

**Carleton University**

## **Progress Report**

**Title Page**

### **Computer Systems Design Lab**

**Group Member 1: Kevin Belanger (101121709) L1**

**Group Member 2: Ezra Pierce (100991590) L1**

**Group Member 3: Mohammad Issa (101065045) L1**

**Group Member 4: Haoyu Xu (101088272) L1**

**Course:** SYSC 4805

**Team:** Little Boy Blue

**Date Submitted:** March 4th, 2022

# Table of Contents

Title Page.....	1
Table of Contents.....	2
Summary.....	3
1. Project Charter.....	4
1.1 Project Objective.....	4
1.2 Project Deliverables.....	4
1.3 The Overall Architecture.....	4
1.4 State Chart of the Overall System.....	5
1.5 Sequence Diagram.....	6
2. Scope.....	6
2.1 List of Requirements.....	6
2.2 Work Breakdown Structure.....	6
2.3 Testing.....	8
3. Schedule.....	9
3.1 Schedule Network Diagram.....	9
3.2 Gantt Chart.....	11
4. Human Resources.....	12
4.1 Responsibility Assignment Matrix.....	12
5. Project Budget.....	13
5.1 Budget at Completion (BAC).....	13
5.2 Planned Value (PV).....	13
6. Progress.....	14
6.1 Week 1.....	14
6.1.1 Robot Body Selection.....	14
6.1.2 Robot Sensor Selection.....	15
6.1.3 Plow Controller Selection.....	16
6.1.4 Plow Design.....	16
6.2 Week 2.....	17
6.2.1 Plow and Body Integration.....	17
6.2.2 Sensor Integration.....	17
6.2.3 Plow Controller Attachment.....	18
6.2.4 Matching controller specifications to datasheets from real controller.....	18
6.3 Week 3.....	19
6.3.1 Motor Control Via Python.....	19
6.3.2 Sensor Output to Python.....	19
6.3.3 Plow Control Via Python.....	19
6.3.4 Controller Output to Python.....	19
7. References.....	20

# Summary

The purpose of this project is to propose a robot built using CoppeliaSim that clears off the snow in an enclosed area. This can be done using various tools learned in the previous labs using CoppeliaSim. This proposal plans activities to accomplish within a 7 week period. Work with a team based environment to come up with a plan that is efficient and effective to develop the robot. CoppeliaSim is a robot simulator that is used for fast algorithm development, factory automation simulations, and robotics related education. The proposal will cover these topics: the project objective, deliverables, requirements, and testing. Also, it will consist of diagrams and tables showing the activities to be completed by each team member.

The objective of this project is to design and implement a simulated robot that removes snow from a defined area without hitting any and all obstacles. The deliverables of this project are based on the objective that is needed to be done by the end of each week. The end result is a working robot that satisfies the project objectives accompanied by a report detailing the information related to the project and the work that went into it. The project requirements consist of a list of parameters that the robot should have that would deem the project valid. Testing includes the tests that will take place each week in order to make sure the activities completed work as promised.

The work breakdown structure consists of a detailed timeline in which it divides the workload among the team members. Each member should have an activity to complete each week that follows the guidelines set by the proposal. The schedule network diagram is a diagram that connects all the activities set out to be completed by each team member with respect to their timeslot. Gantt chart: is similar to the schedule network diagram in terms of showing the activities and when they should be completed with respect to time. However, it is more visual and is easy to follow on a weekly basis. Responsibility Assignment Matrix: is a matrix that shows each member responsible for completing an activity as well as approving another member's activity. In conclusion, the proposal will help the team to stay on top of tasks as well as allow for a more agile work environment.

# **1. Project Charter**

## **1.1 Project Objective**

Our goal is to design and implement a simulated robot that removes snow from a defined area without hitting any and all obstacles.

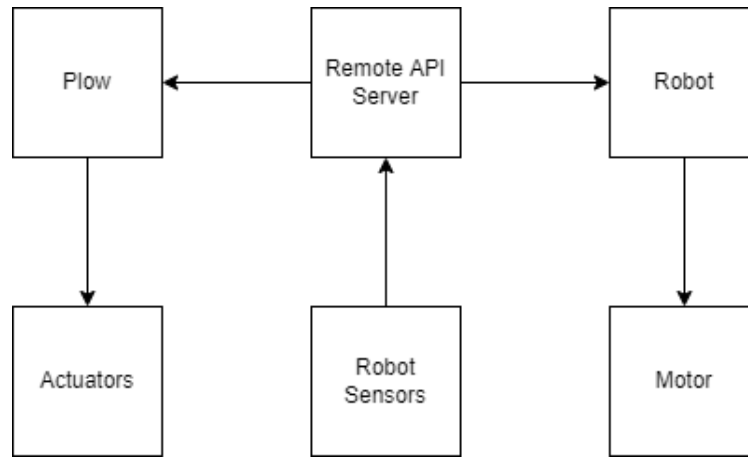
## **1.2 Project Deliverables**

In terms of deliverables for the project, we expect to complete a project proposal that summarizes our goals and outlines the breakdown of milestones that will be achieved during each week and by which members of the group. After a few weeks have passed and progress has been made on the project, we will create a progress report to update the observers of the project on where we will currently be in terms of the previously set milestones and overall project completion. Nearing the end of the term when the project is complete, we will have a finalized version of the snow plowing robot we will have created in CoppeliaSim as an exported model which will be accompanied by the necessary script files that allow the robot to function and meet the outlined requirements of the project. This will be the complete work and it will be consolidated into a final report that will encapsulate the work completed throughout the term and provide the project observers with the results and struggles of the model and any other important information relating to the process of developing the final product.

