UNIVERSITY OF DUBLIN

Exploiting Bluetooth 4.0 for Secure, Cloud-Enabled Monitoring of Palliative Care Patients

by

Will Browne

A dissertation submitted in partial fulfillment for the degree of Master in Computer Science

Submitted to the University of Dublin, Trinity College

May 2014

Declaration of Authorship

I, Will Browne, declare that the following dissertation, except where otherwise stated, is entirely my own work; that it has not previously been submitted as an exercise for a degree, either in Trinity College Dublin, or in any other University; and that the library may lend or copy it or any part thereof on request.

Signed: _____

Date: _____

UNIVERSITY OF DUBLIN

Abstract

School of Computer Science and Statistics
Submitted to the University of Dublin, Trinity College

Master in Computer Science

by Will Browne

Supervised by Dr. Jonathan Dukes

Telemedicine is an important practice that removes the necessity for physical encounters between patients and physicians. Combined with the capabilities of remote monitoring systems, a patient can be monitored using physiological sensors which seamlessly gather data. Using an Internet of Things (IoT) model, non-invasive physiological sensors that capture a patient's current health status can be exposed through a Telemedicine platform. By closely monitoring patient's vital signs, physicians can capture a snapshot of a patient's well being. This is an important factor in the lives of palliative care patients, where focus is placed on quality of life.

This dissertation outlines a secure, cloud-enabled Telemedicine solution aimed at the field of palliative medicine. This dissertation plans to expose the Bluetooth 4.0 protocol in order to provide a multi-stream data transmission protocol of low and high bit-rate data. Use of an adaptive data stream enables efficient sensor traffic with capabilities of complex sensor data transmission in emergency situations and low bandwidth data that can provide episodic updates. Each patient sensor makes use of a gateway to transmit information to a cloud based service, which publishes current and historic data for individual patients. Secure storage, access and transmission of patient data is also an important concern. This dissertation outlines two security mechanisms using a combination of symmetric and asymmetric cryptography.

Acknowledgements

First and foremost I would like thank my supervisor Jonathan Dukes for his support and direction throughout the past year.

Secondly, I would like to thank each of the lads for keeping the last five years entertaining.

I would like to express my gratitude towards my family, in particular my parents, as without them none of this would have been possible.

I would also like to thank my girlfriend Gabriella, who helped bring this idea to life and has continued to support me in whatever I do.

Finally, a thank you to those who proof read this document.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

With the technological advancement in Telemedicine in recent years, new concepts such as wireless body area networks (WBANs), pervasive health monitoring and management services have become increasingly popular [5]. With the support of Cloud Computing, WBANs can be enhanced as the cloud offloads complex operations and provides a scalable and accessible framework that can be made easily available to care-givers. This paper features research into the design of a system to provide remote palliative care of patients by highlighting the methodologies for transmitting vital sign data from Bluetooth sensors through a gateway to the cloud by using energy-efficient transmission of variable bit rate streams, and data security mechanisms.

The system architecture integrates three components:

- Wireless Body Area Sensor Network - A network of energy efficient wireless physiological sensors used to provide the basis for remote patient monitoring
- Gateway - Exposes the sensor network to the wider Internet in order to expose patient data to the cloud and receive sensor control information
- Cloud application and secure data store - Ubiquitous access and control over the sensor network and patient data, whilst ensuring data confidentiality and integrity.

From this, the main areas of research focus are:

- The co-existence of Bluetooth Classic and Bluetooth Low Energy protocols to act as variable bit rate streams for simple and complex medical sensors
- A Cloud Computing model to provide a central repository for live and historic patient health records
- Secure transmission and storage of patient data to ensure confidentiality
- Share-ability of information - through permissions and leases of patient information

1.1 Research Problem

The problem addressed by this dissertation is combining the use of variable bit rate Bluetooth protocols to provide energy efficient transmission of data, which is published to cloud services through a Bluetooth 4.0 and Internet Protocol (IP) enabled gateway. The focus is to design a system that caters towards patients of palliative care that can be used to deliver means of care whilst maintaining a high quality of life. This necessitates a focus on low power wireless technologies to provide a non invasive network of biomedical sensors.

Through the Cloud Computing paradigm, the system enables an Internet of Things (IoT) model over the remote sensor network so that sensors can be individually addressed and controlled via a ubiquitous platform that also offers access to secured patient information.

1.2 Motivation

The author of this research was inspired to conduct this dissertation primarily because of personal experience. Whilst researching the field, Bluetooth 4.0 was identified as a suitable protocol focus for sensor data transmission in the context of health care. Its combined use of a dual protocol under the Bluetooth Smart Ready brand supported the area of palliative medicine because of its multi transmission capabilities. As a result, the Bluetooth 4.0 protocol became the focus of the research.

1.3 Desired Outcomes

The main objective of the proposed solution is to design a Telemedicine system that aids the care and the preservation of quality of life of patients in palliative care. The focus of the design is:

- A energy efficient transmission protocol using Bluetooth Low Energy as a control channel with Bluetooth Classic and Bluetooth Low Energy as concurrent-capable data channels. Patient data can be transmitted over different communication channels to ensure timely care can be delivered by medical professionals.
- To provide a means through which care-givers can access patient data streams, for example, to control the level of detail in the information gathered from sensors or how frequently sensors need to publish information, etc. Furthermore, to provide a centralised service that exposes clinical data that can be accessed by a patient's carers.
- To ensure access, transmission, and storage of patient data is secured at all times. A local gateway is used to securely route each biomedical data stream to the cloud where all information is stored in encrypted form.

The desired outcome is a system that can be used as an effective Telemedicine platform for terminally ill patients in order to maximise quality of life.

1.4 Document Structure

Chapter 2 will cover the background research that formed the foundation of this dissertation. Chapter 3 focuses specifically on the Bluetooth 4.0 protocol and introduces the main focus of the research. Chapters 4 and 5 covers the design and implementation of the multi-tier telemedicine system and its corresponding proof of concept. All system performance evaluations are detailed in Chapter 6. Finally, the document concludes with Chapter 7 following a discussion of future work.

Chapter 2

State of the Art Review

With an overall focus in the context of palliative care, this chapter covers the detailed research required to design a Telemedicine system introduced in Chapter 1.

This chapter begins by examining the current state of Telemedicine to determine its role in palliative care, along with its potential benefits and barriers. To provide the source of biomedical data, wireless medical sensors are then considered to determine what can be achieved in modern day with small sensors, and what information can they provide to those who care for palliative care patients.

In order to form a network of sensors, research of wireless body area networks is carried out to determine their properties to aid care in an energy efficient and non-invasive manner. To provide transmission capabilities of captured sensor data, wireless transmission protocols are studied to determine their versatility, throughput capabilities, energy efficiency, and reliability. Combinations of multiple wireless protocols are also explored.

To expose physiological parameters to the Internet, gateways are investigated to determine their reliability, mobile capabilities and their compatibility with existing wireless transmission protocols. The Cloud computing model is explored as a ubiquitous access and control hub of patient data and sensor control operations. Finally, this chapter looks at how to ensure the confidentiality and integrity of medical information in a Telemedicine system.

2.1 Telemedicine

Telemedicine is the use of IT for remote medical consultation. It involves the provision of clinical services at a distance, removing the need for physical encounters between patients and physicians, which instead can be replaced by images, video, data, and voice services. [6]

Palliative care is an area of healthcare that focuses on the quality of life of a patient when there is no expectation of a medical cure. This typically means that a patient resides at home and is cared for by a primary carer - a family member or local palliative care nurse. Symptom control, pain relief and psychological support are the typical services provided during palliative care. These services may require frequent visits from a local palliative care nurse/doctor or visits to hospital[7].

Palliative care acts as a support system to help patients live as comfortably as possible until death and also to help the patient's family cope during the illness and during bereavement. Palliative care is predominantly used in instances of symptoms caused by cancer. Relief from physical, psychosocial and spiritual problems can be achieved in over 90% of advanced cancer patients through palliative care.[1]

There are multiple challenges faced in situations of palliative care:

- Patients may still require visits to hospitals, specialists, etc. to receive additional care. In some cases, this is made difficult by patients living in isolated communities and remote regions.
- Having multiple sources of care also can potentially result in the segregation of patient data as there is no shared central repository of historical clinical data. In some cases this may lead to misinformation shared between physicians.
- Providing a high quality level of symptom treatment whilst maintaining best quality of life

[1] http://www.who.int/mediacentre/factsheets/fs297/en/

The use of Telemedicine in a situation of palliative care provides benefits such as:

- A patient is not required to travel from their home in order to receive medical consultation, which reduces the need for outpatient visits, thus reducing the overall cost of healthcare. Patients can instead conduct regular health exams at home. Physicians can view the data and perform analysis in real time or at a later stage
- If the Telemedicine service is coupled with a remote monitoring system, there is no longer a requirement for manual vital sign entry which results in decreased workload and less chance for error.
- A centralised IT system can provide a medium for collaboration between medical staff.
- Data collected over long periods of time in a patient's natural environment offers a clearer view of the patient's health to medical professionals compared to short stays at a hospital [8]. This could also give rise to formal electronic health records (EHR) of patients, which can form the basis for future research.

