Date: March 26th, 2022
Revision Date: April 2nd, 2022
Version 2.0

Final Design Report

Team 01 [ECSE211 W2022]

Student 1:Lawi Mwirigi
ID:260831614
Student 2: Muqtadir Ahmed
ID: 260987724
Student 3:Ryan Tabbara
ID:260835629
Student 4: David Zhang
ID: 260381863
Student 5: Nour Ktaily
ID: 260944864
Student 6: Miiyu Fujita
ID: 260394905

Table of Contents

1. Team Management

1.1 Team Members

David Zhang: U2 Electrical Engineering major. In the mini-project, David was responsible for building most of the hardware structure of the instrument. He also provided diagrams and sketches of system ideas in the first stages of the design process as he has an iPad. He is also a member of MRT's avionics subteam, and has knowledge on antenna arrays. He has taken relevant software courses (ECSE202, COMP250).

Muqtadir Ahmed: U2 Electrical Engineering major. In the mini-project, Muqtadir was responsible for testing the sensors (Lab 2), testing the motor and gyro sensor (specific to chosen mini-project design), and took part in the integration of all components of the system. He has taken relevant software courses (ECSE202, COMP250).

Nour Ktaily: U2 Computer Engineering major. In the mini-project, Nour was responsible for documenting the progress of the instrument's design, as well as managing the time and coordination of the group. She also took part in the software development of the instrument. She previously participated in a robotics project in high school, and has taken relevant software courses (ECSE202, COMP250).

Ryan Tabbara: U4 Electrical Engineering major. In the mini-project, Ryan was responsible for documenting the progress of the instrument's design, as well as the Gantt Chart. He has taken relevant software courses (ECSE202, COMP250).

Lawi Mwirigi: U4 Electrical Engineering major. In the mini-project, Lawi was responsible for the documentation of the progress of the instrument, as well as the Gantt Chart. He also helped with the software development and hardware building of the instrument. He has taken relevant the following relevant courses: ECSE202, COMP250, COMP206, COMP251, ECSE451

Miiyu Fujita: U2 Electrical Engineering major. In the mini-project, she was responsible for writing the color detection software, and testing the color, ultrasonic and touch sensors. She also took part in the integration of all components of the system. She has taken relevant software courses (ECSE202), and has previous experience working on software projects in MRT, and has participated in a machine learning bootcamp offered at McGill (MAIS202). She also has had project management experience during her time in MRT.

1.2 Team Organization

1.2.1 Member Roles

Members took on the following roles based on their known expertise in certain domains and previous experiences. For specifics concerning members' relevant experiences and capabilities, please refer to Appendix A.1

Hardware Lead: David Zhang
Software Lead: Muqtadir Ahmed (Muqto in certain appendix documents)
Documentation / Software Engineer: Nour Ktaily

Project Manager: Ryan Tabbara
Testing Lead: Lawi Mwirigi
Documentation Manager / Deputy Team Manager: Miiyu Fujita

1.2.2 Member Schedules and Availabilities

All members except for Lawi had midterms scheduled alongside the design weeks, as outlined in Figure 2. Lawi instead was primarily occupied by projects and assignments. Ryan, alongside the midterms listed below, was occupied by his Capstone project. Based on weekly member availability, status update meetings between members were to take place on Saturdays. Meetings outside of status updates and Senior Engineering meetings were to take place based on necessity (i.e design iteration meetings). Please refer to Appendix A.2 for the full availability table.

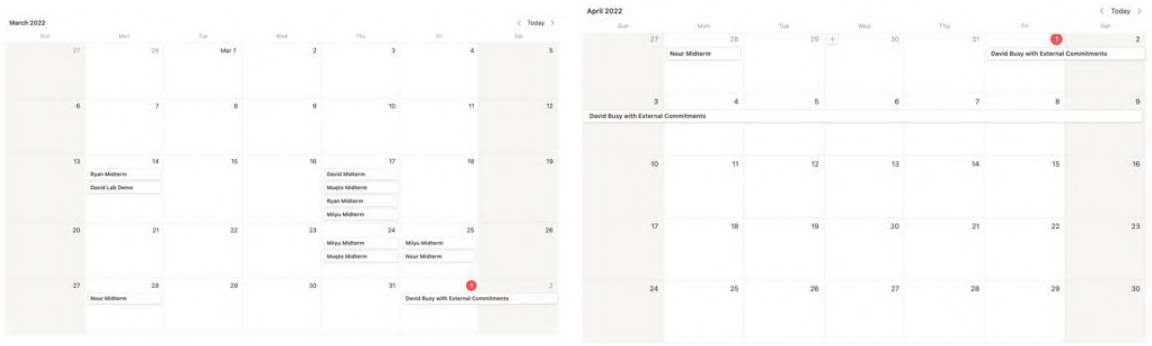


Figure 2. a) March Team Member Midterm Schedule (b) April Team Member Midterm Schedule

1.2.3 Team Contract

The following team contract was formulated as a way to recognize team effort.

1. Members of Team 1 agree to be active enough on communication platforms such that they can answer within a 24 hour window
2. Members of Team 1 recognize that although they are primarily responsible for their own tasks, they must also lend a helping hand to those who are completing tasks that require more man-power
3. Members of Team 1 agree to always keep each other on the same page
4. Members of Team 1 agree to ask for help as soon as they recognize they need it, such that a solution can be rapidly brainstormed

I, as a member of Team 1, agree to follow the aforementioned clauses.

Sign here: Ryan Tabbara

Sign here: David Zhang

Sign here: Lawi Mwirigi

Sign here: Muqtadir Ahmed

Sign here: Nour Ktaily

Sign here: Miiyu Fujita

1.2.4 Shared Goals

These are the shared goals that were agreed upon to motivate and encourage the team.

- Design and implement a functional system that members can be proud of.
- Open communication: keep each other updated throughout the design process, so that members can help each other through roadblocks and keep each other motivated.
- Learning: after undergoing 5 design weeks, to come out with more knowledge about the BrickPi and the engineering design process.
- Open-mindedness: be open about discussions and ideas.

1.3 Project Management

The project management plan revolved around project milestones set at the beginning, and how the team was to go about meeting these deadlines by tracking progress and distributing tasks accordingly.

1.3.1 Project Milestones

All due dates including weekly deliverables, beta and final demos, final deliverables and design reviews were set as project milestones.

1.3.2 Progress Tracking

Team progress on tasks was tracked in relation to the Gantt chart initially set up during week 1 of the design process. To do so, weekly status update meetings were scheduled on Saturdays, as well as weekly meetings with the Senior Engineer every Wednesday. In between these week days, communication was done through WhatsApp, where team members updated each other as they completed specific tasks during the week. The Gantt Chart can be found in appendix A.5.1. The Gantt Chart was further updated during Week 4 to accommodate the new tasks allocated during development. (See Appendix A.5.2 & A.5.3)

1.3.3 Modifications and Adjustments

Our initial budgeted time changed over the weeks, due to reasons such as hardware implementation taking much more time than expected, as well as due to unforeseen circumstances related to the ongoing pandemic. Please refer to A.3 for details on adjustments to budget made during the project. Specific tasks for each week were also allocated specific amounts of hours. Please refer to A.4 for details.

2.0 Design Problem Analysis

2.1 Problem Statement

Using the tools provided in the two DPM kits, the team must design and implement a system that, upon receival of a minimum of 6 unsorted cubes of 3 distinct colors, can sort these cubes within 2 minutes when indicated to do so by the store clerk, in preparation for store opening. During store opening hours, the system must then be capable of responding to the store clerk's request for a specific colored cube (executed in the request area), to then deliver the requested cube to the delivery area from the storage/sorting space, autonomously, and within 5 seconds.

2.2 System Requirements

REQ 1. The system must be able to physically accommodate the weight of 6 foam cubes.

REQ 2. The system must account for the area occupied by 6 foam cubes.
REQ 3. The system must be able to identify 3 uniquely colored foam cubes.
REQ 4. The system must provide the store clerk a simple way of activating the pre-"sorting" process.
REQ 5. The pre-"sorting" process of the system must be completed within a 2 minute margin.
REQ 6. The system's item retrieval process must be easily operable by a single store clerk.
REQ 7. The system must be able to read and understand the requested color (i.e detect the color of the cube placed onto the request area).
REQ 8. The system must be able to retrieve foam cubes from the storage area according to specified colors to then deliver them onto the delivery area.
REQ 9. The system must be able to complete the request to delivery process within a time margin of 5 seconds or less.

2.3 Constraints

1. The system must be designed and implemented in 270 hours, in parallel with midterms and other commitments of team members.
2. The system must be designed and implemented using only tools provided in the DPM kit, as well as 0$ cost materials (i.e cardboard, plastic cups, tape)
3. The system design must take into account the availability and skillset of the team members. See A.1, A.2

2.4 System Specifications

2.4.1 Client's Specifications

1. The system's request area must be clearly marked with square outlines such that the cube's surface is at ~5mm from the color sensor of the system which will read the request.
2. The distance between the center of the request area and the delivery area must be less than 20cm wide.
3. The system must be less than 1m in all dimensions.

2.4.2 System Specifications

1. The system shall consist of four main subsystems: an input tower (includes a motor gate structure), a rotating platform, a delivery system, and a color sensor arm
2. The input tower shall be made such that it may hold the delivered unsorted cubes as it awaits the sorting process
3. The rotating platform shall store the 6 sorted cubes such that each cube is at 60 degrees apart from the other
4. The sorting process shall be triggered by a press of a touch sensor
5. Once the sorting process has been triggered, the motor gate shall guide the cubes as they fall down the input tower
6. Once the sorting process has been triggered the piston of the input tower shall push the lowest cube in the input tower's cube stack onto the platform
7. The color sensor arm shall read the request at the request area such that it may relay the request to the rest of the system
8. The delivery process shall be triggered once a request has been detected
9. Subject to a request, the platform shall rotate such that it may place the requested cube closest to the delivery area

10. Subject to request, the delivery system's arm shall perform a rotation about an axis parallel to the ground such that it may push the requested cube onto the delivery area
11. Subject to a request the system cannot answer (i.e Color is out of stock), the system shall print an "Out of Stock" message to the console

2.5 Assumptions

The main assumptions the team made during the design process were mostly in regards to the hardware tools provided. First, that the stack of cubes would fall gracefully down the input tower without getting stuck, and that the lego pieces provided to the team in the kit would not get stuck on each other as they glided on each others' surfaces.

3.0 Final System Design

3.1 Final System Design Overview

The final system design consists of 2 main processes and four (4) subsystems: the input tower, the rotating platform, the delivery system and the color sensor arm. See B.2 for the general system flow.

3.1.1 Sorting Process

The sorting process is responsible for receiving and storing the 6 foam cubes onto the storage space (i.e rotating platform), while digitally storing each cubes' location and color in the storage space such that it may be delivered subject to requests. The sorting process therefore uses the following subsystems: the input tower and the rotating platform. During the sorting process, the six foam cubes are loaded vertically into the input tower, such that the lowest cube is positioned in front of a piston arm which will push it onto the rotating platform. As the lowest cube is pushed off of the stack, the rest of the cubes are guided down the input tower by the motor gate and gravity, such that the next cube is ready to be pushed off of the stack by the piston arm. The rotating platform rotates 60° after each cube's sorting, such that each cube is placed equidistant from the other. This process repeats until all 6 cubes in the stack are stored in the storage space (rotating platform).

3.1.2 Delivery Process

The delivery process is responsible for reading and responding to the store clerk's request by delivering the requested cube to the delivery area. Subsystems used for the delivery process are the rotating platform, the delivery system, and the color sensor arm. The delivery process is triggered when a request cube is placed in front of the color sensor arm, which will then read the color of the request cube. The color read will then dictate the amount of rotation the rotating platform must undergo such that the requested cube is placed closest to the delivery area. Finally, the delivery system's arm will rotate such that it shall push the requested cube into the delivery area.

3.1.3 Subsystem 1: Input Tower

The input tower subsystem is responsible for initially holding the delivered cubes and storing them into the storage space. There are three main components of the subsystem, namely the input tower itself, the piston arm mechanism, and the motor gate. Upon initial delivery of cubes into the input tower, and subsequent press of the activation touch sensor, the lowest cube of the stack's color is read by the input tower's storing color sensor, and is pushed off of the stack, onto the storage space (rotating platform), using a piston arm mechanism. The color of each cube as well as its angular position on the rotating platform are digitally stored. The motor gate works alongside the hardware

structure of the input tower to guide the stack of cubes down as they fall after the lowest cube has been stored. See B.2.1 for the input tower flow.

3.1.4 Subsystem 2: Rotating Platform

The rotating platform subsystem functions as a rotating storage space of the system. All 6 cubes are loaded onto the platform, at digitally stored equidistant locations, such that the motor controlling the rotating motion of the platform may bring requested cubes to the delivery area during the delivery process. See B.2.2 for the rotating platform flow.

3.1.5 Subsystem 3: Delivery System

The delivery subsystem works with the color sensor arm and the rotating platform to respond to store clerk requests. Upon receiving a request at the color sensor arm, the delivery system waits for the rotating platform to bring the requested cube in front of its delivery area. Once the cube reaches this position, the delivery system's arm rotates about an axis parallel to the ground, pushing the requested cube into the delivery area. See B.2.3 for the delivery system flow.

3.1.6 Subsystem 4: Color Sensor Arm

The color sensor arm reads the store clerk's request once the system is ready for delivery (sorting process completed). See B.2.4 for the color sensor arm flow.

3.2 Final System Hardware

In terms of hardware, all 4 subsystems of the final system are integrated through LegoEV3 structures. Please note that the activation touch sensor is connected solely to the BrickPi, which is why it has not been included in the digital twin modeling of the main system structure. See B.1 for side to side figures of the Digital Twin and the final physical system.

3.2.1 Final Input Tower Hardware

The input tower's physical structure consists of the following: an input tower structure (loading area of delivered unsorted cubes) made of LegoEV3 pieces and 2 strips of clear tape, a piston-like arm consisting of LegoEV3 pieces and 1 motor, a color sensor for digital storage of each cube's color, a motor gate consisting of LegoEV3 pieces and 1 motor, and the supporting structure integrating all these subcomponents with adjacent subsystems, which consists of various LegoEV3 pieces. See B.4.1 for figures of the digital twin of the input tower.

3.2.2 Final Rotating Platform Hardware

The rotating platform's physical structure consists of the following: a circular platform made of cardboard, 1 motor responsible for the rotating functionality of the platform, and its supporting structure integrating the subsystem with its adjacent subsystems (i.e delivery system and input tower), which consists of various LegoEV3 pieces. See B.4.2 for the digital twin of the rotating platform's LegoEV3 structure.

3.2.3 Final Delivery System Hardware

The delivery system's physical structure consists of the following: the delivery system's arm, which consists of 1 motor controlling its motion and LegoEV3 pieces which create the physical surface used to push off the cube onto the delivery area, the delivery area which consists of a cardboard smarties box connected to the supporting structure of the delivery system through LegoEV3 pieces, and finally

the supporting structure which also integrates the subsystem with adjacent subsystems through LegoEV3 pieces. See B.4.3 for digital twin figures of the delivery system.

3.2.4 Final Color Sensor Arm Hardware

The color sensor arm's physical structure consists of the following: a color sensor to read clerk requests, an attached request area made of cardboard, and a LegoEV3 structure which connects the subsystem to adjacent subsystems. See B.4.4 for figures of color sensor arm.

3.3 Final System Software

The final system's software structure consists of 6 classes:

- *rotating_platform.py*
- *color_sensor_arm.py*
- *input_tower.py*
- *delivery_system.py*
- *gate_motor.py*
- *main_robot_ferris.py*

Main_robot_ferris.py contains the main function. Each class defines functions that are used in main_robot_ferris.py's main function such that the system's subsystems may accomplish the storing and delivery processes. Each subsystem and certain sub-components (i.e motor gate) have designated classes which define relevant helper functions used in the main function. Brief descriptions of each class and their functions are outlined below. See B.3 for the final system software structure.

3.3.1 Final input_tower.py Class

The input_tower.py class defines three functions. First, the function **move_cube_to_platform(motor)**, which moves the piston motor by 360º in order to push the cube and come back to its initial position. Second, the function **store_cube_color_and_angle(color, angle, dictionary)**, which appends the absolute value of the rotating platform motor's positional angle to the list of angles corresponding to the specific color in the dictionary. Third, the function **move_platform(platform_motor)**, which is used to rotate the platform motor by -60º to create space for the next cube to be deposited. All three of these helper functions play a crucial role in the system's sorting process.

3.3.2 Final rotating_platform.py Class

The rotating_platform.py class defines one function, **move_motor(motor, angle)**, which brings the cube placed at the specific angle in front of the delivery area. This function plays a crucial role in the system's delivery process.

3.3.3 Final delivery_system.py Class

The delivery_system.py class defines two functions. First, the function **move_requested_cube(motor)** which moves the delivery system motor by 360º, pushing off the cube from the platform to the delivery area. The second function, **remove_delivered(color, angle, dictionary)**, removes the angle from the list of angles associated with the specific color in the dictionary. These functions play a crucial role in the system's delivery process.

3.3.4 Final color_sensor_arm.py Class

The color_sensor_arm.py class contains three functions. First, the function **most_occured(List)**, which creates a list with keys as color names and values as number of occurrences in the array and then returns the color with the most occurrences. The second function, **get_requested_color(color_sensor)**, creates a list of 20 colors using the color sensor and then returns the most occurred color from that list using the function most_occured(List). The third function, **color_to_angle(color, dictionary)** returns the angle associated with the specific color from the dictionary or -1 if the list of angles in the dictionary is empty. These functions play a crucial role in the system's storing and delivering process.

3.3.5 Final gate_motor.py Class

The gate_motor.py class defines 2 helper functions, **open_gate(motor)** and **close_gate(motor)**. The first opens the motor gate by rotating its controlling motor by 20 degrees in the counter-clockwise direction, and the second closes the motor gate (such that it may clamp onto the stack of cubes) by rotating its controlling motor by 20 degrees in the opposite, clockwise direction.

3.3.6 Final main_robot_ferris.py Class

The class, main_robot_ferris.py contains the main function to be run. It imports all the aforementioned classes, such that it may define how all subsystems work together to ensure the system can complete both sorting and delivery processes successfully. Other than the necessary imports and configuring of ports, the main_robot_ferris.py class defines a dictionary to hold each delivered cube's color and location during sorting (**color_angle_dict**), and 3 helper functions. The first, **reset_motor(motor)** is used to set the motor's angle to 0. The second, **get_rp_angle(motor)**, returns the angle at which a colored cube is stored, and the third, **wait_until_stopped(motor)**, is used to wait until the motor is no longer moving. The main function itself is where all of the aforementioned classes and their helper functions come together to construct the desired behavior of the system. Once the class has been run, the sorting process waits for the activation touch sensor to be pressed (indicating cubes have been loaded onto the system). Once all the angles and speed limits of all motors of the system have been set through use of the **reset_motor(motor)** and **set_limits(self, power=0, dps=0)**, the motor gate clamps onto the stack of cubes such that only the lowest cube of the stack is mobile (**close_gate(motor)**). This lower-most cube's color is then read by the color sensor of the input tower (**get_requested_color(color_sensor)**), then gets pushed by the piston arm of the input tower onto the rotating platform (**move_cube_to_platform(motor)**). Once the piston arm has completed its full motion (**wait_until_stopped(motor)**), the motor gate loosens such that the stack of cubes drops into place (**open_gate(motor)**), then clamps back onto the now shorter stack of cubes (**close_gate(motor)**), leaving again the lower-most cube mobile. At this point, both the color of the cube just loaded onto the rotating platform (storage space), as well as the angle of the rotating platform's motor (**get_rp_angle(motor)**), is stored in a dictionary for future use (**store_cube_color_and_angle(color, angle, dict)**). The rotating platform then moves -60° to accommodate for the next cube (**move_platform(motor)**). This process repeats for each of the cubes in the stack, until all cubes are sorted onto the storage space (i.e rotating platform), at which point the rotating platform will move to its default delivery position such that the system is ready for delivery. Delivery is activated once a request is detected. The requested color is read by the color sensor arm's color sensor (**get_requested_color(color_sensor)**), and the software then checks whether the requested color is in stock. If it is in stock, the request color is translated into an angular position of the rotating platform's motor (**color_to_angle(color, dict)**), such that it may rotate to bring the requested cube right in front of the delivery area. Once the rotating platform's motor reaches this

specified angle (**move_motor(motor, angle)**), the delivery system's arm pushes the cube onto the delivery area (**move_requested_cube(motor)**). This angle is then removed from the dictionary (**remove_delivered(color, angle, dict)**), as this location no longer holds a cube. The software then continues to wait for requests.

Please refer to B.2 for information on how the aforementioned classes work together in implementing the system's functionality.

4.0 Design Iterations

4.1 System Design v1.0

The initial system design consisted of the same 2 processes (delivery and sorting), and 4 subsystems: the input tower (cube tower in this initial version), the rotating platform, the delivery system, and the color sensor arm. However, the specifics of each subsystem were different when compared to the final system design. At this stage in the design process, prototyping and software development had not yet started, which is why the subsystem descriptions are verbalized ideas and sketches rather than concrete prototypes.

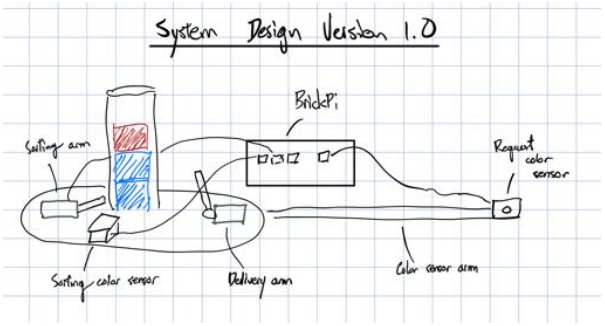


Figure 3. System Design Vesoin 1.0

4.1.1 Cube Tower (Input Tower in Final System) v1.0

The cube tower in its initial version was also to be the drop-off area for the unsorted cubes. It was initially designed to be placed at the center of the rotating platform/delivery system, as shown in Figure 4

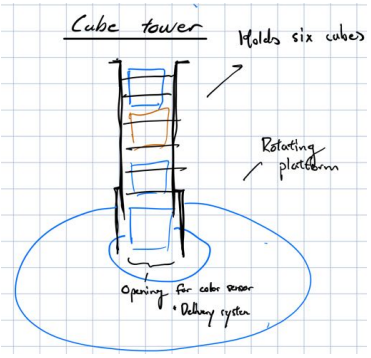


Figure 4. Cube Tower (Input Tower Version 1.0)

4.1.2 Rotating Platform v1.0

The rotating platform was initially designed to have three regions, one for each color, such that it may store the 6 cubes physically by color on its surface. The platform also had a hole in the center to

accommodate the cube tower which initially was designed to be placed in the center of the platform, as seen in Figure 5. The initial subsystem's rotation was made possible through use of a motor.

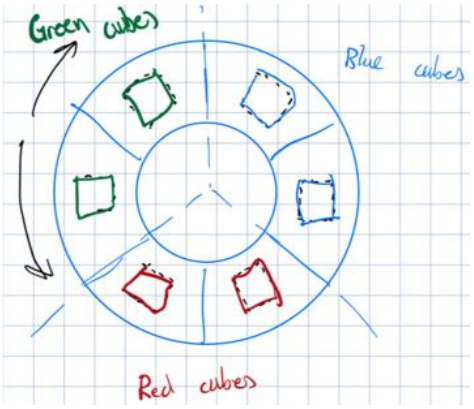


Figure 5. Rotating Platform Version 1.0

4.1.3 Delivery System v1.0

The initial delivery system consisted of two arms, the sorting arm and the delivery arm, where the sorting arm would be responsible for sorting the cubes from the cube stack onto their respective regions of the rotating platform, and the delivery arm would be responsible for delivering requests. The sorting arm was to undergo a rotation about an axis perpendicular to the ground to perform sorting, whereas the delivery arm resembled a "gate"-like mechanism which would lower itself at specific timings such that as the requested cube rotates close to the delivery area, it would be guided by the lowered delivery arm "gate" and slip onto the delivery area.

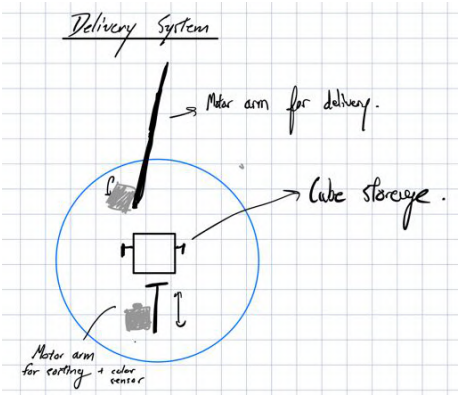


Figure 6 Delivery System Version 1.0

4.1.4 Color Sensor Arm v1.0

The initial design of the color sensor arm subsystem stays close to its final design. It consists of a long arm such that it may be connected to the rest of the system, and a color sensor that reads requests made.

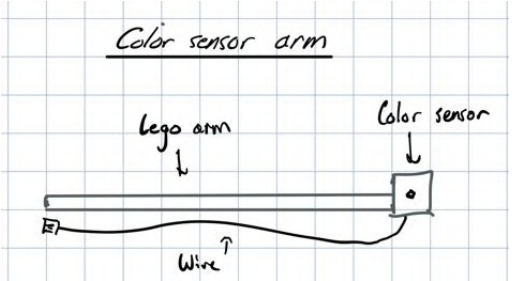


Figure 7. Color Sensor Arm Version 1.0

4.1.5 Software Design v1.0

As software development had not yet been undertaken at this stage of the design process, the system's software design was not yet split up into distinct classes, rather in terms of ideas of how the software would flow in order to implement each subsystem's functionality.

4.1.5.1 Cube Tower v1.0 Software Functionality

It was understood that the initial cube tower software must handle the initial delivery of the cubes. The only role the initial cube tower's software had was to read each cube's color such that it could be sorted into its respective region on the rotating platform.

4.1.5.2 Rotating Platform v1.0 Software Functionality

The initial rotating platform's software was designed such that the rotating platform could, based on the color sensor readings from the cube tower, rotate specific amounts such that the cube from the cube tower could be delivered onto its specific region.

4.1.5.3 Delivery System v1.0 Software Functionality

The delivery system's initial software was kept very simple, as the initial version of the delivery system was hardware focused. The software needed for the initial delivery system was to control the rotation of the motors that were ultimately controlling the physical sorting and delivery arms.

4.1.5.4 Color Sensor Arm v1.0 Software Functionality

The color sensor arm's initial software was to read the store clerk's requested color and to subsequently relay the request to the rest of the system.

4.2 System Design v2.0 / First Design Iteration

Prototyping of subsystems led to this first design iteration, as prototyping made clear what modifications could be made such that the system implementation would be more feasible. From the initial design, modifications were made to the input tower, the sorting process, the rotating platform, and the delivery system.

