SYSC 4805 A
Computer Systems Design Lab
Lab Section L1
Instructor: Abdullah Kadri

Progress Report

Group 5: Periwinkle
Hamnah Qureshi, 101225634
Alvan Chaudhury, 101272922
Noah Oatley, 101189707

Nov 14th, 2025

Table of Contents

1.0 Project Proposal Update

The updated Project Proposal can be seen in Appendix A. Details and explanations will be provided below, including an update on the project itself.

The main loss of points to the Proposal was from the lengthy objective section, which ran over the maximum 200 words and one paragraph format required. The updated Objective section will be reduced to one paragraph, and some of the unneeded words will be removed to make the whole body more concise.

One of the other losses of points was the missing deliverables in section 2. The WBS has been updated to include functional and non-functional requirements. The cost section was also a loss of points due to not factoring in the cost of the kit itself. The table was updated to show the estimated cost before the kit cost, and added the kit cost into the final estimate. The schedule network diagram was also remade to not include any hand drawn parts.

Since the project proposal submission, progress has been made on the drone to integrate all required components. The general functionality of the drone has been complete, and currently the remaining time is being spent ensuring the drone can function optimally. The hour/schedule estimate has been correct so far, as some components have been easier to implement than others.

2.0 Overall Architecture

The autonomous snowplower consists of hardware and software architecture. The hardware architecture includes the Arduino Due, motor driver board, sensors for line follower and detection, and accelerometer. The software architecture leverages the FreeRTOS to control the hardware components and makes decisions based on the reading of the sensors' value. By having the software and hardware components work together, the autonomous snowplower is able to complete its objective of removing artificial snow from a controlled area without colliding with obstacles. Figure 1 shows the architecture diagram and how all the components interact. The Sensor Interface reads values from the hardware sensors and triggers the appropriate tasks such as lineTask() and ObstacleTask(). These tasks call on the motor control functions such as turnLeft(). The motor control module sends the corresponding commands to the motor driver which activates the motors based on the command sent.

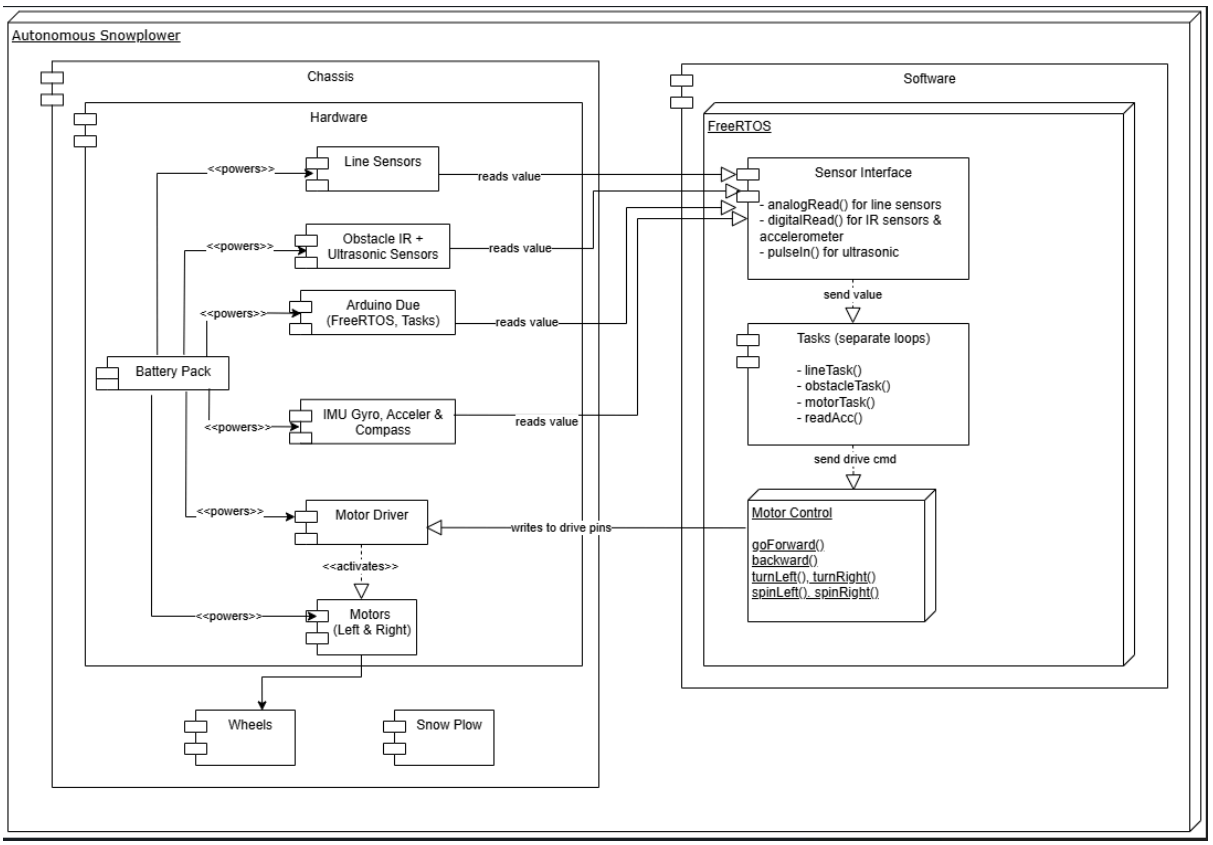

Figure 1: Architecture of the Autonomous Snowplower

2.1 Statechart

This diagram represents the statechart of the autonomous snowplower. The system transitions between states depending on sensor feedback and its detectance and avoidance algorithm. Upon detecting a border or an obstacle the robot transitions into dedicated handling states to avoid obstacles and remain within the area.

