ECSE 211 – Section 001

Department of Electrical and Computer Engineering at McGill University

Design Principles and Methods

ECSE 211 - Fall 2021

Final Design Report

Authors: Dania Pennimpede (260986441), Liamo Pennimpede (260986442),
Sebastien Cantin (260979759), Martin Kruchinski Almeida (260915767),
Aman Chana (260895998) and Ali Therrien Motamedi (260986456)
Group Number 7

Date: November 29th, 2021

Table of Contents

Our Team

Our team is composed of 6 engineering students that all come from software and/or hardware backgrounds. Here is the breakdown of our team:

Liamo Pennimpede:

- U2 Software Engineer with a Minor in Management
- Liamo has been coding for the last four years, he knows Python, Java, SQL and JavaScript.
- Fun Fact:
    - Liamo has built his own desktop computer

Sebastien Cantin:

- U2 Computer Engineer
- Sebastien is involved in multiple Arduino projects and is well versed in Python and Java
- Fun Fact:
    - Sebastien loves soccer. He broke his ankle this summer playing said sport and now feels that he has the capability to overcome anything in his life.

Dania Pennimpede:

- U2 Software Engineer with a Minor in Finance
- Dania has management experience and is familiar with working with documents in an organized manner due to her executive positions in numerous clubs and organizations.
- Fun Fact:
    - When Dania was in high school, she went on a school trip to Vietnam where she biked over 100 km a day for 10 days straight, successfully making it across Vietnam.

Aman Chana

- U2 Software Engineer
- Aman only started coding upon entry to university but is well acquainted in Java, C and most recently arm assembly language.
- Fun Fact:
    - He registered in software engineering the day before the deadline, initially Aman hated it but now he has grown quite fond of programming.

Martin Kruchinski Almeida:

- U2 Computer Engineer
- Martin started coding in university and is now fluent in Java, Python, C, CSS, HTML, JS.
- Fun Fact:
    - Before choosing McGill engineering, Martin was going to attend the University of Virginia in Economics, because he got a scholarship to play in the soccer varsity team. Martin decided that he had to choose between Engineering or soccer, and he ended up choosing McGill.

Ali Therrien Motamedi:

- U2 Software Engineer
- Ali has been learning the Python programming language for the past 3 years and with the help of classes in the previous semesters, he has a considerable foundation in Java and C.
- Fun fact:
    - He built his own Personal Computer in one month and enjoys frequenting hackathons.

Team Organization

For our team to stay organized and on top of our tasks, we thought it would be best to implement individual roles for each team member, create a team contract that outlines our responsibilities, create a weekly schedule to organize our meetings and have tools to contact each other outside of our meeting times.

Roles

Our team decided on splitting up the responsibilities into 6 key roles: Design Team Manager, Design Documentation Manager, Hardware Team Leader, Software Team Leader, Software Engineer and Testing Team Leader. Each team member is charged with fulfilling their primary role as well as documenting their assigned work.

Here is our team allocation of the 6 roles:

- Design Team Manager: Martin Kruchinski Almeida
    - Tracks and manages the integration of the hardware and software components
- Design Documentation Manager: Dania Pennimpede
    - Allocates documentation tasks to members of the team and passes these allocations back to the team manager
- Hardware Team Leader: Sebastien Cantin
    - Manages the hardware engineering team - allocates tasks for the hardware systems and defines the documentation structures to be used.
    - Due to only having six members, Sebastian had to perform most of the hardware tasks, but had help from all team members
- Software Team Leader: Liamo Pennimpede
    - Manages the software engineering team - allocates tasks for the software systems and defines the documentation structures to be used.
    - Due to only having six members, Liamo had to perform most of the software tasks, but had help from all team members
- Testing Engineer: Ali Therrien Motamedi
    - Works with the Testing Team Lead to test the system, software, and hardware components to meet requirements
- Testing Team Leader: Aman Chana
    - Manages the testing engineering team - allocates tasks to test the system, software and hardware systems and defines the documentation structures to be used.

We choose our roles by having an open and honest group conversation where we discuss our preferences, strengths, and weaknesses, and then decide together on who would be responsible for each role.

Team Contract

To keep each other accountable, our group created a team contract that contains 10 rules that aid our design process. We believe signing this contract will formalize the expectations of our group members and outline the ground rules for the team.

The Ten Commandments of Group 7:

1. Our team hopes to have all deliverables complete the night before they are due, so there is time to review our work and there is no last second cramming.
2. We expect responses from our team no later than 24 hours concerning information about the design project.
3. We expect an inclusive environment where every member is encouraged and not frightened to share their ideas or complaints.
4. We require at least one team meeting per week to work on our weekly deliverables and demos.
5. Share individual weekly workloads with other team members to ensure that no one is overcharged with work.
6. There will be a brief 5-minute period at the beginning of each team meeting to discuss what we must accomplish and new ideas that we have thought of.
7. We expect each member to be punctual to the meeting time.
8. Each member of the team will make sure the work they are sharing with the group is solely their own.
9. Feedback is encouraged not sanctioned.
10. We expect to have fun!

Schedules

Our team decided on having, at the minimum, one meeting a week where we can all discuss and distribute the upcoming tasks. Our dedicated meeting time was on Friday's at 9:30 am to 10:30 am. If we had more work to do, we would also meet on Monday and Wednesday at 9:30 am to 10:30 am. Furthermore, we needed to set our weekly Wednesday meeting with our assigned Senior Engineer which we decided on having at 6:15 pm via Zoom.

We were able to decide the meeting times by listing all our availabilities for the week and comparing our class schedules:

- Ali Therrien-Motamedi: I am available every day after 7pm and during class time.
- Aman Chana: I am available from 7 pm to 9 pm from Monday to Thursday and 1pm to 4pm on the weekends. Also, I am available during class time.
- Martin Kruchinski Almeida: I am available to work every day after 4pm. Also, I can meet at class time every day.

- Sebastien Cantin: I am available after 5:30 pm MWF and have almost no obligations all day TTh. Also, I am available during class time.
- Dania Pennimpede: I am available after 2:30 pm on Mondays, after 1 pm on Tuesdays, after 12:30 on Wednesdays, between 8 pm and 10:30 pm on Thursdays, after 2:30 pm on Fridays. Also, I am available during class time.
- Liamo Pennimpede: I am available after 2:30 pm on Mondays, after 6 on Tuesdays, after 2:30 on Wednesday, between 4 pm and 12 am on Thursdays, between 2:30 and 6 pm on Fridays. Also, I am available during class time.

To recap, here is a simplified schedule of what was explained above:

	Monday	Tuesday	Wednesday	Thursday	Friday
9:30 am - 10:30am	OPTIONAL Team Meeting		OPTIONAL Team Meeting		Weekly Team Meeting
6:15 pm - 7 pm			Weekly Team Meeting with Senior Engineer		

Organizational Tools

The organizational tools that our team chose to help us all collaborate were to have a team Discord server and Facebook group chat where we can ask questions and collaborate on group tasks. In addition, we held each other accountable by creating weekly action items after our Wednesday Meetings with our designated TA. Furthermore, we wrote all our deliverable reports on a Google Document since we can all contribute at the same time.

Project Management

For our project to stay on track, our team decided to abide by a Gantt Chart and a time budget where we can monitor our weekly tasks and hours.

Gantt Chart

To facilitate the project management of our design system, we set up a Gantt Chart to assist in the planning and scheduling of our project. We believed creating a Gantt Chart was the most efficient tool for our project since it lists all the tasks and milestones that need to be done and provides information such as who is assigned to each specific task. In addition, it provides a clear overview of our team project over the design weeks. All in all, we were able to track our team's progress over the weeks by using and updating our Gantt Chart.

Furthermore, when developing our Gantt Chart, we tried to make our tasks as attainable as possible. In fact, since they were attainable, we were able to stick closely to our initial plans, however we slightly deviated as it took us more time to complete tasks than we thought. For example, we thought our system would not need any hardware or software improvement during Week 4, but since our system design was not working, we needed to add it into the Gantt Chart.

Please see the attached file that contains our team's Gantt Chart where you can find all our tasks and milestones!

