RYERSON UNIVERSITY

Department Aerospace Engineering

Faculty of Engineering & Architectural Science

Course Code	AER 627
Section	02
Course Title	Introduction to Space Robotics
Semester/Year	Winter 2021
Instructor	John Enright
TA	Joel Moore
Project 2	Barcode Sorting Robot
Submission Date	March 5th 2021

Name	Student ID	Signature*
Jann Cristobal	500815181	Jann C.
Stephanie Chan	500819304	SC

*By signing above you attest that you have contributed to this submission and confirm that all work you have contributed to this submission is your own work. Any suspicion of copying or plagiarism in this work will result in an investigation of Academic Misconduct and may result in a “0” on the work, an “F” in the course, or possibly more severe penalties, as well as a Disciplinary Notice on your academic record under the Student Code of Academic Conduct, which can be found online at: http://www.ryerson.ca/senate/policies/pol60.pdf

Abstract

In this project, a sorting robot was successfully built and succeeded in about 81% of the final test runs. The robot decodes a code 39 glyph that corresponds to one of the four unloading sites. By the end, the decoding algorithm worked for 100% of the 300 randomized characters generated. The sorting and positioning mechanism and logic all worked flawlessly.

Table of Contents

List of Figures

List of Tables

1 Introduction

In this project, the goal is to design a robot that can identify four distinct code 39 glyphs and sort it in the appropriate bin that it is assigned.

2 Theory

This section outlines the mathematical theories utilized in designing and programming the robot.

2.1 Package A: Sorting (Stephanie Chan)

In this project, a robot which can sort barcodes into 4 specific bins will be built. The robot will consist of a conveyor belt which the barcode will be placed on. After passing through a colour sensor, a mechanism will sort the robot into its respective bin. Figure 2.1 below shows a simple sorting robot example given in [1].

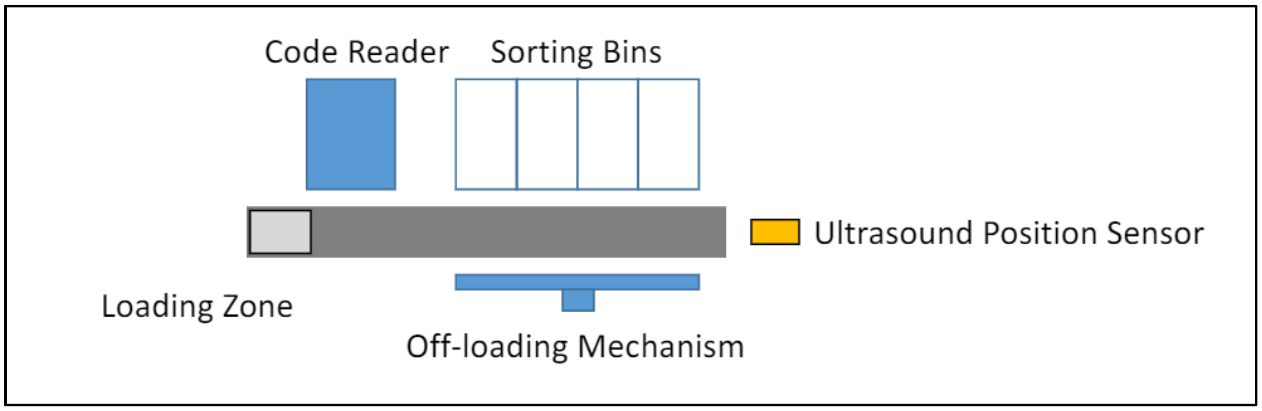


Figure 2.1 A simple, suggested robot design [1]

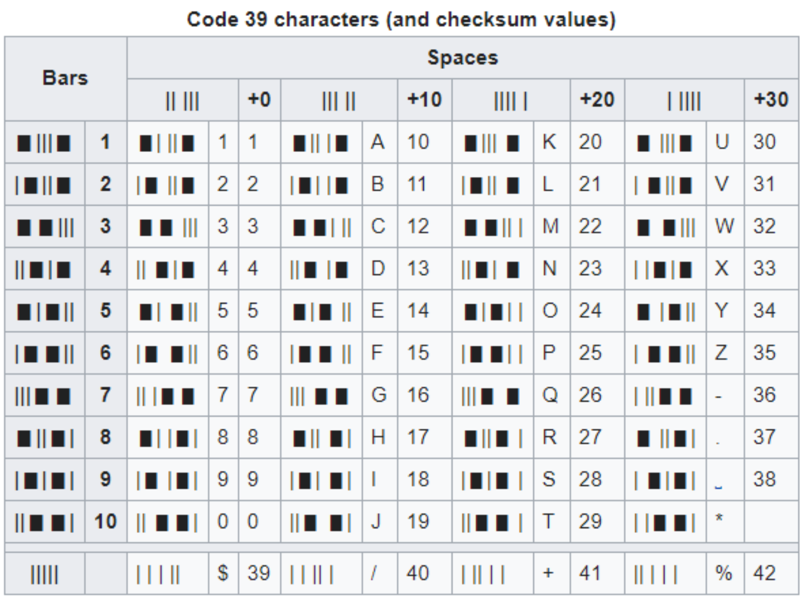
As shown in figure 2.1, the barcode is first placed on the left side of the conveyor belt, the loading zone, then it is passed through the code reader. After that, the conveyor belt continues to move the barcode until it matches up with its correct sorting bin. The ultrasound position sensor indicates when the barcode has reached its bin. Finally, an offloading mechanism will push the barcode into the bin.

2.2 Package B: Barcode (Jann Cristobal)

In this project, the barcode system that will be used is code 39. This system is able to identify 43 characters which are numeric digits (0 to 9), uppercase letters (A to Z), and special character symbols (-, ., $, /, +, %, *, and space). Each character is arranged such that there are nine elements of five black bars and four white spaces. They are differentiated as narrow and wide. The placement and sizing of these elements relative to one another is how they are identified.

In practice, a special character "*" is used as the start and stop character. The inherit asymmetry of the asterisk allows the scanner to easily identify the orientation of the barcode. Thus, if it is scanned forward or backward it is still identifiable.

A single character always starts and end with a bar. Location of two wide bars encodes a number between 1 and 10. Location of wide space (4 possible) classify four distinct group of characters. Refer to Figure 2.2 to see the bar positioning and the 4 distinct groups.



Code 39 characters (and checksum values)

Bars		Spaces											
		|| |||		+0	||| ||		+10	|||| |		+20	| ||||		+30
■|||■	1	■| ||■	1	1	■|| |■	A	10	■||| ■	K	20	■ |||■	U	30
|■||■	2	|■ ||■	2	2	|■| |■	B	11	|■|| ■	L	21	| ■||■	V	31
■■|||	3	■■ |||	3	3	■■| ||	C	12	■■|| |	M	22	■ ■|||	W	32
||■|■	4	|| ■|■	4	4	||■ |■	D	13	||■| ■	N	23	| |■|■	X	33
■|■||	5	■| ■||	5	5	■|■ ||	E	14	■|■| |	O	24	■ |■||	Y	34
|■■||	6	|■ ■||	6	6	|■■ ||	F	15	|■■| |	P	25	| ■■||	Z	35
|||■■	7	|| |■■	7	7	||| ■■	G	16	|||■ ■	Q	26	| ||■■	-	36
■||■|	8	■| |■|	8	8	■|| ■|	H	17	■||■ |	R	27	■ ||■|	.	37
|■|■|	9	|■ |■|	9	9	|■| ■|	I	18	|■|■ |	S	28	| ■|■|	␣	38
||■■|	10	|| ■■|	0	0	||■ ■|	J	19	||■■ |	T	29	| |■■|	*	
|||||		| | | ||	$	39	| | || |	/	40	| || | |	+	41	|| | | |	%	42

Figure 2.2: Code 39 Characters [2].

The two wide bars, out of five possible positions, encode a number between 1 and 10 using a two-out-of-five with the following numeric equivalence: 1,2,4,7,0. Ex: number 6 is encoded NWWNN, wide bars occupying 2 and 4 (2+4 = 6). The last four characters consist of all narrow bars and three wide spaces. There are four possible positions for the single narrow space [2].

3 Design Documentation

This section will show the design process involve of building and programming a sorting robot as well as the decoding of various code 39 glyphs.

3.1 Package A: Sorting (Stephanie Chan)

This section will show the steps taken to design a robot which can accurately sort 4 different barcodes into their respective bins.

The first step in designing this robot was to brainstorm ideas. Initially, a simple design was given in the project handout which can be seen in figure 2.1 and figure 4.1. This design's main benefit was the low number of moving parts. It would result in a durable, simple robot. However, an important drawback was its lack of space for the barcode block. A smaller barcode block resulted in a smaller font for the barcode which reduced the accuracy of the readings.

A second design was made which increased the space for the barcode block. This design can be seen in figure 4.2. This design solved the problem of inaccurate readings due to a small font. However, this second design had issues sorting because of distance readings. A third and final design was made to fix both of the two main problems from the previous two designs. For more details of the brainstorming process, refer to section A in the project operations.

The final design can be seen in figure 4.3. Two sorting bins are placed on either side of the conveyer belt. A rotating arm above the belt pushes the barcode blocks into the correct bin. Designing a mechanism which could sort the blocks into bins placed on both sides of the conveyor belt while also not covering the sensors was imperative to this design. For this reason, a rotating arm was the best choice.

The following procedure was followed to sort the barcode blocks:

1. Place the barcode on the loading zone.
2. The conveyor belt moves the barcode block past the colour sensor.
3. The barcode decoding algorithm takes the colour sensor data and outputs the correct character.
4. The conveyor belt moves the block to the correct position in front of the correct bin.
5. The rotating arm pushes the block into the correct sorting bin.
6. The conveyor belt resets to its starting position and is ready for the next barcode block.

A more detailed breakdown of how the robot works to sort and decode can be seen in section B in project operations.

3.2 Package B: Barcode (Jann Cristobal)

This section will show the various steps taken in coming up with the decoding algorithm, simulation tests, and implementation with the sorting robot.

3.2.1 Decoding Algorithm

In the theory section, the method of encoding all 43 characters was discussed. This method efficient and beneficial for a barcode that contains multiple characters because they are read all at once using a scanner. The errors tend to be less compared to the method used in this project.

Due to the limitation of color sensor, there are several assumptions made in order to decipher the barcodes more precisely. Firstly, only a single character will be used in order to minimize the bars and spaces scanned for more accurate readings. Secondly, the barcode will always be scanned in the correct orientation and direction relative to the conveyer belt. Lastly, is that each of the 9 elements will be assigned a value between 1 to 4 corresponding to narrow bar, wide bar, narrow space, and wide space, respectively.

Initially, the four characters chosen were A, B, C, and D. But after numerous testing, a more robust design will be to choose from the four distinct groups corresponding to the wide space placement will be the smartest. By choosing characters that are far apart from one another in the barcode list, it reduces the chance of misinterpretation when decoding.

