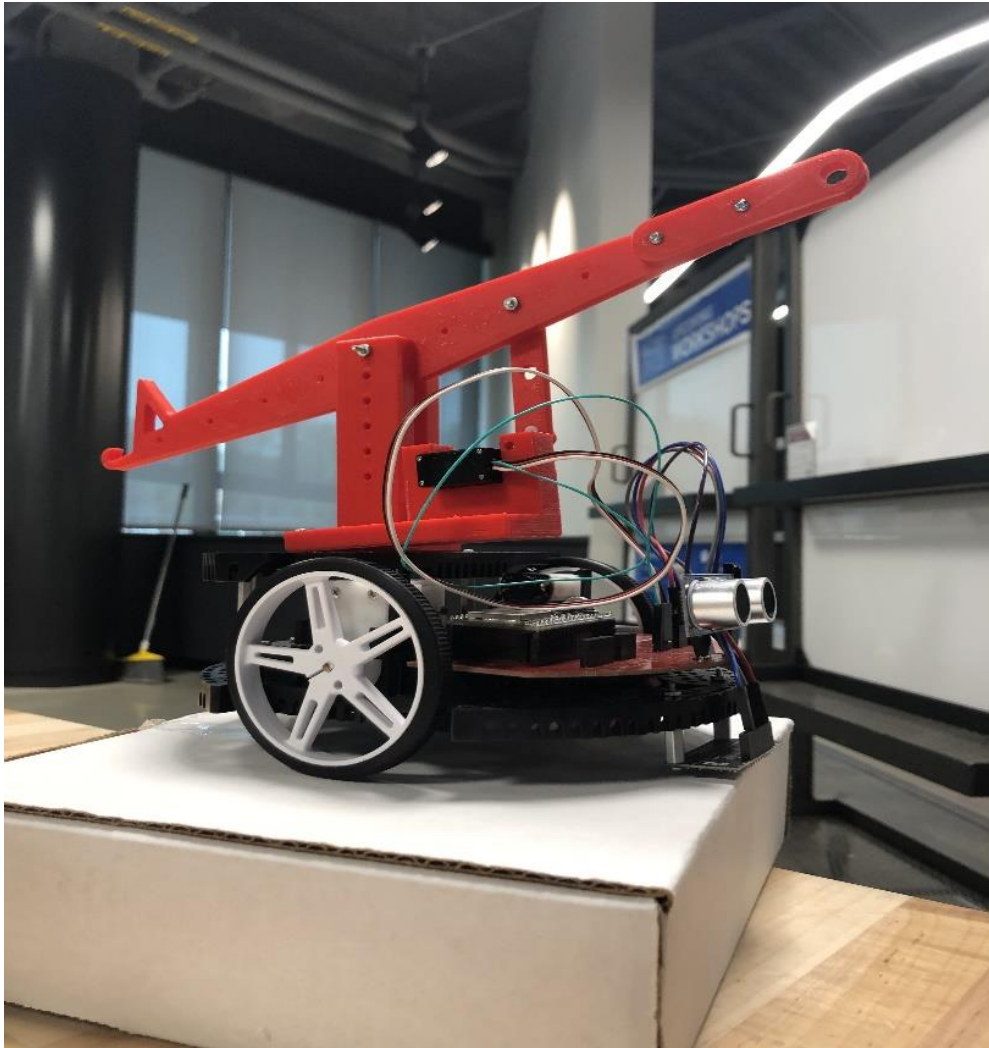


RBE 1001 D21, Introduction to Robotics

Final Project Report



Team 10

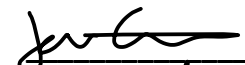
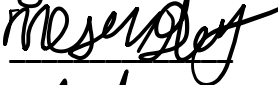

Member	Signature	Contribution(%)
Jacob Chlebowski		33.3
Megan DeSanty		33.3
Aiden Veccia		33.3

Table of Contents

List of Figures	ii
Section One – Introduction	1
Section Two – Preliminary Discussion	3
Section Three – Problem Statement	4
Section Four – Selection of Final Design	5
Section Five – Final Design Analysis	6
Section Six –Pseudocode	14
Section Seven – Project Summary	19

List of Figures

Figure 1: Diagram of Robot Mechanisms	1
Figure 2: Road Map	2
Figure 3: Free Body Diagram of Romi	6
Figure 4: Maximum Torque Configuration 1	7
Figure 5: Force on Link CD	7
Figure 6: Calculated Counterweight Configuration 1	8
Figure 7: Calculated Target Weight with Counterweight	8
Figure 8: Calculations of CM and Percentage on Wheels	9
Figure 9: Center of Mass and Percentage on Wheels with Bag Attached	10
Figure 10: Free Body Diagram of Configuration Two	11
Figure 11: Finding a Counterweight	11
Figure 12: Calculations for Target Lifting Weight	12
Figure 13: Calculations for Center of Mass and Percentage On Wheels	12
Figure 14: Center of Mass and Percentage on Wheels with Bag	13

I. Introduction

Each team in RBE 1001 was tasked with building and programming a robot capable of picking up and delivering bags to designated zones on a course. Every delivery that weighs forty grams is worth one point, and each free-range bag is worth five points. In addition to bag pickups, each team must deal with a roadblock that could be placed at any time along an intersection. If the roadblock is placed, the robot must act accordingly and go around it.

