Engineering Design Notebook

ENED1120: Lego Robot

Written by: Luke Weis, Arpit Gill, AJ Armstrong,

Alex Funk

Table Of Contents

ENED 1120 Project: Design Specification Review

Due: Sunday, February 4th by 11:59 p.m.

TEAM ASSIGNMENT: See the course syllabus for a definition of what constitutes an individual assignment.

OBJECTIVE: To work with your team members to develop a plan for the design and management of your Team's Project.

1. **Design Notebook:** The information you generate in this review should be included in your Design Notebook. More information on the Design Notebook format and expectations can be found in the Project Materials Link.
2. **Meeting Minutes:** Use the template in the Project Materials Link to start documenting your meetings.
3. **Empathize and Define:**
    a. Identify at least five relevant stakeholders.
        i Buy More's higher-ups.
        ii Us, the engineers.
        iii The consumer.
        iv The employees of Buy More.
        v Our engineering firm.
    b. List the wants and needs for each stakeholder (at least two per stakeholder).
        i Buy More's higher-ups want the robot to work efficiently.
        ii Buy More's higher-ups want the robot to be cheap.
        iii We want the robot to work to make us look good.
        iv We want the robot to be easy to design.
        v The robot works well so the consumer can get their product fast.
        vi The consumer wants the robot to work well, so their product does not break.
        vii The employees do not want to be hit by the robot.
        viii The employees want the robot to work well, so they have less work.
        ix Our engineering firm needs the robot to work so they look good.
        x Our firm wants it to work so that they do not lose a client.
    c. Complete Table 2 (add rows as needed) using Table 1 as a reference for the expected entries in each cell. Table 1 includes an example for a water bottle. You should cut and paste your completed table into your design notebook.

Table 1. *Example of a few wants/needs, criteria, and specifications for a water bottle*

Stakeholder Wants/needs	**Criteria**	**Specifications/Target Values**
The customer needs to be able to have liquids available when they're walking	Must hold liquid	30 oz
Customer needs the liquid to stay cold	Must maintain temperature	Within + or – 4 °F of 60 °F for 8 hrs.

Table 2. *Wants/needs, criteria, and specifications*

Stakeholder - Wants/needs	**Criteria**	**Specifications**
Buy More's higher-ups want the robot to work efficiently.	Must work fast.	Completes run in less than 3 minutes.
Buy More's higher-ups want the robot to be cheap.	Must not cost too much money.	Must be made from parts provided to us.
We want the robot to work to make us look good.	Must impress Buy More	Our design is chosen.
We want the robot to be easy to design.	Must not take long to design.	Takes less than April to design
The robot works well so the consumer can	Must work fast.	Take less than 3 mins to complete a loop.

get their product fast.		
The consumer wants the robot to work well, so their product does not break.	Must not hit the boxes.	Hit less than 1 box in a loop.
The employees do not want to be hit by the robot.	Must not hit the employees.	Robot must stop when it sees an employee.
The employees want the robot to work well, so they have less work.	Must not mess up tasks.	Must correctly identify the correct product 95% of the time.
Our engineering firm needs the robot to work so they look good.	Robot successfully meets all of the specifications.	Our design is chosen.
Our firm wants it to work so that they do not lose a client.	Must work efficiently.	The client continues to work with us.

d. List any ethical issues related to profit, politics, person gains, or pressure to perform that might impact your design process.
   i If we are not careful people will get hit and hurt.
   ii If we work too quickly, we might overlook necessary functions.
   iii Putting employees out of business because our robot is too good.

e. List **constraints** impacting the project and your design. Consider any environment factors that might impact the performance of the robot.

   i Must use Legos or 3d printed pieces.

   ii Must finish by the deadline.

   iii Can't hit any boxes or people.

   iv Can't be more than a foot wide.

Notes for 3a-3e

- Please remember, the define stage does not describe how you intended to solve the problem; you are precisely defining the problem, any requirements, how you will evaluate the effectiveness of ideas towards a solution.
- Included in the Project Materials folder is a list of the electronic LEGO EV3 equipment you will have once you pick up your kits. The specifications of this equipment and the number of parts can help identified constraints.
- An environmental factor is an element that is not within your control but could impact the performance of your design.

f. Based on the criteria you developed, **list the major functions of the robot** and provide a brief description of the major functions you envision your robot needing to accomplish.

   i Identify a product by a barcode.

   ii Know its location in the warehouse.

   iii Return to home base.

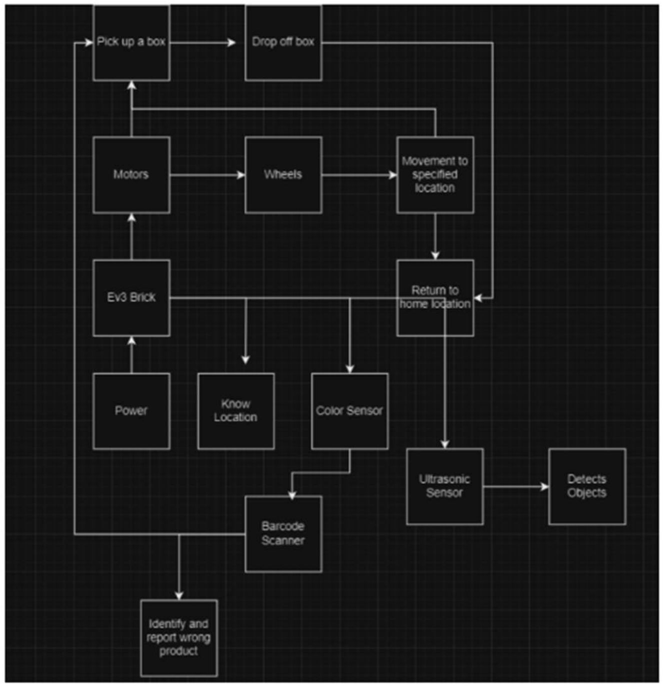
   iv Pick up the correct box.

   v Drop off the box at the right location.

   vi Move to the location specified.

   vii Identify and report a wrong product.

g. Create at least one Functional Block Diagram to demonstrate how the robot's functions relate to one another. Include all inputs and outputs.

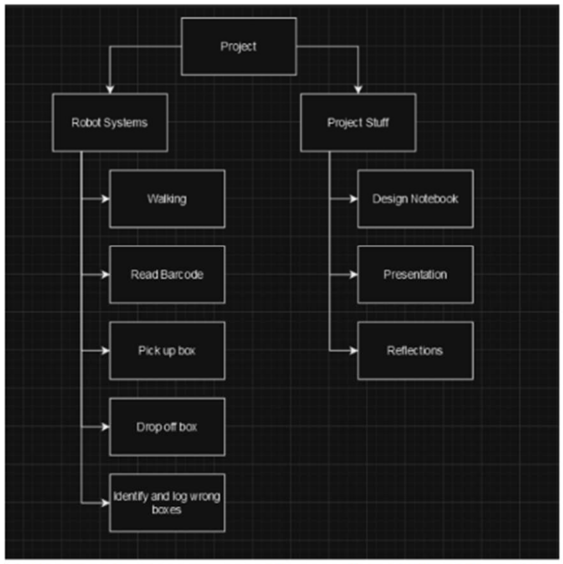


Notes for 3f-3g

• Please remember, this is a description of the required functionality and not a description of your solution or list of parts. In other words, *what* does the robot do to achieve its goals? You are not concerned with *how* it will do that action right now.

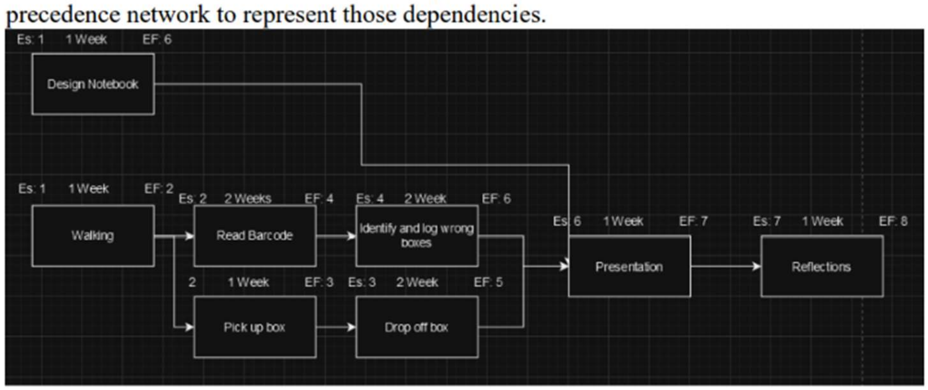
4. **Project Management Information**,
   a. Based on the different functionality you identified and various deliverables associated with the project, consider what major tasks need to be completed in order to create the necessary prototype.
      a Create a movement system.
      b Create a barcode scanning system.
      c Create a presentation.
      d Create a design notebook.
      e Create box identification system.
      f Create bow lowering system.
   b. Create a Work-Breakdown Structure based on your list of major tasks.

ENED 1120 Project – Fall 2024



c. Estimate the time to complete each task based on the timeline of deliverables.
   a Walking – 1 week
   b Read Barcode – 1 week
   c Pick Up box – 1 week
   d Drop Off Box – 1 week
   e Design notebook – entire project
   f Presentation – 3 days
   g Reflection – 1 day

d. Examine your major tasks and deliverables to determine the sequence in which they need to be completed. In other words, in what order do your tasks need to be completed to achieve the project goals? Create a precedence network to represent those dependencies.

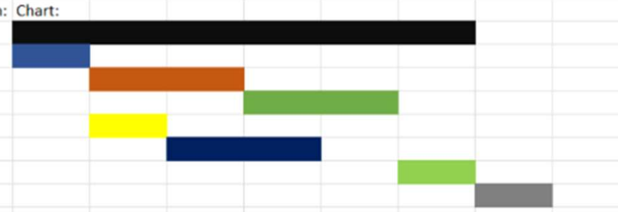


ENED 1120 Project – Fall 2024

6 / 7

e. Using your precedence network, create Gantt chart that indicates your initial conception of a project schedule based on the tasks identified.

Task:	Time Taken:	Chart:
Design NoteBook	6 weeks	
Walking	1 week	
Read Barcode	2 weeks	
Identifiy and log wrong boxes	2 weeks	
Pick up box	1 week	
Drop off box	2 weeks	
Presentation	1 week	
Reflections	1 week	



Notes for 4a-e

• When creating your schedule, consider how you might delegate tasks to in dividuals – not just to the "team" generally. Who can take the lead on spe cific functionality and team management tasks?

• This is also a good time to determine roles on the team. These can vary from management roles to roles on specific functions. These roles can (and should) rotate throughout the semester.

4. **Estimated Project Budget**, the estimated costs will be calculated based on the total number of person hours devoted to the project. Using your project management evaluations, estimate the number of hours required by your team for each task of the project (i.e., estimate the total number of hours everyone will spend on each task from the start of the project to the end). For this type of design activity, the standard billing rate your consulting firm charges is $150 per person hour;

   It will take up each 25 hours. 25 * 4 = 100 hours for a total of 100 hours of work. Total of $15,000 of work.

Submit a document electronically to your CANVAS Section Site using the file name ***Project_DS_TeamXXX.pdf***, where *XXX* is your 3-digit team number. This document is essentially the beginning of your Design Notebook.

Within the same submission link, you should also submit your Team's Code of Cooperation (see file on Canvas Community Page). The file name should be CodeOfCooperation_TeamXXX.docx or .pdf.

Meeting Agenda / Minutes **Project/Team: ENED 1120 Robot Team 238**

Date/Time: 1/31/24, 3:00pm-5:00pm Note Taker: Luke Weis

Present: Luke Weis, Alex Funk, AJ Armstrong, Arpit Gill

Agenda: Complete Part Inventory, Create Project Design Specification Review, Create Code of Cooperation

Review Gantt Chart

☐ Tasks added to chart: Chart was created

Foreseeable issues that will prevent success ☐ No ☐ Yes

Decisions made:

- Defined criteria
- Created Gannt Chart

Important information shared:

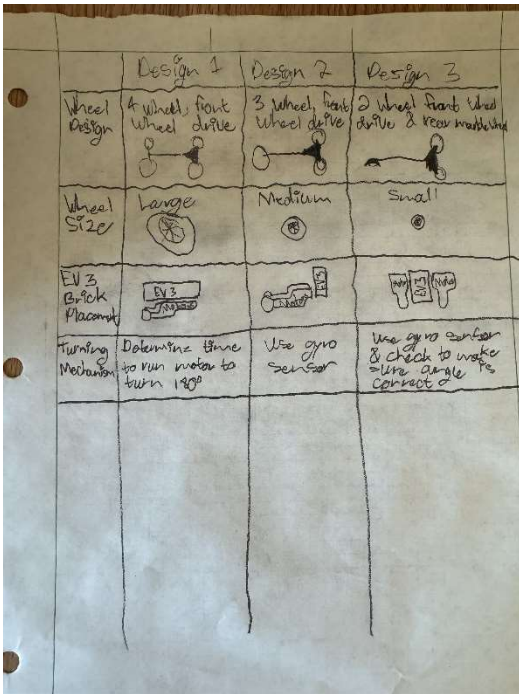
- Shared contact information, team/individual norms, team goals, and defined criteria for success on the project

Ideas we want to remember:

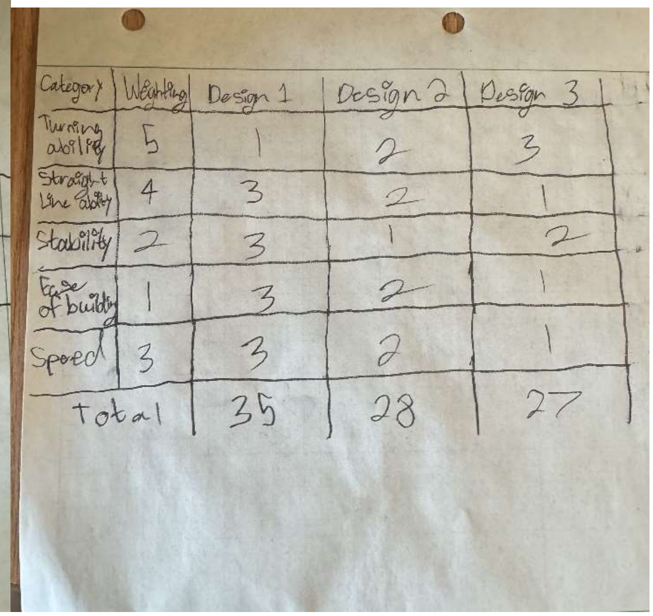
Team member	Hours on project since last meeting (hrs)	Total hours on project (hrs)
Luke Weis	0	2
AJ Armstrong	0	2
Alex Funk	0	2
Arpit Gill	0	2