The logic for decoding a barcode signal is shown in Figure 3.1

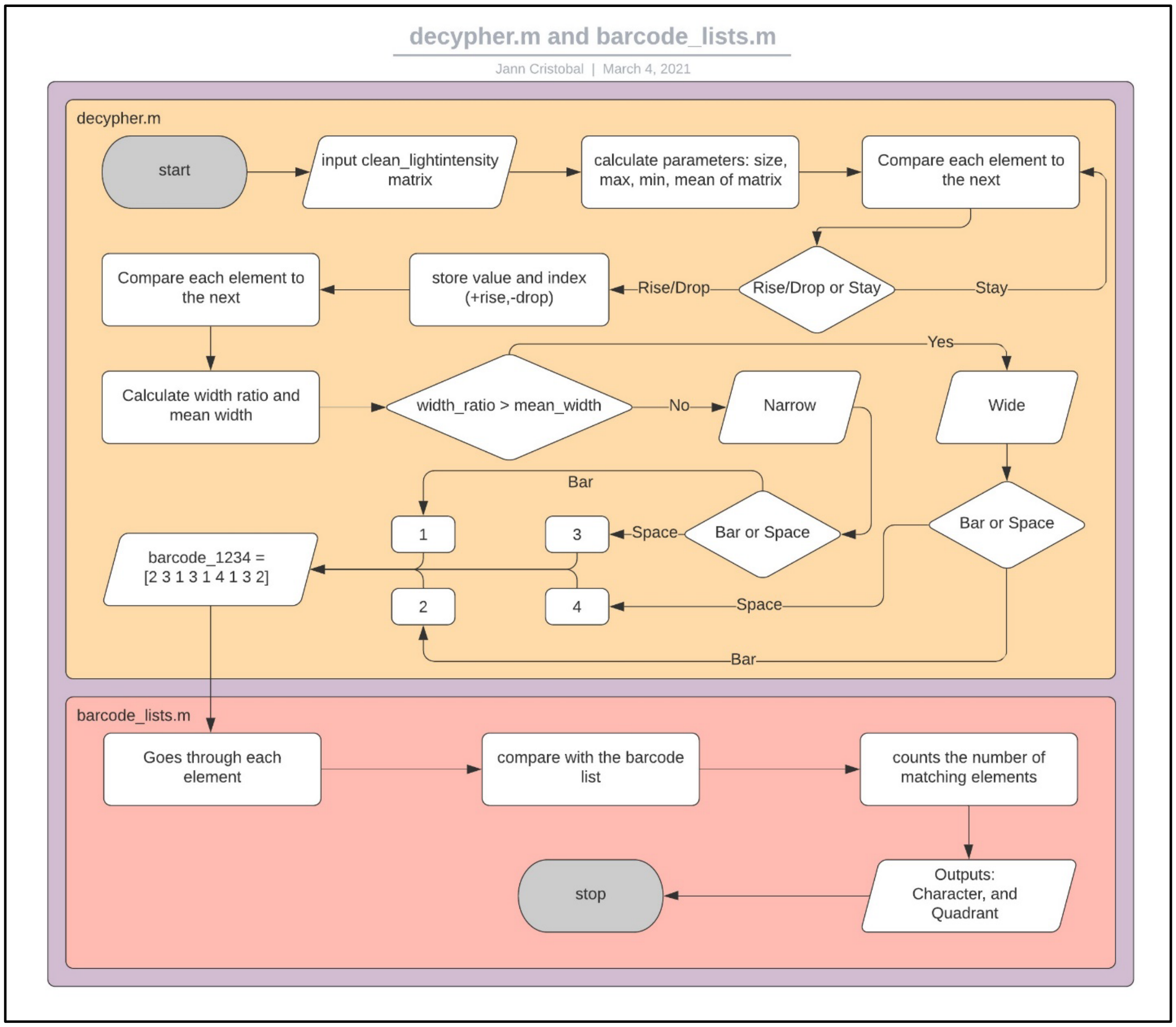

decypher.m and barcode_lists.m
Jann Cristobal | March 4, 2021
decypher.m
start
input clean_lightintensity matrix
calculate parameters: size, max, min, mean of matrix
Compare each element to the next
Rise/Drop or Stay
Stay
Rise/Drop
store value and index (+rise,-drop)
Compare each element to the next
Calculate width ratio and mean width
width_ratio > mean_width
Yes
No
Narrow
Wide
Bar
Bar or Space
Space
Bar or Space
1
3
2
4
Space
Bar
barcode_1234 = [2 3 1 3 1 4 1 3 2]
barcode_lists.m
Goes through each element
compare with the barcode list
counts the number of matching elements
Outputs: Character, and Quadrant
stop

Figure 3.1: Decoding Algorithm Flowchart

3.2.2 Simulation

In order to test the algorithm remotely, ***sim_barcode.m*** was used to generate light intensity values for a given character. This simulation was useful for testing the algorithm and was able to correctly decode the barcode accurately. Some limitation that the simulation has compared to the actual readings from the color sensor is that it does not fully represent the light intensity and variation from external noise.

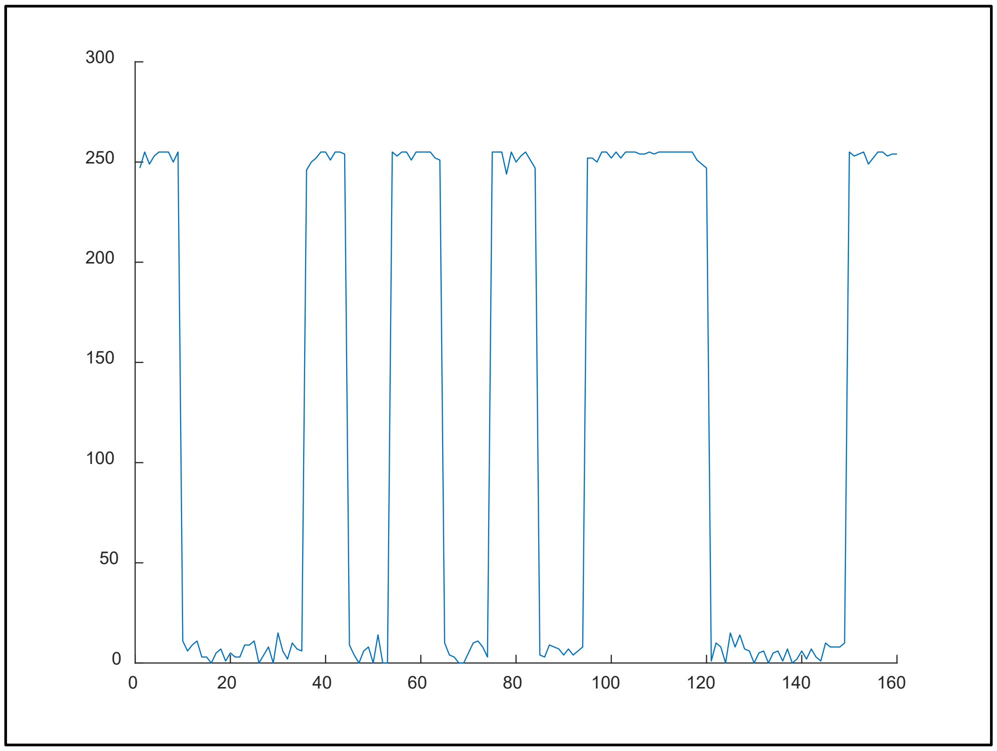
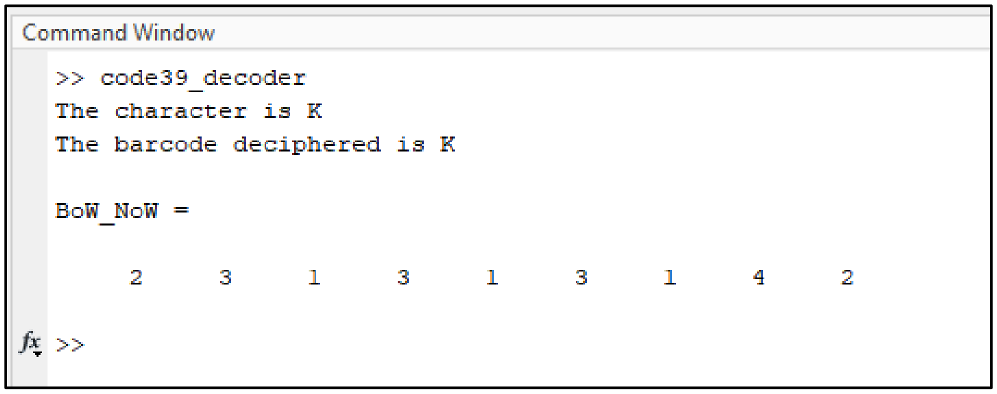


Figure 3.2: Simulated barcode signal for K



```
>> code39_decoder
The character is K
The barcode deciphered is K

BoW_NoW =

     2     3     1     3     1     3     1     4     2

>>
```

Figure 3.3: Result from decoding algorithm

3.2.3 Implementation

When integrating the decoding with the sorting robot, there were several adjustments that were made with regards the method and algorithm. The biggest issue was the large amount of noise. Analyzing the raw light signals could not be possible because the program would find more rise and drop in signal compared to the simulation. Fortunately, Dr. Enright provided us with a function called ***clean_barcode.m*** which improved signal from the color sensor readings. I personally made a small change with this function in order to improve accuracy of the cleaning process. The variables ***opts.rise_thresh_abs*** and ***opts.rise.thersh_abs*** were increased from 2 to 10. This made the function less sensitive to small rise and drop with the signal.

Once fully integrated, the only issue left to solve was the occasional in accurate reading for one or more of the nine-element matrix. This problem is most notable for the bar scanned after a wide space. One of the solutions to fix this problem was to 3D print a block and paint it. The second one was to test more characters from various places in the barcode lists to find the most reliable ones.

4 Project Operations

This section answer questions regarding the robot's operation and the results from all the tests when designing and implementing the various parts.

A. ***Before you even touch the Lego, brainstorm some ideas for how to make your robot work. This document describes the specific parameters of this design challenge, but you are left with quite a lot of latitude to implement your robot in different ways. Describe few of your concepts and outline your logic for selecting your design (you do not need any formal trade off processes for this)***

When trying to find the best design, it is important to compare and rank the various objectives of the project. In this case, there are four main goals which are accuracy, precision, durability, and simplicity. The pairwise comparison chart below ranks the importance of each objective.

Table 4.1: Pairwise Comparison Chart for design goals

Goals	Accuracy	Precision	Durability	Simplicity	
Accuracy	-	0	0	1	1
Precision	1	-	0	1	2
Durability	1	1	-	1	3
Simplicity	0	0	0	-	0

This table ranks the goals for this project. The one with the highest value is the priority.

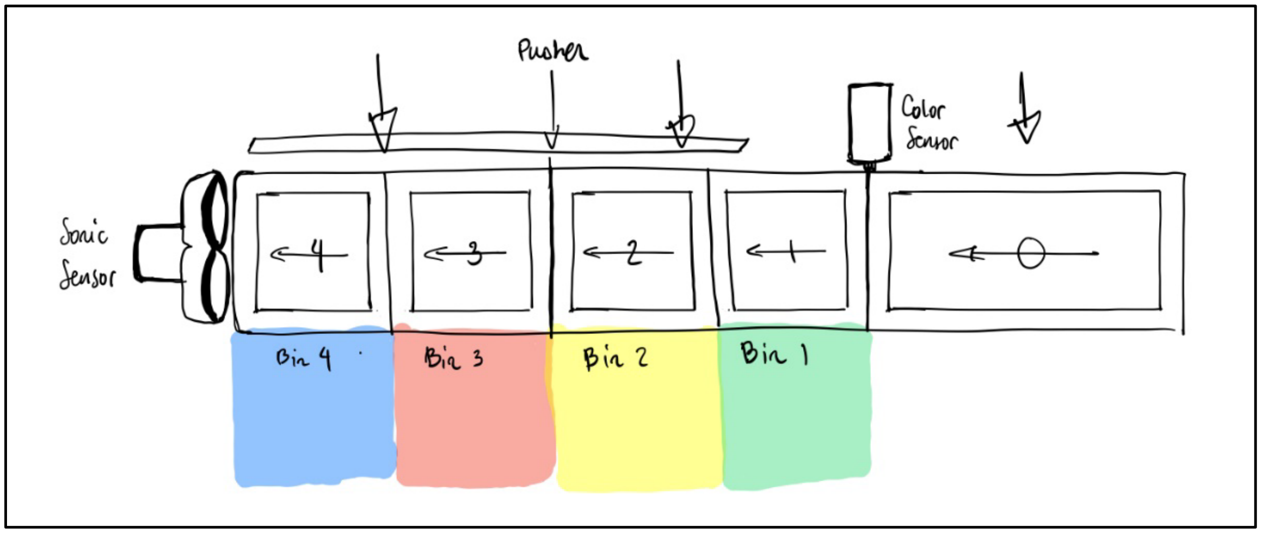


Figure 4.1: Design concept 1

Figure 4.1 shows the design concept 1. This idea was taken directly from the sketch provided in the project description. The benefit to this design is the simplicity and durability. Out of the three design concepts, it has the least number of moving parts. The problem with this design is the lack of space for the barcode block. The smaller size requirement will affect the accuracy and precision of the scanning and decoding.

