A WIRELESS COMMUNICATION SYSTEM FOR PERVASIVE HEALTH-INFORMATION MONITORING

By

JASMINE HIREMATH

Bachelor of Engineering in Computer Science

VTU

Belgaum, KA, India

2008

Submitted to the Faculty of the Graduate College of the

Oklahoma State University in partial fulfillment of

the requirements for the Degree of

MASTER OF SCIENCE

December 2014

A WIRELESS COMMUNICATION SYSTEM FOR PERVASIVE HEALTH-INFORMATION MONITORING

Thesis proposal approved:

Thesis Advisor

Dr. Tingting Chen

____

Dr. Johnson Thomas

____

Dr. David Cline

____

Name:	JASMINE HIREMATH
Date of Degree:	DECEMBER 2014
Title of Study:	A WIRELESS COMMUNICATION SYSTEM FOR PERVASIVE HEALTH-INFORMATION MONITORING
Major Field:	COMPUTER SCIENCE

ABSTRACT

As many areas of the medical field are becoming more and more interested in telemedicine and outpatient care, we focused on the monitoring of health as it could benefit greatly from body sensor network (BSN) technology. We explore some of the hardware sensor interface involved in monitoring procedure into a portable system, specifically for system integration with a smartphone. The main contribution of this work is to define norms for hardware interface with an Android-based smartphone/tablet: using dedicated embedded sensor nodes have a low volume as one of the design objectives, as they have limited processing and memory capacities. Thus, special operating systems such as TinyOS and companion programming languages have been developed. Due to the low-cost and low-volume features, the future widespread use of such nodes is expected. A Bluetooth enabled Android device is used for initiating our monitoring mobile application and Bluetooth enabled sensors provide the input to the Android device. We have developed an Android app; the communication system can be potentially used by a remote monitoring setup without the presence of a medical profession, and even streamline in-patient monitoring process with actual precision for the medical measurements. ***Keywords- Android; TinyOS; BSN; Bluetooth***

TABLE OF CONTENT

Chapter Page

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

A *body sensor network* (BSN), also referred to as a *wireless body area network* (WBAN), is a wireless network of wearable computing devices. The network consists of several miniaturized body sensor units (BSUs) together with a single body central unit (BCU). The development of WBAN technology is based on the idea of using wireless personal area network (WPAN) technologies to implement communications on, near, and around the human body. Later, the term "BAN" came to refer systems where communication is entirely within, on, and in the immediate proximity of a human body.

The quick growth in physiological sensors, low-power consuming integrated circuits, and wireless communication has enabled a new generation of wireless sensor networks. This development can be seen in monitoring traffic, crops, infrastructure, and health etc. The body area network field is an interdisciplinary area, allowing inexpensive and continuous health monitoring with real-time updates of medical records through the Internet. This area relies on the feasibility of implanting very small biosensors on the human body that are comfortable and that don't impair normal activities. The implanted sensors on the human body will collect various physiological changes in order to monitor the patient's health status no matter their location. The information will be transmitted wirelessly to an external processing unit i.e. Android tablet here. This device will transmit all information in real time to the doctors on patient's selection.

1.2 WHY BODY SENSOR NETWORK

Body Sensor Networks are a specific category of wireless sensor networks intended to operate in a pervasive manner for health monitoring applications. Major concentrations of research on Body Sensor Networks (BNSs) are issues such as power optimization, battery life performance and radio design. Concerns about body worn sensors are issues such as usability, durability, robustness, how well the sensor ‘fits’ in with the application and reliability and security of the data.

1.3 WHY HEALTH MONITORING IN BSN

The factors like the increase in the average life span, health costs, advances in miniaturization of electronic devices, sensing, battery and wireless communication technologies have led to the development of Wireless Body Area Networks (WBANs). WBANs consist of smart miniaturized devices (motes) that are able to sense, process and communicate. They are designed such that they can be worn or implanted, and monitor physiological signals and transmit these to specialized medical servers through electronic devices like smart phones/tablets, without much interference to the daily routine of the patient.

1.4 WHY ANDROID APP

The challenge we are particularly interested in relates to the ability for rapid development of applications that make use of a Body Sensor Network (BSN).Especially looking at applications in scenarios where it is required to setup networks of sensors quickly. In recent years, the Android platform has become a de-facto standard for different types of mobile devices from manufacturers. Android is one of the most popular smartphone a platforms at the minute and the popularity is even raising. It is one of the most open and flexible platforms providing software developers easy access to phone hardware and a rich software API. Android mobile devices not only shows as a powerful computing and communication mobile device of choice, but they also comes with a rich set of embedded sensors. Collectively, these sensors are enabling new applications across a wide variety of domains, such as healthcare, social networks, safety, environmental monitoring and transportation.

1.5 RESEARCH OBJECTIVE

The main objective of this project is to build a communication system, where the Body Sensor Network (BSN) interacts with the Android smartphone/tablet through Bluetooth. A Body Sensor Network (BSN) consists of the sensor nodes called motes. These are generally placed on the human body and the data collected from the motes is passed to an Android App through Bluetooth. The Android App is responsible for the data transfer to the doctor/health status update database server via internet, done through the email system embedded in the app.

