Cyclist Health Alert

By

Li, Yuanqing
Sheng, Yutian
Shi, Yingzheng

Final Report for ECE 445, Senior Design, Spring 2016
TA: Luke Wendt

04/May/2016
Project No. 61

Abstract

Cyclist Health Alert is a system that will increase the safety of cyclist. It is designed to alert the user of over-exercised and to notify others of dangerous, unexpected impact. The system can be consolidated into three major components: the sensors, the circuit, and the Android phone app. It involves the use of both software and hardware. The system ran purely on battery power and communicate with smartphone through Bluetooth. Cyclist Health Alert is a robust, user friendly system that establish real time data transmission.

Contents

1. Introduction

1.1 Purpose

In the past, more than 24,000 exercise related injuries were reported each year [1]. Safety is crucial and injuries should always be avoided. Currently, many cyclist protective system already exist. However, those systems primarily focused on the awareness of others instead of reactive measures after accidents. A similar system called “Cycle Alert” [2] was created by a British company that designed to prevent collisions with cyclists and increase driver awareness by making cyclists visible in vehicle blind spots. While the names of two projects are extremely similar, the approach is different- one proactive another reactive. Cyclist Health Alert will alert the user of over exercised and notify others of dangerous, unexpected impact.

1.2 Objectives

1.2.1 Goals and Benefits

a. Adding another layer of protection to cyclist
b. Notify the cyclist when over exercised
c. Notify the emergency contact in case of collision

1.2.2 Features and Functions

a. Measure the acceleration of the bike
b. Measure the heart rate of the cyclist
c. Measure the force of the cyclist on bike
d. Real time transmission of sensor data
e. Warning message from smartphone for over threshold heart rate
f. Emergency text message sent to contact number along with GPS location

