Abstract

E-Health is the use of technology to improve the quality of healthcare and the advances to allow both patients and medical professionals to gain access to a variety of resources. E-health empowers patients to take active role in their treatment, allowing them to gain deeper understanding of their condition and how to effectively manage them. Health informatics focuses on developing tools to improve healthcare while e-health is designed to make it easier to share this valuable information. The two can work in conjunction to provide patients with top quality care, in a much more efficient manner. The project will be using IoT for continuous health monitoring. This IoT based monitoring system will have one bridging device Gateway.We have used Smartphone as the gateway device. It will handle the complex wireless sensor network infrastructure.This type of smart gateway will help us in expanding this kind of health monitoring system in terms of its reliability, energy efficiency etc. The gateway further sends the data to the web server. Therefore, it will establish a platform for both the doctors and the patients to interact.

Contents

List of Figures

List of Tables

1 Introduction

Healthcare has been one of the greatest challenge to mankind. There is always scope for betterment of the available healthcare services. With the advent of technology, the advancements in the healthcare industries have also risen manifold. The integration of technology with the traditional healthcare units have opened doors to numerous research opportunities. Many industries have taken up the challenge of bringing in effectiveness and efficiency to the current medical facilities and in turn increase patients safety. According to the constitution of WHO the highest attainable standard of health is a fundamental right of every individual.

In the traditional diagnosis scenario the treatment highly revolves around the doctor. The doctor has to be physically available to treat the patient. Another major drawback is that the patient has to be glued to the bed-set with all the wires attached to him. We propose a more patient centric approach which will negate both these drawbacks.

E-healthcare focuses on the use of Internet and related technologies to deliver healthcare information and services. E-healthcare, as an industry has evolved over the years. It encompasses a wide variety of technologies ranging from telemedicine to big data. There are various offshoots of E-healthcare like home monitoring systems, interactive health records, automated online therapy, health care system portals. Under E-healthcare we have various technologies like Internet and Smartphones. Now-a-days with a click of a button and the ease of internet, data can transmitted to any remote location where the patient's family or the concerned doctor can be alerted about his well being.

We thus propose a smartphone centric IoT architecture to combat the problems faced in e-healthcare system and also build a platform for the doctors and patients to interact.

1.1 Motivation

The prime most motivation was to bridge the gap between hospitals and homes by creating a decentralized health monitoring architecture. This is one the attributes of telemedicine. Before visiting the doctor on the scheduled check up day the patients physiological parameters can be monitored remotely so that the doctor can be well updated on the patients health to a greater accuracy.

There has been a rising demand for healthcare institutions and many developing nations cannot provide the services due to lacking infrastructure.

To eliminate this problem, we resort to remote health monitoring system where the doctor can remotely address the patients problems and get reported on the various physiological parameters which are needed for diagnosis.

We often come across the difficulty of data portability when we switch doctors. In the proposed E-healthcare system, switching data between doctors will be relatively easy since the records are semantically stored and transferring data on the web is any day easier. This method also increases the efficiency as we can resort to real time data acquisition. Analysis based on a larger data set will increase efficiency.

Import substitute The novelty of our project is that we aim to produce the already existent product at a very low cost which is easily affordable to the consumers. There is currently no producer of a health band in India. When imported we pay almost Rs. 10,000 for a similar band. Our cost of production was only Rs. 1035 which is almost one tenth the cost of the original cost. With the complete designing and branding of the product we can still manufacture it at a cost of Rs. 3000 which is much lesser than the current cost at which it is imported.

Healthcare as we know is of great concern and is applicable to every individual. Hence, we should be looking out for solutions which are affordable to all. Therefore, we aim at building cost efficient wireless wearable sensor network which can acquire and transmit data on a real time basis.

2 Literature Survey

2.1 Literature Overview

1. **Smart Healthcare Monitoring System Based on Wireless Sensor Networks**

   In this Paper, the author suggests solution for a hassle-free, compact and remote monitoring of the patient. Different types of motes such as MoteCare, ReMoteCare and LAURA have been compared in terms of sensors, Inter/Intra/Beyond Body Area Network (BAN) links and their targeted application.

   Different intermediate communication technologies have been compared by the author in terms of range, power and data rate. A 3-Tier Communication System has been proposed. Tier I is the communication between the sensor and the Patient Bed Monitor (PBM) with the help of ZigBee Technology which has a low power, long range and low transmission delay which plays a vital role in medical system. The Tier II is the link between multiple PBMs and the Remote Server using of WiFi which consumes more power but has a high data rate and longer range. Finally Tier III, which connects the Remote Server to the Hospital Network and further to the staffs via Internet for diagnosis. An alert notification is triggered whenever a certain parameter exceeds certain value to the concerned staff for mobile monitoring. The data is also used for real time monitoring as well as long term studies. An RFID system for location tracking of the patient is also proposed.[1]

2. **Remote Monitoring and Medical Devices Control in eHealth**

   In this paper, the author offers a solution which uses IoT Gateways to manage different types of sensors and to process, store and analyze the heterogeneous data collected from sensors.

   It is believed that by 2020, over 50 billion devices will be connected to the internet and out of the tremendous data collected 40 will be from IoT devices and sensors which is equal to 129 yottabytes. This paper says that the total cost of using local resources (edge level) along with cloud resources for right processing, analyzing and storing is one third of only cloud solution. This paper shows the use of IoT gateway at the edge level for communication and storage infrastructure for body sensors, medical devices etc.

   Web real time communication is used for real time data management of body sensors and medical devices and also for client-server communication CoAP protocol is used for sensors management and real time data transmission. In this paper, they implemented modules like device control, analytic engine, alert trigger, Web RTC SDK, secure storage and privacy in the IoT gateway.

They evaluated the performance of the proposed solution considering CPU load percentage, latency for devices remote control and storage occupancy for different video types on different gateway platform (NUC box broadwell CPU - core i5 -2 cores and 109G HD Desktop (DT) device Haswell CPU-core i7-4 cores).They also evaluated their proposed solution against market products like google Nestcam, Samsung smart things, milestone etc. This paper lacks in providing encrypted storage approach for more data security and also it does not provide solutions add more edge devices for heavy processing.[2]

3. An IoT-Aware Architecture for Smart Healthcare Systems

The current proceedings in patient monitoring, supervision and management are manually executed. This represents an efficiency bottleneck. This paper aims at providing an IoT based solution for automatic identification and tracking of people and biomedical devices in the hospital along with real time detection of patients physiological parameters.

With the IoT vision in mind, a complex network infrastructure relying on a CoAP, 6LoWPAN, and REST paradigms have been implemented to allow the inter-operation among UHF RFID Gen2, WSN, and smart mobile technologies. Taking advantage of the zero-power RFID-based data transmission acting on theWireless Sensor Network(WSN), an ultralow-power Hybrid Sensor Network(HSN) has been implemented. The RFID technology provides not only standardized EPCglobal identification and tracking of patients and staff wearing the nodes, but also enables zero power read write memory operations. The system is expected to collect real-time data and deliver it to the control system for further analysis. At the gateway, the monitoring application exploits the received data to inform the nursing staff (remote user) about the patients location and health status. The doctor (local user) can check the patients vital since through a Web Application directly on his phone.[3]

4. An IoT-inspired Cloud-based Web Service Architecture for e-Health Applications

Research these days have been steered into the direction of finding heterogeneous devices to sense peoples physiological parameters, human activity as well and ambient parameters. This paper gives a possible implementation through REST web services which will help in creating different health care applications by using the same infrastructure.

The proposed cloud based model suggests that the cloud will host all of the users data and it features various APIs for the different healthcare services. The APIs transmits the data acquired from the end users to the healthcare services who further use it for further analysis.

An open API was developed to uses. Open web services to and identified only by the Unique Resource Identifier (URI). The resources are accessed using the semantics of HTTP. GET is used to access the resource representation. POST is used to create a new resource. HATEOAS has also been used which is hypermedia as an engine of application states. This provides means by which REST can approach associates resources with other resources. This approach also allows the user to navigate between the REST resources and to easily understand the allowed CRUD(Create, Read, Update and Delete) operations. To make the API accessible on multiple platforms the resources are described using JavaScript Object Notation (JSON). This feature helps to connect the IoT world to the Internet.

Therefore Cloud based architecture will open gates to increased research work in the m-health, s-health and e-health domains.[4]

5. A Smartphone-Centric Platform for Personal Health Monitoring using Wireless Wearable Biosensors

This platform is made up of three key elements: the wearable biosensor, the controller for the biosensor, and the mobile monitoring unit. The biosensors generally have analog input/output signals. The analog outputs from the sensors are acquired by the wearable device and converted to digital signals. Depending on the type of the analog signals generated by the device controller, the A/D conversion ranges from 8 to 10 bits resolution. The cleaned-up data is then transmitted over a short range wireless communication interface, such as Bluetooth, WIFI. The software architecture is implemented as three layers. The main interaction lies at the top in the user interface and the alarm system. The signal processing and the intelligent closed loop control lies at the middle layer. And the bottom layer consists of physical interface for short range and long-range communication.

The closed loop control of device reduces the power consumption of the whole device. It could intelligently determine the mode of operation, the settings and ON/OFF cycle of the device. The platform is portable to different phone operating systems. The Java based software is easily upgradeable and supports different sensors. The advantages of this platform are demonstrated by implementing case study of PPG sensor, Bluetooth and smartphones.[5]

2.2 Gap Findings

1. Most of the companies that provide remote healthcare systems, use cloud computing for real time processing of data which is costly. Hence, we resort to edge computing which provides real time processing of data which also uses low bandwidth and is also cost efficient.

2. In India, there has been a deficit of products which provides the complete e-healthcare package which includes the wearable and the database setup for the enterprise.

3. There is no established platform where the doctors and patients can interact remotely.

4. Inflexible circuits. Once the circuit is mae there is no provision to add more sensors to it.

3 Problem Statement and Objectives

Problem Statement :
To build a real time health monitoring system which will take continuous readings and store it to an active database and perform instantaneous analysis on the collected data set. Alerts will be sent on sensing unusual conditions. Patients' history will be made remotely accessible to the doctors and other staff.

Objectives
The project aims at the following:

1. To reduce the cost we carry out analysis at edge level. Smartphone centric architecture helps in reducing cost of an edge level gateway device as smartphones are easily available these days.

2. Portability and remote accessibility of data.

3. Store real time sensor data on the server and plot real time graphs on an android application.

4. Create alerts and warnings if unusual behavior is observed in the sensor data collected.

5. To create a band which is flexible in the sense that more sensors can be added to the circuit without making changes to the hardware.

6. Create web application to provide user interface for registered patients and doctors where patients history can be seen. We aim at creating a platform where all the patient's data will be made easily available. The doctor can access the data and further suggest medication accordingly.

4 Methodology

4.1 Block Diagram

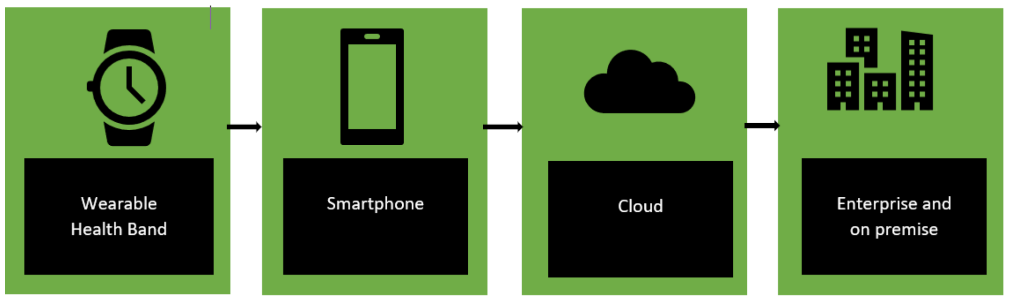


Figure 1: Block Diagram For Smartphone Centric Architecture

In our case, The WSN consists of two sensors, heart rate and temperature. The data is further sent on the the gateway device which is the smartphone. From the smartphone the data will be sent on to the web server using WiFi. The web server will be located in the hospitals where the data can be accessed by doctors and other staff members.

4.2 Detailed Implementation of the Project

We have divided the implementation into the following parts:

4.2.1 Wireless Sensor Network (WSN)

We have used two sensors namely heart rate sensor Invent (INVNT11) and body temperature sensor (LM35).

