SYSC 4805

Final Report

Group L2-7 Byzantium

Pranav Koduvely Hari, 101144482
Christopher Semaan, 101140813
Sagar Syal, 101150341

Due Date: Dec 8th, 2023

Contents

Project Charter

Overall Objective

The project's objective of our team, Byzantium, is to design and develop an autonomous robot tailored for snow removal within a designated area. The robot will be able to traverse an area will be enclosed by a black boundary and obstacles to avoid. Within the field boundaries, the robot will be able to maneuver the obstacles by being able to change speeds and make turns. Our robot will be tasked with effectively clearing simulated snow using a custom designed snow plow installed on the robot chassis. The autonomous snow plow will be able to maneuver using data from several real-time distance sensors and line following sensor. The software solution will seek to take the real-time data to make all control modules for the wheel motor drivers. The real-time data will be measured and compared to known data to confirm accuracy during testing. The wheel motor modules will have unit tests to confirm functionality in isolation. In the end, we will look to integrate all sensors, modules, and chassis to create a cohesive autonomous robot that will complete the goal of clearing snow.

Overall Deliverables

Deliverable Name	**Deadline**
Project Proposal	Oct 20th
Progress Report	Nov 17th
Final Presentation	Nov 23rd
Lab Demo	Dec 5th
Final Project Report	Dec 8th

Scope

Requirements

1. The robot should not exceed a speed of 30 cm/s
2. The robot must be able to avoid collisions with the obstacles by stopping or turning
3. The robot must stay inside the boundary zone and avoid leaving by more than 5 cm
4. The robot should provide enough power to the sensors for all of them to operate.
5. The plow cannot extend by 25 mm in each direction and in length by 30 mm
6. The robot size must be less than 226 x 262 x 150 mm

Detailed Deliverables

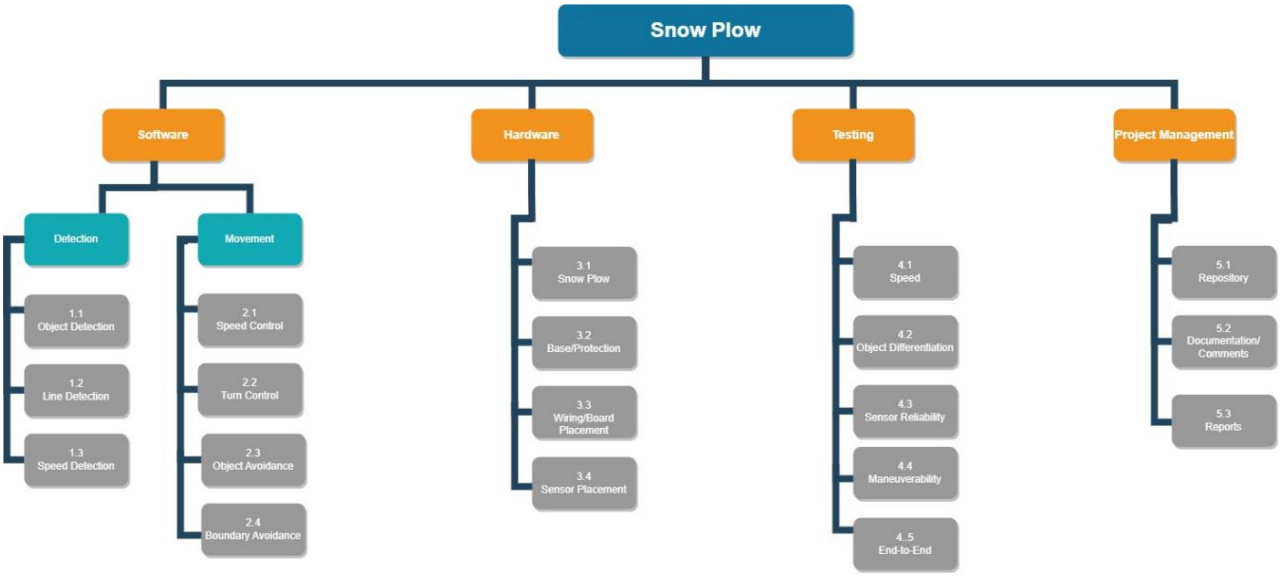

Figure 1 Work Breakdown Structure

Testing Plan

Table 1: Unit testing and Success Criteria

Tests	Success Criteria
Wheel Speed Test	Wheels are able to change speed to accommodate speed limit and complete turning capability
Distance Sensors Test	All distance sensors (Ultrasonic, Analog, ToF) are able to detect distance within a set threshold of a measured value. IR sensor is able to detect the difference between obstacle directly in front or not
IMU Test	The IMU will be able to measure acceleration and direction of robot accurately

Line Detection Test	Line follower sensor is able to detect the difference between flooring and the boundary line