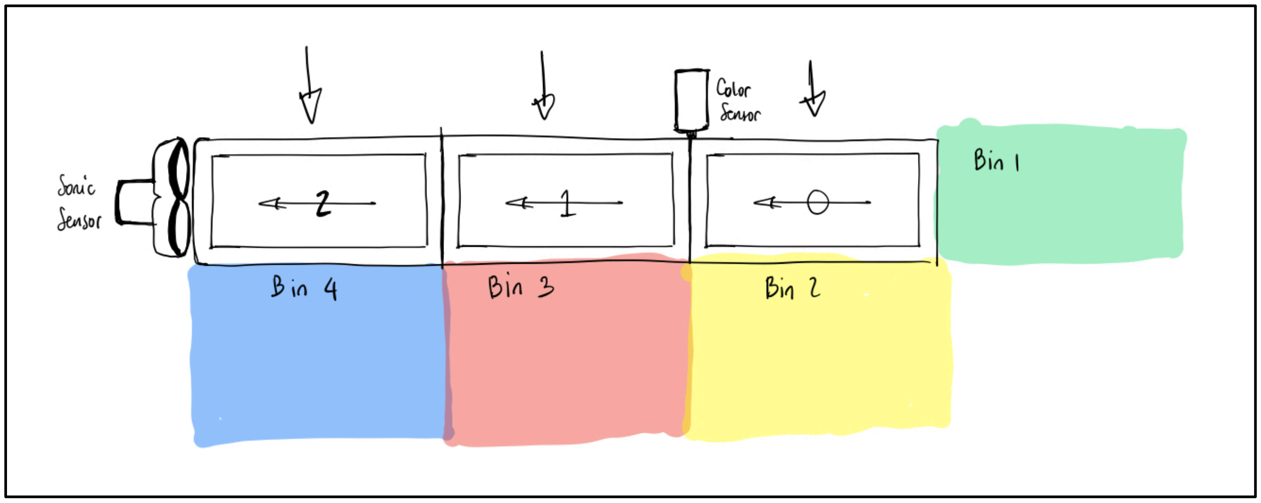


Figure 4.2: Design concept 2

Figure 4.2 shows the second design concept for the robot. This design kept the simplicity of the initial design and allows for a larger block to be scanned. This will increase the accuracy and precision with the scanning but will suffer from distance measuring with the sonic sensor. Initial tests shows that the sonic sensor begins to read inaccurately around 16cm. Thus, position the block to fall in bin 2 is a challenge.

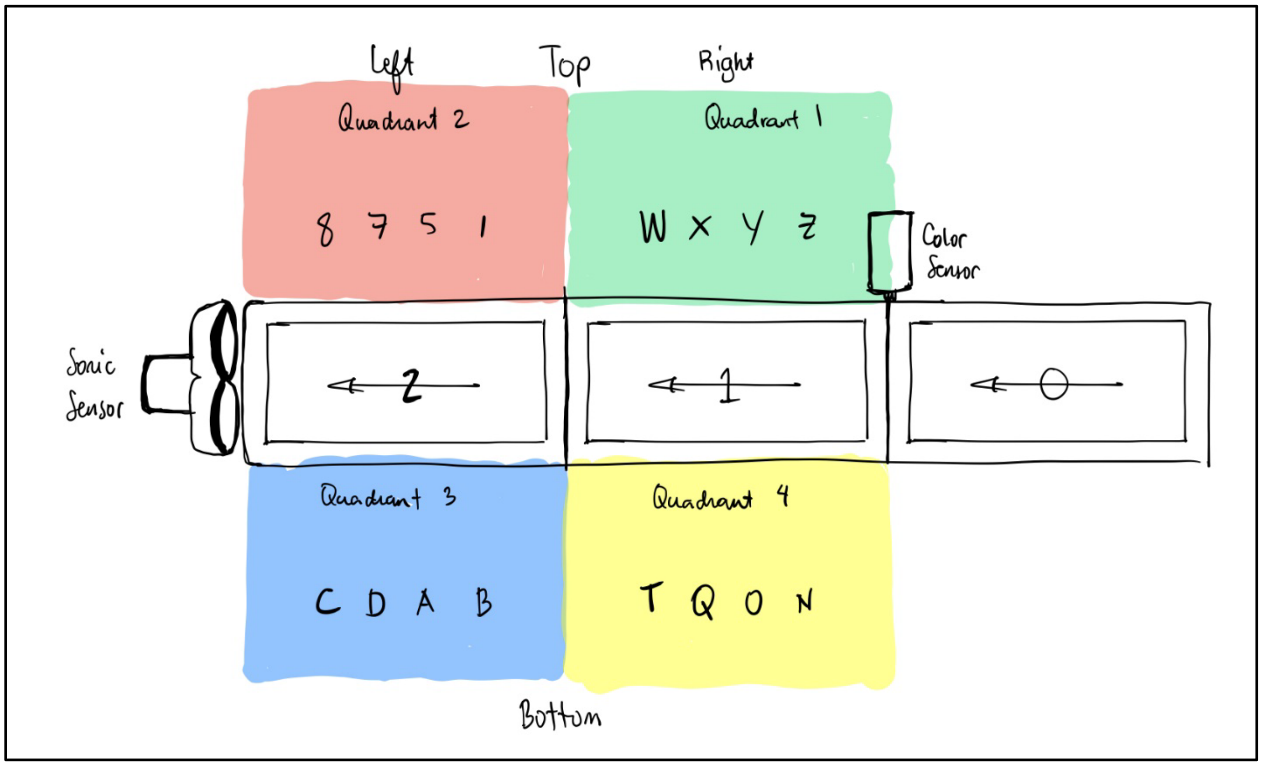


Figure 4.3: Design concept 3

Figure 4.3 shows the third and final design concept. The final design kept the use of larger barcode block for improved accuracy and precision with the barcode scanner. This design does not suffer the same flow as the second one in terms of the limitation with the sonic sensor. By having the blocks and back plates measured closer to the sensor, it minimizes inaccuracies. The challenge that arise from this design is the more complex pusher design required. It would have to be something that will not get in the way of the sonic sensor and be able to pusher towards the top and bottom.

B. ***Once you have the high-level concept of how your robot will work to break down the big task (i.e scan the target) into a series of more specific tasks. You want your design to work reliably, so you should be careful about your logic (e.g., "move target by X cm" vs. "move target by X cm pass the sensor")***

The robot's mission from start to end can be broken down into three main parts, decoding, sorting, and resetting.

Decoding is the first major task and is relied upon by the subsequent tasks. The detail algorithms of how the barcode was decoded can be referred to in the design documentation package B. But to summarize, it starts by having the color sensor read the reflected light intensity of the environment. It does not start storing the magnitude of the intensity until it detects the block. The blocks were designed to always have a white space as the edges to signal the start. The reading process will run for a predefined number of times. The length of the loop is determined by both the speed of the conveyer belt as well as the time delay between each reading. Once the light intensity signal is stored in an array, it gets processed using ***clean_barcode.m*** to removes noise and enhance the peaks and troughs. Then, the clean signal is analyzed to find all the rise and drop locations. Then the width between these rise and drop are measured and compared to the mean value. If the width is greater than the mean value, it is considered wide. If not, then it is narrow. Next, is that a nine-element matrix will contain a set of digits between 1 to 4 which represents narrow bar, wide bar, narrow space, and wide space, respectively. Finally, the character is determined by comparing the nine-element matrix to the list of barcodes.

Sorting begins immediately after a character is determined. If the scan failed, it skips this step and the system resets. If the scan is successful, the first step is to home the mechanism. Homed location is when the block is positioned such that it is less than or equal to 4cm away from the sonic sensor. Once it is homed, the conveyer motor either stops or reverses rotation if the block scanned is for position 1. Once the block is in position, the pusher motor rotate according to which side the block need to drop. The pusher is limited by the touch sensor, which reverse the pusher motor's rotation to reset it back to neutral position. This is achieved by adding a time delay which was experimentally determined. At this point, the sorting is over.

Resetting is the last major step for the robot's mission. Is start by drive the conveyer belt towards the sonic sensor. Once the backplate is detected to be in the homed position, the conveyer belt motor reverses direction and runs for a given time delay of 12 seconds. The loading dock is positioned correctly, and the program stops.

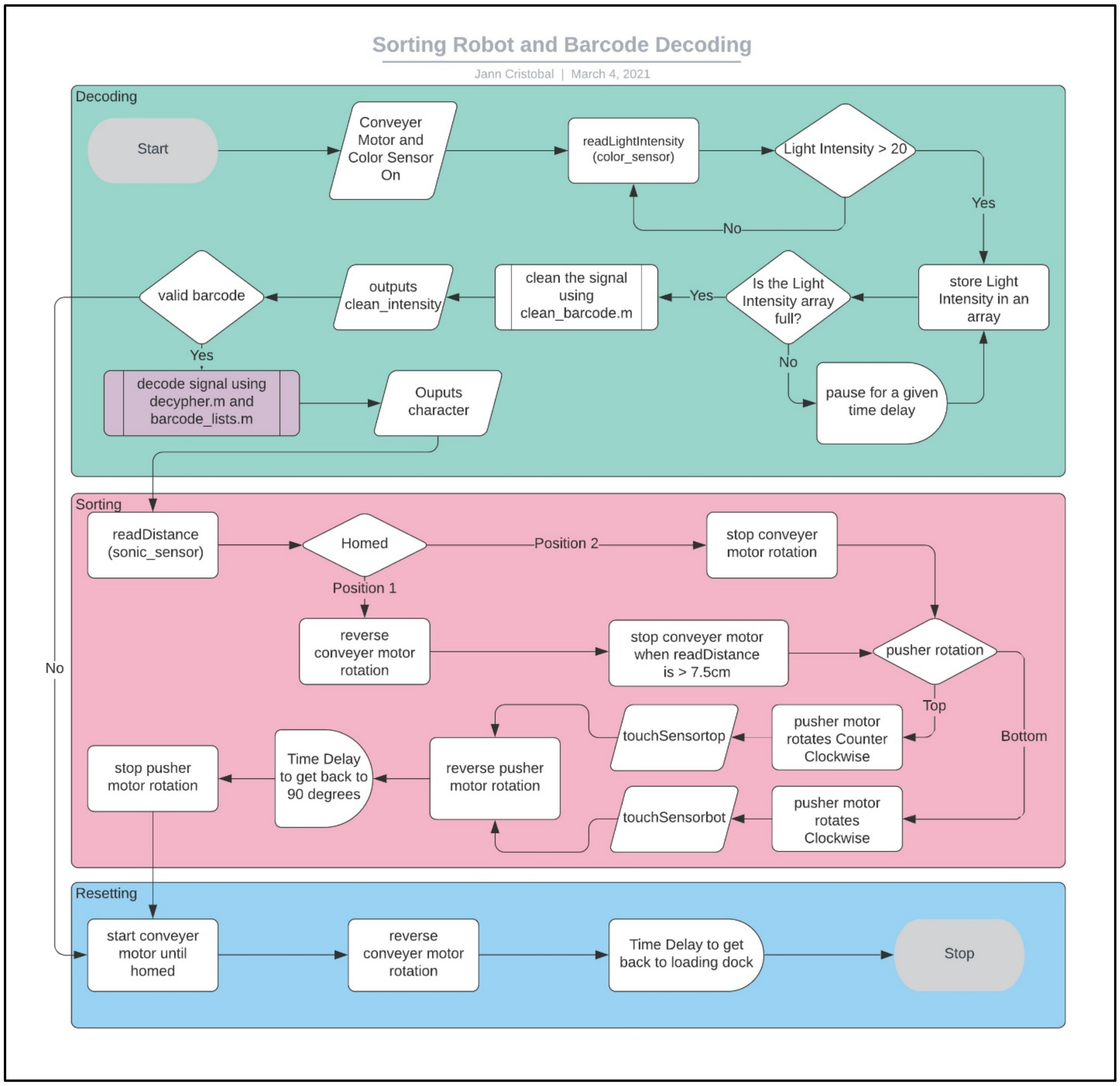

Sorting Robot and Barcode Decoding
Jann Cristobal | March 4, 2021
Decoding
Start
Conveyer Motor and Color Sensor On
readLightIntensity (color_sensor)
Light Intensity > 20
Yes
No
store Light Intensity in an array
Is the Light Intensity array full?
Yes
No
pause for a given time delay
clean the signal using clean_barcode.m
outputs clean_intensity
valid barcode
Yes
No
decode signal using decypher.m and barcode_lists.m
Ouputs character
Sorting
readDistance (sonic_sensor)
Homed
Position 2
Position 1
stop conveyer motor rotation
reverse conveyer motor rotation
stop conveyer motor when readDistance is > 7.5cm
pusher rotation
Top
Bottom
pusher motor rotates Counter Clockwise
pusher motor rotates Clockwise
touchSensortop
touchSensorbot
reverse pusher motor rotation
Time Delay to get back to 90 degrees
stop pusher motor rotation
Resetting
start conveyer motor until homed
reverse conveyer motor rotation
Time Delay to get back to loading dock
Stop

Figure 4.4: Sorting Robot and Barcode Decoding Implementation Flowchart

The logic for the whole system from decoding, to sorting, and resetting can be seen in Figure 4.4.

