Phase 2 Report: Project Design

Ultra Smart Refrigerator

Team #25

A Report

Presented to

The Department of Electrical & Computer Engineering

Concordia University

In Partial Fulfillment

of the Requirements

of ELEC/COEN 490

by

|  |  |
| --- | --- |
| Georgiy Araslanov | ID: 40002321 |
| Gneykou Kengne Yvann Monny | ID: 40015234 |
| Nicholas Pollender | ID: 27398050 |

Project Supervisor

Sébastien Le Beux

Concordia University

November 22, 2019

*“We certify that this is our original work.”*

Abstract

The phase 2 report details the overall design of the project. The revised design specifications and requirements as well as the alternative analysis narrowed down the selection process of the various devices, sensors and services to be used for the project. This report presents a design of the selected components with flow charts, schematics, models that will be used to produce an implementation in phase 3 of the project.

Table of Contents

Abstracti

1. Review of Design Specifications1

2. Completed Work Review1

3. Alternatives2

3.1 Microcontroller Selection2

3.2 Sensor and Device Selection3

3.2.1a Load Cell3

3.2.1b Amplifier/Converter3

3.2.2 Camera Module4

3.2.3 Temperature Sensor4

3.2.4 Hall Effect Sensor5

3.2.5 LED5

3.2.6 Speaker5

3.3 Wireless Communication Selection5

3.4 Server Selection6

3.5 Software Selection8

3.5.1 Development Board8

3.5.2 Application8

4. Development of the Solution Selected9

4.1 Hardware Design9

4.1.1 Refrigerator9

4.1.2 Hardware Schematic10

4.1.3 Load Cell and Converter Configuration11

4.1.4 Camera Module and LED Configuration11

4.1.5 Hall Effect Sensor and Speaker Configuration12

4.1.6 Temperature Sensor Configuration12

4.1.7 Configuration of all Sub-Systems12

4.2 Server Design13

4.2.1 Database Design13

4.2.2 Database Logic14

4.3 Software Design15

4.3.1 Development Board Software15

4.3.2 Application Design16

4.3.3 Application Local Database20

4.3.4 Application User Interface20

5. Deviations21

5.1 Functional Requirements Update21

5.2 Tools Required Update22

6. Updated Schedule23

6.1 Task Breakdown23

6.2 Gantt Chart24

References25

Appendix27

A1. Team Member Contributions27

A2. Application User Interface Page Sequences30

A3. Schematics31

A4. Ethical, Lega, Social, Environmental and Economical Implications of the Project33

List of Tables

1.1 Revised Numerical Design Specifications1

3.1 Development Board Analysis2

3.2 Load Cell Analysis3

3.3 HX711 Characteristics4

3.4 Camera Analysis4

3.5 Temperature Sensor Analysis5

3.6 Connection Type Analysis6

3.7 Connection Type Weighted Objective Analysis6

3.8 Server Selection Analysis7

3.9 Local Database Analysis8

5.1 Updated Functional Requirements22

List of Figures

2.1 System Block Diagram1

3.1 Types of Load Cells3

3.2 Architectural Design of Server 7

4.1 Danby DAR125SLDD9

4.2 DAR125SLDD Dimensions9

4.3 Overall Schematic of Hardware Connections 10

4.4 Load Cell & Tray Design11

4.5 Configuration of Overall System12

4.6 E/R Diagram of the Database13

4.7 Development Board Program Flow Chart16

4.8 Inventory Management, Interior View and Temperature Example Sequence Diagram18

4.9 Recipe Suggestion and Shopping List Example Sequence Diagram19

4.10 Overall Application Data Storage Design20

4.11 UI Design A21

4.12 UI Design B21

6.1 Updated Gantt Chart24

1. Review of Design Specifications

The projects design specifications were reviewed and re-assessed to accommodate the oversights of phase 1. The revised specifications provide a clarification of values as well as additional specifications that were not original considered in the preliminary design concept.

Normal conditions assume the hardware devices are operating in an environment in which no external source is acting upon it. For example, temperature should not fall below 0⁰C, and the refrigerator should remain dry and undamaged.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ID | Requirement/Parameter | Test Conditions | Values | | | Units |
| Min | Typ | Max |
| 1 | Refrigerator Dimensions (HxWxD) | - | 20x20x18 | 65x28x26 | 70x36x32 | in |
| 2 | Maximum Weight on Load Cells | Normal | 0 | 2 | 200 | kg |
| 3 | Converter/Amplifier Output Rate | Normal | 10 | - | 80 | SPS |
| 4 | Operating Temperature | Normal | 2 | 4 | 8 | ⁰C |
| 5 | Wi-Fi Connection Range | Normal | 0 | 15 | 300 | ft |
| 6 | Image Quality | Normal | - | 720 | 1080 | px |
| 7 | MCU/Board Processor Frequency | Normal | 16 | 64 | 600 | MHz |
| 8 | MCU Operating Voltage | Normal | 2.6 | 5 | 5.5 | V |
| 9 | Refrigerator Operating Voltage | Normal | - | 120 | - | V |

Table 1.1: Revised Numerical Design Specifications

2. Completed Work Review

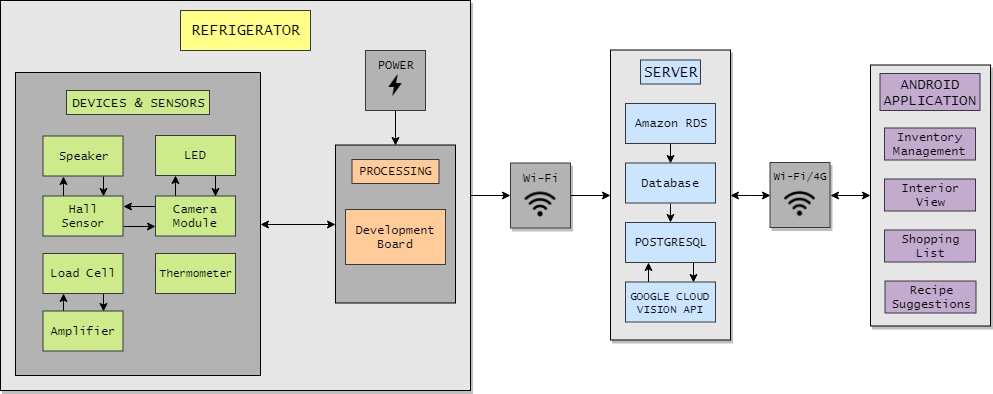
The design of the project for phase 2 has been completed as of the submission of this report. The updated Gantt chart and task breakdown can be found in section 6. Furthermore, the deviations taken from the original proposal of phase 1 can also be found in section 5. The system block diagram below describes the basic functionality of the project that will be further discussed in section 4.

Figure 2.1: System Block Diagram

3. Alternatives

Various components of the project were split into several possible solutions. Each component is analyzed with a form of alternative analysis, followed by a justification as to why the component was chose out of the alternatives listed.

3.1 Microcontroller Selection

The microcontroller is a crucial part of this project. It is important to understand the requirements of the system and to be able to choose the right microcontroller based on those requirements.

The system will be required to simultaneously receive inputs from several sensors and devices, among them are the load cells, thermostat, camera module, hall effect sensors, as well as manage some logic for toggling LED’s and a sound output to a speaker. It is important that the microcontroller selected has enough GPIO pins to ensure that every sensor and device can be connected. Wireless connectivity also plays a role in the selection process, the ability to connect to Wi-Fi natively is a welcome advantage.

In the case of this project, it was determined that a development board would be the best option as they are pre-built with many features, as well as eliminating the need of designing one from scratch, this will also be a major advantage since a large portion of this project will be software based. Development boards are also a much better choice for developing prototypes.

The following tables compare development boards that prove useful for this project.

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Raspberry Pi 3 B+ | Arduino Mega | LAUNCHXL-F28379D |
| Clock Speed | Up to 1.4 GHz | 16 MHz | 200 MHz |
| Wireless | Wi-Fi & Bluetooth | None | None |
| GPIO | 40 Pins | 54 Pins | 80 Pins |
| RAM | 1 GB | 8 KB | 100 KB |
| Storage | Up to 32 GB | 256 KB | 1024 KB |
| Operating Voltage | 5 V | 5 V | 5 V |
| Notes | CSI Camera Port | N/A | N/A |
| Price | $46.95 CAD | $49.95 CAD | $33.79 USD |

Table 3.1: Development Board Analysis

After analyzing the alternatives, it was determined that the Raspberry Pi 3 B+ would be the best option as it is a powerful board that comes equipped with built in Wi-Fi, has many pins as well as a dedicated port for a camera, more cameras can also be connected via the USB ports. The storage capacity of the board will also be useful since images will be stored in memory, an 8-megapixel camera can capture JPEG images that occupy 2.4 MB of space typically, which already exceeds the built-in memory of the other two boards. It is also relatively inexpensive, and almost breaks even with the other two alternatives if including the price of an external Wi-Fi module. The board also has Bluetooth connectivity which can serve as an alternative method for connected the board to a Wi-Fi network without having to hard code the credentials. Although it is more complex than the aforementioned alternatives, the Raspberry Pi still provides a lot of flexibility thanks to the abundance libraries and resources available for the board. It will be able to satisfy all the project requirements and will also have room to improve and add more features in the future.

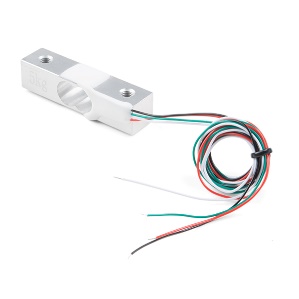
3.2 Sensor and Device Selection

The sensors and devices used for inputs and outputs play a large role in the design of the project. Some of the sensors are among the most important aspects of the project since they drive the main functionality of the system.

3.2.1a Load Cell

The load cells are the main sensor to drive this project. They will be responsible for measuring the weight of the items in each partition. A variety of load cells with different properties are available on the market. Three types of load cells would fit the design of the project, these being the double-ended beam, the S-Type, and the square-shaped body load sensors. The following table compares the three types of load cells.

|  |  |  |  |
| --- | --- | --- | --- |
| Model | Square-shaped body | Double-ended beam | S-Type |
| Space Occupied | 0.25 cm (Vertically) | 1.27 cm (Vertically) | 6.4 cm (Vertically) |
| Max Load | 50 kg | 5 – 10 kg | 200 kg |
| Accuracy | ±0.03% | ±0.05% | ±0.03% |
| Signal Type | Analog | Analog | Analog |
| Notes | Includes ADC | N/A | N/A |
| Price | $12.99 CAD for 4 | $8.95 USD ea. | $59.95 USD ea. |

Table 3.2: Load Cell Analysis

(a) Square-shaped (b) Double-ended Beam (c) S-Type

Figure 3.1: Types of Load Cells

Any one of these load cells would work with the project, in the end it came down to the space they occupied. Both the double-ended beam and the S-Type would severely reduce the vertical space in the refrigerator. Therefore, it was determined that the square-shaped body load cell would be the best option for the project. The maximum capacity is high, a kit of 4 is inexpensive and even comes with the HX711 module which is a major advantage.

