Smart Pill Cabinet

Master's Degree Project Report fulfilling the requirements for the course

COEN 315: Web Architecture and Protocols

Winter 2017

Submitted to

Prof. Amr Elkady

Dept. of Computer Science and Engineering

By:

Team GoIoT

Lakshmi Shankarrao

Varun Srinivasan

Sri Mourya Dommeti

Preface

This Project report has been prepared in fulfilment of the requirement for the course: **COEN 315 - Web Architecture and Protocols** of the programme M.S. in Computer Science in the academic year 2016-2017. The blend of learning and knowledge acquired during our sessions in the class is used to come up with this project.

Lakshmi Shankarrao

Varun Srinivasan

Sri Mourya Dommeti

Acknowledgements

We would like to extend our gratitude towards Prof. Amr Elkady, who enhanced our knowledge and inspired us to make this project. We would also like to thank Santa Clara University for providing us the opportunity to learn the subject and apply its concepts into making this project.

Table Of Contents

Table of Figures

Table of Tables

ABSTRACT

Blame it on the busy life or old age, people often forget to take pills at the right time. And some forget whether they have taken a pill or not and consume more than the required amount which can be detrimental to the health. Using IoT, we have designed and implemented a smart pill cabinet that will not only remind people to take pills and avoid repetitions, but also remind them to order refills. We also display adherence statistics for the day and notify family when pills are missed. We have designed and implemented a web portal, an android application to support and transform the pill cabinet to be smart. We discuss our product implementation and technical details and finally show results obtained. By the use of web of things we provide better health monitoring.

INTRODUCTION

Adherence to medication is categorized as one of the major clinical problem in health care system. It is possible to achieve optimal medication adherence by making it easy for patients filling pills and consuming it properly on a timely basis. After the patient has received appropriate prescribing, it is important to establish a platform to maintain appropriate records that are easily accessible and develop solutions to improve adherence. Poor adherence to medication is a major cause of poor health and also accounts for huge health care costs in the United States. Of all medication-related hospital admissions in the United States, 33 to 69 percent are due to poor medication adherence, costing more than $100 billion annually in increased medical costs. We work towards providing a solution to this problem.

BACKGROUND

Purpose

A recent study shows that nearly 3 in 5 American adults take a prescription drug[1]. In a study published by the Journal of the American Medical Association (JAMA), researchers found that the prevalence of prescription drug use among people (20 and older) had risen to 59% in 2012 from 51% just two years earlier[2]. During the same period, the percentage of people taking five or more prescription drugs nearly doubled, to 15 percent from 8 percent.

There are patients who have to consume different pills at different times, but they don't know which pill to take when. Mostly, the patients (often with dementia) tend to forget if they took their pills. This situation leads to insufficient intake of medicines, causing additional health problems and reducing quality and peace in life.

Theoretical basis and background

The problems primarily faced by the patients are - (i) making a common mistake of skipping a dose, (ii) not avoiding drug interactions, (iii) not staying on schedule[3]. These problems are handled effectively with just one device implemented in this project. Additionally, the project focuses on managing prescriptions electronically (replacing physical handouts) which can be laborious to store safely and produce it every time to the pharmacist for recurrent purchases.

Research

A few electronic pill boxes and dispensers are already available in market, which does not constitute all the features that are proposed and designed in this project. All the devices currently available are designed to remind the patient about his pill time either by cell phone notifications or by vibrating a wrist watch, (which is merely the job of an alarm clock!)

APPROACH

Our Solution

The most important features of the smart pill cabinet are:

1. Timely Reminders
2. Adherence Monitoring
3. Pill Refill Notification

Like generally followed, we consider 3 slots of pill consumption. Morning, Afternoon and Night. The slots are as follows:

Morning Slot:	8am to 10am
Afternoon Slot:	1pm to 3pm
Night Slot:	7pm to 9pm

The doctor can add/update prescriptions for the patient in the Web Portal which the patient can access by logging in using his ID.

When it is time to take a pill, the Pill cabinet will attract the attention of the user with the buzzer sound and a Red LED glowing. The user will also receive an notification(Android App) which will give him the list of pills that are due for the time. Further when the user opens the cabinet drawer, he can see the pill bottle that are due for the time will have a Red LED glowing.

Now that we have successfully indicated to the user that it is time to take pills, the next step is to monitor his adherence. We monitor the draw open/close action to figure out if pills are taken or not with the help of distance ranging sensors. But to make it more foolproof, we ask the user to press a button on the pill bottle after taking the pill. We planned to remind the user to press the button by displaying a message on the LCD. And after the user has pressed the button, we turn off the Red LED on the respective pill bottle and turn on the White LED. This is to help prevent repeated consumption of pills. If any pill was missed, the family will be notified of the same.

We maintain the existing count of the pills in the bottle using the adherence history. We measure the existing quantity of pills by using distance sensors. We use this real time

data along with the count history to determine if and when a refill is needed for a pill. And the patient will be duly notified of the same.

Architecture

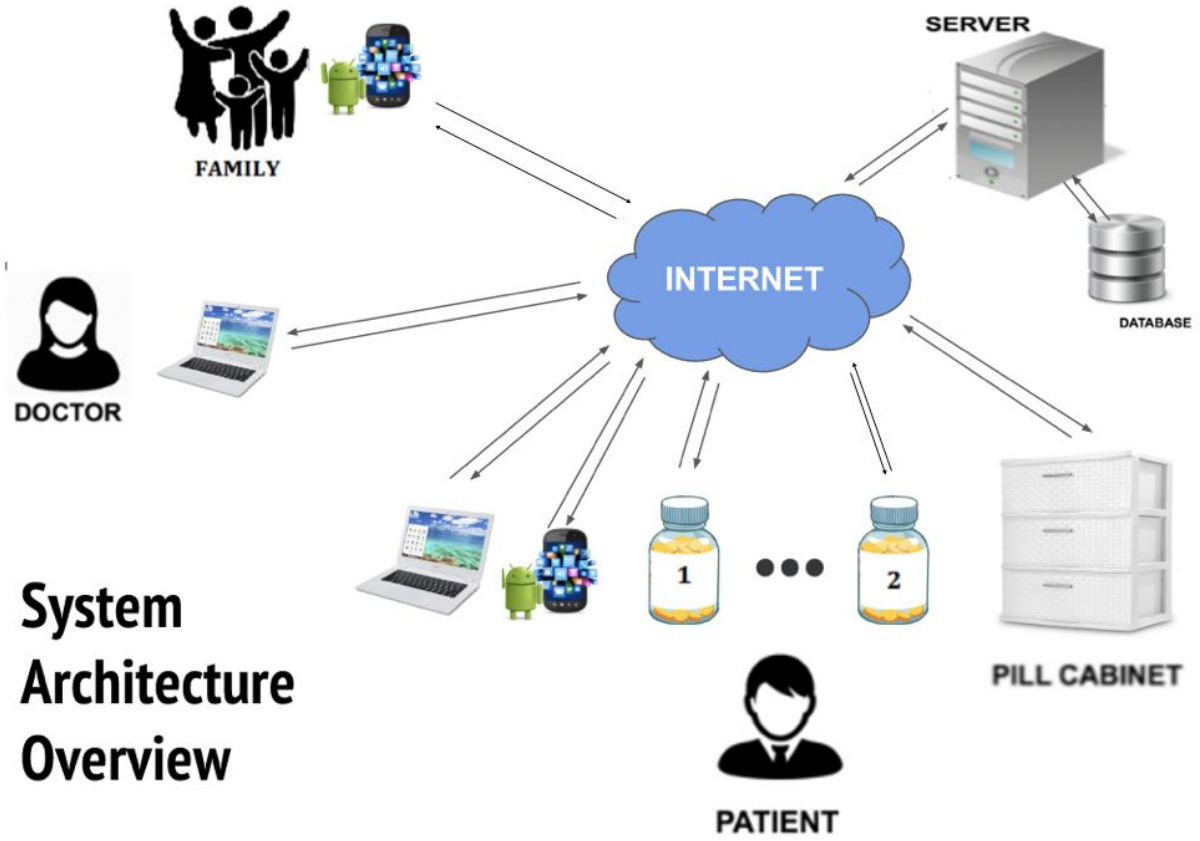


Figure 1: System architecture diagram

Initially, the admin creates login credentials for the doctor and the patients, and associates the respective patients to the doctors on demand. While consulting, the doctor diagnoses the patient and prescribes medicines on the web portal. As soon as the patient is prescribed with his/her medicines, the admin configures the pill cabinet and pill bottles specially tailored for the patient. The patient, then purchases the cabinet and the pill bottles from the admin.

