Pill^3

Senior Design Project: Group 16

DESIGN GOAL

“Create a device to aid people who forget to take their medicine”

- Simple to set up and use
- Have IOT connectivity
- An affordable option

Our Team

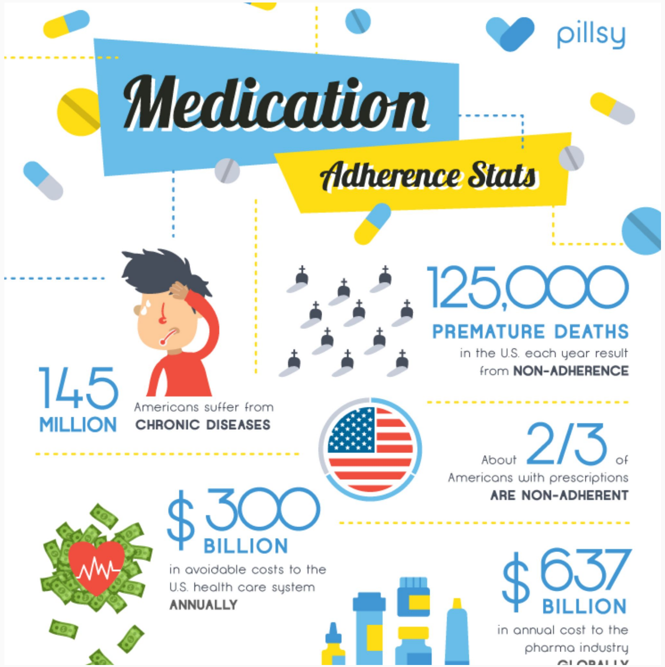
Liam Kenney

Jordan Schneider

Oriana Alcala

Fernando A. Oriundo

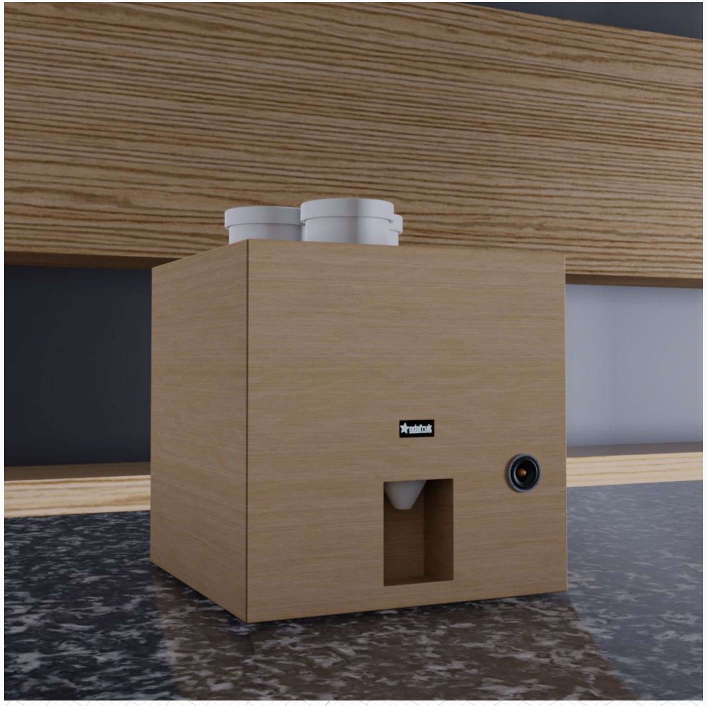
PROJECT MOTIVATION

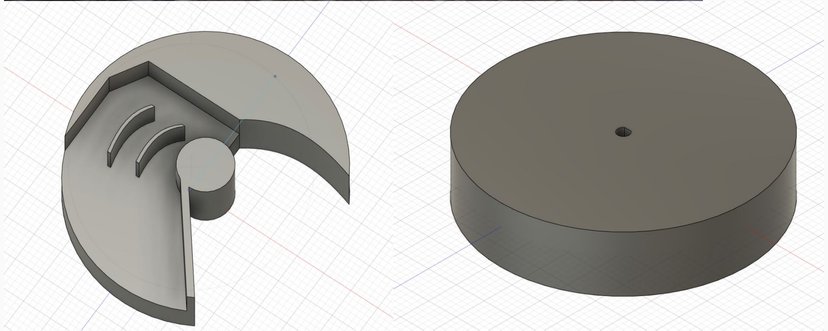

pillsy
Medication
Adherence Stats
125,000
PREMATURE DEATHS
in the U.S. each year result from NON-ADHERENCE
145 MILLION
Americans suffer from CHRONIC DISEASES
About 2/3 of Americans with prescriptions ARE NON-ADHERENT
$300 BILLION
in avoidable costs to the U.S. health care system ANNUALLY
$637 BILLION
in annual cost to the pharma industry
GLOBALLY

Prescription non-adherence, both intentional and unintentional, is one of the biggest issues within the American medical community.

- Contributes thousands of tons to medical waste
- Results in additional spending on medication or procedures
- May lead to otherwise avoidable life-threatening illness and injury

The issue of non-adherence has affected the daily lives of all of our group members in different ways.





Initial Overall Design

The original Pill^3 dispenser design consisted of three main subsystems.

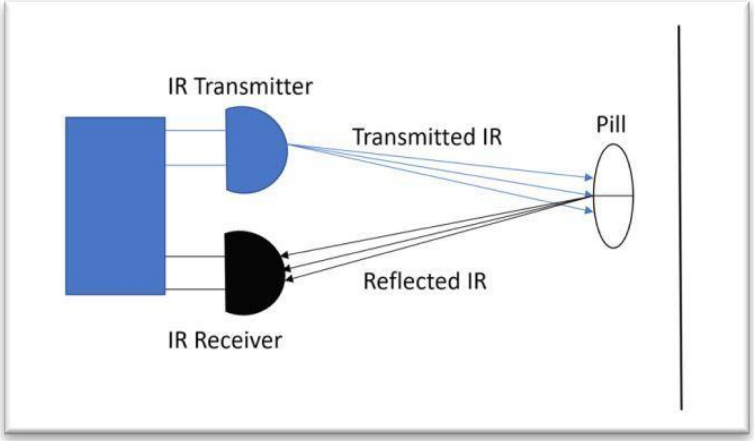
- First Subsystem: Pill storage chambers
  - Pill tracking
  - Pill distribution
  - Size sorting
- Second Subsystem: Device status sensors
  - Temperature
  - Clock

