

ES96 Section C Final Report - Spring 2020

Section Leader: Dr. Jeffrey Paten

Teaching Fellows: Anahide Nahhal, Anthony Aybar

Students: Amy Brooke, Matthias Fischer, Josh Greenstein, Douglas Henze, Coleman Hooper, Nathan Le, Willie Meng, Cecil Myers, Jude Najjar, Davey Schoenberg, Evan Thompson

I. Executive Summary

Our team of undergraduate engineering students was tasked with tackling Sager Electronics' order accuracy issue in their Distribution Center in Middleborough, MA. In order to thoroughly understand this complex problem, the team visited the site and then created a detailed system map. This map encapsulated each step in the product flow, focusing particularly on the different points in the chain at which errors arose. The team also produced a stakeholder map which detailed the influences of and connections between all parties with a stake in the project. After fully analysing the problem space, the team came up with the following problem statement question:

How might we reach Sager's order accuracy target without significantly increasing the cost or complexity of operation?

Using these maps and this question, the team began visualising potential solution pathways and challenges. The team brainstormed a comprehensive list of possible strategies to solve the above defined problem, and then devised a system to first categorise and then evaluate these ideas. After evaluating all the proposed solutions using two tiers of criteria, the team narrowed down the solution space to three main ideas. These are described as follows:

A. Programmable Scales

This solution focused on automating the counting part of the picking process, which involves the use of scales. This is an error-prone procedure that often leads to quantity errors, the most common type of order error. Counting can be made easier and more accurate by using a central database of product weights and by retrieving order quantities from Sager's Warehouse Management System. We identified two strong commercial solutions that would have the capability to implement such a system and provide on-going support. We also created a web application to demonstrate this functionality. It is able to connect to both Sager's Warehouse Management System and a database of weights in Google Sheets. This program has the potential to be implemented on a small scale in the warehouse to show how such a system could be beneficial and to thus inform future communications with a commercial partner.

B. LED lights

This solution focused on tackling the problem of wrong part errors. This is specifically used in the context of inventory accuracy. An LED system will be connected to the warehouse management system (WMS) that will signal on LEDs in the inventory area that are associated with an employee's picking or putaway task. The system will differentiate which employees are working on a task in the shelving unit and assign them a color so that there is no confusion as to who is working where. Commercial solution options work very much like the system we are proposing, the most suitable for SAGER would be a custom system that involves large LED strips on each shelf. This system and any of a number of companies that provide it would provide long term support, just the initial setup of the system would be more expensive than the proposed LED solution. There are other commercial “pick-to-light” technologies, but none are better integrated into the current system than the full LED strip solutions.

C. Employee Engagement Program

This solution focused on tackling the overall issue of order inaccuracy from a more human-centric, employee-focused perspective. The program is intended to boost manager-production employee relations by offering more chances for feedback to be shared between the two parties. To do so, we created a questionnaire that can be administered once per quarter to elicit pointed feedback to help Sager identify when to retrain, what parts of the procedure are confusing, and how to best support its employees. Additionally, we identified strong, web-based commercial appraisal platforms to host the questionnaire and encourage frequent 360° feedback and communication.

II. Table of Contents

- I. Executive Summary
- II. Table of Contents
- III. Introduction
 - A. Background
 - B. Problem Motivation
 - C. History of Order Inaccuracy
- IV. Researching the Problem Space
 - A. Site Visit Findings
 - B. Stakeholders and Influence Map
 - C. Systems Map
 - D. Problem Statement
- V. Investigating the Solutions Space
 - A. Ideation Phase and Categorisation
 - B. Criteria and Process for Convergence

- VI. Solutions
 - A. Solution 1: Preprogrammed Scales
 - B. Solution 2: LEDs
 - C. Solution 3: Employee Engagement Program
- VII. Conclusions
- VIII. Appendix

III. Introduction



A. Background of Sager as a company

Sager Electronics is a Massachusetts-based distributor of Interconnect, Power, and Electromechanical components. Sager started as a single storefront in Boston in 1931, but was later acquired by TTI, Inc. a Berkshire Hathaway Inc. company in 2012. Sager currently employs 406 employees in 10 service centers across the US, and are headquartered in Middleborough, MA. The problem that Sager approached our team with is the order accuracy rate at their distribution center in Middleborough. They want our team to help them reach an order accuracy rate of 99.91%.

B. Motivate the problem

Order accuracy is a serious issue that can affect Sager in a multitude of ways. To establish the significance of the problem, we identified three key areas in which it affects distribution companies like Sager:

- **Reputation:** Sending out an incorrect order to a customer negatively impacts the customer's view of Sager's reliability, which reduces the likelihood of a customer doing business with Sager in the future.
- **Financial Cost:** Receipt of incorrect parts by a customer may set the customer back on important deadlines, resulting in Sager being responsible for covering the customer's financial losses due to the mistake.
- **Safety:** Installing incorrect components into a design may result in a safety hazard. For example, one of Sager's main areas of focus is in power. Installing an incorrect part which may not be rated to provide the necessary current for a certain device could result in overheating of that part, which can in turn result in failure of the part or worse.

C. History of order accuracy

To identify the problem, and the main processes within the distribution center that are causing it, Sager is currently relying on information from random audits and return merchandise authorizations (RMAs) from customers. Our team studied the data to identify the main areas of

errors, the significance of the problem, and the current rate of order accuracy. The data is presented in the figures below.

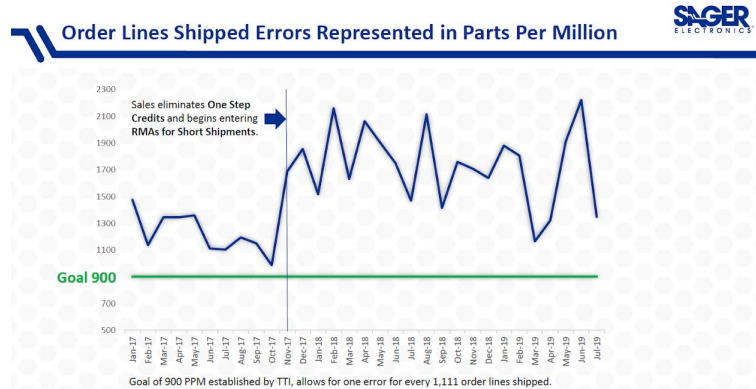


Figure 1: Initial data obtained from RMAs between Jan 2017 and July 2019 presented by Sager

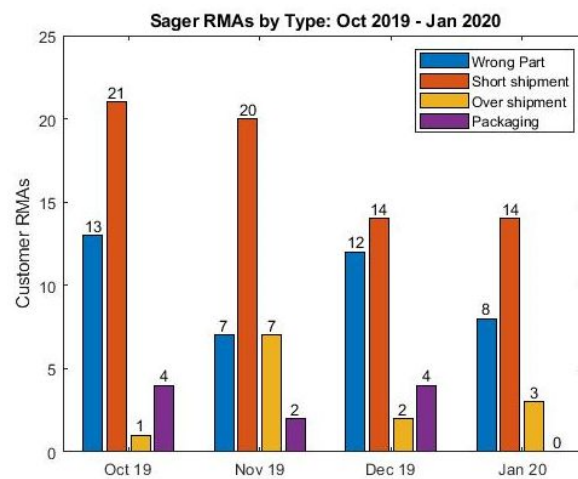


Figure 2: Most recent error data obtained from RMAs between Oct 2019 - Jan 2020

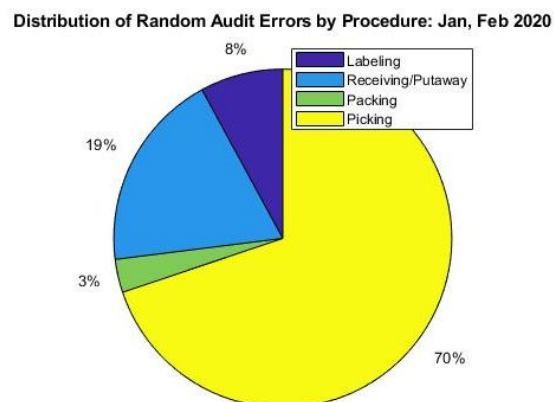


Figure 3: Distribution of errors across processes, data obtained from random audits performed in Jan & Feb 2020

IV. Researching the Problem Space

A. Site Visit Findings

With most of the team initially having only a surface-level understanding of the inner workings of distribution centers like Sager's, our team conducted a site visit of Sager's facility in Middleborough, Massachusetts to gain a deeper understanding of where errors could possibly occur within their operational processes. While visiting Sager's 1000-square foot facility, we were taken to various departments that were identified as delivery, receiving, putaway/picking, packing, and shipping. To help us illuminate how errors could happen, our team took notes and asked Sager employees questions in each area.

Delivery: Packages arrive at the loading dock primarily between 9:00AM and 11:30AM. Bulk orders are then moved to a staging area until they are put away, while smaller, single-line orders are processed by the receiving employees.

Receiving: In this area there are several benches with computers that employees have to stand at for the entirety of their shift. Once an order arrives at an employee's bench, they compare the purchase order of the item with the item's packing slip to ensure that the product received is correct. Features of the purchase order that they compare include product number, quantity, and country of origin. It is important to note that the boxes are not actually opened in this area, however—the products are just checked manually via the labels on the boxes and the paperwork.

If the product is correct, then its information is entered into the computer at the employee's bench and a label is printed that contains the product information. The computer also shows how much of the product is already available in the warehouse and, space-permitting, will ensure that product is put away in the same location. Finally, the printed label is adhered to the package, the package is placed in a yellow-colored tote, and the tote is sent to the putaway area via conveyor belt. Overall, the receiving process could generate errors due to it being highly manual. The employee who worked at this station told us that the serial numbers can be long (8-15 numbers) and look very similar to one another, making it seem reasonable that fatigue, for one, could negatively affect the accuracy of this system.

Putaway: Next, employees are assigned specific shelves from which they can pick from, with any items outside of one employee's shelf range being another employee's responsibility. This system that Sager operates on is called a "pick and pass" system. Once the packages arrive in this area on the conveyor belt, employees will pick them up, scan the barcode on the package to see where it is supposed to be shelved, and then go to that location. Once at the location, the barcode on the shelf location will be scanned, then the package's barcode again. There are some measures in place here to avoid errors. For example, the RF gun will beep loudly if an employee tries to put the wrong item in the wrong place. However, an important potential source of error is

if the correct shelf is scanned, but then the employee loses its location when turning around to pick up the corresponding package, resulting in them placing the package on a nearby (incorrect) shelf location, instead. Considering employees in this area can be responsible for putting away 200 packages in a 4-hour shift, which translates to just over a minute per package, this simple mistake could pose a major issue.

Picking: The picking process is similar and occurs in the same area as putaway. An employee's RF scanner tells them the location and quantity of the item they need to pick, they go to that location to retrieve the package containing the item, manually take the correct quantity of the desired part (or weight-count it if the parts are small), bag and seal it, and produce a new label for it that is scanned before placing the bundle in a tote to be carried to the packing stations. If the correct quantity is not available at that location, the RF scanner will tell them an alternate location in the warehouse where additional pieces of that specific part can be found. Due to manual counting and discrepancies in scale accuracy when trying to count a large number of parts, the picking process can therefore be prone to quantity errors.

Packing: Here, the employee scans the label on the product again and cross-checks it with the purchase order before generating a packing slip and placing the items in a box. The packing employees do not recount the parts at this stage—they just trust that the picking employee counted the items correctly.

Shipping: Finally, the boxes are weighed, scanned, and placed on pallets for shipping companies such as UPS and FedEx to pick up.

Our site visit allowed us to better understand the RMA data that Sager provided to us, and we were able to see how easily mistakes that occur during the putaway and picking processes could have a large effect on the error trend that initially alarmed Sager.

B. Stakeholders and Influence Map

A Stakeholders and Influence map documents all the parties that have a stake in and influence over the problem, and consequently also the solutions. It also shows the relationships and interactions between the different parties, laying out the power structure. It allows for a deeper understanding of the system and so informs the choice of the most effective solution.

The orientation follows the flow of decisions through Sager. Decisions move through the president Frank Flynn, who receives input and ideas from his direct subordinates, the vice presidents. The line weights associated are in accordance with the impact a stakeholder has on a decision. The president has a large impact on subordinates, and managers have a large impact on their employees. Distribution center workers have an impact on customers and mail carriers through their direct interaction with the products. This is the level we want to target because the impact on a customer is the level at which an RMA is triggered. The manufacturers directly

impact distribution center workers through the way they ship and package items, as this affects the ease at which the DC workers can do their job.

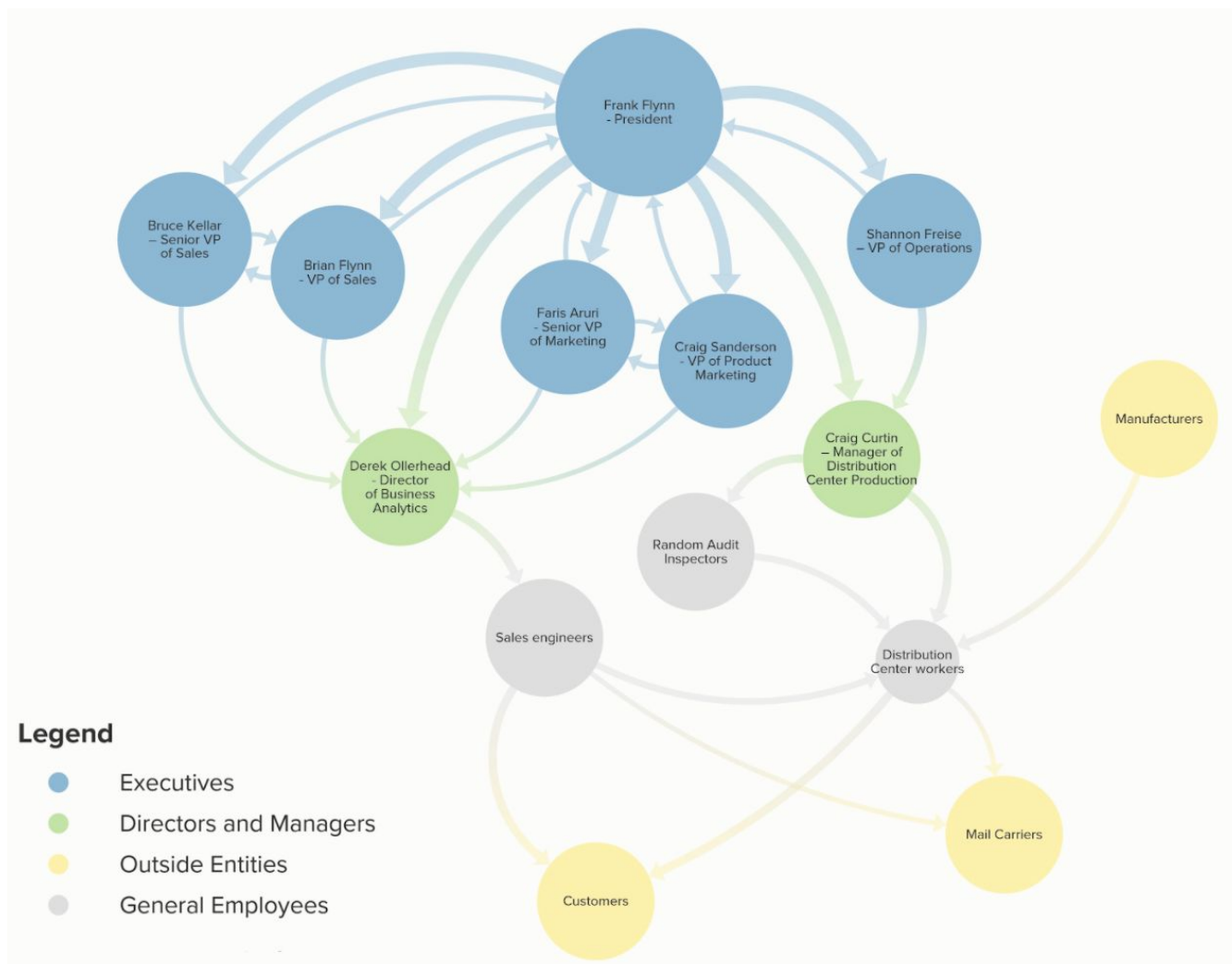


Figure 4: Stakeholders and Influence Map

The presence of order inaccuracy impacts not only the customers, it also impacts the distribution center workers, sales people, and even affects the reputation of the manufacturers that supply Sager. Since the product handling by the distribution center workers has been identified as the level that primarily influences order errors, we narrow our focus to the product flow in the creation of the systems map.