3.2.1b Amplifier/Converter

It should be noted that the load cells report such low changes that an amplifier will be necessary to obtain a value on the board. Thankfully the load cell kit chosen comes paired with the HX711 module. The module works as both an amplifier and an analog to digital converter which is ideal and makes the selection process for the amplifier/converter simple. The table below lists some characteristics of the module.

|  |  |
| --- | --- |
| Model | HX711 |
| Input Signal | Analog |
| Output Signal | Digital |
| Output Rate | 10SPS or 80SPS |
| Interface | Serial |
| Voltage | 2.6 to 5.5 V |

Table 3.3: HX711 Characteristics

3.2.2 Camera Module

The camera module has the simple functionality of taking a photo of the interior of the fridge. Different cameras are compatible with the Raspberry Pi, including a camera specifically designed to be used with the CSI camera port on the board itself. The cameras can be connected to three ways, the CSI port as mentioned, the USB ports, and the GPIO pins which are very lacking. The USB cameras can be generic webcams used for computers. The following table compares two of these connections.

|  |  |  |
| --- | --- | --- |
| Model | Raspberry Pi Camera Module V2 | Logitech C270 |
| Sensor | 8-megapixel | 3-megapixel |
| Low-Light | Yes | No |
| Video | 1080p30 & 720p60 | 720p30 |
| Connection | CSI Camera Port | USB |
| Price | $32.95 CAD | $29.99 CAD |

Table 3.4: Camera Analysis

It was determined that the Raspberry Pi’s camera module V2, which operates using the Sony IMX 219 PQ CMOS image sensor, is the better option as it has great specifications and can even operate in low-light environments. Unfortunately, the downside of using the CSI camera is that only one can be connected to a Raspberry Pi. This eliminates the ability to have a set of identical cameras for individual sections of the refrigerator. However, the USB ports can also be used to other camera module such as webcams to the board along side the CSI camera, enabling the possibility of a multi-camera setup.

3.2.3 Temperature Sensor

The temperature sensor has a singular purpose, to measure the interior temperature of the refrigerator. It is important that the sensor chosen can provide accurate readings since the user will want to ensure that the refrigerator is operating at the right temperature. Temperature sensors come in two variants, analog sensors and digital sensors. The digital sensors will be the better option overall since unlike the analog sensors, the calculations are performed by the sensor itself, and the output is an actual temperature value. The following table compares two different digital temperature sensors.

|  |  |  |
| --- | --- | --- |
| Model | DS18B20 | DHT22 |
| Max Temp | 125**°C** | 80**°C** |
| Min Temp | -55**°C** | -40**°C** |
| Accuracy | ±0.5**°C** | ±0.5**°C** |
| Voltage | 3 to 5.5 V | 3 to 5 V |
| Signal Type | Digital | Digital |
| Price | $3.95 USD | $9.95 USD |

Table 3.5: Temperature Sensor Analysis

It was determined that the DS18B20 temperature sensor would be ideal for the project. It is very affordable, has large temperature ranges that fit within typical ranges of the common refrigerator, and most importantly has a digital signal.

3.2.4 Hall Effect Sensor

The hall effect sensor selection is not very crucial. Most of the sensors function in a similar manner, and do not cost very much. Hall effect sensors give out an output high (5V) whenever it detects the presence of a magnetic field and gives out an output low when the magnetic field is removed. An alternative type of sensor called a latching hall effect sensor will give out an output high when the north pole of a magnet is brought close to it and only give out an output low once the south pole of the magnet is brought close to it. For the purpose of this project, a non-latching hall effect sensor would be the best solution since the only function of the sensor will be to determine whether the door is closed (close to the magnet, output high) or if the door is opened (away from the magnet, output low). The model selected for use in the project is the AH1815 hall sensor as it is non-latching and very inexpensive reaching prices as low as $1 per sensor.

3.2.5 LED

Any LED that can be toggled on and off will suffice for the implementation of the project. Some LED’s can consume more power than any of the previously mentioned sensors and devices and may require an external power source to function properly and prevent any damage to the development board. A short LED strip, such as the WS2812b, will suffice for lighting up the area that the camera will be capturing an image of. As a precautionary measure, an additional 5V adaptor will be required.

3.2.6 Speaker

Any device capable of emitting a loud sound cue will be ideal for the implementation. A generic speaker could be plugged into the auxiliary port of the development board, however a more cost-efficient device called a piezo buzzer fits the requirement just as well. A buzzer can emit different levels of sound depending on the electric potential applied to the device reaching over 90 dB in some cases. For use in the project, the PS1240 buzzer with a rated frequency of 4 kHz would meet the requirements.

3.3 Wireless Communication Selection

The board is required to send the sensor and device data to the mobile application. Hardwiring the connection is an option, however that defeats the purpose of using a mobile application. Nowadays, wireless connectivity is used much more frequently, which is available in a variety of connection types. Bluetooth, Wi-Fi and 4G are amongst the most common wireless technologies. Below is an analysis of different connection types that can be used to monitor the data that the board receives.

|  |  |  |  |
| --- | --- | --- | --- |
| Connection | Wi-Fi + 4G (Server) | Wi-Fi (Local only) | Bluetooth 4.2 |
| Range | Virtually Infinite | 150 to 300 ft | 30 ft |
| Band | N/A | 2.4 GHz | N/A |
| Data Rate | Varies | Varies | 1 Mbps |
| Price | $0 (Built-In) + Server Cost | $0 (Built-In) | $0 (Built-In) |

Table 3.6: Connection Type Analysis

The server option would appear to be a good selection, however, to help fortify the decision, a weighted objective table was created to determine which connection type is best suited for the project. The criteria are scored out of a total of 100, and each criterion can only assign a total of 1 point divided amongst the connection types.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Weight | Wi-Fi + 4G (Server) | Wi-Fi (Local only) | Bluetooth 4.2 |
| Range | 35 | 0.90 | 0.07 | 0.03 |
| Efficiency | 45 | 0.40 | 0.55 | 0.15 |
| Reliability | 20 | 0.35 | 0.50 | 0.15 |
| Score | 100 | 0.565 | 0.372 | 0.063 |

Table 3.7: Connection Type Weighted Objective Analysis

After analyzing the two tables, it was determined that the server option will be the best connection type. While the server option will be more complex to implement, the server would enable the application to connect to it via any kind of Wi-Fi connections, as well as 3G or 4G networking. This means a user would be able to access the refrigerator data from anywhere so long as they have a reliable connection to the internet.

3.4 Server Selection

A server will be used to transmit data from the development board to the application through an online database. It will be important to select a server that fits the requirements of the project and can handle large amounts of data to store the various recipes for example.

The most popular option for server-based apps are cloud hosting services, such as [Google Cloud Platform](https://cloud.google.com/) (GCP), [Amazon Web Services](https://aws.amazon.com/) (AWS), and [Microsoft Azure](https://azure.microsoft.com/). Other platforms such as Firebase and Heroku exist which provide similar services to host applications. Firebase for example, provides a real-time database with the objective of seamless synchronization. It however cannot handle complex (relational) queries [1]. Heroku on the other hand, provides relational database functionality but does not offer persistent local storage (this means that files can be written to the disk, but those files will not persist after the application is restarted) and some of its servers fail to remain *alive* after 30 minutes of inactivity – which will take approximately 20 seconds to respond at first once it wakes up [2].

Before moving forward, it is important to understand the two major query languages used in relational databases, namely MySQL and PostgreSQL [3]. Although they may seem very similar in regard to what both can do/what they have to offer, there exist certain differences between them. Despite these differences, PostgreSQL was chosen for this project because not only is it “the most advanced open-source relational database in the world”, it also an object-relational database, unlike MySQL which is a purely relational database. This means that, PostgreSQL supports certain properties that are not present in MySQL such as table inheritance and function overloading, which can be important for the application. Moreover, PostgreSQL adheres more closely to SQL standards, handles concurrency better than MySQL, and is known for protecting data integrity at the transaction level; which makes it less vulnerable to data corruption [4].

The table below describes the features of the three aforementioned most popular cloud hosting services (for PostgreSQL, in the case of this project).

|  |  |  |
| --- | --- | --- |
| Google Cloud Platform | Amazon RDS | Microsoft Azure |
| -Scalability  -High performance  -Integrated  -Fully managed  -Security  -Standard APIs  -Availability protection | -Easy to set up  -Push button scaling  -Automated backup and recovery  -Resizable hardware capacity and storage management  -Cost efficient  -High availability | -Fully managed community  -Languages and frameworks of customers choice  -Built-in high availability  -Scale within seconds  -Automated backups and point-in-time restore  -Unparalleled security and reach |

Table 3.8: Server Selection Analysis [5]

A close up of a logo

Description automatically generatedGoogle Cloud Platform automatically provides $300 worth of credit to anyone who joins and allows a small (f1-micro) server to run at no cost, indefinitely, along with providing a variety of other free-tier usage limits. AWS (Amazon RDS) offers very similar free-tier limits to GCP, allowing one small compute instance (t2-micro) for free each month. Microsoft Azure offers $200 in free credit when upon joining, but this free credit expires after one month. They also provide a free tier on their "App Service" offering, although this free tier is more limited than the equivalent offerings from AWS and GCP. In the end, the deciding factor was the price with AWS being the winner. Since the team consists of students, GitHub offers an amazing pack of free amenities including a $110 in AWS credits and much more.

Wi-Fi

Wi-Fi/4G

Figure 3.2: Architectural Design of Server

3.5 Software Selection

3.5.1 Development Board

The Raspberry Pi GPIO pins can be controlled with many different languages. It supports common languages such as Java, C++ and Python. Among the languages listed, Python is the most widely used language in RPI development. As a result, many libraries exist for the various components that will be used for this project, this will enhance the compatibility of the devices and sensors as well as the server drastically. It was determined that Python would be the best choice for developing the board software. Furthermore, the Python script can be automatically run upon start up of the board, enabling quick and easy reboots while debugging.

3.5.2 Application

The application will be the final destination for the data obtained by the board. As previously mentioned in phase 1, the application will be developed for Android devices initially, perhaps with room for iOS development after phase 4.

