Licenciatura Engenharia Informática 2016/2017

Trabalho Final de Curso

Monitor remoto de Sensores Biomédicos

Luis Manuel Gonçalves Henriques Pêra Lopes

Nº 21106110

Orientador: Prof. Sérgio Ferreira

Índice:

Introdução/Apresentação do Tema

Com o desenvolvimento do tema deste projeto procura-se criar as condições de controle biomédico que estão associadas a deslocações e eventuais quedas de participantes em práticas desportivas ou de pessoas que nas suas deslocações sofram eventuais quedas que possam vir a requerer a intervenção de responsáveis pelo seu bem estar e ainda de pessoa(s)/organizações que queiram desenvolver atividades associadas à prestação de auxilio.

De igual modo e face ao crescente envelhecimento da população em geral e da portuguesa em particular, este universo de potenciais utilizadores requer a criação de condições para o acompanhamento dessas populações por parte dos seus cuidadores ou familiares e que não acarrete um custo significativo face à crescente vulgarização dos telemóveis de 4ª geração.

Nesse sentido pensou-se que a junção de alguns equipamentos pode ser uma ferramenta útil nessa eventual situação de apoio.

Assim à placa Arduino acrescida do módulo de comunicação Bluetooth, cujo nome comercial Bluno foram adicionados a placa Acelerómetro/Giroscópio e o cinto de batimento cardíaco da Medisana. Dos quais se fará uma descrição mais detalhada.

Considera-se que se verificou uma ocorrência de queda sempre que os valores dos indicadores do Acelerómetro/Giroscópio indicarem que se alteraram os valores normais dos parâmetros e se estes parâmetro se mantiverem nesses valores por um período de tempo superior ao estabelecido como limite para situações normais e simultaneamente se verificar uma aceleração do número de batimentos cardíacos, fato associado normalmente a estas situações, a pessoa em causa requer uma intervenção de um cuidador ou responsável.

Não foi possível desenvolver um complemento desta aplicação que poderia enviar uma mensagem para um telemóvel cujo número seria recolhido antecipadamente à situação que se pretende monitorizar.

1. - Desenvolvimento do Tema

1.1 - Condições Prévias

Para o desenvolvimento do tema foi necessário proceder à aquisição de um telemóvel com as condições técnicas requeridas para comunicação e suporte à aplicação a desenvolver, bem como os restantes equipamentos, anteriormente mencionados.
Para a criação da aplicação e para o necessário desenvolvimento desta foi necessário recorrer ao estudo do livro "Android – Desenvolvimento de aplicações com Android Studio., dado tratar-se de um modelo sistematizado.
Para a criação das aplicações e uma vez que já existem algumas já desenvolvidas e disponíveis para consulta e utilização recorreu-se à internet para a recolha dessa informação sobre projetos e aplicações similares e que de alguma forma permitissem um mais rápido desenvolvimento da aplicação a criar. Todas as aplicações utilizadas são "open source".

1.2 – Lista de Componentes

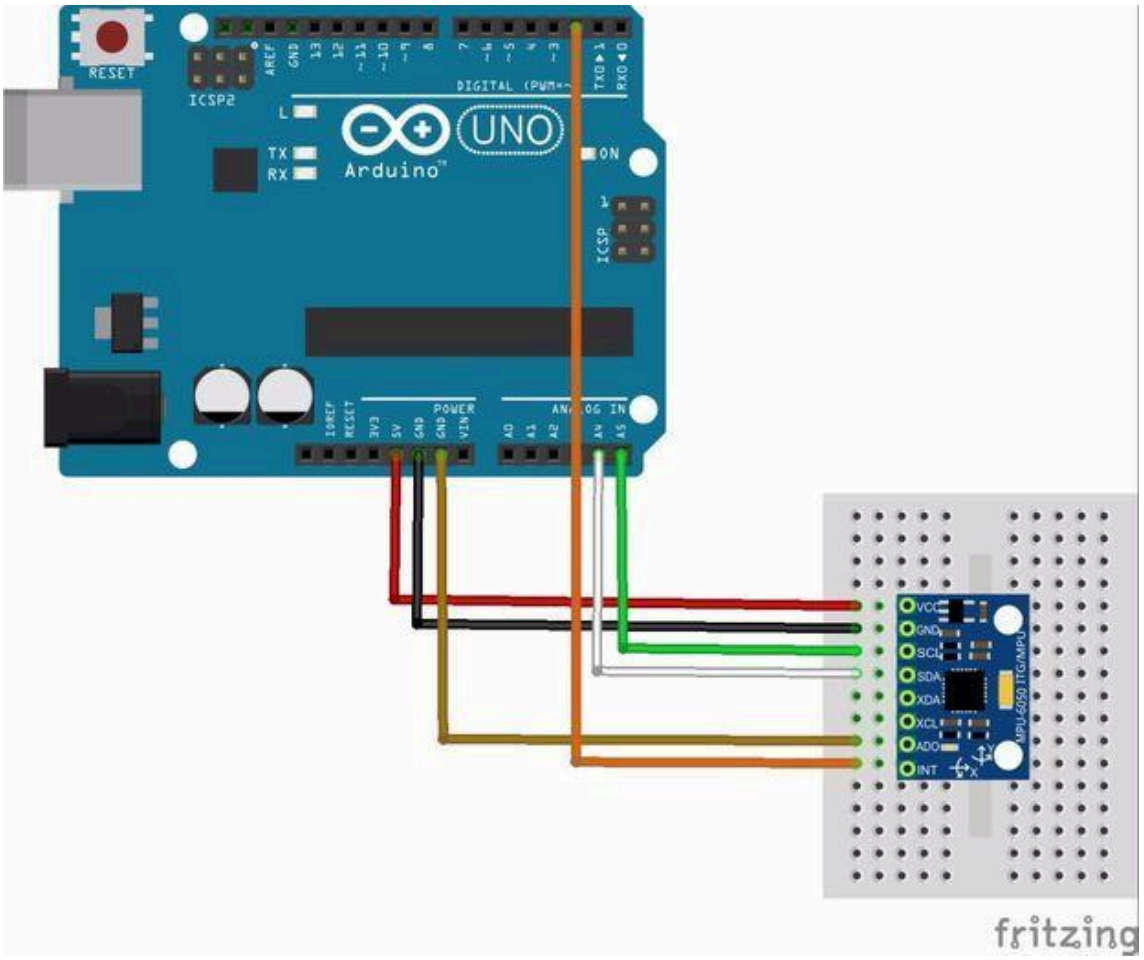
- 1 x Placa Rrduino/Bluno v2.0 da DFRobot+ Cabo USB;
- Componente Acelarometro/Giroscopio MPU6050:
- Breadboard;
- Cabos macho-macho;
- Pinhas AA e respectivo suporte;
- Caixa para montage;
- Cinto de Frequência Cardiaca da Medisana;
- Telemóvel BQ Aquaris U Lite.

No Anexo 4 - Apresento o esquema de montagem integral do protótipo em funcionamento real.

No Anexo 5 – Indico os recursos utilizados.

1.3 – Montagem dos Equipamentos

Com vista a permitir o registo e a comunicação de dados relativos ao giroscópio/acelerómetro procedeu-se à montam dos equipamentos de acordo com os esquemas publicitados para o mesmo.
Este esquema é o que a seguir se apresenta:



1.4 -Ligações Bluno ao Acelerómetro/Giroscópio

1ª UCC ao power 5V;
2ª GND ao power GND;
3ª SCL ao analog A5;
4ª SDA ao analog A4;
5ª INT ao Digital 2.

O esquema esta representado no Anexo 1 é o real testado.

Relativamente ao cinto de batimento cardíaco (Medisana) o mesmo é colocado de acordo com as instruções próprias.

2 – Considerações Gerais sobre o Desenvolvimento das Aplicações

O desenvolvimento das aplicações baseou-se em códigos disponibilizados na internet aos quais foram sucessivamente adicionadas linhas de programação que têm em conta os objetivos pretendidos e que satisfazem assim as necessidades dos potenciais utilizadores.
De igual modo existem vários modelos para a determinação do que se pode considerar validamente uma queda, tendo em conta os valores que se alteram nas posições relativas dos três eixos do giroscópio. Entre ela optou-se pela abaixo indicada que me parece ser a mais adequada, tendo em conta os valores recolhidos neste projeto.

