# Claw Machine

Group Number: 8-10
Jordan Chow ` | Brandon Goh
Masih Bouriayee | Sam Eskandar

MTE 100/GENE 121
03/12/2019

# Table of Contents

## Table of Figures

## Table of Tables

## Summary

The problem we are aiming to solve is to develop an autonomous claw machine that retrieves objects based off of colour through the implementation of mechatronics principles [1].

For our project, the final design required certain altercations from the previous design. Due to the inaccuracy of the colour sensor, certain criteria had to be modified to determine a successful system. Additionally, the touch sensor's functionality in our system had to be modified to ensure our system worked properly.

## Acknowledgements

We would like to express our thanks towards Ethan Childerhose for allowing our group to use his 3D printer to print parts for our mechatronics system.

# Introduction

The information in this section has been referenced from [1].

## Description of Problem

The problem we are aiming to solve is to develop an autonomous claw machine that retrieves objects based off of colour through the implementation of mechatronics principles. The stakeholders for this project are all the members of the GENE 121 and MTE 100 teaching staff, and all the members of our group.

The GENE 121 and MTE 100 teaching staff are directly impacted by our project because our work is a reflection of the teaching staff's knowledge that they have passed on to us. It is important that our group solves this problem because it is a validation of our learning, allowing the teaching team to verify their teaching success through the quality of our mechatronics system.

The members of our group are included as stakeholders as the design process and construction of the claw machine directly impacts how much we learn from the project. It is important that our group solves this problem as it is a chance to gain practical learning skills that will contribute to our success in future work terms. This project allows us to gain not only various technical skills (mechanical, software, and electrical), but also gain experience in project and time management.

## Background Information

A claw machine is commonly found in arcades, where the user controls a claw in order to grab items and deposit them into a drop off location for the user to retrieve as seen in Figure 1 below. Typically, these claw machines are user driven, where the player controls the movement of the claw through a joystick and presses a button in order to lower the claw. The claw extends far enough to grab the object and begins to slowly bring it over the drop off area to be deposited.

The game lasts for a certain interval of time (commonly 30 seconds), where the user is free to control the claw until the time runs out [2]. The main challenge is that the claw's grip strength is only given an appropriate level of power, a small percentage of the time [3]. This means the majority of the time, the claw will not grip the object tightly enough, resulting in the object slipping out and falling before making it above the drop off zone.

Figure 1: Claw Machine in Arcade [4]

## Scope

### Main Functionality

Our mechatronics system is a claw machine, designed to locate and retrieve objects based on the user's input of an object color. The tasks our robot completes are:

- System start-up
    - System activates when user is within 100*cm* of ultrasonic sensor
    - Display screen activates and prompts the user for a colour to be retrieved
- Main Tasks
    - After the user has confirmed a colour to be retrieved, the y-motor activates
    - Next, the x-motor activates to scan across the row of objects
    - Once the correct object color is read by the color sensor, the claw stops above the object
    - If necessary, the claw machine repositions itself directly above the object (to account for the position of the colour sensor mounting)
    - The z-motor activates, lowering a calculated distance
        - This was changed from our initial design, in which the touch sensor would signify when the object is within the grasp. Although, this feature was removed as we discovered another unique application of the touch sensor for later on.
    - The claw's motor activates in order to tighten and grasp the object
    - The z-motor reverses, allowing the claw to return to the original height
    - The claw then travels directly above the drop-off bin and releases the object into the drop off bin

    - The claw returns back to the starting position
    - Once the touch sensor is pressed, the object is ready to be retrieved from the drop-off bin
- Shutdown
    - Once an object has been retrieved, the program prompts the user if they wish to play again
    - If they wish to play again, the system will restart, if they do not wish to play again, the program ends

## Measuring and Detecting

There were three sensors we incorporated into our design: touch sensor, ultrasonic sensor, and colour sensor.

The ultrasonic sensor was used for the start-up procedure and constantly measured the distance from the sensor. Once a physical object was detected within less than 100*cm* of the sensor, the program started.

The colour sensor was used for the main functionality of our system. The colour sensor must be within a 3 cm range in order for it to measure properly. This restriction was taken into consideration when designing our system and we mounted or claw within this range. The colour sensor is constantly detecting values when the claw is in motion, looking for the particular colour reading the user entered.

Finally, the touch sensor was used near the end of our program and acted as a retrieval button, once the object was placed in the drop-off bin. We attached a cardboard sign on top of the touch sensor to increase surface area and to clarify the purpose of the button. The touch sensor measured if a user pressed it or not.

## Interaction with Environment

Our system interacts with the environment, through the involvement of the user. The user is directly involved with the functionality of the system, because they are able to select which object to retrieve based off of color. The system then prompts the user whether they would like to play again or not, also asking them to retrieve their collected item from the drop off bin. The system on its own is stationary and has little to no interaction with its outside environment other than the user themselves.

## Use of Motors

We used a total of 4 motors in our mechatronics system. The three large motors were used to control the x, y, and z planes whereas the remaining medium motor was used to control the opening and closing of the claw. The y axis motor would power on until it moved a distance of approximately 5 cm. The y axis motor would then stop, and the x axis motor would power on until it moved across the entire frame, coming to a stop at the end of its path. This process would repeat until the color sensor located the desired object. The z axis motor would then power on for a given time (enough time to drop right above the object) and the motor controlling the claw would power on until it has a firm grip on the object. The z axis motor would then raise itself to its original position, where the x and y axis motors would move the claw to just over the drop off bin where the motor controlling the claw would release the object.

### Completed Tasks

The system recognizes when tasks are completed through sensor readings and boolean variables. Tasks can also be visually verified if they are completed through our group watching and seeing if certain tasks are done. For example, we can visually verify if an object is picked up by the claw or not. As for the program's verification, once certain tasks are complete, a boolean variable will be switched from false to true or true to false depending on the variable's design.

### Overall Shutdown Procedure

The shutdown procedure involved the program prompting the user if they wished to play again. This occurs after the system has retrieved an object. If they wish to play again, the system will run again. If they do not want to play again the program will end. Alternatively, if an object is dropped during the retrieval procedure, the claw returns to the starting procedure and ends the program. Finally, if all objects have been retrieved and removed from the system, the claw machine will end.

### Changes in Scope

A change in scope in our project, was the integration of the touch sensor. Originally, during the earlier stages of our project we wanted to implement the touch sensor inside the claw in order to verify when the object has been grabbed. This feature was removed from our final design and the touch sensor was instead inserted at the front of the system. From there, the touch sensor acted as a button, where the user presses it, in order to retrieve their toy. We made this adjustment because it was very difficult to mount the touch sensor on the claw, since it was an additional sensor. This addition would also add extra wiring which complicated the system.

We decided to use three objects in our system, instead of the original amount of four. This decision was made because there was simply not enough room for the additional object in the span of our system. The claw's size was larger than expected, therefore there was not enough room for the final object.

## Constraints and Criteria

### Constraints

Our first constraint was to construct the system with the limited supplies provided to us. This constraint was mainly met, although we 3D printed two tracks pads to allow easy movement in the x and z direction.

This constraint was later discarded, since we decided it was more important to create a functional system, than concern ourselves with the amount of materials we could use. Ultimately, this lead to this constraint being invaluable since as the construction of our system progressed, we put a lower value on the materials used and a larger emphasis on proper functionality.

Our second constraint was to have a system that allows for three axes of motion. This constraint was essential for designing our system. This was one of the most important features of our system we ensured we completed. By having three axes of motion, our claw is able to span across the entire box, lower and raise the claw to grab objects, and drop-off items.

This constraint was extremely helpful when guiding our design, as it was crucial out system met this constraint to be seen as a successful claw machine system.

Our third constraint was ensuring the dimensions of the system does not exceed the maximum wire length. This constraint was very important to allow the claw to span across the system without restrictions.

This constraint was essential for our system and therefore very helpful. Our system would not be able to operate correctly if the wires were not long enough.

Finally, there are two constraints regarding the colour sensor and ultrasonic sensor - colour sensor must be within a range of 3 cm and ultrasonic activates within a range of 5 to 200 cm.

