A REPORT

ON

Development of Wearable Safety Device

BY

Akshay Uppal

Abhilash Dhar

Siddharth Gupta

Prepared in partial fulfillment of the

Practice School-I/II Cour se

AT

Telecom Centre of Excellence, New Delhi

A Practice School-I/II station of

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

July 2016

A REPORT

ON

Development of Wearable Safety Device

BY

Akshay Uppal
2014A7PS866H

Abhilash Dhar

2014A8PS514G

Siddharth Gupta

2014A3PS178P

Prepared in partial fulfillment of the

Practice School-I/II Course

AT

Telecom Centre of excellence, New Delhi

A Practice School-I/II station of

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

July 2016

Acknowledgment

We greatly acknowledge Telecom Center of Excellence (TCOE), Center for Development of Telematics and BITS, Pilani for providing us with an excellent environment and facilities to complete this Summer Project.

We would like to show my greatest appreciation to Commander Anurag Vibhuti, Deputy Director, TCOE, Ms. Neeti Bhatia, TCOE and Ms. Shikha Srivastava, CDOT as our advisors. We are very grateful for their tremendous support, guidance and help for the completion of this project.

We would also like to thank our family for supporting us through thick and thin and encouraging us always.

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE

PILANI (RAJASTHAN)

Practice School Division

Station: Telecom Centre of Excellence

Centre .New Delhi

Duration : 6 weeks

Date of Start: 23rd May, 2016

Date of Submission 11th July, 2016

Title of the Project: Android Wearable Technology

ID No./Name : 2014A7PS866H, Akshay Uppal

2014A8PS514G, AbhilashDhar

2014A3PS178P, Siddharth Gupta

Name(s) and designation(s) of the expert(s): Shikha Srivastava, Name(s) of the PS Faculty: M.K. Hamirwasia

Key Words: Android, Wearable, Safety, Alert, Hardware Trigger

Project Areas: Bluetooth Communication, Android Serial Communication Stack

Abstract: The project sought to address the issue of women’s safety through development of a wearable alert trigger and its companion Android application, which work in tandem to send intimation to designated acquaintances about the wearer being in imminent danger.

Signature(s) of Student(s)   Signature of PS Faculty

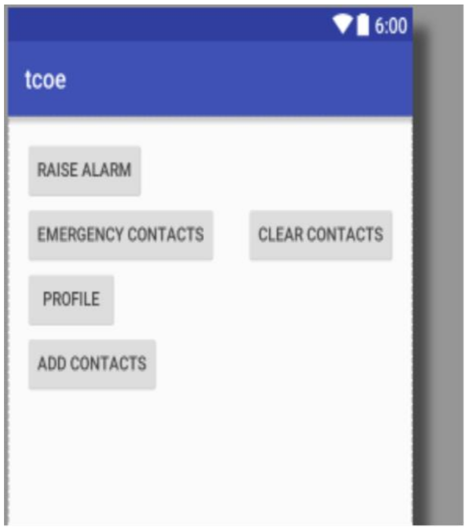
Date   Date

Contents

Synopsis

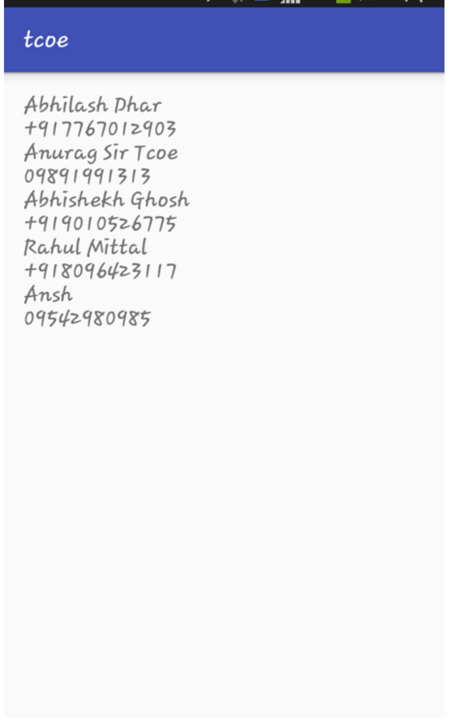
The safety wearable device and its Android app has very simple working. The wearable device Is a simple hc-05 Bluetooth module which has a button on it.When that button is pressed SMS Is sent to all the contacts selected under the emergency contacts list. The SMS sent in the name Of the person that is listed in the profile menu.

Here is what the initial version of the app looks like



When we click on the Profile button the Profile Activity pops up.By clicking on the add contacts Button we can add as many contacts as we want. All these are stored in a Database which I Managed using SQLite language.

Here is what the Emergency Contacts list looks like



Abhilash Dhar
+917767012903
Anurag Sir Tcoe
09891991313
Abhishekh Ghosh
+919010526775
Rahul Mittal
+918096423117
Ansh
09542980985

It contains the contacts names as well as their mobile numbers.
When the Raise Alarm button is pressed a SMS is sent to everyone in the Emergency Contacts List .Its an SOS message and is sent in the name of the person listed in the profile menu.

Because when you are in danger you may not be able to take out your mobile and press the Button on the App in mobile, that's we added a hardware button which can worn in hands as Bracelets or on neck as jewelllery.
When that button is pressed the functions of Raise Alarm button is executed.
We connected the Blutooth device to the Android app using Blutooth classes from android studio And writing code for that respectively.

Introduction

TCOE India **Telecom Center of Excellence India (Delhi)**

Telecom Centres of Excellence, set up in Public Private Partnership (PPP) mode, are an example of the Government, the Academia and the Industry working together for the sustained growth and progress of the country in the Telecom sector. The idea of Telecom Centres of Excellence was initiated with the shared realization, by the Government and the Telecom Industry, that boosting the growth of telecommunications was essential for the overall progress of the country. It was conceptualized in May 2007 and brought into existence by February 2008 with the signing of 7 MoUs between DoT, participating premier Academic Institutes and the sponsors from the Telecom Industry. The eighth TCOE with participation of Railtel came up in Jun 5th 2013.The TCOEs set up in Public Private Partnership (PPP) mode, are an excellent example of the Government, the Academia and the Industry working together for the sustained growth and progress of the country.

The eight largest Telco's have joined the initiative as Principal Sponsors. The pairing and focus areas of each centre were as follows:-

Aircel is with TCOE at IISc Bangalore - Information security and Disaster Management of Telecom Infrastructure

Bharat Sanchar Nigam Ltd. (BSNL) is with TCOE at IIT, Kanpur - Multimedia and Telecom, Cognitive Radio and Computational Mathematics

BhartiAirtel is with TCOE at IIT Delhi - Telecom Technology and Management

Idea Cellular is with TCOE at IIM Ahmedabad - Telecom Policy, Regulation, Customer care

Reliance Communications is with TCOE at IIT Madras - Telecom Infrastructure (Active and Passive) and Energy

Tata Teleservices is with TCOE at IIT Bombay - Rural Telecom Technology

Vodafone Essar is with TCOE at IIT Kharagpur - Next Generation Networks and Technology

RailTel is with TCOE at IIT Roorkee ICT and Broadband Applications

TCOEs are created for promoting development of new technologies, to generate IPRs, incubate innovations and promote entrepreneurship to position India as a global leader in telecom innovation and making India a hub of telecom equipment manufacturing.

TCOE were set up with the sole idea of promoting Research and Development and for promotion of entrepreneurship and innovation aspects in the ICT and Telecom sector. TCOEs have opened up their research platform to all the stakeholders of ICT sector for enabling industry driven research through a competitive process. For this purpose, a TCOE may have collaboration with other academic institutions and funding for the projects would be accessed from Government, industry and VCs etc. on competitive basis.