Para a seleção das aplicações comecei por procurar as que integrassem mais do que um dispositivo e que evitassem o recurso a um servidor. Esta seleção revelou-se muito demorada e mostrou-me que teria de ser desenvolvida uma aplicação que conjugasse os dados recolhidos pelos dois dispositivos.
A recolha dos dados na aplicação foi crítica dado que a recolha de dados de um dos dispositivos era terminada com a recolha do segundo.

Tendo em conta a crescente utilização de equipamentos móveis bem como as facilidades de comunicação, considero que se poderá acrescentar um módulo de comunicação, tornando-se assim ainda mais fácil a tarefa de tomar conta de pessoas com necessidades crescentes de cuidados médicos e que, simultaneamente possibilitando uma maior liberdade de movimentos destas faixas da população com um crescente incentivo à prática desportiva associada à liberdade de movimentos e garantindo uma melhor qualidade de vida.

2.1 - Desenvolvimento das Aplicações propriamente dito

O desenvolvimento das aplicações teve em conta os códigos existentes em programas apresentados para projetos adaptáveis ao fim específico do tema do projeto.
Neste sentido foram desenvolvidas as seguintes aplicações:

I.- Aplicação Android

No desenvolvimento do tema foi necessário desenvolver a aplicação Android designada “detetar_quedas_mod, a qual teve em conta o Esquema do Ciclo de Vida, que consta do Anexo 2.
Esta aplicação destina-se a recolher os valores indicados nos dois dispositivos.

II.- Aplicação Ble_Arduino_queda

Esta aplicação visa tratar os dados recolhidos após se ter verificado uma alteração dos valores normais do batimento cardíaco que está associado à queda.

Algoritmo do Detetor de Quedas

Com vista a criar um algoritmo detetor de quedas pensei que se fosse possível determinar um modelo que tivesse em linha de conta as alterações que ocorrem sempre que se verifique a referenciada queda, a mesma acarreta uma alteração significativa do batimento cardíaco, assim como a alteração dos valores dos eixos do giroscópio.

Considero que se verifica uma queda sempre que essa alteração se mantem por um período determinado, o qual pode ter em linha de conta as características do indivíduo em causa.

Quando se verificar que houve uma aceleração repentina, igual ou superior a 3G (768 AM), ativa o estado 1 e se posteriormente ao estado 1 houver uma desaceleração em que a AM regista um valor entre 250 e 260 o que corresponde ao ponto de impacto e respetiva redução no AM isto é considerado o estado 2. Se posteriormente se verificar que a orientação sofreu uma alteração significativa (mudança brusca de orientação) (angleChange >= 1.396 && angleChange <= 1.745) corresponde a identificação de queda.

3 - Código das Aplicações

Os códigos das aplicações desenvolvidas para este tema/projeto são as seguintes:

Código Arduino: Designado por “detetar_quedas_mod”

```
#include <Wire.h>


// I2Cdev and MPU6050 must be installed as libraries, or else the .cpp/.h files
// for both classes must be in the include path of your project
#include "I2Cdev.h"
#include "MPU6050.h"

// Arduino Wire library is required if I2Cdev I2CDEV_ARDUINO_WIRE implementation
// is used in I2Cdev.h
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
  #include "Wire.h"
#endif

// class default I2C address is 0x68
// specific I2C addresses may be passed as a parameter here
// AD0 low = 0x68 (default for InvenSense evaluation board)
// AD0 high = 0x69
MPU6050 accelgyro;
//MPU6050 accelgyro(0x69); // <-- use for AD0 high


// variaveis para guardar dados e controlo de algoritmo
int16_t ax, ay, az;
int16_t gx, gy, gz;

long acc_total_vector;

boolean fall    = false;  //stores if a fall has occurred
boolean trigger1 = false;  //stores if first trigger (lower threshold) has occurred
boolean trigger2 = false;  //stores if second trigger (upper threshold)  has occurred
byte trigger1count = 0;  //stores the counts past since trigger 1 was set true

//constantes usadas nos calculos
const int bx = 0;
const int by = 0;
const int bz = 256;

// var para enviar dados
int contagem = 0;

//store reference upright acceleration
int BM = pow(pow(bx,2)+pow(by,2)+pow(bz,2),0.5);

// uncomment "OUTPUT_READABLE_ACCELGYRO" if you want to see a tab-separated
```

```
// list of the accel X/Y/Z and then gyro X/Y/Z values in decimal. Easy to read,
// not so easy to parse, and slow(er) over UART.
// #define OUTPUT_READABLE_ACCELGYRO

// uncomment "OUTPUT_BINARY_ACCELGYRO" to send all 6 axes of data as 16-bit
// binary, one right after the other. This is very fast (as fast as possible
// without compression or data loss), and easy to parse, but impossible to read
// for a human.
//#define OUTPUT_BINARY_ACCELGYRO

#define LED_PIN 13
bool blinkState = false;

void setup()
{
  // join I2C bus (I2Cdev library doesn't do this automatically)
   #if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
      Wire.begin();
   #elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
      Fastwire::setup(400, true);
   #endif

   // initialize serial communication
   // (38400 chosen because it works as well at 8MHz as it does at 16MHz, but
   // it's really up to you depending on your project)
   Serial.begin(38400);

   // initialize device
   Serial.println("Initializing I2C devices...");
   accelgyro.initialize();

   // verify connection
   Serial.println("Testing device connections...");
    Serial.println(accelgyro.testConnection() ? "MPU6050 connection successful" : "MPU6050
connection failed");

   // use the code below to change accel/gyro offset values

   Serial.println("Updating internal sensor offsets...");
   // -76  -2359 1688  0 0 0
   Serial.print(accelgyro.getXAccelOffset()); Serial.print("\t"); // -76
   Serial.print(accelgyro.getYAccelOffset()); Serial.print("\t"); // -2359
   Serial.print(accelgyro.getZAccelOffset()); Serial.print("\t"); // 1688
   Serial.print(accelgyro.getXGyroOffset()); Serial.print("\t"); // 0
   Serial.print(accelgyro.getYGyroOffset()); Serial.print("\t"); // 0
   Serial.print(accelgyro.getZGyroOffset()); Serial.print("\t"); // 0
   Serial.print("\n");
   accelgyro.setXGyroOffset(220);
   accelgyro.setYGyroOffset(76);
   accelgyro.setZGyroOffset(-85);
   Serial.print(accelgyro.getXAccelOffset()); Serial.print("\t"); // -76
   Serial.print(accelgyro.getYAccelOffset()); Serial.print("\t"); // -2359
```

```
    Serial.print(accelgyro.getZAccelOffset()); Serial.print("\t"); // 1688
    Serial.print(accelgyro.getXGyroOffset()); Serial.print("\t"); // 0
    Serial.print(accelgyro.getYGyroOffset()); Serial.print("\t"); // 0
    Serial.print(accelgyro.getZGyroOffset()); Serial.print("\t"); // 0
    Serial.print("\n");


    // configure Arduino LED for
    pinMode(LED_PIN, OUTPUT);
}

void loop()
{
    // read raw accel/gyro measurements from device
    accelgyro.getMotion6(&ax, &ay, &az, &gx, &gy, &gz);

    // these methods (and a few others) are also available
    //accelgyro.getAcceleration(&ax, &ay, &az);
    //accelgyro.getRotation(&gx, &gy, &gz);

    #ifdef OUTPUT_READABLE_ACCELGYRO
        // display tab-separated accel/gyro x/y/z values
        Serial.print("a/g:\t");
        Serial.print(ax); Serial.print("\t");
        Serial.print(ay); Serial.print("\t");
        Serial.print(az); Serial.print("\t");
        Serial.print(gx); Serial.print("\t");
        Serial.print(gy); Serial.print("\t");
        Serial.println(gz);
    #endif

    #ifdef OUTPUT_BINARY_ACCELGYRO
        Serial.write((uint8_t)(ax >> 8)); Serial.write((uint8_t)(ax & 0xFF));
        Serial.write((uint8_t)(ay >> 8)); Serial.write((uint8_t)(ay & 0xFF));
        Serial.write((uint8_t)(az >> 8)); Serial.write((uint8_t)(az & 0xFF));
        Serial.write((uint8_t)(gx >> 8)); Serial.write((uint8_t)(gx & 0xFF));
        Serial.write((uint8_t)(gy >> 8)); Serial.write((uint8_t)(gy & 0xFF));
        Serial.write((uint8_t)(gz >> 8)); Serial.write((uint8_t)(gz & 0xFF));
    #endif

    // blink LED to indicate activity
    blinkState = !blinkState;

    digitalWrite(LED_PIN, blinkState);

    contagem++;

    //////// inicio de codigo para detectar quedas ////////////

    int AM; double angleChange=0;
```

```
  // calculo mudanca de angulo
  AM = pow(pow(ax,2)+pow(ay,2)+pow(az,2),0.5);

  if((contagem == 1) or  (contagem == 10)) {
   String varMostrar0 = "AM:";
   String varMostrar1 = varMostrar0 + AM;
   Serial.print(varMostrar1);
  }


  //vector acelaracao total
  acc_total_vector = sqrt((ax*ax)+(ay*ay)+(az*az));  //Calculate the total accelerometer vector
  if((contagem == 1) or  (contagem == 10)) {
   String varMostrar2 = "Vector Acelaracao:";
   String varMostrar3 = varMostrar2 + acc_total_vector;
   Serial.print(varMostrar3);
  }

  if (trigger2 == true){
                                                                    angleChange                =
acos(((double)ax*(double)bx+(double)ay*(double)by+(double)az*(double)bz)/(double)AM/(do
uble)BM);
    if((contagem == 1) or  (contagem == 10)) {
     String varMostrar4 = "AngleChange:";
     String varMostrar5 = varMostrar4 + angleChange;
     Serial.print(varMostrar5);
    }

      if (angleChange >= 1.396 && angleChange <= 1.745){  //if orientation change is between
80-100 degrees
     fall    = true;
     trigger2 = false;
    }
    else{  //user regained normal orientation
     trigger2 = false;
     Serial.print("TRIGGER 2 DEACTIVATED");
    }
  }
  if (fall == true){  //in event of a fall detection
   Serial.print("ALERTA_QUEDA");
   delay(5000);
   Serial.print("ALERTA_QUEDA");
   delay(5000);
   Serial.print("ALERTA_QUEDA");
   delay(5000);
   Serial.print("ALERTA_QUEDA");
   delay(5000);
   Serial.print("ALERTA_QUEDA");
   fall = false;
   exit(1);
  }
  if (trigger1count >= 201){  //allow 20s for AM to return to relatively normal range
```

```
    trigger1     = false;
    trigger1count = 0;
    Serial.print("TRIGGER 1 DECACTIVATED");
   }
   if (trigger1 == true){
    trigger1count++;

        if (AM >= 250 && AM <= 260){  //if AM has returned to a relatively normal range
(0.9768g-1.0159g)
      trigger2 = true;
      trigger1 = false;
      trigger1count = 0;
      Serial.print("TRIGGER 2 ACTIVATED");
    }
   }
   if (AM >= 768){  //if AM breaks upper threshold (3g)
    trigger1 = true;
    //Serial.print("TRIGGER 1 ACTIVATED");
   }

   if(contagem == 10) {
      contagem = 1;
   }
   //It appears that delay is needed in order not to clog the port
   delay(1000);
}
```