When the medical focus becomes about quality of life, measures to ensure such whilst still providing health services in a way that is non-invasive is hugely important. [9] outlines a cancer reporting and monitoring Telemedicine system that focuses on maximising HRQoL (health-related quality of life). The approach uses cloud computing services to facilitate data access and future collaborative cancer research. Alternatively, Telecare and Telediagnosis platforms can be used to transmit information in the event of an urgent case detection or when required by the medical personnel [10]. This demonstrates the basis for a Decision Support System (DSS), where unnecessary data transmission is avoided.

There are however many challenges faced when it comes to Telemedicine [11]:

Infrastructural Issues - Poor bandwidth in some areas while cost expensive in others.

Viability Issues - In order to implement Telemedicine, training is needed for medical professionals and IT staff.

Acceptance Issues - For patients, using high end technology may be too obstructing. However, once the benefits are seen, the acceptance rate will likely be high such as has been seen with mobile telephony and rural Internet services.

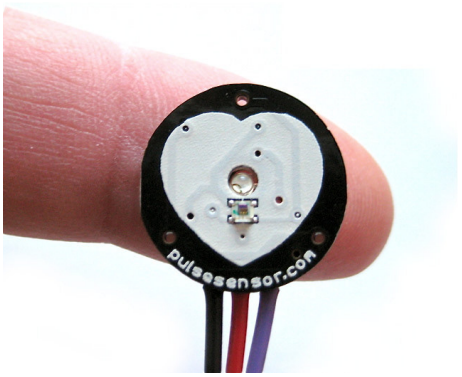
Regulations - Restricting laws that govern medical practice. Costs associated with obtaining licensure.

Financial Barriers - Lack of reimbursement

The application of specialised Telemedicine has shown to improve patient treatment quality and efficiency in various settings [12, 13, 14]. However, there has been much concern about the quality of research concerning Telemedicine. As part of a systematic review carried out in 2010 to determine the effectiveness of Telemedicine [15], it was found that 21 reviews concluded that Telemedicine is effective, 18 found evidence that Telemedicine is promising but incomplete and 41 that evidence is limited and inconsistent. The problematic themes were nature of economic analysis, the benefits for patients, and telemedicine as a complex and ongoing collaborative system. The result found was that information on how best to use Telemedicine in health care is still lacking - *"Large studies with rigorous designs are needed to get better evidence on the effects of telemedicine interventions on health, satisfaction with care and costs"*. In 2011, the same authors conducted a review of the methodologies used for assessing Telemedicine systems [16] (50 reviews). It was found that more comprehensive studies are recommended to provide evidence for a system's effectiveness as evidence for effectiveness of Telemedicine is the primary knowledge gap.

In a traditional Telemedicine patient monitoring system, a patient is equipped with medical sensors that are capable of transmitting health information to an external monitoring station. This allows care-givers to analyse, interpret and treat patients remotely. These devices can be low powered, meaning they do not require frequent battery changes. For patient's with a need for more complex biomedical data, high powered sensors can be used in times of need - for instance, EMG, EEG, etc. For example, one Telemedicine system outlines a non-invasive, wearable neck-cuff system capable of real-time monitoring and of physiological signals using a stethoscope and multiple oximeters. [17]

(a) Polar H7 heart rate sensor [2]



(b) Pulse sensor designed for the Arduino platform [4]

(c) Zebris EMG Bluetooth Measuring System for Recording Muscle Activity [6]

(d) Equivital wireless physiological monitor including ECG and respiratory waveforms [8]

FIGURE 2.1: An example of medical sensors available on the market

2.2 Medical Sensors

In many cases the well being of a person can be determined through the measurement of certain vital signs such as body temperature, heart rate, ECG, EEG. With the advancement in sensor technology, these vital signs can be retrieved through very small electronic sensors attached to a body (wearable or implantable), which are capable of transmitting the information wirelessly to a remote medical data store.

[2] http://www.polar.com/us-en/products/accessories/H7_heart_rate_sensor
[4] http://pulsesensor.myshopify.com/
[6] http://www.zebris.de/english/medizin/medizin-emg-messung.php
[8] http://www.equivital.co.uk/

There are a number of sensors available on the market that can be used to provide meaningful health information (see Figure 2.1). This physiological data can take the form of temperature readings, heart rate, skin conductance, etc. These types of readings typically require small amounts of information at a time. Other medical information such as ECG, EMG, etc. demand much higher data rates (see Table 2.1). ECG is a physiological measurement of the rate and regularity of heartbeats. Its readings provides more detailed information concerning the operation of the heart compared to a simple heart rate sensor. For patients with cardiovascular problems, ECG would be a necessary consideration when determining a diagnosis. As palliative care caters for many different illnesses, it is imperative to support a variety of sensors. Therefore in order to successfully support different sensor data rates, appropriate communication capabilities are required.

There are also a variety of sensors that can are now used in different ways to provide new information to care-givers. For instance, traditional medical technologies such as Electroencephalography (EEG) have recently been investigated to provide new diagnostics such as assessing pain within a patient who is unable to communicate [18]. Similarly, galvanic skin response readings have been shown to measure distress levels [19]. With these capabilities, it is of interest to build solutions to provide new means of care. Combining the advances in electronic sensor technology with traditional health care provides a new way for patients to be cared for.

Application	Data rate	Bandwidth (Hz)	Accuracy (bits)
ECG (12 leads)	288 kbps	100-1000	12
ECG (6 leads)	71 kbps	100-500	12
EMG	320 kbps	0-10,000	16
EEG (12 leads)	43.2 kbps	0-150	12
Blood saturation	16 bps	0-1	8
Glucose monitoring	1600 bps	0-50	16
Temperature	120 bps	0-1	8
Motion sensor	35 kbps	0-500	12
Cochlear implant	100 kbps	-	-
Artificial retina	50-700 kbps	-	-
Audio	1 Mbps	-	-
Voice	50-100 kbps	-	-

TABLE 2.1: Different medical sensors and their corresponding data rates [1]

In order to use medical sensors in a way that is non invasive, they not only have to be wearable or implantable, they have to consume as little power as possible

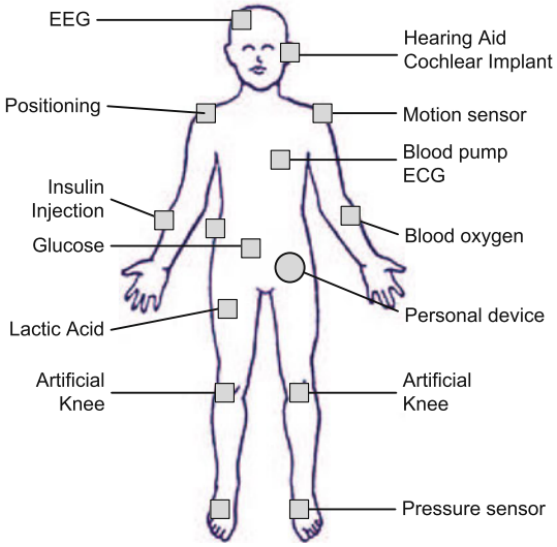


FIGURE 2.2: Placement of different types of medical sensors on a person [1]

while still providing as much useful physiological data as possible. It would not be acceptable to assume that a terminal patient would forgo their comfort and mobility for the sake of remote monitoring. This eliminates sensors that require frequent charging and/or cumbersome cables/wires. Therefore in order to build a suitable network of medical sensors, there are some important considerations:

- Duty cycle - How often are sensors active?
- Protocol efficiency - What work is carried out when they are active?
- TX power - How much is the cost of transmission?
- How long do they have to transmit when they are on?
- How much energy does the signal processing consume?

A lot of this is determined by the wireless protocol chosen (see Section 2.4 for description of wireless protocols). As a result, most radios designed at a low energy market have been designed to minimise air time.

2.3 Wireless Body Area Networks

A Body Area Network (BAN) encompasses communication between devices that are supposed to be carried or worn on the body. This includes mobile phones,

watches, headsets, medical sensors. In a health care context, a BAN consists of multiple electronic sensors placed strategically on a person. In a wireless body area sensor network (WBASN), these sensors are wireless and form the basis of remotely monitoring and tracking information. A WBASN is ideal in the instance of health care as it can provide a non-invasive method of collecting patient data, offering greater physical mobility.

However, a WBAN is limited by multiple factors such as memory, interference, power, computation, communication capabilities. For the purpose of this dissertation, the most important focus of the WBAN is ensuring low power consumption, high availability, reliable communication capabilities through a wireless protocol.

