Autonomous Snowplow

12.09.2022

Team: Old Gold

L3-Group4

GitHub:https://github.com/SYSC4805-Fall2022/sysc4805_term_project-oldgold_l3g4

Wilson Amoussougbo (101119293)

Tyler Mak (101108389)

TABLE OF CONTENTS

1. Project Charter

1.1.Objective

The following document written by the members of Old Gold is a project report for the course SYSC4805 and is based on the conception of an Autonomous Snowplow. The goal of the project is to materialize a robot whose task will be to clear off snow in an enclosed area without hitting any fixed or moving obstacles. The project is composed of diverse components and its completion consists of syncing multiple technologies to ensure a functional assembly. The purpose of the following document is to present a final project report including an updated version of the project proposal as well as UML diagrams and charts depicting the behavior of the overall system.

1.2.Milestone and Final deliverables

This section of the proposal covers the project milestones and final deliverables. The following table lists all the major dates for assignments and team meetings.

Date	Event	Date	Event
September 15th, 2022	Lab 1	November 04th, 2022	-Perimeter Detection -Unit Testing
September 23rd, 2022	Lab 2	November 11th, 2022	-Progress Report -Obstacle Detection -Unit Testing
September 30th, 2022	Lab 3	November 18th, 2022	N/A
October 07th, 2022	Lab 4	November 25th, 2022	-System Testing
October 14th, 2022	Project Proposal	December 02nd, 2022	-Integration Testing -Project Demo -Bug Fixing
October 21st, 2022	-Robot Speed -Unit Testing	December 09th, 2022	-Final Report

Table 1: Project Milestone and Final Deliverable Dates

2. Scope

2.1. Project requirements

Specifying requirements is a crucial part of any project. In this section, all the engineering requirements for this project will be listed.

1. The robot shall not exceed a speed of 30 cm/s while moving
2. The robot shall be a maximum size of 216 by 252 by 150 mm
3. The robot shall be able to detect the perimeter of the testing space
4. The robot shall be able to detect any obstacles before collision
5. Design a plow to attach to the robot for clearing the "snow"
6. The robot shall clear the "snow" in the testing area within 5 minutes

2.2. Project life-cycle

Every project can be broken down into different phases and these phases make the path the project will take to get from the start to the finish. The following image below shows our project's work breakdown structure.

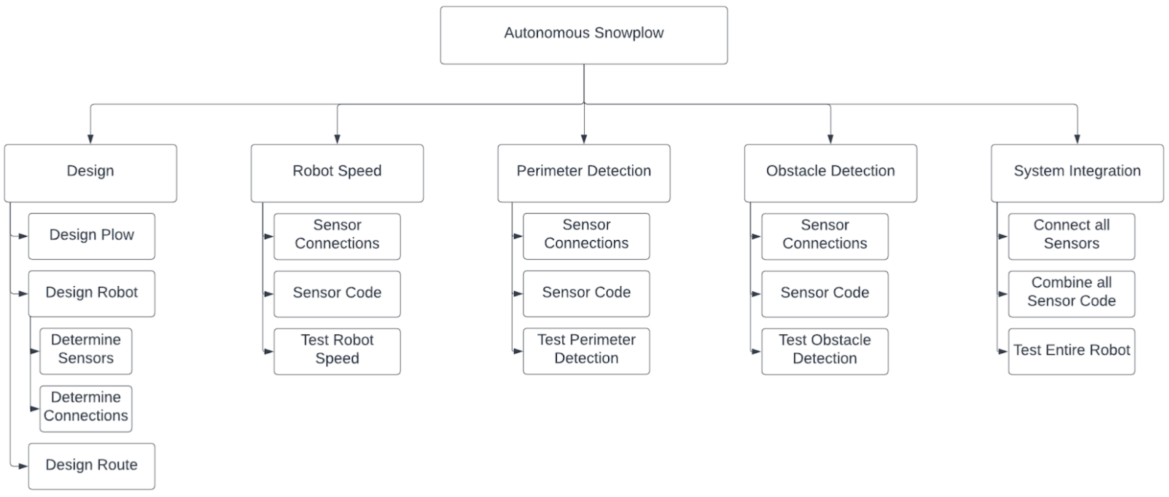

Figure 1: Project Work Breakdown Structure

2.3. Testing

Testing is a fundamental part of our project and a set of testing that covers all the requirements has been put in place. In this section, we will discuss our plan for the unit testing to verify the successful completion of each activity independently from the rest of the project and the integration testing to validate the functionality of the system as a whole.