Código Aplicação Android “Gestor.java”.

```
package com.luislopes.ble_arduino_queda;

import android.app.Activity;
import android.app.AlertDialog;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothGatt;
import android.bluetooth.BluetoothGattCallback;
import android.bluetooth.BluetoothGattCharacteristic;
import android.bluetooth.BluetoothGattDescriptor;
import android.bluetooth.BluetoothGattService;
import android.bluetooth.BluetoothManager;
import android.bluetooth.BluetoothProfile;
import android.bluetooth.le.BluetoothLeScanner;
import android.bluetooth.le.ScanCallback;
import android.bluetooth.le.ScanFilter;
import android.bluetooth.le.ScanResult;
import android.bluetooth.le.ScanSettings;
import android.content.BroadcastReceiver;
import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.ServiceConnection;
import android.content.pm.PackageManager;
import android.os.Handler;
import android.os.IBinder;
```

```
import android.util.Log;
import android.widget.Toast;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.UUID;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public abstract class Gestor extends Activity {


    private Context mainContext = this;

    private final String TAG = this.getClass().getSimpleName();

    private BluetoothAdapter mBluetoothAdapter;
    private Handler mHandler;

    private static final int REQUEST_ENABLE_BT = 1;

    // fazer scan de 10s
    private static final long SCAN_PERIOD = 10000;

    private ArrayList<BluetoothDevice> deviceList = new ArrayList<>();


    public abstract void onConectionStateChange(connectionStateEnum
theConnectionState);

    public enum connectionStateEnum {isNull, isScanning, isToScan,
isConnecting, isConnected, isDisconnecting}

    ;

    public connectionStateEnum mConnectionState =
connectionStateEnum.isNull;

    public final static UUID UUID_HEART_RATE_MEASUREMENT =
UUID.fromString(SampleGattAttributes.HEART_RATE_MEASUREMENT);


    // Novo metodo
    private BluetoothLeScanner mLEScanner;
    private ScanSettings settings;
    private List<ScanFilter> filters;
    private BluetoothGatt mGatt;
    private BluetoothGatt mGatt1;

    private Runnable mConnectingOverTimeRunnable = new Runnable() {

        @Override
        public void run() {
            if (mConnectionState == connectionStateEnum.isConnecting)
                mConnectionState = connectionStateEnum.isToScan;
            onConectionStateChange(mConnectionState);
        }
    };

    private Runnable mDisonnectingOverTimeRunnable = new Runnable() {
```

```
        @Override
        public void run() {
            if (mConnectionState ==
connectionStateEnum.isDisconnecting)
                mConnectionState = connectionStateEnum.isToScan;
            onConectionStateChange(mConnectionState);
        }
    };

    private String displayHeartRate;
    private Runnable displayHeart = new Runnable() {

        @Override
        public void run() {
            onBeatReceived(displayHeartRate);
        }
    };

    private String displayBlunoData;
    private Runnable displayBluno = new Runnable() {

        @Override
        public void run() {

            onSerialReceived(displayBlunoData);
        }
    };

    private Runnable displayQuedaAlerta = new Runnable() {
        @Override
        public void run() {
            new
AlertDialog.Builder(mainContext).setMessage(R.string.alerta).setNeutralB
utton("Fechar", null).show();
        }
    };


    //
    public abstract void onSerialReceived(String theString);

    //
    public abstract void onBeatReceived(String theString);

    //
    private int mBaudrate = 38400;    //set the default baud rate to
115200
    private String mPassword = "AT+PASSWOR=DFRobot\r\n";


    private String mBaudrateBuffer = "AT+CURRUART=" + mBaudrate +
"\r\n";

    // iniciar ligacao serial
    public void serialBegin(int baud) {
        mBaudrate = baud;
        mBaudrateBuffer = "AT+CURRUART=" + mBaudrate + "\r\n";
    }
```

```
    public void onCreateProcess() {

        mHandler = new Handler();

        // Use this check to determine whether BLE is supported on the
device.  Then you can
        // selectively disable BLE-related features.
        if
(!getPackageManager().hasSystemFeature(PackageManager.FEATURE_BLUETOOTH_
LE)) {
            Toast.makeText(this, R.string.ble_not_supported,
Toast.LENGTH_SHORT).show();
            finish();
        }

        // Initialize a Bluetooth adapter.  For API level 18 and above,
get a reference to
        // BluetoothAdapter through BluetoothManager.
        final BluetoothManager bluetoothManager =
                (BluetoothManager)
getSystemService(Context.BLUETOOTH_SERVICE);
        mBluetoothAdapter = bluetoothManager.getAdapter();

        // Checks if Bluetooth is supported on the device.
        if (mBluetoothAdapter == null) {
            Toast.makeText(this,
R.string.error_bluetooth_not_supported, Toast.LENGTH_SHORT).show();
            finish();
        }
    }

    protected void iniciarProcesso() {
        switch (mConnectionState) {
            case isNull:
                mConnectionState = connectionStateEnum.isScanning;
                onConectionStateChange(mConnectionState);
                System.out.println("A iniciar scanLeDEvice ...");
                scanLeDevice(true);
                System.out.println("scanLeDEvice Terminou");
                break;
            case isToScan:
                mConnectionState = connectionStateEnum.isScanning;
                onConectionStateChange(mConnectionState);
                //scanLeDevice(true);
                 break;
            case isScanning:

                break;

            case isConnecting:

                break;
            case isConnected:

                mHandler.postDelayed(mDisonnectingOverTimeRunnable,
10000);
                mConnectionState = connectionStateEnum.isDisconnecting;
                onConectionStateChange(mConnectionState);
                break;
            case isDisconnecting:
                break;
```

```
            default:
                break;
        }
    }


    protected void onResumeProcess() {
        Log.e(TAG, "Funcao onResume");
        if (mBluetoothAdapter == null ||
!mBluetoothAdapter.isEnabled()) {
            Intent enableBtIntent = new
Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
            startActivityForResult(enableBtIntent, REQUEST_ENABLE_BT);
        } else {
            mLEScanner = mBluetoothAdapter.getBluetoothLeScanner();
            settings = new
ScanSettings.Builder().setScanMode(ScanSettings.SCAN_MODE_LOW_LATENCY).b
uild();
            filters = new ArrayList<ScanFilter>();
        }
    }

    public void onActivityResultProcess(int requestCode, int resultCode,
Intent data) {
        // User chose not to enable Bluetooth.
        if (requestCode == REQUEST_ENABLE_BT
                && resultCode == Activity.RESULT_CANCELED) {
            ((Activity) mainContext).finish();
            return;
        }
    }

    protected void onPauseProcess() {
        System.out.println("Funcao onPause");
        scanLeDevice(false);

        mConnectionState = connectionStateEnum.isToScan;
        onConectionStateChange(mConnectionState);

        mHandler.postDelayed(mDisonnectingOverTimeRunnable, 10000);

    }

    public void onStopProcess() {
        System.out.println("Funcao onStop");
        //if (mBluetoothLeService != null) {
        //    mHandler.removeCallbacks(mDisonnectingOverTimeRunnable);

        //}
    }

    public void onDestroyProcess() {

    }


    private void scanLeDevice(final boolean enable) {
        System.out.println("inside scanLeDevice ... ");
        if (enable) {
            System.out.println("scanLeDevice true");
            mHandler.postDelayed(new Runnable() {
```

```
                @Override
                public void run() {
                    System.out.println("A encerrar scan...");
                    mLEScanner.stopScan(mScanCallback);
                }
            }, SCAN_PERIOD);
            System.out.println("A iniciar callback ... ");