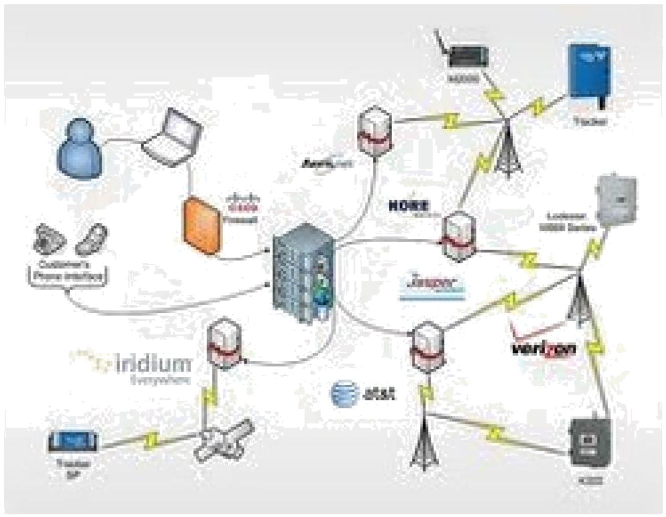
We, thus, invite budding entrepreneurs, researchers, industry and academicians to be a part of TCOE India initiative for Research & Development of application oriented marketable technologies by sharing their research interests and projects. The ideas/projects would be evaluated in confidence and data security would be adhered to by circulation among the TCOEs and the Steering Committee on strictly need to know basis.

2. Machine to machine Communication and Internet of things

Machine to machine (M2M) refers to direct communication between devices using any communications channel, including wired and wireless. In modern times the communication is often via the Internet of Things (IoT). Widespread adoption of Internet Protocol Version 6 (IPv6), with its extremely large address space, is necessary to accommodate all of the sensors and machine-readable identifiers that Internet of Things will require.

Machine to machine communication can include industrial instrumentation, enabling a sensor or meter to communicate the data it records (such as temperature, inventory level, etc.) to application software that can use it (for example, adjusting an industrial process based on temperature or placing orders to replenish inventory). Such communication was originally accomplished by having a remote network of machines relay information back to a central hub for analysis, which would then be rerouted into a system like a personal computer.

More recent machine to machine communication has changed into a system of networks that transmits data to personal appliances. The expansion of IP networks around the world has made machine to machine communication quicker and easier while using less power. These networks also allow new business opportunities for consumers and suppliers.



The internet of things (IoT) revolves around increased machine-to-machine communication; it’s built on cloud computing and networks of data-gathering sensors; it’s mobile, virtual, and instantaneous connection; and they say it’s going to make everything in our lives from streetlights to seaports “smart.”
The internet of things is the network of physical devices, vehicles, buildings and other items—embedded with electronics, software, sensors, and network connectivity that enables these objects to collect and exchange data. In 2013 the Global Standards Initiative on Internet of Things (IoT-GSI) defined the IoT as "the infrastructure of the information society."
The IoT allows objects to be sensed and controlled remotely across existing network infrastructure, creating opportunities for more direct integration of the physical world into computer-based systems, and resulting in improved efficiency, accuracy and economic benefit when IoT is augmented with sensors and actuators, the technology becomes an instance of the more general class of cyber-physical systems, which also encompasses technologies such as smart grids , smart homes, intelligent transportation and smart cities. Each thing is uniquely identifiable through its embedded computing system but is able to interoperate within the existing Internet infrastructure. Experts estimate that the IoT will consist of almost 50 billion objects by 2020.
As per a recent survey and study done by Pew Research Internet Project, a large majority of the technology experts and engaged Internet users who responded—83 percent—agreed with the notion that the Internet/Cloud of Things, embedded and wearable computing (and the corresponding dynamic systems) will have widespread and beneficial effects by 2025. As such, it is clear that the IoT will consist of a very large number of devices being connected to the Internet.
The ability to network embedded devices with limited CPU, memory and poour resour ces means that IoT finds applications in nearly every field. Such systems could be in charge of collecting information in settings ranging from natural ecosystems to buildings and factories,[thereby finding applications in fields of environmental sensing and urban planning.
On the other hand, IoT systems could also be responsible for performing actions, not

just sensing things. Intelligent shopping systems, for example, could monitor specific users' purchasing habits in a store by tracking their specific mobile phones. These users could then be provided with special offers on their favorite products, or even location of items that they need, which their fridge has automatically conveyed to the phone. Additional examples of sensing and actuating are reflected in applications that deal with heat, electricity and energy management, as well as cruise-assisting transportation systems. Other applications that the internet of things can provide is enabling extended home security features and home automation. The concept of an "internet of living things" has been proposed to describe networks of biological sensors that could use cloud-based analyses to allow users to study DNA or other molecules. All these advances add to the numerous lists of IoT applications. Now with IoT, we can control the electrical devices installed in our house while we are sorting out our files in office. Our water will be warm as soon as we get up in the morning for the shower. All credit goes to smart devices which make up the smart home. Everything connected with the help of Internet

However, the application of the IoT is not only restricted to these areas. Other specialized use cases of the IoT may also exist. An overview of some of the most prominent application areas is provided here. Based on the application domain, IoT products can be classified broadly into five different categories: smart wearable, smart home, smart city, smart environment, and smart enterprise. The IoT products and solutions in each of these markets have different characteristic.

3. Smart Wearable

Wearable technology is the one of the fastest growing field within in the Internet of Things.

Wearable technology, wearables, fashionable technology, wearable devices, tech togs, or fashion electronics are clothing and accessories incorporating computer and advanced electronic technologies. The designs often incorporate practical functions and features.

Wearable technology is on the rise in both personal and business use. In the consumer space, sales of smart wristbands (aka activity trackers such as the Jawbone UP and Fit Bit Flex) started accelerating in 2013. One out of five American adults has a wearable device according to the 2014 PriceWaterhouse Coopers Wearable Future Report. Smart watches are a second high-profile sector and while wearable devices have been around for years, it has only started gaining mass market attention with the introduction of new models by Samsung and later by Apple. The now defunct Google Glass gained a lot of media attention, but the project ground to a halt in early 2015, with Google stopping device sales. In healthcare, wearables have long been used, for example in hearing aids and in detecting health disorders such as sleep apnea.

Wearable devices such as activity trackers are a good example of the Internet of Things, since they are part of the network of physical objects or "things" embedded with electronics, software, sensors and connectivity to enable objects to exchange data with a manufacturer, operator and/or other connected devices, without requiring human intervention.

Moreover the wearable technology of the activity tracker can fused with even more advanced sensor technology and can be used a wearable safety devices that can allow help people in the hour of distress and can roam around safely.

3.1 Safety Wearable Device

Wearable technology is not just for sports and fitness. Our advanced fused sensor technology disappears into traditional safety clothing and allows workers and employers to run the safest job sites possible and develop deep insights about safety, efficiency and future projects.

Due to increase in gender-based violence in our country women don’t feel safe while travelling or conducting thing their day to day work. To minimize crime and to make women and children feel protected, a low-energy, wearable Bluetooth safety device is designed that pairs with a safety app. It can be worn on a clip, keychain or snap hook and activated by pressing the emergency button. Like other safety gadgets this one too notifies our contacts and lets them know our location.

Wearable technology is not just for sports and fitness. Our advanced fused sensor technology disappears into traditional safety clothing and allows workers and employers to run the safest job sites possible and develop deep insights about safety, efficiency and future projects.