C. Systems Map

A systems map is a diagram that breaks down a large complex system into its constituent parts so that it is easier to visualise the interactions between the constituent components. This allows for easier identification of the root cause(s) of the problem specified in the problem statement. It also

provides an understanding of the interventions that might be most effective, and where in the system these interventions would be implemented.

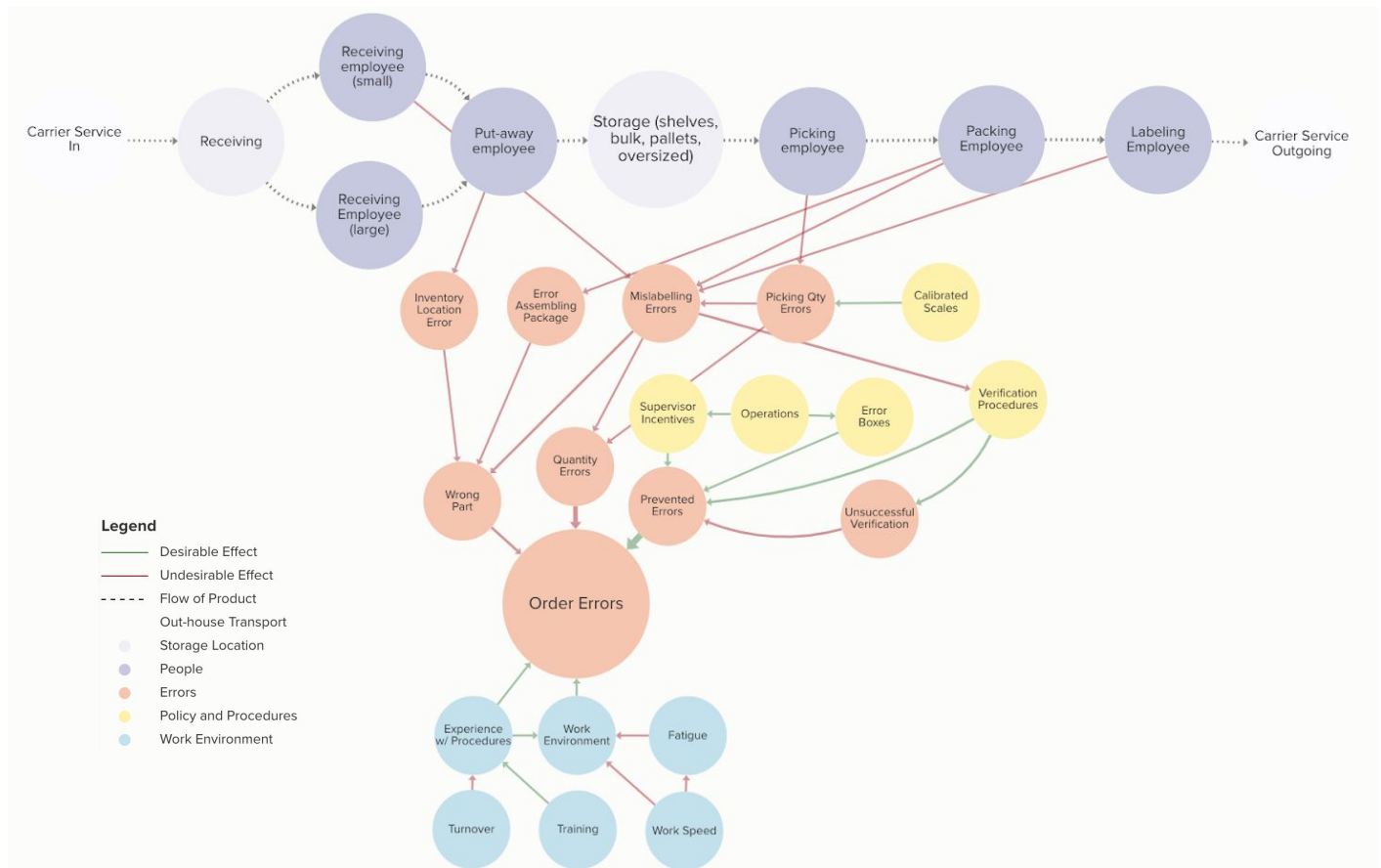


Figure 5: Sager Electronics Order Accuracy Systems Map

Our T-shaped systems map is divided into three sections. First, there is the product flow across the top, detailing all the points along a product's path through the distribution center. These points include external carriers, physical warehouse storage locations and employee handling. The second section (along the stem) details the different errors that occur. It includes the various linkages to the product flow, showing where in the distribution chain a particular type of error occurs. The third section (at the bottom) details the other important factors that impact, both positively and negatively, the order accuracy, such as policies and procedures, or characteristics of the work environment.

As displayed in the legend, the arrows are colored such that a green arrow indicates a desirable effect on the overall order accuracy, and a red arrow indicates an undesirable effect, thus contributing to more order errors.

This map enabled us to identify the two main causes of errors, wrong part and quantity, as well as where in the system they occur, and what factors affect them. All of this greatly informed our choice of the most effective solution approaches.

D. Problem Statement

Once we had a clearer idea of the situation at Sager and what the company's needs, we started our first iteration of the problem statement. The problem statement had four criteria that it was supposed to meet:

1. Summarize the background facts and current state
2. Define the problem and the conditions when the problem is present
3. Identify the impact of the problem
4. Describe the targeted end state

As a result, we developed 4 questions that framed each one of the criteria in a way that directly related to the problem at Sager. They were as follows:

1. What is currently in place to mitigate errors at Sager Electronics?
2. How do errors arise and what type of errors have the most impact?
3. What effect do order inaccuracies have on the company?
4. What do we want to achieve in regards to error mitigation? Is it feasible?

After several iterations, we created a rough draft that met the desired criteria and addressed the questions above by drawing on information in the stakeholder map, system map, and background. However, whilst we were still in the problem space, our problem statement was still vague. Once we made significant progress in the solution space, we were in a better position to further clarify and truly finalize our problem statement. It reads as such:

“Order inaccuracies in the Sager Electronics distribution process negatively impact its **customer relationships, profits, and safety of the end users.**

The largest sources of order inaccuracies at Sager are **short shipments** and shipments of **incorrect parts** as a result of **human error** during the **putaway** and **picking** processes. Current efforts to minimize errors in-house are automation, the use of random audits, anonymous error reporting, and multiple verification points along the distribution chain. These procedures are insufficient for reducing the order inaccuracy from ~1400 to below Sager's **target of 900 parts per million.**

How might we reach the order accuracy target without significantly increasing the cost or complexity of operation?”

V. Investigating the Solution Space

A. Ideation Phase and Categorisation

After researching the problem space and reaching our final problem statement, the time came to begin exploring solutions that will effectively address the problems identified in the problem statement. Before diving into the brainstorming process, the team started by listing the main areas of errors and their identified causes; the purpose of this was to aid the ideation phase by giving people problems to focus on when thinking of solutions. Following this initial step, the team started the actual ideation process.

To ensure that our team was considering all available options and not prematurely discarding ideas, we began the ideation phase with a group brainstorming session where we collected ideas from all the members of the team. Every idea was written down without considering practicality, effectiveness, or any other criteria that might hinder its implementation. After collecting all the solution ideas from the team, the solutions suggested were grouped into five categories, to make it easier to converge the solutions later on. This resulted in the following list of divergent solutions:

- **Solutions related to scanning procedures:**
 - Localized RF scanner
 - Optimizing the number of scanning steps at each stage of the procedure
 - Using LED strips on the shelves for one of two purposes
 - Pre Queuing the next location that a box needs to be placed on
 - The LED strip would light up once the shelf is scanned to indicate which shelf was scanned, and which shelf the box should be placed on
 - Standardizing the location of Sager specific labels on boxes
 - Holsters on employee's belt to hold the RF scanner when not in use
 - Image recognition software to be used instead of scanning
- **Solutions related to the scale used for counting part quantities:**
 - The use of more advanced scales. This was proposed in a variety of different ways
 - Scales with product weights pre-saved in the system
 - Scales that can communicate with the RF scanners currently in use
 - Scales that have the ability to weigh the final order and indicate the number of parts in it
 - Improvement of the scale calibration process
- **Solutions related to the warehouse:**

- Safety rails for the conveyor belt in the warehouse
- Color coded shelves
- Simple automation to help with lifting
- Camera for each set of shelves to detect whether boxes are in the proper location
- Improvement of lighting in the warehouse
- Location tracking for packages inside the warehouse
- Grouping the same products together in one location in the warehouse
- **Solutions related to procedures and rules:**
 - Modification of procedures where necessary
 - More advanced data collection methods
 - Improvement of the organization of parts in the warehouse
 - Intentional part overshipment
 - Descriptors of manufacturer labels
- **Solutions related to employee relations and work environment:**
 - Employees cross-checking each other's work in the picking process
 - Incorporating more breaks to reduce fatigue
 - Employee engagement program and weekly employee surveys
 - Refining retraining procedures based on employee feedback
 - Shift flexibility for employees who are not feeling well
 - Visual reminders of the importance of workers' roles

B. Criteria Process for Convergence

In order to choose which solutions we wanted to pursue as a team, we had to develop different criteria to apply to our pool of ideas to reduce them until we were left with the best options. These criteria were split into three tiers:

Tier 1:

- *Effectiveness*: Did the idea address our problem statement?
- *Total Cost*: Is the idea capable of being completed within the budget provided to us by Sager?
- *Feasibility*: Can we actually do what the idea requires? Is it practical?

Tier 2:

- *Simplicity of Implementation*: Is the idea going to be easy or difficult to implement?
- *Compatibility with Sager*: Is the idea going to add unnecessary steps to the workflow? How willing would Sager be to implement this solution?
- *Impact on Efficiency*: Is the idea going to hinder Sager's current efficiency in any way?

Tier 3:

- *Engineering Interest:* Is the idea something that we, as a team, are interested in pursuing?

After choosing the criteria, we then gave each brainstormed idea a score of 0-3 on tier 1 and 2 criteria. Any idea that had a 0 in any criteria in tier 1 or tier 2 was automatically failed, while the rest of the solutions were further evaluated. In tier three, we simply voted yes or no to gauge our interest in pursuing an idea further or not.

This process left us with five solutions: using LED strips on the shelves, color coding shelves, improving scale calibration, using more advanced scales, and correlating retraining procedures based on employee feedback. From this stage, we grouped together and refined the “LED strip” and “color coded shelves” ideas into one LED-based solution to tackle putaway and picking errors. We also grouped and refined the “improvement of scale calibration” and “use of more-advanced scales” ideas into one scale-based solution to address quantity errors. Finally, we refined the “correlation of retraining procedures based on employee feedback” idea into a more targeted employee-based solution to lessen human errors in the process.

VI. Solutions

A. Programmed Scales

1. Introduction

When we considered the problem space we noticed how the use of scales to count large numbers of items during the picking process could contribute significantly to quantity errors in the final orders. To reduce the potential for error, we focused on increasing the automation of this part of the picking process as far as possible. The relevant aspects of this automation would on the one hand be some form of central weight storage. This would remove the step of having to determine the product weight each picking process using a sample size. This procedure has a significant potential for error, since a small sample size will make the product weight inaccurate, and counting a larger sample size by hand can also result in weight errors. Centralizing the product weight can reduce the risk of using incorrect product weights. A further addition that could improve the automation and further reduce the risk of using an incorrect product weight would be to optimize the retrieval of the product weight from the central weight database. This could be done by trying to retrieve the order a specific employee is currently picking for on the screen of their RF-Scanner or an alternative method requiring minimal user input. This would further reduce the possibility for errors.

In structuring our approach to satisfying these goals, we decided to focus on two main paths. On the one hand we tried to create our own solution using a cheap scale with USB-connectivity and a VPN connection to a Sager testing database. Such a solution coded by ourselves would be quite limited when it comes to long-term support. Therefore, we also researched existing commercial

solutions with different levels of functionality and collected information on these below. A full explanation of this analysis follows.

Upon completion of this analysis, we determined the two best commercial scale options: the Avery Weigh-Tronix ZK840 and the Mettler Toledo ICS445 series. Both can be fully customised to Sager's needs and fall within the price bracket. The Avery Weigh-Tronix ZK840 is an advanced, accurate, robust scale, with a range of suitable tolerances and a user-focused design. The Avery PLU look-up software is fully-customizable and has the ability to connect with both the WMS and an external database of product weights. Avery is a respectable commercial partner that offers full installation and sustained support. Additionally, the Avery offers a wide range of connectivity, able to connect to RF scanners, printers and remote displays. The Mettler Toledo ICS445 series has a number of options with different sizes and tolerances. It can be wired to an RF scanner to read barcodes. It is physically robust, having been made for a warehouse. Using Mettler Toledo's software, the scale can connect to a central database that holds up to 30,000 product weights. Mettler Toledo has full-service setup and analysis, including the ability to set up reading weights from the WMS database, and the applications team may also be able to set up reading orders from the WMS as well.

2. Research of Existing Product Options

We researched a variety of different scales. These included two scales used by TTI in the US and Germany, a scale that Sager was considering purchasing (the Ohaus Ranger 7000), as well as two scales with advanced functionality and connectivity to a central database. We evaluated the options using the template below, in an effort to determine the strongest options for Sager.

Template for Analysis:

Metric	Details
1. Hardware Components	What will it physically consist of, and are all components provided by the company?
2. Space and power requirements	What is the approximate square footage occupied? How is it powered?
3. Cost	What is the total cost and the breakdown?
4. Robustness	Will it be able to withstand and sustain the warehouse conditions?
5. User Interface	What will the final user experience involve? What information will be displayed, what are the user inputs, how

	will it physically appear?
6. Connectivity to WMS	Will the scale connect to the WMS and how?
7. Weight Database capabilities	Does the scale have its own weight database, either internal or external? If so, what is the capacity and how is the connection made? Does the scale have the ability to connect to Sager's weight database?
8. Connectivity to Scale	How can you connect with the scale to retrieve the current weight reading? What softwares does it support and not support?
9. Tolerances and Range	Relative to Sager's requirements, what is the range of weights it can weigh, the accuracy and precision?
10. Installation and Testing	What will implementation and setup look like? Is testing possible? What will that look like?
11. Long term support	What will sustained support look like for this option? (the company, Sager IT, or another commercial partner?) What happens if there is an issue with a scale or the system?
12. Customizability	Is this system able to fulfill Sager's exact needs? In what ways does it and in what ways is it limited?
13. Other	Are there any other defining features that differentiate this product from the others?
14. Summary of Value	What are the main selling points as to why this solution is good?
15. Summary of Limits and Challenges	What are the major issues and significant drawbacks? Why are they issues and why are they hard to solve, or can't be solved by us?