            mLEScanner.startScan(filters, settings, mScanCallback);

        } else {
            System.out.println("A encerrar scan...");
            mLEScanner.stopScan(mScanCallback);
        }
    }

    private ScanCallback mScanCallback = new ScanCallback() {
        @Override
        public void onScanResult(int callbackType, ScanResult result) {

            Log.i("Resultado pesquisa:", result.toString());
            BluetoothDevice btDevice = result.getDevice();
            if (deviceList.size() == 0) {
                deviceList.add(btDevice);
            } else if (deviceList.size() == 1) {
                if
(!deviceList.get(0).getAddress().equals(btDevice.getAddress())) {
                    System.out.println("Dev0:" +
deviceList.get(0).getAddress() + " Dev1:" + btDevice.getAddress());
                    deviceList.add(btDevice);
                    System.out.println("Iniciar dispositivos");
                    iniciarDispositivos();
                }
            }

            Log.e(TAG, "deviceList count = " + deviceList.size());
        }

        @Override
        public void onBatchScanResults(List<ScanResult> results) {
            for (ScanResult sr : results) {
                Log.i("ScanResult - Results", sr.toString());
            }
        }

        @Override
        public void onScanFailed(int errorCode) {
            Log.e("Scan Failed", "Error Code: " + errorCode);
        }
    };

    private void iniciarDispositivos() {


        mGatt = deviceList.get(0).connectGatt(this, false,
gattCallback);
        Log.i(TAG, "Ligado com sucesso ao dispositivo0: " +
deviceList.get(0).getAddress());
        // mConnectionState = connectionStateEnum.isConnected;
        //onConectionStateChange(mConnectionState);
        //mHandler.postDelayed(mConnectingOverTimeRunnable, 10000);
        try {
```

```
            Thread.sleep(2500);
        } catch (InterruptedException e) {
        }

        mGatt1 = deviceList.get(1).connectGatt(this, false,
gattCallback);
        Log.i(TAG, "Ligado com sucesso ao dispositivo1: " +
deviceList.get(1).getAddress());
        //mHandler.postDelayed(mConnectingOverTimeRunnable, 1000);
        //mConnectionState = connectionStateEnum.isConnected;
        //onConectionStateChange(mConnectionState);
        try {
            Thread.sleep(2500);
        } catch (InterruptedException e) {
        }
    }

    private final BluetoothGattCallback gattCallback = new
BluetoothGattCallback() {

        @Override
        public void onConnectionStateChange(BluetoothGatt gatt, int
status, int newState) {
            Log.i("onConnectionStateChange", "Status: " + status);
            switch (newState) {
                case BluetoothProfile.STATE_CONNECTED:
                    String deviceName = gatt.getDevice().getName();
                    Log.i("gattCallback", "STATE_CONNECTED:" +
deviceName);
                    //try {
                     //catch (InterruptedException e) {}
                     gatt.discoverServices();
                    try {
                        Thread.sleep(2500);
                    } catch (InterruptedException e) {
                    }

                    break;
                case BluetoothProfile.STATE_DISCONNECTED:
                    String deviceName1 = gatt.getDevice().getName();
                    Log.i("gattCallback", "STATE_DISCONNECTED:" +
deviceName1);
                    break;
                default:
                    Log.e("gattCallback", "STATE_OTHER");
            }
        }

        @Override
        public void onServicesDiscovered(BluetoothGatt gatt, int status)
{
            mGatt = gatt;
            ligarServicos(gatt.getServices());
        }

        @Override
        public void onCharacteristicRead(BluetoothGatt gatt,
BluetoothGattCharacteristic characteristic, int status) {
            byte[] value = characteristic.getValue();
            String v = new String(value);
            Log.i("onCharacteristicRead", "Value: " + v);
```

```
        }

        @Override
        public void onCharacteristicChanged(BluetoothGatt gatt,
BluetoothGattCharacteristic characteristic) {

            String deviceName = gatt.getDevice().getName();
            if (deviceName.equals("Bluno")) {
                //String v = characteristic.getStringValue(0);
                byte[] bytes = characteristic.getValue();
                String s = new String(bytes);

                displayBlunoData = s;
                mHandler.post(displayBluno);
                Log.i("onCharacteristicChanged", "CHAR CHANGE FOR
DEVIDE:" + deviceName + ":::Data:" + displayBlunoData);

                Pattern p = Pattern.compile("(.*)QUEDA(.*)");
                Matcher m = p.matcher(displayBlunoData);
                if (m.find()) {
                    System.out.println("ALERTA QUEDA DETETADA ");
                    mHandler.post(displayQuedaAlerta);
                }

            } else if (deviceName.equals("Heart Rate Sensor")) {

                int flag = characteristic.getProperties();
                int format = -1;
                if ((flag & 0x01) != 0) {
                    format = BluetoothGattCharacteristic.FORMAT_UINT16;
                } else {
                    format = BluetoothGattCharacteristic.FORMAT_UINT8;
                }
                final int heartRate = characteristic.getIntValue(format,
1);
                displayHeartRate = String.valueOf(heartRate);
                mHandler.post(displayHeart);
                Log.i("onCharacteristicChanged", "ALTERACAO CHAR PARA
DISPOSITIVO:" + deviceName + ":::Dados:" + displayHeartRate);

            }
        }
    };


    private void ligarServicos(List<BluetoothGattService> gattServices)
{

        if (gattServices == null) return;

        String characteristic_uuid = null;


        // Loops through available GATT Services.
        for (BluetoothGattService gattService : gattServices) {

            List<BluetoothGattCharacteristic> gattCharacteristics =
gattService.getCharacteristics();

            // Loops through available Characteristics.
            for (BluetoothGattCharacteristic gattCharacteristic :
```

```
gattCharacteristics) {
                characteristic_uuid =
gattCharacteristic.getUuid().toString();

//System.out.println("Characteristic:"+characteristic_uuid);

                if
(characteristic_uuid.equals(SampleGattAttributes.SerialPortCharUUID)) {

                    System.out.println("A ligar a caracteristica
SerialPortCharUUID");

mGatt.setCharacteristicNotification(gattCharacteristic, true);
                    try {
                        Thread.sleep(2500);
                    } catch (InterruptedException e) {
                    }
                }
                if
(characteristic_uuid.equals(SampleGattAttributes.HeartRateCharUUID)) {
                    System.out.println("A ligar a caracteristica
HeartRateCharUUID:");
                    // This is specific to Heart Rate Measurement.
                     if
(UUID_HEART_RATE_MEASUREMENT.equals(gattCharacteristic.getUuid())) {
                        BluetoothGattDescriptor descriptor =
gattCharacteristic.getDescriptor(

UUID.fromString(SampleGattAttributes.CLIENT_CHARACTERISTIC_CONFIG));

descriptor.setValue(BluetoothGattDescriptor.ENABLE_NOTIFICATION_VALUE);
                        mGatt.writeDescriptor(descriptor);
                        System.out.println("A iniciar writeDescriptor
HeartRateCharUUID:");

                    }
                    try {
                        Thread.sleep(2500);
                    } catch (InterruptedException e) {
                    }



mGatt.setCharacteristicNotification(gattCharacteristic, true);
                    try {
                        Thread.sleep(2500);
                    } catch (InterruptedException e) {
                    }

                    if (mGatt.readCharacteristic(gattCharacteristic)) {
                        System.out.println("A ler a caracteristica
HeartRateCharUUID:");

                    }

                }
            }
        }
    }
}
```

