Title: *Smart Health Care Monitoring System Based on IoT*

I. Summary

This project is significant in various ways because in today's world, everyday many lives are affected because the patients are not timely and properly operated. Also for real time parameter values are not efficiently measured in clinic as well as in hospitals. Sometimes it becomes difficult for hospitals to frequently check patient's conditions. Also continuous monitoring of ICU patients is very difficult. To deal with these types of situations, this system is beneficial. The system is designed to be used in hospitals and homes also for measuring and monitoring various parameters like temperature, ECG, heart rate, blood pressure. The results can be recorded using Arduino. Also the doctors can see those results on android app. The system will also generate an alert notification which will be sent to doctor. The system is useful for monitoring health system of every person by easily attaching the device and record it. In which the users can analysis patient's condition through their past data. The information will recommend medicines if any emergency occurred through symbolic A.I.

II. Objective

The purpose of this project is to gain experience in using the Arduino and android app in practical applications. The project displays real-world application of the products in a medical environment.

III. Industry-Based Application

This project is related to industry by many companies are implementing the internet of things into many products. These products are seen in everyday in products,

such as, televisions, refrigerators, smart speakers and more. Therefore, it is important for student to become familiar with IoT devices and the construction of them. The IoT devices will be the future of many engineering design projects. This may relate to a professional engineering job by an engineer may design smart devices in the medical field and other industries. Many engineers may get in this field with the evolution of smart products, such as smart lights, smart homes, smart appliances, smart televisions, smart phones and more. Also, microcontrollers are being implemented into various fields. Basic understanding of these devices are beneficial to the students.

IV. Methodology

a. Components

i. Arduino UNO and Genuino UNO

ii. HC-05 Bluetooth Module

iii. SparkFun SingleLead Heart Rate Monitor

iv. DFRobot Gravity: Analog LM35 Temperature Sensor for Arduino

v. ProtoCentral Pulse Oximeter & Heart Rate Sensor based on MAX30100

vi. Breadboard

vii. Sunrom Blood Pressure Module

viii. ON semiconductor EU-SIGFOX-GEVB

ix. Jumper Wire Kit

b. Application and Online Services

i. Android Studio

ii. Arduino IDE

iii. Google Firebase

c. Procedure

i. Establish communication between the Arduino and andriod app using Bluetooth. The Code for this communication is in the appendix.