Budget

Our team was able to create this budget by basing our tasks on what needed to be submitted each week for our deliverable and on how much we needed to improve and test our system design for our demos. Overall, as shown in the following table, our team finished the project on budget, since we all spent 45 hours in total over 5 weeks for this project. Even though we did not fulfill the 9 hours during the first week, we transferred the time to the final week because we knew the final report would take longer than anticipated.

Table 1: Budget breakdown

Tasks	Seb	Dania	Liamo	Ali	Amandeep	Martin
Week 1						
Timesheet	0.25	0.25	0.25	0.25	0.25	0.25
Weekly Meeting Agenda		0.25				
Translation of client needs into requirements, constraints, and specifications			1.5		1.5	
Brainstorm design ideas	1.5	1.5	1.5			1.5
Priority order of idea implementation		1.5				
Team Coordination Plan	0.5	0.5	0.5	0.5	0.5	0.5
Project Management Plan	1.5					
Budget Update	0.25	0.25	0.25	1	1	0.25
System description of the candidate design				1.5		1.5
Total for Week 1	4	4.25	4	3.25	3.25	4
Week 2						
Weekly Meeting Agenda		0.25				
Timesheet	0.25	0.25	0.25	0.25	0.25	0.25
Build hardware components	4.5	1.5	1			1
Project and budget status update		1.5				
Write update to the project plans		1.5				
Design software components			3			4
Test System	1.25		3	4	4	2

Reports from tests conducted			1.75	3	3	1.75
Outline of the initial prototype in the system, hardware and software	3	4		2	2	
Total for Week 2	9	9	9	9	9	9
Week 3						
Timesheet	0.25	0.25	0.25	0.25	0.25	0.25
Weekly meeting agenda	0.5	1	0	0	0	0
Post Beta Demo Analysis	1.75	1.75	1.75	1.75	1.75	1.75
Project and budget status update		1.5				
Write update to the project plans		1.5				
Refine Hardware System	5	2	1			
Refine Software System		1	4			4
Testing	1.5		2	4	4	2
Reports of tests conducted				3	3	1
Total for Week 3	9	9	9	9	9	9
Week 4						
Timesheet	0.25	0.25	0.25	0.25	0.25	0.25
Weekly Meeting Agenda	0	1	0	0	0	0
Project and budget status update		1.5				
Write update to the project plans		1.5				
Final Software improvements			3			3
Final Hardware improvements	4			1	1	1
Testing	2		2	5	5	2.75
Reports of final set of tests				2.75	2.75	2
Near final documentation of system, hardware and software designs	2.75	4.75				
Draft presentation for final demo			3.75			
Total for Week 4	9	9	9	9	9	9
Week 5						
Final Report	7	7	7	7.75	7.75	7
Project Debrief	4	4	4	4	4	4
Design Review	3	2.75	3	3	3	3
Total for Week 5	14	13.75	14	14.75	14.75	14
Total For all Weeks	45	45	45	45	45	45

As for our monetary budget, we decided to use 2 pieces of paper and some tape to build our system design. Fortunately, we had both items at home, but if you were to buy the material it would be 4 cents for the 2 pieces of paper and around 3 dollars for scotch tape. Therefore, it would cost around 4 dollars to own the two items including tax.

Translation of Client’s needs

For our team to fully comprehend what is demanded from us by our client when building our system, we translated the problem statement into 4 parts: Project Scope, Requirements, Constraints and Specifications.

Scope of the project

Our team was tasked by a large corporation that provides education and training services for postsecondary institutions to build a design system that can sort various coloured blocks under a specific amount of time. We were allocated this task to reduce the duration of time that students spend organizing boxes full of items at the end of their courses. Our team was given five weeks to complete this project, where we had to perform weekly tasks and submit weekly deliverables to ensure that our design system was progressing as planned and to keep everything on track.

Furthermore, our client gave us a budget of 0$ to cover the cost of the material but provided us with two BrickPis and two DPM kits that contained LEGO pieces, balls, and cubes of different colours to develop our design system. In addition, our team was permitted to use additional material, such as tape and paper, but would then be operating at a loss. As well, our team had a budget of 9 hours a week each to work on this project.

Moreover, from the problem statement, we were able to make some assumptions regarding our project. Our team assumed that we would be able to build our system successfully only using the resources provided to us and that we had no need to look for any more material. As well, we assumed that we would be able to go to the lab weekly to have a common space to work on our project. In addition, we assumed that all our equipment would be in good condition and that our team has the required skill to achieve a working design that fulfills the needs of the client.

Additionally, the large corporation was gracious enough to send us a brief description of their needs where we were able to deduce what the main requirements, constraints and specifications are for the design system. Here are the requirements, constraints, and specifications that our team was able to gather:

Requirements

- The system should be made of components found in the DPM design Kit
- The system should be automatic
- The system should be able to sort items based on their colors
- The system should be able to store these items in distinct locations where they can be easily grabbable
- The user should not need to re-sort the items after the system has finished

- The system should be able to sort a minimum of 10 foam cubes of various colours in 2 minutes and any additional items according to their color in 5 minutes.

Constraints

- We are limited to only using the Lego Mindstorms set and DPM kit for our components
- We are limited to time where we all share breaks from class or at night after school and work to test our machine in person.
- The cost associated with any additional material, not supplied by our clients, will imply that our project is operating at a loss since our client budgeted 0$ to cover the cost of material
- We are limited to only 20 foam cubes
- We are limited to a maximum amount of interaction a user must operate the system which is to load the unsorted set into a specific area and to start a system through a press of a button, or keyboard/mouse input.

Specifications

- Color sensor must be able to distinguish between yellow, red, orange, green, blue and purple without error.
- Transporter must be able to hold one block, place it in six separate locations and be accurate down to 2 cm.
- Tower must be tall enough to hold at least twenty blocks.
- Total system must sort each block in an average of 12 seconds
- The flicker must hit the blocks frequently enough to meet the time requirements but also ensure not too many blocks are on the conveyor belt at the same time
- Conveyor belt must drop blocks perfectly into the transporter to ensure no blocks fall out

Description of Our Designed System

Final System Design

After many iterations and designs, our team decided on a system that uses one colour sensor and four motors that all work cohesively. We believe that the combination of both the colour sensor and motors was the right choice as it made our design faster and more effective. The sensor and motors work cooperatively since they allow the colour blocks to flow easily through our system design.

The colour blocks start off by manually being loaded into the static tower that is made of two pieces of paper connected by tape. Then, our team built a flicker made of Legos, powered by one of the motors, that knocks the colour blocks from the static tower onto the conveyor belt. Once on the conveyor belt, the motor that powers the conveyor belt to spin allows the colour blocks to move until the colour sensor. While the colour sensor is scanning the colour of the block, the motor that makes the conveyor belt spin pauses for 4 seconds. After the sensor scans the colour, the block gently falls into our block transporter that is made completely from Legos.

The block transporter is powered by the final two motors that allow it to move the blocks to different locations and to change the angle of the transporter to let the colour blocks fall into the cups. Therefore, once the block is on the transporter and the colour sensor detects its colour, it moves to a specific location assigned to the scanned colour and then rotates to let the colour block fall.

The motors can move the block transporter to the assigned specific location largely due to our BrickPi system. In fact, since the colour sensor is branched to the BrickPi using cables, the colour scanned is sent to the Brickpi where our team created a software algorithm that instructs the motors to move the transporter to a specific location based on the block's colour and to rotate 90 degrees to let them fall.