- **UNIT TESTING**

1. Design
    - ❖ PASS : Width, Length, Height of the robot chassis does not exceed {216, 252, 150} mm
    - ❖ FAIL : Plow Width and Length exceed 40 mm and 20 mm or width, length, or height of overall robot chassis exceeds mentioned dimensions
2. Robot Speed
    - ❖ PASS : Robot moves at specified speed programmed
    - ❖ FAIL : Maximum speed of the robot exceeds 30 cm/s
3. Perimeter Detection
    - ❖ PASS : When sensor detects black tape/perimeter of arena, the robot changes direction or properly reacts
    - ❖ FAIL : Robot assembly crosses black tape/perimeter of arena
4. Obstacle detection
    - ❖ PASS : When sensor detects obstacle, the robot changes direction or reacts properly
    - ❖ FAIL : Robot assembly hits obstacle and fails to detect obsstacle

- **INTEGRATION TESTING**

PASS	***FAIL***
1. Robot moves in the testing area without hitting any obstacles 2. Robot moves in the testing area without crossing the black path line	1. Plow attached to the robot folds or break when clearing off snow 2. Plow diameter exceeds 42.66 mm

Table 2: Integration Testing Plan Pass/Fail Criteria

3. Design

3.1. Overall Architecture

The system architecture shows the overall organization of the project with all hardware and software components communicating together. In the following diagram, the major components are the robot assembly, the arduino processor, the motor driver and the collection of sensors. All the modules of the system are mainly connected using their respective GPIO pins except for the Time of Flight sensor which uses I2C communication protocol.

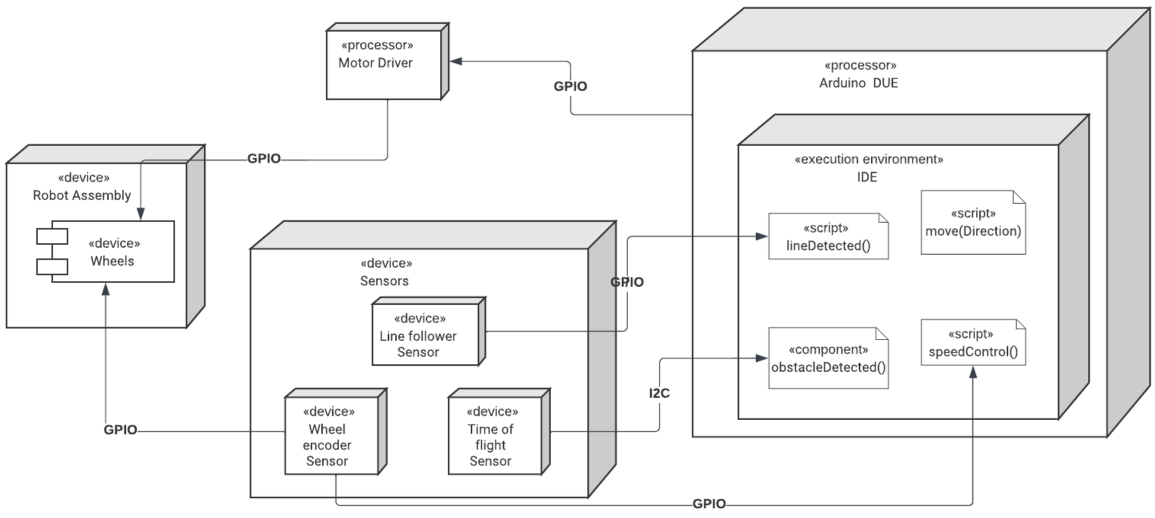

Figure 2: System Architecture

3.2. System Statechart

The statechart or state machine diagram shows the different states of an entity and how it transitions from state to state. It demonstrates how the entity will act depending on the state it is in. The following image below shows our project's statechart diagram for the completed robot.

