GNG1103

Design Project User and Product Manual

Mill St. Optimization of Overall Equipment Efficiency

Submitted by:

OPTML Group C14

Hanna Paik, 300310147

Marie Levin, 300301114

Rafiullah Hamdard 300328829

Leila Simard 300291701

Yifei Li, 300308521

Wednesday, April 12th, 2023

University of Ottawa

Table of Contents

List of Figures

List of Tables

List of Acronyms and Glossary

Table 1. Acronyms

Acronym	Definition
PLC	Programmable logic controller
cpm	Cans per minute
IDE	Integrated Development Environment
UI	User interface

1 Introduction

This document overviews the theoretical design of an application capable of giving the user information useful for optimizing a packing line.

1.1 Background

Mill St's packing department consists of 3 packing lines which package beer in cans, bottles, and kegs. A packing line sends beer containers through a series of machines, connected by conveyors, that each perform a step of the packing process. Our client's job is to ensure the lines are always running at top efficiency. The efficiency of a packing line is the percentage of the actual packing yield, compared to the possible yield of the packing line, over a given period. Things that reduce the actual yield of a packing line are machine failures which can create blockages between units, or a bad line set up which can gradually cause a machine to be starved of containers over time if the machine's infeed is not consistent. Optimizing a packing line consists of configuring the speeds of the units to work together, and setting up conveyor's speeds in such a way that they can act as buffers between units.

This User and Product Manual (UPM) provides the information necessary for Mill St. Supervisors to effectively use the OPTML Application and for prototype documentation.

2 Overview

2.1 The Problem

A need exists for our client and his fellow supervisors to find the optimal speeds of their beer packaging process to ensure it is "always at top efficiency (Project Background, Brightspace)". The solution must have an attractive and straightforward interface that uses a flexible algorithm based on the V-Curve Theory to report optimized speeds of each unit and make recommendations on how to achieve them.

2.2 User Needs

Mill St uses a programmable logic controller (PLC) to set the speeds of the machines in their packing line. Units can be set to a given speed in bpm [bottles per min], kph [keg per hour] and cpm [can per minute]. Conveyors work differently in that their speed is controlled by adjusting the frequency (Hertz) that the motor is running at. Therefore, the speed recommendations of the conveyors should also be displayed in hertz.

The VFD or PLC controllers for various machines are in different areas around the packing line which spans a large area. The client wanted to be able to physically walk up to these devices to observe and input data into our device. Therefore, our product needs to be used on a mobile hardware. The client preferred iOS devices, so our team developed a native iOS application for an iPad.

The client requested that the application would run like a simple offline “calculator-type" software, where the packing line set up would be preconfigured, and the desired speed of the bottleneck could be inputted into the program and the predicted speeds for the other machines would be displayed. Refer to section () which explains V-Curve theory and the importance of the bottleneck of a packing line.

Prioritising Needs

Priority	Need	Design Criteria
5	“Looking for a solution that gives can/bottles per minute and kegs per hour (Ritchie, Client Meeting Transcript)” (KPH) Calculation/ algorithm	Input desired filler speed then output ideal speeds of each unit (cans/bottles per minute or hour). **What the optimised speeds are** in both kegs/hour and cans per minute
5	User interface showing the speed of all the equipment and conveyor **Easy to use**	**Detailed information on an easy-use interface.** Show: Suggested speed in order to optimise yield and current speed of filler station.
3	Must be **maintainable** and **adapt** to expansion of the brewery and adapt to new products.	**Adjustable algorithm** that can add new conveyors into its functions and adapt to new processes.
2	Budget of the design team: 100$	**Maximum 100$**
5	The system should **identify areas that are and are not running at optimal rates**.	**Identify which units in the process are not at the ideal speed** within a range (ex. ± 100 cans/min).
4	It should be able to **make recommendations** if the speeds are not optimal.	**Make recommendations** on which units may be the source of inefficiency in the line.

Figure 1: Table of Prioritized User Needs

2.3 Fundamental Concept

Based on the user needs, our team developed a flowchart to show the fundamental concept design that we attempted to produce.

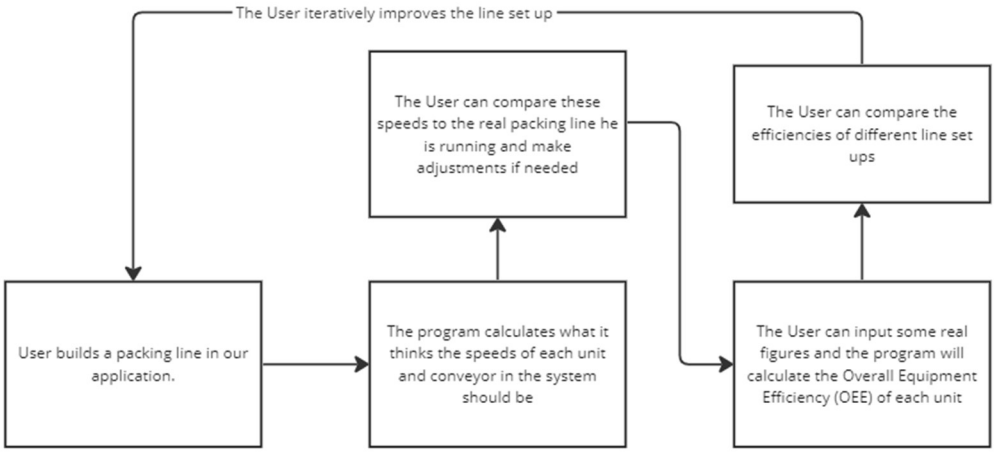


Figure 2: Flowchart of Fundamental Concept of How Product Should Be Used

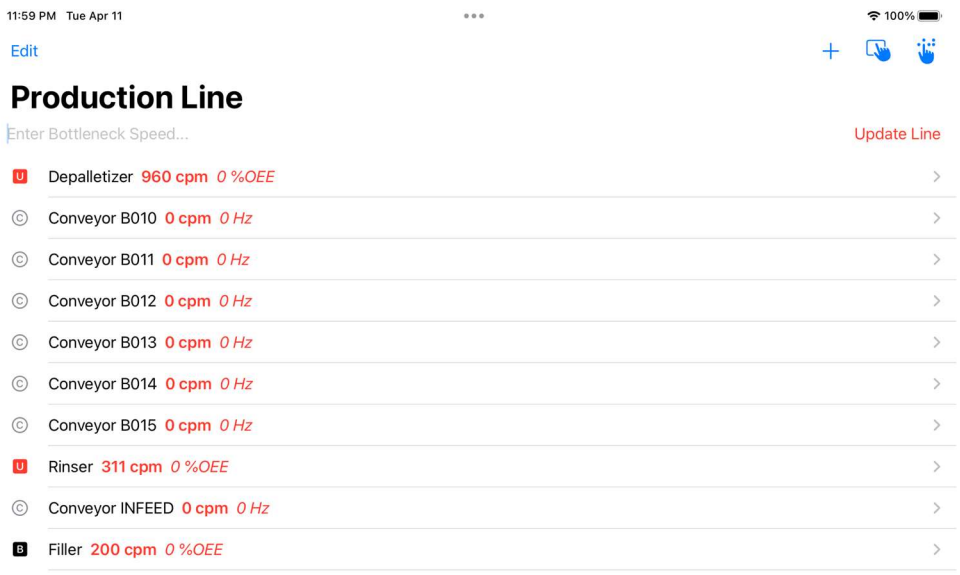
2.4 Key Features

- Deconstructs the packing line and breaks down the behaviors of the packing line into mathematical functions. These functions were layered to create an algorithm capable of predicting the speeds of a production line with minimal information provided.
- Shows the capabilities of a native iOS application and its ability to display the data in a clean, intuitive way.
- Built strongly around the client's needs as it displays the speeds of the conveyors as well as the corresponding motor frequency to apply the recommendations in the workplace.

