WIRELESS SENSOR NETWORKS

& HEALTH MONITORING

Final Project

Abstract

This project describes the development of a combined software and hardware platform for a health monitoring wireless sensor network. I have developed several medical sensors based on the open source Arduino microcontroller boards, including: pulse oximeters, a 2 lead electro cardiogram board, activity and free fall sensor, barometric pressure sensor, humidity sensor, ambient temperature sensor and light sensor. I have also developed a network real-time data visualization tool along with a persistent data storage layer, a web service interface and web service clients.

I present the initial results of the system demonstrating its performance on various Mesh, Tree and Star topologies under varying data rates. The system has been tested on a 10 node wireless sensor network. The platform is based on open source and standards based hardware and software, which is meant to be used as a study tool by undergraduate students researching wireless sensor networks, health monitoring and telemedicine.

Supervisors:

John Aasted Sørensen

Keld Baden-Kristensen

Student name	**Student number**	**Signature**
Ian Jorgensen	5927	

Table of Contents

Table of figures

1. Introduction

Wireless sensor networks have the potential to greatly assist patients and their caregivers by providing continuous medical monitoring, access to medical data, and emergency communication.

This project describes the development of a standards based, open source wireless sensor network platform with health monitoring nodes. The wireless sensor networking platform is to be used by undergraduate student and has therefore been design to have a very low barrier of entry such that students with low or no previous programming skills can setup and extend.

The report is split into the following sections:

- Project delimitation and methods
- Wireless Networks
- Healthcare Monitoring
- Multitier Architecture
- Node Architecture
- Gateway Architecture
- Evaluation

1.1 Problem Start

As the world population ages, those suffering from diseases of the elderly will increase. Traditionally patients are monitored in controlled environments, where the frequency of the measurements normally depends on the state of the patient. Emergency and intensive care patients tend to be closely monitored while less critical cases are not.

Wireless sensor networks have the potential to reach uncontrolled environment such as home networks and provide unobtrusive area and environmental sensors combine with physiological sensors in order to continuously monitor and evaluate the health of spaces and the people who inhabit them. Among many others this can benefit long term rehabilitation patients, the elderly and people with chronicle diseases.

Although many corporations provide wireless sensor network solutions, most have developed their own proprietary systems. This approach might work in specific industry solutions but it's not a sustainable approach for the mass consumer market.

As wireless sensor networks reach consumer market, the adoption of standards will be critical.

As wireless sensor networks become prevalent across all industries; the need to introduce them into undergraduate engineering curriculum increases. Current solutions are easy to setup but require significant programming knowledge to extend and customize. This makes them difficult to introduce in undergraduate engineer courses.

1.2 Problem Analysis

The main problem I am trying to solve is the lack of an open source, standard based wireless sensor network platform that's accessible both technically and economically to undergraduate students. The platform is meant for health monitoring applications, and should also cope with broader usages.

The problem consists of 4 main areas, and each of them can be solved separately. In this section I will attempt to describe the decision process that leads to the final solution.

In order to find possible solutions for the presented problem I need to investigate the current solutions used in health monitoring. I also need to analyze existing wireless networking standards and the nodes in which they are implemented. In order to collect and display the collected data in the network I must analyze possible gateway and data client solutions. The figure below provides an overview of main elements of the problem.

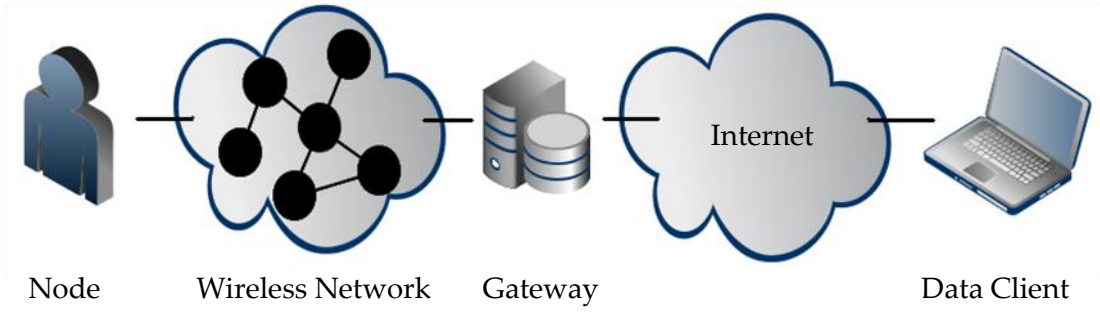

Figure 1 Problem Areas Overview

1.2.1 *Health monitoring*

Research shows that the health of an individual can be related to the conditions of the space he inhabits. The amount of light, pressure, temperature, presence of toxic gases, and humidity of the space a patient inhabits can have significant effect on the overall health of a patient. Data from health monitoring sensors such as electrocardiographs, pulse oximeter, blood pressure monitors and blood sugar monitors can be combined with that of light, humidity, temperature, barometric pressure, and movement sensors. This data can help patients make corrective changes in their life style. Such solutions are still not commercially available on the healthcare market.

The main reasons for the lack of such solutions is that the requirements of a health monitoring wireless sensor networking platform are very different from those of existing wireless sensor network solutions.

The opportunities and specific application scenarios for wireless sensors networks in the medical industry is non exhaustive. A sensor network designed for an emergency situation has very different requirements than one deployed permanently in a hospital. There are however general considerations that all health care networks share.

Health monitoring networks are characterized by mobile nodes with varying data rates. Unlike in traditional sensor networks there is less interest in maximizing individual node lifetimes since it is acceptable to change batteries or charge devices on a relatively frequent basis.

A health monitoring network must support varying data rates. Depending on the sensors in use, sampling rates may range anywhere from less than 1 Hz to 500 Hz or more. This places heavy demands on the nodes and the wireless channel.

In order for Health monitoring sensors to be successful they need to be as unobtrusive as possible. A physiological health monitoring sensors node has to be wearable meaning: lightweight and have a very small footprint.

Health monitoring nodes have to be able to reliably and securely transmit data. Occasional packet loss can be accepted but extended packet loss or noisy data could

be very problematic. Medical data is by nature very sensitive so all communication must be encrypted.

A node be able to move freely with the network, and must be able to quickly find new routes to transmit its packets. This makes a mesh networking protocol ideally for such applications.

1.2.2 *Wireless Network*

Wireless sensor networks exist in many varieties, which can be characterized by their data rates and power consumption requirements. The figure below provides an overview of the wireless networking standards currently available:

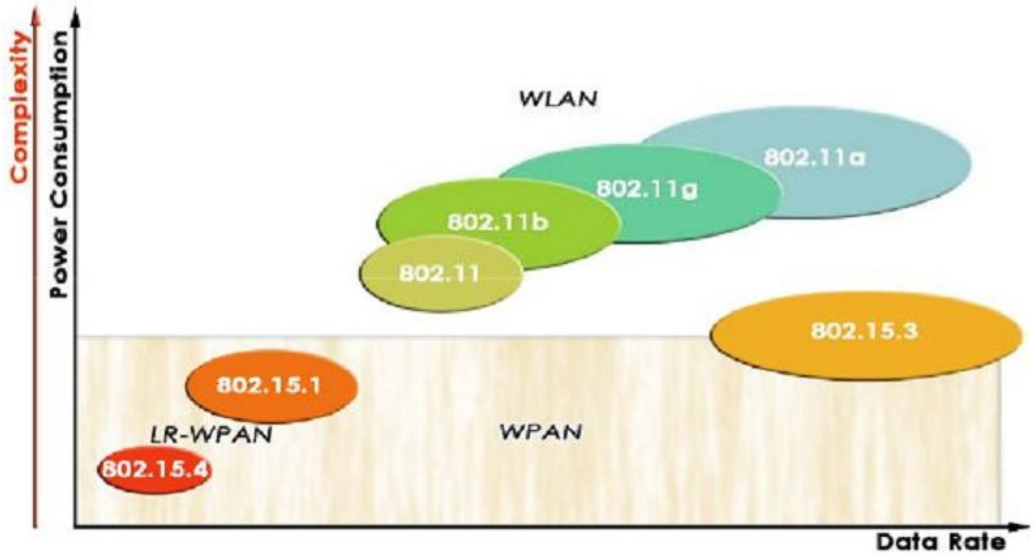

Figure 2 IEEE Wireless Networks (Beltrame, 2008)

This project focus on low rate wireless personal area networks, shown on the bottom left of the figure above.

There are various low rate wireless networking standards available today. Some of them focus on specific application areas, such as industrial automation, environmental monitoring, home automation and more recently healthcare monitoring.

The 6LoWPAN standard is the implementation of the IPv6 stack on top of IEEE 802.15.4 standard; its aim is to enable wireless devices to both be accessible and able to access the Internet. The standard has a lot of potential but is yet to prove itself in

the industry. The standard is open and being developed by the The Internet Engineering Task Force (IETF).

The DigiMesh protocol is a completely distributed mesh protocol where all the nodes can sleep and route their brother's packets. The protocol is a closed and proprietary; it is developed by the Digi Corporation. In order to make a network without distinction between nodes types the network operates on a synchronized clock. This allows all devices to sleep and wake up together. This approach makes the network setup very easy but puts limitations on the types of applications it can be used, since the complexity of the network will grow if the nodes data rates significantly differ.

Another closed and proprietary wireless networking protocol is XMesh developed by Crossbow. The protocol supports full mesh network and over the air programming, but is only implemented in the Xbow devices provided by the corporation.

Code blue is a proprietary medical sensor network platform developed by Harvard university sensor network lab. The specifications are yet to be published. The protocol supports device discovery, publish/subscribe multi-hop routing and RF localization.

Finally, ZigBee is a low-cost, low-power, wireless mesh networking proprietary standard. The standard is specified by the ZigBee Alliance. The full specification for the standard is available to educational and research institutions free of charge and to the alliance members. ZigBee is the most prevalent wireless mesh networking standard to date. Besides specifying the standard the alliance defines application profiles that allow multiple vendors to create interoperable products. Application profiles have been defined for areas such as Home Automation, Energy monitoring, Commercial Building Automation. Most recently the alliance is specifying application profiles for Personal, Home, and Hospital Care.

Section 3.1.1 Protocols using IEEE 802.15.4 Page 21 provides a quick overview of these and other low rate wireless personal area networks LR-WPAN protocols available. These protocol can also be referred to as Low-WPAN, low power WPAN’s.

1.2.3 *Node*

Wireless nodes are available in many varieties, but unfortunately the choice of nodes is greatly determined by the choice of wireless networking standard. The reason is that corporations developing wireless networking standards make profit by selling devices that implement their standard locking their customers into propriety solutions. This generates segmentation in the market, and ultimately limits the end users choice. ZigBee's popularity is slowly changing this. Both Crossbow and Digi provide wireless modules both for their proprietary standards and ZigBee standard.

Crossbow provides the 2 different wireless modules (IMOTE and MICA).

The IMOTE contains a low-power 32 bit XScale processor and an 802.15.4 radio with a built-in 2.4GHz antenna. It is a modular stackable platform and can be expanded with extension boards to customize the system to a specific application. Some version of the product can run the .NET Micro Framework.

The MICA boards are based on the Atmel low power ATmega128L and a 802.15.4 radio. The board can run both sensor application and network communication stack simultaneously. It has a 51-pin expansion connector that supports analog inputs, digital I/O, I2C, SPI and UART interfaces.

Unlike Crossbow, Digi provides a single transceiver module called Xbee with different configurations. Instead of providing a complete wireless sensors module solution, they provide transceiver modules that can be easily connect to external and microcontroller by a simple USART interface. The transceiver has basic input I/O functionality and as such can be used as a standalone wireless node for simple applications.

The Xbee models differ by protocol, range, and operating frequency. They are ideal for low-power, low-cost applications. Xbee-PRO modules are power-amplified, extended-range versions of Xbee modules. All modules share a common hardware and communication interface. Modules can use a simple AT commend interface that is equal for all models or a more advanced API interface, specific to the capabilities of each model.

In order to add multiple sensors and perform processing on the data captured before transmission over to network, a system using the Xbee modules would require an external microcontroller. This can be considered as an advantage over Crossbow, since it allows for free choice of microcontroller platform. A reasonably

new and emerging open source embedded computing platform is Arduino. The platform provides multiple libraries, and the hardware is open source and based on Atmel microcontrollers. Apart from being a good development platform Arduino has strong community that develops hardware projects openly sharing code and hardware schematics. The platform is designed to be accessible to inexperienced programmers and is flexible enough to fulfill the need of advanced applications. Arduino has even been ported to ARM processors. (Matt, 2009)

1.2.4 *Gateway*

A gateway is used to collect the data from the network. The data is classified based on the specific protocol and is usually displayed graphically. A gateway can also provide data persistence mechanism, and make its data available to others.

1.2.4.1 Data Collection & Visualization

Data entering the Gateway can be displayed in charts or simply be presented in text format. Making sense of the data entering the gateway if it's displayed in text format can be very difficult. From a development standpoint displaying the data in raw text is not only simple to implement but can provided valuable debugging feedback during the development phases. A visual representation of the data requires significantly more work but provides the user with a lot of value. A chart could be displayed for every data type, allowing the user to view sensor readings as they are received. The visualization tool could also provided a table giving an overview of the nodes and sensor connected to the network.

Networking diagnostics tools could be included to view the current network topologies, asses packet loss and even signal strength metrics. This can be a very powerful tool to provide a good overview of the state of the network.

1.2.4.2 Data Storage

Besides displaying data the gateway is often responsible for storing the data collected.

There are various ways of approaching this problem. The data could be stored in a CSV or XML file. This approach is simple to implement and flexible. A potential issue that could arrive when storing data in a text file is conflicting file system locks. This can occur when two processes are trying to access a file, for example a process writing data and another trying to read the data. Therefore storing data in a file is best when the file will be accessed only once all the data has been written to it.

Another approach would be to store the data in a database. Storing the data in a database might seem like overkill but it provides a lot of flexibility both when entering and reading data.

A database provides sophisticated queering mechanisms. A client can query the database requesting sensors readings, it can for example query for specific sensors, time range, and thresholds. Besides queering capabilities databases can be easily expanded and backed-up if more storage is needed, or too many clients are requesting data from it. A database can make data easily accessible to multiple users.

1.2.4.3 Data Interface

A gateway can also provide an interface to the stored data. Again this could be done in many ways.

If the data is stored in a CSV or XML file, making it accessible is simply a matter of specify the location of the file. This approach can be very effective for networks with low data rates and very local use base.

Another approach is to make the data collected by the gateway accessible though a web server. A web server can expose web service interfaces for different functions; such are retrieving network configuration setting and sensor readings.

XML web services are the most common web services. The Web Services Description Language (WSDL) is an XML-based language that provides a model for describing Web services. It allows for programmatic definition of web services; it describes a web service function through its inputs and outputs. The advantage of WDSL compliant web services is that the client can load the web service definition and programmatically access the interface without ever needing to read documentation. The disadvantages of WSDL language is that they require special libraries to implement and can be difficult to consume for non WSDL aware clients.

The two main web service technologies are SOAP and REST.

SOAP, defined as Simple Object Access Protocol, is a protocol specification for exchanging structured information in the implementation of Web Services. It relies on XML as its message format, and relies on other protocols such are HTTP for communication (Wikipedia, 2009).

REST defines a set of architectural principles by which to design Web services that focus on a system's resources, including how resource states are addressed and transferred over HTTP. REST has emerged in the last few years alone as a predominant Web service design model. REST has had such a large impact on the Web; it has mostly displaced SOAP- and WSDL-based interface design because of its considerably simpler usage style (IBM, 2008).

Unlike a SOAP web service a REST web service can return data in various formats. Most traditional format is XML, which although very useful has a significant amount of over head and cannot be consumed without first being parsed by a client. Another recently popular format is JSON. JSON (JavaScript Object Notation) is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. In fact since the format is JavaScript, a web browser can simply interpret the data in the same way it interprets code; this is much more efficient than parsing XML.

Below is a list of the main advantages and disadvantages of SOAP and REST web services. (Iskiw, 2009)

REST advantages:

- Lightweight - not a lot of extra xml markup
- Human Readable Results
- Easy to build - no toolkits required
- Easy to call from any browser.
- Can easily be spit out in XML or JSON therefore easily for JavaScript code to parse and work with.

SOAP advantages:

- Easy to consume - sometimes
- Rigid - type checking, adheres to a contract
- Development tools
- Communications are strict and structured
- Can Easily create strongly types defined interface with WSDL

REST disadvantages:

- Hard to make strongly typed objects to work with it in server side code.
- Only works over HTTP (not good for high performance apps that need to work over other TPC or UPD protocol)

SOAP disadvantages:

- Soap Messages basically cannot be cached.
- The response of a SOAP message cannot be used in JavaScript, the main programming language of the web. In fact the response cannot be used unless it's parsed.

1.2.5 *Client*

A data client must be able to communicate with the gateway in order to obtain data collected in the sensor network. The choice of Gateway data interface will greatly determine the type of data client.

A client can operate in one of two modes. It can operate in a subscription mode where data is pushed to it whenever it available, or it can operate in a publish mode where it request data from the gateway whenever it needs it.

If the data is very time sensitive, a client could obtain near real time data from the server using protocols such as UDP. This would classify as a subscription mode. This would require a sophisticated client with an implementation of a UPD protocols. Alternatively TCP sockets could be used, but again this requires relatively complex clients. A real time client would simply get a stream of data from the gateway and display it. The number of clients a gateway could support on this mode would be limited, considering the load a client would create on the server.

If data is not highly time sensitive, the client can instead request the data from the gateway data store. This would classify as a publish mode. A database would allow the client to make sophisticated queries and extract only the data it requires. The number of clients a gateway could cope in this mode would be significant since clients do not need to keep an open connection with the server.

2. Project delimitation and methods

Now I will describe the solution strategy reached by analysis of the 5 main elements described in the previous section. The plan for executing the solution is also included along with the system requirements.

2.1 Solution strategy

The following are the main design considerations that largely determined the outcome of the decisions that lead to the final solution.

1. Standards Based
2. Open Source
3. Accessible

The solution is specified for the Health Monitoring, Wireless Network, Nodes, Gateway and Client elements. I chose ZigBee as wireless networking standard. The nodes are based on Xbee transceiver and the Arduino microcontroller. The Gateway consists of a REST JSON Web service developed in ASP.NET, a MS SQL Database, and real time visualization tool developed in .NET. I use the internet as a communication medium between the gateway and the client. The data clients are any http enabled application such as web browser and Matlab. The figure below provides an overview of the system.

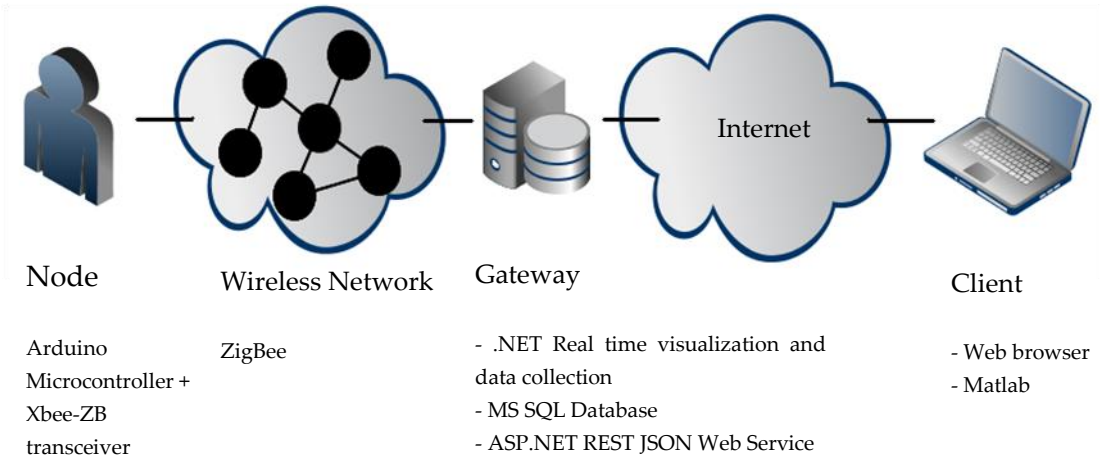

Figure 3 System Overview

With an alliance of over 300 members ZigBee is the most ubiquitous wireless mesh standard to date. The standard is widely adopted and constantly being improved by

the requirements of its members. Other standards such as DigiMesh, XMesh or Code Blue standard might provide some advantages over ZigBee, but any shortcomings of ZigBee will likely be overcome as the technology improves.