1.3 Block Diagram

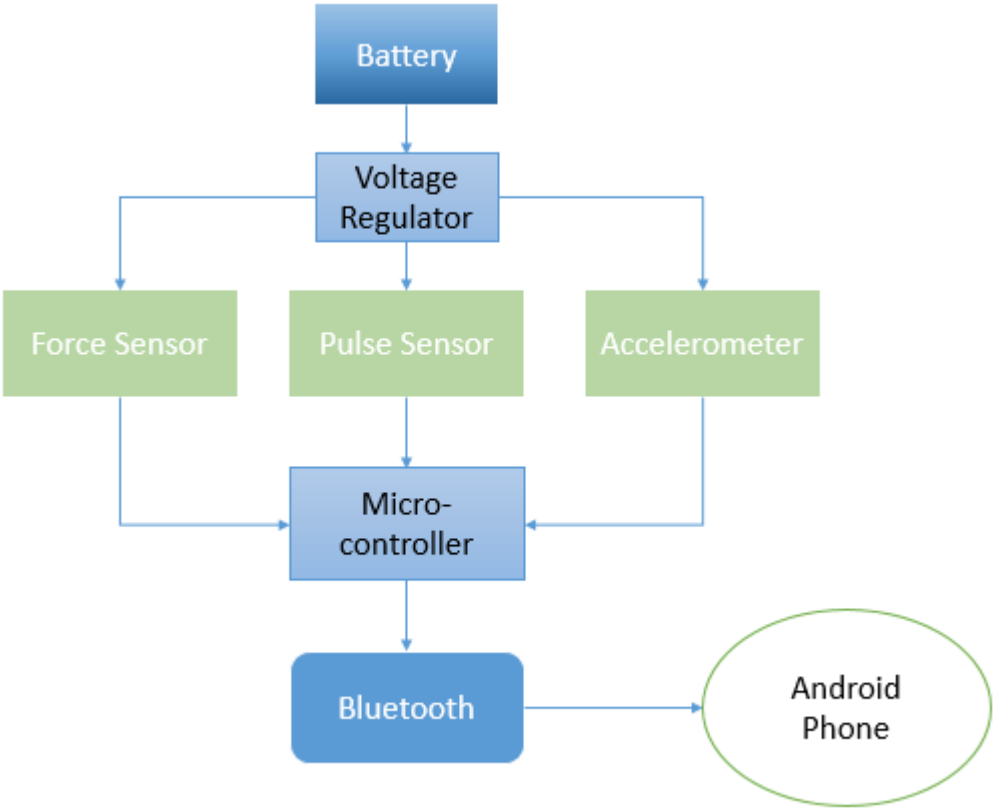


Figure 1: Top-level block diagram

2. Design

2.1 Hardware

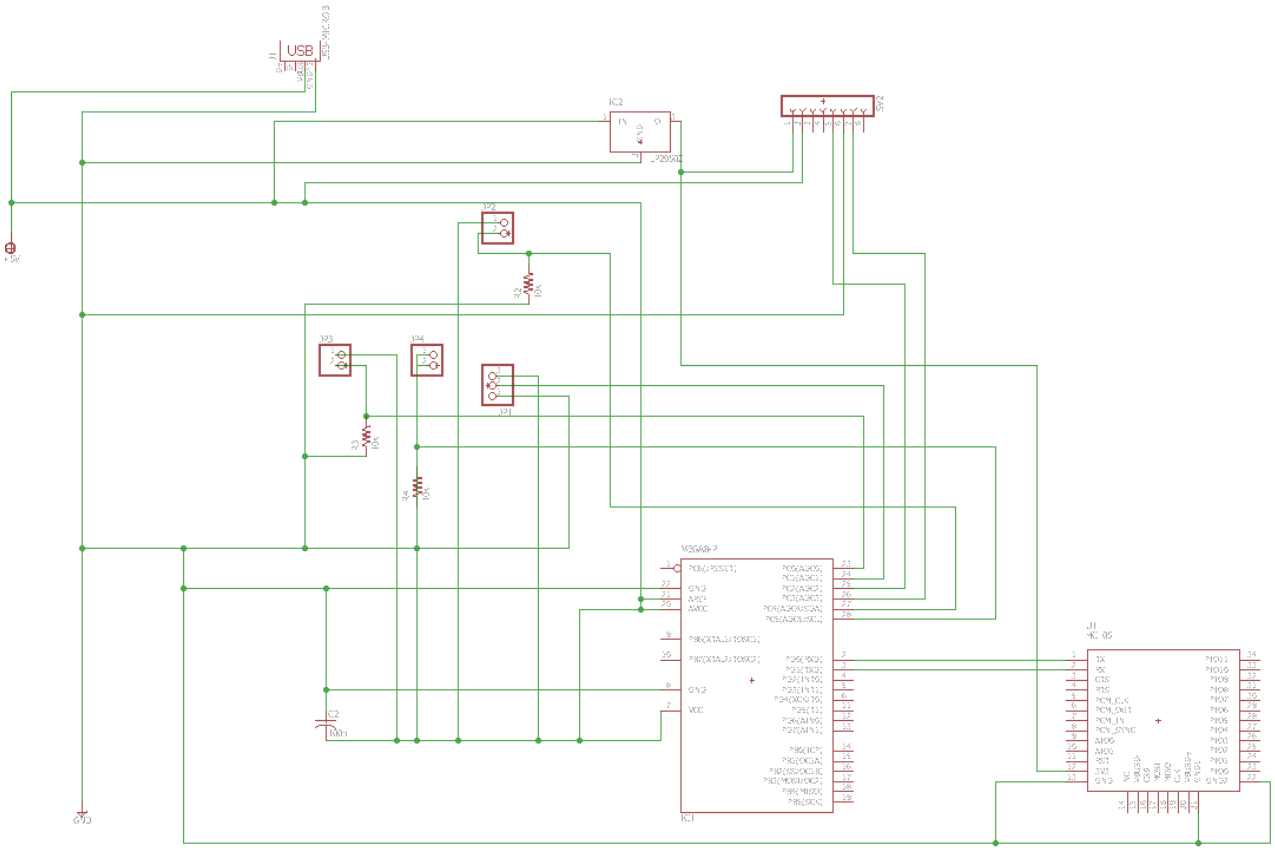


Figure 2: Overall Circuit Schematic

2.1.1 Power Supply

All of our sensors require 5V +/- 0.5V DC power. We chose a traditional 9 Volts battery and a regulator LM7805. The output voltage is 5V +/- 0.2V, the rated current is 1A [3]. The purpose of the LED in the circuit was for indication of power.

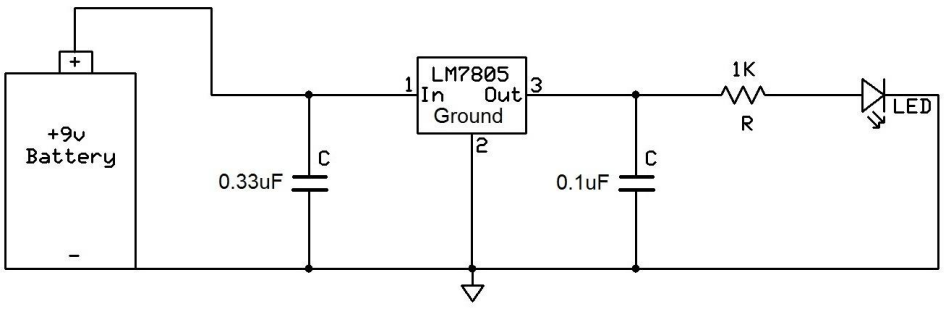


Figure 3: Circuit of Power Supply [4]

2.1.2 Force Sensor

Figure 5 shows the force sensors we used which were FSR402 and FSR406 force sensitive resistors (FSR). We chose FSR 400 series was because it is ultra-thin, robust, and sensitive. The force sensitivity range between 0.2N to 20N [5]. The resistance will exhibit a decreasing trend with increase in force applied to the surface of the sensor. Figure 4 shows the relationship between the resistance and the force applied. We place two FSR402 sensors on the handle, and one square FSR406 sensor on the saddle. Then we used voltage divider circuit to measure the resistance and convert it to the force we applied. Equation (2.1) was used to obtain the resistance value.

$$V_{out} = \frac{R_M V}{(R_M + R_{FSR})} \tag{2.1}$$

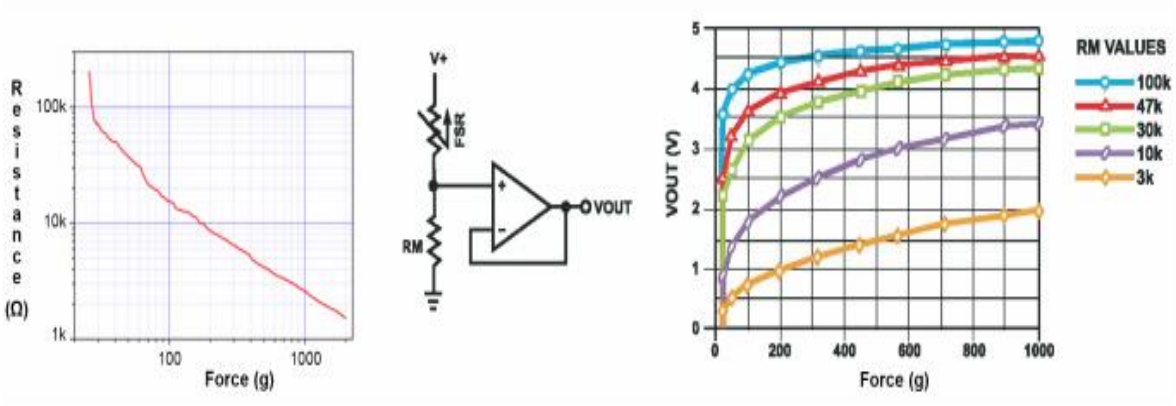


Figure 4: Schematic and figure of force resistance (g as gram) [6]

The R_m was set as 10kΩ to maximize desired force sensitivity range and to limit the current. The user is consider off the bike when all three sensors have the values below the threshold.

2.1.3 Pulse Sensor

Figure 6 is the pulse sensor that we used to measure the heart rate of the biker. We place the pulse sensor on the right handle bar near the location of the thumb. The pulse sensor sent out analog waves signals that fluctuate around the half operating voltage point. The microcontroller monitors then process the signal and decides whether a pulse is found [7]. When the signal drops below the middle point, the microcontroller will watch and wait until the next pulse. After collecting data, we divide the number of pulses by the time interval obtain the heart rate value. We averaged every 50 calculated values for smooth data. Even though we did some calibration, a number of factors still influenced the pulse readings. For instance, the long cable might cause unstable data transmission and unnecessary force applied on the sensor could impact the readings.

2.1.4 Accelerometer

We used MMA8451 triple-axis accelerometer to detect the acceleration of the bike. The accelerometer has a wide usage range from 18 m/s^2 to 96 m/s^2 as well as a high precision with 14-bit ADC [8]. In Figure 7, the fourth line shows the raw data in a number ranging from -8192 to 8191 because of the 14-bit ADC. The next line shows converted data in m/s^2. We gathered accelerations from x, y and, z axis, then normalized them using Equation (2.2). The magnitude that calculated from Equation (2.2) was then compared with the threshold value. Once the acceleration reach the threshold value, along with force sensors' data below the threshold, the emergency system would trigger.

