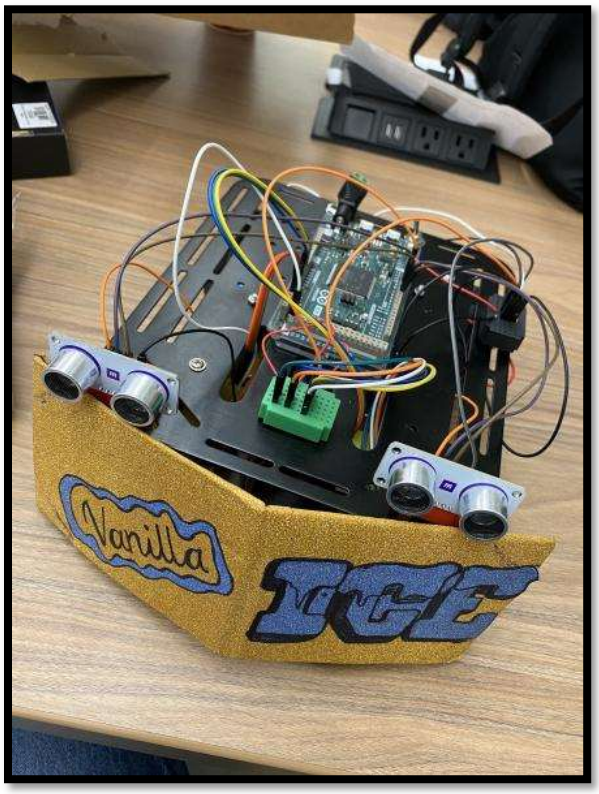


VANILLA ICE SNOWPLOW

Final Report

Aksh Ravishankar, Eline Nuviadenu, Anish Tankala

Table of Contents

Charter

Objective

The goal of this project is to build a small-scale, fully autonomous snow plow to push several ping-pong balls, herein defined as “snow”, while avoiding collisions with objects it may encounter. The vehicle should also be capable of detecting and staying within defined boundaries on the ground while clearing the snow from the region bounded by lines on the ground. The vehicle will be equipped with an array of sensors to help detect the boundaries and objects that are within the area of interest. Using the information gathered by the sensors, the device must autonomously make decisions to avoid obstacles and stay within boundaries while clearing the snow in the region.

Deliverables

Code	**Deliverables**
I	The Autonomous Snow Clearing Vehicle
II	A project proposal that covers the scope, schedule, cost and required resources of the project
III	A progress report that covers the work completed thus far, a comparison between completed tasks and expected task completion, and a comparison between current and baseline cost.
IV	A presentation that covers the architecture and structure of the project.
V	A project demonstration that shows the performance of the product in the lab environment.
VI	A final report that documents the coverage of the project, including a comparison to the original time and budget estimates, experimental data for the average performance of the required operation, and a comparison to the original requirements.

Table 1: Project deliverables and their respective code identifiers

Scope

Requirements

Number Code	**Requirement**
1	When the distance sensor detects an object within 15 centimeters, the system will signal the motors to turn to avoid a collision with the object.
2	When the line detection sensors detect a line, the system will signal the motors to turn to stay within the line boundaries.
3	The system shall signal the motors to push the snow in the same direction as the current heading until a source of interference is detected.
4	The vehicle shall stop moving within 2 seconds when a detected object is unavoidable.
5	The vehicle shall stop moving within 2 seconds when a detected line is unavoidable.
6	The vehicle shall clear all snow detected within the perimeter of the bound region within 5 minutes.
7	The vehicle shall not travel at a velocity greater than 30 centimeters per second.
8	The dimensions of the vehicle’s snow plow shall not exceed 40 millimeters in width and shall not exceed 20 millimeters in length.

Table 2: Project requirements and their respective code identifiers

Task List

Letter Code	Task	Requirements covered
A	Test ultrasonic and IR object detection sensors such that it can signal when detecting an object below a threshold	1
B	Test cluster of line following sensors such that a suitable signal is sent to turn when a line is detected	2, 5
C	Test and calibrate IMU to find and communicate vehicle heading information	3, 6
D	Calibrate motor controller board and verify motor operation and control through controller signals	4, 5, 6
E	Prepare an integrated test plan	6
F	Read sensor data to determine when an object is detected vs when a snowball is detected and make decisions accordingly	1, 3, 4
G	Read sensor data to determine when a line is detected and respond as outlined by requirements	2, 5
H	Calibrate motor/wheel encoders to find motor speed settings	7
J	Design motor control for heading and speed	7
K	Determine optimal mounting locations and mechanism	8
I	Wire sensors to pass data to the controller and mount sensors on board	1, 2, 3, 4, 5
L	Design and mount a “plow” to the front of the vehicle	3, 8
M	System-wide hardware testing to ensure reliable operation under normal conditions	6
N	System-wide software testing to ensure reliable decision-making under normal conditions	4, 5, 6
O	Verification testing for stable integration of hardware, software, and design components	6