CHAPTER 2

REVIEW OF LITERATURE

Mobile applications are developed so that things can be done at the click of few buttons. On the other hand Body Sensor Networks (BSN) are used to collect the information of the human body through sensors, process the data and transfer the data. Integration of these two enables a user to monitor distributed body sensor networks through Bluetooth technology. A strategy proposed in this thesis will allow a mobile device with Android platform to monitor sensors with Tiny OS operating system by a control centre as an intermediary between these two.

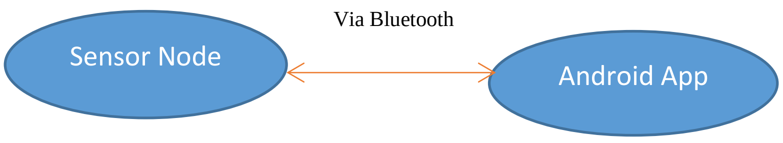


Figure 2.1: Basic Communication of System

There are three main related approaches similar to the model as shown in the above Figure 2.1.These are

1. Using Android IOIO
2. Using Android NFC
3. Using Symbian Bluetooth

The growing popularity of the Android OS and the Bluetooth features like manual setup of connection and the availability of connection over 30 feet distance , enhances the significance of the project.

2.1 ANDROID DEVELOPMENT AND IMPLEMENTATION

Android is an open source operating system for mobile devices (Smartphone and tablet), led by Google. The Android SDK provides a set of tools and APIs to develop Android applications, using Java. In order to develop software applications for the mobile operating system (OS), we need to have an idea about how applications must be structured. In an Android system, applications are composed of activities. They are components of the complete application and provide processes for user interaction. Most applications have more than one activity. These activities in turn can be decomposed into smaller building blocks called views. Views have components for user interfaces such as buttons, text fields, and graphics.

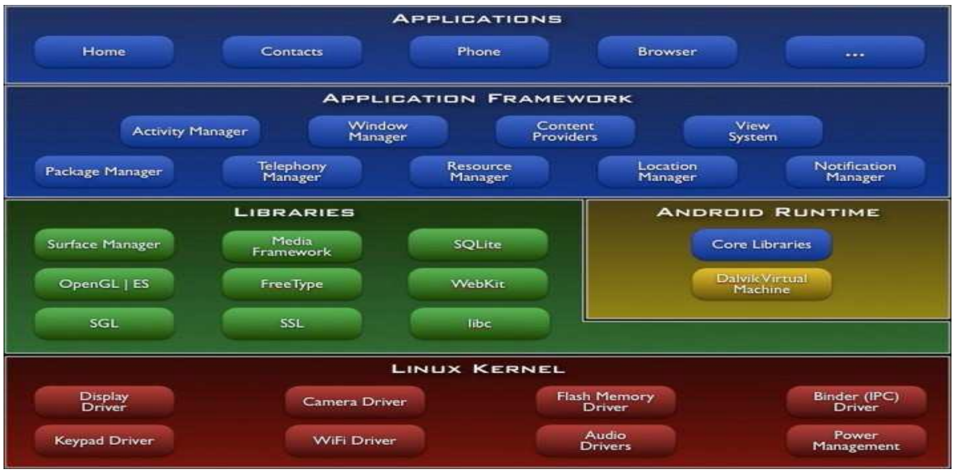


Figure 2.2: Android OS Stack

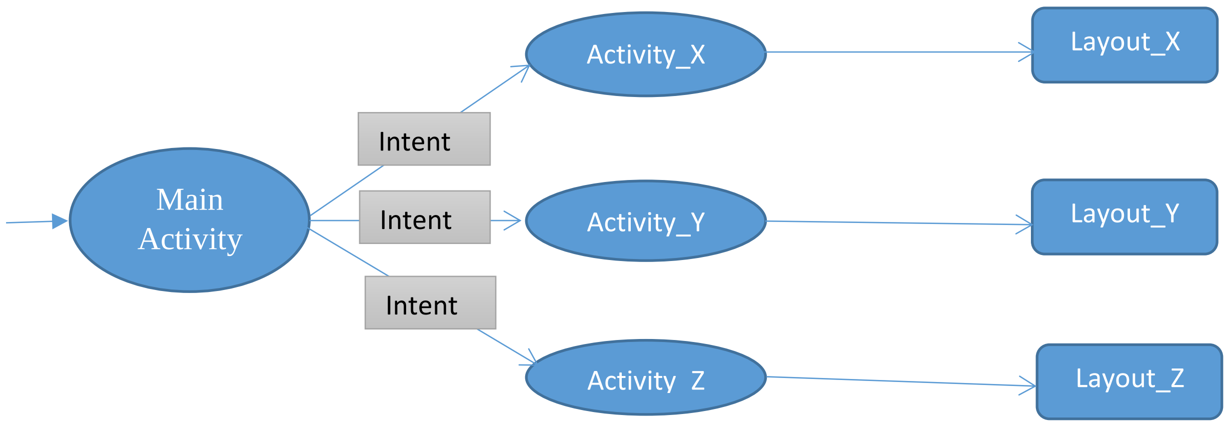


Figure 2.3: Activity Flow in Android

Android includes a set of C/C++ libraries used by various components of the Android system. These capabilities are exposed to developers through the Android application framework. Some of the core libraries are listed below:

- System C library - a BSD-derived implementation of the standard C system library (libc), tuned for embedded Linux-based devices.
- Media Libraries - based on PacketVideo's OpenCORE; the libraries support playback and recording of many popular audio and video formats, as well as static image files, including MPEG4, H.264, MP3, AAC, AMR, JPG, and PNG.
- Surface Manager - manages access to the display subsystem and seamlessly composites 2D and 3D graphic layers from multiple applications.
- LibWebCore - a modern web browser engine which powers both the Android browser and an embeddable web view.
- SGL - the underlying 2D graphics engine.
- 3D libraries - an implementation based on OpenGL ES 1.0 APIs; the libraries use either hardware 3D acceleration (where available) or the included, highly optimized 3D software rasterizer.
- FreeType - bitmap and vector font rendering.
- SQLite - a powerful and lightweight relational database engine available to all applications.