After carrying out the above analysis on all the scales mentioned above, we found that there were essentially three main levels of complexity:

1. Scales that have counting capabilities
2. Scales that have some sort of product weight database

3. Scales that have the ability to connect to the WMS

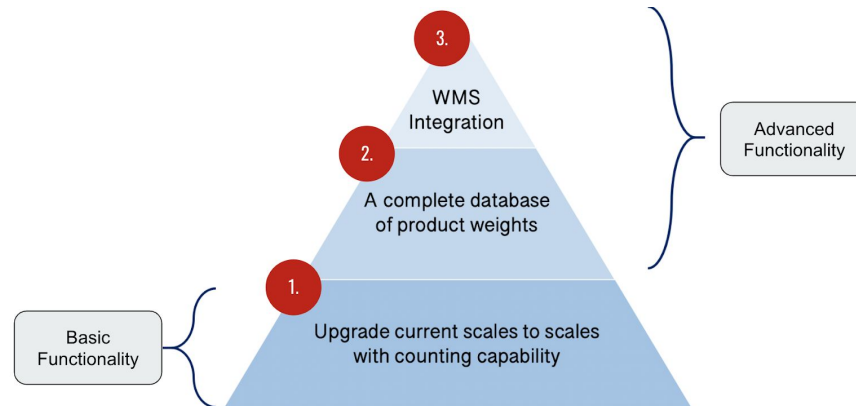

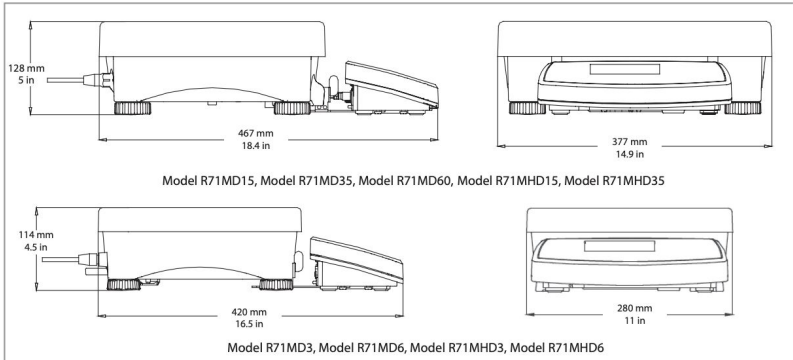



Figure 6: Levels of Complexity of Scale Automation

Notably, some scales lie right on the boundaries between these levels, and some scales have the flexibility to offer all three options. We begin by presenting the scales that have counting capabilities, and no (or limited) product weight storage.

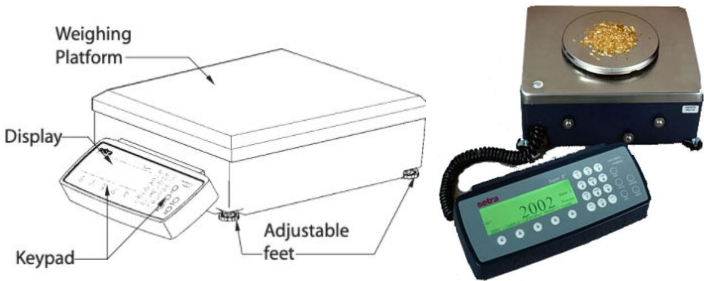
Option 1: Ohaus Ranger 7000 Scales (researched by Sager)

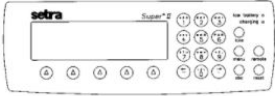
Metric	Details
1. Hardware Components	<ul style="list-style-type: none"> Box-like scale with display/indicator 
2. Space and power requirements	 <ul style="list-style-type: none"> 100-240 VAC, 50/60 Hz

	<ul style="list-style-type: none"> Optional: rechargeable lithium battery with 12 hours battery life of continued use
3. Cost	<ul style="list-style-type: none"> \$1728 each (as per quote received by Sager)
4. Robustness	<ul style="list-style-type: none"> Die cast housing, stainless steel platform Built to withstand heavy duty usage
5. User interface	<ul style="list-style-type: none"> Graphic LCD (10.5 × 4.6 × 2.8 in) and keypad Seems like the target weight/count can be displayed Can connect to discrete I/O kit for external check lights Display is detachable 
6. Connectivity to the WMS	<ul style="list-style-type: none"> None: scale readout can be obtained but no software to do anything with it besides checking weight/count of a sample against target limits, which is already a built-in functionality
7. Weight Database capabilities	<ul style="list-style-type: none"> No built in capabilities Does not seem possible to connect with an external database
8. Connectivity to Scale	<ul style="list-style-type: none"> RS232 and USB ports to connect to a computer or printer (for USB, port appears as a virtual RS232 to the windows application program) Optional: ethernet, digital I/O
9. Tolerances and Range	<ul style="list-style-type: none"> 150% overload capacity 9 different models with widely varying resolutions and readabilities (meet Sager's requirements)
10. Installation and Testing	<ul style="list-style-type: none"> Does not appear to be any installation other than turning the scale on and potentially running the provided software application on a Windows PC
11. Long term support	<ul style="list-style-type: none"> OHAUS Regional Service center offers various repair and

	support services
12. Customizability	<ul style="list-style-type: none"> Has a number of custom applications, including what appears to be a counting application, however the user has to set the piece weight as there is no form of database Requires a PC running windows
13. Other	<ul style="list-style-type: none"> Weight units: kg, g, lb, oz, lb:oz ISO 9001:2008 certified MA Office in Canton
14. Summary of Value	<ul style="list-style-type: none"> Fairly affordable scale that is more advanced than the current DC-190s, with a basic counting application
15. Summary of Limits and Challenges	<ul style="list-style-type: none"> No connection to a central weight database or the WMS Counting application very minimal

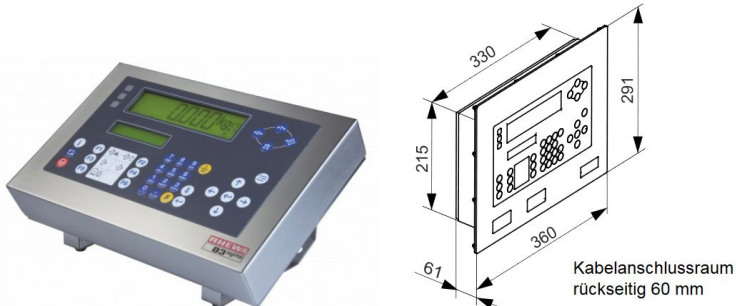
Option 2: Setra Super II, Nicol Scales (used by TTI)

Metric	Details
1. Hardware Components	<ul style="list-style-type: none"> Box-like scale with display and keyboard 
2. Space and power requirements	<ul style="list-style-type: none"> Input 120 VAC, 60 Hz or 230 VAC, 50 Hz Option for portability with internal battery

	<p>Dimensions:</p> <p>25 - 50 Kg Platform 13" W x 15" D (33 x 38 cm) Controller 10.75" W x 4.38" D x 2.5" H (27 x 11 x 6 cm) Overall 13" W x 19.38" x 4.5" H (33 x 48 11.5 cm)</p> <p>12 - 16 Kg Platform 11.5" W x 9.5" D (29 x 24 cm) Controller 10.75" W x 4.38" D x 2.5" H (27 x 11 x 6 cm) Overall 11.5" W x 14"D" x 4.75" H (29 x 35 12 cm)</p> <p>2 - 5 Kg Platform 6.2" Diameter Controller 10.75" W x 4.38" D x 2.5" H (27 x 11 x 6 cm) Overall 11.5" W x 14"D" x 4.75" H (29 x 35 12 cm)</p>
3. Cost	<ul style="list-style-type: none"> • ~\$2000 per scale
4. Robustness	<ul style="list-style-type: none"> • Sturdy and durable, with all metal construction
5. User interface	<ul style="list-style-type: none"> • Graphic display with customizable messages • Option for keyboard connection  <p style="text-align: center;">Keypad</p>
6. Connectivity to the WMS	<ul style="list-style-type: none"> • Not possible
7. Weight Database capabilities	<ul style="list-style-type: none"> • Can store 750 product weights plus part number, 16 character weight descriptions, average piece weight, tare, set points
8. Connectivity to Scale	<ul style="list-style-type: none"> • Ports to connect to printer (out), scanner (in) and PC (bi-directional RS232) • Can connect one controller to many bases (i.e. scales)
9. Tolerances and Range	<ul style="list-style-type: none"> • Internal resolution of 1 in 1 million • Counting capacity of 10 million pieces
10. Installation and Testing	<ul style="list-style-type: none"> • Handled by company
11. Long term support	<ul style="list-style-type: none"> • Tried and tested support given that they are used by TTI
12. Customizability	<ul style="list-style-type: none"> • Not very customizable, it is simply a counting

	scale
13. Other	<ul style="list-style-type: none"> • ISO 9001 certified • Offices in MA • Can generate barcode labels
14. Summary of Value	<ul style="list-style-type: none"> • Sturdy, reliable scale with basic counting capabilities that has been tried and tested by TTI
15. Summary of Limits and Challenges	<ul style="list-style-type: none"> • It does not connect to the WMS • It has as a limited database size



Option 3: 83sigma Indicator (used by TTI Germany)

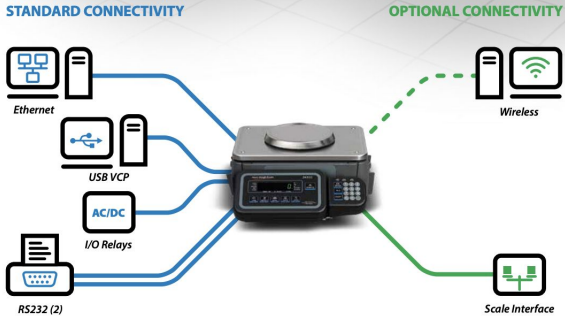
Metric	Details
1. Hardware Components	<ul style="list-style-type: none"> • Small box (22 x 33 x 18 cm), requires scale connected 
2. Space and power requirements	<ul style="list-style-type: none"> • Input 110-230 V, 50/60 Hz, power adaptor integrated, mounting might take more space
3. Cost	<ul style="list-style-type: none"> • 830 Euro, lower depending on number bought
4. Robustness	<ul style="list-style-type: none"> • Corrosion-free steel casing, keyboard withstands most aggressive chemicals, >1M key input cycles
5. User interface	<ul style="list-style-type: none"> • Simple, small LCD indicator, LEDs to further indicate counting status
6. Connectivity to the WMS	<ul style="list-style-type: none"> • Possible, but requires connected PC that connects to WMS via software (software & support not provided)
7. Weight Database	<ul style="list-style-type: none"> • Can store at least 200, likely more than 1000 parts with

capabilities	weight, but no built-in central database connectivity
8. Connectivity to Scale	<ul style="list-style-type: none"> • Connectivity to two scales, output via USB, ethernet, wifi module optional, serial connection to PC possible
9. Tolerances and Range	<ul style="list-style-type: none"> • Depends on scale
10. Installation and Testing	<ul style="list-style-type: none"> • Impossible for us as a class
11. Long term support	<ul style="list-style-type: none"> • Seems provided for the indicator itself
12. Customizability	<ul style="list-style-type: none"> • Customizable through PC serial connection and data evaluation
13. Other	<ul style="list-style-type: none"> • ISO 9001 certified • Built and sold in Germany, may be hard to get in US • Connection to Label printers • Weight logging to USB connection quite easy, may be useful for data collection
14. Summary of Value	<ul style="list-style-type: none"> • Quite versatile, potential for further connections and software extensions, potentially even designed and supported by supplier • Highly robust and reliable, and supported
15. Summary of Limits and Challenges	<ul style="list-style-type: none"> • Valuable extensions require additional non-existing software, no support existing

Option 4: Avery Weigh-Tronix ZK830

Metric	Details
1. Hardware Components	<ul style="list-style-type: none"> • Boxlike scale with a display (indicator fixed to the front) • Options for draft shield, and remote keypad (pictured below)


	
2. Space and power requirements	<ul style="list-style-type: none"> • 3 different options: 12x14in plate, 9x12in plate, round plate of 6in diameter • Screen does not draw much power • AC or battery operated
3. Cost	<ul style="list-style-type: none"> • \$1600 for 70lb capacity
4. Robustness	<ul style="list-style-type: none"> • Durable, rugged and robust, with strong casing (die-cast aluminum) • 1100% overload and spring breakaway protection mechanism
5. User interface	<ul style="list-style-type: none"> • Easy to use six-key indicator designed to keep operator training to a minimum • Can mount the indicator to desk/wall • User focused design: highly visible display, foldable handles, battery pack and clamp for portability 
6. Connectivity to the WMS	<ul style="list-style-type: none"> • Not possible
7. Weight Database capabilities	<ul style="list-style-type: none"> • Capacity for storing up to 40 piece weights internally when fitted with remote keypad • Can also link with external PC controlled databases, using simple SMA commands to import and export data from other sources
8. Connectivity to	<ul style="list-style-type: none"> • Standard range of ports (Ethernet (TCP/IP), RS232, USB,


<p>Scale</p>	<p>I/O relays)</p> <ul style="list-style-type: none"> • Can connect to PCs, printers, scanners, remote displays • Dual base option: allows items to be sampled on a smaller capacity base, while batch counting takes place on a larger capacity base 
<p>9. Tolerances and Range</p>	<ul style="list-style-type: none"> • Capacities from 2lb to 175lb • Minimum of 10mg • Accuracy of 99.7% (10 decimal places) • Digital weighing technology → detect minor weight changes, count smaller parts, count faster and return to zero quicker • 100,000 division resolution
<p>10. Installation and Testing</p>	<ul style="list-style-type: none"> • Handled completely by Avery technicians
<p>11. Long term support</p>	<ul style="list-style-type: none"> • From the website: “We are committed to providing high quality, lifetime service support for all of our weighing equipment. Dedicated technicians are on hand to install and maintain your equipment, including regular calibration, servicing and repair.”
<p>12. Customizability</p>	<ul style="list-style-type: none"> • Count accumulator function (useful for to keep a running total with large quantities) and check counting function (over, under accept windows) • Says it is ideal for most packing operations • Own programming language (Avery Weigh-Tronix Lua)
<p>13. Other</p>	<ul style="list-style-type: none"> • Certified with ISO 9001 • HQ in Minnesota & Distributor in Worcester, MA

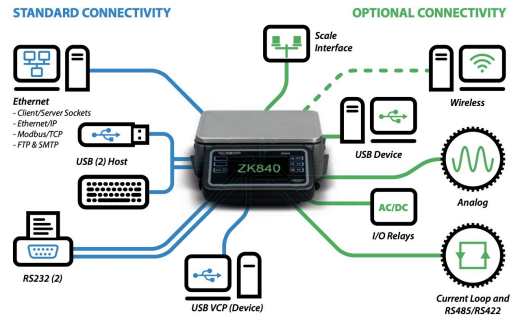
	<ul style="list-style-type: none"> Units of measure (kg, oz, g, lb, lb/oz)
14. Summary of Value	<ul style="list-style-type: none"> Very user orientated, accurate scale Simple to use check count and count accumulator functions Ideal for warehouse conditions
15. Summary of Limits and Challenges	<ul style="list-style-type: none"> Limited database functionality (only 40 internal PLUs) and connection to an external database requires a computer

This concludes the analysis of the scales that have basic counting capabilities, and limited or no product weight storage. Below we present two more advanced options, the Avery Weight-Tronix ZK840 and the Mettler Toledo ICS445, both of which have the flexibility to satisfy both level two (weight database) and level three (WMS connectivity) articulated above.

