DPM Practical Design Tutorial

by Ryan Au

Table of Contents

Section 1: Prepare and Plan for the Project

Final Project Overview

For the Final project, you are given a generally larger or more difficult problem to solve than in the Labs. Any knowledge you used in the Labs can be used in this project. They count as Research and Development.

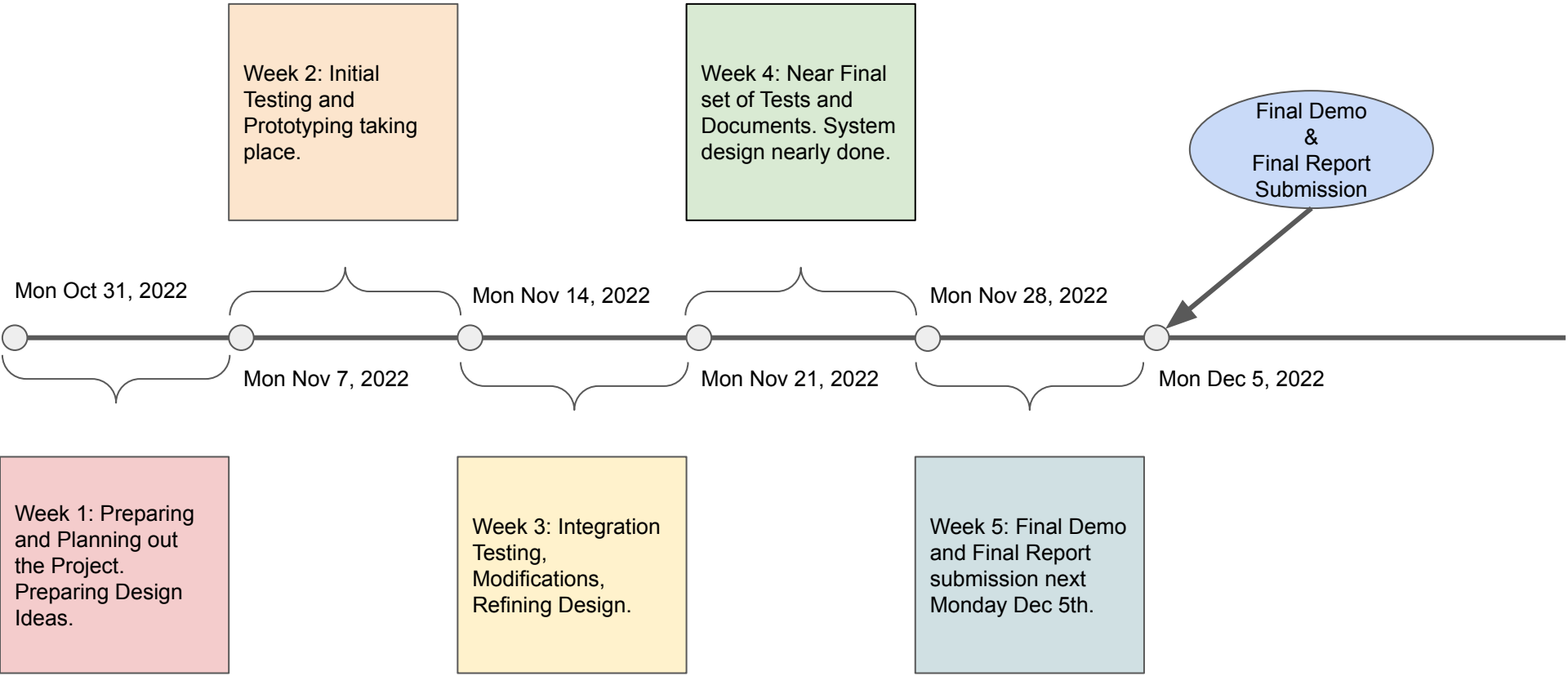
Keep in mind that the true focus of this project is **not to build a robot that does the task fully and successfully.** Your real purpose is to **properly plan and document the process of trying to build a robot that does the task fully and successfully.**

- Teams of 6
- Each person gets a main role in the project (e.g. Project Manager…)
- Documentation is updated weekly, and submitted weekly (probably Monday)
- You will have meetings with the Senior Engineer/Clients (TAs/Profs)
- You have a budget of time each week (9 hrs per person per week)

NOTE: These conditions may change depending on the semester

Final Project Overview - Fall 2022 Schedule

NOTE: This schedule may change depending on the semester.



Capabilities and Roles

We reassess how your team will be structured when it comes to your roles in the final project.

There are 5 main roles:

- Project Manager
- Document Manager
- Software Lead
- Hardware Lead
- Testing Lead
- +(some other person)

Project Manager:

- Budget Updates
- Project Coordination
- (If you are bad at organization, do not do this)

Document Manager:

- Responsible for the quality of the docs
- Ensures docs submissions are done properly
- (If you are bad at organization, do not do this)

Software Lead:

- Makes main decisions on the Software Impl.

Hardware Lead:

- Makes main decisions on the Hardware Impl.

Testing Lead:

- Ensures the quality of test procedures and data

Other Person:

In reality, every person on the team should be doing every task: Software, Docs, Hardware, Budget/Planning, Testing. This includes the 6th person. Each lead/manager is the main person responsible for their category of work, but everyone may work on it. The 6th can choose their name too.

Schedule Planning (Gantt Chart Overview)

For documenting the planning of tasks for your project, we use Gantt Charts as the end result.

I say end result because you can use any other form of documents to roughly plan out tasks, but then use Gantt Chart to show all of these tasks cohesively together.

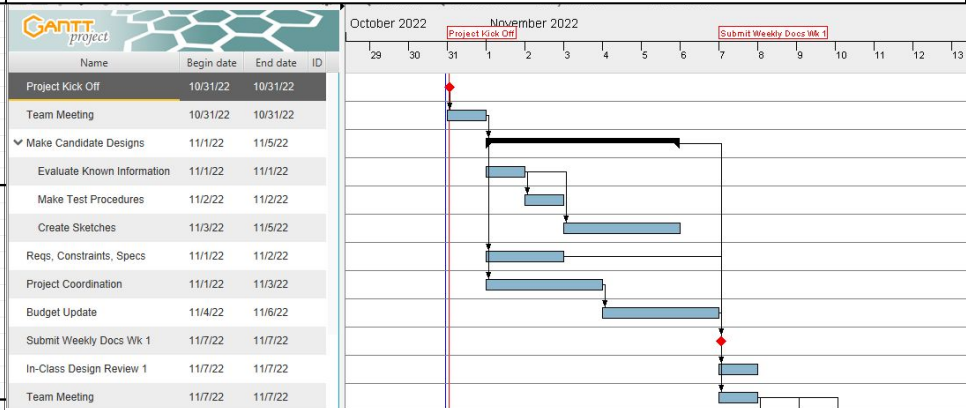
Gantt Charts aren't easy to share with your teammates, so utilizing some other means of tracking your tasks might help you out.

Get all members of your group to help you estimate the time it takes to perform each task. Get them to help you assign hours of each task to a person.

As an example, we use Google Sheets (lightweight collaborative) to create a rough plan for the tasks of the week.

		Project Mgr	Document Mgr	Hardware Mgr	Testing Mgr	Software Mgr	Other Guy	Total Hours	Start Date	End Date
Descriptions	Task Name	11	11	11	11	11	11	66		
Milestone for start of project	Project Kick Off	0.5	0.5	0.5	0.5	0.5	0.5	3	Oct 31st, 2022	Oct 31st, 2022
Understand the Project and Brainstorm	Team Meeting	2	2	2	2	2	2	12		
	Make Candidate Designs		2	5.5	2.5	2	2	14		
	| Make Test Procedures									
	| Evaluate Known Information									
	Reqs, Constraints, Specs				3	3.5	3.5	10		
	Budget Update	3	1					4		
	Project Coordination	3.5	3.5	1	1	1	1	11		
	Submit Weekly Docs Wk 1							0	Nov 7th, 2022	Nov 7th, 2022
	In-Class Design Review 1							0	Nov 7th, 2022	Nov 7th, 2022
	Team Meeting									
	Perform Preliminary Tests							0		
	| Test 1							0		
	| Test 2							0		
	| Test 3							0		
	Outline Prototype							0		
	| Hardware							0		
	| Software							0		
	| Test Procedures for System							0		
	Project/Budget Update							0		
	Senior Engineer Meet 1	0.5	0.5	0.5	0.5	0.5	0.5	3	Nov 9th, 2022	Nov 9th, 2022
	Submit Weekly Docs Wk 2							0	Nov 14th, 2022	Nov 14th, 2022

Then we create the Gantt Chart based on this. Everything should be accurate on the chart, but the spreadsheet doesn't need to be



Schedule Planning (Gantt Chart Advice)

Make the ENTIRE Gantt Chart in Week 1! This is to say that you have created a preliminary plan for the entire project, and you can fully use the budget. We expect this to be done.

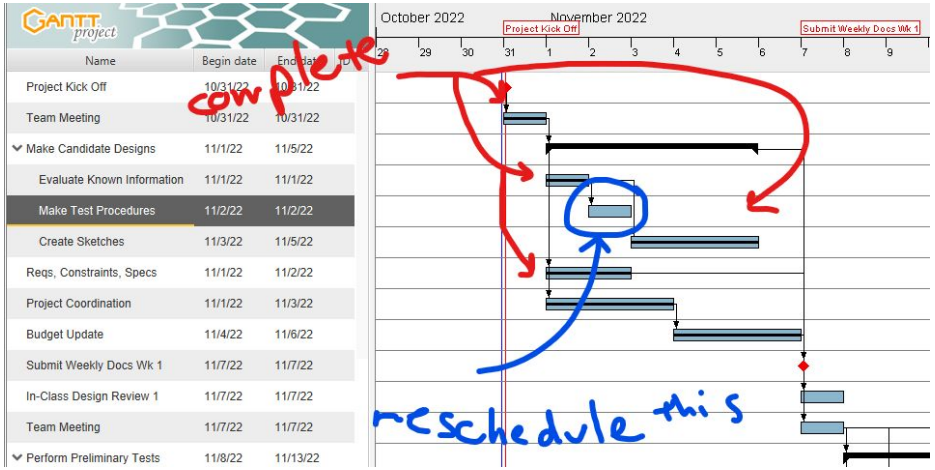
Long tasks (+4 days long) should be turned into a group and broken up into subtasks. A single task or subtask should take 1-3 days.

Planning a Gantt Chart can actually take several hours. Especially when you are doing this alone. Work with your teammates and plan accordingly.

(You also don't have to use Gantt Project software, but it's one of the only free versions that have just enough features for us to use and we know how to use it)

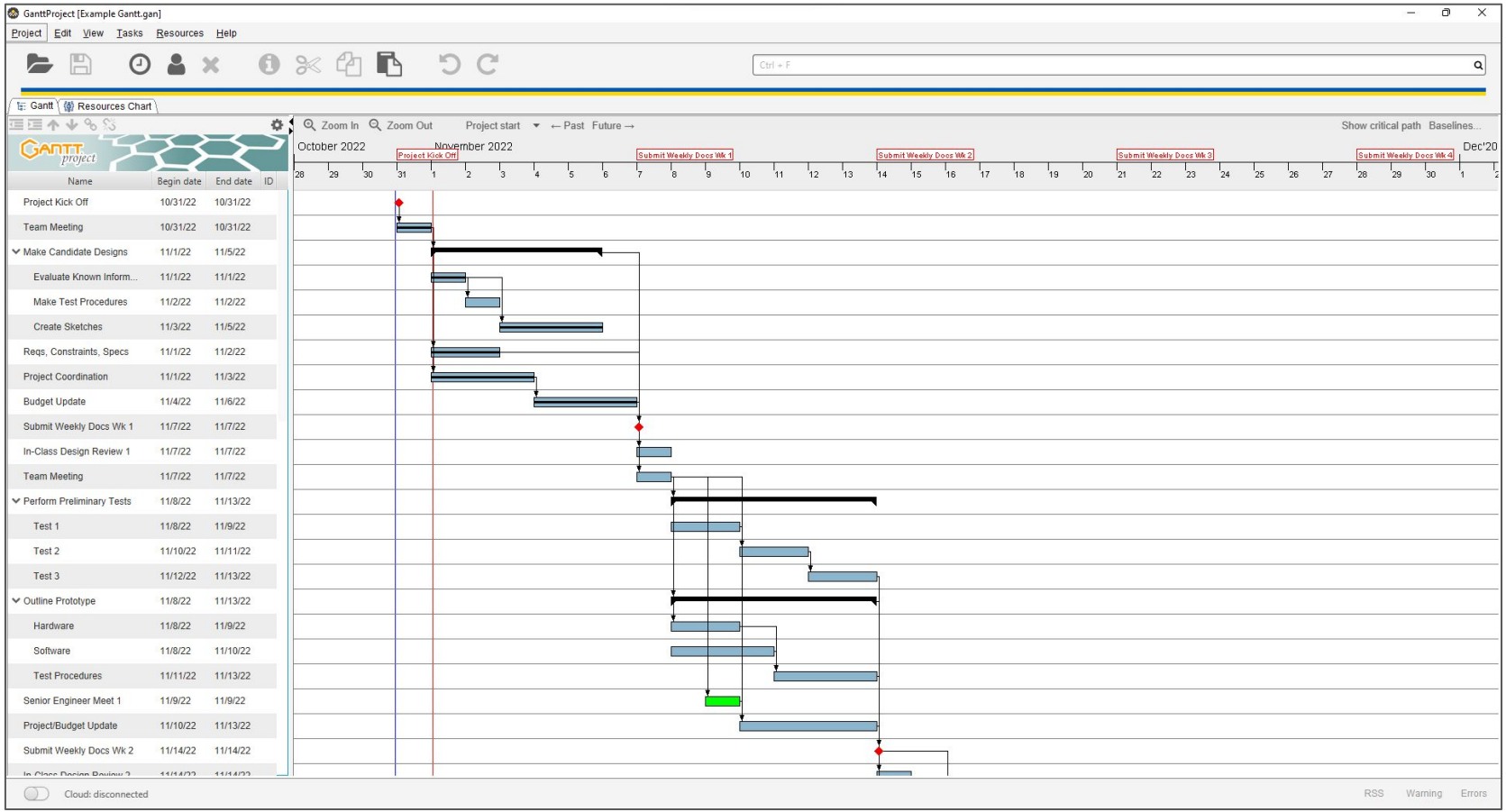
Don't spend all your time on this course! You have other classes! You can save budget hours from one week and use them in others.