More importantly, the application will feature a local database to store information on the device. There are two main methods for storing data in Android, shared preferences and SQLite. Both methods save the stored data even after the application, meaning they will persist in the database until the user deletes them. Below is a table that outlines the key differences between the two storage types.

|  |  |
| --- | --- |
| Shared Preferences | SQLite |
| -Supports small amount of data  -Data is saved under a key  -Must have the key to read the data  -Need to define a key for every single data  -Variables must be pre-defined  -Searching is very inefficient | -Support large amount of data  -Can be queried to obtain a subset of data  -Can search tables for data  -Dynamically update database contents  -May be slower to read data for large databases |

Table 3.9: Local Database Analysis

It was determined that SQLite would be the better option for this project. It can store large amounts of data which will be useful for bookmarking various recipes for example. It can also handle searching for specific data in the tables which will be a necessity for finding items that are low on stock and adding them to the shopping list.

4. Development of the Solution Selected

4.1 Hardware Design

4.1.1 Refrigerator

A traditional mini refrigerator will be used in the implementation of this project since a full-scale refrigerator would be impractical to store and displace. An old mini refrigerator that will be re-purposed for the development of this project. The picture below is the Danby DAR125SLDD model that will be used except for the shelves which will use 2 instead of 3.

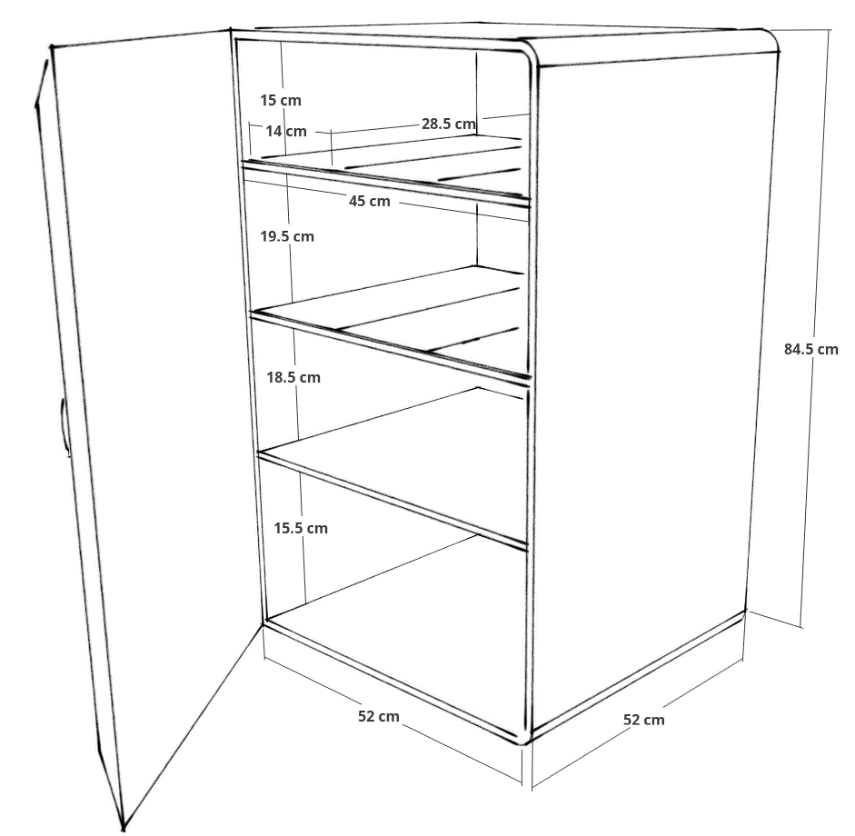
Figure 4.1: Danby DAR125SLDD

Figure 4.2: DAR125SLDD Dimensions

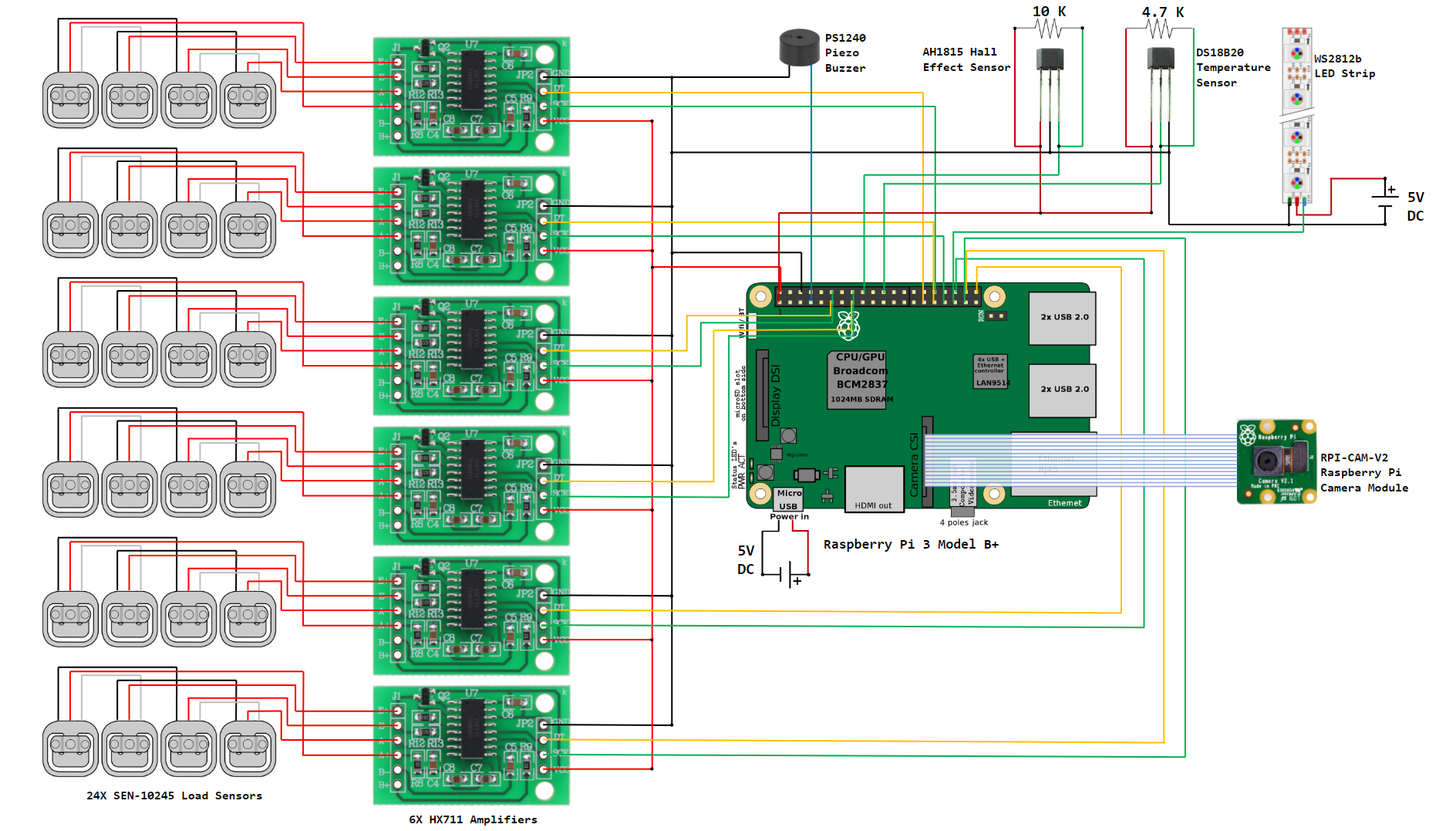
4.1.2 Hardware Schematic

Figure 4.3: Overall Schematic of Hardware Connections

Figure 4.3 represents the interconnections of all the sensors and devices that will be connected to the board. Note that the lengths of the connections depicted will not be the actual lengths in practice.

4.1.3 Load Cell and Converter Configuration

Each of the 2 shelves will be split into 3 partitions, for a total of 6 partitions. Each partition will be equipped with a rectangular tray approximately one third of the width of the shelf. Each tray will have four load cells beneath it located at each corner. The weight of the items placed on the tray are measured by the compression between the tray and the shelf. The weight of the tray itself will be omitted in the logic of the board, much like the tare functions works on a traditional scale.

The load cells output an analog signal and the Raspberry Pi 3 B+ does not have any analog inputs, therefore an analog to digital converter will be required so that the board can read the output. The HX711 module can connect to 4 load cell configuration and output a digital signal that the board will be able to read.

The style of the load cell configurations bears similarities to the common scale found in households as depicted below.



Figure 4.4: Load Cell & Tray Design

4.1.4 Camera Module and LED Configuration

The camera module will be placed in an ideal location to capture as much of the lowest shelf as possible. The proposed location is one of the back corners aiming towards the middle of the shelf. Furthermore, the LED’s will be setup near the camera so that the area the camera will be capturing will be illuminated when the picture is taken. It is important to ensure that the LED is not directly in the line of sight of the camera lens since the glare will ruin the image. The LED will also take a separate 5V input as shown in Figure 4.3. This is because the board will not be able to power the entire strip in most scenarios and may damage the board itself if attempting to power the whole strip from the board [21].

4.1.5 Hall Effect Sensor and Speaker Configuration

The hall effect sensors will be placed near the opening of the door and will require 10 KΩ pullup resistor to function properly. More than one sensor will be used to add an extra layer of certainty on the open/close status of the door. The speaker will also be placed near the opening of the door to maximize the effectiveness of the noise, since if the speaker was deeper in the refrigerator, the sound may be muffled.

4.1.6 Temperature Sensor Configuration

The temperature sensor needs to be placed in an ideal location to be able to record the temperature at all times without any obstructions. The location chosen is the back top right corner since there will not be many opportunities for obstruction from the items stored in that corner. Furthermore, the sensor requires a 4.7 KΩ pullup resistor to function properly similar to the hall effect sensors.