Besides selecting and openly specified and widely adopted standard, I chose a node design that would allow for clear separation of concerns between the wireless transceiver and the microcontroller. Most solutions to date bundle the two components in an attempt to cut production costs and improve battery performance. In this case battery requirements are not very strict and modularization is prioritized over production costs (note that production and sale costs are not the same).

Given that the target users of the platform are to be mainly undergraduate students with little programming knowledge I needed to choose a platform with a short learning curve, flexible functionally, and where help resources and documentation where easy available. The platform as a whole needed to be extensible to cope with future requirements.

My first consideration for wireless sensors network platform Xbow nodes widely used in the wireless sensors research community. These are commonly used among research institutions creating their own wireless protocols on top of the IEEE 802.15.4 standard. They are programmed in an open source operation system called TinyOS, which is a very powerful and flexible but complex operating system. TinyOS has a very long learning curve and is therefore not well suited for inexperienced programmers. The nodes implement a proprietary protocol called XMesh.

In looking for viable nodes platform, the other alternative was to use a standard microcontroller and a transceiver module that communicates with it. This approach reduces complexity by separating concerns (modularizing). The programmers do not need to worry about issues such as code footprint and timing conflicts; instead a simple hardware interface defines the communication between the modules greatly simplifying development. A modularized approach is flexible and allows for changes of any component as long as the communication interface remains the same. This allows for more powerful microcontroller to be added when necessary and easily upgrading of the transceiver if new non-firmware upgradable versions become available.

It is therefore that I selected the Arduino microcontroller boards as the node in the network, and Xbee modules as transceivers.

Arduino is an open-source electronics prototyping platform based on flexible, easy-to-use hardware and software. It's intended for inexperienced programmers and its ease of use has made it popular amongst artists, designers and hobbyists. Every aspect form the hardware to the development and even the C compiler is open source. The compiler is GNU open source compiler and as such does not require any licensing or agreements to operate. The programming environment is implemented in Java and runs on operation system with a Java JVM implementation (MAC, Windows and Linux). One of the main reasons for its popularity is the ecosystem of companies and individuals that contribute to the project. This community makes Arduino very accessible to beginners and advanced users.

The Xbee modules are very powerful wireless transceivers, that can be configure to run IEE 802.15.4, DigiMesh and ZigBee standard. Communication between the transceiver and its host is defined thorough a simple serial interface.

The choice of open source hardware and very flexible transceiver was taken in an attempt to make the system future proof. Selecting closed technologies might offer certain benefits in the short run, but are hard to sustain on the long run. I selected a design that put the dependencies on technologies and not on corporations. Wherever a new better standard or better a microcontroller becomes available, the nodes need simply to be changed in order for the system to take advantage of it improvements over existing one.

The sensors of choice are both health monitoring and ambient monitoring sensors. Health monitoring sensor includes ECG and POX sensors. The ambient sensors are Barometric Pressure, Humidity, Temperature, Light, and an Activity and fall sensor. The wide variety of sensors was made possible by the Arduino community. Many of the sensors are widely used in the community and libraries are already available for them. The health monitoring sensors are new to the Arduino community and need to be developed from scratch.

The Gateway is developed in .NET framework with use of open source charting framework called VisiFire. The data persistence layer was developed in using a SQL Server database and web services are hosted by IIS (Internet Information Services) and expose the browser friendly JSON. The web services are developed in ASP.NET. The data clients include web browser and MATLAB client.

The use of a database greatly facilitates the development of a flexible web service. Not the mention the ability to easily perform backups and system administration tasks. The choice of JSON format was taken because of its native browser support,

ease of use and readability. MATLAB was chosen as a data client because of its powerful data graphing and analysis capabilities and because of it wise use within universities.

The choice of Microsoft technologies was heavily determined by my experience with it. I have worked as a Software Development Engineer at Microsoft for 3 years and am therefore very familiar with its technologies. Choosing a familiar platform allowed me to focus my efforts on learning about wireless sensor networks, selecting a node platform, programming microcontroller, implementing health monitoring sensor and setting up the wireless network.

2.2 Requirements

In order to be able to evaluate the success or failure of the project I specified formal requirements. Besides providing a basis with which to evaluate the outcome, requirements help maintain focus and clear expectations during the development phases. The requirements specified below are the functional and non functional requirements of the system.

- Data provided by the network must be visualized in real time
- All data provided by the network must be persisted
- Persisted data must be made accessible
- The platform must be extensible
- The platform must be accessible to novice programmers
- The platform must be based on standards
- The platform must be open source
- The platform must be able to scale to deal with concurrent users
- The platform must cope with high and low bandwidth data transmissions

Requirements are also specified for the main modules of the platform.

The wireless sensor network must:

- Support tree, star and mesh topologies
- Must be able to secure data transmission
- Must support acknowledgments
- Node must be able to transmit reading of String, Integer and Float data types
- A node must be able to transmit various sensor values
- Node must allow configuration of sensors and node settings

- Nodes must be able to select sensor sampling intervals
- Nodes must be able to select transmission intervals
- Nodes must be able to sleep
- Nodes must be able to successfully transmit ECG and PPG (Plethogram) waveform sampled at least at 50 Hz over the network
- Nodes must be able to transmit Barometric pressure, humidity, temperature, and light measurements

Real time data visualization tool must:

- Show charts of data as they are received
- Select which sensors data to view
- Set database connection
- Receive network data from the network coordinator
- Determine data reading source
- Write parsed initialization frame to database
- Write the data readings to a database

Database must:

- Store sensor readings
- Store node, sensor id and configuration data

Web service must:

- Expose sensor reading data through web service
- Exposes node configuration and sensors connected to it through a web service
- Expose the nodes in the network though a web service

Data client must:

- Be able to query for nodes in the network
- Be able to query for sensor attached to a node
- Be able to query for data from a specific sensor
- Be able to graph and perform data analysis

2.3 Milestone plan

Having identified all major aspects of the development phases I came up with a milestone plan. The table shows the prioritized list of intermediate tasks to be

completed in order to reach our goal of the project. The amount of effort required to complete each task is also included.

Each entry represents the main phases of the project. Most critical tasks were performed at the begging of the project and the less crucial ones are left to the end. The way that this is evaluated is based on my experience. Having had a lot of professional experience programming database and web services I decided to focus my efforts on other phases of the project.

Milestone	**Effort**
Choose wireless sensor platform.	5
Purchase wireless sensor boards	2
Setup a tree network	2
Get data from ECG and POX sensors	5
Define sensor data transmission protocol.	5
Store the data in a database	2
Test network mesh topology	2
Display data in real time	3
Define a web service for clients to access the data	3
Create MATLAB Client	3
Add additional sensors to the network	2
Perform detailed evaluation of the system.	4

2.4 Resources

As described in the problem analysis the platform consists of many different components. Each of those components has a set of resources.

The wireless sensor network is built out of Arduino boards, Xbee transceivers and various sensors. These are listed below.

Wireless Network Components:

Amount	**Name**
9	Arduino Duemilanove Boards
1	Arduino Mega board
1	Arduino Nano board
8	Arduino Xbee shields
2	Xbee Explorers
8	Xbee Series 2 Transceivers
2	Xbee Series 2 PRO Transceivers
2	Arduino Lithium battery shields (Lithium Backpack)

Node Sensors:

Amount	Name
1	Humidity and Temperature Sensor SHT15
1	Light Sensor LDR
1	Barometric Pressure Sensor SCP1000
1	3-Axis accelerometer AXL345
1	EC1000 Two lead ECG board (including electrodes)
2	POX Oxilink boards
2	POX Oxilink finger probes

Additional components (not discussed in the project):

Amount	Name
1	GPS Receiver
1	Cellular GPRS Module

The above lists show the physical resources used. In order to program the nodes and configure the network the following tools were used:

- Arduino Development Environment (see Arduino Programming Language)
- Digi XCTU Xbee Configuration program (see Appendix Xbee configuration).
- Arduino Xbee API (Rapp)

The real time data visualization tool was developed on Visual Studio Express 2008; which uses the following components:

- VisiFire Charts
- NET Framework 3.5
- ASP.NET

The database and web services were also developed in Visual Studio Express 2008; the tool uses the following components.

- IIS 6 or above
- SQL Server Express 2008 or above

The web service clients are platform independent. The web service can be consumed by any http enable application such as a web browser. MATLAB can be used to access the service through the functions I have developed.

3. Wireless Networks

Before discussing the details of ZigBee, the IEEE 802.15.4 and standards I will present the basic principles of network architecture design using the OSI layer model.

The OSI Model is an abstract description for layered communications and computer network protocol design. It divides network architecture into layers which, from top to bottom, are the Application, Transport, Network, Link, Media Access Control, and Physical Layers. A layer is a collection of functions that provide services to the layer above it and receives service from the layer below it. Two instances at one layer are connected by a horizontal protocol connection (Wikipedia, 2009).

In the ZigBee standard the physical and media access layers are defined by the IEEE 802.15.4 standard. The network and application layers are defined by the ZigBee alliance. The ZigBee alliance defined a set of public application profiles for application areas such as home automation environmental monitoring and is in the process of specifying healthcare public profiles.

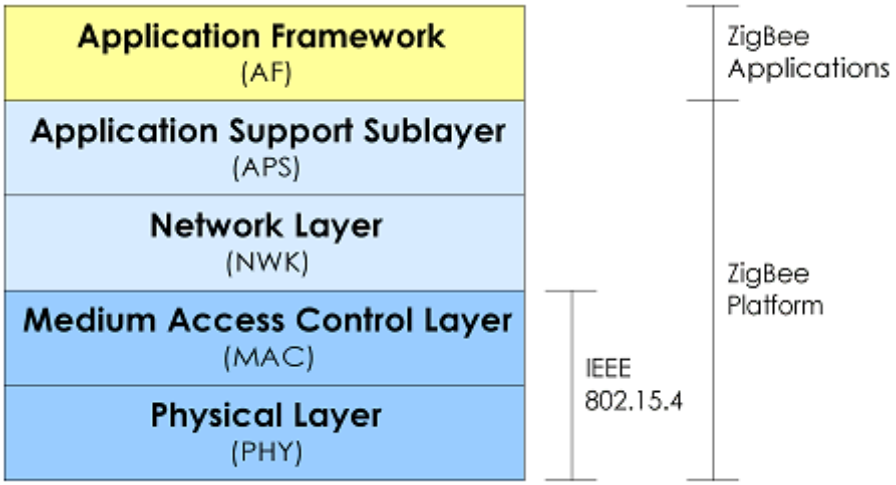

Figure 4 ZigBee / IEEE 802.15.4 OSI Model (Munk-Stander, 2005)

The relationship between IEEE 802.15.4 and ZigBee is similar to that between IEEE 802.11 and the Wi-Fi Alliance.

3.1 IEEE 802.15.4

The IEEE 802.15.4 is a low-rate wireless personal area network - LR-WPAN standard. The standard was created by the Institute of Electrical and Electronics Engineers - IEEE. The IEEE is one of the leading standards-making organizations in the world. The more notable IEEE standards is the IEEE 802 LAN/MAN group of standards which includes the IEEE 802.3 Ethernet standard, the IEEE 802.11 Wireless Networking standard and the increasingly popular and most relevant to this project the IEEE 802.15.4 low-rate wireless personal area networks standard. (IEEE, 2003)

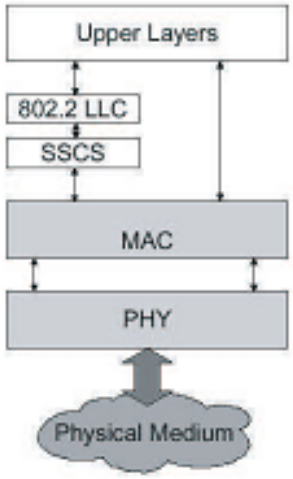

Figure 5 IEEE 802.15.4 Model (IEEE, 2003)

The IEEE 802.15.4 standard defines the lower network communication layers; the Media Access Control and Physical layer as seen in the figure above. Its main purpose is to enable low-cost, low-speed ubiquitous wireless communication between devices. Low cost refers to low computational and subsequently and energy cost. The code size for a standard implementation of 802.15.4 is about ¼ of that of a standard implementation of ZigBee (Wikipedia, 2009). The focus on low cost results in extremely low manufacturing, operation costs and technological simplicity, without sacrificing flexibility or viability of usage in real life applications. The standard supports both peer-to-peer and star topologies.

- Medium Access Control (MAC) Layer

The Media access control layer provides access to the radio channel using CSMA/CA; it is responsible for providing reliable communications between a node and its immediate neighbors, helping avoid collisions, improve efficiency, and

assembling and disassembling data packets and frames. It provides support for transmitting beacon frames and network synchronization and achieves reliable transmission using CRC and retransmissions.

- Physical (PHY) Layer

The physical layer is the lowest layer and is defined in the IEEE 802.15.4 standard. It provides the interface to the radio transmission medium. The PHY layer consists of two layers that operate in two separate frequency bands meant to provide worldwide unlicensed frequency access:

Low frequency bands

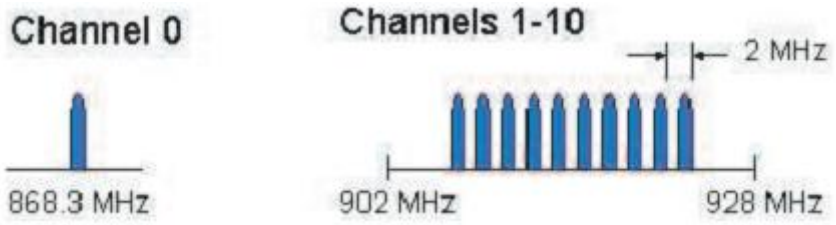

Figure 6 915 and 868 MHz Channels (Munk-Stander, 2005)

868-868.8 MHz: Europe, allows one communication channel

902-928 MHz: North America, up to thirty channels

High frequency band

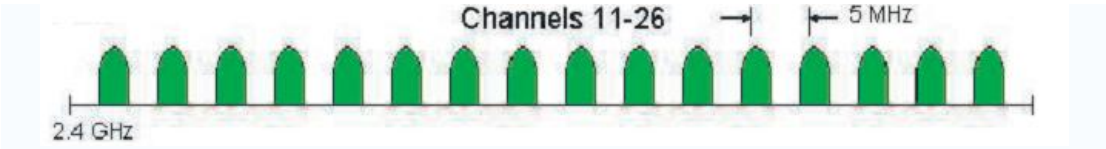

Figure 7 2.4 GHz Channel (Munk-Stander, 2005)

- 2400-2483.5 MHz: worldwide use up to sixteen channels

Nodes in this network can be set to use solely the IEEE 802.15.4 protocol by using the Xbee 805.15.4 Series 1 Digi transceiver modules. If nodes are programmed to communicate through serial AT commands, Series 1 and 2 transceivers can be switched with zero configuration and software updates.

3.1.1 *Protocols using IEEE 802.15.4*

There are several protocols which use 802.15.4 as its MAC and Physical layer. The most known is ZigBee, although there are many more:

- Wireless HART

  The wireless HART protocol is most used in automation and industrial applications which require real time. It uses Time Synchronized Mesh Protocol (TSMP). A time coordinator node is used to assign the time slot for all the nodes.

- ISA - SP100

  SP1000 protocol is centered in the process and factory automation industry. It is being developed by the Systems and Automation Society (ISA) with the aim to become a standard for the automation industry.

- 6LoWPAN

  The 6LoWPAN as the same name points it is the implementation of the IPv6 stack on top of 802.15.4 to let any device be accessible and access from and to Internet.

- DigiMesh

  DigiMesh protocol is a completely distributed mesh protocol where all the nodes can sleep and route their brother's packets. Nodes in this network support the DigiMesh protocol when the Xbee 802.15.4 Series 1 modules are used. The protocol is developed by Digi Corporation.

- XMesh

  The XMesh standard is developed by the Crossbow Corporation and supports full mesh along with over the air programming.

- Code Blue

  Code Blue is a medical sensor network platform developed by Harvard university sensor network lab. The protocol supports device discovery, publish/subscribe multi-hop routing and RF localization. (Hardvard University)

3.1.2 *IEEE 802.15.4a*

The IEEE 802.15a is an amendment to the 802.15.4 for an alternative physical layer. The main interest in providing an alternative physical layer are communications and high precision ranging / location capability (1 meter accuracy and better), high aggregate throughput and ultra low power. The amendment is adding scalability to data rates, longer range, and lower power consumption and cost. These additional capabilities are expected to enable significant new applications and market opportunities.

IEEE 802.15.4a specified four different PHYs, three of which utilized Direct-Sequence Spread Spectrum DSSS, and one which used Parallel Sequence Spread Spectrum PSSS. IEEE 802.15.4a also specified two additional PHYs using Ultra-wideband UWB and Chirp Spread Spectrum (CSS). The UWB physical layer has designated frequencies in three ranges; below 1 GHz, between 3 and 5 GHz, and between 6 and 10 GHz (Wikipedia, 2009).

Direct Sequence UWB was promoted by the ZigBee Alliance and ended up being added into the IEEE 802.15.4a standard. Direct Sequence UWB is spectrally efficient, can support precision ranging, and is very robust even at low transmission powers (Wikipedia, 2009).

The precision ranging capabilities of direct sequence UWB is one the main reasons the ZigBee alliance promoted this spread spectrum. Precision ranging is has very practical use in ZigBee especially in application areas such as healthcare. The first implementations of the 802.15.4a standard are expected to reach market in the first quarter of 2010; ZigBee is expected to update its standards shortly after.

The ability to determine node location is crucial to healthcare applications and ZigBee's lack of it has prompted academic and private institutions to develop property standards that provide this functionality. The most well know is Code Blue a healthcare oriented proprietary standard developed by Harvard University. The hope is that open standards such as ZigBee adopt the 802.15.4a and benefits from its improvement over 802.15.4.

3.2 **ZigBee**

The ZigBee standard defines a communication layer at level 3 and upper in the OSI model a seen in Figure 4 ZigBee / IEEE 802.15.4 OSI Model. Its main purpose is to create a network topology to let a number of devices communicate among them and

to set extra communication features such as authentication, encryption, association and upper layer application services. ZigBee was developed with the following considerations:

- Low cost
- Secure
- Reliable and self healing
- Flexible and extensible
- Low power consumption
- Easy and inexpensive to deploy
- Global with use of unlicensed radio bands
- Integrated intelligence for network set-up and message routing

(Daintree, 2008)

The low cost allows the technology to be widely deployed in wireless control and monitoring applications, the low power-usage allows longer life with smaller batteries, and the mesh networking provides high reliability and large ranges.

