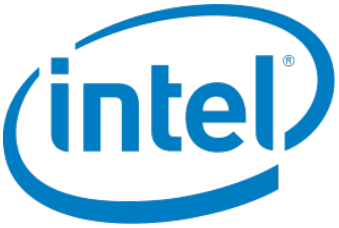
WEARABLE FALL DETECTION, MONITORING AND ALERT SYSTEM FOR THE ELDERLY

A project submitted as a part of the **Intel IoT Internship** conducted in association with the **Electronics and Communication** (ECE) department of **NIT Trichy**

By

Sudharshan Suresh

110113086



A student of

INSTRUMENTATION AND CONTROL ENGINEERING

NATIONAL INSTITUTE OF TECHNOLOGY

TIRUCHIRAPALLI- 620015

DECEMBER 2014

ACKNOWLEDGEMENT

I am deeply grateful to Dr. B. Venkataramani, Ms. R. K. Kavitha and Ms. R. Thilagavathy for their guidance, constant support and assistance in project work. I would like to thank Intel for sponsoring the program and enabling us to learn over the winter vacations. I would like to thank the department of Electronics and Communication Engineering of NIT Tiruchirapalli for providing us with components and a space to work.

In addition, I would also like to thank our 3rd year mentors for helping us during the initial stages of the internship.

Last but not the least; I give my sincere thanks to friends and family, who have assisted me throughout the course of this month.

ABSTRACT

Loss of consciousness and fall-induced injury is one of the leading problems faced by the elderly. They are the leading cause for the transfer of senior citizens from the comfort of their residence to assisted care environments. They also account for 40% of injury related deaths and lack of timely action can prove fatal. On the personal side, such sporadic events stifle the independence of the individuals, requiring a caretaker to maintain constant surveillance.

My aim is to design a system to sense and alert doctors and nearest of kin of falls and other related incidents of distress. It is also to provide the functionality of a pedometer, to constantly monitor and relay the activities of the individual. The device mustn't be cumbersome, but rather wearable, so as to remove any social stigma related to the possession of such a device. It must have a high degree of reliability and differentiate activities of daily life from genuine falls. This would require a signal plotting and thresholding-based approach to arrive upon a fall-detection algorithm.

With the dawn of the IoT (Internet of Things) era, it is ideal that such a device be constantly relaying data via such a wireless network. This ensures seamless connectivity between the patient, doctor and relatives – and swift response.

Keywords: Fall detection, elderly, monitoring, IoT

TABLE OF CONTENTS

Title **Page no.**

INTRODUCTION AND PROJECT MOTIVATION

Status Quo

We live in a generation that is pressed for time and that, arguably, has a lack of compassion. As a result of this negligence, the old and infirm are unfortunately forgotten. The institutions of assisted care environments mushroomed as a result of this alarming trend. Along with a rapid growth in aging population there is also a decrease in the population of caretakers.

In India for example, there is a tremendous increase in the elderly population, with a current estimate of 90 million over the age of 60. This is a result of a disproportionate growth among age groups, with the 60+ population increasingly far more rapidly than the 15-60 age group. The age dependence ratio of the country is now at a perilous 0.58. Studies show almost a quarter of such elders have poor health conditions, as one could expect. This means that an approximate 22.5 million individuals are at risk from occasional health complications.

To quote from the World Health Organizations 'Aging in India' report:

"The UN defines a country as 'ageing' where the proportion of people over 60 reaches 7 percent. By 2000 India will have exceeded that proportion (7.7%)and is expected to reach 12.6% in 2025."

This poses an alarming disadvantage to the elderly from a societal point of view. India has a long and cherished culture of respect and devotion towards elders. However, with the dawn of the nuclear family era, some communities find it hard to fully commit to the senior citizens. This has heralded the growth of NGO's and senior care centers such as HelpAge India. But still, this begs the question- is it possible to attend to our elders, regardless of the time? No, that is a far more pressing issue. It is indeed time India took to caring for senior citizens 24/7 by embracing technology.

Falls and their impact

Falls are one of the most prominent causes of unintentional injury. They are not as simple as what you may perceive them to be. They are coded as E880-E888 in International Classification of Disease-9 (ICD-9), and as W00-W19 in ICD-10, which include a wide range of falls including those on the same level, upper level, and other unspecified falls. For the sake of it, we shall even define it according to WHO literature-

"Inadvertently coming to rest on the ground, floor or other lower level, excluding intentional change in position to rest in furniture, wall or other objects".

Falls are responsible for 40% of all injury related deaths and need immediate medical attention. The reason for a fall may be heart problems, loss of consciousness, fatigue, exhaustion, diseases and loss of balance. Although falls occur through all age groups, the major chunks of incidents are those of senior citizens. Approximately 28-35% of people aged 65 and above fall every year. This figure only increases for the age group of 70+ with 32-42%.

The major talking point is the need for swift medical aid. 50% of all injury-type hospitalizations for the concerned age group are fall related. Thirty-two percent of elderly people aged over 75 years have ever fallen at least once a year, and among them, 24% have been seriously injured. Approximately 3% of all fallers lie unattended for periods greater than 20 minutes, leading to insufficient medical action. An upsetting figure is that 40% of nursing home admissions are directly linked with incidents of falls.

This highlights two major points-

i. Falls require immediate medical attention- and the only way one can establish that is by conceiving a sensing method.
ii. Falls greatly limit the independence of an individual. He or she will not have the confidence to venture or stay alone for extended periods of time. This is primarily due to the fear of medical complications surfacing.

The detection system

Now that the root issue has been established, we shall identify a method of sensing. The major criteria one has in mind for detection of falls are -

i. Non-invasive
ii. Wearable system
iii. No false-positives
iv. No delay time
v. Common route of communication
vi. Interactivity

In the world of technology, wearable have always had a special appeal. This is due to the ease of access and usage, as well as the seamless nature of the product. One does not bear the brunt of social stigma by wearing such a device, thus satisfying the user. Efficiency of the system is paramount, as the message must be delivered from point A to point B in a matter of seconds. An established mode of communication must be employed; in this case the Short Message Service (SMS) can reach people the quickest. False alarms can be both extremely annoying and wastes both time and money. So in order to ensure accuracy, the fall-detection algorithm must be perfected and the device must be made interactive.