Option 5: Avery Weigh Tronix ZK840 Counting Scale

Metric	Details
1. Hardware Components	<ul style="list-style-type: none"> Boxlike scale with a screen (indicator fixed to the front) 
2. Space and power requirements	<ul style="list-style-type: none"> 3 different options: 12x14in plate, 9x12in plate, round plate of 6in diameter Screen does not draw much power AC or battery operated
3. Cost	<ul style="list-style-type: none"> \$2450 for 10 lb scale with standard calibration 10 lb. with 0.0001 lb. graduations. Options <ul style="list-style-type: none"> \$1750 PLU Lookup external pc database software for up to 32 scales (see 7) \$510 Ztools configuration and development suite (V2.x.x.x) (see 12)
4. Robustness	<ul style="list-style-type: none"> Durable and robust, with strong casing (die-cast aluminum)


	<ul style="list-style-type: none"> • 1100% overload and spring breakaway protection mechanism
5. User interface	<ul style="list-style-type: none"> • 5.3" x 2.75" touch screen • Displays graphics and user prompts (these can be pre-defined) • Its embedded web server allows for a wide range of data to be displayed on a web page on a PC or mobile device • Simple and intuitive user interface • Can mount the indicator to desk/wall • Optional discrete I/O lighting system as below 
6. Connectivity to the WMS	<ul style="list-style-type: none"> • Ability to connect to the WMS with custom software
7. Weight Database capabilities	<ul style="list-style-type: none"> • Built in database stores up to 5000 piece weights • Can purchase the PLU lookup software for \$1750 to connect up to 32 scales and create a database for hundreds of thousands of parts • This PLU lookup software is unnecessary for one scale • Database fields: Part number - Alphanumeric (20 characters), Description - Alphanumeric 40 characters long, one to three lines, Lot /location, Piece weight, Tare weight local base, Tare weight for second remote base, Stock on hand & Upper and lower check count limits • Internal expanded data storage can hold up to 4GB extra storage data , ZK840 has one Micro SD slot that is compatible with most Micro SD cards from 4GB to 32GB
8. Connectivity to Scale	<ul style="list-style-type: none"> • Full range of ports (Ethernet (TCP/IP), RS232, USB, I/O relays) • Can connect to PCs, printers, scanners, remote displays,

	<p>memory cards</p> <ul style="list-style-type: none"> Can connect the indicator to several scales 
9. Tolerances and Range	<ul style="list-style-type: none"> Capacities from 2lb to 175lb Readability of >3.5 million divisions Accuracy greater than 99.75% Minimum of 10mg 1 billion internal count resolution enables accuracy down to 10 decimal places
10. Installation & Testing	<ul style="list-style-type: none"> Installation and software development will be done by Avery technicians for a fee. Set-up and calibration, set up for counting parts, check operation Figured at 3 hr @ \$95/hr (preferred rate for semi-annual PM contract customers)
11. Long term support	<ul style="list-style-type: none"> From the website: “We are committed to providing high quality, lifetime service support for all of our weighing equipment. Dedicated technicians are on hand to install and maintain your equipment, including regular calibration, servicing and repair.”
12. Customizability	<ul style="list-style-type: none"> Range of applications: checkweighing, balance and density weighing, grading, fully custom applications Example Custom Application <ul style="list-style-type: none"> Box with counted product would be placed on the scale Custom scanner that was sold to the client would scan the label on the box Order information, product id, product weight, order count, etc. would all automatically be sent to the

	<ul style="list-style-type: none"> scale <ul style="list-style-type: none"> ○ A red or green light would then turn on based upon if the weight was wrong or correct ● Ztools configuration and development suite (V2.x.x.x) (\$510) <ul style="list-style-type: none"> ○ If you want to reconfigure and save set-up and/or write custom Lua software for ZK840 ○ Allows you to alter program in computer and run it on simulator, then send it to scale ● Alternative Free PLU Editor, allows you to import/export CSV files with PLU. Also allows you to average piece weight and other fields from your computer to ZK840 using a USB drive ● Own programming language (Avery Weigh-Tronix Lua)
13. Other	<ul style="list-style-type: none"> ● Provides real-time stock data and statistical analysis to improve product line efficiency ● Certified with ISO 9001 ● HQ in Minnesota, Distributor in Worcester, MA ● Units of measure (kg, oz, g, lb, lb/oz)
14. Summary of Value	<ul style="list-style-type: none"> ● Fully programmable so highly customisable, with all relevant capabilities! ● Offers sustained support from a respectable commercial partner ● User-focused design
15. Summary of Limits and Challenges	<ul style="list-style-type: none"> ● The Avery ZK 840 is a fully programmable robust counting or weighing scale. The custom applications allow the scale to successfully work in any situation. However, the scale and the software is expensive.

Option 6: Mettler Toledo ICS445

Metric	Details
1. Hardware Components	<ul style="list-style-type: none"> ● Square scale with display and button inputs. Multiple models with slightly different features available.

	
2. Space and power requirements	<ul style="list-style-type: none"> • 7.9 in x 9.5 in x 3.7 in for the 6SM/f. • 11 in x 13.8 in x 4.3 in for the 15LA/f. • Has both battery and AC options.
3. Cost	<ul style="list-style-type: none"> • \$1000-\$2000 depending on the exact sub-model.
4. Robustness	<ul style="list-style-type: none"> • Made for the warehouse. "Superior weighing electronics for excellent repeatability combined with a durable die-cast housing with IP65 protection deliver reliable results and make the ICS scale range a perfect fit for industrial environments [sic]. Optional battery power and wireless functions enables the scale for mobile use."
5. User interface	<ul style="list-style-type: none"> • Standard buttons/display. Can use barcode scanner.
6. Connectivity to WMS	<ul style="list-style-type: none"> • Yes. Their applications team can set it up, then Sager pays a regular service fee for maintenance. Can contact MT's applications team to discuss further details.
7. Weight Database capabilities	<ul style="list-style-type: none"> • Connects to manufacturer's database software. Can import weights via csv file
8. Connectivity to Scale	<ul style="list-style-type: none"> • Ethernet, RS232, WLAN or USB to its own software. Does not seem to have connections other than with its own software.
9. Tolerances and Range	<ul style="list-style-type: none"> • Different options. Best options include 6kg maximum with 0.1g or 0.01g tolerances, or same tolerances with 15kg
10. Installation and Testing	<ul style="list-style-type: none"> • Mettler offers extensive services. "Leverage the expertise of an authorized METTLER TOLEDO Service Technician to configure your scale system to perform at its best.

	<p>Through tailored configuration of your equipment, we can optimize its performance and functionality with:</p> <ul style="list-style-type: none"> ○ Application configuration to optimize applications ○ Data setup to ensure your data is properly loaded at startup ○ Integration of weighing equipment into other systems."
11. Long term support	<ul style="list-style-type: none"> ● Mettler Toledo offers long term support
12. Customizability	<ul style="list-style-type: none"> ● Uses Mettler software. Potentially: ● Connections with peripheral devices and systems ● Integration with software applications
13. Other	<ul style="list-style-type: none"> ● Sager has worked with Mettler Toledo in the past
14. Summary of Value	<ul style="list-style-type: none"> ● Mid-range scale that stores up to 30,000 item weights and also stores transactions. Company looks to have extensive customer support. WMS can be integrated.
15. Summary of Limits and Challenges	<ul style="list-style-type: none"> ● Expensive. Unclear how to integrate into the WMS.

These are two very reliable, and fairly affordable commercial options that offer a full range of customizable options, from being simple counting scales, to incorporating a product weight database, to full integration with the WMS. It may be difficult for Sager to decide upfront which of these capabilities will ultimately be the most effective at reducing counting errors (and potentially also increase efficiency). This is where we believe our solution could play a critical role. For consistency, below we apply the same analysis to our custom solution.

Option 7: Our Custom Solution

Metric	Details
1. Hardware Components	<ul style="list-style-type: none"> ● A screen, keyboard, mouse/trackpad, computer that runs Windows, potentially with cheap scanner, further explained below ● Alternative: notebook/ tablet ● Dymo USB scale recommended as scale

2. Space and power requirements	<ul style="list-style-type: none"> • Notebook/tablet will need to be charged likely once per day, no local power requirements • Less than 1 sqft
3. Cost	<ul style="list-style-type: none"> • ~ 150 \$ one-time for notebook/ use existing tablets
4. Robustness	<ul style="list-style-type: none"> • Tablet covers built to survive repeated drops • Tablets & notebook would be robust enough
5. User interface	<ul style="list-style-type: none"> • Typical Laptop or tablet
6. Connectivity to the WMS	<ul style="list-style-type: none"> • Exists
7. Weight Database capabilities	<ul style="list-style-type: none"> • Currently no central weight database exists, we are using a local csv file updated from this Google Sheets document, which acts as our central weight database. However, an actual database could be implemented by the IT department and connected to similarly to the existing WMS connection
8. Connectivity to Scale	<ul style="list-style-type: none"> • Connection to Dymo scale exists, and is recommended for testing • Other scales likely possible, we could not set up and test a protocol
9. Tolerances and Range	<ul style="list-style-type: none"> • Depends on the scale • For the Dymo scale, a product weight of at least 4 g would be safest
10. Installation and Testing	<ul style="list-style-type: none"> • Limited ability to actually test onsite, aggravated by the current public health restrictions and resulting inability to enter the DC
11. Long term support	<ul style="list-style-type: none"> • Finding a commercial partner is an option but they would be unlikely to be interested in supporting our solution • Sager IT department does not have resources to

	<p>support this solution long-term</p> <ul style="list-style-type: none"> • We will provide detailed documentation, but this is not fool-proof • Main problematic scenarios: database structure or connection specs change; external software modules used become outdated (should last longer than 5 years) • Accountability (legal and otherwise) is an issue as we do not have the same level of authority that certified engineers do • If hardware fails, any 12 inch laptop with Windows/Linux will work as replacement. Setup instructions are short and provided by us
12. Customizability	<ul style="list-style-type: none"> • Further customization, e.g. realization of weight database possible, requires setting up such a database on Sager's end • Extended communication with WMS possible, e.g. to enter counts as complete
13. Summary of Value	<ul style="list-style-type: none"> • Could act as a "test" to see if they might like to invest resources into the continued development of a system similar to this • Act as a guide to what they might want to implement in the future with another commercial partner
14. Summary of Limits and Challenges	<ul style="list-style-type: none"> • Lack of Long-term support and accountability • Currently full weight database functionality is not implementable, because such a database does not exist. We are using a Google Sheets document, which acts as our central database, and can be updated. These updates will be reflected in the program, but only if the laptop/tablet has an Internet connection. • Limited testing capabilities -> no connection protocol to Sager's existing scales

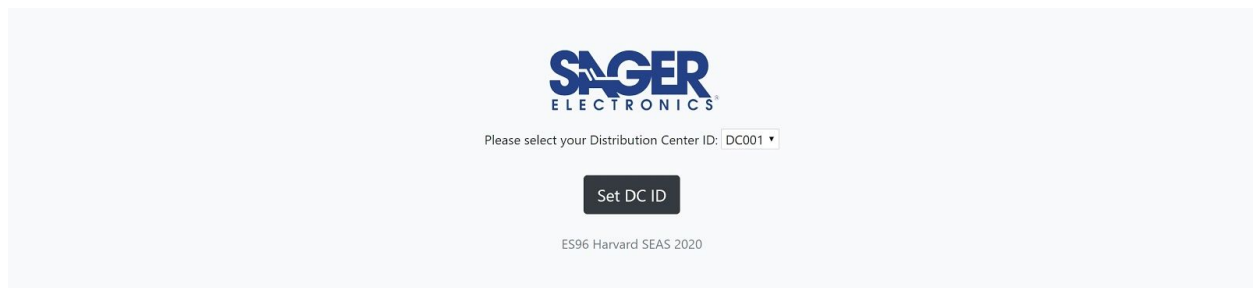
Now that we have seen a short summary of the capabilities of our solution we will next have a look at the user experience of our customized solution. The complete documentation of the code is found in Appendix A.

3. Custom Solution: User Experience

To start the application, a desktop icon named Sager_counting.bat is simply double clicked. This will pull up Chrome at the correct Login page. It may take a second for the server to start up, so refreshing the page a few times may be necessary.

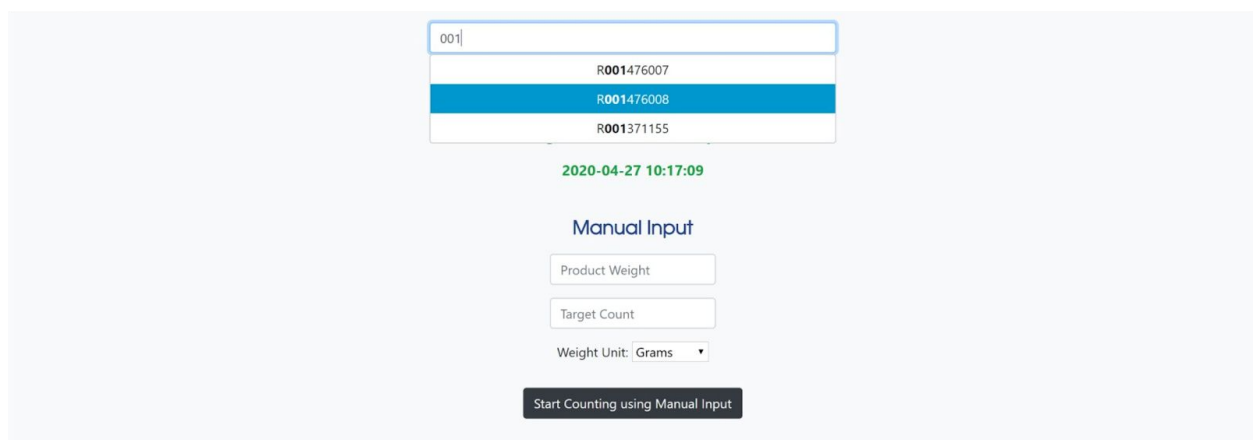
Login Page

On this page the user will enter their DC ID which will be used to retrieve the relevant order information. When the user enters a possible ID, they get to the setup count page.



Setup count page

Here, the user can either retrieve the current order they are picking for. First they need to enter the license plate number either by entering part of it and selecting the correct LP from a list, or by scanning the label of the part with an RF-scanner used as a keyboard input.



Then, they need to select the order they are picking for from a table and confirm the quantity.

Retrieve orders containing this license plate

Orders containing this license plate

Order ID	License plate	Order quantity	Submit?
25133471.0	R001476008	3	<div>Start Count</div>

The weight database was last updated at
2020-04-27 10:17:09

Manual Input

Retrieve orders containing this license plate

Orders containing this license plate

Order ID	License plate	Order quantity	Submit?
25133471.0	R001476008	3	<div>Start Count</div>

The weight database was last updated at
2020-04-27 10:17:09

Confirm that your target count is 3 and submit.
Otherwise please retrieve your license plate again
or select a different order from the table

Manual Input

The weight data will be taken from a local copy of this Google Sheets [document](#) as the weight database. The local copy will be updated using the Google Sheets whenever the page is loaded, but this requires an Internet connection. Alternatively, they can manually input the part weight, the unit of weight and the target quantity.

Retrieve orders containing this license plate

The weight database was last updated at
2020-04-27 10:17:09

Manual Input

Weight Unit:

Start Counting using Manual Input

If this site is filled out correctly, the user is redirected to the counting page.

Counting Page

On the counting page the user will be walked through the correct counting process. First, the user needs to confirm that the scale is turned on, empty (or only with a container) and tared. Next, the user can start the count and start adding items.

The screenshot shows the 'Counting Page' interface. On the left, under the heading 'Order Overview', there is a table with the following data:

License Plate Number	R001476008
Product Weight	0.212
Weight Unit	LB
Target Count	3
Target Weight	0.636

Below the table is a button labeled 'Cancel and Return to Barcode Input'. To the right of the table, the current count and weight are displayed:

Current Count:
0 parts

Current Weight:
0 LB

Below this, a red message states: 'You have counted too few items'. At the bottom right, there is a 'Stop Count' button.

The user will see messages indicating whether the counting process is completed or whether items need to be added or removed.

This screenshot shows the 'Counting Page' after adding items. The 'Order Overview' table remains the same as in the previous screenshot. The current count and weight are now:

Current Count:
3 parts

Current Weight:
0.6875 LB

A green message states: 'Count Complete!'. The 'Stop Count' button is still present at the bottom right.

The user can cancel the count at any time, but to complete the count, the count needs to be correct, and the user is asked to take the items off the scale and repackage them if necessary. Then the user can submit the count as completed.

This screenshot shows the 'Counting Page' after completing the count. The 'Order Overview' table has been updated with the following data:

License Plate Number	R001476007
Product Weight	0.212
Weight Unit	LB
Target Count	1
Target Weight	0.212

The current count and weight are:

Current Count:
1 part

Current Weight:
0.21 LB

A green message states: 'Count Complete!'. Below the table, there is a checkbox with the text 'Please take the items from the scale, and repackage them if necessary', which is currently checked. At the bottom right, there is a 'Continue count' button. Below the checkbox, there is a 'Complete and Setup New Count' button.

Currently this submission does not communicate with the WMS, but this may be implemented. The user gets redirected to the Setup Count page and can continue to set up a new count.

On the setup count page and the count page the user can also go back to change the DC ID. To stop the program, please close the command window which is open in the background.

The technical documentation of the software can be found in Appendix A.

Now that both the commercial options and our custom solution have been presented, we present a table that summarises the key values and limitations of each option.