Members of the ZigBee alliance work on defining the network, security and application layers. They also work on providing interoperability and conformance testing specifications, managing evolution of the technology, building market awareness and creating public application profiles. Public application profiles allow multiple OEM vendors to create interoperable products. The current list of application profiles either published or in the works are:

- Home Automation (published)
- ZigBee Smart Energy (published)
- Commercial Building Automation (in the works)
- Telecommunication Applications (in the works)
- Personal, Home, and Hospital Care (in the works)

Application profiles can also be specified by the manufacturer for a specific application, these are referred to as private profiles.

3.2.1 *ZigBee Protocol Stack*

The ZigBee protocol stack has its origin in the Open Systems Interconnect (OSI) seven-layer model. A detailed ZigBee protocol stack is illustrated in the figure below. The two lower layers are defined by the IEEE 802.15.4 standard while the remaining two layers are defined by the ZigBee Alliance:

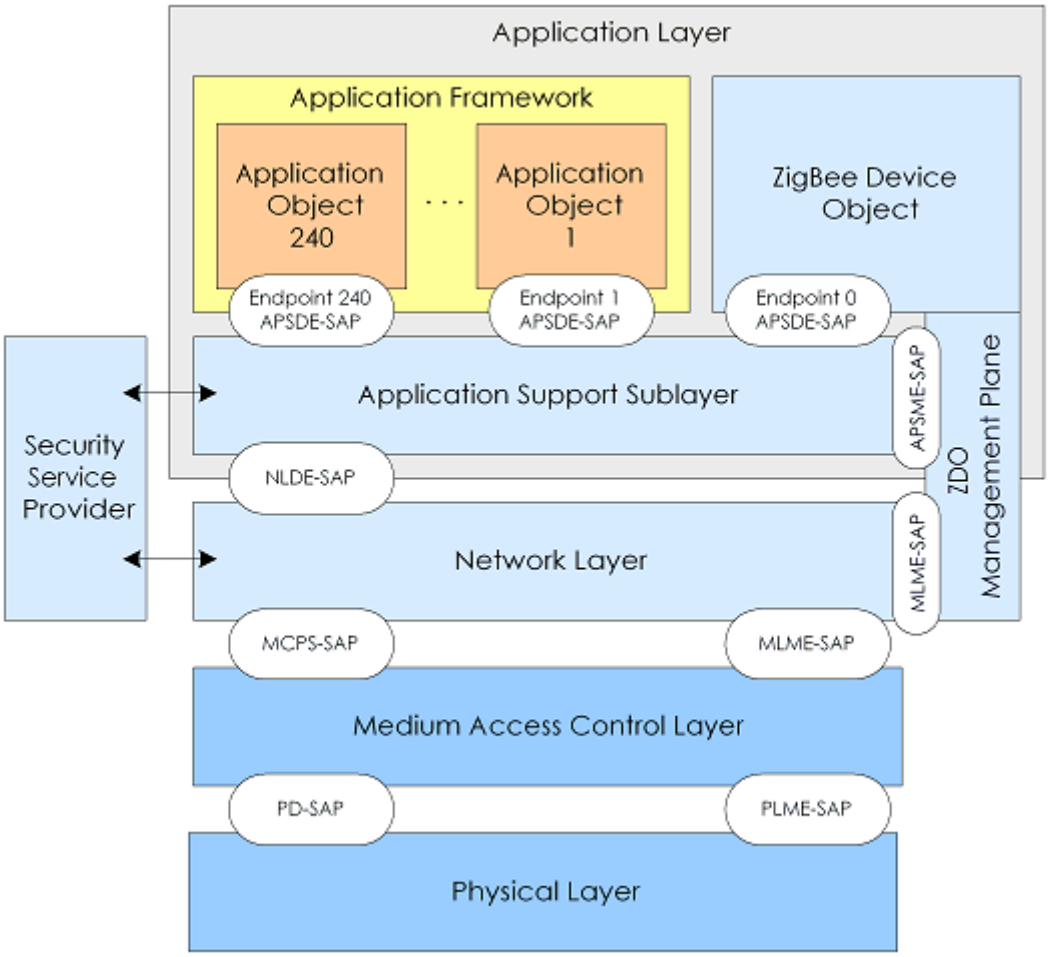

Figure 8 ZigBee Protocol Stack (Munk-Stander, 2005)

- Application Layer

The top layer in the ZigBee protocol stack consists of the Application Framework, ZigBee device (ZDO) and application support (APS) Sublayer.

Application framework

The application framework provides a description of how to build a profile onto the ZigBee stack. It also specifies a range of standard data types for profiles, descriptions to assist in service discovery, frame formats for transporting data and a key value pair construct to rapidly develop simple attribute based profiles.

Application objects

Application object are software modules at an endpoint that controls the ZigBee device. A single ZigBee node supports up to 240 application objects. Each application object supports endpoints numbered between 1 and 240.

ZigBee Device Object (ZDO)

It defines the role of a device within the network , either coordinator, router or end device. Furthermore it initiates and responds to binding and discovery requests, and established secure relationships between nodes in the network.

ZDO Management Plane

The ZDO management plane facilitates communication between the APS and NWK layers with the ZDO. It allows the ZDO to deal with requests from applications for network access and security using the ZigBee device profile.

Application Support Sublayer (APS)

The APS is responsible for providing a data service to the application and ZigBee device profiles. It also provides management services to maintain binding links and the storage of the binding table itself.

- Security Service Provider (SSP)

  The security service provider provides security mechanisms for layers that use encryption such as the network and application support layers. This is configured through the ZigBee device object.

- Network Layer (NWR)

  The network layer handles network address and routing by invoking action in the MAC layer. Its tasks include starting the network (preformed by the coordinator), assigning network addresses, adding and removing network devices, routing messages, applying security and implementing route discovery.

The following two layers being Medium Access Control and Physical layers are defined by the IEEE 802.15.4 standard and where described in the previous section.

3.2.2 *ZigBee and Xbee*

In the following section I will cover ZigBee networking, routing and sleeping functions specific to the Xbee Series 2 ZB firmware. The ZB firmware does not support IEEE 802.15.4 MAC beacon mode and therefore the topics will be constrained solely to non-beacon ZigBee concepts.

3.2.2.1 ZigBee Networking Concepts

ZigBee defines three different device types: coordinators, router and end device.

In order to start a network the coordinator must select a PAN ID (64 and 16 bit) and a suitable frequency channel. After this the coordinator behaves very much like router. Both the coordinator and routers can route data and allow other device to join the network. After an end device joins the network it transmits data through the router of coordinator it's connected to. The router or coordinator, that allowed an end device to join, becomes the parent. If the end device is configured to sleep, the parent will retain incoming packets destined for that end device in a buffer until it's awake and ready to receive. All devices have a universally unique 64 bit ID. Devices on a PAN receive a 16 bit network address when connecting to a PAN. The network address of the PAN coordinator is always 0.

Below is a list of the main characteristics of each device type.

Coordinator:

- Selects a channel and PAN ID (64 and 16 bit) to start the network.
- Can allow routers and end devices to join the network.
- Can assist in routing data.
- Cannot sleep and must be mains powered.

Router:

- Must join a ZigBee PAN before it can transmit, receive or route data.
- Can allow router and end devices to join the network.
- Can assist in routing data.
- Cannot sleep and must be mains powered.

End device:

- Must join a ZigBee PAN before it can either transmit or receive data.
- Cannot allow devices to join the network.

- Cannot route data i.e. must always transmit and receive data through its parent.
- Can sleep and be battery powered.

3.2.2.2 Non-beacon mode VS beacon mode

Beacon mode can be used to save power across the entire network. Beacon mode has the side effect of increasing complexity at the edges of the network i.e. end devices.

Beacon mode is a fully coordinated mode in which all devices know when to coordinate with one another. In this mode, the network coordinator will periodically "wake-up" and send out a beacon to the devices within its network. This beacon subsequently wakes up each device, who must determine if it has any message to receive. If not, the device returns to sleep, as will the network coordinator, once its job is complete.

Non-beacon mode, on the other hand, is less coordinated, as any device can communicate with the coordinator at will. However, this operation can cause different devices within the network to interfere with one another, and the coordinator must always be awake to listen for signals, thus requiring more power. A ZigBee network obtains overall low power consumption because the majority of network devices are able to remain inactive over long periods of time.

Beacon node is popular among environmental sensor networks where the entire network needs to be low powered, highly dynamic and distributable. Healthcare on the other side does not require a highly dynamic network and is very much static in nature. Unlike environmental sensor network routers and coordinators do not need to be moved and will always be mains powered. Again unlike environmental sensor networks end devices are highly mobile and battery powered. This makes networks where the complexity lies in the network and not in the edge, such as non-beacon networks, ideal for healthcare applications.

3.2.2.3 PAN ID

ZigBee networks are personal area networks. Each network is identified by a unique PAN identifier or PAN ID. The ID is shared among all devices on the same network. Devices can either be preconfigured with a PAN ID or they can discover nearby networks and select a PAN ID to join them.

The ZigBee protocol supports a 64-bit and a 16-bit PAN ID. Device on the same network share both 64-bit and 16-bit PAN ID. ZigBee networks operating within range of each other should have unique PAN ID's.

The 16-bit PAN ID is used by the 802.15.4 standard as a MAC layer addressing filed for all transmissions in the network. The limited addressing space (65535 ID's) means that there is a possibility that multiple networks within the same range could be using the same PAN ID. To resolve this potential problem ZigBee defines a 64-bit PAN ID.

The 64-bit PAN ID is meant to be a true unique identifier. It is used when devices join a network and is part of all beacons in order to perform 16-bit PAN ID conflict resolution. To resolve a 16-bit PAN ID, conflict resolution the networks must change their 16-bit PAN ID.

A coordinator can be preconfigured with a 64-bit PAN ID or select one at random when starting a network. If a device has a preconfigured 64-bit PAN ID it will only join a network with the same PAN ID. Otherwise a device will join any detected PAN and inherit the PAN ID from the network.

In this network coordinators are configured to select 16-bit PAN ID. Routers and end devices have preconfigured 64-bit PAN ID and will join a network with any 16-bit PAN ID as long as the 64-bit PAN ID is valid.

3.2.2.4 Application Support Sublayer

The APS layer adds support for endpoints, cluster ID's and application profiles.

3.2.2.4.1 Application Profiles

Application profiles describe a devices required functionality. A collection of device descriptions forms an application profile. Application profiles can be either public or private. Public profiles are defined by manufactures and developers while public profiles are define, developed and maintained by the ZigBee alliance. Public application profiles have a unique profile identifier.

By defining standard communication protocols and device functionality public profiles allow interoperable ZigBee solutions to be developed by independent OEMs.

The Xbee ZB firmware used in this project is currently configured to operate on a private profile developed by Digi called Drop-In Networking Profile.

3.2.2.4.2 Clusters

Clusters are used to define a specific function, service or action. Examples of application clusters are the On/Off and Level control functions defined in the home automation public profile.

Clusters have a 2-byte cluster ID. The ID is used in all application transmissions. Cluster often specify associated request and response messages. The ZigBee alliance defines a cluster library, for general use clusters that can be implemented in any profile. In order for a device to be complaint for a specific application profile it must correctly implement all required clusters for that application profile.

3.2.2.4.3 Endpoints

Endpoints are the analog of port in the TCP/IP protocol. An endpoint is defined by 1 byte value. Endpoints allow device to simultaneously support multiple application profiles.

3.2.2.5 Sleeping

End device are meant to be battery-powered and capable of sleeping. While sleeping end devices cannot receive data, in order to ensure a reliability communication routers and coordinators employ packet buffering and extended transmission timeouts.

When an end device is awake it sends poll request to its parent. When the parent receives a poll request it checks its buffered packet queue to see if it has a packet for the end device. The parent then send a MAC ACK back to the end devices indicating whether it has data for the end device. If the ACK states that there is data available for it, it will remain awake until it has received the data. Otherwise it will return to sleep mode. This mechanism allows the end device to turn off its radio receiver off when no data is expected. This dramatically reduces current consumption and ultimately extends and end device's battery life. (Digi International, 2009)

End point devices can be configured to sleep. Data sent to the device will be buffered in the parent router or coordinator. When the device wakes up it polls its

parent, the parent responds with buffered data, and the end point device goes back to sleep.

Routers and Coordinators maintain a child table with addresses of its end devices. They don't have more than a certain number entries in the child table; an end device will not be allowed to join unless that device has unused entries. Currently Xbee Series 2 modules can have up to 8 entries in their child tables.

Parents are responsible for buffering its children's incoming and outgoing data packages. A parent will buffer data destined to its children if:

- It runs out of buffer space
- Its child polls for the data
- Packet are stored for a period of time that exceeds buffer timeout

When an end device sends data to a remote device in the network it does so through its parent. The parent buffers the data packet until it can establish a route to the desired device. To do so the parent may need to perform a router or 16-bit address discovery request.

3.2.2.6 Mobile end devices

In an application such as healthcare end device mobility is crucial. A mobile end device that moves around the entire network would create child entries in every parent device greatly diminishing the potential size of the network. In order to avoid this problem parents must keep their child tables constantly updated, only containing entries for devices actually in use. This is done by the use of an end device poll timeout. If an end device does not send a poll request to its parent within the poll timeout, the parent will remove the end device form the child table. If it then wishes to transmit data it must first attempt to find an end device. It is therefore highly recommended to set the sleep time of an end device lower than that of its parent's poll timeout.

$$Sleping\ Time_{end\ device} < Poll\ timeout_{parent}$$

3.2.2.7 Sleeping modes

Xbee support two types of sleep; Pin sleep and Cyclic sleep.

Pin Sleep can be controlled by pulling up pin 9. While asleep the Xbee will not transmit nor receive data. The module will wake up and poll its parent every 100ms.

This allows an external microcontroller such as the Arduino to determine when the Xbee should be asleep and when it should be awake. When pin 9 is set high the microcontroller will finish the operation it is performing and go to sleep state. In the worst case scenario if the device is trying to join the network or is waiting for an acknowledgment the module could take a few seconds to enter sleep mode. Once the module enters sleep mode pin 11 On/Sleep is set low.

In cyclic sleep the Xbee module will start a timer when receiving serial or RF data. While awake it will poll its parent every 100ms. When new data arrives the timer is reset. A timeouts will force the module to go to sleep. When the Xbee end device wakes up it will poll for data, the time interval for awaking an end device should ideally not be configured to be more than that of the parent's buffer timeout. Ideally the sleeping time of an end device should be smaller than the buffer timeout of its parent.

$$Sleping\ Time_{end\ device} < Buffer\ timeout_{parent}$$

3.2.2.8 Network topologies

The nodes in a ZigBee network can be arranged using three different network topologies: star, tree and mesh.

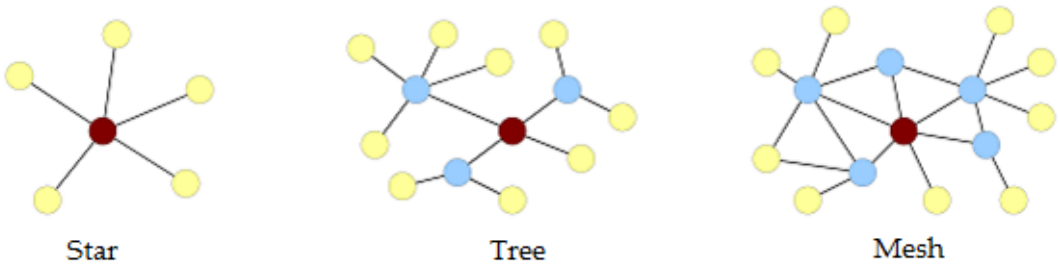

Figure 9 ZigBee network topologies (Munk-Stander, 2005)

The simplest of the three topologies is the star topology, shown in Figure 9 left. Here the ZigBee network contains one coordinator, no routers and a number of end devices. Each end device is within radio range of the coordinator.

In the tree topology, the communication routes are organized in such a way that there exists exactly one route from one device to another, see Figure 9 middle. End devices may either communicate directly with the coordinator or with exactly one of a number of routers.

As with the tree topology, end devices in a mesh communicate either directly with the coordinator or with a router. Unlike the tree topology, there may be several routes between different routers in a mesh topology. This redundant routing is transparent to the end devices, and introduces some reliability in the network, at the cost of added complexity. An example of a mesh network can be seen in Figure 9 right. (Munk-Stander, 2005)

Notice that the, the star topology is a subset of the tree topology, which itself is a subset of the mesh topology.

3.2.2.9 Diagnostics

Xbee modules can be configured remotely through API commands. API devices can send configuration commands to set or read the configuration settings of any device in the network. These commands were used during network setup and evaluation to determine the state of the network. The sub sections below describe the commands used during the evaluation phase.

3.2.2.9.1 Link Testing

A good way to measure the performance of a mesh network is to send unicast data through the network from one device to another to determine the success rate of many transmissions. To simplify link testing, Xbee modules support a loopback cluster ID on the data endpoint. Any data sent to this cluster ID on the data endpoint will be transmitted back to the coordinator.

3.2.2.9.2 RSSI

It is possible to measure the received signal strength on a device using the DB command. DB returns the RSSI value (measured in –dBm) of the last received packet. The DB value only indicates the received signal strength of the last hop. If a transmission spans multiple hops, the DB value provides no indication of the overall transmission path, or the quality of the worst link – it only indicates the quality of the last link and should be used sparingly.

The DB command value is measured in -dBm. For example if DB returns 0x50, then the RSSI of the last packet received was -80dBm. As of 2x6x firmware, the DB command value is also updated when an APS acknowledgment is received. (Digi International, 2009)

Power modes can configure in all modules except Xbee-PRO modules. Boost mode improves the receive sensitivity by 1dB and increase the transmit power by 2dB.

#	Xbee	Xbee-PRO
0	-8 dBm	-8 dBm
1	-4 dBm	-4 dBm
2	-2 dBm	-2 dBm
3	0 dBm	0 dBm
4	+2 dBm	+17 dBm

Tabel 1 Xbee Power Levels (Digi International)

3.2.2.9.3 Network Topology

The NC AT command can be used to determine the number of end devices that can join a device. This command was used in determining network topologies.

The Node Discover (ND) AT command is used to discover and reports all RF modules found. The following information is of great aid in determining the networks topology.

- PARENT_NETWORK ADDRESS (2 Bytes)
- DEVICE_TYPE<CR> (1 Byte: 0-Coordinator, 1-Router, 2-End Device) (Digi International, 2009)

4. Healthcare Monitoring

The cardiovascular and respiratory systems are key to an individual's health. Measurement of these systems are essential for any potentially unstable hospital patient. Telemedicine and home care are becoming increasingly viable scenarios for measuring these systems over extended periods of time.

In order to better understand the ECG waveform and Plethysmogram (PPG) I needed to learn basic principles of cardiovascular and respiratory systems.

The cardiovascular and respiratory systems are very complex systems and as such are very difficult to measure. Nevertheless measurements that gauge these systems performance are widely available. These are mainly ECG electrocardiogram and POX pulse oximeter. An electrocardiogram records the electrical current moving through the heart during a heartbeat. The study of an electrocardiogram is a subject of its own, with its own line of medical studies. The pulse oximeter measures the amount of oxygenation of red blood cells. Pulse oximeter produces a photoplethysmograph wave from.

Blood oxygenation and Heart rate can be used to measure the overall performance of the cardiovascular and respiratory systems. Sp02 and heart beats per minute (bpm) and are amongst the most pervasive measurement in the healthcare industry. (Pulse oximetry, 1998)

Measurement of these systems is traditionally preformed in controlled environments such as hospitals and ambulatory centers. Telemedicine enables continuous self monitoring, but without measurement of the largely uncontrolled environments in which the measurements where conducted wrong assumption on the data could be easily made.