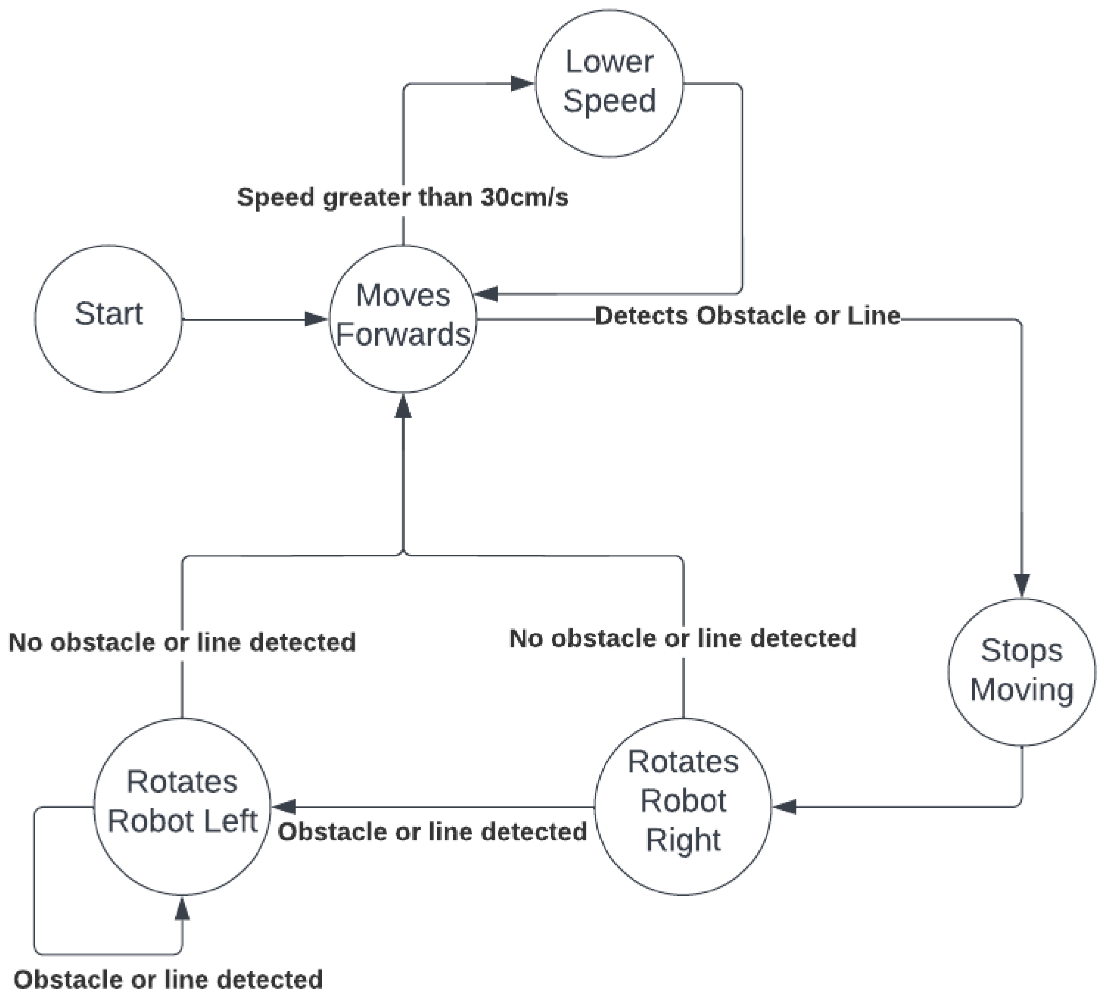
Lower Speed
Speed greater than 30cm/s
Start
Moves Forwards
Detects Obstacle or Line
No obstacle or line detected
No obstacle or line detected
Stops Moving
Rotates Robot Left
Obstacle or line detected
Rotates Robot Right
Obstacle or line detected

Figure 3: Project Statechart

3.3. System Sequence

The following figure shows the sequence diagram highlighting the major events in the project. In fact, when the user starts the program, the function ***move(Forward)*** starts running which allows the assembly to keep continuously moving forward. Both functions ***lineDetected()*** and ***obstacleDetected()*** respectively check for the black path line and an obstacle on the robot trajectory. When either of those are detected, the function ***speedControl()*** is used to decrease robot speed and ***move(Direction)*** allows the robot assembly to move right or left.

