha	Advith
Algorithm Definition	Daniel	Advith, Manasha, Sanya
Integrated Testing	Daniel	Sanya
IMU Integration	Manasha	Daniel
Test IMU	Advith	Manasha
IR Sensor Integration	Sanya, Manasha	Daniel
Sensor Fusion & Obstacle Detection	Manasha	Sanya
Watch Dog Timer	Manasha	Daniel
Algorithm Fortification	Daniel	Advith, Manasha, Sanya
Full Arena Setup	Sanya	Advith
Portable Arena Setup	Daniel	Advith
Incremental Algorithm Testing	Daniel	Advith, Manasha, Sanya
Full System Testing	Daniel, Advith	Sanya, Manasha
Complete Comments	Advith	Daniel
Complete Error Handling Logic	Daniel	Advith

Table 5: Project responsibilities and approvers ordered by major milestone activities

6 References

[1] M. Taha, "Project Description – Autonomous Snowplow" SYSC 4805 – Computer Systems Design Lab, Dept. of Systems and Computer Eng., Carleton Univ., Ottawa, ON, Canada, Sep. 18, 2025. [Project Description Document].

[2] A. Kadri, "Lecture – 2: Project Management," SYSC 4805 – Computer Systems Design Lab, Dept. of Systems and Computer Eng., Carleton Univ., Ottawa, ON, Canada, Sep. 18, 2025. [Lecture slides].