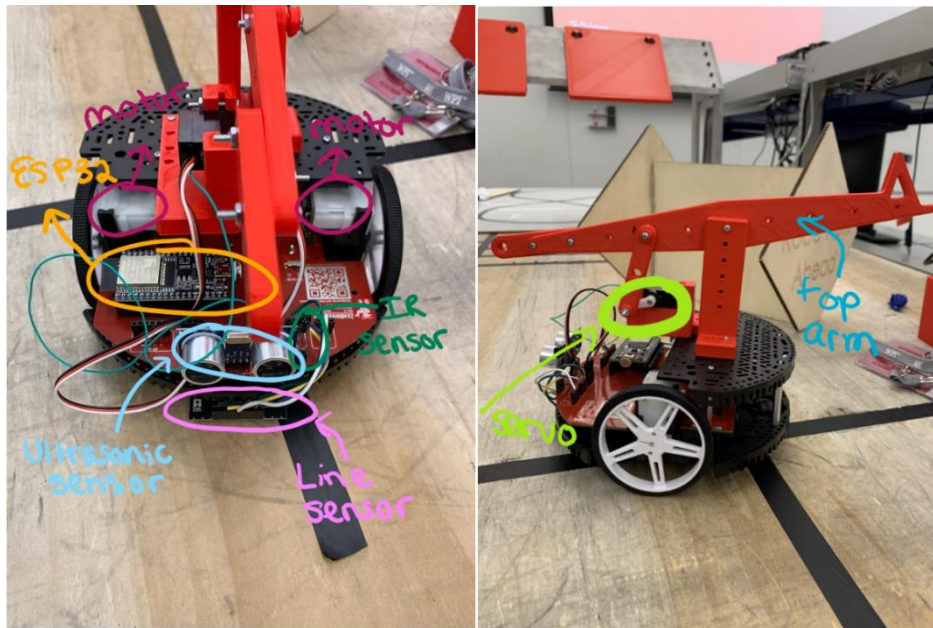


Figure 1: Diagram of robot mechanisms

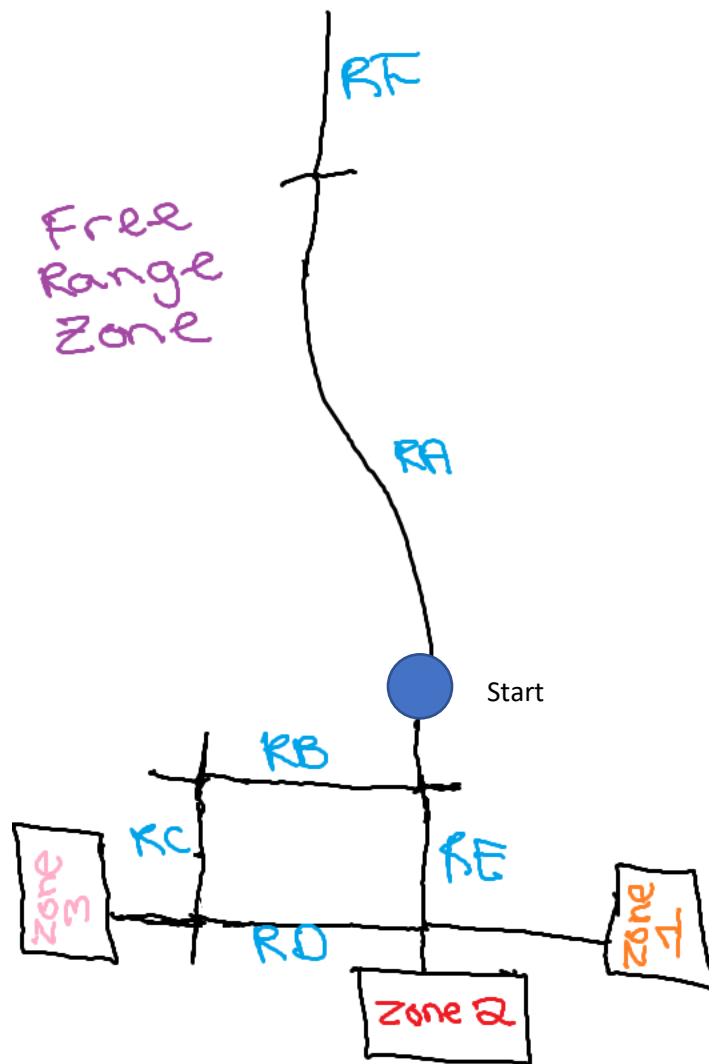


Figure 2: Road Map

II. Preliminary Discussion

Our goal was to receive as many points as possible by delivering at least 1,000g total. To accomplish this goal, we had to make sure our robots were set to a speed such that it can travel to each zone numerous times without exceeding the stall torque of the motor. Because we were limited by the amount of weight the robot could carry, we agreed to focus on getting the robot to move from each zone as quickly as possible to reach our goal of carrying 1,000g.

For each task to be executed in six minutes, the robot was programmed with a range finder, line follower, and IR remote, and included specific mechanical mechanisms. Firstly, the robot incorporates a drivetrain, which is composed of two DC (direct current) motors, that is responsible for turning a wheel on the robot. With the drivetrain, the robot was able to get to the bag. The next step was to execute an action. The lifting mechanism on the robot comprised of the one-hundred-eighty-degree servo (FS90-FB) that was arranged to ensure sufficient torque without tipping the robot when picking up and delivering the bag. For the delivery process to be completed, the ESP-32 Processor was programmed with control algorithms in C++. The IR sensors and the navigation algorithm utilizes the robot's position to accurately traverse through the course.

III. Problem Statement

Our team had three goals for the design of the robot: a robot configuration that can reach each platform, code that allows the robot to understand which section of the road it is on, and a configuration that maximizes the amount of weight the robot can carry to each zone.

To have a robot configuration that can reach each platform while maximizing the amount of weight it can carry to each zone, we determined this could be accomplished by having more than one configuration. This would allow us to maximize weight on one robot while being able to reach the highest platform with another.

For the robot to be able to recognize which road it is on we decided to use state machines and switch statements. This would allow us to program the robot in a way where it can understand which road it is on based on specific commands that allow the robot to switch between states.

IV. Selection of Final Design