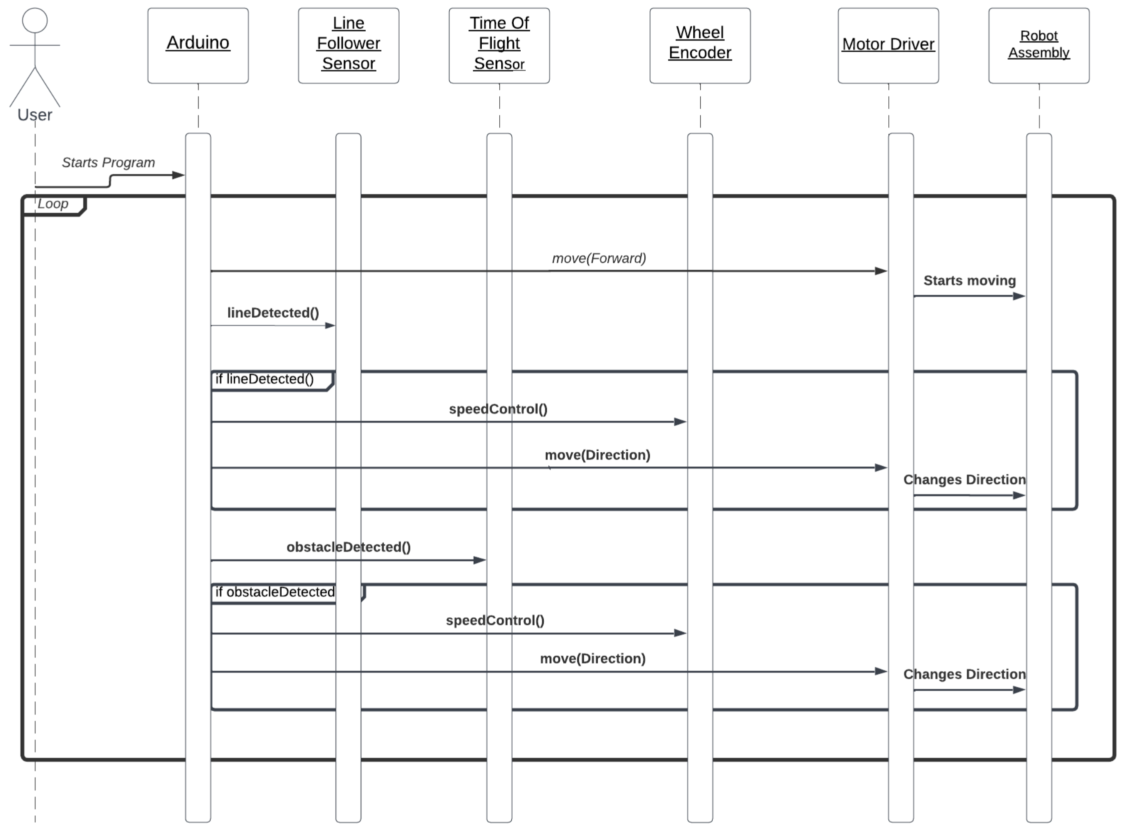

Figure 4: Sequence Diagram

3.4. Watchdog Timer

The watchdog timer is a hardware or software timer that is for detecting and recovering from problems in the code. In order to recover from getting stuck in an infinite loop, the watchdog timer has a timeout value which if reached resets the main program. This ensures that the system can recover from problems and can reset from a stuck state. For our project we will implement a watchdog timer through software using the Arduino Due's included watchdog timer library. The following code will be used to enable the watchdog timer and set the timeout to be 1 second.

```
void watchdogSetup(void){
}
void setup() {
  watchdogEnable(1000); //enable watchdog timer with timeout of 1s
}
void loop() {
  watchdogReset();
}
```

Figure 5: Watchdog Timer Code Implementation

4. Schedule

4.1. Schedule network diagram

A schedule network diagram is used to determine the relationships between activities. The following image below describes the sequential and logical relationships between our project activities.

