Smart Pick Bins for Reinforcement Steel Installation in Silver Line

Nelson Dong	Benjamin Mazel	Steven Viola
nelsondong2005@gmail.com	ben.mazel@gmail.com	steve.viola908@gmail.com

New Jersey Governor's School of Engineering and Technology
25 July 2014

Abstract

Modern manufacturers break down the manufacturing process into smaller functions, creating separate pieces that combine to form a complete product. Many of the stations require the worker to make a decision as to which part to use because a single assembly line may produce more than one type of product, Often, a worker may make a mistake and use the wrong part, meaning the product needs to be discarded.

To solve this problem at the steel reinforcement station at Silver Line Window Company, this project aimed to create a smart pick bin system that incorporates storage bins and a barcode scanner. The worker can use the scanner to scan a barcode on the sash of a window and the bin will indicate which steel reinforcement bar to use. By combining an Arduino microcontroller with LED lights, an LCD screen, and a handheld barcode scanner, a system is created that increases productivity and decreases worker error.

1. Introduction

One primary objective of factories is to streamline the manufacturing process in order to save time and money. This may entail cutting corners to downsize costs and maximize profits, or investing in new machinery that expedites different processes and can speed up production. By automating different processes of manufacturing, companies are also able to reduce the brunt of the labor for workers. The advent of increasingly advanced technology makes the lives of laborers easier and serves as an efficient way of increasing productivity.

Silver Line Window Company hopes to achieve this through the implementation of smart pick bins for reinforcement steel installation. The ultimate purpose of these efforts is to improve the quality of the windows produced by Silver Line. Mistakes as a consequence of human error are frequently made and can result in faulty windows. By using the smart pick bin system, the margin for human error can be reduced, resulting in increased productivity and a consequent increase in the standard of quality at Silver Line as well.[1]

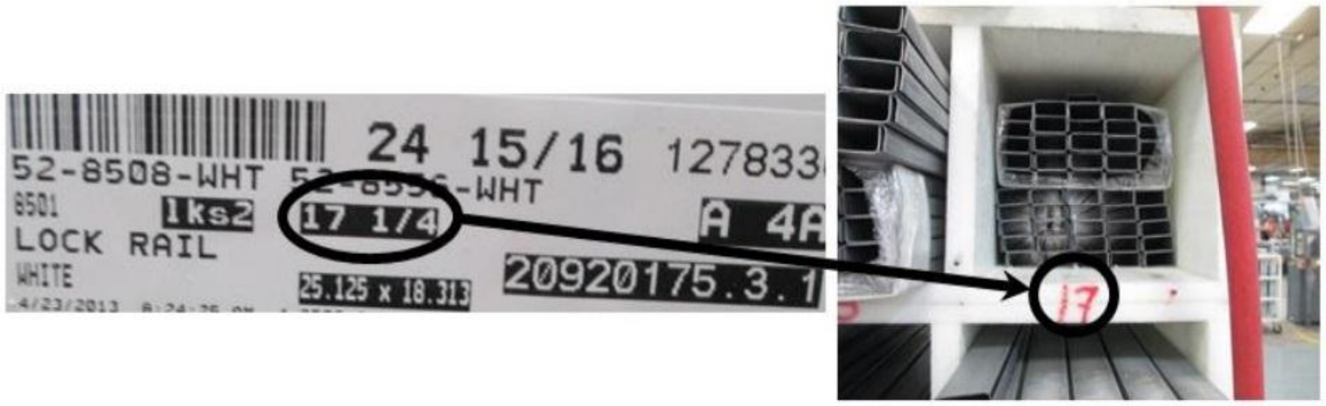


Figure 1. *A barcode label pasted on a window frame indicates the length of steel reinforcement bar needed, as well as the corresponding bin from which the bar must be selected.*

2. Background

At the current Silver Line headquarters in New Brunswick, New Jersey, the reinforcement steel installation process is conducted by first reading the barcode label of a vinyl window frame edge and then matching the width number to a corresponding number written on the steel reinforcement bar bins (Figure 1). The purpose of the smart pick bin system is to reduce the chances of picking a steel reinforcement bar from the wrong bin. Smart pick bins, also known as pick-to-light systems in other factories and warehouses, have been known to increase productivity by 25 to 50 percent. Pick to light Systems[2], Lightning Pick Technologies[3], and SpeasTech[4] have developed pick-to-light systems to use in factories. Their pick-to-light systems have been extremely effective, in part because they require little to no training time, as well as reducing worker error to .01%.[5] SpeasTech reports that it has had employees that have performed 25 million picks with no errors.[6] Some companies achieve even greater results and most see a significant increase in efficiency and picking accuracy in just one year.

2.1 The Window Manufacturing Process

At Silver Line, the window manufacturing process is compartmentalized into sections. Windows consist of two sides; the fixed side and the sash side, which helps them slide open (Figure 2). The factory is therefore split into two parts, each running a process that builds one side of the window. Vinyl powder is heated and extruded into window edges, which is then cooled by a water system. The edges are then assembled and melted together and punched so that the steel reinforcement bars and locks can be inserted into the window frame. Glass is prepared and glued into place.

Steel reinforcement bars provide structural stability to vinyl windows, which cannot withstand strong winds and other atypical forces.

2.2 Takt Time

What determines the viability of the smart pick bin system is whether or not it can increase the average quality of windows while maintaining takt time. Takt time is the ideal amount of time it takes to complete one cycle of any process in the factory. Cycle times are easily monitored because each process at Silver Line, including the installation of reinforcement steel, is determined by one-piece flow. One-

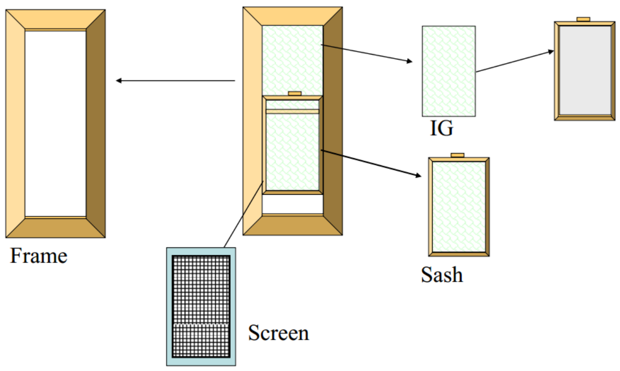


Figure 2. *The components of a window. "IG" stands for insulating glass. The sash is the movable panel of the window.*

piece flow means that a worker focuses on performing his/her function, one piece at a time. This keeps manufacturing moving at a relatively constant pace.