These two constraints were important for planning our system, especially the colour sensor. The colour sensor's extreme restriction required a large amount of planning for how to mount it on the claw and how we designed our objects. As for the ultrasonic sensor, we coded it to activate within 100 cm, to meet this constraint and help activate our system.

### Criteria

The first criteria for our system was to correctly locate the specified item based on the user's input, 15 out of 20 times.

The second criteria for our system was to correctly retrieve this object and release it in the drop-off bin, 15 out of 20 times.

Both of these criteria had to be altered for demo day. After our system was complete and testing was conducted, we found the colour sensor to be incredibly unreliable and inconsistent in its readings. Therefore, we had to adjust the criteria for a working system, since the colour sensor simply failed a large portion of the test runs, even though the code was correct. The new criteria was made to be - the system must correctly locate, retrieve, and drop-off the item at least once out of five trials. This was developed to account for the colour sensor's inaccurate readings.

These two criteria work in conjunction with each other, in order to verify our entire system is functioning properly. Therefore, both of these criteria helped guide our project design and implementation.

## Mechanical Design and Implementation

## Description of Overall Design

The claw machine was designed to be able to move a claw in the x, y and z axes of a set space. Three continuous motors were used to provide motion in these axes, while one servo motor was used to power the claw. The claw spans a space of $20cm_x * 20cm_y * 15cm_z$, in which it can pick up and drop off objects. The Lego EV3 brick was mounted at the front of the system and higher up allowing for the simplest method of wiring the motors and sensors as seen below in Figure 2Figure 2: Top Front View of System. Since most of the motors were located in the upper most part of our design, having the brick near the top of the chassis allowed for shorter wires to be used, and less excess slack. Finally, the target objects were constructed to allow for an efficient pick up by the claw. Assuming the worm gear powering the claws fingers didn’t close enough to grip the objects, the irregular shapes of the objects reduced slippage and allowed for consistent pickups as seen below in Figure 3.

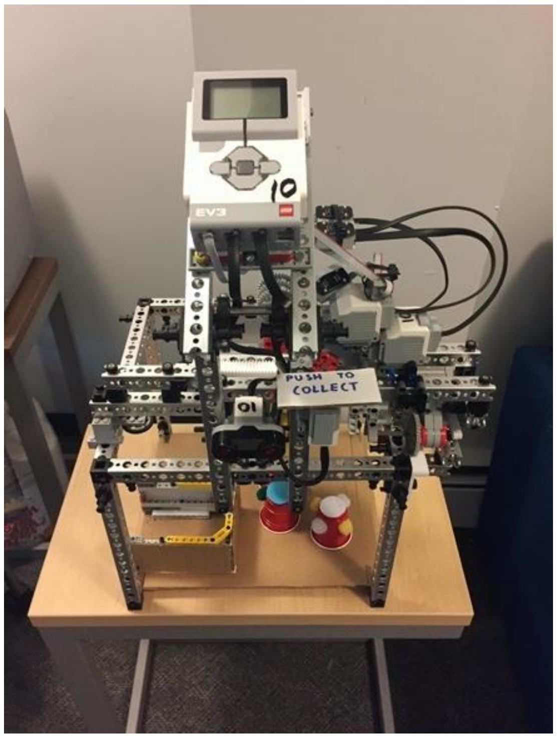


*Figure 2: Top Front View of System*

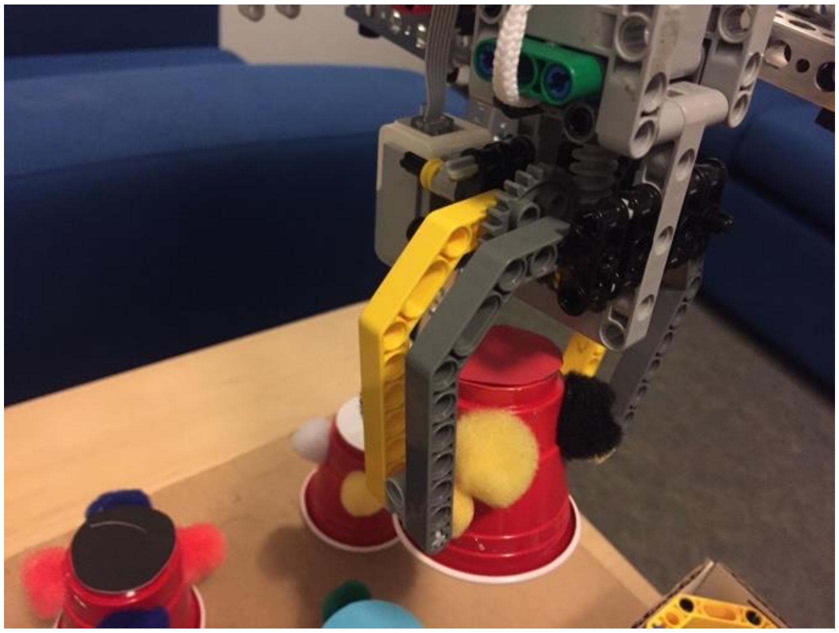


*Figure 3: Claw Design and Pickup*

## Chassis Design

The claw machine chassis was constructed out of Tetrix aluminum square tubes and measured 32$cm_x$ * 28$cm_y$ * 24$cm_z$. This allowed for the maximum space our claw could move in without any interferences. The base of our system was made out of brown cardboard so that there would be no interference with the colour sensor as seen below in Figure 4. As well, our drop off zone had an area of 5$cm_x$ * 10$cm_y$, allowing our objects to consistently land in its designated spot.

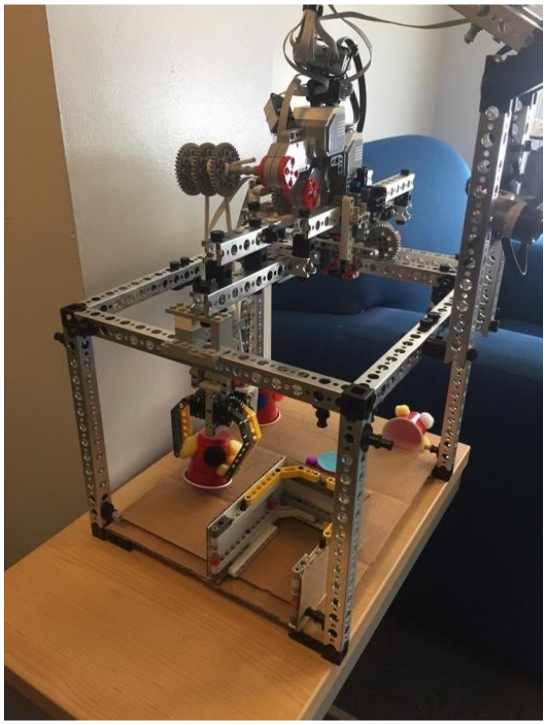


*Figure 4: Isometric View of System*

## Motor Drive Design

For the movement across the x and y axes, instead of the conventional belt/pulley design used in most claw machines, a mechanism was designed that would work better with the LEGO pieces. For the x and y motors, a 60 tooth gear was attached to the motors via a 5*cm* shaft. A gear track based on the 60 tooth gear's spacing and pitch was designed on SolidWorks. The gear track was designed with slots where it would be bolted to the Tetrix frame to allow for adjustability in height. Once completed, the gear tracks were 3D printed and bolted onto the frame as seen below in Figure 5.

The motors and gears were then placed onto the track and sliders were fitted on to maintain a linear motion. For the z axis, the motor was hooked up to a double winch with a coiled-up shoelace attached to the claw. The double winch helped the claw avoid excessive swinging and rotation, making it easier for the claw to detect and pick up the objects. Finally, a servo motor was used to power a worm gear which in turn would rotate to power the claws fingers. The torque provided by the worm gear was powerful enough to grip the objects tight and reduce the chances of any objects slipping off.

