

**IMPLEMENTING FALL DETECTION AND MONITORING ECG WITH WIRELESS
SENSOR NETWORKS**

by

Onur Dündar

Submitted to the Department of Computer Engineering

in partial fulfillment of the requirements

for the degree of

Bachelor of Science

in

Computer Engineering

Boğaziçi University

June 2010

Contents

ABSTRACT.....	3
1.) INTRODUCTION	3
2.) PREVIOUS WORK WITH IMOTE2.BUILDER	4
2.1) Introduction to Imote2.Builder	4
2.2) Introduction to IMB400 Sensor Board	4
2.3) Implementation of Image Sensor Process.....	5
2.3.2) Formatting Imote2.builder	5
2.3.2.1) Windows Environment for Imote2	5
2.3.2.2) Linux Environment for Imote2 (Xubuntos).....	5
2.3.2.3) Starting To Format Progress.....	6
2.3.3) Implementing Driver for Image Sensor on .Net Micro Framework	8
2.3.3.1) Collecting Data Sheets.....	8
2.3.3.2) Implementing Camera Chip Driver with I2C Bus Property	9
3.) IMPLEMENTING FALL DETECTION AND MONITORING ECG WITH SHIMMER	12
3.1) Introduction to SHIMMER.....	12
3.2) Available SHIMMER Applications.....	12
3.2.1) Shimmer Windows Bootstrap Loader.....	12
3.2.2) Shimmer Displayer	13
3.3) Introduction to BioMobius	14
3.4) Implementation Fall Detection	15
3.5) Monitoring ECG Values	18
4.) GOALS.....	21
5.) FUTURE WORK	21
REFERENCES	22

ABSTRACT

Project Name : Implementing Fall Detection and ECG Monitoring With Wireless Sensor Networks.

Keywords : IMB400, Imote2.builder, Shimmer, ECG (electrocardiogram), Accelerometer, image sensor, TinyOS, NesC, .Net Micro Framework.

Summary : This project is all about the wireless sensors. In the project development process two different wireless sensor kit used. The first one is named as Imote2.builder and second one is Shimmer. We made two different works for each sensor kit. Imote2.builder has external sensor kit, we work on the external sensor kit IMB400 which has an image sensor, we tried to make it work with various ways but we had to stop because of insufficient hardware and datasheet about the IMB400. We also made some implementation about Shimmer sensors. At the first stage we implemented a fall detection algorithm with Shimmer's embedded accelerometers and for a second work we attached the external electrocardiogram sensor and monitored the data on another device.

In this document, information will be provided in terms of progress, research and development.

1.) INTRODUCTION

This document is written for Cmpe 492 Senior Project course given by Computer Engineering Department at Boğaziçi University. This project is mainly aimed to make Imote2.builder wireless external image sensor work and implementing fall detection algorithm with accelerometer data and monitoring ECG values which came from Shimmer wirelessly.

2.) PREVIOUS WORK WITH IMOTE2.BUILDER

2.1) Introduction to Imote2.Builder

Imote2.builder is an advance wireless sensor node platform. It is built around the low power PXA271 XScale CPU and also integrates an 802.15.4 compliant radio. The design is modular and stackable with interface connectors for expansion boards on both the top and bottom sides, communicates with external boards with those connectors. There are kinds of boards which can be connected on the Imote2.builder such as power board which supplies power to the board, ITS400 sensor board which includes humidity, temperatures, light sensors on it.

Imote2.builder kit has SDRAM, SRAM and FLASH on it, users can program the board as their necessity. Developing firmware for this board can be done on Microsoft .Net Micro Framework, but if user formats board and installs Tinyos bootloader it would be able to develop firmware on TinyOS with nesC.

2.2) Introduction to IMB400 Sensor Board

IMB400 board is an external board of the Imote2.builder sensor kit, connected on the bottom expansion of Imote2.builder. This board includes image sensor, pir motion sensor, microphone and speaker. To use any of the sensors it is required to implement firmware of that sensor. The main problem with this board is the absence of the firmware of image sensor on .Net Micro Framework but they have a firmware for TinyOS version. Progress with this board is about to write a firmware for image sensor or formatting the Imote2.builder to install TinyOS bootloader and programming the image sensor with existing firmware on Imote2.builder.

