Pill-Matic Product Design Specification

Rev. 1.1

Connor Dueck , Devon Louie, Adam Gabriel, Jerry Yao, Peter Hsu
ENSC305W, ENSC440W

10/16/03

Contact Person:
Connor Dueck
cdueck@sfu.ca
604.839.5671

Health-Assist

Table of Contents

List of Figures

List of Tables

Abstract

This document provides an outline of the design specifications for Health-Assist's Pill-Matic, an automated pill dispenser and reminder system. The purpose of this document is to provide technical information and design requirements for each component of the Pill-Matic to the user. The design specifications of the Pill-Matic has been divided into software, hardware, and firmware sections.

The software section covers the design approach and functionality of our companion smartphone application. Our software team decided to develop in Python with a Kivy library, using Buildozer to compile into an android application. The final product will include the following screens: Configuration, Menu, Lock, Log, Add Pills, Calendar and Options.

The hardware design section includes technical specifications and design approach for each component of the Pill-Matic prototype including Storage Disk Rotation, Calibration, Vibration, Break Beam Sensor, and Motor/Sensor Controller. The main storage disk is capable of holding eight distinct medications, while DC motors are responsible for the disk rotation required in the dispensing of medication. Before dispensing, a photointerrupter is utilized to insure that the main storage disk is calibrated at the zero position. The Pill-Matic then uses vibration such that the medication lines up correctly and dispenses one pill at a time. As a pill leaves the main storage disk, it is detected by the break beam sensor, which then communicates to the vibration motor to stop. Finally servo motors are responsible for closing the main storage disk to stop the output of additional pills.

The firmware section outlines the design choice and responsibilities of our microcontroller. The Raspberry Pi firmware is written in the C++ programming language and its graphical user interface will be developed using PyGobject. The Raspberry Pi will communicate with the Arduino Pro Micro in order to control the motors specified in the hardware section.

Finally, the system test plan consists of detailed test cases to insure that our prototype meets all requirements specified in previous documents. Health-Assist expects to have a fully functional prototype adhering to the following design specifications by April 1st, 2016.

1. Introduction

Pill-Matic is an automated pill dispenser and reminder system designed for patients with multiple daily prescriptions. Pill-Matic will be linked to a mobile device through an Android application capable of communicating pill schedules, dosages, and missed alerts. The design of our product has been divided into software, hardware, and firmware sections. This document provides the technical specifications for each component of the Pill-Matic.

1.1 Scope

This document is intended to describe the design specifications of the Pill-Matic and any related software. All design choices are made with careful consideration for justifications based on the Pill-Matic's functional specifications in the hopes of creating a simple and concise product scheme. All details within this document will be applied primarily to our proof-of-concept prototype, rather than for a mass production model. There are however, some considerations within this document that could be used in a finalized product. The designs themselves will focus mainly on covering higher-level design details, but might dabble on particulars if need be.

1.2 Intended Audience

The primary users of the design specifications is intended for the members of the Health-Assist team, who plan to use this document as a guiding reference for any design considerations that they will have in the development and making of the Pill-Matic prototype. Any future engineers who wish to make use of this document in its entirety can also use any of its contents for further ideas and improvements if they wish to.

2. Software

2.1 Software Overview

In order to develop the mobile application, we sought out a development tool that was cross-platform, easy to use and accommodated with variety of built in functions. Table [1] shows a comparison of the tools we researched. Our conclusion led us to use Python with Kivy library and Buildozer to compile into an android application. Python by nature is a simple language to program in. The Kivy library contains a variety of tools that allows us to easily develop a GUI that is cross-platform. Both Kivy and Python have active community members that are constantly developing new tools, thus we will not lack in terms of additional functionalities.

Table [1]**: A comparison of different development tools* ***[1],[2],[3],[4]

Development Tool (Language)	Complexity	Cross-Platform	Extensive functionality
Kivy (Python)	Simple	✓	✓
Corona SDK (Lua)	Simple	✓	✗
Android Studio / SDK (Java)	Complex	✗	✓
Xcode (Interactive, C)	Simple	✗	✓

2.2 Purpose

The purpose of the mobile application is to provide extra assistance to the user while using Pill-Matic. This is achieved by providing data, schedules, and alarms to the user independently from our product. However, in order to obtain these information, the phone must communicate with Pill-Matic via Bluetooth. Once a connection has been established, they will acknowledge each other and regularly exchange information when the phone's Bluetooth is in range of Pill-Matic. Pill-Matic will uniquely identify each phone it connects with by their phone number. Likewise, the phones will uniquely identify Pill-Matic through a product number. Having a unique identifier is important when multiple Pill-Matic machines are linked to multiple mobile devices.

2.3 Breakdown

The application is divided into the following screens: Configuration, Menu, Lock, Log, Add Pills, Calendar and Options. Figure [1] provides an overview of the application and how these screens interact with each other. Due to Kivy's cross-platform features, the prototype screens will be shown using Windows.