Luke Weis, Arpit Gill, AJ Armstrong, Alex Funk, 11

Subtask 1 Design



	Design 1	Design 2	Design 3
Wheel Design	4 wheel, front wheel drive	3 wheel, front wheel drive	2 wheel front wheel drive & rear [illegible]
Wheel Size	Large	Medium	Small
EV3 Brick Placement			
Turning Mechanism	Determine time to run motor to turn 180°	Use gyro sensor	Use gyro sensor & check to make sure angle is correct



Category	Weighting	Design 1	Design 2	Design 3
Turning ability	5	1	2	3
Straight line ability	4	3	2	1
Stability	2	3	1	2
Ease of building	1	3	2	1
Speed	3	3	2	1
Total		35	28	27

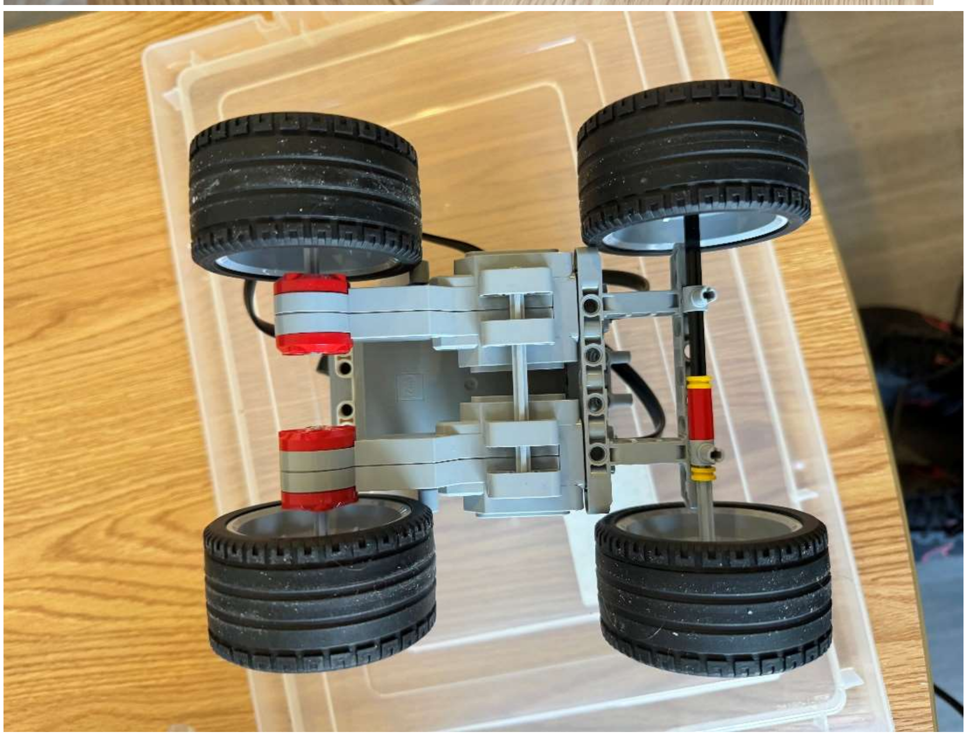
We decided to go with design 1 as it scored the highest after creating a morphological chart and a design decision matrix.

2/13/24

Luke Weis, Arpit Gill, AJ Armstrong, Alex Funk, 12

Iteration 1

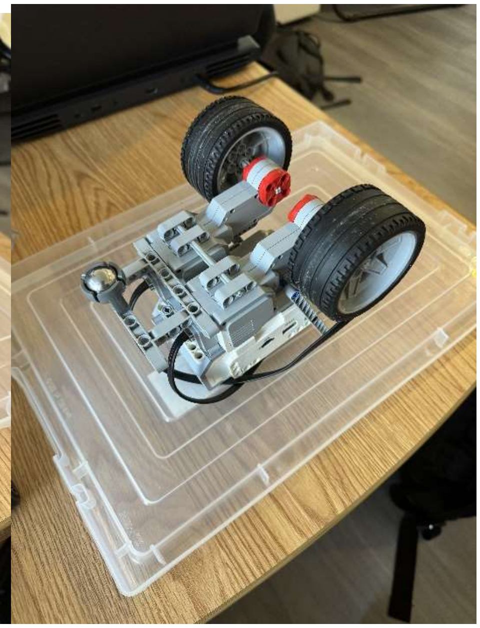




This is our first iteration of our robot. We chose a four wheel design because it offered high stability and a high probability of the robot moving in a straight line. The robot did successfully move in a straight line, but failed to be able to efficiently and predictably turn.

2/13/24

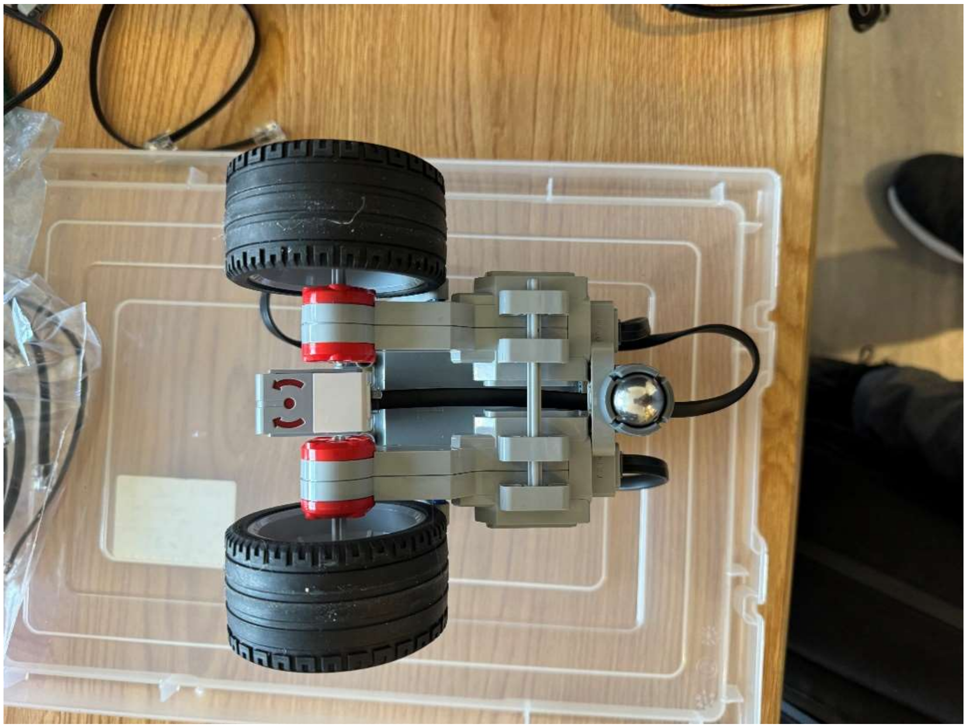
Iteration 2

The second iteration of our robot is similar to our first, but the back wheels were replaced by a singular marble wheel. We found that due to the marble wheel’s ability to turn 360 degrees, the robot was able to easily turn, while stilling reliably traveling in a straight line.

2/13/24

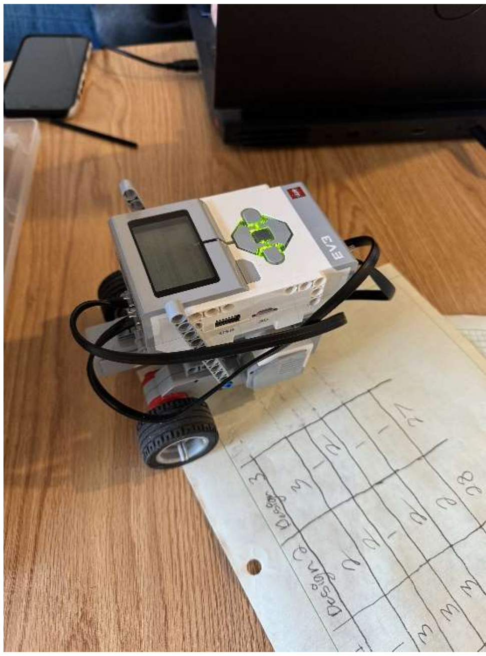
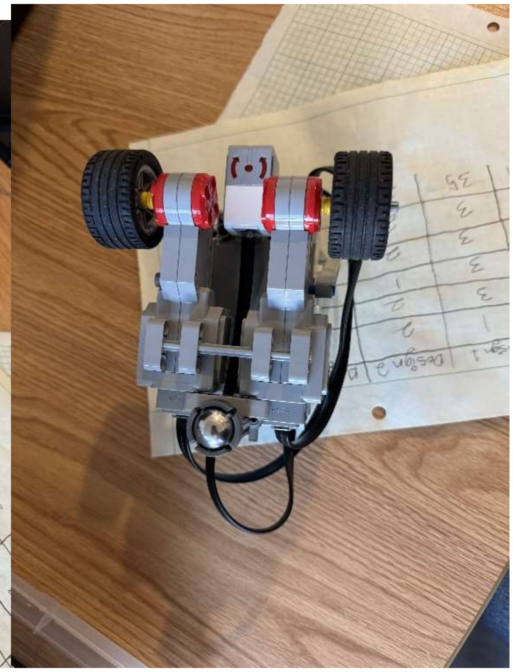
Iteration 3



Iteration 3 improves on iteration 2 by positioning the marble in the center of the back, as well as adding additional support. We also added a gyro sensor to detect the angle the robot is turning at.

2/13/24

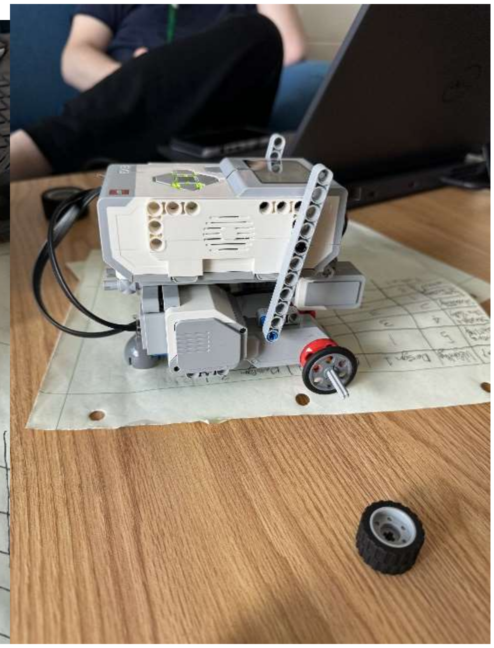
Iteration 4



Iteration 4 is slightly different from Iteration 3 in that the wheels are smaller. We found that smaller wheels allow the robot to more accurately follow a straight path.

2/14/24

Iteration 5



We were still struggling with the robot going off center, so we used even smaller wheels in iteration 5. This helped us stay in a straight line even more.

2/15/24

Subtask 1 Part 1 Testing

Lengths	Laps	Travel Distance (Y) (cm)	Travel Distance (X) (cm)	Movement
1	.5	60	0	Forward
2	1	61	1	Reverse
3	1.5	59	1	Forward
4	2	60	2	Reverse
5	2.5	61	2	Forward
6	3	60	2	Reverse
7	3.5	61	3	Forward
8	4	60	3	Reverse

Lengths	Laps	Travel Distance (Y) (cm)	Travel Distance (X) (cm)	Movement
1	.5	102	0	Forward
2	1	100	1	Reverse
3	1.5	98	1	Forward
4	2	99	2	Reverse
5	2.5	100	3	Forward
6	3	101	3	Reverse
7	3.5	99	3	Forward
8	4	101	4	Reverse

Lengths	Laps	Travel Distance (Y) (cm)	Travel Distance (X) (cm)	Movement
1	.5	203	0	Forward
2	1	197	1	Reverse
3	1.5	199	2	Forward
4	2	201	3	Reverse
5	2.5	200	4	Forward
6	3	202	4	Reverse
7	3.5	198	5	Forward
8	4	200	6	Reverse

2/15/24

Subtask 1 Part 2 Testing

Lengths	Laps	Travel Distance (Y) (cm)	Travel Distance (X) (cm)	Degrees Turned	Movement
1	.5	62	0		Forward
				175	Turn 180 degrees
2	1	59	1		Reverse
				180	Turn 180 degrees
3	1.5	50	1		Forward
				180	Turn 180 degrees
4	2	60	2		Reverse
				90	Turn 180 degrees
5	2.5	2	59		Forward
				180	Turn 180 degrees
6	3	3	60		Reverse
				185	Turn 180 degrees
7	3.5	3	61		Forward
				180	Turn 180 degrees
8	4	3	62		Reverse

Lengths	Laps	Travel Distance (Y) (cm)	Travel Distance (X) (cm)	Degrees Turned	Movement
1	.5	100	0		Forward
				185	Turn 180 degrees
2	1	102	3		Reverse
				180	Turn 180 degrees
3	1.5	101	4		Forward
				180	Turn 180 degrees

4	2	98	4		Reverse
				185	Turn 180 degrees
5	2.5	97	5		Forward
				180	Turn 180 degrees
6	3	100	5		Reverse
				180	Turn 180 degrees
7	3.5	102	6		Forward
				183	Turn 180 degrees
8	4	103	6		Reverse

Lengths	Laps	Travel Distance (Y) (cm)	Travel Distance (X) (cm)	Degrees Turned	Movement
1	.5	202	0		Forward
				185	Turn 180 degrees
2	1	200	2		Reverse
				185	Turn 180 degrees
3	1.5	201	5		Forward
				180	Turn 180 degrees
4	2	198	4		Reverse
				175	Turn 180 degrees
5	2.5	197	6		Forward
				177	Turn 180 degrees
6	3	203	8		Reverse
				182	Turn 180 degrees
7	3.5	205	8		Forward
				185	Turn 180 degrees
8	4	203	9		Reverse

Code for SubTask1:

```
#!/usr/bin/env pybricks-micropython
from pybricks.hubs import EV3Brick
from pybricks.ev3devices import (Motor, TouchSensor, ColorSensor,
                                 InfraredSensor, UltrasonicSensor, GyroSensor)
from pybricks.parameters import Port, Stop, Direction, Button, Color
from pybricks.tools import wait, StopWatch, DataLog
from pybricks.robotics import DriveBase
from pybricks.media.ev3dev import SoundFile, ImageFile
import time

ev3 = EV3Brick()
rightMotor = Motor(Port.A)
leftMotor = Motor(Port.B)
gyro = GyroSensor(Port.S1)
Left = "left"
Right = "right"
```

```
def moveForward(speed, timeToRotate):
    rightMotor.run(speed)
    leftMotor.run(speed)

    time.sleep(timeToRotate)

    rightMotor.brake()
    leftMotor.brake()
```

```python
def rotate(speed, degree, direction):
    gyro.reset_angle(0)

    speed /= 10
    degree -= 20

    if (direction == Left):
        while (gyro.angle() < degree):
            rightMotor.run(speed)
            leftMotor.run(-speed)

    elif (direction == Right):
        while (gyro.angle() > -degree):
            rightMotor.run(-speed)
            leftMotor.run(speed)

    rightMotor.hold()
    leftMotor.hold()

