1

Optimización de Sensor de Pulsoximetria: Creación de App Android con comunicación Bluetooth y transvase de información multiplataforma

Giorgio Zamataro, *MSs, IEEE*

***Abstract*—En este Articulo se comentan los pasos que han llegado a la optimización de un oximetro de pulso desarrollado por la Universidad de Sevilla donde ha sido añadida una interfaz para la visualización de los datos en móviles Android a través de comunicación vía Bluetooth. Ademas la aplicación desarrollada permite la toma de notas sobre el estado de salud del paciente y da una medida continua de parámetros vitales importante. Se ha configurado un servidor web en la plataforma Google Firebase que permite de *Autenticarse* y de *Gestionar* las informaciones guardándolas en una plataforma de Storage Cloud**

***Index Terms*—Telemedicina, eSalud, Oximetria, Android, Arduino, Bluetooth.**

I. Introducción

EL Pulsoximetro es un dispositivo potable y de bajo coste que permite la medida continua no invasiva del ritmo cardíaco y de la saturación de oxigeno en la sangre. La oximetría de pulso es un avance tecnológico relativamente nuevo, desarrollado para uso médico durante los últimos 30 años. Desde su introducción para la monitorización preoperatoria se ha convertido en un estándar de de control y atención cardiovascular no invasivo debido a que es un método muy útil para evaluar la gravedad de la enfermedad y el control eficaz de las intervenciones terapéuticas.

Para conseguir una mejora en la salud de la ciudadanía y en su bienestar, se están realizando grandes avances en eSalud. El término eSalud hace referencia a la aplicación de las Tecnologías de la Información y Comunicación (TIC's) a todos aquellos aspectos que afectan al cuidado de la salud, incluyendo el diagnóstico, el seguimiento de los pacientes y la gestión de las organizaciones implicadas en estas actividades. Gracias al nacimiento y aplicación de la eSalud, los pacientes tienen la oportunidad de acceder a la atención médica con independencia del lugar en el que se encuentren. De esta manera, se fomenta la prevención, el bienestar y la vida independiente y autónoma de las personas. Asimismo, supone un ahorro de costes y un aumento de la eficacia de los sistemas sanitarios, contribuyendo a impulsar la calidad asistencial, reduciendo los tiempos de respuesta y diagnóstico de los facultativos, mejorando la efectividad terapéutica y la implantación de alertas.

En este sentido, los sistemas de eSalud junto a los dispositivos sensores inteligentes que permiten la supervisión en tiempo real de parámetros clínicos relevantes se consideran como un enfoque prometedor para la prevención y tratamiento de las enfermedades respiratorias, motivando a los pacientes a adoptar habilidades para su autotratamiento, mejorando así su calidad de vida y, por lo tanto, minimizando los costes en salud pública.

En este estudio se optimiza un sensor de pulsoximetria trasmitivo desarrollado en la Universidad de Sevilla que envía dos señales de intensidad luminosa a un fotodiodo a través de la luz absorbida por el dedo del paciente emitida por dos LED (Rojo y Infrarrojo). El enfoque de este trabajo es el desarrollo de una App Android para móvil para la visualización y el almacenamiento de los datos que llegan del sensor a través de la comunicación Bluetooth entre sensor y móvil.

G. Zamataro was with the Department of Biomedical Engineering, Universidad de Sevilla, Sevilla, AD, 41010 Spain
e-mail: giorgio.zamataro@mail.polimi.it

II. Materiales y Métodos

A. El principio físico

La cantidad de luz absorbida por el dedo del paciente depende de tres propriedades según la Ley de Beer-Lambert:

- Concentración de la sustancia que absorbe la luz (Fig.1)
- Longitud de la trayectoria en la sustancia que absorbe la luz (Fig.2)
- Diferencia de absorción de luz entre oxihemoglobina y hemoglobina reducida (Fig.3)

Se puede expresar dicha formula como:

$$I(d) = I_0 * e^{-\varepsilon_\lambda * c * (d-a)}$$

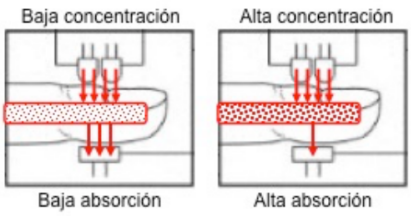

Fig. 1. Absorción de luz en función de la concentración de Hb

Como se nota en Fig.3 al aumentar de la longitud de onda de la radiación electromagnética que pasa por el tejido se encuentra un cambio de la absorción de dicha radiación por los dos tipos de hemoglobina. Se nota que por longitudes de onda de 660 nm (Luz Roja) la absorción de luz por la hemoglobina reducida es mayor que la absorción por la oxihemoglobina En cambio, e nota que por longitudes de onda de 940 nm (Luz