The pill cabinet and pill bottle is plugged to the power supply to pair it with the web portal through APIs, and alert the patient through buzzer(in the cabinet) and a RED light (on the pill bottle) in the respective slots if a pill is due at that time. Also, a notification on the patient's android phone is posted to remind the patient for his pill time. Once the patient takes his pill, he presses the button on the pill bottle. This raises a POST request to the web portal with the patient ID, pill ID and the timing slot (m for morning, a for afternoon and e

for evening). As a result, the adherence table is updated, and is immediately reflected on the web portal. The RED LED on the pill bottle ceases to glow, and the WHITE LED starts glowing till the end of the time slot to inform the patient that the pill is already taken. After the cycle is over, both the cabinet and pill bottle starts listening to the API again by posting GET requests with the patient ID, pill ID and the timing slot (m/a/e) correspondingly and repeats the process for the next time slot.

TECHNICAL DETAILS

Front-End

Technologies used: HTML5, CSS3, JS

Welcome page link: http://goiot.varuns.me

Web portal link: http://35.167.209.121:8080/info.mourya.goiot/Login.html

Website Navigation Flow

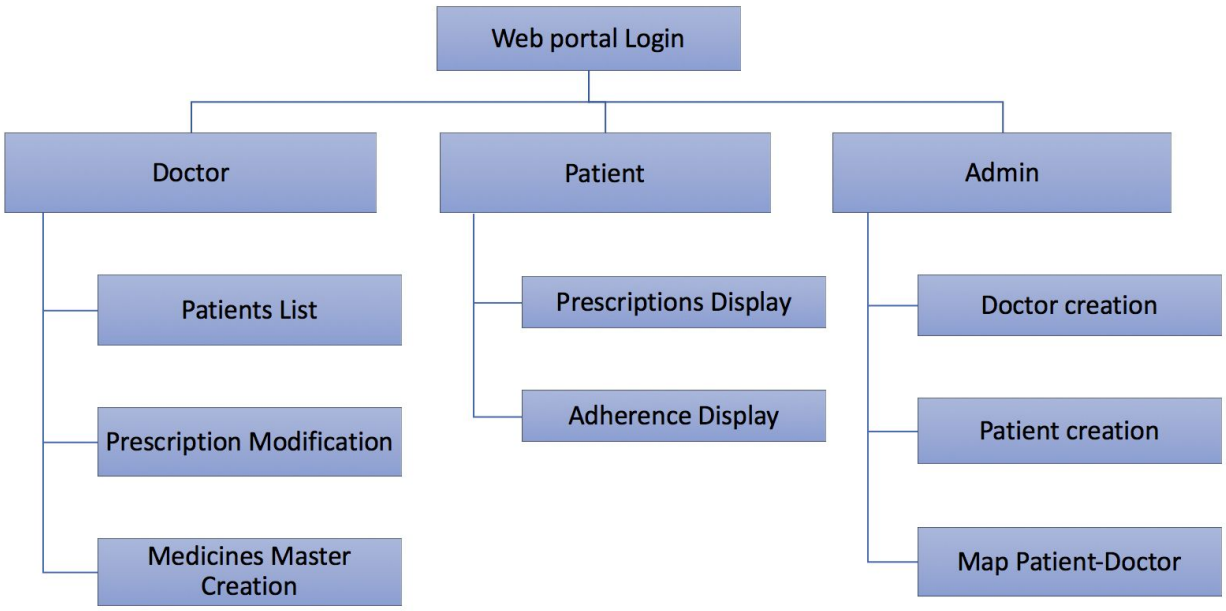


Fig 2 : Web portal navigation flow diagram

Snapshots

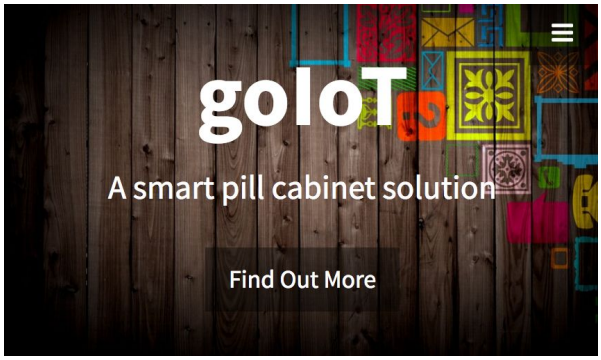


Fig 3a : goIoT welcome page

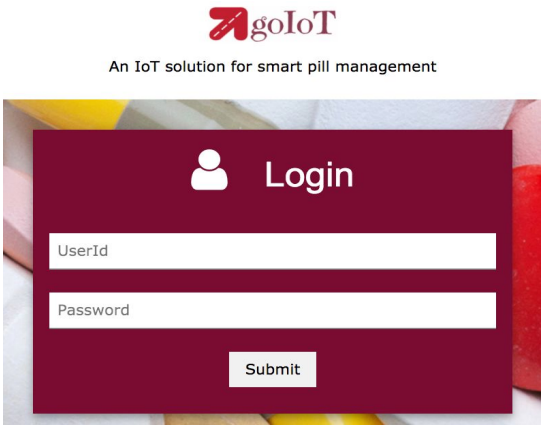


Fig 3b : Web portal Login page

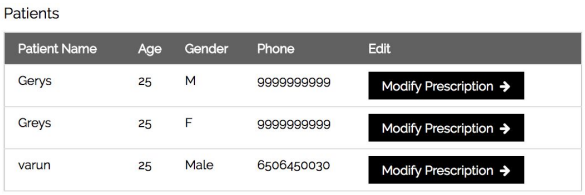


Fig 4a : Patient List page

