Implementation of Smart Health Adherence Tool using NFC Communication

Akib Islam
Electrical and Computer Engineering
University of Waterloo
Waterloo, ON, Canada
a54islam@uwaterloo.ca

Rupakeerthana Vemulapalli
Electrical and Computer Engineering
University of Waterloo
Waterloo, ON, Canada
rvemulap@uwaterloo.ca

Mahsa Javadi
Electrical and Computer Engineering
University of Waterloo
Waterloo, ON, Canada
mjavadi@uwaterloo.ca

***Abstract*—Approximately 40-50% of all patients struggle with medication nonadherence for chronic conditions such as diabetes/hypertension [1]. Patients must monitor their medication adherence by understanding trends and patterns in medication intake. In being equipped with a greater level of knowledge, they will be able to cultivate better habits. Currently, there exist solutions that monitor and track medication adherence, but they cannot connect to other wireless devices and do not offer functionality to notify patients when they fail to take medication at their prescribed time. In this paper, we present a way to track patient adherence and alert the patient if they are at risk due to medication nonadherence based on their level of deviation from the prescribed time and other metrics. By using RFID/NFC communication to transfer data from a patient's unique blister pack to a server, we will analyze key metrics such as date and time of consumption. We will further use advanced algorithms to check important conditions such as a maximum 1-hour deviation from the prescribed time of medication intake, and low inventory to send a notification to patients to take their medication. Further, Adherence for this paper is defined as taking the pill within 1 hr of** the medication prescribed time

INTRODUCTION

Medication adherence is a common struggle amongst patients. While doctors consider it the patient's responsibility and others are even unaware of the existence of such issues, patients struggle with numerous barriers such as lack of motivation, depression, cognitive impairment, and many more. As a result, society suffers from 100,000 preventable deaths and $100 billion in preventable medical costs each year [1]. Older people and those who suffer from physical/mental conditions are more vulnerable to skipping their medicines. Further, this makes it hard for doctors to manage and monitor the well-being of their patients. And so, this results in increasing the disease severity, and in some sensitive cases when the medicine is not consumed on time, can lead to mortality.

Barriers to medication adherence can be specifically broken down into patient, treatment, and other areas according to this research paper. Regarding patient-related barriers, in addition to the ones mentioned previously, patients can also suffer from denial, drug/alcohol abuse, cultural issues, and low educational level. Treatment-related barriers include the complexity of treatment, side effects/fear of side effects and inconvenience, cost, and time. Other barriers include poor doctor-patient relationships and asymptomatic disease being related [1].

One of the most optimal ways to improve adherence to lift patients from these barriers is through improving medication adherence. This can reduce the risk of unintended consequences for patients. Currently, 7-day pill organizers and another related pill counting, and sorting trays are used. However, users of this tool struggle with the same issues. Additionally, older people and those who suffer from physical/mental conditions find it most difficult to follow a schedule and organize medications when empty. Although patients are constantly reminded of their medication intake by doctors and pharmacists, they continue to suffer from some of the above-mentioned barriers and fall negligent to the expectations. And so, patients struggle with holding themselves accountable. They need to be equipped with the knowledge to understand their medication adherence and realize it early so they can change their behavior.

The objective of this paper is to present a smart blister pack that can inform patients about their medication intake history by tracking their daily consumption by recording the timestamp of their medicine intake. By providing the patients with this information regarding their deviation of intake from recommended dates/times provided by the pharmacist, the project aims to improve their daily medication intake habits. By doing so, the goal of the project is to improve the well-being of the patient and reduces their risk of serious medical episodes. The paper also describes the methodologies of informing the patient by sending them phone notifications if there is non-compliance to the medications and preventing serious medical episodes. The key information about the medication intake can also help the system track the inventory of the patients and alert the patient if they are running low on medicines. The user will also be provided with a dashboard that contains an analysis of their medicine intake and deviation from their schedule.

BACKGROUND

Raspberry Pi

This small device which is considered a tiny computer is used for different purposes in the home and in the industry ranging from industrial automation to controlling the utilities at home. Also, individuals use this pi for programming electronics as well as learning codes. The operating system for this small device is Linux. Also, to manage electronic parts, a set of pins are implemented. Raspberry pi has different models which were introduced over the years. Generally, it has some models which are pi zero, pi400, A, and B. A and B models have different categories. Raspberry pi A models are cheaper than the B models [2]

MQTT

There are different messaging protocols and MQTT is one of them. MQTT is a publish-subscribe-based protocol that needs a TCP connection for exchanging information. The architecture which is used in this messaging protocol is a client-server. In figure 1 we can see the architecture for MQTT. In this protocol information can be shared within groups which means in a group, members get data from different parts who send the information (many to many communication)

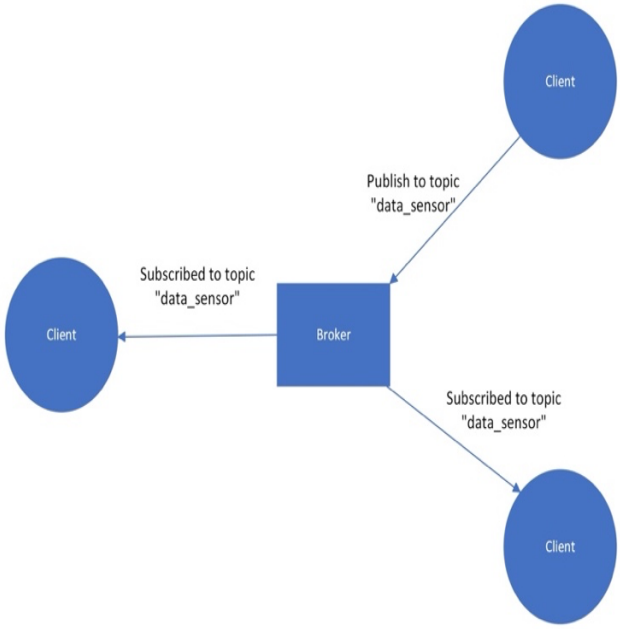

Fig 1 - MQTT Client-Broker Architecture

Here the role of the client and broker is that the broker provides communication among clients, receives information from the client, and then distributes it to the client who has subscribed to receive messages. There are some responsibilities for the broker such as getting all the messages which are sent by the client, analyzing them, and then figuring out the clients who subscribed to the message to send the message to them. In terms of security, to associate with clients, brokers might require authentication keys from them. [3]

MQTT is appropriate for restricted environments. In this messaging protocol, the size of the message is undefined and small. Also, the methods which are used in this messaging protocol are: subscribe, unsubscribe, publish, close, connect, and disconnect. This messaging protocol has three levels of quality of service such as at most once (QoS 0), at least once (QoS 1), and exactly once (QoS 2). The messages in MQTT are distributed to the topics, so the clients who subscribed to the topics can get the data that are distributed to those topics. There are some pros to MQTT, some of which are scalability, Time decoupling, Space decoupling, and synchronization decoupling. [4]