Takt time is taken in the context of the reinforcement steel installation process. The respective takt time for each process in window manufacturing is the calculated value that maximizes worker productivity and profits. If workers perform more quickly than the takt time, they will overproduce and materials will be wasted. If they do not meet the takt time, they will fall behind the rest of the factory and will delay the completion of windows.

An LCD screen displays the cycle time of workers based on the time between barcode scans. With the ability to view their own cycle times, workers can compare their pace with the constant takt time in order to maximize efficiency while reducing overproduction.

2.3 Importance of Steel Reinforcement Bars

Steel reinforcement bars maintain the structural stability of each window with different design pressure ratings depending on the application. Design pressure is the official scale that corresponds to success of a window in high wind scenarios. Windows at risk of greater impacts and stronger storms will use a higher design pressure rated steel reinforcement bar so that the window can withstand a greater force before breaking. This is especially important in vinyl windows because of their lesser strength when compared to other types of windows. In extreme conditions, these windows can crack when a strong force is applied. Due to the relatively flexible material, warping and expansion can also become a problem. By adding in proper steel reinforcements, a much safer and more durable window is created.

The reinforcement bar is also inserted into the vinyl window frames that the lockers and keepers latch onto. These latches act as locks on the window to prevent any break-ins and ensure the window is securely closed. The steel provides a better mounting point for the lock and improves the window's resistance from outside forces such as strong winds. It also helps to prevent break-ins through the window. Reinforcements can be placed in intermediate jambs, and can stretch throughout the entire sash instead of just the rails to withstand even greater impacts.

Although vinyl is created to withstand exposure to the elements, such as fire and impact, it needs steel reinforcement to prevent softening from radiant heating.[7] If improperly installed, a window is at risk of failure. The quality and strength of the window decreases if the incorrect steel reinforcement bar is inserted into the window. Once inserted into the sash of the window, the bars are secured and are difficult to remove. If the mistake is realized further into the production process, the window must be broken to retrieve the steel. This wastes resources and any previous labor is lost.

The reinforcement bars inside the vinyl windows are extremely important and thus quality control in this area is vital. The addition of smart pick bins into the reinforcement steel lines would prevent mistakes, save time and money, and create an overall more consistent and higher quality product.

2.4 Arduino

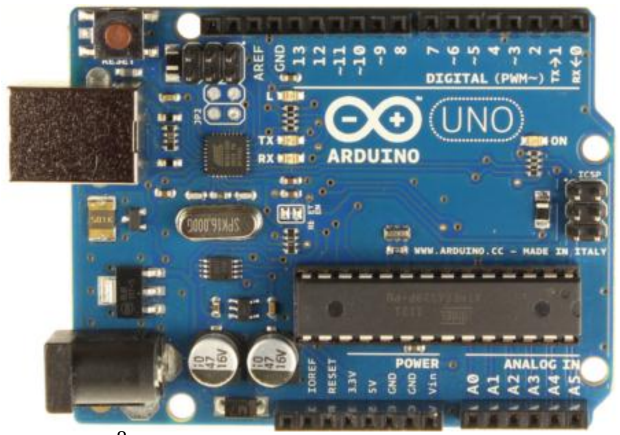


Figure 3.[8] *The Arduino Uno board. Digital pins, USB jack, (upper left), power jack (lower left).*

Arduino is a single-board microcontroller that facilitates the interaction between computers and peripheral devices (Figure 3). The software controls the actions of the hardware through a special programming language. Arduino compiles the code into virtual machine code and the central processing unit (CPU) carries out the functions of the code. The smart pick bins system is based on the Arduino platform. The Arduino uses methods specific to the Arduino programming language to interface with connected devices. The language works similarly to C++, but utilizes Java syntax.

The Arduino board comes with a USB and power cable as well as several digital pins. These pins allow for the attachment of interactive hardware. The version being used in this project is Arduino Uno, along with a full sized breadboard and a miniature breadboard for the LCD screen. By using the single-board microcontroller, the programming of barcode scanners, LEDs, HIDs and other peripheral devices to perform a predetermined function can all be connected through a single programmable class. The standard Arduino Uno did not come equipped with a USB port through which the barcode scanner can communicate data back to the Arduino, so a USB Host Shield was required. This shield added the necessary USB port to the Arduino while still allowing most of the original pins to function. Pre-assembled boards are available, but the board used required pin headers to be soldered onto the pins before a connection could be made. After orienting the necessary hardware with Arduino software allowed the scanner to read the labels on window frames, and light up a bin with the corresponding steel reinforcement bar to save time and effort for workers.

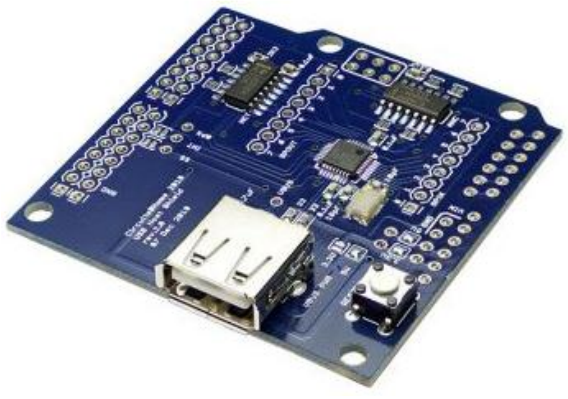


Figure 4. *The unassembled USB host shield.* [11]

3. Methods / Experimental Design

The creation of smart pick bins for reinforcement steel installation is centered around the use of a barcode scanner to read information from barcode labels, and an Arduino program to convert the scanned information into an action. The action is, of course, lighting up bins with steel reinforcement bars of varying sizes so workers can more easily insert the correct reinforcement bars into window edges. With the smart pick bin system, the barcode will be scanned, the frame width inputted, and the bar length retrieved. A single bin like the one shown in Figure 5 will then illuminate.