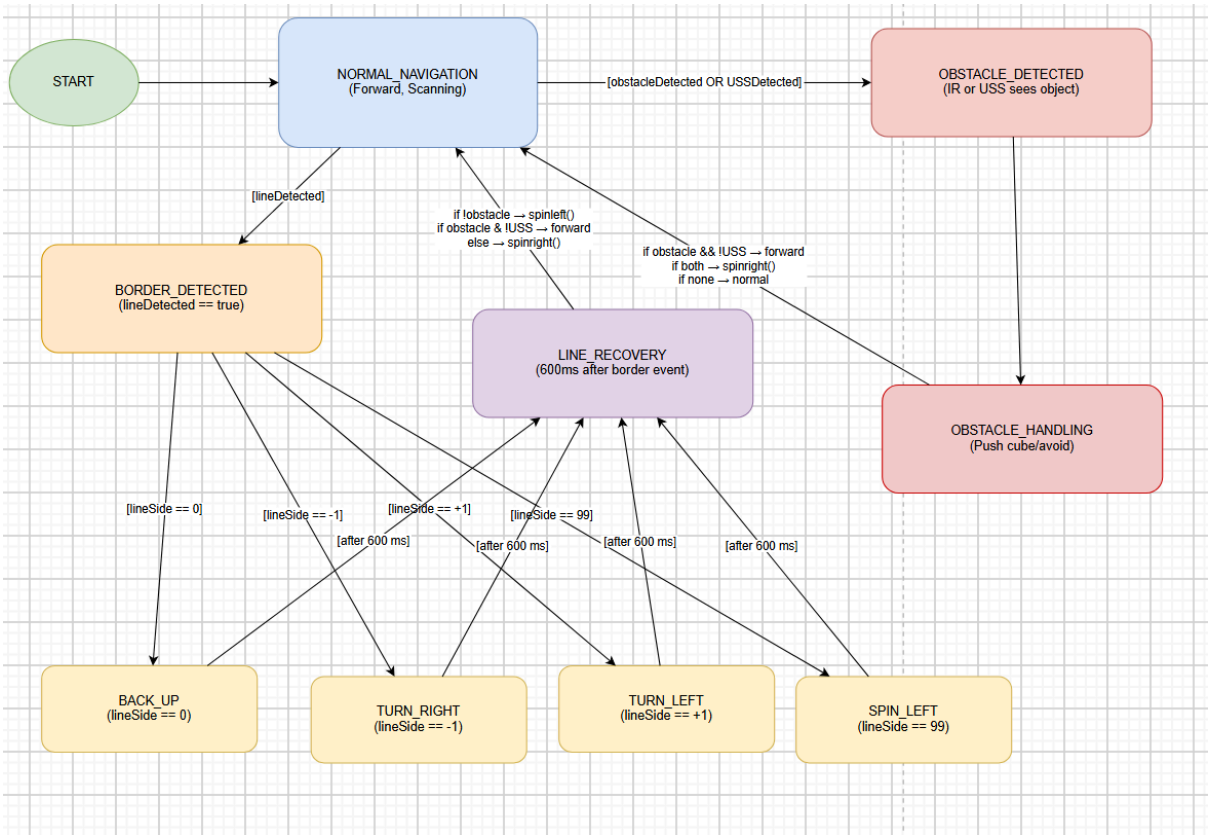

Figure 2: Statechart Diagram of the Autonomous Snowplower

2.2 Sequence Diagram

Figure 3 shows the flow of how the components of the autonomous snowplower work. Since the RTOS system handles the tasks of the robot, it will read the front and back line follower sensor every 15ms and check if it's outside. If the robot is out of bounds, it will send a command to the motors to move backwards or turn. Additionally every 60ms, it reads the value of the obstacle IR and ultrasonic sensors to make sure it won't collide with any object or is close to snow cubes, the motors will be positioned to move backwards or forwards, respectively.