NFC/RFID

NFC and RFID are tracking innovations that are used in a variety of fields. In RFID which has a reader, an antenna, and a tag, radio waves are employed for recognizing objects, and tags or cards are used for data storage. This technology unlike NFC can work at larger distances and can scan a huge variety of tags at once. Also, for doing its job it does not need the user to do some actions. How the communication is done in RFID is through the tag to the reader. In the NFC which stands for Near Field Communication, there is a tiny chip that is a tag, and this tag has data storage capability, which can store various formats such as media, text, and URLs. The NFC tag can be scanned by electronic devices like most phones but at a short distance of around 4 inches (2.54 cm). In this wireless technology for tasks like payment to be completed users must be involved and do some actions. This technology can be used in various fields such as banking transactions like contactless payment, at universities like student cards, etc. [5]

CLOUD PLATFORM

A cloud platform is a platform that provides storage for sharing and accessing data over the internet without storing information on the local computer. In a cloud system, the back-end part is responsible for providing the servers and system for the front-end users. As many users need to use the cloud platform, the companies who provide this technology need to be able to handle this number of clients and their data and use fewer physical systems, so they can use a method called server virtualization to tackle this issue. Figure 2 shows the general overview of cloud computing [6].
Cloud technology has different categories such as public cloud, private cloud, and hybrid cloud [7]. Thingspeak is a cloud platform for IoT devices. These devices take information from the environment using sensors. The information generated from these devices can be sent and stored in Thing speak. There is a possibility of retrieving those data which is stored in Thing speak. These data can be processed and visualized to develop applications. In Thing speak without any requirement of server development and software development, the IoT systems can be constructed [8].

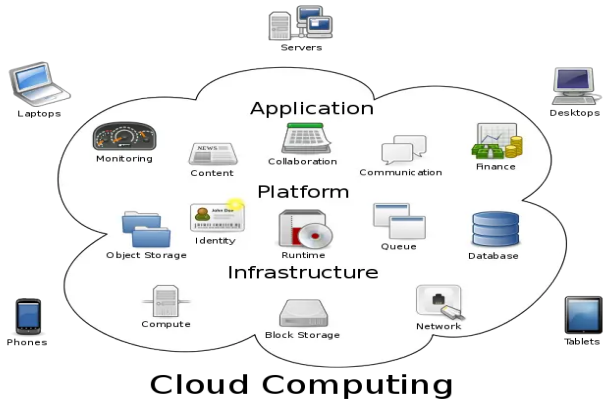

Fig 2 – Cloud Computing Architecture

LITERATURE REVIEW

There are lots of methods suggested to solve the problem of measuring and reinforcing adherence to the prescribed medication schedule. The MEMS Cap ecosystem claims to be one of the best in terms of innovation. The product utilizes a reusable blister medicine packet that contains a microcircuit that can detect the removal of the pill and records the timestamp. It supports up to 4000 removals of pills from the blister pack.

The product consists of a small LCD to display key information such as doses taken the previous day and the timestamp of the last taken medicine. The reusable blister medicine pack doesn't require any connectivity and lacks a communication stack or modules. The data can be transferred wirelessly from the reusable blister to the reader using Near Field Communication (NFC). Their architecture consists of small non-volatile memory to store the database of the timestamp logs and product life ranges from 18 to 36 months. Figure 3 shows the MEMS Cap ecosystem Helping-Hand product [9].

Fig 3 – MEMS Cap Helping-Hand product

Health beacon [10] is another application-based platform that helps patients in adhering to their medication schedules. The software application provides services that make the life of patients easier by providing them customized notifications, sending them SMS as their medication timing approaches, showing them the schedule of their medicine, and providing their health records and real-time analytics of their health history. The best feature of this application is that it reminds patients to take their medicine by sending them SMS on their prescribed medication timing. However, it might cause too many notifications every day even when the patient consumed medicine on time.

This application, however, lacks the functionality of integrating with actual smart blister medicine packets. Due to this drawback of limited functionality, there is no way that the application can know whether the user has taken the medicine on time or not. It reminds the user to take their medicine if they forget to take the medicine on time, but it might not be useful for the patients who occasionally forget to take the medicine on time. The Artificial Intelligence technology can keep track of patients' adherence schedules by analyzing the patient data and can provide customized behavior plans to help the patients stick to the proper adherence behavior. Figure 4 shows the health beacon application.

Fig 4 – Health Beacon smartphone application

Spencer [11] is a smart medication hub that contains basic and advanced features for in-home medications. The device is a bulky and heavy device, unlike the smart medicine blister pack which is light weighted. The device contains the technology of unpacking the medicines and putting the medicines in the compartment inside the device. The device is connected to the cloud platform for the exchange of data such as patient information, patient medication schedule, patient medications, timestamp logs of patient's medicines, etc. The device comes with a TFT touch screen which can take inputs from the user. The device shows key information such as patient medicine schedule, doctor appointments, medicines remaining in the compartments, etc.

The device can be connected to the spencer cloud using Wi-Fi. The device also facilitates online consultation with the doctors through Voice over IP (VOIP) and video call facility. The spencer smart-hub device can be connected to various peripherals like ECG electrodes, Blood Oxygen monitoring devices, etc., and can record data from various peripherals. The spencer mobile application can fetch all the patient's information from their server and present insights to the patients on their smartphones. The drawback of spencer is that it cannot be carried along with the patient due to its size and weight. The platform claims that 97%of users were able to adhere to the prescribed medication schedule after using their platform and their data shows more than 90% patient engagement rate. Figure 5 shows the spencer smart medication hub.

Fig 5 – Spencer smart medication hub

Design and Implementation

The design of the smart health adherence tool consists of two major parts – A hardware device to record data and a cloud platform to process the data. The actual hardware device consists of Smart Blister which includes an event-detection circuitry, low-power microcontroller, and NFC Reader to read the NFC data. The event-detection circuitry is designed in such a way that whenever there is s removal of the medicine from the smart blister label, the circuitry will detect that event and log the timestamp in the flash memory of the microcontroller. The flash memory will maintain a database of timestamp logs which provides key information about the time at which the circuit breaker detects the event. In our proposal, Raspberry pi flash memory is used to capture this database. Once the smart blister medicine label is returned to the pharmacy, the pharmacist can get the patient's medication time logs by scanning the blister label using RFID/NFC scanner.

The blister microcontroller will fetch the database from the flash memory and transfer this data to the RFID/NFC reader. In our implementation, we used an NFC tag to transfer the information from the Raspberry Pi flash memory which contains the database. The NFC tag will be programmed with the required information such as patient id, patient medicines logs, patient name, patient age, etc. This NFC tag will be scanned at the PN532 NFC reader which is connected to another Raspberry pi. The second raspberry pi would capture the patient information and database from the NFC tag and store it as a CSV (Comma Separated Value) file in the filesystem of the internal flash memory.

The application source code will read from the CSV file and based on the user identification from the NFC tag, it will parse the CSV file, process the patient medication database, and send the information to the Thing speak Cloud Platform for further analysis using TCP/IP protocol. Raspberry Pi is connected to the internet via WIFI and connected to the Thing speak cloud platform utilizing the Message Queuing Telemetry Transport (MQTT) protocol. The raspberry pi will publish its data to a specific topic and the Thing speaks channel will subscribe to that topic to fetch the information from the device.