For every week's submission, you will give an updated Gantt Chart.
(1) Rearrange and fix all of the tasks of the previous week to reflect the actual work that you did.
(2) Set the percentage of the work that you did for each task (should be 100% on each task that was completed).
(3) Any unfinished tasks should be rescheduled for the future
(4) ALL remaining budget hours should be assigned to future tasks.



Example of the Process to Make Gantt Chart

Note: Done in person/in the video



Understanding the Problem

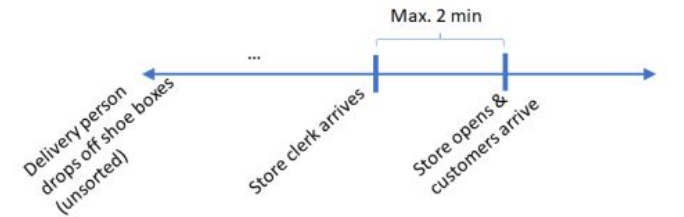
NOTE: This example is the problem for the Final Project of the Winter 2022 Semester.

Problem statement

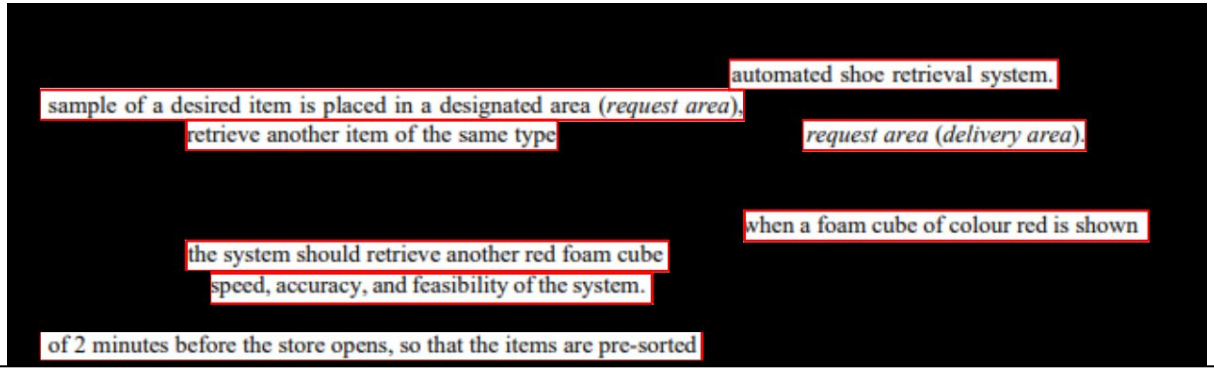
The executives at the client company wants to see a prototype of an automated shoe retrieval system. When a sample of a desired item is placed in a designated area (*request area*), they would like the designed system to automatically retrieve another item of the same type and deliver it nearby the *request area* (*delivery area*). **This way, the store clerk never needs to enter the storage area after the store opens.** While being able to sort a variety of items automatically is the ultimate goal of the client, this can prove challenging. Instead, the client wants a working system that can sort the foam cubes by colour. That is, when a foam cube of colour red is shown to the system, the system should retrieve another red foam cube from a collection of coloured cubes. The client's top priorities are speed, accuracy, and feasibility of the system. The store clerk can be instructed to trigger a sorting process (using simple interfaces such as pressing on a button, or placing something in a particular area) maximum of 2 minutes before the store opens, so that the items are pre-sorted before the first item retrieval request.

Problem Summary:

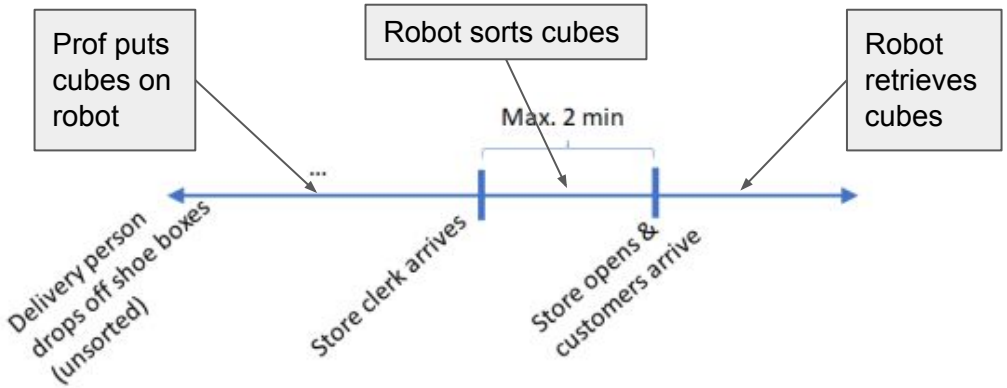
Automated Colored Cube Retrieval System. Given a sample of a desired item, the system returns another item of the same color, all foam cubes. The user can activate an automatic sorting phase of the system, that lasts a maximum of 2 minutes.



Understanding the Problem

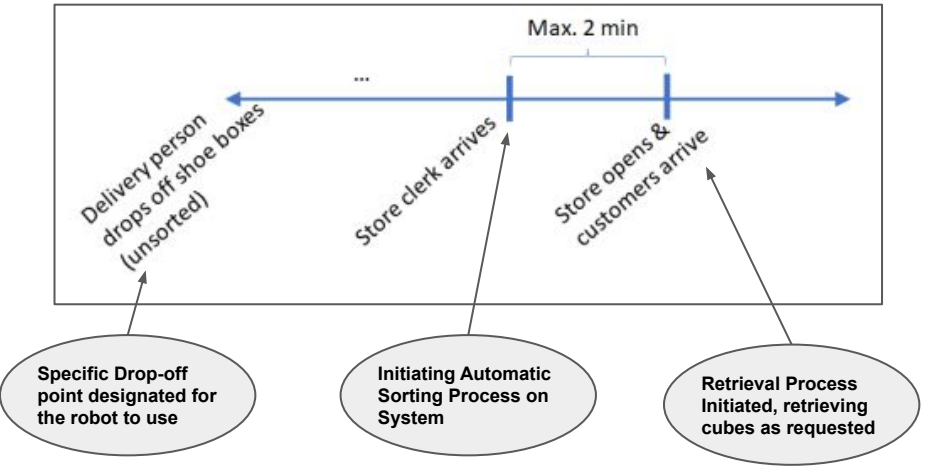


Problem Summary:
Automated Colored Cube Retrieval System. Given a sample of a desired item, the system returns another item of the same color, all foam cubes. The user can activate an automatic sorting phase of the system, that lasts a maximum of 2 minutes.



Understanding the Problem

Problem Summary:
Automated Colored Cube Retrieval System. Given a sample of a desired item, the system returns another item of the same color, all foam cubes. The user can activate an automatic sorting phase of the system, that lasts a maximum of 2 minutes.



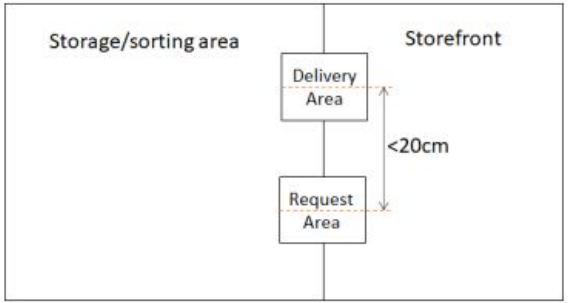


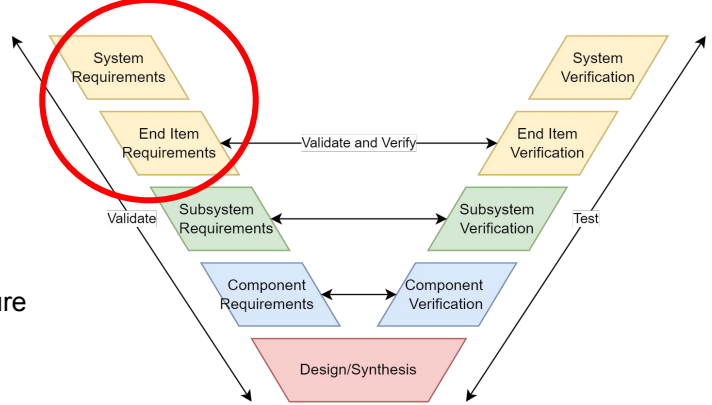
Figure 3. A sample floor plan.

The storage and storefront zones are undefined and left up to the design of the system.

The Delivery Area and Request Area must be clearly designated. They are <20cm apart.
These can be located on the robot, the floor, or any surface nearby.

System size constraint: <1m in all 3 dimensions.
Retrieval takes 5 seconds max.
All cubes delivered unsorted all at once.
6 foam cubes, 3 different colors. Assume Random Distribution.

Understanding the Problem
(Requirements Engineering)



Requirements - <u>These define the goals of the operation of your system.</u>

Requirements follow a specific wording format. We will give examples and overview of the structure that we expect so that you are all on the same page.

Example: The system shall sort the cubes in less than 2 minutes

Constraints - <u>These define the limitations of your project. These are things that you must use in the implementation.</u>

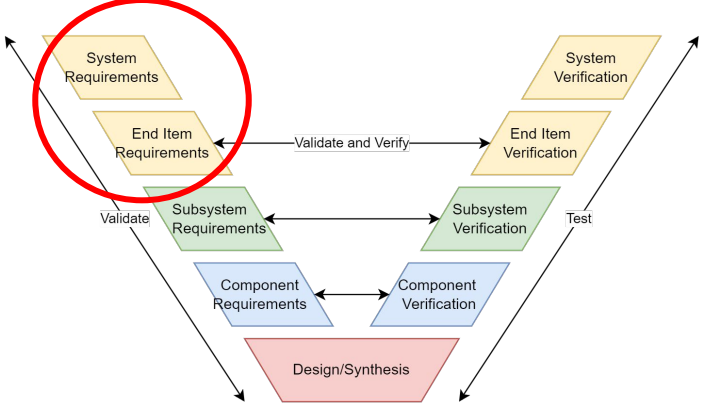
Constraints follow a specific wording format. We will give examples and overview of the structure that we expect so that you are all on the same page.

Example: The robot must be programmed using Python

Capabilities

This is just reassessing what your team is capable of. You should know this by now, but we'll review it for the final project anyways.

Requirements



A requirement follows the concept of your goals for the final system, things that you want it to be able to do.

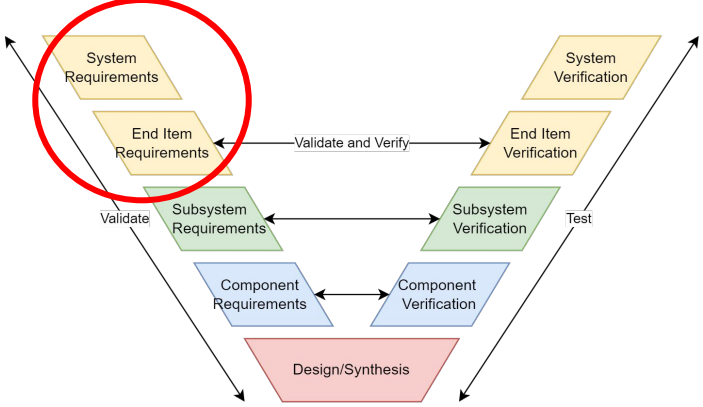
Non-related-to-DPM examples:

- This building shall be handicap-accessible
- This car shall brake when the brake is pressed down
- The system shall be able to lift weights of at least 3 metric tons
- The application shall display the user's profile picture upon startup

Each requirement has the feeling of the "system shall do". It is the product that you are working on that does the action.