Summary of Options:

	Basic Counting Scale	Our Custom Solution	Commercial, Advanced Option
Main Features & Advantages	<p>Scale upgrade</p> <p>Basic counting application to somewhat automate counting process</p> <p>Reliable commercial partners for calibration</p>	<p>A system with a simple user interface that retrieves order information from the WMS and product weight data from an online, editable file, then displays the scale readout with intuitive prompts</p> <p>Can be implemented on a small scale for testing purposes</p> <p>Demonstrates more advanced levels of functionality to inform what level of customizability may be desired from a commercial partner</p>	<p>Connects scale to a database containing product weights</p> <p>Able to integrate completely with the WMS to almost completely automate the counting process</p> <p>Allows for electronic double-checking and backtracking of errors</p> <p>Updated product weights in database visible on picking end</p>
Limitations	<p>No (or limited) central weight database so still need to initially determine piece weight</p> <p>No integration with the WMS, so no storage of completed count weight data, so still room for human error</p>	<p>No support or testing from us. If it is taken further it will need to be handled by someone in Sager's IT team</p> <p>Weight database functionality is implemented only to a limited degree, using a Google Sheets doc</p>	<p>Will require extensive communication with a commercial partner</p> <p>More expensive</p> <p>Unclear if the added functionality of e.g. WMS communication will impact effectivity at reducing errors</p> <p>Will require collecting more product weights to be useful</p>
Hardware Specifications	Box-like scale with	Laptop or tablet connected to a	Same as basic option:

	display designed to fit on bench and withstand warehouse conditions	scale	box-like scale with display designed to fit on bench and withstand warehouse conditions
Cost	\$1500-2000 per scale	\$150 one-time for a notebook, or can use existing Windows tablets; a barcode scanner (~150\$) might be useful but may exist already, Dymo USB scale costs about 30\$	~\$2000 for scale + \$1000-2000 for customized software

Based on our research and the solution we came up with it seems best to use our customized solution to collect experience that would be useful for a cooperation with a commercial partner like Avery or Mettler Toledo. Since we could not complete a protocol to one of Sager's actual scales, we would recommend they use the same scale as we used, a Dymo USB scale (as described in the hardware part of our code documentation). This would hopefully allow Sager to gain enough experiences with an advanced counting solution that they could have a strong cooperation with a commercial partner and minimize order inaccuracies in this part of the picking process. Such a commercial cooperation might be significantly more expensive than our solution, however, it is the only feasible option to implement a consistently powerful solution to the counting scale problem. We will not be able to offer long-term (10 years) support for our solution, and the limited capacities with Sager's IT department do not allow for extensive internal support of the counting scales. We hope Sager will be able to implement a counting system that is capable of improving order accuracy based on the work outlined here.

B. LED Lighting System

1. The Idea

1.1. Goals of the system

1.1.1. Overarching goal

Develop and implement a system, compatible with the current standard operating procedures, that reduces human errors that contribute to wrong part errors and increases efficiency.

1.1.2. Reduction of wrong part errors

A major contributor to wrong part errors at Sager is the misplacement of boxes and parts inside the warehouse during the putaway process, or picking up parts from the wrong location during the picking process. The standard operating procedures dictate that an employee must scan both the item and the shelf that it is being placed on during putaway to ensure that it is not misplaced. However, one common practice that employees do is that they scan a product and the shelf that the product should be placed on, but when they actually put the product on the shelf, they end up placing it at the wrong location. This small misstep could be the cause for a wrong part error. Our solution aims to mitigate this problem by using LEDs to give a visual indicator to employees that points to the correct putaway/picking location to reduce the misplacement of parts during putaway and the picking of wrong parts to ultimately reduce wrong part errors.

1.1.3. Increase efficiency

Our system which was designed to tackle wrong part errors has a built-in advantage; it helps increase efficiency since employees will spend less time looking for the correct locations. While this was not the problem we set out to solve, it is a problem that is addressed by the LED pre-queuing system.

1.2. How it works

1.2.1. Employee Perspective

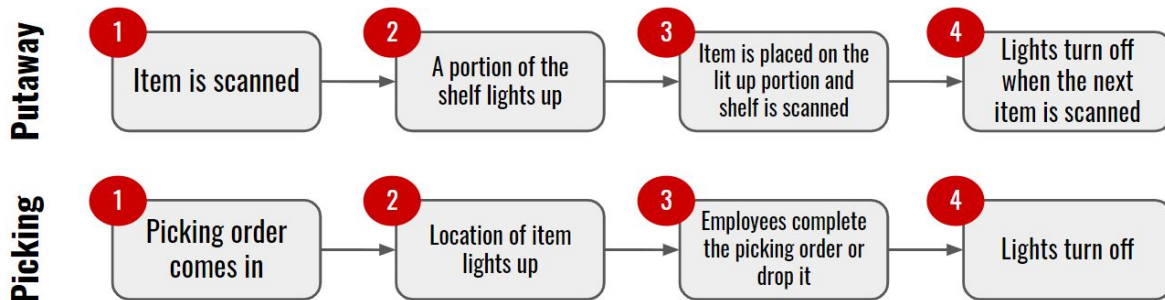


Figure 7: Process flow for putaway and picking for workers in the distribution center

This is what the system will look like for employees working in the warehouse. It is important to note that the procedure that is currently followed in the warehouse will not be altered after the installation of the system.

Process flow for Putaway

1. The employee scans the item that needs to be put away
2. The LED at the designated put away location for the item will light up
3. The employee scans both the location and the item, and places the item on the lit up location

4. The LED at that location turns off when the next item is scanned or when the employee signs off

Process for picking

1. The employee scans the empty tote waiting to be filled with items
2. The LED at the location of the item that needs to be picked lights up
3. The LED turns off when the employee indicates one of the following
 - a. The picking order is complete
 - b. The picking order is short
 - c. The end of their shift

1.2.2. Flow of Information

The main source of information is the Warehouse Management System (WMS). The software developed by our team will extract information about the putaway/picking location and the employee performing the task, then it will send this information to the firmware. The firmware will use this information to turn LEDs on/off with the correct color (each color corresponds to a different employee). The trigger for the start of each process is a scan performed using the RF scanner.

2. System Components

2.1. Hardware

The hardware component consists of the following items

Microcontroller

- Receives the location of the LEDs and turns them on/off

SD card (SDHC)

- Must be an SD/SDHC card, Arduino SD card library cannot work with any other type (i.e. SDXC)
- Stores two tables
 - One that has all the LED locations in the format received from the WMS, and their corresponding LED index as it is stored in the microcontroller
 - One that has the RFIDs of the employees, and the LED color corresponding to each RFID

LEDs

- One LED at each location in the warehouse that lights up when an item is to be placed at the location, or an item is to be picked from that location

Computer

- Used to run the software needed for our system

Wires

- Connect all the components together

110v/5v step down transformer

- Used to step down the voltage of the power source in the warehouse to the operating voltage of the microcontrollers

2.2. Software

The main function of the software is to extract information from the WMS, process the information, then send it to the correct microcontroller in the warehouse. This is done through the following functions

Main.py

This is our main function that will run continuously which will call all the following functions and run any supporting code needed for the following functions. It will also connect to and communicate with the microcontrollers.

findNewTransactions.py

This function detects when a new transaction is inserted into the WMS database, and it outputs the information stored in that transaction.

OnOrOff.py

This function takes in the location information output from *findNewTransactions.py* and determines if the LED needs to be turned on or off at that location, and if any other locations need to be turned off in addition to the location received.

getIP.py

This function gets the IP address of the microcontroller corresponding to a specific location. Each microcontroller will control about 40 locations, so that software needs to know which microcontroller is responsible for the location we want to turn on/off.

The full code is provided separately from this report.

2.3. Firmware

The main function of the firmware is to receive information from the software in order to trigger LEDs to turn on or off. This is done with an embedded SD card that is updated through the firmware that translates pickerID inputs into the individual colors pickers have in each section. It receives an on or off command from the software so that it can handle multiple LEDs in the row being on.

The firmware being run on an arduino MKR WIFI 1010 microcontroller, is designed to receive signals with WiFi communication from the WMS. The firmware receives all signals from the software unit as a datagram packet, designed so as not to account for occasional loss of connection, the datagram packet is designed with, addressable location (string), RFID (string), and an on or off command (int).

Upon receiving the string for addressable location, as a secondary check to make sure the right signal was sent to the microcontroller, it checks the received address against the SD card file LED.txt to see if that address is actually in the warehouse section. If the location is contained, it will determine the pin associated with the certain shelves of each unit, and convert the shelf address to the correct LED index on that shelf. This association of pins to shelves of the unit is used as a power and wire saving measure.

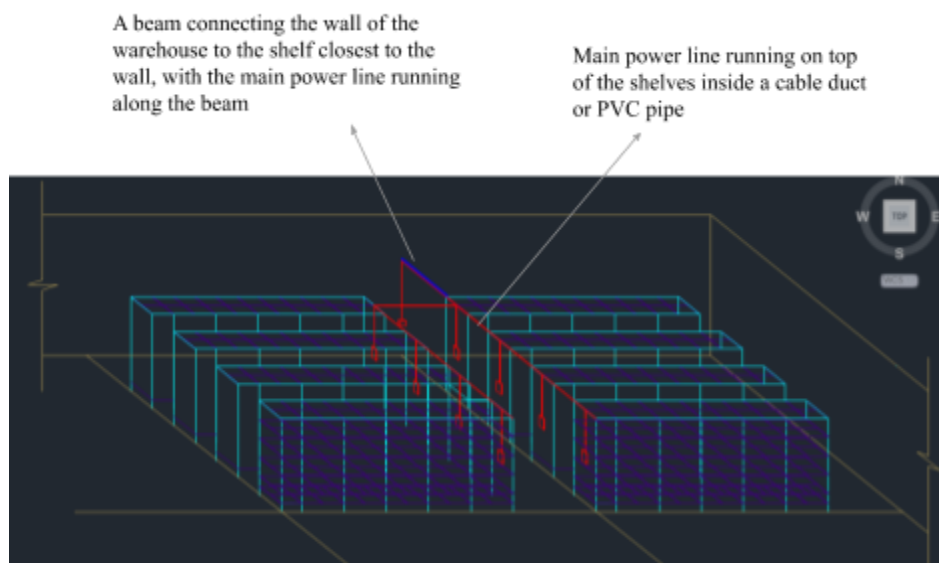
When reading the RFID, it checks it against current RFID in the SD card file pick.txt, this is determining if the RFID is one of the last eight in the region (amount of available colors). If the RFID is not in the pick.txt file it adds it to the file and removes the ID of the least recent picker, upon doing so it assigns the color of the removed picker to the picker that entered the area. The full code is provided separately from this report.

3. Wide-scale Implementation in the Sager Distribution Center

3.1. Integration with the WMS

Our team reached out to the IT department at Sager Electronics to investigate the possibility of connecting our system to the WMS. They supplied us with VPN information to connect but they did not supply us with the necessary commands to extract information from their database. So at this moment, it is not possible for us to integrate into their WMS and run our system on it. However, their database is an SQL database, so our team set up a test SQL database that resembles their WMS to test our code. The transition from the test SQL database to full integration with their WMS will only require simple changes to commands in the software since both databases are in SQL and the command structure is the same.

3.2. Power and wiring



To power the microcontrollers which will in turn power the LEDs, we will need to install a main distribution board to draw power from. Two main power lines will be drawn from that distribution board; one will power the bulk area of the warehouse and one will power the smaller shelf area. The main power lines will be run on top of the shelving units in cable ducts or PVC pipes.

As for wiring on individual shelving units, the microcontroller will be mounted on one side of the shelving units and parallel wires will run to each shelf. The LEDs on one shelf will all be connected in series, while different shelves will be connected in parallel; this daisy chain method reduces the need for LED drivers and connecting the shelves in parallel ensures that if an LED breaks down, only one shelf will be affected rather than the entire shelving unit.

3.3. Attachment of components to shelving units

Microcontrollers

The microcontroller will be placed in a housing unit that is attached to the shelving unit. This way if the microcontroller needs to be replaced, it can just be removed from the housing and the new one can be installed.

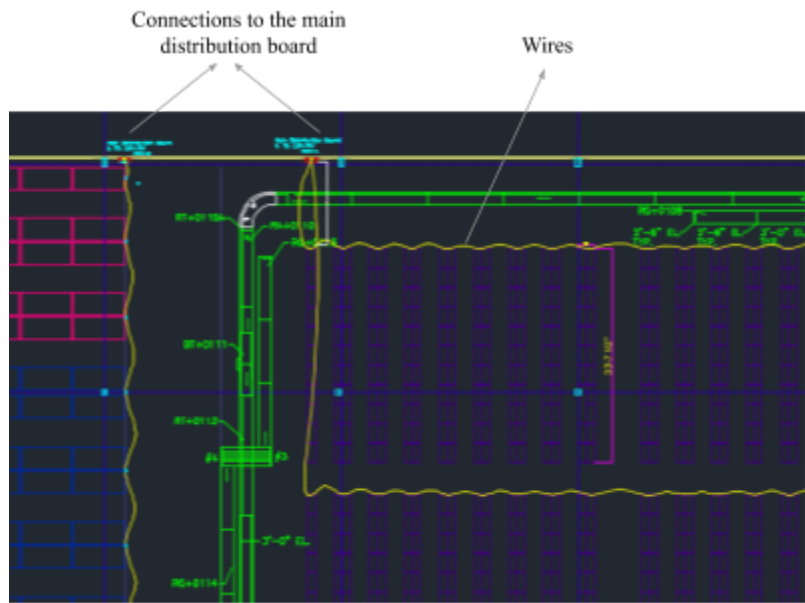


Figure 9: top view of how the wires will be running from the wall on top of the shelves

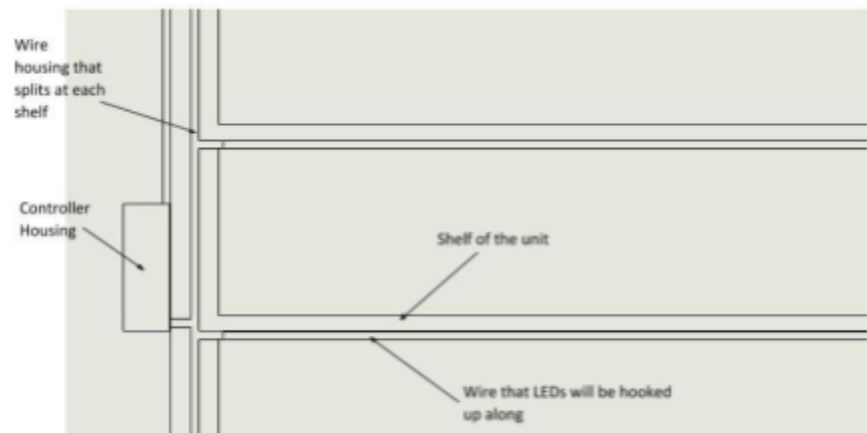


Figure 10: Wiring diagram for a single shelving unit

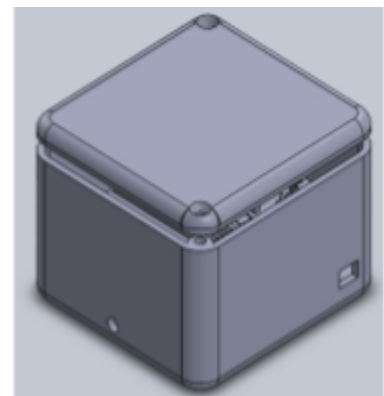


Figure 11: microcontroller housing

LEDs

Terminal blocks will be used to avoid the need for soldering the LEDs onto the shelves. The terminal blocks used will have three inputs and three outputs; the wires going in will power the LED, and the wires coming out will power the next LED in the daisy chain.

LEDs can simply be screwed on or off the terminal block, which makes it easy to replace the LEDs when they need replacement.



