**Chapter 1**

**Introduction**

**Chapter 1**

**Introduction**

* 1. **Introduction**

The objective of this project is to monitor and improve the quality of care of people in remote location and to provide continuous information about the patient for making better healthcare decisions in critical situation and to reduce the regular check-up of the aged patients. It helps the doctors to monitor their patients at any time apart from their consulting hours. Improved home care facilities and regular health updates to clinicians reduce the chances of redundant or inappropriate care. It improves patient care and safety by reduction in overall costs for care. The patient’s device gathers data from the sensors and sends to the server automatically and if physical parameters are in abnormal then the server sends a notification with a message to the doctor. The automation reduces the risk of error. This type of solution employs sensors to collect patient’s health status and the server analyses and stores the information. It replaces the process of having a health professional come by at regular intervals to check the patient’s vital signs, instead providing a continuous automated flow of information. In this way, it simultaneously improves the quality of care through constant attention and lowers the cost of care by eliminating the need for a caregiver to actively enhance in data collection and analysis. These solutions can be used to securely capture patient health data from a variety of sensors, apply algorithms to analyse the data on the server. E-health is targeted to provide low cost and everyday household usage [1]. Martin et al discuss in [2] issues surrounding wearable computers used for health monitoring where the devices provide real-time feedback to the patient. Expansion of information technology in healthcare tools and services such as, tele-monitoring platform and mobile health application [3], have created new opportunities for patients and doctors in health care system and provide opportunity for remote monitoring of clinically relevant and parameters in nonclinical settings [4]. These devices can be integrated into routine care of acute and chronic diseases and provides essential information for management to both the healthcare providers and patients [5].

**1.2 Health Challenges**

Health is primary element that human require. So physical and mental fitness is very important as it plays important role for development. Now a days Global health issue is major concern. Definition of health according to World Health Organization [6] is - “A state of complete physical, mental and social well-being and not merely absence of disease and infirmity.” Most of the countries having health care as their top most concern. The statistics [7] shows the population density of Bangladesh was around 1,252 people per square kilometre in 2016, health issue effect the growth and development of human being. There are many chronicle and non-chronicle disease which plays major roll in health issue.

**1.3 Objectives**

The objectives of this project are listed as below:

1. User-friendly operation process and lightweight body temperature sensors: Physiologic parameters are measured by a wearable belt-like sensor and recorded in the web server for long run monitoring.
2. Easy information sharing between patients and doctors through the web server interface: The doctors and family members can observe the patient’s chronic condition and the doctor can monitor remotely. If farther information required the doctor can share that concerns.
3. Real-time response for emergency conditions: The proposed system supports a real time alarming service in urgent situations, such as blood pressure rising or trace condition. So that unexpected events can be handled on time.
   1. **Project Outline**

Chapter-2 outlines some of the relevant research projects on E-Health Monitoring System. Chapter-3 provides the system architecture and design and a basic outlook to the system operation. Chapter-4 provides Full overview of the hardware configuration and working flowchart. Chapter-5 demonstrates the Server design and database management. Chapter-6 tells about the Android Application development. Chapter-7 gives the conclusion and future work.

**Chapter 2**

**System Requirement**

**Chapter 2**

**System Requirement**

**2.1 Previous Work**

HealthGear [8] is a real-time wearable system for monitoring, visualizing and analysing physiological signals. HealthGear consists of a set of non-invasive physiological sensors wirelessly connected via Bluetooth to a cell phone which stores, transmits and analyses the physiological data, and presents it to the user in an intelligible way. In this paper, they focus on an implementation of HealthGear using a blood oximeter to monitor the user’s blood oxygen level and pulse while sleeping.

CodeBlue [9] is a wireless infrastructure intended for deployment in emergency medical care, integrating low-power, wireless vital sign sensors, PDAs, and PC-class systems. CodeBlue will enhance first responders’ ability to assess patients on scene, ensure seamless transfer of data among caregivers, and facilitate efficient allocation of hospital resources

MobiHealth [10] project has developed and trailed a highly customisable vital signals' monitoring system based on a Body Area Network (BAN) and an m-health service platform utilizing next generation public wireless networks. The developed system allows the incorporation of diverse medical sensors via wireless connections, and the live transmission of the measured vital signals over public wireless networks to healthcare providers.

Chen [11] developed a friendly web-based interface that is convenient to the observation of immediate human physiological signals. Moreover, this study also proposes an intelligent data analysis scheme based on the modified cosine similarity measure to diagnose abnormal human pulses for exploring potential chronic diseases. Therefore, the proposed system provides benefits in terms of aiding long-distance medical treatment, exploring trends of potential chronic diseases, and urgent situation informing for sudden diseases.

In recent years, there has been a proliferation of consumer health monitoring devices. A good portion of these devices have been developed for the sports conditioning and weight management areas. There are sophisticated watches available today that provide real-time heart rate information and let users store and analyse their data on their home PCs.

.

**2.2 Requirement of the proposed System**

All the previous work done based on the health monitoring systems mainly focus on the data collection, while processing and analysing of the data is performed offline. This is the major problems of the existence systems. In some cases, the portability of monitoring device is also a problem.

In HealthGear project, cell phone is used for storing and processing of the physiological data which provides lack of storage capacity and data processing. The Codeblue project comes with the solution of real time response. In Chen’s system data was processed in the server which solves the problem of HealthGear project.

In the recent years, almost every person has an Android Smart Phone. Android is the most widely used smartphone operating system. We have designed a system for the health monitoring system using Android Smart Phone. Using this phone can easily upload the patient current health status to the server and can communicate with doctors.

**Chapter 3**

**System Overview and Architecture**

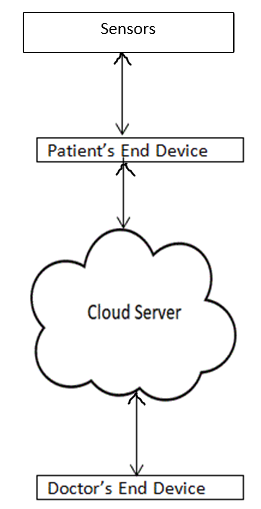
**Chapter 3**

**System Overview and Architecture**

**3.1 System Overview**

The proposed system consists of four principal parts. These are Microcontroller (Arduino Board), Patient device, Server, Doctor device as shown in figure-1.

* Arduino Board: The Arduino board with sensors and Bluetooth module is used to get the patient current health status in the patient’s device.



* Patients Device: The patient’s end device contains an Android phone. After getting the data from Arduino circuit board in the android phone, then the data is send to the server using an Android Application.
* Server: The is the main processing unit that store the data, analyse the data and make a decision to send notification to the doctor under some condition of patient’s physical parameters. The sensor also notify the patient about his/her condition.

Figure 1: Basic components of SHMS

* Doctor Device: The doctor’s end device is an Android phone with and Android Application through this App the doctor can monitor the patient’s health and can communicate with the patient.

**3.2 System Architecture**

Figure 1 depicts the block diagram of the proposed system client-server architecture of the system for a remote health monitoring.