4.1.7 Configuration of all Sub-Systems

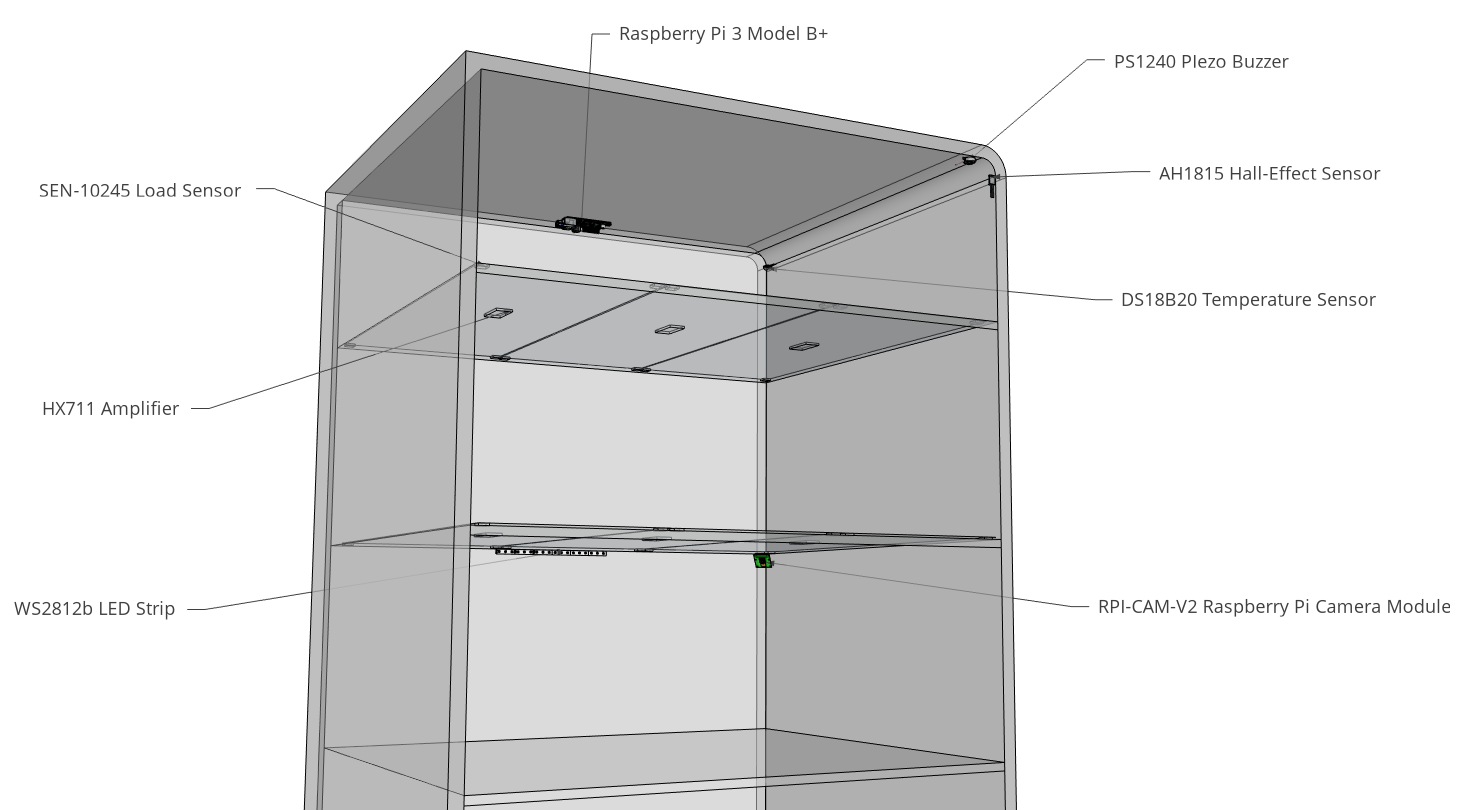
The development board will be placed on the ceiling of the refrigerator. This way, it will be out of the way and minimize the overall length and clutter of the cables. Below is a figure that displays the location of each component, the cables, although not shown, will be concealed as much as possible by being installed on the walls of the refrigerator.

Figure 4.5: Configuration of Overall System

4.2 Server Design

A close up of a map

Description automatically generated4.2.1 Database Design

Figure 4.6: E/R Diagram of the Database

4.2.2 Database Logic

Below are the characteristics of the E/R Diagram mentioned in section 4.2.1:

* Users will be identified with their email addresses, together with other information related to them such as name, birthday, ID of the device(system) and password to the application.
* Reviews for this product will be identified with an ID, together with other information related to them such as the email of the reviewer, ID of the device (system) and a description of the review.
* Devices (system) will be identified with an ID, together with other information related to them such as Wi-Fi Password, temperature of the device (system) and Wi-Fi information used in connecting them to the server.
* Cameras will be identified with an ID, together with other information related to them such as the image captured, processed images and things found (obtained with the use of Google Vision API on the images).
* An Inventory will be identified with partition name/ID, together with other information related to them such as the name of the item in that partition and the quantity of the item in question.
* A Shopping List will be identified using the name of the item wished to be bought (either for inventory replenishment or for a particular recipe), together other information related to it such as with a tag and its quantity.
* A Recipe will identified with an ID, together with other information related to it such as the name of the recipe, the description, ingredients required for the recipe, its source, attachments (may be thumbnails), under which food categories it falls, its preparation time, number of servings, nutritional values and quantity of ingredients to be used.
* A food category will be identified with the use of an ID and the title of the category itself.

This database has been designed taking into consideration the notion of scalability.

* Many recipes can fall under the same food category.
* The inventory can lead to multiple shopping items (in the list) and could be used to search different recipes.
* The recipe can also lead into multiple shopping items (in the list) – in case the user is missing some ingredients.
* The camera(s) is/are connected to a device.
* Many users can have many reviews, but a device (system) can be either be for one user or shared among users (say members of the same family or living in the same house).
* Since an inventory is associated with a particular device (system), and the user(s) connected to a device, the entire application can be seen and used by multiple users (member of the same household) if need be, with proper authentication.

4.3 Software Design

4.3.1 Development Board Software

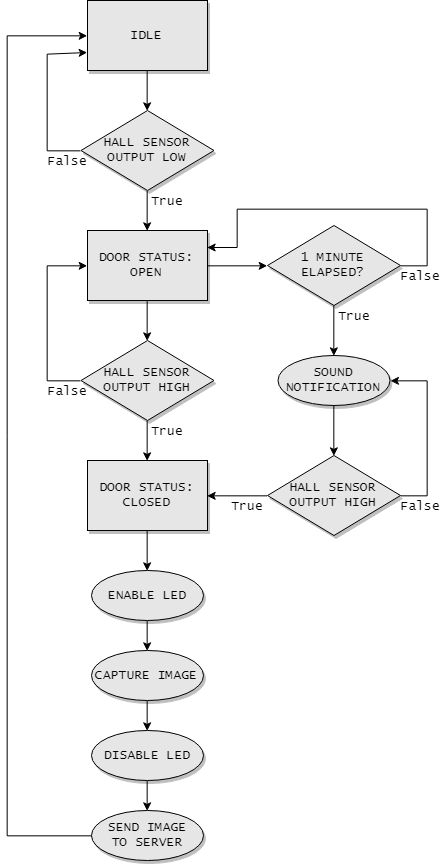
The board software will be coded in Python. It will handle basic data processing such as normalizing the tray weight for the load cell configuration as well as the operations for taking pictures with the camera module and detecting whether the door is opened or closed. To implement these features, the board will be connected to a separate computer and visually display the data captured to initially validate the functionality of each sensor and device. Once everything is in working order, the board can be connected to the database.

The board logic has 3 main functions:

**Weight Normalization** – Initially when there are no items on the load cell configuration, a weight will still be recorded by the load cells since there will be a tray resting on them, this weight will cause inaccurate values. To account for this, the weight of the tray will be recorded, it will be stored as a static variable and be subtracted from any future value the weight they record. This way, when no items are on the load cell configuration, the recorded value will be zero.

**Door Open/Close Status** – The hall effect sensors will determine whether the door is opened or closed. When the door is closed, the hall effect sensors will be next to magnets and therefore in a magnetic generating the output high. If the sensors are reporting an output high, then the door is assumed to be closed. Once the door is opened, the sensors are no longer subjected to the magnetic field and generate an output low. When the door is opened, a timer is started. If the timer surpasses 2 minutes, the speaker is enabled emitting a loud noise. Once the door is closed and the sensors are back to output high, the timer is stopped and reset.

**Image Capture** – An image will be taken of the unweighted shelf to account for any additional items in the refrigerator that aren’t being monitored on the partitioned shelves. The image capture will occur whenever the refrigerator door is closed (after being opened), the new image will replace the old image. It must be considered that the door will be closed while the image is being captured in most cases. To account for this, before the image is taken the LED strips installed on the walls will be toggled on and then toggled back off once the image is captured. In case of a failure to turn the LED strips on, the camera does function in low-light environments, but the image quality will be worse. The image is then uploaded to the server.

Figure 4.7: Development Board Program Flow Chart

4.3.2 Application Design

This project is heavily software based, so the application will be a very important factor. It is used to visually provide the user with the data obtained by the hardware and sensors as well as an interface to modify a shopping list and search for recipes based on user defined criteria. The application will be developed for Android devices using the Java programming language in the Android Studio IDE.

The application will initially ask the users to sign into their refrigerator. Next is to set up the partition information, as well as providing various personal information such as name, birthday and an email if the wishes to do so. It will have a local database using an SQLite table to store values received from the server while the device is connected to the internet. Each element of the database will be matched with the data from the server to ensure that the values are being displayed in the correct areas of the application. If a connection cannot be made, the latest values stored on the local database will still be available to the user.

The application was designed to have 6 main features:

**Authentication** – To connect to the server and gain access to the refrigerator data, the board will have a unique ID and password associated to it that will also match up with values in the server database. The application will use these credentials to login to the application. Upon a successful login, the user will gain access to the refrigerator whose unique ID was connected to. The purpose of the authentication system is to be able to run multiple instances of the same refrigerator on different mobile devices in the same household.

