Major Project

Goal

This space will be used to organize the team for the major project. All information will be stored on here.

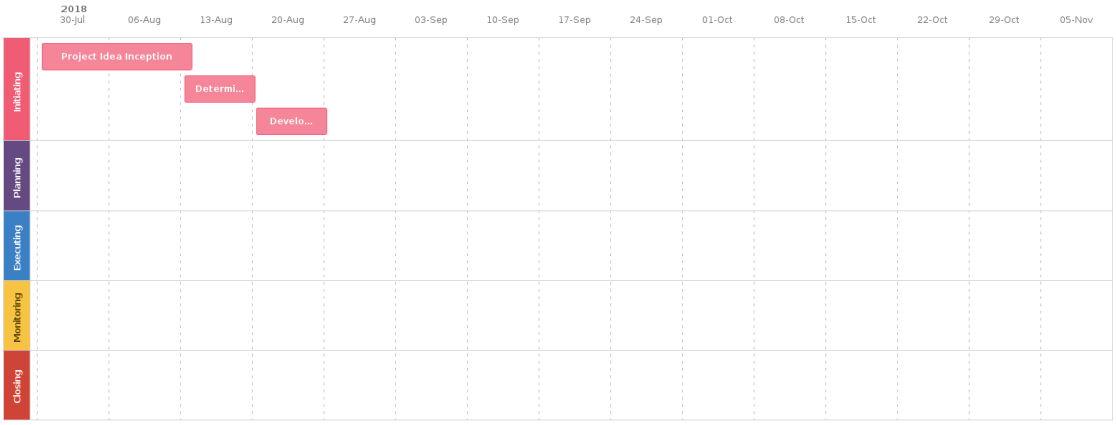
Pages

Core team

Daniel Moore	Lachlan Gabb	Fariha Uddin	Terry Ng
danielmoore.****@g****.com	****lan.gabb@gmail.com	****ha.uddin66@gmail.com	****.com.hk@gmail.com

Project Board

Roadmap



Quick navigation

When you create new pages in this space, they'll appear here automatically.

Recent Changes

Recently Updated

Results/Conclusions
Nov 23, 2018 • updated by fariha uddin • view change

image2018-10-29_20-21-14.png
Oct 29, 2018 • attached by Lachlan Gabb

image2018-10-28_18-20-38.png
Oct 28, 2018 • attached by Lachlan Gabb

Practical Assessment
Oct 22, 2018 • updated by Daniel Moore • view change

API_Architecture.png
Oct 19, 2018 • attached by Daniel Moore

API_Architecture.svg
Oct 19, 2018 • attached by Daniel Moore

Project Management Plan (Mel's "Practical Assessment")
Oct 16, 2018 • updated by Lachlan Gabb • view change

Physical Dispenser
Oct 14, 2018 • updated by Lachlan Gabb • view change

20181014_153047.jpg
Oct 14, 2018 • attached by Lachlan Gabb

image2018-10-14_15-29-27.png
Oct 14, 2018 • attached by Lachlan Gabb

image2018-10-14_15-28-5.png
Oct 14, 2018 • attached by Lachlan Gabb

Board Layout Page 2.png
Oct 14, 2018 • attached by Lachlan Gabb

Board Layout Page 1.png
Oct 14, 2018 • attached by Lachlan Gabb

Major Project Board Layout.png
Oct 14, 2018 • attached by Lachlan Gabb

Notification System
Oct 14, 2018 • created by Lachlan Gabb

All pages which should not be viewed by the public should be a child of this page.

Private Pages

Back and Front End Code Repositories

The backend application for the dispenser is written in Python using a backend framework called Django. The code for the backend API can be found at the following URL:

https://github.com/danielmoore-info/magic_meds

The web front end for the dispenser is written in JavaScript using Facebook's front-end development tool called ReactJs. The code for the web front end can be found at the following URL:

https://github.com/danielmoore-info/magic_meds_react

Feel free to make a pull request and contribute to the development of these project's if you feel you have something useful you can add.

Decision log

Create decision

Decision	Status	Stakeholders	Outcome	Due date	Owner
Which project idea to go with	COMPLETED	Lachlan Gabb, fariha uddin, TERRY NG, Daniel Moore	Pill Dispenser Project	14 Aug 2018	Daniel Moore

Which project idea to go with

Status	COMPLETED
Stakeholders	Lachlan Gabb, fariha uddin, TERRY NG, Daniel Moore
Outcome	Pill Dispenser Project
Due date	14 Aug 2018
Owner	Daniel Moore

BACKGROUND

For the major group project we need to come up with an idea were all happy with, including the glorious leader David Halfpenny. The idea needs to be decided before the above due date or we need to drop the subject. Please contribute your ideas to the project ideas list.

ACTION ITEMS

- [x] Come up with a list of project ideas
- [x] Pitch the ideas to David who will allow or deny an idea
- [ ] Create the project summary

Medication Despenser

Here is top page for all pages related to our medication dispenser

Responsibility Assignment

- Lachlan Gabb
    - Develop physical dispenser
    - Develop code to control physical dispenser
    - Develop RFID system and associated code
    - Link above systems to the backend database
- Daniel Moore
    - Develop Backend Database
    - Develop User Interface
- fariha uddin
- TERRY NG

All Pages

Physical Dispenser

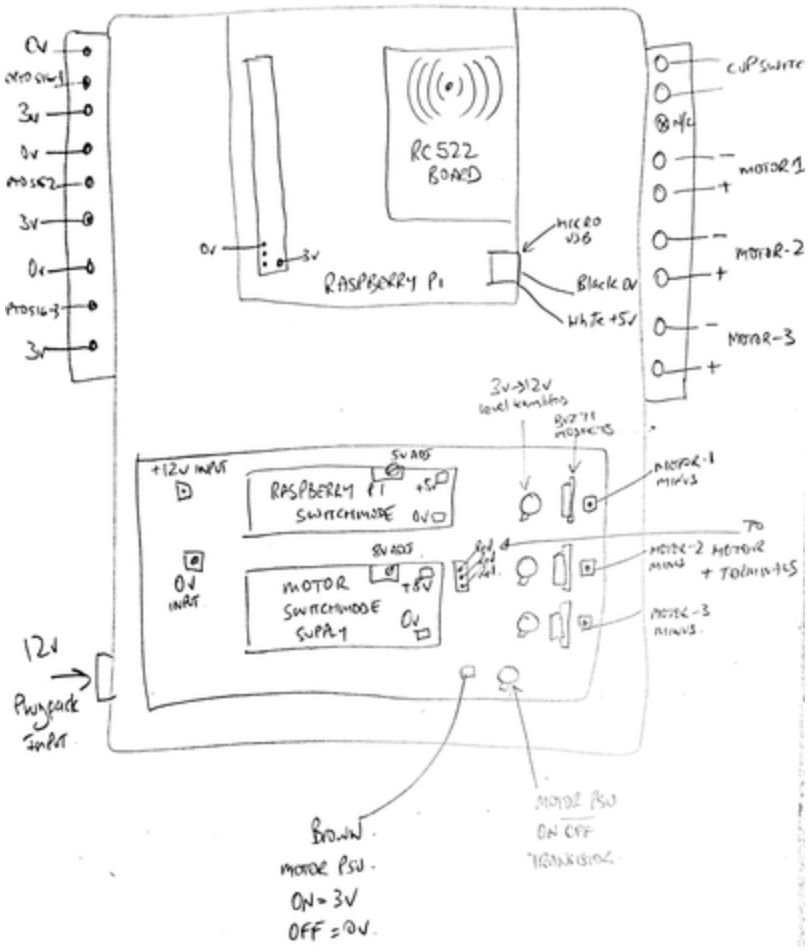
MECHANICAL OPERATION

CONTROL OF MOTOR

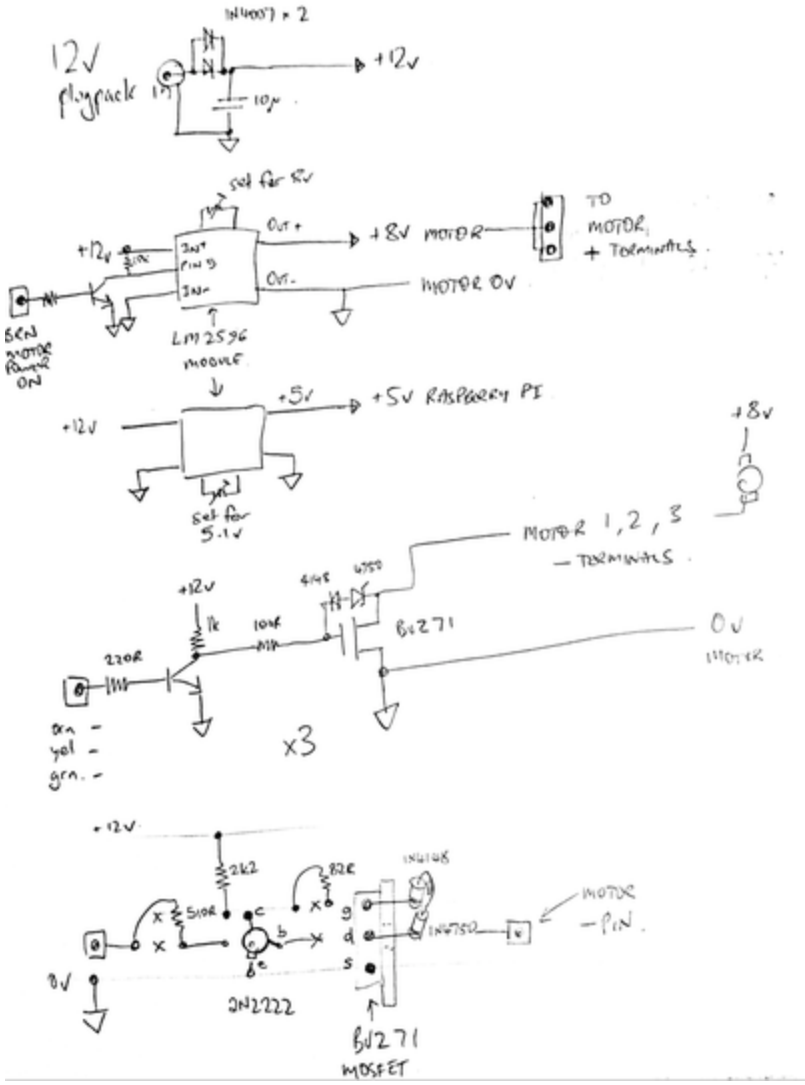
- The Pi will send a signal to a transistor
- The transistor will switch the 12 volt supply to the motor

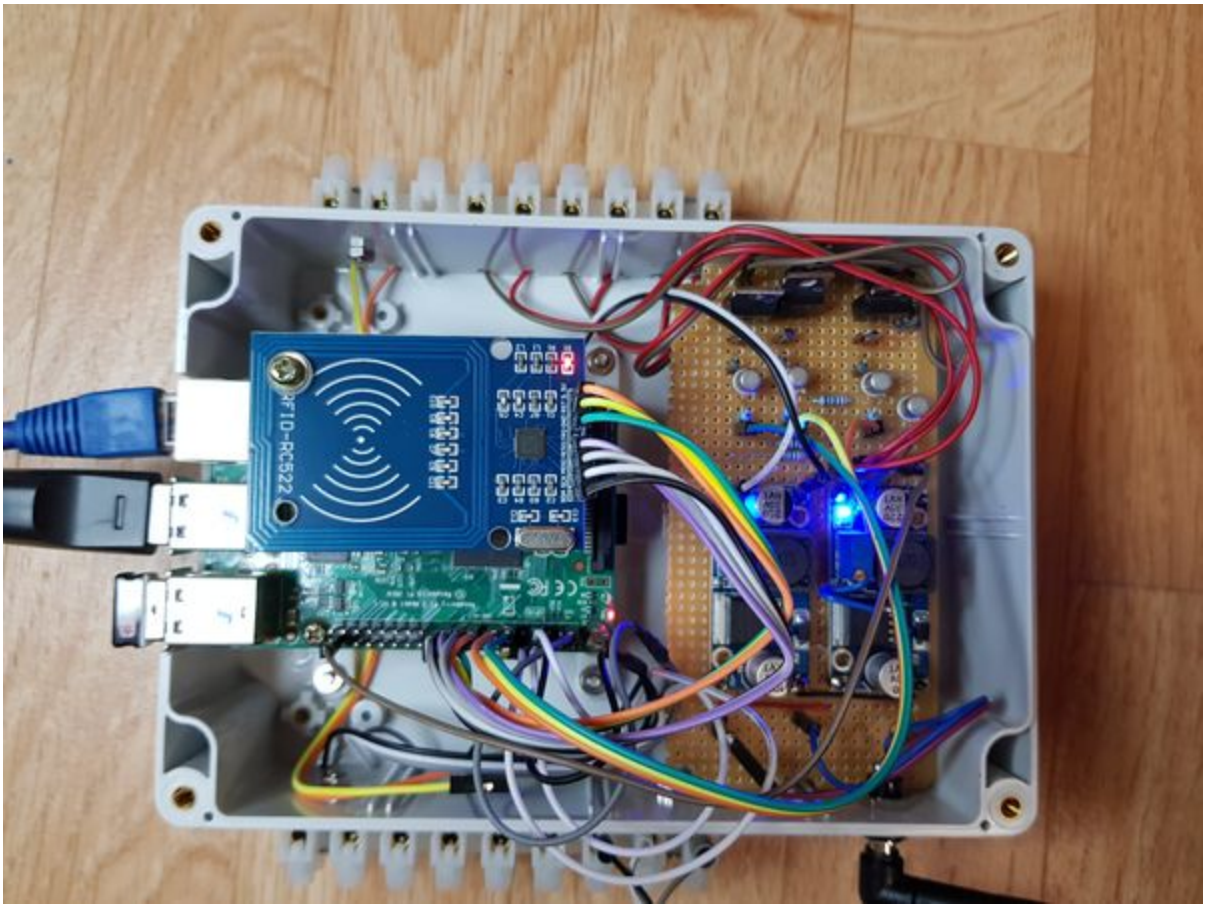
Electrical Configuration

- Tablet detection sensors are connected directly to the Raspberry Pi itself.
- The motor outputs are connected to transistor, which switch Mosfets which in turn switch the motors themselves on.
- The designed board takes in a 12 volt DC supply and through two power supplies creates a 5 volt feed for the Raspberry Pi and a 8 volt feed for the motors.
- The configuration is as shown:


0V
MOSI SW1
3V
0V
MOSI2
3V
0V
MOSI3
3V
RC522 BOARD
0V
3V
RASPBERRY Pi
MICRO USB
Black 0V
White +5V
CUP SWITCH
8 n/c
MOTOR 1
MOTOR-2
MOTOR-3
3V->12V level translators
BUZZER MOSFETS
+12V INPUT
5V ADJ
RASPBERRY PI SWITCHMODE
+5V
0V
MOTOR-1 MOSFET
TO
0V INPUT
5V ADJ
MOTOR SWITCHMODE SUPPLY
+5V
0V
MOTOR-2 MOSFET + TERMINALS
MOTOR-3 MOSFET
12V
Pwrjack input
BROWN
MOTOR PSU
ON=3V
OFF=0V
MOTOR PSU ON OFF SIGNALLING

•

- 
12V
flyback
+12V
10µ
set for 8V
Out+
+8V MOTOR
TO
MOTOR
+ Terminals
+12V
Out-
MOTOR 0V
LM2596
module
SEN
MOTOR
POWER
ON
+5V
+5V RASPBERRY PI
set for
5.1V
+8V
MOTOR 1, 2, 3
— TERMINALS
+12V
1k
10nF
BU271
0V
MOTOR
220R
grn -
yel -
grn. -
x3
+12V
2k2
82R
510R
c
b
e
2N2222
g
d
s
BU271
MOSFET
MOTOR
— PIN.
0V

- 

Project Proposal: Medication Reminders

BACKGROUND

People often forget to take their medications, especially those of older generations. In addition, people in older generations often forget the correct dosage of medications, meaning they take to much or to little of a particular medication. Ultimately, keeping track of how much medication is left can also become a tedious and time-consuming task. All these problems can become overwhelming and hard to manage for the elderly as they usually have a large number of medications to keep track of.

This project will endeavor to create a system that will manage the issuing, tracking and taking of medications. This will improve the quality of life of people who have trouble managing their medications.

OBJECTIVES

- Reduce the number of times a user forgets to take their medication
- Reduce the number of times a user takes the incorrect dose/type of medication
- Improve the quality of life of users by reducing the mental stress of taking the correct medication
- Reduce the number of times a user does not get another prescription

SCOPE

This project aims to create a smart medication dispenser, which must include the following features:

- Provide a reminder to take medication at set times
- Authenticate the user in order to dispense medications
    - RFID or,
    - Facial recognition
- Create a mechanism for dispensing medications
- Dispense the correct dosage of each medication
- Implement a sensor to detect if a cup is in place for the dispensing of medications
- Track the number of days left until each medication is empty
    - Prove reminders via a web interface
    - Emails

This project will be undertaken in the following phases:

1. Initiating

2. Planning
3. Execution
4. Monitoring
5. Closure

SCHEDULE

Phase Name	Description of Work	Start and End Date
Initiation	• Define the project • Determine the feasibility of the project • Develop the project proposal	30 Jul 2018 to 21 Aug 2018
Planning	• Define specific product requirements • Plan the work that must be performed to complete the project • Create the project budget • Gather resources to build the product • Create the project schedule • Plan for risks which may impact the project	30 Jul 2018 to 24 Aug 2018
Execution	• Begin construction • Modify design to accommodate unexpected obstacles • Creation of project code	24 Aug 2018 to 15 Oct 2018
Monitoring	• Evaluate performance (Weekly progress report) • Effort and cost tracking • Quality assurance of deliverables • Adjust tasks and schedules to ensure the project is on track	30 Jul 2018 to 30 Oct 2018
Closure	• Gain project acceptance • Document lesson learned	30 Oct 2018 to 01 Nov 2018

PROJECT BUDGET

Phase Name	Anticipated Costs
Initiation	$0
Planning	$0
Executing	$300
Monitoring	$20
Closure	$0

Budget Breakdown

Product	Cost ($)
Raspberry Pi	$53
LEDs	$10
RFID Reader	$40
RFID Chips	$10
Motor	$20
Reserves	$150

KEY STAKEHOLDERS

Client	Anyone who has trouble taking their medication

Sponsor	David Halfpenny
Project Manager	Daniel Moore
Project Team	Daniel Moore Lachlan Gabb Fariha Uddin Terry Ng

MONITORING AND EVALUATION

Progress will be evaluated by tracking the completion of project milestones as described in the above schedule and scope sections. The project will be completed once all milestones are completed.

APPROVAL SIGNATURES

Project Client	David Halfpenny, Project Sponsor	Daniel Moore, Project Team Member
Lachlan Gabb, Project Team Member	Fariha Uddin, Project Team Member	Terry Ng, Project Team Member

Literature Review

DOCUMENT CONTROL

Version	Authors
Draft	Daniel Moore Fariha Uddin Lachlan Gabb NG Wai Pan (Terry)
Final	Daniel Moore Fariha Uddin Lachlan Gabb NG Wai Pan (Terry)

TABLE OF CONTENTS

INTRODUCTION

People often forget to take their medications, especially those of older generations. In addition, people in older generations often forget the correct dosage of medications, meaning they take to much or to little of a particular medication. Ultimately, keeping track of how much medication is left can also become a tedious and time-consuming task. All these problems can become overwhelming and hard to manage for the elderly as they usually have a large number of medications to keep track of.

This project will endeavour to create a system that will manage the issuing, tracking and taking of medications. This will improve the quality of life of people who have trouble managing their medications.

EXISTING SOLUTIONS AND RESEARCH

While researching the topic of medication dispensers, we discovered there are a great number of simple mechanical dispensers available. However, as these devices were simply mechanical, they were unable to provide any kind of notification to patients to inform them to take their medication at pre-defined times, or to their carers when the medication was not taken on time. Additionally, all of these devices required pre-loading of the pills into compartments for specific time intervals. For example, a person would have to separate out all the necessary pills for Sunday morning and place them into the correct compartment on the device. The same procedure would also have to be completed for every other time interval tablets needed to be dispensed for.