Table 1: List of tasks and letter codes

Work Breakdown Structure

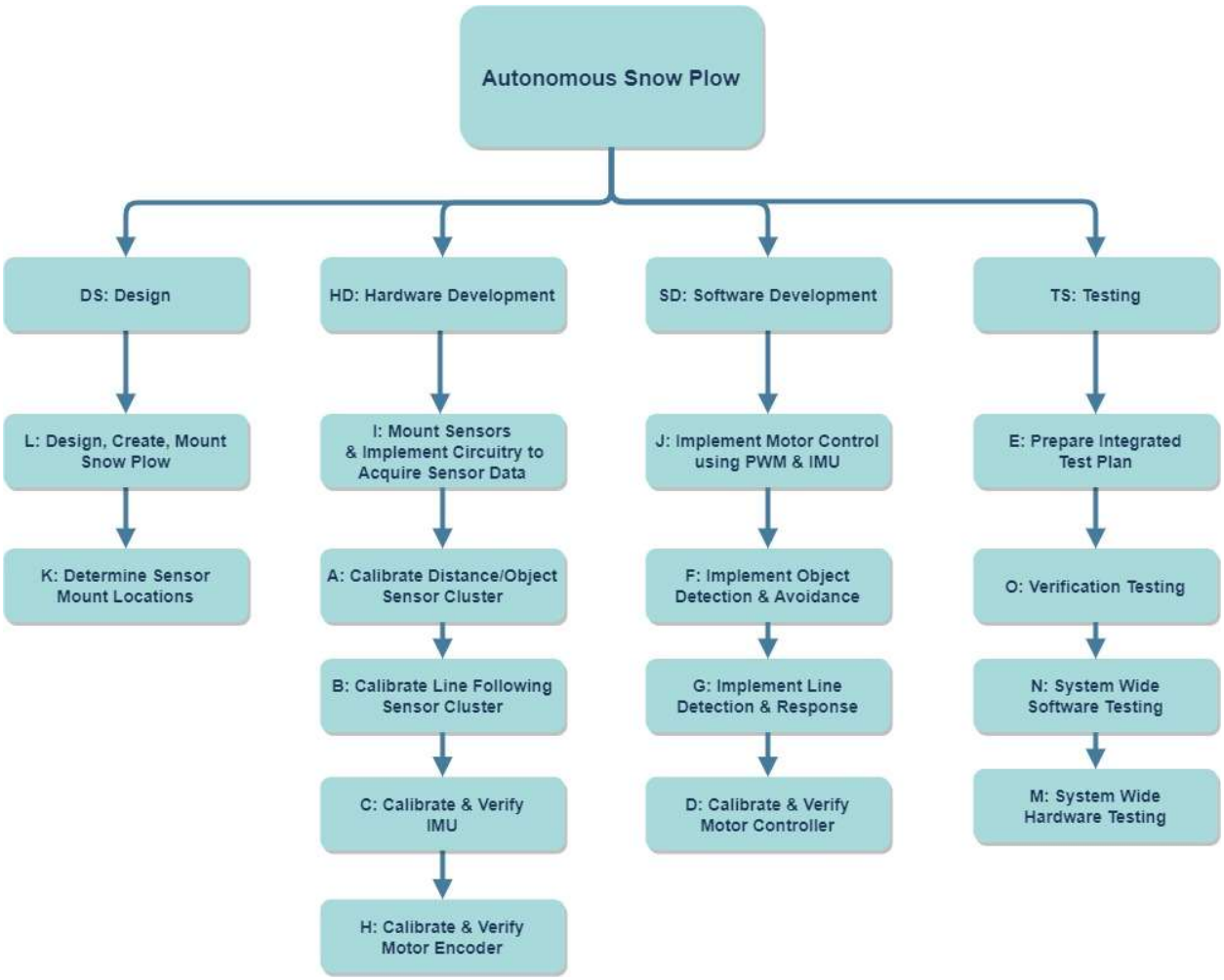


Figure 1: Work Breakdown Structure

Test Plan

WBS Item	Requirement connected to WBS	Verification Method	Pass Condition	Fail Condition
A	1	Test object detection at known distances and compare against reported distances.	Error below 5%	Error above 5%
B	2	Test line detection for a black line on white using printed line and minimize latency.	Latency is within 10 ms	Latency is above 10 ms
C	1,2	Test IMU for known direction and speed values and compare against reported values. Pass if below, fail if above.	Direction heading error is below 5% Speed error is below 10%	Speed error is above 10% Direction heading error is above 5%
D	4,5	Increment controller output in discrete steps and measure motor speed output.	Error is below 5%	Error above 5%
F	1,4	With vehicle wheels raised above the ground, pass objects within 10 cm of the sensor, and observe vehicle response.	Response time is below the time taken to get within 10 cm of an object	Response time is above the time taken to get within 10 cm of an object
G	2,5	With vehicle wheels raised above the ground, pass a line under the sensor, and observe the vehicle's response.	Response time is below the time taken to cross the line	Response time is above the time taken to cross the line
H	3,4,5	Record time to cover a known distance and compare calculated speed to measured speed.	Error is below 5%	Error is above 5%
J	3,4,5,	Pass input to change motor heading by known quantity and measure the vehicle's speed and modified direction.	Error is below 5%	Error is below 5%
I	1,2	Once sensors are connected to the microcontroller, ensure the inputs are received in the same way as individual tests.	Sensor readings are within 2% of values recorded from initial readings	Sensor readings are outside 2% of values recorded from initial readings

L	6	Hold the plow while pushing ping-pong balls out of the way and ensure the balls move out of the way. Once mounted, ensure the balls do not block sensors or vehicle movement.	Plow pushes snow out of the path of the vehicle within 2 seconds	Plow does not push snow out of the path of the vehicle within 2 seconds