## **1.3 The Overall Architecture**

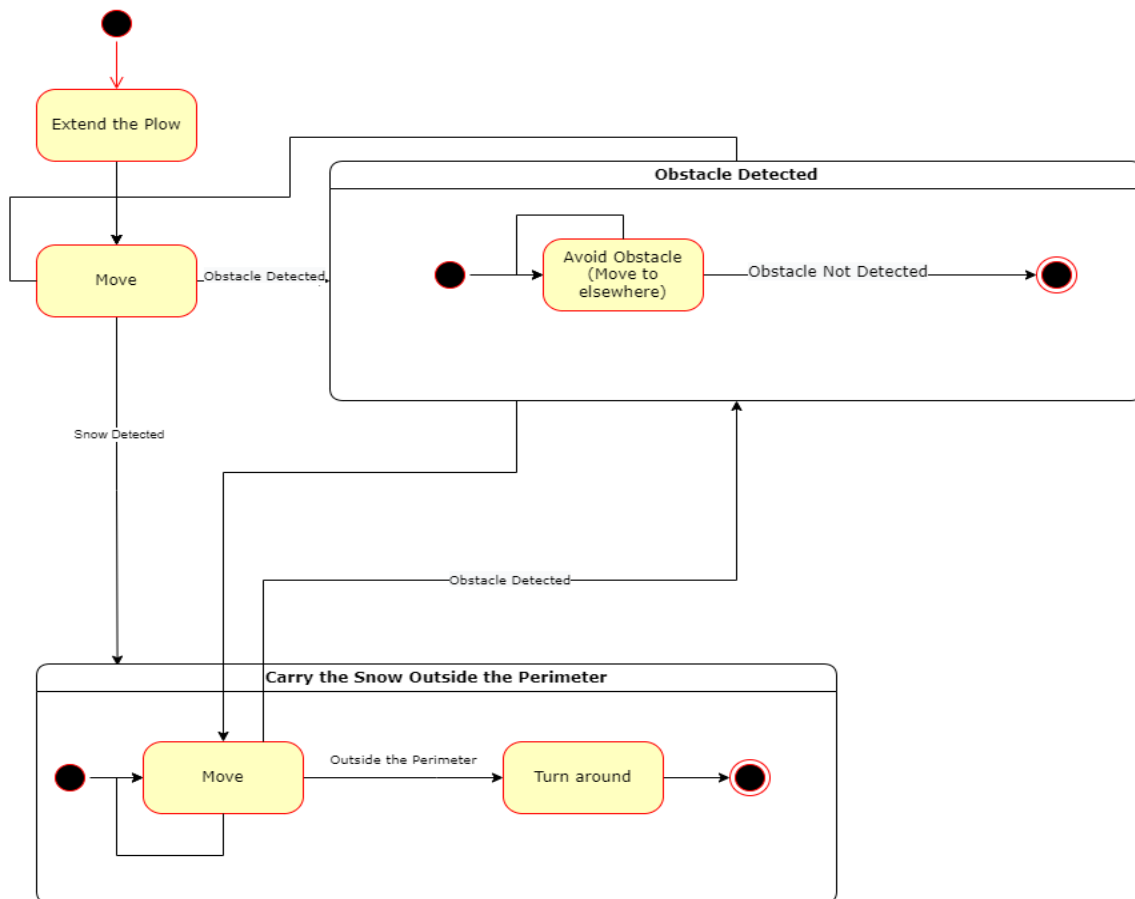
We have designed a robot body that has a few different major sections when it comes to the functionality and efficiency of the robot. These components include the body, wheels, motors, joints, arm, plow, and sensors. First we look at the dynamic pieces: the body consists of a rectangular prism shaped cuboid attached to four wheels and a motor for each wheel. These components combined allow the robot to move. Next is the plow, it has four long and thin plates connected with moving joints to allow it to extend once set in motion. This plow section is then connected to the body by an arm attached to a tall cuboid sitting on the center of the body. This arm also has a joint where it connects to the center piece which allows the plow and arm to move up and down. These moving parts give the robot the ability to reach its maximum stationary size limitations while extending further once moving to take advantage of a larger plow as allowed by the moving size restrictions. Our goal is to create this robot while keeping in mind that we want it to act in an event triggered manner. This will allow the robot to continue working on removing the snow until it encounters an obstacle or the edge of the border (significant events). As for the testing scenarios implemented for the robot, all scenarios have a success and fail criteria. Currently the robot passes the first four tests, where it is able to achieve the success criteria mentioned in section 2.3 regarding testing. First, the attachments are connected to the robot body and perform as intended during simulation. Then, the python connection over RemoteAPI is successful during

simulation. Third, the robot components are created explicitly not to exceed the robot dimension requirements. Finally, the robot is able to move and turn properly through the python APIs provided by CoppeliaSim. This architecture is visualized in the following figure:



*Figure 1: Architectural Diagram*

## 1.4 State Chart of the Overall System



*Figure 2: State Chart of system*

## 1.5 Sequence Diagram

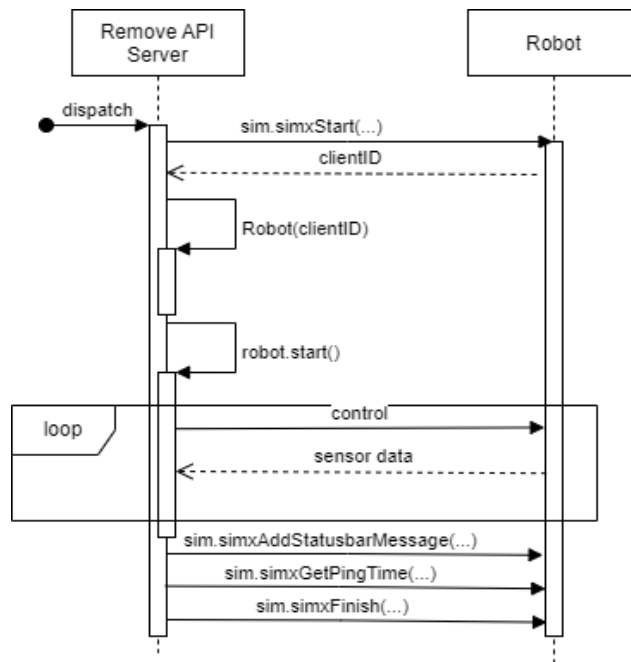


Figure 3: Sequence Diagram for system

## 2. Scope

### 2.1 List of Requirements

Customer Requirements:

- The robot should be able to clear all snow spheres off of a 12m x 12m perimeter track
- The robot should clear the snow in less than 5 minutes
- The robot should not hit any obstacles while it is clearing the snow
- The robot must be smaller than 0.5m x 0.8m x 1.0m when parked
- The robot must be smaller than 1.0m x 0.8m x 1.0m at all times
- The speed of the robot should not exceed 2 m/s
- The simulation should be able to run in CoppeliaSim with additional Python scripts

### 2.2 Work Breakdown Structure

Activity	Description
1	Robot Body Selection: Select and decide on an appropriate robot body to use
2	Robot Sensor Selection: Select and decide on appropriate sensors to use
3	Plow Controller Selection: Select and decide on appropriate actuators to use

4	Plow Design: Design the plow (Shape, size, operation)
5	Plow and Body integration: Integrate the plow on the robot body
6	Sensor and Body integration: Integrate the sensors on the robot body
7	Matching sensor specifications to datasheets from real sensors
8	Matching controller specifications to datasheets from real controller
9	Motor Control via Python: Control the motors using Python
10	Plow Control via Python: Control the plow using Python
11	Sensor Output to Python: Retrieve sensor data using Python
12	Controller Output to Python: Retrieve controller data using Python
13	Robot Plowing Behavior Initial Training: To extend and retract the plow
14	Robot Moving Behavior Initial Training: To move the robot in the perimeter
15	Robot Obstacle Avoidance Initial Training: To avoid the robot colliding the obstacles
16	Robot Perimeter Detection Initial Training: To detect the black boundary of the perimeter
17	Robot Behavior Training in training map 1: Train the robot in Map 1
18	Robot Behavior Training in training map 2: Train the robot in Map 2
19	Robot Behavior Training in training map 3: Train the robot in Map 3
20	Robot Behavior Improvement: To make the movement of the robot better
21	Improved Robot Behavior Training in training map 1: Train the updated robot in Map 1
22	Improved Robot Behavior Training in training map 2: Train updated the robot in Map 1
23	Improved Robot Behavior Training in training map 3: Train the updated robot in Map 1
24	Robot Behavior Adjustment: To adjust the movement of the robot based on the training results
25	Final Report Preparation: To prepare the final report
26	Robot Improvement: To make the robot look better and run better
27	Robot Testing: To test the overall robot behavior
28	Final Project Demonstration Preparation: To prepare the final demonstration

## 2.3 Testing

Test	Description	Success/Fail Criteria
1	Robot and Attachments test	Success: Attachments are connected to the robot body properly Fail: Attachments are not connected to the robot body properly (ex. They fall off when the simulation starts)
2	Robot and Python connection test	Success: Console prints “Connected Through RemoteAPI server” Fail: Connection was not established
3	Robot Dimension test	Success: The body parts are measured to satisfy the project requirements Fail: The body parts exceed the project requirement measurements (ex. During simulation mode the plow extends outside of the specified requirement)
4	Robot Movement test	Success: The robot can move straight, left, right, and back normally without exceeding the maximum speed. Fail: The robot cannot move as intended.
5	Plow shoveling test	Success: The plow is able to carry the snow outside of the perimeter Fail: The plow doesn’t interact with the snowballs properly
6	Obstacle avoidance test	Success: The robot successfully avoids the obstacles ahead Fail: The robot is not able to detect obstacles appropriately and avoid them
7	Map testing	Success: The robot can pass all test maps and clear all snow spheres off. Fail: The robot cannot clear all snow spheres off on all test maps.