4 – Registo da Actividade do Código

Aplicação Android:

02/27 21:24:53: Launching app
Cold swapped changes.
$ adb shell am start -n "com.luislopes.ble_arduino_queda/com.luislopes.ble_arduino_queda.MainActivity" -a android.intent.action.MAIN -c android.intent.category.LAUNCHER
Client not ready yet..Waiting for process to come online
Connected to process 27793 on device bq-aquaris_u_lite-UH036788
I/art: Late-enabling -Xcheck:jni
W/System: ClassLoader referenced unknown path: /data/app/com.luislopes.ble_arduino_queda-2/lib/arm
I/InstantRun: Instant Run Runtime started. Android package is com.luislopes.ble_arduino_queda, real application class is null.
W/System: ClassLoader referenced unknown path: /data/app/com.luislopes.ble_arduino_queda-2/lib/arm
I/System.out: Funcao onResume
E/MainActivity: Funcao onResume
D/OpenGLRenderer: Use EGL_SWAP_BEHAVIOR_PRESERVED: true
I/Adreno-EGL: <qeglDrvAPI_eglInitialize:379>: EGL 1.4 QUALCOMM build: (Ic7d72507eb)
OpenGL ES Shader Compiler Version: XE031.08.00.00
Build Date: 09/15/16 Thu
Local Branch:
Remote Branch:
Local Patches:
Reconstruct Branch:
I/OpenGLRenderer: Initialized EGL, version 1.4
I/System.out: A iniciar processo
I/System.out: A iniciar scanLeDEvice ...
I/System.out: inside scanLeDevice ...
I/System.out: scanLeDevice true
I/System.out: A iniciar callback ...
D/BluetoothAdapter: STATE_ON
D/BluetoothLeScanner: onClientRegistered() - status=0 clientIf=5
I/System.out: scanLeDEvice Terminou
I/Resultado pesquisa:: ScanResult{mDevice=20:CD:39:87:DE:4E, mScanRecord=ScanRecord [mAdvertiseFlags=5, mServiceUuids=[0000dfb0-0000-1000-8000-00805f9b34fb, 00001812-0000-1000-8000-00805f9b34fb], mManufacturerSpecificData={76=[2, 21, -30, -59, 109, -75, -33, -5, 72, -46, -80, 96, -48, -11, -89, 16, -106, -32, 0, 0, 0, 0, -59]}, mServiceData={}, mTxPowerLevel=0, mDeviceName=Bluno[illegible], mRssi=-69, mTimestampNanos=1821976931133981}
E/MainActivity: deviceList count = 1