ii. Send data to Arduino

iii. Data transfer between Android and Arduino

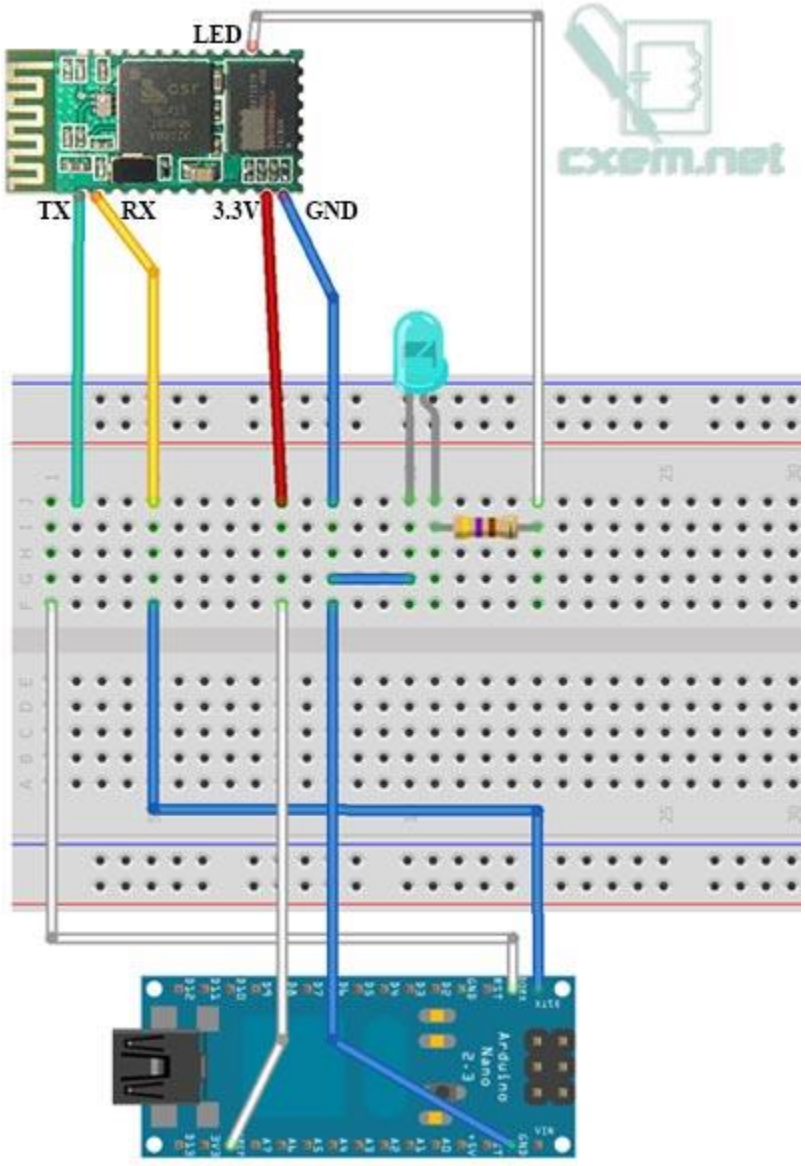


Figure 1- This is the Arduino Nana V3 and Bluetooth module HC-06

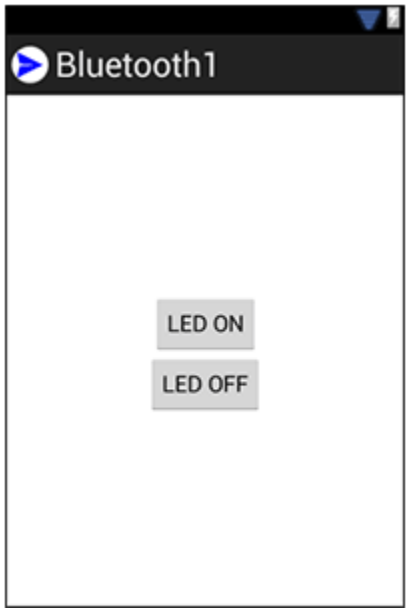
iv. The program is code 1 to program the Arduino

v. A java code with an explicit MAC-address is used

vi. Find MAC-address can be program for Android:Bluetooth Terminal

vii. The MAC-address is 00:15:FF:F2:19:4C. and the device name is BOLUTEK.

viii. The first program is very simple, the main activity contain two buttons: turn on LED and turn off LED. Data transfer in the program will be implemented only on Android devices to Arduino.



ix. The java code main activity is listed in code 2.

x. To receive data from Arduino, in Android application we need to use threads. On the main window activity, we add a new element TextView, which will be used to display the received data from the Arduino. The code is listed in code 3

xi. If the student gets lost use the following link: https://solderer.tv/data-transfer-between-android-and-arduino-via-bluetooth/

xii. Start creating the android app from the following steps and link above.

xiii. After this send some data to arduino in our scenario we have 4 module so we have 16 combination like patient want only temperature data or ecg, Blood pressure or temperature and ecg etc. so for these 16 combination we use hexa table which stars from 0 and ends at 'F' so after reading the incoming value arduino decide which type of data user wants

xiv. After that arduino start reading that data from sensors and transfer that data to arduino using HC05 and android receive that data and send to firebase which is real time database below is the structure of firebase database