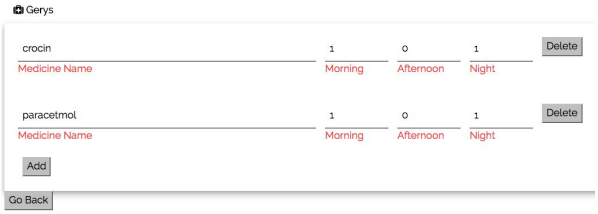


Fig 4b : Prescription Modification page

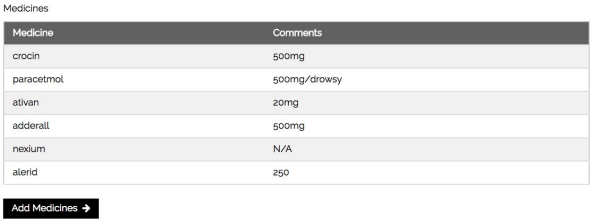


Fig 4c : Medicines master data page

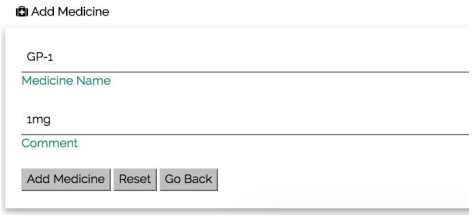


Fig 4d : Medicines master data addition page

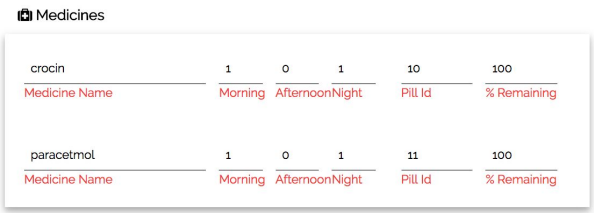


Fig 5a: Patient's prescription page

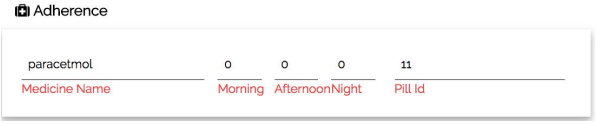


Fig 5b: Patient's adherence page

Fig 6a: Doctors list page

Fig 6b: Doctor creation page

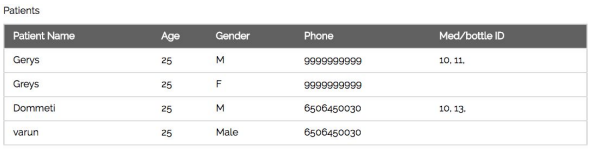


Fig 6c: Patients list page

Fig 6d: Patient creation page

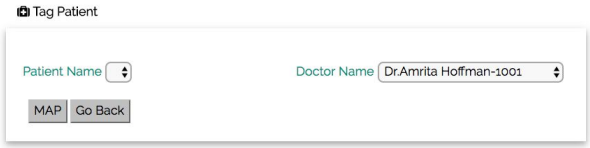


Fig 6e: Patient-Doctor Mapping page

Web Application

Languages/Technologies	java, javascript, html, jsp, sql, xml , phpmyadmin
Application Server/Software	Apache Tomcat 7, AWS, Eclipse
Framework	Spring MVC (4.0.4)
Database	MySQL (5.7)
Code available at	https://github.com/varunsme/goIoT

Database

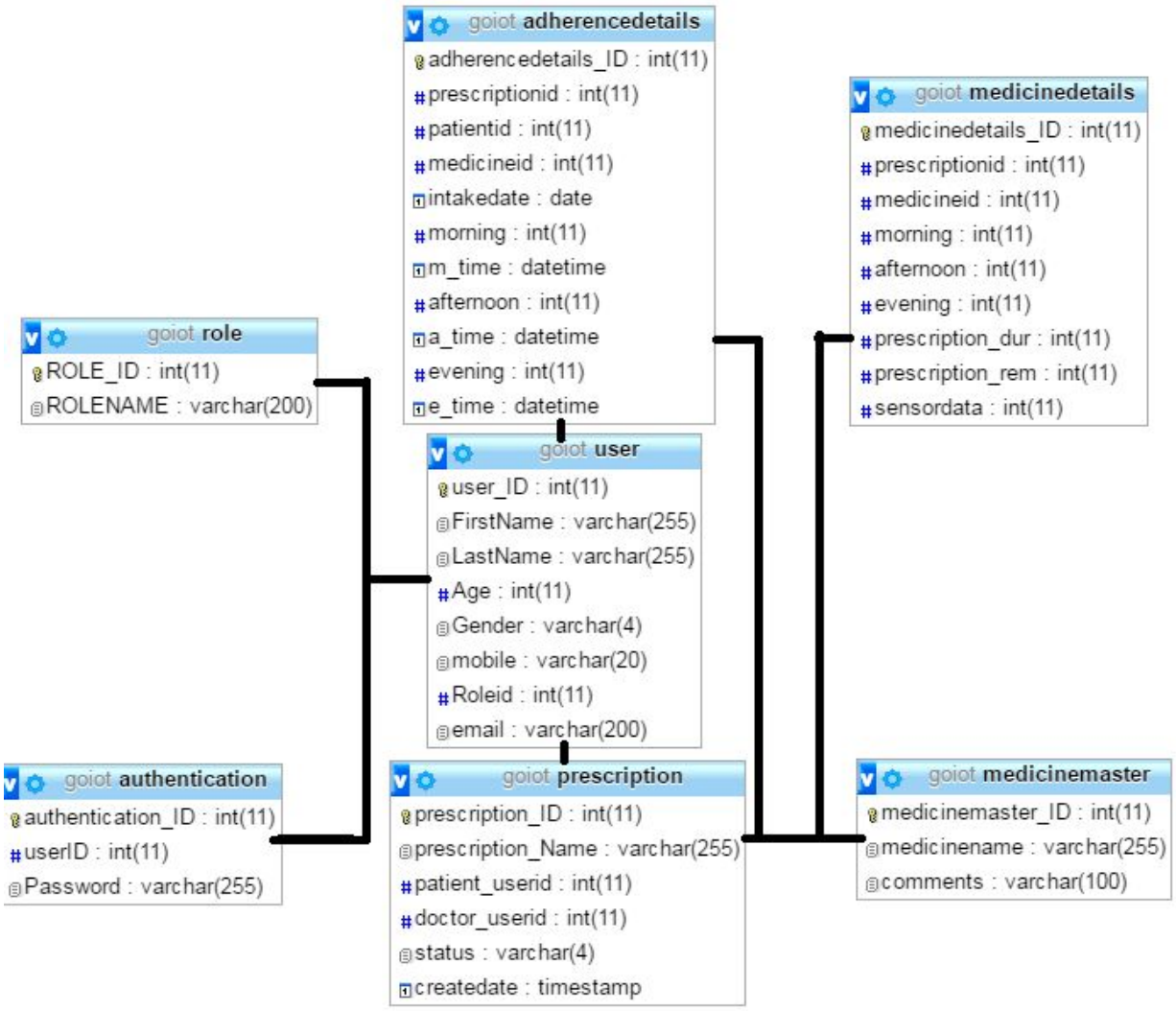


Fig 7: DB EER diagram

UML Class Diagrams

Fig 8: UML Controller class

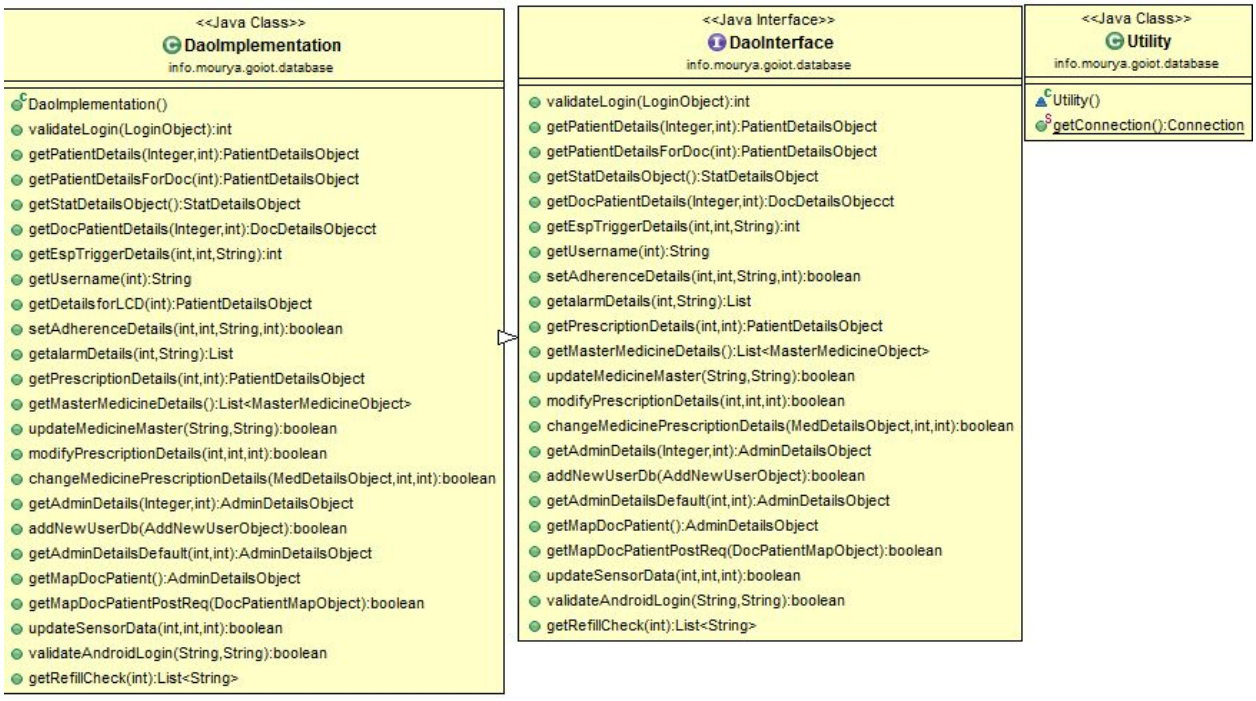


Fig 9: UML Model class

REST APIs

Table 1: Status API

API Format	Usage	Data returned
/espTrigger/patient_id/medicine_id/m	Info whether pill to be taken or not	1 or 0
/prescriptions/patient_id	Pill time table Info for android, LCD	{"Gerys" "Crocin 1-0-1, Paracetamol 1-1-1"}
/alarmInput/patient_id/m	Timely check whether pill is taken or not	{"Crocin, Paracetamol"} or 1 if pills taken
/refillCheck/patient_id	Reminder for refill, if less than 30%	{"Crocin, Alerid"}
/androidLoginDetails/patient_id/password	Android app login check	1 or 0

