**Main Class**

*Code for socket connection for the sensor patch.*

package com.example.thesis\_glucose;

import android.content.Context;

import android.content.DialogInterface;

import android.os.Bundle;

import android.support.v7.app.AppCompatActivity;

import android.util.Log;

import android.view.View;

import android.widget.Button;

import android.widget.EditText;

import android.widget.TextView;

import android.widget.Toast;

import java.io.DataInputStream;

import java.io.IOException;

import java.io.OutputStreamWriter;

import java.net.Socket;

import android.app.AlertDialog;

import android.view.inputmethod.InputMethodManager;

import org.w3c.dom.Text;

public class MainActivity extends AppCompatActivity {

Button connectButton, disconnectButton, saveButton;

EditText heightEdit, weightEdit;

TextView BMITextView, glucoseLevelTextView, insulinTextView;

GlucoseLevelMeasurement glucObj;

Socket s;

String message;

DataInputStream din;

MyThread myThread = new MyThread();

Thread t = new Thread(myThread);

boolean isThreadStarted = false;

boolean isSocketNext = false;

boolean isDisconnectNext = false;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_main);

connectButton = (Button) findViewById(R.id.connectButton);

disconnectButton = (Button) findViewById(R.id.disconnectButton);

saveButton = (Button) findViewById(R.id.saveButton);

heightEdit = (EditText) findViewById(R.id.heightEdit);

weightEdit = (EditText) findViewById(R.id.weightEdit);

BMITextView = (TextView) findViewById(R.id.BMITextView);

glucoseLevelTextView = (TextView) findViewById(R.id.glucoseLevelTextView);

insulinTextView = (TextView) findViewById(R.id.insulinTextView);

glucObj = new GlucoseLevelMeasurement();

final AlertDialog.Builder dlgAlert = new AlertDialog.Builder(this);

final android.support.v7.app.AlertDialog.Builder builder = new android.support.v7.app.AlertDialog.Builder(this);

builder.setTitle("Confirm");

builder.setMessage("Are you sure you want to save current reading?");

builder.setPositiveButton("YES", new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog, int which) {

String BMI = BMITextView.getText().toString();

String glucose\_level = glucoseLevelTextView.getText().toString();

String insulin\_level = insulinTextView.getText().toString();

if(!glucose\_level.isEmpty() && !BMI.isEmpty() && !insulin\_level.isEmpty()) {

}

writeToFile("Test");

dialog.dismiss();

}

});

builder.setNegativeButton("NO", new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialog, int which) {

// Do nothing

dialog.dismiss();

}

});

connectButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

try {

double height = Double.parseDouble(heightEdit.getText().toString());

double weight = Double.parseDouble(weightEdit.getText().toString());

BMITextView.setText(Double.toString(glucObj.getBMILevel(weight, height)));

} catch (Exception e) {

dlgAlert.setMessage("Invalid height and/or weight!");

dlgAlert.setTitle("Error");

dlgAlert.setPositiveButton("OK", null);

dlgAlert.setCancelable(true);

dlgAlert.create().show();

return;

}

try {

if (!isThreadStarted) {

t.start();

isThreadStarted = true;

}

if (!myThread.getStarted()) {

myThread.setStarted(true);

isSocketNext = true;

}

} catch (Exception e) {

dlgAlert.setMessage("Network error, please restart the app and connect to the device patch again.");

dlgAlert.setTitle("Error");

dlgAlert.setPositiveButton("OK", null);

dlgAlert.setCancelable(true);

dlgAlert.create().show();

}

InputMethodManager inputManager = (InputMethodManager)

getSystemService(Context.INPUT\_METHOD\_SERVICE);

inputManager.hideSoftInputFromWindow(getCurrentFocus().getWindowToken(),

InputMethodManager.HIDE\_NOT\_ALWAYS);

}

});

disconnectButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

try {

if (myThread.getStarted()) {

myThread.setStarted(false);

isDisconnectNext = true;

heightEdit.setText("");

weightEdit.setText("");

BMITextView.setText("-");

glucoseLevelTextView.setText("-");

insulinTextView.setText("-");

}

heightEdit.setText("");

weightEdit.setText("");

BMITextView.setText("-");

glucoseLevelTextView.setText("-");

insulinTextView.setText("-");

} catch (Exception e) {

Toast.makeText(getApplicationContext(), e.getMessage(), Toast.LENGTH\_LONG);