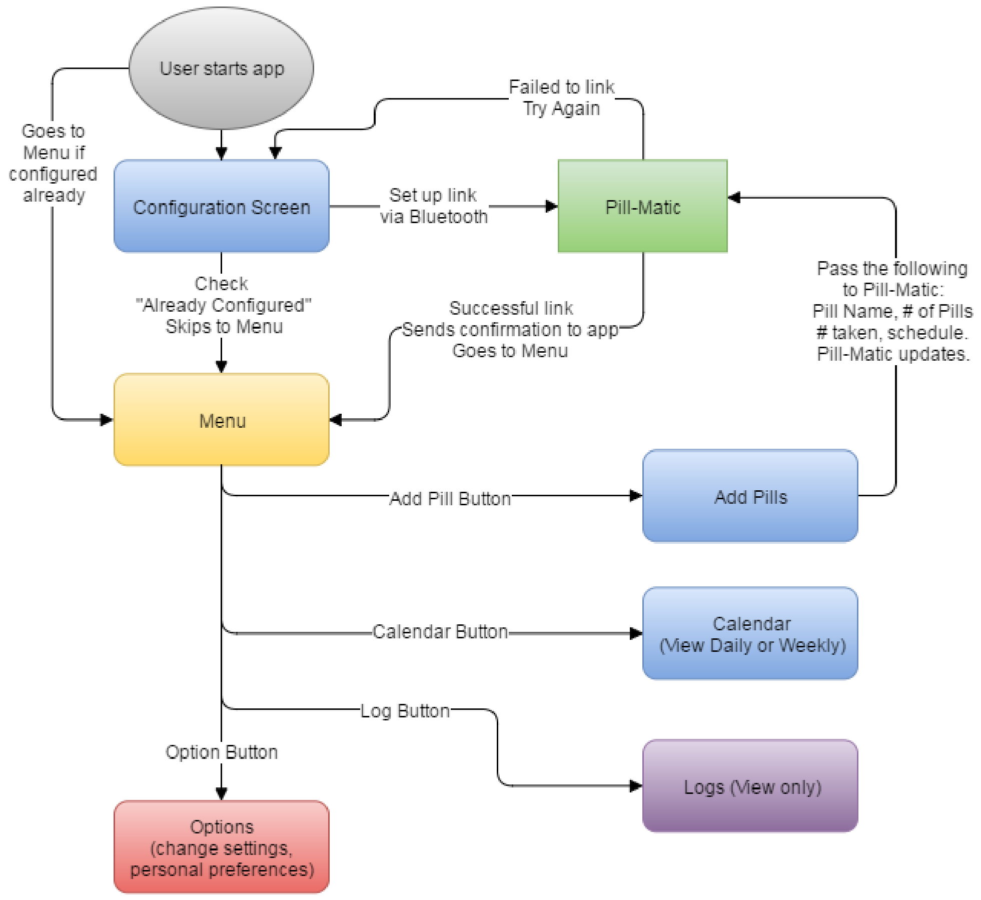

User starts app
Goes to Menu if configured already
Failed to link Try Again
Configuration Screen
Set up link via Bluetooth
Pill-Matic
Check "Already Configured" Skips to Menu
Successful link Sends confirmation to app Goes to Menu
Pass the following to Pill-Matic: Pill Name, # of Pills # taken, schedule. Pill-Matic updates.
Menu
Add Pill Button
Add Pills
Calendar Button
Calendar (View Daily or Weekly)
Log Button
Logs (View only)
Option Button
Options (change settings, personal preferences)

Figure [1]: *Breakdown of the smartphone application*

2.4 Configuration Screen

The configuration screen is a setup screen that a user should see when they have just installed the application and ran it for the first time. Its role is to ensure that the user establishes a connection with Pill-Matic so they can acknowledge each other in order to exchange information later on. By clicking "Connect", the application will attempt to search for a Pill-Matic machine and connect to it. If it cannot find a machine, it will prompt an alert message notifying the user. At the bottom, there is a checkbox that allows the user to skip this step. This is only to be used if the Pill-Matic is pre-configured to locate this particular phone. Once this has been setup, this page should not appear again until the phone is needed to connect to a new Pill-Matic machine or it has lost its saved data.

2.5 Menu Screen

This is the main screen for the application. From here, the user has four options to choose from the view and manage the data received from Pill-Matic.

2.6 Lock Screen

This is a protective screen to prevent accidental changes to the application without the user being aware.

2.7 Add Pills Screen

This screen allows the user to add pills remotely with their application. The user needs to eventually be around Pill-Matic for the phone to send this information to the machine. It allows the user to configure the type of medications and also allows them to program a schedule. A prototype visual representation of this screen is shown in Figure [2].

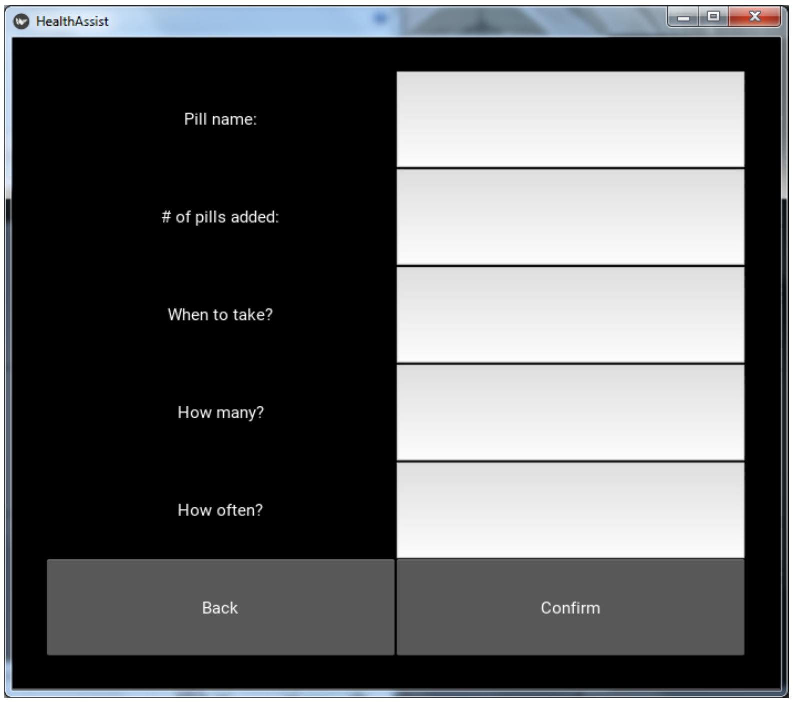


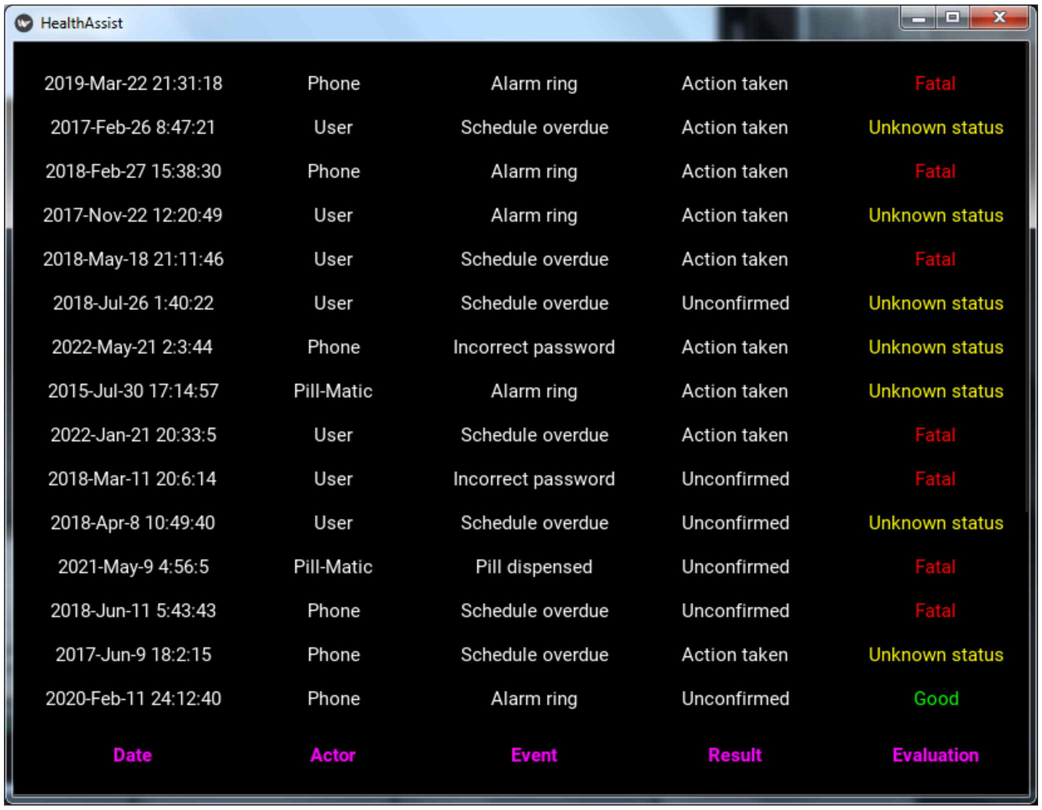
Figure [2]: *Prototype GUI of the Add Pills Screen*

2.8 Calendar Screen

This screen provides a daily or weekly view of their pill dispense schedules. The daily view will display dispense schedules and alarm schedules by the hour vertically. The weekly view is an extension of the daily view with the whole week's display.