### 3. Schedule

#### 3.1 Schedule Network Diagram

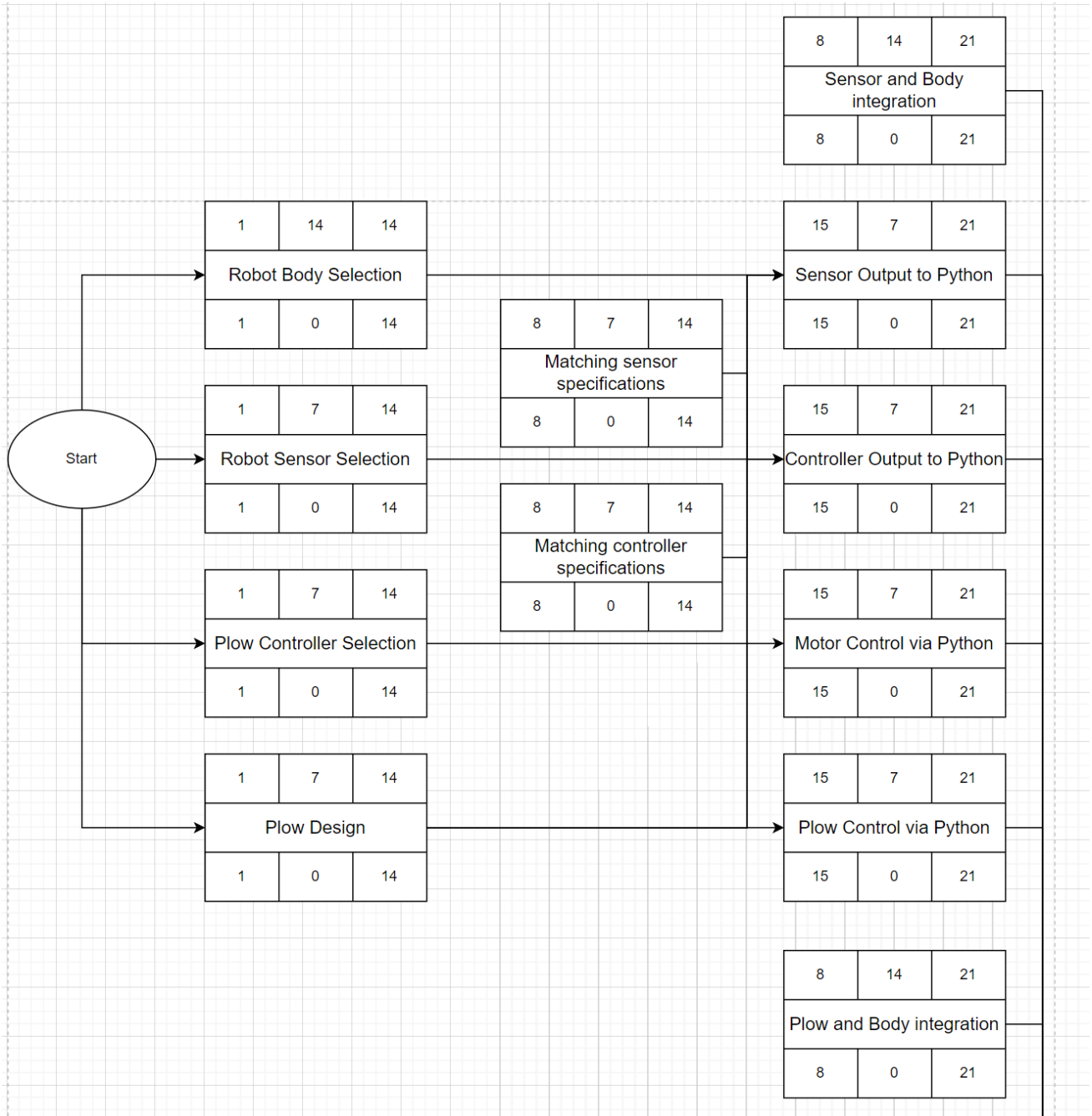


Figure 4: Part 1 of the Schedule Network Diagram

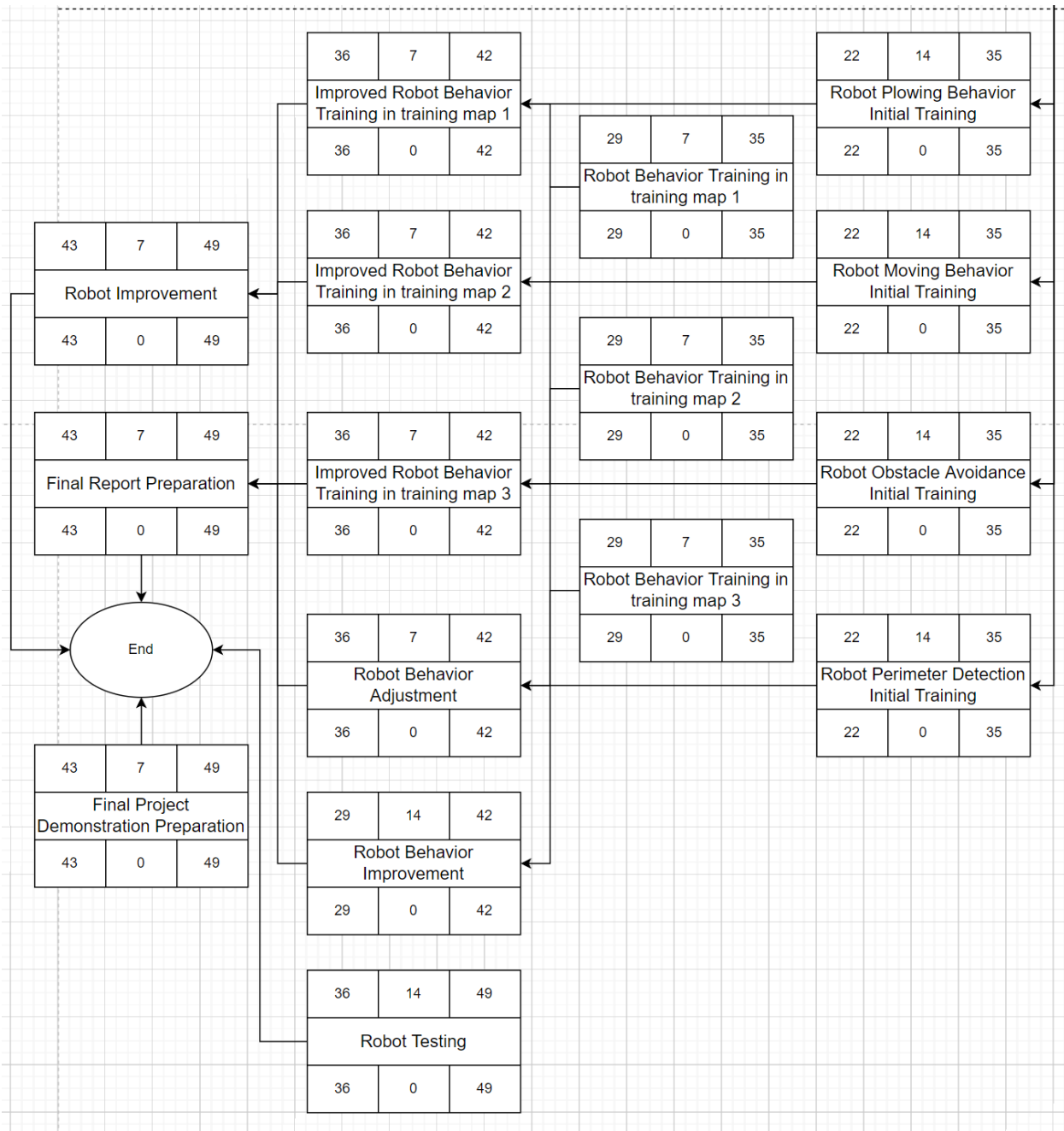


Figure 5: Part 2 of the Schedule Network Diagram

## 3.2 Gantt Chart

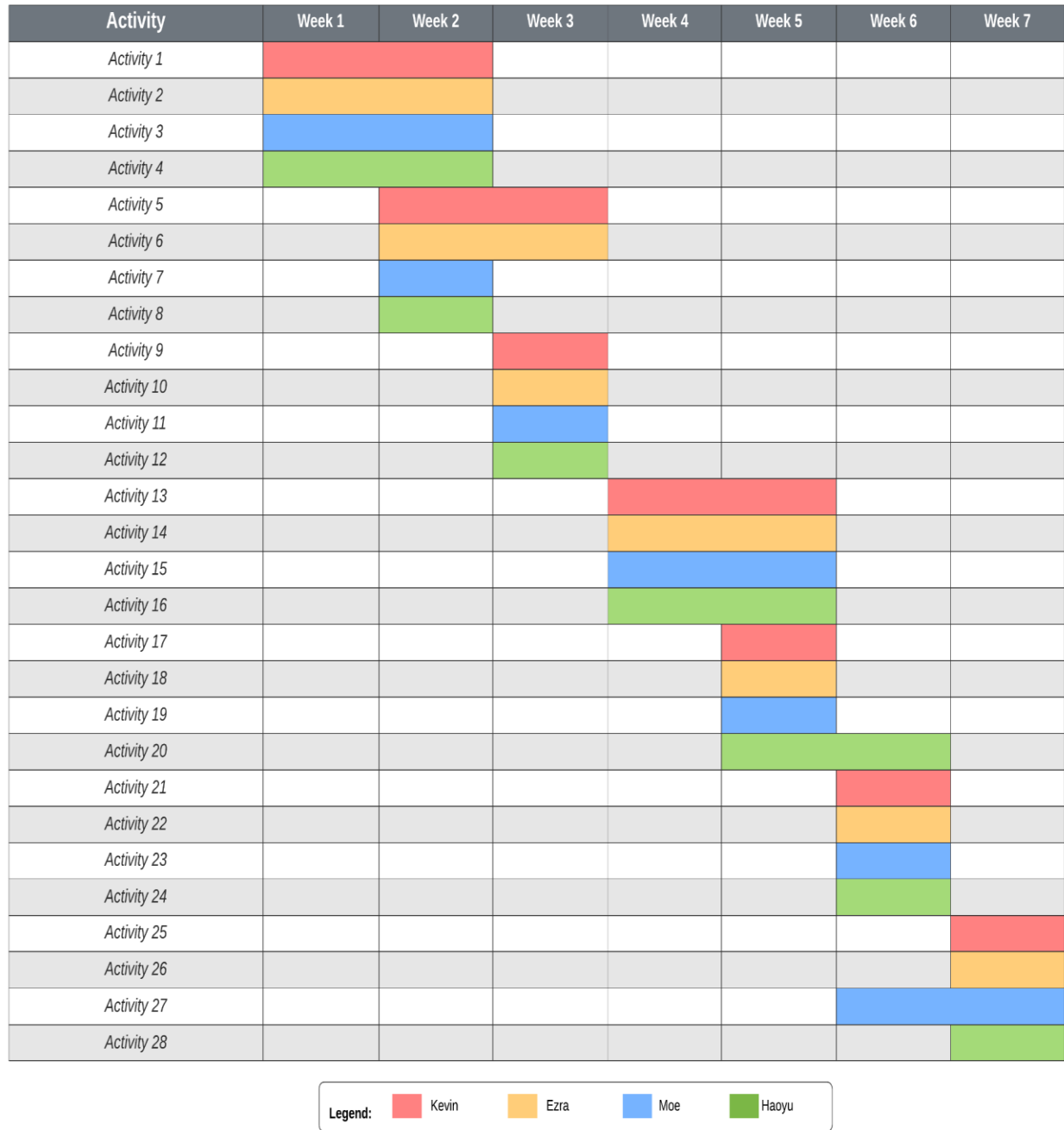


Figure 6: Gantt Chart of the system

## 4. Human Resources

### 4.1 Responsibility Assignment Matrix

\***R** - Responsible (Work on), **Q** - Quality Reviewer

Activity	Kevin	Ezra	Moe	Haoyu
1	<b>R</b>	<b>Q</b>		
2		<b>R</b>	<b>Q</b>	
3			<b>R</b>	<b>Q</b>
4	<b>Q</b>			<b>R</b>
5	<b>R</b>	<b>Q</b>		
6		<b>R</b>	<b>Q</b>	
7			<b>R</b>	<b>Q</b>
8	<b>Q</b>			<b>R</b>
9	<b>R</b>	<b>Q</b>		
10		<b>R</b>	<b>Q</b>	
11			<b>R</b>	<b>Q</b>
12	<b>Q</b>			<b>R</b>
13	<b>R</b>	<b>Q</b>		
14		<b>R</b>	<b>Q</b>	
15			<b>R</b>	<b>Q</b>
16	<b>Q</b>			<b>R</b>
17	<b>R</b>	<b>Q</b>		
18		<b>R</b>	<b>Q</b>	
19			<b>R</b>	<b>Q</b>
20	<b>Q</b>			<b>R</b>
21	<b>R</b>	<b>Q</b>		
22		<b>R</b>	<b>Q</b>	