Figure 2- The figure displays the complete architecture of the project

Figure 3- The figure displays the complete architecture of the project

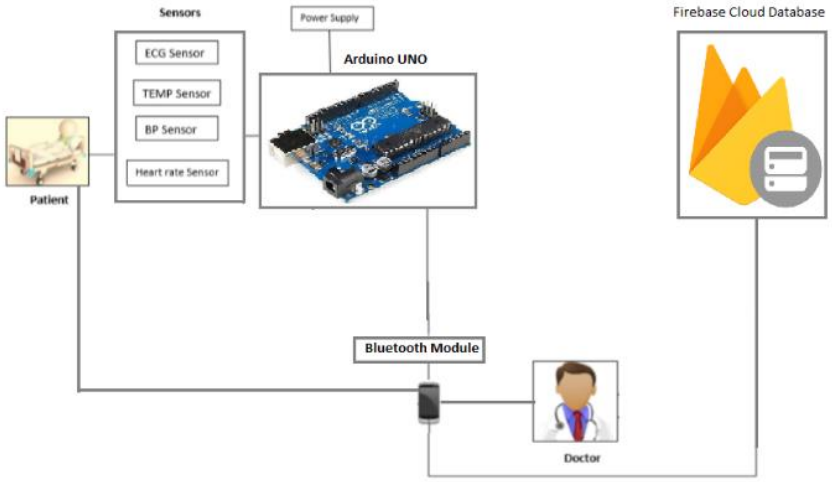
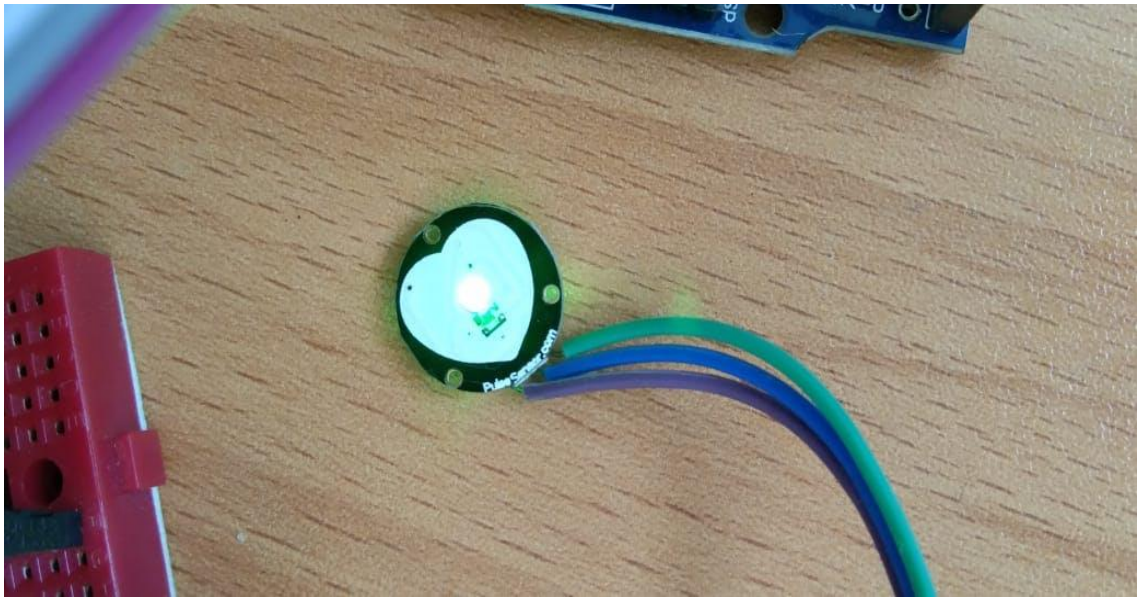