When selecting each part of our robot, we considered our design goals and competition strategies. Firstly, because we wanted the robot to understand what road it was on, we used state machines. This allowed the robot to switch between states depending on certain actions (such as hitting an intersection, crossing a line a certain number of times, etc.). In addition to state machines, we also used a cup counter that allowed the robot to determine which drop off location to travel to. If the robot only had one cup it would go to zone one, two cups to zone two, three cups to zone three, and back to the beginning. The cup count was crucial to our design because the robot could understand which zone to go to.

In terms of mechanics, we decided to have our robot pick up the bag in the rear. We decided on this configuration because if the bag were picked up in the front, there was a chance the bag could have blocked the ultrasonic sensor. If the sensor were to be blocked, it would not be possible to detect platforms and the free-range bag.

The servo bracket was placed at the same location as it was during the activities throughout the term. We kept this placement because with the arm configuration, this allowed for the robot to deliver a greater amount of weight without hitting its stability limit.

In the end, we agreed on two arm configurations. One configuration was able to place the bags in zones one and two, and the other was able to place the bag in zone three. Since our goal was to deliver 1,000g, we decided to have the robot configured for zone three only go to zone three because it was not able to carry as much weight as the configuration for zones one and two. This meant that the configuration for zones one and two would have to take less trips than the robot configured for zone three.

V. Final Design Analysis

A. Mechanical Analysis

For our final design selections, the team was able to create two different configurations to successfully gather points in each of the zones and pickup/drop-off bags.

Configuration1: The range finder was mounted in the front to compute the robot's distance from any bag or zone. This was especially useful when the romi had to find a way around the roadblock, which was accounted for in the program using state machines. It was also useful in finding the free-range bag, which would check within 50cm to find the bag.

The arm was properly measured to ensure maximum torque. This occurred when the distance from the pivot point to the end of the arm was roughly equal.

One of our first priorities was to be able to pick up as much weight as possible without the robot tipping. This was calculated using free body diagrams and equations of equilibrium.