Everything belongs in a context; data without context has the potential to be grossly misinterpreted. In hospital environments the context is mostly know, out in real world everyday lifetime situations context various greatly. It is therefore crucial to try and provide as much insight into the context as possible.

I have therefore included sensors to measure external factors such as humidity, pressure, temperature, ambient light, and activity sensors along with widely accepted physiological measure provided by the ECG and Pulse Oximeter.

4.1 ECG

Blood returning from the body is sent to the heart and then to the lungs to pick up oxygen and release carbon dioxide, the oxygenated blood is collected, then sent to the heart and back to the body, where oxygen is liberated and carbon dioxide is collected. The heart's pumping action comes from a built-in, electrical, conduction system. (Villafañe)

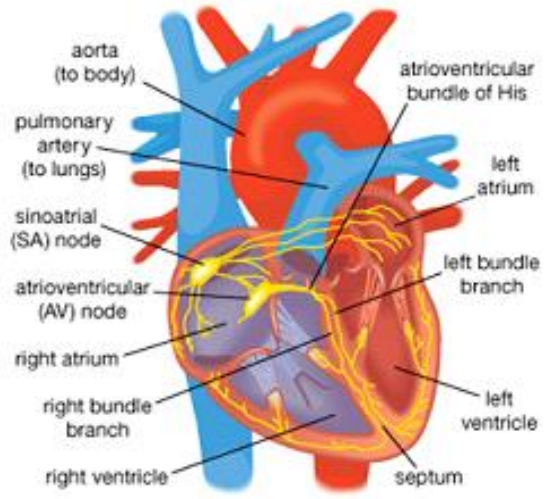

Figure 10 Heart Compartments and electrical system (Villafañe)

An electrocardiogram records the electrical current moving through the heart during a heartbeat. The current's movement is divided into parts, an illustration of an ECG period can be seen in Figure 11 ECG Period P-QRS-T.

Each heartbeat begins with an impulse from the heart's pacemaker; sinoartial node. This impulse activates the upper chambers of the heart (atria). The P wave represents activation of the atria.

Next, the electrical current flows down to the lower chambers of the heart (ventricles). The QRS complex represents activation of the ventricles.

The electrical current then spreads back over the ventricles in the opposite direction. This activity is called the recovery wave, which is represented by the T wave. (Paul H. Tanser, MD, 2006)

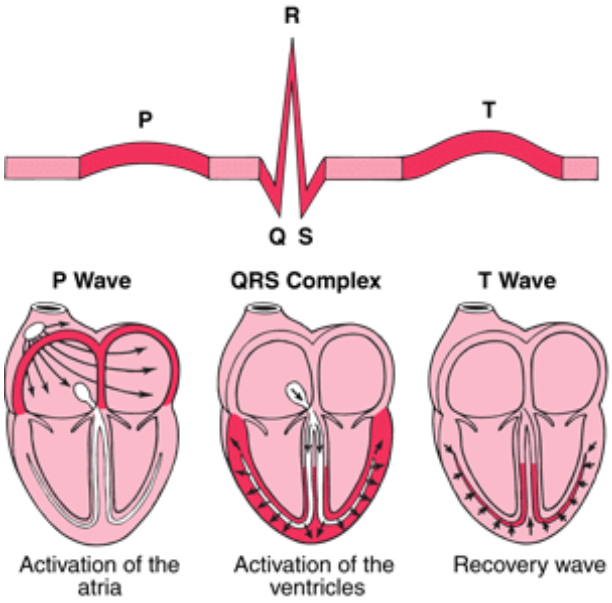

Figure 11 ECG Period P-QRS-T (Paul H. Tanser, MD, 2006)

Many abnormalities can be seen on an ECG. They include among others:

- Arrhythmia - An abnormal heart rhythm
- Ischemia - An inadequate supply of blood and oxygen to the heart
- Hypertrophy - Excessive thickening of the heart's muscular walls.
- Myocardial infarction - A previous heart attack

Arrhythmias are abnormal heartbeats such as an extra heartbeat (premature contraction), a fast heart rate (tachycardia) or a slow heart rate (bradycardia). Arrhythmias in old people can be life-threatening while arrhythmias in young people with normal hearts are usually benign and require no treatment at all.

Usually, an ECG is obtained if a heart disorder is suspected. It is also obtained as part of a routine physical examination for most middle-aged and older people, even if they have no evidence of a heart disorder. It can be used as a basis of comparison with later ECGs if a heart disorder develops. Abnormal heart rhythms and inadequate blood flow to the heart muscle may occur only briefly or unpredictably. (Paul H. Tanser, MD, 2006)

To detect such problems, doctors use continuous ambulatory ECG (Holter systems), in which the ECG is recorded continuously while the person engages in normal daily activities.

4.1.1 *EG 0100*

The EG 01000 is a one channel ECG module. The power consumption of the module is between 25 and 30mA while transmitting. The dimension of the module is 77x53x10 mm see Figure 12. The board has galvanic isolation and can measure pulse between 30 and 250 beats per minutes (averaged over 8 beats). The board can be configured with 3 amplification stages, 3 speed transmissions and 2 notch filter frequencies. The module also has a lead-off detector as well as a calibration signal and a simulation mode. Transmissions to the module are done through serial asynchronous communication at 9600 baud, 8 data bits, one stop bit, no parity bit.

The ECG channel is measured over a three lead cable. This is done by measuring the differential input voltage between two leads cables. The leads carry the differential voltage to the input stage of the amplifier, the third lead, is the isolated ground of the module. (Medlab, 2008)

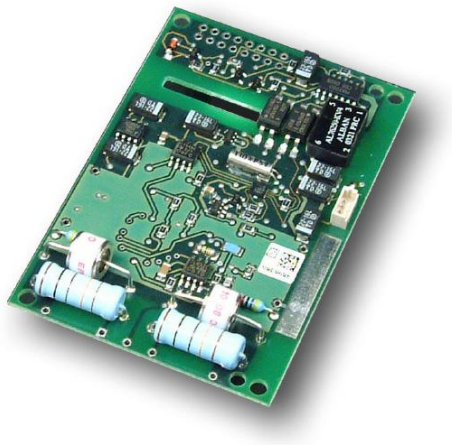

Figure 12 EG 01000

The board is connected to the microcontroller through serial CMOS connection (0,5volts) a RS232 connection of +5,-5 volts is available. The EG01000 sends a continuous data stream and is able to receive serial commands.

Data Stream

The board has an internal pulse detection algorithm. Each time a pulse is detected a pulse maker is transmitted along with the pulse value.

The ECG wave sample points are transmitted continuously and values lie between 0 and 246. Values that are higher than 246 are used as markers. Wave markers are used for marking the following byte as a new data values:

Marker byte	Description
0xF8	Wave
0xFA	Pulse
0xFB	Info

Tabel 2 Market bytes

The only info byte defined is 0x11 used to inform when the ECG lead is not connected. The neutral line of the ECG always lies at 128, the module transmits unsigned byte data. While a lead off condition is active, a pulse value of 0 and a neutral line of 128 is transmitted. Below is a sample data stream.

Figure 13 EC 01000 Data Frame (Medlab, 2008)

Commands

Commands are one byte characters, some of them have one byte as additional parameter, and others are executed after the ECG board receives the command byte alone.

The sampling frequency for ECG signal can be configured to be 300 Hz, 100 Hz or 50 Hz. The command that changes the sampling frequency is “S” as in speed. This command takes one byte between "0" and "2" as parameter.

The board offers 3 amplification stages that can be configured through the amplification command "A". This command requires one parameter between "0" and "2". For the low amplification stage a 1mV signal lies in between 110 and 147 , in medium amplification it lies between 90 and 165 and in high amplification it lies between 50 and 200.

For evaluation purposes the board can be configured to output a previously stored ECG waveform. The command "M" brings the module into an internal simulation mode, issuing the "N” command returns the module to its normal state. Simulation mode proved extremely helpful during development phases of the project.

In order to calibrate the microcontroller the board can be set to output a 1mw reference curve, it does so for about a second. After this the module returns to sending ECG waveforms. This is done by issuing the "C" command.

The module has an internal notch filter that passes all frequencies except those in a stop band centered on a center frequency. The center frequency can be set to be either 50 Hz or 60 Hz by issuing a "5" or "6" command respectively.

The default factory configuration of the module is normal transmission mode with lowest amplification stage, 300Hz sampling frequency and the notch filter is set to 50 Hz. The table below shows all available commands:

Command	Description
N	Normal mode
S 0	Simulation mode
S 1	ECG sampling frequency 300 Hz
S 2	ECG sampling frequency 100 Hz
A 0	ECG sampling frequency 50 Hz
A 1	Amplification stage 0
A 2	Amplification stage 1
A 3	Amplification stage 2
5	50 Hz notch filter
6	60 Hz notch filter
C	Calibrate

Tabel 3 ECG 01000 Serial Commands

The code implementing the communication between microcontroller and the EG 0100 board is included in the CD.

4.2 Pulse-oximeter

A Pulse-oximeter is capable of indirectly measuring the oxygen saturation of a patient's blood and changes in blood volume in the skin, producing a PPG waveform. The heart rate can also be determined by the PPG waveform using signal analysis. The device basically calculates the ratio of red to infrared light absorption of pulsating components at the measuring site (usually finger or ear lobe).

Figure 14 Oxilink

The Pulse-oximeter used in this project is a Oxilink produced by Smith-Medical. The device consists of a finger sensor and an electronic board.

The OxiLink oximeter determines the pulse rate and SpO_2 by the passing two wavelengths of light, one red 660nm, 2.0mW and one infrared 905nm, 2.0-2.4mW, through the fingers body tissue to a photo-detector.

The signal strength from the photo-detectors resulting from each of the light sources depends on the color and thickness of the finger tissue, the sensor placement, the intensity of the light sources, and the absorption of the arterial and venous blood, including the time varying effects of the pulse in the finger tissue. The OxiLink then processes these signals, separating the time invariant parameters such are tissue thickness, skin color, light intensity, and venous blood from the time variant parameters such as arterial volume and SpO_2 to identify the pulse rate and oxygen saturation (BCI Inc., 2002).

Oxygen saturation calculations can be performed due to the fact that oxygen saturated blood predictably absorbs fewer red lights than oxygen depleted blood. This relationship is dictated by Beers-Lambert law that describes the transmission, absorption and depression of light that passes through the hemoglobin (Derek Densmore, 2006).

The figure below illustrates the basic operation of a pulse oximeter.

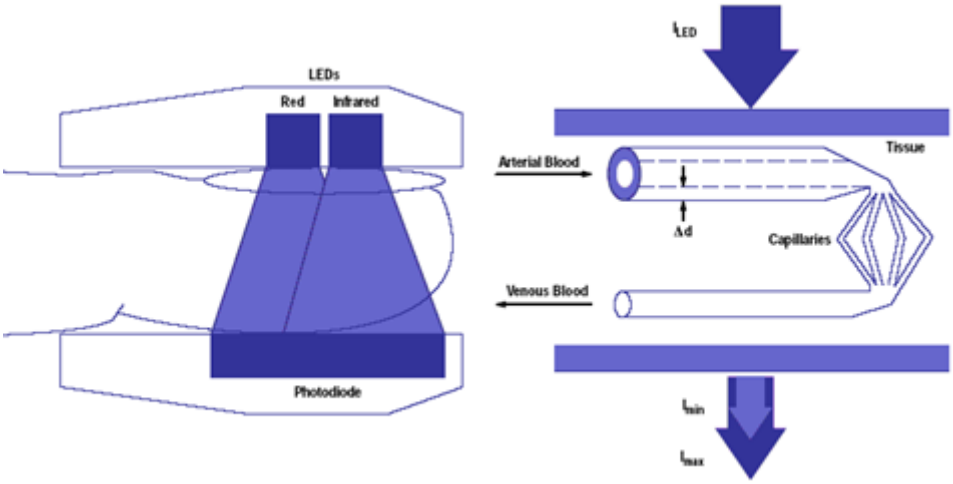

Figure 15 Pulse oximeter operation (Derek Densmore, 2006)

Each time the heart beats the heart contracts and there is a surge of arterial blood, which momentarily increases arterial blood volume across the measuring site. This results in more light absorption. The light signals received at the photo detector is therefore seen a waveform with peaks at each heartbeat and slopes between heartbeats.

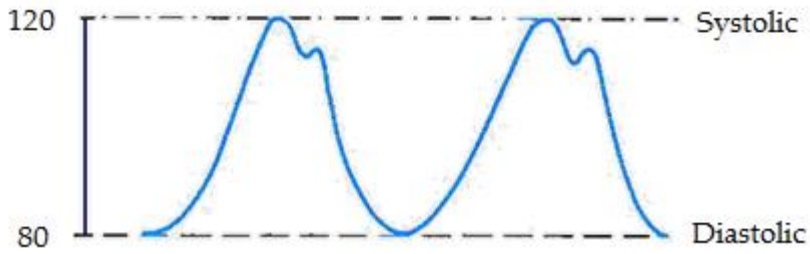

Figure 16 PPG Waveform

In continuous measurements scenarios such as the one presented in this project it becomes interesting to measure changes and abnormalities in the characteristics of the PPG over time. These changes will likely represent a problem in the patient's cardiovascular system. The waveform has other qualities from which respiratory rate and the state of the cardiovascular system can be obtained. Certain diseases cause characteristic changes in these qualities.

4.2.1 *Interfacing with the Oxilink board*

The Oxilink board communicates with the host computer through a single asynchronous serial channel at CMOS voltage levels (4800 baud, 8 bits, 1 stop bit, No parity).

The board provides %SpO2, Pulse Rate, Signal Strength, Bargraph, Plethysmogram, and Status Bits data. The board can receive two commands, RESET and SYNC.

The board transmission rate is of 60 packs per second. Each pack consists of four data bytes. In order to determine the first byte the MSB (bit 7) of every byte is used as a marker. A value of 1 at bit 7 is reserved for the first byte of every packet.

The structure of the data pack is shown in the table below:

Byte 1	Byte 2	Byte 3	Byte 4
Info	PPG Waveform	Bar Graph	Processed Data

Tabel 4 Data Pack

A detailed description of the contents of each byte in the data pack is shown below:

Bit	Description
Byte 1	
0	Pulse beep signal
1	Processed data address, bit 0
2	Processed data address, bit 1
3	Processed data address, bit 2
4	0 – Unused
5	0 – Unused
6	1– if bytes 1, 2, 3 are used for the software revision number, 0 – if bytes 1, 2, 3 are used for Oximetry data
7	1 - Marker bit
Byte 2	
0-6	Real time PPG (Plethysmogram) waveform (range 0-100)
7	0 – Unused
Byte 3	
0-3	Real time Bar Graph (range 0-15)
4	0 – Unused
5	0 – Unused
6	Pulse rate MSB (bit 7)
7	0 – Unused
Byte 4	
0-6	Value of these bits depends on the 3 bit address field defined in Byte 1

7	0 – Unused

Tabel 5 Data Pack byte description

The table below describes the values of the first 7 bits of Byte 4.

Processed Address	Description
0	SpO2 (range 0-99).
1	Pulse rate first 6 bits. MSB bit 7 is defined in Byte 3
2	Signal Strength
3	Alert Condition: 0 –No alerts 1 – Sensor is unplugged 2 – No finger in sensor or sensor problem 3 – Searching for pulse 4 – Searching too long 5 – Lost pulse
4	Instantaneous SpO2 (w/o board averaging preformed)

Tabel 6 Byte 4 data format

The code implementing the communication protocol between the microcontroller and the Oxilink board is included in the appendix CD Contents.

4.3 Accelerometer

Accelerometers are normally used to perform motion detection. The list below show some common used of accelerometers in the medical industry.

- Limb motion analysis of Parkinson's Disease and stroke patients
- Detect Seizures , Epileptic Patients
- Posture Analysis
- Sleep Studies
- Activities of Daily Living Studies
- Fall detection

The figure below shows the footprint of a breadboard with a mounted ADXL345 3-axis accelerometer used in this project.

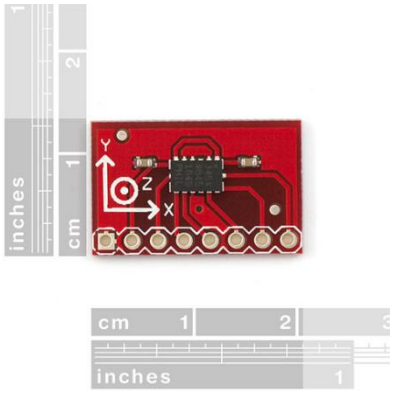

Figure 17 ADXL345 Footprint (Sparkfun Electronics, 2009)

The ADXL345 is a small, low power, 3-axis accelerometer with high resolution 13-bit measurement at up to ±16 g. Digital output data is formatted as 16-bit twos complement and is accessible through either a SPI or I^2C digital interface.

The device can measure the static acceleration of gravity in tilt-sensing applications, as well as dynamic acceleration resulting from motion or shock. Its high resolution enables measurement of inclination changes less than 1.0°.

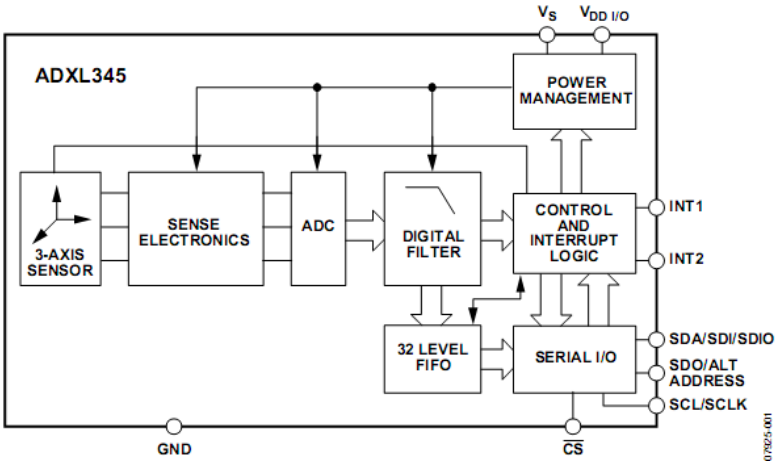

Figure 18 ADXL345 Functional Block Diagram (Analog Devices, 2009)

Several special sensing functions are provided. Activity and inactivity sensing detect the presence or lack of motion through user set acceleration thresholds. Tap sensing detects single and double taps. Free-fall sensing detects if the device is falling. These functions can be mapped to interrupt pins INT 1 and INT 2.

Low power modes enable intelligent motion-based power management with threshold sensing and active acceleration measurement.

The accelerometer is being used as a free fall and motion detection. This is done via the on board free fall and activity detection function of the device. As previously mentioned the device provides two output pins for driving interrupts INT1 and INT2. All functions can be used simultaneously. Interrupts are enabled by setting the appropriate bit in the INT_ENABLE register and are mapped to either the INT1 or INT2 pin based on the contents of the INT_MAP register.

The activity bit is set when acceleration greater than the value stored in the THRESH_ACT register is experienced. The inactivity bit is set when acceleration of less than the value stored in the THRESH_INACT register is experienced for more time than is specified in the TIME_INACT register. The maximum value for TIME_INACT is 255 sec (Analog Devices, 2009).