A low -energy, wearable Bluetooth safety device that pairs with a safety app. It can be worn on a clip, keychain or snap hook and activated by pressing the emergency button. Like other safety gadgets this one too notifies our contacts and lets them know where we are.

The Wearable Device developed, has 2 modes of functioning 1. The Hardware Device Mode :

In the Hardware Device Mode the user can click on the button on the Device, on doing so the caregivers get a SMS Alert with a link to track the user in Real-Time. 2. The Smartphone Mode :

In the Smartphone Mode the user can open the Android App and Click on the Alert Button. On doing so an Alert SMS will be sent to the caregivers with a link to track the user in Real-Time.

Hardware

Firstly, for the device to communicate with the mobile App, the device has to have a setup for wireless communication. For that, various wireless communication techniques were available. Basic overview of those techniques is as follows.

Radio-frequency identification (RFID): RFID uses EMF to automatically identify and track tags attached to objects. The tags contain electronically stored information and need not be within the line of sight of the reader. Tags can either be active or passive

- Active RFID tags : It is required to have its own power supply, extending range up to a 100 meters hence unnecessary for the purpose of the project. It also requires Constant Monitoring on part of detector tags and has no physical trigger available which is further more required for the project.
- Passive RFID tags: the range of the passive RFID can be varied dependent on power, provided redesigning the entire circuit according to the needs is feasible. Overall it requires very less poor which is advantageous for the project but also requires a specific module to be attached on Android, Windows phones – NFC readers operate on separate frequencies, which is not quite feasible.

Near Field Communications (NFC): Evolved from radio frequency identification (RFID) tech, an NFC (Near field communication) chip operates as one part of a wireless link. Once it's activated by another chip, small amounts of data between the two devices can be transferred when held a few centimeters from each other. The Range of 8-10 cm. and no hardware trigger. Also most of the companies like Apple, Micromax, Intex etc. don’t provide NFC

Bluetooth: defines real-time and non-real-time traffic as synchronous connection-oriented and asynchronous connectionless links, respectively. It is a TDD system and uses an FHSS technique with a maximum of 1,600 hops per second over 79 RF channels 1 MHz wide. Bluetooth has one-, three-and five-slot packets that may or may not be encoded with forward error correction, enabling a wide range of data rates for different kinds of applications.

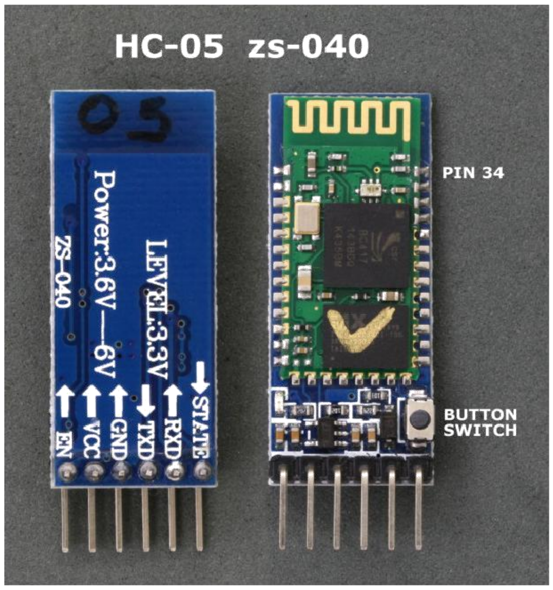
Hardware Setup

Components Required:

1. **Arduino Uno Module**
2. **Bluetooth HC-05 Module**
3. **Breadboard**
4. **Resistors 1.6Mohm**
5. **Connecting wires**
6. **Micro USB Cable**

Arduino With HC-05 Bluetooth Module

The HC-05 is based on the EGBT-045MS Bluetooth module. It can operate as either a slave device or a master device. As a slave it can only accept connections. As a master it can initiate a connection.



Connections

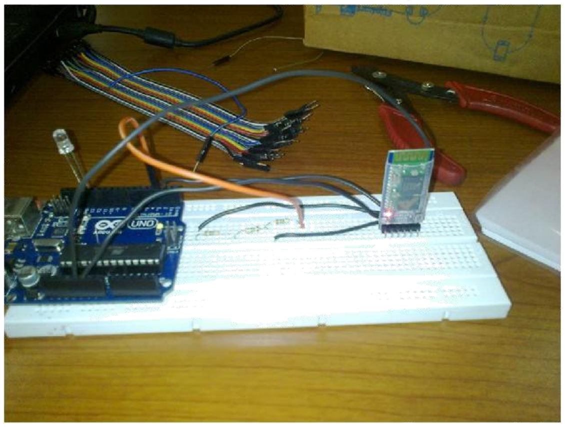
HC-05 Vcc to 5V (can be from the +5V out from the Arduino) HC-05 GND to common GND

HC-05 RX to Arduino pin D3 (TX) via a voltage divider

HC-05 TX to Arduino pin D2 (RX) connect directly

The HC-05 break out board has a 3.3v regulator that allows an input voltage of 3.6v to 6v but the TX and RX pins are still 3.3v. This means we can use the 5V out from the Arduino to poour the boards but we cannot connect the Arduino directly to the HC-05 RX pin.

For the HC-05 RX pin (data in) we need to convert the Arduino's 5V to 3.3v. A simple way to do this is by using a voltage divider made from a couple of resistors. In this case 1.6M ohm resistor was used.



Pair with an Android device

Before we can make a connection between blue tooth devices they need to be paired. So, with the Arduino and HC-05 powered, on the Android device;

– turn on bluetooth,

– scan for devices and the HC-05 should be listed

– pair with the HC05 and enter the password "1234" assuming the module have the default password.

Once paired the blinking LED on the HC-05 will change to a single short blink every 2 seconds or so.

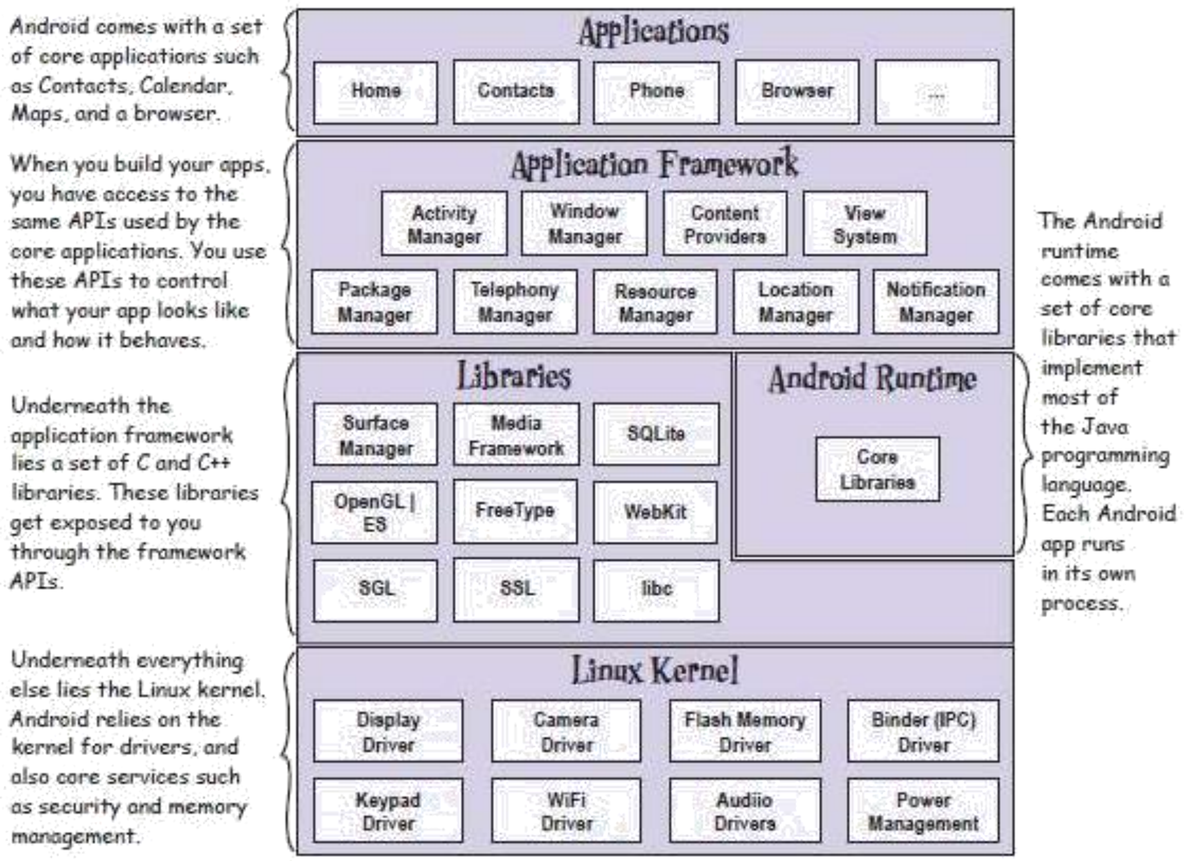
Interfacing with Mobile Application:

ANDROID:

Operating Systems have developed a lot in last 15 years. Starting from black and white phones to recent smart phones or mini computers, mobile OS has come far away. Especially for smart phones, Mobile OS has greatly evolved from Palm OS in 1996 to Windows pocket PC in 2000 then to Blackberry OS and Android.

One of the most widely used mobile OS these days is ANDROID. Android is a software bunch comprising not only operating system but also middleware and key applications. Android Inc was founded in Palo Alto of California, U.S. by Andy Rubin, Rich miner, Nick sears and Chris White in 2003. Later Android Inc. was acquired by Google in 2005. After original release there have been number of updates in the original version of Android.

Android is a powerful Operating System supporting a large number of applications in Smart Phones. These applications make life more comfortable and advanced for the users. Hardwares that support Android are mainly based on ARM architecture platform.



Android comes with a set of core applications such as Contacts, Calendar, Maps, and a browser.

When you build your apps, you have access to the same APIs used by the core applications. You use these APIs to control what your app looks like and how it behaves.

Underneath the application framework lies a set of C and C++ libraries. These libraries get exposed to you through the framework APIs.

Underneath everything else lies the Linux kernel. Android relies on the kernel for drivers, and also core services such as security and memory management.

The Android runtime comes with a set of core libraries that implement most of the Java programming language. Each Android app runs in its own process.

How Java and Android wor   together

a program in

Java for Android, **Dalvik EXecutable (DEX** **Compiling**

. In the projec

Development platform

executes **Dalvik**

Virtual Machine (DVM

version o

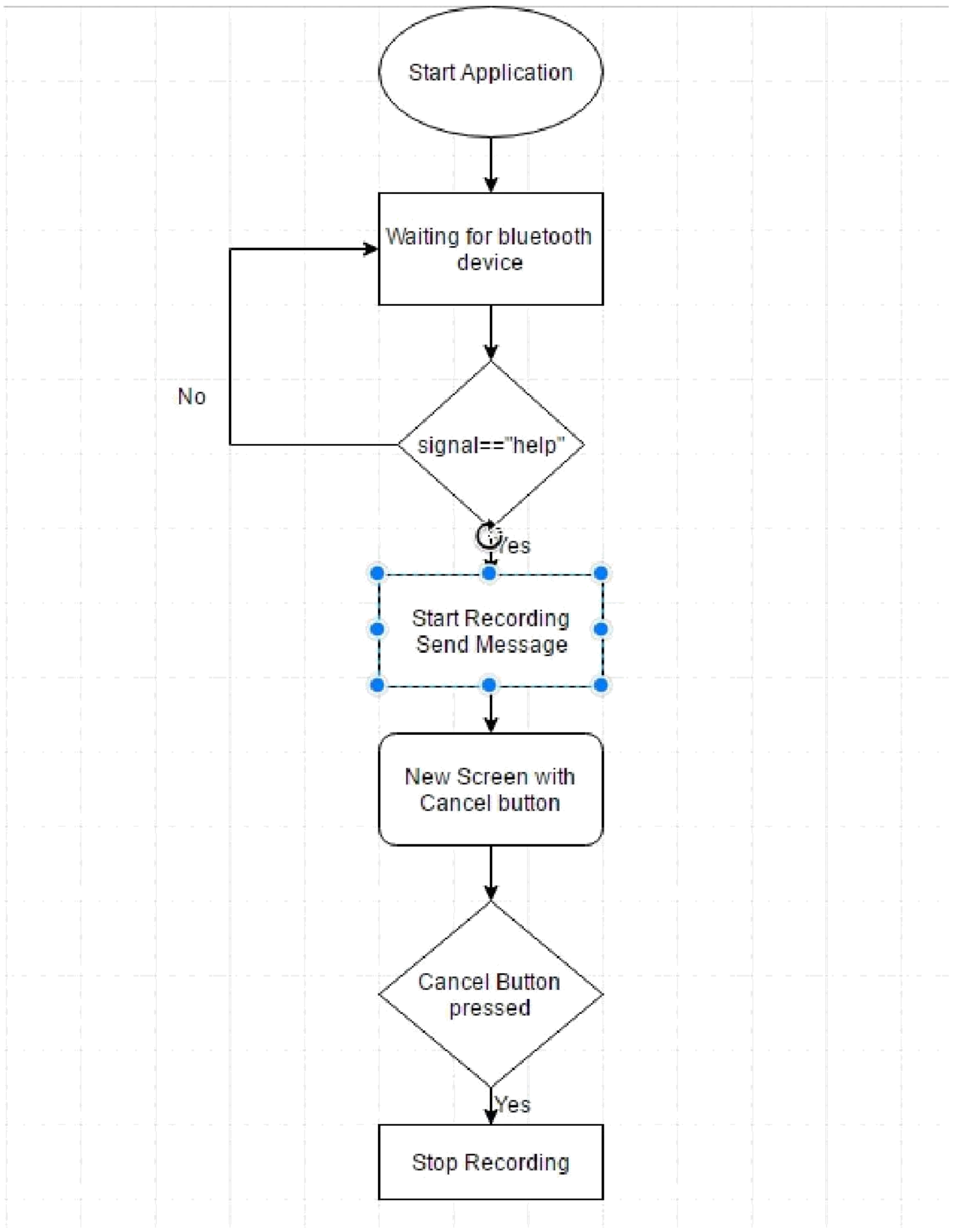
to the end user Android

friendly and easy to use. The way the DVM allows us access is indeed easy to use because of the Android Application Programming Interface (API). The Android API acts as an interface that allows us to use the features already shipped with Android.

Basic Components of android applications

- **Android applications are composed of one or more application components (activities, services, content providers, and broadcast receivers)**
- **Each component performs a different role in the overall application behavior, and each one can be activated individually (even by other applications)**
- **The manifest file must declare all components in the application and should also declare all application requirements, such as the minimum version of Android required and any hardware configurations required**
- **Non-code application resources (images, strings, layout files, etc.) should include alternatives for different device configurations (such as different strings for different languages)**

The Application follows the following pattern:


Start Application
Waiting for bluetooth device
No
signal=="help"
Yes
Start Recording Send Message
New Screen with Cancel button
Cancel Button pressed
Yes
Stop Recording

Application Layout

The Application has a number of buttons on the start page which list out the main functions performed.