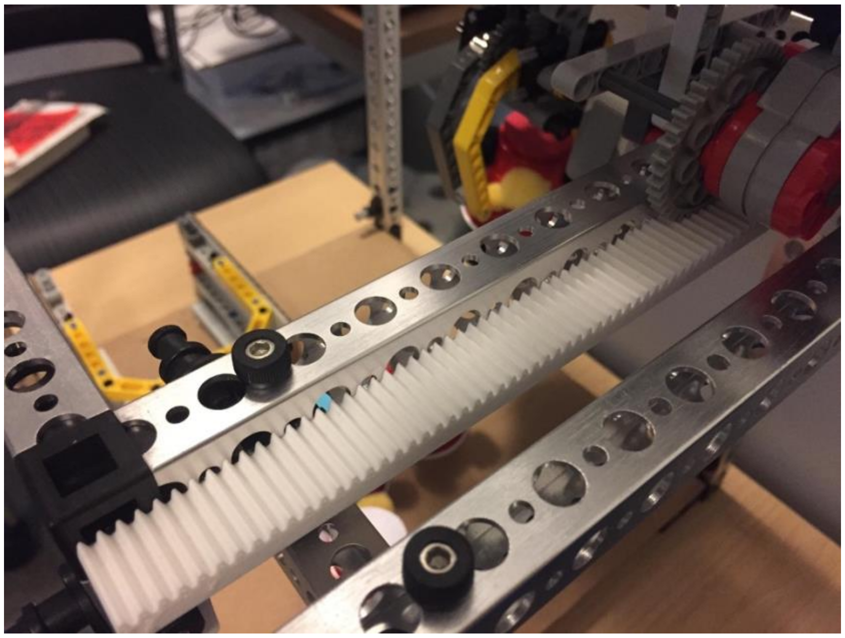


*Figure 5: 3D Printed Gear Track and Motor Drive System*

## Sensor Attachment Design

**Colour Sensor**

The colour sensor is mounted on the side of the claw apparatus, facing downwards. This location proved optimal for our design, as the colour sensor functions within a range of 3 cm. Therefore, mounting the sensor on the claw, very close vertically to the objects was the best design for us. Although, since the sensor was mounted on the side of the claw, we had to do additional calculations for the claw to travel back a certain distance from where the reading was taken in order to grab the object. The colour sensor was used to take readings of the colours of each object as it scanned across the box. Once the correct colour was located, the retrieve procedure would occur.

**Ultrasonic Sensor**

For the ultrasonic sensor, we wanted to mount it somewhere facing towards the user at the front of the machine. We chose a centered location which would read the user's body as they approach the machine. This design was intentional, so the user could walk up to the machine and have it activated from there.

**Touch Sensor**

The touch sensor is located on the front of the claw machine, pointing upwards. It is easily accessible to the user and incorporates a small sign on top to increase surface area and notify the user that they must "push to collect" their item. The touch sensor is used after the item has been placed in the drop-off. Once the object has been dropped off, the user presses the button, allowing them to access and retrieve their toy. This functionality was designed to represent the flap used in classic claw machines that must be moved in order to retrieve a toy. Our touch sensor acts as a button, the user must press before they can remove their item as seen below in Figure 6.

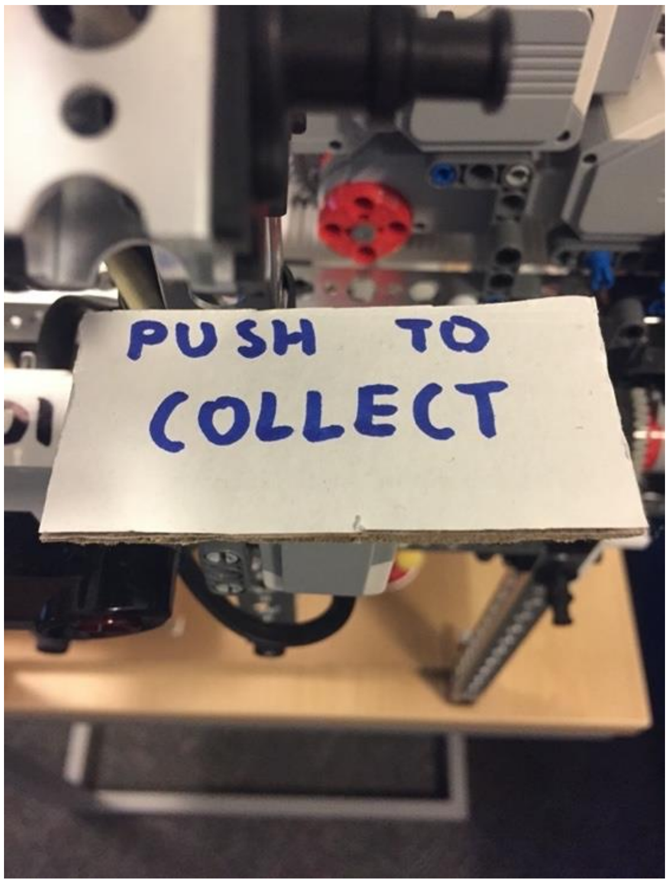


*Figure 6: System Retrieve Button*

## Overall Assembly

The entire system was put assembled slowly, focusing on the main frame first. The frame was constructed using the Tetrix steel beams. Next, the trackpads, gears, and corresponding motors for the claw's x and y movement were assembled. These pieces all worked in conjunction with each other, therefore they were constructed and assembled together. As for the claw, it was built separately and later attached. Once all sensors and motors were mounted, wiring was completed through the use of wire extenders and zip ties to organize the cables. Finally, the drop-off bin was installed, and the objects were placed on the floor of the claw machine. Figure 7 below shows our final sketch of the mechatronics system done in AutoCAD.

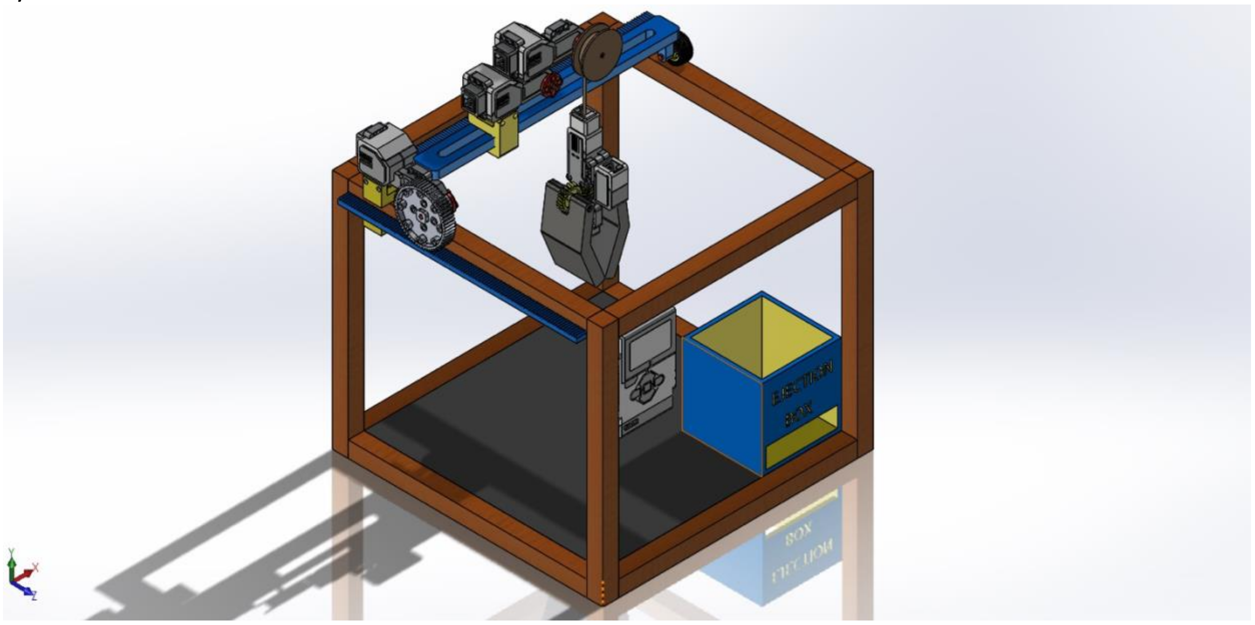


*Figure 7: SolidWorks Sketch of the System*

## Design Decisions and Trade-offs

Given our material constraints, there were a few design options that had to be modified or changed altogether to accommodate for our parts. From our original CAD, we had wanted to build our chassis out of wood to provide a larger space for the claw to span. Instead, we soon realized that the LEGO pieces were made in metric and wood was sold in imperial. The difference in measurement systems would result in a less accurate gear track system which would lead to less accurate object pickups, so we opted to build our frame out of Tetrix beams instead.