There are also a number of electric medication dispensers available which solve some of the problems presented. One popular solution is the Phillips Lifeline appliance. This device provides automated delivery of medication at specific times and can even provide notification to the patient when they are due to take their medication. If the patient fails to take their medication within a specific time window, the device can also notify a designated carer. (Philips,2009)

However, the Phillips Lifeline dispenser still requires a person to separate out all of the necessary medication and place it into containers. This severely limits the amount of time the device can function autonomously for (Ren,2015). The device has a total of 42 containers, assuming the average user requires three sets of tablets a day, this would only allow the device to operate unattended for 14 days. (Philips,2009)

Figure: Philips Lifeline Dispenser

Another electronic device, known as the Livi Automatic Pill Dispenser solves this problem. The device is capable of taking up to 15 containers of different pill types and then dispensing them in the correct combination automatically. (A Medium Corporation, 2017) As a result of this ability, the device is capable of storing enough pills to operate automatically for 90 days. Livi can accommodate a variety of pills shapes and sizes but cannot dispense inconsistently shaped or gummy-textured pills. It is battery operated and the battery is charged when plugged into electricity. Livi communicates with the LiviWeb server over a secure and encrypted channel. (Liviathome,2018) However, the device does come at a considerable cost. Starting at $2000 USD, it would be too expensive for the average person. (Livi, 2018)

Figure: Livi Automated Medication Dispenser

By researching the various solutions available solutions, we were able to more clearly define the objectives of this project. The solution we propose to design will have the ability to dispense a combination of different pills at the necessary time periods, without the need for a carer to pre-sort the medication. Additionally, the solution should be able to remind patients to take their medication as the designated time periods and also provide notification to their carers when a medication interval is missed. Finally, to make the solution more obtainable to average people we will aim to have the whole project completed for $300.

CONTROLLING DEVICE

For the project, we required a computing platform that would be capable of the following tasks:

- Small enough to fit into our tablet dispenser
- Energy efficient as it would most likely be running continuously
- Powerful enough to process the RFID authentication and monitor the distribution of tablets
- Be able to connect to a standard 802.11 LAN and host a website allowing tablet information to be viewed and updated.
- Be able to interface with a variety of different sensors and control a number of motors that would be used within the system.
- Be affordable within our limited project budget.

Using these requirements as a baseline, the following options were considered:

Arduino

One option is the Arduino, which is a low power microprocessor used for many small projects such as this one. The Arduino is capable of interpreting data from a number of inputs. The Arduino's connectors also support controlling a number of motors and other actuators which would be useful for this project. In addition, all inputs and outputs on the Arduino support both digital and analogue inputs and outputs, allowing a vast number of sensors and motors to be connected to it. The Arduino would also be the most cost-effective option evaluated, only costing approximately $25.

A pill dispenser project has been developed using Arduino that integrates with SparkFun ESP8266 thing board and PIR motion sensor. Arduino code was written and a Blynk app was designed to insert the number of dosages and send notifications to the user (Ren 2015, p.18-20)

However, the boards are severely limited in processing power and most do not come with any networking capability out of the box. In addition, this means they cannot run a well known operating system such as Linux, reducing the number of tasks we can get it to perform. These two factors make the Arduino unsuitable to run many of the other components of our project, such as the database and web interface. Therefore, the Arduino could perform some of the requirements for the project but would ultimately require the support of another computing platform.

Small PC

Another option for the project would be to build a small desktop quality PC into the tablet dispenser. This would allow us to run a full-featured desktop operating system making it easy to add functionality and develop scripts to run upon it. Additionally, a PC would provide more than enough processing power to perform all of the tasks we require of it.

However, the overall cost of such as device would be prohibitive to start with. A low power PC with the minimum requirements is still going to cost around $400. Additionally, the PC would require considerable space, increasing the overall size of our tablet dispenser. A standard PC also doesn't come out of the box with any inputs or outputs capable of connecting to sensors or controlling motors, therefore requiring us to purchase additional hardware to facilitate the required connectivity. Finally, a full power PC would consider considerably more energy than any of the other solutions presented, making it less than ideal for an application where it would likely be running constantly.

Raspberry Pi

The Raspberry Pi is a reasonably powerful single board computer which can be obtained for around $35. The board runs a Broadcom manufactured ARM processor with four cores running at 1.4GHz, providing more than enough processing power required for the project. The board also supports a wide range of well known operating systems, making it easy to add functionality. The Raspberry Pi also supports reading data from sensors and controlling motors using Python scripts, making it extremely easy to control. The general purpose input and output connectors on the raspberry pi support a number of digital communication protocols, allowing it to interface with a wide range of sensors and motors. Finally, the Raspberry Pi also supports Ethernet networking and WiFi out of the box, making it simple to integrate our tablet dispenser with an existing network.

However, all input and outputs available on the Raspberry Pi only support digital communications, which does limit out sensor and motor choice slightly. Additionally, with only 40 input and output connection, which many reserved for specific purposes, it does limit the overall number of sensors and devices that can be attached to a single raspberry pi.

ELECTRIC MOTOR

An electric motor of some description is required to rotate the wheel of the tablet dispenser. It is the actual device that energizes and de-energizes the circuit of the motor so that it can start or stop (Mukund and Srinath,2012) The electric motor required would need to have the following attributes to be considered suitable:

- Have enough torque to rotate the dispensing wheel without difficulty.
- Capable of being controlled by the Raspberry Pi
- Be small enough to fit into the dispensing unit
- Low voltage to ensure the safety of the device
- Ability to rotate the dispensing mechanism at approximately 1 revolution per second

Hobby Motor YM2706

The Hobby motor is easily obtainable for a low price of only $3.50. The motor is designed to run off three volts, ensuring the safety of the device and also allowing it to be easily controlled with a Raspberry Pi. However, the motor has a very low torque output of only 0.000291257505 Newton Meters. The low torque produced by the motor may not be sufficient to turn the dispensing mechanism, making the motor unsuitable. Additionally, the motor rotates at a very high speed of 8450 RPM which would be considered too fast for the dispensing mechanism to function correctly as it would dispense over 140 tablets per second. For these reasons, this motor was eliminated from the selection.

(Jaycar.com.au, 2018)

Jaycar YG2734

The Jaycar YG2734 is available from Jaycar, making it easy to obtain. The motor is designed to be driven off 12 volts which is still low enough to ensure safety and also be controllable by our Raspberry Pi and some accompanying circuitry. This stepper motor can divide a full rotation into a large number of steps. Its position can be controlled precisely as long as it is carefully sized to the application (Mohammed,2011). At the designed voltage of 12 volts, the motor also rotates at 36RPM, therefore completing just under a single revolution of the dispensing mechanism in a second. Additionally, the 36 RPM is achieved by running the motor at a higher speed and then using a gearbox connected to the motor to significantly reduce the output speed. A side effect of this gearing is an increase in the torque produced by the motor. With the gearbox, the motor produces just over 1 Newton Meter of torque, more than enough to rotate our dispensing mechanism. As this motor meets all of the criteria, it is the one we shall be using for the project.

(Jaycar.com.au, 2018)

PATIENT IDENTIFICATION

A requirement of this project is that users must authenticate/provide some form of identification in order to dispense their medication. The authentication/identification mechanism should have the following features:

- Patients are not required to memorise credentials (remembering credentials may be difficult for patients with certain disabilities)
- The authentication mechanism should be easy to use

Based on the above requirements, two technologies have been identified which may be suitable for this project: Facial recognition and RFID paired with a Universally Unique Identifier. These technologies are briefly described below.

Facial Recognition

Facial recognition is the process of authenticating or identifying a person based on patterns which exist in their facial contours or through three-dimensional modeling of a person's face (Techopedia.com, n.d.). Facial recognition would allow the patients to be authenticated seamlessly to the dispenser, without the need for much interaction. Facial recognition is becoming increasingly simple to implement through programming libraries such as OpenCV (Docs.opencv.org, 2018) or through service providers such as Amazon's Rekognition (Amazon Web Services, Inc., 2018) service (Fox-Brewster, 2018). A downfall of implementing facial recognition is that if the dispenser were to be deployed in a hospital or nursing home, photos of patients would be required to implement the facial recognition. This method would be hard to scale, as multiple photos of each patient would be required, and storing patients photos would become a security risk.

RFID Tags and Universally Unique Identifiers

RFID tags are small electronic chips which data can be read from or written to using radio waves (Bonsor and Fenlon, 2007). When a patient is enrolled into the system, a Universally Unique Identifier (UUID) would be generated and written to the RFID chip. This would allow the patient to simply place their RFID bracelet near the dispenser in order to authenticate. This is relatively easy to use and does not require a large amount of information to be stored about the patient, reducing security risks related to storing health information. This method scales well as RFID tags can be erased and reused. Furthermore, the cost of RFID tags is low, for example, the UHF RFID Classic Tag from Core Electronics costs $1.74 (Core Electronics, 2018).

MECHANICAL DISPENSER DESIGN

The mechanical dispenser posed a real challenge in the development of the project as it was both difficult and crucial to the success of the project. The dispenser mechanism was determined to have the following requirements:

- Ability to dispense a single tablet at a time from a container full of tablets
- Ability to successfully dispense tablets repetitively without jamming

Sliding Dispenser

Upon researching the topic of tablet dispensers, one of the main prototyping ideas found was to use a sliding dispenser mechanism. This method relied on a chamber into which a single table could fall. The whole chamber then moved out of the container of tablets and then across an opening which allowed the single tablet to then fall out. However, it was found that this kind of method was vulnerable to jamming if tablets turned sideways and more than one attempted to enter the chamber (Algoryx Momentum, 2018). As one of the requirements is to reliably dispense a single tablet, this design was deemed not to be suitable.

Rotating Dispenser

The rotating dispenser was inspired by the mechanism used in Gumball dispensing units. The mechanism consists of a rotating disk with one or more holes in it which allows a tablet to fall into it. As the disk rotates through the tablet container, a tablet will become caught in the hole and begin to move with the disk. At some point in the 360 degrees rotation of the disk, the tablet would pass over a hole in the base of the unit, allowing it to fall out (Lee,Groeteke and Hewaparakrama,2016) Unfortunately, research uncovered very little useful information regarding other people's experience with such a design.

To gain a better understanding of how well the design would work, we created a small prototype using an old CD with a hole drilled through it. Upon testing the concept, it was found to reliably dispense our fake tablets with every rotation. As we can verify the design is likely to work, it will be the design we use for this project.

SOFTWARE ARCHITECTURE

When designing a software product, the application will have requirements which align with a certain software architecture. Software Architecture defines, at a high level, the components of a software system and how they interact with each other (Bass, Clements and Kazman, 2013). There are a number of architectures which are well documented including (Mallawaarachchi, 2017) :

- Client-server
- Microservices
- Master-slave

As part of the project, a variety of software will be produced which must integrate into one robust application. The application has the following requirements, these requirements will assist in selecting the correct architecture of the application:

- Patient and medication information needs to be stored, with the ability to create, read, update and delete (CRUD) information.
- Access to stored data needs to be controlled
- Data will need to be accessed and updated by a variety of client applications including:
    - Scripts which control the sensors on the dispenser
    - Web clients to display information

Based on the above requirements, it was identified that the project software would align with a three-tiered client-server application architecture. This architecture is based on three layers, each layer and its relation to the requirements of the project are described below:

Data Layer

The data layer is responsible for the storage of information. The data layer also provides CRUD functionality to the application layer. An example of the data layer is SQL which provides tables and values for storage, as well as SQL queries for CRUD functionality. This satisfies the requirement around the storage of Medication and Patient information as well as CRUD functionality.

Application Layer

The application layer is the middle-man between the client and the data. This is so logic can be placed between the client and the data. The application layer will interface with the data layer, allowing information in the data layer to be modified. The application layer will define the specific set of operations which can be performed on the data layer, preventing direct access from the client to the database. This allows businesses to

define requirements for accessing data such as authentication and permissions, and other logic (Stackify, 2017). This satisfies the requirement that access to the data layer must be controlled in some way.

Presentation Layer

The presentation layer is the set of applications with which the users interact with. The presentation layer interfaces with the API's defined at the application layer so the user is able to perform CRUD operations on the information stored in the data later (Stackify, 2017). Multiple application can exist at the application layer which satisfies the requirement that the data will need to be accessed by a variety of client applications.

LIGHT EMITTING DIODE FOR NOTIFICATION :

A light emitting diode (LED) will be used as part of the notification system for the project. It provides information such as power on or emergency indication. Pre-selected precautions to the patient concerning the medications being correctly dispensed will be provided by the LED (Broadbent,Tanagawa,Kerse,knock,Patience & McDonald,2009). Different coloured LEDs will be used for certain purposes. For example, a red LED will notify the user that it is the time to take a medication while a green one will indicate that the pills within the container are running low and need to be refilled. An LED with the following capabilities is required for the project :

- Can emit different coloured lights
- Can be controlled individually from a single GPIO
- Can be connected to a Raspberry Pi

Single coloured LEDs :

One of the options is to use LEDs that only show a single colour. However, these LEDs do not have digital control which makes them more suitable for smaller projects like adding LEDs to wearables (Adafruit Learning System,2018). Also, they cannot be controlled individually from a single GPIO (SunshowerOnline)

Tri coloured LEDs :

The other option is to use a roll of Red, Green and Blue LEDs.This type of LEDs generally come on a white coloured strip. Each LED contains a built-in controller chip and can be controlled individually from a single GPIO (Core Electronics,2018). For our project, we have chosen 5M WS2812 RGB weatherproof LED strip from Core electronics. This roll has 30 LEDs per meter on a white coloured strip and each can be controlled individually.

FORCE SENSITIVE RESISTOR

This is the sensor to detect physical pressure, squeezing and weight. This sensor is small, simple and low cost at around $10 such as the interlink model 402 FSR with ½ diameter sensing region.
This sensor is a resistor that will vary its resistive value (in ohms) based on how much pressure is applied to it. By measuring the change in ohms, we will be able to approximate the weight of the cup sitting on top of the sensor.

This sensor requires an MCP3008 analog to digital converter to convert the analog signals from the FSR to digital signals (Caird, 2018) so that the Raspberry Pi can read and process the data. This sensor will be used with a breadboard to connect all components and to deliver power to the sensor.

WIRING

Jumper wires will be used to connect all the various sensors and motors to the Raspberry Pi and to power. Jumper wires come with connectors on either end allowing for easy connection to both our Raspberry Pi and the other components. The wires are available at very low cost, with a pack of 10 costing only around $4. Additionally, these wires are easy to obtain, both online and in the shops.

METHODOLOGY

After the literature research was undertaken to identify studies relating to similar projects, the following components and methodologies have been chosen to make the medication dispenser :

- Raspberry Pi as a controller device
- Jaycar YG2734 to control the motor
- RFID tag for patient identification
- WS2812 RGB LED strip
- Force sensitive resistor to determine the weight
- Jumper wires for connecting the sensors and motors
- Rotating mechanism to dispense the pills
- Three tiered client-server application architecture

CONCLUSION

This document has briefly outlined the objective of our project and discussed the currently available solutions and hardware. While an existing solution to the problem does exist, its high cost means it is not accessible to the majority of people who require its help. Additionally, we have

outlined a variety of devices and sensors that are required to achieve our objective and chosen one of each component we believe to be the most suitable at this time.

REFERENCES

Adafruit Learning System (2018) . Adafruit LED sequins . [online] Available at : https://cdn-learn.adafruit.com/downloads/pdf/adafruit-led-sequins.pdf

Amazon Web Services, Inc. (2018). Amazon Rekognition – Video and Image - AWS. [online] Available at: https://aws.amazon.com/rekognition/ [Accessed 30 Aug. 2018].

A Medium Corporation (2017), 2017: the year of the IoT, automated pill dispenser [online] Available at: https://medium.com/@medipense/2017-the-year-of-the-iot-automated-pill-dispenser-ca1d41f0592b [Accessed 31 Aug. 2018]

Bass, L., Clements, P. and Kazman, R. (2013). Software architecture in practice. 2nd ed. Upper Saddle River, NJ: Addison-Wesley.

Bonsor, K. and Fenlon, W. (2007). How RFID Works. [online] HowStuffWorks. Available at: https://electronics.howstuffworks.com/gadgets/high-tech-gadgets/rfid1.htm [Accessed 30 Aug. 2018].

Broadbent E ,Tamagawa R ,Kerse B ,Knock B, Patience A and Mcdonald B (2009) . Retirement home staff and residents: preferences for healthcare robots, 18th IEEE International Symposium on Robot and Human Interactive Communication

Caird, A. (2018). Using a Force Sensitive Resistor with a Raspberry Pi. [online] Acaird.github.io. Available at: https://acaird.github.io/computers/2015/01/07/raspberry-pi-fsr [Accessed 2 Sep. 2018].

Core Electronics. (2018). UHF RFID Classic Tag. [online] Available at: https://core-electronics.com.au/uhf-rfid-classic-tag.html [Accessed 30 Aug. 2018].

Core electronics (2018). 5M RGB LED Strip - WS2812 30 Per Meter - White Strip - Weatherproof . [online] Available at : https://core-electronics.com.au/5m-rgb-led-strip-ws2812-30-per-meter-white-strip-weatherproof.html [Accessed 2 September,2018]

Docs.opencv.org. (2018). FaceRecognizer — OpenCV 2.4.13.7 documentation. [online] Available at: https://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_api.html [Accessed 30 Aug. 2018].

Fox-Brewster, T. (2018). We Built A Powerful Amazon Facial Recognition Tool For Under $10. [online] Forbes.com. Available at: https://www.forbes.com/sites/thomasbrewster/2018/06/06/amazon-facial-recognition-cost-just-10-and-was-worryingly-good/ [Accessed 30 Aug. 2018].

Jaycar.com.au. (2018). Hobby Motor - Low Torque | Jaycar Electronics. [online] Available at: https://www.jaycar.com.au/hobby-motor-low-torque/p/YM2706 [Accessed 29 Aug. 2018].

Jaycar.com.au. (2018). 36RPM 12VDC Reversible Gearhead Motor | Jaycar Electronics. [online] Available at: https://www.jaycar.com.au/36rpm-12vdc-reversible-gearhead-motor/p/YG2734 [Accessed 29 Aug. 2018].

Lee C, Groeteke E , Hewaparakrama J (2016), Automated pill dispenser: Final Report for ECE 445, Senior Design, Spring 2016

Livi (2018), Livi Automatic Pill Dispenser. [online] Available at: https://www.amazon.com/Livi-Automatic-Pill-Dispenser/dp/B077G4W772 [Accessed 31 Aug. 2018]

Liviathome (2018) Livi Hands on support: Frequently asked questions. [online] Available at: https://liviathome.com/help [Accessed 31 Aug. 2018].

Mallawaarachchi, V. (2017). 10 Common Software Architectural Patterns in a nutshell. [online] Towards Data Science. Available at: https://towardsdatascience.com/10-common-software-architectural-patterns-in-a-nutshell-a0b47a1e9013 [Accessed 30 Aug. 2018].

Mohammed A.a (2011). Automated medical dispenser system . [online] Available at: http://khartoumspace.uofk.edu/bitstream/handle/123456789/18590/Automated%20medical%20dispenser%20system.pdf?sequence=1&isAllowed=y [Accessed 29 Aug. 2018]

Mukund S and Srinath N.K (2012). Design of Automatic Medication Dispenser. [online] Available at: https://airccj.org/CSCP/vol2/csit2324.pdf [Accessed 29 Aug. 2018]

Philips Lifeline. (2009) . The Philips Medication Dispensing Service. [online] Available at : http://affordablehomecarenc.com/wp-content/uploads/philips-medication-dspenser-information.pdf [Accessed 31 Aug. 2018]

Ren Y (2015) . MAE 540-Advanced Product Design Methodology: Smart Pill Dispenser

Techopedia.com. (n.d.). What is Facial Recognition? - Definition from Techopedia. [online] Available at: https://www.techopedia.com/definition/32071/facial-recognition [Accessed 30 Aug. 2018].

YouTube. (2018). How to invent a pill dispenser - Algoryx Momentum. [online] Available at: https://www.youtube.com/watch?v=iwnlcyby1cw [Accessed 30 Aug. 2018].

SunshowerOnline . Buying guide : LED Strip Lighting. [online] Available at : https://sunshoweronline.com.au/image/assets/Guides/BuyingGuideLEDStripLighting.pdf [Accessed 2 September,2018].

Stackify. (2017). What is N-Tier Architecture? How It Works, Examples, Tutorials, and More. [online] Available at: https://stackify.com/n-tier-architecture/ [Accessed 30 Aug. 2018].

Part List

Part	URL	Cost
Weight Sensor	https://core-electronics.com.au/force-sensing-resistor-0-2-diameter-circle.html https://acaird.github.io/computers/2015/01/07/raspberry-pi-fsr	$8.32
RFID Sensor and Cards		
Motor Controlling Transistor	https://pimylifeup.com/raspberry-pi-rfid-rc522/ https://www.instructables.com/id/RFID-RC522-Raspberry-Pi/ https://www.raspberrypi-spy.co.uk/2018/02/rc522-rfid-tag-read-raspberry-pi/	
Raspberry Pi	https://core-electronics.com.au/raspberry-pi.html	$55

Project Plan

Table of Contents

Introduction

PROJECT BACKGROUND

People often forget to take their medications, especially those of older generations. In addition to ignoring their medications, people often become confused as a result of complicated medication regimes and choose the incorrect dosage. Ultimately, keeping track of the drugs they are required to consume, and the schedules are becoming increasingly complicated and time-consuming. As a result of this, staying on top of the required medication is often something people struggle with, and the consequences of getting it wrong can be severe. This project will create a system that will manage the issuing, tracking and taking of medications, aiming to increase the quality of life for its users.