The major components of the architecture are discussed below.

**Data Acquisition** is performed by multiple wearable sensors that measure physiological biomarkers, such as ECG, body temperature, Heart rate and blood pressure. The sensors connect to the Arduino uno microcontroller which sends data periodically to Android Application via the Bluetooth module.

**Data Transmission** components (Android Application) of the system are responsible for conveying recordings of the patient from the patient’s house (or any remote location) to the server in the database with assured security and privacy, ideally in near real-time.

**Cloud Processing,** Server is the core of this project. The system is designed for long term storage of patient’s biomedical information in MySQL database as well assisting health professionals with diagnostic information. We have designed a server to handle the data transmission between the patient and the doctor. To analyse the data sent by the patient we have designed an API (Application Programming Interface) that check the data stored in the database and make decision to send a notification to the doctor about the patient’s physiological condition. The server sends a notification to the doctor Application about the patient’s health condition whenever there is abnormality in the database. Every message with notification is saved in the server database. The doctor may ask to do further test through the android application.

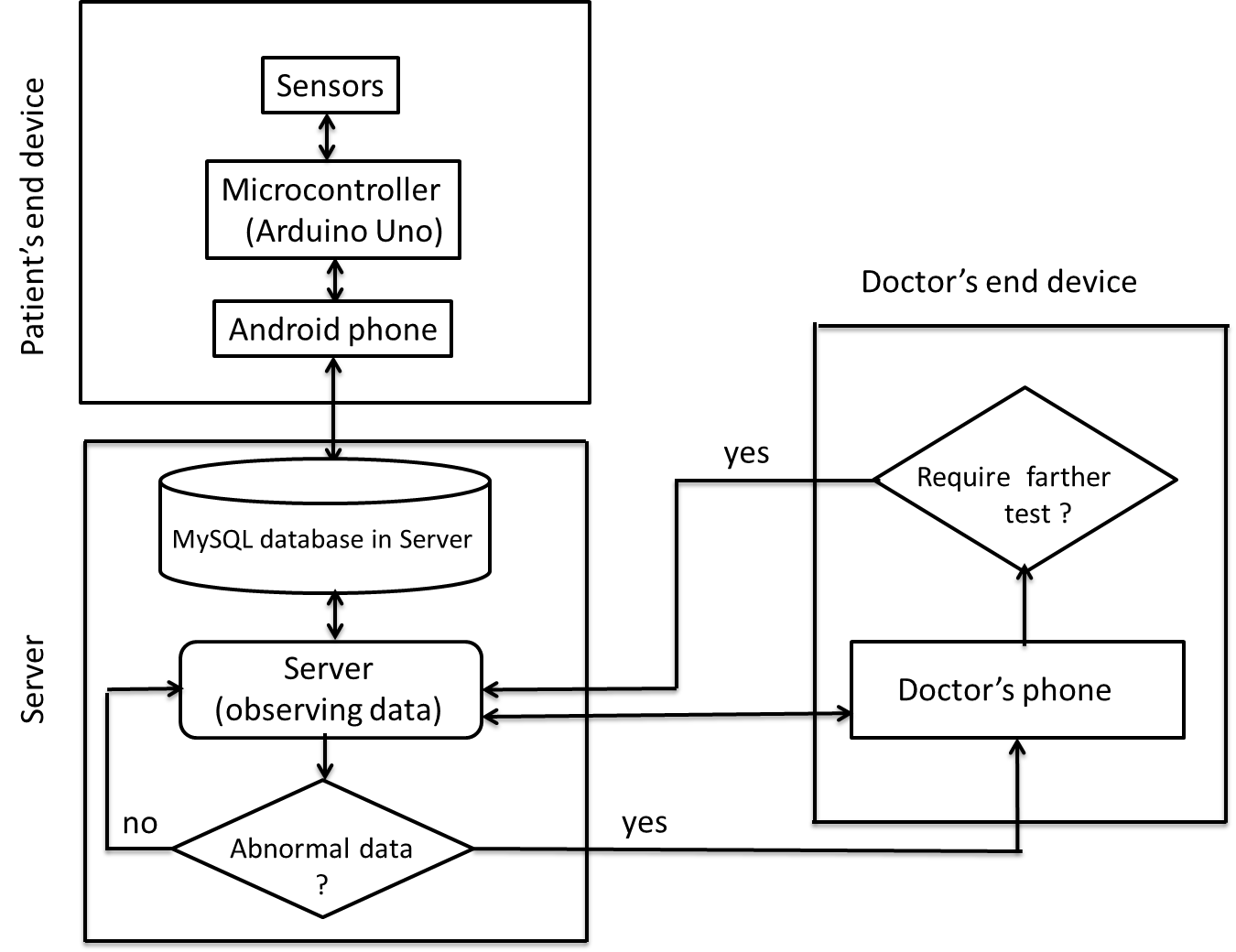


Figure 2: System architecture.

**Chapter 4**

**Experimental Setup**

**Chapter 4**

**Experimental Setup**

**4.1 Introduction**

The following components are needed in our project.

**4.1.1 Sensor**

In the broadest definition [12], a **sensor** is a device, module, or subsystem whose purpose is to detect events or changes in its environment and send the information to other electronics, frequently a computer processor. A sensor is always used with other electronics, whether as simple as a light or as complex as a computer.

**4.1.2 Temperature Sensor**

This probe is a DS18B20 temperature sensor [13] enclosed in a stainless steel head on a durable 90cm rubberized cable, making this sensor ideal for exposed temperature measurements including aquarium applications. A resistor is included. Operating Voltage: 3.0-5.5VDC, Operating Current: <1mA, Temperature Range: -55°C to 125°C (±0.2°C), 36" waterproof cable,4.7kΩ resistor included.



Figure 3 : Temperature sensor Figure 4 : Heart Rate Sensor

**4.1.3 Heart Rate Sensor**

This is a passive electrode for a SHIELD-EKG-EMG shield [14] which allows Arduino like boards to capture Electrocardiography Electromyography signals. The shield opens new possibilities to experiment with bio feedback. **Biofeedback** is the process of gaining greater awareness of many physiological functions primarily using instruments that provide information on the activity of those same systems, with a goal of being able to manipulate them at will. Some of the processes that can be controlled include brainwaves, muscle tone, skin conductance, heart rate and pain perception.

**4.1.4 Arduino Uno**

**Arduino** uno [15] is a single-board microcontrollers and microcontroller kits for building digital devices and interactive objects that can sense and control objects in the physical world. Arduino boards are available commercially in preassembled form.



Figure 5 : Arduino Uno

Arduino board designs use a variety of microprocessors and controllers. The boards are equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (*shields*) and other circuits. The boards feature serial communications interfaces, including Universal Serial Bus (USB) on some models, which are also used for loading programs from personal computers. The microcontrollers are typically programmed using a dialect of features from the programming languages C and C++. In addition to using traditional compiler toolchains, the Arduino project provides an integrated development environment (IDE) based on the Processing language project.