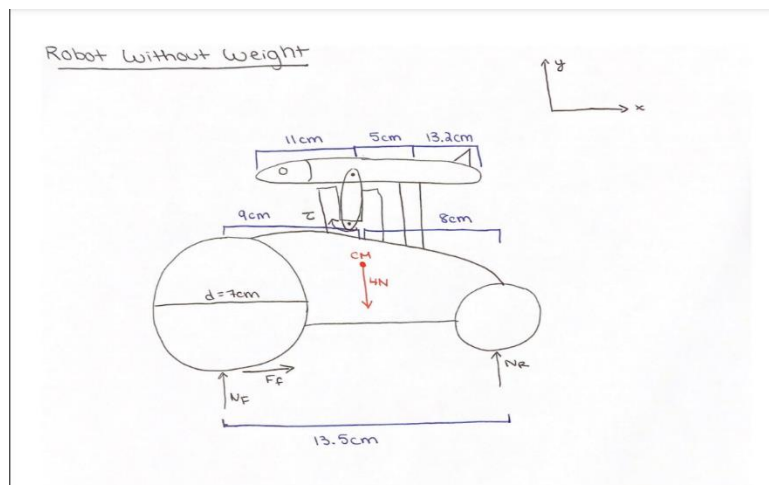
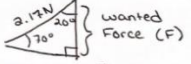


Figure 3: Free Body Diagram of Romi.

Based on this arm configuration, and the force on link CD, we were able to determine the maximum torque that the servo arm can handle.



$$\cos(20) = \frac{F}{2.17}$$

$$0.9397 = \frac{F}{2.17}$$

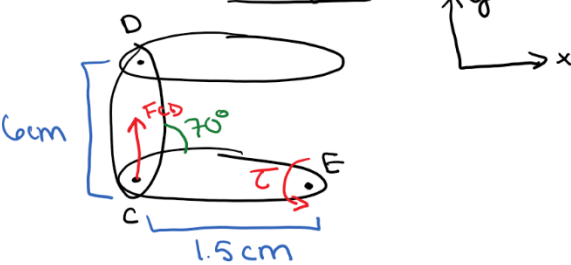
$$F = 2.04 \text{ N}$$

$$\tau = 2.04 \text{ N} \times 1.5 \text{ cm}$$

$$= 3.06 \text{ N-cm}$$

Figure 4: Maximum Torque Configuration 1.

Finding F_{CD} :



$$\sum M_E = 0 = -1.5 \text{ cm} (F_{CD} \sin(70)) + \tau_{\text{stall}}$$

$$-1.5 \text{ cm} (F_{CD} \sin(70)) = -3.06$$

$$F_{CD} = 2.17 \text{ N}$$

Figure 5: Force on link CD.

For the robot to be able to carry more weight than our calculated target weight, we calculated a counterweight.

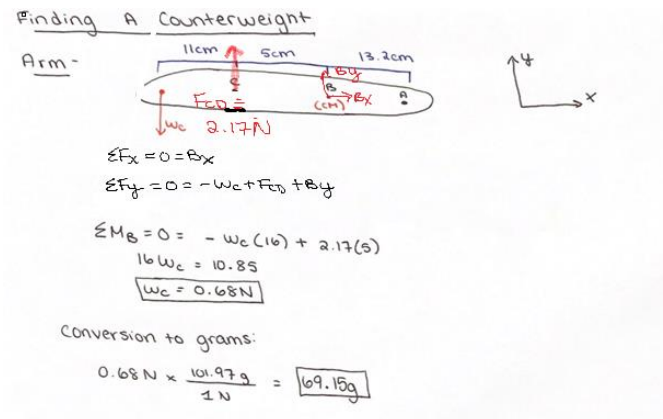


Figure 6: Calculated Counterweight Configuration 1.

Given the calculated counterweight, we were able to determine how much the robot can lift when the counterweight was attached. These calculations allow the robot not to exceed the stall torque because the force on link CD was calculated using the stall torque of the romi at 5V.

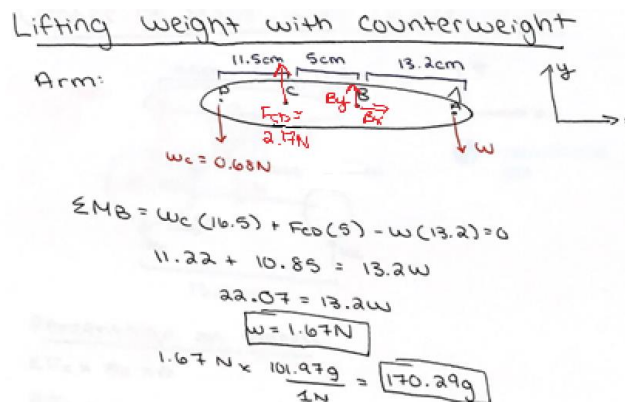


Figure 7: Target Lifting Weight with Counterweight Attached.

Now that the counterweight is calculated, we can calculate the percentage of the weight on each wheel. First, we had to calculate the center of mass with the counterweight attached. This was calculated using a proportion of the distances to the weights. Once this was calculated, we found the percentage by solving for the normal force on each wheel given the distances from the center of mass to each wheel. As seen in Figure 7, most of the weight was placed on the large wheel when only the counterweight is attached.