PROJECT OBJECTIVE

The project aims to create a device which will be capable of automatically tracking and automating the delivering of medication to a patient.

- To achieve this goal, we will create a prototype system that will:

- Check that a cup is under the dispenser.
- Scan a patients RFID tag to identify the patient.
- Check if medications are pending.
- If medicines are pending, dispense the correct medication into their cup.
- Ensure an empty container returns to the machine.

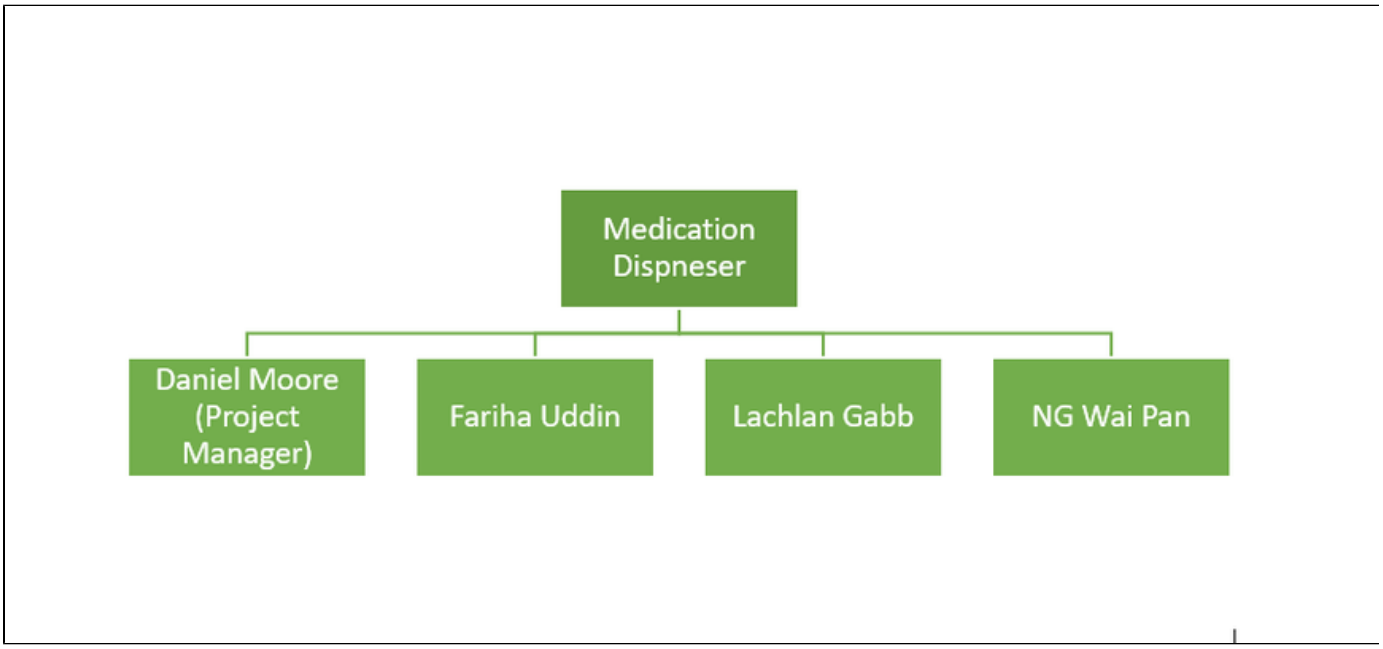
The above steps only work when the patient remembers to take their medication. The device will have features that assist the user when they forget medicines including:

- Indicate that tablets are overdue by displaying a flashing red light (Warning Level 1).
- Providing an audible beeping sound if tablets are still not taken (Warn Level 2).
- Sending a notification to a designated carer (Final Warning Level).

The dispenser will achieve this functionality by using a Raspberry Pi to monitor sensors and use a database to determine what tablets a particular user requires and at what time. While this early prototype will only support a limited number of pills, the mechanisms are designed to be scalable.

The prototype dispenser will be released within three months with a budget of ~$350. The progress of the project will be tracked using a blog and at the conclusion of the project both a final report and presentation will be provided outlining the project.

Organisation



Dr. David Halfpenny and Mehrdad Razmjoo will oversee the project. The following group members will perform the tasks to create the product:

- Lachlan Gabb
  - Design and construct the physical dispensing mechanism.
  - Integrate the RFID reading system.
  - Design and build the physical dispenser.
- Daniel Moore
  - Project Manager.
  - Design and implement the database and backend software.
  - Design and implement a user interface.
  - Perform a vulnerability assessment and penetration test on the System.
- fariha uddin
  - Documentation Manager.
  - Design and implement the visual and auditory warning systems.
  - Design and implement the notification scheme (Microsoft Flow or Pushover)
- TERRY NG
  - Design and Implement the weight management.
  - Design and Implement the Heartbeat System.

All four members will perform testing of the product with stand-in medication as well as writing the final report and the final presentation. All four members will also be responsible for ensuring any code they write interacts correctly with the backend database, as it will provide the single source of truth for the system as a whole.

Risk

As with any project, many uncertainties exist which could affect the outcome of the project. This section aims to identify perils which could occur and defines strategies to mitigate such risks.

RISKS IDENTIFIED

Incorrect Medication Dosage

The most significant risk of this project is dispensing the wrong dosage of medications. Incorrect dosages will result in the patient's medical condition deteriorating. If this occurs, the creators of the product may be legally liable for any damage caused to the health of a patient. Controls will be put in place to mitigate this risk, most notably a mechanism to verify the correct number of pills. This mechanism will measure the weight of the dispensed medications comparing the measured value to the expected value. Where the measured value is not reasonably close to the predicted value, a warning will be given to check the medications.

Failure of Device

Another significant risk arises should the device fail to function Without the device functioning correctly, the patient will likely be unable to access their necessary medication. If this occurs, the patient's medical condition will worsen.

To ensure the machine is working correctly, any errors generated by machine components report to a central database. Should any of these errors indicate the device is likely to fail, a notification will be sent to the designated carer. Additionally, the system will provide heartbeat messages to a central server. Should the heartbeat messages not be received, a notification will also be sent to the carer to advise them to check the status of the machine.

Internet Outage

The tablet dispenser will make use of an Internet connection to send alerts. However, if something were to happen to the Internet connection, the device would be unable to transmit signals. Without the ability to send alerts, it would not be able to notify a carer when a patient was not taking their medication, ultimately placing their health at risk. The Internet outage is not a high risk, as the device will continue to function properly without an internet connection.

Furthermore, the product will make use of the heartbeat messages to reduce this risk further. If heartbeat messages vanish, the external monitoring system will alert the designed carer allowing them to investigate the situation and take any steps deemed necessary. For example, they may wish to call the patient to verify the medication has been consumed just while the Internet connection is unavailable. Additionally, when connectivity to the device established, the carer will receive another notification.

Security Vulnerabilities

As the dispenser must be connected to a network for administration and to the Internet for the notification system, there is the chance that a cyber adversary or malware may attempt to take over the system. The consequences of such an attack may range from extracting personal information to rendering the device inoperable or even modifying the configuration to dispense the incorrect medication. Any of these outcomes can come with severe ramifications for the health of the user. All unnecessary ports and services have been stopped to reduce the attack surface of the dispenser. Additionally, as a member of the project team is a professional penetration tester, a penetration test will also be performed against the device to ensure it is not vulnerable to attack. It would also be recommended to ensure the device is placed behind a firewall, or an IPS device to provide an extra layer of protection.

Miscommunication

Miscommunication is a common problem when working as a team. Miscommunication can lead to many issues ranging from subtle differences in the end product to significant differences which place the whole project at risk of failure. To overcome miscommunication risks, we have taken several steps. Firstly, group chat has been created to allow all members of the group to communicate efficiently. Group chats ensure everyone is aware of the communication between all parties at all times. Secondly, all documentation will be on the collaboration platform Confluence. Confluence allows everyone to work on the same document and editing a report sends an email alert to all users. Alerts allow all group members to be aware of changes and voice any concerns. Finally, the group has a weekly meeting where all members are present, allowing for a discussion of the projects progress as well as understanding where each person is up too. With these three methods, we have significantly reduced the chances of miscommunication surrounding our project.

QUALITATIVE RISK ANALYSIS

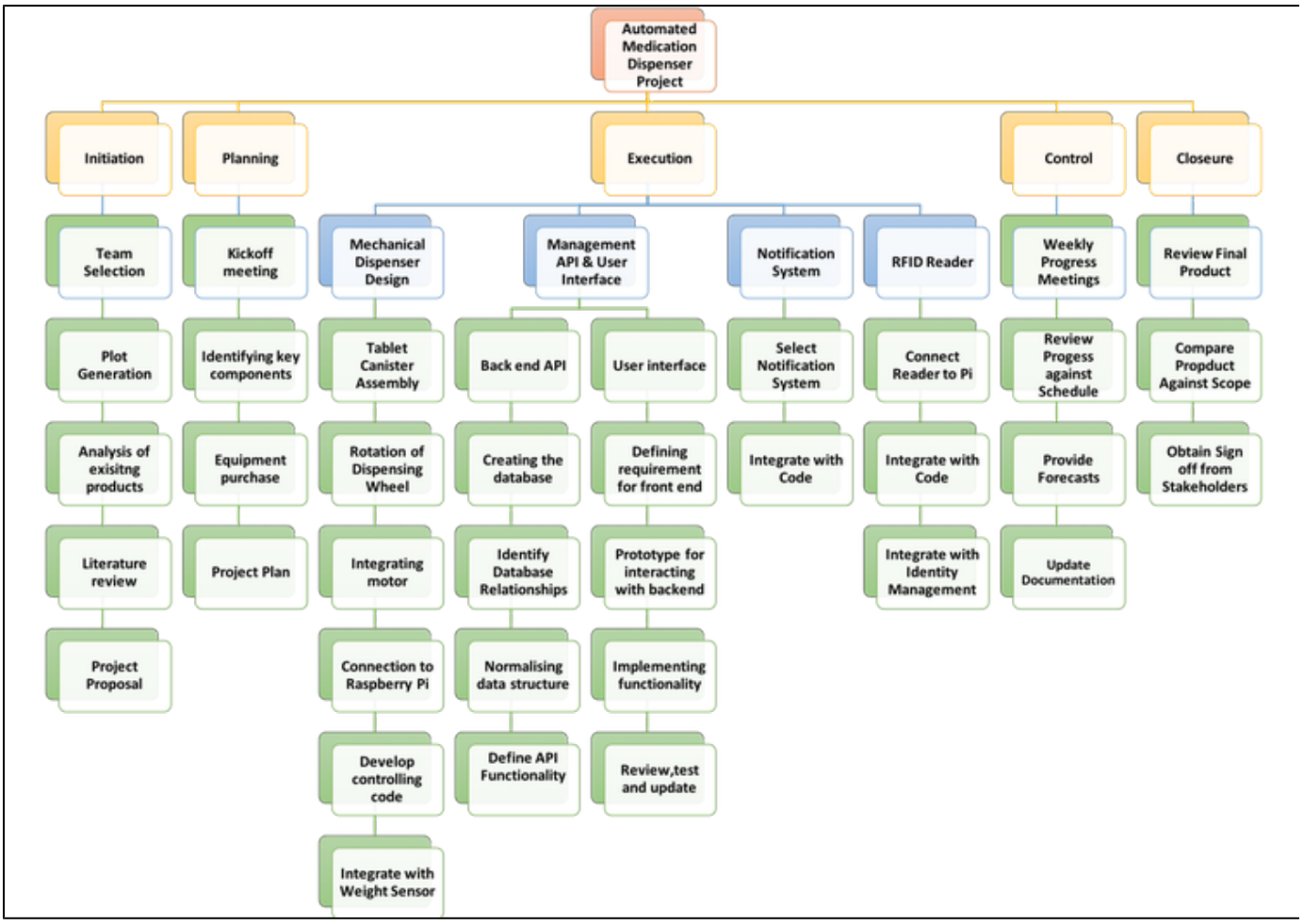
The risks identified will be analysed against the risk management standard AS/NZ ISO 31000: 2009 (See Appendix A for the risk assessment criteria). The following table lists the identified risks and their associated risk levels.

Risk	Impact	Likelihood	Risk Rating
Incorrect Medication Dosage	Extreme	Possible	High
Failure of Device	Extreme	Possible	High
Internet Outage	Moderate	Possible	Medium

Security Vulnerabilities	Extreme	Unlikely	High
Miscommunication	Moderate	Rare	Low

Work To Perform

WORK BREAKDOWN STRUCTURE



Phase 1 Initiation

In the initiation phase, a team of four was formulated. Potential project ideas were discussed resulting in the idea of making an automated medication dispenser. Following this, research was performed on existing medication dispensers and found their price is upwards of a thousand dollars. Additionally, the machines still require assistance from other people to operate. A project proposal was developed and the project initiated with the approval of the course coordinator.

Phase 2 Planning

A thorough project plan is often the difference between project success and failure. The planning phase begins with a kick-off meeting which will help ensure all project team members share a general understanding of the project. Additionally, in this meeting, communication and collaboration channel are established.

Next, the team will analyse the product requirements to identify the critical components of the device. This identification generates thought into the budget and schedule of the project and prompts the creation of these documents. Once the budget and timing of the project are clarified, equipment is purchased to align with critical components and the budget.

Lastly, the results of the above processes have been recorded and written in this document.

Phase 3 Execution

Phase 3.1 Mechanical Dispenser

One of the first obstacles to overcome was to develop a physical dispenser capable of dispensing a single tablet at a time. Research on designs such as sliding and rotating wheel mechanisms was performed. The sliding devices were found to jam easily. Therefore it was decided that our project would use a rotating wheel dispenser.

Once the design was confirmed, we develop a small prototype dispenser to verify the design worked. The prototype consisted of a CD with a hole cut in it to allow a tablet sized object to fall through at a particular section of its rotation. The prototype operated successfully, dispensing a single pill for each 360-degree rotation of the CD.

Once the concept was proven to be successful, stronger components were used to reproduce the working design. This time a wheel for a cart was used to replace the CD and the canister in which it rotated was made out of PVC pipe. After combining the cartridge and rotating wheel, we then integrated an electric motor to turn the wheel. Finally, the engine was connected to the Raspberry Pi which will then control it. The next stage will be to develop the code to stop and start the rotation from the Raspberry Pi automatically.

The final step will be to integrate the dispenser with a sensor which will detect when a tablet has been dispensed. Once a tablet has been discovered to be dropped, the dispenser mechanism will stop. The system will also verify that just a single pill was administered, to ensure the machine is operating correctly.

Phase 3.2 Management API and User Interface

After defining the requirements for the medication dispenser, it was clear that there would be many scripts which would be reliant on each other for information. For example, the dispensing mechanism would rely on the cup-sensing device to determine if a cup is in place. A single source of truth would be needed to facilitate these relationships and ensure consistency of information throughout the project. Additionally, a graphical user interface is necessary to manage patients medications efficiently.

A database will be created to facilitate the single source of truth by following the processes outlined below.

1. Define the requirements of the database to aid in identifying the needed data and any relationships.
2. Create the database and normalise tables where appropriate.
3. Define an API to allow interaction with the database via HTTP.
4. Revise and repeat based off new information acquired.

The GUI will be developed by following the processes outlined below:

1. Define the requirements of the GUI
2. Define requirements for the front end.
3. Implement basic prototype to test interfacing with the back end API.
4. Implement additional functionality as needed.
5. Review, test, and update.

Once the backend is in a usable state, the various scripts which control components of the dispenser will interface with the API to ensure each separate script has access to consistent and accurate information.

Phase 3.3 Notification System

The notification system will be used to alert carers when the patient has not taken their medication as well as provide status updates about the health of the system. The first step in designing the system will be to select the type of notification system that will be used. Secondly, the chosen notification system will be integrated with the existing code developed for the system.

Phase 3.4 RFID Reader

The first step when getting the RFID card reader to work will be to connect it to the Raspberry Pi physically. The code will have to be developed that will allow the Raspberry Pi and existing software to read and write to the RFID cards. Finally, the RFID cards and their unique identifiers will need to be integrated with the existing identity system used by the dispenser.

Phase 4 Control

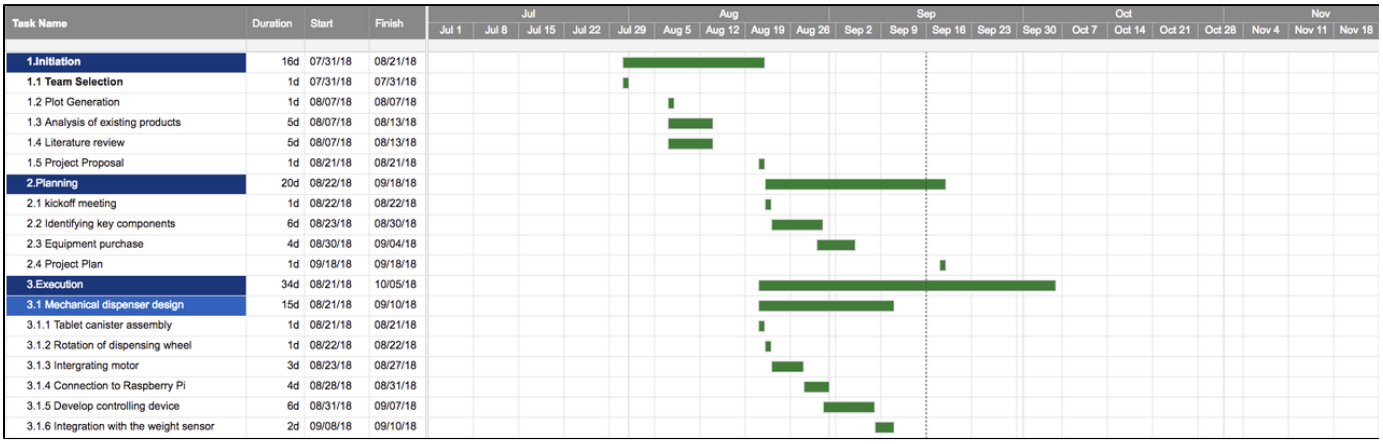
In the Control phase overall monitoring, assessing and comparing planned results with actual results is done to determine the progress of the project.We will have weekly progress meetings to ensure the tasks given to each member is completed within the desired time frame and to resolved any issues found while doing the tasks assigned.Each member is to submit a weekly report at the end of the week to the project manager to receive feedback.The tasks carried out during the project life cycle will be documented and updated accordingly throughout this phase.

Phase 5 Closure

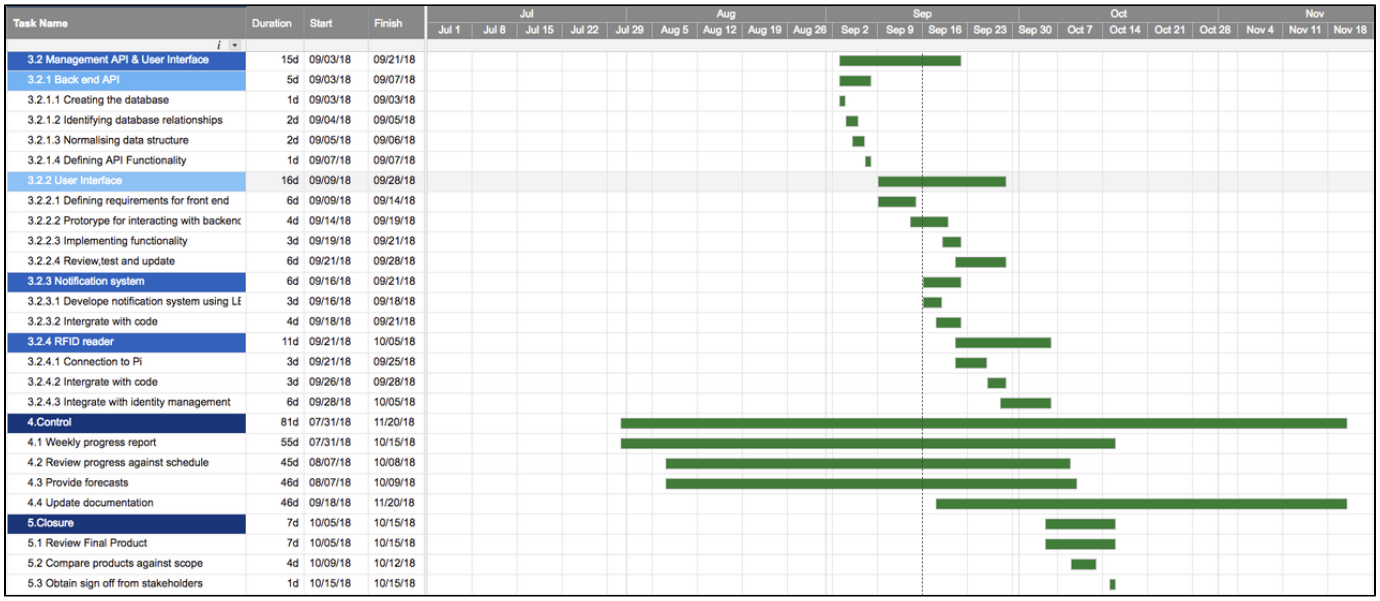
A project closure report will be submitted in this phase.All the activities required to close the project will be listed in the report to ensure the project closure is completed smoothly and efficiently.The product will be reviewed and tested for the practical assessment.The project will close with sign off from the stakeholders.