```

```python
def angleAdjust(speed, maxAngle):

    if (gyro.angle() < -maxAngle):
        rotate(speed / 5, abs(gyro.angle()), Left)

    elif (gyro.angle() > maxAngle):
        rotate(speed / 5, abs(gyro.angle()), Right)

```

```
def Part1(speed, distance, laps):
    badAngle = 45

    TIME = distance / 20.32
    for i in range(laps):
        gyro.reset_angle(0)

        moveForward(speed, TIME)
        time.sleep(0.5)

        # angleAdjust(speed, badAngle)
        # time.sleep(0.5)

        moveForward(-speed, TIME)
        time.sleep(0.5)

        # angleAdjust(speed, badAngle)
        # time.sleep(0.5)
```

```
def Part2(speed, distance, laps):
    badAngle = 15

    TIME = distance / 20.32
    gyro.reset_angle(0)
    for i in range(2 * laps):

        moveForward(speed, TIME)
        time.sleep(0.5)

        # angleAdjust(speed, badAngle)
        # time.sleep(0.5)

        rotate(speed, 180, Left)
        time.sleep(0.5)
```

2/15/24

◆ **Meeting Agenda / Minutes** **Project/Team: ENED 1120 Robot Team 238**

Date/Time: 2/13/24, 12:30pm-4:30pm Note Taker: Luke Weis

Present: Luke Weis, Alex Funk, AJ Armstrong, Arpit Gill

Agenda:

- Determine robot design for subtask 1
- Create prototype of robot
- Create code basis for robot

Review Gantt Chart

Tasks behind schedule: None

☐ Tasks added to chart: None

Foreseeable issues that will prevent success ☐ No ☐ Yes

Decisions made:

- The robot will have 2 front wheels and a marble as the rear wheel to ensure it is able to turn accurately
- The gyroscopic sensor will be used to determine how far the robot should turn

Important information shared:

- Next meeting will be held tomorrow to create test cases
- Determined subtask 1 requires us to have the robot go in a straight line and then reverse, and go in a straight line and then turn around and go back to the starting point

Ideas we want to remember:

Team member	Hours on project since last meeting (hrs)	Total hours on project (hrs)
Luke Weis	0	6
AJ Armstrong	0	6
Alex Funk	0	6
Arpit Gill	0	6

Meeting Agenda / Minutes **Project/Team: ENED 1120 Robot Team 238**

Date/Time: 2/14/24, 3:30pm-5:30pm Note Taker: Luke Weis

Present: Luke Weis, Alex Funk, AJ Armstrong, Arpit Gill

Agenda:

- Test robot
- Document tests of robot

Review Gantt Chart

Tasks behind schedule: None

☐ Tasks added to chart: None

Foreseeable issues that will prevent success ☐ No ☐ Yes

- The robot sometimes does not stop at the right time and gets off center

Decisions made:

Important information shared:

- Subtask 1 is on Monday, 2/14 and we are waiting to find out the time

Ideas we want to remember:

Team member	Hours on project since last meeting (hrs)	Total hours on project (hrs)
Luke Weis	0	8
AJ Armstrong	0	8
Alex Funk	0	8
Arpit Gill	0	8

Meeting Agenda / Minutes **Project/Team: ENED 1120 Robot Team 238**

Date/Time: 2/15/24, 2:30pm-6:30pm Note Taker: Luke Weis

Present: Luke Weis, Alex Funk, AJ Armstrong, Arpit Gill

Agenda:

- Complete tests of robot
- Document tests of robot

Review Gantt Chart

Tasks behind schedule: None

☐ Tasks added to chart: None

Foreseeable issues that will prevent success ☐ No ☐ Yes

- The robot sometimes does not stop at the right time and gets off center

Decisions made:

- Have the code try to check when the robot gets off center and correct itself
- Robot will have very small wheels to increase accuracy
- Calculated how long to run motors to go x amount of centimeters

Important information shared:

- Subtask 1 is on Monday, 2/19 at 11:40am

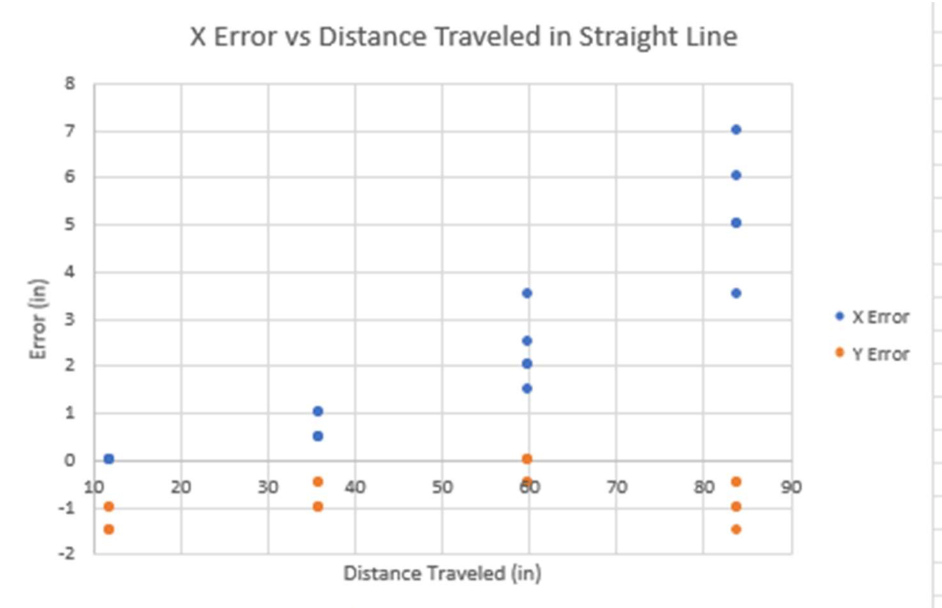
Ideas we want to remember:

Team member	Hours on project since last meeting (hrs)	Total hours on project (hrs)
Luke Weis	0	12
AJ Armstrong	0	12
Alex Funk	0	12
Arpit Gill	0	12

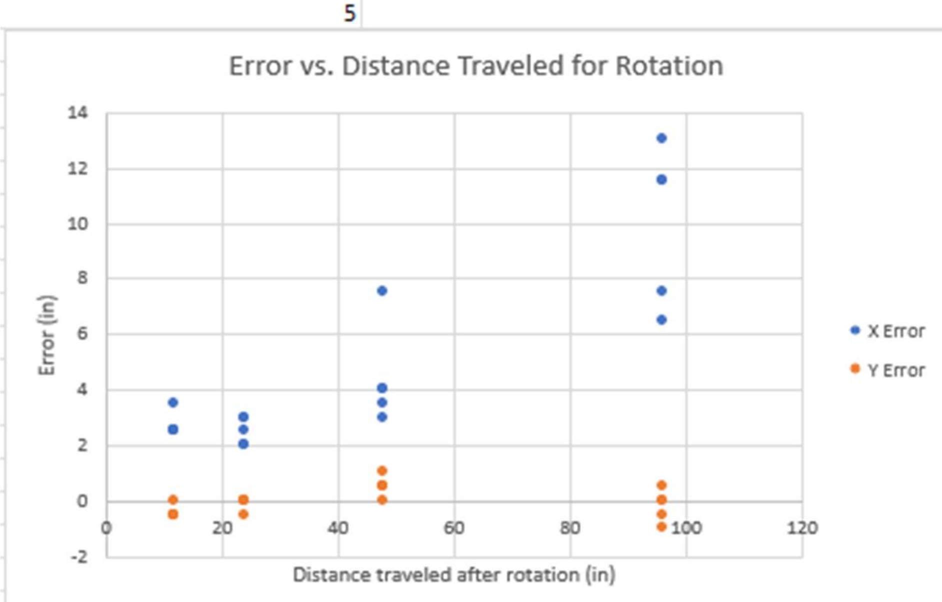
Mid Project Review:

Test Plan Data

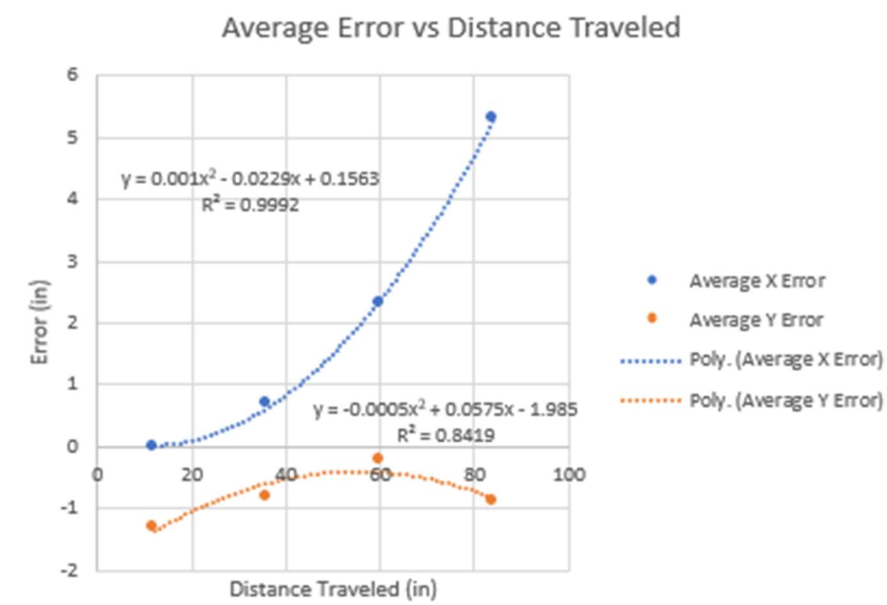
Test	Distance away from expected - x (in)	Distance away from expected - y (in)				
1	0	-1				
1	0	-1.5				
1	0	-1.5				
1	0	-1.5	X average for 12 in	Y average for 12 in	X STD	Y STD
1	0	-1	0	-1.3	0	0.27386
2	0.5	-1				
2	1	-1				
2	1	-0.5				
2	0.5	-0.5	X average for 36 in	Y average for 36 in	X STD	Y STD
2	0.5	-1	0.7	-0.8	0.27386	0.27386
3	2	0				
3	1.5	0				
3	2	-0.5				
3	3.5	-0.5	X average for 60 in	Y average for 60 in	X STD	Y STD
3	2.5	0	2.3	-0.2	0.75829	0.27386
4	3.5	-0.5				
4	7	-1.5				
4	5	-1				
4	5	-1	X average for 84 in	Y average for 84 in	X STD	Y STD
4	6	-0.5	5.3	-0.9	1.30384	0.41833
5	3.5	0				
5	2.5	-0.5				
5	2.5	-0.5				
5	2.5	-0.5	X average for 12 in	Y average for 12 in	X STD	Y STD
5	2.5	-0.5	2.7	-0.4	0.44721	0.22361
6	3	0				
6	2	0				
6	2	-0.5				
6	3	0	X average for 24 in	Y average for 24 in	X STD	Y STD
6	2.5	0	2.5	-0.1	0.5	0.22361
7	4	0.5				
7	4	0.5				
7	7.5	0				
7	3.5	0.5	X average for 48 in	Y average for 48 in	X STD	Y STD
7	3	1	4.4	0.5	1.78185	0.35355
8	6.5	0.5				
8	11.5	-1				
8	13	-0.5				
8	11.5	0	X average for 96 in	Y average for 96 in	X STD	Y STD
8	7.5	0	10	-0.2	2.82843	0.57009


X Error vs Distance Traveled in Straight Line
Error (in)
Distance Traveled (in)
X Error
Y Error

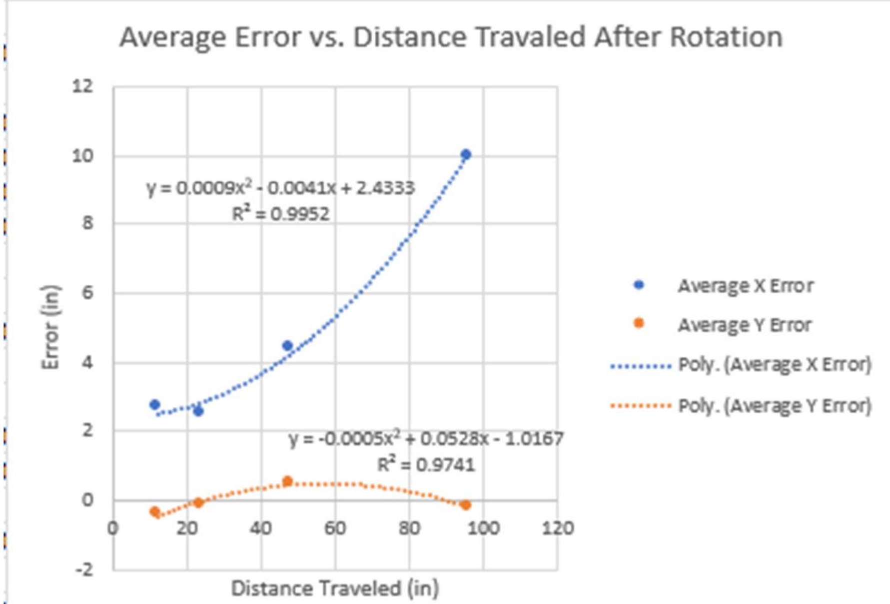
5


Error vs. Distance Traveled for Rotation
Error (in)
Distance traveled after rotation (in)
X Error
Y Error

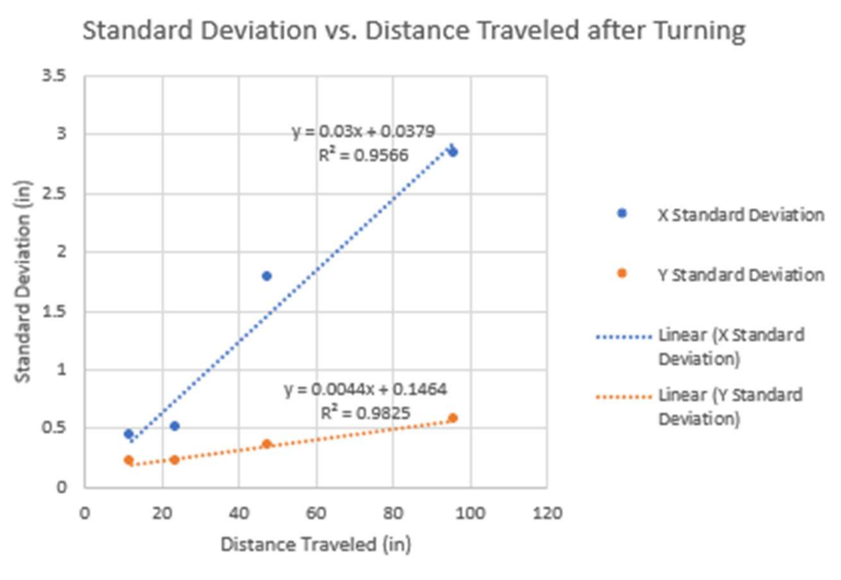
Luke Weis, Arpit Gill, AJ Armstrong, Alex Funk, 28