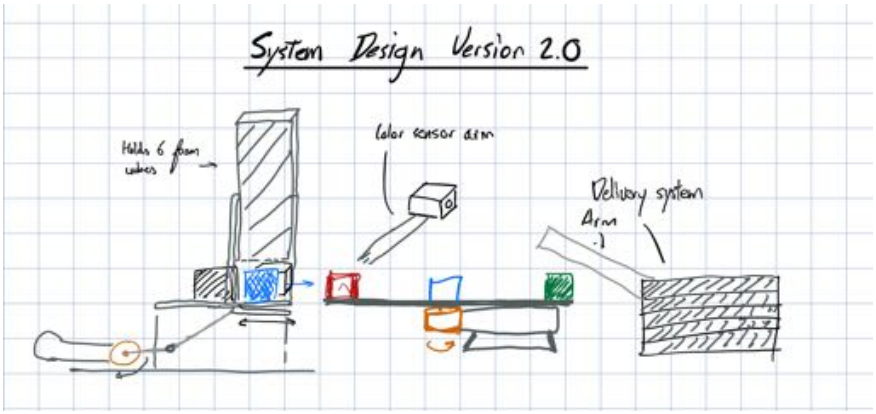


Figure 8. System Design Version 2.0

4.2.1 Input Tower v2.0

Placement of the Input Tower

The input tower was moved from the center of the rotating platform, to be placed next to the rotating platform, with its own supporting structure, which rendered the structural design of the input tower and rotating platform much simpler. This modification was made during the prototyping process of the rotating platform and the input tower, following modifications made to the rotating platform (See 4.2.2 for further details on modifications to the rotating platform). The surface of the platform no longer had a "donut hole" in its center, thus leaving no room for the input tower. See Figure 8 for previous and modified placement of the input tower, respectively.

New Piston Mechanism

This second version of the input tower now has additional functionalities, namely an additional piston mechanism. Initially, the mechanism to push the cubes of the stack onto the platform was part of the delivery system, but as the input tower was now placed by the side of the rotating platform, the input tower now had space to itself have an integrated piston mechanism that would push the cubes off of the stack onto the rotating platform, such that the sorting process could be centralized near one region of the system, and the delivery process near the other region of the system.

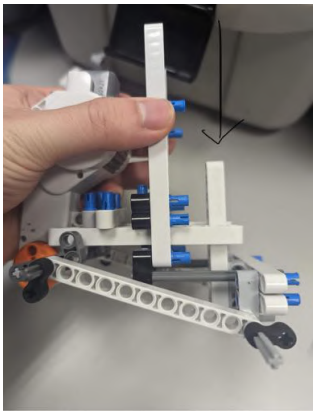


Figure 9. Piston Mechanism with indicating arrows

Input Tower Hardware Structure

Subject to the above modifications made to the subsystem, the overall hardware structure evolved from its initial version, as there were now added components such as the piston mechanism and the input tower's physical supporting system made of LegoEV3 pieces.

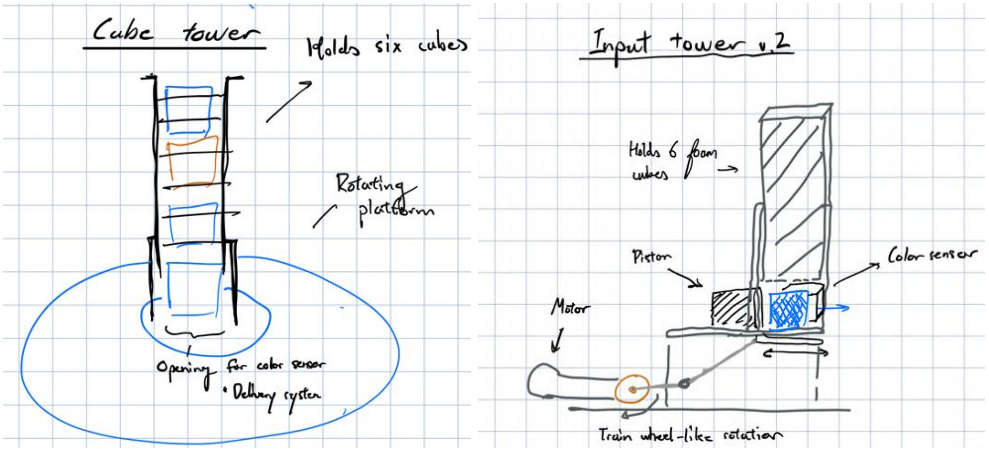


Figure 10. (a) Previous Hardware Structure (b) Modified Hardware Structure with Piston

4.2.2 Rotating Platform v2.0

During the prototyping process of the platform, the decision was made not to cut out a hole in the center of the platform, as placing the rotating motor in the center of the platform's surface would be the simplest means of rotating the platform about its central axis. Thus, for ease of connection between the platform and the motor at its center, the platform's surface in the center was kept. To note also is the omission of the physical sorting process, which implies that in this version, there is no longer a need to have color coded regions on the platform.

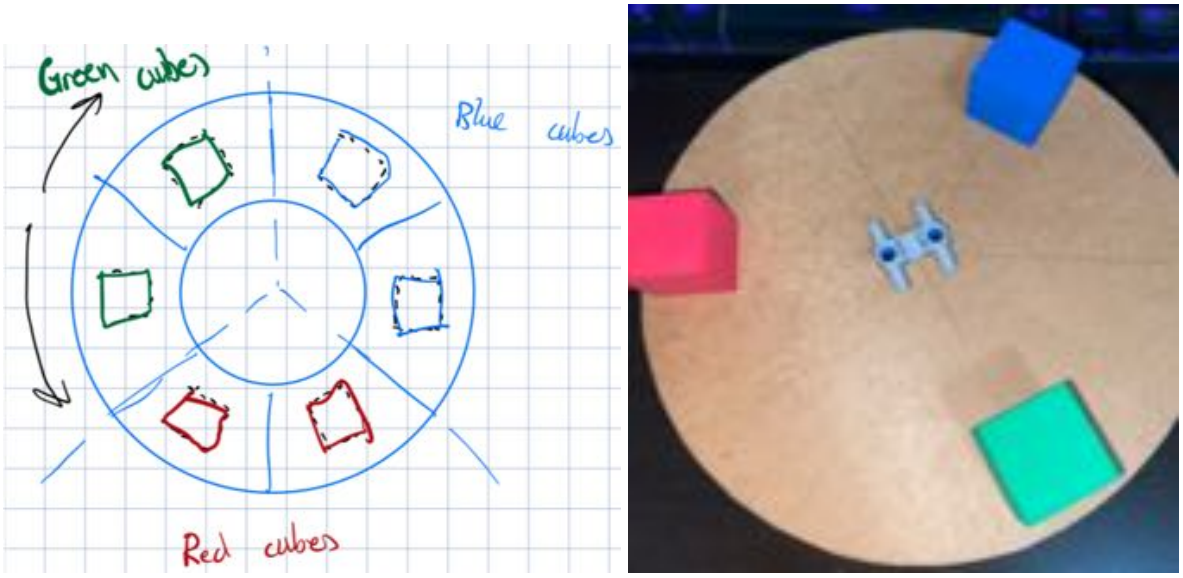


Figure 11. (a) Previous Rotating Platform Design b) Modified Rotating Platform Prototype without hole

(

4.2.3 Delivery System v2.0

Due to the addition of the piston in the input tower, the delivery system now consists of only one motor arm for delivery. From the prototyping process, the gate-like structure that would "guide" the

cubes to slip off at a specific angle was opted for a motor controlled arm which rotates about the axis perpendicular to the ground and hovers above the rotating platform, which pushes cubes into a new funnel structure which then guides the cubes to fall to the delivery area.

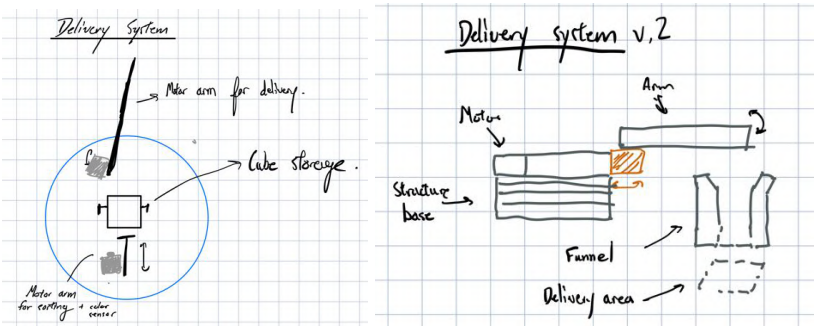


Figure 13. (a) Previous Delivery System Design (b) Modified Delivery System Design

4.2.5 Software Design v2.0

This version of the software is the first time there was actual software development done for the system. Therefore, there are no modifications specific to the software structure (i.e changes to specific functions), but rather specific to the role the software is now expected to play for the system. Subject to modifications made to the hardware of the system, i.e delivery system now having 1 arm instead of 2, the input tower having an additional piston mechanism, no physical storage, the software must accommodate by coding to move the correct number of motors (1 instead of 2), implement the piston mechanism, and digitally store the cubes. At this stage of the design process, the software implementation was constructed around each subsystem, with each having its own respective class.

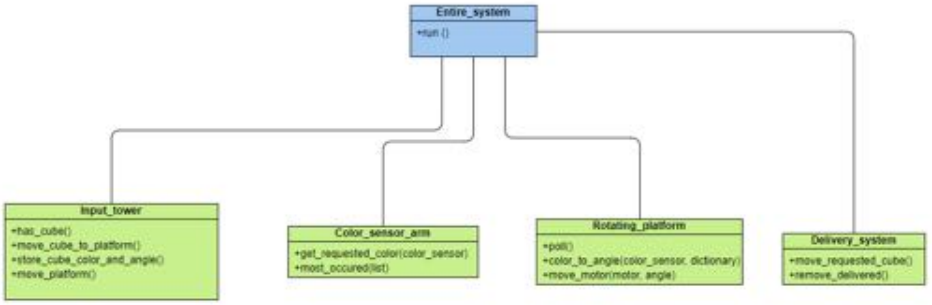


Figure 14. Software Design v2.0 UML Diagram

4.2.5.1 input_tower.py v2.0

The input_tower.py class defines 4 functions, as seen in Figure 14. With the cubes loaded into the tower, the color sensor starts polling to read the lowest cube's color (**has_cube()**). If the color detected is either red, green or blue, the *piston mechanism* is activated (motor rotates, which turns into translational motion of the piston) such that it pushes the cube from the stack to the platform (**move_cube_to_platform()**). The color of this cube, and its angular position on the rotating platform is then *digitally stored* into a dictionary (**store_cube_color_and_angle()**). The platform's motor will then rotate by 60° (and thereby rotating the platform by 60°), such that each cube may be placed equidistant from the other. This process loops until all cubes are placed onto the platform. See C.1.1 *Input Tower v2.0* for software flowchart.

4.2.5.2 rotating_platform.py v2.0

The rotating_platform.py class defines 2 functions, **color_to_angle(color_sensor, dict)** and **move_motor(motor, angle)**, which both take part in the delivery process. Once a request is received at the polling color sensor arm that is either red, green or blue, **color_to_angle(color_sensor, dict)** will return the angular position at which this cube is placed. Subsequently, **move_motor(motor, angle)** will move the rotating platform's motor based on the specified angular position such that the requested cube is brought in front of the delivery area, such that the delivery system may push it onto the delivery area. See C.1.1 *Rotating Platform v2.0* for software flowchart.

4.2.5.3 delivery_system.py v2.0

The delivery_system.py class defines 2 functions, **move_requested_cube()** and **remove_delivered()**, which control the now 1 delivery arm of the system. Once the cube has been placed in front of the delivery area (See 4.2.5.2), **move_requested_cube()** is called to activate the rotation of the motor, which in turn moves the arm to push the cube off of the platform into the funnel which would guide the cube to the delivery area. Subsequently, this angular position is removed using **remove_delivered()**, as there is no longer a cube in stock at this position. See C.1.1 *Delivery Systemv2.0* for software flowchart.

4.2.5.4 color_sensor_arm.py v2.0

The color_sensor_arm.py class defines 2 functions, **get_requested_color(color_sensor)** and **most_occured(list)**. The **get_requested_color(color_sensor)** returns a list of 20 samples of the color from the color sensor, which is then passed to **most_occured(list)**, which returns the color if it is either red, green or blue, and the string "unknown" otherwise. See C.1.1 *Color Sensor Arm v2.0* for software flowchart.

4.3 System Design v3.0 / Second Design Iteration

This third system design was developed based on various test results obtained during Week 3 of the design process.

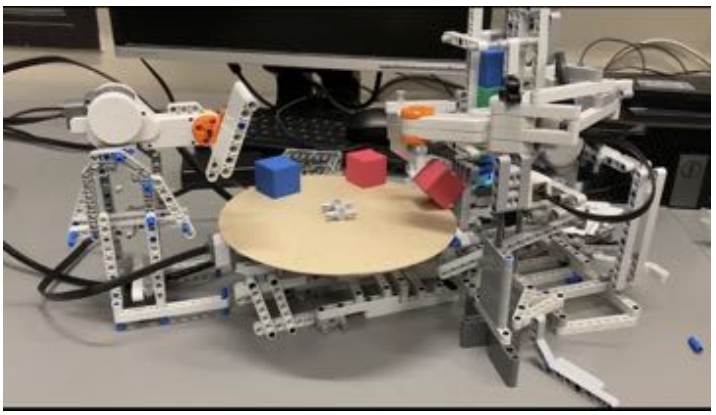


Figure 15. System Design Version 3.0

4.3.1 Input Tower v3.0

The initial input tower prototype during Week 3 is shown in Figure 18. As seen in the figure, the input tower walls are made of paper, with the piston having access only to the lowest cube in the stack. The piston is controlled by a motor, and the supporting structure of the tower is made of LegoEV3 pieces. However, when this implementation of the input tower was tested (See D.2.1), delivery of the 6 cubes was not successful, as the cubes in the stack would get stuck on their way down the tower. It was suspected that one of the main reasons for failure was the lack of guidance the input tower walls

(made of paper) provided the cubes as they were pulled down by gravity. Subject to this result, the paper was removed and LegoEV3 pieces were instead used to have stable walls. In addition, a motor gate component was implemented to control the front wall of the input tower to guide the cubes downward, such that it would open and close to better accommodate the size and placement of each cube in the stack as it fell down the tower.

Figure 16. (a) Input Tower Initial Implementation (b) Input Tower With Modifications

4.3.2 Rotating Platform v3.0

The height of the rotating platform was increased to match the height of the input tower.

Figure 17. (a) Previous Height of Rotating Platform (b) Increased Height of Rotating Platform

The material of the platform was initially curved and frail which caused the cubes to fall off the ledge of the platform during the storing process. The material of the platform was replaced by a sturdier and flatter cardboard which could catch the cubes from the input tower without errors

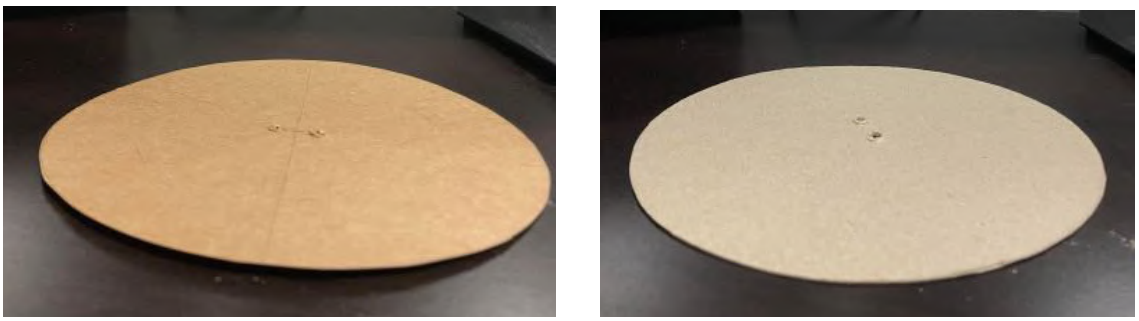


Figure 18. Platform Material Changes
(a) Previous platform material (b) New platform material

4.3.3 Delivery System v3.0

The delivery system's arm rotation axis was changed from rotating about an axis perpendicular to the ground, to rotating about an axis parallel to it. This change was made as the first rotation would have too big of a sweep, possibly knocking off unwanted cubes into the delivery area along with the requested one. Thus, rotation about an axis parallel to the ground was chosen for more precision.

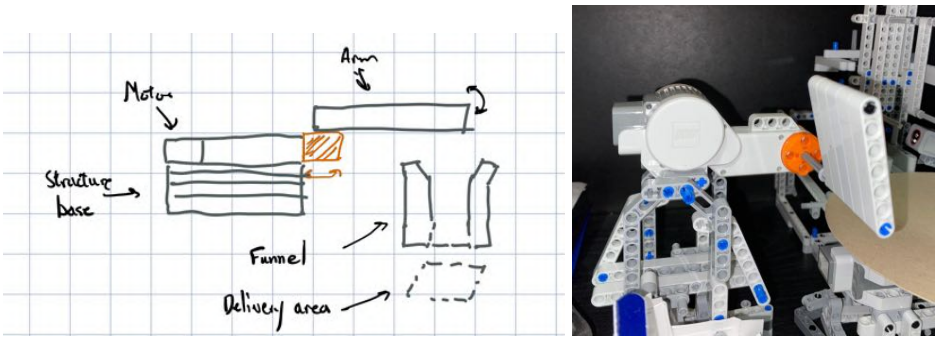


Figure 19. Delivery System Arm Rotation Axis
(a) Before with a vertical axis (b) After with a horizontal axis

4.3.4 Software Design v3.0

Following testing done during Week 3, the software structure evolved as follows.

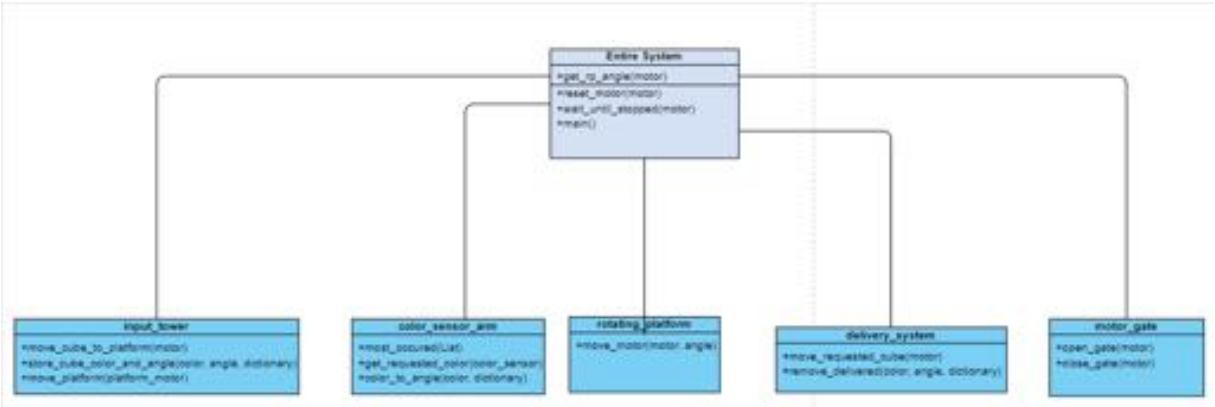


Figure 20. Software v3.0 UML Diagram

4.3.4.1 input_tower.py v3.0

As subsystems had started to get integrated during this week, the has_cube() function was removed from this class and the color sensor was polled from within the main function instead, using **color_sensor.get_requested_color(color_sensor)** instead. Subject to the input tower test conducted in Week 3 (See D.2.1), the pushing range of the piston arm was increased, such that cubes would be pushed far enough to avoid getting stuck on their way onto the platform. See C.1.2 *Input Tower v3.0* for software flowchart.

4.3.4.2 rotating_platform.py v3.0

The **poll()** and **color_to_angle(color_sensor)** functions were removed as they are now part of the color_sensor_arm.py class. See C.1.2 *Rotating Platform v3.0* for software flowchart.

4.3.4.3 delivery_system.py v3.0

Additional arguments to **remove_delivered(color, angle, dict)** function, to allow for removal of desired angles of colors from desired dictionary. See C.1.2 *Delivery System v3.0* for software flowchart.

4.3.4.4 color_sensor_arm v3.0

Additional function, **color_to_angle(color, angle)** moved from rotating_platform.py to color_sensor_arm.py for better organization. See C.1.2 *Color Sensor Arm v3.0* for software flowchart.

4.3.4.5 gate_motor.py v1.0

Two functions implemented to control the new component of the input tower, the motor gate: **open_gate(motor)** and **close_gate(motor)**, both of which control the front wall of the input tower. See C.1.2 *Motor Gate v1.0* for software flowchart.

4.4 System Design v4.0 / Third Design Iteration

This fourth system design reflects changes made subject to further testing results and to answer to system requirements not yet fulfilled.

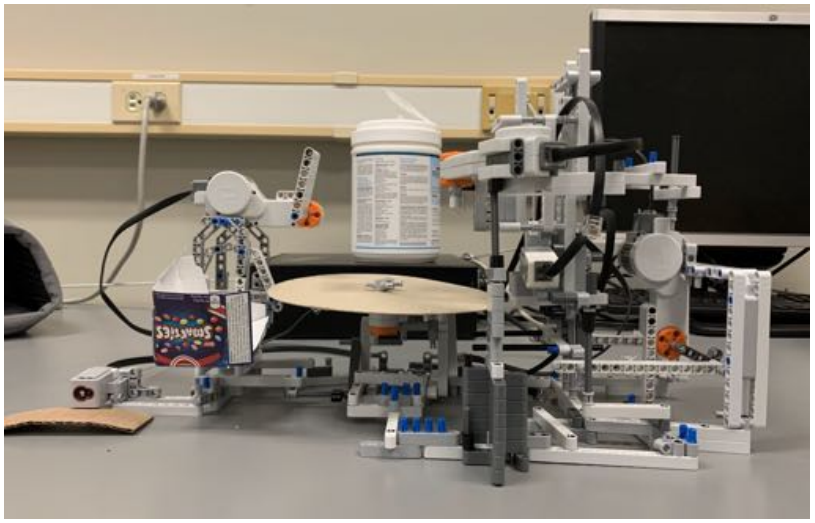


Figure 21. System Design Version 4.0

4.4.1 System-Level Modifications

Addition of Activation Touch Sensor

REQ4 of the system indicates that the store clerk must have a means to activate the system's sorting process. Thus, a touch sensor was connected to the BrickPi, and written into the main function of the software such that the system waits for a press of the touch sensor to start the sorting process. As shown in D.3.6, the touch sensor activation was integrated seamlessly into the system.

Figure 22. Activation Touch Sensor

Addition of Delivery and Request Areas
Through integration, the delivery and request areas could now have designated positions in the system. See Figure 21.

Full Integration of Subsystems
Full integration of subsystems to obtain the system in Figure 21 through LegoEV3 connecting structures was completed for version 4.0.

4.4.2 Input Tower v4.0

Piston Arm Modification
It was suggested after increasing the piston arm range in v3.0 to increase the distance between the input tower base's wall and the motor, as it was interfering with the piston arm being able to pull back fully during its full rotation. This modification was subsequently tested, as seen in D.3.1, which led to modifications made to the input tower's platform.

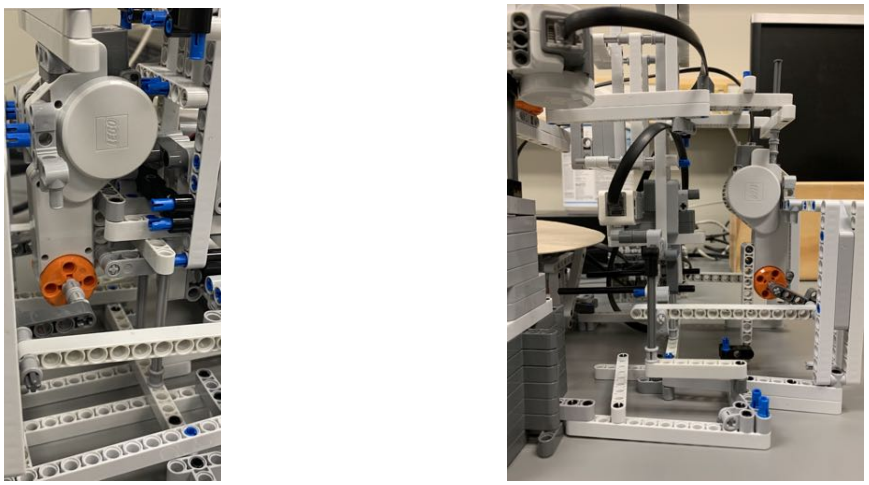


Figure 23. Input Tower Wall Distance Modification
(a) Wall Pre-Piston Range Modifications (b) Wall Post-Piston Range Modifications

Input Tower Platform Modifications
As seen in D.3.1, the earlier piston arm modification allowed for successful delivery of most cubes, but not all. The last cube would get knocked off of the input tower's cube platform before the piston arm pushed it off of the stack, and the piston arm itself often underwent a flicking motion, as if it were stuck on the hardware of the input tower. To account for this, lego pieces were added to the side and the surface of the input tower's cube platform. The first, to guide the piston arm as it undergoes its

motion, and the second, to avoid cubes bouncing off of the cube platform without being pushed by the piston. It was suspected that cubes bounced off due to the large drop as it was released from the motor gate's grip, because the input tower's cube platform was far from the position at which the motor gate would release the cube. Subsequent testing of the effect of the input tower modifications (See D.3.2 and D.3.3) show that both modifications were effective in avoiding the concerns brought up in previous tests.

Figure 24. Input Tower Platform Modifications
(a) input tower platform without guiding side pieces (b) input tower platform with guiding pieces

Figure 25 (c) previous height of input tower platform (d) modified height of input tower platform

Structural Lego Piece Removal
Unnecessary lego pieces were removed from the base of the input tower's structure to straighten and clean it.

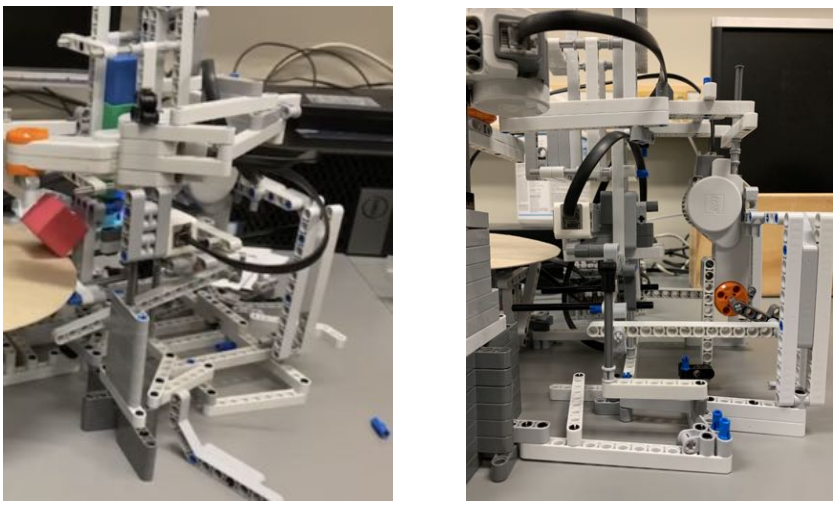


Figure 26. (a) pre-cleaning input tower (b) post-structural cleaning input tower

Rotating Platform & Input Tower Physical Dimension Modifications

As part of the integration process, the input tower and rotating platform subsystems were integrated and tested first. From this subsystem integration test result, two concerns were brought up: 1) the last cube's color was not detected by the color sensor due to it being outside of the color sensor's field of view, and 2) one cube rolled off of its intended section of the rotating platform (See D.3.4). To fix the second issue, the height of the rotating platform was raised, and it was pulled closer to the input tower, such that the drop of the cube from the input tower onto the platform would be smoother, thus decreasing its momentum, which would keep it on the edge of the platform instead of it rolling on the surface of the platform.