2.9 Log Screen

This screen is meant for the user to view activities from Pill-Matic, the phone and their own actions. It is an extra layer of safety measures to ensure that if errors are to occur on Pill-Matic or the application, there will be a trace as to what went wrong. It can also be used to track how the user has been using the product to determine if they are abusing the product at any point or time. A sample display can be seen in Figure [3].



2019-Mar-22 21:31:18	Phone	Alarm ring	Action taken	Fatal
2017-Feb-26 8:47:21	User	Schedule overdue	Action taken	Unknown status
2018-Feb-27 15:38:30	Phone	Alarm ring	Action taken	Fatal
2017-Nov-22 12:20:49	User	Alarm ring	Action taken	Unknown status
2018-May-18 21:11:46	User	Schedule overdue	Action taken	Fatal
2018-Jul-26 1:40:22	User	Schedule overdue	Unconfirmed	Unknown status
2022-May-21 2:3:44	Phone	Incorrect password	Action taken	Unknown status
2015-Jul-30 17:14:57	Pill-Matic	Alarm ring	Action taken	Unknown status
2022-Jan-21 20:33:5	User	Schedule overdue	Action taken	Fatal
2018-Mar-11 20:6:14	User	Incorrect password	Unconfirmed	Fatal
2018-Apr-8 10:49:40	User	Schedule overdue	Unconfirmed	Unknown status
2021-May-9 4:56:5	Pill-Matic	Pill dispensed	Unconfirmed	Fatal
2018-Jun-11 5:43:43	Phone	Schedule overdue	Unconfirmed	Fatal
2017-Jun-9 18:2:15	Phone	Schedule overdue	Action taken	Unknown status
2020-Feb-11 24:12:40	Phone	Alarm ring	Unconfirmed	Good
Date	**Actor**	**Event**	**Result**	**Evaluation**

Figure [3]: *A sample of the Log Screen with labels at the bottom*

2.10 Options Screen

This screen provides various settings allowing the user to customize the application to their own preference. There is a slight distinction in terms of user level. Phones with elevated user levels will have the option to override schedules and send an immediate request for Pill-Matic to dispense a pill. This is meant to be used as an emergency measures only.

2.11 User Privileges

Some of the features on the mobile application are only accessible through an elevated user. Being able to dispense a pill at will is helpful in emergencies, but can be abused. Thus, promoting a phone to be a privileged user can only be performed on the particular Pill-Matic machine that the user is attempting to gain privilege to. Accessing these functions will also be protected by a password that the user must enter each time to prevent abusive use.

Figure [4] demonstrates an example of how two Pill-Matic machines and two Phones could function. If Pill-Matic 1 is configured to link and elevated Phone A, then the user will be able to use privileged functions on Phone A for Pill-Matic 1 and view regular information that the phone receives from Pill-Matic 1. Likewise, if Pill-Matic 2 is configured to link both Phone A and B, then the user can view information on Pill-Matic 2 but will not be able to access privileged function through either phones. Table [2] provides additional clarification as to what the privileged user is entitled to.

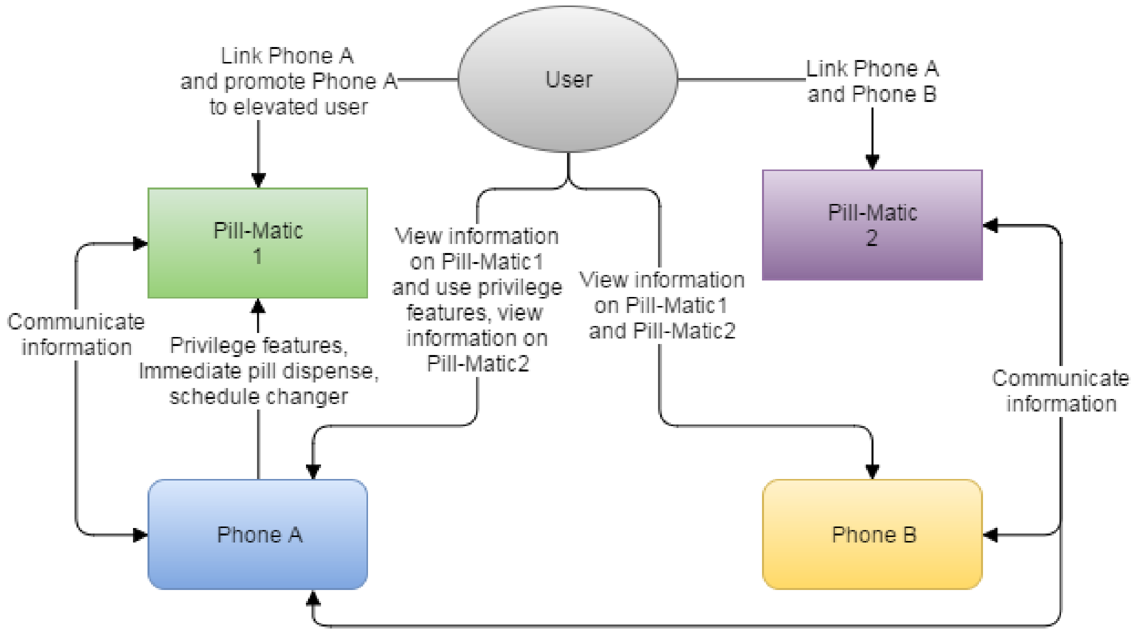

User
Link Phone A and promote Phone A to elevated user
Link Phone A and Phone B
Pill-Matic 1
Pill-Matic 2
View information on Pill-Matic1 and use privilege features, view information on Pill-Matic2
View information on Pill-Matic1 and Pill-Matic2
Communicate information
Privilege features, Immediate pill dispense, schedule changer
Communicate information
Phone A
Phone B

***Figure [4]**: A sample interaction of two Pill-Matic machines and two phones*

***Table [2]**: A comparison between regular and elevated user on functions*

Functions	Regular User	Elevated User
Add Pills	✓	✓
View Calendar/Schedule	✓	✓
View logs	✓	✓
Update Calendar/Schedule	✗	✓
Immediate pill dispense	✗	✓

3. Hardware

3.1 Hardware Overview

Figure [5] provides an overview of the Pill-Matic's hardware design. Note that servo motors and vibration motors are not shown.

The main storage disk, shown in white, is responsible for holding up to eight separate medications of any size or shape. The disk was designed to rotate 360 degrees to minimize the number of dispensing mechanisms as well as making it easier for the user to add medications to the system. Vibration is used to move pills throughout the disk.

To keep pills from being dispensed at the wrong time or being moved throughout the machine during storage disk rotation, an outer ring and gate were designed. The gate is operated via a small servo motor and is used to open or cover an opening in the outer ring. The upper grey section in Figure [5] shows the mechanics of the ring and gate.

