DESIGN NOTEBOOK

EV3 ROBOT PROJECT

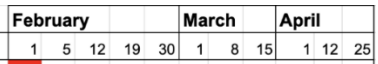
Team 004: Sarrah Barazi, Hoang Bao Chau Le
Jacob Haire

Date started: 2/1/24

Gantt Chart:

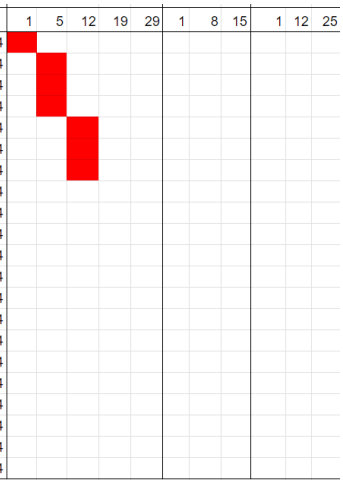
1st version – 02/02/24

Task	Lead	Start	End
Outline design specifications	Chau,Jacob,Sarrah,Nathan	02/01/24	02/05/24
Add/Work on design documents	Sarrah, Chau	02/01/24	02/25/24
Brainstorm walking/turning mechanisms	Jacob,Nathan	02/05/24	02/12/24
Brainstorm lifitng mechanisms	Sarrah, Chau	02/05/24	02/12/24
Choose design to implement	Chau, Jacob, Nathan, Sarrah	02/05/24	02/12/24
Build walking mechanism	Nathan	02/12/24	02/30/24
Build overall Robot Structure	Jacob, Nathan	02/12/24	02/30/24
Build lifting mechanism	Nathan	02/12/24	02/19/24
Attach motors	Jacob	2/30/24	03/08/24
Attach ultrasonic sensor	Jacob, Nathan	2/30/24	03/08/24
Program Lifting object	Chau, Sarrah	03/08/24	03/15/24
Program to move straight/ turn	Chau, Sarrah	03/08/24	03/15/24
Finalize all building and programming requirements	Chau, Sarrah, Jacob, Nathan	3/15/24	04/12/24
Test Robot	Chau, Sarrah, Jacob, Nathan	04/12/24	04/25/24
Final Demo/Presentation	Chau, Sarrah, Jacob, Nathan	04/25/24	04/25/24



02/15/24

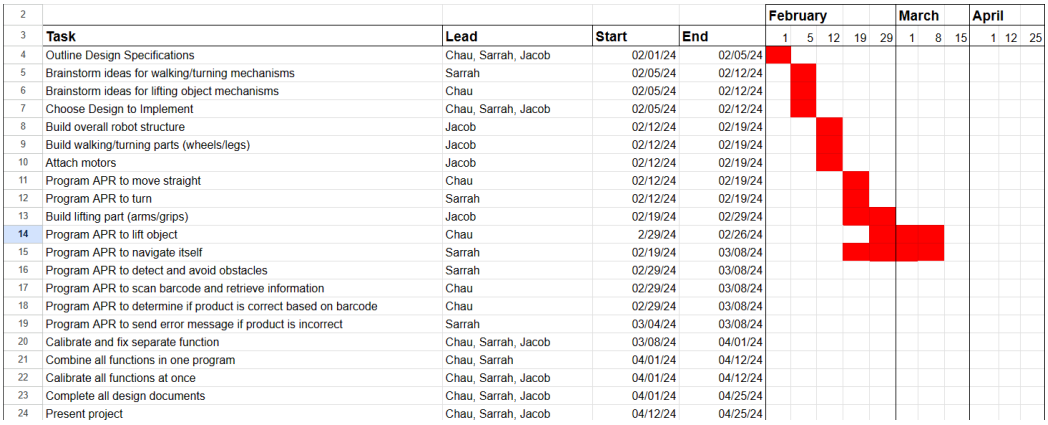
	Task	Lead	Start	End
4	Outline Design Specifications	Chau, Sarrah, Jacob	02/01/24	02/05/24
5	Brainstorm ideas for walking/turning mechanisms	Sarrah	02/05/24	02/12/24
6	Brainstorm ideas for lifting object mechanisms	Chau	02/05/24	02/12/24
7	Choose Design to Implement	Chau, Sarrah, Jacob	02/05/24	02/12/24
8	Build overall robot structure	Jacob	02/12/24	02/19/24
9	Build walking/turning parts (wheels/legs)	Jacob	02/12/24	02/19/24
10	Attach motors	Jacob	02/12/24	02/19/24
11	Program APR to move straight	Chau	02/12/24	02/19/24
12	Program APR to turn	Sarrah	02/12/24	02/19/24
13	Build lifting part (arms/grips)	Jacob	02/19/24	02/29/24
14	Program APR to lift object	Chau	2/29/24	02/26/24
15	Program APR to navigate itself	Sarrah	02/19/24	03/08/24
16	Program APR to detect and avoid obstacles	Sarrah	02/29/24	03/08/24
17	Program APR to scan barcode and retrieve information	Chau	02/29/24	03/08/24
18	Program APR to determine if product is correct based on barcode	Chau	02/29/24	03/08/24
19	Program APR to send error message if product is incorrect	Sarrah	03/04/24	03/08/24
20	Calibrate and fix separate function	Chau, Sarrah, Jacob	03/08/24	04/01/24
21	Combine all functions in one program	Chau, Sarrah	04/01/24	04/12/24
22	Calibrate all functions at once	Chau, Sarrah, Jacob	04/01/24	04/12/24
23	Complete all design documents	Chau, Sarrah, Jacob	04/01/24	04/25/24
24	Present project	Chau, Sarrah, Jacob	04/12/24	04/25/24



02/18/24

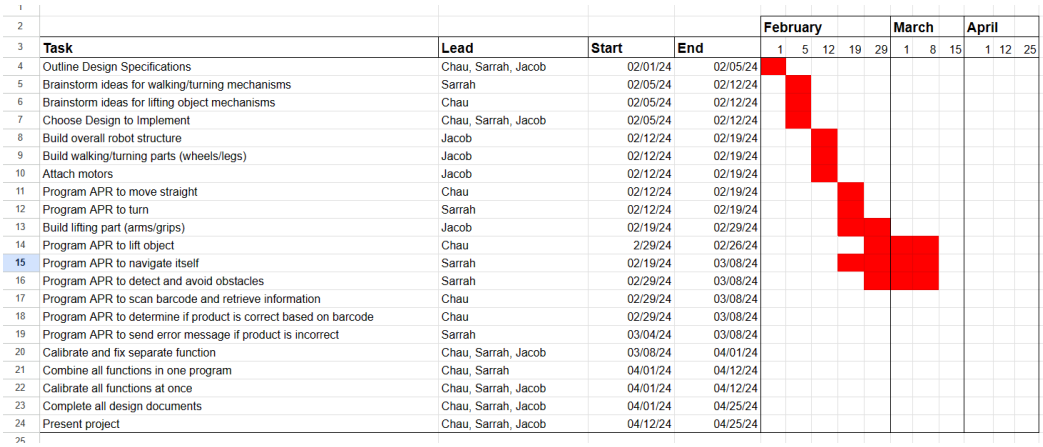
	Task	Lead	Start	End
4	Outline Design Specifications	Chau, Sarrah, Jacob	02/01/24	02/05/24
5	Brainstorm ideas for walking/turning mechanisms	Sarrah	02/05/24	02/12/24
6	Brainstorm ideas for lifting object mechanisms	Chau	02/05/24	02/12/24
7	Choose Design to Implement	Chau, Sarrah, Jacob	02/05/24	02/12/24
8	Build overall robot structure	Jacob	02/12/24	02/19/24
9	Build walking/turning parts (wheels/legs)	Jacob	02/12/24	02/19/24
10	Attach motors	Jacob	02/12/24	02/19/24
11	Program APR to move straight	Chau	02/12/24	02/19/24
12	Program APR to turn	Sarrah	02/12/24	02/19/24
13	Build lifting part (arms/grips)	Jacob	02/19/24	02/29/24
14	Program APR to lift object	Chau	2/29/24	02/26/24
15	Program APR to navigate itself	Sarrah	02/19/24	03/08/24
16	Program APR to detect and avoid obstacles	Sarrah	02/29/24	03/08/24
17	Program APR to scan barcode and retrieve information	Chau	02/29/24	03/08/24
18	Program APR to determine if product is correct based on barcode	Chau	02/29/24	03/08/24
19	Program APR to send error message if product is incorrect	Sarrah	03/04/24	03/08/24
20	Calibrate and fix separate function	Chau, Sarrah, Jacob	03/08/24	04/01/24
21	Combine all functions in one program	Chau, Sarrah	04/01/24	04/12/24
22	Calibrate all functions at once	Chau, Sarrah, Jacob	04/01/24	04/12/24
23	Complete all design documents	Chau, Sarrah, Jacob	04/01/24	04/25/24
24	Present project	Chau, Sarrah, Jacob	04/12/24	04/25/24

03/05/24



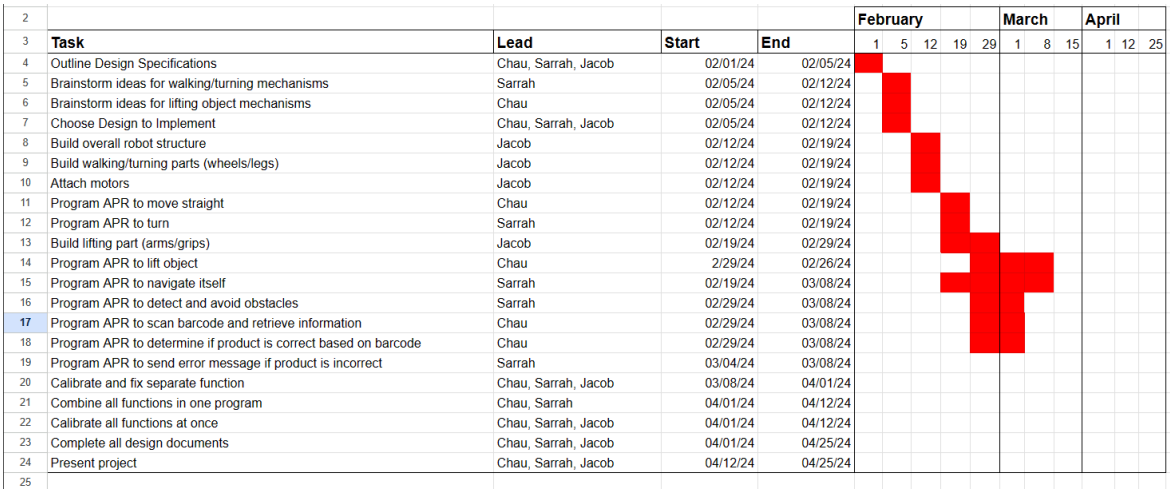
Task	Lead	Start	End
Outline Design Specifications	Chau, Sarrah, Jacob	02/01/24	02/05/24
Brainstorm ideas for walking/turning mechanisms	Sarrah	02/05/24	02/12/24
Brainstorm ideas for lifting object mechanisms	Chau	02/05/24	02/12/24
Choose Design to Implement	Chau, Sarrah, Jacob	02/05/24	02/12/24
Build overall robot structure	Jacob	02/12/24	02/19/24
Build walking/turning parts (wheels/legs)	Jacob	02/12/24	02/19/24
Attach motors	Jacob	02/12/24	02/19/24
Program APR to move straight	Chau	02/12/24	02/19/24
Program APR to turn	Sarrah	02/12/24	02/19/24
Build lifting part (arms/grips)	Jacob	02/19/24	02/29/24
Program APR to lift object	Chau	2/29/24	02/26/24
Program APR to navigate itself	Sarrah	02/19/24	03/08/24
Program APR to detect and avoid obstacles	Sarrah	02/29/24	03/08/24
Program APR to scan barcode and retrieve information	Chau	02/29/24	03/08/24
Program APR to determine if product is correct based on barcode	Chau	02/29/24	03/08/24
Program APR to send error message if product is incorrect	Sarrah	03/04/24	03/08/24
Calibrate and fix separate function	Chau, Sarrah, Jacob	03/08/24	04/01/24
Combine all functions in one program	Chau, Sarrah	04/01/24	04/12/24
Calibrate all functions at once	Chau, Sarrah, Jacob	04/01/24	04/12/24
Complete all design documents	Chau, Sarrah, Jacob	04/01/24	04/25/24
Present project	Chau, Sarrah, Jacob	04/12/24	04/25/24

