Mobile Ontology-Based Reasoning & Feedback Health Monitoring System

By

Luke Docksteader

A Thesis Submitted to the Department of Electrical Engineering

in Partial Fulfillment of the Requirement for the Degree of

Master of Science

at

Lakehead University

Thunder Bay, Ontario, Canada

September 2009

ACKNOWLEDGEMENTS

- I would like to thank my supervisor, Dr. Rachid Benlamri for all his help and support. This research would not have been possible without his aid.

- I would also like to thank Dr. Arnold Kim and Vincent Ruberto for their suggestions, encouragement and help with understanding the medical side of this research.

- Finally, I would like to thank both my colleagues and friends who have been supporting me in this research from the beginning, whether it be through simple encouragement or prototype testing. Thank you!

Abstract

As medical technology has developed over the past few decades, the average human life-span has increased and as a side effect, chronic illness has become more prevalent. Many patients now require constant, everyday monitoring of their vitals to ensure their health conditions don't deteriorate. Such monitoring is often carried out in a hospital or hospital-like environment when it need not be. In more recent years, many research efforts have been made to develop robust health monitoring systems. Most of these systems are being used in hospitals to monitor patients' vitals in order to provide timely treatment. Introducing a ubiquitous system to remotely monitor patients outside of the hospital provides huge financial savings to the health care system, as non-critical patients who have partial autonomy and mobility can then be discharged from the hospital. This research describes a mobile ontology-based health monitoring system that seamlessly integrates vital data from various health sensors in order to identify the health status of the patient and to take the appropriate actions for a timely support.

The proposed approach makes use of recent advances in semantic web, ubiquitous computing, and mobile communication technologies to build a Mobile Ontology-based Reasoning and Feedback (MORF) health monitoring system. MORF includes a wearable monitor that uses various sensors to record a patient's vitals. To make MORF mobile, the sensed data, once acquired, is routed through a mobile device such as a cell phone or personal digital assistant (PDA) before being transmitted to the local monitoring station for analysis. This analysis is designed to embed a sense of context awareness into the system which enables it to understand its environmental circumstances and act based on that understanding. Using ontology-based reasoning, the server at the monitoring station manages properly the system's overall context and decides what actions to perform if the patient's vitals exceed their prescribed norms. The goal of MORF is to expand above, improve upon and advance beyond the many systems that currently exist in this field. Such advancements include the MORF's feedback capabilities which allow for the controlling of the frequency at which the sensed data is acquired. Based on the previously sensed data and the patient's current condition, the reasoning engine will tune the sensory data acquisition rate of the various sensors to provide the most effective monitoring possible. These are among the main contributions of the proposed MORF system whose aims are delivering context-aware intelligent autonomous services for healthcare monitoring.

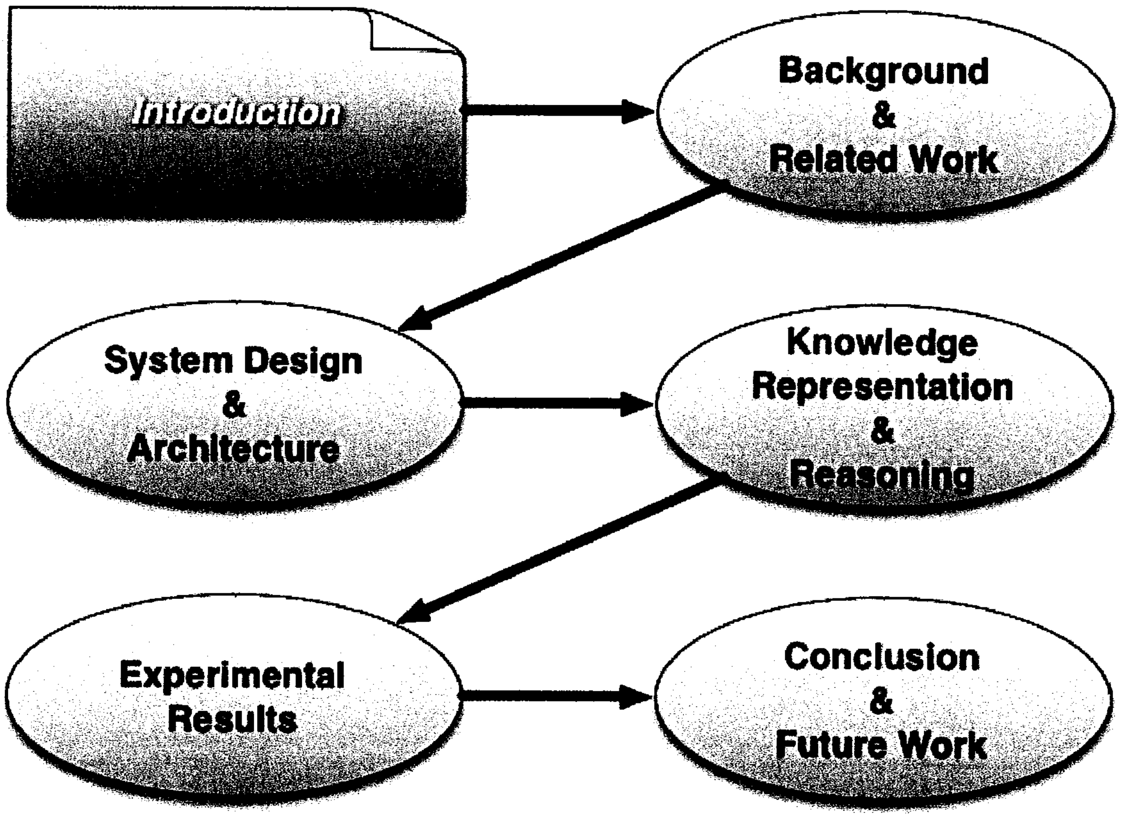
Keywords: Healthcare, Telemedicine, Mobile Computing, Semantic Web, Ontology, Ontology-based Reasoning, Ubiquitous Computing, Web Services, Client-Server, Knowledge Management, Context Modelling.

Table of Contents

List of Figures

LIST OF TABLES

chapter 1


Introduction
Background & Related Work
System Design & Architecture
Knowledge Representation & Reasoning
Experimental Results
Conclusion & Future Work

introduction

1.1 – Motivation & Objectives

This research pertains to the design and implementation of a mobile health monitoring system geared towards non-critical hospital patients that still require some degree of monitoring. Modelling of medical knowledge, context acquisition, management and analysis of patient sensor data and an adaptive monitoring environment based on sensory feedback are all key elements in the design of a system to meet the needs of such a monitoring platform.

The goal of this research is to design a system that meets the above criteria. This system must be secure and reliable as patient confidentiality is crucial. With a system such as MORF on the market, the number of overcrowded hospitals will decrease. Doctor-to-patient ratios within the hospital will decrease as well as there will be fewer patients required to be present in the hospital for monitoring purposes.

As there are other systems in this field of research there were a few problems that needed to be overcome to make the MORF health monitoring system a more viable option. These problems include the following:

- Existing mobile health monitoring systems often require trained medical personnel to continuously monitor the incoming sensor data from a patient.
- Lack of contextual processing and reasoning of the incoming sensor data.
- Limited mobility and lack of viable methods of monitoring a patient effectively

from a distance.

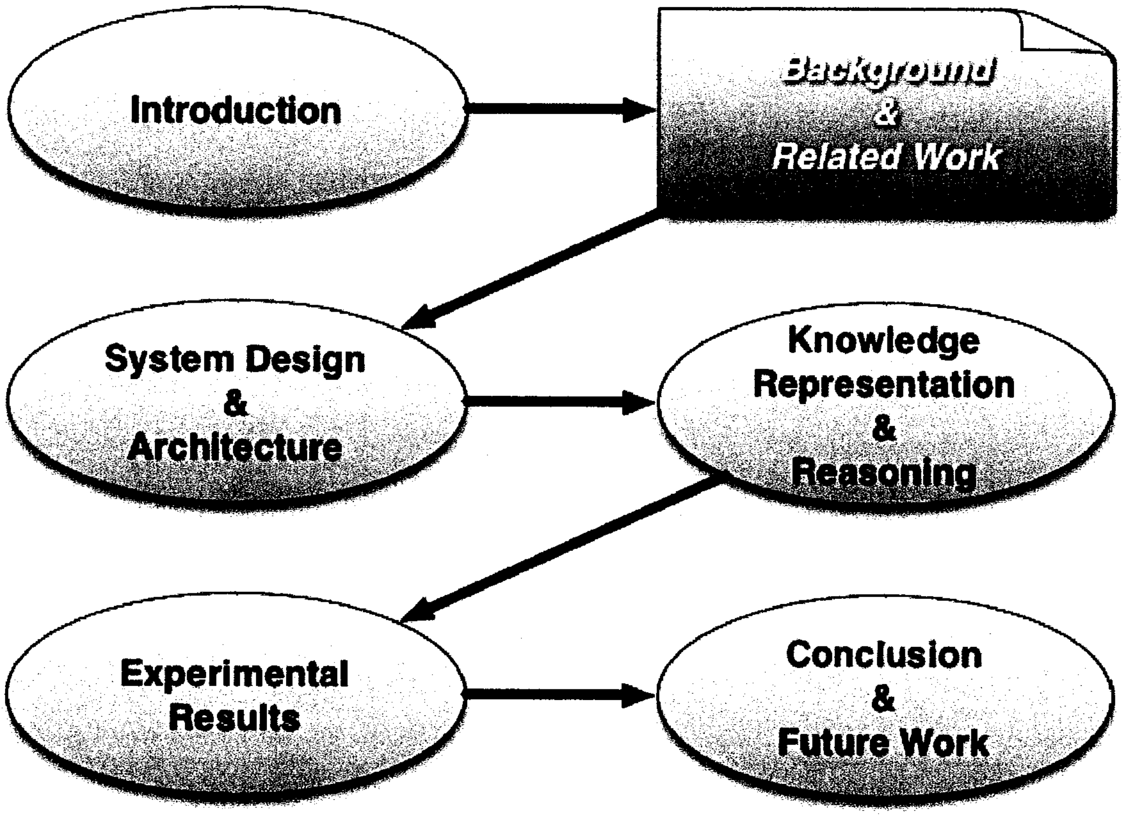
The MORF system overcomes these limitations by utilizing a platform that is truly mobile and, on the server end, processes all of the sensor data without the need of human monitoring. The importance of this research is that it provides a mechanism for which hospitals can safely discharge non-critical patients while still keeping a close eye on their health. As mentioned earlier, this not only frees up beds in hospitals but is one more step to help relieve overworked nurses and doctors within the hospital environment.

The major contribution that the MORF health monitoring system brings to the table in this field of research is that of combining mobility, ontology-based knowledge management and rule-based reasoning. This combination, geared towards the medical field, creates an endless number of possible applications and creates innumerable avenues on which continued research may be performed.

1.2 – Thesis Structure

Chapter 2 introduces a basic background understanding of some of the key concepts involved in our research. It also provides insight into past research performed in the field of medical monitoring systems, specifically mobile systems. Chapter 3 describes the overall hardware design and the functional operation of the system. Chapter 3 also describes in detail some of the technologies used in they system as well as why they were used. In chapter 3, we also describe, in detail, how the sensor data is acquired, the operational environment of the system, infrastructure for data storage and management and a simple fictional scenario to help better understand the general operation of the system. Chapter 4 covers both and in-depth theoretical understanding as well as a functional understanding of the knowledge management, contextual reasoning and alarm management portions of our system. Chapter 5 walks through the different user interfaces used in the MORF system and covers, in detail, some experimental case studies to show how the system operates. Finally, in chapter 6 we will conclude the results of this research and expound upon any further research or future work that could be done to improve the MORF health monitoring system.

chapter 2


Introduction
Background & Related Work
System Design & Architecture
Knowledge Representation & Reasoning
Experimental Results
Conclusion & Future Work

background & related work

In this chapter we will outline some general background information required to have a good understanding of how the MORF system functions. In doing so, we will also compare some of the basic technologies used in our system with other technologies on the market today. Finally, we will perform an analysis of other related research in the field of mobile health monitoring platforms, compare each of these systems to our own and explain how our system has been designed to expand beyond the boundaries of these other systems.

2.1 – Medical Technologies

Over the past few years many medical professionals have shifted their focus to ubiquitous health monitoring systems. With continuous advancements in both bio-medical and information technologies, such futuristic monitoring systems have become a present reality. When intermixed with ubiquitous computing, a mobile health-monitoring system provides unrestrained potential and innumerable applications. The Mobile Ontology-based Reasoning and Feedback (MORF) system aims to improve upon and advance beyond modern medical monitoring systems by incorporating ontology-based reasoning to make the system independent and automated.

The only medical technologies used in the MORF system are the medical monitoring sensors. Since we are not designing the medical monitors themselves from scratch we must look to see what sort of devices are currently on the market. As there are a plethora of medical sensor on the market today, it is important that the sensors we choose be robust and accurate. The only other criteria for the medical sensors, as it pertains to our system, is that they be specifically designed for mobility. Sensors that incorporate the use of short-range wireless communication technologies, such as Bluetooth, are ideal and will integrate with ease into the MORF health monitoring system.

2.2 – Web 2.0 vs. Semantic Web

The term Web 2.0 refers to the design and development of documents on the web, in a user centered manner, that increases the interactive sharing and collaboration of information on the Internet Error: Reference source not found. Web 2.0 isn't so much a technical upgrade or a new version of web technologies, but rather an overarching change in the manner in which the web is used by developers and end-users alike Error: Reference source not found. It is composed of documents based on technologies, or written in languages, such as Hyper Text Markup Language (HTML), JavaScript, Extensible Markup Language (XML), Ajax, Adobe Flash and more. One of the primary goals of Web 2.0 is that of presenting data to the user in a human-readable format.

On the other hand, the primary goal of the Semantic Web is that of making data and resources found on the web processable and usable by machines by giving them meaning. Giving a resource meaning is performed by establishing relationships between this resource and other resources which is expressed using Resource Description Framework (RDF). RDF encodes these relationships in sets of triples much like how a standard sentences is formed by a subject, predicate and object. The subject and object are described by Universal Resource Identifiers (URIs) which link them to their appropriate resources. The predicate is also described by a URI and is used to form a relationship between the subject and object [2].

2.3 – Communications Technologies

For mobility, MORF utilizes a few key communication technologies to achieve it's goal of providing a mobile solution to monitoring a patient from outside of the hospital environment. To communicate with the various sensors connected to the system, we use the Bluetooth communication protocol. This short-range wireless technology is most widely used for device-to-device communications and was chosen above other such technologies as the ZigBee protocol because it is more widely used and is a standard amongst most mobile phones. A more detailed comparison of these two technologies can be found in section 3.3.1.

When it comes to mobile data communication technologies, we are using 3G technologies to send the sensor data retrieved from the mobile unit to our central monitoring server. 3G technologies, such as Global Systems for Mobile Communication (GSM) and High Speed Packet Access (HSPA) operate through the cell networks to enable a mobile device to connect to the Internet.

2.4 – Related Work

There are a number of other research projects that have developed monitoring systems for medical use. Many of these projects are simply hardware oriented systems that, for the most part, ignore context reasoning and ubiquitous applications in their research. Projects that fall under the aforementioned category include the Telzuit BioPatch Cardiac Monitoring System [3]. This system is specific to the use of ECG monitoring. It takes sensor data read from the ECG monitor and sends it to a local monitoring station where employees constantly monitor the data readout of each patient using the monitor. The patient wears an electrode array patch which transmits sensor data through a PDA or phone to the database center through the cell networks. In the database center, employees monitor the patient's incoming data and forward that data on to medical professionals should it be necessary.

The mSens Mobile Health Monitoring System is a system focused around an XML-based communications framework to allow for a self-configurable health monitoring system [4]. Similar to MORF, mSens uses Bluetooth to communicate between a patient's monitor and the monitor's control unit. Unlike MORF, mSens's control unit is stationary. It is, therefore, in the true sense of the meaning, not a fully mobile system. It is only mobile within the transmission range of the Bluetooth module from the control unit.

Another system designed by Loughborough University in the U.K. sends sensor data remotely to a doctor's phone where he can remotely monitor a patient's vitals [5]. The sensor data is collected and transmitted through a mobile device and then on through the mobile communications networks where it is received by the patient's doctor. The doctor can review the data in real-time.

Another system designed by researchers at the Seoul National University in Korea has focused on designing a smart bedroom that is used to monitor the ECG as well as the snoring habits and movements of its patients while sleeping [6]. Using Bluetooth and wireless LAN technologies, information gathered from sensors connected to the patient's bed is transmitted to a monitoring station outside of the room where the data is processed and analyzed.

One last system under this category includes a system being designed by Tsinhua University in China which monitors a patients vitals and sends the acquired data to a local monitoring station [7]. This system monitors vital signs such as ECG, SpO2, body temperature and respiration. Data is transmitted via mobile communications networks to a mobile monitoring station and then to the hospitals central management system where, again, the data must be reviewed and interpreted by a physician or other medical personnel.

Other, more advanced, research projects that are being developed which utilized both ubiquitous computing and context reasoning include a project developed by the University of Florence that aims at providing at-home monitoring of a patient and ontology-based reasoning performed by the patient's personal computer. Data is collected, analyzed and alarm notification is sent out to appropriate personnel if necessary. Allowing the data to be processed by the patient's personal computer puts great requirements on the patient prior to the use of this device. It also allows for the processing of the patients personal information on an unsecured device [8].

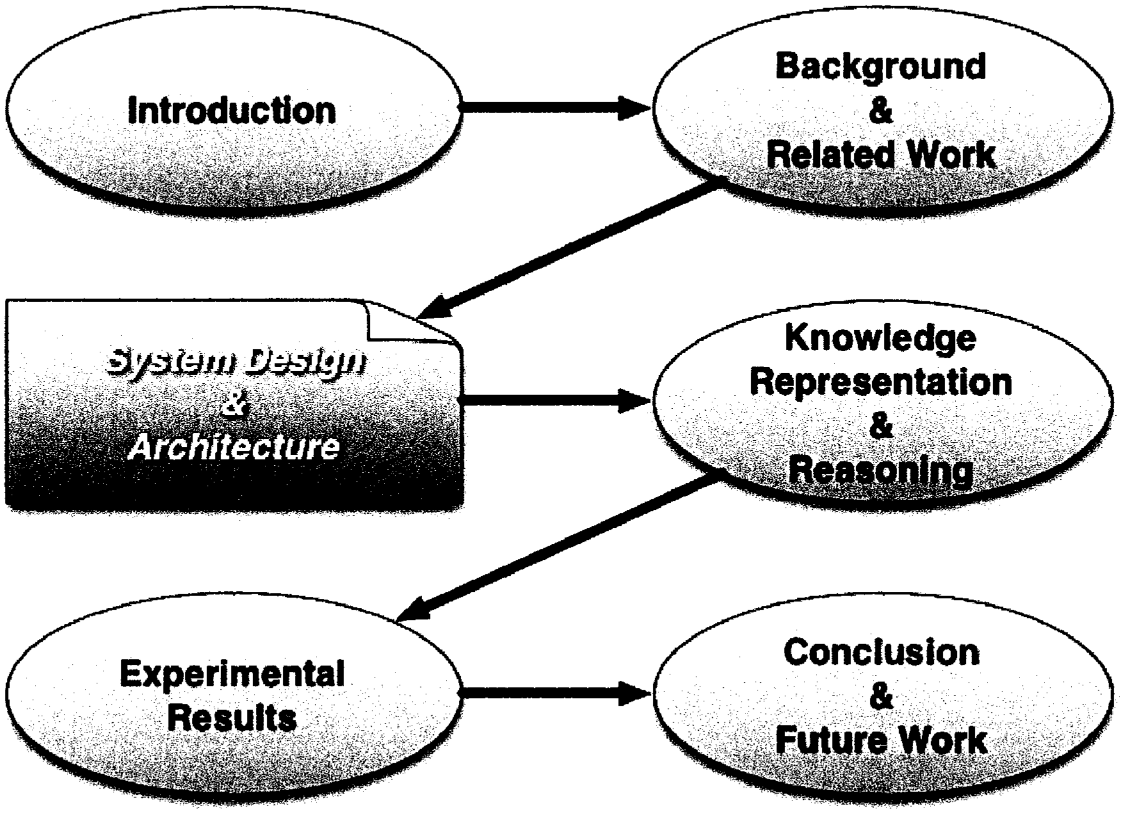
A recent work written by Lee and Jung from the Electronics and Telecommunications Research Institute of the Republic of Korea outlines the design of a ubiquitous monitoring system using the ZigBee protocol to wirelessly transmit patient sensor data so that it may be monitored at a local health station. A bio-signal sensor is worn by the patient which transmits sensor data to a ZigBee dongle connected to a mobile phone or PDA. This sensor data is then transmitted through the internet to a local monitoring station where it is processed and analyzed by a reasoning server [9].

An interesting concept by Ho, Shiao and Liu presented the idea of a ubiquitous hospital environment where patients within the hospital could be constantly monitored without the need of a doctor. Each patient wore a monitor which had a built-in RFID tag to allow the hospital server to monitor the patient's mobility and current location. The mobile device also used Bluetooth wireless communications to transmit the acquired sensor data to the hospital's servers [10].

Other projects that don't necessarily fall under the health domain but still utilize ubiquitous computing and contextual ontology-based reasoning include the Freeband Awareness project, Ubiquitous e-Learning with Multimodal Multimedia Devices and many more. Systems such as these have provided invaluable background knowledge required for the proper implementation of the MORF system [11]-[13].

The aim of the system described in this thesis is to seamlessly integrate vital data from various health sensors in order to identify the health status of the patient and to take the appropriate measures for a timely support. Combining ubiquitous computing, context reasoning and mobile communication technologies, this project will yield a system that is mobile, automated and easy to use. MORF utilizes a true mobile architecture based around a mobile sensor unit and a mobile phone. Combining these two devices allows for a patient to achieve true mobility, free to roam wherever a cell network exists, all the while keeping a close eye on the patient's status and ready to act should something go wrong. MORF implements sensor communication with a mobile device via the Bluetooth protocol. This data is received by a mobile phone and forwarded on to the central server which houses the reasoning engine. Ontology-based context modelling is encoded in the Web Ontology Language (OWL) via the Protégé environment and the reasoning engine encodes rules using the Semantic Web Rule Language (SWRL) to provide autonomicity to the system. Finally, MORF's feedback system control is driven by the sensed information and the reasoning engine which will enable greater control over the monitoring system when more frequent measurements or even more data is/are required.

chapter 3


Introduction
Background & Related Work
System Design & Architecture
Knowledge Representation & Reasoning
Experimental Results
Conclusion & Future Work

system design & architecture

In this chapter we will discuss the overall system design and architecture of the MORF health monitoring platform. This includes the functional operation of the system, the hardware design, sensory data acquisition, system mobility and the infrastructure for data storage and management. In addition to these, we will discuss a fictional case scenario to help form a basic understand of how the system has been designed to operate from a functional perspective.

3.1 – Functional Operation

In its most general sense, the term context is used to describe any information or knowledge that can be used to depict or describe the current situation [14]. In the case of MORF, electronic sensors that are attached to the patient's body are used to create a sense of context. This context specifically pertains to the patient's health status. The sensors gather their appropriate data and transmit that information to the patient's mobile device via Bluetooth communication protocol. This mobile device acts as a conduit between the patient and the monitoring server. Its job is to forward any incoming sensor data to the hospital server via the Internet. The mobile device is also designed to execute any commands sent to it from the main server. This allows the hospital to have a greater control on how the patient is monitored, and how frequently measurements are taken. This creates a sense of feedback in the system. Once the data is uploaded to the server, it is stored and analyzed. This analysis is performed autonomously, without humans, comparing the patient's vitals against pre-existing knowledge of his/her condition as well as any recommendations prescribed by the patient's doctor or healthcare professional.

Upon completing such an analysis, the server executes one of many predetermined action plans based on the outcome of the analysis. These action plans are general to all patients being monitored, but can also be specifically catered and modified for each individual patient. Such modifications, although made by healthcare professionals, are implemented and executed automatically by the MORF system. The determination of which action plan is to be executed is based on the analysis of the sensor data and parameters laid out by the healthcare professional. Once a specific action plan is selected, it is then executed immediately to ensure prompt responses should they be necessary. The action plans can consist of such simplistic tasks such as data logging, or more complicated tasks such as some form of communication of the patient's vitals to a healthcare professional. There are also action plans that take into account emergency protocols, such as alarming Emergency Medical Services (EMS) and/or the patient's personal doctor.