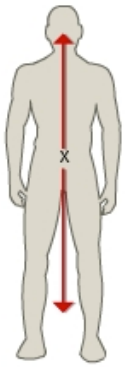


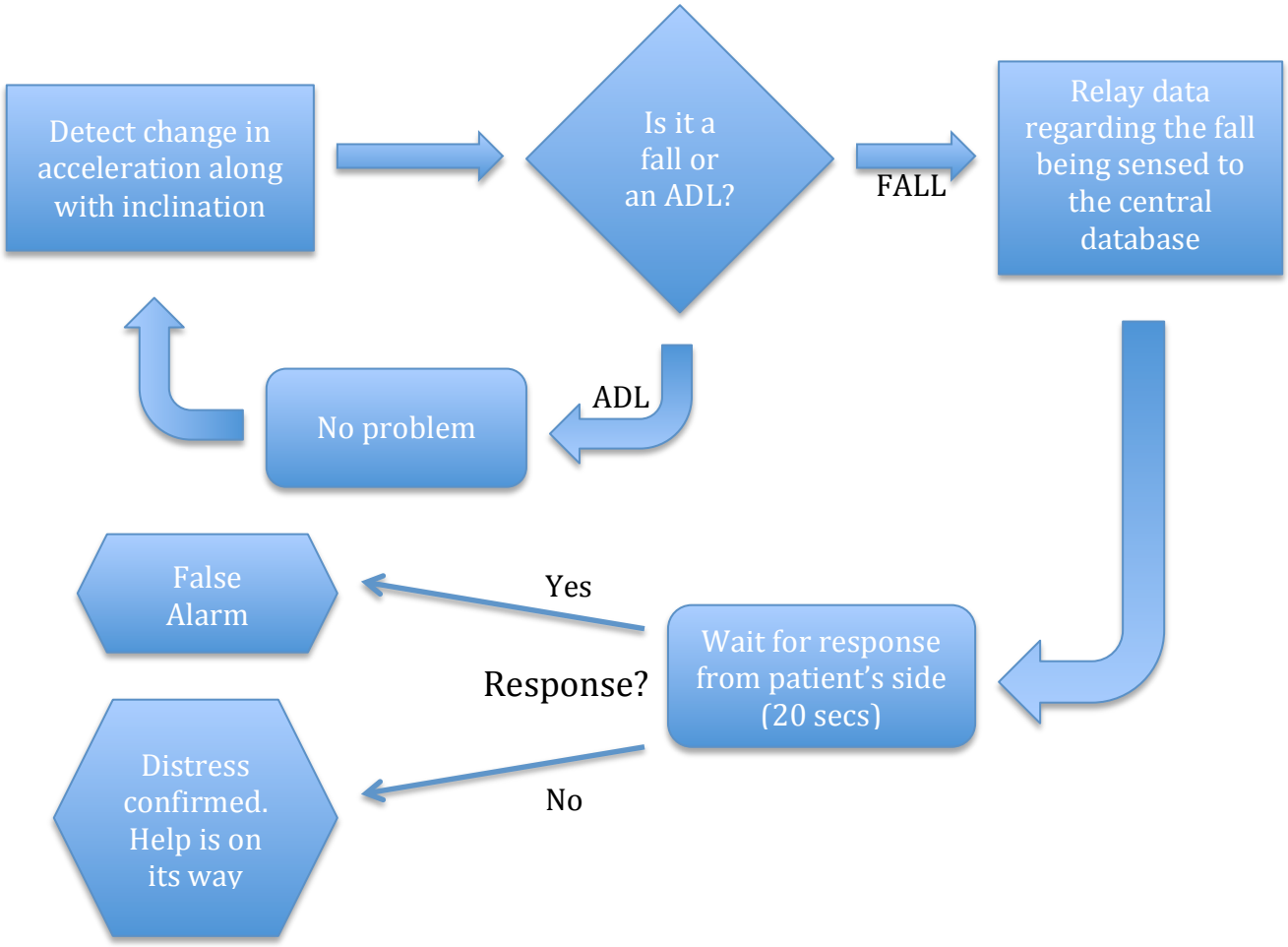
Figure 1: Human body's center of gravity

Taking all the above factors into consideration, I would propose the use of an accelerometer-based sensing device around the waistline. This would not only mimic the look of a commonplace belt, but is also the location of the center of gravity **(Figure 1)**. As a result, readings that one obtains from the sensor will be accurate and trustable.

Communication can be made possible via wireless means to a central database. This database is to be located at the doctor's end, giving a constant stream of information related to the patient. In case of any issues detected, a suitable alert is issued both at the central display, as well as via SMS to the doctors and relatives. SMS is the most rudimentary, ever-accessible means of communication in the modern world and cannot be missed by any individual.

One uses the Internet of things to communicate from machine-to-machine. We can employ various available modules such as Wifi-modules (eg: ESP 8266), Zigbee protocol, Ethernet shields, etc. However for the purpose of prototyping, we employ a Bluetooth serial transfer protocol between the microcontroller and the computer.

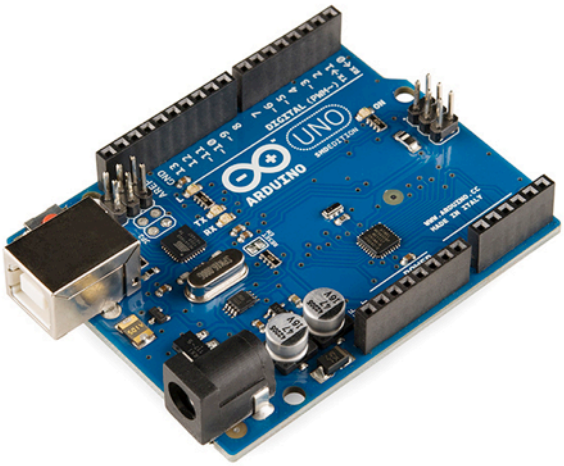
BASIC FLOWCHART



COMPONENT DESCRIPTION

I. Arduino Uno:

The Uno is a microcontroller board that is based on the **ATmega328**. It is widely preferred for its ease-of-use, programming capabilities and large pool of online documentation. It operates on very low current of 40-50 mA and has a logic level of 0 to 5V. It possesses 14 digital I/O pins (6 with PWM functionality). 6 analog inputs, onboard LEDs, a reset switch and ceramic resonator. It can be supplied power through USB connection or external power jack. It has a memory size of 32 KB, programmable via Arduino IDE.



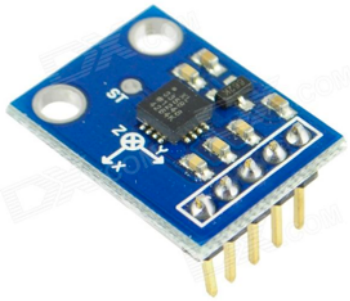
One could otherwise employ IoT ready boards such as the **Texas Instruments cc3200** or **Arduino Yun** or IoT implementable boards such as the **Intel Galileo**.

II. ADXL 335 (Accelerometer)

In order to sense angular position and acceleration we use the ADXL 335 MEMS-based accelerometer. It is a 3 axis system which gives analog output corresponding to the X, Y and Z axes of the orthogonal coordinate system.

The advantage of this sensor are twofold-

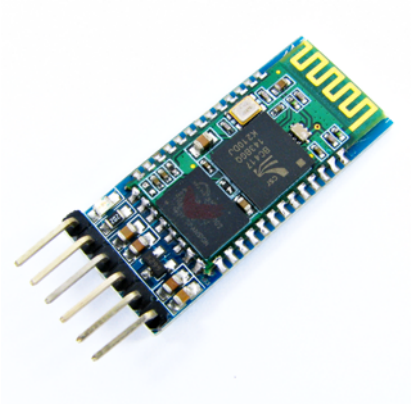
i. Extremely low noise
ii. Low power consumption of approximately just 320 micro Amperes.



The module has an entire sensing range of +/-3g.