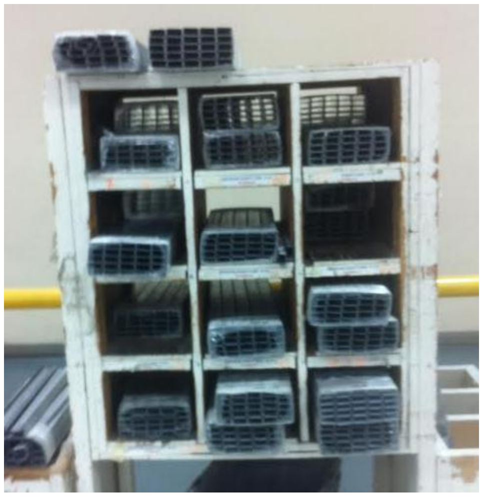


Figure 5. *The current workstation at the steel reinforcement line.*

In order to control any peripherals using the breadboard of the Arduino, said peripherals need to be connected to a computer through the Arduino board, which essentially serves as the segway between the compilation of code and the performance of programmed tasks. LED lights connect to the breadboard through wires and the breadboard to the Arduino board through wires attached to pins. The Arduino board has only 14 pins, making the breadboard a necessary component in order to reduce the amount of pins used.

Ultimately, the prototype design consists of three bins, each of which holds 15 ¼, 17 ¼ or 19 ¼ inch steel reinforcement bars. LED strips line the top of the bins, which are stacked horizontally, and light up with a single color. The provided prototype "bins" are also few in number: five bins are connected horizontally, with four being allocated to the steel reinforcement bars of difference sizes and the other dedicated to housing the Arduino board.

After combining all of these components the smart pick bin system is complete. A barcode can be scanned by an employee and the Arduino will light the bin with the correct sized reinforcement. This prototype, while small scale, accurately represents the same steps that are necessary to construct the full size system. (Figure A1)

3.1 Programming LEDs

Since smart pick bins require LEDs to light up the steel reinforcement bar slots corresponding to window frame width, the first step was to test the Arduino programming platform with the LED strips that are used in the prototype bin. These LED strips work by sending patterns to the lights which contain an internal processor. The light processor then takes the pattern and converts it into a signal that turns individual lights on or off. The actual LEDs for the smart pick bins are LED strips, and are more pin-efficient than the LED bulbs. Each strip uses only one pin for data, one for power, and one for ground (Figure 6).

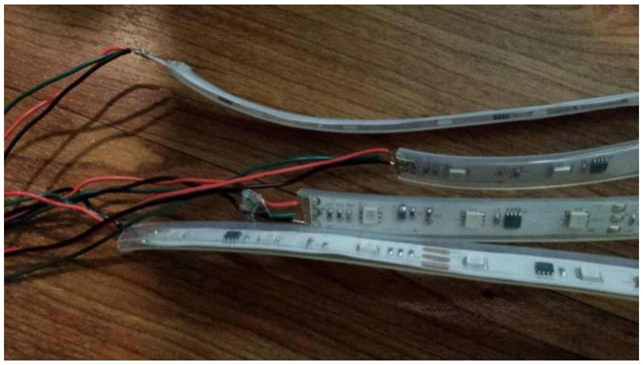


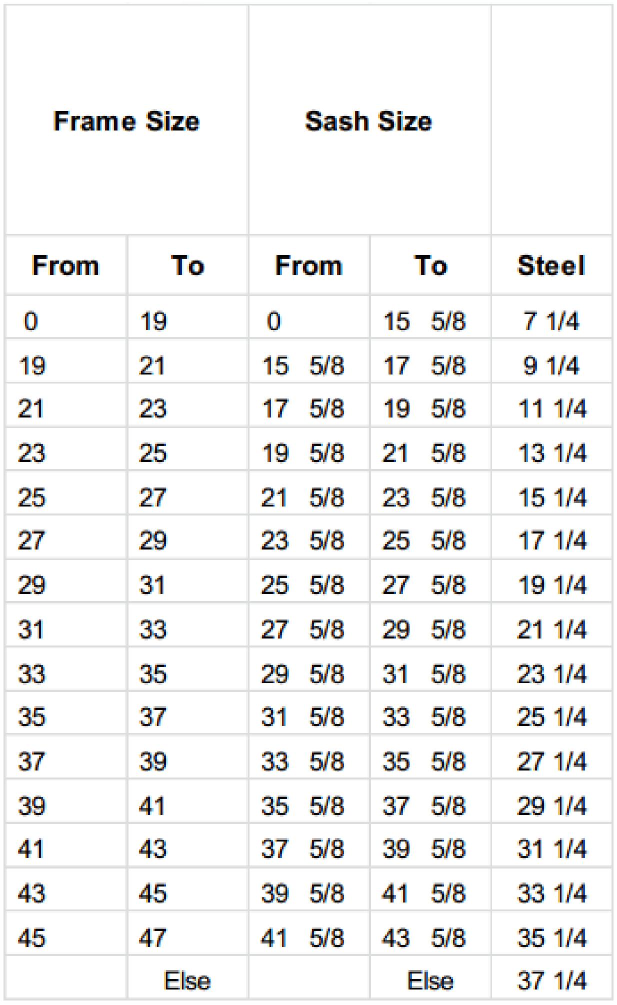
Figure 6. *The LED lights before they are mounted to the prototype.*

3.2 Interfacing the Barcode Scanner with the Arduino

Since the Arduino board and USB Host Shield need to be connected for the barcode scanner to properly function, the Host Shield was soldered in order to attach the pins. Rows of digital pin headers had to be soldered to the Host Shield, with the slots facing upward and the metallic components facing downward. The Arduino board lies on the bottom of the setup with the metal portion of the headers inserted into the pins from the Arduino board.

With the components soldered in place,

the code could finally be compiled and executed. For each window frame and its width, there is a corresponding length that indicates which bin the steel reinforcement bars should be drawn from (Figure 7). The program uses a row of *if* and *else if* statements to set a variable *x* to different reinforcement steel lengths, returning the correct length, *x*, in the end. In a second set of *if* statements, the LED strips which indicate different steel reinforcement bar lengths are activated when a reinforcement steel length is inputted. Based on each input, a different LED strip will light up (Figure A4).