Table 2: Test plan for each task and satisfied requirement

Schedule

Schedule Network Diagram

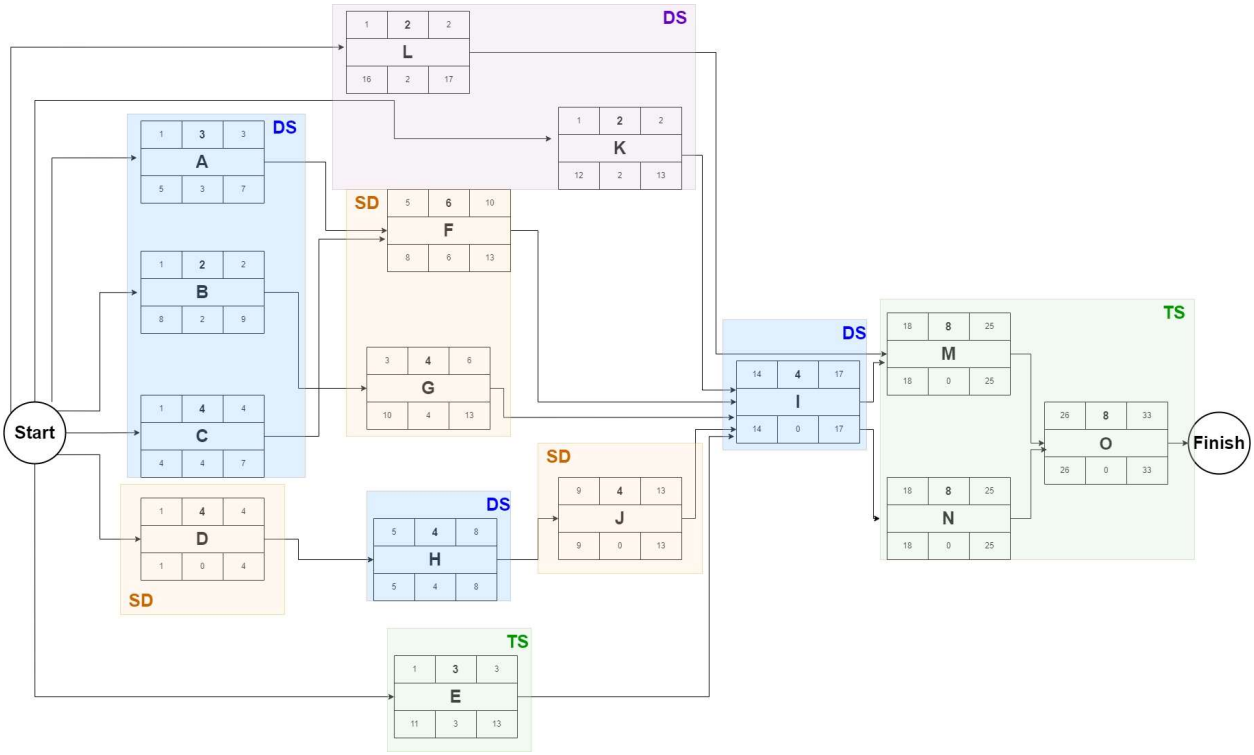


Figure 2: Schedule Network Diagram

Gantt Chart

Figure 3: Gantt Chart

Cost

Baseline

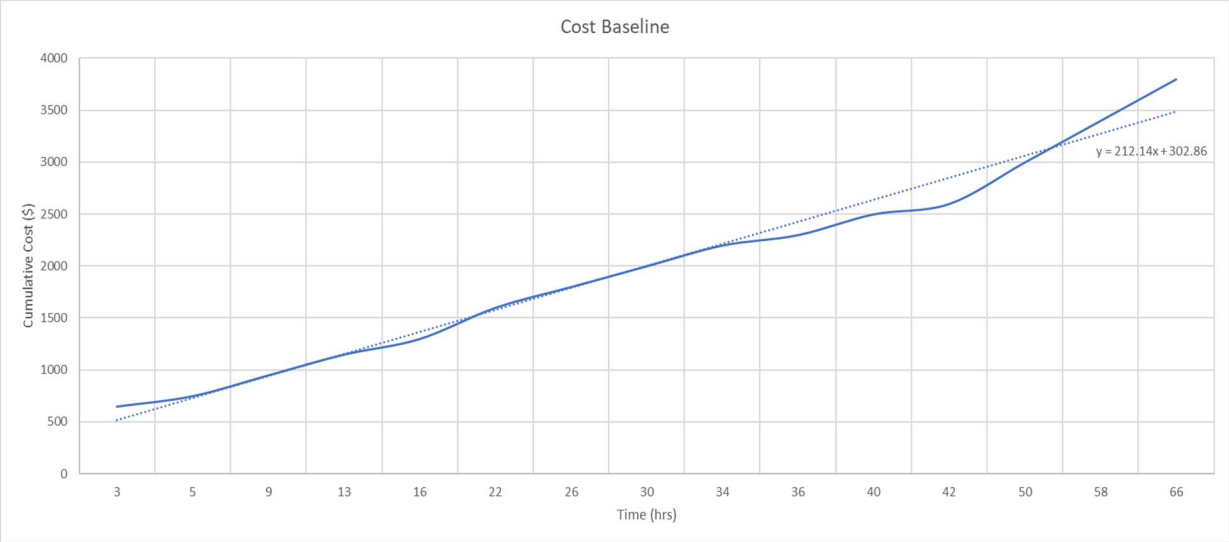


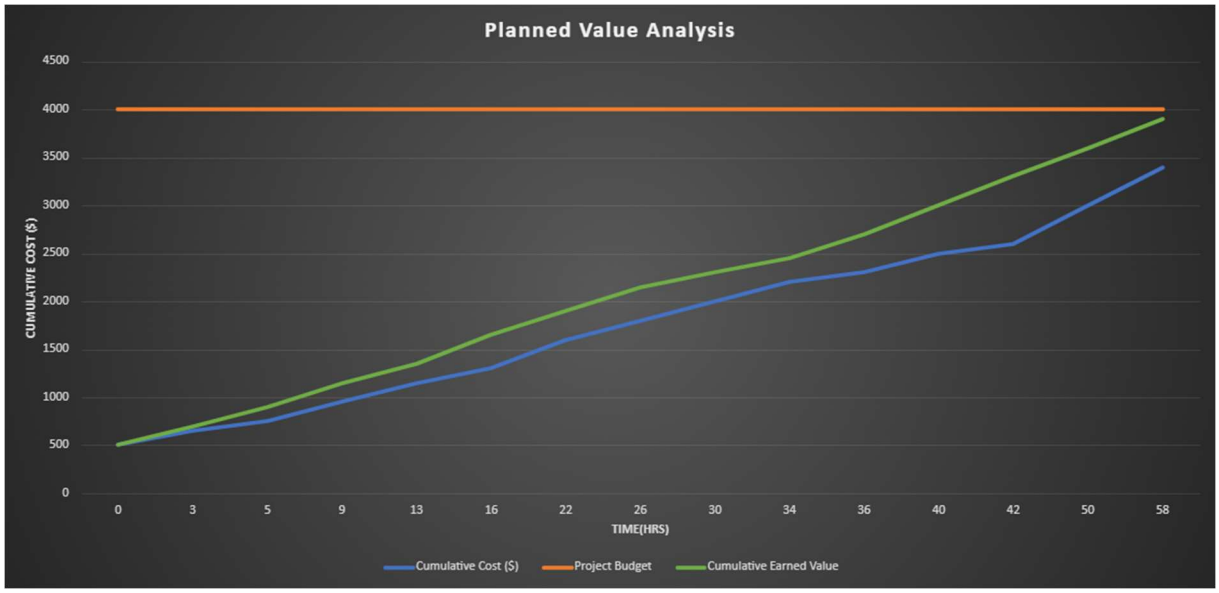
Figure 4: Baseline Cost Estimate

Cost Time Analysis Table

Cost ($)	Cumulative Time	Cumulative Cost ($)
500	0	500
150	3	650
100	5	750
200	9	950
200	13	1150
150	16	1300
300	22	1600
200	26	1800
200	30	2000
200	34	2200
100	36	2300
200	40	2500
100	42	2600
400	50	3000
400	58	3400
400	66	3800

Planned Value Analysis

The project budget was estimated to be around CAD 4000, which is the sum of the cumulative cost with a $200 dollar buffer. At completion, the cumulative sum spent on the project amounted to CAD 3900. This value was calculated by monitoring the hours of work put into each activity from the start of the project. A more detailed depiction of values used to generate the graph shown below can be found in Table 3 below.



Cost ($)	Cumulative Time	Cumulative Cost ($)	Cumulative Earned Value
500	0	500	500
150	3	650	700
100	5	750	900
200	9	950	1150
200	13	1150	1350
150	16	1300	1650
300	22	1600	1900
200	26	1800	2150
200	30	2000	2300
200	34	2200	2450
100	36	2300	2700
200	40	2500	3000
100	42	2600	3400
400	50	3000	3600
400	58	3400	3800

Table 3 Cumulative Earned Value

Human Resources

Responsibility Assignment Matrix

	Responsibility			
WBS Code	Item	Aksh	Anish	Eline
A	Distance Sensor Calibration	A		P
B	Line Sensor Calibration	A	P	
C	IMU Calibration		A	P
D	Motor Control Board Calibration	P		A
E	Integrated Test Plan	P	A	
F	Object Response	A	P	
G	Line Response	P	A	
H	Motor Speed Sensor Calibration	A		P
J	Motor Control	A		P
K	Sensor Design	A	P	
I	Sensor Integration	P	A	
L	Plow Design		P	A
M	Hardware Testing	A		P
N	Software Testing		P	A
O	Verification Testing	P	A	A

Table 4: Responsibility Assignment Matrix
Legend: A - Approver, P - Primary

System Architecture

The figure below shows the behaviour of the system and the interactions that happen within the robot. Each sensor would have a header file and a C++ file for the behaviour implementation. These C++ sensor files will then be included and called in one Arduino file which will serve as the controller. The figure also shows the communication protocols between the Arduino and the sensors, as well as the functions implemented within the various nodes.