III. HC-05 Bluetooth Module

For reasons of prototyping, we shall use this Bluetooth module to interface our device with the computer.



The HC-05 is a class 2 bluetooth module, capable of easy connectivity with android platforms or serial port communication with the computer. It serves as an efficient replacement for wired solutions, offering mobility. It operates on 3.3V DC logic with a current drawing capacity of 50mA.

IV. Piezo buzzer

V. Push Button Switch

VI. RGB 3 color LED

SOFTWARE AND LIBRARY PACKAGES REQUIRED

I. Arduino IDE

This is an open-source Arduino environment that enables us to type and upload code to the board via serial connection. Available for Windows, Mac OS X and Linux, this environment is the ubiquitous go-to software for Arduino projects. The environment is structure using Java and based off Processing, another software we shall be using.

II. Processing

Processing is an open-source programming language and IDE that is aimed towards computer programming in a visual context. Each project is referred to as a sketch – providing computational and graphics functions. This allows us to receive instant visual feedback to any process. A major advantage processing possesses is the wide-array of 3rd party libraries available for use. Similar to Arduino, it too is built upon the Java model.

III. Temboo

Temboo is a virtual library that is our key to the Internet of Things. With streamlined access to over 100+ APIs across various programming languages, this intuitive web-based coding framework helps us perform machine-to-machine communication. We are required to create an ID for their service and download the library package for Processing. Using this, one can transmit text messages to cellular device via an online server.

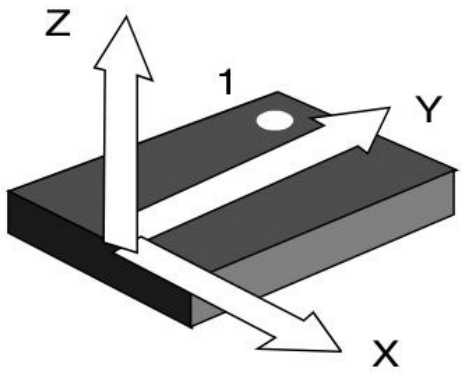
IV. Twilio

Twilio is a cloud-based communication portal that allows developers to make calls and send SMS programmatically. The web service API can be accessed used the choreos of Temboo and billing is based on usage. For this, we can use a trial

account on Twilio that gives us limited access to generating text messages to a pre-determined number.

OBTAINING MEANINGFUL DATA FROM THE ACCELEROMETER

Accelerometers measure the acceleration, most likely due to motion of a body. However, when the accelerometer is fixed, only the gravity pulling down on it is sensed. The device detects linear acceleration along three perpendicular axes. If one was to sample the x, y and z axes data, they would get an accurate idea regarding the orientation of the object.



The analog values that the microcontroller receives are the acceleration quantities along each axis. However, one must first calibrate their device to ensure accuracy keeping the device in a position such as the one shown in the figure. We know that both X and Y-axis readings must be 0 as it is parallel to the ground while the Z-axis reading must be the maximum. Adjust the correcting factor to obtain this.

In the above orientation, the values in g's must be – **(0g, 0g, 1g).** Simple mathematical manipulation tells us to divide the corrected analog value by the maximum value.

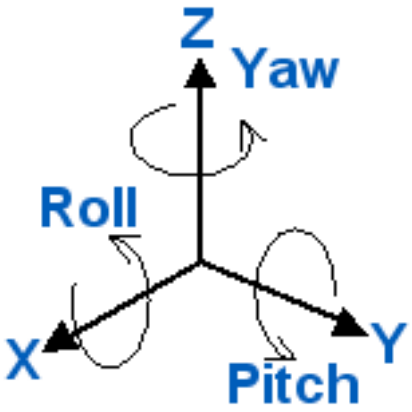
Thus, we have successfully converted the voltage values into g's value.

The net acceleration is the **root of the sum of the squares of all 3 values.**

$$\mathbf{A_{net}} = \sqrt{Ax^2 + Ay^2 + Az^2}$$

Net acceleration is an incredibly valuable quantity for fall detection. When a body undergoes free-fall, the phenomenon of weightlessness will occur. This implies that during a fall, the A_{net} will tend towards 0g. It is true an ordinary fall is different from free-fall; but A_{net} will still be substantially lesser than 1g.

The final quantities of interest we obtain are those of orientation – **pitch and roll**.



These values give us an understanding of orientation of the user, which is vital when it comes to detecting falls. The acceleration along each axis can be used, along with a bit of trigonometry, to find out the pitch and roll angles.

Pitch is $tan^{-1}(\frac{Ax}{\sqrt{Ay^2+Az^2}})$. Roll is $tan^{-1}(\frac{Ay}{\sqrt{Ax^2+Az^2}})$.

FALL DETECTION ALGORITHM

It was an extremely difficult process to generalize falls based on only two parameters. However after many trial and error sessions and a whole lot of signal analysis, I present to you my algorithm:

To safely conclude that a given action is a fall, you must eliminate the other activities of daily life. These include lying down, sitting, jumping, running, squatting, hopping, etc. Therein lies a major difficultly, because many of these actions give similar analog readings as that of falling. Thus assigning simple thresholds will never do the job. What I have done is create a series of checkpoints that must all be satisfied to conclude that a certain event is a fall.

Sample graph showing net acceleration on fall

Checkpoint 1: As elucidated in the previous section, there will be a fall in the A_{net} value when the fall commences. Thus if the value goes below a specific threshold (in this case, 0.8g) it passes the first checkpoint.

Checkpoint 2: Following the fall, the user makes impact with the ground or surface. A sharp spike in the A_{net} readings characterizes this. For our purposes, we consider

this as greater than 1g. The time lag between the checkpoints must be no greater than 2 seconds.

Checkpoint 3: The pitch and roll of the user is checked and compared with a predefined threshold (In this case, greater than 60 degrees). The time lag between checkpoint 2 and 3 must be very small – in the order of milliseconds.

Checkpoint 4: There is now a good chance that the user has fallen. We now check for a period of 3 seconds whether the user gets up, by setting a threshold on inclination. If 3 seconds have elapsed, we can assume a fall has been sensed.