Figure 27. Modified Rotating Platform Height

Color Sensor Positioning

In order to address the first concern brought up during the subsystem integration (See D.3.4) between the input tower and the rotating platform, the color sensor was placed such that it would be angled upwards and created the necessary space between the sensor and cube for proper functionality. This was further tested to be successful (See D.4.1).T

Figure 28. Color Sensor Angular Position

4.4.3 Rotating Platform v4.0

The height of the rotating platform was increased. For specifics and motives, see the above.

4.4.4 Delivery System v4.0

The delivery system has now an attached delivery area (See Figure 27).

4.4.5 Color Sensor Arm v2.0

The color sensor arm is now attached through LegoEV3 pieces to its adjacent subsystems, and is directly in front of the request area.

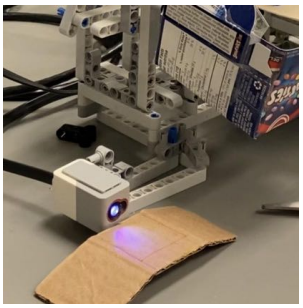


Figure 29. Integrated Color Sensor Arm

4.4.6 Software Design v4.0

Code was added to main_robot_ferris.py to implement the activation functionality of the touch sensor. The system sleeps until the touch sensor is pressed. All other classes remain unchanged.

4.5 Final System Design / Fourth Design Iteration

4.5.1 Input Tower v5.0

Piston Arm Surface Modifications

During the input tower's color sensor testing (See D.4.1), the piston arm was noticeably getting stuck during its motion. Suspecting the grey pieces of the piston arm were getting stuck on the holes of the LegoEV3 pieces on the input tower's cube platform, the grey pieces were replaced by rounder, black pieces. This did not seem to resolve the issue, as subsequent trials showed that the piston would still get stuck on the holes of the platform (see D.4.2). Thus, clear tape was added on the surface of the holes, such that it would smooth out the surface (see D.4.3). It was noted in D.4.3 that the entire structure was lifting as the piston arm was moving, as there was no mechanism to stabilize the two rods on which the piston arm was performing its translational motion. Thus, the pressure of the piston going back and forth would slightly lift up the structure at each run. To counter this, a lego stabilizing structure was added to the two rods. As seen in D.4.4 Pre-Final Demo testing results, the addition of tape and the stabilizing structure resolved the issue of the structure getting stuck/lifting.

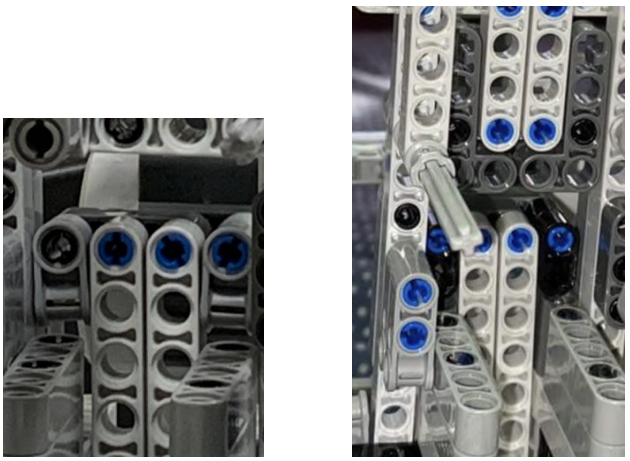


Figure 30.
(a) Previous piston arm with grey pieces (b) Input tower with modifications

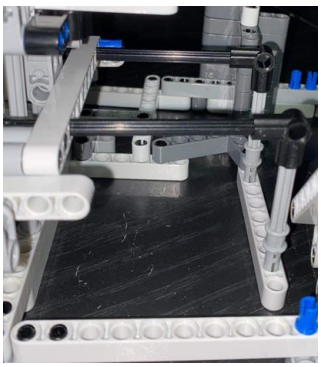


Figure 31 Rod Stabilizing Structure

Input Tower Wall Additional Padding

During pre-final demo testing performed during Week 5, it was noted that at times the cubes landed on their corners instead of their flat surfaces. It was suspected that the reason for this was that there was too much space on the back surface of the input tower's wall, such that cubes had enough space to rotate and fall on their corners instead of on their flat surfaces. Thus, additional lego pieces were added near the bottom of the input tower's back wall, such that cubes were forced into staying in place. (See Figure 41 (b)) As seen in D.4.5, this final modification allowed for a system that could successfully handle the final demo conditions.

Other subsystems and the software implementation of the system remain unchanged from v4.0. Please refer to 3.0 for details on the final system design.

5.0 Tests and Results

Tests were done for every modification that was made for the system in order to make sure that the system/subsystem works and satisfies the system specifications. The test schedules can be seen in D.0 in which the tests, testers and the date at which the test was conducted is shown. All the detailed test procedures are shown in APPENDIX D. Furthermore, the tests conducted in week two are shown in D.1, the tests conducted in week three are shown in D.2, the tests conducted in week four are shown in D.3 and the tests conducted in week five are shown in D.4.

5.1 Trial runs

Initial testing did not document trial runs. However, it was brought by the Senior Engineer that test procedures should incorporate multiple trials. The change is shown in the subsequent tests.

6.0 System Performance and Limitations

6.1 Achieving system requirements

The system meets all the requirements listed in section 2.2.

6.1.1 ***REQ 1.***
The system must be able to physically accommodate the weight of 6 foam cubes.
The system is designed to operate with 6 foam cubes only. However, it is possible to increase the size of the input tower and rotating platform, and change the software correspondingly, such that it is able to accommodate any number of cubes.
6.1.2 ***REQ 2.***
The system must account for the area occupied by 6 foam cubes.
See 6.1.1
6.1.3 ***REQ 3.***
The system must be able to identify 3 uniquely colored foam cubes.
The system is able to identify 3 uniquely colored foam cubes through the software. If necessary, the system can be modified to identify and store more varieties of color as only the color and angle are needed to store a cube.
6.1.4 ***REQ 4.***
The system must provide the store clerk a simple way of activating the pre-"sorting" process.
The system uses an explicit touch sensor to activate the pre-"sorting" process.
6.1.5 ***REQ 5.***
The pre-"sorting" process of the system must be completed within a 2 minute margin.
The system sorts the unsorted stack well below a 2 minute margin. Furtherly discussed in section 6.2.
6.1.6 ***REQ 6.***
The system's item retrieval process must be easily operable by a single store clerk.
The system's retrieval process is operable by a single store clerk as the clerk needs only to place a cube at the request area and wait to pick up the delivered cube at the delivery area.
6.1.7 ***REQ 7.***
The system must be able to read and understand the requested color (i.e detect the color of the cube placed onto the request area).
The system is able to consistently deliver the requested cube. See section 6.2.2
6.1.8 ***REQ 8.***
The system must be able to retrieve foam cubes from the storage area according to specified colors to then deliver them onto the delivery area.
See above and section 6.2.2
6.1.9 ***REQ 9.***
The system must be able to complete the request to delivery process within a time margin of 5 seconds or less.
The system is able to complete the delivery request well under 5 seconds. See section 6.2.2.

6.2 Performance

6.2.1 Sorting Performance

Sorting time was tested in test D.3.7 and yielded a result of 24.53 seconds, which is well under the required 120 seconds (2 minutes).

Sorting consistency was tested in the overall second pre-demo test (D.4.5) and all ten trial runs went successfully, attesting to the efficiency of the system.

6.2.2 Delivery Performance

Delivery time was tested in test D.3.9 and resulted with an average delivery time of 2.186 seconds, and ranged from 1.79 to 2.89 seconds.

Delivery consistency was tested as well in test D.4.5 and all ten trial runs were successful, attesting to the efficiency of the system.

6.3 Usage

The system is ideal for stores with a large variety, but a low quantity of products, as it is able to store digitally every unit, but uses a centralized rotating storage which is limited by size.

6.4 Limitations

Storage

Prof. Ferri(e)'s Wheel is able to hold only six cubes at a time, due to the size of the rotating platform and the input tower.

However, it is possible to increase the size of the rotating platform and the vertical storage of the input tower, as well as the software to account for the change in angle distribution, in order to accommodate more foam cubes.

Fragile

Prof. Ferri(e)'s wheel, though sturdy if untouched, is sensitive to external perturbations as the rotating platform is exposed and unprotected. Should the system be slanted, or exposed to external touch, cubes might be displaced from the rotating platform and disrupt the delivery process.

Loading

Prof. Ferri(e)'s wheel needs its unsorted cubes to be loaded into its input tower, which might require some precision or instructions as the cubes must fit into the input tower in the same orientation such that they are able to stack upon each other.

APPENDIX A

CONTENTS

A.1 Member Capabilities

Member	Capabilities
David	Mini project: - Did the structure subsystem (handling all the lego pieces) - Can draw sketches with IPad Else: - Completed COMP250, familiar with Java, okay with python. Decent in coding. - Rocket Team, Antenna arrays knowledge - Ultimate Frisbee for a few years - Lives near campus, can pull all-nighters if needed.
Muqto	Mini-Project: - Integration of components - tested all the sensors - wrote software for the motor and gyro for integration. Else: - took comp 250, ecse 202, therefore, i am comfortable with coding. - I know java and python.
Nour	Mini-Project: - Documentation of the progress of the robot - Time management and coordination between the group and meeting with the TA. - Helped with the software programming of the robot itself. Else: - Took ECSE 202 and currently taking COMP 250; familiar with Java but very beginner in Python. - Worked on a Robotics project in high school but had coding guidance.
Ryan	Mini-Project: - Documentation of the progress of the robot - Gantt chart Else: I took comp 250, ecse 202, therefore, i know java and python.
Lawi	Mini-Project: - Documentation of the progress of the robot - Gantt chart - Helped with the software programming of the robot itself (Python Knowledge) - Building the robot designs Else: - Took the following classes: ECSE202, COMP250, COMP206, COMP251, ECSE415 Currently.
Miiyu	Mini-Project: - Integration of software components

	- Wrote software for color detection - Tested color & us & touch sensors Else: - Project Lead (McGill Rocket Team): Project management experience (budget, timeline, organizational documents, design process) - ECSE202 (comfortable with Java) - Contributed to software projects in MRT: Graphical User Interface for Rocket Telemetry and interfacing with hardware components (Front-End and Backend) - Have used Python before for a computer vision project (Numpy, YOLOv3)

A.2 Member Availabilities

Member	Availabilities
David	- March 17 ECSE307 Midterm - Same week, have Lab demo for ECSE 324 - First week of April very busy, not able to do anything. (Until 9th of April) Weekly: - Usually available at night, after 6pm. - Friday afternoon and might not be free.
Muqto	- Midterm on march 17th and 24th. - Available after 5:30 on the weekdays. - Available on weekends
Nour	- March 25 and 28 Midterms - Free on weekends, preferably earlier in the day or later at night - Free after 6pm for week days
Ryan	- March 17 ECSE307 Midterm - March 14-30 ECSE354 Midterm - Capstone project Weekly: - Usually available at night, after 6pm.
Lawi	- Easily modify my availability - Live close to campus - No midterms/finals only projects and assignments
Miiyu	- Midterm on March 17 (ECSE307) - Midterm on 24th & 25th of March Weekly: - Available during the day, I have big breaks in between classes from 8am to 5pm - I can also stay at school or join meetings during the afternoon

A.3 Budget Modifications

A.3.1 Week 1 Budget

Member	**Week 1 hours**	**Week 2 hours**	**Week 3 hours**	**Week 4 hours**	**Week 5 hours**	**Total Hours**
David	**Total hours week 1 =** 10.166h	**Total hours week 2 =** 9.1h	**Total hours week 3 =** 6h	**Total hours week 4 =** 11.75	**Total hours week 5 =** 5	45
Muqto	**Total hours week 1 =** 10.166h	**Total hours week 2 =** 10.2h	**Total hours week 3 =** 7.5	**Total hours week 4 =** 6.41	**Total hours week 5 =** 7.72	45
Nour	**Total hours week 1 =** 4h	**Total hours week 2 =** 8.75	**Total hours week 3 =** 8.25	**Total hours week 4 =** 12.67	**Total hours week 5 =** 7.33	45
Ryan	**Total hours week 1 =** 3h	**Total hours week 2 =** 8.5	**Total hours week 3 =** 9.5h	**Total hours week 4 =** 12.41	**Total hours week 5 =** 8.58	45
Lawi	**Total hours week 1 =** 4h	**Total hours week 2 =** 9.5	**Total hours week 3 =** 9.5	**Total hours week 4 =** 12.41	**Total hours week 5 =** 6.75	45
Miiyu	**Total hours week 1 =** 10.166h	**Total hours week 2 =** 10.167h	**Total hours week 3 =** 3.75h	**Total hours week 4 =** 2.167h	**Total hours week 5 =** 15.75	45
Total hours budgeted	45*6 = 270					

A.3.2 Week 2 Budget

The Week 2 Budget reflects changes made after the senior engineer advised us that although member availability is to be considered, we are to avoid extreme discrepancies between members' hours per week. Members of the team agreed to work on the following 2 points, once they had reflected on what led to such big gaps in member hours:

1. Open communication: the goal is for all members of the team to feel comfortable and excited about working on the project. To do so, we can focus on nurturing an open and supportive team dynamic.
2. Adequate task delegation and time management: So as to avoid members burning the night oil the night that the weekly deliverable is due, we have decided as a team to finish our respective parts of the weekly deliverable report at least 24 hours before it is due, so that all members of the team can then review the report together before submission in order to resolve any last minute issues together.

Considering the above points, we have made the following changes to our project budget and plans.

David's Hours

Week #	**Tasks Completed**						**Total Hours per Week**
Week 1							10.166
Week 2	Input Tower Prototyping = 3	Delivery System = 3	Writing Test Procedures = 1	Weekly Deliverable Writing = 1	Design Iteration Meeting = 0.834	Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) = 1	9.834
Week 3	Integration Testing & Debugging = 2	Beta Demo & Post-Beta Demo Reflections = 1	Post-Beta Demo Debugging = 3	Weekly Deliverable Writing = 2		Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) = 1	9
Week 4	Final Tests/Testing = 2	Final Report Writing = 2	Final Presentation (Draft) = 2	Weekly Deliverable = 2		Weekly Scheduled Meetings (Senior Engineer	9

						Meetings & Weekly Status Updates) = 1	
Week 5	Final Demo = 1	Final Report Writing = 4	Final Project Presentation = 2			Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) = 0	7

Lawi's Hours

Week #	**Tasks Completed (hrs)**						**Total Hours per Week (hrs)**
Week 1							4
Week 2	Input Tower Prototyping = 3	Delivery System = 3	Writing Test Procedures = 1	Weekly Deliverable Writing = 1	Design Iteration Meeting = 0	Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) = 1	9
Week 3	Integration Testing & Debugging = 4	Beta Demo & Post-Beta Demo Reflections = 1	Post-Beta Demo Debugging = 4	Weekly Deliverable Writing = 2		Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) = 1	12
Week 4	Final Tests/Test	Final Report	Final Presentati	Weekly Deliverab		Weekly Schedule	10

	ing = 3	Writing = 2	on (Draft) = 2	le = 2		d Meetings (Senior Engineer Meetings & Weekly Status Updates) = 1	
Week 5	Final Demo = 1	Final Report Writing = 4	Final Project Presentation = 5			Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) = 0	10

Muqto's Hours

Week #	**Tasks Completed (hrs)**					**Total Hours per Week (hrs)**
Week 1						10.166
Week 2	Rotating Platform Prototyping = 5	Writing Test Procedures = 1	Weekly Deliverable Writing = 2	Design Iteration Meeting = 0.834	Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) = 1	9.834
Week 3	Integration Testing & Debugging = 2	Beta Demo & Post-Beta Demo Reflections = 1	Post-Beta Demo Debugging = 2	Weekly Deliverable Writing = 2	Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) = 1	8
Week 4	Final	Final	Final	Weekly	Weekly	8

	Tests/Testing = 2	Report Writing = 2	Presentation (Draft) = 2	Deliverable = 1	Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) = 1	
Week 5	Final Demo = 1	Final Report Writing = 5	Final Project Presentation = 3		Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) = 0	9

Miiyu's Hours

Week #	**Tasks Completed (hrs)**						**Total Hours per Week (hrs)**
Week 1							10.166
Week 2	Rotating Platform Prototyping = 2	Preparing Meeting Minutes = 1	Weekly Deliverable Writing = 5	Design Iteration Meeting = 0.834		Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) = 1	9.834
Week 3	Integration Testing & Debugging = 2	Beta Demo & Post-Beta Demo Reflections = 1	Post-Beta Demo Debugging = 1	Weekly Deliverable Writing = 1	Preparing Meeting Minutes = 1	Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) = 1	7

Week 4	Final Tests/Testing = 1	Final Report Writing = 2	Final Presentation (Draft) = 2	Weekly Deliverable = 2	Preparing Meeting Minutes = 1	Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) = 1	9
Week 5	Final Demo = 1	Final Report Writing = 5	Final Project Presentation = 3			Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) = 0	9

Nour's Hours

Week #	**Tasks Completed (hrs)**					**Total Hours per Week (hrs)**
Week 1						4
Week 2	Motor Testing = 5	Writing Test Procedures = 1	Weekly Deliverable Writing = 5	Design Iteration Meeting = 0	Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) = 1	12
Week 3	Integration Testing & Debugging = 2	Beta Demo & Post-Beta Demo Reflections = 1	Post-Beta Demo Debugging = 2	Weekly Deliverable Writing = 3	Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) =	9

					1	
Week 4	Final Tests/Testing = 2	Final Report Writing = 2	Final Presentation (Draft) = 3	Weekly Deliverable = 2	Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) = 1	10
Week 5	Final Demo = 1	Final Report Writing = 5	Final Project Presentation = 4		Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) = 0	10

Ryan's Hours

Week #	**Tasks Completed (hrs)**					**Total Hours per Week (hrs)**
Week 1						3
Week 2	Color Sensor Arm Prototyping = 5	Writing Test Procedures = 1	Weekly Deliverable Writing = 1	Design Iteration Meeting = 0	Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) = 1	8
Week 3	Integration Testing & Debugging = 2	Beta Demo & Post-Beta Demo Reflections = 1	Post-Beta Demo Debugging = 5	Weekly Deliverable Writing = 3	Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) =	12

					1	
Week 4	Final Tests/Testing = 2	Final Report Writing = 2	Final Presentation (Draft) = 2	Weekly Deliverable = 4	Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) = 1	11
Week 5	Final Demo = 1	Final Report Writing = 5	Final Project Presentation = 5		Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) = 0	11

A.3.3 Week 3 Budget

David's Hours

Week #	Tasks Completed						Total Hours per Week
Week 1							10.166
Week 2	~~Input Tower Prototyping = 3~~	~~Delivery System = 3~~	~~Writing Test Procedures = 1~~	~~Weekly Deliverable Writing = 1~~	~~Design Iteration Meeting = 0.834~~	~~Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) = 1~~	9.834
Week 3	~~Integration Testing & Debugging = 2~~	~~Beta Demo & Post-Beta Demo Reflections = 1~~	~~Post-Beta Demo Debugging = 3~~	~~Weekly Deliverable Writing = 2~~		~~Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) = 1~~	~~9~~ 22 (13h overshoot)
Week 4	Final Tests/Testing = ~~2~~ 1	Final Report Writing = ~~2~~ 1	Final Presentation (Draft) = ~~2~~ 1	Weekly Deliverable = ~~2~~ 1		Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) = 1	~~9~~ 5
Week 5	Final Demo = 1	Final Report Writing = ~~4~~ 1	Final Project Presentation = ~~2~~ 1			Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status	~~7~~ 3

						Updates) = 0	

Lawi's Hours

Week #	**Tasks Completed (hrs)**						**Total Hours per Week (hrs)**
Week 1							4
Week 2	~~Input Tower Prototyping = 3~~	~~Delivery System = 3~~	~~Writing Test Procedures = 1~~	~~Weekly Deliverable Writing = 1~~	~~Design Iteration Meeting = 0~~	~~Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) = 1~~	9
Week 3	~~Integration Testing & Debugging = 4~~	~~Beta Demo & Post-Beta Demo Reflections = 1~~	~~Post-Beta Demo Debugging = 4~~	~~Weekly Deliverable Writing = 2~~		~~Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) = 1~~	12
Week 4	Final Tests/Testing = 3	Final Report Writing = 2	Final Presentation (Draft) = 2	Weekly Deliverable = 2		Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) = 1	10
Week 5	Final Demo = 1	Final Report Writing = 4	Final Project Presentation = 5			Weekly Scheduled Meetings (Senior Engineer	10

						Meetings & Weekly Status Updates) = 0	

Muqto's Hours

Week #	**Tasks Completed (hrs)**					**Total Hours per Week (hrs)**
Week 1						10.166
Week 2	~~Rotating Platform Prototyping = 5~~	~~Writing Test Procedures = 1~~	~~Weekly Deliverable Writing = 2~~	~~Design Iteration Meeting = 0.834~~	~~Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) = 1~~	9.834
Week 3	~~Integration Testing & Debugging = 2~~	~~Beta Demo & Post-Beta Demo Reflections = 1~~	~~Post-Beta Demo Debugging = 2~~	~~Weekly Deliverable Writing = 2~~	~~Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) = 1~~	~~8~~ 28 (20 hr overshoot)
Week 4	Final Tests/Testing = 2	Final Report Writing = ~~2~~ 1	Final Presentation (Draft) = ~~2~~ 1	Weekly Deliverable = 1	Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) = 1	~~8~~ 6
Week 5	Final Demo = 1	Final Report Writing = ~~5~~ 1	Final Project Presentation = ~~3~~ 1		Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly	~~9~~ 3

					Status Updates) = 0	

Miiyu's Hours

Week #	**Tasks Completed (hrs)**						**Total Hours per Week (hrs)**
Week 1							10.166
Week 2	~~Rotating Platform Prototyping = 2~~	~~Preparing Meeting Minutes = 1~~	~~Weekly Deliverable Writing = 5~~	~~Design Iteration Meeting = 0.834~~		~~Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) = 1~~	9.834
Week 3	~~Integration Testing & Debugging = 2~~	~~Beta Demo & Post-Beta Demo Reflections = 1~~	~~Post-Beta Demo Debugging = 1~~	~~Weekly Deliverable Writing = 1~~	~~Preparing Meeting Minutes = 1~~	~~Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) = 1~~	~~7~~ 10
Week 4	Final Tests/Testing = 1	Final Report Writing = ~~2~~ 1	Final Presentation (Draft) = ~~2~~ 1	Weekly Deliverable = 2	Preparing Meeting Minutes = 1	Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) = 1	~~9~~ 7
Week 5	Final Demo = 1	Final Report Writing = ~~5~~ 4	Final Project Presentation = 3			Weekly Scheduled Meetings (Senior	~~9~~ 8

						Engineer Meetings & Weekly Status Updates) = 0	

Nour's Hours

Week #	**Tasks Completed (hrs)**					**Total Hours per Week (hrs)**
Week 1						4
Week 2	~~Motor Testing = 5~~	~~Writing Test Procedures = 1~~	~~Weekly Deliverable Writing = 5~~	~~Design Iteration Meeting = 0~~	~~Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) = 1~~	12
Week 3	~~Integration Testing & Debugging = 2~~	~~Beta Demo & Post-Beta Demo Reflections = 1~~	~~Post-Beta Demo Debugging = 2~~	~~Weekly Deliverable Writing = 3~~	~~Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) = 1~~	~~9~~ 8
Week 4	Final Tests/Testing = 2	Final Report Writing = 2	Final Presentation (Draft) = 3	Weekly Deliverable = ~~2~~ 3	Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) = 1	~~10~~ 11
Week 5	Final Demo = 1	Final Report Writing = 5	Final Project Presentation = 4		Weekly Scheduled Meetings (Senior Engineer Meetings &	10

					Weekly Status Updates) = 0	

Ryan's Hours

Week #	**Tasks Completed (hrs)**					**Total Hours per Week (hrs)**
Week 1						3
Week 2	~~Color Sensor Arm Prototyping = 5~~	~~Writing Test Procedures = 1~~	~~Weekly Deliverable Writing = 1~~	~~Design Iteration Meeting = 0~~	~~Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) = 1~~	~~8~~10
Week 3	~~Integration Testing & Debugging = 2~~	~~Beta Demo & Post-Beta Demo Reflections = 1~~	~~Post-Beta Demo Debugging = 5~~	~~Weekly Deliverable Writing = 3~~	~~Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) = 1~~	~~12~~ 6
Week 4	Final Tests/Testing = 2	Final Report Writing = ~~2~~ 3	Final Presentation (Draft) = 2	Weekly Deliverable = 4	Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) = 1	~~9~~ 12
Week 5	Final Demo = 1	Final Report Writing = ~~5~~ 7	Final Project Presentation = ~~5~~ 6		Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status	~~11~~ 14

					Updates) = 0	

A.3.4 Week 4 Budget

Following Week 3, which was very hardware-focused and therefore highly time-consuming, the budget for the following week was modified such that members who had overshot greatly no longer had to work as many hours, and other members could compensate. It is to note that during week 4, 1 member tested positive for Covid (David), as well as one member who came into contact with someone who tested positive for Covid. These two members therefore could not take part in in person tasks.