2.2 TINYOS AND IMPLEMENTATION

TinyOS is a free and open source software component-based operating system and platform targeting wireless sensor networks (WSNs). It is an embedded operating system written in the nesC programming language as a set of cooperating tasks and processes. Programs in TinyOS are built out of software components, some of which present hardware abstractions. Components are connected to one another using interfaces. TinyOS provides interfaces and components for common abstractions such as packet communication, routing, sensing, actuation and storage.

CHAPTER 3

DESIGN OF COMMUNICATION MODEL FOR HEALTH MONITORING

3.1 OVERVIEW

The goal was to design a communication system in which the body sensor network (BSN) interacts with Android app developed through Bluetooth technology. The main objective is to develop and an Android app with sensors programmed in TinyOS, to monitor health of a person and keep it updated to concerned medical practitioners.

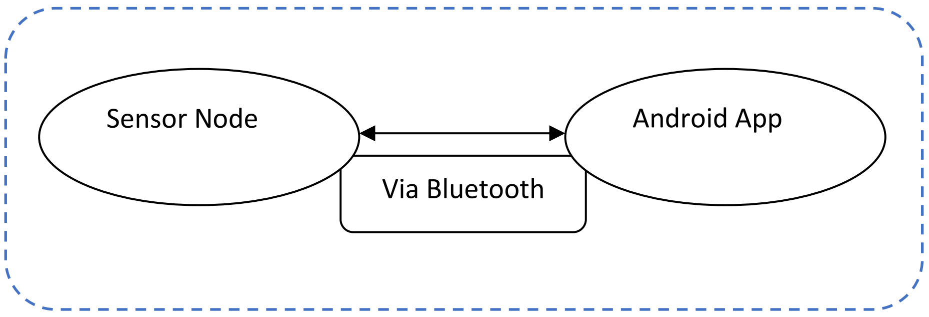


Figure 3.1 Basic Communication Systems

3.2 COMPONENTS AND INTERFACES

1. Android App: Android has many concepts, but they are packaged differently and structured to make phones more crash-resistant:

 Activities: The basic building block of the user interface is the activity. You can think of an activity as being the Android analogue for the window or dialog box in a desktop application or the page in a classic web application. Android is designed to support lots of trivial activities, so we can allow users to keep tapping to open new activities and tapping the back button to back up, just like it’s done in any web browser.

 Services: Activities are short-lived and can be shut down at any time. Services, on the other hand, are designed to keep running, if needed, independent of any activity, similar to the notion of services or daemons on other operating systems. We can use a service to check for updates to an RSS feed or to play back music even if the controlling activity is no longer operating.

 Content providers: Content providers provide a level of abstraction for any data stored on the device that is accessible by multiple applications. The Android development model encourages us to make our own data available to other applications, as well as our own applications. Building a content provider lets us do that, while maintaining complete control over how the data gets accessed. Content providers can be anything from web feeds, to local SQLite databases.

 Intents: Intents are system messages that run around the inside of the device and notify applications of various events, from hardware state changes, to incoming data, to application events. Intents are much like messages or events on other operating systems. Not only can we respond to Intent, but we can create our own to launch other activities or to let us know when specific situations arise.

While Android programs are written in Java, they are not deployed in exactly the same way. A typical Java deployment will package the class files of an application in a Java jar file. However, Android deploys an application in an APK file. Whereas a Java jar file has many class files, each APK file has only a single classes .dex file.

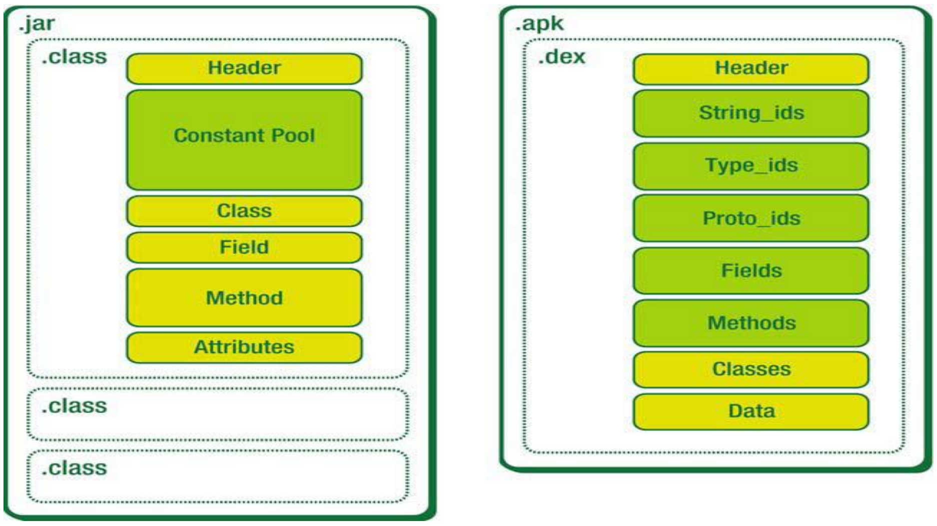


Figure 3.2: Class file vs DEX file

Install the Android SDK

The Android SDK gives you all the tools you need to create and test Android applications. It comes in two parts: the base tools, and version-specific SDKs and related add-ons.

