Autonomous Snowplow
Final Report

SYSC 4805 - L1
Group 10 - Zomp

Date: 12/09/2022

Team Members:
Ryan Fife - 101103898
Yunzhou Liu - 101027110
Azizul Hasan - 101124159

1. Update the Project Proposal and Progress Report documents, addressing all the raised concerns.

Overall objective

Our mission is to clear snow, using our Autonomous Snowplow, from an area bounded by a closed black route. There will be both stationary and moving obstacles in the area. The snow removal device must finish its assignment without colliding with any obstacles. To achieve this goal, our team proposes the development of an intelligent robot equipped with a plow. To adapt to the layout of the target area, the robot will be equipped with various sensors. Our team has done sensor research and testing to ensure that the gathered sensors and motors will meet the required functionality. Among the functional objectives, the robot will also be implemented within the required dimensions, linear velocity limitation, and performance requirements. It is critical to delivering on these requirements as our client (Prof. Mostafa Taha) has invested resources, time and effort and it is our duty to implement our clients' demands. The project has been broken down into multiple different aspects such as navigation, detection, avoidance and validation testing. All of these aspects are divided amongst the 3 developers to ensure efficiency and deadline adherence. To validate that our product meets customer requirements testing will be carried out concurrently with development. Additionally, the extra time has been allotted near the end of development for validation testing and debugging.

List of deliverables

Table 1: Indexed deliverables for project [Group 10]

Code	Date	Deliverable	Description
D1	Nov 11, 2022	Progress Report	The team will deliver a report detailing achieved progress up to that point.
D3	Nov 28, 2022	In-Lecture Presentation	The team will deliver a 13-minute presentation with 2 additional minutes for questions. The presentation will detail our approach to the problem and our developed solution.
D4	Dec 1, 2022	Testing and Demo	Development officially finishes. The team will validate the

			developed solution with stakeholders. 10 minutes will be spent on a robot demo, 5 minutes will be spent on a code review.
D5	Dec 9, 2022	Final Report	The team will deliver a final report and finalized code. The report will contain all the content from the progress report as well as retrospective analysis and testing results.

List of milestones

Table 1B: Indexed milestones for project [Group 10]

Code	Date	Deliverable	Finish Criteria	Progress
M1	Nov 16, 2022	Design	Circuit diagrams, and high-level software diagrams completed.	Finished
M2	Nov 10, 2022	Frame	Plow hull produced and attached to the robot.	Finished
M3	Nov 23, 2022	Navigation	Fully assembled robot can turn, go forward, and reverse.	Finished
M4	Nov 16, 2022	Detection	Robot can detect moving and stationary obstacles and the line perimeter.	Finished
M5	Nov 23, 2022	Avoidance	Robot can avoid collision with moving and stationary obstacles. The robot can stay	Finished

			within the line perimeter.	
M6	Nov 30, 2022	Testing	Robot has been fully tested as per the testing plans.	Finished

As requested in the proposal feedback, all milestones from the group's gantt chart have been included in the list above.

Scope

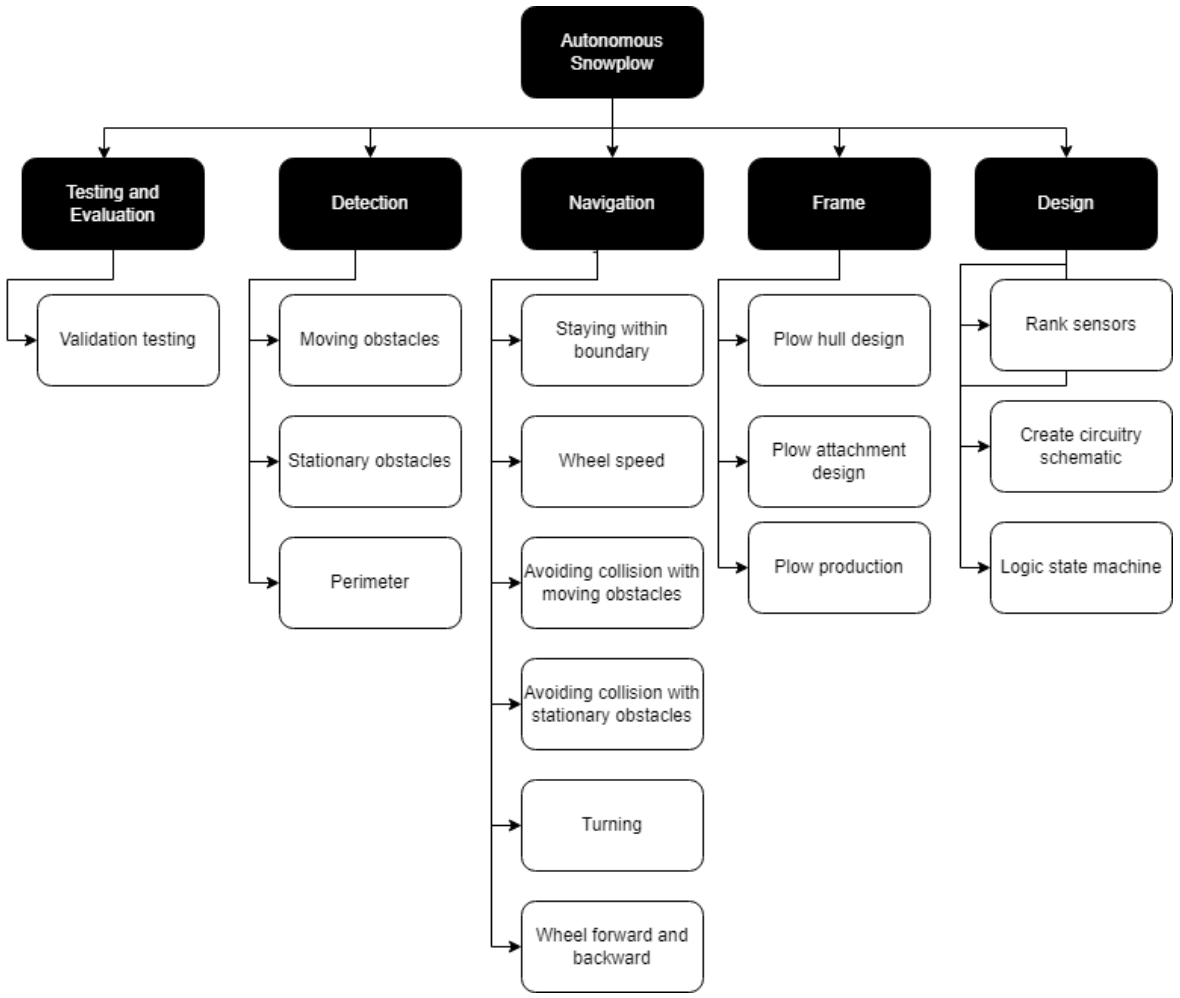
The following section details the scope of the proposed project. The opening subsection states the requirements of the project. The section then closes by detailing the project's activity breakdown and testing plan

Requirements

Table 2: Indexed requirements for the project [Group 10]

Code	Requirement
R1	The robot shall remove all spheres with a diameter of 42.6mm from a 6 m^2 area outlined with a black perimeter within 5 minutes
R2	The robot's power supply shall supply the required voltage load to its peripherals for proper operation for at least 5 minutes
R3	The robot shall travel with a maximum linear velocity of 30 cm/s
R4	The robot frame shall occupy a maximum size of 216 x 252 x 150 mm
R5	The robot shall start operation from within the black perimeter.
R6	The robot shall not collide with obstacles that are moving or stationary.
R7	The robot shall be able to continue normal operation after avoiding collisions.

Activities



WBS for autonomous snowplow [Group 10]

Table 3: Work activity items [Group 10]

Code	Title	Description
A1	Perimeter detection	Robots should only remove debris from within the perimeter. Therefore the robot should be able to detect the perimeter. Choose and place a sensor for perimeter adherence. Write code for detecting perimeter.
A2	Moving obstacle detection	Robot should detect moving obstacles. Choose and place a sensor for 3D moving obstacle detection. Write code for detecting moving obstacles.
A3	Stationary obstacle detection	Robot should detect stationary obstacles. Choose and place a sensor for 3D stationary

		obstacle detection. Write code for detecting a stationary obstacle.
A4	Wheel speed code	Decide on wheel speed considering the following factors: power draw, direction, sensor reaction time, and performance. The code should supply a proper signal.
A5	Wheel direction code	Code wheel forward/backward function.
A6	Turning code	Robot should turn when needed. Write turning code.
A7	Staying within the boundary	Once the robot detects the perimeter it should alter its movement to stay within the perimeter.
A8	Plow hull design	Design the plow hull, should effectively move balls within the perimeter.
A9	Plow attachment design	Design the attachment that connects the plow to the robot.
A10	Plow production	Procure the physical plow. Attach the plow to the robot.
A11	Rank sensors	Rank sensors based on power draw, and performance.
A12	Code stationary obstacle avoidance	Write the code for avoiding a stationary obstacle.
A13	Code moving obstacle avoidance	Write the code for avoiding a moving obstacle
A14	Create circuitry schematic	Create a schematic detailing the connections of the circuitry of the entire system. Including Arduino, motor board, motors, and sensors.
A15	Validation testing	Ensure that the assembled robot meets all requirements. Carry out the tests in the testing case section.
A16	Logic state machine	Create a state machine for the robot logic. Including different obstacle detection states, navigation states, data collection states, etc. Perhaps include high-level sequence diagrams for major operations.