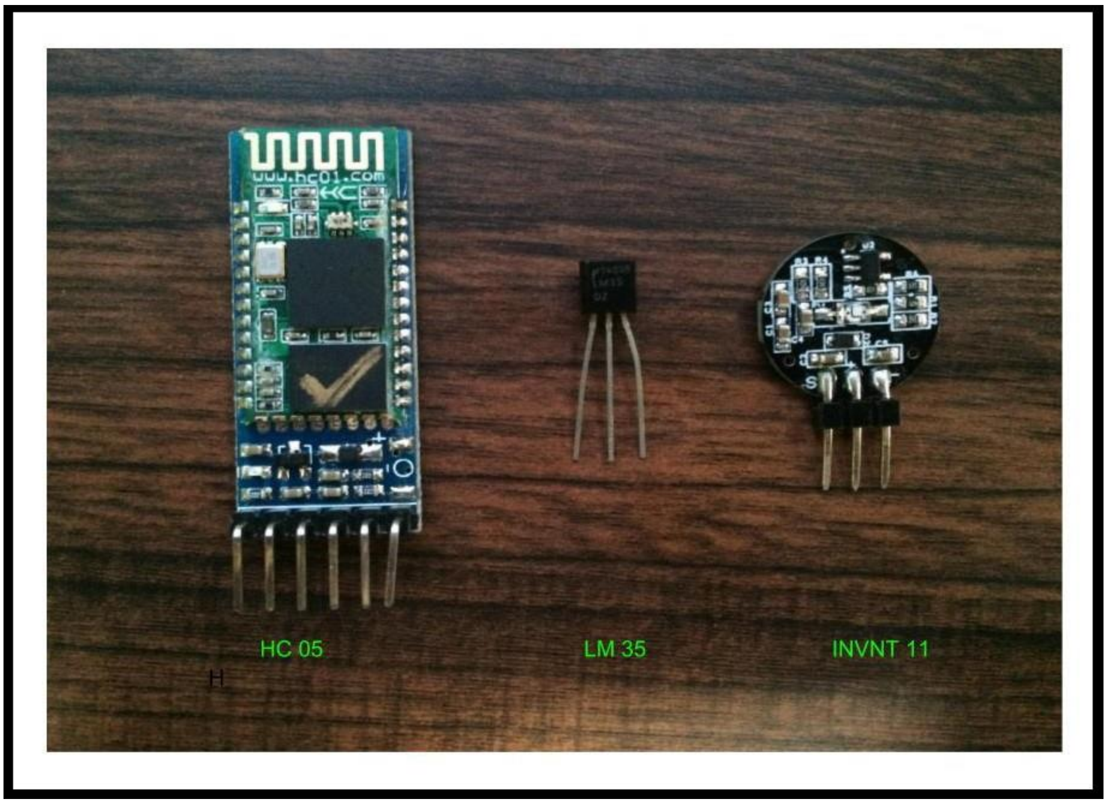


Figure 2: Sensors and Bluetooth Module

Heart Rate Sensor is used to measures the pulse rate of the patient. There are two ways to measure the heartbeat. First is the Manual Way in which the index and middle finger is placed against either the neck (carotid pulse) or the wrist (radial pulse). This reading is monitored for a span of 30 seconds and multiplied by 2 to finally get the pulse rate. The second way is to use a sensor that uses the optical power variation. This method exploits the fact that light is absorbed and scattered as light travels through the blood.

It uses the principle of photo plethysmography. The change of volume of blood through any organ of the body causes a change in light intensity through that organ. This change is measured by the sensor. The blood flow is decided by the heart rate. Therefore, the signal pulses are equivalent to the heart beat pulses.

We have calibrated the sensor values against a commercially available device called ”Fingertip Pulse Oximeter With Heart Rate Monitor”.

Temperature Sensor measures the hotness or coldness of the object. This is proportional to the resistance, current or voltage output which is further measured or processed depending on the application.

4.2.2 Printed Circuit Board

We used Atmega328 instead of Arduino to cut down on the cost and make the circuit more compact. The sensor is interfaced with the microcontroller. The data is further transmitted to the Android Application using the Bluetooth Module.

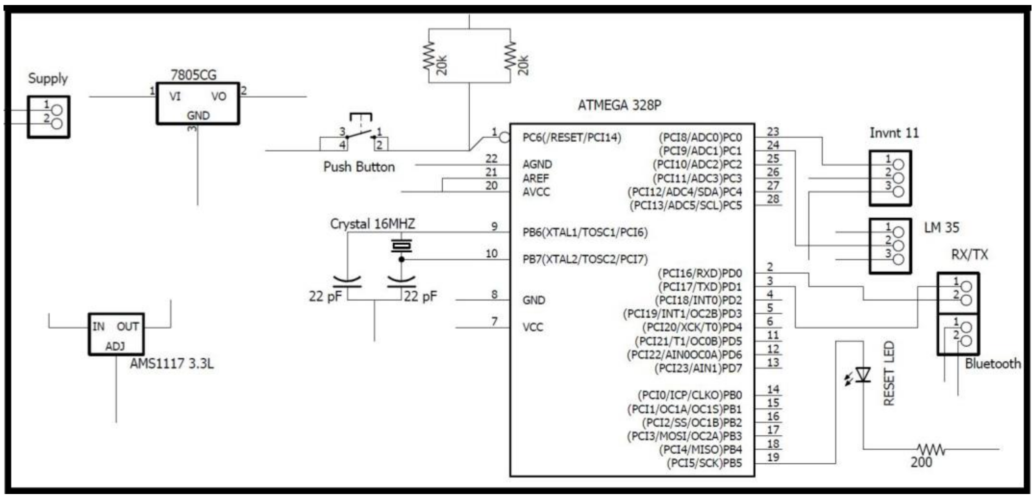


Figure 3: Schematic of WSN board

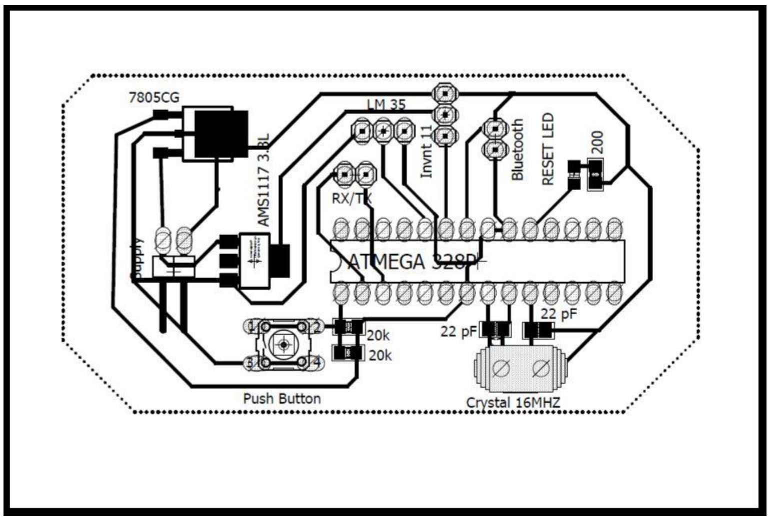
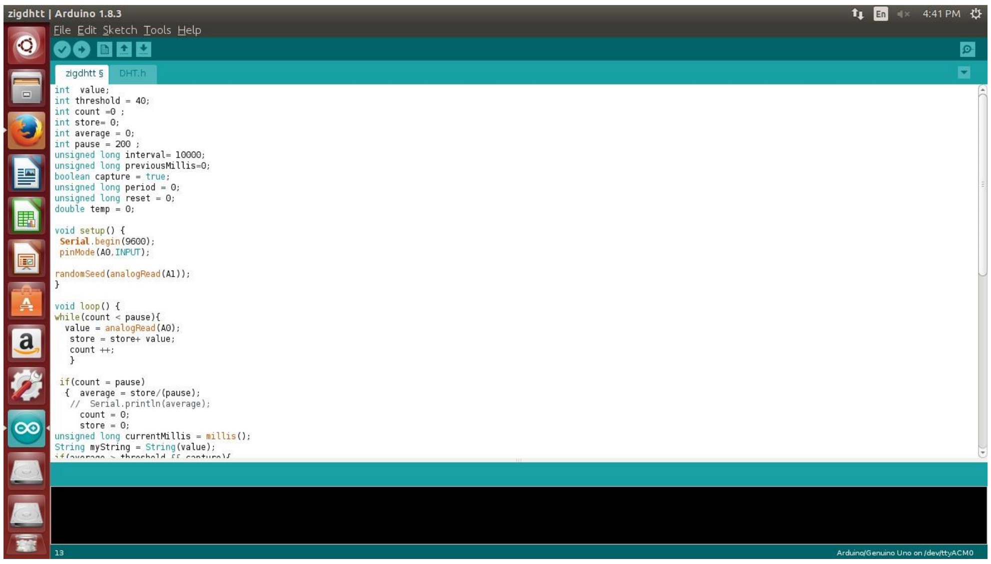


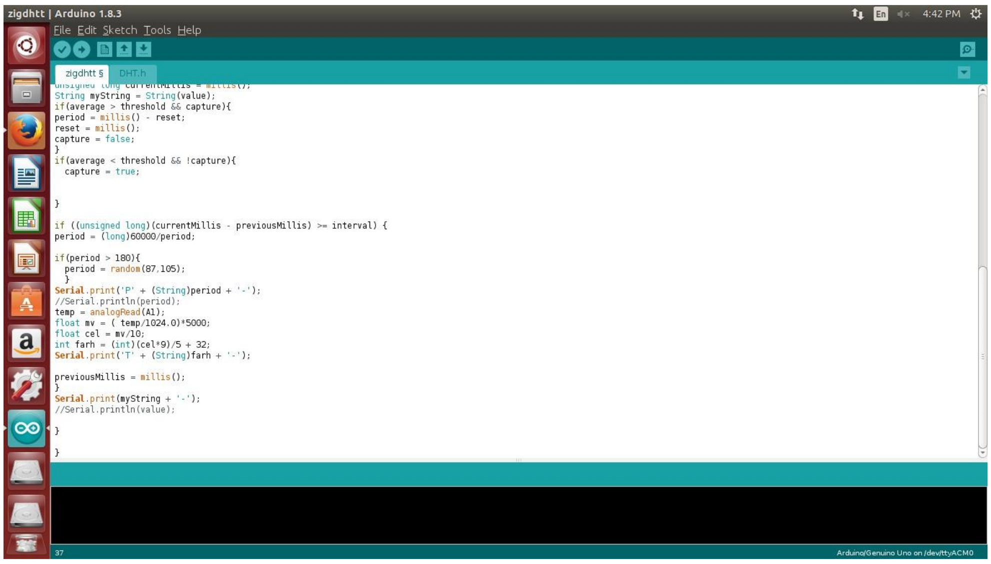
Figure 4: Board

The following code was used to send data from the micro controller to the android application.


zigdhtt | Arduino 1.8.3
File Edit Sketch Tools Help
zigdhtt §
DHT.h
4:41 PM
13
Arduino/Genuino Uno on /dev/ttyACM0

```
int  value;
int threshold = 40;
int count =0 ;
int store= 0;
int average = 0;
int pause = 200 ;
unsigned long interval= 10000;
unsigned long previousMillis=0;
boolean capture = true;
unsigned long period = 0;
unsigned long reset = 0;
double temp = 0;

void setup() {
 Serial.begin(9600);
 pinMode(A0,INPUT);

randomSeed(analogRead(A1));
}

void loop() {
while(count < pause){
  value = analogRead(A0);
   store = store+ value;
   count ++;
   }

 if(count = pause)
  {  average = store/(pause);
   //  Serial.println(average);
     count = 0;
     store = 0;
unsigned long currentMillis = millis();
String myString = String(value);
if(average > threshold && capture){
```

Figure 5: Arduino Code Snippet


zigdhtt | Arduino 1.8.3
File Edit Sketch Tools Help
zigdhtt §
DHT.h
4:42 PM
37
Arduino/Genuino Uno on /dev/ttyACM0

```
unsigned long currentMillis = millis();
String myString = String(value);
if(average > threshold && capture){
period = millis() - reset;
reset = millis();
capture = false;
}
if(average < threshold && !capture){
  capture = true;


}

if ((unsigned long)(currentMillis - previousMillis) >= interval) {
period = (long)60000/period;

if(period > 180){
  period = random(87,105);
  }
Serial.print('P' + (String)period + '-');
//Serial.println(period);
temp = analogRead(A1);
float mv = ( temp/1024.0)*5000;
float cel = mv/10;
int farh = (int)(cel*9)/5 + 32;
Serial.print('T' + (String)farh + '-');

previousMillis = millis();
}
Serial.print(myString + '-');
//Serial.println(value);

}

}
```

Figure 6: Arduino Code Snippet

4.2.3 Data Transmission Between WSN and Android Application

This was achieved using Bluetooth.

