CpE 191/EEE 193B Senior Design

Project DispenSUM

Ben Green, Dana Natov, Nael Numair, Jennifer Ong, Nick Rarick

Table of Contents

List of Figures

List of Tables

EXECUTIVE SUMMARY

Almost everyone must take medication in their lifetime. Poor medication adherence, or not adhering to medication treatment plans, takes the lives of 125,000 Americans annually and costs nearly $300 billion a year in additional use of the medical system. The problem affects all demographics but people 85 year and older take an average of 13 different medications and those 65 years and older are the most adversely affected by poor medication adherence. Solutions to the problem are numerous. The DispenSUM smart medication dispensing device is an easy to use, safe, and a potentially effective solution that will be unique in the medical devices market.

The DispensSUM platform provides a new method of medication adherence by reducing the patient's required effort to organize, store, refill, and remember when as well as how much of each medication they must take. The uniqueness of the machine is primarily due to the implementation of cheap to manufacture and reusable medication cartridge. The cartridge arrives at the patient's door pre-loaded with their required medication and a dosing schedule stored in onboard memory. The patient simply taps the cartridge to the side of the machine, which loads the dosing schedule into the machine, and is instructed to place the cartridge into an available slot. The DispenSUM platform then dispenses the patient's medication on the pre-loaded schedule, helping the patient adhere to their medication therapy while removing the burden of organizing and remembering when to take their pills. This truly unique aspect also gives rise to a new method of medication delivery, presenting a business opportunity in the form of the cartridge delivery services. The DispenSUM system has also been designed with caretakers in mind.

The DispenSUM machine is part of the new generation of, "the internet of things". By implementing internet connectivity, the platform provides caretakers and loved ones with the tools to help keep track of the patient using the machine. It tracks medication adherence and updates anyone that is programmed into the database. It can also warn caretakers and doctors if a dose is missed and order refills from the pharmacy when a cartridge is running low. This communication allows for the patient to retain their independence while helping caretakers and doctors feel confident that medication is being taken on time and in the right dose.

The DispenSUM platform has been designed with safety in mind. It includes several system redundancies, data logging, internet connectivity, and user feedback to ensure safe operation by the user and reduce downtime. By following FDA guidelines concerning electronic medication control devices more closely than any competitor on the market, the DispenSUM platform may be more capable of quickly going through regulatory "red tape" on its way to the market. It may also allow more insurance plans to cover the medical device, helping the platform reach a much larger market.

Though there are existing medication assistive devices on the market, The DispenSUM platform provides an innovative way to reduce the burden of taking medication while increasing medication adherence. The programmable and reusable cartridge reduces the user interaction to a level equivalent to receiving a DVD in the mail and watching it at home. The use of these cartridges also presents the opportunity to develop a subscriber based service to potentially change how medication arrives in the homes of the user. The platform adheres to FDA guidelines more than the existing competition and is designed with safety in mind, allowing it to potentially reach a larger user base more quickly. Ultimately, the goal of the DispenSUM platform is to help increase medication adherence while delivering independence and freedom to the end user.

DispenSUM - Deployable Prototype Documentation

A Solution to Medication Non-Adherence

Ben Green, Dana Natov, Nael Numair, Jennifer Ong, Nick Rarick

College of Engineering and Computer Science - California State University, Sacramento

***Abstract* — Our device, a medication adherence aid, is designed to properly dispense accurate medication on a unique timed schedule tailored to the owner of the device. Named DispenSUM, this device will help tackle our societal problem of medication non-adherence by keeping an individuals medication adherence in check. Our device, now functioning as a deployable prototype, has many different elements of organizational, analytical, logistical, mechanical, electrical and programming aspects. This document describes all the aspects regarding the design process and demonstrates our results.**

Keyword Index— DispenSUM, Electronic Pill-box, Software Testing, Electronic Testing, Mechanical Testing

I. INTRODUCTION

The use of medication is very common, as almost everyone at one point has or will take some type of medication. Doctors and pharmacists prescribe patients with medication but some do not take their medication as prescribed because they are taking their medication incorrectly or are they are not taking it at all. Over the years, this has become an increasing concern to health organizations, the government and even pharmaceutical companies due the multitude negative consequences that arise from medication non-adherence. The term "medication adherence" is commonly used when discussing medication use as it describes the extent to which a patient follows medical instructions [1]. Throughout this report medication non-adherence and medication non-compliance are used interchangeably.

After taking a closer look at the impact of medication non-adherence, our team decided to develop an effective engineering solution to address this problem. We designed a automated pill dispensing device. DispenSUM is the name of the design and it is an electronic pill organizer that manages and dispenses medication. With DispenSUM, users will never forget to take their medications again or double dose as it provides timed, accurate, and consistent medication. It is designed specifically for those who need help managing their medication such as the elderly who are prone to forgetting, people who have to take multiple medications, and those taking medication without professional supervision. The goal of DispenSUM is to provide a solution which will improve quality of life, reduce health care costs, and help retain people's independence as it can used in the comfort of one's home.

One of the key philosophies behind our design was the idea of Safety, Usability and Medication Adherence (SUM). We use the SUM philosophy to help guide every single design decision. This not only gave us a defining principle that helped guide us through this process, but it helped us identify clear goals in which the whole team could work together on.

Throughout our design process one of our main priorities was to develop a design that differs from the other pill-boxes currently on the market. In order to do so, we modified some of the features that already exist on other pill boxes while adding several new features. The team considered several different implementations for each feature in order to come up with a design that would improve safety, user friendliness, and medication adherence. The team developed the unique idea of the smart pill cartridge based design. The idea is to collect the dosage and pill information from the doctor or pharmacy and

load that information on a pill cartridge. User's of DispenSUM simply need to load the cartridge into the system. They are automatically reminded when their pills need to be taken.

This project ran over two semesters. At the end of the first semester we had created a working laboratory prototype. We followed a clear design process which included creating a design idea, work breakdown structure, project timeline, and risk assessment. In the second semester the team worked on improving and implementing additional features to the device in order to take our device from a laboratory prototype to a deployable prototype. The following report documents and describes the work that was required to develop the team's design idea, assemble a laboratory prototype, perform appropriate testing and market research in order to create a deployable prototype.

II. SOCIETAL PROBLEM

Medication adherence has been identified as a societal problem due to the overwhelming statistics that highlights the extent and impact it has on society. Approximately 50% of patients do not take their medication as prescribed and this results in increased morbidity and mortality rates, as well as healthcare spending [2]. Research shows that poor medication adherence takes the lives of 125,000 Americans annually, and costs the health system nearly $300 billion a years in additional doctor visits, emergency department visits and hospitalizations [3].

Medication non-adherence is most prevalent among the elderly and people suffering from a chronic disease. This is due to the fact that elderly people are more prone to forgetting to take their medication which is the number one reasons for medication non-compliance. As for people with a chronic disease many have to take medication consistently which can become fairly costly so they stop taking it. Statistics for non-compliance due to certain reasons mentioned previously is shown below in Figure 1 among with several other reasons for non-compliance.

Reasons for non-compliance*	
Forgot to take	79%
Ran out of medication	19
Too costly	9
Patient-perceived lack of need	9
Side effects	7
No improvement seen	3

*Totals more than 100% due to multiple mentions.
Credit: Wilson Health Information and The J. Scott Group, 2006.

Fig. 1 Reasons for non-compliance

Research has shown that the rate of medication non-adherence is most likely to continue to increase due to the growing rate of population ageing and increase in life expectancy. The reason why an increase in population ageing and life expectancy affects rates of medication adherence is because medication non-adherence mainly affects the 65 years or older age group. Not only does medication non-adherence effect patients but also others, for example it can cause financial and emotional stress on patient's family and caregivers. Therefore, it is imperative that this issue be addressed to mitigate the impact it has on society.

III. DESIGN IDEA CONTRACT

This device is a medication management system intended for use by the elderly at home. The device will be able to accurately and timely dispense medication tailored to the patient's medication needs. Our whole device centers on a smart pill cartridge based design, simplifying the reloading process and medication identification. The cartridges will hold all the necessary information on patient dosages and medication amounts. Also the cartridge will be designed so that medication can be pre-filled. This ultimately reduces user effort and helps promote adherence. Now, all the user has to do is insert the cartridge and press a single button to get medication. A quick conceptual sketch of our desired final product is shown in Figure 2.

Fig. 2 Concept Art of Final Product

Some guidelines and features were highlighted and suggested by the FDA, specifically the Center for Devices and Radiological Health [4]. We also reviewed the Code of Federal Regulations to help us identify our device as Class II (special controls) [5]. Using these guidelines help us achieve a higher level of compliance with regulations in an attempt to prepare for when we want to bring our device to market. Using what we learned in the Problem Statement, our SUM ideal and Federal Regulations, we came up with a feature list we see as adequate in scope.

A. Features List

Swappable Cartridge(s) - The swappable cartridge design helps us provide a versatile, yet easy to use, device. This makes it possible to change or renew your medication schedule in the product by simply placing it in the base system. The cartridge will come pre-programmed with a medication schedule. The cartridges will communicate with the base wirelessly with built in radio frequency identification tags (RFID) which will program the base with the correct medication schedule for the cartridge medication. This will prevent misuse, accidental overdosing and promote proper adherence. It also allows for a diverse selection of pill or capsule medication. This is the cornerstone piece of our prototype.

Rotary Base Module - The rotary base is simply a cylindrical base used to rotate the cartridges over a dispense cup. This module provides versatility, allows us to use multiple cartridges in one system and helps us comply with several FDA regulations. A drawing of the rotary base module can be found in Appendix D Part III.

Two Button Design - The two button design is simply a two button panel for the user to interact with the device. This design was heavily influenced by our SUM principal. It is used to simplify the experience of the user. One button will be to dispense medication and the other button allows the user call for help if needed.

Raspberry Pi (RPi) Touchscreen - Similar to our three button design, the RPi touchscreen is another feature implemented for the user to interact with the device. The "touch" functionality of the screen is simply to provide more capability to the end user if needed and give caretakers or technicians the ability to configure the device. The large screen will allow us to give the patient feedback on what medication they are about to take.

Light and Sound Notifications - The light and sound notifications alerts users that their medication is ready to dispense. The notification will turn off when the dispense button is pushed otherwise it will continue for a specified number of minutes until turning off. If the dispense button is not pushed within the specified time then an email notification will be sent to the users caretaker email.

Redundant Systems - In order to comply with FDA regulations we decided to introduce many sensors that allow for us to very accurately identify when a pill has or has not been dispensed. A load cell will be used to accurately count the amount of pills in the dispense cup. In addition, we are also integrating a close-proximity infrared sensor that detect if a pill has been dispensed through the pill shoot. Many software redundancies will also be utilized to ensure accurate dispense amounts. These systems are critically important for the function of our device. This is also heavily influenced by

our SUM philosophy and our compliance with FDA guidelines.

Interaction Unit GUI Program - The Interaction Unit features the three button panel and RPi touch screen. This unit will be integrated on the front end of the physical device. A GUI program will be created for the RPi touch screen so users can easily interact with the device. The GUI program will display what medication the user is taking, the current remaining pill count, and all possible warnings you will see on a standard pill bottle.

Pharmacist GUI Program - For the cartridge system the overall idea is that the pill cartridges will be programmable by implementing RFID technology. To keep things simple a graphical user interface (GUI) program will be created for a medical professional (e.g. pharmacist) to enter in the medication information. The program will provide the user with a form for them to fill-in, which then is converted into a text file and transferred to the interaction unit (raspberry pi). This form would have fields for the following, medication name, emergency contact, medication quantity and dose, medication times, and any other medication notes.

Email Notifications - A notification will be sent to the emergency contact via email when medication is not taken within a certain timeframe. The Raspberry Pi will be used to implement this feature by connecting the Raspberry Pi to Wi-Fi. Then a python script will be created which will monitor the sensor that detects whether the medication has been taken from the pill-dispensing box. The python script will have code that enables an email notification to be sent using the programmed emergency contact information if the medication has not been taken. Also, an email notification will be sent when medication is running low so medication refill can be arranged.

RFID System - The RFID system is used as the medium memory between the pharmacist GUI and the Interaction GUI. This consists of a two part system. The first part consists of writing to the tag. This process takes the information from the pharmacist GUI using Mifare read-writer which is connected to an Arduino uno and sends the information over. The second half to this system is associated with the Raspberry Pi 3. Using a python equivalent version of the code, the information is then transferred from the RFID tag into the Pi in the form of a text file. From there a python script takes the information and converts it from decimal to ASCII.

IV. FUNDING BREAKDOWN

Project DispenSUM was a non-sponsored project. Each group member has contributed money to buy the necessary items in order to complete the project. The SUM design philosophy does not specify or require the need for the system to be low cost. However, it has been a goal to make the project as affordable as possible. Total cost to date is $1375.22 which includes everything that we have bought since day one of this project. Total does not include items we had already owned. The total cost of the project does not reflect that actual cost to make because not all items bought were used in this final iteration of the project.

Cost	**Supplier**
$26.00	Ace Hardware
$808.92	Amazon
$50.14	Fry's Electronics
$18.08	Home Depot
$313.10	McMaster-Carr
$158.98	Others

Table 1 - List of Suppliers

The table above shows a list of suppliers that we went to for parts. Others consist of IEEE student branch at Sacramento state, Rockler Woodworking and Hardware, DFRobot.com, Blick Art Supply, Smoothon.com, WireCare and FRC Team 1678 these groups listed in other were contacted for either one time purchase or use.

V. PROJECT MILESTONE

Originally, project DispenSUM has six key components. These included the module, the cartridge, light and sound notification, redundant systems, LCD Screen, and the pill crusher. Due to time limitations, the pill crusher was not included in the laboratory or deployable prototype. The design components that were kept were verified in the laboratory prototype and included in the deployable prototype. Major product redesigns were executed in the second semester development cycle. This included redesigns of all of our project components. Most of these features are independent of each other until the end when each needed to be integrated into the final product.

During the development of both the laboratory and deployable prototype, the completion of each key component represented a project milestone. Integration of all the components into the full design represented a milestone, followed by the final development step of testing and finalizing each prototype. This final step was the last project milestone in both of the product development cycles. There were a total of eight project milestones each semester, or development cycle.

VI. WORK BREAKDOWN STRUCTURE

Project DispenSUM is a large project with many interconnected pieces. In order to break the DispenSUM project down into more manageable pieces, a work breakdown structure was developed. The work breakdown structure (WBS), allows the project members to schedule individual work packages, determine the order they must be completed in, and assign group members to work on them. The tables included in this section outlines the tasks each group member worked on and how much time was spent on each task.

Table 2 Jennifer's Task Hours

Hours	Task
4	Setup Raspberry Pi
15	Programming push-button, LED, and motors
10	Communication between microcontrollers
25	Read time program
20	Laboratory prototype full system integration and debug
50	Pharmacist GUI
35	User/Interaction GUI
25	Deployable prototype full system integration and debug
100	Documentation
284	**Total Time**

Table 3 Nael's Task Hours

Hours	Task
80	Documentation
18	Microcontroller Setup
10	Communication between microcontrollers
25	Full system integration and debug Laboratory Prototype
40	Full system integration and debug Deployable Prototype
30	RFID System
35	Function Code
238	**Total Time**

Table 4 Ben's Task Hours

Hours	Task
97	CAD Design
38	Fabricate
39	Debugging/Integration
89	Documentation and paperwork
263	**Total Time**

Table 5 Nick's Task Hours

Hours	Task
70	CAD Design
22	Audio Circuit
8	Cartridge Design
39	Debugging
41	Documentation
13	Linkage System
38	Arduino System Logic
241	**Total Time**

Table 6 Dana's Task Hours

Hours	Task
60	CAD Design
18	Testing system integration and feedback sensors
19	Machining
38	3D Printing
11	Wiring
60	Documentation
15	Hardware Implementation and testing
10	Programming
230	**Total Time**

VII. RISK ASSESSMENT

The risk mitigation plans utilized in the prototype phase of project DispenSUM include deploying parallel critical paths, executing redesigns early in the design process, and utilizing feedback systems to ensure proper functionality of the device. The parallel critical paths are used to ensure as many tasks are being completed early in the design process by all group members. The major required redesign involved the cartridge and it was executed early in the process.

A quick overview of our risk assessment for our prototype produced a risk matrix we used to identify the riskiest sections of our design. We used this matrix to identify what needed sufficient mitigation plans. The risk matrix is shown in Figure 3 below.

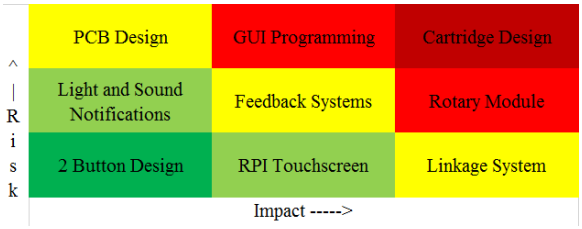


Fig. 3 Risk Matrix

The risk matrix allowed us to focus on getting the critical parts in our project done first, such as the cartridge design, GUI, and Rotary Module. Focusing on the high impact, high risk parts gave us enough time to pivot to a new design if a design failed. Using this strategy we were able to complete the project in the allowed time.

VIII. DESIGN OVERVIEW

Our design philosophy has always followed our principle of SUM, which is safety, user friendly, and medication adherence. The project has been designed from the ground up with these goals in mind. The device is meant to be used in the home to keep track of the user's medication schedules and to help them take their medicine correctly. For a device to achieve this it has to be very easy to use and to be autonomous needing little to no maintenance from the user.

To meet these requirements we created a cartridge that is designed to store a specific type of medication. These cartridges are delivered to the user's house from a secondary service. The cartridges come delivered with medication inside as well as the dosing schedule for that medication stored onto an RFID tag attached to the cartridge. The cartridge system was designed so all the patient has to do is receive the cartridge and insert it into the device. Once inserted the device will read the RFID tag and then be able to alert the user when they need to take their medication. Once alerted all the user has to do is press the dispense button and the machine will dispense all their necessary medication for that time slot.

This functionality meets the design requirements since the user has very little interaction with the machine which makes it very easy to use. This allows the user to get their medication safely and easily. The purpose is to alleviate the chore and safety issues associated with a patient trying to manage many different times of medication and their corresponding schedules. The machine achieves this since the user doesn't have to manage their medication at all, the system does it for them.

IX. PROTOTYPE STATUS

The prototype is in functioning order and is ready to demonstrate. Appendix D Part I shows the whole system drawing and the sections of our device. A smaller version of our deployable prototype can be found in Figure 4.

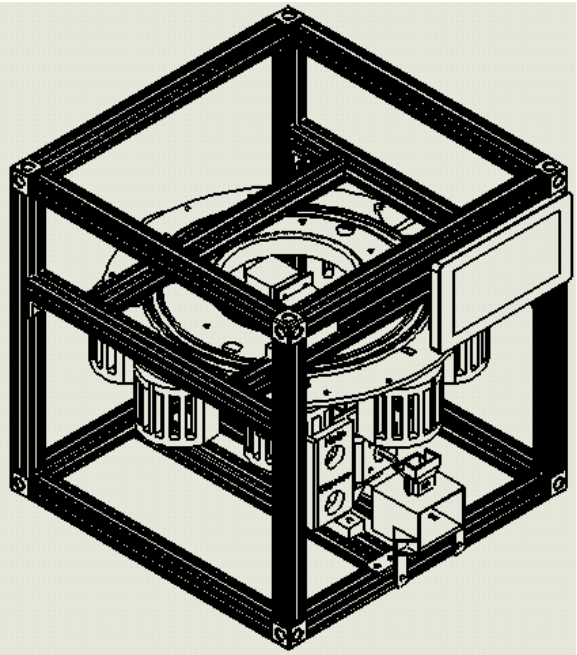


Fig. 4 Deployable Prototype

The deployable prototype satisfies all the conditions we laid out in our design contact above. It scans and collects cartridge information, dispenses accurate amounts of medication on a timer, uses light and sound notifications to warn the user, integrates a two-button design with an easy to use GUI and provides the necessary feedback mechanisms to ensure errors in the dispense cycle do not happen.

X. MARKETABILITY FORECAST

A market review was performed to determine the market size and current market status for our product. The major factors that drive the medication management market need include growth in population, health care expenditures and their related adverse effects of

medication non-adherence [6]. We found that many large and small companies are currently vying to fill the niche that the DispenSUM design fills [7], a medication control device used to help solve the problem of medication adherence. It has been determined that we are following market trends with our current design, but we will need to modify several aspects to compete directly with other products. It should be noted that we closely followed the Code of federal Regulation [8] and the Guidance for Industry and FDA Staff: Class II Special Controls Guidance Document: Remote Medication Management System, FDA [9] to increase our market viability. Following these guidelines and regulations allows our product to pass through FDA review and approval faster and make it more attractive to potential customers. However, we will need significantly more documentation in order to satisfy the FDA approval process. This is based on Dana Natov's work experience with Gold Standard Diagnostics and engineering for the medical field. We would also require significantly more testing of the design to prove that it can safely dispense medication. Testing and documentation are not the only necessary changes to help compete in the market.

Significant software changes are required to improve functionality as well as assist in scalability. Most of the code functionality implemented in our software, including the C and Python code, utilize functions from pre-existing libraries. In order to ensure full ownership, improve processing speed, and ensure cheap scalability, we must write our own libraries and functions. Doing so would improve code processing speed by reducing unnecessary functions and only processing the code we need. Proprietary code will also ensure no outside entity could claim ownership to any part of our project. Lastly, lean and original code will ensure maximum scalability by reducing any costs, which would be paid to license third party code, to zero.

The hardware changes that would improve market viability are similar to those that are necessary for the software. We are currently using three different microcontrollers in the DispenSUM design. We can reduce the number of microcontrollers by utilizing a multi-core microprocessor capable of supporting a full operating system such as Linux. Doing so will allow us to control the various design components, host a full graphical user interface as well as reduce the cost in terms of scalability. The ARM microcontroller family is currently the most viable candidate to replace our three different microcontrollers. They are low power, support operating system functionality, have multiple cores, and their speed is more than required to operate our hardware and software simultaneously [10]. Reducing our microcontrollers from three to one would reduce production costs and speed up development time. We can also reduce cost by not using the proprietary Raspberry Pi touch screen, replacing it with a lower cost touch screen and generic LCD driver. The aluminium extrusion utilized for framing should also be reduced or replaced with cheaper materials such as plastic. This will reduce unit production cost and allow for more competitive market pricing or a greater net profit per unit.

The findings from our Market Review Report indicate that we are on track with the current development trends [7]. We also found that the current return on investment for products similar to the DispenSUM design is low [6]. Implementing the changes discussed here will reduce the production cost per unit and drive the price down in terms of large scale production. It will also encourage faster development times with the reduction from three microcontrollers to one. Further testing and

documentation will expedite the regulatory validation requirements the design must pass through. Collectively, the changes can either reduce the cost to the end user or increase the rate of return on investment for the design. Either outcome will increase the DispenSUM design's ability to compete in the medication control and distribution market.

X. CONCLUSION

Medication non-adherence is a prevalent issue in our society. We decided to tackle this societal problem implementing an engineering solution. With a few changes in our product DispenSUM, we believe that it has the potential to make an impact on the problem of medication non-adherence. We also believe that our product has significant aspects that differentiate from other products on the market. Our design process consisted of meticulous research, constant iteration and principal guidelines. We believe our design would suffer substantially if these guidelines did not exist. We deem DispenSUM to be successful in demonstrating our ability as future engineers and a genuine product that is marketable beyond the scope of this course.

REFERENCES

[1] R. Scott Leslie (2013, July, 19). *Pharmaceutical Programming.* [Online], Available: http://www.wuss.org/proceedings08/08WUSS%20Proceedings/papers/anl/anl09.pdf

[2] W. H. Organization, E. Sabate, W. H. O. Staff, and Sabate Eduardo, Adherence to long-term therapies: Evidence for action. Geneva: World Health Organization, 2003.

[3] Medication Adherence - "Taking Your Meds as Directed." American Heart Association.

