SYSC4805 - Final Report

Team Kobe

Group 3

Dearell Tobenna Ezeoke	**101245819**
Tobiloba Ola	**101244412**
Partha Das	**101191302**
Ahmad Mohamad	**101230403**

Section: L3

Date: December 5th, 2025

Overall Objective:

The purpose of this project is to design, develop, and test a self-driving snowplow robot capable of efficiently clearing simulated snow from an enclosed arena marked by a black boundary. The robot will operate entirely on its own, navigating the area using a combination of sensors, control algorithms, and a plow mechanism designed to detect boundaries, identify, and avoid fixed and moving obstacles, pushing snow outside the perimeter.

Our design focuses on modularity, precision, and reliability. Each subsystem will be developed and tested independently to ensure reliable performance, system integration, and longevity. The snowplow will be built to follow strict size, speed, and time constraints, allowing for safe and accurate operation.

The project will be carried out using a structured engineering design process with clear goals and regular testing at each stage. Our team will work together to plan, build, and test each component, making adjustments as needed based on results. By managing the work efficiently and documenting our progress, we aim to deliver a robot that meets all project requirements and performs reliably.

Overall Deliverables:

The deliverables are coded and described in brief and concrete language in order to plan them in the WBS, construct them by the owning subgroup, and evaluate them against special evidence of acceptance. The set encompasses all the Functional and Non-Functional requirements, such as the ability to move around with the assistance of an encoder, IMU-based orientation, boundary detection and obstacle sensing and avoidance, a purpose-specific plow, supervisory control, and the documentation and repository artifacts that document function handlers and test logs. Every item has a clear definition of done and direct connection to the requirement IDs that it fulfills and the tests that will confirm this.

Ownership of the subsystems and the flow of integration are grouped. Early deliverables generate separately testable handlers and hardware which can pass unit tests prior to any system interconnection. Subsequent deliverables are concerned with incremental integration, complete system testing within the arena limits and performance optimization to reduce penalties. Last deliverables include code freeze, dimensional and speed tests, demo script, and preparation checks on the in-lab demonstration on the appointed date.

The approval of every deliverable is based on short yet objective evidence. Hardware products give pictures or illustrations with measurements against size constraints. Software items also contain links to repository, commit hashes and short unit or integration test logs and pass or fail results. Integrated items give a checklist of the requirement IDs that have been addressed, the latest test outcomes, and possible corrective measures that have been done. This makes the traceability apparent between requirement and deliverable to verification and makes it easy to be marked against the rubric.

Deliverable No.		Deliverable name	Description
D1	week1	Motor System	Have a functional motor system which can adjust its speed using the wheel encoders.
D2	week2	IMU Integration	Having an IMU sensor on board to track the orientation and correcting the direction of the robot during navigation.
D3	week3	Line Follower Sensor Integration	Have a system to follow the black line around the snow pile.
D4	week4	Obstacle Detection System	Ultrasonic, ToF Distance, and IMU Sensors for dating obstacles in front of the robot.
D5	week5	Snow Plowing Attachment	Attach a 3d printed shovel to the robot
D6	week6	Integration	Ensure all the sensors work with each other
D7	Week 7	Testing	
D8	Week 8	Code Finalization and demo	Review code check for requirement fulfilment for the system and demonstrate the completed project
D9	Week 9(lab 11)		

Table 1: Project Deliverable timeline

Scope:

This project provides an autonomous snowplow robot which removes simulation snow from an indoor arena bounded by an enclosed black path. The arena consists of fixed obstacles and moving obstacles and the robot has to perform the task without colliding with obstacles. Snow is represented by light-weight wooden cubes. The testing space is about 6 m2, and the operations are performed according to the size, speed, and time that are actually controlling the design. The maximum robot envelope is 226 * 262 * 150 mm with a plow extension allowed of up to 25 mm on each side in widths and 30 mm in length. The maximum allowable speed is at 30 cm per second and the maximum operation time is 5 minutes from start to finish. The demonstration begins from one corner of the inside of the perimeter edge of one edge. These constraints provide the physical and operational limitations used in defining the design and verification of the system.

Performance is measured by the amount of cubes pushed out of the arena accounting for penalties. Penalties are size violations, obstacle contacts, boundary excursions - more than 5 cm, touching the robot to adjust or reset and overspeed events. Due to the high penalty that these errors translate into direct score hits, careful sensing, control and mechanical design becomes necessary for success.

The system will be built as a collection of modular subsystems which include mobility with motor control and encoders, orientation with an IMU, boundary sensing with line sensors, obstacle detection/avoidance, and a purpose designed plow that directs cubes to the perimeter. Each sensor handler will be individually developed and unit tested, checked into a common store, and checked against clear success criteria. Integration is done in levels of planned integration, we could use motors with IMU first, then line sensing, then obstacle detection and finally the plow with measurable testing after each addition. The lab schedule allows for several sessions for unit level development and the rest for full system integration and testing to ensure a stable demonstration.