As pills are dispensed one at a time, they end up in a temporary holding section, lower grey element, built into the Pill-Matic. Only after the user has requested their medication and entered a security code will the pills be released from this section and into the medication cup.

Individual components of the Pill-Matic will be discussed in detail and can be found in their respective sections.

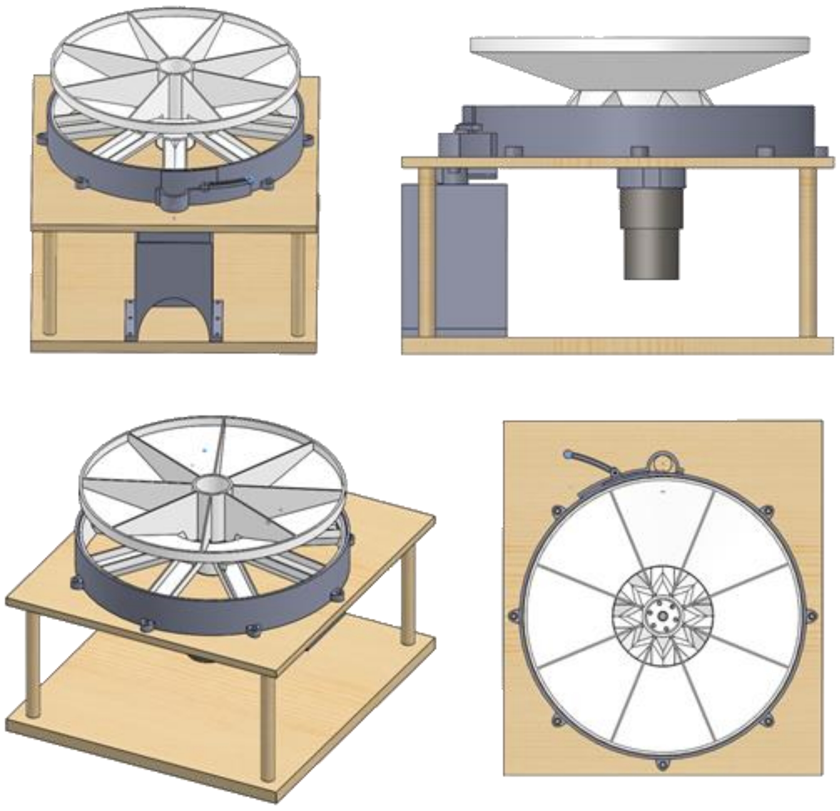


Figure [5]: *Overview of the Pill-Matic's hardware design*

3.2 Storage Disk Rotation

While designing for the main storage disk rotation, three motor options were considered:

- Servo motors
- Stepper Motors
- DC motors

Servo motors, the simplest option to implement, with their built in feedback were ruled out because of their 180 degree rotational limit. Stepper motors, although extremely accurate, did not allow for any feedback to the motor controller. If for some reason the storage disk was restricted from rotating, the motor controller would not be able to detect it and could result in the incorrect medications being dispense. The low cost of DC motors, as well as the option to purchase them with built in incremental encoders were the deciding factors in the design.

Specifications such as rotational speed, torque, power, noise, cost, and motor size were all considered when determining which DC motor to use in the final design. To keep the rotation of the disk from affecting the stored medication and dispensing mechanism, a low rotational speed was needed. To cope with friction and possibility of pills blocking rotation, a higher torque motor was required. Unfortunately, slow speed and higher torque result in higher gear ratios which directly contribute to the noise of the motor. The specifications for the DC motor used in the design are below in Table [3].

***Table [3]**: DC motor specifications*

Rated Voltage	Rated Torque	Rated Speed	Gear Ratio	Rated Current	Stall Current
12V	784mN*m	17RPM	200:1	410mA	1.8A

The motor purchased for this design is equipped with an incremental quadrature encoder. Two Hall Effect sensors, placed 90 degrees apart, are mounted to the motors rear shaft. Each revolution of the motors shaft results in 3 output pulses for sensor. Using both sensors and accounting for the motors gear ratio, the output signal from the encoder comes at a rate of 1800 pulses per main shaft revolution, resulting in a rotational accuracy of 0.2 degrees per pulse. More information on the quadrature encoder is available in Table [4].

***Table [4]**: Quadrature encoder information*

Operating Voltage	Resolution	Input Current	Output
4.5V to 5.5V	3PPR	50mA	200:1

By feeding both output waveforms from the quadrature encoder back into the motor controller, rotational speed, change in position and direction of rotation can be calculated. Through the use of a PID controller, the motor can be rotated to any position.

3.3 Calibration

To ensure the main storage disk is in the correct position at boot up and before each dispensing cycle, a calibration sequence is used. The storage disk is rotated slowly until a photointerruptor detects the zero position of the disk. This position is then recorded and referenced during future motor rotations.

3.4 Vibration

Controlling pill movement through the main storage disk is done through vibration. Objects naturally line up in a convenient orientation when succumb to vibrating materials, which the Pill-Matic takes advantage of to accurately dispense one pill at a time, despite varying pill shapes and sizes. Small DC motors with offset weights are used to vibrate the Pill-Matic's main storage disk.

Table [5]: *Vibration Motor Specifications*

Rated Voltage	Rated Current	Stall Current	Rated Speed
5V	70mA	120mA	3,00RPM

3.5 Break Beam Sensor

As a pill leaves the main storage disk, it must be detected by the motor controller which will then determine if the vibration motor must be disabled. Multiple options were researched to determine the best way of detecting when and how many pills leave the storage disk, but after much deliberation, the below solution was implemented.

Instead of using an off-the-shelf sensor, an infrared break beam sensor was designed. The custom sensor uses an infrared led to create a beam of light which is picked up by a 38kHz infrared receiver. As an object, such as a pill, breaks the infrared light beam, a signal is sent to our motor controller. In order for our infrared led to be picked up by our sensor, it must be pulsed at 38kHz. A schematic of the circuitry required to pulse the led is shown in Figure [6].

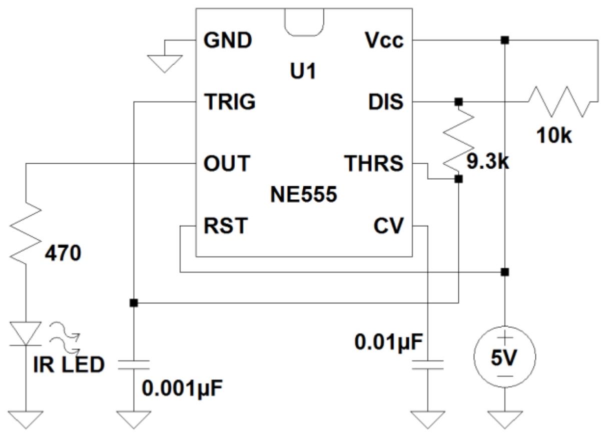


Figure [6]: *Infrared Break Beam Sensor Schematic*

Implementing a simple photointerruptor would have been a simper solution to this problem, but using our approach yielded many benefits. Because the sensor uses infrared light pulsed at a specific frequency, it is much less susceptible to interference which is important for a high accuracy medical device. Secondly, because the sensor is built from individual components and not packaged into one body, it can be used in a wide range of designs. This allows the hardware design to change and adapt without having to implement a different sensor.