Table 2: Integration and End-to-End testing and Success Criteria

Tests	Success criteria
Obstacle Detection	Obstacle is detected and vehicle stops completely without hitting the obstacle. Only avoiding obstacles and not the “snow”
Boundary Detection	Boundary line is detected and the vehicle/ wheel motors reverse back into bounds
Speed/Acceleration Control	Robot is able to accelerate and maintain legal speed when there is no obstacle in front
Maneuverability Control	Robot is able to change direction by only using 2 motors and rotating the directed amount.
Complete Performance Test	Robot is able to push snow out of the boundaries

Schedule

List of Activities

Software

- Sensor Functions
    - Implement object detection code using distance sensors including Ultrasonic, Time of Flight
    - Implement line follower code using line follower sensor for boundary detection
    - Implement speed monitoring code using encoder units to collect speed data
- Movement
    - Implement speed control code for the wheel motors using the encoder unit speed data
    - Implement turn control code for the wheel motors for maneuvering control
    - Implement object avoidance using object detection data and speed/turn control modules
    - Implement boundary avoidance code using line detection data and speed/turn control modules

Hardware

- Build and install snow plow on the robot chassis
- Install distance sensors and line follower sensor in appropriate directions for object avoidance and boundary avoidance
- Install full board and sensor wiring for all components on the chassis

Testing

- Test robot maneuverability including turning and speed control as a unit
- Test object detection by demonstrating ability to differentiate obstacle and goal
- Test sensor readability, reliability and accuracy for use in all units
- Test full robot implementation

Project Management

- Create GitHub repository for source code management
- Create documentation in the repository and in-code documentation for understanding
- Write Progress Report
- Write Final Report