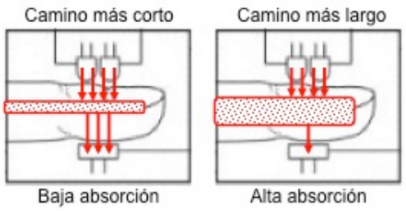

Fig. 2. Absorcion de luz en funcion de la longitud del recorrido

Infrarroja) la absorcion de luz por la hemoglobina reducida es menor que la absorción por la oxihemoglobina. A eso se debe la elección de usar dos señales (Rojo y Infrarrojo) para la identificación precisa de la cantidad de oxihemoglobina y de hemoglobina reducida en la sangre

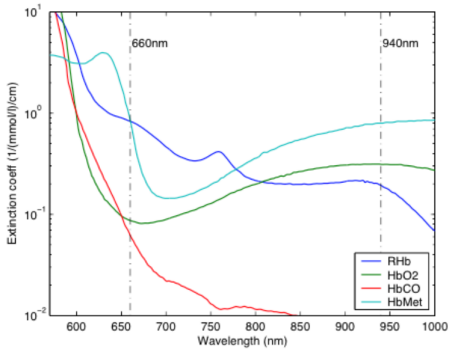

Fig. 3. Absorción de HbO 2 y RHb para diferentes longitudes de onda

B. Medida de la saturacion de oxigeno

La saturacion de oxigeno en la sangre ha sido calculada mediante la formula:

$$Sa02 = 115 - 30 * R$$

Donde R es la Razón de oximetría que puede ser calculada mediante dos métodos:

- El método de picos y valles
- El método de absorción Delta.

[Imagen dos metodos] Por simplicidad en este proyecto se ha elegido el método de picos y valles donde R se calcula como:

$$R = \frac{\partial A_{\lambda rojo}}{\partial A_{\lambda irojo}} = \frac{AC_{\lambda rojo}/DC_{\lambda rojo}}{AC_{\lambda irojo}/DC_{\lambda irojo}}$$

Que no es mas que el ratio de la normalizacion respecto a la continua de las dos señales roja y infrarroja.

C. Las Herramientas

Las herramientas software usada para el desarrollo software de este proyecto han sido:

- Android Studio®
- Arduino IDE®
- Matlab®
- Firebase®

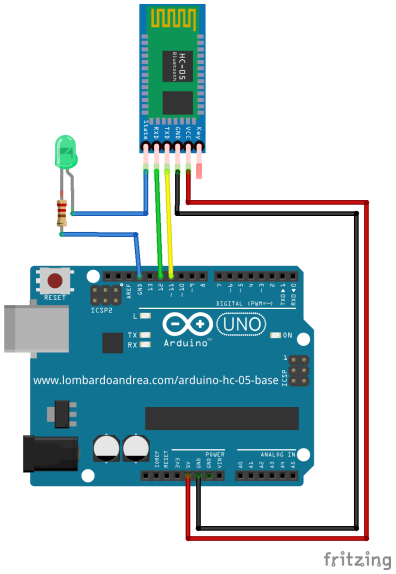

Fig. 4. Configuracion Hardware Arduino por el modulo HC-05/SPP-c

1) Matlab®: En fase de prototipizacion no ha sido posible utilizar el sensor entonces se ha optado por el envió de una señal fotopletismografica (Photoplethysmographic Signal, PPG) al Arduino vía serial por Matlab. Dicha señal ha sido creada usando la herramienta *"grabit.m"* que ha permitido a partir de una imagen de crear un array de puntos *(DATA.mat)* según una referencia estandar. Una vez elegido la señal y usado *grabit.m* para crear el array *DATA.mat* de los puntos salientes se ha creado otro array *(DATA_FITTING.mat)* mas denso que contenga todos los valores interpolados con método **Spline** de los puntos contenidos en el array *DATA.mat* así que la longitud del nuevo array alcance las **10.000 muestras**.

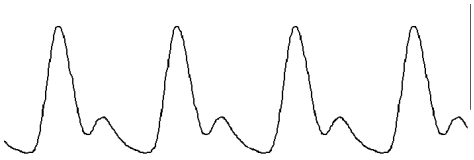

Fig. 5. Señal PPG usado para el envío tramite serial

Se ha creado un Script en Matlab® que permite de variar el pulso cardiaco y de condensar los valores del eje X de nuestro array en relación a esta variación así que se pueda probar la aplicación con señales con diferentes frecuencias cardíaca.