2.3) Implementation of Image Sensor Process

2.3.2) Formatting Imote2.builder

2.3.2.1) Windows Environment for Imote2

To work with Imote2 you need to install cygwin environment. Cygwin is a Linux-like environment for Windows. It includes the TinyOS 1.1x to work with wireless sensor networks so it is also compatible for Imote2 nodes which have TinyOS bootloader. Although we know that our wireless sensor is not a TinyOS compatible device because it includes .Net Micro Framework bootloader. Our purpose to install cygwin to use Imote2.builder after format process in Windows environment with cygwin.

Also we installed the necessary tools in cygwin such as NesC, TinyOS. We installed NesC because NesC is an extension to the C programming language designed to embody the structuring concepts and execution model of TinyOS.

In addition to above requirements and installation we also need to install the Imote2 driver on Windows and this is provided by the Imote2 community, all installation instructions are included in the Imote2 community.

2.3.2.2) Linux Environment for Imote2 (Xubuntos)

In order to prepare Linux environment for Imote2 we also need to install TinyOS to Linux OS. But in this situation it is more easy than preparing environment for Windows because there is a live cd to install Linux with TinyOS full configuration. Installing this version of the Linux is enough to work with Imote2 but there are missing packages in this version of Linux because it

can become impossible to install some new application in this version of Linux. This version of Linux is named as Xubuntos.

Maybe it can be another solution to follow the installation instructions in TinyOS web site but this became challenging for new starters. We followed the installation instructions but it is not installed correctly so we installed Xubuntos again.

2.3.2.3) Starting To Format Progress

In order to format Imote2.builder and installing TinyOS bootloader in it, we need a JTAG interface board and compatible JTAG cable. The JTAG interface board is an external board of Imote2.builder and it is connected from bottom connectors of the Imote2.builder board. It is also named as Intel Mote 2 Debug Board. Its picture is attached to below



Figure 2.1) Intel Mote 2 Debug Board

JTAG interface board has 20-Pin JTAG cable connection and a serial USB connection on it. JTAG interface board is already given by the Imote2.builder kit but the JTAG cable was missing and it is too hard find the compatible JTAG cable from market. Compatible JTAG cable was Intel JTAG but it is not on sale. We found an alternative JTAG cable which is Amontec's JTAGKey. Amontec JTAGKey cable fits to the interface board it also have 20 pin input and it is

connected to computer with USB cable. We also need to install drivers of cable and urJTAG program to work properly with Amontec JTAGKey. urJTAG is necessary to configure JTAGKey and check the cable. urJTAG installed successfully and it detected the cable.

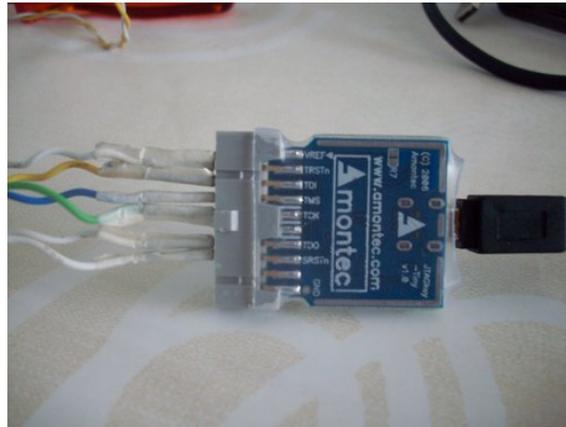


Figure 2.2) Amontec JTAGKey

Last step is to formatting the Imote2.builder with existing software JFlashMM. JFlashMM was implemented to restore Imote2 device with JTAG cable, it makes restore process with erasing indicated addresses on flash memory and restoring the TinyOS boot loader. We plugged the JTAGKey to the Imote2.builder and ran the JFlashMM but software did not work, it gave an error with screening out “Error, unable to find parallel port”. We understand that JFlashMM need parallel JTAG cable and Amontec JTAGKey is not a parallel cable.

We also tried to get work Amontec JTAGKey on Xubuntos and Ubuntu 9.10. There has been two necessity to function Amontec JTAGKey on Linux. One of them is OpenOCD software and other is installed TinyOS in related Linux. The Open On-Chip Debugger (OpenOCD) aims to provide debugging, in-system programming and boundary-scan testing for embedded target devices it is necessary to format Imote2.builder. There has been two different error on two

different Linux versions. Xubuntos does not have necessary packages to install or upgrade the Linux so OpenOCD could not be installed. Ubuntu 9.10 have all properties to install any new software and we installed OpenOCD installed successfully and the JTAGKey is recognized but there was an error about TinyOS so it could not recognized the Imote2.builder to format its flash. This became an end point because we needed an parallel JTAG cable to format in Windows and a stable version of Linux which TinyOS installed correctly.