The FREE_FALL bit is set when acceleration of less than the value stored in the THRESH_FF register is experienced for more time than is specified in the TIME_FF register. Unlike inactivity detection free fall always uses the acceleration of x, y and z to detect both linear and rotational free falls (Analog Devices, 2009).

All interrupts expect Activity and Free fall are disabled by setting the INT_ENABLE register to HEX 14. The free fall interrupt is mapped to INT1 and the Activity

interrupt is mapped to INT2 in the INT_MAP register, this was done by setting the value HEX 14. Detection activity and free fall is then a simple matter of connecting the interrupts to the microcontroller.

4.4 Light Sensor

The light sensor uses a Light Dependent Resistor LDR. An LDR this is a simple component that provides a variable resistance that changes with light. The resistance falls predictably with increasing light intensity.

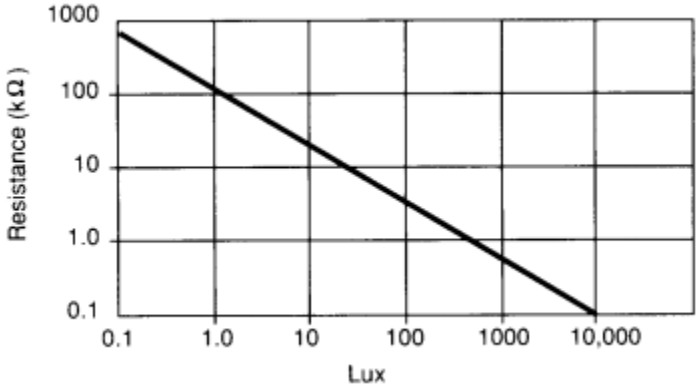

Figure 19 Resistance vs. Illumination (RS, 1997)

As a matter of reference Sunlight is ~10,000 LUX and Moonlight ~0.1 LUX.

The light sensor uses an LDR as part of a voltage divider. A voltage divide is a simple circuit used to determine the voltage drop in a series of resistors. The analog input will vary between 0 (for 0V) and 1023 (for 5V).

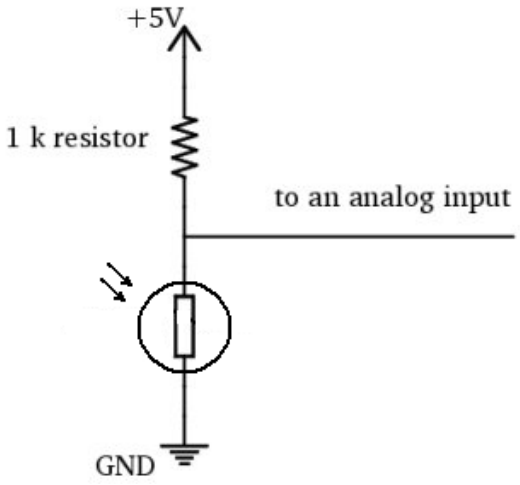

Figure 20 Light Sensor Circuit Diagram

4.5 Barometric Pressure Sensor

Barometric pressure is defined as the force per unit area exerted against a surface by the weight of air above that surface at any given point in the Earth's atmosphere. Changes in barometric pressure can have severe implications to our health.

Research shows that migraines are more likely when pressure rises. These types of headaches are often referred to as barometric pressure headaches. (Vergano)

Research also shows that there is relationship between drops in barometric pressure and the occurrence of heart attacks. Results confirmed that heart attacks increased within 24 hours of an atmospheric pressure drop (And the bigger the drop in pressure, the bigger the danger) (Relation of atmospheric pressure changes and the occurrences of acute myocardial infarction and stroke., 2005)

The figure below shows the footprint of a breadboard with a mounted MEMs Barometric Pressure Sensor SCP1000 used in this project (VTI Technologies)

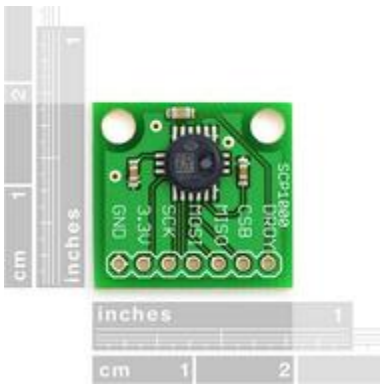

Figure 21 Barometric Pressure Sensor Footprint (Sparkfun Electronics, 2009)

The SCP1000 is the very first absolute pressure sensor on the market to use MEMs technology to grant 17-bit resolution. Under ideal conditions, this sensor can detect the pressure difference within a 9cm column of air.

The sensor is intended for barometric pressure measurement and altimeter applications for 30kPa...120kPa and -20C...70C measuring ranges. The pressure and temperature output data is calibrated and compensated internally (Oxer, 2009). The module communicated with the microcontroller via an SPI interface.

The code implementing the communication between microcontroller and the SCP1000 is included in the appended CD.

4.6 Humidity and Temperature Sensor

Research shows that continuous measurement of ambient humidity and temperature can help prevent respiratory disease and even death in the elderly. A review of the health effects of relative humidity in indoor environments suggests that relative humidity can affect the incidence of respiratory infections and allergies. (Indirect health effects of relative humidity in indoor environments., 1986 Mar).

Being constantly exposes to minor degrees cold can have serious health consequences. Professor Bill Keating, an expert from Queen Mary University of London, has studied the issue extensively. In England and Wales alone more than 23,000 die every year as a result of being constantly exposed to minor degrees of cold. Most deaths are not due to hypothermia, most of them are due to strokes and heart attacks. This is because the blood becomes more liable to clot in people who are exposed to the cold. (Roberts, 2006)

24 °C	Top range of Comfort
21 °C	Recommended living room temperature
>20 °C	Risk begins
18 °C	Recommended bedroom temperature
16 °C	Resistance to respiratory dieses weakened
12 °C	More than 2 hours in this temperature raises blood pressure and increases heart attack and stroke risk
5 °C	Significant risk of hypothermia

Tabel 7 Temperature effects on comfort and health (Roberts, 2006)

The figure below shows a simple breakout board for the SHT15 humidity sensor and temperature sensor used in this project.

Figure 22 SHT15 Footprint (Sparkfun Electronics, 2009)

The SHT15 digital humidity and temperature sensor is fully calibrated and offers high precision and excellent long-term stability. The digital CMOSens Technology integrates two sensors and readout circuitry on one single chip.

Features:

- 2 factory calibrated sensors for relative humidity & temperature
- Digital 2-wire interface
- Precise dew point calculation possible
- Measurement range: 0-100% RH
- Absolute RH accuracy: +/- 2% RH (10...90% RH)
- Repeatability RH: +/- 0.1% RH
- Temp. accuracy: +/- 0.3°C @ 25°C
- Fast response time < 4 sec.
- Low power consumption (typ. 30 µW)

(Sensirion, 2002)

Connecting the SHT15 to the Arduino is simple. The DATA and SCK lines each go to a digital input/output on the Arduino and the VCC goes to +5V.

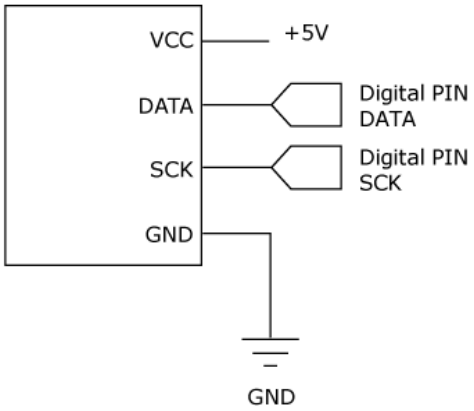

Figure 23 SHT15 Connection Diagram (Sensirion, 2002)

Practical Arduino a project dedicated to creating standard libraries for the Arduino has created a library that wraps all functionally needed to extract data from the SHT15 temperature / humidity sensor. The functions below are used to read temperature and humidity values from the sensor. (Oxer, 2009)

```
humidity = sht1x.readHumidity();

temp_celc = sht1x.readTemperatureC();

temp_f = sht1x.readTemperatureF();
```

5. Multitier Architecture

In the previous two sections I discussed the Wireless Networking protocols and Health monitoring sensor used in the project. The upcoming sections focus on describing the system architecture. This includes the architecture of the wireless nodes, the gateway and data client architecture.

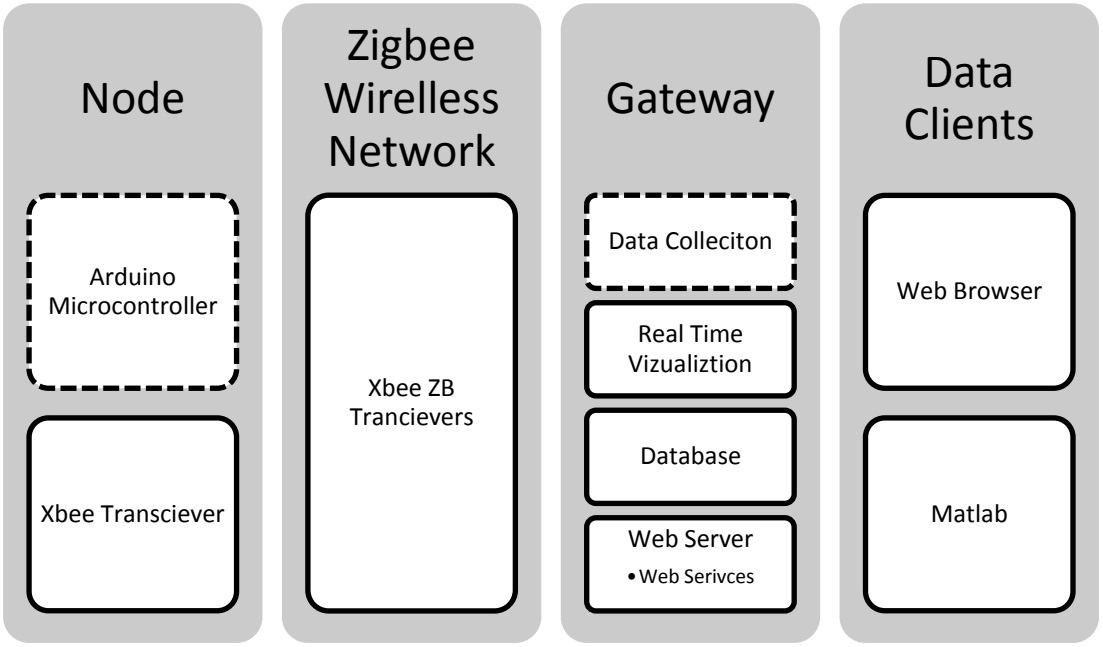

Figure 24 System Architecture

A data transmission protocol has been defined. The protocol specifies the structure and workflow of packets being transmitted over the wireless network. The protocol is implemented both in the individual nodes and in the data collection module (dashed in the figure above).

These sections intend to provide an overview of the system and not a walkthrough of the implemented code. The interested reader can analyze the code which is included in the appendix Xbee configuration.

5.1 Data transmission protocol

In order to transfer sensor reading from the wireless nodes over the networks and into the database a data transmission protocol needed to be defined.

The protocol defines the structure and workflow of the data packets being sent over the wireless sensor network.

The protocol supports two types of frames. An identification frame sent during initialization, and a data frame. A data frame can only be sent once the initialization frame as been successfully transmitted.

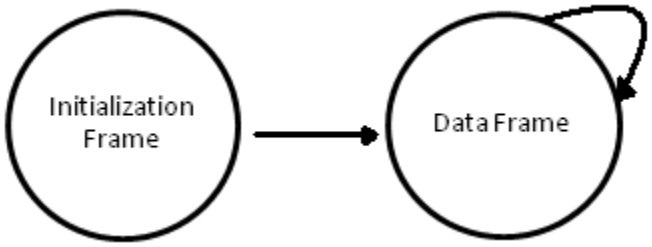

Figure 25 Protocol Workflow

5.1.1 *Initialization frame*

Before data can be stored in the database the node needs to identify itself. The identification frame can include the name of the nodes, the number of sensor attached to it and any meta-data for sensor readings such as data type, data format and sampling frequency.

The identification frame is sent, once a node has joined a network. The identification frame is sent reliably, with acknowledgments. Unlike standard data frames that do not require acknowledgments, due to significant reduction in transmission performance.

The main design principle behind the decision to provide initialization frame containing node meta-data is decentralization. Information about the state of the node and sensors attached to it is stored on the edge of the network. This decentralized architecture has the advantages that it's easy to deploy requires no setup or management.

An initialization frame consist of on Node identifier pack and one or more sensor pack. A pack consists of a marker followed by a format identifier and a value. A pack is used to wrap a both identify a format a value correctly from a data stream. The table bellow illustrates the format of a data pack.

Marker	Format	Value
1 byte	1 byte	1 or more bytes
Used to identify the pack	Specified the value format. 0 - String 1 - Integer 2 - Float	Holds the pack value

Tabel 8 Data pack

The format, as its name suggests, is used to specify the format of the value in the pack. The format can currently either be of types Integer, Float and String.

Supporting various data types introduces the requirement for bit marker collision avoidance schemes. Marker collision avoidance schemes are used in many will widely used protocols as MIME and IMAP. In order to avoid marker collision, which would lead to misclassification of data packs the protocol, reserves some character for markers. If any value in the data pack contains that marker bit value, the value is simply repeated in the frame. This simple scheme indicates that the value itself was meant to be sent and not a marker. If 3 marker bytes are present, then the protocol appends another marker byte to indicate that 3 bytes are meant to be transmitted and not 3 markers. A marker is only identified as a marker if accompanied by values different than itself. Marker collision decoding is preformed on the Gateway.

Currently the format can only be unsigned Integers and/or Float. This simple identification format allows for dynamically sized data types to be transmitted. A data pack can have a minimum size of 3 bytes and a maximum size of 32 bytes. The data pack will have a size of 3 bytes when sending a string with 1 character (char) or integer value less than 255. If needed the protocol could easily be extended to support signed Integers and/or Float. Unlike other packs, the value in the data format pack is used to specify the number of bytes used to transmit the sensor value without taking into account marker collisions.

The figure below shows the structure of a node initialization pack along with its sensors.

Node Pack	Sensor	Sensor	…

Tabel 9 Initilizaton Frame

The sensors components consist of various packs, which are used to identify the sensors, its data format, its name and sampling frequency. A maximum of 6 sensors are allowed per node. This is due to the number of reserved markers in the data frames note that due to the use of markers the value of a Sensor Maker pack will need marker collision.

Sensor Name	Data Type	Data Format	Sampling Frequency	Sensor Marker

Tabel 10 Sensor Values Packs

The table below shows the contents of the node identification frame used in the ECG enabled node.

	Marker	Format	Value
Node Name	250	0: String	"ECGNode"
Sensor Name	251	0: String	"ECG_Waveform"
Data Type	252	0: String	"mV"
Data Format	253	1: Integer	1
Sampling Frequency (Hz)	254	1: Integer	50
Sensor Marker	255	1: Integer	250
Sensor Name	251	0: String	"ECG Heartbeat"
Data Type	252	0: String	"BPM"
Data Format	253	1: Integer	1
Sensor Marker	255	1: Integer	251

Figure 26 Sample ECG Initialization Frame

The initialization frame is stored in the database tables Sensor and Node described in section 7.2 Database Architecture.

5.1.2 *Data Frame*

Unlike the initialization frame the data frame has a very simple format. The data frame has a minimum amount of meta-data and consists of 1 byte data markers each followed by the values corresponding to the marker.

Marker	Value	Value	…	Marker	Value	…

Tabel 11 Data Frame Fromat

Since the initialization frame defines the meta-data for the marker, the Gateway can process the data stream simply by finding the markers, splitting the values to the corresponding size and parsing them to the correct format.

Data from the data frame is stored in the readings table once the data source has been determined (using the Xbee api discussed in section 7.1 Real Time Data View) The readings table is described in section 7.2 Database Architecture.

5.1.2.1 Buffering

In some scenarios a node might need to buffer the data from a sensor instead of sending it as it's available. The ECG and POX nodes buffer the readings, mostly due to the high data rates of these sensors (50HZ – 150 Hz).

The size of the buffer is selected by the node and transmission is executed if the buffer is full or the transmission time out is reached.

Node with low data rates such as barometric sensor node and the temperature node will not fill the buffer with data, but instead the data will simply be transmitted over the network every transmission timeout.

A node can have up to 6 sensors attached to it, and each of them can be sampled at different frequencies, but all will share a common transmission timeout.

6. Node Architecture

The nodes consist of an Arduino microcontroller board and an Xbee transceiver. The figure below shows the current nodes with and without Xbee transceiver.

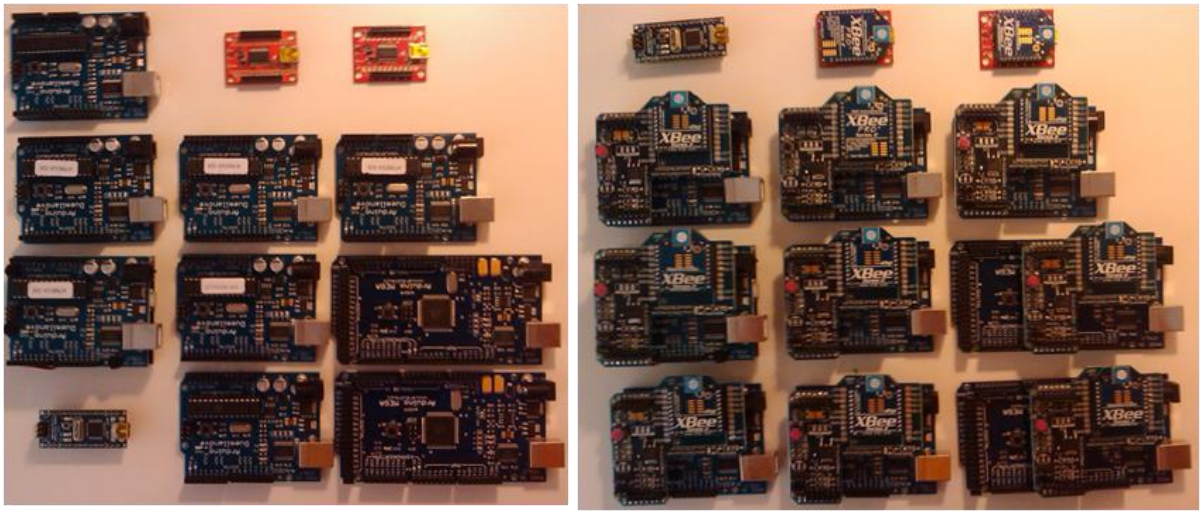

Figure 27 Sensor Nodes

6.1 Arduino

As already mentioned, Arduino is an open-source inexpensive computing platform based on a microcontroller board, and a simple cross platform development environment. Arduino's main advantages over other microcontroller platforms are its modularity and community developed open tools and software libraries.

Figure 28 Arduino Duemilanove

Arduino boards are relatively inexpensive compared to other microcontroller platforms. The least expensive version of the Arduino module can be assembled by hand, and pre-assembled Arduino modules cost less than 250 DKK.

The Arduino development environment is written in Java and runs on Windows, Macintosh OSX, and Linux operating systems. Most microcontroller development environments are limited to Windows.

The Arduino programming environment is easy-to-use for beginners, yet flexible enough for advanced users to take advantage of as well. It is based on the Processing programming environment.

All Arduino software and is published as open source and as such can be extended by experienced programmers. The Arduino programming language is based on Wiring and be expanded through C++ libraries. Alternatively one can choose to program Arduino using the AVR-C programming language on which the Arduino programming language is based. AVR-C code can be added directly into an Arduino program.