1. **Raise Alarm**
2. **Emergency Contacts**
3. **Clear Contacts**
4. **Profile**
5. **Add Contacts**

1. RAISE ALARM **: This is the main button of the application which on being triggered sets in motion all the other defined functions. When the button is pressed on the application, the onClick() method of the onClickListener() is called. This method in turn calls three methods, sendMsg(), startRecording(), emergencyPage() . Each of these methods do the work as suggested by their names.**
2. Emergency Contacts **: The button shows the contacts which the user has listed in as his/her emergency contacts.**
3. Clear Contacts **: The button clears the contacts saved by the user. This is useful is a person wishes to edit his/her list of emergency contacts.**
4. Profile**: This shows information about the user himself. The name fed in the profile page is used in the message that is sent out to the emergency contacts in times of distress.**
5. Add contacts**: This takes the user to the contacts activity in his/her phone where they can select the contacts to be listed as their emergency ones. After selecting the activity returns to the main screen of the app and the selected contacts are added in the emergency contacts list.**

For a full documentation of the code see Appendix 2.

Conclusions and Recommendations

1. The breadboard model of a Bluetooth wearable device was successfully developed which sent an alert to a connected phone, which in turn sent an SMS to chosen contacts and alerted them of the wearer's situation.

2. While the module selected (HC-05) was the preferred choice for the development of the project considering its cost effectiveness, for deployment purpose a Bluetooth Low Energy chip (which has the same interface and connections, but uses less power) might be more suitable. This module costs nearly 3 times as much as an HC-05 but these costs can be recovered easily when removing the requirement of a development board.

3. Since the versatility of the HC-05 extends well beyond simple data communication, the module can be used to route audio through and to an attached microphone and speaker, extending the functionality to voice commands, and talkback. Note that for maintenance of audio quality, this version cannot boast a upgradeable BLE configuration (which doesn't have the required data rate)

4. Support for multiple Bluetooth devices that can be connected to the application.

5. The application can also be made to send the location of the user to their emergency contacts via a link in the SMS. This way, they can come to know the location of the person with Google Maps. This can be done using the Google Play Services API.

6. Setting up of a login page for the application where the information of the user can be collected in the beginning only. Currently the user has to specifically enter his/her name in the application once.

APPENDIX

Appendix 1.A

```
// Sends "Bluetooth Test" to the serial monitor and the software serial once
every second.
// Connect the HC-05 module and data over Bluetooth
// The HC-05 defaults to commincation mode when first
powered on. // The default baud rate for communication is 9600
#include <SoftwareSerial.h>
SoftwareSerialBTserial(2, 3); // RX | TX
// Connect the HC-05 TX to Arduino pin 2 RX.
// Connect the HC-05 RX to Arduino pin 3 TX through a voltage
divider. char c = ' ';
void setup()
{
Serial.begin(9600); Serial.println("Enter
AT commands:");
   // HC-06 default serial speed for communication mode is 9600
BTserial.begin(9600);
}
void loop()
{
BTserial.println("Bluetooth Test");
Serial.println("Bluetooth
Test"); delay(1000);
}
```