Figure 12: picture of the LEDs that will be used

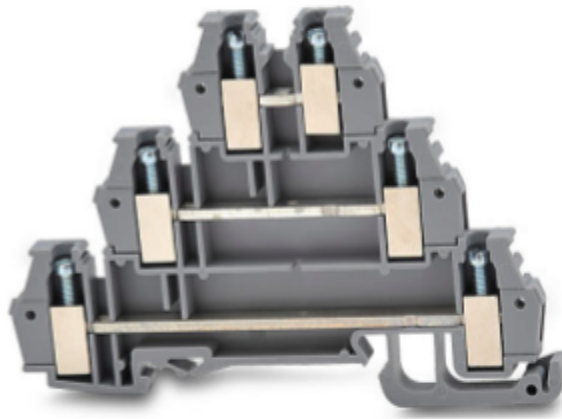


Figure 13: 6 pin terminal block, with 3 inputs and 3 outputs

3.4. Cost Estimates

One of the main concerns for the customer was the price of the system. Therefore, our team came up with a comprehensive budget that shows price estimates for two scenarios. Estimate 1 shows the budget for partial implementation in one area of the warehouse to test the system. Estimate 2 shows the budget for the system if it is implemented across the entire warehouse.

Estimate 1: 1000 locations (One Area of the Warehouse)

Item	Cost/unit	Quantity	Cost
LEDs	\$1.60	1000 (or number of locations in the warehouse)	\$1600

WIFI enabled microcontroller	\$25 - \$35	10 (or number of distinct shelving units being outfitted)	\$300
Memory shield	\$23	10	\$230
SD card	\$7	10	\$70
Cable and Power Brick*	\$15	10	\$150
Wiring on Shelves	\$0.1/foot	3000 ft (estimated length of wiring required for 1000 locations)	\$300
Computer	\$500	1	\$500
Implementation costs (see last page for more details)	\$0.70 / foot + \$0.70 / location	3000 feet + 1000 locations	\$2800
Total			\$5950

*A power drop is unlikely to be necessary for testing a small number of locations, and we are assuming that the microcontrollers can be connected to power outlets.

Running power cost estimate: \$36 / year

Estimate 2: 10000 locations (Full Warehouse)

Item	Cost/unit	Quantity	Cost
LEDs	\$1.60	10000 (or number of locations in the warehouse)	\$16000
WIFI enabled microcontroller	\$25 - \$35	100 (or number of distinct shelving units in the warehouse*)	\$3000
Memory shield	\$23	100*	\$2300
SD card	\$7	100*	\$700

Power Drop and Wiring for Power Distribution	\$2000 + \$2.5 / foot (including labor)	1 + 400 feet	\$3000
Wiring on Shelves	\$0.10/foot	30000ft (estimated length of wiring required for 10000 locations)	\$3000
Computer / Raspberry Pi	\$500	1	\$500
Implementation costs for wiring on shelves (see last page for more details)	\$0.70/ foot + \$0.70 / location	30000ft + 10000 locations	\$28000
Total			\$56500

*This number accounts for the fact that large shelving units may require 2 Microcontrollers (and corresponding memory shield and sd card).

Running power cost estimate: \$360 / year

Implementation Notes:

- It takes an hour to wire ~100 ft of wiring, and labor costs per hour for wiring are 40-100\$ (assume \$70 / hour), giving an estimated labor cost of 0.70\$ / foot.
- We estimate the same labor costs for wiring each LED (\$0.70 per location).

4. Design process

4.1. Initial idea

The initial idea of our system was a much simpler one that did not involve integration with the WMS and did not apply to the picking process. These were the initial features of the system

- The employee would scan the location on the shelf
- The light turns on after the employee scans the shelf; the light would have been triggered manually not through a command from the WMS
- The light turns off after a set time

4.2. Switching to a pre-queuing system

The initial idea had some flaws that we needed to address:

- The idea of manually triggering the LED was not something that the client wanted. It would have added steps to the already existing procedures, making it more likely that the employee will just skip the step, not do it if they thought it was unnecessary, or simply

forget to do it since the process can be done without it. Therefore, the system was still highly prone to human error.

- Turning the lights off after a set time was not ideal because warehouse conditions cannot be predicted which makes it difficult to come up with ideal wait time before shutting the LEDs off.

Because of these concerns we decided to switch to the automated pre-queuing system that we currently have. The pre-queuing system will not add any steps to the already existing procedure, will be triggered and run automatically without employee input, and will not depend on a timer. Therefore, the team decided that the pre-queuing system will reduce human error and increase efficiency more than the manual system.

4.3. Expansion to include picking

Picking and putaway happen at the same locations. Therefore, since we were already installing the hardware at all of these locations, the same hardware can be used for both if some edits were made to the software and firmware. So we decided to expand the software to include picking to make use of the hardware and make the cost more attractive to the client since it would tackle errors in two processes instead of just one.

4.4. Technical considerations when choosing system components

WIFI enabled microcontrollers

We chose WIFI enabled microcontrollers to reduce the amount of wiring needs in the warehouse area and to simplify the wiring scheme of the entire warehouse

Single LEDs at each location

We chose single LEDs at each location over other options such as LED strips or lasers because of the following advantages of single LEDs and the following disadvantages of the other systems

Single LEDs advantages

- Cheapest option; both the startup cost and the running costs were lower for this option
- If one LED breaks, only that LED needs to be replaced
- Lowest power consumption

LED strip disadvantages

- More expensive; both the startup cost and the running costs would have been more expensive
- If one LED breaks in the strip, the entire strip would need to be replaced

- Higher power consumption

Laser disadvantages

- More need for consistent calibration of the lighting system (greater cost to maintain)
- Power considerations will be more due to increased power draw from the system
- Potential of employees standing in front of location and blocking the light during put away (Thus not reducing wrong part errors)

C. Employee Engagement Program

1. Introduction and Intentions

The Employee Engagement program is focused on tackling the overall issue of order inaccuracy from a more human-centric, employee-focused perspective. The program is intended to boost manager to distribution center employee relations by offering more chances for feedback to be shared between the two parties. This increased feedback is intended to serve three main purposes: to help Sager identify when and where to retrain, as well as what parts of the procedure are confusing, and how to best support its employees. In addition to optimising procedures, which will ideally ultimately improve order accuracy, and potentially even increase efficiency, this program has the potential to increase job satisfaction, motivation, and loyalty.

2. Goals

The goal of the Employee Engagement program takes into account two aspects - targets and requirements. Our target goals are what we hope to achieve by implementing our engagement program, while requirements are any Sager-specific prerequisites we must consider for any changes we wish to implement.

Targets:

We built our solution targets to achieve four objectives: first, to encourage communication about procedures and performance; second, to reduce turnover; third, to gauge employee performance; and fourth, to provide suggestions to increase engagement.

Requirements:

Any change to be implemented with the goal of achieving one of the four targets listed above must also meet a set of basic requirements as defined by Sager. First, any changes to current procedures must not significantly increase the workload of any employees; second, the changes must be easy to integrate; and third, they must be within Sager's budget.

3. Features of the Program

3.1. Electronic Questionnaire to Measure Employee Performance

Employees will be required to fill out a short questionnaire three to four times per quarter. The survey contains five questions. The first three questions are multiple choice, while the last two require a yes/no answer. The questions are:

- i. I understand the processes, methods, systems, and procedures
All Most Some Few None
- ii. I comply with company policies, regulations, and codes of conduct
All Most Some Few None
- iii. I focus more on:
Accuracy Efficiency
- iv. I accept and act on feedback given to me
Yes No
- ii. I seek help from my supervisor when needed.
Yes No

There is a section for any additional comments at the bottom. All questionnaires will be completed and turned in anonymously. Through these questions, we hope to give employees an outlet to voice concerns, frustrations, or comments in a risk-free environment. A complete version of the questionnaire is attached in Appendix C.

3.2. Commercial Platform Integration

The implementation of the questionnaire along with the data analytics will be done via a commercial solution. The two companies we recommend are Impraise and Engagedly, who have both developed online platforms constructed to handle all of a companies' HR needs. The two platforms are similar in many ways, and a comparison of the two companies can be seen below.

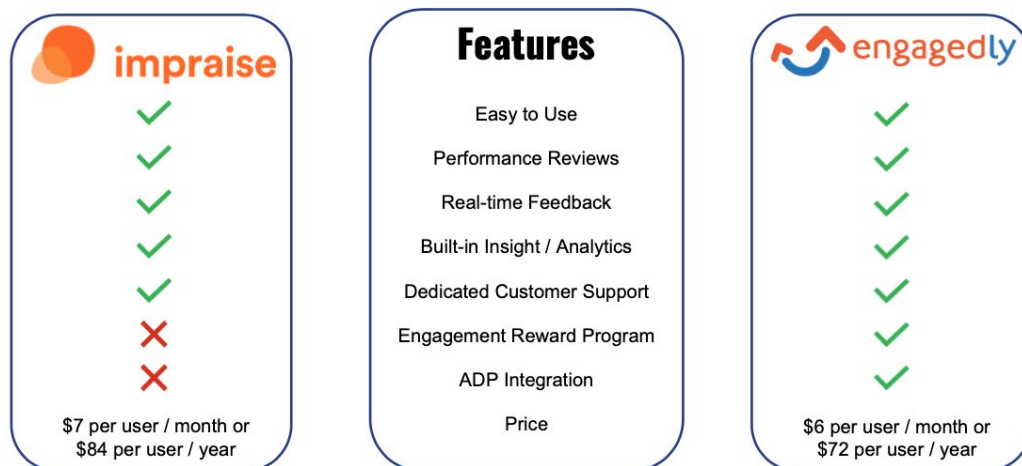


Figure 14: Commercial Platform Features

As shown in the chart, the two solutions offer many of the same features, although Engagedly offers slightly more features that Sager would benefit from (an engagement reward program that gamifies the system, and ADP integration) at a lower price point. Moreover, since Sager has been in contact with both companies, we believe they are in a better position to choose the company they wish to proceed with.

3.3. Employee Engagement Suggestions

Based on the material we have researched on this topic, we are also recommending the implementation of a few suggestions to further increase the effectiveness of the Employee Engagement Program. These suggestions include incorporating a group meeting system, embedding additional employee appreciation into current Sager work culture, and implementing a robust retraining system.

3.3.1. Group Meetings

We recommend adding weekly or bi-weekly group meetings. These informal, discussion-based meetings would include five or six production employees, who ideally would lead the conversations, along with one supervisor. The goal of these meetings is to increase communication among employees and supervisors and for supervisors to hear feedback from employees that isn't covered by the survey.

3.3.2. Employee Appreciation Events

We also recommend adding small social events targeting employee appreciation on a monthly or bi-monthly basis. These events could be used as a platform to acknowledge employee tenure, showcase company highlights, and more. The goal of this is to increase camaraderie among employees

3.3.3. Retraining System

Finally, we recommend developing a system of retraining to be conducted whenever Sager feels that it is necessary. In addition to refreshing employees' memories in completing different jobs that they may have experienced skill atrophy in, retrainings would also be helpful in guiding the implementation of any procedure modifications based on information gathered from the data analysis provided by the commercial solution.

We hope that in providing Sager with this list of employee-focused strategies, in addition to the more technical scale and LED strategies, it will expand their arsenal of tools for tackling their order accuracy problem.

VII. Conclusions

The problem that our team was tasked with solving was the order accuracy issue in Sager Electronics's distribution center in Middleborough, MA. Order inaccuracies in the distribution process negatively impact their customer relationships, profits, and the safety of the end users. Our team created detailed stakeholder and system maps to identify areas in the distribution process where errors might occur. The team came up with a problem statement and used it to start visualizing potential solutions. After generating a comprehensive list of potential solutions, the team produced a set of solution criteria and used them to evaluate the list of solutions. This evaluation process helped narrow down the solution space to three ideas which we then pursued.

The first solution aimed to address quantity errors that occurred when scales were used during the counting part of the picking process by automating the counting procedure. We created a web application that can connect to Sager's Warehouse Management System and a database of product weights in Google Sheets. It automates counting using the order quantities from the Warehouse Management System and the weight information from the product weight database. The second solution focused on tackling wrong part errors that were the result of inventory accuracy issues. We designed an LED system that could be connected to Sager's Warehouse Management System which would turn on location-specific LEDs in the inventory area associated with an employee's picking or putaway task. The third solution focused on the order accuracy issue from an employee-focused viewpoint. This solution is an employee engagement program which allows for more feedback to be shared between the employees and managers in order to identify retraining needs and optimise procedures.

Moving forward, we identified potential paths for each solution. For the scales solution, this program could be implemented on a small scale in the distribution center to show the benefits of this type of system, and this trial would help inform future discussions with a commercial partner. For the LED solution, our system could also be implemented on a small scale in the warehouse at a reduced cost, and this would help with future discussions with a company that offers a "pick-to-light" solution with long-term support. A possible extension to our employee engagement solution could be a web-based commercial platform which would host the questionnaire and encourage frequent feedback and communications.

We hope that both the solutions our team has created and the research we have done into existing commercial options can help inform Sager Electronics of the best next steps to take to implement changes to current procedures in their distribution center that will ultimately address their order inaccuracy issue.

VIII. Appendix

A. Programmable Scales Documentation

1. Introduction

In this document we are documenting the software we have created to automate counting within Sager's distribution center and give support instructions to set up and troubleshoot our software. Some parts of the document will require basic programming knowledge, but we have included links to online resources that explain the external libraries used. This document can both act to support the testing of our software as well as set up Sager or an external company to provide ongoing support and improvements.

2. Setup for Windows

Hardware options

Since we were unable to implement a connection to one of Sager's scales, we recommend using the same scale as we used. This would be the Dymo USB scale, and it can be purchased e.g. [here](#). Additionally, a tablet or notebook with Wifi connectivity and a USB port would be necessary. Shannon Freise told us Sager has Windows tablets on-site, these should work well. Additionally, it may be helpful to use simple barcode scanners as keyboard inputs to the tablets to minimize the time spent entering license plate numbers. These might already exist at Sager. Potential solutions could be found [here](#).

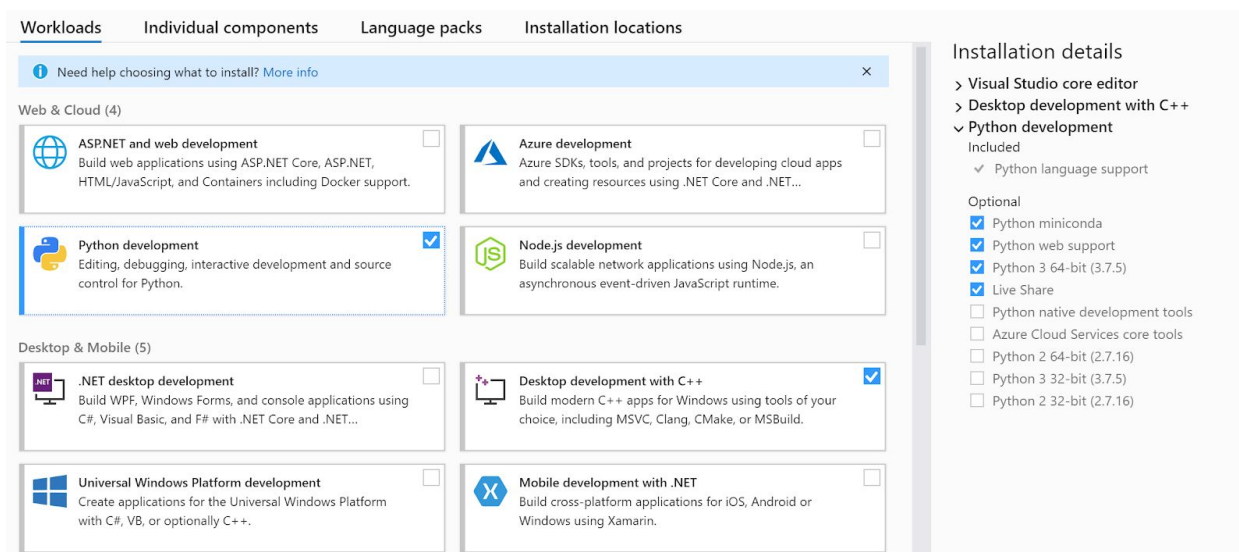
Setup

First, save the folder "application" somewhere on your system, and note its location.

