Mech-Claw-Tron: Lego EV3 Claw Machine

Group 8-10

Jana Hamidaldeen - 21080950

Alyssa Medina - 21075735

Yichen Zhao - 21079993

Ania Szczeszyncki

MTE 100 and MTE 121

December 5th, 2023

Summary

The following report will explore all components that were needed to create a Lego EV3 robot in the form of the claw machine: Mech-Claw-Tron. First, the scope of the robot will be explained, touching on the use of all of the components for robot input (such as the sensors and EV3 buttons), as well as the components for robot output, which consist of 4 EV3 motors. The criteria and constraints of the robot will also be mentioned, as it is the guide for all mechanical design which follows shortly after. The mechanical design explanations cover all main components of the machine, including the x, y, and z components, as well as additional parts including the coin slot, prize chute, and main frame. The movement of all mechanical parts are explained with the software design. Overall code is highlighted, with a breakdown of all individual functions that make the claw machine functionable. The verification portion refocuses on the status of all constraints mentioned in the earlier portion. Specifically, if a game on the claw machine meets all requirements. Finally, there is a reflection on the whole process of this project. The management plans throughout the project and recommendations for the future of the design are all touched upon, and conclude this report.

Acknowledgements

To Daniel P. for his assistance during the mechanical design of the Mech-Claw-Tron throughout.

To Forrest, for his assistance during software brainstorming and integration.

And to EGAD, Robot-C, and Engineering Communications, for giving us the tools required to embark on this journey.

Table of Contents

Section	Pages
1. Introduction…….	1
2. Scope……...	2
2.1 Robot Inputs……..	2
2.1.1 Ultrasonic Sensor…….	2
2.1.2 Colour Sensor……...	2
2.1.3 Touch Sensor……	3
2.1.4 Lego EV3 Brick Buttons…….	3
2.2 Robot Outputs……	3
2.2.1 Motor A: X-Axis……	3
2.2.2 Motor B: Claw……	3
2.2.3 Motor C: Z-Axis…….	4
2.2.4 Motor D: Y-Axis…….	4
2.3 Task Completion and Shut-Down Procedure……	4
2.4 Scope Alterations…….	4-5
3. Constraints and Criteria……..	6
4. Mechanical Design and Implementation……	7
4.1 Claw……..	7
4.2 X and Y Component……..	7-8
4.3 Z Component (with ultrasonic) …….	9
4.4 Coin Slot…….……	9-10
4.5 Prize Chute……..	10-12

4.6 Frame……. 12-13

4.7 Overall Design…….. 13

4.7.1 Initial Design…….. 14

4.7.2 Final Design…….. 14-16

5. Software Design and Implementation……. 17

5.1 Overall Code…….. 17

5.2 Breakdown of Functions…… 18-24

5.3 Task List……. 24-25

5.4 How Data is Stored…… 25

5.5 Integration Testing…… 25-26

5.6 Significant Problems…… 26

6. Verification……. 27

6.1 Met Constraints……. 27

6.2 Unmet Constraints……. 27

7. Project Plan……. 28

7.1 Distribution of Tasks……. 28

7.2 Revision of Project Plan…… 28

7.3 Deviations from Project Plan…… 28

8. Conclusions…… 29

9. Recommendations…….. 30

9.1 Mechanical Recommendations……. 30

9.2 Software Recommendations…….. 30-31

Back Matter……..... 32

A. Robot-C Code……. 32-38

B. References…… 39

List of Figures

Figure	Name	Page
1	Ultrasonic detection…….	5
2	Final claw design…….	7
3	X and Y axis……	8
4	Color sensor final design……..	10
5	Touch sensor initial design……..	10
6	No pressure plate: touch sensor…….	11
7	Pressure plate: touch sensor…….	11
8	Enclosed touch sensor……	12
9	Initial frame design……..	13
10	Final frame design……	13
11	Initial overall design…….	14
12	Final overall design…….	15
13	Separated coin slot……...	16
14	Overall code flow chart……	18
15	What button pressed flow chart……..	19
16	X-Axis flow chart…….	20
17	Y-Axis flow chart…….	21
18	Claw flow chart……	22
19	Pickup flow chart…….	23
20	Prize flow chart ……..	23
21	Reset flow chart……	24

List of Tables

Table	Name	Pages
1	Breakdown of functions……..	18-24

1. Introduction

The project's aim was to create a fully functioning robot using the Lego EV3 robotics kit and Lego parts. For this project, the programming software, RobotC, taught in the MTE121 course, is used to program the robot. To do this project, the team used skills implemented from both MTE100 and MTE121 to come up with an objective and implement proper skills learnt. As a result, the team decided to make a Claw Machine to solve the personal objective of creating a nostalgic structure and maintain entertainment using commonly found objects.

In general, a claw machine [1] consists of a mechanical claw able to move around in all 3 axes, enclosed within a frame. Several other features including a prize chute, some sort of coin slot, and translucent enclosures all add to the list of components that make a claw machine what it is. With this in mind, the team implemented designs using Legos to similarly replicate an actual claw machine's motions and mechanical components. Moreover, to simulate the flow of tasks, the Lego EV3 and Robot-C software was implemented to wrap it all together and create a fully functioning claw machine, the Mech-Claw-Tron. The Mech-Claw-Tron's design solves the initial problem by providing entertainment to all users.

2. Scope

The scope of the Mech-Claw-Tron is one built on the basis of entertainment and ultimately, creating a fully functioning claw machine using the Lego EV3 and lego parts. This includes creating a frame that can provide a big enough playing space, allowing users to move the claw in 3 axes, pick up a prize, and return to both the prize chute and original startup position once finished. These tasks occur in quick succession with one another, taking in both user inputs and pre-programmed steps to complete the flow of tasks.

2.1 Robot Inputs

In order to complete tasks, many of the robot's functions rely on parameters inputted by the user, without the use of any files throughout. The following sensors are employed in such a way that measures, detects and uses these values to make decisions in the Mech-Claw-Tron's tasks.

2.1.1 Ultrasonic Sensor

The ultrasonic sensor is attached next to the claw (refer to section 4.3 for more information) and used as a boundary check for the claw machine. Its purpose is to measure the distance between itself and the nearest flat surface. Whether this be the game floor or the beam of a frame, it returns the distance which is then used to determine whether it is safe to proceed with the next step of the game or not.

2.1.2 Colour Sensor

The color sensor detects three different coloured lego pieces to determine what difficulty mode the user would like to play in. The sensor waits 3 seconds after a change in sensor value, allowing for the user to properly place the lego piece directly in front of it and accurately depict the value.

2.1.3 Touch Sensor

The touch sensor is implemented into the hardware with a pressure plate directly above it in the prize chute. Its value detected depends on whether a prize has been won in the game. It remains 0 (unpressed) when there is no prize pushing down on the button, and 1 (pressed) when there is.

2.1.4 Lego EV3 Brick Buttons

The feature that contributes most to first-hand user interaction are the Lego EV3 buttons used to dictate the directions that the claw moves in the x,y, and z component. The up, down, left, and right buttons are translated into integers, each of which are used to enable certain motors in the x and y axis. Then, the enter button is a final single input taken in from the user, indicating to the robot to move onto the pre-programmed list of tasks.

2.2 Robot Outputs

Interaction of the robot with its surrounding environment largely depends on the input taken from the EV3 Lego Buttons. The integers received dictate what each motor does and in what direction (depending on positive and negative values). The following explores the exact outputs of each motor.

2.2.1 Motor A: X-Axis

Motor A is used to move the claw along the x-component with its pulley system (see section 4.2 for more details). Inputs correspond to integers 1 and 3 of the Lego EV3 buttons' translation; moving in the positive direction (forward; relative to user) with an input of 1 and in the negative direction (backwards) with an input of 3.

2.2.2 Motor B: Claw

The claw's movements are dictated by Motor B's inputs. When given a positive motor power, the claw opens for the specified motor encoder limit, and closes when a negative motor power is inputted, allowing for prize pickup and drop off.

2.2.3 Motor C: Z-Axis

Motor C is responsible for the deployment and retraction of the claw in the z-axis. A negative motor power uses the z-components pulley system to lower the claw into the playing area, then retracts back to its original position using a positive motor power.

2.2.4 Motor D: Y-Axis

Similarly to Motor A, Motor D takes inputs - integers 2 and 4 - to dictate which direction the y-component's pulley system moves the claw in. The claw moves to the right (integer 2) with a positive motor power and left (integer 4) with a negative value.