Unit and integration testing plan:

Test ID	Unit/Test	Test Description	Pass Criteria	Fail criteria
Ut1	Motor control	Run motor with varions pwm and measure the output speed	Motor speed, 30cm/s 5%	Unstable at constant pwm speed deviates >5% from target.
Ut2	Motor direction control	Command robot to move forward, reverse, turn left/right	Correct direction response without delay	Incorrect direction or no movement delayed (>0.5s)
Ut3	IMU Sensor	Read orientation data during movement	roll/pitch/yaw stable and within 3%accuracy	Fluctuations >3%drifts,or inconsistent readings.
Ut4	Line Sensor	Detect the black line boundary	Line correctly detected in >95% of trials	Detection accuracy <95% or frequent false positives /negatives.
Ut5	Obstacle Detection Sensor	Detect stationary object at different distances	Object detected within +or - 5cm distance	Missed detection or distance error > 5cm
Ut6	Moving obstacle detection sensor	Detect moving objects crossing robot path	Object detected within 0.3 s of motion entry	Detection delay > 0.3 s or failure to detect motion
Ut7	Plow Servo Control	Move plow up down using command	Servo moves to commanded position in +-5in <1s	Position error >5 or motion delays 1s.
Ut8	Power Monitoring	Measure current and voltage during operation	Voltage and current reading within +-5%of multimeter reading.	Deviation >5% or sensor dropouts.

Ut9	Error Handling /Diagnostics	Trigger invalid input to each handler	Error detected and logged safe state are maintained.	No error flag system crash or unsafe behaviour.

Table 2: Project testing approach

The testing plan validates the functionality accuracy and reliability of the robot's Key subsystems. Each of the unit tests targets a specific function of the robot, such as the motors or the sensors. There are also integration tests there to make sure each of the functionalities work well with each other, such as obstacle detection and moving obstacle detection.Sensor performance is tested using the UT3 - UT6.

List of Requirements:

Category	Functional/Non-Functional	Engineering requirements
Physical Dimensions	Non-Functional	● The robot shall not exceed the dimensions of 226 mm x 262 mm x 150 mm without including the length of the plow extension of 30 mm in width and 25 mm in width.
Speed of Robot	Non-Functional	● The robot shall not exceed a maximum speed of 0.3 m/s.
Boundary Detection & Navigation	Functional	● The robot shall employ a line-following sensor system. ● The robot shall navigate within an enclosed arena autonomously.
Obstacle Avoidance	Functional	● The robot shall detect obstacles without colliding into them.
Autonomy & Reliability	Functional/Non-Functional	● All sensors shall provide feedback for navigation with function handlers.(Non-Functional) ● The robot shall be fully autonomous.(Functional)
Performance	Non-Functional	● The Robot shall be able to recover from trajectory errors caused by obstacles without human intervention.

Table 3: Functional and Non-Functional requirements

This list of requirements highlights both the functional and non-functional specifications and requirements for the robots design. For the functional requirements, the robot must be able to navigate within a boundary using a line follower system, detect and avoid obstacles and maintain sensor feedback for navigation. The non functional requirements include the performance and physical constraints of the robot. The robot has to adhere to specific dimensions and should not exceed a speed of 0.3 m/s and should recover from trajectory deviations.

WBS:

1.0 Mobility Subsystem

- **1.1 Motor drivers & wiring** - mounted, powered, polarity correct.
- **1.2 Encoder speed control** - closed-loop speed with max-speed limit.

2.0 IMU Navigation Subsystem

- **2.1 IMU driver & calibration** - bias and heading available.
- **2.2 Heading hold** - drift correction API for straight travel/turns.

3.0 Boundary Detection Subsystem

- **3.1 Line sensor array** - installed and aligned.
- **3.2 Line tracking controller** - follows black boundary reliably.

4.0 Obstacle Sensing and Avoidance

- **4.1 Proximity sensor module** - ToF/ultrasonic integrated and filtered.
- **4.2 Avoidance logic** - maneuvers that avoid hits and displacement.

5.0 Plow Mechanism

- **5.1 CAD, mount, and clearance** - within size allowance.
- **5.2 Plow efficiency proof** - pushes cubes outside the perimeter.

6.0 Supervisory Control

- **6.1 Mission state machine** - start-corner, modes, transitions.

- **6.2 Limits & recovery** - speed/time enforcement and penalty minimization.

7.0 System Integration Builds

- **7.1 L1 Motors+IMU → 7.2 +Boundary → 7.3 +Obstacle → 7.4 +Plow**.

8.0 Verification and Validation

- **8.1 Unit tests (UT-01…UT-06)** - pass/fail logs.
- **8.2 Integration tests (IT-01…IT-04)** - staged proofs.
- **8.3 System trials** - full-run results with penalties noted.

9.0 Documentation and Repository

- **9.1 Handlers & code in repo** - tagged commits.
- **9.2 Test logs & evidence** - brief acceptance snippets.
- **9.3 Demo script & checklist** - code freeze, readiness checks