The Arduino is based on Atmel's ATMEGA microcontrollers. The diagrams for the boards are published under a Creative Commons license, so circuit designers can make their own version of the module, extending it and improving it. There are

currently many microcontroller boards available that have improvements on size and power consumption and are based on the Arduino design.

6.1.1 *Arduino Programming Language*

The Arduino language is based on C/C++ and supports all standard C constructs and some C++ features. Programs can be divided in three main parts: structure, values (variables and constants), and functions. It links against AVR Libc and allows the use of any of its functions. AVR Libc is a Free Software project whose goal is to provide a high quality C library for use with GCC on Atmel AVR microcontrollers.

Programming is usually carried out through the Arduino IDE (Integrated Development Environment) which is a cross-platform application written in Java. The IDE is derived from the Processing IDE made by the Wiring project (http://processing.org/).

The IDE is designed to introduce programming newcomers unfamiliar with software development. It includes a code editor with features such as syntax highlighting, brace matching, and automatic indentation, and is also capable of compiling and uploading programs to the board with great ease. A program is first cross-compiled through avr-gcc, and then downloaded to the attached Arduino board. The IDE can act as a serial terminal emulator, allowing interaction with the running board. A screen shot of the IDE is shown below.

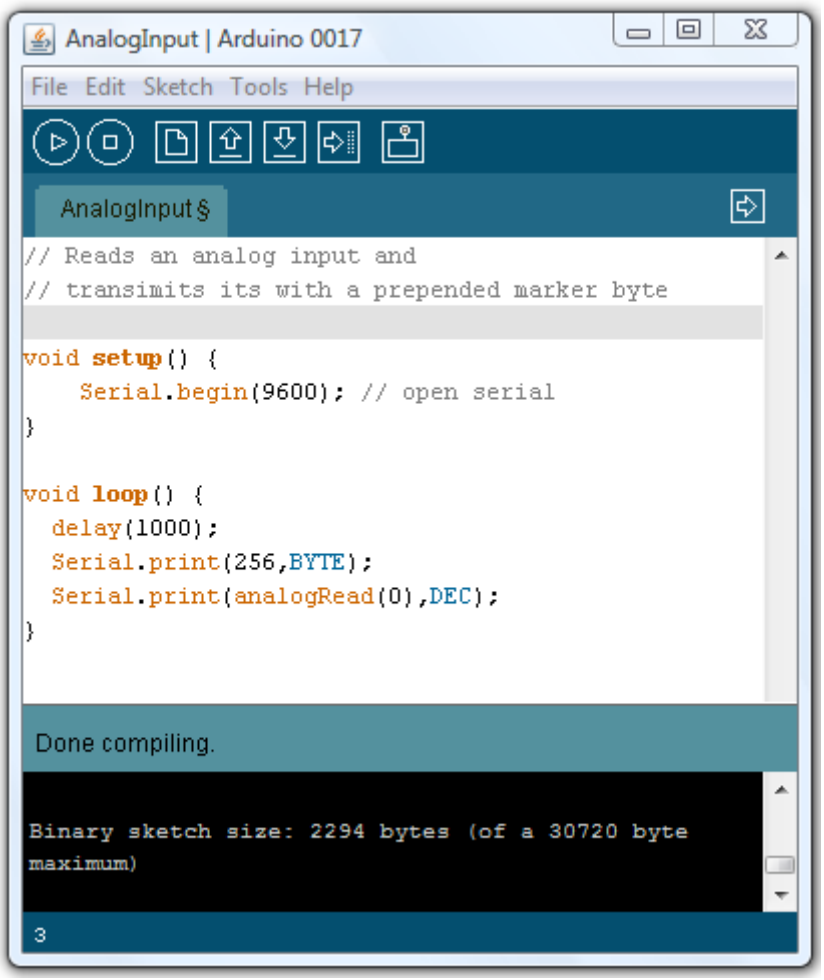

Figure 29 Arduino IDE

The figure above shows the code required to read and transmit an analog reading, As seen on the figure above only two functions are needed in order to make a program:

- setup() : The setup function is run once at the start of a program which can be used to initialize settings. The initialization frame is sent in this block. The block does not terminate until an acknowledgment is received.

- loop() : The loop function is called repeatedly until the board is powered off. Here sensor values are obtain and either buffered or sent immediately over the network.

6.1.2 *Arduino Boards*

All Arduino boards are open source, and new boards are constantly being developed by member of the community. In this project I am using the Arduino Duemilanove, the Arduino Mega and the Arduino Nano (shown below).

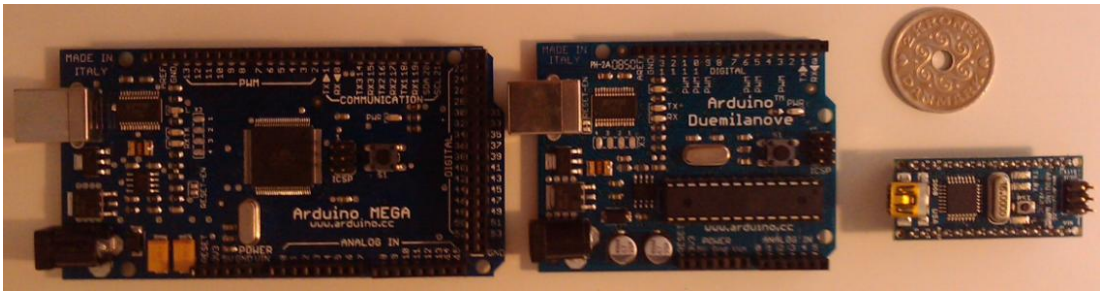

Figure 30 Arduino Boards

All Arduino boards contain everything needed to support the microcontroller; they can simply be connect to a computer with a USB cable or powered with a AC-to-DC adapter or battery to be used. All pins in the board are conveniently labelled.

The Arduino Mega (left Figure 30) is a microcontroller board based on the ATmega1280. It has 54 digital input/output pins (of which 14 can be used as PWM outputs), 16 analog inputs, 4 UARTs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. The Mega is compatible with most shields designed for the Arduino Duemilanove.

The Arduino Duemilanove (middle Figure 30) is a microcontroller board based on the ATmega168 or ATmega328. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button.

The Arduino Nano (right Figure 30) is a small breadboard-friendly microcontroller board based on the ATmega328 or ATmega168. Besides lacking a power jack and having a different USB connector the board has the same functionality as the Arduino Duemilanove; just in a different package.

6.1.2.1 IO

Arduino programming language has functions to communicate with the microcontroller boards digital, analog and built in communication ports. The table

below provides an overview of the functions available to read and write digital and analogue values.

Function	Syntax	Description
pinMode	pinMode(pin, mode)	Configures the specified pin to behave either as an input or an output.
digitalWrite	digitalWrite(pin, value)	Write a HIGH or a LOW value to a digital pin.
digitalRead	Byte digitalRead(pin)	Reads the value from a specified digital pin, either HIGH or LOW.
analogWrite	analogWrite(pin, value)	Writes an analog value (PWM wave) to a pin.
analogRead	Int analogRead(pin)	Reads the 10-bit value from the specified analog pin

External Interrupts pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. The function attachInterrupt() specifies the function to call when an external interrupt occurs. This replaces any previous function that was attached to the interrupt. Arduino Duemilanove and Nano boards have two external interrupts: numbers 0 (on digital pin 2) and 1 (on digital pin 3). The Arduino Mega has an additional four: numbers 2 (pin 21), 3 (pin 20), 4 (pin 19), and 5 (pin 18). This attachInterrupt() function is used in the accelerometer fall and activity detection node.

In addition to the function described above there are libraries available to access the boards various communication ports. All boards support SPI, I^2C and Serial USART communication.

6.1.2.2 Power

The boards can be powered via the USB connection or with an external power supply. The power source is selected automatically.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. More than 12V may cause the voltage regulator to overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are:

- **5V.** The regulated power supply used to power the microcontroller and other components on the board.
- **3V3.** A 3.3 volt supply generated by the on-board FTDI chip. Maximum current draw is 50 mA.
- **GND.** Ground pins.

6.1.2.3 Memory

	ATmega 168	Atmega 328	Atmega 1280
Falsh Memory	16 KB*	32 KB*	128 KB**
SRAM	1KB	2KB	8KB
EERPOM	512 bytes	1KB	4KB

*2KB , **4KB used for the boot loader

The EEPROM (non-volatile memory) can be accessed by two very convenient functions provided by the Arduino EEPORM library. The functions are read(address) and write(address, value). Read takes an address to read from and returns the value stored in that location. Write takes the address to write to and the value to write.

6.1.3 *Arduino Shields*

The Arduino can easily be extended. Several extension boards are available for the Arduino platform; these boards are referred to as shields.

Shields are boards that can be plugged on top of the Arduino PCB extending its capabilities. The different shields follow the same philosophy as the Arduino boards: they are easy to mount, and cheap to produce.

The Xbee transceiver is mounted onto a Xbee Shield seen in Figure 32. I also use Lithium battery shield to power the nodes. Several other shields are available these include a Wi-Fi, GPS, GPRS, Motor Control and Ethernet shield among many more. Shields are usually accompanied by Arduino programming libraries that provide simple interface to shields functionality.

6.2 Xbee

Xbee RF modules are embedded solutions providing wireless end-point connectivity to devices. Modules can use the ZigBee or IEEE 802.15.4 networking

protocol for point-to-multipoint, peer-to-peer and mesh networking. When using the ZigBee protocol modules can be configured to act as a coordinator, routers and end devices. The modules consist of a microcontroller and an RF antenna (Digi International, 2009).

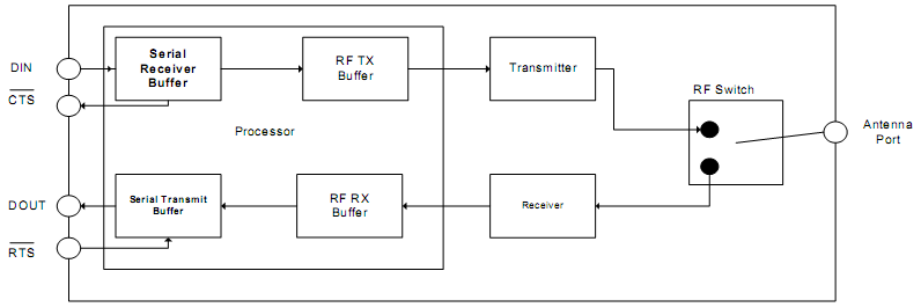

Figure 31 Xbee Module (Digi International, 2009)

Both coordinators and routers/end devices support the following two types of firmware Transparent Mode and API Mode.

In transparent mode, the Xbee works just like a serial line. Data input is buffered until a certain threshold is reached. It is then packetized and transmitted over RF. Serial AT commands are used to configure the internal Xbee parameters.

- Simple
- Compatible with any serial device that
- Limited to point to point communication between two XBee's

This mode will not be very useful for this project.

API mode allows:

- Xbee I/O Line reading
- Receive acknowledgement that a packet was successfully delivered
- Receive Source address of incoming packets
- Allows for Broadcast communication and communication with more than one Xbee.
- Obtain RSSI
- Remote Configuration
- ZigBee endpoints, cluster IDs and profile IDs

This mode is frame based. It allows the application running on the device to change Xbee parameters and create data payloads using frames. It also provides frames to the application in response to events (PAN association, reset, etc). This is the mode

all of our devices should be in. All API methods can be access through a community developed library called Xbee-Arduino (Rapp).

The Xbee transceiver is mounted on the Arduino board thought the use of the Xbee shield. Communication to the shield is done over Serial Arduino commands for transparent mode and using the Xbee Api (Rapp) in API mode. Code interfacing the Xbee Shield can be found in the CD appendix.

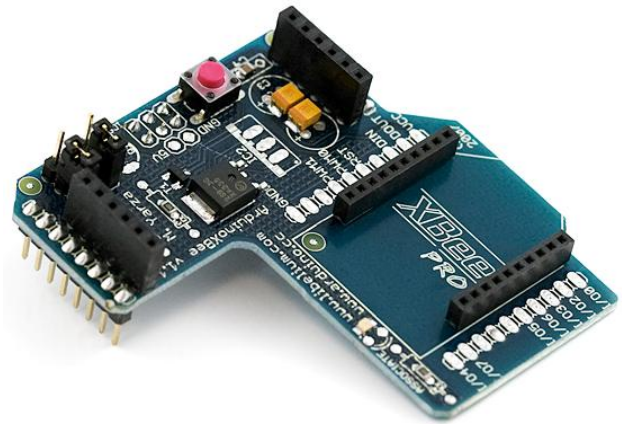

Figure 32 Xbee Shield

6.2.1 *Xbee Series 1 & 2*

XBee's are available in two different product lines. The 804.15.4 (Series 1) and the ZNet/ZB Pro (Series 2), each series comes in a standard or higher power (Pro) version.

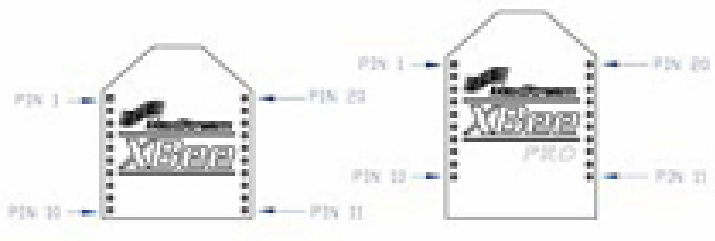

Figure 33 Xbee and Xbee Pro package (Digi International, 2009)

Below is a brief description of each series along with a cost a comparison of their relative advantages and disadvantages.

Series 1:

Series 1 is based on the IEEE 802.15.4 standard. It supports point-to-point and point-to-multipoint broadcast communication. This radio provides 6 analog and 8 digital I/O pins, with 6 pins shared between digital and analog. (Rapp, 2009)

Advantages over Series 2:

- Simpler
- Does not require a firmware change to switch between Coordinator and End Device, or AT mode
- I/O Line Monitoring supports > 1 sample per packet.
- Supports a larger payload size per packet than series 2 (100 bytes vs. 72 or 84)

Disadvantages over Series 2:

- No mesh networking

Series 2:

Series 2 is ZigBee compliant. ZigBee introduces mesh networking, where networks can be extended beyond typical wireless range limits through the use of routers. This radio supports 4 analog and 11 digital I/O pins, with 4 pins shared between digital and analog. Digi has recently released the ZigBee Pro (ZB) firmware and is the recommended firmware for this device. (Rapp, 2009)

Advantages over Series 1:

- ZigBee mesh networking
- Non-pro version has better range than series 1 equivalent (120m vs. 90m)
- Both pro and non-pro versions consume less power than their series 1 equivalent.
- Multi hop support. Extend the range of the network beyond the limit of a single transmission by adding routers.
- Interoperability with ZigBee hardware from other vendors that support the ZigBee PRO Feature Set

Disadvantages over Series 1:

- Analog inputs can only measure voltages between 0 and 1.2, requiring a voltage divider to step down from the reference voltage of 3.3V.
- ZB Pro API End Device firmware (2941) does not support the "No Sleep" option (SM=0). The default sleep option is "Cyclic Sleep" (SM=4).
- Requires separate firmware for API and transparent (AT) mode. It is not possible to switch between API and transparent mode with the same firmware instead a net firmware needs to be installed. The firmware is specific for Coordinator, and Router/End Devices.
- Only eight children are allowed for each single device router/coordinator.

7. Gateway Architecture

This section attempts to describe the Gateway architecture. The Gateway consists of 3 layers, which are shown in Figure 34 Gateway layered architectureFigure 34. Each layer can run independently and be configured in different environments.

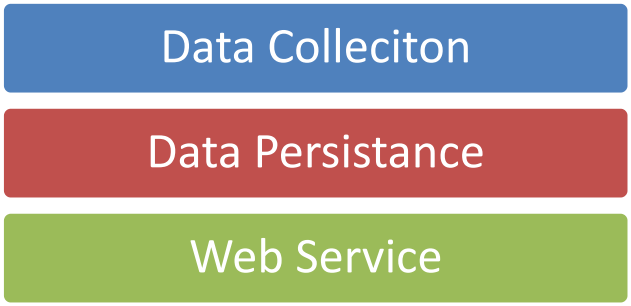

Figure 34 Gateway layered architecture

The data collection layer is responsible of receiving data from the network, providing real time data visualization and sending the data to the Data persistence layer.

The data persistence layer is responsible for storing all data sent from the data collection layer and making it accessible to the next layer.

The web service layer is responsible for exposing the data in the data persistence layer though a web server.

During development stages the 3 layers where configured on a single computer. On a production environment the Web service would run on its own server exposed to the internet while the remaining layers could either be configured to run on the same or separate machines.

7.1 Real Time Data View

In order to collect the data from the network the real time data view needed to communicate with the ZigBee Xbee coordinator through API mode. In ordered to do so from C# I have ported the Xbee java API (Rapp) into C# the API provide functions for diagnostics, routing, asynchronous and synchronous messaging and device configuration.

Java code was ported to C# using the Microsoft Java Language Conversion Assistant (JLCA). Code that calls Java APIs can convert to comparable C#

code that uses the .Net Framework. The JLCA converts 90 percent of JDK-level 1.1.4 calls and emits issues in code for the other 10 percent. Each issue is linked to a topic with guidelines for modifications needed to finish the conversion. Though converting to C# was very time consuming having API access to Xbee coordinator can be of great use in future version of the platform where network diagnoses is included as a part of the Gateway.

The application was developed on Windows Presentation Foundation (WPF) . The charts are created using VisiFire (www.visifire.com), a WPF based open source charting library.

The system provides real time data visualization for every sensor reading form the network. The application consists of 3 main components, the chart, the sensor panel and settings.

The sensor panel displays the meta-data for each sensor, along with the last data reading. A refresh can be used to update the panel in case a new node has joined the network, in such case rows for each sensors attached to that node will be displayed in the panel. Selecting a row in the channel activates the chart.

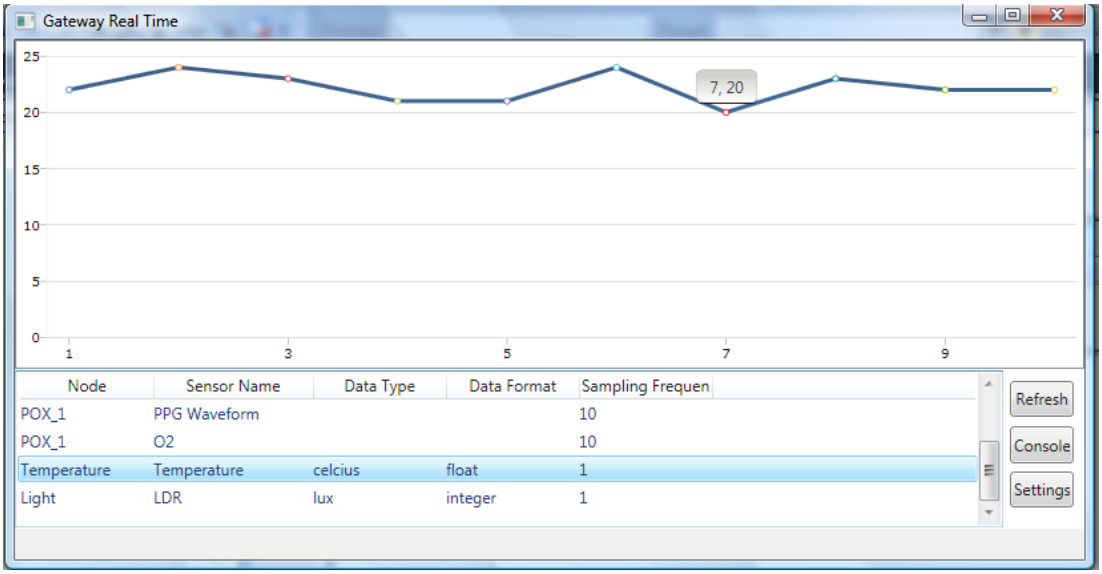

Figure 35 Temperature Readings

The chart is loaded with data from the sensor selected in the sensor panel; the chart automatically adjusts the Y axis so that only the relevant range for the sensor is displayed. The figure below shows an ECG wave form sampled at 50Hz.