One requirement, focuses on one thing at a time.

Constraints



A constraint follows the concept of your limitations, things that you are sort of forced to do or use in the development of your project.

Non-related-to-DPM examples:

- This architectural project has a budget of $5 billion
- This architectural project may go over-budget by $50 million
- The car must be developed within the span of a single year
- This project must utilize carbon fiber or carbon nanotubes
- This software cannot access 3rd-party resources on the internet

Each constraint has the feeling of "must do" or "cannot do" or generally something similar.

One constraint, focuses on one thing at a time.

For Our Example Project…

Problem Summary:
Automated Colored Cube Retrieval System. Given a sample of a desired item, the system returns another item of the same color, all foam cubes. The user can activate an automatic sorting phase of the system, that lasts a maximum of 2 minutes.

Requirements:

- The system shall sort the cubes within 2 minutes
- The system shall retrieve a cube within 5 seconds
- The system shall place a retrieved cube within 20cm of the requested cube location
- The system shall be smaller than 1m*1m*1m
- The system shall be capable of handling at least 3 different types of colors
- The system shall be capable of handling at least 6 cubes at one time

Constraints:

- The code must be written in Python
- There is no monetary budget for the project
- The project must be done within 6 weeks
- The electronics parts used to build the robot are limited to the contents of the BrickPi kit
- The budget hours is 9 hours per person per week

Section 2: Creating Ideas for the Project

Research and Understand your Equipment

By now, you will have done Labs 1, 2, and 3. You are used to working with and programming on the robot, but we will overview the usage of Motors and some Sensors so that we are all on the same page.

These Labs count as **Research and Development (R&D).**

You can refer to test/data/information you gathered from previous labs in your final project.

Touch Sensor

TouchSensor class

There is a push button in your robot kit. This class is for that button.

Tells you True/False if the button being pushed or not.

```
from utils.brick import wait_ready_sensors, TouchSensor

touch = TouchSensor(1)
touch2 = TouchSensor(2)

wait_ready_sensors() # init all sensors

touch.get_raw_value() # => 0 or 1
touch.is_pressed()    # => False or True

touch2.is_pressed() # => returns bool on other sensor
```

Ultrasonic Sensor

EV3UltrasonicSensor class

Primary usage is to collect the distance detected from the specified Ultrasonic Sensor.

Secondary usage is to detect other Ultrasonic sensors that are on and reading distances.

```
from utils.brick import wait_ready_sensors, EV3UltrasonicSensor

ultra = EV3UltrasonicSensor(2)

wait_ready_sensors()

ultra.get_raw_value() # => starts with centimeter reading

ultra.detects_other_us_sensor() # switch mode, False or True
ultra.get_cm() # switch mode, centimeter distance
ultra.get_inches() # switch mode, inches distance
ultra.get_inches() # no mode switch, it's unnecessary
```

Ultrasonic Sensor

Typically, the ideal specifications of the sensors differ from reality. You'll have to test these actual capabilities for yourself, because every piece of equipment is slightly different.

This is called characterization. If you don't already know, experiment, and figure out the capabilities of the sensors you have…

Ideally reads 0 to 255cm (100in)
*Actually reads 5cm to ~150cm

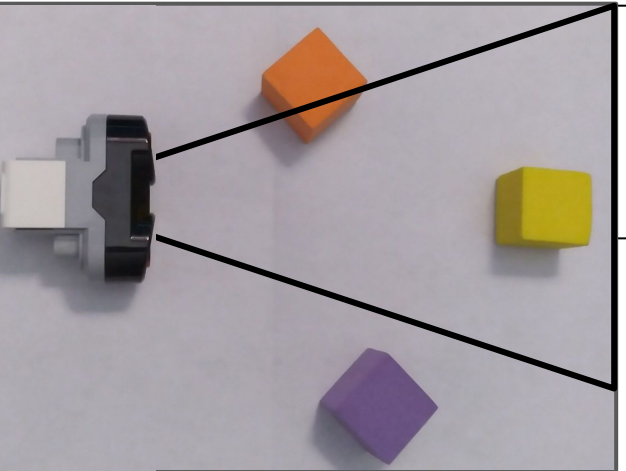
Ideally reads 10cm when object is 10cm away from the face of the sensor.

*May actually read 12cm when object is 10cm away from the face of the sensor.

Assume speed of sound is 343m/s. Ideal response time for a round trip (2x) of a single sound wave:

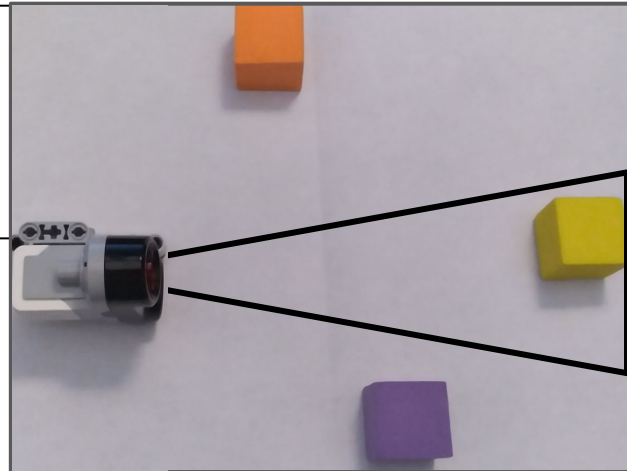
response time =
2 * (dist cm) / 34300 (seconds)

max response time =
2 * 255.0 / 34300 = 255 / 171.5 = 0.01487s



For FOV, it is narrower when the sensor is on its side, wider when it is flat, as can be seen in the following illustrations.

(FOV shown are inaccurate, test this yourself)



Color Sensor

EV3ColorSensor class

Detects RGB light values. Has its own light bulbs (for each color) Lightbulbs can be on or off, and there is a mode for Red Only detection.

```
from utils.brick import wait_ready_sensors, EV3ColorSensor

color = EV3ColorSensor(3)

wait_ready_sensors()

color.get_raw_value() # => returns raw value of current mode.
color.get_ambient() # => 1 float value. detects ambient light. lightbulb stays off
color.get_red() # => 1 float value. detects red light. red light turns on.
color.get_color_name() # => Use color sensor's color detection, return a string name of color
color.get_rgb() # => Returns a list of floats: [Red, Green, Blue] (excludes unknown 4th value)
color.get_raw_value() # => raw rgb value, includes "unknown 4th value"
```

Color Sensor

EV3ColorSensor Color Detection
We give an example of a method for color detection of learned color samples.

(1) Determining the Profile of a Color

Start the color sensor, and place an object in front of the sensor.

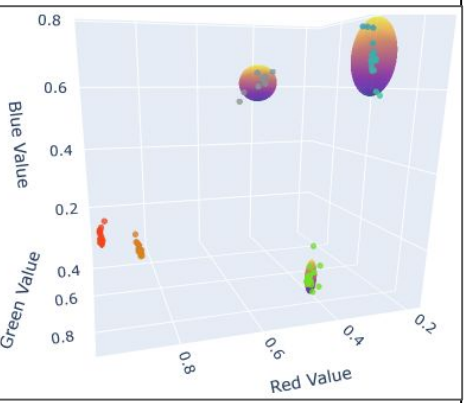
Collect multiple RGB samples of the object. Rotate/Shift the object in all potential orientations that you expect the object to be in during normal detection.

Red	Green	Blue
0.2214170196	0.6895558365	0.6895558365
0.2403091117	0.7150660907	0.6564541239
0.226433019	0.6918786407	0.6855888232
0.2077342612	0.6631516901	0.7190801851
0.1950355903	0.6111115392	0.7671400167
0.2068463555	0.6057643592	0.7682864871
0.2351960739	0.7055882279	0.6684519874
0.2430934595	0.7233512226	0.6462728361

(2) Recording Data

For each RGB, treat it as a 3D vector and normalize it.
Find the mean of normalized R values, then G values, and B values. Find the standard deviations. Record these (3+3 values in total), it is all you need.

```
>>> mean = [mean(r), mean(g), mean(b)]
>>> stdev = [stdev(r), stdev(g), stdev(b)]
>>> print(mean)
[0.2220081113375, 0.6756834508625, 0.700103787]
>>> print(stdev)
[0.017462239277766977, 0.045336027792894494, 0.04725264102293032]
>>>
```



(3) Using the Data

The 3 means of RGB form a point in 3D space. When we get a new sample RGB, we check if it is **close enough** to our existing point in space. If it is, we say it matches that color.

We can use the following formula to calculate the distance (in units of standard deviation) for new color

If the (std_dist <= 2), then we can say it matches the color. Change this threshold as you need for your usage.

```
mR, mG, mB = mean
sR, sG, sB = stdev
r, g, b = newSample

diffR = (mR-r)/sR
diffG = (mG-g)/sG
diffB = (mB-b)/sB

std_dist = sqrt(diffR**2 + diffG**2 + diffB**2)
```

Motors

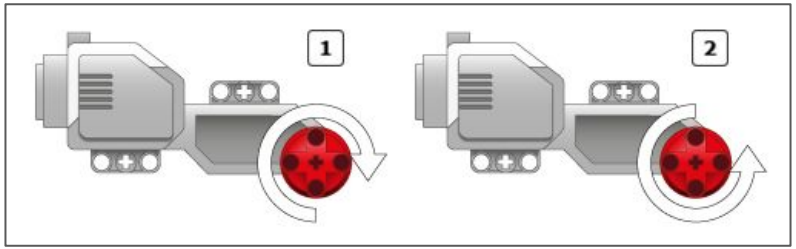
Motor class
The one class for all the motors. NXT, EV3, and Medium.

Each motor rotates based on power (0-100%) or “position” (degrees of rotation)

Each motor also has an encoder to track total degrees rotated. You can read this value.

```python
from utils.brick import Motor
import time

"""Power-based Control:"""

motor_left = Motor("A")

motor_left.set_power(50) # power 50%

time.sleep(1) # move for 1 sec

motor_left.set_power(0) # always do 0% to stop motor

# float_motor lets it move freely. 0% power resists movement.
motor_left.float_motor()

"""Position-based Control:"""

# Set target speed first, 360 deg/sec
# Reset power limit to limitless with 0,
#     default values:(power=0, dps=0)
motor_left.set_limits(dps=360)

# set current position to absolute pos 0deg
motor_left.reset_encoder()

# command to move to absolute pos 270deg
motor_left.set_position(270)

time.sleep(2) # gotta wait for the rotation to finish

# command to rotate 90deg away from current position
motor_left.set_position_relative(90)

while abs(motor_left.get_speed()) > 1:
    pass # Pause until the motor is slower than 1 deg/sec
```

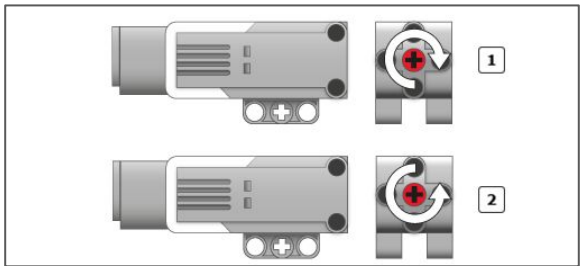
Motors



Referring to the motor's orientations in the following diagrams,
(1) Clockwise is the Positive Direction, and
(2) Counter-clockwise is the Negative Direction.

This applies to all methods:

- set_power(+/- power_to_rotate_with)
- set_dps(+/- speed_to_rotate_at)
- set_position(+/- position_relative_to_zero)
- set_position_relative(+/- degrees_to_travel)




```python
from utils.brick import Motor
import time

"""Power-based Control:"""

motor_left = Motor("A")

motor_left.set_power(50) # power 50%

time.sleep(1) # move for 1 sec

motor_left.set_power(0) # always do 0% to stop motor

# float_motor lets it move freely. 0% power resists movement.
motor_left.float_motor()

"""Position-based Control:"""

# Set target speed first, 360 deg/sec
# Reset power limit to limitless with 0,
#     default values:(power=0, dps=0)
motor_left.set_limits(dps=360)

# set current position to absolute pos 0deg
motor_left.reset_encoder()

# command to move to absolute pos 270deg
motor_left.set_position(270)

time.sleep(2) # gotta wait for the rotation to finish

# command to rotate 90deg away from current position
motor_left.set_position_relative(90)

while abs(motor_left.get_speed()) > 1:
    pass # Pause until the motor is slower than 1 deg/sec
```

Creating Design Ideas

We have assessed enough prior knowledge of the Robot and its hardware, that we can now begin creating ideas for the Project solution.

We still are using the example from Winter 2022, the Color Cube Sorting and Delivery system.