2.3 Task Completion and Shut-Down Procedure

In order to determine when tasks are completed, each step has a specific end-point that allows for software to move onto the subsequent step. These indicators are as follows:

1. Colour sensor detects a value. Prompts for the Enter button to be pressed to move on.
2. User moves claw in x and y axes until the Enter button is pressed or the timer runs out.
3. Claw machine picks up the prize and moves to drop in prize chute, moving on once motor encoder limits are reached.
4. Claw returns back to origin after deploying the prize (if won). Moves onto the next step once motor encoders of motors A and D return to 0.

The system waits 5 seconds and then shuts down once all steps have been completed.

2.4 Scope Alterations

In one case, the ways in which inputs were employed for the x and y axes had to be altered due to the complexity of its code. Initially, the group hoped to have the component move continuously in one direction for however long the user held the EV3 button for. However, the complex use of timers within several functions complicated several aspects of the code, and instead opted for an easier translation to complete the same task. Instead, the press of a button would activate the motor for 0.5 secs. In this case, a continuous press of the button would allow for continuous movement, with momentary pauses every 0.5 secs. Although less smooth, it simplified the process tremendously to match the group's capabilities.

Moreover, the first thought for ultrasonic sensor implementation was to use it to pick up a toy itself. It would detect how far the toy was away from the claw's maximum retraction, and return a distance, using this value to determine how far to drop the claw. However, its implementation and testing was shaky, detecting several different objects at once and returning incorrect values. Therefore, the group opted to use it as a fail-safe instead. It now detects whether a frame beam is found directly below it and stops the claw from deploying is true, ensuring the safety of the Mech-Claw-Tron's frame. See figure 1 for further understanding.

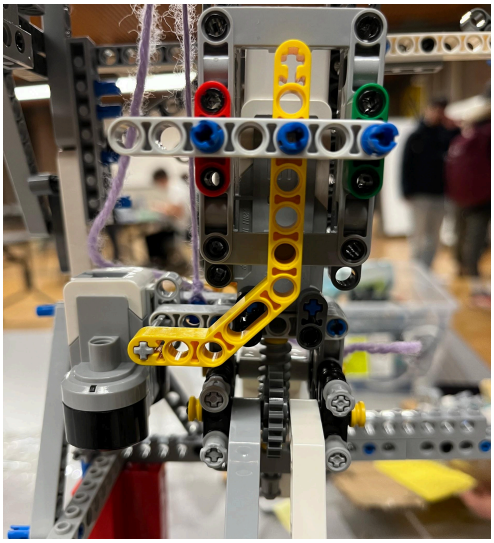

Figure 1: demonstrates an example of the ultrasonic detecting a beam directly underneath it.

3. Constraints and Criteria

The Mech-Claw-Tron's final list of constraints used for demo day consist of actions and tasks that the robot must complete to be a functioning claw machine:

- Instructions must be displayed on EV3 brick for the user after a colour token is inserted.
- The game must start after the user interacts with the EV3 buttons.
- The claw must be able to open and close to pick up and deploy objects.
- The claw's movement must span 3 axes (left and right, back and forth, up and down).
- The claw must travel to the prize chute after being deployed.
- The claw must drop into the prize chute to press the object onto the touch plate.
- At the end of the game, the claw must automatically travel back to the original position diagonal from the prize chute.

The criteria for the robot consists of useful actions and qualities that are desired for best functionality.

- Deploying the object from the claw should be accurate.
- Movement along the x and y axes should be smooth.
- The z-component holding the claw should be upright (perpendicular to the base).
- The structure of the machine should withstand all game movement.
- The claw should be able to hold objects while there is x and y movement (for user enjoyment).

4. Mechanical Design and Implementation

The Mech-Claw-Tron's mechanical components are the largest determining factor for the scope of this project and its software components' performance. The following subsections outline the design decisions and final designs made for this integral part of the project.

4.1 Claw

The claw is a crucial part of the claw machine. This feature used one motor that utilized toothed gears and gear worms. First, the gear worms were attached to a small motor and tested with three different gear sizes before settling for the 24-toothed gear. It was the perfect radius for the claw hands to open and close with ease. Then, different claw arms were tested to determine the perfect size to grip toys. The two options were the bent 53 degrees 4x4 and the 3 by 3.8x7. After going through the pros and cons of both options, the group settled on using the 3x3.5x7 lego piece as it provided more versatile options for toys that can be used in the claw machine. The overall design was inspired using a Lego Claw video (see video [2] in Appendix B), once the building process was down the piece came together smoothly, as shown in figure 2.

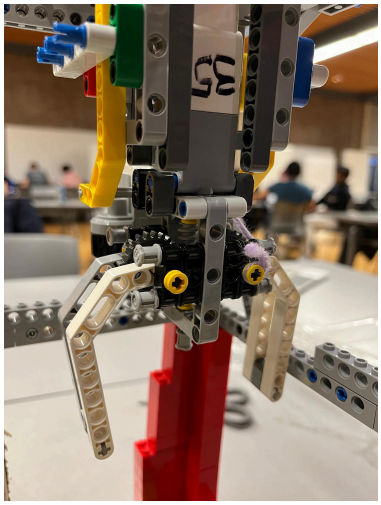

Figure 2: Final claw design

4.2 X and Y Component

The final design of the x and y component is highly different from the ideas in the beginning. Originally, the idea for the x-component consisted of two gears attached to either side

of a motor, which then rides along the axis on a track of rack gears, similar to the video [3] in Appendix B. With the claw attached to a frame that is connected to the motor, the claw would then be able to move back and forth in a precise line, along the x-axis. The y-component consisted of two tracks of rack gears located on either side of the x-component track. This way, the entire x-component frame lies on the tracks of the y-component, so the frame can move back and forth with the use of gears connected to a second motor, also giving the claw the ability back and forth.

However, due to the scarcity of rack gears, the x and y component design changed drastically. Instead of relying on rack gears to lay out the specific path the claw can move, the concept changed to controlling 2-axis movement with a string pulley system. The new design consists of two long beams/frames that are positioned 90 degrees to each other. One beam lies along the x-axis and the other in the y-axis. The shape of both frames allow for precise movement in the direction of the axes. The movement in both axes is allowed by two motors in either direction, that are attached to wheels connected by string. Two wheels that are located on either side of the frames, with a string whose length loops around the wheels with the right amount of tension. As the motor (connected to the same pin as the wheel) turns in the clockwise direction, the wheels spin, pulling the string to move the perpendicular frame to the right. Turning counterclockwise will move the perpendicular frame to the left. There is the same pulley system on the other component that moves only the claw instead of the frame, which allows for movement in both the x and y direction, as seen in figure 3.

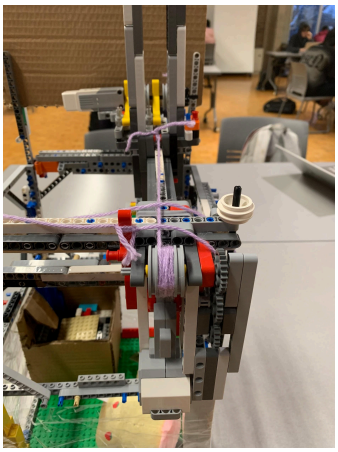

Figure 3: Close-up of the x and y axis pulley system

4.3 Z Component (with ultrasonic)

The z-component was built off the claw and extended up using chassis frames. The chassis frames were connected together to a length that ensured that the claw would reach the floor of the machine. The component then utilized a pulley system method to descend and ascend the claw. To construct the pulley, a motor held two wheels which would rotate when the motor is powered on in order to bring the z component up and down. Both wheels had two separate pieces of yarn, each wheel had the yarn tied around its circumference, then wrapped around the circumference a couple of times, then the other end of the yarn was brought to the chassis frame closest to the bottom of the structure. One end of the yarn was tied to the chassis facing the front of the machine while the other end was tied to the chassis facing the back of the machine. This method was used to ensure that the z component is fully upright and is not relying all of its weight on one side. An ultrasonic sensor was then attached to the side of the z-component, directly facing the floor of the claw machine. This ultrasonic served as a tool to measure the distance the z-component travels.

However, the weight of the claw with the z-axis often caused slanted orientation. The beams in the z-axis would get stuck as the z-axis moved and the z-axis would descend on an angle, making the claw's aim inaccurate. To fix this issue, the team decided to add another string pulley that would go over to the other side to ensure that both sides are being pulled with equal force. The z-axis then was no longer slanted in the y direction, but was still tilted in the x direction due to the motor's weight. To fix this, a counterweight of coins was hung on the other side. These two solutions resulted in an upright z-axis.