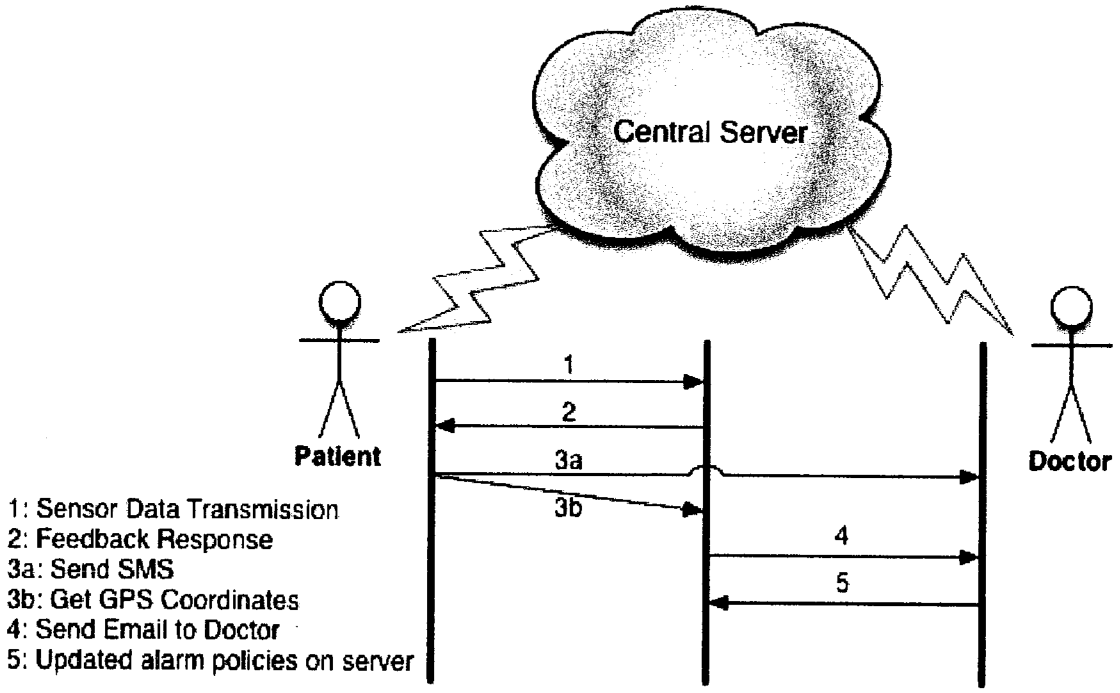


Figure 1 - MORF Handshaking Process

Figure 1 shows the handshaking process for the MORF system. In step 1 the mobile unit transmits its sensor data to the server for analysis. Upon completion of this analysis the server responds back to the mobile unit with a list of actions that must be taken. Steps 3a, 3b and 4 are taken only if the server requests that these actions be taken in its feedback response (step 2). Finally, the doctor can modify the alarm management profile or alert notification profile of the patient and updating the server with these modifications, as seen in step 5.

The feedback system for this project has been designed to give the reasoning server greater control over the mobile monitor. When a patient's vitals are abnormal in any way, the system can be set up so that the monitor is instructed to take more frequent measurements or to get more information, such as GPS location, from the sensors or mobile phone. If an alert message must be sent out, sensory and location information are sent along with the alert to both give the medical professional an idea of the patient's health status and to alert him of the patient's current location in the case of an emergency.

3.2 – System Architecture

The MORF platform is being designed in such a manner that allows for the easy implementation of a hospital-patient monitoring system. This platform is easily adaptable to monitor any vital sign of the patient to be monitored. The best way to view this system is to split it into three sections, namely, the sensory, mobile and server platforms. For the purpose of helping to explain these three sections, they will be referred to as the low, mid and high level implementations of this project, respectively. Figure 2 depicts an overview of the system architecture, including each subsystem.

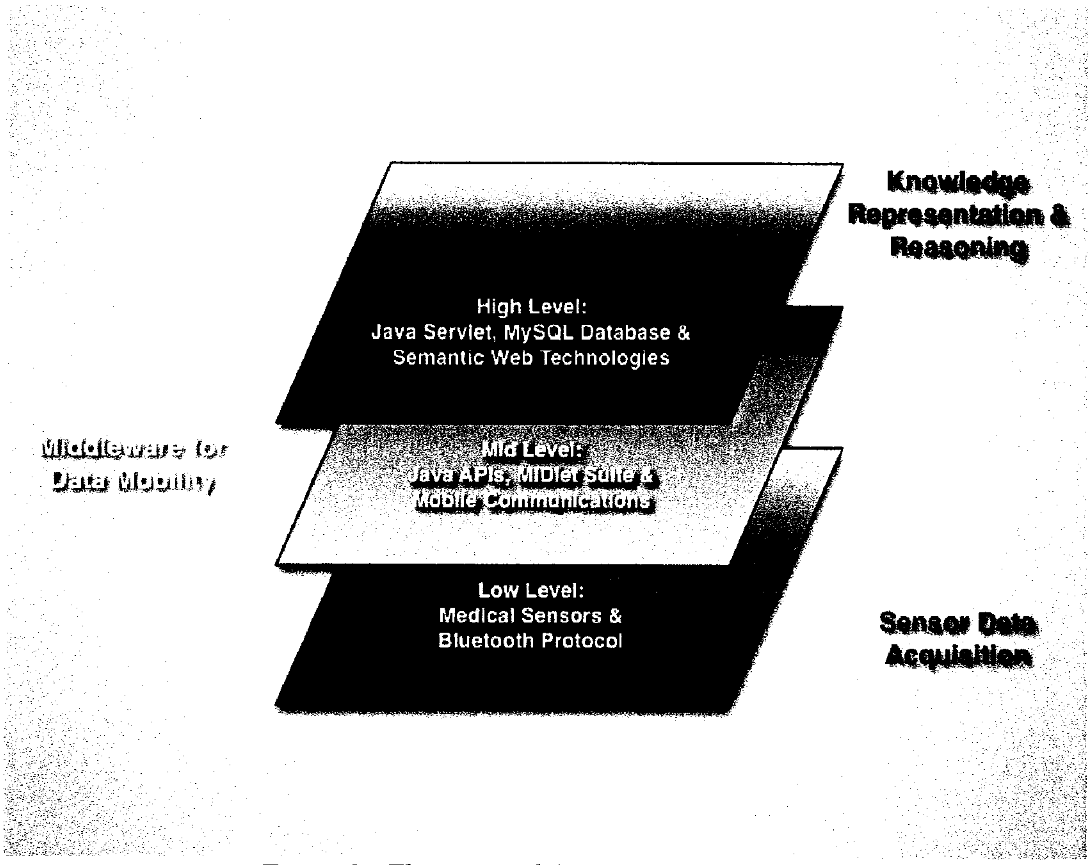

Knowledge Representation & Reasoning
High Level:
Java Servlet, MySQL Database &
Semantic Web Technologies
Middleware for Data Mobility
Mid Level:
Java APIs, MIDlet Suite &
Mobile Communications
Low Level:
Medical Sensors &
Bluetooth Protocol
Sensor Data Acquisition

Figure 2 - Three-Tiered Conceptual Architecture

By splitting this system into subsections we can have a more intricate and specific understanding about how the system is designed to function. It will also allow for a greater general understanding of the system's purpose and design. The low-level tier deals with the sensor data acquisition and the medical sensors. Sensor data acquisition is concerned with how the data from the medical sensors is retrieved in a usable format. The mid-level tier is concerned with the platform's middleware. It consists primarily of the function of the mobile phone, its programming and communication with the central server. The high-level tier covers how knowledge is managed in the system and how reasoning is performed based on that knowledge. Technologically, this tier is composed of the Apache server, Java servlets, MySQL database and various semantic web technologies.

3.2.1 – Hardware Design & Sensor Data Acquisition

MORF has been designed in such a way that a number of different sensors would be easily integrated into the system. The system will function with both wired and wireless sensors. In the case of a wired sensor, it will connect to a wireless Bluetooth module which then transmits the sensed data to the phone when it is requested. The only requirement for the wired sensors is that they have the capability to transmit their data electronically to the Bluetooth module. This data can be in either an analog format or digital format. If the data is in analog format, the microcontroller onboard the Bluetooth module will convert that data into a digital format using its analog-to-digital converter (ADC). On the other hand, if the data is digital, no ADC is required because the module can read and understand it directly without conversion. In order for this system to have maximum flexibility it is crucial that a large variety of sensors be supported. The number of different types of wired sensors that are supported is directly proportional to the number of communication protocols supported by the Bluetooth module's onboard microcontroller. In this particular case, there are numerous communication protocols supported by this device. These protocols include: USB, RS232, Universal Asynchronous Receive and Transmit (USART), Serial Programming Interface (SPI) and Inter-Integrated Circuit (I2C). Supporting all of these different serial communication protocols provides maximum flexibility as it pertains to which sensors can be used in this system.

The μIceBlue module is an embedded control module designed to allow the easy integration of Bluetooth technology into embedded circuit applications. Its structure is based around the PIC18F4550 microcontroller from Microchip Technology Inc. as well as the core design of the PearlBlue module by Emxys Embedded Instruments [15]-[16].

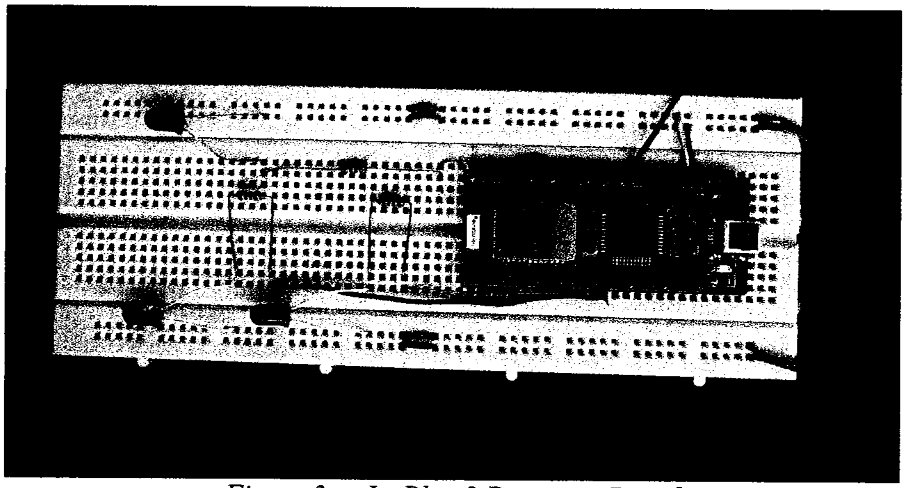


Figure 3 - μIceBlue 2 Prototype Board

The photograph shown in Figure 3 shows the current configuration that is being used for this board. The LEDs are simply for testing purposes and are not required for the actual circuit operation. The μIceBlue's low electrical operating requirements are perfect for mobile applications such as MORF. The μIceBlue has been designed to connect directly to the USART of the microcontroller. This allows the microcontroller to send any data via Bluetooth by simply writing it to the USART transmission buffer. This simplifies wireless transmission by completely bypassing the Bluetooth stack as it is already implemented in the Bluetooth module itself. This allows for a plug and play style interface that makes the wireless communication of virtually any data as simple as connecting it to the microcontroller and sending that data through the USART. The onboard PIC18F4550 microcontroller is the brains behind the entire operation of this module. Microcontrollers are small microprocessors designed for simple embedded applications. This particular microcontroller supports various technologies such as USB, various serial transmission protocols, multiple power-managed modes for low power applications, 4 timer modules, 32KB of flash program memory, 4096B of data memory, a 10-bit ADC module, and a plethora of interrupts, input/output (I/O) ports, and other features [15]. Hardware features such as these are very useful to customize the various hardware and software control functions needed to synchronize and tune sensor data acquisition and feedback operations. The most important feature of the PIC18F4550 is the USART module. It is configurable as an 8-bit or 9-bit module and can operate in either synchronous or asynchronous modes. In synchronous mode the transmissions require a clock source to keep their timing correct whereas in asynchronous mode they do not. For this system, the PIC is configured as an 8-bit, asynchronous module, operating with a baud rate of 19200 bauds [16]. Since we will be transmitting the data in form of ASCII characters, we only require the use of the 8 bits. In this case, the 9th bit will act as a parity check to ensure the proper reception of the data. The baudrate directly affects the speed of the transmission. It is for this reason that we have chosen to use a high baudrate in order to have a higher transmission rate from the microcontroller to the Bluetooth module. This rate is close enough to being instantaneous that it doesn't need to be any faster for the purposes of this project. Any faster of a transmission rate would just increase the likelihood of packet loss and is unnecessary.

In the case of the wireless sensors, the only requirement is that they be Bluetooth capable. Using a Bluetooth sensor is much simpler than a wired sensor as it completely removes one step from the process of transmitting the data to the phone, that step being sending it from the sensor to the μIceBlue module. The current sensor being used in the MORF system is the Nonin 4100 pulse oximeter. It measures blood-oxygen saturation levels (SpO2) as well as heart rate. The 4100 connects to the phone via Bluetooth at which point it transmits it's data which is then relayed through the phone to the server for processing. Below is a picture of the Nonin 4100 in operation. It consists of both a finger-clip and the SpO2 module which houses the units Bluetooth transceiver.

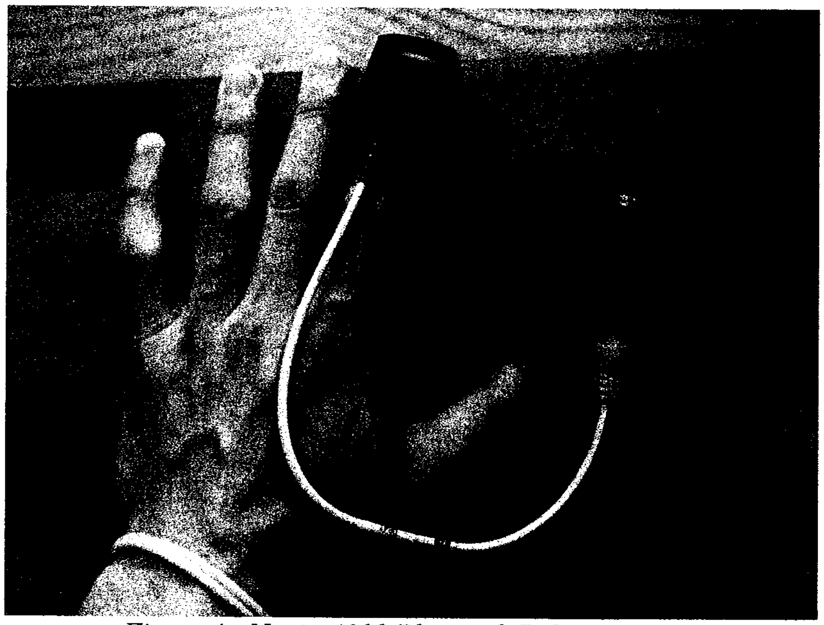


Figure 4 - Nonin 4100 Bluetooth Pulse Oximeter

This unit is capable of a range between the patient and it's host of greater than 30 feet.

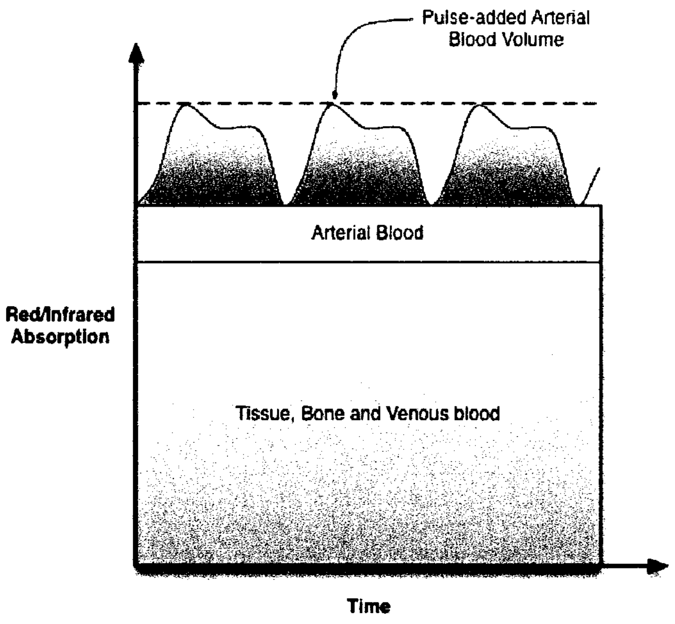
It operates at a frequency of 2.4GHz and utilizes a 32-bit encryption key. Electrically, the 4100 operates on 2 AA batteries and has a life of about 120 hours in continual operation mode, or 6 months at 10 spot checks per day in spot-check mode [17]. See the following table for the full set of operating specifications for the Nonin 4100.

Specification	Value
SpO2 range	0-100%
Heart rate range	18-300 bpm
Measurement wavelengths	Red – 660nm @ 3mW (nominal)
	Infrared – 910nm @ 3mW (nominal)
Operating temperature	-20°C to +50°C
	-20°C to +50°C
Bluetooth specification	v1.1
Bluetooth profiles	Serial Port Profile

Table 1 - Nonin 4100 Specifications and Physical Characteristics [17]

The Nonin 4100 has a no-motion range of 0-100% for SpO2 and 18-300bpm for heart rate [17]. As with the µIceBlue module, it uses the Bluetooth serial port profile to communicate with other Bluetooth devices. The saturation of peripheral oxygen (SpO2) is defined as the relative measure of how much oxygen is in the bloodstream [18].

Pulse oximetry operates by emitting one red and one infrared light through the patient's finger. On the other side of the finger is a light sensor which measures the amount light that passes through the finger and therefore the amount of light that was absorbed by the finger. The amount of light that is absorbed by tissue, bone, venous and arterial blood are all constant, except that when the heart pumps blood through the finger the amount of arterial blood changes. This fact allows the sensor to filter out pulses and therefore calculate the individuals heart rate as shown in Figure 5.



Pulse-added Arterial Blood Volume
Arterial Blood
Red/Infrared Absorption
Tissue, Bone and Venous blood
Time

Figure 5 - Red/Infrared Absorption in Pulse Oximetry

Hemoglobin is the oxygen-carrying part of the red blood cells. Oxygenated hemoglobin (HbO_2) absorbs more infrared light than red light, where deoxygenated hemoglobin (Hb) absorbs more red light than infrared light. By comparing this discrepancy the sensor can effectively calculate the SpO2 percentage. An SpO2 measurement is calculated by the following equation:

$$SpO_2 = \frac{HbO_2}{HbO_2 + Hb}$$

Equation 1 - Formula for Calculating SpO2 [18]

It is from the equation shown in Equation 1 we can derive a true understanding of the meaning of Sp02. As can be seen above, Sp02 is the amount of oxygenated hemoglobin in the blood relative to the total amount of hemoglobin in the blood (oxygenated and deoxygenated) and is represented as a percentage. For the average human being, oxygen saturation levels of anywhere from 95-100% are considered to be normal [19].

	Accuracy for Sp02 of 70-100%	Accuracy for Heart Rate
No motion	2 digits	3 digits (18-300 bpm)
Motion	2 digits	5 digits (40-240 bpm)
Low perfusion	3 digits	3 digits (40-240 bpm)

Table 2 - Nonin 4100 Sp02 & Heart Rate Sensor Accuracy [17]

The above table lists the accuracy levels of the Nonin 4100 pulse oximeter for both Sp02 and heart rate. The module can operate in one of two different turn-on modes and one of four configuration modes giving a total of eight distinct operating modes. The table below show a quick glimpse of the operational features of each of these modes.

Configuration Mode		Amount of Data Transmitted	Data Rate	Plethysmograph
Sensor	Spot-Check			
1	A	3 Bytes	1 Byte/second	N/A
2	B	5 Frames (125 Bytes)	75 Bytes/second	8-bit
4	C	5 Frames (125 Bytes)	75 Bytes/second	16-bit
8	D	4 Bytes	1 Byte/second	N/A

Table 3 - Nonin 4100 Configuration Modes [17]

The mode that we have chosen to use in our system is mode D. This is a spot-checking mode which is perfect for this application since we only need the module to gather data when it is requested by the server. This helps to lengthen the module's battery life and gives our system complete control over the device.

Every time a Bluetooth connection is established between the mobile phone and the Nonin 4100, the configuration command is sent to tell the pulse oximeter what mode to operate in. In our case, the command "D" is sent to the oximeter to tell it to operate in the 4-byte spot-check mode. After configuration is complete, the module begins sending data to the mobile phone via Bluetooth. The data is transmitted one byte at a time until all four bytes have been received. The table below shows the bit-wise formatting of the data packets transmitted from the oximeter.

Byte #	Bit #	Name/State	Description
Byte 0	7	1	Always set
	6	SNSD	Sensor disconnect. Set if finger-clip sensor is disconnected.
	5	OOT	Out of track. Set if lack of consecutively good pulse signals.
	4	LPRF	Low perfusion. Low signal quality.
	3	MPRF	Marginal perfusion. Medium signal quality.
	2	ARTF	Artifact.
	1	HR8	HR8-HR0: 4-beat average heart rate value
	0	HR7	
Byte 1	7	0	Always cleared
	6	HR6	HR8-HR0: 4-beat average heart rate value
	5	HR5	
	4	HR4	
	3	HR3	
	2	HR2	
	1	HR1	
	0	HR0	
Byte 2	7	0	Always cleared
	6	SP6	SP6-SP0: 4-beat average SpO2 Level
	5	SP5	
	4	SP4	
	3	SP3	
	2	SP2	
	1	SP1	
	0	SP0	
Byte 3	7	0	Always cleared
	6	R	Reserved
	5	R	Reserved
	4	R	Reserved
	3	SNSF	Sensor fault. Data provided by sensor is unusable for analysis by the module.
	2	TURN ON	Turn on mode. 1 = Spot check mode. 0 = Sensor mode.
	1	CRIT BAT	Critical battery condition. Upon sending this status the module will power down.
	0	LOW BAT	Low battery condition. Less than 30 minutes run-time remaining.

Table 4 - Nonin 4100 Data Format for Mode D [17]

For the purposes of this research, the only data that we are interested in are the bits containing the SpO2 and heart rate data (highlighted in blue). Once this data is received it is converted from a binary value to its decimal equivalent in preparation for being relayed on to the server.

For the purposes of this research we chose to use the Nokia E71 as our test phone. The phone itself is running version 9.2 of the Symbian Operating System (OS) and has a screen resolution and colour depth of 320x240 pixels and 24 bits respectively. Weighing in at only 127 grams, this device has a full QWERTY keyboard and a single ARM 11 processor running at 369 Mhz [20].

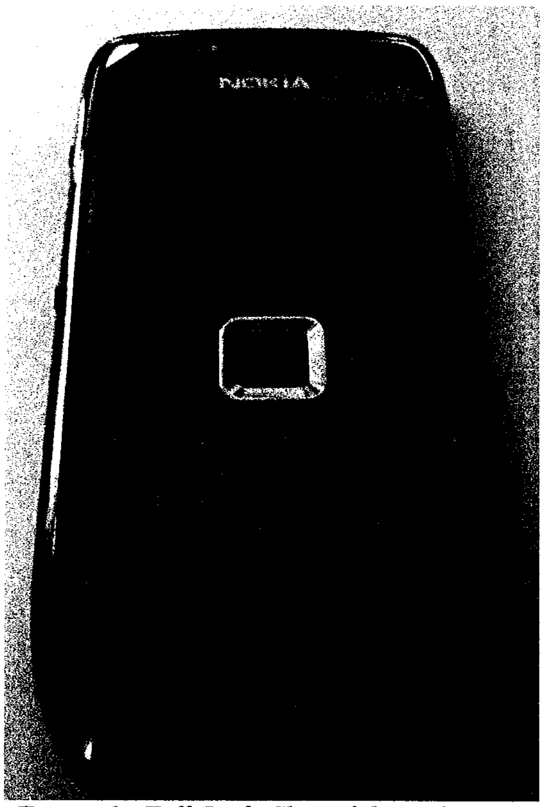


Figure 6 - Full-Body Shot of the Nokia E71

There were several criteria that needed to be met in the mobile phone that finally led us to this particular model. Table 5 summarizes the minimum requirements for a smart phone to have in order to function with the MORF health monitoring system.

	Required Specifications
Generic Features	Java-capable – Java 2.0 Mobile Edition (J2ME) GPS module Bluetooth module Data-capable (EGPRS, GPRS, HSCSD, HSPA, WCDMA etc...) Messaging (MMS or SMS)
Java Application Programming Interfaces (APIs)	JSR 139 – Connected, Limited Device Configuration (CLDC) JSR 118 – Mobile Information Device Profile (MIDP) JSR 82 – Bluetooth API JSR 172 – J2ME Web Services Specification (XML Parser Package) JSR 179 – Location API JSR 205 – Wireless messaging API
Bluetooth Profiles	Serial Port Profile (SPP)

Table 5 - Smart Phone Minimum System Requirements [20]

Since, for the purposes of the research, we wanted to test the maximum amount of features with this system we had to choose a phone that supported all of the Java Application Programming Interface (APIs) listed in Table 5. These APIs are required to allow a mobile Java program, called a MIDlet, to access and control the various modules or services provided by the phone. Without these APIs it would be impossible to write an application that would, for example, control the Bluetooth or GPS modules on the smart phone. It is by no means necessary for a phone to support all of these APIs. It is only necessary for a phone to be used by this system to support the APIs linked with the features desired for operation. For example, it is neither necessary for a mobile device to support the GPS API or to even have a GPS module if this function is not desired in the operation of the system for a specific patient. This creates a multitude of layers of features that can be used by this system. As some of these features are not necessary for simple monitoring purposes, they do not all need to be supported.

3.2.2 – Operational Environment & Case Scenario

This section describes the major software components that compose the MORF system environment. Figure 7 shows the main technologies used in our system and how they relate to each other.