2) Bluetooth: Para la comunicación Bluetooth entre nuestro sensor y la aplicación móvil se ha elegido un modulo Bluetooth ***HC-05/SPP-C*** (Fig.4) que comunica con Arduino® por tres pines (**RX, TX, GND**) y tiene la ventaja de ser programable de manera sencilla a través de la libraría Arduino *<SoftwareSerial.h >* Dicha libraría contiene todas las funciones necesarias para empezar una comunicación asíncrona entre el Arduino y cualquier dispositivo que tiene los pines de escritura (TX) y lectura (RX).

Al principio se ha testado la comunicación Bluetooth a través del software **"Bluetooth Electronics"** de *Keuwlsoft®* que muestra en pantalla los paquetes enviados.

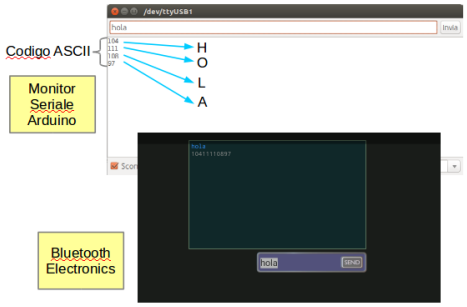

Fig. 6. Ejemplo de envío de la stringa "hola" mediante Bluetooth Electronics®

Pues con Android Studio se ha desarrollado la función *ConnectedThread()*(Ver Appendix C, Java Code) que permite el intercambio de paquetes de datos entre móvil y dispositivo.

3) El Paquete de informacion: Se ha elegido de enviar un paquete de 23 caracteres por cada instancia de comunicación. Cada paquete tiene un carácter iniciador (**#**) y un carácter terminador ($\sim$) mas un carácter de ajuste de linea para la visualización ($\backslash n$) que no tiene que ser considerado. Las informaciones proporcionadas en el paquete se refieren al nivel de voltaje (entre 0-5 V) que entra en cada uno de los cuatros pines analógicos (A0, A1, A2, A3) del Arduino®. El Arduino® incluye un ADC con 10 bit que divide el rango entre 0-5V en 1024 niveles. Cada nivel de voltaje leído se incluye dentro del paquete expresado con dos decimales y el carácter "mas" (**+**) como distanciador entre los valores de cada pin.

Un ejemplo de un paquete enviado:

$$\#1.23 + 4.56 + 7.89 + 0.12 + \sim \backslash n$$

Los primeros dos datos del paquete se refieren a las dos señales roja y infrarroja que llegan directamente al Arduino® por los pines A0 y A1. Posteriormente se ha elegido de usar el tercero dato del paquete para enviar el porcentaje de la saturación de oxigeno en la sangre y por fin el cuarto dato del paquete podría ser utilizado para enviar la medida de temperatura del cuerpo del paciente u otras informaciones relacionadas a la medida indirecta del oxigeno en la sangre.

4) Firebase: En lo que respecta al almacenamiento de los datos importantes del paciente se ha recurrido al uso de la plataforma Google Firebase. Google Firebase es una plataforma que ofrece servicios dedicados a los desarrolladores de aplicaciones móviles y web. Se ha usado esta plataforma para gestionar la Autenticación inicial de los usuarios así que se pueda tener traza de todas las informaciones, y para el Almacenamiento en un servidor Cloud de los datos de los paciente para evitar la perdida de datos en el caso de fallo de un componente físico. Ademas se han usado los servicios de Analytics para evaluar el trafico de uso del App y el Crush Report para tener traza de los errores y bugs mas frecuentes. Los datos almacenados están agrupados en Folder y Subfolder según un modelo jerárquico:

- Folder Usuario (ej: giorgio_zamataro@gmail_com)
- Folder Actividad (ej: Nota, Medida)
- Folder Fecha (ej: 15/06/2017_21:30:15)

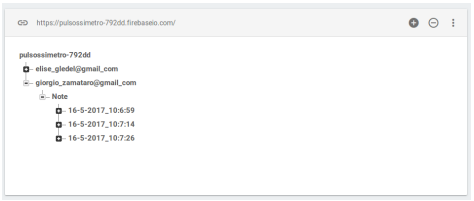

Fig. 7. Ejemplo de visualizacion del Database de los Usuarios en Google Firebase

D. Las Activities de Android Studio®

La App Android está dividida en Activities que son los bloques fundamentales para la construcción del App:

- **Login Activity**, que permite de acceder al servicio de Authenticacion de Firebase mediante e-mail y Password del Usuario
- **Signup Activity**, que permite de registrarse en el Servidor Firebase mediante e-mail y Password
- **Reset password Activity**, que permite de enviar tu Password tramite correo
- **Menu Activity**, que permite de elegir entre los botones:
  - Live Pulsoximetro
  - Histórico Medidas
  - Diario Personal
  - Configuración Account
- **Device List Activity**, que muestra el listado de los perifericos Bluetooth disponibles y permite de elegir nuestro dispositivo
- **Live Activity**, que permite de visualizar en tiempo real los valores de los señales de pulso y de la saturación graficando en una ventana la señal roja recibida
- **Historical Measures Activity**, que muestra el listado de las medidas salientes tomadas ordenando por fecha cada evento
- **Personal Note Activity**, que permite de añadir notas personales
- **View Notes Activity**, que muestra el listado de todas las notas tomadas ordenadas por fecha
- **Show Notes Activity**, que muestra el contenido de la nota seleccionada
- **Modify Note Activity**, que permite de modificar el contenido de una nota si modificar la fecha
- **Config Account Activity**, que permite de configurar el proprio usuario

III. Conclusion

Se ha desarrollado una Aplicación Android para la optimización un oximetro de pulso. La posibilidad de enviar datos a través de la comunicación Bluetooth entre Arduino® y el móvil ha sido el enfoque principal de este proyecto.

Ademas se ha usado la plataforma Google Firebase para la autenticación de los usuarios y para el almacenamiento de todas las informaciones salientes del paciente que han sido grabadas en un servidor propietario dentro la plataforma

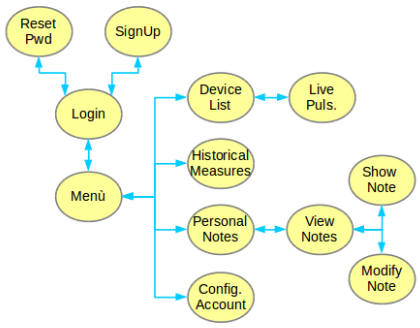

Fig. 8. Estructura del App segun las Activities usadas

Firebase. Se ha mostrado el transvase de la señal PPG entre diferentes plataformas pasando por Matlab®, Arduino® y al final llegando al móvil vía Bluetooth.

El avance aportado a este proyecto permite una evaluación y un procesamiento adecuado de las señale fisiológicas del paciente permitiendo de una optimización en el proceso de monitorización continua del estado de salud.

APPENDIX A
ARDUINO CODE

El siguiente código ha sido usado para una primera evaluación de la comunicación Bluetooth entre Arduino® y el móvil:

```
//Includo libreria SoftwareSerial
#include <SoftwareSerial.h>

//definisco pin RX e TX da Arduino verso modulo BT
#define BT_TX_PIN 12
#define BT_RX_PIN 11

//istanzio oggetto SoftwareSerial (il nostro futuro bluetooth)
SoftwareSerial bt = SoftwareSerial(BT_RX_PIN, BT_TX_PIN);

void setup() {

//definisco modalit\'{a} pin
pinMode(BT_RX_PIN, INPUT);
pinMode(BT_TX_PIN, OUTPUT);

//inizializzo comunicazione Seriale
Serial.begin(9600);

//inizializzo comunicazione Bluetooth
bt.begin(9600);

}

void loop() {
//se ci sono dati sul buffer della Serial
while (Serial.available() > 0) {
//mandali al modulo bluetooth
bt.print(Serial.read());
}

//se ci sono dati sul buffer SoftwareSerial (il buffer del bluetooth)
while (bt.available() > 0) {
//mostrali nel Serial Monitor
Serial.println(bt.read());
}
}
```