3.6 Servo Motors

Along with the main storage motor and vibration motors, two small servo motor are needed for the design. The first motor is responsible for closing the opening between the storage disk and the output, which stops pills from being dispensed while the disk is rotating. The duty of the second servo is to dispense the medication after the user has requested their dose.

3.7 Motor/Sensor Controller

The motor and sensor controller board for the Pill-Matic is built around the Arduino Pro Micro. The microcontroller is responsible for reading and interpreting all sensor data as well as control the operations of all motors. Dispenser data such as pill location, number of pills to dispense and calibrate commands are sent from a Raspberry Pi to the Arduino.

The Arduino Pro Micro was selected because of its small size and the microcontroller it uses. The ATMEGA328U microcontroller, unlike most other Arduino board microcontrollers, allows for five hardware interrupts and five Pulse Width Modulation (referred to as PWM from now) outputs. The large number of hardware interrupts are necessary as many of our sensors run at high frequencies and the PWM outputs are needed to run servos and the main storage disk motor. Table [6] shows the pinouts of the Arduino.

Alongside the Arduino is a TB6612FNG dual output motor driver breakout board. This board, shown in Figure [7], is responsible for the 12V control of the main storage disk motor. Two digital inputs control the direction of rotation and a PWM input is required to set the rotational speed.

Table [6]: *Arduino Pro Micro pinout*

Arduino Pin	Function
Digital Pin 0 (Interrupt)	Encoder A input
Digital Pin 1 (Interrupt)	Encoder B input
Digital Pin 2 (Interrupt)	Break Beam input
Digital Pin 3 (Interrupt)	Break Beam input
Digital Pin 4 (Interrupt)	Photointerruptor input
Digital Pin 5 (PWM)	Servo 1 output
Digital Pin 6 (PWM)	Servo 2 output
Digital Pin 9 (PWM)	Servo 3 output
Digital Pin 10 (PWM)	Motor PWM output
Digital Pin 11 (Digital Output)	Storage Motor output A
Digital Pin 12 (Digital Output)	Storage Motor output B
Analog Pin 2 (Digital Output)	Vibration Motor output
Analog Pin 3 (Digital Output)	Vibration Motor output

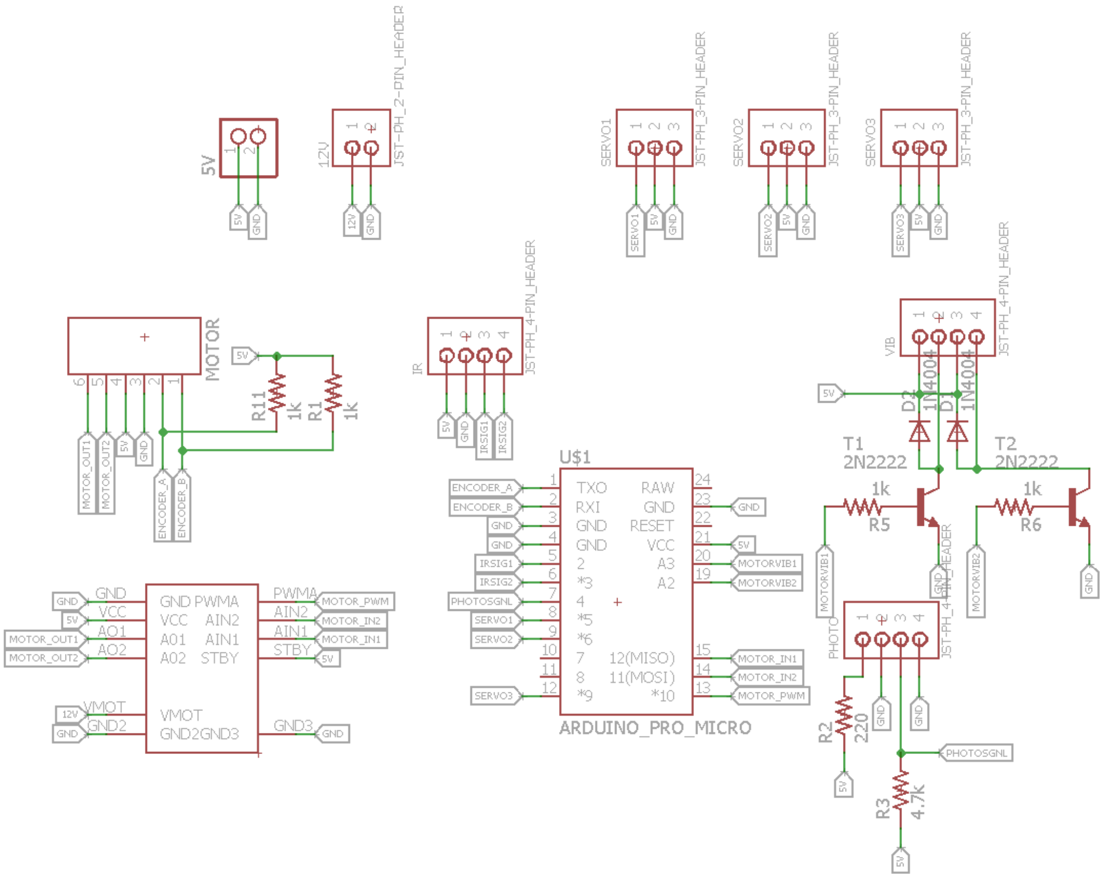


Figure [7]: *Motor/Sensor Controller Board Schematic*

3.8 Power Supply

The Pill-Matic will be powered by one 12V supply. The only component of the device which requires a 12V input is the main storage disk motor, which has a max current load of 1.8A. The rest of the components will receive power via a 5V regulator with a max output of 3A. A full diagram of the Pill-Matic power system is available on Figure [8].