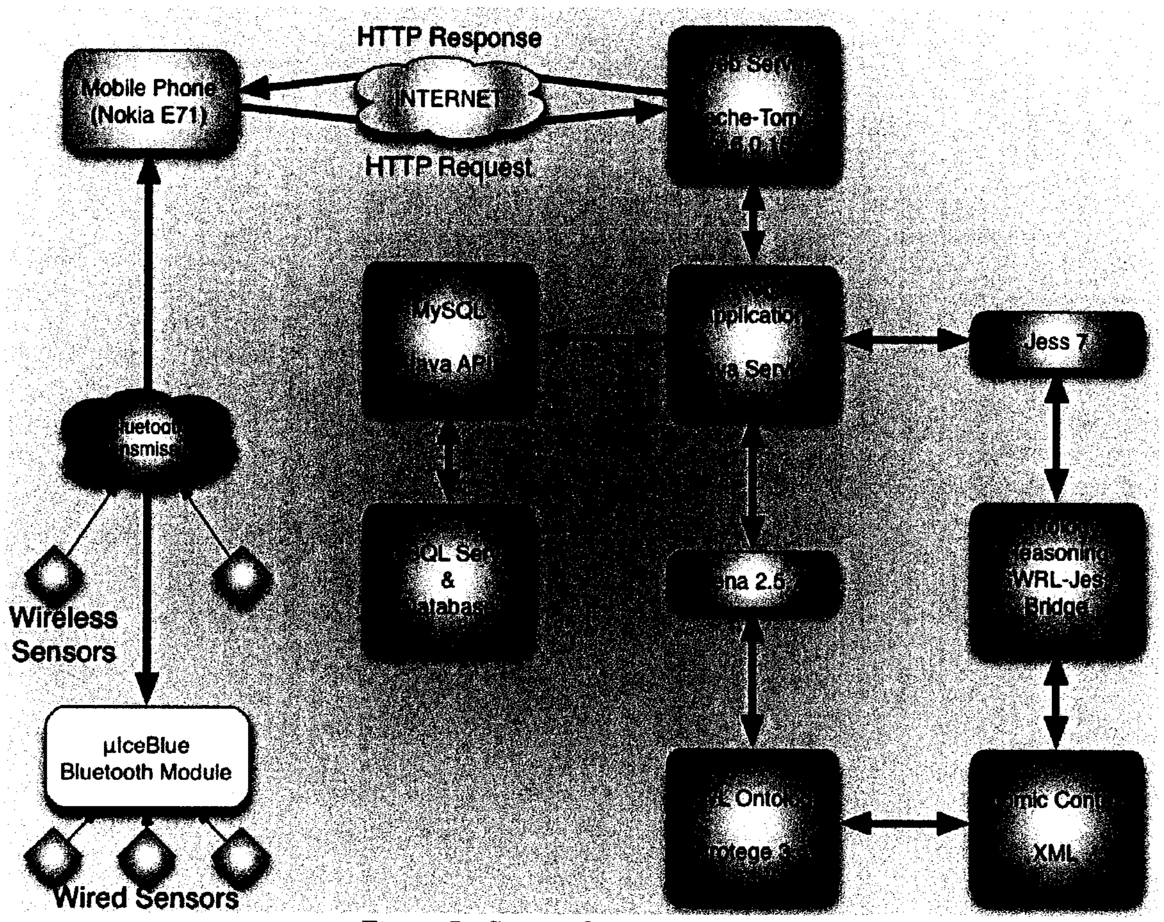


Figure 7 - System Overview Diagram

Using the Eclipse IDE (v3.4.2) we developed a Java application for smart phones, known as a MIDlet, which gathers the sensor data from the sensors. Once received, the mobile phone forwards that data on to the Apache-Tomcat (v6.0.18) server. The servlet, written in Java and hosted on the Apache-Tomcat server, receives the incoming data and stores it in a MySQL database using the MySQL Java API. This data is also stored temporarily in the system ontology. System queries and modifications of the ontology are performed using Jena, a Java framework designed for building semantic web applications. Context modelling and ontology-editing is performed in Protégé 3.4. SWRL rules are designed using the SWRL Tab plugin for Protégé and inferred using the SWRL-Jess bridge via the Jess 7.0 rule engine API for Java. Each of these components will be discussed in further detail in sections 3.4, 4.1 and 4.2.

Before we get into any further detail about the technical operation of the MORF platform, we feel it necessary to further explain the functional operation of this system. To do this, take the following fictional example:

Mr. Smith has recently suffered a heart attack and has been admitted into a local hospital. His physician, Dr. Johnson has prescribed that he use the MORF Health Monitoring System so that he could recover while being at home with his family. Dr. Johnson set up Mr. Smith with heart rate, blood pressure and temperature sensors to be monitored while he recovers.

As Mr. Smith goes about his day, sensor readings are taken in intervals of approximately 30 minutes and sent to the hospital monitoring server to be analyzed and then logged away for archiving. One day, Mr. Smith's blood pressure and heart rate slightly exceeded what his physician prescribed and MORF went to a low level alert increasing the frequency of the measurements to once every 15 minutes in order to keep a closer eye on Mr. Smith. Dr. Johnson was notified via email and SMS of Mr. Smith's slightly elevated vitals and contacted him to inform him that the monitor would be taking more frequent measurements to ensure his condition stabilized. Dr. Johnson also prescribed him some medication to help lower his blood pressure, as this was the source of the problem as discovered by MORF.

Over the next week, Mr. Smith's vitals returned to normal and the monitor returned to taking measurements at intervals of 30 minutes. However, a few days later, Mr. Smith's heart rate and blood pressure began to rise again. This time they greatly exceeded that which was prescribed by the doctor. MORF went into a high level alert and began taking measurements of Mr. Smith's vitals at 1 minute intervals. Since his vitals kept getting worse and not better, Emergency Medical Services were contacted and given Mr. Smith's GPS location and Dr. Johnson was notified of the situation. With the help of the MORF Health Monitoring System, Mr. Smith was admitted to the hospital and was treated to prevent what would have become a second heart attack. Also, with the data recovered from this situation the doctors were able to learn more about the preceding symptoms of heart attacks and how better to treat them. This is but one possible situation of how MORF could be used in practice in everyday life.

3.3 – System Mobility

This section covers the technologies that make the MORF system mobile as well as the software components that enable data acquisition from the sensors. The two key technologies that allow our system to function as a mobile unit are Bluetooth and GSM. On the other hand, the software components that enable data acquisition are comprised of a number of Java APIs. These APIs allow our MIDlet to communicate with some of the hardware components of the mobile phone such as its Bluetooth transceiver and GSM modem.

3.3.1 – Bluetooth

The Bluetooth protocol is a very complex protocol to implement from scratch. It has a layered protocol architecture, called a stack, which defines the operating guidelines for the wireless communication. Bluetooth's main application is in that of short-range wireless communications for PANs (Personal Area Networks) between devices such as laptops, mobile phones and PDAs. It operates in the 2.45 GHz frequency range within the unlicensed industrial, scientific and medical (ISM) band. Within the protocol, two classes of devices are defined to indicate a relative range and transmission power. Class 1 devices, such as the µIceBlue and the Nonin 4100, have a range of up to 100m with a power output of 20dBm, whereas class 2 devices have a more limited range of 10m and power output of 0dBm [21].

Bluetooth devices adopts a transmission method know as Frequency Hopping Spread Spectrum (FHSS). FHSS is achieved by rapidly changing the signal's carrier frequency to different frequency channels in a pseudorandom format. The reason this switching is only pseudorandom and not just random is that both the transmitting and receiving devices know the sequence by which the carrier frequency switches. This has multiple advantages such as high resistance to narrowband interference, and low interference for other devices operating in the same frequency band. But perhaps the most important advantages of FHSS is that it makes signal interception extremely difficult. This is because, to any other device, the transmissions between two communicating Bluetooth devices is only seen as a slight elevation in background noise. Any device that doesn't know the switching sequence, can't understand the data transmissions and therefore sees them as random noise [22]. On top of this, Bluetooth utilizes encryption keys to further increase the security of its data transmissions. Such security measures are important when dealing with sensitive data such as the patient data that will be transmitted by MORF [21].

The Bluetooth specification defines what is known as the Serial Port Profile (SPP). The SPP is implemented in the lower level of the Bluetooth stack and provides a way for low-level serial devices to implement a high level wireless protocol with ease. This is what makes using the μIceBlue and Nonin 4100 so simple. They utilize the SPP to make wireless communication as simple as wired serial communication. This also works with more complicated devices, such as a computer or a mobile phone. When a Bluetooth device connects to one of these devices it sets up what is known as a virtual serial port. When data is sent through this virtual serial port it is automatically transmitted through the devices Bluetooth transceiver.

Among the many short range protocols that compete with Bluetooth, only one stands out that would have rivalled in suitability for the MORF system. That protocol is the ZigBee protocol. Ideally, ZigBee would have been the most ideal wireless protocol for this project because it is designed for low power consumption and focuses on control and automated applications. The reason we chose to use Bluetooth over ZigBee is due to Bluetooth being much more widely used on most handheld devices. Almost every mobile phone on the market today supports Bluetooth where very few support ZigBee. This is a major flaw since MORF's operation necessitates communication between the sensors and a mobile device and is the primary reason of choosing Bluetooth instead [23].

The Nokia E71 utilizes the JSR-82 Java API for Bluetooth communication. This Bluetooth API for mobile phones allows a developer to access the Bluetooth transceiver on the phone from software. In the Java MIDlet that we developed for mobile smart phones, we utilized this API to communicate with both the μIceBlue Bluetooth module and the Nonin 4100 Bluetooth pulse oximeter. The following section of code in Table 6 should help to explain how this API is used.

```
try {
        stream = (StreamConnection) Connector.open(url);    //Open a stream with the Bluetooth device's URL
        in = stream.openInputStream();                      //Create an input stream to receive data
        out = stream.openOutputStream();                    //Create an output stream to send data
        String msg = "D1";                                  //Send the command to activate the appropriate
        out.write(msg.getBytes());                          //operation mode and write the data to the output
        out.flush();                                        //stream.
        for (int j = 0; j < 4; j++)
                r[j] = in.read();                           //Loop this until all data is received
        int spo2 = r[3];                                    //Set the SpO2 value
        int pulse = 0;
        byte b = (byte)(r[1]);
        //Get Heart Rate
        if ((r[1] & 0x02) == 0x00)                          //This section of code is used to get the
        {                                                   //heart rate from the Bluetooth sensor.
                if((r[1] & 0x01) == 0x00)                   //
                        pulse = r[2];                       //
                else                                        //The heart rate data is spread across two
                        pulse = r[2] + 128;                 //bytes sent from the sensor.
        }                                                   //
        else                                                //
        {                                                   //This code simply interprets the correct
                if((r[1] & 0x01) == 0x00)                   //values from those two bytes of data.
                        pulse = r[2] + 256;                 //
                else                                        //
                        pulse = r[2] + 256 + 128;           //For more information or to see the data
        }                                                   //format see section 3.2.1.
        data[0] = Integer.toString(spo2);
        data[1] = Integer.toString(pulse);
        log(data[0]);                                       //Write the received values to the screen.
        log(data[1]);
        midlet.setSensorData(data);                         //This method sets the sensor data values.
        data[0] = "";                                       //Reset all of the variables used so that when
        data[1] = "";                                       //the sensor data is acquired the next time, the
        pulse = 0;                                          //values are default or null values.
        spo2 = 0;
        out.close();                                        //
        in.close();                                         //Close all streams that are open
        stream.close();                                     //
        }
        catch (IOException e)
        {
                log(e);
        } finally
        {
                log("Bluetooth stream closed.");
        }
```

Table 6 - Java MIDlet Bluetooth Communication Code Sample

First we see that a stream is created from the Nonin 4100's device URL. From this stream, input and output streams are created. The message "D1" is then transmitted to the sensor. This is a command to tell the sensor which operating mode to utilize. Data is then read from the sensor, sorted and then stored in appropriate variables. The streams are then closed and preparation is made for the next time the phone must communicate with the sensor.

3.3.2 – GSM & HSPA

The Nokia E71 is a phone that utilizes the GSM standard and operates on a 3G network. It utilizes High-Speed Packet Access (HSPA), which is a 3G technology, for all of its data communication requirements, that is, for connecting to the Internet.

GSM is currently the leading standard for all mobile phones in the world with a global market use of approximately 80%. This figure translates to over 3 billion people covering more than 212 countries and territories. GSM is part of the 3G family of standards and uses a modulation technique known as Gaussian minimum-shift keying (GMSK). It operates in the 900MHz or 1800MHz frequency bands (850MHz or 1900MHz bands in North America) with a maximum handset transmission power of 2 watts in the 850/900MHz bands and 1 watt in the 1800/1900MHz bands [24].

HSPA is the method by which MORF's mobile unit communicates with the central server. It allows for increased downlink and uplink speeds of up to 14 Mbit/s and 5.8 Mbit/s respectively. It achieves this increase in speed by sacrificing variable spreading factor and fast power control. In terms of modulation, HSPA uses a quadrature phase-shift keying (QPSK) or, in good radio conditions, 16QAM (Quadrature amplitude modulation). It also uses a redundancy technique known as hybrid automatic repeat-request (HARQ) which transmits the user data multiple times, each using a different encoding scheme. When a corrupted packet is detected it is later combined it with the retransmitted packets to ensure a flawless transaction [25].

It is these technologies that give the MORF system the freedom of mobility. Without them, achieving this level of versatility is near impossible.

In order to send the acquired sensor data to the server the MIDlet must create an HTTP connection and then stream it to the server. After streaming the data to the server the program waits until the server is done processing that data and responds back with instructions on what to do next. Below is a section of the MIDlet code that deals with server communication.

```
public static String sendData(String text) {                                      //Method sendData
        String response = "";
  StringBuffer sb = new StringBuffer();                                           //Create a new StringBuffer
  try {
    HttpConnection c = (HttpConnection) Connector.open(url);                      //Open a HTTP connection
    c.setRequestProperty("User-Agent","Profile/MIDP-1.0, Configuration/CLDC-1.0"); //with the specified URL and
    c.setRequestProperty("Content-Language","en-US");                             //set the requested header
    c.setRequestMethod(HttpConnection.POST);                                      //property values.

    DataOutputStream os = (DataOutputStream)c.openDataOutputStream();             //Create an output stream
    os.writeUTF(text.trim());                                                     //with the new connection
    os.close();                                                                   //and write the data to the
    os.flush();                                                                   //stream.

    // Get the response from the servlet page.
    DataInputStream is =(DataInputStream)c.openDataInputStream();                 //Create an input stream and
    int ch;                                                                       //wait for the server response
    sb = new StringBuffer();
    while ((ch = is.read()) != -1) {
      sb.append((char)ch);
    }
    response = sb.toString();                                                     //This is the server response
    is.close();
    c.close();
  }catch (Exception e) {
    showAlert(e.getMessage());
  }
  return response;                                                                //Return the HTTP response
}
```

Table 7 - Java MIDlet HTTP Connection Code Sample

First, the MIDlet opens a new HTTP connection with the server URL. It then specifies the User-Agent and Content-Language properties as requested by the server to help identify what type of device is accessing it. Using HTTP POST it then opens a data output stream through which it streams the sensor data.

3.4 – DATA STORAGE AND MANAGEMENT INFRASTRUCTURE

The MORF system incorporates a MySQL database to store all of the sensor data acquired form the mobile monitors. All of the data is sorted into separate data tables for each individual patient. This makes access to patient-specific data extremely simple and logical. Figure 8 shows the schema "patientdata" which is a collection of all of the data tables for each patient that is created in our system.

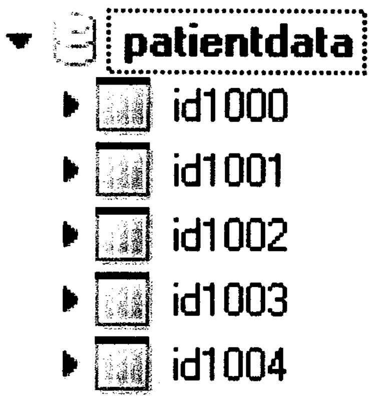
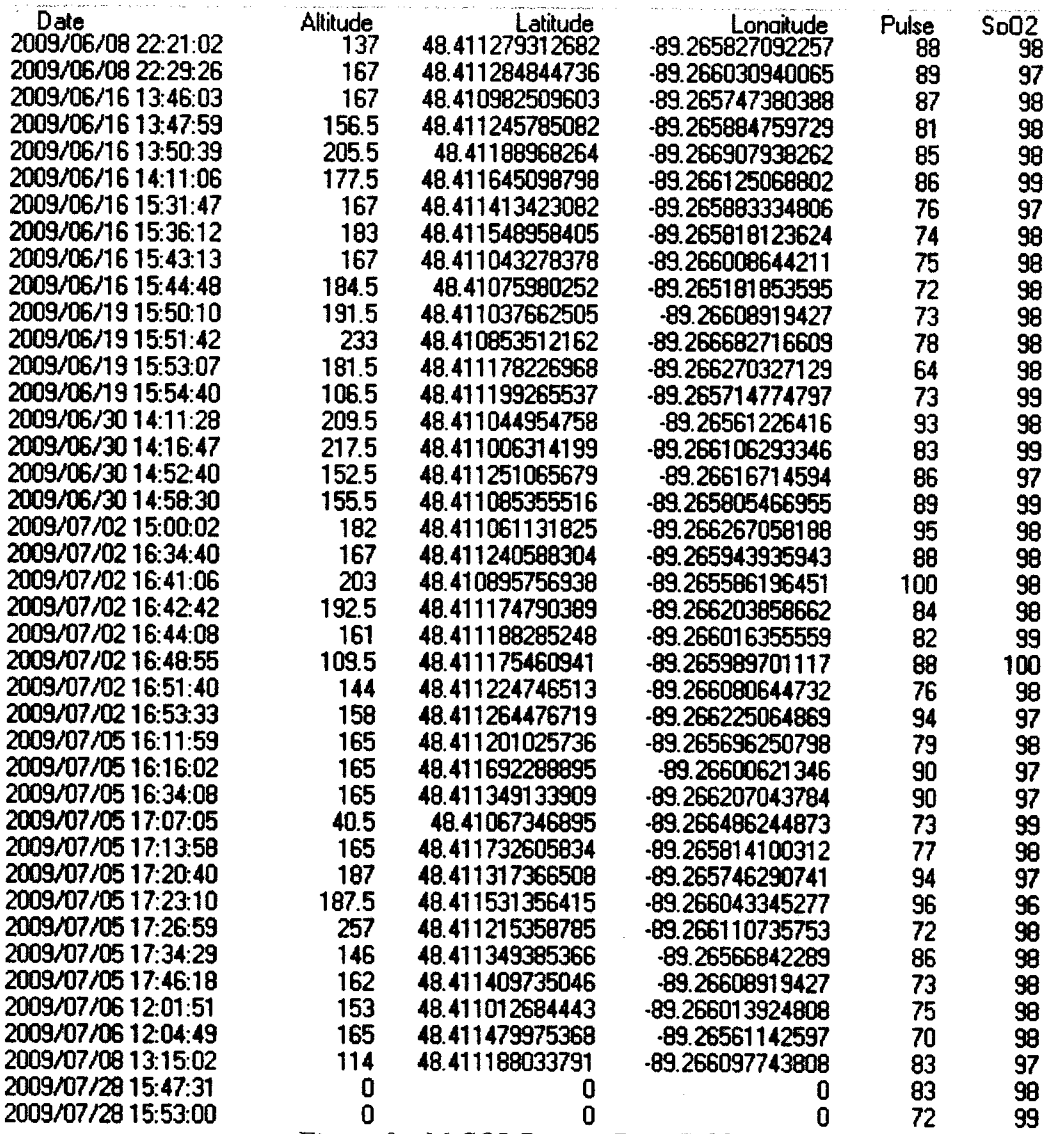

patientdata
id1000
id1001
id1002
id1003
id1004

Figure 8 - MySQL Data Tables

Each one of these data tables represents a different patient as specified by the patient identification number (PID). When new sensor data is received by the server, the server uses the PID of the incoming data to know which data table to modify. A new entry is then created in that table with the received sensor data sorted by time-stamp. Figure 9 shows some real-life data as acquired and stored by the system:



Date	Altitude	Latitude	Longitude	Pulse	SpO2
2009/06/08 22:21:02	137	48.411279312682	-89.265827092257	88	98
2009/06/08 22:29:26	167	48.411284844736	-89.266030940065	89	97
2009/06/16 13:46:03	167	48.410982509603	-89.265747380388	87	98
2009/06/16 13:47:59	156.5	48.411245785082	-89.265884759729	81	98
2009/06/16 13:50:39	205.5	48.41188968264	-89.266907938262	85	98
2009/06/16 14:11:06	177.5	48.411645098798	-89.266125068802	86	99
2009/06/16 15:31:47	167	48.411413423082	-89.265883334806	76	97
2009/06/16 15:36:12	183	48.411548958405	-89.265818123624	74	98
2009/06/16 15:43:13	167	48.411043278378	-89.266008644211	75	98
2009/06/16 15:44:48	184.5	48.41075980252	-89.265181853595	72	98
2009/06/19 15:50:10	191.5	48.411037662505	-89.26608919427	73	98
2009/06/19 15:51:42	233	48.410853512162	-89.266682716609	78	98
2009/06/19 15:53:07	181.5	48.411178226968	-89.266270327129	64	98
2009/06/19 15:54:40	106.5	48.411199265537	-89.265714774797	73	99
2009/06/30 14:11:28	209.5	48.411044954758	-89.26561226416	93	98
2009/06/30 14:16:47	217.5	48.411006314199	-89.266106293346	83	99
2009/06/30 14:52:40	152.5	48.411251065679	-89.26616714594	86	97
2009/06/30 14:58:30	155.5	48.411085355516	-89.265805466955	89	99
2009/07/02 15:00:02	182	48.411061131825	-89.266267058188	95	98
2009/07/02 16:34:40	167	48.411240588304	-89.265943935943	88	98
2009/07/02 16:41:06	203	48.410895756938	-89.265586196451	100	98
2009/07/02 16:42:42	192.5	48.411174790389	-89.266203858662	84	98
2009/07/02 16:44:08	161	48.411188285248	-89.266016355559	82	99
2009/07/02 16:48:55	109.5	48.411175460941	-89.265989701117	88	100
2009/07/02 16:51:40	144	48.411224746513	-89.266080644732	76	98
2009/07/02 16:53:33	158	48.411264476719	-89.266225064869	94	97
2009/07/05 16:11:59	165	48.411201025736	-89.265696250798	79	98
2009/07/05 16:16:02	165	48.411692288895	-89.26600621346	90	97
2009/07/05 16:34:08	165	48.411349133909	-89.266207043784	90	97
2009/07/05 17:07:05	40.5	48.41067346895	-89.266486244873	73	99
2009/07/05 17:13:58	165	48.411732605834	-89.265814100312	77	98
2009/07/05 17:20:40	187	48.411317366508	-89.265746290741	94	97
2009/07/05 17:23:10	187.5	48.411531356415	-89.266043345277	96	96
2009/07/05 17:26:59	257	48.411215358785	-89.266110735753	72	98
2009/07/05 17:34:29	146	48.411349385366	-89.26566842289	86	98
2009/07/05 17:46:18	162	48.411409735046	-89.26608919427	73	98
2009/07/06 12:01:51	153	48.411012684443	-89.266013924808	75	98
2009/07/06 12:04:49	165	48.411479975368	-89.26561142597	70	98
2009/07/08 13:15:02	114	48.411188033791	-89.266097743808	83	97
2009/07/28 15:47:31	0	0	0	83	98
2009/07/28 15:53:00	0	0	0	72	99

Figure 9 - MySQL Patient Data Table

As can be seen above in Figure 9, the table is composed of six columns: date, GPS altitude, GPS latitude, GPS longitude, pulse and SpO2. Where the first four columns are common to all patients, the columns pertaining to the sensor data are customizable and expandable based on which sensors a patient uses. The patient who corresponds to the table above is only using the Nonin 4100 pulse oximeter and therefore only has pulse and

SpO2 data listed in the table.