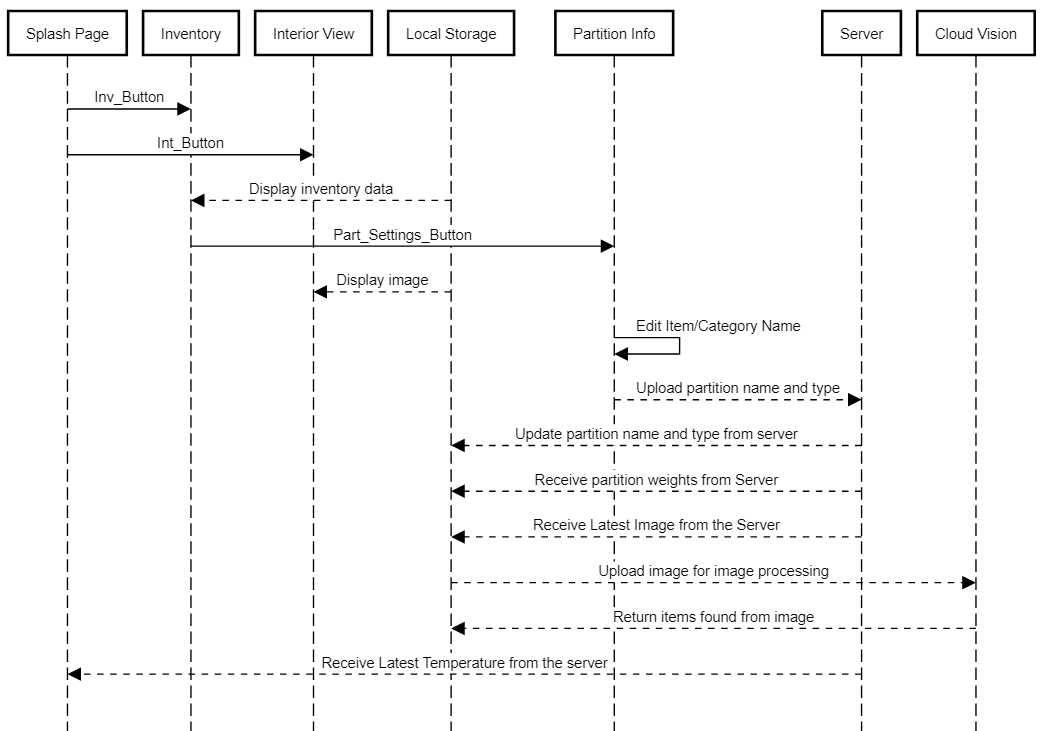
**Inventory Management** – The inventory management handles three main variables for each partition which will be stored in an SQLite table in six separate rows. Firstly, the name of the partition stored as a TEXT variable defined by the user. The second variable is an INTEGER that determines the type of item that is stored in the partition. For example, if the value is 0, then the item type is a singular food item such as eggs If the value is 1, then the item type is a category, such as dairy products. Finally, the weight recorded by each partition will be stored as a REAL variable. REAL variables function similarly to float variables in java, so the decimals can also be recorded. The data stored in this table will be displayed on the corresponding page via a ListView.

The partition names and types can be edited at any time in the settings menu if there is an internet connection. The application updates this data and the weight from the server whenever the user navigates to the inventory management page.

**Interior View** – The interior view handles the image captures by the camera module on the board. It’s sole purpose in this scenario is to identify the unweighted section of the refrigerator. As such, image will display *what* the user has, but not specifically how much they have, however the amount of food can still be estimated by the user by observing the image.

The image will be converted and stored in a lone table as a BLOB, which stands for binary large object. The image can be displayed be converting the BLOB into an ImageView. Every time the user visits the interior view page, the latest image on the server will be stored in the table. Upon receiving the image from the server, it will be sent back up to Google’s Cloud Vision API. This is an image processing service that can identify the items in an image. Part of the output is a long string of text that lists *all* the items found, this includes non-food items which will be filtered out. The items found will be stored in a separate table as a TEXT variable that will be used with the recipe suggestion feature.

**Temperature** – The temperature value will be stored in its own table as a REAL value and displayed directly on the welcome page of the application. This is a read-only value, meaning the user can only view it and cannot alter it from the application itself. The main purpose of this feature is to ensure the refrigerator is operating at a reasonable temperature. The temperature is updated whenever the user launches the application or revisits the welcome page.

Figure 4.8: Inventory Management, Interior View and Temperature Example Sequence Diagram

**Shopping List** – The shopping list handles three main variables per item in the list. An SQLite table will store each item in their own row. The name of the item is stored as a TEXT value and each item has an INTEGER associated to it to add an optional quantity for each item. Each item in the list can be added or removed at will, as well as clear all manually entered items with the press of a button. The third variable is specifically for items that will be suggested based on the weight of the partitions. If a weight falls below 10% for example, that item will be automatically added to the list with a special flag enabled as an INTEGER. If the flag value is set to 0, that means the item was added manually by the user, if the flag is set to 1, that means the item was added automatically. Suggested items on the list will be marked with a unique symbol to distinguish manually added items versus suggested items. The contents the table will be displayed via ListView.

Whenever the list is updated manually, the data is uploaded to the server so that other users connected to the same refrigerator will gain access to the listed items. The application updates the data whenever the user navigates to the shopping list page.

**Recipe Suggestion** – Upon visiting the recipe suggestion page for the first time, the user will be prompted to enter additional ingredients that are not typically found in refrigerators such as flour, various spices, vegetable oil, etc. These ingredients will be stored in an SQLite table as TEXT values. Furthermore, as mentioned previously, the items found string from the image processing will also be stored in a table as TEXT values.

The recipe suggestion feature prompts the user to enter criteria that will then be used to search for recipes on the server, this includes calorie count as an integer and a Boolean value that will determine the search type. If the Boolean value is false, this means the search will *not* use the refrigerator inventory and instead the user will provide is list of ingredients of their own choosing. If the Boolean value is true, this means the search will use the inventory strictly from the refrigerator. This search includes the items from each partition of the inventory management, the items found from the image processing, and the additional ingredients previously defined. The server will return the most accurate results pertaining to the search criteria. Each recipe provides information such as a name, a list of ingredients and a list of nutritional values.

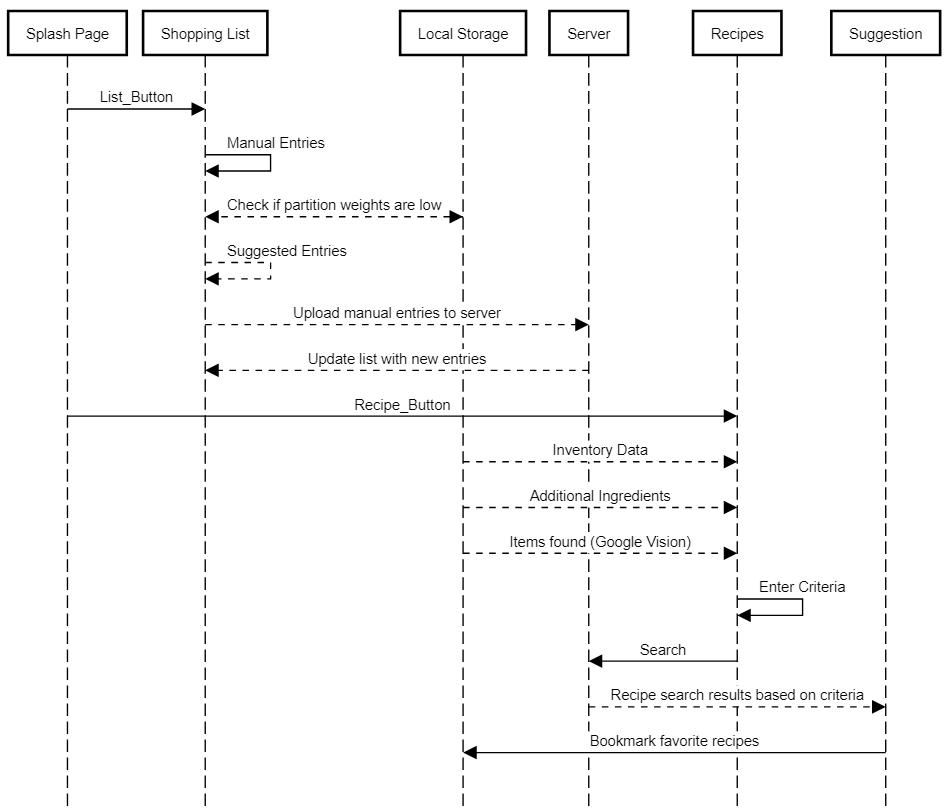
Additionally, users will be able to save their favorite recipes with the press of a button as well as add missing ingredients to the shopping list. The bookmarked recipes will be stored in an SQLite table. Each entry will store the recipes the same way they are stored on the server. The recipes can be reviewed at any time from the recipe suggestion page and will be displayed as a ListView.

Figure 4.9: Recipe Suggestion and Shopping List Example Sequence Diagram

4.3.3 Application Local Database

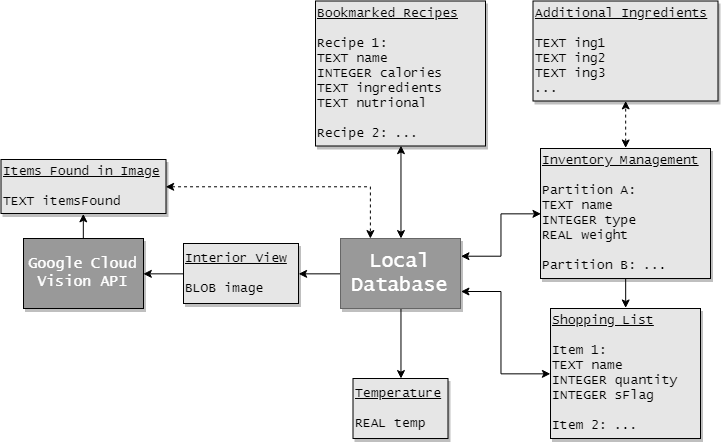
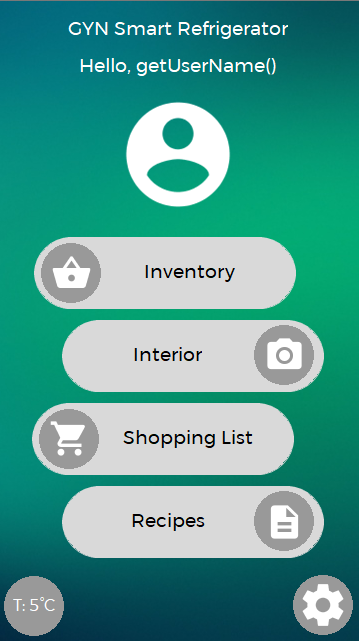
The application will use many SQLite tables to store information. These tables will be used to store data on the application so that even when the application is closed, the information is still available to the user the next time the application is launched. Several tables are used for the inventory management, interior view, temperature and shopping list features. By storing the data in SQLite tables, if the user happens to lose internet connectivity, the applications functionality will be limited, however the current values stored in the tables will still be made available but may no longer be accurate. Below is a block diagram that outlines the interconnection between application features and the data stored. 

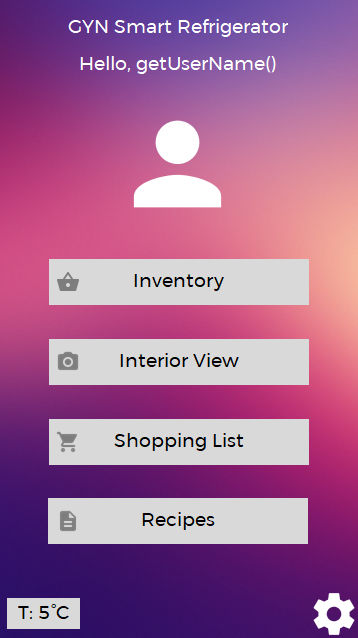
Figure 4.10: Overall Application Data Storage Design

4.3.4 Application User Interface

The application interface was designed with the JustInMind software. The application needs to be user friendly, to reach this goal it follows a minimalistic *less is more* design. Upon launching, the user is presented with a welcoming splash page that is not difficult to follow. All the buttons are big and visible with large easy to read text as well as images related to each button. Transitions between pages are simple, smooth and straightforward so the user does not get lost in the application.