Schedule

Within three months the project will be completed. The project is further broken down into milestones, which each represent an important event or a deliverable.



Task Name	Duration	Start	Finish
1.Initiation	16d	07/31/18	08/21/18
1.1 Team Selection	1d	07/31/18	07/31/18
1.2 Plot Generation	1d	08/07/18	08/07/18
1.3 Analysis of existing products	5d	08/07/18	08/13/18
1.4 Literature review	5d	08/07/18	08/13/18
1.5 Project Proposal	1d	08/21/18	08/21/18
2.Planning	20d	08/22/18	09/18/18
2.1 kickoff meeting	1d	08/22/18	08/22/18
2.2 Identifying key components	6d	08/23/18	08/30/18
2.3 Equipment purchase	4d	08/30/18	09/04/18
2.4 Project Plan	1d	09/18/18	09/18/18
3.Execution	34d	08/21/18	10/05/18
3.1 Mechanical dispenser design	15d	08/21/18	09/10/18
3.1.1 Tablet canister assembly	1d	08/21/18	08/21/18
3.1.2 Rotation of dispensing wheel	1d	08/22/18	08/22/18
3.1.3 Intergrating motor	3d	08/23/18	08/27/18
3.1.4 Connection to Raspberry Pi	4d	08/28/18	08/31/18
3.1.5 Develop controlling device	6d	08/31/18	09/07/18
3.1.6 Integration with the weight sensor	2d	09/08/18	09/10/18



Task Name	Duration	Start	Finish
3.2 Management API & User Interface	15d	09/03/18	09/21/18
3.2.1 Back end API	5d	09/03/18	09/07/18
3.2.1.1 Creating the database	1d	09/03/18	09/03/18
3.2.1.2 Identifying database relationships	2d	09/04/18	09/05/18
3.2.1.3 Normalising data structure	2d	09/05/18	09/06/18
3.2.1.4 Defining API Functionality	1d	09/07/18	09/07/18
3.2.2 User Interface	16d	09/09/18	09/28/18
3.2.2.1 Defining requirements for front end	6d	09/09/18	09/14/18
3.2.2.2 Protorype for interacting with backenc	4d	09/14/18	09/19/18
3.2.2.3 Implementing functionality	3d	09/19/18	09/21/18
3.2.2.4 Review,test and update	6d	09/21/18	09/28/18
3.2.3 Notification system	6d	09/16/18	09/21/18
3.2.3.1 Develope notification system using LE	3d	09/16/18	09/18/18
3.2.3.2 Intergrate with code	4d	09/18/18	09/21/18
3.2.4 RFID reader	11d	09/21/18	10/05/18
3.2.4.1 Connection to Pi	3d	09/21/18	09/25/18
3.2.4.2 Intergrate with code	3d	09/26/18	09/28/18
3.2.4.3 Integrate with identity management	6d	09/28/18	10/05/18
4.Control	81d	07/31/18	11/20/18
4.1 Weekly progress report	55d	07/31/18	10/15/18
4.2 Review progress against schedule	45d	08/07/18	10/08/18
4.3 Provide forecasts	46d	08/07/18	10/09/18
4.4 Update documentation	46d	09/18/18	11/20/18
5.Closure	7d	10/05/18	10/15/18
5.1 Review Final Product	7d	10/05/18	10/15/18
5.2 Compare products against scope	4d	10/09/18	10/12/18
5.3 Obtain sign off from stakeholders	1d	10/15/18	10/15/18

Milestones

Milestone	Date	Status	Deliverable	Responsible	Issues/Comments
Team Selection	31 Jul 2018	Completed	Team	All	
Gain Acceptance of Idea	21 Aug 2018	Completed	Project Proposal	All	
Creation of Physical Dispenser Unit	10 Sep 2018	Completed	Physical Dispenser Unit	Lachlan Gabb and fariha uddin	• Developed prototype with success, now working on developing the version to be used for the project.
Creation of Backend Database	07 Sep 2018	Completed	Database	Daniel Moore	
Creation of Patient and Carer Notification System	21 Sep 2018	Not Started	Notification System	fariha uddin and TERRY NG	• Take a look at pushover or Microsoft flow. • Python scripts are available to work with either of those.

Creation of Verification System	10 Sep 2018	In progress	Weight Sensor Integration	Lachlan Gabb and TERRY NG	
Integration of RFID reader	05 Oct 2018	In Progress	RFID Reader	Lachlan Gabb	
Combining all System (Final Prototype)	10 Oct 2018	Not Started	Working Dispenser System	All	
Penetration Test Performed and Vulnerabilities Mitigated	02 Nov 2018	Not Stated	Penetration Test Report	Daniel Moore	
Construction of Dispensing Unit Structure	02 Nov 2018	Not Started	Completed System	All	

Budget

This project aims to produce a cost-effective medication dispenser. As all of the parts required for the project are available off the shelf, we set a budget of $350. The following table summarises the project budget.

Note: All prices listed are in Australian Dollars.

Product	Quantity	Cost (AUD)
Raspberry Pi	1	$55
LED Lighting	1	$20
RFID Reader	1	$15
RFID Cards	4	$10
Weight Sensor	1	$8.50
Motor	1	$20
Rotating Wheel	1	$3
PVC Pipe	4	$4.50
Reserve	N/A	$214
Total		$350

Project Status Report

Reporting Period : 31.07.2018 to 18.09.2018

Project Name: Automated Medication Dispenser Project

Project Manager : Daniel Moore

Team Members' Name: Fariha Uddin, Lachlan Gabb & NG Wai Pan

Date: 18.09.2018

Work completed this reporting period:

- Team selection
- Acceptance of idea
- Creation of physical dispenser unit
- Back end database

Work to complete next reporting period:

- Notification system
- Integration of RFID reader
- User Interface
- Final Prototype

What's going well and why:

- The medication dispenser and the rotation of the dispenser was successfully built using Jaycar YG2734 motor.
- The back end database for storing patient's information and prescribed medications has been built.

What's not going well and why:

- **Verification system :** We are still working on the code to make the dispenser dispense the right amount of medication at a time.
- **RFID reader :** We need to make the RFID scan the wrist band to identify the user and dispense their medications.

Suggestions/Issues:

- **System analysis :** Test to ensure that individual parts will integrate with the system as a whole.
- **Documentation :** Documentation has to be up-to-date

Project changes : N/A

Appendix

APPENDIX A

Risk Rating Matrix

The table below is the derivation of the impact definitions.

	Impact	**Negligible**	**Low**	**Moderate**	**Major**	**Extreme**
Likelihood						
Almost Certain		Medium	High	High	Critical	Critical
Likely		Medium	Medium	High	High	Critical
Possible		Low	Medium	Medium	High	High
Unlikely		Low	Low	Medium	Medium	High
Rare		Low	Low	Low	Medium	Medium

Risk Likelihood Levels

The table below is the derivation of the Likelihood Levels.

Likelihood	**Description**	**Characteristics**
1	Almost Certain	The event is likely to occur more often than once per month
2	Likely	The event is likely to occur once between a month and six months
3	Possible	The event may occur between six months and two years
4	Unlikely	The event may occur once every 2-5 years
5	Rare	The event may occur once every five years or longer

Risk Impact Levels

Impact	**Organisational Impact**	**Financial & Resources**	**Reputation**

Extreme	External, all clients	Death of staff Major Financial loss Destruction or serious damage to most assets	Organisation found liable in a legal action Sustained media attention.
Major	The whole of an organisation or several clients.	Injury to staff, loss of critical mass of team Highly Significant Financial loss Destruction or severe damage to critical physical or information assets.	Negative public comments by existing clients to potential clients.
Moderate	A single Core Service.	Permanent loss of key staff; Significant or Sustained Financial loss Damage to physical or information assets	Negative public comments that may influence potential clients.
Low	Affects Organisation	Temporary loss of key staff; Financial loss	Adverse or indirect comments in the press/media.
Negligible	Team level	Minor Financial Loss	Internal impact only.

RFID Information

Required Libraries

SPI

- Clone from: https://github.com/lthiery/SPI-Py.git
- Install the Setup.py

READER ITSELF

- Clone from: https://github.com/pimylifeup/MFRC522-python.git

Reading From the RFID Tag

```
import RPi.GPIO as GPIO
import SimpleMFRC522

reader = SimpleMFRC522.SimpleMFRC522()

try:
        id, text = reader.read()
        print(id)
        print(text)
finally:
        GPIO.cleanup()
```

Writing to RFID Tag

```
import RPi.GPIO as GPIO
import SimpleMFRC522

reader = SimpleMFRC522.SimpleMFRC522()

try:
        text = "The Answer is 42"
        reader.write(text)

finally:
 GPIO.cleanup()
```

Sample Research Papers

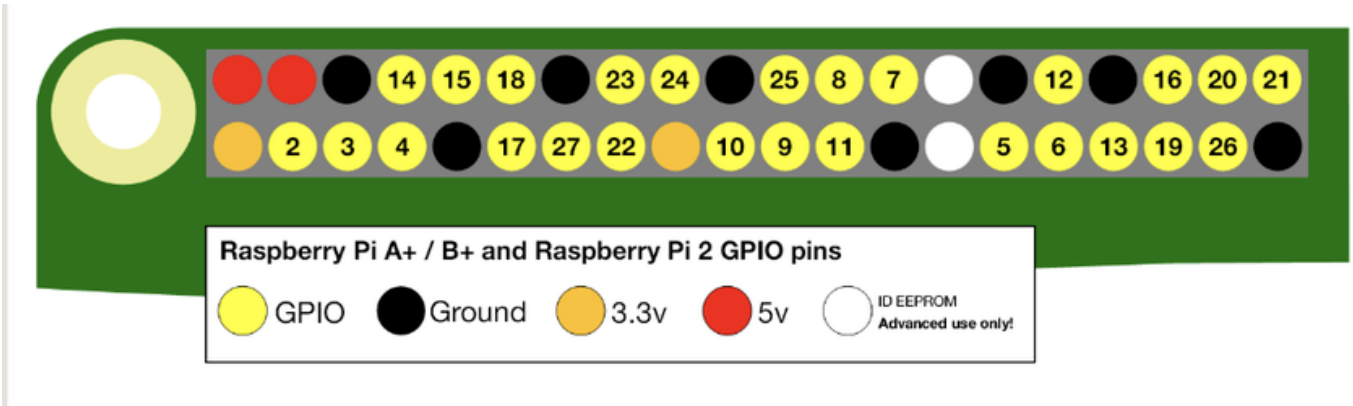
Sensor Connection

Other Sensor Pages

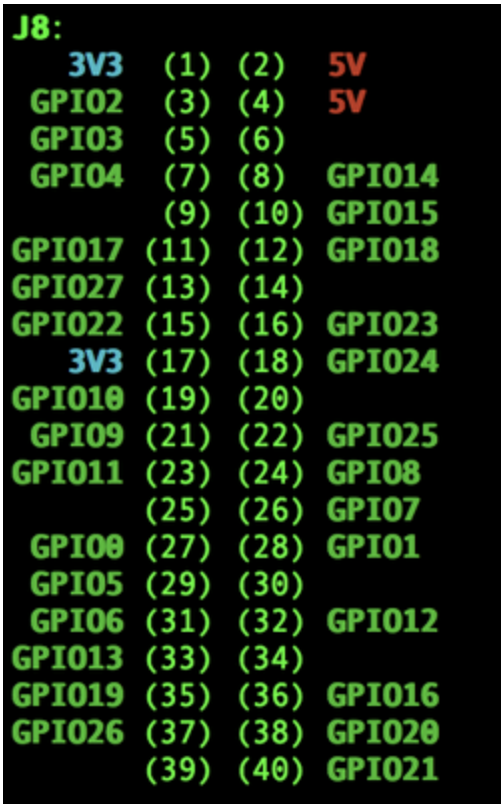
- Drop Meds
- Tablet Detection Sensor

Pinout

The connectors on the Raspberry Pi have the following properties



Their purposes are made clearer in this diagram



Pin Allocation

NOTES

- Pins 19, 21, 23, 24, 26 are reserved for the SPI connections, so they will need to be in use by the RFID card reader
- Pins 3 and 5 have hardware pull up and pull down resistors; they do not work as standard inputs
- The Pi has 40 GPIO pins, however 14 of them are power, ground or special purpose meaning they cannot be used
    - This leaves us with 26 to use
    - Currently we are using 16, leaving 10 available
    - **Note:** The weight sensor or the LED controllers are not yet incorporated into the design

Pin (Board Numbering)	Purpose
1	3.3v Power for Tablet Detection Systems
2	• Has hardware pull up and down resistors • Will not be used
3	• Has hardware pull up and down resistors • Will not be used
6	Ground connection for tablet detection systems
7	Digital input for Tablet Detection System 1
8	Digital input for Tablet Detection System 2
10	Digital input for Tablet Detection System 3
11	Output for Tablet Dispenser Motor 1
12	Output for Tablet Dispenser Motor 2
13	Output for Tablet Dispenser Motor 3
14	Ground Connections for Tablet Dispenser Motor Controllers
17	• 3.3v Power • Used by RFID Reader

19	• SPI MOSI • To be used by the RFID Reader
20	• Ground • To be used by RFID Reader
21	• SPI MISO • To be used by the RFID Reader
22	• SPI Reset • To be used by the RFID Reader
23	• SPI SCK • To be used by the RFID Reader
24	• SPI SDK • To be used by the RFID Reader
26	• SPI CE1 • To be used by the RFID Card Reader
40	• Booted Pin • Must be set to high when the Pi is ready and will enable to motors to work.

Tablet Detector

- The tablet detector requires the following connections to the Raspberry Pi
  - 3.3v
  - Ground
  - One Digital Input Pin
- It will be connected to physical pin 7 on the Raspberry Pi

Drop Meds

This contains the main code to control the dispenser motors and sensors

```
# Drop Pills
# Used to control the dropping of tablets from the dispenser
import RPi.GPIO as GPIO # Import the GPIO for the Raspberry Pi
import time # Import the time code
GPIO.setmode(GPIO.BOARD) # Setup Radspberry Pi pins as board numbering

### Define the GPIO Pins in Use ### Please note: These are the physical
board pin numbers
BootPin = 40

MotorOnePin = 11
MotorTwoPin = 12
MotorThreePin = 13

DetectorOnePin = 7
DetectorTwoPin = 8
DetectorThreePin = 10

DispenserOne = [1, MotorOnePin, DetectorOnePin] # Creates an array so
all dispenser information is in one location
DispenserTwo = [2, MotorTwoPin, DetectorTwoPin]
DispenserThree = [3, MotorThreePin, DetectorThreePin]
```

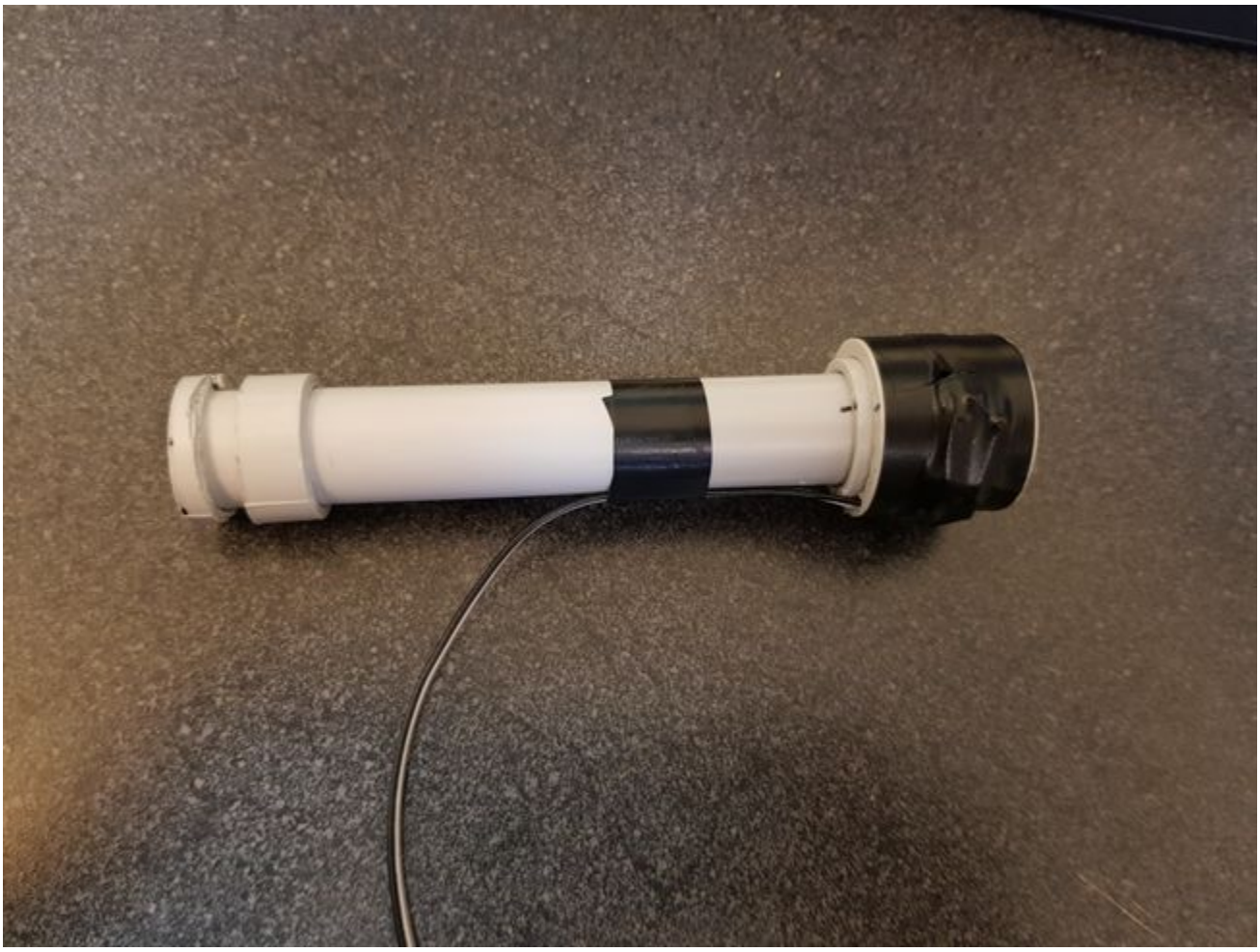
```
### Setup the GPIO Pins for Use ###
# This defines whether the pins are going to be used for input or output
GPIO.setup(BootPin, GPIO.OUT)
GPIO.setup(MotorOnePin, GPIO.OUT) # Setup the motor pins to be output
GPIO.setup(MotorTwoPin, GPIO.OUT)
GPIO.setup(MotorThreePin, GPIO.OUT)

GPIO.setup(DetectorOnePin, GPIO.IN) # Setup the sensor pins to be input
GPIO.setup(DetectorTwoPin, GPIO.IN)
GPIO.setup(DetectorThreePin, GPIO.IN)

# As a 1 means stop for the board, all of the pins must be set to high
before system starts
GPIO.output(MotorOnePin, GPIO.HIGH)
GPIO.output(MotorTwoPin, GPIO.HIGH)
GPIO.output(MotorThreePin, GPIO.HIGH)
GPIO.output(BootPin, GPIO.HIGH) # Set to high to inform the motor driver
board that the script is ready
print("The safety pin has been removed")



def DetectPill(DetectorNumber): # Used to detect when a pill has been
dropped. Needs to be told which sensor to look at
while True: # Just Go
if GPIO.input(DetectorNumber) == True: # Check if the PIN has gone high
time.sleep(0.04) # If it has, wait aproximately 30 milliseconds for the
pill to pass and check it has
if GPIO.input(DetectorNumber) == False: # Check the pill has now passed
the sensor
return True

else: # If it has not, something is likely blocking the sensor
print("A pill may be stuck")
time.sleep(1) # Sleep for a moment to see if the blockage clears
time.sleep(0.001) # Sleep for one millisecond between checks

def Dispense(DispenserNumber, Quantity): # Used to actually dispense
Pills. Provide the dispenser number and the amount of tablets that needs
to be dispensed
for Tablet in range(0, Quantity): # Start from 0 and increment for each
tablet
GPIO.output(DispenserNumber[1], GPIO.LOW) # Turn the motor on
print("Motor is running")
while DetectPill(DispenserNumber[2]) != True: # Contiunue until a pill
has been dropped
pass # Used to setup a POST test loop
GPIO.output(DispenserNumber[1], GPIO.HIGH) # Executes once a pill has
been dropped and turns the motor back off
print("Motor Has Stopped")
```