The MySQL server is what hosts the database on the local network. In order to access the database, the Java servlet opens a connection to the MySQL server. Once the connection is open a query can then be made to the server utilizing standard SQL syntax. The section of Java code below shows the procedure for connecting to the MySQL server and the SQL syntax for inserting a new entry into the patient data table.

```
private static final String DRIVER = "com.mysql.jdbc.Driver";                       //Driver Filename
private static final String SOURCE = "jdbc:mysql://localhost/patientdata";           //MySQL Database URL
private static Connection dbConnection = null;
 try
 {
  Class.forName(DRIVER);
  dbConnection = DriverManager.getConnection(SOURCE,"root","password");              //Login info for database
 } // end try

 catch (ClassNotFoundException cnfe)
 {
  System.err.println(cnfe);
  System.exit(1);
 } // end catch

 catch (SQLException sqle)
 {
  System.err.println(sqle);
  System.exit(1);
 } // end catch

 String sqlStmt;                    //This string contains the MySQL command to add a new entry to the data table

 sqlStmt = "INSERT INTO id" + pID + " (Date, Altitude, Latitude, Longitude, Pulse, SpO2, bpSys, bpDia) VALUES('" +
         date + "', '" + gpsAlt + "', '" + gpsLat + "', '" + gpsLon + "', '" + heartRate + "', '" + spo2 + "', '" + bpSys + "', '" +
         bpDia + "')";

 try
 {
  Statement myStmt = dbConnection.createStatement();
  int updateQuery;
  updateQuery = myStmt.executeUpdate(sqlStmt);
  if (updateQuery != 0) {
    System.out.println("Table is created successfully and " + updateQuery + " row is inserted.");
  }

  dbConnection.close();
  dbConnection = null;
 } // end try

 catch (SQLException sqle)
 {
  System.err.println(sqle);
  System.err.println(sqlStmt);
 } // end catch
```

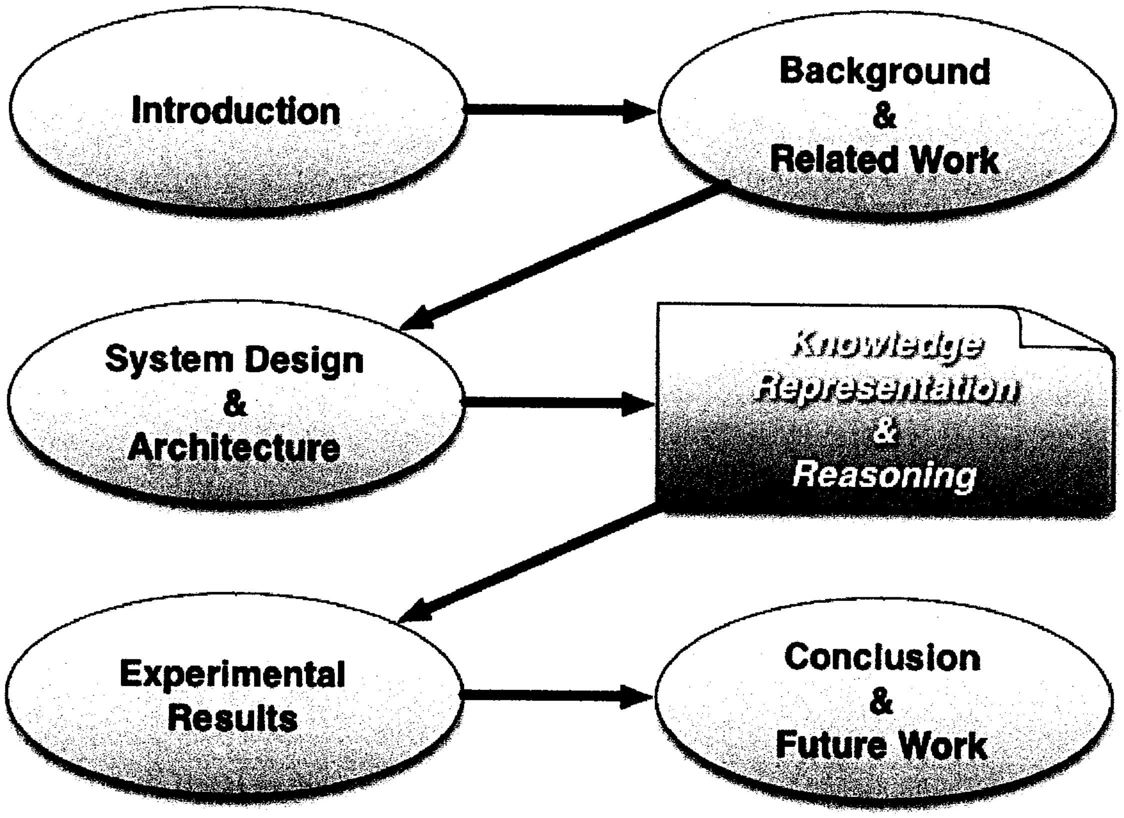
Table 8 - Java Code Sample to Add a New Data Entry via MySQL

First the program opens a connection with the MySQL server and logs in as root. Once a connection is established a SQL statement is created and then executed. This updates the database with the changes specified by the SQL statement. The entries into the database are currently static, in that the number of columns don't change. In future work, all of the SQL statements will be dynamic so as to allow the easy modification of a patient's data table to include more sensors.

In earlier stages of development, this database was also used to store a list of all of the patients connected to the system along with their personal information. This method of approach was abandoned when it was decided that this information be stored within the system's ontology instead. Storing that information in the ontology and the database created an unnecessary level of redundancy. This method of approach will be discussed further in chapter 4.

As briefly mentioned in section 3.2.2, we are using Apache Tomcat to host our server. Servlet programming is all performed in Java which controls the entire operation of the server-side of the system. The servlet is composed of a multitude of Java classes, each designed to perform specific tasks required for the proper operation of the MORF platform. These tasks include interfacing with external systems such as the MySQL database and the OWL ontology, controlling the web-based user interface and it's operation, as well as performing other crucial tasks that keep the system running.

chapter 4


Introduction
Background & Related Work
System Design & Architecture
Knowledge Representation & Reasoning
Experimental Results
Conclusion & Future Work

knowledge representation & reasoning

This chapter covers the section of the MORF system that deals with knowledge representation and reasoning. In section 4.1 we will discuss what knowledge management is and how it is implemented into our system. Furthermore, in section 4.2 we will discuss how we use reasoning techniques to infer further knowledge and model it in our system. Since the reasoning techniques used involve SWRL we will explain in detail a handful of the rules used to infer new knowledge in the MORF system.

4.1 – Knowledge Representation

Knowledge representation, as specific to MORF, deals with how we model our data and gather a sense of context for the patients that are being monitored by the system. This knowledge modelling is achieved through semantic web tools, namely RDF and OWL. RDF is the framework used to define relationships between the objects in our ontology whereas OWL is a language that allows machines to better interpret web content, such as data stored in RDF format, by providing a richer set of vocabulary and semantics [26]-[28]. Using ontologies to model our knowledge, as opposed to other methods, makes the system very scalable and easily modifiable. Adding further devices or sensors can be as simple as adding the appropriate instances within the ontology as opposed to having to hard-code all of that information into a program.

As mentioned above, the context model for this project is a custom model designed using OWL. OWL is a language used to develop web ontologies and is designed for use by applications that need to process the content of information instead of just presenting information to humans. It is a tool used to make information found on the Web meaningful. An ontology is a representation of a group of concepts, and the relationship(s) between those concepts, that reside within a domain [27]. Using ontologies and various reasoning methods, the system can process the incoming sensor data in a more organized and efficient manner. It also allows sharing and reusing such data in a useful way. The ontology in this system will take the sensory data (patient context) and Doctor prescribed parameters to create a method by which the engine can conclude what the health status of the patient is.

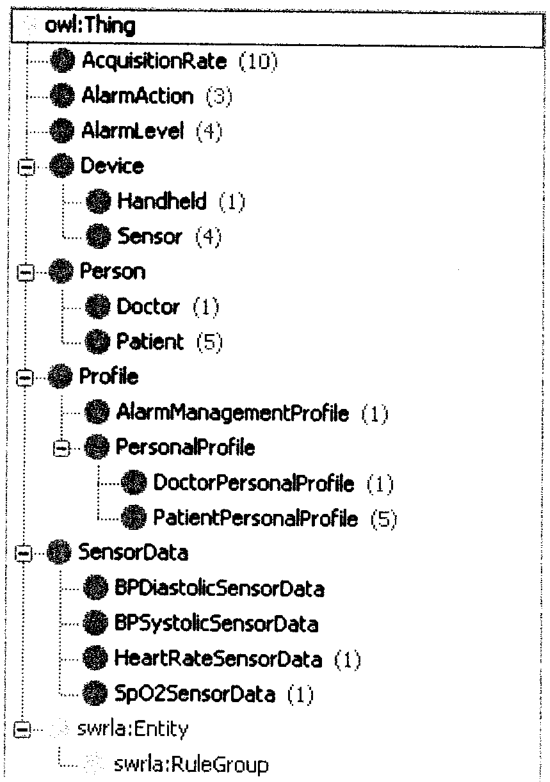

```
owl:Thing
├── AcquisitionRate (10)
├── AlarmAction (3)
├── AlarmLevel (4)
├── Device
│   ├── Handheld (1)
│   └── Sensor (4)
├── Person
│   ├── Doctor (1)
│   └── Patient (5)
├── Profile
│   ├── AlarmManagementProfile (1)
│   └── PersonalProfile
│       ├── DoctorPersonalProfile (1)
│       └── PatientPersonalProfile (5)
├── SensorData
│   ├── BPDiastolicSensorData
│   ├── BPSystolicSensorData
│   ├── HeartRateSensorData (1)
│   └── SpO2SensorData (1)
└── swrla:Entity
    └── swrla:RuleGroup
```
Figure 10 - Ontology Class Hierarchy

Figure 10 shows all of the different classes, in hierarchical form, used to model the knowledge that is gathered by the MORF system. Using properties we can form relationships between classes or instances of classes. This type of property is called an object property. Datatype properties, on the other hand, are used to make an association between a class instance and a given set of data. This data can be of the type integer, float, string, date and more.

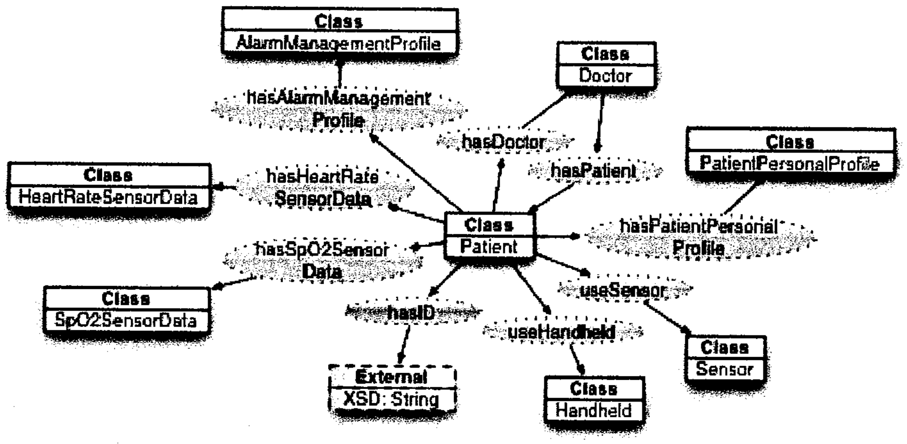


Figure 11 - Patient Relationship Structure

Figure 11 shows the basic property structure around the Patient class. Classes are coloured in yellow, object properties in blue, datatype properties in green and data values in pink. These classes are the major building blocks of our ontology. Of all of these classes, the three most important are the "PatientPersonalProfile" class, the "SensorData" classes and the "AlarmManagementProfile" class. For each patient connected to the system, there is a separate instance for each of these classes. It is the instances of theses classes that we will discuss in further detail below.

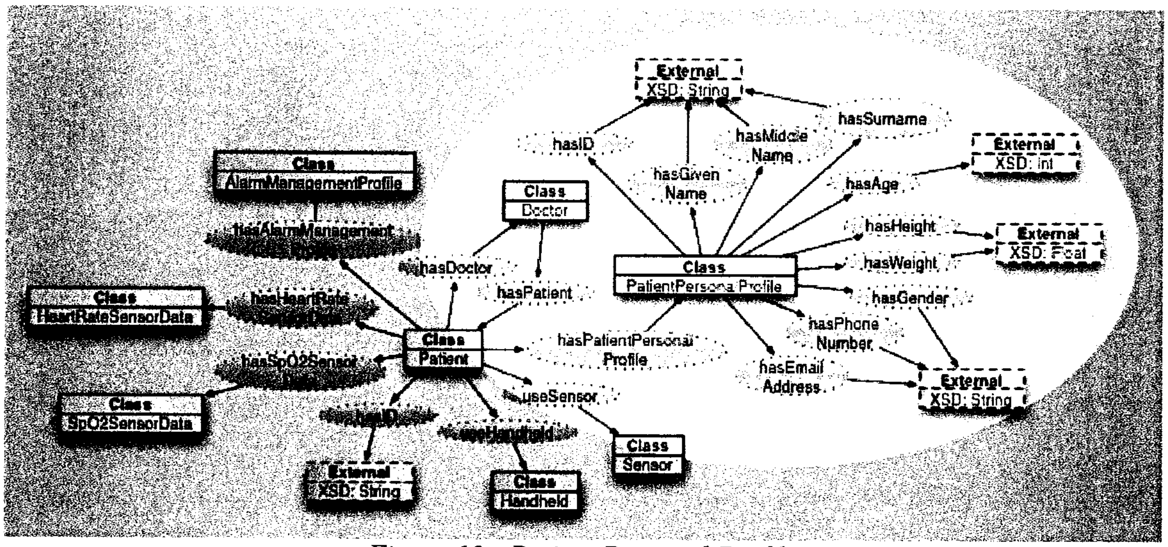


Figure 12 - Patient Personal Profile

Figure 12 is an instance of the "PatientPersonalProfile" class. This class contains all of the personal information about a specific patient. Any time the system requires this information the ontology must be queried in order to retrieve it. As mentioned in the previous chapter, this information was once stored in the database. We decided to use the ontology to store this information instead of the database since all of the other patient-related information, aside from sensor data logs, were going to be stored here as well. In terms of performance, there appears to be no noticeable difference in access time between the two methods.

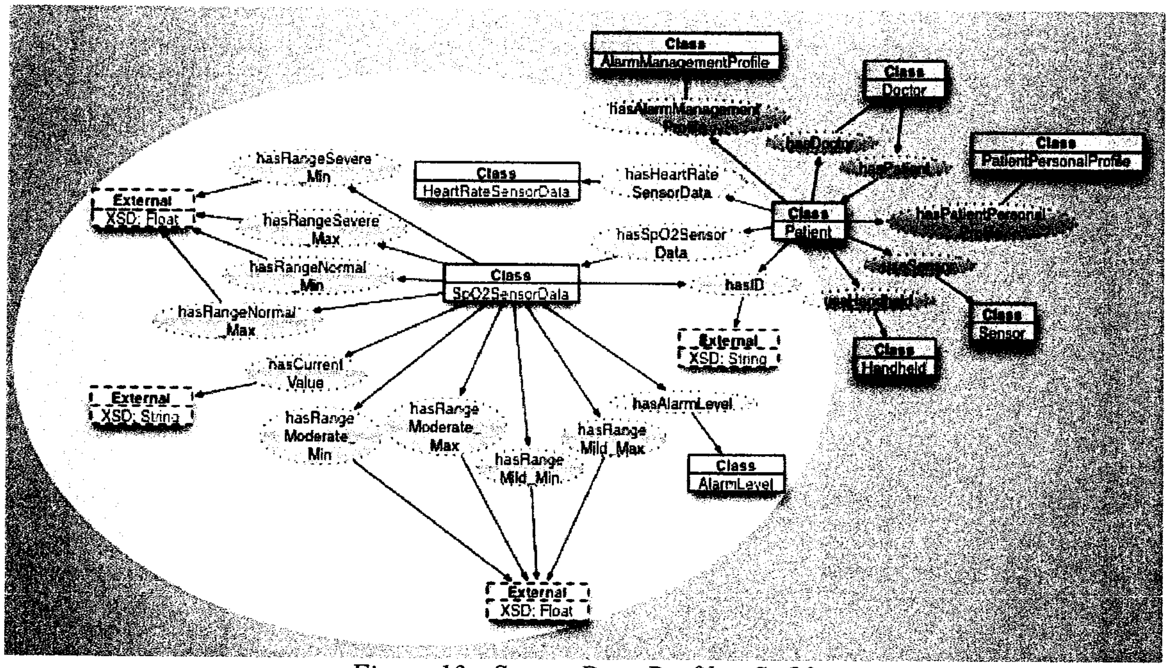


Figure 13 - Sensor Data Profile - SpO2

Figure 13 shows and instance of the "SensorData" class and, in this case, an instance of the subclass "SpO2SensorData". As can be seen from this figure, there are four alert ranges that can be modified by the patient's medical professional to accommodate for that patient's specific needs. These ranges are normal, mild, moderate and severe. In the case of SpO2, they are relative to normal oxygen-saturation levels as well as mild, moderate and severe hypoxemia respectively.

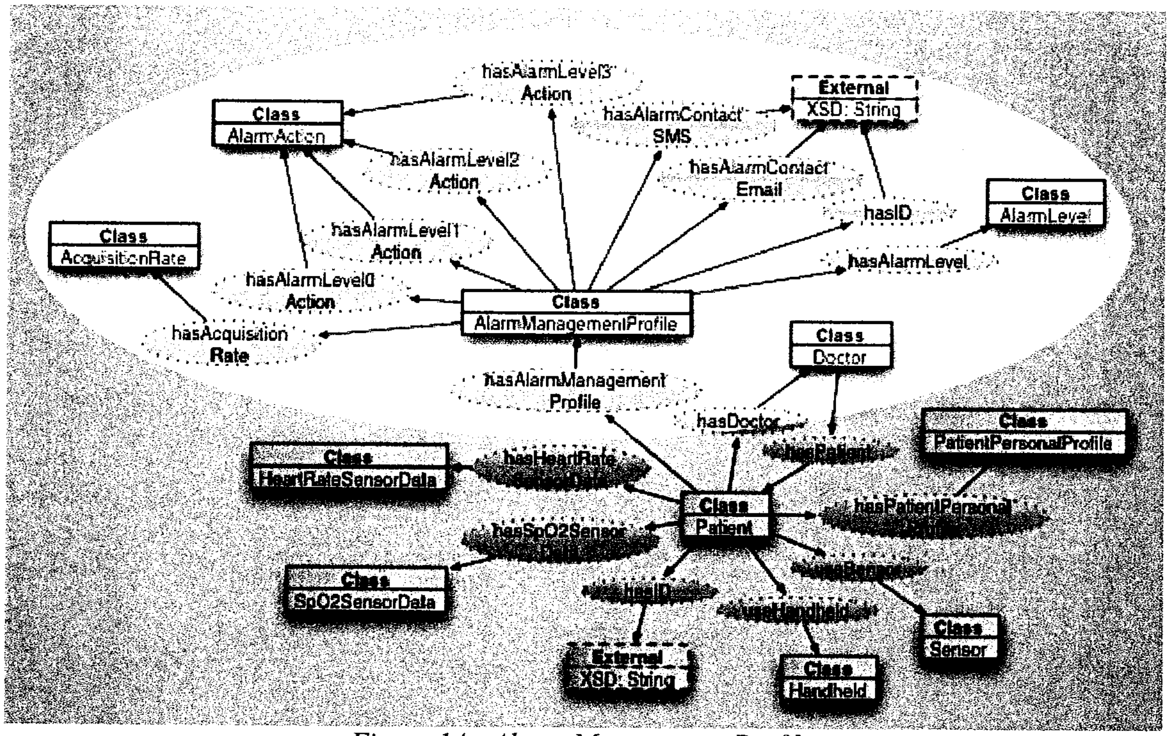


Figure 14 - Alarm Management Profile

Also of significant importance, is the "hasCurrentValue" property. This property contains the most recently received sensor data value from the mobile unit. It is this value that is compared against the aforementioned alert ranges by a set of rules to determine the alarm level for the specific sensor. Once the alarm level is inferred by the rules, it is stored in the "hasAlarmLevel" property. The alarm levels and alert ranges are, therefore, directly related in that a normal range corresponds to an alarm level of 0 and a severe range corresponds to an alarm level of 3.

Perhaps of greatest importance are the alarm management profiles (shown in Figure 14). Instances of this class contain the overall patient alarm level ("hasAlarmLevel" property), alarm action policies ("hasAlarmLevel#Action" properties) and notification contact information ("hasAlarmContactEmail" and "hasAlarmContactSMS"). The overall patient alarm level is determined by the alarm level of the individual sensors. No matter how many sensors the patient is using, the overall alarm level is equivalent to the highest sensory alarm level. This method works well because it allows us to assign a sort of value system to each of the sensors by only letting it trigger a certain number of alarms. Take a heart rate measurement for example, which, medically, has one of two states: normal or abnormal. When a patient's heart rate is normal the heart rate alarm level is 0 but when it is abnormal, the alarm level is 1. This means that no matter what, if the patient's heart rate is abnormal the overall alarm level for the patient is set to 1. Since a heart rate can be abnormal for many reasons, some of which can be good and some of which can be bad, the heart rate alarm is not as important as, say, another sensor alarm and is therefore limited to sounding an alarm level of 1 and nothing higher.

The alarm action properties are perhaps the most important aspect of this particular instance. These properties affect what actions are performed based on each alarm level. There are four distinct actions that can be taken and assigned in whatever combination is desired to each alarm level.

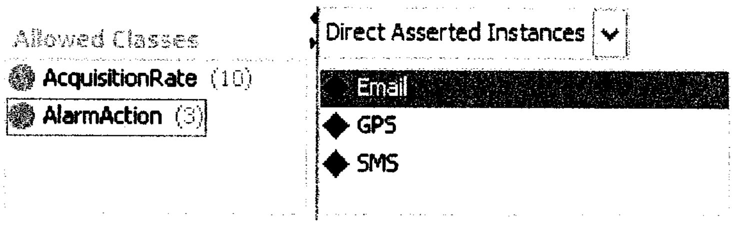
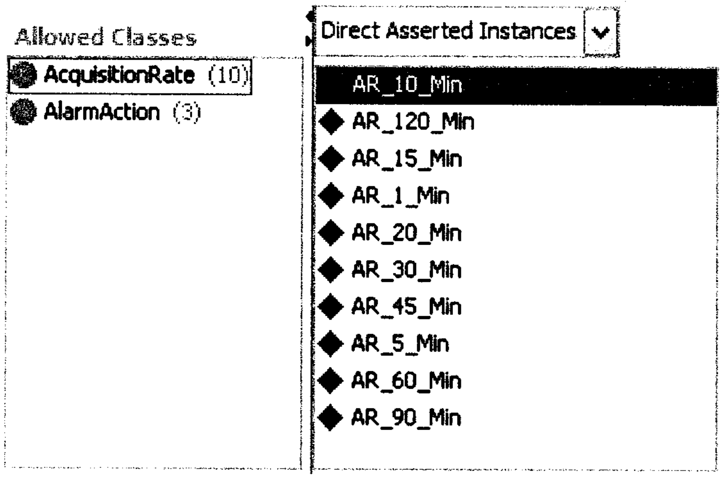


Figure 15 - Alarm Actions

These four actions consist of: changing acquisition rate, getting GPS data, sending a SMS text message to the emergency SMS contact number and sending an email to the emergency contact address. When it comes to acquisition rate, there are a span of 10 distinct rates, from as little as 1 minute to as much as 120 minutes, one of which can be chosen for each alarm level. The acquisition rate is actually more of a time delay or interval between consecutive sensor measurements. Its purpose is that of taking more frequent measurements if a patient's health status is in question.



AR_10_Min
AR_120_Min
AR_15_Min
AR_1_Min
AR_20_Min
AR_30_Min
AR_45_Min
AR_5_Min
AR_60_Min
AR_90_Min

Figure 16 - Acquisition Rates

If a patient is in possible danger it may be necessary to locate and treat him. This is where GPS coordinates come in. Any alarm level can be assigned to gather the patient's coordinates but it is likely only to be necessary in a state of emergency or a state of strong likelihood of emergency. If the GPS action is taken, the GPS coordinates are gathered by the mobile unit and transmitted back to the server. Once the coordinates have been received they can then be sent out in an email or SMS to appropriate personnel in order to take action or simply just logged away in the database. Since ambulance services aren't dispatched to a specific GPS coordinate but rather to an address, a future version of this system would implement a form of mapping system to translate the GPS coordinates into a street address. The system could also include a form of environment similar to Google Maps to display a map of the area to help with identifying the location of the patient.

Upon further testing of our system with the use of ontologies, it was discovered that they have limited support for multiple-user editing. Seidenberg and Rector, in their paper "A Method for Asynchronous Multi-User Editing of Semantic Web Ontologies", expand upon the thought of how current methods of editing ontologies are inadequate due to a lack of multi-user support [29]. The method they suggest to overcome this downfall, involves a "compromise between a very restrictive approach that might offer complete error protection but make useful multi-user interactions impossible and a wide-open anything-goes editing paradigm which offers little to no protection" [29]. Not having an effective way of performing multi-user editing of our ontology is a potential problem for the MORF system. Any time a patient's mobile monitor relays sensor data to the system, this data, and any other inferred knowledge, must be written to the ontology. Having multiple patients logged into the system at the same time could present problems if they all attempt to write to the ontology at the same time. On top of that, any time a medical professional wishes to modify a patient's alarm policies, the ontology must be accessed and written to. In order to avoid the issue with multiple patients, it is possible to have multiple ontologies, on for each patient using the system. This will greatly reduce the issues surrounding multiple-user editing of our ontology but in the future a more permanent and effective approach, such as the one mentioned above, should be implemented.

4.2 – Reasoning & Alarm Management

While knowledge embedded in the ontology is modelled and represented using OWL, ontology-based reasoning is performed by a set of user-defined rules. User-defined reasoning uses SWRL to make various inferences from the ontology.

Automating the reasoning process in order to reliably analyze and reason with the sensor data is much more time-efficient than using a human to do such a task. The reasoning engine processes the received sensor data, cross-referencing it against a profile created by a doctor for each individual patient. This profile includes data such a boundaries and limitations for each individual sensor the patient may be wearing as well as different alarm levels based on how far a vital sign may stray from those boundaries. Within each alarm level the doctor assigns certain services to be carried out such as notifications to doctors or other medical personnel, more frequent measurements for closer monitoring and even Emergency Medical Service (EMS) alarm, where the GPS co-ordinates of the patient are sent to an EMS team, or anyone else for that matter, so that they can find and treat the patient in the case of an emergency. These profiles are written in OWL and the processing of these profiles will be performed using SWRL [27] [30].