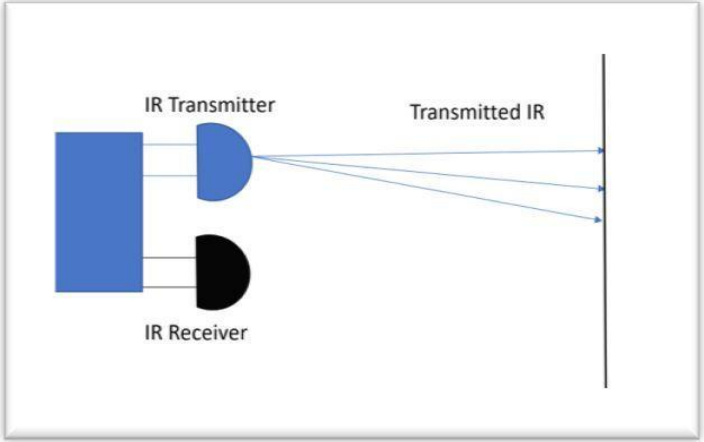
Third Subsystem: User notifications

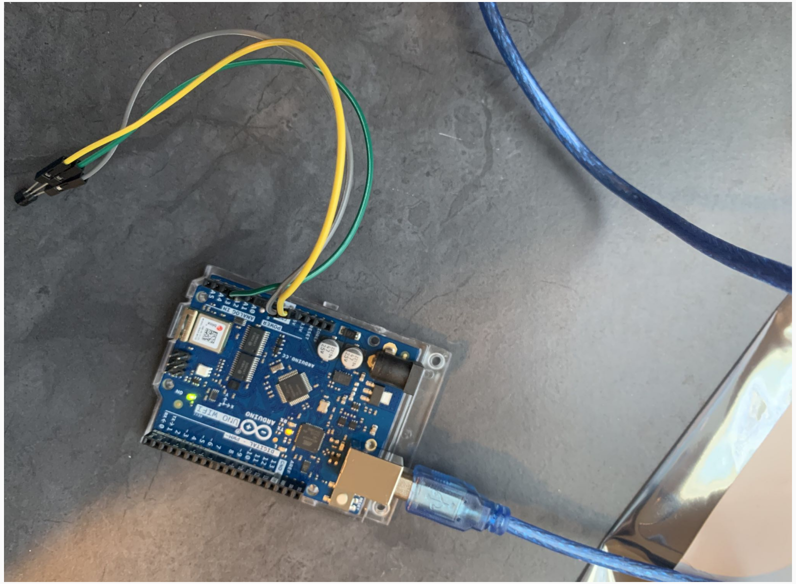
  - Audio
  - Visual
  - Smartphone App

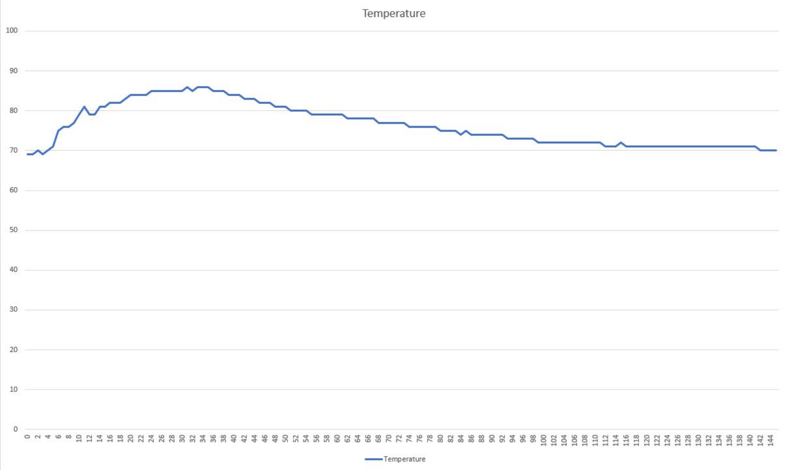
IR SENSORS

Simple infrared emitter and receiver sensor packages were used to measure object movement. Through this, we were able to track when pills passed a certain point on the distribution line and subtract them from the total stored amount. Using this method, one sensor would be able to detect pills of any size using the same metric.









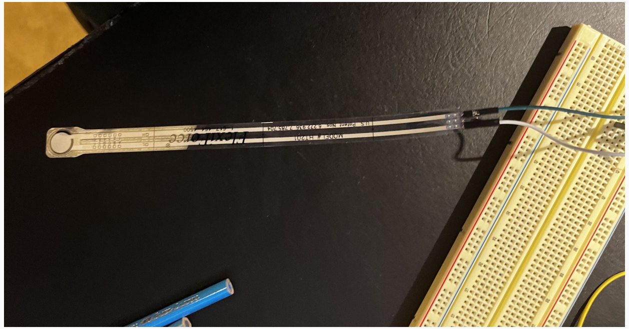
TEMPERATURE SENSOR

A LMT85LP temperature sensor was used to ensure that the device is operating at safe temperature levels. The overall temperature of the device was almost measured to ensure that the medicine was stored at an appropriate level.

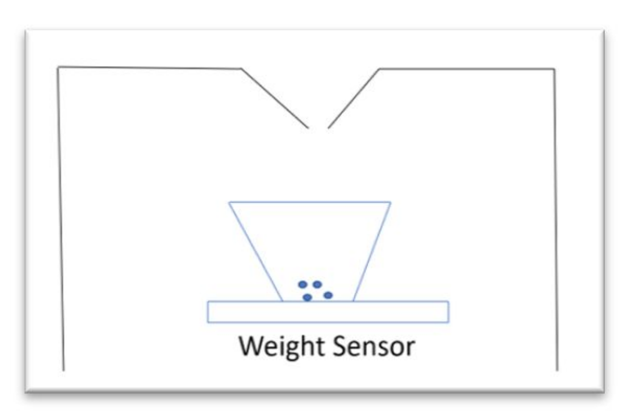
This sensor acts as a variable resistor through which changes in voltage can be linked to changes in temperature. From there the information can be applied to either maintain the temperature or alert the user to a potential issue.

PRESSURE SENSOR

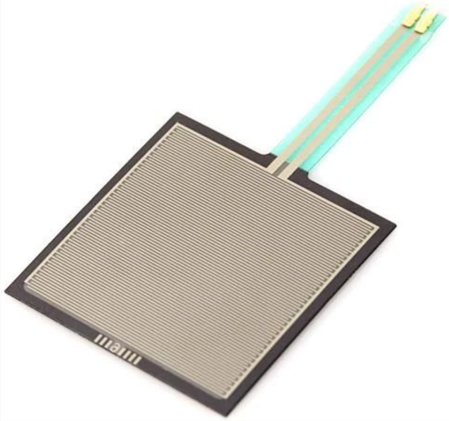
The Flexiforce pressure sensor was used to determine if there is a container positioned below the dispenser for the medicine to be deposited into. This helps to prevent the device from dispensing too many pills or spilling the dispensed pills onto a random surface.



The sensor is adjustable to work with any common household dish or cup, although it does come with a specified receptacle.

