Computer Systems Design Lab SYSC 4805 (Fall 2022)

Final Report: Autonomous Snowplow

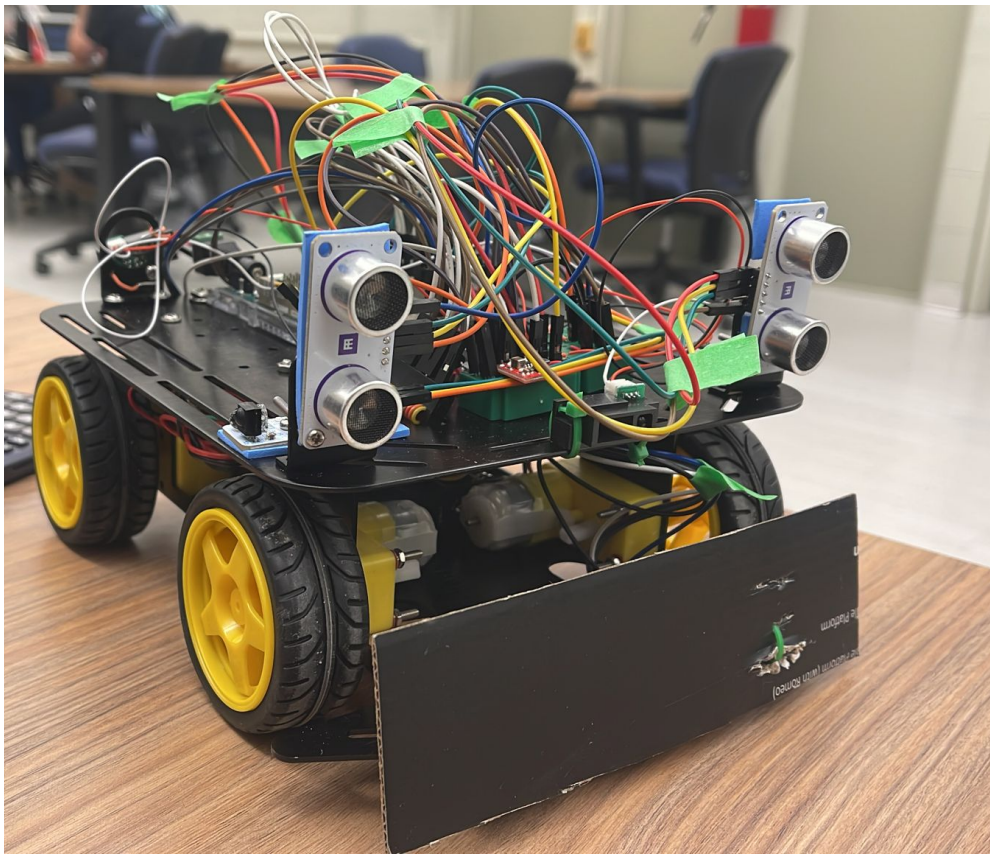

Figure 1. Assembled Robot

Group L1-G9
Team Name: Big Foot Feet
Alina Ahmad, 101111867
Jacob Charpentier, 101105745
Julian Obando Velez, 101124710

GitHub Link: https://github.com/SYSC4805-Fall2022/sysc4805_term_project-bigfootfeet_l1g9

December 9th, 2022

1.0 Project Charter

1.1 Project Objective:

The objective of this project is to make an Autonomous Snowplow. The snowplow's job is to remove snow from a region that is bounded by black tape to form an enclosed area. There will be both stationary and moving obstacles in the area. The robot must complete the mission without running into any impediments. The black trail at our testing facility is made of black painter's tape, and the boxes and other moving robots serve as obstacles, while light plastic balls are used to imitate snow. The robot will also need a unique plow which would push the snow spheres in a direction and eventually plow them outside the dark path. The robot will have different types of steering mechanisms to avoid hitting any moving and stationary obstacles in the area. The robot shown in Figure 1 has a 4-wheel-drive structure with 4 independent DC motors which means that the mechanical orientation of the wheels is around a fixed axis and facing the front of the vehicle.

1.2 Project Milestones and Deliverables:

Activity	Dates	Description
D1: Project Proposal	Oct 06 - Oct 13th	Work with the team to create the project proposal
M1: Line Following	Oct 20th - Lab 7	Add line following to basic robot shell, and test it on small square of black tape to simulate test environment. This milestone includes smaller milestones. - **M1.1** Robot movement with watchdog timer - **M1.2** line following installation - **M1.3 (Deliverable)** code commit of the line following and movement
M2: Obstacle Wall Following	Oct 27th - Reading Week	Enable the robot to detect and follow the wall of an obstacle (Cardboard box) using Ultrasonic and distance sensors - **M2.1** emergency stop - **M2.2** side distance

		sensors - **M2.3 (Deliverable)** code commit of sensors
M3: Clearing Algorithm	Nov 3rd - Lab 8	Through testing with the new known limitations decide on an algorithm to use for clearing the area. - **M3.1** Creating list of algorithms - **M3.2** Compare efficiency and complexity - **M3.3 (Deliverable)** code commit of algorithm
D2: Progress Report	Nov 11th - Lab 9	Deliver Progress Report, see sysc 4805 project description for requirements. - **D2.1** diagram designs - **D2.2** write report - **D2.3** add diagram to report - **D2.4** add algorithm description to report - **D2.5** review report
M4: Combining Algorithms	Nov 17th - Lab 10	Line Following Obstacle Avoidance Ultrasonic Sensor Implementation of the Watchdog Timer - **M4.1** combine line and wall following - **M4.2** clearing algorithm and obstacle wall following - **M4.3 (Deliverable)** implement tape boundary detection

M5: Testing	Nov 24th - Lab 11	Testing the following: - **M5.1** obstacle avoidance - **M5.2** clearing algorithm without obstacles - **M5.3** clearing algorithm with obstacles - **M5.4** Test edge conditions - Results reported in **D4 (Deliverable)**
D3: In Lecture Presentation	Nov 28th	Perform presentation to class, see sysc 4805 project description for details. - **D3.1** create presentation - **D3.2** practice presentation - **D3.3** present to class
M6: Evaluation	Dec 1st - Lab 12	Evaluation on autonomous snow plows ability to clear an area of snow with obstacles.
D4: Final Report	Dec 9th	Submit final report, see sysc 4805 project description for details. - **D4.1** write final report - **D4.2** diagram design - **D4.3** Final report review

Table 1. List of Milestones and Deadlines

2.0 Scope

2.1 Requirements

- **M1:** Line Following
    - **M1.1** The robot plow should not exceed a speed of 30cm/s.
    - **M1.2** The robot plow should decode its position in reference to the boundary (black tape) when receiving the signal from the three line following sensors.
    - **M1.3** When the robot plow detects the boundary, it should stay within 2cm until the perimeter of the boundary condition has been completely covered.
- **M2:** Obstacle Avoidance
    - **M2.1** The robot plow must momentarily stop when an obstacle is detected using any sensor.
    - **M2.2** The robot plow must be stopped until it is determined if the robot plow will turn or move forward or move backward.
    - **M2.3** The robot plow must turn away from obstacles that have been detected until the obstacle is not detected anymore.
- **M3:** Clearing Algorithm
    - **M3.3** The robot should detect one of the edges of the boundary area when it is first released to plow the snow.
    - **M3.3** The robot should follow the algorithm pattern to cover all of the boundary area before 5 minutes.
    - **M3.3** The algorithm pattern should plow the snow towards the outside of the boundary area or towards uncovered bounded space.
- **M4:** Combining Algorithm
    - The requirements for M1, M2 and M3 should all work simultaneously.