Furthermore, another trade-off that had to be made was creating a gear track for the movements across the axes instead of the traditional belt and pulley method used in conventional claw machines. We decided on the gear track design as it was a simpler mechanism to implement on our system. The belt and pulley method would require the purchasing of timing belts, several pulleys to fit those belts, and an accurate positioning of those pulleys or else there would be slack and skipping in the timing belt.

On the other hand, with the gear track mechanism, we only had to CAD two simple gear tracks, get them 3D printed, bolt them onto our frame and run the motors with their gears to obtain our linear motion.

# Software Design and Implementation

## Task List Description and High-Level Software Overview

When first designing the software for our Claw Machine, our group first separated each function of the system. For example, we distinguished the start-up screen actions from the main functionality of the claw, then we separated the claws function of searching for an object from returning it to the drop off bin, and resetting the claws position to the start. By doing this, we were able to clearly identify the functions we needed to write and what each function needed to receive and return.

By breaking up the functionality of the system, the task list was also much easier to create and follow. The task list starts with start-up, ensuring that the user selects a valid colour (must still be in the span of the system). After the user selects a colour, a 1D integer array populated with the integer values relating to the colour readings of the EV3 colour sensor and a 1D boolean array populated with all false values are altered. The boolean array at the index of the colour chosen within the color (integer) array is changed to true, ensuring that the user cannot select the same object more than once.

Regular tasks are next on the task list, this section is essentially the core functionality of the claw machine. The tasks were separated based on the function they relate to. Scanning, stopping above the object, and picking the object up relate to objectPickup, bringing the object to the drop off bin relates to objectTransport, and returning to the original position relates to reset (full calculations are shown in Appendix A). This section of the task list is essential to ensure that the core functionality of the claw machine remained intact throughout testing and demoing.

Unexpected Cases and Shut down procedure came next on the task list and essentially covered that the claw machine handled wrong/invalid inputs and shutdown when the user collected their object (if they did not choose to play again).

## Main and Function Descriptions

Refer to Appendix A for code for each function. Table 1 below, goes into detail about the functions of the code.

*Table 1: Explicit Functions and their Functionality*

| Function Name | Parameters | Return | Functionality |
|---|---|---|---|
| int colourLeft (Jordan) | • int colours[ ]<br>• int coloursIndex | colourChosenNum | Used to essentially iterate through colour options, displaying on the screen the colour that the user can select then resetting and changing the displayed option when the left button is pressed |
| int colourRight (Sam) | • int colours[ ]<br>• int coloursIndex | colourChosenNum | Used to essentially iterate through colour options, displaying on the screen the colour that the user can |

| | | | |
|---|---|---|---|
| | | | select then resetting and changing the displayed option when the right button is pressed |
| int startup (Brandon) | • int colours[ ]<br>• bool coloursPicked[ ] | colourChosenNum<br>OR<br>400 (error indication)<br>OR<br>404 (error indication) | Starts by displaying the colour options. The default colour selected is black. The user can press the left or right button to switch colours and this function will call colourLeft or colourRight to change the display. If the up or down button is pressed, it will prompt the user to use only the left, right and center button. Once the user has pressed enter, it will check if the colour is already chosen from an array, or if all the colours have been chosen already, by returning an integer value of 400 and 404 respectively. If not, it will return the index of the colour chosen which coincides with an array of colour options. |
| void objectPickup | • int scanx<br>• int scany<br>• int powery<br>• int powerx<br>• int colour<br>• bool isThere<br>• int row | n/a | Starts by moving 4.5cm in the y axis at a power of 10% (powery) then stops. Scans in the x axis until it sees the selected colour. Changes isThere to true (for use in main and other functions). If isThere is true, lowers claw to pick up object (aided by timer) then raises the claw. If isThere remains false, displays error message |
| void objectTransport | • int scanx<br>• int scany<br>• float yTogo<br>• float xTogo<br>• bool isThere<br>• int row<br>• int powerx | n/a | Calculates the distance required to travel to the drop off bin using the values scanx and scany from objectPickup.<br>If isThere is true (the object was found) travels to the drop off bin using the calculated values of yTogo and xTogo |
| void reset | • float yTogo<br>• float xTogo | n/a | Returns the claw to the original position using the values of yTogo and xTogo |
| bool playAgain | No parameters | True or false (play again or not) | Based off of selected buttons, returns whether the user should end or if they want to play again |

| | | | |
|---|---|---|---|
| bool gameMode* | No parameters | True or false (race or not) | Based off of selected buttons, returns whether the user wants to play the racing mode or just retrieve the object |
| bool isSuccesful | No parameters | True or false (did the object get returned or not) | Based off of selected buttons, returns whether the user should end or if they want to play again |
| int controls* | No parameters | Returns time taken to retrieve object | User controls the x and y axis movement until they press the enter button, they then control the claw and z axis movement only pressing enter when they retrieve the object |
| main | n/a | n/a | Waits for user to come within 100 cm from the screen. Prompts them for what game mode they want (for demo only autonomous game mode was chosen). Allows user to keep on selecting objects for retrieval until there are no more or the user does not want to play |

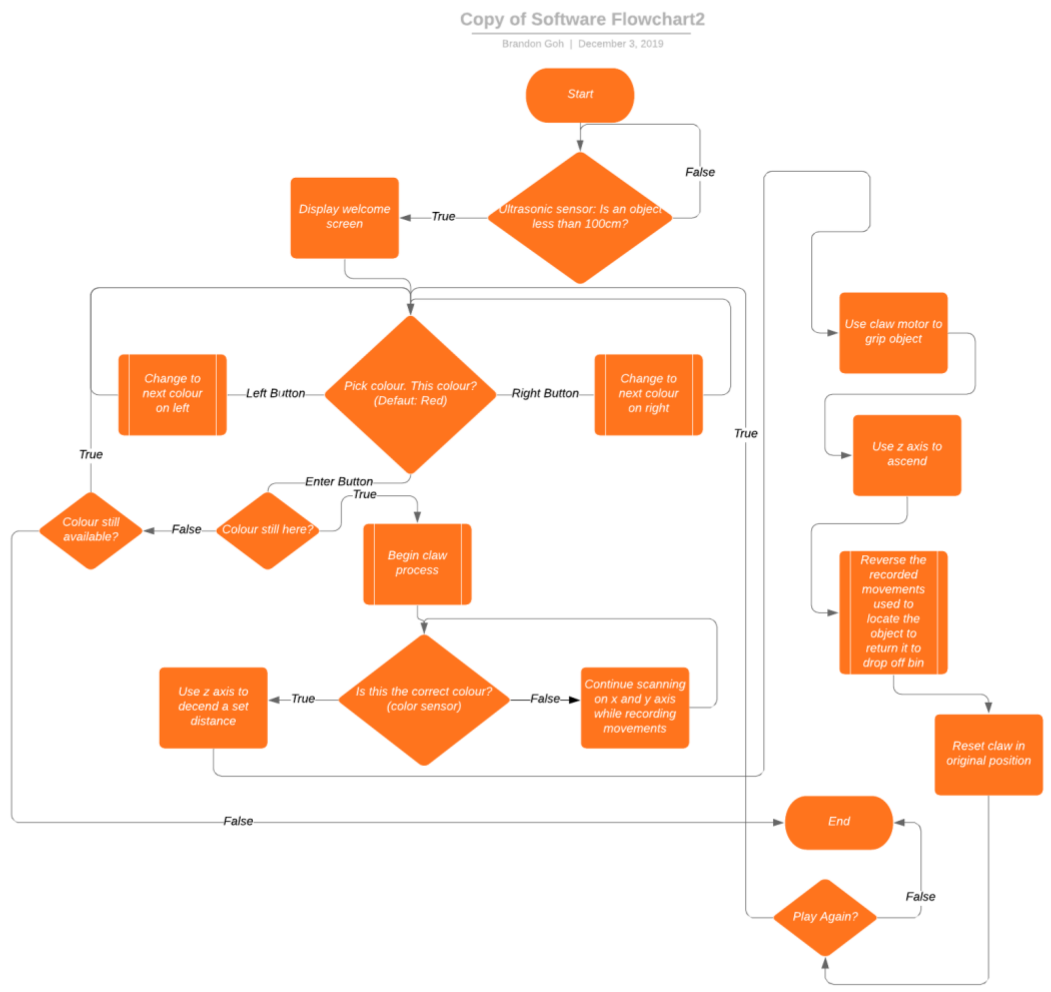
*Extra functionality that was unable to get perfected for the demo

## Testing

The tests started with the functions corresponding to the display screen functionality. We first tested if colourRight and Left worked as expected (reset and display a new option to the screen when correct buttons are selected) and that unexpected inputs were ignored. Startup was then tested to ensure that it handles invalid inputs (already selected colours) properly.