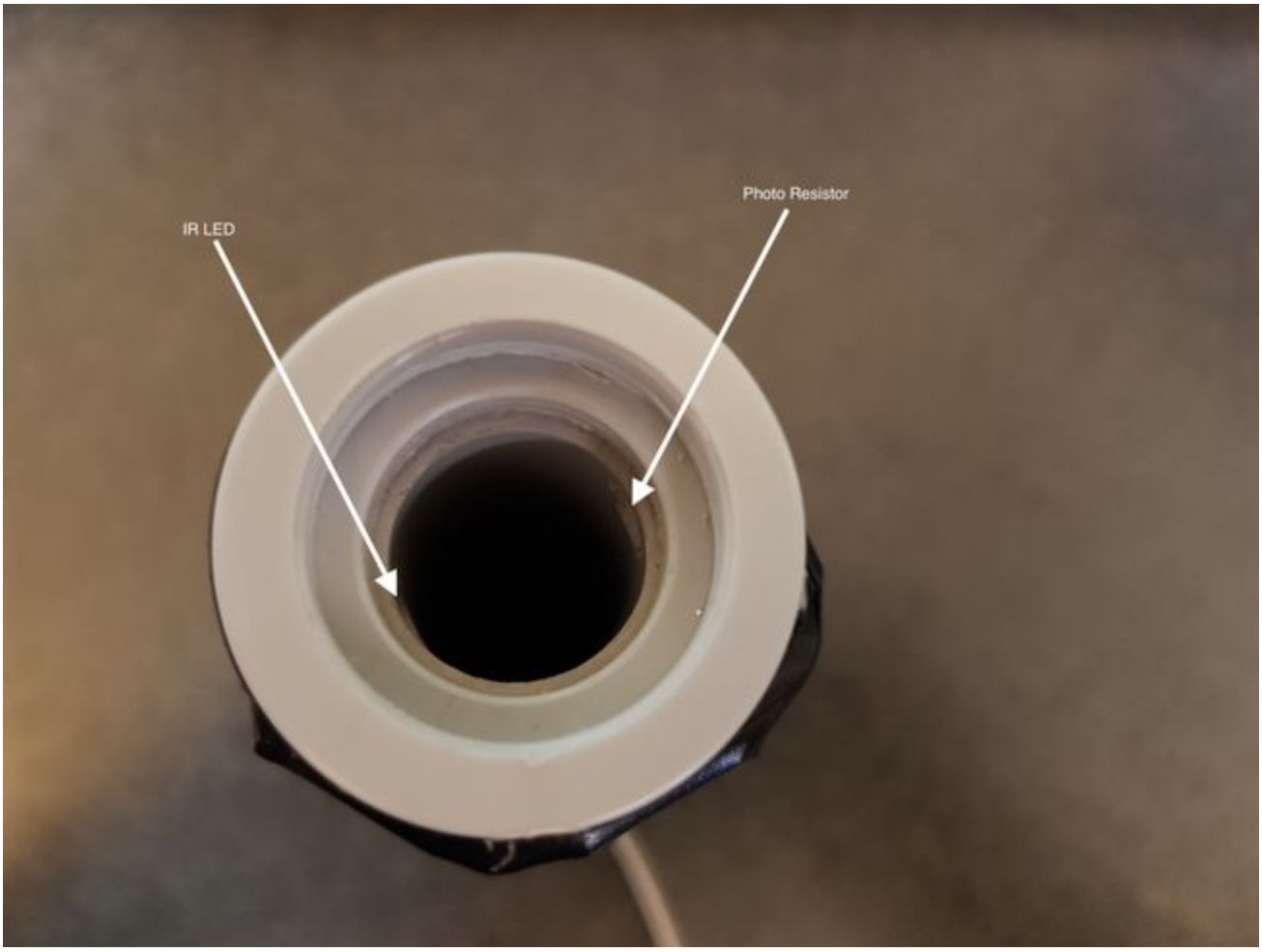
```
Dispense(DispenserThree, 6)
```

Tablet Detection Sensor

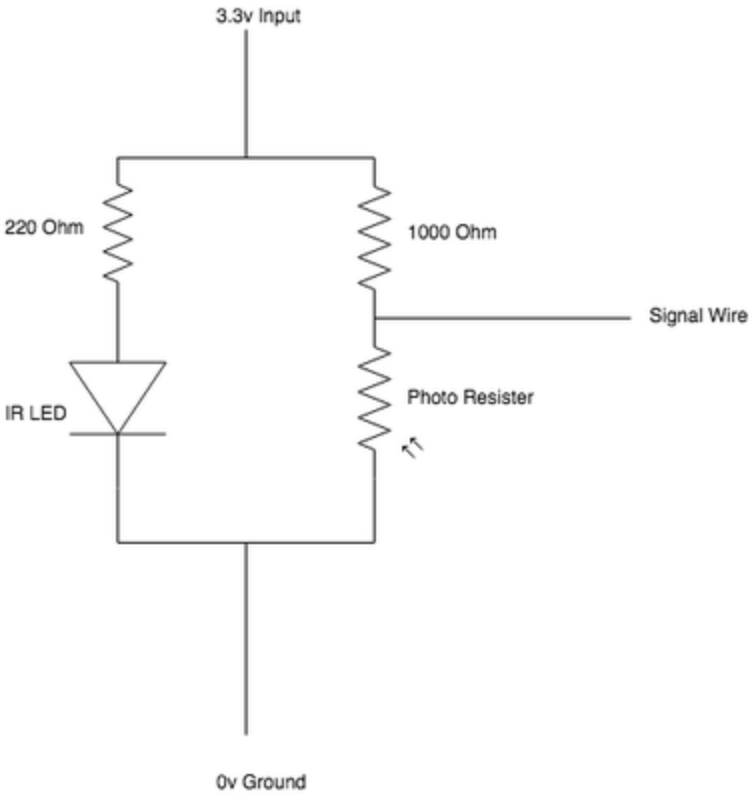
- The tablet detection sensor will be used to detect when the dispenser has successfully dropped a tablet
- The sensor is located in part of the pipe through which the tablet much travel
- The sensors consists of an infra-red LED and a light sensitive resistor
    - While the pipe is clear, the light from the LED shines into the resistor, causing it to be open circuit
        - As a result no power flows through the circuit
    - However, when a tablet passes through the pipe, it will momentarily block the light from the LED reaching the light sensitive resistor
        - When this occurs, the resistor becomes a short circuit and allows power to flow through the circuit
        - This causes the output of the circuit to go from 0v to 3.3v which is then detected by the Raspberry Pi

Physical Design





Electrical Design



BASIC DETECTION CODE

```
PinNumber = 7 # Define the pin on which the Pill Sensor will be detected

import RPi.GPIO as GPIO # Import the GPIO for the Raspberry Pi
import time # Import the time code

GPIO.setmode(GPIO.BOARD) # Setup Radspberry Pi pins as board numbering
GPIO.setup(PinNumber, GPIO.IN) # Setup the selected pin to be an input

Count = 0 # Used for counting

while True: # Just Go
 if GPIO.input(PinNumber) == True: # Check if the PIN has gone high
  time.sleep(0.04) # If it has, wait aproximately 30 milliseconds for
the pill to pass and check it has
  if GPIO.input(PinNumber) == False: # Check the pill has now passed the
sensor
   PillDropped = 1 # If it has, a pill has been dropped
   Count = Count + 1
   print("You have now dropped " + str(Count) + " pills.")

  else: # If it has not, something is likely blocking the sensor
   print("Something appears to be stuck")
   time.sleep(1) # Sleep for a moment to see if the blockage clears
 time.sleep(0.001) # Sleep for one millisecond between checks
```

Main Med

The following code will be used to control the dispensing of medication. This controls:

- RFID Reader
- Tablet Dispensers
- Tablet Sensors

```
# Drop Pills
# Used to control the dropping of tablets from the dispenser
import RPi.GPIO as GPIO # Import the GPIO for the Raspberry Pi
import time # Import the time code
import SimpleMFRC522 # Import the card reader code (For the RFID reader)
import requests
import json
import datetime
GPIO.setmode(GPIO.BCM) # Setup Radspberry Pi pins as board numbering
reader = SimpleMFRC522.SimpleMFRC522() # Setup the Reader Object

# ### Define the GPIO Pins in Use ### Please note: These are the
physical board pin numbers
BootPin = 21
```

```
MotorOnePin = 17 # 11 Physical
MotorTwoPin = 18 # 12 Physical
MotorThreePin = 27 # 13 Physical

DetectorOnePin = 4 # 7 Physical
DetectorTwoPin = 14 # 8 Physical
DetectorThreePin = 15 # 10 Physical

DispenserOne = [1, MotorOnePin, DetectorOnePin] # Creates an array so
all dispenser information is in one location
DispenserTwo = [2, MotorTwoPin, DetectorTwoPin]
DispenserThree = [3, MotorThreePin, DetectorThreePin]

DispenserArray = [DispenserOne, DispenserTwo, DispenserThree]

# print(DispenserArray)

# ### Setup the GPIO Pins for Use ###
# # This defines whether the pins are going to be used for input or
output
GPIO.setup(BootPin, GPIO.OUT)
GPIO.setup(MotorOnePin, GPIO.OUT) # Setup the motor pins to be output
GPIO.setup(MotorTwoPin, GPIO.OUT)
GPIO.setup(MotorThreePin, GPIO.OUT)

GPIO.setup(DetectorOnePin, GPIO.IN) # Setup the sensor pins to be input
GPIO.setup(DetectorTwoPin, GPIO.IN)
GPIO.setup(DetectorThreePin, GPIO.IN)

# # As a 1 means stop for the board, all of the pins must be set to high
before system starts
GPIO.output(MotorOnePin, GPIO.HIGH)
GPIO.output(MotorTwoPin, GPIO.HIGH)
GPIO.output(MotorThreePin, GPIO.HIGH)
GPIO.output(BootPin, GPIO.HIGH) # Set to high to inform the motor driver
board that the script is ready
print("The safety pin has been removed")

APIURL = "http://meds.thegabbs.com:4000"

UPDATE_MEDICATION_QUERY = """
mutation updatePatientMedication($id: ID!, $name: String!, $count:Int!,
$patientId: ID!){
  updatePatientMedication(id: $id, name: $name, count: $count,
patientId:$patientId){
    id
    name
    count
  }
```

```
}
"""

USER_SCHEDULES_QUERY = """
query userSchedules($id: ID!){
    userSchedules(id: $id){
      id
      time
      takenTime
      medications {
        id
        name
        count
        dose
        dispenser
      }
    }
  }
"""

UPDATE_SCHEDULE_MUTATION = """
mutation updatePatientScheduleMutation($id: ID!, $time: Int,
$takenTime:DateTime, $patientId: ID!){
  updatePatientSchedule(id: $id, time:$time, takenTime:$takenTime,
patientId:$patientId){
    id
    time
    takenTime
  }
}
"""

HEADER = {"Authorization":"Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOiJjam12bGplNGYwMDhsMDk
4OG00bjA4eW5kIiwiaWF0IjoxNTM4NzE5MTQxfQ.eF6a36uy_D6eMz3XhD_lTd-4NHzyRnwA
yajtNnHT9l0"}

def DetectPill(DetectorNumber): # Used to detect when a pill has been
dropped. Needs to be told which sensor to look at
    while True: # Just Go
        if GPIO.input(DetectorNumber) == True: # Check if the PIN has
gone high
            time.sleep(0.04) # If it has, wait aproximately 30
milliseconds for the pill to pass and check it has
            if GPIO.input(DetectorNumber) == False: # Check the pill has
now passed the sensor
                return True

            else: # If it has not, something is likely blocking the
sensor
```

```
                print("A pill may be stuck")
                time.sleep(1) # Sleep for a moment to see if the
blockage clears
        time.sleep(0.001) # Sleep for one millisecond between checks

def Dispense(DispenserNumber, Quantity): # Used to actually dispense
Pills. Provide the dispenser number and the amount of tablets that needs
to be dispensed
    for Tablet in range(0, Quantity): # Start from 0 and increment for
each tablet
        GPIO.output(DispenserNumber[1], GPIO.LOW) # Turn the motor on
        print("Motor is running")
        while DetectPill(DispenserNumber[2]) != True: # Contiunue until
a pill has been dropped
            pass # Used to setup a POST test loop
    GPIO.output(DispenserNumber[1], GPIO.HIGH) # Executes once a pill
has been dropped and turns the motor back off
    print("Motor Has Stopped")

def QueryAPI(query='', variables={}, url='', headers={}):
    """
    Make query response
    """
    request = requests.post(url, json={'query': query, 'variables':
variables}, headers=headers) # Sends a post request to the API with the
necessary parameters.
    if request.status_code == 200: # If the request is returned
successfully
        #print(request.json())
        return request.json() # Return the reqest in JSON.
    else:
        raise Exception("Query failed to run by returning code of {}.
{}".format(request.status_code, query)) # Inform the console why the
request was not returned successfully.


while True: # Main Loop
    try: # Try and read the user ID from the RFID card.
        id, UserID = reader.read() # Read the information from the card
        print(UserID)
    except: # If the user ID cannot be read from the card.
        print("Could not read, something must be wrong")
    #UserID = 'cjn5jlmju01l708399k3qmtxs'


    QueryResult = QueryAPI(query=USER_SCHEDULES_QUERY,
variables={'id':UserID}, url=APIURL) # Query the API for the above
user's medication.
    #print(QueryResult)
```

```
    SchedulesList = QueryResult["data"]["userSchedules"] # Get a list of
the different schedules returned by the API

    for Schedule in SchedulesList: # For each schedule in the list
        #print(Schedule)
        LastTaken = Schedule["takenTime"] # Determine the time the
schedule's medication was last taken.

        CurrentTime = datetime.datetime.now() # Get the current time
        CurrentDate = str(datetime.datetime.now().date()) +
"T00:00:00.000Z" # Create the current date string.


        print(LastTaken)
        print(CurrentDate)

        if Schedule["time"] == CurrentTime.hour and CurrentDate !=
LastTaken: # If the scheule is due to be taken within the hour and has
not been previously taken today.
            print("True")

            for Medication in Schedule["medications"]: # For each of the
medications within the schedule.
                #print(Medication)
                Dispenser = Medication["dispenser"] # Extract the
dispenser number associated with the particiular medication.
                Dose = Medication["dose"] # Extract the dose of each
medication.
                Medication['patientId'] = UserID
                #print(Medication)
                for Item in DispenserArray: #Extract the necessary
dispenser to motor and sensor mapping information from the arrays
defined at the top.
                    if Item[0] == Dispenser: # Contiune until it finds
the correct dinspenser mapping.
                        Dispense(Item, Dose) # Actually dispense the
medication.
                        Medication['count'] = Medication['count'] - Dose
# Remove one from the total count of medications available.
                        #print("The medication was just updated")
                        QueryAPI(query=UPDATE_MEDICATION_QUERY,
variables=Medication, url=APIURL, headers=HEADER) # Update the API with
the number of tablets now available.
                        #print(Medication)


            # Update the API with the time the medication was taken.
            Schedule["takenTime"] = str(datetime.datetime.now().date())
# Set the takenTime to be the current date.
            #print(Schedule["takenTime"])
```

```
            Schedule.pop("medications", 0) # Remove the medications from
the dictionary. Needed to make the API call work.
            Schedule["patientId"] = str(UserID) # Add the user ID for
who the schedule needs to be updated.
            #print(Schedule)

            QueryAPI(query=UPDATE_SCHEDULE_MUTATION, variables=Schedule
, url=APIURL, headers=HEADER) # Actually push the updates to the server.


        else: # If it is not time for the medication to be taken
```

```
            print("False")  # Do Nothing.

time.sleep(2)
```

Notification System

The notification system will determine if any patients medications are overdue and send the appropriate person a notification.

The below script should be scheduled to run every 15 minutes or something like that.

Currently, the code does not have the ability to actually send a notification, however there is a function into which the appropriate code can easily be placed.

Code

```
import datetime
import requests
import json

APIURL = "http://meds.thegabbs.com:4000"

USER_SCHEDULES_QUERY = """
query userSchedules($id: ID!){
    userSchedules(id: $id){
      id
      time
      takenTime
      medications {
        id
        name
        count
        dose
        dispenser
      }
    }
  }
"""

HEADER = {"Authorization":"Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOiJjam12bGplNGYwMDhsMDk
4OG00bjA4eW5kIiwiaWF0IjoxNTM4NzE5MTQxfQ.eF6a36uy_D6eMz3XhD_lTd-4NHzyRnwA
yajtNnHT9l0"}

UserList  = ["cjn4d6c5j00830839lhnt3f96"]


def SendNotification(Number, Notification): # The function that actually
sends the notification.
 print(Notification)
```

```
def CheckSchedule(Schedule): # Checks if the schedule provided has any
overdue medications.
 LastTaken = Schedule["takenTime"] # Determine the time the schedule's
medication was last taken.
 CurrentDate = str(datetime.datetime.now().date()) + "T00:00:00.000Z" #
Determine the current date + necessay parameters.
 CurrentTime = datetime.datetime.now().hour # Determine the current hour
 TimeToTake = Schedule["time"] # Determine the time the patient should
have taken their medication.

 if CurrentTime >= TimeToTake and CurrentDate != LastTaken: # Checks if
the patient should have taken their medication as hasn't
  return True # They need to take their medication.
 else:
  return False # They have already taken their medication.



def QueryAPI(query='', variables={}, url='', headers={}): # Same
function as before to query the API
    """
    Make query response
    """
    request = requests.post(url, json={'query': query, 'variables':
variables}, headers=headers) # Sends a post request to the API with the
necessary parameters.
    if request.status_code == 200: # If the request is returned
successfully
        #print(request.json())
        return request.json() # Return the reqest in JSON.
    else:
        raise Exception("Query failed to run by returning code of {}.
{}".format(request.status_code, query)) # Inform the console why the
request was not returned successfully.

def main():
 for UserID in UserList: # For each user of the system.
  QueryResult = QueryAPI(query=USER_SCHEDULES_QUERY,
variables={'id':UserID}, url=APIURL) # Query the API for their
schedules.
  SchedulesList = QueryResult["data"]["userSchedules"] # Get a list of
the different schedules returned by the API

  for Schedule in SchedulesList: # Iterate through each of their
schedules.
   MedicationOverdue = CheckSchedule(Schedule) # Check if the medication
is overdue

   if MedicationOverdue: # If the medication is overdue, send a
notification.
```

```
    SendNotification(40404040, "Your patient has not taken their
medication.")
```

```
if __name__ == '__main__':
  main()
```

Project Management Plan (Mel's "Practical Assessment")

Mel's Verbal Instructions

- How it all works
    - If someone were to read it they understand how it works
    - How everything works together
- How the physical dispenser works

Results/Conclusions

Table of Contents

What Went Right

RASPBERRY PI READING RFID CARD

The process of getting the Raspberry Pi to read and write to the RFID card went well. The RFID module we obtained for the project was designed to operate with the Raspberry Pi. It provided a guide on how to physically connect it to the Pi and also came with a Python library that allows us to easy communicate with the device. Upon connecting the device and integrating its supplied code with our own, we were able to read and write data to the RFID cards without any troubleshooting required.

RFID DATABASE INTEGRATION

Once the RFID scanner was operating, it provided our existing code with the unique ID of the user associated with the RFID card. Our code was then able to build a query to the database using this ID to request medication and scheduling information for the user. This worked perfectly as the code was already using a user's ID to manually request scheduling information.

SCHEDULING

All the code for controlling the dispenser was designed around hard coded values for the medications and schedules it would dispense. Upon transitioning the code over to use the schedules setup in the graphical interface the code worked perfectly again. Regardless of what values were entered in the web interface, the dispenser always dispensed the correct amount of each medication at the correct time.