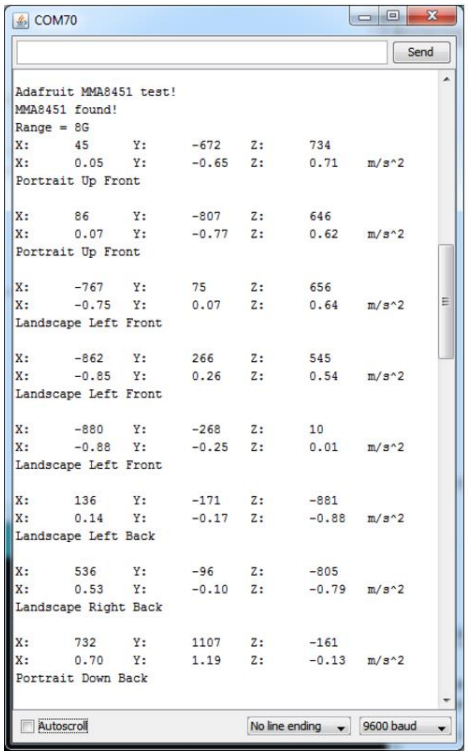


Figure 7: Sample window for accelerometer [9]

$$Acce = sqrt(x^2+y^2+z^2) \quad (2.2)$$

2.1.5 Microcontroller

We used Atmega328P as our microcontroller. Before it is soldered into the per board, Arduino UNO code was uploaded. In order to work independently on the per board, we connected two 24pf capacitors, a 10k resistors, and 16MHz crystal to the microcontroller. The microcontroller receives its power from a 5 Volts source. The inputs of accelerometer, pulse sensor and pressure sensor can be found below in Figure 8. Microcontroller also monitor the acceleration and force. If the acceleration reach beyond the threshold value and the force below the threshold value, microcontroller will send an emergency signal to smartphone through Bluetooth module.

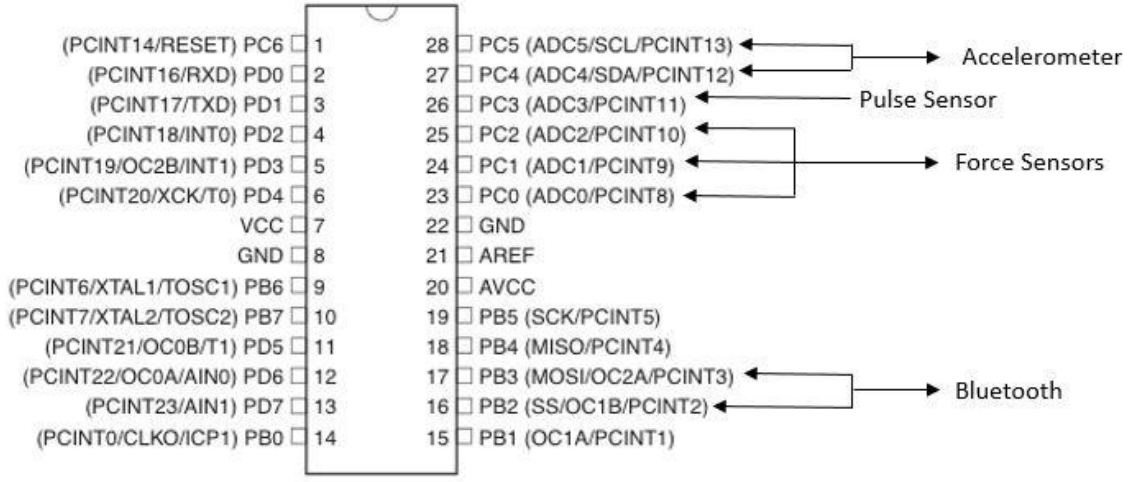


Figure 8: Connection of microcontroller [10]

2.1.6 Bluetooth

The Bluetooth module we used was HC-05 serial port Bluetooth module. HC-05 has slave and master mode in one module and uses V2.0 protocol standards. The working voltage is 3.3V and the default baud rate is 9600 [11]. We used this Bluetooth module to communicate between microcontroller and smart phone. In order to have the Android application knew where and when to pick up the data transmitted from microcontroller, we must write header “AABB” and tail “CC”, along with the data, to the application. This way, the Android application could receive and process the data properly. The transmission rate was set to 200 ms and the data was 6-Bytes long including 3-Bytes of “AABBCC”. One byte of data from pulse sensor, force sensor and accelerometer, respectively.

2.2 Software

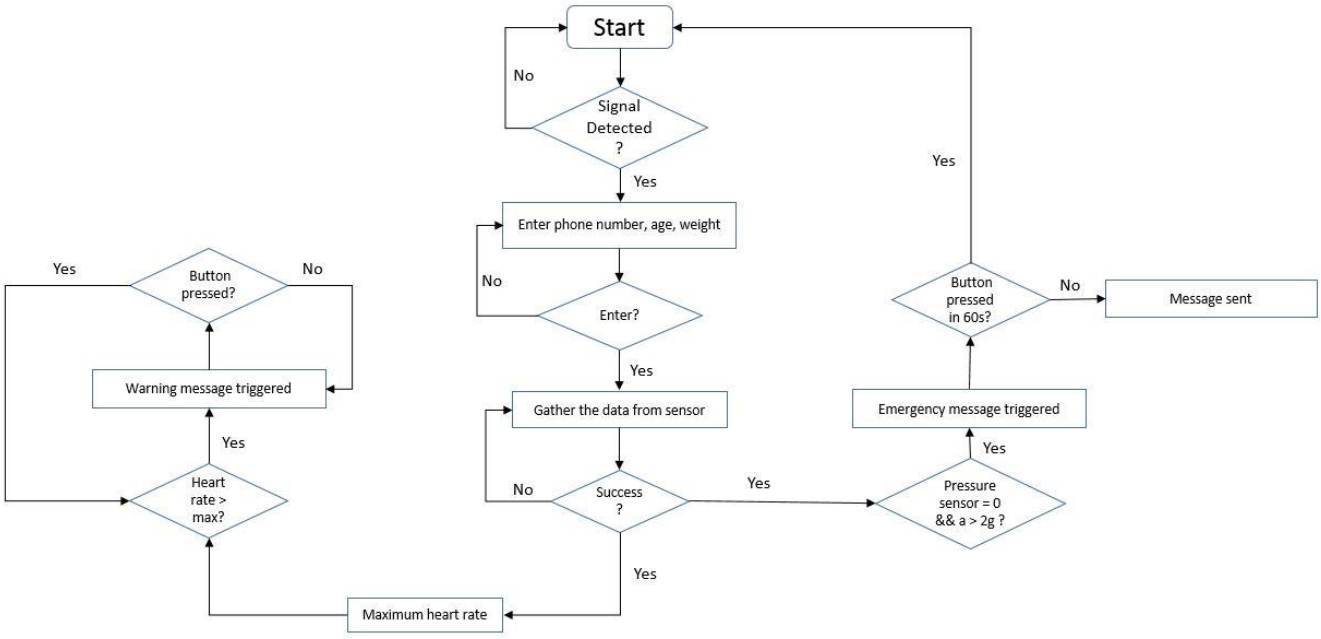


Figure 9: Flowchart of Software

We developed an Android application in order to accomplish the warning and emergency system. Once the basic information of the user, including the emergency contact, age and weight, were typed in, the application could start processing the data. When the heart rate of user reached above the maximum heart rate threshold, the application would trigger the audio alert to warn the user. When force sensor and accelerometer registered values below threshold, the application would send emergency text message to the emergency contact number. There is a 60 seconds delay and cancel option in case of false alert.

3. Design Verification

The Requirement and Verification table can be found in Appendix A.

3.1 Force Sensor Test

We use the Arduino with the sample code [12] to test the force sensors. Figure 10 is showing the results of our test. If there is no force on the sensor, it will read zero; it will have the value once we exert force it.