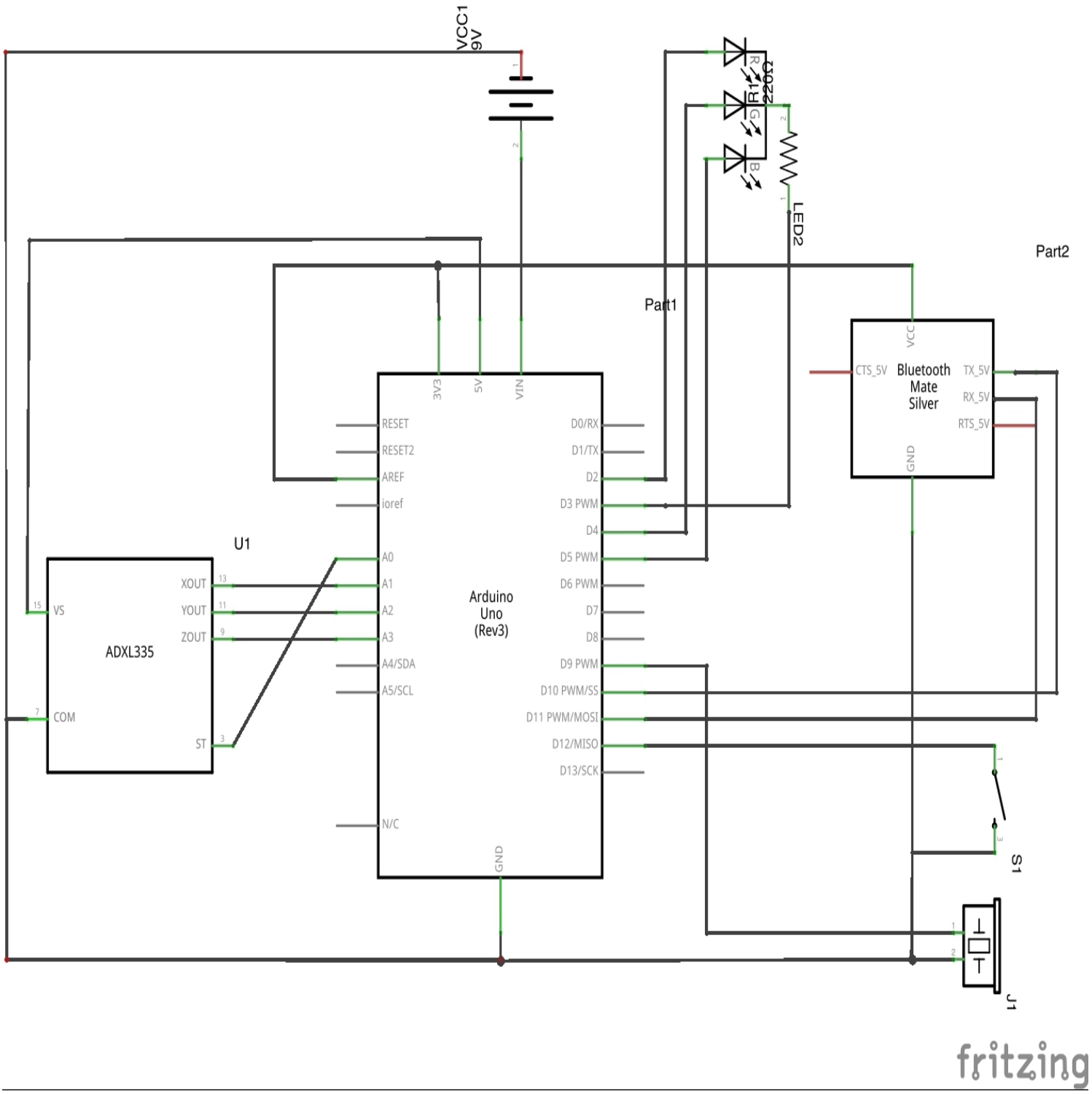
Checkpoint 5: We now wait for a period of 20 seconds for the user to disable the alert, in case of false alarm. If 20 seconds have elapsed, we confirm the fall and required action is taken.

PEDOMETER FUNCTIONALITY

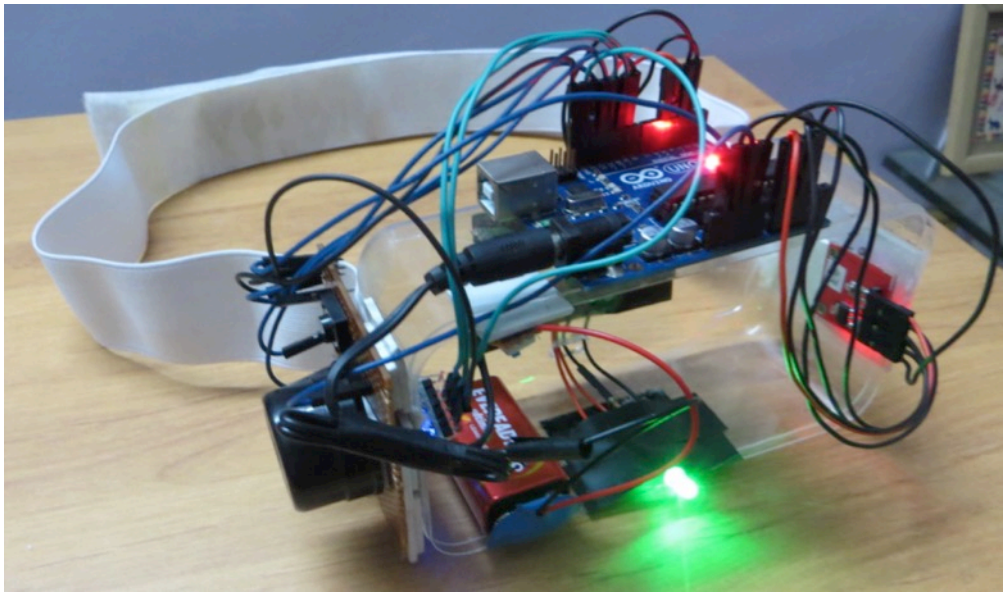
The number of steps and distance travelled by an individual is an indicator of the health of the person. To identify a step taken, we use a series of thresholds, similar to that used in the fall detection.

The first criterion is that the user must be in 'stable' state. This means that he must be upright and not in the process of falling.
The second criteria it checks for is a spike in the net acceleration as a result of foot contact. Each time a step is taken, we detect this and increment the number of steps taken.