Frame Size		Sash Size		
From	To	From	To	Steel
0	19	0	15 5/8	7 1/4
19	21	15 5/8	17 5/8	9 1/4
21	23	17 5/8	19 5/8	11 1/4
23	25	19 5/8	21 5/8	13 1/4
25	27	21 5/8	23 5/8	15 1/4
27	29	23 5/8	25 5/8	17 1/4
29	31	25 5/8	27 5/8	19 1/4
31	33	27 5/8	29 5/8	21 1/4
33	35	29 5/8	31 5/8	23 1/4
35	37	31 5/8	33 5/8	25 1/4
37	39	33 5/8	35 5/8	27 1/4
39	41	35 5/8	37 5/8	29 1/4
41	43	37 5/8	39 5/8	31 1/4
43	45	39 5/8	41 5/8	33 1/4
45	47	41 5/8	43 5/8	35 1/4
	Else		Else	37 1/4

Figure 7. *A part of the conversion table, which converts between frame width and bar length.*

Connection between the barcode scanner and the microcontroller is achieved by utilizing the USB Host Shield 2.0, an Arduino shield which provides a USB port for the Arduino. The USB Host Shield is controlled using the USB Host Shield 2.0 library, which adds methods for receiving and sending output through the USB port into the Arduino. These methods were used to obtain and analyze barcode scan data. The barcode scanner acts as a keyboard, and scanning the barcode is effectively equivalent to typing in the numbers represented by the barcode in rapid succession.

The primary driving methods behind the collection of this data are the methods *OnKeyDown* and *OnKeyPressed.* These functions read each key press of the scanner and return the ASCII character code of the digit represented in the barcode. To collect these individual key presses the Arduino records the time between each input. If the key presses occur in rapid succession, they are strung into a single array of *char*s. *Char* is a variable type which represents a single character of text i.e. a letter, a digit, or a symbol. At this point, the Arduino stores an array of characters, which represent the ASCII codes for the data separated by zeroes. These zeroes are then removed and the remaining character codes are converted into numerical values. These values are then put into an array representing the true value of the barcode. This value is the size of the window frame in inches. The barcode value is then analyzed by the program in order to convert it into a corresponding reinforcement steel size. To achieve this, the Arduino separates the first six digits of the barcode which correspond to the width of the window. This width is then run through a conversion table which returns the length of reinforcement steel needed. (For further clarification, see Figure A2 and A3 in the Appendix).

3.3 Interfacing the LCD Screen with Arduino

The LCD (Liquid Crystal Display) screen displays average cycle time in the smart pick bin system. In order to set up the LCD screen, a diagram was first drawn up using free software called Fritzing (Figure 8).[10] The screen connects to the Arduino using eight wires and it controls the LCD screen through the built-in LiquidCrystal library. This allows the user to instantiate the LCD screen as a LiquidCrystal object and indicate which pins the screen will use. In addition, the library grants the user the ability to use methods such as lcd.begin() and lcd.print() to display information on the screen, which can show up to 32 characters of text. The official Arduino guide to LiquidCrystal was used to implement the necessary methods into the program.[9]

The LCD screen, being somewhat rudimentary, also needed soldering to allow wires to connect the Arduino, USB Host Shield, and the screen together. Soldering allows a row of pins to become attached to the upper edge of the screen. The pins dig into a miniature bread board, which is connected to USB Host Shield and, by extension, the Arduino board, (Figure 9). Whenever a barcode is scanned, a counter is initiated that keeps track of the seconds between the current barcode scan and the next barcode scan. The value is returned in seconds to inform the worker of his/her proximity to the takt time.

The counter is then reset upon another barcode scan so that a new cycle time can be counted. However, the new cycle time is averaged with every other recorded cycle time and the LCD screen displays this average cycle time for the steel reinforcement process. One cycle refers to the reinforcement of a single window and the use of one reinforcement bar. To achieve this, the Arduino stores several values, which are necessary to determine the average cycle time. The LCD also resets the average and begins the count again if the time between two scans is much larger than normal. This accounts for the production time lost during lunch break or on a change of shift. The reset allows a worker to track their current cycle time accurately, without false data interfering with the average.