Brainstorming Ideas

Splitting up Sorting and Retrieval can help you think of solutions! 👍

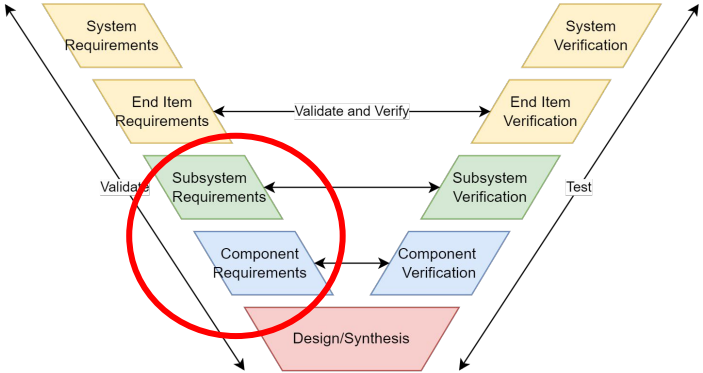
Sorting Ideas

Storing the Cubes!

(1) Conveyor Belt
(2) Silo/Chimney for Cubes
(3) Wide Flat Tray
(4) Leave it on the Floor
(5) *(could be a lot of things)

Sorting

(1) Put cubes of a certain color into a box/bin
(2) Push the cubes around into groups on a table
(3) Scan the cubes and remember where each color is



Retrieval Ideas

Retrieving Cubes

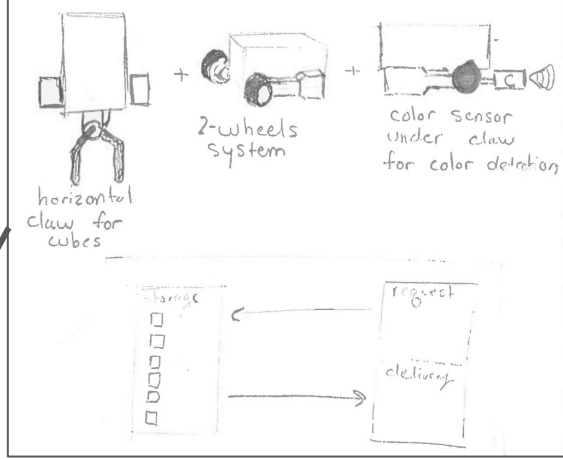
(1) *Don't sort at all, and just go search for a cube of the matching color
(2) (retrieve a cube based on the sorting style)

Grab a cube from one of the boxes

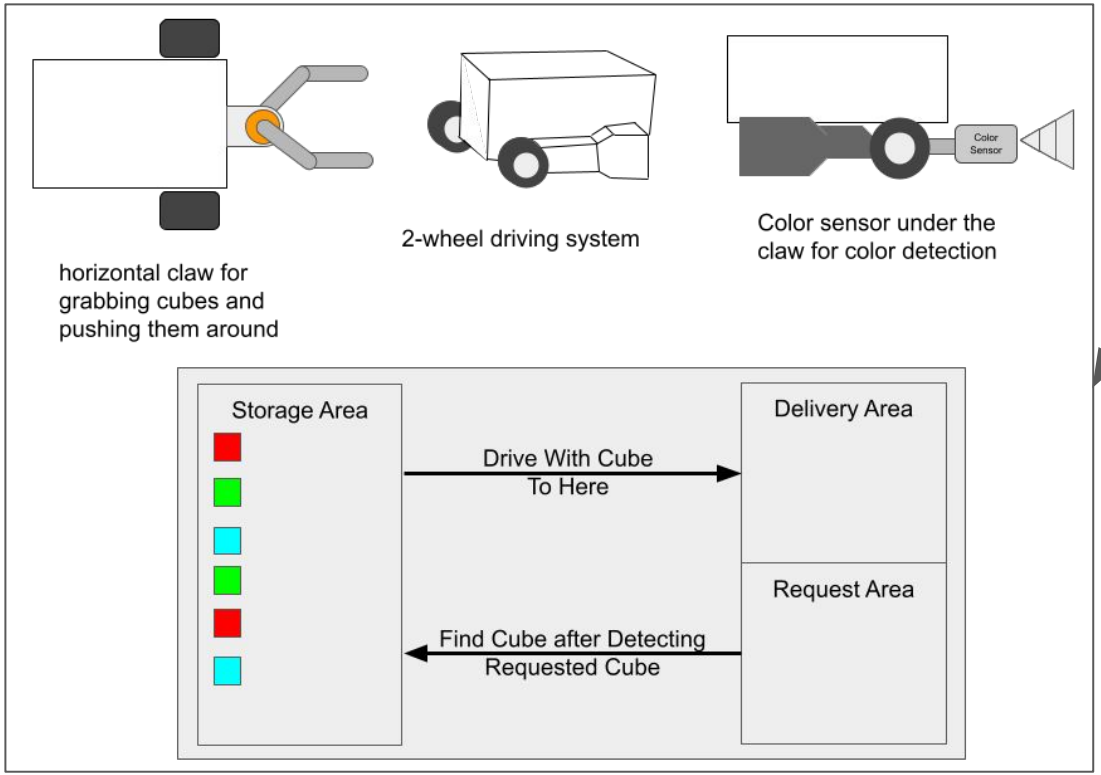
Push a cube from a group, to the client

Get a cube from a remembered location

Potential Idea #1 - Cube Grabber



Rough Draft

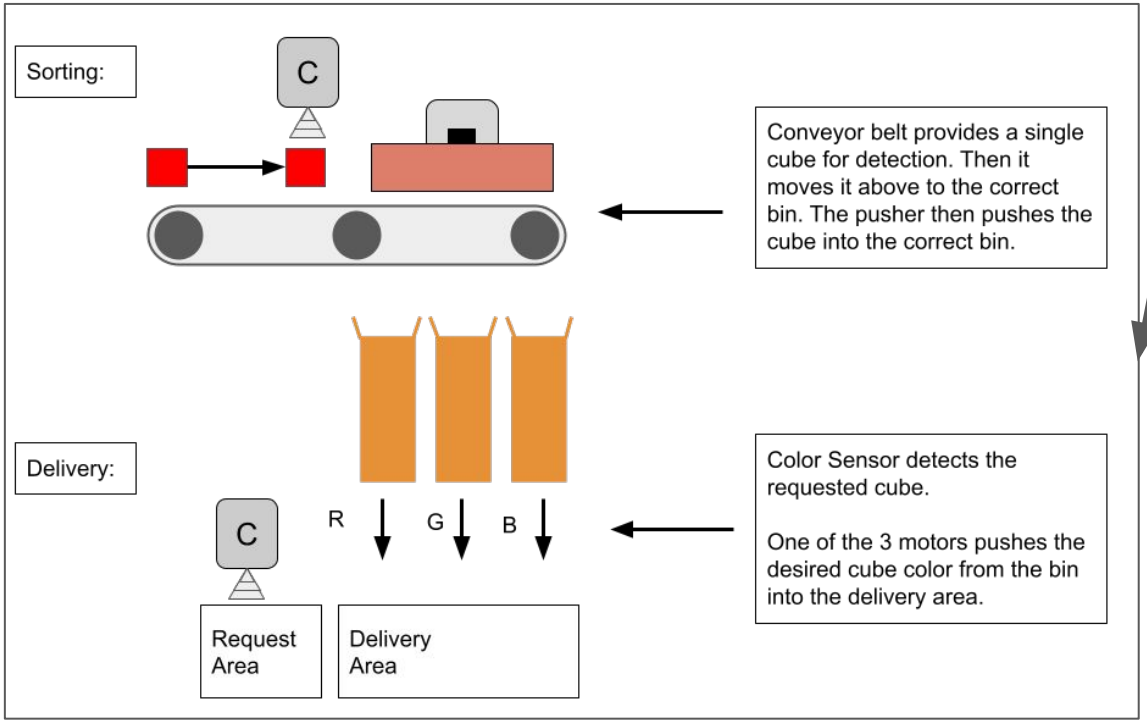


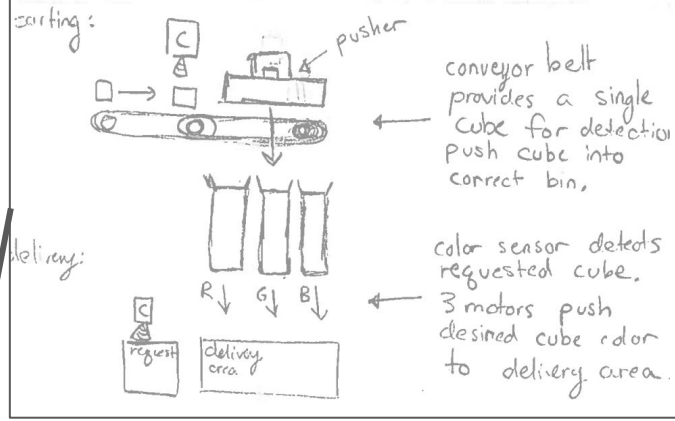
horizontal claw for grabbing cubes and pushing them around

2-wheel driving system

Color sensor under the claw for color detection

Potential Idea #2 - Conveyor+Bins

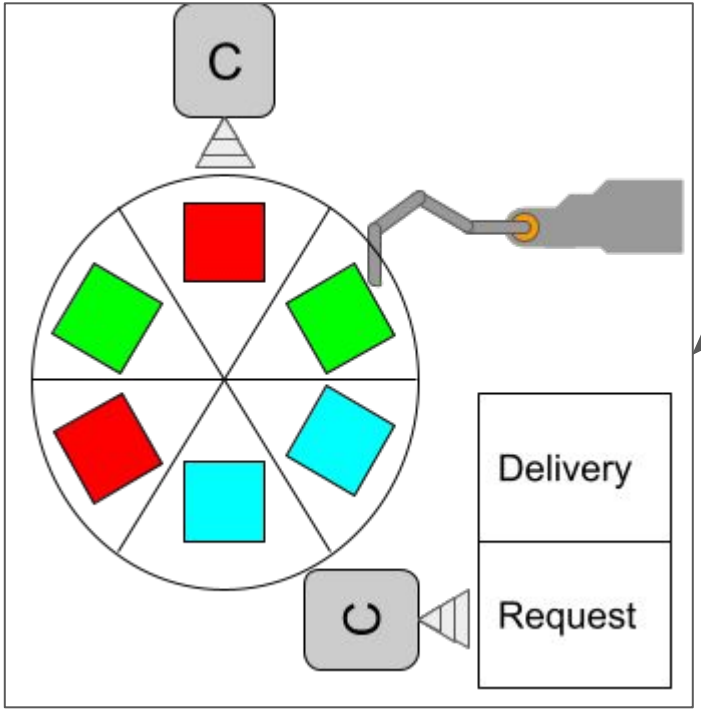


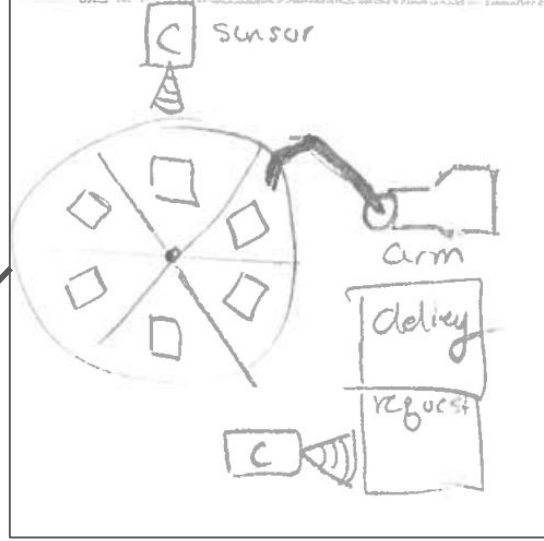


Rough Draft

Potential Idea #3 - Remember Disk

One color sensor remembers the color of each position on the rotating disk. The Robot arm allows us to push the corresponding cube into the delivery area upon receiving a request.