C. Build and program your robot. DO some initial testing and try to correct any glaring problems with the implementation (programming or the mechanisms).

The final design drawing design can be seen in Figure 4.5.This design solved issues pertaining to inaccurate distance reading from the sonic sensor, asymmetric push are hitting the back plate and largest possible block size. The initial concern with the pushing mechanism blocking the sonic sensor was also solved.

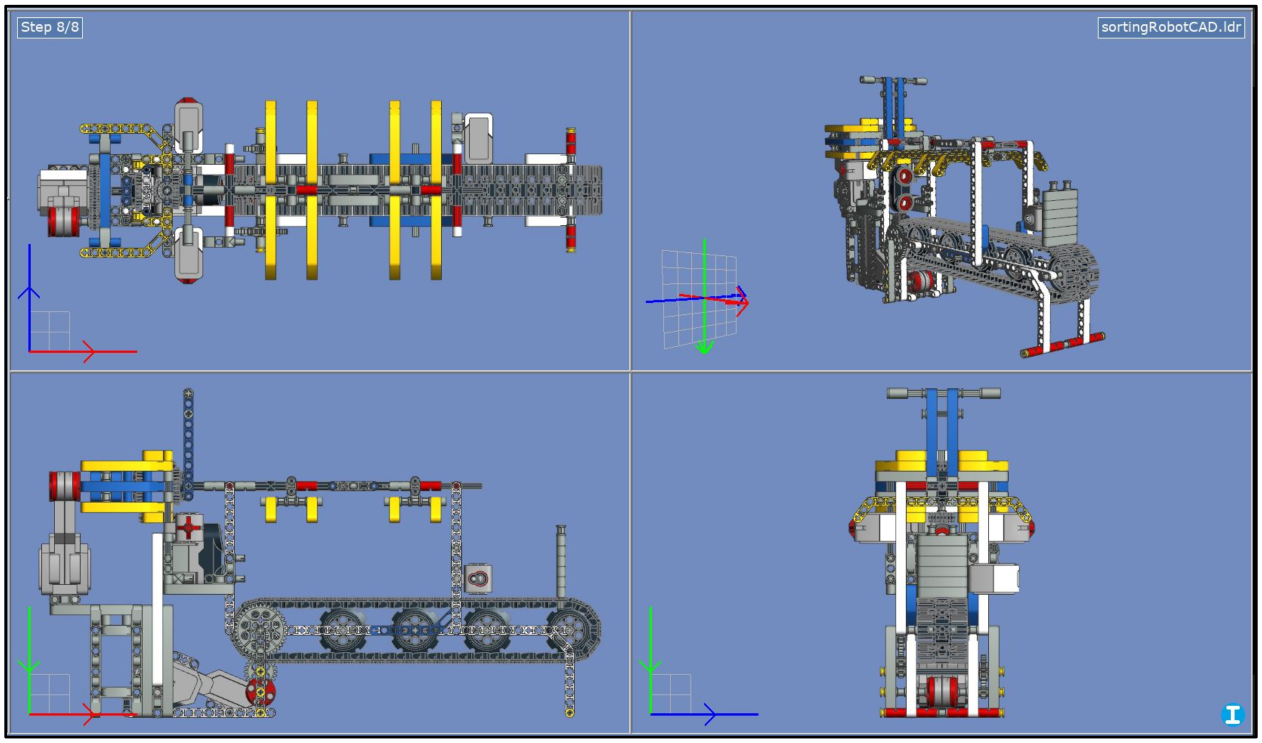


Figure 4.5: Final Design Drawings

One of the solutions to improve accuracy of the reading was to design a block with the exact dimension that is optimized to be scanned by the color sensor. Figure 4.6 shows the final design for the barcode block.

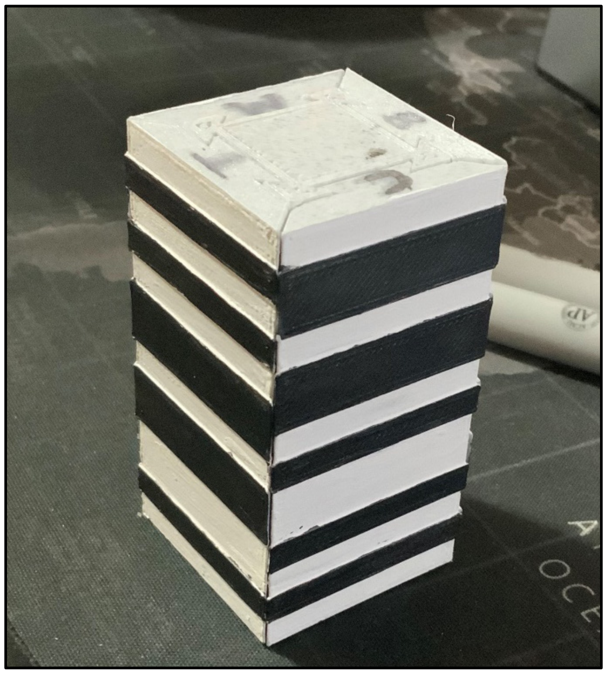


Figure 4.6: 3D printed barcode block design

Figure 4.7 shows the demonstration for sorting Y barcode block. The raw data is represented by the noisy orange curves. The processed signal is shown in blue and are used to decode the glyph. The troughs represent black bars and peaks represent white spaces. Refer to the appendix to see the rest of the test results.

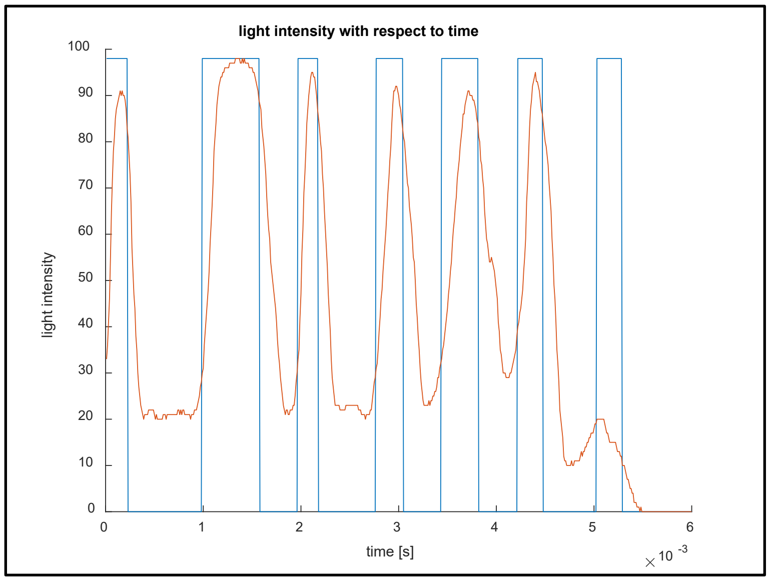


Figure 4.7: Light Intensity signal for Y

D. ***Discuss the necessary calibration of the scanning and motion control components. How do actual system behaviors differ from ideal expectations?***

There were numerous factors that affected the accuracy and precision of the scanned bar codes. The first factor is the width of the bars. Based on the tests results, the minimum bar width that the color sensor can recognize was around 0.46cm. This was just based off of a small number of font sizes tested. The second factor is the speed of the conveyer belt. Higher speed had significantly more noise than lower speed. Even if the block is placed squarely at the start, the uneven surface causes it to move resulting in more noise in the signal. Utilizing a gear reduction, the speed of the conveyer belt was reduced and showed improved results in terms of reducing the noise. Thirdly, when scanning a printed barcode on a piece of paper, the peaks and troughs were not as separated in terms of intensity. To alleviate this problem, a 3D printed block was painted black and white for each of the given patterns. Fourth is the distance of the color sensor from the block. Based on numerous tests, having the light reflection diameter of about 1cm yield the best results. Lastly, by modifying the ***clean_barcode.m*** function to be less sensitive with magnitude changes, it gave more consistent processed signals.

E. ***Run a series of about 12 sorting runs, with a variety of glyphs. What is your success rate? For any failed attempts, identify the cause of failure and relate these failures to any deficiencies in your design. Redesign your robot, explaining any software or hardware changes and rerun the tests. Compare your new success rate to the old. Note: for this part of the project, try and concentrate on interesting design problems and not just mistakes. For example, forgetting to stop the conveyor before you push the target off the belt is a mistake (implementation problem); realizing that you need to better logic to position the target is a design problem.***

Trial 1: Inconsistent distance readings

At this stage, sorting and decoding are still being worked on separately. The success rate for sorting is about 25%. The main issue was that the distance measured was inconsistent. The block would miss its target going passed it or falling short of it. Having the sonic sensor oriented horizontally kept reading the wrong distance. But by rotating it to be oriented vertically, it solved the issue. Another issue was that initial position of the sonic sensor limited the available space in the conveyer belt. Since the minimum distance it can measure precisely was around 3cm, moving it 3cm backward allowed for more room in the conveyer belt.

Trial 2: Positioning Issues

At this stage, sorting and decoding are not fully integrated together. The success rate in terms of positioning is about 70%. The issue was trying to find exact positioning of the block by using == would sometimes fail because the sensor was skipping some distance reading. But this was easily fixed by using inequalities <= or >=. Another issue is with the back plate used for resetting the system was getting hit by the pusher arms specifically for quadrant 4. The solution was to redesign the pusher arm such that it is symmetrical so that it would miss the back plate for both the top and bottom side. Lastly, resetting the system by measuring the backplate distance to be 30cm was unreliable. Instead, using a time delay to reset the position proved to be more effective.

Trial 3: Inconsistent scanning results

At this stage, the decoding is implemented to the sorting mechanism. Sorting is fully working as long as the character decoded is one of the barcodes assigned to one of the four quadrants. The main issue that remains was that for about six of out ten scans, one or more of the nine digits used to identify the glyphs were scanned incorrectly. One of the solutions was to pick characters that are from the four distinct different groups identified by the placement of the wide space. Another solution that greatly improved the scan was by 3D printing a block with the exact dimensions required to be at the optimum distance of the color sensor. Another benefit to this idea is that painted surface absorbs and reflect light better which gave more pronounced light intensity signals.

Trial 4: Final Design

The sorting and decoding is now fully functioning together. After testing 16 different unique glyphs from all four distinct groups, the success rate for the decoding and sorting was 13/16 or about 81% as seen in table 4.2. The characters used in the demo were ‘Y’, ‘7’, ‘D’, and ‘Q’. The demonstration will show the effectiveness of the robot as it was able to decode and sort all four characters.