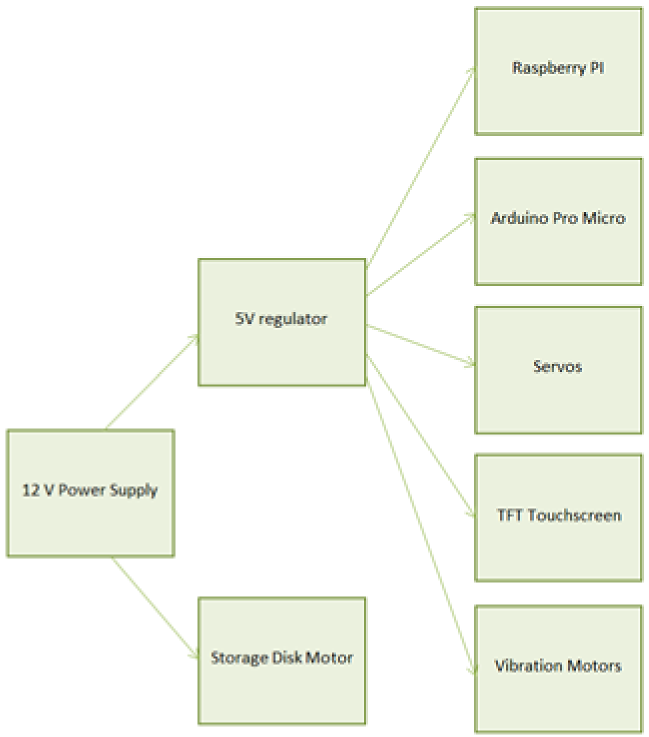


Figure [8]: *Overview of Pill-Matic Power System*

4. Firmware

4.1 Microcontroller

The Raspberry Pi 2 Model B board was selected for use as the platform for this project because of the ease of implementing the desired features. This includes Bluetooth, which can easily be added on via the USB ports on the Raspberry Pi, the acoustic notification system, which can be added on via the 3-Pole audio jack or the GPIO pins as well as the 3.5 inch TFT touch screen, which can be added on via the GPIO pins as well. In addition to the great adaptability of the platform, the Raspberry Pi 2 Model B has a 1.0GHZ Broadcom BCM2835 processor, which is more than powerful enough to power all the required systems as well as query for interrupts from the digital GPIO pins. Furthermore, the Raspberry Pi 2 only draws 4.0W while under full load, thus powering the device is possible from a very wide range of power supplies.

Physically, the Raspberry Pi 2 Model B fits in an area of 85.60 mm × 56.5 mm and weighs only 45g. This compact design saves space in the end unit, and since the health-assist device is very flat, it enables us to not have to add cavity space to store the microcontroller.

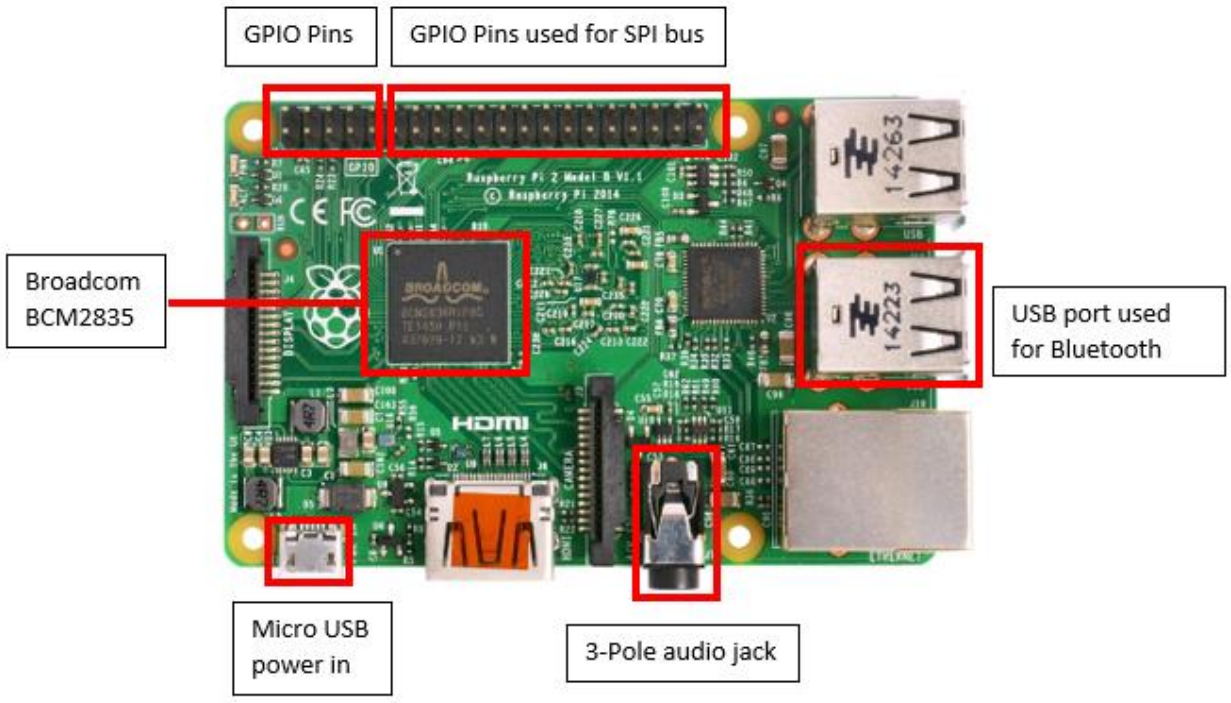


Figure [9]: *Raspberry Pi 2 Model B board*

Table [7]: *Specification of Raspberry Pi 2 Model B*

SoC	Broadcom BCM2835
CPU	ARM-A7 900 MHz (overclocked to 1GHz)
GPU	Broadcom VideoCore IV @250MHz
RAM	1GB (shared with GPU)
USB Ports (2.0)	4
Audio Output	3.5mm 3-Pole, HDMI
Storage	MicroSD
Network Adapters	10/100 USB Ethernet Adapter
GPIO	40× GPIO
Operating Voltage	5V via microUSB
GPIO Voltage (HIGH)	3.3V

Table [8]: *Raspberry Pi 2 Model B GPIO Pinout*

Pin #	Pint Type	Usage
3V3	Power	Screen SPI Bus
5v	Power	Screen SPI Bus
GPIO2	I2C	Screen SPI Bus
5V	Power	Screen SPI Bus
GPIO3	I2C	Screen SPI Bus
Ground	Ground	Screen SPI Bus
GPIO4	GPIO	Screen SPI Bus
GPIO14	UART0_TXD	Screen SPI Bus
Ground	Ground	Screen SPI Bus
GPIO15	UART0_RXD	Screen SPI Bus
GPIO17	GPIO	Screen SPI Bus
GPIO18	GPIO	Screen SPI Bus
GPIO27	GPIO	Screen SPI Bus
Ground	Ground	Screen SPI Bus
GPIO22	GPIO	Screen SPI Bus
GPIO23	GPIO	Screen SPI Bus
3V3	Power	Screen SPI Bus
GPIO24	GPIO	Screen SPI Bus
GPIO10	GPIO	Screen SPI Bus
Ground	Ground	Screen SPI Bus
GPIO9	GPIO	Screen SPI Bus
GPIO25	GPIO	Screen SPI Bus
GPIO11	GPIO	Screen SPI Bus
GPIO9	GPIO	Screen SPI Bus
Ground	Ground	Screen SPI Bus
GPIO7	GPIO	Screen SPI Bus
ID_SD	I2C ID	Not used
ID_SC	I2C ID	Not used
GPIO5	GPIO	Not used
Ground	Ground	Not used
GPIO6	GPIO	Not used
GPIO12	GPIO	Not used
GPIO13	GPIO	Not used
Ground	Ground	Not used
GPIO19	GPIO	Not used
GPIO16	GPIO	Pulse alarm
GPIO26	GPIO	Pulse alarm
GPIO20	GPIO	Pulse alarm
Ground	Ground	Not used
GPIO21	GPIO	Not used