4.4 Coin Slot

The purpose of the coin slot is to start the game, which is achieved by using the colour sensor to detect a colour-coded token. When a coloured Lego piece (the token) is placed in a specific slot a set distance away from the sensor, the sensor will detect the specific colour value and start the game by displaying instructions (figure 4). The three available colours correspond with the game difficulty. White, red, and blue are the easy, medium, and hard levels, which give the user 20, 15, and 10 seconds to win a prize, respectively.

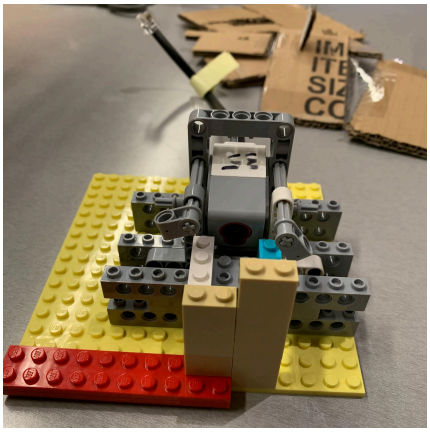

Figure 4: Close-up of the colour sensor

4.5 Prize Chute

Near the end of the game, the claw travels to the prize chute to drop off a prize. At the bottom of the prize chute lies a touch sensor, whose purpose is to detect whether a prize has been won or not. To achieve this, the team's initial design relied solely on the weight of the toys to press down on the touch sensor. The design featured a pressure plate placed slightly above the EV3's touch sensor supported by springs, all surrounded by a box of stacked lego. This way, the weight of the toy dropped into the prize chute would cause the springs to compress and the pressure plate would press on the touch sensor, indicating a prize won. This initial design is seen in figure 5.

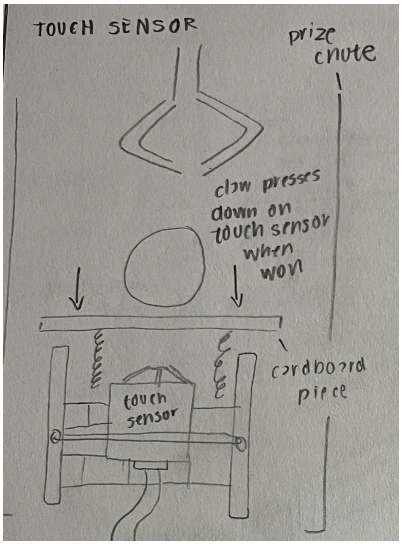

Figure 5 : initial design of the touch sensor

However, physical implementation and testing quickly showed that the toys used in the Mech-Claw-Tron were not heavy enough for the touch sensor to press down on, only successfully detecting a won prize 25% of the time. Therefore, the design was altered to rely more on additional software steps.

This new design (displayed in figures 6 and 7) eliminated the use of springs, and instead used a fastener to install the pressure plate above the touch sensor. Lego wedges were then added on all 4 sides of the plate to secure it in place. The touch sensor would no longer rely on solely weight to change its value. The claw would now be deployed, pressing down just above the pressure plate using motor encoders after every play. That way, if there was a prize in the chute, the claw would press down on the prize and change the sensor value, otherwise pressing on nothing. This logic now increased accuracy to 90%, still occasionally detecting incorrectly due to where the toy landed in the chute. However, this was a major improvement from the initial design.

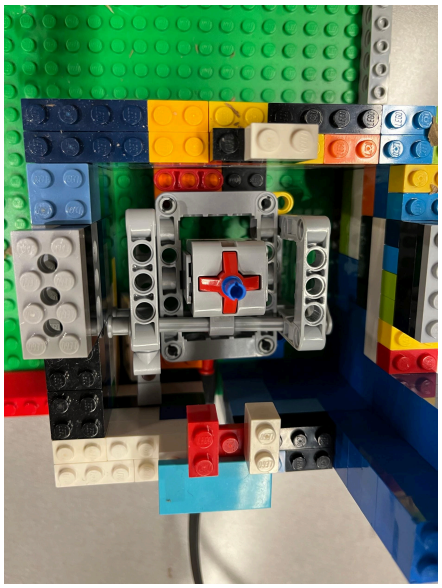

Figure 6: touch sensor without pressure plate.

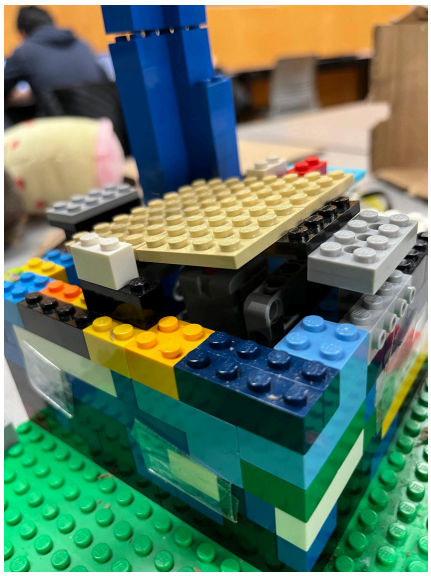

Figure 7: touch sensor finished with wedges and pressure plate implemented.

Finally, to integrate the prize chute into the playing field, cardboard was used to build a box 7 cm higher than the pressure plate's height, ensuring that prizes could not fall out once inside the chute. Height was determined based on the claw's maximum retraction, ensuring that it was low enough for the claw to properly drop a prize in. The final lego design of the touch sensor with the cardboard supports are shown in figure 8.

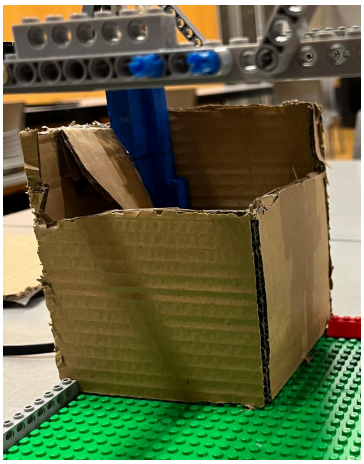

Figure 8: touch sensor enclosed by cardboard prize chute

4.6 Frame

The design of the frame varied greatly throughout the design cycle and was largely dependent on the supplies available to the team. Intending to use aluminum extrusion, the lack thereof forced a complete change during the brainstorming process. In the end, it was decided to use lego pieces instead.

The first design used a 32x32 lego square base as the platform for the Mech-Claw-Tron's playing field. This was then built up on the corners of the base, using typical lego pieces provided by a groupmate, providing a strong base and ensuring structural integrity. From there, beams were used to continue building upwards, accounting for a total height of approximately 40 lego pieces. This initial frame is featured in figure 9 below.

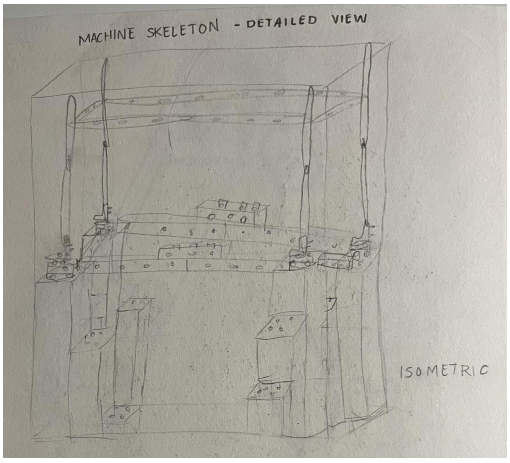

Figure 9 : initial brainstorm of frame using lego parts

As other components were added on, the team discovered that using lego parts for the frame was much more beneficial than aluminum extrusions, as parts could now snap into one another rather than finding a way to install lego into extrusion holes. However, it was quickly evident that the current frame would not support the current weight of the x,y, and z components installed at the top. Therefore, several trusses and layers of beams were added to the final design, effectively increasing the integrity of the build. Moreover, 1x1 lego pieces were lined up along the outer edges of the 32x32 base to limit bending of the platform. The final frame design is found in figure 10.