In this implementation, we created three topics – day, hours, and minutes and the device will publish the day of taken medication, hours of taken medication, and minutes of taken medication to these topics respectively. The dashboard of the Thing speaks cloud will plot these data points and show the trend of user medication adherence. The raspberry pi application code is also equipped with logic to inform the user if they fail to take the medicine within one hour of the prescribed time. The user will be notified using SMS if they miss taking their medicine on time. A similar SMS facility is used to inform the user if their inventory of medicines becomes low.

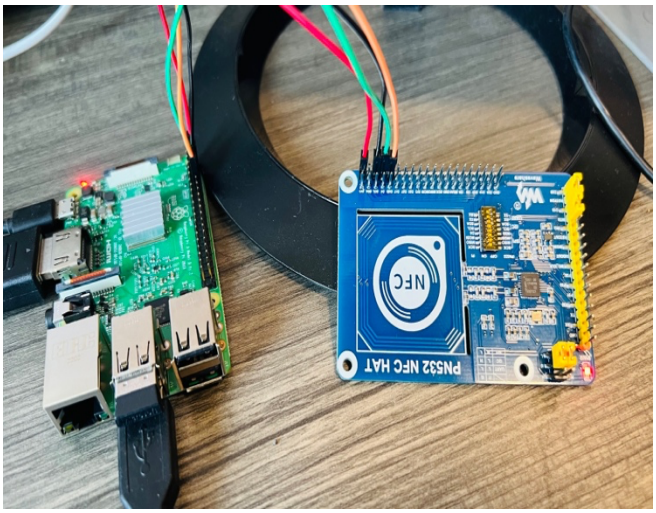

Fig 6 - Interface of Raspberry pi with PN532 NFC reader over UART

Figure 6 shows the NFC reader setup and actual connection of the raspberry pi to the NFC reader over the Universal Asynchronous Receiver Transmitter (UART) interface. The circuit diagram of the connection is shown in the following figure.

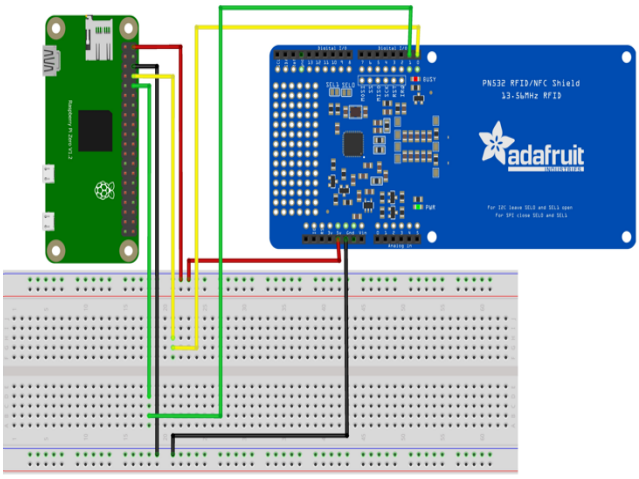

Fig 7 - Connection diagram of raspberry pi with PN 532 NFC Reader

Figure 7 shows the connection diagram between the raspberry pi and PN 532 Near Field Communication reader module. The PN532 NFC module can connect to the microcontroller over Serial Peripheral Interface (SPI), Universal Asynchronous Receiver Transmitter (UART), or Inter-Integrated Circuit (I2C) communication interface. In our experimental setup, we utilize the UART interface to interface the PN532 module to the raspberry pi due to the low pin requirement over the UART interface. UART communication protocol needs only two pins – Transmitter (TX) and Receiver (RX) pin to establish communication between PN532 and Raspberry Pi. The TX pin of the raspberry pi would be connected to the RX pin of the PN532 NFC module, and the RX pin of the raspberry pi would be connected to the TX pin of the PN532 NFC module.

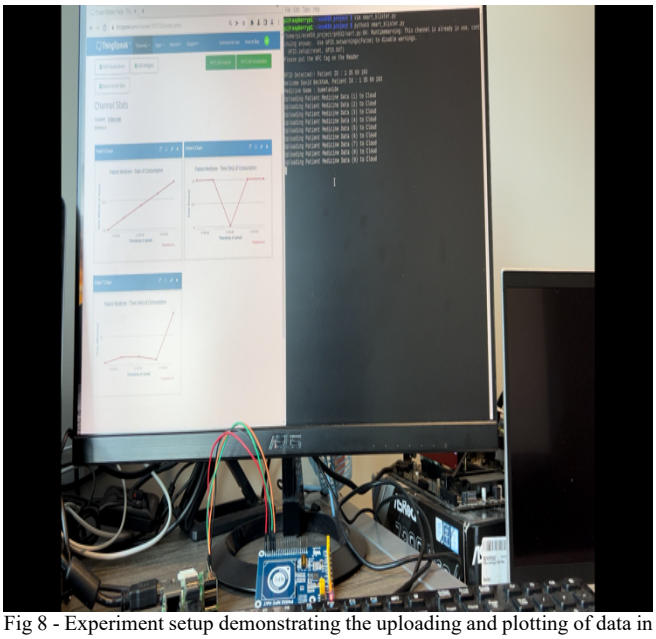

Fig 8 - Experiment setup demonstrating the uploading and plotting of data in Thingspeak cloud

Figure 8 shows the actual hardware setup and the results obtained from the project.

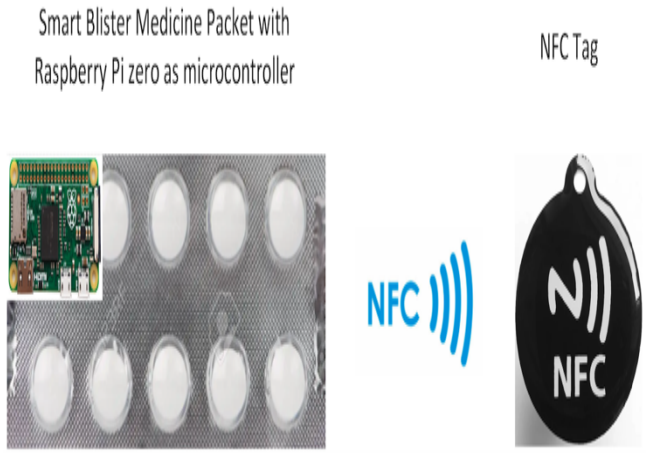

Fig 9 - Programming the NFC Tag with patient medicine log

Figure 9 shows the communication flow between the smart blister medicine packet and the NFC tag. The smart blister medicine packet consists of compartments with a circuit breaker and raspberry pi zero. The medicines are packaged inside the compartments. When the user opens the compartments, the circuit breaker detects this event and sends the message to the raspberry pi zero about the event.

The pi zero fetches the date and time information from the Real Time Clock (RTC) peripheral and logs the event in the flash memory of pi zero. This database of logged timestamped events can be transferred to the NFC tag by bringing the unique NFC tag near to the NFC reader connected to pi zero. The pi zero will program the NFC tag with the required information.