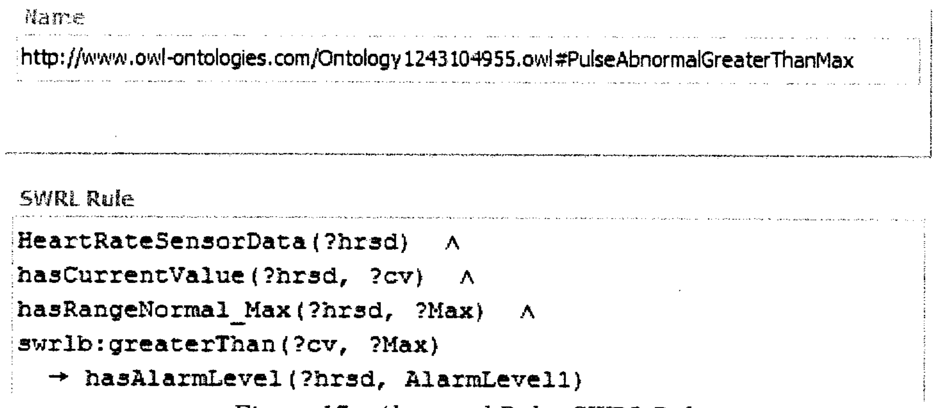
Using a reasoning process to infer new knowledge into the ontology gives it the ability to mature as new inferences are made. Every time that new sensor data is received and processed by the system, the ontology is updated to take this new knowledge into account. From this new knowledge, new inferences are made; this is the concept behind ontology maturation. As more knowledge is used in making inferences, such as previous sensor data as opposed to only the current sensor data, this maturing process becomes more involved because more detailed inferences are made.

During development, SWRL turned out to be a very problematic language to use for carrying out actions. It has no support for sequential inferences, which means that when an inference is made all rules are executed for all objects, to which the rules apply. This creates problems and limits its use severely. There are methods of working around these issues, but the time and effort it would take to do so makes such an approach undesirable. In light of this fact we decided to only use SWRL for a limited number of rules. The rules that we have chosen to use SWRL for pertain to the assignment of the current alarm level to each of the sensor values. Even this seems to be a bit of overkill, since, when an inference is made all alarm levels are recalculated as opposed to the alarm levels specific to an individual patient. In the future, it is our goal to move away from SWRL to a more mature rule language or to simply move away from use of rule languages altogether and use an object-oriented programming language instead. Table 9 shows the SWRL rules used to calculate the sensor-specific alarm levels.

Rule Name	Rule Description
PulseAbnormalGreaterThanMax	HeartRateSensorData(?hrsd) hasCurrentValue(?hrsd, ?cv) hasRangeNormal_Max(?hrsd, ?Max) swrlb:greaterThan(?cv, ?Max) hasAlarmLevel(?hrsd, AlarmLevel1)
PulseAbnormalLessThanMax	HeartRateSensorData(?hrsd) hasCurrentValue(?hrsd, ?cv) hasRangeNormal_Min(?hrsd, ?Min) swrlb:lessThan(?cv, ?Min) hasAlarmLevel(?hrsd, AlarmLevel1)
PulseNormal	HeartRateSensorData(?hrsd) hasCurrentValue(?hrsd, ?cv) hasRangeNormal_Min(?hrsd, ?Min) hasRangeNormal_Max(?hrsd, ?Max) swrlb:greaterThanOrEqual(?cv, ?Min) swrlb:lessThanOrEqual(?cv, ?Max) hasAlarmLevel(?hrsd, AlarmLevel0)
SpO2Mild	SpO2SensorData(?spo2) hasCurrentValue(?spo2, ?cv) hasRangeMild_Min(?spo2, ?Min) hasRangeMild_Max(?spo2, ?Max) swrlb:greaterThanOrEqual(?cv, ?Min) swrlb:lessThanOrEqual(?cv, ?Max) hasAlarmLevel(?spo2, AlarmLevel1)
SpO2Moderate	SpO2SensorData(?spo2) hasCurrentValue(?spo2, ?cv) hasRangeModerate_Min(?spo2, ?Min) hasRangeModerate_Max(?spo2, ?Max) swrlb:greaterThanOrEqual(?cv, ?Min) swrlb:lessThanOrEqual(?cv, ?Max) hasAlarmLevel(?spo2, AlarmLevel2)
SpO2Normal	SpO2SensorData(?spo2) hasCurrentValue(?spo2, ?cv) hasRangeNormal_Min(?spo2, ?Min) hasRangeNormal_Max(?spo2, ?Max) swrlb:greaterThanOrEqual(?cv, ?Min) swrlb:lessThanOrEqual(?cv, ?Max) hasAlarmLevel(?spo2, AlarmLevel0)
SpO2Severe	SpO2SensorData(?spo2) hasCurrentValue(?spo2, ?cv) hasRangeSevere_Max(?spo2, ?Max) swrlb:lessThanOrEqual(?cv, ?Max) hasAlarmLevel(?spo2, AlarmLevel3)

Table 9 - SWRL Rules for MORF Alarm Management

Since, for testing purposes, we are only using the Nonin 4100 sensor, the only alarm levels that are being calculated are those of heart rate and oxygen saturation. Rules that govern abnormal heart rate, mild hypoxemia and sever hypoxemia will be discussed in further detail below.


Name

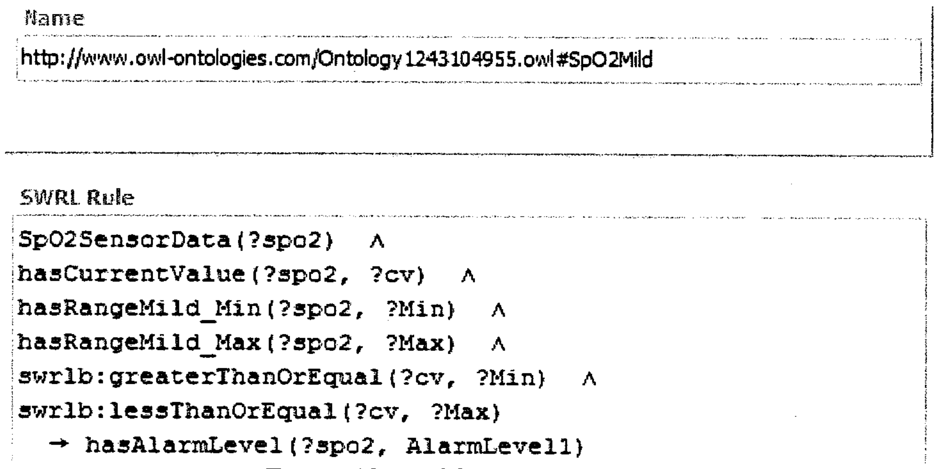
http://www.owl-ontologies.com/Ontology1243104955.owl#PulseAbnormalGreaterThanMax

SWRL Rule

```
HeartRateSensorData(?hrsd)  ∧
hasCurrentValue(?hrsd, ?cv)  ∧
hasRangeNormal_Max(?hrsd, ?Max)  ∧
swrlb:greaterThan(?cv, ?Max)
  → hasAlarmLevel(?hrsd, AlarmLevel1)
```

Figure 17 - Abnormal Pulse SWRL Rule

The rule above governs the assignment of the alarm level in the case of an abnormally high heart rate. A normal heart rate is specified by the "hasRangeNormal_Min" and "hasRangeNormal_Max" properties as defined in each instance of the "HeartRateSensorData" class. If a patient's heart rate is greater than the specified normal maximum, then the heart rate alarm level for that patient is set to alarm level 1.


Name

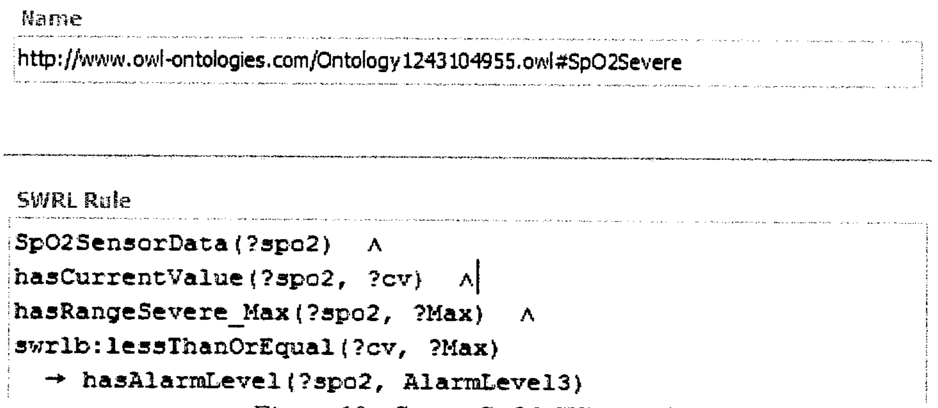
http://www.owl-ontologies.com/Ontology1243104955.owl#SpO2Mild

SWRL Rule

```
SpO2SensorData(?spo2)  ∧
hasCurrentValue(?spo2, ?cv)  ∧
hasRangeMild_Min(?spo2, ?Min)  ∧
hasRangeMild_Max(?spo2, ?Max)  ∧
swrlb:greaterThanOrEqual(?cv, ?Min)  ∧
swrlb:lessThanOrEqual(?cv, ?Max)
  → hasAlarmLevel(?spo2, AlarmLevel1)
```

Figure 18 - Mild SpO2 SWRL Rule

Mild hypoxemia is usually defined as having an oxygen saturation from 90-94%. This rule sets the SpO2 alarm level to 1 if the patient's current SpO2 levels are in this range. As with all sensor parameters, these values can be changed on a per-patient basis since each patient will have slightly different norms.



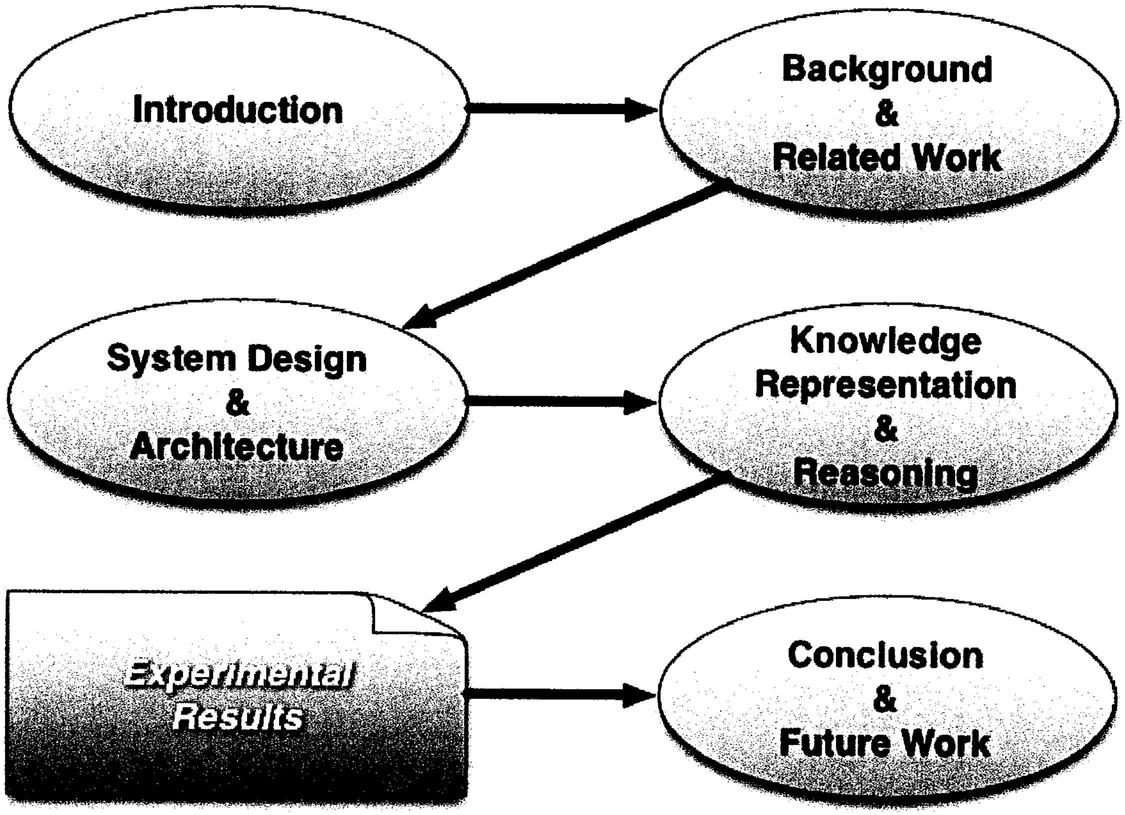
http://www.owl-ontologies.com/Ontology1243104955.owl#SpO2Severe

```
SpO2SensorData(?spo2)  ∧
hasCurrentValue(?spo2, ?cv)  ∧
hasRangeSevere_Max(?spo2, ?Max)  ∧
swrlb:lessThanOrEqual(?cv, ?Max)
  → hasAlarmLevel(?spo2, AlarmLevel3)
```

Figure 19 - Severe SpO2 SWRL Rule

The above rule sets the SpO2 alarm level to 3 when a patient has a current oxygen saturation reading of less than that which is specified in the "hasRangeSevere_Max" property in the "SpO2SensorData" class. This rule represents a case of severe hypoxemia which has an equivalent SpO2 value of less than 70%.

chapter 5


Introduction
Background & Related Work
System Design & Architecture
Knowledge Representation & Reasoning
Experimental Results
Conclusion & Future Work

experimental results

This chapter covers the experimental results gathered while researching, designing, building and testing the MORF health monitoring system. We will begin by discussing the user interface of the system followed by examining a few test scenarios to see the system in operation. Finally, we will evaluate the performance of the system, analyzing its advantages and disadvantages.

5.1 – User Interface

This section describes the user interface of the MORF system. There are three separate interfaces that will be discussed here: the mobile unit's interface, the standard web interface and the mobile web interface. The mobile unit interface is the main interface that controls the operation of the mobile unit, where the standard and mobile web interfaces have been designed to provide control over the entire operation of the MORF system on a per-patient basis.

5.1.1 – Mobile Application Interface

The application that has been developed for the mobile phone is a Java MIDlet that controls the entire operation of the mobile unit. The interface for this application is currently being used for diagnostic purposes only, much like a console in a computer system. It outputs debugging information to the screen to ensure the proper operation of the mobile unit's software program as shown in Figure 20.

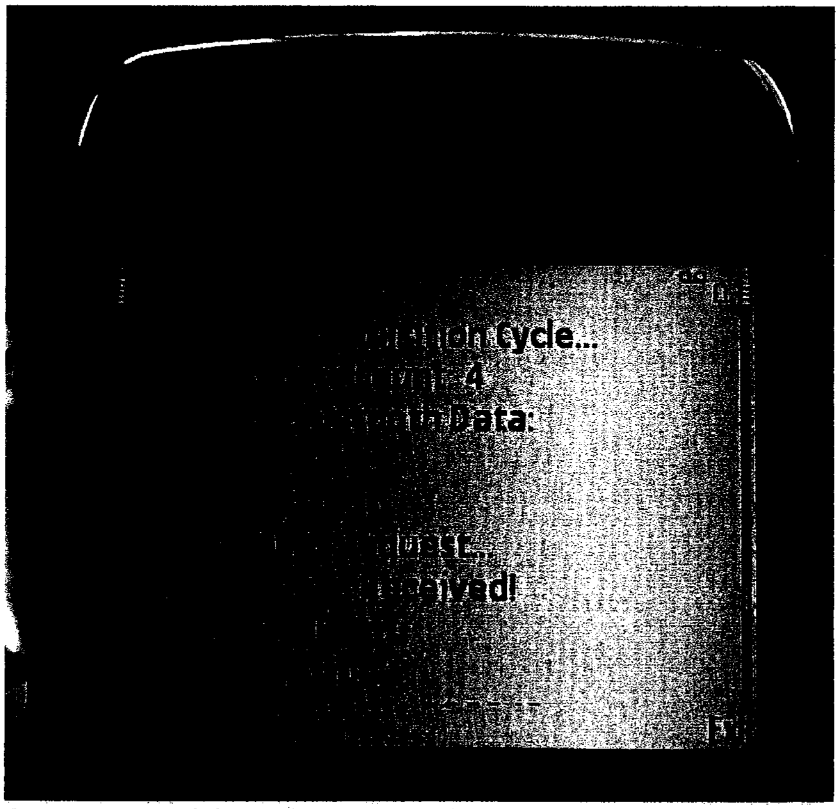
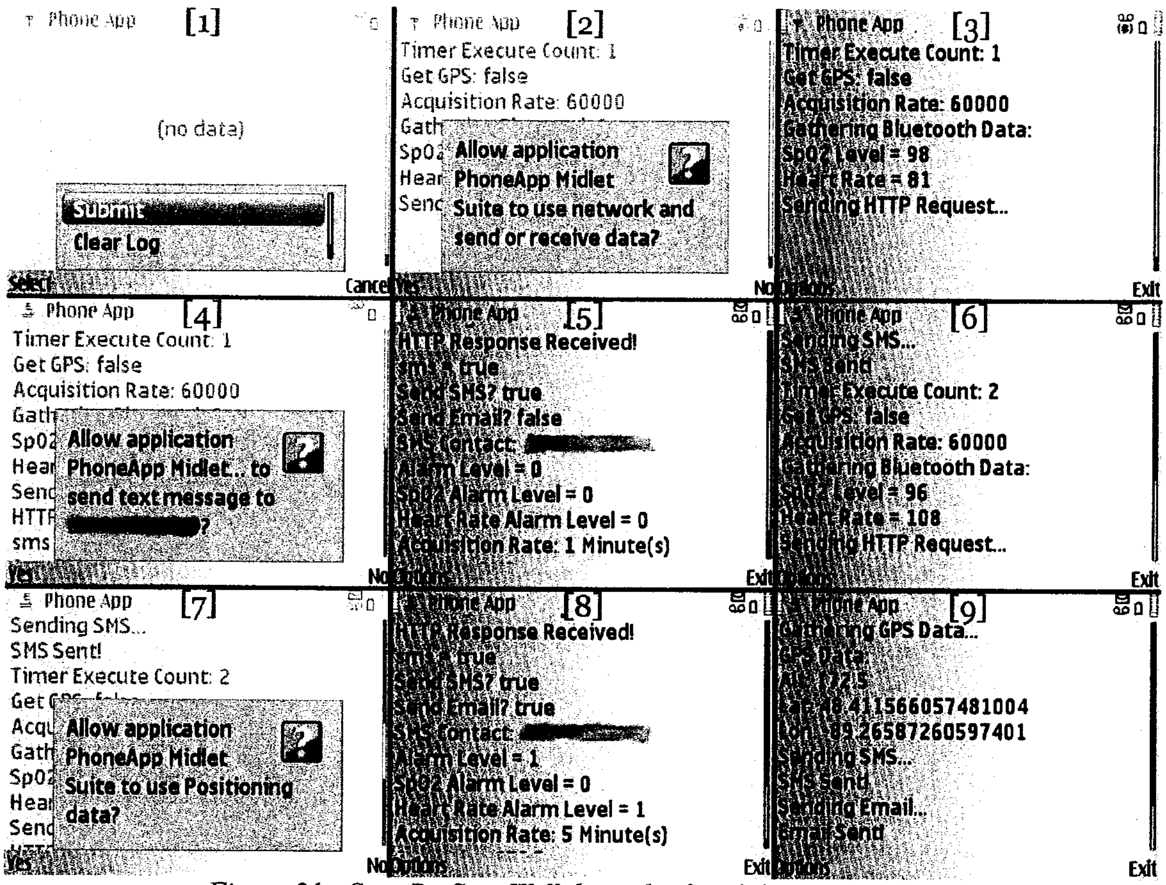


Figure 20 - Mobile Application Interface Operating on the Nokia E71

Since the mobile unit is to be used by the patient, the ideal is to minimize the need for a user interface so that the system is simple to operate and requires minimal to no input from the user. Figure 21 is a collage of a series of nine screenshots taken on our test phone during operation. The Java MIDlet "ScreenSnap" was used to take these pictures and can be found here [31]. These screenshots cover two full data acquisition cycles, one resulting in an alarm level of 0 and the other resulting in an alarm level of 1.



```
Phone App [1]
(no data)
Submit
Clear Log
Select                    Cancel
```

```
Phone App [2]
Timer Execute Count: 1
Get GPS: false
Acquisition Rate: 60000
Gath
SpO2  Allow application
Hear  PhoneApp Midlet
Senc  Suite to use network and
      send or receive data?
Yes                       No
```

```
Phone App [3]
Timer Execute Count: 1
Get GPS: false
Acquisition Rate: 60000
Gathering Bluetooth Data:
SpO2 Level = 98
Heart Rate = 81
Sending HTTP Request...
Options                   Exit
```

```
Phone App [4]
Timer Execute Count: 1
Get GPS: false
Acquisition Rate: 60000
Gath
SpO2  Allow application
Hear  PhoneApp Midlet... to
Senc  send text message to
HTTP  [redacted]?
sms
Yes                       No
```

```
Phone App [5]
HTTP Response Received!
sms = true
Send SMS? true
Send Email? false
SMS Contact: [redacted]
Alarm Level = 0
SpO2 Alarm Level = 0
Heart Rate Alarm Level = 0
Acquisition Rate: 1 Minute(s)
Options                   Exit
```

```
Phone App [6]
Sending SMS...
SMS Sent!
Timer Execute Count: 2
Get GPS: false
Acquisition Rate: 60000
Gathering Bluetooth Data:
SpO2 Level = 96
Heart Rate = 108
Sending HTTP Request...
Options                   Exit
```

```
Phone App [7]
Sending SMS...
SMS Sent!
Timer Execute Count: 2
Get G
Acqu  Allow application
Gath  PhoneApp Midlet
SpO2  Suite to use Positioning
Hear  data?
Senc
Yes                       No
```

```
Phone App [8]
HTTP Response Received!
sms = true
Send SMS? true
Send Email? true
SMS Contact: [redacted]
Alarm Level = 1
SpO2 Alarm Level = 0
Heart Rate Alarm Level = 1
Acquisition Rate: 5 Minute(s)
Options                   Exit
```

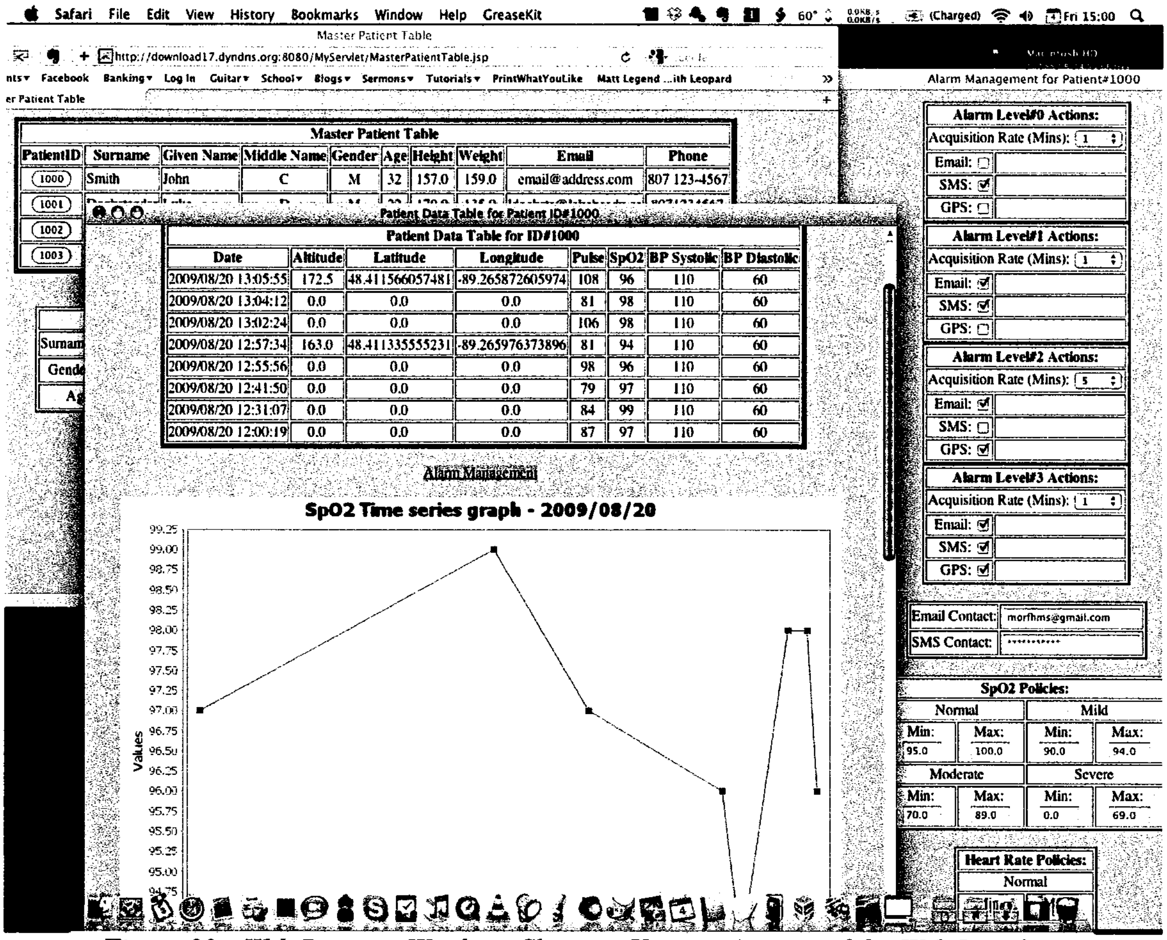
```
Phone App [9]
Gathering GPS Data...
GPS Data:
Alt: 172.5
Lat: 48.411566057481004
Lon: -89.2658726059740
Sending SMS...
SMS Sent!
Sending Email...
Email Sent!
Options                   Exit
```

Figure 21 - Step-By-Step Walkthrough of Mobile User Interface

Frame 1 shows the phone application's menu screen. The "Submit" option begins the data acquisition cycle. This is how patient monitoring is initiated. The "Clear Log" option is used to clear the screen of any console messages. Frame 2, 4 and 7 are confirmation alerts to permit or deny the application from using the network connection, sending an SMS or using the GPS module to gather GPS coordinates. These alerts are build into the Nokia Symbian OS and can't be removed without signing the MIDlet with a third party certificate from Verisign or Thawte. Luckily, they can be set to only ask once during the operation of the MIDlet with the exception of the SMS alert which, by the MIDP 2.0 standard, must always ask permission. In the future, to avoid this issue, SMS messages will be sent from the server using a GSM modem. This would remove the need of using the phone for sending out alert messages and also reduces the likelihood of a communications error preventing the system from notifying emergency personnel. Frame 3 was taken after the phone requested and received the sensor data from the Nonin 4100 pulse oximeter. This data is printed to the screen and then transmitted to the server in the form of a simple HTTP request. Once the data received by the server is finished being processed, a set of commands are sent back to the mobile unit via the HTTP response. In this case, one of the actions to be taken is that of sending an SMS to the patient's emergency contact SMS number. Frame 5 shows all of the feedback parameters sent back to the phone from the server. Data such as the overall and sensor-specific alarm levels are listed here as well as the newly acquired acquisition rate. Frame 6 shows the application executing a second data acquisition cycle and presents the new sensor data received from the sensors. Since the patient's heart rate was slightly elevated, an alarm level of 1 was reached and GPS coordinates were requested. Frame 8 lists the feedback parameters, as sent by the system in response to the alarm level being changed to 1. A SMS message and an email are both sent out, containing all of the relevant sensor data and the newly acquired GPS coordinates. Frame 9 shows the acquired GPS data and confirms that a message was sent to the appropriate contact address.