Schedule Network Diagram

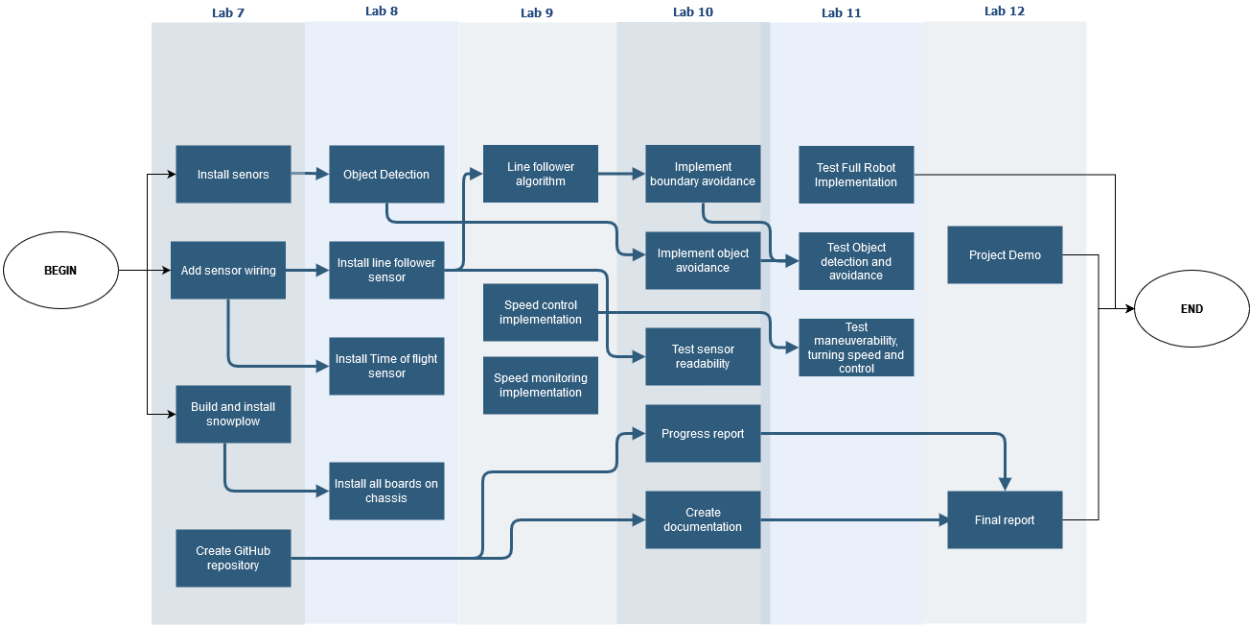

Figure 2 Schedule Network Diagram

Gantt Chart

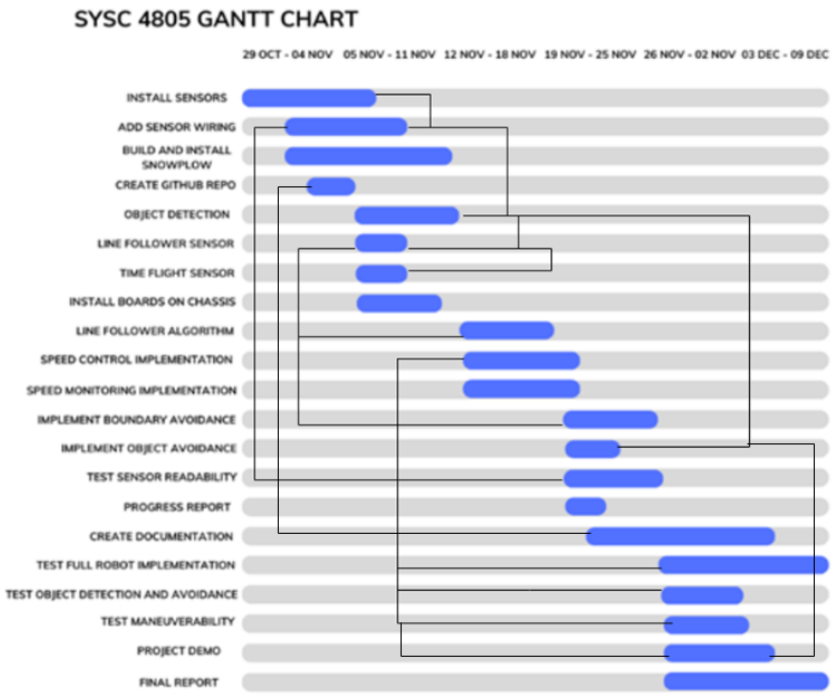

Figure 3 Gantt Chart

Cost

Hour Calculations

Project Start Day: Oct 17th, 2023.

Project Demonstration Date: Nov 23rd, 2023.

Total Duration: 5 weeks

Individual Hours per week: 4hr lab period + 2hr = 6 hr/week

Total Hours: No. weeks * Individual Hours/Week * No. Individuals

= 5 weeks * 6 hr/week * 3 individuals

= 90hrs

Cost Calculations

Hourly Rate = $50/hr

Robot Kit = $500

Total Cost = (50 $/hr * 90hr) + 500 = $5000

Weekly Cost Breakdown

Week No.	**Hours spent (hrs)**	**Total Cost ($CAD)**
Week 1	(4h lab + 2 outside of lab) * 3 members = 18	18hrs * $50/hr = 900
Week 2	18	900
Week 3	18	900
Week 4	18	900
Week 5	18	900
Totals	90 hrs	4500

Table 3: Weekly Cost Breakdown tab

Total expenses = Initial Kit Fee + Total of Weekly Rate = $500 + $4500 = $5000

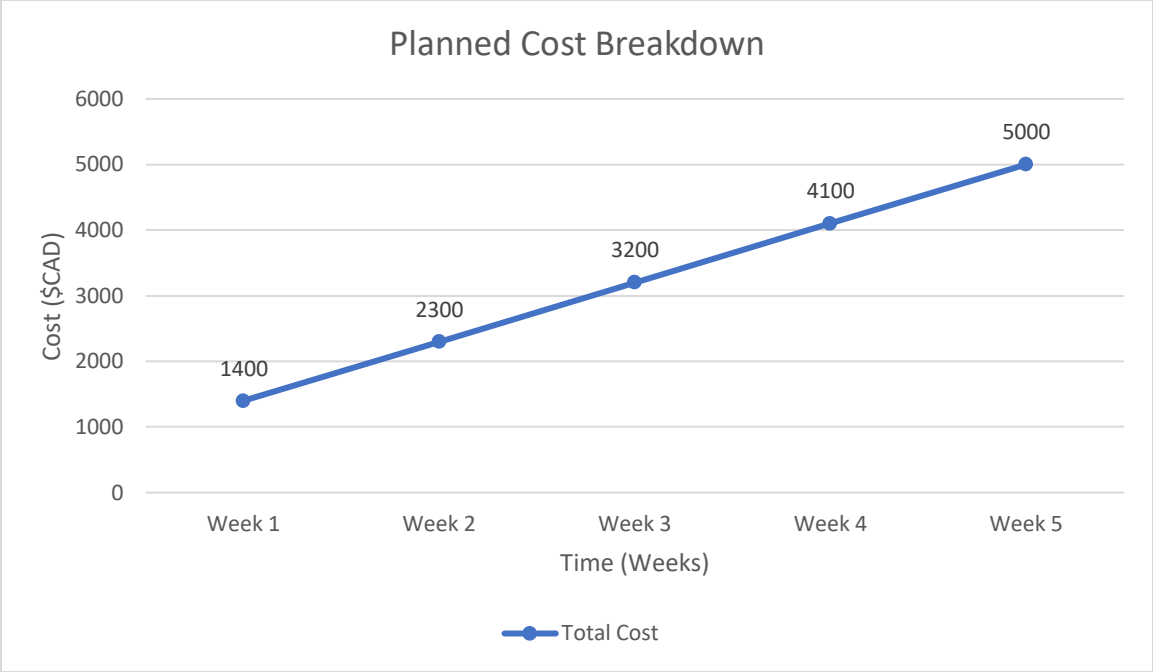

Figure 4 Cost Breakdown Chart

Planned Value Analysis

Period	Planned Value (PV)	Earned Value (EV)	Actual Cost (AC)
1	900	900	900
2	1800	1816	900
3	2700	2716	900
4	3600	3632	900
5	4500	4548	900