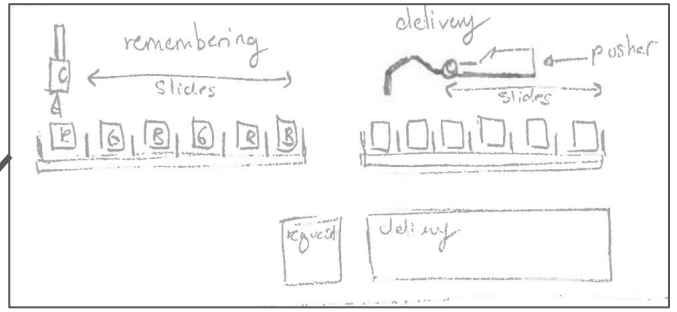


Rough Draft

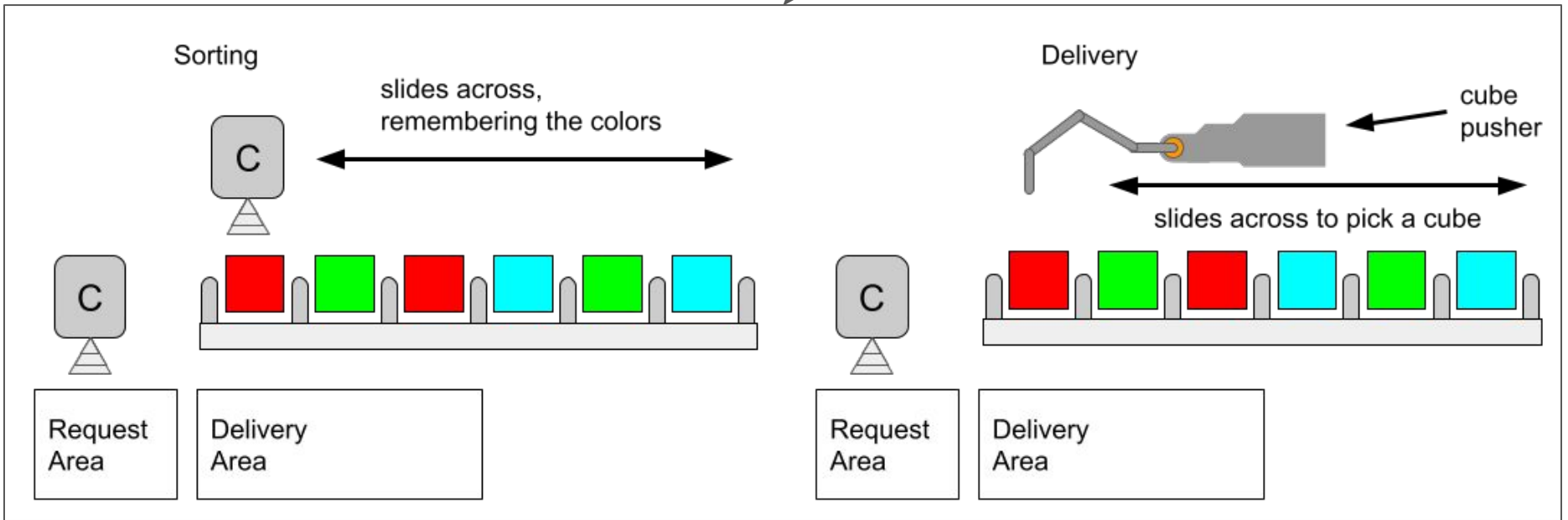
Potential Idea #4 - Remember Slide

One color sensor remembers the color of each position on the sliding track.

The Robot arm allows us to push the corresponding cube into the delivery area upon receiving a request.

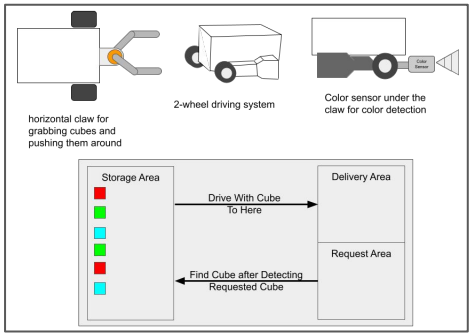


Rough Draft

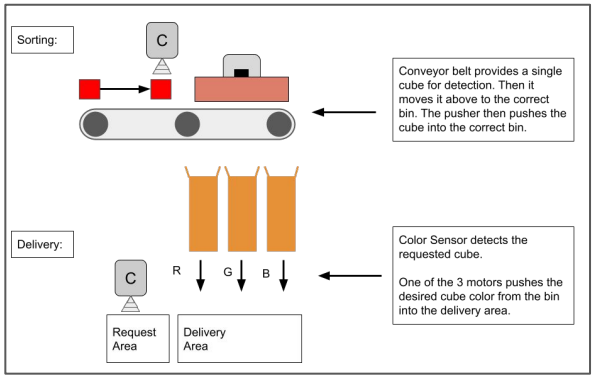


Which one would you decide to implement?

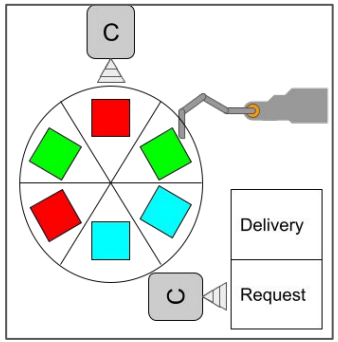
1



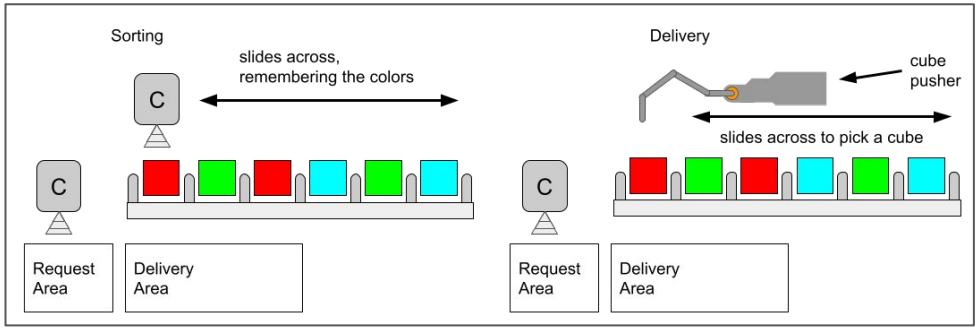
2



3

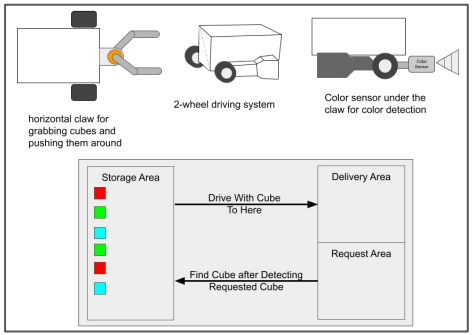


4



Evaluating the Designs

1



This is a very bad choice for our team

The main aspect of this build is actually the Driving System.

This robot needs to drive. This requires logic called Odometry and Localization to to properly navigate an area. There are many factors to account for, such as motor drift, that you are not familiar with.

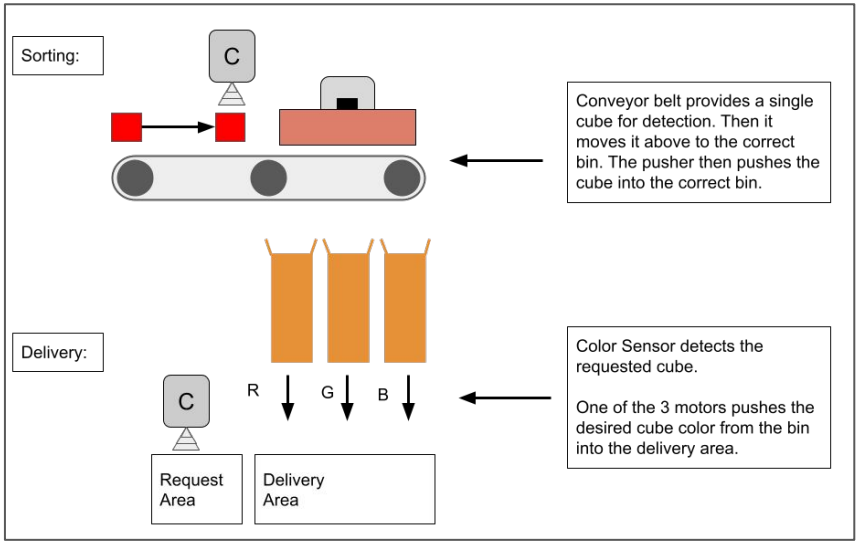
You know how self-driving cars are really complicated? Same thing.

You're making a "lego self-driving car" that also has to go and find cubes of various colors somehow, using a only a single distance sensor and a short-range color sensor. No cameras.

For most teams, the hardware is simple to construct, but the LOGIC is complex and intense to create for this simple robot. If you have experience with it, then it is doable, but you probably don't.

Evaluating the Designs

2



This is a decent choice for our team

This robot is a decent choice, and most operations are done by simple actions and logic, unlike Idea #1.

There is a problem, in that this design actually needs to use 5 motors, while the BrickPi only supports 4 motors at a time. You will need a 2nd BrickPi running a different program to make this.

Both subsystems are clearly defined and visible. And hardware implementations can easily add safeguards to account for various types of problems like cubes falling into the wrong spots.

Evaluating the Designs

These two robots do the same thing, and
either one is a good choice for our team.

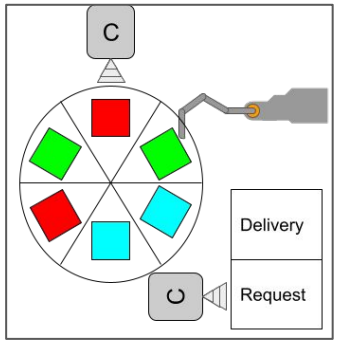
There are fewer moving parts (i.e. conveyors/cubes moving around) such that during operation, less problems may occur while processing the cubes.

Uses 2 motors, 2 color sensors, (some buttons opt.)
One BrickPi needed only

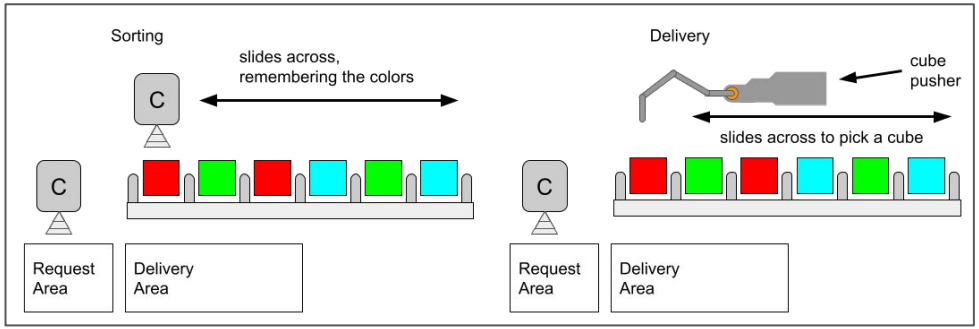
The sorting being done logically also means that the sorting process does not require cubes to physically move. In other words, sorting can be a fast process.

The slide/disk can also be extended or made larger to account for more cubes. (We only need to do 6.)

3



4



Choosing a System Design

Only the first Design (exactly as shown) was not made for the final project in the semester that this problem was given…

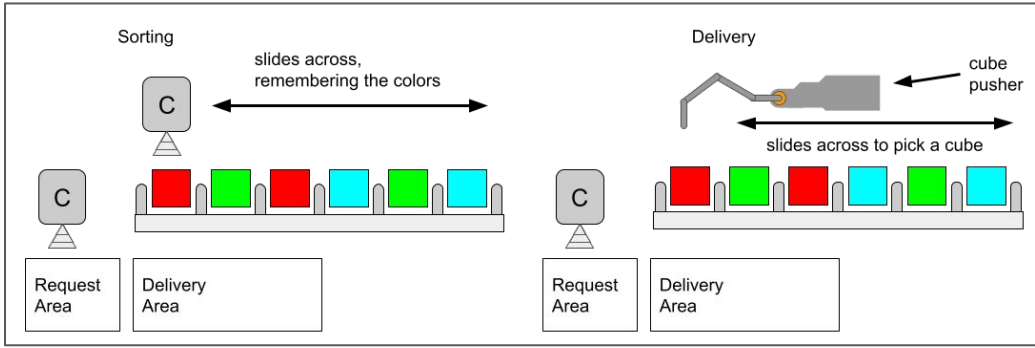
There will be many designs that you could think of, and many of them would work ***eventually***.

You need to consider your time constraints and the capabilities of the team. If your team is highly knowledgeable in self-driving cars, then perhaps Design 1 is doable for your team…but it isn't doable for our team (or many others).