[4] Code of Federal Regulations 72 FR 59177, Oct. 19, 2007

[5] Guidance for Industry and FDA Staff: Class II Special Controls Guidance Document: Remote Medication Management System, FDA

[6] Transparency Market Research, "Medication Management Market - Global Industry Analysis, Size, Share, Growth, Tends and Forecas, 2016-2024," 2016. [Online]. Available: http://www.transparencymarketresearch.com/medication-management-market.html

[7] Benjamin Green, et al. "Market Analysis and Review," California State University, Sacramento, 2017

[8] Code of Federal Regulations 72 FR 59177, Oct. 19, 2007

[9] Guidance for Industry and FDA Staff: Class II Special Controls Guidance Document: Remote Medication Management System, FDA

[10] ARM Ltd., "www.arm.com," ARM Ltd., 1995 - 2017. [Online]. Available: https://www.arm.com/products/processors. [Accessed 1 May 2017].

GLOSSARY

1. **CAD** - Acronym for Computer Aided Design
2. **Cartridge** - One of the main parts of our design. The cartridge is a separate entity from the main unit that will hold all medication. The cartridge is designed to be swappable and will come with pre-programmed medical instructions.
3. **DispenSUM** - The project name for Group four's senior design project
4. **FDA** - U.S. Food and Drug Administration
5. **GUI** - Graphical User Interface
6. **Medication Adherence** - Adherence to, or compliance with, a medication regimen is generally defined as the extent to which a person takes medications as prescribed by their healthcare providers.
7. **RFID** - Radio-Frequency IDentification

8. **Rotary Carousel** - The rotating portion of the dispense module
9. **Slip Ring** - A rotating brushed contact design that enables a circuit to maintain contact through continuous 360 degree rotation
10. **SUM** - Safety, User Friendly, and Medication Adherence
11. **WBS** - Work Breakdown Structure

Pharmacist User Manual

Step one open application:

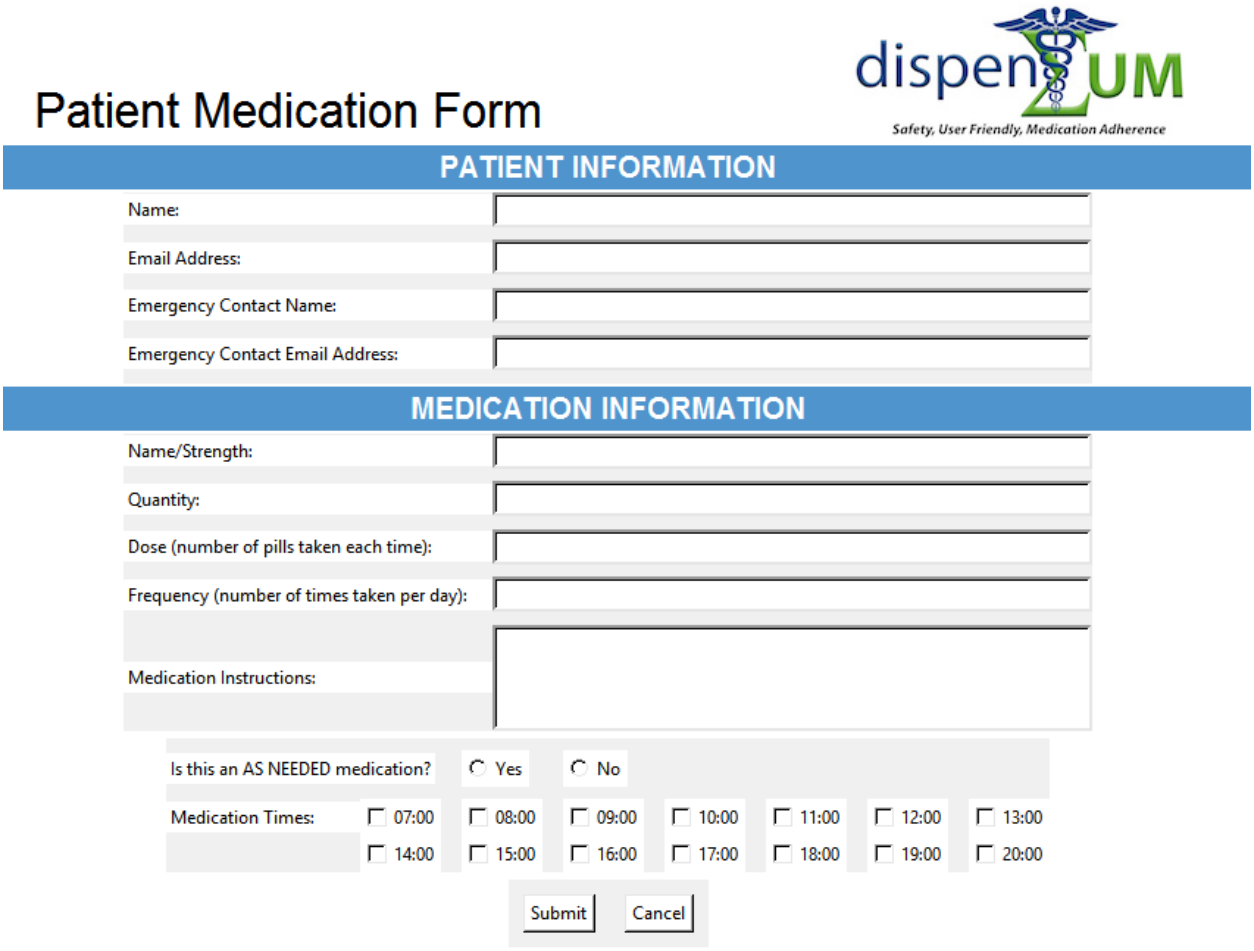


Figure 1 Patient Medication Form

Step two load information:

- Fill in all name and contact information accordingly. (no more then 28 characters allowed including spaces)
- Quantity, Dose, and Frequency only numerical value allowed
- Must select yes or no to "As Needed" field
- Then number of time stamps check need to match frequency number

Step three transfer data:

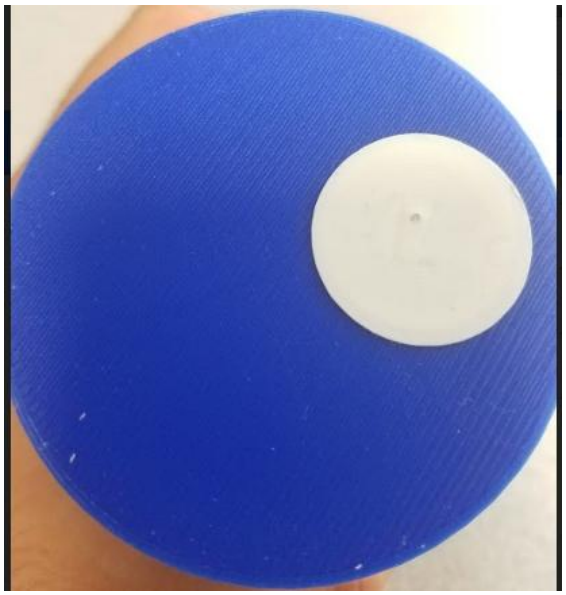


Figure 2 Bottom of cartridge and memory sticker

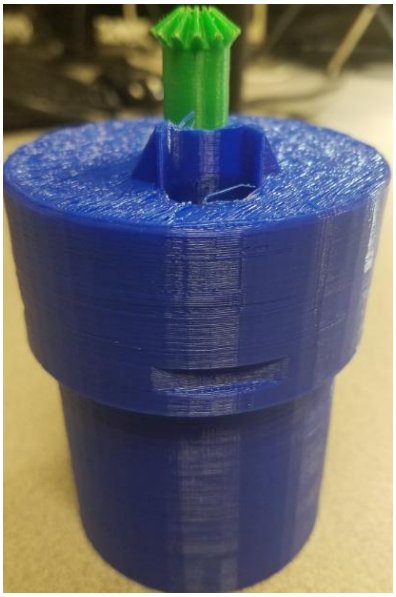


Figure 3 Medication Cartridge

- Make sure that the cartridge has an RFID Tag or Memory Sticker at the bottom of it

Figure 4 Data Transfer Box

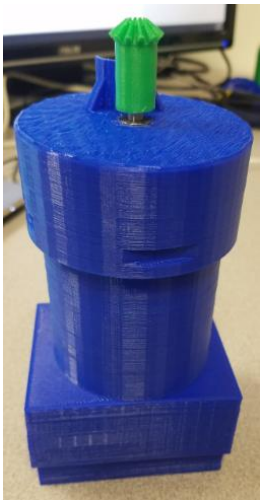


Figure 5 Cartridge and DTB

- Place cartridge onto the data transfer box(DTB) like shown in *Figure 5*
- Then select the **Submit** button on screen

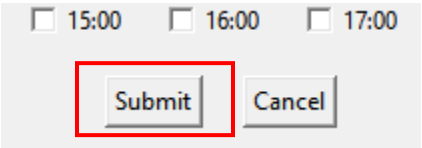


Figure 6 Fill Forum Submit Buttom

Home User Manual

Loading The Medication:

Step 1: Power on device

Step 2: Select the load button on screen

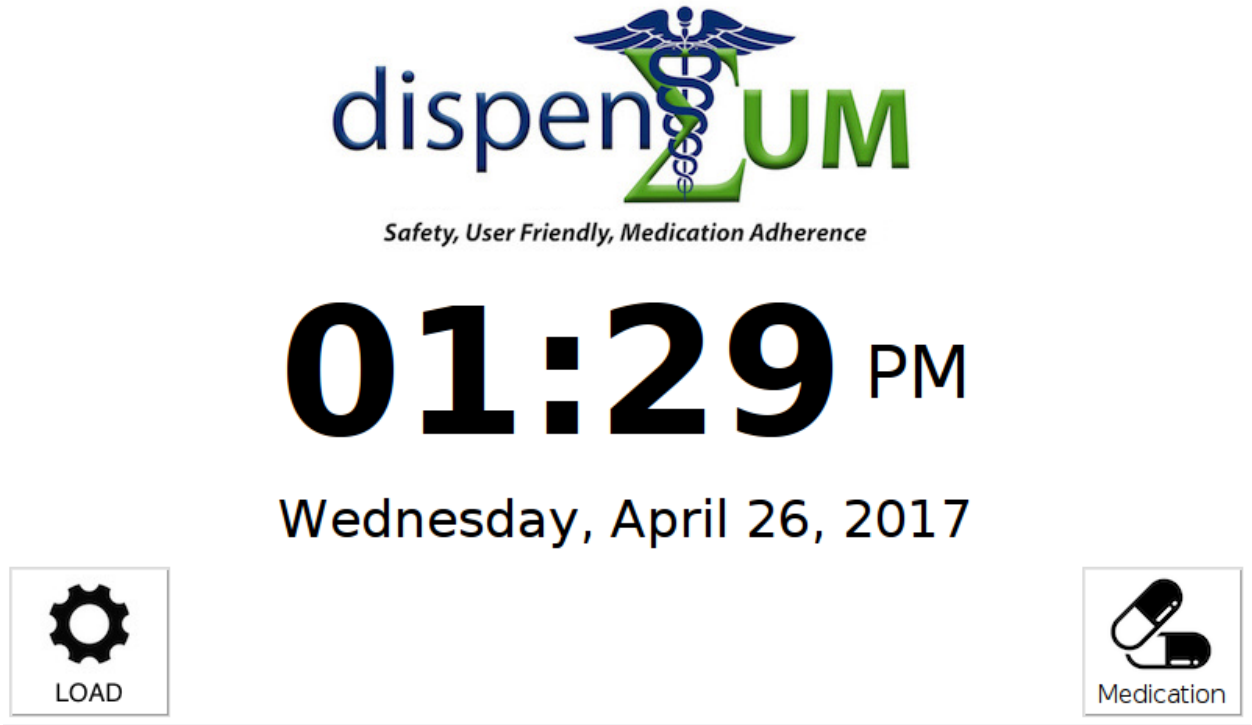


Figure 7 Main Menu

Step 3: Insert Cartridge into the holder upside down with the opening side closest to you.

Figure 8 Cartridge Holder

Step 4: Check to see if medication is loading by pressing the Medication button

If you don't see you medication on the list either try again or press the help button and your caregiver will be notified

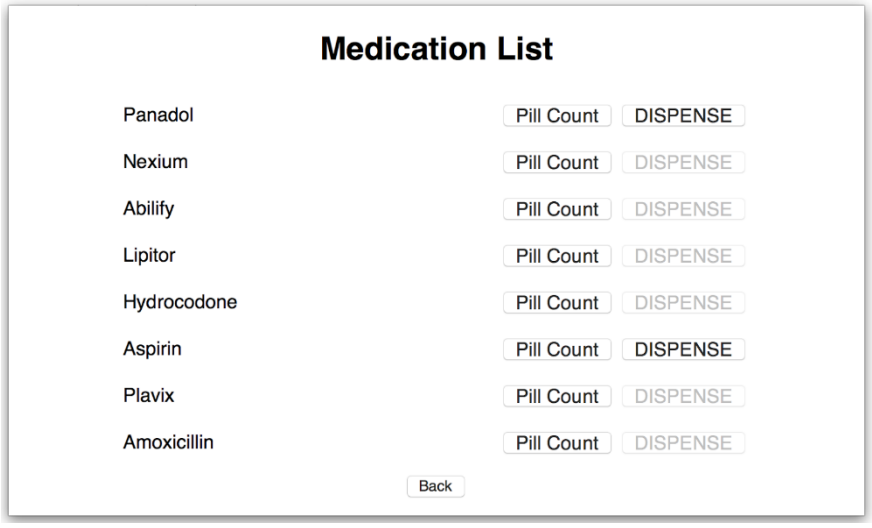


Figure 9 Medication List

Taking Your Medication:

Now comes the easy part all you have to do is wait for the notification sound to go off and then press the dispense button

Figure 10 Help and Dispense Button

Appendix B - Hardware

I. Micrcontroller

The microcontroller is the one of main components of the DispenSUM design, as it is responsible for performing all the functions and commands of pill-dispensing. We decided early on that our design would require two microcontrollers, the Arduino Uno and the Raspberry Pi. A third microcontroller was implemented to offload the processing demands of the load cell scale. The Adafruit Trinket, shown in Figure 1, was chosen for its small footprint, fast processing power, and the desired number of input and output pins it has.

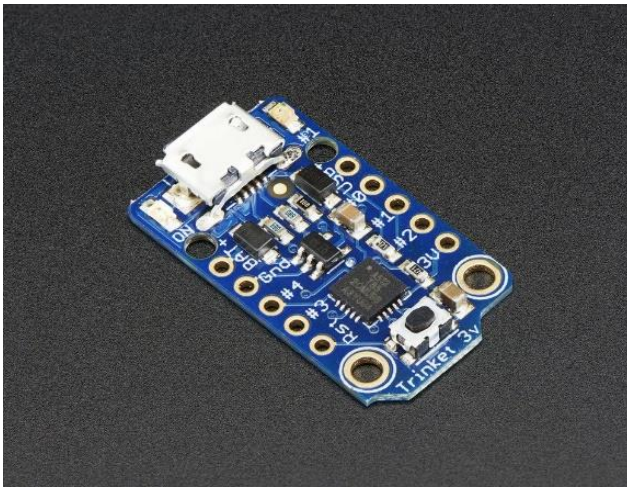


Figure 1 - The Adafruit Trinket Microcontroller – image provided by Adafruit

The addition of the Trinket microcontroller is necessary due to the processing demands of both the load cell scale and stepper motors. The two required fast code execution times.

Our design required the Arduino Uno and Raspberry Pi. pictured in Figure 2 and 3, to be able to interact with one another, therefore the USB port was used for communication between Arduino and the Raspberry Pi.

Figure 2 - The Arduino Uno Microcontroller – image provided by Arduino

The third microcontroller that was used was the Raspberry Pi 3. For the device we have the Raspberry Pi connected to the internet via Wifi, and it sends signals to the Arduino via the serial console through the USB communication port. Having the Raspberry Pi automatically connect to the internet is a key feature because it provides our device with an accurate clock which is need as medication is dispensed on a time schedule.

Figure 3 - The Raspberry Pi 3 Model B System on a Chip - image provided by Raspberry Pi Foundation

II. User Interface

The Raspberry Pi hosts a custom graphical user interface, utilizing a touch screen for user interface. Two large tactile push buttons are also implemented for easy user interaction. Figure 5 shows the touch screen and the two tactile pushbuttons.

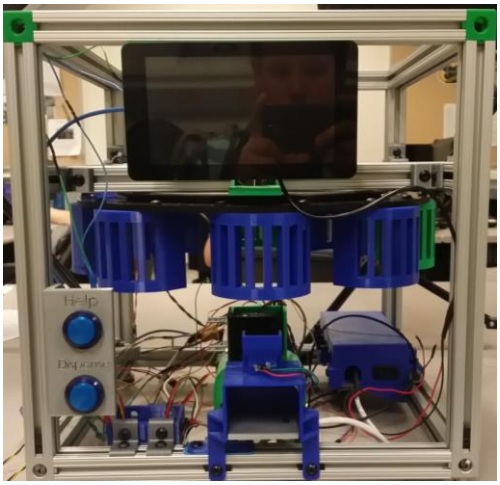


Figure 4 - The DispenSUM design with the touchscreen mounted on the top and the help and dispense button located on the bottom left of the design

The Raspberry Pi touch screen is the proprietary touch screen offered directly from the Raspberry Pi foundation. Figure 6 is the touch screen, unmounted.

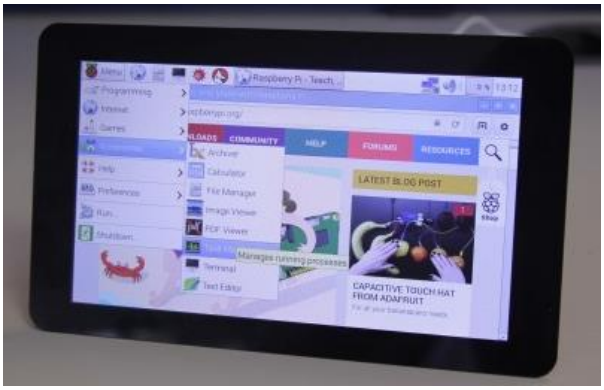


Figure 5 - The Unmounted Raspberry Pi touch screen – image provided by Raspberry Pi Foundation

III. Mechanical Drive Hardware

The hardware utilized for the drive system of the DispenSUM design include two 12 V 500 mA per phase NEMA 17 stepper motors, two Pololu DRV8825 stepper motor drivers, a single 7.2 V 180-degree non-continuous servo, and a single buck converter. Figure 6 is an image of the NEMA 17 stepper motor mounted in the design.

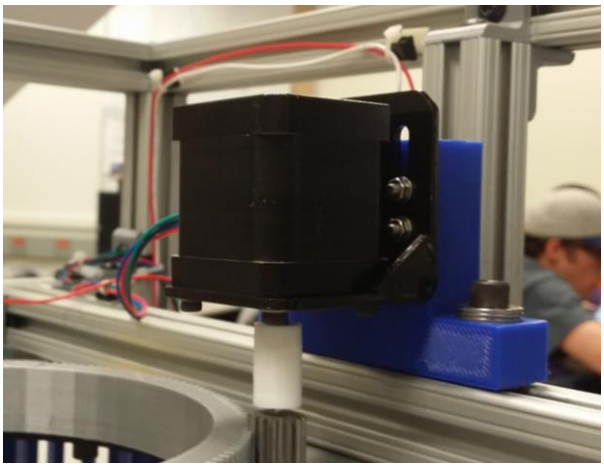


Figure 6 - The NEMA 17 stepper motor mounted in the DispenSUM design, driving the rotary carousel

The NEMA 17 motors require driver circuits to function. We chose the Pololu DRV8825 motor drivers, shown in Figure 7.

Figure 7- The Pololu DRV8825 stepper motor driver – image provided by Pololu Corporation

Figure 8 is an image of the servo mounted in the DispenSUM design.

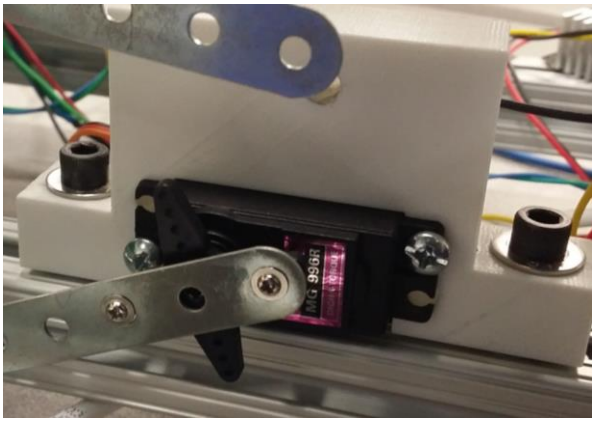


Figure 8 - The servo mounted in the design

The servo requires 7.2 V to operate so we utilized a buck converter to step the 12 V supply voltage to 7.2 V. Figure 9 is an image of the DROK buck converter implemented in our project.

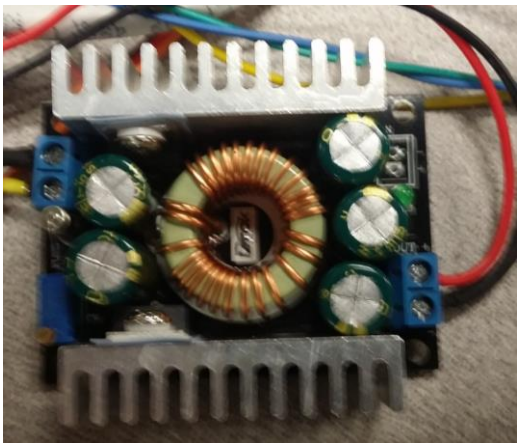


Figure 9 - The DROK buck converter

IV. Feedback Systems

There are multiple sensors and a RFID communication device utilized for feedback systems in the DispenSUM design. These sensors include a Uxcel load cell and SMAKN HX711 weighing sensor AD module for the load cell, a modified QTI sensor, and the MIFARE RFID reader. Figure 10 is the load cell.

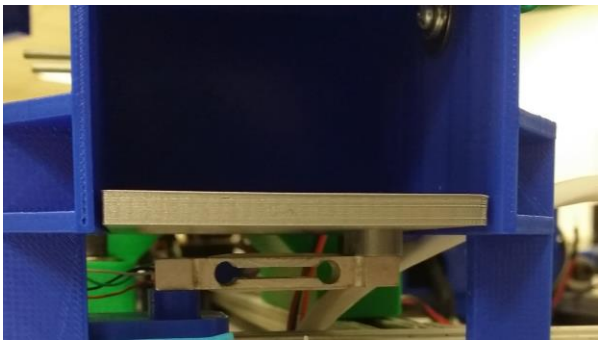


Figure 10 - The load cell mounted under the QTI chute and weigh plate

Figure 11 is an image of the HX711 weighing sensor mounted in the design.

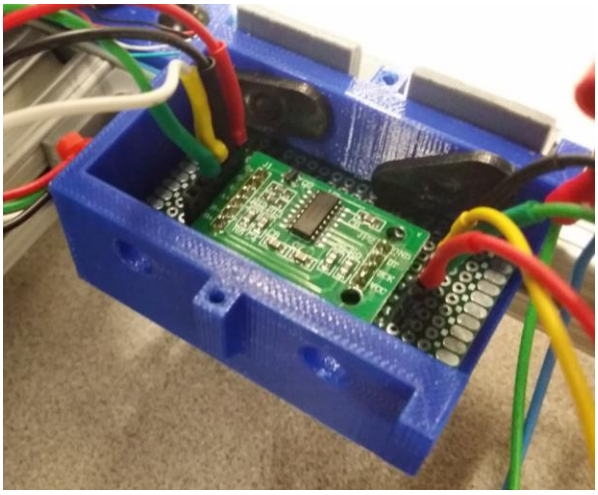


Figure 11 - The SMAKN HX711 weighing sensor

Figure 12 is an image of the modified QTI sensor mounted on the pill chute.