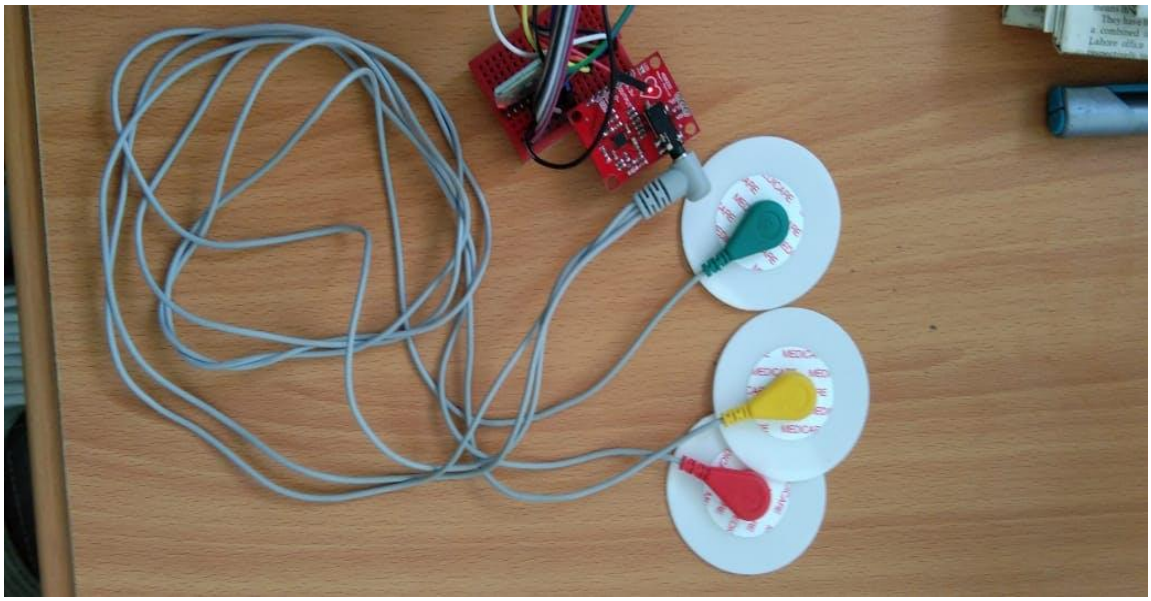
Figure 4- The figure above the block diagram for the system.