2.5 The Product

The product that was developed is a native iOS application. Its features include:

- The ability to construct a Production/Packing line by adding units and conveyors to a list, (that represents the packing line) with the top of the list being the start of the packing process.
- When you add a machine, you can give it characteristic depending on the machines' function in the packing line.
- Existing machines can be edited, moved around, and deleted.
- The behaviors of the units can be configured, such as selecting the bottleneck or allowing the user to manually set the speed of a unit.
- The bottleneck speed can be set, and the speeds of the other units will be predicted upon updating the line.



2.6 Cautions & Warnings

Please be warned that the formulas used to calculate V-Curve Theory were tested only on two production lines at two different speeds. It could be the case that the production line you create with this application does not correspond to our calculations and thus, our results may not be applicable to your production line.

3 Getting started

Our product is a native iOS application meant to run on an iPad. To use this application, it must first be deployed to the app store using an Apple Developer Account. If you just plan on viewing the simulation of the application, you would need an Apple device capable of running Xcode, and then open the product with Xcode. There, you can select the iPad Pro as the device to simulate the application, and then run the simulation of the application.

Upon successfully opening the application you would see the main screen of our application:

The application is empty due to this being the first time you open the application on your device.

3.1 Configuration Considerations

The configuration of this application was all built around the manipulation of an object that was called a “machine”. The machine was given a handful of characteristics such as name or width. These characteristics would indicate if it was a Unit, the bottleneck, and so on. This application can store a set of machines in its local memory called Core Data. The User can create and add machines to the set, remove machines from the set, move around machines, and edit characteristics of machines through the different UI inputs like text fields, toggles, and sliders. Every time the User changes the set of machines, ,t is saved to Core Data. When the User updates the speed of the bottleneck machine, it takes the speed of the bottleneck and the other Unit’s already known location from the bottleneck and uses those values to calculate the speed of all the other Units. It then updates their characteristics accordingly, which appear along with the change in bottleneck speed. If a Unit is selected to not follow V-Curve Theory, it uses the distance of the Unit beside and closest to the bottleneck and gets updated with the corresponding speed of that Unit instead. This was done because the characteristic was observed in historical data. The manual set speed of a Unit is not overridden by this process because the condition applies that if the Unit is manually inputted, it will be skipped as it iterates through the Units to calculate its speed.

3.2 User Access Considerations

The Apple Developer Account can restrict access to the application only to the Mill St business and its company devices (Distributing your app to registered devices, 2023). Our client stated that the ones using this device would be himself and his fellow managers/supervisors of the production line. Therefore, restricting access to within the business is enough security for this application.

3.3 Accessing/setting up the System

Refer to (What you need to enroll, 2023) and (Distributing your app to registered devices, 2023) for information on creating an Apple Developer Account and Deploying an application.

3.4 System Organization & Navigation

Navigating the application is very simple. There is a toolbar at the top of the application with various buttons. The buttons on the right and the › symbol leads to different popup windows, while the Edit button changes the view of the main window to allow for the elements of the production line to be deleted or moved around.

3.4.1 Add Machine Window

To navigate to the Add Machine Window, you would click the + button. When you are done with the Add Machine Window, you can click Submit to return to the main page.

3.4.2 Unit Configuration Window

To navigate to the Unit Configuration Window, you would click the button. When you are done, you can click the main page to return.

3.4.3 Editing Machine Window

To navigate to the Editing Machine Window, you would click on the gray arrow on the left side of a machine . When you are done with the Editing Machine Window. You can click Submit to return to the main page.

3.5 Exiting the System

Simply closing the application and turning off your device is the way to exit this application.

4 Using the System

When you first start using the application you will be greeted with an empty Production Line.

Figure 3: Initial View of Application

The application is empty due to this being the first time you open the application on your device.

4.1.1 Add Machine Window

The first thing you can do is add machines to your production line. To do this, you would click the

+ button to add a machine to your line.

Upon pressing the button, you will see the Add Machine Window prompt pop onto the screen. Here, you can name your machine, give your machine a width, and define whether it is a Unit or Conveyor. When you are done creating your machine, click Submit to add it to your production line.