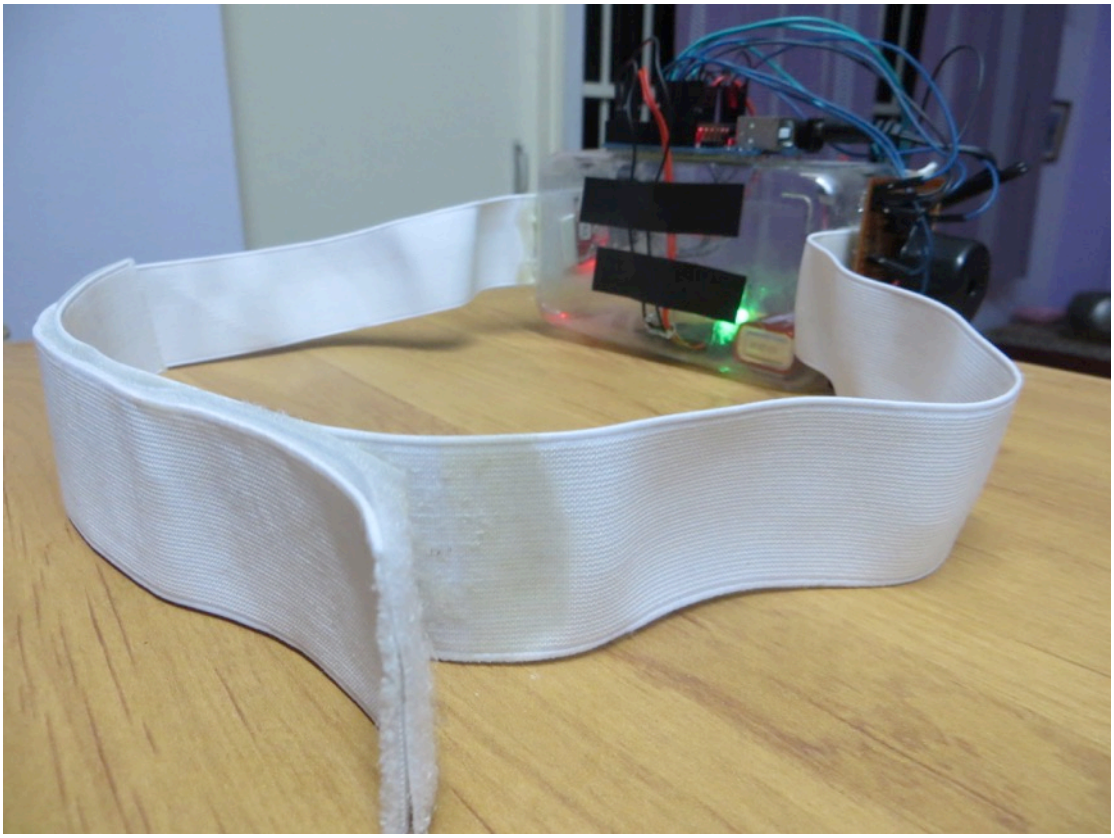
CIRCUIT DIAGRAM



WORKING OF SYSTEM



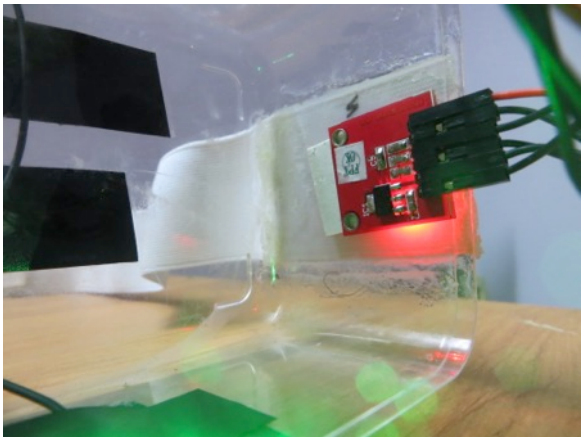
Finished device



Velcro strap and main frame

We have highlighted the fall-detection algorithm that we shall use in the previous section. This has been translated into code (as you will see in the code section) and burned into the microcontroller. The other modules and components are integrated with the MCU as you have seen in the circuit diagram. Here is the breakdown-

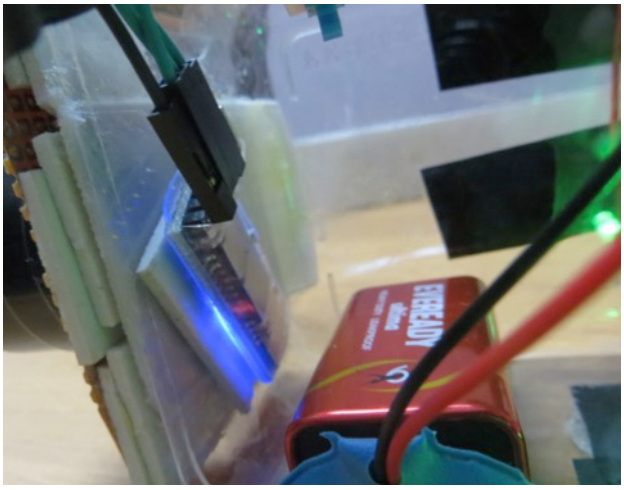
ADXL335: Connected to the analog pins 0, 1, 2 and 3 of the Arduino. They receive the analog values of acceleration and necessary computation is performed on them. The unit is powered with the +5V and the GND pins.



Accelerometer in my device

The unit is position perpendicular with the ground and fixed onto the inside of the wearable unit. It is calibrated in accordance to this position.

HC-05 Bluetooth Module: It is connected to the pins 10 and 11, which function as software serial pins of the Arduino. One line transmits data to the module while the other line receives data. The module is powered by 3.3V pin and the GND.



Bluetooth module attached to frame

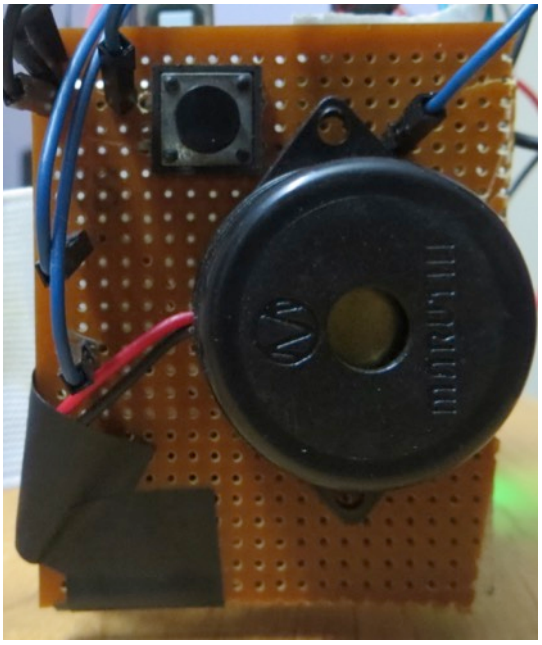
To connect the Bluetooth module for serial communication, we must identify and open both the **incoming** and **outgoing** ports. For this project I have opened the outgoing port with **TeraTerm** while the Processing sketch handles data from the incoming port.

RGB 3-color LED:

It consists of 4 pins, with one being a common cathode, while the other three power the red, green and blue LEDs respectively. We connect digital pin 2, 4 and 5 to R, G and B respectively. By using the **digitalWrite** functions we can change the color of the LED based on the state. Thus it is green when the user is table, blue when fall is sensed and red when fall is confirmed.

Piezo buzzer:

The buzzer is connected between the pin 9 and ground. We use the tone() function to generate a pwm pulse of specified frequency at that pin. When a fall is first sensed, we activate a 250Hz pulse and when confirmed, a 4000Hz pulse.