Arduino [16] is a microcontroller board based on the ATmega328P (a single-chip microcontroller). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz quartz crystal, a USB connection, a power jack, an ICSP header and a reset button

**4.1.5 SHIELD-EKG-EMG**  This is an EKG/EMG shield [17] which allows Arduino like boards to capture Electrocardiography Electromyography signals. The shield opens new possibilities to experiment with bio feedback. SHIELD-EKG-EMG converts the analog differential signal (the ECG/EMG bio potentials generated by muscles), attached to its CH1\_IN+/CH1\_IN- inputs, into a single stream of data as output. The output signal is analog and have to be discretized further with aim to give the option of digital processing.

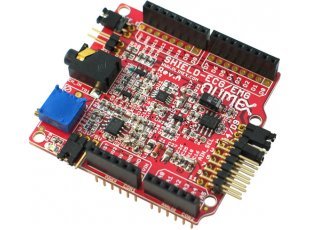


Figure 6 : EKG/EMG shield

**4.1.6 Bluetooth Module**

**HC‐05 module** [18] is an easy to use **Bluetooth SPP (Serial Port Protocol) module**, designed for transparent wireless serial connection setup. The HC-05 Bluetooth Module can be used in a Master or Slave configuration, making it a great solution for wireless communication. This serial port Bluetooth module is fully qualified **Bluetooth V2.0+EDR (Enhanced Data Rate)** 3Mbps Modulation with complete 2.4GHz radio transceiver and baseband. It uses **CSR Bluecore 04**‐External single chip Bluetooth system with CMOS technology and with AFH (Adaptive Frequency Hopping Feature). Figure 7 shows the Bluetooth module and Figure 8 shows the connection between Bluetooth module and Arduino uno.

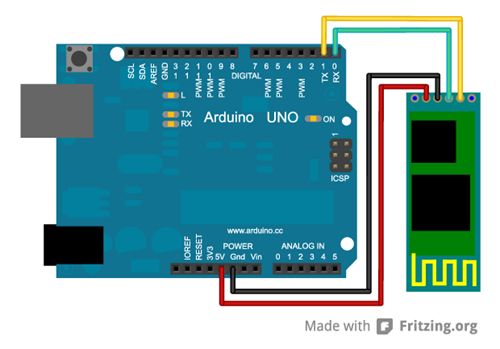
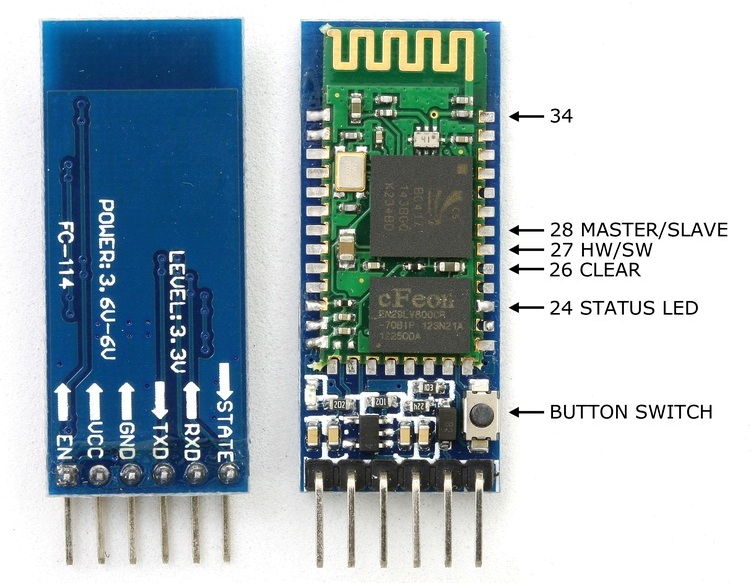


Figure 7 : Bluetooth module Figure 8 : Bluetooth module with arduino uno

The Bluetooth module HC-05 is a MASTER/SLAVE module. By default the factory setting is SLAVE. The Role of the module (Master or Slave) can be configured only by AT COMMANDS. The slave modules cannot initiate a connection to another Bluetooth device, but can accept connections. Master module can initiate a connection to other devices. The user can use it simply for a serial port replacement to establish connection between MCU and GPS, PC to the embedded project, etc. Vcc and Gnd of the Bluetooth module goes to Vcc and Gnd of the Arduino. The TXD pin goes to RXD pin of Arduino and RXD pin goes to TXD pin of Arduino i.e. (digital pin 0 and 1).

### Hardware Features

* Typical ‐80dBm sensitivity.
* Up to +4dBm RF transmit power.
* 3.3 to 5 V I/O.
* PIO (Programmable Input/Output) control.
* UART interface with programmable baud rate.
* With integrated antenna.
* With edge connector.

**Software Features**

* Slave default Baud rate: 9600, Data bits:8, Stop bit:1,Parity:No parity.
* Auto‐connect to the last device on power as default.
* Permit pairing device to connect as default.
* Auto‐pairing PINCODE:”1234” as default.

**4.1.7 Interfacing DS18B20 with Arduino Uno**

The DS18B20 temperature sensor as it requires only 3 wires. Power, ground and a data wire which is connected to digital pin 2 on the Arduino board. I have connected pins 1 to Vcc, pin 3 to Gnd. Furthermore connect pin 2 to digital pin of Arduino. And the pull up resistor between pins 2 and 3 of value 4.7kΩ. The DS18B20 sensor used here is a pre-wired sensor. There is a 4.7K resistor mounted inside the heat shrink. The resistor is soldered between the power (red) and data (yellow) wires. It was then covered with black heatshrink to protect it as shown in the picture below.

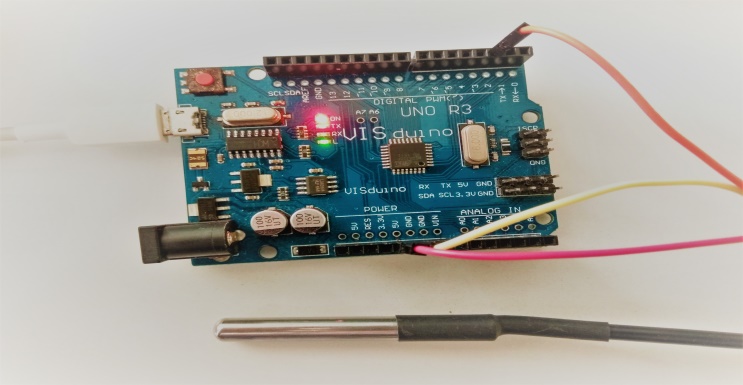
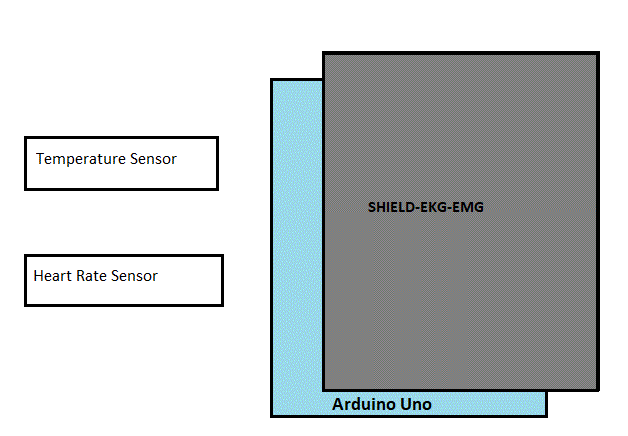


Figure 9: Arduino with temperature sensor

**4.2 Hardware Configuration**

The following figure 10 shows the entire circuit setup.