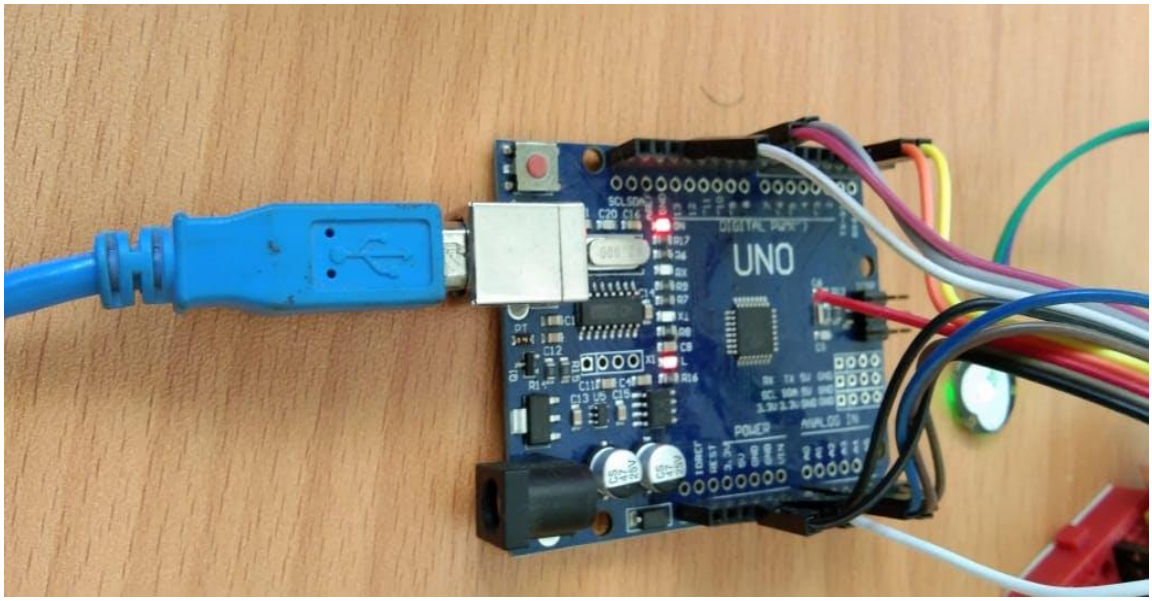
V. References

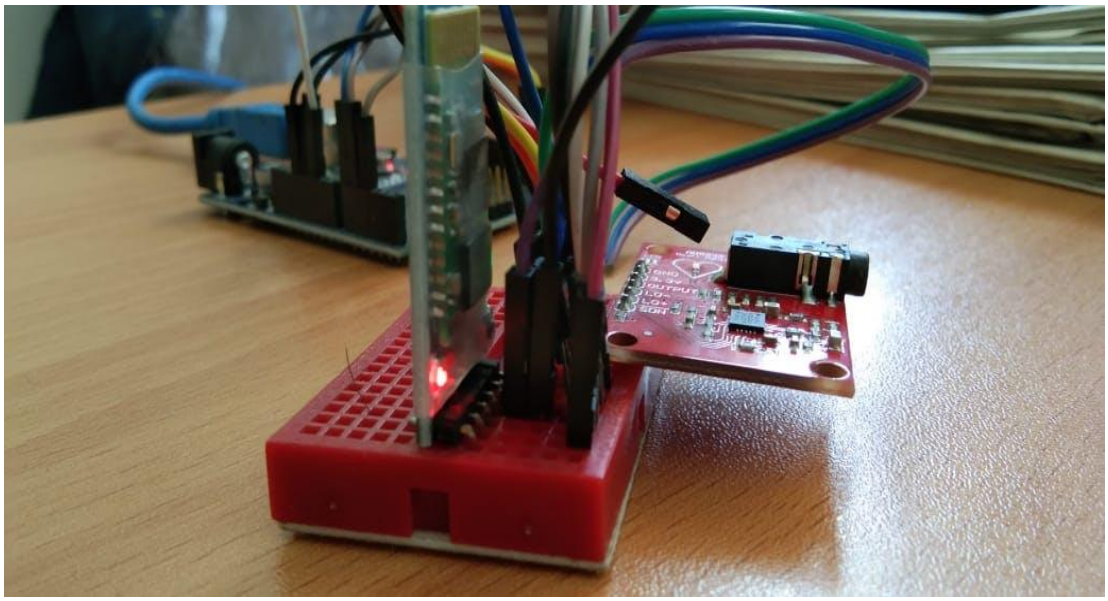
Website:https://create.arduino.cc/projecthub/156471/smart-health-care-monitoring-system-based-on-iot-f559b3

VI. Appendix






UNO



Code 1:

```
char incomingByte;  // incoming data
int  LED = 12;      // LED pin

void setup() {
  Serial.begin(9600); // initialization
  pinMode(LED, OUTPUT);
  Serial.println("Press 1 to LED ON or 0 to LED OFF...");
}

void loop() {
  if (Serial.available() > 0) {  // if the data came
    incomingByte = Serial.read(); // read byte
    if(incomingByte == '0') {
       digitalWrite(LED, LOW);  // if 1, switch LED Off
```

```
      Serial.println("LED OFF. Press 1 to LED ON!");  // print message
    }
    if(incomingByte == '1') {
      digitalWrite(LED, HIGH); // if 0, switch LED on
      Serial.println("LED ON. Press 0 to LED OFF!");
    }
  }
}
```

Code 2:

```
ackage com.example.bluetooth1;

import java.io.IOException;
import java.io.OutputStream;
import java.lang.reflect.Method;
import java.util.UUID;

import com.example.bluetooth1.R;

import android.app.Activity;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.Intent;
```

```
import android.os.Build;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.Toast;

public class MainActivity extends Activity {
  private static final String TAG = "bluetooth1";

  Button btnOn, btnOff;

  private BluetoothAdapter btAdapter = null;
  private BluetoothSocket btSocket = null;
  private OutputStream outStream = null;

  // 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 = "00:15:FF:F2:19:5F";
```

```
  /** Called when the activity is first created. */
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    setContentView(R.layout.activity_main);

    btnOn = (Button) findViewById(R.id.btnOn);
    btnOff = (Button) findViewById(R.id.btnOff);

    btAdapter = BluetoothAdapter.getDefaultAdapter();
    checkBTState();

    btnOn.setOnClickListener(new OnClickListener() {
      public void onClick(View v) {
        sendData("1");
        Toast.makeText(getBaseContext(), "Turn on LED",
Toast.LENGTH_SHORT).show();
      }
    });

    btnOff.setOnClickListener(new OnClickListener() {
```

```
    public void onClick(View v) {
      sendData("0");
      Toast.makeText(getBaseContext(), "Turn off LED",
Toast.LENGTH_SHORT).show();
    }
   });
  }

  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 e1) {
       errorExit("Fatal Error", "In onResume() and socket create failed: " +
e1.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...");

    try {
      outStream = btSocket.getOutputStream();
    } catch (IOException e) {
```

```
    errorExit("Fatal Error", "In onResume() and output stream creation failed:" +
e.getMessage() + ".");
   }
  }

  @Override
  public void onPause() {
   super.onPause();

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

   if (outStream != null) {
    try {
     outStream.flush();
    } catch (IOException e) {
     errorExit("Fatal Error", "In onPause() and failed to flush output stream: " +
e.getMessage() + ".");
    }
   }

   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();
  }

  private void sendData(String message) {
    byte[] msgBuffer = message.getBytes();

    Log.d(TAG, "...Send data: " + message + "...");

    try {
      outStream.write(msgBuffer);
    } catch (IOException e) {
      String msg = "In onResume() and an exception occurred during write: " +
e.getMessage();
      if (address.equals("00:00:00:00:00:00"))
        msg = msg + ".\n\nUpdate your server address from 00:00:00:00:00:00 to the
correct address on line 35 in the java code";
        msg = msg +  ".\n\nCheck that the SPP UUID: " + MY_UUID.toString() + " exists
on server.\n\n";

        errorExit("Fatal Error", msg);
    }
```

```
  }
}
```