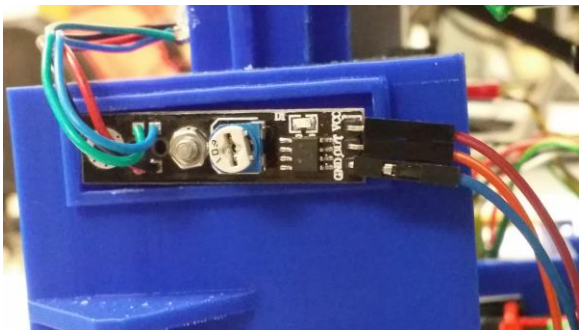


Figure 12 - The modified QTI sensor mounted onto the pill chute

Figure 13 is an image of the MIFARE RFID reader implemented in the design.

Figure 13 - The MIFARE RFID reader – Image provided by Gear Best

Appendix C. Software

I. Software System Overview

The software for this project is broken down into several parts which is shown below in Figure 1. By splitting the software into several parts we were able to work on different parts simultaneously and it made the testing process more efficient. There is four main software parts and they all depend on one another for correct functioning. The pharmacist GUI was the first program that was created and this was integrated with the RFID code which is a crucial part of the overall software system as connects the pharmacist GUI to the codes on the Raspberry Pi.

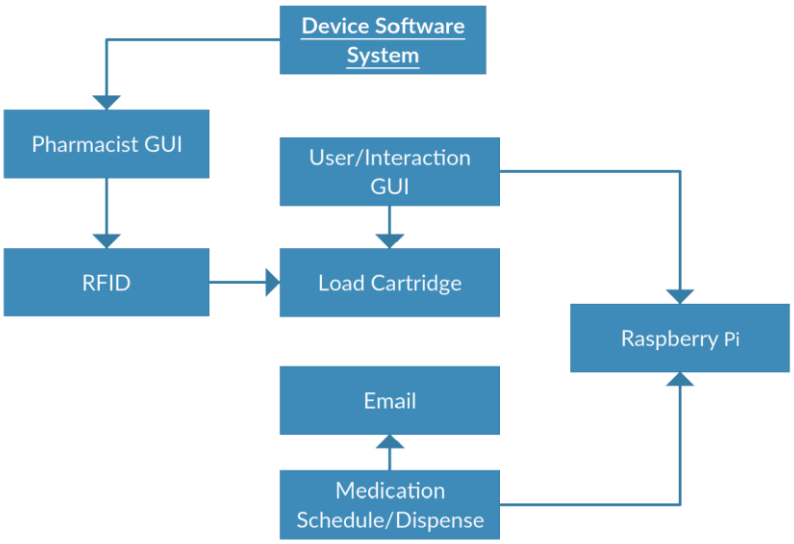


Figure 1. Software Block Diagram

II. Pharmacist GUI

The purpose of the pharmacist GUI program is to obtain medication information for our device. Below in figure 2. is a flowchart showing how the program functions. Included in the flowchart is the RFID code which will be covered in more detail in the RFID section on this appendix.

A. Code

The pharmacist GUI program was written in python language using the Tkinter package and the code is shown below.

```
#!/usr/bin/python

from Tkinter import *
from PIL import ImageTk, Image

   #if user presses the tab key the cursor
moves to the next text widget
   def focus_next_window(event):
      event.widget.tk_focusNext().focus()
      return("break")
```

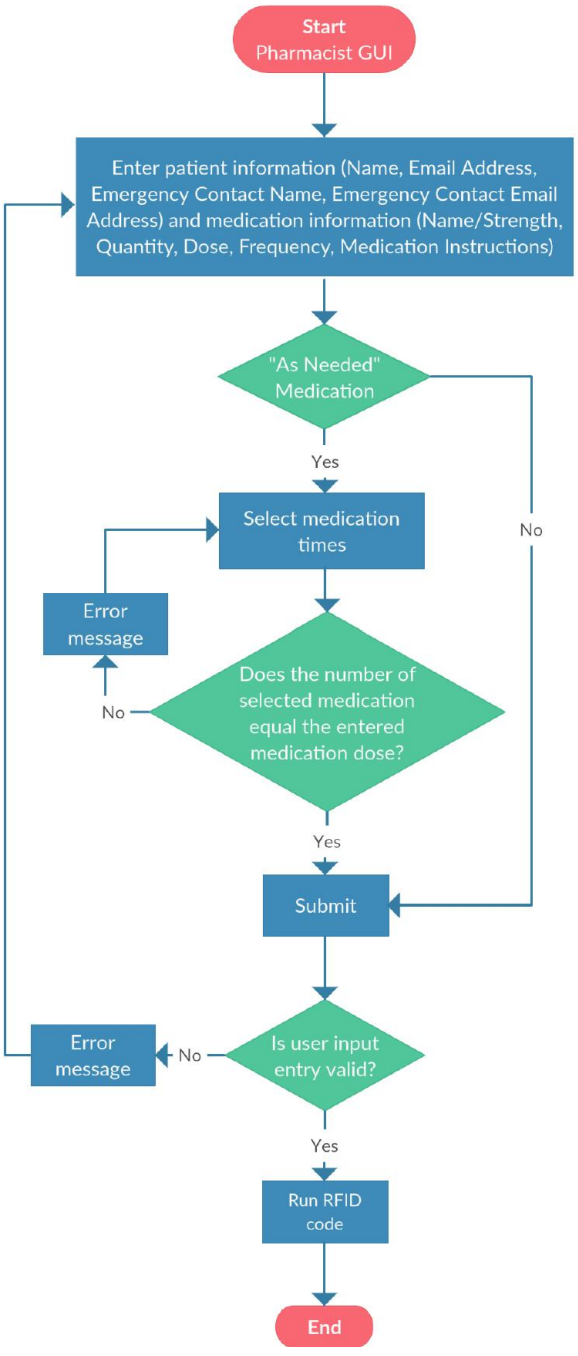


Figure 2. Pharmacist GUI

Flowchart

```
   #check if number of medication times entered and selected match
```

```
   def count():
      button_count = checkVar0.get() + checkVar1.get() + checkVar2.get() +
checkVar3.get() + checkVar4.get() + checkVar5.get() + checkVar6.get() +
checkVar7.get() + checkVar8.get() + checkVar9.get() + checkVar10.get() +
checkVar11.get() + checkVar12.get() + checkVar13.get()
      print button_count
      freq_string = frequency.get("1.0", 'end-1c')
      freq_int = int(freq_string)
      if button_count > freq_int:
            t1 = Toplevel(root)
            t1.configure(bg='white')
            t1.geometry("600x100")
            w = 600
            h = 100
            # get screen width and height
            ws = root.winfo_screenwidth()
            hs = root.winfo_screenheight()
            x = (ws/2) - (w/2)
            y = (hs/2) - (h/2)
            # set the dimensions of the screen and where it is placed
            t1.geometry('%dx%d+%d+%d' % (w, h, x, y))
            #t1.grab_set()                 #had to comment out on RPi
            t1.attributes("-topmost", True)
            Label(t1, text='Number of selected medication times does not
match entered medication frequency').pack(padx=10, pady=5)
            Label(t1, text='Please deselect one of the medication time or
change medication frequency to continue').pack(padx=10)

            #t1.grab_release()
            b1 = Button(t1, text="Ok", bg='white',
command=t1.destroy).pack(pady=10)

   #enables medication time checkboxes
   def enable():
      check0.config(state=NORMAL)
      check1.config(state=NORMAL)
      check2.config(state=NORMAL)
      check3.config(state=NORMAL)
      check4.config(state=NORMAL)
      check5.config(state=NORMAL)
      check6.config(state=NORMAL)
      check7.config(state=NORMAL)
      check8.config(state=NORMAL)
      check9.config(state=NORMAL)
      check10.config(state=NORMAL)
      check11.config(state=NORMAL)
      check12.config(state=NORMAL)
      check13.config(state=NORMAL)

   #disables medication time checkboxes
   def disable():
      check0.config(state=DISABLED)
      check1.config(state=DISABLED)
      check2.config(state=DISABLED)
      check3.config(state=DISABLED)
      check4.config(state=DISABLED)
      check5.config(state=DISABLED)
```

```
      check6.config(state=DISABLED)
      check7.config(state=DISABLED)
      check8.config(state=DISABLED)
      check9.config(state=DISABLED)
      check10.config(state=DISABLED)
      check11.config(state=DISABLED)
      check12.config(state=DISABLED)
      check13.config(state=DISABLED)

   def getData():
      #check if text widgets are empty
      if name.compare("end-1c", "==", "1.0"):
            t1 = Toplevel(root)
            t1.configure(bg='white')
            t1.geometry("400x100")
            w = 400
            h = 100
            # get screen width and height
            ws = root.winfo_screenwidth()
            hs = root.winfo_screenheight()
            x = (ws/2) - (w/2)
            y = (hs/2) - (h/2)
            # set the dimensions of the screen and where it is placed
            t1.geometry('%dx%d+%d+%d' % (w, h, x, y))
            t1.grab_set()
            t1.attributes("-topmost", True)
            Label(t1, text='Patient Name entry invalid',
bg='white').pack(padx=10, pady=5)
            Label(t1, text='Please complete this field to continue',
bg='white').pack(padx=10)

            #t1.grab_release()
            b1 = Button(t1, text="Ok", bg='white',
command=t1.destroy).pack(pady=5)
            return

      if name_addr.compare("end-1c", "==", "1.0"):
            t2 = Toplevel(root)
            t2.configure(bg='white')
            t2.geometry("400x100")
            w = 400
            h = 100
            # get screen width and height
            ws = root.winfo_screenwidth()
            hs = root.winfo_screenheight()
            x = (ws/2) - (w/2)
            y = (hs/2) - (h/2)
            # set the dimensions of the screen and where it is placed
            t2.geometry('%dx%d+%d+%d' % (w, h, x, y))
            t2.grab_set()
            t2.attributes("-topmost", True)
            Label(t2, text='Patient Name Email Address entry invalid',
bg='white').pack(padx=10, pady=5)
            Label(t2, text='Please complete this field to continue',
bg='white').pack(padx=10)

            #t1.grab_release()
```

```
            b1 = Button(t2, text="Ok", bg='white',
command=t2.destroy).pack(pady=5)
            return

      if emergency_name.compare("end-1c", "==", "1.0"):
            t3 = Toplevel(root)
            t3.configure(bg='white')
            t3.geometry("400x100")
            w = 400
            h = 100
            # get screen width and height
            ws = root.winfo_screenwidth()
            hs = root.winfo_screenheight()
            x = (ws/2) - (w/2)
            y = (hs/2) - (h/2)
            # set the dimensions of the screen and where it is placed
            t3.geometry('%dx%d+%d+%d' % (w, h, x, y))
            t3.grab_set()
            t3.attributes("-topmost", True)
            Label(t3, text='Emergency Contact Name entry invalid',
bg='white').pack(padx=10, pady=5)
            Label(t3, text='Please complete this field to continue',
bg='white').pack(padx=10)

            #t1.grab_release()
            b1 = Button(t3, text="Ok", bg='white',
command=t3.destroy).pack(pady=5)
            return

      if emergency_addr.compare("end-1c", "==", "1.0"):
            t4= Toplevel(root)
            t4.configure(bg='white')
            t4.geometry("400x100")
            w = 400
            h = 100
            # get screen width and height
            ws = root.winfo_screenwidth()
            hs = root.winfo_screenheight()
            x = (ws/2) - (w/2)
            y = (hs/2) - (h/2)
            # set the dimensions of the screen and where it is placed
            t4.geometry('%dx%d+%d+%d' % (w, h, x, y))
            t4.grab_set()
            t4.attributes("-topmost", True)
            Label(t4, text='Emergency Contact Email Address entry invalid',
bg='white').pack(padx=10, pady=5)
            Label(t4, text='Please complete this field to continue',
bg='white').pack(padx=10)

            #t1.grab_release()
            b1 = Button(t4, text="Ok", bg='white',
command=t4.destroy).pack(pady=5)
            return

      if med_name.compare("end-1c", "==", "1.0"):
            t5 = Toplevel(root)
            t5.configure(bg='white')
```

```
            t5.geometry("400x100")
            w = 400
            h = 100
            # get screen width and height
            ws = root.winfo_screenwidth()
            hs = root.winfo_screenheight()
            x = (ws/2) - (w/2)
            y = (hs/2) - (h/2)
            # set the dimensions of the screen and where it is placed
            t5.geometry('%dx%d+%d+%d' % (w, h, x, y))
            t5.grab_set()
            t5.attributes("-topmost", True)
            Label(t5, text='Medication Name entry invalid',
bg='white').pack(padx=10, pady=5)
            Label(t5, text='Please complete this field to continue',
bg='white').pack(padx=10)

            #t1.grab_release()
            b1 = Button(t5, text="Ok", bg='white',
command=t5.destroy).pack(pady=5)
            return

      if quantity.compare("end-1c", "==", "1.0"):
            t6 = Toplevel(root)
            t6.configure(bg='white')
            t6.geometry("400x100")
            w = 400
            h = 100
            # get screen width and height
            ws = root.winfo_screenwidth()
            hs = root.winfo_screenheight()
            x = (ws/2) - (w/2)
            y = (hs/2) - (h/2)
            # set the dimensions of the screen and where it is placed
            t6.geometry('%dx%d+%d+%d' % (w, h, x, y))
            t6.grab_set()
            t6.attributes("-topmost", True)
            Label(t6, text='Medication Quantity entry invalid',
bg='white').pack(padx=10, pady=5)
            Label(t6, text='Please complete this field to continue',
bg='white').pack(padx=10)

            #t1.grab_release()
            b1 = Button(t6, text="Ok", bg='white',
command=t6.destroy).pack(pady=5)
            return
      else:
            try:
                  quantity_string = quantity.get("1.0", 'end-1c')
                  quantity_int = int(quantity_string)
                  print 'true'
            except:
                  print 'false'
                  t7 = Toplevel(root)
                  t7.configure(bg='white')
                  t7.geometry("400x100")
                  w = 400
```

```
                  h = 100
                  # get screen width and height
                  ws = root.winfo_screenwidth()
                  hs = root.winfo_screenheight()
                  x = (ws/2) - (w/2)
                  y = (hs/2) - (h/2)
                  # set the dimensions of the screen and where it is placed
                  t7.geometry('%dx%d+%d+%d' % (w, h, x, y))
                  t7.grab_set()
                  t7.attributes("-topmost", True)
                  Label(t7, text='Medication Quantity entry invalid',
bg='white').pack(padx=10, pady=5)
                  Label(t7, text='Please enter in a valid number to
continue', bg='white').pack(padx=10)

                  #t1.grab_release()
                  b1 = Button(t7, text="Ok", bg='white',
command=t7.destroy).pack(pady=5)
                  return

      if dose.compare("end-1c", "==", "1.0"):
            t8 = Toplevel(root)
            t8.geometry("400x100")
            t8.configure(bg='white')›
            w = 400
            h = 100
            # get screen width and height
            ws = root.winfo_screenwidth()
            hs = root.winfo_screenheight()
            x = (ws/2) - (w/2)
            y = (hs/2) - (h/2)
            # set the dimensions of the screen and where it is placed
            t8.geometry('%dx%d+%d+%d' % (w, h, x, y))
            t8.grab_set()
            t8.attributes("-topmost", True)
            Label(t8, text='Medication Dose entry invalid',
bg='white').pack(padx=10, pady=5)
            Label(t8, text='Please complete this field to continue',
bg='white').pack(padx=10)

            #t1.grab_release()
            b1 = Button(t8, text="Ok", bg='white',
command=t8.destroy).pack(pady=5)
            return
      else:
            try:
                  dose_string = dose.get("1.0", 'end-1c')
                  dose_int = int(dose_string)
                  print 'true'
            except:
                  print 'false'
                  t9 = Toplevel(root)
                  t9.configure(bg='white')
                  t9.geometry("400x100")
                  w = 400
                  h = 100
                  # get screen width and height
```

```
                  ws = root.winfo_screenwidth()
                  hs = root.winfo_screenheight()
                  x = (ws/2) - (w/2)
                  y = (hs/2) - (h/2)
                  # set the dimensions of the screen and where it is placed
                  t9.geometry('%dx%d+%d+%d' % (w, h, x, y))
                  t9.grab_set()
                  t9.attributes("-topmost", True)
                  Label(t9, text='Medication Dose entry invalid',
bg='white').pack(padx=10, pady=5)
                  Label(t9, text='Please enter in a valid number to
continue', bg='white').pack(padx=10)

                  #t1.grab_release()
                  b1 = Button(t9, text="Ok", bg='white',
command=t9.destroy).pack(pady=5)
                  return

      if frequency.compare("end-1c", "==", "1.0"):
            t10 = Toplevel(root)
            t10.configure(bg='white')
            t10.geometry("400x100")
            w = 400
            h = 100
            # get screen width and height
            ws = root.winfo_screenwidth()
            hs = root.winfo_screenheight()
            x = (ws/2) - (w/2)
            y = (hs/2) - (h/2)
            # set the dimensions of the screen and where it is placed
            t10.geometry('%dx%d+%d+%d' % (w, h, x, y))
            t10.grab_set()
            t10.attributes("-topmost", True)
            Label(t10, text='Medication Frequency entry invalid',
bg='white').pack(padx=10, pady=5)
            Label(t10, text='Please complete this field to continue',
bg='white').pack(padx=10)

            #t1.grab_release()
            b1 = Button(t10, text="Ok", bg='white',
command=t10.destroy).pack(pady=5)
            return
      else:
            try:
                  f_string = frequency.get("1.0", 'end-1c')
                  f_int = int(f_string)
                  print 'true'
            except:
                  print 'false'
                  t11 = Toplevel(root)
                  t11.geometry("400x100")
                  w = 400
                  h = 100
                  # get screen width and height
                  ws = root.winfo_screenwidth()
                  hs = root.winfo_screenheight()
                  x = (ws/2) - (w/2)
```

```
                  y = (hs/2) - (h/2)
                  # set the dimensions of the screen and where it is placed
                  t11.geometry('%dx%d+%d+%d' % (w, h, x, y))
                  t11.grab_set()
                  t11.attributes("-topmost", True)
                  Label(t11, text='Medication Quantity entry invalid',
bg='white').pack(padx=10, pady=5)
                  Label(t11, text='Please enter in a valid number to
continue', bg='white').pack(padx=10)

                  #t1.grab_release()
                  b1 = Button(t11, text="Ok", bg='white',
command=t11.destroy).pack(pady=5)
                  return

      if var_ans.get() == 0:
            t12 = Toplevel(root)
            t12.configure(bg='white')
            t12.geometry("400x100")
            w = 400
            h = 100
            # get screen width and height
            ws = root.winfo_screenwidth()
            hs = root.winfo_screenheight()
            x = (ws/2) - (w/2)
            y = (hs/2) - (h/2)
            # set the dimensions of the screen and where it is placed
            t12.geometry('%dx%d+%d+%d' % (w, h, x, y))
            t12.grab_set()
            t12.attributes("-topmost", True)
            Label(t12, text='Is this an AS NEEDED medication?',
bg='white').pack(padx=10, pady=5)
            Label(t12, text='Please provide an answer on the medication
form', bg='white').pack(padx=10)

            #t1.grab_release()
            b1 = Button(t12, text="Ok", bg='white',
command=t12.destroy).pack(pady=5)
            return

      button_count = checkVar0.get() + checkVar1.get() + checkVar2.get() +
checkVar3.get() + checkVar4.get() + checkVar5.get() + checkVar6.get() +
checkVar7.get() + checkVar8.get() + checkVar9.get() + checkVar10.get() +
checkVar11.get() + checkVar12.get() + checkVar13.get()
      print button_count
      freq_string = frequency.get("1.0", 'end-1c')
      freq_int = int(freq_string)
      print freq_int

      if var_ans.get() == 2 and button_count != freq_int:
            t13 = Toplevel(root)
            t13.configure(bg='white')
            t13.geometry("600x100")
            w = 600
            h = 100
            # get screen width and height
            ws = root.winfo_screenwidth()
```

```
            hs = root.winfo_screenheight()
            x = (ws/2) - (w/2)
            y = (hs/2) - (h/2)
            # set the dimensions of the screen and where it is placed
            t13.geometry('%dx%d+%d+%d' % (w, h, x, y))
            t13.grab_set()
            t13.attributes("-topmost", True)
            Label(t13, text='Number of selected medication times does not
match entered medication frequency', bg='white').pack(padx=10, pady=5)
            Label(t13, text='Please change medication time or medication
frequency to continue', bg='white').pack(padx=10)

            #t1.grab_release()
            b1 = Button(t13, text="Ok", bg='white',
command=t13.destroy).pack(pady=10)
            return
      else:
            file1 = open("Data.txt", "w+")

            na = ("%s#" % name.get("1.0", 'end-1c'))
            addr = ("%s#" % name_addr.get("1.0", 'end-1c'))
            ename = ("%s#" % emergency_name.get("1.0", 'end-1c'))
            eaddr= ("%s#" % emergency_addr.get("1.0", 'end-1c'))
            mname = ("%s#" % med_name.get("1.0", 'end-1c'))
            quant = ("%s#" % quantity.get("1.0", 'end-1c'))
            f = ("%s#" % frequency.get("1.0", 'end-1c'))
            d = ("%s#" % dose.get("1.0", 'end-1c'))
            nt = ("%s#" % notes.get("1.0", 'end-1c'))

            file1.write("%s\n" % na)
            file1.write("%s\n" % addr)
            file1.write("%s\n" % ename)
            file1.write("%s\n" % eaddr)
            file1.write("%s\n" % mname)
            file1.write("%s\n" % quant)
            file1.write("%s\n" % f)
            file1.write("%s\n" % d)
            file1.write("%s\n" % nt)

            if var_ans.get() == 1:
                  file1.write("As needed#\n")
            if var_ans.get() == 2:
                  file1.write("Schedule#\n")

            #checkbox selected --> checkbox state = 1; checkbox unselected --
> checkbox state = 0
            if checkVar0.get() == 1:
                  file1.write("7#\n")
            if checkVar1.get() == 1:
                  file1.write("8#\n")
            if checkVar2.get() == 1:
                  file1.write("9#\n")
            if checkVar3.get() == 1:
                  file1.write("10#\n")
            if checkVar4.get() == 1:
                  file1.write("11#\n")
            if checkVar5.get() == 1:
```

```
                file1.write("12#\n")
            if checkVar6.get() == 1:
                file1.write("13#\n")
            if checkVar7.get() == 1:
                file1.write("14#\n")
            if checkVar8.get() == 1:
                file1.write("15#\n")
            if checkVar9.get() == 1:
                file1.write("16#\n")
            if checkVar10.get() == 1:
                file1.write("17#\n")
            if checkVar11.get() == 1:
                file1.write("18#\n")
            if checkVar12.get() == 1:
                file1.write("19#\n")
            if checkVar13.get() == 1:
                file1.write("20#\n")

            file1.write("!")
            file1.close()
            root.destroy()

   root = Tk()
   root.configure(bg='white')
   root.geometry("700x680")
   w = 700
   h = 680
   # get screen width and height
   ws = root.winfo_screenwidth()
   hs = root.winfo_screenheight()
   x = (ws/2) - (w/2)
   y = (hs/2) - (h/2)
   # set the dimensions of the screen and where it is placed
   root.geometry('%dx%d+%d+%d' % (w, h, x, y))
   root.title('DispenSUM Medication Form')

   frame1 = Frame(root, bg='white')

   heading = Label(frame1, text="Patient Medication Form                ",
font="None 18", bg='white')
   heading.grid(row=0, column=0, sticky=S)

   temp = Image.open("image.jpg")
   temp = temp.save("image.ppm", "ppm")
   image = PhotoImage(file = "image.ppm")
   imagepanel = Label(frame1, image=image, bg='white')
   imagepanel.grid(row=0, column=1, sticky=E)

   frame1.grid_columnconfigure(0, weight=1)
   frame1.grid_columnconfigure(1, weight=1)

   frame2 = Frame(root, bg='SteelBlue3', width=85)

   subheading1 = Label(frame2, text="PATIENT INFORMATION", font="None 12
bold", bg='SteelBlue3', fg='white')
   subheading1.pack()
```

```
   frame3 = Frame(root, bg='white')

   Label(frame3, text="Name:", width=35, anchor=W, bg='white').grid(row=0,
column=0, sticky=E)
   name = Text(frame3, bd=3, width=50, height=1, relief=SUNKEN, bg='white')
   name.grid(row=0, column=1, sticky=W)
   name.bind("<Tab>", focus_next_window)

   Label(frame3, text="Email Address:", width=35, anchor=W,
bg='white').grid(row=1, column=0, sticky=E, pady=10)
   name_addr = Text(frame3, bd=3, width=50, height=1, relief=SUNKEN,
bg='white')
   name_addr.grid(row=1, column=1, sticky=W)
   name_addr.bind("<Tab>", focus_next_window)

   Label(frame3, text="Emergency Contact Name:", width=35, anchor=W,
bg='white').grid(row=2, column=0, sticky=E)
   emergency_name = Text(frame3, bd=3, width=50, height=1, relief=SUNKEN,
bg='white')
   emergency_name.grid(row=2, column=1, sticky=W)
   emergency_name.bind("<Tab>", focus_next_window)

   Label(frame3, text="Emergency Contact Email Address:", width=35, anchor=W,
bg='white').grid(row=3, column=0, sticky=E, pady=10)
   emergency_addr = Text(frame3, bd=3, width=50, height=1, relief=SUNKEN,
bg='white')
   emergency_addr.grid(row=3, column=1, sticky=W)
   emergency_addr.bind("<Tab>", focus_next_window)

   frame3.grid_columnconfigure(0, weight=1)
   frame3.grid_columnconfigure(1, weight=1)

   frame4 = Frame(root, bg='SteelBlue3')

   subheading2 = Label(frame4, text="MEDICATION INFORMATION", font="None, 12
bold", bg='SteelBlue3', fg='white')
   subheading2.pack()

   frame5 = Frame(root, bg='white')

   Label(frame5, text="Name/Strength:", width=35, anchor=W,
bg='white').grid(row=0, column=0, sticky=E)
   med_name = Text(frame5, bd=3, width=50, height=1, relief=SUNKEN,
bg='white')
   med_name.grid(row=0, column=1, sticky=W)
   med_name.bind("<Tab>", focus_next_window)

   Label(frame5, text="Quantity:", width=35, anchor=W,
bg='white').grid(row=1, column=0, sticky=E, pady=10)
   quantity = Text(frame5, bd=3, width=50, height=1, relief=SUNKEN,
bg='white')
   quantity.grid(row=1, column=1, sticky=W)
   quantity.bind("<Tab>", focus_next_window)

   Label(frame5, text="Dose (number of pills taken each time):", width=35,
anchor=W, bg='white').grid(row=2, column=0, sticky=E)
   dose = Text(frame5, bd=3, width=50, height=1, relief=SUNKEN, bg='white')
```

```
   dose.grid(row=2, column=1, sticky=W)
   dose.bind("<Tab>", focus_next_window)

   Label(frame5, text="Frequency (number of times taken per day):", width=35,
anchor=W, bg='white').grid(row=3, column=0, sticky=E, pady=10)
   frequency = Text(frame5, bd=3, width=50, height=1, relief=SUNKEN,
bg='white')
   frequency.grid(row=3, column=1, sticky=W)
   frequency.bind("<Tab>", focus_next_window)

   Label(frame5, text="Medication Instructions:", width=35, anchor=W,
bg='white').grid(row=4, column=0, sticky=E)
   notes = Text(frame5, bd=3, width=50, height=4, relief=SUNKEN, bg='white')
   notes.grid(row=4, column=1, sticky=W)
   notes.bind("<Tab>", focus_next_window)

   frame5.grid_columnconfigure(0, weight=1)
   frame5.grid_columnconfigure(1, weight=1)


   frame6 = Frame(root, bg='white')

   Label(frame6, text="Is this an AS NEEDED medication?", anchor=W,
bg='white').grid(row=0, column=0, columnspan=2, sticky=W, pady=10)
   var_ans = IntVar()
   Radiobutton(frame6, text="Yes", variable=var_ans, value=1, bg='white',
highlightbackground='white', command=disable).grid(row=0, column=2, padx=10,
sticky=W)
   Radiobutton(frame6, text="No", variable=var_ans, value=2, bg='white',
highlightbackground='white', command=enable).grid(row=0, column=3, sticky=W)

   Label(frame6, text="Medication Times:", anchor=W, width=20,
bg='white').grid(row=1, column=0, sticky=E)
   checkVar0 = IntVar()
   check0 = Checkbutton(frame6, text="07:00", variable=checkVar0, bg='white',
highlightbackground='white', command=count)
   check0.grid(row=1, column=1, sticky=W)
   checkVar1 = IntVar()
   check1 = Checkbutton(frame6, text="08:00", variable=checkVar1, bg='white',
highlightbackground='white', command=count)
   check1.grid(row=1, column=2, padx=9, sticky=W)
   checkVar2 = IntVar()
   check2 = Checkbutton(frame6, text="09:00", variable=checkVar2, bg='white',
highlightbackground='white', command=count)
   check2.grid(row=1, column=3, sticky=W)
   checkVar3 = IntVar()
   check3 = Checkbutton(frame6, text="10:00", variable=checkVar3, bg='white',
highlightbackground='white', command=count)
   check3.grid(row=1, column=4, padx=9, sticky=W)
   checkVar4 = IntVar()
   check4 = Checkbutton(frame6, text="11:00", variable=checkVar4, bg='white',
highlightbackground='white', command=count)
   check4.grid(row=1, column=5, sticky=W)
   checkVar5 = IntVar()
   check5 = Checkbutton(frame6, text="12:00", variable=checkVar5, bg='white',
highlightbackground='white', command=count)
   check5.grid(row=1, column=6, padx=9, sticky=W)
```

```
   checkVar6 = IntVar()
   check6 = Checkbutton(frame6, text="13:00", variable=checkVar6, bg='white',
highlightbackground='white', command=count)
   check6.grid(row=1, column=7, sticky=W)
   checkVar7 = IntVar()
   check7 = Checkbutton(frame6, text="14:00", variable=checkVar7, bg='white',
highlightbackground='white', command=count)
   check7.grid(row=2, column=1, sticky=W)
   checkVar8 = IntVar()
   check8 = Checkbutton(frame6, text="15:00", variable=checkVar8, bg='white',
highlightbackground='white', command=count)
   check8.grid(row=2, column=2, padx=9, sticky=W)
   checkVar9 = IntVar()
   check9 = Checkbutton(frame6, text="16:00", variable=checkVar9, bg='white',
highlightbackground='white', command=count)
   check9.grid(row=2, column=3, sticky=W)
   checkVar10 = IntVar()
   check10 = Checkbutton(frame6, text="17:00", variable=checkVar10,
bg='white', highlightbackground='white', command=count)
   check10.grid(row=2, column=4, padx=9, sticky=W)
   checkVar11 = IntVar()
   check11 = Checkbutton(frame6, text="18:00", variable=checkVar11,
bg='white', highlightbackground='white', command=count)
   check11.grid(row=2, column=5, sticky=W)
   checkVar12 = IntVar()
   check12 = Checkbutton(frame6, text="19:00", variable=checkVar12,
bg='white',highlightbackground='white', command=count)
   check12.grid(row=2, column=6, padx=9, sticky=W)
   checkVar13 = IntVar()
   check13 = Checkbutton(frame6, text="20:00", variable=checkVar13,
bg='white', highlightbackground='white', command=count)
   check13.grid(row=2, column=7, sticky=W)

   frame6.grid_columnconfigure(0, weight=1)
   frame6.grid_columnconfigure(1, weight=1)
   frame6.grid_columnconfigure(2, weight=1)
   frame6.grid_columnconfigure(3, weight=1)
   frame6.grid_columnconfigure(4, weight=1)
   frame6.grid_columnconfigure(5, weight=1)
   frame6.grid_columnconfigure(6, weight=1)
   frame6.grid_columnconfigure(7, weight=1)

   frame7 = Frame(root, bg='white')

   b1 = Button(frame7,text="Submit", command=getData, bg='white')
   b2 = Button(frame7,text="Cancel", command=root.destroy, bg='white')
   b1.pack(side=LEFT,padx=10, pady=10)
   b2.pack(side=LEFT,padx=10, pady=10)

   frame1.pack()
   frame2.pack(pady=2, fill=BOTH)
   frame3.pack(padx=5)
   frame4.pack(pady=2, fill=BOTH)
   frame5.pack(padx=5)
   frame6.pack(pady=5, padx=5)
   frame7.pack()
```

```
root.mainloop()
```