2.3.3) Implementing Driver for Image Sensor on .Net Micro Framework

Since we cannot format Imote2.builder we started to investigate how we could implement a driver for IMB400 Camera. In order to implement driver we have seek all information which need for implementing the driver.

Before starting to implement driver for IMB400 camera chip we need to get ready for development environment. At first step we installed the drivers for Imote2.builder with using the Imote2.builder CD. Also we need a development environment so we installed MS Visual Studio 2005 and .Net Micro Framework 2.0 installed, we got ready to implement code with .Net Micro Framework in Visual Studio 2005 with deploying the code in Imote2.builder.

2.3.3.1) Collecting Data Sheets

We have limited information about the IMB400 board so we started to collect all necessary data. At first step we need to know the connector's assignment and what kind of signal they carry. As I mentioned before IMB400 camera is connected bottom side of the Imote2. We could learn the Imote2 pin signal properties from Imote2 datasheet but IMB400's are not available so we

requested them from producer of Imote2, they replied that the pin descriptions of Imote2's bottom side matches with the IMB400 board's connector pins.

But we also need more information about the pin assignment we could reach the camera chip's datasheet. It includes all information about image sensor, how we get gray picture, color picture with sending correct signal to the chip. In order to send correct signal to the chip we require the chip's pins assignment with the IMB400's connector pins because signals are controlled by Imote2 and we send and control signals (clock, data) with Imote2. We did not have any document about that assignment so requested from producer (Crossbow). It is waited a long time to get answer and finally the producer told us that they cannot provide that information to us.

Without necessary information we could not proceed anymore so we tried another method to implement driver in .Net Micro Framework. We learned that camera chip is compatible for I2C bus communication from camera chip datasheet (Camera Chip Name is OV7670).

2.3.3.2) Implementing Camera Chip Driver with I2C Bus Property

The I2C bus was designed by Philips in the early '80s to allow easy communication between components which reside on the same circuit board. Some sensors of the Imote2 such as temperature, humidity and light sensor drivers are implemented in .Net Micro Framework and using I2C bus.

At first step I survey the drivers already implemented on .Net Micro Framework and I wrote below code to test camera chip if it works on I2C bus. We told the code step by step and what was the required output from the device.

```
➔ I2CDevice i2c = new I2CDevice(new I2CDevice.Configuration(0x42, 400));
```

At this step of the code we introduce the camera chip on I2C bus with giving device address (0x42), in I2C bus communication all devices have an address. Also we introduced the bus frequency (400) these information learned from camera chip datasheet.

```
➔ I2CDevice.I2CWriteTransaction command = i2c.CreateWriteTransaction(read);
```

In this line of the code we initialize a command. We are initializing the related register address with creating this command. “read” is the address of the related register, it is a byte variable. The address taken from camera chip datasheet.

```
➔ I2CDevice.I2CReadTransaction data = i2c.CreateReadTransaction(new  
byte[1]);
```

At this point we introduce a transaction which reads the register which is told by command. “data” variable is an buffer array and reads the register value into the first place in the buffer array.

```
➔ I2CDevice.I2CTransaction[] xAction = { command, data };
```

This line creates an I2C transaction and takes the above read and write transaction to execute them respectively in below line.

```
➔ i2c.Execute(xAction, 100);
```

This is the line which executes both write and read transactions. In I2C communication if the user wants to read a register it gives the register address to the write transaction command and reads it by read transaction variable. If the user only wants to write some value into the related register it gives the register address and the value to the write transaction variable.

```
➔ Debug.Print("address is " + 0x42 + " and value is " +  
    data.Buffer[0].ToString());
```

At this point we just read the value of the register and print but it never read any value. We want to read the product id of the chip which is kept in register addressed 0AH and its value is 76H, but always read 0. I tried all other read only registers such as register 0BH default value is 73H and read value is 0.

We tried same code on light and temperature sensor and got the register value but could not read camera chip's registers. We send emails to the imote2 community and got answers:

“There is a missing pull up resistor that prevents reading anything back from the sensor via the i2c bus and this chip isn't connected to the same i2c bus as the rest of the sensors. It is connected to a pair of gpio pins. It is possible to modify the board so that it is on the main i2c bus.”

As we understand from the above answer it is nearly impossible to write driver with I2C but it has been suggested that removing two resistors (R1 and R4) from IMB400 and populating other two (R8 and R11) would make it work but it is not tried before.

Without reading any value from registers it would make no sense to implement any driver so this approach to solve driver problem of the camera chip of the IMB400 board ends.