03/07/24



Task	Lead	Start	End
Outline Design Specifications	Chau, Sarrah, Jacob	02/01/24	02/05/24
Brainstorm ideas for walking/turning mechanisms	Sarrah	02/05/24	02/12/24
Brainstorm ideas for lifting object mechanisms	Chau	02/05/24	02/12/24
Choose Design to Implement	Chau, Sarrah, Jacob	02/05/24	02/12/24
Build overall robot structure	Jacob	02/12/24	02/19/24
Build walking/turning parts (wheels/legs)	Jacob	02/12/24	02/19/24
Attach motors	Jacob	02/12/24	02/19/24
Program APR to move straight	Chau	02/12/24	02/19/24
Program APR to turn	Sarrah	02/12/24	02/19/24
Build lifting part (arms/grips)	Jacob	02/19/24	02/29/24
Program APR to lift object	Chau	2/29/24	02/26/24
Program APR to navigate itself	Sarrah	02/19/24	03/08/24
Program APR to detect and avoid obstacles	Sarrah	02/29/24	03/08/24
Program APR to scan barcode and retrieve information	Chau	02/29/24	03/08/24
Program APR to determine if product is correct based on barcode	Chau	02/29/24	03/08/24
Program APR to send error message if product is incorrect	Sarrah	03/04/24	03/08/24
Calibrate and fix separate function	Chau, Sarrah, Jacob	03/08/24	04/01/24
Combine all functions in one program	Chau, Sarrah	04/01/24	04/12/24
Calibrate all functions at once	Chau, Sarrah, Jacob	04/01/24	04/12/24
Complete all design documents	Chau, Sarrah, Jacob	04/01/24	04/25/24
Present project	Chau, Sarrah, Jacob	04/12/24	04/25/24

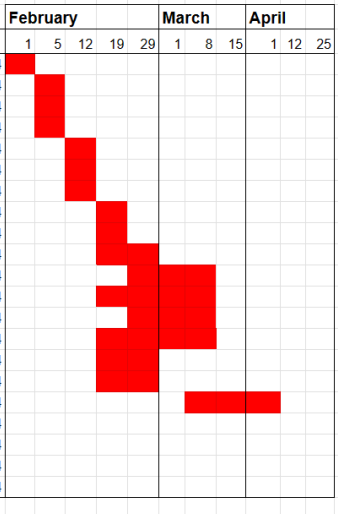
03/09/24



Task	Lead	Start	End
Outline Design Specifications	Chau, Sarrah, Jacob	02/01/24	02/05/24
Brainstorm ideas for walking/turning mechanisms	Sarrah	02/05/24	02/12/24
Brainstorm ideas for lifting object mechanisms	Chau	02/05/24	02/12/24
Choose Design to Implement	Chau, Sarrah, Jacob	02/05/24	02/12/24
Build overall robot structure	Jacob	02/12/24	02/19/24
Build walking/turning parts (wheels/legs)	Jacob	02/12/24	02/19/24
Attach motors	Jacob	02/12/24	02/19/24
Program APR to move straight	Chau	02/12/24	02/19/24
Program APR to turn	Sarrah	02/12/24	02/19/24
Build lifting part (arms/grips)	Jacob	02/19/24	02/29/24
Program APR to lift object	Chau	2/29/24	02/26/24
Program APR to navigate itself	Sarrah	02/19/24	03/08/24
Program APR to detect and avoid obstacles	Sarrah	02/29/24	03/08/24
Program APR to scan barcode and retrieve information	Chau	02/29/24	03/08/24
Program APR to determine if product is correct based on barcode	Chau	02/29/24	03/08/24
Program APR to send error message if product is incorrect	Sarrah	03/04/24	03/08/24
Calibrate and fix separate function	Chau, Sarrah, Jacob	03/08/24	04/01/24
Combine all functions in one program	Chau, Sarrah	04/01/24	04/12/24
Calibrate all functions at once	Chau, Sarrah, Jacob	04/01/24	04/12/24
Complete all design documents	Chau, Sarrah, Jacob	04/01/24	04/25/24
Present project	Chau, Sarrah, Jacob	04/12/24	04/25/24

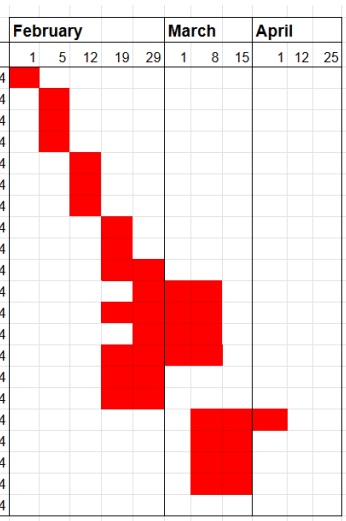
04/06/24

Task	Lead	Start	End
Outline Design Specifications	Chau, Sarrah, Jacob	02/01/24	02/05/24
Brainstorm ideas for walking/turning mechanisms	Sarrah	02/05/24	02/12/24
Brainstorm ideas for lifting object mechanisms	Chau	02/05/24	02/12/24
Choose Design to Implement	Chau, Sarrah, Jacob	02/05/24	02/12/24
Build overall robot structure	Jacob	02/12/24	02/19/24
Build walking/turning parts (wheels/legs)	Jacob	02/12/24	02/19/24
Attach motors	Jacob	02/12/24	02/19/24
Program APR to move straight	Chau	02/12/24	02/19/24
Program APR to turn	Sarrah	02/12/24	02/19/24
Build lifting part (arms/grips)	Jacob	02/19/24	03/31/24
Program APR to lift object	Chau	2/19/24	03/31/24
Program APR to navigate itself	Sarrah	02/19/24	03/31/24
Program APR to detect and avoid obstacles	Sarrah	02/19/24	03/31/24
Program APR to scan barcode and retrieve information	Chau	02/19/24	03/31/24
Program APR to determine if product is correct based on barcode	Chau	02/19/24	03/31/24
Program APR to send error message if product is incorrect	Sarrah	02/19/24	03/31/24
Calibrate and fix separate function	Chau, Sarrah, Jacob	03/08/24	04/01/24
Combine all functions in one program	Chau, Sarrah	04/01/24	04/12/24
Calibrate all functions at once	Chau, Sarrah, Jacob	04/01/24	04/12/24
Complete all design documents	Chau, Sarrah, Jacob	04/01/24	04/12/24
Present project	Chau, Sarrah, Jacob	04/12/24	04/19/24



04/11/24

Task	Lead	Start	End
Outline Design Specifications	Chau, Sarrah, Jacob	02/01/24	02/05/24
Brainstorm ideas for walking/turning mechanisms	Sarrah	02/05/24	02/12/24
Brainstorm ideas for lifting object mechanisms	Chau	02/05/24	02/12/24
Choose Design to Implement	Chau, Sarrah, Jacob	02/05/24	02/12/24
Build overall robot structure	Jacob	02/12/24	02/19/24
Build walking/turning parts (wheels/legs)	Jacob	02/12/24	02/19/24
Attach motors	Jacob	02/12/24	02/19/24
Program APR to move straight	Chau	02/12/24	02/19/24
Program APR to turn	Sarrah	02/12/24	02/19/24
Build lifting part (arms/grips)	Jacob	02/19/24	03/31/24
Program APR to lift object	Chau	2/19/24	03/31/24
Program APR to navigate itself	Sarrah	02/19/24	03/31/24
Program APR to detect and avoid obstacles	Sarrah	02/19/24	03/31/24
Program APR to scan barcode and retrieve information	Chau	02/19/24	03/31/24
Program APR to determine if product is correct based on barcode	Chau	02/19/24	03/31/24
Program APR to send error message if product is incorrect	Sarrah	02/19/24	03/31/24
Calibrate and fix separate function	Chau, Sarrah, Jacob	03/08/24	04/01/24
Combine all functions in one program	Chau, Sarrah	04/01/24	04/12/24
Calibrate all functions at once	Chau, Sarrah, Jacob	04/01/24	04/12/24
Complete all design documents	Chau, Sarrah, Jacob	04/01/24	04/12/24
Present project	Chau, Sarrah, Jacob	04/12/24	04/19/24



Meeting Agenda / Minutes	Project/team: Autonomous Product Retriever (APR) – Team 004
Date/time: 2/1/2024 – 4:30 to 6:00 PM	Note taker: Chau Le
Present: Chau Le, Sarrah Barazi, Jacob Haire	

Agenda:

- Complete the Lego Inventory

- Complete the Code of Cooperation

- Assign tasks to each member to finish in the shared Design Specifications document

Gantt Chart

				February					March			April		
Task	**Lead**	**Start**	**End**	1	5	12	19	30	1	8	15	1	12	25
Outline design specifications	Chau,Jacob,Sarrah,Nathan	02/01/24	02/05/24											
Add/Work on design documents	Sarrah, Chau	02/01/24	02/25/24											
Brainstorm walking/turning mechanisms	Jacob,Nathan	02/05/24	02/12/24											
Brainstorm lifitng mechanisms	Sarrah, Chau	02/05/24	02/12/24											
Choose design to implement	Chau, Jacob, Nathan, Sarrah	02/05/24	02/12/24											
Build walking mechanism	Nathan	02/12/24	02/30/24											
Build overall Robot Structure	Jacob, Nathan	02/12/24	02/30/24											
Build lifting mechanism	Nathan	02/12/24	02/19/24											
Attach motors	Jacob	2/30/24	03/08/24											
Attach ultrasonic sensor	Jacob, Nathan	2/30/24	03/08/24											
Program Lifting object	Chau, Sarrah	03/08/24	03/15/24											
Program to move straight/ turn	Chau, Sarrah	03/08/24	03/15/24											
Finalize all building and programming requirements	Chau, Sarrah, Jacob, Nathan	3/15/24	04/12/24											
Test Robot	Chau, Sarrah, Jacob, Nathan	04/12/24	04/25/24											
Final Demo/Presentation	Chau, Sarrah, Jacob, Nathan	04/25/24	04/25/24											

Foreseeable issues that will prevent success No Yes

Decisions made:

- Project documents and files will be shared through OneDrive

- Sarrah will keep the original design notebook file

Important information shared:

Contact information of each team member (stated in the Code of Cooperation)

Project timeline (indicated in the Gantt Chart)

Ideas we want to remember:

The robot in this project can be built with wheels and programmed with Python in Visual Studio Code.

Team member	Hours on project since last meeting (hrs)	Total hours on project (hrs)
Sarrah Barazi	1.0	1.0
Hoang Bao Chau Le	1.0	1.0
Jacob Haire	1.0	1.0

Meeting Agenda / Minutes

Project/team:	

Date/time: Thu Feb 15 4:30-7:40	Note taker: Sarrah

Present: Chau Sarrah Jacob

Agenda:

- Finish Building first Robot Prototype
- Begin working on Code

Review Gantt Chart

Tasks behind schedule

Task	Why behind	Plan to get up to date
	none	

☐ Tasks added to chart

Foreseeable issues that will prevent success ☒ No ☐ Yes

Decisions made:

We're continuing the code on Sunday 2/18

Important information shared:

Ideas we want to remember:

Team member	Hours on project since last meeting (hrs)	Total hours on project (hrs)
Chau	1	4.2 hours
Sarrah	1	4.2 hours
Jacob	1	4.2 hours

Meeting Agenda / Minutes

Project/team:

Date/time: Sun Feb 18 12:20-7:30

Note taker: Sarrah

Present: Chau, Sarrah

Agenda:

- complete code to move forward and backwards and turn certain angle

Review Gantt Chart

Tasks behind schedule

Task	Why behind	Plan to get up to date
	None	

☐ Tasks added to chart

Foreseeable issues that will prevent success ☒ No ☐ Yes

Decisions made:

Important information shared:

Ideas we want to remember:

Team member	Hours on project since last meeting (hrs)	Total hours on project (hrs)
Sarrah Barazi	7.5	11.7 hours
Chau	7.5	11.7 hours
Jacob	5.5	9.7 hours

Meeting Agenda / Minutes	Project/team: Mid Project Review
Date/time: March 2nd 3:00	Note taker: Sarrah Barazi
Present: Chau, Sarrah, Jacob	

Agenda:

Brief recap of the project's objectives and goals.

Quick overview of the research conducted on existing EV3 robot projects.

Share progress on building and testing the prototype.

Highlight any challenges faced during the prototyping phase.

Review Gantt Chart

Tasks behind schedule

Task	Why behind	Plan to get up to date
	NONE	

☐ Tasks added to chart NONE

Foreseeable issues that will prevent success ☐ No ☐ Yes

Decisions made:

Decided to prioritize resolving sensor calibration issues to keep the project on schedule.

Important information shared:

Shared a relevant tutorial on obstacle avoidance algorithms.