Table 4: Planned Cost Breakdown table

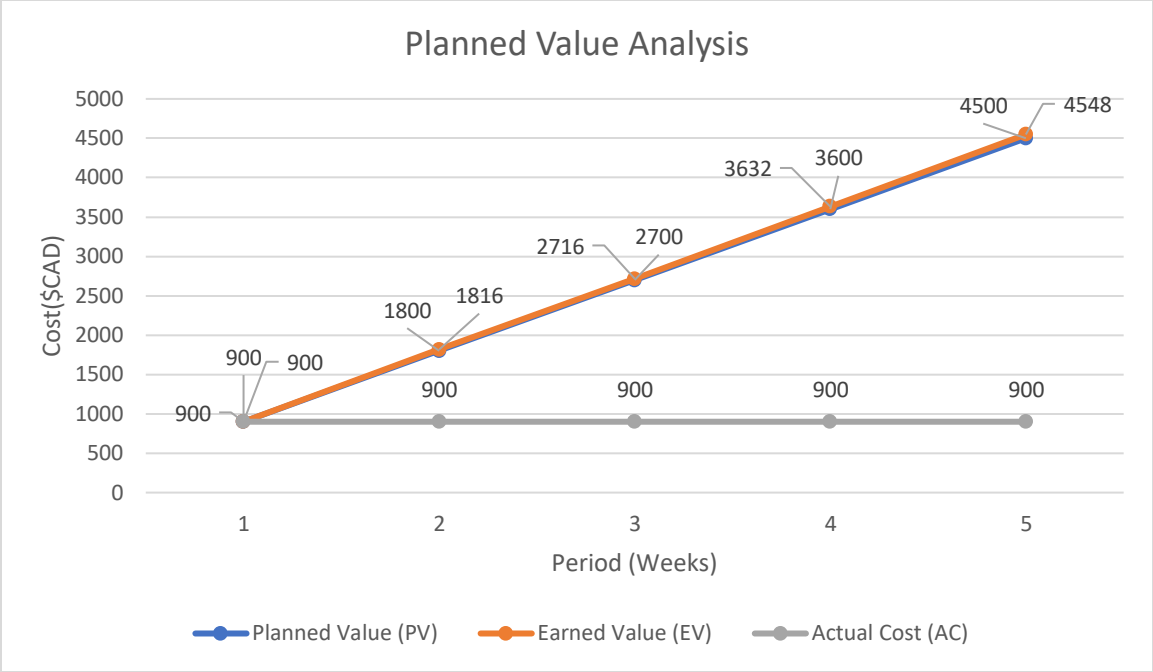

Figure 5: Planned value analysis

Actual Cost – The Sum of the costs for that period.

AC = 6 hr/week * 3 individuals * $50/hr = $900

Planned Value – The planned budget Allocated for that period.

Earned Value – The actual work completed by the end of the period.

Human Resources

Table 4: Responsibility Matrix (Responsible – R, Approver – A, Informed/consulted – I)

Activity	Pranav Koduvely Hari	Christopher Semaan	Sagar Syal
Create Git	**R**	**R**	**R**
Install Distance sensor	R	I	A
Install Full board and Wire sensors	I	A	R
Implement Object detection	A	R	I
Implement Object Avoidance	R	A	I
Implement speed Monitoring	I	R	A
Implement speed control	A	I	R
Implement Line follow	R	A	I
Write Progress report	**R**	**R**	**R**
Implement control code	I	R	A
Implement Boundary Avoidance	A	I	R
Build & install Snow-plow	**R**	**R**	**R**
Test maneuverability	R	I	A
Test Object Detection	A	R	I
Test Sensor readability	A	I	R
Full Robot implementation test	**R**	**R**	**R**
Create Documentation in the repository	**R**	**R**	**R**
Write Final Report	**R**	**R**	**R**

Note: We combined Informed and Consulted into 1 role since we only have 3 teammates

Overall Architecture

Hardware Design

Our microcontroller board (Arduino) is mounted on the top of the robot chassis as well as all the wiring that connect to the Arduino. The motor driver board is mounted underneath the top of the chassis and is directly wired to digital IO of the Arduino and to the motors of the wheels. On our robot chassis we decided to use three distance sensors along the top for object detection. There are two ToF sensors placed on the front corners pointed outward from the corners as well as one ultrasonic sensor pointing forward. For the boundary detection, we have the line follower sensor installed at the bottom of the front on the chassis. The final sensor is the IMU which will be used to differentiate turning range.

The ultrasonic sensor is wired up as digital IO signal on the Arduino to receive a digital value for range. The ToF sensors are both wired to the SDA and SCL on the Arduino as well as digital IO pins. Finally, the line follower sensor is wired to digital IO for output as well. The IMU will also use I2C and connect to SDA and SCL on the Arduino.

Our planned design for the snow plow is a 3D printed model that has less than 5 cm clearance from the ground and is mounted on the bottom rail of the chassis.