2.2 List of activities

The project will use the following break down of activities:

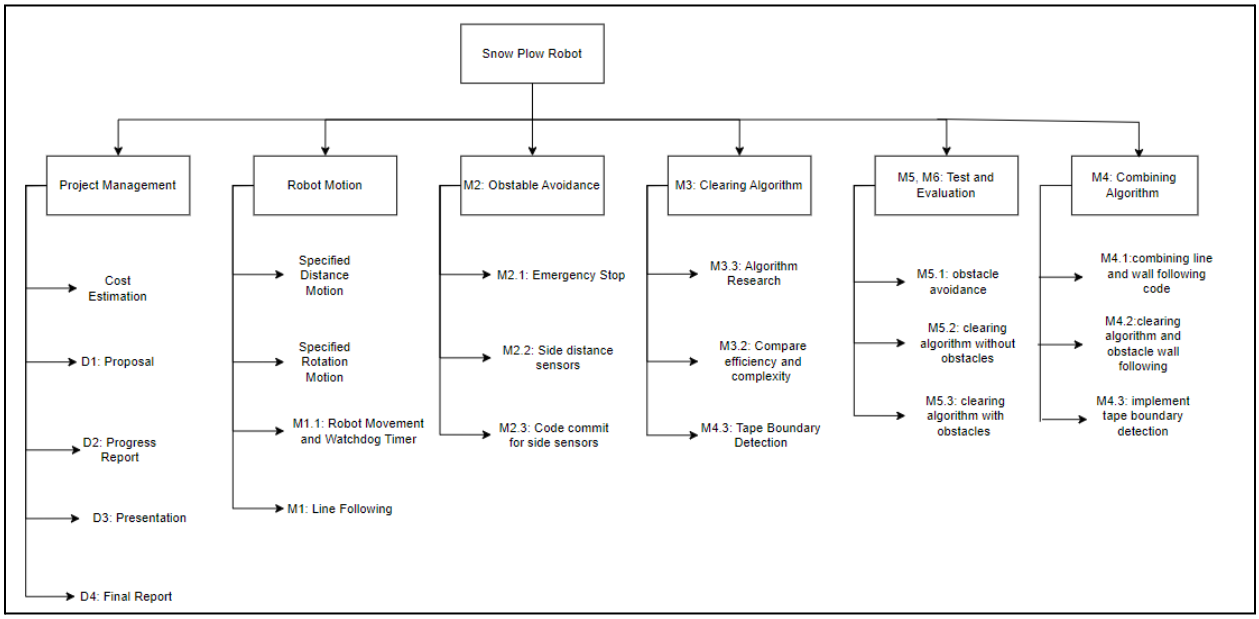

Figure 2. WBS Project Activities

(Oct. 14th - Oct. 20th) Lab 7: Line Following => **M1**

- M1.1 Robot Movement: Make robot turn in 27.5, 45, 90 and 180 degrees turns, make robot move specified distances, and get the watchdog timer working
- M1.2 Line Following Sensor Installation: Install and confirm the line following sensor works
    - Interface line follower
- M1.3 Integration of line following and movement: Integrate the two functionalities into a single working line following robot and commit code

(Oct. 21st - Oct. 27th) Reading Week: Obstacle Wall Following => **M2**

- M2.1 Emergency Stop: Make robot sense obstacles in front with VMA330 IR sensor and back up to a safe zone.
    - Install and interface VMA330 IR sensor
    - Install Plow in front of the robot.
- M2.2 Side Distance Sensors: Install the remaining sensors in the robot and interface:
    - Ultrasonic Distance Sensor
    - GP2Y0A51SK0F Distance Measuring Sensor
- M2.3 Integrate Sensors and movement: Make robot rotate past an obstacle using sensor readings and commit code

(Oct. 28th - Nov. 3rd) Lab 8: Progress Report and Clearing Algorithm => **D2 and M3**

- D2.2 Report Writing: Write the remainder of progress report
- D2.3 Diagram Design: Determine architecture of the system, statechart and sequence diagram of system
- M3.3 Algorithm Research : Research and determine the ideal algorithm for the project.

(Nov. 4th - Nov. 17th) Lab 9 and Lab 10: Combing Algorithm => **M4**

- M4.3 Tape Boundary Detection: Adding the capability of detecting the edges of the taped area.
- M4.2 Integrate Line and Wall Following: Make robot move in the enclosed path avoiding obstacles
- M4. 1 Integrate Algorithm: Adding to Integration 1 the clearing algorithm

(Nov. 18th - Nov. 24th) Lab 11:Testing => **M5**

- M5.1,4 Test for obstacle avoidance and edge conditions
- M5.2,3 Test for clearing algorithm with and without obstacles
- M5 Test for full system with obstacles

(Nov. 25th - Dec 1st.) Lab 12: Presentation and Evaluation => **D3 and M6**

- D3.1 Create Presentation
- D3.2 Practice Presentation
- M6 Set-up robot for evaluation

(Dec 2nd - Dec. 9th) Final Report => **D4**

- D4.2 Diagram Design: Update previous diagrams, create control chart, create table with testing results.
- D4.1 Report Writing
- D4.3 Report Review

2.3 Testing Plan

The following tests, seen in Table 2, were planned to determine the fulfillment of the requirements with their corresponding success criteria.

Code	Requirement	Test	Success Criteria	Success Criteria Bounds
M1.1	The robot plow should not exceed a speed of 30 cm/s.	Measure time to move 60 cm to calculate the average speed over a straight path	Distance is within bounds	Upper Limit: 30 cm/s Lower Limit: 20 cm/s
M1.2	The robot plow should decode its position in reference to the boundary.	Allow the robot to run into the boundary to check if the robot gets positioned parallel to the boundary	The robots positions itself parallel to the tape	Pass / Fail
M1.3	When the robot plow detects the boundary, it should stay within 2 cm until the perimeter of the boundary condition has been completely covered.	Allow the robot to move parallel to the boundary	The robot does not lose track of the boundary	Upper Limit: 2 cm/s away towards the outside Low Limit: 2 cm/s away towards the inside
M2.1	The robot plow must momentarily stop when an obstacle is detected using any sensor.	Robot is started while aiming at an obstacle to detect whether it stops or not.	The robot stops when an obstacle is detected	Pass / Fail
M2.2	The robot plow must be stopped until it is determined if the robot plow will turn or move forward or move backward.	Robot is started while aiming at an obstacle to measure how long it waits before moving again.	The robot stops for some time when an obstacle is detected	Upper Limit: 1 second Lower Limit: 10 milliseconds (Essentially smallest time perceptible by human)
M2.3	The robot plow must turn away from obstacles that have been detected until the obstacle is not detected anymore.	Robot is obstructed a number of times by a obstacle for which it should move away	The robot must go into the back up and turn routine	Pass / Fail

Code	Requirement	Test	Success Criteria	Success Criteria Bounds
M3.1	The robot should detect one of the edges of the boundary area when it is first released to plow the snow.	Allow the robot follow the boundary and checking if the robot turn at the edge following to next side	The robot finds the boundary and turns to the other sides of the boundary	Pass / Fail
M3.2	The robot should follow the algorithm pattern to cover all of the boundary area before 5 minutes.	The robots is released in the boundary area without obstacles and without ball simulating snow	The robot goes around the surface of the boundary area	Lower Limit: Covers 70% of area
M3.3	The algorithm pattern should plow the snow towards the outside of the boundary area or towards uncovered bounded space.	The robot is released in the boundary area without obstacles and with 100 balls simulating snow.	The robot pushes the balls outside of the boundary area	Lower Limit: 80% balls pushed outside
M4	The requirements for M1, M2 and M3 should all work simultaneously.	The robot is released in the boundary area with three obstacles and with 100 balls simulating snow.	All of M1, M2 and M3	All of M1, M2 and M3

Table 2. Testing plan for requirement verification

3.0 Schedule

3.1 Gantt Chart

Figure 3 displays the Gantt chart which shows how the team will allocate time for the activities outlined in section 2.2.