WBANs can be extended through gateways in order to interface with the cloud. This means that information formed at a BAN can be transmitted to the wider Internet/cloud. This is referred to as a cloud-enabled BAN. Health care devices within a cloud-enabled BAN can send physiological data to a gateway device. [20] describes a system consisting of an unobtrusive wireless body area network and a localised health server. Time-stamped patient information is periodically uploaded the home server, which may integrate this information into a local database for user's inspection or it may forward the information further to a medical server. Other research describes a cloud-enabled WBAN architecture for pervasive healthcare systems using energy-efficient routing, cloud resource allocation, and data security mechanisms to provide the seamless integration between Mobile Cloud Computing (MCC) and WBANs [5].

There are many different technologies used in a BAN in order to provide communication between the network of devices. An investigation was carried out to determine the best protocol capable of to use to providing long term non-invasive care whilst supporting multi sensor capabilities.

2.4 Wireless transmission protocols

This section is dedicated to covering the most relevant wireless transmission protocols viable for use in the context of Telemedicine (see Figure 2.2). The protocol must be capable of outputting physiological data at data rates required for a range of medical sensors such as ECG and EMG. Moreover, the protocol must offer low

power consumption capabilities in order to support non invasiveness at a sensor level by not obstructing the user.

Technology	*Datarate*	*Operatingspace*	*Peakpower*	*Topology*	*Jointime*
Bluetooth Classic	1–3 Mb/s	1–10 m	45mA @ 3.3V	Scatternet	3s
Bluetooth Low Energy	1 Mb/s	1–10 m	28mA @3.3V	Piconet, Star	<100ms
ZigBee	250 kb/s	10–100 m	16.5mA @1.8V	Star, Mesh	30 ms
ANT	1 Mb/s	10–30 m	22mA @3.3V	Star, tree, or Mesh	-

TABLE 2.2: A comparison of wireless protocol technologies [3]

2.4.1 Bluetooth

Bluetooth is a technology used for exchanging data wirelessly over short distances. Bluetooth version 4.0 was conceived in 2010. It is made up of two components – Bluetooth Classic and Bluetooth Low Energy. Bluetooth Low Energy technology is ideal for applications requiring episodic transfer of small amounts of data. Where streaming or real time data transfer is required, Classic Bluetooth technology is the preferred choice as it achieves substantially greater throughput than Bluetooth Low Energy technology. Both protocols can co-exist in Bluetooth hardware which is referred to a "dual mode" or Bluetooth Smart Ready device.

A major advantage with Bluetooth is that billions of mobile phones and PCs have already embraced the technology. Using a Smart Ready chip, an infrastructure of billions of devices will quickly become available. Furthermore, combined as a single chip, a Bluetooth Smart Ready module can save space in an already space constrained smartphone device. This gives Bluetooth Low Energy a *'free-ride that will lead to economies of scale for chip vendors, and a vibrant ecosystem of devices for products to connect to'* [21] - page 176. By 2018, more than 90 percent of Bluetooth enabled smartphones are expected to be Smart Ready devices[9]. Additionally, the total shipments of Bluetooth 4.0 devices are expected grow to 2.9 billion per year by 2016, with a cumulative shipment figure of over 20 billion by 2017, according to ABI research[22].

The throughput of Bluetooth Low Energy is lower than Bluetooth Classic for continuous communications, since Bluetooth Low Energy is based on attributes transfer instead of offering a communication session where data can be transmitted in as many packets as required, without any size constraint. Therefore, it is necessary to perform data compression when the amount of data to send is too

[9] http://www.bluetooth.com/Pages/Mobile-Telephony-Market.aspx

large to be stored in a Bluetooth Low Energy attribute [23]. Other research demonstrates a successful approach using an 2-lead ECG sensor with Bluetooth Low Energy [24]. The approach uses an RS232 connection to collect the ECG data (a rate of 250 samples per second) through a signal processing unit. The signal processing unit then converts the data into a Bluetooth profile and transmits the information to a mobile device for analysis. [25] outlines a portable and low-cost system for non-invasive and real-time measurement of heart beat, blood pressure, blood flow and arrhythmias for athletes through the use of Bluetooth. The system uses an Android smartphone to provide data concerning the response of the body to fatigue.

In order to obtain maximum benefit from Bluetooth Low Energy, its parameters require fine tuning depending on its use cases [26]. The research provides experimental results on Bluetooth Low Energy performance evaluation by investigating the impact of its adjustable parameters. These parameters, *connInterval* and *connSlaveLatency*, represents the time between the start of two consecutive connection events and the number of consecutive connection events during which a slave device is not required to listen so that the radio can be powered down.

There is a full chapter dedicated to all research carried out on the Bluetooth protocol - see Chapter 3.

2.4.2 ANT and ANT+

ANT is a ultra-low power (ULP) wireless protocol that runs in the 2.4 GHz ISM band. It is designed for ultra-low power, ease of use, efficiency and scalability in sectors such as sports, fitness, and health. The protocol uses the concept of *device profiles* to communicate. Ant have published a list of device profiles online that can be used within devices. Some examples are blood pressure, muscle oxygen, and heart rate[10]. The ANT protocol has been shown to demonstrate the concept of non-invasive wearable jewellery consisting of heart-rate, galvanic skin response, and nicotine sensors [27].

ANT+ is an interoperability function that can be added to the base ANT protocol which provides the collection, transfer, and tracking of sensor data, whilst

[10] http://www.thisisant.com/developer/ant-plus/device-profiles

maintaining all ULP features of the base protocol. It represents the agreed upon definitions for what the information is transmitted over the ANT network.

2.4.3 ZigBee

ZigBee is a low-cost, low-power, wireless technology built on top of 802.15.4 MAC/-PHY. It is designed for small devices of limited processing and memory capabilities, such as wireless sensors. As a result, ZigBee is highly optimised for low-duty cycle operation of sensing devices (IE a sensor can shut off its radio majority of the time). This is in contrast to Bluetooth Classic, where a slave needs to keep synchronisation to the master, resulting in much longer radio on time, and hence much higher average power consumption [3]. However other research analyses power consumption for the ANT, ZigBee, and Bluetooth Low Energy protocols in a cyclic sleep scenario and found that Bluetooth Low Energy achieved the lowest power consumption, followed by ZigBee and ANT [28]. In some cases it is also possible to combine two protocols into one solution. For instance, a prototype referred to as BlueBee acts as a gateway tool that combines the functions and capabilities of both Bluetooth and ZigBee [29].

2.5 Routing Protocols

In this instance, the term *routing protocols* refers to the concept of managing two or more concurrent channels of information that use different protocols. This provides some useful benefits such as combining capabilities of multiple protocols, and also heightens its interoperability features. A scheduler and a control protocol that can manage both Bluetooth Low Energy and ANT wireless protocols at the same time has been demonstrated [30]. Similarly, [31] demonstrates the concurrent use of Bluetooth and ZigBee for high capacity multimedia data streams.

Routing protocols for WBANs used in a Telemedicine system must provide:

- Low end to end delay
- Low packet drop rate
- Low energy consumption

- A throughput to support a variety of sensor devices

2.6 Internet Gateways

A gateway is a link between two devices that facilitates the sharing of information when no common communication protocol exists between both parties. Gateways can be used to expose a private network of sensors to the wider Internet, by bridging the gap between the sensor's transmission protocol and the Internet Protocol (IP). Middleware in the gateway can access each sensor device directly as if it were a collector talking to it locally, while also being capable of receiving IP packets.

For example, using a Bluetooth 4.0 enabled device, a patient's vital information can be captured at a WBAN and transmitted to the cloud in order to provide medical information to care-givers. The gateway acts as an intermediary between the sensor network and the cloud application, with a responsibility of transferring the information securely (see Security Section 2.8) via WiFi to the cloud, and relaying received commands to an individual or group of sensors. In order to ensure the longevity of the gateway's battery, any processing of the information is left up to the cloud application. As long as the gateway device is in close proximity of the wearable sensors, the sensors can transmit their data and also receive commands in order to adapt to the requirements of the care-givers. For instance if the patient's Bluetooth Smart Ready smartphone acts as the WBAN gateway, data can be monitored 24/7 as long as the phone is within the vicinity of the sensor(s). However compared to a laptop/PC, smartphones are limited in computational capacity and power consumption. For instance, the technical performance of an Android smartphone as a wireless personal area network (WPAN) has been investigated [32]. The research concludes that smartphones are capable of functioning as a gateway in an mHealth context while also simultaneously performing normal operations, with battery power being the limiting factor.

2.6.1 Internet of Things

The Internet of Things (IoT) refers to the concept of uniquely addressable devices within the Internet. It is based on the idea that the Internet Protocol could be

applied to even the smallest of devices. This is an idea that would eliminate the requirement for gateways in systems operating throughout the Internet.