Testing Plan

Table 4: Testing plan items [Group 10]

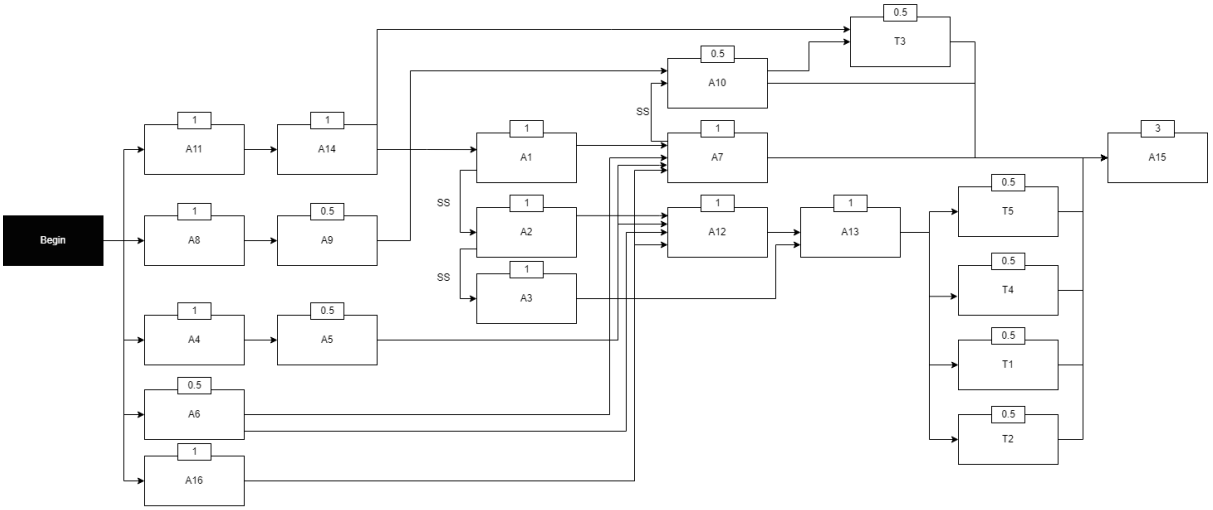
Code	Requirement code	Setup	Pass/Fail
T1	R1 + R3	Set black taped perimeter of 6m^2 on the floor. Place N balls in the perimeter. Place the robot inside the perimeter, and turn it on.	(R1) PASS if the robot removes N balls from the black taped area within 5 minutes. (R3) PASS iff max linear velocity of the robot is 30 cm/s.
T2	R2	Ensure robot wheels are not touching the ground. Ensure all sensors are set up correctly, outputting realistic readings. Turn on the robot.	PASS if the robot can maintain max wheel speed and correct sensor readings for 5 minutes.
T3	R4	Robot is fully assembled with wheels, sensors, and a plow.	PASS if measured dimensions do not exceed 216 x 252 x 150 mm.
T4	R6 + R7	Robot powered on and moving forward then placed on a track with a stationary object in trajectory.	PASS if the robot alters its trajectory resulting in no collision.
T5	R6 + R7	Robot powered on and moving forward then placed on a track with a moving object set to collide with the robot.	PASS if the robot alters its trajectory resulting in no collision.

Schedule

The following section details the schedule of the proposed project. The section contains various schedule graphs and closes with an activity assignment table.

Schedule Network Diagram

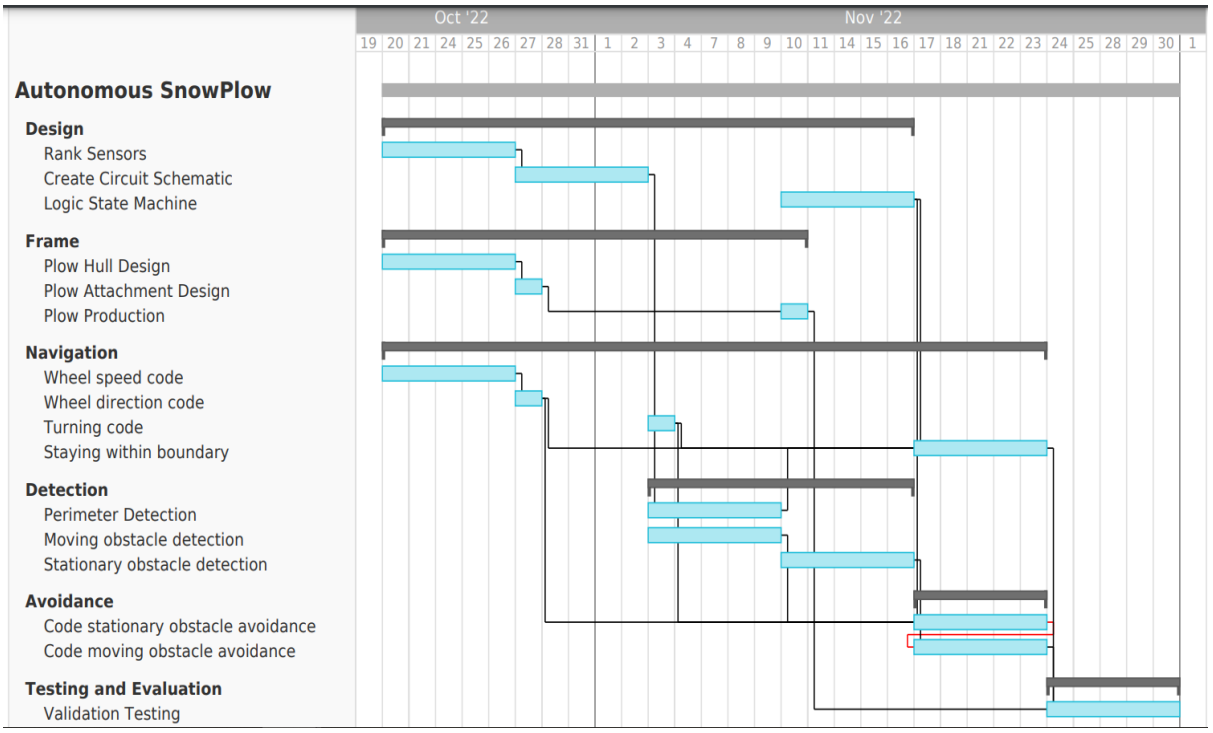
Schedule Network Diagram [Group 10]



The schedule network diagram shown above depicts activity duration as well as its dependencies. Activity duration is specified in lab sessions for a single developer. 1-time unit = 1 lab session or 4 man hours. All activity dependencies shown are finished to start unless specified otherwise.

As requested in the proposal feedback, all testing activities have been included in the SND above.

Gantt chart



Gantt Chart [Group 10]

The Gantt chart allows us to schedule the task with start and end dates. At any given lab session, each member is scheduled with 1 activity. The duration of the activity is already planned using the schedule network diagram and now using the Gantt chart, all the team members know exactly what is due and when exactly is its deadline. The dependencies are also shown which results in a possible sequence of tasks to follow.

Responsibility Assignment Matrix

R = Responsible, A = Approver

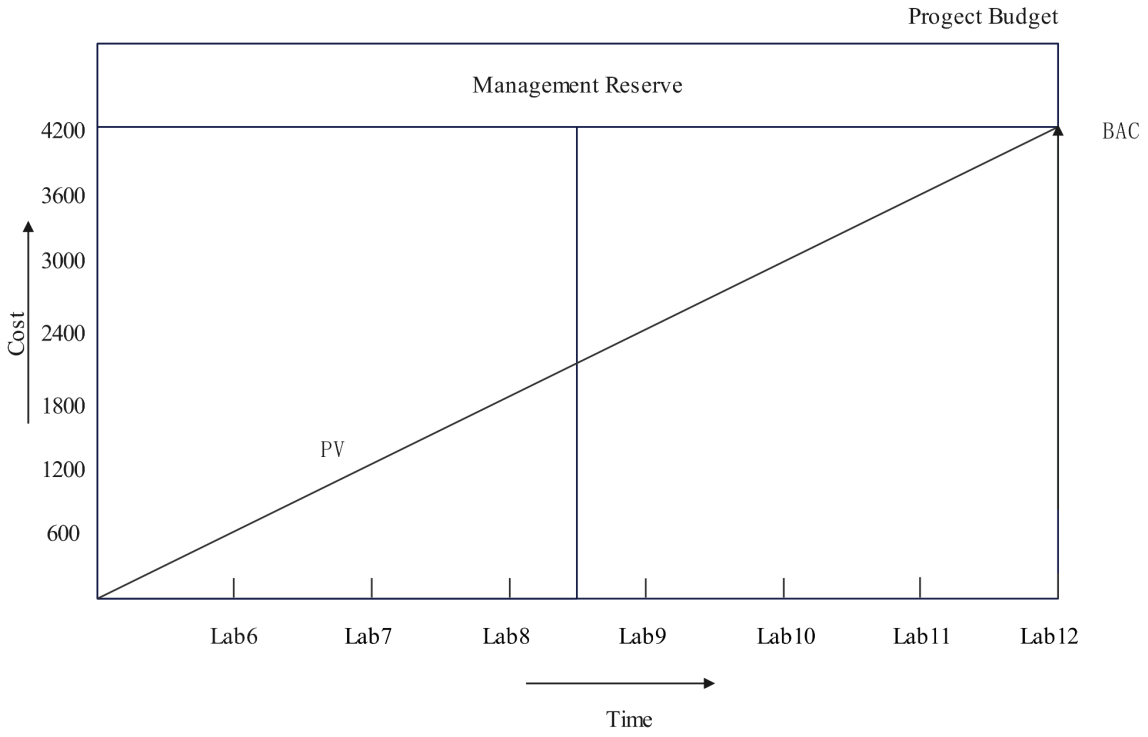
Table 5: Responsibility assignment matrix [Group 10]

Activity	Ryan Fife	Yunzhou Liu	Azizul Hasan
A1 Perimeter detection		A	R
A2 Moving obstacle detection	R		A
A3 Stationary obstacle detection	A	R	
A4 Wheel speed code	R		A
A5 Wheel direction code	R	A	
A6 Turning code		A	R
A7 Staying within boundary	A	R	
A8 Plow hull design	A		R
A9 Plow attachment design		A	R
A10 Plow production	A		R
A11 Rank sensors		R	A
A12 Code stationary obstacle avoidance		R	A
A13 Code moving obstacle avoidance	R	A	
A14 Create circuitry schematic	R	A	
A15 Validation testing	R A	R A	R A
A16 Logic state machine	A	R	

Budget

Cost Baseline

7 more lab sessions after the proposal is due.
7 * 4 hours * 3 developers * $50 / hour/developer = $ 4200



The Cost Baseline diagram describes the budget of our team in the next lab project. The budget for each lab is 600, and a total of 4200 is required. An additional reserve has been included for unforeseen costs and potential plow production costs.