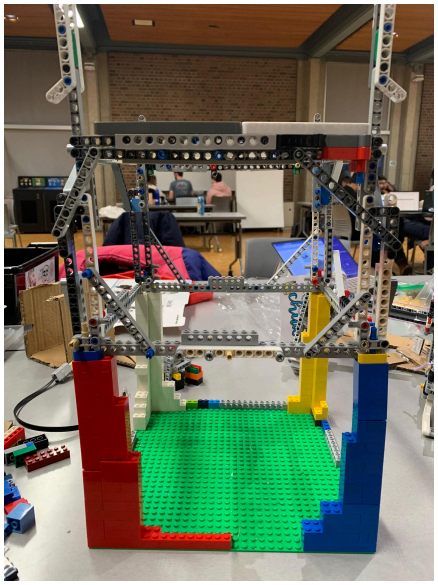

Figure 10: final design of claw machine's frame

4.7 Overall Design

The overall design of the Mech-Claw-Tron continuously evolved as several individual components were altered, different materials were presented, and integration began to take place.

4.7.1 Initial Design

The initial design of the claw machine featured the use of several different materials, featuring a frame completely made of cardboard, prize chute outside of the play area, and x and y components using track gears. Although a good preliminary sketch, this idea lacked several considerations for structural integrity, the implementation of all components together, and wire management that would later be discovered during building of hardware. The preliminary design is shown in figure 11 below.

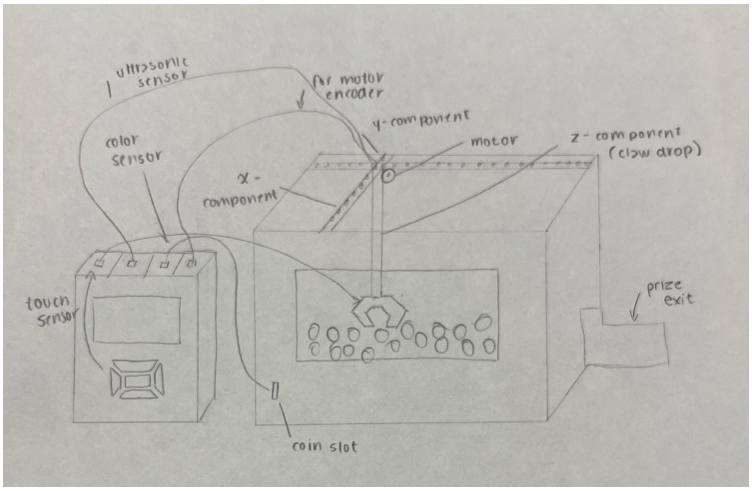

Figure 11: first brainstorm of Mech-Claw-Tron

4.7.2 Final Design

As building progressed and changes were made to subcomponents (as outlined from sections 4.1-4.6), the overall design evolved into a system that maximized all its components and space provided. Figure 12 provides a full view of this final design.

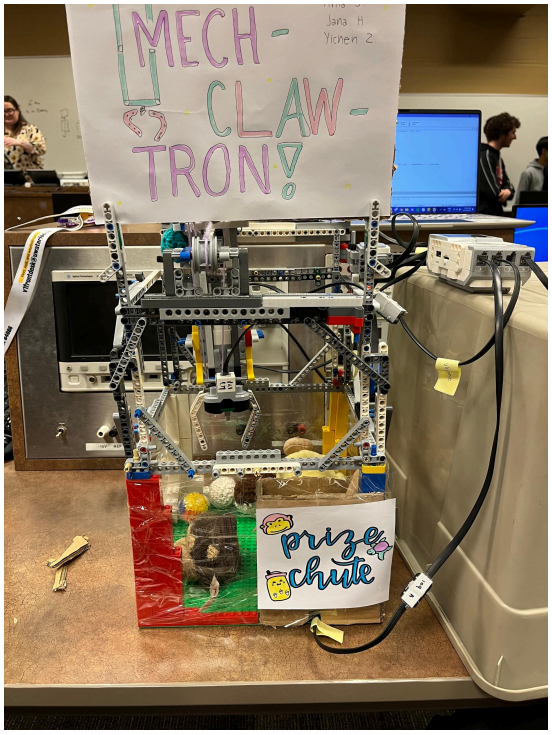

Figure 12: final design of the Mech-Claw-Tron

Integration began with implementing the x,y, and z components together. Several spacing alterations needed to be changed to allow for each axis of the build to naturally “slide” into one another. This was essential, as the string used in all 3 pulley systems axes required a smooth surface in order to operate smoothly. With all 3 moving axes now installed, it could now be implemented atop of the frame.

This installation was approached with caution, as the frame at the time was not able to withstand the weight of all three components on its Lego supports. Following installation of extra beams to support the required weight (refer to section 4.6 for more information) of the three-axis machine, angular connector pieces were employed to snap the x and y component beams into the frames. This limited the unsteadiness of the top of the frame, something that would otherwise be unavailable if aluminum extrusion was used.

Following this, it was decided that the touch sensor would be better implemented within the playing field rather than outside of it due to operational success rates. (see section 4.5 for more information). As seen in figure 12, this was placed at the corner of a lego base, leaving as much space for the playing field as possible. As a result, the coin sensor that was once found

within the claw machine's frame would have to be moved outside to allow for enough claw movement in the game, as seen in figure 13. To account for this, a cardboard box was built around it to secure the collection of tokens while also hiding its inner workings.

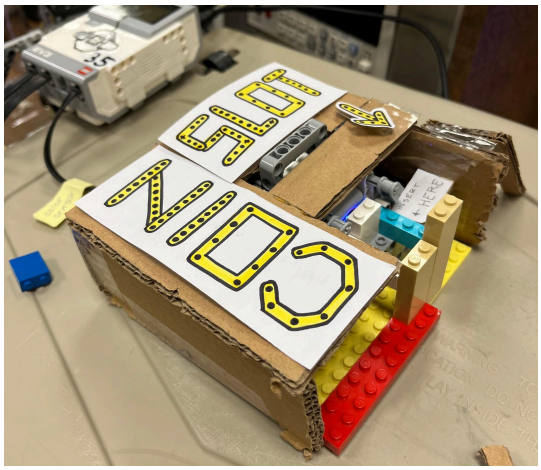

Figure 13: coin slot separated from the claw machine's frame

To wrap it all together, small details were added to the final design in order to truly make it seem like a claw machine. Signs were added to indicate what specific components were, clear tape was added around the base to encapsulate the prizes, and a box was used to elevate the EV3 in order to assist with wire management. Although the final design varied tremendously from its initial brainstorm, the stages of the design cycle it underwent assisted in improving and resulting in the most efficient system possible.

5. Software Design and Implementation

5.1 Overall Code

The program was separated into the following functions, chosen to complete tasks the claw machine must perform and repeat multiple times (refer to the full robot-C code in Appendix A).

- What button pressed - assigns each EV3 button to an integer.
- X-Axis movement - allows the claw to move along the x-axis.
- Y-Axis movement - allows the claw to move along the y-axis.
- Claw movement - controls the claw open and close motions.
- Pickup - moves the claw in the z-direction to pick up an object.
- Prize - returns an object to the prize chute and pushes on the touch sensor.
- Reset - moves the claw to either the original position or to the prize chute.

The overall flow of the program is described as follows (refer to the flow chart in figure 14):

1. The program is first initialized and will ask the user to insert a token.
2. If a coloured token is detected, the display start instructions will appear, giving the user the opportunity to press a button to start. If no token is detected, the message will stay displayed.
3. Once the game starts, a timer starts.
4. If the enter button is pressed or the timer exceeds the indicated time limit, the claw will deploy and x and y movement will be deactivated. Otherwise, the x and y components continue normal function.
5. When the claw is deployed and picks up the object, it will move to the prize chute.
6. The claw will open and drop down to the prize chute to activate the touch sensor.
7. A message will be displayed back to the user depending on whether the touch sensor is pressed from an object.
8. The claw returns to the origin and the game ends.