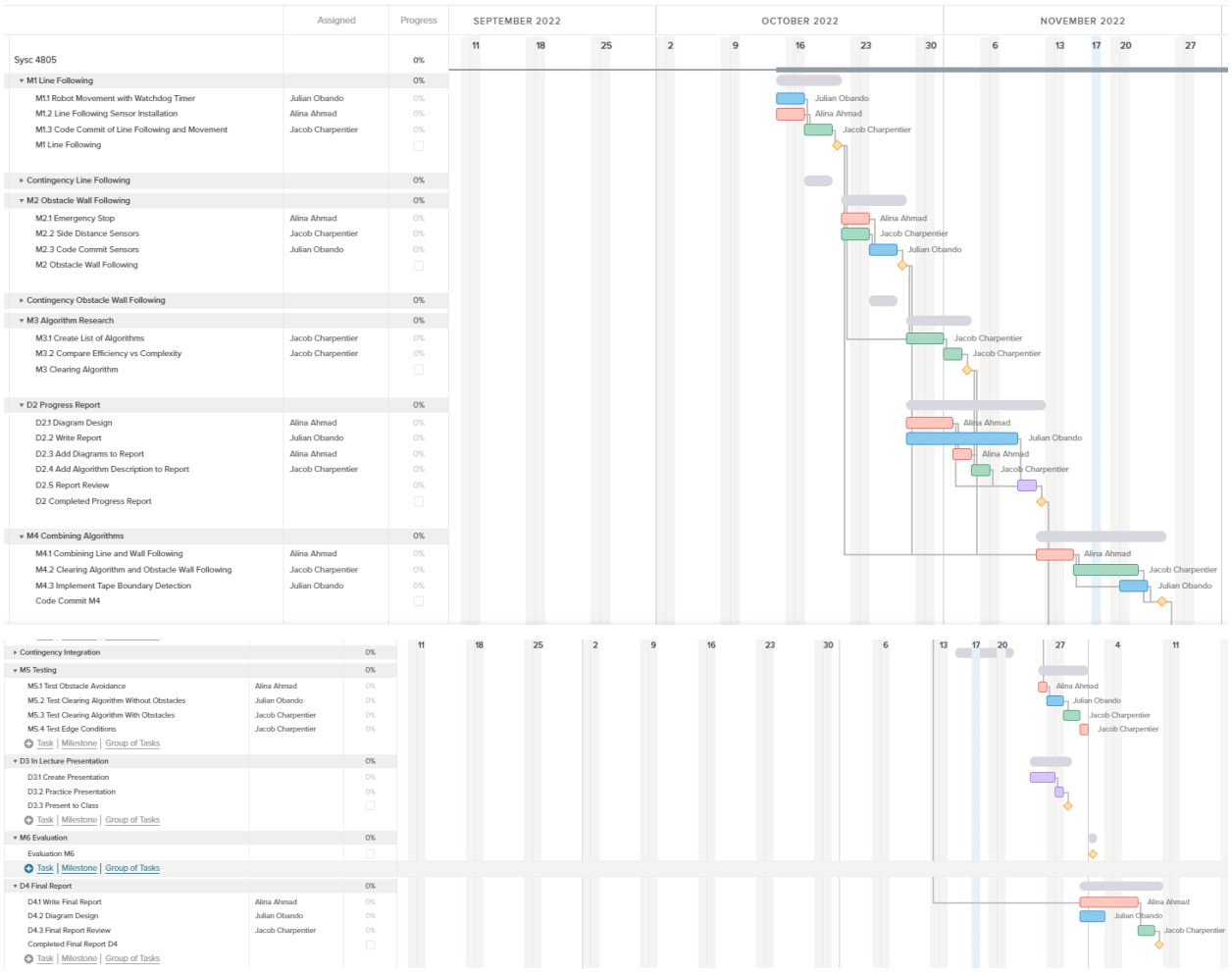

Figure 3. Gantt Chart of Project Plan, note: Green - Jacob Charpentier, Red - Alina Ahmad, Blue - Julian Obando, Purple - All Group Members, Yellow - Contingency Plan

3.2 Schedule Network Diagram

The figure below displays the Schedule Network Diagram (SND) which shows the dependencies among all the activities outlined in section 2.2 of this report. Milestones have been split into smaller milestones e.g, M1.1, for documenting reasons.

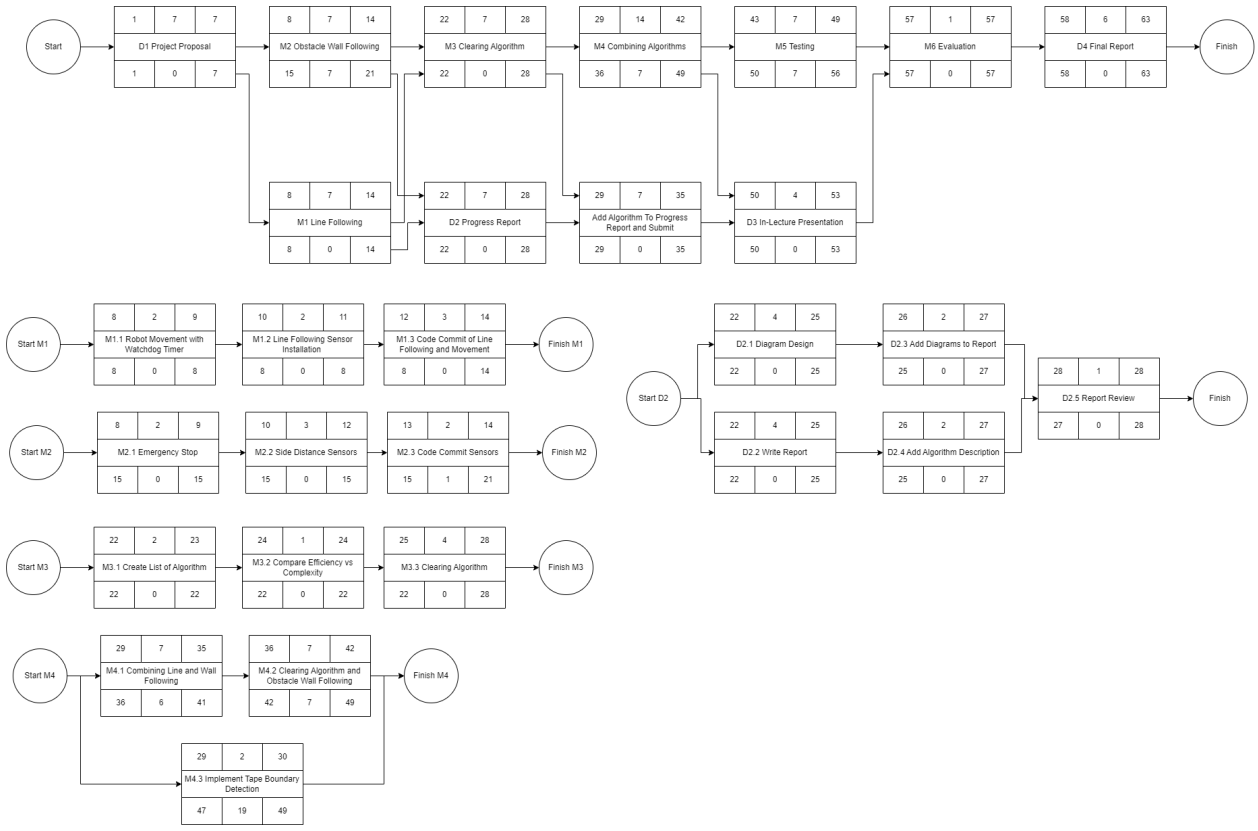

Figure 4. Network Schedule Diagram which displays predicted timelines and flexibility. Each milestone and deliverable has its own SND. Note that day 1 is Oct. 13th

4.0 Cost

4.1 Cost Baseline Figure: Assume $50/hour/developer

Activity	Developer	Completed Hours	Time Taken (days)	Total ($)
M1.1: watchdog timer and robot movement	Julian Obando	4	3	$200
M1.2 Line Following and sensor installation	Alina Ahmad	2	2	$100
M1.3 Line Following and sensor code	Jacob Charpentier	3	3	$150
M2.1 emergency stop	Alina Ahmad	3	3	$150
M2.2 side distance sensors	Jacob Charpentier	4	3	$200
M2.3 code commit of sensors	Julian Obando	3	3	$150
M3.1 Creating list of algorithms	Jacob Charpentier	2	2	$100
M3.2 Compare efficiency and complexity	Alina Ahmad	2	2	$100
M4.1 combine line and wall following	Alina Ahmad	3	2	$150
M4.2 clearing algorithm and obstacle wall following	Julian Obando	4	2	$200
M4.3 implement tape boundary detection	Jacob Charpentier	4	3	$200
M5.1 Test obstacle avoidance	Alina Ahmad	4	4	$200
M5.2 Test clearing algorithm without obstacles	Julian Obando	4	4	$200
M5.3 Test clearing algorithm with obstacles	Jacob Charpentier	4	3	$200
M5.4 Test edge conditions	Jacob Charpentier	3	2	$150
Project Budget = $4,000 **Planned Value** = $2,750 **Management Reserve (20% of PV) =** $550				