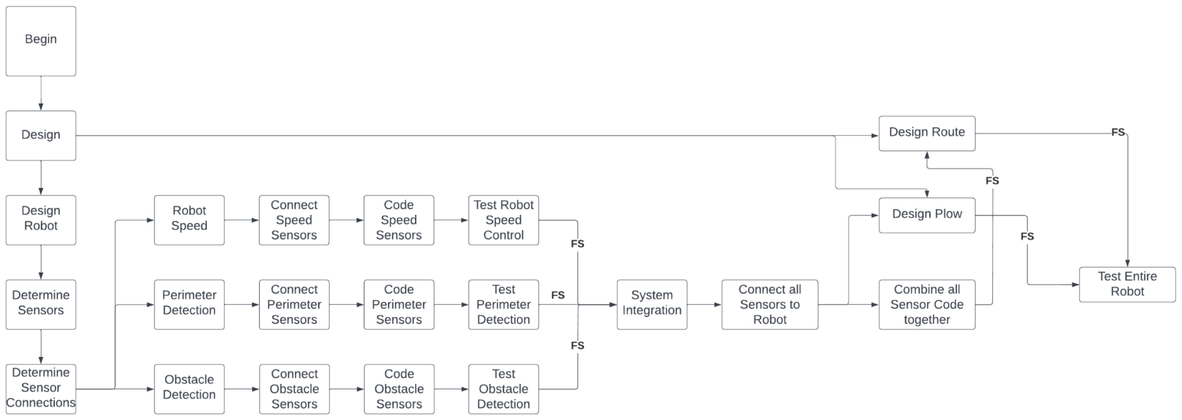

Figure 6: Project Schedule Network Diagram

5. Cost

5.1. Updated Planned Value

The following figure represents a updated cost baseline diagram of the overall project :