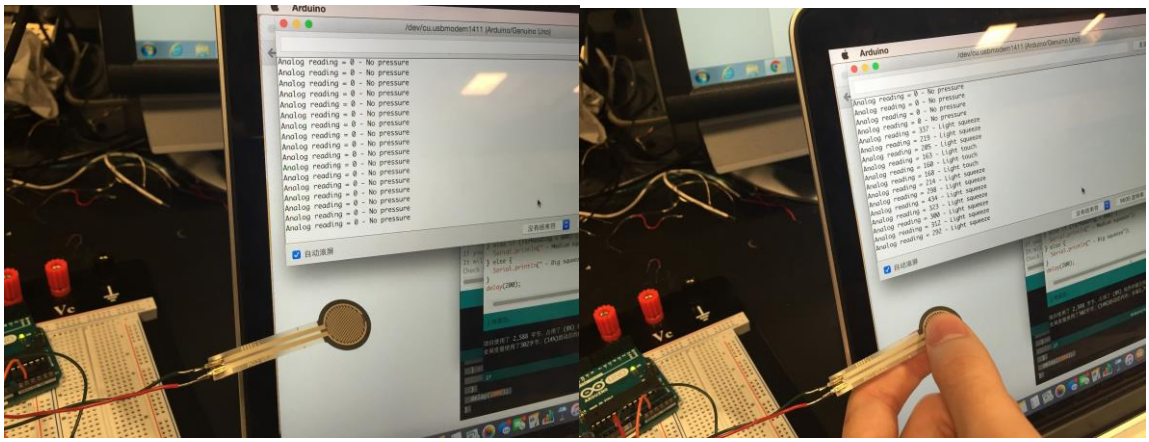


Figure 10: Test result of force sensor

3.2 Pulse Sensor Test

We used the Arduino sample code [13] to test the pulse sensors. The output data shown in Figure 11 is Beats Per Minute (BPM) which will be calibrated and send to the Android phone.

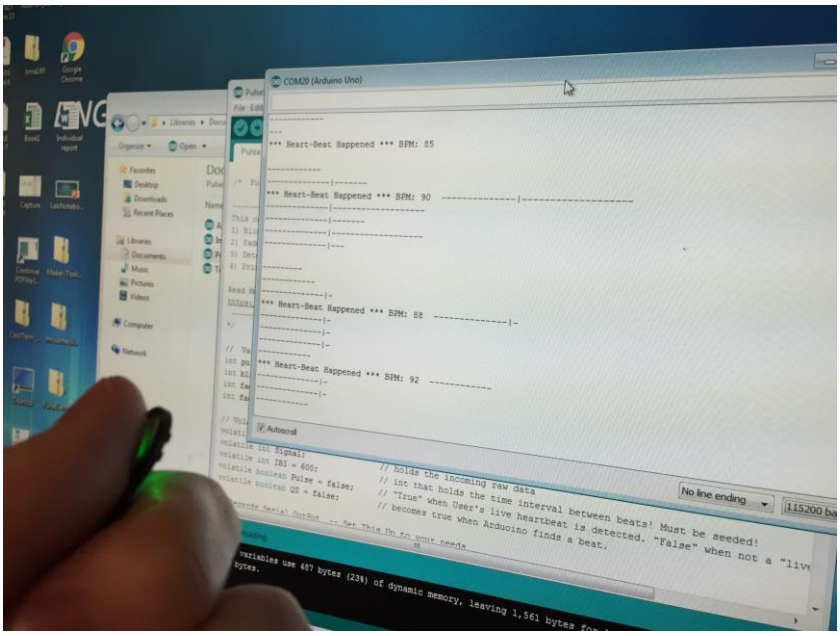


Figure 11: Test result of pulse sensor

3.3 Accelerometer Test

We used the demo code of MMA8451 on the Adafruit website [14] to test the accelerometer. Figure 12 shows the result of the test. The first line of numbers are the raw data of acceleration, ranging from -8912 to 8191 because of the 14-bits ADC. The second line is the processed output with unit in m/s^2. The third line is the orientation of the accelerometer.

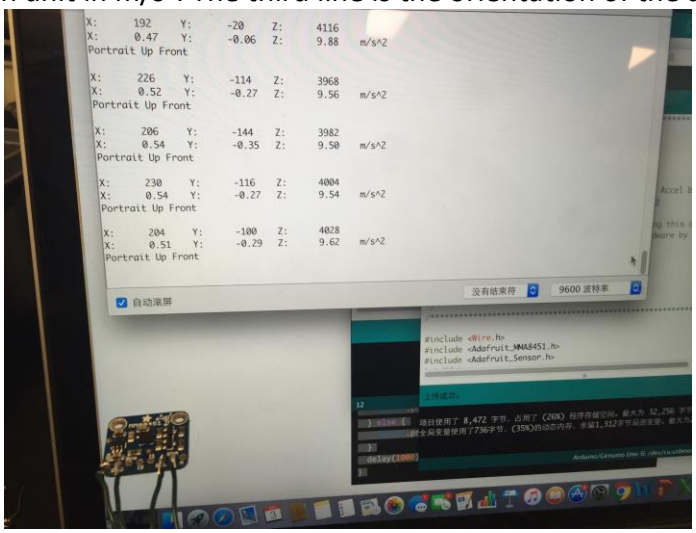


Figure 12: Test result of accelerometer

3.4 App test

Figure 13 and Figure 14 show the demonstration result of warning alert and emergency system.