3.) IMPLEMENTING FALL DETECTION AND MONITORING ECG WITH SHIMMER

3.1) Introduction to SHIMMER

SHIMMER a small sensor platform well suited for wearable applications. The integrated 3-axis accelerometer, large storage, and low-power standards based communication capabilities enable emerging applications in motion capture, long-term data acquisition, and real-time monitoring.

SHIMMER sensor platforms have extensible boards that are attached on small connectors of SHIMMER platform. Kinematics (3-axis MEM's Gyro), 3 Lead ECG, Infrared Motion Detection and etc are existing boards. In the following process we used ECG sensor to monitor ECG values wirelessly and 3-axis Accelerometer which is on board sensor, in order to implement fall detection.

3.2) Available SHIMMER Applications

There are some applications for programming SHIMMER and displaying values which read from SHIMMER nodes.

3.2.1) Shimmer Windows Bootstrap Loader

To program the SHIMMER there is an application supplied with Shimmer USB stick, programs existing or implemented firmware in SHIMMER. Firmware implements in TinyOS and after compiling and building the firmware it creates “.ihex” files and opening this “.ihex” file with this application and clicking the “program” button is enough to program SHIMMER with desired firmware.

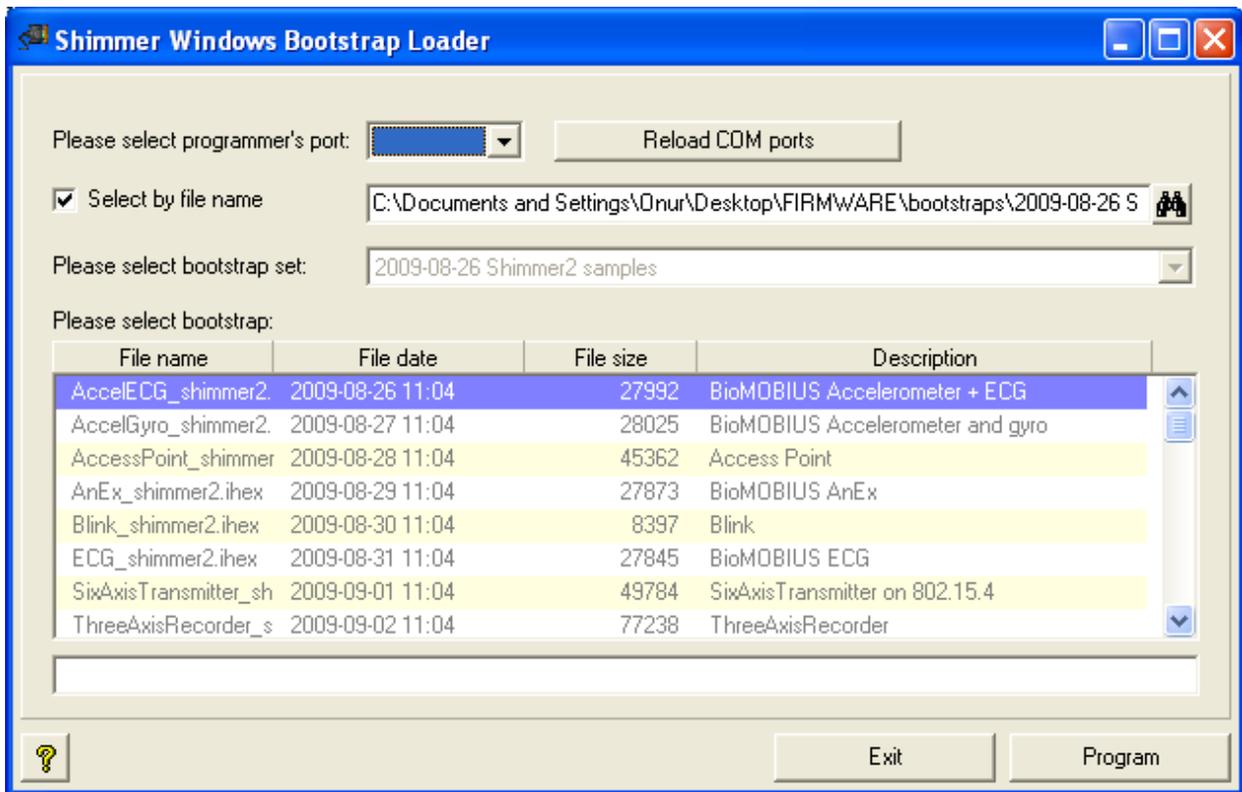


Figure 3.1) Shimmer Windows Bootstrap loader

3.2.2) Shimmer Displayer

Shimmer displayer has a left panel which has 4 buttons, they control connection and transmission and there are 6 small windows. These windows are named as accelerometer x, y, z and gyroscope x, y, z whatever the coming signals are.