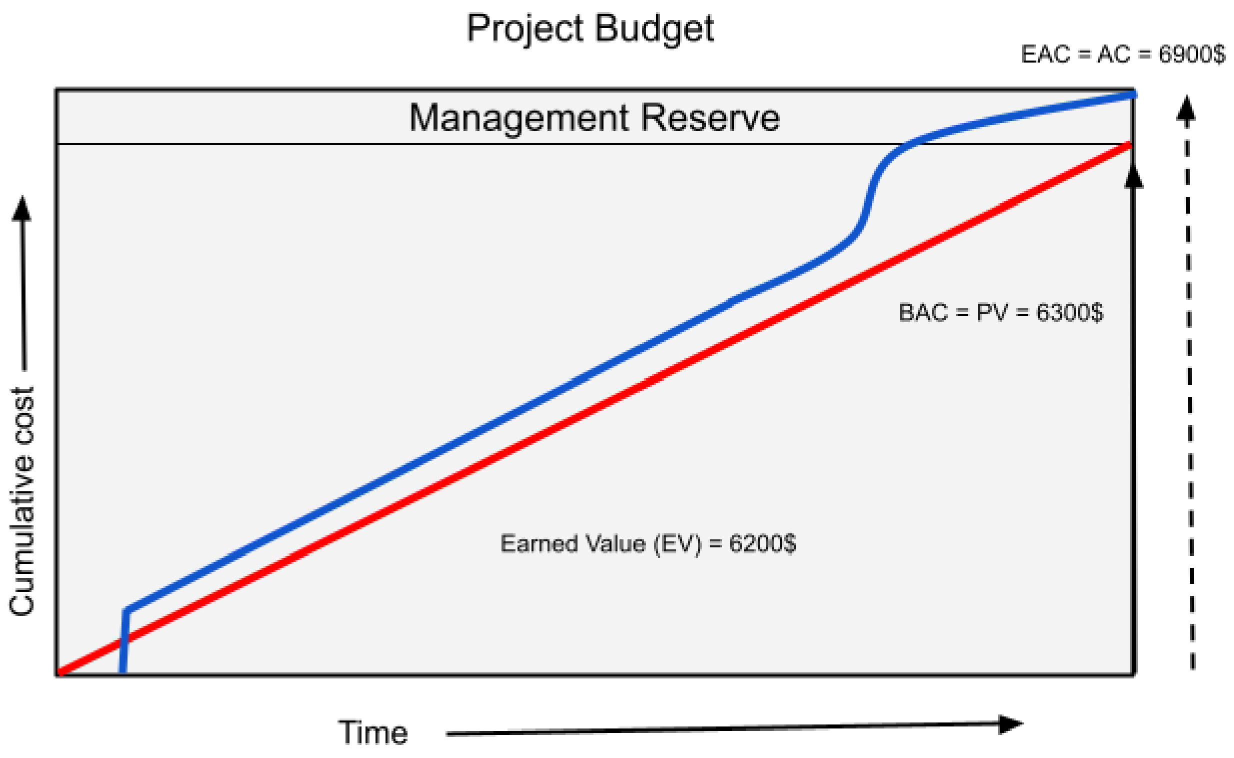

Figure 7: Current Baseline Cost Diagram

The BAC is equal to the PV because the end of the project is reached and all activities are supposed to be completed. To calculate the BAC, we assume we will put in 62 hours per person total and multiply it by our salary which is 50$/hr as well as adding the robot kit which is valued at 100$. The EAC is equal to the AC which was calculated by adding the BAC to the overtime done on the project which is estimated around 6 hours per developer. The EV was calculated using the 62 hours on the hourly pay.

BAC = PV = 124 hrs x 50$/hr + 100= 6300 $

EAC = AC = 100 + 2 x (68 hrs x 50$/hr) = 6900 $

EV = 2 x (62 hrs x 50$/hr) = 2600 $

From the diagram we can see that we are over budget and behind schedule because of the BAC<EAC and EV<PV. As our group did not meet the deadline, we would like to ask for an extension and more funding to be able to meet the requirements.

6. Human Resources

This section of the proposal will cover the project human resources and the person responsible and an approver for each activity will be assigned.

Activities	Responsable	Approver
Connect Speed Sensors	Wilson	Tyler
Code Speed Sensors	Tyler	Wilson
Test Robot Speed control	Tyler	Wilson
Connect Perimeter Sensors	Wilson	Tyler
Code Perimeter Sensors	Wilson	Tyler
Test Perimeter Detection	Tyler	Wilson
Connect Obstacle Sensors	Tyler	Wilson
Code Obstacle Sensors	Wilson	Tyler
Test Obstacle Detection	Tyler	Wilson
Combine all Sensors to Robot	Wilson	Tyler
Combine all Sensor Code together	Wilson	Tyler
Design Plow	Tyler	Wilson
Design Route	Wilson	Tyler
Test entire Robot	Wilson	Tyler

Table 3: Responsibility Assignment Matrix

7. Control Charts

The following figures show the different control charts ensuring requirements were being met once sample test runs were done.