Figure 1: Front view

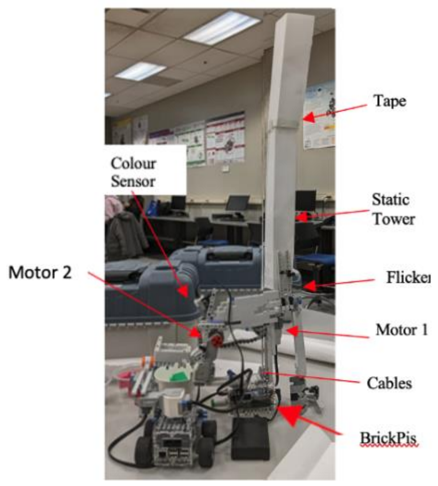


Figure 2: Top view

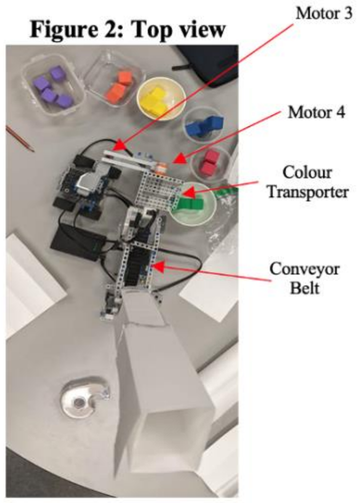
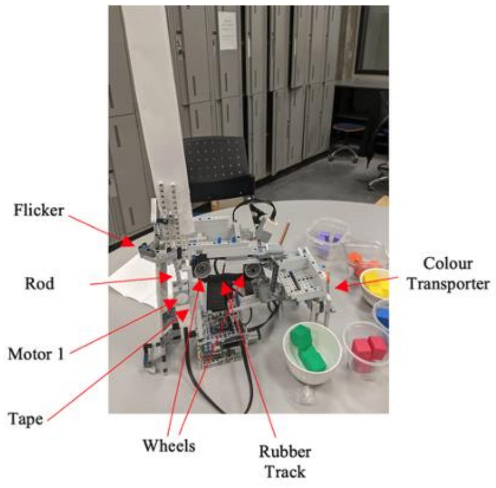


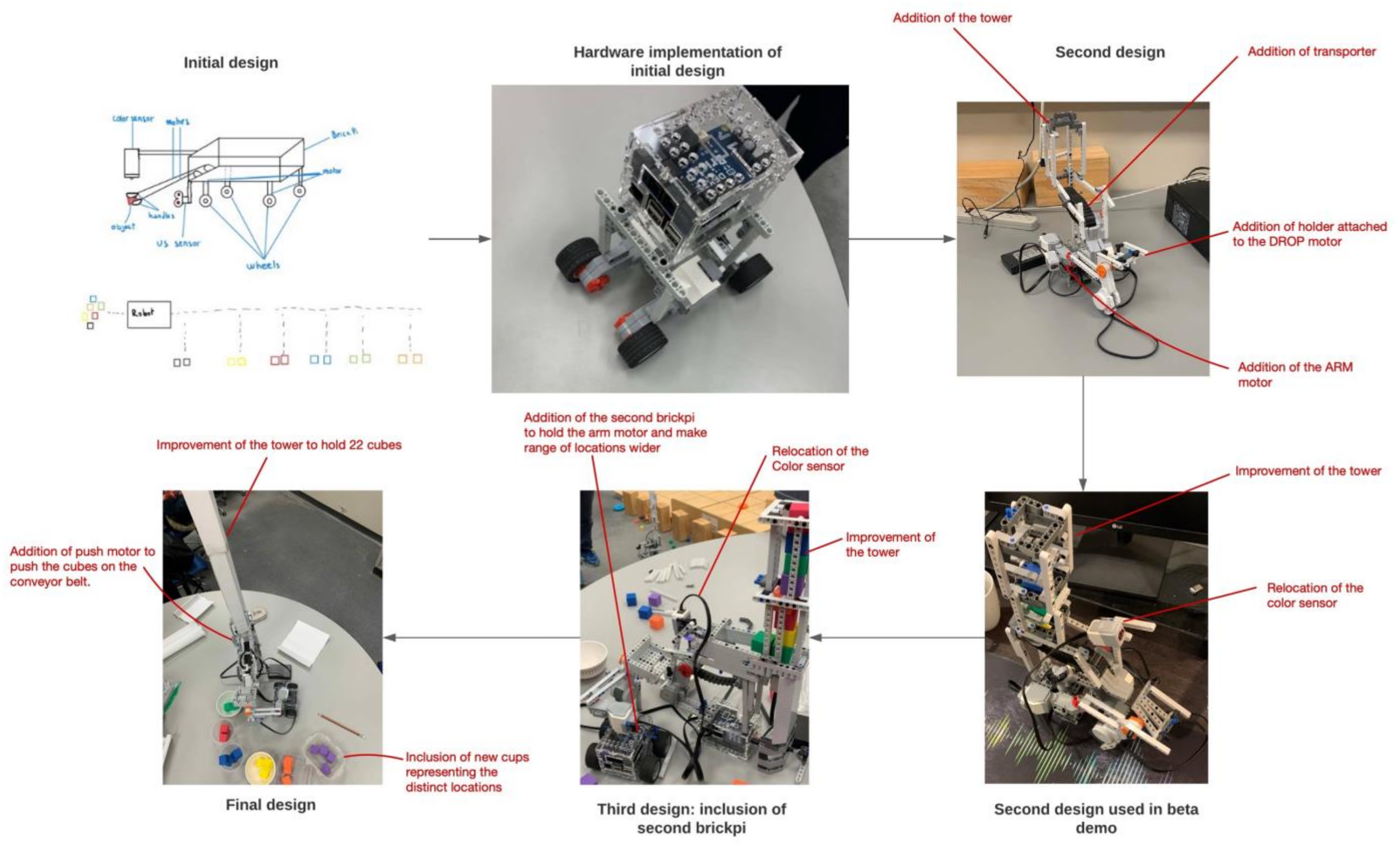
Figure 3: Side view



Iterative Design Evolution

Our design evolved quite a bit over its lifetime. Originally, the plan was a robot with wheels, two arms that would grab each block, a colour sensor to determine its colour and an ultrasonic sensor to detect the nearest block. It would grab one block at a time and drag it to one of the 6 planned locations. Due to fears of not achieving time constraints, a complete overhaul was done after week 1, choosing our backup design instead, a tower holding the blocks on a conveyor belt. At the end of the belt would be a colour sensor to detect the block's colour and an arm to place the blocks in different places after determining its colour. After the failure to properly perform at the beta demo, the tower was moved from directly on top of the conveyor belt to placing them on a platform and using a flicker (push motor) to smack them onto the conveyor belt.

Figure 4: Design evolution


Initial design
color sensor
motors
Brick
motor
handles
object
US sensor
wheels
Robot
Hardware implementation of initial design
Addition of the tower
Second design
Addition of transporter
Addition of holder attached to the DROP motor
Addition of the ARM motor
Improvement of the tower
Relocation of the color sensor
Second design used in beta demo
Addition of the second brickpi to hold the arm motor and make range of locations wider
Relocation of the Color sensor
Improvement of the tower
Third design: inclusion of second brickpi
Improvement of the tower to hold 22 cubes
Addition of push motor to push the cubes on the conveyor belt.
Inclusion of new cups representing the distinct locations
Final design

Final Hardware Design

There are four main components in our hardware design. The first is a static tower that holds all the cubes. Our team was able to assemble the tower by shaping the paper into a rectangular cylinder that had an open top and bottom. In fact, we were able to produce such a shape by measuring the length of a side of a colour block, adding one to two centimeters to that measurement, and folding the paper four times with such length. In addition, to hold 20 blocks, we were obliged to stick two pieces of paper together, using tape, to fit them all since one paper was not long enough.

The second is a flicker made up of Lego blocks that flicks the colour block from the static tower to the conveyor belt. As shown in Figure 4, the flicker is made up of three Legos that are bound together by two Lego connectors (technic axle pins). In addition, the flicker is connected by a rod to the motor that powers the flicking motion. To get the motor close enough to the flicker, we taped the motor to a long Lego piece that holds the whole design system up.

The third is a conveyor belt that moves the colour blocks until it reaches the transporter. Our team assembled it using a rubber track that was wrapped around two wheels at each end. Then, the wheel nearest to the colour sensor was connected to a motor that powered it to spin and rotate the conveyor belt. In addition, as shown in figure 4, we added walls made of multiple long Lego pieces to each side of the conveyor belt to ensure that the blocks would accidentally fall off when moving.

The fourth is a block transporter that moves the colour block to its assigned location and drops it in the cup. As shown in Figure 4, the colour transporter is square shaped and only has high walls built on two sides since the blocks need to fall from the other two sides. The transporter was placed on our second BrickPi, however this BrickPi was not powered, it was just so the transporter had a larger range of motion. Moreover, the transporter had two motors, one that allowed it to transport the blocks to different locations of sorting and another to allow it to drop the pieces where they belong.