5.1.2 – Web Interface

In order for a medical professional to interact with the MORF system there needs to be an interface through which he can operate. Since the entire system is based heavily on Web standards, it followed logically to create a Web-based interface to enable this sort of interaction. The Web-interface was designed using a combination of Web technologies such as HTML, Javascript, CSS, and JSP. The first three are being used to mould the aesthetics and general functionality of the interface where the use of JSP is mainly geared towards the interaction of the Web-interface with the server-side operation of the system. It is used to fetch data from both the ontology and the database as well as governing browser sessions and properties.


Safari File Edit View History Bookmarks Window Help GreaseKit
Master Patient Table
http://download17.dyndns.org:8080/MyServlet/MasterPatientTable.jsp
Facebook Banking Log In Guitar School Blogs Sermons Tutorials PrintWhatYouLike Matt Legend ...ith Leopard
Fri 15:00
Alarm Management
SpO2 Time series graph - 2009/08/20
Values
99.25
99.00
98.75
98.50
98.25
98.00
97.75
97.50
97.25
97.00
96.75
96.50
96.25
96.00
95.75
95.50
95.25
95.00
94.75

Master Patient Table									
PatientID	Surname	Given Name	Middle Name	Gender	Age	Height	Weight	Email	Phone
1000	Smith	John	C	M	32	157.0	159.0	email@address.com	807 123-4567
1001									
1002									
1003									

Patient Data Table for Patient ID#1000

Patient Data Table for ID#1000							
Date	Altitude	Latitude	Longitude	Pulse	SpO2	BP Systolic	BP Diastolic
2009/08/20 13:05:55	172.5	48.411566057481	-89.265872605974	108	96	110	60
2009/08/20 13:04:12	0.0	0.0	0.0	81	98	110	60
2009/08/20 13:02:24	0.0	0.0	0.0	106	98	110	60
2009/08/20 12:57:34	163.0	48.411335555231	-89.265976373896	81	94	110	60
2009/08/20 12:55:56	0.0	0.0	0.0	98	96	110	60
2009/08/20 12:41:50	0.0	0.0	0.0	79	97	110	60
2009/08/20 12:31:07	0.0	0.0	0.0	84	99	110	60
2009/08/20 12:00:19	0.0	0.0	0.0	87	97	110	60

Alarm Management for Patient#1000

Alarm Level#0 Actions:
Acquisition Rate (Mins): 1
Email: ☐
SMS: ✓
GPS: ☐

Alarm Level#1 Actions:
Acquisition Rate (Mins): 1
Email: ✓
SMS: ✓
GPS: ☐

Alarm Level#2 Actions:
Acquisition Rate (Mins): 5
Email: ✓
SMS: ☐
GPS: ✓

Alarm Level#3 Actions:
Acquisition Rate (Mins): 1
Email: ✓
SMS: ✓
GPS: ✓

Email Contact: morfhms@gmail.com
SMS Contact: ***********

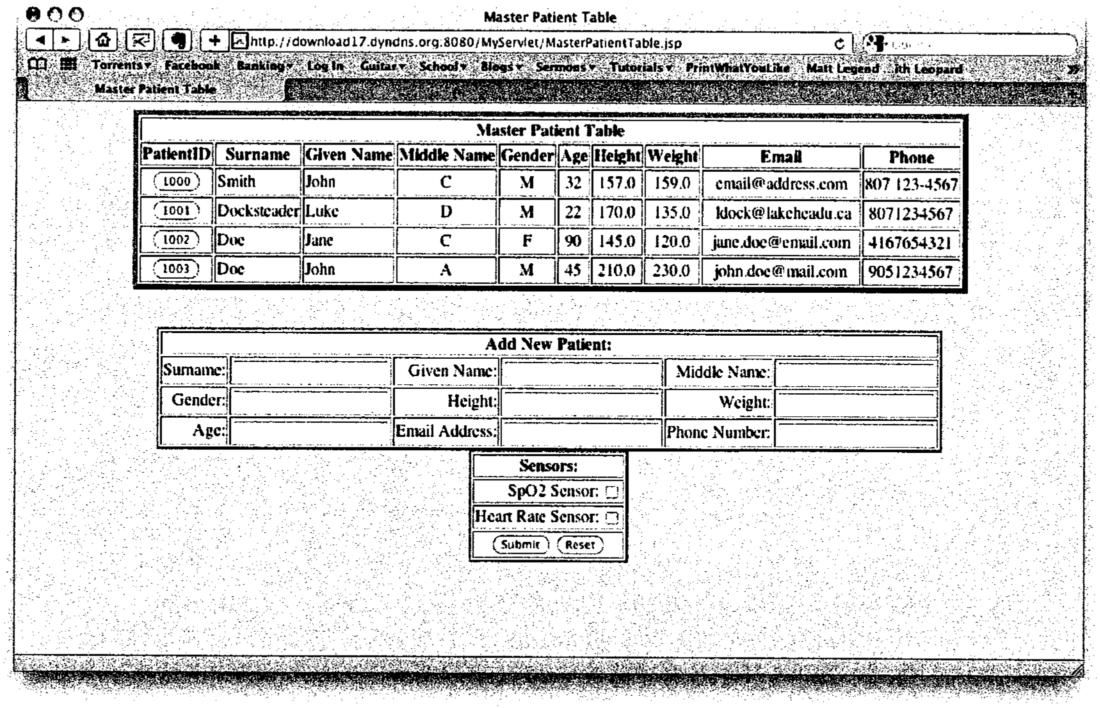
SpO2 Policies:			
Normal		Mild	
Min: 95.0	Max: 100.0	Min: 90.0	Max: 94.0
Moderate		Severe	
Min: 70.0	Max: 89.0	Min: 0.0	Max: 69.0

Heart Rate Policies:
Normal

Figure 22 - Web Browser Windows Showing Various Aspects of the Web Interface

This portion of the user interface consists of a user management/creation section, user data monitoring section and alarm policy management section. These three sections, or pages, are interfaces into the back-end of the server operations giving the medical professional full control of how the system functions around any given patient.

The user management and creation page is the main page for the Web interface. It consists of a table that displays each patient registered in MORF system. Below the master patient table is the patient registration form. This is a simple form that requires general patient information to be input into the system for record-keeping purposes. When a new patient is registered in the system new tables are created in the database and instances are created in the ontology as necessary.



Master Patient Table

PatientID	Surname	Given Name	Middle Name	Gender	Age	Height	Weight	Email	Phone
1000	Smith	John	C	M	32	157.0	159.0	email@address.com	807 123-4567
1001	Docksteader	Luke	D	M	22	170.0	135.0	ldock@lakeheadu.ca	8071234567
1002	Doe	Jane	C	F	90	145.0	120.0	jane.doe@email.com	4167654321
1003	Doe	John	A	M	45	210.0	230.0	john.doe@mail.com	9051234567

Add New Patient:

Surname:		Given Name:		Middle Name:	
Gender:		Height:		Weight:	
Age:		Email Address:		Phone Number:	

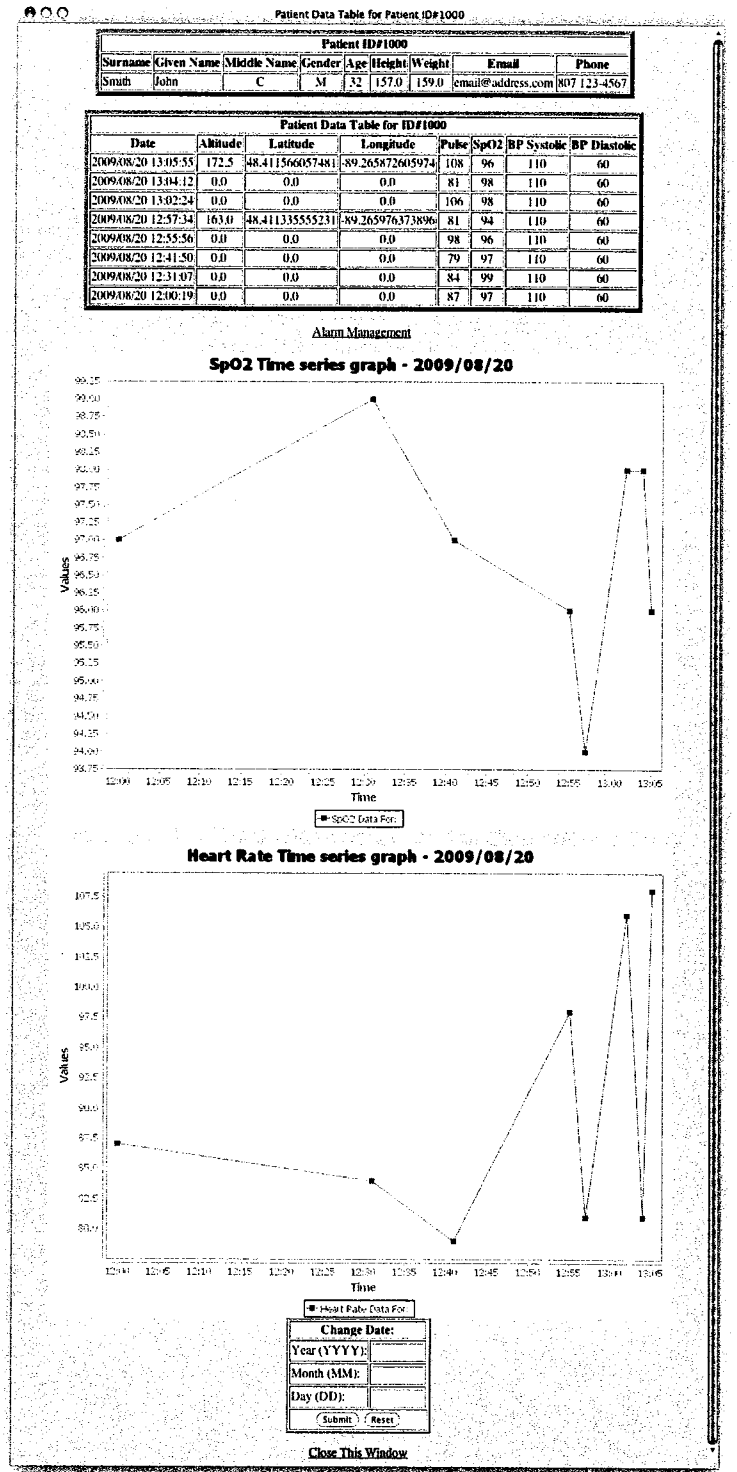
Sensors:

SpO2 Sensor: ☐

Heart Rate Sensor: ☐

Figure 23 - Web UI - Master Patient Table

When a user clicks on the “Patient ID” button next to the patient name he is brought to the user data monitoring page. The purpose of this page is that of providing a means of visually monitoring a patient's vitals throughout the course of a day.


Patient Data Table for Patient ID#1000

Patient ID#1000								
Surname	Given Name	Middle Name	Gender	Age	Height	Weight	Email	Phone
Smith	John	C	M	32	157.0	159.0	email@address.com	807 123-4567

Patient Data Table for ID#1000							
Date	Altitude	Latitude	Longitude	Pulse	SpO2	BP Systolic	BP Diastolic
2009/08/20 13:05:55	172.5	48.411566057481	-89.265872605974	108	96	110	60
2009/08/20 13:04:12	0.0	0.0	0.0	81	98	110	60
2009/08/20 13:02:24	0.0	0.0	0.0	106	98	110	60
2009/08/20 12:57:34	163.0	48.411335555231	-89.265976373896	81	94	110	60
2009/08/20 12:55:56	0.0	0.0	0.0	98	96	110	60
2009/08/20 12:41:50	0.0	0.0	0.0	79	97	110	60
2009/08/20 12:31:07	0.0	0.0	0.0	84	99	110	60
2009/08/20 12:00:19	0.0	0.0	0.0	87	97	110	60

Alarm Management

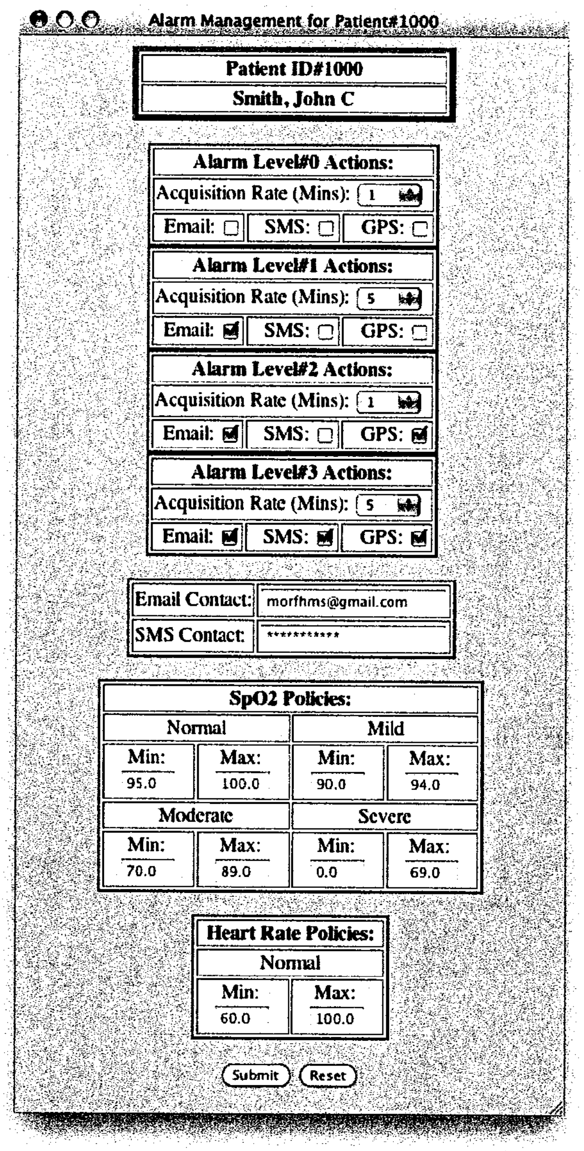
SpO2 Time series graph - 2009/08/20
Values
Time
SpO2 Data For:
Heart Rate Time series graph - 2009/08/20
Values
Time
Heart Rate Data For:

Change Date:	
Year (YYYY):	
Month (MM):	
Day (DD):	

Figure 24 - Web UI - Patient Data and Graphs

This page lists a patient's sensor data in table-format on a per-day basis. The data in the table is also plotted on a line chart to give a graphical representation of the change in a patient's sensor data over a given period of time. At the bottom of the page is a form for changing the date which allows the user to view the patient's past sensor data. This is helpful in that, by splitting the data up by days, it prevents too much data from being displayed on the screen at the same time, which could become very confusing to look at.

Directly above the line charts is a link that leads the user to the "Alarm Management" page. This page contains the settings to control which actions are taken and what acquisition rate is used for each alarm level. The user can also control which contact email address and phone number are used for sending out emails and SMS text messages when an alarm is triggered. Finally, the user can modify the data ranges for each alarm level for each of the specific sensors. This allows a medical professional to further tune the system to the patient on the fly. For example, if, after monitoring a patient for little while, it is discovered that the patient's normal SpO2 range is between 93% and 96%, the SpO2 alarm ranges can be adjusted accordingly to take into account a lower oxygen saturation norm.



Alarm Management for Patient#1000

Patient ID#1000
Smith, John C

Alarm Level#0 Actions:		
Acquisition Rate (Mins): 1		
Email: ☐	SMS: ☐	GPS: ☐

Alarm Level#1 Actions:		
Acquisition Rate (Mins): 5		
Email: ☑	SMS: ☐	GPS: ☐

Alarm Level#2 Actions:		
Acquisition Rate (Mins): 1		
Email: ☑	SMS: ☐	GPS: ☑

Alarm Level#3 Actions:		
Acquisition Rate (Mins): 5		
Email: ☑	SMS: ☑	GPS: ☑

Email Contact:	morfhms@gmail.com
SMS Contact:	***********

SpO2 Policies:			
Normal		Mild	
Min: 95.0	Max: 100.0	Min: 90.0	Max: 94.0
Moderate		Severe	
Min: 70.0	Max: 89.0	Min: 0.0	Max: 69.0

Heart Rate Policies:	
Normal	
Min: 60.0	Max: 100.0

Figure 25 - Web UI - Alarm Management Page

As with the user data monitoring page, the alarm management page has been specifically adapted to function on a mobile device so that a medical professional can interact with the system from outside of the hospital. This concept will be discussed further in section 5.1.3.

One issue we soon discovered about the web interface was that it was rather slow. We then proceeded to perform some latency measurements, using ManageEngine's Application Manager 9, to calculate how much of a delay was present in our system [32].

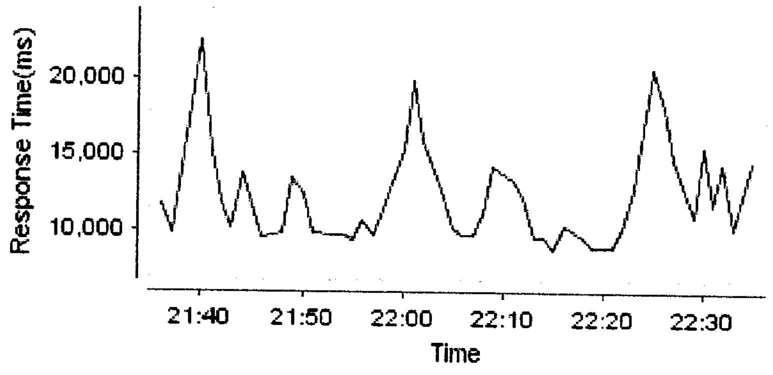


Figure 26 - Master Patient Table Page - Response Times

Figure 26 shows a graphical representation of the response times of the Master Patient Table page (shown in Figure 23). As can be seen clearly above, these response times are in the magnitude of seconds, with a maximum of over 22 seconds to access this particular page on the server. The average response time for this page, as calculated by Application Manager 9 over the period of one hour, was 12,083 ms.

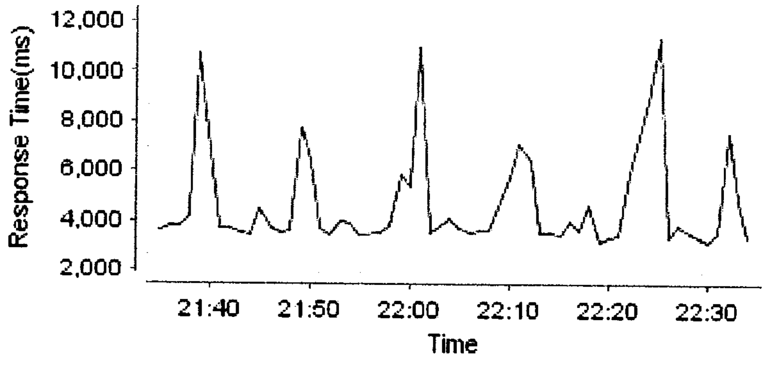


Figure 27 - Patient Data Page - Response Times

Figure 27 shows the response times Patient Data page (shown in Figure 24). As with the previous graph, these response times are still very large, with a maximum of around 11 seconds. The average response time for this page was 4,594 ms.

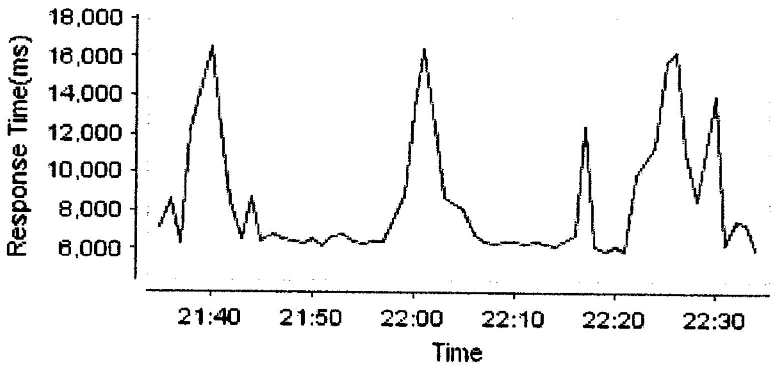


Figure 28 - Alarm Management Profile Page - Response Times

Figure 28 shows the results for the response time for our final test page, the Alarm Management page (shown in Figure 25). As with the previous two test results, the response times are much higher than they should be. This particular page had a maximum of over 16 seconds with an average response time of 8,421 ms.

We soon discovered that these results were caused by the use of the ontologies in our system. When the Tomcat server accesses the data from the ontology using the JENA API, it takes an unusually large amount of time to do so. This has proven to be the bottleneck of our system. The reason for this is the number of times the server needs to read from the ontology in order to display information to the user on the web interface. Currently, any time that a new piece of data is required from the ontology, a query is made to the ontology. For pages such as the Master Patient Table page, a lot of queries are needed to get all of the patient information which, in turn, causes a very large response time. To avoid this, information from the ontology that is not required for making knowledge inferences, such as the patient's personal information, will be moved to a database. Storing the information this way will allow for one request to be made that will gather all of that information at once as opposed to making multiple requests to do so.

As the response time of the user interface is extremely long, the response time of the server, when accessed by the mobile unit, is short. When new data is received by server, new knowledge is inferred by the system and stored in the ontology. As can be seen from the difference between each of the timestamps listed in Table 10 (Page 83), the delay this procedure has on the system is minimal. Comparing this difference with the acquisition rate (time delay) between the cases yields the amount of time the system takes to acquire the new sensor data, transmit it to the server, process it and respond back. This usually takes around 30-40 seconds and is quite acceptable. See section 5.2 for more details.

5.1.3 – Mobile Web Interface

Up until now, we have been explaining the mobile application interface and the standard web interface for the mobile unit. Now we will discuss the mobile web interface. The mobile web interface is that which is used by medical professionals to monitor a patient's vitals on the go. In theory, any mobile phone with web browsing capabilities can access any part of the web interface, but there are a few key parts that are designed specifically for use by a mobile device. These parts are specific pages hosted by the server that, once the server detects are being accessed by a mobile device, modifies the content of the page to fit a smaller screen. The server detects mobile devices by searching the user-agent value in the HTTP header information for specific keywords that indicate use by a mobile device. Determining which device has accessed the server page allows the server to generate content specific to, and specifically formatted for, that device.

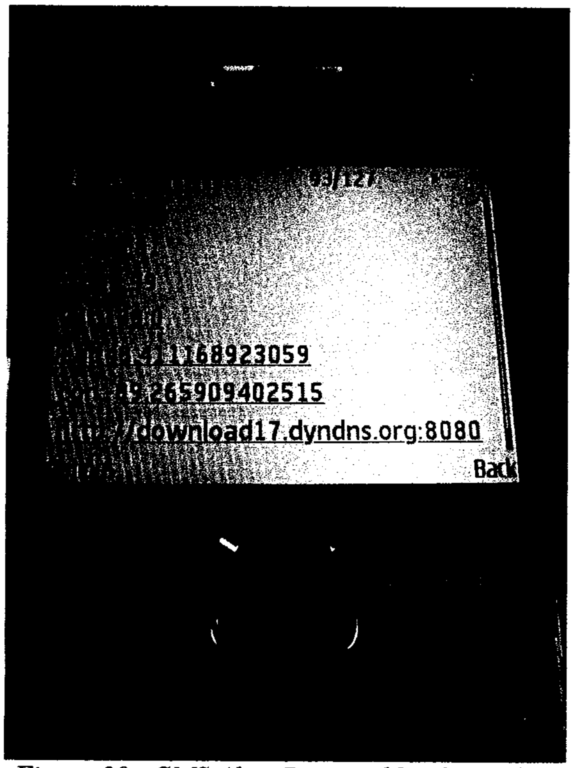
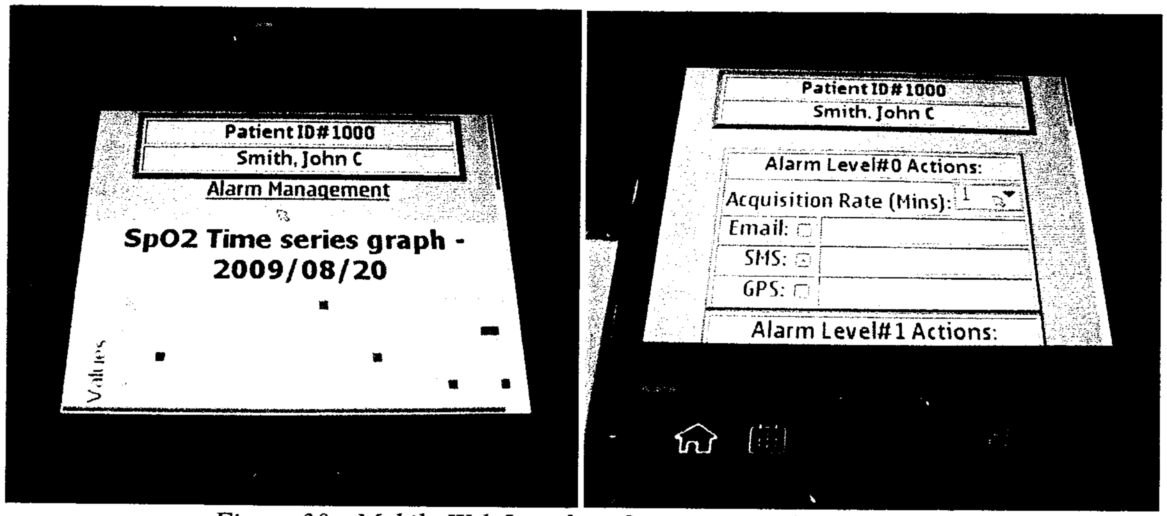


Figure 29 - SMS Alert Received by the Mobile Unit

When an alert SMS, such as the one shown in Figure 29, or email message are sent out, they contain a link to the server page that displays the patient's sensor data in graphical format. This link can be opened and viewed from a mobile device so that, if necessary, a medical professional can view the patient's vital data on the go as shown in Figure 30.