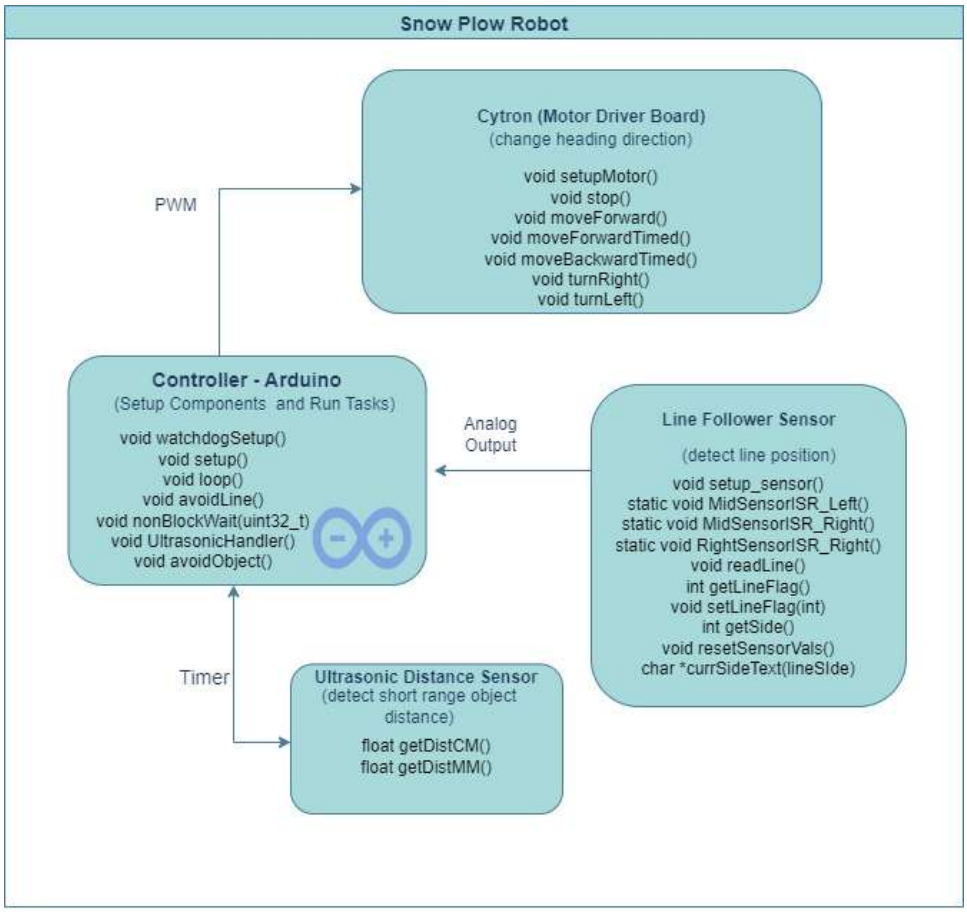


Figure 4: System Architecture Diagram

The Arduino is the brain of our system. It is responsible for setting up the watchdog timer and all the other sensors connected to it. It utilizes the functions of the sensors to avoid lines and objects. The Arduino has an Ultrasonic Handler which is an Interrupt Service Routine (ISR) that runs whenever an object is detected. It also has a non-blocking wait function so that any interrupts can interrupt and start their corresponding ISR if necessary. The Arduino communicates with the Distance Sensor using timers, Line follower sensor by analog outputs, and motor driver board using pulse width modulation.
The Cytron Motor Driver Board is the board that controls the motors of the robot. Functions were created to setup the wheels, change directions (turn left and turn right), stop, and move forward. The timed move forward and timed move backwards functions were created to allow the robot to move forward or backwards for a short amount of time to assist in path corrections.

The Line Follower Sensor is responsible for detecting boundary lines. It has functions for setting up the various sensors and reading from the individual sensors on the board. It sets a flag when there is a line and allows for reading that set flag. It can also get the exact side of the sensor the line is on.
The Ultrasonic Distance Sensors simply return the distance to the sensors.

State Chart

The following state machine diagram shows the planned responses to various inputs from the different sensors to avoid colliding with objects or crossing line boundaries. When the vehicle begins it will move forward until either a line or object is detected. When one of those is detected, the vehicle will follow the states to avoid the obstacle or remain within the lines. Once the obstruction has been avoided, the system will always return to its original course due to the design of the state machine. From there, it will continue forward until another obstruction is encountered or the 5-minute timer has passed.

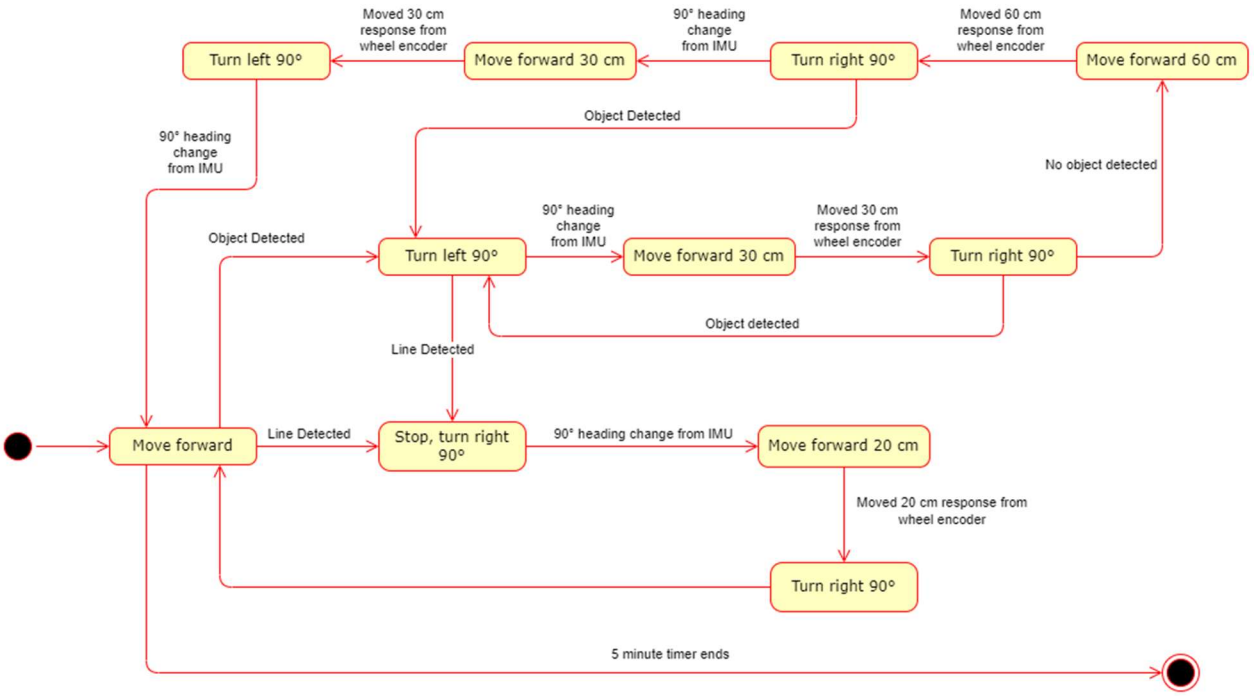


Figure 5: State Machine

Sequence Diagram

The following sequence diagram shows the communication between the different actors when the line sensor or the object sensor reports an obstruction. Once detected, the sensors will send a signal to the Arduino, which will follow the state machine from Figure 5: State Machine to respond appropriately. The responses from the assorted intrinsic passive sensors will provide the needed information to follow the state machine, and it will follow the order described in the sequence diagrams shown below.