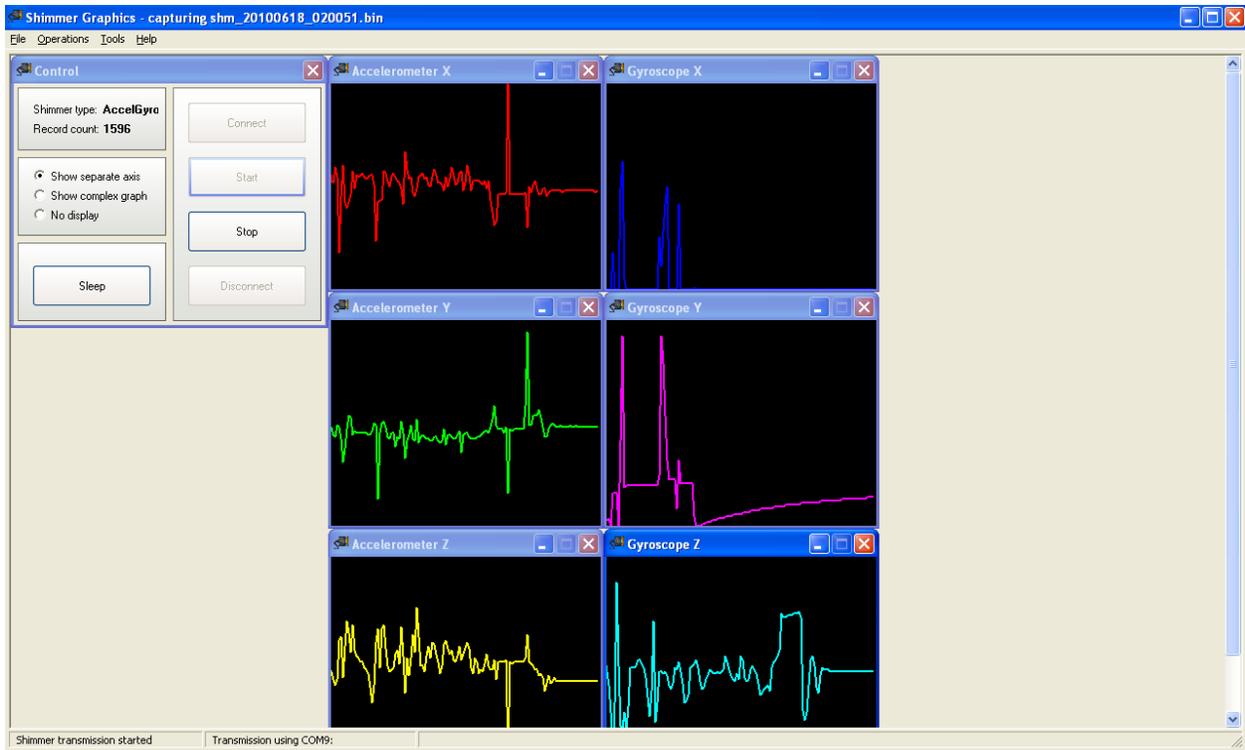


Figure 3.2 Example Shimmer Displayer Screenshot with Accelerometer and Gyroscope Values

3.3) Introduction to BioMobius

The BioMOBIUS research platform is a rapid biological signal application development platform, designed to create robust applications for large scale deployment with minimal effort. BioMOBIUS provides support for the SHIMMER wireless sensor platform and other third party hardware and sensor devices. BioMOBIUS research platform includes EyesWeb application to produce patches which are analyzing and developing signals with respect to user's desire for wireless sensors. There is a GUI Designer for to make interfaces for EyesWeb patches. With using GUI Designer it is possible to give input signals to related EyesWeb patches and displaying outputs from EyesWeb patches. EyesWeb patches have file extension "eywx" and GUI Designer has file extension "eywrad".

3.4) Implementation Fall Detection

SHIMMER sensor platforms have a 3-Axis accelerometer on board. In order to implement fall detection we used this accelerometer.