For example, the inventory page displays each partition of the refrigerator, and each partition is indicated by a large letter ranging from A to F. Upon pressing the letter, an image is displayed of that partition to help the user find it in the refrigerator. The item/category name and weight are displayed in large text for each partition. The interior view page simply displays the image of the interior of the refrigerator. Below are a couple of examples of the applications user interface featuring the design explained. The remaining feature of the application follow the same minimalistic design, the additional pages and page sequences of the UI based on design A can be found in appendix A2.



Figure 4.11: UI Design A Figure 4.12: UI Design B

5. Deviations

5.1 Functional Requirements Update

Since the project designed will use a real mini refrigerator in the place of a wooden frame, mechanically opening/closing the door would be a daunting task. It was determined that the servo motor initially researched would not be powerful enough to open and close the refrigerator door, a more expensive servo would be required in that case, thus FR-4 was removed from the functional requirements. To accommodate the change, as denoted by the updated FR-5 in table 5.1, the refrigerator will now include a speaker that performs an audible cue when the door has been open for too long. This change will positively affect the budget since the speaker chosen is inexpensive. The change will not impact the schedule significantly, the task for implementing the speaker will replace the servo motor task. FR-8 was also added to highlight the local database that will be used in the application as it was not previously considered. This change will not impact the schedule significantly.

|  |  |
| --- | --- |
| ID | Description of the Functional Requirements (FR) |
| FR - 1 | The refrigerator will be able to measure the quantities of food placed within each partition inside. |
| FR - 2 | A mobile application will be able to connect wirelessly to the refrigerator, will receive and visually display the data collected by the refrigerator. |
| FR - 3 | The refrigerator will have a camera installed within its interior and the user will be able to have a view of the inside of the refrigerator through the mobile application. |
| ~~FR - 4~~ | ~~The refrigerator’s door will be automatically opened & closed through the app.~~~~1~~ |
| FR – 5 | The user will be alerted **via audible cue from a speaker** in case the refrigerator has been opened for too long. |
| FR - 6 | A secured connection/interaction should be established within the system (mobile application, microcontroller, database, user). |
| FR - 7 | The smartphone application will suggest healthy recipes, inform about nutritional values of foods and provide visual data about food usage/habits and potentially expired food. |
| FR - 8 | **The smartphone application will store recent data from the server on the device so that the information can still be accessed without an internet connection.** |

Table 5.1: Updated Functional Requirements

5.2 Tools Required Update

The required tools section of phase 1 was lacking devices that were not initially considered. It is especially important to include these tolls as they will be vital for the testing and validation of the project. Modeling software was also included for developing the user interface. Below are the additional tools that will be required:

Oscilloscope: For displaying the changes in electrical signals over time for the various sensors and devices that will be used for the project. This tool will be very useful for testing and validation.

Multimeter: For measuring the voltages, resistances, currents across the various sensors and devices of the project. Another useful tool for testing and validation.

JustInMind Prototyping Software: For modeling the user interface of the application.

6. Updated Schedule

6.1 Task Breakdown

The task breakdown was updated to accommodate the changes in the functional requirements. Additional tasks regarding the design/implementation server and software components as well as remaining sensors and devices that were initially omitted were also included.

1. Research

1.1 Research on microcontroller: Yvann – COMPLETED

1.2 Research on wireless connectivity (Bluetooth, Wi-Fi): Yvann – COMPLETED

1.3 Research on cloud server: Yvann – COMPLETED

1.4 Research on load cell: Georgiy – COMPLETED

1.5 Research on camera module & image processing: Nicholas – COMPLETED

1.6 Research on amplifier & converter: Georgiy – COMPLETED

1.7 Research on mobile application requirements: Nicholas – COMPLETED

* 1. ~~(Optional) Research on servo motors: TBA~~

~~1.9 (Optional) Research on USB charging: TBA~~

2. Prototype Design

2.1 Design of load cell configuration: Georgiy – COMPLETED

2.2 Design of amplifier & converter configuration: Georgiy – COMPLETED

2.3 Design of camera configuration: Nicholas – COMPLETED

~~2.4 (Optional) Design of mechanical door: TBA~~

* 1. **Design of speaker configuration: Georgiy** – COMPLETED

2.5 Design of hall effect sensor & LED configuration: Georgiy – COMPLETED

2.6 Design of temperature sensor configuration: Georgiy – COMPLETED

2.7 Implementation of preliminary MCU setup: Georgiy, Nicholas

3. Cloud Server

3.1 Design of database: Yvann – COMPLETED

* 1. Design of data processing: Yvann, Nicholas – COMPLETED
  2. Server implementation: Yvann, Nicholas

4. Software

4.1 Microcontroller data processing design: Georgiy, Yvann – COMPLETED

4.2 Microcontroller data processing implementation: Georgiy, Yvann

4.3 Application design: Yvann, Nicholas – COMPLETED

4.4 Application development:

4.4.1 User interface: Nicholas

4.4.2 General application features (i.e. User information): Yvann

4.4.3 Recipe suggestion algorithm: Yvann, Nicholas

4.5 Application data receival: Nicholas

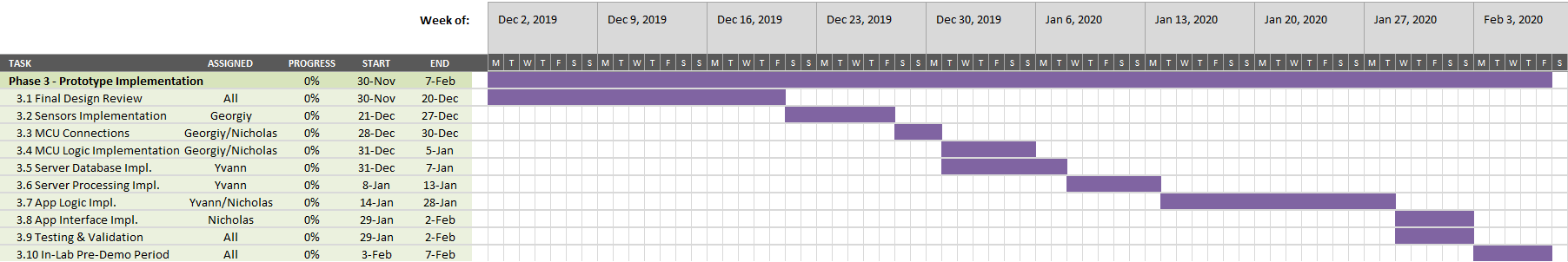
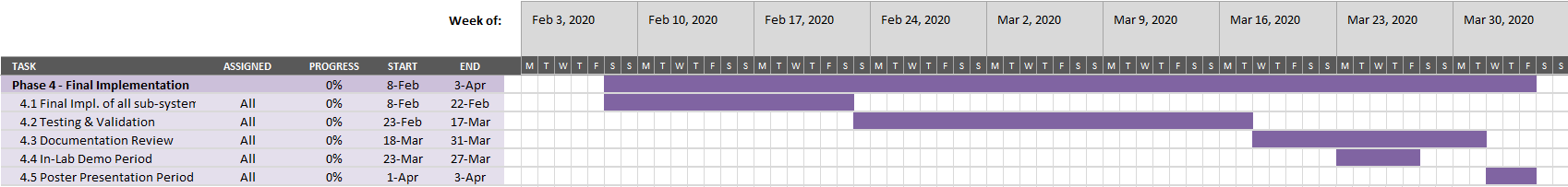
5. Implementation of Final Design

5.1 Implementation of all sub-systems: Georgiy, Yvann, Nicholas

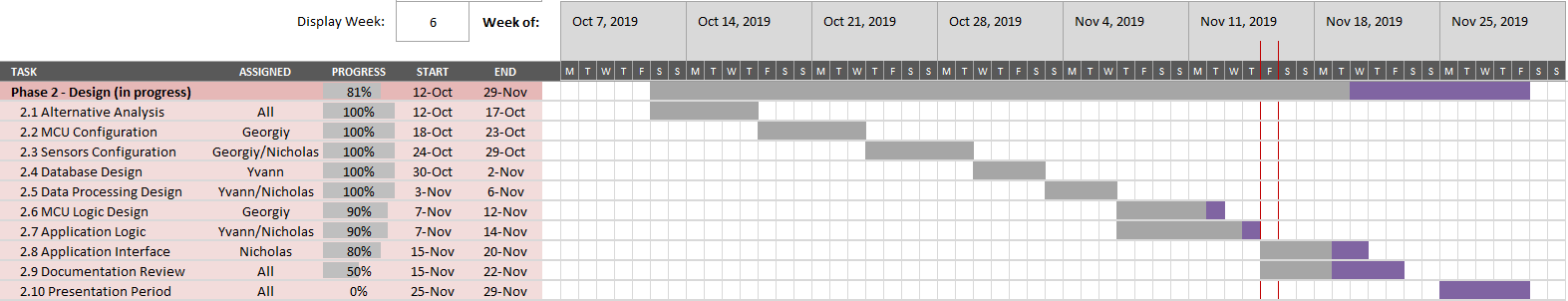
6. Testing & Validation

6.1 Testing overall system: Georgiy, Yvann, Nicholas

6.2 Validating overall system: Georgiy, Yvann, Nicholas

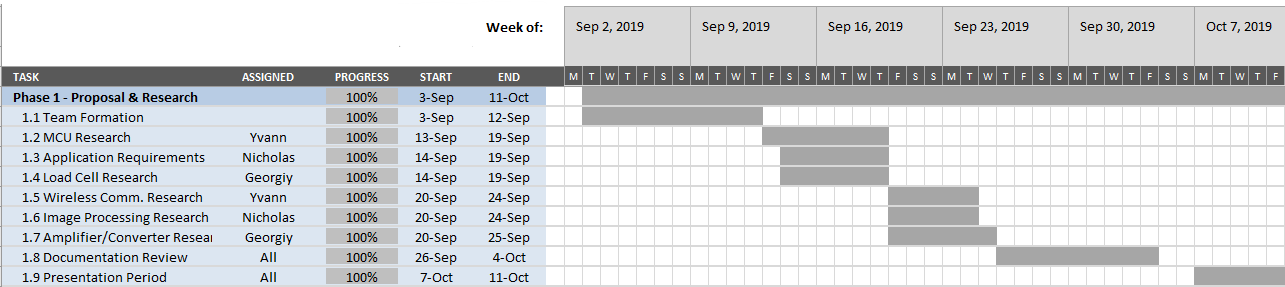


6.2 Gantt Chart as of November 15th



24

Figure 6.1 Updated Gantt Chart



References

[1] Croos, Peter de. “When You Should (and Shouldn't) Use Firebase.” Codementor, https://www.codementor.io/cultofmetatron/when-you-should-and-shouldn-t-use-firebase-f62bo3gxv.