I/Resultado pesquisa:: ScanResult{mDevice=54:4A:16:5B:31:88, mScanRecord=ScanRecord [mAdvertiseFlags=6, mServiceUuids=[0000180d-0000-1000-8000-00805f9b34fb, 0000180f-0000-1000-8000-00805f9b34fb], mManufacturerSpecificData={}, mServiceData={}, mTxPowerLevel=-2147483648, mDeviceName=Heart Rate Sensor], mRssi=-77, mTimestampNanos=1821978637251636}
I/System.out: Dev0:20:CD:39:87:DE:4E Dev1:54:4A:16:5B:31:88
I/System.out: Iniciar dispositivos
D/BluetoothGatt: connect() - device: 20:CD:39:87:DE:4E, auto: false
D/BluetoothGatt: registerApp()
D/BluetoothGatt: registerApp() - UUID=4d830f1a-af1d-4baf-b730-aa4179282506
I/MainActivity: Ligado com sucesso ao dispositivo0: 20:CD:39:87:DE:4E
D/BluetoothGatt: onClientRegistered() - status=0 clientIf=6
D/BluetoothGatt: onClientConnectionState() - status=0 clientIf=6 device=20:CD:39:87:DE:4E
I/onConnectionStateChange: Status: 0
I/gattCallback: STATE_CONNECTED:Bluno
D/BluetoothGatt: discoverServices() - device: 20:CD:39:87:DE:4E
D/BluetoothGatt: connect() - device: 54:4A:16:5B:31:88, auto: false
D/BluetoothGatt: registerApp()
D/BluetoothGatt: registerApp() - UUID=6657d51a-661d-4a7d-8ac2-81951a2e1b72
D/BluetoothGatt: onClientRegistered() - status=0 clientIf=7
I/MainActivity: Ligado com sucesso ao dispositivo1: 54:4A:16:5B:31:88
D/BluetoothGatt: onSearchComplete() = Device=20:CD:39:87:DE:4E Status=0
I/System.out: A ligar a caracteristica SerialPortCharUUID
D/BluetoothGatt: setCharacteristicNotification() - uuid: 0000dfb1-0000-1000-8000-00805f9b34fb enable: true
D/BluetoothGatt: onClientConnectionState() - status=0 clientIf=7 device=54:4A:16:5B:31:88
I/onConnectionStateChange: Status: 0
I/gattCallback: STATE_CONNECTED:Heart Rate Sensor
D/BluetoothGatt: discoverServices() - device: 54:4A:16:5B:31:88
E/MainActivity: deviceList count = 2
I/Resultado pesquisa:: ScanResult{mDevice=54:4A:16:5B:31:88, mScanRecord=ScanRecord [mAdvertiseFlags=6, mServiceUuids=[0000180d-0000-1000-8000-00805f9b34fb, 0000180f-0000-1000-8000-00805f9b34fb], mManufacturerSpecificData={}, mServiceData={}, mTxPowerLevel=-2147483648, mDeviceName=Heart Rate Sensor], mRssi=-80, mTimestampNanos=1821979630126792}
E/MainActivity: deviceList count = 2
D/BluetoothGatt: onSearchComplete() = Device=54:4A:16:5B:31:88 Status=0
I/System.out: A ligar a caracteristica HeartRateCharUUID:
I/System.out: A iniciar writeDescriptor HeartRateCharUUID:
I/System.out: A encerrar scan...
D/BluetoothAdapter: STATE_ON
I/onCharacteristicChanged: CHAR CHANGE FOR DEVIDE:Bluno:::Data:AM:14151Vector Acela
I/onCharacteristicChanged: CHAR CHANGE FOR DEVIDE:Bluno:::Data:racao:-2147483648ALE
I/onCharacteristicChanged: CHAR CHANGE FOR DEVIDE:Bluno:::Data:RTA_QUEDA
I/System.out: ALERTA QUEDA DETETADA
V/BoostFramework: mAcquireFunc method = public int com.qualcomm.qti.Performance.perfLockAcquire(int,int[])
V/BoostFramework: mReleaseFunc method = public int com.qualcomm.qti.Performance.perfLockRelease()
V/BoostFramework: mAcquireTouchFunc method = public int com.qualcomm.qti.Performance.perfLockAcquireTouch(android.view.MotionEvent,android.util.DisplayMetrics,int,int[])

```
V/BoostFramework: mIOPStart method = public int
com.qualcomm.qti.Performance.perfIOPrefetchStart(int,java.lang.String)
V/BoostFramework: mIOPStop method = public int
com.qualcomm.qti.Performance.perfIOPrefetchStop()
V/BoostFramework: BoostFramework() : mPerf = com.qualcomm.qti.Performance@ba99cd4
V/BoostFramework: BoostFramework() : mPerf = com.qualcomm.qti.Performance@4f88e7d
D/BluetoothGatt: setCharacteristicNotification() - uuid:
00002a37-0000-1000-8000-00805f9b34fb enable: true
I/onCharacteristicChanged: ALTERACAO CHAR PARA DISPOSITIVO:Heart Rate Sensor:::Dados:73
I/onCharacteristicChanged: ALTERACAO CHAR PARA DISPOSITIVO:Heart Rate Sensor:::Dados:73
I/onCharacteristicChanged: ALTERACAO CHAR PARA DISPOSITIVO:Heart Rate Sensor:::Dados:73
I/onCharacteristicChanged: ALTERACAO CHAR PARA DISPOSITIVO:Heart Rate Sensor:::Dados:73
I/onCharacteristicChanged: ALTERACAO CHAR PARA DISPOSITIVO:Heart Rate Sensor:::Dados:73
I/onCharacteristicChanged: ALTERACAO CHAR PARA DISPOSITIVO:Heart Rate Sensor:::Dados:73
I/onCharacteristicChanged: ALTERACAO CHAR PARA DISPOSITIVO:Heart Rate Sensor:::Dados:73
I/onCharacteristicChanged: CHAR CHANGE FOR DEVIDE:Bluno:::Data:AM:14092Vector Acela
I/onCharacteristicChanged: CHAR CHANGE FOR DEVIDE:Bluno:::Data:racao:114ALERTA_QUED
I/onCharacteristicChanged: CHAR CHANGE FOR DEVIDE:Bluno:::Data:A
I/onCharacteristicChanged: ALTERACAO CHAR PARA DISPOSITIVO:Heart Rate Sensor:::Dados:73
I/onCharacteristicChanged: ALTERACAO CHAR PARA DISPOSITIVO:Heart Rate Sensor:::Dados:73
I/onCharacteristicChanged: ALTERACAO CHAR PARA DISPOSITIVO:Heart Rate Sensor:::Dados:73
I/onCharacteristicChanged: ALTERACAO CHAR PARA DISPOSITIVO:Heart Rate Sensor:::Dados:73
I/onCharacteristicChanged: ALTERACAO CHAR PARA DISPOSITIVO:Heart Rate Sensor:::Dados:73
I/onCharacteristicChanged: ALTERACAO CHAR PARA DISPOSITIVO:Heart Rate Sensor:::Dados:72
I/onCharacteristicChanged: ALTERACAO CHAR PARA DISPOSITIVO:Heart Rate Sensor:::Dados:72
I/onCharacteristicChanged: ALTERACAO CHAR PARA DISPOSITIVO:Heart Rate Sensor:::Dados:72
I/onCharacteristicChanged: ALTERACAO CHAR PARA DISPOSITIVO:Heart Rate Sensor:::Dados:72
I/onCharacteristicChanged: ALTERACAO CHAR PARA DISPOSITIVO:Heart Rate Sensor:::Dados:72
I/onCharacteristicChanged: ALTERACAO CHAR PARA DISPOSITIVO:Heart Rate Sensor:::Dados:72
I/onCharacteristicChanged: CHAR CHANGE FOR DEVIDE:Bluno:::Data:AM:14217Vector Acela
I/onCharacteristicChanged: CHAR CHANGE FOR DEVIDE:Bluno:::Data:racao:121ALERTA_QUED
I/onCharacteristicChanged: CHAR CHANGE FOR DEVIDE:Bluno:::Data:A
I/onCharacteristicChanged: ALTERACAO CHAR PARA DISPOSITIVO:Heart Rate Sensor:::Dados:72
I/onCharacteristicChanged: ALTERACAO CHAR PARA DISPOSITIVO:Heart Rate Sensor:::Dados:72
I/onCharacteristicChanged: ALTERACAO CHAR PARA DISPOSITIVO:Heart Rate Sensor:::Dados:72
I/onCharacteristicChanged: ALTERACAO CHAR PARA DISPOSITIVO:Heart Rate Sensor:::Dados:72
I/onCharacteristicChanged: ALTERACAO CHAR PARA DISPOSITIVO:Heart Rate Sensor:::Dados:72
I/onCharacteristicChanged: ALTERACAO CHAR PARA DISPOSITIVO:Heart Rate Sensor:::Dados:72
I/onCharacteristicChanged: ALTERACAO CHAR PARA DISPOSITIVO:Heart Rate Sensor:::Dados:73
I/onCharacteristicChanged: ALTERACAO CHAR PARA DISPOSITIVO:Heart Rate Sensor:::Dados:73
I/onCharacteristicChanged: CHAR CHANGE FOR DEVIDE:Bluno:::Data:AM:14031Vector Acela
I/onCharacteristicChanged: CHAR CHANGE FOR DEVIDE:Bluno:::Data:racao:130ALERTA_QUED
I/onCharacteristicChanged: ALTERACAO CHAR PARA DISPOSITIVO:Heart Rate Sensor:::Dados:74
I/onCharacteristicChanged: ALTERACAO CHAR PARA DISPOSITIVO:Heart Rate Sensor:::Dados:74
I/onCharacteristicChanged: CHAR CHANGE FOR DEVIDE:Bluno:::Data:A
I/onCharacteristicChanged: ALTERACAO CHAR PARA DISPOSITIVO:Heart Rate Sensor:::Dados:74
I/onCharacteristicChanged: ALTERACAO CHAR PARA DISPOSITIVO:Heart Rate Sensor:::Dados:74
I/onCharacteristicChanged: ALTERACAO CHAR PARA DISPOSITIVO:Heart Rate Sensor:::Dados:74
I/onCharacteristicChanged: ALTERACAO CHAR PARA DISPOSITIVO:Heart Rate Sensor:::Dados:74
I/onCharacteristicChanged: ALTERACAO CHAR PARA DISPOSITIVO:Heart Rate Sensor:::Dados:74
I/onCharacteristicChanged: ALTERACAO CHAR PARA DISPOSITIVO:Heart Rate Sensor:::Dados:74
I/onCharacteristicChanged: ALTERACAO CHAR PARA DISPOSITIVO:Heart Rate Sensor:::Dados:75
```

I/onCharacteristicChanged: ALTERACAO CHAR PARA DISPOSITIVO:Heart Rate Sensor:::Dados:75
I/onCharacteristicChanged: CHAR CHANGE FOR DEVIDE:Bluno:::Data:AM:13925Vector Acela
I/onCharacteristicChanged: CHAR CHANGE FOR DEVIDE:Bluno:::Data:racao:-2147483648ALE
I/onCharacteristicChanged: CHAR CHANGE FOR DEVIDE:Bluno:::Data:RTA_QUEDA
I/System.out: ALERTA QUEDA DETETADA
V/BoostFramework: BoostFramework() : mPerf = com.qualcomm.qti.Performance@c78958
V/BoostFramework: BoostFramework() : mPerf = com.qualcomm.qti.Performance@2de7ab1
I/onCharacteristicChanged: ALTERACAO CHAR PARA DISPOSITIVO:Heart Rate Sensor:::Dados:76
I/onCharacteristicChanged: ALTERACAO CHAR PARA DISPOSITIVO:Heart Rate Sensor:::Dados:77
I/onCharacteristicChanged: ALTERACAO CHAR PARA DISPOSITIVO:Heart Rate Sensor:::Dados:77
I/onCharacteristicChanged: ALTERACAO CHAR PARA DISPOSITIVO:Heart Rate Sensor:::Dados:77
I/onCharacteristicChanged: ALTERACAO CHAR PARA DISPOSITIVO:Heart Rate Sensor:::Dados:77
I/onCharacteristicChanged: ALTERACAO CHAR PARA DISPOSITIVO:Heart Rate Sensor:::Dados:77
I/onCharacteristicChanged: ALTERACAO CHAR PARA DISPOSITIVO:Heart Rate Sensor:::Dados:76
I/onCharacteristicChanged: ALTERACAO CHAR PARA DISPOSITIVO:Heart Rate Sensor:::Dados:76
Application terminated.