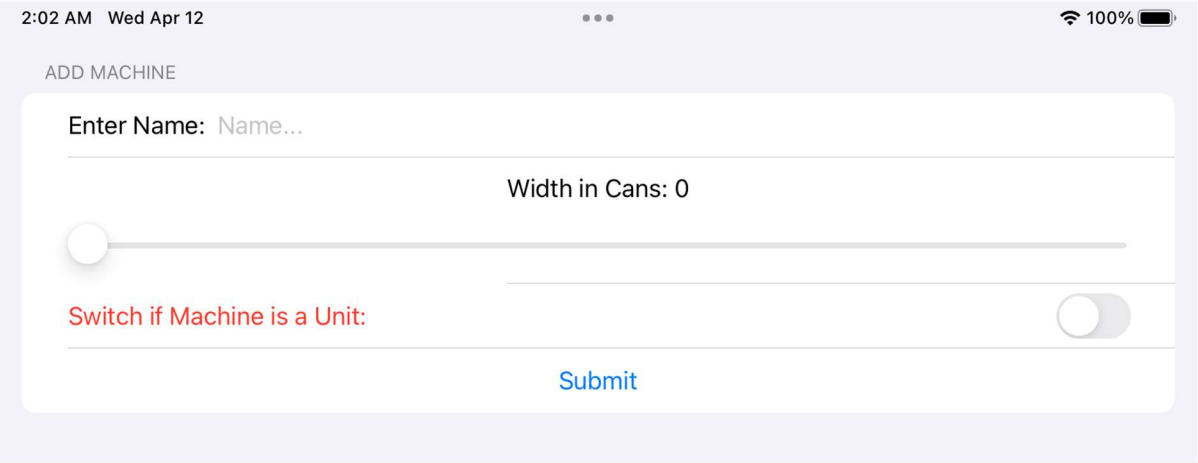


Figure 4: Empty Add Machine Window

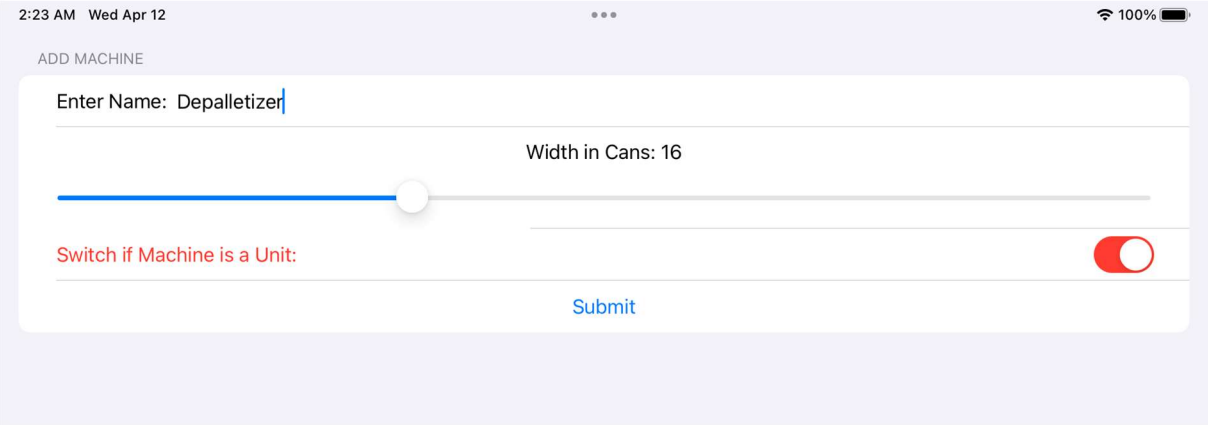


Figure 5: Filled Add Machine Window

4.1.2 Main Page

Once you have added all the machines you want, your main page should look something like this:

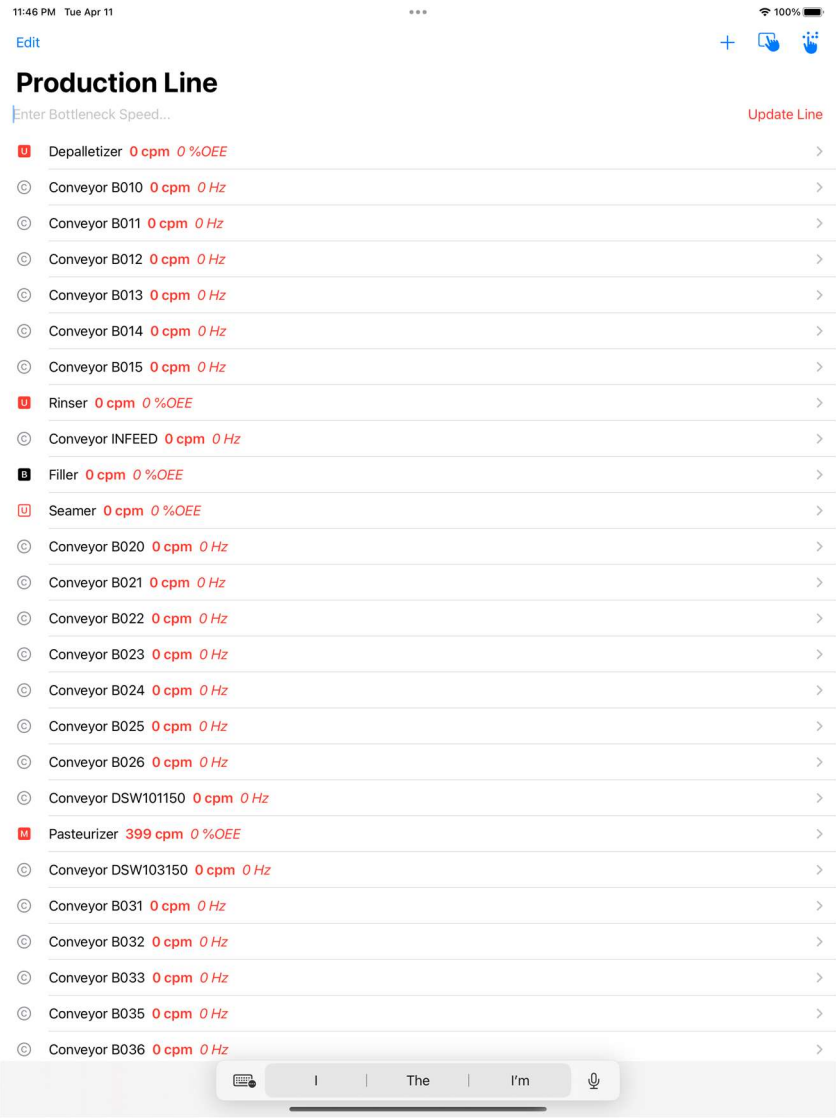


Figure 6: Main Page Filled

This is not a full view of all the machines, you can scroll down to see the rest of the line.

At this moment, everything is set up so that if you close the app and reopen it, all your data should be saved.

4.1.3 Unit Configuration Window

Some of the icons at the side of the machines are coloured differently and have different letters because the Units were configured by clicking the icon. Upon clicking this icon you will see a popup view of just the Units. Here you can select the bottleneck of the system and select units that follow V-Curve Theory.

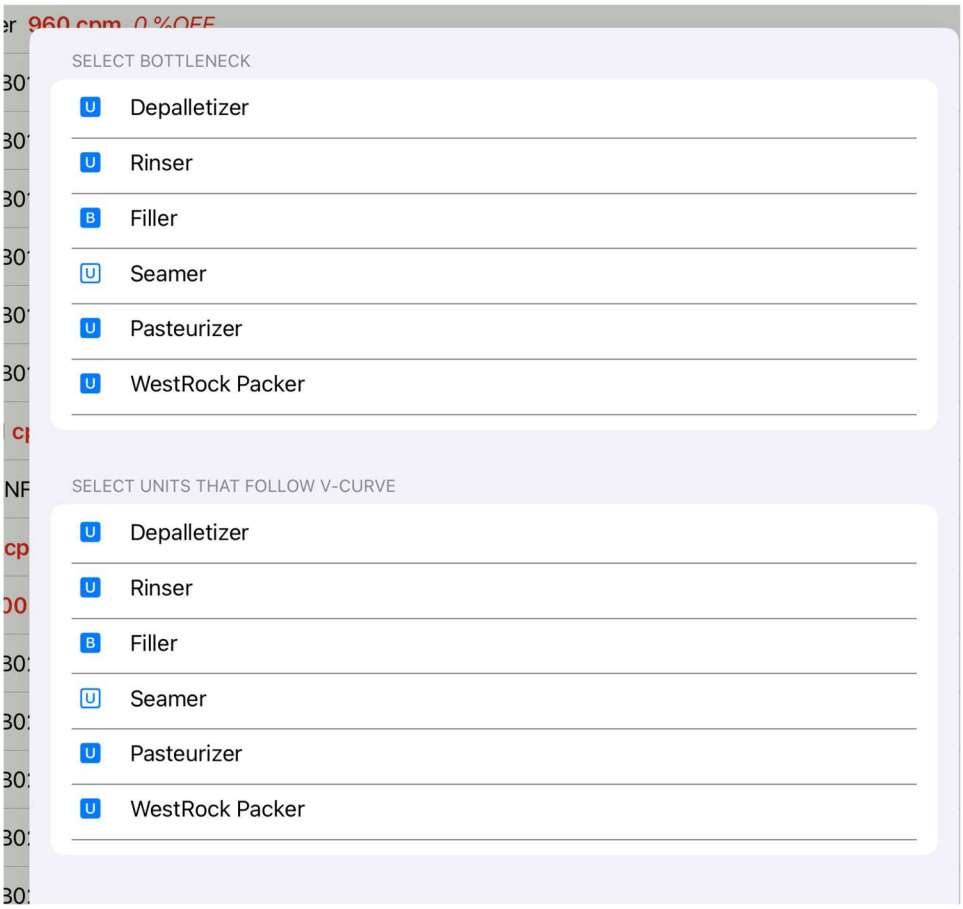


Figure 7: Unit Configuration View

Unit Icon Meanings	
	Unit's speed is calculated with V-Curve Theory
	Unit's speed will take on the speed of the Unit beside it (the one closest to the bottleneck) If this Unit is set up before the bottleneck, it will take on the speed of the Unit after it. If this Unit is set up after the bottleneck it will take on the speed of the Unit before it.
	Unit's speed is manually inputed by the User. (Can be configured when editing the Unit)
	This Unit is the bottleneck of the production line.