Figure 2. Pharmacist GUI Code

B. Testing

The GUI features multiple entry fields, checkboxes, and buttons. Since the purpose of the GUI is to create a text file containing all the data entered, every component was tested for correct functionality. When testing each text field everything worked as expected. All the dummy text entered into the text field appeared in the text file that was created (refer to Figure 1 and Figure 2).

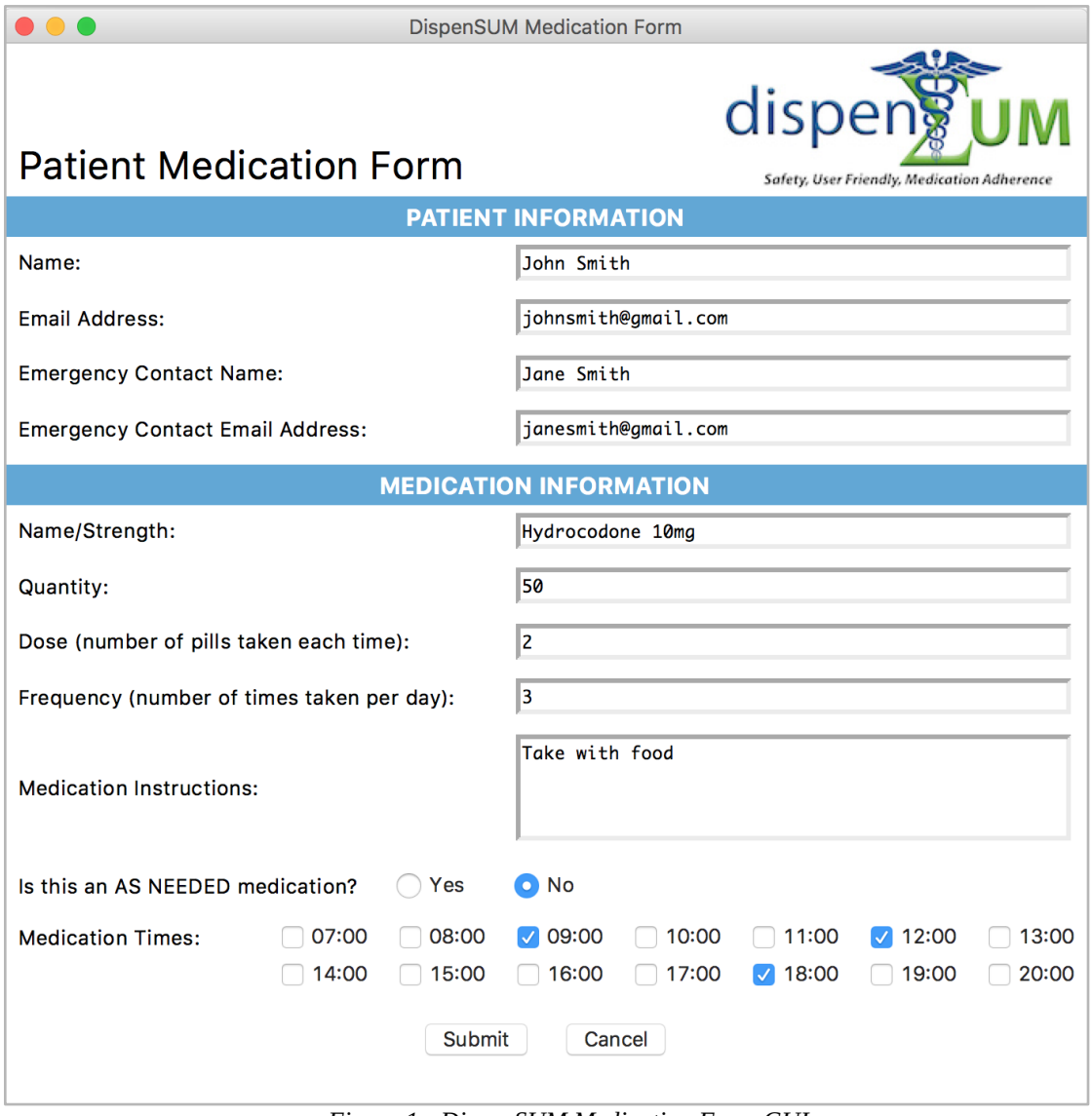


Figure 1 - DispenSUM Medication Form GUI

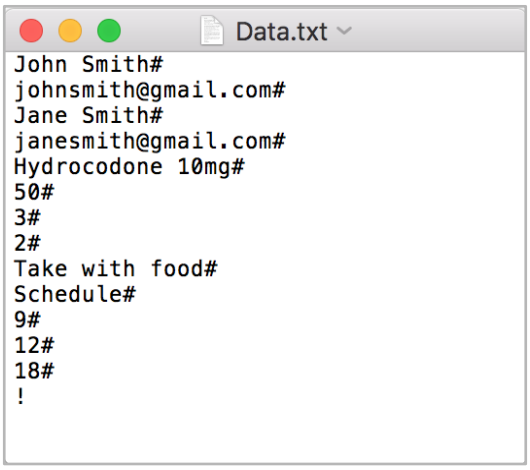


Figure 2 - ASCII Text File generated from DispenSUM Medication Form GUI

In this process the button for submit was verified to be working correctly as this button is used to start the function that creates a text file and writes to it. Also testing was run on empty text fields and the result was that if there was a empty text field and the user pressed the submit button it would not create a text file and write data to it. Instead a window would pop up notifying the user that they need to complete the empty text field. Testing was completed on the checkboxes by adding a line of code which would print the state value of the checkbox on the command line of terminal. With the GUI program running all the checkboxes were selected and deselected one at a time. Each time a checkbox was checked a '1' was printed on the command line and when the checkbox was unchecked a '0' was printed in the command line (refer to Fig 3 and Fig 4). This result verified that the checkboxes function correctly as the state of the checkbox is 1 for checked and 0 for unchecked.

```
210
211         # testing checkboxes
212         print checkVar0.get()
213         print checkVar1.get()
214         print checkVar2.get()
215         print checkVar3.get()
216         print checkVar4.get()
217         print checkVar5.get()
218         print checkVar6.get()
219         print checkVar7.get()
220         print checkVar8.get()
221         print checkVar9.get()
222         print checkVar10.get()
223         print checkVar11.get()
224         print checkVar12.get()
225         print checkVar13.get()
226
```

Figure 3 - DispenSUM Medication Form GUI code for testing checkboxes

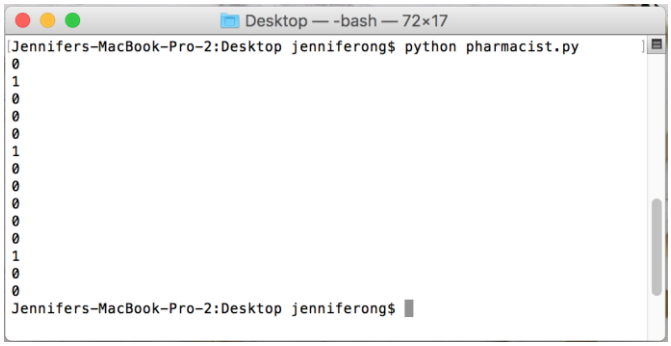


Figure 4 - DispenSUM Medication Form GUI command line output

Initially the GUI program was tested on a computer running on a Mac OS however when the RFID code was added it was tested on a Linux OS because the RFID code that is being used can only function on a Linux OS. When running the GUI program on a Linux OS The appearance of this graphical user interface varies slightly when run on different operating systems.

III. RFID

For memory connection between the pharmacist GUI and User GUI we used an RFID system. We used two Mifare rc522 RFID read writeres. The first one was attached to an arduino uno. This system used used a modified version of the demo code to write. Changed that were made allowed us to access all 64 sections. Couple of limitations with this system is that we are required to have a termination character while the information is being sent through because of this that special character can not be used for any other purposes, this can be seen in figure 2

The second thing is that there is a character limit of 28 characters for each line that can be sent though. Using a python equivalent version of the code was used to then transfer the information from the RFID tag to the RPI.

A. Code

UNO CODE:

```
#include <SPI.h>

#include <MFRC522.h>


#define RST_PIN         9          // Configurable, see typical pin layout above

#define SS_PIN          10         // Configurable, see typical pin layout above


MFRC522 mfrc522(SS_PIN, RST_PIN);   // Create MFRC522 instance


void setup() {

        Serial.begin(9600);        // Initialize serial communications with the PC

        SPI.begin();               // Init SPI bus

        mfrc522.PCD_Init();        // Init MFRC522 card

        Serial.println(F("Write personal data on a MIFARE PICC "));

}


void loop() {



        //int k = 9;

        // Prepare key - all keys are set to FFFFFFFFFFFFh at chip delivery from the factory.


        MFRC522::MIFARE_Key key;

        for (byte i = 0; i < 6; i++) key.keyByte[i] = 0xFF;


        // Look for new cards

        if ( ! mfrc522.PICC_IsNewCardPresent()) {

                return;
```

```
}

// Select one of the cards
if ( ! mfrc522.PICC_ReadCardSerial())    return;

Serial.print(F("Card UID:"));    //Dump UID
for (byte i = 0; i < mfrc522.uid.size; i++) {
  Serial.print(mfrc522.uid.uidByte[i] < 0x10 ? " 0" : " ");
  Serial.print(mfrc522.uid.uidByte[i], HEX);
}
Serial.print(F(" PICC type: "));   // Dump PICC type
MFRC522::PICC_Type piccType = mfrc522.PICC_GetType(mfrc522.uid.sak);
Serial.println(mfrc522.PICC_GetTypeName(piccType));

byte buffer[34];
byte block;
MFRC522::StatusCode status;
byte len;
int chip[] =
{1,2,4,5,6,8,9,10,12,13,14,16,17,18,20,21,22,24,25,26,28,29,30,32,33,34,36,37,38,40,41,42,44,45,46,48,49,50,52,53
,54,56,57,58,59,60,61,62};
int j;
for(j = 0; j < 48; j++)
{
Serial.setTimeout(2000000L) ;     // wait until 20 seconds for input from serial
// Ask personal data: Family name
//Serial.println(j);

Serial.println(F("Type Family name, ending with #"));
len=Serial.readBytesUntil('#', (char *) buffer, 30) ; // read family name from serial
for (byte i = len; i < 30; i++) buffer[i] = ' ';     // pad with spaces
```

```
      //Serial.println(j);

      /*if(j == 3 || j == 7 || j ==11 || j ==15 ||j == 19 ||j == 23 ||j == 27 ||j == 31 ||j == 35 ||j == 39 ||j == 43 ||j == 47 ||j
== 51 ||j == 55 ||j == 59)

      {

        Serial.println("Entered IF");

        j++;

        //Serial.println("New J:" j);

      }*/


      block = chip[j];

      //Serial.println(j);

      //Serial.println(F("Authenticating using key A..."));

      status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, block, &key,
&(mfrc522.uid));

      if (status != MFRC522::STATUS_OK) {

        Serial.print(F("PCD_Authenticate() failed: "));

        Serial.println(mfrc522.GetStatusCodeName(status));

        return;

      }

      else Serial.println(F("PCD_Authenticate() success: "));


      // Write block

      status = mfrc522.MIFARE_Write(block, buffer, 16);

      if (status != MFRC522::STATUS_OK) {

        Serial.print(F("MIFARE_Write() failed: "));

        Serial.println(mfrc522.GetStatusCodeName(status));

          return;

      }

      else Serial.println(F("MIFARE_Write() success: "));

      Serial.println(j);

      j++;
```

```
      Serial.println(j);

      block = chip[j];

      //Serial.println(F("Authenticating using key A..."));

      status = mfrc522.PCD_Authenticate(MFRC522::PICC_CMD_MF_AUTH_KEY_A, block, &key,
&(mfrc522.uid));

      if (status != MFRC522::STATUS_OK) {

        Serial.print(F("PCD_Authenticate() failed: "));

        Serial.println(mfrc522.GetStatusCodeName(status));

        return;

      }


      // Write block

      status = mfrc522.MIFARE_Write(block, &buffer[16], 16);

      if (status != MFRC522::STATUS_OK) {

      Serial.print(F("MIFARE_Write() failed: "));

      Serial.println(mfrc522.GetStatusCodeName(status));

         return;

       }

      else Serial.println(F("MIFARE_Write() success: "));

      }

      Serial.println(" ");

      mfrc522.PICC_HaltA(); // Halt PICC

      mfrc522.PCD_StopCrypto1();  // Stop encryption on PCD


}
```

Raspberry Pi RFID System

```
!/usr/bin/env python
# -*- coding: utf8 -*-

import RPi.GPIO as GPIO
import MFRC522
import signal
```

```
import sys
import time
import os

#Create File for Cartridge
orig_stdout = sys.stdout
i = 0
while os.path.exists("Transfered%s.txt" %i):
      i +=1
file = open("Transfered%s.txt" %i,"w")
sys.stdout = file

continue_reading = True

# Capture SIGINT for cleanup when the script is aborted
def end_read(signal,frame):
    global continue_reading
    #print "Ctrl+C captured, ending read."
    #write to file
    #file.write("Ctrl+C captured, ending read.")
    continue_reading = False
    GPIO.cleanup()

# Hook the SIGINT
signal.signal(signal.SIGINT, end_read)

# Create an object of the class MFRC522
MIFAREReader = MFRC522.MFRC522()

# This loop keeps checking for chips. If one is near it will get the UID
and authenticate
while continue_reading:

    # Scan for cards
    (status,TagType) =
MIFAREReader.MFRC522_Request(MIFAREReader.PICC_REQIDL)

    # If a card is found
    if status == MIFAREReader.MI_OK:
        #print "Card detected"
        #Another Write
        #file.write("Card detected")

    # Get the UID of the card
    (status,uid) = MIFAREReader.MFRC522_Anticoll()

    # If we have the UID, continue
    if status == MIFAREReader.MI_OK:

        # Print UID
        print str(uid[0])+str(uid[1])+str(uid[2])+str(uid[3])
        # Write UID
        #file.write "Card read UID:
"+str(uid[0])+","+str(uid[1])+","+str(uid[2])+","+str(uid[3])
```

```
        # This is the default key for authentication
        key = [0xFF,0xFF,0xFF,0xFF,0xFF,0xFF]

        # Select the scanned tag
        MIFAREReader.MFRC522_SelectTag(uid)

        # Dump the data
        MIFAREReader.MFRC522_DumpClassic1K(key, uid)

        # Stop
        MIFAREReader.MFRC522_StopCrypto1()

        time.sleep(2)
    file.close()
    sys.exit()
```

B. Testing

Results yield a success when a tag isn't bricked.

IV. User/Interaction GUI

The user/interaction GUI runs on the Raspberry Pi and is displayed on the attached touch screen. Below in Figure 5 is a flowchart showing the different functions that the program executes and how the functions are connected to one another. The RFID part of the user/interaction GUI is discussed in detail in the RFID section of this appendix.