Código da Aplicação Bluno:

Aplicacao bluno/arduino

Initializing I2C devices...

Testing device connections...

MPU6050 connection successful

Updating internal sensor offsets...

-1936 -703 860 220 76 -85

-1936 -703 860 220 76 -85

AM:14072

Vector Acelaracao:-2147483648

AM:14060

Vector Acelaracao:-2147483648

AM:14242

Vector Acelaracao:140

AM:7839

Vector Acelaracao:-2147483648

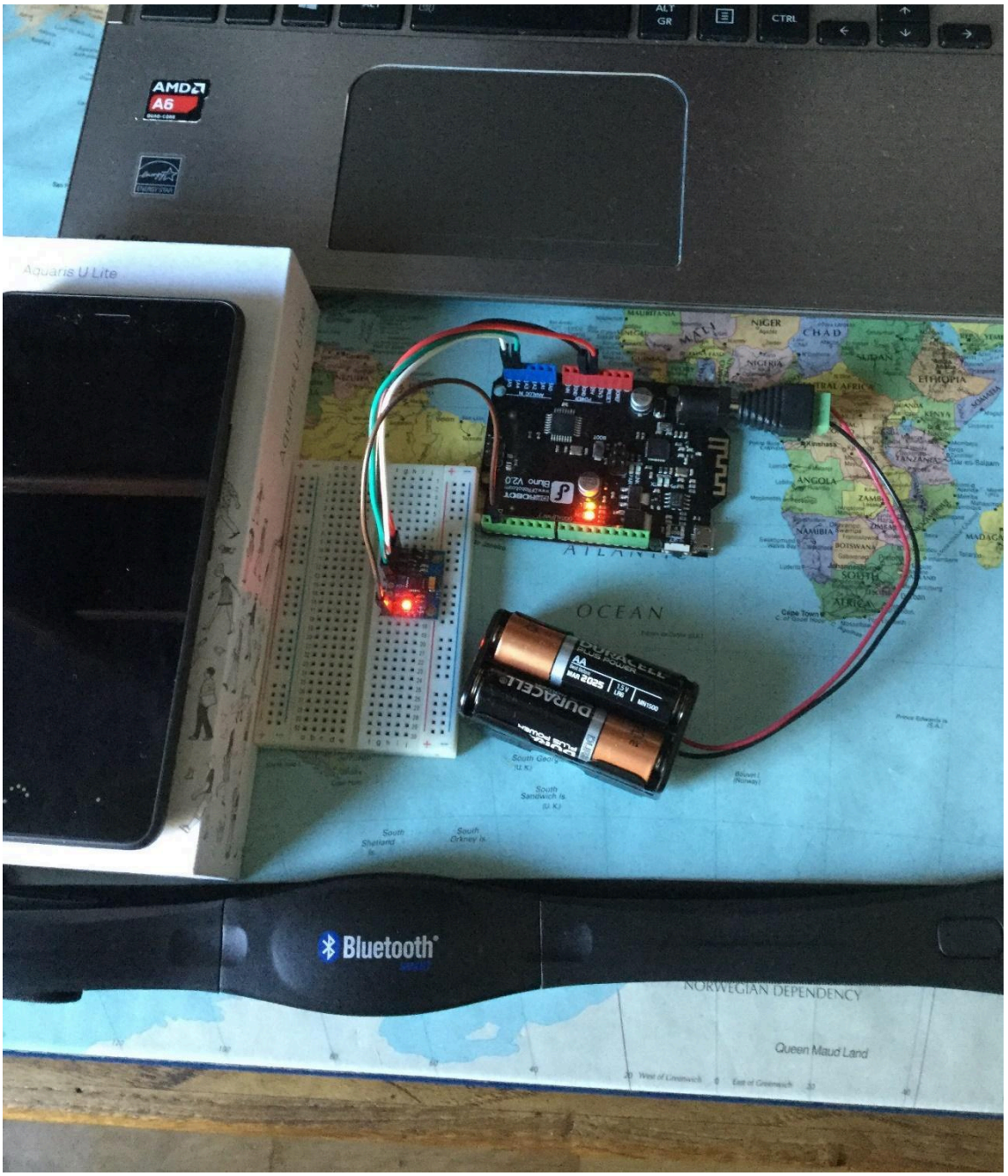
AM:8484

Vector Acelaracao:161

5 - Conclusão

Apresento um protótipo que integra os vários dispositivos que tendo em conta a informação recolhida, tratada e comunicada permite um controle parcial da postura bem como os batimentos cardíacos.
O registo dos dados é analisada e deteta o processo de alerta de queda.

Anexo 1 – Montagem dos Equipamentos



Anexo 2 – Estados do Ciclo de Vida de uma Atividade

Estados do

, and `onDestroy()`. The system invokes each of these callbacks as an activity enters a new state.

Figure 1 Ciclo de Vida de uma Atividade

Activity-lifecycle concepts

To navigate transitions between stages of the activity lifecycle, the Activity class provides a core set of six callbacks: `onCreate()`, `onStart()`, `onResume()`, `onPause()`, `onStop()`presents a visual representation of this paradigm.