Code 3:

```
  package com.example.bluetooth2;


  import java.io.IOException;
  import java.io.InputStream;
  import java.io.OutputStream;
  import java.lang.reflect.Method;
  import java.util.UUID;


  import com.example.bluetooth2.R;


  import android.app.Activity;
  import android.bluetooth.BluetoothAdapter;
  import android.bluetooth.BluetoothDevice;
  import android.bluetooth.BluetoothSocket;
  import android.content.Intent;
  import android.os.Build;
  import android.os.Bundle;
  import android.os.Handler;
  import android.util.Log;
```

```
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

public class MainActivity extends Activity {
  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 = "00:15:FF:F2:19:5F";

  /** Called when the activity is first created. */
  @Override
  public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    setContentView(R.layout.activity_main);

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

    h = new Handler() {
        public void handleMessage(android.os.Message msg) {
            switch (msg.what) {
            case  RECIEVE_MESSAGE:                                    // if receive
massage
                byte[] readBuf = (byte[]) msg.obj;
```

```
            String strIncom = new String(readBuf, 0, msg.arg1);                 // create
string from bytes array
            sb.append(strIncom);                                                // append string
            int endOfLineIndex = sb.indexOf("\r\n");                            // determine the
end-of-line
            if (endOfLineIndex > 0) {                                           // if end-of-line,
               String sbprint = sb.substring(0, endOfLineIndex);                // extract
string
               sb.delete(0, sb.length());                                       // and clear
               txtArduino.setText("Data from Arduino: " + sbprint);             // update
TextView
               btnOff.setEnabled(true);
               btnOn.setEnabled(true);
            }
            //Log.d(TAG, "...String:"+ sb.toString() +  "Byte:" + msg.arg1 + "...");
            break;
         }
      };
   };

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

```
    btnOn.setOnClickListener(new OnClickListener() {
      public void onClick(View v) {
        btnOn.setEnabled(false);
        mConnectedThread.write("1");    // Send "1" via Bluetooth
        //Toast.makeText(getBaseContext(), "Turn on LED",
Toast.LENGTH_SHORT).show();
      }
    });

    btnOff.setOnClickListener(new OnClickListener() {
      public void onClick(View v) {
        btnOff.setEnabled(false);
        mConnectedThread.write("0");    // Send "0" via Bluetooth
        //Toast.makeText(getBaseContext(), "Turn off LED",
Toast.LENGTH_SHORT).show();
      }
    });
  }

  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();
}

private class ConnectedThread extends Thread {
    private final InputStream mmInStream;
    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() + "...");
              }
        }
    }
}
```

Code 4:

```
#define USE_ARDUINO_INTERRUPTS true
#include <PulseSensorPlayground.h>
#include <SoftwareSerial.h>

SoftwareSerial mySerial(10, 11); // RX, TX
const int PulseWire = 0;
int tempPin = 1;
int ECGPin = 2;
String incomingByte;
int value;
int myBPM =0;
int finalBPM ;
float volts=0.0;
float cel =0;
String stringTemp;
String stringBPM;
```