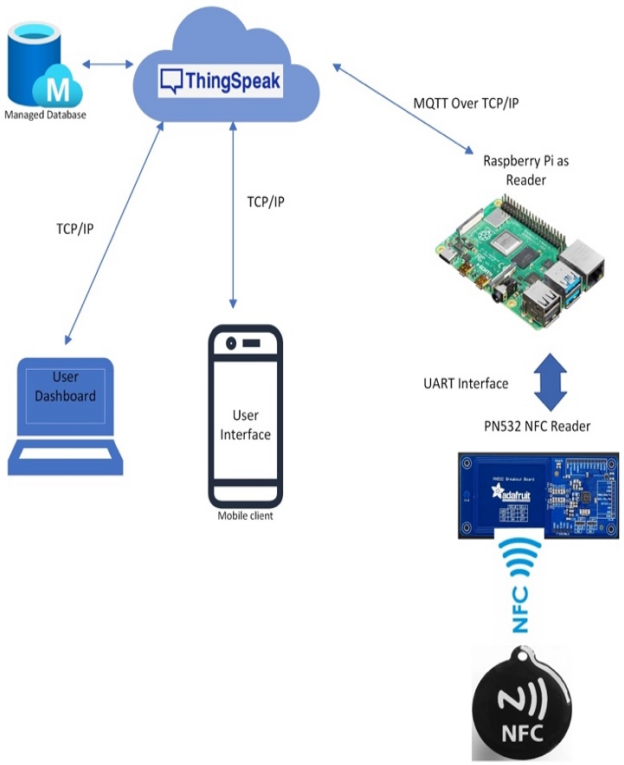

Fig 10 - Architecture of Smart Health Adherence Tool

Figure 10 depicts the architecture of the Smart Health Adherence Tool. The three major blocks of the architecture are cloud platform, hardware devices and application code, and communication stack. Once the NFC device is programmed with the required information, the patient information can be transferred to the Raspberry pi by tapping the NFC tag on the PN532 NFC Reader connected to Raspberry pi 3 over UART Interface. The application code of Raspberry pi would facilitate the transfer of patient data from the NFC reader to its internal flash memory. The contents of the raw patient data would be parsed and stored as a Comma Separated Value (CSV) file in the internal flash memory. The application code contains the logic to parse this CSV file and convert it into the format which is required by Thing Speak Cloud Platform Post Application Programming Interface (API). The raspberry pi would publish the patient data on three different topics namely “day of medicine taken”, “hours of medicine taken” and “minutes of medicine taken”. These 3 topics would be created for each smart blister medicine packet. The day extracted from the raw timestamp would be published on the topic “day of medicine taken”, hours extracted from the raw timestamp would be published on the topic “hours of medicine taken” and minutes extracted from the raw timestamp would be published to the topic “minutes of medicine taken”. The Thing Speak cloud platform would subscribe to these three topics. Whenever there is a new value pushed to any of these subscribed topics, the Thing speaks cloud platform would pull those values and plot the values in the graph on the dashboard. The user or pharmacist can visualize the dashboard, graph, and analytics of the adherence to the medication by logging into the Thing speak website. These analytics can be viewed either on the PC or using any smartphone browser.

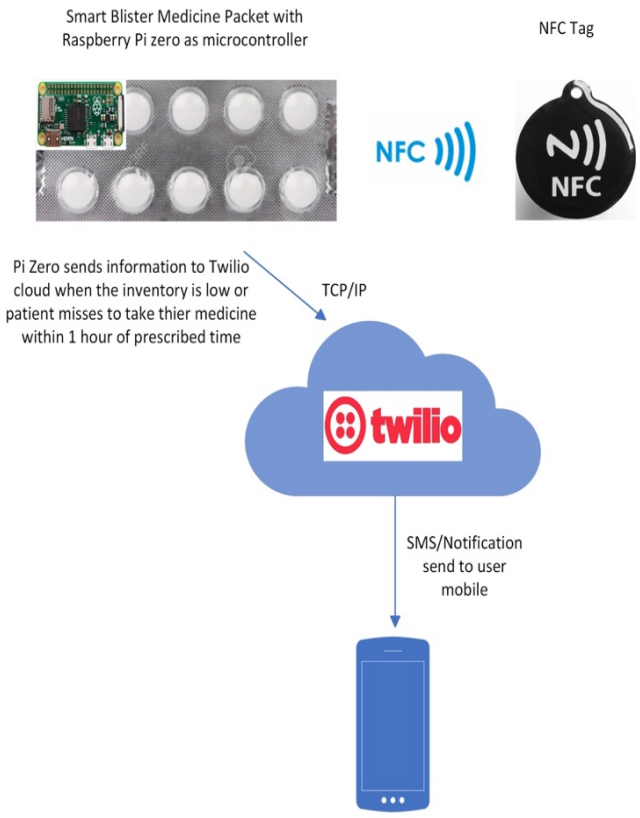

Fig 11 - Architecture of notification mechanism to the user client

Figure 11 represents the architecture of the notification mechanism to the user client. The pi zero microcontrollers on the smart blister medicine packet keep track of the inventory of the medicines and patient medication schedule. If the inventory of the medicines is below the threshold value, the pi zero sends this low inventory request to the Twilio platform. The threshold value is decided in such a way that the medication inventory will reach zero in 72 hours. The raspberry pi zero needs to be connected to the WIFI to send information to the Twilio server. The pi zero can be dynamically programmed to automatically establish the connection to Wi-Fi only when there is a need to send the request to the Twilio server thus saving energy and battery life. Once the information is sent to the Twilio server, the connection to Wi-Fi can be disconnected. The raspberry pi zero also needs to send information to the Twilio server when the user doesn't consume the medicine within 1 hour of the medication schedule.

The raspberry pi zero would follow a similar mechanism of dynamically establishing the connection to the Wi-Fi during the event of sending a message to the Twilio server [12] and disconnecting from the Wi-Fi after the information is successfully sent to the server. The Twilio server would be responsible to send the notification and SMS to the registered user's mobile. The pi zero uses HTTP Post request to send messages to the Twilio server using cURL. After sending the request to the Twilio server, raspberry pi zero monitors the status of the posted message, and when the message is successfully delivered, raspberry pi zero would disconnect the Wi-Fi. Twilio uses the Amazon Web Service (AWS) cloud and provides API to integrate the HTTP and PSTN (Public Switched Telephone Network) to reliably send information to the registered user's mobile.