Table 3. Project Budget

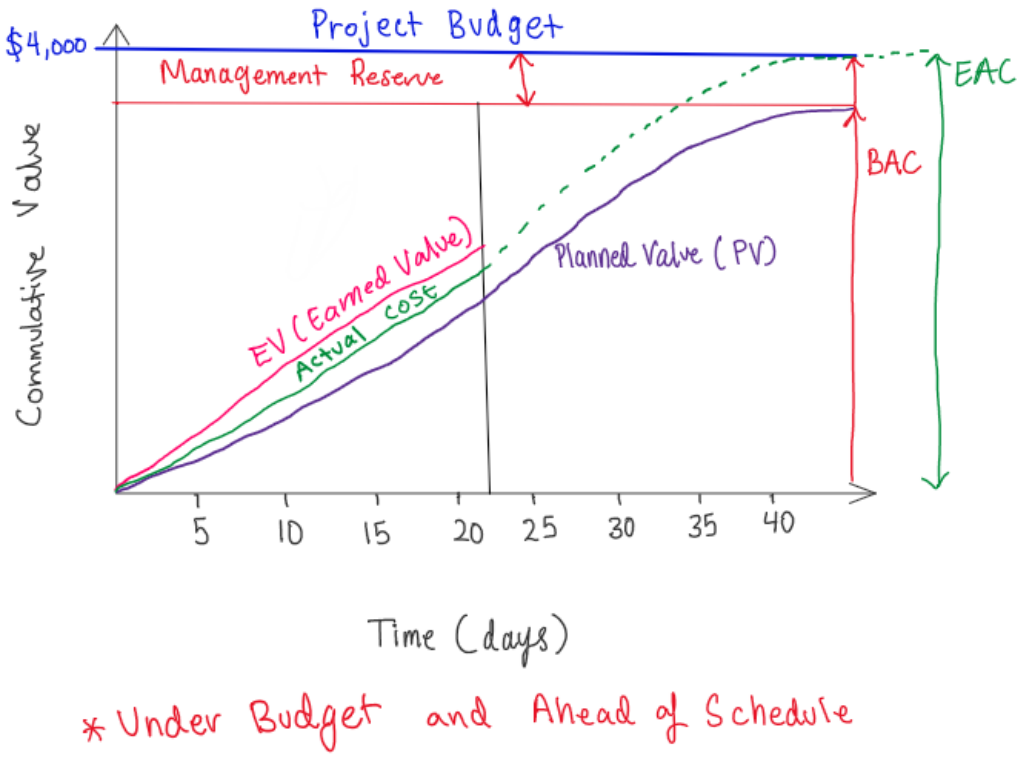

Figure 5. Cost Baseline Figure

5.0 Human Resources

5.1 Assignment Matrix

	Jacob	Alina	Julian
Robot Movement	A		R
Interface Line Following Sensor		R	A
Integration of Line Following and Movement	R	A	
Emergency Stop	A	R	
Interface Side Distance Sensors	R		A
Integrate Sensors and Movement		A	R
Progress Report Writing	A		R
Progress Report Diagram Design		R	A
Algorithm Research	R	A	
Integrate Line and Wall Following	A	R	
Progress Report Algorithm Description	R		A
Test Line and Wall Following		A	R
Integrate Algorithm	R	A	
Implement Tape Boundary Detection	A		R
Test Algorithm	R		A
Test Obstacle Avoidance	A	R	
Test Clearing Algorithm Without Obstacles		A	R
Test Clearing Algorithm With Obstacles	R		A
Final Report Writing	A	R	
Final Report Diagram Design		A	R
Final Report Review	R		A

Table 4. Responsibility Assignment Matrix, R = Responsible, A = Approver

6.0 System Architecture

The robot plow is made up of the following components seen in the architecture diagram in Figure 6.

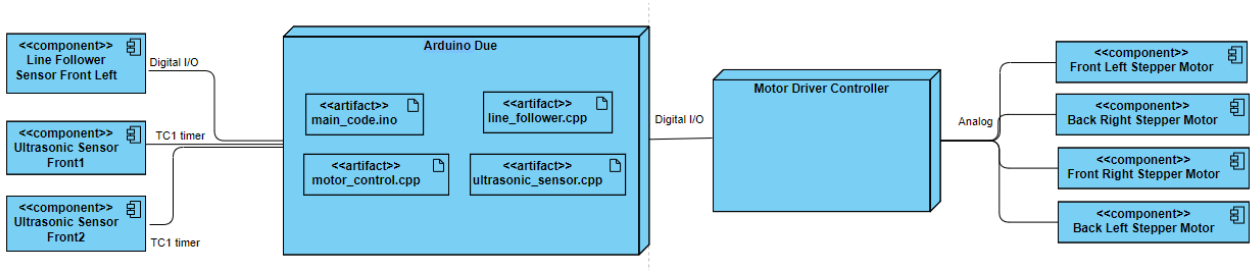

Figure 6. Overall architecture diagram

The Arduino Due board takes the measurement from the sensors and directs the motor driver to make the robot move as desired.

The placement of the sensors can be seen below in Figures 8 and 9 and was determined using Figure 7 below. In essence, the robot plow uses two ultrasonic sensors in front to detect the obstacles for when the robot plow is moving forward. It uses one line follower sensor, placed diagonally at the front-left side of the robot. Finally, it uses two ultrasonic sensors to check for obstacles in the sides for when the robot is turning. The ultrasonic sensor have an active range of 5-15 cm for our robot. This because we want it to react to objects that are within 15 cm and we have a 5cm debouncing value.

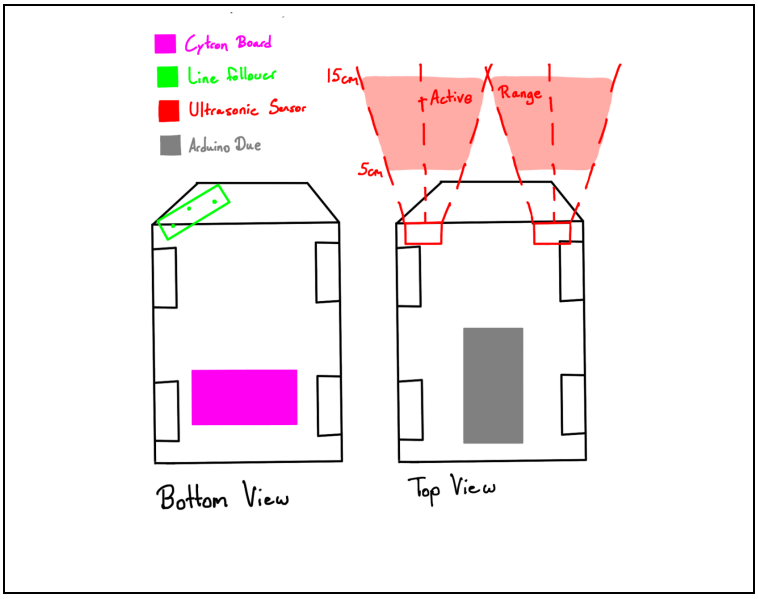

Figure 7. Evaluation of required sensor placements for robot.

Figure 8. Line sensor placement on the robot

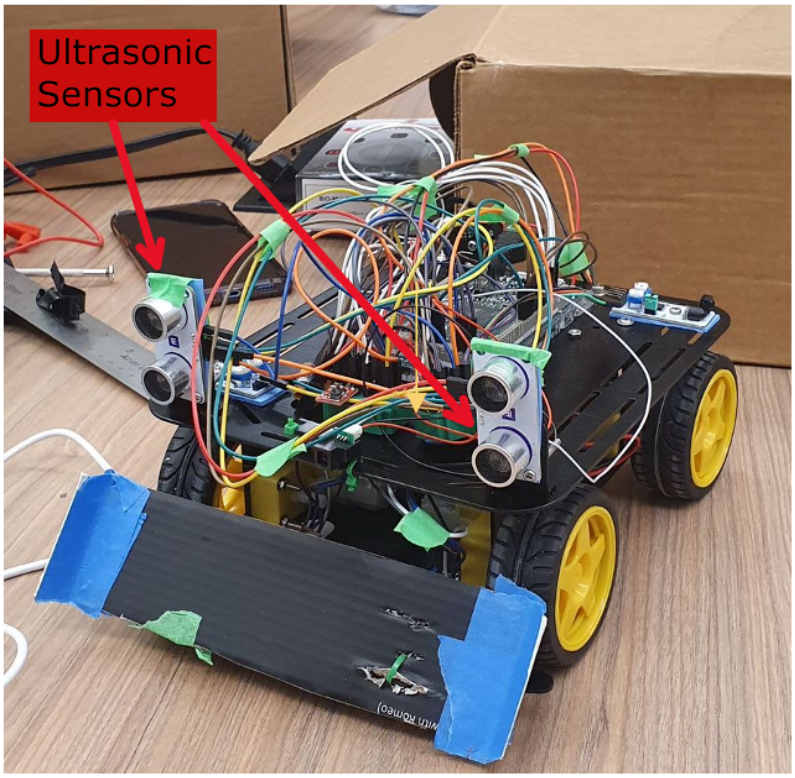

Figure 9. Two ultrasonic sensos

6.1 Completed Activities

This section will highlight and link the completed activities to its supporting github commit.