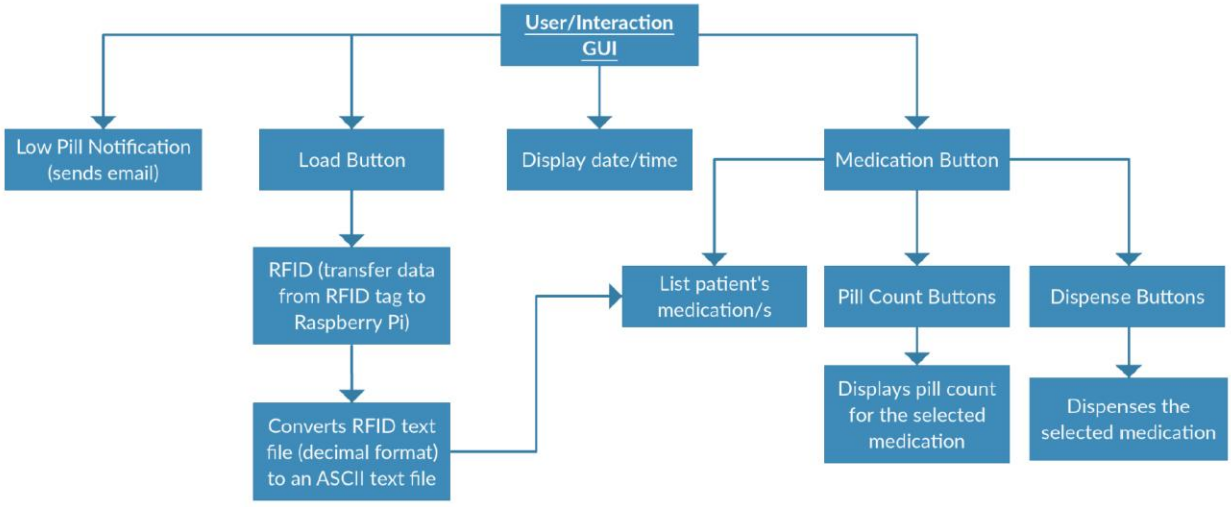


Figure 5 User/Interaction GUI Flowchart

A. Code

The user/interaction GUI program was written in python language using the Tkinter package and the code is shown below.

Note: One additional feature that it yet to be added into the program is a window that displays the name of the medication that is being dispensed.

```
from Tkinter import *
from PIL import ImageTk, Image
import time
import smtplib
import RPi.GPIO as GPIO
import MFRC522
import signal
import os
import serial

#Status2Reg     = 0x08

#SET UPS FOR GPIO PINS
GPIO.setmode(GPIO.BOARD)
GPIO.setup(11,GPIO.IN)
input = GPIO.input(11)

#Set Up for Time
localtime = time.localtime()
hour = localtime.tm_hour
minute = localtime.tm_min
TimerVal = None
TimeValTrig = None

#Here we will be listing all variables needed for the system
#UID# is the UID
#T# is the Time Stamp
#P# will be the pill count is the system
#D# how many times the pill will be dispensed
#Feq# is how many to allow a dispense in a time window.
#LDS: this will allow us to keep track if a cartridge is loading into the
system or not
#TimerST this will let us know if we have a time going

#~~~~~~~~~~~~~~~~~~~~~~~~~~Variable
Declarations~~~~~~~~~~~~~~~~~~~~~~~~~~#

email = None
TimerST = False

LDS1 = False
LDS2 = False
LDS3 = False
LDS4 = False
LDS5 = False
LDS6 = False
LDS7 = False
LDS8 = False

UID1 = None
UID2 = None
UID3 = None
UID4 = None
UID5 = None
UID6 = None
UID7 = None
UID8 = None
```

```
P1 = None
P2 = None
P3 = None
P4 = None
P5 = None
P6 = None
P7 = None
P8 = None

Feq1 = None
Feq2 = None
Feq3 = None
Feq4 = None
Feq5 = None
Feq6 = None
Feq7 = None
Feq8 = None

D1 = None
D2 = None
D3 = None
D4 = None
D5 = None
D6 = None
D7 = None
D8 = None

T1 = None
T2 = None
T3 = None
T4 = None
T5 = None
T6 = None
T7 = None
T8 = None

#~~~~~~~~~~~~~~~~~~~~~~~~~~~PROGRAM
DEFINITIONS~~~~~~~~~~~~~~~~~~~~~~~~~~~#
def time():
    global time1, date1
    # get the current local time from the PC
    time2 = time.strftime('%I:%M')
    pm_txt = "%s" % (time.strftime('%p'))
    date_txt = "%s, %s %s, %s" % (time.strftime('%A'),
time.strftime('%B'), time.strftime('%d'), time.strftime('%Y'))

   # if time string has changed, update it
    if time2 != time1:
        time1 = time2
        clock.config(text=time2)
        pm.config(text=pm_txt)
        date.config(text=date_txt)
    # calls itself every 200 milliseconds to update the time display as
needed
    clock.after(200, tick)
```

```
def make_loadbutton():
    b = Button(frame4, compound=TOP, width=75, height=100, text="LOAD",
font="helvetica', 12", bg='white', command=load_state)
    image = ImageTk.PhotoImage(file="settings.jpg")
    b.config(image=image)
    b.image = image
    b.pack(side=LEFT, padx=5)

def make_medbutton():
    b = Button(frame4, compound=TOP, width=75, height=100,
text="Medication", font="helvetica, 12", bg='white', command=med_window)
    image = ImageTk.PhotoImage(file="medication.jpg")
    b.config(image=image)
    b.image = image
    b.pack(side=RIGHT, padx=5)

def As_Need1():
    usbCOM = serial.Serial('/dev/ttyACM0', 9600)
    usbCOM.close()
    usbCOM.open()
    usbCOM.write('01')
    usbCOM.close()
    file = open("./count1.txt", "r")
    pill_num = file.readline()
    file.close()
    P1 = int(pill_num)
    P1 = P1 - 1
    writeto = open("./count1.txt", "w+")
    writeto.write('%d' % P1)

def As_Need2():
    usbCOM = serial.Serial('/dev/ttyACM0', 9600)
    usbCOM.close()
    usbCOM.open()
    usbCOM.write('02')
    usbCOM.close()
    file = open("./count2.txt", "r")
    pill_num = file.readline()
    file.close()
    P2 = int(pill_num)
    P2 = P2 - 1
    writeto = open("./count2.txt", "w+")
    writeto.write('%d' % P2)

def As_Need3():
    usbCOM = serial.Serial('/dev/ttyACM0', 9600)
    usbCOM.close()
    usbCOM.open()
    usbCOM.write('03')
    usbCOM.close()
    file = open("./count3.txt", "r")
    pill_num = file.readline()
    file.close()
    P3 = int(pill_num)
    P3 = P3 - 1
    writeto = open("./count3.txt", "w+")
    writeto.write('%d' % P3)
```

```
def As_Need4():
    usbCOM = serial.Serial('/dev/ttyACM0', 9600)
    usbCOM.close()
    usbCOM.open()
    usbCOM.write('04')
    usbCOM.close()
    file = open("./count4.txt", "r")
    pill_num = file.readline()
    file.close()
    P4 = int(pill_num)
    P4 = P4 - 1
    writeto = open("./count4.txt", "w+")
    writeto.write('%d' % P4)

def As_Need5():
    usbCOM = serial.Serial('/dev/ttyACM0', 9600)
    usbCOM.close()
    usbCOM.open()
    usbCOM.write('05')
    usbCOM.close()
    file = open("./count5.txt", "r")
    pill_num = file.readline()
    file.close()
    P5 = int(pill_num)
    P5 = P5 - 1
    writeto = open("./count5.txt", "w+")
    writeto.write('%d' % P5)

def As_Need6():
    usbCOM = serial.Serial('/dev/ttyACM0', 9600)
    usbCOM.close()
    usbCOM.open()
    usbCOM.write('06')
    usbCOM.close()
    file = open("./count6.txt", "r")
    pill_num = file.readline()
    file.close()
    P6 = int(pill_num)
    P6 = P6 - 1
    writeto = open("./count6.txt", "w+")
    writeto.write('%d' % P6)

def As_Need7():
    usbCOM = serial.Serial('/dev/ttyACM0', 9600)
    usbCOM.close()
    usbCOM.open()
    usbCOM.write('07')
    usbCOM.close()
    file = open("./count7.txt", "r")
    pill_num = file.readline()
    file.close()
    P7 = int(pill_num)
    P7 = P7 - 1
    writeto = open("./count7.txt", "w+")
    writeto.write('%d' % P7)
```

```
def As_Need8():
    usbCOM = serial.Serial('/dev/ttyACM0', 9600)
    usbCOM.close()
    usbCOM.open()
    usbCOM.write('08')
    usbCOM.close()
    file = open("./count8.txt", "r")
    pill_num = file.readline()
    file.close()
    P8 = int(pill_num)
    P8 = P8 - 1
    writeto = open("./count8.txt", "w+")
    writeto.write('%d' % P8)

def pill_count1():
    t = Toplevel(root)
    t.configure(bg='white')
    t.geometry("800x480")
    t.overrideredirect(True)
    w = 800
    h = 480
    # get screen width and height
    ws = root.winfo_screenwidth()
    hs = root.winfo_screenheight()
    x = (ws/2) - (w/2)
    y = (hs/2) - (h/2)
    # set the dimensions of the screen and where it is placed
    t.geometry('%dx%d+%d+%d' % (w, h, x, y))
    #t.grab_set()

    file = open("./count1.txt", "r")
    pill_num = file.readline()
    file.close()

    msg = ("Pill Count: " + str(pill_num))

    Label(t, text=msg, font="helvetica 22", bg='white').pack(pady=200)
    b1 = Button(t, text="Back", bg='white', font="helvetica 12",
command=t.destroy).pack()

def pill_count2():
    t = Toplevel(root)
    t.configure(bg='white')
    t.geometry("800x480")
    t.overrideredirect(True)
    w = 800
    h = 480
    # get screen width and height
    ws = root.winfo_screenwidth()
    hs = root.winfo_screenheight()
    x = (ws/2) - (w/2)
    y = (hs/2) - (h/2)
    # set the dimensions of the screen and where it is placed
    t.geometry('%dx%d+%d+%d' % (w, h, x, y))
    #t.grab_set()

    file = open("./count2.txt", "r")
```

```
    pill_num = file.readline()
    file.close()

    msg = ("Pill Count: " + str(pill_num))

    Label(t, text=msg, font="helvetica 22", bg='white').pack(pady=200)
    b1 = Button(t, text="Back", bg='white', font="helvetica 12",
command=t.destroy).pack()

def pill_count3():
    t = Toplevel(root)
    t.configure(bg='white')
    t.geometry("800x480")
    t.overrideredirect(True)
    w = 800
    h = 480
    # get screen width and height
    ws = root.winfo_screenwidth()
    hs = root.winfo_screenheight()
    x = (ws/2) - (w/2)
    y = (hs/2) - (h/2)
    # set the dimensions of the screen and where it is placed
    t.geometry('%dx%d+%d+%d' % (w, h, x, y))
    #t.grab_set()

    file = open("./count3.txt", "r")
    pill_num = file.readline()
    file.close()

    msg = ("Pill Count: " + str(pill_num))

    Label(t, text=msg, font="helvetica 22", bg='white').pack(pady=200)
    b1 = Button(t, text="Back", bg='white', font="helvetica 12",
command=t.destroy).pack()

def pill_count4():
    t = Toplevel(root)
    t.configure(bg='white')
    t.geometry("800x480")
    t.overrideredirect(True)
    w = 800
    h = 480
    # get screen width and height
    ws = root.winfo_screenwidth()
    hs = root.winfo_screenheight()
    x = (ws/2) - (w/2)
    y = (hs/2) - (h/2)
    # set the dimensions of the screen and where it is placed
    t.geometry('%dx%d+%d+%d' % (w, h, x, y))
    #t.grab_set()

    file = open("./count4.txt", "r")
    pill_num = file.readline()
    file.close()

    msg = ("Pill Count: " + str(pill_num))
```

```
    Label(t, text=msg, font="helvetica 22", bg='white').pack(pady=200)
    b1 = Button(t, text="Back", bg='white', font="helvetica 12",
command=t.destroy).pack()

def pill_count5():
    t = Toplevel(root)
    t.configure(bg='white')
    t.geometry("800x480")
    t.overrideredirect(True)
    w = 800
    h = 480
    # get screen width and height
    ws = root.winfo_screenwidth()
    hs = root.winfo_screenheight()
    x = (ws/2) - (w/2)
    y = (hs/2) - (h/2)
    # set the dimensions of the screen and where it is placed
    t.geometry('%dx%d+%d+%d' % (w, h, x, y))
    #t.grab_set()

    file = open("./count5.txt", "r")
    pill_num = file.readline()
    file.close()

    msg = ("Pill Count: " + str(pill_num))

    Label(t, text=msg, font="helvetica 22", bg='white').pack(pady=200)
    b1 = Button(t, text="Back", bg='white', font="helvetica 12",
command=t.destroy).pack()

def pill_count6():
    t = Toplevel(root)
    t.configure(bg='white')
    t.geometry("800x480")
    t.overrideredirect(True)
    w = 800
    h = 480
    # get screen width and height
    ws = root.winfo_screenwidth()
    hs = root.winfo_screenheight()
    x = (ws/2) - (w/2)
    y = (hs/2) - (h/2)
    # set the dimensions of the screen and where it is placed
    t.geometry('%dx%d+%d+%d' % (w, h, x, y))
    #t.grab_set()

    file = open("./count6.txt", "r")
    pill_num = file.readline()
    file.close()

    msg = ("Pill Count: " + str(pill_num))

    Label(t, text=msg, font="helvetica 22", bg='white').pack(pady=200)
    b1 = Button(t, text="Back", bg='white', font="helvetica 12",
command=t.destroy).pack()

def pill_count7():
```

```
    t = Toplevel(root)
    t.configure(bg='white')
    t.geometry("800x480")
    t.overrideredirect(True)
    w = 800
    h = 480
    # get screen width and height
    ws = root.winfo_screenwidth()
    hs = root.winfo_screenheight()
    x = (ws/2) - (w/2)
    y = (hs/2) - (h/2)
    # set the dimensions of the screen and where it is placed
    t.geometry('%dx%d+%d+%d' % (w, h, x, y))
    #t.grab_set()

    file = open("./count7.txt", "r")
    pill_num = file.readline()
    file.close()

    msg = ("Pill Count: " + str(pill_num))

    Label(t, text=msg, font="helvetica 22", bg='white').pack(pady=200)
    b1 = Button(t, text="Back", bg='white', font="helvetica 12",
command=t.destroy).pack()

def pill_count8():
    t = Toplevel(root)
    t.configure(bg='white')
    t.geometry("800x480")
    t.overrideredirect(True)
    w = 800
    h = 480
    # get screen width and height
    ws = root.winfo_screenwidth()
    hs = root.winfo_screenheight()
    x = (ws/2) - (w/2)
    y = (hs/2) - (h/2)
    # set the dimensions of the screen and where it is placed
    t.geometry('%dx%d+%d+%d' % (w, h, x, y))
    #t.grab_set()

    file = open("./count8.txt", "r")
    pill_num = file.readline()
    file.close()

    msg = ("Pill Count: " + str(pill_num))

    Label(t, text=msg, font="helvetica 22", bg='white').pack(pady=200)
    b1 = Button(t, text="Back", bg='white', font="helvetica 12",
command=t.destroy).pack()

def med_window():
    t = Toplevel(root)
    t.configure(bg='white')
    t.geometry("800x480")
    t.overrideredirect(True)
    w = 800
```

```
    h = 480
    # get screen width and height
    ws = root.winfo_screenwidth()
    hs = root.winfo_screenheight()
    x = (ws/2) - (w/2)
    y = (hs/2) - (h/2)
    # set the dimensions of the screen and where it is placed
    t.geometry('%dx%d+%d+%d' % (w, h, x, y))

    heading = Label(t, text="Medication List", font="helvetica 30 bold",
bg='white')
    heading.pack(pady=20)

    i = 0

    if os.path.isfile("./Convert1.txt"):
        file = open("Convert1.txt", "r")
        file.readline()                     #id tag
        file.readline()                     #name
        file.readline()                     #email
        file.readline()                     #contact name
        file.readline()                     #contact email
        read_medname = file.readline()      #med name
        alter_read_medname = read_medname.replace("\n", '')
        file.readline()                     #med quantity
        file.readline()                     #med dose
        file.readline()                     #med freq
        file.readline()                     #med notes

        needed_schedule = file.readline()  #as needed/schedule

        frame = Frame(t, bg='white')

        Label(frame, text=alter_read_medname, font="helvetica 14",
bg='white', width=35, anchor=W, pady=5).grid(row=i, column=0, sticky=W,
pady=5)
        Button(frame, text="Pill Count", font="helvetica 12", bg='white',
command=pill_count1, pady=5).grid(row=i, column=1, pady=5)
        button = Button(frame,text="DISPENSE", font="helvetica 18",
bg='white', pady=5)
        button.grid(row=i, column=2, pady=5, padx=5)
        if needed_schedule == "Schedule\n":
            button.config(state=DISABLED)
        frame.pack()

        file.close()

        i = i+1

    if os.path.isfile("./Convert2.txt"):
        file = open("Convert2.txt", "r")
        file.readline()                     #id tag
        file.readline()                     #name
        file.readline()                     #email
        file.readline()                     #contact name
        file.readline()                     #contact email
        read_medname = file.readline()      #med name
```

```
        alter_read_medname = read_medname.replace("\n", '')
        file.readline()                     #med quantity
        file.readline()                     #med dose
        file.readline()                     #med freq
        file.readline()                     #med notes
        needed_schedule = file.readline()  #as needed/schedule

        frame = Frame(t, bg='white')

        Label(frame, text=alter_read_medname, font="helvetica 14",
bg='white', width=35, anchor=W, pady=5).grid(row=i, column=0, sticky=W,
pady=5)
        Button(frame, text="Pill Count", font="helvetica 12", bg='white',
command=pill_count2, pady=5).grid(row=i, column=1, pady=5)
        button = Button(frame,text="DISPENSE", font="helvetica 18",
bg='white', pady=5)
        button.grid(row=i, column=2, pady=5, padx=5)
        if needed_schedule == "Schedule\n":
            button.config(state=DISABLED)
        frame.pack()

        file.close()

        i = i+1

    if os.path.isfile("./Convert3.txt"):
        file = open("Convert3.txt", "r")
        file.readline()                     #id tag
        file.readline()                     #name
        file.readline()                     #email
        file.readline()                     #contact name
        file.readline()                     #contact email
        read_medname = file.readline()     #med name
        alter_read_medname = read_medname.replace("\n", '')
        file.readline()                     #med quantity
        file.readline()                     #med dose
        file.readline()                     #med freq
        file.readline()                     #med notes

        needed_schedule = file.readline()  #as needed/schedule

        frame = Frame(t, bg='white')

        Label(frame, text=alter_read_medname, font="helvetica 14",
bg='white', width=35, anchor=W, pady=5).grid(row=i, column=0, sticky=W,
pady=5)
        Button(frame, text="Pill Count", font="helvetica 12", bg='white',
command=pill_count3, pady=5).grid(row=i, column=1, pady=5)
        button = Button(frame,text="DISPENSE", font="helvetica 18",
bg='white', pady=5)
        button.grid(row=i, column=2, pady=5, padx=5)
        if needed_schedule == "Schedule\n":
            button.config(state=DISABLED)
        frame.pack()

        file.close()
```

```
        i = i+1

    if os.path.isfile("./Convert4.txt"):
        file = open("Convert4.txt", "r")
        file.readline()                     #id tag
        file.readline()                     #name
        file.readline()                     #email
        file.readline()                     #contact name
        file.readline()                     #contact email
        read_medname = file.readline()      #med name
        alter_read_medname = read_medname.replace("\n", '')

        file.readline()                     #med quantity
        file.readline()                     #med dose
        file.readline()                     #med freq
        file.readline()                     #med notes
        needed_schedule = file.readline()  #as needed/schedule

        frame = Frame(t, bg='white')

        Label(frame, text=alter_read_medname, font="helvetica 14",
bg='white', width=35, anchor=W, pady=5).grid(row=i, column=0, sticky=W,
pady=5)
        Button(frame, text="Pill Count", font="helvetica 12", bg='white',
command=pill_count4, pady=5).grid(row=i, column=1, pady=5)
        button = Button(frame,text="DISPENSE", font="helvetica 18",
bg='white', pady=5)
        button.grid(row=i, column=2, pady=5, padx=5)
        if needed_schedule == "Schedule\n":
            button.config(state=DISABLED)
        frame.pack()

        file.close()

        i = i+1

    if os.path.isfile("./Convert5.txt"):
        file = open("Convert5.txt", "r")
        file.readline()                     #id tag
        file.readline()                     #name
        file.readline()                     #email
        file.readline()                     #contact name
        file.readline()                     #contact email
        read_medname = file.readline()      #med name
        alter_read_medname = read_medname.replace("\n", '')
        file.readline()                     #med quantity
        file.readline()                     #med dose
        file.readline()                     #med freq
        file.readline()                     #med notes
        needed_schedule = file.readline()  #as needed/schedule

        frame = Frame(t, bg='white')

        Label(frame, text=alter_read_medname, font="helvetica 14",
bg='white', width=35, anchor=W, pady=5).grid(row=i, column=0, sticky=W,
pady=5)
```

```
        Button(frame, text="Pill Count", font="helvetica 12", bg='white',
command=pill_count5, pady=5).grid(row=i, column=1, pady=5)
        button = Button(frame,text="DISPENSE", font="helvetica 18",
bg='white', pady=5)
        button.grid(row=i, column=2, pady=5, padx=5)
        if needed_schedule == "Schedule\n":
            button.config(state=DISABLED)
        frame.pack()

        file.close()

        i = i+1

    if os.path.isfile("./Convert6.txt"):
        file = open("Convert6.txt", "r")
        file.readline()                     #id tag
        file.readline()                     #name
        file.readline()                     #email
        file.readline()                     #contact name
        file.readline()                     #contact email
        read_medname = file.readline()      #med name
        alter_read_medname = read_medname.replace("\n", '')
        file.readline()                     #med quantity
        file.readline()                     #med dose
        file.readline()                     #med freq
        file.readline()                     #med notes
        needed_schedule = file.readline()  #as needed/schedule

        frame = Frame(t, bg='white')

        Label(frame, text=alter_read_medname, font="helvetica 14",
bg='white', width=35, anchor=W, pady=5).grid(row=i, column=0, sticky=W,
pady=5)
        Button(frame, text="Pill Count", font="helvetica 12", bg='white',
command=pill_count6, pady=5).grid(row=i, column=1, pady=5)
        button = Button(frame,text="DISPENSE", font="helvetica 18",
bg='white', pady=5)
        button.grid(row=i, column=2, pady=5, padx=5)
        if needed_schedule == "Schedule\n":
            button.config(state=DISABLED)
        frame.pack()

        file.close()

        i = i+1

    if os.path.isfile("./Convert7.txt"):
        file = open("Convert7.txt", "r")
        file.readline()                     #id tag
        file.readline()                     #name
        file.readline()                     #email
        file.readline()                     #contact name
        file.readline()                     #contact email
        read_medname = file.readline()      #med name
        alter_read_medname = read_medname.replace("\n", '')
        file.readline()                     #med quantity
        file.readline()                     #med dose
```

```
        file.readline()                     #med freq
        file.readline()                     #med notes
        needed_schedule = file.readline()  #as needed/schedule

        frame = Frame(t, bg='white')

        Label(frame, text=alter_read_medname, font="helvetica 14",
bg='white', width=35, anchor=W, pady=5).grid(row=i, column=0, sticky=W,
pady=5)
        Button(frame, text="Pill Count", font="helvetica 12", bg='white',
command=pill_count7, pady=5).grid(row=i, column=1, pady=5)
        button = Button(frame,text="DISPENSE", font="helvetica 18",
bg='white', pady=5)
        button.grid(row=i, column=2, pady=5, padx=5)
        if needed_schedule == "Schedule\n":
            button.config(state=DISABLED)
        frame.pack()

        file.close()

        i = i+1

    if os.path.isfile("./Convert8.txt"):
        file = open("Convert8.txt", "r")
        file.readline()                     #id tag
        file.readline()                     #name
        file.readline()                     #email
        file.readline()                     #contact name
        file.readline()                     #contact email
        read_medname = file.readline()     #med name
        alter_read_medname = read_medname.replace("\n", '')
        file.readline()                     #med quantity
        file.readline()                     #med dose
        file.readline()                     #med freq
        file.readline()                     #med notes
        needed_schedule = file.readline()  #as needed/schedule

        frame = Frame(t, bg='white')

        Label(frame, text=alter_read_medname, font="helvetica 14",
bg='white', width=35, anchor=W, pady=5).grid(row=i, column=0, sticky=W,
pady=5)
        Button(frame, text="Pill Count", font="helvetica 12", bg='white',
command=pill_count8, pady=5).grid(row=i, column=1, pady=5)
        button = Button(frame,text="DISPENSE", font="helvetica 18",
bg='white', pady=5)
        button.grid(row=i, column=2, pady=5, padx=5)
        if needed_schedule == "Schedule\n":
            button.config(state=DISABLED)
        frame.pack()

        file.close()

        i = i+1

    frame2 = Frame(t, bg='white')
```

```
    b_ok = Button(frame2, text="Back", font="helvetica 12", bg='white',
command=t.destroy)
    b_ok.pack()

    frame2.pack(pady=5)

#LOW PILL COUNT NOTIFICATION
def low_pill():
    if (GPIO.input(11)):
        to = 'nnumair@ieee.org'
        gmail_user = 'team4dispensum@gmail.com'
        gmail_pwd = 'LetsDispens'
        smtpserver = smtplib.SMTP_SSL("smtp.gmail.com",465)
        smtpserver.ehlo()
        smtpserver.login(gmail_user, gmail_pwd)
        header = 'To: ' + to + '\n' + 'From: ' + gmail_user + '\n' +
'Subject: Dispensum Alert! \n'
        print header
        msg = header + 'The pill count in one of the cartdridges is low
refill needed soon. \n\n'
        smtpserver.sendmail(gmail_user, to, msg)
        print 'Help Sent'
        smtpserver.close()

def help_me():
    if (GPIO.input(11)):
        to = 'nnumair@ieee.org'
        gmail_user = 'team4dispensum@gmail.com'
        gmail_pwd = 'LetsDispens'
        smtpserver = smtplib.SMTP_SSL("smtp.gmail.com",465)
        smtpserver.ehlo()
        smtpserver.login(gmail_user, gmail_pwd)
        header = 'To: ' + to + '\n' + 'From: ' + gmail_user + '\n' +
'Subject: Dispensum Alert! \n'
        print header
        msg = header + 'Your Family Member needs help. \n\n'
        smtpserver.sendmail(gmail_user, to, msg)
        print 'Help Sent'
        smtpserver.close()
        root.after(100, help_me)

# Capture SIGINT for cleanup when the script is aborted
def end_read(signal,frame):
    global continue_reading
    continue_reading = False
    GPIO.cleanup()

def MFRC522_StopCrypto1(self):
    self.ClearBitMask(self.Status2Reg, 0x08)

#LOAD STATE FOR RFID
def load_state():
    global LDS1
    global LDS2
    global LDS3
    global LDS4
    global LDS5
```

```
global LDS6
global LDS7
global LDS8

while True:
    if LDS1 == False:
        usbCOM = serial.Serial('/dev/ttyACM0', 9600)
        usbCOM.close()
        usbCOM.open()
        usbCOM.write('11')
        time.sleep(5)
        LDS1 = True
        break
    if LDS2 == False:
        usbCOM = serial.Serial('/dev/ttyACM0', 9600)
        usbCOM.close()
        usbCOM.open()
        usbCOM.write('12')
        time.sleep(1)
        load_state
        LDS2 = True
        break
    if LDS3 == False:
        usbCOM = serial.Serial('/dev/ttyACM0', 9600)
        usbCOM.close()
        usbCOM.open()
        usbCOM.write('13')
        time.sleep(1)
        load_state
        LDS3 = True
        break
    if LDS4 == False:
        usbCOM = serial.Serial('/dev/ttyACM0', 9600)
        usbCOM.close()
        usbCOM.open()
        usbCOM.write('14')
        time.sleep(1)
        load_state
        LDS4 = True
        break
    if LDS5 == False:
        usbCOM = serial.Serial('/dev/ttyACM0', 9600)
        usbCOM.close()
        usbCOM.open()
        usbCOM.write('15')
        time.sleep(1)
        load_state
        LDS5 = True
        break
    if LDS6 == False:
        usbCOM = serial.Serial('/dev/ttyACM0', 9600)
        usbCOM.close()
        usbCOM.open()
        usbCOM.write('16')
        time.sleep(1)
        load_state
        LDS6 = True
```

```
            break
        if LDS7 == False:
            usbCOM = serial.Serial('/dev/ttyACM0', 9600)
            usbCOM.close()
            usbCOM.open()
            usbCOM.write('17')
            time.sleep(1)
            load_state
            LDS1 = True
            break
        if LDS8 == False:
            usbCOM = serial.Serial('/dev/ttyACM0', 9600)
            usbCOM.close()
            usbCOM.open()
            usbCOM.write('18')
            time.sleep(1)
            load_state
            LDS8 = True
            break
    time.sleep(1)

    #Create File for Cartridge
    orig_stdout = sys.stdout
    i = 1
    if i == 9:
        i = 1
    while os.path.exists("Transferred%s.txt" %i):
            i +=1
    file = open("Transferred%s.txt" %i,"w+")
    sys.stdout = file

    continue_reading = True

    # Hook the SIGINT
    signal.signal(signal.SIGINT, end_read)

    # Create an object of the class MFRC522
    MIFAREReader = MFRC522.MFRC522()

    # This loop keeps checking for chips. If one is near it will get the
UID and authenticate
    while continue_reading:

        # Scan for cards
        (status,TagType) =
MIFAREReader.MFRC522_Request(MIFAREReader.PICC_REQIDL)

        # If a card is found
        if status == MIFAREReader.MI_OK:
            #print "Card detected"
            #Another Write
            #file.write("Card detected")

        # Get the UID of the card
            (status,uid) = MIFAREReader.MFRC522_Anticoll()

        # If we have the UID, continue
```

```
        if status == MIFAREReader.MI_OK:

            # Print UID
            print str(uid[0])+str(uid[1])+str(uid[2])+str(uid[3])
            # Write UID
            #file.write "Card read UID:
"+str(uid[0])+","+str(uid[1])+","+str(uid[2])+","+str(uid[3])

            # This is the default key for authentication
            key = [0xFF,0xFF,0xFF,0xFF,0xFF,0xFF]

            # Select the scanned tag
            MIFAREReader.MFRC522_SelectTag(uid)

            # Dump the data
            MIFAREReader.MFRC522_DumpClassic1K(key, uid)

            # Stop
            MIFAREReader.MFRC522_StopCrypto1()

            time.sleep(2)
            file.close()
            break

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~CONVERT
FILE~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#
    while os.path.isfile("Transferred%s.txt" %i):
        file1 = open("Transferred%s.txt" %i, "r")
        file2 = open("Convert%s.txt" %i, "w+")
        break

    id_tag = file1.readline()
    alter_id_tag = id_tag.replace("\n", '')
    file2.write("%s \n" % alter_id_tag)

    file1.readline()            #sector 0

    line = file1.readline()                  #sector 1
    alter1_line1 = line.replace("[", '')
    alter2_line1 = alter1_line1.replace("]", ',')
    alter3_line1 = alter2_line1.replace("\n", '')

    line = file1.readline()                  #sector 2
    alter1_line2 = line.replace("[", ' ')
    alter2_line2 = alter1_line2.replace("]", '')
    alter3_line2 = alter2_line2.replace(", 75, 189", '')
    alter4_line2 = alter3_line2.replace("\n", '')

    combine = eval('[' + alter3_line1 + alter4_line2 + ']')
    Convert = ''.join(chr(i) for i in combine)
    file2.write("%s \n" % Convert)

    stop = "[0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]" + "\n"

    while line and line != stop:
            line = file1.readline()          #sector 3
```

```
            line = file1.readline()          #sector 4
            if line == stop:
                    break
            alter1_line1 = line.replace("[10,", '')
            alter2_line1 = alter1_line1.replace("]", ',')
            alter3_line1 = alter2_line1.replace("\n", '')

            line = file1.readline()          #sector 5
            alter1_line2 = line.replace("[", ' ')
            alter2_line2 = alter1_line2.replace("]", '')
            alter3_line2 = alter2_line2.replace(", 75, 189", '')
            alter4_line2 = alter3_line2.replace("\n", '')

            combine = eval('[' + alter3_line1 + alter4_line2 + ']')
            Convert = ''.join(chr(i) for i in combine)
            file2.write("%s \n" % Convert)

            line = file1.readline()          #sector 6
            if line == stop:
                    break
            alter1_line1 = line.replace("[10,", '')
            alter2_line1 = alter1_line1.replace("]", ',')
            alter3_line1 = alter2_line1.replace("\n", '')

            line = file1.readline()          #sector 7

            line = file1.readline()          #sector 8
            alter1_line2 = line.replace("[", ' ')
            alter2_line2 = alter1_line2.replace("]", '')
            alter3_line2 = alter2_line2.replace(", 75, 189", '')
            alter4_line2 = alter3_line2.replace("\n", '')

            combine = eval('[' + alter3_line1 + alter4_line2 + ']')
            Convert = ''.join(chr(i) for i in combine)
            file2.write("%s \n" % Convert)

            line = file1.readline()          #sector 9
            if line == stop:
                    break
            alter1_line1 = line.replace("[10,", '')
            alter2_line1 = alter1_line1.replace("]", ',')
            alter3_line1 = alter2_line1.replace("\n", '')

            line = file1.readline()          #sector 10
            alter1_line2 = line.replace("[", ' ')
            alter2_line2 = alter1_line2.replace("]", '')
            alter3_line2 = alter2_line2.replace(", 75, 189", '')
            alter4_line2 = alter3_line2.replace("\n", '')

            combine = eval('[' + alter3_line1 + alter4_line2 + ']')
            Convert = ''.join(chr(i) for i in combine)
            file2.write("%s \n" % Convert)

    file1.close()
    file2.close()

#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#
```

```
root = Tk()
root.geometry("800x480")                # screen size of RPi
root.overrideredirect(True)
root.configure(bg='white')
w = 800
h = 480
# get screen width and height
ws = root.winfo_screenwidth()
hs = root.winfo_screenheight()
x = (ws/2) - (w/2)
y = (hs/2) - (h/2)
# set the dimensions of the screen and where it is placed
root.geometry('%dx%d+%d+%d' % (w, h, x, y))

frame1 = Frame(root, bg='white')

temp = Image.open("user_image.jpg")
temp = temp.save("user_image.ppm", "ppm")
image = PhotoImage(file = "user_image.ppm")

imagepanel = Label(frame1, image=image, height=175, bg='white')
imagepanel.grid(row=0, column=0, sticky=N)

frame2 = Frame(root, bg='white')

time1 = ''
clock = Label(frame2, font=('times', 85, 'bold'), bg='white')
clock.grid(row=0, column=0, sticky=E, padx=5)

pm = Label(frame2, font=('times', 35), bg='white')
pm.grid(row=0, column=1, sticky=W, padx=5)

frame3 = Frame(root, bg='white')

date = Label(frame3, font=('times', 25), bg='white', pady=5)
date.pack()

time()

frame4 = Frame(root, bg='white')

make_loadbutton()

make_medbutton()

frame1.pack(pady=5)
frame2.pack()
frame3.pack()
frame4.pack(fill=BOTH, pady=10)

time.sleep(0.5)

root.mainloop()
```