Ideas we want to remember:

Establish clear documentation standards for code, design decisions, and testing procedures to facilitate collaboration and knowledge transfer.

Team member	Hours on project since last meeting (hrs)	Total hours on project (hrs)
Sarrah	10	30
Chau	10	30
Jacob	10	30

Meeting Agenda / Minutes: Project/Team: 004

Date/time: 4/4/2024 4:30 – 7:30

Note Taker: Jacob

Present: Sarrah and Chau

Agenda:

- Move around color sensor so that robot will be able to be close to barcode
- Keep updating design notebook putting in all our ideas and prototypes along with test plans
- **Review Gantt Chart**
- **Tasks behind schedule**

Task	Why behind	Plan to get up to date
Scanning Barcode	We haven't tested much of the barcode and we are having trouble finding if the color sensor is working correctly.	Test both color sensors with an easy code that can easily diagnose whether it's a code issue or a malfunction
Testing	Some parts aren't built yet nor are the codes ready so we can't test yet	Get all parts finalized and get the code ready to at least run tests to measure accuracy.

Decisions made:

- Put color sensor on the left side of robot so it can reach the barcode with the forklift getting in the way.

Important information shared:

- Figure out if the color sensor will need to constantly be scanning or be programmed to scan only when it gets to a box.

Ideas we want to remember:

- Each of the four colors on the barcode are equally spaced apart at a half an inch.

Individual Work:

Team member	Hours on project since last meeting (hrs)	Total hours on project (hrs)
Chau	3	36
Jacob	3	36
Sarrah	3	36

Meeting Agenda / Minutes: Project/Team: 004

Date/time: 4/6/2024 4:30 – 7:30

Note Taker: Jacob Haire

Present: Jacob, Sarrah, and Chau

Agenda:

- Program barcode to work
- Look at the individual subtasks for the final demo and start testing and making sure we can complete all the tasks
- **Review Gantt Chart**
- **Tasks behind schedule**

Task	Why behind	Plan to get up to date
Programming barcode	The color sensor is picking up the wrong color	Figure out if the color sensor needs to be closer or if the code is messed up
Testing	Not all of the codes are finished so testing is hard to do	Get everything ready and then use the test plans created at the mid-project review and use those to see how good the robot works
Subtasks	Haven't had time to look at the final demo subtasks the robot must complete in order to get to the final track	Create codes for the subtasks after we make a flow diagram for what the robot must do for every task

Decisions made:

- Complete subtasks for final demo before making the final track code

Important information shared:

- Make sure the forklift can lift while navigating around

Ideas we want to remember:

- The final track code must be able to incorporate all the mechanisms and navigation features in one code

Individual Work:

Team member	Hours on project since last meeting (hrs)	Total hours on project (hrs)
Chau	3	39
Jacob	3	39
Sarrah	3	39

Meeting Agenda / Minutes: Project/Team: 004

Date/time: 3/26/2024 4:30 – 7:30

Note Taker: Jacob

Present: Sarrah, Chau

Agenda:

- **Get plan for status update**
- **Attach sensors and forklift**
- **Start on code for status update features**
- **Review Gantt Chart**
- **Tasks behind schedule**

Task	Why behind	Plan to get up to date
Building Lifting Mechanism	Got pushed off to complete midterm project review	Get this done before the status update
Program Lifting Mechanism	The mechanism must be built before any programming can start	Get this done before the status update
Program Sensors	Problems debugging the code and figuring out where the sensors can go without getting in the way of cords	Get this done before the status update

Decisions made:

- Put the color sensor on the upper left where the barcode will be. Manage to do this while leaving room for the cord to reach the sensor. Put an ultrasonic sensor on the front of the robot to avoid obstacles.

Important information shared:

- A rack and pinion approach might work for the forklift given the Lego materials we have.
- Gears and a tread could also possibly work to lift the object up.
- Make sure the lifting mechanism is high enough to get over the box.

Ideas we want to remember:

- The handle is 6 inches in the air.
- The box can weigh up to 200 grams.
- The color sensor must be close enough to the box to see the barcode

Individual Work:

Team member	Hours on project since last meeting (hrs)	Total hours on project (hrs)

Chau	3	33
Jacob	3	33
Sarrah	3	33

Meeting Agenda / Minutes: Project/Team: 004

Date/time: 4/9/2024 4:30 – 11:30

Note Taker: Jacob

Present: Jacob, Chau and Sarrah

Agenda:

- Test the robot of its functions\
- Test both its navigation skills again along with the barcode and lifting mechanism
- **Review Gantt Chart**
- **Tasks behind schedule**

Task	Why behind	Plan to get up to date
Testing	Worked on finishing the coding and making sure everything is built right so it can properly perform the task	Go through all the test plans today and record results along with analyzing the data
Final demo code	We must get the subtasks properly completed before putting all the code together for the final track	Finish the subtasks and work on compiling the codes into one final code

Decisions made:

- Change the angle for which the gyro sensor uses to determine if it turns.
- Complete project next meeting

Important information shared:

- For subtasks 1 and 2 we must program the robot to not hit the shelves in the way.

Ideas we want to remember:

- Finish updating design notebook next meeting

Individual Work:

Team member	Hours on project since last meeting (hrs)	Total hours on project (hrs)
Chau	7	46
Jacob	7	46
Sarrah	7	46

DESIGN SPECIFICATIONS

I. Empathize and Define

a. *Five stakeholders*: Buy More store, WTW inventory system, Buy More store's customers, Buy More employees, APR's controllers

b. Stakeholders' wants and needs

Stakeholder - Wants/needs	**Criteria**	**Specifications**
Buy More store 1. APR must avoid collisions with other humans/objects. 2. APR can quickly scan barcodes for necessary information.	1. APR can detect obstacles by scanning the surroundings while moving. 2. APR takes as little time as possible to recognize and decode the barcodes.	1. APR can detect obstacles within 3.2 in from its attached scanner. 2. APR can recognize barcodes from at least 1.5 in away and retrieve information within 5 seconds.
WTW inventory system 1. APR can move easily around the warehouse. 2. APR is efficient in retrieving and loading products.	1. APR is lightweight enough to balance itself while moving. 2. APR can move quickly and hold the products firmly while carrying them to the designated area.	1. APR together with the product weighs no more than 2 kg. 2. APR can go straight at a speed of at least 0.5 ft/s and turn at a speed of at least 10 deg/s.
Buy More store's customers 1. APR can help streamline the delivery process so that they will receive their products after less waiting time. 2. There will be minimal occasions when the products they receive are incorrect.	1. APR can process as many orders as possible in one working day. 2. APR can retrieve correct products for the orders made.	1. APR can process at least 50 orders every working day. 2. APR can verify and give back the correct barcode data of a wrong product no later than 10 seconds after the incorrect information is detected.
Buy More employees	1. APR can firmly hold and carry large boxes of	1. APR has a lifting mechanism that allows it to flexibly grip and

1. APR can handle heavy tasks related to lifting and carrying products around warehouse. 2. APR does not emit harmful chemical excess while and after working.	products with different weights. 2. APR runs on environmentally friendly energy and is energy-saving.	carry objects of up to 200 g in weight without damaging the initial shapes of the boxes. 2. APR does not use fossil fuel and has a rechargeable battery that can work 1-2 full days after each charge.
APR's controllers 1. APR is easy to operate and maintain. 2. APR is safe to handle.	1. APR has a simple operation technique and is long-lasting (i.e. does not need much maintenance). 2. APR can be stopped easily once the controller recognizes any risk of injuries or damage.	1. APR does not have many complex wires and its operation does not take more than 4 minutes to learn. 2. APR can be stopped easily within 2 seconds after the controller recognizes any risk.

c. Ethical issues related to profit, politics, person gains, or pressure to perform that might impact design process

- Prioritize profits over APR's quality/safety which can lead to sacrifice of some oulined specifications
- APR may lead to many employees losing their jobs due to Buy More's automation policy to reduce labor costs
- Data privacy may become a concern since the APR can scan barcode for information
- Pressure to meet deadlines may result in rushed design and building or not testing all aspects of APR's functions
- APR operates on low-quality or cheap energy source to maximize profitability, which can seriously affect its performance

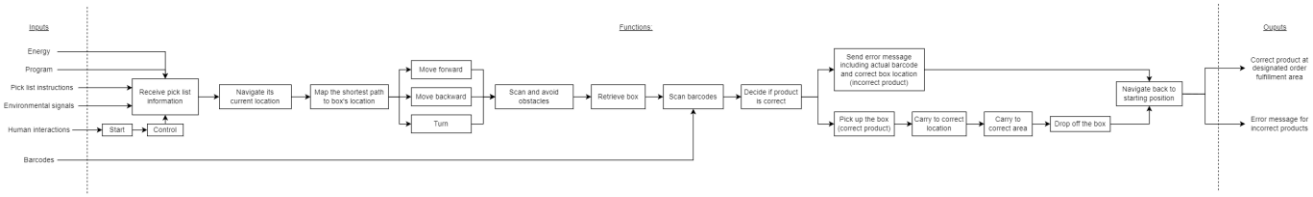
d. Constraints impacting the project and design (environment factors that might impact the performance of the robot)

- Uneven terrain
- Narrow aisles/tight spaces
- Inconsistent lighting conditions within the warehouse
- Extreme temperatures (especially if the warehouse is not heated/cooled enough during harsh weather conditions)
- Dust covers the APR's sensor after long periods of time using
- Exposure to high humidity

e. Major functions of the robot

- Receive pick list information (product barcode, box location, order fulfillment area)
- Navigate its location
- Map the shortest path to the box's location
- Scan and avoid obstacles while moving/turning
- Retrieve the box
- Scan barcodes and decide if the product is correct
- If correct product → Pick up and carry the box to the designated area
- Drop off product at order fulfillment area
- If incorrect product → Send error message including actual barcode and correct box location
- Navigate and move back to starting position

f. Functional Block Diagram of robot's functions

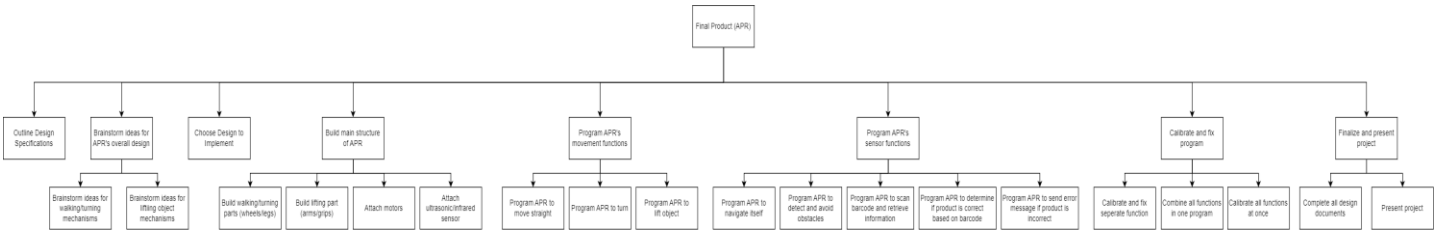


II. Project Management Information

a. Major tasks need to be completed to create the necessary prototype

- Outline design specifications
- Brainstorm ideas for overall APR design
- Choose design to implement
- Build main structure of APR
- Program APR's movement functions
- Program's APR's sensor functions
- Calibrate and fix program
- Finalize project and present

b. Work Breakdown Structure

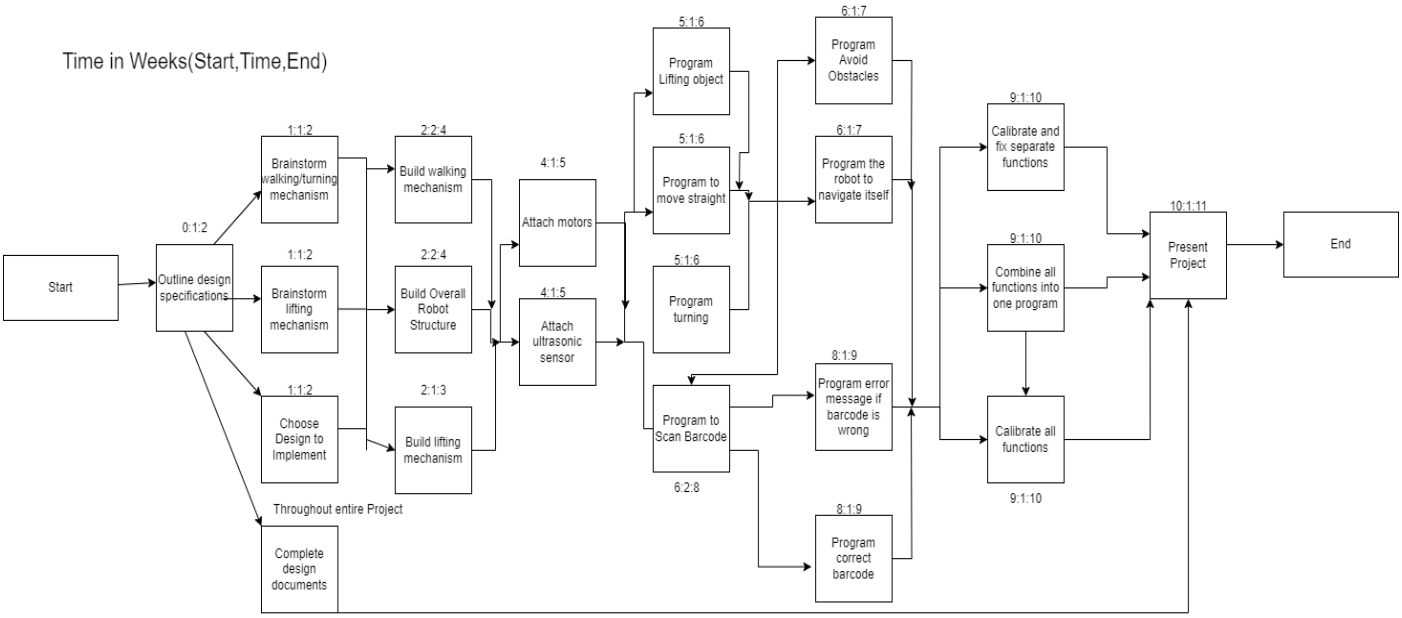


c. Estimate the time to complete each task based on the timeline of deliverables

- Outline Design Specifications: 1 week
- Brainstorm ideas for walking/turning mechanisms: 1 week
- Brainstorm ideas for lifting object mechanisms: 1 week