Table 6: Activity cost breakdown [Group 10]

Activity code	Cost - dollars
A1	200
A2	200
A3	200
A4	200
A5	100
A6	100
A7	200

A8	200
A9	100
A10	100
A11	200
A12	200
A13	200
A14	200
A15	600
A16	200

Example computation: Activities projected to occupy 1 time unit (4 man-hours) will cost $200. man-hours * developer cost/hr.

Completion of activities mentioned in proposal

A1 Perimeter detection & A7 Staying within boundary
In this lab, the robot requires running in the area that is around by a black line. Only the Line Follower Sensor can detect the black line. Our group decided to install the sensor in the front of the robot. If the sensor gives a signal to the Arduino Due board, then the sensor will change direction. The program also takes care of adhering to the boundaries in the testing arena. The program is available in our GitHub repository (https://github.com/SYSC4805-Fall2022/sysc4805_term_project-zomp_l1g10/blob/main/AS_main/AS_sensors.h)

A2 Moving obstacle detection & A13 Code moving obstacle avoidance
The robot uses an Ultrasonic Distance Sensor to detect moving obstacles which are other robots. The sensor will find another robot around 20 cm, then send a signal to the Arduino Due board. The robot will have a response.

The robot also uses IR and distance sensors to detect objects. The sensors are positioned in an arc to cover all the blind spots at the front of the snowplow. The program is available in our GitHub repository (https://github.com/SYSC4805-Fall2022/sysc4805_term_project-zomp_l1g10/blob/main/AS_main/AS_sensors.h)

A3 Stationary obstacle detection & A12 Code stationary obstacle avoidance
The Stationary obstacles are different sizes of boxes. The robot uses a VL53L1X sensor to detect obstacles, then change direction. The code is present in our GitHub repo

(https://github.com/SYSC4805-Fall2022/sysc4805_term_project-zomp_l1g10/blob/main/Obstacle_detection/VL53L1Xtimeoff.ino)

The initial idea of utilizing the ToF sensor was altered and instead, are the more simple but effective approach of using multiple IR sensors and a distance sensor. The wide angle of view of the ToF Sensor is reimplemented by aligning the sensors in an arc, which allows all blindspots to be covered. You can find all about the sensor software implementation in the following file
(https://github.com/SYSC4805-Fall2022/sysc4805_term_project-zomp_l1g10/blob/main/AS_main/AS_sensors.h)

A4 Wheel speed code & A5 Wheel direction code

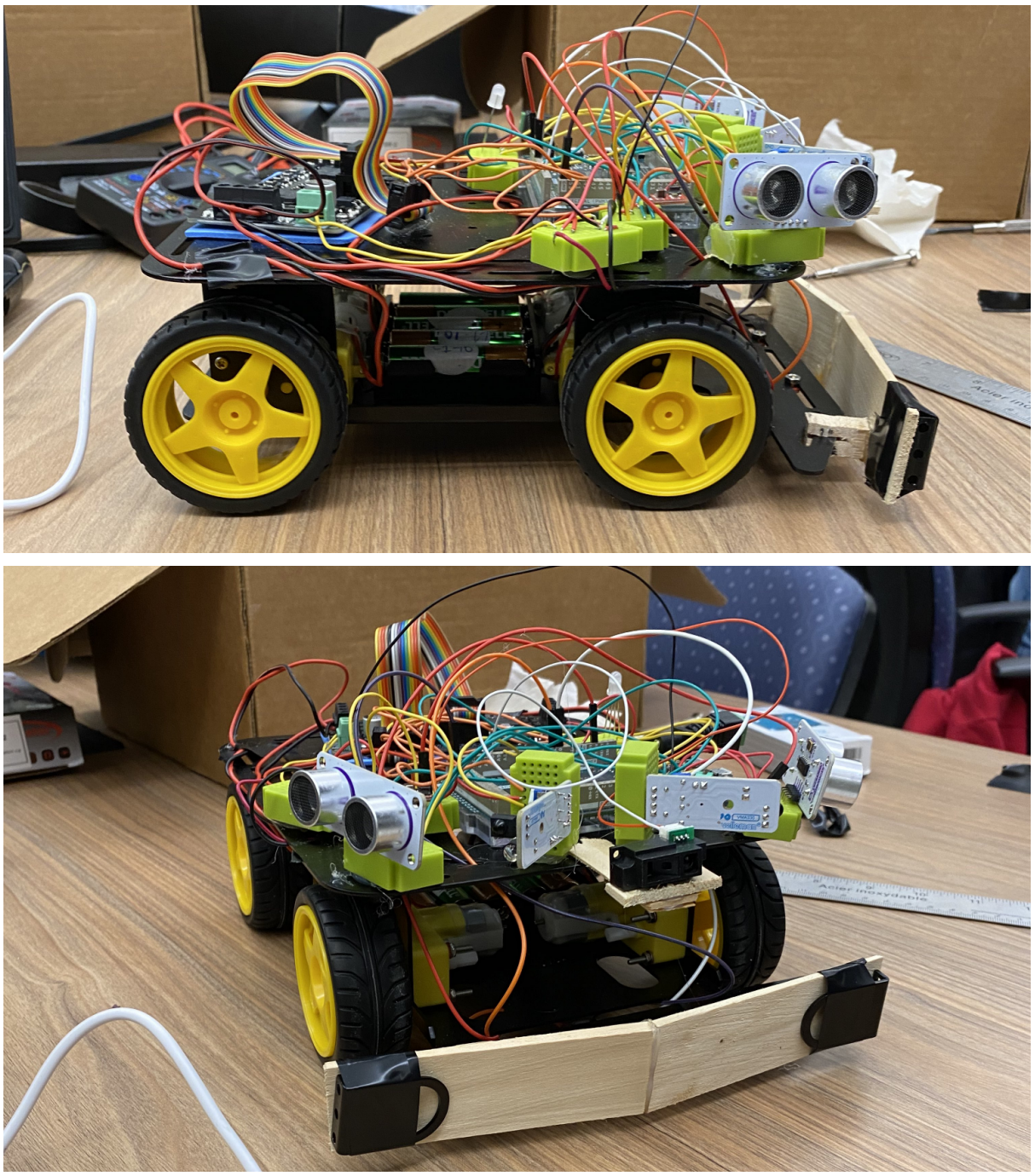
The navigation of the snowplow is managed through the business logic in the wheel program file. The speed and direction is handled by the wheel program which is present in our GitHub repository
(https://github.com/SYSC4805-Fall2022/sysc4805_term_project-zomp_l1g10/blob/main/AS_main/AS_wheels.ino)

A6 Turning code

The turning of the snowplow is handle by the main program. It manipulates the direction and the speed of the wheels and allows turning. Our solution is present in our GitHub repo
(https://github.com/SYSC4805-Fall2022/sysc4805_term_project-zomp_l1g10/blob/main/AS_main/AS_main.ino)

A8 Plow hull design

The plow hull design is planned to be independent of running the Snowplow. The obstacle detection navigation and sensor integration of the snowplow is prioritized. After completion of the navigation and obstacle detection, the hull will be designed.



Robot plow [Group 10]

The picture above shows the finalized production of the plow. Whilst designing the plow the team considered aesthetics, weight, functionality, and staying within the dimension requirement. Functionality-wise the team started with a slightly arched plow that pushes balls to the side. We decided to add two small implements to the plow (attached with black tape in the picture) to flatten the plow. This adjustment collected balls far better than the arched plow alone. Material-wise, the first plow iteration was made of cardboard. Whilst cardboard is lightweight, the plow was not sturdy or aesthetically pleasing. For the final version, the produced plow is made of lightweight, sturdy, more visually-appealing cedarwood.

A9 Plow attachment design

As mentioned in the last section, due to planned priorities of snowplow navigation and obstacle detection, the attachment design of the hull is planned to be completed after hull design, navigation and obstacle detection

For the plow attachment, the group considered attachment modularity, attachment aesthetics, and maintaining plow stability throughout the operation. The final attachment is two clothespin-like maple attachments.