EXPERIMENTAL WORK TO DEMONSTRATE THE WORKING AND PERFORMANCE OF THE SOLUTION

The previous section described the overall architecture and data flow mechanism of the Smart Health Adherence Tool. In this section, we will closely look at the software implementation of the application code.

```
if __name__ == "__main__":
    nfc_obj = PN532_UART(debug=False, reset=20)
    nfc_obj.SAM_configuration()

    patient_id_list = []

    print("Please put the NFC tag on the Reader \r\n")
    tag = False

    while tag == False:
        tag_uid = nfc_obj.read_passive_target(timeout=1)
        if tag_uid is None:
            tag = False
        else:
            for i in tag_uid:
                patient_id_list.append(i)
            tag = True

    patient_id = ' '.join(map(str,patient_id_list))
    patient_id.replace(' ','')


    patient_med_database = []

    print("RFID Detected!! Patient ID : {}".format(patient_id))
```

Fig 12 - Source code for detecting the PN532 RFID tag

The code in figure 12 establishes the logic to detect the PN532 RFID tag. The RFID reader is connected to the raspberry pi through Universal Asynchronous Receiver Transmitter (UART) protocol. The initial line of codes establishes the UART connection by turning on the required clock to the UART peripheral, adjusting power modes, configuring the GPIO pins in Alternative Configuration Mode, etc.

It also initializes the NFC Reader module and writes boot-up configurations to the module. The next line of codes tries to read the NFC tag and wait for the NFC tag to appear in the communication range of the NFC reader. Once the NFC reader detects the tag, it reads the information from the tag. The code inside the while loop waits for the NFC tag to

appear and reads the information from the NFC tag. Once the information is read, it is stored in the data structure in stack memory for further processing

```
if patient_id == patient1_static_id:
    print("Welcome {}, Patient Id : {}".format(patient_name[patient_id], patient_id))

    #Read the patient1 csv file to record the medicine timings
    excel_data = Path("/home/pi/Downloads/David Beckham.xlsx")
    excel_obj = openpyxl.load_workbook(excel_data)
    worksheet = excel_obj.active

    for i in range(0, worksheet.max_row):
        count = 0
        col_date = 0
        col_time = 0

        for columns in worksheet.iter_cols(1, worksheet.max_column):
            if(count == 0):
                col_date = columns[i].value
                count += 1
            else:
                col_time = columns[i].value
                count += 1

        cur_date = col_date.date().strftime("%d/%m/%Y")
        cur_time = col_time.isoformat()
        tmp_list = [cur_date, cur_time]
        patient_med_database.append(tmp_list)

    print("Medicine Name : {}".format(patient_medicine[patient_id][0]))
```

Fig 13 - Source code for parsing the CSV file

Figure 13 depicts the logic to read the comma-separated values (CSV) file and parse the fields. The code first opens the file and loads the file in the RAM. Then, it iterates row by row capturing all the column fields in the columns object. The column object would then contain a structure of all the column entries which are then read sequentially and stored in the local variables named cur_date and cur_time. The logic would then build the list of these local variables to capture all the row and column values.

```
thingspeak_channel_id = 1810732
thingspeak_write_api_key = "NM6ORTE3U5PIUVX7"

#Upload the patient database to thingspeak cloud
channel_obj = thingspeak.Channel(id=thingspeak_channel_id, api_key=thingspeak_write_api_key)

count = 0
for i in patient_med_database:
    count += 1
    print("Uploading Patient Medicine Data ({}) to Cloud".format(count))
    dd = int(str(i[0][0]) + str(i[0][1]))
    hr = int(str(i[1][0]) + str(i[1][1]))
    mi = int(str(i[1][3]) + str(i[1][4]))
    channel_obj.update({'field5': dd, 'field6': hr, 'field7': mi})
    time.sleep(30)
```

Fig 14 - Source code for formatting and uploading data to Thingspeak cloud

Figure 14 captures the unique channel id and writes API keys from Thing speak to write the correct channel with sufficient write privileges. Each channel created in the Thing speaks cloud would have a unique channel id and read and write API keys are bound to that channel id. The next line of code creates the channel object which would act as a unique handler for other Thing speak APIs [13]. From the local database, we parse the day, hour, and minutes information and store them in dd, hr, and mi local variables. These variables map to the topics in which we want to publish the data using the MQTT protocol. In the Thing speak dashboard, we subscribed to these topics from the unique channel id. The update function of the Thing speaks API is responsible for publishing these values to the respective topics using the Message Queuing Telemetry Transport (MQTT).


```
▼<channel>
  <id type="integer">1810732</id>
  <name>Smart Blister Pack</name>
  <description>Patient Dashboard for Smart Blister Pack Project</description>
  <latitude type="decimal">0.0</latitude>
  <longitude type="decimal">0.0</longitude>
  <field5>Date (Month of June)</field5>
  <field6>Time (Hours)</field6>
  <field7>Time (Minutes)</field7>
  <created-at type="dateTime">2022-07-25T02:12:34Z</created-at>
  <updated-at type="dateTime">2022-07-28T21:12:41Z</updated-at>
  <last-entry-id type="integer">48</last-entry-id>
 ▼<feeds type="array">
  ▼<feed>
     <created-at type="dateTime">2022-07-28T21:12:57Z</created-at>
     <entry-id type="integer">1</entry-id>
     <field5>1</field5>
     <field6>10</field6>
     <field7>5</field7>
   </feed>
  ▼<feed>
     <created-at type="dateTime">2022-07-28T21:13:27Z</created-at>
     <entry-id type="integer">2</entry-id>
     <field5>1</field5>
     <field6>20</field6>
     <field7>3</field7>
   </feed>
  ▼<feed>
     <created-at type="dateTime">2022-07-28T21:13:58Z</created-at>
     <entry-id type="integer">3</entry-id>
     <field5>2</field5>
     <field6>10</field6>
     <field7>8</field7>
   </feed>
  ▼<feed>
     <created-at type="dateTime">2022-07-28T21:14:28Z</created-at>
     <entry-id type="integer">4</entry-id>
     <field5>2</field5>
     <field6>21</field6>
     <field7>5</field7>
   </feed>
  ▼<feed>
     <created-at type="dateTime">2022-07-28T21:14:58Z</created-at>
     <entry-id type="integer">5</entry-id>
     <field5>3</field5>
     <field6>9</field6>
     <field7>8</field7>
   </feed>
  ▼<feed>
     <created-at type="dateTime">2022-07-28T21:15:29Z</created-at>
     <entry-id type="integer">6</entry-id>
     <field5>3</field5>
     <field6>20</field6>
     <field7>6</field7>
   </feed>
```


Fig 15 - XML Response from Thingspeak server validating the publishing of data to the topics

Figure 15 shows the XML response of the posted topic from the Thing speaks cloud platform. In this XML form, we can see that publishing values to all three topics were successfully done. The code waits for the successful response from the Things speak server before publishing the next set of values to the topics.