- Choose Design to Implement: 1 week
- Build overall robot structure: 2 weeks
- Build walking/turning parts (wheels/legs): 2 weeks
- Build lifting part (arms/grips): 1 week
- Attach motors: 1 week
- Attach ultrasonic/infrared sensor: 1 week
- Program APR to move straight: 1 week
- Program APR to turn: 1 week
- Program APR to lift object: 1 week
- Program APR to navigate itself: 2 weeks
- Program APR to detect and avoid obstacles: 1 week
- Program APR to scan barcode and retrieve information: 2 weeks
- Program APR to determine if product is correct based on barcode: 1 week
- Program APR to send error message if product is incorrect: 1 week
- Calibrate and fix separate function: 1 week
- Combine all functions in one program: 1 week
- Calibrate all functions at once: 1 week
- Complete all design documents: throughout project (1 week to finalize)
- Present project: 1 week

d. Precedence Network



e. Gantt Chart

(colored is completed)

				February					March			April		
Task	**Lead**	**Start**	**End**	1	5	12	19	30	1	8	15	1	12	25
Outline design specifications	Chau,Jacob,Sarrah,Nathan	02/01/24	02/05/24											
Add/Work on design documents	Sarrah, Chau	02/01/24	02/25/24											
Brainstorm walking/turning mechanisms	Jacob,Nathan	02/05/24	02/12/24											
Brainstorm lifitng mechanisms	Sarrah, Chau	02/05/24	02/12/24											
Choose design to implement	Chau, Jacob, Nathan, Sarrah	02/05/24	02/12/24											
Build walking mechanism	Nathan	02/12/24	02/30/24											
Build overall Robot Structure	Jacob, Nathan	02/12/24	02/30/24											
Build lifting mechanism	Nathan	02/12/24	02/19/24											
Attach motors	Jacob	2/30/24	03/08/24											
Attach ultrasonic sensor	Jacob, Nathan	2/30/24	03/08/24											
Program Lifting object	Chau, Sarrah	03/08/24	03/15/24											
Program to move straight/ turn	Chau, Sarrah	03/08/24	03/15/24											
Finalize all building and programming requirements	Chau, Sarrah, Jacob, Nathan	3/15/24	04/12/24											
Test Robot	Chau, Sarrah, Jacob, Nathan	04/12/24	04/25/24											
Final Demo/Presentation	Chau, Sarrah, Jacob, Nathan	04/25/24	04/25/24											

III. **Estimated Project Budget**

- Estimate the number of hours required by the team for each task of the project
- Estimated budget ($150/hour/person)

For one-week tasks: 4 hours each task

For two-week tasks: 8 hours each task

Completing Design Documents: 18 hours in total

Estimated Project Budget: $68,400

Meeting Agenda / Minutes: Project/Team: 004

Date/time: 4/11/2024 4:30 – 12:00

Note Taker: Jacob

Present: Jacob, Chau, and Sarrah

Agenda:

- Complete final code and subtasks
- **Review Gantt Chart**
- **Tasks behind schedule**

Task	Why behind	Plan to get up to date
Final code	The subtasks need to be completed before the final code	Complete the final code and subtasks tonight
Subtasks	Some testing was needed to be completed before the subtasks could be finished	Complete all 4 of the subtasks

Decisions made:

- Create separate codes for all the subtasks.
- Have two different codes prepared for subtask 3 in case one code doesn't work since some codes we're working on and off.
- The final code is compiled through all the subtask skills.

Important information shared:

- The robot goes forward from A to B in subtask 1
- The robot goes backward and then turns to go from B to A in subtask 2
- Subtask 3 requires a message to be displayed saying correct barcode
- Subtask 4 needs the robot to start in the position of reading the barcode

Ideas we want to remember:

- Get up tomorrow and make it to the check in by 9:10 am

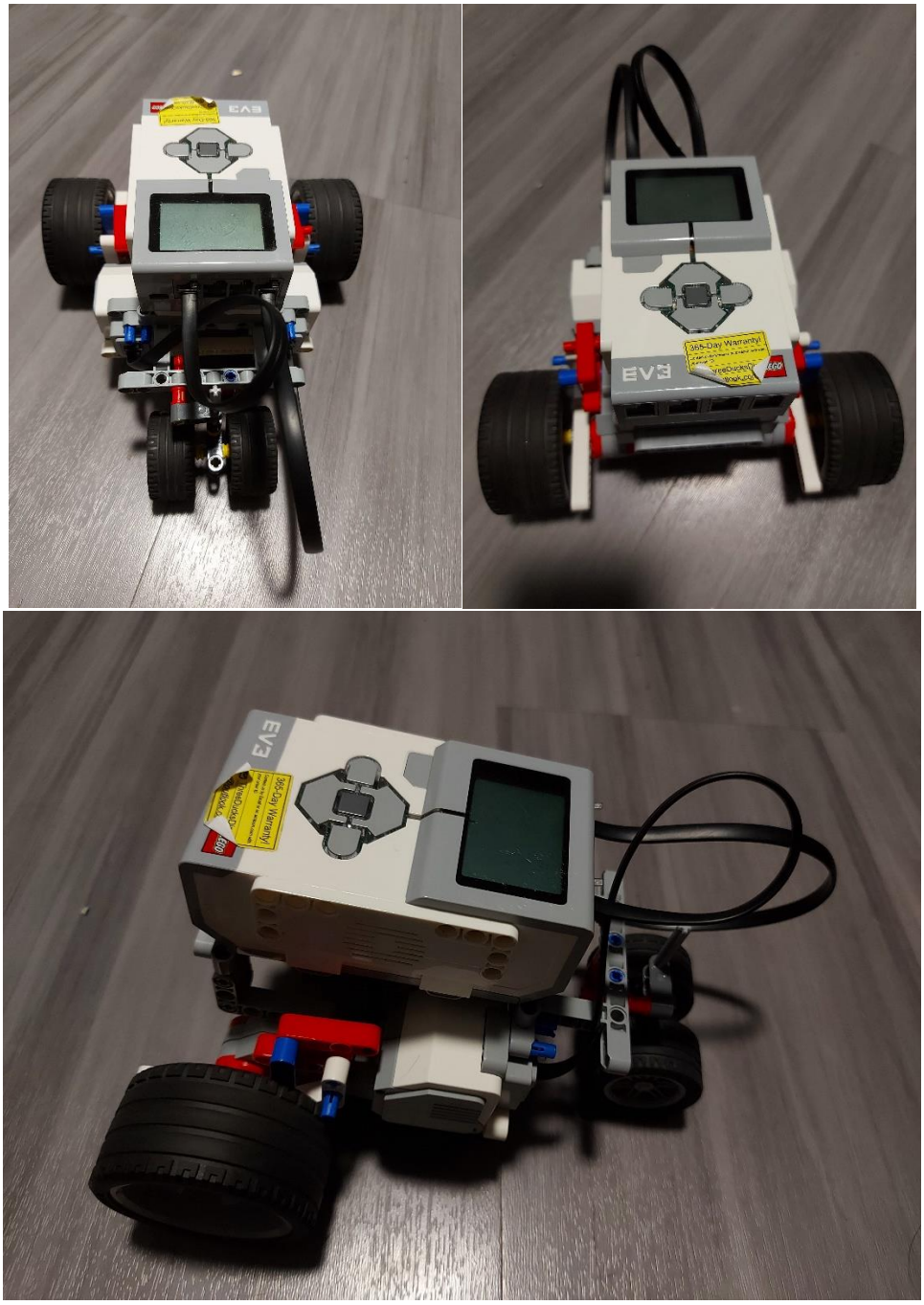
Individual Work:

Team member	Hours on project since last meeting (hrs)	Total hours on project (hrs)
Chau	7.5	53.5
Jacob	7.5	53.5
Sarrah	7.5	53.5

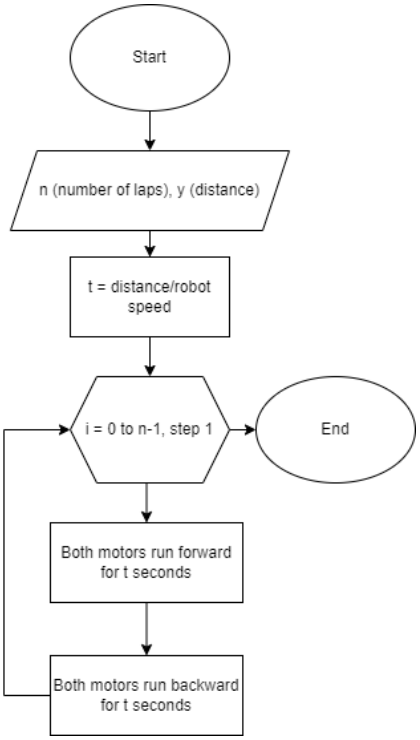
2/18/2024

First Demo Approach:

1st Prototype Design:



Pseudocode of robot's function of moving forward and reverse:



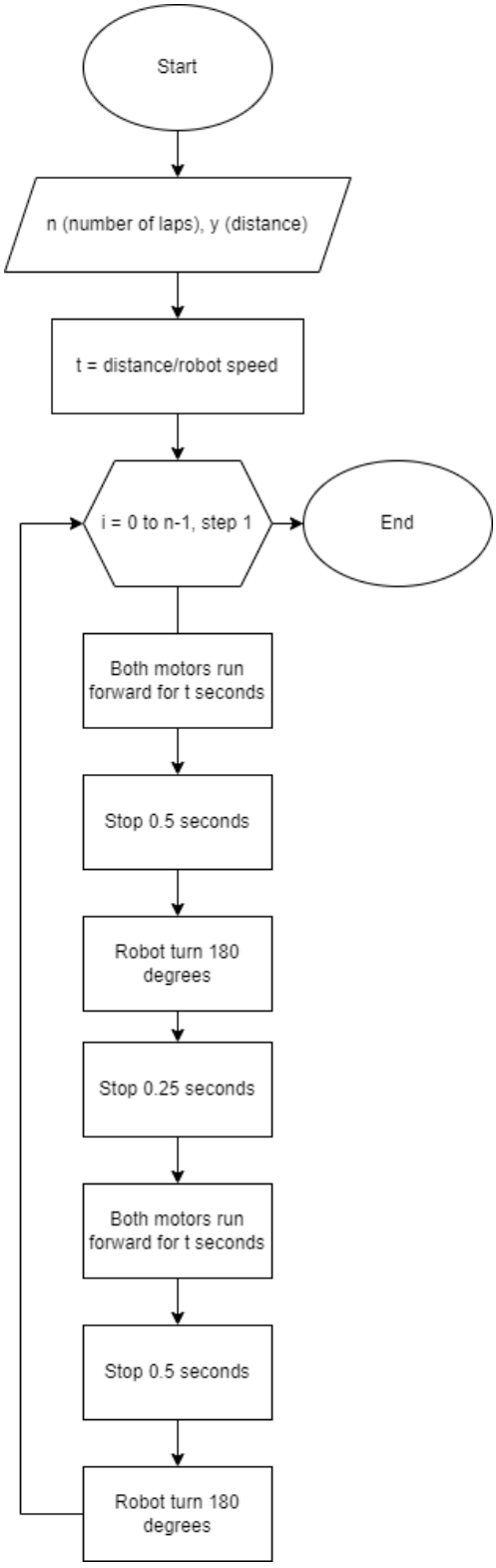
Python program of the moving straight function:

RunBothMotors.py ● Rotate180.py ●

C: > Users > lehoa > OneDrive > Documents > Python Projects > RobotCode > RunBothMotors.py > ...

```python
#!/usr/bin/env python3

## Imports all Objects and Methods from the motor module
from ev3dev2.motor import *

## Sets mt Equal to the MoveTank of two motors allowing
## them to run concurrently
mt = MoveTank(OUTPUT_A,OUTPUT_D)

# Calculate time run based on distance input
distance = float(input("Enter the given length to travel: "))
lap = int(input("Enter the number of laps: "))
time = distance/(37.5)

# Run Motor 1 at 60% max speed and Run Motor 2 at 60% Max Speed for t seconds, repeat n times
for i in range(lap):
    mt.on_for_seconds(SpeedPercent(-60),SpeedPercent(-60),time)
    mt.on_for_seconds(SpeedPercent(60),SpeedPercent(60),time)
```

Pseudocode of robot's function of moving forward and turning 180^{o} after each length:



Python program of the moving straight and turning 180^{o} function:

Rotate180.py

C: > Users > lehoa > OneDrive > Documents > Python Projects > RobotCode > Rotate180.py > ...

```python
#!/usr/bin/env python3

## Imports all Objects and Methods from the motor module
from ev3dev2.motor import *

## Sets mt Equal to the MoveTank of two motors allowing
## them to run concurrently
mt = MoveTank(OUTPUT_A,OUTPUT_D)

# Gather inputs and calculate run time

distance = float(input("Enter the given length to travel: "))
lap = int(input("Enter the number of laps: "))
time = distance/(37.5)

# Move forward, stop and rotate for n times
for i in range(lap):
    mt.on_for_seconds(SpeedPercent(-60),SpeedPercent(-60),time)
    mt.on_for_seconds(SpeedPercent(0),SpeedPercent(0),0.5)
    mt.on_for_degrees(SpeedPercent(22),SpeedPercent(-22),1755)
    mt.on_for_seconds(SpeedPercent(0),SpeedPercent(0),0.25)
    mt.on_for_seconds(SpeedPercent(-60),SpeedPercent(-60),time)
    mt.on_for_seconds(SpeedPercent(0),SpeedPercent(0),0.5)
    mt.on_for_degrees(SpeedPercent(-22),SpeedPercent(22),1755)

```

Project Subtask 1 Demo

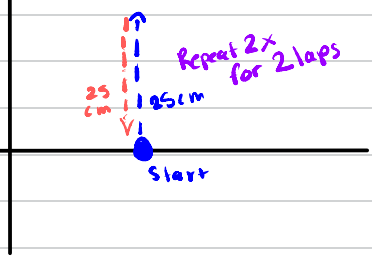
Demo Objective

- predict the precision of robots movements and where the robot will end up relative to start position

Example Run

- Gyro Sensor used to keep robot in straight line

n = 2
y = 25 cm



Calculating avg speed

Speed: 60% Time: 5 sec

Trial 1	74.5 in
Trial 2	73 in
Trial 3	73.8 in

avg distance: 73.8 in

187.4 cm

37.5 cm/sec | 14.8 in/sec

100 cm will have to run for 2.67 seconds

Predicting x, y displacement

Trial 1

Laps	(x, y) from (0,0)	Trial 2	x, y avg
1	(1.905cm, 2.54cm)	(1.905cm, 1.27cm)	(1.905, 1.905)
2	(4.064cm, 2.54cm)	(1.445cm, 0cm)	(4.255, 1.27)
3	(5.625cm, 5.08cm)	(8.573cm, 5.08cm)	(7.099, 5.08)
4	(13.653cm, 2.54cm)		

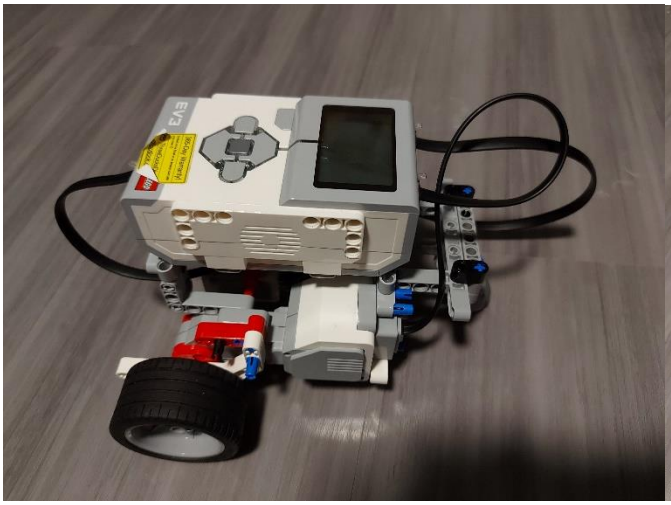
When predicting, if y > 100 cm x and y will be ≥ 5 when n = 3

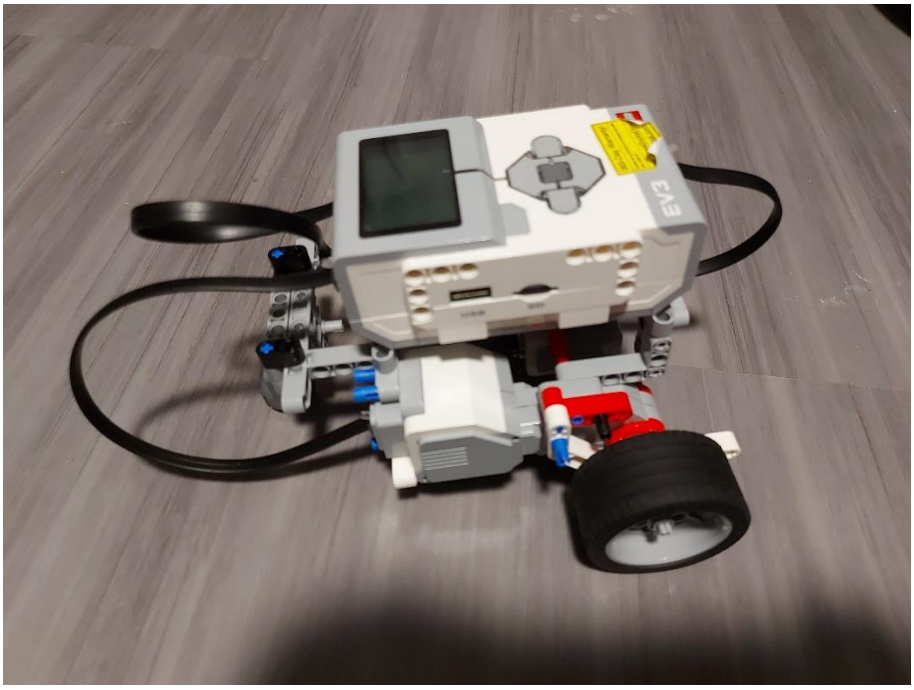
5 3/8 in

(6, 2cm)

n = 3 laps
y = 65 cm

Robot New Design (Adjusted 2/28/2024)





Reasons for changing design:

The old design features two front wheels that are right next to each other with no motors attached, thus making it difficult for the robot to rotate or turn. Besides, the front baller provides better stability and balance in the robot's movement compared to the old design.

Design Notebook Entry: EV3 Robot Project

Date: March 2nd 2024

Objective and Goals:

Our objective is to create an EV3 robot capable of autonomously navigating a predefined path, avoiding obstacles, and performing basic tasks. The primary goals are to enhance our understanding of robotics programming and explore the capabilities of the EV3 platform.

Research:

Researched existing EV3 robot projects and online resources to gain insights into programming challenges and sensor integration. Reviewed EV3 sensor documentation to understand their capabilities and limitations.

Brainstorming:

Concept 1: Line Following Robot

- Utilize color sensors for line detection.
- Implement PID control for precise movement along the path.
- Pros: Simple, efficient for predefined paths.
- Cons: Limited adaptability to dynamic environments.

Concept 2: Obstacle Avoidance Robot

- Employ ultrasonic sensors to detect obstacles.
- Implement a decision-making algorithm for obstacle avoidance.
- Pros: Suitable for dynamic environments.
- Cons: May struggle with complex obstacle arrangements.

Decision-Making:

After thorough consideration, we decided to pursue Concept 2 - the Obstacle Avoidance Robot. This decision is based on our interest in creating a versatile

robot capable of navigating dynamic environments. The use of ultrasonic sensors aligns with our goal of real-time adaptability.

Next Steps:

Began prototyping the basic chassis and integrated the EV3 brick with ultrasonic sensors. Tests should show promising obstacle detection capabilities. However, fine-tuning is required to optimize the decision-making algorithm for smoother navigation. Refine the obstacle avoidance algorithm for better responsiveness. Integrate additional sensors for improved environmental awareness. Test the robot in various scenarios to identify potential challenges.

Reflection:

The decision to focus on obstacle avoidance aligns with the project's goals, providing a balance between complexity and feasibility. The prototyping phase highlighted the need for precise tuning in sensor integration and algorithm development. Looking forward to refining the design in subsequent iterations.

Design Notebook Entry: EV3 Robot Test Plan

Date: March 5th

Objective and Goals:

Develop a comprehensive test plan to evaluate the functionality and performance of three major components of the EV3 robot: Barcode Scanning, Lifting Mechanism, and Overall Navigation.

Barcode Scanning:

Test Setup: A paper track with predefined barcode configurations.

Test 1: First Margin Filled

- Program the robot to pick up the box if the first margin is filled.
- Program the robot to identify the first margin as incorrect, resulting in no box pickup and a 5-inch backup.

- Measure the accuracy of the robot's decision-making.
- Repeat the test 5 times.

Test 2: Second Margin Filled

- Repeat the procedure with the second margin filled.
- Measure and document the accuracy of the barcode scanning function.
- Repeat the test 5 times.

Test 3: Third Margin Filled

- Repeat the procedure with the third margin filled.
- Assess accuracy and document results.
- Repeat the test 5 times.

Test 4: Fourth Margin Filled

- Repeat the procedure with the fourth margin filled.
- Evaluate accuracy and document observations.
- Repeat the test 5 times.

Lifting Mechanism:

Test Setup: Paper track with defined distances and angles.

Test 1: Straight Lifting

- Program the robot to go straight for 10 inches, pick up the box, turn 90 degrees, and move another 10 inches to the right.
- Repeat the procedure 5 times.
- Measure and calculate average distance from the expected x and y values.
- Create scatter plots to assess lifting mechanism accuracy.

Test 2: 135-Degree Lifting

- Repeat the procedure with a 135-degree turn.
- Assess accuracy and measure deviation.
- Repeat the test 5 times.

Test 3: 180-Degree Lifting

- Repeat the procedure with a 180-degree turn.
- Document accuracy and calculate standard deviation.
- Repeat the test 5 times.

Test 4: 270-Degree Lifting

- Repeat the procedure with a 270-degree turn.
- Evaluate accuracy and record results.

- Repeat the test 5 times.

Overall Navigation:

Test Setup: Simulated course with walls and boxes using an ultrasonic sensor.

Test 1: Square Format Course
- Arrange boxes in a square format, 10 inches apart.
- Program the robot to navigate through the course without collisions.
- Measure the success rate and the number of obstacles encountered.
- Repeat the test 5 times.

Test 2: Triangle Format Course
- Repeat the procedure with boxes arranged in a triangle format.
- Assess the robot's adaptability to different layouts.
- Repeat the test 5 times.