Table 2: Update API

API Format	Usage
/adherenceUpdate/patient_id/medicine_id/m/1	Updates adherence table if pill is taken to DB
/pillRemainingPercent/patient_id/medicine_id/45	Updates refill percentage to DB

Note: patient_id is unique for the cabinet; patient_id and medicine_id is unique for the pill bottle.

Hardware Setup

Components List

ESP8266 Microcontroller - 3nos.

USB to TTL converter - 3nos.

Breadboards.

HC-SR04 Ultrasonic Ranging Sensors - 3nos.

Passive Buzzer Module 5V.

4.7K Ohms Resistor.

SainSmart IIC/I2C/TWI 1602 Serial LCD Module Display - 1nos.

Red LEDs - 3nos.

White LEDs - 2nos.

Push button - 2nos.

Breadboard Power Supply Module 3.3V/5V - 1nos.

9V 1A Power Adapter for Arduino (2-Flat-Pin Plug / 100CM Cable) - 1nos.

Dupont Wire Male to Male, Male to Female, Female to Female Jumper Cables - reqd nos.

**The HC-SR04 Ultrasonic Ranging Sensors have a measuring range ranging from 2cm to 400cm and an accuracy of 3mm[23].

ESP8266 pin diagram

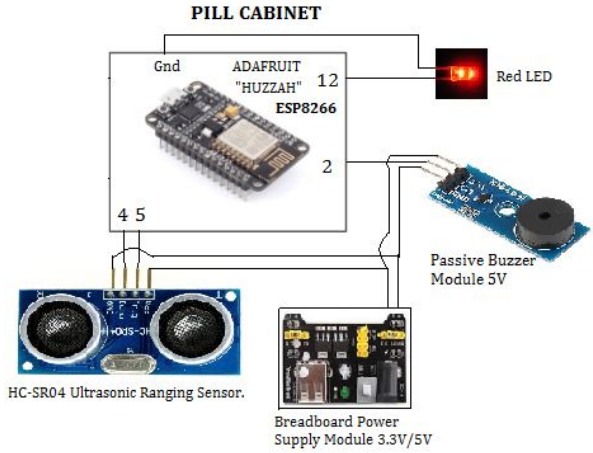


Fig 10: Pill Cabinet Pin Diagram

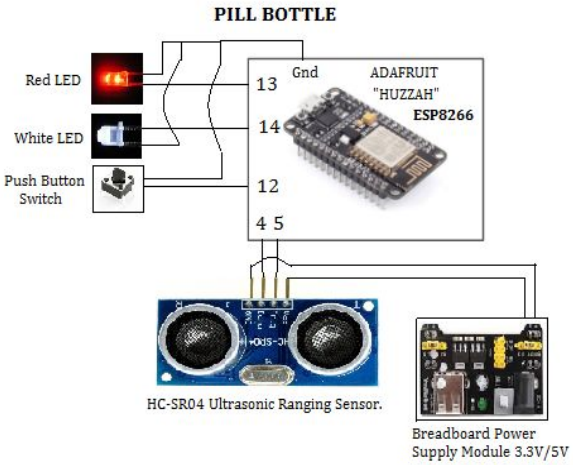


Fig 11: Pill Bottle Pin Diagram

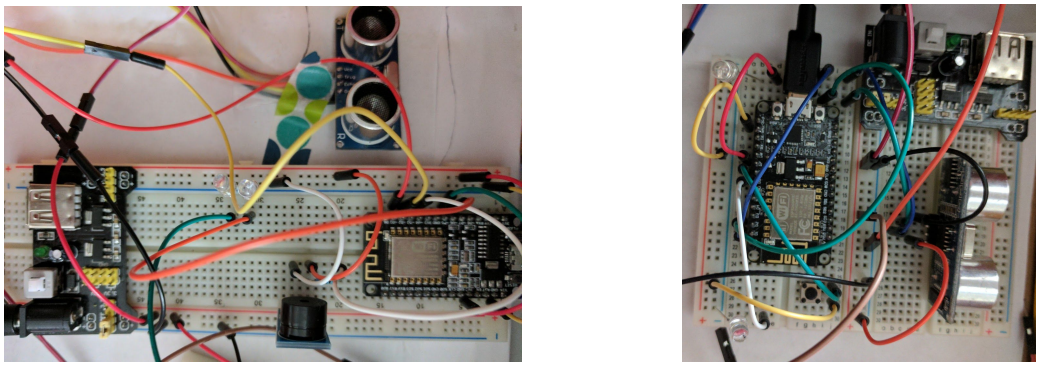


Fig 12: Implemented circuitry of (a) Pill cabinet & (b) Pill bottle

Firmware

Pill Cabinet Overview

The pill cabinet code runs at particular time intervals and is implemented using a variety for alarms and corresponding alarm handlers. Alarms are triggered by ESP8266 software based interval timers. When the alarms are triggered, we set a corresponding boolean to true to enable a particular functionality to be executed in the loop(). After the alarm handler gets called by the loop(), as a first step in handling any alarm, we set the alarm for the next day for the same time. We have 2 kinds of alarms. Ones pertaining to the general slots and the other pertaining to getting the general prescription for the day.

For the slot alarms, we send a GET Request to the server to see if any pills are due for the time. We read the response and parse it to extract the actual information. Based on that we handle the behaviour of the Buzzer[25][26], LED and LCD. After all the appropriate indicators are set, we start the draw open/close monitoring which extends throughout the slot period.

For the alarm pertaining to getting the general prescription for the day, we send a GET Request to the server to receive info about the same. We parse the response and extract information pertaining to the name and the prescription for the whole day and plan to display it on the LCD. At the end we wrap-up the slot by turning off the LED.

Finally, in any alarm handler, we set the boolean back to false to indicate task completion and to stop the loop from continuing to execute it.

Pill Bottle Overview

Here too, similar to the Pill Cabinet, alarms are triggered by ESP8266 software based interval timers. And when the alarms are triggered, we set a boolean to true to enable a particular functionality to start running in the loop(). We first set the alarm for the next day. We have 2 kinds of alarms. Ones pertaining to the general slots and the other pertaining to the pill bottle quantity status check.

If the alarm triggered belongs to one of the general slots, then we first set an alarm for the next day. send a GET Request to the server to check if the pill is taken for that particular slot. If the response indicates a yes, we go ahead and set the Red LED. We start to monitor the push button. We monitor the push button pin for the pressed status twice every second. If and when the button is pressed, we send an adherence update to the server. When the slot period is completed, we check if LEDs are glowing to switch them off.

If the alarm triggered is pertaining to the pill bottle quantity status check, then we take about a thousand readings to accurately calculate the remaining pill percentage(rpp) in the bottle. And send a GET request to the server to update the rpp.

Finally, we set the boolean back to false to stop the loop from continuing to execute it.