The actual program stores the time at which the previous scan was performed. The

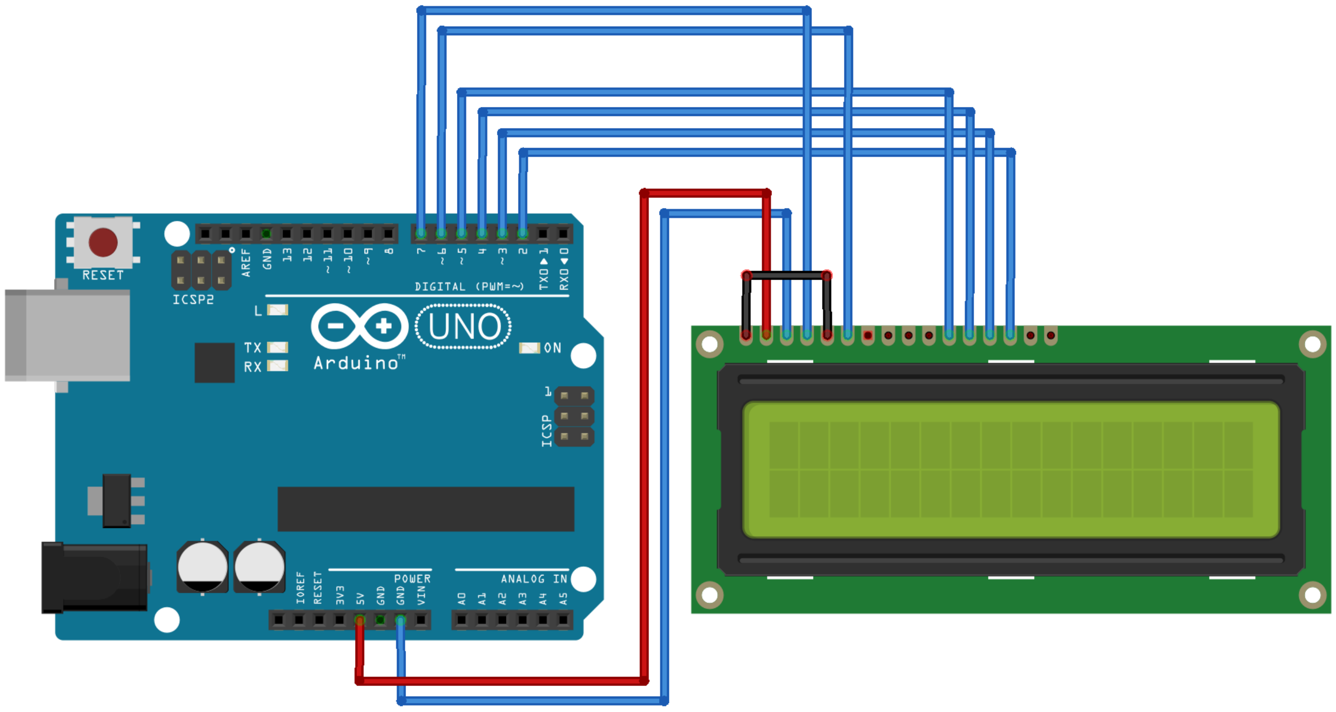


Figure 8. *The layout of the circuitry of the Arduino board and the LCD screen. This diagram shows the wires being connected to pins on the Arduino board itself (without a breadboard). For this project, a USB Host Shield was needed since the Arduino was not USB compatible by itself.*

time is stored as a value of milliseconds from the time the Arduino was turned on. It also uses the millis() function of the Arduino to determine the time of the current scan; subtracting these two values yields the total time between scans or the duration of a single cycle. The Arduino stores the number of scans performed as well as the current average cycle time. Once a new scan is performed, a new average cycle time needs to be calculated. To do this, the Arduino multiplies the average time by the number of scans, yielding the time for all of the previous scans combined and, then adds the time of the current cycle. Finally it divides the total time by the new number of scans to find the new average cycle time. This value is then displayed on the LCD screen.

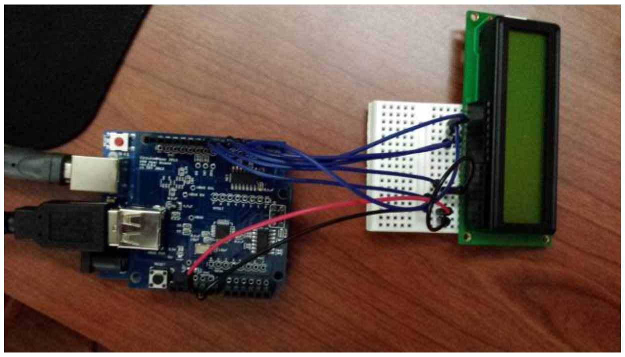


Figure 9. *The Arduino and USB host shield connected to the LCD screen. The average cycle time is displayed.*

3.4 Constructing the Prototype

The bins are constructed of flame-retardant wood and consist of three spaces for steel reinforcement bars, one space for the Arduino board and USB Host Shield, and one vacant space. The bins are aligned horizontally and each was constructed with a LED strip lining the top. The wires from the strips are connected to the pins from the board in the last space on the bins. The LEDs are hot glued to the top of the bins and stretch from end to end.

The final product will have a different method of attaching these LEDs. In a system designed for daily use, holding the LED lights against the top of the bin with metal brackets will provide a much stronger hold that can withstand the abuse of factory use. Metal brackets will also allow for easier replacement of the LEDs after their possible failure. However, this approach is impractical in the prototype. Because of its small size, it is impossible to mount brackets inside the bins. In the full scale final set up, the bins will be much larger and mounting a bracket will be a simple task. It will take two brackets, one at each end of the LED strip to effectively hold the lights. The wiring will remain the same as the prototype with three colored wires for each LED strip braided together for neatness and connected to the Arduino board, which hides in the very last slot in the prototype.

Housing the Arduino and breadboard inside an empty bin is both convenient and safe for the board itself. In a factory setting, however, it is impractical to sacrifice a full size bin for such a small apparatus. In use, the board will be mounted to the side in a protective case, freeing the main bin for more reinforcement bars while still allowing easy access to the Arduino and attached barcode scanner. The LCD screen will also be mounted on the side or top of the bin to prevent bin obstruction and allow easy viewing. The LED lights will remain inside the bins in the factory because of their small profile and the ability to mount them at the top of the bin, out of the way of any steel. The LEDs also have a lifespan of over 1,000 hours, meaning they will rarely need to be replaced.

3.5 Cost Analysis

When considering any modifications in a factory, it is important to consider the cost to implement versus the return on investment. Overall, the setup used in the prototype costs $235 with over half of this coming from the barcode scanner alone (Figure 10). In the factory, however, because there are fifteen separate bins as opposed to three, more LED strips would be needed bringing the total up to $345, assuming that the barcode scanner costs the same. This cost multiplied by the 10 lines who would use this system bring the total to $3450.

This system does offer a return of investment. Because the likelihood of making a mistake is less, less money and time will be lost trying to fix a faulty window. Every window that has an incorrectly inserted steel reinforcement bar costs Silver Line approximately $40. If this system prevents 9 windows from having errors over the entire lifetime of the system, it will pay for itself.

Item	Price	Quantity	$25.00
Arduino Uno	$25.00	1	$25.00
USB Host Shield	$25.00	1	$25.00
Honeywell Voyager Barcode Scanner	$125.00	1	$125.00
LCD Screen	$6.00	1	$6.00
LED Tricolor Strip	$10.00	4	$40.00
12V Charger	$14.00	1	$14.00
Total			$235.00

Figure 10. *The cost to build the prototype smart pick bins system. Compared to a factory-implemented unit, there are twelve less LED tricolor strips and the barcode scanner is much cheaper.*

4. Results

While no completely functional prototype was created, the individual parts have been tested and are functioning as expected. The barcode scanner outputs data to the Arduino which then correctly interprets the raw information and transforms it into the correct barcode. The Arduino also successfully uses the barcode to determine the width of the window and consequently the sash size. The lights installed in the prototype correctly turn on, but the previous setback involving these lights prevents us from constructing the final system. Without access to another USB host shield to replace the broken one, it remains impossible to communicate with the scanner. The system is prepared for the replacement part and as soon as it is acquired final testing will commence.