Table 4.2: Final Design Test

Character	Pass or Fail	Reason for failure
W	Failed	2 out of 9 elements were wrong
8	Passed	
C	Passed	
T	Passed	
Q	Passed	
Z	Passed	
7	Passed	
B	Passed	
X	Passed	
5	Passed	
Y	Passed	
A	Passed	
N	Failed	1 out of 9 elements was wrong
1	Failed	1 out of 9 elements was wrong
O	Passed	
D	Passed	

F. *Conduct a longer test (say a few hundred trials) with the barcode simulator and report your success rate. How well does your decoding system behave?*

Using the barcode simulator, the success rate of running 300 trials was 100%. The decoding works as long as the barcode is for a single character. Any signals generated that contain multiple character will only output the first character. Outside the simulation, the only time that the decoding algorithm would fail is when the barcode is scanned incorrectly.

Table 4.3: 20 Sample from 300 trials of barcode simulator

Input	Output	Barcode	Pass or Fail
E	E	[2 3 1 3 2 4 1 3 1]	Pass
7	7	[1 3 1 4 1 3 2 3 2]	Pass
5	5	[2 3 1 4 2 3 1 3 1]	Pass
G	G	[1 3 1 3 1 4 2 3 2]	Pass
X	X	[1 4 1 3 2 3 1 3 2]	Pass
1	1	[2 3 1 4 1 3 1 3 2]	Pass
X	X	[1 4 1 3 2 3 1 3 2]	Pass
Y	Y	[2 4 1 3 2 3 1 3 1]	Pass
R	R	[2 3 1 3 1 3 2 4 1]	Pass
D	D	[1 3 1 3 2 4 1 3 2]	Pass
X	X	[1 4 1 3 2 3 1 3 2]	Pass
6	6	[1 3 2 4 2 3 1 3 1]	Pass
F	F	[1 3 2 3 2 4 1 3 1]	Pass
Z	Z	[1 4 2 3 2 3 1 3 1]	Pass
S	S	[1 3 2 3 1 3 2 4 1]	Pass
6	6	[1 3 2 4 2 3 1 3 1]	Pass
K	K	[2 3 1 3 1 3 1 4 2]	Pass
S	S	[1 3 2 3 1 3 2 4 1]	Pass
C	C	[2 3 2 3 1 4 1 3 1]	Pass
T	T	[1 3 1 3 2 3 2 4 1]	Pass

5 Conclusion

Overall, the mission of decoding barcodes and sorting it with a robot were accomplished. Once both subsystems were fully integrated, the final design test yielded a success rate of 81%. Several changes with the physical design were made in order to both successfully sort and improve the scanned results. The decoding algorithm was 100% successful when tested using the simulated scans. For further improvement with the scanning and decoding could be adding the most frequent misinterpretation into the list of barcodes in the database. Since for more than 70% of the failed scan, the incorrect readings are always for the same digit. In term of future design consideration, it would be excellent if more than one character can be deciphered by the decoding script.

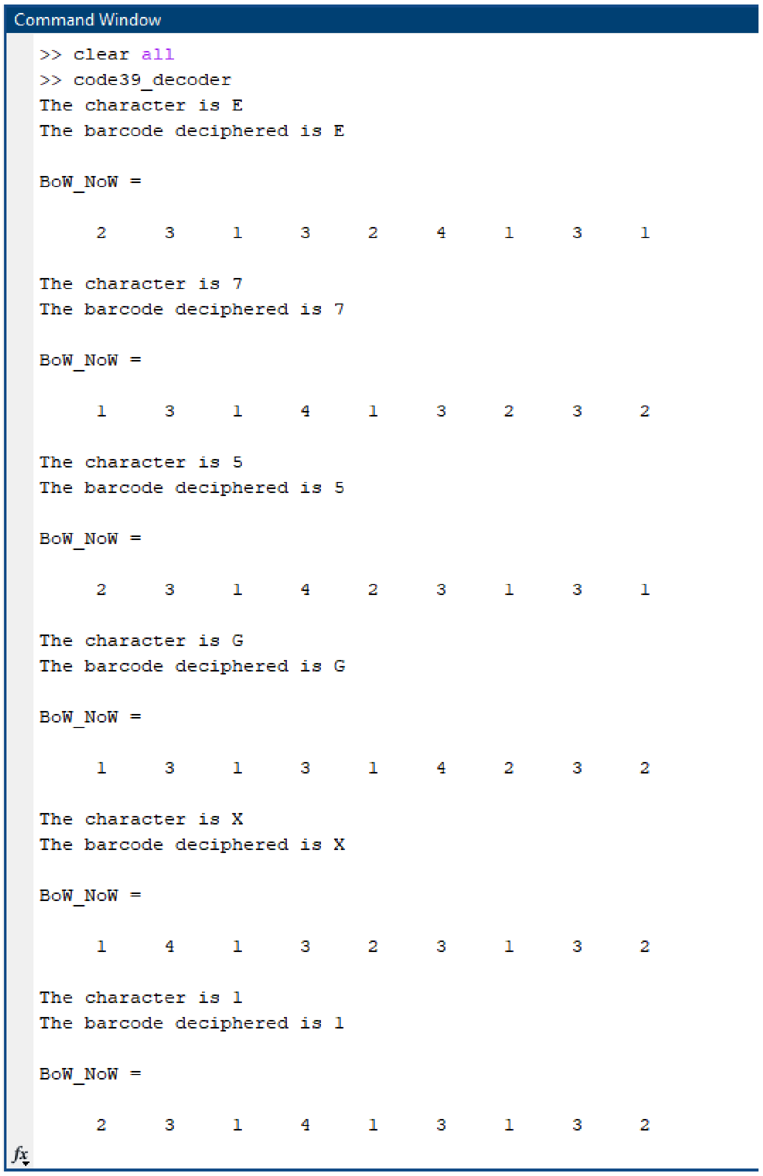
6 References

[1] J. Enright, "AER 627 Project 2," Ryerson University, Toronto, 2021.

[2] Wikipedia, "Code 39," 10 Feb 2021. [Online]. Available: https://en.wikipedia.org/wiki/Code_39. [Accessed 3 March 2021].

7 Appendix

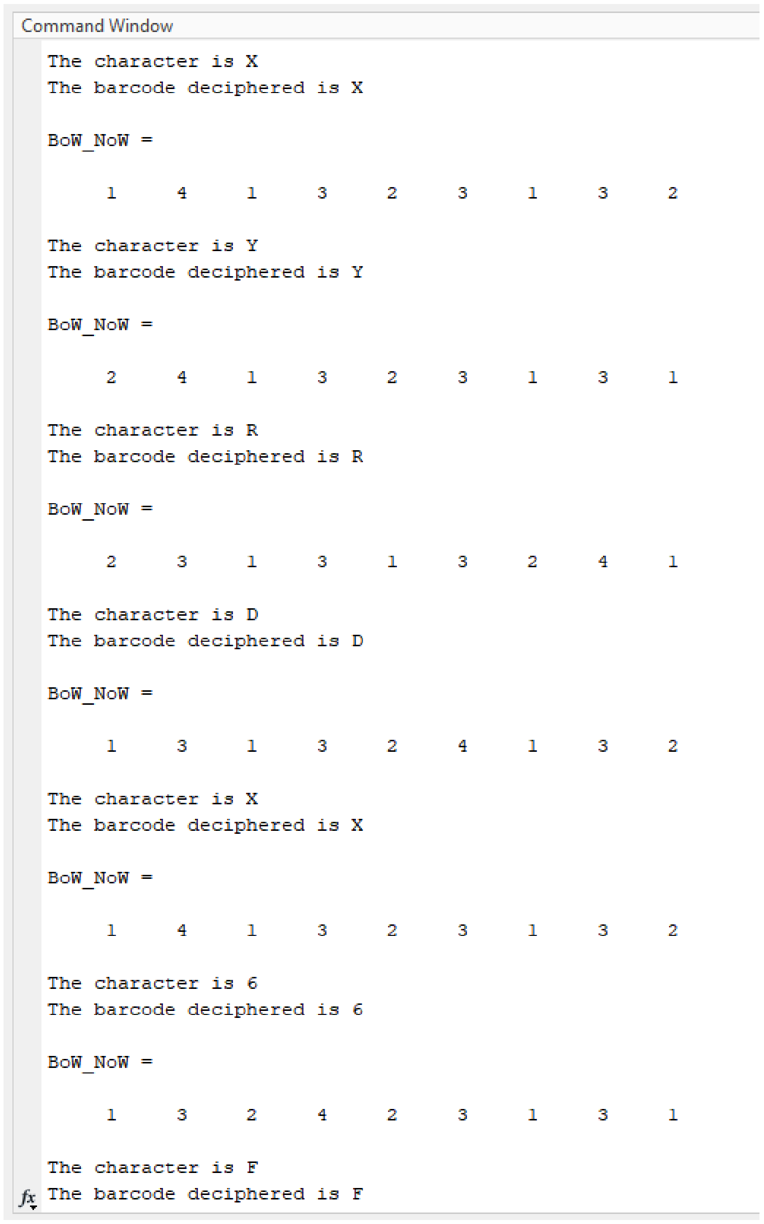
7.1 Barcode simulator test results



Command Window

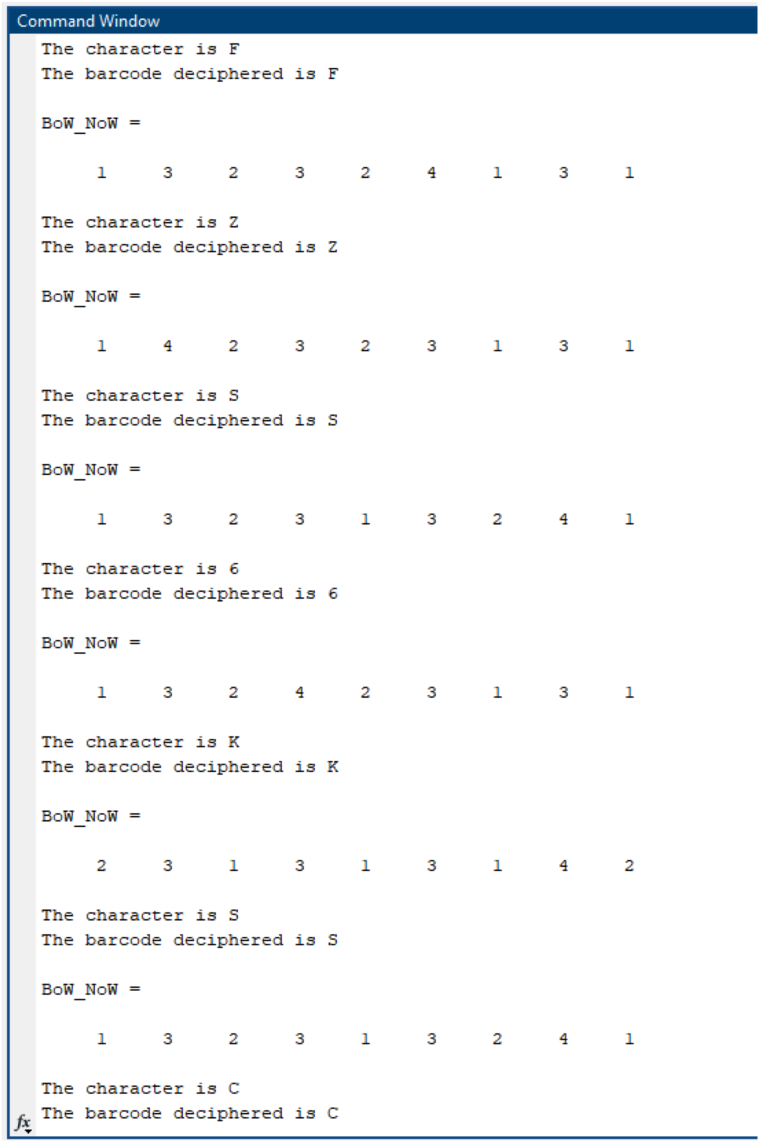
```
>> clear all
>> code39_decoder
The character is E
The barcode deciphered is E

BoW_NoW =

     2     3     1     3     2     4     1     3     1

The character is 7
The barcode deciphered is 7

BoW_NoW =

     1     3     1     4     1     3     2     3     2

The character is 5
The barcode deciphered is 5

BoW_NoW =

     2     3     1     4     2     3     1     3     1

The character is G
The barcode deciphered is G

BoW_NoW =

     1     3     1     3     1     4     2     3     2

The character is X
The barcode deciphered is X

BoW_NoW =

     1     4     1     3     2     3     1     3     2

The character is 1
The barcode deciphered is 1

BoW_NoW =

     2     3     1     4     1     3     1     3     2
```