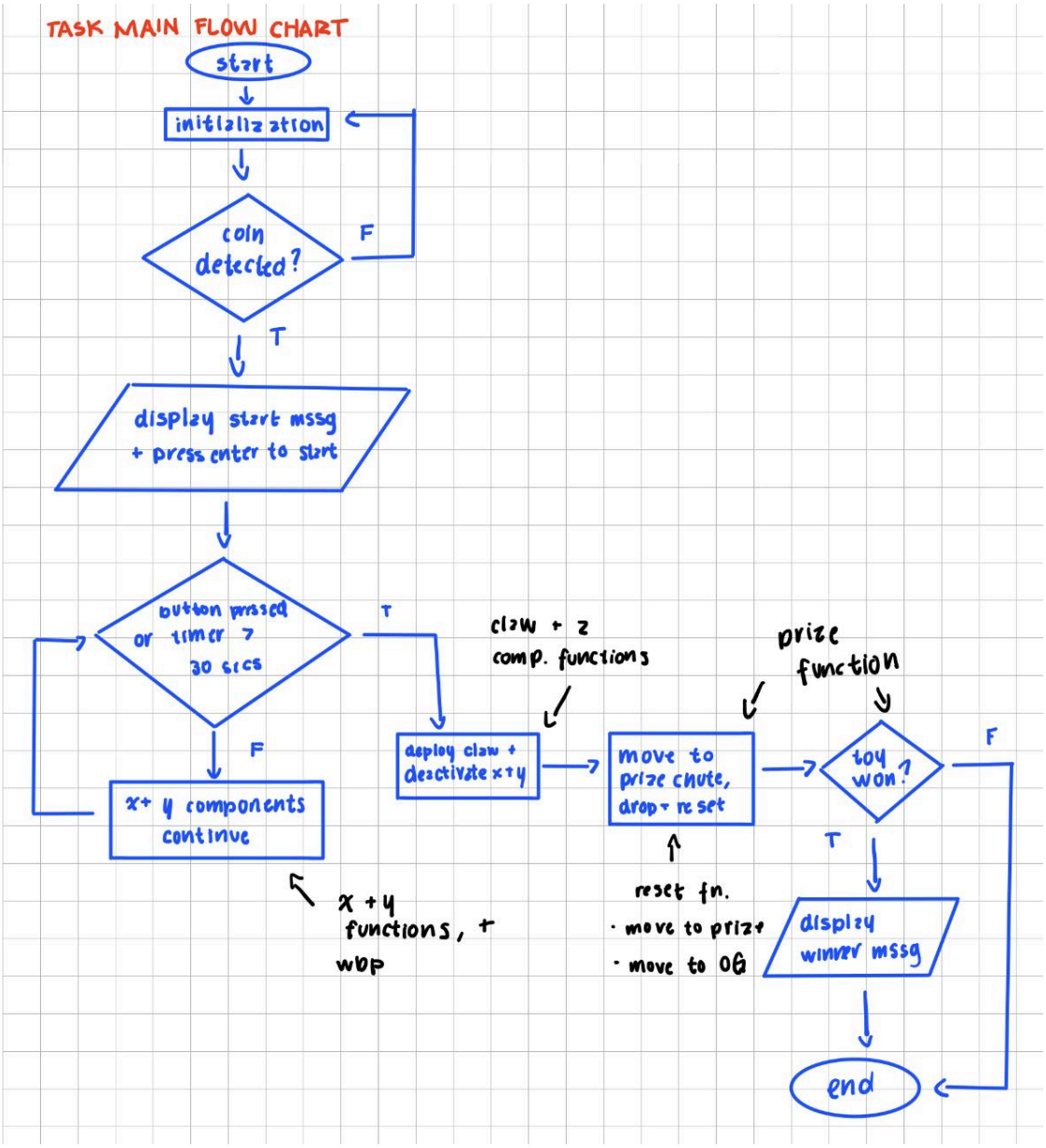

Figure 14: flowchart of overall code

5.2 Breakdown of Functions

Table 1: Breakdown of Functions

Function Description	Return	Parameters
What Button Pressed (wbp) (Written by Alyssa): Assigns an integer to each button on the EV3 and returns the integer value. Right == 2, left == 4, up ==1, down == 3. No button pressed == 0.	int	No parameters

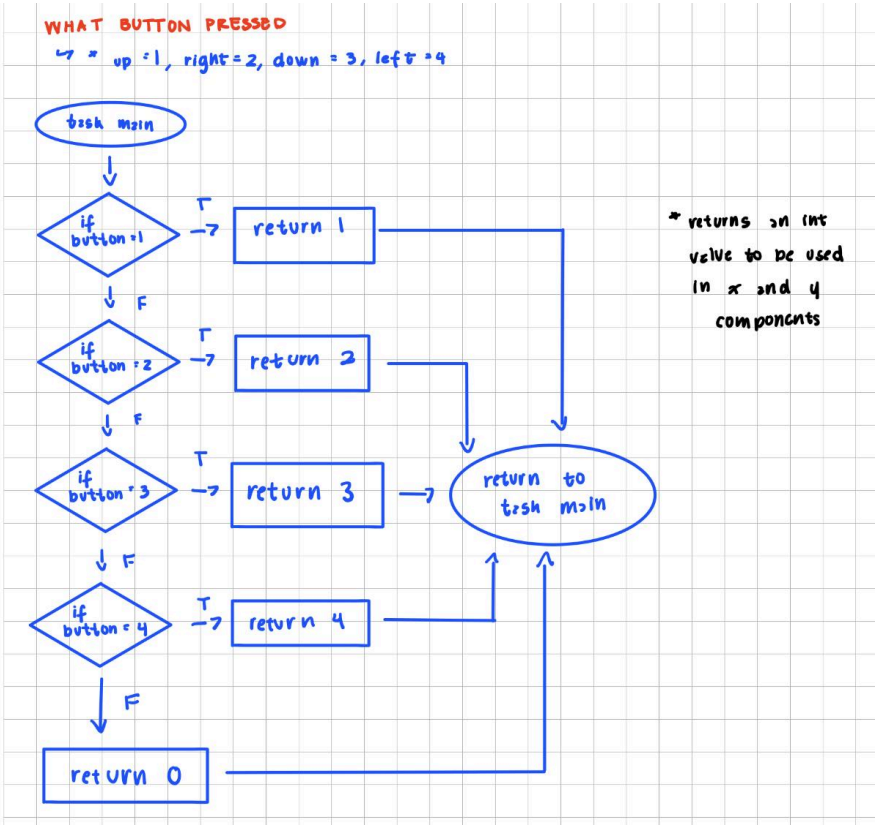

Figure 15: Flowchart for what button pressed function

X-Axis movement (Written by Yichen): The claw can move using the left and right buttons, which takes in the assigned button values from the wbp function. If wbp == 2, the motor power is positive to move the claw to the right. If wbp == 4, the motor power is negative to move the claw to the left. Each press of the button turns the motor for 500 milliseconds (or 0.5 seconds).	void	int wbp

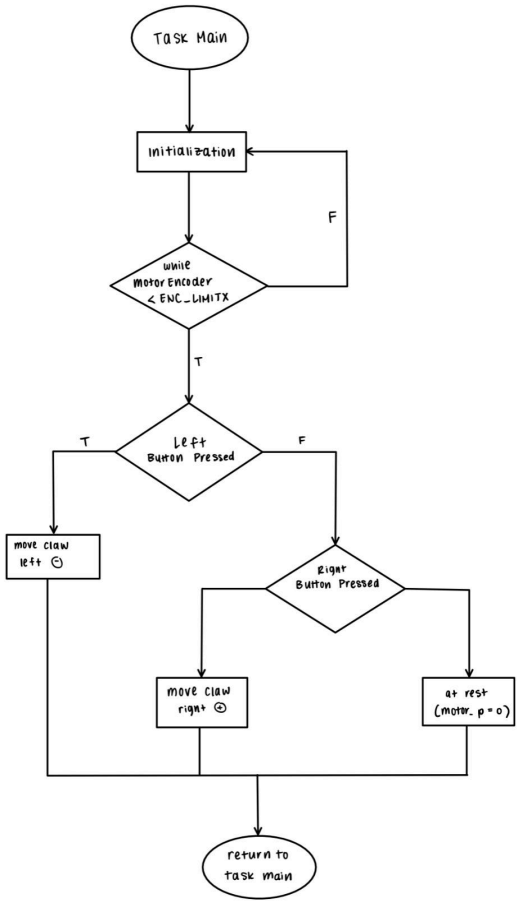

Figure 16: Flowchart for x-axis function

Y-Axis movement (Written by Yichen): The claw can move using the up and down buttons, which takes in the assigned button values from the wbp function. If wbp == 1, the motor power is negative to move the claw backwards. If wbp == 3, the motor power is positive to move the claw forwards. Each press of the button turns the motor for 500 milliseconds (or 0.5 seconds).	void	int wbp