We then tested the objectPickup and Transport. The way these were tested were first seeing if the motors ran for the proper distance (ensuring it spanned the entire system). Then ensuring that the claw stops over the object regardless of what colour was selected and where it was placed. This was tested by placing objects of various colours in different spots and ensuring the claw stopped over the right colour regardless of where it was found. This test was to ensure that the system was dynamic, allowing for changes to be made without affecting the functionality of the machine.

The shutdown procedures were tested next. This was done by running through the program then selecting to play again and ensuring that the system allowed you to play again and did not shut off/terminate instead. The same test was then repeated but the system was shutdown instead to ensure that the user would not be stuck playing longer than they wanted to. Figure 8 below shows the planning process through a flowchart made for the software portion of the system.



Copy of Software Flowchart2
Brandon Goh | December 3, 2019
Start
Ultrasonic sensor: Is an object less than 100cm?
False
True
Display welcome screen
Pick colour. This colour? (Defaut: Red)
Left Button
Right Button
Change to next colour on left
Change to next colour on right
Enter Button
True
Colour still here?
False
Colour still available?
True
Begin claw process
Is this the correct colour? (color sensor)
True
False
Use z axis to decend a set distance
Continue scanning on x and y axis while recording movements
Use claw motor to grip object
Use z axis to ascend
Reverse the recorded movements used to locate the object to return it to drop off bin
Reset claw in original position
End
Play Again?
False
True
False

*Figure 8: Software Flowchart*

# Verification

## Constraints Met

**System must allow for 3 axes of motion**

This constraint was met, since our system incorporated movement in the x, y, and z direction. The claw was able to scan across the dimensions of our box by moving up, down and across rows. A motor-controlled movement in the x direction, which was moving across the rows. An additional motor-controlled movement in the y direction which was moving from row to row. Finally, the color sensor continually scanned and when a correct measurement was read, the y motor would activate. This motor would actuate to allow for the claw to raise and lower. These three axes of motion were proven to be successful in the demonstration of our robot as seen below in Figure 9.

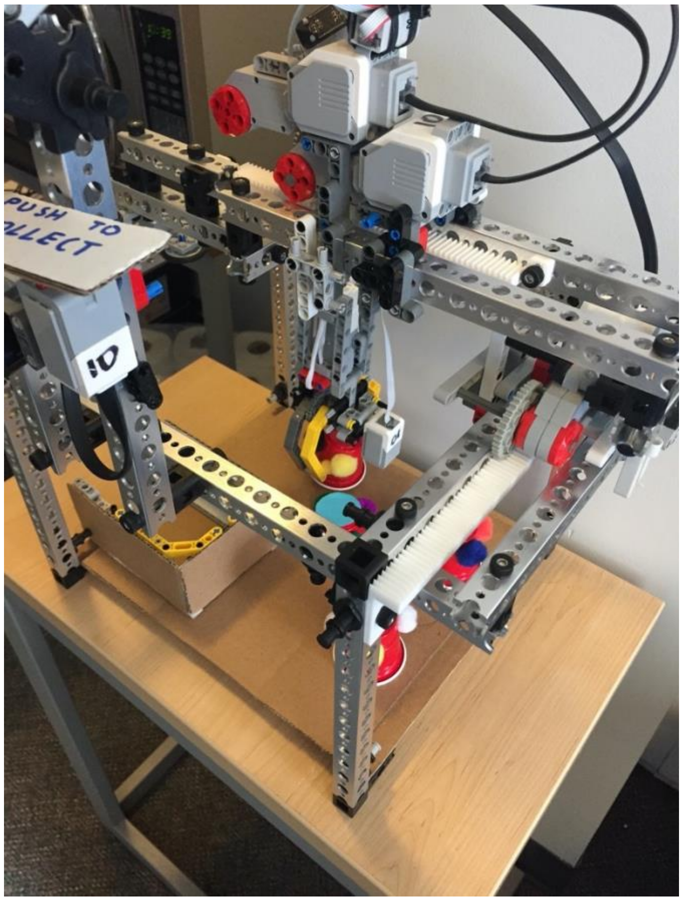


*Figure 9: Axes of Motion of System*

**Must not exceed maximum wire length**

Our system used almost every port of the LEGO EV3 brick, therefore requiring a large amount of wires. By incorporating wire extenders, we were able to create long enough wires for our claw to span the entire area. The wires were organized with zip ties allowing for clean cable management in our system. From using the extenders, our system had more than enough wire slack, and therefore we met the constraint of not exceeding the maximum wire length as seen below in Figure 10.

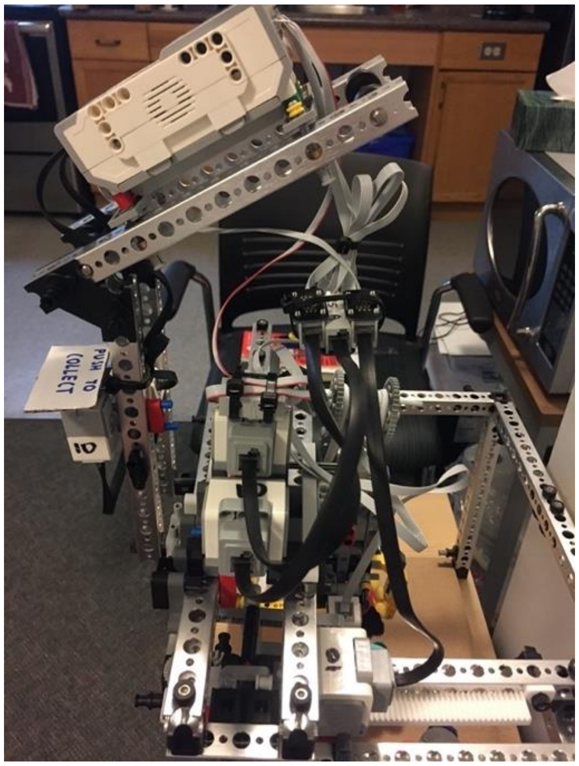


*Figure 10: Wires of the System*

**Colour sensor within 3 cm range**

This criterion was met in our system, since we mounted the color sensor onto the claw, held at a constant height of roughly 1 cm above the objects. All the objects used in our system, have the exact same height. This distance of approximately 1 cm allowed for our colour sensor to activate and take readings as seen below in Figure 11.

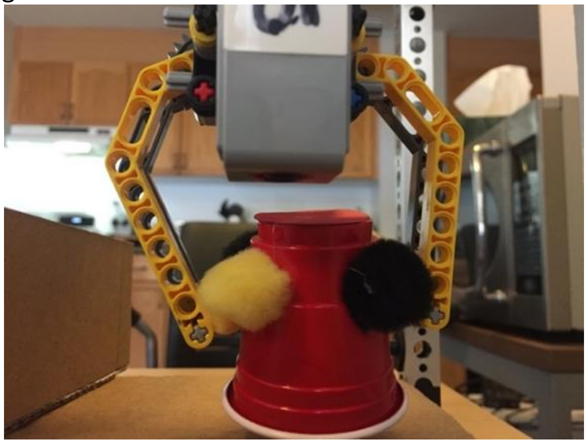


*Figure 11: Colour Sensor of System*

**Ultrasonic within 5 cm and 200 cm range**

The start up procedure for our program revolves around the ultrasonic sensor activating within 100 cm. By using this 100 cm range, we do not have to concern ourselves with the 200 cm boundary of the ultrasonic sensor. As for the 5 cm range, the ultrasonic sensor is mounted within the system, making it difficult for the user to come too close to the machine. Therefore, the 5 cm distance is not a large issue for our system and can be seen below in Figure 12.

