

March 13th, 2019
Dr. Craig Scratchley
School of Engineering Science
Simon Fraser University
British Columbia, V5A 1S6



RE: ENSC 405W/440 Design Specification for FoodSavr

Dear Dr. Scratchley,

The attached document specifies the design for our capstone project, the FoodSavr. Our goal for this project is to help people minimize the amount of food that gets wasted on a daily basis. FoodSavr is an autonomous robotic system that easily allows a user to keep track of what food they have in their home.

This document aims to list all design specifications associated with our product including software, mechanical, electrical and more. They will be categorized as proof of concept, prototype, and final design to clarify our timeline for this project. Besides the general design specifications, a list of engineering standards is included for which our product will conform to. Finally, a thorough test plan and user interface design can be found at the end of this document.

Our team at Savr Robotics would like to thank you in advance for taking the time to review this document. If there are any raised concerns or questions, feel free to contact me at loshaugh@sfu.ca.

Sincerely,

A handwritten signature in black ink, appearing to be 'LO', written in a cursive style.

Liam O'Shaughnessy
Chief Executive Officer
Savr Robotics



Design Specification

Smart Food Storage Device

Team 6:

Liam O'Shaughnessy

Jay Zhao

Vivian Pan

Zavier Aguila

Abstract

With a growing sense of global responsibility, consumers are now looking at their lifestyles and finding ways, small and large, to lessen their negative impact and increase their positive impact on the world.

FoodSavr is an automated food tracking device that aims to address these growing concerns while saving money for the consumer. It will make use of robotics and image processing to aid the user in dealing with their grocery-bought items. From the moment foodstuff arrives on your counter, it will track the item along with its expiration date and amount to help with inventory management. It will prevent redundant purchases and food spoilage by providing an accessible, easy-to-use inventory list to reference during their next trip to the supermarket. It will also provide appropriate notifications when an item nears its expiration and will suggest ways to use it up.

This document lays out the numerous requirements needed to create such a product. It goes into detail of the System, Software, Mechanical, Electrical and Hardware requirements, providing justification and explanation of the requirements. It also discusses concerns with Engineering Standards and Sustainability and Safety and lays out requirements regarding these concerns.

With the requirements clearly stated and outlined, this document will provide a good reference during the product's development as well as detail what the product will do when finished.

Table of Contents

Abstract.....	I
Glossary.....	IV
List of Tables and Figures	IV
List of Figures	IV
List of Tables.....	V
1. Introduction/Background	1
1.1 Scope	1
1.2 Intended Audience	2
1.3 Requirement Classification.....	2
2. System Design.....	3
2.1 Arduino Microcontroller	3
2.2 Raspberry Pi Computer	5
2.3 Camera Component.....	7
3. Mechanical Design	9
3.1 Arm component.....	9
3.2 Base Component.....	12
4. Software Design	14
4.1 Barcode Detection	14
4.2 Expiry Date Detection	16
4.3 Online List.....	19
5. Power Distribution Design	21
5.1 Maximum Power Requirement	21
5.2 Power Design Circuit.....	22
6. Conclusion	23
6.1 System Design	23
6.2 Mechanical Design.....	23
6.3 Software Design.....	23
6.4 Power Distribution Design	24
7. References.....	25
Appendix A: Acceptance Test Plan.....	27

8.1 Mechanical Testing	27
8.2 System Testing	28
8.3 Software Testing	28
8.3.1 Image Processing Testing	28
8.3.2 User Interface Testing	31
8.4 Usability Testing	33
8.5 Safety Testing	33
Appendix B: User Interface.....	35
9.1 Introduction.....	35
9.1.1 Purpose.....	35
9.1.2 Scope	35
9.2 User Analysis	35
9.3 Technical Analysis	36
9.3.1 Discoverability/Visibility	36
9.3.2	37
9.3.3 Mappings	37
9.3.4 Consistency	38
9.3.5 Affordances.....	38
9.3.6 Signifiers	38
9.3.7 Constraints.....	38
9.4 Engineering Standard.....	39
9.5 Analytical Usability Testing.....	39
9.6 Empirical Usability Testing	40
9.6.1 Indicator LEDs	40
9.6.2 Item Placement.....	40
9.6.3 Web Page.....	41
9.7 Conclusion	41

Glossary

API	Application Programming Interface
Bootstrap	Mobile Friendly UI Library
EAN	European Article Number
Firebase	Google's server solution
FoodSavr	Product Name
GPIO	General Purpose Input/Output
JSON	JavaScript Object Notation
LED	Light Emitting Diode
PWM	Pulse Width Modulation
PoC	Proof of Concept
RGB	Red Green Blue
Transformation Matrix	The matrix that describes a rotational and translational from one one frame to another. Ex. ${}^A_B T$ from frame A to frame B
UI	User Interface
UPC	Universal Product Code
USB	Universal Serial Bus
Zbar	Barcode Detection and Identification Library

List of Tables and Figures

List of Figures

FIGURE	DESCRIPTION
FIGURE 2.1	System Overview
FIGURE 3.1	Engineering Prototype of FoodSavr
FIGURE 3.1.1	Proof of concept mechanical arm
FIGURE 3.1.2	Minimal reachable workspace
FIGURE 3.1.3	Maximum reachable workspace
FIGURE 3.1.4	Reachable path
FIGURE 4.1	Software Design Overview
FIGURE 4.1.1	Barcode Example 1
FIGURE 4.1.2	Barcode Example 2, Cola
FIGURE 4.1.3	Barcode Example 3, Shaving Cream
FIGURE 4.1.4	Barcode test image and Zbar test results
FIGURE 4.2.1	Vision API Feature List
FIGURE 4.2.2	Vision API Text Detection
FIGURE 4.2.3	Example Image to be Processed
FIGURE 4.2.4	Vision API Response and Performance
FIGURE 5.2.1	Power Distribution Circuit

List of Tables

TABLE	DESCRIPTION
TABLE 1.3.1	Requirement Classification Symbol
TABLE 2.1.1	Arduino Leonardo Specifications
TABLE 2.1.2	Arduino Microcontroller Design Specification
TABLE 2.1.3	Ultrasonic Sensor HC-SR04 Specifications
TABLE 2.2.1	Raspberry Pi 3 Model B Specifications
TABLE 2.2.2	Main Computer Design Specification
TABLE 2.3.1	Camera Design Specification
TABLE 3.1.1	Mechanical Arm Design Specification
TABLE 3.1.2	Motor Dimensions Specifications
TABLE 4.1.1	Barcode Processing Design Requirements
TABLE 4.2.1	Expiry Date Detection Design Requirements
TABLE 4.3.1	Online UI Design Requirements
TABLE 5.1.1	Max operating power requirement for each component
TABLE 5.2.1	Electrical Requirements

1. Introduction/Background

In recent times, consumers have enjoyed a surge in the variety and availability of various food products. Seasonal fruits and vegetables have now been a year-round staple, ever available in the produce aisle of the local grocery. Big-box retailers encourage consumers to buy large number of products with discounted prices.

With the arrival of the luxury of abundance and choices comes with its pitfall- waste, specifically food waste. The Food and Agriculture Organization of the United Nations have estimated that $\frac{1}{3}$ of all food products are wasted. In North America, about 40% of the total food waste happens with the consumer, while 60% happens between the production and retailing life cycle of the products [1].

An increase in food waste also causes in an increase in harmful gas emissions. Landfills filled with food waste decompose and release methane gas, and they have become one of the largest contributors to greenhouse gases [2].

The aim of FoodSavr is to address the food waste on the consumer front by addressing the redundant product purchases and the eventual spoilage caused by unused or excessive food products.

There's currently no device on the market which organizes food items effectively and intelligently. We saw this as an opening to create a product which can be integrated into the home to easily catalog our food purchases. The FoodSavr can help consumers keep track of all the items in their fridge and pantry, as well as their expiry dates. The automation system we are creating will minimize user efforts in the cataloging process.

1.1 Scope

This document will outline each of the design specifications of FoodSavr. It provides a system overview of the product, as well as descriptions on each subsystem. Each section provides the details and justifications of the design requirements for each component of the product. This documents also contains a engineering prototype test plan in Appendix A as well as information on the User Interface of the product in Appendix B.

1.2 Intended Audience

This document is written as a functional and structural requirement guide for our product FoodSavr. The intended audience are members of Savr Robotics, Craig Scratchley, Andrew Rawicz, teaching assistants, and any future partners or clients.

1.3 Requirement Classification

In this document, we are using the following qualifiers for each of our requirements:

Table 1.3.1: Requirement Classification Symbol

Legend	Stage
C	Proof of Concept
E	Engineering Prototype
F	Final Product

2. System Design

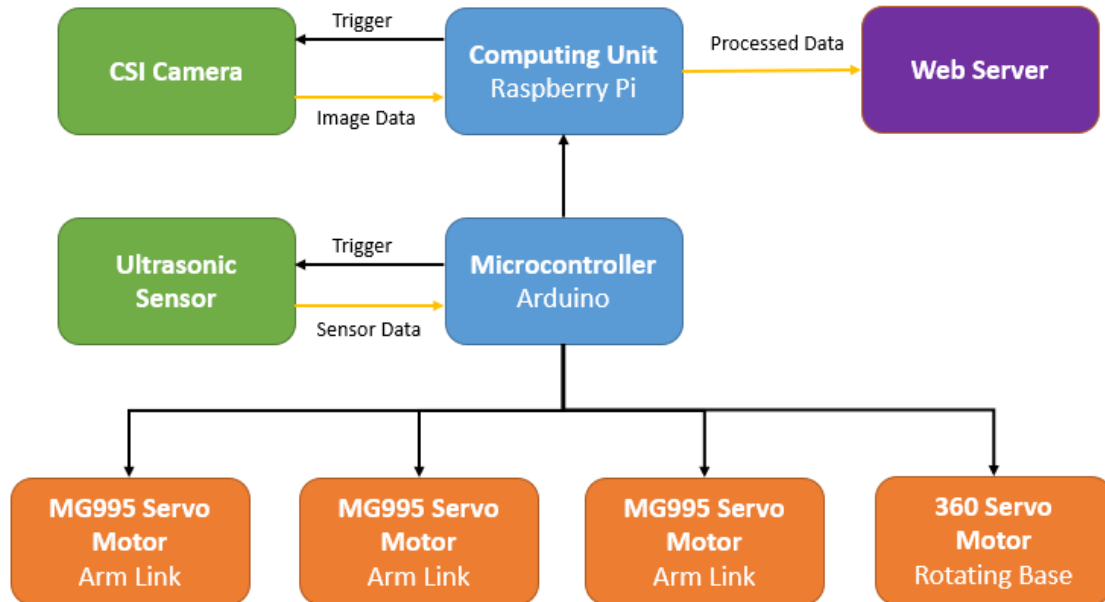


Figure 2.1 – System Overview

2.1 Arduino Microcontroller

The Arduino Leonardo will be used as the sensors and actuator controller for the PoC and Engineering Prototype of the FoodSavr. The Arduino Leonardo will actively listen for a signal from the ultrasonic sensor which will actuate the motors attached to the robotic arm and rotating base. The key specifications of the Arduino Leonardo are shown in Table 2.1.1.