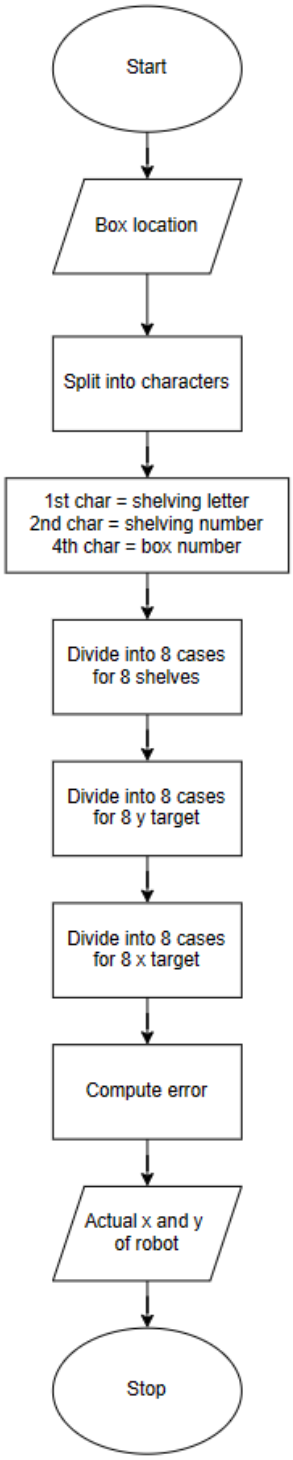
Test 3: Random Layout Course
- Scatter boxes randomly and test the robot's ability to navigate.
- Record success rates and obstacle encounters.
- Repeat the test 5 times.

Test 4: Object Detection
- Step in front of the robot during navigation to simulate a moving object.
- Evaluate the robot's ability to detect and respond to dynamic obstacles.
- Repeat the test 5 times.

Conclusion:

Through the execution of these tests, we aim to assess and ensure the effectiveness of each individual function of the EV3 robot. The insights gained from these tests will guide the final code integration for seamless operation.

Predicting error:



Functions used in error prediction model:

```python
# Function to calculate the x and y distance of the box location from the bottom left corner position of the shelf
def boxPosition(boxNumber):
    if (boxNumber >= 1 and boxNumber <= 6):
        finalX = initialX + 3 * boxColumnDistance[boxNumber - 1]
        finalY = initialY
    elif (boxNumber > 6 and boxNumber <= 12):
        finalX = initialX + 3 * boxColumnDistance[boxNumber - 7]  # Adjusted indexing for second row
        finalY = initialY + SHELF_WIDTH
    else:
        print("Invalid box number!")
        return None, None  # Return None for invalid box number
    return finalX, finalY
```

```python
def predictError1(yBox):  # Error caused by driving straight to the target y only
    xOff, yOff = 0, 0
    if (yBox == 12):  # Shelf A1, B1
        xOff += 0.2125
        yOff += 0.025
    elif (yBox == 24):
        xOff += 0.26875
        yOff += 0.29375
    elif (yBox == 36):
        xOff += 0.325
        yOff += 0.5625
    elif (yBox == 48):
        xOff += 0.575
        yOff += 0.3625
    elif (yBox == 60):
        xOff += 0.825
        yOff += 0.1625
    elif (yBox == 72):
        xOff += 0.70625
        yOff += 0.275
    elif (yBox == 84):
        xOff += 0.5875
        yOff += 0.3875
    elif (yBox == 96):
        xOff += 0.734375
        yOff += 0.484375
    return xOff, yOff
```

```python
def predictError2(xBox,xOff,yOff):  # Error caused by driving straight to the target x after turning 90 degrees from the target y
    if (xBox == 12):  # Shelf A1, A2
        xOff += 0.3
        yOff += 0.8625
    elif (xBox == 24):
        xOff += 0.75625
        yOff += 1.075
    elif (xBox == 36):
        xOff += 1.2125
        yOff += 1.2875
    elif (xBox == 48):
        xOff += 2.0125
        yOff += 2.1875
    elif (xBox == 60):
        xOff += 2.8125
        yOff += 3.0875
    elif (xBox == 72):
        xOff += 3.45
        yOff += 3.975
    elif (xBox == 84):
        xOff += 4.0875
        yOff += 4.8625
    elif (xBox == 96):
        xOff += 6.13125
        yOff += 7.29375
    return xOff, yOff
```

```
96  def predictError3(current_x_off, current_y_off, xBox, yBox):
97      # Error caused by the final turning 90 degrees to face the box
98      # Calculate actual final position of the robot
99      current_x_off += X_errorOf1turn90
100     current_y_off += Y_errorOf1turn90
101     actual_x = xBox + current_x_off
102     actual_y = yBox + current_y_off
103     return actual_x, actual_y
```

Determining Approach to Build Robot (Ideation and Decision Process):

Decision matrix for moving mechanism (front wheel):

Rating: Out of 3		**Option 1**	**Option 2**	**Option 3**
Criteria	**Weight**	One front wheel	Baller	Two front wheels
Accuracy when operating	5	1	3	2
Ease of production/use	4	2	3	1
Cost efficiency	3	3	1	2
Robot's durability	2	1	3	2
Energy efficiency	1	3	2	1
	Total	27	**38**	25

Decision matrix for turning mechanism:

Rating: Out of 3		**Option 1**	**Option 2**	**Option 3**
Criteria	**Weight**	Gyro sensor	Input angular speed and calculate time of rotation	Input time of rotation and calculate motors' speed
Accuracy when operating	5	3	2	1
Ease of production/use	4	3	2	1
Cost efficiency	3	1	3	2
Robot's durability	2	2	3	1
Energy efficiency	1	2	1	3
	Total	**36**	34	20

Decision matrix for lifting mechanism:

Rating: Out of 3		**Option 1**	**Option 2**	**Option 3**
Criteria	**Weight**	Rotating joints (lifting arms)	Linkage	Forklift
Accuracy when operating	5	3	1	2
Ease of production/use	4	2	1	3
Cost efficiency	3	1	3	2
Robot's durability	2	1	2	3
Energy efficiency	1	2	3	1
	Total	30	25	**35**

Decision matrix for barcode scanning mechanism:

Rating: Out of 2		**Option 1**	**Option 2**
Criteria	**Weight**	Color sensor	Infrared sensor
Accuracy when operating	5	2	1
Ease of production/use	4	2	1
Cost efficiency	3	1	2
Robot's durability	2	1	2
Energy efficiency	1	2	1
	Total	**25**	20

Morphological chart:

Functions	**Option 1**	**Option 2**	**Option 3**
Picking up objects	Rotating joints (Lift arms)	Linkage	Forklift
Front wheel structure	One front wheel	Baller	Two front wheels
Turning mechanism	Gyro sensor	Input angular speed and calculate time of rotation	Input time of rotation, and calculate motors' speed
Barcode scanning	Color sensor	Infrared sensor	

A description of the chosen concept based on the morphological chart:

The expected robot design will have two back wheels and one baller at the front, which keeps the robot from bouncing when driving or rotating compared to using front wheels. The lifting mechanism features a forklift that will go under the handle of the box to lift it up smoothly. This will reduce the chances of the box falling out of the robot's arms better than using the rotating joints or linkage. The Gyro sensor is employed in the robot's rotation for the highest accuracy and simplicity of the program. For scanning barcodes, a color sensor is chosen since it is more sensitive compared to the infrared sensor we have. Besides, its ability to detect different colors will make the scanning process more flexible.

<u>Function: Picking up & dropping-off the box:</u>

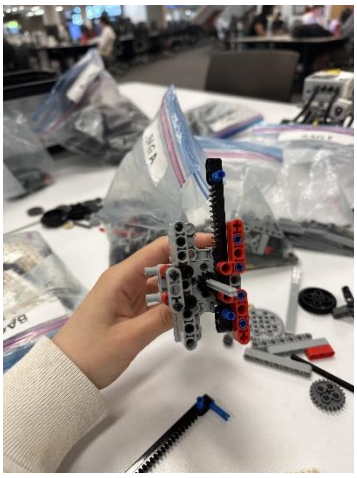
Mechanism: Forklift

1. Empathize: warehouse workers, responsible for tasks such as inventory management and order fulfillment, would be the main users of forklifts. Users emphasize the importance of a forklift attachment that is easy to operate, precise in its movements, and capable of handling a variety of box sizes and weights.
2. Define: A box at a given location needs to be transported to another location. The forklift should be able to go under the box handle and successfully lift the box & carry it to the drop off location.
3. Ideate:

- Forklift should utilize the medium motor
- Needs to be centered in the front of the robot
- Should lower to a maximum of 6 ½ inches above the ground

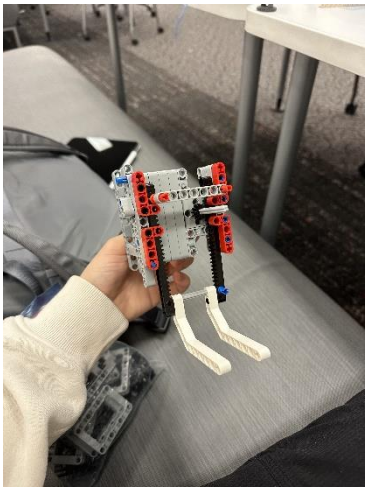
4. Prototype

Basis of mechanism:

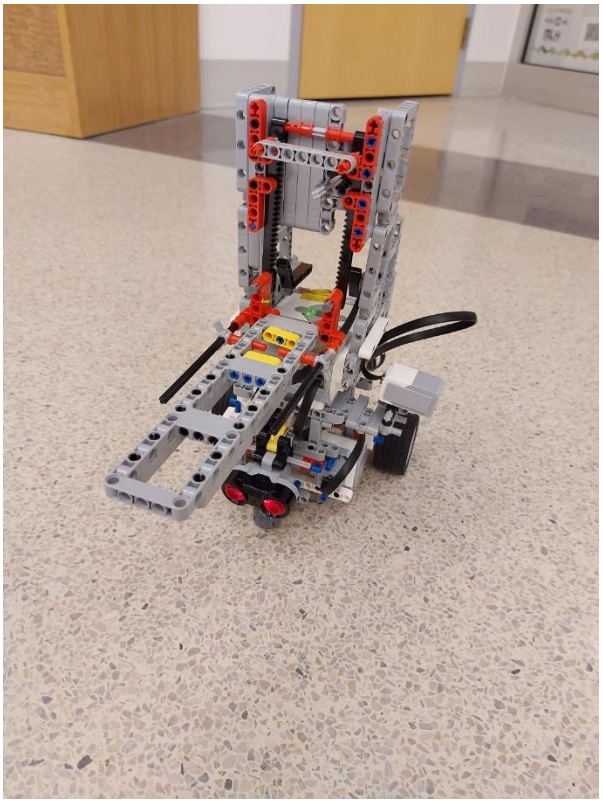


Small gear moves the black treads, which is held in place by the red pieces

First Prototype:



Final Prototype:



5. Test: Final prototype succeeded in the testing plans for lifting mechanism outlined in this notebook.
6. Implement: Forklift design is communicated and agreed upon in the team and shown in the subtask video; to be implemented in final demo.

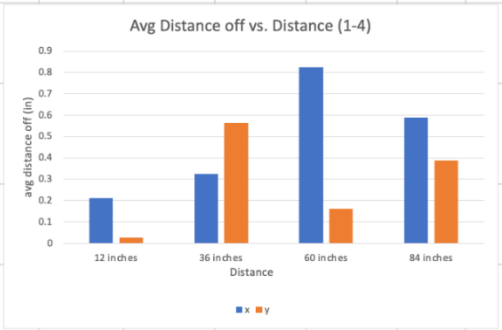
TEST PLAN DATA

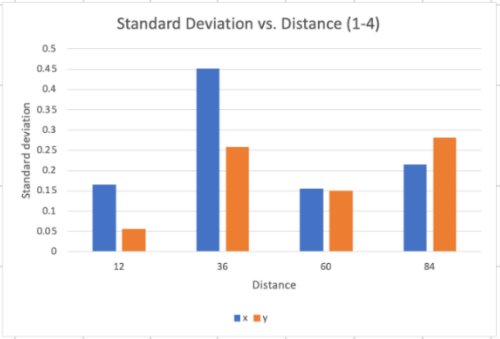
1. Locomotion Tests

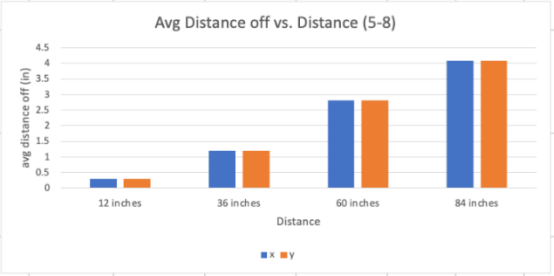
NOTE: Your robot should be capable of holding a box.