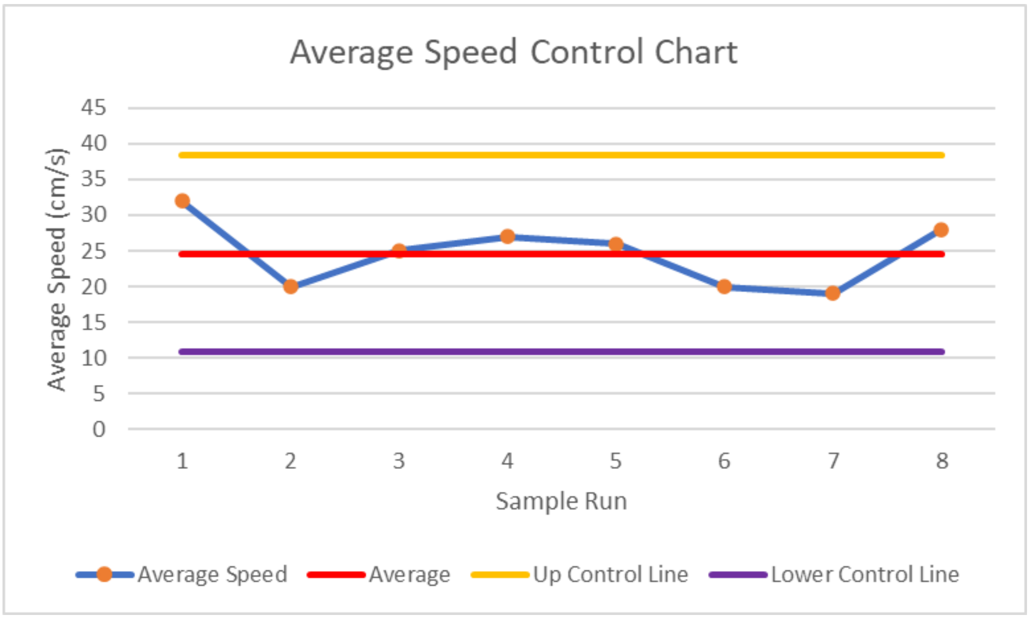

Figure 8: Average Speed Control Chart

Using the wheel encoder, we were able to measure the exact speed our robot was moving at and could calculate an average speed of the entire run. Our first sample run, we received a value that was over 30 cm/s. This is above the required speed and allowed us to immediately change certain values in our code to ensure that for the remaining sample runs we would have a speed below 30 cm/s.

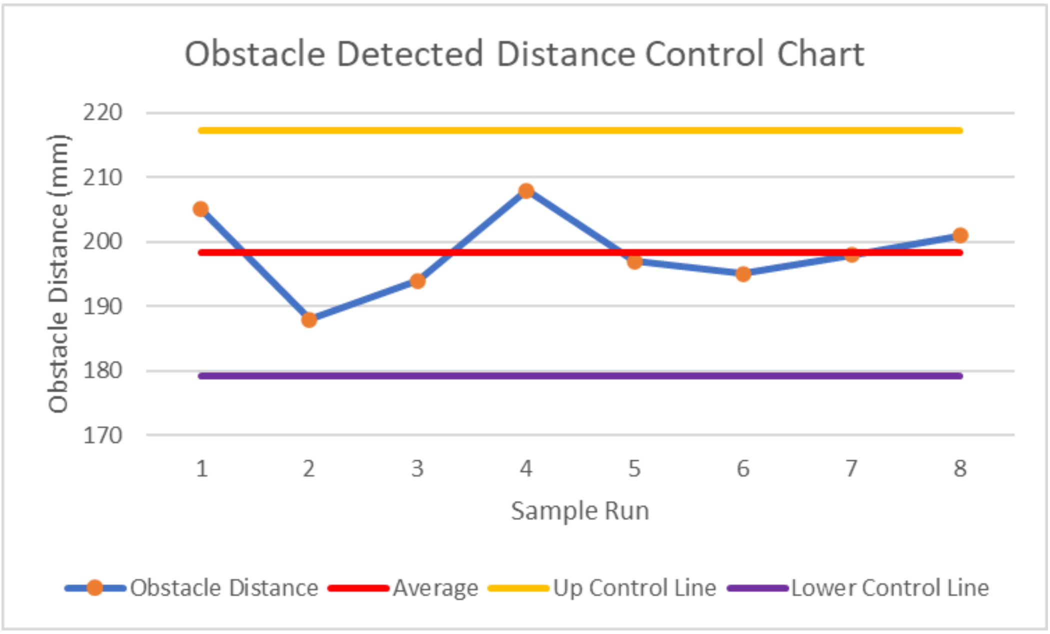

Figure 9: Obstacle detected distance Control Chart

In order to ensure our robot wouldn't run into obstacles we had to set a certain distance at which the time of flight sensor would alert the robot of detecting an obstacle. This value couldn't be too high otherwise we could have false positives, and not too low otherwise the robot may turn into the obstacle. We set the distance of detection to be 200mm or 20cm. This control chart shows that over the sample runs, the time of flight sensor was roughly able to detect the obstacle at the distance of detection we set with an error of about ±10 mm which is acceptable for our case.