For this Tutorial, we will use Design #4 because:

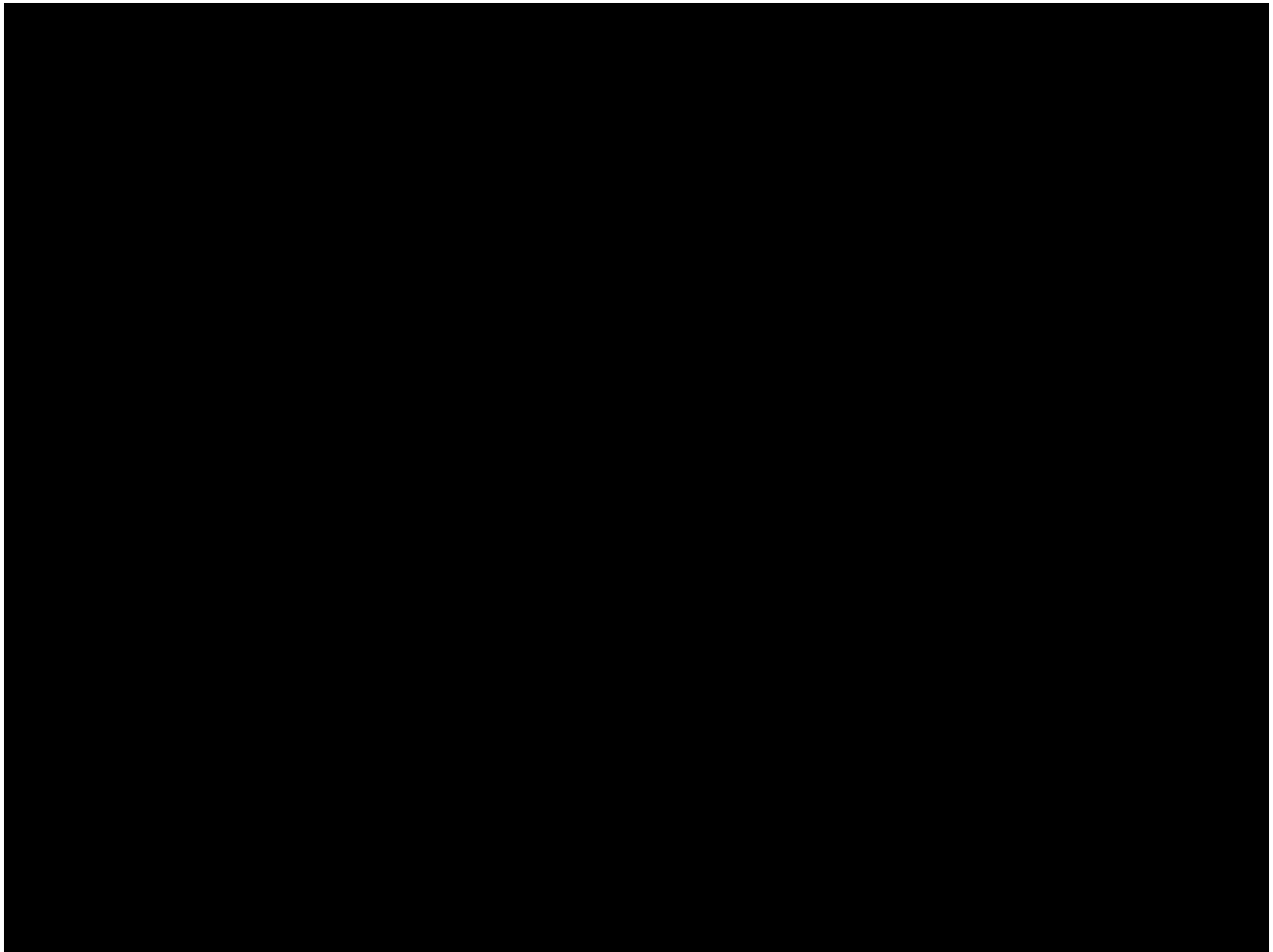
(1) Linear slides are slightly simpler than the disk
(2) It can use just legos. The disk is usually cardboard
(3) It uses 2 motors only, 1 BrickPi only
(4) I like it

These are all valid reasons to make a particular design, and your reasons could be similar too.

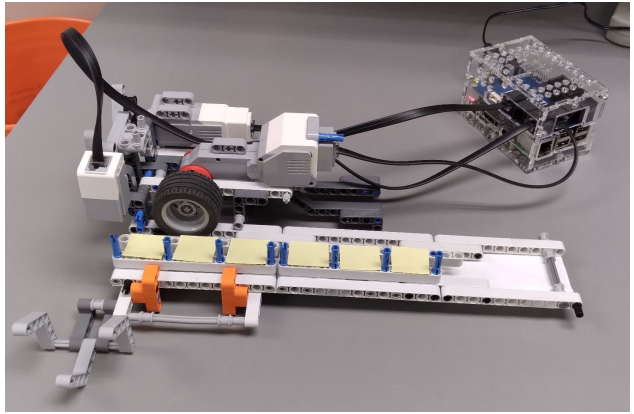


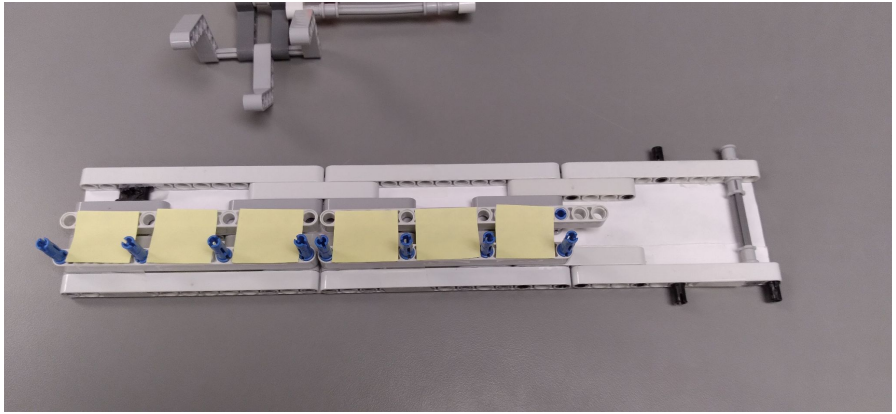
Demo of the Example Final Project, Working

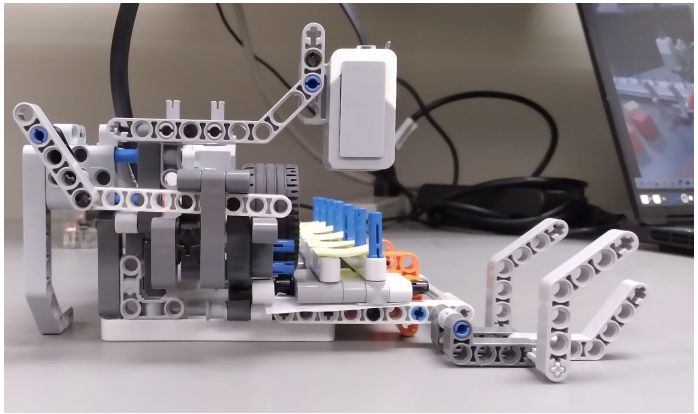
NOTE: Also posted on MyCourses.

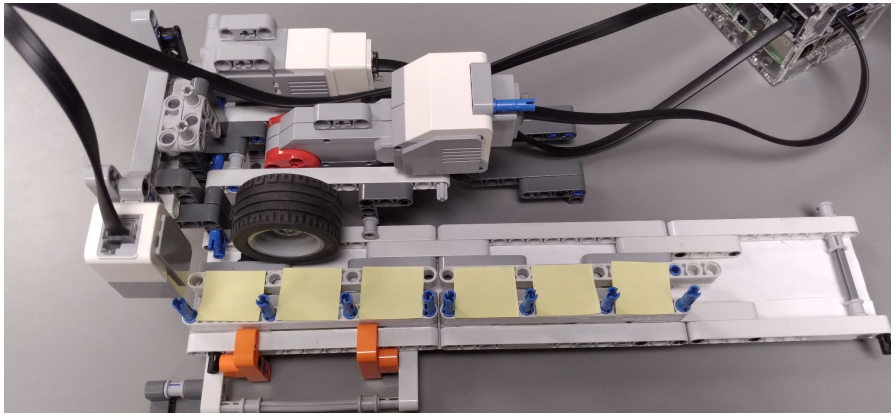


The Final System (in images)





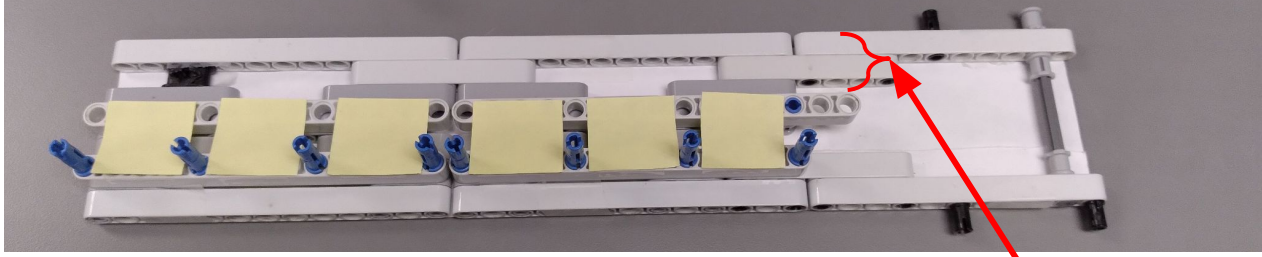




The First Goal

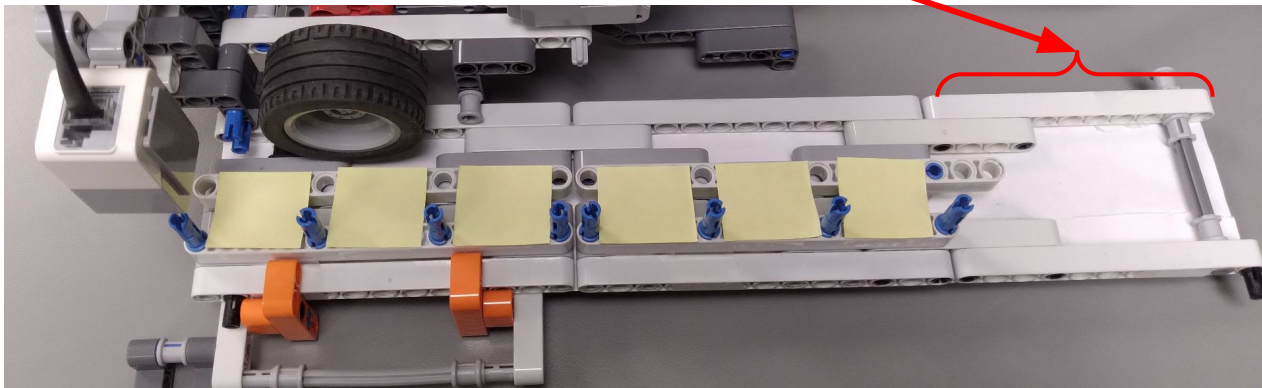
The track was the first thing to be prototyped. It is the basis of the robot.

Hardware tests determined that a cube will fit into a space of “3 Lego Circles”. Since it is a cube, we needed to reserve a space of 3x3 lego circles.



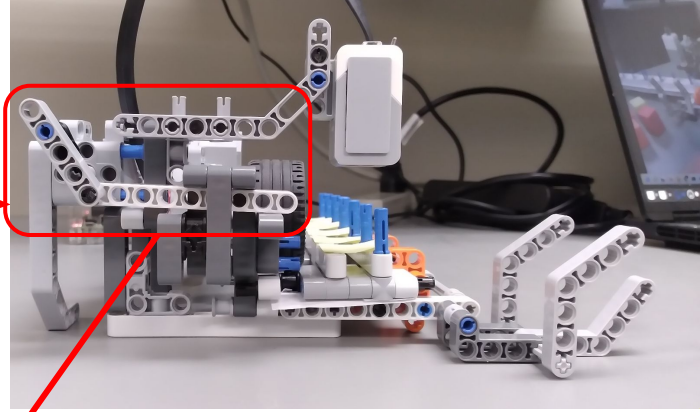
The track needed space for 6 cubes. Behind those six cubes is extra width for the tire to sit, and slide the track.

Testing revealed the need for extra length on the right-hand side so that the track can slide all the way to place the 6th cube slot under the color sensor, and the tire still contacts the track.

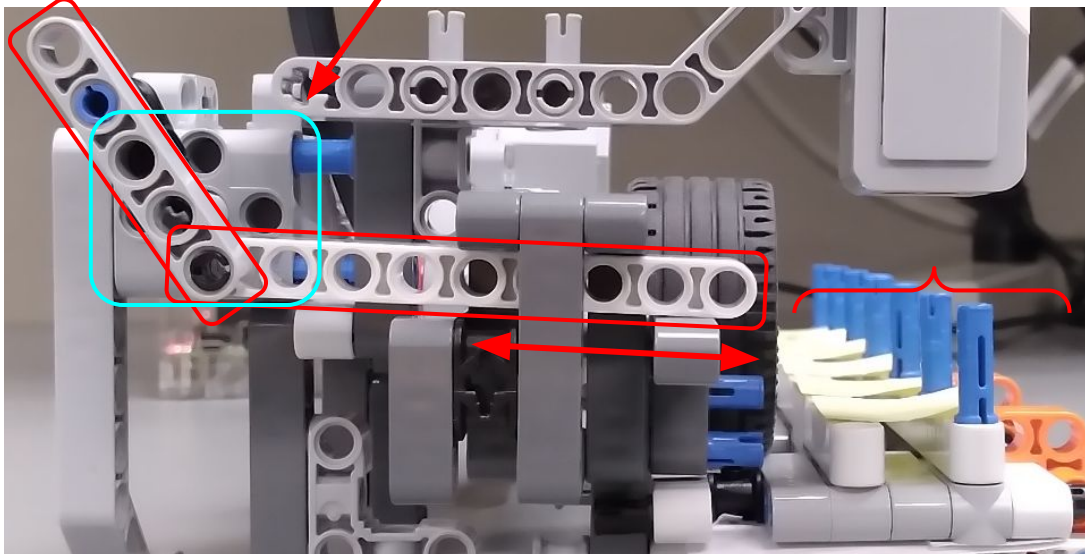


The Second Goal

The pusher arm is the next most important thing on this robot.