Este segundo código ha sido usado en la evaluación final usando Matlab® como medio de envío de los datos. Se nota que ha sido necesaria una conversión de los caracteres desde **Código ASCII** hasta el correspondiente carácter **Unicode**.

```
//Includo libreria SoftwareSerial
#include <SoftwareSerial.h>

//definisco pin RX e TX da Arduino verso modulo BT
#define BT_TX_PIN 12
#define BT_RX_PIN 11
String testo="#";
byte inbyte=0;
//istanzio oggetto SoftwareSerial (il nostro futuro bluetooth)
SoftwareSerial bt = SoftwareSerial(BT_RX_PIN, BT_TX_PIN);

void setup() {

//definisco modalit\'{a} pin
pinMode(BT_RX_PIN, INPUT);
pinMode(BT_TX_PIN, OUTPUT);

//inizializzo comunicazione Seriale
Serial.begin(9600);

//inizializzo comunicazione Bluetooth
bt.begin(9600);

}

void loop() {
//se ci sono dati sul buffer della Serial
while (Serial.available() > 0) {
//mandali al modulo bluetooth
inbyte = Serial.read();
if (inbyte == 35)//'#' IN CODICE ASCII
{
while(inbyte != 126)//finche diverso da "~"
{
inbyte = Serial.read();
//inizio
if (inbyte == 48)//'0' IN CODICE ASCII
{
testo=testo+"0";

}
if (inbyte == 49)//'1' IN CODICE ASCII
{
testo=testo+"1";

}
if (inbyte == 50)//'2' IN CODICE ASCII
{
testo=testo+"2";

}
if (inbyte == 51)//'3' IN CODICE ASCII
{
testo=testo+"3";

}
if (inbyte == 52)//'4' IN CODICE ASCII
{
testo=testo+"4";

}
if (inbyte == 53)//'5' IN CODICE ASCII
{
testo=testo+"5";
```

```
  }
  if (inbyte == 54)//'6' IN CODICE ASCII
  {
  testo=testo+"6";

  }
  if (inbyte == 55)//'7' IN CODICE ASCII
  {
  testo=testo+"7";

  }
  if (inbyte == 56)//'8' IN CODICE ASCII
  {
  testo=testo+"8";

  }
  if (inbyte == 57)//'9' IN CODICE ASCII
  {
  testo=testo+"9";

  }
  if (inbyte == 46)//'.' IN CODICE ASCII
  {
  testo=testo+".";

  }
  if (inbyte == 43)//'+' IN CODICE ASCII
  {
  testo=testo + "+";

  }
  }
  testo=testo+"~";
  bt.print(testo);
  testo="#";
  }
  }

  //se ci sono dati sul buffer SoftwareSerial (il
    buffer del bluetooth)
  while (bt.available() > 0) {
  //mostrali nel Serial Monitor
  Serial.println(bt.read());
  }
  }
```

APPENDIX B
MATLAB CODE

```
%%Uso matlabascii.ino
%creo un Segnale PPG virtuale
%load('Data002.mat');
x=Data002(:,1);
y=Data002(:,2);
xi=linspace(0,10000,10000);
yi=interp1(x,y,xi,'spline');
%plot(xi,yi);
findpeaks(yi,xi,'MinPeakProminence',4,'Annotate','
  extents')
[pks,locs,widths,proms] = findpeaks(yi,xi);
widths(2:3)
%mapping of yi fra 0 e 1023
for i=1:10000
yi(i)=yi(i)/1000;
end
%Apro la comunicazione seriale

arduinoCom = serial('/dev/ttyUSB0');  % insert
  your serial properties here
set(arduinoCom, 'Baudrate', 9600);
set(arduinoCom, 'Parity', 'none');
set(arduinoCom, 'Databits', 8);
set(arduinoCom, 'Stopbit', 1);
set(arduinoCom, 'Terminator', 'CR/LF');
set(arduinoCom, 'OutputBufferSize', 22);
fopen(arduinoCom);%NB: non chiudo la comunicazione
   con un fclose()
%parameti da settare per l'onda PPG
bpm=60;
saturacion=98;
SATUR=saturacion/100*5;%converto in volt
interv=100;
paso=1;
t=1;
Y=0;
i=1;
FREQ=bpm/60;%converto in HZ
figure('name','Pulsossimetro');

while(t<interv)
while (i<10000)
%ATTENZIONE: Invio solo caratteri ASCII
fprintf(arduinoCom,'#%0.2f+%0.2f+%0.2f+%0.2f+~',[
  yi(i),SATUR,0,0]);%

Y=[Y,yi(i)];
plot(Y,':.');
ylim([0 5]);
xlim([0 100]);

grid
t=t+paso;
i=ceil(i+150*FREQ);%150 e' un fattore correttivo
  legato al fitting del segnale PPG
drawnow;
if (i>10000)
i=1;
end
if (t>interv)
t=0;
Y=yi(i);
end
end
end
```