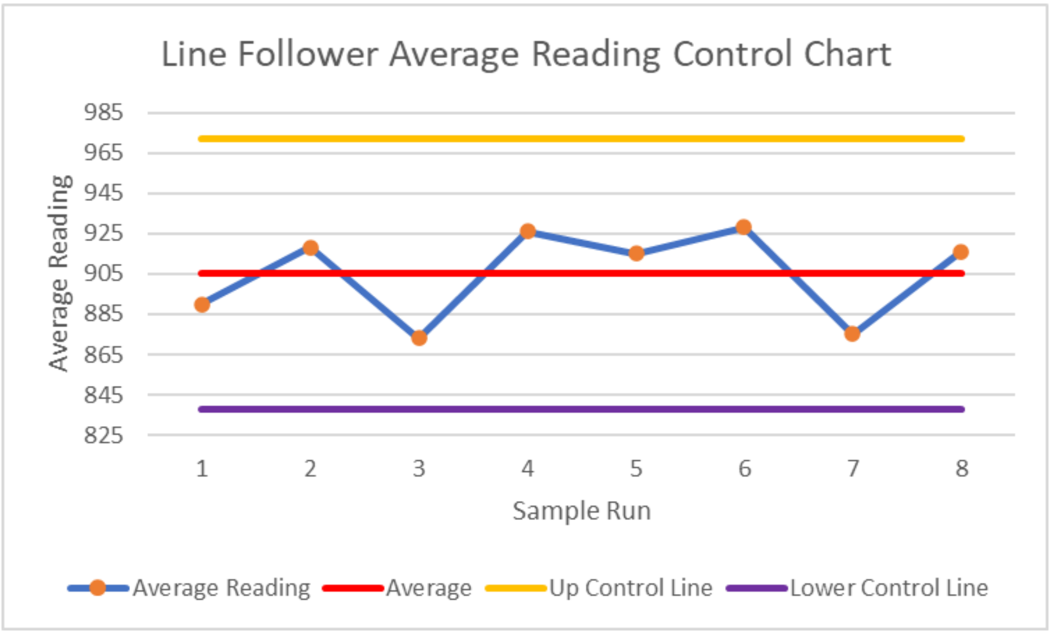

Figure 10: Line followed average reading control chart

In order to detect the perimeter which was black tape, the line follower sensor was used. Initially, using it to read digital values was very sporadic and the line was rarely detected properly if the sensor was not setup perfectly. Instead we used the line follower sensor to read analog values, this gave much more accurate readings and allowed us to determine what values were white and what were black. Using the three sensors on a single line follower module, the control chart above shows what the average analog reading value was when black was detected. Using this we could say an analog reading of 900 would mean black (the perimeter) was detected.

8. Testing Results

For this project, the final testing completed were system testing and customer testing.

★ SYSTEM TESTING

Before the customer testing, the system was tested in multiple ways to ensure full functionality of the robot assembly. Both the line follower sensor and the time of flight sensor were tested simultaneously by setting up boxes as obstacles in an enclosed area and using a white paper with black stripes representing the closed path. We tested the system in two ways by having the robot idle and manually moving the robot and by having the robot in displacement.

While manually moving the robot in the testing area, we were able to preview the printed output of both sensors and validate that the robot can detect obstacles and the closed black path. In the same way, while the robot was moving, it was able to change direction when facing an obstacle and detect when there was a black line and thus not cross it.

★ CUSTOMER TESTING

During the customer testing, the robot was put in a testing area with two boxes representing the obstacles and multiple snowballs all around the testing area. The robot was given 2 trials consisting of 5 minutes each to remove all the snowballs from the testing area without hitting any obstacles or crossing the black path line.

During the first trial, the robot was able to remove 21 snowballs from the area while detecting and avoiding obstacles but would cross the black path line. During the second trial, the robot was able to remove 28 snowballs from the testing area but had a significant decrease in speed and did cross the black path line a couple of times.

It was later determined that the robot was crossing the testing area due to the line follower sensor hanging too close to the floor. We also realized the wires connected to the battery pack were not properly connected to their respective connection points and were not providing enough power for the assembly to move at the expected speeds.

9. Github Repository

The following link is the project Github repository including a working code:
https://github.com/SYSC4805-Fall2022/sysc4805_term_project-oldgold_l3g4

In the repository you can find documented reports of our project in the Documents folder, pictures of our completed robot assembly in the Pictures folder, UML diagrams we have created along the creation of this project in the UML_Diagram folder, and all our code with comments in the projectCode folder.