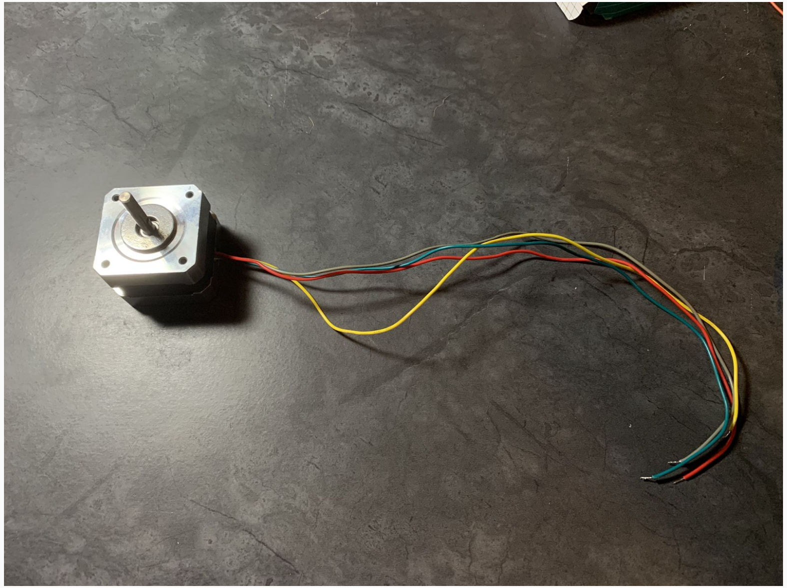


Due to the small contact area of the Flexiforce sensor and it’s low sensitivity, we decided to instead use the SEN-09376, which offered a larger sensing area and the precision necessary to detect cups weighing as low as 15 grams.



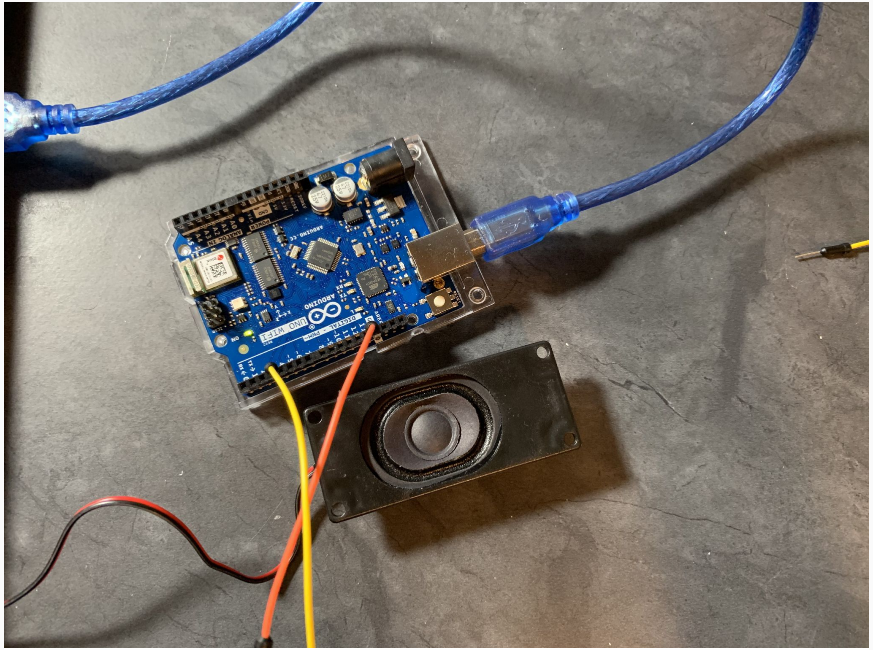
STEPPER MOTORS

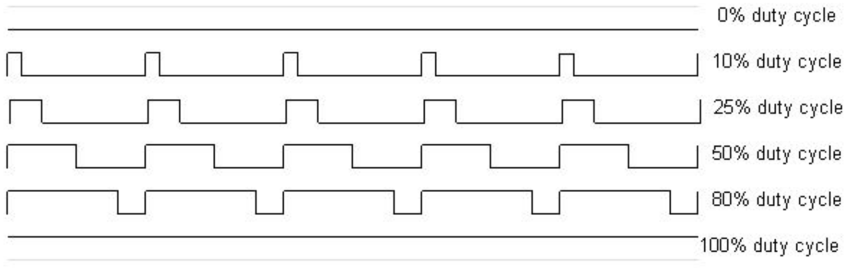
For what we wanted to achieve, a stepper motor was the most sensible choice as it had a full range of rotation and could start and stop more accurately than DC motors of the same power requirements. This specific adafruit stepper motor was chosen as it combines the right amount of precision and strength needed to control the pill distribution device.



Adafruit NEMA-17 stepper motor specifications

- 0.2 Nm of torque
- 360 degrees of rotation
- 1.8 degrees of accuracy





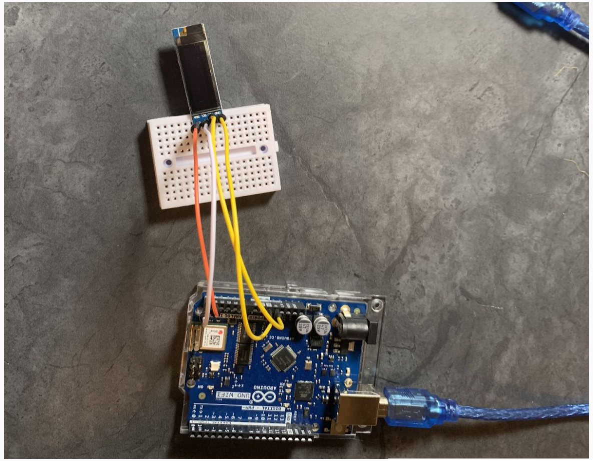
SPEAKER

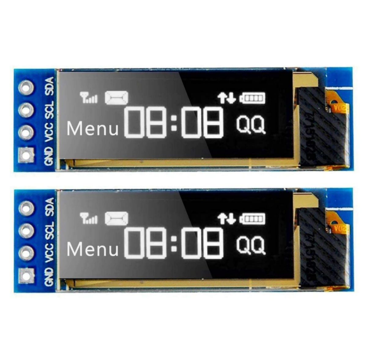
The speaker used was simple to implement and customize, and allowed for the user to have auditory alerts that can be heard from several rooms away. We decided to forego an audio driver in favor of driving the speaker with a pulse width modulator voltage. Changes in voltage and duty cycle translate to changes in pitch and volume of the tone produced

DISPLAY

The tertiary user alert system implemented into the Pill^3 was a 0.91” OLED screen. It can both indicate that a prescription is ready and display information about the system at a glance without needing to communicate with the smartphone application. It will typically display general system and environmental information like the time and temperature.

It cycles between an idle screen displaying a minimalistic Pill^3 logo and the current time and temperature of the device. It also instructs the user to place a cup onto the weight sensor to being dispensing the pills.