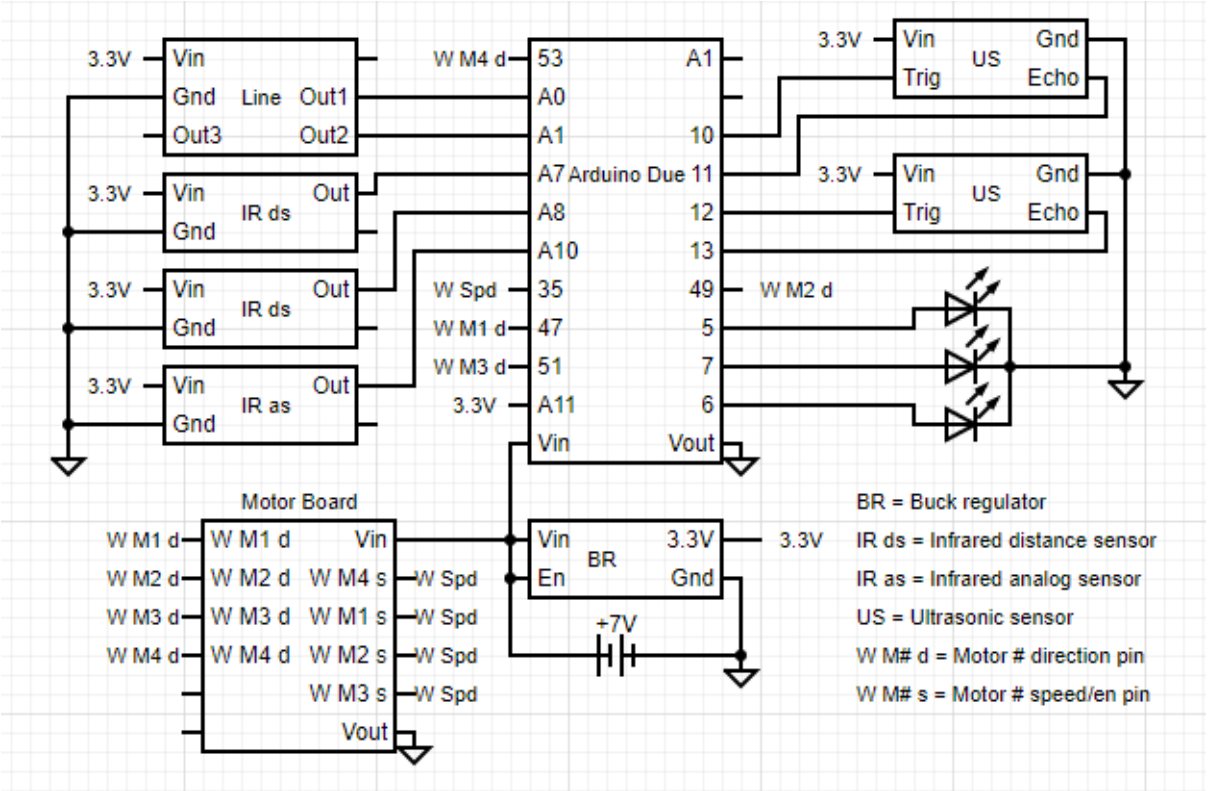
A10 Plow production
The team had ideas about 3-D printing the plow or utilizing the cardboard to engineer a plow that would simulate snow removal (clearing snowballs). As navigation and other critical aspects pertaining to the snowplow, production of the snowplow will provide more attention after hull design, attachment design, and navigation.

The final plow consists mostly of lumber. As such, to produce the plow some saws were used to cut the pieces. Hot glue was used to attach the sawed pieces. The materials needed to produce the plow were approximately 2 dollars.

A11 Rank sensors
According to the data obtained in previous labs. VL53L1X Time of Flight Distance Sensor and Ultrasonic Distance Sensor has the lowest error in a 20cm working environment.

The current implementation uses a combination of ultrasonic, IR and distance sensors. The decision was made as the implementation was effective and functional.

A14 Create circuitry schematic
The following circuit diagram illustrates our hardware setup for the autonomous snowplow.



Updated Circuit Diagram [Group 10]

The diagram above shows the final version of the robot's pinout. The components of the system consist of one Arduino due board, one buck regulator, one motor board, six sensors, and one 7V battery pack. The battery pack powers the DUE, motor board, and buck regulator. The buck regulator supplies a lower 3.3 voltage to all of the sensors. All of the sensors are analog, except for the ultrasonic sensors. For observing the system state an RGB led has been added to the system.

A15 Validation testing

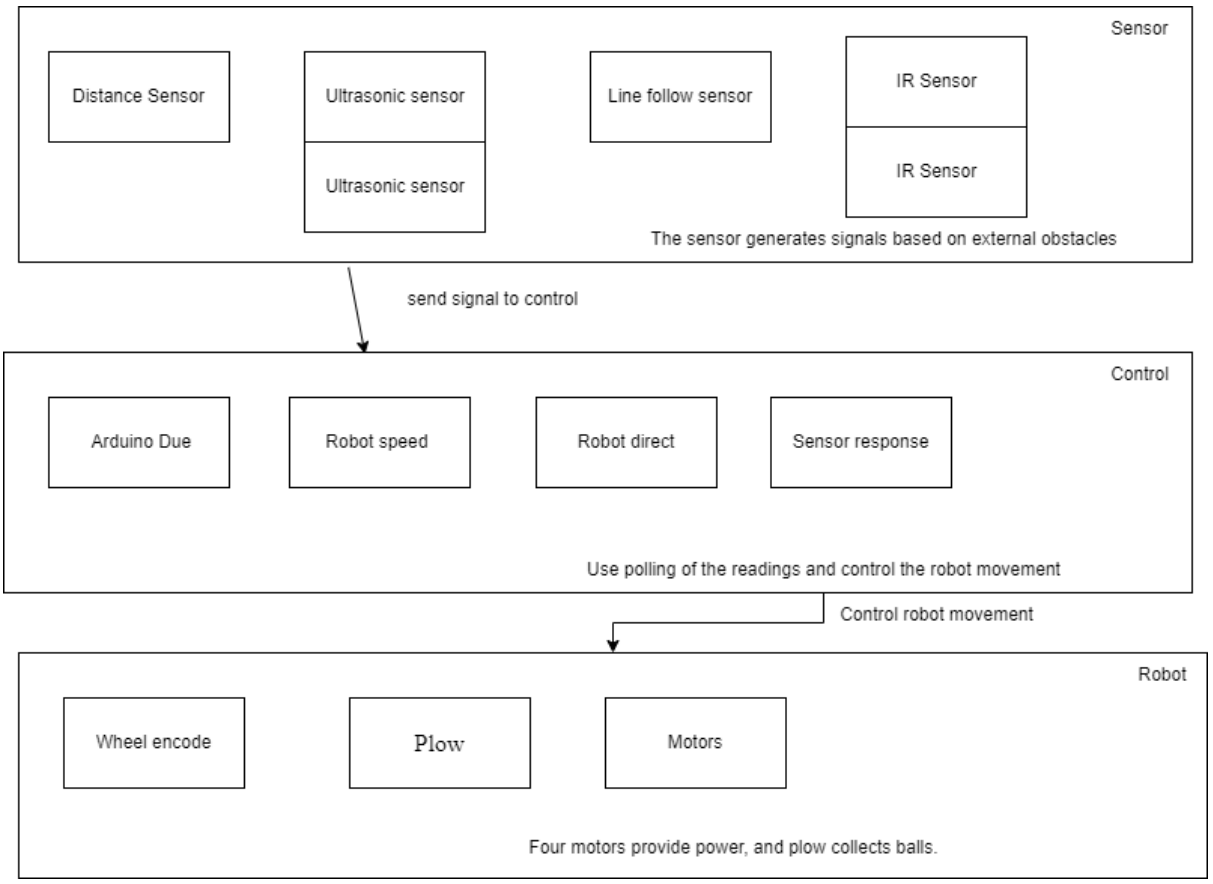
Validation testing is scheduled in the last two labs. This is an overall test of all functions of the robot.

Validation testing results are covered in the ‘Results of Testing’ section.

A16 Logic state machine

The requirement is represented in the state chart section.

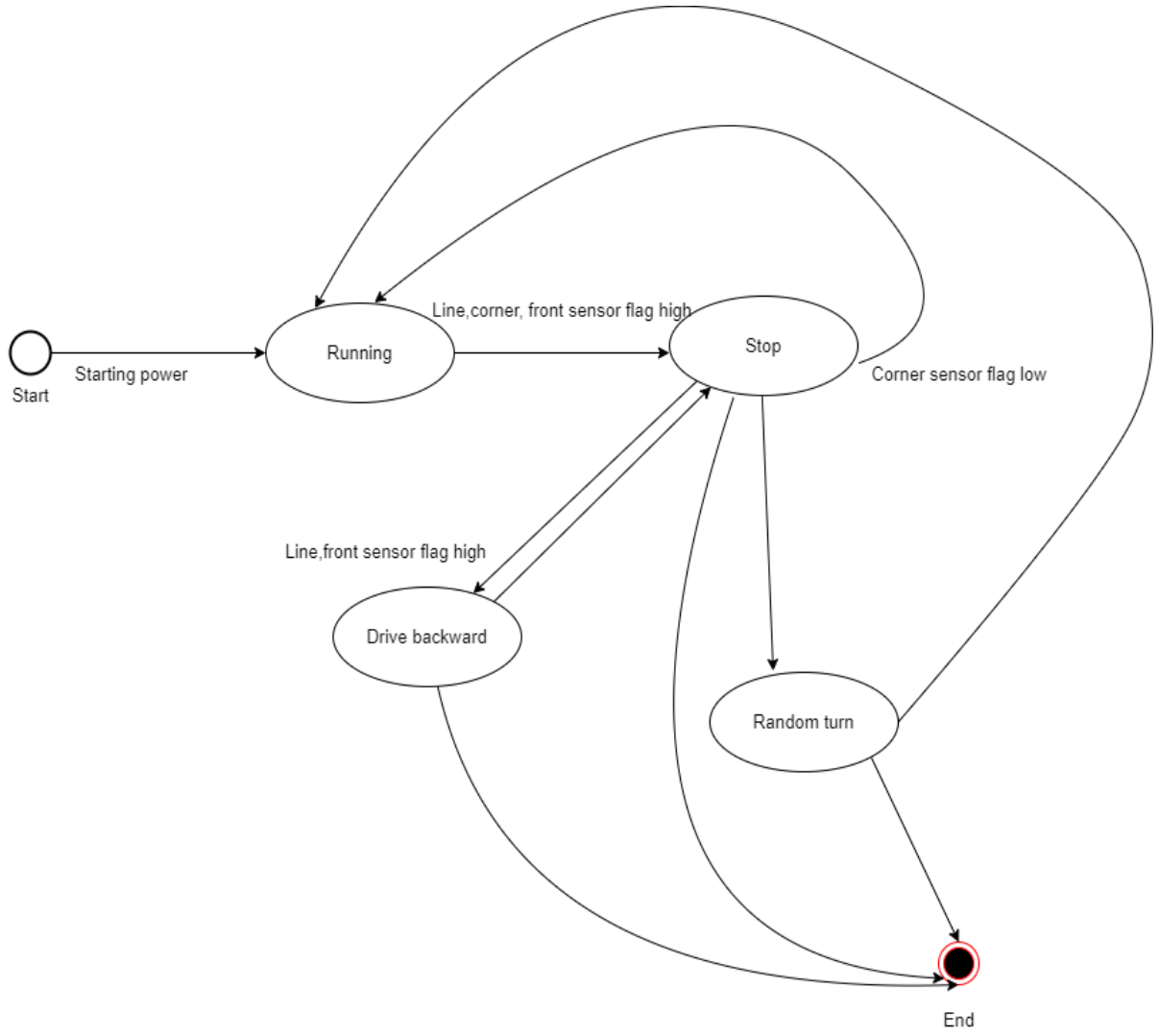
1. The overall Architecture of your system.