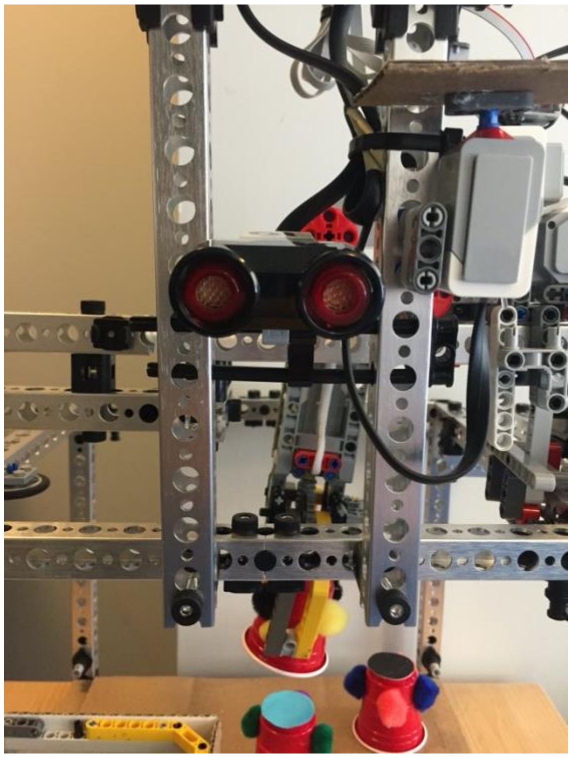


*Figure 12: Ultrasonic Sensor of System*

## Constraints Not Met

**Frame successfully developed from the limited building supplies given to us**

This constraint was not met because our mechatronics system incorporated 3D printed parts in order to properly function. Although, the majority of our system was constructed from the materials supplied, there were certain pieces that we needed a certain way. The only way to accomplish this was to design and print these parts using a 3D printer. These pieces were the trackpads for the x and z motor to allow for the gear to rotate and travel across when the motor activated. Since these pieces were vital for the functionality of our system, we chose to design the parts ourselves in SolidWorks, to achieve the exact design we wanted. To address this issue, we could alter this constraint, stating our design must be built using the parts supplied and no more than 5 $in^3$ of volume of 3D print can be used. That way, we can still design the crucial pieces in SolidWorks, although our new constraint is adjusted to limit the amount of 3D printed parts used as seen below in Figure 13.

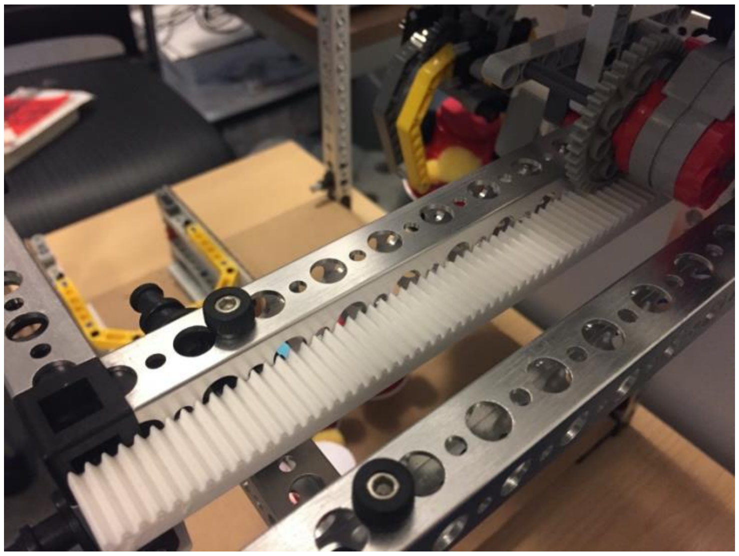


*Figure 13: 3D Printed Part of System*

## Project Plan

The information in this section has been referenced from [1].

### Division of Tasks

**Mechanical Component:**

One of the major components is the mechanical design of the system which consists of all physical pieces used in systems such as the claw, frame, motors, sensor mounts and much more. Table 2 below shows the project plan of the mechanical components of our design.

*Table 2: Project Plan of Mechanical Components*

| Sub-tasks | Task Entailments | Task Dependencies |
|---|---|---|
| Frame Build (Masih) | • Making sure the frame is sturdy enough to support the weight of the arm, claw and item<br>• Will be most time-consuming and must consider other factors such as motor size and wire length | Must be completed first in order to mount other devices |

| | | |
|---|---|---|
| Motor Mounts (Jordan) | • Making sure the wires can be attached to the EV3 and will not interfere with the movement of the arm | Must be completed second in order for the sensors to have working functionality with motors |
| Sensor Mounts (Sam) | • Making sure the sensors are mounted properly and are able to function to the best of its ability<br>• Colour sensor: Making sure it does not interfere with the claw<br>• Ultrasonic sensor: Making sure nothing blocks the view and has reading above 100cm without any users<br>• Touch sensor: Making sure it can be placed inside the claw without disrupting the grip of the item or claw | Must be completed third after most of structure is complete |
| System Design (Brandon) | • Making sure every component fits together properly and all pieces work as one | Must be completed last once everything has been mounted |

**Software Component:**

Another major component is the software design of the system which consisted of all software and code used in the claw machine. Table 3 below shows the project plan of the software components of our design.

*Table 3: Project Plan of Software Components*

| Sub-tasks | Task Entailments | Task Dependencies |
|---|---|---|
| Welcome screen (Brandon) | • Ultrasonic functionality<br>• Button functionality<br>• Choose colour<br>• Colour not chosen already | Must be completed first because the device needs to know what colour is chosen before proceeding |
| Correct colour (Jordan) | • Colour sensor functionality<br>• Knows desired colour | Should be completed 2nd or 3rd so the robot knows what colour it is looking for constantly |
| Motor movement (Sam) | Scan<br>• Movement of claw that will scan all items in the box<br>• Z motor will be used to lower the claw so that the colour sensor is less than 3 cm above the items | Should be completed 2nd or 3rd so the robot can scan and transport the item |

| | | |
|---|---|---|
| | • X and Y motor will be used to scan through items and search for the correct colour<br>Pickup item<br>• Touch sensor functionality<br>• Z motor will be used to pick up item and bring it back up above the other items<br>Move item<br>• X and Y motor will be used to move the item to the bin dropoff<br>Place item<br>• Z motor will be used to place item and bring the claw back up to original height | |
| Play again (Masih) | • Asks the user if they would like to play again<br>• Button functionality<br>• Reset claw machine to original position | Should be completed 4th after the program has successfully run |

## Revisions to Project Plan

There were a few revisions to the project plan due to the lack of time, material or expertise when building the claw machine. Initially, the group followed the project plan but there were many conflicts when merging the mechanical and software components of the system. The overall base functionality of selecting a colour, picking it up and moving it to the drop off bin was successfully done, but additional features such as play again and the touch sensor on the claw could not be done.

For the software side, the play again feature was not able to reset to its original position due to inaccurate measurement values regarding the motor encoder. We also tried to add a race feature by hardcoding x and y values where the objects would be at so it would be faster than scanning the whole span of the box. Then, the display would compare the time it took to retrieve an object of the users choice and of the robots choice and see which one was faster. We were limited on our time due to various other assignments, quizzes and reports throughout the weeks therefore this feature was abandoned.

For the mechanical side, the claw could only handle 1 sensor due to the weight on the axle of the claw that started to bend. Since the colour sensor was the main “autonomous” part of our system, we decided to forgo the touch sensor. Then, we thought about implementing the touch sensor at the bottom of the drop-off bin in order to confirm the object was not lost in transportation, but too heavy objects would bend the z axle too much. Therefore, the touch sensor was used at the end of the system process to “collect their item” as though the user reached into the claw machine and grabbed the retrieved object.