Standards are being developed for generic gateways and for connecting sensors directly to IPV6. On December 4th 2013, Bluetooth Special Interest Group (SIG) announced Bluetooth version 4.1 which provided a software update to all existing Bluetooth 4.0 hardware. The specification improved integration with Long-Term Evolution (LTE) and shows an increased interest in pushing Bluetooth into the area of IoT by laying the groundwork for IP-based connections. The Bluetooth SIG have also published a white paper describing of RESTful APIs. These APIs, implemented in a gateway, can allow an Internet client application to find, connect and operate Bluetooth Smart servers.

6LoWPAN (IPv6 over Low power Wireless Personal Area Networks) is a protocol definition that enables the transmission of IPv6 packets in low power wireless networks, specifically IEEE 802.15.4 [33]. 6LoWPAN is a developing standard from the Internet Engineering Task Force (IETF). The 6LoWPAN WG is developing a specification for the transmission of IPv6 packets over Bluetooth Low Energy [34]. Glowbal IP [35] is an example which provides homogeneous access to devices through IPv6. It offers a reduction in the overhead of 6LowPAN networks and the integration of legacy systems that cannot implement IP protocols. Similarly, ZigBee IP is a scalable architecture that offers end-to-end IPv6 networking capabilities using ZigBee hardware. ZigBee IP enables the IoT without the need for intermediate gateways.

2.7 Cloud Computing

Cloud Computing refers to the ubiquitous provision of computational resources across the Internet [36]. Cloud Computing has become a widely accepted computing paradigm. Furthermore, it has been established as a suitable health applications model for its cost effective services (including data management and storage, and computational resources) and features (portability, reliability, and scalability).

- Portability - Data can be accessed and shared globally

- Reliability - Cloud service providers offer 24/7 monitoring and automatic updates
- Scalability - Dynamic provisioning of resources based on customer's needs

Health care related Cloud Computing solutions are grouped into five categories:

- Emergency Medical System (EMS)
- Health Cloud eXchange (HCX)
- Health ATM Kiosks
- Digital Imaging and communications in Medicine (DICO)-based system
- HealthCloud [11]

In the context of palliative care, two models are most relevant:

EMS - Provides access to patient health records and helps to provide care when required

HCX - A distributed Web interactive system for sharing health records on the Cloud. HCX also facilitates the sharing of health records between EHR systems. [37]

The Cloud acts as a perfect medium for Telemedicine, allowing health professionals to communicate and provide healthcare services through the Cloud application. Similarly, WBASNs are resource constrained and yet have high demands for real-time data transmission and processing. These requirements can be met using a Cloud Computing model by providing processing and storage on demand.

Cloud Computing has had a revolutionary effect on Telemedicine. Serving information through the Cloud enables better data availability and the sharing of patient data to authorised parties. This information can form the basis of a patient's treatment and also can form insight into things such as disease evolution, rehabilitation process, the effects of drug therapy etc. However, data security and legal issues may be the strongest resistance to the adoption of Cloud Computing in the health IT sector [38].

Decision support systems can be used to provide automatic actions based on certain recorded events. For instance, [39] creates an association between patient data and data levels that indicate an urgent status, which can then be used to alert care-givers.

2.8 Security and Privacy

When concerned with people's well being and one's personal information, confidentiality is a very important consideration. There are a number of security challenges faced in a Telemedicine system [40]:

- How to ensure the privacy and integrity of the patient's data, given that the wireless channel at the BAN and the medical data stored in the cloud are easily subject to many forms of attack?
- How to ensure that only those who are authorised can access the data?
- How to prevent someone from using captured sensors to recover sensitive medical information or inject false information?
- How to prevent outsiders from committing replay attacks?

2.8.1 Bluetooth and WBAN Security

It is a challenge to implement traditional security infrastructures in such lightweight networks, since they are by design limited in both computational and communication resources. Key management of a WBAN has been identified as a crucial component to support the security architecture, which is an area not very well explored [1].

An efficient and energy saving approach known as SEKEBAN has been identified where the generation and distribution of symmetric cryptographic keys to constituent sensors of a WBAN are managed [41]. Other approaches have utilised the human body as a medium through which to provide secure inter-BASN communication [42]. The approach uses the human body as the authentication identity or the means of securing the distribution of a symmetric cipher key.

Existing work [43] has provided a review of recent studies in the analysis of Bluetooth security issues. Other work [4] classifies a range of threats against the Bluetooth 4.0 protocol (see Figure 2.3) and provides steps that can be taken to attempt to reduce the threats as much as possible. The security issues of the Bluetooth-enabled devices have also been highlighted by demonstrating the possibility of injecting or recording sounds in Bluetooth enabled headphones without an authenticated connection. [44].

Surveillance	Blueprinting, bt_audit, redfang, War-nibbling, Bluefish, sdptool, Bluescanner, BTScanner
Range extension	BlueSniping, bluetooone, Vera-NG
Obfuscation	Bdaddr, hciconfig, Spooftooph
Fuzzer	BluePass, Bluetooth Stack Smasher, BlueSmack, Tanya, BlueStab
Sniffing	FTS4BT, Merlin, BlueSniff, HCIDump, Wireshark, kismet
Denial of service	Battery exhaustion, signal jamming, BlueSYN, Blueper, BlueJacking, vCard-Blaster
Malware	BlueBag, Caribe, CommWarrior
Unauthorized data access	Bloover, BlueBug, BlueSnarf, BlueSnarf++, BTCrack, Car Whisperer, HeloMoto, btpincrack
Man in the middle	BT-SSP-Printer-MITM, BlueSpooof, bthidproxy

TABLE 2.3: Bluetooth attacks [4]

According to the Bluetooth 4.0 Core Specification, the security provided through Bluetooth transmission is 128-bit AES, meaning that all information transmitted from a Bluetooth device to another is secured. In the context of a Cloud enabled WBASN, the secured information received from a sensor by the Internet gateway is then left unprotected.

2.8.2 Gateway security

With the majority of smartphones being both WiFi and Bluetooth enabled, a smartphone can function as a gateway device for a Bluetooth sensor network (See below list of Bluetooth Smart Ready enabled phones available on the market). Functioning as a mobile device, the gateway can be carried around with a patient

to provide a continuous snapshot of their vital signs. This means that the system is not restricted to a patient's home. A patient can carry out their day to day activities whilst being monitored seamlessly in the background. This is an important aspect of palliative care.

Bluetooth Smart Ready smartphones on the market:

- Apple iPhone (5s, 5c, 5 & 4s)
- Google Nexus 5, Nexus 4
- BlackBerry Q10, Z10
- HTC One, One Max
- Motorola Droid RAZR, Ultra, Maxx, Mini, Moto G, Moto X
- Samsung Galaxy Series
- Sony Xperia Series

Depending on the security model used, the gateway may be responsible for encrypting all received data traffic. This requires a keystore mechanism. [45] identifies a Subscriber Identity Module (SIM) card as a potential solution for key and credential storage for smartphone devices.

2.8.3 Cloud security

With an application hosted in the Cloud, reliability and security become very big concerns. Trust must be placed in Cloud hosting providers to supply sufficient security measures. Security issues such as Distributed Denial of Service attacks (DDoS), confidential data leakage, data ownership and access control are considered the biggest issues involved in the data storage and sharing of information through a cloud platform [46].

One model has been identified that does not require trust to be placed in a Cloud provider - A *Host Proof model*. No confidential information is ever exchanged with the cloud in plain text, so any breaches could not expose any secret user information. However, key management remains one of the limiting factors [47].

Key management is a critical aspect of a security system. Using a Cloud Computing model, there are multiple options for where secret keys may be stored [5].

Centralised in the Cloud provider

- *Advantages* - Utilises the scalable computational and network resources of the Cloud. Relies upon the direct user-to-cloud link.
- *Disadvantages* - Requires trust in the Cloud provider to not decode encrypted user data stored on its servers

Centralised in a trusted authority outside of the Cloud domain

- *Advantages* -Does not require trust in the Cloud provider. May control access to Cloud data as an intermediary node.
- *Disadvantages* - Requires maintenance of a scalable authority server by the client, or trust in a third -party guardian as a paid service

Fully decentralised among users

- *Advantages* - Requires no additional network elements. Key sharing may utilise cheap local links such as WiFi or Bluetooth
- *Disadvantages* - Obtaining keys may require arbitration by an authority which entails additional traffic. Revocation is inefficient.

2.9 Summary

This chapter has presented a state of the art research into the design of a comprehensive secure Telemedicine system that supports the care of palliative care patients. The following chapter will provide an in depth look at the wireless transmission protocol used to capture physiological data formed at a WBAN through variable bit rate streams - Bluetooth 4.0.

Chapter 3

Bluetooth

This dissertation describes a wireless body area network solution for palliative care based on Bluetooth 4.0 technology. Specifically, the proposed solution combines the use of Bluetooth Classic - for high data-rate, real-time sensor data - and Bluetooth Low Energy for - low-bit-rate, episodic data and control signalling for the sensor network. This chapter provides an overview of Bluetooth and will form the basis for the design and implementation described in later chapters.