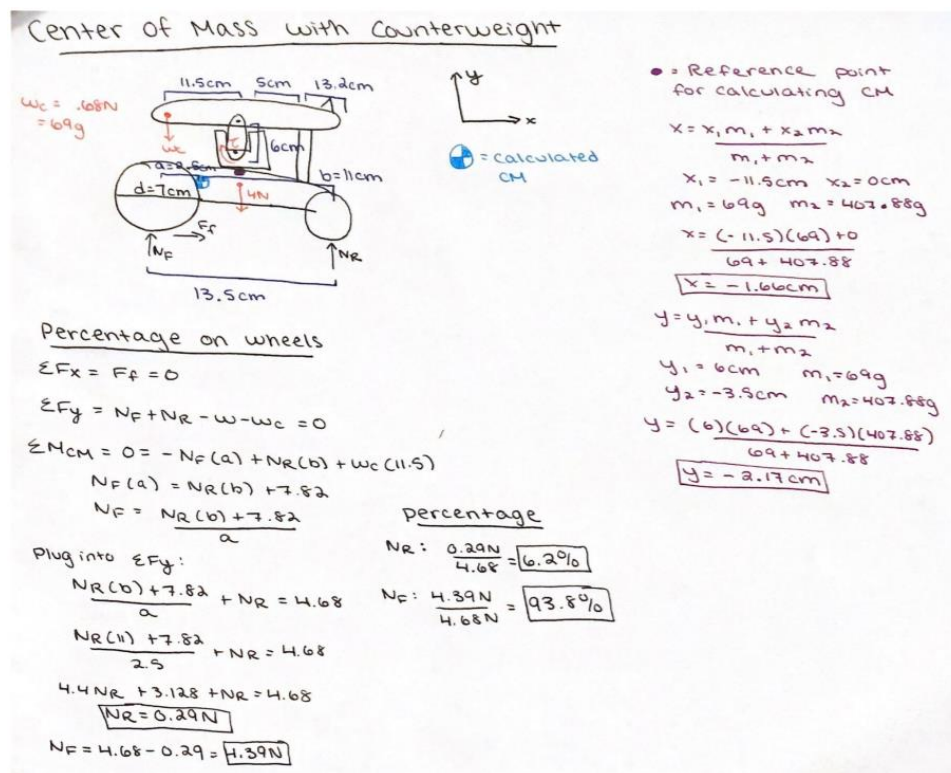


Figure 8: Calculations for Center of Mass and Percentage on Wheels.

The same method in Figure 7 was then applied to Figure 8 below. The center of mass with the counterweight and the bag was calculated. In the end, there was more weight put on the large wheel but not as much when only the counterweight was attached.

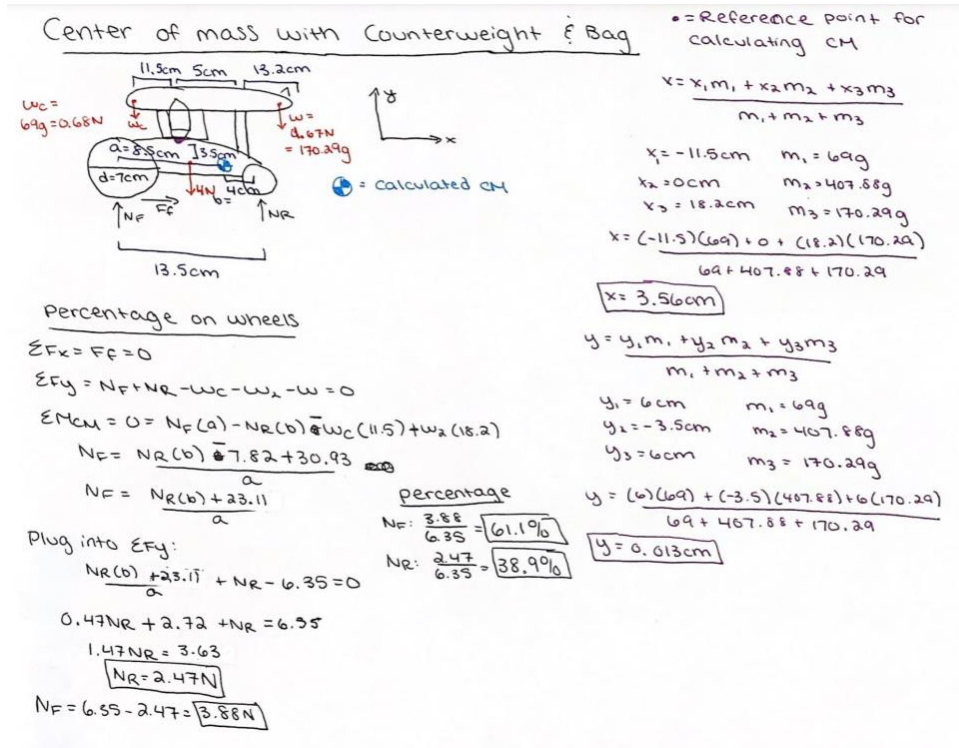


Figure 9: Center of Mass and Percentage on Wheels with Bag Attached.