[2] Triest, Patrick. “10 Tips To Host Your Web Apps For Free.” Break | Better, Break | Better, 8 Sept. 2017, https://blog.patricktriest.com/host-webapps-free/.

[3] “PostgreSQL vs. MySQL: Everything You Need to Know.” Hackr.io, <https://hackr.io/blog/postgresql-vs-mysql>.

[4] Hristozov, Krasimir. “MySQL vs PostgreSQL -- Choose the Right Database for Your Project.” Okta Developer, Okta Inc., 19 July 2019, https://developer.okta.com/blog/2019/07/19/mysql-vs-postgres.

[5] Severalnines. “Comparing Cloud Database Options for PostgreSQL.” Severalnines, Severalnines, 26 Aug. 2019, https://severalnines.com/database-blog/comparing-cloud-database-options-postgresql.

[6] HX711 Schematic: <https://cdn.sparkfun.com/assets/f/5/5/b/c/SparkFun_HX711_Load_Cell.pdf>

[7] RBPi 3 B+ Schematic: <https://www.raspberrypi.org/documentation/hardware/raspberrypi/schematics/rpi_SCH_3bplus_1p0_reduced.pdf>

[8] Kumar, Deepak. “Working With Shared Preferences and SQlite Android.” Medium, Medium, 3 July 2017, <https://medium.com/@dipakkr/working-with-shared-preferences-and-sqlite-android-512abb1b2745>.

[9] Studio, Tubik. “15 Hot Trends in UI Design for Web and Mobile in 2018.” Medium, UX Planet, 31 Aug. 2018, https://uxplanet.org/15-hot-trends-in-ui-design-for-web-and-mobile-in-2018-eff86df6d868.

[10] Justo, Patrick Di. “Raspberry Pi or Arduino? One Simple Rule to Choose the Right Board: Make:” Make, 5 Jan. 2017, https://makezine.com/2015/12/04/admittedly-simplistic-guide-raspberry-pi-vs-arduino/.

[11] “Different Types of Load Cells: Load Cell Manufacturer.” HBM, 19 June 2019, <https://www.hbm.com/en/7231/the-different-types-of-load-cells-supplied-by-hbm/>.

[12] Hildenbrand, Jerry. “Add an Eye to Your Raspberry Pi with These Awesome Camera Add-Ons.” Android Central, Android Central, 25 Sept. 2019, <https://www.androidcentral.com/best-raspberry-pi-camera>.

[13] “Raspberry Pi DS18B20 Temperature Sensor Tutorial.” Circuit Basics, 27 Aug. 2018, <http://www.circuitbasics.com/raspberry-pi-ds18b20-temperature-sensor-tutorial/>.

[14] “Bluetooth vs Wi-Fi.” Diffen, <https://www.diffen.com/difference/Bluetooth_vs_Wifi>.

[15] Oulton, Randal. “Refrigerator Thermometers.” CooksInfo, 10 May 2018, <https://www.cooksinfo.com/refrigerator-thermometers>.

[16] Kumar, Deepak. “Working With Shared Preferences and SQlite Android.” Medium, Medium, 3 July 2017, <https://medium.com/@dipakkr/working-with-shared-preferences-and-sqlite-android-512abb1b2745>.

[17] Koch, Wendy. “Why Your Fridge Pollutes and How It's Changing.” National Geographic, 9 May 2016, https://www.nationalgeographic.com/news/energy/2015/03/150306-why-your-fridge-pollutes-and-how-its-changing/.

[18] Instructables. “How to Build Arduino Weighing Scales.” Instructables, Instructables, 11 June 2019, <https://www.instructables.com/id/How-to-Build-Arduino-Weighing-Scales/>.

[19] Codippa. “Home.” *CoDippa*, 17 June 2017, <https://codippa.com/how-to-convert-a-file-to-byte-array-in-java/>.

[20] Menon, Gopinath. “All You Need to Know about Google Cloud Vision API.” *Digital Product Insights*, Digital Product Insights, 25 July 2017, <https://www.cognitiveclouds.com/insights/all-you-need-to-know-about-google-cloud-vision-api/>.

[21] <https://tutorials-raspberrypi.com/connect-control-raspberry-pi-ws2812-rgb-led-strips/>

Appendix

A1. Team Member Contributions

A1.1 Nicholas

General Contributions

* Distributed tasks for phase 2
* Kept track of phase 2 progress to ensure deadlines are met
* Attended weekly meetings with project supervisor
* Attended ELSEE meeting
* Updated the Gantt chart and task breakdown
* Began learning python language for use with the development board
* Began learning SQLite for use with the application database

Report Contributions

* Abstract
* Review of Design Specifications
* Completed Work Review
* Alternative analysis of temperature sensor, hall effect sensor, camera module, software
* Software design of board, application, user interface
* Deviations
* Updated Schedule
* ELSEE: Ethical, Legal, Environmental

Software Design

* Responsible for selecting software components
* Responsible for designing part of the application
  + Designed the user interface
  + Designed the local databases for board data from server
  + Designed recommended shopping list logic
  + Designed interior view logic
  + Designed various flow charts and diagrams
* Responsible for designing part of the development board code
  + Designed the door open/close status logic
  + Designed the image capture logic
  + Designed the load cell weight measurement logic
  + Designed various flow charts and diagrams

Hardware Design

* Responsible for selection process of components
  + Selected temperature sensor
  + Selected hall effect sensor
  + Selected camera module

A1.2 Georgiy

General Contributions

* Attended weekly meetings with project supervisor
* Attended ELSEE meeting
* Began learning python language for use with the development board

Report Contributions

* Alternative analysis of load cells, amplifier, buzzer, led
* Hardware schematic
* Hardware configurations
* ELSEE: Economical

Software Design

* Responsible for assisting in designing part of the development board code
  + Assisted in design of the door open/close status logic
  + Designed load cell weight normalization

Hardware Design

* Responsible for selection process of components
  + Selected load cells
  + Selected amplifier
  + Selected buzzer & LED
  + Selected refrigerator
* Responsible for designing hardware components
  + Designed hardware schematic
  + Designed 3D models
  + Designed all sensor and device configurations

A1.3 Yvann

General Contributions

* Attended weekly meetings with project supervisor
* Attended ELSEE meeting
* Attended lectures to keep up to date on course information
* Began learning SQLite for use with the application database
* Began learning PostgreSQL for use with the server database

Report Contributions

* Alternative analysis for development board, server
* Server design
* Server logic
* ELSEE: Social

Software Design

* Responsible for selection process of server
* Responsible in assisting in designing part of the application
  + Assisted in designing the local databases for board data from server
  + Designed inventory management
  + Designed recipe suggestion search
  + Designed authentication
* Responsible for designing part of the server
  + Designed database structure
  + Designed database logic
  + Designed various figures and diagrams