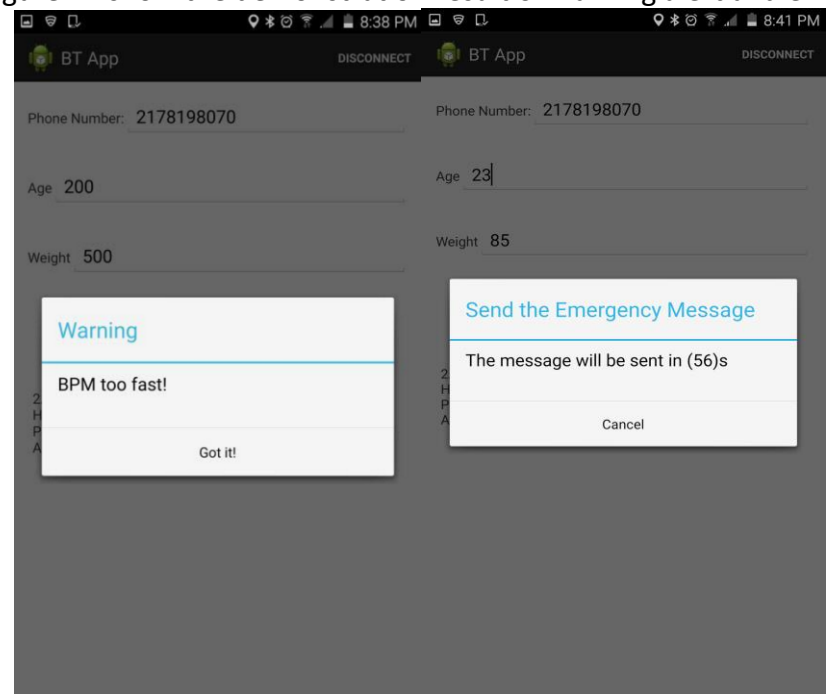


Figure 13: Test result of warning alert

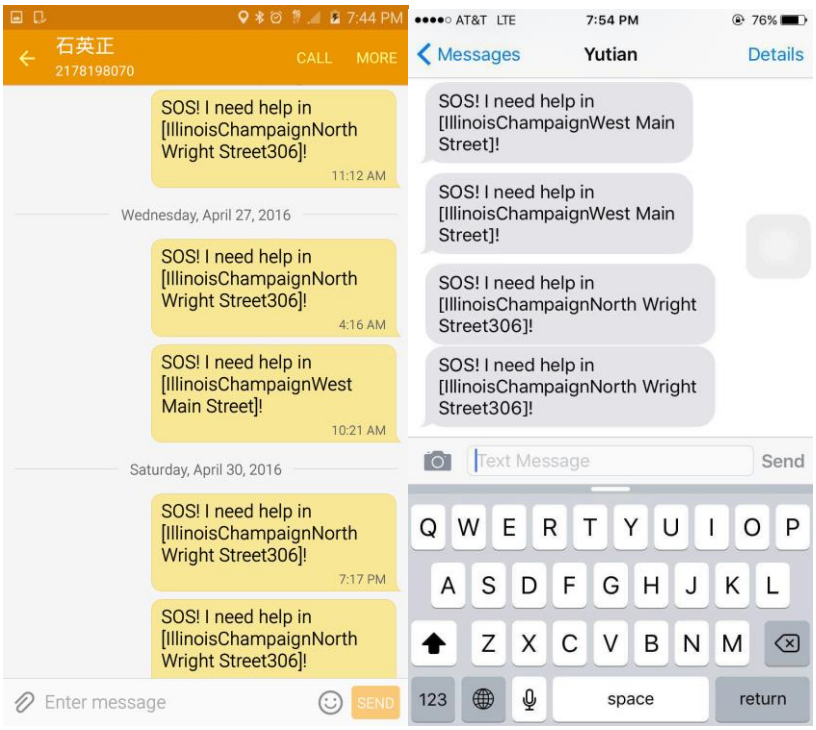


Figure 14: Test result of emergency system

4. Costs

4.1 Parts

Table 1. Parts Costs

Parts	Manufacturer	Amount	Bulk Purchase Cost ($)	Actual Cost ($)
Pulse Sensor	Asiawill	1	24.99	24.99
Force Sensitive Resistor FSR® 402	Interlink	2	12.99	25.98
Force Sensitive Resistor FSR® 406	Interlink	1	14.99	14.99
Voltage Regulator L7805CV	Addicore	1	5.95	5.95
Accelerometer BMA180	Kingduino	1	11.95	11.95
Bluetooth Hc-05	JBtek	1	8.59	8.59
Ribbon Cables	Kalevel	1	9.99	9.99
Microcontroller ATmega328	Atmel	1	1.69	1.69
E522 9V Alkaline battery	Energizer	3	2.99	8.97
Total				113.1

4.2 Labor

Table 2. Cost for Labor

Name	Hours Invested	Hourly Rate	Total
Li	150	$35	$13,125
Sheng	150	$35	$13,125
Shi	150	$35	$13,125
Total	450		$39,375

5. Conclusion

5.1 Accomplishments

Data from every sensor proved to be within reasonable range. The accelerometer were able to capture acceleration of the bike within unit digit. The force sensor respond to slightest weight on it. Microcontroller was able to process the raw data from sensor into standardized units, which made threshold detection possible. The Bluetooth connection with Android phone app was established without interference. Real time data transmission was achieved and the refreshing rate was set as 100 millisecond. The Android phone app had a user friendly interface that allow easy navigation.

5.2 Uncertainties

In this project there are two major challenges we encountered. The first was the difficulty with real time data transmission. Measurement data was collected from each sensor and sent to smartphone app through bluetooth. Since the pulse sensor and the accelerometer each have separate clock cycles, interference happened due to this issue. The solution was adding an extra interrupt function inside the microcontroller code, which stopped one device of transmitting while the other one is functioning. The second challenge we faced was how to write an Android phone app. Our team composed of only electrical engineers, which means we had minimal coding background. We had to learn everything from scratch. Through time and practice, trial and error we were able to compile a perfect working code.

5.3 Ethical Considerations

Our group members, do hereby commit ourselves to the highest ethical and professional conduct and agree the following rules:

1. to accept responsibility in making decisions consistent with the safety, health, and welfare of the public, and to disclose promptly factors that might endanger the public or the environment;
We take all responsibility of the safety and health related issues, including any inflicted damage to ourselves as well as the others.

2. to be honest and realistic in stating claims or estimates based on available data;
We will make all claims based on solid evidence. There will be no fake data.

3. to seek, accept, and offer honest criticism of technical work, to acknowledge and correct errors, and to credit properly the contributions of others;
We will respect all criticism and advice, as well as taking it seriously. No advice will be ignored.

4. to treat fairly all persons and to not engage in acts of discrimination based on race, religion, gender, disability, age, national origin, sexual orientation, gender identity, or gender expression;
We will treat all team members with respect. No one's idea will be discredited due to race, religion, gender, disability, age, national origin, sexual orientation, gender identity, or gender factor.

5. to avoid injuring others, their property, reputation, or employment by false or malicious action;
We will avoid injuring third party personal and properties during testing.

6. to assist colleagues and co-workers in their professional development and to support them in following this code of ethics.
We will help each team member to follow the ethics code mentioned above.

5.4 Future work

There are three main focus for the project to improve upon: power efficiency of the system, data accuracy from pulse sensor, and integration of wires with the bike.

The first possible improvement is the power efficiency of the system. The current logic of the system required constant real time data transmission, which burdens the power consumption of the battery. The lifetime of battery could only last three to four hours. To solve this issue, we could move the data processing from Android App end to the microcontroller. This way the bluetooth is only transmitting data at alert conditions.

Second, we would add filter to pulse sensor for more accurate data. Due to the limitation of small size and cost, the pulse sensor was not as accurate as we want. The results experience random fluctuation from noise factors such as pressure and body oil. Adding a filter would eliminate the signal's noise portion.

Third, we want to increase the overall aesthetic value of the bike by sorting out the wires. We used hard wiring for connection from each sensor to the circuit board. Each force sensor required two wires, pulse sensor required three wires, and accelerometer required five.