At this stage, all necessary set up preperations have been made. Now, you can go back to the main window and type in your desired bottleneck speed, and then press the Update Line button.

Upon updating the line, you will observe that the speeds of the Units have been predicted by the application.

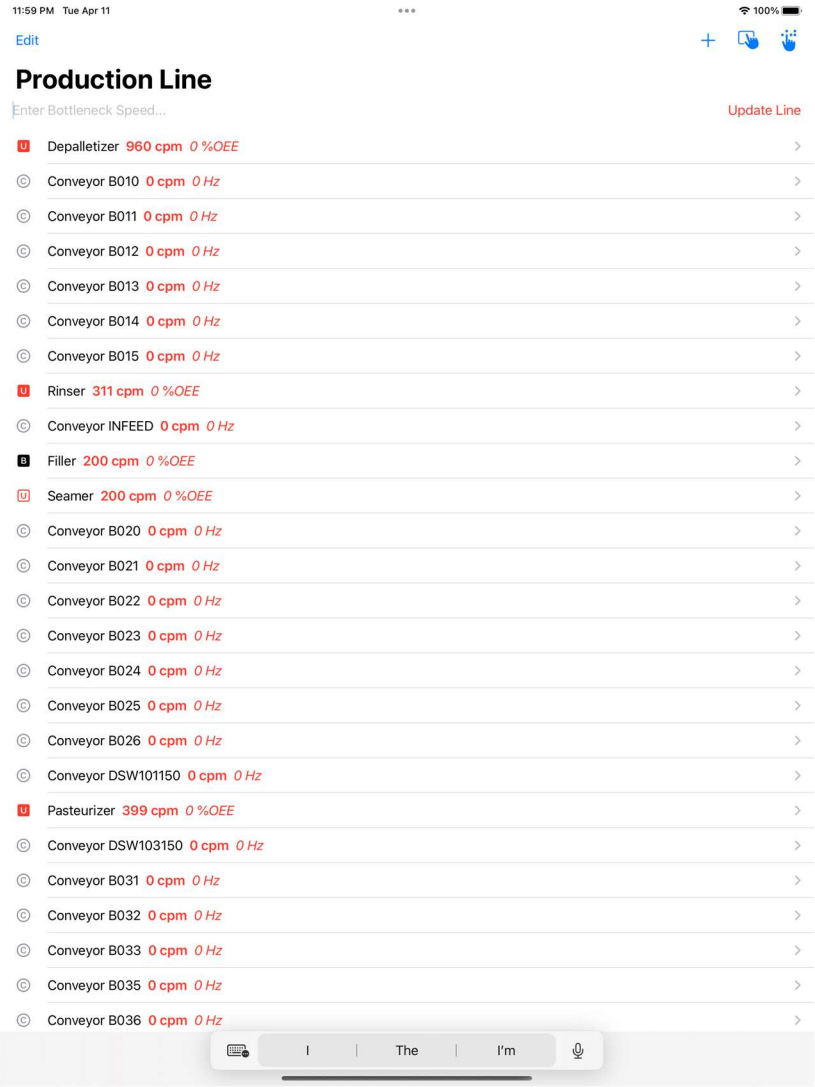


Figure 8: View of Predicted Speeds

4.1.4 Editing Machine Window

If you want to edit existing machines on your line, you can simply tap on the machine you want to edit and an editing window will pop up for you to change existing characteristics of the machine. In addition, there is an option to set their speed manully.

This is a view of the editing window:

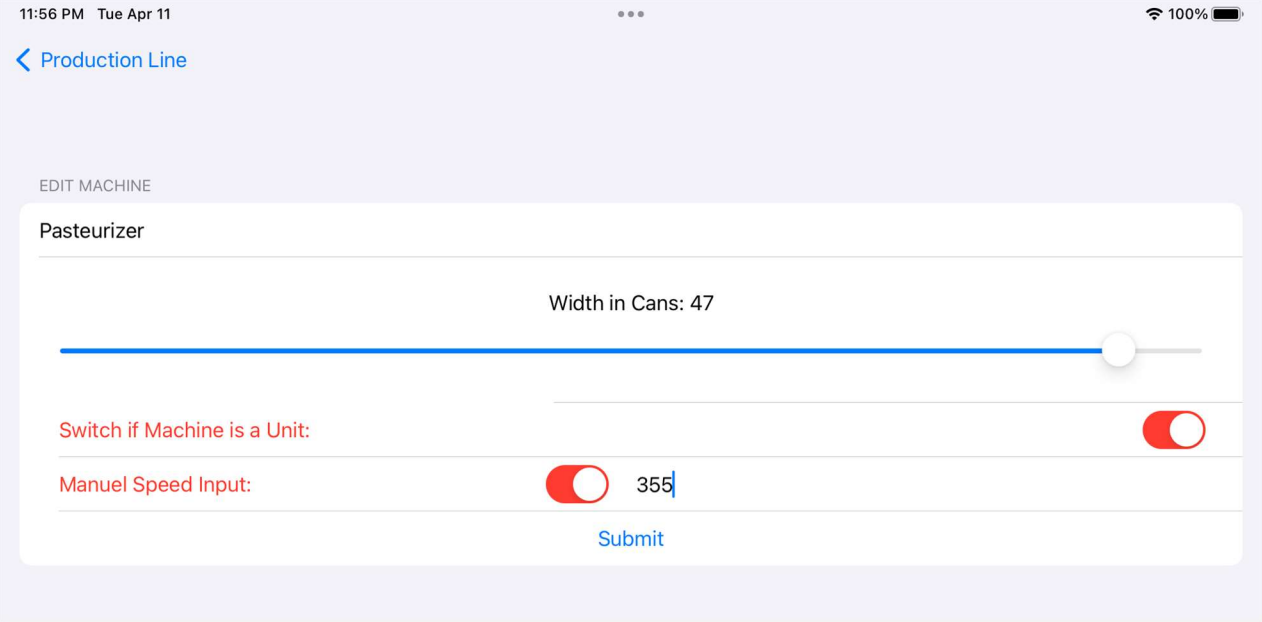


Figure 9: Edit Window

Here, you can change the name, width, if it's a Unit, or set a manual speed. Select the Submit button to save your changes.

4.1.5 Edit Mode of Main Page

If you want to just delete a machine or move existing machines around, you can click the Edit button.