APPENDIX C
JAVA CODE

A. *MainActivity.class*

La MainActivity.class es la que gestiona el Live Pulsoximetro y la comunicación Bluetooth con el dispositivo pareado.

```
package zamataro.arduinoandroidmionew;

import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.Intent;
import android.net.ParseException;
import android.os.Bundle;
import android.os.Handler;
import android.support.v7.app.AppCompatActivity;
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;

import com.jjoe64.graphview.GraphView;
import com.jjoe64.graphview.Viewport;
import com.jjoe64.graphview.series.DataPoint;
import com.jjoe64.graphview.series.LineGraphSeries;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
```

```
import java.util.ArrayList;
import java.util.List;
import java.util.UUID;


public class MainActivity extends AppCompatActivity {

//public int FLAG = 1;
private static final String TAG = "TAG";
// SPP UUID service - this should work for most devices
private static final UUID BTMODULEUUID = UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");
// String for MAC address
public static String address;
final int handlerState = 0;                        //used to identify handler message
public double sensor0 = 0.00;
public double sensor1 = 0.00;
public double sensor2 = 0.00;
public double sensor3 = 0.00;
public List<Double> muestras = new ArrayList<>();
public boolean flag = false;
public double timeOld = System.currentTimeMillis();
public double diffTime;
public int index = 0;
public double muestras_sum = 0;
public double freq_cardiaca = 0;
Button btnOn, btnOff;
TextView txtString, txtStringLength, sensorView0, sensorView1, sensorView2, sensorView3;
Handler bluetoothIn;
private BluetoothAdapter btAdapter = null;
private BluetoothSocket btSocket = null;
private StringBuilder recDataString = new StringBuilder();
//private static final Random RANDOM = new Random();
private LineGraphSeries<DataPoint> series;
private int lastX = 0;
private ConnectedThread mConnectedThread;

@Override
public void onCreate(Bundle savedInstanceState) {
super.onCreate(savedInstanceState);

setContentView(R.layout.activity_main);

//Link the buttons and textViews to respective views
btnOn = (Button) findViewById(R.id.buttonOn);
btnOff = (Button) findViewById(R.id.buttonOff);
txtString = (TextView) findViewById(R.id.txtString);
txtStringLength = (TextView) findViewById(R.id.testView1);
sensorView0 = (TextView) findViewById(R.id.SensorView0);
sensorView1 = (TextView) findViewById(R.id.SensorView1);
sensorView2 = (TextView) findViewById(R.id.SensorView2);
sensorView3 = (TextView) findViewById(R.id.SensorView3);

// we get graph view instance
GraphView graph = (GraphView) findViewById(R.id.graph);
// data
series = new LineGraphSeries<DataPoint>();
graph.addSeries(series);
// customize a little bit viewport
Viewport viewport = graph.getViewport();
viewport.setYAxisBoundsManual(true);
viewport.setXAxisBoundsManual(true);
viewport.setMinY(0);
viewport.setMaxY(5);
viewport.setMinX(0);
viewport.setMaxX(1000);

viewport.setScrollable(true);

bluetoothIn = new Handler() {
public void handleMessage(android.os.Message msg) {
if (msg.what == handlerState) {                    //if message is what we want
String readMessage = (String) msg.obj;                // msg.arg1 = bytes from connect thread
recDataString.append(readMessage);                    //keep appending to string until ~
int endOfLineIndex = recDataString.indexOf("\n");                    // determine the end-of-line
if (endOfLineIndex > 0) {                           // make sure there data before ~
String dataInPrint = recDataString.substring(0, endOfLineIndex);    // extract string
txtString.setText("Data Received = " + dataInPrint);
//mConnectedThread.write("&");    // Send the message via bluetooth
int dataLength = dataInPrint.length();                    //get length of data received
txtStringLength.setText("String Length = " + String.valueOf(dataLength));
if (Float.parseFloat(String.valueOf(dataLength)) < 24) {
if (recDataString.charAt(0) == '#')                    //if it starts with # we know it is what we are looking for
{
String Sensor0 = recDataString.substring(1, 5);             //get sensor value from string between indices 1-5
String Sensor1 = recDataString.substring(6, 10);            //same again...
String Sensor2 = recDataString.substring(11, 15);
String Sensor3 = recDataString.substring(16, 20);
try {
sensor0 = Float.parseFloat(Sensor0);
sensor1 = Float.parseFloat(Sensor1);
sensor2 = Float.parseFloat(Sensor2);
sensor3 = Float.parseFloat(Sensor3);

//Log.d("TAG",""+sensor0);

//detecto discesa
if ((sensor0 < 3.5) && (flag == true)) {
//Sto scendendo
flag = false;
}
if ((sensor0 > 3.5) && (flag == false)) {
//Sto salendo
flag = true;
double timeNew = System.currentTimeMillis();
diffTime = timeNew - timeOld;
//Log.d("TAG","diffTime "+diffTime);

timeOld = timeNew;

muestras.add(index, 60 * 1000 / diffTime);
//Log.d("TAG",muestras.get(index).toString());

index = index + 1;
//ho un picco
if (index > 15) {
for (int i = 1; i < 10; i++) {
muestras_sum = muestras_sum + muestras.get(index - i).doubleValue();
}