Then, make sure you have Python 3 installed, by hitting the Windows button and searching for "Python 3". If it is not installed, install it from the Microsoft store. Any version of Python 3 will do. Then right-click Sager_counting and select edit. In the line containing "set location_submit_to_sager= ..." replace the directory here with the where you stored the folder containing the batch scripts and the application folder (the relevant path in the submission zip file is: "/submit_to_sager/programmable_scales_solution/"). You can also find the address by opening the Windows File Explorer, navigating to the location of the application folder and right-clicking the address line at the top of the window. Select "Copy address as text" and paste it in Sager_counting. Now save the file and close it.

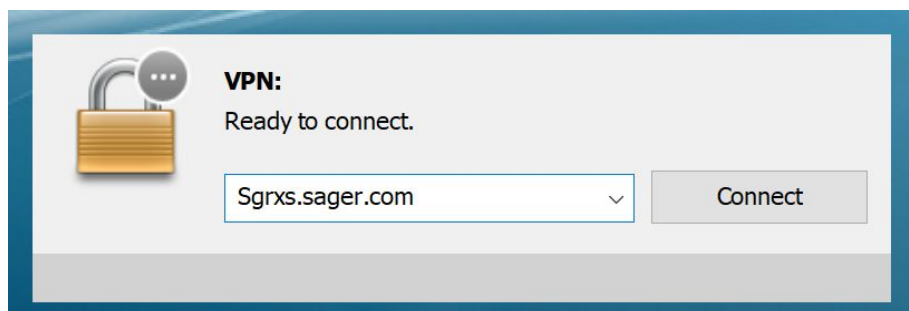
For the Dymo scale you will also need to set up a USB device filter. Download Libusb-Win32 e.g. from [here](#), and run it. Follow the installation instructions, and launch the program on the last step of the setup. Select "Install a device filter". Connect your scale and turn it on. Under Hardware ID, look for an entry where vid:0922 and pid:8003, select the entry and click Install. You can then close the window.

To set up the connection to the WMS, first download Microsoft Visual Studio Community Edition from [here](#). Click the downloaded file to install it. The installer will first download more files. Next you can select which packages to install. Select Python Development and C++ Desktop Development like shown below. The default packages checked will suffice. When the download is complete, restart your computer.



Next, download the Java development installer [here](#), selecting the Windows x64 Installer. When the download is complete, click the downloaded file and install. Note the file path you used to install. Then click the Windows button and search for “Edit the System Environment Variables” and click it. Under the Advanced Tab click Environment Variables at the bottom. In the lower part of the new window under System variables, click “New”. Enter “JAVA_HOME” under variable name and under variable value, enter the Java installation file path. If you used the default installation path into Program Files, use “C:\Progra~1\Java\jdk14.0.1”. Make sure this file path is correct using the file explorer.

Then store the file ojdbc10.jar in the programmable_scales_solution folder in the main windows directory C:\. Next, you need to actually connect to the network the database is connected to. On-site, connecting to the Sager Wifi should work. Remotely, you will need to use Cisco AnyConnect Secure Mobility Client. If you are at Harvard you can download it for the [Harvard VPN](#). Once it is installed, enter Sgrxs.sager.com as the server and hit connect. Select HarvardES96 as group, and then use 2020ES96User as the user name and Gr0upES96#19L as the password. Click OK to log in. Note that you will be unable to access the regular internet while you are connected to the Sager VPN.



Now, the setup is almost complete. Run `Sager_counting_setup` by double clicking it. When prompted, press any key to close the command window. When you double-click `Sager_counting` now it will open up Chrome (please ensure it's installed) at the login page. To terminate the script just close the command window.

3. Rough Structure Webapp

The script `application.py` is the backbone for the website and uses [flask](#). It contains the Python backend for the pages and sets up the html sites. It also calls various outside functions that connect to the databases used and to the scale connected.

Setup_DC_ID page

The `setup_dc_id` page is located at `127.0.0.1:5000`. This can be changed in `application.py`. The `setup_dc_id` page contains a Login, where the employee would enter the DC ID. The form is created using `wtforms`, which allows for easier form handling. There is documentation on `wtforms` [here](#). Usage is straightforward and further explained later.

Count Setup

When the employee correctly enters their DC ID, they are redirected to the site under `/setup_count`. On this site the user can either retrieve the current order they are picking for by entering the license plate number, or they can manually enter the product weight, target count and unit of the product weight. If they enter a license plate number, the Javascript on this page will retrieve all orders containing this license plate from the Python backend and the WMS and display the possible orders in a list. The Python backend will use a local copy of this Google Sheets [document](#) as the weight database. The local copy will be updated using the Google Sheets whenever the page is loaded, but this requires an Internet connection. If the user selects one of the orders they will then need to check a checkbox to confirm the target quantity is correct. The manual order form uses `wtforms` to ensure correct usage and completion. The user can also change the DC ID, by clicking on a button to the top left of the page.

Count Site

When the user has either retrieved an order or entered the details manually, they can start the counting process. On the left part of this site, relevant information is displayed to the user, including various identifiers like the license plate number and information relevant to the counting process. On the right side of the screen, the user can confirm that the scale is connected and turned on with no products on the scale. When this is confirmed, the user can start the count. This behavior is powered using Javascript and further explained later. While the exact behavior is detailed in the comments of `static/scripts.js` the main item of note in this script is `update_displays()`, which uses [Jquery](#) to retrieve the current reading from the Python backend. It

queries a website /check_weight. When this site is queried, the Python backend will retrieve a reading from the scale. The scale readout functions are contained in readScale.py, explained under the section Helper functions. It will then return this stable value to the Python script and the Javascript which will adapt the displays on the website to reflect the new reading.

When the count is stopped, the current reading process is finished. If the count is complete, the user is asked to take the items off the scale and repackage them if necessary. If the count is incomplete, the user is asked to continue counting, or cancel the count. It is not possible to submit the count, if it is incomplete.

When the user submits the count or cancels the counting process, they are redirected to the /setup_count page, where they can continue with the next count.

The user can also change the DC ID, by clicking on a button to the top left of the page.

4. Detailed Explanation Application.py

As explained, this script sets up the website functionalities using flask. It uses various modules supplied by flask, and also imports all helper functions written by us.

It then configures the website. Essentially all commands here are standard procedure, e.g. found [here](#). Items of note include the configuration of a secret_key, which will be required to use wtforms to improve the form handling of the various sites. Additionally, an IP address is configured for the app to run on, named host_address. 127.0.0.1 is the default location of the localhost, so it is recommended this address is used. If you decide to change the host address, please adapt Sager_counting.bat to reflect the changed host address.

After this, global variables are created and initialized. Five of the variables are related to the counting order, denoting the task ID, the product weight, the weight unit, the target count and the current license plate. A variable specifying the DC ID is created, as well. Additionally, a variable called all_current_orders_at_location is initialized, which will contain all relevant orders for the employee to choose from.

Next, classes for the wtforms are created. Info on wtforms can be found [here](#). First, the style of each form field is modified, to look better. Then follows the actual creation of the form fields. WTForms allows us to specify the type of input expected, e.g. an integer for IntegerField, include a label, include additional validators (like a range for the integer) and a customized error message if the form entry is invalid. For each form we also need to include a submit field. WTForms also requires specific code on the html side, which is explained later. The forms include the login form on the login site as well as the manual order form.

Here you can also add additional DC IDs. If the `setup_dc_id` page does not contain all DC IDs, these can be added when the `dc_id_entry` choices are created in the `select_dc_id` class. Just add another entry, the same way as the previous were included.

Additionally, the variable `within_target_count` is set here. This determines how close the actual count has to be to the target count, to be considered correct. This accounts for slight inaccuracies of the weights. The default value is set to `.2`.

Next, the `setup_dc_id` page is created, the relevant function is `setup_dc_id()`. Both GET requests and POST requests are allowed, the page can be simply loaded, and a form can be submitted on it. Three different scenarios can lead to this function being called.

If the form on the site was correctly filled out, the `dc_form` will validate, and the ID entered will be saved in the appropriate global variable. Next, the script will GET the `setup_count` page.

If the form was posted but did not validate, which seems almost impossible, the `setup_dc_id` page will be rendered again, `first_load=False` indicates to the html to show relevant wtform error messages.

If the `setup_dc_id` page was retrieved using a GET request (like after opening the browser for the first time), the DC ID is reset to zero, and the `setup_dc_id.html` is rendered. Any error messages are suppressed in this scenario.

Next, the `setup_count` page is created. This site again uses GET and POST methods. It checks that the user is logged in by checking whether the `employee_id` is nonzero. Four different scenarios can lead to this function being called.

If any of the forms was posted on the site, the request method will be POST.

If the manual order form was completed correctly, the entered data is assigned to the global variables describing the current order, like product weight etc. The script then GETS the count page.

If the user selected their order from a list after entering the license plate number of their product and submitted it, we can retrieve the `task_id` from the request, which will be non-zero in this case. This value was created when Javascript submitted the retrieval form. Using this task ID the selected order is retrieved from `all_orders_at_current_location`, a list with infos on all tasks at the current location created during the GET request for the `setup_count` page. Here, no errors need to be caught, since it will be impossible to reach this function, if any errors occurred when `all_orders_at_current_location` was created. Next, the info corresponding to the current order is assigned to the relevant global variables. If neither form was filled out correctly, the `setup_count` page is rendered again showing any form errors.

If the request method is GET, the user was just redirected to the page e.g. after completing a count or logging in. First, info on all current orders including task id, license plate, quantity requested, product weight, weight unit will be retrieved using `retrieve_all_tasks` in `retrieve_order.py`, which is documented below. Additionally, a function to attempt to update the weight database will be called, and the timestamp of the last update will be saved to pass it on to the html. If any errors occur within that function, it will print error messages to the command line, and indicate error messages in the return values. These error messages and other possible errors will then be passed to the html and include:

1. No connection could be established to the databases
(`database_connection_unavailable`)
2. There was some error during the retrieval process (`retrieval_error`)
3. No orders are available for this location, or none have weight data associated
(`no_orders`)
4. There was some other unknown error (`all_order_gen_error`)

Additionally, the timestamp of the last weight database update will be passed to the html.

Then, the `setup_count.html` file is rendered, with any wtform-related error messages suppressed, by setting `first_load = True`.

Next, the count page is created. Again, GET and POST are allowed, and the user needs to be logged in. If the POST method is used, this will redirect the user to the `setup_count` page.

If the site is accessed via GET, the `count.html` file is rendered and passed relevant information to be displayed on the site, like target count or product weight.

Next, the `check_weight` page is created. This page is somewhat unusual. It does not return an html file or a redirect but rather returns data in the JSON format. This data consists of a list with current count (index 0), the current weight (index 1), an indicator whether the count is complete (index 2) and the weight unit (index 3). These values are determined by retrieving a stable reading from the scale using `accurate_reading()`, a function in `readScale.py`. The corresponding count is calculated using the product weight. If the count is within `within_target_count` (default 0.2) of the target count, it will set the indicator at index 2 to 0. Otherwise it will set it to -1 (count short) or 1 (count over). The list is converted to JSON format using `jsonify`, a flask function, and returned. If `readScale.py` indicates that no connection could be established, 'Not connected' is returned in JSON format.

A site similar to `/check_weight` is `/get_ids`. This function is called by the Javascript on the `setup_count` page, which creates a so-called type-ahead for the license plate number form. The

Javascript calls `get_ids()` with any input the user has provided so far, and `get_ids()` will return all license plate numbers of orders assigned to the current location that contain the same sequence of numbers somewhere within the ID. This allows for quick input if no RF-scanner is used as a keyboard.

After checking whether the user is logged in, the current input is retrieved. This input is passed to the page by providing an argument in the website url. E.g. if `/get_ids?q=111` is loaded, the current entry by the user is 111. More on this [here](#). Next, the license plate numbers in the global list `all_current_orders_at_location` created previously when the `setup_count` page was loaded are compared with the ID entered, and any matches are added to the list to be returned. Then, this list is returned in the JSON format as a list of dicts.

A site similar to `/get_ids` is `/get_full_orders`. This function is called by the Javascript on the `setup_count` page, using the license plate input from the user. The Javascript calls `get_ids()` with what is assumed to be a complete license plate, and `get_full_orders()` will return the order id, license plate and requested quantity for all orders containing the input license plate. This information will then be displayed in a table on the `setup_count` page

After checking whether the user is logged in, the current input is retrieved. This input is passed to the page by providing an argument in the website url. E.g. if `/get_ids?q=111` is loaded, the current entry by the user is 111. More on this [here](#). Next, the license plate numbers in the global list `all_current_orders_at_location` created previously when the `setup_count` page was loaded are compared with the ID entered, and any matches and the corresponding order id and quantity requested are added to the list to be returned. Then, this list is returned in the JSON format as a list of dicts.

Lastly, the application is launched at the `host_address` specified.

5. Helper functions

Retrieve_order.py

The function `update_weights()` is used to attempt to update the local copy of the Google Sheets [document](#) 'DC001 On Hand Items.csv'. Since Sager does not yet have a central weight database, this is an attempt to simulate it. The code here will attempt to retrieve the data from the Google sheets [document](#), and write it to the local csv. The code is using the `requests` module, which is documented [here](#), with a more relevant application template [here](#). The code will also write a time stamp to the last row of the csv file, such that this time stamp can later be displayed on the website. It will then return the timestamp to `application.py`.

If no data could be downloaded from the Google sheet, it will print the exception that occurred and continue to use the older, local file with weights. It will return the time stamp at the bottom

of the local csv file. When the website is used with a VPN, it is not possible to also connect to the Internet, so in that case a connection to the Google sheet cannot be established, and the data will not be updated. On-site, a VPN connection should not be necessary, so this will be fine.

The function `retrieve_all_tasks()` takes in the DC zone code, and returns all the available picking orders assigned to shelves. Each order returned contains the following information: task ID, license plate number, product weight, weight unit, product id and picking quantity.

A variable `use_actual_database` determines where the order information will be retrieved from. If False, it will return dummy values (find these at the bottom of `retrieve_order.py`). If True, order information will be taken from the Sager test Oracle database and product weight information from a local csv file based on the Excel document of the on-hand items we received. Below, the case when `use_actual_database` is True is detailed.

First, a connection is made to Sager's test Oracle database by calling the function `setup_conn` (detailed below). If no connection to the database could be made it will return 'No connection'. Then, using a SQL command, executed and fetched by the cursor object, the `task_id`, `license_plate_no`, `allocated_qty`, `product_id` and `source_location_no` are retrieved from the table called `task_master`, where the `task_type` is 'PICKING' and the `task_status` is 'AVL'. It will return 'Retrieval Error' if there was an error retrieving this data. To see particular retrieval errors, see the exceptions printed in the terminal window.

These orders are then filtered for those in the shelves (i.e. those whose `source_location_no` has the first letter 'S'). This is done by using a [lambda](#) function which takes in the first letter of the `source_location_no`, checks if it is 'S', and returns a list of only those orders for which this is true. The remaining orders are then cross-checked with the file 'DC On Hand Items.csv', loaded from the Google Sheets file, as described above. This is done using a python module called [pandas](#) which reads in the relevant fields, "Weight", "Weight UOM" and "MOVE Part Number" into a data array. The `product_ids` of the orders extracted from the database, which correlate to the MOVE Part Number in this file, are then used to check if the file contains data for the product's weight and a corresponding weight unit. If so, these are extracted, added to the order, and the order is added to a final list of valid orders. If no orders can be found in the database, or if no retrieved orders have weight data on file, then 'No orders' is returned. Finally, this list of valid orders is reformatted into a dictionary, and returned. If any other error occurs in the process it will return 'General Error', and print exceptions in the terminal window.

SetupConn.py

The function `setup_conn()` sets up JDBC connection to Sager's test Oracle database. It takes no inputs and returns a cursor object. This cursor object is a control structure that enables the execution of SQL commands to manipulate records in the database.

The python modules necessary to do this are [jaydebeapi](#) and [jpytype](#). Jaydebeapi allows a connection to be made from Python code to a database (like [Oracle](#)) that uses Java JDBC (i.e. Java Database Connectivity) and provides a database [API v2.0](#). For the JDBC connection to be made, a relevant driver is required (this was provided by us and installed during setup). It is then necessary to have JPytype Java integration to make use of this driver, and provide access to Java from within Python.