M1.1: Basic Movement, Watchdog Timer
https://github.com/SYSC4805-Fall2022/sysc4805_term_project-bigfootfeet_l1g9/commit/cd38b1ab0d388d31511717cdba93f4d0d2f20b68

https://github.com/SYSC4805-Fall2022/sysc4805_term_project-bigfootfeet_l1g9/commit/6a19d8087effa5e140915ccd3e458b21572133ed

M1.2: Line Following

https://github.com/SYSC4805-Fall2022/sysc4805_term_project-bigfootfeet_l1g9/commit/7b09205e9817effe8c2ceadc8b5a77a620579a30

M1.3: Line Follower Sensors

https://github.com/SYSC4805-Fall2022/sysc4805_term_project-bigfootfeet_l1g9/commit/3f56c166a81c578b7908a097f054bc077e545be4

M2.1: Motor Control, Ultrasonic Sensors, Emergency Stop

https://github.com/SYSC4805-Fall2022/sysc4805_term_project-bigfootfeet_l1g9/commit/b8228f0dada0a698772ad56f4bb5070a49806081

https://github.com/SYSC4805-Fall2022/sysc4805_term_project-bigfootfeet_l1g9/commit/889765e5f1939cd5d1110e20d45234dbc04cbcf8

https://github.com/SYSC4805-Fall2022/sysc4805_term_project-bigfootfeet_l1g9/commit/da2ddac75dadf6a09598b891cd1e5153504ff4fa

M3.1: 2 Algorithms were compared, Back and Forth, and Encircle + Random.

Back and Forth: Traverse along line and scan for max distance on ultrasonic sensor, if distance is found travel down the path and turn 90 degrees to the right. Continue to scan for gaps and traverse them.

Encircle + Random: Travel around the perimeter by line following, then when entire perimeter has been covered begin random pathing. Random pathing involves travelling until a wall or line is detected, then turning a random amount to the right and continuing this pattern.

M3.2:

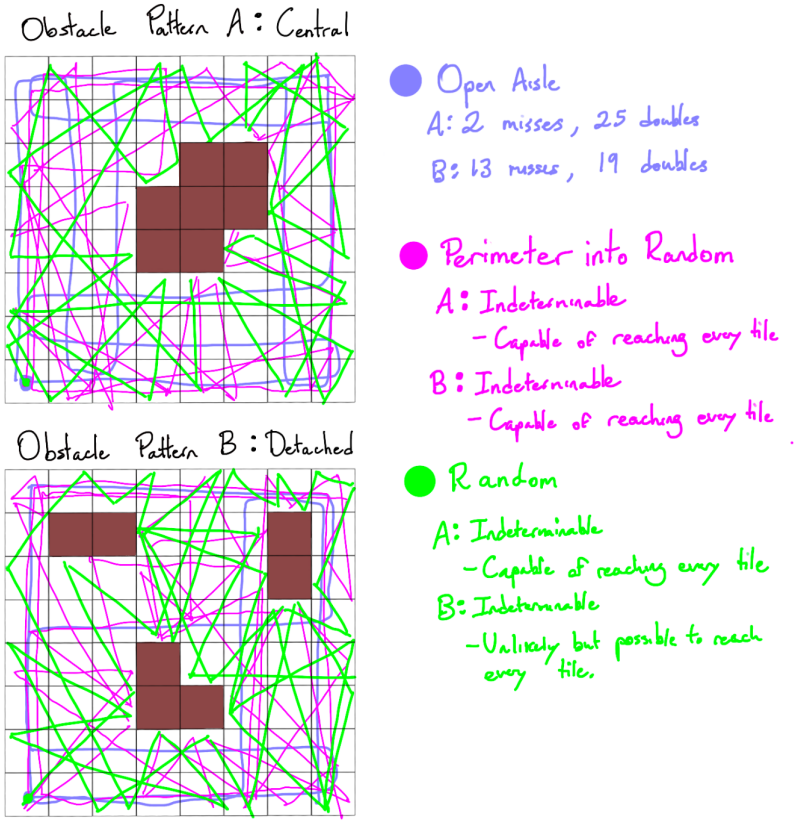

Figure 10. Comparison of two algorithms detailed above.

M4.1, M4.2, M4.3: Line Following, Backup, Random Turn Right

https://github.com/SYSC4805-Fall2022/sysc4805_term_project-bigfootfeet_11g9/commit/7b09205e9817effe8c2ceadc8b5a77a620579a30

https://github.com/SYSC4805-Fall2022/sysc4805_term_project-bigfootfeet_11g9/commit/da2ddac75dadf6a09598b891cd1e5153504ff4fa

https://github.com/SYSC4805-Fall2022/sysc4805_term_project-bigfootfeet_11g9/commit/9f46335880f7602b4cd7b1ffe380c5347678e64f

https://github.com/SYSC4805-Fall2022/sysc4805_term_project-bigfootfeet_11g9/commit/3d1c67e6f35a647b5628bdf56ae1d3e56728571c

7.0 Statechart

The behavior of the robot is described in detail below:

- Obstacle in front or side of the robot: “Back up is initiated and once backup is completed then “Random Turn Right” is used to turn the robot away from obstacle and then it returns to moving “Forward”.
- If line is detected while moving forward then “Backup” state will be initiated. Once backup is complete, “Random Turn Right” state would be initiated. The robot will enter “Forward” state once turn is complete.

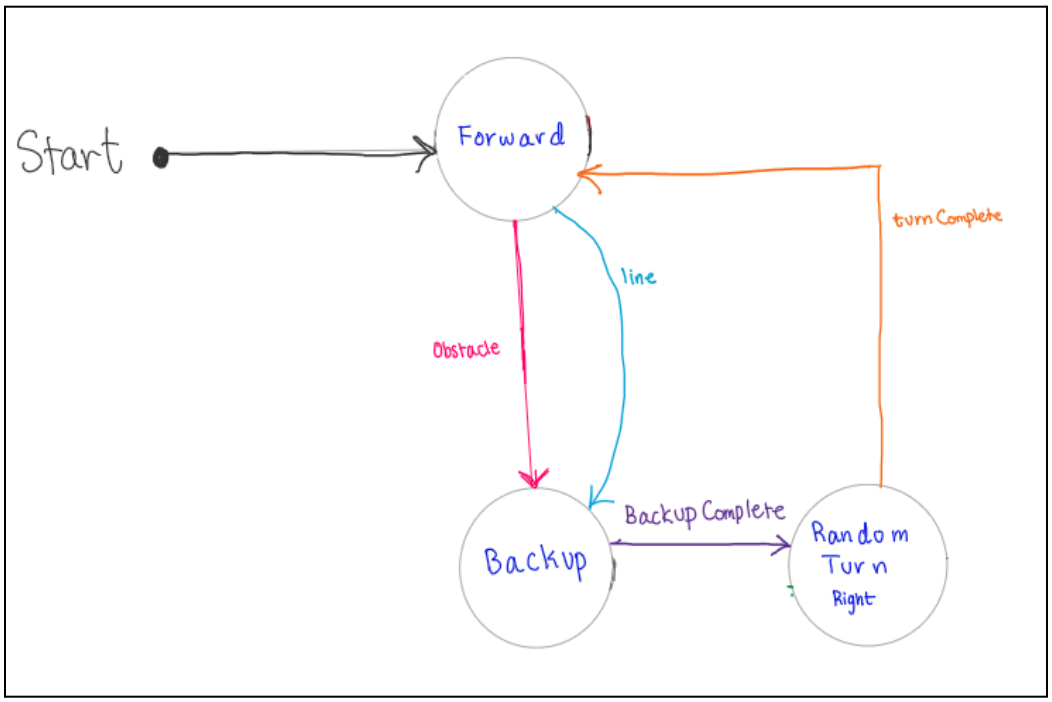

Figure 11. Statechart diagram for the robot

8.0 Sequence Diagram

By following the statechart diagram shown in Section 7.0, the robot plow follows the sequence shown below in Figure 12 when the ultrasonic sensors detect an obstacle or when the front line follower finds the tape boundary. In this case, the robot plow will back up for a determined distance to then turn to the right at a randomized degree of rotation.