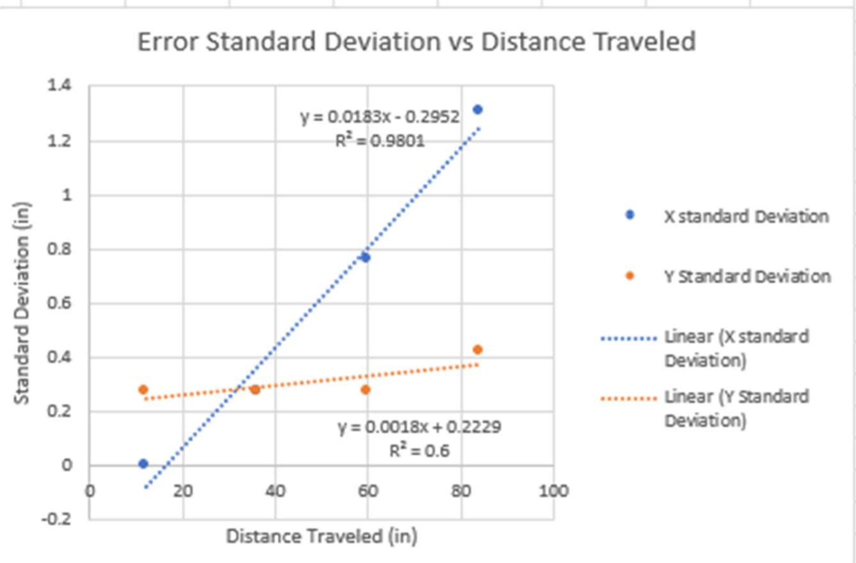


-1



3/5/24


Standard Deviation vs. Distance Traveled after Turning
y = 0.03x + 0.0379
R² = 0.9566
y = 0.0044x + 0.1464
R² = 0.9825
Standard Deviation (in)
Distance Traveled (in)
X Standard Deviation
Y Standard Deviation
Linear (X Standard Deviation)
Linear (Y Standard Deviation)


Error Standard Deviation vs Distance Traveled
y = 0.0183x - 0.2952
R² = 0.9801
y = 0.0018x + 0.2229
R² = 0.6
Standard Deviation (in)
Distance Traveled (in)
X standard Deviation
Y Standard Deviation
Linear (X standard Deviation)
Linear (Y Standard Deviation)

3/5/24

Code for our model:

```python
def targetDistance(forward, right, turns):
    #Assumptions: starting at 0,0; All units are in inches;
    #Robot goes forward some distance (y), turns 90 degrees, goes forward some distance (x);
    errorForwardX = 0
    errorForwardY = 0
    errorRightX = 0
    errorRightY = 0
    #Adding errror for each turn based on robots tendency to under/overrotate
    if turns > 0:
        for i in range(turns):
            errorForwardX += .03*forward
            errorForwardY += .02*forward
            errorRightX += .03*right
            errorRightY += .02*right
    #Straight line before turning
    #Model error for X: f(x) = .001x^2 - .0229x + .1563
    #Model error for Y: f(x) = -.0005x^2 + .0575x - 1.985
    errorForwardX += .001*(forward)**2 - .0229*(forward) + .1563
    errorForwardY += -.0005*(forward)**2 + .0575*(forward) - 1.985
    #Straight line after turning
    #Model error for X: f(x) = .0009x^2 - .0041x + 2.4333
    #Model error for Y: f(x) = -.005x^2 + .0528x - 1.90167
    errorRightX += .0009*(right)**2 - .0041*(right) + 2.4333
    errorRightY += -.0005*(right)**2 - .0528*(right) - 1.90167
    finalPositionX = errorForwardX + errorRightX
    finalPositionY = errorForwardY + errorRightY
    print(forward + finalPositionX, right + finalPositionY)
```

3/5/24

Meeting Agenda / Minutes **Project/Team: ENED 1120 Robot Team 238**

Date/Time: 3/5/24, 2:30pm-6:30pm Note Taker: Arpit Gill

Present: Luke Weis, Alex Funk, AJ Armstrong, Arpit Gill

Agenda:

- Mid Project Review
- Discussed potential CCCD assignments

Review Gantt Chart

Tasks behind schedule: None

☐ Tasks added to chart: None

Foreseeable issues that will prevent success ☐ No ☐ Yes

- There is still some error present within the robot, making the robot sometimes not able to go straight over long distances.

Decisions made:

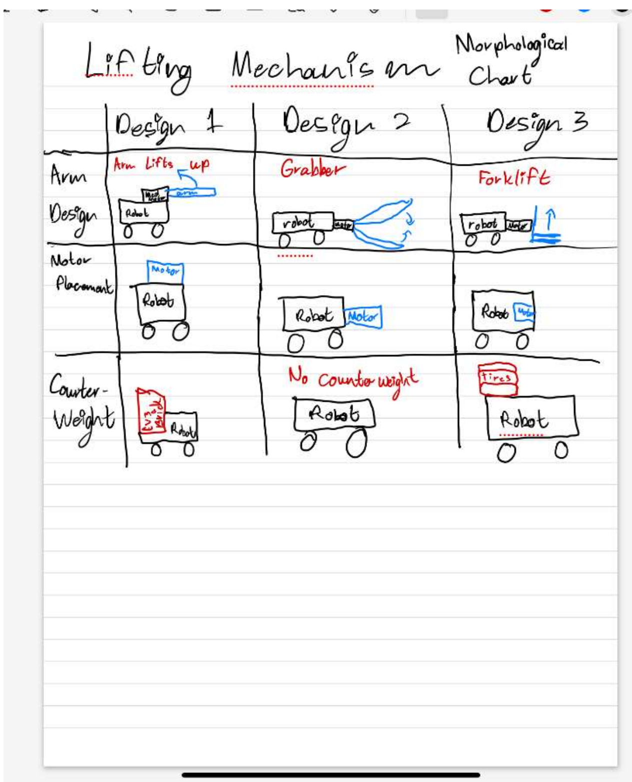
- Later we will use the error.py file made to calculate a more accurate final position because of the error present with the robot

Important information shared:

- We will likely listen to the podcast for the CCCD assignment

Ideas we want to remember:

Team member	Hours on project since last meeting (hrs)	Total hours on project (hrs)
Luke Weis	0	16
AJ Armstrong	0	16
Alex Funk	0	16
Arpit Gill	0	16

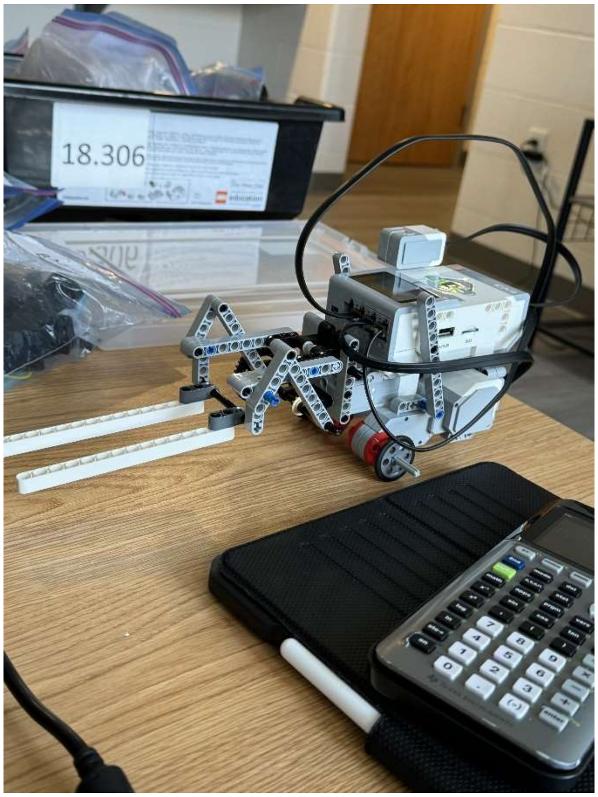


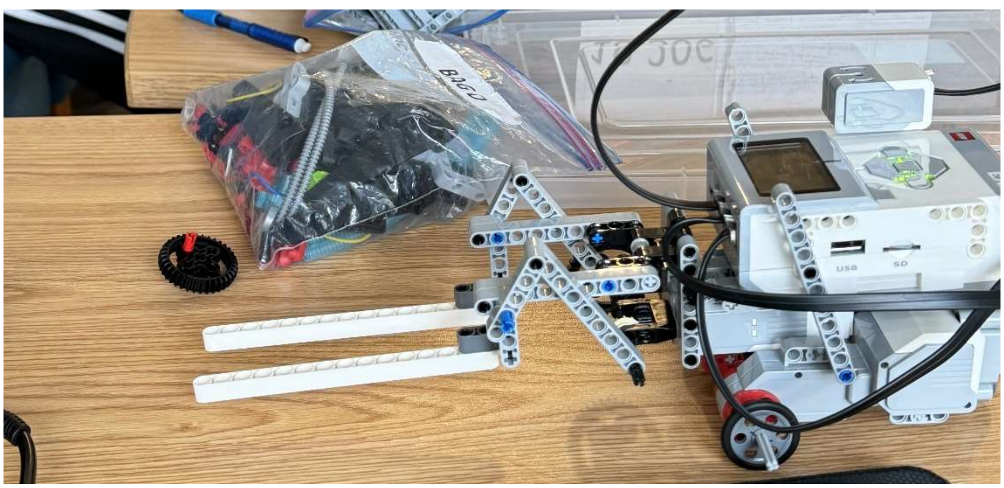
Lifting Mechanism Design Matrix

Category	Weighting	Design 1	Design 2	Design 3
Robot Stability	5	3	1	2
Strength	4	2	1	3
Package Stability	3	1	2	3
Ease of Design	1	3	2	1
Interoperability With other Functions	2	2	1	3
Total		33	19	38

We decided to go with Design 3 of the morphological chart for the lifting mechanism as it scored the highest in the design matrix.

3/26/24





This is the first iteration of our arm design. The design was not successful as it was not able to carry much weight (less than 200 grams).

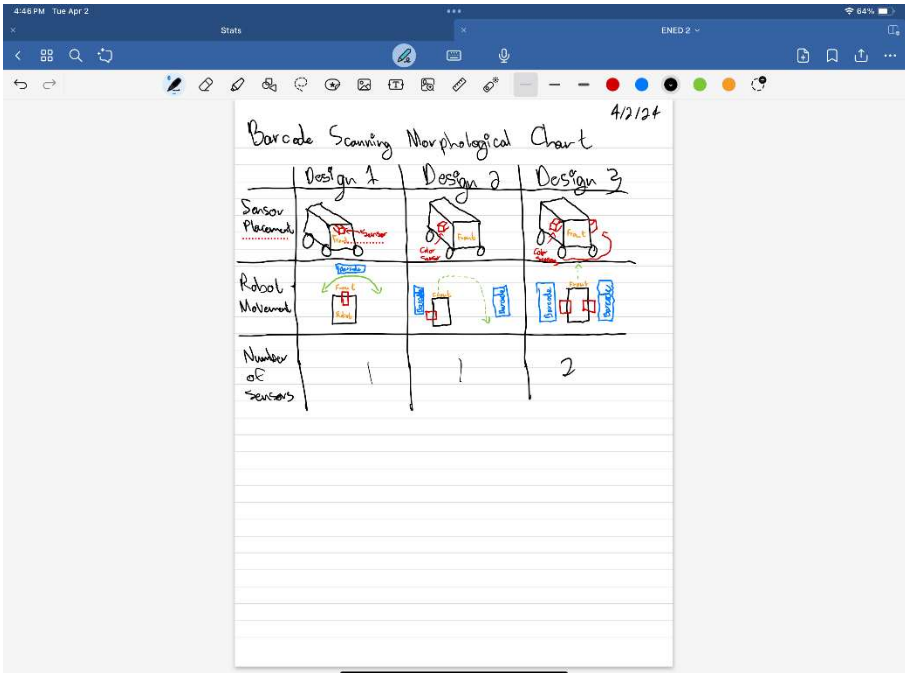
3/26/24

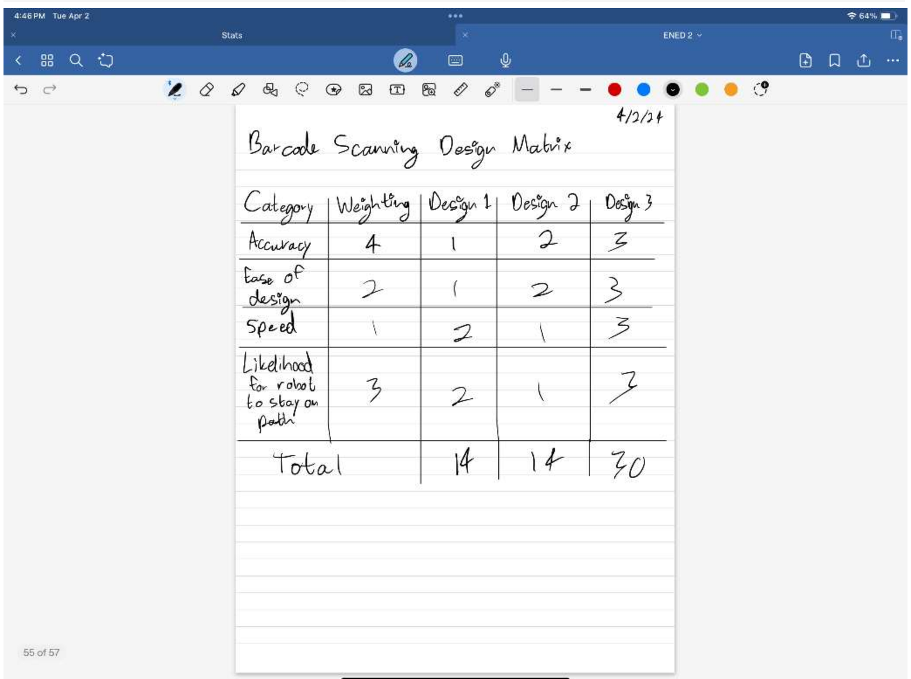
This is the 2nd iteration of our arm design. We moved the arm to the top of the robot as it provides more stability and allows the EV3 brick to act as more of a counterweight. The arm is also higher up, which allows it to grab the handle of the box. We used gear ratios to create a higher torque by having a small gear on the medium motor spinning a larger gear. The increased torque provided the arm more power.

3/26/24


This is our 3rd iteration of our arm design. We found that the 2nd iteration still did not have enough torque to lift 200 grams, so we increased the torque by having a smaller gear on the medium motor and a larger gear attached to it. The gear ratio is 1:4. We found that this design was able to lift 200 grams.