BLE- Bluetooth Low Energy or Bluetooth Smart is an important short-range communication technology. Its particularly famous in wearable product space and smartphone. It requires significantly reduced power consumption. It is more suitable for small chunks of data. It works on a Frequency band of 2.4GHz and covers a range of 50-150m and has a data rate of 1 Mbps. The modulation technique used is Frequency Hoping Spread Spectrum Technique.

Bluetooth vs ZigBee
Since we propose a smartphone centric architecture, Bluetooth becomes the most optimum transmission protocol as smartphones are by default Bluetooth enabled. Bluetooth module is cheaper compared to two ZigBee modules together. Bluetooth module is intended for frequent recharging whereas ZigBee is non-rechargeable. Bluetooth provides a maximum network speed of 1Mbps against ZigBee which provides a speed of only 250 Kbps. Bluetooth provides a range of 150 m but ZigBee provides only 70m.

4.3 Android Application

Temperature and heart rate values were sent to the android application which was then plotted.

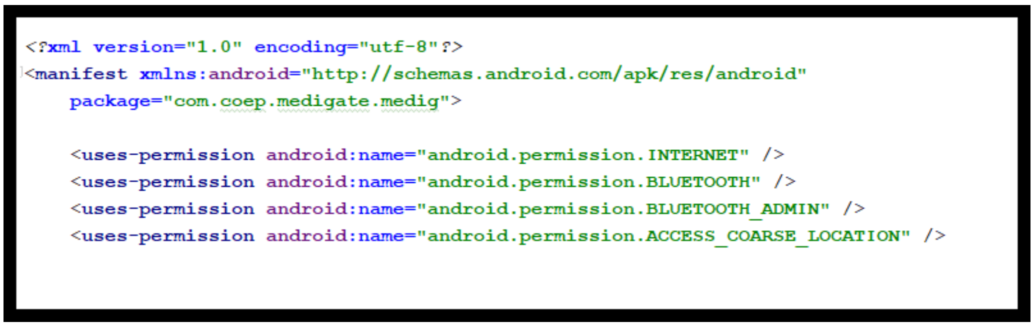
1. To receive data from the Wireless Sensor Network we use Bluetooth API. Android supports Bluetooth network stacks for wireless exchange of data. This functionalities are accessed using android Bluetooth API. The following actions were performed using the Bluetooth module:

   (a) Scan for other Bluetooth devices.

   (b) Connect to other devices through discovery service

   (c) Get data from the WSN.

   To use a Bluetooth API we first need to define permission in the android manifest of the project using Following this we create a Bluetooth handler. It reads data from the created socket. The data sent from the WSN is read from the socket. Multiple data were segregated using the keywords used to with the data set while transmitting from the WSN. This data is further sent via the message handler which plots and sends the data to the server.



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.coep.medigate.medig">

    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.BLUETOOTH" />
    <uses-permission android:name="android.permission.BLUETOOTH_ADMIN" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
```

Figure 7: Snippet of Manifest file

2. To plot the graph, we used the GraphView.

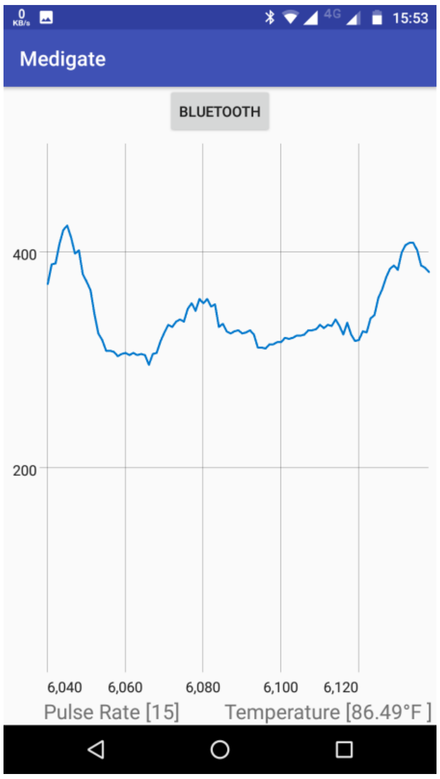
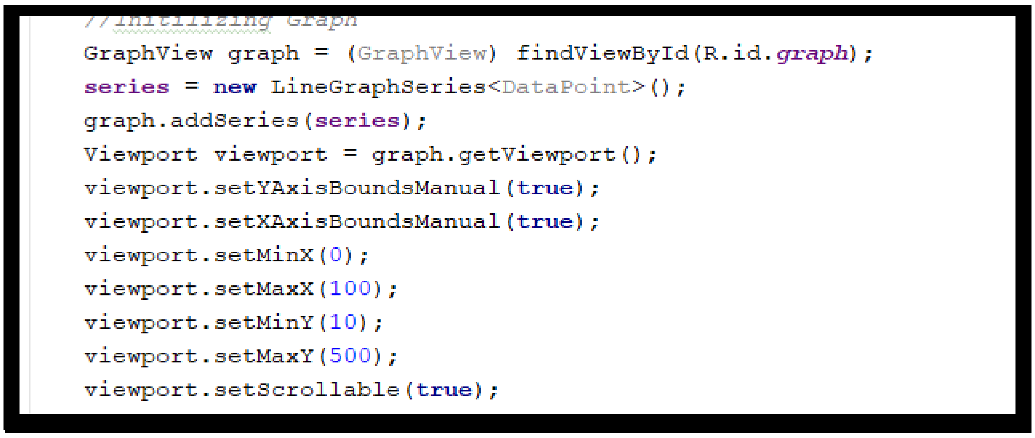


Figure 8: Plot of Heartrate Sensor Using GraphView

GraphView is an Open Source android library, which is used to programmatically create line, bar, point and other types of graphs. It comes with features like XML integration which is the key requirement as this is what enables us to use it with Android Studio. It also provides real time or live chart plotting with the help of auto scrolling. Draw multiple series of data simultaneously. It can be used to scroll scale and zoom the plot. Its also very customisable.



```
//Initializing Graph
GraphView graph = (GraphView) findViewById(R.id.graph);
series = new LineGraphSeries<DataPoint>();
graph.addSeries(series);
Viewport viewport = graph.getViewport();
viewport.setYAxisBoundsManual(true);
viewport.setXAxisBoundsManual(true);
viewport.setMinX(0);
viewport.setMaxX(100);
viewport.setMinY(10);
viewport.setMaxY(500);
viewport.setScrollable(true);
```

Figure 9: Snippet for initializing GraphView

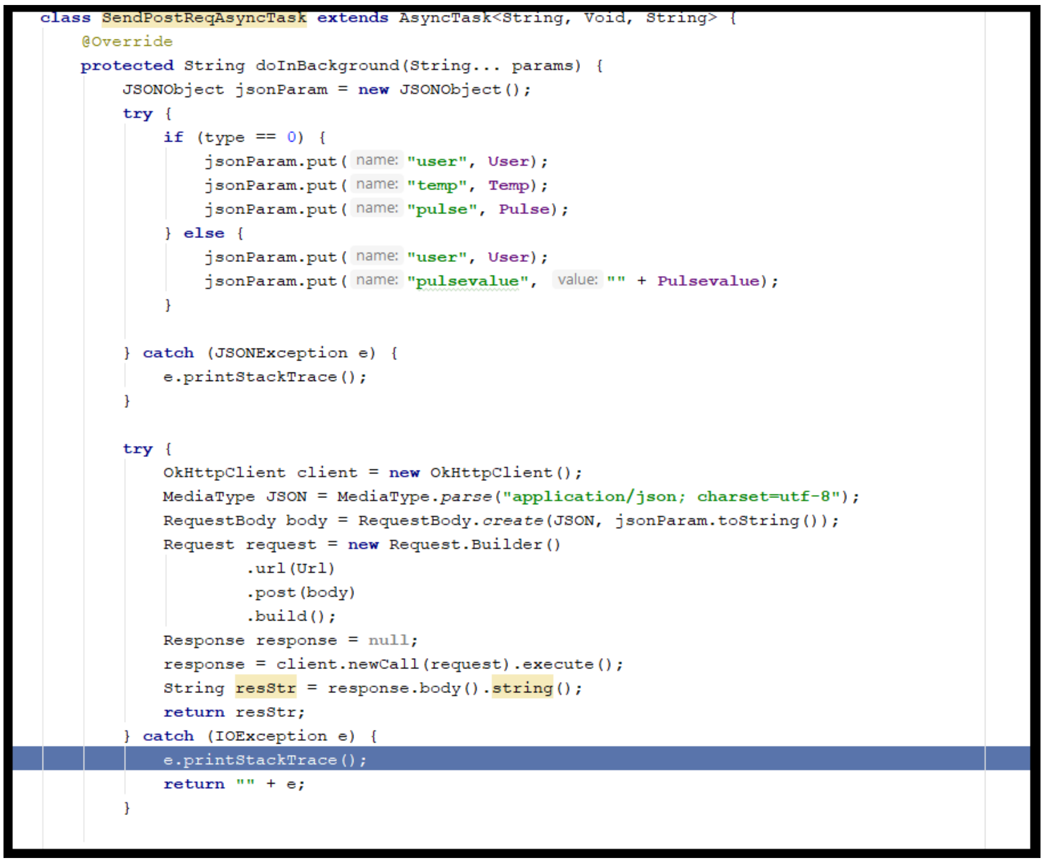
Graph is initialized to the above values. We set minimum and maximum values so as the set the origin for the graph.

4.3.1 Authentication

To provide authentication for the application, we built a login form. We provided two modes of logging in. One being email id based and the other is using GoogleSignInClient API. The GoogleSignInClient API allows the user to Sign In using the google account registered to that particular android phone. The GoogleSignInClient if used, gives us information such as Name, email id and profile picture. The other method to sign in is using an email id and password.

A form was created which asks for users for their details. If a doctor registers, a choice is given to him if he would want to authorize himself as a doctor to extend his services to the patients. Additional information such as doctor id and specialization was requested. The username was stored for future login to the app to avoid going through the login procedure repeatedly upon opening the application. This username also helps us to uniquely identify and extract the data from various tables using object relational mapping which is a key feature of server side software, Django.

4.3.2 Server-Client Communication



```java
class SendPostReqAsyncTask extends AsyncTask<String, Void, String> {
    @Override
    protected String doInBackground(String... params) {
        JSONObject jsonParam = new JSONObject();
        try {
            if (type == 0) {
                jsonParam.put( name: "user", User);
                jsonParam.put( name: "temp", Temp);
                jsonParam.put( name: "pulse", Pulse);
            } else {
                jsonParam.put( name: "user", User);
                jsonParam.put( name: "pulsevalue",  value: "" + Pulsevalue);
            }