All in all, our hardware components were either made from Lego blocks that can be found in the DPM kit or from pieces of paper and tape.

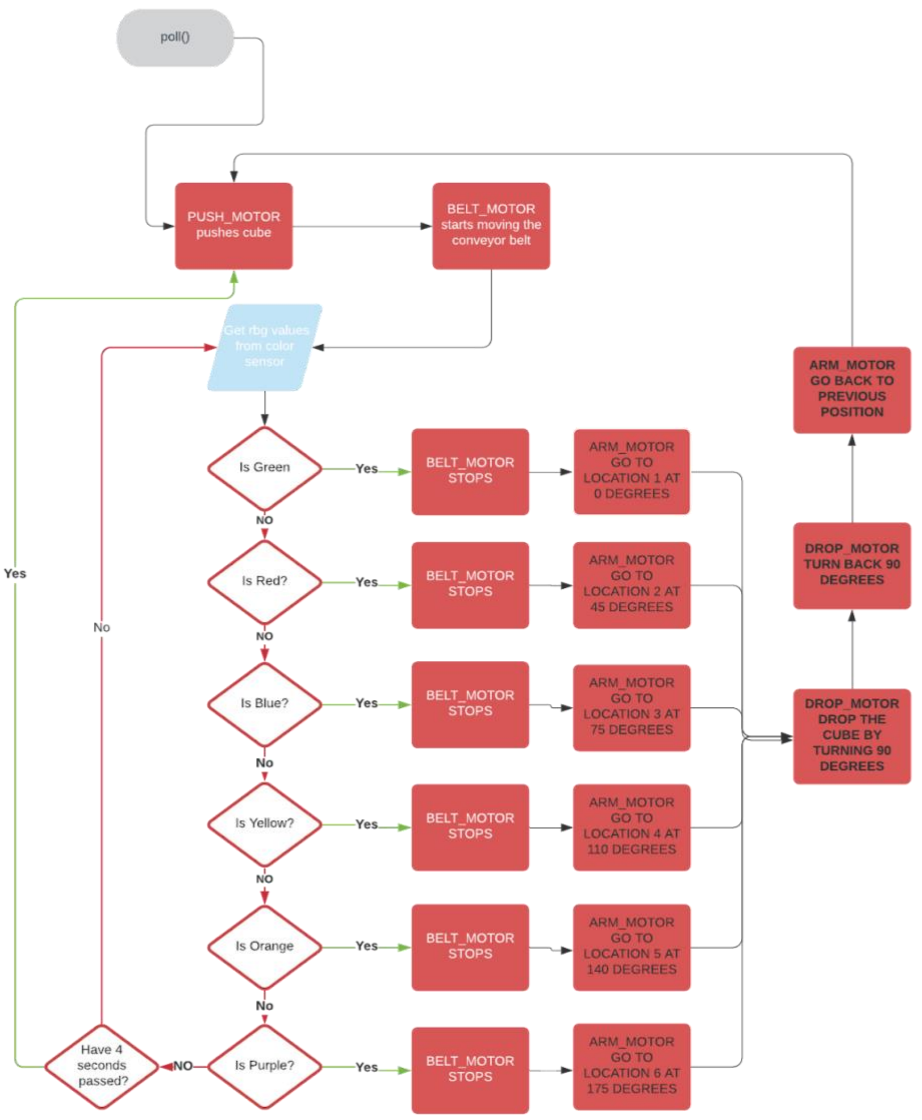
Final Software Design

There are two main parts in our software design. The first part is the movement of the colour blocks across the system and the detection of the block's colour. As shown in Figure 6, the methods push_motor, belt_motor and get_rgb were crucial to our first part of our software design. These methods allowed the BrickPi to send signals to our motors to flick the colour block from the static tower to the conveyor belt and to rotate the conveyor belt until the blocks reached the colour sensor. Once at the colour sensor, the method get_rgb constantly polls rgb values until it determines which colour the cube is out of red, orange, yellow, green, blue and purple. To detect the correct colour, we compared the obtained rgb values scanned of the colour block to the range of rgb values that we had for each colour.

The second part of our software system represents the methods arm_motor and drop_motor that allow the transporter to move to the desired location and rotate 90 degrees. Therefore, depending on the colour detected, the motor that moves the transporter turns to the set position of the colour and the motor that makes the transporter drop the cube rotates 90 degrees. Then, the two motors are returned to their initial positions and the program restarts for the other cubes.

Furthermore, to write the algorithm for the system, we first thought about how we wanted our hardware to behave and the different tasks it had to do. We used predefined functions from the brickpi class such as *set_motor_position_relative()*, *set_motor_power()*, *get_rgb()* to make our sensor and motors behave as we wanted to. Moreover, we implemented the algorithm to detect the color of the cubes by defining through testing an exact range of rgb values for every color, that we would constantly compare with the values that the color sensor reads. Also, the feature that makes the push motor push the cubes was invented by us since we wanted to deal with cases where the cubes would get stuck. Finally, we came up with the final sorting algorithm, where the arm motor would move a certain number of degrees depending on the location, using trial and error and trying to make our system as fast and efficient as possible.

Figure 5: Flow-chart of the software design:


poll()
PUSH_MOTOR pushes cube
BELT_MOTOR starts moving the conveyor belt
Get rbg values from color sensor
Is Green
Is Red?
Is Blue?
Is Yellow?
Is Orange
Is Purple?
Have 4 seconds passed?
Yes
No
NO
BELT_MOTOR STOPS
ARM_MOTOR GO TO LOCATION 1 AT 0 DEGREES
ARM_MOTOR GO TO LOCATION 2 AT 45 DEGREES
ARM_MOTOR GO TO LOCATION 3 AT 75 DEGREES
ARM_MOTOR GO TO LOCATION 4 AT 110 DEGREES
ARM_MOTOR GO TO LOCATION 5 AT 140 DEGREES
ARM_MOTOR GO TO LOCATION 6 AT 175 DEGREES
DROP_MOTOR DROP THE CUBE BY TURNING 90 DEGREES
DROP_MOTOR TURN BACK 90 DEGREES
ARM_MOTOR GO BACK TO PREVIOUS POSITION

Reference

This flow-chart represents our algorithm.

PUSH_MOTOR: Pushes the cubes that fall from the tower on the conveyor belt, using the calls:

```
BP.set_motor_position_relative(PUSH_MOTOR, 150)
sleep(0.1)
BP.set_motor_position_relative(PUSH_MOTOR, -110)
```

BELT_MOTOR: Moves the conveyor belt at a power of -12, using the following call:

```
BP.set_motor_power(BELT_MOTOR_1, -12)
```

Get rgb values: Gets the rbg values from the color sensor. Each color has a range of values. If the rgb values are within these ranges, then it detects the color. It uses the following call to get the rgb values:

```
COLOUR_SENSOR.get_rgb()
```

ARM_MOTOR: Moves the ARM_MOTOR a certain amount of degrees depending on the location, and then moves back the same amount of degrees after dropping the cube. It uses the following calls:

```
BP.set_motor_position_relative(ARM_MOTOR,degree)
BP.set_motor_position_relative(ARM_MOTOR,-degree)
```

DROP_MOTOR: Drops the cube by turning 90 degrees and then goes back to its initial position:

```
BP.set_motor_position_relative(DROP_MOTOR,90)
sleep(1)
BP.set_motor_position_relative(DROP_MOTOR, -90)
```

<u>Conducted Tests</u>

Milestone 1: Make all the motors and sensors work concurrently

Unit Test 1

<u>Testing</u>: Motor functionality

<u>Goal</u>: To determine if our written software activates and deactivates the motors

<u>Implication to meeting design specifications:</u>

Such a test would enable us to determine whether or not motions of the concepts we had in mind would be feasible to implement. In essence if the test passed then the fundamentals of our robot would be secure allowing us to meet the specifications of sorting cubes in separate piles based on color and displacing cubes from the tower into its designated area that is from a high area to a low one.

<u>Effect on design decisions (Hardware/Software/System Design):</u>

As a result of this test, our group was able to pursue the design outlined that is the creation of a motor-powered conveyor belt, holder, and the sorter. Furthermore, the test revealed that the ultrasonic sensor could halt the motor's movement, hence we decided to utilize a color sensor as a substitute for the ultrasonic sensor to not only check what complexion the cube exhibits but whether or not one is present.