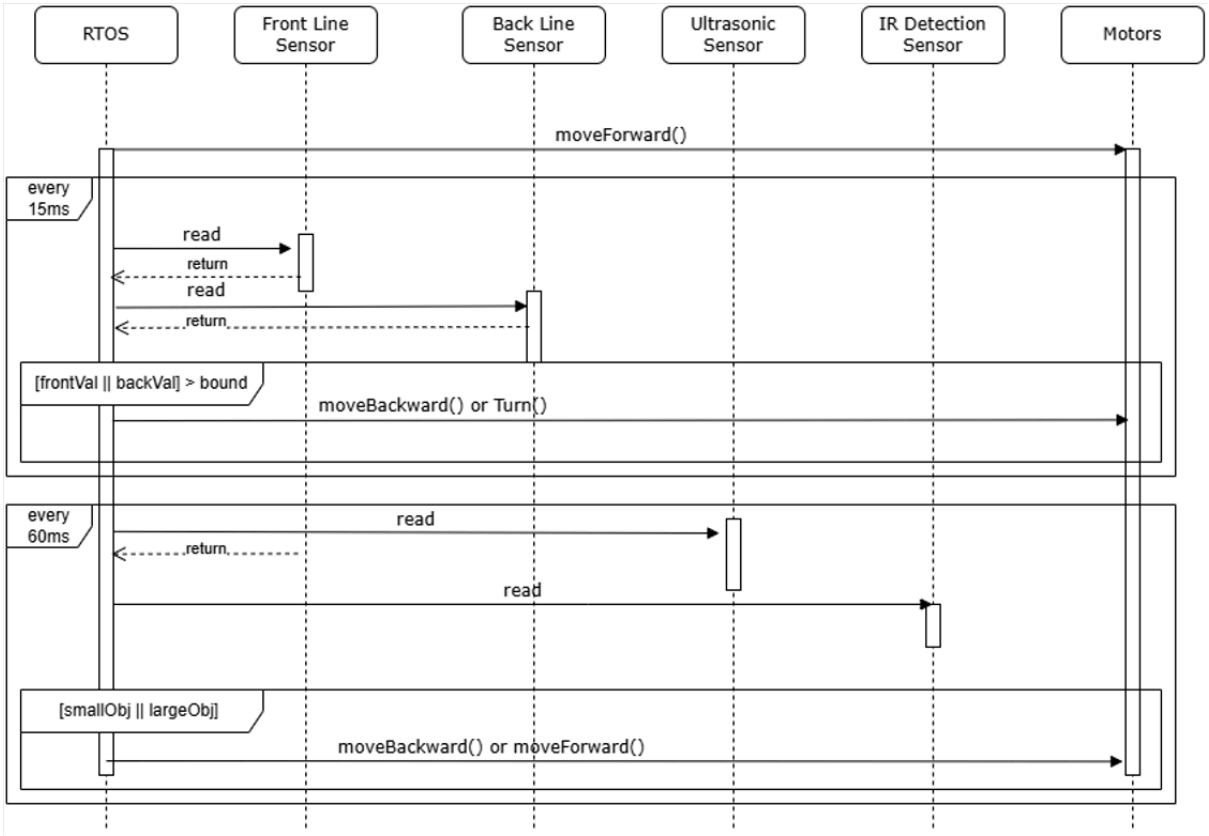

Figure 3: Sequence Diagram of the Autonomous Snowplower

3.0 Updated Planned Value Analysis

As of November 13, our project progress is ahead of schedule, with earned value exceeding both planned value and actual cost. We have completed more work than was initially scheduled for this phase, while managing to keep spending below the anticipated budget. Our costs have remained controlled even as we increased productivity, ensuring that milestones were met faster than expected. Figure 4 shows that our current track indicates that we are well positioned to complete upcoming project phases without risk of delay or budget overruns. Table 1 shows the completion percentage and the associated cost of each activity.

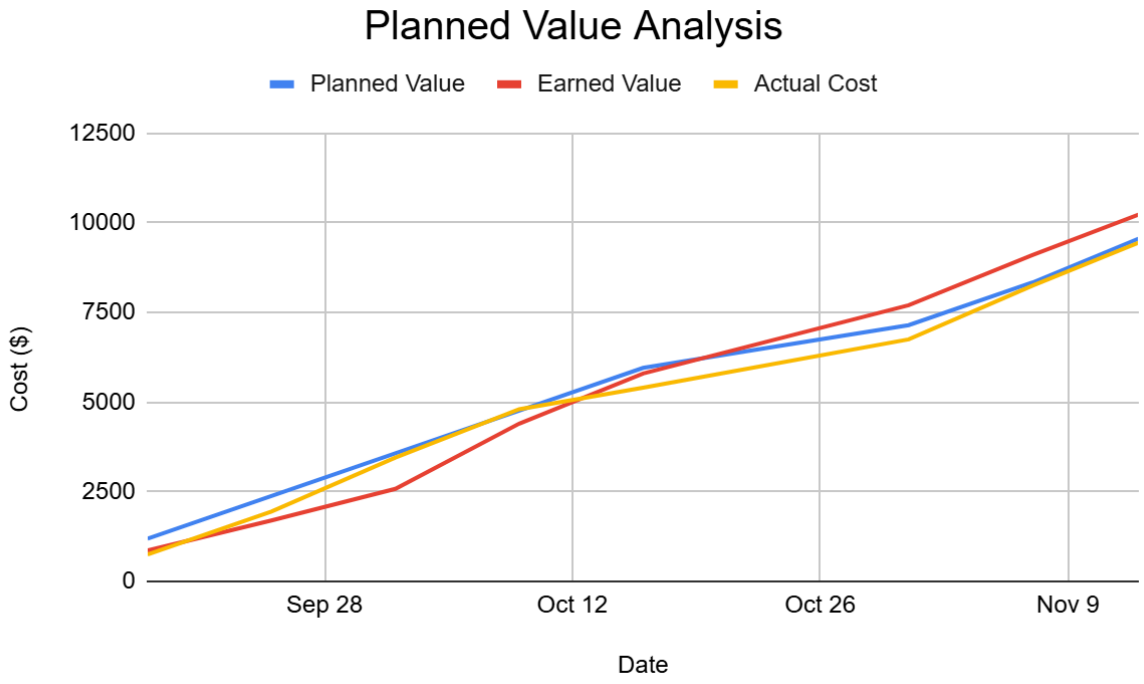

Figure 4: Planned Value Analysis

Table 1: The estimated cost for each activity

Activity	**Cost ($)**	**Completion (%)**
A1.0 Preparing/Testing initial drone	3 000	100
A2.0, A2.1 Coding, Testing, Integration of Line Follower	1 350	100
A3.0, A3.1 Coding, Testing, Integration of Ultrasonic Sensor	1 800	100
A4.0, A4.1 Coding, Testing, Integration of Accelerometer	1 200	100
A5.0, A5.1 Coding, Testing, Integration of Obstacle Sensor	1 800	100
A6.0, A6.1 Coding, Testing, Integration of whole drone	1 350	80
A7.0, A7.1 Creating, Testing of the Plow	150	0
Final Demo	150	0

Earned Value	10 230	78

Planned Value: $13100 * 73% = $9563
Earned Value: $3000 + $1350 + $1800 + $1200 + $1800 + $1350*(0.8) + $500 = $10230
Actual Cost: 63 * $150 = $9450

4.0 Code

The code architecture uses a Free Real Time Operating System (FreeRTOS) which runs many different functions at the same time without blocking each other [1]. Instead of doing a single forever loop with delays and interrupts, each job becomes a task which RTOS schedules them to make the snowplower more responsive and predictable. FreeRTOS manages several tasks such as reading the front and back line sensor, checking the IR obstacle sensor and ultrasonic sensor, and the task to decide what motors should do. Each of these tasks run in its own loop with its own timing. FreeRTOS switches between each task quickly so it feels parallel [1].