## Project Plan vs Actual Schedule

The project plan did not go as planned but it was anticipated that we would have errors and setbacks along the way. The initial project plan to complete our system in 1 week from the Preliminary Design Report due date turned out to be completed on the night before demo day. There were various setbacks that we took into consideration along the way such as using a 3D printer to print the customized trackpad for the x and y axis. Furthermore, the watIAM lab at Waterloo was fully booked therefore we used a friend's 3D printer instead. Overall, there were various challenges along the way from lack of materials to idea variations that members didn't agree upon that caused the projected plan to go longer than expected.

## Conclusion

The problem we are trying to solve is designing and constructing an autonomous claw machine that retrieves specific objects based on colour and user input, through the implementation of mechatronics principles.

Our design was not capable of meeting the original constraints, due to the inaccuracy of the colour sensor. Therefore, new criteria were created of successful location and retrieval of the correct object at least once out of five times. Our final system was able to successfully meet these altered criteria. As for constraints, all constraints were met, except designing the frame with only the parts given. In order to have the precision of parts we required, we decided to 3D print certain parts, failing to meet this constraint.

Important features regarding the mechanical design are the motors, gears, and trackpads that allow three degrees of motion. This requirement was crucial for us to achieve, in order to have a complete claw machine. The motors, gears, and trackpads all worked in conjunction with each other in order to move the claw across the span of the frame. This mechanical movement was executed through a detailed software design.

Finally, key features for the software aspect included a screen display that allows the user to see exactly what colour they would choose rather than simply being told to push a button corresponding to an object/colour as well as keeping track of which colours have been taken. A scanning of the system that locates the object based on colour while also keeping track of the distances it travels in order to return the object to the drop off bin and reset its positioning. A successful transport of the object the required distances from a previous function to the drop off bin. Lastly, a successful shut down procedure based on the progress of the game (how many objects have been retrieved) and user input (whether they choose to play again or not).

## Recommendations

### Mechanical Design

With our current design, our claw only spanned a space of *$20cm_X$ x $20cm_Y$ x $15cm_Z$*. If we were able to create a system that could span a greater area, then there would be more room to place objects and there would be less worry of interfering with objects during our pickups. This could've been done by having longer Tetrix beams to make our frame out of. As well, if we had been supplied with sturdier rods, the rod holding up our claw machine would be able to support more weight on its end, and we would have been able to incorporate a touch sensor onto our claw mechanism. This would've allowed for both these design improvements would result in more versatile mechanisms, allowing us to produce a better overall mechanical system.

## Software Recommendations

Some ways the software design could have been improved is by adding more functions to the source code. As it stands right now, there are a lot of functions written, although the main is still crowded. In addition to the main being very crowded, there are a lot of repeated tasks within main. For example, the overall function of the system is repeated twice (once for each game mode) so this could have been replaced with one single function.

By making this change, debugging the program would have been much easier. In the programs current state, when something was not working, we would have to check every function that relates to the issue, along with main itself. If more functions were added and main was tidied up, we would simply need to look through the functions and debugging could be done in a more systematic and simpler process.

## References

[1] J. C. S. E. B. G. M. Bouriayee, "Preliminary Design Report," University of Waterloo, Waterloo, 2019.

[2] Wikipedia, "Claw Crane," 27 October 2019. [Online]. Available: https://en.wikipedia.org/wiki/Claw_crane. [Accessed 3 December 2019].

[3] P. Edwards, "Claw machines are rigged — here's why it's so hard to grab that stuffed animal, " Vox Media, 3 June 2015. [Online]. Available: https://www.vox.com/2015/4/3/8339999/claw-machines-rigged. [Accessed 3 December 2019].

[4] "Claw Crane," Wikipedia, [Online]. Available: https://upload.wikimedia.org/wikipedia/en/thumb/0/0a/Snorkelling_with_the_swollen_purple_head.JPG/1920px-Snorkelling_with_the_swollen_purple_head.JPG. [Accessed 3 December 2019].