Configuration 2: In our second configuration we changed the position of the arm relative to the servo. This configuration was used only for zone three, as it has a wider range of vertical motion but at the cost of not being able to carry as much weight. Similar to the calculations for configuration one, we used free body diagrams and equations of equilibrium to determine how much weight the robot is able to carry to zone three.

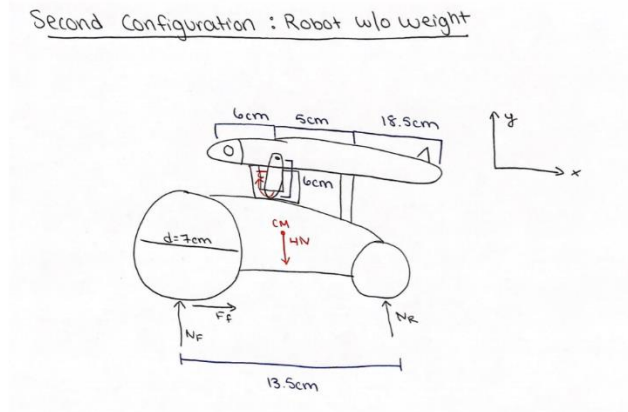


Figure 10: Free Body Diagram of Configuration Two.

To find the counterweight for configuration two, we used the same methods as we used for configuration one. The force on link CD and the stall torque have not changed (refer to Figures 4 and 5).

and Configuration : Finding a counterweight

$$\sum F_x = 0$$

$$\sum M_B = 0 = \frac{1}{2} W_C (11) - 2.17(5)$$

$$11 W_C = -10.85$$

$$W_C = -0.99 \text{ N}$$

Conversion:

$$0.99 \text{ N} \times \frac{101.97 \text{ g}}{1 \text{ N}} = 101 \text{ g}$$

Figure 11: Finding a Counterweight.

Because the counterweight is known, we can find the target lifting weight with the counterweight attached.

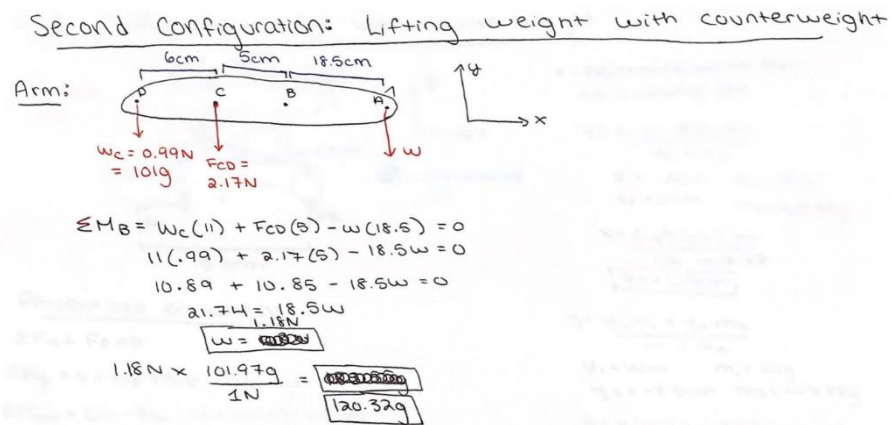


Figure 12: Calculations for Target Lifting Weight.

The center of mass and percentage on the wheels used the same method as configuration one. After calculations, we found that with only the counterweight attached, most of the weight was placed on the large wheel.