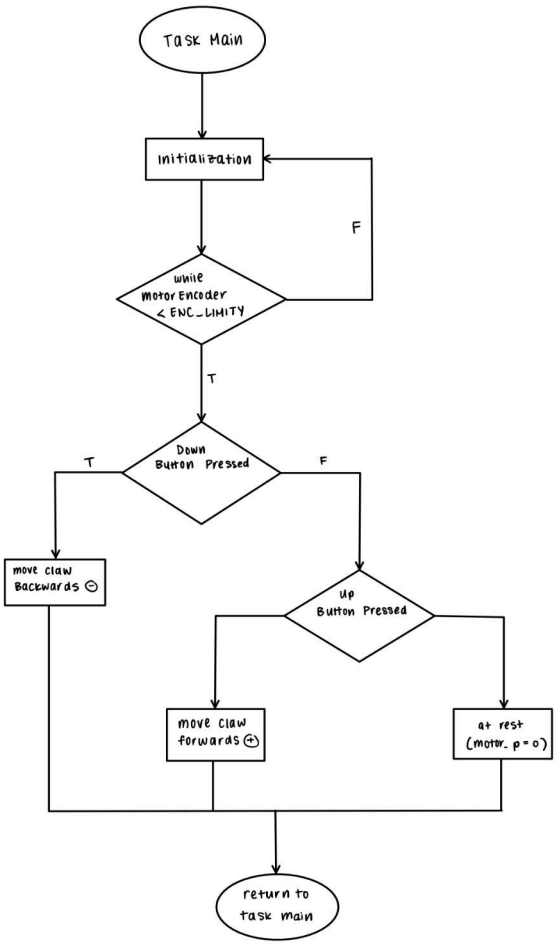

Figure 17: Flowchart for y-axis function

Claw movement (Written by Jana): The claw opens and closes with a motor, where negative motor power closes the claw, and positive power opens it. The boolean direction parameter will decide which motor power to use. The motor encoder controls how much the claw will open, with a motor encoder limit calculated based on the desired number of centimeters to move (integer dist_cm parameter passed).	void	bool direction, int dist_cm

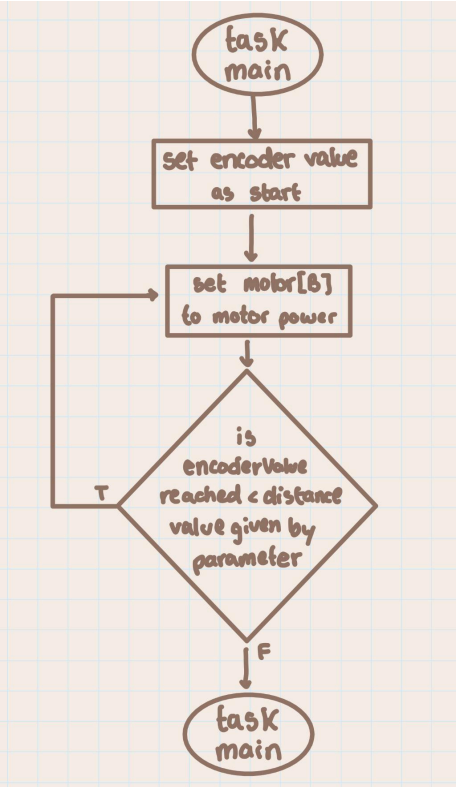

Figure 18: Flowchart for claw movement

Pickup (Written by Jana): An encoder limit and a sensor value limit on the ultrasonic sensor is used to limit the distance for the claw to deploy in the downwards direction. The claw first opens a certain amount enough to pick up the object (using the claw function mentioned above). Then the motor power is negative (to move down) and continues moving while the encoder and ultrasonic limits have not been met. Once it has reached the desired distance, the motor stops and the claw is closed. The claw moves back up while the motor encoder is less than 0.	void	No parameters

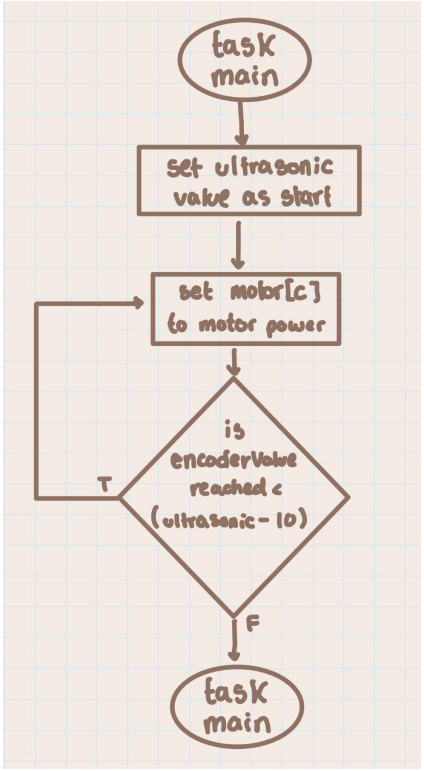

Figure 19: Flowchart for pickup function

Prize (Written by Alyssa): The claw is opened to allow the object to fall, then is closed to a certain amount. The motor power is turned to negative as the claw moves downwards to push down on the object (while the motor encoder is less than an assigned value). The claw then opens to the default amount and if the touch sensor is activated (if an object was won), the program returns win. If there is no object, the program returns a loss. A positive motor power then brings the claw up to the original position (while the motor encoder meets the condition).	int	No parameters

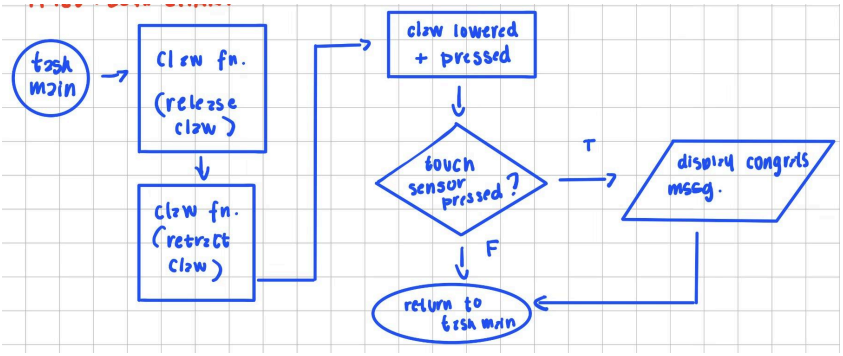

Figure 20: Flowchart for prize function

Reset (Written by Ania): The function takes in the direction parameter to decide where to move. Direction == true will bring the claw to origin, and direction == false moves the claw to the prize chute. If true, the x-axis motor is positive while the motor encoder is greater than 0. When it reaches zero, the y-axis motor does the same thing. This ensures the claw returns to (0,0). If false, the x-axis motor is negative to move the claw in the opposite direction, to the x-coordinate of the prize chute. After it is reached, the y-axis motor will be turned negative until the y-coordinate is reached with the motor encoder.	void	bool direction

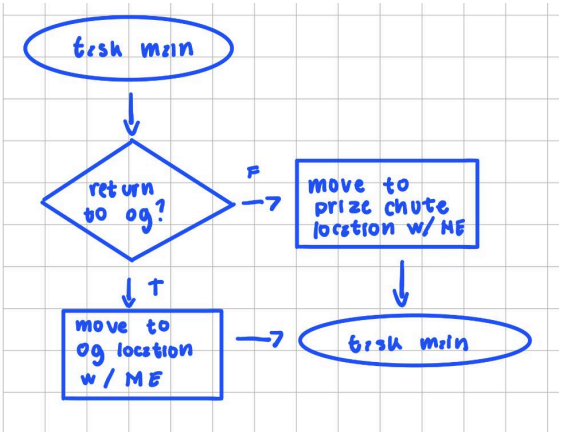

Figure 21: Flowchart for reset function

5.3 Task list

The task list for the robot is sorted into three parts: start-up, normal operation, exceptional operations, and shutdown.

Start Up Tasks:

- Token is inserted, which determines the time limit and displays instructions to start.
- Pressing the enter button starts the main tasks.

Normal Operation:

- X and y components react with user interaction with EV3 buttons (wbp function).
- Z-component and claw picks up a prize.
- Resets the claw to either the start position or the prize chute.
- Prize chute detects whether a prize has been won.

Exceptional Operations:

- Cannot start the game unless a token is inserted; enter button does not work.

- Depending on mode difficulty, if the user takes too long to pick up the prize, the claw automatically drops.
- User interface (right, left, forwards, backwards) does not operate after the enter button is pressed to deploy the claw; limits control.

Shut Down Tasks:

- The system shuts down 5 seconds after the claw has returned to its original position.

5.4 How Data is Stored

The Claw machine did not store data from a text file or in an array, instead it stored real time data in variable buttonStatus using the "what button pressed" function**.** This value is constantly updated whenever the user presses a different button so long as the enter button has not been pressed and the timer has not run out. Otherwise, data is stored using variables in memory that are constantly updated.