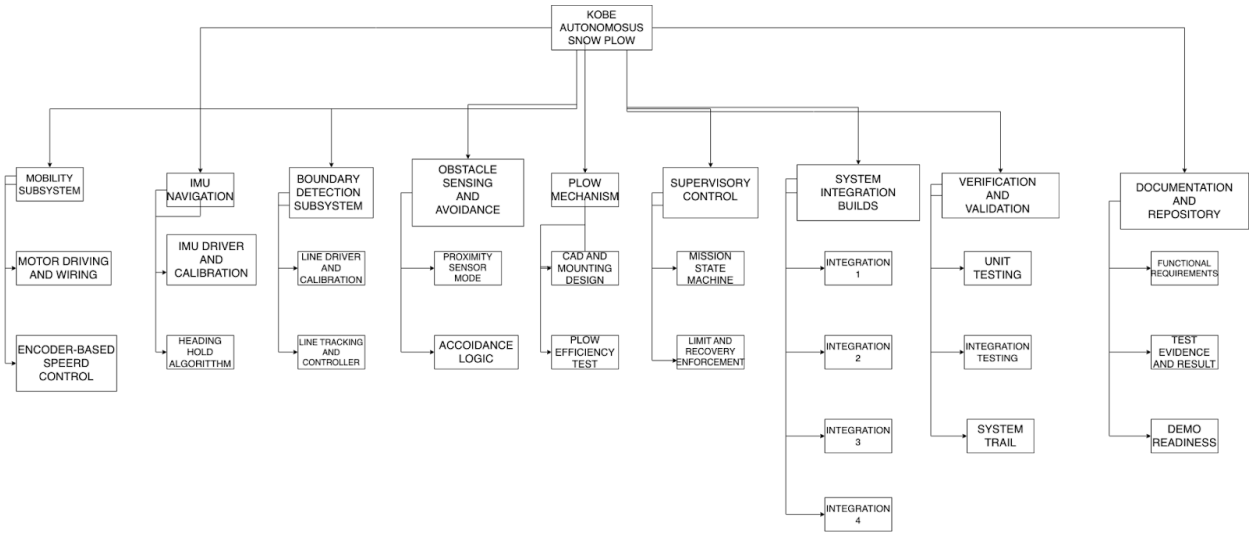


Figure 1: Work Breakdown Structure (WBS)

Schedule Network Diagram:

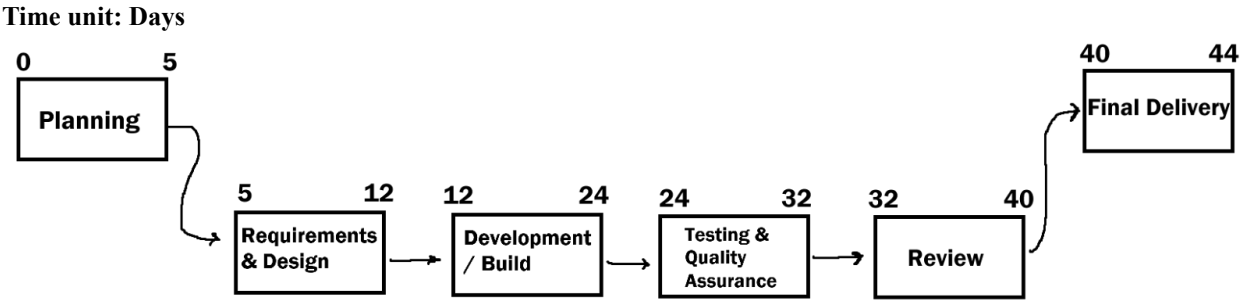


Figure 2: Schedule Network Diagram

Gantt Chart:

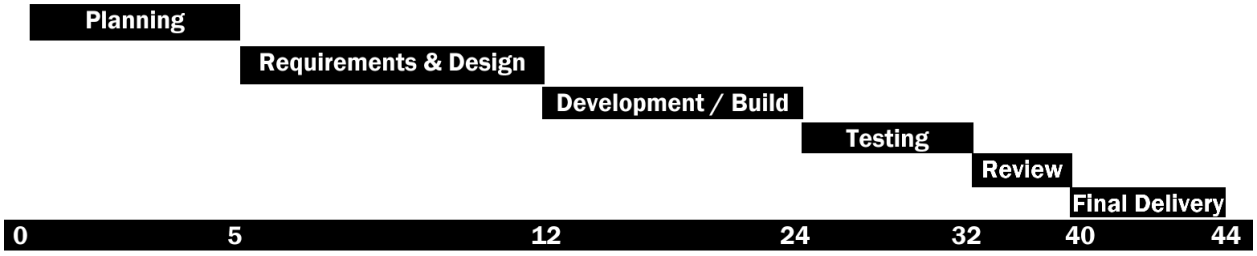


Figure 3: Gantt Chart for the project

The schedule network diagram and the Gantt Chart shows the visual representation of the projects sequential workflow. Starting off with the planning stage, where objectives and goals are discussed. The requirements and design phase contains finalizing the functional and non functional requirements. During the development phase the main components of the robot are developed, such as the sensors and the motors. The testing phase involves all of the components working well with each other and the review phase of the project is about ensuring that all the goals and requirements are met.

Cost:

Phase	Activity	Estimated Hours	Devs	Cost ($50/hr)	Description
1. Planning & Design	Requirement Analysis	6	4	$1,200	Defining the Functional, non functional requirements and the constraints.
	System Design	4	1	$200	Developing the high level block diagrams
	Plowing Mechanism	6	1	$300	Developing a model and mounting it on the robot for the plow.
2. Component Development	Motor Control Function	8	1	$400	Developing the motor functions for the wheel of the robot.
	IMU integration	8	1	$400	Integrate IMU sensors for the orientation and drift correction
	Line Detection	8	1	$400	Develop a line following algorithm using IR sensor
	Obstacle Detection	8	1	$400	Using Ultrasonic Sensor implement obstacle detection
	Plow Control	6	1	$300	Develop a control mechanism for

					the plow
3. Integration and testing	Motor Control+ IMU	6	1	$300	Have the motor control integrated with imu to have the robot moving in the desired direction
	Line + Obstacle Detection	6	1	$300	Ensure smooth coordination between line and obstacle detection
	Full System Integration	8	4	$1600	Make sure the entire system world well together

Table 4: Cost breakdown of the project

The total estimated cost for this project is $6,100 given that each developer gets $50/hr and the kit costs $300. The cost distribution also demonstrates a structured approach towards the planning of this project. The component development is the most important part of the project as it ensures each sensor and part is working well on its own. The final part is the testing and integration part of this project ensures that all of the modules work well with each other. Overall, the cost allocation of the project displays the project's goals of being a fully functional robot plow.