Testing the barcode scanner on standard factory labels revealed an issue with the current system. With the Honeywell Voyager scanner many of the barcodes were unreadable and the ones that successfully scanned often took multiple tries. In the factory environment this would translate into worker frustration and lost time, the opposite of what the smart pick bin system is designed to do.

4.1 Setbacks

When working on any project, especially those involving sensitive electronics, drawbacks are to be expected. The first obstacle originated from the USB Host Shield connection. The initial soldering of the pin heads onto the shield resulted in a faulty connection. This problem was unknown at the time and effort was spent trying to correct the code rather than addressing the real problem. Once the issue was realized, it was a quick fix to re-solder the pins.

The next issue was a result of the LED strips. Because of the high voltage draw of the lights and the many data pins needed to control the lights, the first attempt to connect with the Arduino was unsuccessful. The wiring combined with the high voltage resulted in a short circuit on the board, rendering it useless. This problem was overcome by soldering the lights in tandem, allowing them to be controlled from a single data port. While needing much more wire to span the large distance between the ends, this prevents the board from burning and requires fewer ports to send data to the lights. The components of the smart pick bin system work independently and together without issues, but no data could be collected regarding a decrease or increase in takt time. However, considering the volume of error that occurs in the reinforcement steel process, smart pick bins should prove a viable and cost-effective method to reduce error significantly.

5. Conclusions

Any application involving human choice has the chance for error. In the steel reinforcement line, simple human errors can result in a structurally unsound window that, if left unfixed, can cause the window to fail. Fixing these errors after the fact is very costly. However, avoiding and reducing these mistakes is an achievable task. Smart pick bins are an easy and effective improvement to the steel reinforcement line that can help eliminate human error and help the worker with his/her decisions. Errors that ruin the integrity of the window are less likely with the use of smart pick bins. The smart pick bins also work to improve one-piece flow by informing workers about their cycle times. This allows them to adjust to meet the current takt time and keep production running smoothly.

The future of smart pick bins can go in multiple directions. Because of its simplistic and flexible design, these bins can be used in production lines throughout the factory. Any location where a worker must accurately choose the correct part for the window would benefit greatly from the smart pick bin system. For regular factory use some changes would be necessary. By using a higher quality barcode scanner similar to the ones used further down the line workers will no longer have to struggle with getting the barcode to scan. This will increase the cost of the system but will make it much easier to use. To counteract this cost increase, cheaper, single colored LED strips could be used which are 50% cheaper than the ones used in the prototype. The system can also be improved to collect data about the lines. With a wireless connection to the Silver Line local network through the smart pick bin system, information about a lines cycle time, inventory, and usage can be tracked, logged and later evaluated. This data can help determine the optimal setup and division of labor amongst different stations. The application of smart pick bins in the steel reinforcement line is the first step in what can become a factory wide improvement.

6. Acknowledgments

The authors thank Rutgers University, the State of New Jersey, Morgan Stanley, Lockheed Martin, Silver Line Windows and Doors, South Jersey Industries, Inc., The Provident Foundation, Novo Nordisk, the NJ GSET alumni community and current NJ GSET scholars. The research group would like to specifically acknowledge Taylor Au, the residential teaching assistant, for keeping the group on track and for proofreading the research paper. The group also thanks Silver Line Quality Engineer Wanda Duran for providing the necessary direction, mentorship, and materials for construction of the prototype and for helping the group explore Arduino software and hardware. Special thanks also go out to Dean Jean Patrick Antoine, Dean Ilene Rosen, and the rest of the NJ GSET staff for their guidance during the program period.

7. References

[1]Wanda Duran (private communications,2014).

[2]P*ick to Light Systems* (Pick To Light, 2014).

[3]*The Best-in-Class Pick to Light System* (Lightning Pick Technologies, 2014).

[4] *SpeasTech* (SpeasTech, Inc., 2014).
[5] *Pick to Ligt and Put to Light* (Genco Product Lifecycle Logistics, Pittsburgh, 2013).
[6] STI Customer Makes 25 Million Picks with Zero Errors (*SpeasTech*, 2014).
[7] *Vinyl Windows: Designed for Performance* (AIA/CES, 2014), pp. 6.
[8] *Arduino Uno* (OnkeLwik, 2012).
[9] *Arduino – LiquidCrystal* (Arduino, 2014).
[10] *Fritzing* (Fritzing, 2014).
[11] *Simple USB Host Shield for Arduino* (Solarbotics Ltd., 2014)