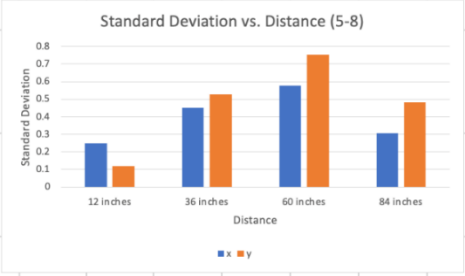
Straight line

Test Number	Criteria / specification	Description of test procedure	Data to be collected	Summary of Results
1	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 12 inches 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 averages versus distance traveled on a chart with results from Test Numbers 1 through 4. Plot the standard deviations versus distance traveled for Test Numbers 1 through 4. Model the distance away from the expected locations.
2	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 36 inches 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	
3	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 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	
4	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 84 inches 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				
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% - 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 12 inches 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 averages versus distance traveled on a chart with results from Test Numbers 1 through 4. Plot the standard deviations versus distance traveled for Test Numbers 5 through 8. Model the distance away from the expected locations.
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% - 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 24 inches 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% - 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 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% - 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 96 inches 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% - 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 96 inches 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

Results (It is okay to put the Results on a Separate Sheet within the same workbook)

Test	Distance away from expected - x (in inches)	Distance away from expected - y (in inches)	
1	0	0	
1	0.125	0.125	
1	0.125	0	
1	0.375	0	
1	0.4375	0	
2	0.5	0.875	
2	0.125	0.8125	
2	-0.375	0.375	
2	0.375	0.375	
2	1	0.38	
3	1	0	
3	0.75	0.125	
3	0.875	0.0625	
3	0.5625	0.375	
3	0.9375	0.25	
4	0.625	0.125	
4	0.875	0.25	
4	0.375	0.625	
4	0.75	0.75	
4	0.3125	0.1875	
5	0.0625	0.9375	
5	0.125	0.8125	
5	0.1875	0.6875	
5	0.625	1	
5	0.5	0.875	
6	1	2	
6	1.125	1.375	
6	0.875	0.625	
6	1.0625	0.9375	
6	2	1.5	
7	2.125	1.875	
7	2.8125	3.125	
7	3.25	3.0625	
7	2.375	3.5	
7	3.5	3.875	
8	3.625	4.1875	
8	4.25	4.875	
8	4.375	5.25	
8	3.9375	4.625	
8	4.25	5.375	

X distance off test 1-4				
	12 inches	36 inches	60 inches	84 inches
1	0	0.5	1	0.625
2	0.125	0.125	0.75	0.875
3	0.125	-0.375	0.875	0.375
4	0.375	0.375	0.5625	0.75
5	0.4375	1	0.9375	0.3125
average:	0.2125	0.325	0.825	0.5875
STDEV:	0.16583124	0.451386752	0.155120921	0.215058132

X distance off test 5-8	12 inches	36 inches	60 inches	84 inches
1	0.0625	1	2.125	3.625
2	0.125	1.125	2.8125	4.25
3	0.1875	0.875	3.25	4.375
4	0.625	1.0625	2.375	3.9375
5	0.5	2	3.5	4.25
averge:	0.3	1.2125	2.8125	4.0875
STDEV:	0.247645159	0.449826355	0.576221529	0.304907773

Y distance off test 1-4	12 inches	36 inches	60 inches	84 inches
1	0	0.875	0	0.125
2	0.125	0.8125	0.125	0.25
3	0	0.375	0.0625	0.625
4	0	0.375	0.375	0.75
5	0	0.38	0.25	0.1875
averge:	0.025	0.5625	0.1625	0.3875
STDEV:	0.055901699	0.257694102	0.150519932	0.280902563

Y distance off test 5-8	12 inches	36 inches	60 inches	84 inches
1	0.9375	2	1.875	4.1875
2	0.8125	1.375	3.125	4.875
3	0.6875	0.625	3.0625	5.25
4	1	0.9375	3.5	4.625
5	0.875	1.5	3.875	5.375
averge:	0.8625	1.2875	3.0875	4.8625
STDEV:	0.12022115	0.529593004	0.752080448	0.480884602

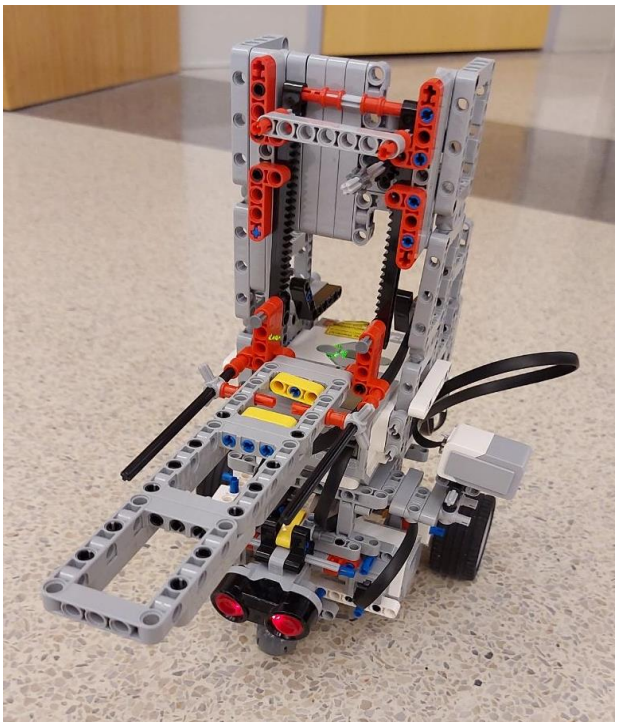
2.

Test Plan

Scanning Barcode Test	Criteria/Specification	Description of Test Procedure	Data to be collected	Summary of Results
1	Know what barcode is being processed based on horizonal half inch margins with some margins filled in with black	Determine surface to test on. Program robot to read barcode with the first margin filled in. If it's correct it it will pick it up. If it's incorrect robot is programmed to backup 5 inches. Repeat 5 times.	Determine if the robot does the correct action	For each test mark what type of barcode worked and which didn't.
2	Know what barcode is being processed based on horizonal half inch margins with some margins filled in with black	Determine surface to test on. Program robot to read barcode with the second margin filled in. If it's correct it it will pick it up. If it's incorrect robot is programmed to backup 5 inches. Repeat 5 times.	Determine if the robot does the correct action	For each test mark what type of barcode worked and which didn't.
3	Know what barcode is being processed based on horizonal half inch margins with some margins filled in with black	Determine surface to test on. Program robot to read barcode with the third margin filled in. If it's correct it it will pick it up. If it's incorrect robot is programmed to backup 5 inches. Repeat 5 times.	Determine if the robot does the correct action	For each test mark what type of barcode worked and which didn't.
4	Know what barcode is being processed based on horizonal half inch margins with some margins filled in with black	Determine surface to test on. Program robot to read barcode with the fourth margin filled in. If it's correct it it will pick it up. If it's incorrect robot is programmed to backup 5 inches. Repeat 5 times.	Determine if the robot does the correct action	For each test mark what type of barcode worked and which didn't.
Lift Object/Drop Off				
1	Robot is able to pick up a 200 gram cardboard 6 x 4 x 6 inch cardboard box with a metal handle and deliver it to the correct location.	Determine surface to test on. Program robot to go 10 inches straight. Program robot to pick up the box. Program robot to turn 90 degrees and bring the box 10 inches to the right. Repeat 5 times.	Determine if the robot does the correct action. Measure how far distance is away from expected x and y.	Calculate the average distance the x and y were off. Calculate the standard deviations of how far off the robot was. Model the distance away from the expected locations using a scatterplot.
2	Robot is able to pick up a 200 gram cardboard 6 x 4 x 6 inch cardboard box with a metal handle and deliver it to the correct location.	Determine surface to test on. Program robot to go 10 inches straight. Program robot to pick up the box. Program robot to turn 135 degrees and bring the box 10 inches to the right. Repeat 5 times.	Determine if the robot does the correct action. Measure how far distance is away from expected x and y.	Calculate the average distance the x and y were off. Calculate the standard deviations of how far off the robot was. Model the distance away from the expected locations using a scatterplot.
3	Robot is able to pick up a 200 gram cardboard 6 x 4 x 6 inch cardboard box with a metal handle and deliver it to the correct location.	Determine surface to test on. Program robot to go 10 inches straight. Program robot to pick up the box. Program robot to turn 180 degrees and bring the box 10 inches to the right. Repeat 5 times.	Determine if the robot does the correct action. Measure how far distance is away from expected x and y.	Calculate the average distance the x and y were off. Calculate the standard deviations of how far off the robot was. Model the distance away from the expected locations using a scatterplot.
4	Robot is able to pick up a 200 gram cardboard 6 x 4 x 6 inch cardboard box with a metal handle and deliver it to the correct location.	Determine surface to test on. Program robot to go 10 inches straight. Program robot to pick up the box. Program robot to turn 270 degrees and bring the box 10 inches to the right. Repeat 5 times.	Determine if the robot does the correct action. Measure how far distance is away from expected x and y.	Calculate the average distance the x and y were off. Calculate the standard deviations of how far off the robot was. Model the distance away from the expected locations using a scatterplot.

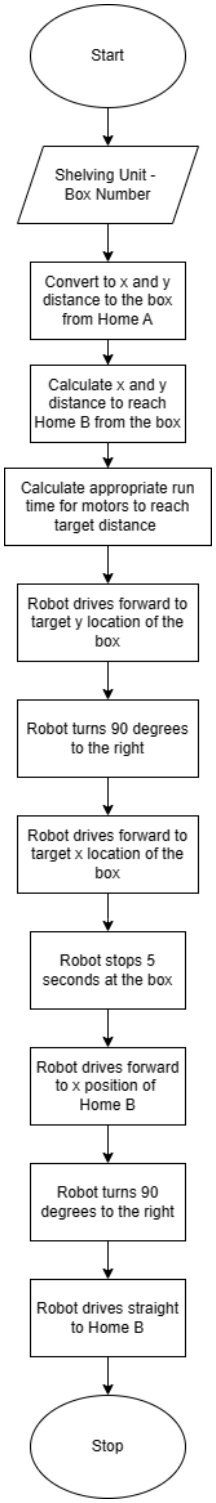
Navigate Around Obstacles				
1	Robot is able to navigate around obstacles and boxes to find it's way to the correct box.	Determine correct surface to test on. Put boxes up in a square format with 10 inches in between each. Put walls around the course. Allow robot to navigate to the end of the boxes without stopping or running into the obstacles. Repeat 5 times.	Measure how many times the robot makes it to the end. Measure how many boxes or obstacles the robot runs into.	Calculate the average number of obstacles the robot hit. Calculate the number of times the robot made it to the end. Make a scatterplot of all four tests to model both the obstacles hit and the amount of the times the robot made it through the course.
2	Robot is able to navigate around obstacles and boxes to find it's way to the correct box.	Determine correct surface to test on. Put boxes up in a triangle format with 10 inches in between each. Put walls around the course. Allow robot to navigate to the end of the boxes without stopping or running into the obstacles. Repeat 5 times.	Measure how many times the robot makes it to the end. Measure how many boxes or obstacles the robot runs into.	Calculate the average number of obstacles the robot hit. Calculate the number of times the robot made it to the end. Make a scatterplot of all four tests to model both the obstacles hit and the amount of the times the robot made it through the course.
3	Robot is able to navigate around obstacles and boxes to find it's way to the correct box.	Determine correct surface to test on. Put boxes up in a scattered format with 10 inches in between each. Put walls around the course. Allow robot to navigate to the end of the boxes without stopping or running into the obstacles. Repeat 5 times.	Measure how many times the robot makes it to the end. Measure how many boxes or obstacles the robot runs into.	Calculate the average number of obstacles the robot hit. Calculate the number of times the robot made it to the end. Make a scatterplot of all four tests to model both the obstacles hit and the amount of the times the robot made it through the course.
4	Robot is able to navigate around obstacles and boxes to find it's way to the correct box.	Determine correct surface to test on. Put boxes up in a scattered format with 10 inches in between each. Put walls around the course. Step in front of the robot to see if it detects a moving object. Allow robot to navigate to the end of the boxes without stopping or running into the obstacles. Repeat 5 times.	Measure how many times the robot makes it to the end. Measure how many boxes or obstacles the robot runs into.	Calculate the average number of obstacles the robot hit. Calculate the number of times the robot made it to the end. Make a scatterplot of all four tests to model both the obstacles hit and the amount of the times the robot made it through the course.

Final Robot Design Summary and Description



After multiple rounds of trial and error, the robot structure reached its final design; tailored to handle box lifting, navigation, and barcode scanning in a warehouse setting. The forklift mechanism, calibrated to achieve a vertical movement range that precisely lowers to 6 5/8 inches, following the standard box dimensions. Originally our color sensor was positioned at the front of the robot but through iterative testing we strategically relocated it to the left side. This adjustment allowed for the robot to scan the barcode without having to turn each time. This enhanced the overall identification process and operational efficiency. While the overall frame, including tires and pivot ball proved functional, there were minor stability issues faced, specifically due to the slight leveling discrepancy. Despite this, our structure remained functional and effective. Our design journey was marked by a user centric approach where feedback and real-world requirements steered our decisions.