Appendix 2.A

MainActivity.java

This is the main activity in the our application and opens up when our app is started.

```
package com.example.siddharth.tcoe;

import android.app.Activity; import
android.os.Handler; import
android.content.Context; import
android.content.Intent; import
android.database.Cursor; import
android.media.MediaPlayer;
import android.media.MediaRecorder;
import android.net.Uri;
import android.os.Environment; import
android.provider.Contacts;
import android.provider.ContactsContract; import
android.support.v7.app.AppCompatActivity; import
android.os.Bundle;
import android.telephony.SmsManager;
import android.util.Log;
import android.view.View; import
android.view.ViewGroup; import
android.widget.Button;
import android.widget.LinearLayout; import
android.widget.TextView; import
android.widget.Toast;
import  android.bluetooth.BluetoothAdapter;   import
android.bluetooth.BluetoothDevice;                import
android.bluetooth.BluetoothSocket;
import java.io.IOException; import
java.io.InputStream; import
java.io.OutputStream; import
java.lang.reflect.Method; import
java.util.UUID;
import java.io.IOException;
import java.util.ArrayList;
import android.os.Build;

public class MainActivity extends AppCompatActivity { private
        Button emergency;
        public static MediaRecorder myRecorder;
        private String outputFile = null; private Button
        addContacts;
        private Button emergencyContacts ; final
        int PICK_CONTACT_REQUEST = 1;
        MyDBHandler dbHandler; ProfileDBHandler
        pr;
        data toStore, stored;
        ArrayList<data> contacts = new ArrayList<data>(); private

        static final String TAG = "bluetooth2";
        Button btnOn, btnOff;
        TextView txtArduino;
        Handler h;

        final int RECIEVE_MESSAGE = 1;                    // Status  for Handler
        private BluetoothAdapter btAdapter = null ;
        private BluetoothSocket btSocket = null;
        private StringBuilder sb = new StringBuilder();

        private ConnectedThread mConnectedThread;

        // SPP UUID service
        private static final UUID MY_UUID = UUID.fromString("00001101-0000-1000-8000-
00805F9B34FB");

        // MAC-address of Bluetooth module (you must edit this line)
        private static String address = "98:D3:31:30:AD:78";


        @Override

        protected void onCreate(Bundle savedInstanceState) {   //This method is
automatically called when the android screen is opened//
            super .onCreate(savedInstanceState);
            setContentView(R.layout.activity_main);

            emergency = (Button) findViewById(R.id.emergency_button);

            addContacts = (Button) findViewById(R.id.add_button);

            outputFile = Environment. getExternalStorageDirectory().

                    getAbsolutePath() + "/javacodegeeksRecording.3gpp";

            myRecorder = new MediaRecorder(); myRecorder
            .setAudioSource(MediaRecorder.AudioSource.MIC);
            myRecorder.setOutputFormat(MediaRecorder.OutputFormat.THREE_GPP);
            myRecorder.setAudioEncoder(MediaRecorder.OutputFormat.AMR_NB);
            myRecorder.setOutputFile(outputFile);
            emergency.setOnClickListener(new View.OnClickListener() { @Override

                public void onClick(View v) {
                    sendMsg(v);
                    //startRecording(v);
                    //emergencyPage(v);

              }
            });
            dbHandler = new MyDBHandler(this , null , null , 2); pr = new
            ProfileDBHandler(this, null, null, 3 );
            addContacts.setOnClickListener( new View.OnClickListener() { @Override


                public void onClick(View v) { }); }


            addContacts(v);

            h = new Handler() { //This is the handler object which is used for getting the Trigger message
from the Android Blutooth Device//
                public void handleMessage(android.os.Message msg) { switch
                    (msg.what) {
                        case RECIEVE_MESSAGE:
// if receive massage
```

```
                                    //btnOn.setEnabled(true);
                                    if (sbprint.contains( "HELP" )) {
                                        for (int i = 0; i < 40; i++)
                                            mConnectedThread.write("OK");

                                        pr.sendSMS();
                                    }
                                }
    + "...");                       //Log.d(TAG, "...String:"+ sb.toString() +  "Byte:" + msg.arg1
                                break;
                                byte[] readBuf = (byte[]) msg.obj;
                                String strIncom = new String(readBuf, 0, msg.arg1);
// create string from bytes array
// append string                sb.append(strIncom);
                                int endOfLineIndex = sb.indexOf("N");
// determine the end-of-line
// if end-of-line,              if (endOfLineIndex > 0) {
// extract string                   String sbprint = sb.substring(0, endOfLineIndex);
// and clear                        sb.delete(0, sb.length());
                                    txtArduino.setText("Data from Arduino: " + sbprint);
//String to work with!!                 // update TextView
                                    //btnOff.setEnabled(true);

                    }
                }


            };      ;
            btAdapter = BluetoothAdapter.getDefaultAdapter();               // get Bluetooth
adapter
            checkBTState();

        }

        private class ConnectedThread extends Thread { //this extends Thread class which
is actually used for running two processes
            private final InputStream mmInStream;                //in parallel if we create a
different Thread for one process//

            private final OutputStream mmOutStream;

            public ConnectedThread(BluetoothSocket socket) {
                InputStream tmpIn = null;

                OutputStream tmpOut = null;

                // Get the input and output streams, using temp objects because
                // member streams are final
                try {
                    tmpIn =    socket.getInputStream();
                    tmpOut  = socket.getOutputStream();
                } catch (IOException e) {

                }
                mmInStream = tmpIn;

                mmOutStream = tmpOut;
        }
            public void run() {
                byte[] buffer = new byte[256 ];  // buffer store for the stream
                int bytes; // bytes returned from read()

                // Keep listening to the InputStream until an exception occurs
```

```
                    while ( true) {
                            try    {
                                    // Read from the InputStream

                                    bytes = mmInStream.read(buffer);                          // Get number of bytes and
message in "buffer"          h.obtainMessage( RECIEVE_MESSAGE, bytes, -1,
buffer).sendToTarget();                   // Send to message queue Handler
                            } catch (IOException e) {
                                    break;

                            }
                    }
            }
            /* Call this from the main activity to send data to the remote device */
            public void write(String message) {
                    Log.d( TAG, "...Data to send: " + message + "...");
                    byte[] msgBuffer = message.getBytes();
                    try {
                            mmOutStream.write(msgBuffer);
                    } catch (IOException e) {

                            Log.d(TAG, "...Error data send: " + e.getMessage() + "...");
                    }
            }
    }


    public void devices(View v) {
            Intent intent = new Intent(this, emergencyActivity.class);

            startActivity(intent);
    }

    //btnOn = (Button) findViewById(R.id.btnOn); //btnOff =                          //    button LED ON
    (Button) findViewById(R.id.btnOff);                                              //    button LED OFF
    //txtArduino = (TextView) findViewById(R.id.txtArduino); received data           //    for display the

from the Arduino


    private BluetoothSocket createBluetoothSocket(BluetoothDevice device) throws
IOException {
            if (Build.VERSION.SDK_INT >= 10) {
                    try {
                            final Method m =
device.getClass().getMethod("createInsecureRfcommSocketToServiceRecord", new
Class[]{UUID.class});
                            return (BluetoothSocket) m.invoke(device, MY_UUID);
                    } catch (Exception e) {

                            Log.e(TAG, "Could not create Insecure RFComm Connection", e);
                    }
            }

            return device.createRfcommSocketToServiceRecord(MY_UUID);
    }
    @Override
    public void onResume() {

            super.onResume();

            Log.d(TAG, "...onResume - try connect...");

            //    Set up a pointer to the remote node using it's address.
            BluetoothDevice device = btAdapter.getRemoteDevice(address );

            //    Two things are needed to make a connection:
```

```
            //      A MAC address, which we got above.
            //      A Service ID or UUID.  In this case we are using the
            //          UUID for SPP.

            try {
                    btSocket = createBluetoothSocket(device); } catch
            (IOException e) {
                    errorExit("Fatal Error", "In onResume() and socket create failed: " + e.getMessage() +
".");
            }

            // Discovery is resource intensive. Make sure it isn't going on // when you

            attempt to connect and pass your message. btAdapter.cancelDiscovery();

            // Establish the connection.  This will block until it connects.
            Log.d(TAG, "...Connecting..."); try {

                    btSocket.connect();
                    Log. d(TAG, "....Connection ok..."); } catch
            (IOException e) {
                    try { btSocket.close();

                    } catch (IOException e2) {
                            errorExit( "Fatal Error", "In onResume() and unable to close socket during
connection failure" + e2.getMessage() + ".");
                    }
            }

            // Create a data stream so we can talk to server.
            Log.d(TAG , "...Create Socket...");


            mConnectedThread = new ConnectedThread(btSocket); }

        mConnectedThread .start();

        @Override
        public void onPause() {

            super.onPause();

            Log.d(TAG, "...In onPause()...");

            try { btSocket.close();

            } catch (IOException e2) {
                    errorExit("Fatal Error", "In onPause() and failed to close socket." + e2.getMessage() +
".");
            }
        }

        private void checkBTState() {
            // Check for Bluetooth support and then check to make sure it is turned on // Emulator
            doesn't support Bluetooth and will return null
            if (btAdapter == null) {
                    errorExit( "Fatal Error", "Bluetooth not support"); } else {

                    if (btAdapter.isEnabled()) { Log.d(TAG,
                            "...Bluetooth ON...");
                    } else {
                            //Prompt user to turn on Bluetooth
                            Intent enableBtIntent = new
Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
                            startActivityForResult(enableBtIntent, 1);
        }  }  }


        private void errorExit(String title, String message) {
                Toast.makeText(getBaseContext(), title + " - " + message,

    Toast.LENGTH_LONG).show(); }


        finish();

        public void sendMsg(View view) { //This method is used for sending a Message to all the contacts in
    the Emergency Contacts list//

                pr.sendSMS();

        }

        public void startRecording(View view) { //This method starts the recording in the
    app         try {

                        myRecorder.prepare();
                        myRecorder.start();
                } catch (IllegalStateException e) {
                        // start:it is called before prepare()
                        // prepare: it is called after start() or before setOutputFormat() e.printStackTrace();

                } catch (IOException e) {
                        // prepare() fails

                }       e.printStackTrace();
                //text.setText("Recording Point: Recording"); emergency.setEnabled(false);
                Toast.makeText(getApplicationContext(), "Start recording...",


        }                       Toast.LENGTH_SHORT).show();

        public void contacts(View view) //This method opens the Activity page for Emergency
    Contacts,where you can see all your emergency contacts//
        {
                //
    Toast.makeText(this,dbHandler.databaseToString(),Toast.LENGTH_LONG).show();
                Intent i = new Intent(this, EmActivity.class);
                //i.putExtra("mess",dbHandler);

        }       startActivity(i);


        public void deleteAll(View view) //This method is clearing the emergency contacts list {.It removes all
    the contacts//

        }       pr.deleteAllValues_c();

        public void changeProfile(View view) {
                Intent in = new Intent(this, ProfilePage.class);

                startActivity(in);

        }


        public void addContacts(View view) { //This method is for adding the contacts in our Emergency
    contacts list
                Intent intent = new Intent(Intent.ACTION_PICK,
Uri.parse("content://contacts"));
                intent.setType(ContactsContract.CommonDataKinds.Phone.CONTENT_TYPE);
                startActivityForResult(intent, PICK_CONTACT_REQUEST);                                    //
add a constant integer as a request code

        }

        private static final String[] projection = new

String[]{ContactsContract.CommonDataKinds.Phone.NUMBER};

        @Override
        protected void onActivityResult(int requestCode, int resultCode, Intent data) {
                super.onActivityResult(requestCode, resultCode, data);
                switch (requestCode) {
                        case ( PICK_CONTACT_REQUEST):
                                if (resultCode == Activity.RESULT_OK) {
                                        Uri contactData = data.getData();

                                        Cursor c = getContentResolver().query(contactData, null, null,
null, null);
                                        //startManagingCursor(c);
                                        if (c.moveToFirst()) {
                                                int numberIndex =
c.getColumnIndex(ContactsContract.CommonDataKinds.Phone.NUMBER);
                                                String num = c.getString(numberIndex);
                                                int nameIndex =
c.getColumnIndex(ContactsContract.CommonDataKinds.Phone.DISPLAY_NAME);
                                                String name = c.getString(nameIndex);
//                                                  Toast.makeText(this,  name + " has number " + num,
Toast.LENGTH_LONG).show();
                                                Product product = new Product(name, num);
                                                pr.addProduct_c(product);

                                        }
                                }

                                break;
                }
        }

        public class data {
                String number = null;

                String name = null;

                public data(String name, String number) {
                        this.name = name;

                        this.number = number;
                }
        }

  }
```

Appendix 2.B

Main_activity_xml

This is how the main activity looks.