Figure 6: Line Sensor Sequence Diagram below shows the typical exchange between the different components when a line is detected. The line sensor sends a signal to the Arduino to let the main program know that a line was detected. When the Arduino detects this signal it will signal the motors to stop the snow plow. Once the plow has stopped, the controller sends a signal to the motor controller to

turn the vehicle around using a pre-defined turn function. Once the vehicle has turned around 180°, the vehicle continues to move forward.

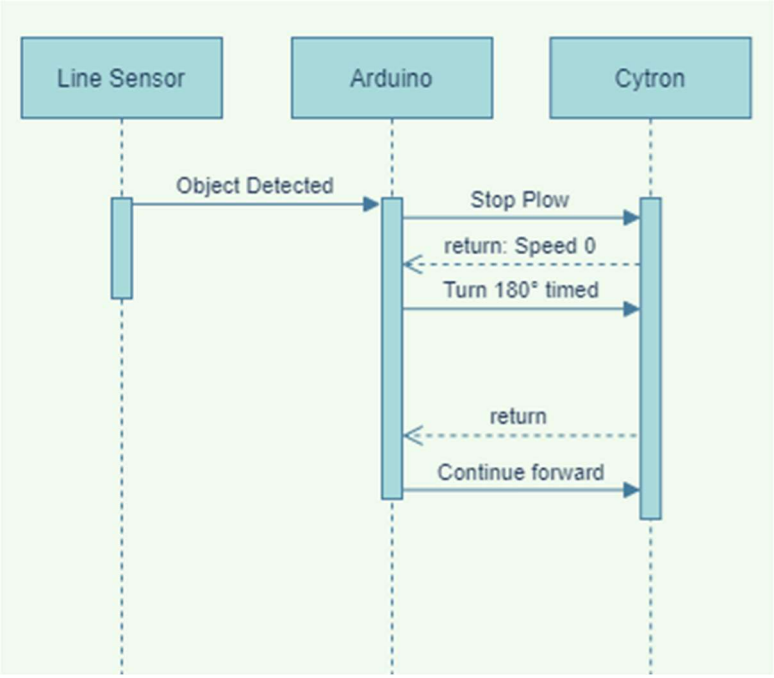


Figure 6: Line Sensor Sequence Diagram

Figure 7: Distance sensor sequence diagram below shows the typical exchange between the different components when an object is detected. The object sensor sends a signal to the Arduino to let the main program know that an object was detected. When the Arduino detects this signal, it will signal the motors to stop the snow plow. Once the plow has stopped, the controller sends a signal to the motor controller to turn the vehicle around using a pre-defined turn function. The direction of the turn is dependent on which ultrasonic distance sensor triggered the event since they are mounted on the left and right side of the vehicle. Once the vehicle has turned around 90°, the vehicle continues to move forward.

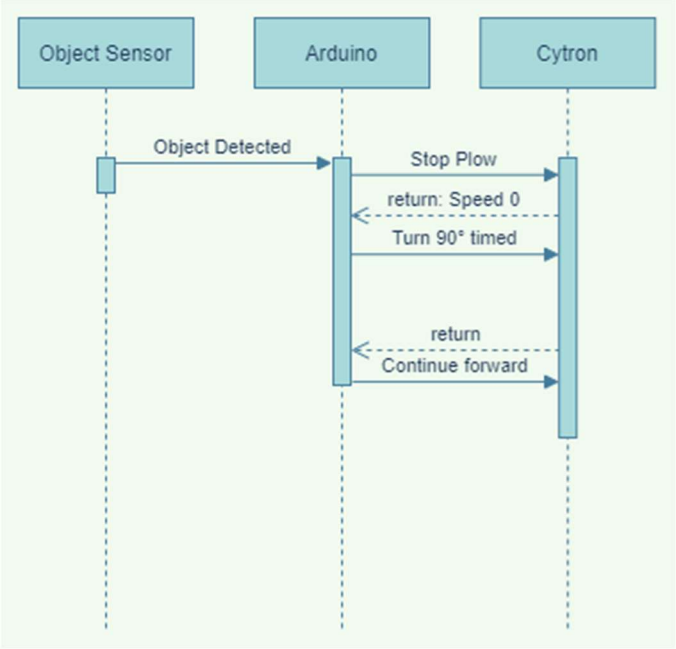


Figure 7: Distance sensor sequence diagram

Watchdog Timer Use

A watchdog timer is necessary to ensure that when a function takes too long to execute, or a subroutine hangs, the program returns to its default behaviour. In our Arduino program, the watchdog timer has been set up to reset the system if it has been longer than 2 seconds since the previous time it has been serviced. In the code snippet shown in Figure 8, the watchdog timer has been enabled in the setup loop. Whenever the main loop is called, it resets the watchdog timer so that it may start counting again. If a function takes longer than 2 seconds to complete, the watchdog will stop that function's execution and return to the main loop.

```
1  #include "UltrasonicSensor.hpp"
2  #define ULTRASONIC_BOUND 11.000000
3
4  volatile boolean US_FLAG_1; //detection flag to be checked
5  volatile double US_OBJECT_DIST; //distance of object detected
6  UltrasonicSensor US1(1);
7
8  void watchdogSetup(){}; //setup watchdog
9
10 void setup()
11 {
12   watchdogEnable(2000); //enable watchdog timer to check in 2 sec intervals
13   US1.setupTimer(); //starting ultrasonic sensor
14 }
15
16 void loop()
17 {
18   watchdogReset();
19   if (US_FLAG_1) //checking if flag is raised
20   {
21     US_FLAG_1 = 0;// Reset the flag,
22     printf("Object Detected at \r %f cm \n", US1.getDistCM());
23   }
24
25 }
26
```

Figure 8 Watchdog Timer Example

Control Charts by Requirement

The following control charts show the results of testing and describe values for the average and standard range of operation for this vehicle, categorized by each of the original requirements. The charts show the range of values for each test performed, the overall average from those results and supporting Upper and Lower Control lines that describe the standard operating range. Values within this range are considered valid, and values outside this range will be considered erroneous and will require further investigation.