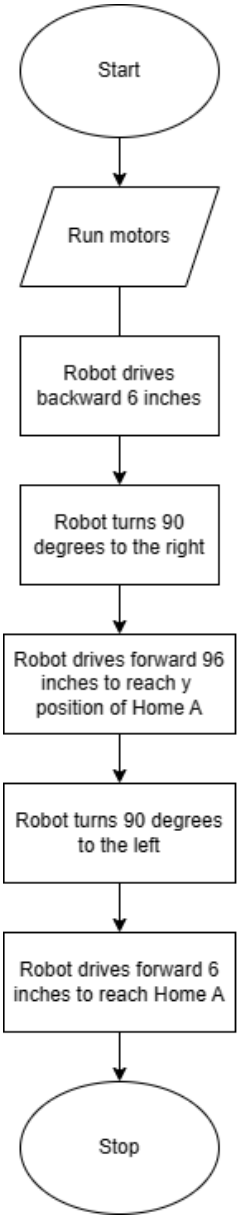
Subtask 1: Navigate to go from Home A to box location then to Home B



```python
#!/usr/bin/env python3

## Imports all Objects and Methods from the motor module
from ev3dev2.motor import *
from time import sleep
from ev3dev2.sensor import *
from ev3dev2.sensor.lego import GyroSensor

## Sets mt Equal to the MoveTank of two motors allowing
## them to run concurrently
mt = MoveTank(OUTPUT_A,OUTPUT_D)
gyro_sensor = GyroSensor(INPUT_4)

# Initialize motors
y_distance = 36
distance_from_edge = 30.5 # Adjust according to box number
x_distance = 6 + distance_from_edge
TURN_SPEED = 20
BASE_SPEED_PERCENT = -25
STRAIGHT_SPEED = 4.9375
run_time1 = y_distance/STRAIGHT_SPEED
run_time2 = x_distance/STRAIGHT_SPEED

# Move to the target y location
mt.on_for_seconds(SpeedPercent(BASE_SPEED_PERCENT),SpeedPercent(BASE_SPEED_PERCENT),run_time1)

# Rotate 90 deg
gyro_sensor.reset()
while abs(gyro_sensor.angle) < 90:
    mt.on(SpeedPercent(-TURN_SPEED),SpeedPercent(TURN_SPEED))

# Move to the target x location
mt.on_for_seconds(SpeedPercent(BASE_SPEED_PERCENT),SpeedPercent(BASE_SPEED_PERCENT),run_time2)

# Stop 5 seconds
time.sleep(5)
mt.on_for_seconds(SpeedPercent(2),SpeedPercent(-2),0.25)
```

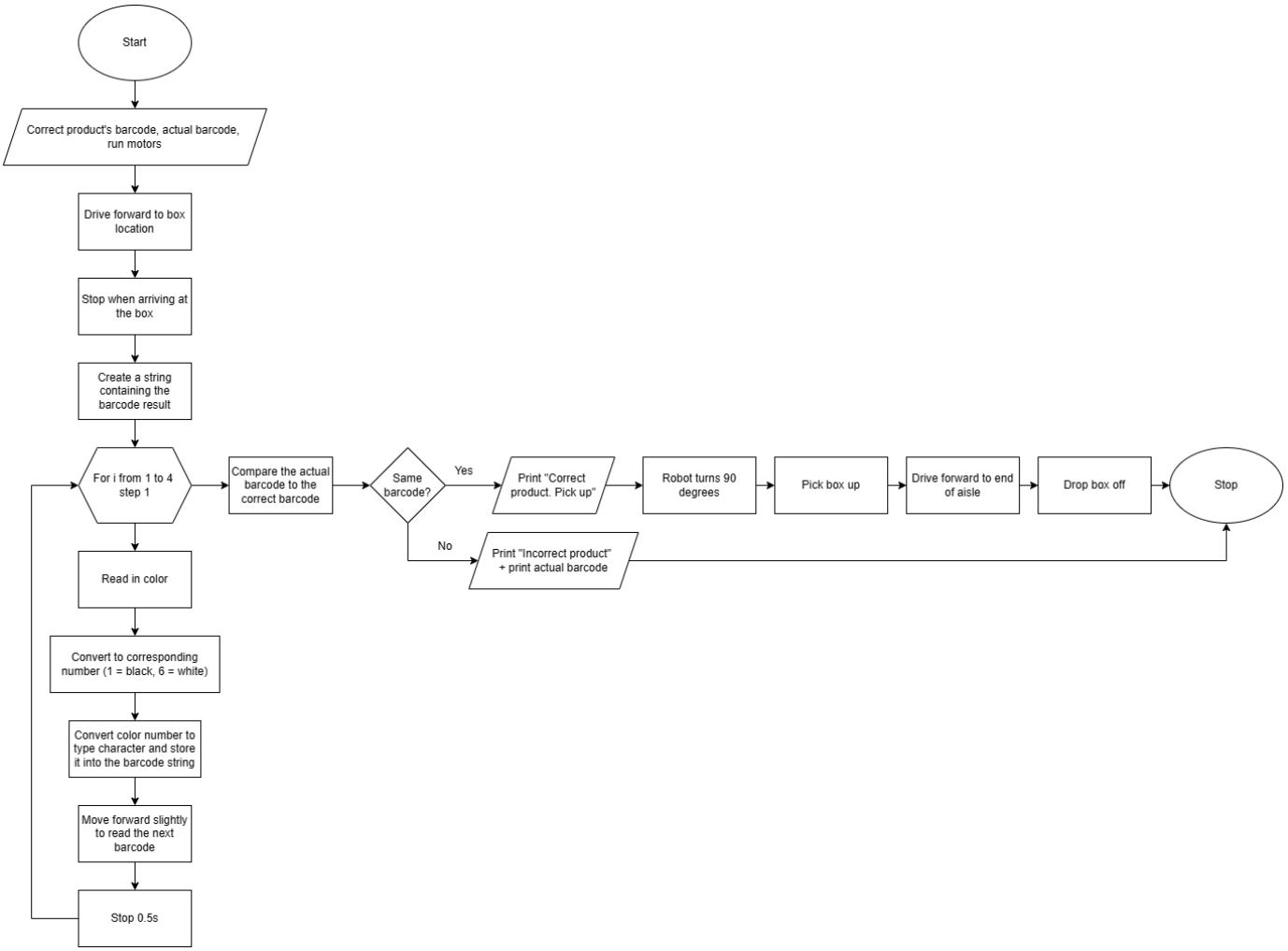
```python
# Move to x of home b
xdistance_to_b = 96 - x_distance
run_time3 = xdistance_to_b/STRAIGHT_SPEED
mt.on_for_seconds(SpeedPercent(BASE_SPEED_PERCENT),SpeedPercent(BASE_SPEED_PERCENT),run_time3)

# Rotate 90
time.sleep(1)
gyro_sensor.reset()
while abs(gyro_sensor.angle) < 90:
    mt.on(SpeedPercent(-TURN_SPEED),SpeedPercent(TURN_SPEED))

# Move to y of home b
ydistance_to_b = 34.5
run_time4 = ydistance_to_b/STRAIGHT_SPEED
mt.on_for_seconds(SpeedPercent(BASE_SPEED_PERCENT),SpeedPercent(BASE_SPEED_PERCENT),run_time4)
```

Subtask 2: Navigate to go from Home B back to Home A



```python
#!/usr/bin/env python3

## Imports all Objects and Methods from the motor module
from ev3dev2.motor import *
from time import sleep
from ev3dev2.sensor import *
from ev3dev2.sensor.lego import GyroSensor

## Sets mt Equal to the MoveTank of two motors allowing
## them to run concurrently
mt = MoveTank(OUTPUT_A,OUTPUT_D)
gyro_sensor = GyroSensor(INPUT_4)

TURN_SPEED = 20
BASE_SPEED_PERCENT = -25
STRAIGHT_SPEED = 4.9375
runtime1 = 8/STRAIGHT_SPEED
runtime2 = 92/STRAIGHT_SPEED

# Drive backward 6 inches
gyro_sensor.calibrate()
mt.on_for_seconds(SpeedPercent(-BASE_SPEED_PERCENT),SpeedPercent(-BASE_SPEED_PERCENT), runtime1)
time.sleep(1)

# Turn 90 degrees
gyro_sensor.reset()
while abs(gyro_sensor.angle) < 90:
    mt.on(SpeedPercent(-TURN_SPEED),SpeedPercent(TURN_SPEED))

# Drive forward 96 inches to reach x location of Home A, stop halfway to adjust itself to move straight
mt.on_for_seconds(SpeedPercent(BASE_SPEED_PERCENT),SpeedPercent(BASE_SPEED_PERCENT), runtime2/2)
time.sleep(0.5)
mt.on_for_seconds(SpeedPercent(2),SpeedPercent(-2),0.5)
mt.on_for_seconds(SpeedPercent(BASE_SPEED_PERCENT),SpeedPercent(BASE_SPEED_PERCENT), runtime2/2)
time.sleep(0.5)

# Turn 90 degrees and drive straight to Home A
gyro_sensor.reset()
while abs(gyro_sensor.angle) < 90:
    mt.on(SpeedPercent(TURN_SPEED),SpeedPercent(-TURN_SPEED))
mt.on_for_seconds(SpeedPercent(BASE_SPEED_PERCENT),SpeedPercent(BASE_SPEED_PERCENT), runtime1)


```

Subtask 3 + 4: Drive to box location, scan barcode and pick up if the box is the correct product / print an error message otherwise



```python
#!/usr/bin/env python3

## Imports all Objects and Methods from the motor module
from ev3dev2.motor import *
from time import sleep
from ev3dev2.sensor import *
from ev3dev2.sensor.lego import GyroSensor
from ev3dev2.sensor.lego import ColorSensor
from ev3dev2.display import Display

## Initialize motors and sensors
mm = Motor(OUTPUT_C)
gyro_sensor = GyroSensor(INPUT_4)
TURN_SPEED = 20
BASE_SPEED_PERCENT = -25
STRAIGHT_SPEED = 4.9375
runtime = 21/STRAIGHT_SPEED
mt = MoveTank(OUTPUT_A,OUTPUT_D)
color_sensor = ColorSensor(INPUT_2)
screen = Display()
barcode_data = []

# Function to drive to box location
def MoveForward():
    mt.on_for_seconds(SpeedPercent(10), SpeedPercent(10), 4.5)

# Function to read barcode
def ScanBarcode():
    strBarcode = ""
    for i in range(4):
        barcode_data.append(color_sensor.color())
        strBarcode += str(barcode_data[i])
        time.sleep(0.5)
    return strBarcode

# Adjust the correct barcode as needed
correct_product = "1166"

# Function to determine if the barcode is correct
def IsCorrectProduct():
    barcode = ScanBarcode()
    if barcode == correct_product:
        return True
    else:
        return False

# Implement functions
barcode = ScanBarcode()
print("Scanned Barcode:", barcode)
screen.clear()

```

```
  # Robot picks up if the box is correct
if IsCorrectProduct():
    print("Correct product detected. Pick up.")
    screen.text_pixels("Correct product detected. Pick up.", x=0, y=0, text_color='black')
    screen.update()
    mm = MediumMotor()

    gyro_sensor.calibrate()

    gyro_sensor.reset()
    while abs(gyro_sensor.angle) < 90:
        mt.on(SpeedPercent(TURN_SPEED),SpeedPercent(-TURN_SPEED))

    mm.on_for_seconds(SpeedPercent(5),7)
    time.sleep(1)

    time.sleep(2)
    mt.on_for_seconds(SpeedPercent(-3),SpeedPercent-(3),1)

    # Pick up box
    mm.on_for_seconds(SpeedPercent(-5),7)
    time.sleep(2)

    # Rotate 90
    gyro_sensor.reset()
    while abs(gyro_sensor.angle) < 75:
        mt.on(SpeedPercent(TURN_SPEED),SpeedPercent(-TURN_SPEED))

    # Move forward to end of aisle
    mt.on_for_seconds(SpeedPercent(-25),SpeedPercent(-25),5)
    time.sleep(1)

    # Drop off box
    mm.on_for_seconds(SpeedPercent(5),6)
```

```
  # Robot displays error message if product is incorrect
else:
    print("Incorrect product detected. This is box {0}.".format(barcode))
    screen.text_pixels("Incorrect product detected. Ignore.", x=0, y=0, text_color='black')
    screen.update()
```