David's Hours

Week #	**Tasks Completed**						**Total Hours per Week**
Week 1							10.166
Week 2	~~Input Tower Prototyping = 3~~	~~Delivery System = 3~~	~~Writing Test Procedures = 1~~	~~Weekly Deliverable Writing = 1~~	~~Design Iteration Meeting = 0.834~~	~~Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) = 1~~	9.834
Week 3	~~Integration Testing & Debugging = 2~~	~~Beta Demo & Post-Beta Demo Reflections = 1~~	~~Post-Beta Demo Debugging = 3~~	~~Weekly Deliverable Writing = 2~~		~~Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) = 1~~	~~9~~ 22 (13h overshoot)
Week 4	~~Final Tests/Testing = 2 1~~	~~Final Report Writing = 2 1~~	~~Final Presentation (Draft) = 2 1~~	~~Weekly Deliverable = 2 1~~		~~Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) = 1~~	~~5~~ 5
Week 5	Final Demo = 1	Final Report	Final Project			Weekly Schedule	~~7~~ 3

		Writing = ~~4~~ 1	Presentation = ~~2~~ 1			d Meetings (Senior Engineer Meetings & Weekly Status Updates) = 0	

Lawi's Hours

Week #	**Tasks Completed (hrs)**						**Total Hours per Week (hrs)**
Week 1							4
Week 2	~~Input Tower Prototyping = 3~~	~~Delivery System = 3~~	~~Writing Test Procedures = 1~~	~~Weekly Deliverable Writing = 1~~	~~Design Iteration Meeting = 0~~	~~Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) = 1~~	9
Week 3	~~Integration Testing & Debugging = 4~~	~~Beta Demo & Post-Beta Demo Reflections = 1~~	~~Post-Beta Demo Debugging = 4~~	~~Weekly Deliverable Writing = 2~~		~~Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) = 1~~	12
Week 4	~~Final Tests/Testing = 3~~	~~Final Report Writing = 2~~	~~Final Presentation (Draft) = 2~~	~~Weekly Deliverable = 2~~		~~Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates)~~	~~10~~ 1

						~~= 1~~	
Week 5	Final Demo = 1	Final Report Writing =~~4~~ ~~3~~ 12	Final Project Presentation =~~5~~ 4	Pre-final Demo Testing = 2		Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) = 0	~~10~~ 19

Muqto's Hours

Week #	**Tasks Completed (hrs)**					**Total Hours per Week (hrs)**
Week 1						10.166
Week 2	~~Rotating Platform Prototyping = 5~~	~~Writing Test Procedures = 1~~	~~Weekly Deliverable Writing = 2~~	~~Design Iteration Meeting = 0.834~~	~~Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) = 1~~	9.834
Week 3	~~Integration Testing & Debugging = 2~~	~~Beta Demo & Post-Beta Demo Reflections = 1~~	~~Post-Beta Demo Debugging = 2~~	~~Weekly Deliverable Writing = 2~~	~~Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) = 1~~	~~8~~ 28 (20 hr overshoot)
Week 4	~~Final Tests/Testing = 2~~	~~Final Report Writing = 2 1~~	~~Final Presentation (Draft) = 2 1~~	~~Weekly Deliverable = 1~~	~~Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) = 1~~	~~6~~ 5

Week 5	Final Demo = 1	Final Report Writing = ~~5~~ 0	Final Project Presentation = ~~3~~ 0	Color Sensor Position Test = 1	Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) = 0	~~9~~ 2

Miiyu's Hours

Week #	**Tasks Completed (hrs)**						**Total Hours per Week (hrs)**
Week 1							10.166
Week 2	~~Rotating Platform Prototyping = 2~~	~~Preparing Meeting Minutes = 1~~	~~Weekly Deliverable Writing = 5~~	~~Design Iteration Meeting = 0.834~~		~~Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) = 1~~	9.834
Week 3	~~Integration Testing & Debugging = 2~~	~~Beta Demo & Post-Beta Demo Reflections = 1~~	~~Post-Beta Demo Debugging = 1~~	~~Weekly Deliverable Writing = 1~~	~~Preparing Meeting Minutes = 1~~	~~Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) = 1~~	~~7~~ 10
Week 4	~~Final Tests/Testing = 1~~	~~Final Report Writing = 2 1~~	~~Final Presentation (Draft) = 2 1~~	~~Weekly Deliverable = 2~~	~~Preparing Meeting Minutes = 1~~	~~Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status~~	~~7~~ 9

						~~Updates) = 1~~	
Week 5	Final Demo = 1	Final Report Writing = ~~4~~ 2	Final Project Presentation = ~~3~~ 2	Color Sensor Position Test = 1		Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) = 0	~~9~~ 6

Nour's Hours

Week #	**Tasks Completed (hrs)**					**Total Hours per Week (hrs)**
Week 1						4
Week 2	~~Motor Testing = 5~~	~~Writing Test Procedures = 1~~	~~Weekly Deliverable Writing = 5~~	~~Design Iteration Meeting = 0~~	~~Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) = 1~~	12
Week 3	~~Integration Testing & Debugging = 2~~	~~Beta Demo & Post-Beta Demo Reflections = 1~~	~~Post-Beta Demo Debugging = 2~~	~~Weekly Deliverable Writing = 3~~	~~Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) = 1~~	~~9~~ 8
Week 4	~~Final Tests/Testing = 2~~	~~Final Report Writing = 2~~	~~Final Presentation (Draft) = 3~~	~~Weekly Deliverable = 2 3~~	~~Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) =~~	~~11~~ 17

					~~1~~	
Week 5	Final Demo = 1	Final Report Writing = ~~5~~ 2	Final Project Presentation = ~~4~~ 1		Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) = 0	~~10~~ 4

Ryan's Hours

Week #	**Tasks Completed (hrs)**					**Total Hours per Week (hrs)**
Week 1						3
Week 2	~~Color Sensor Arm Prototyping = 5~~	~~Writing Test Procedures = 1~~	~~Weekly Deliverable Writing = 1~~	~~Design Iteration Meeting = 0~~	~~Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) = 1~~	~~8~~10
Week 3	~~Integration Testing & Debugging = 2~~	~~Beta Demo & Post-Beta Demo Reflections = 1~~	~~Post-Beta Demo Debugging = 5~~	~~Weekly Deliverable Writing = 3~~	~~Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) = 1~~	~~12~~ 6
Week 4	~~Final Tests/Testing = 2~~	~~Final Report Writing = 2 3~~	~~Final Presentation (Draft) = 2~~	~~Weekly Deliverable = 4~~	~~Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) = 1~~	~~12~~ 10.5

Week 5	Final Demo = 1	Final Report Writing = ~~7~~ 7.5	Final Project Presentation = ~~6~~ 7		Weekly Scheduled Meetings (Senior Engineer Meetings & Weekly Status Updates) = 0	~~14~~ 15.5

A.3.5 Week 5 Budget

The week 5 budget was modified such that each member gets close to the 45h/person mark during the fifth week. In this way, member contributions would be balanced in total for the design process. It is to note however that actual hours completed differ from those budgeted.

Member	**Hours (h)**
David	2.666
Lawi	8
Muqto	2.166
Miiyu	4.166
Nour	4
Ryan	15.5

A.4 Hours per Task Weekly Modifications

A.4.1 Week 1 List of Tasks

Milestones	Task	Time Estimate (hours)
Week 1 Deliverable (March 8th, 2022)	Team Management/Project Management Plan	6
	Translation of client needs into requirements, constraints, specifications	6
	Design Brainstorm	8
	Weekly Deliverables	17
Week 2 Deliverable (March 14th, 2022)	Test the motor accuracy	1
	Meeting Agenda for meeting w/ Senior Engineer	0.5
	Input Tower Prototype	6
	Rotating Platform Prototype	6
	Delivery System Prototype	6
	Color Sensor Arm Prototype	1
	Integration Testing Procedure	2
	Subsystem Debugging	2
	Weekly Deliverables	17
	Integration Testing	20
Beta Demo Day (March 18th, 2022)	Debugging Post-Integration Testing	6
	Beta demo	3

Week 3 Deliverable (March 21st, 2022)	Post-Beta Demo Reflections	12
	Meeting Agenda for meeting w/ Senior Engineer	0.5
	Debugging period Post-Beta demo	3
	Preparing final design report	10
	Weekly Deliverables	17
Week 4 Deliverable (March 28th, 2022)	Final Tests	11.5
	Meeting Agenda for meeting w/ Senior Engineer	0.5
	Final Report Writing	17
	Final Project Presentation Preparation (Draft Presentation)	6
	Week Deliverables	17
Final Demo Day (April 1st, 2022)	Final Report Writing	17
	Final Project Presentation Preparation	24
Week 5 Deliverable (April 4th, 2022)	Final Project Documents & Submission	9
Final Design Review (April 6th/8th/11th)	n/a	n/a
Total Hours = 252 (+15 h to meetings with the Senior Engineers + 3h for Final Demo Day) = 270hr total		

List of Remaining Tasks v1.0

A.4.2 Week 2 List of Tasks

Week	Milestones	Task	Time Estimate (hours)
Week 2	**Week 2 Deliverable (March 14th, 2022)**	Test the motor accuracy	5
		Meeting Agenda for meeting w/ Senior Engineer & Weekly Meetings	1
		Input Tower Prototype	6
		Rotating Platform Prototype	7
		Delivery System Prototype	6
		Color Sensor Arm Prototype	5
		Writing Testing Procedures	5
		Weekly Deliverables	15
		Design Iteration Meeting	2.5
		Meetings (w/ Senior Engineer & Status Update Weekly Meetings)	6
Week 3	**Beta Demo Day (March 18th, 2022)**	Integration Testing	14
		Beta demo	3
	Week 3 Deliverable (March 21st, 2022)	Post-Beta Demo Reflections	3
		Meeting Minutes Prep	1
		Debugging period Post-Beta demo	17
		Weekly Meetings	6

		Weekly Deliverables	13
Week 4	**Week 4 Deliverable (March 28th, 2022)**	Final Tests/Testing	12
		Meeting Minutes Preparation	1
		Final Report Writing	12
		Final Project Presentation Preparation (Draft Presentation)	13
		Week Deliverables	13
		Weekly Meetings	6
Week 5	**Final Demo Day (April 1st, 2022)**	Final Report Writing	28
		Final Demo	6
		Final Project Presentation Preparation	22
	Week 5 Deliverable (April 4th, 2022)	Final Project Documents & Submission	n/a
	Final Design Review (April 6th/8th/11th)	n/a	n/a
	Total Hours = 270		

List of Remaining Tasks v2.0

A.4.3 Week 3 List of Tasks

Week	Milestones	Task	Time Estimate (hours)
Week 4	**Week 4 Deliverable (March 28th, 2022)**	Final Tests/Testing (March 25th, 2022)	11
		Meeting Minutes Preparation	1
		Final Report Writing	10
		Final Project Presentation Preparation (Draft Presentation)	10
		Week Deliverables	13
		Weekly Meetings	6
Week 5	**Final Demo Day (April 1st, 2022)**	Final Report Writing	22
		Final Demo	6
		Final Project Presentation Preparation	20
	Week 5 Deliverable (April 4th, 2022)	Final Project Documents & Submission	n/a
	Final Design Review (April 6th/8th/11th)	n/a	n/a
	Total Hours Planned = 99 hours		

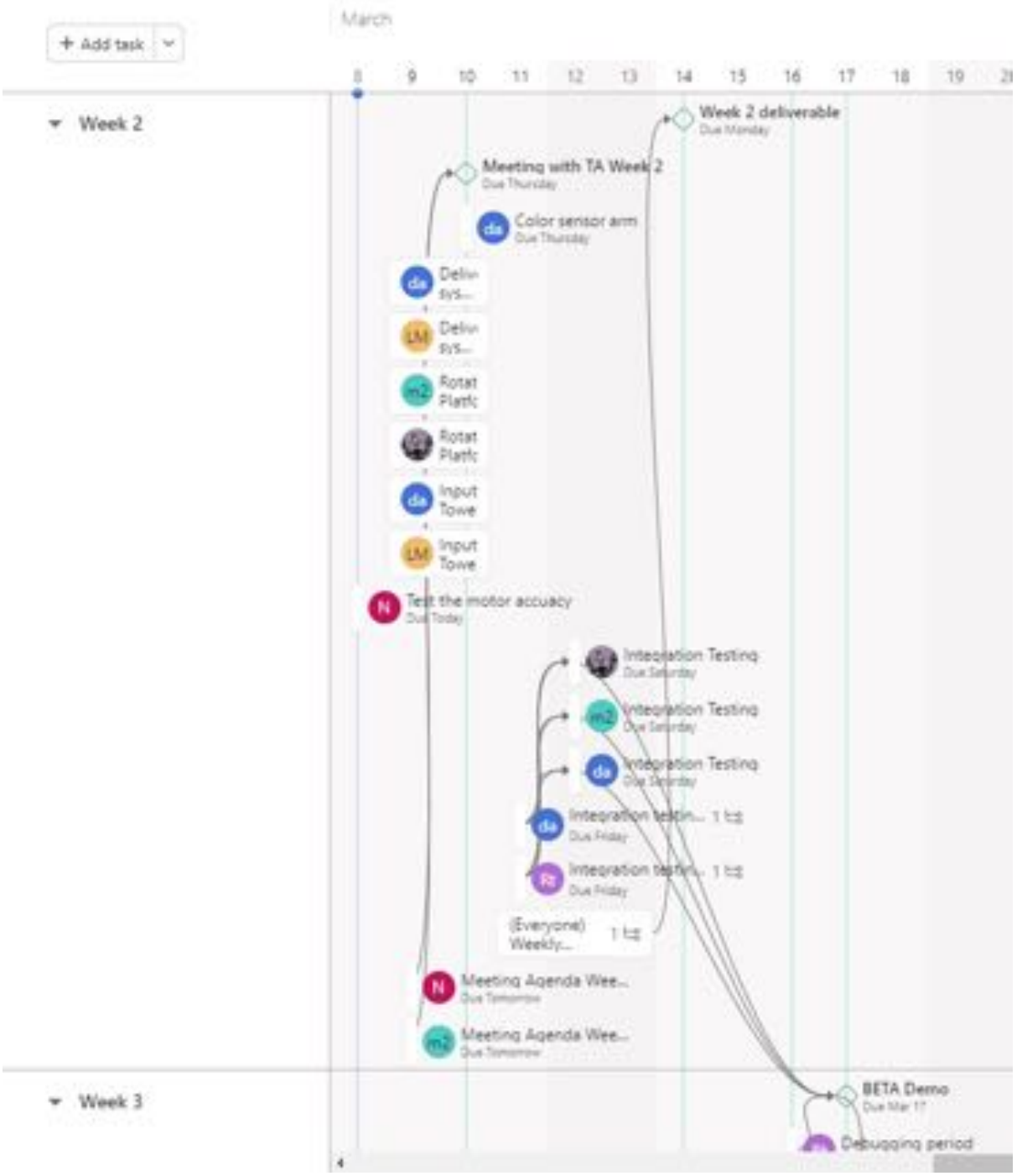
List of Remaining Tasks v3.0

A.4.4 Week 4 List of Tasks

Week	Milestones	Task	Time Estimate (hours)
Week 5	**Final Demo Day (April 1st, 2022)**	Final Report Writing	24.5
		Final Demo	6
		Final Project Presentation Preparation	15
	Week 5 Deliverable (April 4th, 2022)	Color Sensor Position Test	2
		Pre-Final Demo Test	2
		Final Project Documents & Submission	n/a
	Final Design Review (April 6th/8th/11th)	n/a	n/a
	Total Hours Planned = 45.5 hours		

List of Remaining Tasks v4.0

A.5.1 Week 1 Gantt Chart



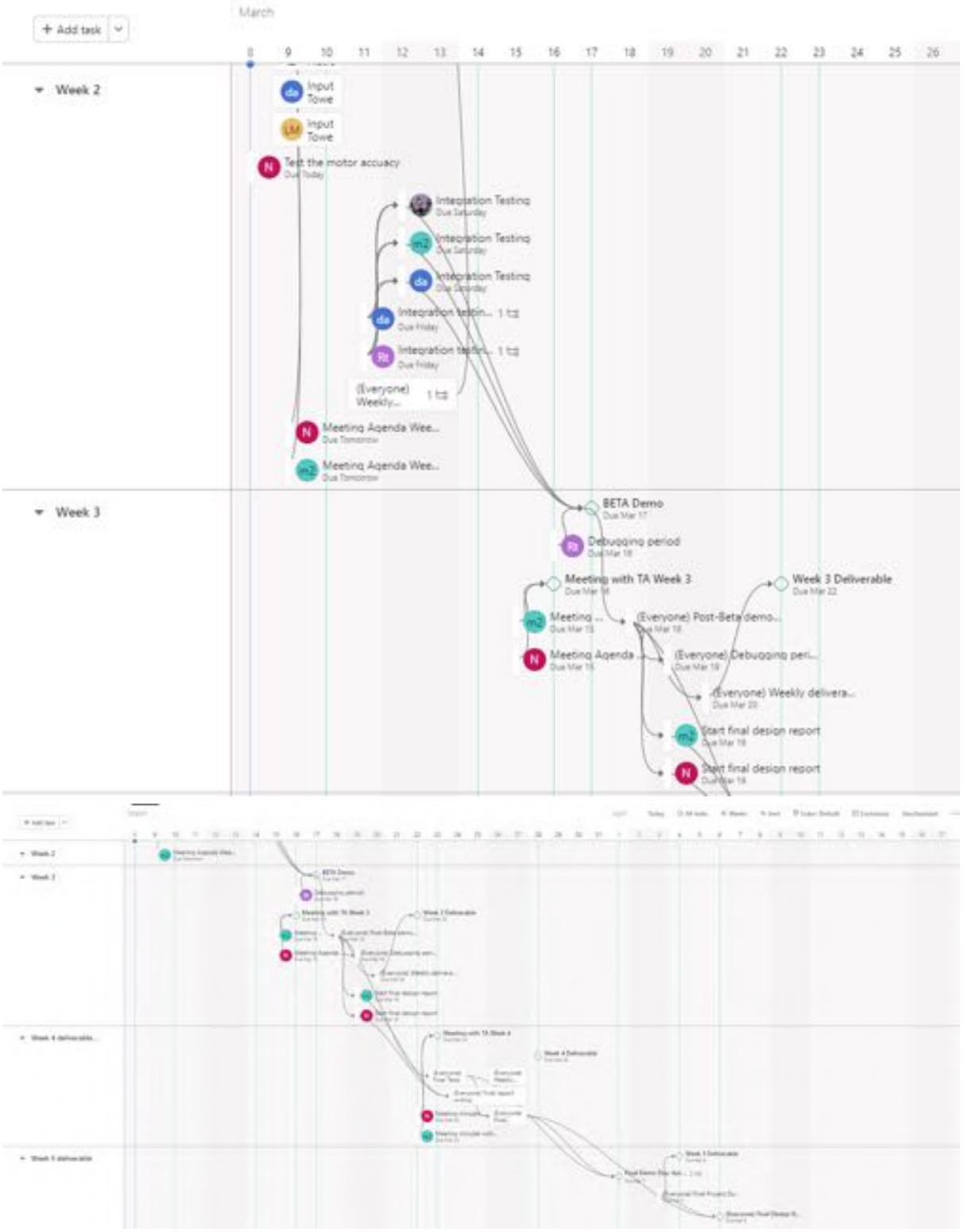

March
+ Add task
Week 2
Input Towe
Input Towe
Test the motor accuacy
Integration Testing
Integration Testing
Integration Testing
Integration testin...
Integration testin...
(Everyone) Weekly...
Meeting Agenda Wee...
Meeting Agenda Wee...
Week 3
BETA Demo
Debugging period
Meeting with TA Week 3
Week 3 Deliverable
Meeting ...
(Everyone) Post-Beta demo...
Meeting Agenda
(Everyone) Debugging peri...
(Everyone) Weekly delivera...
Start final design report
Start final design report

Figure 44.

A.5.2 Week 4 & 5 Gantt Chart

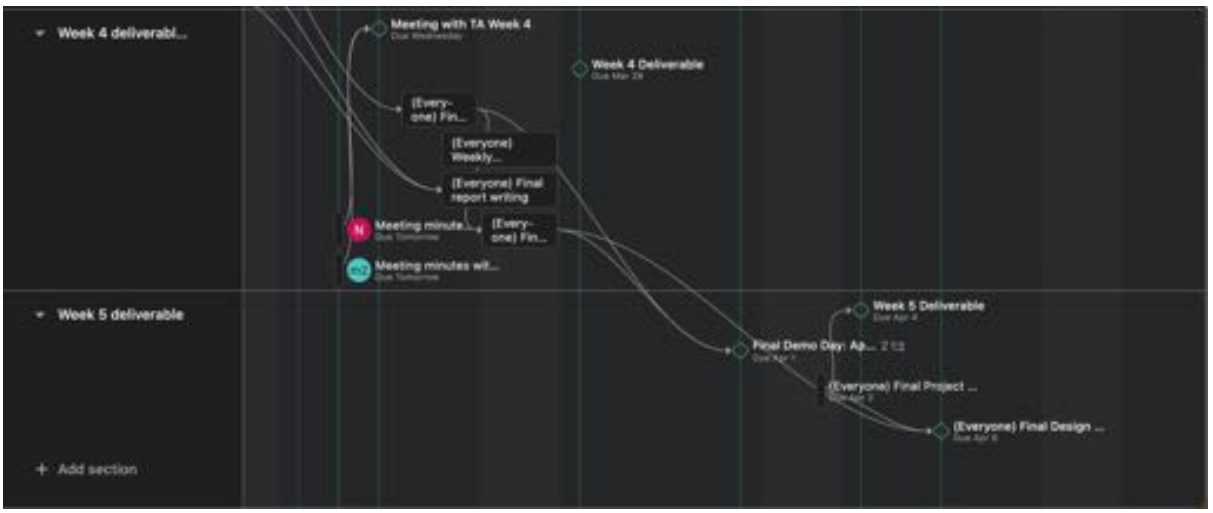


Figure 45.

A.5.3 Week 5 Gantt Chart

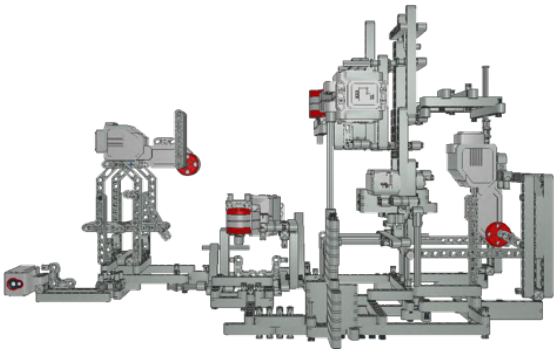
WBS NUMBER	TASK TITLE	TASK OWNER	START DATE	DUE DATE	PHASE TWO							PHASE THREE							PHASE FOUR							PHASE FOUR						
					WEEK 2							WEEK 3							WEEK 4							WEEK 5						
					7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	1	2	3	4

Figure 46.

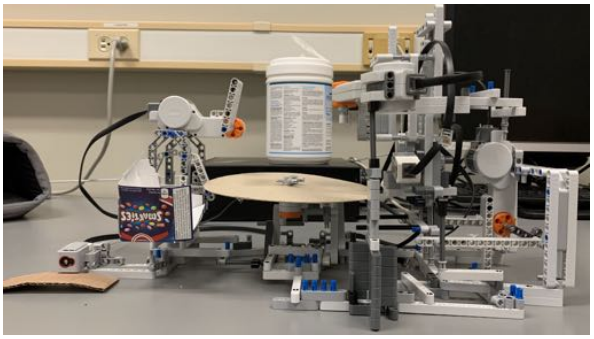
APPENDIX B

CONTENTS

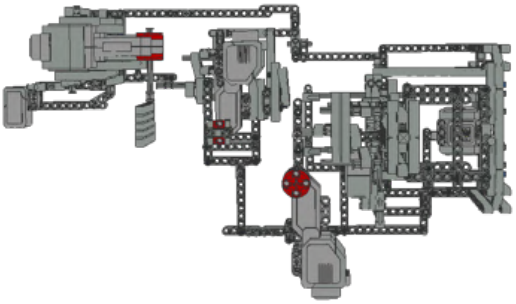
B.1 Digital Twin vs Physical Implementation



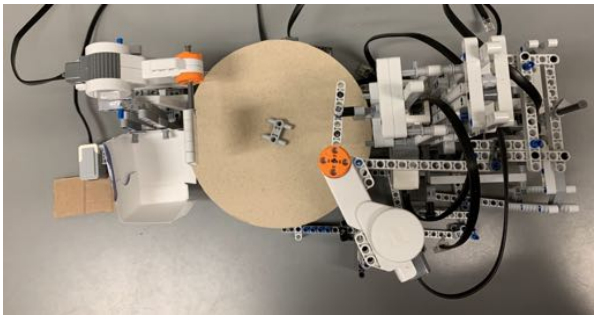
(a) Digital Twin Left Side View



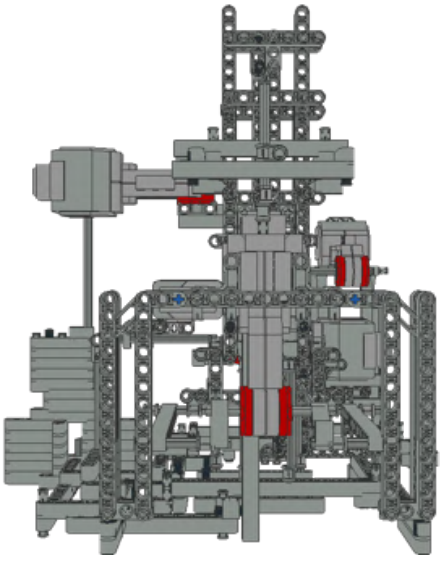
(b) Physical Implementation Left Side View