Figure 7.1: barcode simulator test results



```
The character is X
The barcode deciphered is X

BoW_NoW =

     1     4     1     3     2     3     1     3     2

The character is Y
The barcode deciphered is Y

BoW_NoW =

     2     4     1     3     2     3     1     3     1

The character is R
The barcode deciphered is R

BoW_NoW =

     2     3     1     3     1     3     2     4     1

The character is D
The barcode deciphered is D

BoW_NoW =

     1     3     1     3     2     4     1     3     2

The character is X
The barcode deciphered is X

BoW_NoW =

     1     4     1     3     2     3     1     3     2

The character is 6
The barcode deciphered is 6

BoW_NoW =

     1     3     2     4     2     3     1     3     1

The character is F
The barcode deciphered is F
```

Figure 7.2: barcode simulator test results


Command Window

```
The character is F
The barcode deciphered is F

BoW_NoW =

     1     3     2     3     2     4     1     3     1

The character is Z
The barcode deciphered is Z

BoW_NoW =

     1     4     2     3     2     3     1     3     1

The character is S
The barcode deciphered is S

BoW_NoW =

     1     3     2     3     1     3     2     4     1

The character is 6
The barcode deciphered is 6

BoW_NoW =

     1     3     2     4     2     3     1     3     1

The character is K
The barcode deciphered is K

BoW_NoW =

     2     3     1     3     1     3     1     4     2

The character is S
The barcode deciphered is S

BoW_NoW =

     1     3     2     3     1     3     2     4     1

The character is C
The barcode deciphered is C
```