        } catch (JSONException e) {
            e.printStackTrace();
        }

        try {
            OkHttpClient client = new OkHttpClient();
            MediaType JSON = MediaType.parse("application/json; charset=utf-8");
            RequestBody body = RequestBody.create(JSON, jsonParam.toString());
            Request request = new Request.Builder()
                    .url(Url)
                    .post(body)
                    .build();
            Response response = null;
            response = client.newCall(request).execute();
            String resStr = response.body().string();
            return resStr;
        } catch (IOException e) {
            e.printStackTrace();
            return "" + e;
        }
```

Figure 10: Snippet of AsyncTask()

Data was sent from the android device to the server using JSON. JAVA SCRIPT OBJECT Notation (JSON). Its an open standard file format in human readable text, to transmit objects consisting of key and value pairs. It is a language independent format. This is a very strong feature of JSON which enables us to make the client independent of the server side language. AsyncTask is a class which enables us to implement functions such as doInBackground() and onPostExecute().

(a) The doInBackground enables us to continuously send data to the server without interrupting or providing latency in reading data from the WSN. We have used OkHttp client. Which is an HTTP client for android and java applications. HTTP is the way modern application works. We exchange data and media through it with

the server. It efficiently makes the objects load faster and saves bandwidth too.
OkHttp is an HTTP client thats efficient by default:

i. HTTP/2 support allows all requests to the same host to share a socket.
ii. Connection pooling reduces request latency (if HTTP/2 isnt available).
iii. Transparent GZIP shrinks download sizes.
iv. Response caching avoids the network completely for repeat requests. OkHttp perseveres when the network is troublesome: it will silently recover from common connection problems. If your service has multiple IP addresses OkHttp will attempt alternate addresses if the first connect fails. This is necessary for IPv4+IPv6 and for services hosted in redundant data centers. OkHttp initiates new connections with modern TLS features (SNI, ALPN), and falls back to TLS 1.0 if the handshake fails. Using OkHttp is easy. Its request/response API is designed with fluent builders and immutability. It supports both synchronous blocking calls and async calls with callbacks. OkHttp supports Android 2.3 and above. For Java, the minimum requirement is 1.7.[6]

(b) The onPostExecute() function helps to schedule tasks as and when a doInBackground() task is executed. We have used this function to get feedback from the server such as complete post, missing attributes, socket creation error, json stack error, input/output errors etc.

4.3.3 Django

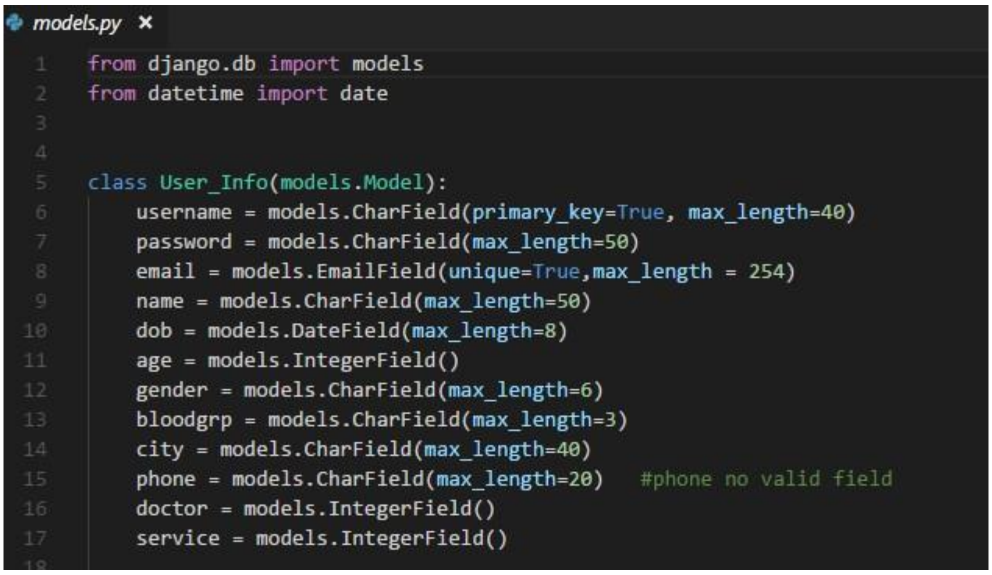
Python VS PHP

We had two languages to use for web development and backend development. Python along with Django (python web framework) is faster than PHP. Compared to other programming languages like C++, Perl etc. its easier to learn python and PHP. Detailed documentation is available for both the languages. Python and PHP both are open source, so the source code is available readily and has huge community support. All the major operating systems like MacOS, Windows, Linux have IDEs for python and PHP. Python is an aspect-oriented language while PHP is an object-oriented language. So, the developer has to create separate models and then link those models in python while in PHP the developer creates an object which executes actions as per the input given by the user.

The syntax differs for both the languages. Python has a syntax based on the separation of the code with spaces and tabs, this increases the speed of coding, but the chances of careless errors increases as well. PHP syntax is same as that of C. It uses curly brackets, additional characters and operators while writing the code. White spaces are neglected in PHP while compiling code. Additional libraries are loaded in Python using the special packages available in python. PHP requires loading of additional libraries manually. The scope of python is not limited to the web development only unlike PHP. Python can be used for processing and security purpose also. Major mobile platform like iOS, android, windows develops web application with the help of python.

Django is a free and open source web application framework, written in Python. A web framework is a set of components that helps you to develop websites faster and easier. When you're building a website, you always need a similar set of components: a way to handle user authentication (signing up, signing in, signing out), a management panel for your website, forms, a way to upload files, etc. Luckily for you, other people long ago noticed that web developers face similar problems when building a new site, so they teamed up and created frameworks (Django being one of them) that give you ready-made components to use. Frameworks exist to save you from having to reinvent the wheel and to help alleviate some of the overhead when youre building a new site.

The data at the web application can be structured and manipulated using the abstraction layer called models provided by Django. We have created different classes for patients, doctor, patient information, sensor data, etc. inside the model. These models can be used by editing the settings file and adding the name of the model to the installed apps. The list of database fields defined in a model are very important. The class created for patient information is given below.



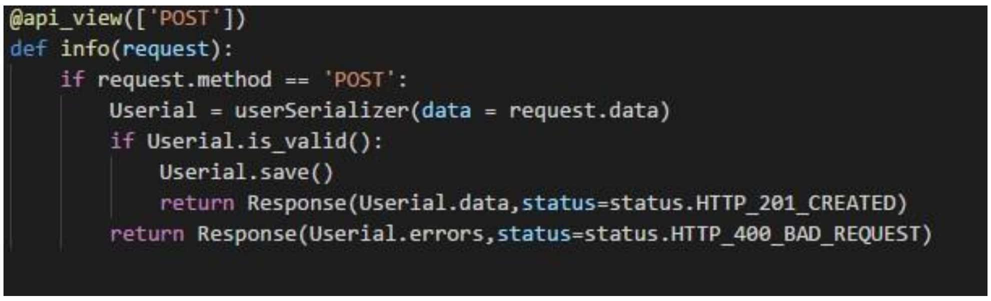
```python
from django.db import models
from datetime import date


class User_Info(models.Model):
    username = models.CharField(primary_key=True, max_length=40)
    password = models.CharField(max_length=50)
    email = models.EmailField(unique=True,max_length = 254)
    name = models.CharField(max_length=50)
    dob = models.DateField(max_length=8)
    age = models.IntegerField()
    gender = models.CharField(max_length=6)
    bloodgrp = models.CharField(max_length=3)
    city = models.CharField(max_length=40)
    phone = models.CharField(max_length=20)   #phone no valid field
    doctor = models.IntegerField()
    service = models.IntegerField()
```

Figure 11: User Model

The users request can be processed and the response can be returned by using the concept of view by Django. A python module called URL configuration is created which is a mapping between URL path expressions to python functions. Django scans each URL pattern, and import the view from the matched URL. The view is a simple python function. The view gets passed the following arguments:

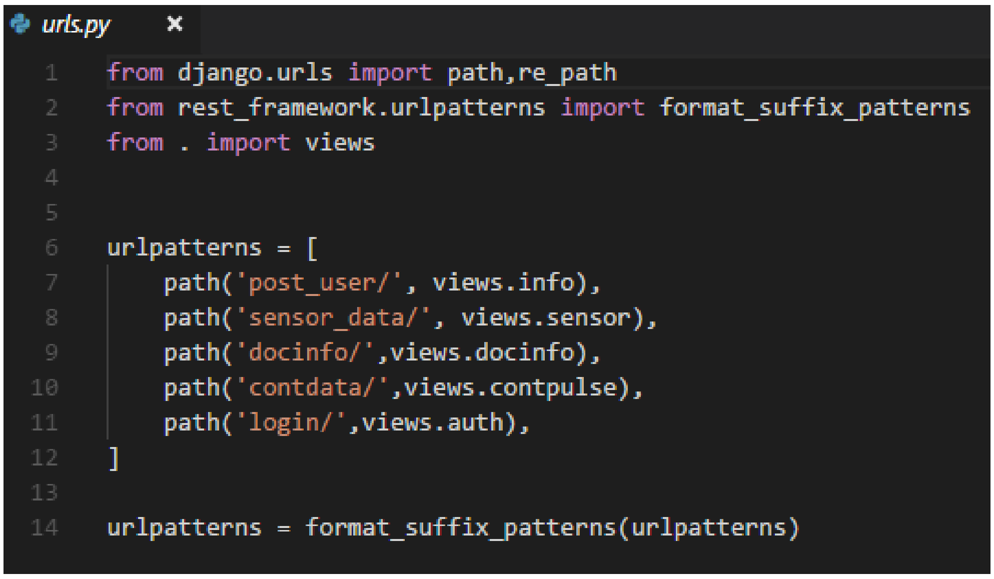
(a) An instance of HttpRequest.

(b) The matches from the regular expression are provided as positional arguments in case if the matched URL pattern returned no named groups. The view for the class patient information is given below.



```python
@api_view(['POST'])
def info(request):
    if request.method == 'POST':
        Userial = userSerializer(data = request.data)
        if Userial.is_valid():
            Userial.save()
            return Response(Userial.data,status=status.HTTP_201_CREATED)
        return Response(Userial.errors,status=status.HTTP_400_BAD_REQUEST)
```

Figure 12: Views for Patient Class



urls.py

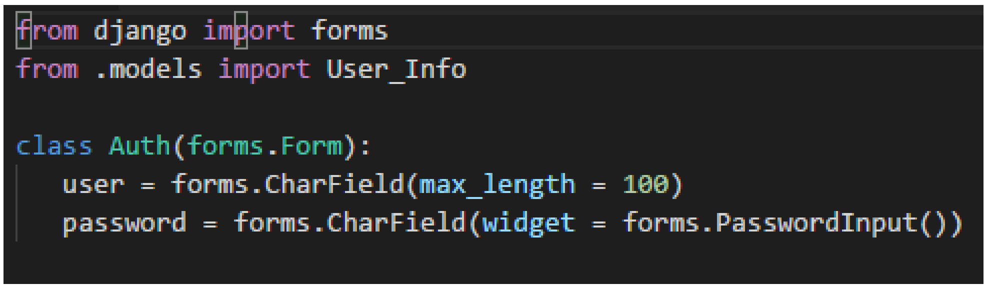
```
from django.urls import path,re_path
from rest_framework.urlpatterns import format_suffix_patterns
from . import views


urlpatterns = [
    path('post_user/', views.info),
    path('sensor_data/', views.sensor),
    path('docinfo/',views.docinfo),
    path('contdata/',views.contpulse),
    path('login/',views.auth),
]

urlpatterns = format_suffix_patterns(urlpatterns)
```

Figure 13: URL Patterns for Patient Class

A designer-friendly syntax is provided by the template layer for giving the information to be presented to the user. A static part of the desired HTML output and some special syntax describing how dynamic content will be inserted is included in the template.

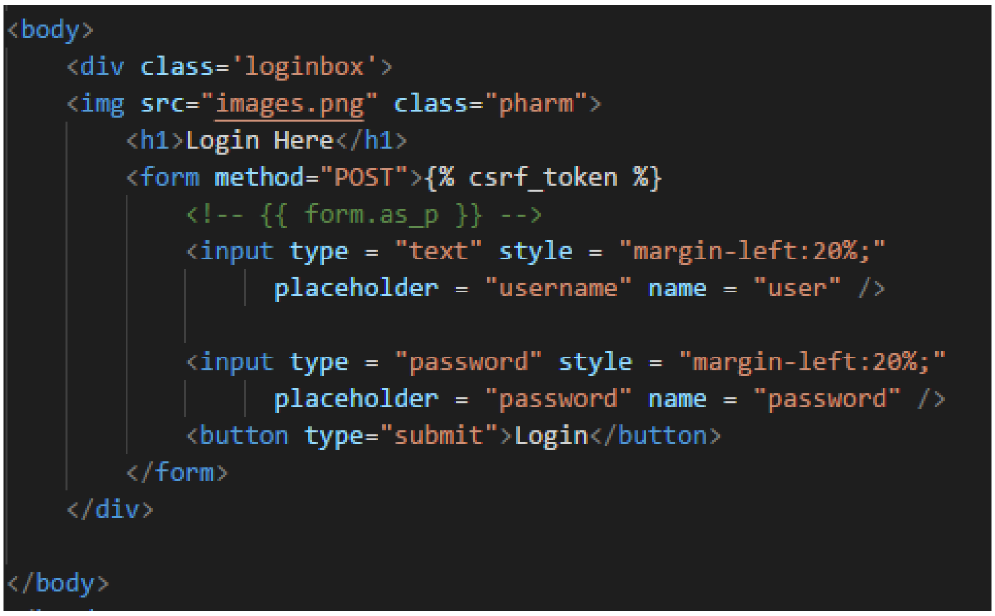
A range of tools and libraries is provided by Django which helps in building forms to accept input from site visitors, and then process and respond to the input.

A form is a collection of elements which lies inside ¡form¿¡/forms¿ that allows users to enter text, select options, manipulate objects or so on and then send that information back to server.