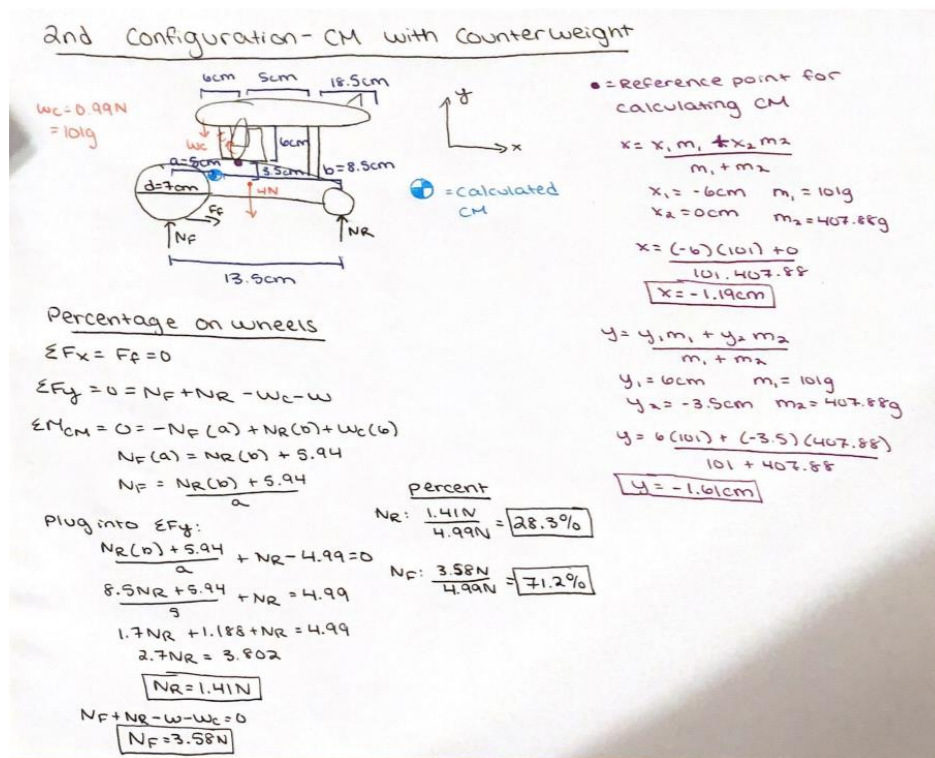


Figure 13: Calculation for Center of Mass and Percentage on Wheels.

Similar calculations were made when the bag was attached. According to Figure 13 there is more weight placed on the small roller wheel in the front. This is likely because the arm was configured so that there is more distance from the pivot point to the end of the arm.

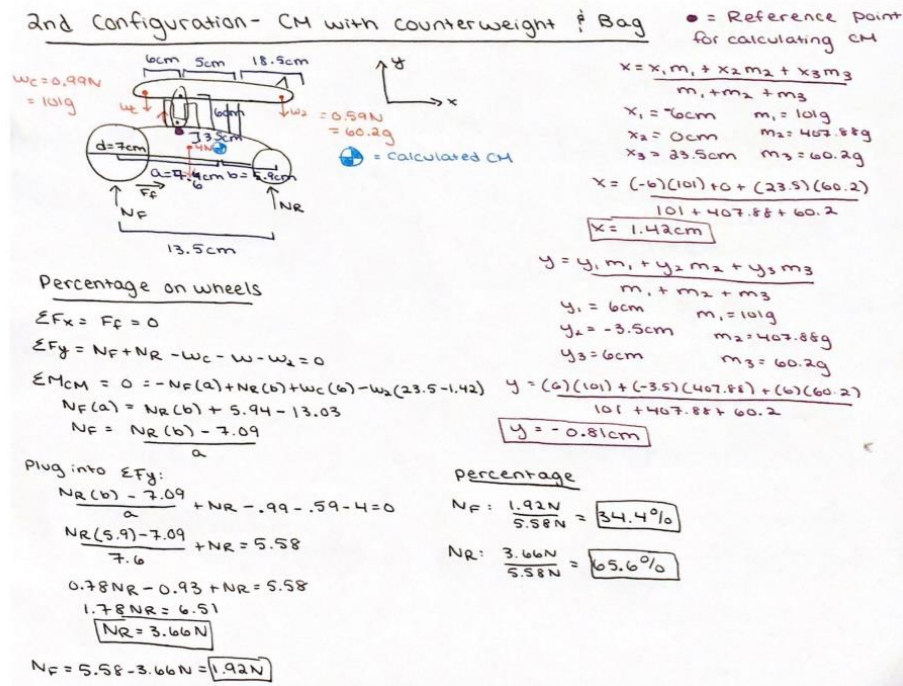


Figure 14: Center of Mass and Percentage on Wheels with Bag.

B. Programming Methodology

Our program utilized state machines to have the robot understand which road it was located on while traversing through the roads. This allowed us to be able to program in a more concise versus using “if statements”. Additionally, the states allowed us to tell the robot to only drop the arm when it was in the searching mode or at one of the zones.

VI. Pseudocode

Refer to Figure 2 for where each road and zone is located. Each road and zone correspond to each state in the code.

Robot 1 will get the bag from the free range and drop it in zone 1.

Robot 2 will get the bag from road F and drop it in zone 2.

Robot 3 will get the bag from road F and drop it in zone 3.

All robots will START and STOP at the bottom of road A (right next to road E), facing away from road E.