Table 2.1.1 – Arduino Leonardo Specifications

Specification	Arduino Leonardo
Operating Voltage	5V
Digital I/O Pins	20
PWM Channels	7
Analog Input Channels	12
Flash Memory	32 KB
SRAM	2.5 KB
Length	68.6 mm
Width	53.3 mm
Weight	20g

The Arduino Leonardo provides enough pins for four motors and the ultrasonic sensor. However, it does not provide enough voltage to power all these peripherals. Therefore, the motors will be powered by an alternate power source.

We chose the Arduino Leonardo board as our controller unit due to the open source resources available for programming an Arduino microcontroller and its easiness to use.

Table 2.1.2: Arduino Microcontroller Design Specification

Des 2.1.1 C	The Arduino controller will actively listen to sensor signals when powered on
Des 2.1.2 C	The Arduino controller will perform inverse kinematics calculations for each arm positions base on sensor output
Des 2.1.3 C	The Arduino controller will control the rotation of the motors
Des 2.1.4 C	The Arduino controller will relay signal to Raspberry pi to acquire an image when the arm positions are set

We will be using the standard HC-SR04 ultrasonic sensor for object detection. The ultrasonic sensor will be mounted by the edge of the base pointing towards the center of the base. Therefore, when an object is present the echo from the ultrasonic sensor will display a distance between the edge of base and the center of the base. [3]

$$distance = \frac{speed\ of\ sound * time}{2}$$

Table 2.1.3 – Ultrasonic Sensor HC-SR04 Specifications

Specification	Ultrasonic Sensor HC-SR04
Operating Voltage	5V
Operating Current	15mA
Max Range	4m
Min Range	2cm
Measuring Angle	15 degree

The arm component will be set to appropriate positions to perform image acquisition, therefore inverse kinematics much be performed to obtain the angles for each arm segment. These calculations will be depending on the distance between the arm and the food item. We have a transformation between the base of the arm to the camera component of the arm as the matrix ${}^B_C T$, and we want to solve for $\theta_1 \theta_2$

$${}^B_C T = \begin{bmatrix} c_{123} & -s_{123} & 0 & l_2c_{12} + l_1c_1 \\ s_{123} & c_{123} & 0 & l_2s_{12} + l_1s_1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

To solve for θ_1 θ_2 we will use the following formulas [4].

$$x = l_1c_1 + l_2c_2$$

$$y = l_1s_1 + l_2s_2$$

$$c_2 = \frac{x^2 + y^2 - l_1^2 - l_2^2}{2l_1l_2}$$

$$s_2 = \sqrt{1 - c_2^2}$$

$$\theta_2 = \tan^{-1} \frac{s_2}{c_2}$$

$$k_1 = l_1 + l_2c_2$$

$$k_2 = l_2s_2$$

$$\theta_1 = \tan^{-1} \frac{y}{x} - \tan^{-1} \frac{k_2}{k_1}$$

2.2 Raspberry Pi Computer

The Raspberry Pi 3 Model B will be the embedded computer within the FoodSavr. This will be the main driver of all the software components to the system. This includes the barcode detection, expiry date detection, and the communication control that interfaces with the Arduino microcontroller. Some of the key features of this single board computer can be seen below in Table 2.2.1.

Table 2.2.1 - Raspberry Pi 3 Model B Specifications

Specification	Raspberry Pi 3 Model B
SoC	Broadcom BCM2837 64bit
CPU	Quad Core 1.2GHz Arm Cortex A53
RAM	1GB SDRAM
Storage	Micro-SD
Network Connectivity	Ethernet 100
GPIOs	40
Ports	CSI, HDMI, Headphone, Composite
Bluetooth	Bluetooth 4.0
Power Source	Micro USB (up to 2.5 A)

This board provides us with enough computing power to process all data while keeping the board compact. The CSI camera port will be used with the Raspberry Pi Camera Module V2 to allow the FoodSavr to capture images of food items for processing.

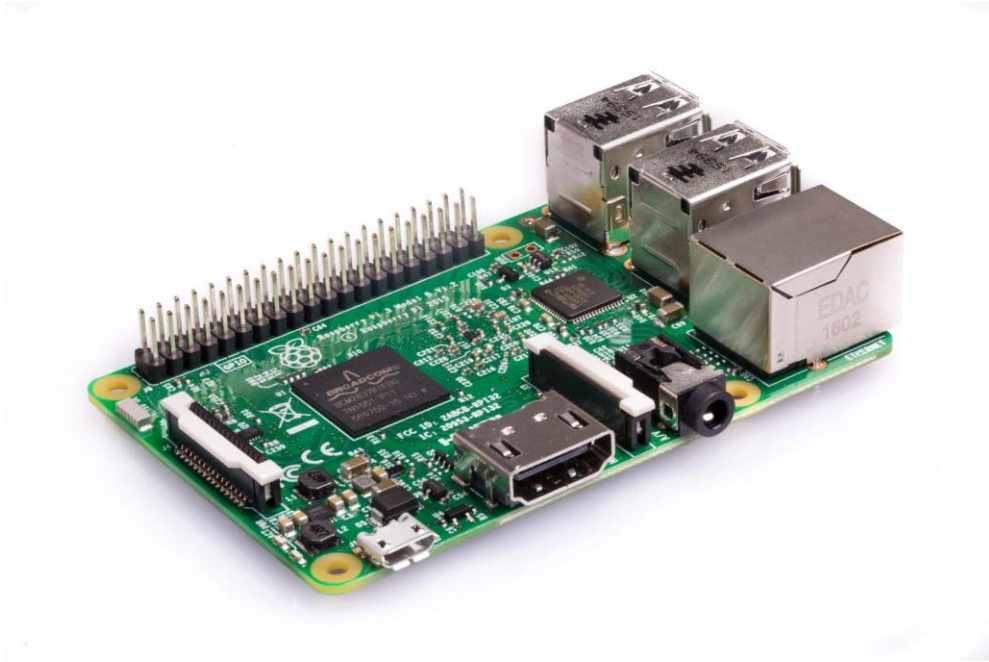


Figure 2.2.1 – Raspberry Pi 3 Model B

Since the board does not contain a WiFi module, a CanaKit Raspberry Pi WiFi adapter will be used instead of ethernet. This wireless adapter can support up to 150 Mbps network speed which is plenty for data we are sending and receiving. Considering the FoodSavr is intended to be used on a counter top within a kitchen, no assumption can be made that there will be Ethernet available. This will also give users more flexibility on where they want to place the device.

The Raspberry Pi will communicate with the Arduino through a serial connection. This is vital in syncing the robotic arm and the camera. For example, once an item has been placed onto the base, the ultrasonic sensor will detect that object and will signal the Raspberry Pi to begin capturing an image.

Table 2.2.2 - Main Computer Design Specification

Des 2.2.1 C	Processor will handle all data processing
Des 2.2.2 C	Includes CSI interface for camera module
Des 2.2.3 C	The single-board computer will fit within the base component
Des 2.2.4 C	The single-board computer will have wireless communication via WIFI

- It can be focused to a focal length of 15 cm which is ideal for the design of our arm. The focus can be changed, but it has to be done manually.
- Its size a (24mm x 25mm) makes it small enough to easily mount on FoodSavr’s robotic arm.
- It is compatible with the Raspberry Pi 3.

The camera itself will be mounted on the tip of the robotic arm. The camera will be directly attached to the Raspberry PI’s CSI port through a 1 meter ribbon cable which is a long enough that it poses no obstacle to the maneuvering of the robotic arm.

For the purpose of FoodSavr there is no need to stream video data from the camera, it will only take a number of pictures of the item’s packaging. This considerably reduces the processing strain the image processing might take on the raspberry pi.

Table 2.3.1: Camera Design Specification

Des 2.3.1 C	Camera sensor interfaces with Raspberry Pi which will power it as well as collect the image data.
Des 2.3.2 C	Camera sensor focuses to a focal length of 15 cm.
Des 2.3.3 C	Camera shall have a picture resolution of at least 8 megapixels.
Des 2.3.4 E	Camera shall be mounted at the tip robotic arm
Des 2.3.5 E	Camera will take multiple photos of the item’s packaging
Des 2.3.6 F	Camera must be protected from tampering from the outside.

3. Mechanical Design

The mechanical design of FoodSavr is the physical scanning device, which composes of two components. A rotating base, and a robotic camera arm. The FoodSavr's key selling component is autonomous scanning with minimal user interaction. Therefore, both arm and base components are crucial to our autonomous scanning process. Figure 3.1 shows the Engineering Prototype design of the FoodSavr.