Upon selecting this button, the main page will change it's view to Edit Mode. This mode allows you to hold the bars at the end of each machine. When the bars are held, you can drag the machine to a different place on the line. You can also select the button which reveals a Delete swipe that can be swiped left to delete the machine.

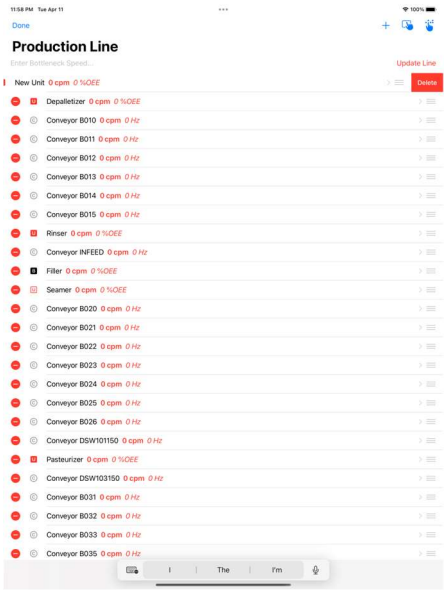


Figure 10: Edit Mode

This is the full extent of this application. If you close and reopen the application, all your data will remain.

5 Troubleshooting & Support

Most actions you can perform on this application are easily reversible by deleting the machine you created. Errors are most common in the display aspect of the application.

5.1 Display Errors

When you select the Unit Configuration Window, sometimes the Units are displayed in the wrong order, or when the units are selected, they do not show that they have been selected on the popup window. This is a simple display error, meaning that the actions have been performed and the Units are in the order displayed on the main page. This is just a bug in the display of the popup window. This is why it is set to a smaller frame so you can still see the main page when performing actions. When a display error occurs, you should exit the popup and check the main page for the changes you made and go back to the Unit Configuration Window to continue editing if needed.

If an error occurs with the display of the speeds, you should open the Unit Configuration window, change the bottleneck, and change it back. This will reset the speeds of the line.

If the app crashes when you enter the bottleneck speed, make sure your line has less than 6 consecutive Units before or after the filler. It may be the case that the V-Curve formula is calculating speeds that are a higher number than the display can show, which will crash the system.

5.2 Special Considerations

The historical data used to formulate the speeds of each Unit, based off the bottleneck, had a bottleneck with a max speed of 300 cpm. Therefore, if you put numbers higher than 300 cpm, t is possible to get large enough numbers to crash the system because the application cannot display the numbers.

5.3 Support

Contact hpaik052@uottawa.ca if you need emergency support.

6 Product Documentation

6.1 iOS Application

This design was constructed using Xcode, an IDE for developing native iOS applications. To develop an application on this software, you need to create a free Apple ID which allows you to access Apple platforms and documentation necessary to create an application. The type of project we built in Xcode was an iOS application that was coded with Swift UI in Swift language. Along with the IDE, there were also a few libraries used to add extra functionality to our product.

Core Data is a built-in library in Xcode that was used to implement the ability to save data created with the app locally to the device. With Core Data, we added a Data Model file to our project that allowed us to create our 'Machine' Object. This was used to store the characteristics of each machine as individual variables tied to a 'Machine' Object so that in our code, we could refer to the Can Rinser as a Machine Object with several variables, including a variable called "name" that holds a String of symbols representing its name "Can Rinser" and variable called "width" that holds the number 1 for its width. Core Data allowed our team to create a more intuitive reference of the data being manipulated and allow the data to be stored locally on the device as sets of Machine Objects. https://developer.apple.com/sf-symbols/

SF Symbols 4 is a free library that can be downloaded off the Apple Developer website. It contains 4,400 symbols that can be used for icons in the UI of an application. This was used in our design in various places such as the tool bar, and the letter icons beside the machine names which indicate its function in the packing line. This library allowed images for icons to be referred to as their file name, without needing to describe the directory path that allows the computer to know where to go to find the image file. Instead, it is referred to as Image (systemImage: "u.square.filled") to produce the icon  .

6.2 Algorithm

The Optimal Speed of the Units are derived from the V-Curve Theory presented by LineView.

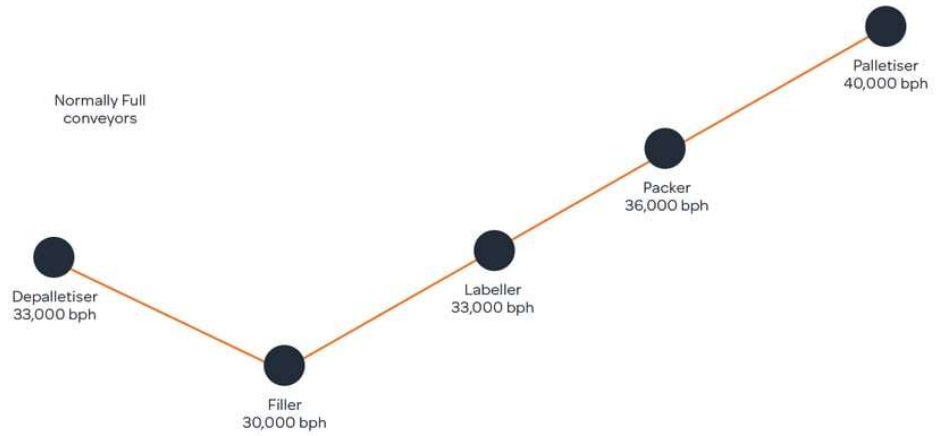


Figure 11: Theoretical V-Curve Speed Graph (LineView 2022)

Essentially, all the Units before and after the bottleneck should be set to 10% faster increments of the bottleneck speed as they get farther away from the bottleneck. This can be represented by the formula:

$$VCT speed = Bottleneck Speed * (1 + \frac{distance}{10})$$

*distance is a whole number value representing the distance from the bottleneck. i.e., the Packer in Figure 3's distance would be 2 as it is the second unit away from the filler.

With this information and historical data given by the client, the following formulas can be derived:

$$Speed\ of\ Desired\ Unit = \begin{cases} Theoritical\ Vcurve \cdot Ratio\ to\ match\ Historical\ Data \\ Theoritical\ Vcurve \cdot Ratio\ to\ match\ Historical\ Data\ +\ Addition\ Because\ Units\ are\ After\ Filler \end{cases}$$

$$\text{Speed of Desired Unit } = \begin{cases} Filler\ Speed\left(1+\frac{distance}{10}\right)\times 2^{\frac{distance^2}{2}} & :Before\ Filler \\ Filler\ Speed\left(1+\frac{distance}{10}\right)\times 2^{\frac{distance^2}{2}}+\frac{1}{3} & :After\ Filler \end{cases}$$

$$Theoritical\ Vcurve\ =\ Filler\ Speed\ (1+\frac{distance}{10})$$

$$Filler\ Speed\ =\ Speed\ of\ Filler\ Unit$$

$$distance\ =\ number\ of\ Units\ from\ Filler\ (That\ Are\ in\ Vcurve)$$

$$Ratio\ to\ match\ Historical\ Data\ = 2^{\frac{distance^2}{2}}$$

$$Addition\ Because\ Units\ are\ After\ Filler\ =\ \frac{1}{3}$$

*FillerSpeed is the name for the bottleneck speed variable because the Filler station was the bottleneck of the packing line being analyzed.