System Architecture [Group 10]

The whole system is divided into three layers. Sensor layer, control layer and machine layer. The sensor layer contains three sensors that send signals to the control layer when obstacles and black lines are found. The main body of the control layer is Arduino due. The Arduino due contains three logical codes, namely, machine speed control, machine direction control and sensor signal response. The sensor signal response part changes the behaviour of the robot when receiving the sensor signal. The machine layer includes a plow, wheel encode and motors. These three components physically drive the robot.

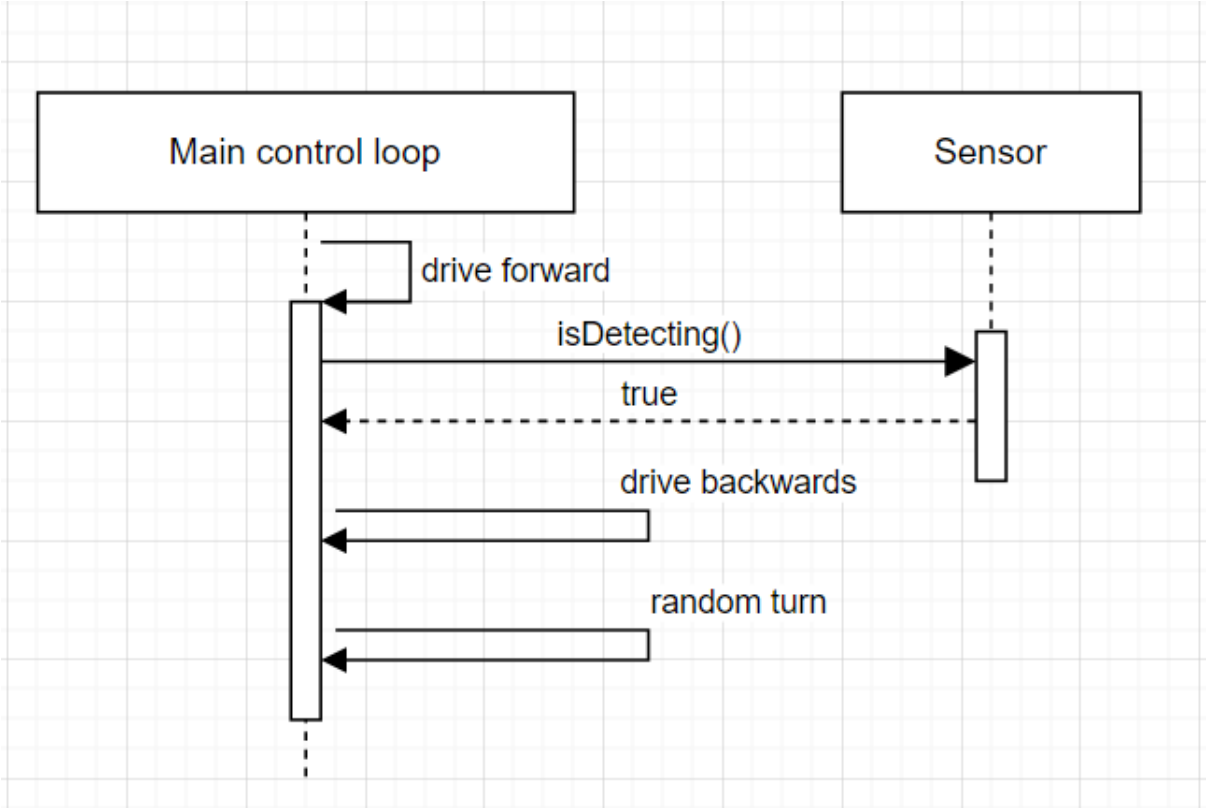
2. State chart of the overall system.



State Chart[Group 10]

The robot starts in a stop state. First, start the power switch, the robot will be in a running state. The robot will keep going straight in the running state. The robot will be changed to a stop state when any sensor sends a signal to change the flag. If the corner sensor flag changes to low, the state will become a running state. If the Line sensor or Ultrasonic sensor sends signal and changes flags, the state of the robot will change to a drive backward state. After [] second, the state will change to stop state. After [] second, the state will change to a random state. After [] second, the state will change to a running state. In any state, the next state will be the end state if the power switch is closed.

3. A sequence diagram

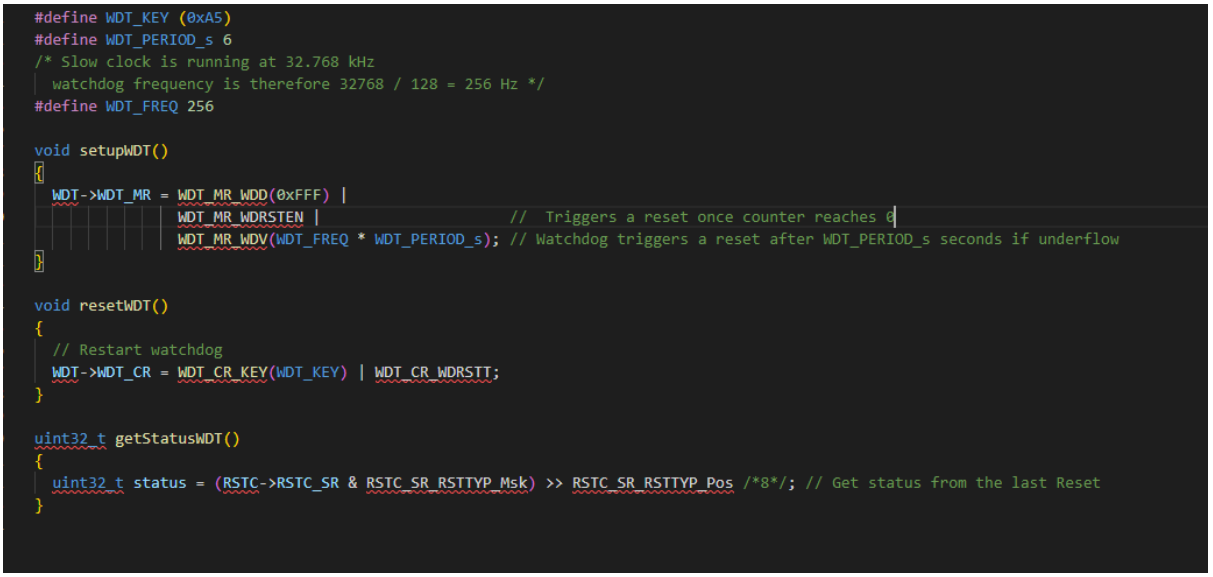


Sequence Diagram [Group 10]

The system sequence diagram is much simpler in the final iteration. Whenever the main loop restarts it checks each sensor. If any sensor is detecting an object, the robot will back up for a specified time, then execute a random turn either left or right depending on which sensor detected (turn right if the left sensor detects it and vice versa). If no sensor is detecting then the robot moves forward. Additionally, for reading stability some sensors will debounce after the first detection. The debounce workflow is not pictured for conciseness.