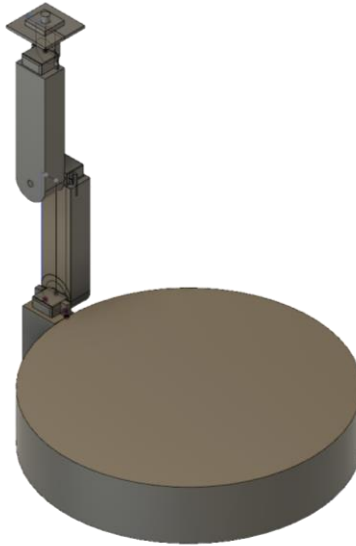


Figure 3.1: Engineering Prototype of FoodSavr

3.1 Arm component

Table 3.1.1: Mechanical Arm Design Specification

Des 3.1.1 E	The mechanical arm will be able to move up and down holding the weight of the arm and the weight of the arm
Des 3.1.2 C	The mechanical arm will have two degrees of freedom
Des 3.1.3 E	The mechanical arm will be able to reach a maximum height of 30 cm
Des 3.1.4 C	Each joint of the mechanical arm will be able to rotate 180 degrees

When the different sizes of items are placed on the base of our device, the robotic camera arm will self-adjust to an appropriate position for image capturing. The arm will contain two links

(arms) with two motors controlling each arm, and a third motor controlling the camera joint to the arms. The proof of concept design of the arm is shown in Figure 3.1.1.

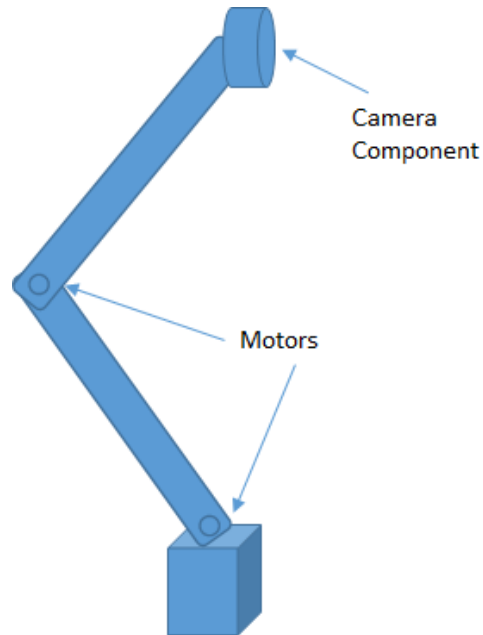


Figure 3.1.1: Proof of concept mechanical arm

The mechanical arm component will have two degrees of freedom, which allows the arm to move freely upwards and downwards. Each motor at the respectively link will have 180 degree of rotational freedom, we only require two degrees of freedom from the arm component. The reachable workspace is shown in figures 3.1.2 – 3.1.4. There exists another degree of freedom at the arm-camera joint, which will rotate the camera component to tilt upwards or downwards.

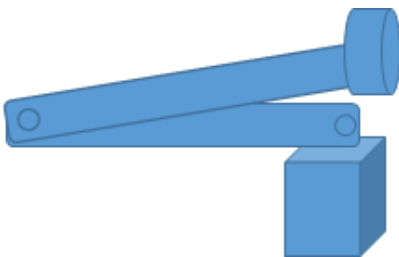


Figure 3.1.2: Minimal reachable workspace



Figure 3.1.3: Maximum reachable workspace

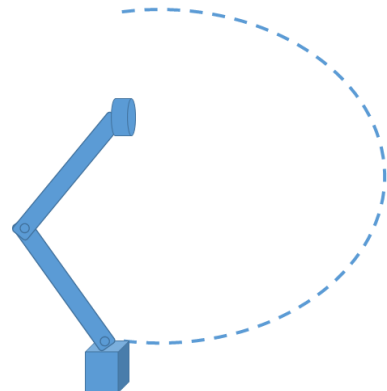


Figure 3.1.4: Reachable path

The motors we will be using the arm component is the MG995 and the motor specifications are shown in Table 3.1.2 and Table 3.1.3. These specifications will be used as our design guideline for Engineering Prototype for the 3D printed mechanical arm, as we will need to leave the right amount of spacing for the motors to fit.

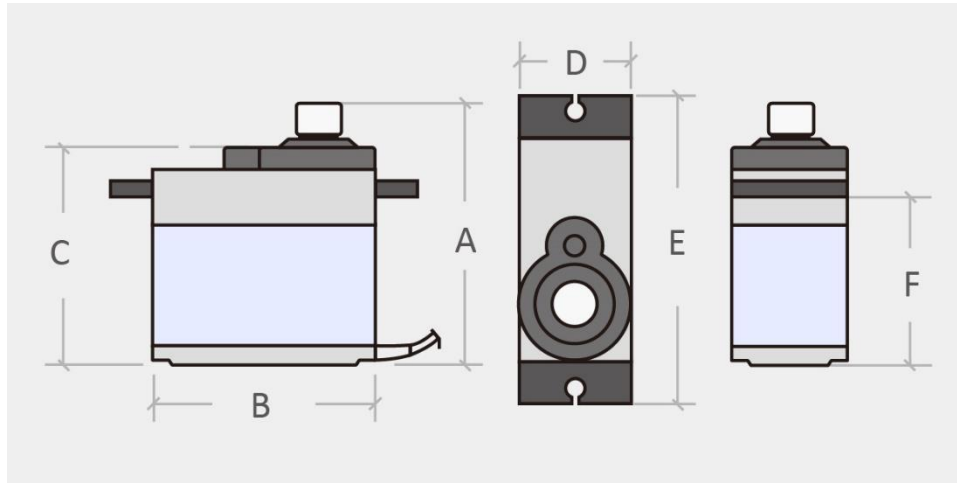


Figure 3.1.5: MG995 Motor front and side views

Table 3.1.2: Motor Dimensions Specifications

A (mm)	42.7
B (mm)	40.9
C (mm)	37
D (mm)	20
E (mm)	54
F (mm)	26.8

The motor in the first link needs to withstand the amount of force exerted by the camera component, weight of two motors, as well as weight of two arms. Whereas the second motor needs to withstand the amount of force exerted by the camera component, the weight of one motor, and the weight of one arm. Given that the motor can lift maximum 11kg per 1cm armature or 366g per 30cm armature, this motor will fit our design requirements if the weight of the camera mount and the weight of two arms are within 360g.

$$\text{weight of two links} = 366g - 3g (\text{raspberry pi camera weight}) - \text{camera mount weight}$$

We are in the process of designing the arms of this robot, in the proof of concept phase of our design we will be using plastic material to construct these links. As plastic will weigh much less compare to wood. For the engineering prototype design for the arm, we will be 3D printing the arm segments to custom fit our motor sizes. The material used for the 3D printing process is PLA which is also light weight in comparison to wood.

Table 3.1.3: Motor Detailed Specifications [5]

Weight	55g
Dimension	40.7×19.7×42.9mm
Stall torque	9.4kg/cm (4.8v); 11kg/cm (6v)
Operating speed	0.20sec/60degree (4.8v); 0.16sec/60degree (6.0v)
Operating voltage	4.8 - 6.6v
Operating temperature	0 – 55 degrees C

3.2 Base Component

The base component of our mechanical design is composed of a rotating plate being driven by a 360-degree motor. The rotating base will serve as an additional degree of freedom to our entire mechanical degree. This degree of freedom will be used to rotate the food item being placed onto the base while the robotic arm segment moves to find the appropriate height to take an image from.

Table 3.2.1: Mechanical Arm Design Requirements

Des 3.2.1 C	The base component will rotate 360 degrees
Des 3.2.2 E	The base component will withhold food items up to 5 kg

The base component design composes of three parts, a stationary base shown in Figure 3.2.1, a rotational plate shown in Figure 3.2.2 and a gear to drive the rotational plate shown in figure 3.2.3. The overall design is shown in Figure 3.2.4.

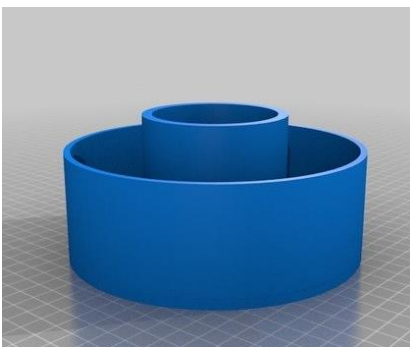


Figure 3.2.1: Stationary Base

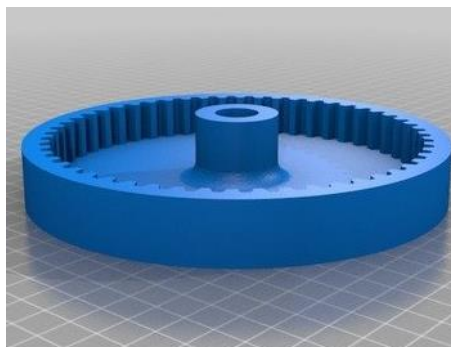


Figure 3.2.2: Rotational Plate

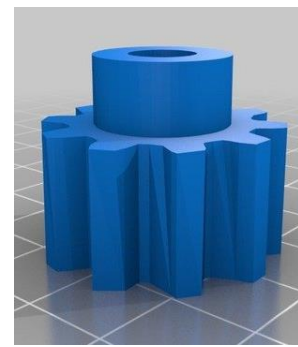


Figure 3.2.3: Gear

The gear's teeth will be aligned to the teeth of the rotational plate. Where the center of the rotational plate will be attached to a center bearing which also fits into the center of the

stationary base. When using a 360-degree servo motor to drive the gear, it will rotate the entire plate component. This is shown in Figure 3.2.4.