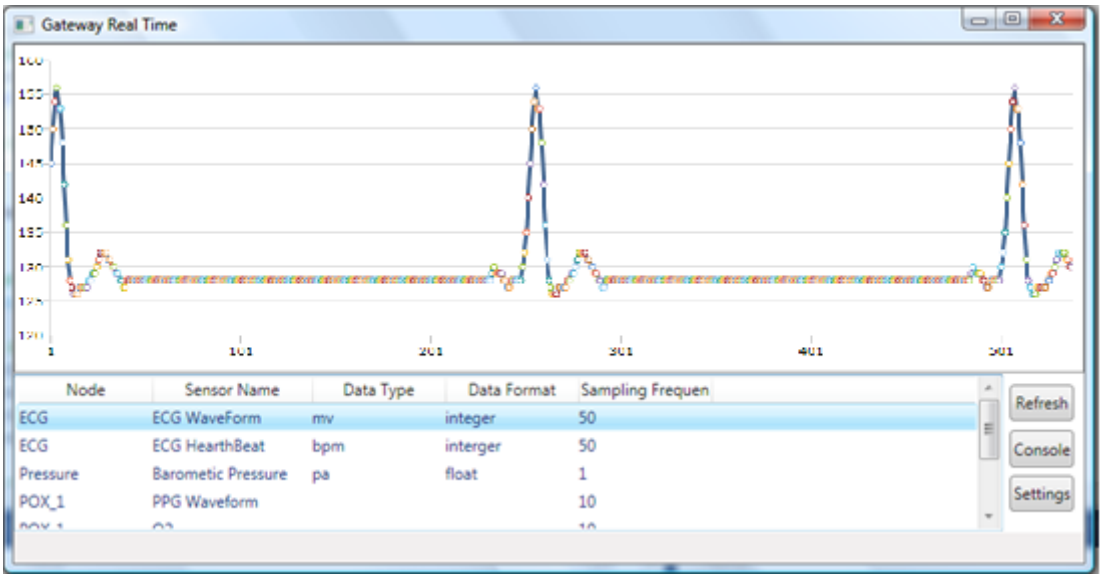

Figure 36 ECG Waveform @50HZ

Finally a configuration panel is available. The view below is the initial view of the application. Before data can be displayed, one must select the Com Port and Baud Rate of the Xbee coordinator (attached to the gateway) and the database connection string.

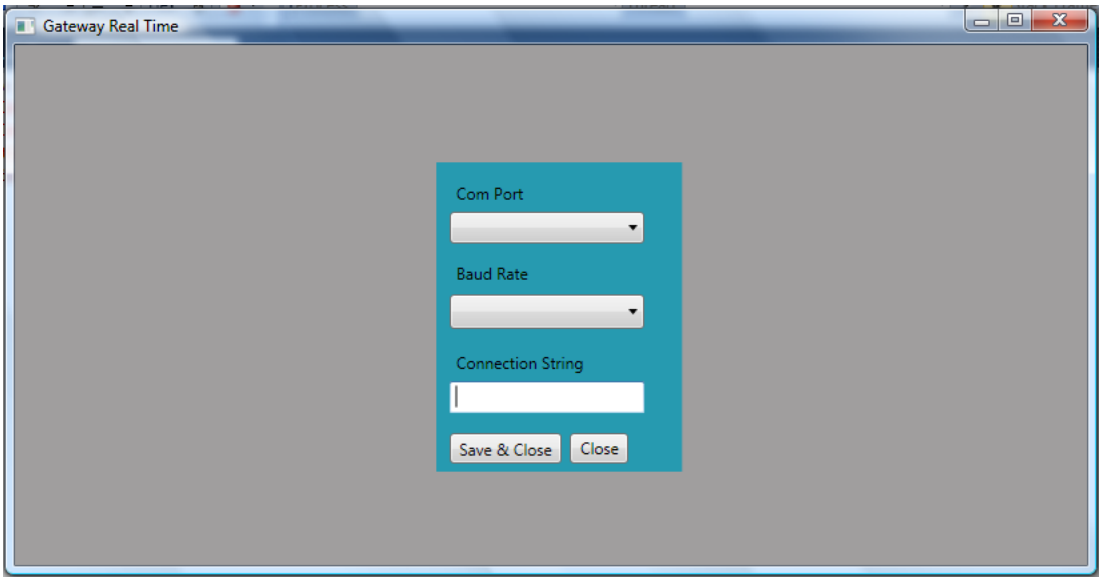

Figure 37 Settings Window

The user can open a window where he can view data entered in real time by pressing the console button.

The figure below shows the system diagram of the application. It provides an overview of the applications functionality and its main classes. The SerialComPort

class provides the XbeeAPI with a continuous stream of byte data. The XbeeAPI provides data to the VisiFire Charts, the Sensor Panel and Database writer classes. The settings class provides configuration information to the SerialComPort and Database writer class. And finally the SensorPanel class provides the VisiFire Charts class with information about the designed sensors reading to display.

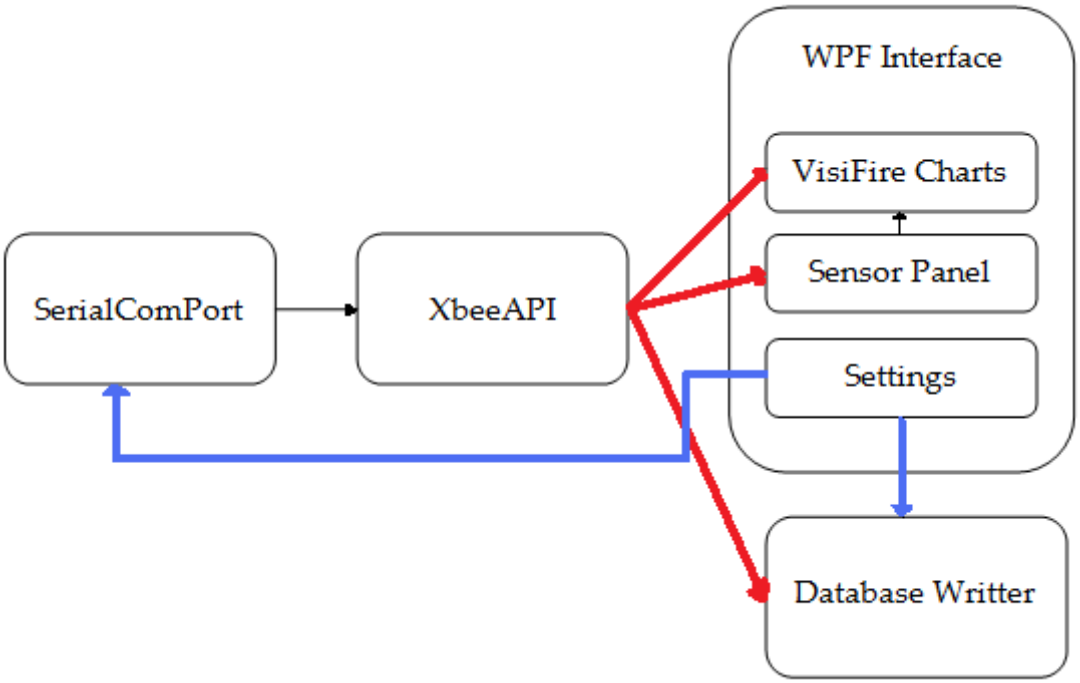

Figure 38 Real Time Application Gateway Class Overview

7.2 Database Architecture

As previously mentioned the database is a MS SQL database created in Microsoft SQL Server Express 2008. Since the Gateway was developed using Microsoft technologies there were certain development benefits in choosing the company's database technology, however there is nothing preventing a future user from migrating the database to other database solutions such as MySQL.

The data consist of 3 tables: a nodes, sensor and reading table. The relationship between those tables is shown below.

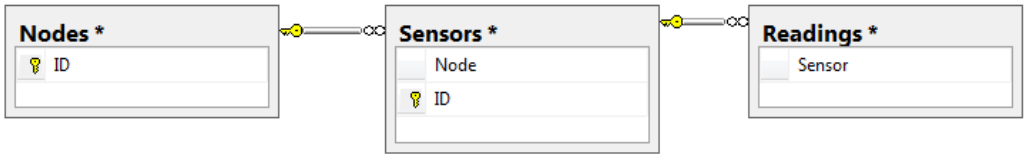

Figure 39 Database Tables Relations

The relations are simple yet abstract enough to hold all necessary information about the network.

In fact the structure of the tables is determined by the data transmission protocol described in section 5.1. The database is simply meant to store the values sent using the protocol. An extension to the protocol would likely require some modification to the database. The contents of each table are shown below. Note that a reading is given a timestamp. Time stamping is performed on every reading as soon as the data reaches the gateway. The time stamp is of format YY:MM:DD HH:MM:SS Millisecond.

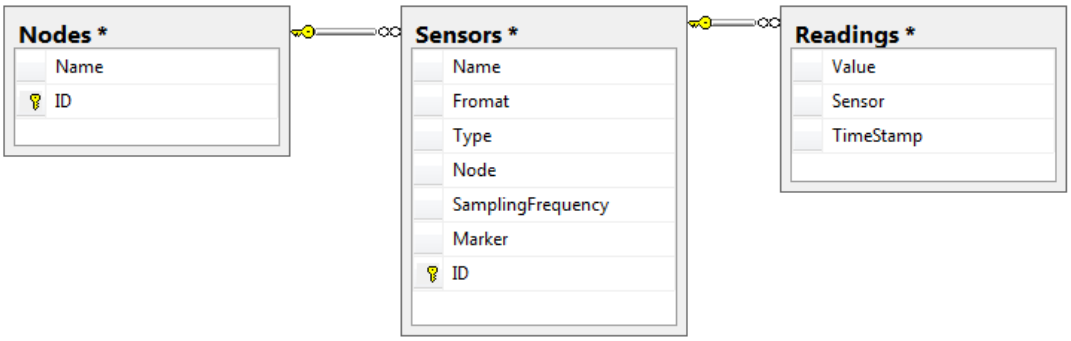

Figure 40 Database Tables

In favor of simplicity, the database stores all data entries in the Readings table as strings. This provides flexibility for supporting various data types. Parsing from string to the desired data type can be performed in by the data client.

This approach follows the design principle of keeping complexity on the edge. The main advantages are that an edge (data client in this case) is accessible and can be easily managed. While a database is central to the system, it can be difficult to access and has many dependencies. These dependencies will require modification, if the data storage structure changes.

The node and sensor tables are rarely updated while the sensor reading table is constantly being updated. This structure of the tables is meant to optimize write

performance for sensor reading. Similar considerations needed to be made in order to allow for fast reads. Reads occur when data clients request data through the web service or other application querying data directly. They are optimized through the use of multiple indexes for the tables and non locking mechanisms. These schemes help in keeping read time fast. A detailed description of the databases configuration is outside the scope of this section, further analyses can be performed by the interested reader by inspection of the source contained in the attached CD.

7.3 Web Server

In this section I describe the web services implemented in this project. The web server was developed in Microsoft ASP.NET platform and it hosted by any IIS (Internet Information Services) Server.

7.3.1 *JSON*

I use JavaScript Object Notation (JSON) to expose a Representational State Transfer (REST) web service that data clients can use to access the wireless networks data.

JSON is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. REST is a key design idiom that embraces a stateless client-server architecture in which the web services are viewed as resources and can be identified by their URLs. (Crockford, 2008)

The accessibility of rest web services makes it is ideal for such application where non experienced programmers will access the service.

The main reason for of using a JSON web service is that the web server returns only data, and not how it's formatted. This method of developing web applications is referred to as Service Oriented Architecture and emphasizes the principles of low coupling and high cohesion.

7.3.2 *Web Services*

The system defines 3 web services, one for each database table. The purpose of the web service is the expose the data for every table in the database, it is not meant to perform custom queries. This can however be done through any SQL server database client. The web services are Nodes, Sensor and Readings.

7.3.2.1 Nodes

This method is accessible through http://hostname/WSN/Nodes, it returns a list of all nodes stored in the network. It does not require any parameters since the database current implementation is meant to store data for a single wireless sensor network. This method queries both the node and sensor tables.

The service returns an array of Node data. The node is described by its unique identifier, its name and the list of sensor attached to it. The output of the web service for a network with a Motion Analysis and ECG node is shown below.

```
{ "Nodes": [
   "Name" : "ECGNode",
   "ID": 2,
   "Sensors": [
                {"Name": "ECG_Waveform", "ID" : 3},
                {"Name": "HeartBeat", "ID" : 5},
         ]
   ] , [
   "Name" : "MotionAnalysisNode",
   "ID": 2,
   "Sensors": [
             {"Name": "Activity", "ID" : 9},
             {"Name": "FreeFall", "ID" : 2}
         ]
   ]
}
```

7.3.2.2 Sensor

This method is accessible through http://hostname/WSN/Sensor?ID='ID_Value'. The method takes a single parameter: Sensor ID and returns all meta-data stored for the sensor. This web service call would query the sensor info table only. The output of the web service for ECG sensor is shown below.

```
{ "Sensor": {
         "Name" : "ECG_Waveform",
   "ID": 3,
         "Format" : 1,
         "Type" : "mV",
```

```
        "SamplingFrequency" : 50Hz
}
```

7.3.2.3 Readings

This method is accessible through http://hostname/WSN/Reading?SensorID='ID_Value'. The method takes a single parameter: Sensor ID and returns a list of all readings in reverse chronological order. This method queries the Readings data table, which allows high speed reads since no computationally expensive joins are executed and the database already maintains index for the table. 1 second of data from the ECG sensor is shown below.

```
{ "Readings": {
   "ID": 2,
   "Value": [145,150,154,156,153,153,148,142,136,131,128,127,126,
             126,126,127,127,127,127,127,128,128,129,129,130,131,
             132,132,132,132,131,131,130,130,129,129,128,128,127,
              127,128,128,128,128,128,128,128,128,128,128,128,128,
             ]
   }
}
```

The response consists of a simple list of value. Timestamp could be included along with the value; such extension could be easily added since the data is already available in the readings table.

7.4 Data Clients

Web browsers are the native clients for the web service. Entering the URL for the web service on a web browser will return the data in the format discussed in the previous section. The web browser displays the data in text format by default but, a simple web application could be added on the client to display the data in more intuitive forms such as table or possibly graphs. Currently a web client will only display the data returned in text format since no web browser client code has been implemented.

In order to allow users to perform sophisticated analysis on the data I have developed a Matlab client. The client consists of a JSON to data structure parser and 3 simple web service wrappers, the provided functions are:

NodesStruct = WSNNodes(WebServiceURL);

SensorStruct = WSNSensor(WebServiceURL,SensorID);

DataStruct = WSNReadings(WebServiceURL,SensorID);

The JSON parser converts JSON arrays into cell arrays and JSON objects into structures. Sample usage can be seen below:

MatlabStruct = parse_json(urlread(WebSErviceURL));

The figure below shows a sample ECG trace displayed in Matlab.

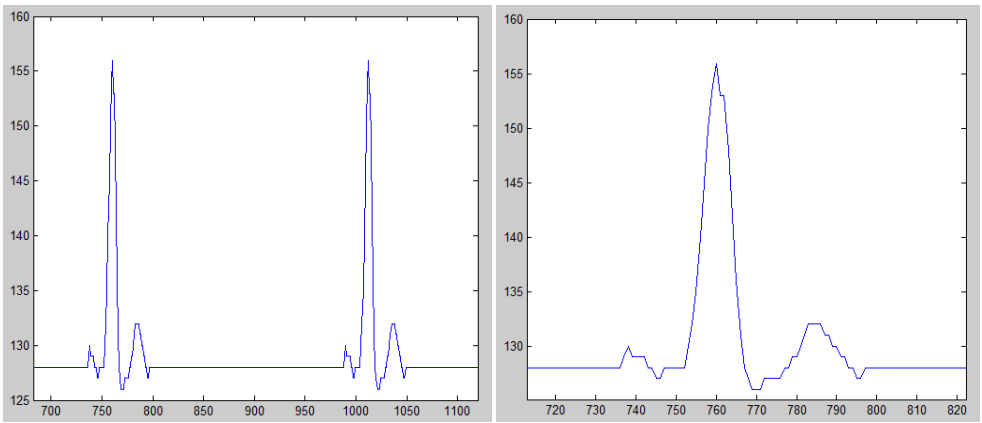

Figure 41 ECG Waveform in Matlab

8. Evaluation

This section describes the test conducted in order to evaluate the system.

The systems mayor components needed to be tested. Testing each of the components is a very complicated task, and the number of tests that can be performed on each aspect of a component are non exhaustive. The following Wikipedia article "List of system quality attributes" provides a comprehensive list of the viewpoints form which a system can be tested. These attributes are referred to in the software testing industry as the "ilities" e.g. Reliability, Accessibility, Scalability, Usability, etc...

Since a system cannot be comprehensively tested without evaluation criteria, it has been important for me to define it. The evaluation criteria I chose are in essence specified by the requirements that are listed in section 2.2. and consist of both functional and non functional requirements.

8.1 Tests

In this section I will take the requirements listed in section 2.2 and verify whether they were successfully met. The sub sections below contain the verification of each individual requirement.