Bluetooth Module

Power Supply

Figure 10: block diagram of entire circuit setup

* Arduino is powered by a DC power supply of 5 Volt.
* SHIELD-EKG-EMG shield is placed over the Arduino. Now all the circuit setup will be completed in the SHIELD-EKG-EMG shield.
* Now Temperature sensor and heart rate sensor is connected to the shield.
* Bluetooth module is connected to the shield.

**4.3 Hardware Operation**

Figure 11 shows the hardware operation.

Start

Configure Bluetooth and wait for a connection

Wait for a certain period

Send parameters to server

Observe the sensor data

Yes

No

Is connected?

Figure 11: Workflow in hardware

Arduino is supplied the DC 5V power after the circuit setup. It needs to enable the Bluetooth in the android device and wait for a connection. Bluetooth module of the circuit is found name HC-05, pair with it by using default password 1234.

After the Bluetooth connection establishment, data are transferred to the android device and check it is connected or not to the Android app. If not then again try to be connected with the android phone. If connected then, the android phone (patient’s device) send the data to the server for further processing.

Patient’s device waits for a certain period for getting information from the Bluetooth module. After some time it takes the data from Arduino via the Bluetooth module periodically send the data to the server.

**Chapter 5**

**Server Design**

**Chapter 5**

**Server Design**

**5.1 Introduction**

In computing, a **server** [19] is a computer program or a device that provides functionality for other programs or devices, called "clients". This architecture is called the client–server model, and a single overall computation is distributed across multiple processes or devices. Servers can provide various functionalities, often called "services", such as sharing data or resources among multiple clients, or performing computation for a client. A client process may run on the same device or may connect over a network to a server on a different device. Typical servers are database servers, file servers, mail servers, print servers, web servers, game servers, and application servers.

**5.2 Database Design**

**Database design** [20] is the process of producing a detailed data model of a database. This data model contains all the needed logical and physical design choices and physical storage parameters needed to generate a design in a data definition language, which can then be used to create a database. A fully attributed data model contains detailed attributes for each entity.

The term database design can be used to describe many different parts of the design of an overall database system. Principally, and most correctly, it can be thought of as the logical design of the base data structures used to store the data. In the relational model these are the tables and views. In an object database the entities and relationships map directly to object classes and named relationships. However, the term database design could also be used to apply to the overall process of designing, not just the base data structures, but also the forms and queries used as part of the overall database application within the database management system (DBMS).

**5.2.1 MySQL Database**

**MySQL** is an open-source relational database management system (RDBMS). Its name is a combination of "My", the name of co-founder Michael Widenius's daughter and "SQL", the abbreviation for Structured Query Language

* MySQL is a database system used on the web
* MySQL is a database system that runs on a server
* MySQL is ideal for both small and large applications
* MySQL is very fast, reliable, and easy to use
* MySQL uses standard SQL
* MySQL compiles on a number of platforms
* MySQL is free to download and use
* MySQL is developed, distributed, and supported by Oracle Corporation

**5.2.2 Database Model**

A **database model** is a type of data model that determines the logical structure of a database and fundamentally determines in which manner data can be stored, organized and manipulated. The most popular example of a database model is the relational model.

**5.2.3 Database Table**

We have designed three tables in the server database. These are given below.

* Mytab
* Messages
* Users

**Mytab Table :** This table is used to store the patient’s physical data which is received from the sensor.

|  |  |  |  |
| --- | --- | --- | --- |
| Id | Temperature(centigrade) | Heart Rate (bpm) | time |
|  |  |  |  |
|  |  |  |  |

**Messages table :** This table is used to store the messages between the doctor and the patient.

|  |  |  |
| --- | --- | --- |
| Name | Message | date |
|  |  |  |
|  |  |  |

**Users Table :** This table is use to store the FCM token of the doctor’s and the patient’s devices.

|  |  |
| --- | --- |
| Id | Token |
|  |  |
|  |  |

**5.3 Web Server Design**

A **web service** [21] is a service offered by an electronic device to another electronic device, communicating with each other via the World Wide Web. In a web service, the Web technology such as HTTP—originally designed for human-to-machine communication—is utilized for machine-to-machine communication, more specifically for transferring machine-readable file formats such as XML and JSON. In practice, a web service typically provides an object-oriented web-based interface to a database server, utilized for example by another web server, or by a mobile app, that provides a user interface to the end user.

**5.3.1 HTTP Methods**

HTTP defines a set of **request methods** [22] to indicate the desired action to be performed for a given resource. Although they can also be nouns, these request methods are sometimes referred to as *HTTP verbs*. Each of them implements a different semantic, but some common features are shared by a group of them: e.g. a request method can be safe, idempotent, or cacheable.

GET

The GET method requests a representation of the specified resource. Requests using GET should only retrieve data.

HEAD

The HEAD method asks for a response identical to that of a GET request, but without the response body.

POST

The POST method is used to submit an entity to the specified resource, often causing a change in state or side effects on the server

PUT

The PUT method replaces all current representations of the target resource with the request payload.

DELETE

The DELETE method deletes the specified resource.

CONNECT

The CONNECT method establishes a tunnel to the server identified by the target resource.

OPTIONS

The OPTIONS method is used to describe the communication options for the target resource.

TRACE

The TRACE method performs a message loop-back test along the path to the target resource.

PATCH

The PATCH method is used to apply partial modifications to a resource.

**5.3.2 HTTP Status Codes**

This is a list of Hypertext Transfer Protocol (HTTP) response status codes [23]. Status codes are issued by a server in response to a client's request made to the server. It includes codes from IETF Request for Comments (RFCs), other specifications, and some additional codes used in some common applications of the Hypertext Transfer Protocol (HTTP). The status code is part of the HTTP/1.1 standard (RFC 7231).

The Internet Assigned Numbers Authority (IANA) maintains the official registry of HTTP status codes.

|  |  |  |  |
| --- | --- | --- | --- |
| 1×× Informational | 2×× Success | 3×× Redirection | 4×× Client Error |
| 100 Continue | 200 OK | 300 Multiple Choices | 400 Bad Request |
| 101 Switching Protocols | 201 Created | 301 Moved Permanently | 401 Unauthorized  403 Forbidden |
| 102 Processing | 202 Accepted | 302 Found | 404 Not Found |

**5.3.3 Firebase Cloud Messaging (FCM)**

Firebase Cloud Messaging (FCM) [24] is a cross-platform messaging solution that lets you reliably deliver messages at no cost.

Firebase Cloud Messaging (commonly referred to as FCM), formerly known as Google Cloud Messaging (GCM), is a cross-platform solution for messages and notifications for Android, iOS, and web applications, which currently can be used at no cost.