2. Sensor Nodes: Wireless body area sensor nodes (WBAN), due to their size, use miniaturized batteries. The battery life of a sensor node is usually proportional to the size of the battery. Hence, WBANs must sense, process and communicate data in a power efficient manner. The network architecture of WBANs can be broadly classified into two major categories: flat architectures and multi-tier architectures. Flat architectures comprise of a single data gathering unit that sends its data to a personal computer or a personal server application running on a PDA. We are using the flat architecture in our project.

A *sensor* is a device that responds to a physical stimulus (heat, light, sound, pressure, etc) and produces a corresponding measurable electrical signal. In general, a sensor node consists of four basic components: sensor unit, processing unit, transceiver unit, and power unit.

- A sensing unit is usually composed of sensors and analog-to-digital converters (ADCs). The analog signals are produced by the sensors that observe the phenomenon. These signals are converted to digital signals by the ADC, and then sent into the processing unit.
- The processing unit instructs the sensor node to carry out the assigned sensing tasks and manages the sensor node's collaboration with the other nodes. A processing unit has enough storage for the real-time operating system, protocols, and other application-specific algorithms.
- The transceiver unit connects the node to the network.
- The power unit supplies the energy for the sensor node, which may be batteries (coincell, lithium, etc.) or solar cells.

According to the design purposes the sensor nodes are divided into three categories:

- Augmented general-purpose computers dedicated embedded sensor nodes.
- System-on-chip nodes.

A sensor node usually is much more than described above two types, and has relatively higher processing and memory capabilities. Usual operating systems such as Linux and Win CE run in real time, and standard wireless communication protocols such as IEEE 802.11 or Bluetooth are used in the nodes.

The operating system used by this project is TinyOS, designed for low-power wireless embedded systems. Basically, TinyOS is a work scheduler and a collection of drivers for microcontrollers and other ICs commonly used in wireless embedded platforms.

TinyOS is written in nesC, which is a dialect of C. TinyOS program has to be installed on the mote. A simulator for TinyOS also exists, so that programs can be tested without installing them on physical mote.

Installing a program on a mote (Micaz): Mica2 mote and the serial-based programming board (mib510) are used. The programming board is attached to the computer through UART port. The mote, which has sensors, will be placed on the programming board. The mote will pass the data to computer through this programming board.

In our project we have used two sensors, which communicate to each other .One of the motes will collect the actual sensor data; this mote has sensors placed. The data from this mote is passed on to the mote placed on the programming board, which is connected to the computer. The green LED (PWR) on the programming board will be on when power is supplied. If you are using batteries to power the mote, be sure the mote is switched on (the power switch should be towards the connector). The ON/OFF switch on the mib510 board should normally be left in the OFF position. When the switch is ON the mote cannot send data to the PC.

make mica2 reinstall mib510,serialport

where *serialport* is the serial port device name. On Linux, the device name is typically `/dev/ttyS`*n* for a regular serial port and `/dev/ttyUSB`*n*

For Linux, you will typically need to make this serial port world writeable. As superuser, execute the following command:

chmod 666 serialport

For assigning a different identifier for each node you have to enter:

make mica2 reinstall.ID mib510,serialport

TinyOS code is written in nesC, which is nothing but a C program with some additional language features for components and concurrency.

Components and Interfaces

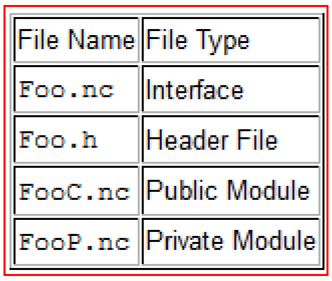
A nesC application consists of one or more *components* assembled, or *wired*, to form an application executable. Components define two scopes: one for their specification which contains the names of their *interfaces*, and a second scope for their implementation. A component *provides* and *uses* interfaces. The provided interfaces are intended to represent the functionality that the component provides to its user in its specification, programmer defines the functionality; the used interfaces represent the functionality the component needs to perform its task or job in its implementation. Interfaces are bidirectional: they specify a set of *commands*, which are functions to be implemented by the interface's provider, and a set of *events*, which are functions to be implemented by the interface's user. For a component to call the commands in an interface, it must implement the events of that interface. A single component may use or provide multiple interfaces and multiple instances of the same interface.

The set of interfaces which a component provides together with the set of interfaces that a component uses is considered that component's *signature*.

Configurations and Modules

There are two types of components in nesC: modules and configurations. Modules provide the implementations of one or more interfaces. Configurations are used to assemble other components together, connecting interfaces used by components to interfaces provided by others. Every nesC application is described by a top-level configuration that wires together the components inside.