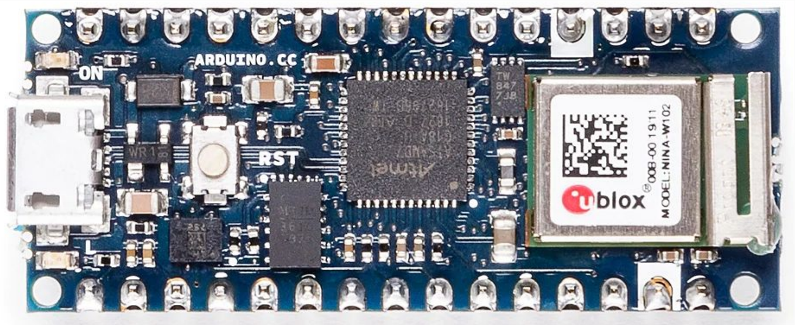




MICROCONTROLLER

The Pill[3] contains an Arduino Nano Iot 33. This MCU has a total of 30 pins and is able to connect to the established app through wifi. Due to this, the user is able to send information of their pill schedule as well as the quantity of pills to be stored in the Pill[3] through Firestore to the MCU.

Once the instruction has been sent, the motors will active and the MCU will be waiting for the IR sensor to detect if the pill left the container. If the pill successfully left the container, the user will be notified that the pill is in the cup. The MCU is able to do all this as long as the cup has been placed in the corresponding area.



POWER SUPPLY

When comparing different types of power sources we can see that AC power from a wall outlet…

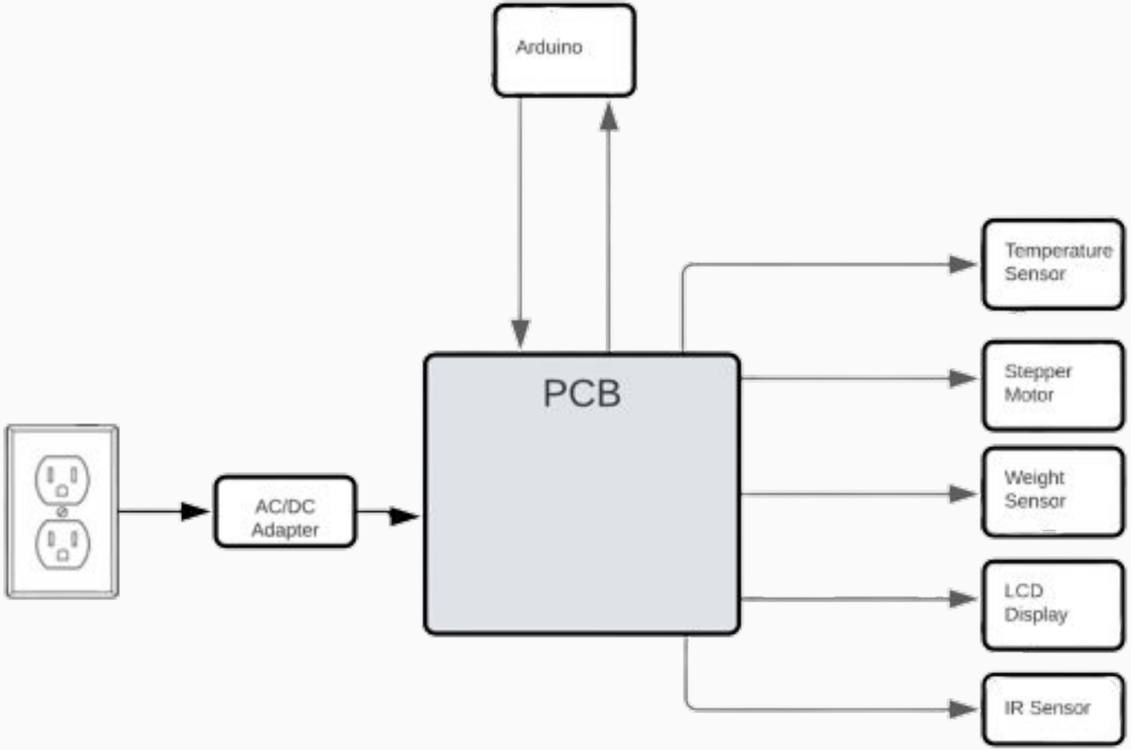
- Long lifecycle
- Great energy storage
- Low cost
- Versatility
- Does not need replacement
- Is more reliable than solar energy
- Provides constant power
- Provides more interior room in the design of the device

However…
For our device to function, we need to have an outlet available.
Cords can also cause unneeded messes to occur

POWER DISTRIBUTION

Hardware	Voltage Requirement	Current Requirement
Arduino Iot 33	7 V - 21 V	7 mA per I/O pin
Adafruit NEMA-17 Motor	12 V	0.35 A
Obstacle avoidance IR Sensor	3.3 V - 5 V	20 mA
Pressure Sensor	5 V	1 mA
MakerFocus Display	3.3 V - 5 V	430 uA
LMT85LP Temp. Sensor	3.3 V - 5 V	8.1 uA
Controller	1V - 36V	2A

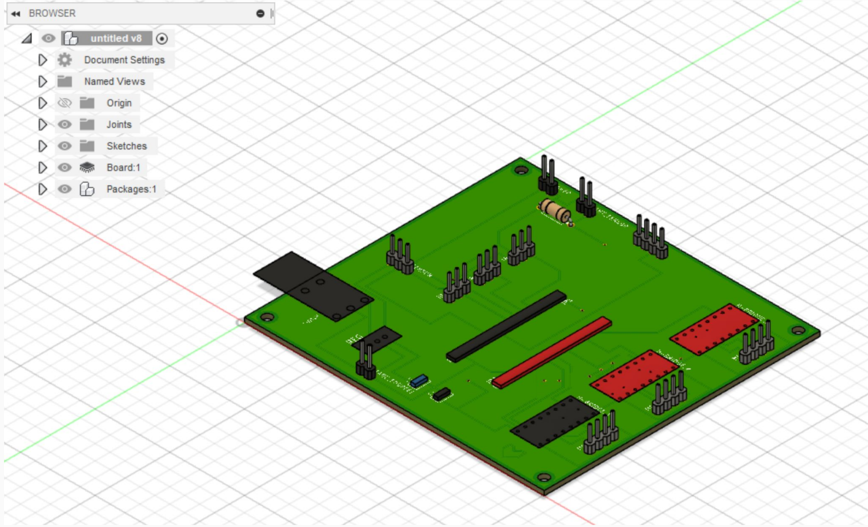
OUR DESIGN


Arduino
PCB
AC/DC Adapter
Temperature Sensor
Stepper Motor
Weight Sensor
LCD Display
IR Sensor

PCB DESIGN

During the overall design of the Pill^3 certain components were incorporated in the PCB board that will be used for the project. Throughout the development of the board certain decisions were made:

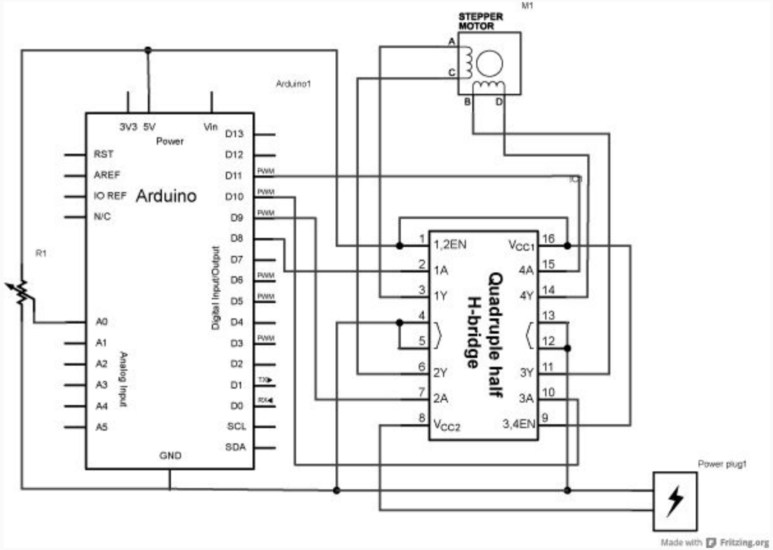
- Software
- Components
- Vendors
- MCU
- Voltage Regulators
  - Efficiency vs Functionality



PCB DESIGN CHALLENGES

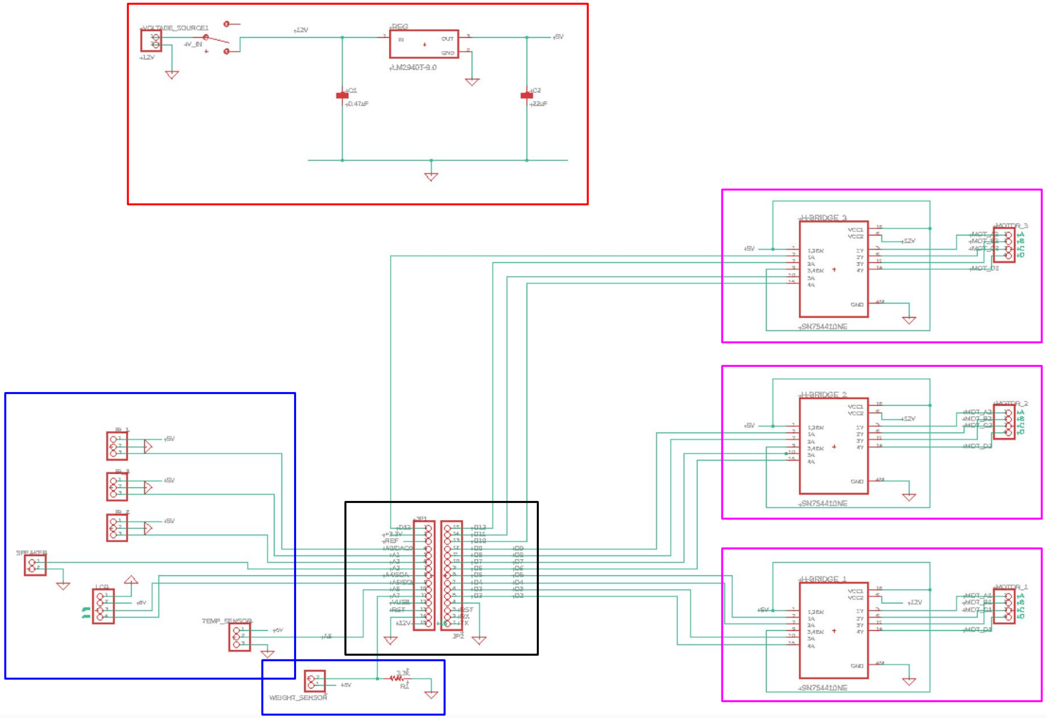
Originally for the PCB design we had various ideas on how to incorporate certain components, however, in the process of making the PCB we encounter certain issues that lead us to incorporate more components that were not predicted at the early stages of the design. Some of these challenges faced during the design were:

- Component Shortage
- Multiplexer/Demultiplexer
- Quadruple H-Bridge
- AC/DC Voltage