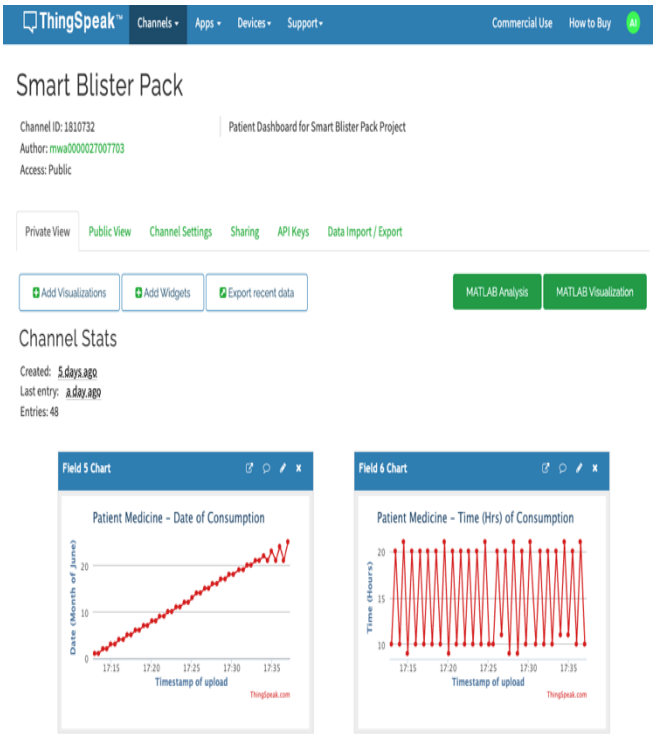

Fig 16 - User Dashboard of Thing Speak showing analytics and graphs of the patient data

Figure 16 shows the user interface of the things speak cloud platform where the graph and analysis of patient data are displayed along with some key insights about the adherence metrics.

```
SSD_TWILIO = 'AC523a7d74585d05d39c3abc96f1eaf6ce'
TOKEN_TWILIO = '6b4b21e3b438582b0deba1387673d0ef'
TO_NUM = "+12268811655"
FROM_NUM = "+13253264484"
```

```
num_medicine -= 1
if(num_medicine == 6):
    #Send text message to user about low inventory
    client_obj = Client(SSD_TWILIO, TOKEN_TWILIO)
    msg = client_obj.messages.create(to=TO_NUM, from_=FROM_NUM, body=msg_str)
    print("Low inventory Sending message notification to user \r\n")
```

Fig 17 - Source code to send SMS and notification to the patient's registered mobile

The code in figure 17 shows the integration of Twilio API to send the SMS to the registered patient's mobile device. The code requires configuration data such as SSD and Token information which can be obtained by creating the account on Twilio. The TO_NUM and FROM_NUM are the Twilio registered mobile number and patient mobile numbers respectively. The code creates the client object which can be used in the Twilio API and acts as a unique handle. The next line of code invokes the Twilio SMS API which is responsible for sending SMS to the patient mobile. The logic for sending the SMS to patient data is written such that when the number of medicines in the smart blister medicine packet reaches six, a notification in the form of SMS would be sent to the registered user's mobile phone.

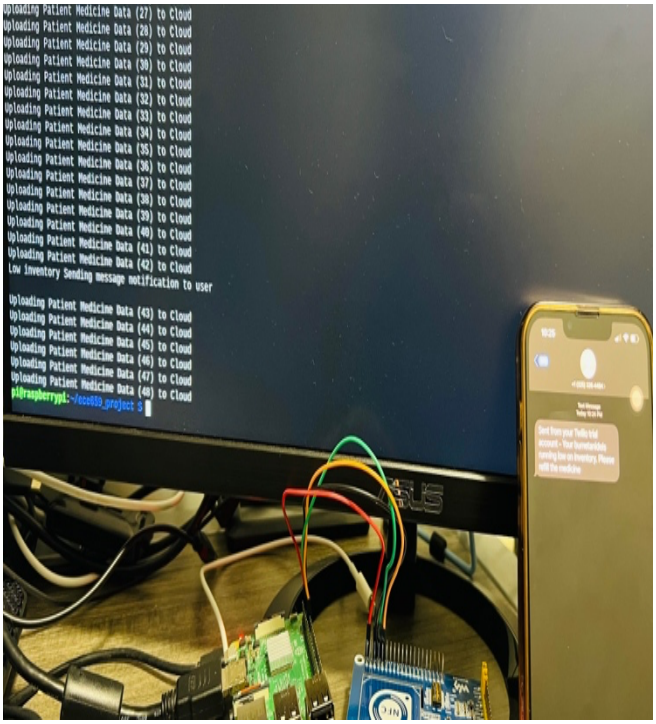

Fig 18 - Experiment setup demonstrating the sending of SMS on low inventory to patient's registered mobile

Figure 18 demonstrates the working of the Twilio notification mechanism triggering from the code when the inventory of the medicine in the smart blister medicine packet becomes low.

Recent Messages View all Message Logs

DATE	SERVICE	DIRECTION	FROM	TO	# SEGMENTS	STATUS
2022-07-31 2:29:41 UTC	–	Outgoing API	(325) 326-4484	(226) 881-1655	1	Delivered
2022-07-31 2:28:20 UTC	–	Outgoing API	(325) 326-4484	(226) 881-1655	1	Delivered
2022-07-31 2:24:53 UTC	–	Outgoing API	(325) 326-4484	(226) 881-1655	1	Delivered
2022-07-31 2:21:13 UTC	–	Outgoing API	(325) 326-4484	(226) 881-1655	1	Delivered

Fig 19 - Response from the Twilio server validating the successful sending of SMS to the user's mobile

Figure 19 shows the response of the Twilio SMS Send API from the raspberry pi source code. We can see that the SMS was delivered to the user on time without any significant delay. A similar notification mechanism is triggered by code when the user forgets to take their medicine within 1 hour of the prescribed time. Figure 20 shows the actual SMS received by the user when the inventory of the medicines in the smart blister medicine pack becomes low.

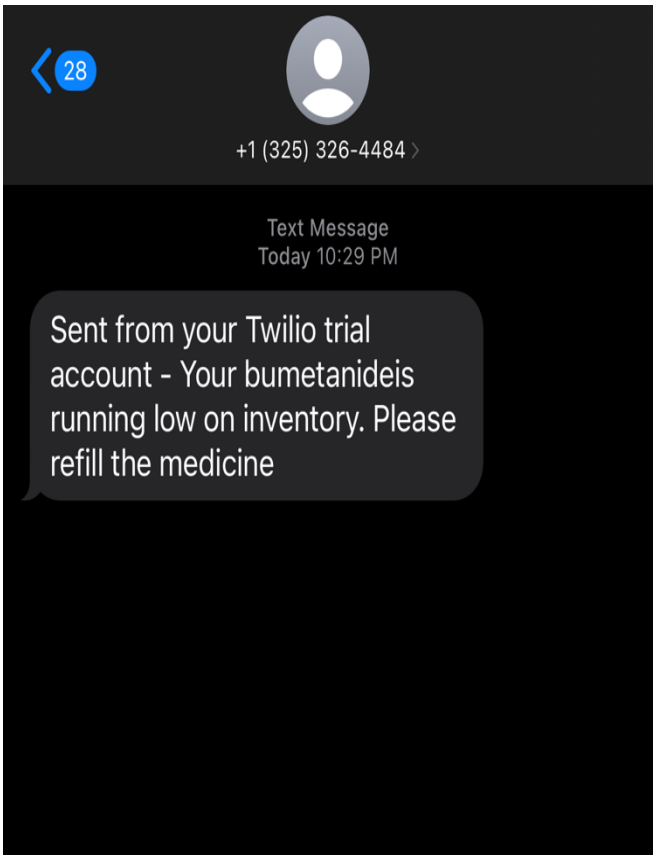

Fig 20 - User mobile receiving SMS on low inventory

RESULTS AND DISCUSSIONS

The graphs and analysis on the ThingSpeak dashboard [14] present some key information about the patient's adherence to the prescribed medication schedule. Figure 21 shows the date of the month vs the timestamp of the upload which shows the day of usage of medication. This graph depicts key information about whether the patient consumed medicine on the prescribed day of medication. Failure to skip medicine for two consecutive days can result in the development of negligence behavior and this could have a serious impact in the long run. To mitigate this risk, the pi zero on the smart blister medicine packet detects this deviation and sends a notification to the user.