3.1 Background

In 2010, the Bluetooth Special Interest Group (SIG) published their Core Specification 4.0. It represents two technologies, both designed with different capabilities [48, 2, 49]:

- Bluetooth Classic - designed as a continuation of the previous versions of Bluetooth (V1.0 - V3.0), with a focus on high data rates and high throughput
- Bluetooth Low Energy - an entirely new Bluetooth protocol stack aimed at very low power applications that run off a coin cell battery or similar power source. This new low energy technology provides a low power alternative to previous Bluetooth versions, which focused on feature enhancements or an increase in throughput. As a result, Bluetooth Low Eenergy is not backward compatible with the previous Bluetooth versions.

SpecificationVersion	*ReleaseDate*	*KeyFeatures*
1.0 and 1.0a	Jul 1999	These were the very first versions of the Bluetooth specification. The primary objective was to replace the serial cables with a wireless link.
1.0b	Dec 1999	This version added minor updates to fix some issues.
1.1	Feb 2001	Bluetooth was ratified as IEEE 802.15.1.1-2002 standard.
1.2	Nov 2003	This release of the Bluetooth standard added new facilities including Adaptive Frequency Hopping (AFH) to provide better resistance to interference in noisy environments and Extended Synchronous Connection Oriented (eSCO) links were added to provide better voice quality. This was also ratified as IEEE 802.15.1.1-2005. This was the last version issued by IEEE and after that Bluetooth technology evolved independently.
2.0 + EDR	Nov 2004	This release of the Bluetooth standard introduced enhancements to the throughput using Enhanced Data Rates (EDR). The previous versions of the standard supported a throughput of 721 kbps. This version increased it to 2.1 Mbps. This made it suitable for applications that required fast data transfers like file transfer, browsing, printing, etc.
2.1 + EDR	Jul 2007	This version brought in several enhancements and add SSP (Secure Simple Pairing) to both simplify the pairing mechanisms and to improve security.
3.0 + HS	Apr 2009	This version provided a significant increase in throughput (24Mbps) by introducing the support for multiple radios.
4.0	Jun 2010	This version went into a completely different direction compared to the previous versions. While in the previous versions the main focus was to introduce new features and enhance the throughput, this version addressed the markets where the need was not for high throughput but of ultra-low power. This was referred to as Bluetooth Low Energy.

Table 3.1: Evolution of the Bluetooth technology [2]

The Bluetooth 4.0 specification permits devices to implement either or both of the Bluetooth Low Energy and Bluetooth Classic systems. Those that implement both are known as Bluetooth 4.0 dual-mode devices, or **Bluetooth Smart Ready** devices. On the other hand the term **Bluetooth Smart** indicates an Bluetooth Low Energy-only device. A Bluetooth Smart device features a single mode radio, meaning it only supports Bluetooth Low Energy connections, thus requiring either a Smart Ready or another Smart device in order to function. Bluetooth Classic implementations are single-mode devices.

3.2 Bluetooth Classic

Originally introduced in 1994 at Ericsson labs, Bluetooth was conceived as a wireless ad hoc peer to peer protocol used to replace RS-232 serial data cables. Operating in the 2.4 GHz radio frequency, Bluetooth creates a short-range network ideal for applications requiring streaming capabilities. Since version 1.2, the technology is defined and maintained by the Bluetooth SIG in its "Core Specification," which serves as a uniform structure for devices to inter-operate. Over the years its uses have grown to support file exchange, voice/audio streaming, GPS, etc. As a result there are multiple different data rates achievable by the protocol depending on the protocol version.

- Bluetooth 1.0 - BR (Basic Rate) → supports a maximum data rate of 721 kbps
- Bluetooth 2.0+EDR (Enhanced Data Rate) → supports a maximum data rate of 2.1 Mbps
- Bluetooth 3.0+HS (High Speed) → further increased throughput to a rate of 24 Mbps

[Note: Hereafter, Bluetooth Classic will be referred to as BR/EDR. This term is often used to refer to a device that supports Bluetooth Classic capabilities. Similarly, the Bluetooth Low Energy protocol will be denoted by the term LE]

A BR/EDR use case might be to establish a hands-free connection that persists for a long period of time to ensure low latency for potential incoming calls. Another

use case might be to maintain a connection between an external device such as a mouse peripheral to a PC. There must be low latency between the two Bluetooth end points, so as not to impact the user experience.

3.2.1 Key Features

Ad hoc - Does not rely on pre-existing infrastructure

Wide range - A range of 10m up to 100m that does not require line of sight

Low power - 10 μA in standby, 50mA while transmitting.

Secure - 128-bit encryption that offers protection against data eaves dropping

Profiles - Availability of real world profiles such as heart rate, proximity, and alert notification

Interoperable - Operates in the open, license free 2.4 GHz frequency band

3.2.2 Architecture

The Bluetooth architecture comprises multiple layers:

- Controller - Typically a physical device which performs low level operations such as device discovery, initiating connections, exchanging data packets, security, low power modes, etc.
- Host - Is exposed to the functionality of the controller in order to perform serial port emulation, splitting of packets, data streaming, etc.
- Profiles - Responsible for providing a specific use case model for communicating Bluetooth devices
- Applications - Provides the user with the capability of interacting with the Bluetooth functionality to enable a use case.

The primary Bluetooth functionality can be considered to be split into two logical parts for each Bluetooth device - the Host and the Controller. The Host executes

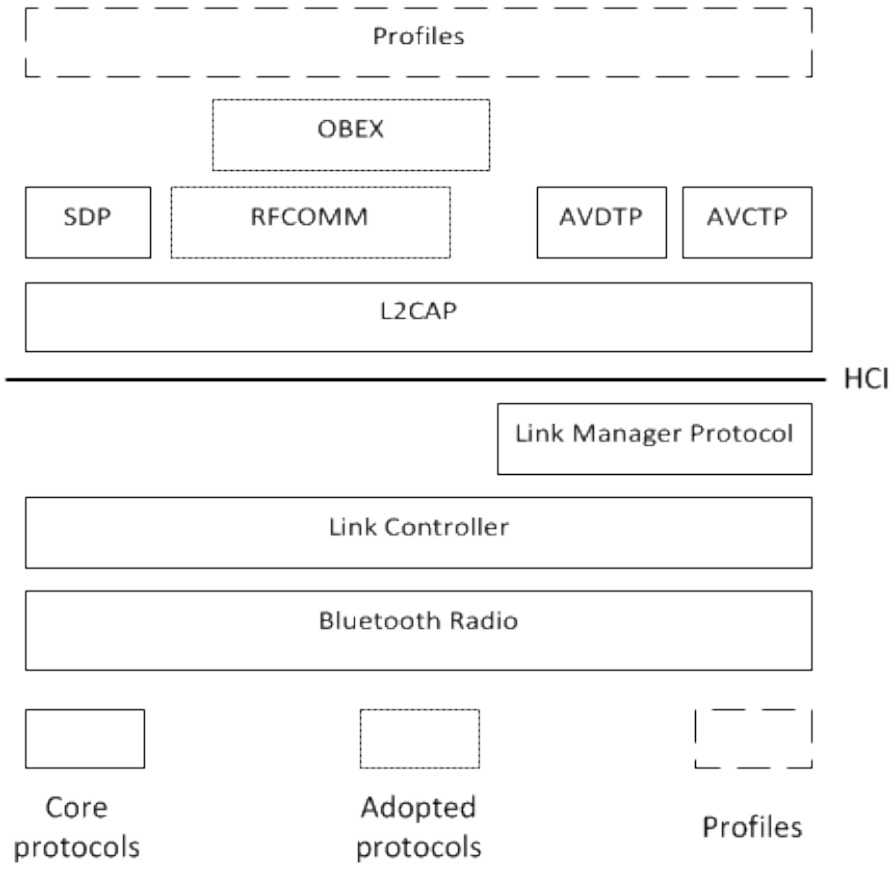


FIGURE 3.1: Bluetooth classic protocol stack [2]

the upper layers of the Bluetooth protocol stack, typically on a CPU or microcontroller of a phone/PC etc. The Controller executes the lower layers of the Bluetooth protocol stack and is typically embedded into the Bluetooth chip. The Host Controller Interface (HCI) provides an interface between the Host stack and the Controller. This communication typically happens over UART, RS-232, SD or USB.

For example, in the scenario of a Bluetooth dongle attached to a PC:

PC → where the Host executes
USB cable → the HCI interface is provided through USB
Bluetooth Dongle → the Bluetooth chip in the dongle represents the Controller

The software for both Host and Controller are independent, meaning that development for both can occur simultaneously. Furthermore, the host from one vendor is fully compatible with the controller of another vendor. This is how USB Bluetooth dongles from different vendors are capable of offering Bluetooth functionality through a PC.

Communication over the HCI interface occurs in the form of packets. There are four types of packets that can be sent over the HCI interface:

1. HCI Command Packet → used for sending commands from the host to the controller.
2. HCI Asynchronous Data Packet → used to exchange data between the host and controller
3. HCI Synchronous Data Packet → used to exchange synchronous data between the host and controller
4. HCI Event Packet → used by the controller to notify the host when an event has occurred

BR/EDR Stack

This section will cover the BR/EDR protocol stack - see Figure 3.1 for overview.