Using FCM, we can notify a client app that new email or other data is available to sync. We can send notification messages to drive user re-engagement and retention. For use cases such as instant messaging, a message can transfer a payload of up to 4KB to a client app.

The following figure [figure 12] demonstrate the working principle of Firebase Cloud Messaging.

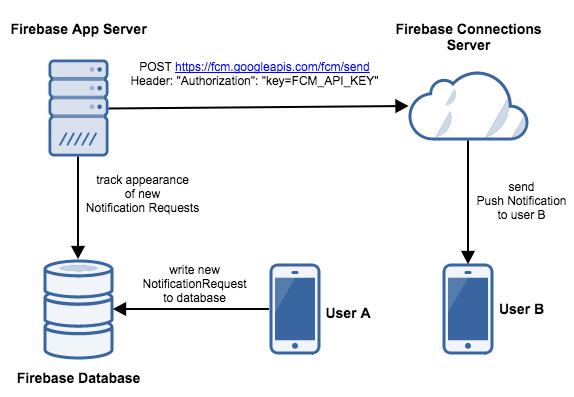


Figure 12: FCM

Firebase Cloud Messaging (FCM) offers a broad range of messaging options and capabilities.

Message types:

With FCM, we can send two types of messages to clients:

* Notification messages, sometimes thought of as "display messages." These are handled by the FCM SDK automatically.
* Data messages, which are handled by the client app.

Notification messages contain a predefined set of user-visible keys. Data messages, by contrast, contain only your user-defined custom key-value pairs. Notification messages can contain an optional data payload. Maximum payload for both message types is 4KB, except when sending messages from the Firebase console, which enforces a 1024 character limit.

Messages sent by the FCM v1 HTTP protocol can contain two types of JSON key pairs:

* a common set of keys to be interpreted by all app instances that receive the message.
* platform-specific blocks of keys interpreted only by app instances running on the specified platform.

Platform-specific blocks give you flexibility to customize messages for different platforms to ensure that they are handled correctly when received. In many scenarios, it makes sense to use both common keys and platform-specific keys in a given message.

**5.3.4 JSON**

**JSON** (JavaScript Object Notation) [25] is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language, Standard ECMA-262 3rd Edition - December 1999. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, including C, C++, C#, Java, JavaScript, Perl, Python, and many others. These properties make JSON an ideal data-interchange language. The official Internet media type for JSON is application/json. JSON filenames use the extension .json.

JSON is built on two structures:

* A collection of name/value pairs. In various languages, this is realized as an *object*, record, struct, dictionary, hash table, keyed list, or associative array.
* An ordered list of values. In most languages, this is realized as an *array*, vector, list, or sequence.

These are universal data structures. Virtually all modern programming languages support them in one form or another. It makes sense that a data format that is interchangeable with programming languages also be based on these structures.

**5.3.5 Server Side Scripting**

The server side scripts are written in php language and uploaded to the server. All the php file links of this project can be found in the following link. Link : http://www.zahed.ml/api/. Index of /api aregiven below.

* Parent Directory
* DoctorMessage.php
* PatientMessage.php
* gettingmessage.php
* gettingnews.php
* newspostfromapp.php
* push\_notification.php
* push\_notification\_to\_doctor.php
* push\_notification\_to\_patient.php
* register.php[zahed.ml/fcm/register.php]

**Data Uploading and Storing:**  There is a button in the home activity in the patient’s Android Application (PatientApp) called “*upload the patient health status*”. If we click on the button then it will go to a new activity where we see three text fields 1. Enter the patient name 2. Enter the temperature (Centigrade) 3. Enter the Heat Rate (bpm). After filling the text fields we click on the button called “submit information” to upload the data to the server. The url [*http://www.zahed.ml/api/newspostfromapp.php*] link is used to in the Android Application program. A request is sent with data to the server in that link. The *newspostfromapp.php* file is stored in the server which catch the data that are sent by the Andriod Application. This php file is responsible to store the data in the server database. There is a table in the database named mytab. The columns of the table are title, Temperature, HeartRate, time. This four parameters are sent by the PatientApp. The above php file store the data in the mytab table.

**Notification Sending to Doctor:**  The *push\_notification.php* file is included in the *newspostfromapp.php*. The *push\_notification.php* is called when the temperature is in abnormal state. There is a “if condition” in the *newspostfromapp.php* which check the uploaded the is normal or not. If not, then the push\_notification.php file is called. Then the push\_notification.php send a request to the Firebase Cloud Messaging (FCM) to send a notification to the doctor’s application (DoctorApp) with patients abnormal data which is taken from the mytab table. To send a request to FCM we need a FCM server key and the server need a FCM token key of the doctor’s application which is unique. We get the FCM server key and FCM token key by creating project in https://console.firebase.google.com/ and adding the doctor application to the project. When the push\_notification.php send a request to the FCM server, then the FCM server send a notification to the DoctorApp using the FCM token of the DoctorApp which is generated by the FCM. When DoctorApp get the notification with message from the FCM server, There is text fields to show the server notification message in the DoctorApp.

**Messaging from Doctor to Patient:** After getting the notification from the FCM server, If the doctor has any query or suggestion, then the doctor can send a message to the patient. To complete this task, there is an option “*send message to the patient*” in the home activity of the DoctorApp. After clicking that option, it leads to a new activity where doctor can write his/her query/message and clicks on the “*submit information”* button to send POST request to the server, then the server receives the request and stores the message in the database and forward the message to FCM server, the FCM server sends the query/message to the PatientApp in the JSON format. To perform the above task weed two php files. *DoctorMessage.php* catches the message sent by the doctor and stores the message in table named Message. The columns of the Message table are name, message, date. This php file includes an another php file called *push\_notification\_to\_patient.php*. This *push\_notification\_to\_patient.php* file sends a request with the doctor’s message to FCM server to send a notification message to the PatientApp. FCM server key and FCM token key for the PatientApp is provided in the request.

**Previous Message Between Doctor and Patient:** In both the PatientApp and the DoctorApp has an option “show previous messages” to see the previous messages between the doctor and patient. If he/she clicks on the option, then an activity shows all messages. To do this we need a php file named *gettingmessage.php*. When we clicked on that option then both PatientApp and DoctorApp send a GET request the server to get the messages. The gettingmessage.php receives the request and retrieves all the messages from the Messages table of the database and sends that messages to the DoctorApp and the PatientApp in JSON format. After getting messages from the server in JSON format, the processes the JSON format data and shows in an activity.

**Previous Health Status:** The doctor and the patient can see the previous health status of the patient. We shows the patient’s health status in the website *zahed.ml* that are uploaded by the PatientApp. Actually the sensor parameters of the patient is uploaded by the PatientApp and is shown in that website. The doctor can monitor the website to see the previous health status of the patient. There is an option “show previous health status” in the DoctorApp. If the doctor click on the option, the DoctorApp send a GET request a to the server to get the previous health status. The DoctorApp request is received by the *gettingnews.php*. This file retrieves the data from the mytab table and sends to the DoctorApp in JSON format. The DoctorApp processes the JSON format data and shows in an activity.