Figure 7.3: barcode simulator test results

7.2 Sorting Robot test run and demonstration results

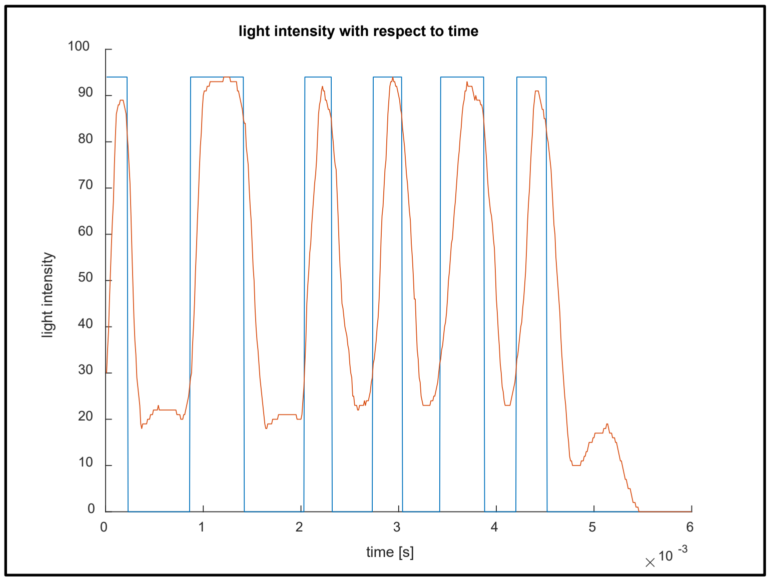


Figure 7.4: W failed test results

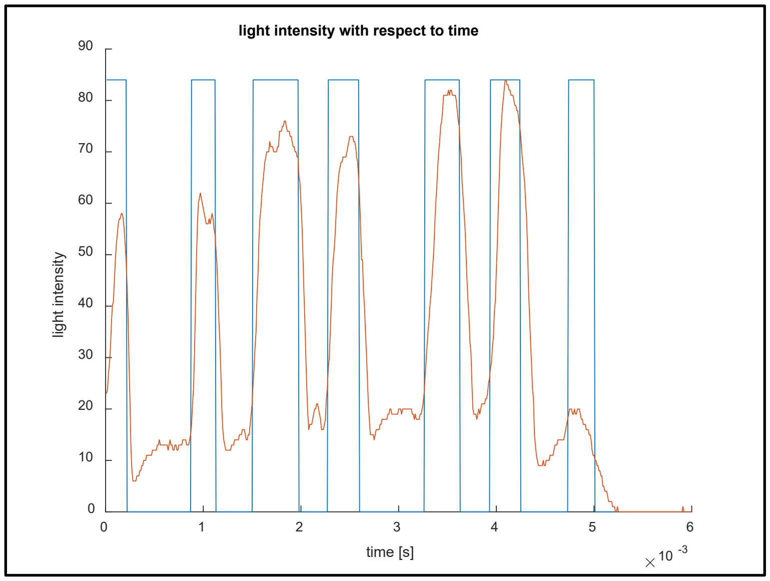


Figure 7.5: 8 passed test results

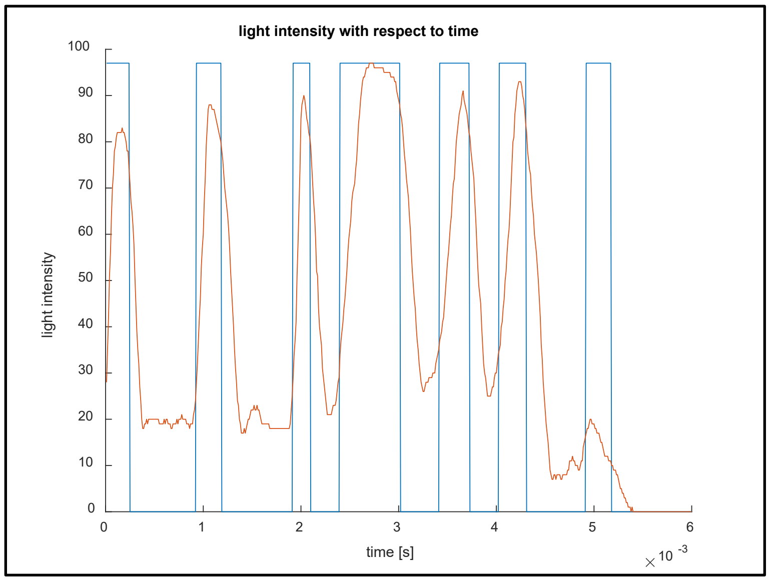


Figure 7.6: C passed test results

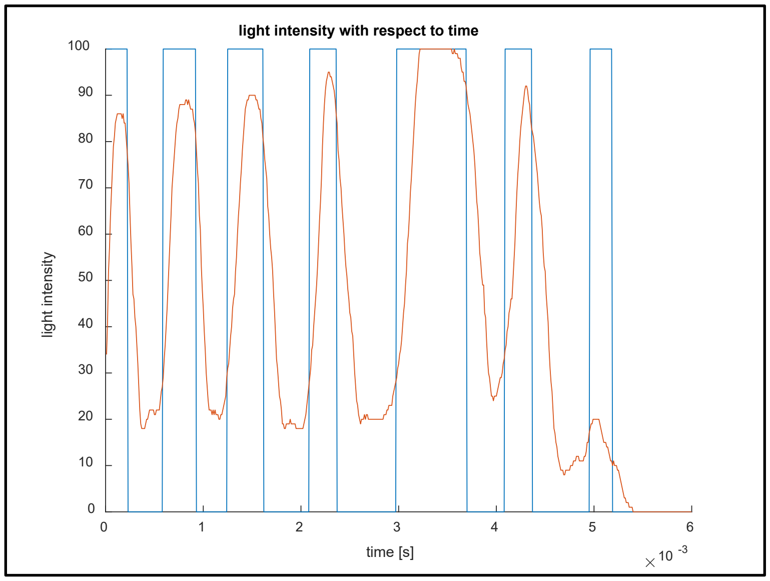


Figure 7.7: T passed test results

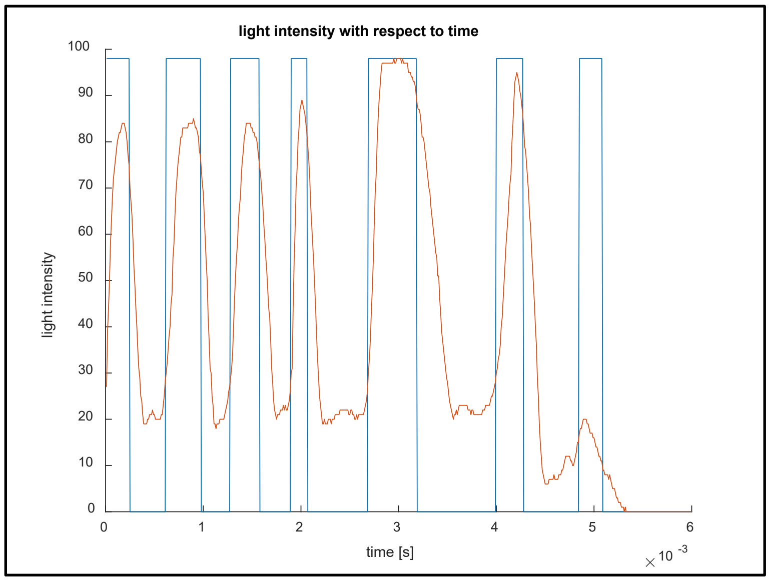


Figure 7.8: Q passed test results

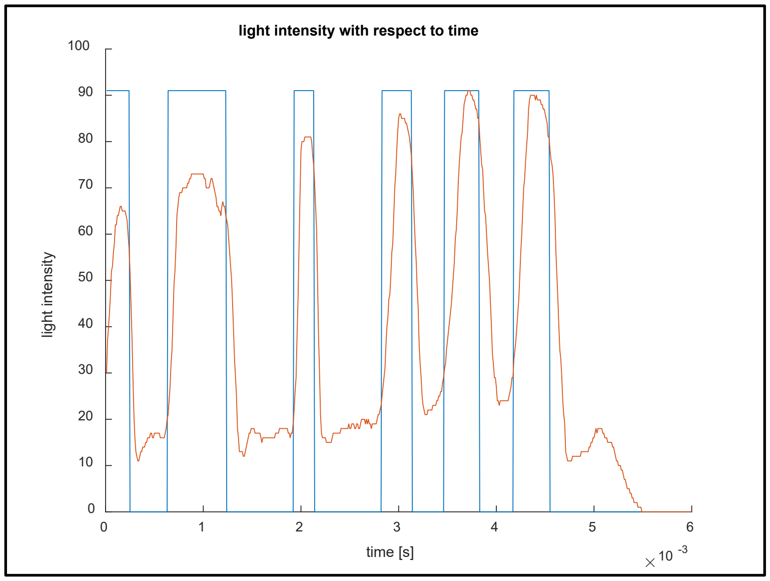


Figure 7.9: Z passed test results

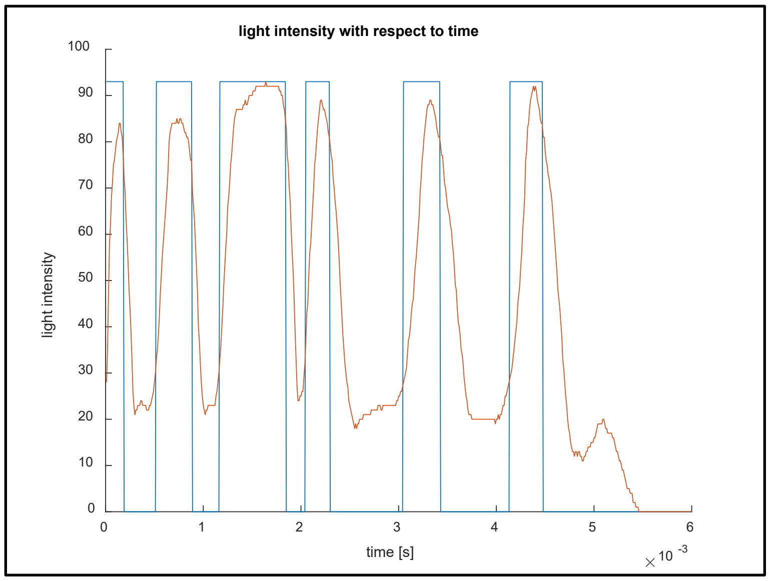


Figure 7.10: 7 passed test results

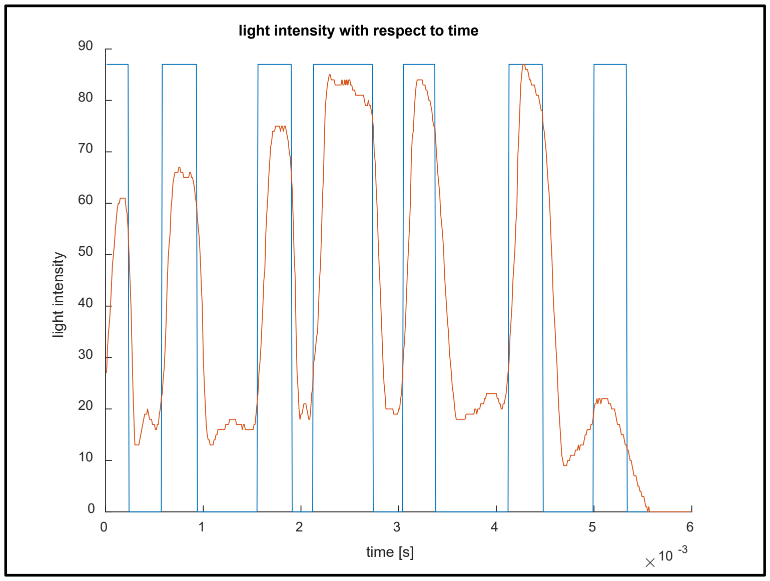


Figure 7.11: B passed test results

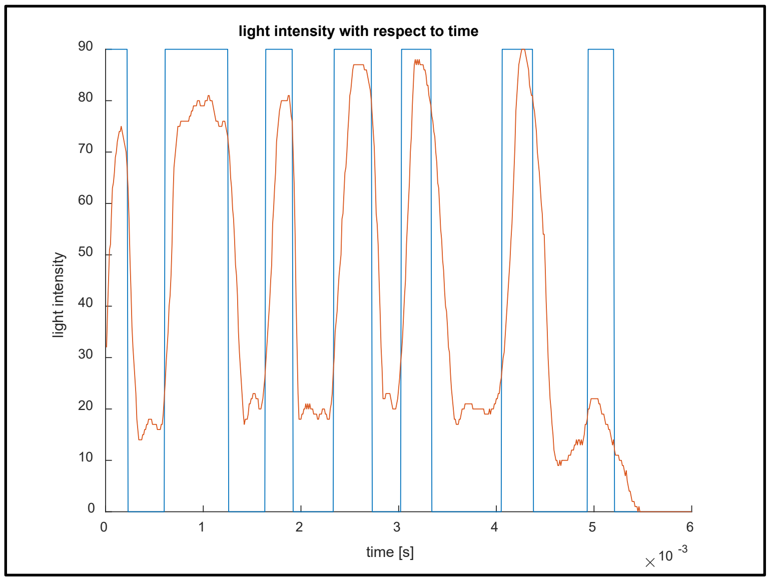


Figure 7.12: X passed test results

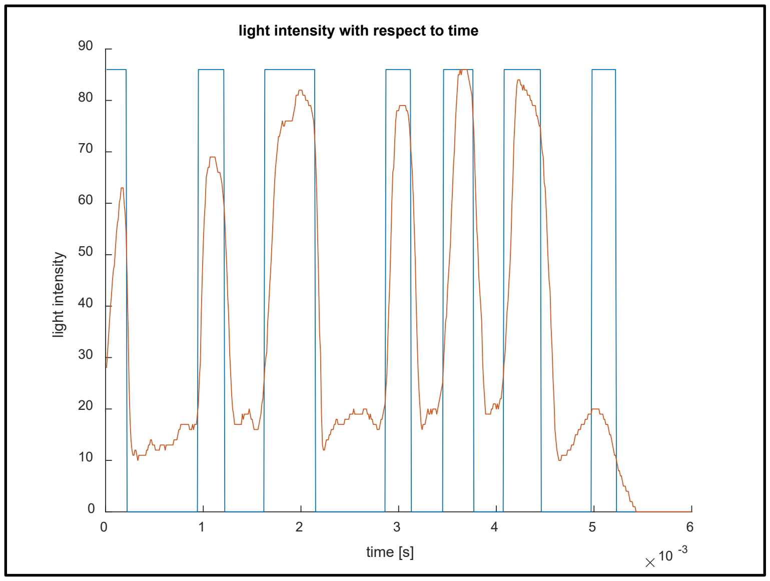


Figure 7.13: 5 passed test results

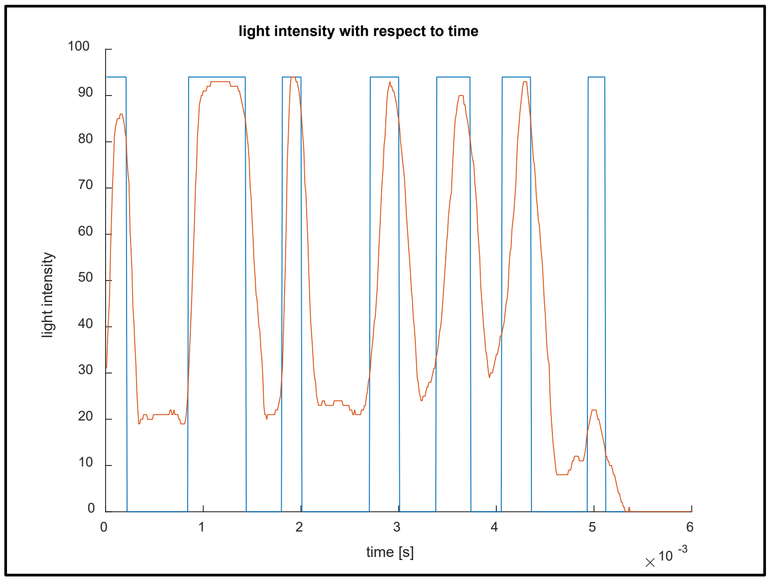


Figure 7.14: Y passed test results

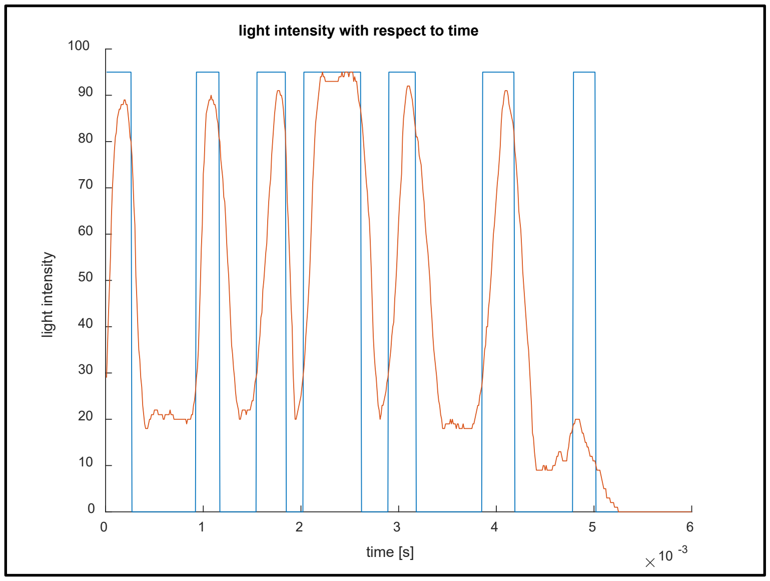


Figure 7.15: A passed test results.

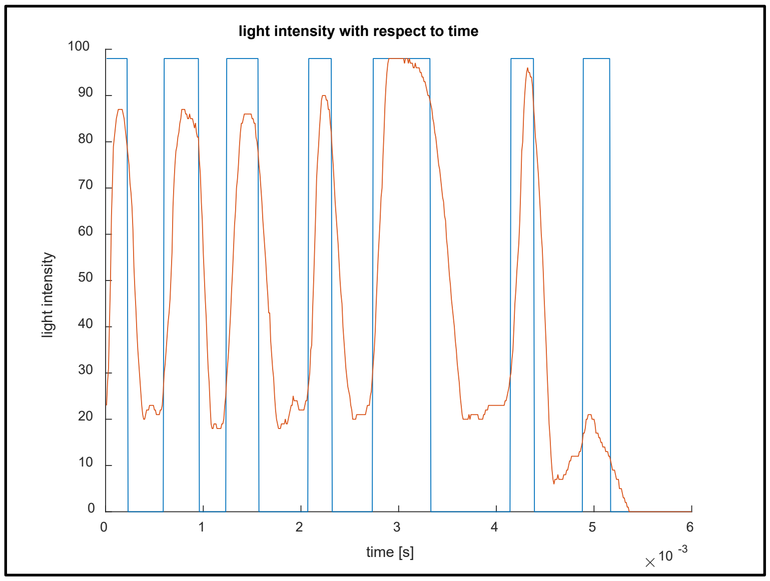


Figure 7.16: N failed test results.