Each task is assigned a priority such as line detection is assigned the highest priority as we do not want the snowplower to leave the boundary under no circumstance. The ultrasonic and IR tasks have medium priority while the motors have the lowest priority as it only needs to respond to the commands. If a high priority task wakes up, it interrupts any lower-priority task. Tasks are also woken up based on the time interval they are assigned. They will read the sensor values and check if it's inside the boundary lines. If the robot is outside, the task sends a correction command and sets the event flag telling the motor tasks that if the robot is outside the zone.

4.1 GitHub Repository

The GitHub repository is https://github.com/SYSC4805/project-l1-g5-periwinkle [2]. The repository includes several iterations of the main code on the Arduino Due. For example, code LineFollower.ino was committed to the repo when the front line follower was working as intended. Likewise, BackSensorWorking and ObstacleDetecting were committed when those following sensors were also functioning.

Most of the functions of the components are done in the sensorCheck function, which is called every 1ms. This function updated all of the required values for detecting obstacles, lines, and values that control how the drone should respond. The main loop function is used for the response of the drone, such as turning when it sees a line on its side, and backing up when the line is directly in front of the drone. Other functions in the code are written to make the drone modular, being able to change speeds, thresholds and responses.

Additionally, there is a docs folder that includes the proposal document and the diagrams from the progress report which gives more insight to the users on the architecture of the autonomous robot such as the state and sequence diagram. There is also a README.md that explains how to run the code, which hardware connections to make, and which libraries need to be installed.

4.2 Watchdog Timer

The Watchdog timer was implemented to ensure that the drone will always be functioning and will not stall randomly. In the setup function, the line WDT_Enable(WDT, 0x2000 | 0x20); enables the watchdog timer with a roughly 2 second timeout (0x20). In the loop function, the first line WDT_Restart(WDT); restarts the timer, ensuring that the timeout does not occur when the drone should be functioning correctly. Figure 5 shows both lines of code, where the timer is enabled in line 54, and the timer is reset each loop at line 83.

```
53      // Enable watchdog timer to detect if program hangs
54      WDT_Enable(WDT, 0x2000 | 0x20); // Enable with roughly 2-second timeout
55
56      // Setup motors
57      for (int i = 0; i < numLeft; i++) {
58        pinMode(leftDirPins[i], OUTPUT);
59        pinMode(leftEnPins[i], OUTPUT);
60      }
61      for (int i = 0; i < numRight; i++) {
62        pinMode(rightDirPins[i], OUTPUT);
63        pinMode(rightEnPins[i], OUTPUT);
64      }
65
66      // Setup sensors
67      pinMode(obstacleSensor, INPUT);
68      pinMode(trig, OUTPUT);
69      pinMode(echo, INPUT);
70
71      // Set PWM resolution for Due (valid)
72      analogWriteResolution(8);
73
74      // Start timer interrupt
75      Timer3.attachInterrupt(sensorCheck).start(1000);
76
77
78    }
79
80    // ---------------- Main Loop ----------------
81    void loop() {
82      // Reset watchdog timer
83      WDT_Restart(WDT);
84
85      if (lineDetected) {
86        if (lineSide == 0) {         // Black in front
87          backward();
88          currentAction = 1;
89        }
```

Figure 5: Watchdog Timer in ObstacleSensing.ino

5.0 Testing

Sensor Testing can be found on the GitHub repository under SensorTesting.ino. When run on the arduino, it takes in values from the sensors and prints whether the readings are showing correctly. Calibration can then be made to the working code of the drone based on how differently the testing is from the expected results. Figure 6 shows the Unit Testing for the Line Follower, Obstacle Sensor and Ultrasonic Sensor.

```
void loop() {
  // ---- Line Sensors ----
  int leftVal  = analogRead(lsensor);
  int midVal   = analogRead(msensor);
  int rightVal = analogRead(rsensor);
  int backVal  = analogRead(bmsensor);

  Serial.println("\n--- Line Sensors ---");
  Serial.print("Left: "); Serial.print(leftVal);
  Serial.print(" | Mid: "); Serial.print(midVal);
  Serial.print(" | Right: "); Serial.print(rightVal);
  Serial.print(" | Rear: "); Serial.println(backVal);

  // Evaluate detection
  bool lineFront = (midVal < blacklevl || leftVal < blacklevl || rightVal < blacklevl);
  bool lineBack  = (backVal < backblacklevl);
  Serial.print("Front Line Detected: "); Serial.println(lineFront ? "YES" : "NO");
  Serial.print("Rear Line Detected: "); Serial.println(lineBack ? "YES" : "NO");

  // ---- Ultrasonic Sensor ----
  Serial.println("\n--- Ultrasonic Sensor ---");
  float distance = readUltrasonic();
  if (distance < 0)
    Serial.println("No echo detected (timeout)");
  else {
    Serial.print("Distance: "); Serial.print(distance); Serial.println(" cm");
    Serial.print("Obstacle: ");
    Serial.println(distance < distancethresh ? "TOO CLOSE" : "CLEAR");
  }

  // ---- Obstacle IR Sensor ----
  Serial.println("\n--- IR Obstacle Sensor ---");
  int obstacleVal = digitalRead(obstacleSensor);
  Serial.print("Digital Value: "); Serial.println(obstacleVal);
  Serial.println(obstacleVal == HIGH ? "Obstacle Detected!" : "No Obstacle");

  Serial.println("----------------------------");
  delay(1000);
}
```

Figure 6: Unit Testing Code

The testing is done by taking in the reading values, and printing whether the obstacle or line is detected. For example, when placed on a line, if it prints that there is no line detected, some recalibration is needed.

5.1 Unit Test Results

5.1.1 Boundary Detection Test

Relates to FR-2, FR-6

Objective: This test will evaluate whether the snowplower can remain in the perimeter using a line follower sensor. The line follower sensor should be able to distinguish between black or white surfaces.

Pass criteria: The line follower sensor should have 90% detection accuracy and a latency response of 3-5ms or less.