At a Radio Frequency (RF) level, within the ISM band of 2.4000-2.4835 Ghz, BR/EDR divides the spectrum in 79 evenly spaced 1 MHz channels. To prevent interference, BR/EDR uses Frequency Hopping Spread Spectrum (FHSS) where hopping between frequencies leads to the use of time-slots to control communication between devices (see Figure 3.2). The Bluetooth Radio is responsible for transmitting and receiving packets of information on the physical channel. The radio transforms a stream of data to and from both the physical channel and the Link Controller.

The Link Controller performs a variety of functions:

- Management of physical channels
- Formation of the piconet and scatternet
- Formation of packets
- Providing Bluetooth radio with packets
- Carrying out an *inquiry*
- Carrying out a *connection* and *page scan*
- Security

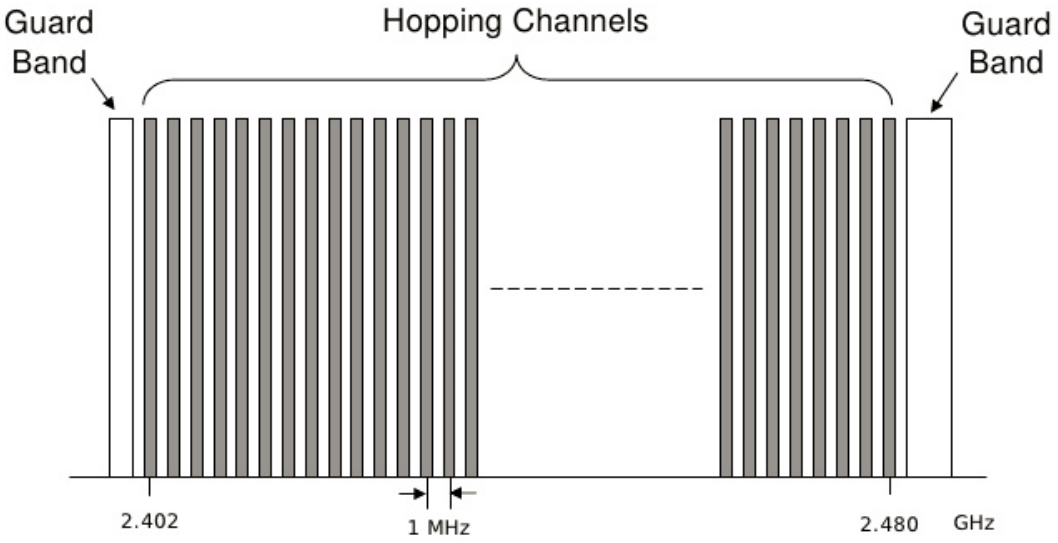


FIGURE 3.2: Frequency Hopping Spread Spectrum (FHSS) causes the Bluetooth device to switch the 1 megahertz (MHz) wide channel 1600 times per second across all 79 channels available to Bluetooth [2]

- Power management

The Link Manager controls and negotiates the operation of a Bluetooth connection between two devices. This includes the setup and control of logical transports and logical links, and for control of physical links. The Link Manager Protocol is used to communicate between the Link Managers (LM) on the two devices connected by the Asynchronous Connection Oriented Logical Transport (ACL). The ACL is used to carry signals, user data and broadcast traffic.

The protocols within Bluetooth can be grouped into two categories: Core and Adopted Protocols (see Figure 3.1). The Core protocols refer to are the protocols central to the Bluetooth technology, whereas the Adopted protocols have originated from existing standards which have been adapted to the Bluetooth protocol.

The L2CAP protocol (a core protocol) sits above the Baseband layer and provides data services to the upper layer protocols using ACL links. Communication with the L2CAP is based on the concept of *Channels*, where a Channel represents a data flow between entities. L2CAP performs a variety of major functions:

1. Protocol and Channel Multiplexing
2. Per channel flow control

3. Error control and retransmissions
4. Streaming channels
5. Quality of Service (QoS)
6. Group Management

There are three main protocols which are used to move data between applications and the L2CAP layer – RFCOMM, MCAP and AVDTP. Two profiles – HID and AVDTP – include their own protocol transports and interact directly with L2CAP. The RFCOMM protocol is the most widely used protocol and is derived from the GSM 07.10 serial multiplexing protocol. It presents a virtual serial port. This is covered in detail in Section 3.2.3.

Communication

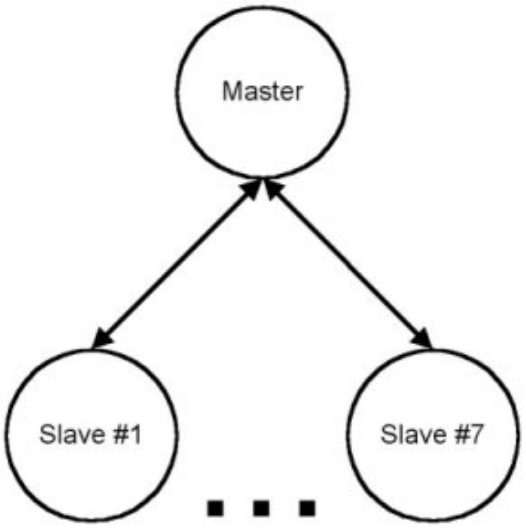
BR/EDR compatible devices can either function as a Master or Slave. Communication between a Master and Slave is based on two network topologies - **Piconet** and **Scatternet**.

Piconet - The smallest unit of Bluetooth communication which can consist of a single Bluetooth Master and up to seven Slaves (see Figure 3.3(a))

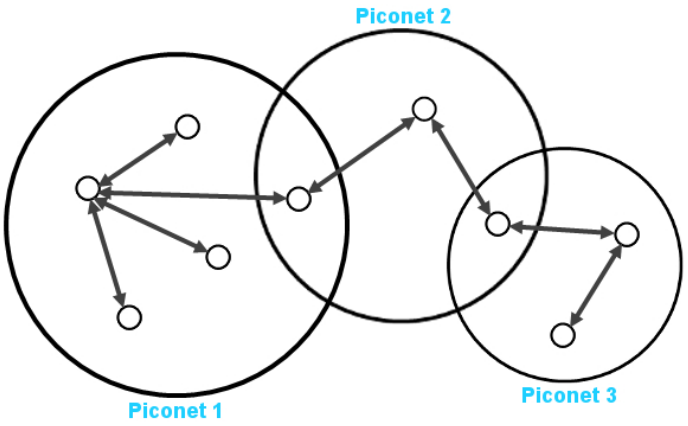
Scatternet - Consists of two or more piconets connected with a shared device (see Figure 3.3(b))

In order to setup a Bluetooth communication between two devices A and B:

- One of the devices (for example B) needs to need in a mode what it can be found by another device - *discoverable*
- The other device (A) must be capable of searching for Bluetooth devices nearby - *inquiry*
- In order to connection the devices, B must be *connectable*
- Once a connection is made - *paging* occurs - device A becomes the Master and device B becomes the Slave - *connected*
- In order to finalise communication - *disconnection* - can be initiated by either Master or Slave



(a) Piconet[1]



(b) Scatternet[2]

FIGURE 3.3: Bluetooth BR/EDR Topologies

3.2.3 Profiles

In order to setup a network of communicating Bluetooth modules, both devices involved require defined behaviour and interoperability guidelines. As the technologies are tailored for different use cases, both subsets of Bluetooth 4.0 use different profile and service architectures. The Generic Access Profile (GAP) defines the way in which Bluetooth devices discover each other and make their connections. It is the most basic of Bluetooth profiles, but is used by every other profile as the foundation for establishing the link.

[1] http://www.wirelessdevnet.com/channels/bluetooth/features/bluetooth.html
[2] http://www.summitdata.com/blog/ble-overview/

SIG specifications offer profiles ready for mainstream adoption:

- Serial Port Profile (SPP) - RS-232 serial communication
- Hands Free Profile (HFP) - Hands free phone communication
- Health Device Profile (HDP) - Medical devices
- Dial-up Networking Profile (DUN) - Dial-up Internet connections
- Advanced Audio Distribution Profile (A2DP) - Audio streaming
- Object Exchange (OBEX) - Generic data exchange

The SPP is the simplest of the Bluetooth profiles, which emulates an RS-232 serial port connection using RFCOMM between two peer devices. A serial port is a nine pin I/O port that exists on many PCs and can be emulated through USB. It is a quick and simple way to add wireless connectivity to a device that has an existing serial port. However, its simplicity is due to the fact that neither the data protocol nor the data formats are defined. The effect of this is that there is no higher-level application interoperability within SPP implementations. Designers are required to define their own protocols and formats.

3.2.4 Security

There are four security modes for Bluetooth access between two devices where a security mode is determined by the vendor of each device.