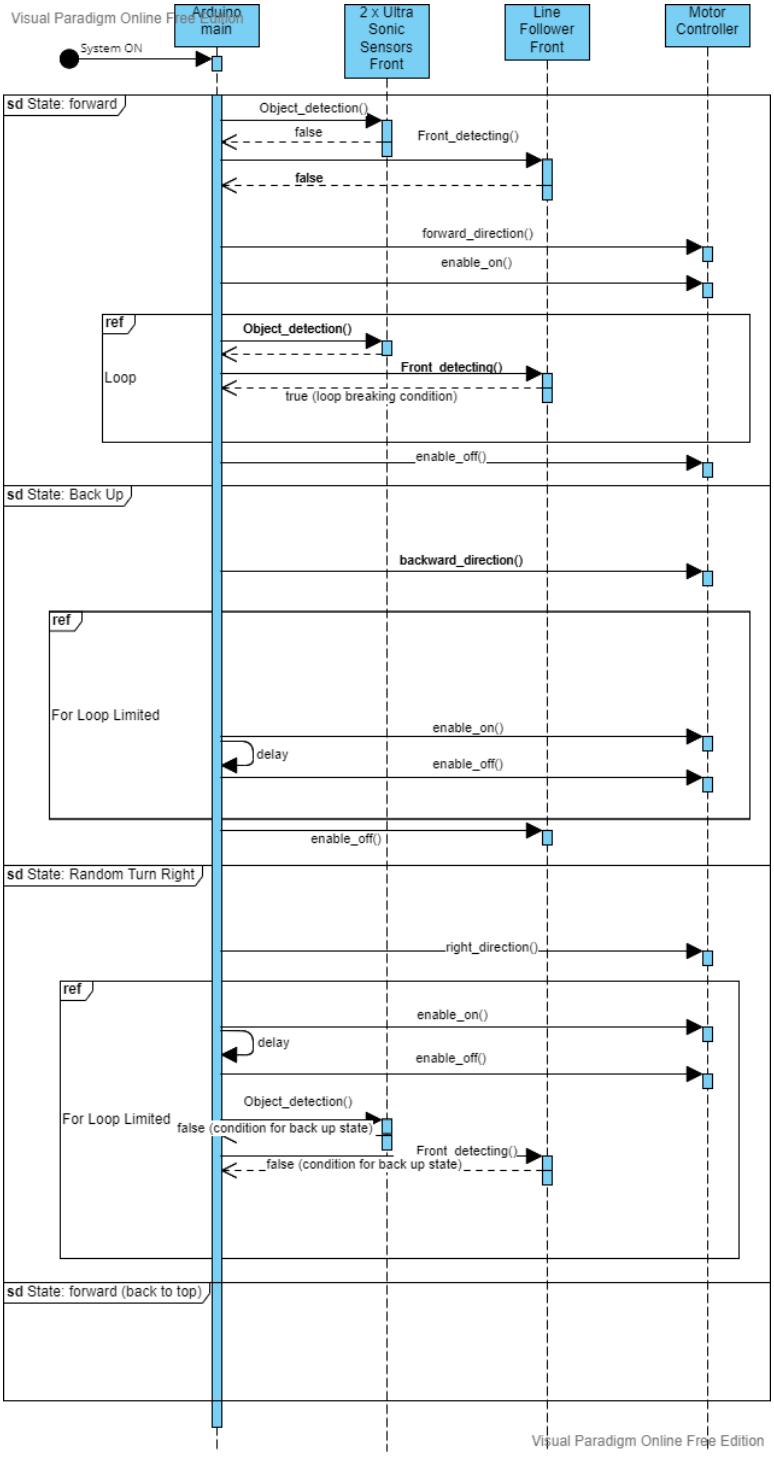

Figure 12. Sequential diagram of the system

As it can be seen in the Figure 12 above, every time the robot plow is moving, it constantly checks for the sensors that are towards the direction that it's moving. The robot plow does this to detect any possible obstacles that are in the way to avoid collisions.

9.0 Watchdog Timer

main_code.ino.ino | distance_gp2.cpp | distance_gp2.h | ir_sensor.cpp | ir_sensor.h | line_follower.cpp | line_follower.h | motor_control.cpp | mot

```
138  }
139
140  void setup() {
141
142    // Enable watchdog.
143    WDT->WDT_MR = WDT_MR_WDD(0xFFF) |
144                  WDT_MR_WDFIEN |  //  Triggers an interrupt or WDT_MR_WDRSTEN to trigger a Reset
145                  WDT_MR_WDV(256 * 5); // Watchdog triggers a reset or an interrupt after 5 seconds if underflow
146    // 2 seconds equal 84000000 * 5 = 420000000 clock cycles
147    /* Slow clock is running at 32.768 kHz
148      watchdog frequency is therefore 32768 / 128 = 256 Hz
149      WDV holds the period in 256th of seconds  */
150    NVIC_EnableIRQ(WDT_IRQn);
151
```

Figure 13. Watchdog timer in main_code.ino setup()

```
void loop()
{

  //Restart watchdog
  WDT->WDT_CR = WDT_CR_KEY(WDT_KEY)
                | WDT_CR_WDRSTT;

  switch (state) {
    case state_forward:
      state_forward_handler();
      break;
    case state_turn_right:
      state_turn_right_handler();
      break;
    case state_backup:
      state_backup_handler();
      break;
    case state_inch_forward:
      state_inch_forward_handler();
      break;
    case state_random_turn_right:
      state_random_turn_right_handler();
      break;
  }
}
```

Figure 14. Watchdog timer restart in main_code.ino

```
void watchdogSetup(void) {
  /*** watchdogDisable (); ***/
}

void WDT_Handler(void)
{
  if (phase == 1){
    phase = 2;
    line_following = false;
    //Restart watchdog
    WDT->WDT_CR = WDT_CR_KEY(WDT_KEY)
                | WDT_CR_WDRSTT;
  } else {
    /* Clear status bit to acknowledge interrupt by dummy read. */
    WDT->WDT_SR; // Clear status register
    enable_off(BL_Wheel_Enable, BR_Wheel_Enable, FL_Wheel_Enable, FR_Wheel_Enable);
    printf("help! in WDT\n");
  }
}
```

Figure 15. Watchdog timer setup override and watchdog timer handler in main_code.ino

Figures 13, 14, and 15 above show how the watchdog timer is initialized in setup, where the timer has a period of 10 seconds. During normal operation the timer will be reset during every loop call, if this does not occur it indicates that something has caused the code to become stuck. The timer will trigger after these 10 seconds; if this occurs, the first time the robot will change from phase 1 (line following) to phase 2 (random pathing). This will cause the efficiency of the robot to decrease as random pathing has trouble reaching the outer edges of the test space, however the robot will still be functional. The timer will be reset, however, if the watchdog timer triggers a second time the robot will stop moving and print a failure message. This two phase event enables the robot to attempt to continue with most of its functionality if the issue is localized to the first phase while still being capable of a full halt of processes should it be required.

10.0 Control Charts

During the testing of the robot plow, its performance was controlled by keeping track of its capacity to fulfill the different requirements. Therefore, the tests were performed during Lab 11 so that each requirement was tracked over a number of sample runs.

Figure 16 below shows how the robot plow maintained a reasonably constant speed when provided with full power to all of its wheels. Moreover, it was always within the required speed.