Engineering details

Alarms

The alarms are triggered by ESP8266 timers. There are 2 kinds of timers. One is the hardware based timer(hw_timer). It might be used by the ESP8266 which provides less flexibility. And just 1 timer can be set. But our functionality needed more. There is also very less documentation about it. The second kind is a software based interval timer(os_timer)[22]. A maximum of 7 timers can be set. We're using 4. We get the current time using HTTP HEAD request to google.com[27]. HEAD is a request in which the server returns just the header and no message-body in response. But otherwise similar to GET. The HTTP RFC defines a Date header so pretty much all websites will return the current date/time in all http responses. We use www.google.com for our purposes. We extract the time and convert it to the current PST timezone time and calculate the time in millis for the next alarms to be triggered based on the slot timings defined and initiate the alarms.

Refill check

We declare an integer array of size equal to the length of the pill bottle. We take 10 readings every second. We use the New Ping Arduino Library for getting the sensor readings. We ignore all readings that are greater than the pill bottle length and lesser than the error threshold. And increment the value in the array with index equal to the reading. And we find the most frequent reading by choosing the index which contains the maximum value.

Draw open/close monitoring

We take 4 readings every second for 2 hours. We use the New Ping Arduino Library for getting the sensor readings. We pre-define the open draw height, the closed draw height and the error threshold(based on our observation we have fixed the value to 4). We ignore all values that are lesser than the threshold and greater than the sum of the open draw height and the threshold. We experimented and found that a minimum of 15 seconds is needed for a person to open the draw and take the pill out of a bottle. So we make a decision about the open and close draw state once every 15 seconds. We keep a count of open draw height values and the closed draw height values and compare them at the end of each 15 second cycle to make a decision about the state of the drawer. But we observed that the sensor is more sensitive and quicker in detecting objects that are moved closer towards it from a height and starts giving accurate values instantly. While it takes comparatively more time when objects are moved farther away from. Just about 3-4 seconds. But affects the overall reading greatly. Thus we introduce a weight for the higher distance values and multiply it with the open draw count before comparing it with the closed draw count. So when we detect a state change, and the drawer is closed back after

being opened, we send a GET Req to the server to check if all the pills are taken. If all are taken, then we stop the draw monitoring and exit from the loop. (We planned to visually indicate to the user about the status of the pill consumption and remind him to press a the button on the pill bottle if the server responds that not all pills are taken). If not, the monitoring continues throughout the slot period.

NewPing

We initialize NewPing to use pin 12 for trigger output, pin 11 for echo input, with a maximum ping distance equal to the maximum length of the bottle. It is, however possible to use same pin for both trigger output and echo input. We use in-built methods to get the echo time. The echo time will be in microseconds will be 0 if it detects no ping within the maximum distance limit set. The distance in cm is calculated as follows: distance = (echo time×velocity of sound (340M/S) / 2. Everything else is a constant except the echo time. The Arduino library provides a constant(US_ROUNDTRIP_CM) dividing by which we can obtain the distance in cm[24][28].

Android Mobile Application

Logic Flowchart

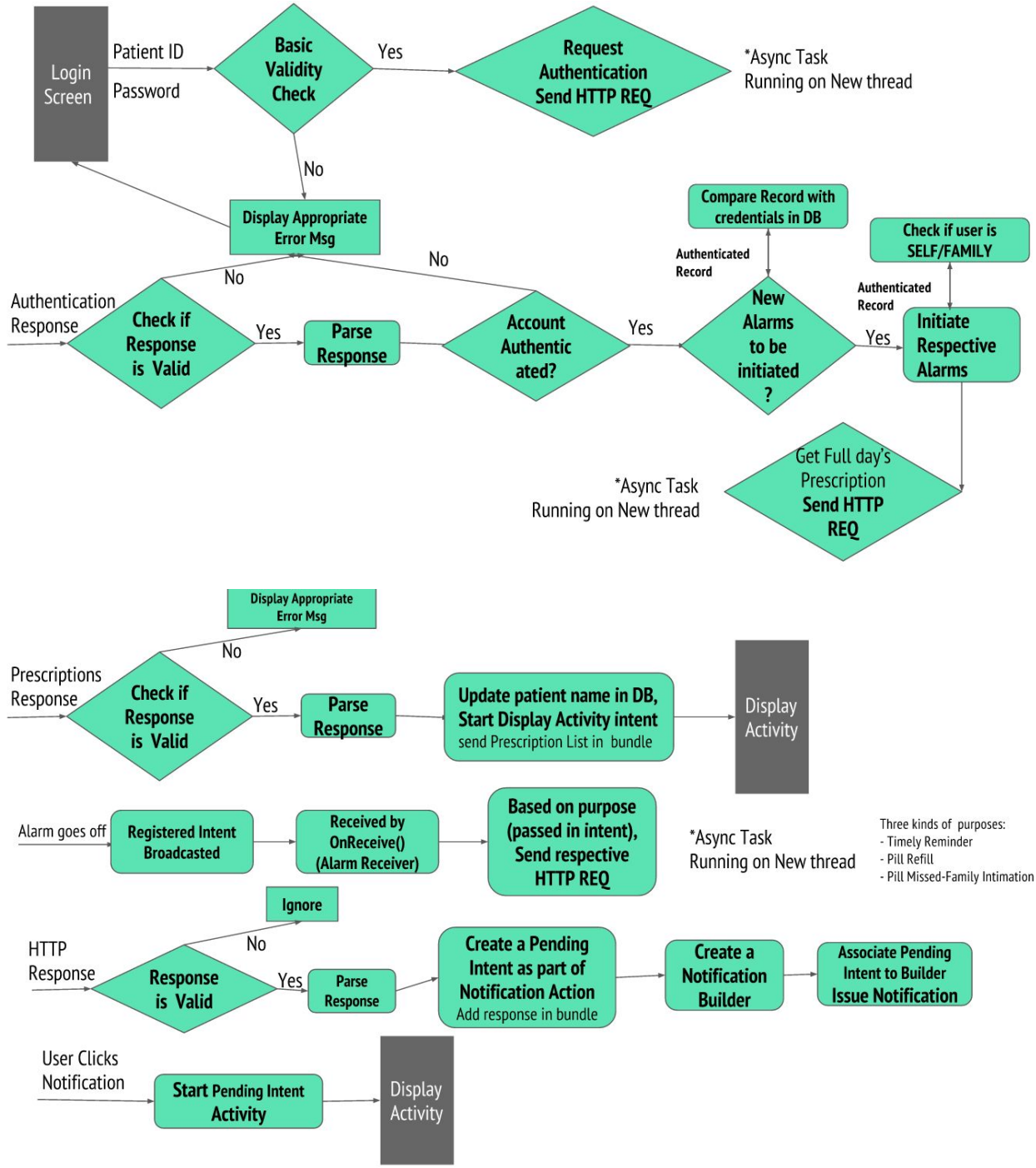


Fig 13: App Logic Flow Chart

The detailed logic flow of the app is shown in the flowchart above.

The Alarm Manager class and Receivers(registered receivers[9]) are used to help trigger alarms even when the app is not active. Android runtime notifies the registered receivers[10] when an event occurs i.e., Here, when current time is equal to the time set by our receiver, the onReceive() method of our receiver is triggered. Our receiver handles the alarm and appropriately builds and issues notifications[11][12].

We build upon the basic Login Activity template[13].

To display the medicines, we dynamically get the list of appropriate medicines and populate the listview in the "Detailed Display Activity"[14].

We handle HTTP requests using HttpURLConnection class[15]. For this we need to include permissions to allows applications to open network sockets[16]. Accessing the API is performed in a background thread[17].