There are three different behaviors that the Units were seen to exhibit:

1. The Unit matches the speeds derived from the equations above.
2. The Unit follows the speed of the Unit that comes before it (Seamer + Mead Packer)
3. The Unit is set to a speed regardless of changes in line set up (Manuel speed input) (Pasteurizer)

*The Pasteurizer often needs to run at a certain speed to ensure that the cans and bottles are pasteurized for a certain time frame to ensure food safety.

Upon creating an algorithm, the last step is implementing it into the UI.

6.2.1 Implementation of Algorithm

Translating these 3 series of behaviors into code requires sufficient coding knowledge.

```
    }
    modifiedIndex = 0
    count = 0
    for (index, _) in Units.enumerated().reversed() {
        if (Units[index].isBottle){
            if(!Units[index].isInput){
                Units[index].speed = (text as NSString).doubleValue
            }
            n = 0

        }else if(index < bottleIndex){
            if(Units[index].isBefore){
               modifiedIndex = index + 1 + count
                count = count + 1
            }else{
                modifiedIndex = index + count
            }
            n = abs(modifiedIndex - bottleIndex)
            multipler = pow(Double(n),2)
            multipler = multipler/2
            multipler = pow(2,multipler)
            if(!Units[index].isInput){
                Units[index].speed = (1 + (Double(n)/10.0)) * multipler * (text as NSString).doubleValue
            }

        }


    }
    modifiedIndex = 0
    count = 0
    for (indexj, _) in Units.enumerated() {
        if (Units[indexj].isBottle){
            Units[indexj].speed = (text as NSString).doubleValue
            n = 0
        //Units[indexj].oee = Double(n)
        }else if (indexj > bottleIndex){
            if(Units[indexj].isBefore){
                if(!Units[indexj].isInput){
                    Units[indexj].speed = Units[indexj - 1].speed
                }
                modifiedIndex = indexj - 1 - count
                count = count + 1
            }else{
                modifiedIndex = indexj - count
                n = abs(modifiedIndex - bottleIndex)
                multipler = pow(Double(n),2)
                multipler = multipler/2
                multipler = pow(2,multipler) + 0.4
                if(!Units[indexj].isInput){
                    Units[indexj].speed = (1 + (Double(n)/10.0)) * multipler * (text as NSString).doubleValue
                }
            }

        }
        DataController().save(context: manageObjContent)
```

Figure 12: Snippet of code that calculates speed of Units.

6.3 User Interface

To design the User Interface, you need to have a good idea of the capabilities of the Swift language and the user interface toolkit of Swift UI.

Here is some learning material for Swift UI:

SwiftUI | Apple Developer Documentation

Hacking with Swift – learn to code iPhone and iPad apps with free Swift tutorials

Our first part of designing the final UI was to make a flow chart of all the navigations and action we wanted our design to have.

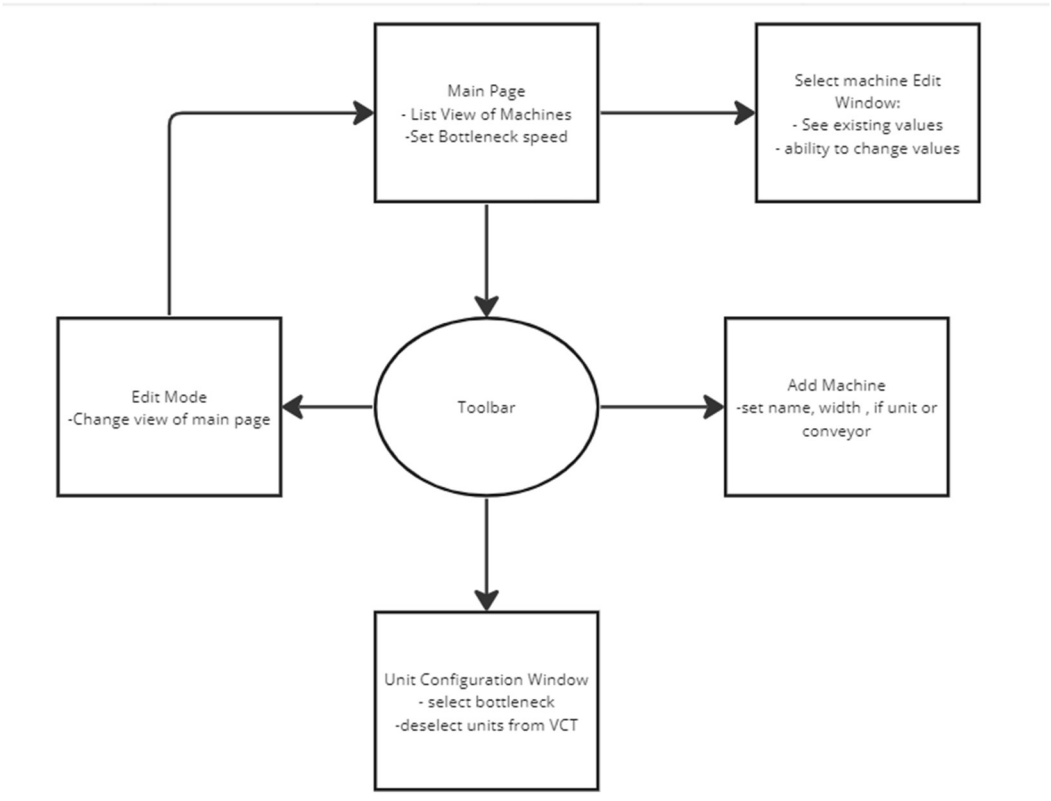


Figure 13: Flowchart of UI concept

All that's left is translating the concept into Swift and implementing it into the Xcode project.

6.3.1 BOM (Bill of Materials)

No	Item name		Description	Units of measure	Quantity	Unit Cost	Extended Cost	Link
1	Xcode		IDE for native ios application	Software application	1	0	0	https://developer.apple.com/xcode/
2	Xcode Object Libraries	Core Data	Xcode Libraries	additional software download	1	0	0	https://developer.apple.com/xcode/swiftui/
		SF Symbols 4						SF Symbols - Apple Developer
		Swift UI						https://developer.apple.com/xcode/swiftui/
3	Excel		Windows Application for data analytics	Software application	1	0	0	Core Data | Apple Developer Documentation
4	Apple developer account		To deploy the application for the company	Subscription	1	99	99$ per year the app is used	https://developer.apple.com/programs/enroll
Total product cost (without taxes or shipping)							99$	
Total product cost (including taxes and shipping)							111.87$	

6.3.2 Equipment list

Equipment needed to develop and run this application:

- iPad to run application.
- Mac to run Xcode.