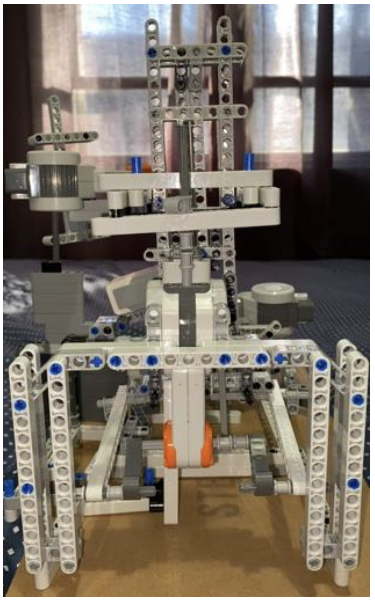
(a) Digital Twin Top View



(b) Physical Implementation Top View

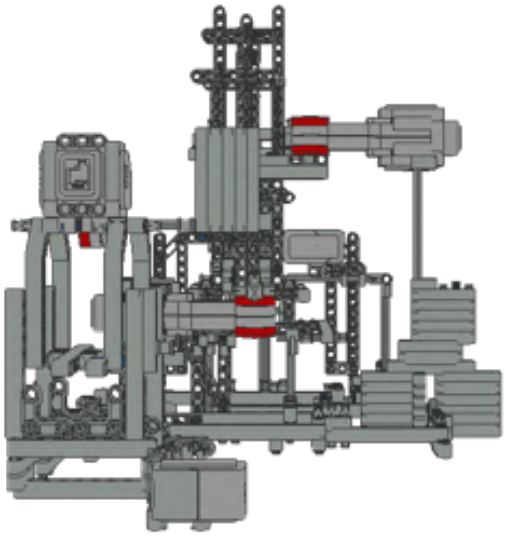


(a) Digital Twin Behind Input Tower

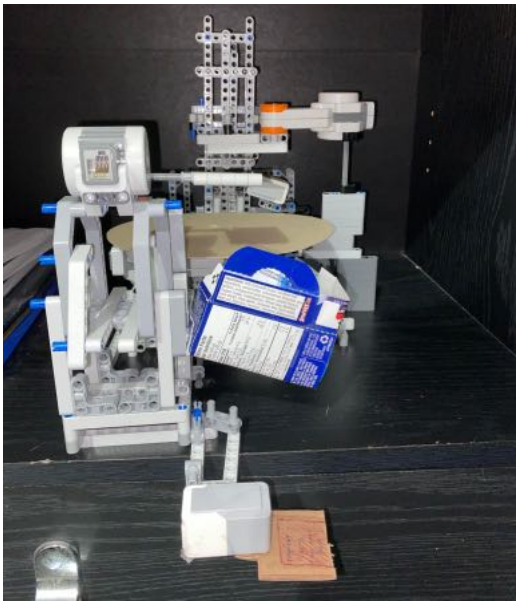


(b) Physical Implementation Behind Input Tower

Figure ___ Behind Delivery System POV Digital Twin vs Physical Implementation



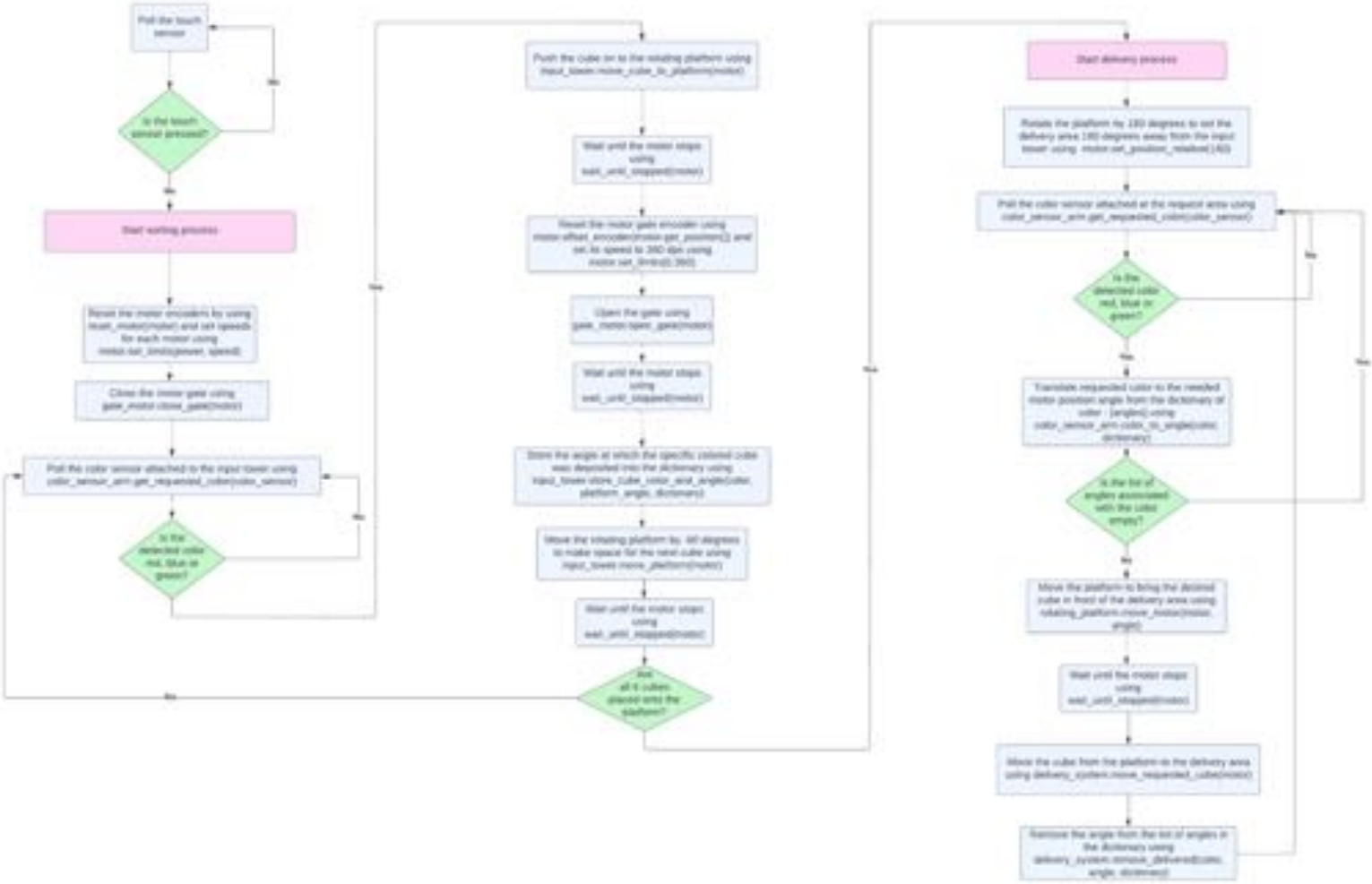
(a) Digital Twin Behind Delivery System



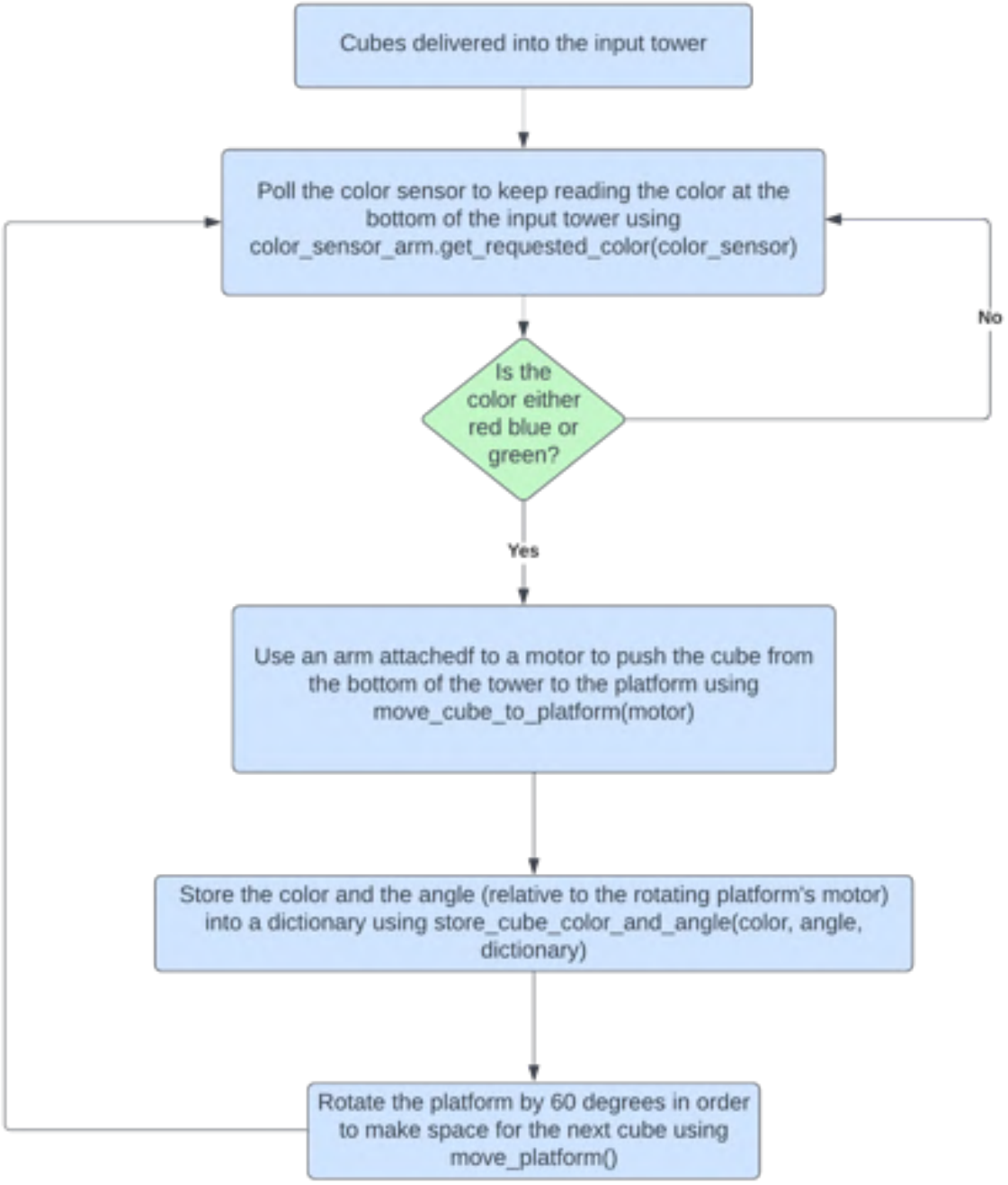
(b) Physical implementation Behind Delivery System

B.2 Final System Flow Flowchart

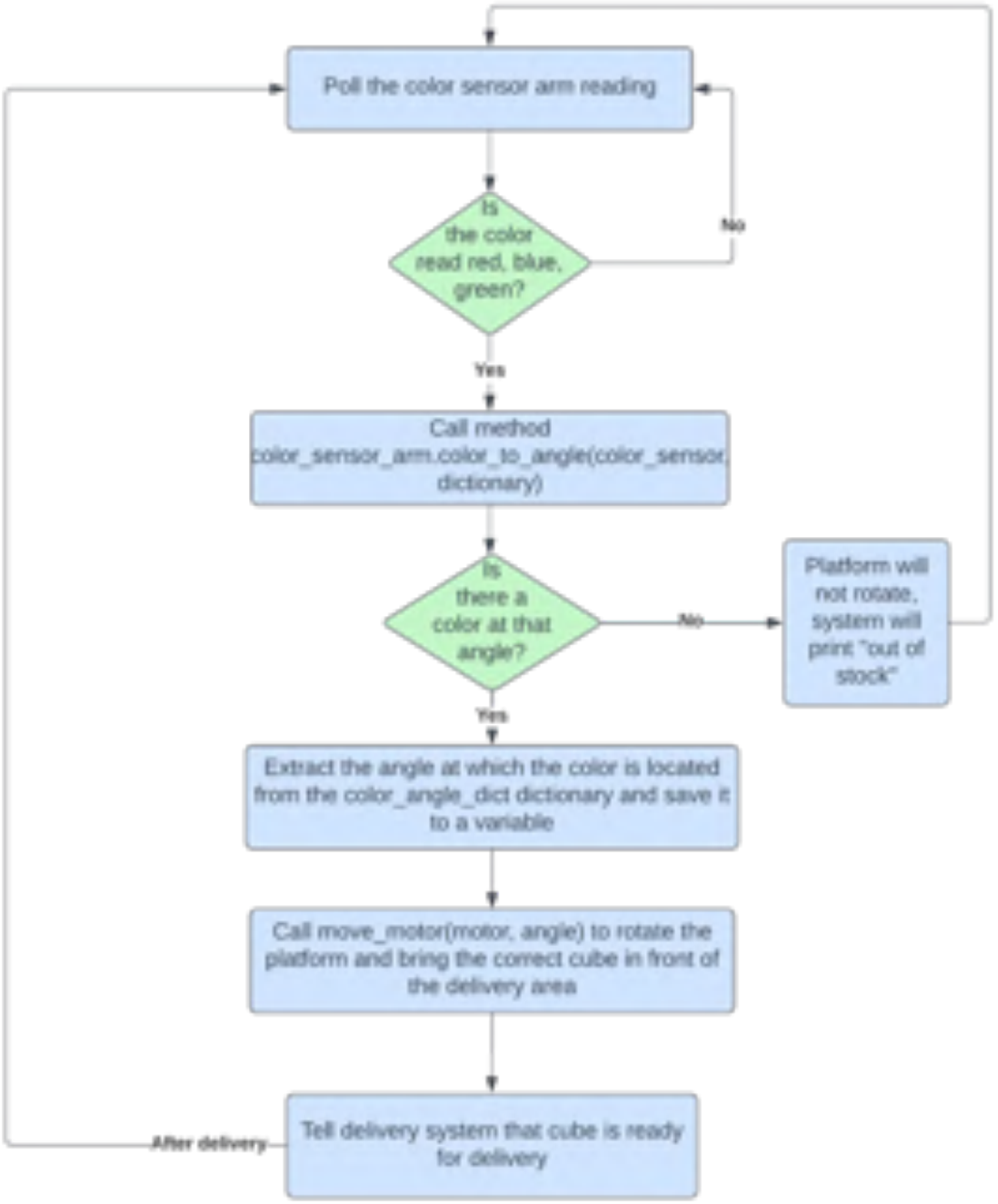
Figure ____ Final System Software Flow Flowchart



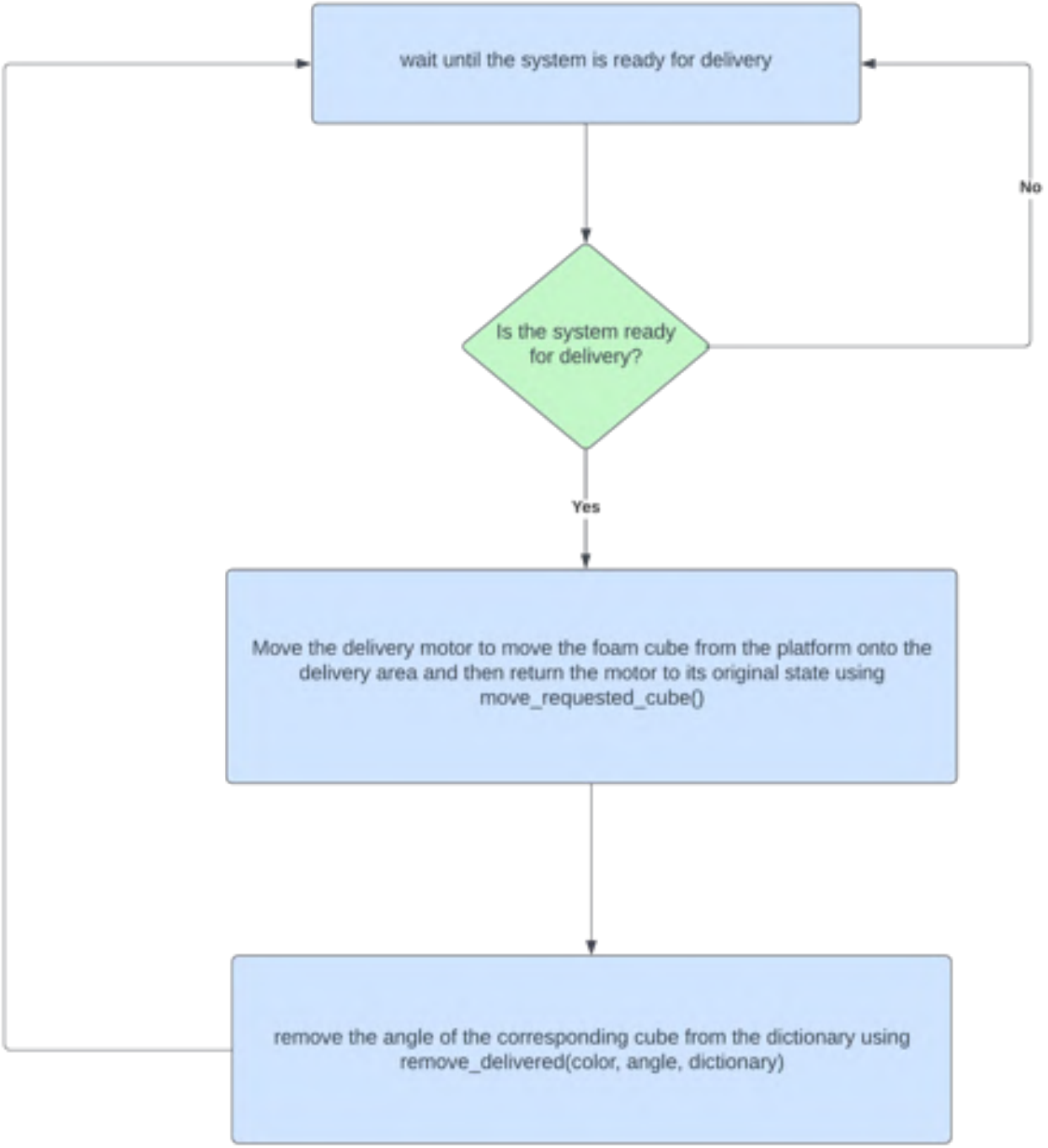
B.2.1 Input Tower Flowchart



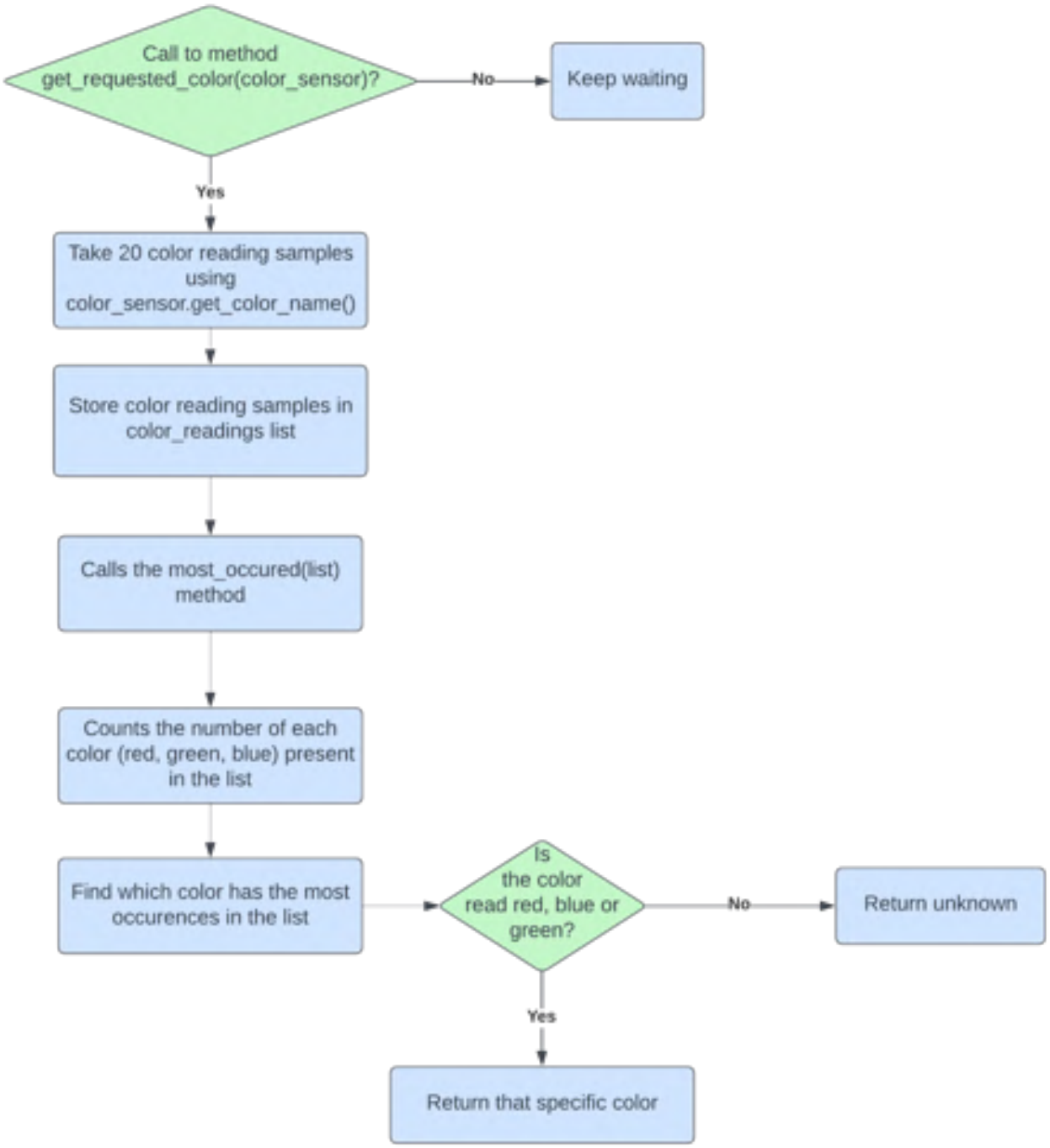
B.2.2 Rotating Platform Flowchart



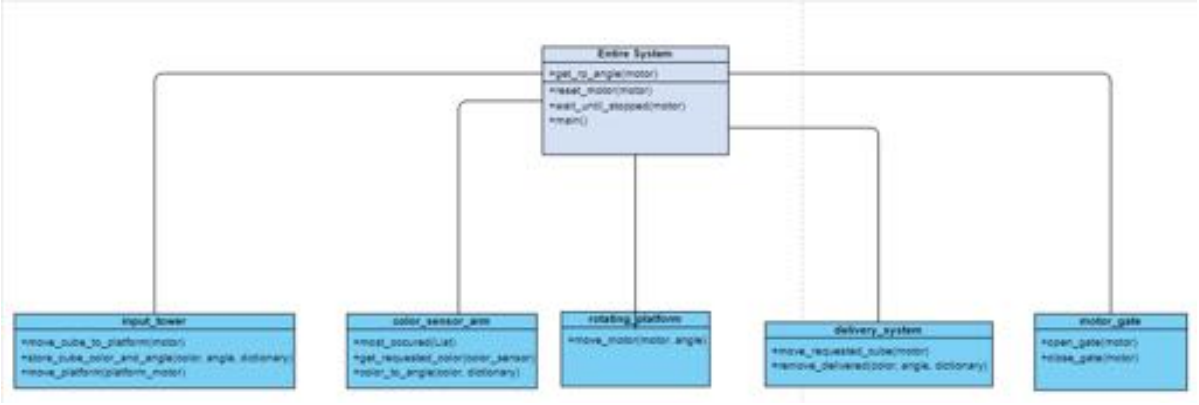
B.2.3 Delivery System Flowchart



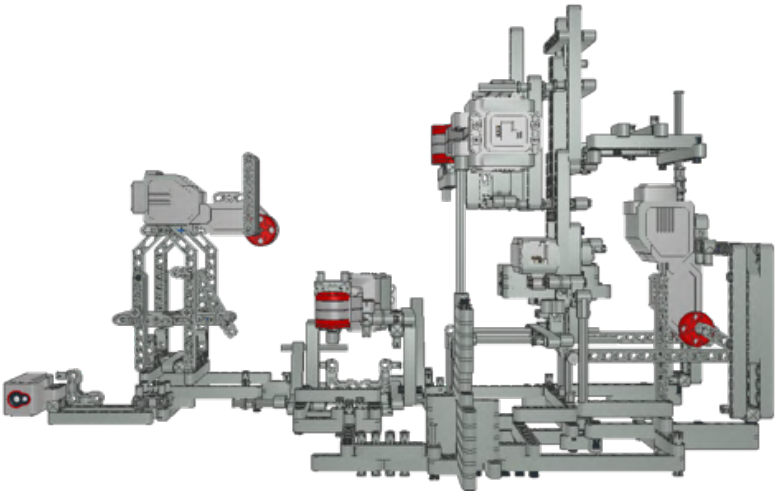
B.2.4 Color Sensor Arm Flowchart



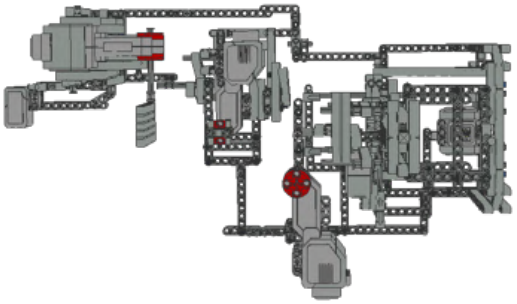
B.3 Final System Software Structure



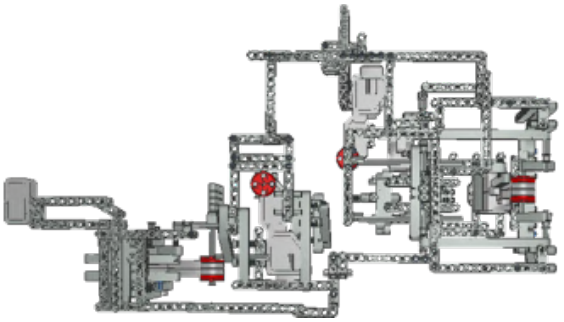
B.4 Final System



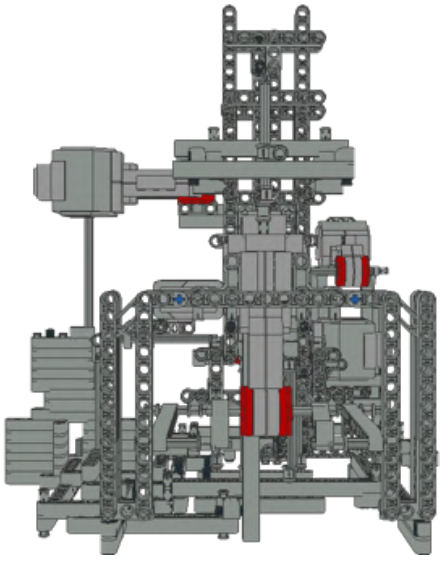
(a) Left Side View



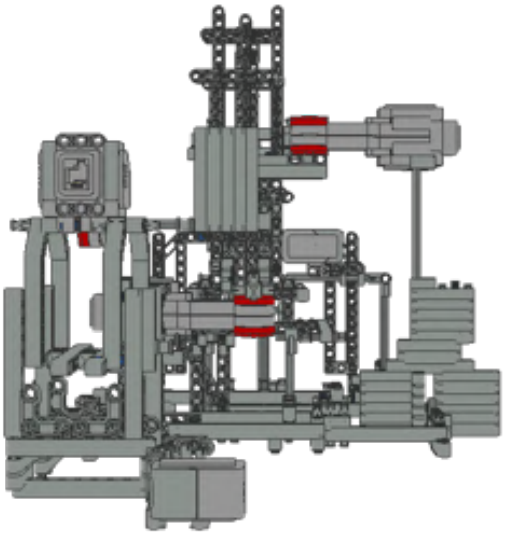
(b) Top View



(c) Bottom View

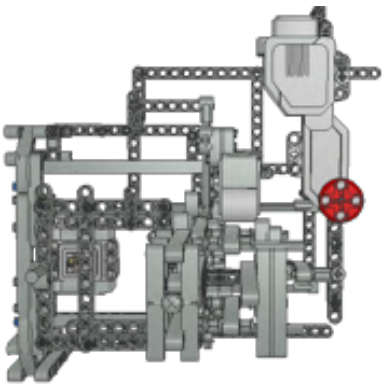


(d) Behind Input Tower POV

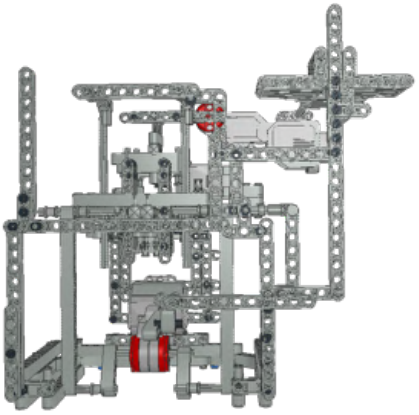


(e) Behind Delivery System POV

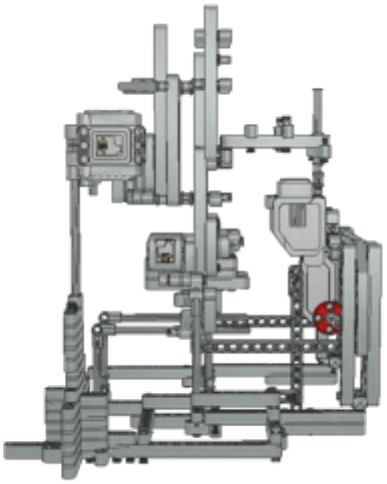
B.4.1 Final Subsystem 1: Input Tower Hardware