Method: The sensor was placed over the black tape and white tiled floor while connected to our laptop where in the terminal we outputted the sensor reading. We measured the time for the sensor to notice the transition. We had set the black level threshold as 850.

Observation & Conclusion:

The line follower was able to detect the black line from the white tiled floor with a response time of 3-5ms. Table 2 and Table 3 shows the results from the three tests of the line follower sensor.

Table 2: Black Line test result

Test	Out1	Out2	Out3	Response Time (ms)	Pass/Fail
1	873	864	883	2.43	Pass
2	867	888	854	2.56	Pass
3	865	871	890	3.45	Pass

Table 3: White Line test result

Test	Out1	Out2	Out3	Response Time (ms)	Pass/Fail
1	756	723	736	3.78	Pass
2	743	721	767	3.42.	Pass
3	737	724	743	2.36	Pass

5.1.2 Obstacle Detection Test

Relates to FR-1, FR-6

Objective: This test verifies the ultrasonic and IR sensors that can detect obstacles at varying distances and sizes.

a. Large Object Test

Pass criteria: All obstacles larger than $10cm^3$ should be detected from a distance of at least 30cm with a +/- 10% tolerance.

b. Small Object Test

Pass criteria: All smaller obstacles ($5cm^3$ or less) should be detected from a distance within a minimum of 15cm with a ± 10% tolerance.

Method: While connected to the computer, the distance of the ultrasonic sensor from an object was outputted to the terminal and compared to the actual distance from the object.

Observation & Conclusion:

This test showed the ultrasonic sensor was able to detect small objects through the IR detection sensor and large objects were detected from the ultrasonic sensor. Table 4 shows the results of the three tests of large and small objects using the IR detection and ultrasonic sensor.

Table 4: Large object test result

Test	Target (Distance cm)	Measured (cm)	Absolute Error (cm)	Pass/Fail
1	25	23	2	Pass
2	30	29	1	Pass
3	35	38	3	Pass

Table 5: Small object test result

Test	Target	Measured (cm)	Absolute Error (cm)	Pass/Fail
1	10	10	0	Pass
2	15	15	0	Pass
3	20	21	1	Pass

5.1.3 Motor Speed Test

Relates to NFR-3

Objective: This test measures the motor to maintain a consistent speed below 30cm/s. The speed would be measured using an accelerometer.

Pass Criteria: The robot has a speed of 30cm/s or less across all test conditions with minimum fluctuations.

Method: On the floor, we marked different distances and measured the time it took the robot to reach that distance. We calculated the speed by applying the formula of distance/time.

Observation & Conclusion:

For all the different distances, we noticed that the speed of the snowplower was below 30cm/s. Table 6 shows results of the three tests of the motor speed of the autonomous snowplower.

Table 6: Motor speed test result

Test	Distance (cm)	Time (s)	Speed (cm/s)	Pass/Fail within limit (≤30cm/s)
1	35	1.6	21.875	Pass
2	30	1.17	25.6	Pass
3	24	1	24	Pass

5.1.4 IMU Line Test

Relates to FR-6

Objective: This test validates the IMU sensor is able to maintain directional stability when the robot is following a straight line.

Pass criteria: The snowplower should not deviate more than 5cm from the straight path and should have an orientation accuracy of 95%.

Method: The drone was connected to the computer for Serial Monitor output reading, and was given code to drive straight and print the yaw values. The drone was held in the air before being placed to obtain the initial yaw, which was compared to the constant output for observation.

Observation & Conclusion:

While connected to the computer, the output of the IMU was within a marginally acceptable accuracy when driving shorter distances in a straight line.

5.1.5 Rotation Accuracy Test

Relates to FR-6

Objective: This test checks if the IMU and motor control can perform accurate turns.

Pass Criteria: The snowplower does 90° and 180° turns with an acceptable margin of error of ± 5°.

Method: The drone was connected to the computer for Serial Monitor output, and was given code to constantly turn. The turn speed that was initially determined was compared to how long the drone took to make 90° and 180° turns.

Observation & Conclusion:

While connected to the computer, the IMU showed acceptable deviations in turn accuracy. Notes were taken and the drone was able to make an acceptable 90° turn about 85% of the time. Approximately the same rate was shown for 180° turns.

5.1.6 Time Test

Relates to NFR-4

Objective: The test measures the time it takes for the snowplower to complete a path with efficiency.

Pass Criteria: The snowplower should clear the snow within 5 minutes while staying below the speed limit.

Method: The drone was placed into the square on the ground and was given the functioning code. The drone was timed as it moved around and was stopped when the full perimeter was driven over.

Observation & Conclusion:

The drone was able to consistently move across the entire perimeter in under 5 minutes. Some extra time was taken when stuck in a corner, but was able to correct and move around after a time.

Appendix A: Updated Project Proposal

1 Project Charter

1.1 Overall Objective

Team Periwinkle's objective for the next several months is to design, build and demonstrate an autonomous snowplow capable of efficiently clearing simulated snow from a controlled indoor arena while navigating static and moving obstacles. The snowplow must also be designed to stay within defined physical boundaries. Our team's robot will be engineered to operate within strict dimensional and speed constraints, beginning from a designated corner and using a custom-designed plow to push lightweight wooden cubes out of the enclosed area. The robot must complete its task within a five-minute limit, optimizing path planning and obstacle avoidance using sensor inputs and robust control algorithms [3]. Points will be awarded based on the quantity of snow cubes removed and penalties applied for boundary violations, collisions, or manual intervention. To ensure reliability and modularity, we will develop and unit-test software handlers for all sensors and subsystems, following strict project management and version control practices with regular progress reports and documented criteria for success. Integration will be conducted in stages, allowing for thorough system-level testing and performance verification.

1.2 Overall Deliverables

Team Periwinkle had defined two types of deliverables to make sure this project successful: milestones and final deliverables.