What Went Wrong

DIFFICULTY DISPENSING THE PILLS

The process of actually dispensing pills in a predictable manner was quite a challenge throughout the project. The problems occurred as a result of trying to make the dispenser dispense tablets of vastly different sizes. This resulted in the dispenser jamming, dispensing more than one tablet at a time or sometimes even crushing the tablets. The decision was made to narrow the range of tablets that could be dispensed for this prototype. Once this occurred the dispenser was able to reliably dispense either 0 or 1 tablets for each rotation. As the sensor is able to detect if a tablet is not dispensed, we could just keep the dispenser running until it is dispensed.

ELECTRICAL CIRCUIT DESIGN

When designing the electrical circuit to allow the Raspberry Pi to control the motors, there were a few problems. Firstly, the Mosfets obtained to switch the motors on and off were not capable of being turned on using the 3.3 volts the Pi could product. To overcome this problem, another transistor was added. The Pi would switch the transistor, which would then switch on a 12 volt supply to the mosfet, which would in turn switch on the motor.

Another small obstacle came from the type of Mosfets we obtained for the project. The original concept was to have the Raspberry Pi send a 3.3 volt signal to turn the motors on, however the mosfets obtained for the project operated opposite to what was expected. Instead of applying power to turn the motors on, power had to be applied to turn the motors off. This required the logic of our code to be inverted so the motors operated correctly.

Additionally, this presented another problem. While the Raspberry Pi was booting, a 20 to 30 second process, it could not supply power to keep the motors in the off state. Therefore, all three motors would operate and dispense medication continuously from the moment the Raspberry Pi booted until it had loaded our code. To overcome this problem, another GPIO pin from the Raspberry Pi was used and connected to the power supply providing power to the motors. In order for the power supply to be turned on this GPIO pin must be set at 3.3 volts. Therefore, once the Pi boots and sets the motors outputs to high (to stop the motors) the additional GPIO pin is set to high turning on the motor power supply.

CHANGED SCOPE FOR LED

A part of the initial plan was to include Light emitting diode (LED) as part of the notification system. But the project includes a fair amount of connections and most of the GPIO pins have already been occupied by the sensors. Under the circumstances, we ran out of connections and space for the LEDs. So the final project shall not have any LEDs for notification. Instead it will have an SMS reminder system to notify the user of the scheduled medication.

Test Procedure

PHYSICAL DISPENSER

Once the physical dispenser was deemed to be operational, it was tested a number of times to ensure reliability. To do this, the dispenser will filled with exactly 200 pills and programmed to dispense all 200. The process was observed to ensure the motor stopped rotating at the moment the last pill was dispensed. This process ensured:

- The dispenser assembly was able to successfully dispense all 200 pills without jamming, stopping or destroying any of the pills.
- The Raspberry Pi was accurately counting the number of pills the dispenser dispensed.
- There were no other problems with the delivery mechanisms.

The above process was repeated three times for each of the three dispenser modules, ensuring they were all operating correctly. During the testing it was found that dispenser number three was consistently failing to dispense pills. This was found to be a result of an error during the building of the unit and was quickly corrected.

Sample Data Collection

TIME TAKEN TO DROP PILL

The following data was collected to analyse how long the dispenser takes to drop a single pill. To collect the data, the RFID card was swiped and the Raspberry Pi calculated the time in seconds between starting the dispenser motor and the moment the sensor detects a pill has been dispensed.

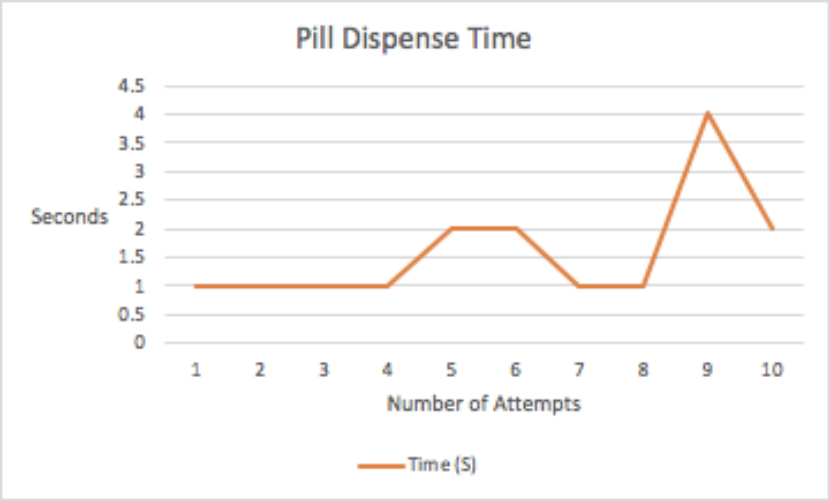
Note: The values have been rounded to the nearest whole second.

The following data was collected:

Attempt Number	Time (In Seconds)
1	1
2	1
3	1
4	1
5	2
6	2

7	1
8	1
9	4
10	2

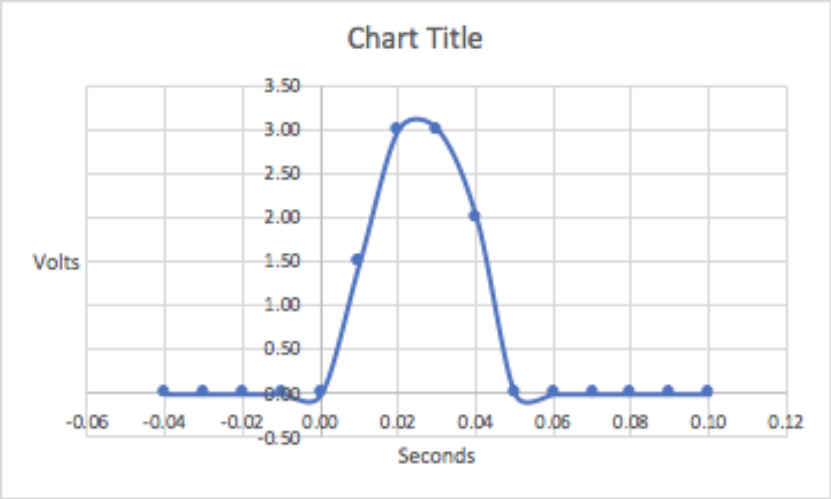
This results in an average time of 1.6 seconds to dispense a single pill, which correlates well with the results observed throughout the testing of the product. The graph below shows a summary of the data collected:



ELECTRICAL PULSE FROM PILL SENSOR

The sensors that detects pills creates a short pulse of electricity out its sensor connector when the pill passes through it and blocks an infra-red beam. Having built the sensor ourselves, there was no documentation available to tell us what this pulse of electricty would look like, therefore making it difficult to program the Raspberry Pi to look for such an event. To overcome this problem, we connected our sensor to an oscilloscope which was able to graph the pulse of electricity. The following information was gained from the process:

Time (Seconds)	Voltage
-0.04	0
-0.03	0
-0.02	0
-0.01	0
0	0
0.01	1.5
0.02	3
0.03	3
0.04	2
0.05	0
0.06	0
0.07	0
0.08	0
0.09	0
0.1	0



Note: The tablet was dropped through the sensor a t=0

Lessons Learned

This section includes the lessons and knowledge gained during the entire procedure of the project :

- Controlling an electric motor by Raspberry Pi
- Functioning method of RFID scanners
- Integration between Raspberry Pi and different types of sensor
- How Infrared LED and light sensitive resistors can detect dispensed pills
- Using python scripts to retrieve data from the database

Recommended Future Work

REDUCING DISPENSER SIZE

The dispenser itself is a bit bulky in size and hard to move. The containers to hold the pills for the dispenser and the wheels used are quite large. A smaller size of these components would reduce the size of the dispenser and make it easier to move it around. All cirucit board sizes can also be optimized to reduce surface area greatly.

DISPENSING DIFFERENT SHAPED PILLS

Currently the dispenser is capable of dispensing only a certain shape of pills. The physical design of the wheel only allows it to drop circular pills. The wheels can be designed to drop pills of different shapes as a future work.

ALERTING COMPONENTS

For the project we have used an SMS reminder system for user to remind them to take the medication. Even though the initial plan was to include LEDs to the dispenser to notify the user to take pills and to inform them whether the pills were running low. But the LEDs could not be added to the project due to lack of space and connection. One of the recommended future works is to add an LED light and a small speaker to alert the user when to take their medication. The lights will flash and the speaker will emit a sound starting at the time the medication is dispensed and ending within a few seconds. These can be powered through the microcontroller and will be turned on and off through commands sent from the microcontroller.

ABSTRACT :

This report has outlined the major project called "Magic Meds" by our group. The paper covers the entire project lifecyle and how the project was divided into different steps to meet the objectives and successfully complete the project.

Magic Meds is an automated medication dispenser, designed for elderly people to reduce the chances of taking wrong medication and incorrect dosage. As people grow older, they depend more upon the support for medical care and assistance with medication. Especially people with Alzheimer and Dementia have higher risk of forgetting their medication and sometimes they might end up taking incorrect dosage as well. Magic Meds has been designed in order to assist elderly people with their medication and also sends them a notification while it is time to take medications. The pill dispensing process only requires RFID scanning of the user to get the correct medication.

The project proposal, literature review, project plan and methodoloy, everything has been presented in the report in order. It has comprehensively discussed how the dispenser has been designed. It is a full overview of all components used in the system and how they integrate with each other to accomplish it's goal of dispensing pill according to a schedule set by a user. It further includes what accomplishments were made, what

difficulties and challenged the group encountered and the possibilities for future work on the project, as well as ethical consideration of the design.

ACCOMPLISHMENTS :

At the end of the project, we successfully made a working prototype. The dispenser itself is a bit chunky but the end result was a fully functional proof of concept that meets all the goals described in the Project Proposal section. The project has been developed using as much as possible suing modern technology such as - Raspberry Pi, electric motors, infrared LED and sensors. The user interface is now able to add new users, enter the names and amount of medication needed and set schedules for the user. Each user is identified by their unique RFID and once they scan the ID, the dispenser dispenses the predefined medication required at that time. The dispenser is self contained and can be moved and will still function as expected. Furthermopre, as a part of the notification system, we have used SMS service provided by AWS. It will send an SMS to the user when they are expected to take the medication. The initial estimated cost for the project was $300 but the final prototype cost us only $90. So the project was done under the estimated budget.

ETHICAL CONSIDERATION :

The ACS code of Ethics and IEEE code of Ethics have been used as guidelines for the ethical consideration of this project. We have adopted the ACS code of Ethics (2) Enhancement of Quality of life. The dispenser has been designed bearing in mind that it can enhance life of other particularly people with certain diseases and disabiltiies. We have also adopted the IEEE code of Ethics and under the ethical consideration of IEEE code of Ethics : the dispenser will always dispense the amount of pills specified by the user and this is ensured by precise experimentation and testing. Also the sole motivation behind the project is from sheer interest of producing a usable product for the elderly people. And any other form of motivation such as bribery will not be used to alter the desired end goals of the project (Lee, Groeteke & Hewaparakarama, 2016).

Figure 1 : Risk Rating Matrix

Figure 2 : Associated Risk Ratings

Figure 3 : Work Breakdown Structure

Figure 4 : Gantt Chart

Figure 5 : Pill Dispense Time Chart

Figure 6 : Voltage Change with Time Chart

Product requirements

Add Product requirements

Title	Designer	Developers	Document owner	Document status	Epic	QA	Target release
Anti- keyless theft			TERRY NG	DRAFT	Anti-keyless theft		
Beach Bag Anti-Theft			Lachlan Gabb	DRAFT			0.0.1
Washing Rain Detector			Lachlan Gabb	DRAFT			
Medication Reminder			Lachlan Gabb	DRAFT			
Doorbell for the Blind			Lachlan Gabb	DRAFT			
The Odd Sock Basket	Daniel Moore		Daniel Moore	DRAFT			0.0.1
Canary Token Detection System			Daniel Moore	DRAFT			0.0.1

Anti- keyless theft

Target release	
Epic	Anti- keyless theft
Document status	**DRAFT**
Document owner	TERRY NG
Designer	
Developers	
QA	

GOALS

Create a small device that could track the GPS / internet /signal between owner mobile to identify the owner or pre-set user is nearby the car when the car engine is starting. the device is connecting to the fuse box of the starter relay if the owner is not nearby the car and someone is going to start the engine. then because of the starter relay is control by the device so the car is not able to startup to prevent car theft.

BACKGROUND AND STRATEGIC FIT

- High tech car theft keep radip increasing in different country, thieves using relay / signal capture device to steal the car without a key

ASSUMPTIONS

- Installation is no technique require and user friendly app / web application. easy to manage and worry free of the car.

REQUIREMENTS

#	Title	User Story	Importance	Notes
1	GPS tracker/ Signal receiver (Wifi, Bluetooth..)	To receive signal that eh owner is nearby the car	Must	
2	Relay	to switch on / off for the starter motor	must	
3	Raspberry Pi	To contorl relay, GPS tracker/ reciver	must	

USER INTERACTION AND DESIGN

QUESTIONS

Below is a list of questions to be addressed as a result of this requirements document:

Question	Outcome

NOT DOING

Project Idea List

Listed below are some project ideas. Follow the links to find more information about each problem/idea:

- Medication Reminder
- Beach Bag Anti-Theft
- Canary Token Detection System
- The Odd Sock Basket
- Washing Rain Detector
- Doorbell for the Blind

Medication Reminder

Target release	
Epic	
Document status	DRAFT
Document owner	Lachlan Gabb
Designer	
Developers	
QA	

THE PROBLEM

When left alone for the day, some older people forget to take their medication at the necessary times. As these times can vary depending on the medication, it can be difficult to remember. If they had a device that provided them with notification when it was time to take the medication. Additionally, should they forget to take the medication within the necessary timeframe, a notification could be sent to their carer to indicate it has been taken.

GOALS

- Provide notification, through the use of a bright light and auditory alert, to the 'patient' when it is time to take their medication. Or through a phone notification depending on the technical ability of the patient.
    - Most likely for regular intervals such as breakfast, lunch and dinner.
- Use the weight of the medication webster pak to determine if the medication has been taken
    - Could use the weight associated with the tablets disappearing or if that is not detectable enough, just the fact that the weight was removed temporarily and then re-added.
- Provide notification to their carer when they take the medication
- Provide notification to a carer if the medication is not taken within a certain buffer timeframe.
- Provide an audible alert when the webster pak is removed from the scale, but not placed back upon it within a certain period of time.
    - To ensure the user is 'encouraged' to follow the procedure.

BACKGROUND AND STRATEGIC FIT

Create a device which uses the weight of a webster pak to determine if an older person has taken their medication. Provide notification to them when it is time to take the medication. Additionally, if the medication is not taken within a certain buffer time period, send a notification to a nominated carer who can then take over.

ASSUMPTIONS

- The person will be capable of placing the webster pak back on the scale after taking their medication.
- The scale will be sensitive enough to detect the drop in weight associated with taking the medication.
- There will be an Internet connection in place which can be used to send the notification messages.

REQUIREMENTS

#	Title	User Story	Importance	Notes
1	Tablet Detection	The device must be able to recognise when I remove my tablets.	Must Have	
2	Notification User	The device must notify me when it is time to take my tablet.	Must have	
3	Forget Notification Carer	The device must notify my carer if I do not take my tablet.	Must have	
4	Taken Notification Carer	The device should alert my carer when I take my medication	Good to have	Could be useful in tracking whether they had it on time or not. If they have it late, do adjustments need to be made to the next medication interval.
5	Bottle nearing empty	When the pill bottle is nearing empty, I should be notified so I am able to get another prescription.		• This could also be extended in the future to provide automatic notification to doctors through a dashboard/API

USER INTERACTION AND DESIGN

QUESTIONS

Below is a list of questions to be addressed as a result of this requirements document:

Question	Outcome

NOT DOING

DAVID PITCH

Problem: My grandfather, when left home along, often forgets to take his medication at the specified time.

Project Idea: Create a device, which holds a number of small containers which medication will be placed into. The containers will have a bright LED and weight sensor underneath them. At programmable times, the LED beneath the container will illuminate reminding them to take the medication within it. The weight sensors will be used to determine if the medication has been taken. If additional time passes without them taking it, an auditory reminder would sound (perhaps also an email/push notification). Finally, if they still don't take the medication, a notification will be sent to a designated career who can take it from there.

POSSIBLE QUESTIONS

- What sensor will we be using
    - We were looking to use a load sensor based around the HX711 chip
    - There is a module from Sparkfun that will integrate well with a raspberry pi
    - Alternative, if the weight sensor cannot get the measurements we need, we will detect the removal of the container using a capacitance sensor like this one here: https://learn.adafruit.com/capacitive-touch-sensors-on-the-raspberry-pi/overview.
        - We would add something to the bottom of the container which would be able to trigger the sensor.
        - Two metallic plates would do the trick.
- Notification System
    - Would use the Microsoft flow API to generate the alerts
- How would we generate the audible reminder
    - Peizo Buzzer connected to Raspberry Pi
        - He will know what that is

Beach Bag Anti-Theft

Target release	0.0.1
Epic	
Document status	**DRAFT**
Document owner	Lachlan Gabb
Designer	
Developers	
QA	

GOALS

- Create a sensor to detect when a beach bag is moved
- Trigger an alarm and provide notification to the owner
- Provide GPS tracking of stolen bags
- Create a web interface to track a stolen bag and show telemetry data (statistics)

BACKGROUND AND STRATEGIC FIT

Beach theft is prevalent as it is easy to steal while people are swimming, and their bags are unattended. This is occurring at an increasing rate in Australia. This project aims to produce a product that will:

- Deter thieves
- Track thieves
- make users aware of potential theft to intervene
- Provide statistical data on number of incidents across different beaches

ASSUMPTIONS

- Users are non technical
    - Device should be simple to set up and use

REQUIREMENTS

#	Title	User Story	Importance	Notes
1	Mobile Friendly	As a user, I want to be able to use any device, mobile or desktop, to view the user interface	Must have	
2	Notifications	As a user, I want prompt notification if my bag is moved significantly. This notification should be able to reach a smart watch which has cellular capabilities	Must have	
3	Battery Life	As a user, I want the device to have a decent battery life that will sustain a typical beach expedition	Must have	
4	Tracking	As a user, I want to be able to track a bag if it is stolen	Must have	

USER INTERACTION AND DESIGN

QUESTIONS

Below is a list of questions to be addressed as a result of this requirements document:

Question	Outcome

NOT DOING

DAVID PITCH

Problem: When people go to the beach alone, they have to leave their bag unattended on the sand, allowing it to be easily stolen.

Project Idea: The project would aim to create a device or even an app that would use motion sensors to detect when the bag is moved. Once movement is detected it could then trigger an audible alarm to inform other beach goers that the bag may in the process of being stolen and send a notification to the owner. The owner notification would rely on them wearing some sort of water proof smart watch in the water with them. The app/device will also have GPS tracking, increasing the chances of recovering a bag if it is stolen. Additionally, if the previous steps are successful, we intend to create a web page 'heatmap' to show the locations where our sensor detects beach bags are commonly being stolen. This could allow users to make informed decisions about which beaches are safe to leave their bag on.

POSSIBLE QUESTIONS

- What motion sensor will we use
    - This one https://www.instructables.com/id/3-Axis-Accelerometer-ADXL345-With-Raspberry-Pi-Usi/
    - Alternatively we use a phone's built in acceleromoter
- How do we send notifications
    - Use pushover notifications as it has an app for apple watch
    - As the watches now support cellular connectivity this could work at a good range
- How do we track
    - GPS module, like the following for a raspberry pihttps://www.digikey.com.au/product-detail/en/adafruit-industries-llc/746/1528-1153-ND/5353613?utm_adgroup=Eval+Boards+and+Dev+Kits&mkwid=s&pcrid=254327262627&pkw=&pmt=&pdv=c&productid=5353613&slid=&gclid=EAIaIQobChMIifD-t4Hq3AIVWaSWCh27pgTHEAQYAiABEgKrGfD_BwE
    - OR a phones inbuilt GPS

Canary Token Detection System

Target release	0.0.1
Epic	

Document status	DRAFT
Document owner	Daniel Moore
Designer	
Developers	
QA	

GOALS

- Create a intrusion detection monitoring system based similar to Canary Tokens.
- Create tokens for several actions including:
    - Opening a fake sensitive email
    - Browsing a fake sensitive directory
    - Port scanning a particular service
    - etc
- Create a web interface to monitor tokens
- Create email notification system to notify administrators
- log actionable information

BACKGROUND AND STRATEGIC FIT

Canary tokens are a way to detect intrusions based off when certain actions are performed. Traditional intrusion detection systems detect potentially malicious network traffic which is good, however, they cannot detect host based actions which might appear as normal use e.g. Opening and email, browsing a directory, loading a website etc. The idea of canary tokens, is similar to a honey pot, in the sense that the device/site/email/file may appear legitimate, but it actually isn't. Instead, when opened/viewed/scanned/etc, the canary token creates an alert that the object has been viewed/opened/scanned.