**Messaging from Patient to Doctor:** There is an option “send message to the doctor” in the home activity of the PatientApp. If the patient wants to send a message to the doctor, then he/she need to click on that option. It leads to a new activity in which we can write the message for the doctor. After writing the message when the he/she click on the button “*send the message to doctor*”, PatientApp sends a POST request to the server with the message. The PatientMessage.php receives the request and stores the message in the Messages table of the database. It also sends a request to the FCM server to send message notification to the DoctorApp. The FCM server receives the request and forward the message the DoctorApp.

There is an another php file called register.php to handle the devices registration token generated by the FCM server. We have an another table in the database to store the FCM token key which is unique for every single device. This is done by the register.php file.

Here is the link of all the php program files. Link is https://goo.gl/GgqEUn.

**Chapter 6**

**Application Design**

**Chapter 6**

**Application Design**

**6.1 Introduction**

We need to develop two Android Application for our project, one for the Patient named PatientApp and other for the Doctor named DoctorApp. So we need some basic introduction about the android, android application.

**6.2 Android**

Android[26] is a mobile operating system developed by Google, based on a modified version of the Linux kernel and other open source software and designed primarily for touchscreen mobile devices such as smartphones and tablets.

## 6.2.1 Android OS Architecture

The Android operating system [27] provides many services that include support for security, virtual memory, multitasking, and threads—all features of modern day operating systems. This section discusses the architecture components of the Android platform. Each Android application uses its own file system and is packaged in one .apk file. The Android platform is organized into the following sections:

* **Android applications:** This is the topmost layer in the Android platform stack and is comprised of applications that are built-in (developed by the Android team) or any other third party applications that have been installed on the device. Applications that we develop are also installed in this layer. Typical applications include: Camera, Alarm, Clock, Calculator, Contacts, Calendar, Media Player, and so forth.
* **Application framework:** This layer is built using Java and provides high level services and APIs (for example, notifications, sharing data, and so on) that are leveraged by the applications. The key services of the Android framework include: Activity Manager, Content Providers, Resource Manager, Location Manager, Notifications Manager, View System, and Telephony Manager.
* **Native libraries layer:** The Android runtime is comprised mainly of the core libraries and the Dalvik Virtual Machine. The native libraries layer is responsible for providing support for the core features. The WebKit Web rendering engine and the Dalvik virtual machine are shipped as part of this layer. The Dalvik Virtual Machine (commonly called the Dalvik VM), like the Java Virtual Machine (JVM), is a register based virtual machine that provides the necessary optimizations for running in low memory environments.

The Dalvik Virtual Machine converts the byte codes (Java class files) having .class extensions that generated by the Java compiler into the Dalvik—executable files that have .dex extensions. Such binaries are optimized to execute on smaller processors and low memory environments. The Dalvik Virtual Machine takes advantage of the

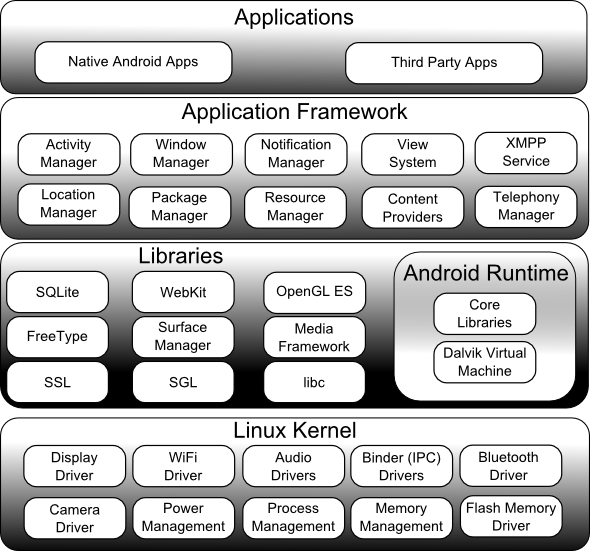


Figure 13: Android OS architecture

core features of Linux that include multi-threading, process and device management, and memory management. Moreover, the DVM is provides support for platform neutrality (the .dex files are platform neutral) and you can have multiple virtual machine instances execute at the same time efficiently. It should be noted that each Android application executes in its own process—inside its own instance of the Dalvik Virtual Machine.

* **Linux Kernel:** The Linux Kernel is the bottom most layer in the Android architecture. The Android platform is built on top of the Linux 2.6 Kernel with a few architectural changes. Note that the term *kernel* refers to the core of any operating system. The Linux Kernel provides support for memory management, security management, network stack, process management, and device management. The Linux Kernel contains a list of device drivers that facilitate the communication of an Android device with other peripheral devices. A device driver is software that provides a software interface to the hardware devices. In doing so, these hardware devices can be accessed by the operating system and other programs.

**6.3 Patient’s Application Design**

After launching the patient’s application program, The login activity is come to login, if the user in new then there is another option to register as a patient. After successful login using the email and password, the login information is stored in the database. The application is manually connected to the Arduino to get the data via Bluetooth module. After the getting the data it send the patient physical information to the server for further processing.

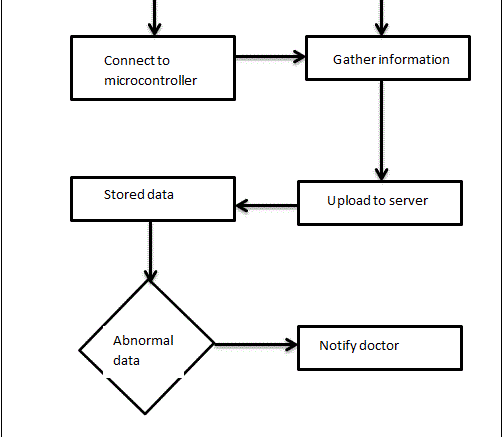
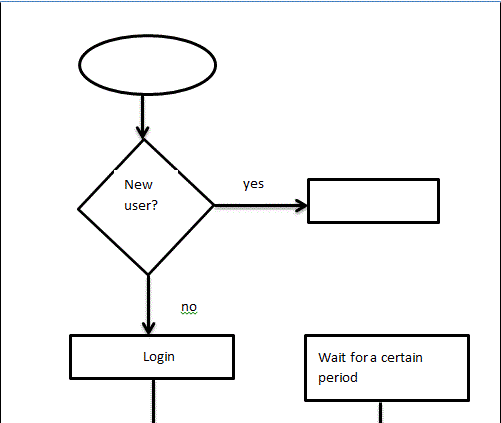
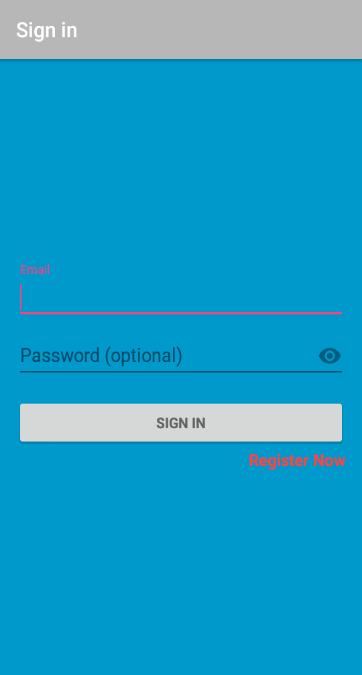
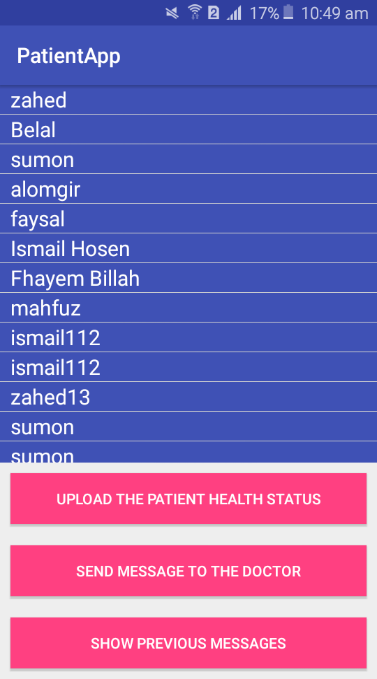