In order to implement fall detection we should use the human readable accelerometer values from Shimmer. There is an EyesWeb patch and documents about calibration of the read values from Shimmer. With using the “AccelerometerCalibrationShimmer.eywx” patch we implemented the algorithm with getting last calibrated values and making a simple algorithm to detect fall. Pseudo code of the algorithm is showed below.

```
while true  
read calibrated accelerometer's x , y and z values  
x=x/9.8g,y= y/9.8g,z=z/9.8g  
v=sqrt(pow(x)+pow(y)+pow(z))  
if v>1.6 g  
fall detected
```

The threshold value is detected with testing it. The minimal condition to detect fall for this algorithm in Shimmer is getting over 1.6g.

This algorithm is implemented in EyesWeb with using arithmetic patches. The algorithm model can be seen below figure. It is a screenshot from EyesWeb.

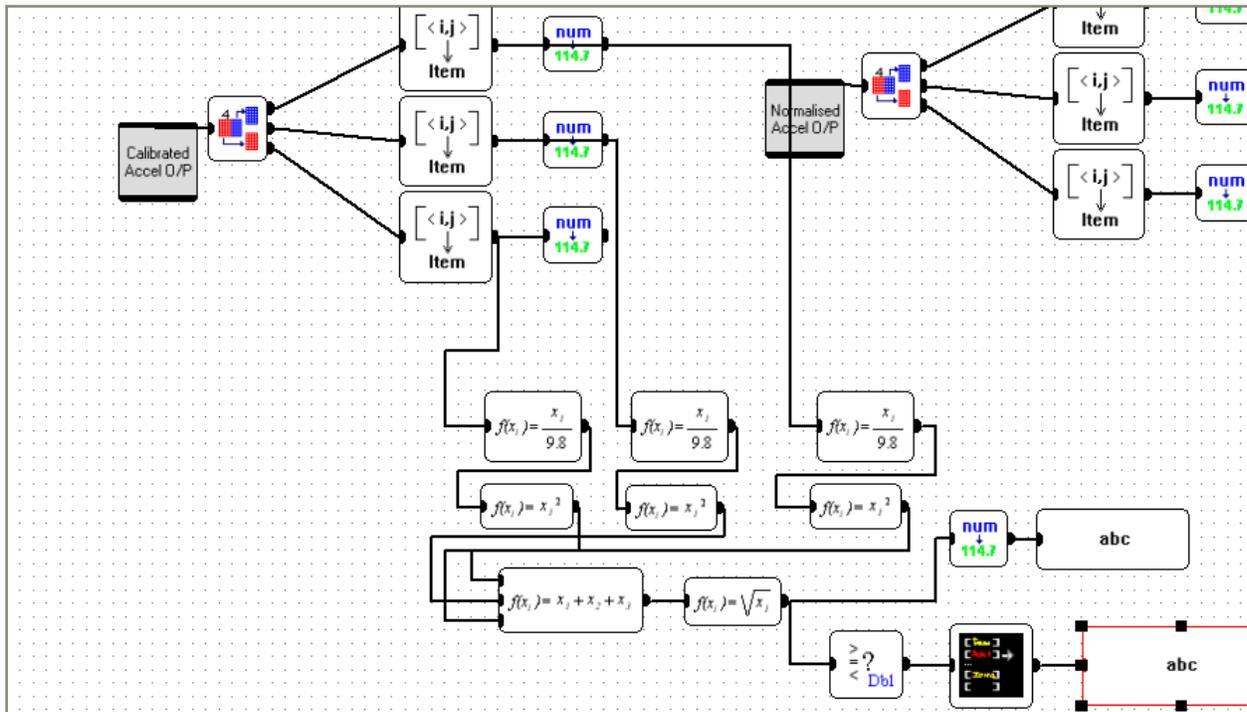


Figure 3.3) Fall Detection Algorithm implementation on EyesWeb with using simple functional blocks.

After implementing the EyesWeb patch we also used the GUI Designer to work with this patch and show the “Fall Detection” output and accelerometer X, Y and Z values. After completing the interface we imported the EyesWeb patch and connected the buttons and outputs to the patch’s related blocks.

In figure 3.4 we showed our GUI Design. With clicking “Connect” button patch works behind the scene and connects to the Shimmer and gets values, “Disconnect” button makes the opposite.

There is also three graphical and one text output are existing. Text output shows situation “Fine” or “Fall Detected” other three shows accelerometer values which read from accelerometer.