freq_cardiaca = muestras_sum / 10;
//Log.d("TAG","sum "+muestras_sum);
muestras_sum = 0;
//Log.d("TAG","freq "+freq_cardiaca);
}
```

```
}

}catch (NumberFormatException e) {
}


sensorView0.setText(" Sensor 0 Voltage = " + Sensor0 + " V");    //update the textviews with sensor values
sensorView1.setText(" Sensor 1 Voltage = " + Sensor1 + " V");
sensorView2.setText(" Freq_Cardiaca = " + ((int) freq_cardiaca) + " BPM");//Se podria poner Sensor3 del arduino
sensorView3.setText(" Temp = " + "36.5" + " \ textdegreeC");// Se podria poner Sensor4 del Arduino

}
}//if float parse < 24


recDataString.delete(0, recDataString.length());                    //clear all string data
// strIncom =" ";
dataInPrint = " ";

}
}
}
};


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

}


private BluetoothSocket createBluetoothSocket(BluetoothDevice device) throws IOException {

return device.createRfcommSocketToServiceRecord(BTMODULEUUID);
//creates secure outgoing connecetion with BT device using UUID
}

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

//Get MAC address from DeviceListActivity via intent
Intent intent = getIntent();

//Get the MAC address from the DeviceListActivty via EXTRA
address = intent.getStringExtra("EXTRA_DEVICE_ADDRESS");
//address = intent.getExtras().toString();
//Toast.makeText(getApplicationContext(), address, Toast.LENGTH_LONG).show();
//create device and set the MAC address
BluetoothDevice device = btAdapter.getRemoteDevice(address.toString());
int a = 1;
//BluetoothDevice device = btAdapter.getRemoteDevice(address);
try {
btSocket = createBluetoothSocket(device);
} catch (IOException e) {
Toast.makeText(getBaseContext(), "Socket creation failed", Toast.LENGTH_LONG).show();
}
// Establish the Bluetooth socket connection.
try {
btSocket.connect();
} catch (IOException e) {
try {
btSocket.close();
} catch (IOException e2) {
//insert code to deal with this
}
}
mConnectedThread = new ConnectedThread(btSocket);
mConnectedThread.start();

// Set up onClick listeners for buttons to send 1 or 0 to turn on/off LED
btnOff.setOnClickListener(new OnClickListener() {
public void onClick(View v) {
mConnectedThread.write("0");    // Send "0" via Bluetooth
Toast.makeText(getBaseContext(), "Turn off LED", Toast.LENGTH_SHORT).show();
}
});

btnOn.setOnClickListener(new OnClickListener() {
public void onClick(View v) {
mConnectedThread.write("1");    // Send "1" via Bluetooth
Toast.makeText(getBaseContext(), "Turn on LED", Toast.LENGTH_SHORT).show();
}
});
//I send a character when resuming.beginning transmission to check device is connected
//If it is not an exception will be thrown in the write method and finish() will be called
mConnectedThread.write("x");


}//onResume

//////////////////////////////////////////////////
// add data to graph
private void addEntry() {
// here, we choose to display max 10 points on the viewport and we scroll to end
series.appendData(new DataPoint(lastX++, sensor0), true, 1000);
}

//////////////////////////////////////////////////

@Override
public void onPause() {
super.onPause();
try {
//Don't leave Bluetooth sockets open when leaving activity
btSocket.close();
} catch (IOException e2) {
//insert code to deal with this
}
}

//Checks that the Android device Bluetooth is available and prompts to be turned on if off
private void checkBTState() {

if (btAdapter == null) {
Toast.makeText(getBaseContext(), "Device does not support bluetooth", Toast.LENGTH_LONG).show();
} else {
if (btAdapter.isEnabled()) {
} else {
Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
startActivityForResult(enableBtIntent, 1);
}
}
```

```
}

//create new class for connect thread
private class ConnectedThread extends Thread {
private final InputStream mmInStream;
private final OutputStream mmOutStream;

//creation of the connect thread
public ConnectedThread(BluetoothSocket socket) {
InputStream tmpIn = null;
OutputStream tmpOut = null;

try {
//Create I/O streams for connection
tmpIn = socket.getInputStream();
tmpOut = socket.getOutputStream();
} catch (IOException e) {
}

mmInStream = tmpIn;
mmOutStream = tmpOut;
}


public void run() {
byte[] buffer = new byte[256];
int bytes;

// Keep looping to listen for received messages
while (true) {
try {
bytes = mmInStream.read(buffer);            //read
    bytes from input buffer
String readMessage = new String(buffer, 0, bytes);
// Send the obtained bytes to the UI Activity via
    handler
bluetoothIn.obtainMessage(handlerState, bytes, -1,
    readMessage).sendToTarget();
} catch (IOException e) {
break;
}
//Log.d("TAG","Sono prma del for");
runOnUiThread(new Runnable() {

@Override
public void run() {
addEntry();
//Log.d("TAG","HO AGGIUNTO UN ENTRATA");

}
});

}
}

//write method
public void write(String input) {
byte[] msgBuffer = input.getBytes();           //
    converts entered String into bytes
try {
mmOutStream.write(msgBuffer);                //write
     bytes over BT connection via outstream
} catch (IOException e) {
//if you cannot write, close the application
Toast.makeText(getBaseContext(), "Connection Failure
    ", Toast.LENGTH_LONG).show();
finish();

}
}
}
}
```