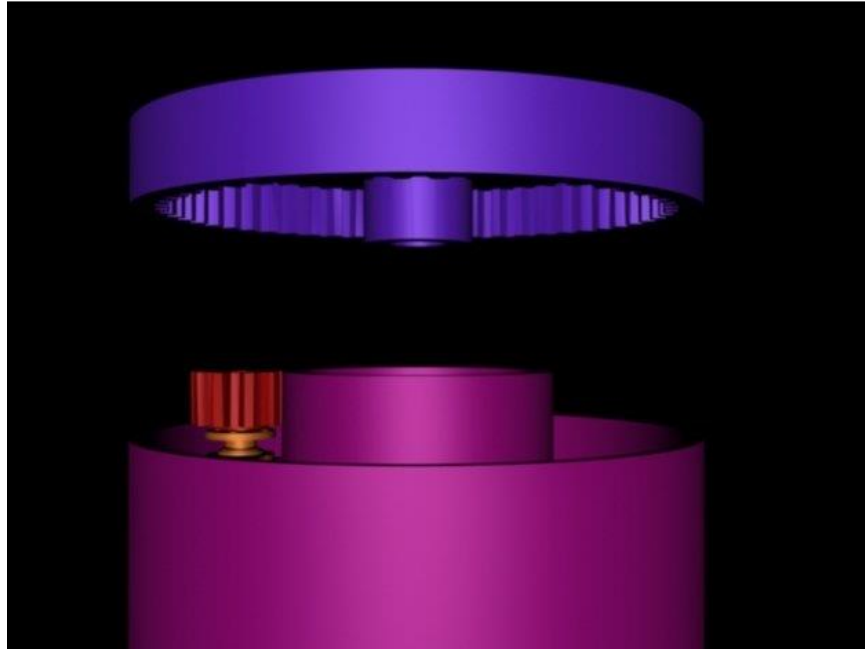


Figure 3.2.4: Rotational Base

For the measurement of our design, we will be designing base on the dimensions of our center bearing. As we will be 3D printing this component of our design, we can easily retrofit the dimensions of each base component base on the diameter of the bearing we will be purchasing.

4. Software Design

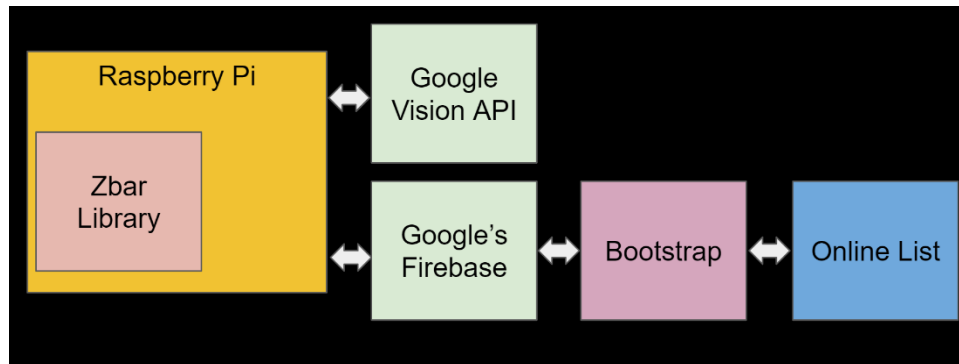


Figure 4.1: Software Design Flow Overview

4.1 Barcode Detection

Since most food items contains barcode information, we will be utilizing this information to acquire most information for our food items. When an image is acquired the barcode information will be scanned. The information extracted from the food items will be stored onto the user profile and database.



Figure 4.1.1 - 4.1.3: Different barcode designs on commonly found grocery items

We will be using the python library *Zbar* to process the barcode information from the acquired images. *Zbar* is an open sourced barcode reader library, its low code complexity and small memory footprint attracts us, as this aligns with the limited computing power of a Raspberry Pi. Therefore, we can perform all barcode processing on the Raspberry Pi. Additionally, the *Zbar* library supports many popular forms of barcode standards, including including EAN-13/UPC-A, UPC-E, EAN-8, Code 128, Code 39, Interleaved 2 of 5 and QR Code [12]. The most common barcode system for grocery packaged goods are the European Article Number or EAN-13 barcode. This library's ability to detect multiple standards of barcodes will be crucial in helping the product scan more products.

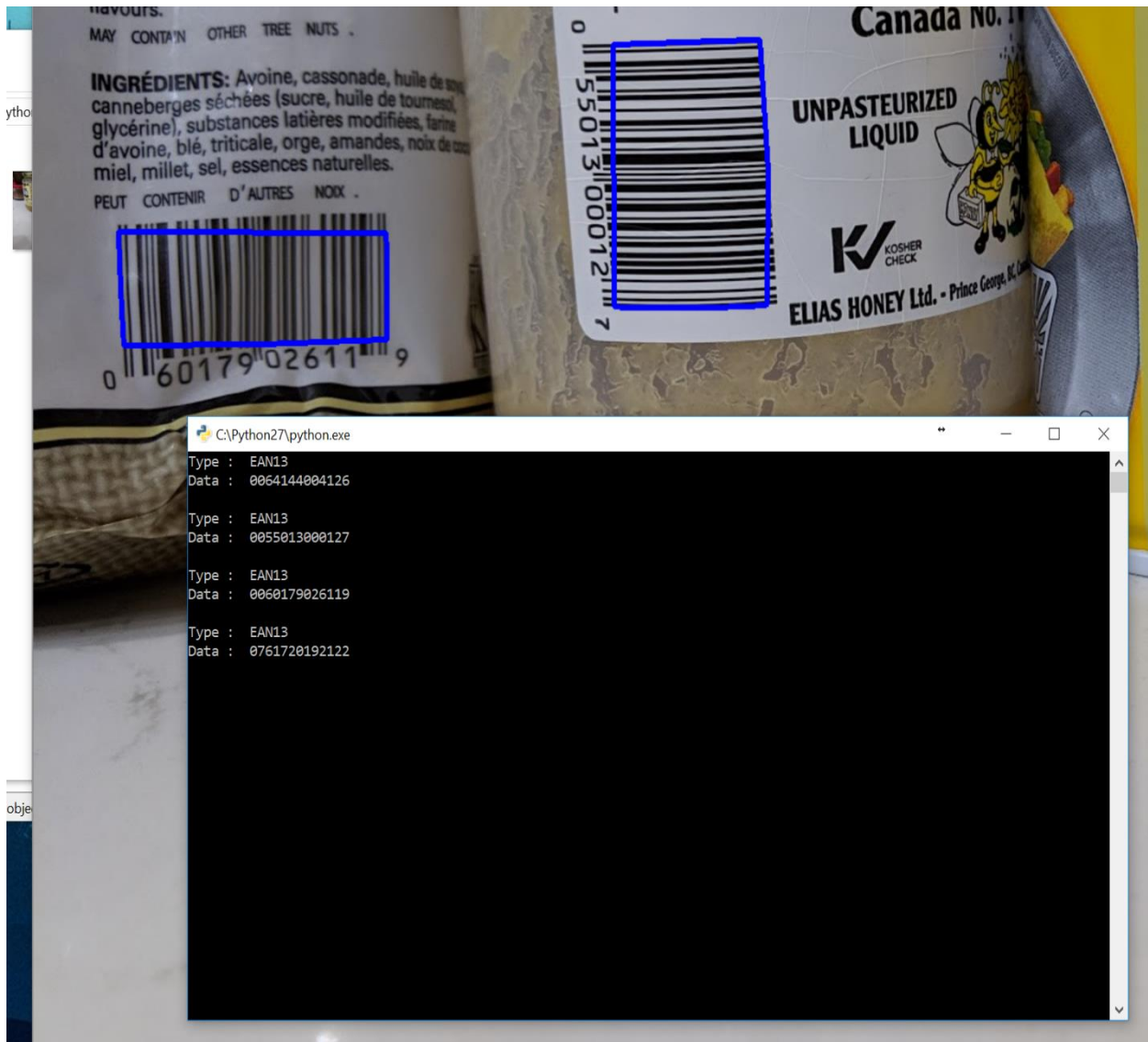


Figure 4.1.4: Barcode Test Image and Zbar results

Zbar only decodes the barcode and provides the number it represents; this number needs to be compared to a barcode database to retrieve the item information. This has presented some difficulty as there is no one database that contains a complete collection of supermarket products, especially if the products are only locally produced. To mitigate some of this loss of data, multiple databases will be accessed to obtain the product's information.

Because multiple barcode database will be used, we will need a method to deal with multiple entries from different databases. A simple ranking system will be used. Each database will be ranked by the team considering their completeness and correctness of the information available. If there are multiple entries from multiple databases, the result from the highest-ranking database will be used.

If FoodSavr comes across any item that it cannot find the information for, it will default to saving an image of item and present that to the user when they accessed their online shopping list.

Table 4.1.1 - Barcode Processing Design Requirements

Des 4.1.1 C	The software shall recognize an item's barcode.
Des 4.1.2 C	The software shall parse the barcode to obtain the UPC.
Des 4.1.3 C	The UPC will be compared to multiple UPC databases and the top result will be used.
Des 4.1.4 E	If item cannot be found the software will instead store the item pictures on the online shopping list.
Des 4.1.5 E	If barcode cannot be found the software will instead store the item pictures on the online shopping list.

4.2 Expiry Date Detection

Along with the barcode, the FoodSavr will also be extracting an expiry date from the image. This allows the user to not only know what food they have in their homes, but also how long it will last and when they need to use it.

To enable this functionality, Google's Vision API will be used. The Vision API is a powerful, cloud-based, machine learning library that allows users to extract different features from an image. These features can be seen below in Figure 4.2.1.

Feature Type	Description
LABEL_DETECTION	Execute Image Content Analysis on the entire image and return
TEXT_DETECTION	Perform Optical Character Recognition (OCR) on text within the image (character limit applies)
DOCUMENT_TEXT_DETECTION	Perform Optical Character Recognition (OCR) on dense text image (premium feature not subject to character limit)
FACE_DETECTION	Detect faces within the image
LANDMARK_DETECTION	Detect geographic landmarks within the image
LOGO_DETECTION	Detect company logos within the image
SAFE_SEARCH_DETECTION	Determine image safe search properties on the image
IMAGE_PROPERTIES	Compute a set of properties about the image (such as the image's dominant colors)

Figure 4.2.1 – Vision API Feature List

For the expiry date detection, the “TEXT_DETECTION” feature will be used. Using this API offers us pre-trained models on huge sets of data collected by Google. For FoodSavr, since the main computer is a Raspberry Pi 3, utilizing Google Vision saves us from doing the processing locally and instead relying on cloud processing. This allows us to process an image in seconds, rather than minutes.