23			R	Q
24	Q			R
25	R	Q		
26		R	Q	
27			R	Q
28	Q			R

## 5. Project Budget

### 5.1 Budget at Completion (BAC)

To determine the planned value of our project, we use the following equation:

$$\text{BAC} = (\text{Total cost of components}) + (\text{Total cost of labor})$$

$$\text{BAC} = [(\text{Orthogonal Sensor} \times 3) + (\text{Proximity Sensor} \times 3) + (\text{Plow Actuators} \times 5)] + [(\text{Hourly Wage}) \times (\text{hours/week}) \times (\text{number of weeks}) \times (\text{number of team members})]$$

$$\text{BAC} = [(\$3.50 \times 3) + (\$15.87 \times 3) + (\$129.99 \times 5)] + [(\$24.50/\text{h}) \times (4\text{h/week}) \times (7 \text{ weeks}) \times (4 \text{ members})]$$

$$\text{BAC} = (\$708.06) + (\$2744.00)$$

$$\text{BAC} = \$3452.06$$

### 5.2 Planned Value (PV)

To determine the planned value of our project, we use the following equation:

$$\text{Planned Value} = (\text{Planned \% Complete}) \times (\text{BAC})$$

$$\text{Planned Value} = \left( \frac{12}{28} \text{ activities} \right) \times (\$3,452.06)$$

$$\text{Planned Value} = (42.86\%) \times (\$3,452.06)$$

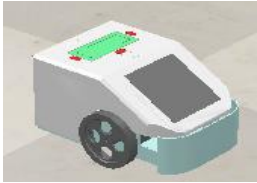

$$\text{Planned Value} = \$1,479.55$$

## 6. Progress

### 6.1 Week 1

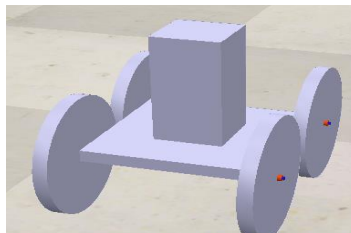
#### 6.1.1 Robot Body Selection

In the beginning, the prebuilt robot templates from CoppeliaSim were looked at to see if any of the robot bodies fit our requirements and objectives. They were narrowed down to two bodies that looked reasonable for our purposes and a list of pros and cons of potential pre-existing robot templates for use in our project was created as follows:

Robot Body	Pros	Cons
dr12: 	<ul style="list-style-type: none"><li>- Simple</li><li>- Easy to Implement</li></ul>	<ul style="list-style-type: none"><li>- Limit on time to perform task is the biggest issue</li></ul>
Omnidirectional Platform: 	<ul style="list-style-type: none"><li>- Advanced Movement</li><li>- Time Efficient</li></ul>	<ul style="list-style-type: none"><li>- Placement of plow may be difficult</li><li>- Complexity</li></ul>

After testing the bodies out in CoppeliaSim, we found that they were not specifically meant for what we were looking to achieve so we began looking into building our own body using the custom blocks in the program.

We aimed to create a body that was built in a similar fashion to the labs we had completed prior to the beginning of the project. However, this time we created a body with the maximum allowable size as specified in the project description and allowed room for a future plow to be attached to it. The body we had at this point is shown in the figure below:



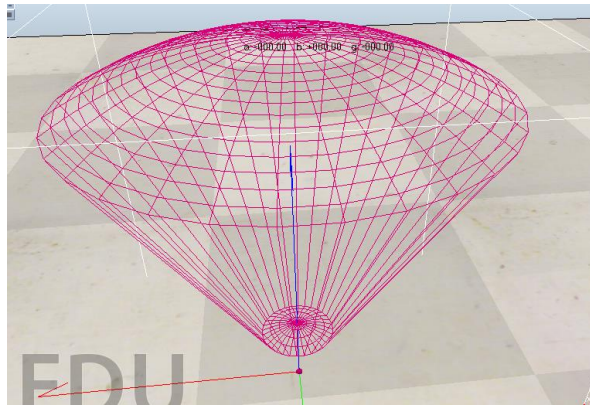
*Figure 7: Initial design of robot body*

This design has the ability for the robot to move within the movement restrictions, is within the size restrictions, and has a simple frame which will allow for easy integration with the plow.

### 6.1.2 Robot Sensor Selection

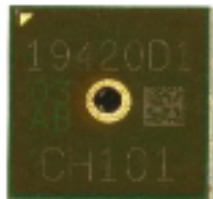
Multiple proximity sensors will be used to detect obstacles, one for the front left, one for the front middle and one for the front right.

The type of proximity sensor will be a randomized ray type sensor. These will be set as ultrasonic sensors in CoppeliaSim.



*Figure 8: Randomized ray proximity sensor in CoppeliaSim*

The real life sensor that this will be mimicking is the TDK CH101 [1], datasheet [here](#). This IC has a range of 4cm to 1.2m and can have a field of view of up to 180 degrees. The proximity sensor in CoppeliaSim will be edited to match these characteristics.

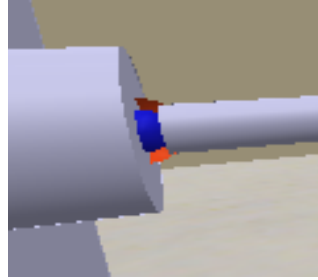


*Figure 9: CH101 Ultrasonic Sensor*

In addition to the object detection sensors, there will also be line-following sensors [2]. These will be the orthogonal vision sensors in CoppeliaSim. There will be three of these sensors for the left, middle and right. These sensors will be modified to match the [SparkFun RedBot Sensor - Line Follower sensor](#). [2]

### 6.1.3 Plow Controller Selection

The Plow controller will allow the body to control the plow and clear off the snow. The connection between the plow and the body is ultimately going to be an arm that allows for movement of the plow in the case of obstacles. Currently the spherical joint is going to be used to connect the body to the plow to allow for smooth movement of the plow.