5.5 Integration Testing

The method used to test the different components in the code was very detailed and well thought-out. The first checkpoint was writing the main code with the main expectations of the robot's reactions using function prototypes.

The second checkpoint was using the EV3 brick without the code. First, the motor view port was opened on the EV3 and one component was connected at a time to see how it would react when its motor is turned on (ex. Connecting the cable to motorB to check that the claw component can open and close). This was done separately for the following motor driven components: the x-axis, the y axis, the z-axis, and the claw. The sensor port view was then used to check how the touch sensor would react when a toy is dropped from different heights, and the color sensor also used the sensor port view in order to determine what colors are most distinct and best to use for the project.

The third checkpoint is to check each function, to do so, each function is copied into a new RobotC file separately and debugged to ensure that the code runs as expected. This was the most important checkpoint as the motor encoder values were determined and any errors in the code were easier to find considering the length of the code when it is split up.

The fourth checkpoint worked simultaneously with the third, once a function was fully operational and ready, it was copied into a new RobotC file that had all the working functions and components of the code.

The fifth checkpoint required the final edits and changes to be done. Final overall debugging needed to take place to ensure that the overall program can run smoothly when operated.

5.6 Significant Problems

1. What button pressed function not responding

A challenge faced was figuring out how to have the buttons change the X and Y axes movement in real time. Multiple attempts were made to try and have the buttonStatus variable update whenever a different button is pressed. First, the team attempted to use a while loop within the overall playing loop that consistently checks whether a value is changed before exiting the loop and calling the X and Y axes functions. However, after testing, the output did not react as expected and instead the button pressed had its corresponding axis move in one direction non-stop. After more trial and error, the team realized that having the nested while loop was unnecessary, and instead of using the motor encoder for the x and y axis functions, a timer was used. This meant that the motor would be on for 0.5 seconds before checking if the playing loop is still valid, then checking the wbp function and continuing until the playing loop is exited. This method worked as it meant that the buttonStatus variable was changed every 0.5 seconds, allowing the functions to react to the buttons in real time.

2. Unknown wheel radius

The radius of the wheels and gears used in the x,y,z axes and claw were unknown as they were embedded into the hardware and could no longer be measured. This meant that the team could not use the usual “180/PI*radius of wheel” calculation for the motor encoder value in the functions. Instead, the team had to do multiple trial and error runs to determine the encoder limits of each motor for their given functions. This required a lot of time and patience in order to find the exact value to the nearest decimal place, ensuring that each motor went to the exact end point of the axes.

6. Verification

Revision of the constraints presented leading up to demo day revealed several tasks that were altered, met, or completely discarded as follows.

6.1 Met Constraints

- *Instructions must be displayed on EV3 brick for the user after a colour token is inserted*: tested to find different detected colour sensor values and implemented onto display.
- *The game must start after the user interacts with the EV3 buttons*: once any button other than the Enter button (discovered through testing) is pressed, exits the loop
- *The claw must be able to open and close to pick up, hold, and deploy objects*: claw opens and closes using motor encoders, whose values were found after tests and checks.
- *The claw's movement must span 3 axes*: using functions and user input, displayed live movement in all three directions positively and negatively.
- *The claw must travel to the prize chute after being deployed*: met 90% of the time. Failed when motor encoder was not reached due to lack of tension in pulley strings.
- *At the end of the game, the claw must automatically travel back to the original position diagonal from the prize chute:* again, met 90% of the time due to motor encoder issues.

6.2 Unmet Constraints

1. The claw must drop into the prize chute to successfully press the object onto the touch plate.

Seeing as 2 different display messages were outputted to the Lego EV3 console depending on whether or not the touch sensor was pressed, it was clear that the constraint was not met when a prize was in the prize chute, but displayed the wrong message. This was due to a lack of force (gravity and claw's missed force) being employed on the pressure plate. The claw was opened when dropped to cover a wider surface area thereafter, but still remained unreliable.

2. The structure of the machine must withstand all game movement

After several runs of the game, it was common to see the frame (especially at the encoder limit boundaries) fall apart due to stress employed by the motors, indicating a missed constraint. Since the motor encoders would at times reach the end of the frame without reaching its encoder limit (due to pulley system string's tension), this became a problem. Layering and trusses were added to the frame to strengthen itself and withstand all game movement, resulting in longer gaps of time before it would break once again.

7. Project Plan

The robot project was broken into milestones to ensure every component of the project could be completed and on track.

1. Robot ideation (Oct 24 - 30)
2. Physical robot build (Oct 30 - Nov 10)
3. Software ideation (Nov 11 - 13)
4. Software code writing (Nov 14 - 20)
5. Software testing (Nov 21 - 23)
6. Final demo (Nov 24)

7.1 Distribution of Tasks

Every member of the team experienced every stage of the project; everyone contributed to the ideation, design, mechanical, hardware, and software parts of the project. For the mechanical parts, each group member worked on a specific part, and the functions for the program were equally distributed among members. It is important to note that one of our members went MIA and was unresponsive following the mechanical presentation.

7.2 Revision of Project Plan

The software portion of the project took longer than originally expected. To account for the changes, integration testing was done while individual functions were written. This left adequate time for the rest of the original timeline.

7.3 Deviations from Project Plan

The team called a group meeting right after deciding on an idea to pursue for this project. During the group meeting, a list of components the claw machine would include was constructed, then members of the team each picked a component to do research on. A meeting was set for the Monday after the weekend in order to discuss the research that has been done as well as go over the main pseudo code to ensure that the whole team is on the same page. That meeting then consisted of building ideation and starting the actual lego portion. There were consistent meetings between classes and after school to ensure that all the deadlines are met and that we are well ahead of time.

8. Conclusions

In conclusion, this project used multiple components using numerous skills learnt in the courses MTE100 and MTE121. This project went through all stages of the design cycle to achieve the finished desired product. The objective of making a claw machine to solve the problem of creating entertainment was attained, as all constraints to ensure regular game operation were met. Most criteria were also met, as the x and y component provided smooth movement, the claw was firm to hold objects, and the deployment of the claw through the prize function was always accurate, as it landed in the prize chute each time. While the criteria of the frame withstanding game movement was sometimes lost, the Mech-Claw-Tron was successful in bringing nostalgic memories of childhood to its users.

With the combination of both software and mechanical components, a functioning claw machine was finally constructed. It is important to note that although the final design was not perfect, there are several places for improvement that, if given the time, could be maximized and improve overall performance. But until then, it is the errors of the machine that make it an unwinnable entertainment device and truly capture the essence of a claw machine.

9. Recommendations

9.1. Mechanical Recommendations

1. Opting for aluminum extrusion based frame instead of Lego.

Although Lego proved to be much more beneficial to the team when implementing all components of the Mech-Claw-Tron, it was hard to judge whether the benefits outweighed the costs. The reliability of the claw machine's structure remained shaky throughout the process, and in the future, it may be a good idea to strengthen this with aluminum extrusion that is much more stable and secured. Not only would this increase the frame's reliability, but may also improve the performance of all other components when coming together (eg. smooth transitions of x axis)

2. Make the touch sensor's pressure plate out of a heavier material.

As mentioned, a common issue was the misidentification of a won prize or not by the touch sensor in the prize chute. Seeing as a Lego piece was used as the pressure plate, not enough weight was applied on the button to keep it almost pressed down, causing the team to shift our way of thinking in a completely different direction. In the future, using a heavier material to ensure every prize dropped is detected would increase reliability of the feature.

3. Using gears instead of string for the pulley system.

Due to lack of resources, string was used for the pulley systems found in the x,y, and z axes. However, this caused several issues with tension and the subsequent fail of the motor and its motor encoders to accurately slide the claw along the playing field. In the future, gears would be a much more reliable component to use, ensuring precise and smooth plays.

9.2 Software Recommendations

1. Make the "what button pressed (wbp)" function more continuous

Due to time constraints, the what button pressed function implemented into the main code relied on time; moving 0.5 secs for every press and causing pauses every time interval. In the future, employ "whileButtonPress" values and timers to ensure motors move in smooth live time.

2. Allow for two buttons to be pressed at once (integrate into wbp function)

To account for an additional exceptional operation, implement code for diagonal movement when two buttons are pressed at the same time. For example, if both buttons 1 and 2 are pressed (up and right button), then both motors for the x and y axis should be powered to move the claw to the upper-righthand corner.

3. Call the z-component function within the prize function

It would be more efficient in the future to call the z-component function within the prize function, instead of manually writing code for the claw to descend by setting new motor encoder values and opening and closing the claw.