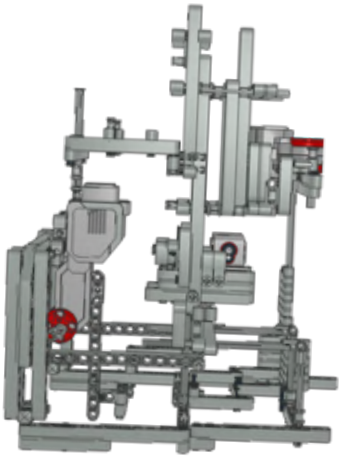
(a) Top view



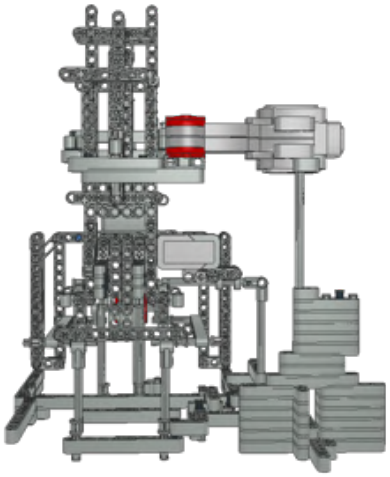
(b) Bottom view



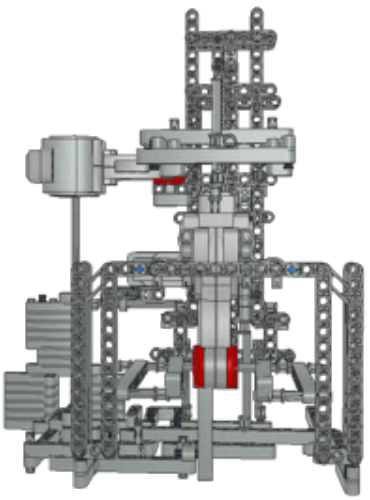
(c) Left side view



(d) Right side view

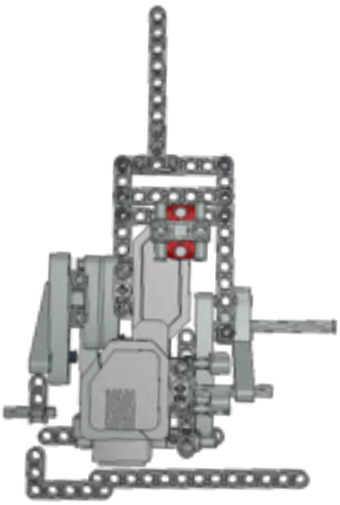


(e) Front view

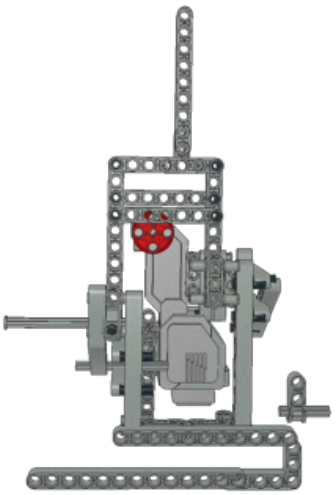


(f) Back view

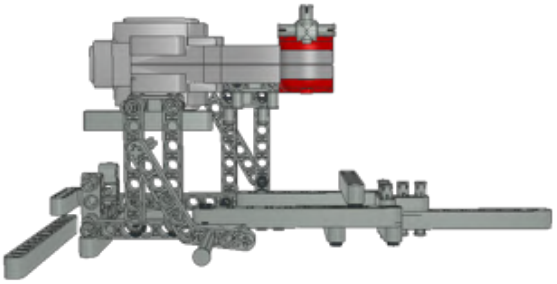
B.4.2 Final Subsystem 2: Rotating Platform Hardware



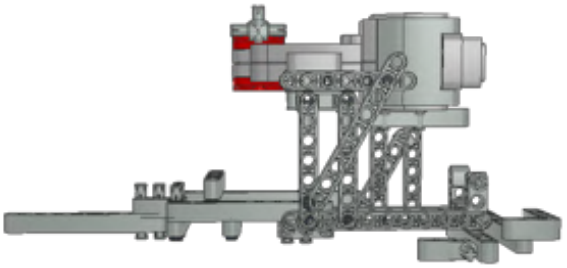
(a) Top View



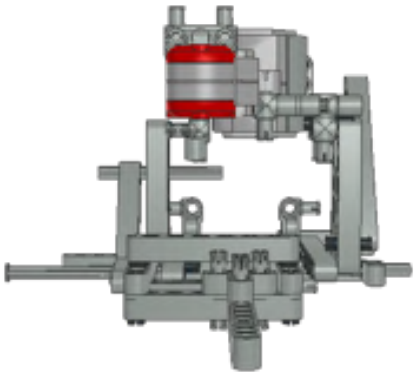
(b) Bottom view



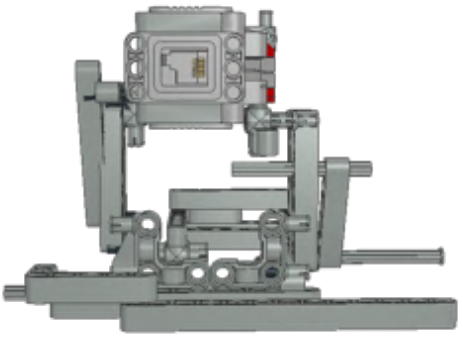
(c) Right side view



(d) Left side view

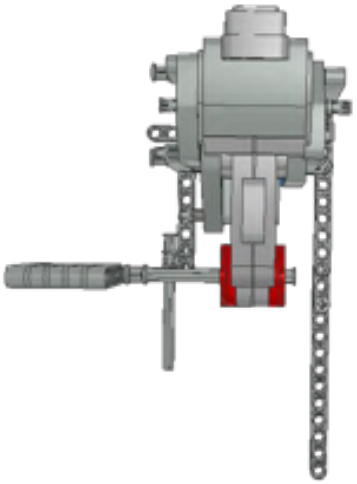


(e) Front View

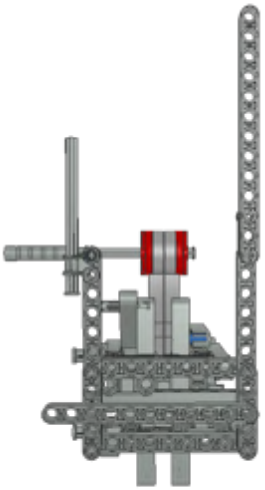


(f) Back View

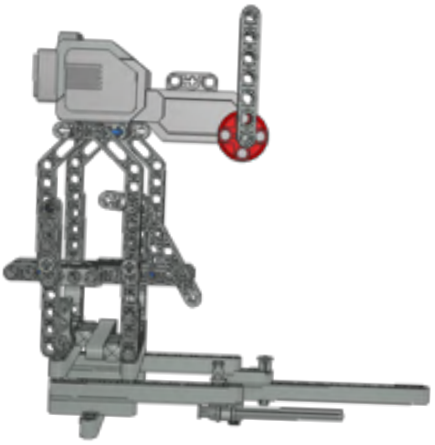
B.4.3 Final Subsystem 3: Delivery System Hardware



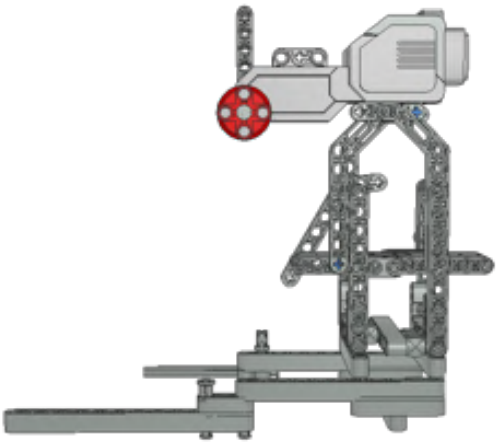
(a) Top View



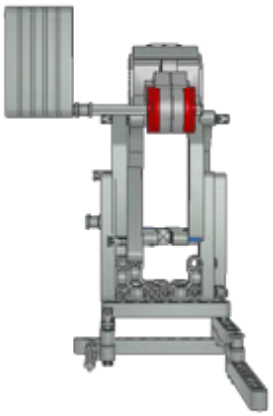
(b) Bottom View



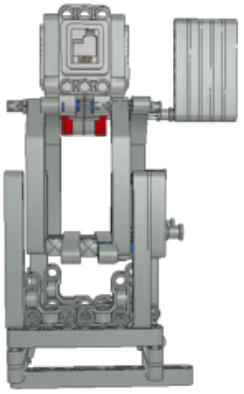
(c) Right side view



(d) Left side view



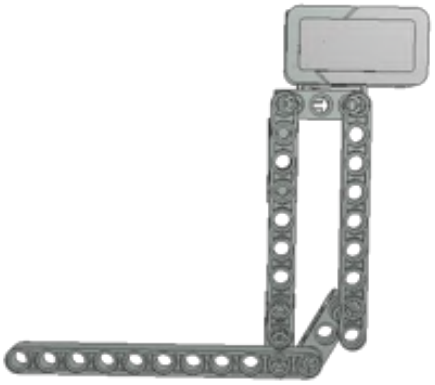
(e) Front view



(f) Back view

B.4.4 Final Subsystem 4: Color Sensor Arm Hardware

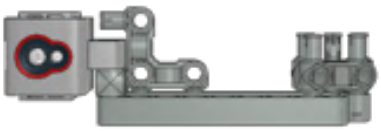
(a) Top view



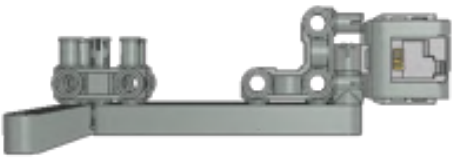
(b) Bottom view

(c) Right side view

(d) Left side view



(e) Front view



(f) Back view

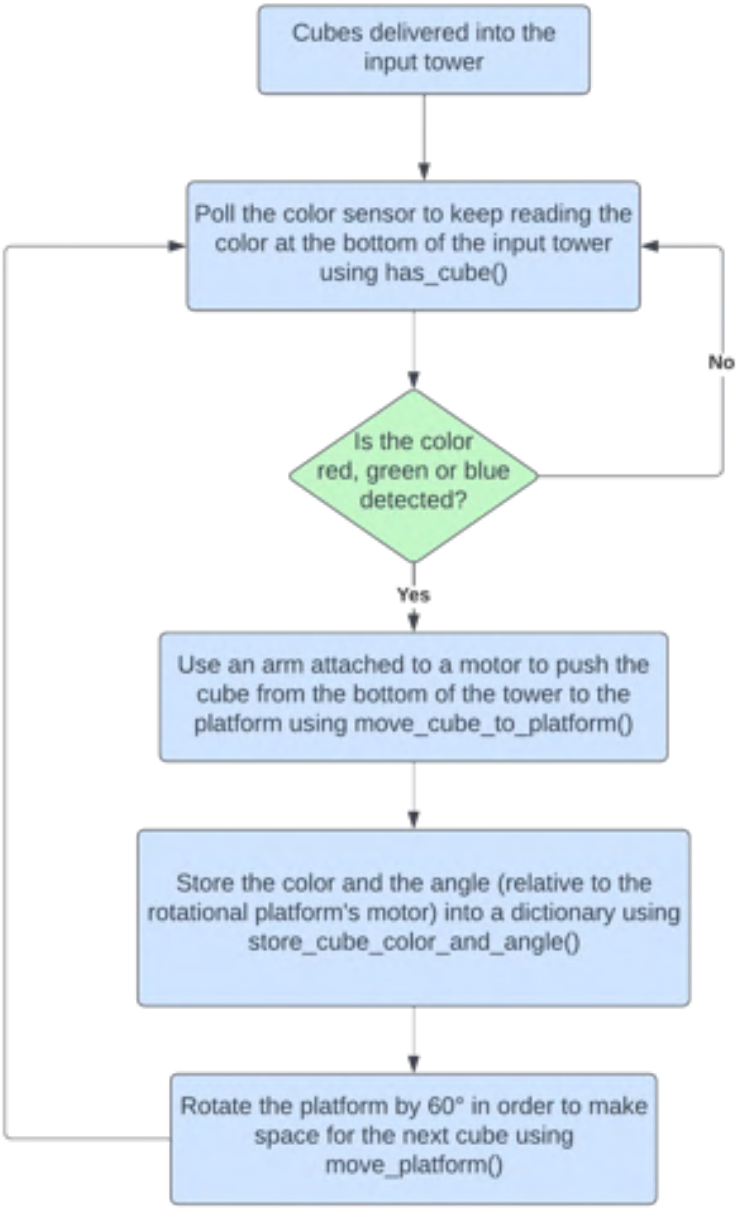
APPENDIX C

CONTENTS

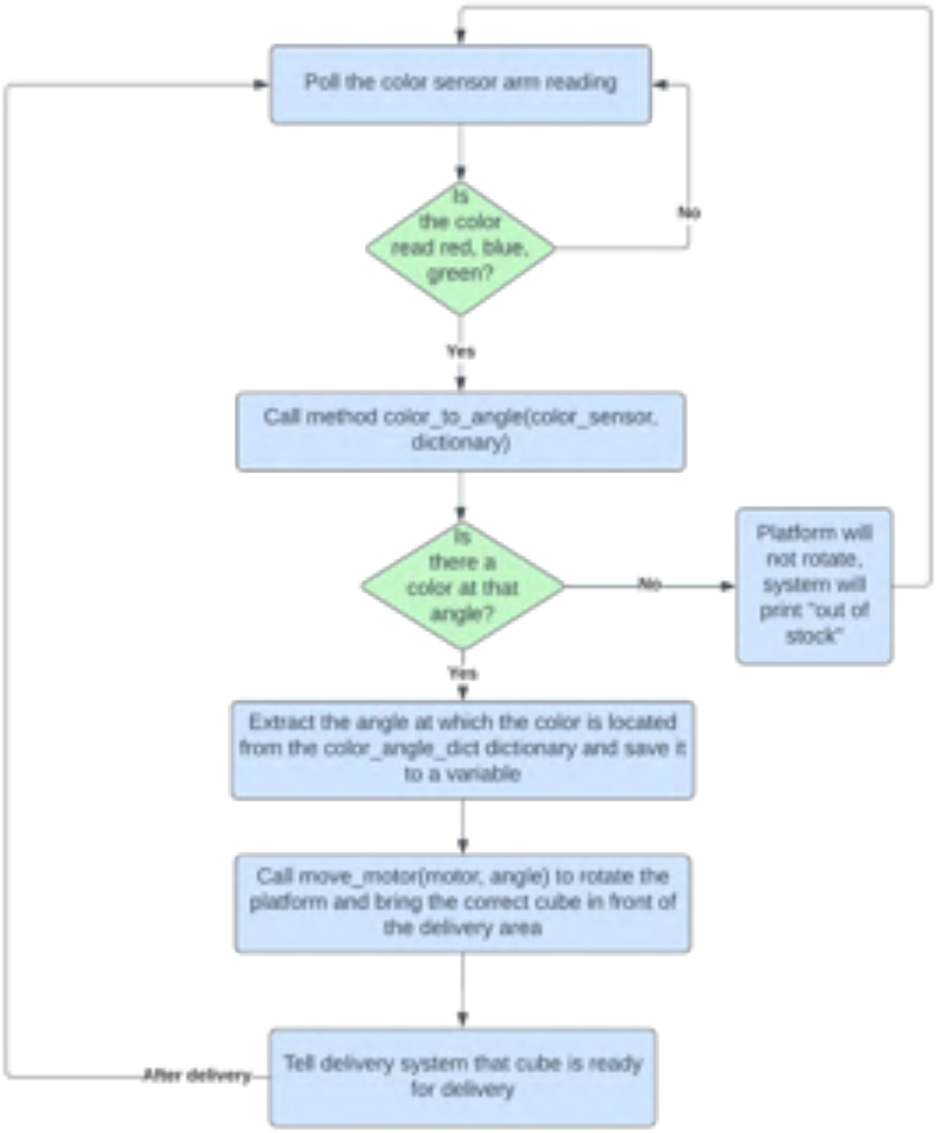
C.1 System Version Flowcharts

C.1.1 System Version 2.0

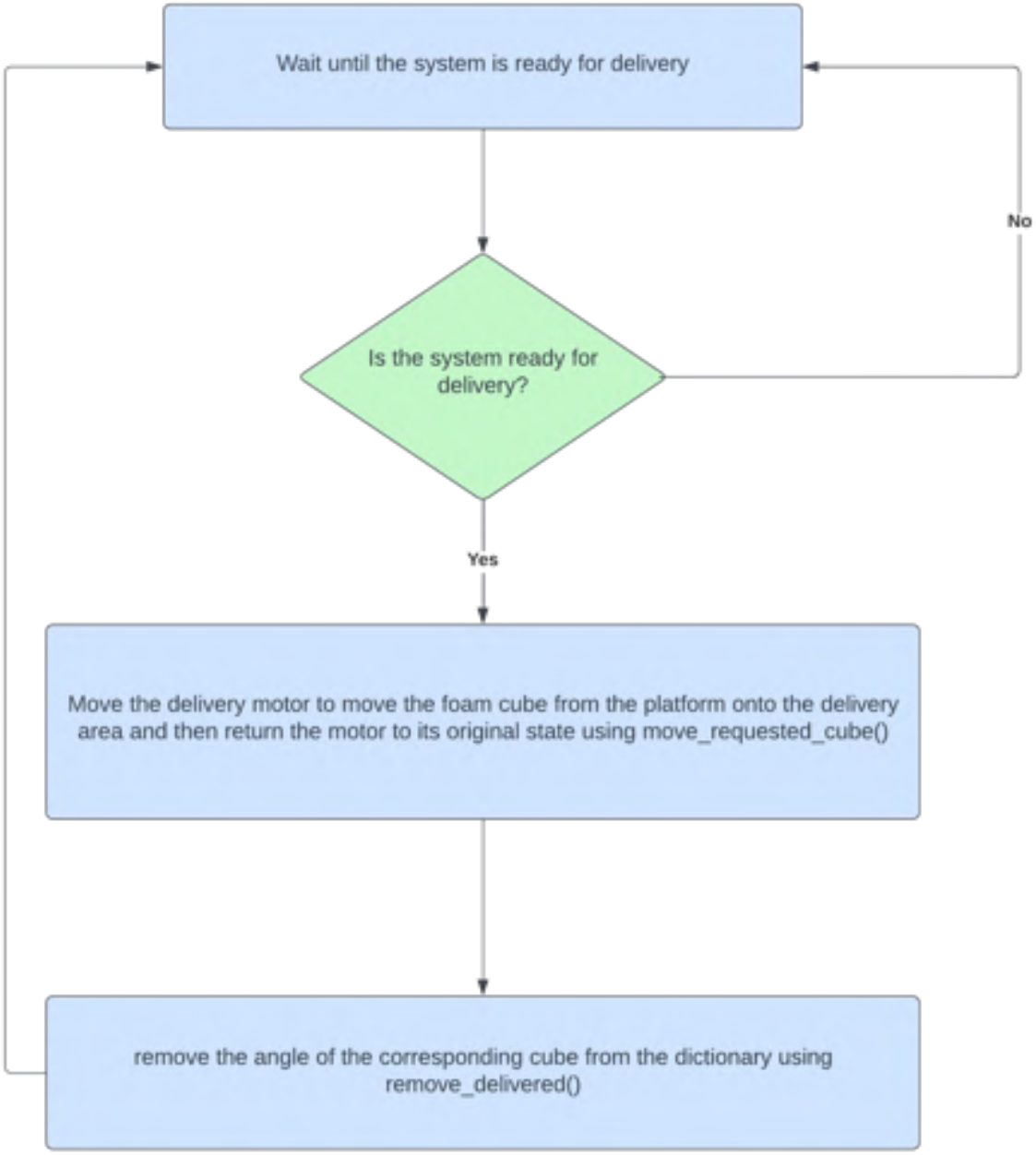
Input Tower v2.0

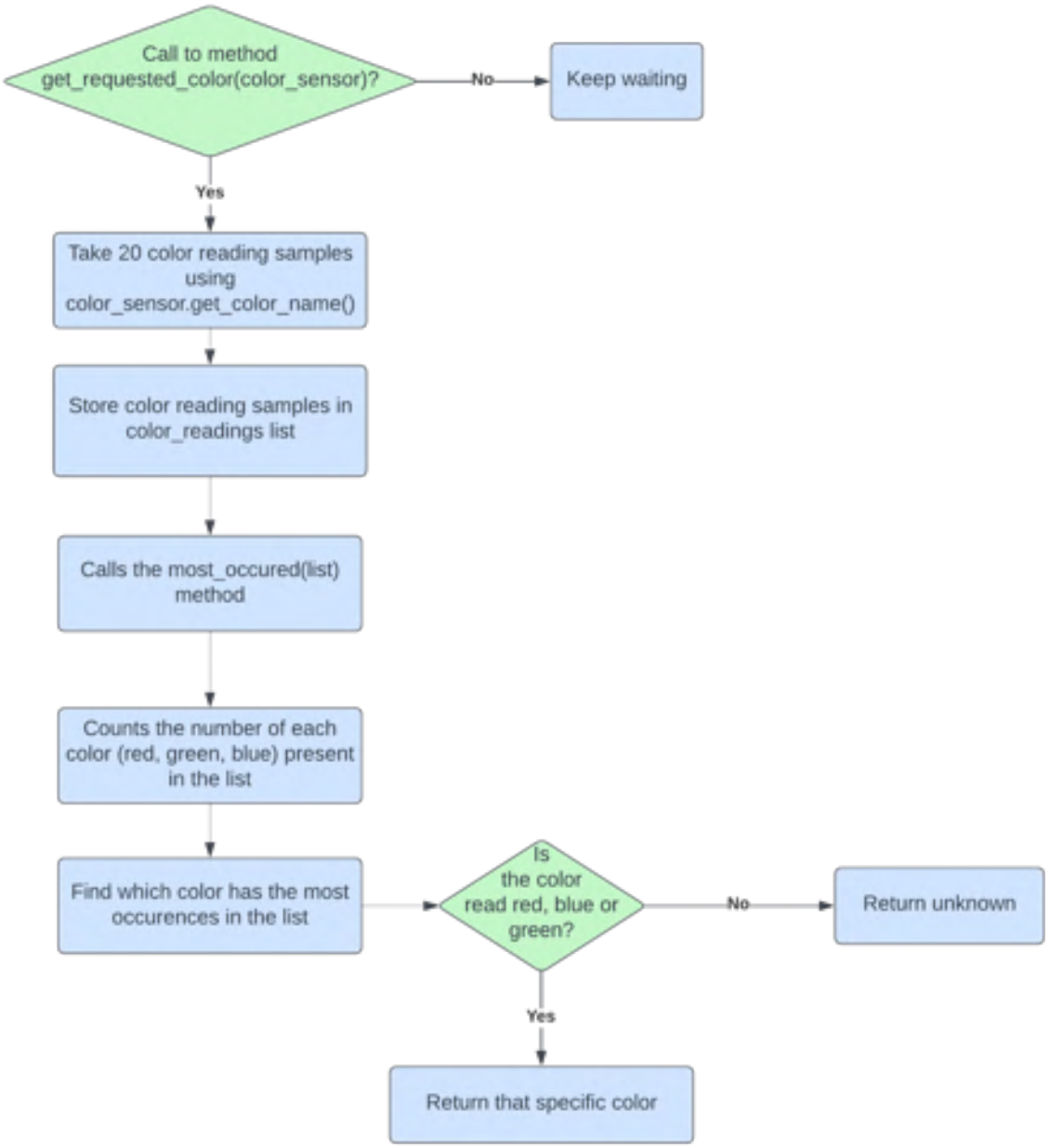


Rotating Platform v2.0



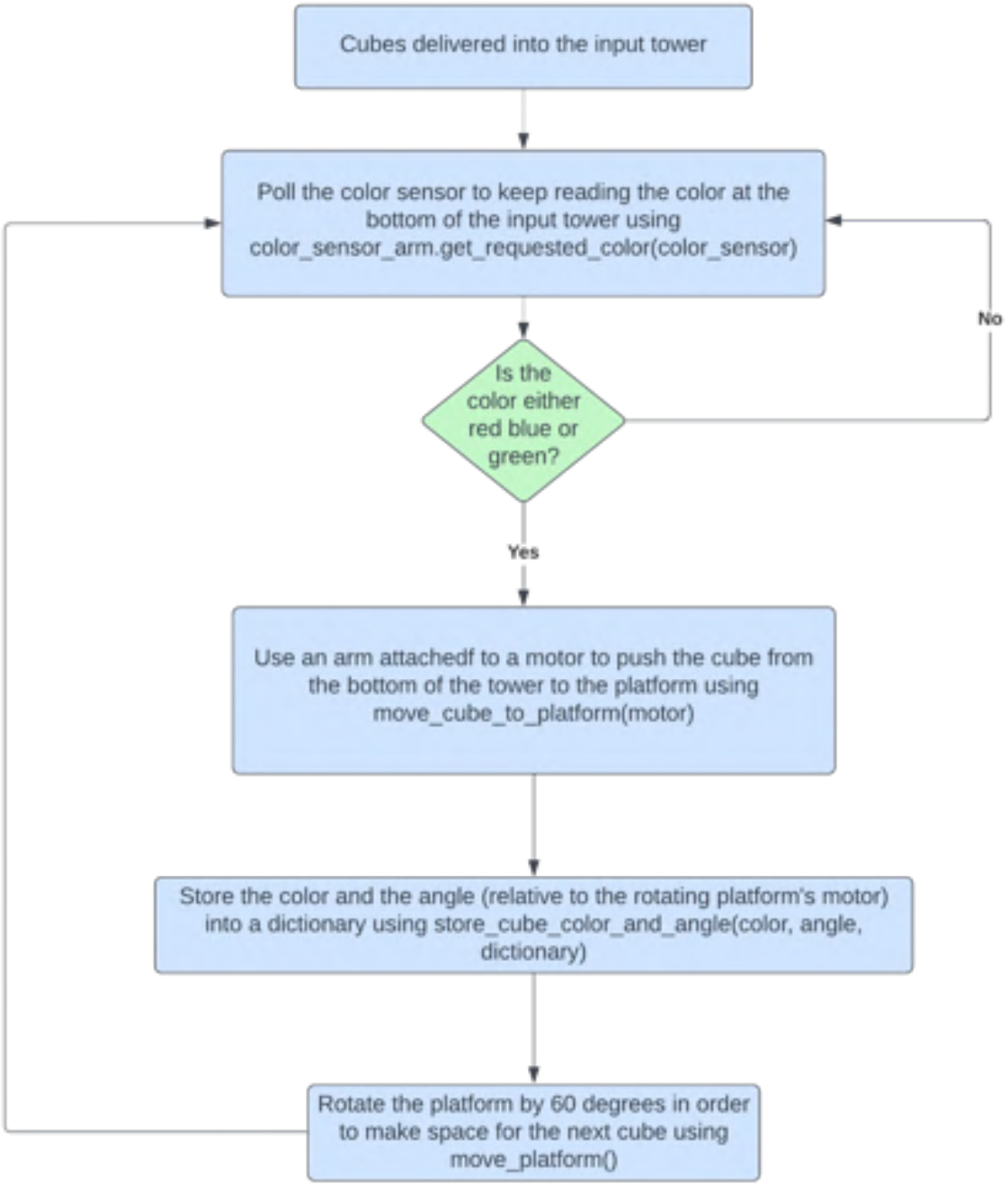
Delivery System v2.0



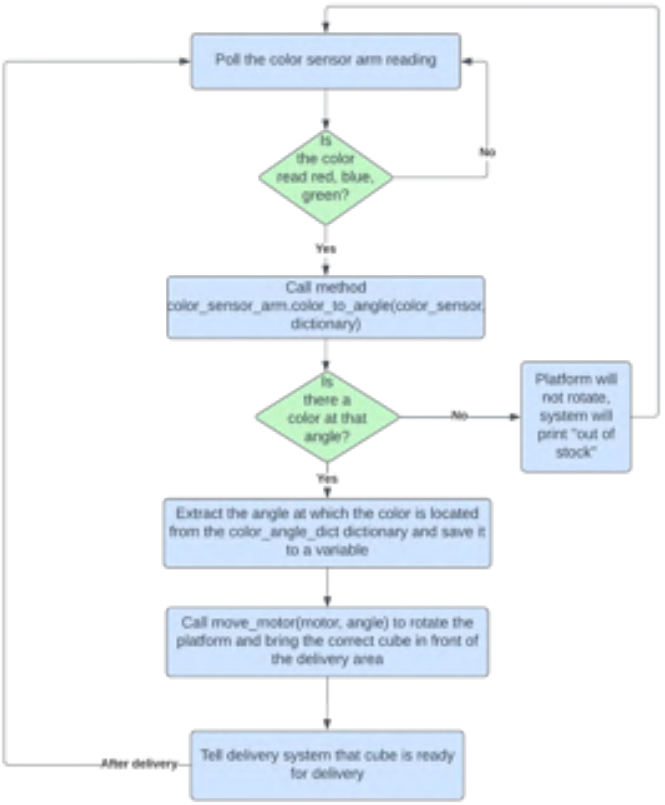

Call to method get_requested_color(color_sensor)?
No
Keep waiting
Yes
Take 20 color reading samples using color_sensor.get_color_name()
Store color reading samples in color_readings list
Calls the most_occured(list) method
Counts the number of each color (red, green, blue) present in the list
Find which color has the most occurences in the list
Is the color read red, blue or green?
No
Return unknown
Yes
Return that specific color