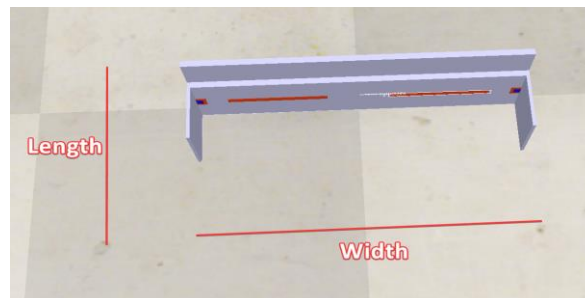


*Figure 10: Spherical joint attached to an arm to show connectivity*

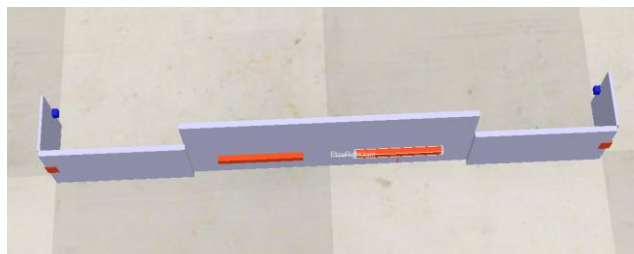
Another idea is having something close to a forklift, where the plow is stationary then is placed on the ground when it is ready to start shoveling or removing the snow spheres

### 6.1.4 Plow Design

The initial plow design is shown in the figures below:



*Figure 11: Plow before expansion with dimensions -> 0.5m (width) x 0.105m (length) x 0.15m (height)*



*Figure 12: Plow after expansion with dimensions -> 1.0m (width) x 0.11m (length) x 0.15m (height)*

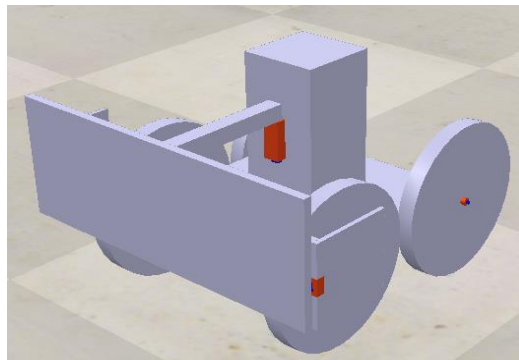
The spare space above the moveable part is used to attract the plow to the body.



## 6.2 Week 2

### 6.2.1 Plow and Body Integration

Now that the body, sensors, and plow have all been designed and selected, we began progress on integrating the components together into one scene. Connecting the plow to the body of the robot had a few complications in terms of the static nature of the connection of the arm to the dynamic body, but this was resolved and connected to look like the following figure:

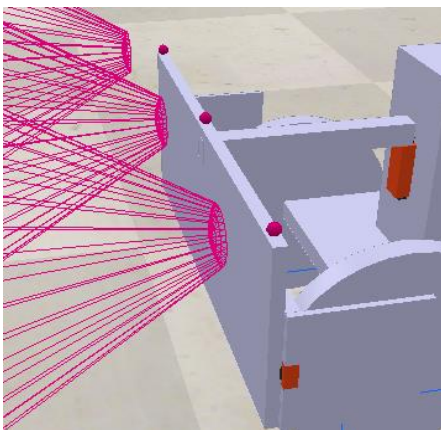


*Figure 13: Integration of the plow and robot body*

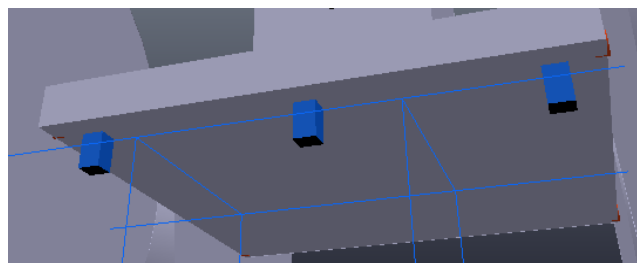
With the plow now attached to the robot with the arm, the robot can now move, and plow as planned for. This allowed us to begin to test the movement, speed, and the sizes at different stages of its movement. It currently has no behavior other than moving in a straight line because it does not have sensors yet although this will be implemented in following activities.

### 6.2.2 Sensor Integration

This activity consisted of integrating the object detection sensors into the plow as well as integrating the line-following sensors into the underside of the robot body.



*Figure 14: Object detection sensors on the front of the plow*



*Figure 15: Line-following sensors on robot*

### 6.2.3 Plow Controller Attachment

After testing the spherical joint controller, it was decided that it's not the best option due to its randomness in motion. Then, the forklift controller was implemented as shown below on a static cuboid. This was done to test if the forklift is able to attach to the plow and maneuver it such that when it brings the plow down, it doesn't interfere with the world frame. Also, changes to the controller were made so that it doesn't interfere with the plow's movement, where the plow is able to close and open properly.

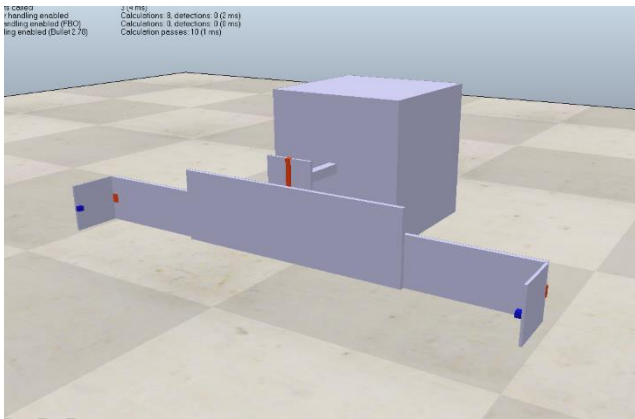


Figure 16: Plow after extended using controller

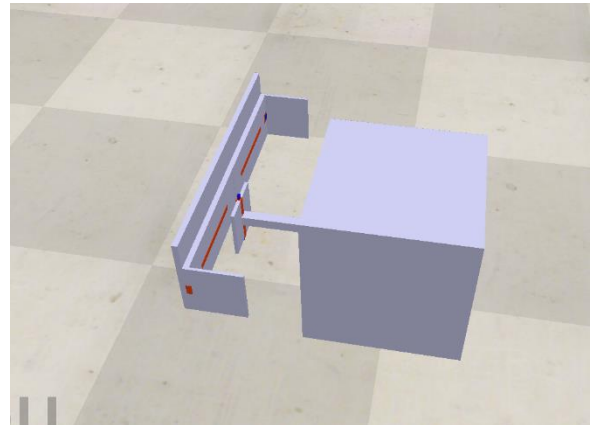


Figure 17: Initial state of the plow

After testing the forklift mechanism, it was implemented on the robot body that was chosen. Shown above is the attachment of the forklift mechanism to the robot body as the simulation starts. The attachment lowers the plow properly and doesn't interfere with the world frame.