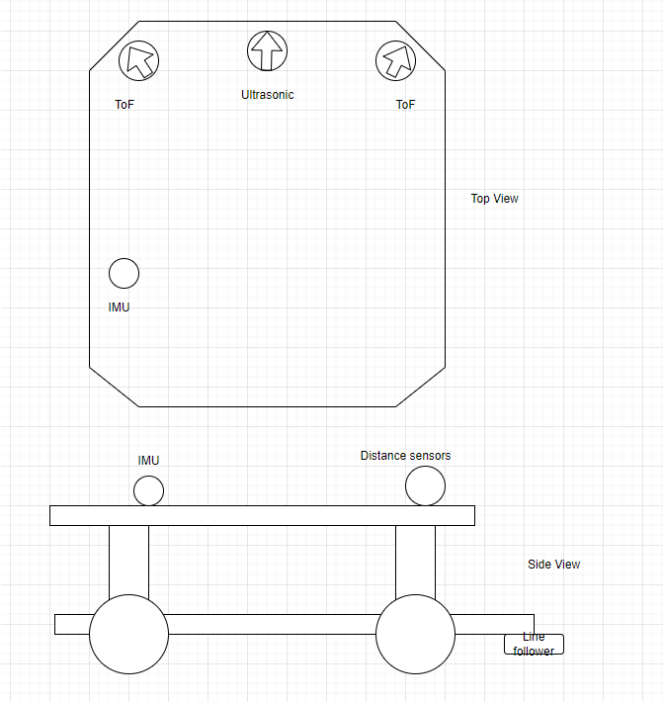

Figure 5 Diagram of sensor placement

Software Design

The current includes several different Arduino modules for the sensors installed on the robot. These modules contain simplified polling methods to read the sensors once. In the final implementation will have a combined implementation for all the sensors to be polled quickly in the main file.

The object detection and boundary detection will use the polling methods at the beginning of the algorithm loop. The object detection logic will poll the ultrasonic and ToF sensors methods. The object detection algorithm will check the direction of objects in the path before instructing the motors to move in the opposite direction. If there is an object detected on the left or right, the robot will turn the opposite to avoid collision. Turning is done by pushing the motors on one side forward while reversing the opposite side. If there is an object is directly in front, the robot will decide any direction to turn ahead of the time. We decided to randomize direction to avoid repetitive pathing. Randomized turning will lead to higher likelihood of accessing areas not reached in the boundaries previously. The IMU will be used to check turning range.

The boundary detection logic will need to poll just the line follower sensor methods. The boundary is detected once the line follower detects a high voltage for more than one poll to avoid noise. Once the line follower detects the black boundary, the robot will first reverse direction of all wheels. After reversing until the boundary no longer detected, the robot will turn in a random direction. We decided to limit the amount of reversing as we do not have a sensor on the rear of the robot. Similarly to the object detection when directly in front, the turning direction will be randomized to avoid repetitive pathing.