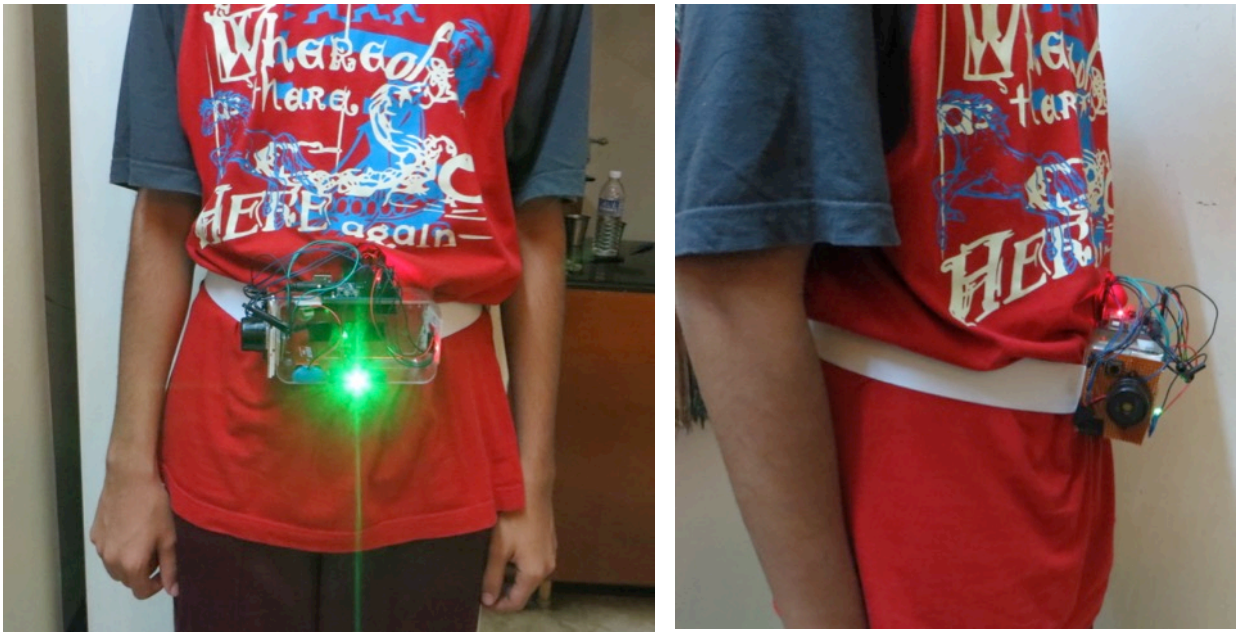
Piezo buzzer & switch circuit

Push button:

It is wired between pin 12 and ground, and pulled up by default. When it is pressed, the pin is grounded- triggering an event. The event in this case is the false alarm signal.

Implementation as wearable:

It can be used as a belt-type accessory tied around the waist with the help of the Velcro strap. In the prototype, frontward falls may damage the structure, although a compact, rugged casing can be used instead.

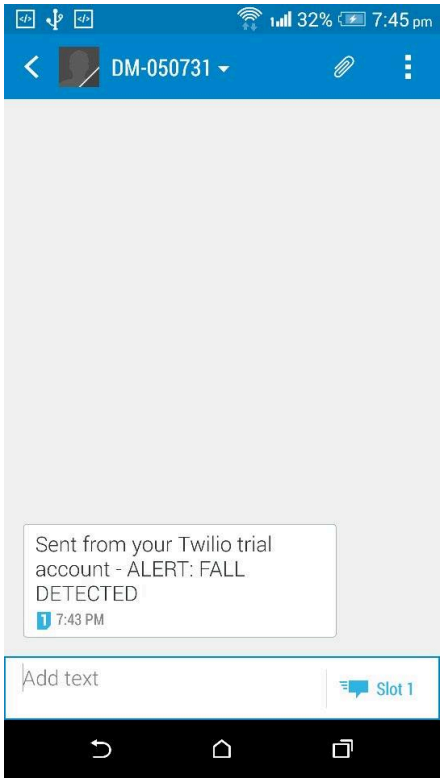


Belt-like wearable

Fall testing

PROCESSING + TEMBOO + TWILIO

As already explained in the previous sections, Temboo is a powerful portal to access APIs. Using their web service, I was able to obtain the processing code snippet required for integrating with a programmable virtual number on Twilio. The function RunSendSMSChoreo() initiates the response with the text I have supplied (In this case being a fall alert).



Text message received

ARDUINO CODE

The Arduino receives the data, performs calculations and outputs the data via Bluetooth serial connection. In addition it controls the LED and piezo buzzer.

```
/*

 Fall Detection, Alert and Action monitoring system
for the elderly-- An Intel IoT project by Sudharshan
Suresh
 December 2014

 The circuit:

 ADXL335 (Accelerometer)
 ----------------------
 analog 0: accelerometer self test
 analog 1: z-axis
 analog 2: y-axis
 analog 3: x-axis
 GND: ground
 +5V: vcc

 HC-05 (Bluetooth Module)
 ------------------------
 +3.3V: Vcc
 GND: ground
 Digital Pin 10(RX): TX
 Digital Pin 11(TX): RX

 Piezobuzzer
 -----------
 Digital Pin 9(PWM): +ve
 GND: -ve

 3-color LED
 -----------
 Digital Pin 2: RED
 Digital Pin 3(GND): Ground
 Digital Pin 4:GREEN
 Digital Pin 5:BLUE

Push Button
-----------
Connected between Digital Pin 12 and GND

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Arduino Uno @ COM18---- via Arduino IDE
```

```
Standard Serial over bluetooth link @ COM21
(Incoming)---- via Processing
Standard Serial over bluetooth link @ COM23
(Outgoing)---- via TeraTerm

*/

#include <SoftwareSerial.h>

// Accelerometer pin description
const int xpin = A3;                  // x-axis
const int ypin = A2;                  // y-axis
const int zpin = A1;                  // z-axis

float Xval, Yval, Zval;  // acceleration in g's
along each axis
float Anet, pitch, roll; // Net acceleration ( 1
when stationary), pitch and roll convey orientation
of accelerometer

unsigned long time = 0;  // variable to compute time
differences
unsigned long steps = 0; // number of steps taken
unsigned long steptime;  // variable to compute time
between steps

boolean stepcheck = false;
boolean firstfoot = false;

int state = 0;          // condition of subject ( 0
indicates stable and 5 indicates fallen )
int count =0;
int buttonrel = 0;
int button = 0;
int ALERT = 0;          // fall, perhaps?
int blinkstate = 0;

//below are the calibration values for the 3 axes,
such that in stable condition the values are (1g,
0g, 0g)
float xZero = 505;
float yZero = 510;
float zZero = 525;

SoftwareSerial mySerial(10, 11); //RX, TX


void setup()
  {
    mySerial.begin(9600);   // initialize the
software serial communication
```

```
    pinMode(12, INPUT_PULLUP); // Pull-up switch
connection
    pinMode(9, OUTPUT); // PWM output for
piezobuzzer
    pinMode(13, OUTPUT);
    digitalWrite(13, LOW);

    //RGB LED Pins
    pinMode(2, OUTPUT); //Red
    pinMode(3, OUTPUT); // Ground
    pinMode(4, OUTPUT); //Green
    pinMode(5, OUTPUT); //Blue

    digitalWrite(2, LOW);
    digitalWrite(3, LOW);
    digitalWrite(4, HIGH);
    digitalWrite(5, LOW);

    steptime = millis();
  }


void loop()
  {
    analogReference(EXTERNAL); // Enable Aref @ 3.3V

    // Compute acceleration values in g's
    Xval = (analogRead(xpin)- xZero)/102;
    Yval = (analogRead(ypin) - yZero)/102;
    Zval = (analogRead(zpin) - zZero)/102;
    Anet = sqrt(sq(Xval) + sq(Yval) + sq(Zval));

    // Pitch and roll caluculation
    pitch = atan(Xval/sqrt(pow(Yval,2) +
pow(Zval,2)));
    roll = atan(Yval/sqrt(pow(Xval,2) +
pow(Zval,2)));

    //convert radians into degrees
    pitch = pitch * (180.0/PI) - 90;
    roll = roll * (180.0/PI);

    // Serial transfer of required variables to
Processing
    mySerial.print(pitch);
    mySerial.print(",");
    mySerial.print(roll);
    mySerial.print(",");
    mySerial.print(state);
    mySerial.print(",");
```

```
    mySerial.println(float(steps));


    pedometer(); // Function to sense steps taken by
wearer
    distress();  // Manually operated distress-
signal can be sent via pushbutton hold
    fall_detect(); // FALL DETECTION FUNCTION

    delay(20); // Delay between passing values
  }


void pedometer()
  {
    stepcheck = !stepcheck; // To sense every
alternate iteration


if((Anet>1.2)&&(stepcheck==true)&&(state==0)&&(first
foot==false)) // Sense initial spike in acc. reading
      {
        firstfoot = true;
        steptime = millis();
      }

    if((Anet<0.9)&&(((millis()-
steptime)/1000)<2)&&(firstfoot == true)) // Sense
subsequent drop in acc. reading
      {
        steps++;
        firstfoot = false;
      }

    else if((((millis()-
steptime)/1000)>2)&&(firstfoot == true))       //
maximum permissible time between above 2 conditions
        firstfoot = false;
  }


void distress()
  {
    if ((state==0)&&(digitalRead(12) == LOW))
// detects button PRESS
    {
      button++;
      if(button>70)
// detects button HOLD
        {
```

```
          tone(9,4000);
// Generate piezotone for distress
          state = 42;
        }
    }

    else if((state==0)&&(digitalRead(12) == HIGH))
// detects button release
    {
      buttonrel++;
      if(buttonrel>10)
      {
        button = 0;
        buttonrel = 0;
      }
    }
  }


void fall_detect()
  {
    if(state ==0)
      {
        digitalWrite(2, LOW);
        digitalWrite(5, LOW);
        digitalWrite(4, HIGH);
      }

     if((Anet<0.8)&&(state==0)&&((abs(pitch)<60)&&(a
     bs(roll)<60)))               // Initial drop in
     acc. due to free-fall like condition
      {
        state = 1;
        time = millis();
      }

    if(state == 1)
      {
        if(Anet>1)                          // Increase
in acceleration due to impact of fall
        {
          state = 2;
          time = millis();
        }

        if(((millis()-time)/1000)>2)    // max.
permissible time between states 1 and 2
          state = 0;
      }

    if(state == 2)
```