Figure 4.2.2 - Vision API Text Detection

To extract the text, Vision uses Optical Character Recognition (OCR) to convert each individual character within the image to machine-encoded text. This is generally done by breaking down each individual character into different element like curves, corners, and other physical features. Many implementations include post-processing techniques that include using a type of dictionary to determine what kinds of words are “allowed” to be in that image [10].

Vision API returns a JSON object that includes each individual word detected in the image. Using this JSON object, the extraction date will then be parsed locally by the FoodSavr.

Some problems arise when detecting an expiry date that is printed onto the product in a dot matrix style as seen in Figure 4.2.3. To compensate for this, the FoodSavr will pre-process the image before sending a text extraction request. Currently, the pre-processing done locally will only be passing the image through a smoothing filter.

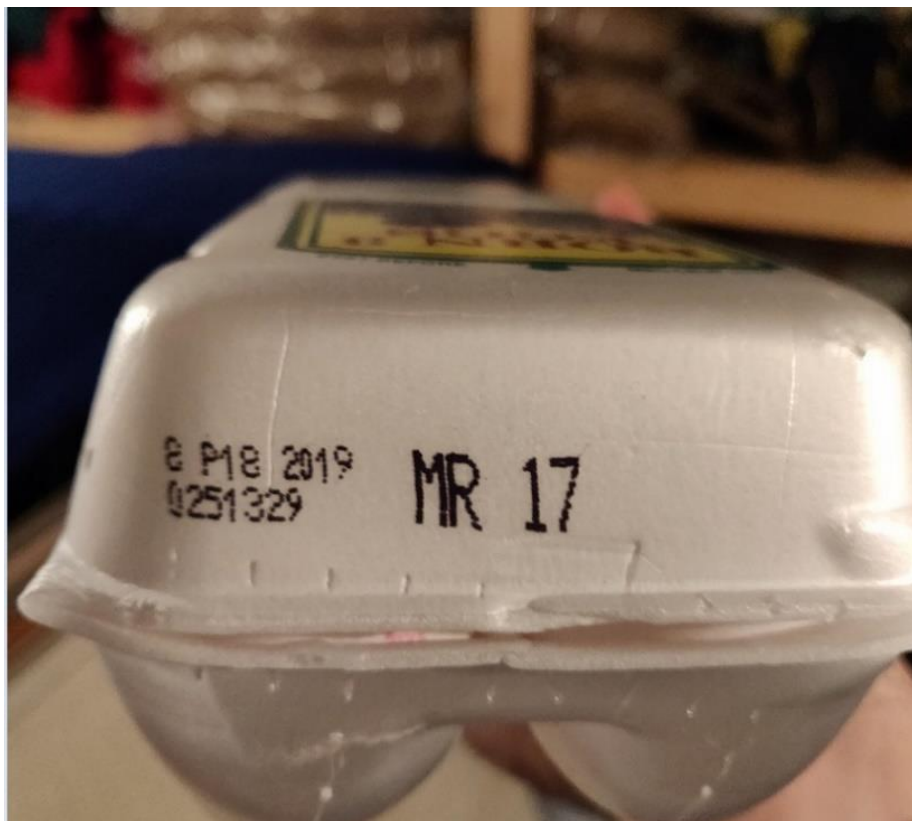


Figure 4.2.3 – Example Image to be Processed

```
(enst013-s2019) C:\Users\rosina\OneDrive\Documents\Spring 2019\ENSC 405 (ImageRecognition)\python_vision_test.py
Start Time: %s 1552504824.5818286
End Time: %s 1552504829.0516992
Total Time Elapsed: %s 4.469870567321777
P18 2019
1251329
MR 17
{'vertices': [{'x': 593, 'y': 2357}, {'x': 2145, 'y': 2357}, {'x': 2145, 'y': 2777}, {'x': 593, 'y': 2777}]}
```

Figure 4.2.4 – Vision API Response and Performance

The image in Figure 4.2.3 contains an expiry date March 17th, 2019 and after passing it through the Vision API, the extracted text is printed out from the JSON object. It also returns the bounding box coordinates, for example these coordinates would correspond to the green boxes found in Figure 4.2.2 above. This process to send a request and receive a response takes roughly 4 seconds as you can see above. This is much faster than doing the image processing locally on the 1.2 GHz Quad Core Broadcom chip found on the Raspberry Pi 3 Model B.

In the case that the expiry date can't be found, the item images will be stored on the online shopping list, similar to when a barcode isn't detected. This would be the case for fresh produce that generally don't contain expiry dates or barcodes.

Table 4.2.1: Expiry Date Detection Design Requirements

Des 4.2.1 C	The software will pre-process an image with a smoothing filter
Des 4.2.2 C	The software will send this image to the cloud for text processing
Des 4.2.3 C	The returned JSON object will parse all text and extract the expiry date
Des 4.2.4 E	If expiry date cannot be found, then the software shall store the item pictures in the online shopping list

4.3 Online List

It is important that the online grocery list is easy to navigate and presents all relevant information to the user. It should also be easy to navigate and edit, it is ideal to mimic the freedom a user might have if they had instead used a handwritten list.

Jane's Online List +








Item Name	Item Quantity	Item Expiration	Date Bought	
Avalon 3.25% Milk, 1L	1	07-23-2019	03-15-2019	 
Roger's Granola, Cranberry Almond, 700g	3	05-15-2019	01-21-2018	 
Oranges	5	03-25-2019	03-15-2019	 
London Drugs, B-12 Capsule, 120 tables	1	12-4-2018	01-01-2020	 

Figure 4.3.1: Online List Mock Design

Just on the main page of the list, the user will be able to edit the items without having to leave the page. If they desire to view a more detailed breakdown of the item, they could click on the item, and they will be brought to a page where more of the item's information can be found. The user can also delete any of the items quickly by pressing on the delete icon that is lined up with the item to be deleted.

The online list's backend will be maintained using Google's Firebase, this design choices helps in the abstraction of server design making it easier to maintain and use an online database that is reliable, scalable and safe.

The online list's front end will be created using Bootstrap. Much like Firebase, it is a well-written and well-maintained toolkit that allows the easy development of a UI while abstracting low-level components that would have otherwise taken up more development time.

Table 4.3.1: Online User Interface Requirements

Des 4.3.1 E	Web UI will be a desktop browser with mobile friendly design
Des 4.3.2 E	The UI will display the food item's name, quantity, date bought and expiration date
Des 4.3.3 E	The user will be able to sort through the entries according to its fields.
Des 4.3.4 E	The user will be able to manually add an item to the list.
Des 4.3.5 E	The user will be able to edit any item's field.
Des 4.3.6 E	The user will be able to add a field to an item.

5. Power Distribution Design

In this section, we will talk about the power design that fits all power requirement for each component. First, we need to find out the maximum current that each part will draw under maximum load.

5.1 Maximum Power Requirement

Table 5.1.1 Max operating power requirement for each component

Component	Max current (mA)	Operating Voltage (V)
Raspberry Pi 3 [13]	980	5
RGB LED X3	30	5
Servo motor (MG995) [14]	350	4.8
Arduino Leonardo [15]	200	5
Ultrasonic Sensor [16]	35	5

The maximum current for each component is shown above, for the Raspberry Pi, there will be a camera, 3 RGB LEDs and an ultrasonic sensor. The maximum power draw from it is:

$$I_{rasp} = 980mA + 3 \times 30mA + 35mA = 1105mA = 1.1A$$

For the Arduino Leonardo, since we are only using it to control the motor, the current draw won't be maximized. However, to consider the safety standards, the maximum current draw will be assumed in this power design calculation, which is:

$$I_{Ard} = 200mA = 0.2A$$

For the Servo motors, the maximum running current is 350mA, and the maximum stalling current is 1500mA. We are only considering the maximum running current, since for this project, the robotic arm is only carrying the camera module. Therefore, the motor won't have its maximum torque applied. The maximum current draw is:

$$I_{servo} = 350mA \times 4 = 1400mA = 1.4A$$

To sum up, the total maximum current draw is:

$$I_{sum} = I_{rasp} + I_{Ard} + I_{servo} = 1.1A + 0.2A + 1.4A = 2.7A$$

Then, we can calculate the maximum power consumption, which is:

$$P = I_{sum} \times V_{oper} = 2.7A \times 5V = 13.5W$$

5.2 Power Design Circuit

Based on the 5.1 maximum power requirement, we have two major current consumption parts, one is the Raspberry Pi related sensors and microcontrollers, the other one is the Servo motors that control the base rotation and robotic arm. We are going to have the two major power consumption parts with two separated power regulators. The power distribution circuit is shown below in Figure 5.2.1.