Figure 14: patient’s application lifecycle

After launching the app, then it is opened an sign in

  Figure 15: login and registration activity figure 16: home activity.

The activity is used to upload patient’s physical information to the server

The following activity is used to send message to the data

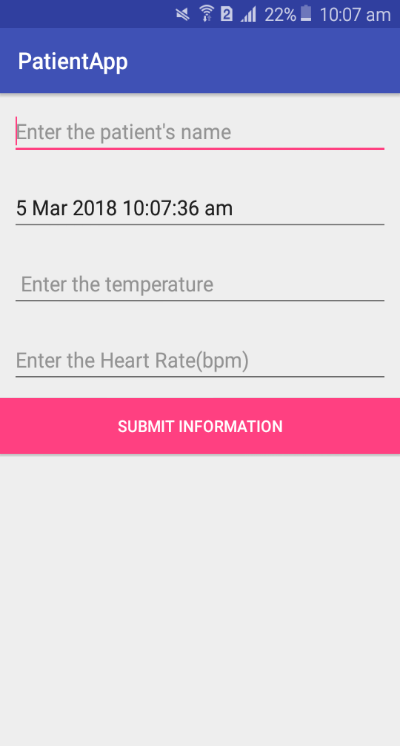
 

Figure 17: data upload activity Figure 18: message sending activity

This activity is used to see the previous message between the doctor and patient.

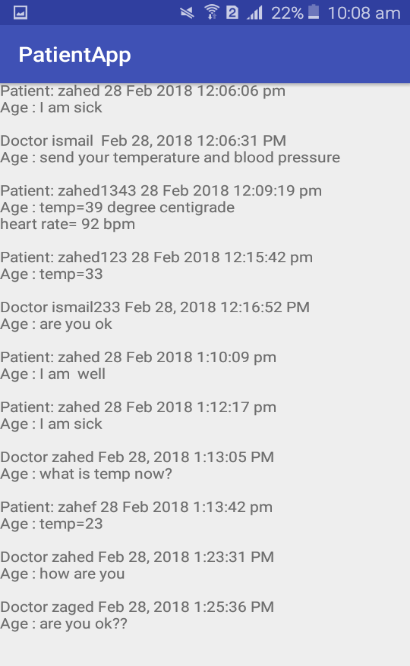


Figure 19: previous message showing activity.

**6.4 Doctor’s Application Design**

When the app is first launched then it go to the sing in activity. If the doctor is new then click on the register option go to the registration activity. After successfully sign in, Home activity open.

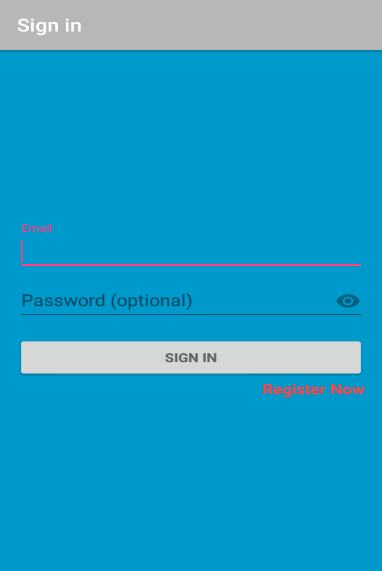
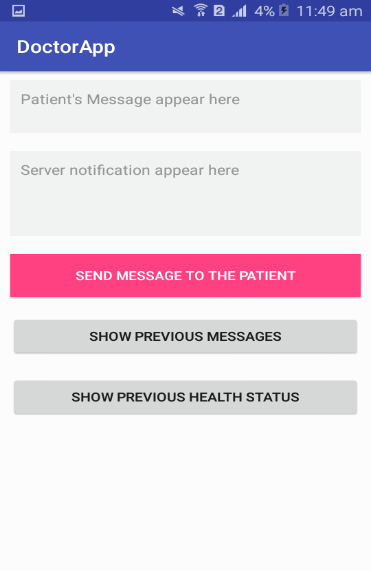
 

Figure 20: Sign in Activity Figure 21: Home activity

In the home activity, there are two text fields, first one is used to see the Patient’s Message, Notification from the server is appeared in the second text fields. There are three other option. These are send message to the patient, show previous messages, message, show previous health status.

The following two activity is used to see the previous message between doctor and the patient which is shown in figure 23 and the other activity is used to see the previous health status which is uploaded to the server by the patient application as shown in figure 24.

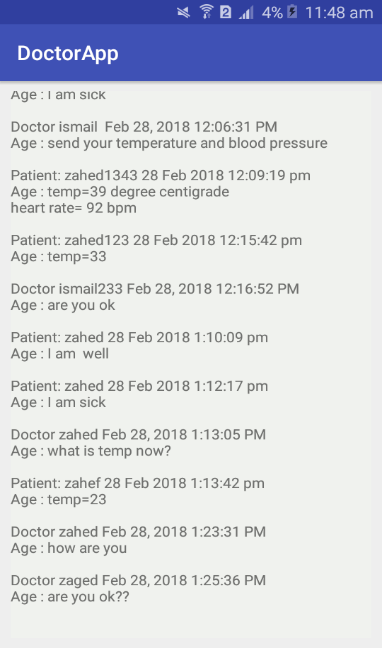
 

Figure 22: message activity Figure 22: previous message showing activity

The next one activity is responsible to send a reply message or any query to the patient as shown in figure 25.