State Chart

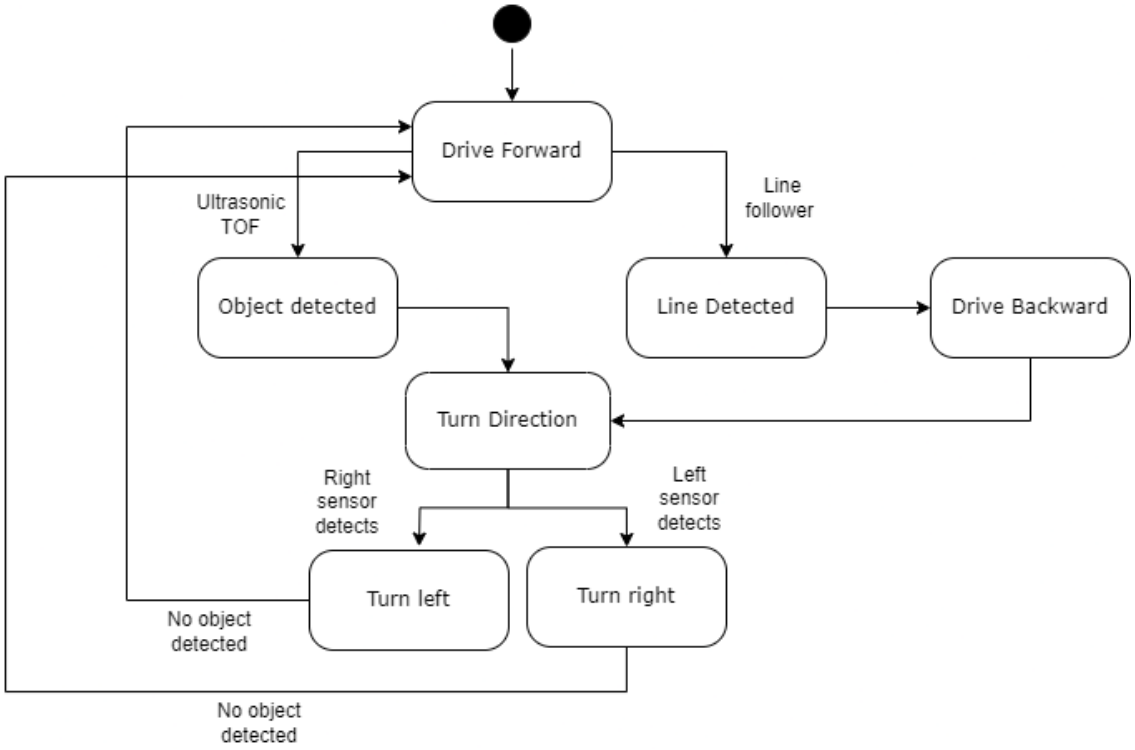

Figure 6: State Chart

The initial state of the robot is at rest. Once the robot starts to move, it is in the “drive forward” state as default. Sensors are polled continuously to check for objects to avoid and to ensure that the robot does not cross the boundary. If an object is detected by the ultrasonic or time-of-flight sensor, the robot stops and moves into the “turn direction” state where it determines which direction to turn in. The direction of the turn is based on the direction of the oncoming object. Similarly, if a black line is detected by the line follower, the robot will drive a certain distance backwards and a random turn radius will be selected. The cycle repeats until the obstacle course have been cleared.

Sequence Diagram

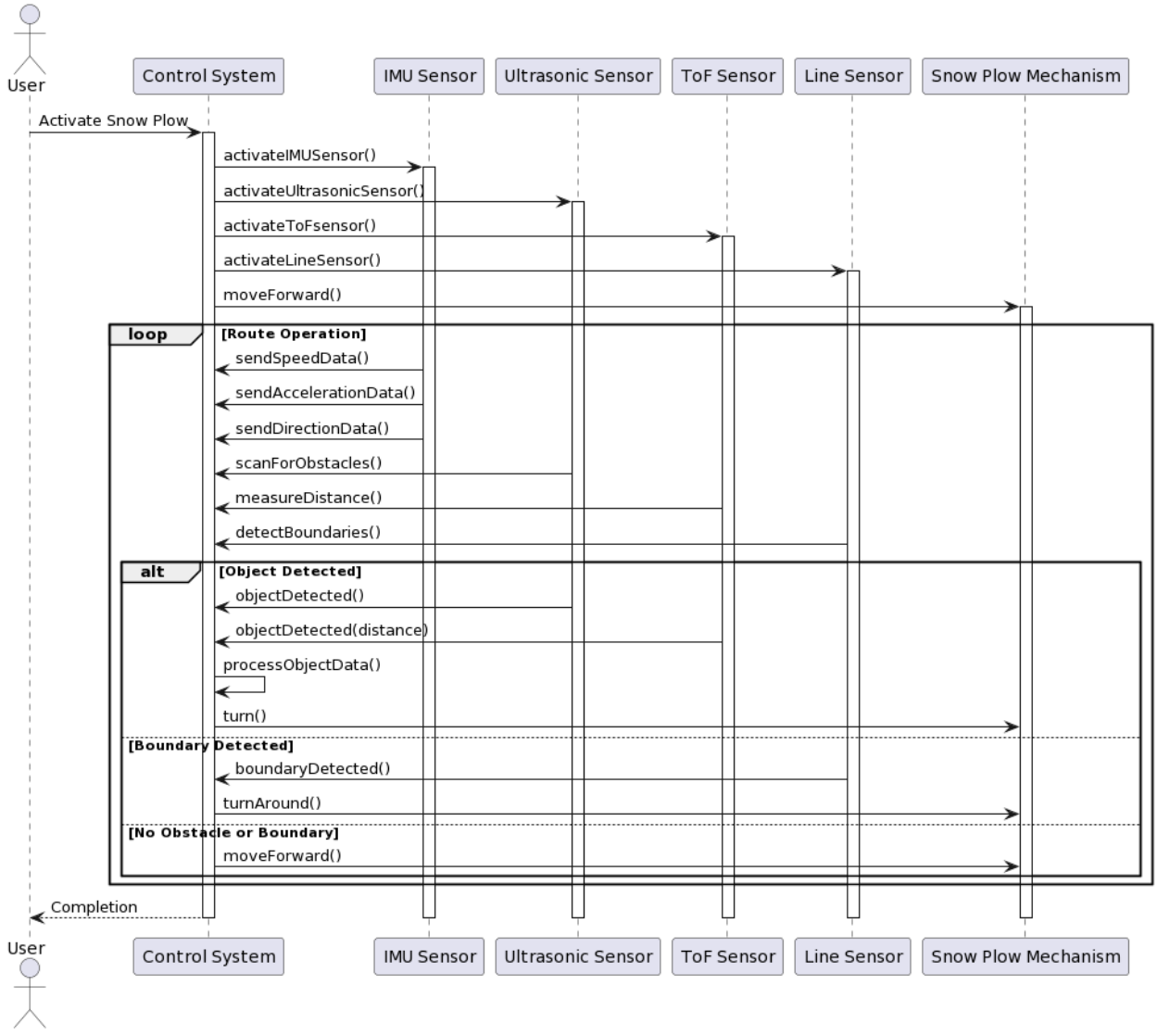

Figure 7: Sequence Diagram

Initially a user will activate the snow plow, which triggers the control system to initialize the ultrasonic sensor, time of flight (ToF) sensor, line sensor and the Inertial measurement unit (IMU) sensor. Upon activation, these sensors will start collecting data such as velocity, acceleration, direction as well as