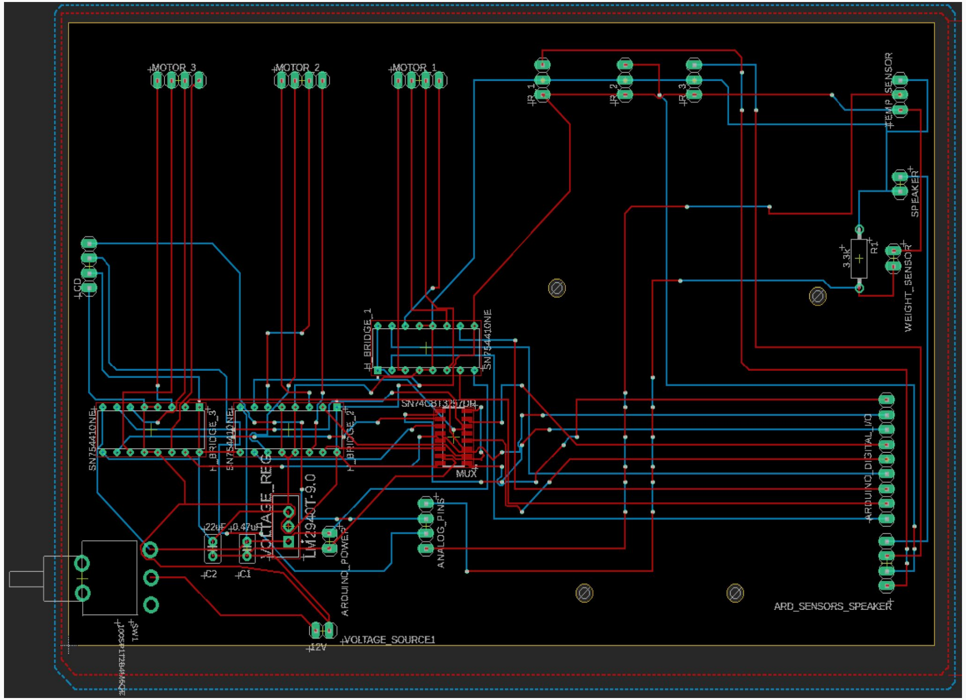


PCB SCHEMATIC

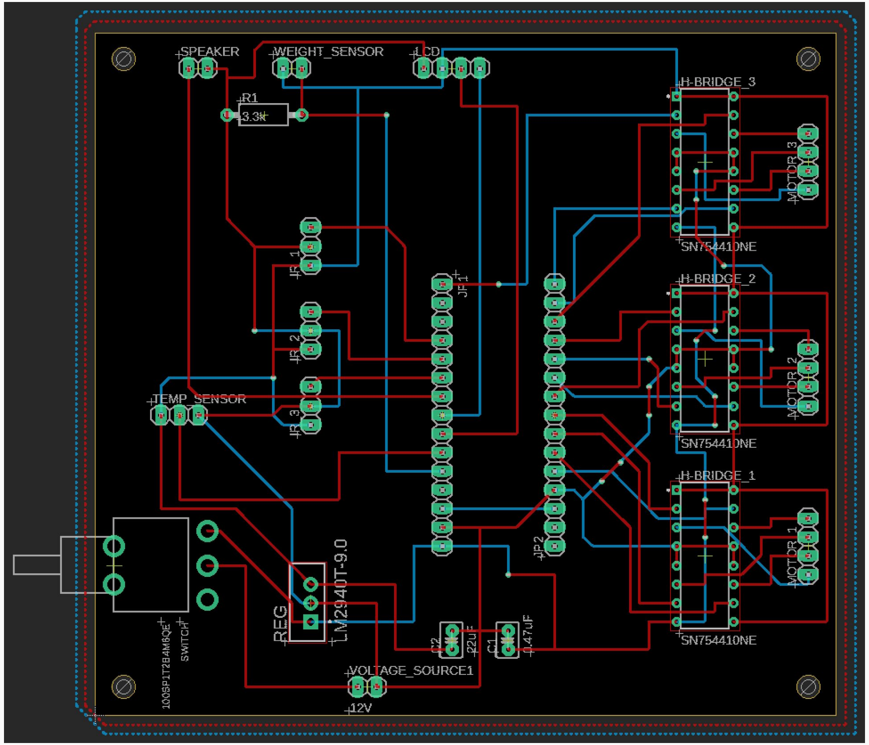
5V Voltage Regulator



PCB BOARD



Old



New

WHY A MOBILE APPLICATION?

For a platform, it needs to be as accessible as possible. Using a mobile device allows easy notifications and on-the-go management of the system.

Not using a mobile application requires the user to have to be around either a computer or the actual device. This is not optimal and we would have to use a text message to notify the user or not notify them at all.

Platform	**Requires Internet**	**Requires Backend**	**Notifications**	**Support**	**Modularity**
Native Screen	No	No	Around device	2/5	Not very modular
Web Application	Yes	Yes	Around PC	5/5	Very modular
Desktop Application	Yes	Yes	Around PC	3/5	Somewhat modular
Mobile Application	Yes	Yes	Everywhere	4/5	Very modular

WHAT PLATFORM ON MOBILE?

Native to Android or iOS

Up to preference

Frontend Framework/ Language	Android	iOS	Hot-reloading	Ease of Use	Support
Java/Kotlin	Yes	No	No	3/5	3/5
Swift	No	Yes	No	4/5	3/5
React Native	Yes	Yes	Yes	5/5	5/5
Flutter	Yes	Yes	Yes	5/5	3/5

Our software lead has used React Native and React in multiple projects so it made the most sense to use React. Javascript is also an industry-standard compared to Dart- which Flutter uses.

GOOGLE FIREBASE CLOUD DATABASE

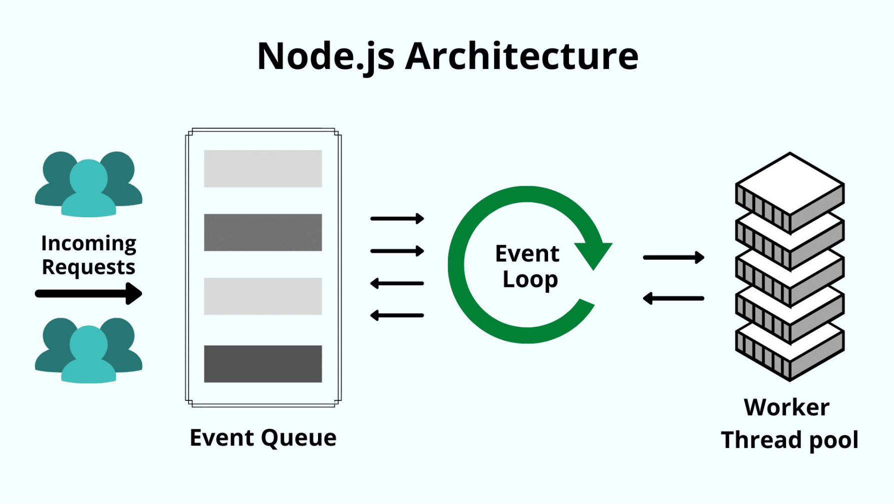
Our software lead has used Firebase and AWS before so it made sense to use either. AWS in general is more complicated and not really used as much for smaller projects due to its complexity.

Data	**Offers NoSQL**	**Free tier**	**Infrastructure Complication**
Google Firebase	Yes	Yes	2/5
Amazon Web Services	Yes	Yes	5/5
Microsoft Azure	Yes	Yes	4/5
Heroku	Yes	Yes; but at a much lower scale	2/5

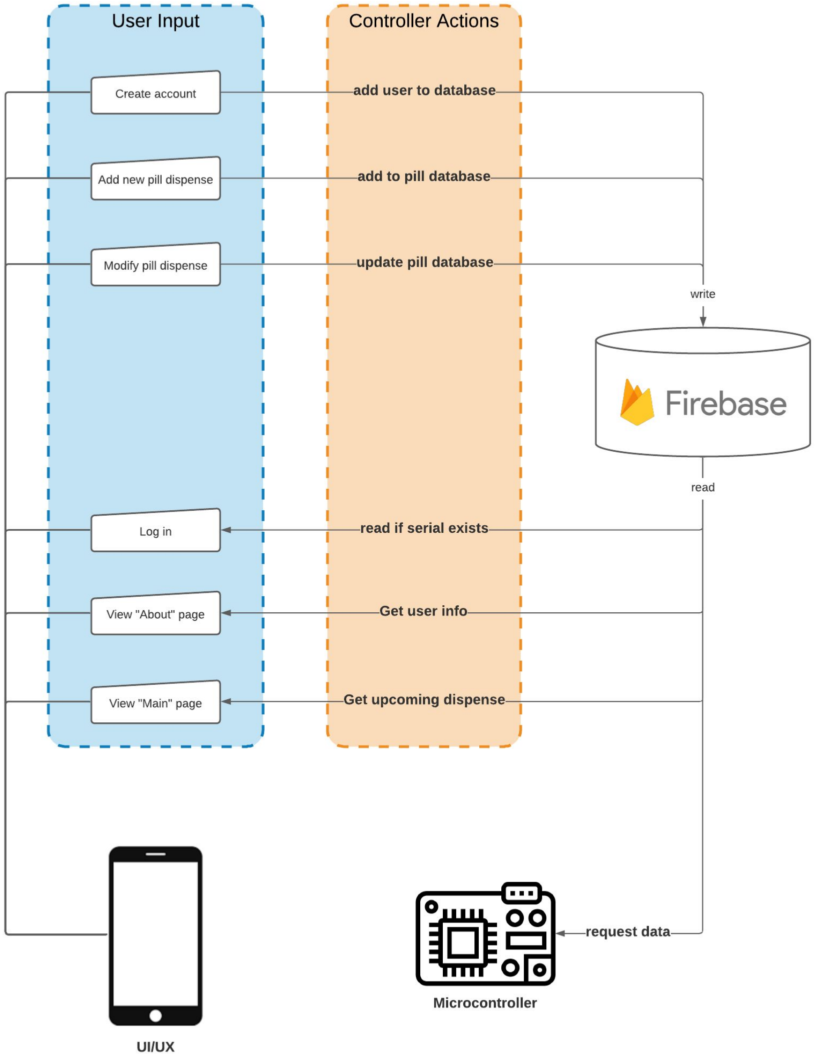
Another big factor is due to Firebase’s NoSQL database, which stores data as JSON-like data in documents and collections. This is easier to manage for accounts and also offers basic security features such as encryption and authorization.