4/2/24




4/2/24

Barcode Scanning Design Matrix

Category	Weighting	Design 1	Design 2	Design 3
Accuracy	4	1	2	3
Ease of design	2	1	2	3
Speed	1	2	1	3
Likelihood for robot to stay on path	3	2	1	3
Total		14	14	30

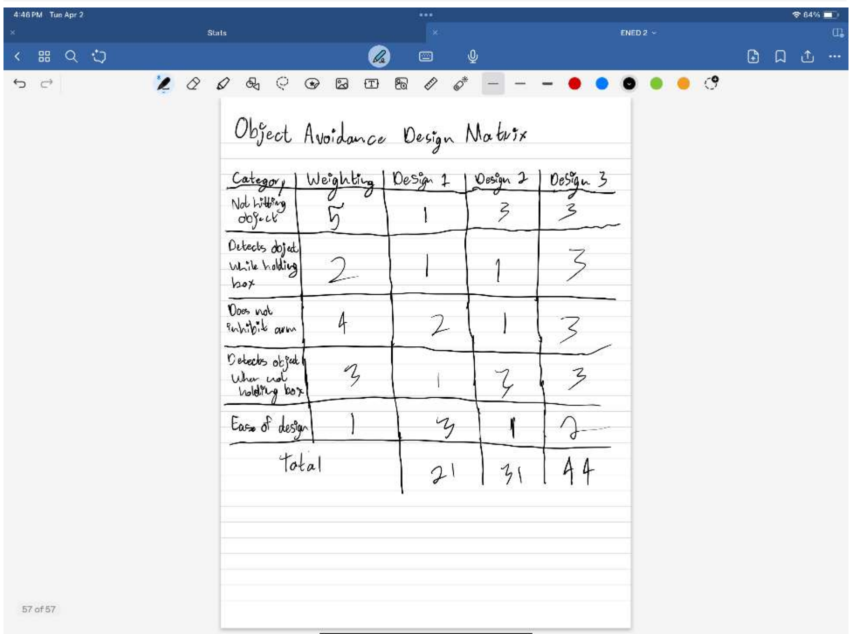
This is our morphological chart and design matrix for our barcode scanning system. We decided to go with design 3 as it scored the highest in the design matrix.

4/2/24

Luke Weis, Arpit Gill, AJ Armstrong, Alex Funk, 37

Object Avoidance Morphological Chart

	Design 1	Design 2	Design 3
Sensor Placement	Sensor Front	arm Sensor Front	Front Sensor
Sensor type	Touch	Ultrasonic	Ultrasonic
Logic	Stop if hits something	Stop if detects 10cm in front	stop if detects 10 cm in front

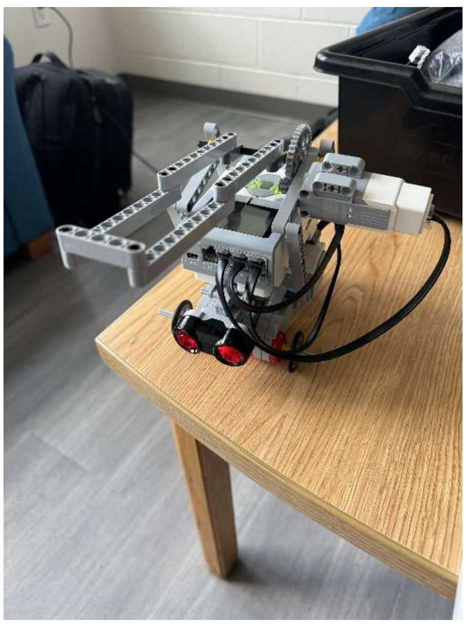


Object Avoidance Design Matrix

Category	Weighting	Design 1	Design 2	Design 3
Not hitting object	5	1	3	3
Detects object while holding box	2	1	1	3
Does not inhibit arm	4	2	1	3
Detects object when not holding box	3	1	3	3
Ease of design	1	3	1	2
Total		21	31	44

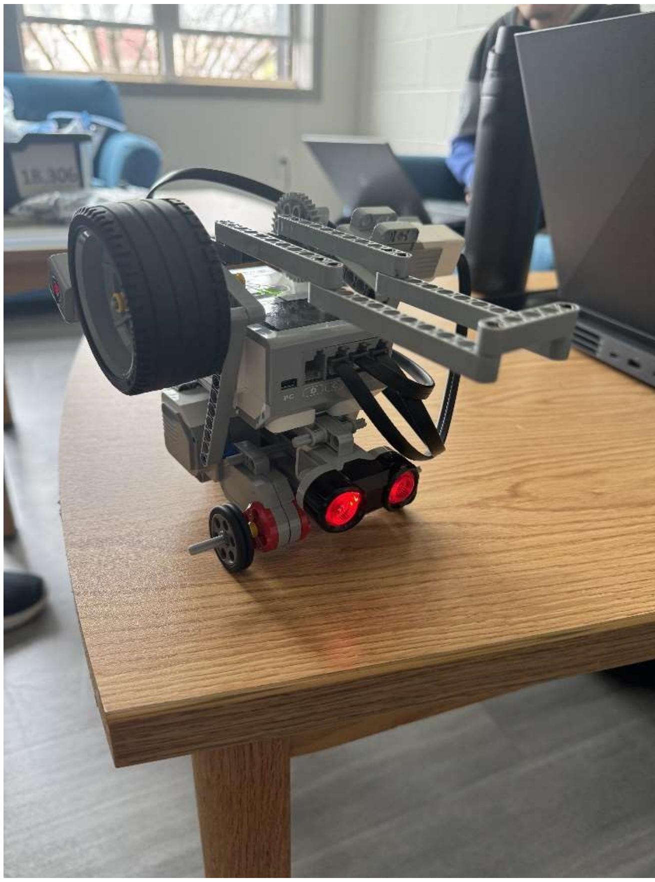
This is our Morphological chart and design matrix for our object avoidance system. We decided to go with design 3 as it scored the highest in the design matrix.

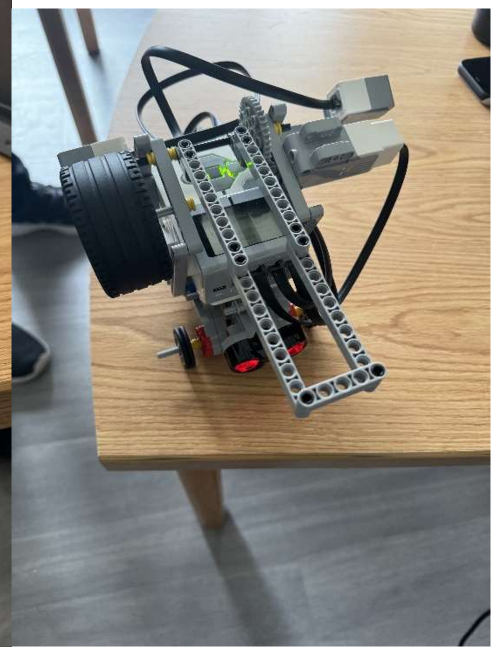
4/2/24



This is the next iteration of the robot. We added an ultrasonic sensor to the front of the robot to detect obstacles. Due to this placement, we had to move the gyroscopic sensor to the bottom of the robot.

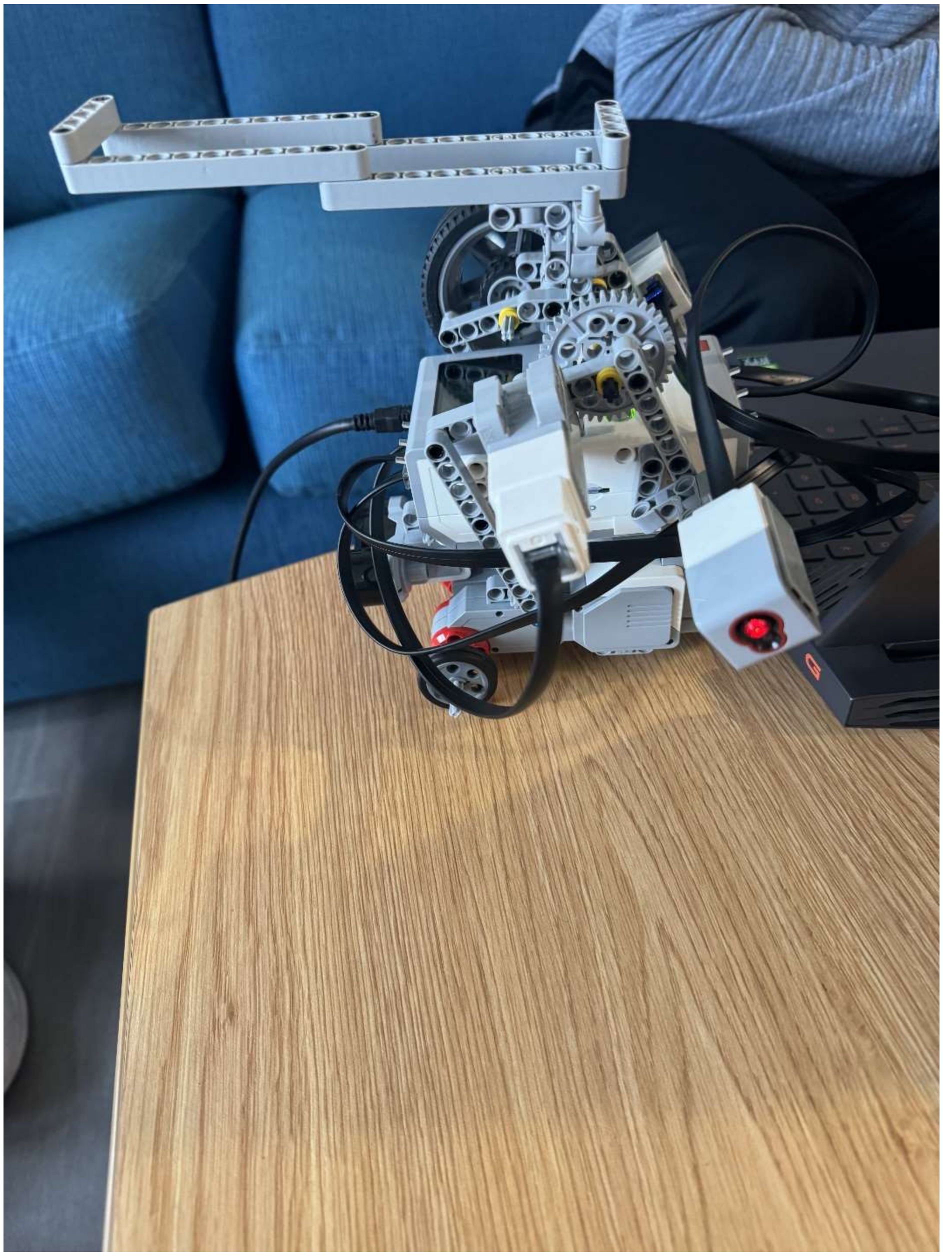
4/2/24





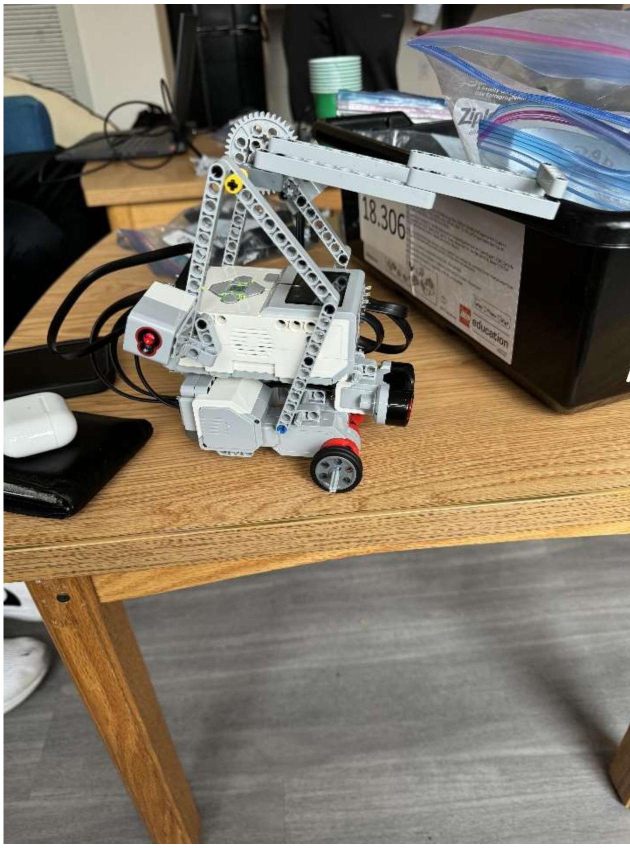
This is the next iteration of our robot's design. We added a large wheel to the left side of the robot to counteract the weight of the medium motor. We found that this allowed the robot to travel straighter. We also added color sensors on both sides to allow the robot to scan barcodes on either side of the isle.

4/3/24



This is the next iteration of our robot. We discovered that the arm was not high up enough to lift the box, so we had to move it upwards. This model faces issues with arm strength and stability, though.

4/9/24



This is the next iteration of our robot. We moved the motor up with the arm and increased the support of the arm and motor, which improved the stability of the arm and allowed it to lift up to 200 grams. We also removed the right color sensor as it was not being used and added unnecessary complexity.

4/9/24

This is the design of the box we used in our testing. The front of the box is 6 5/8 in tall, the width is 4 3/8 inches, and the height is 6 3/8 inches, and the middle of the barcode is placed 4.5 inches up the height of the box, which is the same as the specification for the box used in the final demo. We chose this box design as it was easy to create and obtain materials for our testing.

4/9/24

This is the final design of our robot. We found that when testing picking up the box, the arm on our previous design was not long enough to reach the handle. We extended the arm to be able to reach the handle.

4/9/24

EV3imports.py

```
#!/usr/bin/env pybricks-micropython
'''
This file is just the imports needed in every file to make the robot work.
Code provided by EV3
'''
from pybricks.hubs import EV3Brick
from pybricks.ev3devices import (Motor, TouchSensor, ColorSensor,
                                 InfraredSensor, UltrasonicSensor, GyroSensor)
from pybricks.parameters import Port, Stop, Direction, Button, Color
from pybricks.tools import wait, StopWatch, DataLog
from pybricks.robotics import DriveBase
from pybricks.media.ev3dev import SoundFile, ImageFile
```

This is the first part of our code. It imports the EV3 sensors and functions, which allows python to be compatible with our robot. This code did not need iteration.