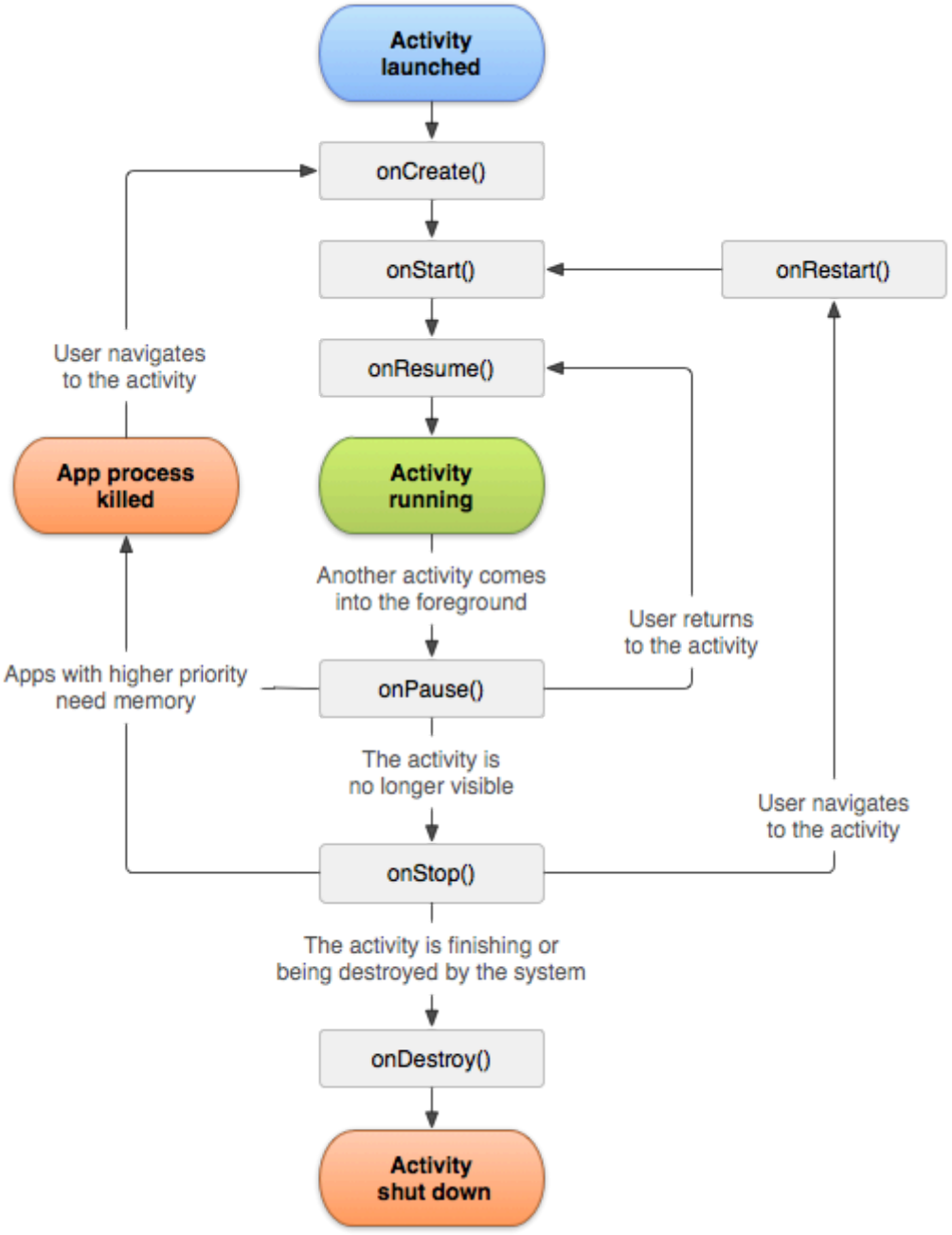
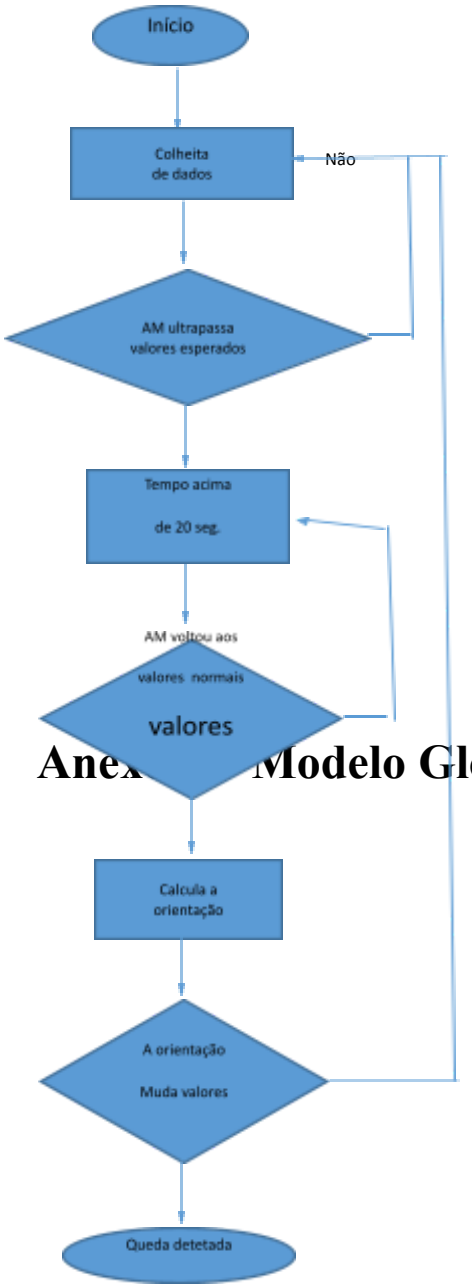
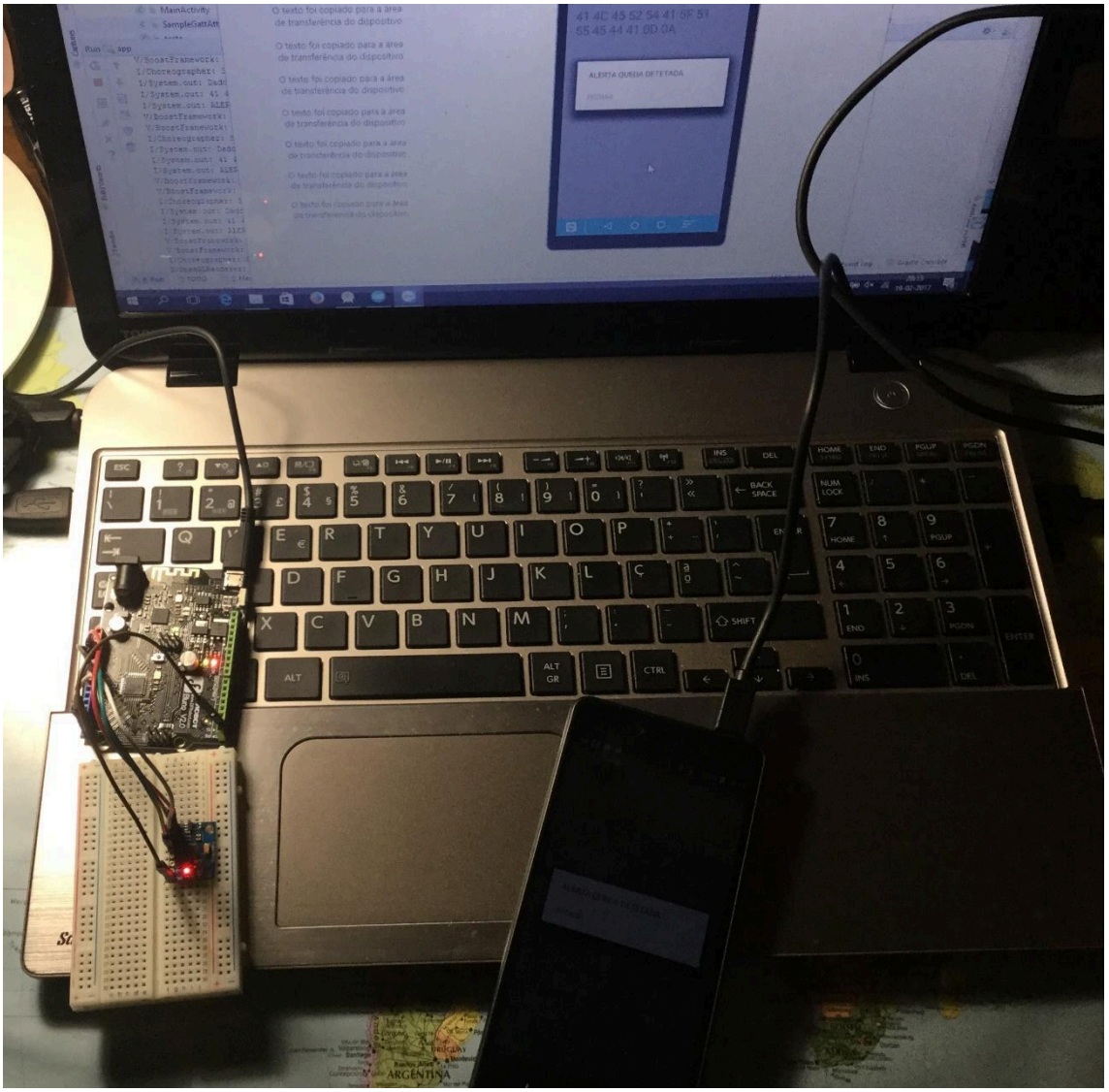


Figure 1. A simplified illustration of the activity lifecycle.

Anexo 3 – Fluxograma de Queda



Anex Modelo Global em Funcionamento


ALERTA QUEDA DETETADA

Anexo 5 – Recursos utilizados

Ambiente Arduino Uno – https://www.arduino.cc/en/main/software

Ambiente Android Studio – https://developer.android.com/studio/index.html

Aquisições de Hardware:

Equipamento	Vendedor
Telemóvel BQ SPH Aquarius U Lite	Media Market
Kit Arduino Uno	Box Electrónica
ITEAD Bluetooth Low Energy BLE…	Box Electrónica
Módulo Acelerómetro Giroscópio 3 eixos MPO 6050	ElectrFun
Caixa	Mauser Electronics

Bibliografia utilizada:

O livro ““Android – Desenvolvimento de aplicações com Android Studio de Ricardo Queirós da editora FCA.