1.2.1 Milestone Deliverables

The milestone deliverables are interim stages that mark significant progress in the project . They help track development, identify potential issues early and ensure the project stays on schedule. The following milestones are set for this project:

- ***MD-1*: Labs**
  Due Date: October 2, 2025
  The guided labs 1-3 represent the first milestone of the *Autonomous Snowplow* project. During these labs, the team focused on learning about the various sensors such as distance, ultrasonic, line follower, and obstacle detection IR, etc and how to effectively use them in the project [4]. These guided labs are an essential milestone in the development of the *Autonomous Snowplow,* as they gave the team the basic foundation of the sensors required to ensure the snowplower meets its requirements.

- ***MD-2*: Project Proposal**
  Due Date: October 16, 2025

The project proposal is a report written by Team Periwinkle that outlines the requirements of the project. This document goes over the deliverables, testing plan and project responsibilities [5]. It provides the team with a plan of action, helping them stay organized and on how to stay on track through this project.

- ***MD-3*: Progress Report**
  Due Date: November 14, 2025
  The progress report is a document written by Team Periwinkle that outlines the progress the team has made during the semester. It details the problems that the team faced and the solutions that were found [6]. It provides an updated version of the milestone deadline which is important to be informed about, in case the team issues any delays in the project.

1.2.2 Final Deliverables

The final deliverables are the completed products of the project. They represent the end goals, and will show the fully developed solution. The following are the final deliverables:

- ***FD-1:* Final Presentation**
  Due Date: November 18, 2025
  The final presentation is where Team Periwinkle presents the *Autonomous Snowplow* project to their professor, TA and fellow classmates. The team outlines their design of the project and their accomplishments.

- ***FD-2:* Final Demo**
  Due Date: November 28, 2025
  The final demo is where Team Periwinkle gives a demo of the *Autonomous Snowplow* in action to their peers. This demonstration would include clearing the wooden cubes which represent the snow in an enclosed area while it attempts to avoid hitting moving obstacles. This milestone highlights the team's achievements and what they were able to accomplish.

- ***FD-3:* Final Report**
  Due Date: December 4, 2025
  The final report is a document written by Team Periwinkle that summarizes the project design and the work that was completed. It also reflects on the lessons learnt throughout the project and outlines future improvements.

2 Scope

2.1 Requirements

The requirements of the project are divided into functional and nonfunctional requirements.

2.1.1 Functional Requirements

This section defines the specific actions and behaviours of the *Autonomous Snowplow* that it must do to achieve its objectives.

- *FR-1:* The robot shall detect and avoid both static and moving obstacles within the area
- *FR-2:* The robot shall follow the black boundary tape and not cross it
- *FR-3:* The robot shall push as many simulated snow cubes as possible out of the defined area
- *FR-4:* The robot shall start from the designated corner
- *FR-5:* The robot shall operate autonomously with no manual interventions during its route
- *FR-6:* The robot shall use ultrasonic, line follower and IMU sensors for environment detection

2.1.2 Non-Functional Requirements

This section outlines the non-functional requirements which describe the performance, reliability, and usability criteria necessary for an optimal design.

- *NFR-1:* The robot must not exceed 226 x 262 x 150 mm in size
- *NFR-2:* The plow must not be not be greater in width than 50mm and not greater in length than 30mm
- *NFR-3:* The robot's speed must not exceed 30cm/s
- *NFR-4:* The robot must clear the snow in 5 minutes or less
- *NFR-5:* The robot's code shall be version controlled and modular for testing individual components
- *NFR-6:* The robot shall undergo unit and integration testing for all major subsystems

2.2 Deliverables

Figure 1 shows the Work Breakdown Structure (WBS) of the *Autonomous Snowplow* project. This hierarchical structure shows how this project is divided into smaller components to make it more manageable for the team.

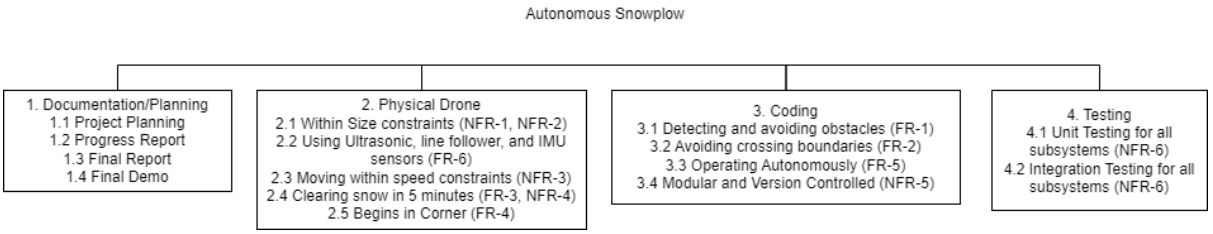

Figure 7: Work Breakdown Structure (WBS) of Autonomous Snowplow Diagram

2.3 Testing Plan

To measure the success of this project, a testing plan has to be developed. The testing plan can be divided into unit and integration testing.

2.3.1 Unit Testing of Components

The unit testing of the components verifies whether each component functions correctly on its own without any interference from other components. This allows the team to establish a baseline and make it easier to identify issues.

Boundary Detection Test
Relates to FR-2, FR-6
This test will evaluate whether the snowplower can remain in the perimeter using a line follower sensor. The line follower sensor should be able to distinguish between black or white surfaces.
Pass criteria: The line follower sensor should have 90% detection accuracy and a latency response of 50ms or less.

Obstacle Detection Test
Relates to FR-1, FR-6
This test verifies the ultrasonic and IR sensors that can detect obstacles at varying distances and sizes.

c. Large Object Test
   Pass criteria: All obstacles larger than 10cm should be detected from a distance of at least 30cm with a +/- 10% tolerance.
d. Small Object Test
   Pass criterial: All smaller obstacles (5cm or less) should be detected from a distance within a minimum of 15cm with a ± 10% tolerance.