ASSUMPTIONS

- Members of the organization will be aware of the canary tokens, this will aid in reducing false positives
- Users of the product will primarily IT security professionals

REQUIREMENTS

#	Title	User Story	Importance	Notes
1	Mobile Friendly	As a user, I want to be able to use any device, mobile or desktop, to view the user interface	Must have	
2	Range of Situations	As a user, I want to be able to deploy canary tokens in a variety of forms, including: • As Emails • As Files • As Network Services	Must have	
3	Notifications	As a user, I want prompt notification if a canary token is triggered, via email and the web interface	Must have	

USER INTERACTION AND DESIGN

QUESTIONS

Below is a list of questions to be addressed as a result of this requirements document:

Question	Outcome

NOT DOING

The Odd Sock Basket

Target release	0.0.1
Epic	

Document status	DRAFT
Document owner	Daniel Moore
Designer	Daniel Moore
Developers	
QA	

GOALS

- Simplify the process of matching socks pairs
- Reduce the size of the odd sock basket
- Keep a registry of odd socks
- Provide a user interface with a interface which shows a list of odd socks, a leader board for socks which have been lost the longest...

BACKGROUND AND STRATEGIC FIT

In my house, there is a laundry basket full of odd socks. When the washing is brought in, socks are paired and put away and any remaining socks which do not have a pair are dumped into the basket. Eventually someone goes through the basket and finds pairs that are in the basket. This product will use RFID chips implanted in socks to keep a registry of socks that are in the odd sock basket. As socks are placed into the odd sock basket, they pass over and RFID reader. This reader will notify the user if the socks pair already exists in the basket, therefore allowing the user to immediatly pair the socks.

ASSUMPTIONS

- Users will be non-technical

REQUIREMENTS

#	Title	User Story	Importance	Notes
1	Speed	As a user, i expect the process of scanning in socks to be fairly fast	Must have	
2	Mobile Friendly	As a user, I expect to be able to access the UI on any device	Must have	
3	Notification	As a user, I expect to be notified when I scan a sock which already exists in the odd sock basket, allowing for it to be immediately paired up	Must have	
4	Leader boards	As a user, I expect to be able to view a leader board of socks which have been lost the longest	Nice to have	
5	Suggestions	As a user, I expect that the application will make suggestions of when socks should be thrown out	Nice to have	

USER INTERACTION AND DESIGN

QUESTIONS

Below is a list of questions to be addressed as a result of this requirements document:

Question	Outcome

NOT DOING

Washing Rain Detector

Target release	
Epic	
Document status	DRAFT

Document owner	Lachlan Gabb
Designer	
Developers	
QA	

THE PROBLEM

You have cloths on the line drying when it starts to rain. As you are not outside, it is often sometime before you notice the rain and go to bring the cloths in. By that time, the cloths are already significantly dampened by the rain. By making a device that could provide notification when the rain is detected, it should be possible to bring them in before they are soaked.

GOALS

- Detect when rain is falling above a cloths line.
- Provide notification to the owner to take their cloths inside.
- Provide advanced warning using other devices within a similar geographic area.
- Interface with a weather API to predict rain times, alerting the user that it is likely to rain

BACKGROUND AND STRATEGIC FIT

Create a device that detects when it is starting to rain above the cloths line and sends a notification to the owner to bring their washing in. Could integrate them together to provide advanced warning when other devices within the same area detect rain.

ASSUMPTIONS

- An Internet connection will be available for sending the notifications.
- A permanent source of electricity will be available at the cloths line.

REQUIREMENTS

#	Title	User Story	Importance	Notes
1	Rain Detection	As a user, I want to be notified when it rains so I can promptly bring in the washing	Must have	
2	Experience Sharing	As a user, I want to know when other users around my area have detected rain	Must have	
3	Dashboard	As a user, I want a mobile friendly dashboard to view the required information	Must have	

USER INTERACTION AND DESIGN

QUESTIONS

Below is a list of questions to be addressed as a result of this requirements document:

Question	Outcome

NOT DOING

DAVID PITCH

Problem: Noticing it has been raining for some time and all of the washing outside is wet.

Project Idea: Create a rain sensor that would be placed with the washing and send a notification when rain was detected. Additionally, if multiple devices were places around a neighbourhood, they could send rain information back to a central source. Using the pooled rain data, the system could provide advanced warning of approaching rain.

The project aims to solve the problem of realising it has been raining for some time and all of the washing on the line is not wet. The project would create a rain sensor that could be placed with the washing which would send a notification to a smartphone indicating it has detected rain. Additionally, if multiple devices were deployed throughout a neighbourhood they could provide rain information back to a central source. Using the pooled rain information, it would be possible to provide people advanced warning when rain is approaching.

POSSIBLE QUESTIONS

- What sensor will we use
    - This one for now https://www.instructables.com/id/Arduino-Modules-Rain-Sensor/

Doorbell for the Blind

Target release	
Epic	
Document status	DRAFT
Document owner	Lachlan Gabb
Designer	
Developers	
QA	

THE PROBLEM

My great aunt, who has little to no sight, is sometimes left at home on her own. If someone arrives at the door, she will often have to open it in order to determine who it is. By opening the door, before authentication, she may be placing herself and the house at risk. By creating a device that could allow her to identify who is at the door without having to open it, the security of the home would be protected.

GOALS

- Use facial recognition to identify a set of pre-defined people arriving at the door.
- Act as a doorbell and speak the name of the person it has identified to the user who resides in the home.
- Provide notification to the user if the person at the door cannot be identified.
- Take a photo of the unidentified person and send it along with a message to a nominated contact, who can take the product from there.

BACKGROUND AND STRATEGIC FIT

Create a device that can use facial recognition to identify a set of know people arriving at the home of someone with limited sight. Upon recognising the person, it will act like a doorbell, and say the name of the person. This allows the vision impaired individual to determine who is at the door before they open it. Additionally, if the person at the door cannot be identified, it will notify them of the fact and also send a message to a nominated point of contact with a photo.

ASSUMPTIONS

- All regular and necessary visitors to the home will have their face scanned and loaded into the device.
- There will be an Internet connection available to send the notification messages to the nominated point of contact.
- Adequate warning will be provided to visitors of the home that their face is going to be scanned for identification purposes.

REQUIREMENTS

#	Title	User Story	Importance	Notes
1				
2				

USER INTERACTION AND DESIGN

QUESTIONS

Below is a list of questions to be addressed as a result of this requirements document:

Question	Outcome

NOT DOING

DAVID PITCH

Problem: My aunt, who has very little sight, is often left along at home and answers the door if someone rings the bell. Due to her limited site, will often only identify the person after opening the door, presenting a security risk.

Project Idea: Create a doorbell which could identify pre-programmed individuals at the door and speak their names to my aunt. The doorbell could use either facial recognition or a PIN number or even a combination of the two, depending on what identification accuracy we could obtain. Additionally, if the person could not be identified, it could send a notification to another family member, indicating an unknown person is at the door. The notification could even include a picture.

POSSIBLE QUESTIONS

- What sensors
    - Raspberry Pi camera module
    - Google
- How to perform facial recognition
    - Looking at trying to use the Amazon Rekognition module
    - Could also use a Python library if anyone can find an appropriate one
- What if the facial recognition doesn't work/internet is not available
    - Fall back to a pin code entered on a pin pad
    - Everyone will have a unique pin code, ensuirng they are identified correctly
- Wouldn't an intercom be just as good
    - No as anyone could claim to be a person who is known to the blind person
    - It is far less likely they are able to authenticate correctly
    - Additionally, an intercom would not have the ability to send a notification to
- Privacy Concerns
    - Say something about securing the box
    - If we use the PIN code identification, there is little to no risk
    - There would need to be clear signage indicating you concent to having your picture taken when entering the property
- Notification System
    - Microsoft flow API

Practical Assessment

Table Of Contents

Components Overview

The project has been developed with the aim to dispense the right amount of medication for an user at a certain period of time. It has been designed to hold three different types of medications in three different compartments and dispenses pills according to the program. The major components to get the mechanical dispenser work include:

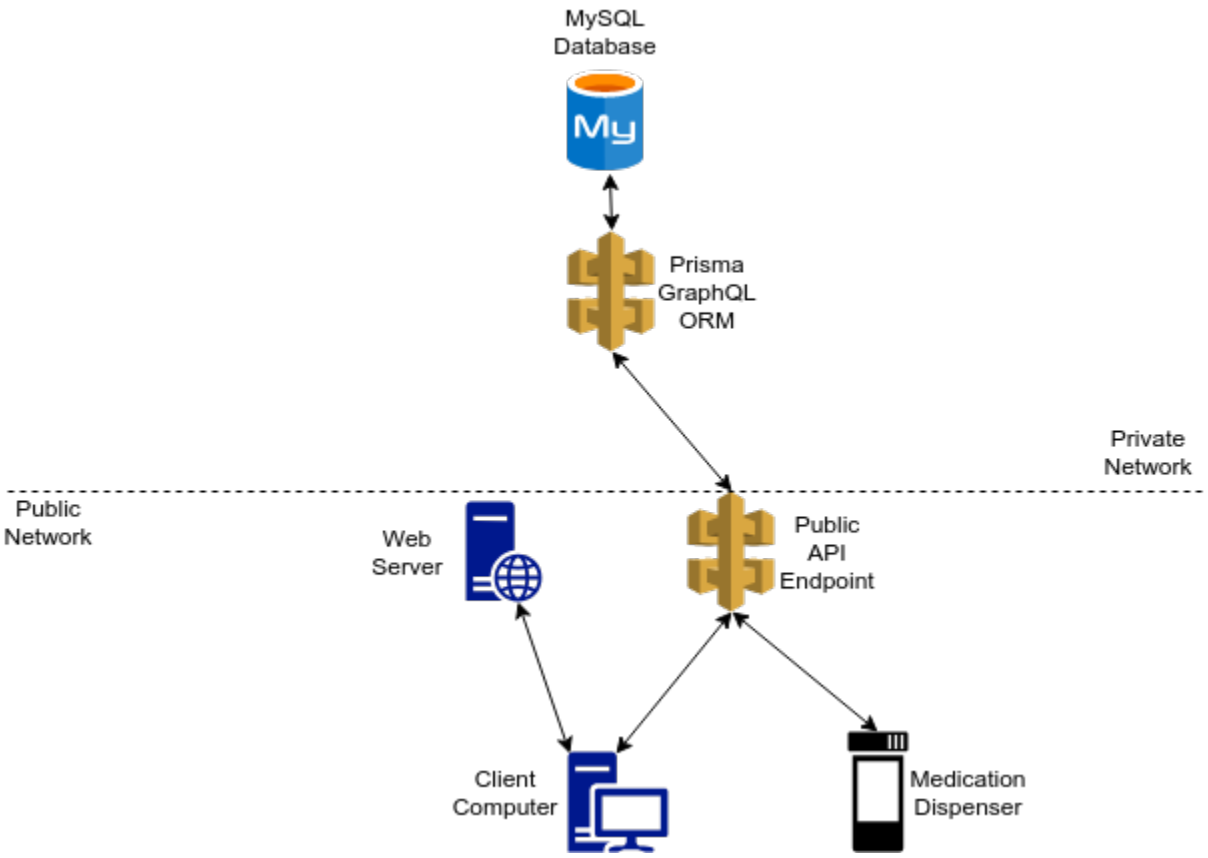
- Jaycar Electric Motor (YG 2734)
- Jaycar Infrared Phototransister
- Jarcar Photosensitive Resistor
- Raspberry Pi
- Bunnings 100mm White Plastic Wheel
- Bunnings 110mm White Plastic Water Pipe
- Jaycar Variable Switchmode Power Supply

- GraphQL API
- MySQL Database
- Interactive web application

API

To allow the dispenser and it's users to interact with that same database, an API was created. This project uses a GraphQL API to expose various database functions such as querying, updating and deleting. The architecture of the API and frontend is shown in the following diagram.

ARCHITECTURE DIAGRAM



HIGH-LEVEL DESCRIPTION OF COMPONENTS

Each component of the API diagram is explained below at a high level.

Client Computer

- Any device used to access the magic meds web application.
- Requires internet access and a web browser

Medication Dispenser

- The device made to dispense pills in the magic meds project.
- Requires internet access

Web Server

- The web server is responsible for handling requests for the web application and returning the web application to users.

Public API Endpoint

- Defines the data models and associated values, for example, a medication has a name (which is a string) and a count (which is an integer).
- Defines the publicly exposed API schema, e.g. what are application/users able to query.
- Business logic is also defined at this layer, such as authentication, filtering, etc.

Prisma GraphQL ORM

- Takes the data models defined by the Public API Endpoint and creates the corresponding tables in the MySQL Database.

- Takes the models and defines all possible actions which could be performed on each data (CreateReadUpdateDeleteSubsribe)
- Takes incoming GraphQL queries from the Public API Endpoint, and translates them into the corresponding MySQL queries.

MySQL Database

- This is where the actual data is stored.

An example communication process is as follows:

1. The client computer sends an HTTP GET request to the web server.
2. The web server sends the React based web application in a JS file.
3. The client loads the application in the browser.
4. Through the web application, the user queries the public API endpoint to get a list of medications belonging to the logged in user.
5. The public API endpoint then reaches out to the Prisma GraphQL ORM, which executes the required SQL queries on the MySQL database. The Prisma server then returns the information to the public API endpoint, which in turn, returns the information requested to the web application.
6. The medication dispenser interacts with the API in the same fashion.

Tablet Dispenser Module

- The dispenser module contains a wheel with tablet sized holes cut out of it
- The wheel is housed in 110mm pipe, which also stores the tablets
- The wheel is connected to one of the electric motor, which turns the wheel when power is applied
- As the wheel turns, tablets are dispensed.

Tablet Sensor Module

The tablet detector sensor is used to detect when the dispenser successfully drops a tablet. The steps below describes how the sensor functions :

- The sensor is located in part of the pipe through which the tablet much travel
- The sensor consists of an infrared LED and a light sensitive resistor.
    - When the pipe is clear, the light from the LED shines into the resistor, causing it to be open circuit.
    - As a result no power flow through the circuit.
- However, when a tablet passess through the pipe, it momentarily block the light from the LED reaching the light sensitive resistor.
    - When this occurs, the resistor becomes a short circuit and allows power to flow through the circuit.
    - This causes the output of the circuit to go from 0v to 3.3v which is then detected by the Raspberry Pi.
    - The output of the tablet detector circuit is connected directly to the GPIO pins of the Raspberry Pi.

Refer to Appendix section A for the physical, the electrical circuit design of the tablet detection sensor and the basic detection code.

Motor Controller Circuit

The motor controlling circuit is designed to allow the comparatively small 3.3 volt output of the Pi to power the electric motors, which require 8 volts.

- The Raspberry Pi is connected to a number of transistors, one for each of the motors it intends to control.
    - The Raspberry Pi can turn these transistors on with its 3.3v output.
- The transistors alone are not capable of switching the power for the motor, so they are in turn connected to three Mosfets.
    - When the transistors are turned on by the Pi, they provide 8 volts to the Mosfets causing them to conduct power.
- Finally, the Mosfets are connected to the electric motors.

Please see appendix B for an overview of the motor controlling circuit.

Overall Operation Process

- Scanning of RFID card
    - When the RFID is scanned , it reads the user's unique identifier from the card and loads it into the Python script
- Retrieval of user data
    - The Raspberry Pi then queries the API using the user's unique ID and retrieves the list of medications and schedules associated with that user.
- Scheduling
    - Once the information is retrieved, it checks if any medication is due to be taken at that time.
    - Additionally, it also checks the user has not already taken their medication.
- Dispense
    - The Raspberry Pi uses the information associated with the schedules to turn on the appropriate GPIO pins to start the dispenser

motors.

- Counting
  - The Raspberry Pi uses the tablet detection sensors to determine when the appropriate amount of each medication has been dispensed.
  - Once this occurs, power to the GPIO pins is turned back off, stopping the dispensing.
- Update
  - Finally, the Raspberry Pi uses the API to update the database on the current status of user's medication.
  - It also adds a timestamp of the last time the medication was taken to prevent the ability for them to take medication twice.

Please see Apendix C for the operational code.

Power Delivery Circuit

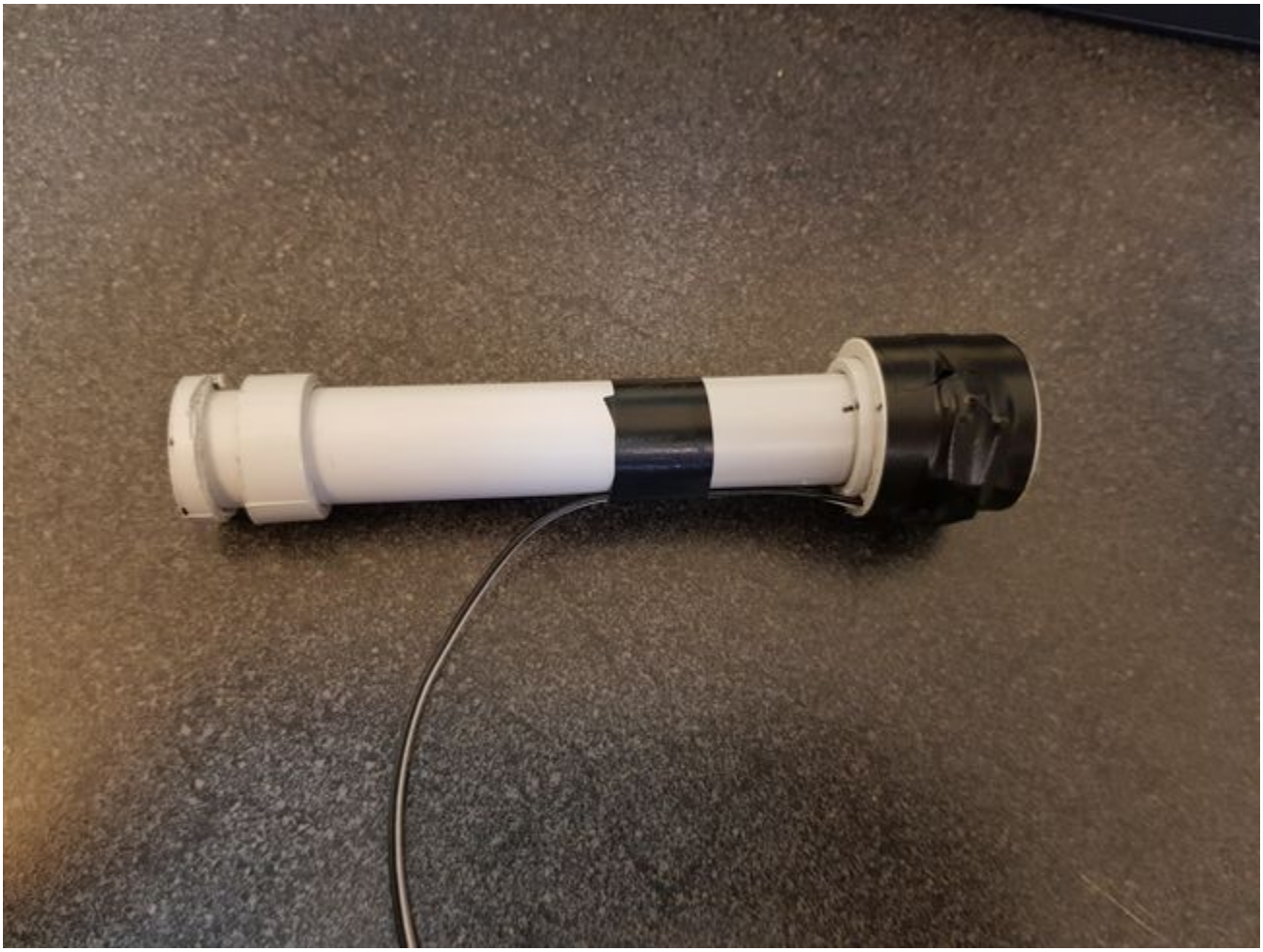
To operate the various different components required for the project, it is required to have multiple voltages present in the dispenser. The following steps have been taken to achieve this:

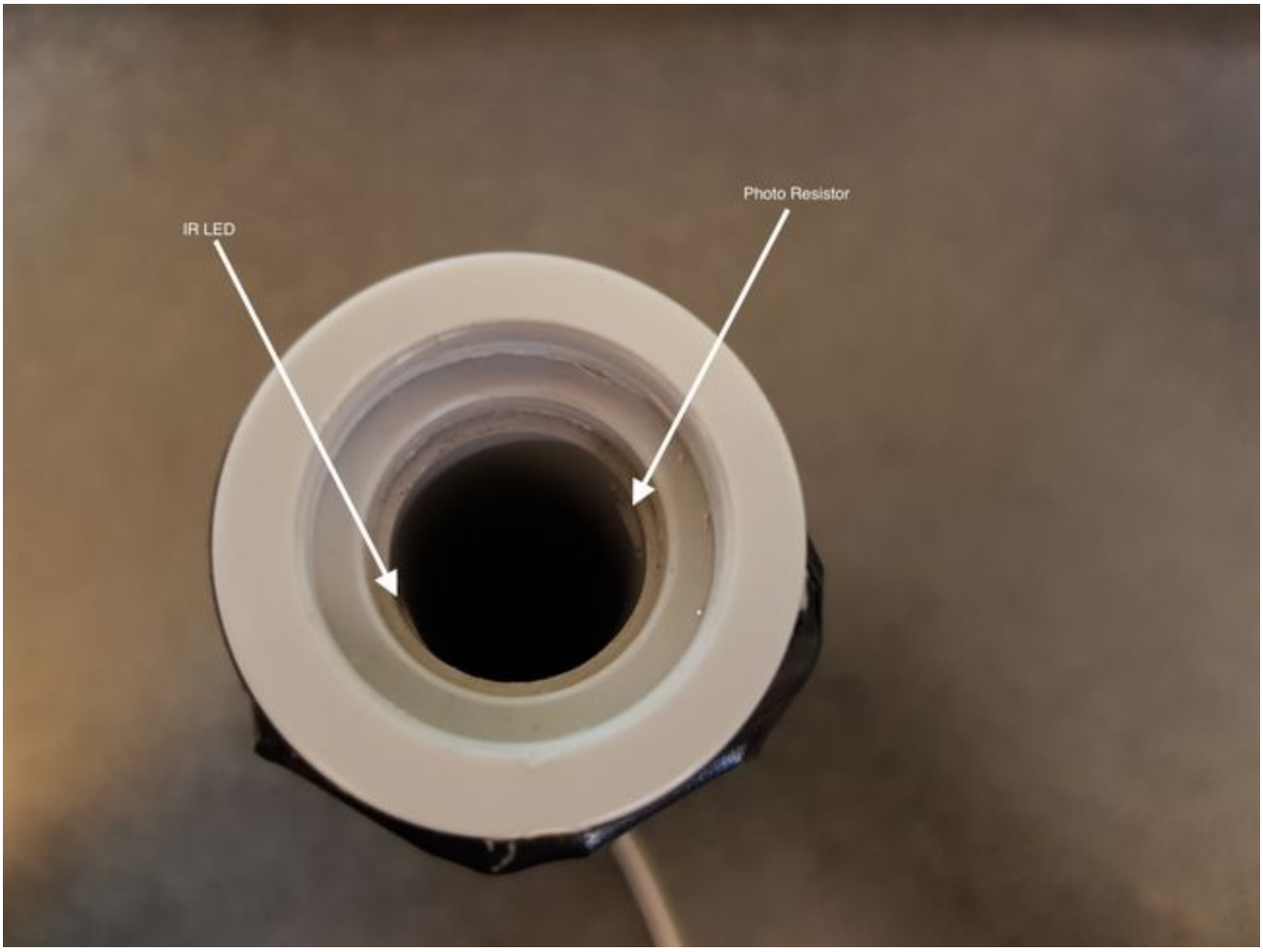
- A single 12 bolt power supply is connected to the dispenser unit
- The power is split and connected to two variable switchmode power supplies
- Each power supply is configured to output different voltages
  - One outputs 5 volts to operate the Raspberry Pi
  - The other outputs 8 volts to operate the Mosfets and electric motors

Please see Appendix B for the circuit overview of the power delivery system.

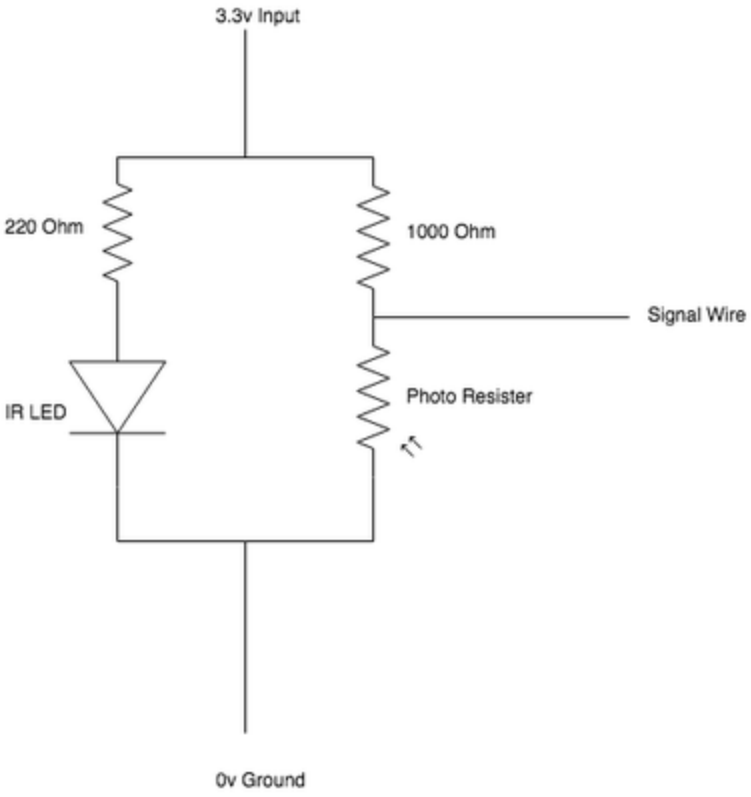
Appendix A

Physical Dispenser





Electrical Design

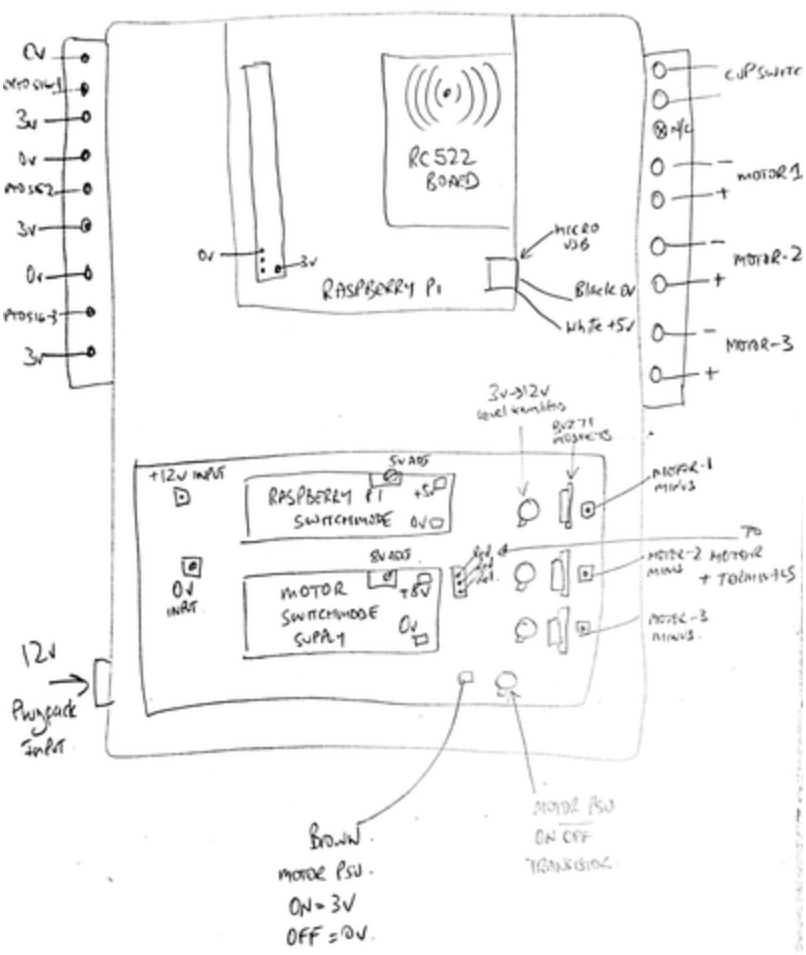


Basic Tablet Detection Code

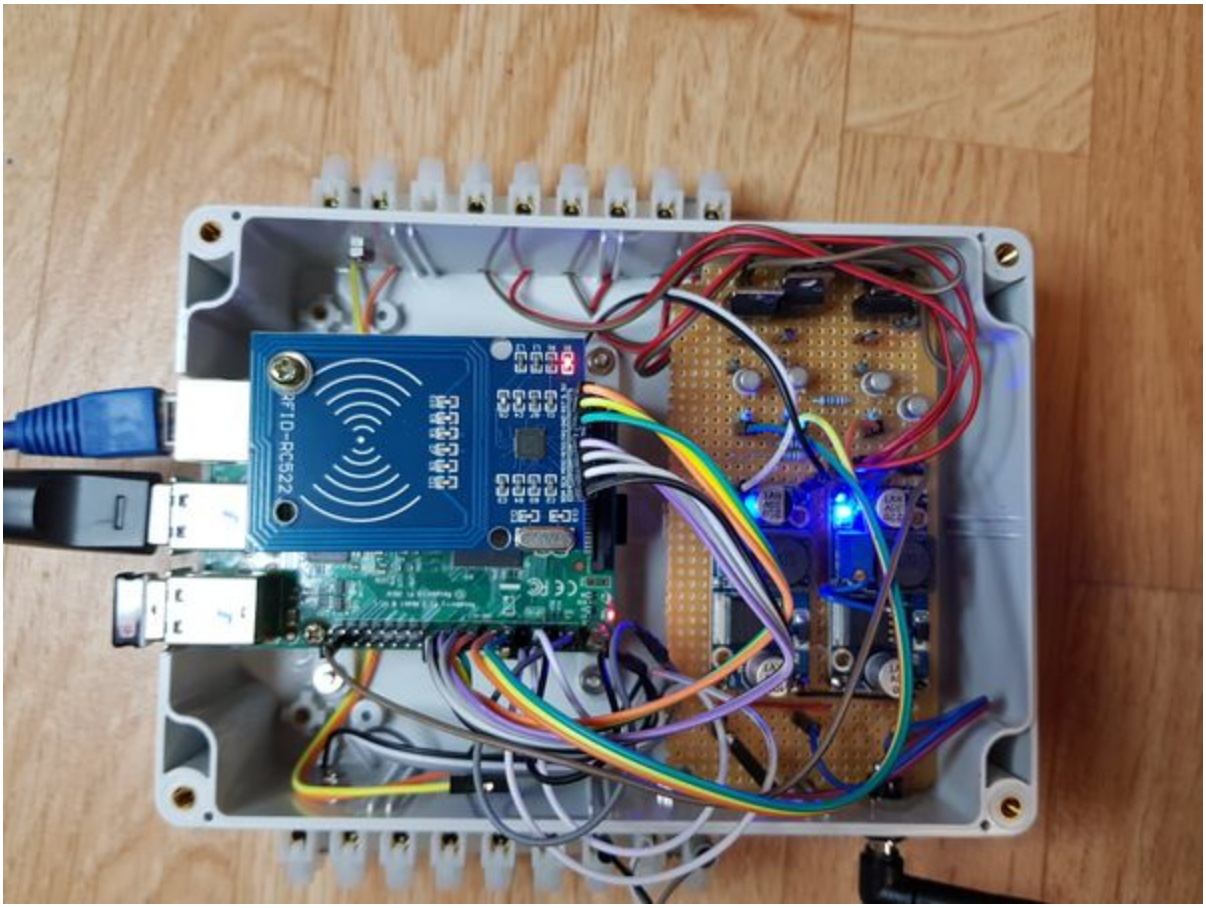
```
PinNumber = 7 # Define the pin on which the Pill Sensor will be detected

import RPi.GPIO as GPIO # Import the GPIO for the Raspberry Pi
import time # Import the time code

GPIO.setmode(GPIO.BOARD) # Setup Radspberry Pi pins as board numbering
GPIO.setup(PinNumber, GPIO.IN) # Setup the selected pin to be an input

Count = 0 # Used for counting

while True: # Just Go
    if GPIO.input(PinNumber) == True: # Check if the PIN has gone high
        time.sleep(0.04) # If it has, wait aproximately 30 milliseconds for the pill to pass and check it
has
        if GPIO.input(PinNumber) == False: # Check the pill has now passed the sensor
            PillDropped = 1 # If it has, a pill has been dropped
            Count = Count + 1
            print("You have now dropped " + str(Count) + " pills.")

        else: # If it has not, something is likely blocking the sensor
            print("Something appears to be stuck")
            time.sleep(1) # Sleep for a moment to see if the blockage clears
    time.sleep(0.001) # Sleep for one millisecond between checks
```

Appendix B

Circuit Overview



Physical Control Box



Appendix C

The following is the main Python script that links all functions of the dispenser together

```
# MainMed.py - Major Project Lachlan, Daniel, Fariha, Terry
# Used to control the dropping of tablets from the dispenser
import RPi.GPIO as GPIO # Import the GPIO for the Raspberry Pi
import time # Import the time code
import SimpleMFRC522 # Import the card reader code (For the RFID reader)
import requests
import json
import datetime
GPIO.setmode(GPIO.BCM) # Setup Radspberry Pi pins as board numbering
reader = SimpleMFRC522.SimpleMFRC522() # Setup the Reader Object

# ### Define the GPIO Pins in Use ### Please note: These are the
physical board pin numbers
BootPin = 21

MotorOnePin = 17 # 11 Physical
MotorTwoPin = 18 # 12 Physical
MotorThreePin = 27 # 13 Physical

DetectorOnePin = 4 # 7 Physical
```

```
DetectorTwoPin = 14 # 8 Physical
DetectorThreePin = 15 # 10 Physical

DispenserOne = [1, MotorOnePin, DetectorOnePin] # Creates an array so
all dispenser information is in one location
DispenserTwo = [2, MotorTwoPin, DetectorTwoPin]
DispenserThree = [3, MotorThreePin, DetectorThreePin]

DispenserArray = [DispenserOne, DispenserTwo, DispenserThree]

# print(DispenserArray)

# ### Setup the GPIO Pins for Use ###
# # This defines whether the pins are going to be used for input or
output
GPIO.setup(BootPin, GPIO.OUT)
GPIO.setup(MotorOnePin, GPIO.OUT) # Setup the motor pins to be output
GPIO.setup(MotorTwoPin, GPIO.OUT)
GPIO.setup(MotorThreePin, GPIO.OUT)

GPIO.setup(DetectorOnePin, GPIO.IN) # Setup the sensor pins to be input
GPIO.setup(DetectorTwoPin, GPIO.IN)
GPIO.setup(DetectorThreePin, GPIO.IN)

# # As a 1 means stop for the board, all of the pins must be set to high
before system starts
GPIO.output(MotorOnePin, GPIO.HIGH)
GPIO.output(MotorTwoPin, GPIO.HIGH)
GPIO.output(MotorThreePin, GPIO.HIGH)
GPIO.output(BootPin, GPIO.HIGH) # Set to high to inform the motor driver
board that the script is ready
print("The safety pin has been removed")

APIURL = "http://meds.thegabbs.com:4000"

UPDATE_MEDICATION_QUERY = """
mutation updatePatientMedication($id: ID!, $name: String!, $count:Int!,
$patientId: ID!){
  updatePatientMedication(id: $id, name: $name, count: $count,
patientId:$patientId){
    id
    name
    count
  }
}
"""

USER_SCHEDULES_QUERY = """
query userSchedules($id: ID!){
    userSchedules(id: $id){
```

```
      id
      time
      takenTime
      medications {
        id
        name
        count
        dose
        dispenser
      }
    }
  }
"""

UPDATE_SCHEDULE_MUTATION = """
mutation updatePatientScheduleMutation($id: ID!, $time: Int,
$takenTime:DateTime, $patientId: ID!){
  updatePatientSchedule(id: $id, time:$time, takenTime:$takenTime,
patientId:$patientId){
    id
    time
    takenTime
  }
}
"""

HEADER = {"Authorization":"Bearer
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VySWQiOiJjam12bGplNGYwMDhsMDk
4OG00bjA4eW5kIiwiaWF0IjoxNTM4NzE5MTQxfQ.eF6a36uy_D6eMz3XhD_lTd-4NHzyRnwA
yajtNnHT9l0"}

def DetectPill(DetectorNumber): # Used to detect when a pill has been
dropped. Needs to be told which sensor to look at
    while True: # Just Go
        if GPIO.input(DetectorNumber) == True: # Check if the PIN has
gone high
            time.sleep(0.04) # If it has, wait aproximately 30
milliseconds for the pill to pass and check it has
            if GPIO.input(DetectorNumber) == False: # Check the pill has
now passed the sensor
                return True

            else: # If it has not, something is likely blocking the
sensor
                print("A pill may be stuck")
                time.sleep(1) # Sleep for a moment to see if the
blockage clears
        time.sleep(0.001) # Sleep for one millisecond between checks

def Dispense(DispenserNumber, Quantity): # Used to actually dispense
```

```
Pills. Provide the dispenser number and the amount of tablets that needs
to be dispensed
    for Tablet in range(0, Quantity): # Start from 0 and increment for
each tablet
        GPIO.output(DispenserNumber[1], GPIO.LOW) # Turn the motor on
        print("Motor " + str(DispenserNumber[0]) + " is running")
        while DetectPill(DispenserNumber[2]) != True: # Contiunue until
a pill has been dropped
            pass # Used to setup a POST test loop
    GPIO.output(DispenserNumber[1], GPIO.HIGH) # Executes once a pill
has been dropped and turns the motor back off
    print("Motor Has Stopped")

def QueryAPI(query='', variables={}, url='', headers={}):
    """
    Make query response
    """
    request = requests.post(url, json={'query': query, 'variables':
variables}, headers=headers) # Sends a post request to the API with the
necessary parameters.
    if request.status_code == 200: # If the request is returned
successfully
        #print(request.json())
        return request.json() # Return the reqest in JSON.
    else:
        raise Exception("Query failed to run by returning code of {}.
{}".format(request.status_code, query)) # Inform the console why the
request was not returned successfully.


while True: # Main Loop
    try: # Try and read the user ID from the RFID card.
        id, UserID = reader.read() # Read the information from the card
        print(UserID)
    except: # If the user ID cannot be read from the card.
        print("Could not read, something must be wrong")
    #UserID = 'cjn5jlmju01l708399k3qmtxs'


    QueryResult = QueryAPI(query=USER_SCHEDULES_QUERY,
variables={'id':UserID}, url=APIURL) # Query the API for the above
user's medication.
    #print(QueryResult)

    SchedulesList = QueryResult["data"]["userSchedules"] # Get a list of
the different schedules returned by the API

    for Schedule in SchedulesList: # For each schedule in the list
        #print(Schedule)
        LastTaken = Schedule["takenTime"] # Determine the time the
```

```
schedule's medication was last taken.

        CurrentTime = datetime.datetime.now() # Get the current time
        CurrentDate = str(datetime.datetime.now().date()) +
"T00:00:00.000Z" # Create the current date string.


        print(LastTaken)
        print(CurrentDate)

        if Schedule["time"] == CurrentTime.hour: # and CurrentDate !=
LastTaken: # If the scheule is due to be taken within the hour and has
not been previously taken today.
            print("True")

            for Medication in Schedule["medications"]: # For each of the
medications within the schedule.
                #print(Medication)
                Dispenser = Medication["dispenser"] # Extract the
dispenser number associated with the particiular medication.
                Dose = Medication["dose"] # Extract the dose of each
medication.
                Medication['patientId'] = UserID
                #print(Medication)
                for Item in DispenserArray: #Extract the necessary
dispenser to motor and sensor mapping information from the arrays
defined at the top.
                    if Item[0] == Dispenser: # Contiune until it finds
the correct dinspenser mapping.
                        Dispense(Item, Dose) # Actually dispense the
medication.
                        Medication['count'] = Medication['count'] - Dose
# Remove one from the total count of medications available.
                        #print("The medication was just updated")
                        QueryAPI(query=UPDATE_MEDICATION_QUERY,
variables=Medication, url=APIURL, headers=HEADER) # Update the API with
the number of tablets now available.
                        #print(Medication)


            # Update the API with the time the medication was taken.
            Schedule["takenTime"] = str(datetime.datetime.now().date())
# Set the takenTime to be the current date.
            #print(Schedule["takenTime"])
            Schedule.pop("medications", 0) # Remove the medications from
the dictionary. Needed to make the API call work.
            Schedule["patientId"] = str(UserID) # Add the user ID for
who the schedule needs to be updated.
            #print(Schedule)
```

```
            QueryAPI(query=UPDATE_SCHEDULE_MUTATION, variables=Schedule
, url=APIURL, headers=HEADER) # Actually push the updates to the server.


        else: # If it is not time for the medication to be taken
            print("False")  # Do Nothing.

time.sleep(2)
```