Human resources:

Activity	Dearell	Tobiloba	Partha	Ahmad
Requirements	Responsible	Responsible		
Planning	Responsible		Responsible	
Risk	Responsible			Responsible
Cost	Responsible		Responsible	
IMU		Responsible		Responsible
Mechanical			Responsible	Responsible
Sensors			Responsible	Responsible
Logic		Responsible		Responsible
Obsticle Detection				
Integration	Responsible	Responsible	Responsible	Responsible
Testing	Responsible	Responsible		
Documentation	Responsible	Responsible	Responsible	Responsible

Table 5: Project Human Resources

Conclusion:

This project aims to build a fully autonomous, modular, and reliable snowplow capable of operating under strict physical and time constraints.

Upon completion, the robot will demonstrate the team's ability to design, build, and program an autonomous platform that can perform basic snow clearing under controlled conditions. The project will showcase practical skills in sensor integration, control logic, and system testing. The final result will provide a solid foundation for learning, demonstrating core robotics concepts, and meeting the course requirements.

Progress Report

Overall Architecture of the System:

1. Power and Control:
    - Arduino Due to power the sensors
    - DC Battery Supply to power the motors and the Arduino
2. Mobility System:
    - DC motors with motor drivers
    - Motor function handler for direction and PWM control
3. Orientation Subsystem:
    - IMU sensor for pitch, roll and yaw tracking
4. Boundary Detection:
    - 2 IR line follower sensor (one in front and one at the back)
    - Upon detection of the boundary the system redirects itself.
5. Obstacle avoidance:
    - Ultrasonic sensors read object infront
    - Changes direction if there is an obstacle
6. Plow Mechanism:
    - A 3D printed plow that will push obstacles in front.

State Chart:

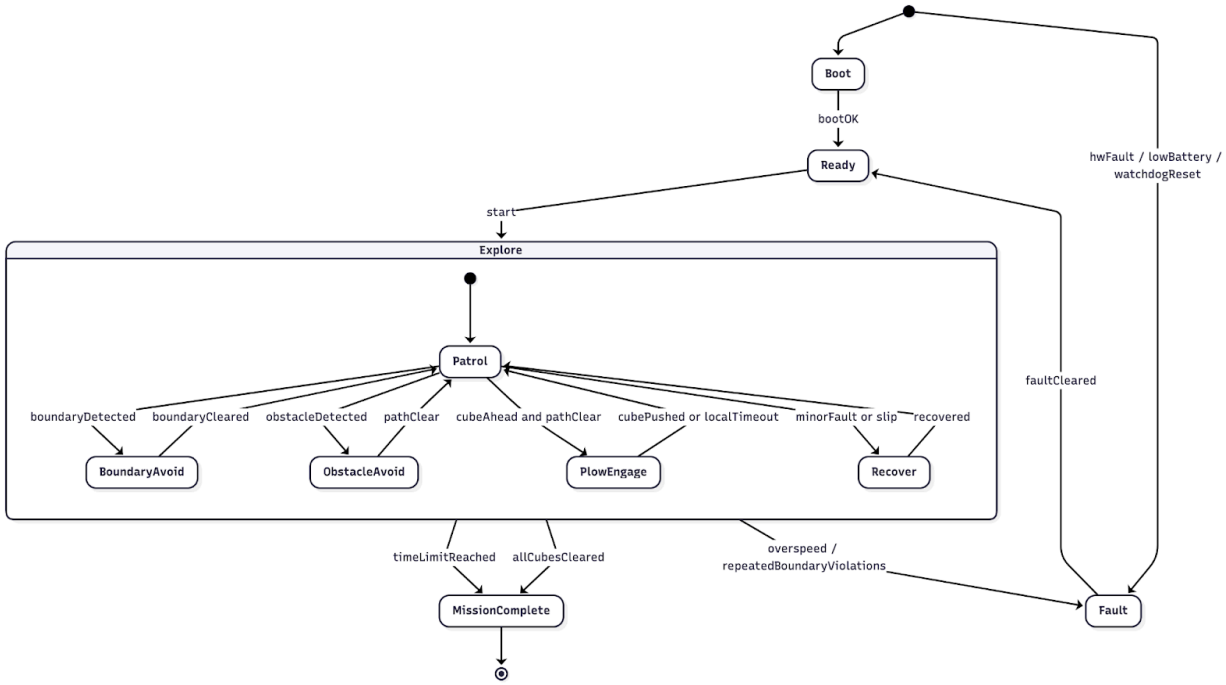


Figure 4: Start Chart Diagram for the project

The state chart is based on the lifetime of the robot since the booting up to the end of the robot, and fault distribution. Boot system goes into the Ready state and Explore is then started. Explore has the following operational states: Patrol is the default state, BoundaryAvoid goes inside when the boundary is found, ObstacleAvoid avoids the obstacles that are found and PlowEngage pushes the cubes when the path is free. Recover is used in cases of small slip or trajectory error and it goes back to Patrol after correction. The MissionComplete state happens when one of these two things happen: the time is over or all the cubes have been cleared. If there is any failure detected in the hardware, watchdog reset or repeated violation, this would cause the system to enter Fault state and the motors would stop, and return to the Ready state.