1. Non-secure
2. Service level enforced security
3. Link level enforced security
4. Link level enforced security with encrypted key exchange

Devices and its services have different security levels. For devices, there are two levels - *trusted device* and *untrusted device*. A *trusted device* is a paired device,

and has unrestricted access to all services. An *untrusted device* is an unpaired device, and therefore access is restricted.

Available service security levels depend on the security mode. Services have three security levels:

- Services requiring authorisation and authentication
- Services requiring authentication only
- Services open to all devices

Bluetooth uses a process in which two devices need to be paired to communicate with each other. This process, referred to as *pairing*. The process involves the exchanging of a 4-digit passkey once two devices have agreed to communicate with each other. This passkey can be considered as a secret password that is shared between the two Bluetooth devices. For example, if two mobile phones wanted to communicate, the user must input the passkey on both devices. The passkey is then used to create a link key, which is subsequently used for authentication between the two devices. Once a pairing has been established it is remembered by the device, eliminating the need to enter the passkey again in the future.

Secure Simple Pairing (SSP) is another pairing process which was introduced to make the pairing procedure easier for the user. The main goals of SSP was to protect against *passive eavesdropping* and *man-in-the-middle* (MITM) attacks.

Passive Eavesdropping protection - The introduction of 16-digital alphanumeric passkeys meant a stronger link key. SSP uses Elliptic Curve Diffie Hellman (ECDH), which provides very good defence against eavesdropping.

MITM protection - SSP provides two mechanisms to prevent MITM attacks - Numeric Comparison and Passkey Entry.

Numeric Comparison - The user is required to determine whether two 6-bit numeric codes appear the same on both devices. If the user enters yes on both devices, then pairing is initiated.

Passkey Entry - The passkey mechanism is used when one device has input capability and no display capabilities but the second device has display

capabilities. Similar to Numeric Comparison, the user is required to input a 6 digit number from the display capable device to the input capable device. Again, pairing is successful if both numbers match.

In order to provide encrypted communication between two paired Bluetooth devices (authentication/pairing is mandatory in this case), both Master and Slave must agree. If agreed, all data exchanged is encrypted with an encryption key, which can be of length 8-bits to 128-bits.

A process known as 'Just Works' allows pairing two devices seamlessly and without any PIN code entry. This can be used in the cases where security is not required as it provides no authentication. This leaves communication susceptible to MITM attacks.

3.3 Bluetooth Low Energy

Bluetooth Low Energy (originally Bluetooth Wibree) is a wireless personal area network technology that became apart of the Bluetooth Standard in 2010, after having originally been developed by Nokia in 2006. The main goal of this technology was to create a protocol with very low energy consumption and low latency. It achieves this through the design of low duty cycles with burst transmissions. Although features of the BR/EDR protocol are inherited in the technology, LE is completely independent of BR/EDR and instead is targeted at new market segments that haven't previously used open wireless standards. This typically refers to use cases where data is required from once a second to once every week or more.

There are many different use cases for the Bluetooth LE technology. Some are:

- Health care - devices such as thermometers, glucose monitors
- Sports and fitness - speedometers, GPS
- Smart homes - presence detection, central heating controls
- Mobile payments - electronic wallet applications
- Locator - pet tracking

3.3.1 Key Features

Low power - Providing up to years of operation on coil cell batteries. *Peak Current* - tens of mA, *Idle Mode Current* - tens of nA, *Average Current* $\sim\mu$A (assuming <1% duty cycle)

Throughput - Maximum of 305 kbps (in typical use cases however, devices will not require such a high data rate)

Short range - Supports a typical range of 30m - 100m

Low cost - Cheaper radio chipsets

Robust - Uses a strong 24 bit CRC on all packets ensuring the maximum robustness against interference and uses frequency hopping to secure a robust transmission

Very low latency - Connection setup and data transfer can be achieved as fast as 3ms. In LE it is faster to re-establish a connection instead of keeping a connection alive like in BR/EDR

Secure - AES-128 for both authentication and encryption

3.3.2 Architecture

LE maintains a layered architecture like BR/EDR and in many ways is very similar to architecture defined by Bluetooth Classic systems. Both Host and Controller remain the primary protocol components. The LE stack modifies some of the existing components such as the L2CAP layer and GAP profile, and also replaces some layers in an effort to gain power savings.

LE Stack

The LE radio is constrained by the requirement that it can be implemented using the same Radio Frequency (RF) chain already present in a BR/EDR chip. Like BR/EDR, it uses the 2.4 GHz ISM band, with (Gaussian frequency-shift keying) GFSK modulation. It divides the spectrum into 40 channels, each 2 MHz wide, and incorporates FHSS to avoid interference. Three of these channels are used for advertising, initiating connections and broadcasting, while the remaining 37 are used for data transfer during an active connection (see Figure 3.5). The maximum/peak power consumption is set to less than 15 mA and the average power

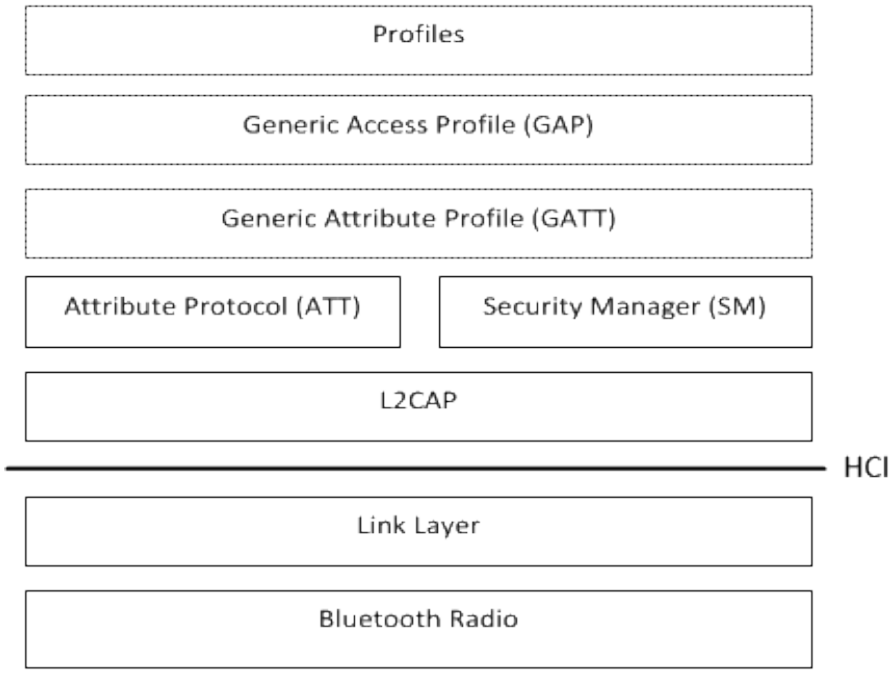


FIGURE 3.4: Bluetooth low energy protocol stack [2]

consumption is at about 1 μA. A foundation for the low energy consumption is the very fast connection set-up (few ms) and use of short messages.

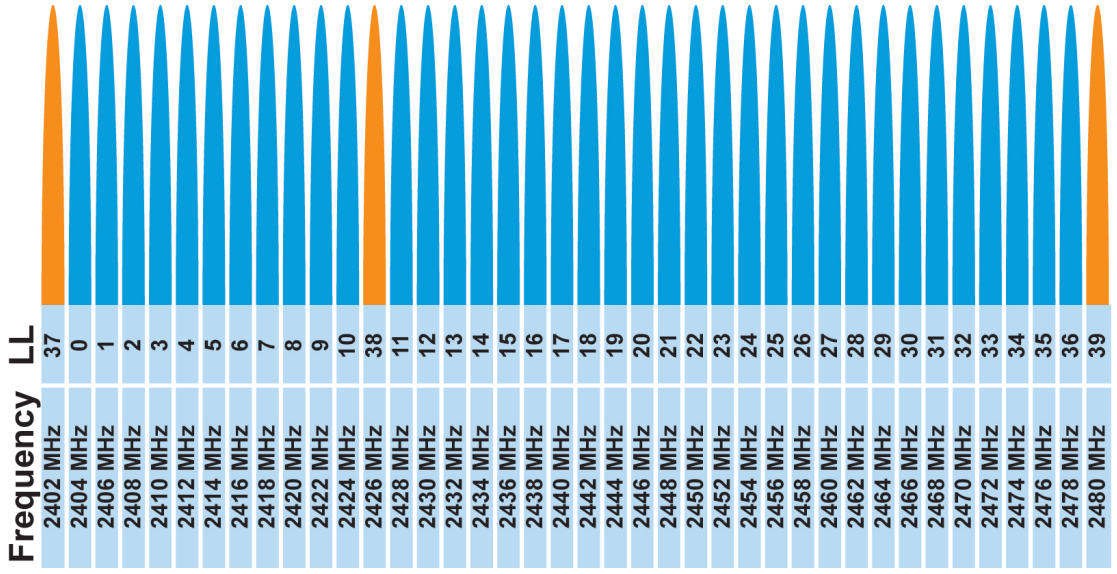


FIGURE 3.5: LE technology, like BR/EDR, features adaptive frequency hopping in order to secure a robust transmission even in harsh industrial environments[3]

The LE link layer is responsible for the maintenance of the physical link. LE uses a very simple link layer with only 5 states to maintain low silicon space and low power consumption. There are two communication roles defined at the LE link layer -

[3] http://www.connectblue.com/press/articles/shaping-the-wireless-future-with-low-energy-appl

the Master and the Slave. During an active connection, these devices can assume either *initiator* or *advertiser*. A device that transmits advertising packets is called an advertiser. The advertising process occurs through advertising channels which occurs in time intervals referred to as advertising events. In order for a device to be discoverable, an advertiser must transmit a short messages three times every few seconds (three times because three frequencies are used for robustness - a compromise between robustness and low power). A connection between devices is established through an asymmetric procedure by which an advertiser announces through the advertising channels that it is connectable. The initiator listens for such advertisements. When an initiator finds an advertiser, it may transmit a connection request to the advertiser, thus creating a point-to-point connection between the two devices.