Below is the convention used in the TinyOS source tree



File Name	File Type
`Foo.nc`	Interface
`Foo.h`	Header File
`FooC.nc`	Public Module
`FooP.nc`	Private Module

The nesC compiler compiles a nesC application when given the file containing the top-level configuration. Typical TinyOS applications come with a standard Makefile that allows platform selection and invokes ncc with appropriate options on the application's top-level configuration.

TinyOS systems often have many layers of configurations, each of which refines the abstraction in simple ways, building something robust with very little executable code.

Getting to the modules deep down or just navigating the layers with a text editor can be tedious.

Directory Structure:

• The default packages in a TinyOS distribution are:

- *tos/system/.* Core TinyOS components. This directory's components are the ones necessary for TinyOS to actually run.
- *tos/interfaces/.* Core TinyOS interfaces, including hardware-independent abstractions. Expected to be heavily used not just by *tos/system* but throughout all other code. *tos/interfaces* should only contain interfaces named in TEPs.
- *tos/platforms/.* Contains code specific to mote platforms, but chip-independent.
- *tos/chips/.* Contains code specific to particular chips and to chips on particular platforms.
- *tos/libs/.* Contains interfaces and components which extend the usefulness of TinyOS but which are not viewed as essential to its operation. Libraries will likely contain subdirectories.
- *apps/*, *apps/demos*, *apps/tests*, *apps/tutorials*. Contain applications with some division by purpose. Applications may contain subdirectories.

All nesC files must have a *.nc* extension. The nesC compiler requires that the filename match the interface or component name.

TinyOS defines a standard error return type, error_t, similar to Unix's error return types except that error codes are positive:

```
enum {
SUCCESS        = 0,
  FAIL           = 1,
  ESIZE          = 2, // Parameter passed in was too big.
  ...
};
```

SUCCESS represents successful execution of an operation, and FAIL represents some undescribed failure.

CHAPTER 4

IMPLEMENTATION AND TEST RESULTS

4.1 SYSTEM BUILDING

By providing an open development platform, Android offers developers the ability to build extremely rich and innovative applications. Developers are free to take advantage of the device hardware, access location information, run background services, set alarms, add notifications to the status bar, and much, much more. Developers have full access to the same framework APIs used by the core applications. The application architecture is designed to simplify the reuse of components; any application can publish its capabilities and any other application may then make use of those capabilities.

Thus, using Android as the base for the communication system initiation, resetting the Bluetooth on an Android tablet/smartphone, would also initiate the Bluetooth of the body sensors for the collecting the data of the human body for which they are programmed. The data can then be transferred from the body sensors to Android app.

4.2 CONFIGURATIONS

Android Development

- Eclipse IDE, Android SDK
- Testing done on Nexus 7 Android Tablet

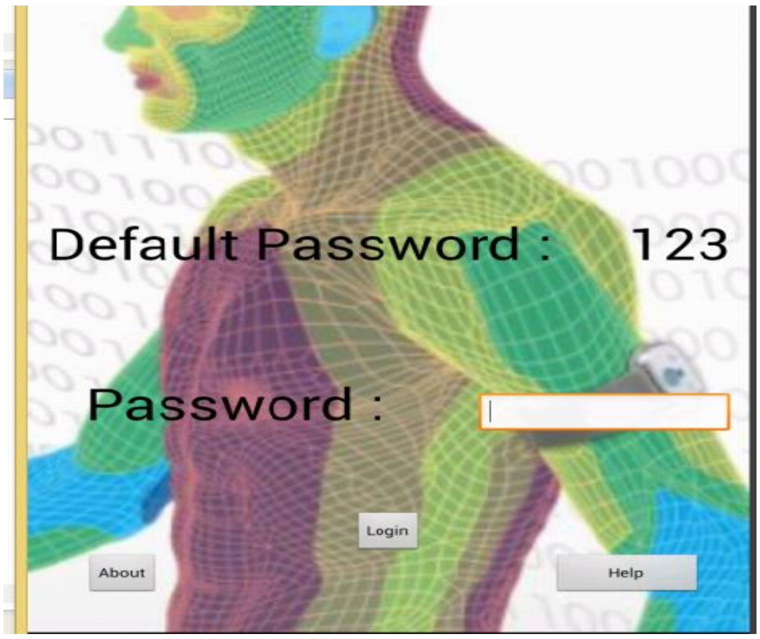
TinyOS

- Installed TinyOS in Ubuntu OS
- Example programs run on MIB510 programming board and mica2 board