<u>Limitations of different components of your system:</u>

This unit test reveals one flaw of the actuator: The delay associated with starting it and stopping it. Although it is minute it is something we must keep in mind because stopping slightly early or late could be the difference between the cube ending up in the holder or remaining on the track. Aside from this, no other limitations were noticeable given the simplicity of the test.

-Test results plus additional information can be found in the appendix in section 1.0.

Integrated Test 1

<u>Testing</u>: Motor and Sensor validation

<u>Goal</u>: Determine if all motors and sensors run concurrently based on our code

<u>Implication to meeting design specifications:</u>

This test is a step up from the previous one because our final design must both check the color of the block while turning the tracks at the same time which must come to a stop upon detecting a color. Ergo, the simultaneous nature of the robot means that by passing this test the requirement of building a sorting robot will be substantiated since it is critical to its development.

<u>Effect on design decisions (Hardware/Software/System Design):</u>

The implication on our decisions were substantial because not only are we now confident that the concurrency that we implemented, done by adding necessary instructions one after another then calling one function in main only, can be expanded to add more software features such as the transporter which is next in line. Moreover, we realized that threads were not necessary in our software application.

Limitations of different components of your system:

No noticeable limitations were observed considering the simplicity of the test. From the previous section, we are now aware of the motor delay, but the color sensor seems not affected in any way shape or form by the ambient light within its proximity. It is noteworthy that the sensor is held at most three centimeters away from the cubes under scrutiny.

-Test results plus additional information can be found in the appendix in section 1.1.

Integrated Test 2

Testing: Tower and Color Sensor functionality

Goal: Determine whether 10 blocks housed in a tower can correctly fall onto the conveyor belt and be readily detected by the color sensor.

Implication to meeting design specifications:

Although the test may seem insignificant in the grand scheme of the project, it was pivotal because it enabled us to house 10 cubes on our robot while attempting to sort them. This means that our robot does not need to scan for blocks on the ground then select them for placement in specific areas; rather now it can meet the specification informing users to refrain from interacting with the robot during processing by just selecting them and placing them. As such one fewer step is required, permitting us to sort the cubes in less time.

Effect on design decisions (Hardware/Software/System Design):

On a hardware level, this meant the construction of a tall yet delicate structure to support the 10 cubes. Furthermore, the weight distribution had to be ideal because if the front was lacking in mass, then the robot would be leaning backwards consequently, the motor of the track had to be shifted slightly forward to counteract this phenomenon. The color sensor had to be held in a spot with the aid of Lego blocks, facing the track only. To the extent of software, code to stop the track from moving when a cube’s detected was integrated into our design.

Limitations of different components of your system:

Surprisingly, the color sensor’s accuracy was 100%, not failing to detect a single-colored block; however, the tower does instill some fear given that cubes may get clobbered in the future as the motor runs faster to accomplish the sorting within two minutes. Therefore, perhaps another device is required to cease this from ever happening

-Test results plus additional information can be found in the appendix in section 1.2.

Milestone 2: Integrate the sorting algorithm with the hardware components

Integrated Test 3

Testing: Sorting ability

Goal: Determine the robot’s capacity to sort 10 cubes in preset piles

Implication to meeting design specifications:

The following integrated test encompasses most of the specifications listed by our members including color sensor processing as well as proper distinction, motor M1's continuous rotation, tower height requirements and the transporter's need to bring a certain block to its allocated destination that is testing its path directives as well because the current model is essentially our first iteration of the final robot.

Effect on design decisions (Hardware/Software/System Design):

Upon completion, we noticed that our insecurity from the previous test had been realized, the blocks would at times get stuck within the tower due to the faster rotary speed of the track. Hence, future iterations would require the tower to be placed independent of the track and a flicker must be incorporated to repeatedly push a cube onto the track until the tower was empty. Evidently, the code associated with the device would have to be added, in this case it would act as control to a gyro sensor.

Limitations of different components of your system:

The color sensor seemed to fail for some reason which came as a shock given that on the previous test it functioned without failure, in fact we deemed it reliable. Although the source of its abnormal behaviour is not fully understood we believe that it was caused due to misalignment reasons since the light was being reflected off the edge of the cubes only plus the grey Lego pieces surrounding it. Additionally, it took approximately fifteen seconds to sort each cube which would not suffice the specifications.

-Test results plus additional information can be found in the appendix in section 1.3.

Integrated Test 4

Testing: Color sensor functionality

Goal: Determine whether the color sensor and the algorithm running it can correctly detect the color of the colored cubes passed in front of the sensor.

Implication to meeting design specifications:

This test is essential to the functionality of our design given that it directly tests whether it meets the specification that it must be able to distinguish all the colors of the colored blocks of the DPM kit. It can be clearly seen that this is one of the most important aspects of the robot, if not the most important, given that its purpose is to sort colored blocks. To put this into perspective, without this functionality, it is irrelevant if all the other components of the design work perfectly since it will not fulfill the main requirement.

Effect on design decisions (Hardware/Software/System Design):

On a hardware level, this test led us to find the optimal position of the color sensor with reference to the cube it had to analyze. As this distance got either larger or smaller, the number of incorrect readings increased or decreased, which led towards which distance was the best. In terms of software, it was also helpful because it allowed us to find a few small issues in the code, making it as efficient as possible for us. All in all, this had significant effects on our system design since it helped us make adjustments that made our design be more efficient and accurate.

Limitations of different components of your system:

At first, the color sensor and the code running it were nearly perfectly accurate since they could classify all the cubes correctly except the yellow ones. On certain occasions, the yellow cubes were categorized as purple cubes. With some changes made to the code and the position of the sensor, this issue was fixed, making the sensor classify cubes with perfect accuracy.

-Test results plus additional information can be found in the appendix in section 1.4.

Integrated Test 5

Testing: Transporter functionality

Goal: Determine whether the robot can correctly deposit the cubes in their assigned color-coded locations once its color is detected.

Implication to meeting design specifications:

This test ensures that the transporter is properly able to hold one block and place it in one of the six allocated positions based on its colour

Effect on design decisions (Hardware/Software/System Design):

This test helped determine the optimal movement speed for the arm so it would not go too fast and move the BrickPi or too slow and not meet requirements. It also helped determine that a better position was needed for the transporter, since it did not have a large enough range of motion to store the blocks in 6 different locations.

Limitations of different components of your system:

The cups being used to hold the blocks are a major limitation as it is difficult to find any that are heavy enough to not get moved by the blocks, big enough to hold 4-5 blocks yet small enough to have 6 spots available within the transporter's radius

-Test results plus additional information can be found in the appendix in section 1.5.

Milestone 3: Synchronize the unloading of the cubes from the tower

Integrated Test 6

Testing: Flicker functionality

Goal: Determine if the push motor successfully pushes cubes from the tower onto the conveyor belt

Implication to meeting design specifications:

This test ensures that the machine is able to move blocks from the tower to the holder that places the blocks and meets the requirement concerning the frequency of hits so that there are no jams on the conveyor belt.

Effect on design decisions (Hardware/Software/System Design):

The optimal position to place the flicker was discovered during this test and required a minor redesign of the tower to accommodate this new position as well as the optimal timing to ensure not too many blocks make their way on the conveyor belt. This test allowed construction of the final tower to begin for it to be tall enough to hold all the blocks.

Limitations of different components of your system:

It is a semi-common occurrence for the flicker to require many hits to properly knock a block onto the conveyor belt if it has more blocks on top of it, so if the sensor does not detect a block within four seconds, the flicker hits as well. Rarely, the flicker hits far too hard for little reason and pushes its Lego connectors out of the frame and falls off.

Test results plus additional information can be found in the appendix in section 1.6.

Milestone 4: Final design is ready for the final demo

Integrated Test 7

Testing: Entire System's functionality

Goal: Determine whether the new design using a higher cube tower and a second BrickPi can correctly store colored cubes in their color-coded location.