6.4 Testing & Validation

4	[Algorithm Test] The objective of this Test is to prove our ability to calculate the optimal speeds before implementing it into the application.	Test of the algorithm that predicts the optimal speeds. Prototype of formula in excel was used. An excel spreadsheet, that displays all the elements and functions of the algorithm and when given the historical data it can replicate the optimal speeds within a small percentage of error.	**Record:** Margin of Error: 10% Pass. **Response:** Algorithms will be expanded to include the conveyors in the future.	4-7th March

Test of algorithms ability to predict speeds:

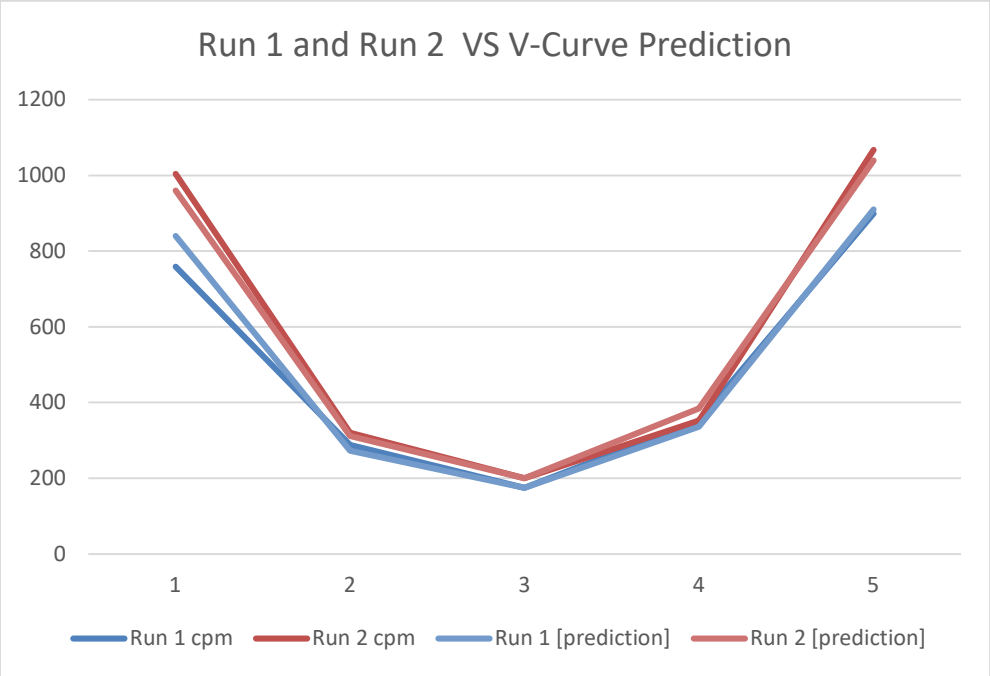


Figure 14: Graph of Algorithm VS Historical Data

2	[Pure UI Test] The objective of the test is to make sure the program can be interacted with without issue before focusing on its ability to perform its task.	Test the functionality and ease of use of the UI. Prototype UI Shell will be used. A prototype of the UI that does not perform any real calculations or simulations. Can only navigate to different windows and add machines	**Recorded:** Score 5/5 5 navigation windows each performed desired task with no bugs. **Response:** Will implement Algorithms in the future.	4 - 7th March
5	[Integration Test] The objective of this test is to make sure that the display of the input and output is well formatted on the UI.	Tests of the algorithm's integration into the UI. A first rendition of the application will be created that includes the UI and the algorithm working together, the focus will be on the formatting of the input and output numbers and other variables on the UI and the ability of the user to change aspects of the production line and the algorithm still calculates things accordingly to test cases made from Excel prototype.	**Record:** **Scale from 100 cpm to 300cpm was set for the bottleneck and the displayed numbers were tested to match excel data.** **Score: 5/5** **Pass**	9-10th March

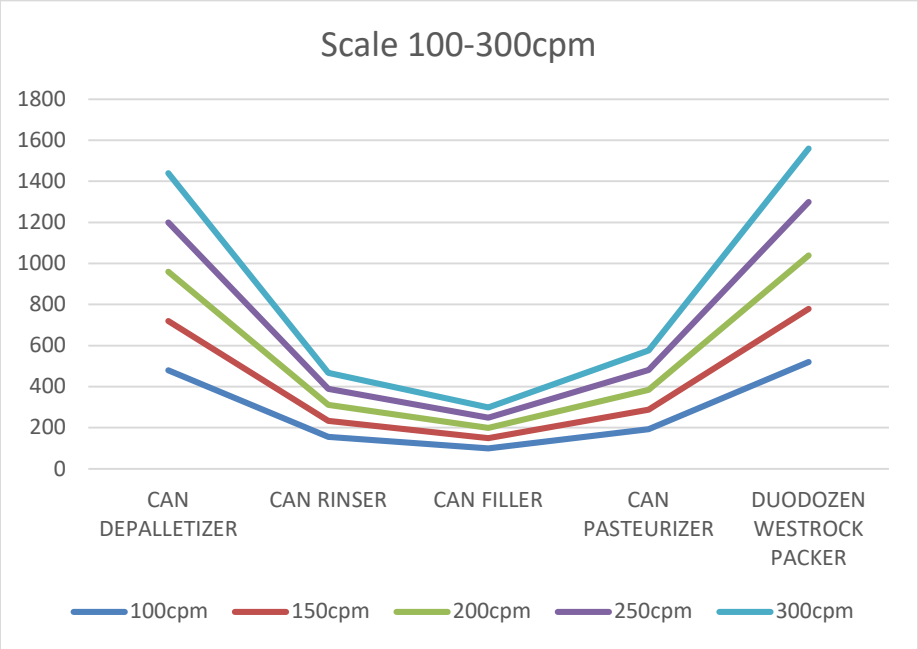


Figure 15: Graph of Algorithm's ability to scale from 100-300cpm

7 Conclusions and Recommendations for Future Work

In conclusion, our project work with our teammates has been a great experience. We really enjoyed collaborating with each other and demonstrated a strong sense of teamwork throughout the project. Our ability to support each other and learn from one another made it possible to overcome the most challenging aspects of the project.

Working together, we developed a range of skills that will serve us well in our future projects. We learned how to manage our time effectively and prioritize tasks to meet deadlines. We also demonstrated our ability to give and receive constructive feedback, which helped us improve our work and build better relationships with each other.

As a team, we were able to address complex tasks and find solutions that met the project's requirements. Overall, our experience by working together was a good opportunity for us and we learned a lot from it.

Although our final design doesn’t have all the features we planned, this experience has allowed us to learn the design process and experience designing a product from start to finish.

If we had more time to work on this design, we would do more research on calculating the OEE of a packing line and implement more of our calculation into the application.