Reference

[1] Madhani, A. Treadmill injuries send thousands to the ER every year. USA Today. Retrieved from http://www.usatoday.com/story/news/2015/05/04/treadmill-emergency-room-injuries-exercise-equipment/26898487/

[2] The new HGV Blind spot technology. Cycle Alert. Retrieve fromhttp://www.cyclealert.com/

[3] LM78XX/LM78XXA 3-Terminal 1 A Positive Voltage Regulator. FAIRCHILD. Retrieved from https://www.fairchildsemi.com/datasheets/LM/LM7805.pdf

[4] How to connect a voltage regulator in a circuit. Learning about electronics. Retrieved from http://www.learningaboutelectronics.com/Articles/How-to-connect-a-voltage-regulator-in-a-circuit

[5] FSR 400 Series Data Sheet. INTERLINK Electronics. Retrieved from https://www.interlinkelectronics.com/datasheets/Datasheet_FSR.pdf

[6] FSR 400 Series Data Sheet. INTERLINK Electronics. Retrieved from https://www.interlinkelectronics.com/datasheets/Datasheet_FSR.pdf

[7] Pulse Sensor. Sparkfun. Retrieved from https://www.sparkfun.com/products/11574

[8] Overview. Adafruit MMA8451 Accelerometer Breakout. Retrieve from https://learn.adafruit.com/adafruit-mma8451-accelerometer-breakout/overview

[9] Wiring & Test. Adafruit MMA8451 Accelerometer Breakout. Retrieve from https://learn.adafruit.com/adafruit-mma8451-accelerometer-breakout/wiring-and-test

[10] Gilbert, D and Buechley, L. AVR Programing Tutorial. High-Low Tech wiki. Retrieved from http://highlowtech.org/wiki/pmwiki.php?n=Main.AVRProgrammingAdvanced

[11] BlueTooth-HC05-HC06-Modules-How-To. arduino-info-Bluetooth. Retrieve from https://arduino-info.wikispaces.com/BlueTooth-HC05-HC06-Modules-How-To

[12] Using an FSR. Force Sensitive Resistor. Retrieve from https://learn.adafruit.com/force-sensitive-resistor-fsr/using-an-fsr

[13] GitHub. Retrieve from https://github.com/WorldFamousElectronics/PulseSensor_Amped_Arduino

[14] Wiring & Test. Adafruit MMA8451 Accelerometer Breakout. Retrieve from https://learn.adafruit.com/adafruit-mma8451-accelerometer-breakout/wiring-and-test

[15] IEEE Code of Ethics. IEEE. Retrieve from

http://www.ieee.org/about/corporate/governance/p7-8.html

[16] Android Baidu positioning API use. Programmer Share. Retrieve from http://www.programmershare.com/2685108/

[17] Android Developers. Retrieve from http://developer.android.com/index.html

[18] GitHub. Retrieve from https://github.com/wyouflf/xUtils3

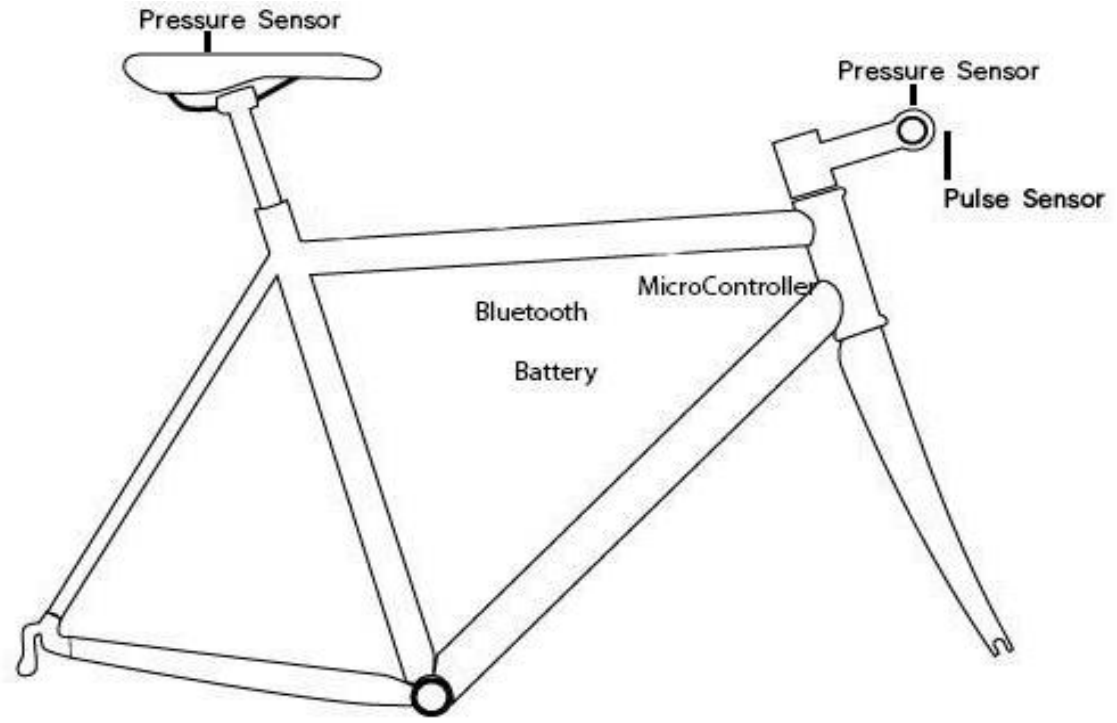
Appendix A

Requirement	Verification	Verification Status
Power Supply (Battery) a. Supply of 5V +/- 0.25V with 1A +/- 0.25A	a. First use multimeter to draw 0A and measure the voltage. Then draw 1A and measure the voltage again. The values should be within the bounds. b. Place a load and measure the current delivered to the load.	a.Y b.Y
Pressure Sensor (Force Sensitive Resistor) a. Can detect (> 150g) of force b. Accuracy measure force within +/- 25%	a. Set up a circuit with power supply (5V), pressure sensor, 1k resistor in series. b. Set up a voltmeter in parallel with pressure sensor. c. Apply 200g weight on the pressure sensor, record the value in voltmeter. d. Apply 500g weight on the pressure sensor, record the value in voltmeter. e. Apply 1000g weight on the pressure sensor, record the value in voltmeter. f. Plot the voltage vs. force, the voltage should decrease as the force increasing g. Apply a force of 100g to check for boundary sensitivity. Since the pressure sensor is only check for yes or no case, the accuracy range is only for the lower bound.	a.Y b.Y c.Y d.Y e.Y f.Y g.Y
Pulse Sensor a. Correct reading of heart rate b. Vin < 5V +/- 0.25V, I <4mA +/- 0.02A	a. Clip the pulse sensor to earlobe or fingertip and plug it into 5V Arduino to get test results b. Measure the heart rate by using smartphone app Safe Health or other methods. Then compare the values. Theoretically two values should be nearly the	a.Y b.N