Implication to meeting design specifications:

If this test is to be successful, the system must be meeting all the design specifications, or failing by a very small margin

Effect on design decisions (Hardware/Software/System Design):

No effects since the test was successful and tested the final state of the system and all the constraints

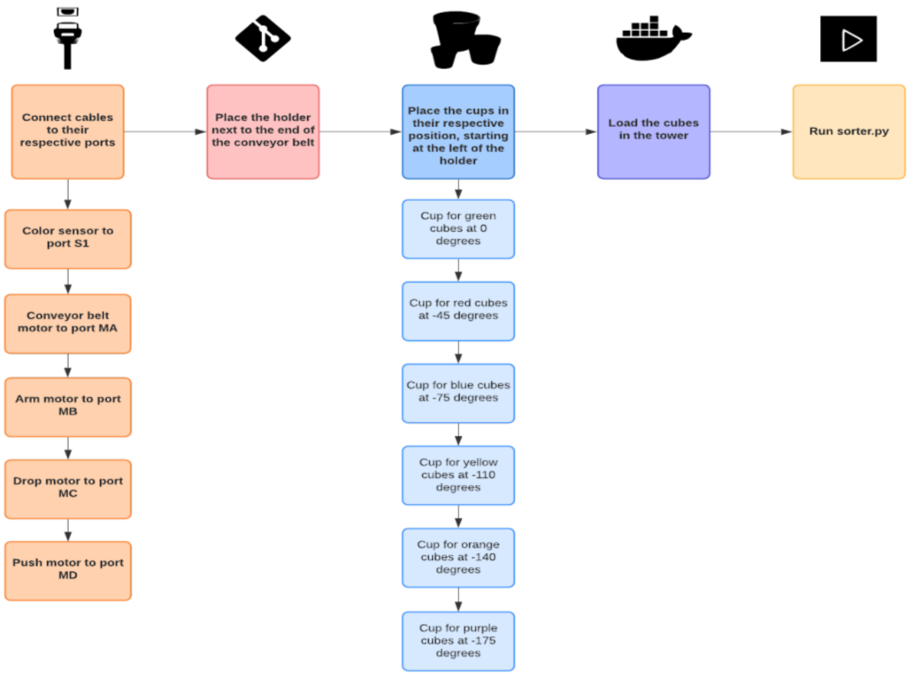
Limitations of different components of your system:

The transporter's arm length meant that small cups were needed to hold the blocks in their final location, and this resulted in lighter cups being used that can get slightly moved when a block is placed in them. The final five blocks often get caught at the joint in the tower where the two pieces of paper join each other. The cable for the arm of the transporter sometimes gets caught on the BrickPi it was on, resulting in either the arm not moving all the back to the conveyor belt or the BrickPi moving a little bit, if the robot had to sort many more cubes this could become a problem, but due to the short runtime of the robot, it was not deemed a serious problem.

Test results plus additional information can be found in the appendix in section 1.7.

Performance and Limitations

Figure 6: How to use the system


Connect cables to their respective ports
Color sensor to port S1
Conveyor belt motor to port MA
Arm motor to port MB
Drop motor to port MC
Push motor to port MD
Place the holder next to the end of the conveyor belt
Place the cups in their respective position, starting at the left of the holder
Cup for green cubes at 0 degrees
Cup for red cubes at -45 degrees
Cup for blue cubes at -75 degrees
Cup for yellow cubes at -110 degrees
Cup for orange cubes at -140 degrees
Cup for purple cubes at -175 degrees
Load the cubes in the tower
Run sorter.py

System capabilities and performance:

Sorting:

The system is 80% of the time able to sort 10 cubes in 2 minutes and 22 cubes in 5 minutes. In fact, 100% of the time, the system is able to sort 22 cubes in under 4 minutes if the set up was done correctly. All 6 types of coloured cubes will be found in 6 distinct locations at the end of the sorting.

Colored cubes:

The robot is able to detect all the coloured cubes very accurately even if the cube was close to the color sensor for just 0.1 seconds. The system can precisely detect the color of green, red, orange, blue and purple cubes every time, while it can detect a yellow cube 95% of the time.

Autonomy:

Once the program has started, the system is able to sort the cubes autonomously without human interaction.

System limitations:

Push motor:

The push motor might take a long time pushing the cube onto the conveyor belt depending on the position of the cube. That might affect the timing of the sorting. Also, the connection of the push motor to the robot is very fragile. In fact, 1% of the time, the push motor would detach from the robot.

Color recognition:

The color recognition algorithm consists of constantly polling rgb values using the color sensor and comparing them to a range of values that represent each color. That being said, sometimes, the color sensor might detect the color purple when it is supposed to detect yellow. That is due to the fact that the color sensor might be detecting colors from only the edge of the cube. We improved our algorithm to deal with this problem, but 5% of the time it might detect the incorrect color for yellow.

Tower:

The tower has two parts, one that holds 16 cubes and an added part on top to hold the rest of the cubes. The connection between the two parts is not perfect, which might lead to the cubes being stuck if they are not perfectly loaded.

Appendix

1.0: Unit Test 1

Date: November 08, 2021

Tester: Martin & Liamo Pennimpede

Hardware version: Robot version 1.0

Software version: 1.5

Goal: Determine if our software implementation moves the motors.

Procedure: Robot is being lifted by one teammate while the other teammate executes the program file that contains the relevant code.

Expected Results: The two back wheels of our robot should continuously spin after the activation of our motor, only stopping if the person executing the test put their hand in front of the ultrasonic sensor.

Test report: The test was performed three times in accordance with the protocol. The first time, only one of the wheels spun and after carefully examining that each wire was wired correctly, we noticed that the wire connecting the second motor was not inserted. After attaching the wire correctly, we proceeded with our two other tests. For the next two tests, the robot's back wheels continuously spun after every run and stopped when we put our hand in front of the ultrasonic sensor.

Conclusion: The robot's back wheels were able to spin and were able to stop after the activation of the ultrasonic sensor.

Action: Test should be reviewed one more time by the member who is responsible for it. The Gantt chart should be updated to show revised tasks.

Distribution: Software development & project management

1.1: Integrated Test 1

Group: Main Project Group 7

Date: November 9, 2021

Tester: Martin, Dania, Liamo, Sebastien, Aman, Ali

Author: Martin Kruchinski

Hardware version: Robot version 1.1

Software version: 1.6

Goal: Determine if all the motors and sensors of the robot run concurrently using the software we implemented.

Procedure: Place a cube in the conveyor belt and start the program. Since all the hardware is not put in its correct spot as yet, scan the object with the US sensor when it arrives at the holder and then scan the color of the object. The program should then be autonomous.

Expected Results: The Conveyor belt should move the cube and drop it in the holder. Then, the US sensor should detect the object and stop the conveyor belt, then the color sensor should detect the color of the object and make the holder move to the desired position. Finally, the holder turns 90 degrees to drop the object, all the elements come back to their original position and the belt starts moving again.

Test report: The test was performed many times. We fixed our software according to some bugs we have. In the end everything worked out as expected.

Conclusion: The system performance did meet our requirements.

Action: Reorganize the hardware so that the whole system can be autonomous, place the color sensor and the ultrasonic in the desired position to detect the object when the conveyor belt drops it in the holder.

1.2: Integrated Test 2

Group: Main Project Group 7

Date: November 11th, 2021

Tester(s): Everyone

Author: Aman Chana

Hardware version: Robot Version 1.2

Software version: Software Version 1.7

1.0) Goal of the Test

The purpose of this test is to determine whether the blocks correctly fall into place on the tracks from the tower that houses the 10 cubes, then verify that the color sensor can differentiate accurately between each block color after exiting the tower. To surmise, this test merges the displacement of cubes across a conveyor belt along with color sensor activation.

2.0) Test Procedure

The following test should be performed three times for constant results

1. Connect the Color Sensor to port S1 and the actuator to port M1

a. Ensure the Color Sensor as well as the actuator are firmly attached to the robot.
b. Use the battery to conduct the test to avoid motor failures

2. Run the function main () on the robot which should contain two functions in an endless loop: One that calls the color sensor and the other that calls the motor control function

a. Place the robot on a table
b. Place 10 blocks sequentially in the tower
c. Ensure that conveyer belt is strapped on correctly
d. Run the python code
e. Check to see if cubes read by the color sensor are indeed accurate using print statements in the color sensor function

3.0) Conclusion

The test was successful in properly processing the cubes. As a result of this test, our group is now ready to move onto to the sorting algorithm

1.3: Integrated Test 3

Group: Main Project Group 7

Date: November 13th, 2021

Tester(s): Everyone

Author: Aman Chana

Hardware version: Robot Version 1.3

Software version: Software Version 1.8

1.0) Goal of the Test

The goal of this test is to assess the robot's capability to sort 10 cubes out in set piles. Note that this test combines tower stacking, proper conveyor belt movement, color sensing, displacement of the robot via the tires and accurate knowledge of where to place the cubes.