}

}

});

saveButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

android.support.v7.app.AlertDialog alert = builder.create();

alert.show();

}

});

}

private void writeToFile(String data) {

try {

OutputStreamWriter outputStreamWriter = new OutputStreamWriter(this.openFileOutput("data.txt", this.MODE\_APPEND));

outputStreamWriter.write(data+"\n");

outputStreamWriter.close();

}

catch (IOException e) {

Log.e("Exception", "File write failed: " + e.toString());

}

}

class MyThread implements Runnable {

private boolean started = false;

int ctr = 0;

@Override

public void run() {

try {

while (true) {

if (getStarted()) {

if (isSocketNext) {

// Connect to socket.

// Needs to be done on this thread.

s = new Socket("192.168.4.1", 8888);

isSocketNext = false;

}

// Get data.

din = new DataInputStream(s.getInputStream());

message = din.readLine();

String NIR\_reading=message;

glucObj.setGlucoseLevel(Double.parseDouble(NIR\_reading));

glucoseLevelTextView.setText(String.valueOf(glucObj.getGlucoseLevel()));

insulinTextView.setText(String.valueOf(glucObj.getInsulin()));

runOnUiThread(new Runnable() {

@Override

public void run() {

if (glucObj.getGlucoseLevel() < 60 && glucObj.getGlucoseLevel()>300) {

glucoseLevelTextView.setText("Out of calibration range.");

insulinTextView.setText("Out of calibration range");

}

else

{

}

}

});

} else if (isDisconnectNext) {

s.close();

isDisconnectNext = false;

}

else

{

}

Thread.sleep(500);

}

} catch (Exception e) {

Toast.makeText(getApplicationContext(), e.getMessage(), Toast.LENGTH\_LONG);

}

}

public void setStarted(boolean started) {

this.started = started;

}

public boolean getStarted() {

return this.started;

}

}

}

**Logic Class**

package com.example.thesis\_glucose;

public class GlucoseLevelMeasurement {

private double BMI;

private double glucoseLevel;

public GlucoseLevelMeasurement() {

}

public double getInsulin()

{

if(this.BMI<25)

{

// Get glucose level.

if(this.glucoseLevel<151)

{

return 0.0; // No insulin.

}

else

{

if(this.glucoseLevel>=151 && this.glucoseLevel<=175)

{

return 1.4;

}

else if(this.glucoseLevel>=176 && this.glucoseLevel<=200)

{

return 2.4;

}

else if(this.glucoseLevel>=201 && this.glucoseLevel<=225)

{

return 3.4;

}

else if(this.glucoseLevel>=226 && this.glucoseLevel<=250)

{

return 5.5;

}

else if(this.glucoseLevel>=251 && this.glucoseLevel<=275)

{

return 7.5;

}

else if(this.glucoseLevel>=276 && this.glucoseLevel<=300)

{

return 9.4;

}

else if(this.glucoseLevel<300)

{

return 12;

}

}

}

else if(this.BMI>=25 && this.BMI<=30)

{

// Get glucose level.

if(this.glucoseLevel<151)

{

return 0.0; // No insulin.

}

else

{

if(this.glucoseLevel>=151 && this.glucoseLevel<=175)

{

return 2.5;

}

else if(this.glucoseLevel>=176 && this.glucoseLevel<=200)

{

return 4.4;

}

else if(this.glucoseLevel>=201 && this.glucoseLevel<=225)

{

return 6.6;

}

else if(this.glucoseLevel>=226 && this.glucoseLevel<=250)

{

return 8.5;

}

else if(this.glucoseLevel>=251 && this.glucoseLevel<=275)

{

return 10.5;

}

else if(this.glucoseLevel>=276 && this.glucoseLevel<=300)

{

return 12.5;

}

else if(this.glucoseLevel<300)

{

return 14.5;

}

}

}

else if(this.BMI>=25)

{

// Get glucose level.

if(this.glucoseLevel<151)

{

return 0.0; // No insulin.