	same.	
Accelerometer a. Can detect >1g of acceleration with error of +/- 5%	a. Power the microcontroller with 3.3 V and connect it to accelerometer b. Apply force to the device by body motion to test detection of acceleration	a.Y b.Y
Bluetooth Module a. Able to transmit signal from microcontroller to smartphone b. Must send data to the phone at 1 +/- 0.5kb/s	a. Put bluetooth and smartphone in a range of 0.5m and pair them up b. Sent out a signal from bluetooth transceiver and check to see if the app on the smartphone could receive the signal and react to the signal c. Send data from bluetooth to the phone and measure the time it takes	a.Y b.Y c.Y
Microcontroller a. The operating voltage is between 1.8V to 5V b. The active mode current is 0.2mA(+/- 0.01mA), where the frequency is 1 Mhz and voltage should be 1.8V. c. Must react to the data sent to the chip and generate the correct output	a. Connect the chip to multimeter and measure the voltage to see if it is in the range of 1.8V to 5V. Also verify that the active mode current is approximately 0.2mA by setting the frequency to 1Mhz and voltage to 1.8V. b. At first, implement a simple computer program to test the input/output pins. c. Sending signals from sensor system and see if the microcontroller can react and produce the emergency and warning signal.	a.Y b.Y c.Y
Smartphone a. Receive signal from bluetooth device within 20 seconds b. Android system triggers	a. First establish connection with bluetooth transceiver b. receive a test signal from bluetooth transceiver c. Check the signal detail using computer	a.Y b.Y c.Y d.Y

audio alert and sending text message	programs d. Compare the data received with the data transmitted from bluetooth	
Voltage Regulator a. Input Voltage is 5V +/- 0.1V b. Output Voltage is 3.3V +/- 0.1V	a. Connect the input to a 5 volt power source b. Use multimeter to measure the output voltage to check if its 3.3V	a.Y b.Y