Sequence Diagram:

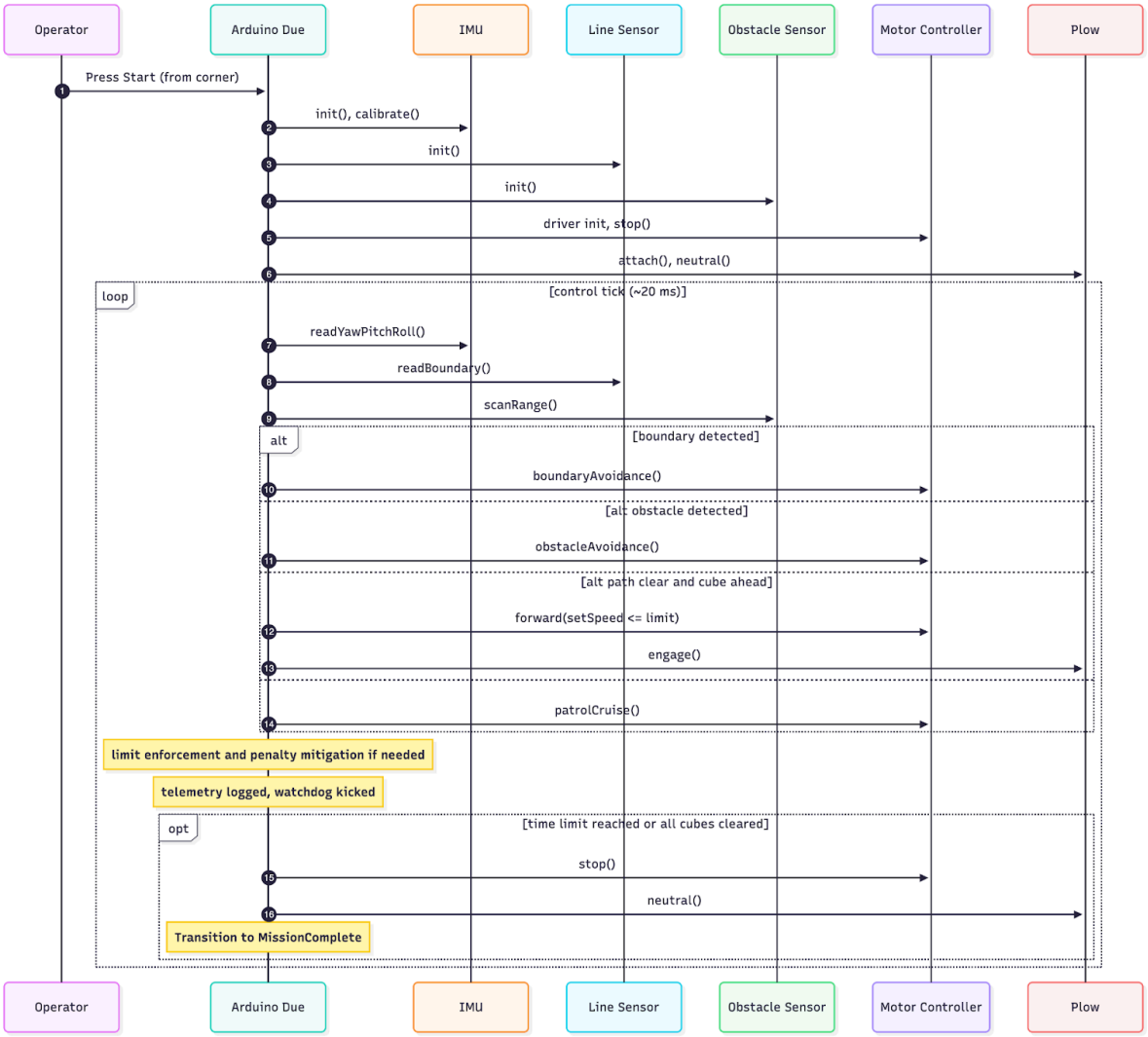


Figure 5: Sequence diagram for the project

The sequence diagram follows a single control cycle of start to stop. The operator triggers the running, the MCU triggers the IMU, line sensors, obstacle sensors, motor driver and plow before entering a control period. Per MCU reading of IMU orientation, boundary status, obstacle ranges on every tick and making the decision of either contacting with the boundary or obstacles,

plowing or normal patrol, subject to speed and time limits. Each tick Telemetry is recorded and the watchdog is kicked. Upon clearing all cubes or time elapsing, the MCU will shut off the motors, put the plow in a neutral position and write a run stat and go to MissionComplete.

Demonstrate the use of a Watchdog timer:

```
#include <avr/wdt.h> // Include watchdog timer library
```

Figure 6: Watchdog Timer Library

```
//Enable Watchdog Timer (2 seconds)
wdt_enable(WDTO_2S);
Serial.println("Watchdog Timer Enabled (2s)");
```

Figure 7: Enabling the watchdog timer.

Figure 4 shows that the watchdog timer library is included in the code, and Figure 5 shows the watchdog timer being enabled in the code. What this does is whenever the loop takes too long to run, it will call the watchdog reset and restart the setup process.

Updated Planned Value Analysis Table:

Week	**Deliverables**	**Planned hours**	**Planned Cost**	**Total Planned Cost**
1	Motor System	8 hrs	$400	$400
2	Line Following Sensor	8 hrs	$400	$800
3	Obstacle Detection Sensor	8 hrs	$400	$1200
4	Plow integration	6 hrs	$300	$1500
5	IMU Integration	8 hrs	$400	$1900
6	System Integration	10 hrs	$1000	$2900
7	Integration Test	10 hrs	$1000	$3900
8	System Test	10 hrs	$1000	$4900
9	Final Documentation	5 hrs	$600	$5400