simultaneously scan for obstacles and boundaries. After this initialization phase the snow plow will start its snow clearance by moving forward to clear the snow in front of it. When an object is detected by the ultrasonic or the ToF sensor the data is processed, and the motors adjusts its direction accordingly. Correspondingly if a boundary is detected by the snow plow's line sensor trigger the movement algorithm described in the previous section.

Watchdog Timer

The purpose of the watchdog timer is to hard stop anything that may cause an infinite loop to run. This will run in our main file and code will run in a loop for a predetermined time until it resets. This will ensure that the robot's functionality is correct. The watchdog timer will be implemented as a loop in the main file that will have a timeout of 2s, as indicated in figure 7 below.

```
void loop()
{
  WDT->WDT_CR = WDT_CR_KEY(WDT_KEY)  // Restart timer
              | = WDT_CR_WDRSTT;

  Serial.println("Restart watchdog");
  delay(500);

  while (true)
  {
    Serial.println("Deadlock!");
    delay(500);
  }
}
```

Figure 8: Watchdog timer implementation

GitHub

Each member of the group has contributed at least 3 commits to the repository as can be seen here.

Control Charts

Obstacle Avoidance

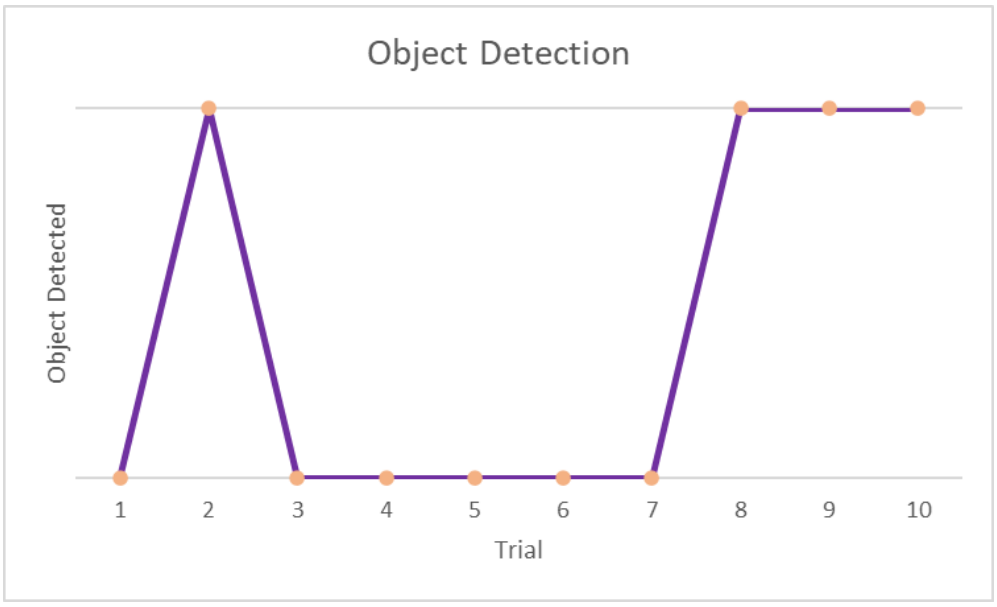

Figure 9 Graph showing obstacle detection tests. The object is considered detected if the distance sensors hit the threshold and the robot reverses and turns.