Figure 6. User/Interaction GUI Program Code

B. Testing

The user/interaction GUI program consists of several functions so each was testing separately before integrating with other functions. Since this GUI program uses that text file from the RFID transfer we ran the RFID code for transferring a file to the raspberry pi multiple times to check for consistent data transfer. Next the convert code was tested multiple times with different RFID transfer text files to verify if it converts format of the text file from decimal to ASCII. All the other functions Testing the program was fairly simple because it was easy to check if the program was not functioning correctly. Since the code for dispense medication is similar to the one in the medication schedule/dispense program no further testing was needed for that function. All the other functions of the program were tested by running the GUI program and checking if all the widgets (e.g. buttons, labels) appeared correctly on the Raspberry Pi screen with the correct data.

When testing the main screen of the user/interaction GUI program some of the widgets were either not displaying properly or not aligned. The code was modified and then tested several of times before the main screen displayed correctly with the project logo, current time, date, and two buttons. The final iteration of main screen is shown below in Figure 7.

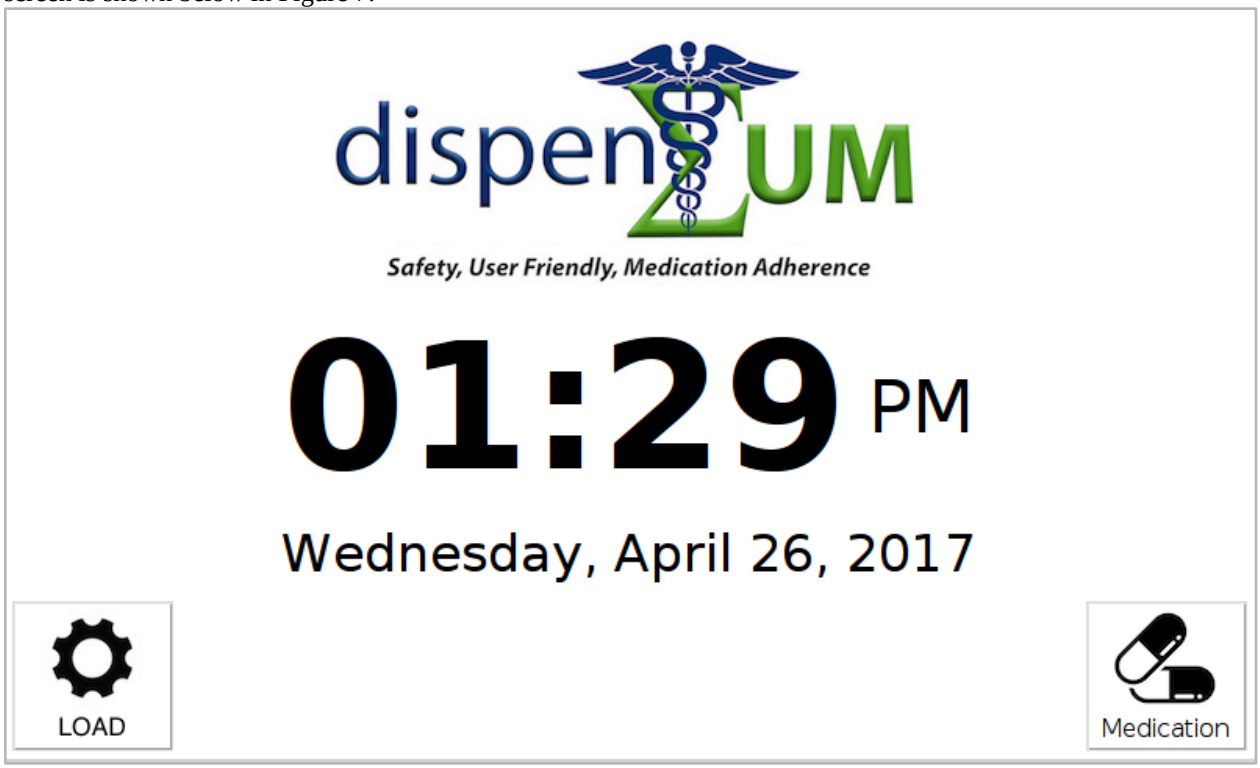


Figure 7. User/Interaction GUI Main Window

When the "Medication" button on the main screen is clicked a new window (Medication List) appears as expected (refer to Figure 8). When testing the medication list window some of the medications did not appear on the screen when trying to display all eight medications. However, when the font size was reduced all eight medications displayed correctly as shown below in Figure 8. For the medication that is taken on a schedule when the corresponding dispense button on the medication list window is clicked nothing happens because that button is deactivated. Only the medication that is taken as needed has their dispense button enabled (refer to Figure 8) which verifies correct functionality.

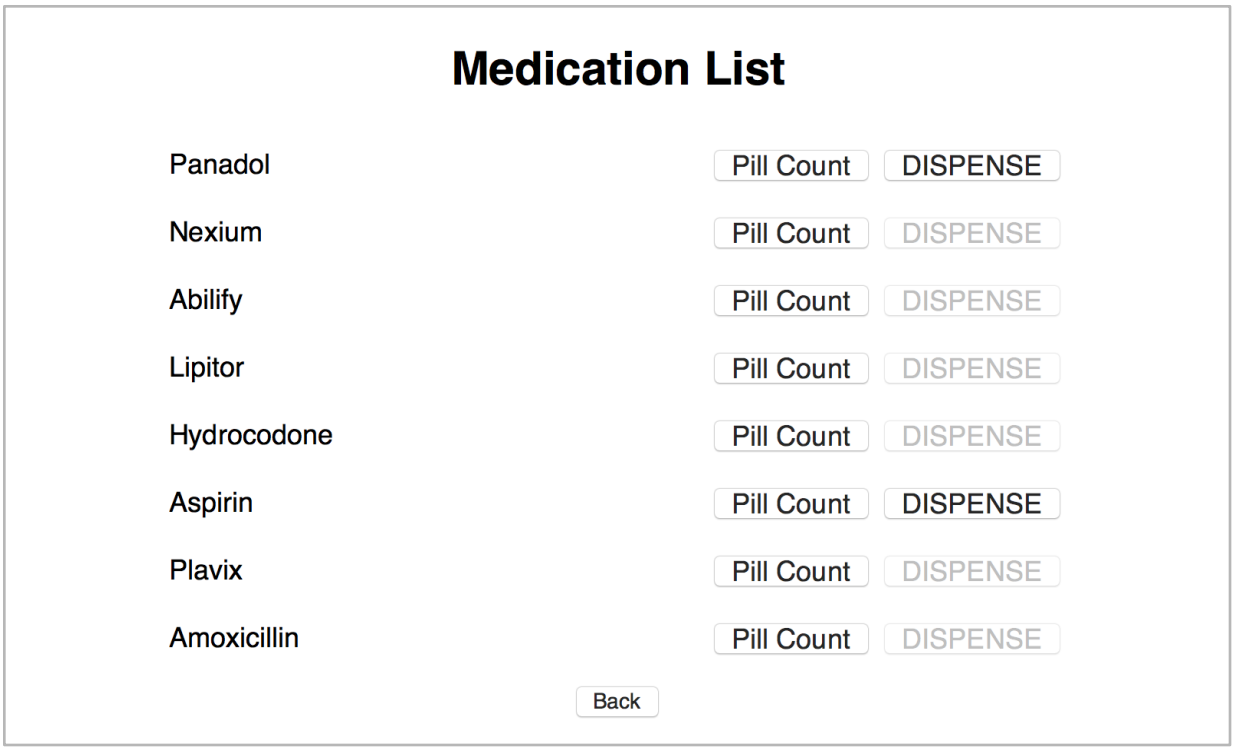


Figure 8. Medication List Window

The "Pill Count" button when tested functions as expected as it reads the correct pill count text file generated by the medication schedule/dispense program (refer to the following section of this appendix) and displays the pill count. Currently the "Dispense" button has not been fully test as the dispense medication code has yet to be implemented to the button. However, as mentioned earlier the dispense code has been tested in the medication schedule/dispense program so we do not expect to have any issues getting the medication to dispense when the "Dispense" button is clicked. The "Load" button on the main window works correctly as when pressed it executes the convert code which reads the RFID text file and converts the data from decimal to ASCII format and stores it into a new text file as shown below.

V. Medication Schedule/Dispense Program

- Discuss pill count text file

While the user GUI shows all the foreground information for the end user the main mind of what makes DispenSUM work all works in the background code. This code will open the information from the converted transferred file and store the information into the system..After all the formation is stored we wait for a match in the times. Once then we enter a dispensing cycle. This is where the user is then notified that is time to take their medication with a sound notification. From there the system will send a signal to the motors to rotate to the appropriate location. From here we decrement the pill count and update the file that founds the pill count between the two systems. If the pill count is less than 5 an email is send out saying there is a low pill count. If the pill count is zero then it removes all related info regarding that pill. Once everything is done we loop back to wait state for time trigger.

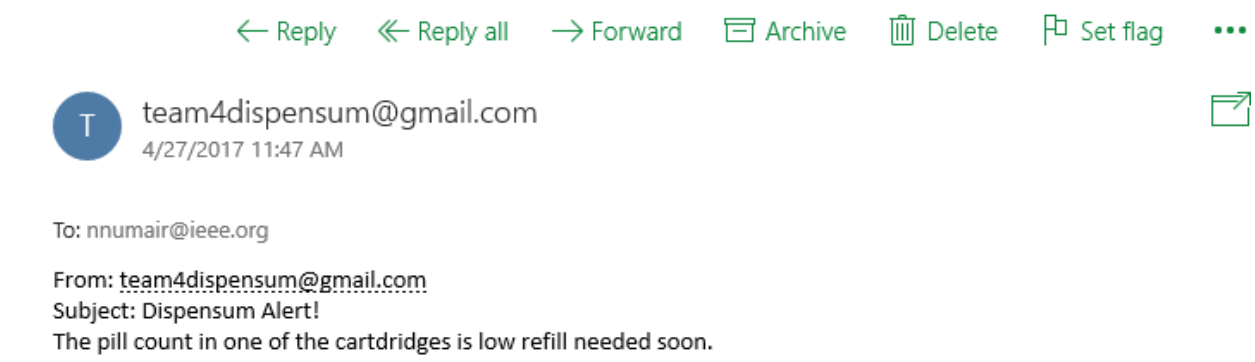


Figure 9: low pill count email

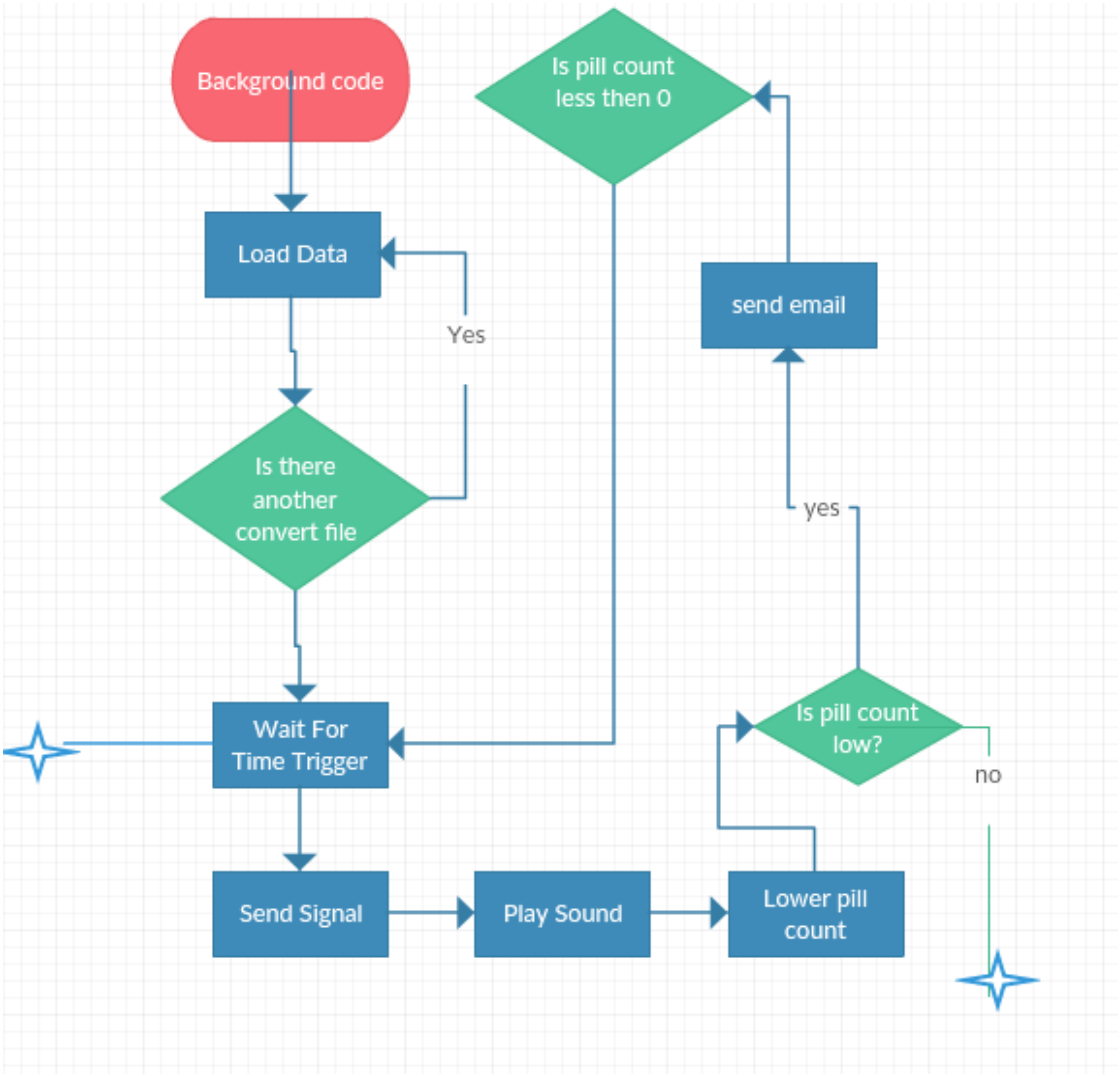


Figure 10: Dispensing flow chart

A. Code

```
import time
```

```
import smtplib
import RPi.GPIO as GPIO
import MFRC522
import signal
import os
import serial



GPIO.setmode(GPIO.BOARD)
GPIO.setup(11,GPIO.IN)
input = GPIO.input(11)

LDS1 = False

LDS2 = False

LDS3 = False

LDS4 = False

LDS5 = False

LDS6 = False

LDS7 = False

LDS8 = False
```