4. Demonstrate the use of a watchdog timer.

A watchdog timer (WDT) is a hardware timer that produces a system reset if the main application fails to service it on a regular basis. It is frequently used to automatically reset an embedded device that has been stuck due to a software or hardware failure. The watchdog timer is implemented in the main program in our GitHub repo (https://github.com/SYSC4805-Fall2022/sysc4805_term_project-zomp_l1g10/blob/main/AS_main/DUE_WDT.ino)



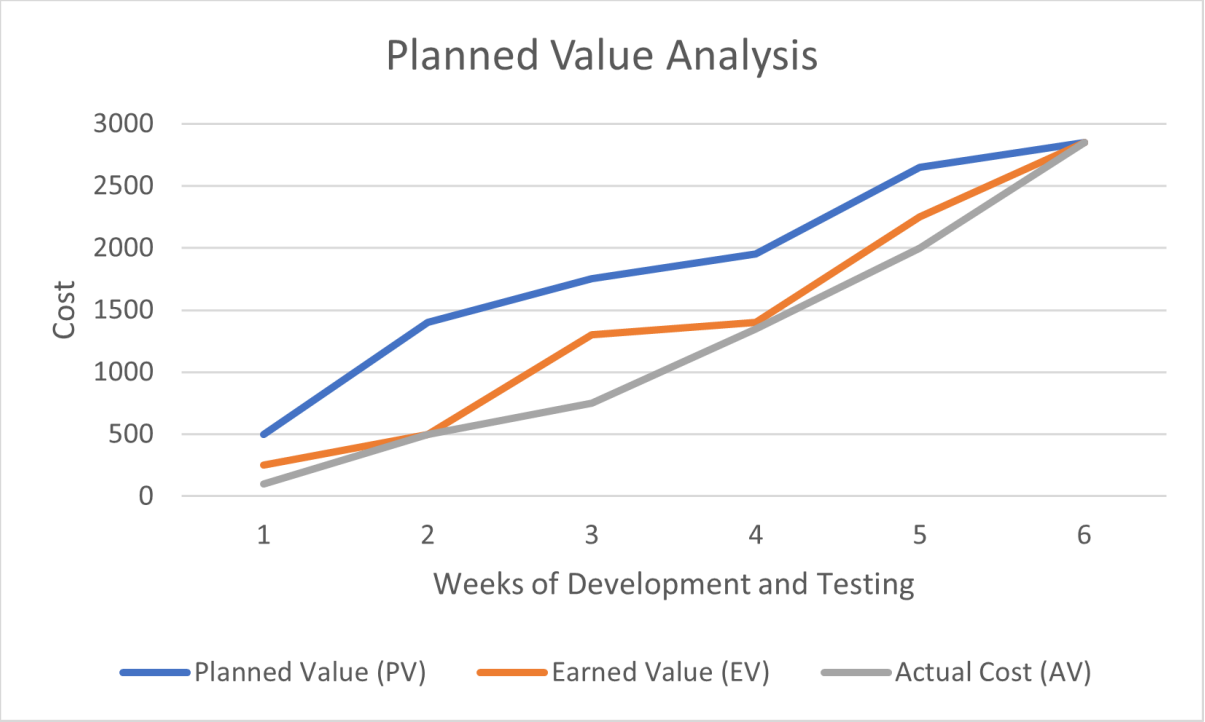
```
#define WDT_KEY (0xA5)
#define WDT_PERIOD_s 6
/* Slow clock is running at 32.768 kHz
  watchdog frequency is therefore 32768 / 128 = 256 Hz */
#define WDT_FREQ 256

void setupWDT()
{
  WDT->WDT_MR = WDT_MR_WDD(0xFFF) |
                WDT_MR_WDRSTEN |                    //  Triggers a reset once counter reaches 0
                WDT_MR_WDV(WDT_FREQ * WDT_PERIOD_s); // Watchdog triggers a reset after WDT_PERIOD_s seconds if underflow
}

void resetWDT()
{
  // Restart watchdog
  WDT->WDT_CR = WDT_CR_KEY(WDT_KEY) | WDT_CR_WDRSTT;
}

uint32_t getStatusWDT()
{
  uint32_t status = (RSTC->RSTC_SR & RSTC_SR_RSTTYP_Msk) >> RSTC_SR_RSTTYP_Pos /*8*/; // Get status from the last Reset
}
```

WDT snippet in the main program [Group 10]

The snippet above shows the watchdog timer (WDT) module implemented for this project. The DUE watchdog timer is a down counter that triggers an interrupt or a system reset once the timer reaches 0. The max counter size is 12 bits and the frequency is 256 Hz. As a result, the WDT's max timer period is 16 seconds. For the autonomous snowplow, the group wanted to leverage the watchdog timer as a protective measure against unwanted blocking loops. As such the watchdog timer should be configured to be slightly longer than the maximum execution time of a single loop. Interrupts are not present in our system, therefore ISR execution time does not need to be considered in this calculation. The max execution time for the system's main loop is ~4 seconds. The group configured the WDT period to be 6 seconds but this value could be adjusted to anything > 4.5 seconds safely. Finally, to reset the WDT for normal operation the 'resetWDT' function can be used (used in main loop here: https://github.com/SYSC4805-Fall2022/sysc4805_term_project-zomp_l1g10/blob/main/AS_main/AS_main.ino#L154).

As requested in the progress report feedback the group has added a more detailed watchdog section above.

5. An updated Planned Value Analysis figure.

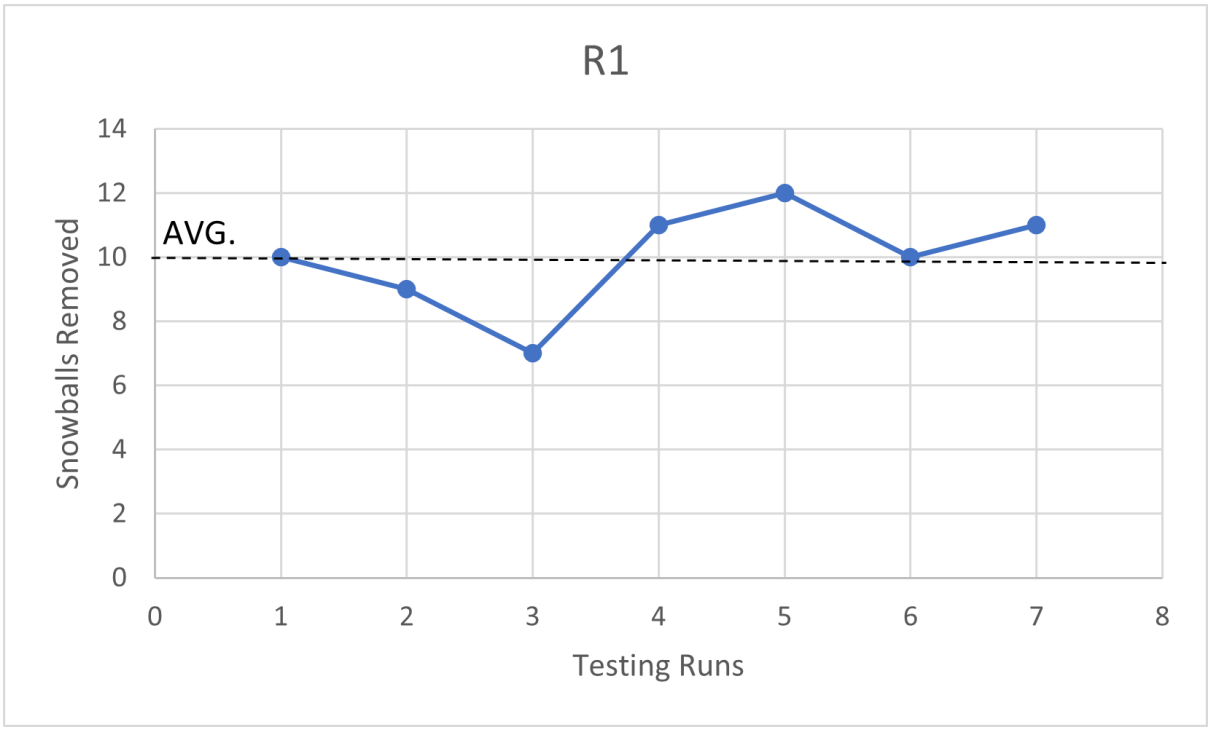


Updated planned Value Analysis[Group 10]

As the planned value analysis diagram illustrates our progress through the project. From the graph, we can see that we were behind schedule (PV>EV) however, we have exceeded expectations on the earned value metric compared to actual costs.

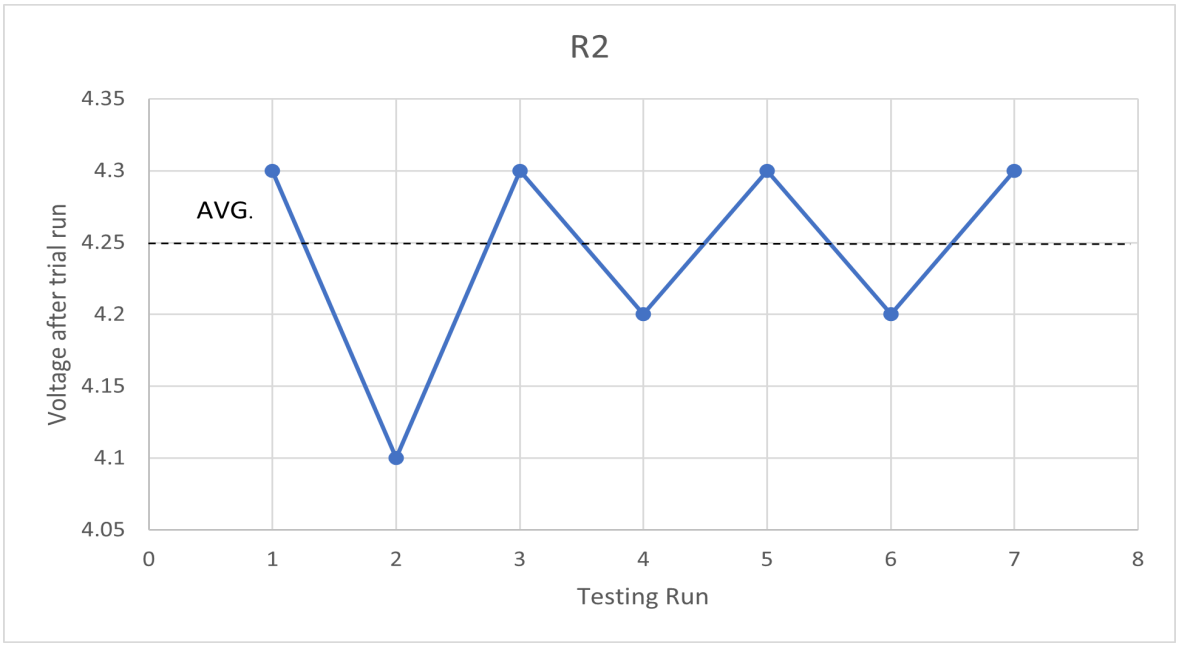
In the last 2 weeks of development and testing, the team garnered momentum and delivered a successful project on schedule and within the budget.