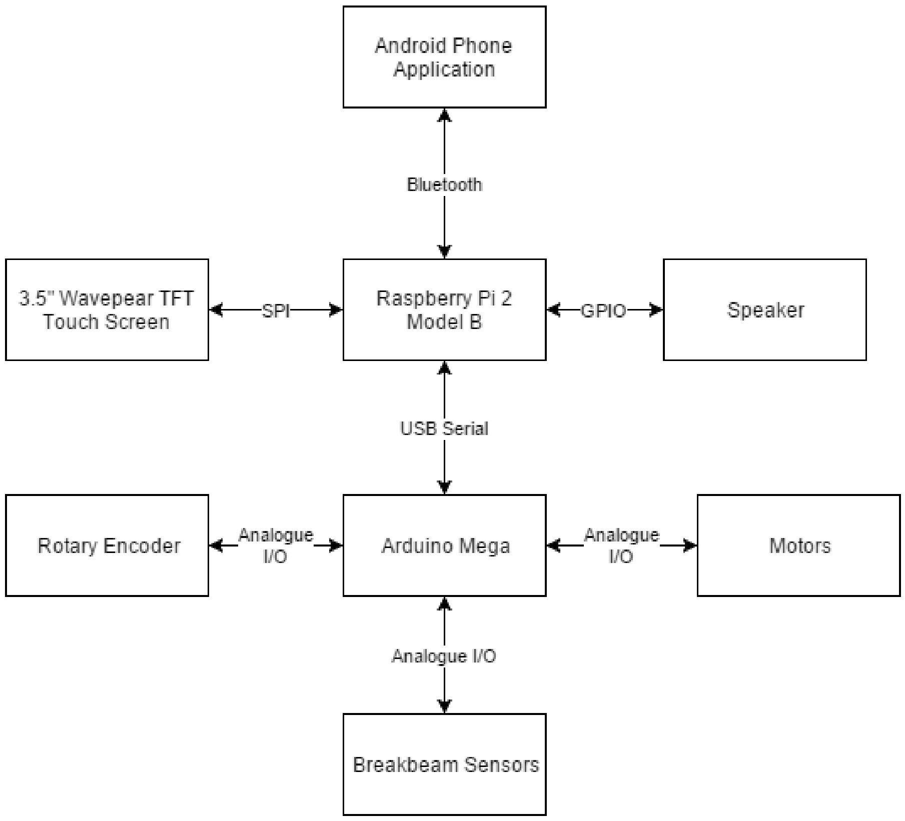


Figure [10]: *System Overview*

4.2 Raspberry Pi Firmware

The firmware for the Raspberry Pi will be written in the C++ programing language, using the latest G++ compiler. All programming and configuration is compiled on a development use Raspberry Pi, which is then used to produce a system image. This contains all programs, configuration and user data which was stored on the development device. From there the firmware can be easily loaded on to any device by simply copying the system image to a micro SD card. Firmware updates are done by removing the microSD card, saving user data, updating the micro SD card and copying the data back to it.

4.3 Graphical User Interface

The GUI for the Raspberry Pi will be created using PyGobject, due to the simplicity. This program allows for a GUI to be created which can call bash scripts and other executable programs. This will allow us to trigger events in the C++ code from the GUI, such as dispensing pills or logging into the device. PyGobject uses python, which is a well-known highly adaptable programming language used in a multitude of different applications.