```
from django import forms
from .models import User_Info

class Auth(forms.Form):
    user = forms.CharField(max_length = 100)
    password = forms.CharField(widget = forms.PasswordInput())
```

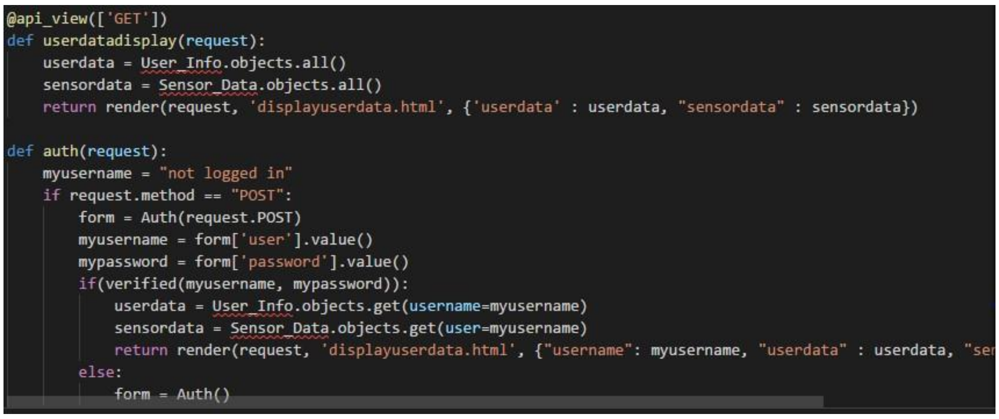
Figure 14: Forms for Patient Class



```html
<body>
    <div class='loginbox'>
    <img src="images.png" class="pharm">
        <h1>Login Here</h1>
        <form method="POST">{% csrf_token %}
            <!-- {{ form.as_p }} -->
            <input type = "text" style = "margin-left:20%;"
                placeholder = "username" name = "user" />

            <input type = "password" style = "margin-left:20%;"
                placeholder = "password" name = "password" />
            <button type="submit">Login</button>
        </form>
    </div>

</body>
```

Figure 15: HTML Code for Login Page

GET and POST are the HTTP methods to be used while dealing with the forms. The browser bundles up the Djangos login form data and returns it using the POST method. The form data is then encoded for transmission, send to the server, and then received as response. GET bundles the submitted data into a string, and uses this to make a URL. URL consists of the address where the data must be sent, and also the data keys and values. POST is used in case if the request changes the state of the system. GET is used only for the systems where request does not affect the state of the system.

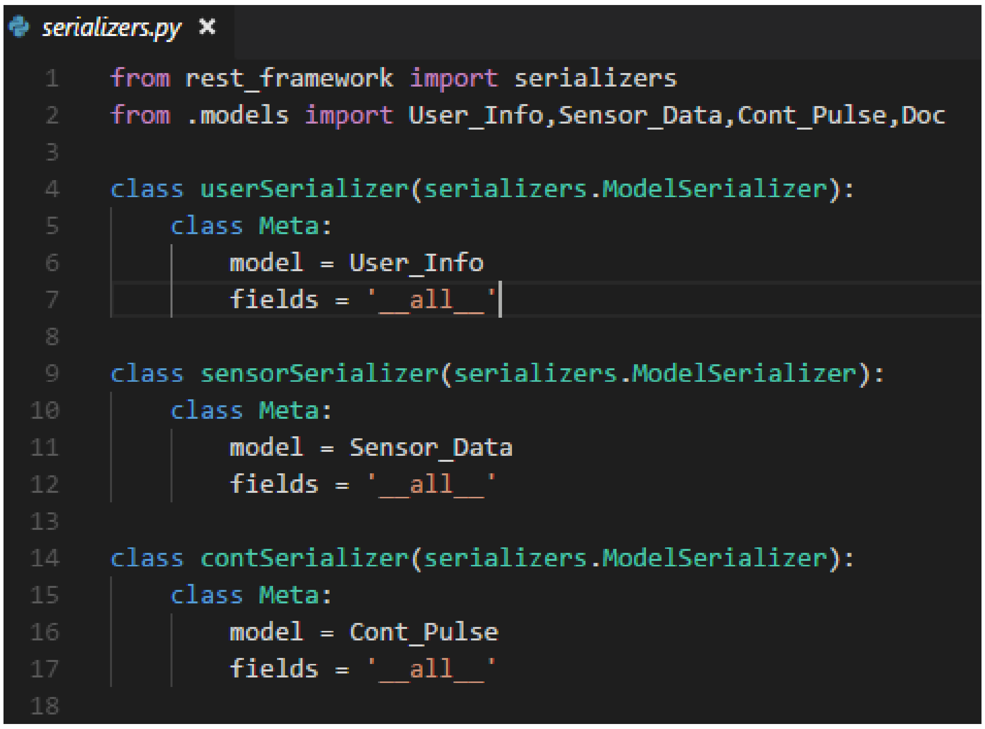


```
@api_view(['GET'])
def userdatadisplay(request):
    userdata = User_Info.objects.all()
    sensordata = Sensor_Data.objects.all()
    return render(request, 'displayuserdata.html', {'userdata' : userdata, "sensordata" : sensordata})

def auth(request):
    myusername = "not logged in"
    if request.method == "POST":
        form = Auth(request.POST)
        myusername = form['user'].value()
        mypassword = form['password'].value()
        if(verified(myusername, mypassword)):
            userdata = User_Info.objects.get(username=myusername)
            sensordata = Sensor_Data.objects.get(user=myusername)
            return render(request, 'displayuserdata.html', {"username": myusername, "userdata" : userdata, "se
        else:
            form = Auth()
```

Figure 16: GET and POST HTTP methods

Django prepares and restructure data and makes it ready for rendering. It creates HTML forms for the data. Django receives, and process submitted forms and data from the client.

Django models can be translated from one format to other formats using Djangos serialization framework. The other formats are usually text-based and can be used for sending data over a wire. The serializer can handle any format (text-based or not).

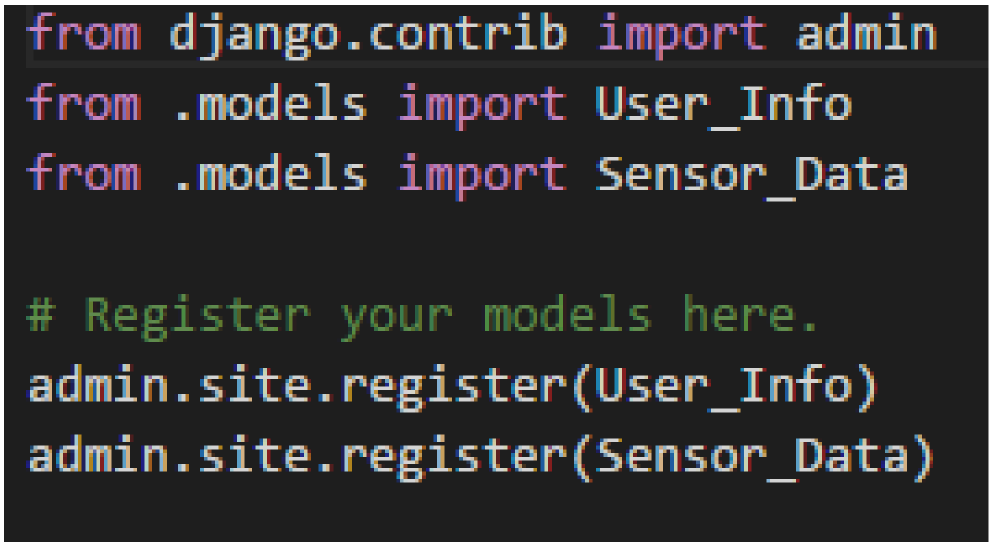


serializers.py

```python
from rest_framework import serializers
from .models import User_Info,Sensor_Data,Cont_Pulse,Doc

class userSerializer(serializers.ModelSerializer):
    class Meta:
        model = User_Info
        fields = '__all__'

class sensorSerializer(serializers.ModelSerializer):
    class Meta:
        model = Sensor_Data
        fields = '__all__'

class contSerializer(serializers.ModelSerializer):
    class Meta:
        model = Cont_Pulse
        fields = '__all__'

```

Figure 17: Serializers Framework

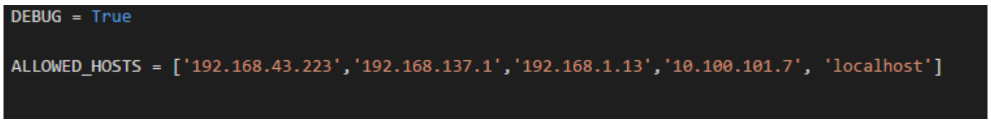
The automatic admin interface of Django is used which reads meta-data from the models to provide quick, model-centric interface where the user can manage the content on the site.The admin.py is not the complete front end.



```
from django.contrib import admin
from .models import User_Info
from .models import Sensor_Data

# Register your models here.
admin.site.register(User_Info)
admin.site.register(Sensor_Data)
```

Figure 18: Admin Interface

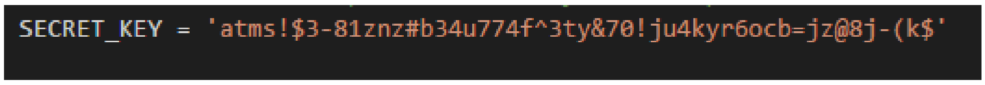
The settings.py file is used to configure things like Django applications, databases, templates and middleware. This file is known as the central place for all the configurations for all Django projects. The allowed_hosts variable in settings.py is to validate the requests HTTP host header. To enhance security DEBUG is set to FALSE and allowed_hosts is kept empty. This makes the Django to refuse to serve requests and respond with HTTP 400 bad request pages.



```
DEBUG = True

ALLOWED_HOSTS = ['192.168.43.223','192.168.137.1','192.168.1.13','10.100.101.7', 'localhost']
```

Figure 19: Settings File

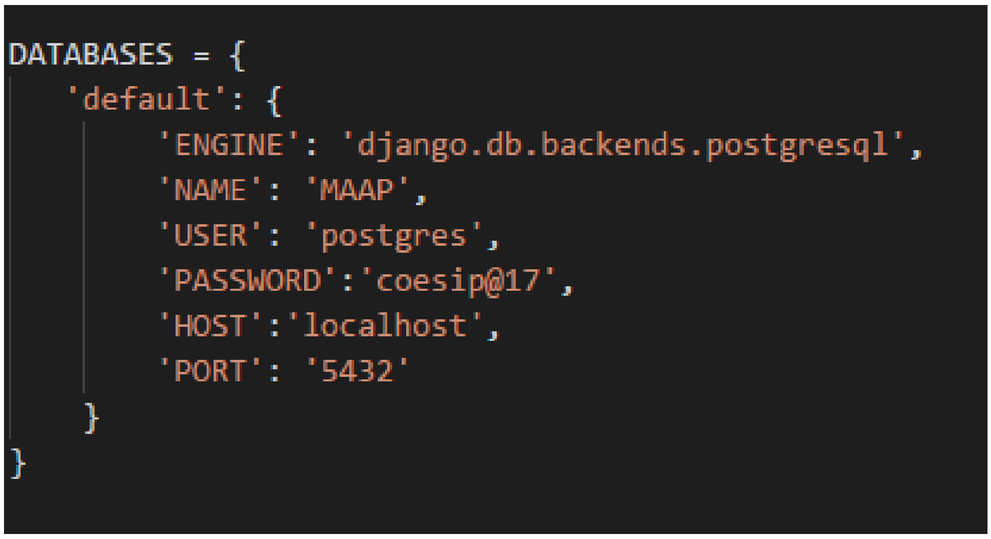
Secret key is used to cryptographically protect any sensitive data structure in a Django project. The secret key is provided to any of these sensitive data structure before the Django sends it to the users on the internet. Whenever the data structure is called to execute any action, Django re-checks these sensitive data structures against the secret key again. If there is any tampering on the data structure then the Django stops running the process.