2. Control Charts



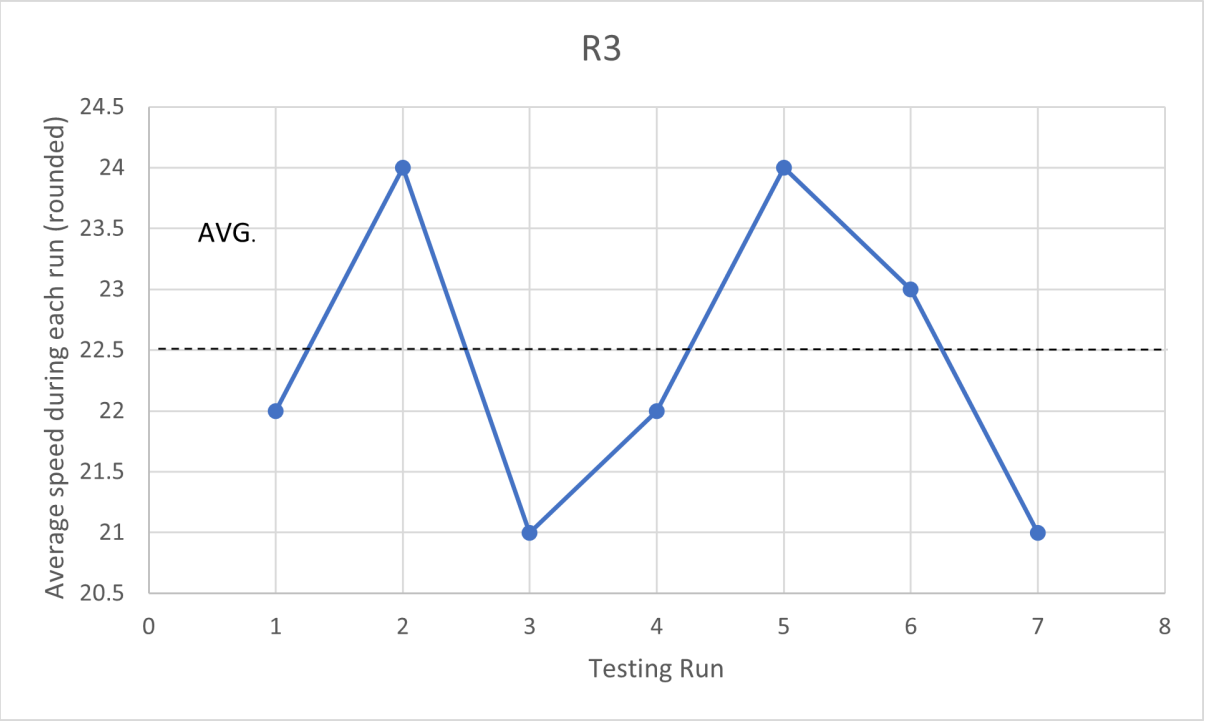
[R1] *The robot shall remove all spheres with a diameter of 42.6mm from a 6 m^2 area outlined with a black perimeter within 5 minutes* **[Group 10]**

The testing was done with 15 balls. There were 7 trial runs



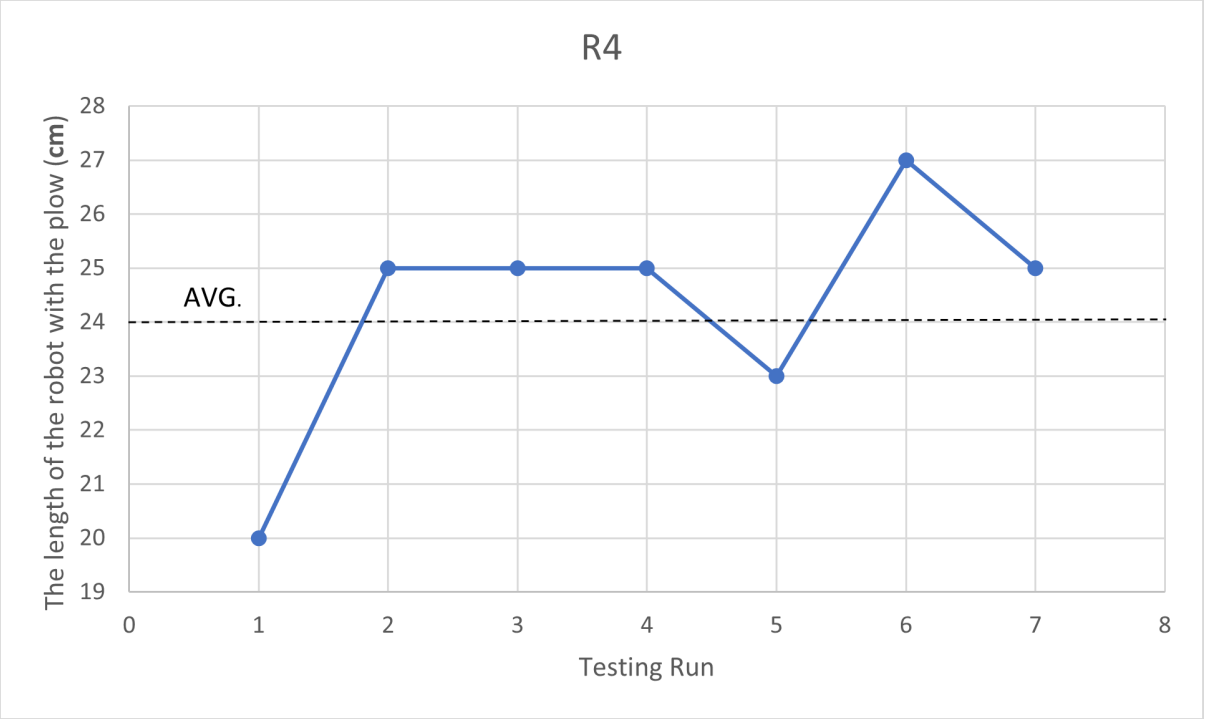
[R2] *The robot's power supply shall supply the required voltage load to its peripherals for proper operation for at least 5 minutes* **[Group 10]**

The power supply was checked after 5 minutes. There were 7 trial runs. The batteries were also recharged to full capacity after each run.



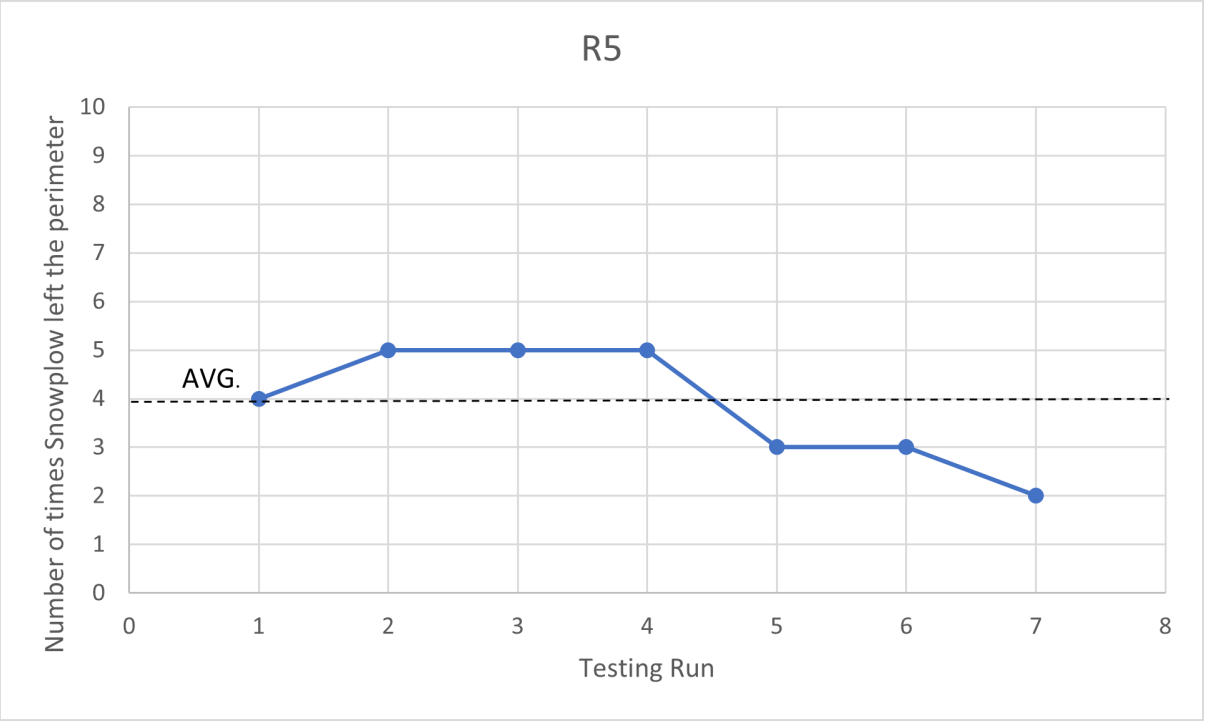
[R3] *The robot shall travel with a maximum linear velocity of 30 cm/s* **[Group 10]**

The above chart shows the average speed achieved during the trial runs. We ran the trial seven times.