Motor Speed Test
Relates to NFR-3
This test measures the motor to maintain a consistent speed below 30cm. The speed would be measured using an accelerometer.
Pass Criteria: The robot has a speed of 30cm/s or less across all test conditions with minimum fluctuations.

IMU Line Test
Relates to FR-6
This test validates the IMU sensor is able to maintain directional stability when the robot is following a straight line.
Pass criteria: The snowplower should not deviate more than 5cm from the straight path and should have an orientation accuracy of 95%.

Rotation Accuracy Test

Relates to FR-6
This test checks if the IMU and motor control can perform accurate turns.
Pass Criteria: The snowplower does 90° and 180° turns with an acceptable margin of error of ± 5°.

Time Test
Relates to NFR-4
The test measures the time it takes for the snowplower to complete a path with efficiency. Pass Criteria: The snowplower should clear the snow within 5 minutes while staying below the speed limit.

2.3.1 Integration Testing of Components

Integration testing of components evaluates how the components work together with the system and if the system works as expected.

1. **Obstacle Avoidance and Boundary Test**
   Relates to FR-1, FR-2. FR-6
   This tests the robot's ability to detect and avoid static and moving obstacles while staying inside the perimeter.
   Pass Criteria: The snowplower has no collisions and does not cross the boundary in 3 consecutive runs.

2. **Snow Clearing and Navigation Test**
   Relates to FR-3, FR-5
   This test evaluates how the snowplower removes the snow which is represented by the wooden cubes by following its designated path autonomously.
   Pass Criteria: The snowplower clears at least 80% of the wooden cubes in the perimeter.

3. **Speed and Performance Test**
   Relates to NFR-3. NFR-4
   This test checks whether the snowplower maintains the speed and performance after the integration of all the hardware components.
   Pass Criteria: The snowplower has a speed below 30cm/s while pushing cubes in the 5 minute time-limit.

4. **Start and Stop Automation Test**
   Relates to FR-4, FR-5
   This test checks whether the snowplower can start its navigation from a corner and complete its route without any manual interference.
   Pass criteria: The snowplower has done runs multiple times and does not require any manual inputs.

5. **Sensors Integrations Test**
   Relates to FR-6
   This test assesses how well the ultrasonic, distance, IMU, and line follower sensors integrate and communicate with each other to have a smooth navigation. The system should take in data from all the sensors and make decisions based on them.
   Pass Criteria: All the sensors provide consistent data for smooth and accurate navigation.

6. **System Reliability Test**
   Relates to NFR-5, NFR-6
   This is one of the final tests that ensures that the snowplower is responsible to complete its tasks without any component or bugs occurring during its operation.
   Pass Criteria: The snowplower has done runs multiple times and has 95% confidence in its reliability.

3 Schedule

3.1 Activities

The list of activities needed that covers the project is as follows:

- A1.0 Preparing/testing the drone with arduino and motor control board for components

- A2.0 Coding and testing of the obstacle sensor
- A2.1 Integration of obstacle sensor with drone

- A3.0 Coding and testing of the line follower
- A3.1 Integration of line follower with drone

- A4.0 Coding and testing of the ultrasonic sensor
- A4.1 Integration of ultrasonic sensor with drone

- A5.0 Coding and testing of the accelerometer, gyroscope and magnetometer
- A5.1 Integration of accelerometer with drone

- A6.0 Integration of the components, arduino, and motor control board
- A6.1 Testing of the fully integrated drone

- A7.0 Creating the shovel for the snowplower
- A7.1 Integration and testing of the shovel with removing blocks

- A8.0 Demonstration of final product

These activities may not be completed in the order presented, but it is a good idea to code, test, and integrate each component sequentially to keep the project modular and iterative, while making final integration testing easier.