```
bool is_data_send=false;
bool ISBP=false;
bool ISECG=false;
bool ISHR=false;
bool IStemp=false;
const int LED13 = 13;
int Threshold = 550;
String data="";


PulseSensorPlayground pulseSensor;


void setup() {

Serial.begin(38400);
Serial.println("Enter AT commands:");
mySerial.begin(38400);

  pulseSensor.analogInput(PulseWire);
  pulseSensor.blinkOnPulse(LED13);
  pulseSensor.setThreshold(Threshold);
```

```
  if (pulseSensor.begin()) {
   Serial.println("We created a pulseSensor Object !");
  }
}



void loop() {

if (mySerial.available()) {

   incomingByte = mySerial.read();
   Serial.println(incomingByte);

   if(incomingByte=="49")
   {ISECG=false;ISBP=false;ISHR=false;IStemp=true;is_data_send=true;}
   else if(incomingByte=="50")
   {ISECG=false;ISBP=false;ISHR=true;IStemp=false;is_data_send=true;}
   else if(incomingByte=="51")
   {ISECG=false;ISBP=false;ISHR=true;IStemp=true;is_data_send=true;}
   else if(incomingByte=="52")
   {ISECG=false;ISBP=true;ISHR=false;IStemp=false;is_data_send=true;}
```

```
else if(incomingByte=="53")
{ISECG=false;ISBP=true;ISHR=false;IStemp=true;is_data_send=true;}
else if(incomingByte=="54")
{ISECG=false;ISBP=true;ISHR=true;IStemp=false;is_data_send=true;}
else if(incomingByte=="55")
{ISECG=false;ISBP=true;ISHR=true;IStemp=true;is_data_send=true;}
else if(incomingByte=="56")
{ISECG=true;ISBP=false;ISHR=false;IStemp=false;is_data_send=true;}
else if(incomingByte=="57")
{ISECG=true;ISBP=false;ISHR=false;IStemp=true;is_data_send=true;}
else if(incomingByte=="65")
{ISECG=true;ISBP=false;ISHR=true;IStemp=false;is_data_send=true;}
else if(incomingByte=="66")
{ISECG=true;ISBP=false;ISHR=true;IStemp=true;is_data_send=true;}
else if(incomingByte=="67")
{ISECG=true;ISBP=true;ISHR=false;IStemp=false;is_data_send=true;}
else if(incomingByte=="68")
{ISECG=true;ISBP=true;ISHR=false;IStemp=true;is_data_send=true;}
 else if(incomingByte=="69")
{ISECG=true;ISBP=true;ISHR=true;IStemp=false;is_data_send=true;}
 else if(incomingByte=="70")
{ISECG=true;ISBP=true;ISHR=true;IStemp=true;is_data_send=true;}
```

```
}
//Getting Values from Sensors
if(is_data_send)
{
  int counter=0;
  if(IStemp||ISHR){
  while(counter<10){
  if(IStemp){
    value=analogRead(tempPin);
    volts=(value/1024.0)*5.0;      //conversion to volts
    cel = cel+(volts*100.0);
  }
  if(ISHR){
    if (pulseSensor.sawStartOfBeat()) {
   myBPM = myBPM+pulseSensor.getBeatsPerMinute();
}
      }


    counter=counter+1;
    }}
  if(ISBP){
    }
  int counter2=0;
```

```
  if(ISECG){
  while(counter2<500){
  if((digitalRead(2)==1)||(digitalRead(3)==1)){}
  else{
    String abc=String(analogRead(ECGPin));
    Serial.println(abc);
      data+=abc+"|";
  }
  counter2=counter2+1;
  }
  }
}
//Sending Data
  if(is_data_send){
    if(ISBP){
    }
    if(ISHR){
    float final_bpm=myBPM/10;
    stringBPM = String(final_bpm);
    data+=stringBPM+"|";
    }
     if(IStemp){
      float final_temp=cel/10;
```

```
      stringTemp = String(final_temp);
      data+=stringTemp+"|";
     }


    Serial.print("Data sending");
    Serial.print(data);
    //Serial.print(stringTemp+"|"+stringBPM);
    mySerial.print(data);
    mySerial.println();
    is_data_send=false;
  }
delay(20);                  // considered best practice in a simple sketch.
}
```