It is a piston, but put at a little diagonal angle to fit it into the design. It can only rotate Counter-Clockwise because of this.
It slides back and forth 4-6 Lego Circles.
Tested to ensure that it is just enough to push cubes off of the track.



It was tested specifically, if it could consistently push the correct cube off of the track.

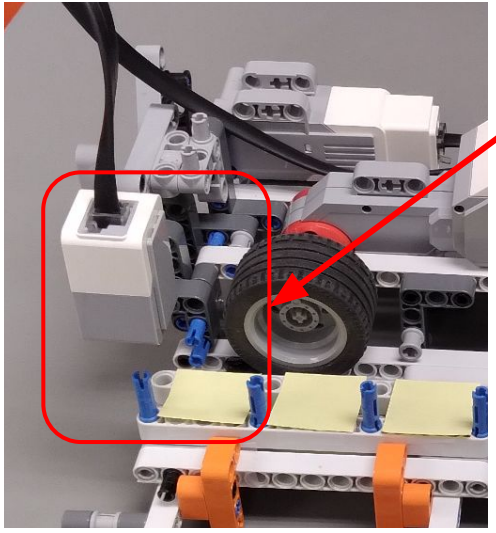
The error would come from the drift in the placement of the track. (source: motor drift)

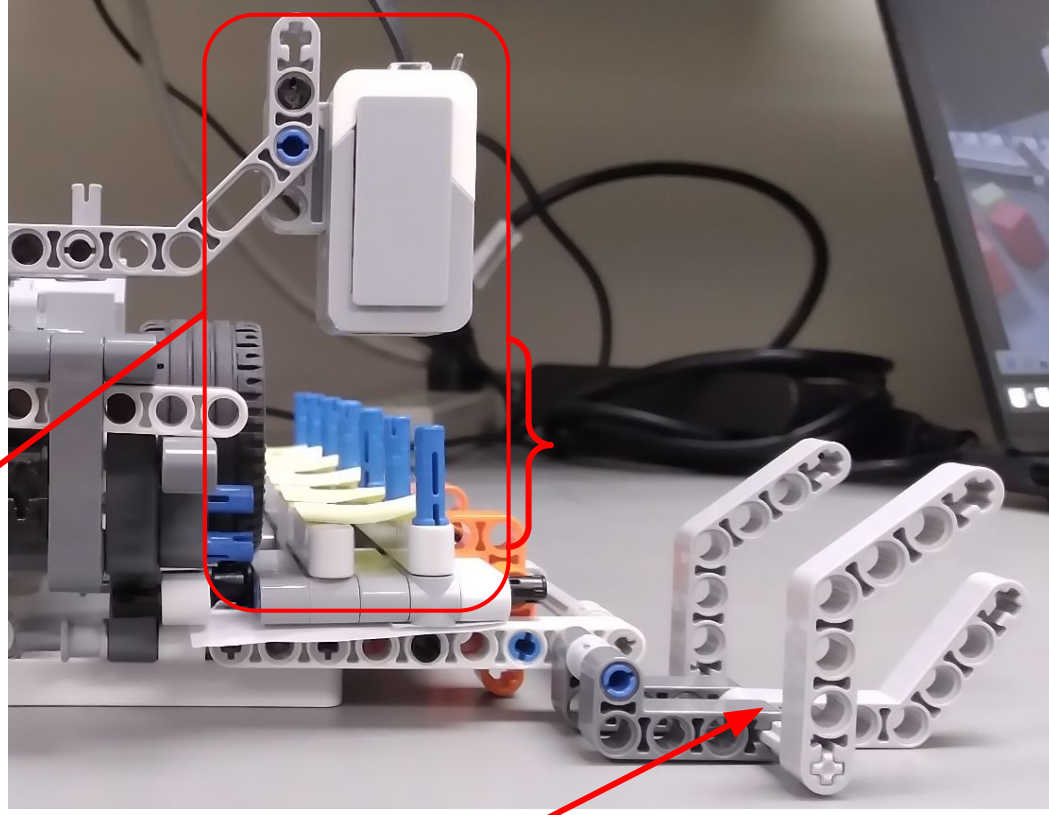
Testing revealed that a thinner pusher rod was more forgiving, when presented with any track alignment drift

The Third Goal

Placement of the color sensor is the final step.

It is placed at a height of approx 3.5 to 4 Lego Circles above the track, giving a bit of clearance to the cubes on the track.



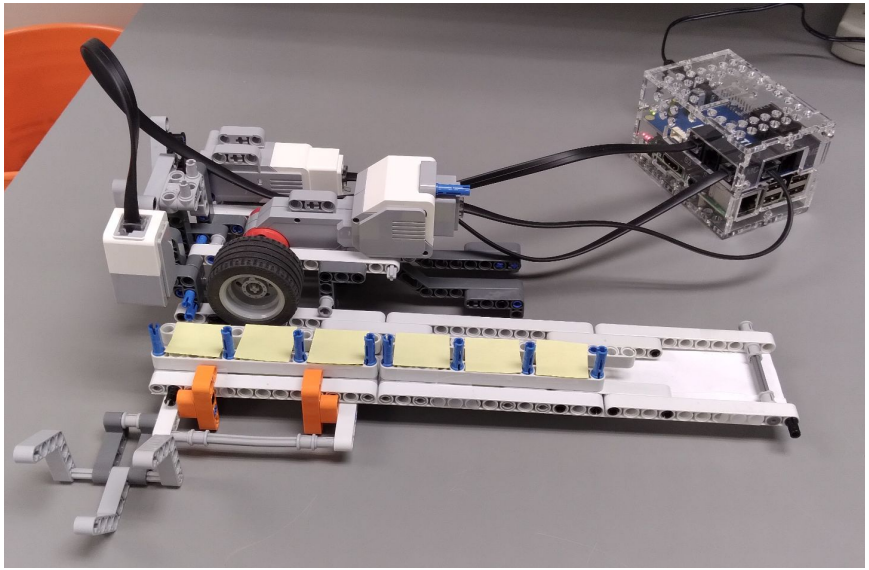


The cubes are pushed out of the track and into this holder. It can hold one cube at a time, and it was tested to make sure it does not interfere with the movement of the track.

The Final System (Conclusion)

The system was determined to be very fast, and very accurate at sorting and retrieving cubes. With less moving parts, there are less movements required to be made, therefore less time taken, and less error in sorting/retrieving.

There is a hard maximum limit of 6 cubes, but the system can be modified so that it can store more.



It can handle all 6 colors of cubes as well. The color detection algorithm that it utilizes factors out brightness through normalization, and converts all color distances to units of standard deviation, so that all measurements are then in the same 1:1 units.

The color detection algorithm is flexible enough that recalibration of the sensor for different lighting conditions can be done through the changing of a single value threshold for each color cube.

For example, in the video demo, calibration consisted of changing all thresholds to 7.5 maximum, and only green had a maximum threshold of 10 because it varied too much.

Section 3: Doing the Work During the Project

Meetings with the Senior Engineer (mentor TA) every week

At the end of every week, you will submit all of your documents that you created during the week.

The Mentor TA will then review your documents. Once they are done, you will have a meeting.

These are some things you should do for this meeting.

(1) **Prepare an Agenda** that includes the following:
  (a) Points to review about your documents
  (b) **Any and All Questions**
  (c) Overview of your future plans
(2) **Take Notes.**
(3) Your team should be presenting the work that you did.
(4) **Be honest.**
  (a) You're graded for completion and showing up
  (b) Telling the TA what is going wrong means that they can help you

Things to remember:

Share advice between teams in the class! Sometimes different teams get good advice that others don't. Help each other out!

Your TA will probably not know or remember everything, they are students like you, not really Senior Engineers…so sometimes your TA will have to check with other TAs for a correct answer on some topics.

Maintaining Documentation

Keep all your documents, and submit (to the Weekly Deliverable) whatever you create each week.

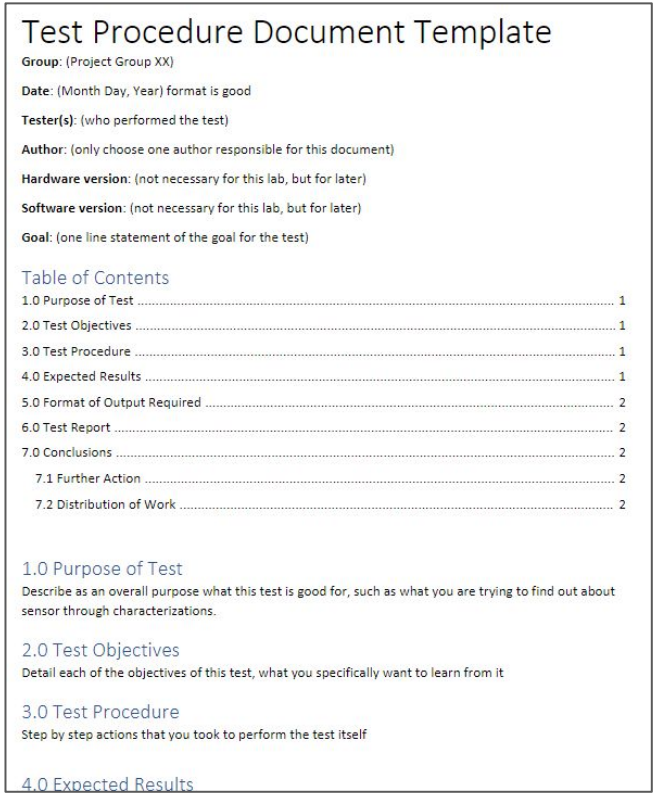
- Code
- Test Data
- Hardware/Software/Test Documents
- Project Planning
- Any Gantt Charts
- Any Timesheets

At the end of the project, you will condense and compile all of this relevant information into a single **Final Report.**

Use any style that you want! Documents during the project can contain large amounts of information, that is fine, and it is good.

It means you don't have to remember something in Week 3, that you forgot to write down in Week 1!

Using a format like this or similar for your team, means that you will have at least enough information for your final report. You won't be omitting any important details, and you might get full marks! Yay!



Test Procedure Document Template

Group: (Project Group XX)

Date: (Month Day, Year) format is good

Tester(s): (who performed the test)

Author: (only choose one author responsible for this document)

Hardware version: (not necessary for this lab, but for later)

Software version: (not necessary for this lab, but for later)

Goal: (one line statement of the goal for the test)

Table of Contents

1.0 Purpose of Test

Describe as an overall purpose what this test is good for, such as what you are trying to find out about sensor through characterizations.

2.0 Test Objectives

Detail each of the objectives of this test, what you specifically want to learn from it

3.0 Test Procedure

Step by step actions that you took to perform the test itself

4.0 Expected Results

Prototyping, Testing, Re-evaluating designs

When you test something. Write that down.

When you change something significant. Write that down. Draw it out.

When you change EVERYTHING. Write that down. Draw it out.

When you change the way the code works. Write that down too.

I know it sounds obvious, but when it comes time to write your Final Report, **you're going to realize that you don't have**:

(1) Dates/Times/Conclusions for Test Executions
(2) Descriptions of Major Design Decisions
(3) Sketches/Descriptions for each Design Iteration of every week
(4) Versions of your Software
(5) (Literally so many things that you overlooked because you were lazy and thought you'd write it down later)

Prototypes and Design Re-evaluation

Creating a Design implies that you will eventually have:

- Validation that it would satisfy the Requirements
- Advantages and Disadvantages
- Time Estimate to Build (for planning)
- Hardware Perspective
- Software Perspective
- Testing Perspective

Having these mean that you can feasibly create the Design for your project

If you **write this information down when** you make a design or re-evaluate one, then you **don't need to remember what you did later!**

Re-evaluating a design implies that you are answering:

- Why did this design fail?
- What did you change?
- Why did you change?
- What data showed that you need change?
- How does this affect your project/budget?
- Who is there to work on it?

Testing. Explained.

UNIT TESTS
These are the tests of individual **components** of any System.

- “Touch Sensors have low latency”
- “Ultrasonic Sensors are precise”
- “Color Sensors detect color consistently”
- “Motors move”
- “Speaker makes the correct noise”

Each Unit Test only tests **one functionality** of **one component.**

INTEGRATION TESTS
These are the tests of multiple **components combined together:**

- “Touch Sensors trigger the Speaker”
- “Motor moves according to Ultrasonic distance”
- “Color Sensor and Touch Sensor trigger the Speaker”
- “Motor moves according to Ultrasonic distance, and stops when a Touch Sensor is pressed against a wall”

Each Integration Test gradually tests **more and more components combined together.**

SYSTEM TESTS
These are the tests of your **entire System**. These tests combine every single component into a big test.

The reason why we have so many tests is because :
(1) We check that all components work “individually, with no outside interference”
(2) We check that components can work “together, in various combinations”
(3) We check that the whole System works altogether
Errors can occur at any point in time. Some because of a broken component, some because of weird interactions between components. That is why test in this manner.