Screenshots

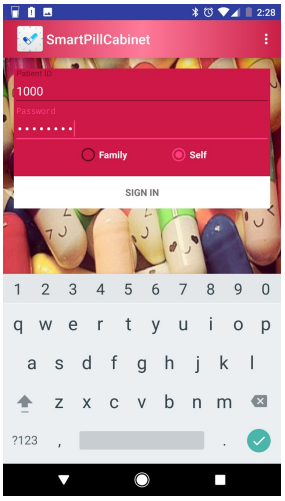


Fig 14a: App Login Screen

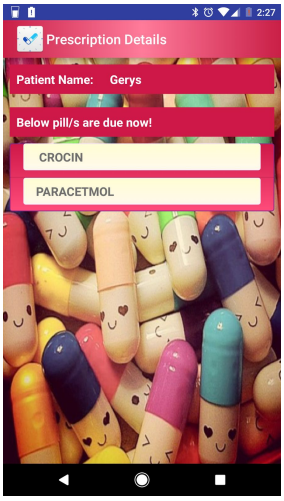


Fig 14b: Details Display Screen

Finished Product Design & Packaging

We planned to design and print the pill cabinet and pill bottles using Makerbot. But, since the objects to be printed is hollow and takes around 22 hours to print, the technicians in the maker's lab refrained from printing for longer times. The pictures of the designed objects are as follows.

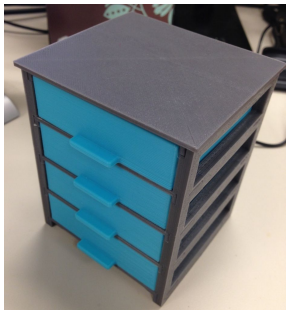


Fig: Designs for Pill Cabinet

Fig: Designs for Pill Bottle

Hence, we went ahead building our own pill cabinet with cardboards - by making a rectangular box, and attaching partitions in the box for making them supports for the draws. And we made three draws and covered everything with colored papers to an appealing look. We had to pierce holes in a couple of places for the wiring to pass through the cabinet and for the sensor to read the draw state of being open/close.

We also developed a pill bottle with a pen stand, attaching the sensor to the bottom and a closure cut out of cardboard at the top, and inverted the object to take pills by sliding a strip back and forth that covered a slit cut open in the cardboard closure.

Fig: Pill Cabinet

Fig: Pill bottle

RESULTS

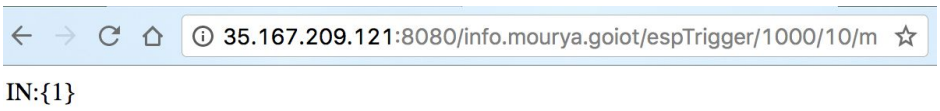
Welcome Page: http://goiot.varuns.me/

Website Link: http://35.167.209.121:8080/info.mourya.goiot/

Considering one cycle for 1 patient (with patient ID: 1000) who is intended to take two medicines (Pill Bottle 1 ID: 10, Pill Bottle 2 ID: 11) with pill bottles of height **10cm** each. Additionally, the closed Draw height is **7cm** and open Draw height is **21cm**. We are recording results of the three aspects discussed in the project, viz., timely reminders, adherence monitoring and pill refill reminders.

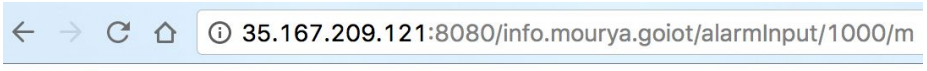
Timely reminders

- Sending GET requests:
  - From Pill Bottle 1:
    http://35.167.209.121:8080/info.mourya.goiot/espTrigger/1000/10/m



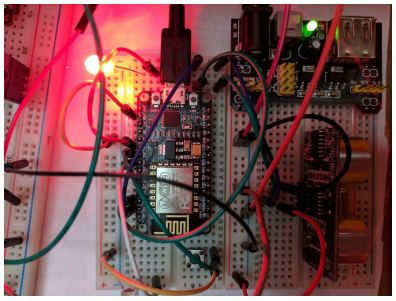
IN:{1}

  - From Pill Cabinet:
    http://35.167.209.121:8080/info.mourya.goiot/alarmInput/1000/m

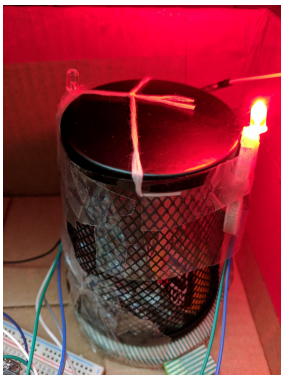


{" adderall, "}

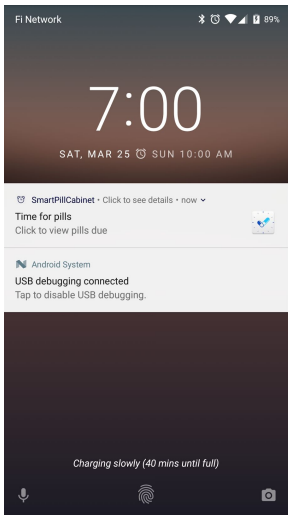
- Since the pill bottle receives “1” and the pill cabinet receives the medicine name “adderall”, the patient is due to take the medicine. Hence, the buzzer rings, and the Cabinet’s RED LED goes ON.

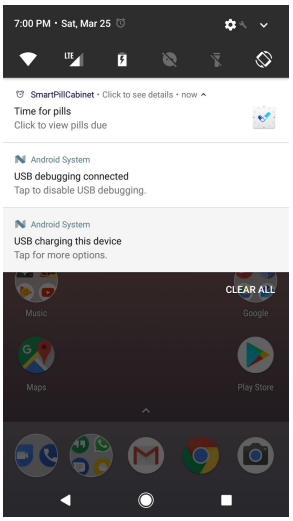


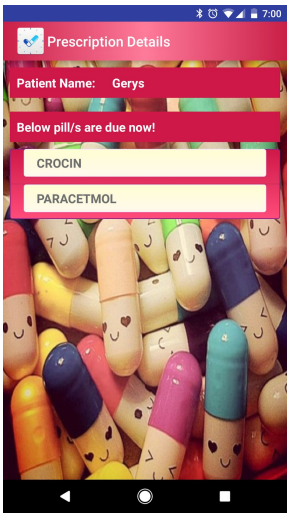
- Simultaneously, the Pill Bottle’s RED LED goes ON too.



  - A reminder notification is posted in the android app.





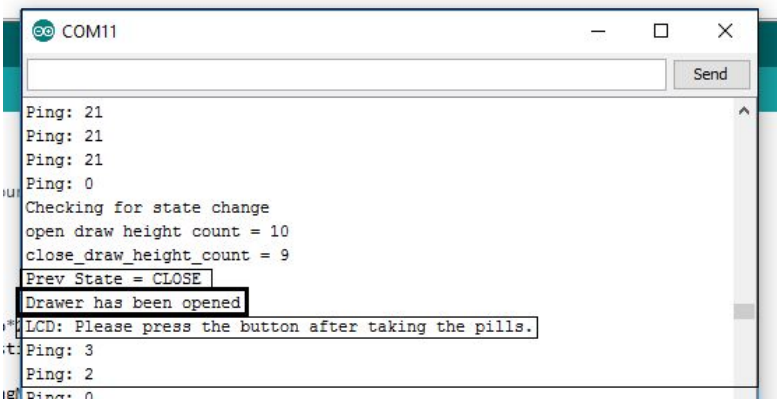


Adherence monitoring

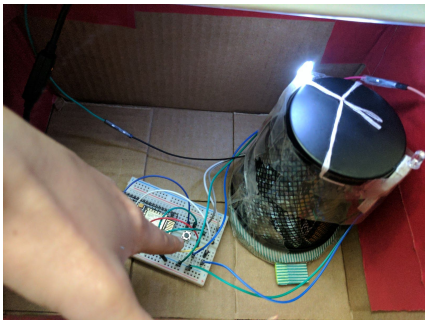
Initially, we had implemented an API that would update the “pill taken” status of all pills due for the patient. This was done for a particular slot when the Draw-Open event was detected. THe status “1” was set to indicate TAKEN status. Later we changed the logic to intimate the user to press the button on the pill bottle when the Draw-Open event is detected with the help of an LCD. And when the user presses the push button on the pill bottle, update the “pill taken” status to 1 for that pill bottle. The LCDs we tried with didn’t work properly for reasons discussed in the Lessons Learnt section. Thus please refer to the logs for the Draw open/close event detection.