```
SECRET_KEY = 'atms!$3-81znz#b34u774f^3ty&70!ju4kyr6ocb=jz@8j-(k$'
```

Figure 20: Secret Key

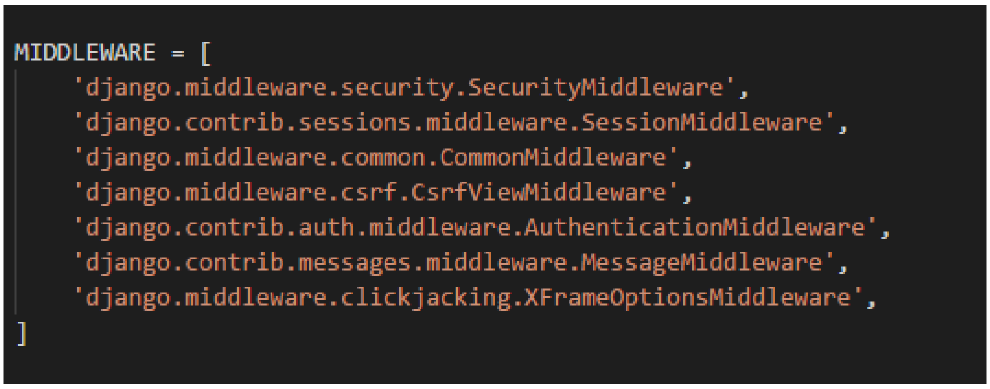
Databases contains settings for all the databases to be used with Django. The parameters needed for connecting to PostgreSQL database back-ends are given below.



```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'MAAP',
        'USER': 'postgres',
        'PASSWORD':'coesip@17',
        'HOST':'localhost',
        'PORT': '5432'
    }
}
```

Figure 21: Databases

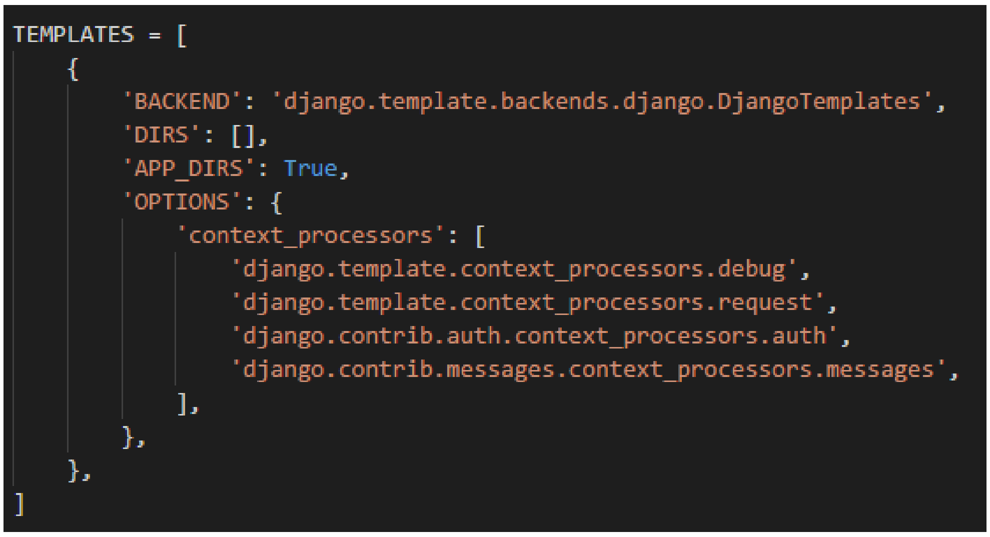
Middleware is light low-level system used to alter Djangos input or output globally. Its a framework used into Djangos request/response processing. Authentication middleware in Django associates users with requests using sessions. The middleware component is added in Django settings to activate a middleware. The order in a middleware should be considered because sometimes a middleware can depend on other middleware.



```
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
```

Figure 22: Middleware

Templates contains item which together forms the setting for each template engine used by the Django.