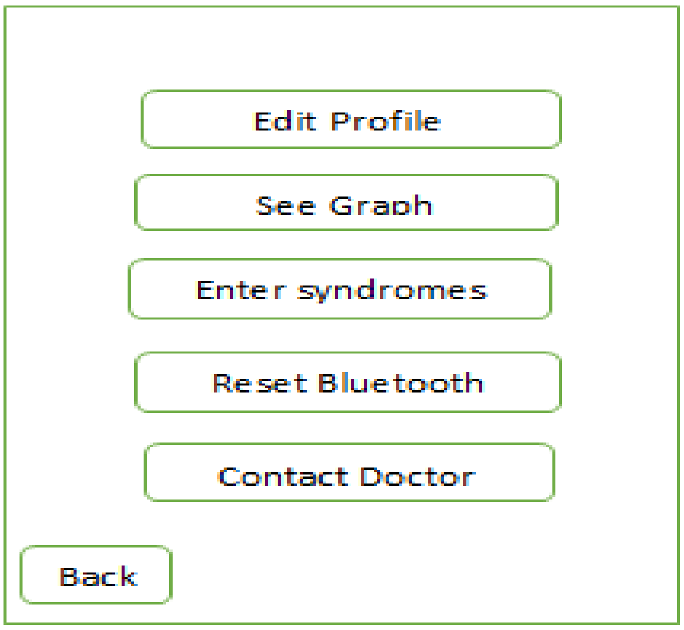
4.3 TEST RESULTS

The Android app developed has the following screens:

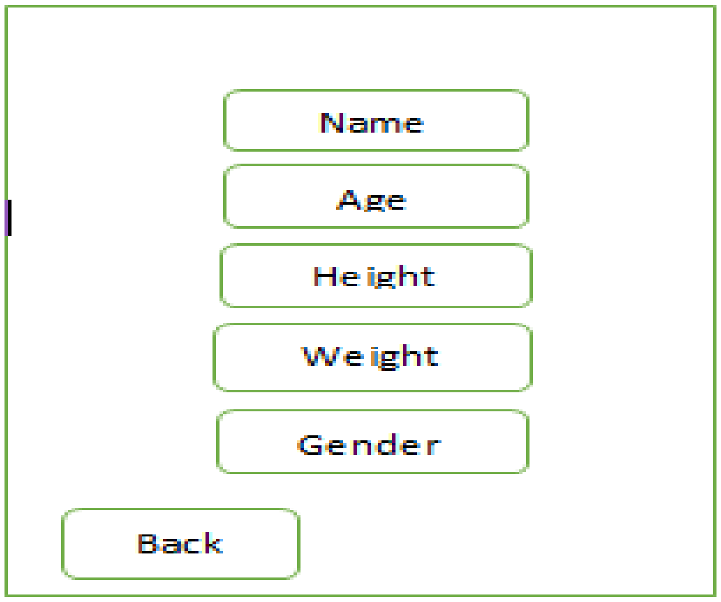
- Below is the login / the first screen that appears when the app loads, requires authentication before the next screen appears.



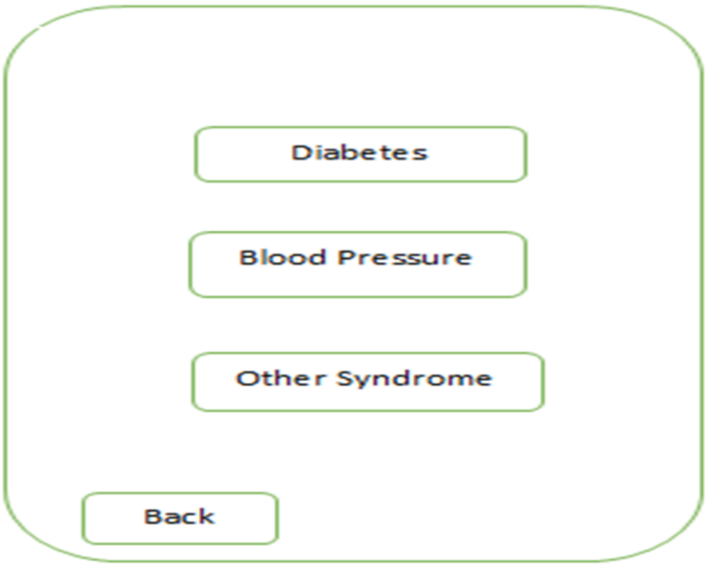
- User Activity Screen: The user can make his selection based on the service he needs.



- User Edit Screen: The user can edit his personal details.

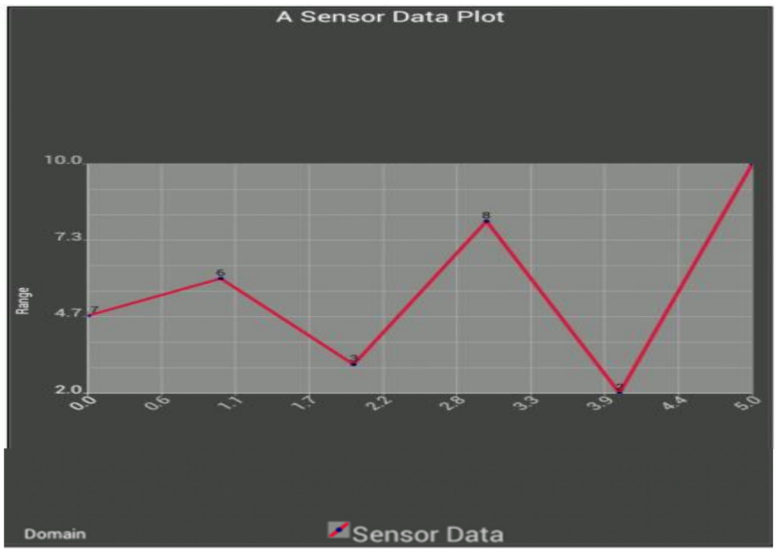


- Graph Selection Screen: The user can select the syndrome for which he wants to see the graph