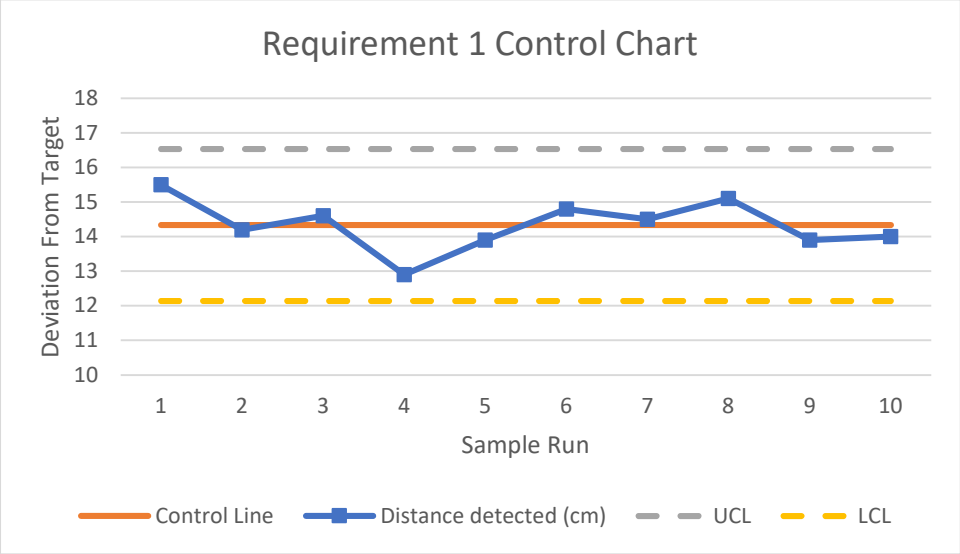


Figure 9: Requirement 1 Control Chart

The chart above shows the testing values for the requirement that the vehicle shall begin corrective action when an object is detected within 15 cm. The distance at which the object was detected and corrective action was taken is shown above for 10 test runs.

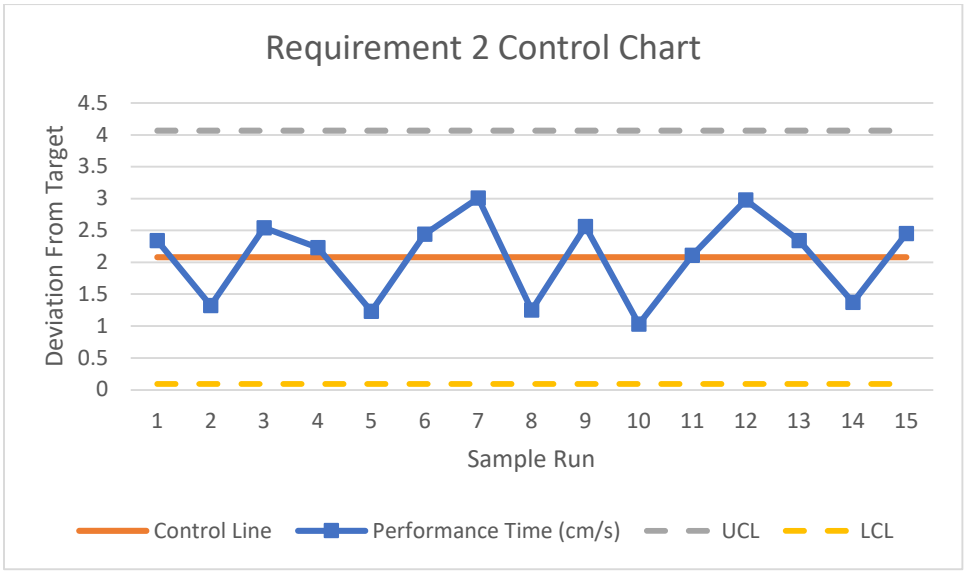


Figure 10: Requirement 2 Control Chart

The requirement above shows the performance time of the robot during multiple testing iterations. This test was conducted to meet the requirement of the robot turning and staying within the boundaries upon detecting an object. The control chart above shows the performance times for 15 sample runs.

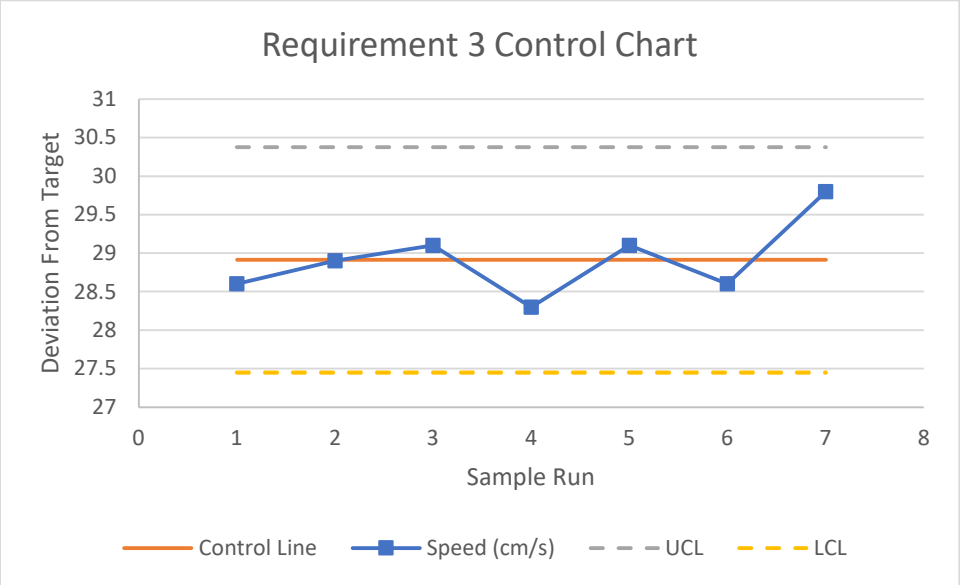


Figure 11: Requirement 3 Control Chart

The above control chart illustrates the performance of the robot when testing for requirement 3. The test was conducted to ensure that the robot does not move faster than 30 cm/s when no object or lines are detected. The test was conducted for 7 sample runs.

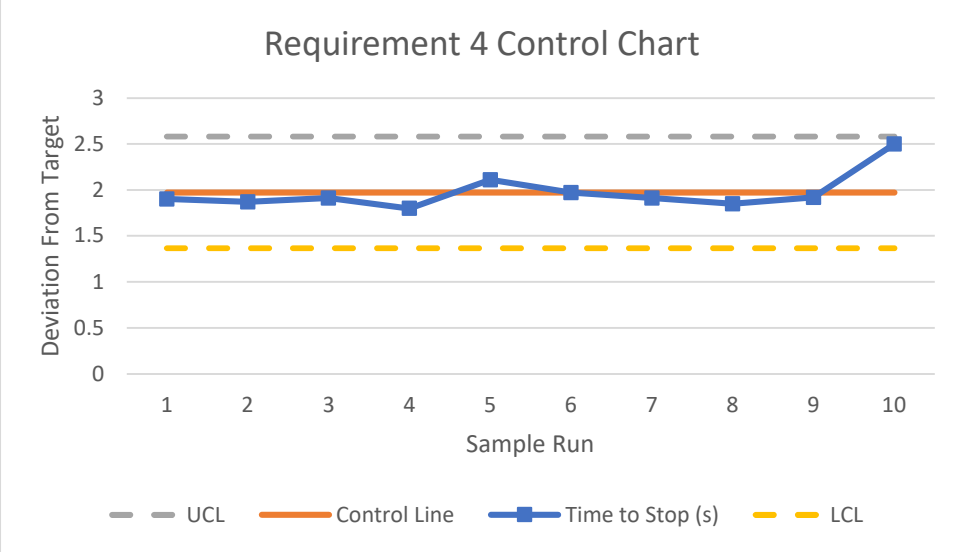


Figure 12: Requirement 4 Control Chart

The chart above shows the testing values for the requirement that the vehicle shall stop within 2 seconds when an object is detected. The stop time for when an object was detected is shown above for 10 test runs.