- Taken
  - Initial condition -> Pill not taken yet

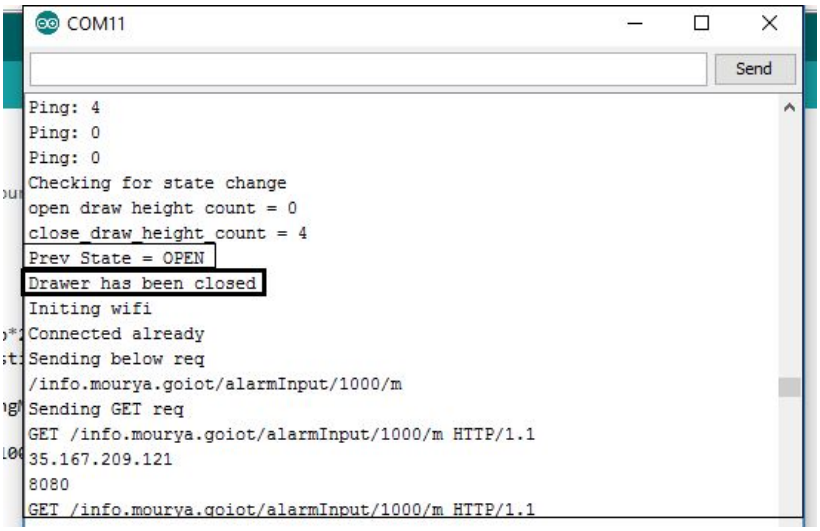
  - Draw open close Logs → Arduino



  - Push button press White LED glowing



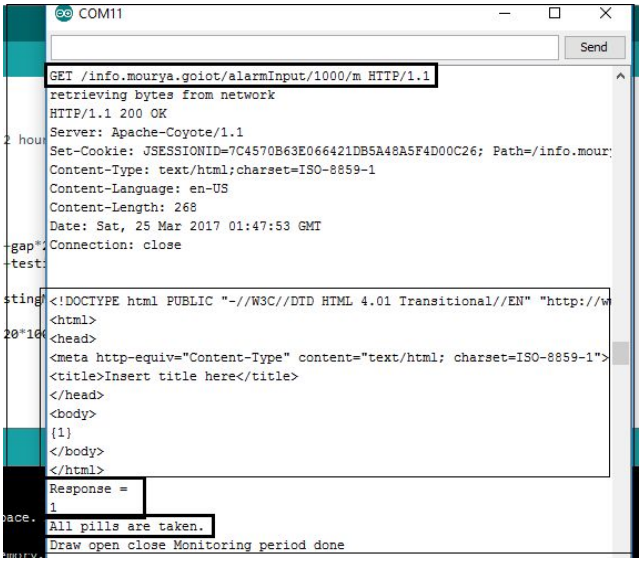
  - Draw close -> Arduino



- API output

{1}

- All pills taken -> Arduino



- Not taken
  - Button not pressed→ Adherence

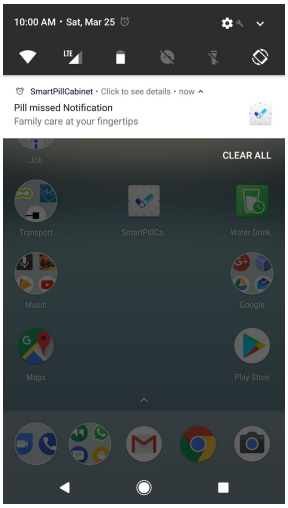
Medicines

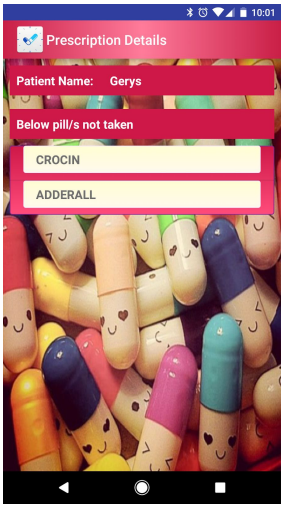
Medicine Name	Morning	Afternoon	Night	Pill Id	% Remaining
crocin	1	0	1	10	70
adderall	0	0	1	13	100

Adherence

Medicine Name	Morning	Afternoon	Night	Pill Id
crocin	0	0	0	10
adderall	0	0	0	13

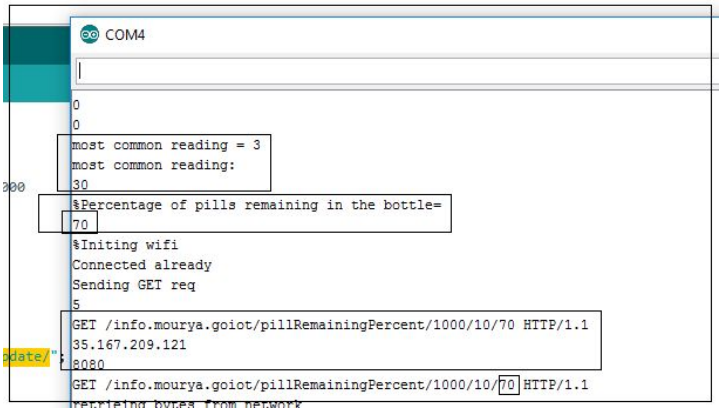
- Pill not taken Family notification Screenshot -> Android



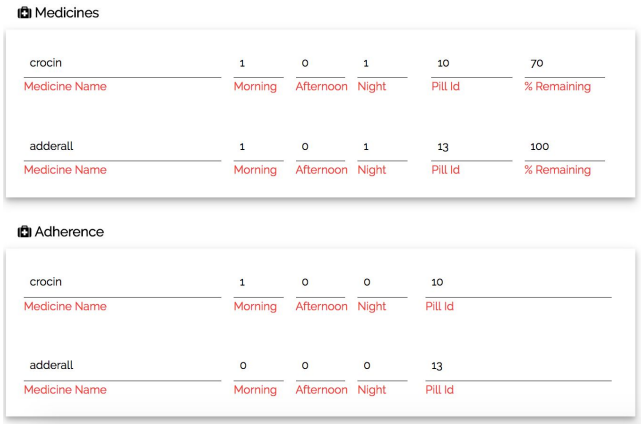


Pill Refill

- If the pill bottle has pills ***greater*** than 30%:
  - Pill remaining percent Arduino Log

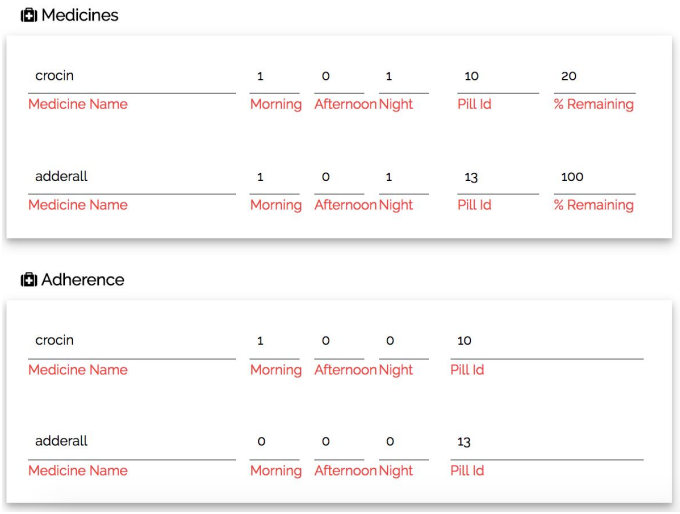


- Web portal screenshot of updated (pill remaining) value

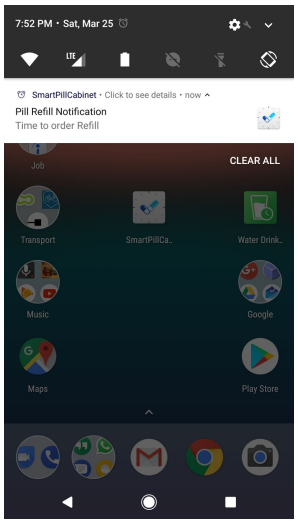


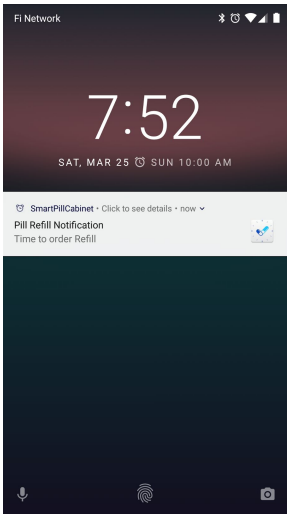
- If the pill bottle has pills ***lesser*** than 30%:
  - Pill remaining percent Arduino Log