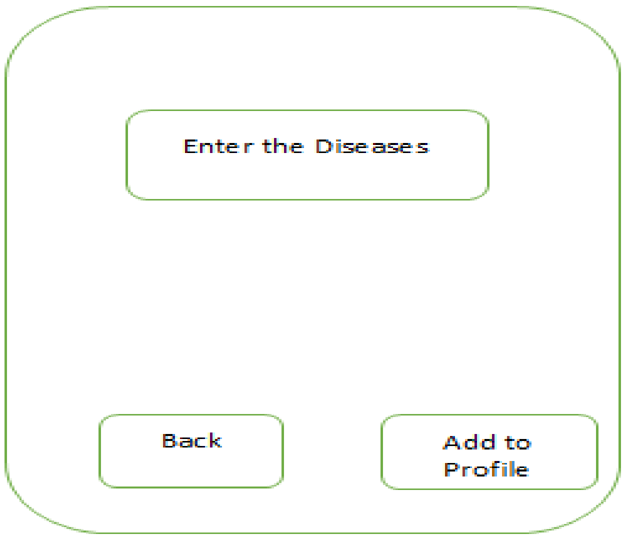


.

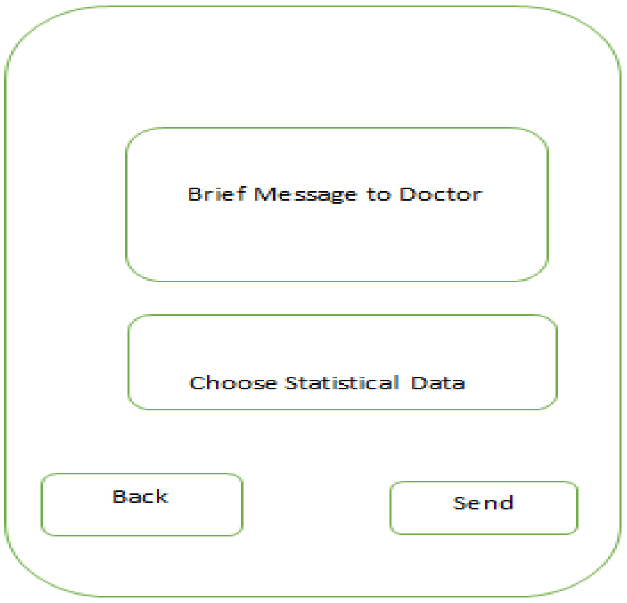
- Based on the previous selection of the user, the graph appears with the values that are stored in database against that syndrome selected.



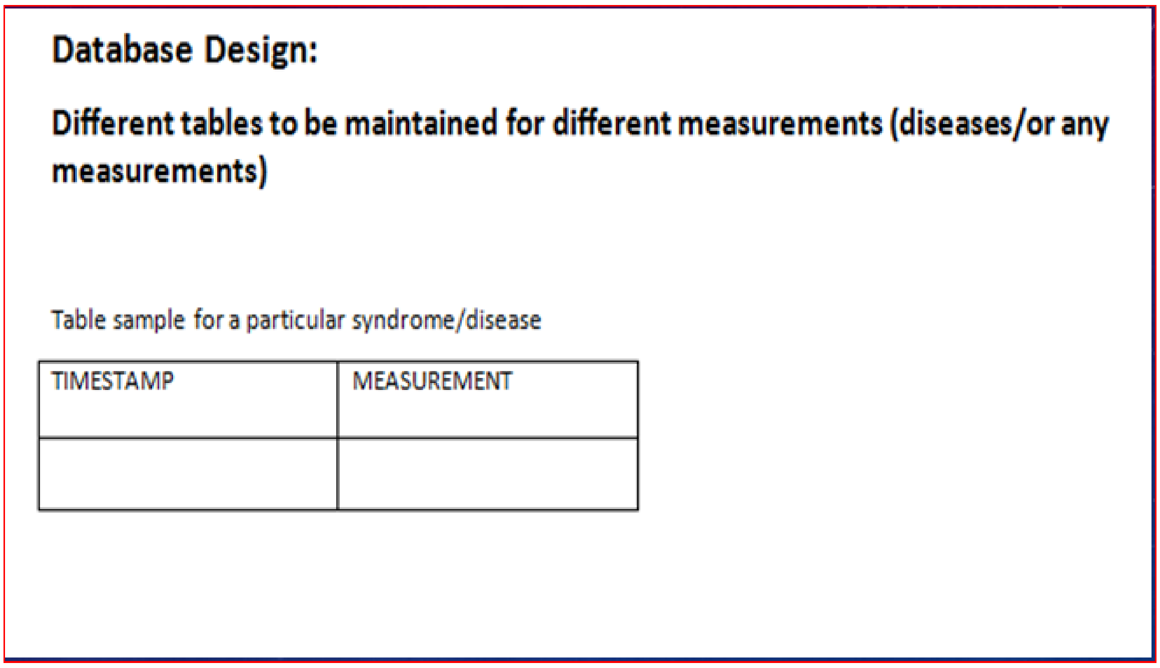
- If user wishes to enter new syndromes, the following screen is used



- Contact Doctor Screen,user defined data with statistical data is sent to the doctor through email.