NODE.JS BACKEND SERVICE

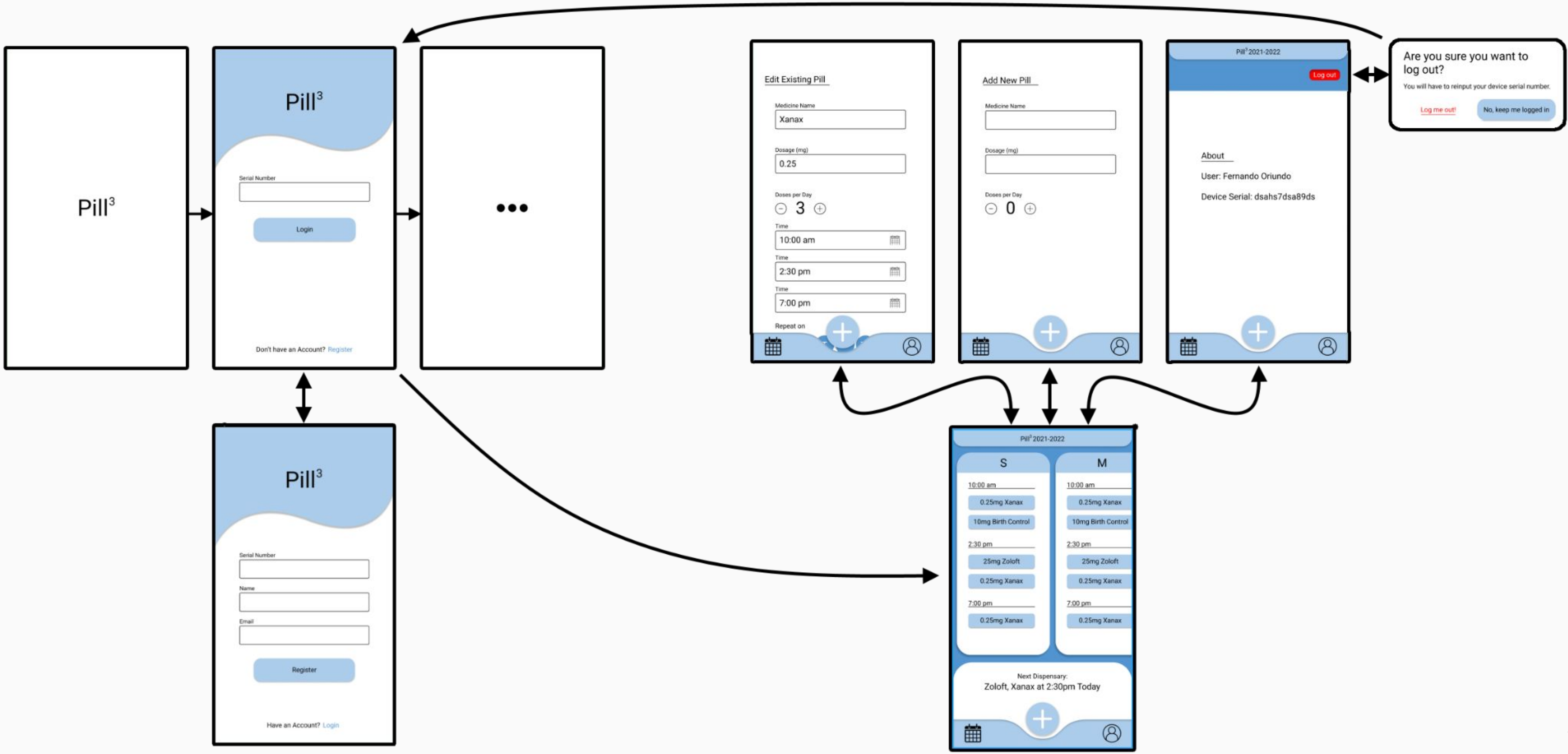
We are using Node.js as it is one of the easiest-to-use, most industry-wide, and has very little competition when it comes to asynchronous I/O in most applications. It is an event-driven architecture and makes communication from requests to data straightforward.



BACKEND DESIGN



ORIGINAL UI/UX DESIGN



Edit Existing Pill

Medicine Name: Xanax

Dosage (mg): 0.25

Doses per Day: 3

Time: 10:00 am

Time: 2:30 pm

Time: 7:00 pm

Repeat on

Add New Pill

Medicine Name

Dosage (mg)

Doses per Day: 0

Pill3 2021-2022

Log out

About

User: Fernando Oriundo

Device Serial: dsahs7dsa89ds

Are you sure you want to log out?

You will have to reinput your device serial number.

Log me out! | No, keep me logged in

Pill3 2021-2022

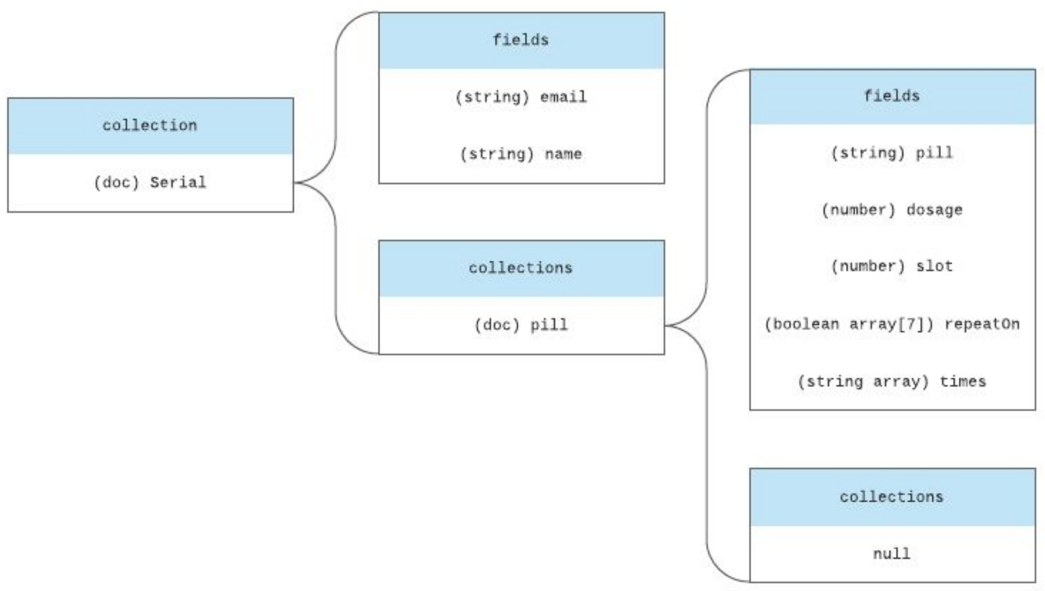
S	M
10:00 am	10:00 am
0.25mg Xanax	0.25mg Xanax
10mg Birth Control	10mg Birth Control
2:30 pm	2:30 pm
25mg Zoloft	25mg Zoloft
0.25mg Xanax	0.25mg Xanax
7:00 pm	7:00 pm
0.25mg Xanax	0.25mg Xanax