Table 6: Updated Planned Value Analysis

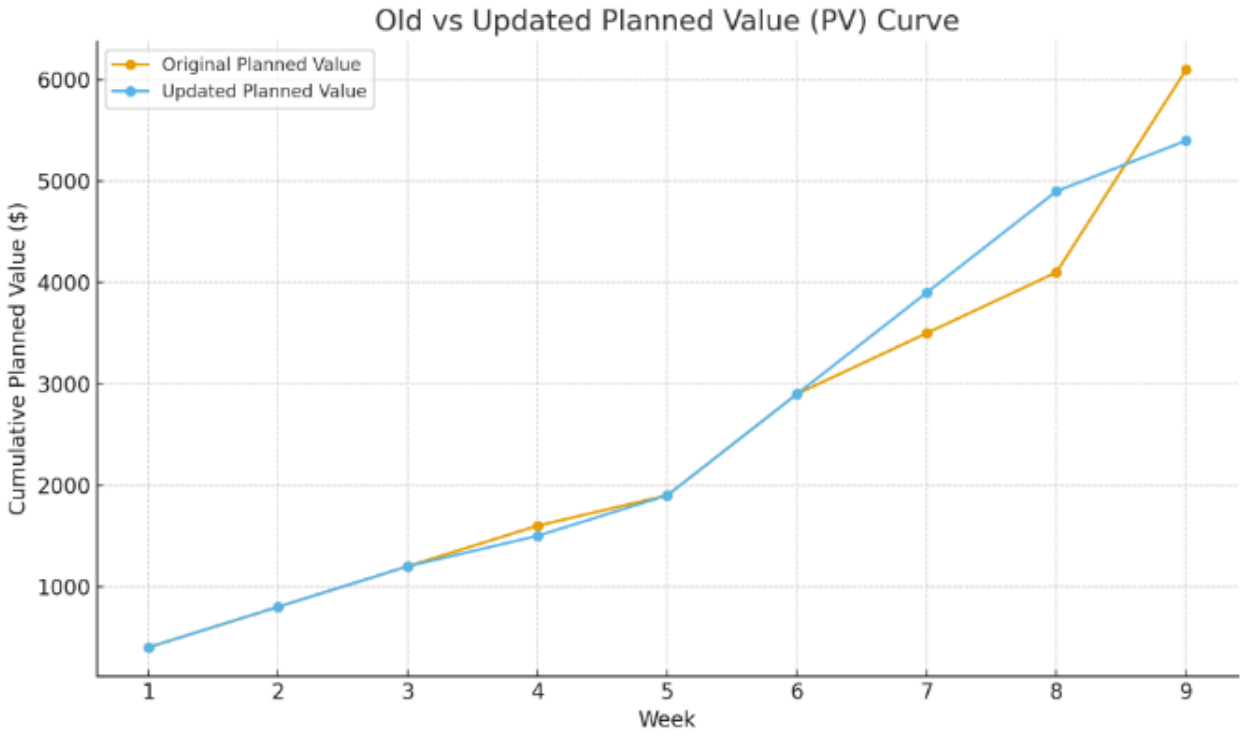


Figure 8: Original Planned value vs updated planned value comparison

The figure above shows the planned value curve and how the work has been distributed over the duration of the semester. The results show that the original planned value is higher than what we should be and we are under budget for this project.

Github Code:
https://github.com/Partha619/SYSC4805

Results of Unit Tests:

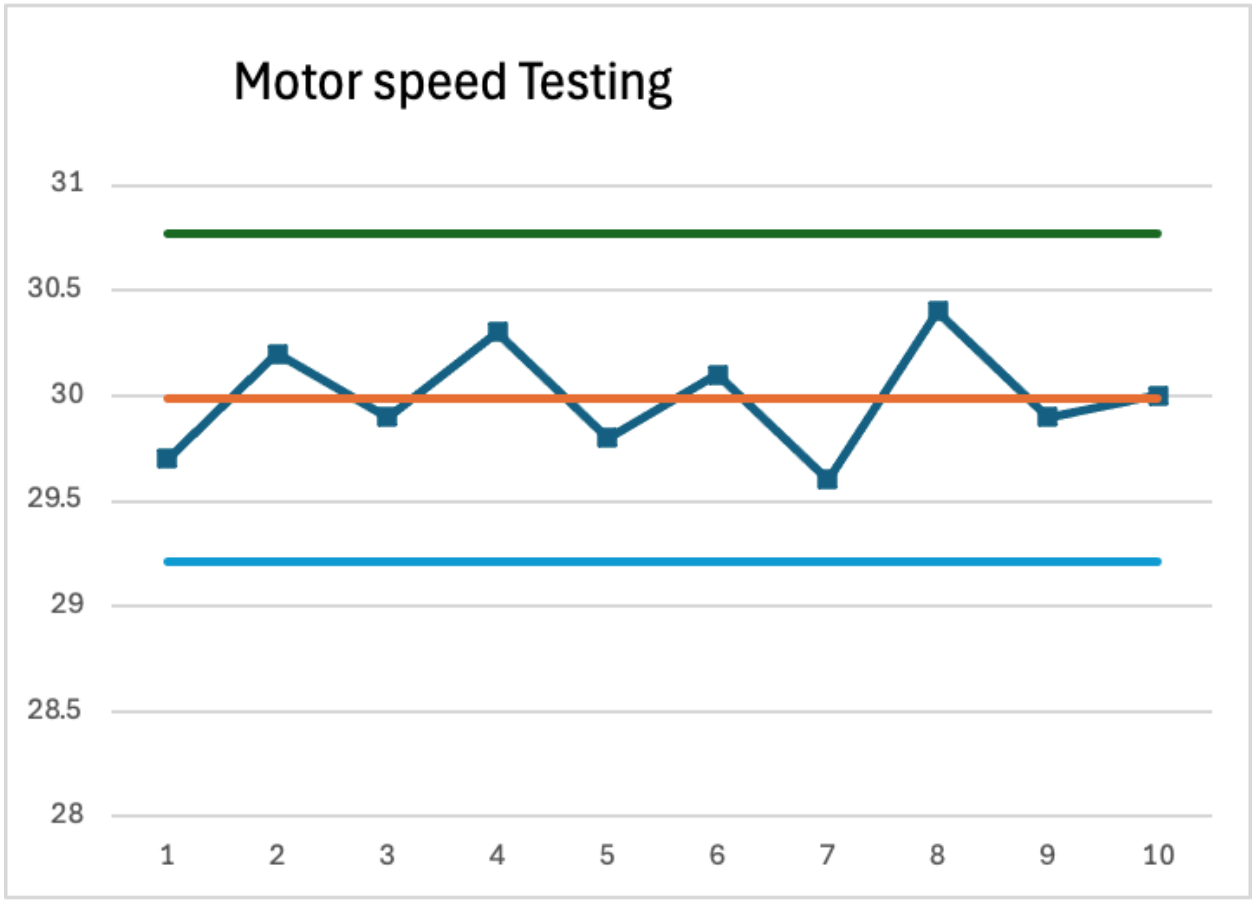
Unit test results for Line Sensor Follower:

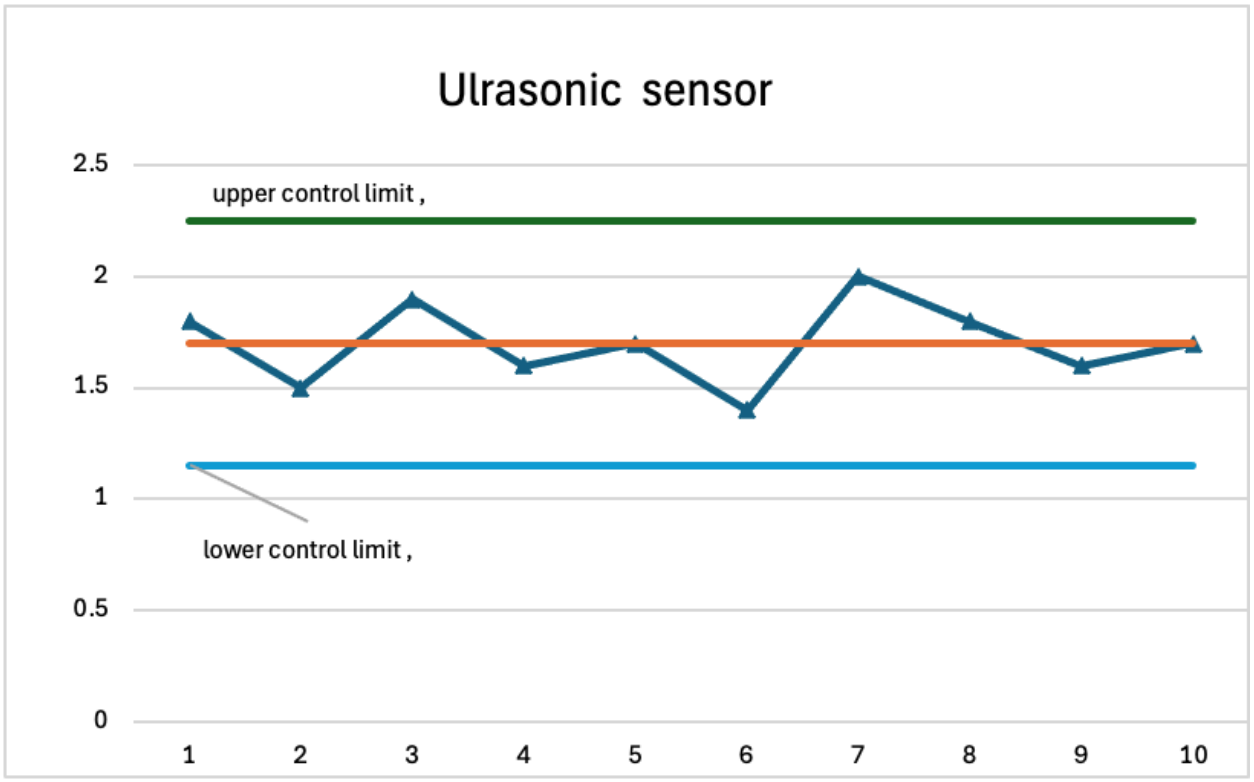
```
PASS: All sensors high | Expected: Stop | Actual: Stop
PASS: Left sensor low | Expected: Forward | Actual: Forward
PASS: Middle sensor low | Expected: Forward | Actual: Forward
PASS: Right sensor low | Expected: Forward | Actual: Forward
PASS: Left+Middle sensors low | Expected: Forward | Actual: Forward
PASS: All sensors low | Expected: Forward | Actual: Forward
```