8. Appendix

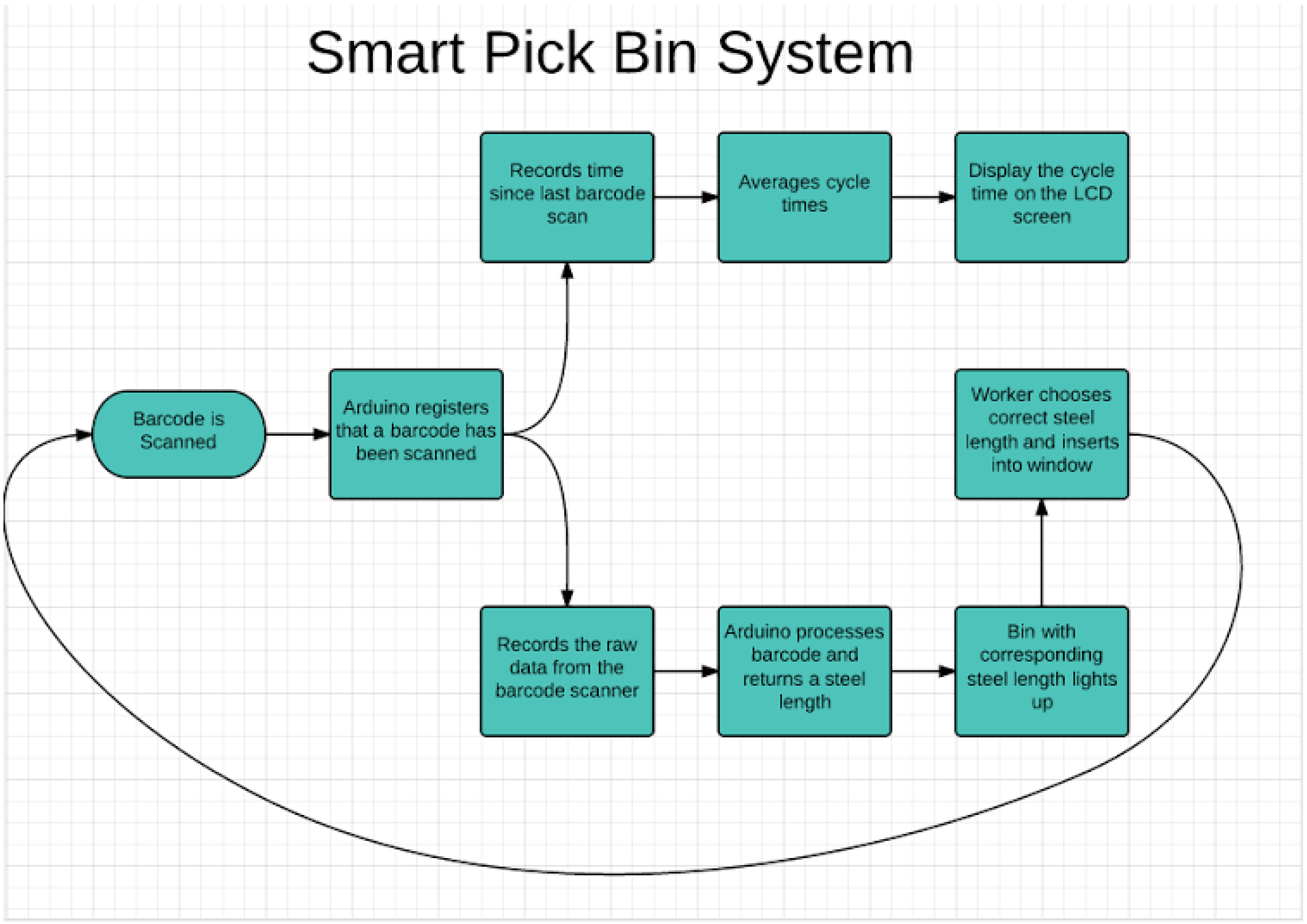
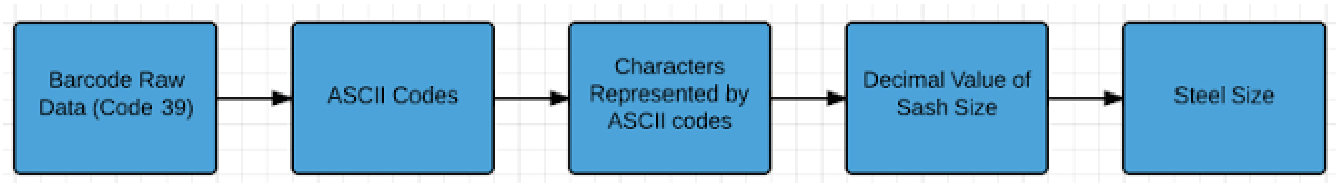
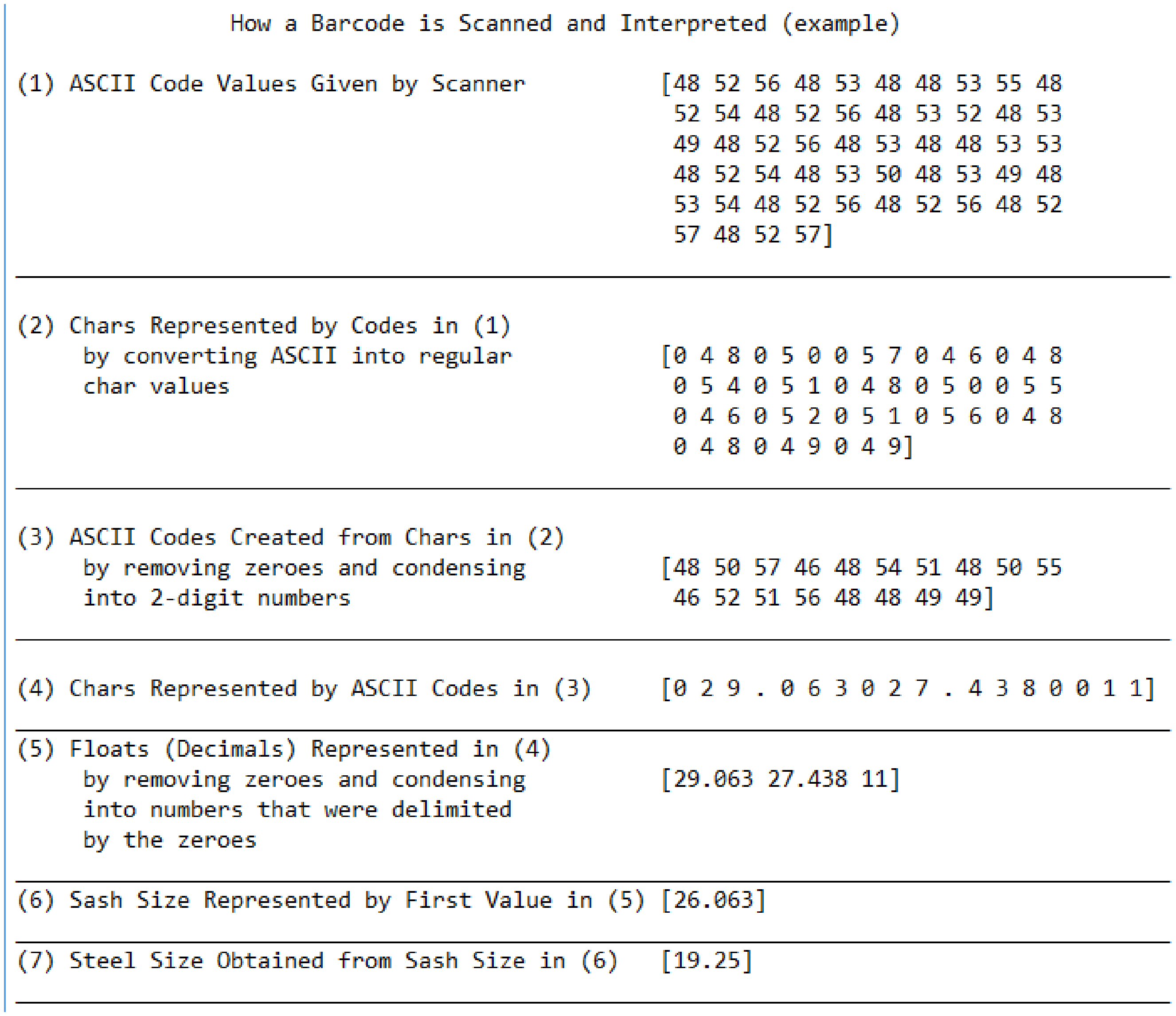

Smart Pick Bin System
Barcode is Scanned
Arduino registers that a barcode has been scanned
Records time since last barcode scan
Averages cycle times
Display the cycle time on the LCD screen
Records the raw data from the barcode scanner
Arduino processes barcode and returns a steel length
Bin with corresponding steel length lights up
Worker chooses correct steel length and inserts into window

Figure A1.