4/3/24

getCoords.py

```python
#!/usr/bin/env pybricks-micropython
'''
Functions for the retrival of the coordinates of different items in the warehouse.

By Arpit Gill
'''

# Description of the function:
#   This function gets the coordinates for a specific shelving unit and box.
# Arguments:
#   shelf: The specific shelf needed as a string
#   number: The number of the box needed as an int.
# Return values:
#   x: The x coordinate of the box in inches.
#   y: The y coordinate of the box in inches.
#   direction: which side of the shelf the box is on (north or south).
def shelvingUnit(shelf, number):
    # Values are in inches
    x = 0
    y = 0
    direction = ""
    shelfdict = {
        "A1": [12, 12],
        "A2": [12, 36],
        "B1": [60, 12],
        "B2": [60, 36],
        "C1": [12, 60],
        "C2": [12, 84],
        "D1": [60, 60],
        "D2": [60, 84]
    }
    numberdict = {
        1: [3, 0],
        2: [9, 0],
        3: [15, 0],
        4: [21, 0],
        5: [27, 0],
        6: [33, 0],
        7: [3, 12],
        8: [9, 12],
        9: [15, 12],
        10: [21, 12],
        11: [27, 12],
        12: [33, 12]
    }
    x += shelfdict[shelf][0]
    y += shelfdict[shelf][1]
    x += numberdict[number][0]
    y += numberdict[number][1]
    if (number >= 7):
        direction = "North"
    else:
        direction = "South"
    return(x, y, direction)



# Description of the function:
#   This function gets the coordinates for a home base.
# Arguments:
#   letter: the letter of the home base as a string
# Return values:
#   x: The x coordinate of the base in inches.
#   y: The y coordinate of the base in inches.
def endArea(letter):
    # Values are in inches
    fulfilldict = {
        "A": [6, -6],
        "B": [102, -6],
        "C": [6, 114],
        "D": [102, 114],
    }
    x = fulfilldict[letter][0]
    y = fulfilldict[letter][1]
    return(x, y)
```

The function shelvingUnit is used to store and get the coordinates of where boxes are placed in the warehouse, which is used to determine where the robot should stop and pick up a box. The code uses a dictionary to store the data. This code did not require iteration. The function endArea was used to get the coordinates of the home base that the robot was supposed to stop at. The code uses a dictionary to store the data. This code did not require iteration.

4/3/24

movement.py

```
#!/usr/bin/env pybricks-micropython
'''
Functions for the movement of the robot and its different parts

By Alex Funk
'''
from EV3imports import *
import time

ev3 = EV3Brick()
rightMotor = Motor(Port.A)
leftMotor = Motor(Port.B)
arm = Motor(Port.C)
gyro = GyroSensor(Port.S1)
Left = "left"
Right = "right"


# Description of the function:
#   This function moves the robot forward at a specified speed and for a specified time.
# Arguments:
#   speed: speed of the robot in the rpm of the motors
#   timeToRotate: The time for the robot to move in seconds
def moveForward(speed, distance):
    distance /= 5


    rightMotor.run(speed+2)
    leftMotor.run(speed)

    time.sleep(distance)

    rightMotor.brake()
    leftMotor.brake()

    time.sleep(0.5)
```

This is the 1st part of our code for defining the movement functions of our robot. We found that to get the correct distance, we had to divide the inputted distance by 5. We had the robot move based on time. We also found that the right motor was slightly weaker than the left motor, which is why the right motor runs +2 faster. We had the robot brake at the end to make sure the motors stopped.

4/3/24

```
# Description of the function:
#   The function rotates the robot to a specified angle.
# Arguments:
#   speed: speed of the robot in the rpm of the motors
#   degree: The amount of degrees to rotate the robot.
#   direction: the direction of the robot: Left or Right
def rotate(speed, degree, direction):
    gyro.reset_angle(0)

    speed /= 10

    if (direction == Left):
        degree += 0
        while (gyro.angle() < degree):
            rightMotor.run(speed)


    elif (direction == Right):
        degree += 0
        while (gyro.angle() > -degree):

            leftMotor.run(speed)

    rightMotor.hold()
    leftMotor.hold()

    time.sleep(0.5)
```

This is the 1st iteration of our code to rotate the robot. We found that it did not turn quickly or precisely enough.

4/3/24

```
# Description of the function:
#   The function rotates the robot to a specified angle.
# Arguments:
#   speed: speed of the robot in the rpm of the motors
#   degree: The amount of degrees to rotate the robot.
#   direction: the direction of the robot: Left or Right
def rotate(speed, degree, direction):
    gyro.reset_angle(0)

    speed /= 10

    if (direction == Left):
        degree += 0
        while (gyro.angle() < degree):
            rightMotor.run(speed)
            leftMotor.run(-speed)

    elif (direction == Right):
        degree += 0
        while (gyro.angle() > -degree):
            rightMotor.run(-speed)
            leftMotor.run(speed)

    rightMotor.hold()
    leftMotor.hold()

    time.sleep(0.5)
```

This is the final iteration of our code to rotate the robot. We found that by running both motors in opposite directions, the robot was able to turn quicker and more accurately.

4/3/24

```
# Description of the function:
#   This function moves the arm of the robot to a specific angle.  Negative is down and positive is up.
# Arguments:
#   speed: speed of the robot in the rpm of the motors.
#   angle: the desired angle to rotate the arm to in degrees.
#   gearRatio: The ratio between the first and last gears in the train.  Used alter the amount of rotation.
def moveArm(speed, angle):
    arm.reset_angle(0)
        if angle < 0:
        while arm.angle() > angle:
            arm.run(-speed)
    else:
        while arm.angle() < angle:
            arm.run(speed)

    arm.hold()
    time.sleep(0.5)
```

This is the first iteration of our code to move the arm of the robot. It worked at first, but we found that as we modified the sizes of the gears and therefore the gear ratio, the code would get off.

4/4/24

```
# Description of the function:
#   This function moves the arm of the robot to a specific angle.  Negative is down and positive is up.
# Arguments:
#   speed: speed of the robot in the rpm of the motors.
#   angle: the desired angle to rotate the arm to in degrees.
#   gearRatio: The ratio between the first and last gears in the train.  Used alter the amount of rotation.
def moveArm(speed, angle, gearRatio = 1):
    arm.reset_angle(0)
    angle *= gearRatio
    if angle < 0:
        while arm.angle() > angle:
            arm.run(-speed)
    else:
        while arm.angle() < angle:
            arm.run(speed)

    arm.hold()
    time.sleep(0.5)
```

This is the final iteration of our arm movement code. We added a gear ratio multiplier, which made arm movements far more accurate.

4/4/24

main.py

```
#!/usr/bin/env pybricks-micropython
'''
ENED 1120 Robot Code Spring 2024
Team 238
Section 014

Alex Funk, Arpit Gill, Anthony Armstrong, and Luke Weis
'''
# Import libraries
from EV3imports import *
import movement
import getCoords
import time
import multiprocessing as mp

# declare ev3 brick
ev3 = EV3Brick()

# declare motors
rightMotor = Motor(Port.A)
leftMotor = Motor(Port.B)
arm = Motor(Port.C)

# declare sensors
gyro = GyroSensor(Port.S1)
ultraSonic = UltrasonicSensor(Port.S2)
rightColor = ColorSensor(Port.S4)

# lazy variables to make typing easier
Left = "left"
Right = "right"
```

This is the first part of our main function. It imports relevant functions and declares global variables to be used throughout the code.
4/4/24

```
def finalDemo1(speed, shelf, box):
    x, y, direction = getCoords.shelvingUnit(SHELF, box)
    xf, yf = getCoords.endArea("B")

    movement.moveForward(speed, y)
    movement.rotate(speed, 90, Right)
    movement.moveForward(speed, x)
```

This is the first iteration of our final demo subtask 1 code. This code is designed to get the robot to the inputted box and stop once it gets there. We found that this code had issues where the robot would not travel far enough in the x – direction and too far in the y – direction.

4/4/24

```
def finalDemo1(speed, shelf, box):
    x, y, direction = getCoords.shelvingUnit(SHELF, box)
    xf, yf = getCoords.endArea("B")

    movement.moveForward(speed, y+6)
    movement.rotate(speed, 90, Right)
    movement.moveForward(speed, x-3)
```

This is the 2nd iteration of our subtask 1 code. We found that it was still too short in the y direction and too far in the x direction.

4/4/24

```
def finalDemo1(speed, shelf, box):
    x, y, direction = getCoords.shelvingUnit(SHELF, box)
    xf, yf = getCoords.endArea("B")

    movement.moveForward(speed, y+12)
    movement.rotate(speed, 90, Right)
    movement.moveForward(speed, x-6)
```

This is the 3rd iteration of our subtask 1 code. This code traveled the correct distance in both directions and stopped at the box.

4/4/24

```
def finalDemo1(speed, shelf, box):
    x, y, direction = getCoords.shelvingUnit(SHELF, box)
    xf, yf = getCoords.endArea("B")

    movement.moveForward(speed, y+12)
    movement.rotate(speed, 90, Right)
    movement.moveForward(speed, x-6)
    time.sleep(5)
    movement.moveForward(speed, (xf - 6) - (x - 6))
    movement.rotate(speed, 90, Right)
    movement.moveForward(speed, y + 12)
```

This is the final iteration of our subtask 1 code. It is able to travel to the box, stop, and head to the base.

4/4/24

```
def finalDemo2(speed):
    movement.rotate(speed, 180, Left)
    movement.moveForward(speed, 12)
```

This is the first iteration of our subtask 2 code. It was able to turn the robot 180 degrees and move it forward 12 inches.

4/5/24

```
def finalDemo2(speed):
    movement.rotate(speed, 180, Left)
    movement.moveForward(speed, 12)
    movement.rotate(speed, 90, Left)
    movement.moveForward(speed, 96)
    movement.rotate(speed, 90, Left)
    movement.moveForward(speed, 12)
```

This is the final iteration of our subtask 2 code. It can turn the robot 180 degrees, travel 12 inches, turn the robot 90 degrees, travel 96 inches, turn the robot another 90 degrees, and travel 12 inches to the home base.
4/5/24

```
def finalDemo3(speed, armSpeed):
    movement.moveForward(speed, 20)

    black = Color.BLACK
    white = Color.WHITE
    realBarCodes = [[white, white, white, black],
                    [white, black, white, black],
                    [white, white, black, black],
                    [black, white, white, black]]
    ourBarCode = [white, black, white, black]
    readBarCode = []
    for i in range(4):
        readBarCode.append(rightColor.color())
        ev3.speaker.beep()
        movement.moveForward(speed, 1)
        time.sleep(2)

    if readBarCode == ourBarCode:
        ev3.screen.draw_text(0,0, "Barcodes are")
        ev3.screen.draw_text(0,20, "the same")
    else:
        ev3.screen.draw_text(0,0, "Barcodes are")
        ev3.screen.draw_text(0,20, "not the same")

    time.sleep(3)
```

This is the first iteration of our code for subtask 3. It was able to scan the barcode, but we found that it was not going the correct distance to reach the barcode.

4/9/24

```
def finalDemo3(speed, armSpeed):
    movement.moveForward(speed, 21)

    black = Color.BLACK
    white = Color.WHITE
    realBarCodes = [[white, white, white, black],
                    [white, black, white, black],
                    [white, white, black, black],
                    [black, white, white, black]]
    ourBarCode = [white, black, white, black]
    readBarCode = []
    for i in range(4):
        readBarCode.append(rightColor.color())
        ev3.speaker.beep()
        movement.moveForward(speed, 1.2)
        time.sleep(2)

    if readBarCode == ourBarCode:
        ev3.screen.draw_text(0,0, "Barcodes are")
        ev3.screen.draw_text(0,20, "the same")
    else:
        ev3.screen.draw_text(0,0, "Barcodes are")
        ev3.screen.draw_text(0,20, "not the same")

    time.sleep(3)
```

This is the 2nd iteration of our subtask 3 code. We found that this version traveled too far to get to the box and when reading the barcode.

4/9/24

```
def finalDemo3(speed, armSpeed):
    movement.moveForward(speed, 20.5)

    black = Color.BLACK
    white = Color.WHITE
    realBarCodes = [[white, white, white, black],
                    [white, black, white, black],
                    [white, white, black, black],
                    [black, white, white, black]]
    ourBarCode = [white, black, white, black]
    readBarCode = []
    for i in range(4):
        readBarCode.append(rightColor.color())
        ev3.speaker.beep()
        movement.moveForward(speed, 1.1)
        time.sleep(2)

    if readBarCode == ourBarCode:
        ev3.screen.draw_text(0,0, "Barcodes are")
        ev3.screen.draw_text(0,20, "the same")
    else:
        ev3.screen.draw_text(0,0, "Barcodes are")
        ev3.screen.draw_text(0,20, "not the same")

    time.sleep(3)
```

This is the final iteration of our subtask 3 code. This code is able to travel the correct distance to the box and correctly read the barcode.

4/9/24

```
def finalDemo4(speed, armSpeed):

    movement.rotate(speed, 90, Right)

    movement.moveArm(armSpeed, -90, 5)

    movement.moveForward(speed, 4)

    movement.moveArm(armSpeed, 45, 5)

    movement.moveForward(-speed, 4)

    movement.rotate(speed, 90, Left)

    movement.moveForward(speed, 24)

    movement.moveArm(armSpeed, -45, 5)
```

This is the first iteration of our subtask 4 code. We found that the robot would hit the box while rotating.

4/10/24

```
def finalDemo4(speed, armSpeed):
    movement.moveForward(-speed, 5)
    movement.rotate(speed, 90, Right)

    movement.moveArm(armSpeed, -90, 5)

    movement.moveForward(speed, 4)

    movement.moveArm(armSpeed, 45, 5)

    movement.moveForward(-speed, 4)

    movement.rotate(speed, 90, Left)

    movement.moveForward(speed, 24)

    movement.moveArm(armSpeed, -45, 5)
```

This is the second iteration of our subtask 4 code. We added code to have the robot move backwards before turning. We found that the robot did not travel far back enough to pick up the box.

4/10/24

```
def finalDemo4(speed, armSpeed):
    movement.moveForward(-speed, 6.3)
    movement.rotate(speed, 90, Right)

    movement.moveArm(armSpeed, -90, 5)

    movement.moveForward(speed, 4)

    movement.moveArm(armSpeed, 45, 5)

    movement.moveForward(-speed, 4)

    movement.rotate(speed, 90, Left)

    movement.moveForward(speed, 24)

    movement.moveArm(armSpeed, -45, 5)
```

This is the 3rd iteration of our subtask 4 code. We changed the distance that the robot back up before picking up the box. We found that this worked but the robot travelled too far forward after turning and hit the box.

4/10/24

```python
def finalDemo4(speed, armSpeed):
    movement.moveForward(-speed, 6.3)
    movement.rotate(speed, 90, Right)

    movement.moveArm(armSpeed, -90, 5)

    movement.moveForward(speed, 2.5)

    movement.moveArm(armSpeed, 45, 5)

    movement.moveForward(-speed, 2.5)

    movement.rotate(speed, 90, Left)

    movement.moveForward(speed, 24)

    movement.moveArm(armSpeed, -45, 5)
```

This is the final iteration of our subtask 4 code. It is able to move the robot backwards, rotate it, have it pick up the box, travel to the destination, and drop off the box.