# Appendices

## Appendix A – Code of System

```
// Sam Eskandar, Masih Bouriayee, Jordan Chow, Brandon Goh
const int COLOURSIZE = 3;
const float XMAX = 15.5 * 360 / (4 * PI);
const float YMAX = 4.5 * 360 / (4 * PI);
// radius of wheel = 2cm

int colourLeft(int* colours,int & coloursIndex)
{
    coloursIndex--;
    if (coloursIndex < 0)
    {
        coloursIndex += 4;
    }
    int colourChosenNum = 0;

    colourChosenNum = colours[coloursIndex];
    displayString(2,"Black | Red | Blue");

    return colourChosenNum;
}

int colourRight(int *colours,int & coloursIndex)
{
    coloursIndex++;
    if (coloursIndex > 3)
    {
        coloursIndex -= 4;
    }
    int colourChosenNum = colours[coloursIndex];
    displayString(2,"Black | Red | Blue");

    return colourChosenNum;
}


int startup(int *colours,bool *coloursPicked)
{
    int allColours = 0;
    int coloursIndex = 0;
    int colourChosenNum = colours[coloursIndex];

    while(!getButtonPress(buttonEnter))
    {
        displayString(2,"Black | Red | Blue");
        if(getButtonPress(buttonLeft))
        {
            while(getButtonPress(buttonLeft))
            {}
            eraseDisplay();
            colourChosenNum = colourLeft(colours, coloursIndex);
            displayString(3,"Selected Colour : ");
            if (colourChosenNum == 1)
                displayString(4,"Black");
            else if (colourChosenNum == 2)
                displayString(4,"Blue");
            else
                displayString(4,"Red");
        }

        else if(getButtonPress(buttonRight))
        {
            while(getButtonPress(buttonRight))
            {}
            eraseDisplay();
            colourChosenNum = colourRight(colours,coloursIndex);
            displayString(3,"Selected Colour : ");
            if (colourChosenNum == 1)
            {
                displayString(4,"Black");
            }
            else if (colourChosenNum == 2)
            {
                displayString(4,"Blue");
            }
            else
            {
                displayString(4,"Red");
            }
        }
        else if(getButtonPress(buttonUp))
        {
            displayString(9,"Please use the left, right");
            displayString(10,"and center button Only");
        }
        else if(getButtonPress(buttonDown))
        {
            displayString(9,"Please use the left, right");
            displayString(10,"and center button Only");
        }
    }
    //colour already chosen error 400
    if (coloursPicked[coloursIndex] == 1)
    {
        eraseDisplay();
        displayString(9,"Colour already chosen");
        return 400;
    }

    //all objects are gone error 404
    coloursPicked[coloursIndex] = true;
    for (int index = 0; index < 3; index++)
    {
        if (coloursPicked[index] == 1)
            allColours += 1;
    }
    if (allColours == 3)
        return 404; //terminates program

    else
        return colourChosenNum; //continues program
}

void objectPickup(int &scanx, int &scany, int powery, int powerx, int color, bool &isThere,
int &row)
{
    isThere = false;

    //for (int counter = 0; counter < 2; counter++)
    //{
    if (SensorValue[S1] != color) // color is the variable
    {
        nMotorEncoder[motorB] = 0;
        while (abs(nMotorEncoder[motorB]) < YMAX)
        {
            motor[motorB] = powery;
        }
        motor[motorB] = 0;
        nMotorEncoder[motorA] = 0;
        while (abs(nMotorEncoder[motorA]) < XMAX && SensorValue[S1] != color)
        {
            motor[motorA] = powerx;
            if(SensorValue[S1] == color)
            {
                wait1Msec(500);
                isThere = true;
            }
        }

        motor[motorA] = 0;
        //powerx *= -1;
        scanx = (4 * PI * nMotorEncoder[motorA]) / 360;
        scany += 4.5;
    }
    //row++;
    //}
    if (isThere == true)
    {
        motor[motorA] = (-1)*powerx;
        wait1Msec(1000);
        motor[motorA] = 0;
        motor[motorC] = -15;
        wait1Msec(1000);
        motor[motorC] = 0;

        time1[T1] = 0;
        while(time1[T1] < 5000) // to be changed after testing
        {
            motor[motorD] = -20;
        }
        motor[motorD] = 0;

        motor[motorC] = 15;
        wait1Msec(4000);
        motor[motorC] = 0;
    }

    else
    {
        eraseDisplay();
        displayString(8, "No Object detected, is it");
        displayString(9, "out of bounds?");
    }
}

void objectTransport(int scanx, int scany, float &yTogo, float &xTogo, bool isThere, int row,
int powerx)
{
    nMotorEncoder[motorB] = nMotorEncoder[motorA] = 0;
    xTogo = (17 - scanx) * 360 / (4 * PI);
    //if (row == 2)
    //{
    //    xTogo = (17 - scanx) * 360 / (4 * PI);
    //    powerx *= -1;
    //    //xTogo = scanx* 360 / (4 * PI);
    //    ////powerx *= -1;
    //}

    //else
    //{
    //    xTogo = (17 - scanx) * 360 / (4 * PI);
    //    powerx *= -1;
    //}

    yTogo = (scany + 1) * 360 / (4 * PI);
    if (isThere == true)
    {
        while(abs(nMotorEncoder[motorB]) < yTogo)
        {
            motor[motorB] = 10;
        }

        motor[motorB] = 0;

        while(abs(nMotorEncoder[motorA]) < xTogo)
        {
            motor[motorA] = powerx;
        }
        motor[motorA] = 0;

        motor[motorD] = 20;
        wait1Msec(6000);
        motor[motorD] = 0;
    }
}

void reset(float yTogo, float xTogo)
{
    nMotorEncoder[motorA] = nMotorEncoder[motorB] = 0;
    while(abs(nMotorEncoder[motorB]) < yTogo)
    {
        motor[motorB] = 20;
    }

    motor[motorB] = 0;

    nMotorEncoder[motorA] = 0;
    while(abs(nMotorEncoder[motorA]) < xTogo)
    {
        motor[motorA] = -20;
    }
    motor[motorA] = 0;

    motor[motorD] = -15;
    wait1Msec(6000);
    motor[motorD] = 0;
}

bool playAgain()
{
    bool status = false;

    while (SensorValue[S2] != true)
    {
        displayString(8, "Would you like to play again?");
        displayString(9, "Press touch for yes and");
        displayString(10, "any other button for no");
        if (getButtonPress(buttonAny))
        {
            return status;
        }
        status = false;
    }
    status = true;
    while(!getButtonPress(buttonAny))
    {
    }
    return status;
}

bool gameMode()// returns true if the user wants to race the robot
{
    displayString(8, "Would you like to race?");
    displayString(9, "Press enter for yes and");
    displayString(10, "any other button for no");

    if (getButtonPress(buttonEnter))
    {
        return true;
    }
    else
        return false;
}

bool isSuccesful()
{
    eraseDisplay();
    bool status = false;

    while (SensorValue[S2] != true)
    {
        displayString(8, "Would you like to play again?");
        displayString(9, "Press touch for yes and");
        displayString(10, "any other button for no");
        if (getButtonPress(buttonAny))
        {
            return status;
        }
        status = false;
    }
    status = true;
    while(!getButtonPress(buttonAny))
    {
    }
    return status;
}


int controls() // unfinished extra functionality (not part of requirements)
{
    eraseDisplay();
    displayString(6, "The controls are as follows");
    displayString(7, "Up and Down buttons control y");
    displayString(8, "axis and right and left for x");
    displayString(9, "Once centred above the object,");
    displayString(10, "press the enter button once, then");
    displayString(11, "the up and down buttons will raise");
    displayString(12, "and lower the claw, the right button");
    displayString(13, "will close the claw, and the left open");
    displayString(14, "GLHF");
    wait1Msec(10000);
    time1[T1] = 0;

    while (!getButtonPress(buttonEnter))
    {
        if (getButtonPress(buttonRight))
        {
            while (getButtonPress(buttonRight))
            {
                motor[motorA] = 20;
            }
            motor[motorA] = 0;
        }

        if (getButtonPress(buttonLeft))
        {
            while (getButtonPress(buttonLeft))
            {
                motor[motorA] = -20;
            }
            motor[motorA] = 0;
        }

        if (getButtonPress(buttonUp))
        {
            while (getButtonPress(buttonUp))
            {
                motor[motorB] = 20;
            }
            motor[motorB] = 0;
        }

        if (getButtonPress(buttonDown))
        {
            while (getButtonPress(buttonDown))
            {
                motor[motorB] = -20;
            }
            motor[motorB] = 0;
        }
    }

    while (!getButtonPress(buttonEnter))
    {
        if (getButtonPress(buttonRight))
        {
            while (getButtonPress(buttonRight))
            {
                motor[motorC] = 20;
            }
            motor[motorC] = 0;
        }

        if (getButtonPress(buttonLeft))
        {
            while (getButtonPress(buttonLeft))
            {
                motor[motorC] = -20;
            }
            motor[motorC] = 0;
        }

        if (getButtonPress(buttonUp))
        {
            while (getButtonPress(buttonUp))
            {
                motor[motorD] = 20;
            }
            motor[motorD] = 0;
        }

        if (getButtonPress(buttonDown))
        {
            while (getButtonPress(buttonDown))
            {
                motor[motorD] = -20;
            }
            motor[motorD] = 0;
        }
    }
    return time1[T1];
}

task main()
```

```
{
    SensorType[S1] = sensorEV3_Color;
    wait1Msec(50);
    SensorMode[S1] = modeEV3Color_Color;
    wait1Msec(50);
    SensorType[S2] = sensorEV3_Touch;
    SensorType[S3] = sensorEV3_Ultrasonic;
    motor[motorA] = motor[motorB] = motor[motorC] = motor[motorD] = 0; //assuming A and    B
are x-y respectively, C controls z and D controls claw
    nMotorEncoder[motorA] = nMotorEncoder[motorB] = 0;

    int scanx = 0, scany = 0, powery = -10, powerx = 10, row = 1;
    int colourChosenNum = 0;
    float yTogo = 0, xTogo = 0;
    int colours[COLOURSIZE] = {1,5,2};
    bool plays = true, race = false, isThere = false;
    bool coloursPicked[COLOURSIZE] = {false,false,false};

    while(SensorValue[S3] > 100)
    {}

    while (colourChosenNum != 404 && plays == true)
    {
        while(!getButtonPress(buttonAny))
        {
            race = gameMode();// race false signifies that the "race" game mode has not been
selected
        }

        if (race == false)
        {
            displayString(1,"Please select a colour: ");
            displayString(2,"Black | Red | Blue");
            displayString(3,"Selected Colour : Black");

            colourChosenNum = startup(colours, coloursPicked);
            while(colourChosenNum == 400)
            {
                wait1Msec(1000);
                int newNum = startup(colours,coloursPicked);
                colourChosenNum = newNum;
            }
            objectPickup(scanx, scany, powery, powerx, colourChosenNum, isThere, row);
            objectTransport(scanx, scany, yTogo, xTogo, isThere, row, powerx);
            eraseDisplay();
            if (isThere == true)
            {
                if(isSuccesful() == false)
                {
                    reset(yTogo, xTogo);
                }
            }
            plays = playAgain();
            if (plays == true)
            {
                reset(yTogo, xTogo);
            }
        }

        else
        {
            displayString(1,"Please select a colour: ");
            displayString(2,"Black | Red | Blue");
            displayString(3,"Selected Colour : Black");
            colourChosenNum = startup(colours, coloursPicked);
            float userTime = controls()/1000.0;
            eraseDisplay();
            displayString(8, "Your time is %f", userTime);
            displayString(9, "Please return the object");
            wait1Msec(20000);
            eraseDisplay();
            colourChosenNum = startup(colours, coloursPicked);
            objectPickup(scanx, scany, powery, powerx, colourChosenNum, isThere, row);
            objectTransport(scanx, scany, yTogo, xTogo, isThere, row, powerx);

        }
    }
}
```