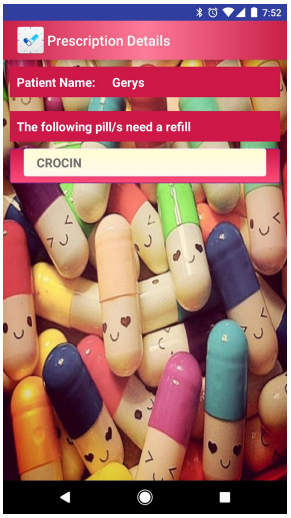
- Web portal screenshot of updated (pill remaining) value



- Pill Refill Notification







Costs

The IoT product with features of pairing itself to the web, and solving three major purposes of pill reminders, adherence monitoring & pill refill reminders is new to the market. Monthly pill organizer that works on cellular network[4], pill reminder boxes that pairs to the cellphone by Bluetooth[5], pill reminding watches[6] are some of the products currently available which costs $40-$60/month rentals, $75 and $100 respectively. The smart pill cabinet that is designed in this project costed $20, and the pill bottle costed $15. On an average if two pill bottles are sold to a customer, the expenditure would be **$50** with a pill cabinet($20) and $30($15 x 2 pill bottles), which is way cheaper compared to the other products.

CONCLUSIONS

Lessons Learnt and Findings

ESP8266 related

- ESP8266 Timers run ISRs (Interrupt Service Routines).
    - These ISRs needed to be lightweight as doing a lot of stuff in the callback would block the main thread and would reset the chip.
    - Limited number of timers (only 7).
- The wary behaviours of sensors. Actual readings need a lot of manipulations.
    - The depth sensor gives variable readings and we need to take multiple readings and estimate a heuristic like the mode of values. Ignore the impossible values (like 0) etc to actually get a good estimate of the correct reading.
- LCD implementation.
    - One of them did not glow due to problems with the hardware. We had to replace it. And then we ordered 2 more of different make. We could get the backlight to glow for one of them after adjusting the contrast using the potentiometer. Finding the address at which the device works was a challenge. We scanned for all addresses possible and found the device to be working at 0x27. But could not get the text to be displayed even after finding the right address. We have found many of them complaining about the device not working online too and tried all the suggested solutions.[18][19][20][21][29] Tried with different libraries, changing pin configurations, device address etc.,. But none of them worked.

Web Portal related

- Accessing static resources using spring.
- Iterating object with objects use JSTL jar.
- Placing new jars in lib folder, will be useful while exporting war file.
- Storing date format in database by using sql date and calendar.
- Keeping track of SQL changes.
- Extreme time delays caused because of misunderstanding between DB2 & MySQL.

Web Design related

- CSS/Javascript file inclusions.
- Tomcat listening only to 8080 port.

Future Work

- Extend the functionality to other forms of medicine such as syrups.
- Extend functionality to measure number of pills removed each time to enhance accuracy in reminders for pill refill, better adherence monitoring and also provide a better control over the actual number of pills to be taken.
- Secure access for APIs through unique user key.
- Mobile Application for doctors to help improve interaction and provide better care.
- Extend users/doctors to set timings for their pills, instead of the preset timings (Morning: 8am to 10am, Afternoon: 1pm to 3pm, Night: 7pm to 9pm) for taking pills.
- Use wearables fitted with sensors and bluetooth technology to monitor and log information of the user's health.

REFERENCES

[1] http://www.detoxrehab.org/3-in-5-u-s-adults-take-a-prescription-drug/

[2]https://www.washingtonpost.com/news/to-your-health/wp/2015/11/03/more-americans-than-ever-are-taking-prescription-drugs/?utm_term=.cf430a6de698

[3] http://www.cancer.net/blog/2016-09/tips-managing-multiple-medications

[4] https://www.medminder.com/?gclid=CI3VrOPQ8dICFRSUfgodIl4G2A

[5]http://www.target.com/p/tricella-bluetooth-smart-pillbox-with-family-notification-7-day/-/A-50716177?ref=tgt_adv_XS000000&AFID=google_pla_df&CPNG=PLA_Health+Beauty+Shopping&adgroup=SC_Health+Beauty&LID=700000001170770pgs&network=g&device=c&location=9032151&gclid=COuFmtrU8NICFU5gfgodhIIKRg&gclsrc=aw.ds

[6] http://www.epill.com/medicalwatches.html

[7] http://www.micropik.com/PDF/HCSR04.pdf

[8] https://www.w3schools.com/w3css/tryw3css_templates_analytics.htm

[9] https://developer.android.com/reference/android/content/BroadcastReceiver.html

[10] https://developer.android.com/training/scheduling/alarms.html

[11] https://developer.android.com/guide/topics/ui/notifiers/notifications.html

[12] https://developer.android.com/training/notify-user/build-notification.html

[13] https://developer.android.com/studio/projects/templates.html

[14] https://developer.android.com/guide/topics/ui/layout/listview.html

[15] https://developer.android.com/reference/java/net/HttpURLConnection.html

[16] https://developer.android.com/reference/android/Manifest.permission.html#INTERNET

[17] https://developer.android.com/reference/android/os/AsyncTask.html

[18] https://arduino-info.wikispaces.com/LCD-Blue-I2C

[19]http://arduino.stackexchange.com/questions/6782/i2c-lcd-serial-interface-board-not-displaying-text-wrong-pins

[20]https://www.losant.com/blog/how-to-connect-lcd-esp8266-nodemcu

[21] http://forum.arduino.cc/index.php?topic=206035.0

[22] http://www.switchdoc.com/2015/10/iot-esp8266-timer-tutorial-arduino-ide/

[23] http://www.micropik.com/PDF/HCSR04.pdf

[24] http://playground.arduino.cc/Code/NewPing

[25] https://www.arduino.cc/en/Reference/Tone

[26] https://techtutorialsx.wordpress.com/2016/05/07/esp826-controlling-a-buzzer/

[27] http://www.esp8266.com/viewtopic.php?f=29&t=6007&start=5#sthash.TJmGB90D.dpuf

[28] https://forum.arduino.cc/index.php?topic=212146.0

[29] http://imaginen4tion.blogspot.com/2015/10/how-to-connect-esp8266-to-lcd-via-i2c.html

APPENDICES

Execution Instructions

1. ESP8266
    - Install Arduino IDE.
    - Open the code from the attached folder and upload to ESP8266.
    - Setup hardware as per Figure 10 and 11.
    - Open Serial Monitor to observe logs.
2. Web portal
    - Login to the Linux instance using pem file attached.
    - Install Tomcat7 in the server.
    - Place the war file attached in the var/lib/tomcat7/webapps directory
    - Restart the server to reflect the changes.
    - Go to the link - <IP address>:8080/info.mourya.info/Login.html to access the portal.
3. Android application
    - Install Android Studio
    - Download the code from the attached folder.
    - Run the application on the emulator or an android phone

Team

We are team **goIoT.** Below are the dynamics of the project by the team members:

Lakshmi Shankarrao

- Hardware research and concept.
- Implement firmware for ESP8266 micro controller and sensor interface.
- Android Application development.
- Finished Product Design and Packaging.

Varun Srinivasan

- Finished Product Design and Packaging.
- Web Portal design
- Project deployment on EC2

Sri Mourya Dommeti

- DB level design and implementation
- J2EE Application with Spring MVC architecture
- Web APIs implementation