4/10/24

```
ROBOTSPEED = 500
ARMSPEED = 200
SHELF = "A1"
BOX = 7


if __name__ == '__main__':
    ev3.speaker.set_speech_options(voice = "m3")
    ev3.speaker.set_volume(100, which = "_all_")
    ev3.speaker.say("Starting")

    # finalDemo1(ROBOTSPEED, SHELF, BOX)
    # time.sleep(3)
    finalDemo2(ROBOTSPEED)

    # finalDemo3(ROBOTSPEED, ARMSPEED)
```

This is how we defined our robot's variables and how we called the functions.

4/10/24

main.py

```
#!/usr/bin/env pybricks-micropython
'''
ENED 1120 Robot Code Spring 2024
Team 238
Section 014

Alex Funk, Arpit Gill, Anthony Armstrong, and Luke Weis
'''
# Import libraries
from EV3imports import *
import movement
import getCoords
import time
import multiprocessing as mp

# declare ev3 brick
ev3 = EV3Brick()

# declare motors
rightMotor = Motor(Port.A)
leftMotor = Motor(Port.B)
arm = Motor(Port.C)

# declare sensors
gyro = GyroSensor(Port.S1)
ultraSonic = UltrasonicSensor(Port.S2)
rightColor = ColorSensor(Port.S4)

# lazy variables to make typing easier
Left = "left"
Right = "right"

def finalDemo1(speed, shelf, box):
    x, y, direction = getCoords.shelvingUnit(SHELF, box)
    xf, yf = getCoords.endArea("B")

    movement.moveForward(speed, y + 12)
    movement.rotate(speed, 90, Right)
    movement.moveForward(speed, x - 6)
    time.sleep(5)
    movement.moveForward(speed, (xf - 6) - (x - 6))
    movement.rotate(speed, 90, Right)
    movement.moveForward(speed, y + 12)

def finalDemo2(speed):
    movement.rotate(speed, 180, Left)
    movement.moveForward(speed, 12)
    movement.rotate(speed, 90, Left)
    movement.moveForward(speed, 96)
    movement.rotate(speed, 90, Left)
    movement.moveForward(speed, 12)


def finalDemo4(speed, armSpeed):
    movement.moveForward(-speed, 6.3)
    movement.rotate(speed, 90, Right)

    movement.moveArm(armSpeed, -90, 5)

    movement.moveForward(speed, 2.5)

    movement.moveArm(armSpeed, 45, 5)

    movement.moveForward(-speed, 2.5)

    movement.rotate(speed, 90, Left)

    movement.moveForward(speed, 24)

    movement.moveArm(armSpeed, -45, 5)

```

```
def finalDemo3(speed, armSpeed):
    movement.moveForward(speed, 20.5)

    black = Color.BLACK
    white = Color.WHITE
    realBarCodes = [[white, white, white, black],
                    [white, black, white, black],
                    [white, white, black, black],
                    [black, white, white, black]]
    ourBarCode = [white, black, white, black]
    readBarCode = []
    for i in range(4):
        readBarCode.append(rightColor.color())
        ev3.speaker.beep()
        movement.moveForward(speed, 1.1)
        time.sleep(2)

    if readBarCode == ourBarCode:
        ev3.screen.draw_text(0,0, "Barcodes are")
        ev3.screen.draw_text(0,20, "the same")
    else:
        ev3.screen.draw_text(0,0, "Barcodes are")
        ev3.screen.draw_text(0,20, "not the same")

    time.sleep(3)

    finalDemo4(speed, armSpeed)

ROBOTSPEED = 500
ARMSPEED = 200
SHELF = "A1"
BOX = 7


if __name__ == '__main__':
    ev3.speaker.set_speech_options(voice = "m3")
    ev3.speaker.set_volume(100, which = "_all_")
    ev3.speaker.say("Starting")

    # finalDemo1(ROBOTSPEED, SHELF, BOX)
    # time.sleep(3)
    finalDemo2(ROBOTSPEED)

    # finalDemo3(ROBOTSPEED, ARMSPEED)
```

This is our final main function put together.

4/11/24

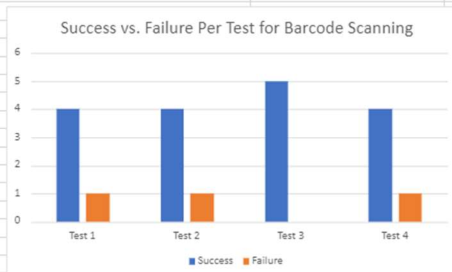
Plan

Barcode Scanning

Test Number	Criteria / specification	Description of test procedure	Data to be collected	Summary of Results
1	Know type of boxes based on scanning barcode: Scanning of barcode produces correct type of box 100% of the time	1) Place box with barcode 1 in front of robot 2) Instruct the robot to scan the barcode 3) Report the type of box/barcode identified 4) repeat 5 times or more.	Type of box identified	Calculate rate of accuracy for correctly scanning the barcode and identifiying the box. Plot number correct and incorrect for each type of barcode on a bar graph with results from Test Numbers 1 through 4.
2	Know type of boxes based on scanning barcode: Scanning of barcode produces correct type of box 100% of the time	1) Place box with barcode 2 in front of robot 2) Instruct the robot to scan the barcode 3) Report the type of box/barcode identified 4) repeat 5 times or more.	Type of box identified	
3	Know type of boxes based on scanning barcode: Scanning of barcode produces correct type of box 100% of the time	1) Place box with barcode 3 in front of robot 2) Instruct the robot to scan the barcode 3) Report the type of box/barcode identified 4) repeat 5 times or more.	Type of box identified	
4	Know type of boxes based on scanning barcode: Scanning of barcode produces correct type of box 100% of the time	1) Place box with barcode 4 in front of robot 2) Instruct the robot to scan the barcode 3) Report the type of box/barcode identified 4) repeat 5 times or more.	Type of box identified	

Results

Test	Success?	
1	Yes	
1	Yes	
1	Yes	
1	No	
1	Yes	
2	Yes	
2	Yes	
2	No	
2	Yes	
2	Yes	
3	Yes	
3	Yes	
3	Yes	
3	Yes	
3	Yes	
4	No	Accuracy: 85%
4	Yes	
4	Yes	
4	Yes	
4	Yes	



This is our test plan and data for barcode scanning. We achieved an 85% accuracy rate, as shown above.

4/9/24

Plan

Pickup

Test Number	Criteria / specification	Description of test procedure	Data to be collected	Summary of Results
1	Able to pick up box without falling over: Lifting box causes robot to move 0 cm and tilt 0 degrees.	1) Determine surface to test on and way to measure distance 2) Instruct the robot to pick up a 50g box. 3) Measure the distance moved and whether the robot tilted 4) repeat 5 times or more.	Distance robot moves in the x and y direction, and if robot tilted	Calculate average and standard deviation for movement in the x and y direction. Plot distance away on a chart with results from Test Numbers 1 through 4. Calculate number of times robot tilted vs. number of times without tilt for Test Numbers 1 through 4.
2	Able to pick up box without falling over: Lifting box causes robot to move 0 cm and tilt 0 degrees.	1) Determine surface to test on and way to measure distance 2) Instruct the robot to pick up a 100g box. 3) Measure the distance moved and whether the robot tilted 4) repeat 5 times or more.	Distance robot moves in the x and y direction, and if robot tilted	
3	Able to pick up box without falling over: Lifting box causes robot to move 0 cm and tilt 0 degrees.	1) Determine surface to test on and way to measure distance 2) Instruct the robot to pick up a 150g box. 3) Measure the distance moved and whether the robot tilted 4) repeat 5 times or more.	Distance robot moves in the x and y direction, and if robot tilted	
4	Able to pick up box without falling over: Lifting box causes robot to move 0 cm and tilt 0 degrees.	1) Determine surface to test on and way to measure distance 2) Instruct the robot to pick up a 200g box. 3) Measure the distance moved and whether the robot tilted 4) repeat 5 times or more.	Distance robot moves in the x and y direction, and if robot tilted	

Movement

Test Number	Criteria / specification	Description of test procedure	Data to be collected	Summary of Results
5	Know location in the warehouse at all times: Recorded location and actual location of the robot in the warehouse differ by less than 5% - moving straight	1) Determine surface to test on and way to measure distance and location 2) Instruct the robot to travel in a straight line for 60 inches with a 50g box 3) measure the distance off expected y and expected x 4) repeat 5 times or more.	Distance robot is away from the expected x and expected y	Calculate average and standard deviation for distance away from expected x and y location. Plot distance away on a chart with results from Test Numbers 5 through 8.
6	Know location in the warehouse at all times: Recorded location and actual location of the robot in the warehouse differ by less than 5% - moving straight	1) Determine surface to test on and way to measure distance and location 2) Instruct the robot to travel in a straight line for 60 inches with a 100g box 3) measure the distance off expected y and expected x 4) repeat 5 times or more.	Distance robot is away from the expected x and expected y	
7	Know location in the warehouse at all times: Recorded location and actual location of the robot in the warehouse differ by less than 5% - moving straight	1) Determine surface to test on and way to measure distance and location 2) Instruct the robot to travel in a straight line for 60 inches with a 150g box 3) measure the distance off expected y and expected x 4) repeat 5 times or more.	Distance robot is away from the expected x and expected y	
8	Know location in the warehouse at all times: Recorded location and actual location of the robot in the warehouse differ by less than 5% - moving straight	1) Determine surface to test on and way to measure distance and location 2) Instruct the robot to travel in a straight line for 60 inches with a 200g box 3) measure the distance off expected y and expected x 4) repeat 5 times or more.	Distance robot is away from the expected x and expected y	

TURNING

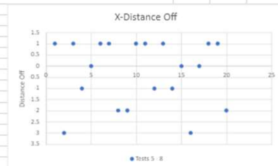
Test Number	Criteria / specification	Description of test procedure	Data to be collected	Summary of Results
9	Know location in the warehouse at all times: Recorded location and actual location of the robot in the warehouse differ by less than 5% - turning	1) Determine surface to test on and way to measure distance and location 2) Instruct the robot to travel in a straight line for 12 inches and then make a 90° turn and then travel 48 inches while carrying a 50g box 3) measure the distance off expected y and expected x 4) repeat 5 times or more.	Distance robot is away from the expected x and expected y	Calculate average and standard deviation for distance away from expected x and y location. Plot distance away on a chart with results from Test Numbers 9 through 12.
10	Know location in the warehouse at all times: Recorded location and actual location of the robot in the warehouse differ by less than 5% - turning	1) Determine surface to test on and way to measure distance and location 2) Instruct the robot to travel in a straight line for 12 inches and then make a 90° turn and then travel 48 inches while carrying a 100g box 3) measure the distance off expected y and expected x 4) repeat 5 times or more.	Distance robot is away from the expected x and expected y	
11	Know location in the warehouse at all times: Recorded location and actual location of the robot in the warehouse differ by less than 5% - turning	1) Determine surface to test on and way to measure distance and location 2) Instruct the robot to travel in a straight line for 12 inches and then make a 90° turn and then travel 48 inches while carrying a 150g box 3) measure the distance off expected y and expected x 4) repeat 5 times or more.	Distance robot is away from the expected x and expected y	
12	Know location in the warehouse at all times: Recorded location and actual location of the robot in the warehouse differ by less than 5% - turning	1) Determine surface to test on and way to measure distance and location 2) Instruct the robot to travel in a straight line for 12 inches and then make a 90° turn and then travel 48 inches while carrying a 200g box 3) measure the distance off expected y and expected x 4) repeat 5 times or more.	Distance robot is away from the expected x and expected y	

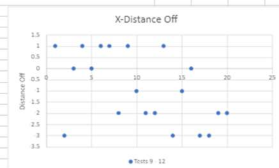
Luke Weis, Arpit Gill, AJ Armstrong, Alex Funk, 70

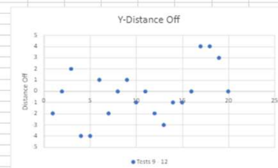
Results

Test	X-Distance Off	Y-Distance Off	Tilted?
1	1	0	No
1	-1	0	No
1	-1	0	No
1	1	0	No
1	-1	0	No
2	1	0	No
2	0	0	No
2	0	0	No
2	0	0	No
2	1	0	No
3	0	0	No
3	0	0	No
3	0	0	No
3	-1	0	No
3	0	0	No
4	1	0	No
4	-1	0	No
4	1	0	No
4	-1	0	No
4	1	0	No
Average	0.05	Average: 0	
Standard Deviation:	0.825577947	Standard Deviation: 0	

Test	X-Distance Off	Y-Distance Off	
5	1	3	
5	-3	0	
5	1	2	
5	-1	3	
5	0	0	
6	1	3	
6	1	0	
6	-2	0	
6	-2	3	
6	1	1	
7	1	3	
7	-1	1	
7	1	1	
7	-1	1	
7	0	2	
8	-3	1	
8	0	2	
8	1	3	
8	1	3	
8	-2	2	
9	1	-2	
9	-3	0	
9	0	2	
9	1	-4	
9	0	-4	
10	1	1	
10	1	-2	
10	-2	0	
10	1	1	
10	-1	-1	
11	-2	0	
11	-2	-2	
11	1	-3	
11	-3	-1	
11	-1	-1	
12	0	0	
12	-3	4	
12	-3	4	
12	-2	3	
12	-2	0	
Average	-0.6	Average:	0.725
Standard Deviation:	1.532552756	Standard Deviation:	2.062951582







This is our test plan and data for pickup of the box and movement with the box.

4/10/24

Luke Weis, Arpit Gill, AJ Armstrong, Alex Funk, 71

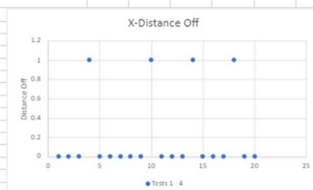
Plan

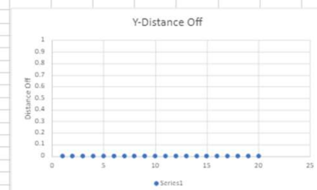
Drop Off the Bin

Test Number	Criteria / specification	Description of test procedure	Data to be collected	Summary of Results
1	Drop off the bin and not fall over	1) Set up the robot with a 50 g bin already lifted on the arm. 2) Instruct the robot to set the bin down. 3) See if the robot falls over.	Distance robot moves in the x and y direction, and if robot tilted	Calculate average and standard deviation for movement in the x and y direction. Plot distance off on a graph.
2	Drop off the bin and not fall over	1) Set up the robot with a 100 g bin already lifted on the arm. 2) Instruct the robot to set the bin down. 3) See if the robot falls over. 4) Repeat 5 times	Distance robot moves in the x and y direction, and if robot tilted	
3	Drop off the bin and not fall over	1) Set up the robot with a 150 g bin already lifted on the arm. 2) Instruct the robot to set the bin down. 3) See if the robot falls over. 4) Repeat 5 times	Distance robot moves in the x and y direction, and if robot tilted	
4	Drop off the bin and not fall over	1) Set up the robot with a 200 g bin already lifted on the arm. 2) Instruct the robot to set the bin down. 3) See if the robot falls over. 4) Repeat 5 times	Distance robot moves in the x and y direction, and if robot tilted	

Results

Test	X-Distance	Y-Distance	Titled?
1	0	0	No
1	0	0	No
1	0	0	No
1	1	0	No
1	0	0	No
2	0	0	No
2	0	0	No
2	0	0	No
2	0	0	No
2	1	0	No
3	0	0	No
3	0	0	No
3	0	0	No
3	1	0	No
3	0	0	No
4	0	0	No
4	0	0	No
4	1	0	No
4	0	0	No
4	0	0	No
Average:	0.2	Average: 0	
Standard Deviation:	0.410391341	Standard Deviation: 0	





This is our test plan and data for dropping off a bin the robot was already carrying.