8.1.1 *Overall System*

Requirement	Description	Outcome
Data provided by the network must be visualized in real time	This is achieved by the data visualization tool	✓
All data provided by the network must be persisted	This is achieved through the Database.	✓
Persisted data must be made accessible	This is achieved by the Web services	✓
The platform must be extensible	This was achieved by selecting standard based and open source hardware and software.	✓
The platform must be accessible to novice programmers	The used of Arduino and simple communication interface and gateway makes the platform, ideally suited for novice users.	✓
The platform must be based on standards	ZigBee and JSON and Arduino central technologies to the solution.	✓
The platform must be open source	Not every single component is open source. E.G MS SQL Server, .NET, Xbee Modules, and health monitoring sensors.	✗

The platform must be able to scale to deal with concurrent users	This was achived by demonstration fo the test shown in section 8.1.7 and through web service stress tests.	✓
The platform must cope with high and low bandwidth data transmissions	This is demonstrated in section 8.1.7	✓

8.1.2 *Gateway and real time data visualization tool*

Requirement	Description	Outcome
Show charts of data as they are received.	These can be performed using the application shown in Figure 36 ECG Waveform @50HZ	✓
Select which sensors data to view		✓
Set database connection	This can be performed as seen on Figure 37 Settings Window	✓
Receive network data from the network coordinator.	Connection settings with the coordinator are shown on Figure 37 Settings Window and data received is shown on Figure 35 Temperature Readings	✓
Determine data reading source	Data reading source is determined by the using the Node Identification Indicator in the Xbee api. The outcome of this can be seen by the ability to view a data for a data reading.	✓
Write parsed initialization frame to database	The initialization frame is written to the database can be seen in sections 7.3.2.1 and 7.3.2.2.	✓
Write the data readings to a database.	The data reading are written to the database, as seen in the sample database stored in the CD. Alternately one can see the output of the web service that reads sensor readings from the database in section 7.3.2.3	✓

8.1.3 *Data Client*

Requirement	Description	Outcome
Be able to query for nodes in the network	This can be done using the WSNNodes function shown in section 7.4 Data Clients	✓
Be able to query for sensor attached to a node	This can be done using the WSNSensor function shown in section 7.4 Data Clients	✓
Be able to query for data from a specific sensor	This can be done using the WSNReadings function shown in section 7.4 Data Clients	✓
Be able to graph and perform data analysis	A Matlab plot showing a ECG signal obtained from WSNReadings() is shown in section 7.4 Data Clients	✓

8.1.4 *Web Service*

Requirement	Description	Outcome
Expose sensor reading data through web service	The sensor readings web service and its output is	✓

	shown in section 7.3.2.3 Nodes	
Exposes node configuration and sensors connected to it through a web service	The sensor web service and its output is shown in section 7.3.2.2 Sensor	✓
Expose the nodes in the network though a web service	The nodes web service and its output is shown in section 7.3.2.1 Readings	✓

8.1.5 *Database*

Requirement	Description	Outcome
Store sensor readings	The readings are stored to the database, as seen in the sample database stored in the CD. Alternately one can see the output of the web service that reads sensor readings from the database in section 7.3.2.3	✓
Store node, sensor id and configuration data	The values are stored the database, as seen in the sample database stored in the CD. Alternately one can see the output of the web service that exposes these values in sections 7.3.2.1 and 7.3.2.2.	✓

8.1.6 *Wireless Sensor Network*

Requirement	Description	Outcome
Support tree, star and mesh topologies	ZigBee Xbee ZB modules support these topologies, and test where performed for each (described in following section)	✓
Must be able to secure data transmission	Xbee modules support encryption. This is done by setting EE AT command with a value of 1	✓
Must support acknowledgments	The initialization frame is sent using acknowledgments.	✓
Node must be able to transmit reading of String, Integer and Float data types	The data transmission protocol support string, integer and float types. (only signed integer and floats are currently supported)	✓
A node must be able to transmit various sensor values	Each node can transmit data for up to 6 sensor values as described in section 5.1.2.1	✓
Node must allow configuration of sensors and node settings	The initialization frame is used to specify sensor and node settings. This is described in section 5.1.1	✓
Nodes must be able to select sensor sampling intervals	A node can freely chose its sampling interval by the use of built in delay() function	✓
Nodes must be able to select transmission intervals	A node can buffer sampled values and transmit them at a specified interval as described in section 5.1.2.1	✓
Nodes must be able to sleep	Although Xbee transceivers can be configured to sleep, the Arduino boards where not configured to sleep in between transmissions. The chosen Arduino board have very inefficient voltage regulator, shutting down the microcontroller would decrease the board's power consumption by a mere 10%. Low power Arduino	✗

	boards are available and should be used if sleeping nodes are required.	
Nodes must be able to successfully transmit ECG and PPG (Plethogram) waveform sampled at least at 50 Hz over the network	As shown in the section below the network can cope with up to 3 devices transiting at 50 bytes per second, under certain topologies.	✓✗ inconclusive
Nodes must be able to transmit Barometric pressure, humidity, temperature, and light measurements.	Besides transmitting, light, humidity, temperature and barometric pressure, nodes can transit motion activity measure and perform free fall detection. These capabilities are described in section 0.	

8.1.7 *Indoor network load and stress test*

In order to determine the capabilities of the network, stress and load test where preformed. These tests attempt to simulate real life use of the network. The Xbee X-CTU tool was used to perform the tests, using the network diagnostic capabilities of Xbee transceiver presented in section 3.2.2.9 Diagnostics.

Before describing the tests and their results I need to specify the different data rate terms used and % Packet Loss success criteria. Data rates are defined by the overall data transmission rate of the network in bytes/second, the table below shows the definition criteria for Low, Medium and High data rates (in the context of these tests).

Name	*Network Transmission Rate criteria.*
Low	< 20 bytes/second
Medium	> 20 < 100 bytes/second
High	> 100 bytes/second

In order to determine whether a test was successful, a success cretin needs to be defined. Percentage packet loss is used to determine the success of the test, a percentage packet loss greater than 95% is considered a failure.

8.1.7.1 Topologies Test

The network was tested for percentage packet loss versus varying data rates under various topologies. The figure below provides an overview of the different topologies under which the test was preformed.

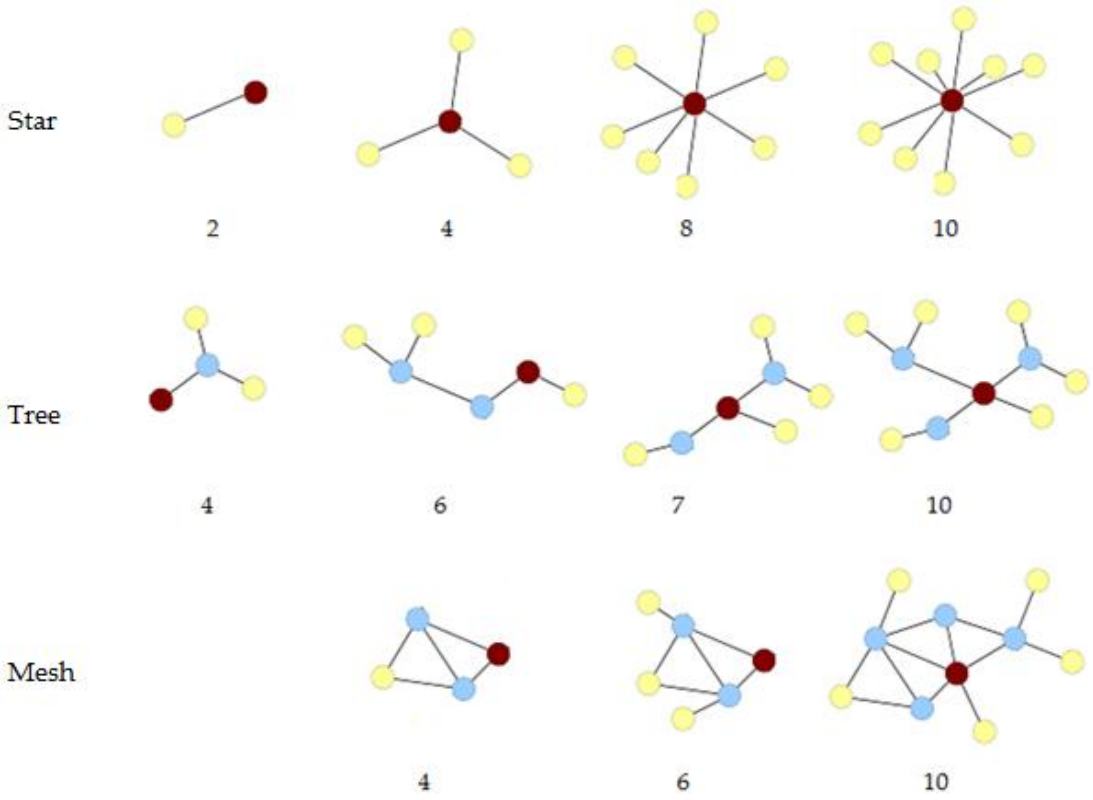

Figure 42 Tested Network Topologies

The table below shows the results of the test. Note that both mesh and tree protocol could not cope with medium and high data rates under certain configurations.

		Data Rates		
Topologies	# Nodes	Low	Medium	High
Star	2	✓ - %100	✓ - %100	✓ - %100
	4	✓ - %100	✓ - %100	✓ - %99.7
	8	✓ - %100	✓ - %100	✓ - %99.6
	10	✓ - %100	✓ - %99.5	✓ - %99.2
Tree	4	✓ - %100	✓ - %100	✓ - %100
	6 – hop	✓ - %100	✓ - %96.3	× - %89.5
	7	✓ - %100	✓ - %97.5	✓ - %95.9
	10	✓ - %98.6	× - %92.4	× - %97.5
Mesh	4	✓ - %100	✓ - %100	✓ - %100
	6	✓ - %100	✓- %96.9	× - %89.5
	10	✓ - %95.6	✓ - %95.1	× - %78.8

Tabel 12 Topologies vs. Data Rates

9. Conclusion

The main contributions of this project are the design an open source, standard based wireless sensor network platform that's accessible both technically and economically to undergraduate students. The platform is meant for health monitoring applications, but can also cope with broader usages. One goal of the report was to describe the system but most importantly the design and technological considerations that lead to the final solution. This will ensure that the next person who designs a similar system doesn't make the same mistakes and takes advantage of some of described solutions.

A performance evaluation of the platform obtained through analyses, simulation or measurements would be a logical progression of this project. Performance tests could be included such as: RSSI vs. Distance, Current Consumption vs. Data Transmission Rates, Security Enabled Throughput vs. Non Security Enabled, and many more.

The system has been tested on a 10 node wireless sensor network with successful results. All requirements specified have been successfully met, with the exception of three; being:

- Nodes must be able to sleep

  Although Xbee transceivers can be configured to sleep, the Arduino boards where not configured to sleep in between transmissions.

- The platform must be open source

  At the start of the project I set myself to create a fully open source platform. Although many of the components in the platform and open source some are proprietary. Because of ease of development the Gateway was built on property Microsoft technologies such as .NET and SQL server, and the wireless sensors network uses Xbee modules and sensor which are also proprietary. This is not a failure but instead is a reality that designers must face when creating a system, (even the Arduino IDE is built on Java, a propriety platform). Such compromises have to be made when designing complex system.

- Nodes must be able to transmit ECG and PPG (Plethogram) waveform sampled at least at 50 Hz.

The network can cope with two nodes transmitting at 50 Hz but it can only do so in peer to peer and simple star topology and mesh topologies. Mesh networking and multi hop routing introduces a lot of overhead on the network that has a significant effect on its overall data transmission capabilities. Truly scalable wireless sensor network application transmitting data at high rates are better suited to decentralized protocols where the entire networks data is not aggregated on a single node (coordinator).

The platform remains to undergo its most difficult test, namely a usability test. Although every part of the network was designed to be accessible and easy to learn, it has never been truly tested to see if it meets that requirement. An interesting study would be to present the system to a novice programmer and see how well he can understand the concepts and put them into practice. This could range from adding a new sensor, using the gateway real time visualization tool or even requesting data through the web service.

This is the first version of the platform and there are many directions in which it could be improved.

Sensor nodes are currently not optimized for power consumption. Creating low powers sensor nodes based on the Arduino platform would be a very interesting project in itself.

Other interesting focus areas could be expanding the number of health monitoring nodes. Adding blood pressure and blood sugar sensors could be included in the network. An interesting scenario would be to attach a memory card to a node and create Holter device that records values during an entry day and downloads then when in range of the network.

The data transmission protocol could benefit from the addition of a flag mechanism. Flag could be used to indicate whether a device is running low on power or to transmit internal error codes.

Networks diagnostic could be added as a building component of the Gateway. This could be very powerful in helping users of the platform setup up a good network with low packets loss high RSSI and redundant routes.

Bibliography

Analog Devices. 2009. *Datasheet Digital Accelerometer ADXL345.* 06 2009.

BCI Inc. 2002. *Technical Description for Micro Power Oximeter Board 31392B1.* 2002.

Beltrame, Marco. 2008. *Sviluppo di una rete wireless di sensori per il monitoraggio di strutture in tempo reale.* s.l. : Pavia University, 2008.

Crockford, Douglas. 2008. *JavaScript: The Good Parts .* s.l. : O'Reilly, 2008.

Daintree. 2008. *Getting Started with ZigBee and IEEE 802.15.4.* 2008.

Derek Densmore, Joe Santarcangelo. 2006. Pulse Oximetry Theory. *Arterial Pulse Meter.* [Online] 2006. http://www.ee.ryerson.ca/~jsantarc/html/introduction.html.

Digi International. 2009. *XBee®/XBee-PRO® ZB OEM RF Modules.* 03. 03 2009.

Guy, Some. Arduino Xbee API. [Online]

Hardvard University. *Sensor Networks for Medical Care.*

IBM. 2008. RESTful Web services. *IBM Technical Library.* [Online] 06. 11 2008.

IEEE. 2003. *IEEE 802.15.4 Standard Specification.* s.l. : IEEE, 2003.

Indirect health effects of relative humidity in indoor environments. **Arundel AV, Sterling EM, Biggin JH, Sterling TD. 1986 Mar.** 1986 Mar, Environ Health Perspect. , s. 65:351-61.

Iskiw, Mike. 2009. REST vs. SOAP . *Mike Iskiw .* [Online] 25. 03 2009. http://weblogs.asp.net/mikeiskiw/archive/2009/03/26/rest-vs-soap-feeling-restful.aspx.

Matt. 2009. Illuminato X Machina . *Liquidware Antipasto.* [Online] 19. 08 2009. http://antipastohw.blogspot.com/2009/08/introducing-illuminato-x-machina.html.

Medlab. 2008. *ECG OEM Module.* 14. 07 2008.

Munk-Stander, Jacob. 2005. *Implementing a ZigBee Protocol Stack.* 2005.

Oxer, Jonathan. 2009. Library for SHT15 temperature / humidity sensor. *Practical Arduino.* [Online] 16. June 2009. http://www.practicalarduino.com/news/id/211.

Paul H. Tanser, MD. 2006. Interpreting ECG. *Symptoms and Diagnosis of Heart and Blood Vessel Disorders .* [Online] April 2006. http://www.merck.com/mmhe/sec03/ch021/ch021c.html.

Pulse oximetry. **Jubran, Amal. 1998.** 1998, PubMed Central .

Rapp, Andrew. Arduino library for communicating with XBees in API mode. *xbee-arduino.* [Online] http://code.google.com/p/xbee-arduino/.

—. 2009. ChoosingAnXBee . [Online] 2009. http://code.google.com/p/xbee-api/wiki/ChoosingAnXBee.

—. xbee-api. *A Java API for Digi XBee/XBee-Pro OEM RF Modules.* [Online] http://code.google.com/p/xbee-api/.

Relation of atmospheric pressure changes and the occurrences of acute myocardial infarction and stroke. **Houck, Philip D. 2005.** 2005, American Journal of Cardiology, s. 96[1]:45-51.

Roberts, Michelle. 2006. Why more people die in the winter . *BBC Health.* [Online] 27. 10 2006. http://news.bbc.co.uk/2/hi/health/5372296.stm.

RS. 1997. *Data Sheet Light Dependant Resistor.* 1997.

Sensirion. 2002. *DataSheet - Humidity and Temperature Sentimeter.* 2002.

Sparkfun Electronics. 2009. *Sparkfun .* [Online] 2009. http://www.sparkfun.com.

Vergano, Dan. Weather Forecast May Call For Migraines. *Medical Tribune News Service.* [Online] http://www.personalmd.com/altmedicine_update1.shtml.

Villafañe, Juan. ECG (electrocardiogram). *My Kentucky Heart.* [Online] http://mykentuckyheart.com/services/ECG.htm.

VTI Technologies. *DatSheet SCP1000 Series Absolute pressure sensor .*

Wikipedia. 2009. IEEE 802.15.4a. *Wikipedia.* [Online] 01. 08 2009. http://en.wikipedia.org/wiki/IEEE_802.15.4a.

—. 2009. OSI Model. *Wikipedia.* [Online] 17. 08 2009. http://en.wikipedia.org/wiki/OSI_model.

—. 2009. SOAP. *Wikipedia.* [Online] 15. 07 2009. http://en.wikipedia.org/wiki/SOAP.

—. **2009.** ZigBee. *Wikipedia.* [Online] 29. 08 2009. http://en.wikipedia.org/wiki/ZigBee.

Appendix

Xbee configuration

Digi provides a Windows X-CTU configuration tool for configuring module parameters and updating firmware. The XCTU has the capability to do the following:

• Discover all Xbee devices in the network

• Update firmware on a local module (requires USB or serial connection)

• Read or write module configuration parameters on a local or remote device

• Save and load configuration profiles containing customized settings.

Xbee modules can be configured by means of AT commands. The Arduino can be in fact used for this purpose. To configure it from the Arduino board, the jumpers in the Xbee Shield need to be set in the Xbee position. To configure it from the computer, you'll need to have the jumpers in the USB configuration and have removed the microcontroller from the Arduino board. Alternately to configure it from the computer one can use an Xbee explorer module from spark fun.

To get the module into configuration mode, you need to send it three plus signs: +++ followed by a one second delay where no commands are sent to the module. If the modules under configuration mode it will reply 'OK' and carriage return.

Send Command	Expected Response
+++	OK<*CR*>

Once in configuration mode, you can send AT commands to the module. Command strings have the form ATxx (where xx is the name of a setting). To read the current value of the setting, send the command string followed by a carriage return. To write a new value to the setting, send the command string, immediately followed by the new setting value and a carriage return.

For example, to read the network ID (PAN ID) of the module you can use the ID command:

Send Command	Expected Response

ATID<*enter*>	3332<*CR*>

To change the network ID of the module:

Send Command	Expected Response
ATID3331<*enter*>	OK<*CR*>

Now, check that the setting has taken effect:

Send Command	Expected Response
ATID<*enter*>	3331<*CR*>

The changes in configuration shown above are written to volatile memory. If configuration is meant to be preserved during power loss you must write the settings to the EEPROM (non-volatile) memory. These can we done through the WR command:

Send Command	Expected Response
ATWR<*enter*>	OK<*CR*>

To reset the module to the factory settings, use the RE command:

Send Command	Expected Response
ATRE<*enter*>	OK<*CR*>

Configuration possibilities of Xbee modules are non-exhaustive. Every aspect of a wireless sensors network, presented in sections 3.2.2 ZigBee and Xbee can be configure through AT commands. The interested reader is encouraged to read the product datasheet (Digi International, 2009), which contains detailed descriptions of every command available.

CD Contents

The CD contains the source for the entire platform. The content and requirements to execute and develop the individual layers of the platform is described in the table below. The CD contains 5 root folders on corresponding to each physically separable layer.

The entire source as described in the table below has been uploaded to Google's open source hosting site and can be found at http://code.google.com/p/owsnp. I've posted the code online so that those interested could download and/or contribute to the project. The code is published under the MIT license.

Folder	Description
SensorNodes	This folder contains the source code for every single node in the network. The source code for the following types of nodes is included. • ECG Node • POX Node • Activity Monitoring Node • Temperature Monitoring Node • Humidity Node • Barometric Pressure Node • Panic Button Node • Light Sensor Node There is also source code for integrates nodes. The functionally of the nodes listed above can be combined in type of configuration. I include the source for a • POX + ECG Node • ECG + Activity Monitoring Node • Temperature + Humidity + Barometric Pressure Node In order to program the microcontrollers you need the Arduino IDE. This can be found online at http://arduino.cc
Gateway	This includes both the source code and executable file for data collection and real time visualization application.

	The application can run on any installation of windows operation system. Visual Studio Express 2008 development environment (or higher) is needed to run the source code. This tool is free and available for download at http://web.microsoft.com
DB_Scripts	Contains the scripts needed to generate the database and database tables. A sample database is included with sample data for ECG, POX , Humidity and Temperature sensor nodes. SQL Server Express 2008 (or greater) is required to be able to execute these scripts. SQL Server Express 2008 can be downloaded free of charge at http://web.microsoft.com
Server	Contains the code and executable file for the web services. The application can run on any installation of windows operation system. Visual Studio Express 2008 development environment (or higher) and ASP.NET MVC is needed to run the source code. These tools are free and available for download at http://web.microsoft.com.
Client	Contains the MATLAB function needed to access the web service. Only built in MATLAB function are used so any installation of MATLAB can be used.