Boundary Avoidance

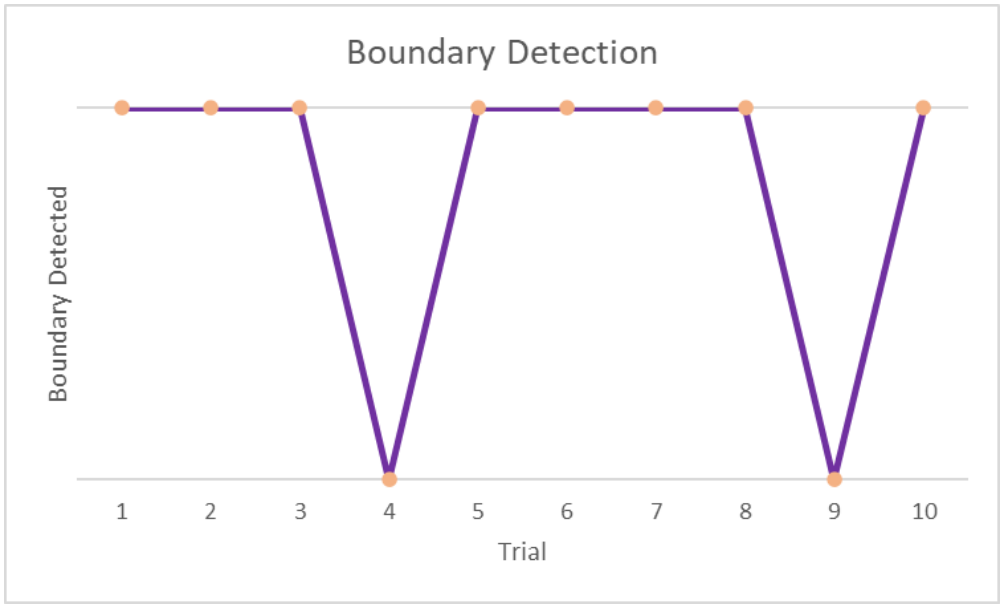

Figure 10 Graph showing boundary detection tests. The boundary is considered detected if the line follower sensor hit the threshold and the robot reverses and turns.

Snow cube Clearance in 1 min test

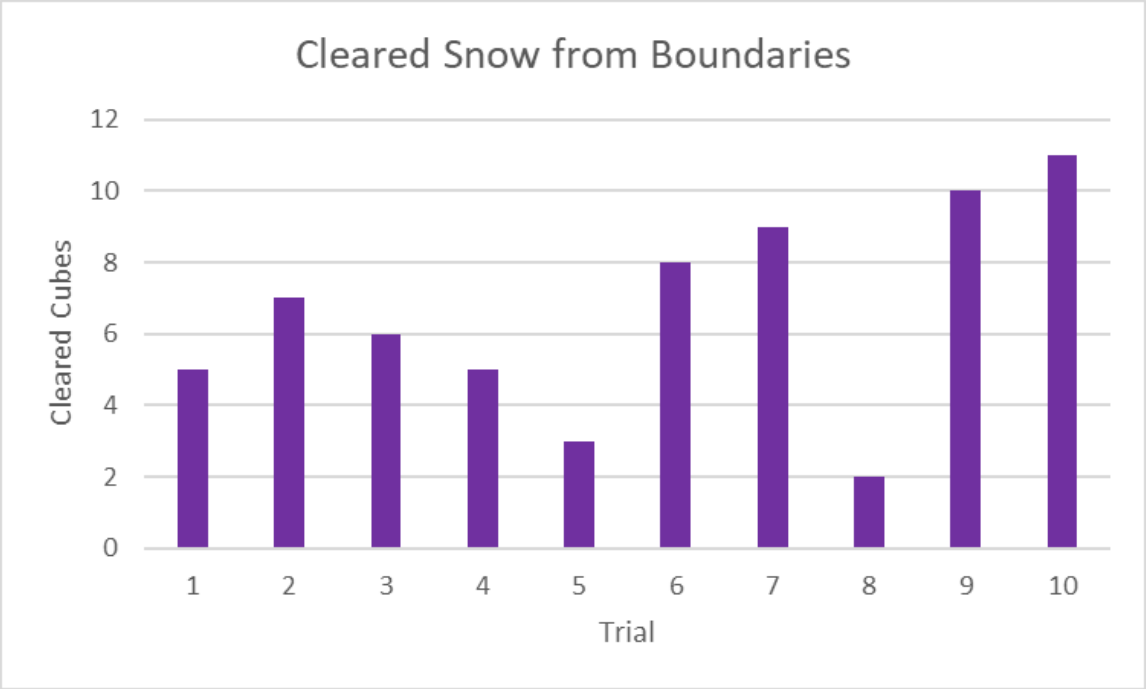

Figure 11 Graph showing cubes cleared from a simulated boundary. To simulate the cubes used in the final demo, paper balls were used instead.

System Testing

Individual Testing Before Lab 12

Trial Number	Snow Cubes Cleared	Boundary Crossed	Obstacles Hit
1	15	1	4
2	20	0	5
3	22	0	4
4	27	1	1
5	12	0	1

Customer Testing Demo Lab 12

Trial Number	Snow Cubes Cleared	Boundary Crossed	Obstacles Hit
1	33	1	1
2	47	0	1

Contributions

Section	Main Contributor	Secondary Contributor
Objective	Christopher	Sagar
Deliverables	Pranav	
Requirements	Pranav	
Detailed Deliverables	Christopher	
Testing Plan	Christopher	
List of Activities	Pranav	Christopher
Schedule Network Diagram	Pranav	
Gantt Chart	Pranav	Christopher
All Cost Sections	Sagar	
Responsibility Matrix	Christopher	
Hardware Design	Christopher	Sagar
Software Design	Christopher	
State Chart	Pranav	
Sequence Diagram	Sagar	
Watchdog Timer	Sagar	Pranav
Control Charts	Sagar	Christopher
System Testing	Sagar	Pranav