```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]
```

Figure 23: Templates

The template engine looks for template source files in the directories DIRS:[ ]. The extra parameters to be passed to the template backend are stored in OPTIONS. The available parameters in OPTIONS varies according to the template backend.

4.4 Hardware/Software Used

4.4.1 Hardware

1. **Bluetooth Module**

   HC05 module 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 04External single chip Bluetooth system with CMOS technology and with AFH (Adaptive Frequency Hopping Feature). [7] Key Features of Bluetooth Module are: Hardware Features

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

   Software Features

   (a) Slave default Baud rate: 9600, Data bits:8, Stop bit:1,Parity:No parity.
   (b) Autoconnect to the last device on power as default.
   (c) Permit pairing device to connect as default.
   (d) Autopairing PINCODE:1234 as default.

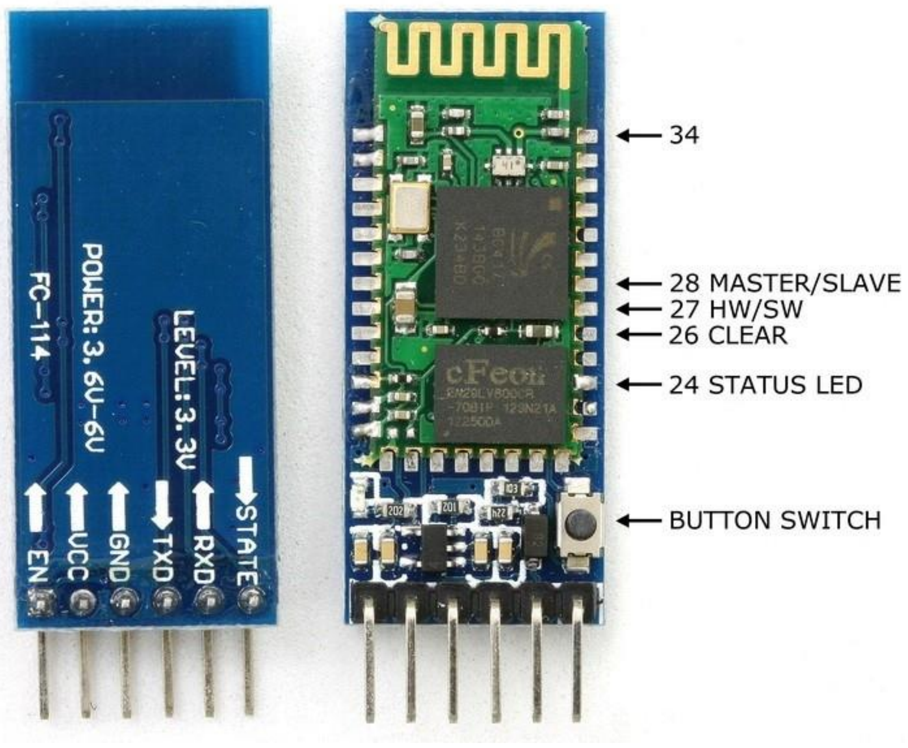
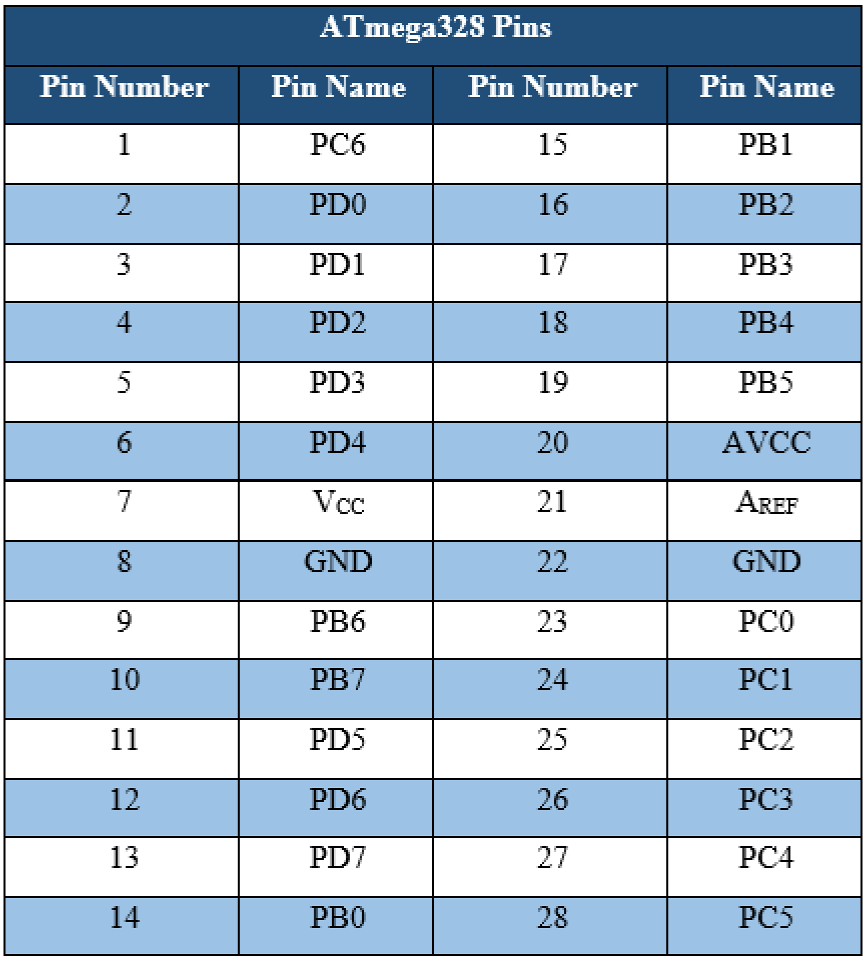


Figure 24: Bluetooth Module

2. **Atmega 328 PCB** It is a low-power CMOS 8-bit micro-controller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega328/P achieves throughputs close to 1MIPS per MHz. This empowers system designer to optimise the device for power consumption versus processing speed. The Key features are[8]: The various pins and functionalities



ATmega328 Pins

Pin Number	Pin Name	Pin Number	Pin Name
1	PC6	15	PB1
2	PD0	16	PB2
3	PD1	17	PB3
4	PD2	18	PB4
5	PD3	19	PB5
6	PD4	20	AVCC
7	Vcc	21	AREF
8	GND	22	GND
9	PB6	23	PC0
10	PB7	24	PC1
11	PD5	25	PC2
12	PD6	26	PC3
13	PD7	27	PC4
14	PB0	28	PC5

Figure 25: Pin Diagram of Atmega 328P

3. Sensors(Temperature, Heart rate sensor)

(a) Temperature Sensor (LM35) The LM35 series are precision integrated circuit temperature sensors, whose output voltage is linearly proportional to the Celsius (Centigrade) temperature. The LM35 thus has an advantage over linear temperature sensors calibrated in Kelvin, as the user is not required to subtract a large constant voltage from its output to obtain convenient Centigrade scaling. The LM35 does not require any external calibration or trimming to provide typical accuracies of 14C at room temperature and 34C over a full 55 to +150C temperature range.

(b) Heart rate Sensor It uses the optical power variation to sense the change of volume of blood which is directly proportional to pulse rate of the patient.

4.4.2 Software

1. Android Studio

Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development.It is available for download on Windows, macOS and Linux based operating systems It is a replacement for the Eclipse Android Development Tools (ADT) as primary IDE for native Android application development.

Figure 26: Android Studio

Key Features[9]:-

(a) Gradle-based build support

(b) Android-specific refactoring and quick fixes

(c) Lint tools to catch performance, usability, version compatibility and other problems

(d) ProGuard integration and app-signing capabilities

(e) Template-based wizards to create common Android designs and components

(f) A rich layout editor that allows users to drag-and-drop UI components, option to preview layouts on multiple screen configurations[16]

(g) Support for building Android Wear apps

(h) Built-in support for Google Cloud Platform, enabling integration with Firebase Cloud Messaging (Earlier 'Google Cloud Messaging') and Google App Engine[17]

(i) Android Virtual Device (Emulator) to run and debug apps in the Android studio.

2. EAGLE

It is a scriptable electronic design automation (EDA) application with schematic capture, printed circuit board (PCB) layout, auto-router and computer-aided manufacturing (CAM) features. EAGLE stands for Easily Applicable Graphical Layout Editor

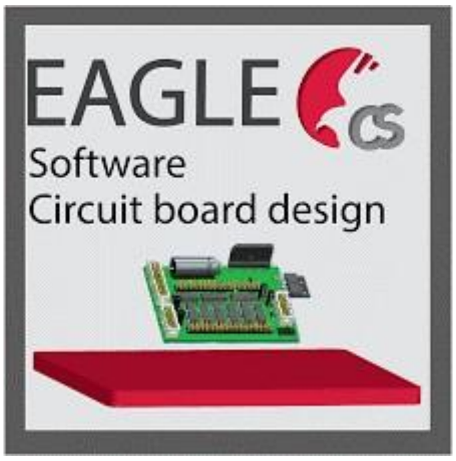


Figure 27: Eagle

Key Features[10]:-

(a) EAGLE contains a schematic editor, for designing circuit diagrams. Schematics are stored in files with .SCH extension, parts are defined in device libraries with .LBR extension. Parts can be placed on many sheets and connected together through ports.

(b) The PCB layout editor stores board files with the extension .BRD. It allows back-annotation to the schematic and auto-routing to automatically connect traces based on the connections defined in the schematic.

(c) EAGLE saves Gerber and PostScript layout files as well as Excellon and Sieb and Meyer drill files. These are standard file formats accepted by PCB fabrication companies, but given EAGLE's typical user base of small design firms and hobbyists, many PCB fabricators and assembly shops also accept EAGLE board files (with extension .BRD) directly to export optimized production files and pick-and-place data themselves.

(d) EAGLE provides a multi-window graphical user interface and menu system for editing, project management and to customize the interface and design parameters. The system can be controlled via mouse, keyboard hotkeys or by entering specific commands at an embedded command line. Multiple repeating commands can be combined into script files (with file extension .SCR). It is also possible to explore design files utilizing an EAGLE-specific object-oriented programming language (with extension .ULP).

5 Results

5.1 Collection of Data Through Wireless Sensor Network

The following figure shows a plot of the sensor data once the user has logged

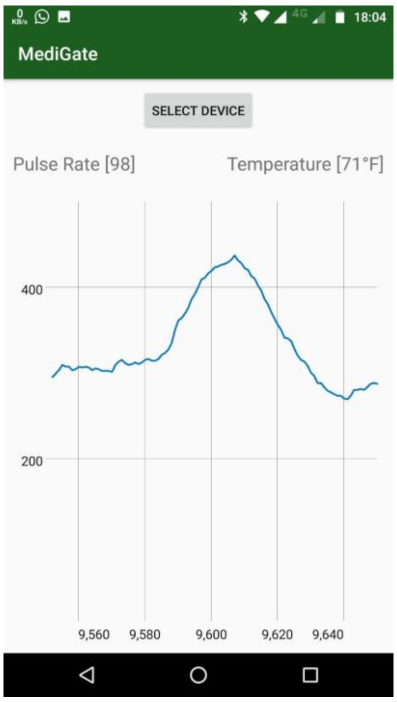
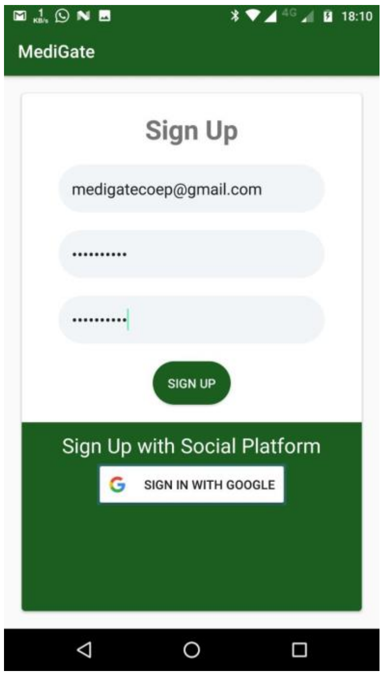


Figure 28: Plot of Heart Rate and Sensor Values

5.2 Android Application

The data from the WSN is sent to the application To view this data the user first needs to sign up if he hasn't already. The user, if a doctor, can check in the doctor field and he shall be redirected to a separate form which also enquires about his specialisation. The patient is redirected to another form which collects his basic health details. After logging in they can view the plot of the sensor data.


MediGate

Sign Up

medigatecoep@gmail.com

••••••••••

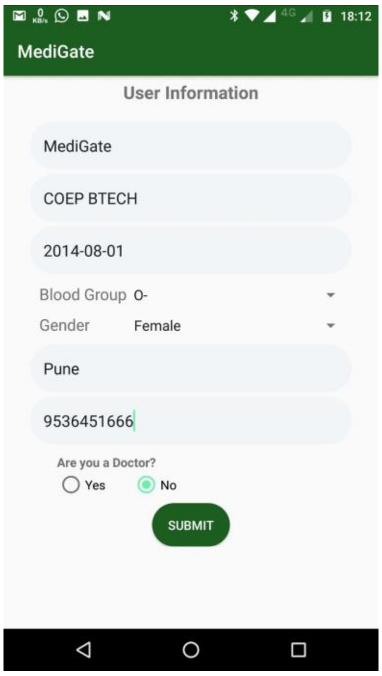
••••••••••

SIGN UP

Sign Up with Social Platform

SIGN IN WITH GOOGLE

(a) Sign Up Page


MediGate

User Information

MediGate

COEP BTECH

2014-08-01

Blood Group O-

Gender Female

Pune

9536451666

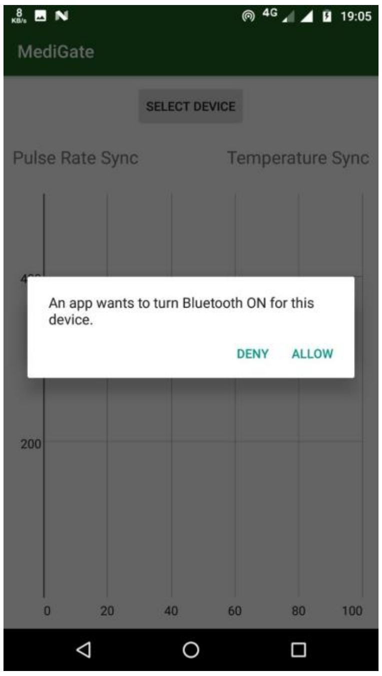
Are you a Doctor?

Yes [✓ No]

SUBMIT

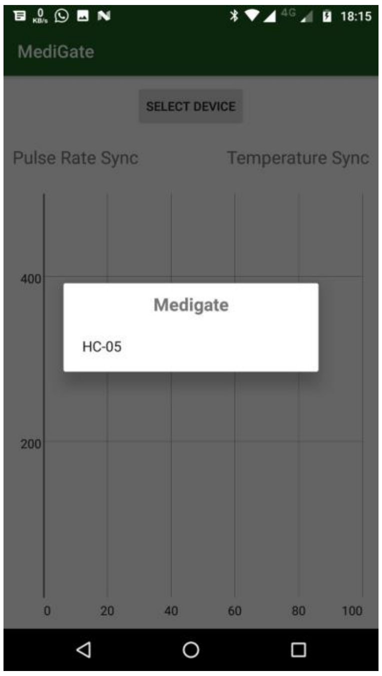
(b) Form for Patient Login

Once the profile has been created, the patient can view the graph plot and values by first connecting to a Bluetooth device.The application first asks permission to use Bluetooth.Then, an option of connecting to already paired devices is given to the user.



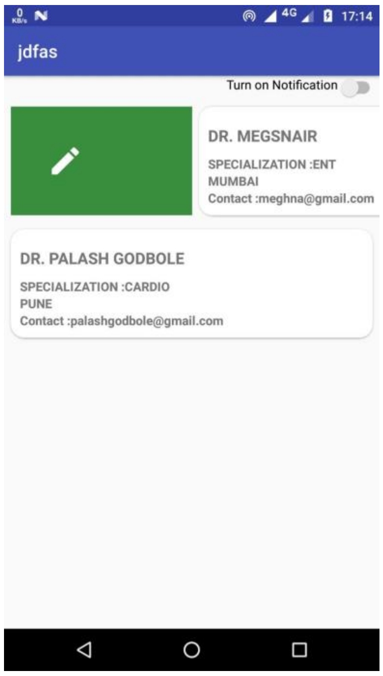
An app wants to turn Bluetooth ON for this device.

(c) Permission to turn on Bluetooth



(d) List of discovered devices

To access the list of doctors who are available in the particular hospital one can click on the doctor list and choose the doctor they want to consult by swiping right. On doing so the doctor receives a notification asking him if he'd like to extnd his service to the particular patient. To which he can respond by clicking on the add button.

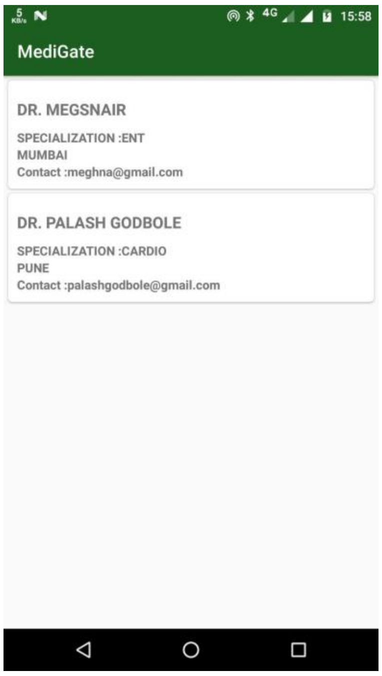

jdfas

Turn on Notification

DR. MEGSNAIR
SPECIALIZATION :ENT
MUMBAI
Contact :meghna@gmail.com

DR. PALASH GODBOLE
SPECIALIZATION :CARDIO
PUNE
Contact :palashgodbole@gmail.com

(e) Doctor Request


MediGate

DR. MEGSNAIR
SPECIALIZATION :ENT
MUMBAI
Contact :meghna@gmail.com

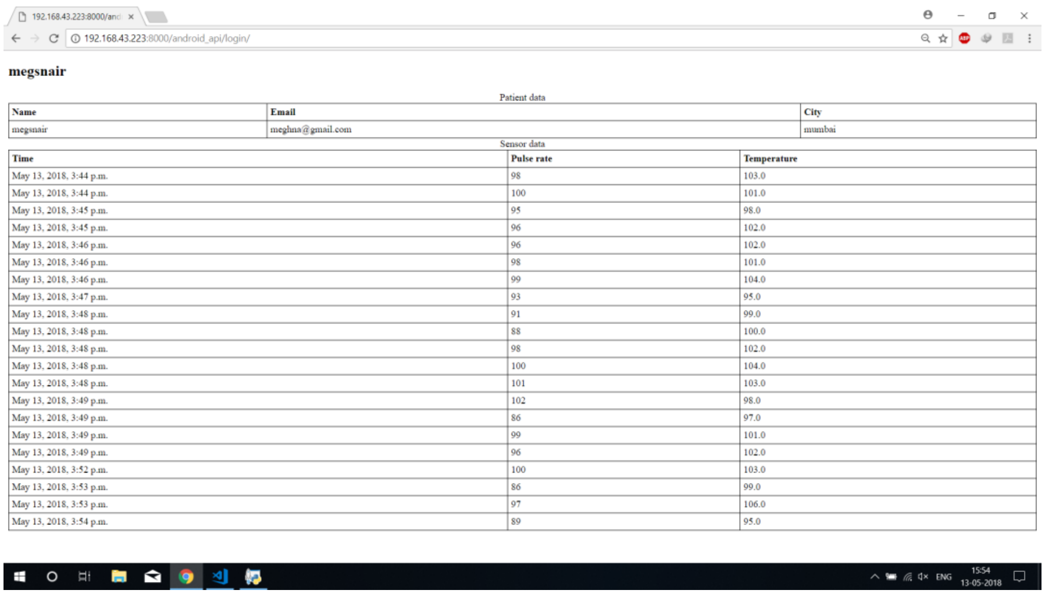
DR. PALASH GODBOLE
SPECIALIZATION :CARDIO
PUNE
Contact :palashgodbole@gmail.com

(f) Doctor's Profile