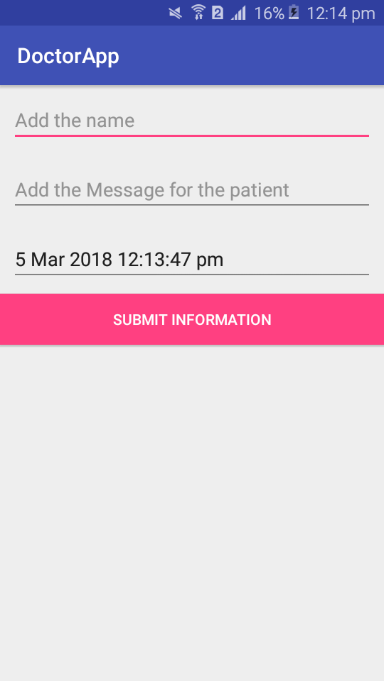


Figure 24: sending message to patient activity

**Chapter 7**

**Conclusion and Future work**

**Chapter 7**

**Conclusion and Future work**

**7.1 CONCLUSION**

Health of the patients are monitored using android phone and enables the doctor to monitor their patients outside the clinic and also apart their consulting hours. Connected health care devices utilize resources to provide an improved quality of care, leading to better clinical outcomes. This android app reduces clinic visits, including reduction in bed days of care and length of stays in hospitals. Using the patient end device patient’s conditions are obtained and stored in the server for further analysis.

In this paper, we reviewed the current state and projected future directions for integration of remote health monitoring technologies into the clinical practice of medicine with wearable sensors, offer attractive options for enabling observation and recording of data in the server.

We highlighted several of the challenges in sensing, analytics, and visualization that need to be addressed before systems can be designed for seamless integration into clinical practice.

A Smart Health care monitoring system is integration of hardware  
and software. Here hardware includes sensor & microcontroller and cloud system is software part. This is complete remote monitoring system which includes task like data acquisition, data monitoring, data storing & managing. Using this system one can measure their own body parameters without help of health provider.

After studying and understanding literature review and other existing work, I propose a technique which uses fewer amounts of hardware i.e. sensor & micro controller, Bluetooth module and then data is stored in the data base for processing in the server. Doctor is notified by the server if data is abnormal.

**7.2 Future work**

In this system future work can be done by providing more sensors & providing more settings to android apps such as nice UI, calling features, GPS tracking for location features, so that person can have more user friendly environment and will be able to do more task automatically rather than manually. This will enhance system performance.

**References**

1. Sundararajan T V P, Sumitra M G, Maheswar R.” **A Novel Smart Routing Protocol for Remote Health Monitoring in Medical Wireless Networks**.” Journal of healthcare engineering, 2004, 5(1)95-122.
2. Martin. T., E. Jovanov, and Raskovic. D. **“Issues in wearable computing for medical monitoring applications: A case study of a wearable ecg monitoring device**.” In Proc. Intl. Symp. Wearable Computers (ISWC’00), pages 43–49, 2000.
3. Hayakawa M, Uchimura Y, Omae K, WakiK, Fujita H, Ohe, :” **A smartphonedbased medication self-management system with realtime medication monitoring**.” Appl Clin inform 2013, 4:37-52.
4. Swan M : “**Emerging patient-driven health care models: an examination of health social networks, consumer personalized medicine, and quantified self tracking”**. Int J Environ Res Public Health 2009, 6:492-525**.**
5. Chen KY, Besset DR: “**The technology of accelerometry-based activity monitors: current and future**.” Med Sci Sports Exerc 2005, 37:S490-S500.
6. World Health Organization : “**Constitution of WHO: principles**.” Link http://www.who.int/about /mission/en.
7. The Statistics Portal : “**Population density of Bangladesh from 2005 to 2016 (in people per square kilometer).”** Link https://www.statista.com/statistics/778381/bangladesh-population-density/.
8. Nuria Oliver1 & Fernando Flores-Mangas, “**HealthGear: A Real-time Wearable System for Monitoring and Analyzing Physiological Signals**” Wearable and Implantable Body Sensor Networks, 2006. BSN 2006. International Workshop on, 2006, pp. 4 pp.-64.
9. David Malan, Thaddeus Fulford-Jones, Matt Welsh, and Steve Moulton, “**CodeBlue: An Ad Hoc Sensor Network Infrastructure for Emergency Medical Care**” International Workshop on Wearable and Implantable Body Sensor Networks, April, 2004, London, UK.
10. Aart van Halteren and Richard Bults and Katarzyna Wac and Nicolai Dokovsky and George Koprinkov and Ing Widya and Dimitri Konstantas and Val Jones and Rainer Herzog "**Wireless body area networks for healthcare: the MobiHealth project**." Studies in health technology and informatics, 2004, volume-108, pages(181-93).
11. Chun-Hsien Chen, “**Web-based remote human pulse monitoring system with intelligent data analysis for home health care**”, Expert Systems with Applications, Volume 38 Issue 3, March, 2011,Pages 2011-2019.
12. Wikipedia : **“Sensor”**. Link : <https://en.wikipedia.org/wiki/Sensor>
13. Temperature Sensor : **“DS18B20 Digital Temperature Sensor - Stainless Steel Probe, Waterproof.”**  Link : <https://www.rpelectronics.com/sgh-ds-002-digital-temperature-probe-steel.html>.
14. Heart Rate Sensor : **“PAsive electrodes for Electrocardiography Electromiography shields for DUINOMITE, PINGUINO, MAPLE, ARDUINO like development boards”.**  Link : <https://www.olimex.com/Products/Duino/Shields/SHIELD-EKG-EMG-PA/open-source-hardware>.
15. Wikipedia : “**Arduino Uno**”. Link https://en.wikipedia.org/wiki/Arduino.
16. **Arduino** uno : “**Arduino Uno Rev3**”. Link : <https://store.arduino.cc/usa/arduino-uno-rev3>.
17. SHIELD-EKG-EMG : “**Open Source Hardware Electrocardiography electromyography shield for DUINOMITE, PINGUINO, MAPLE, ARDUINO like development boards**”. Link : https://www.olimex.com/Products/Duino/Shields/SHIELD-EKG-EMG/open-source-hardware
18. Bluetooth Module HC-05: “**Introduction, Bluetooth Module HC-05”,** Link : https://wiki.eprolabs.com/index.php?title=Bluetooth\_Module\_HC-05.
19. Wikipedia : **“Server (Computing)”.** Link : https://en.wikipedia.org/wiki/Server\_(computing).
20. Wikipedia : “**Database design**”. Link : https://en.wikipedia.org/wiki/Database\_design.
21. Wikipedia : “**Web service**”. Link : https://en.wikipedia.org/wiki/Web\_service.
22. HTTP Methods : “**HTTP request methods**”. Link : https://developer.mozilla.org/en-US/docs/Web/HTTP/Methods.
23. HTTP status codes : “**List of HTTP status codes**”. Link : https://en.wikipedia.org/wiki/List\_of\_HTTP\_status\_codes.
24. Firebase Cloud Messaging (FCM) : “**Firebase Cloud Messaging**”. Link : https://firebase.google.com/products/cloud-messaging/.
25. JSON : “**Introducing JSON**”. Link : <https://www.json.org/>.
26. Wikipedia : “**Android (operating system)”.** Link : <https://en.wikipedia.org/wiki/Android_(operating_system)>.
27. Android OS Architecture : “**Understanding the Android Platform Architecture**”. Link : <https://www.developer.com/ws/android/understanding-the-android-platform-architecture.html>.