3.2 Schedule Network Diagram

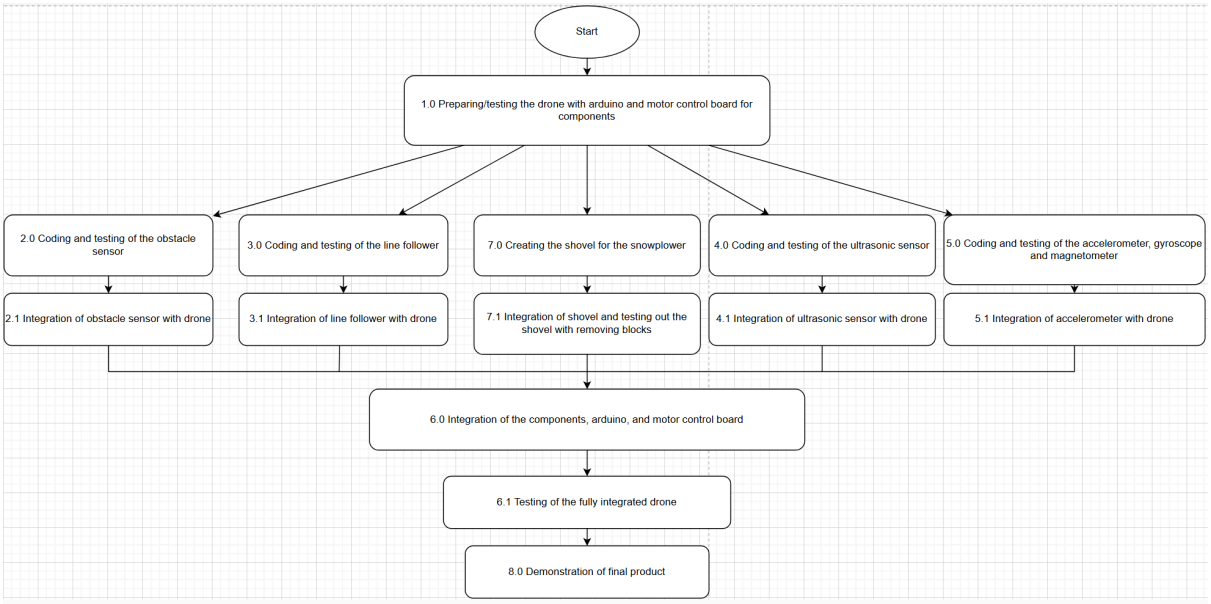

Figure 8: Schedule Network Diagram of Autonomous Snowplow Diagram

3.2 Gantt Chart

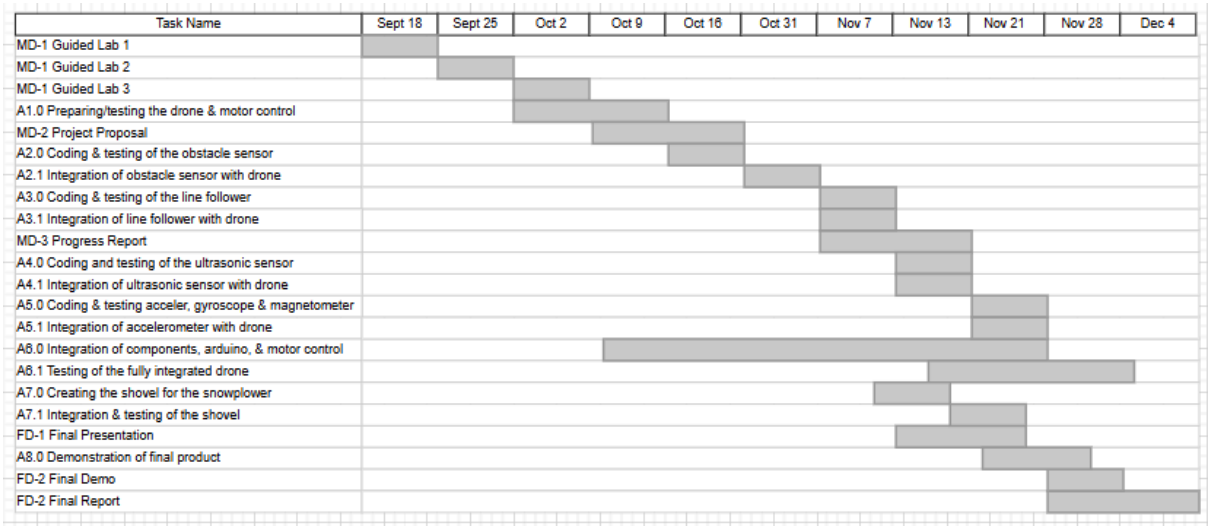

Figure 9: Gantt Chart of Autonomous Snowplow Diagram

4 Cost Baseline

For our team of three members, the estimated cost of the project is around $500 for the drone kit, and an additional $150/hr for working hours.

Table 7: The estimated cost for each activity

Activity	Estimated Time (hours)	Cost ($)
A1.0 Preparing/Testing initial drone	20	3,000
A2.0, A2.1 Coding, Testing, Integration of Line Follower	9	1,350
A3.0, A3.1 Coding, Testing, Integration of Ultrasonic Sensor	12	1,800
A4.0, A4.1 Coding, Testing, Integration of Accelerometer	8	1,200
A5.0, A5.1 Coding, Testing, Integration of Obstacle Sensor	12	1,800
A6.0, A6.1 Coding, Testing, Integration of whole drone	9	1,350
A7.0, A7.1 Creating, Testing of the Plow	1	150
A8.0 Final Demonstration	1	150
Total	72	10,800

Using our estimates from the schedule and future responsibilities, an estimated time of work per week is around 6 or 7 hours per week for the semester. Factoring in the cost of the drone kit ($500) and using the Carleton schedule of 12 weeks, the estimated total cost of the project is between $11,300 and $13,100.

5 Human Resources

5.1 Responsibility Assignment Matrix

Table 8: The responsibility of each activity in the project

Activity	Who is Responsible	Who Approves
Preparing and Testing Drone for Component Addition	All	All
Coding and Testing Obstacle Sensor	Hamnah	Noah
Integrating Obstacle Sensor	Hamnah	Noah
Coding and Testing Line Follower	Hamnah	Alvan
Integrating Line Follower	Noah	Hamnah
Coding and Testing Ultrasonic Sensor	Alvan	Hamnah
Integrating Ultrasonic Sensor	Alvan	Hamnah
Coding and Testing Accelerometer	Noah	Alvan
Integrating Accelerometer	Noah	Alvan
Creating the shovel for the snowplower	Alvan	Noah
Integration of shovel and testing out the shovel with removing blocks	All	All
Integration of the components, arduino, and motor control board	All	All
Testing of the fully integrated drone	All	All
Demonstration of final product	All	Professor/TAs

6 References

[1] “bdmihai/DueFreeRTOS: FreeRTOS library for Arduino Due,” *GitHub*, 2025. https://github.com/bdmihai/DueFreeRTOS

[2] “SYSC4805/project-l1-g5-periwinkle: project-l1-g5-periwinkle created by GitHub Classroom,” *GitHub*, 2025. https://github.com/SYSC4805/project-l1-g5-periwinkle/tree/main

[3] Dr. M. Taha, “Computer System Design Lab - Project Description Autonomous Snowplow,” brightspace.carleton.ca, Aug. 2024. https://brightspace.carleton.ca/d2l/le/content/372623/viewContent/4450646/View

[4] Dr. A. Kadri, “Computer Systems Design Lab - Inventory,” brightspace.carleton.ca, Aug. 2025. https://brightspace.carleton.ca/d2l/le/content/372623/viewContent/4431287/View

[5] Dr. A. Kadri, “Computer System Design Lab - Project Proposal,” brightspace.carleton.ca, Aug. 2025. https://brightspace.carleton.ca/d2l/le/content/372623/viewContent/4450656/View

[6] Dr. A. Kadri, “Computer Systems Design Lab - Progress Report,” brightspace.carleton.ca, Aug. 2025. https://brightspace.carleton.ca/d2l/le/content/372623/viewContent/4450655/View