Next Dispensary:
Zoloft, Xanax at 2:30pm Today

NOSQL DATABASE DESIGN

- Firebase uses a NoSQL database storage method which stores data as JSON objects.

- Device’s serial number
- User’s email
- User’s name
- Pill dispensing
  - What pill
  - What slot to dispense from
  - Dosage
  - When to dispense
    - What days to dispense
    - What times to dispense on each day



collection

(doc) Serial

fields

(string) email

(string) name

collections

(doc) pill

fields

(string) pill

(number) dosage

(number) slot

(boolean array[7]) repeatOn

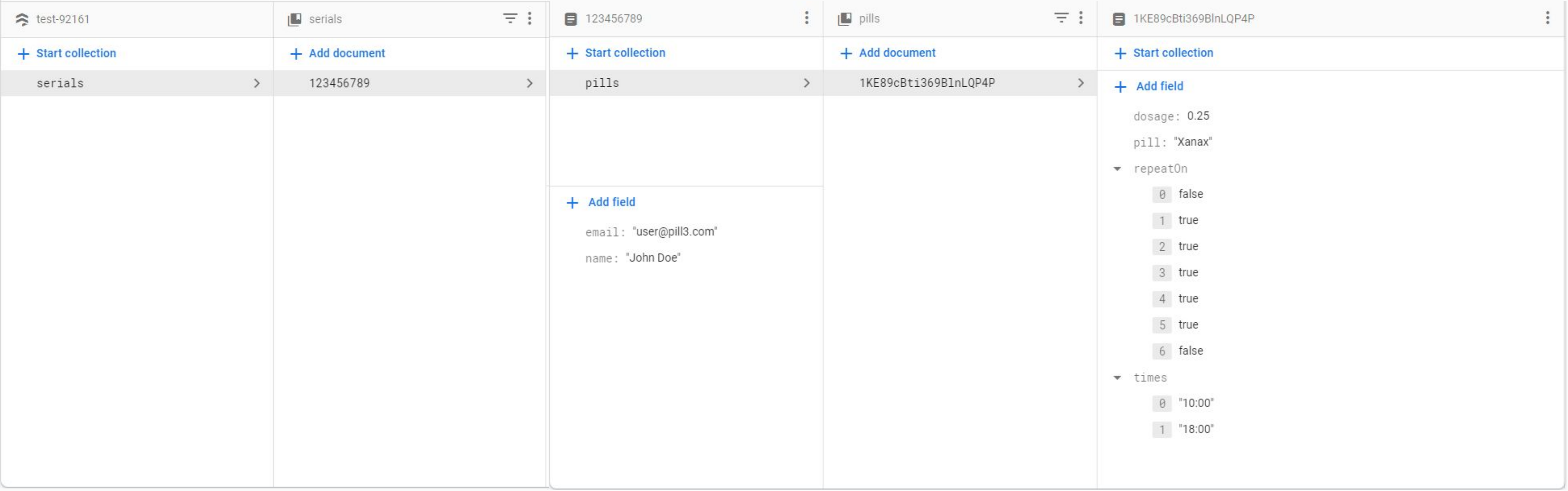
(string array) times

collections

null

Firestore allows for *collections*, *documents*, and *fields*.
Collections contain *documents*.
Documents contain either *collections* (subsets) or *fields*.
Fields are composed of some **data type**

NOSQL DATABASE DESIGN ***(CONT.)***



serials > 123456789 > pills > 1KE89cBti369BlnLQP4P

123456789:

```
email: "user@pill3.com"
name: "John Doe"
```

1KE89cBti369BlnLQP4P:

```
dosage: 0.25
pill: "Xanax"
▾ repeatOn
    0  false
    1  true
    2  true
    3  true
    4  true
    5  true
    6  false
▾ times
    0  "10:00"
    1  "18:00"
```

TESTING - WHY MANUAL?

Testing is wonderful; however, it can get complicated very fast. This begs the question if using a testing framework is even worth it.

Testing Framework	Ease of Use	Support	Applicable	Readability
Jest	3/5	5/5	Yes	2/5
Detox	3/5	1/5	Yes	3/5
Manual	5/5	N/A	Yes	5/5

The choice of not including a testing framework is mainly due to time- as it is unnecessary to test something on such a small scale. If time permits, we can then decide on the best framework.

BUDGET

Quantity	**Part**	**Manufacture**	**Manufacture Code**	**Price**
1	Arduino nano IoT 33	Arduino	ABX00032	$25.55
3	IR Sensor	Esooho	EK1254x5	$8.28
5	PCB	Jlcpcb	-	$22.63
3	Stepper Motor	Adafruit	XYU42STH34-0354A	$56.01
1	Speaker	Arduino	TPX00080	$7.99
1	Display	MakerFocus	SSD1306	$7.99
1	Temperature Sensor	Texas Instrument	LMT85LP	$1.60
1	Pressure Sensor	SparkFun	SEN-09376	$18.34
1	AC/DC Converter	ALITOVE	CJ-1230	$10.64
1	Chassis Materials	Various	-	$128.77
			Total	$287.80

MILESTONE

	Hardware	Software
Week 1	Complete PCB design and start CDR	Finalize UI/UX diagram
Week 2	Get materials for chassis and finish CDR	Create UML class diagrams
Week 3	Start assembly of chassis	
Week 4	Test electrical components and continuation of chassis built	Set up programming environment
Week 5	Breadboard and PCB testing	Set up Firebase cloud environment ro connect to backend controller
Week 6	Solder all components to the PCB Board	Get backend controller working with writing to database

MILESTONE (CONT.)

	Hardware	Software
Week 7	Finish soldering	Get backend controller working with reading from database
Week 8	Finish built of chassis	Create basic frontend components components to connect to backend functions
Week 9	Test runs and polish device	Login authentication and login UI/UX
Week 10	Beautify interior and exterior	Create primary dashboard
Week 11	Beautify interior and exterior	Beautify & bug fixes
Week 12	Final test run and adding finishing touches	Beautify & bug fixes

QUESTIONS?