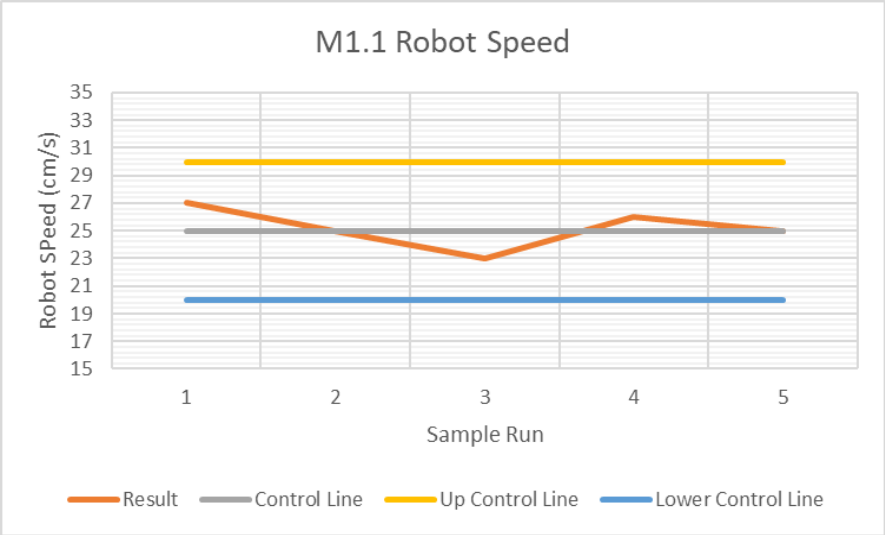

Figure 16. Control chart for the requirement M1.1 for robot speed

Figure 17 and Figure 18 shows how the robot car was very successful at finding the boundary with all of its runs passing and also at momentarily stopping when either this boundary was detecting, or an obstacle.

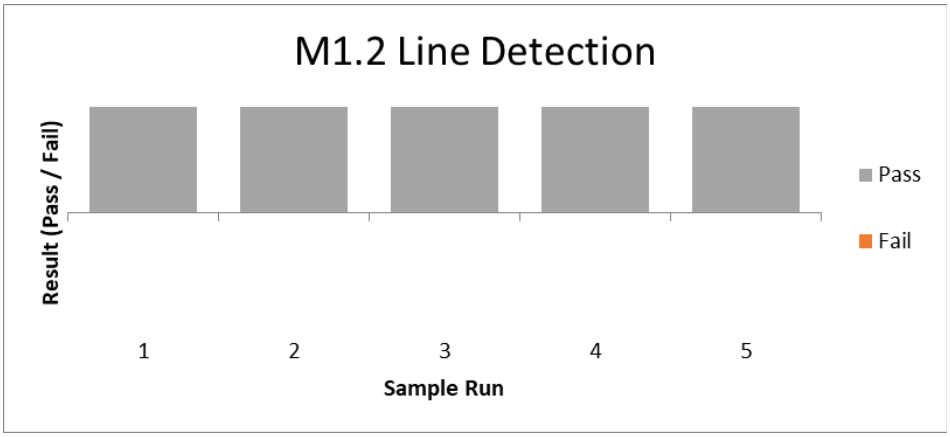

Figure 17. Control chart for the requirement M1.2 for the line detection

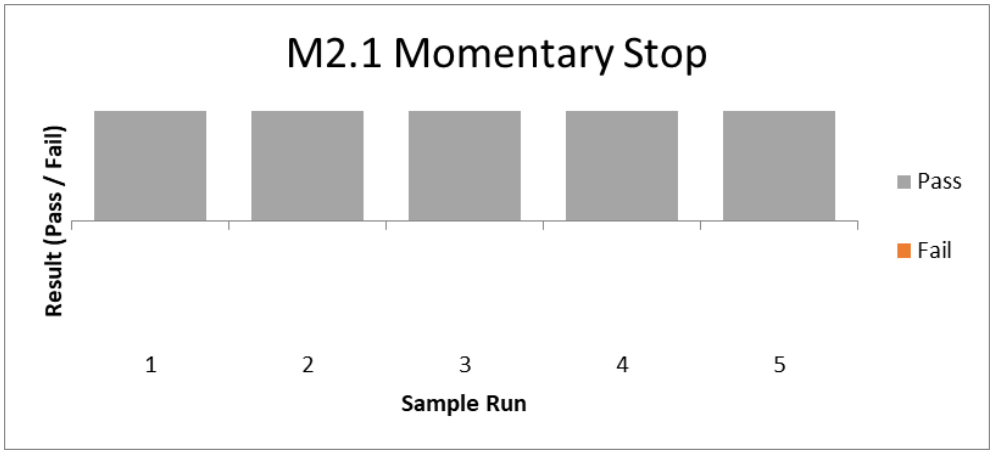

Figure 18. Control chart for the requirement M2.1 for the momentary stop.

Now the following control chart in Figure 19 shows how robot plow usually had a reaction time of about 30 ms, but in one occasion it failed because it too a longer time to make the decision to back up. This occurred for an unknown reason. Finally, in the last sample run the behavior of going back into forward motion immediately was observed. This same behavior was the one that showed up more frequently during the demo.

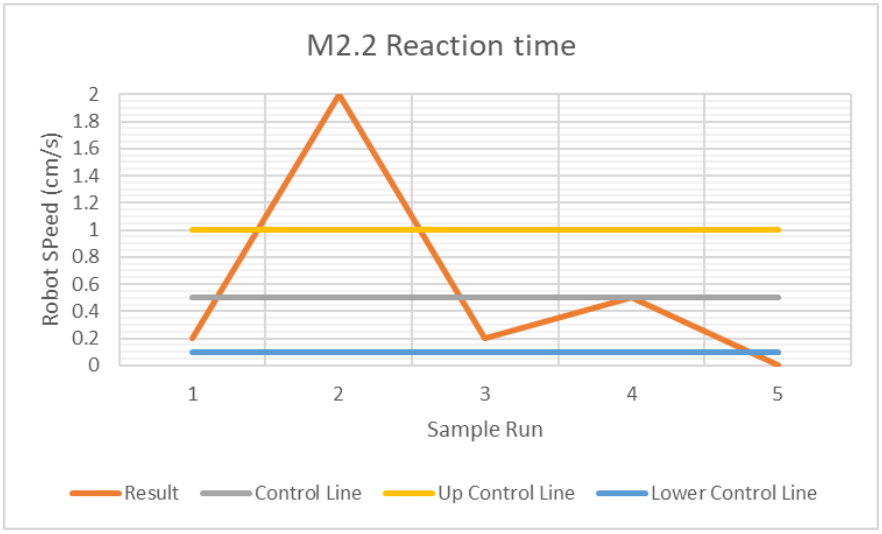

Figure 19. Control Chart for the requirement M2.2 for reaction time

The control chart for the requirement for avoiding obstacles shows that in the final sample runs the robot plow was failing to detect the obstacles, however these were the runs that helped us to improve the placement of these distance sensors because we noticed that these had to be facing directly to the front and in a vertical position.

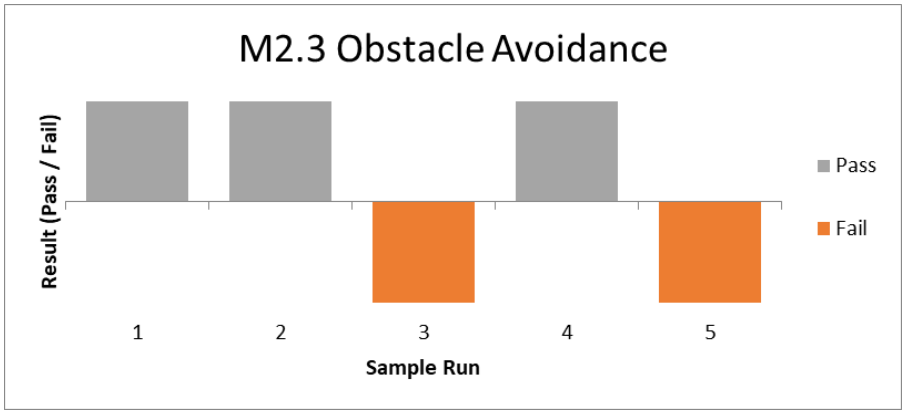

Figure 20. Control Chart for the requirement M2.3 for the obstacle avoidance

The Figure 21 below shows how the random turn algorithm had difficulties in cases where the obstacles were placed close together, especially close to the edges of the boundary area. This was observed in the sample run 4, where it only covered 30%. It was also observed that the batteries were not very reliable since sample 2 failed because the car lost power.