C.1.2 System Version 3.0

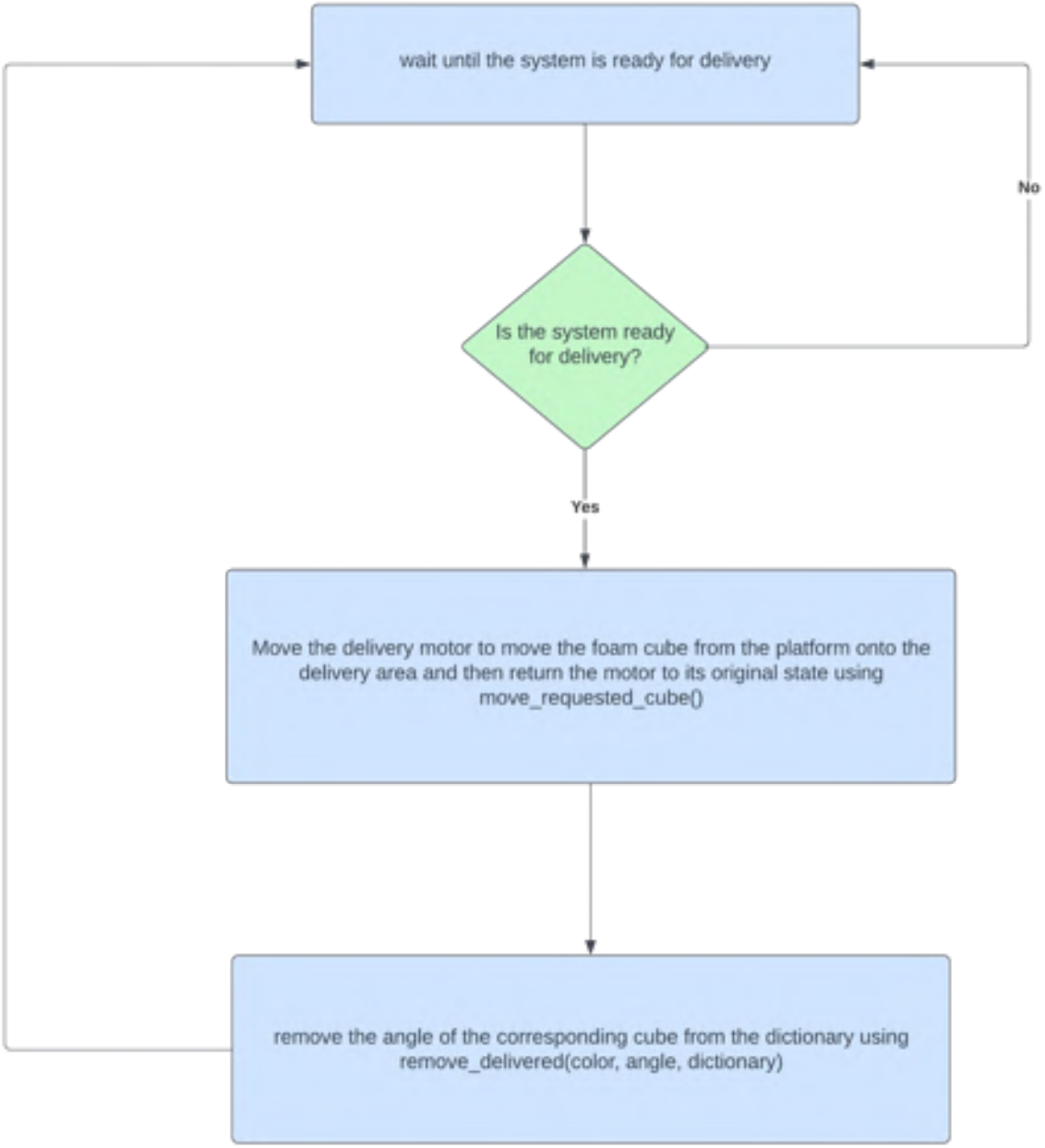
Input Tower v3.0

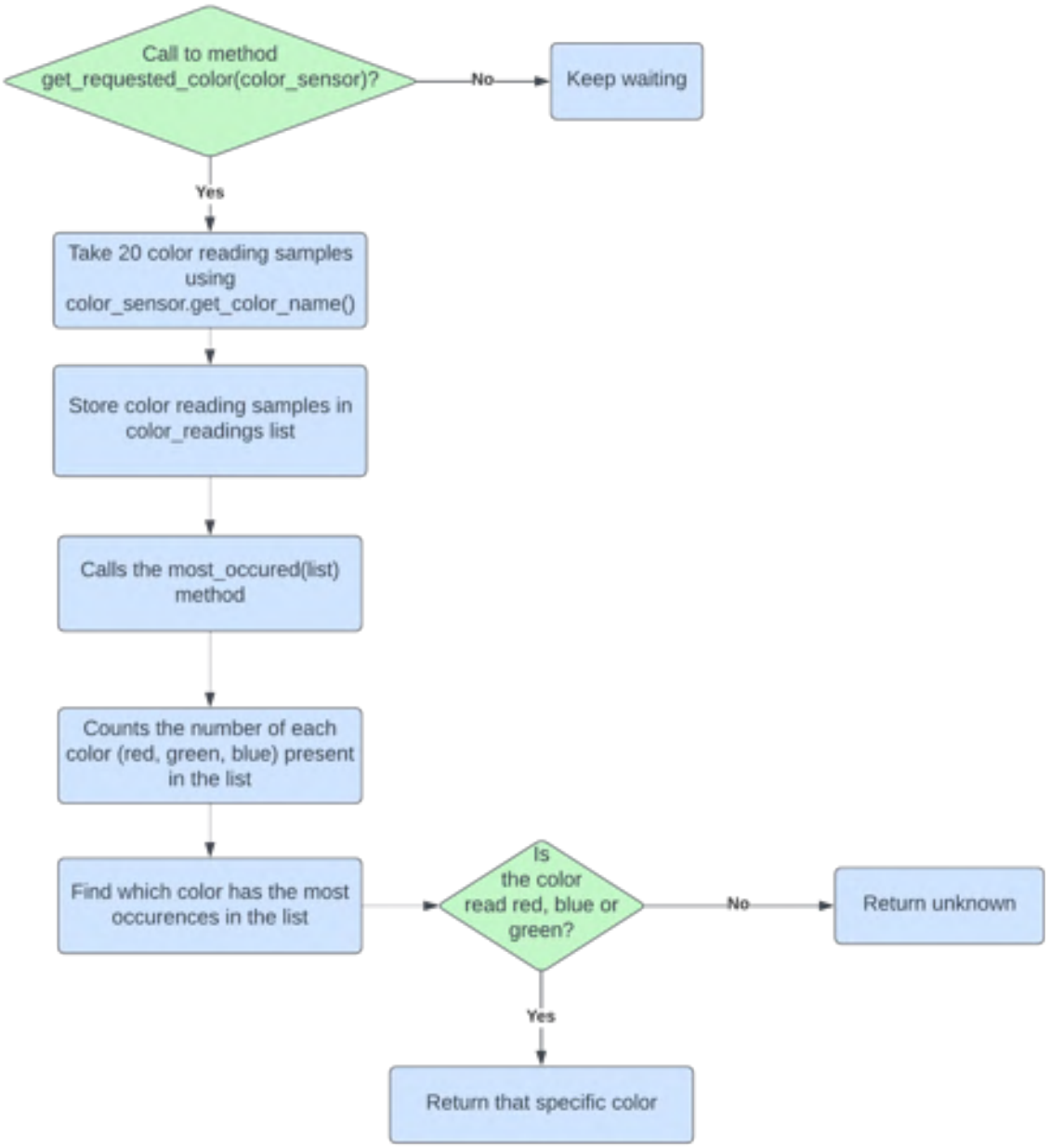


Rotating Platform v3.0

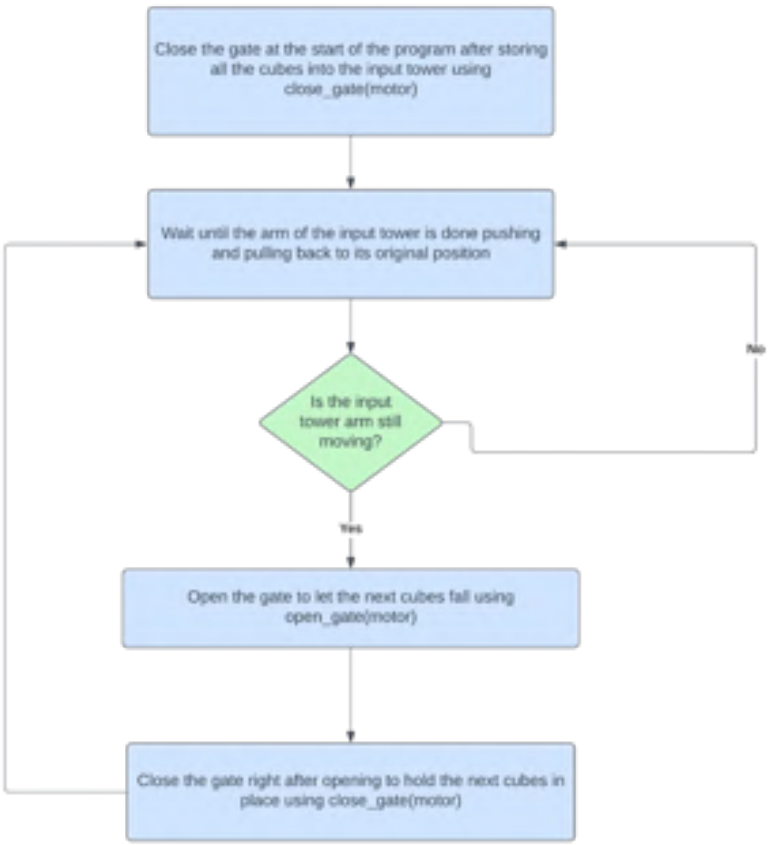


Delivery System v3.0




Call to method get_requested_color(color_sensor)?
No
Keep waiting
Yes
Take 20 color reading samples using color_sensor.get_color_name()
Store color reading samples in color_readings list
Calls the most_occured(list) method
Counts the number of each color (red, green, blue) present in the list
Find which color has the most occurences in the list
Is the color read red, blue or green?
No
Return unknown
Yes
Return that specific color

Motor Gate v1.0



APPENDIX D

CONTENTS

D.0 Test Schedule

D.0.1 Week 2 Test Schedule

Test	Date	Tester
Rotating Platform Test	March 12th, 2022	Muqtadir Ahmed
Color Sensor Arm Test	March 14th, 2022	Ryan Tabbara

D.0.2 Week 3 Test Schedule

Test	Date	Tester
Cube/Input Tower Hardware Test	March 17th, 2022	Muqtadir Ahmed
Motor Gate Test	March 20th, 2022	Muqtadir Ahmed
Delivery System Test	March 20th, 2022	Muqtadir Ahmed

D.0.3 Week 4 Test Schedule

Test	Date	Tester(s)
Modified Input Tower Test	March 25th, 2022	Nour, Muqtadir
Input Tower Platform Test	March 25th, 2022	Nour, Muqtadir
Input Tower Piston Test	March 25th, 2022	Nour, Muqtadir
Input Tower & Rotating Platform Integration Test	March 26th, 2022	Nour, Muqtadir
Input Tower & Updated Rotating Platform Integration Test	March 26th, 2022	Muqtadir, Nour
System “Sorting” Activation Test	March 26th, 2022	Muqtadir, Miiyu
System Integration Test	March 26th, 2022	Muqtadir, Miiyu
“Sorting” Process Timing Test	March 26th, 2022	Muqtadir, Miiyu
Delivery Process Timing Test	March 26th, 2022	Muqtadir, Miiyu

D.0.4 Week 5 Test Schedule

Test	Date	Tester
Color Sensor Position Test	March 30th, 2022	Miiyu, Muqtadir
Piston Arm Surface Modification test	March 30th, 2022	Miiyu, Muqtadir
Piston Arm Surface Clear Tape Addition Test	March 31st, 2022	Miiyu, Muqtadir
Pre-final Demo Test	March 31st, 2022	Miiyu, Muqtadir
Final Pre-final Demo Test	March 31st, 2022	Miiyu, Muqtadir

D.1 Week Two: Testing Procedures

D.1.1 Motor Accuracy Testing Procedure

Group: Project Group 01

Date: March 14, 2022

Tester(s): Nour Ktaily

Author: Nour Ktaily

Goal: Understand the accuracy of the motors.

1.0 Purpose of Test

The purpose of this test is to understand how accurate the motors are.

2.0 Test Objectives

- Figure out how to control the motors.
- Figure out how to deal with any bugs and causes of inaccuracy.

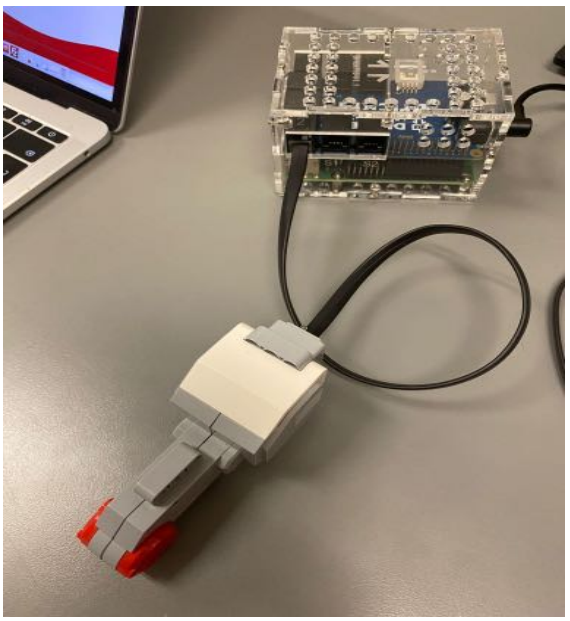
3.0 Test Procedure

- Use the _sample_motors.py file provided in the code handout to write a motor.py script which contains helper methods from _sample_motors.py with a main method to run the code .
- Attach the motor to port MA.
- Run the Power-based Control script first.
- Record the different outputs for different power levels.
- Compare the angle at which the motor spun to the real angle.
- Run the Position-based Control script.
- Record the different results for the different positions.
- Compare the angle at which the motor spun to the real angle.

4.0 Expected Results

It is expected that there are errors with the motor accuracy but only slight errors, such as +/- 5 degrees of the angle we want the motor to rotate at.

5.0 Format of Output Required



6.0 Test Report

It was seen through the tests conducted physically through observation and through trials that we ran at various speeds that as the speed of the motor increases, the final position and the angle recorded by the motor is subject to more chance of error of stopping further from the desired angle. This is mostly due to the inertia of the motor.

7.0 Conclusions

In conclusion, and after the tests conducted, we can say that the motor is accurate to a certain level, as long as the speed is not too high so that it would not drift away from the angle we want it to stop at. Therefore, we could say that the motor passed the tests and could be used in the design.

D.1.2 Rotating Platform Test Procedure
Group: (Design Project Team 01)

Date: March 11th, 2022

Tester(s): Muqtadir Ahmed

Author: Muqtadir Ahmed

Goal: Bringing the desired color in front on the delivery zone by scanning the desired color

1.0 Purpose of Test

The purpose of this test is to see if a cardboard platform of a circular shape can hold 6 foam cubes at equal distance from each other and also to understand how the motor can bring the desired color to the destination point within 5 seconds.

2.0 Test Objectives

- Figure out the material and the radius of the circular platform
- Figure out how to poll until the correct colors are detected
- Figure out how to rotate the platform only for the correct color detections
- Figure out how to bring the desired angle to the destination angle
- Figure out the fastest possible speed without causing errors

3.0 Test Procedure

- Connect the motor to port A and the color sensor to port 1
    - The color sensor may be held freely
    - The motor should have a stable base
- Cut out a cardboard in a circular shape, for which the circle has a radius of approximately 9cm
    - Draw 6 sections that are 60 degrees (measured from the middle point of the cube) apart from each other on the circular platform.
- Set the circular plate on top of the motor, on which foam cubes could be placed
    - Draw 6 different regions 60 degrees apart from each other
- Create a dictionary with color and list of angles, key value pair in the class main_robot_ferris
    - Place the colored foam cubes on the circular plate corresponding to the dictionary (angle 0 should be in front of the delivery zone)
- Run the main method in main_robot_ferris.py
    - Try different speeds until you find the speed that rotates the circular plate the fastest without making them fall
    - The main method sets the current angle of the motor to 0 degrees
    - It then polls until the correct colors (which are red, green or blue) are detected by the color sensor which is held freely using the method) using the method get_requested_color(color_sensor) which is created in the class rotating_platform.py
    - Once the color is detected, the angle will be generated using the color_to_angle(color, dictionary), the motor will then rotate until that specific color is in front of the delivery zone
- Observe if the right color cube is front of the delivery area

4.0 Expected Results

The platform will be able to hold all 6 foam cubes.It is expected that after 120 dps, the color cubes will fall unless there is something holding the cubes from falling. The error from the motors part should be minimal.

5.0 Format of Output Required

Notes taken during the test, recording observations throughout testing.

6.0 Test Report

From the output of the system, it can be seen that the platform can rotate at a maximum of 210 dps, beyond that the color cubes tend to displace or even fall off.

7.0 Conclusions

After the test, it was observed that the platform could hold 6 cubes at 60 degrees apart from each other when taking the center of the cubes as the reference. It has been realized that the speed of the motor can go up to 210 dps without causing the cubes to fall, which is much higher than our expected value. At this speed, the platform would be able to rotate to the destination location within 1 second. Thus, it respects the specification of delivery within 5 seconds. By scanning the chosen color, the system behaves as it should by bringing the color in front of the delivery zone.

D.2 Week Three: Testing Procedures

D.2.1 Input Tower Hardware Test Procedure

Group: Design Project Group 01

Date: March 17th, 2022

Tester(s): Muqtadir Ahmed

Author: Miiyu Fujita

Hardware version: 2.0

Software version: 1.0

Goal: Receive delivery of 6 cubes and push them off onto the rotating platform

1.0 Purpose of Test

The purpose of this test is to evaluate the performance of the cube/input tower. Evaluate whether it can receive a delivery of 6 cubes and push these cubes onto the storage space (rotating platform)

2.0 Test Objectives

- Determine if the tower can hold 6 cubes
- Determine whether the piston can push cubes onto the rotating platform

3.0 Test Procedure

- Deliver 6 cubes into the Input Tower
- Make the motor spin 360 degrees in order to push the cube and then come back to its original position
    - This should push the cube that is placed in front and prevent the cube above from falling.
- Check if the next cube falls into place after the motors full rotation is done

4.0 Expected Results

The expected results of this test is that the cube tower is able to hold 6 cubes and that it can push off the lowest cube onto the platform.

5.0 Format of Output Required

Results of the test should be written out as notes or pictures elaborating what difficulties the system faced during testing.

6.0 Test Report

The cube/input tower is able to hold 6 cubes.

The cube/input tower is unable to deliver the 6 cubes, due to a lack of force when pushing the cube onto the platform.

The cube/input tower is unable to deliver the 6 cubes, as the cubes regularly get stuck on their way down the tower.

7.0 Conclusions

The cube tower has yet to pass this test. The cube tower was unable to successfully deliver the 6 cubes onto the storage space, although it was able to hold the 6 cubes.

7.1 Further Action

Subject to these test results, adjustments will be made to the length of the piston, such that the cube tower has enough force to deliver the cubes successfully onto the platform. Adjustments will also be made to the physical makeup of the cube tower walls. A new component will be added in order to prevent the cube from getting stuck

7.2 Distribution of Work

The Hardware team will be responsible for making the above modifications.

D.2.2 Motor Gate Test Procedure
Group: (Design Project Team 01)

Date: March 20th, 2022

Tester(s): Muqtadir Ahmed

Author: Muqtadir Ahmed

Hardware version: 1.0

Software version: 1.0

Goal: Clamping on to the cubes above the lowest cube to prevent them from falling or getting stuck while the bottom cube is being pushed.

1.0 Purpose of Test

The purpose of this test is check if it is possible to hold the cubes from falling from the tower by using a clamping method to grip on the cubes using a motor connected to a vertical arm

2.0 Test Objectives

- Figure out how to transform the rotational movement of the motor into a linear movement
- Figure out where to apply the pressure when clamping
- Figure out the degree of rotation needed to open and close the gate without causing any errors
- Figure out how to mount the motor to a higher position

3.0 Test Procedure

- Connect the motor to port A
    - The motor should have a stable base that is connected to the input tower
- Make a structure such that the motor's rotational movement can be transform into a linear movement
    - Connect the motor wheel to 2 lego pieces that are connected with a pivot
    - Connect a long vertical bar at the end of the stick so that it can clamp all the cubes
    - Connect to bars on each side of the tower that will guide the motors pivot to move in a linear fashion
- Rotate the motor a little amount to see if the movement flows properly
- Write a code that allow the motor to turn approximately 20 degrees to close and then return to its initial angle to open
- Insert color cubes into the tower and run the program to close the gate and check if the cubes are well clamped
- Run the code to open the gate and check if the cubes are free to move

4.0 Expected Results

The gate should be able to clamp tightly on all cubes and we will be able to control the flow of the cube downwards. It will prevent the errors that were caused by the previous structure for which the errors were that the cube would get stuck on the walls of the tower and would not go down for it to be delivered onto the rotating platform.

6.0 Test Report

From the results, it can be seen that the cubes still tend to get stuck on the way down but more rarely. The gate motor does not stay accurate when closing and opening (either it does not close enough or it opens too much)

7.0 Conclusions

After the test, it can be seen that the component still needs to be modified in order for it to do its job. Solutions to the fix would be to apply more pressure to the 2nd lowest cube since that is the one that gets stuck when the clamp does not close properly. Changing the software to let the motor close tightly and to open the gate slightly. Making the base of the gate motor more stable will help the motor's body to stay stable whenever the motor spins.

D.2.3 Delivery System Test Procedure
Group: Final Project Group 01

Date: March 20th, 2022

Tester(s): Muqtadir

Author: Muqtadir

Hardware version: Initial prototype of the delivery system.

Software version: Initial code.

Goal: Goal is to calculate the accuracy of the delivery systems.

1.0 Purpose of Test
This test verifies that the delivery system is able to properly deliver the cube into the delivery area

2.0 Test Objectives
- Figure out how to precisely push the cube from the rotational platform into the delivery area
- Figure out the width of the arm in order to push the cube

3.0 Test Procedure
- Connect a motor to port A
- Insert a plus shaped long lego piece in between the motor and stack lego pieces at the edge of the plus shaped piece at a 90 degree angle
    - It should look sort of like a flag
- Make a base that is high enough to push off any cubes from the top of the rotating platform
- Mount the motor such that the rotation is rotating along the x-axis
    - Rotation along the x-axis should let the delivery system be more precise
    - It will only push one cube at a time without interfering with any other cubes
- Write a piece of code that rotates the motor by 360 degrees
- Place a cube in front of the motor and run the code to see if the cube gets pushed off
- Add a funnel which will guide the cube to the delivery area

4.0 Expected Results
It is expected that the cube is going to be pushed over. It is predicted that the push will not be accurate.

6.0 Test Report
The cube does get pushed over when the delivery system rotates 360 degrees. The accuracy depends heavily on the placement of the cube.

7.0 Conclusions
In order for this delivery system to be more accurate it must have a bigger funnel in order to catch the cube if it does not go straight towards the funnel. The arm of the delivery system should be thicker in order to push the cube even if it's slightly off of the desired position (to account for errors).

7.1 Further Action
- Build a wider funnel
- Make the arm thicker

7.2 Distribution of Work

The hardware team should work on that.

D.3 Week Four: Testing Procedures

D.3.1 Modified Input Tower Test Procedure

Group: Final Project Group 01

Date: March 25th, 2022

Tester(s): Nour, Muqtadir

Author: Miiyu

Hardware version: 3.1

Software version: 1.0

Goal: Test whether the input tower is capable of pushing off 6 cubes individually post-modifications.

1.0 Purpose of Test

Upon making changes to the input tower's pushing piston length, this test's purpose is to evaluate whether this modification was sufficient in order to ensure the cubes are delivered as desired. Previously, the cubes would get stuck after a cube is delivered on the front end of the piston, as the piston surface did not pull back enough after it had pushed the first cube. The modification was to make the piston's range of motion longer, such that it would pull back enough to avoid the cubes getting stuck on its surface. This test will determine whether this modification suffices to fix the issue of the cubes getting stuck in the tower, or if there are further modifications that must be made.