Patient ID#1000

Smith, John C

Alarm Management

SpO2 Time series graph - 2009/08/20

Patient ID#1000

Smith, John C

Alarm Level#0 Actions:

Acquisition Rate (Mins): 1

Email:

SMS:

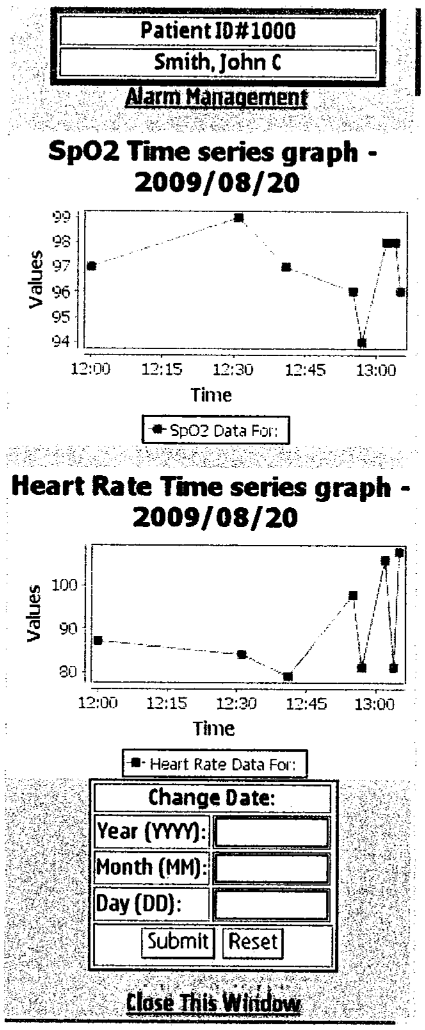
GPS:

Alarm Level#1 Actions:

Figure 30 - Mobile Web Interface Operating on the Nokia E71

Not only can the medical professional view the data, but he can also modify the alarm management protocols and alarm notification policies for that patient, even during run-time. This makes the system flexible and adaptable to the needs of both the patient and his physician while the system is still in operation.

For a more complete view of the operational environment as it pertains to the mobile web interface we have created a collage of screenshots, taken on our test phone by the Screensnap software.



Patient ID#1000
Smith, John C

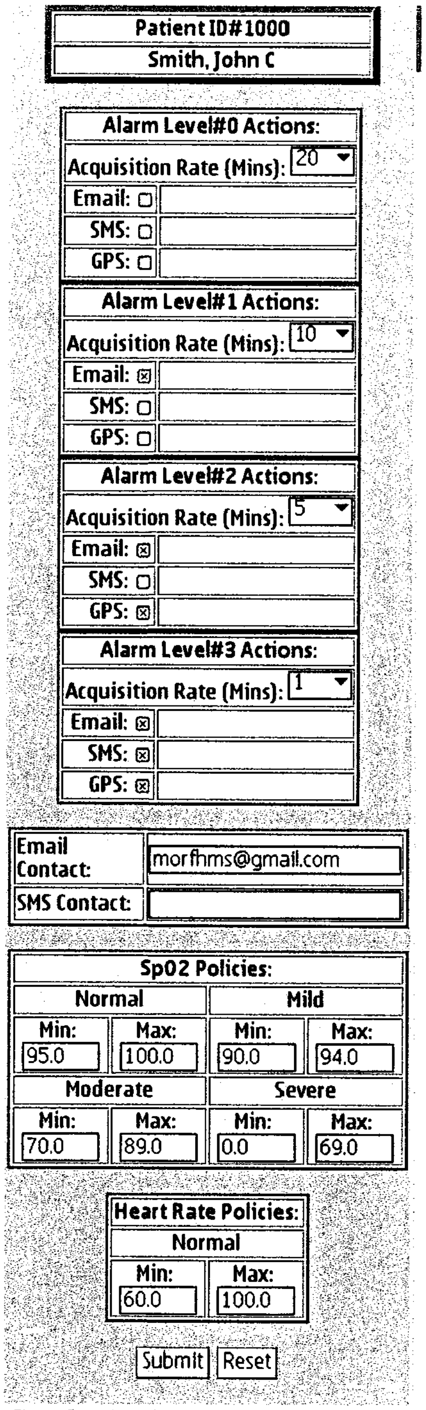
Alarm Management

SpO2 Time series graph - 2009/08/20

Heart Rate Time series graph - 2009/08/20

Change Date:	
Year (YYYY):	
Month (MM):	
Day (DD):	
Submit Reset	

Close This Window



Patient ID#1000
Smith, John C

Alarm Level#0 Actions:	
Acquisition Rate (Mins):	20
Email: ☐	
SMS: ☐	
GPS: ☐	

Alarm Level#1 Actions:	
Acquisition Rate (Mins):	10
Email: ☒	
SMS: ☐	
GPS: ☐	

Alarm Level#2 Actions:	
Acquisition Rate (Mins):	5
Email: ☒	
SMS: ☐	
GPS: ☒	

Alarm Level#3 Actions:	
Acquisition Rate (Mins):	1
Email: ☒	
SMS: ☒	
GPS: ☒	

Email Contact:	morfhms@gmail.com
SMS Contact:	

SpO2 Policies:			
Normal		Mild	
Min:	Max:	Min:	Max:
95.0	100.0	90.0	94.0
Moderate		Severe	
Min:	Max:	Min:	Max:
70.0	89.0	0.0	69.0

Heart Rate Policies:	
Normal	
Min:	Max:
60.0	100.0

Submit Reset

Figure 31 - Mobile Web UI – Sensor Graphs & Alarm Management

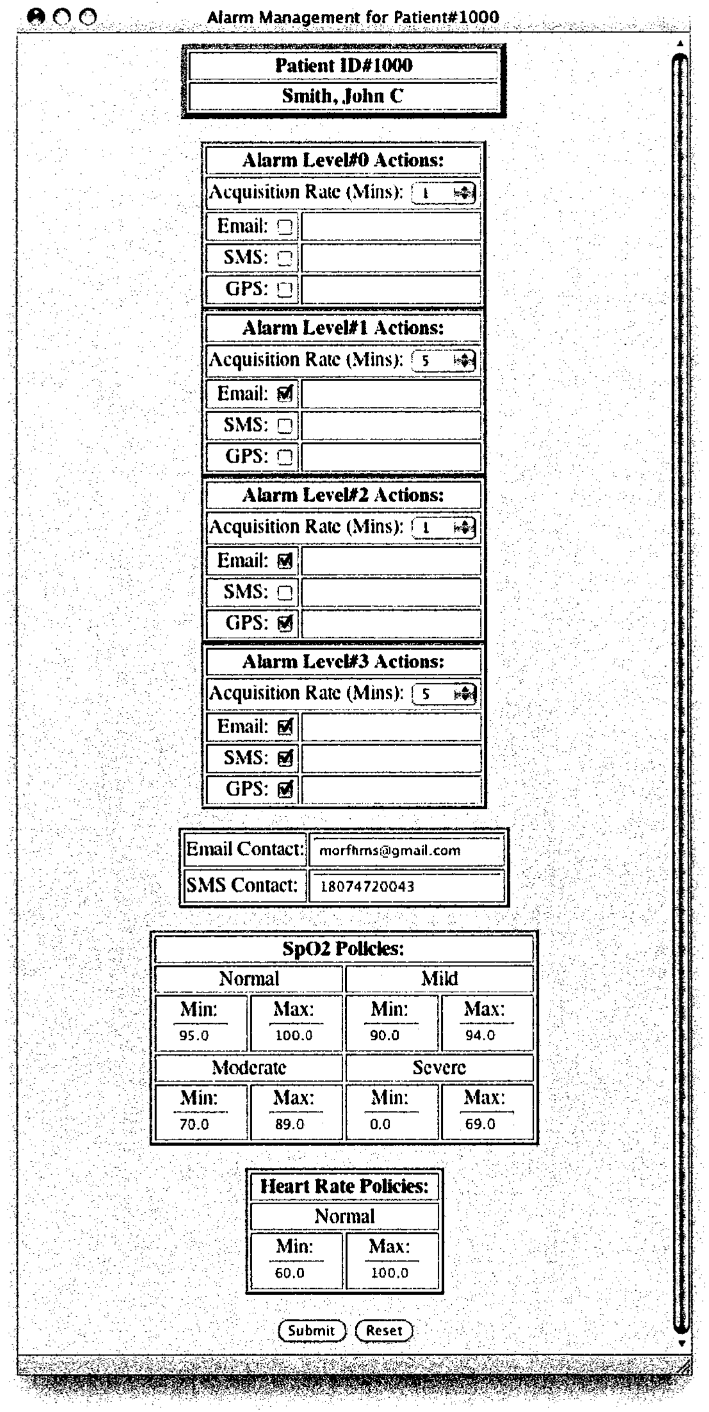
As can be seen in Figure 31, the mobile Web interface is very similar to the standard

Web interface except that it is formatted to fit the smaller screen. In addition to this formatting, a few features such as the numerical sensor data table on the patient data page, have been removed to make the interface less crowded on the smaller screens.

5.2 – Experimental Case Studies

This section covers an exhaustive study of the MORF health monitoring system. Similar to the scenario found in section 5.1.1, we will now discuss a more complete scenario to test each of the alarm levels and their associated actions. In order to test each case we will be modifying the SpO2 policies that govern each of the alarm levels. This is necessary since it is very difficult, if not impossible, to manipulate one's SpO2 levels to the degree required to test each case.

The goal of case #0 was the simplest in that it merely had to trigger an alarm level of 0, which isn't really an alarm level at all but is more of an alarm-less state. Before beginning with this case we chose the following settings, as shown in Figure 32.



Patient ID#1000
Smith, John C

Alarm Level#0 Actions:	
Acquisition Rate (Mins):	1
Email: ☐	
SMS: ☐	
GPS: ☐	

Alarm Level#1 Actions:	
Acquisition Rate (Mins):	5
Email: ☑	
SMS: ☐	
GPS: ☐	

Alarm Level#2 Actions:	
Acquisition Rate (Mins):	1
Email: ☑	
SMS: ☐	
GPS: ☑	

Alarm Level#3 Actions:	
Acquisition Rate (Mins):	5
Email: ☑	
SMS: ☑	
GPS: ☑	

Email Contact:	morfhms@gmail.com
SMS Contact:	18074720043

SpO2 Policies:

Normal		Mild	
Min: 95.0	Max: 100.0	Min: 90.0	Max: 94.0
Moderate		**Severe**	
Min: 70.0	Max: 89.0	Min: 0.0	Max: 69.0

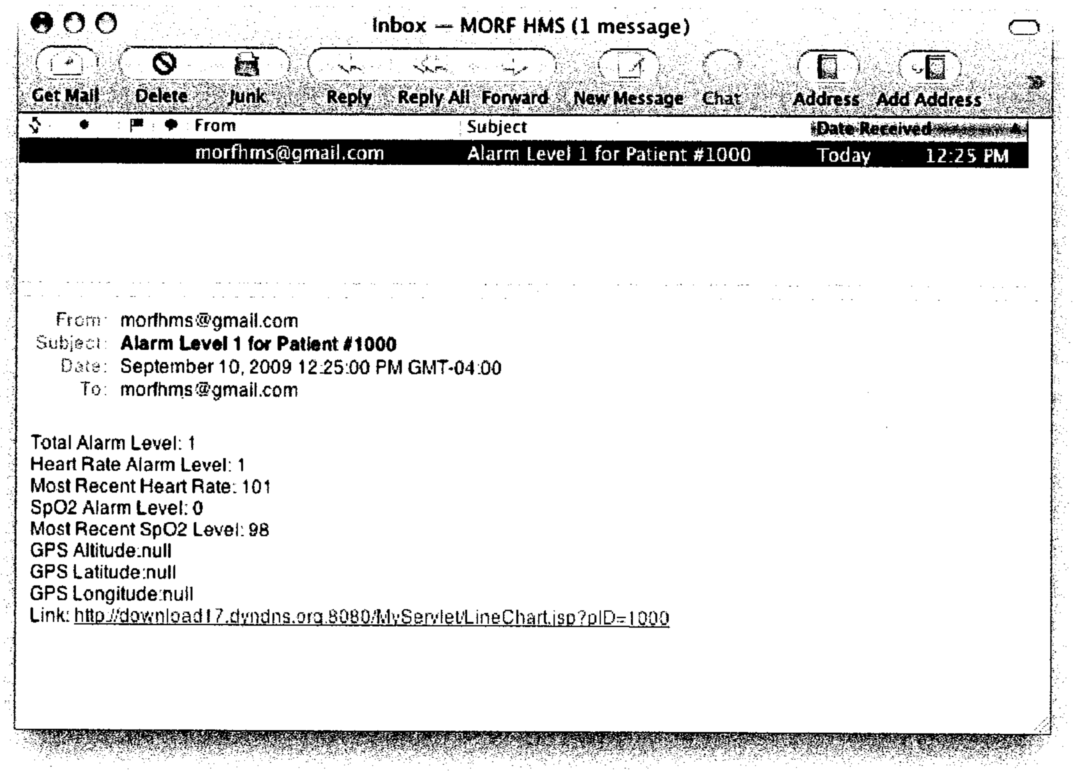
Heart Rate Policies:

Normal	
Min: 60.0	Max: 100.0

Figure 32 - Initial Settings for Experimental Case Study

As can be seen in Figure 32 the SpO2 policies are set for standard levels of hypoxemia. These are the values that we will be modifying in order to simulate each case. For case #0 I allowed the mobile unit to measure my vitals and, as expected, the results came back within normal tolerances setting the alarm status to 0. Since no actions were specified for this alarm level, no actions were taken.

For Case #1, I attempted to raise my heart rate to that of an abnormal state to cause a level 1 alarm to be triggered. I was able to successfully achieve this goal and caused the system to set the alarm status to 1. This caused and email to be sent out to the emergency contact email address. This email can be seen below in Figure 33.



morfhms@gmail.com Alarm Level 1 for Patient #1000 Today 12:25 PM

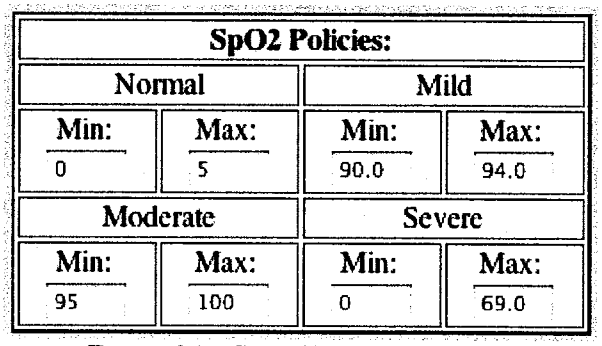
From: morfhms@gmail.com
Subject: **Alarm Level 1 for Patient #1000**
Date: September 10, 2009 12:25:00 PM GMT-04:00
To: morfhms@gmail.com

Total Alarm Level: 1
Heart Rate Alarm Level: 1
Most Recent Heart Rate: 101
SpO2 Alarm Level: 0
Most Recent SpO2 Level: 98
GPS Altitude:null
GPS Latitude:null
GPS Longitude:null
Link: http://download17.dyndns.org:8080/MyServlet/LineChart.jsp?pID=1000

Figure 33 - Case #1 - Alarm Level 1 Email

Since the heart rate was 101 bpm and exceeded the normal heart rate range by being over 100 bpm, the heart rate alarm level was set to 1. GPS coordinates were not specified as an action for this alarm level and are therefore null. As with all alert messages, as link is attached which leads to the patient data page so that the medical professional who receives the alert can view, in detail, the patient's vital data.

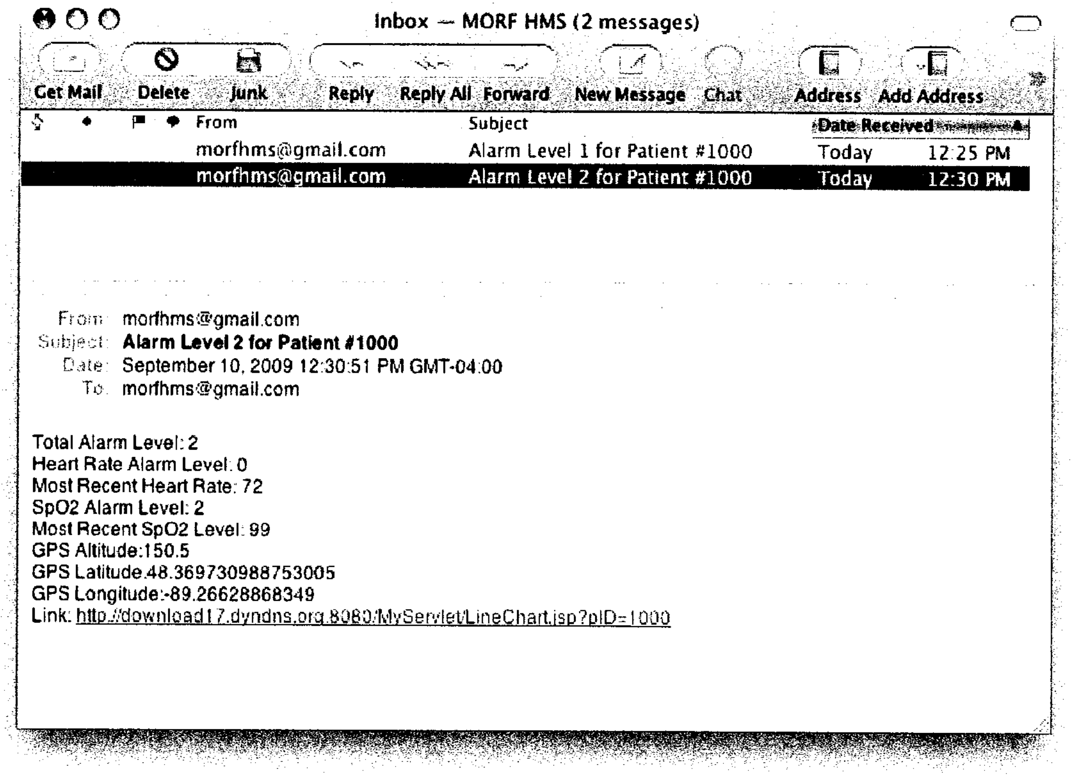
For case #2, In order to trigger a level 2 alarm we proceeded to set the SpO2 policy for mild hypoxemia, which triggers a level 2 alarm, to that of a normal SpO2 level. This way we can trick the system into thinking that a level 2 alarm has been triggered without the test subject's vitals being dangerously critical. The range for normal SpO2 levels were also modified so that they don't interfere with the inference of the alarm level. These modifications can be seen in screenshot in Figure 34.



SpO2 Policies:			
Normal		Mild	
Min:	Max:	Min:	Max:
0	5	90.0	94.0
Moderate		Severe	
Min:	Max:	Min:	Max:
95	100	0	69.0

Figure 34 - Case #2 - SpO2 Policies

Triggering alarm level 2 resulted in GPS coordinates being requested by the system and an email sent out to the emergency contact email address similar to case #1. The email message sent out can be seen, this time with the GPS coordinates present, in Figure 35.



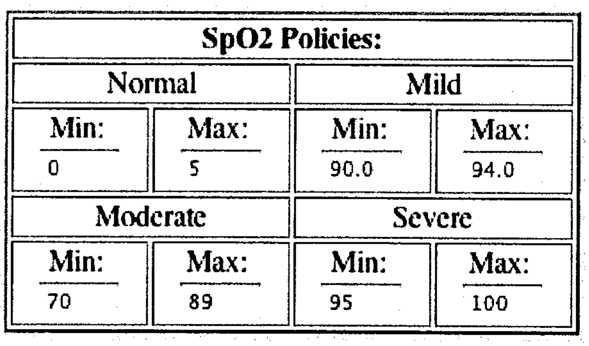
From	Subject	Date Received	
morfhms@gmail.com	Alarm Level 1 for Patient #1000	Today	12:25 PM
morfhms@gmail.com	Alarm Level 2 for Patient #1000	Today	12:30 PM

From: morfhms@gmail.com
Subject: **Alarm Level 2 for Patient #1000**
Date: September 10, 2009 12:30:51 PM GMT-04:00
To: morfhms@gmail.com

Total Alarm Level: 2
Heart Rate Alarm Level: 0
Most Recent Heart Rate: 72
SpO2 Alarm Level: 2
Most Recent SpO2 Level: 99
GPS Altitude:150.5
GPS Latitude:48.369730988753005
GPS Longitude:-89.26628868349
Link: http://download17.dyndns.org:8080/MyServlet/LineChart.jsp?pID=1000

Figure 35 - Case #2 - Alarm Level 2 Email

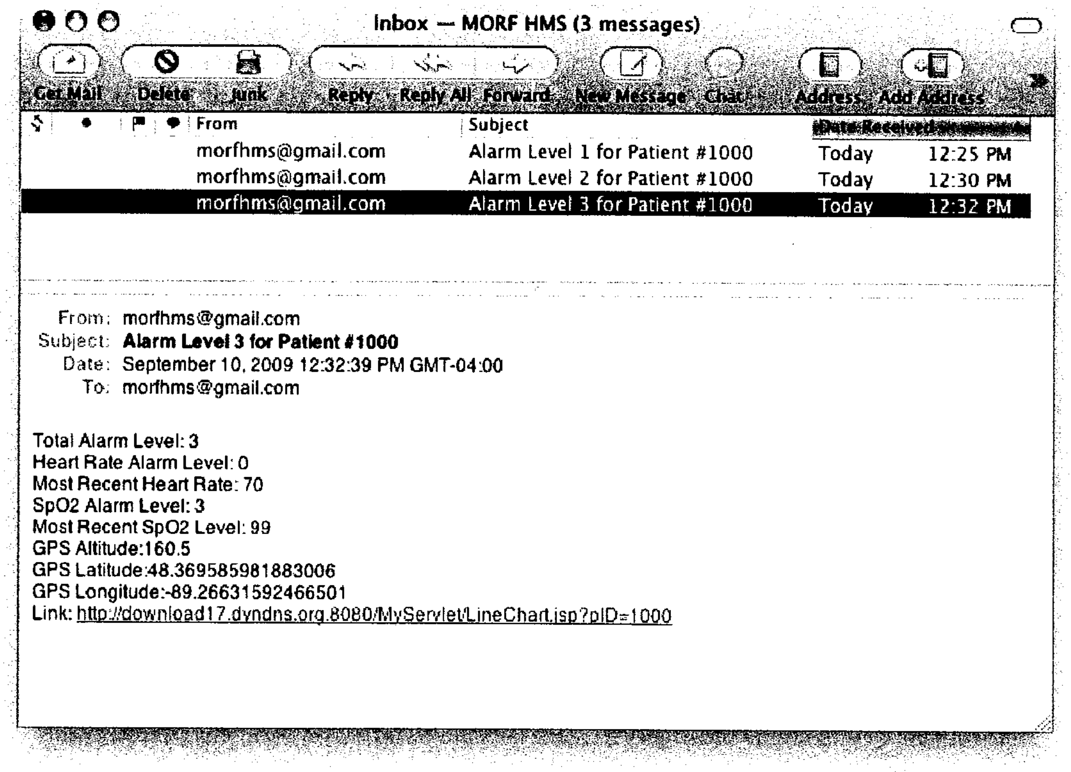
The SpO2 policies for Case #3 have been modified in much the same way as for Case #2, in that, since it is impossible to trigger a level 3 alarm without endangering the test subject, we have tricked the system into thinking that level 3 alarm has been triggered when the values are normal. These modifications to the SpO2 policies can be seen in Figure 36.



SpO2 Policies:			
Normal		Mild	
Min:	Max:	Min:	Max:
0	5	90.0	94.0
Moderate		Severe	
Min:	Max:	Min:	Max:
70	89	95	100

Figure 36 - Case #3 - SpO2 Policies

As with Case #2, GPS coordinates were requested by the system and an alarm email message was sent out. Figure 37 shows the email message sent to the emergency contact.