}

else

{

if(this.glucoseLevel>=151 && this.glucoseLevel<=175)

{

return 3.6;

}

else if(this.glucoseLevel>=176 && this.glucoseLevel<=200)

{

return 6.6;

}

else if(this.glucoseLevel>=201 && this.glucoseLevel<=225)

{

return 7.6;

}

else if(this.glucoseLevel>=226 && this.glucoseLevel<=250)

{

return 9.6;

}

else if(this.glucoseLevel>=251 && this.glucoseLevel<=275)

{

return 11.6;

}

else if(this.glucoseLevel>=276 && this.glucoseLevel<=300)

{

return 14.6;

}

else if(this.glucoseLevel<300)

{

return 18.6;

}

}

}

return 0.0;

}

public double getBMILevel(double weight, double height) {

// BMI formula, imperial

//this.BMI=Math.round(((float) weight / (Math.pow(height, 2))) \* 100.0) / 100.0;

this.BMI = Math.round(((weight\*703)/Math.pow(height,2))\*100.0)/100.0;

return this.BMI;

}

public double getGlucoseLevel()

{

return this.glucoseLevel;

}

public void setGlucoseLevel(double NIR\_reading)

{

this.glucoseLevel=NIR\_reading;

}

}

**History Class**

package com.example.thesis\_glucose;

import android.content.Context;

import android.content.DialogInterface;

import android.support.v7.app.AlertDialog;

import android.support.v7.app.AppCompatActivity;

import android.os.Bundle;

import android.util.Log;

import android.view.View;

import android.widget.Button;

import android.widget.TextView;

import java.io.BufferedReader;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.io.InputStream;

import java.io.InputStreamReader;

import java.io.OutputStreamWriter;

import java.io.PrintWriter;

public class HistoryActivity extends AppCompatActivity {

TextView dataTextView;

Button clearHistoryButton;

@Override

protected void onCreate(Bundle savedInstanceState) {

super.onCreate(savedInstanceState);

setContentView(R.layout.activity\_history);

dataTextView = findViewById(R.id.dataTextView);

clearHistoryButton = findViewById(R.id.clearHistoryButton);

final AlertDialog.Builder builder = new AlertDialog.Builder(this);

builder.setTitle("Confirm");

builder.setMessage("Are you sure you want to clear the history?");

builder.setPositiveButton("YES", new DialogInterface.OnClickListener() {

public void onClick(DialogInterface dialog, int which) {

try {

OutputStreamWriter outputStreamWriter = new OutputStreamWriter(getApplicationContext().openFileOutput("data.txt", Context.MODE\_PRIVATE));

outputStreamWriter.close();

String data = readFromFile(getApplicationContext());

dataTextView.setText(data);

} catch (IOException e) {

e.printStackTrace();

}

dialog.dismiss();

}

});

builder.setNegativeButton("NO", new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialog, int which) {

// Do nothing

dialog.dismiss();

}

});

clearHistoryButton.setOnClickListener(new View.OnClickListener() {

@Override

public void onClick(View v) {

AlertDialog alert = builder.create();

alert.show();

}

});

//writeToFile("RomTest Space", this);

String data = readFromFile(this);

dataTextView.setText(data);

//Log.e("Data", data);

}

private void writeToFile(String data, Context context) {

try {

OutputStreamWriter outputStreamWriter = new OutputStreamWriter(context.openFileOutput("data.txt", Context.MODE\_APPEND));

outputStreamWriter.write(data+"\n");

outputStreamWriter.close();

}

catch (IOException e) {

Log.e("Exception", "File write failed: " + e.toString());

}

}

private String readFromFile(Context context) {

String ret = "";

try {

InputStream inputStream = context.openFileInput("data.txt");

if ( inputStream != null ) {

InputStreamReader inputStreamReader = new InputStreamReader(inputStream);

BufferedReader bufferedReader = new BufferedReader(inputStreamReader);

String receiveString = "";

StringBuilder stringBuilder = new StringBuilder();

while ( (receiveString = bufferedReader.readLine()) != null ) {

Log.e("Data", receiveString);

stringBuilder.append(receiveString).append("\n");

}

inputStream.close();

ret = stringBuilder.toString();

}

inputStream.close();

}

catch (FileNotFoundException e) {

Log.e("login activity", "File not found: " + e.toString());

} catch (IOException e) {

Log.e("login activity", "Can not read file: " + e.toString());