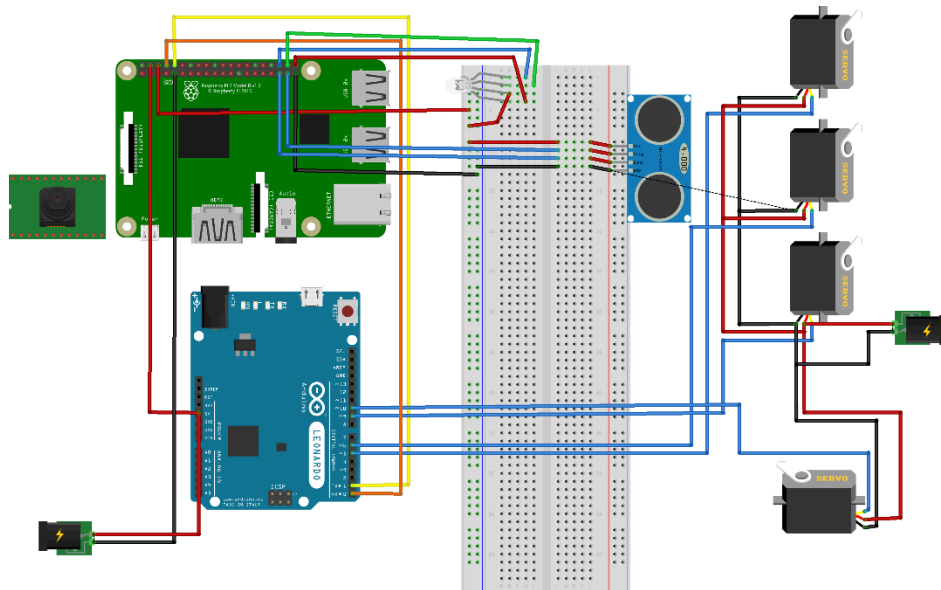


Figure 5.2.1 Power Distribution Circuit

As shown in the diagram, one power regulator located at the bottom left is powering all the components on the left side including the ultrasonic sensor, and the other power regulator located at the right edge is powering all four of the Servo motors. There will be two power regulators that needs to be at least 1.5A/5V current/voltage rating.

Table 5.2.1: Electrical Requirements

Des 5.2.1 C	The Raspberry Pi 3 and the Raspberry Pi camera will take 5V as power input
Des 5.2.2 C	The LEDs will take 5V from the Raspberry Pi GPIO pins
Des 5.2.3 C	The ultrasonic sensor will take 5V from Raspberry Pi
Des 5.2.4 E	The servo motors will take 5V from an external power source

6. Conclusion

This document details the design requirements set out for FoodSavr, a product the team hopes can help with the reduction of consumer's total food waste and keep their food spending in check.

This design specification includes the system, software, mechanical, electrical and hardware perspective which must work correctly individually and as a cohesive unit to accomplish FoodSavr's goal.

6.1 System Design

The overall control and processing system of our product utilizes an Arduino Microcontroller, Raspberry Pi Computer, and a Raspberry Pi Camera. The Arduino will be actuating and controlling the sensors and motors. Whereas the Raspberry Pi will act as the device's main processing unit. The Arduino will communicate to the Raspberry Pi when it is ready to begin the image acquisition and image processing process.

6.2 Mechanical Design

The mechanical component of our product contains both a rotational base plate and a two-link robotic arm. The robotic arm will maneuver the vertical position of our camera, whereas the rotational base will rotate the food item. With this design combination the device will be able to acquire images from all sides for food item within our product size limit.

6.3 Software Design

The software component of our design describes the method of implementing barcode detection, expiry date detection and website UI functionalities. The expiry date detection will be implemented using a cloud-based machine learning API to prevent heavy processing on the main computer. Described a possible implementation for the online list graphical user interface (GUI) using Bootstrap. The backend design for the online list will use Google Firebase.

6.4 Power Distribution Design

Due to the use of multiple electronic components in our design, we need to supply enough power to each of these components. The power design outlines the calculation for the maximum power consumption of the whole system. As well as describing how to satisfy the requirement with a power distribution design.

This document goes into the team's reasoning behind the requirements. The adherence of the product with engineering standards as well as its sustainability and safety are also detailed and discussed. This document also includes an Acceptance test plan which can be used to evaluate the team's body of work, but also test its fulfillment of the previously laid out requirements.

It is important that these requirements are fulfilled to deliver a full working product, and the document aims to shed some light into FoodSavr's functionality.

7. References

- [1] M. Rezaei and B. Liu, "Food loss and waste in the food supply chain", Nutfruit, no. 71, pp. 26-27, 2017.
- [2] B. Adhikari, S. Barrington and J. Martinez, "Predicted growth of world urban food waste and methane production", Waste Management & Research, vol. 24, no. 5, pp. 421-433, 2006. Available: [10.1177/0734242x06067767](https://doi.org/10.1177/0734242x06067767).
- [3] "Ultrasonic Ranging Module HC - SR04", Elec Freaks [Online], Available: <https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf>. [Accessed: 14-Mar-2019]
- [4] John J. Craig, "Introduction to Robotics: Mechanics and Control", Pearson, Fourth Edition, 2017
- [5] "MG665", Tower Pro [Online], Available: <https://www.towerpro.com.tw/product/mg995/>. [Accessed: 12-Mar-2019]
- [6] "IEC TR 61997:2001 | IEC Webstore", Webstore.iec.ch, 2018. [Online]. Available: <https://webstore.iec.ch/publication/6269>. [Accessed: 08-Mar-2019].
- [7] ISO, "ISO 9241-161:2016 Ergonomics of human-system interaction – Part 161: Guidance on visual user-interface elements," <https://www.iso.org/standard/60476.html>, 2016, [Accessed: 08-Mar-2019].
- [8] IEEE, "IEEE Std 1621-2004 - IEEE Standard for User Interface Elements in Power Control of Electronic Devices Employed in Office/Consumer Environments," <https://standards.ieee.org/findstds/standard/1621-2004.html>, 2005, [Accessed: 08-Mar-2019].
- [9] "IEEE Std 1012-2012 (Revision of IEEE Std 1012-2004) - IEEE Standard for System and Software Verification and Validation," <http://standards.ieee.org/findstds/standard/1012-2012.html>, 2012 [Accessed: 08-Mar-2019].
- [10] "Optical Character Recognition (OCR) – How it works", Nicomsoft, 2012. [Online]. Available: <https://www.nicomsoft.com/optical-character-recognition-ocr-how-it-works/>. [Accessed: 13-Mar-2019].
- [11] "Raspberry Pi Camera Module Mechanical Dimensions - Raspberry Pi Spy", Raspberry Pi Spy, 2019. [Online]. Available: <https://www.raspberrypi-spy.co.uk/2013/05/pi-camera-module-mechanical-dimensions/>. [Accessed: 14-Mar-2019].
- [12] "ZBar bar code reader", Zbar.sourceforge.net, 2019. [Online]. Available: <http://zbar.sourceforge.net/>. [Accessed: 14-Mar-2019].
- [13] Power consumption Benchmarks – RaspberryPi Drumble, <https://www.pidramble.com/wiki/benchmarks/power-consumption> [Accessed: 14-Mar-2019].

[14] MG995 Standard Servo, <https://store.makeblock.com/mg995-standard-servo> [Accessed: 14- Mar- 2019].HYPERLINK "https://store.makeblock.com/mg995-standard-servo"HYPERLINK "https://store.makeblock.com/mg995-standard-servo"HYPERLINK "https://store.makeblock.com/mg995-standard-servo"HYPERLINK "https://store.makeblock.com/mg995-standard-servo"HYPERLINK "https://store.makeblock.com/mg995-standard-servo"HYPERLINK "https://store.makeblock.com/mg995-standard-servo"

[15] What's Leonardo 5V pin current limit? - StackExchange <https://arduino.stackexchange.com/questions/28950/whats-leonardo-5v-pin-current-limit> [Accessed: 14- Mar- 2019].HYPERLINK "https://arduino.stackexchange.com/questions/28950/whats-leonardo-5v-pin-current-limit"HYPERLINK "https://arduino.stackexchange.com/questions/28950/whats-leonardo-5v-pin-current-limit"HYPERLINK "https://arduino.stackexchange.com/questions/28950/whats-leonardo-5v-pin-current-limit"HYPERLINK "https://arduino.stackexchange.com/questions/28950/whats-leonardo-5v-pin-current-limit"HYPERLINK "https://arduino.stackexchange.com/questions/28950/whats-leonardo-5v-pin-current-limit"HYPERLINK "https://arduino.stackexchange.com/questions/28950/whats-leonardo-5v-pin-current-limit"HYPERLINK "https://arduino.stackexchange.com/questions/28950/whats-leonardo-5v-pin-current-limit"HYPERLINK "https://arduino.stackexchange.com/questions/28950/whats-leonardo-5v-pin-current-limit"

[16] Ultrasonic Distance Sensor (#28015) <https://www.parallax.com/sites/default/files/downloads/28015-PING-Sensor-Product-Guide-v2.0.pdf> [Accessed: 14- Mar- 2019].HYPERLINK "https://www.parallax.com/sites/default/files/downloads/28015-PING-Sensor-Product-Guide-v2.0.pdf"HYPERLINK "https://www.parallax.com/sites/default/files/downloads/28015-PING-Sensor-Product-Guide-v2.0.pdf"HYPERLINK "https://www.parallax.com/sites/default/files/downloads/28015-PING-Sensor-Product-Guide-v2.0.pdf"HYPERLINK "https://www.parallax.com/sites/default/files/downloads/28015-PING-Sensor-Product-Guide-v2.0.pdf"HYPERLINK "https://www.parallax.com/sites/default/files/downloads/28015-PING-Sensor-Product-Guide-v2.0.pdf"HYPERLINK "https://www.parallax.com/sites/default/files/downloads/28015-PING-Sensor-Product-Guide-v2.0.pdf"HYPERLINK "https://www.parallax.com/sites/default/files/downloads/28015-PING-Sensor-Product-Guide-v2.0.pdf"HYPERLINK "https://www.parallax.com/sites/default/files/downloads/28015-PING-Sensor-Product-Guide-v2.0.pdf"

Appendix A: Acceptance Test Plan

This appendix document describes our acceptance test plan for the FoodSavr. This includes end to end system tests as well as specific module tests.

8.1 Mechanical Testing

The following tests will be performed on the mechanical components of the FoodSavr device, mainly the motors and arm.

Date:	Test Name: Arm Adjustment Test
Test Description: Ensure the mechanical arm with three motors is fast and smoothly moving around the food item	
Expected Outcome: Once the scanning procedure has started, the mechanical arm should be able to move the camera mounted on the tip around the food item to reach its desired position and orientation.	
Actual Outcome:	

Date:	Test Name: Load Support Test
Test Description: Ensure the base can support items with maximum weight of 5 kilograms	
Expected Outcome: The base should be able to rotate freely when holding the maximum weight of the food item without the device falling apart.	
Actual Outcome:	

Date:	Test Name: Base Rotation Test
Test Description: Ensure the base can rotate items smoothly to its desired position	
Expected Outcome: the base plate will rotate the item to certain angle.	
Actual Outcome:	

8.2 System Testing

The following tests will be performed on the system components of the FoodSavr device, concerning the inter-device communications.