TimerST = False

UID1 = None

UID2 = None

UID3 = None

UID4 = None

UID5 = None

UID6 = None

UID7 = None

UID8 = None

P1 = None

P2 = None

P3 = None

P4 = None

P5 = None

P6 = None

P7 = None

P8 = None

Feq1 = None

Feq2 = None

Feq3 = None

Feq4 = None

Feq5 = None

Feq6 = None

Feq7 = None

Feq8 = None

D1 = None

D2 = None

```
D3 = None
D4 = None
D5 = None
D6 = None
D7 = None
D8 = None

T1 = None
T2 = None
T3 = None
T4 = None
T5 = None
T6 = None
T7 = None
T8 = None

def low_pill():
    to = email
    gmail_user = 'team4dispensum@gmail.com'
    gmail_pwd = 'LetsDispens'
    smtpserver = smtplib.SMTP_SSL("smtp.gmail.com",465)
    smtpserver.ehlo()
    smtpserver.login(gmail_user, gmail_pwd)
    header = 'To: ' + to + '\n' + 'From: ' + gmail_user + '\n' + 'Subject: Dispensum Alert! \n'
    print header
    msg = header + 'The pill count in one of the cartdridges is low refill needed soon. \n\n'
    smtpserver.sendmail(gmail_user, to, msg)
    print 'Help Sent'
    smtpserver.close()
    time.sleep(0.05)
```

```
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~Cartridge Population~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#
while True:
    #hour = localtime.tm_hour
    #minute = localtime.tm_min
    #Populate Cartridge one and Email
    if LDS1 == False:
        if os.path.isfile("./Convert1.txt"):
            print "pop 1"
            pop1 = open("Convert1.txt")
            LDS1 = True
            UID1 = pop1.readline()     #loads ID into File
            pop1.readline()            #Skip line Name Line
            pop1.readline()            #Skip line Email line
            pop1.readline()            #Skip line Help name line
            email = pop1.readline()    #Loads email
            pop1.readline()            #Skip line Med name line
            P1 = pop1.readline()       #How many pills we have
            Feq1 = pop1.readline()     #How many Times a day
            D1 = pop1.readline()       #How many signles to dispense per time stamp
            pop1.readline()            #Skip notes line
            pop1.readline()            #Skip Type line
            T1 = pop1.readline()       #Load first time stamp


    #Populate Cartridge two and no email
    if os.path.isfile("./Convert2.txt"):
        if LDS2 == False:
            #print "pop 2"
            pop2 = open("Convert2.txt")
```

```
        LDS2 = True
        UID2 = pop2.readline()     #loads ID into File
        pop2.readline()            #Skip line Name Line
        pop2.readline()            #Skip line Email line
        pop2.readline()            #Skip line Help name line
        pop2.readline()            #Skip Loads email
        pop2.readline()            #Skip line Med name line
        P2 = pop2.readline()       #How many pills we have
        Feq2 = pop2.readline()     #How many Times a day
        D2 = pop2.readline()       #How many signles to dispense per time stamp
        pop2.readline()            #Skip notes line
        pop2.readline()            #Skip Type line
        T2 = pop2.readline()       #Load first time stamp


    #Populate Cartridge Three and no email
if os.path.isfile("./Convert3.txt"):
    if LDS2 == False:
        #print "pop 3"
        pop3 = open("Convert3.txt")
        LDS3 = True
        UID3 = pop3.readline()     #loads ID into File
        pop3.readline()            #Skip line Name Line
        pop3.readline()            #Skip line Email line
        pop3.readline()            #Skip line Help name line
        pop3.readline()            #Skip Loads email
        pop3.readline()            #Skip line Med name line
        P3 = pop3.readline()       #How many pills we have
        Feq3 = pop3.readline()     #How many Times a day
        D3 = pop3.readline()       #How many signles to dispense per time stamp
```

```
        pop3.readline()          #Skip notes line
        pop3.readline()          #Skip Type line
        T3 = pop3.readline()     #Load first time stamp


    #Populate Cartridge two and no email
if os.path.isfile("./Convert4.txt"):
    if LDS3 == False:
        #print "pop 4"
        pop4 = open("Convert4.txt")
        LDS4 = True
        UID4 = pop4.readline()    #loads ID into File
        pop4.readline()          #Skip line Name Line
        pop4.readline()          #Skip line Email line
        pop4.readline()          #Skip line Help name line
        pop4.readline()          #Skip Loads email
        pop4.readline()          #Skip line Med name line
        P4 = pop4.readline()     #How many pills we have
        Feq4 = pop4.readline()    #How many Times a day
        D4 = pop4.readline()     #How many signles to dispense per time stamp
        pop4.readline()          #Skip notes line
        pop4.readline()          #Skip Type line
        T4 = pop4.readline()     #Load first time stamp


    #Populate Cartridge Five and no email
if os.path.isfile("./Convert5.txt"):
    if LDS5 == False:
        #print "pop 5"
        pop5 = open("Convert5.txt")
```

```
        LDS5 = True
        UID5 = pop5.readline()     #loads ID into File
        pop5.readline()          #Skip line Name Line
        pop5.readline()          #Skip line Email line
        pop5.readline()          #Skip line Help name line
        pop5.readline()          #Skip Loads email
        pop5.readline()          #Skip line Med name line
        P5 = pop5.readline()      #How many pills we have
        Feq5 = pop5.readline()     #How many Times a day
        D5 = pop5.readline()      #How many signles to dispense per time stamp
        pop5.readline()          #Skip notes line
        pop5.readline()          #Skip Type line
        T5 = pop5.readline()      #Load first time stamp


    #Populate Cartridge Six and no email
if os.path.isfile("./Convert6.txt"):
    if LDS6 == False:
        #print "pop 6"
        pop6 = open("Convert6.txt")
        LDS6 = True
        UID6 = pop6.readline()     #loads ID into File
        pop6.readline()          #Skip line Name Line
        pop6.readline()          #Skip line Email line
        pop6.readline()          #Skip line Help name line
        pop6.readline()          #Skip Loads email
        pop6.readline()          #Skip line Med name line
        P6 = pop6.readline()      #How many pills we have
        Feq6 = pop6.readline()     #How many Times a day
        D6 = pop6.readline()      #How many signles to dispense per time stamp
```

```
        pop6.readline()          #Skip notes line
        pop6.readline()          #Skip Type line
        T6 = pop6.readline()     #Load first time stamp


    #Populate Cartridge Seven and no email
if os.path.isfile("./Convert7.txt"):
    if LDS7 == False:
        #print "pop 7"
        pop7 = open("Convert6.txt")
        LDS7 = True
        UID7 = pop7.readline()    #loads ID into File
        pop7.readline()          #Skip line Name Line
        pop7.readline()          #Skip line Email line
        pop7.readline()          #Skip line Help name line
        pop7.readline()          #Skip Loads email
        pop7.readline()          #Skip line Med name line
        P7 = pop7.readline()     #How many pills we have
        Feq7 = pop7.readline()    #How many Times a day
        D7 = pop7.readline()     #How many signles to dispense per time stamp
        pop7.readline()          #Skip notes line
        pop7.readline()          #Skip Type line
        T7 = pop7.readline()     #Load first time stamp


    #Populate Cartridge Eight and no email
if os.path.isfile("./Convert8.txt"):
    if LDS8 == False:
        #print "pop 8"
        pop8 = open("Convert8.txt")
```

```
        LDS8 = True
        UID8 = pop8.readline()     #loads ID into File
        pop8.readline()          #Skip line Name Line
        pop8.readline()          #Skip line Email line
        pop8.readline()          #Skip line Help name line
        pop8.readline()          #Skip Loads email
        pop8.readline()          #Skip line Med name line
        P8 = pop8.readline()      #How many pills we have
        Feq8 = pop8.readline()     #How many Times a day
        D8 = pop8.readline()      #How many signles to dispense per time stamp
        pop8.readline()          #Skip notes line
        pop8.readline()          #Skip Type line
        T8 = pop8.readline()      #Load first time stamp



#~~~~~~~~~~~~~~~~~~~~~~~~~~~~DISPENSING SCHUDULE~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~#
    localtime = time.localtime()
    hour = localtime.tm_hour
    minute = localtime.tm_min
    #Type Cast to int
    if LDS1 == True:
        D1 = int(D1)
        P1 = int(P1)
        T1 = int(T1)
        writeto = open("./count1.txt", "w+")
        writeto.write('%d' % P1)
        writeto.close
    if LDS2 == True:
        D2 = int(D2)
```

```
    P2 = int(P2)
    T2 = int(T2)
    writeto = open("./count2.txt", "w+")
    writeto.write('%d' % P2)
    writeto.close
if LDS3 == True:
    D3 = int(D3)
    P3 = int(P3)
    T3 = int(T3)
    writeto = open("./count3.txt", "w+")
    writeto.write('%d' % P3)
    writeto.close
if LDS4 == True:
    D4 = int(D4)
    P4 = int(P4)
    T4 = int(T4)
    writeto = open("./count4.txt", "w+")
    writeto.write('%d' % P4)
    writeto.close
if LDS5 == True:
    D5 = int(D5)
    P5 = int(P5)
    T5 = int(T5)
    writeto = open("./count5.txt", "w+")
    writeto.write('%d' % P5)
    writeto.close
if LDS6 == True:
    D6 = int(D6)
    P6 = int(P6)
    T6 = int(T6)
```

```
    writeto = open("./count6.txt", "w+")
    writeto.write('%d' % P6)
    writeto.close
if LDS7 == True:
    D7 = int(D7)
    P7 = int(P7)
    T7 = int(T7)
    writeto = open("./count7.txt", "w+")
    writeto.write('%d' % P7)
    writeto.close
if LDS8 == True:
    D8 = int(D8)
    P8 = int(P8)
    T8 = int(T8)
    writeto = open("./count8.txt", "w+")
    writeto.write('%d' % P8)
    writeto.close
TimerST = False
#For Cartridge 1
if LDS1 == True:
    if hour == T1 and minute == 00:
        os.system('mpg321 Dispensum.mp3 &')
        usbCOM = serial.Serial('/dev/ttyACM0', 9600)
        usbCOM.close()
        usbCOM.open()
        for k in range (0,D1):
            time.sleep(5)
            usbCOM.write('01')
            usbCOM.close()
            usbCOM.open()
```

```
        usbCOM.close()
        P1 = P1 - D1
        writeto = open("./count1.txt", "w+")
        writeto.write('%d' % P1)
        writeto.close
        if P1 < 5:
            low_pill()
            time.sleep(0.05)
        if P1 <= 0:
            UID1 = None
            P1 = None
            Feq1 = None
            D1 = None
            T1 = None
            os.remove("./Convert1.txt")
            os.remove("./Transferred1.txt")
            LDS1 = False
            break
        time.sleep(10)
        if TimerST == False:
            TimerST = True
            TimerVal = minute

#For Cartridge 2
if LDS2 == True:
    if hour == hour and minute == minute:
        os.system('mpg321 Dispensum.mp3 &')
        usbCOM = serial.Serial('/dev/ttyACM0', 9600)
        usbCOM.close()
        usbCOM.open()
```

```
        for k in range (0,D2):
            time.sleep(5)
            usbCOM.write('02')
            usbCOM.close()
            usbCOM.open()
        usbCOM.close()
        P2 = P2 - D2
        writeto = open("./count2.txt", "w+")
        writeto.write('%d' % P2)
        writeto.close
        if P2 < 5:
            low_pill
        if P2 <= 0:
            UID2 = None
            P2 = None
            Feq2 = None
            D2 = None
            T2 = None
            #LDS2 = False
            os.remove("./Convert2.txt")
            os.remove("./Transferred2.txt")
        time.sleep(60)
        if TimerST == False:
            TimerST = True
            TimerVal = minute

#For Cartridge 3
if LDS3 == True:
    if hour == T3 and minute == 00:
        os.system('mpg321 Dispensum.mp3 &')
```

```
        usbCOM = serial.Serial('/dev/ttyACM0', 9600)
        usbCOM.close()
        usbCOM.open()
        for k in range (0,D3):
            time.sleep(5)
            usbCOM.write('03')
            usbCOM.close()
            usbCOM.open()
        usbCOM.close()
        P3 = P3 - D3
        writeto = open("./count3.txt", "w+")
        writeto.write('%d' % P3)
        writeto.close
        if P3 < 5:
            low_pill
        if P3 <= 0:
            UID3 = None
            P3 = None
            Feq3 = None
            D3 = None
            T3 = None
            #LDS3 = False
            os.remove("./Convert3.txt")
            os.remove("./Transferred3.txt")
        time.sleep(60)
        if TimerST == False:
            TimerST = True
            TimerVal = time.localtime()

#For Cartridge 4
```

```
if LDS4 == True:
    if hour == T4 and minute == 00:
        os.system('mpg321 Dispensum.mp3 &')
        usbCOM = serial.Serial('/dev/ttyACM0', 9600)
        usbCOM.close()
        usbCOM.open()
        for k in range (0,D2):
            usbCOM.write('04')
            usbCOM.close()
            usbCOM.open()
        usbCOM.close()
        P4 = P4 - D4
        writeto = open("./count4.txt", "w+")
        writeto.write('%d' % P4)
        writeto.close
        if P4 < 5:
            low_pill
        if P4 <= 0:
            UID4 = None
            P4 = None
            Feq4 = None
            D4 = None
            T4 = None
            #LDS4 = False
            os.remove("./Convert4.txt")
            os.remove("./Transferred4.txt")
        #time.sleep(60)
        if TimerST == False:
            TimerST = True
            TimerVal = time.localtime()
```

```
#For Cartridge 5
if LDS5 == True:
    if hour == T5 and minute == 00:
        os.system('mpg321 Dispensum.mp3 &')
        usbCOM = serial.Serial('/dev/ttyACM0', 9600)
        usbCOM.close()
        usbCOM.open()
        for D5 in range (1,D5):
            time.sleep(5)
            usbCOM.write('05')
            usbCOM.close()
            usbCOM.open()
        usbCOM.close()
        P5 = P5 - D5
        writeto = open("./count5.txt", "w+")
        writeto.write('%d' % P5)
        writeto.close
        if P5 < 5:
            low_pill
        if P5 <= 0:
            UID5 = None
            P5 = None
            Feq5 = None
            D5 = None
            T5 = None
            #LDS5 = False
            os.remove("./Convert5.txt")
            os.remove("./Transferred5.txt")
        time.sleep(60)
```

```
        if TimerST == False:
            TimerST = True
            TimerVal = time.localtime()


#For Cartridge 6
if LDS6 == True:
    if hour == T6 and minute == 00:
        os.system('mpg321 Dispensum.mp3 &')
        usbCOM = serial.Serial('/dev/ttyACM0', 9600)
        usbCOM.close()
        usbCOM.open()
        for D6 in range (1,D6):
            usbCOM.write('06')
            usbCOM.close()
            usbCOM.open()
        usbCOM.close()
        P6 = P6 - D6
        writeto = open("./count6.txt", "w+")
        writeto.write('%d' % P6)
        writeto.close
        if P6 < 5:
            low_pill
        if P6 <= 0:
            UID6 = None
            P6 = None
            Feq6 = None
            D6 = None
            T6 = None
            #LDS6 = False
            os.remove("./Convert6.txt")
```

```
            os.remove("./Transferred6.txt")
        time.sleep(60)
        if TimerST == False:
            TimerST = True
            TimerVal = time.localtime()


#For Cartridge 7
if LDS7 == True:
    if hour == T7 and minute == 00:
        os.system('mpg321 Dispensum.mp3 &')
        usbCOM = serial.Serial('/dev/ttyACM0', 9600)
        usbCOM.close()
        usbCOM.open()
        for D7 in range (1,D7):
            usbCOM.write('07')
            usbCOM.close()
            usbCOM.open()
        usbCOM.close()
        P7 = P7 - D7
        writeto = open("./count7.txt", "w+")
        writeto.write('%d' % P7)
        writeto.close
        if P7 < 5:
            low_pill
        if P7 <= 0:
            UID7 = None
            P7 = None
            Feq7 = None
            D7 = None
            T7 = None
```

```
            #LDS7 = False
            os.remove("./Convert7.txt")
            os.remove("./Transferred7.txt")
        time.sleep(60)
        if TimerST == False:
            TimerST = True
            TimerVal = time.localtime()


#For Cartridge 8
if LDS8 == True:
    if hour == T8 and minute == 00:
        os.system('mpg321 Dispensum.mp3 &')
        usbCOM = serial.Serial('/dev/ttyACM0', 9600)
        usbCOM.close()
        usbCOM.open()
        for D8 in range (1,D8):
            usbCOM.write('08')
            usbCOM.close()
            usbCOM.open()
        usbCOM.close()
        P8 = P8 - D8
        writeto = open("./count8.txt", "w+")
        writeto.write('%d' % P8)
        writeto.close
        if P8 < 5:
            low_pill
        if P8 <= 0:
            UID8 = None
            P8 = None
            Feq8 = None
```

```
            D8 = None
            T8 = None
            #LDS8 = False
            os.remove("./Convert8.txt")
            os.remove("./Transferred8.txt")
        time.sleep(60)
        if TimerST == False:
            TimerST = True
            TimerVal = time.localtime()

#FOR REHOMING INTO POSITION
if TimerST == True:
    while True:
        localtime = time.localtime()
        hour = localtime.tm_hour
        minute = localtime.tm_min
        TimerValTrig = localtime.tm_min
        print TimerVal
        print TimerValTrig
        if TimerVal+15 <= TimerValTrig:
            TimerST = False
            usbCOM = serial.Serial('/dev/ttyACM0', 9600)
            usbCOM.close()
            usbCOM.open()
            usbCOM.write('09')
            time.sleep(1)
            usbCOM.close()
            usbCOM.open()
            usbCOM.write('19')
            usbCOM.close()
```

```
        break

#Email function

if (GPIO.input(11)):
    to = email
    gmail_user = 'team4dispensum@gmail.com'
    gmail_pwd = 'LetsDispens'
    smtpserver = smtplib.SMTP_SSL("smtp.gmail.com",465)
    smtpserver.ehlo()
    smtpserver.login(gmail_user, gmail_pwd)
    header = 'To: ' + to + '\n' + 'From: ' + gmail_user + '\n' + 'Subject: Dispensum Alert! \n'
    print header
    msg = header + 'Your Family Member needs help. \n\n'
    smtpserver.sendmail(gmail_user, to, msg)
    print 'Help Sent'
    smtpserver.close()
    time.sleep(0.05)
```

VI. Arduino Control Code

```
#include <digitalWriteFast.h>
#include <AccelStepper.h>
#include <Arduino.h>
#include <Servo.h>
Servo myservo;
AccelStepper Linkage_stepper(1,9,8);
AccelStepper Rotary_stepper(1,3,2);

int x=0;
int homeval=0;
int previousMillis=0;
int currentMillis=0;
int y=0;
int set=0;
int cartPos[] = {0, 0, 1250, 2500, 3750, 5000, 6250, 7500, 8750};
int usbRead=0;
int usbRead1=0;
int interval= 15000;
int input=0;
int buttonstate=0;
int pill=0;
int dispensed=0;
int LED_interval=500;
int LEDSTATE=0;
int qtiRead=1;
int z=0;
int incomingByte=0;
int incomingByte1=0;

#define Rotary_DIR 2
#define Rotary_Step 3
#define Rotary_Reset 4
#define m0 5
#define m1 6
#define m2 7
```

```
#define Linkage_DIR 8
#define Linkage_Step 9
#define Linkage_Reset 10
#define Limit_Switch 11
#define Dispense_Button 12
#define Servo_Signal 13
#define Load_Cell A0
#define Button_LED A1
#define Extend  0
#define Retract  70
#define QTI A3


void setup() {
  Serial.begin(9600);
  Rotary_stepper.setMaxSpeed(40000);
  Rotary_stepper.setAcceleration(4000);
  Linkage_stepper.setMaxSpeed(1250);
  Linkage_stepper.setAcceleration(4000);
  myservo.attach(Servo_Signal);


  pinMode(Rotary_Reset, OUTPUT);
  pinMode(m0, OUTPUT);
  pinMode(m1, OUTPUT);
  pinMode(m2, OUTPUT);
  pinMode(Linkage_Reset, OUTPUT);
  pinModeFast(Limit_Switch, INPUT);
  pinMode(Dispense_Button, INPUT);
  pinModeFast(Load_Cell, INPUT);
  pinMode(Button_LED, OUTPUT);
```

```
  digitalWrite(Rotary_Reset,HIGH);
  digitalWrite(Linkage_Reset,HIGH);
  digitalWrite(m0,LOW);
  digitalWrite(m1,HIGH);//high
  digitalWrite(m2,LOW);
  myservo.write(70);
}

void loop() {
  Rotary_stepper.run();//Have to call for stepper to work
  Linkage_stepper.run();// Have to call for stepper to work

 if (set==0){ // initialize home position
    homePosition();
  }
  if (set==1){
    digitalWrite(Linkage_Reset,LOW);
    digitalWrite(Button_LED,HIGH);
    if(Serial.available()){
      digitalWrite(Button_LED,LOW);
      incomingByte = Serial.read()-'0';
      delay(50);
      incomingByte1 = Serial.read()-'0';
      usbRead=incomingByte*10+incomingByte1;
      Serial.println(usbRead);
      if (usbRead > 0){// If read is greater than one, move to next state
         set=2;
         Serial.println(usbRead);
         digitalWrite(Button_LED,HIGH);
         digitalWrite(Rotary_Reset,HIGH); // Turn motors HIGH to get ready to move
         digitalWrite(Linkage_Reset,HIGH);
         if (usbRead==9){
          set=0;
          y=0;
         }
```