When the request is accepted by the doctor, the patient receives a notification suggesting that the doctor has acknowledged his request and will provide him with the necessary treatment. We have also provided an alert system. In any case of emergency, the user can press the alert button. This will ring an alarm on the patient's phone alerting everybody around him. It also sends an alarm to notify every doctor who has accepted the patients request along with the patient's current location.

5.3 Web Server

The sensor data collected from the patient is sent from the app to the web server. The web server shows a logged view of the data values stored in a tabulated form for the different users.



192.168.43.223:8000/android_api/login/

megsnair

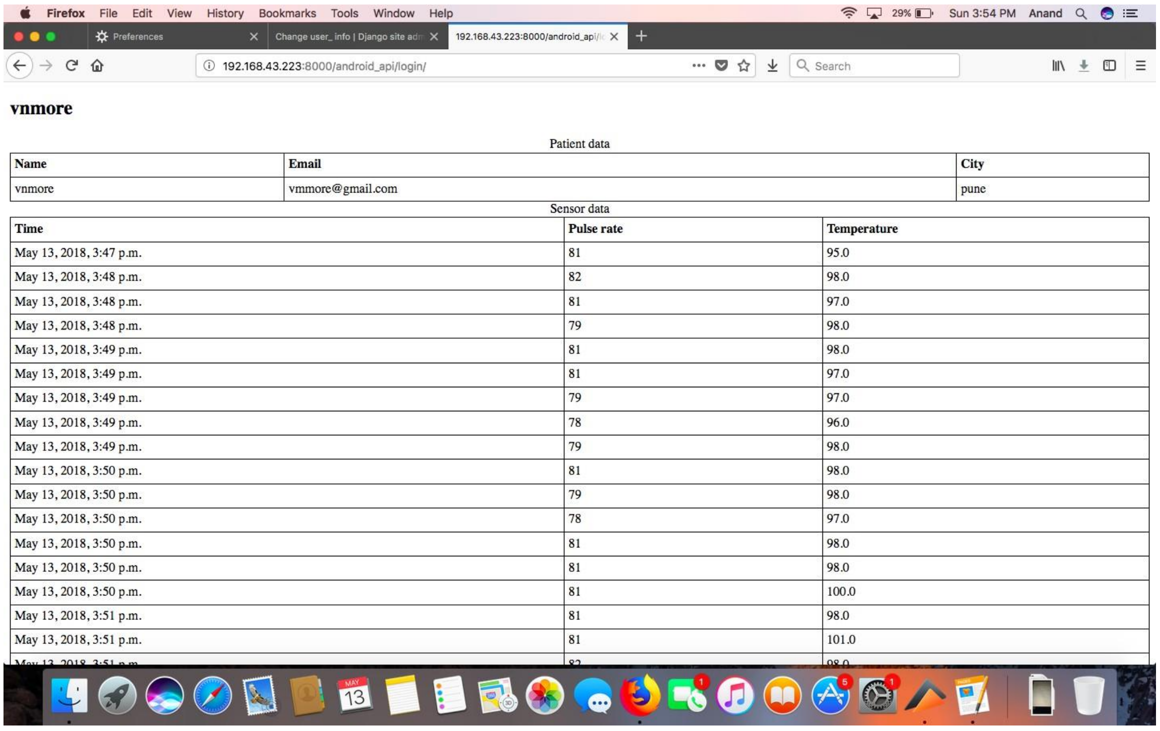
Patient data

Name	Email	City
megsnair	meghna@gmail.com	mumbai

Sensor data

Time	Pulse rate	Temperature
May 13, 2018, 3:44 p.m.	98	103.0
May 13, 2018, 3:44 p.m.	100	101.0
May 13, 2018, 3:45 p.m.	95	98.0
May 13, 2018, 3:45 p.m.	96	102.0
May 13, 2018, 3:46 p.m.	96	102.0
May 13, 2018, 3:46 p.m.	98	101.0
May 13, 2018, 3:46 p.m.	99	104.0
May 13, 2018, 3:47 p.m.	93	95.0
May 13, 2018, 3:48 p.m.	91	99.0
May 13, 2018, 3:48 p.m.	88	100.0
May 13, 2018, 3:48 p.m.	98	102.0
May 13, 2018, 3:48 p.m.	100	104.0
May 13, 2018, 3:48 p.m.	101	103.0
May 13, 2018, 3:49 p.m.	102	98.0
May 13, 2018, 3:49 p.m.	86	97.0
May 13, 2018, 3:49 p.m.	99	101.0
May 13, 2018, 3:49 p.m.	96	102.0
May 13, 2018, 3:52 p.m.	100	103.0
May 13, 2018, 3:53 p.m.	86	99.0
May 13, 2018, 3:53 p.m.	97	106.0
May 13, 2018, 3:54 p.m.	89	95.0

Figure 29: Data from User1



192.168.43.223:8000/android_api/login/

vnmore

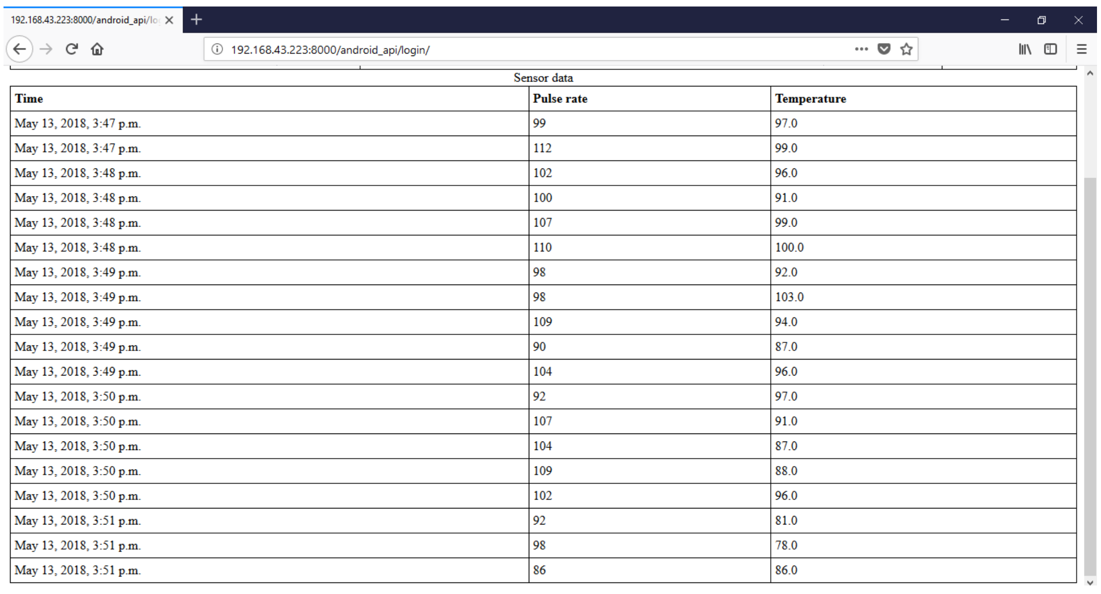
Patient data

Name	Email	City
vnmore	vmmore@gmail.com	pune

Sensor data

Time	Pulse rate	Temperature
May 13, 2018, 3:47 p.m.	81	95.0
May 13, 2018, 3:48 p.m.	82	98.0
May 13, 2018, 3:48 p.m.	81	97.0
May 13, 2018, 3:48 p.m.	79	98.0
May 13, 2018, 3:49 p.m.	81	98.0
May 13, 2018, 3:49 p.m.	81	97.0
May 13, 2018, 3:49 p.m.	79	97.0
May 13, 2018, 3:49 p.m.	78	96.0
May 13, 2018, 3:49 p.m.	79	98.0
May 13, 2018, 3:50 p.m.	81	98.0
May 13, 2018, 3:50 p.m.	79	98.0
May 13, 2018, 3:50 p.m.	78	97.0
May 13, 2018, 3:50 p.m.	81	98.0
May 13, 2018, 3:50 p.m.	81	98.0
May 13, 2018, 3:50 p.m.	81	100.0
May 13, 2018, 3:51 p.m.	81	98.0
May 13, 2018, 3:51 p.m.	81	101.0

Figure 30: Data from User2



192.168.43.223:8000/android_api/login/

Sensor data

Time	Pulse rate	Temperature
May 13, 2018, 3:47 p.m.	99	97.0
May 13, 2018, 3:47 p.m.	112	99.0
May 13, 2018, 3:48 p.m.	102	96.0
May 13, 2018, 3:48 p.m.	100	91.0
May 13, 2018, 3:48 p.m.	107	99.0
May 13, 2018, 3:48 p.m.	110	100.0
May 13, 2018, 3:49 p.m.	98	92.0
May 13, 2018, 3:49 p.m.	98	103.0
May 13, 2018, 3:49 p.m.	109	94.0
May 13, 2018, 3:49 p.m.	90	87.0
May 13, 2018, 3:49 p.m.	104	96.0
May 13, 2018, 3:50 p.m.	92	97.0
May 13, 2018, 3:50 p.m.	107	91.0
May 13, 2018, 3:50 p.m.	104	87.0
May 13, 2018, 3:50 p.m.	109	88.0
May 13, 2018, 3:50 p.m.	102	96.0
May 13, 2018, 3:51 p.m.	92	81.0
May 13, 2018, 3:51 p.m.	98	78.0
May 13, 2018, 3:51 p.m.	86	86.0

Figure 31: Data from User3

6 Cost Analysis

Component	Cost
AtMega328P	150
Bluetooth Module HC-05	300
Heart Rate Sensor	250
LM35 Temperature Sensor	30
Wrist Band	190
Battery	15
Miscellaneous	100
Total Cost	**1035**

Table 1: Cost Estimate

We have tried to minimize the cost of the project to a great extent. To manufacture this product with a more compact body we might have to spend double the cost. This still happens t be much lesser than the current market rice of the product. The web server and android application can be commercialized too. Health care centers that really look forward to implementing a decentralized architecture can highly benefit out of our cost effective product.

7 Conclusion and Future Work

1. We have been able to set up a wearable sensor network circuit which remotely sends data to the smartphone. The data is pre-processed. At the Android Application, the values are updated and displayed in an interval of every 10 seconds. The Application is so designed, that at the first instance of usage, the user can either login as a patient or a doctor. The patient is asked for his basic details along with his medical conditions. Whereas, the doctor is asked for his specialisation along with his basic details. Based on the values entered in the form a profile is created. The next time the user logs in, he'll be kept signed in. The data which has been monitored is plotted and the graph is shown on the next screen. This data is further sent on to the web server. This is how the doctor can monitor the patient's health remotely. We have created a web server where all the patients data will be logged in. The doctor can login and check for a particular patient's physiological parameters.

2. We can extend the scope of this project by adding more details which include Data Analytics. Based on the previous health record of the patient we can interpolate the future values of the particular health parameter of the patient. This can be achieved by using machine learning algorithms. Based on the predicted values we can alert the patients and trusted contacts about their well being.

3. We can further make the hardware circuitry more flexible and compact. We can also add new sensors to it. This makes the product very versatile as it you can plug almost any health parameter measuring device.

4. The current model constantly sends data to the server which might pose a constraint on the battery power of the system. to resolve this issue we can create a system which automatically switches onto a standby mode and hence saves the device from draining too much battery under normal health conditions.

5. The next thing that can be improvised further is the security aspect if it. Since data today is such an expensive commodity we should ensure that the data is not being tampered with. The database should be encrypted and there should be no evasdropping while the data is transmitted over from the phone to the server.

References

[1] Hassnaa Moustafa , Eve M. Schooler , Gang Shen and Sanjana Kamath, *“Remote monitoring and medical devices control in eHealth”,* In Proc. of Wireless and Mobile Computing, Networking and Communications (WiMob), 2016 IEEE 12th International,pp. 1-8.

[2] Uttara Gogate and Jagdish W. Bakal, *“Smart Healthcare Monitoring System Based on Wireless Sensor Networks”,* In Proc. of Computing, Analytics and Security Trends (CAST), International Conference,pp. 594-599.

[3] Alba Amato and Antonio Coronato, *“An IoT-Aware Architecture for Smart Healthcare Coaching Systems”,* In Proc. of Advanced Information Networking and Applications (AINA), 2017 IEEE 31st International Conference,pp. 1027-1034.

[4] Pescosolido, Loreto, Riccardo Berta, Lorenzo Scalise, Gian Marco Revel, Alessandro De Gloria, and Gianni Orlandi,*“An IoT-inspired cloud-based web service architecture for e-health applications”,*In Proc. of Smart Cities Conference (ISC2), 2016 IEEE International, pp. 1-4. IEEE, 2016.

[5] Lam, Stephen Chiu Kwok, Kai Lap Wong, Kwok On Wong, Wenxiu Wong, and Wai Ho Mow,*“A smartphone-centric platform for personal health monitoring using wireless wearable biosensors”,*In Proc. of Information, Communications and Signal Processing, 2009. ICICS 2009. 7th International Conference on, pp. 1-7. IEEE, 2009

[6] quare.github.io/okhttp/

[7] ww.components101.com/wireless/hc-05-bluetooth-module

[8] ww.microchip.com/wwwproducts/en/ATmega328P

[9] eveloper.android.com/studio/

[10] ww.autodesk.com/products/eagle/features

[11] Amir-Mohammad Rahmani, Nanda Kumar Thanigaivelan ,Tuan Nguyen Gia et al., *“Smart e-Health Gateway: Bringing intelligence to Internet-of-Things based ubiquitous healthcare systems”,* In Proc. of Consumer Communications and Networki ng Conference (CCNC), 2015 12th Annual IEEE,pp. 826-834.