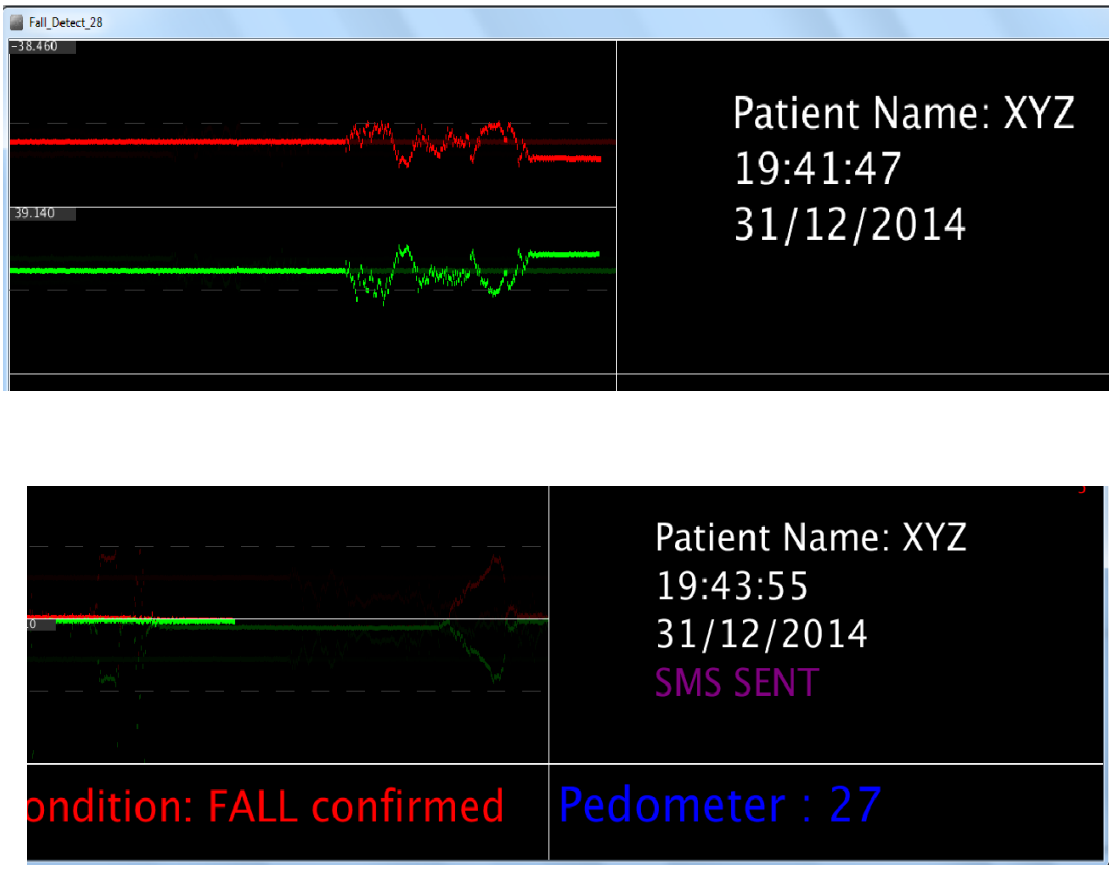
```
      {
        if((abs(pitch)>60)||(abs(roll)>60))    //
Senses a fall in any orientation by factoring pitch
and roll
          {
            state = 3;
            time = millis();
          }
        if((millis()-time)>100)         // max.
permissible time between states 2 and 3
            state = 0;
      }

    if(state == 3)
      { digitalWrite(4, LOW);
        digitalWrite(5, HIGH );
        if((abs(pitch)<60)&&(abs(roll)<60))    //
Detects getting-up and prevents false alarm
          {
            count++;
            if(count>20)
              {
                state = 0;
                count = 0;
              }
          }

        if(((millis()-time)/1000)>3)          // 3
second window after fall between states 3 and 4
          {
            state = 4;
            ALERT = 1;
            tone(9,250);         // Fall, perhaps?
          }
      }

    if((state ==4)&&(ALERT==1))
      {
        digitalWrite(5, (blinkstate) ? LOW: HIGH);
        digitalWrite(2, (blinkstate) ? HIGH: LOW);
        blinkstate = !blinkstate;

        if(digitalRead(12) == LOW)          //
Checks for button press to indicate false alarm
          {
            ALERT = 0;
            state = 0;
          }
        if(((millis()-time)/1000)>20)     // if
patient is unresponsive for more than 20 secs, fall
is confirmed-- necessary action taken via processing
```

```
            {
              state = 5;
              tone(9,4000);                // Generate
piezotone for distress
              digitalWrite(5, LOW);
            }
        }
    }
```

PROCESSING CODE

The processing sketch acts as the central monitoring unit for the patients. By graphical means, we represent-