### 6.2.4 Matching controller specifications to datasheets from real controller

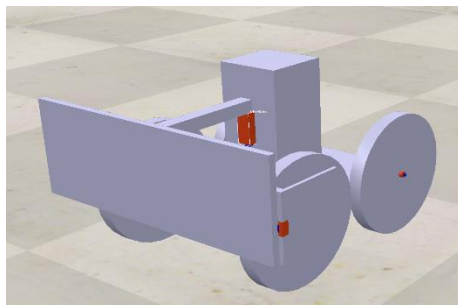


Figure 19: Arm controller

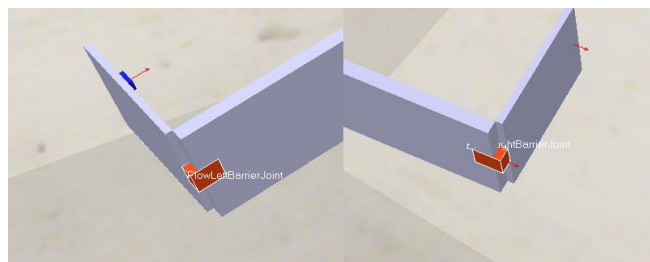
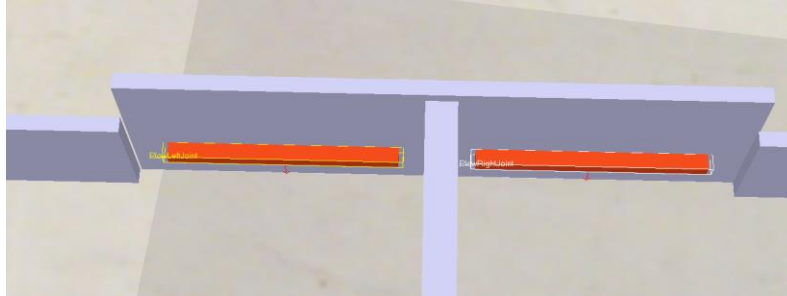


Figure 18: Barrier controllers

These three joints match the specifications from the following link:

[4" Stroke 115 lb Thrust Heavy Duty Linear Actuator - ServoCity](#) [3].



*Figure 20: Extension of plow controllers*

These two joints match the specifications from the following link:

[10" Stroke 115 lb Thrust Heavy Duty Linear Actuator - ServoCity](#) [4].

## 6.3 Week 3

### 6.3.1 Motor Control Via Python

A few functions are added to get the orientation (yaw, pitch, and roll in degrees) and the position of the robot ([move\\_straight](#), [move\\_right](#), and [move\\_left](#)). This was done using remote API functions from CoppeliaSim.[5]

### 6.3.2 Sensor Output to Python

Two Python functions were written to read sensor data from both the object detection sensors and the line-following sensors ([get\\_line\\_following\\_sensor\\_data](#) and [get\\_object\\_detection\\_sensor\\_data](#)). This was done using remote API functions from CoppeliaSim.[5]

### 6.3.3 Plow Control Via Python

Lua code to control the plow is rewritten to Python ([extend\\_plow](#) and [retract\\_plow](#)). This was done using remote API functions from CoppeliaSim.[5]

### 6.3.4 Controller Output to Python

The Python file to control the robot is initialized.

A few functions are added to get the orientation (yaw, pitch, and roll in degrees) and the position of the robot ([get\\_orientation](#), [quaternionToYawPitchRoll](#), and [get\\_position](#)).

These were done using remote API functions from CoppeliaSim.[5]

## 7. References

- [1] “CH101-00ABR,” *DigiKey*. [Online]. Available: [https://www.digikey.ca/en/products/detail/CH101-00ABR/1428-CH101-00ABRCT-ND/13174319?WT.z\\_cid=ref\\_netcomponents\\_dkc\\_buynow&utm\\_source=netcomponents&utm\\_medium=aggregator&utm\\_campaign=buynow](https://www.digikey.ca/en/products/detail/CH101-00ABR/1428-CH101-00ABRCT-ND/13174319?WT.z_cid=ref_netcomponents_dkc_buynow&utm_source=netcomponents&utm_medium=aggregator&utm_campaign=buynow). [Accessed: 01-Mar-2022].
- [2] AI8Z, M. #619676, M. #665168, Plexi, M. #673009, and Skye, “SparkFun Redbot sensor - line follower,” *SEN-11769* - *SparkFun Electronics*. [Online]. Available: <https://www.sparkfun.com/products/11769>. [Accessed: 01-Mar-2022].
- [3] ServoCity. 2022. 4" Stroke 115 lb Thrust Heavy Duty Linear Actuator. [online] Available at: <https://www.servocity.com/4-stroke-115-lb-thrust-heavy-duty-linear-actuator/> [Accessed 1 March 2022].
- [4] ServoCity. 2022. 10" Stroke 115 lb Thrust Heavy Duty Linear Actuator. [online] Available at: <https://www.servocity.com/10-stroke-115-lb-thrust-heavy-duty-linear-actuator/> [Accessed 1 March 2022].
- [5] Coppeliarobotics.com. 2022. *Remote API functions (Python)*. [online] Available at: <https://www.coppeliarobotics.com/helpFiles/en/remoteApiFunctionsPython.htm> [Accessed 1 March 2022].