2.0 Test Objectives

- Determine whether all 6 cubes are pushed off of the input tower one at a time

3.0 Test Procedure

- Connect all necessary ports to the BrickPi and to the input tower
- Deliver 6 cubes into the input tower
- Run input_tower.py
- Observe the delivery of cubes

4.0 Expected Results

From this test, the expectation is that all 6 cubes will be pushed off of the input tower's stack one at a time, with the piston (now modified to have a longer pushing and pulling back distance) pushing it off.

5.0 Format of Output Required

The results of the test will be documented through notes and observations, and by quantifying the number of cubes the input tower can successfully push off of the stack individually.

6.0 Test Report

- 5 out of 6 cubes are successfully delivered. The last cube gets knocked off of the input tower's cube platform without the piston pushing the cube off.
- Piston arm gets stuck on an unknown component of the input tower, causing it to undergo a "flicking" motion as it goes through its cycle of pushing and coming back.

7.0 Conclusions

The test has proven that the modification to the piston arm's range has fixed the issue of the cube getting stuck on the piston's surface after the "sorting" of the first cube. However, it has introduced new issues that require attention, such as the fact that the last of the 6 cubes gets involuntarily knocked off of the platform without the piston pushing it, due to the fact that there are no other cubes that are holding it in place on the platform of the input tower. Another concern is the flicking motion of the piston arm. Although this does not disrupt the performance of the input tower in terms of this test, it may have repercussions when trying to integrate the tower with other subsystems, as the recoil of the flicking motion may cause other subsystems to fail. It is therefore worth looking into modifications that will resolve these newfound issues.

7.1 Further Action

Based on the results of this test and the newfound issues, there will be subsequent modifications to the input tower, in order to ensure that 1) the last cube does not get knocked off, and 2) the piston arm does not get stuck on another part of the input tower's structure.

7.2 Distribution of Work

Further modifications and testing of the input tower will be done by Muqtadir and Nour, who are the testers of this test as well.

D.3.2 Input Tower Platform Testing Procedure
Group: Final Project Group 01

Date: March 25th, 2022

Tester(s): Nour, Muqtadir

Author: Miiyu

Hardware version: 3.1.1

Software version: 1.0

Goal: Test whether the input tower is capable of pushing off 6 cubes individually post-modifications.

1.0 Purpose of Test

Upon making changes to the input tower's platform height in order to account for the last cube getting knocked off without being pushed off of the input tower, as well as modifications to the motor gate software, this test aims to observe whether this modification to the subsystem has proved to be successful in terms of keeping the last cube on the input tower until it is pushed off by the piston arm.

2.0 Test Objectives

- Determine whether the last cube can stay stationary in the input tower and wait to be deliberately pushed off by the piston arm

3.0 Test Procedure

- Connect all necessary ports to the BrickPi and to the input tower
- Deliver 6 cubes into the input tower
- Run input_tower.py
- Observe the delivery of cubes, specifically the last cube to be delivered

4.0 Expected Results

From this test, the expectation is that the last cube will wait to be deliberately pushed off of the input tower's stack.

5.0 Format of Output Required

The results of the test will be documented through notes and observations.

6.0 Test Report

- The last cube now waits for the piston arm to be pushed off of the platform. All 6 cubes are pushed off individually.

7.0 Conclusions

The test has proven that the modification to the input tower's platform height was effective in stopping the last cube in the stack to be delivered without being pushed deliberately by the piston arm. The suspicion was that due to the big drop of the cube upon being released from the hold of the motor gate, the last cube would bounce off of the platform unwillingly. The modification allowed for a smaller drop for each cube as it was released from the hold of the motor gate which prevented

unwanted bouncing, and therefore kept the last cube in place until it was pushed off by the piston arm. The input tower has passed this test.

7.1 Further Action

N/A

7.2 Distribution of Work

N/A

D.3.3 Input Tower Piston Testing Procedure
Group: Final Project Group 01

Date: March 25th, 2022

Tester(s): Nour, Muqtadir

Author: Miiyu

Hardware version: 3.1.2

Software version: 1.0

Goal: Test whether the flicking motion of the piston is avoided post-modifications to the input tower platform structure

1.0 Purpose of Test

Upon making changes to the input tower's platform such that the piston is guided as it moves back and forth on the platform, this test will help determine whether the flicking motion of the piston arm is now successfully avoided.

2.0 Test Objectives

- Determine whether the flicking motion of the arm is now avoided post modifications to the input tower platform.

3.0 Test Procedure

- Connect all necessary ports to the BrickPi and to the input tower
- Deliver 6 cubes into the input tower
- Run input_tower.py
- Observe the motion of the piston. Look for the occurrence of a "flicking" motion of the arm.

4.0 Expected Results

From this test, the expectation is that the piston arm will no longer get stuck on the platform, as there are now guides for the piston arm as it moves back and forth during the pushing motion.

5.0 Format of Output Required

The results of the test will be documented through notes and observations.

6.0 Test Report

- There is no longer a flicking motion detected during the run of the program.

7.0 Conclusions

The test has proven that the modification to the input tower's platform was effective in stopping the flick of the piston arm. The system has passed this test.

7.1 Further Action

N/A

7.2 Distribution of Work

N/A

D.3.4 Input Tower & Rotating Platform Integration Testing Procedure
Group: Final Project Group 01

Date: March 26th, 2022

Tester(s): Nour, Muqtadir

Author: Miiyu

Hardware version: 3.2

Software version: 1.0

Goal: Test whether the Input Tower and Rotating Platform subsystems can be integrated successfully

1.0 Purpose of Test

The input tower and the rotating platform are now connected. Thus, this test serves to determine whether the input tower and the rotating platform can now work in conjunction in order to successfully "sort" the colored cubes into the storage space.

2.0 Test Objectives

- Determine whether the input tower can successfully deliver 6 cubes onto different regions of the rotating platform, each at around 60 degrees apart.

3.0 Test Procedure

- Connect all necessary ports to the BrickPi, the input tower, and the rotating platform
- Deliver 6 cubes into the input tower
- Run main_robot_ferris.py
- Observe the delivery of cubes

4.0 Expected Results

From this test, the expectation is that the cubes are successfully delivered onto the rotating platform from the input tower.

5.0 Format of Output Required

The results of the test will be documented through notes and observations.

6.0 Test Report

- The last cube's color could not be detected by the color sensor, as it was placed too far back and was outside of the color sensor's field of view.
- One cube rolled off of the intended section of the rotating platform

7.0 Conclusions

The test has proven that the input tower and rotating platform are able to work in conjunction in order to "store" cubes. However, there are certain issues that have come up from this test, namely that 1) the color sensor's position has to be fixed in order to avoid errors related to the color sensor's FOV and 2) that there must be a way to control the cube position on the rotating platform such that cubes do not roll away from the region they are initially pushed off onto by the piston.

7.1 Further Action

Based on the results of this test, it is necessary to do further testing on the positioning of the color sensor, as well as determine a way to control the way the cubes fall onto their respective regions of the rotational platform.

7.2 Distribution of Work

Further testing of the regions of the rotational platform will be done by Muqtadir and Nour, and further testing of the positioning of the color sensor based on its FOV will be done by members who have the most time available during the upcoming week.

D.3.5 Input Tower & Updated Rotating Platform Integration Testing Procedure
Group: Final Project Group 01

Date: March 26th, 2022

Tester(s): Nour, Muqtadir

Author: Miiyu

Hardware version: 3.2

Software version: 1.1

Goal: Test whether the modifications to the physical dimensions of the rotating platform have resolved previously encountered issues in relation to the integration of the input tower and the rotating platform.

1.0 Purpose of Test

From the previous integration test between the input tower and rotating platform results, modifications were made to the physical dimensions of the platform and the distance between the input tower subsystem and the rotating platform subsystem. Namely, the platform was brought up in height, so as to decrease the drop the cube has to endure as it is pushed off of the input tower onto the rotating platform. The input tower and the rotating platform are now closer. The logic behind these changes is that with these changes, there is less momentum for the cube to roll off of its initial position that it is pushed off onto.

2.0 Test Objectives

- Determine whether the cubes delivered onto the rotating platform stay in place.

3.0 Test Procedure

- Connect all necessary ports to the BrickPi, the input tower, and the rotating platform
- Deliver 6 cubes into the input tower
- Run main_robot_ferris.py
- Observe the delivery of cubes and their positions on the rotating platform

4.0 Expected Results

From this test, the expectation is that the cubes are pushed onto the rotating platform such that they stay immobile after initial “sorting”.

5.0 Format of Output Required

The results of the test will be documented through notes and observations.

6.0 Test Report

- Successful delivery of all 6 cubes onto the platform, no cubes move from their initial positions
- The cubes now turn over onto the rotating platform without bouncing and excessive rolling

7.0 Conclusions

The test has proven that the modification to the height of the rotating platform has been effective in decreasing the chances of the cubes rolling off of their initial positions onto which they are pushed. 0 cubes out of 6 rolled out of their initial positions once pushed off of the input tower onto the rotating platform.

7.1 Further Action

N/A

7.2 Distribution of Work

N/A

D.3.6 System "Sorting" Activation Testing Procedure
Group: Final Project Group 01

Date: March 26th, 2022

Tester(s): Muqtadir, Miiyu

Author: Miiyu

Hardware version: 3.3

Software version: 2.0

Goal: Test whether the "sorting" process can be triggered by a simple switch, such that the system satisfies the requirement that it must be easy to use by the store clerk.

1.0 Purpose of Test

There is now an added touch sensor that is used to activate the system's sorting process. This is to ensure that the clerk can easily use the system without having to navigate the software.

2.0 Test Objectives

- Determine whether a press of the touch sensor can trigger the sorting system.

3.0 Test Procedure

- Connect all necessary ports to the BrickPi, the input tower, and the rotating platform
- Deliver 6 cubes into the input tower
- Run main_ferris.py
- Press the touch sensor to activate the system's "sorting" process
- Observe whether the system is activated upon the touch sensor being pressed

4.0 Expected Results

From this test, the expectation is that the system will start its "sorting" process once the touch sensor has been pressed.

5.0 Format of Output Required

The results of the test will be documented through notes and observations.

6.0 Test Report

- The system remains immobile until the touch sensor is pressed.
- Upon the touch sensor being pressed, the system's sorting process is activated, as expected.

7.0 Conclusions

The test has proven that the touch sensor is successfully controlling the activation of the system's "sorting" process, such that the touch sensor controls when to activate the preparation process the system must undergo before a request is made by the clerk.

7.1 Further Action

N/A

7.2 Distribution of Work

N/A

D.3.7 System Integration Testing Procedure
Group: Final Project Group 01

Date: March 26th, 2022

Tester(s): Muqtadir, Miiyu

Author: Miiyu

Hardware version: 4.0

Software version: 2.0

Goal: Test whether the system, now all fully connected, can work as a unit in order to sort and deliver cubes as requested.

1.0 Purpose of Test

The purpose of the test is to determine whether all the systems are successfully integrated. It will determine whether the touch sensor can activate the sorting process, such that all 6 cubes are delivered from the input tower to the rotating platform. Then, the test will evaluate whether the color sensor arm is capable of responding to requests made in the request area, such that the delivery subsystem may work in conjunction with the rotating platform in order to perform delivery of the cubes to the delivery area.

2.0 Test Objectives

- Determine whether the touch sensor is activating the system's sorting process
- Determine whether the input tower can "sort" the 6 cubes onto the rotating platform (storage space)
- Determine whether the delivery system can respond to requests in order to deliver the requested cube to the delivery area
- Determine whether the color sensor arm is capable of reading requests and relaying the information to the rest of the system for successful delivery of the requested cube

3.0 Test Procedure

- Connect all necessary ports to the BrickPi, and all other subsystems
- Deliver 6 cubes into the input tower
- Run main_robot_ferris.py
- Activate the system's sorting process by pressing the touch sensor
- Wait for the sorting process to be complete
- Use the color balls in the DPM kit in order to request a color at the request area
- Observe and evaluate the performance of the system in terms of whether it can deliver the correct cube when requested to do so

4.0 Expected Results

The expectation for this test is that the system works as a whole together in order to sort, read requests, and deliver requests as desired by the store clerk.

5.0 Format of Output Required

The results of the test will be documented through notes and observations.

6.0 Test Report

- Successful sorting of all 6 cubes
- Successful request and delivery of all 6 cubes

7.0 Conclusions

The test has shown that the system is integrated. It has passed the test, as the integrated system as a whole worked in tandem in order to successfully sort and deliver cubes upon request.

7.1 Further Action

Now that the system works in terms of its functionality, there is a need for specification testing pre-final demon, such that we may be sure that the system is meeting the client specifications.

7.2 Distribution of Work

Further testing of the specifications will be conducted by Muqtadir and Miiyu.

D.3.8 Sorting Process Timing Testing Procedure
Group: Final Project Group 01

Date: March 26th, 2022

Tester(s): Muqtadir, Miiyu

Author: Miiyu

Hardware version: 4.0

Software version: 2.0

Goal: Test whether the system meets the specification that the sorting process must take less than 2 minutes

1.0 Purpose of Test

The purpose of the test is to determine whether the system meets the system requirement that the sorting process must take less than 2 minutes.

2.0 Test Objectives

- Determine the time it takes for the system to complete the sorting process.

3.0 Test Procedure

- Connect all necessary ports to the BrickPi, and all other subsystems
- Deliver 6 cubes into the input tower
- Run main_robot_ferris.py
- Activate the system's sorting process by pressing the touch sensor
- At the same time as pressing the touch sensor, activate a stopwatch (iPhone Stopwatch)
- Wait for the sorting process to be complete
- Stop the stopwatch at the same time that the system is ready for delivery
- Note down the time it takes for the system to be ready for delivery from the press of the touch sensor

4.0 Expected Results

The expectation for this test is that the system will be able to sort the 6 cubes within a 2 minute margin.

5.0 Format of Output Required

The results of the test will be documented through notes and numerical time values that it has taken the system to complete its sorting process.

6.0 Test Report

- Successful sorting of all 6 cubes done in 24.53 seconds

7.0 Conclusions

The test has shown that the system meets the requirement that the system must be able to sort within a 2 minute margin from its activation until it is ready for delivery. The system has therefore passed the test.

7.1 Further Action

N/A

7.2 Distribution of Work

N/A

D.3.9 Delivery Process Timing Testing Procedure
Group: Final Project Group 01

Date: March 26th, 2022

Tester(s): Muqtadir, Miiyu

Author: Miiyu

Hardware version: 4.0

Software version: 2.0

Goal: Test whether the system meets the specification that the delivery process must take less than 5 seconds

1.0 Purpose of Test

The purpose of the test is to determine whether the system meets the system requirement that the delivery process must take less than 5 seconds

2.0 Test Objectives

- Determine the time it takes for the system to complete the delivery process.

3.0 Test Procedure

- Connect all necessary ports to the BrickPi, and all other subsystems
- Deliver 6 cubes into the input tower
- Run main_robot_ferris.py
- Activate the system's sorting process by pressing the touch sensor
- Wait for the sorting process to be complete
- Make a request at the request area using a colored ball from the DPM kit
- At the same time as making the request, start a stopwatch
- Wait for the delivery to be made to the delivery area
- At the same time as the cube being placed onto the delivery area, stop the stopwatch
- Note down the time it takes for the system to deliver a cube

4.0 Expected Results

The expectation for this test is that the system will be able to deliver the requested cube within a 5 second margin.

5.0 Format of Output Required

The results of the test will be documented through notes and numerical time values (table format, containing times for each of the 6 cubes as there may be discrepancies according to their position on the platform and their proximity to the delivery area) that it has taken the system to complete its sorting process.

6.0 Test Report

- Successful sorting of the different 6 cubes are all done within 5 seconds.

Cube	Time for Delivery (seconds)
1	2.25
2	1.79
3	2.23
4	2.89
5	2.15
6	1.81

7.0 Conclusions

The test has shown that the system meets the requirement that the system must be able to deliver requested cubes within a 5 second margin, as it is capable of doing so for all 6 cubes on the rotational platform, regardless of its relative proximity to the delivery area. The system has therefore passed the test.

7.1 Further Action

N/A

7.2 Distribution of Work

N/A

D.4 Week Five: Testing Procedures

D.4.1 Color Sensor Position Test

Group: Final Project Team 1

Date: March 30th, 2022

Tester(s): Muqtadir, Miiyu

Author: Miiyu Fujita

Hardware version: 4.1

Software version: 3.0

Goal: Determine whether angling up the color sensor fixes errors related to the color sensor occasionally failing to detect color of cubes in the stack

1.0 Purpose of Test

From the integration test between the input tower and the rotating platform, modifications were made to the angular position of the input tower's color sensor. This test's purpose is to determine whether angling the color sensor upwards resolves the non-detection issue.

2.0 Test Objectives

- Determine whether the color sensor (now angled upwards) can detect color of all needed cubes

3.0 Test Procedure

- Connect all necessary ports to the BrickPi and all other subsystems
- Deliver 6 cubes into the input tower, in a disordered manner
- Run main_robot_ferris.py
- Activate the system by pressing the touch sensor
- Observe the performance of the sorting process
- Repeat steps 2-5 8 times, to obtain results from 8 trials

4.0 Expected Results

The color sensor is now expected to successfully read all colors of all 6 cubes in the stack.

5.0 Format of Output Required

The results of this test will be documented through notes and observations.

6.0 Test Report

1st run: Successful sorting of 6 cubes, all cube colors detected.

2nd run: Successful sorting of 6 cubes, all cube colors detected.

3rd run: Successful sorting of 6 cubes, all cube colors detected.

4th run: Successful sorting of 6 cubes, all cube colors detected.

5th run: Successful detection of cube colors, but unsuccessful sorting process, as last cube landed on platform diagonally. When pushed onto the platform had a lot of momentum, which made it roll around on the rotating platform. Piston arm noticeably getting stuck during motion.

6th run: Successful sorting of 6 cubes, all cube colors detected.

7th run: Successful sorting of 6 cubes, all cube colors detected.

8th run: Piston arm gets stuck during motion.

7.0 Conclusions

The main objective of this test, which was to determine the effect of the modification made to the color sensor, has been met. Namely, the cubes are now all successfully detected. However, this test has introduced new concerns in terms of the piston arm getting noticeably stuck on the structure. The system has passed this test, but further testing will be needed to resolve the piston arm issue.

7.1 Further Action

Observing the way the piston arm gets stuck on the structure, it is suspected that the grey pieces of the piston are getting stuck on the holds of the LegoEV3 pieces of the input tower's cube platform. The problem may be resolved by replacing the grey piston pieces which seem to get stuck to a rounder LegoEV3 piece, or by covering the holes of the input tower cube platform.

7.2 Distribution of Work

Further testing will be conducted by Muqtadir and Miiyu, who also performed this test.

D.4.2 Piston Arm Surface Modifications Test
Group: Final Project Team 1

Date: March 30th, 2022

Tester(s): Muqtadir, Miiyu

Author: Miiyu Fujita

Hardware version: 4.2

Software version: 3.0

Goal: Determine whether replacing the LegoEV3 pieces helped to prevent the flicking motion of the piston during motion (i.e piston getting stuck during motion).

1.0 Purpose of Test

From previous tests (See D.4.1), it was observed that the piston was getting stuck during motion. The purpose of this test is to determine whether the lego piece modifications were effective in resolving the issue encountered.

2.0 Test Objectives

- Determine whether the replacement of LegoEV3 pieces was effective in resolving the issue of the piston arm getting stuck on the holes of the input

3.0 Test Procedure

- Connect all necessary ports to the BrickPi and all other subsystems
- Deliver 6 cubes into the input tower
- Run main_robot_ferris.py
- Activate the system by pressing the touch sensor
- Observe the performance of the piston arm mechanism
- Repeat steps 2-5 10 times to obtain 10 run results, or until the system fails

4.0 Expected Results

The piston now is expected to move smoothly during its pushing and pulling motion.

5.0 Format of Output Required

The results of this test will be documented through notes and observations.

6.0 Test Report

1st run: Observable flicking of the piston arm, the whole structure lifts up

2nd run: Piston is very forceful, still stuck on the structure. The whole structure lifts up.

7.0 Conclusions

From this test, it can be seen that replacing the lego pieces does not suffice in preventing the piston arm from getting stuck on the input tower cube platform's surface. The system has failed the test.

7.1 Further Action

From the results of the test, it is necessary to find another solution to prevent the piston from getting stuck on the tower's structure. Testing of other solutions will be done the following day.

7.2 Distribution of Work

Further testing will be conducted by Muqtadir and Miiyu, who also performed this test.

D.4.3 Piston Arm Surface Clear Tape Addition Test
Group: Final Project Team 1

Date: March 31st, 2022

Tester(s): Muqtadir, Miiyu

Author: Miiyu Fujita

Hardware version: 4.3

Software version: 3.0

Goal: Determine whether addition of clear tape allows for smooth piston movement.

1.0 Purpose of Test

From previous tests (See D.4.2), it was observed that the piston was getting stuck during motion despite changes made to the lego structure of the input tower. As it was observed that the piston was possibly getting stuck on the holes of the platform, clear tape was added to its surface in order to allow for a smoother gliding surface.

2.0 Test Objectives

- Determine whether the addition of clear tape was effective in resolving the issue of the piston arm getting stuck on the holes of the input

3.0 Test Procedure

- Connect all necessary ports to the BrickPi and all other subsystems
- Deliver 6 cubes into the input tower
- Run main_robot_ferris.py
- Activate the system by pressing the touch sensor
- Observe the motion of the piston arm, whether it is getting stuck or not
- Repeat steps 2-5 10 times to obtain 10 run results, or until the system fails

4.0 Expected Results

The piston now is expected to move smoothly during its pushing and pulling motion.

5.0 Format of Output Required

The results of this test will be documented through notes and observations.

6.0 Test Report

1st run: Piston no longer gets stuck during motion.

2nd run: Piston no longer gets stuck, but the structure still lifts up slightly

3rd run: Piston no longer gets stuck, does not lift up structure

4th run: Piston no longer gets stuck, but does lift up the structure

7.0 Conclusions

From this test, it can be seen that covering the holes with clear tape has prevented the piston from getting stuck. However, the whole structure still seems to lift up as the piston moves. The system passes this test, but there is still need for further action.

7.1 Further Action

From the results of the test, it is clear that it is necessary to find a solution that prevents the structure from lifting up during piston motion.

7.2 Distribution of Work

Further testing will be conducted by Muqtadir and Miiyu, who also performed this test.

D.4.4 Pre-Final Demo Test
Group: Final Project Team 1

Date: March 31st, 2022

Tester(s): Muqtadir, Miiyu

Author: Miiyu Fujita

Hardware version: 4.4

Software version: 3.0

Goal: Determine whether the full system functions as intended, now with added lego structure to prevent rods from moving (prevent structure from lifting).

1.0 Purpose of Test

Following modifications made to answer issues brought up during previous testing, the pre-final demo test's purpose is to evaluate the full system's performance subject to final demo conditions.

2.0 Test Objectives

- Determine whether the addition of the lego structure to prevent rods from moving has proved successful in preventing the structure from lifting
- Determine whether the system can answer to final demo conditions

3.0 Test Procedure

- Connect all necessary ports to the BrickPi and all other subsystems
- Deliver 6 cubes into the input tower
- Run main_robot_ferris.py
- Activate the system by pressing the touch sensor
- Observe the motion of the piston arm, whether it is getting stuck or not
- Repeat steps 2-5 10 times to obtain 10 run results, or until the system fails

4.0 Expected Results

The system is now expected to successfully store and deliver cubes, without all previous issues now being fixed.

5.0 Format of Output Required

The results of this test will be documented through notes and observations.

6.0 Test Report

1st run: Successful storing/sorting and delivery of cubes. Structure no longer lifts up.

2nd run: Successful storing/sorting and delivery of cubes. Structure no longer lifts up.

3rd run: Cube gets stuck, lands diagonally on the platform.

4th run: Successful storing/sorting and delivery of cubes. Structure no longer lifts up.

7.0 Conclusions

As the third run contained a pressing issue to be fixed, testing was stopped after the 4th run. The issue with the cube getting stuck during sorting means that the system has not passed this test. However, it is noted that the structure no longer lifts, indicating the addition of the rod stabilizing lego structure was useful in preventing the structure from lifting. The system has not passed this test.

7.1 Further Action

From the results of the test, it is clear that a solution must be determined in order to prevent cubes from falling diagonally onto the input tower's cube platform.

7.2 Distribution of Work

Further testing will be conducted by Muqtadir and Miiyu, who also performed this test.

D.4.5 Second Pre-Final Demo Test
Group: Final Project Team 1

Date: March 31st, 2022

Tester(s): Muqtadir, Miiyu

Author: Miiyu Fujita

Hardware version: 5.0

Software version: 3.0

Goal: Determine whether the full system functions as intended, now with added lego padding of the input tower wall.

1.0 Purpose of Test

Following modifications made to answer issues brought up during previous final demo testing, the second pre-final demo test's purpose is to evaluate the full system's performance subject to final demo conditions, now with added modifications.

2.0 Test Objectives

- Determine whether the addition of the additional padding on the input tower wall prevents cubes from landing diagonally on the input tower's cube platform.
- Determine whether the system can answer to final demo conditions

3.0 Test Procedure

- Connect all necessary ports to the BrickPi and all other subsystems
- Deliver 6 cubes into the input tower
- Run main_robot_ferris.py
- Activate the system by pressing the touch sensor
- Wait for the sorting process to complete
- Request each cube in storage one at a time, await delivery
- Repeat steps 2-6 10 times to obtain 10 run results, or until the system fails

4.0 Expected Results

The system is now expected to successfully store and deliver cubes, without all previous issues now being fixed.

5.0 Format of Output Required

The results of this test will be documented through notes and observations.

6.0 Test Report

1st run: Successful storing/sorting and delivery of cubes. Cubes land flat onto input tower cube platform.

2nd run: Successful storing/sorting and delivery of cubes. Cubes land flat onto input tower cube platform.

3rd run: Successful storing/sorting and delivery of cubes. Cubes land flat onto input tower cube platform.

4th run: Successful storing/sorting and delivery of cubes. Cubes land flat onto input tower cube platform.

5th run: Successful storing/sorting and delivery of cubes. Cubes land flat onto input tower cube platform.

6th run: Successful storing/sorting and delivery of cubes. Cubes land flat onto input tower cube platform.

7th run: Successful storing/sorting and delivery of cubes. Cubes land flat onto input tower cube platform.

8th run: Successful storing/sorting and delivery of cubes. Cubes land flat onto input tower cube platform.

9th run: Successful storing/sorting and delivery of cubes. Cubes land flat onto input tower cube platform.

10th run: Successful storing/sorting and delivery of cubes. Cubes land flat onto input tower cube platform.

7.0 Conclusions

All 10 runs of the full system subject to final demo conditions were successful. All cubes landed flat on their surface, and no cubes landed diagonally, thanks to the added padding of the back wall of the input tower. The system has passed this test.

7.1 Further Action

N/A

7.2 Distribution of Work

N/A