The Application has a number of buttons on the start page which list out the main functions performed.

6. **Raise Alarm**
7. **Emergency Contacts**
8. **Clear Contacts**
9. **Profile**
10. **Add Contacts**

Here is the XML code for the Main_activity_xml.

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:tools="http://schemas.android.com/tools"
  android:layout_width="match_parent"
  android:layout_height="match_parent"
  android:paddingBottom="@dimen/activity_vertical_margin"
  android:paddingLeft="@dimen/activity_horizontal_margin"
  android:paddingRight="@dimen/activity_horizontal_margin"
  android:paddingTop="@dimen/activity_vertical_margin"
  tools:context="com.example.siddharth.tcoe.MainActivity"
  android:id="@+id/mainActivity_layout">
  <Button
    android:id="@+id/emergency_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:onClick="emergency"
    android:text="@string/emergency" />
  <Button
    android:id="@+id/contacts_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

```
        android:layout_alignStart="@+id/emergency_button"

        android:layout_below="@+id/emergency_button"

        android:onClick="contacts"

        android:text="@string/emergency_contacts" />

    <Button
        android:id="@+id/profile_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:onClick="changeProfile"
        android:text="@string/edit_profile"
        android:layout_alignStart="@+id/contacts_button"

        android:layout_below="@+id/contacts_button" />

    <Button android:id="@+id/add_button"

        android:layout_width="wrap_content"

        android:layout_height="wrap_content"

        android:layout_alignStart="@id/profile_button"

        android:layout_below="@+id/profile_button"

        android:onClick="editProfile"

        android:text="@string/add_contacts" />

    <Button
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Clear Contacts"
        android:id="@+id/button"
        android:layout_alignBottom="@+id/contacts_button"

        android:layout_alignParentEnd="true"

        android:onClick="deleteAll"

        android:nestedScrollingEnabled="false" />


</RelativeLayout>
```

Appendix 2.C

Profile.java

This is the class which contains details about the of the user .We use objects of this class to communicate with the database.

Here is the code for that

```
package com.example.siddharth.tcoe;

    /**
  * Created by Akshay Uppal on 7/11/2016.
     */
public class Profile {
        private int _id;
        private String _productname;
        public Profile(){
        }
        public Profile(String productname){
                this._productname = productname;
        }
        public void set_id(int _id) {
                this._id = _id;
        }
        public void set_productname(String _productname) {
                this._productname = _productname;
        }
        public int get_id() {
                return _id;
        }
        public String get_productname() {
                return _productname;
        }
}
```

Appendix 2.D

Product.java

This is the class which stores the contacts name and their mobile numbers. Here is the code for Product.java

```
package com.example.siddharth.tcoe;

/**
 * Created by Akshay Uppal on 7/9/2016. */

public class Product {

    private int _id;
    private String _productname;

    private String _num;

    public Product(){
    }


    public Product(String productname,String num){
        this._productname = productname;

        this._num=num;
    }

    public void set_id(int _id) {

        this._id = _id;
    }

    public void set_productname(String _productname) {

        this._productname = _productname;
    }
    public void set_num(String num)

    {
        this._num=num;
    }

    public int get_id() {

        return _id;
    }
    public String get_productname() { return
        _productname;
    }
    public String get_num()
    {
      return _num;
  }
}
```

Appendix 2.E

ProfilePage.java

This file contains the details about the user and how the user can update his/her name and how it gets saved to the database

```
package com.example.siddharth.tcoe;

import android.support.v7.app.AppCompatActivity; import
android.os.Bundle;
import  android.widget.EditText;  import
android.text.TextWatcher;         import
android.text.Editable;

public class ProfilePage extends AppCompatActivity {

    EditText name;
    ProfileDBHandler p;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_profile_page);
        name = (EditText) findViewById(R.id. name);
        p = new ProfileDBHandler(this, null, null, 3);
        String s = p.databaseToString();
        if(!p.equals(""))
        {
            name.setText(s);
        }

        name.addTextChangedListener(new TextWatcher() {

            public void afterTextChanged(Editable s) {
                String st= s.toString();
                p.deleteAllValues();
                Profile pro = new Profile(st);

                p.addProduct(pro);
            }

            public void beforeTextChanged(CharSequence s, int start, int count, int
after) {}

            public void onTextChanged(CharSequence s, int start, int before, int
count) {}

        });
    }
}
```

Appendix 2.F

ProfilePage XML file

This is contains details about how the Profile Page looks

Here is the XML code for that

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android" xmlns:tools
    ="http://schemas.android.com/tools" android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.example.siddharth.tcoe.ProfilePage">

    <TextView android:layout_width="wrap_content"
        android:layout_height="wrap_content"

        android:textAppearance="?android:attr/textAppearanceSmall"
        android:text="Name : "
        android:id="@+id/textView"
        android:layout_alignParentTop="true"
        android:layout_alignParentStart= "true"
        android:layout_marginTop="50dp" />
    <EditText
        android:layout_width="300dp"
        android:layout_height="wrap_content"
        android:id="@+id/name"
        android:layout_alignBottom="@+id/textView"
        android:layout_toEndOf="@+id/textView"
        />

</RelativeLayout>
```

Appendix 2.G

ProfileDBHandler.java

This is the Database Handler file which controls the handling of the database.It makes sure that the data such as contacts

and user profile info gets saved and are not deleted even if user closes the app.

There is the table which contains the details about the user profile

There is the table which contains the details about the contacts.

All the information the user enters gets saved in these Tables. The name of the database is ProfileDB.db.
We have SQLite language which is used for communicating between Android Studio and the database file.
Using SQLite we save the profile information the user provides as well as the information about the emergency contact numbers and their names.
Here is the Java code for ProfileDBHandler.java

```
package com.example.siddharth.tcoe;


  // Created by Akshay Uppal on 7/11/2016.

  //This is the Database Handler file which controls the handling of the database.It makes sure that the
data such as contacts
    //and user profile info gets saved and are not deleted even if user closes the app.

import android.database.sqlite.SQLiteDatabase; import
android.database.sqlite.SQLiteOpenHelper; import
android.database.Cursor;
import android.content.Context; import
android.content.ContentValues; import
android.telephony.SmsManager;
public class ProfileDBHandler extends SQLiteOpenHelper {

        private static final int DATABASE_VERSION = 3;
        private static final String DATABASE_NAME = "profileDB.db";
        public static final String TABLE_PRODUCTS = "products";// This is the table which contains the
details about the user profile
        public static final String TABLE_PRODUCTS_C = "profiles";//This is the table which contains the
details about the contacts.
        public static final String COLUMN_ID = "_id";//This is a common attribute for both tables which is
basically the primary key
        public static final String COLUMN_PRODUCTNUM = "productnum";//This is the
```

attribute for the contacts table which contains the info about Mobile number of the user.