8 Bibliography

Apple. (n.d.). *Before You Enroll*. Apple Developer. Retrieved April 12, 2023, from https://developer.apple.com/programs/enroll/

Apple. (n.d.). *Distributing your app to registered devices*. Apple Developer. Retrieved April 12, 2023, from https://developer.apple.com/documentation/xcode/distributing-your-app-to-registered-devices

Apple. (n.d.). *Home*. YouTube. Retrieved April 12, 2023, from https://developer.apple.com/documentation/xcode/distributing-your-app-to-registered-device

Apple. (n.d.). Submit your iPadOS apps to the App Store. Apple Developer. Retrieved April 12, 2023, from https://developer.apple.com/ipados/submit/

Härte, F.L. Efficiency analysis of packaging lines. Delft University Press. WBBM Report Series 33. 1997

Efficiency analysis of packaging lines | TU Delft Repositories

Is the V-Curve the best option for setting accumulation line speed? (n.d.). LineView Solutions. Retrieved March 5, 2023, from https://news.lineview.com/is-the-v-curve-theory-the-best-option-for-setting-accumulation-line-speed

APPENDICES

9 APPENDIX I: Extra Calculations

9.1.1 Calculating Hertz

PURE CALCULATION METHOD (no scaling)
Required units:
Motor rpm at 60 hz (rpm)
Gearbox reduction (Gr)
Sprocket reduction (Sr) = $\frac{(driven\ sprocket\ teeth)}{(drive\ sprocket\ teeth)}$
Diameter of driven drum (D) in
VFD(variable frequency drive) output Hertz (Hertz recorded from conveyor) (VFD_{output})

1. Calculate overall reduction (Or)

$Gr \times Sr = Or$

2. Calculate final rpm (rpm_{final})

$\frac{rpm}{Or} = rpm_{final}$

3. Calculate circumference of driven drum (C)

$\frac{(D \times \pi)}{12in(1foot)} = C$

4. Calculate final meters per minute (mpm)

$rpm_{final} \times C = mpm$

5. Calculate Output speed

$\frac{VFD_{output}}{60hz \times mpm\ @\ 60\ hz}$ = output speed in mpm on the surface of the driven drum

$$\frac{VFD_{output}[Hz]}{60\,[Hz] \times (\frac{rpm}{Gr \times Sr} \times \frac{(D \times \pi)}{100cm})\,[@\ 60\ Hz]} = fpm\ at\ surface\ of\ driven\ drum\ for\ given\ VFD_{output}$$

TACHOMETER METHOD (scaling)

1) Set drive to zero and adjust everything so the conveyor is not moving
2) Set drive to 60 hz
3) Use a handheld tachometer to measure the speed (mpm) of the conveyor
4) Repeat for every 10 Hz and double check
5) Use the numbers to make a scale (with Parameter block and output the data to a read out or HMI → PLC method)
6) The scale will be used to estimate the conveyors meters per minute at different Hertz

Figure 16: Methods to Estimate the Hertz of a Conveyor

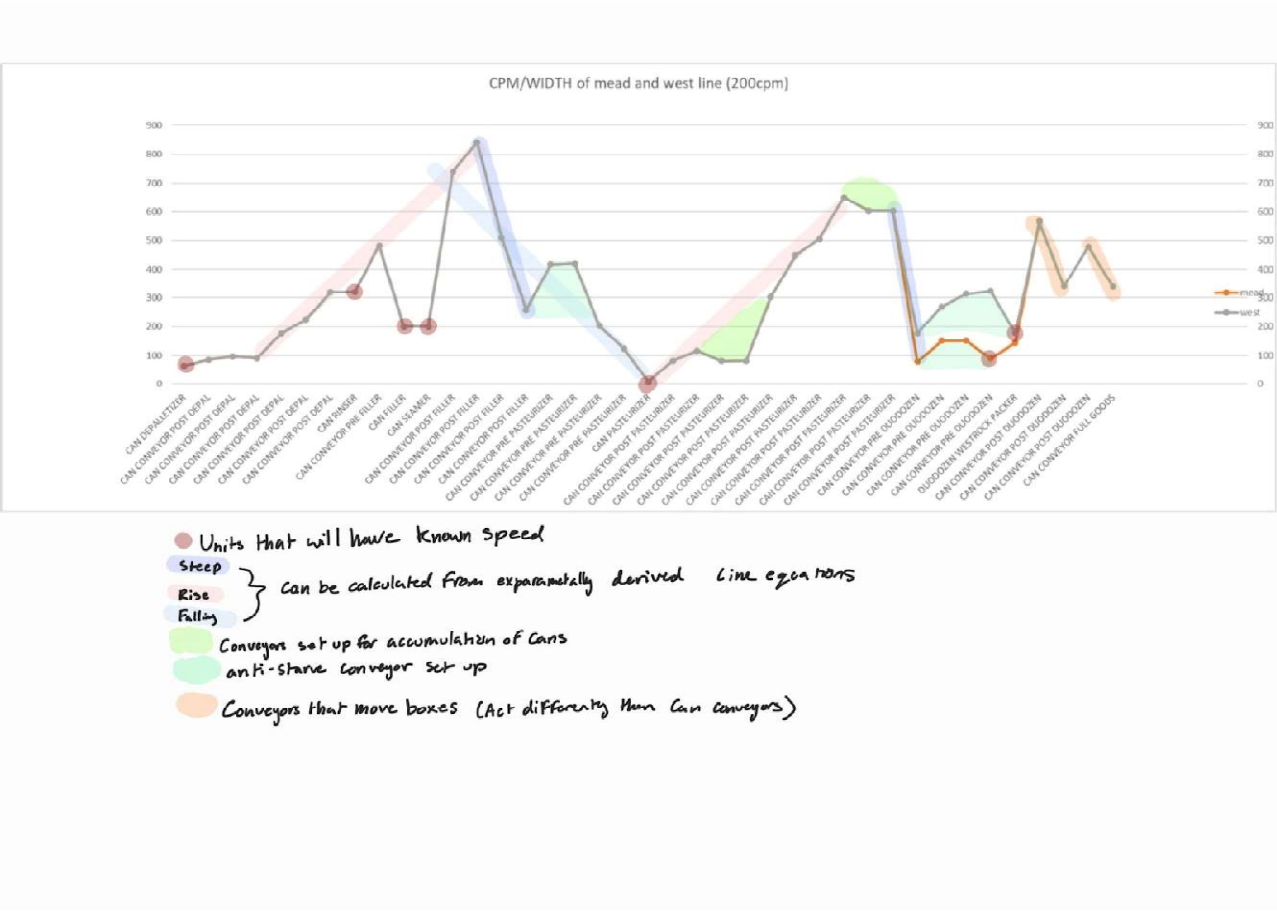
Notes On Encoder:

- It is possible to use rotary position encoders to measure the speed of the conveyors.
- They would have to be connected directly to the shafts of the driven roller. And then connected to a PLC to receive the data

We would have to figure out the relationship between each encoder pulse and the linear belt distance (i.e. radius of the roller x Pi / encoder pulse per revolution (also known as encoder resolution))

Link to Industrial Encoders:
https://www.rls.si/eng/products/rotary-magnetic-encoders?gclid=CjwKCAiAuaKfBhBtEiwAht6H7zGjI9dUnGln9fM3ctMHgJsQ3J9eXcG6TY8CUzKjCjyYGa9IUux93hoCBdkQAvD_BwE

9.1.2 Characteristics of Conveyors



Linear Interpolation Between 2 points:

$$x_i = x_1 + i \times \frac{(x_2 - x_1)}{(n+1)}: \text{ where } i = 1, 2 \ldots n$$

$$y_i = y_1 + i \times \frac{\left(y_i + 1 - y_1\right)\left(x_i - x_1\right)}{\left(x_2 - x_1\right)}: \text{ where } i = 1, 2 \ldots n$$

$$Speed\ of\ Conveyor\ after\ Pasturizer = (Cans\ per\ second\ into\ the\ pasturizer) \times (sum\ of\ conveyor\ widths)$$

$$Sum\ of\ Conveyor\ Widths = 4 + 47 + 4 = 55$$

$$Cans\ per\ second\ into\ the\ pasturizer = \frac{\left(Speed\ of\ Pasturiser\ cpm\right)}{60}$$