Within `setup_conn()`, the `if` statement determines whether or not the database connection has been made previously, that is, if the VM (Java Virtual Machine) has been started. It does this by calling the JPytype methods `isJVMStarted()` and `isThreadAttachedtoJVM()`.

If the connection has not been made previously (as in the `else` case), then the JVM is started. This is done by specifying the path to the JDBC driver and calling the `jaydebeapi` `connect` method. The first argument to connect is the name of the Java driver class. We used an `OracleDriver`, and more specifically, the [Thin Client](#), as recommended by [Oracle](#). The second argument is a string containing the connection url: “hostname:port:dbname”. The third and final argument is the username and password, [“username”, “password”]. The cursor object is then called from this connection, and is then returned.

If the JVM has been started previously, the method `attachThreadToJVM()` must be called to make the JVM usable from the cursor. This is documented [here](#).

readScale.py

This script contains two functions to read out data from the Dymo scale. `accurateReading()` at the bottom is the recommended function, it will return a stable value by recalling `readScale()` until it returns a consistent weight. `accurateReading()` then returns this stable value.

`readScale()` takes in the requested unit of the weight, and returns the current readout from the scale. It is largely based on this [website](#), which has detailed information and documentation. Using the `pyusb` module it will retrieve a packet of data from the USB scale. The data takes the form of a list. Indices 0 and 1 are unused during normal operation. Index 2 indicates whether the weight units supplied are in oz (=11) or in grams(=2). Index 3 denotes a scaling factor, which is only used if the scale is using ounces. The scaling factor can be calculated as $10^{(data[3] - 256)}$. The actual reading can then be determined using indices 4 and 5, such that in grams mode the weight is `data[4]+256*data[5]`, and in ounces it is `=scaling factor * (data[4]+256*data[5])`. The script will then convert the reading to the weight unit requested and return it. If no connection to the scale could be established, ‘Not connected’ is returned to the Python script. If the weight unit requested is unsupported (not contained in (lower or upper case): ‘g’, ‘kg’, ‘kgs’, ‘oz’, ‘lb’, ‘lbs’, ‘pounds’), it will return ‘Unsupported UOM’, which will be displayed on the count page.

6. HTML/Javascript Explanation

To ensure a user-friendly interface we used HTML and Javascript on the actual websites. The HTML files extend a `layout.html` file and fill the layout with relevant content.

Layout.html

At the top of the layout file we import some css stylesheets both from online sources and our own stylesheet. These will only affect the visual appearance of the forms and other components of the website. We also import scripts to set up the JQuery used to communicate with the Python backend.

We can use conditional statements and other dynamic content in html when surrounded by double curly brackets. This allows us to create conditional statements and evaluate any variables passed to the html during `render_template` in Python.

If the user has specified a DC ID, indicated by the `dc_id` being nonzero, a button to change the DC ID is shown, which will GET the `setup_dc_id` page.

After that, the block for the specific site's html is created. All other html files extend the layout by including their html in this block.

setup_dc_id.html

This file holds the form to enter the current DC ID. At the top of the script the Python function to be called on submit is specified, i.e. `setup_dc_id()`. Next, holders for the wtforms are included. In the `render_template` command in the Python backend we passed various variables to the html, including the wtform. In the html the different entries in that form are now accessed and displayed. This is done using the `{{}}` functionality. If the form was previously filled out incorrectly, this will create error messages which are also displayed here. However, on the first GET of the page these errors would usually display as well, which is why a variable `first_load` is passed from Python to indicate whether the error messages should be displayed or suppressed.

Setup_count.html & setup_count.js

This page will display error messages based on the error indicators passed on by the Python `render_template`. It will also display the timestamp of the last weight database update passed from Python.

This file holds two separate forms, both submitting to `setup_count()`. The first form, `retrieve_order_form`, will be used by `setup_count.js` to submit a task ID to the Python backend. It includes a text input for the license plate of the current product, and a button which will trigger the javascript. This button only occurs if no errors occurred on the Python backend, otherwise

the button shown will just reload the page. The second form has wtform entries for the manual order similar to those in begin.html with relevant error messages.

Any error messages passed on by the Python backend are displayed here, as well.

Additionally, at the top of setup_count.html the Javascript for this page is included, which is located in setup_count.js.

The setup_count.js has two main functions. It creates content based on the license plate number input to show matching tasks, and it creates a type-ahead for the license plate number.

When no errors occurred previously and the button under the text input is clicked, `get_task_ids()` in the script is called. This function will retrieve and check the input, by comparing it to the global variable `min_length_correct_LP`, which is defined at the top of the script and describes the minimum length for an LP to be possibly correct. If the ID is incomplete, it will call `incomplete_id()` which creates and displays an error message to the user. `get_user_ids` will then call `get_full_orders()` passing the LP. That function will use [Jquery](#) to retrieve all orders containing products with the LP entered from the Python backend. It passes the LP as an argument. It will then pass on the data retrieved to `create_table_from_order_info()`. That function is used to dynamically generate a table with a header to contain the order ID, LP, quantity requested and a submit button for each matching order retrieved. If there were errors in the Python backend, the table will display an error message. Most of this function is just straight-forward HTML creation. The buttons added will call `table_button_clicked()` passing on a variable indicating which order was clicked, and the data retrieved from the Python backend previously. This new function will then generate a checkbox that the user needs to check to confirm the correct target quantity. If the quantity is incorrect, they are asked to input the LP again. If the checkbox is clicked, `confirm_check_box_clicked()` is called passing in the data retrieved and the indicator which order was selected. This function will now create a hidden field inside the `retrieve_order_form` and set it's value to the order ID to be able to pass the order ID to the Python backend. It then submits the form.

This Javascript also generates the type-ahead on the license plate number input. It contains two functions. The first function, `get_ids()` is used to retrieve possible matches from the Python backend given the input provided so far by the user. It uses [Jquery](#), which allows you to retrieve data from the Python script by loading a site. In this case, we also need to specify the input provided, which is done by declaring and passing on a parameter. If the query is successful, the results, which are in the form of a list of dicts are returned. Otherwise, an empty list is returned and the error is logged to the console.

These results are requested and evaluated by the second function, which is anonymous. It is called when the site is loaded, and will configure and fill the type-ahead. Here, a few

characteristics of the type-ahead can be configured, like whether the matching parts of the suggestions are highlighted (highlight), how many characters need to be entered before the type-ahead will show anything (minLength), how many suggestions are shown (limit), which function will supply the data (source), and which part of the data will be shown, once a user selects an option (display). Additionally, the suggestions are included in the html (templates) using [Handlebars](#), which is just a tool to make HTML creation within Javascript easier.

Count.html & count.js

In count.html we first set up a table including relevant information for the user and fill it with variables passed from the Python backend. Next we create the initial counting status indicators, a checkbox that confirms the scale is ready for use, and then a variety of empty paragraphs. Many of these have ids to be able to change their inner text later from the javascript. We also create a submit button that cancels the count. To dive deeper into the functionality of this site we will now consider the javascript included at the very top of count.html.

At the top of scripts.js, two global variables are created, one of them describing whether the counting process is currently running, and the other describing whether the script is currently waiting for a response from the readScale functions in Python.

As the page is loaded, these variables are reset to 0, and an event listener is added to the checkbox that is on the initial html page. When the checkbox is clicked, the function confirm_tare_clicked() is called. This function removes the checkbox and creates a start button with an event listener and adds it to a div in the html called start_stop_buttons. When the start button is clicked, the function start_button_clicked() is called. This function removes the start button, and tries to remove any existing error messages (e.g. if the scale is not connected) and creates a stop button with an event listener. It also calls retrieve_readings().

When called, retrieve_readings() will check whether the counting process is active, and if so, call the update_displays() function. update_displays() sets up the connection to the Python backend using [Jquery](#) as described previously and indicates that a reading is being retrieved by setting the relevant global variable to 1. It will attempt to get the current readings in a JSON format from the /check_weight page. If it is successful, it will first check whether a connection could be established. If not, it will stop the counting process, remove the stop button and create a start button. If successful, it will evaluate the data, which consists of a list ([current weight, current count, indicator whether count is correct (0), over (1) or under (-1), unit of the weight]). It will update the displays on the website by changing the inner html of the relevant paragraphs and then reset the weight_retrieval_in_process variable back to 0. It will then call retrieve_readings again. If it cannot get valid data from /check_weight due to an internal server error, this means that some other error occurred. This will stop the count and reset the counting_process_active

variable to 0, and show a message to the user to check the errors in the command window. It will also remove the stop button and create a start button.

If the readings could be properly retrieved and the user clicks the stop button, `stop_button_clicked()` is called. This will stop the counting process (`counting_process_active = 0`) and wait until the current weight retrieval process is completed. After replacing the stop button with a start button, this function will also check whether the count is complete. If the count is incomplete, it will show a message to the user to continue counting or cancel, by adding a message to one of the `<p>` paragraph holders on the page. If the count is complete, the function will create another checkbox, to confirm the user takes the item off the scale and repackages them if necessary. This checkbox is added to the bottom of the page, and when clicked it will trigger `confirm_checkbox_clicked()`.

`confirm_checkbox_clicked()` will remove the confirm checkbox and create a submit button. When this button is clicked the function `complete_count()` is called.

`complete_count()` will then create a hidden field on the page and submit this field to the Python backend.

7. Batch Scripts

Sager_counting.bat

First the location of the application folder is defined. Next, another batch script at the location specified will be started. That other batch script will open up CHrome after a delay at the localhost site. Then, this batch script will start `application.py`, by running the script at the specified location. These locations need to be accurate.

Sager_counting_setup.bat

This batch script will first uninstall and then reinstall all python modules required.

Start_chrome_after_delay.bat

This batch script will first wait for about 5 seconds and then open up Chrome at the localhost. The delay is used to make sure that the server is started once Chrome is opened up. Starting up the server can take a few seconds, but rarely more than 5, unless your computer is running a lot in parallel.

8. Troubleshoot scenarios

In every case, make sure the system is set up properly as described in the setup section.

You can also follow those instructions again, this can help clear out most common errors. You may need to contact the IT department or another partner to debug.

If there is an issue with the connection to the scale, as indicated on the count site, please make sure the scale is connected and turned on.

If a general error occurred during the retrieval process, take a look at the command window. If there is any indication of “usb”, “pyusb” or the message “Error in readscales”, in Windows rerun the Sager_counting_setup.bat file by double-clicking, and open up libusb_win32 again. This time, instead of adding a device filter, remove the device filter with the vid:0922 and pid:8003, and then add the filter again like before. In Linux, rerun install_sager_counting.sh. If this does not solve the issue, you should either refer to the IT department or any other person in charge of taking care of temporary support for this solution.

If the setup_count says that no connection to the database could be established, please make sure your computer is correctly connected to the Wifi on-site. If this is the case, please refer to the IT department or any other person in charge of taking care of temporary support for this solution. Show the command line errors, e.g. by taking a screenshot. These should be helpful in finding the error.

If the setup_count says that there was an error during the retrieval of data from the databases, please take screenshots of any error messages and refer to the IT department or any other person in charge of taking care of temporary support for this solution. Show the command line errors. These should be helpful in finding the error. The error would have occurred in retrieve_order.py, most likely. Please check the browser console for any errors, as well.

If the setup_count site says that there has been an unspecified error retrieving the order or a general error, please refer to the IT department or a commercial software partner. Show the command line errors, these should be helpful in finding the error.

If the setup_dc_id page does not contain all DC IDs, these can be added when the select_dc_id class is created. Just add another entry, the same way as the previous were included. This should be done by the IT department or any other person in charge of taking care of temporary support for this solution.

If the count page says that the unit of measurement is unsupported, the uom listed in the on hand items is not one of (lower or upper case): ‘g’, ‘kg’, ‘kgs’, ‘oz’, ‘lb’, ‘lbs’, ‘pounds’. Please either add the unit used in readScale() or change the unit in the on hand items file.

If the setup_count page says that the license plate is incomplete, it is shorter than 3 characters. If this is still valid, please change the variable min_length_correct_LP at the top of setup_count.js.

If the table on the setup_count page says that no orders are available and asks the user to check the command line, there was an error retrieving data from the Python file. Please take a screenshot of the command window and refer to the IT department or any other person in charge of taking care of temporary support for this solution.

If that table says that no orders are available for the LP, this means no task containing this LP with weight data could be found. If the LP is correct, and there is weight data, please check the command line, and potentially have a look at retrieve_order.py, there may be an error occurring there. Again, this should be done by the IT department or any other person in charge of taking care of temporary support for this solution.

If the last update of the weight database is too long ago, please make sure you have Internet access. If you are using a VPN connection, it will not be possible to update the weight data. In that case you can only update the weight data, by disconnecting from the VPN, rerunning the program and loading the setup_count page. Then, the weight database is updated. You can now reconnect to the VPN until you deem an update necessary again. If this still does not work, please take screenshots of the command line window and use the error messages there to try and locate the error in cooperation with whoever can supply support for our solution.

In any other case, please take screenshots of the command line window and use the error messages there to try and locate the error.

B. LED Lighting Solution Documentation

Firmware Installation

Installing the Arduino IDE:

Uploading the firmware to the microcontroller:

1. Open the firmware in the Arduino IDE.
2. Check that the microcontroller is plugged into a USB port, and that the arduino IDE correctly identified the COM port that the microcontroller was plugged into. The following link has additional instructions for this step:
<https://www.digikey.ca/en/maker/blogs/2018/how-to-get-started-with-arduino>
3. Click the “Upload” button to upload the firmware to the microcontroller.

Software Installation:

Installing Anaconda and Spyder:

1. Go to <https://www.anaconda.com/products/individual>.
2. Download the Python 3.7 64-Bit Graphical Installer executable file
3. Run the executable, and click “Next”, “I Agree”, “Next”, “Next”, and “Install” (this will install Anaconda with the default settings).

4. Run Spyder (type “Spyder” in the search bar and hit enter).

Running the code:

1. Unzip the code to a chosen location on the computer.
2. Type “cd \$CHOSEN_DIR” in the Spyder console window, where \$CHOSEN_DIR is replaced by the directory where the code is located.
3. Type “run main.py” in the Spyder console window and hit Go to <https://www.arduino.cc/en/main/software>.
4. Click the Windows installer download link and then click “Just Download”.
5. Run the downloaded executable file. Click “I Agree”, “Next”, and “Install” (this will install the Arduino IDE with the default settings).
6. In the Arduino IDE, go to “Tools/Manage Libraries” and search and install the following libraries:
 - a. Wi-FiNINA
 - b. SD
 - c. Adafruit NeoPixel
7. In the Arduino IDE, go to “Tools/Board:/Boards Manager” and search and install the “Arduino SAMD Boards” package.

Formatting the SDHC Card and uploading the tables:

1. Insert the SDHC card into your computer.
2. Right click on it in Windows Explorer and click “Format”. In the pop-up window that appears, select “FAT32”, check “Quick Format”, and click “Start”.

Uploading the location database:

1. Create a text file named “LED.txt”.
2. Add rows to the file that are formatted as follows:
“[storage location] + tab + [pin number] + tab + [led index]”
Where storage location is the location ID, pin number is the pin connected to the row containing the storage location, and LED index is the index of the LED at that storage location. This must be done for every location that will have an LED connected to the microcontroller.
3. Upload this file to the SDHC card, eject the SDHC card, and insert the SDHC card into the Microcontroller memory shield.
4. Press enter to run the code.

C. Employee Engagement Questionnaire



Employee Questionnaire

Thank you for taking the time to answer these questions. The following questions are used to help improve the work environment at Sager Electronics.

Answer the following multiple-choice questions

1. I understand the processes, methods, systems and procedures

All Most Some Few None

2. I comply with company policies, regulations and codes of conduct

All Most Some Few None

3. I focus more on

Accuracy

Efficiency

Answer the following Yes or No Questions

1. I accept and act on feedback given to me

Yes

No

2. I seek help from my supervisor when needed

Yes

No

Comments: _____