}

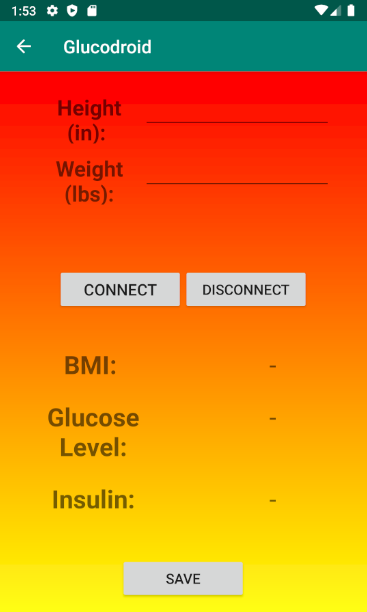
return ret;

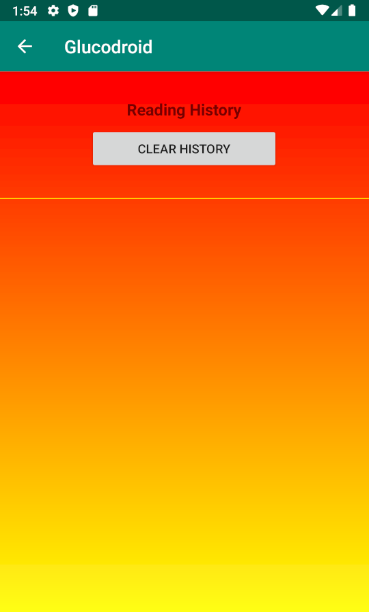
}

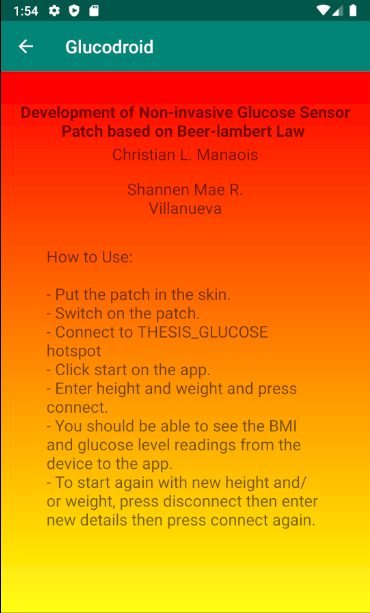
}

**Screenshots of the App**









**Sensor Patch Code**

#include <ESP8266WiFi.h>

#include <WiFiClient.h>

#include <ESP8266WebServer.h>

#include "AS726X.h"

AS726X sensor;

#ifndef APSSID

#define APSSID "THESIS\_GLUCOSE"

#define APPSK "thesis\_glucose"

#endif

/\* Set these to your desired credentials. \*/

const char \*ssid = APSSID;

const char \*password = APPSK;

// Socket

int port = 8888; //Port number

WiFiServer server(port);

// For testing only.

int ctr = 0;

void setup()

{

Serial.begin(9600);

Serial.println("Initializing...");

Wire.begin();

sensor.begin();

sensor.disableBulb();

Serial.println();

Serial.print("Configuring access point...");

/\* You can remove the password parameter if you want the AP to be open. \*/

WiFi.softAP(ssid, password);

IPAddress myIP = WiFi.softAPIP();

Serial.print("AP IP address: ");

Serial.println(myIP);

server.begin();

}

void loop()

{

WiFiClient client = server.available();

if (client) {

while (client.connected()) {

//sensor.enableBulb();

Serial.println("Client connected!");

sensor.takeMeasurements();

if (sensor.getVersion() == SENSORTYPE\_AS7263)

{

//Near IR readings

// Serial.print(" Reading: R[");

// Serial.print(sensor.getCalibratedR(), 2);

// Serial.print("] S[");

// Serial.print(sensor.getCalibratedS(), 2);

// Serial.print("] T[");

// Serial.print(sensor.getCalibratedT(), 2);

// Serial.print("] U[");

// Serial.print(sensor.getCalibratedU(), 2);

// Serial.print("] V[");

// Serial.print(sensor.getCalibratedV(), 2);

// Serial.print("] W[");

// Serial.print(sensor.getCalibratedW(), 2);

String data = sensor.getCalibratedR();

Serial.println(data);

client.println(data);

}

// // Testing for receiving from client.

// while (client.available() > 0) {

// char c = client.read();

// Serial.write(c);

// }

//

// while (Serial.available() > 0)

// {

// client.println(Serial.readString());

// }

//

//

// client.println(ctr);

// Serial.println(ctr);

// ctr++;

delay(1000);

}

client.stop();

sensor.disableBulb();

Serial.println("Client disconnected");

}

}