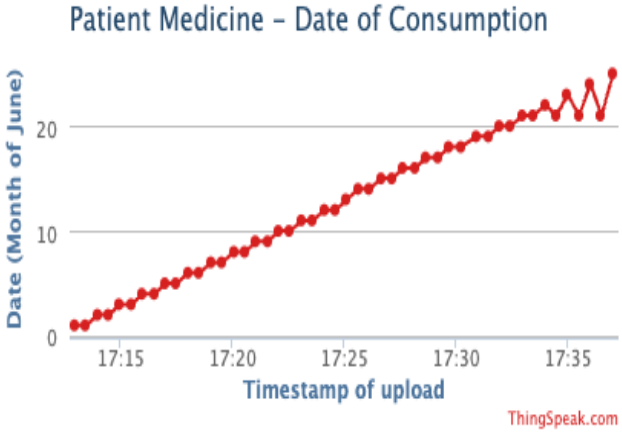

Fig 21 – Date of consumption data plot in Thingspeak cloud platform

Another important metric in determining adherence is the time at which the medication is taken by the patient. This information is very crucial in determining the efficiency of adherence. The time at which the medicine is taken is broken down into two components – the hour at which the medicine is taken and the minute at which the medicine is taken. The minutes' information presents key information about the strict adherence to the medication, although there is no notification being sent if the user doesn't adhere to the minutes. However, it is generally advisable to stick to a definite minute timeframe and consume the medicine in the same minute timeframe every day. Gathering the minute's information could help in positive reinforcement of this good behavior in the patient.

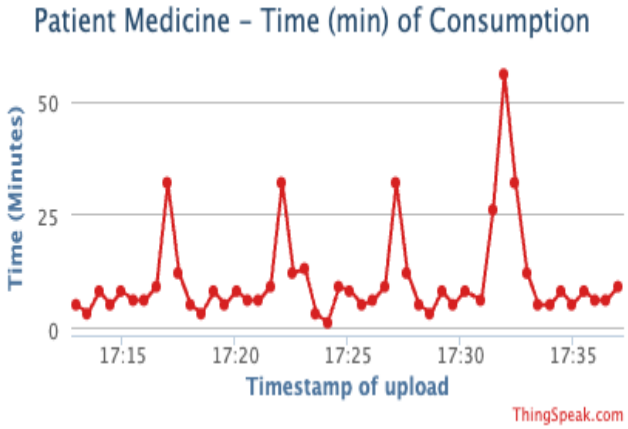

Fig 22 – Minutes data for consumption data plot in Thingspeak cloud platform

Figure 23 shows important information about hours of consumption of medicine which is a piece of key information in sending the notification to the user. If the user fails to take the medicine within an hour of the scheduled medication time, the pi inside the smart blister medicine packet would send the notification to the user's registered mobile number reminding the patient to take their medicine. The notification sent to the user would be logged in the database and this information would be presented in the dashboard suggesting the number of times the user needs to be reminded about their medication. If the user fails to take the medicine within one hour of getting the notification, it would be logged in the der "Severe category" in the database and this information would be presented in the dashboard.

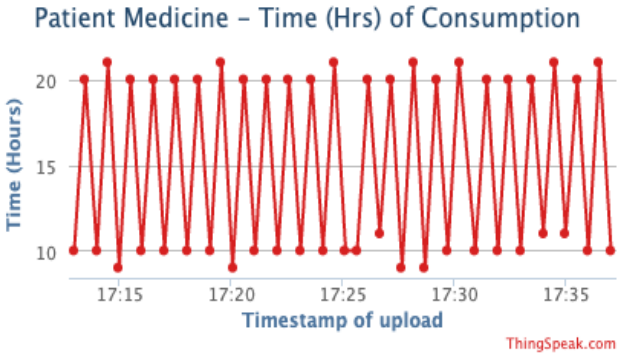

Fig 23 – Hours data for consumption data plot in Thingspeak cloud platform

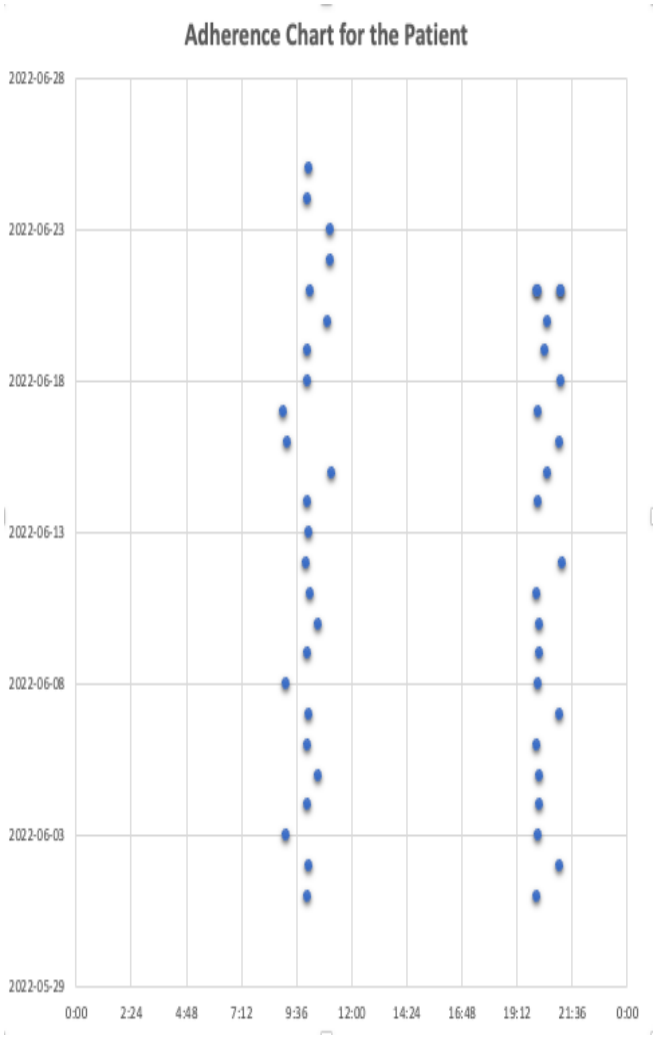

Fig 24 - Adherence chart of medication for the patient

Figure 24 shows the adherence chart of the patient medication usage. The patient was given the medication schedule of 24 days with prescribed medication timing as 10 am and 8 pm medicine consumption was monitored for 24 days with each blue dot in the figure representing the representation at which the medicine was removed from the smart blister medicine packet. The patient medicine consumption database from the smart blister medicine packet was analyzed to plot the adherence chart. The graph clearly shows the adherence to the prescribed timings.