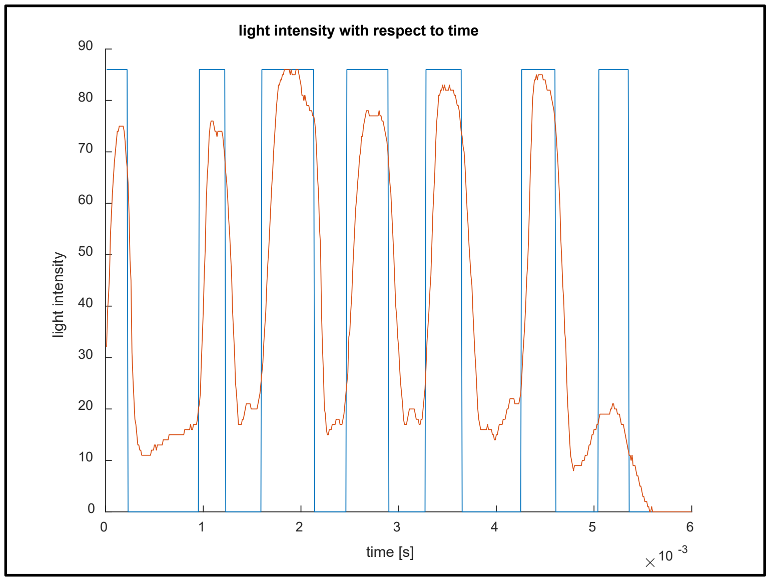


Figure 7.17: O passed test results

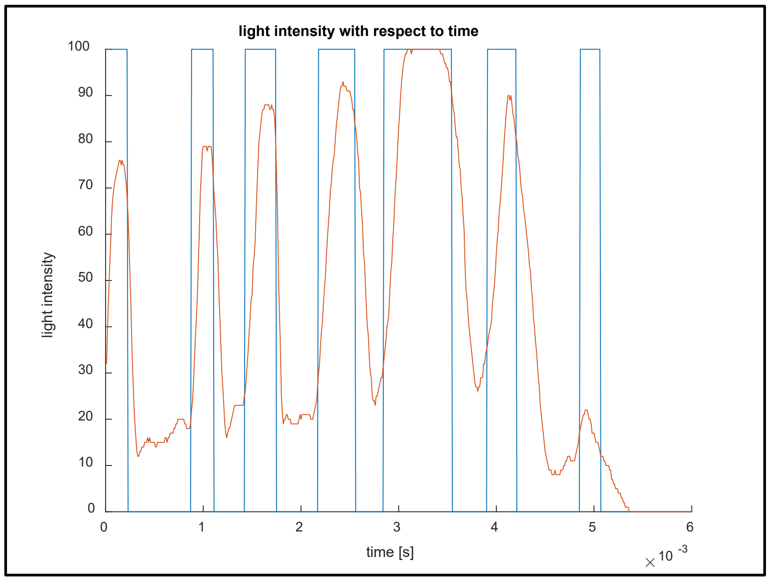


Figure 7.18: D passed test results

7.3 Final Design Drawings

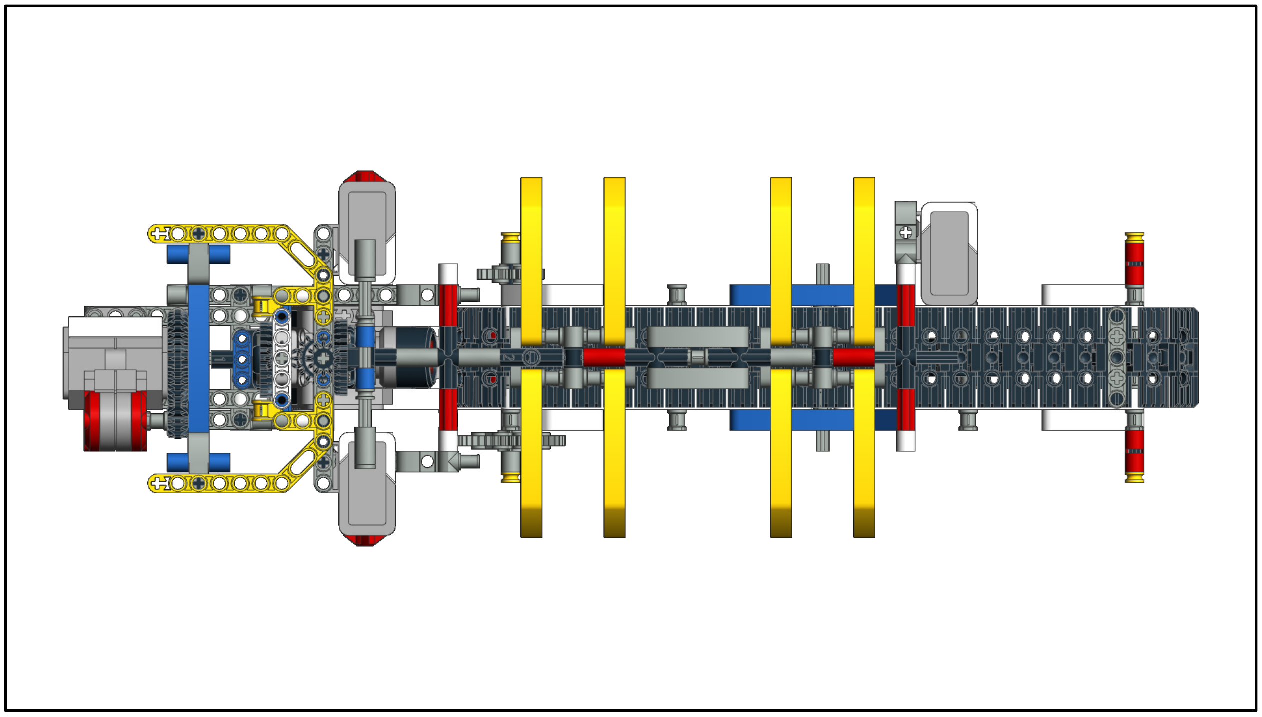


Figure 7.19: Final Design top view

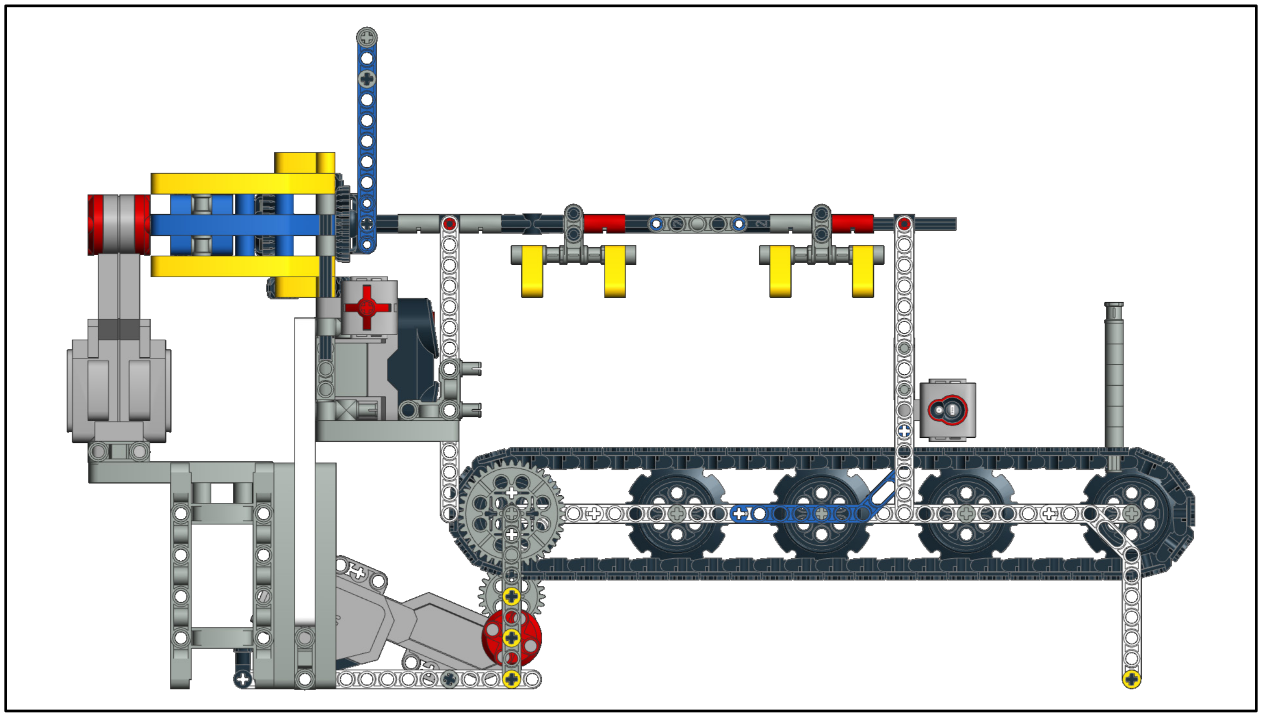


Figure 7.20: Final Design front view

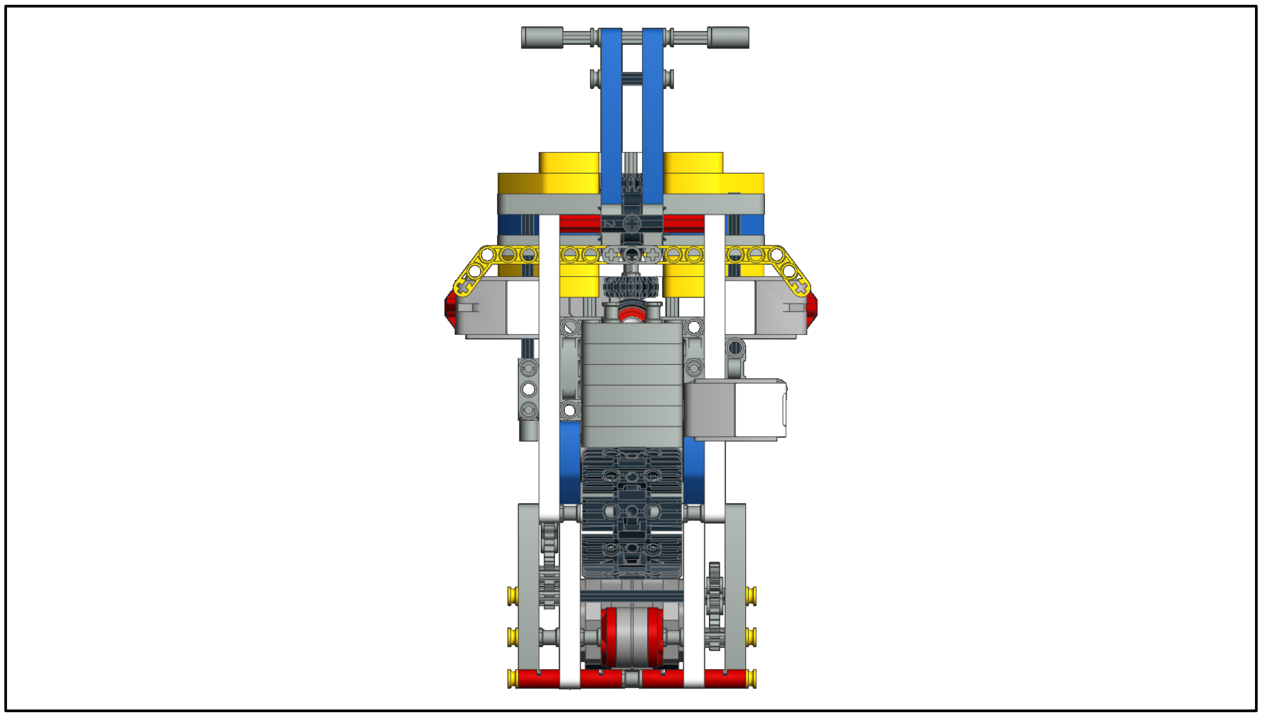


Figure 7.21: Final Design right view

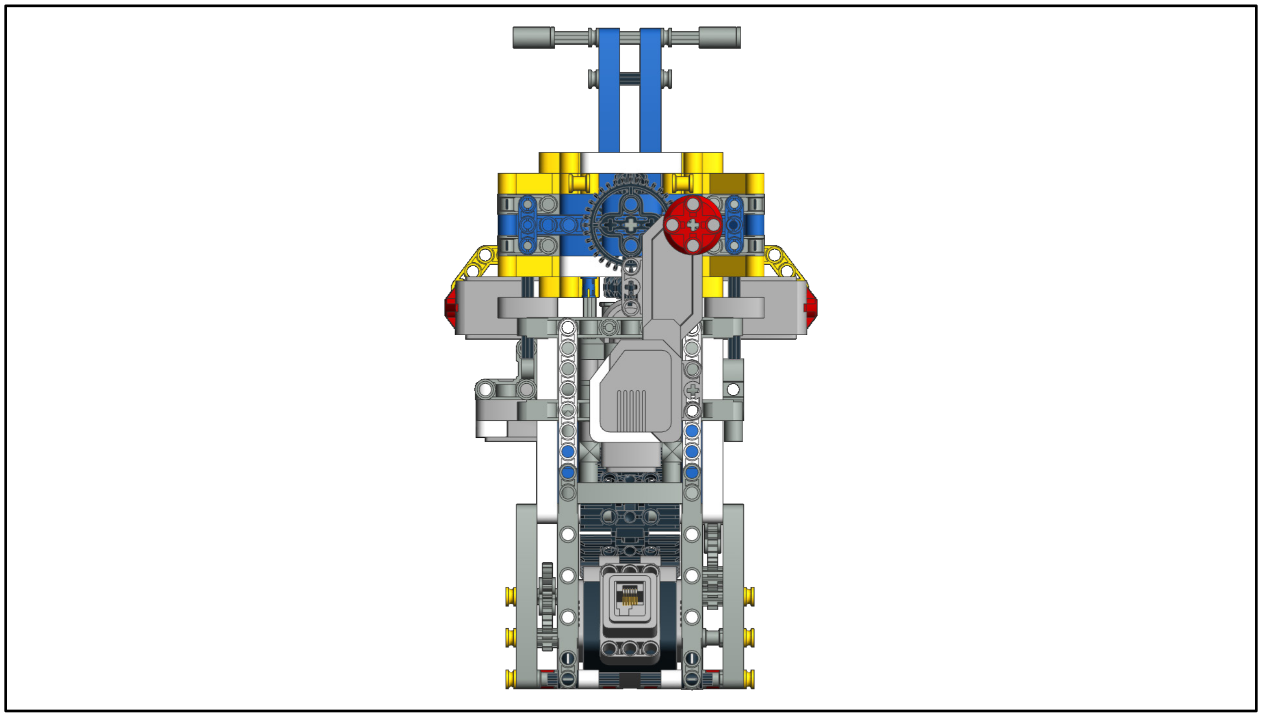


Figure 7.22: Final Design left view