APPENDIX D
LAYOUT

Se muestran ahora los Layouts de cada Activity en las Fig.8, Fig.9 y Fig.10:

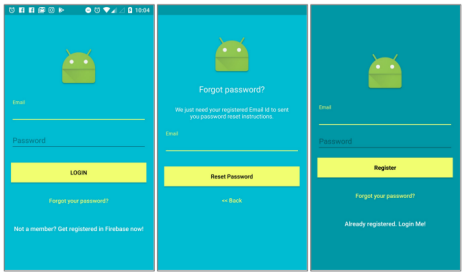

Fig. 9. Login Activity (1), Reset Password Activity (2) y SignUp Activity (3)

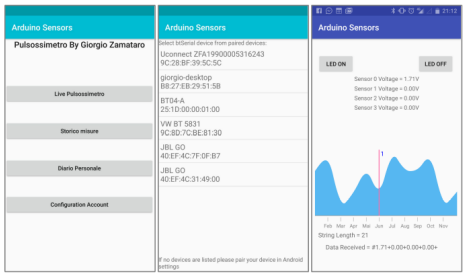

Fig. 10. Menu Activity (1), Device List Activity (2) y Live pulsoximetro Activity (3)

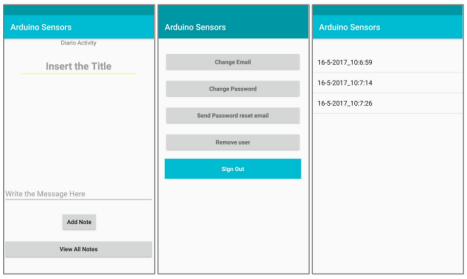

Fig. 11. Personal Note Activity (1), Config Account Activity (2), View Notes Activity (3)

ACKNOWLEDGMENT

El autor quiere agradecer todos los que han suportado este proyecto, la disponibilidad de los profesores y el personal del Laboratorio de Tecnología Electrónica del Universidad de Sevilla para la posibilidad de trabajar en un proyecto concreto como eso. Por cualquier duda sobre este proyecto se puede consultar la Repository en GitHub:
https://github.com/zamathebest/AndroidRepository.git

REFERENCES

[1] L. Moreno et al. *Diseño e Implementación de un Dispositivo para la Detección de la Oxigenación en Sangre* TFG Universidad de Sevilla (2015)

[2] Askie LM et al. *Effects of targeting lower versus higher arterial oxygen saturations on death or disability in preterm infants.* Cochrane Database Syst Rev. 2017 Apr 11;4
[3] Drum ET et al. *Pulse Oximeter: Disruptive Technology or Standard of Care?* A A Case Rep. 2016 Jun 15;6(12)
[4] Green MS et al. *Near-Infrared Spectroscopy: The New Must Have Tool in the Intensive Care Unit?* Semin Cardiothorac Vasc Anesth. 2016 Sep;20(3)
[5] Enoch AJ et al. *Does pulse oximeter use impact health outcomes? A systematic review.* Arch Dis Child. 2016 Aug;101(8)
[6] https://developer.android.com/index.html Ultimo Acceso: 06/17
[7] https://stackoverflow.com/ Ultimo Acceso: 06/17
[8] https://www.hackster.io/arduino Ultimo Acceso: 06/17
[9] https://it.mathworks.com/hardware-support/arduino-matlab.html Ultimo Acceso: 06/17
[10] http://www.lombardoandrea.com/arduino-hc-05-base/ Ultimo Acceso: 06/17
[11] http://www.android-graphview.org/ Ultimo Acceso: 06/17
[12] https://firebase.google.com/ Ultimo Acceso: 06/17