The sample data for the patient shows that the patient consumed the medicine within one hour of the prescribed timings with a few blue dots deviating from the one-hour interval which indicated that the patient got the notification for the same. Figure 25 shows the time when the notification was triggered and sent to the patient mobile. This key information can indicate the number of times a user needs to be informed to consume their medicine and whether the user consumed the medicine after getting the notification or not. The time gap between the notification and user consumption also provides key information – the user's importance towards the notification. The red dot in the figure shows the time at which the notification was sent to the user as the one-hour interval of the prescribed medicine time was over.

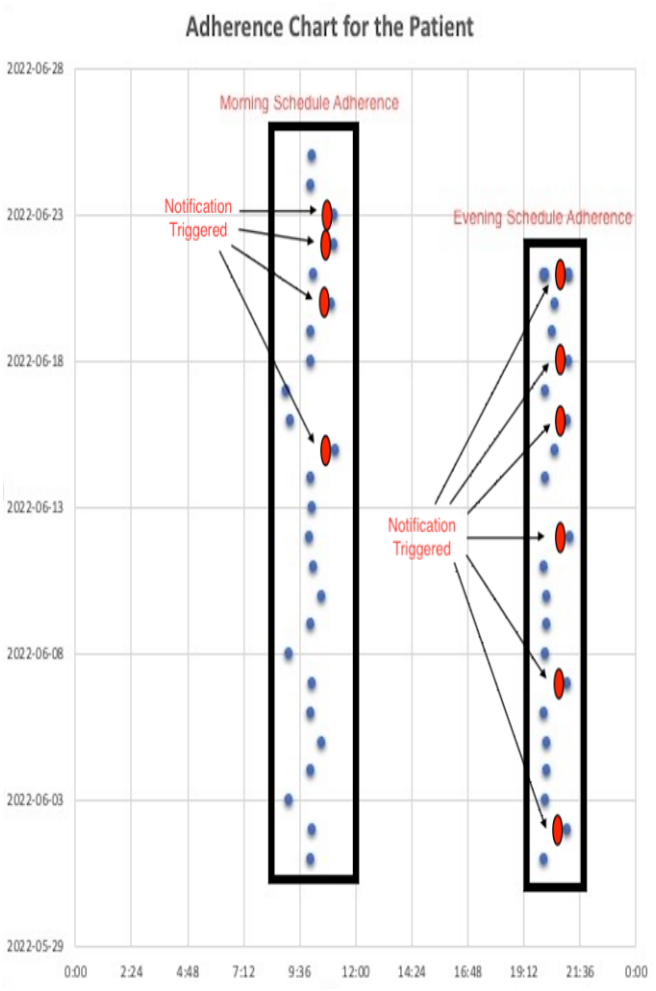

Fig 25 - Adherence chart of medication for the patient showing the notification triggered

CONCLUSION

Our prototype demonstrated the use of NFC to transfer patient database from a smart blister packet to an NFC reader and proposes a notification mechanism to let users remind about the medicines if they failed to take the medicine on time. There are a lot of scopes to improve on the proposed experimental setup. The choice of microcontroller could impact the overall battery life of the microcontroller present on the smart blister packet. A similar experiment can be performed using ultra-low-power microcontrollers such as Texas Instruments TIVA/MSP [15] series. The TCP/IP stack could be replaced with a lightweight communication stack that consumes less power, typically the networking stack which is used in wireless sensor networks such as 6LoWPAN. Instead of the ThingSpeak cloud platform, other hyper scalars cloud platforms such as Amazon Web Services (AWS), Google Cloud, or Microsoft Azure can be used to host the MQTT broker. Advanced analytic tools such as Power BI could be used to create the dashboard UI showing the analysis and graphical representation of the patient's key data and metrics.

In conclusion, through our implementation, we were able to effectively implement NFC communication to monitor and track medication adherence. We did this through prototyping

with an NFC where we loaded it with the sample data and uploaded it to a database by tapping it into an NFC reader. We then parsed the data and performed data analysis. Through our analytics, we measured the patient's deviation for adherence during both their morning and evening schedule. If the deviation was more than 1 hour from when they need to take their medication, the user is sent an SMS notification. Similarly, if the patient also suffers from low inventory, they are also sent a notification.

Our objective is that the patient should take their prescribed medications on time and schedule. This is achieved by sending notifications and positive reinforcement of effective medication intake habits. We were able to successfully show this through our prototype simulation.

REFERENCES

[1] F. Kleinsinger, "The unmet challenge of medication nonadherence," The Permanente journal, 2018. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6045499/. [Accessed: 31-Jul-2022].

[2] Open Source, "What is a Raspberry Pi?." Available: https://opensource.com/resources/raspberry-pi [Accessed: 31-Jul-2022]

[3] Shruti Kuber, "MQTT vs CoAP: What's the best protocol for IoT Solutions?," Available: https://www.emnify.com/developer-blog/mqtt-vs-coap [Accessed: 31-Jul-2022]

[4] Otman Basir, "Application Layer." Available: https://learn.uwaterloo.ca/d2l/le/content/808277/viewContent/4392544/View [Accessed: 31-Jul-2022]

[5] Wireless Links, "NFC Vs RFID: What's The Difference?," Available: https://wlius.com/blog/rfid-vs-nfc-whats-the-difference/ [Accessed: 31-Jul-2022]

[6] Fastmetrics, "What is Cloud Computing & How Does It Work?," Available: https://www.fastmetrics.com/blog/tech/what-is-cloud-computing/ [Accessed: 31-Jul-2022]

[7] CloudBolt Software, "What is a cloud platform?" Available: https://www.cloudbolt.io/what-is-a-cloud-platform/ [Accessed: 31-Jul-2022]

[8] ThingSpeak, "Learn More About ThingSpeak?," Available: https://thingspeak.com/pages/learn_more [Accessed: 31-Jul-2022]

[9] AARDEX Group, "MEMS HH (Helping Hand)," Available: https://www.aardexgroup.com/solutions/mems-adherence-hardware/ [Accessed: 31-Jul-2022]

[10] HealthBeacon, "The world's leading digital health platform for injectable medications," Available: https://healthbeacon.com/ [Accessed: 31-Jul-2022]

[11] Spencer health solutions, "Meet spencer," Available: https://spencerhealthsolutions.com/home/meet-spencer/ [Accessed: 31-Jul-2022]

[12] Twilio, "The next generation of business messaging," Available: https://www.twilio.com/messaging [Accessed: 31-Jul-2022]

[13] Mathworks, "API Reference," Available: https://www.mathworks.com/help/thingspeak/channels-and-charts-api.html [Accessed: 31-Jul-2022]

[14] Thingspeak, "Channel Stats," Available: https://thingspeak.com/channels/1810732 [Accessed: 1-Aug-2022]

[15] Texas Instruments, "MSP-EXP432E401Y," Available: https://www.ti.com/tool/MSP-EXP432E401Y [Accessed: 1-Aug-2022]