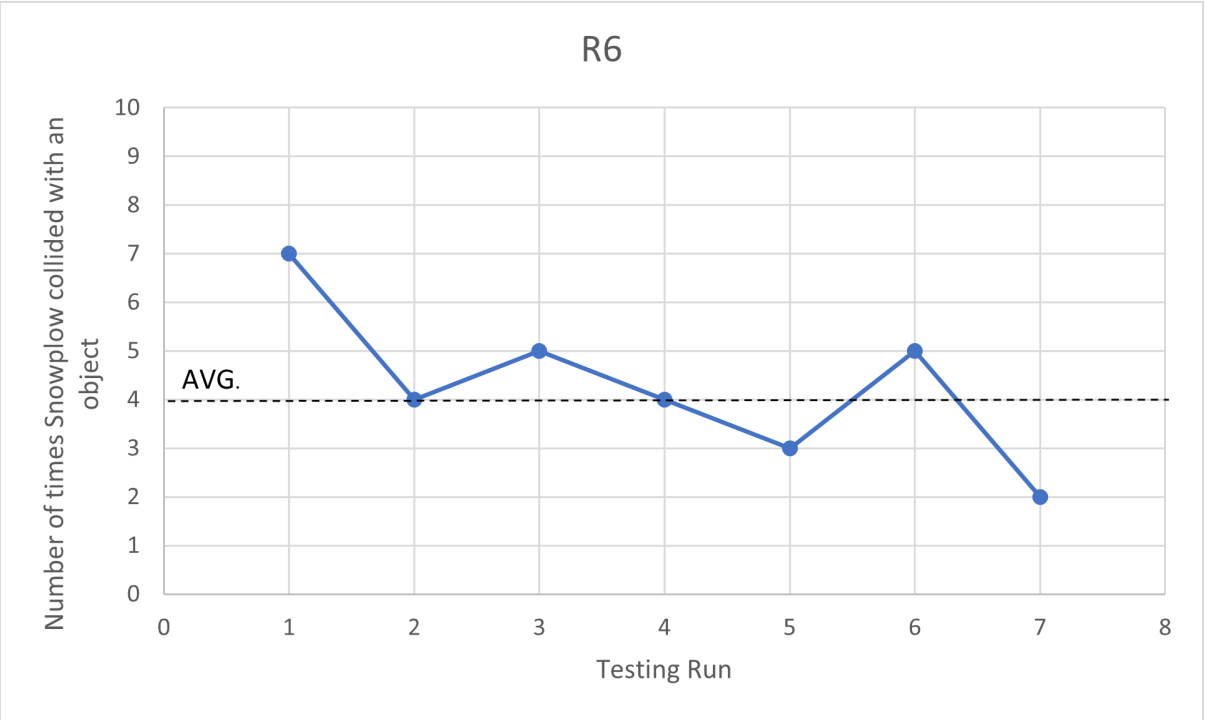
[R4] *The robot frame shall occupy a maximum size of 216 x **252** x 150 mm* **[Group 10]**

The length of the snowplow was iterated 7 times with a final length of 25.0 cm.



[R5] *The robot shall start operation from within the black perimeter.* **[Group 10]**

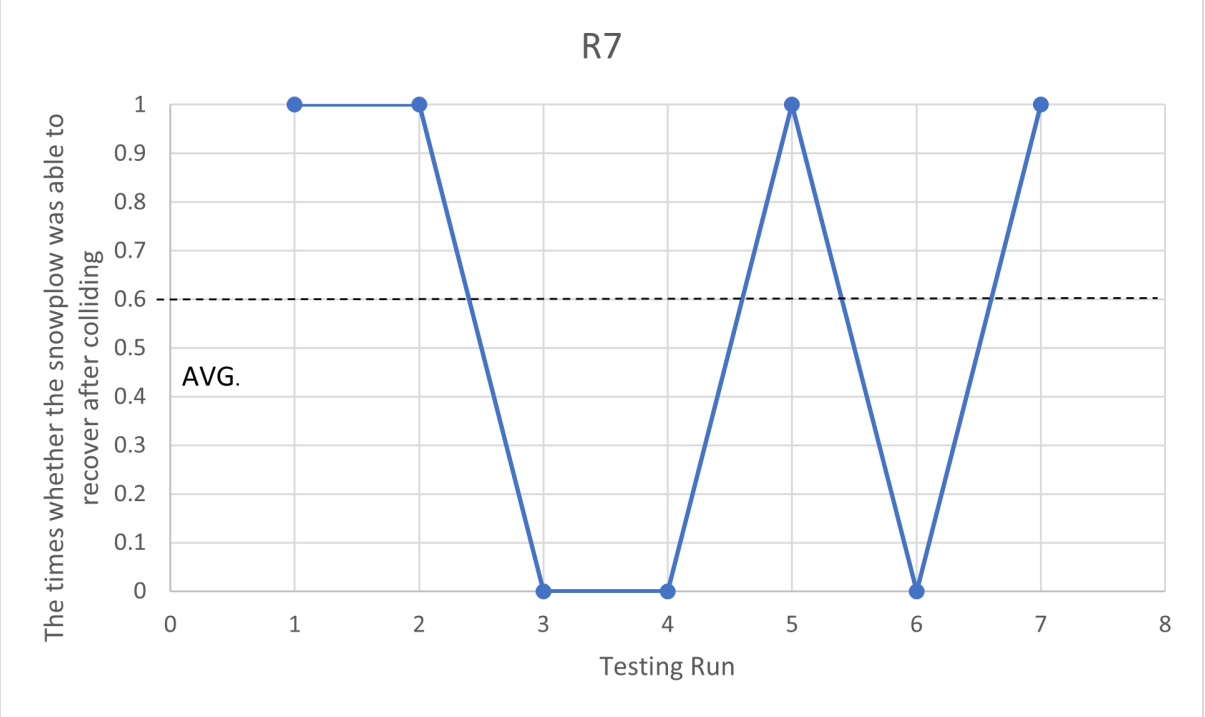
The above chart shows the number of times the snowplow left the perimeter. The team was able to get the number down to an acceptable number.



[R6] *The robot shall not collide with obstacles that are moving or stationary.* **[Group 10]**

The above chart shows the number of times the snowplow collided with an object.

The team was able to get the number down to an acceptable number.



.[R7] *The robot shall be able to continue normal operation after avoiding collisions.* **[Group 10]**

The above chart shows the number of times the snowplow was able to recover from a collision. 0 is for a failure to recover and 1 is for a successful recovery.

Overall, the control charts above show that on average the robot performs adequately per the listed requirements.

3. Results of Testing

The group encountered unexpected development delays toward the second half of the project schedule. The most impactful delay was motor signal noise interfering with the sensor operation. This blocking issue delayed development by three weeks. As a result the group only had one week for testing instead of two.

The test content is divided into four parts, robot motor test, boundary sensing test, obstacle test and the overall test.

We first test the robot motor. The goal of the test is that the robot can move forward normally and maintain a suitable speed. (**T1 & T2**, please refer to the testing plan section for further reading)

In the first test, the robot can move. The movement speed is slow.

By debugging the code, you can adjust the duty and frequency in the code to speed up.

In the second test, the robot moves normally and the speed is suitable.

The second part tests the robot's sensing boundary line and makes a response. (**T1,** please refer to the testing plan section for further reading)

In the first test, the robot did not sense the boundary line. The robot crossed the boundary line and directly drove out of the boundary.
By adjusting the code, set the limit according to the reading of the line following sensor. Only the reading exceeding the limit can be triggered.

In the second test, the robot sensed the boundary line and responded correctly.

In the third part, the test robot senses obstacles and reacts. (**T4 & T5**, please refer to the testing plan section for further reading)

In the first test, the robot directly hit the obstacle.

Through inspection, the main problem is that the voltage will be affected when the robot motor and all sensors operate together. The voltage problem was finally corrected by readjusting the circuit layout and adjusting the positions of the Arduino due board and the Motor Driver board.

In the second test, the robot successfully sensed the obstacle and responded correctly.

The goal of the overall test is that the robot can move in the boundary line and react perfectly when sensing obstacles and boundary lines. (**T1, T2, T3**, **T4 & T5**, please refer to the testing plan section for further reading)

In the first test, the robot could have performed better. When the robot senses the boundary line, it responds slowly and partially incorrectly. Ideally, the robot should back off for a distance before turning. The reality is that the robot does not retreat. The robot responds correctly when sensing obstacles.

Modify the distance the robot retreats by adjusting the code.

The robot performed well in the second test. When the robot senses the boundary line, it retreats about 15cm and then turns. When the robot senses an obstacle, it will stop and then turn randomly.

4. Complete Code

The working code is currently on our GitHub repository (https://github.com/SYSC4805-Fall2022/sysc4805_term_project-zomp_l1g10) where all the team members have contributed to building the software for the Autonomous Snowplow.

5. Contributions

R = Responsible, A = Approver

Table 6: Contributions matrix [Group 10]

Activity	Ryan Fife	Yunzhou Liu	Azizul Hasan
A1 Perimeter detection		A	R
A2 Moving obstacle detection	R		A
A3 Stationary obstacle detection	A	R	
A4 Wheel speed code	R		A
A5 Wheel direction code	R	A	
A6 Turning code		A	R
A7 Staying within boundary	A	R	
A8 Plow hull design	A		R
A9 Plow attachment design		A	R
A10 Plow production	A		R
A11 Rank sensors		R	A
A12 Code stationary obstacle avoidance		R	A
A13 Code moving obstacle avoidance	R	A	
A14 Create circuitry schematic	R	A	
A15 Validation testing	R A	R A	R A
A16 Logic state machine	A	R	

The contribution matrix is similar to the responsibility matrix as the team members contributed to their assigned responsibilities. The team also had the forethought to review each others' contributions to ensure proper practices have been followed.