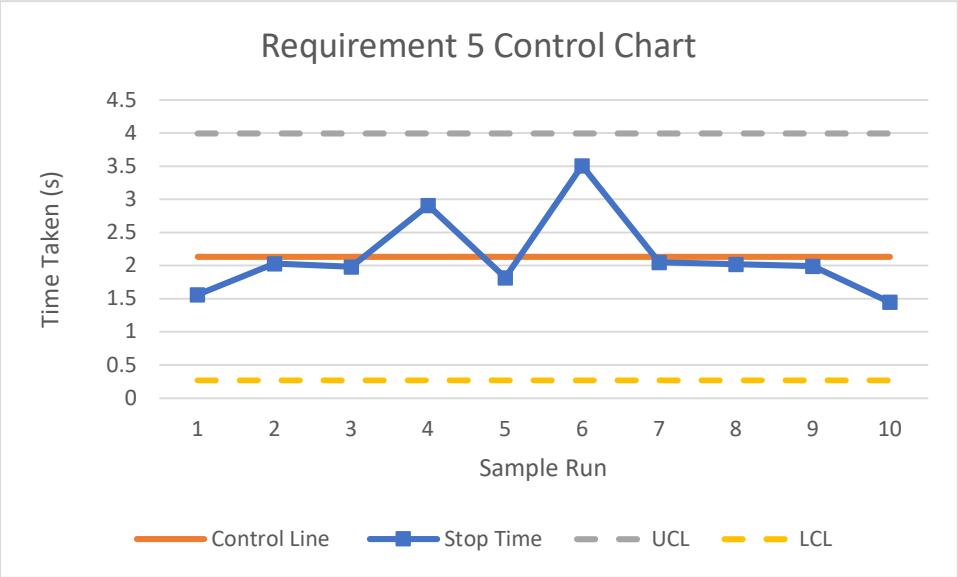


Figure 13: Requirement 5 Control Chart

The requirement above shows the testing values for the requirement that the vehicle would stop moving within 2 seconds when a line is unavoidable. The chart above shows the testing values and improvements made in 10 runs.

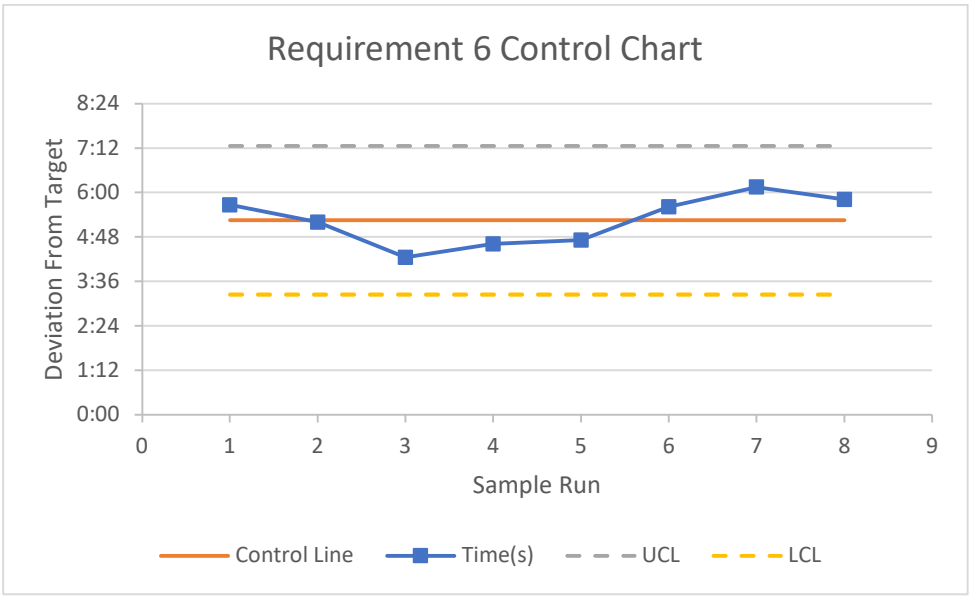


Figure 14: Requirement 6 Control Chart

The control chart above depicts testing values for requirement 6. Each test run was conducted to observe whether the robot was able to clear the testing ground of snowballs in under 5 minutes. The test was conducted for 8 sample runs.

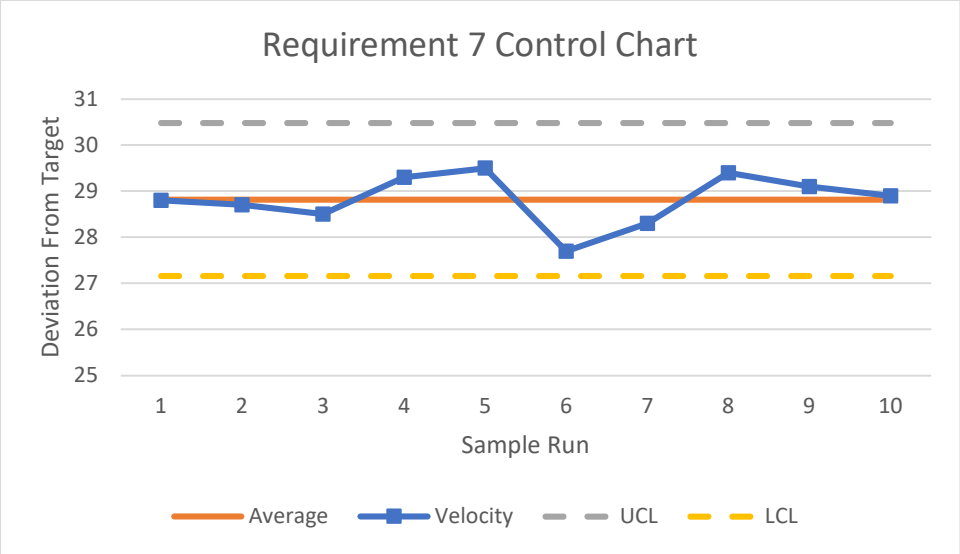


Figure 15: Requirement 7 Control Chart

The chart above shows the testing values for the requirement that the vehicle shall not exceed a speed of 30cm/s. The speed of the vehicle at full power is shown above for 10 test runs.