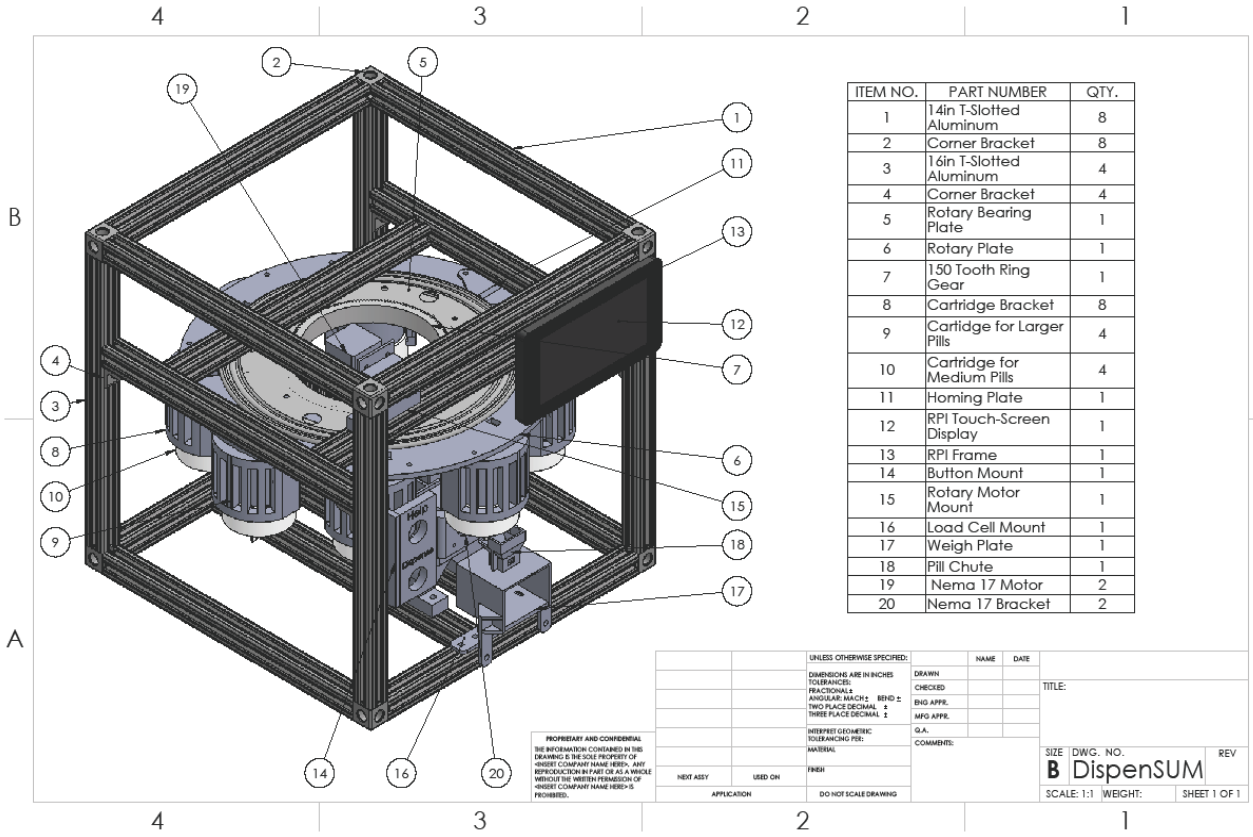
```
        if (usbRead==19){
           digitalWrite(Rotary_Reset,LOW);
           usbRead=0;
           set=1;
           buttonstate=0;
        }
      }
    }
  }
 if (set==2){
    if (usbRead>0 && usbRead <9){
      Dispense(); // calls Dispense Function with usbRead from PI
    }
    if (usbRead>10 && usbRead <19){
      Rotate();
    }
  }
}



void homePosition (){
  if (y==0){
    digitalWrite(Rotary_Reset,HIGH);
   Rotary_stepper.move(40000);
  homeval=digitalRead(Limit_Switch); // Checks Limit switch for plate
  }
  if (homeval==(HIGH)){
    Rotary_stepper.stop();
    Rotary_stepper.setCurrentPosition(0);// Sets home position to cartridge 0
    set=1; // Sets Pi Read State
  }
}

```

```
 if (x==0){ // In an if statement so it only runs once per dispense cycle
   digitalWrite(Rotary_Reset,HIGH);
   Rotary_stepper.moveTo(cartPos[usbRead]);//Sets move location to cartridge sent by PI
   x++;
 }
 if(Rotary_stepper.distanceToGo()==0){// Once Rotary is in position do rest
   if (z==0){
    digitalWrite(Linkage_Reset,HIGH);
    if (digitalRead(Dispense_Button)==HIGH){
     buttonstate=1;
    }
    if (buttonstate==1){
    Linkage_stepper.moveTo(40000);
    myservo.write(5);
    z++;
    }
   }
     qtiRead=digitalReadFast(QTI);
pill=digitalReadFast(Load_Cell);
    if (pill==1 || qtiRead==0){
     Linkage_stepper.stop();
     Linkage_stepper.setCurrentPosition(0);
     digitalWrite(Linkage_Reset,LOW);// Turn off linkage if pill is detected

     Serial.println("Pill Dispensed");
     pill=0;
     qtiRead=1;
     dispensed++;
       myservo.write(70);
       usbRead=0;
       x=0;
       y=0;
       input=0;
       set=1;
       z=0;
         if (dispensed==3){
           set=0;
           dispensed=0;
         }
    }
  }
}
void Rotate(){

  Rotary_stepper.moveTo(cartPos[usbRead-10]);
  //set=3;
  if(Rotary_stepper.distanceToGo()==0){
    set=1;
    Serial.println("rotating finished");
  }
}
```

Appendix D. Mechanical

I. Device Drawing

The device drawing consists of an isometric view of our overall CAD file. It gives a small glimpse into the mechanical aspects of the project, a short list of a bill of materials and labeling for each individual portion.

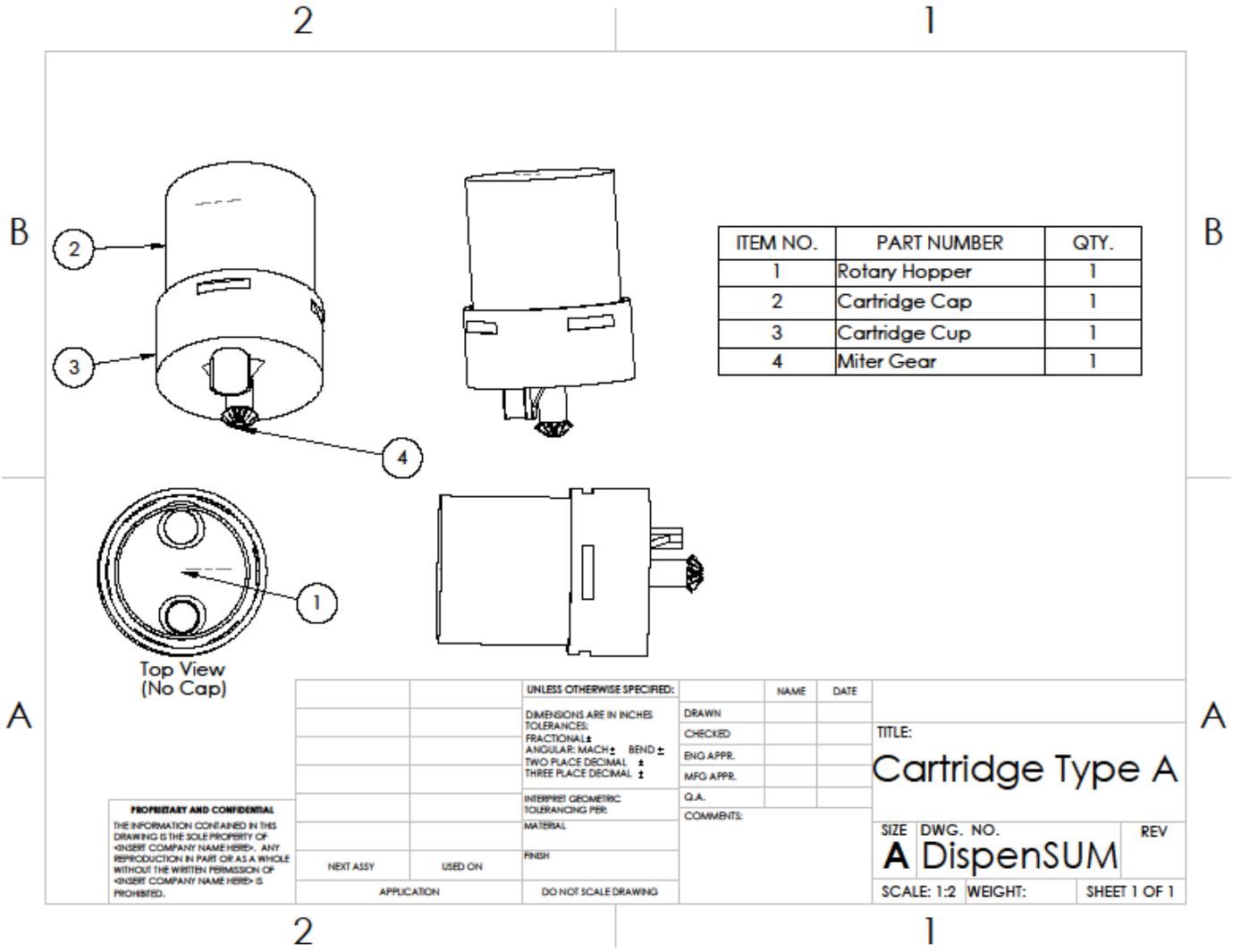


ITEM NO.	PART NUMBER	QTY.
1	14in T-Slotted Aluminum	8
2	Corner Bracket	8
3	16in T-Slotted Aluminum	4
4	Corner Bracket	4
5	Rotary Bearing Plate	1
6	Rotary Plate	1
7	150 Tooth Ring Gear	1
8	Cartridge Bracket	8
9	Cartridge for Larger Pills	4
10	Cartridge for Medium Pills	4
11	Homing Plate	1
12	RPI Touch-Screen Display	1
13	RPI Frame	1
14	Button Mount	1
15	Rotary Motor Mount	1
16	Load Cell Mount	1
17	Weigh Plate	1
18	Pill Chute	1
19	Nema 17 Motor	2
20	Nema 17 Bracket	2

UNLESS OTHERWISE SPECIFIED: NAME DATE
DRAWN
CHECKED
ENG APPR.
MFG APPR.
Q.A.
COMMENTS:
TITLE:
SIZE B DWG. NO. DispenSUM REV
SCALE: 1:1 WEIGHT: SHEET 1 OF 1
DIMENSIONS ARE IN INCHES
TOLERANCES:
FRACTIONAL ±
ANGULAR: MACH ± BEND ±
TWO PLACE DECIMAL ±
THREE PLACE DECIMAL ±
INTERPRET GEOMETRIC TOLERANCING PER:
MATERIAL
FINISH
NEXT ASSY USED ON
APPLICATION DO NOT SCALE DRAWING
PROPRIETARY AND CONFIDENTIAL
THE INFORMATION CONTAINED IN THIS DRAWING IS THE SOLE PROPERTY OF <INSERT COMPANY NAME HERE>. ANY REPRODUCTION IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF <INSERT COMPANY NAME HERE> IS PROHIBITED.

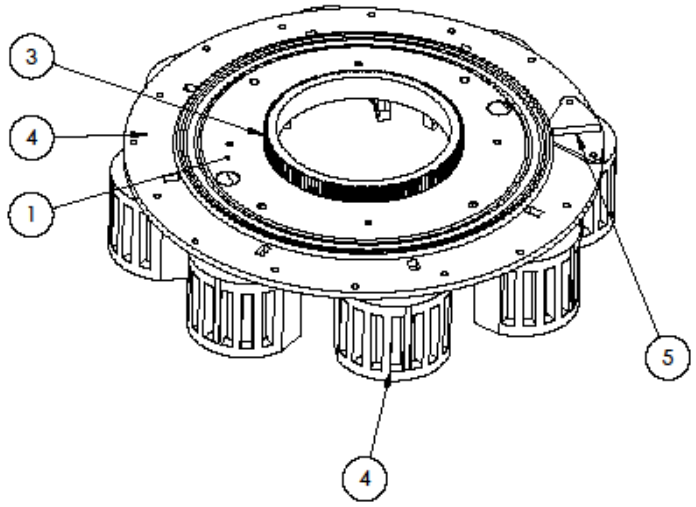
II. Cartridge Drawing

The cartridge drawing gives a couple views of the cornerstone of the device, the cartridge. It identifies some of the key components of the cartridge based design.



III. Rotary Carousel Drawing

This drawing demonstrates a key part of our design, the rotary carousel. It shows some of the key components of that subassembly.



ITEM NO.	PART NUMBER	QTY.
1	Rotary Bearing Plate	1
2	Rotary Plate	1
3	150 Tooth Ring Gear	1
4	Cartridge Bracket	8
5	Homing Plate	1

UNLESS OTHERWISE SPECIFIED:			NAME	DATE	
DIMENSIONS ARE IN INCHES TOLERANCES: FRACTIONAL ± ANGULAR: MACH ± BEND ± TWO PLACE DECIMAL ± THREE PLACE DECIMAL ±		DRAWN			TITLE: Rotary Carousel
		CHECKED			
		ENG APPR.			
		MFG APPR.			
INTERPRET GEOMETRIC TOLERANCING PER:		Q.A.			SIZE: A — DWG. NO.: DispenSUM — REV
MATERIAL		COMMENTS:			
FINISH					
NEXT ASSY	USED ON				SCALE: 1:8 — WEIGHT: — SHEET 1 OF 1
APPLICATION	DO NOT SCALE DRAWING				

Nicholas Rarick

Neuspeed9288@yahoo.com

OBJECTIVE

Looking for a career position in electrical engineering.

EDUCATION

In progress: B.S. Electrical Engineering, Control systems, California State University, Sacramento

Related Courses

Electronic I & II	Systems and Signals	Network Analysis
Electromechanical Conversion	Introduction to Microprocessors	Intro to Logic Design
Introduction to Feedback Systems	Applied Electromagnetics	Modern Communication Systems
Robotics	CMOS & VLSI*	Digital Control Systems*
Machine Vision*		

** In progress as of Spring 2017*

SKILLS

Programming:
Experienced in microcontroller and logic systems use - C / Arduino / Verilog / PIC18 / FPGA / X86 Assembly

Software:
Analog / digital circuit design, simulation, and hardware experience - Multisim / Orcad PSPICE / Agilent ADS / Matlab / Visual Studio/ Solidworks

Computer Skills:
Very strong Windows computers skills and networking with experience in a variety of software - Microsoft Excel / Microsoft Word / Vizio / Powerpoint

Tools:
Many hours of experience in a laboratory environment working with the necessary tools - Oscilloscopes / Function Generators /Logic Analyzer

Communication:
Strong communication skills, able to work well in teams to easily communicate complex information.

PROJECTS

Senior Design - Medication Adherence Aid
Designing and prototyping a unique medication adherence aid with many electromechanical systems. Device uses a multitude of sensors programmed for maximum device redundancy controlled with microcontrollers. Currently in progress.

Temperature and Humidity Room Control System
Developed a temperature and humidity control system using an Arduino microcontroller. Uses a variety of sensor inputs with complex control logic to control a relay system to keep the room stable.

Autonomous Robot
Created an autonomous robotic car that utilizes PID control systems to do a variety of tasks. Utilized many sensors to do such tasks as navigate a maze, follow a moving target at a specific distance, and navigating a bridge while not falling off while stopping for obstacles.

ACCOMPLISHMENTS and ACTIVITIES

- State Major GPA of 3.78
- Deans Honor List - Fall 2015, Spring 2016 , Fall 2016
- Member of Tau Beta Pi
- Member of IEEE

Benjamin Deubel Green

CAREER OBJECTIVE

To operate as an integral part of a team, working with and learning about the latest technology to improve the future for all people.

EDUCATION

California State University, Sacramento (Expected Graduation: Spring 2017)
Bachelor of Science: **Electrical and Electronics Engineering**
Concentration and Academic Interests: **Controls, Robotics, and Medical Technology**
Cumulative GPA: 3.756 Major GPA: 3.821

CSUS DEAN'S HONOR LIST: Fall 2013, Fall 2014, Fall 2015, Spring 2016, and Fall 2017

Associates Degree in Commercial Music Recording with Highest Honors, American River College, 2012
Associates Degree in Commercial Music Business with Highest Honors, American River College, 2012

RELEVANT COURSEWORK

EEE 178 – Intro to Machine Vision
EEE 187 – Robotics
EEE 108 and 109 – Electronics I & II
EEE 184 – Intro to Feedback Systems
EEE 188 – Digital Control Systems
EEE 193A and 193B – Product Design Project

TECHNICAL SKILLS

Proficient with Excel – Organizing, tabulating, and making complex visual representations of data
Experience with Lab Testing Equipment – Oscilloscope, function generator, and digital testing equipment
ORCAD PSpice
Multisim
MATLAB & Simulink
C Programming
Solidworks

AFFILIATIONS

Recruitment Officer • Institute of Electrical and Electronics Engineers, Sacramento Chapter 2016 – Present
Member • Power Engineers Society 2015 – Present
Member • Tau Beta Pi Engineering Honor Society 2017

WORK EXPERIENCE

Café Bernardo and the Berkley Bar • Sacramento, California (2015 - 2016)
Position: Bartender and Lead Server

Stirling Bridges • Sacramento, California (2014 - 2015)
Position: Lead Server and Server Trainer

The Old Spaghetti Factory • Rancho Cordova, California (2004 - 2014)
Position: Server Trainer and Bartender Trainer

VOLUNTEER WORK

Juvenile Diabetes Research Foundation • Sacramento, California
Event: Walk to Cure Diabetes (2011-2013)

- Coordinated and directed the main stage audio operation

DANA S. NATOV

1859 Imperial Ave, Davis CA 95616 (530) 219-0995 dana.natov@gmail.com

EDUCATION

CSU Sacramento 2013-Present

- Bachelors of Science, Electrical and Electronic Engineering – *Graduating May 2017*
- Academic Interests: Analog/Digital Control Systems, Robotics
- Cumulative GPA: 3.502 Major GPA: 3.56
- Dean's Honor Roll

WORK EXPERIENCE

Gold Standard Diagnostics - Manufacturing Intern 2012-Present

- Assisting in manufacture of the *Thunderbolt*, a medical diagnostic robot

Responsible for the production of multiple sub-assembly parts including stepper motors, PCBs, chemical luminescence hardware, assembly frames, mechanical rails and cabling. Familiar with strict regulations regarding Good Manufacturing Processes, Code of Federal Regulations, FDA auditing, quality and manufacturing. Trained in soldering, programming and troubleshooting of multiple sub-assemblies of the final product.

SKILLS

- Soldering, SMD Soldering, PCB Programming, PCB Troubleshooting
- Programming: Python, C, x86 and PowerPC Assembly, VHDL, G Code, OpenCV
- Software: Solidworks, MATLAB, PSpice, Multisim, Cadence Virtuoso
- Network analysis, AC/DC Circuits, Transmission Lines, Amplifier Circuits, BJTs, MOSFETs, Control Systems, PID Loops
- Internetworking, Subnets, Routing
- Machining: Mills, Lathes, CNC, Routers, 3D Printing

PROJECT EXPERIENCE

FRC team 1678: Citrus Circuits 2012-2015

- Student from 2012-2013, Mentor from 2013-2015
- 2015 Season World Champions

Senior Design

- One year design project, in progress, requiring first semester prototype and second semester deployable prototype
- Automated in-home medication dispenser with a custom cartridge design allowing automatic uploading of medication schedules.

PROFESSIONAL ORGANIZATIONS

- IEEE Member\CSUS IEEE Club Officer 2015-Present
- Tau Beta Pi Engineering Honor Society Member 2016-Present

Nael Numair

EDUCATION: *In progress:* B.S. Compute Engineering, California State University, Sacramento Expect Graduation May 2017

Courses:

Computer Hardware Design	Data Structures and Algorithm Analysis	Computer Interfacing
Computer Networks and Internets	Network Analysis	Signals and Systems
Discrete Structures	Micro Computers and Assembly	Advance Logic Design
Electronics I	Operating System Principles	Advance computer Origination
CMOS and VLSI		Operating System Pragmatics

PROJECT EXPERIENCE

LED Gaming Board

This was a team project in which we used an arduino uno to create a control unit for and 8x8 LED matrix. My job was to construct the matrix and wire it up to the board. The object of the matrix was to play "Snake".

VPN Tunnel

I was able to create a hard line VPN tunnel connecting a bay area business's two offices together.

Senior Project Smart Pill Dispensing Box:

The objective of this project is to create a smart pill box. Currently my job right now is to program all controls that will be used in this project. Micro controllers that I am currently using are the Raspberry Pi 3 and Arduino uno. Created an automated mail messaging system. I was also responsible to creating a way to transfer information from one system end to another using RFID communication. I also created how the dispensing schedule would be handled using python. Finally it was my job to do the top level system integration between the system GUIs, background functions and mechanical components.

KNOWLEDGE AND SKILLS

Communication/Organization/Leadership:

I am in charge of coordinating events and organizing the "SWAT" team for a bay area martial arts school. Leading and motivating other students in training.

Operating Systems:

Windows XP, 7, 8, 10, Unix and Linux

Computer Languages:

C#, Java, Python, x86, visual basic, C, VHDL, Verilog

Hardware/Software:

MS Word, Powerpoint, Excel, Matlab, Eclipse, Visual Studio, Adobe Photoshop, Adobe Indesign, pinnacle studios, analog discovery, Mulitsim, PSpice, Candance Virtoso

Tools:

Oscilloscope, Function generator, Multimeter, Raspberry Pi, Parallax Propeller, Arduino

WORK EXPERIENCE:

Instructor	***Tony Ramos Kajukenbo***	5/10 to present
Free Lance Intern	***Tutis Group***	6/09 to present

ACTIVITIES AND ACCOMPLISHMENTS:

- Obtained my instructorship, 1st Degree Black Belt and 1st Degree Black Sash in Kajukenbo
- Became a "Shadow" candidate for CSUS chapter of ACM
- Treasure for IEEE chapter at CSUS 2016-2017

Working 6 hours per week, while carrying 15 units per semester and maintaining a 2.8 GPA

Jennifer Ong

EDUCATION

In progress: B.S. Computer Engineering, California State University, Sacramento
Expected graduation: May 2017 (Current GPA: 3.415)

HONORS & AWARDS

Deans Honors List, CSUS – Fall 2012/2016, Spring 2013
Big Sky Conference All-Academic Award – 2013, 2016
Intercollegiate Tennis Association Scholar-Athlete - 2013

RELEVANT COURSEWORK COMPLETED OR IN PROGRESS

- Programming Concepts & Methodology I & II
- Introduction Computer Architecture
- Introduction System Program Unix
- Signals & Systems
- Computer Interfacing
- Microcomputers and Assembly Language
- Operating Systems Principles
- Advanced Logic Design
- Database Management Systems
- Computer Network + Internet

PROFESSIONAL SKILLS

- **Programming languages:** C, x86 Assembly, Verilog, VHDL, Java, SQL
- **Operating Systems:** MS Windows, Unix, Linux
- **Software:** Quartus, Xilinx ISE, Multisim, MS Office
- **Tools**: Diligent Analog Discovery, oscilloscope, function generator, Amani GTX CPLD, Arduino Platform, Spartan 3E board, Microchip PICkit 3, Parallax Propeller, Raspberry Pi.
- **Bilingual:** English and Vietnamese (intermediate)

PROJECT EXPERICENCE

- **Android TV:** Two-week work experience project at Lime Rocket (application developer). Learnt how to created a custom boot loading for the Android TV by changing the boot logo in kernel.
- **Remote Controlled Vehicle:** Member of a three-member team that designed, developed, and implemented a remote controlled vehicle. The Raspberry Pi was used to implement the design. A third party application enabled live video streaming that could be viewed on a computer or phone. Sensor was implemented to detect oncoming objects.
- **Multiplier, *logic design*:** Designed, developed, and implemented the logic that multiples a 4-bit by 4-bit. The design was coded in Verilog and implemented on the Spartan 3E board.
- **Banking Program:** Designed a graphical user interface (GUI) that allows users to make deposits and withdraws from a bank account. Balance of bank account is recorded and displayed to the user. The design was coded using Java programming language.
- **Automatic Pill Dispenser - Senior Design Project:** Developing hardware and software project planning and engineering design skills using the system design methodology. Gained experience in design philosophies, problem definition, project planning, budgeting, written and oral communication skills, working with others in a team arrangement, development of specifications and effective utilization of available resources. My role in the project include interfacing different components to a microcontroller, and creating graphical user interfaces.

WORK EXPERIENCE

Deloitte Consulting LLP

2016 Business Technology Analyst Summer Scholar Program. Worked in the Systems Integration service line on a Public Health Services project. Part of the Functional Team that focused on assisting the client provide an electronic exchange of patient's medical information. Analyzed and documented existing processes, gained knowledge on Agile methodology, determined business requirements for several different processes, and created a functional design specification document for a website.

COMMUNICATION/ORGANIZATION

- Robust analytical, problem solving and writing skills developed through composing laboratory reports
- Strong communication and leadership skills developed through constant interaction with team members
- Reliable and self-motivated. Strong ability to quickly adapt and react to unexpected circumstances

ACTIVITIES & ACCOMPLISHMENTS

- Full Division I Athletic Tennis Scholarship - Sacramento State University (2012-16)
- ***Sacramento State University, Women's Tennis Team (2012-16)***: #1 player on the team (2014), Team captain (2014-16)