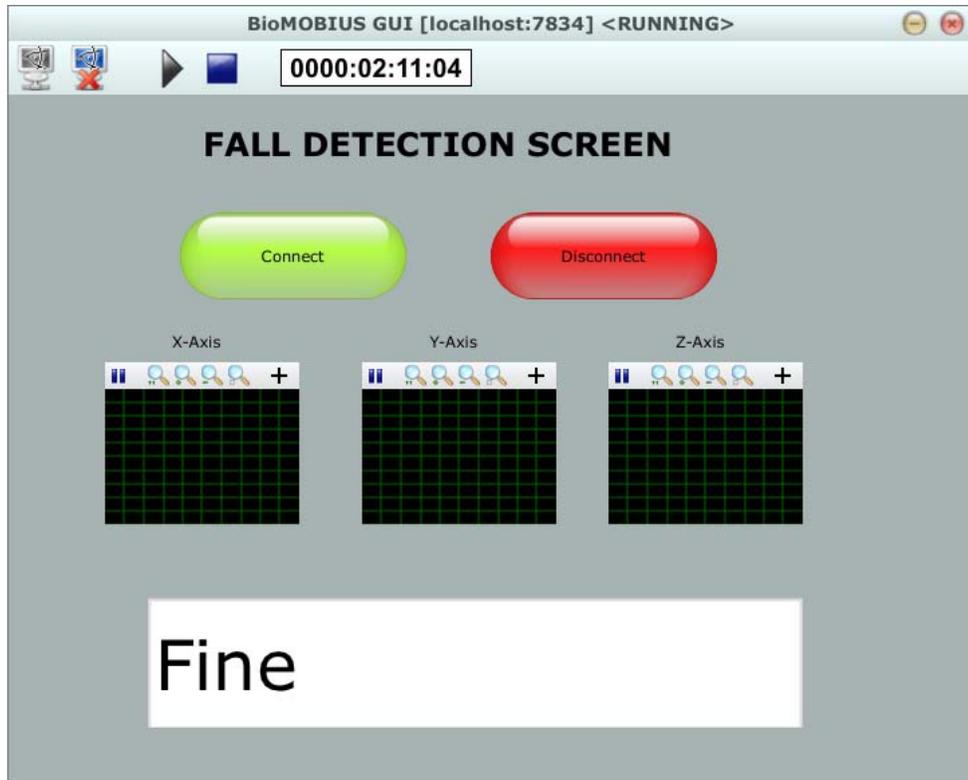


Figure 3.4) GUI Interface when people stable or not falling situation with Shimmer

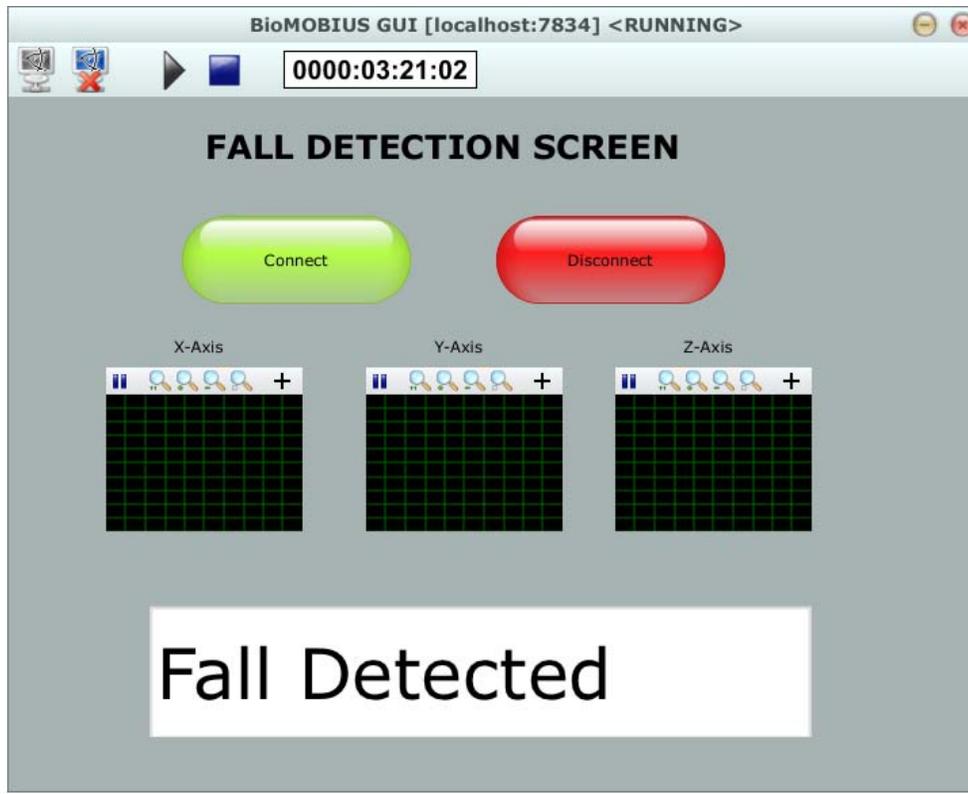


Figure 3.5) GUI Interface when Fall Detected

3.5) Monitoring ECG Values

Monitoring the ECG values was made by three steps. At first step we attached the ECG board on the Shimmer board, to get the ECG values we need to program shimmer with related firmware. Hence we used Shimmer Windows Bootstrap Loader to program Shimmer with “AcceECG.ihex”. This firmware gets accelerometer values and ECG values together. As a second step we connected cables to the human body like figure 3.4.