2.0) Test Procedure

The following test should be performed three times for constant results

1. Connect the Color Sensor to port S1 and three actuators to port M1, M2, M3 respectively 2 for the wheelbase, one for the conveyor belt)
   a. Ensure the Color Sensor as well as the actuators are firmly attached to the robot.
   b. Use the battery to conduct the test to avoid motor failures

2. Run the function main () on the robot which should contain two functions in an endless loop, one that calls the motor control function and one that calls the color sensor but if it detects a color within the parameters of the blocks it must call a routine within it that displaces the robot to the desired location and back
   a. Place the robot on the ground
   b. Place 10 blocks sequentially in the tower
   c. Ensure that the conveyor belt is strapped on correctly
   d. Monitor the playground for any potential disturbances the robot may incur, remove them if any
   e. Run the python code
   f. Check to see if cubes read are sorted in the separate locations depending on their color

3.0) Conclusion

The test was not successful, as the robot was only able to properly sort out 40% of the cubes. Thus, our group must now return to the drawing board to fix the software aspect of our project given that the hardware side had no fallouts.

1.4: Integrated Test 4

Group: Main Project Group 7

Date: November 18th, 2021

Tester: Martin Kruchinski & Ali Therrien-Motamedi

Author: Ali Therrien-Motamedi & Aman Chana

Hardware version: Robot version 2.0

Software version: 2.0

Goal: Determine whether the color sensor and code running it are capable of correctly detecting each color of the colored blocks in the DPM kit.

1.0 Purpose of Test

We want to know whether colors that are clearly different to the eye are also properly distinguished by the color sensor algorithm. This is one of the most important subcomponents of the build given that the functionality of the entire system and of many of the other subcomponents depend on its correct behaviour.

2.0 Test Objectives

We want to obtain a color sensor and the Python code behind it capable of correctly categorizing all the differently colored cubes of the DPM kit without any mistakes. This should be accomplished whenever all the cubes of the kit are passed in front of the sensor. The color sensor should correctly analyze the color of the cube placed in front of it so that the behaviour of the subcomponents dependent on this sensor (e.g. robotic arm) can correctly do their tasks.

3.0 Test Procedure

The following test should be performed three times for each differently colored cube to ensure valid results.

1. Connect the color sensor of the DPM kit to a BrickPi already connected to the computer on which the Python code written for this component is running.
    1. You must make sure the battery is connected to the robot and it is sufficiently charged
2. Set up the robot correctly by following the successive steps
    1. Place the robot on a table
    2. Place a colored cube on the conveyor belt at a position right under the color sensor
    3. Run the Python code for the color sensor

4. Once having observed what color is detected, take note of whether the sensor has successfully detected the color of the cube.

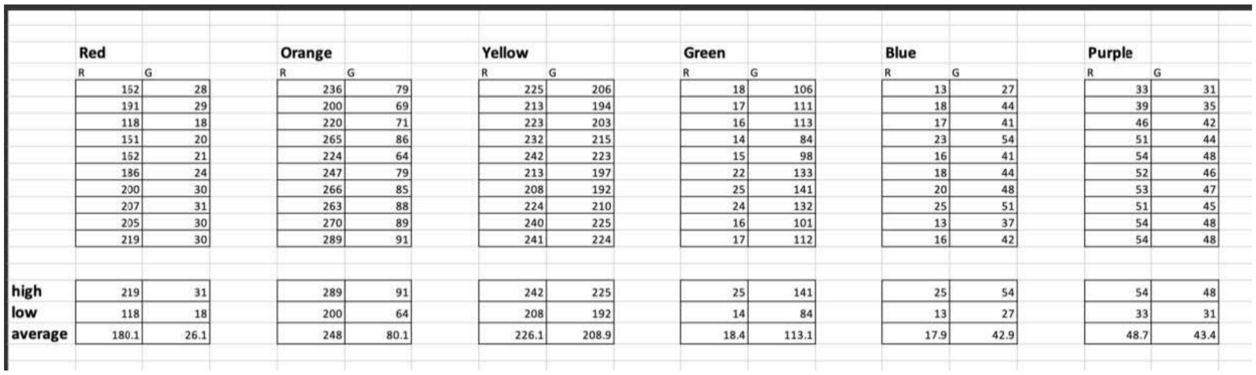
4.0 Expected Results

The test should demonstrate that the color sensor and the code that runs it should detect all the colors of the cubes at 100% success rate. In other words, none of the colors of the cubes should be confounded with each other.

5.0 Format of Output Required

The results should be formatted as a table containing all the readings of the colored cubes passed in front of the sensor. They should be categorized per color to show what data the sensor outputs whenever faced with each color.

6.0 Results



	Red		Orange		Yellow		Green		Blue		Purple	
	R	G	R	G	R	G	R	G	R	G	R	G
	162	28	236	79	225	206	18	106	13	27	33	31
	191	29	200	69	213	194	17	111	18	44	39	35
	118	18	220	71	223	203	16	113	17	41	46	42
	151	20	265	86	232	215	14	84	23	54	51	44
	162	21	224	64	242	223	15	98	16	41	54	48
	186	24	247	79	213	197	22	133	18	44	52	46
	200	30	266	85	208	192	25	141	20	48	53	47
	207	31	263	88	224	210	24	132	25	51	51	45
	205	30	270	89	240	225	16	101	13	37	54	48
	219	30	289	91	241	224	17	112	16	42	54	48
high	219	31	289	91	242	225	25	141	25	54	54	48
low	118	18	200	64	208	192	14	84	13	27	33	31
average	180.1	26.1	248	80.1	226.1	208.9	18.4	113.1	17.9	42.9	48.7	43.4

7.0 Conclusions

This test demonstrated that the color sensor and the Python code implemented into it were able to correctly detect the color of the colored cubes in the DPM kit, except for the yellow ones. It occurred multiple times where the yellow-colored cube was categorized as purple by the robot, which is completely wrong. This was worrisome since yellow is the opposite of purple.

7.1 Further Action

The observation that the yellow color was not correctly detected led us to slightly modify certain possible causes of this problem, such as light angles, distance between the cube and the color sensor and the Python code. We then did more color polling to verify if our changes had fixed the issue and the yellow color was finally correctly detected by the color sensor on our robot. This test will be conducted by Martin K, while Ali T-M takes note of the successful detections done by the sensor.

7.2 Distribution of Work

Two team members will work on finding the right distance to have the colored cubes be in reference to the sensor while two members devote time to making any possible improvements to the code on which the sensor is running.

1.5: Integrated Test 5

Group: Main Project Group 7

Date: November 18th, 2021

Tester: Martin Kruchinski & Ali Therrien-Motamedi

Author: Ali Therrien-Motamedi & Martin Kruchinski

Hardware version: Robot version 2.1

Software version: 2.1

Goal: Determine whether the robot can correctly deposit the cubes in their assigned color-coded locations, once its color is detected.

1.0 Purpose of Test

The purpose of this test is to verify that the motors controlling the robotic arm that drops the cubes into their locations move at the optimal speed so that the event where a cube rolls or falls off the robot does not occur. The motors that are part of the arm can go really fast, making it possible for the cubes to overshoot. This test has the purpose to verify that this does not happen.

2.0 Test Objectives

We want to obtain a robotic arm capable of displacing the cubes from the conveyor belt to their color-coded locations as fast as possible without having any of the cubes be mistakenly dropped. The robot arm should have the cube gently fall onto it once it is off the conveyor belt after its color has been detected by the color sensor. It should then move the block at optimal speed towards its assigned location and drop it, so that as many cubes as possible can be sorted successfully in the least amount of time.