```
        public static final String COLUMN_PRODUCTNAME = "productname";//This also a common attribute
for both the tables

        //We need to pass database information along to superclass
        public ProfileDBHandler(Context context, String name, SQLiteDatabase.CursorFactory
factory, int version) {

                super(context,      DATABASE_NAME, factory, DATABASE_VERSION);
        }
        @Override
        public void onCreate(SQLiteDatabase db) { //Here we create both the tables
                String query = "CREATE TABLE " + TABLE_PRODUCTS + "(" +
                                COLUMN_ID + " INTEGER PRIMARY KEY AUTOINCREMENT, " +
                                COLUMN_PRODUCTNAME + " TEXT " +
                                ");";

                db.execSQL(query);

                  query = "CREATE TABLE " + TABLE_PRODUCTS_C + "(" + COLUMN_ID +
                                " INTEGER PRIMARY KEY AUTOINCREMENT, " +
                                COLUMN_PRODUCTNAME + " TEXT, " +
                                COLUMN_PRODUCTNUM + " TEXT " +
                                ");";

                db.execSQL(query);

        }

        @Override
        public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) { //in case the tables are
to be updated

                db.execSQL("DROP TABLE IF EXISTS " + TABLE_PRODUCTS); }

        onCreate(db);
        //Add a new row to the database
        public void addProduct(Profile product){  //for updating the name in the Profile//
                ContentValues values = new ContentValues();
                values.put(COLUMN_PRODUCTNAME, product.get_productname());
                SQLiteDatabase db = getWritableDatabase();
                db.insert(TABLE_PRODUCTS, null, values);

        }     db.close();

        public void addProduct_c(Product product){ //for adding a name and a number in Emergency
contacts list//
                ContentValues values = new ContentValues();
                values.put(COLUMN_PRODUCTNAME , product.get_productname());
                values.put(COLUMN_PRODUCTNUM , product.get_num());
                SQLiteDatabase db = getWritableDatabase();
                db.insert(TABLE_PRODUCTS_C, null, values);

        }     db.close();

        //Delete a product from the database
        public void deleteProduct(String productName){
                SQLiteDatabase db = getWritableDatabase();
                db.execSQL("DELETE FROM " + TABLE_PRODUCTS + " WHERE " + COLUMN_PRODUCTNAME +
"=\"" + productName + "\";");
        }
        public void deleteAllValues() //Deleting the name from the profile
        {
```

```
        SQLiteDatabase db = getWritableDatabase();
        db.execSQL("DELETE FROM " + TABLE_PRODUCTS + ";");
        db.close();
                                                                                        }

    public void deleteAllValues_c()//Deleting all the contacts from the Emergency
                                                                                          List//
                                                                                          {
        SQLiteDatabase db = getWritableDatabase();
        db.execSQL("DELETE FROM " + TABLE_PRODUCTS_C + ";");
        db.close();
                                                                                          }


    public String databaseToString(){ String
         dbString = "";
         SQLiteDatabase db = getWritableDatabase();
         String query = "SELECT * FROM " + TABLE_PRODUCTS + " WHERE 1";

         //Cursor points to a location in your results
         Cursor c = db.rawQuery(query, null);
         //Move to the first row in your results

         c.moveToFirst();

         //Position after the last row means the end of the results while
         (!c.isAfterLast()) {
              if (c.getString(c.getColumnIndex("productname")) != null) { dbString +=
                   c.getString(c.getColumnIndex("productname" ));
              }

         }     c.moveToNext();
         db.close(); return
         dbString;
    }
    public String databaseToString_c(){ String
         dbString = "";
         SQLiteDatabase db = getWritableDatabase();
         String query = "SELECT * FROM " + TABLE_PRODUCTS_C + " WHERE 1";

         //Cursor points to a location in your results
         Cursor c = db.rawQuery(query, null);
         //Move to the first row in your results

         c.moveToFirst();

         //Position after the last row means the end of the results while
         (!c.isAfterLast()) {
              if (c.getString(c.getColumnIndex("productname")) != null) { dbString +=

                   c.getString(c.getColumnIndex("productname" )); dbString += "\n";


              }
              if (c.getString(c.getColumnIndex("productnum")) != null) { dbString +=
                   c.getString(c.getColumnIndex("productnum" )); dbString += "\n";

              }

         }     c.moveToNext();
         db.close();

    }    return dbString;


    public void sendSMS() //This method sends the sms to all the contacts in the
```

```
Emergency Contacts List//
        {

                String d = "";
                SQLiteDatabase db = getWritableDatabase();
                String query = "SELECT * FROM " + TABLE_PRODUCTS + " WHERE 1";

                //Cursor points to a location in your results
                Cursor c = db.rawQuery(query, null);
                //Move to the first row in your results

                c.moveToFirst();

                //Position after the last row means the end of the results while
                (!c.isAfterLast()) {
                        if (c.getString(c.getColumnIndex("productname")) != null) { d +=
                                c.getString(c.getColumnIndex("productname" ));
                        }

                } c.moveToNext();

                db.close();
                  db = getWritableDatabase();
                  query = "SELECT * FROM " + TABLE_PRODUCTS_C + " WHERE 1";

                //Cursor points to a location in your results c =
                  db.rawQuery(query, null);
                //Move to the first row in your results

                c.moveToFirst();

                //Position after the last row means the end of the results while
                (!c.isAfterLast()) {
                        String phoneNo = null;
                        String sms = null;
                        String num;
                        num = c.getString(c.getColumnIndex("productnum")); if (num

                        != null) {


                                phoneNo = num;
                                sms ="This is "+ d +". I am in danger." + "Help me!";
                                try {
                                        SmsManager smsManager = SmsManager. getDefault();
                                        smsManager.sendTextMessage(phoneNo, null, sms, null, null);
                                } catch (Exception e) {

                                        e.printStackTrace();
                                }

                        }

                }       c.moveToNext();

                db.close();

          }


    }
```

Appendix 2.H

Android Manifest file

This file basically controls all the Activities of the app and we specify all the permissions in this file.

Here is the XML code for that

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

    package="com.example.siddharth.tcoe">

/>  <permission android:name="android.permission.BLUETOOTH" android:label="BLUETOOTH"
    <permission android:name="android.permission.BLUETOOTH_ADMIN" /> <uses-permission
    android:name="android.permission.BLUETOOTH"/> <uses-permission
    android:name="android.permission.BLUETOOTH_ADMIN"/> <uses-permission
    android:name="android.permission.ACCESS_COARSE_LOCATION"/>
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.RECORD_AUDIO" /> <uses-permission
    android:name="android.permission.SEND_SMS" /> <uses-permission
    android:name="android.permission.READ_CONTACTS" />
    <application android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">

        <activity android:name=".MainActivity"> <intent-
            filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" /> </intent-
            filter>
        </activity>
        <activity android:name=".emergencyActivity" /> <activity
        android:name=".EmActivity" /> <activity
        android:name=".ProfilePage"></activity>
    </application >


</manifest>
```

References

1) https://www.arduino.cc/en/Reference/SoftwareSerial Working of Software Serial software module in Arduino
2) https://developer .android.com/guide/topics/connectivity/bluetooth.html Interfacing of android application using Android Studio's Bluetooth class support
3) https://en.wikipedia.org/wiki/Wearable_technology Page on wearable technology's advent in the market.
4) https://www.arduino.cc/en/Guide/Introduction Page on the Arduino development board.
5) https://developer.android.com/ Reference material for any topic related to development of android application.
6) https://www.javacodegeeks.com/tutorials/android-tutorials/android-core-tutorials/ Reference for specific topics like SMS, recording.