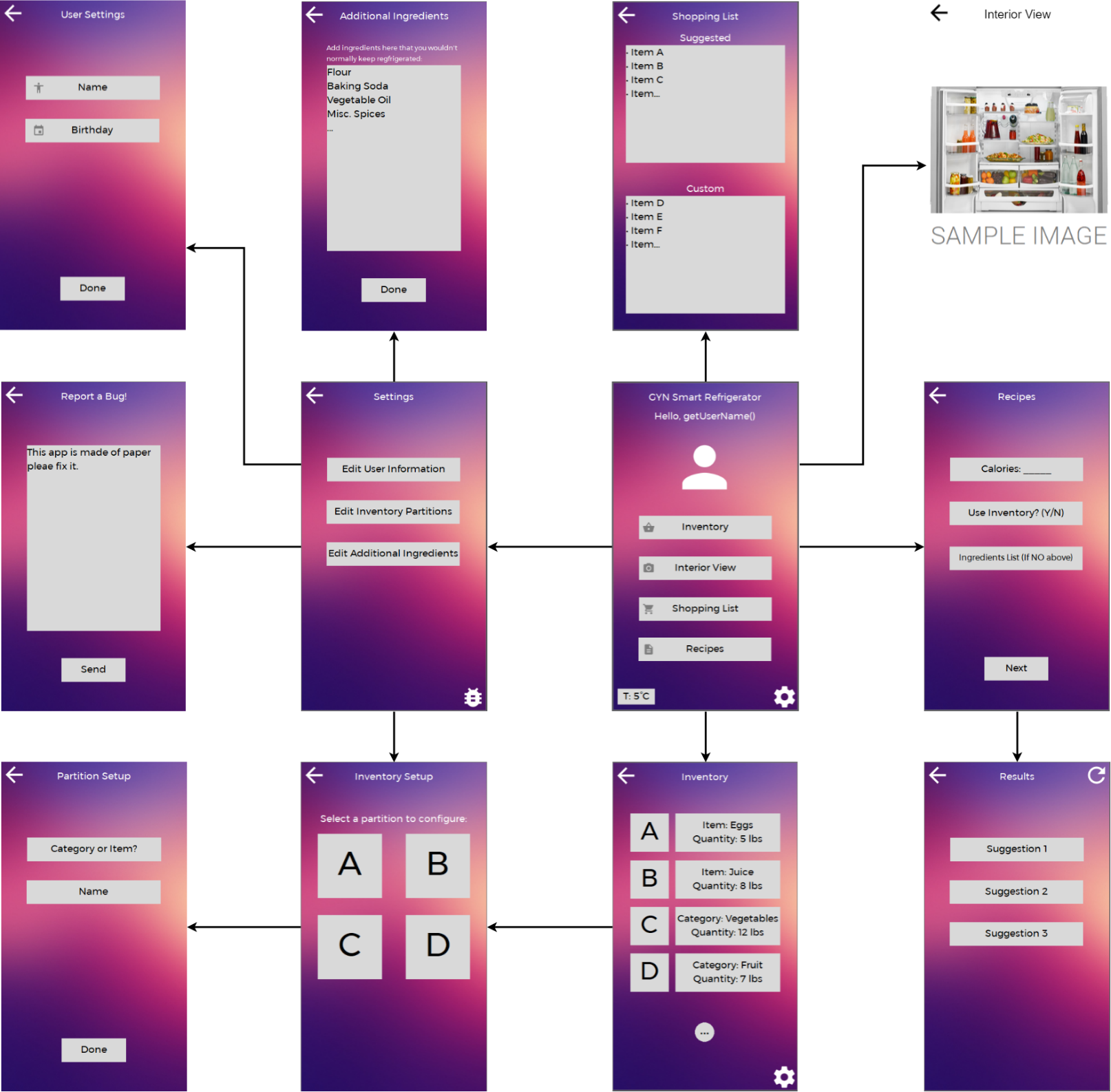
Hardware Design

* Responsible for selection process of the development board

A2. Application User Interface Page Sequences

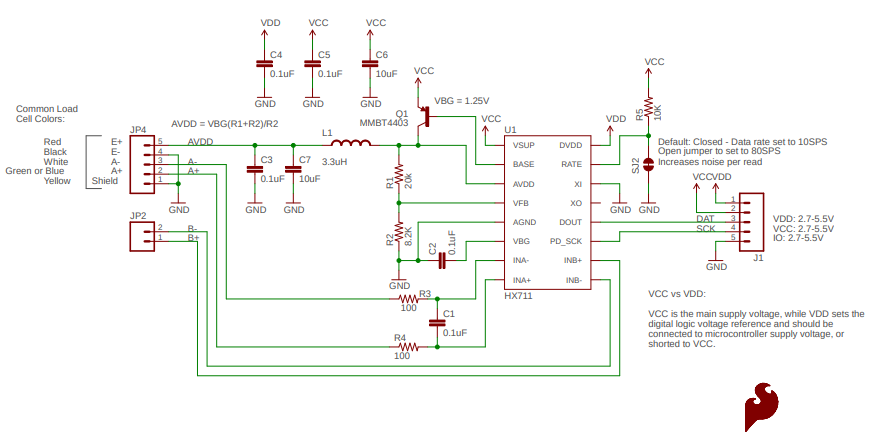
A2.1 UI Page Sequence

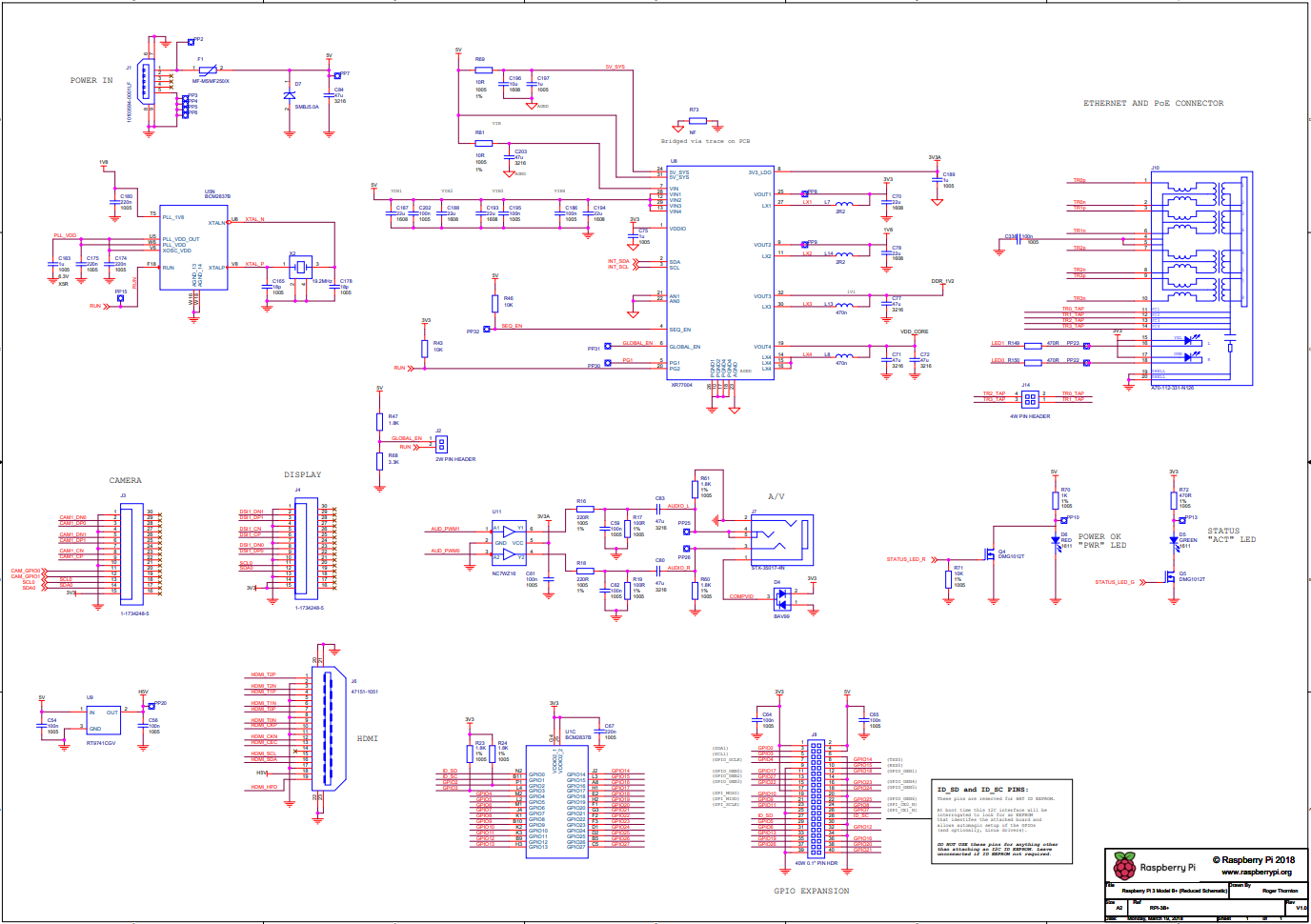
Below is a proposed page sequence of the designed user interface. A maximum of 3 transitions can be made from the welcome page. The star shape denotes the welcome page of the application.



A2.2 UI Sequence Diagram

A3. Schematics

A3.1 HX711 | Source: <https://cdn.sparkfun.com/assets/f/5/5/b/c/SparkFun_HX711_Load_Cell.pdf>

A3.2 Raspberry Pi 3 B+

Source: <https://www.raspberrypi.org/documentation/hardware/raspberrypi/schematics/rpi_SCH_3bplus_1p0_reduced.pdf>

A4. Ethical, Legal, Social, Environmental and Economical Implications of the Project

A4.1 Ethical

This project uses various devices and sensors to record data such as food quantities and images of the interior contents of the refrigerator. This information must remain secured. As mentioned in the legal section, a privacy policy can be included with the application to inform the user on how the information will be used. As such, it is very important that those policies aren’t broken in any way. This includes immoral actions such as selling the users data to a third party. Furthermore, there needs to be a way to protect the user’s data from potential malicious attacks, unprotected data can result in incidents such as identity theft. A possible solution would be to encrypt the data, adding a layer of security, different types of encryption provide different levels of security.

Another privacy issue arises with the use of the camera. With a camera comes the possible misuse of the device. If the refrigerator is being used in an apartment setting, some tenants may not feel comfortable with the camera. For example, if in some case, a tenant’s face is captured in one of the images, this can be a violation of their privacy. To avoid any such instance, the camera should be installed at a location that minimizes the chances of said scenario.

A4.2 Legal

Privacy of data has become an important issue in modern society. The Personal Information Protection and Electronic Documents Act (PIPEDA) provides rules on how to collect, use and disclose the personal information of the users using the product. Before any data can be collected for use with the application, the user must give their consent to collect, use or disclose the information. This information may only be used for the purpose to which the user has consented, and the user must also be able to access said information and change it as needed. A privacy policy can be generated to gain consent, outlining some important criteria such as the data that will be collected and if it will be shared or not, how it will be used, how it will be stored, the security measures to protect it, the duration it is stored, and how to contact.

Since the project is using some personal information such as dietary habits with the food stored in the fridge, as well as a camera that takes photos of refrigerator contents, it is important that this data remains safe. An important risk to consider is the possibility of a malicious attack to breach data stored on the server which could compromise the information. Furthermore, vulnerabilities in the software of the refrigerator must be considered, which could give attackers far more information than just the data from the project. While traditional refrigerators do not have the quality of life features that smart refrigerators do, they do not have any risks of leaking any sensitive information.

A4.3 Social

Obesity in Canada has always been prominent. The recipes suggestion feature can encourage users to maintain a healthy lifestyle by offering a plethora of healthy recipes. The potential impact of the refrigerate to incite users to go for healthier food options could ultimately reduce the percentage of obese adults. Furthermore, the recipe suggestion could also encourage users to stay at home and cook their own dinner or cook dinner for friends and family instead of ordering food through a food delivery service. Overall, the ultra smart refrigerator would positively affect the social aspects of users lives.

A4.4 Environmental

The ultra smart refrigerator being developed in Quebec and used in Quebec assumes that the refrigerator will be operating in homes that use hydroelectricity, a clean source of electricity. One major concern is the disposal of appliances. Refrigerators, despite being designed for cooling your food, have been a factor in global warming. If not recycled properly, the freon that keeps the contents of the refrigerator cool is responsible for depleting the ozone layer. Over time, appliances become old and will need to be replaced, many facilities exist in Quebec for disposing old appliances so that they can be properly recycled and not left in a landfill. The concern arises with the fact that it cannot be guaranteed that the users disposing this device at a proper facility.

A4.5 Economic

The ultra smart refrigerator in its current state is priced far below the typical smart refrigerator from the bigger brands. The price point enables new buyers to experience the features of a smart refrigerator at a reasonable cost. Smart home technology is becoming more prominent in the modern day and a new affordable smart refrigerator could negatively affect competing refrigerators by pushing them off the market. Furthermore, with the implementation of the recipe suggestion feature, it might become ineffective to purchase certain cooking resources such as recipe books, ultimately having a negative impact on the sales of these products.

|  |  |  |
| --- | --- | --- |
| Storage Type | Shared Preferences | SQLite |
| Data Capacity | Small | Large |
| Queryable | No | Yes |
| Access Time | Very Fast | Depends on DB Size |
| Efficiency | Low | High |
| Notes | Must be pre-defined | Dynamically update data |

|  |  |
| --- | --- |
| Shared Preferences | SQLite |
| -Supports small amount of data  -Data is saved under a key  -Must have the key to read the data  -Need to define a key for every single data  -Variables must be pre-defined  -Searching is very inefficient | -Support large amount of data  -Can be queried to obtain a subset of data  -Can search tables for data  -Dynamically update database contents  -May be slower to read data for large databases |