Barcode Raw Data (Code 39)
ASCII Codes
Characters Represented by ASCII codes
Decimal Value of Sash Size
Steel Size
Figure A2.


```
                 How a Barcode is Scanned and Interpreted (example)

(1) ASCII Code Values Given by Scanner          [48 52 56 48 53 48 48 53 55 48
                                                 52 54 48 52 56 48 53 52 48 53
                                                 49 48 52 56 48 53 48 48 53 53
                                                 48 52 54 48 53 50 48 53 49 48
                                                 53 54 48 52 56 48 52 56 48 52
                                                 57 48 52 57]
------------------------------------------------------------------------------------

(2) Chars Represented by Codes in (1)
     by converting ASCII into regular           [0 4 8 0 5 0 0 5 7 0 4 6 0 4 8
     char values                                 0 5 4 0 5 1 0 4 8 0 5 0 0 5 5
                                                 0 4 6 0 5 2 0 5 1 0 5 6 0 4 8
                                                 0 4 8 0 4 9 0 4 9]
------------------------------------------------------------------------------------

(3) ASCII Codes Created from Chars in (2)
     by removing zeroes and condensing          [48 50 57 46 48 54 51 48 50 55
     into 2-digit numbers                        46 52 51 56 48 48 49 49]
------------------------------------------------------------------------------------

(4) Chars Represented by ASCII Codes in (3)     [0 2 9 . 0 6 3 0 2 7 . 4 3 8 0 0 1 1]
------------------------------------------------------------------------------------
(5) Floats (Decimals) Represented in (4)
     by removing zeroes and condensing          [29.063 27.438 11]
     into numbers that were delimited
     by the zeroes
------------------------------------------------------------------------------------
(6) Sash Size Represented by First Value in (5) [26.063]
------------------------------------------------------------------------------------
(7) Steel Size Obtained from Sash Size in (6)   [19.25]
------------------------------------------------------------------------------------
```

Figure A3.

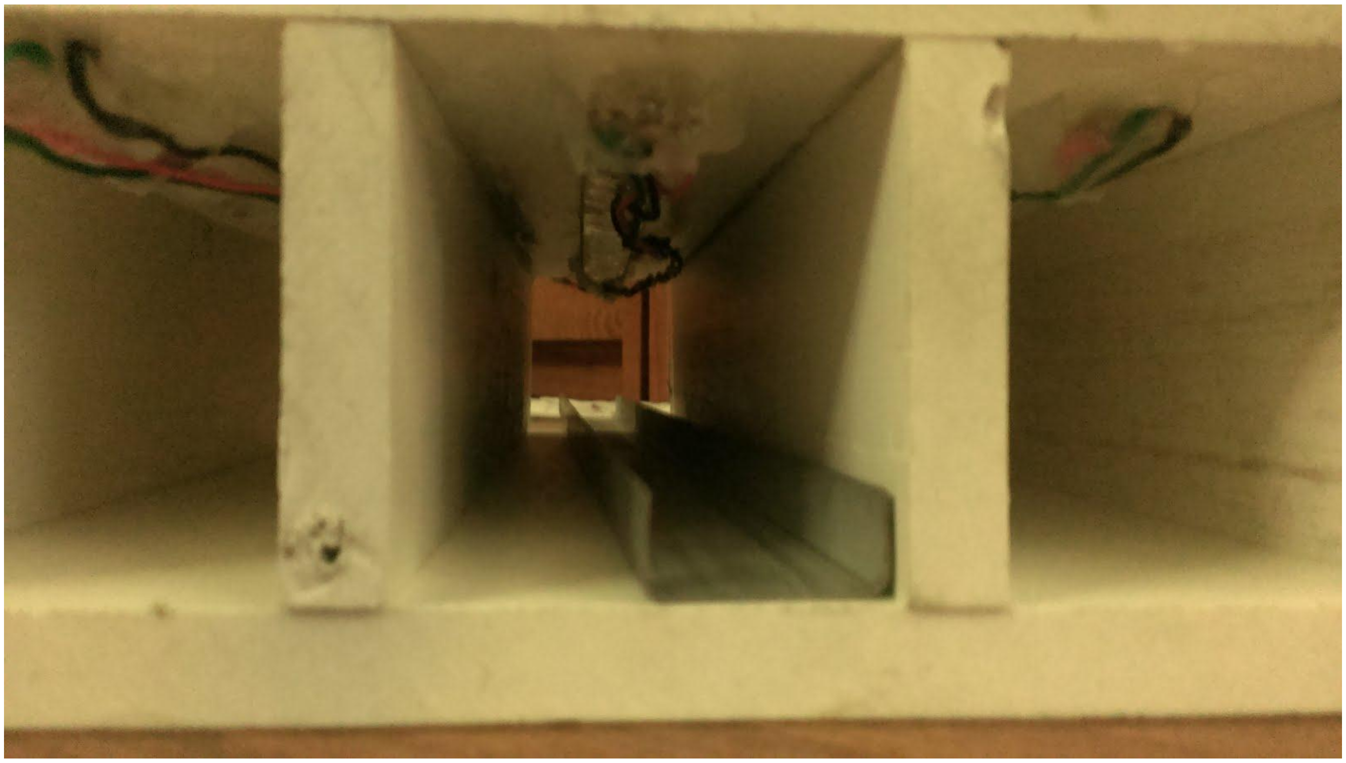


Figure A4.