Figure 3.6) Shimmer with ECG

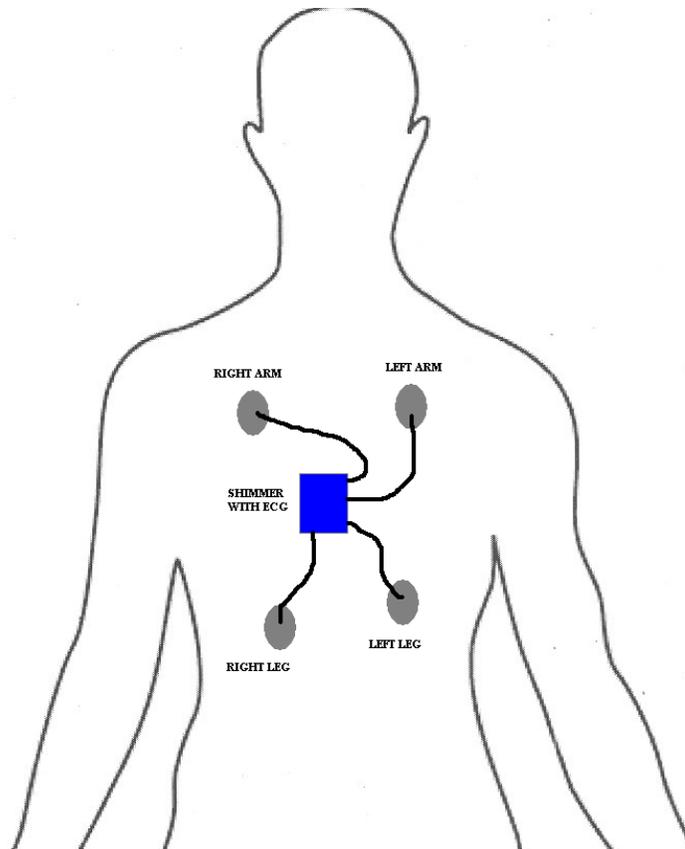


Figure 3.6) ECG Cable Positions on Human Body

For the last step we observed the ECG values with using Shimmer Display. After opening the Shimmer Displayer pressing “Connect” and “Start” buttons respectively will be enough to show accelerometer and ECG values on screen. There are accelerometer values on the screen because our firmware includes both accelerometer and ECG values. ECG values are displayed on Gyroscope X and Y small windows in Shimmer Display figure 3.7.



Figure 3.7) Shimmer ECG Values read by Shimmer Display

4.) GOALS

The goal of this project to get work image sensor of the wireless sensor Imote2. Also fall detection and ECG monitoring with another sensor. Researching all possible methods to make work IMB400 image sensor such as trying to format device or implementing a new driver for camera chip is the most challenging part of the project.

Also getting used to with Shimmer Wireless Sensors and understanding its development environments are the main goals of the project. In order to find out Shimmer capabilities we study BioMOBIUS and got the idea behind BioMOBIUS.

5.) FUTURE WORK

There are some future works for both Imote2 and Shimmer work. In order to format Imote2 it is possible to find a parallel cable and trying it with related JFlashMM software and both TinyOS and OpenOCD could be well installed on a Ubuntu and formatting process could be done with Amontec JTAGKey but restoring process could not be done because JFlashMM software only works on Windows.

Also It is possible to implement more efficient fall detection algorithms with EyesWeb. It is so hard to implement complex algorithm with only accelerometer because of insufficient patches but this could be override with implementing new patches. Also it is possible to calibrate gyroscope values and implementing some existing algorithms on the EyesWeb. EyesWeb's capabilities are limited but it has a SDK to implement new patches to use development process.

REFERENCES

- 1.) Imote2 Datasheet
- 2.) Imote-Community Yahoo Group
- 3.) www.tinyos.net
- 4.) www.shimmer-research.com
- 5.) SHIMMER User Manual