3.0 Test Procedure

Repeat these steps for every single block of different color that is part of the DPM kit:

1. Run the Python code for the robot and wait until the motors and sensors are active
2. Feed a colored block to the conveyor belt for it to be analyzed by the color sensor
3. Wait to observe where the block is dropped by the robot arm
4. Take note of whether this block is correctly deposited and which color it is
5. Take note of the apparent reason the block is not correctly deposited, if it is the case

4.0 Expected Results

The test should reveal that the robotic arm deposits the majority of the blocks onto their assigned location, with only a few cubes either rolling off the robot or possibly getting stuck on certain lego blocks.

5.0 Format of Output Required

The output should be a visual observation of the operation of the robotic arm during this test and a series of notes on whether the cubes are correctly displaced by the arm. Since we want to verify that the cubes are moved at the right speed for them to be physically categorized correctly by the system, it is only necessary to see this operation of the arm and keep a series of notes on the movement of the cubes.

6.0 Conclusions

This test demonstrated that our robot is well capable of organizing the colored blocks in their correct locations. It has shown that the speeds we used to move the parts of our system around are not right and can sometimes cause issues, such as pushing cubes too hard or moving them too slowly.

6.1 Further Action

This will require us to make some slight changes in the Python code to find the correct speeds for the motors in the robotic arm, so that no such issues reoccur. To do so, a method of trial and error will be necessary to try different speeds for the motors, in order to find the optimal ones.

6.2 Distribution of Work

One team member will be responsible for taking note of whether each block is moved and dropped correctly, while another member loads the blocks one at a time onto the conveyor belt. This test will be completed by Martin Kruchinski, and Ali Therrien-Motamedi.

1.6: Integrated Test 6

Group: Main Project Group 7

Date: November 19th, 2021

Tester(s): Martin Kruchinski & Sebastien Cantin

Author: Aman Chana

Hardware version: 3.0

Software version: 3.0

Goal: Determine if the arm successfully pushes cubes from the tower onto the conveyor belt

1.0 Purpose of Test

The purpose of this test is to verify that upon an empty holder where no cube is detected by the color sensor, the robotic arm, in charge of knocking cubes onto the track, performs its duty while not compromising the integrity of the tower or interfering with subsequent blocks falling into place.

2.0 Test Objectives

We want to obtain a functional device that can automatically bump the colored cubes from the tower onto the conveyor belt. This must be done when the container is empty, and the color sensor has not detected a block. In such a case, the robotic arm should push a single cube onto the track only, and quickly return to its starting position without being caught under the remaining cubes in the tower. This arm should be activated n times where n is the number of cubes in the tower.

3.0 Test Procedure

The following test should be performed three times with distinct cubes to ensure valid results

1. Connect the PUSH_Motor to port MD, the BELT_MOTOR for the conveyor belt to port MA, the actuator for the rotation of the holder to port MB and motor responsible for the tilting of the holder to port MC, and the color sensor to PORT_1 a. Ensure the sensors are firmly attached to the I/O ports.

a. You must use the battery to power the robot

2. Set up the robot correctly by following the successive steps

a. Place the robot on a table
b. Stack 10 cubes into the tower carefully ensuring no interference with surrounding Lego blocks
c. Place 6 cups 175 around the holder starting from its initial position
d. Connect the color sensor to port S1

e. Run the function BP.set_motor_position_relative (PUSH_MOTOR, 150) on the robot followed by a sleep (0.1) then finally a BP.set_motor_position_relative (PUSH_MOTOR, -110) while the track is idle.

3. Film the output for reference and save it to your camera roll to document progression

4.0 Expected Results

The tests should reveal proper cube displacement on the track every time a cube is not being sorted without ruining future cubes that need to be sorted so that there is no disruption of the towers' structure or contents.

5.0 Format of Output Required

One phone recorded video will be included without the need for graphs since it is a visual observation one must make given that the sensors do not know the status of the robot such as if the tower is clobbered or if a cube was hit onto the conveyor belt.

6.0 Conclusions

The results were as expected because the arm was active only times, each instance hitting the cube on the first try without smudging the remaining cubes in the tower on all three attempts. As such the mechanism built is a reliable device, providing a 100% hit rate. The cubes were sorted as expected as detailed in our previous tests.

6.1 Further Action

We must do 2 things as a consequence of our results

a. Construct a taller tower that can house 20 cubes now that we know that our robotic arm is efficient at its task hence the final stage of our project is prepped to purse.
b. As an insurance policy, if the color sensor does not detect a cube within 4 seconds, the arm will be activated again to establish a safeguard in the moment a cube isn't pushed completely onto the track.

6.2 Distribution of Work

Three people will work on the increased tower height while the others will work on the lab report as a precursor to avoid wastage of time on unnecessary or rather unimportant work. The following test will be completed by Martin K along with Aman taking note of the procedure.

1.7: Integrated Test 7

Group: Main Project Group 7

Date: November 24th, 2021

Tester: Everyone

Author: Ali Therrien-Motamedi & Martin Kruchinski

Hardware version: 4.0

Software version: 4.0

Goal: Determine whether the new design using a higher cube tower and a second brickpi can correctly store colored cubes in their color coded location.

1.0 Purpose of Test

The purpose of this test is to assure that the new design will be capable of fulfilling its purpose. Since the new design is quite different from the previous implementation, it is quite possible that it can have issues and not correctly sort the colored cubes. Hence, this test is necessary to determine whether this new design is functional or not.

2.0 Test Objectives

Quite simply, the objectives include proper robotic arm displacement along with retraction, conveyor belt motion, color sensor activation, transporter functionality plus accurate rotary motion from the holder.

3.0 Test Procedure

The following test should be performed multiple times for each different colored cubes for better assurance of the functionality of the system:

- Fill the cube tower with differently colored cubes, as a stack
- Run the Python code for the system and wait until sensors and motors are ready
- Take note the color of the cube that is being analyzed by the color sensor
- Wait until the cube is dropped by the robot
- Take note of where the cube is dropped and whether it is the correct location
- Take note of any possible visible issues that occurred during this test

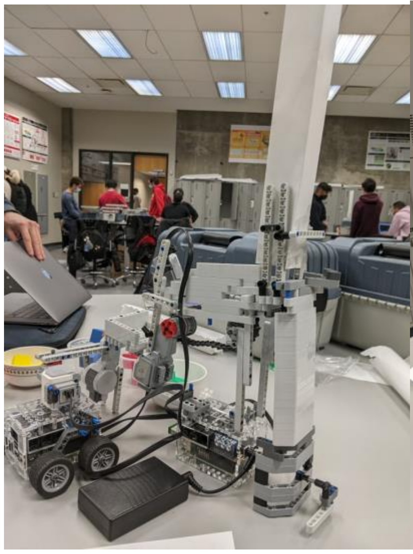
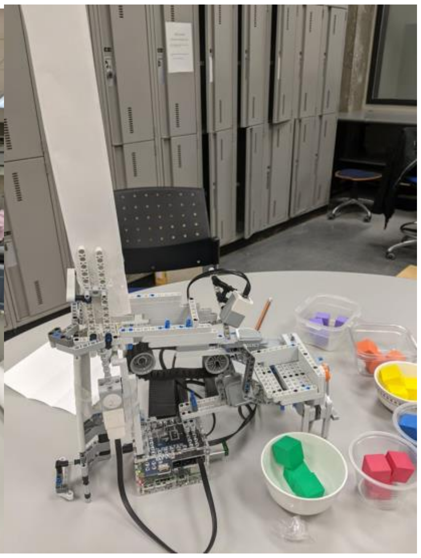
4.0 Expected Results

The system should be able to process and sort all the cubes in under 5 minutes. All the cubes should end in 6 distinct cups with the rest of their kind.

5.0 Format of Output Required

The output is not one which requires graphs, tables or some array given that it is a visual observation rather than a quantitative one. Thus, the results shall be illustrated via pictures to show the end product.

6.0 Results

6.0 Conclusions

The system was accurately able to sort all the cubes in their respective location. It was able to sort 10 cubes in 2 minutes and the rest of the cubes in under 5 minutes. All cubes are to be found in 6 different locations, where each location corresponds to the color of a cube.

6.1 Further Action

Aside from preparing our presentation for the demo, no further action is required as we have reached our final desired milestone.

6.2 Distribution of Work

Liamo, Martin and Seb will begin the presentation preparation for Friday while the others will continue working on both the final Design report as well as the project debrief.