- The following is the database design



Database Design:

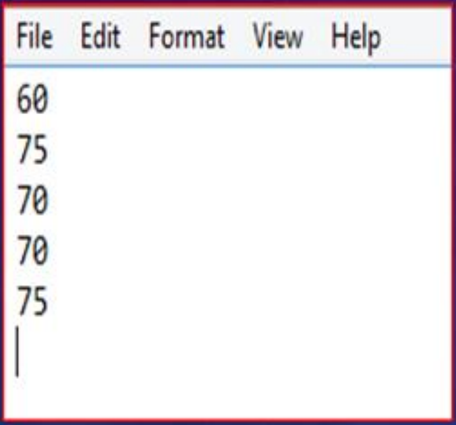
Different tables to be maintained for different measurements (diseases/or any measurements)

Table sample for a particular syndrome/disease

TIMESTAMP	MEASUREMENT

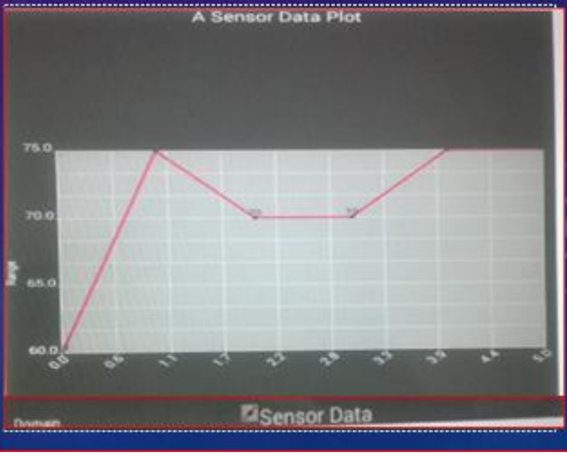
- The data from the sensor is collected by the Android tablet in the form of a .txt file.
- The user of the app can use this file to send it to the concerned physician or the hospitals where a minute by minute track of patient's health is updated.
- The user can analyze the variations in readings through graphs that we provided in database of the app.

Output file from the sensor data



```
60
75
70
70
75
```

Graph from database values in app



LIMITATIONS

The communication system developed is a strong communication system, in which a mobile app in Android communicates with the body sensor node .The Bluetooth technology is used to communicate between two entities.

The project that we are developing is been divided into two parts:

1. Development of an Android app:

   The basic facilities like resetting Bluetooth, creating a profile of the user of the app in the database, entering syndromes, contacting the doctor through basic email facilities are provided.

2. Developing an app such that it receives data from body sensors:

   Need to use TinyOS in the programming boards with Bluetooth enabled, the data is sent from programming boards to the Android app and stored in the database.

CONCLUSION

The ultimate goal of this project is to provide obstetricians and gynaecologists, and other healthcare professionals with a new portable wireless health monitoring system that will improve accurate, reliable and secure data collection. The communication system will provide medical professions with up-to-the-minute information and aggregate data with existing diagnostic systems to notify physicians and patients with rapid feedback through an Android app whenever readings are collected. Our design goal was to develop a system that will enable an easy and efficient monitoring of health without the need to have without the need to have physical connection for power or data collection efficiently anywhere they go. The developed communication system, if fully utilized, should be able to replicate the data collection as the same way in hospitals, with the exception that this system will be more efficient and provide valuable logistical human health monitoring.

REFERENCES:

[1] Michael Korostelev, Li Bai, Jie Wu, Chiu Chiang Tan, Dimitrios Mastrogiannis,"Body Sensor Networks in Fetal Monitoring with NFC Enabled Android Devices”.

[2] Omer Aziz, Benny Lo, Rachel King, Ara Darzi, Guang-Zhong Yang,"Pervasive Body Sensor Network:An Approach to Monitoring the Post-operative Surgical Patient".

[3] S. Sitharama Iyengar, Nandan Parameshwaran, Vir V. Phoha, N. Balakrishnan, Chuka D. Okoye, "FUNDAMENTALS OF SENSOR NETWORK PROGRAMMING".

[4] http://tinyos.stanford.edu/tinyos-wiki/index.php/TinyOS_Documentation_Wiki

[5] http://developer.android.com/guide/index.html

[6] Garth V. Crosby, Tirthankar Ghosh, Renita Murimi, Craig A. Chin, " Wireless Body Area Networks for Healthcare: A Survey ", International Journal of Ad hoc, Sensor & Ubiquitous Computing (IJASUC) Vol.3, No.3, June 2012

[7] Navid Amini, Jerrid E. Matthews, Foad Dabiri, Alireza Vahdatpour, Hyduke Noshadi and Majid Sarrafzadeh ,"A Wireless Embedded Device For Personalized Ultraviolet Monitoring".

VITA

JASMINE SOMAYYA HIREMATH

Candidate for the Degree of

Master of Science

Thesis: A WIRELESS COMMUNICATION SYSTEM FOR PERVASIVE HEALTH-INFORMATION MONITORING

Major Field: COMPUTER SCIENCE

Biographical:

Education:

Completed the requirements for the Master of Science in Computer Science at Oklahoma State University, Stillwater, Oklahoma in December 2014.

Completed the requirements for the Bachelor of Engineering in Computer Science at VTU, Belgaum, Karnataka, India in 2008.

Experience:

Worked as a Graduate Research Assistant under Dr.Tingting Chen from November 2013 to May 2014.