4/10/24

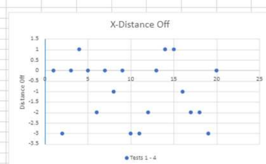
Luke Weis, Arpit Gill, AJ Armstrong, Alex Funk, 72

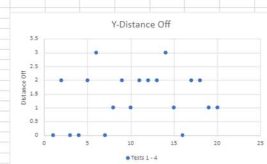
Plan

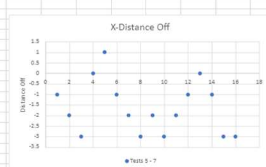
Test Number	Criteria / specification	Description of test procedure	Data to be collected	Summary of Results
Get to the right location				
1	Move around the course and get to the correct bin location	1) Setup the robot at the starting location. 2) Specifiy the robot to move to shelf A1 box 3 3) Repeat for 5 test	X and Y error in inches that the robot is off	Calculate average and standard deviation for distance away from expected x and y location. Plot the distance off.
2	Move around the course and get to the correct bin location	1) Setup the robot at the starting location. 2) Specifiy the robot to move to shelf C1 box 6 3) Repeat for 5 test	X and Y error in inches that the robot is off	
3	Move around the course and get to the correct bin location	1) Setup the robot at the starting location. 2) Specifiy the robot to move to shelf D1 box 9 3) Repeat for 5 test	X and Y error in inches that the robot is off	
4	Move around the course and get to the correct bin location	1) Setup the robot at the starting location. 2) Specifiy the robot to move to shelf B1 box 12 3) Repeat for 5 test	X and Y error in inches that the robot is off	
Get back to home base				
5	Move from a drop off location to home base	1) Setup the robot at the a home base B 2) Specifiy the robot to move to homebase A 4) Calculate how far off it is. 3) Repeat for 5 test	X and Y error in inches that the robot is off	Calculate average and standard deviation for distance away from expected x and y location. Plot the distance off.
6	Move from a drop off location to home base	1) Setup the robot at the a home base C 2) Specifiy the robot to move to homebase A 4) Calculate how far off it is. 3) Repeat for 5 test	X and Y error in inches that the robot is off	
7	Move from a drop off location to home base	1) Setup the robot at the a home base D 2) Specifiy the robot to move to homebase A 4) Calculate how far off it is. 3) Repeat for 5 test	X and Y error in inches that the robot is off	

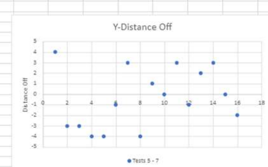
Results

Test	X-Distance Off	Y-Distance Off	
1	0		0
1	-3		2
1	0		0
1	1		0
1	0		2
2	-2		3
2	0		0
2	-1		1
2	0		2
2	-3		1
3	-3		2
3	-2		2
3	0		2
3	1		3
3	1		1
4	-1		0
4	-2		2
4	-2		2
4	-3		1
4	0		1
5	-1		4
5	-2		-3
5	-3		-3
5	0		-4
5	1		-4
6	-1		-1
6	-2		3
6	-3		-4
6	-2		1
6	-3		0
6	-2		3
7	-1		-1
7	0		2
7	-1		3
7	-3		0
7	-3		-2
Average:	-1.25	Average:	0.583333333
Standard Deviation:	1.380993431	Standard Deviation:	2.156385865









This is our test plan and data for the robot navigating the warehouse

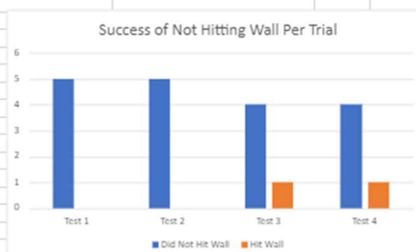
4/10/24

Object Avoidance

Test Number	Criteria / specification	Description of test procedure	Data to be collected	Summary of Results
1	Move forward in a 12 in wide hallway and not hit the walls.	1) Setup a straight path with walls on either side 12 in apart 2) Instruct the robot to travel in a straight line for 12 inches 3) see if the robot hits the walls. 4) Repeat for a total of 5 tests	Whether it not the robot hits the wall (true or false)	Calculate the percent the robot will hit the wall for each test.
2	Move forward in a 12 in wide hallway and not hit the walls.	1) Setup a straight path with walls on either side 12 in apart 2) Instruct the robot to travel in a straight line for 36 inches 3) see if the robot hits the walls. 4) Repeat for a total of 5 tests	Whether it not the robot hits the wall (true or false)	
3	Move forward in a 12 in wide hallway and not hit the walls.	1) Setup a straight path with walls on either side 12 in apart 2) Instruct the robot to travel in a straight line for 48 inches 3) see if the robot hits the walls. 4) Repeat for a total of 5 tests	Whether it not the robot hits the wall (true or false)	
4	Move forward in a 12 in wide hallway and not hit the walls.	1) Setup a straight path with walls on either side 12 in apart 2) Instruct the robot to travel in a straight line for 84 inches 3) see if the robot hits the walls. 4) Repeat for a total of 5 tests	Whether it not the robot hits the wall (true or false)	

Results

Test	Hit Wall?
1	No
1	No
1	No
1	No
1	No
2	No
2	No
2	No
2	No
2	No
3	No
3	No
3	No
3	Yes
3	No
4	No
4	No
4	Yes
4	No
4	No



Success Rate: 90%

This is our test plan and data for the robot's object avoidance system.

4/10/24

Meeting Agenda / Minutes **Project/Team: ENED 1120 Robot Team 238**

Date/Time: 3/26/24, 2:30pm-6:30pm Note Taker: Luke Weis

Present: Luke Weis, Alex Funk, AJ Armstrong, Arpit Gill

Agenda:

- Brainstorm ideas for lifting mechanism
- Develop lifting mechanism
- Test lifting mechanism
- Brainstorm cccd
- Brainstorm barcode scanning
- Brainstorm how to avoid collisions

Review Gantt Chart

Tasks behind schedule:

- Develop Lifting Mechanism
- Develop Code for Navigation
- Develop Barcode Scanner
- Develop Code to Scan barcode
- Complete Code for robot

☐ Tasks added to chart:

- Develop code for lifting box
- Develop code for barcode scanner
- Complete code for robot
- Test robot & make adjustments

Foreseeable issues that will prevent success ☐ No ☐ Yes

Decisions made:

- Lifting mechanism will use a forklift like design and grab the box from the top handle

Important information shared:

- Project Status Update is due Tuesday, 4/2

Ideas we want to remember:

Team member	Hours on project since last meeting (hrs)	Total hours on project (hrs)
Luke Weis	0	16
AJ Armstrong	0	16
Alex Funk	0	16
Arpit Gill	0	16

Meeting Agenda / Minutes

Project/Team: ENED 1120 Robot Team 238

Date/Time: 4/2/24, 2:30pm-6:30pm

Note Taker: Luke Weis

Present: Luke Weis, Alex Funk, AJ Armstrong, Arpit Gill

Agenda:

- Finish developing lifting mechanism
- Test lifting mechanism
- Brainstorm barcode scanning
- Develop barcode scanning
- Brainstorm how to avoid collisions
- Develop solution to avoid collisions
- Film Project Status Update

Review Gantt Chart

Tasks behind schedule:

- Develop Lifting Mechanism
- Develop Code for Navigation
- Develop Barcode Scanner
- Develop Code to Scan barcode
- Complete Code for robot

☐ Tasks added to chart:

Foreseeable issues that will prevent success ☐ No ☐ Yes

Decisions made:

- Lifting mechanism will use a forklift like design and grab the box from the top handle

Important information shared:

Ideas we want to remember:

Team member	Hours on project since last meeting (hrs)	Total hours on project (hrs)
Luke Weis	0	20
AJ Armstrong	0	20
Alex Funk	0	20
Arpit Gill	0	20

Meeting Agenda / Minutes **Project/Team: ENED 1120 Robot Team 238**

Date/Time: 4/3/24, 2:30pm-5:30pm Note Taker: Luke Weis

Present: Luke Weis, Alex Funk, AJ Armstrong, Arpit Gill

Agenda:

- Test lifting mechanism
- Develop barcode scanning
- Develop navigation code
- Test Barcode scanning
- Film Project Status Update
- CCCD Pre-approval

Review Gantt Chart

Tasks behind schedule:

- Develop Code for Navigation
- Develop Barcode Scanner
- Develop Code to Scan barcode
- Complete Code for robot

☐ Tasks added to chart:

Foreseeable issues that will prevent success ☐ No ☐ Yes

Decisions made:

- Robot will have barcode scanner on each side so it is easier to scan boxes on each side of isle
- Robot with have a weight on one side to counteract the weight of the medium motor
- Motors will run at different speeds to have the robot move straight

Important information shared:

Ideas we want to remember:

Team member	Hours on project since last meeting (hrs)	Total hours on project (hrs)
Luke Weis	0	23
AJ Armstrong	0	23
Alex Funk	0	23
Arpit Gill	0	23

Meeting Agenda / Minutes **Project/Team: ENED 1120 Robot Team 238**

Date/Time: 4/4/24, 2:30pm-6:30pm Note Taker: Luke Weis

Present: Luke Weis, Alex Funk, AJ Armstrong, Arpit Gill

Agenda:

- Develop subtask 1 and 2 code
- Test subtask 1 and 2 code

Review Gantt Chart

Tasks behind schedule:

- Develop Code for Navigation
- Develop Code to Scan barcode
- Complete Code for robot

☐ Tasks added to chart:

Foreseeable issues that will prevent success ☐ No ☐ Yes

Decisions made:

- We will have separate code for each subtask
- There will be a pause in the code to give the robot time to adjust

Important information shared:

Ideas we want to remember:

Team member	Hours on project since last meeting (hrs)	Total hours on project (hrs)
Luke Weis	0	27
AJ Armstrong	0	27
Alex Funk	0	27
Arpit Gill	0	27

Meeting Agenda / Minutes **Project/Team: ENED 1120 Robot Team 238**

Date/Time: 4/9/24, 2:30pm-6:30pm Note Taker: Luke Weis

Present: Luke Weis, Alex Funk, AJ Armstrong, Arpit Gill

Agenda:

- Develop Subtask 3 and 4 code
- Test Subtask 3 and 4 code

Review Gantt Chart

Tasks behind schedule:

- Develop Code to Scan barcode
- Complete Code for robot

☐ Tasks added to chart:

Foreseeable issues that will prevent success ☐ No ☐ Yes

Decisions made:

- Barcode scanning is done by having robot travel forward .25 inches and read each segment of the barcode (distance is based on time running motors for)
- Once the barcode is scanned, the robot will move backwards 5 inches and then turn to pick up the box
- The arm needs to be higher up to lift the box

Important information shared:

Ideas we want to remember:

Team member	Hours on project since last meeting (hrs)	Total hours on project (hrs)
Luke Weis	0	31
AJ Armstrong	0	31
Alex Funk	0	31
Arpit Gill	0	31

Meeting Agenda / Minutes **Project/Team: ENED 1120 Robot Team 238**

Date/Time: 4/10/24, 3:30pm-5:30pm Note Taker: Luke Weis

Present: Luke Weis, Alex Funk, AJ Armstrong, Arpit Gill

Agenda:

- Complete Subtask 4 code
- Test Subtask 4 code
- Test current code on paper in Rhodes 801

Review Gantt Chart

Tasks behind schedule:

- Complete Code for robot

☐ Tasks added to chart:

Foreseeable issues that will prevent success ☐ No ☐ Yes

Decisions made:

- Robot will travel for a certain distance to pick up box

Important information shared:

- We are all available Thursday from 12:30 – 3:30

Ideas we want to remember:

- The robot may behave differently on paper than the floor in our dorm

Team member	Hours on project since last meeting (hrs)	Total hours on project (hrs)
Luke Weis	0	33
AJ Armstrong	0	33
Alex Funk	0	33
Arpit Gill	0	33

Meeting Agenda / Minutes

Project/Team: ENED 1120 Robot Team 238

Date/Time: 4/11/24, 12:30pm-2:30pm

Note Taker: Luke Weis

Present: Luke Weis, Alex Funk, AJ Armstrong, Arpit Gill

Agenda:

- Test on paper in Rhodes 801

Review Gantt Chart

Tasks behind schedule:

☐ Tasks added to chart:

Foreseeable issues that will prevent success ☐ No ☐ Yes

Decisions made:

Important information shared:

- The robot performed well

Ideas we want to remember:

Team member	Hours on project since last meeting (hrs)	Total hours on project (hrs)
Luke Weis	0	35
AJ Armstrong	0	35
Alex Funk	0	35
Arpit Gill	0	35

Project name: ENED 1120 Robot

Project team: 238

Task ID	Task	Start	finish	Responsible	% Complete
A	Obtain materials for prototype	26-Jan	26-Jan	Luke & Arpit	100
B	Create & Maintain Design Notebook	29-Jan	15-Apr	Whole Team	100
C	Develop Movement Mechanism	29-Jan	11-Feb	Building Team	100
D	Develop Code for Movement	29-Jan	11-Feb	Coding Team	100
E	Test Movement & Make Adjustments	12-Feb	18-Feb	Whole Team	100
F	Develop Lifting Mechanism	19-Feb	3-Mar	Building Team	100
G	Develop Code for Navigation & Pickup	19-Feb	3-Mar	Coding Team	100
H	Develop Barcode Scanner on Robot	4-Mar	11-Mar	Building Team	100
I	Develop Code to Scan Barcode	4-Mar	18-Mar	Coding Team	100
J	Complete Code For Robot	18-Mar	24-Mar	Coding Team	100
K	Test Robot & Make Adjustments	25-Mar	7-Apr	Whole Team	100
L	Final Demonstration	12-Apr	12-Apr	Whole Team	100
M	Create Presentation	15-Apr	17-Apr	Whole Team	100

Date: 22-Jan, 29-Jan, 5-Feb, 12-Feb, 19-Feb, 26-Feb, 4-Mar, 11-Mar, 18-Mar, 25-Mar, 1-Apr, 8-Apr, 15-Apr

This is our completed Gannt chart.

4/14/24