Beta/Final Demo

Your semester may have a Beta Demo, or it may just have a Final Demo.

In either case, there will be one very important and specific description of how the demo will be run, and how it will be graded. Pay attention to every little detail.

We will break down the rubric and procedure, shown on the right.

Beta demo procedure:

- Team members should expect to briefly explain how the system works (or is supposed to work), and outline how the system is to be loaded. There will be no powerpoint/slide presentation (we will not have any projector set-up).
- One of the professors will place six cubes (two of each colour, with three colours of the team's choosing) as instructed by the team.
- The team will be allowed to trigger the system to start the sorting process (if any) to demonstrate the prototype in action.
- Within 2 minutes of having loaded the system or started the sorting process (whichever is later), one of the professors will place one cube (of one of the three colours) onto the request area, and trigger the retrieval process.

Grading rubric (5% of final grade): /10

- The system holds at least 6 cubes (Yes/No): __ (1 point)
- Request area clearly marked (Yes/No): __ (1 point)
- One cube retrieved upon request (Yes/No): __ (3 points)
- The colour of the retrieved cube matches the request (Yes/No): __ (3 points)
- Distance between request area and location of cube delivery is less than 20cm (Yes/No): __ (1 point)
- Time between cube request and delivery is <5 seconds (Yes/No): __ (1 point)

Beta/Final Demo

At first, we thought that the distribution of how many of each cube, given to the robot would be random.

In the beta demo, it turned out to be 2 of each color, 3 colors of choice. This greatly simplifies the problem for the project.

And some teams didn't have a clear sorting process. It was supposed to be separate, and it should run first before the retrieval process.

6 cubes! 2 of each color!

Beta demo procedure:

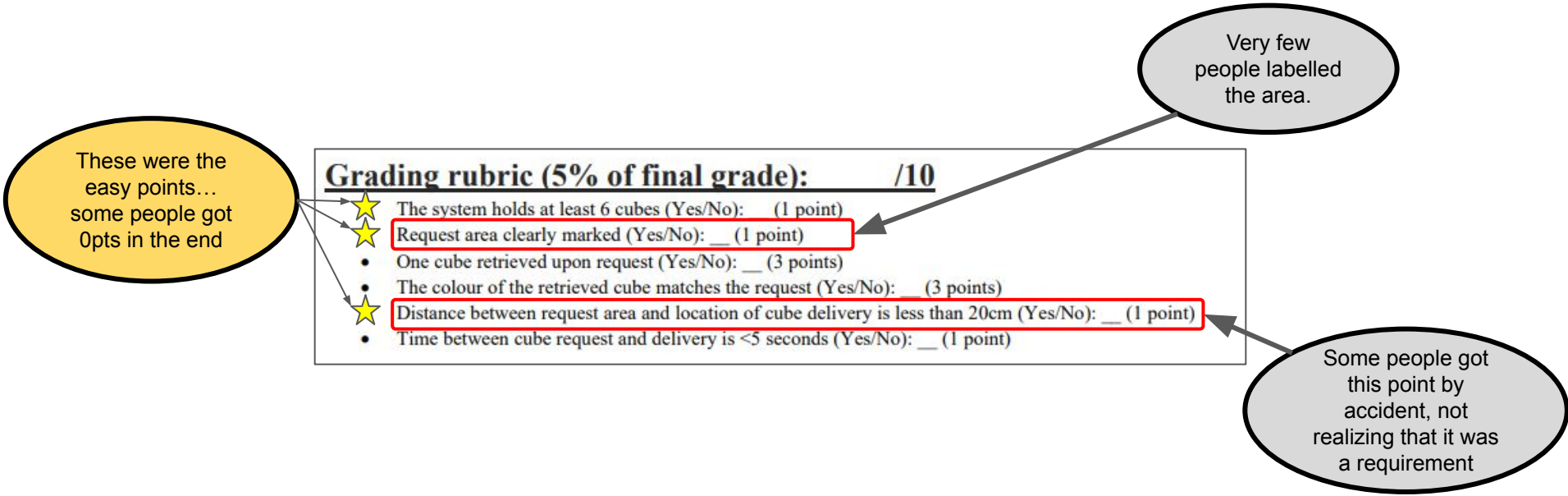
- Team members should expect to briefly explain how the system works (or is supposed to work), and outline how the system is to be loaded. There will be no powerpoint/slide presentation (we will not have any projector set-up).
- One of the professors will place six cubes (two of each colour, with three colours of the team's choosing) as instructed by the team.
- The team will be allowed to trigger the system to start the sorting process (if any) to demonstrate the prototype in action.
- Within 2 minutes of having loaded the system or started the sorting process (whichever is later), one of the professors will place one cube (of one of the three colours) onto the request area, and trigger the retrieval process.

Sorting Process is separate!

Beta/Final Demo

The rubric in this semester, was given ahead of time. Students had a decent amount of time to review and use it to get easy points. They didn't all do that.

If you are given the Beta/Final Demo rubric ahead of time, analyze for at least 1 15 min, and use it as a checklist for the demo. What you need to include, and what you need to do for easy points, even if your robots completely fails.



Final Presentation to the Professors

NOTE: This is all based on the Winter 2022 Semester “Final Design Review”. Subsequent semesters may ask for more or less information.

Design decisions, project management, and test details.

These are the things that professors will focus on. If your whole team stays up to date on this information, then you will have no problem whenever they ask something.

By your team’s scheduled design review session, all three professors will have reviewed your design submissions. They will be asking questions to your team based on your written submissions (design files, final design report etc) for your team to address. These questions include questions related to the design decisions you’ve made, how the project was managed, and details of the tests that were conducted.

Single-slide presentation, 1 slide

You should be able to read the important information with a computer screen, no zooming in required.

The point is to showcase your final system. Simple, no worries.

Preparing for the design review

Your team should prepare **one visual** that best depicts your final system and your design project journey. Think of the visual as **a single-slide presentation or a poster** made for online presentation. It could be a photo of your final system with subsystem labels, a drawing/model of your system, or series of drawings representing the evolution of your design.

Conclusion: The General Idea of the Final Report

Throughout the history of ECSE 211, the exact grading of the Final Report (the Final Submission of all your work for the Final Project) has been something of a mystery, a black box. Basically:

Don't ask the TAs for a rubric, we don't have one. There is no grading rubric.

The grading is a little confusing, so here's the gist of it:

If you do well in the course, your Letter Grade will reflect that in the end. Do not pay attention to the number value of the grade posted on MyCourses.

The real problem is that it is easy to neglect the Final Report. Students will make a great robot, but write nothing down…

<u>This is your main advice for the Final Report.</u>

Use the Labs as a baseline. Ask your friends for their Lab reports. Take the nicest one you can find, and make your Final Report *at least as good as this Lab Report.*

For some reason, students like to write less on their Final, than in the Lab reports. That is a mistake.

This report determines your final project grade. Make it clean and concise. Do not omit important information.

Final Report Guidelines

The guidelines for each semester may change. But this semester's general guidelines will be used to tell you what to include in your report.

Each of the sections highlighted in red indicate sections that were left up to the interpretation of the team.

Each section asks for, essentially, **Reproducibility**

Through this document, some random nobody must be able to reproduce the exact steps you took in doing this entire project. **You cannot just summarize.**

You must include all decisions and procedures that you made and followed during this project.

We will now go over each section, and break down what you can include. In the end, it is still up to your decision, how to write this report.

Final Design Report

There is no template provided to the students for the final project report. There is a page limit of 25 pages maximum of content (i.e., title page, table of contents, and contents in the Appendix don't count towards your page count). You should feel free to arrange the subheadings, ordering of contents and the presentation of the contents as you find most appropriate for your team. However, at minimum, the following contents must be included in the report, and presented in a way that makes it easy for your team members and your readers navigate the contents:

a) A title page listing the member of your team, version of the report being submitted, and date associated with the version

b) Introduction of the team members, how the team was organized (e.g., roles, team contracts, schedules etc.), and a description of your team's project management (e.g., what milestones did you have, how did you track your team's progress over the design weeks, how did the budget shift over time accordingly etc.)
   - At the end of reading the report, readers should have an understanding of how they might tackle the same design problem differently (e.g., form a team with different set of expertise, budget the time or prioritize tasks differently, have a different team contract to enable more effective teamwork).

c) The problem statement translated into the scope of the problem, requirements, constraints, and specifications by your team
   - At the end of reading the report, readers should have a clear idea of how you understood the design problem (in your team's words), your team's assumptions about the task, what it meant for your team's design process.

d) Description of your designed system, including how the system and its components evolved from the initial idea to the final product, and hardware and software design of your system.
   - At the end of reading the report, readers should have a clear idea of how your system works, how each components of the system functions and are built, how to recreate the final system using your existing documentation, and what design modifications to avoid trying (because you've tried them before and they didn't work, for reasons you've outlined).

e) Tests that have been conducted and their results, their implications to meeting the design specifications, with a description of how these test results affected design decisions for your hardware/software/system design.
   - At the end of reading the report, readers should know what tests were conducted when, what the results were, and how they impacted your subsequent design decisions. They should also have an idea of limitations of different components of your system (e.g., sensors and actuators used) based on your test reports, as well as be able to reproduce the results following your team's documented procedures.

f) Performance and limitations of the final system
   - At the end of reading the report, readers should know what the system is capable of, how performant the system is, how one might use the system, and what the system cannot do or should not be used for.

Final Report Guidelines

b) Introduction of the team members, how the team was organized (e.g., roles, team contracts, schedules etc.), and a description of your team's project management (e.g., what milestones did you have, how did you track your team's progress over the design weeks, how did the budget shift over time accordingly etc.)

- At the end of reading the report, readers should have an understanding of how they might tackle the same design problem differently (e.g., form a team with different set of expertise, budget the time or prioritize tasks differently, have a different team contract to enable more effective teamwork).

"Introduction of the team members, how the team was organized" are fairly straightforward parts, and should be easily done.

But "a description of your team's project management" may be misleading. It is not just "a description," it should include many things.

Milestones and Goals. How you track progress. Task Prioritization. Weekly Budget Reallocation. Schedule Conflicts. Team Dynamics. How you decided which members did which tasks. Necessary Decisions made in case of roadblocks. Unmet expectations of the team. Communication/Miscommunication?

Final Report Guidelines

c) The problem statement translated into the scope of the problem, requirements, constraints, and specifications by your team
   - At the end of reading the report, readers should have a clear idea of how you understood the design problem (in your team's words), your team's assumptions about the task, what it meant for your team's design process.

This one is actually incredibly simple, because you should have done this entire point (C) for **one or two Lab reports by now.** Find the best Lab report of "problem, requirements, constraints, and specifications" and do as much the same as you can for this section of the Final Report.

Final Report Guidelines

d) Description of your designed system, including how the system and its components evolved from the initial idea to the final product, and hardware and software design of your system.
   - At the end of reading the report, readers should have a clear idea of how your system works, how each components of the system functions and are built, how to recreate the final system using your existing documentation, and what design modifications to avoid trying (because you've tried them before and they didn't work, for reasons you've outlined).

This should be a long section.

Design Ideas, Advantages, Disadvantages, Prototypes, Modifications, Design Iterations, Major Design Decisions, Hardware Iterations, Software Changes/Versions.

There are many things to talk about, and you need to explain **how these things should work** as well.

This is the main description of the development of your System. These are "the things that you did, for this project". If this section ends up being small, then you are telling the reader that you did nothing for this project.

Final Report Guidelines

e) Tests that have been conducted and their results, their implications to meeting the design specifications, with a description of how these test results affected design decisions for your hardware/software/system design.
   - At the end of reading the report, readers should know what tests were conducted when, what the results were, and how they impacted your subsequent design decisions. They should also have an idea of limitations of different components of your system (e.g., sensors and actuators used) based on your test reports, as well as be able to reproduce the results following your team's documented procedures.

Once again, this may be slightly misleading in wording.

You'll need to include BOTH explicit **Test Procedures** and **Test Execution Results**. These are two different things.

The Procedure says "this is how to do the test, and how it should help us". The Execution says "these were the results of this test, these are the actual conclusions that we made, and the discoveries we had found".

AND you need to include how each of these test results affected your Design decisions. **The Design Implications**.

3 major pieces of information should be in this one section.

Final Report Guidelines

f) Performance and limitations of the final system
   - At the end of reading the report, readers should know what the system is capable of, how performant the system is, how one might use the system, and what the system cannot do or should not be used for.

Lastly, this one is a short description, but it should be a medium length size compared to the other sections.

Performance Assessment. System Limitations. Implemented Features. Unimplemented Features. Intended Use. Final System Specifications. Guidelines on Proper Usage.

Any of the above phrases could refer to the information that you should include in this section. It's up to your decision in this area.

It's ultimately a full description/summary of the end result of this whole project.
What did you make? Did it do the job?

Section 4: Q&A for students with a lot of questions

Table of Contents