- Graphs of pitch and roll and the angle data for the same.
- Patient bio-data, time and date.
- Condition of patient
- Number of steps taken (Pedometer)



Processing window

```
/*Processing sketch to process and interpret serial
data received via bluetooth module. Serves as the
central reference for doctors/ hospitals giving
alerts for possible falls, loss of consciousness and
also number of steps taken by patient. Also requires
active Internet service to send message via
3rd party server.
*/

// for serial data interpretation
import processing.serial.*;
Serial myPort;
```

```
/*Temboo is a cloud-based platform with 2000+
processes for APIs, databases, and more. The
Processing + Temboo library enables us to connect
with TWILIO,
  a cloud communication company that allows
developers to send/receive SMS and phone calls
programmatically. This solution is cost efficient,
effective and
  is an alternate to a bulky, power hungry GSM
module. */

import com.temboo.core.*;
import com.temboo.Library.Twilio.SMSMessages.*;

int graphPara = 2;    // Graph parameters -- pitch
and roll

float[] values   = new float[graphPara+2];
//total of 4 serial variables
int[]   min      = new int[graphPara];
// minimum value to graph
int[]   max      = new int[graphPara];
// max value to graph
color[] valColor = new color[graphPara];
// graph line color

float effHeight;          // effective height of
graph window

int xPos = 1;             // horizontal position of
the graph
String status = "Stable";
int flag = 0;

// Create a session with your Temboo account details
TembooSession session = new TembooSession("suddhus",
"myFirstApp", "412da22525b64a10bd01b819434f15de");


void setup()
{
  size(1260, 400);        // Window dimensions
  effHeight = 300 / graphPara;

  // List all the available serial ports:
  printArray(Serial.list());
  // Port [0] is generally found to be the incoming
bluetooth serial line, if no USB COM is active:
  myPort = new Serial(this, Serial.list()[0], 9600);
  // don't generate a serialEvent() until you get a
newline character:
```

```
  myPort.bufferUntil('\n');

  background(0);


  // initialize pitch:
  values[0] = 0;
  min[0] = -90;
  max[0] = 90;
  valColor[0] = color(255, 0, 0); // red

 // initialize roll:
  values[1] = 0;
  min[1] = -90;
  max[1] = 90;
  valColor[1] = color(0, 255, 0); // green

  // state variable
  values[2] = 0;

  // steps variable
  values[3] = 0;
}


 // empty draw() loop
void draw()
{
}


// Everything occurs in the serialEvent() loop
void serialEvent(Serial myPort)          // get the
ASCII string:
{
  fill(0);
  stroke(255);
  rect(630,0,630,300);

  textSize(40);
  fill(255);
  text("Patient Name: XYZ", 750,80);

  String time = hour() + ":" + minute() + ":" +
second();
  String date = day() + "/" + month() + "/" +
year();
  text(time , 750, 130);
  text(date, 750, 180);

  textSize(45);
```

```
fill(0);
rect(0,300,630,100);

//Check state of patient
if(status == "Stable")
  fill(0,255,0);
else if(status == "FALL sensed")
  fill(0,0,255);
else
  fill(255,0,0);

//display appropriate state message
String condition = "Condition: " + status;
text(condition, 0, 360);

textSize(20);
text(int(values[2]), 1230, 20);

// Send SMS if fall Confirmed/ Distress
if((values[2] ==5)||(values[2]==42))
{
  if(flag==0)
    runSendSMSChoreo();
  textSize(40);
  fill(128, 0, 128);
  text("SMS SENT", 750, 230);
  flag = 1;
 }

textSize(50);
fill(0);
rect(630,300,630,100);

//Display number of steps taken
fill(0,0,255);
String steps = "Pedometer : " + int(values[3]);
text(steps,640,360);

String inputString = myPort.readStringUntil('\n');
noFill();
stroke(255);
textSize(12);
rect(0,0,630,300);

if (inputString != null)
{
  // trim off any whitespace:
  inputString = trim(inputString);
```

```
    // splits the serial string on the delimiters
and assigns it to a float array, here the delimiter
can be a ,--[comma] ,  --[space] , \t--[tab]
    values = float(splitTokens(inputString, ",
\t"));

    //check if you recieved the whole packet and map
the data
    if (values.length >= graphPara)
    {
      for (int i=0; i<graphPara ; i++)
      {
        fill(50);
        noStroke();
        rect(0, effHeight*i+1, 70, 12);
        fill(255);
        text((values[i]), 2, effHeight*i+10);


        // map to the range of effective screen
height:
        float mappedVal = map(values[i], min[i],
max[i], 0, effHeight);

        // draw lines:
        stroke(valColor[i]);
        line(xPos, effHeight*(i+1)- mappedVal + 2,
xPos, effHeight*(i+1) - mappedVal-2);

        // draw dividing line:
        stroke(255);
        line(0, effHeight, 630,effHeight);

        //dotted line (origin):
        stroke(64,64,64);
        for( int j=0, k=0;j<=610;j=j+20)
        {
          if((k%2)==0)
          {
            line(j,75,j+20,75);
            line(j,225,j+20,225);
          }
          k++;
        }

      }

      // reset graph if at the edge:
      if (xPos >= 630)
      {
        xPos = 0;
```

```
        // erase graph window with translucent black
        fill(0,200);
        rect(0,0,630,300);
      }

      else
        xPos++;         // increment graph position

    }
  }


  print(values[2]);
  if(values[2]==5)
    status = "FALL confirmed";

  else if(values[2] ==4)
    status = "FALL sensed";

  else if(values[2] == 0)
    status = "Stable";

  else if(values[2] == 42)
    status = "Distress";
}

// Function to generate SMS
void runSendSMSChoreo()
{
  // Create the Choreo object using your Temboo
session
  SendSMS sendSMSChoreo = new SendSMS(session);

  // Set credential
  sendSMSChoreo.setCredential("AlertSMS");

  // Set inputs

  // Run the Choreo and store the results
  SendSMSResultSet sendSMSResults =
sendSMSChoreo.run();

  // Print results
  println(sendSMSResults.getResponse());

}
```

SAMPLE TEST DATA

Activity	No. of tests	Fall detected	Not detected
Front fall	5	5	0
Backward fall	5	5	0
Side right fall	5	5	0
Side left fall	5	5	0
Sitting	5	0	5
Jumping	5	0	5
Lying down	5	1	4

$$\textbf{Accuracy} = \left(\frac{\mathbf{34}}{\mathbf{35}}\right) \times \mathbf{100} = \mathbf{97.15\%}$$

REFERENCES

http://www.who.int/ageing/publications/Falls_prevention7March.pdf

http://www.futurehealthsystems.org/publications/heterogeneity-in-self-assessed-health-status-among-the-elder.html

http://www.helpageindia.org/about-us.php

http://www.gjmedph.org/uploads/r2-vo2no4.pdf

http://bildr.org/2011/04/sensing-orientation-with-the-adxl335-arduino/

http://physics.rutgers.edu/~aatish/teach/srr/workshop3.pdf

http://www.arduino.cc/

https://temboo.com/

https://www.twilio.com/