10. Back Matter

Appendix A - Robot-C Code

```
void configureAllSensors();
int wbp();
void xAxis(int wbp);
void yAxis(int wbp);
void clearDisplay();
void pickup();
void claw(bool direction, int dist_cm);
int prize();
void reset(bool direction);

//START OF TASK MAIN

task main()
{
configureAllSensors();

int TIME_LIM = 0;//initialize time limit variable

displayString(3, "Insert token!");
displayString(5, "White = EASY");
displayString(6, "Red = MEDIUM");
displayString(7, "Blue = HARD");
//displays initial instructions

while(SensorValue[S4] == 1)
{}

wait1Msec(5000);
//wait 5 secs after value changes to allow user to properly insert piece
clearDisplay();

if(SensorValue[S4] == 6)
{
      displayString(3, "You've chosen EASY");
      TIME_LIM = 20;
}
else if(SensorValue[S4] == 5)
{
      displayString(3, "You've chosen MEDIUM");
      TIME_LIM = 15;
```

```
}
else if(SensorValue[S4] == 2)
{
	displayString(3, "You've chosen HARD");
	TIME_LIM = 10;
}
//displays mode difficulty based on sensor value

wait1Msec(1000);

	displayString(5, "How to play: ");
	displayString(6, "Right/Left Buttons as normal");
	displayString(7, "Up/Down for Back/Forth");
	displayString(8, "Enter button deploys claw");
	displayString(9, "%d seconds until claw deploys", TIME_LIM);
	displayString(11, "Click any button to start!");
//display next instructions

	while(!getButtonPress(buttonAny))
	{}
	clearDisplay();
	clearTimer(T1);

wait1Msec(500);

//x and y components activated
//while loop exits if enter pressed or time limit runs out
while(!getButtonPress(buttonEnter) && time1[T1]<(TIME_LIM*1000))
{
	displayString(3,"Time left:    ");
	if (time1[T1]%1000)
	{
		int timeleft = TIME_LIM - (time1[T1]/1000);
		displayString(3, "Time left: %d", timeleft);
	}

	int buttonStatus = wbp();
	displayString(5, "The button pressed was %d", buttonStatus);

	if(buttonStatus == 1 || buttonStatus == 3)
			xAxis(buttonStatus);
	else if (buttonStatus == 2 || buttonStatus == 4)
			yAxis(buttonStatus);
```

```
      else if (buttonStatus == 0)
                  motor[motorA]=motor[motorD] = 0;
      //moves in respective directions, using wbp integers
}
//x and y components deactivated

pickup();//picks up prize

clearDisplay();
reset(false);//false = claw goes to prize chute

int win=-1; // initializes win variable
win=prize();

if(win==1)
      displayString(5, "Congrats! Claim your prize");
else
      displayString(5, "Try again #L");

reset(true); // resets claw to origin
claw(1, 3.5); // closes claw to original position
clearDisplay();

wait1Msec(5000);//allows enough time for user to read final message
}
//END OF TASK MAIN - BEGINNING OF FUNCTIONS

void configureAllSensors()
{
      SensorValue[S1] = sensorEV3_Touch;
      wait1Msec(50);
      SensorValue[S2] = sensorEV3_Ultrasonic;
      wait1Msec(50);
      SensorValue[S4] = sensorEV3_Color;
      wait1Msec(50);
      SensorMode[S4] = modeEV3Color_Color;
      wait1Msec(50);
}

//wbp returns integers used in x and y axis
int wbp()
{
      if(getButtonPress(buttonUp))
```

```
            return 1;
      if(getButtonPress(buttonRight))
            return 2;
      if(getButtonPress(buttonDown))
            return 3;
      if(getButtonPress(buttonLeft))
            return 4;

      return 0;
}

//tells motors what number indicates what motor power/direction
void xAxis(int wbp)
{
            if(wbp == 1)
            {
                  motor[motorA] = 10;
                  wait1Msec(500);
            }
            else if(wbp == 3)
            {
                  motor[motorA] = -10;
                  wait1Msec(500);
            }
            motor[motorA]=0;
      }

//tells motors what number indicates what motor power/direction
void yAxis(int wbp)
{
            if (wbp == 2)
            {
                  motor[motorD] = -15;
                  wait1Msec(500);
            }
            else if (wbp == 4)
            {
                  motor[motorD] = 15;
                  wait1Msec(500);
            }
            motor[motorD] = 0;
}
```

```
//clears EV3 display
void clearDisplay()
{
	displayString(1, "                                                ");
	displayString(2, "                                                ");
	displayString(3, "                                                ");
	displayString(4, "                                                ");
	displayString(5, "                                                ");
	displayString(6, "                                                ");
	displayString(7, "                                                ");
	displayString(8, "                                                ");
	displayString(9, "                                                ");
	displayString(10, "                                                ");
	displayString(11, "                                                ");
	displayString(12, "                                                ");
	displayString(13, "                                                ");
	displayString(14, "                                                ");
	displayString(15, "                                                ");
	displayString(16, "                                                ");
	displayString(17, "                                                ");
	displayString(18, "                                                ");
	displayString(19, "                                                ");
	displayString(20, "                                                ");
}

//direction 0 is opening, direction 1 is closing
void claw(bool direction, int dist_cm)
{
	float currentPos = nMotorEncoder[motorB];
	motor[motorB] = (direction*-1 + (!direction)*1)*10;
	const float ENC_LIMIT = (dist_cm*(180/PI*2.75));
	while(abs(nMotorEncoder[motorB]-currentPos) < ENC_LIMIT)
	{}
	motor[motorB] = 0;
}

//picks up prize
void pickup()
{
	//int starts = SensorValue[S3];
	const float ENC_LIMIT = 13*(180/PI);
	const float U_LIMITDOWN = 8;
	nMotorEncoder[motorC]=0;
```

```
      claw(0,5);
      wait1Msec(50);
      motor[motorC] = -10;
      //ultrasonic detects whether safe to deploy claw
      while (abs(nMotorEncoder[motorC])<ENC_LIMIT &&
SensorValue[S2]>U_LIMITDOWN)
      {}
      motor[motorC]=0;
      claw(1,3);
      wait1Msec(50);
      motor[motorC]=10;
      while(abs(nMotorEncoder[motorC])>0)
      {}
      motor[motorC]=0;
      wait1Msec(500);
}

//drops prize in prize chute and presses down on touch sensor
//message displayed whether a prize is won or not
int prize()
{
      int win = -1;

      motor[motorC] = 0;
      claw(0, 5);
      wait1Msec(50);
      claw(1, 6.5);

  nMotorEncoder[motorC] = 0;
  wait1Msec(50);
  motor[motorC] = -30;
  while(abs(nMotorEncoder[motorC])<(5.5)*180/PI)
  {}
      motor[motorC]=0;

      claw(0,3)

      if(SensorValue[S1] == 1)
            win= 1;
      else
            win= 0;

  wait1Msec(1000);
```

```
  motor[motorC]=10;
  while(abs(nMotorEncoder[motorC])>0)
  {}
  motor[motorC]=0;

  return win;
}

//moves claw to extremes
//true = back to origin, false = to prize chute
void reset(bool direction)
{
      if (direction == true)
      {
            motor[motorA] = 10;
            while(abs(nMotorEncoder(motorA)) > 0)
            {}
            motor[motorA] = 0;
            motor[motorD]=15;
            while(abs(nMotorEncoder(motorD)) > 0)
            {}
            motor[motorD]=0;
      }
      else
      {
            motor[motorA] = -10;
            while(abs(nMotorEncoder(motorA)) < (5.5*180/PI))
            {}
            motor[motorA] = 0;
            motor[motorD] = -15;
            while(abs(nMotorEncoder(motorD)) < (11.0*180/PI))
            {}
            motor[motorD] = 0;
      }
}
```

Appendix B - References

[1] Instructables, “Arduino Claw Machine,” *Instructables*, Dec. 03, 2015. https://www.instructables.com/Arduino-Claw-Machine/ (accessed Dec. 5, 2023).

[2] The LEGO Master, “Lego Mindstorms | Basic Claw Attachment |,” YouTube, https://www.youtube.com/watch?v=xZ018UoiboE&ab_channel=TheLEGOMaster (accessed Dec. 5, 2023).

[3] Ord, “Lego A4 Plotter,” YouTube, https://www.youtube.com/watch?v=fGQu90EPVAM&t=14s&ab_channel=ord (accessed Dec. 5, 2023).