Date:	Test Name: Sensors and Actuators Test
Test Description: Ensure each sensor is reading proper values to actuate its corresponding components. Such as the pressure sensor and the hall effect sensors.	
Expected Outcome: Pressure sensor should send signal to Arduino to activate the entire system. Hall effect sensors should monitor the arm positions and deliver feedback to Arduino.	
Actual Outcome:	

Date:	Test Name: Arduino and Raspberry Pi Communication
Test Description: Ensure the Arduino can send signals to Raspberry Pi in real time to activate camera.	
Expected Outcome: Once the mechanical arms have reached its desired position, the Arduino should immediately send signal to Raspberry Pi to activate the camera system.	
Actual Outcome:	

8.3 Software Testing

The following tests will be performed to ensure that the data extraction and upload work as intended.

8.3.1 Image Processing Testing

Date:	Test Name: Text Extraction Proper Orientation
Test Description: Test the text extraction code by starting with an image that has text and is properly oriented towards the face of the object.	
Expected Outcome: The code should return all text labels found within the image	
Actual Outcome:	

Date:	Test Name: Text Extraction Non-Proper Orientation
Test Description: Test the text extraction code by starting with an image that has text and is not oriented properly towards the face of the object (on an angle)	
Expected Outcome: The code should return all text labels found within the image	
Actual Outcome:	

Date:	Test Name: False Text Extraction
Test Description: Test the text extraction code by starting with an image that has no text within the image	
Expected Outcome: The code should return no text labels	
Actual Outcome:	

Date:	Test Name: Expiry Date Parsing with Expiry Date
Test Description: Test the parsing code by starting with an image that has an expiry date within the text	
Expected Outcome: The code should return the expiry date	
Actual Outcome:	

Date:	Test Name: Expiry Date Parsing without Expiry Date
Test Description: Test the parsing code by starting with an image that does not have an expiry date within the text	
Expected Outcome: The code should not return an expiry date	
Actual Outcome:	

Date:	Test Name: Barcode Extraction Proper Orientation
Test Description: Test the barcode extraction code on an image that is properly oriented towards the face of the object and contains a barcode	
Expected Outcome: The code should return the barcode	
Actual Outcome:	

Date:	Test Name: Barcode Extraction Non-Proper Orientation
Test Description: Test the barcode extraction code on an image that is not properly oriented towards the face of the object and contains a barcode	
Expected Outcome: The code should return the barcode	
Actual Outcome:	

Date:	Test Name: False Barcode Extraction
Test Description: Test the barcode extraction code on an image that does not contain a barcode	
Expected Outcome: The code should not return a barcode	
Actual Outcome:	

Date:	Test Name: Barcode Processing
Test Description: Once pictures have been taken by image sensor, the software shall process scan and process the barcode information	
Expected Outcome: Barcode should be processed and core information such as item description, and size and weight information (if available) should be store in the online database	
Actual Outcome:	

8.3.2 User Interface Testing

The following test will be performed on the user interface to ensure users can easily view and edit their inventory list.

Date:	Test Name: Finished Scan Notification
Test Description: Once scanning procedure has completed, the user will be notified by means of an LED Light	
Expected Outcome: An appropriately labeled LED light will turn once scanning is finished, and device is ready for next item.	
Actual Outcome:	

Date:	Test Name: Display Scanned Items
Test Description: Once an item has been added to the database, the user will be able to see it added to the online list	
Expected Outcome: The item's name will appear in the list of scanned items.	
Actual Outcome:	

Date:	Test Name: Scanned Item Fields
Test Description: On the online list, the item should be accompanied by its name, expiry date and quantity	
Expected Outcome: The item will appear in the list of scanned items along with its name, expiry date, date added and quantity.	
Actual Outcome:	

Date:	Test Name: Failed Scan Failsafe
Test Description: If an item is scanned and its expiry date or barcode cannot be identified, the picture of the item will be visible on the online list	
Expected Outcome: The item's pictures will appear in the list of scanned items and be initially labeled as "unknown item"	
Actual Outcome:	

Date:	Test Name: Editing Item's Field
Test Description: On the online list, the user is presented with an option to edit any field of the item.	
Expected Outcome: On the online list, the user has a button to edit the item's information and is able to edit any of the item's field.	
Actual Outcome:	

Date:	Test Name: Adding a field to an Item
Test Description: On the online list, the user is presented with an option to add a field to the item.	
Expected Outcome: On the online list, the user has a button to add a field to the item's information.	
Actual Outcome:	

Date:	Test Name: Delete Item's Field
Test Description: On online list, the user is presented with an option to delete items off the list	
Expected Outcome: The user has a button to delete an item's field	
Actual Outcome:	

Date:	Test Name: Delete Item
Test Description: On online list, the user is presented with an option to delete items off the list	
Expected Outcome: The user has a button to delete an item	
Actual Outcome:	

8.4 Usability Testing

The following test will be performed on the user interface to ensure ease of use.

Date:	Test Name: Automation
Test Description: To start the scanning process, the user only must put the item on top of the plate.	
Expected Outcome: The scanning process should start after user places item on plate.	
Actual Outcome:	

8.5 Safety Testing

Date:	Test Name: Collision safety
Test Description: During arm movement, arm should not damage items being scanned.	
Expected Outcome: The arm should never collide with the item.	
Actual Outcome:	

Date:	Test Name: Moisture lock-out
Test Description: Residual moisture from products must not make it the inner electrical components.	
Expected Outcome: electrical components are safe from moisture	
Actual Outcome:	

Date:	Test Name: Food safety
Test Description: All parts of the product that comes in direct contact with food is made of food-safe material.	
Expected Outcome: all components are made of food-safe material.	
Actual Outcome:	

Appendix B: User Interface

9.1 Introduction

As FoodSavr is targeted as a general consumer product, therefore the user interface behind this product can significantly impact the success with our target audience. Our goal is to design a simple product with minimal interactions between the user and the product. This design approach considers that our users will want a product that can be easily integrated into their current lifestyle. This aligns with our goal for the FoodSavr in helping consumers to easily organize their food items.

9.1.1 Purpose

This appendix details the user interface design for FoodSavr. This document will provide an in-depth analysis on how the user will interact with our device, as well as all the design considerations behind FoodSavr.

9.1.2 Scope

This document focuses on the user interface design and user considerations for FoodSavr. We begin by addressing the target audience and an analysis on what designs will work for this audience. Then we discuss the design considerations behind our design, which is based on the “Seven elements of UI Interaction” by Don Norman. Which includes discoverability, feedback, conceptual models, affordances, signifiers, constraints.

Engineering standards are also addressed in this document as they are used as a guideline towards safe and intuitive operation for FoodSavr. Furthermore, usability testing plans are outlined. As analytical and empirical testing plan will help us gain more insight into our product.

9.2 User Analysis

Our target audience for FoodSavr is everyday consumers, ideally consumers who are interested in smart home technology, however our product is designed for all consumers. The FoodSavr is designed to have minimal user product interaction to suite a large demographic of consumers. The FoodSavr scanning system is automatically activated when the user places an item on the base of the system. Once scanning is complete LED lights are used to alert the user. The item scanned is then automatically uploaded to the user profile with appropriate information. This process is designed to encourage users to use our product, as it can be integrated into their current food storing process with minimal efforts. This process is similar to the self-checkout lane in large supermarkets, however instead of screen user interaction we have an LED indicator. Users who are comfortable with the self-checkout lane should be able to use our physical product comfortably.

Additional to our physical product, all information of the food items scanned are stored on a web-based user profile for our prototype product. Users who are familiar with using computers and smart-phones will be able to access the webpage comfortably. The webpage design will serve as a virtual inventory list for our users. A quick browse through will allow our users to see the food items they have in their pantry

or fridge. Users who are unfamiliar with accessing webpages will not be able to use our product to its full potential.

9.3 Technical Analysis

The following section will describe the considerations made with respect to Don Norman's "Seven elements of UI Interaction" when designing this product.

9.3.1 Discoverability/Visibility

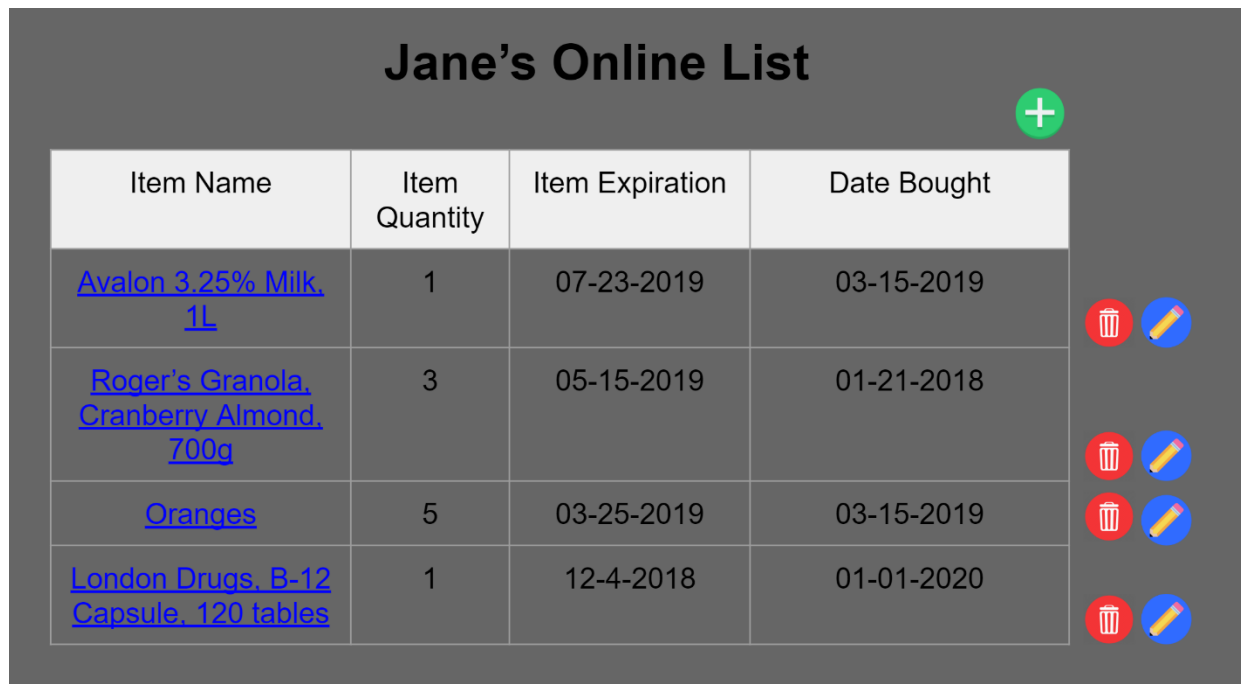
The more visible a function is the more likely to know what to do next.