Appendix B: Location of Parts



Appendix C: Source Code

```
package com.example.bt;
import java.util.ArrayList;
import java.util.List;
import com.baidu.location.BDLocation;[16]
import com.baidu.location.BDLocationListener;
import com.baidu.location.LocationClient;
import com.baidu.location.LocationClientOption;
import com.example.bt.R;
import cn.fly2think.btlib.BluetoothService;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.telephony.SmsManager;
import android.app.Activity;
import android.app.AlertDialog;
import android.app.AlertDialog.Builder;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.content.DialogInterface;
import android.content.Intent;
import android.media.MediaPlayer;
import android.view.KeyEvent;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView; [17]

public class MainActivity extends Activity {

 // Intent requestcode
 private static final int REQUEST_CONNECT_DEVICE = 1;
 private static final int REQUEST_ENABLE_BT = 2;

 // Bluetooth name
 private String mConnectedDeviceName = null;
 private List<Byte> byteArray = new ArrayList<Byte>();

 // Bluetooth  adapter
 private BluetoothAdapter mBluetoothAdapter = null;
 private BluetoothService mBTService = null;
 private boolean menuEnable = true;
 private String menuTitle = "Connect";
 private boolean isReconnet = false;
 private BluetoothDevice mDevice;
 private String addr;
 private AlertDialog messageDialog;
```

```
private AlertDialog warningDialog;
private int time;
private Runnable runnable = new Runnable() {

  @Override
public void run() {
 // TODO Auto-generated method stub
  if (time > 0) {
       time--;
       messageDialog.setMessage("The message will be sent in (" + time + ")s");
       mHandler.postDelayed(this, 1000);
       messageDialog.show();

     } else {
       String message = "SOS! I need help in [" + addr + "]!";
       SmsManager manager = SmsManager.getDefault();
       ArrayList<String> list = manager.divideMessage(message); // if the message is too long, divide
       for (String text : list) {
         manager.sendTextMessage(PrefUtils.getPhone(MainActivity.this), null, text, null, null); [18]
       }
       messageDialog.dismiss();
     } } };

// get the location
private LocationClient mLocationClient;
private MediaPlayer mdp;

// BluetoothService back to Handler
private final Handler mHandler = new Handler() {
   @Override
   public void handleMessage(Message msg) {
     switch (msg.what) {
     //Change the bluetooth status
     case BluetoothService.MESSAGE_STATE_CHANGE:
       switch (msg.arg1) {
       // Connect success
       case BluetoothService.STATE_CONNECTED:
         menuEnable = true;
         menuTitle = "Disconnect";
         invalidateOptionsMenu();
         break;
       //if connecting
       case BluetoothService.STATE_CONNECTING:
        menuEnable = false;
         menuTitle = "Connect";
         invalidateOptionsMenu();
         break;
       // if bluetooth isn't connected
```

```
  case BluetoothService.STATE_LISTEN:
  case BluetoothService.STATE_NONE:
    menuEnable = true;
    menuTitle = "Connect";
    invalidateOptionsMenu();
    break;
  }
  break;
case BluetoothService.MESSAGE_WRITE:
  break;
// the message received by bluetooth
case BluetoothService.MESSAGE_READ:
  byte[] readBuf = (byte[]) msg.obj;
  int len = msg.arg1;

  for (int i = 0; i < len; i++) {
    if (byteArray.size() < 17) {
      byteArray.add(readBuf[i]);
      if (byteArray.size() == 1) {
        if (byteArray.get(0) != (byte) 0xaa) {
          byteArray.clear();
        }
      } else if (byteArray.size() == 2) {
        if (byteArray.get(1) != (byte) 0xbb) {
          byteArray.clear();
        }
      } else if (byteArray.size() == 6) {
        if (byteArray.get(5) == (byte) 0xCC) {
          int size = byteArray.size();
          Byte[] result = (Byte[]) byteArray.toArray(new Byte[size]);
          forwardData(toPrimitive(result));
       }
        byteArray.clear();
      } }  }

  break;
// The name of Bluetooth
case BluetoothService.MESSAGE_DEVICE_NAME:
  // save the connected device's name
  mConnectedDeviceName = msg.getData().getString(BluetoothService.DEVICE_NAME);
  ToastUtils.showToast(MainActivity.this, "Connect to" + mConnectedDeviceName);
  break;
case BluetoothService.MESSAGE_TOAST:
  switch (msg.arg1) {
  // Bluetooth isn't connected
  case BluetoothService.MESSAGE_FAILED_CONNECTION:
    ToastUtils.showToast(getApplicationContext(), R.string.unable_connect);
    if (isReconnet && mDevice != null) {
```

```
            mBTService.connect(mDevice);
          }
          break;
        // Lost connection
        case BluetoothService.MESSAGE_LOST_CONNECTION:
          ToastUtils.showToast(getApplicationContext(), R.string.lost_connection);
          if (isReconnet && mDevice != null) {
            mBTService.connect(mDevice);
          }
          break;
        default:
          break;
        }
        break;
      // If Bluetooth isn't connected
      case BluetoothService.MESSAGE_NOT_CONNECT:
        ToastUtils.showToast(getApplicationContext(), R.string.not_connected);
        break;
      }   } };

  @Override
  protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);


    // Get library
    mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();

    // Initialize location
    mLocationClient = new LocationClient(this, getLocationClientOption());
    mLocationClient.registerLocationListener(mLocationListener);

    if (!mLocationClient.isStarted())
      mLocationClient.start();
    mLocationClient.requestLocation();

    // Check if the smartphone has Bluetooth
    if (mBluetoothAdapter == null) {
      ToastUtils.showToast(this, R.string.not_support);
      finish();
      return;
    }
    messageDialog = new AlertDialog.Builder(this).setNegativeButton("Cancel", new
DialogInterface.OnClickListener() {

      @Override
      public void onClick(DialogInterface dialog, int which) {
```

```
        // TODO Auto-generated method stub
        mHandler.removeCallbacks(runnable);
        messageDialog.dismiss();
      }
    }).create();
    messageDialog.setTitle("Send the Emergency Message");
    messageDialog.setCancelable(false);

    warningDialog = new AlertDialog.Builder(this).setNegativeButton("Got it!", new
DialogInterface.OnClickListener() {

      @Override
      public void onClick(DialogInterface dialog, int which) {
        // TODO Auto-generated method stub

        mdp.stop();
        messageDialog.dismiss();
      }
    }).create();
    warningDialog.setMessage("BPM too fast!");
    warningDialog.setTitle("Warning");
    warningDialog.setCancelable(false);

  }

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

    // If Bluetooth close, open it
    if (!mBluetoothAdapter.isEnabled()) {
      Intent enableIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
      startActivityForResult(enableIntent, REQUEST_ENABLE_BT);

    } else {
      if (mBTService == null) {
        // Initialize Bluetooth
        mBTService = new BluetoothService(this, mHandler);
      }    } }

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

    // Check
    if (mBTService != null) {
      // STATE_NONE shows that Bluetooth never be connected
      if (mBTService.getState() == BluetoothService.STATE_NONE) {
```

```
        mBTService.start();
      }    } }

@Override
public void onDestroy() {
  super.onDestroy();
  isReconnet = false;
  // Connection will lost automatically if the app is closed
  if (mBTService != null)
    mBTService.stop();
  if (mLocationClient != null)
    mLocationClient.stop();

}

@Override
public boolean onCreateOptionsMenu(Menu menu) {
  // Inflate the menu; this adds items to the action bar if it is present.
  getMenuInflater().inflate(R.menu.main, menu);
  return true;
}

@Override
public boolean onPrepareOptionsMenu(Menu menu) {

  MenuItem menuItem = menu.findItem(R.id.action_settings);
  menuItem.setTitle(menuTitle);
  menuItem.setEnabled(menuEnable);

  return super.onPrepareOptionsMenu(menu);
}

@Override
public boolean onOptionsItemSelected(MenuItem item) {
  // Handle action bar item clicks here. The action bar will
  // automatically handle clicks on the Home/Up button, so long
  // as you specify a parent activity in AndroidManifest.xml.
  int id = item.getItemId();
  if (id == R.id.action_settings) {

    if (mBTService.getState() == BluetoothService.STATE_NONE
        || mBTService.getState() == BluetoothService.STATE_LISTEN) {

      Intent serverIntent = new Intent(this, DeviceListActivity.class);
      startActivityForResult(serverIntent, REQUEST_CONNECT_DEVICE);

      // isReconnet = true;
      isReconnet = false;
```

```
    } else {

      if (mBTService != null)
        mBTService.stop();

      menuEnable = true;
      menuTitle = "Connecting";
      invalidateOptionsMenu();

      isReconnet = false;
    }

    return true;
  }
  return super.onOptionsItemSelected(item);
}

@OnClick({ R.id.btn_save })
public void click(View v) {
  // TODO Auto-generated method stub
  switch (v.getId()) {
  case R.id.btn_save:

    String phone = field_phone.getText().toString().trim();
    String weight = field_weight.getText().toString().trim();
    String age = field_age.getText().toString().trim();

    if (phone.equals("")) {
      ToastUtils.showToast(this, "Please type in Emergency Contact");
      return;
    }
    if (age.equals("")) {
      ToastUtils.showToast(this, "Please enter your age");
      return;
    }
    if (weight.equals("")) {
      ToastUtils.showToast(this, "Please enter your weight");
      return;
    }
    ToastUtils.showToast(this, "Saved!");
    break;

  default:
    break;
  } }

public void onActivityResult(int requestCode, int resultCode, Intent data) {
```

```
  switch (requestCode) {
  case REQUEST_CONNECT_DEVICE:
    if (resultCode == Activity.RESULT_OK) {
      String address = data.getExtras().getString(DeviceListActivity.EXTRA_DEVICE_ADDRESS);
      mDevice = mBluetoothAdapter.getRemoteDevice(address);
         mBTService.connect(mDevice);
    }
    break;
  case REQUEST_ENABLE_BT:
    if (resultCode == Activity.RESULT_OK) {
      mBTService = new BluetoothService(this, mHandler);
    } else {
      ToastUtils.showToast(this, R.string.not_connected);
      finish();
    }    } }

@Override
public boolean onKeyDown(int keyCode, KeyEvent event) {
  // TODO Auto-generated method stub
  if (keyCode == KeyEvent.KEYCODE_BACK) {
    AlertDialog dialog = new Builder(this).setTitle(R.string.exit_title).setMessage(R.string.exit_message)
        .setPositiveButton(R.string.exit_ok, new DialogInterface.OnClickListener() {
          @Override
          public void onClick(DialogInterface dialog, int which) {
            // TODO Auto-generated method stub
            isReconnet = false;
            finish();
          }
        }).setNegativeButton(R.string.exit_no, null).create();
    dialog.setCanceledOnTouchOutside(false);
    dialog.show();

    return true;
  }

  return super.onKeyDown(keyCode, event);
}

private byte[] toPrimitive(Byte[] array) {
  byte[] result = new byte[array.length];
  for (int i = 0; i < array.length; ++i) {
    result[i] = array[i].byteValue();
  }
  return result;
}

private void forwardData(byte[] datas) {
  int bpm = datas[2] & 0xff;
```

```
    int pressure = datas[3] & 0xff;
    int acceleration = datas[4] & 0xff;

    StringBuffer sbf = new StringBuffer();
    sbf.append(TransUtils.Bytes2Hex(datas) + "\n");
    sbf.append("Heart Rate : " + bpm + "\n");
    sbf.append("Pressure : " + pressure + "\n");
    sbf.append("Acceleration : " + acceleration);

    tv_data.setText(sbf.toString());

    String ageStr = PrefUtils.getAge(this);
    String weightStr = PrefUtils.getWeight(this);
    String phone = PrefUtils.getPhone(this);

    if (!ageStr.equals("") && !weightStr.equals("")) {
      int age = Integer.parseInt(ageStr);
      float weight = Float.parseFloat(weightStr);

      // Set the maximum heartrate
      float maxBPM = 211.415f - (0.5f * age) - (0.05f * weight) + 4.5f;

      if (bpm > maxBPM) {
        if (mdp == null || !mdp.isPlaying()) {
          // Initialize the audio
          mdp = MediaPlayer.create(this, R.raw.warning);
          mdp.setLooping(true);
          mdp.start();
        }
        if (!warningDialog.isShowing()) {
          warningDialog.show();
        }      }    }
    if (!phone.equals("")) {
      if (pressure == 0 && acceleration == 0) {
        if (!messageDialog.isShowing()) {
          time = 60;
          mHandler.post(runnable);
        }      }    } }
  private LocationClientOption getLocationClientOption() {
    LocationClientOption option = new LocationClientOption();
    option.setOpenGps(true);
    option.setAddrType("all");
    option.setServiceName(this.getPackageName());
    option.setScanSpan(5000);
    option.disableCache(true);
    return option;
  }      }    } };
}
```