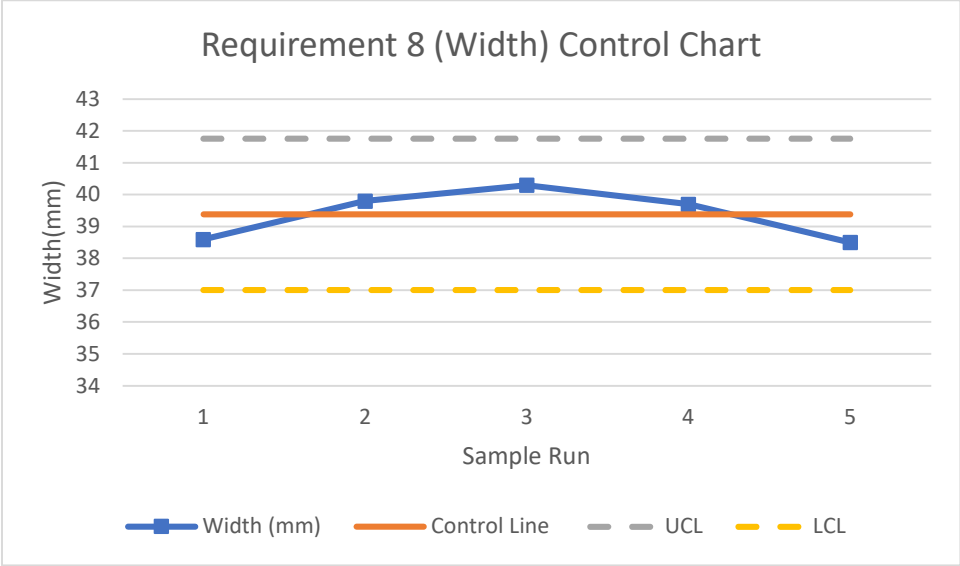


Figure 16: Requirement 8 (Width) Control Chart

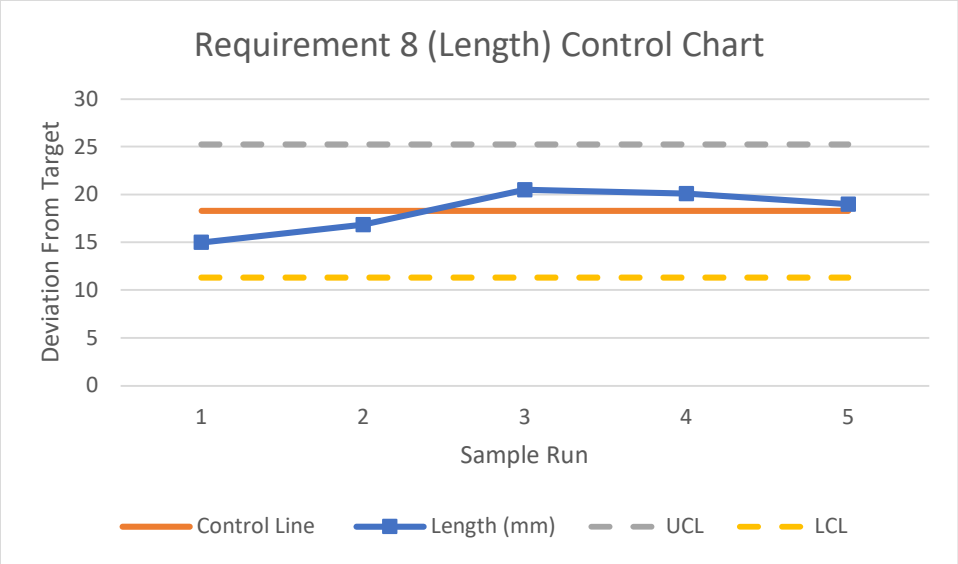


Figure 17: Requirement 8 (Length) Control Chart

Figure 16 and Figure 17 above show the 5 iterations of the design of the snow plow in bid to meet the requirements of being 40mm in width and 20mm in length. The final design (iteration 5) has a width of 38.5mm and a length of 19mm.

Testing

System Testing

To create a project that satisfies customers' needs, time is dedicated to testing each component individually to determine what components work best to achieve the project goal. This process also helps understand the limits of each component and where to mount it on the robot to get the best out of that component.

The initial goal was to utilize distance sensors, line follower sensors, an IMU and a motor encoder. Upon making a priority list, and testing each component individually, it was found out that the object detection sensor measured objects accurately within 1cm to 15 cm, the line follower sensor identified a line when its values were above 900 and below 600 when there was no line present. To reduce the complexity of our robot the IMU and the motor encoder were not used, as adding more sensors stalled the responsiveness of other components. This format of sensor testing provided information about the how the sensors get readings and used that information to decide ideal mounting points for each sensor at parts of the robot where the readings are least affected by other sensors or devices. Another aspect of hardware design included various plow designs to determine the best way to remove the balls from the arena. A V-shaped plow design was chosen to efficiently clear the bound region of snow.

When integrating the software with the hardware, issues with compatibility are to be expected. Ensuring that all the components interact accordingly raises its own set of complications. For each hardware component, a corresponding software component was programmed. Once each software

component and hardware component were ready to be tested, they were first tested separately. For example, when the software written to handle the ultrasonic sensor detected an object less than 15 cm away (and greater than 0 cm), the handler sets a flag indicating that an object was detected. When the flag is raised and seen by the main loop, it is written into the console that a flag was and the flag is reset. Each individual software/hardware pair were tested in a similar fashion. The testing method was repeated for many iterations with minor tweaks between each iteration until the results of testing were deemed satisfactory. At this point, each software/hardware component pair were begun to be tested in concert with each other – if they could interact with each other in the overall system.

When testing the integration of the overall system, a similar set of testing methodologies were used. Verification methods for each of requirements were developed and subsequently performed. Building on the previous example, if the left ultrasonic sensor detected an object in the range of 1 to 15 cm, a flag was raised and vice versa for the right ultrasonic sensor. When the raised flag is seen by the main loop, a subroutine that turns the snow plow robot 90 degrees – left or right according to the raised flag – is performed to prevent collision with the object. This testing scenario was repeated with tweaks and adjustments between iterations until it met the pass condition: meaning that the requirement was met. The pattern of developing a verification method, performing the method, and making changes until the requirement was deemed to be met was used for all requirements.

Customer Testing

In the final stages of testing, with a minimum viable product (MVP) ready to present to the customer, an arena was built to simulate realistic conditions, and in this arena a demonstration of the product was shown to the clients. In this region, obstacles to be avoided were present and the snow to be cleared was depicted using a collection of plastic balls. At the beginning of the demonstration, the vehicle was placed at an ideal starting point and began to follow the algorithm to clear the bound region of snow. As the vehicle traversed the bound region, it pushed the snow out of the region, stopping and turning to avoid objects and lines. When objects were detected, the vehicle would turn to change the path it followed, and when a line was found, the vehicle would simply turn around and continue to push snow within the bound region. During a test period of 5 minutes, the vehicle was able to successfully remove around 35 balls from the bound region, and 55 balls in the second round. While removing the balls, the vehicle exited the bounds once, and did not collide with any objects. This can be seen as a successful result as the vehicle encountered the object multiple times and approached the region boundary several times as well. This concluded a successful demonstration of the product to the client and showed the reliability of the system within this five-minute period to be sufficient within the constraints set by the original requirements.

As an added insight, the implementation of the system's code and structure was also revealed following the product demonstration. The implementation for each type of sensor and motor were shown and the correlation between the hardware and the software was described. This was to help the client gain a deeper understanding of the inner workings of the product and showed a translation of the diagrams and plans laid out in the progress report into a tangible functional system, one that was proven to be able to fulfill the requirements that were agreed upon between the client and the team.

Appendix A

The following is a link to the GitHub code repository which contains all the code used in the development of the Vanilla Ice Snow plow.

https://github.com/SYSC4805-Fall2022/sysc4805_term_project-vanillaice_l1g5