On the physical machine, there will be an icon right in the middle of the device. This signifies the place where the user must put the product down, so that it can be scanned. Because the device is designed to be automated, not much more action from the user is needed (other than clear away from the robotic arm) during the scanning process.

For the software UI, it is pertinent that the user has easy access to important information as well as quick access to frequently repeated tasks:

For our users the most important pieces of information needed is: *Item name, quantity and expiration*. The most repeated tasks will be: *Adding Items, deleting Items, editing items*.

These pieces of information and actions can be accessed and done on the main page of the Web UI:



Jane's Online List 









Item Name	Item Quantity	Item Expiration	Date Bought	
Avalon 3.25% Milk, 1L	1	07-23-2019	03-15-2019	 
Roger's Granola, Cranberry Almond, 700g	3	05-15-2019	01-21-2018	 
Oranges	5	03-25-2019	03-15-2019	 
London Drugs, B-12 Capsule, 120 tables	1	12-4-2018	01-01-2020	 

Figure 5.4.1: UI Mock Up

The user will instantly see a table with columns that detail the product name, quantity and expiration date. They can edit the displayed field on this main page by simply clicking on the

field that needed editing. The user can also delete the item by clicking the “X” on the right side of each product row.

At the top of this page, there is an Add Item button.

9.3.2 Feedback

Signalling the user about what action has been done and what has been accomplished.

For our physical product, it is crucial that the user knows when it is appropriate to put down an item to be scanned, when the scanning process is happening, and then pick up an item that has been scanned. To achieve this, Savr Robotics agree that simplicity is the key.

Right on the side of the base, there will be 2 LED lights- red and green.

When idle and ready to scan, the red light will be off, and the green light will be alternating on and off. When scanning, the red light will be solid on while the green light will be off, and when just finished scanning the red light will turn off, and the green light will be solid for 2 seconds before starting to alternate on and off similar to the idle state.

State	Lights
Idle/Ready for next item	Green – blinking Red – off
Scanning	Green – Off Red – Solid
Finished Scan	Green – solid for 2 seconds Red – off

Table 5.4.1: LED States

On the UI side, feedback is also very direct. Whenever you add, remove or edit an item, the user will see that change happen instantly. The new item will be added to the list, the old item will be deleted and any changes to field will instantly be seen.

9.3.3 Mappings

The relationship between the controls and their effects

For the physical device, there aren't any controls because the device strives to automate the procedure.

On the other hand, the web UI has several buttons and controls, they mostly pertain to the maintenance of the user's list. There will be a delete “X” icon which will delete the entry the

icon is aligned with. There is the edit icon which would resemble a pen. There is also the add an entry “+” icon. These buttons have a clear relationship with their individual effects.

9.3.4 Consistency

Interfaces use similar operations and user similar elements.

For the physical component, the action of placing an item on the base, and waiting for the arm to scan makes the process consistent and familiar.

For the web UI, all the products are presented on the same table, so the same fields for all items are displayed at once.

To add an item through the UI, the user will always use the add item button. To remove an item, they will have to use the items “X” delete icon. To edit a field of an item, they will simply have to click on the pertinent field. Because FoodSavr revolves around tracking of products, it is the product’s intent to store and display item information in a consistent manner for user’s ease.

9.3.5 Affordances

Device’s attributes that allow the user to know how to use it

The device will be structured to have a large base which invites the users to place their item on the it. With the lack of buttons on the device itself, this design choice suggests that the scanning process is largely automated.

The Web UI is doing to display as a large inventory list. Each row contains information for a single item. The delete and edit icons align with each row and will only effect the item in that row.

9.3.6 Signifiers

When affordance is not enough to guide the user’s interaction with the product.

On the base itself, there will be an icon that serves two purposes. First, it signifies the center of the device which is the ideal place to set the user’s item. Second, instruct the user to put down the item onto the base for scanning.

On the web UI, each of the columns will be clearly labeled – *Name, Quantity, Expiration Date, etc* to aid the user in navigating their inventory list. The list can also be sorted in multiple ways depending on the user's preference. All buttons will either be labeled properly or distinctly symbolize what their purpose is without any space for confusion.

9.3.7 Constraints

Restrictions on the interactions between the user and the device.

With the lack of any input buttons on the physical device, the user is very constrained in interacting with it, this goes inline with the automation and hassle-free aspect of FoodSavr.

On the Web interface, there is a freedom in adding any item you want manually, but you still have the same fields you can fill out- *Name, Quantity etc.* Just like any regular list, there is a need to be able to edit the list to suit your needs, if you don't think the expiry date is important to you, you rather prefer date bought, these fields can be exchange., the UI is flexible, but makes sure that everything is still presented in a table, in an orderly manner.

9.4 Engineering Standard

To make sure the product is well construct with the camera interact system, we build and test our FoodSavr following Engineering Standards list below:

IEC TR 61997:2001 Guidelines for the user interface in multimedia equipment for general purpose use [6]

ISO 9241-161-2016 Ergonomics of human-system interaction – Part 161: Guidance on visual user-interface elements [7]

To make our product the best on serving the customer with fast, safe and convenient user interact experience, we accommodate the following Engineering Standard:

IEEE 1621-2004 IEEE Standard for User Interface Elements in Power Control of Electronic Devices Employed in Office/Consumer Environments [8]

To test and verify both the mechanical and software system of FoodSavr, we follow the Engineering standard below:

IEEE 1012-2012 IEEE Standard for System and Software Verification and Validation [9]

9.5 Analytical Usability Testing

This section outlines the analytical usability testing taken by the designers. The tests performed by our team will determine if the UI is intuitive and functional for our intended users. We separated the testing of our product into two categories, physical device and webpage.

The physical device for the FoodSavr is designed to have minimal user-product interaction, therefore it only contains the following three user interaction components.

- Power on/off switch
- Food item placement
- LED Indicators

When powered on the device LEDs will also be turned on to alert the user. The placement of the food item is marked with “+” at the base of device.

As seen in Table 5.4.1, the device LEDs tell the user when the system is ready to accept another item, scanning an item, and when the scan is finished. The colors of the LEDs were chosen to be green and red purposefully as green is universally signified as “ready” or “okay to proceed” whereas red would let the user think “not ready” or “don’t go”. We will be testing the intuitiveness of the chosen indicators throughout the development cycle, we will be tweaking these parameters to improve the easy to use factor of our design.

The webpage user interface is designed to be a virtual inventory list. The user interactions are to click buttons such as “add” and “delete” items. These buttons were chosen to be green “+” for adding an item and red “X” for deleting. In general, users will not need to add multiple items unless they purchased many food items without labels. For the web UI testing we will need to test for reusability such as accidentally deleting items or needing to delete multiple items. Through our development process we will need to continuously improve this aspect with user feedback, as our team only serve as a small demographic.

9.6 Empirical Usability Testing

Even though the project is still in the proof of concept stage of development, the team was able to observe a few key details about the device and its interface. These observations will be taken from team members, other students at SFU, and family of team members. Many of these observations will be based on user’s technical background and previous experience with other similar systems.

9.6.1 Indicator LEDs

Tell the user to approach the device with an item in hand that they are ready to scan and observe their interaction during different states of the device (Ready to Scan, Scanning, Finished Scan).

- During the idle state, did the user understand that the blinking green LED meant it was ready to take an item or did they expect it not to blink?
- During the scanning state, did the user attempt to remove the item from the device?
- When the device finished scanning, how long did the user take to realize it was finished?

9.6.2 Item Placement

Tell the user to place an item onto the device platform, wait for the scanning procedure to be finished, then remove the item.

- Did the user put the item in the center of the platform?
- Did the user “carelessly” put the item on the platform?
- Did the item slide off the platform?
- Did the user have any trouble removing the item from the device?

9.6.3 Web Page

Tell the user to access the web page URL and ask them to edit the quantity field for an item that they just scanned in. Then tell them to delete the item and add a new item. Observe their interaction with the inventory table.

- Did the user access the web page with a laptop or their smartphone?
- How long did it take the user to edit the quantity field? Did they intuitively know to tap on the field?
- Did the user know which button to use to delete an item?
- Did the user know which button to use to add an item?

Once these tests have been performed by at least 10 users, the team will gather the results and seek to improve any deficiencies in the interface and ensure the safety of the user.

9.7 Conclusion

The UI appendix outlines the aspects of the user interface design by analyzing the knowledge a typical user would need to efficiently operate the system. It also covers the testing methods used by the designers in developing the best user relatable interface. The current state of the UI design focusses on the interface that interact with user including the LEDs that indicate the status of the scanning process and the Web UI that record the food data. In the next four months the rest of the user interface will developed for the prototype to be aesthetically pleasant and easy to use by end consumers while meeting all the functional requirements.