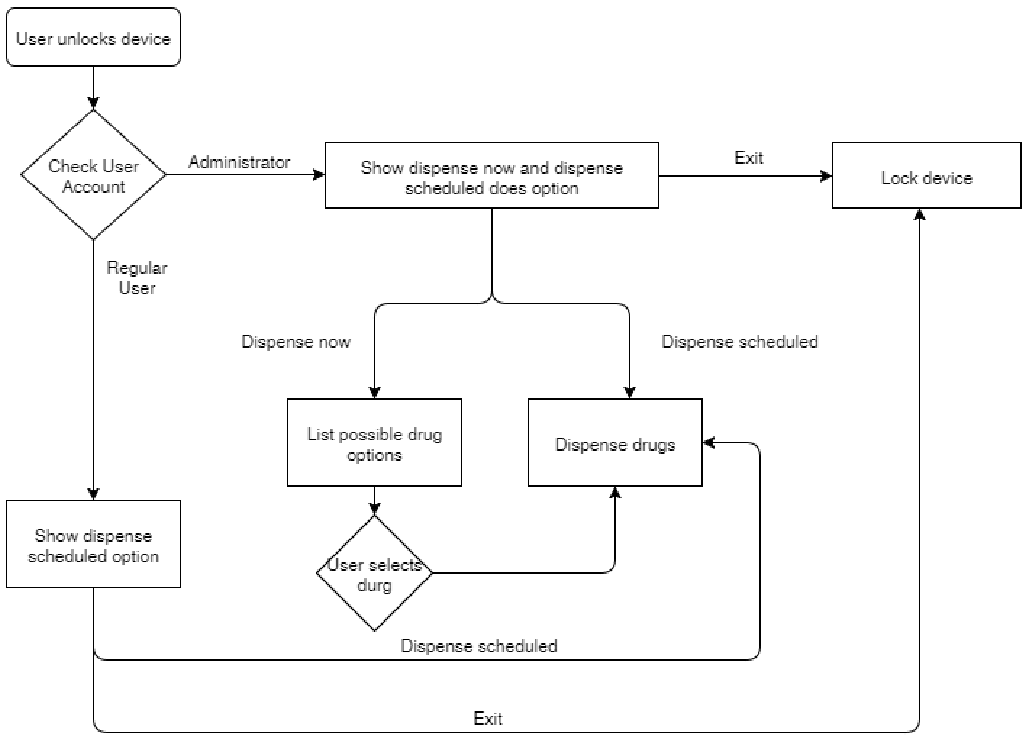


Figure [11]: *Flow chart of GUI when user is taking pills*

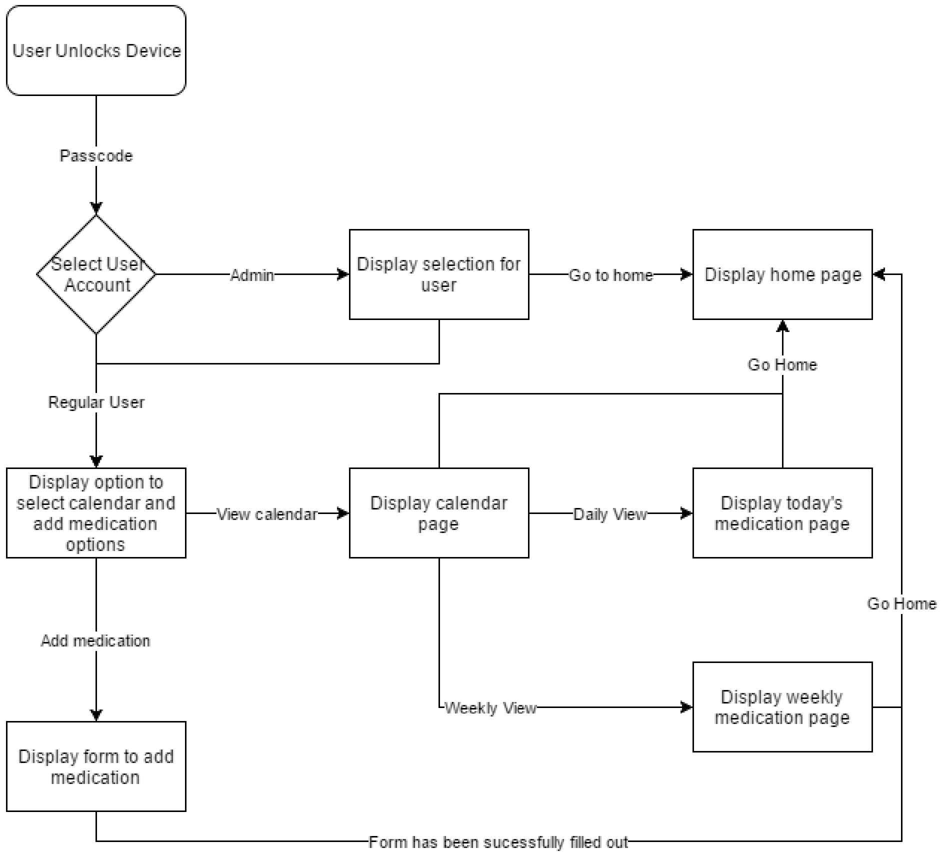

User Unlocks Device
Passcode
Select User Account
Admin
Display selection for user
Go to home
Display home page
Go Home
Regular User
Display option to select calendar and add medication options
View calendar
Display calendar page
Daily View
Display today's medication page
Go Home
Add medication
Weekly View
Display weekly medication page
Display form to add medication
Form has been sucessfully filled out

Figure [12]: *Flow chart of GUI when user is adding medication / viewing calendar*

4.4 Connection to Arduino Pro Micro

The Raspberry Pi will communicate with the Arduino Pro Micro via USB serial, using a custom protocol, in order to control the motors, for details see table 3 and table 4.

Table [9]: *Serial communication protocol from Raspberry Pi 2 to Arduino Pro Micro*

Bit Offset (from the right)	Functionality
0	Zero the hoppers
1,2,3	Chamber numbers
4,5,6	Number of pills to dispense from that chamber

Table [10]: *Serial communication protocol from Arduino Pro Micro to Raspberry Pi 2*

Bit Offset (from the right)	Functionality
0	Success/Failure
1,2,3	Chamber Number
4,5,6	Number of pills to dispense from that chamber

5. System Test Plan

5.0 Software Test Plan

5.0.1 Mobile Application Test Plans

T-0 Data Transfer	
Procedures: Attempt to access all elevated user features through the Android mobile application while connected through Bluetooth to the Pill-Matic	**Applicable Requirement:** S5.1.1-III, S5.1.2-II, S5.1.4-III, S5.1.5-III
	Expected Results: Controller shall respond correctly to any mobile application request received via Bluetooth and return the designated response signals

5.0.2 Mobile Application Privacy and Safety

T-1 Lock Screen	
Procedures: Attempt to modify pill schedules or change connection options	**Applicable Requirement:** S5.3.3-II, S5.4.2-III, S5.4.3-III
	Expected Results: Lock Screen prompt should activate and ask users for a password for further access

5.1 Hardware Test Plan

5.2.1 Power Modules

T-2 Power supply and adapter shall maintain voltage and current stability at all times	
Procedures: Maintain steady operation under maximum power constraints including: All motors at maximum chamber loads, LCD screen touch display ON, microcontroller running, Bluetooth connection	**Applicable Requirement:** H2.4.1-III
	Expected Results: All features are working as intended with no significant impact to user experience

5.2.2 Mechanical Modules

T-3 Dispenser Mechanism	
Procedures: Send a dispense signal either through “immediate dispense” option or as a normal scheduled dispense to the Pill-Matic	**Applicable Requirement:** G2.10.1-III, G2.10.3-III, S3.3.1-III
	Expected Results: Dispensing mechanism shall release the correct amount of pills as tasked from the correct chambers without being obstructed by the physical mechanisms

T-4 Holding Cell Mechanism	
Procedures: Allow break beam sensor to trigger or allow pills to be stored in holding cells for too long during its normal operational period	**Applicable Requirement:** H3.3.1-III, H3.4.7-II
	Expected Results: The holding cell will release any pills in its chamber to the separate discard chamber when prompted

5.1 Firmware Test Plan

5.3.1 Raspberry Pi Firmware:

T-5 The firmware recovers from system crashes	
Procedures: 1. Cause a power loss to the system. 2. Cause the execution of a program to segmentation fault 3. Cause an out of index error in a program	**Applicable Requirement:** F4.1.7-III
	Expected Results: The device reboots and comes back online

T-6 Firmware updates time when syncing with android application	
Procedures: Purposefully change android phone time so that the file transferred via Bluetooth has the incorrect time	**Applicable Requirement:** F4.1.4-III
	Expected Results: The Raspberry Pi adopts the time written in the file, instead of its' own system time.

T-7 Firmware loses less than 1 second per day when powered off and less than 100ms per day when powered on	
Procedures: Completely power off Raspberry Pi, record current device time, against internet time, power on in one day, compare time again.	**Applicable Requirement:** F4.1.4-III
	Expected Results: The current device time compared to internet time should differ by at most one second

T-8 The firmware can read from a template file shared from and to a Bluetooth device	
Procedures: Check to ensure that the firmware is able to read configuration information from the file transferred via Bluetooth and that it can write to said file	**Applicable Requirement:** F4.1.5-III, F4.1.6-III
	Expected Results: The firmware can read and write to the template file and adopts its' configuration to match it.

5.3.2 GUI:

T-9 User interface is reasonably fast so it does not interfere with everyday usage	
Procedures: Navigate between all GUI pages	**Applicable Requirement:** F4.5.1-I
	Expected Results: Time to get from page to page is on average less than 1 second

T-10 User interface does not allow user to get stuck in a menu	
Procedures: Navigate between all GUI pages and finally back to the home page	**Applicable Requirement:** F4.5.2-III
	Expected Results: All pages either link to another page or link to home

6. Conclusion

The design specification document clearly outlines the technical information and design requirements of each software, hardware or firmware component in the Pill-Matic. The test plan will serve as a guideline for implementing the functional requirements specified in previous documents. Health-Assist is currently developing the prototype and expects full functionality by April 1st, 2016.

7. References

[1] "Kivy: Cross-platform Python Framework for NUI", *Kivy.org*, 2016. [Online]. Available: https://kivy.org/#home. [Accessed: 09- Mar- 2016].

[2] "Develop cross Platform Mobile Apps and Games | Corona Labs!", *Coronalabs.com*, 20216. [Online.] Available: https://coronalabs.com/products/corona-sdk/. [Accessed: 09- Mar- 2016].

[3] "Download Android Studio and SDK Tools | Android Developers", *Developer.android.com*, 2016. [Online]. Available: http://developer.android.com/sdk/index.html. [Accessed: 09- Mar- 2016].

[4] "Xcode – What's New – Apple Developer", *Developer.apple.com, 2016. [Online]. Available: https://developer.apple.com/xcode/. [Accessed: 09- Mar- 2016].*