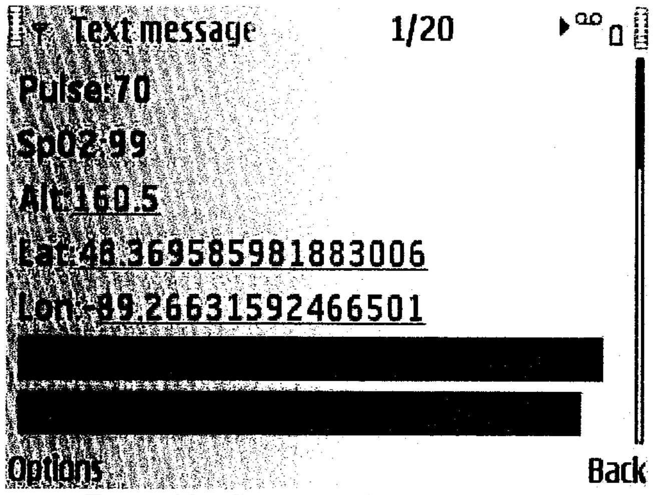
From	Subject	Date Received	
morfhms@gmail.com	Alarm Level 1 for Patient #1000	Today	12:25 PM
morfhms@gmail.com	Alarm Level 2 for Patient #1000	Today	12:30 PM
morfhms@gmail.com	Alarm Level 3 for Patient #1000	Today	12:32 PM

```
   From: morfhms@gmail.com
Subject: Alarm Level 3 for Patient #1000
   Date: September 10, 2009 12:32:39 PM GMT-04:00
     To: morfhms@gmail.com

Total Alarm Level: 3
Heart Rate Alarm Level: 0
Most Recent Heart Rate: 70
SpO2 Alarm Level: 3
Most Recent SpO2 Level: 99
GPS Altitude:160.5
GPS Latitude:48.369585981883006
GPS Longitude:-89.26631592466501
Link: http://download17.dyndns.org:8080/MyServlet/LineChart.jsp?pID=1000
```

Figure 37 - Case #3 - Alarm Level 3 Email

In addition to an email message being sent, one further action specified for an alarm level of 3 is that of sending an SMS text message to the emergency SMS contact number. Figure 38 shows a screenshot of the recieved SMS message taken on the mobile phone using the ScreenSnap software.



Text message 1/20

Pulse:70

Sp02:99

Alt:160.5

Lat:48.369585981883006

Lon:-89.26631592466501

[redacted]

Options Back

Figure 38 - Case #3 - Alarm Level 3 SMS

Together, all of these cases are an effective test of the operation of the MORF system for each possible alarm scenario. Table 10 summarizes the results gathered from this experiment.

Case	Date & Time	Acquisition Rate (Delay until next Cycle)	SpO2	Heart Rate	GPS Altitude	GPS Latitude	GPS Longitude
0	2009/09/10 12:23:16	1 min	97%	67 bpm	N/A	N/A	N/A
1	2009/09/10 12:24:42	5 mins	98%	101 bpm	N/A	N/A	N/A
2	2009/09/10 12:30:07	1 min	99%	72 bpm	150.5m	48.36973°	-89.26629°
3	2009/09/10 12:31:58	5 mins	99%	70 bpm	160.5m	48.36959°	-89.26632°

Table 10 - Results for Experimental Case Study

From Table 10 it can also be seen that the time gaps between each measurement correspond to the appropriate acquisition rate (Figure 32) set for the alarm level that was triggered in each case. Due to the amount of time it takes for the mobile unit to retrieve the sensor data, send it to the server and receive back it's instructions, there will be an additional 30 to 40 second time delay added to the acquisition rate.

The URL included in the alert messages, whether SMS or email, leads to the patient data page, similar to the one shown in Figure 24. On this page we can see the images shown in figures 39 and 40.

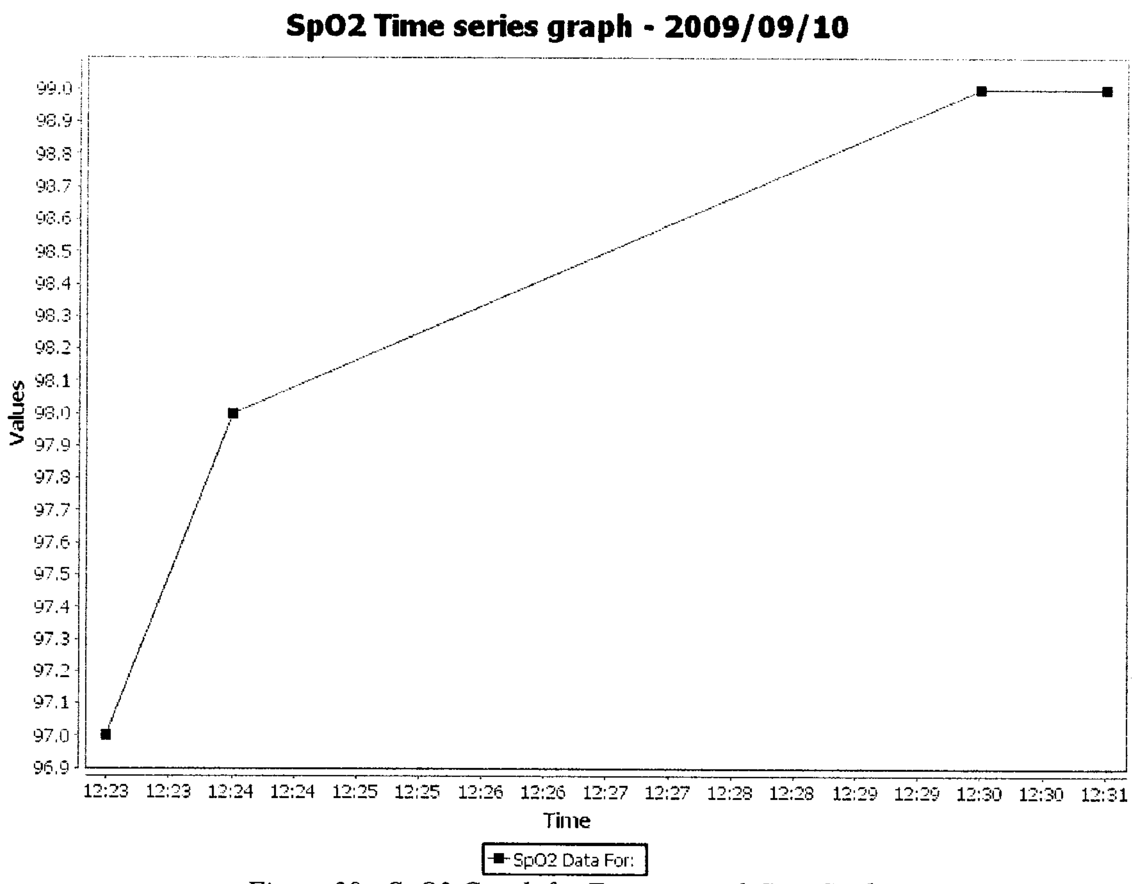


Figure 39 - SpO2 Graph for Experimental Case Study

Figure 39, as shown above, displays the results gathered from this experimental case study for the SpO2 values of our test patient. Plotting these points in graphical format make the information much easier to read, whether it be through a web browser or on a smart phone. Figure 40 displays the experimental data for the patient's heart rate in graphical format, much like Figure 39 displayed the SpO2 data.

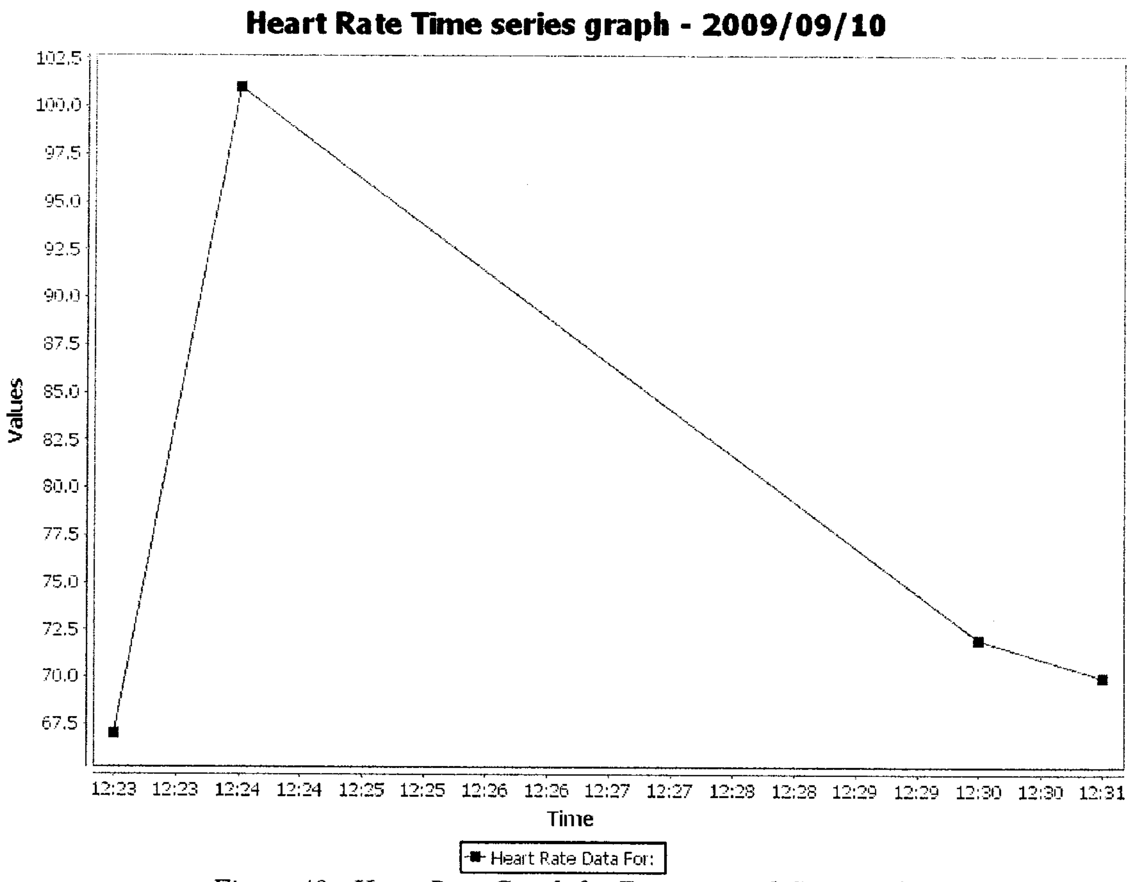
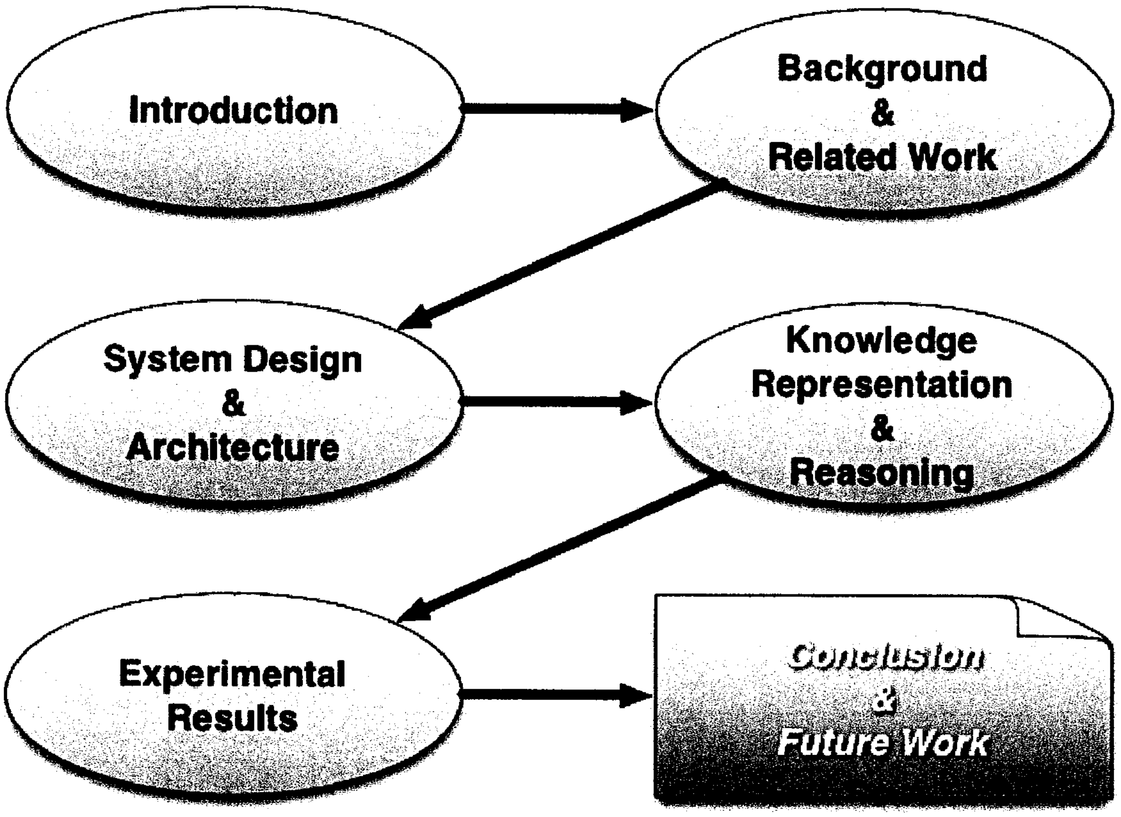


Figure 40 - Heart Rate Graph for Experimental Case Study

chapter 6


Introduction
Background & Related Work
System Design & Architecture
Knowledge Representation & Reasoning
Experimental Results
Conclusion & Future Work

conclusion & future work

Our research, as discussed in this thesis, outlines the prototype system that we designed which monitors the health status of a patient through the use of a mobile monitoring unit. This platform utilizes an ontology-based context model from which processing and reasoning of a patient's health status are performed. By incorporating true mobility, context modelling, knowledge management and various reasoning techniques we have developed a platform that greatly improves upon the current research in this field. The work presented here focuses on the three major aspects of the system design: Low-level sensor data acquisition, mid-level architecture for mobile communications and the high-level reasoning engine and web server.

The low-level section of the MORF system is comprised of the methods and technologies used to acquire the sensor data from the medical sensors. Technologies such as the Bluetooth communication protocol allow the mid-level to communicate with and acquiring data from the low-level sensors. The mid-level section of our system acts as a mobile relay that forwards the incoming sensor data on to the central server. It also carries out commands received from the server, such as modifying the data acquisition rate, gathering GPS coordinates and sending out alert messages. These commands are apart of MORF's feedback system. Once the sensor data is received by the central server it is stored and processed. The reasoning engine takes the numerical sensor data and infers an alarm level based on the doctor-prescribed norms for each specific patient. These alarm levels are used to give meaning to the numerical sensor data. Each alarm level also has a set of actions that can be carried out should that alarm be triggered.

The mobile unit has been designed in such a way that it requires little to no experience to be able to use. It is to be almost fully automated and ultimately controlled by the central monitoring server. The system utilizes a hierarchical control system which simply means that the reasoning engine controls the mid-level system which, in turn, controls the low-level system.

As the system is fully functional in its current state, there are a few additions and modifications that can be made to improve the overall functionality and performance of the MORF platform. The first step would be to increase the reliability of the system. This can be done by adding support for multiple-user ontology editing as mentioned in section 4.1. Adding layers of redundancy in case a particular system or function fails is another way of increasing the system's reliability. This would involve creating a backup system that stores the patient's sensor data on the phone itself in the event of a loss or lack of signal. One final method of achieving greater reliability in the system would be to reduce the amount of communication between the mobile device and the central server. This can be done by moving away from performing all of the reasoning on the server to performing some of the reasoning on the mobile device. The next step would be that of security. As is, the Bluetooth communication between the sensors and the mobile unit is secure and encrypted, but the connection between the mobile unit and the server is not since we are currently using a standard HTTP connection. Since the security of patient information is of vital concern, adding support for secure HTTP connection (HTTPS) is, in the future, a must. Adding a login system to the web interface would improve system security as well. Another improvement to the web interface would involve adding support to the data table page for graphing sensor data over a period of multiple days and weeks. This would enable medical professionals to view their patients' vital trends over a larger period of time. Another modification to the system would involve moving away from the use of rule languages such as SWRL and limiting the use of ontologies for the storage of data that is irrelevant to the making of inferences. One final addition to the overall operation of the system would be that of increasing support for more mobile devices. This would make the system easily adaptable to almost any cell phone on the market today.

References

[1] T. O'Reilly, "What Is Web 2.0 - Design Patterns and Business Models for the Next Generation of Software," O'Reilly Media, Inc. 2009. [Online]. Available: http://oreilly.com/web2/archive/what-is-web-20.html [Accessed: July 14, 2009].

[2] Luciano Floridi, "Web 2.0 vs. the Semantic Web: A Philosophical Assessment", University of Oxford, 2009. [Online]. Available: http://www.philosophyofinformation.net/publications/pdf/w2vsw.pdf [Accessed: August 20, 2009].

[3] PDA Cortex. "Telzuit BioPatch Cardiac Monitoring System". 2005. [Online]. Available: http://www.rnpalm.com/Telzuit_BioPatch_Cardiac_Monitoring_System.htm [Accessed: August 20, 2008]

[4] S. Kreo, S. Kostic, D. Sakac, and Z. Lukic. "mSens Mobile Health Monitoring System". Proc. The International Conference on Computer as a Tool, EUROCON, Belgrade, Serbia & Montenegro: LM Ericsson, vol. 1, pp. 80-83, 2005. [Online]. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=1629863 [Accessed: January 10, 2009]

[5] medGadget. "Another Mobile Monitoring System Promised". 2007. [Online]. Available: http://www.medgadget.com/archives/2007/01/another_mobile.html [Accessed: September 4, 2008]

[6] J. M. Choi, B. H. Choi, J. W. Seo, R. H. Sohn, M. S. Ryu, W. Yi, et al. "A System for Ubiquitous Health Monitoring in the Bedroom via a Bluetooth Network and Wireless LAN". Proc. The 26th Annual International Conference of the IEEE EMBS, San Fransisco, CA, USA: Engineering in Medicine and Biology Society, vol. 2, pp. 3362-3365, 2004. [Online]. Available: http://ieeexplore.ieee.org/Xplore/login.jsp?url=/ielx5/9639/30463/01403944.pdf?arnumber=1403944 [Accessed: March 9, 2009]

[7] S. Dai, and Y. Zhang, "Wireless Physiological Multi-parameter Monitoring System Based on Mobile Communication Networks". In 19th IEEE Symposium on Computer-Based Medical Systems Based on Mobile Communication Networks, Washington, DC, USA: IEEE Computer Soceity, pp. 473-478, 2006. [Online]. Available: http://portal.acm.org/citation.cfm?id=1153145 [Accessed: October 11, 2008]

[8] D. Giuli, and F. Paganelli, "An Ontology-based Context Model for Home Health Monitoring and Alerting in Chronic Patient Care Networks". Proc. The IEEE 21st International Conference on Advanced Information Networking and Applications Workshops, Niagara Falls, Canada: IEEE Computer Society, vol. 2, pp. 838-845, 2007. [Online]. Available: http://portal.acm.org/citation.cfm?id=1250203 [Accessed: October 16, 2008]

[9] J. W. Lee, and J. Y. Jung, "ZigBee Device Design and Implementation for Context-Aware U-Healthcare System". Proc. The IEEE 2nd International Conference on Systems and Networks Communications, Cap Esterel, French Riviera, F: IEEE Computer Society. pp. 22, 2007. [Online]. Available: http://portal.acm.org/citation.cfm?id=1306499 [Accessed: June 5, 2008]

[10] C. S. Ho, L. J. Shiao, and L. Liu, "An Efficient Solution to Ubiquitous Health Care System – from Clinics to Patients", Proc. 7th Int. Workshop on Enterprise Networking and Computing in Healthcare Industry, pp. 150-155, 2005. [Online]. Available: http://ieeexplore.ieee.org/Xplore/login.jsp?url=/ielx5/10018/32162/01500427.pdf?arnumber=1500427 [Accessed: November 3, 2008]

[11] H. van Kranenburg, M. S. Bargh, S. Iacob, and A. Peddemors, "A context management framework for supporting context-aware distributed applications," Communications Magazine, IEEE, vol. 44, no. 8, pp. 67-74, 2006. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1678112 [Accessed: November 3, 2008]

[12] T. K. Shih, T.-H. Wang, C.-Y. Chang, T.-C. Kao, and D. Hamilton, "Ubiquitous e-learning with multimodal multimedia devices," IEEE Transactions on Multimedia, vol. 9, no. 3, pp. 487-499, 2007. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4130376 [Accessed: November 3, 2008]

[13] F. Paganelli, G. Bianchi, and D. Giuli, "A context model for context-aware system design towards the ambient intelligence vision: Experiences in the etourism domain,"pp. 173-191. 2007. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-71025-7_12 [Accessed: November 4, 2008]

[14] A. Krause, A. Smailagic, and D. P. Siewiorek, "Context-aware mobile computing: Learning context-dependent personal preferences from a wearable sensor array," Mobile Computing, IEEE Transactions on, vol. 5, no. 2, pp. 113-127, 2006. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1563997 [Accessed: August 10, 2009].

[15] Microchip Technology Inc., "PIC18F4550 Datasheet". 2007. [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/39632D.pdf [Accessed: March 16, 2008]

[16] Emxys Embedded Instruments, "uIceBlue Wireless embedded control module Getting Started Manual," 2006. [Online]. Available: http://www.emxys.com/media/pdf/MANUAL_Uiceblue.pdf [Accessed: September 9, 2008]

[17] Nonin Medical Inc., "Nonin Model 4100 containing Bluetooth Technology Specifications," 2006. [Online]. Available: http://www.nonin.com/documents/4100%20Specifications.pdf [Accessed: May 20, 2009].

[18] W. B. Saunders, "Oxygen Saturation Monitoring by Pulse Oximetry," AACN Procedure Manual for Critical Care, 4th ed. 2001. [Online]. Available: http://classic.aacn.org/AACN/practice.nsf/Files/PO1/$file/ch%2014%20PO.pdf [Accessed: August 10, 2009].

[19] Vincent Ruberto, "Normals for O2 Therapy," June 26, 2009.

[20] Nokia, "Nokia E71 Device Details," 2009. [Online]. Available: http://www.forum.nokia.com/devices/E71 [Accessed: August 9, 2009].

[21] "Specification of the Bluetooth System, vol 1: Core, v3.0," Bluetooth SIG, April 2009. [Accessed: March 5, 2009]

[22] B. Forgue, "Secure wireless networks for industrial applications," Reed Business Information. 2009. [Online]. Available: http://www.ferret.com.au/n/Secure-wireless-networks-for-industrial-applications-n690552 [Accessed: July 14, 2009].

[23] C. I. Diamond, "ZigBee vs. Bluetooth," 2005. [Online]. Available: http://homepage.uab.edu/cdiamond/ZigBee%20vs%20Bluetooth.htm [Accessed: March 6, 2009]

[24] "GSM," GSM Association. 2009. [Online]. Available: http://www.gsmworld.com/technology/gsm/index.htm [Accessed: July 14, 2009].

[25] "HSPA, the undisputed choice for mobile broadband," Telefonaktiebolaget LM Ericsson. 2009. [Online]. Available: http://www.ericsson.com/technology/whitepapers/hspa_Rev_b.pdf [Accessed: July 14, 2009]

[26] D. L. McGuinness and F. van Harmelen, "OWL Web Ontology Language Overview," February 10, 2004. [Online]. Available: http://www.w3.org/TR/owl-features/ [Accessed: August 9, 2009].

[27] M. K. Smith, C. Welty, and D. L. McGuinness, "OWL Web Ontology Language Guide," February 10, 2004. [Online]. Available: http://www.w3.org/TR/owl-guide/ [Accessed: August 9, 2009].

[28] B. McBride, "RDF Primer," February 10, 2004. [Online]. Available: http://www.w3.org/TR/rdf-primer/ [Accessed: August 10, 2009].

[29] J. Seidenberg and A. Rector, "A Methodology for Asynchronous Multi-User Editing of Semantic Web Ontologies", Proc. 4th International Conference on Knowledge Capture, Whistler, BC, Canada, pp 127-134, 2007. [Online]. Available: http://portal.acm.org/citation.cfm?id=1298406.1298430 [Accessed: September 1, 2009].

[30] I. Horrocks, P. F. Patel-Schneider, H. Boley, S. Tabet, B. Grosof and M. Dean. "SWRL: A Semantic Web Rule Language Combining OWL and RuleML," May 21, 2004. [Online]. Available: http://www.w3.org/Submission/SWRL/ [Accessed: August 10, 2009].

[31] SmartPhoneWare, "Best Screen Snap," 2009. [Online]. Available: http://nokia-e71-software.smartphoneware.com/screen_snap.php [Accessed: August 10, 2009].

[32] ManageEngine, "Applications Manager," 2009. [Online] Available: http://www.manageengine.com/products/applications_manager/download.html?100 [Accessed: August 10, 2009].

Appendix A – Published Papers

- Luke Docksteader and Rachid Benlamri, "Mobile Ontology-Based Reasoning and Feedback System", Proc. 3rd International Conference on Digital Information Management – ICDIM 2008, University of East London, London, UK, 13-16 November, 2008.