Define state freeRange()

Line follow until `getCurrentDegrees` \geq 900 (this is a timing method).

Spin left 70 degrees.

Slowly spin right until bag is sighted.

Keep spinning until bag is no longer visible (record degrees turned).

Turn back to face bag (half the recorded degrees).

Move forward until `ultrasonic.getDistanceCM()` \geq 12 (bag is within 12 cm).

Use function `pickup()`.

Turn until robot is perpendicular with the line.

Drive towards the line until it is directly under the robot.

Turn right 90 degrees (robot will be on road A after this function).

Define function roadF()

Line follow until new road is reached (the robot will go from road A to road F).

Move forward until `ultrasonic.getDistanceCM() >= 12` (bag is within 12 cm).

Use function `pickup()`.

Line follow until new road is reached (robot will be on road A after this function).

Define function `pickup()`

Turn 180 degrees.

Reverse for half a second.

Raise arm to 180 degrees (max height).

Define function `dropoff()`

Turn 180 degrees.

Drop arm to 0 degrees (min height).

Define function goAround()

If the robot is holding a package (it will always be on road A facing road E at this time if it has a package):

Turn right 90 degrees onto road B.

Line follow until new road is reached.

Turn left 90 degrees onto road C.

Line follow until new road is reached.

Robots 1 and 2 will turn left 90 degrees onto road D, then line follow until a new road is reached.

Robot 3 will turn right 90 degrees.

Robot 1 will not turn.

Robot 2 will turn right 90 degrees.

(All robots will be in front of their respective zones at the end of this function).

If the robot is not holding a package (it will always be coming back from a drop off zone if it does not have a package).

Turn left 90 degrees onto road D, then line follow until a new road is reached.

Turn right 90 degrees onto road C, then line follow until a new road is reached.

Turn right 90 degrees onto road B, then line follow until a new road is reached.

Turn left 90 degrees onto road A.

START

If this is the robot 1, use function freeRange()

If this is the robot 2 or robot 3, use function roadF()

Line follow until new road is reached (all robots will be on road A at this point facing towards road E).

If roadblock is not present, line follow down road E until a new road is reached.

Robot 1 will turn left 90 degrees (it is now in front of zone 1).

Robot 2 will go straight (it is now in front of zone 2).

Robot 3 will turn right 90 degrees, line follow down road D until a new road is reached, and go straight (it is now in front of zone 3).

If roadblock is present, use function goAround().

Robot 1 will move forward to zone 1 and use function dropOff().

Robot 2 will move forward to zone 2 and use function dropOff().

Robot 3 will move forward to zone 3 and use function dropOff().

Robot 1 will line follow until a new road is reached, turn right 90 degrees.

Robot 2 will line follow until a new road is reached.

If a roadblock is present both robot 1 and robot 2 will use function goAround().

Robot 3 will line follow until a new road is reached.

Robot 3 will turn left 90 degrees onto road C, then line follow until a new road is reached.

Robot 3 will turn right 90 degrees onto road B, then line follow until a new road is reached.

Robot 3 will turn left 90 degrees onto road A.

STOP

The above code will run two times each for robots 1 and 2 (who each carry 170g of weight), and three times for robot 3 (who carries 110g of weight). This results in a total of 1,010 grams of weight delivered into the three zones in under six minutes utilizing two free range bags, all robots, and all zones.

VII. Project Summary

Our first task was to get the free-range bag. Starting at Zone A, the first romi successfully traversed until it reached the curve, and then searched for the bag. In our first attempt, the range finder did not quite pick up the bag, and as a result, turned the opposite direction away from the bag (as an emergency mechanism to not finding the bag). Understanding the risks of moving the robot to pick up the bag, we were able to gather the points right back after going to Zone one using the IR remote. In our second attempt at the free-range, this time placed a little further, the robot correctly calculated the midpoint distance of the bag and was able to pick it up and bring it to Zone 1.

Besides the free-range complications, the rest of the demonstration went smoothly. Both other robots with their Zone 2 and Zone 3 configurations were able to grab the bags and line follow. Again, the IR remote in the beginning was able to specify which zones we wanted to go to.

If we more time for this project, we could have updated the search state for when the romi searches for the free-range bag. We would have added more of a turn for the robot to be perpendicular to the line to then turn and line follow back to the respective zone. If the right sensor hit first, we would have added an extra twenty degrees (which could change with testing) and add even more for when the left sensor hits first. This is all to ensure that the robot would correctly lineup with the line to then carryout the rest of the delivery.

We feel we succeeded in both the demo and in teamwork. We were able to focus on the final project starting in the fifth week of the term which gave us more time to think diligently about how we were going to execute this task. For being the only three people who did not have

a group at the beginning of the term, we can successfully say that everything worked out in the end.