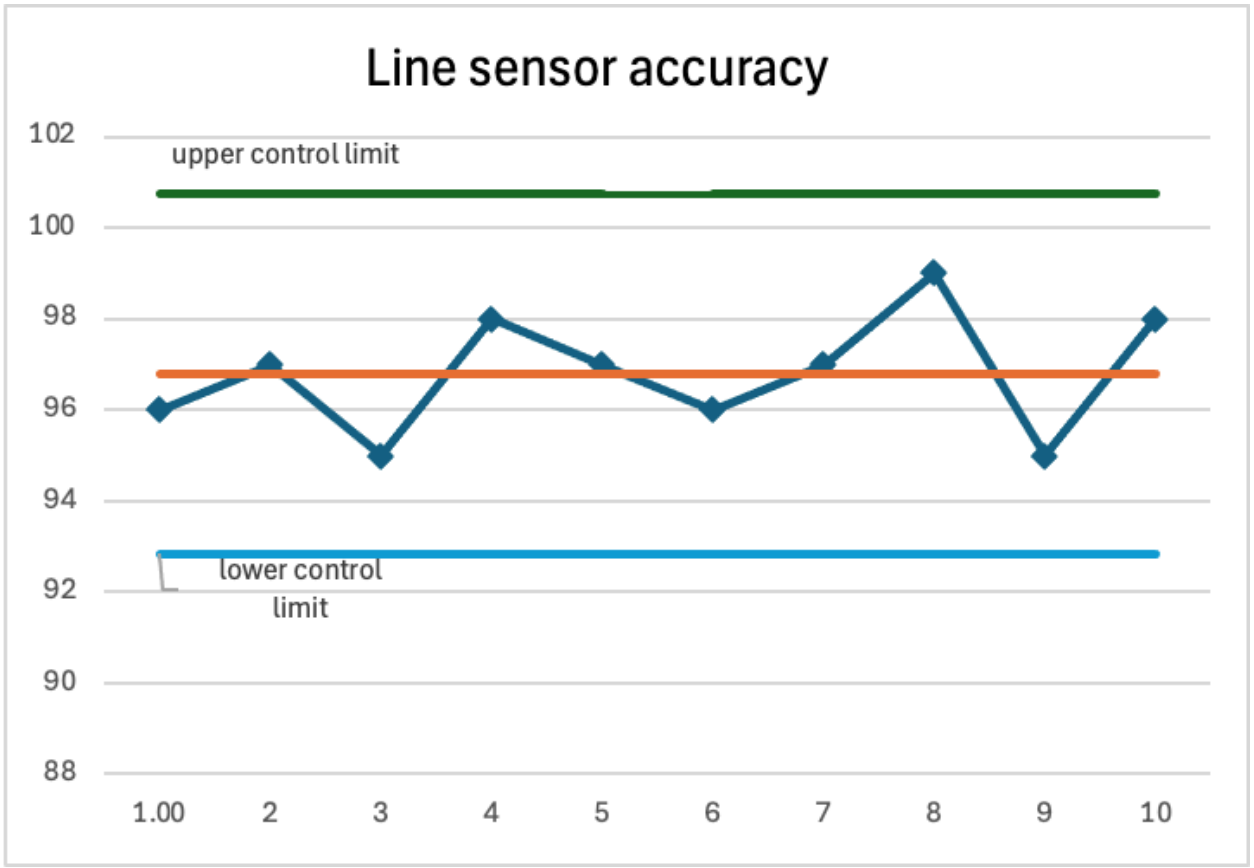
Unit test Results for Distance sensor:

```
PASS: Distance within range | Expected: Clear | Actual: Clear
PASS: Distance below minimum | Expected: Obstacle Detected | Actual: Obstacle Detected
PASS: Distance above maximum | Expected: Obstacle Detected | Actual: Obstacle Detected
PASS: Distance at minimum boundary | Expected: Clear | Actual: Clear
PASS: Distance at maximum boundary | Expected: Clear | Actual: Clear
```

Control Charts:




Ulrasonic sensor
2.5
2
1.5
1
0.5
0
upper control limit ,
lower control limit ,
1
2
3
4
5
6
7
8
9
10


Line sensor accuracy
102
100
98
96
94
92
90
88
upper control limit
lower control limit
1.00
2
3
4
5
6
7
8
9
10

Results of System Integration Testing:

The system integration testing was performed primarily in the Labs 9 and 10 when we had concluded on a simplified sensor set-up which comprised of three central sensors, including an ultrasonic distance sensor, a line-following sensor, and a ultrasonic distance sensor. The line-following sensor was mounted on the bottom of the plough in a way that was very close to the ground, allowing it to detect the black boundary lines on which the robot was not allowed to cross. The ultrasonic sensors have been mounted on the top of the robot in front, on the right side and on the left side, to scan all the frontal area for potential obstacles. In addition to this, a cardboard body kit was fitted on the sides of the robot so that snow and other obstacles do not get in the wheels of the robot.

After running several tests, we concluded that the selected sensor position and control logic were functioning as desired. The line sensor was capable of continually detecting the line of the boundary at various approach angles, and therefore, the robot could stay within the permitted space as it moved forward. The ultrasonic sensors could pick up the obstacles in front of the robot and when an obstacle was detected, the robot would stop and then turn the opposite direction where the obstacle was being detected (For example: when the right ultrasonic was detecting an obstacle, the robot would stop, reverse then turn away in the opposite direction of the obstacle then proceed as normal). A series of tests was used to fine-tune the tilt of the ultrasonic sensors so that they would not be too sensitive to the floor or the objects at a great distance but still be able to detect the relevant obstacles.

During the first complete run in Lab 11, we observed that one of the ultrasonic sensors was tilted slightly downwards and one of the wires of the sensor occasionally flew in front of the sensor, causing the sensor to detect snow as an obstacle and the obstructing wires caused the robot to stop and make avoidance manoeuvres on objects that were not there. During the second run, we corrected the sensor angle and improved the wire management, so that the sensor can have a clear field of view. The second run resolved the following issues: the snow was no longer considered an obstacle to the robot, it could maintain its position within the boundaries with the aid of the line sensor, and could carry out obstacle avoidance as expected. Overall, this system integration test indicated that the simplified three-sensor set-up, along with the mechanical body kit, could be applied to achieve effective boundary detection and obstacle avoidance in the final environment.

Github Code:

https://github.com/Partha619/SYSC4805