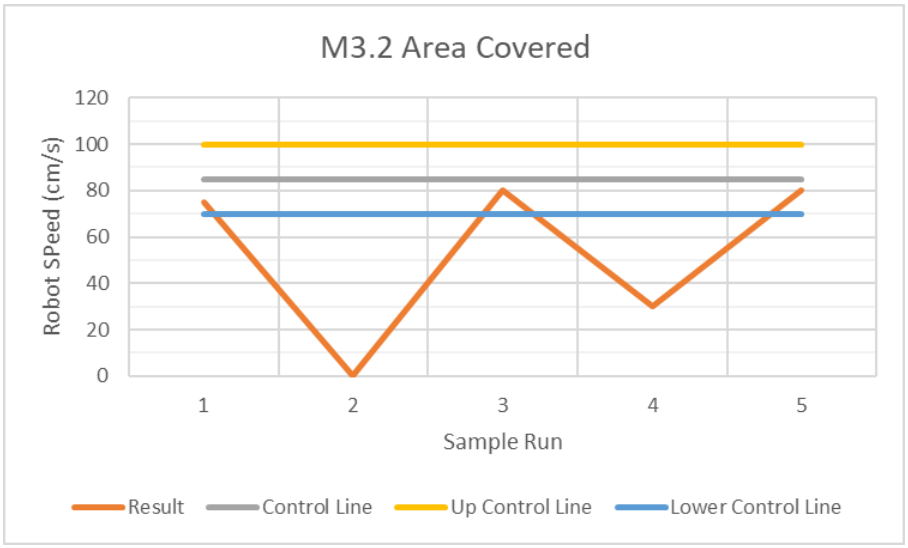

Figure 21. Control Chart for the requirement M3.2 for the area covered

During the testing of the Snow Clearing, as seen in Figure 22, the robot showed more of this back and forward behavior. Initially, this was thought to be resolved by decreasing the complexity of the algorithm by removing some of the sensors. However, during the demo the same behavior was shown again.

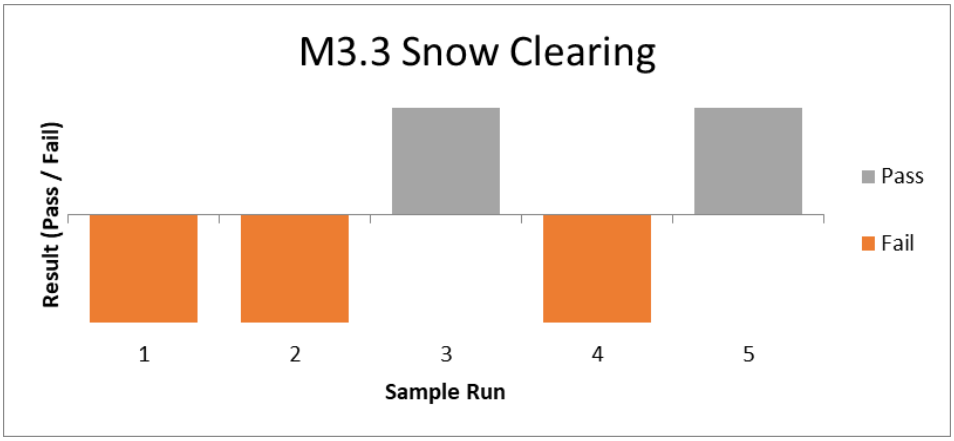

Figure 22. Control Chart for the requirement M3.3 for the clearing of snow

Finally, Figure 23 shows the control chart that included the last two sample runs of the demo.

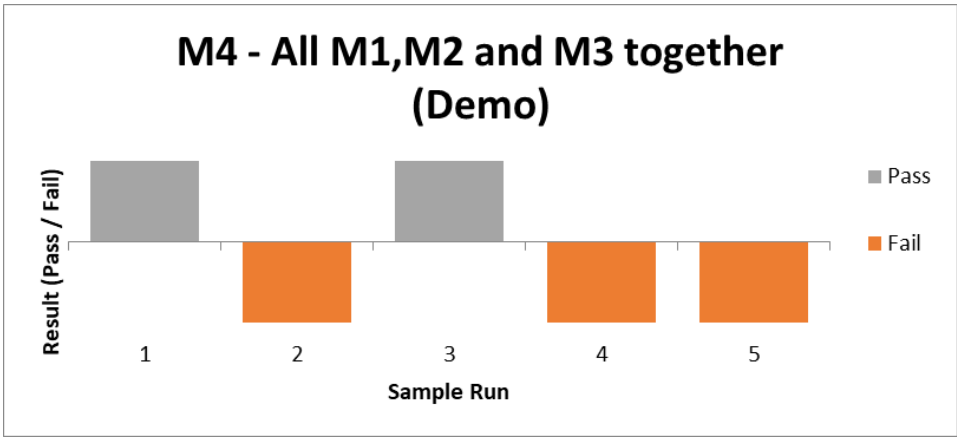

Figure 23. Control Chart for the requirement M4 for all of the requirement tested together, included the costumer testing in the demo

11.0 Results of Testing

During initial implementation and testing of individual requirements it was decided to change some of the requirement tests so that these were more appropriate for the implementation of the random turn right algorithm only. These were changed as following;

Code	Original Requirement	New Requirement	Reason for Change
M1.2	The robot plow should decode its position in reference to the boundary when receiving the signal from the three line following sensors.	When a line is detected the robot plow shall perform the obstacle avoidance sequence without leaving the boundary.	Testing revealed that corner detection was unreliable resulting in the removal of phase 1 from the algorithm.
M1.3	When the robot plow detects the boundary, it should stay within 2 cm until the perimeter of the boundary condition has been completely covered.	N/A	Due to the removal of phase 1 this requirement is no longer necessary.
M3.1	The robot should detect one of the edges of the boundary area when it is first released to plow the snow.	N/A	Due to the removal of phase 1 this requirement is no longer necessary.

Table 5. Update to the requirements during implementation

Table below shows the results of the testing done to the requirements.

Code	Requirement	Sample Run 1	Sample Run 2	Sample Run 3	Sample Run 4	Sample Run 5
M1.1	The robot plow should not exceed a speed of 30cm/s.	Pass - 27cm/s	Pass - 25 cm/s	Pass - 23 cm/s	Pass - 26 cm/s	Pass - 25 cm/s
M1.2	When a line is detected the robot plow shall perform the obstacle avoidance sequence without leaving the boundary.	Pass	Pass	Pass	Pass	Pass
M2.1	The robot plow must momentarily stop when an obstacle is detected using any sensor.	Pass	Pass	Pass	Pass	Pass
M2.2	The robot plow must be stopped until it is determined if the robot plow will turn or move forward or move backward.	Pass - 200 ms	Fail - 2 secs	Pass - 200 ms	Pass - 500 ms	Fail - 0 secs (Moved forward immidiatly)
M2.3	The robot plow must turn away from obstacles that have been detected until the obstacle is not detected anymore.	Pass	Pass	Fail (Moved forward instead)	Pass	Fail (Moved forward instead)
M3.2	The robot should follow the algorithm pattern to cover all of the boundary area before 5 minutes.	Pass - 75% area covered	Fail (The robot lost power in the middle of the test)	Pass - 80% area covered	Fail - 30% area covered	Pass - 80% area covered
M3.3	The algorithm pattern should plow the snow towards the outside of the boundary area or towards uncovered bounded space.	Fail (The robot lost power in the middle of the test)	Fail (The robot got stuck moving front and forward, rather than rotating)	Pass	Fail (The robot got stuck moving front and forward, rather than rotating)	Pass
M4	The requirements for M1, M2 and M3 should all work simultaneously.	Pass	Fail (Hit obstacles four times)	Pass	Fail - Demo	Fail - Demo

Table 6. Results of testing

12.0 Members Contributions

Code	Main Contributor(s)	Secondary Contributor(s)	Approver(s)
M1.1	Julian Obando	Jacob Charpentier	Alina Ahmad
M1.2	Jacob Charpentier		Julian Obando, Alina Ahmad
M1.3	Alina Ahmad		Jacob Charpentier, Julian Obando
M2.1	Alina Ahmad	Julian Obando, Jacob Charpentier	Alina Ahmad, Julian Obando, Jacob Charpentier
M2.2	Jacob Charpentier		Julian Obando, Alina Ahmad
M2.3	Julian Obando	Jacob Charpentier	Alina Ahmad
M3.2	Jacob Charpentier	Julian Obando, Alina Ahmad	Julian Obando, Alina Ahmad, Jacob Charpentier
M3.3	Jacob Charpentier	Alina Ahmad, Julian Obando	Julian Obando, Alina Ahmad, Jacob Charpentier
M4	Alina Ahmad, Julian Obando, Jacob Charpentier		Jacob Charpentier, Julian Obando, Alina Ahmad

Table 7. Table of contributors