In order to limit power consumption, slaves are in sleep mode by default and wake up periodically to listen for potential packet transmission from the master. The master determines when slaves are required to listen, and thus control access to the medium using a Time Division Multiple Access (TDMA) scheme [26].

The HCI interface provides communication capabilities between the Bluetooth Host and Controller (discussed previously in BR/EDR's Section 3.2.2). It is common for BR/EDR and LE. However, LE reuses the specification for BR/EDR and extends it with its own functionality. Similarly with the L2CAP layer, LE reuses the functionality utilised in BR/EDR and simplifies it with modifications to suit its own purposes - by supporting a limited number of Channel Identifiers.

While having borrowing some of the original Bluetooth functionality, LE incorporates some reductions in functionality such as no support for scatternet (only supports a single piconet). Through a piconet, LE allows a single master device to connect with up as many slaves as needed. Additionally, LE is not required to implement both transmitter and receiver - LE can be either or both.

3.3.3 Attribute Protocol

The Attribute Protocol (ATT) is a mechanism used for name discovery, service discovery, and reading and writing attributes of remote devices. As previously

mentioned, the behaviour of a Bluetooth connection is determined by the Bluetooth profiles a device has implemented. The ATT provides the means for interacting with *attributes* of a remote device. It follows a *client-server* model, in which the *server* publishes a set of attributes and a *client* can discover, read and write those attributes. The *server* is also capable of *notifying* or *indicating* the *client* about an updated attribute.

An *attribute* represents a piece of data and is made up of four parts. An example of an attribute might be a heart rate reading, a person's current temperature, whether or not a sensor is active, etc.

- Value - represents value of the attribute, e.g. heart rate in BPM
- Type - represents the attribute type so that a remote device knows how to gain access
- Handle - uniquely identifies an attribute
- Permissions - defines access (read/write/both)

Generic Attribute Profile (GATT) is a service framework for using Attribute Protocol for discovering services and reading/writing attribute/characteristic values 3.7. There is a short set of terminology required to understand how the GATT framework functions.

A **Client** is a device that initiates GATT commands and requests, and accepts responses (for example an Internet gateway device). A **Server** is a device that receives GATT commands and requests, and returns responses (for example an LE sensor). A **Server** (slave) is connected with a **Client** (master), but does not initiate any complex procedures as it is intended to be simplistic so as to ensure low cost, low energy, low power. Any complexity is left up to the Client. See Figure 3.6 for example of GATT interaction.

[Note: Some terminology commonly used: Master ↔ Slave, Client ↔ Server, Peripheral ↔ Central]

A **Characteristic** refers to a data value transferred between client and server (a patient's heart rate, body temperature etc.). A **Service** represents a collection of related characteristics, which operate together to perform a particular function

(for instance an ECG service, supplying various information concerning a patient's heart measurement) - see Figure 3.7. Finally, a **descriptor** provides additional information about a characteristic. All services, characteristics, and descriptors are identified by a UUID.

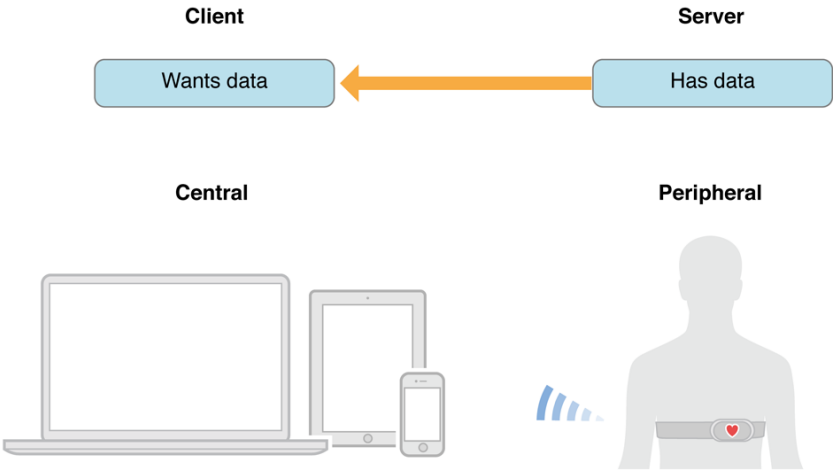


FIGURE 3.6: A GATT server transmitting heart rate information to a GATT client[4]

There are multiple options available for a client to access the attributes of a server:

Write Request → Request the server to write the value of an attribute and send an acknowledgement as a response. This command requires an attribute handle and attribute value as parameters. (The response from the server is known as a Write Response)

Write Command → Similar to Write Request but does not make use of acknowledgements

Notifications → The client may request a notification for a particular characteristic from the server. The server can then send the value to the client whenever it becomes available and will periodically transmit to the client when the value changes. This avoids the need for the client to poll the server, which would require the server's radio circuitry to be constantly operational. This command requires an attribute handle and attribute value as parameters.

[4] https://developer.apple.com/library/ios/documentation/NetworkingInternetWeb/Conceptual/CoreBluetooth_concepts/CoreBluetoothOverview/CoreBluetoothOverview.html

Indications → An indication is similar to a notification, except that it requires a response from the client, acknowledging that it has received the message. This command also requires an attribute handle and attribute value as parameters.

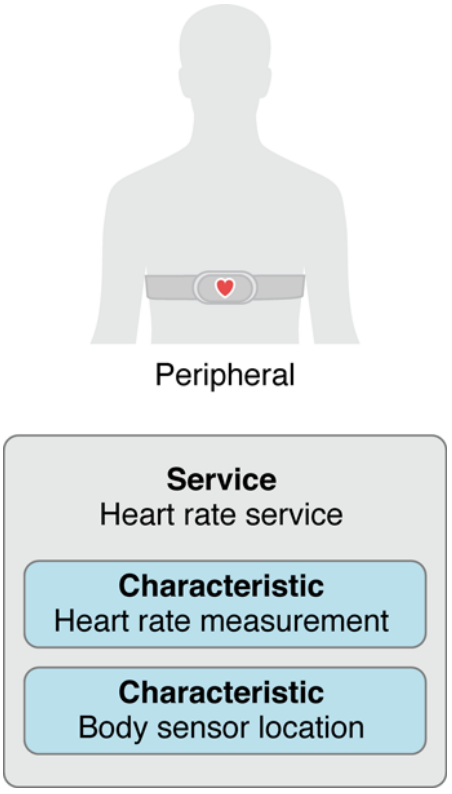


FIGURE 3.7: GATT architecture[5]

Bluetooth SIG provide listings for generic GATT services to enable the development of interoperable Bluetooth products. Here is an example of the Blood Pressure GATT profile:

```
<!-- Copyright 2011 Bluetooth SIG, Inc. All rights reserved. -->
<Profile name="Blood Pressure">
    <Role name="Blood Pressure Sensor">
        <Service type="service.blood_pressure">
            <Declaration>Primary</Declaration>
            <Requirement>Mandatory</Requirement>
        </Service>
        <Service type="service.device_information">
            <Declaration>PrimarySingleInstance</Declaration>
            <Requirement>Mandatory</Requirement>
            <Characteristic type="characteristic.manufacturer_name_string">
                <Requirement>Mandatory</Requirement>
            </Characteristic>
            <Characteristic type="characteristic.model_number_string">
                <Requirement>Mandatory</Requirement>
            </Characteristic>
            <Characteristic type="characteristic.system_id">
            <Requirement>Optional</Requirement>
```

```
            </Characteristic>
        </Service>
    </Role>
    <Role name="Collector">
        <Client type="service.blood_pressure">
            <Requirement>Mandatory</Requirement>
        </Client>
        <Client type="service.device_information">
            <Requirement>Optional</Requirement>
        </Client>
    </Role>
</Profile>
```

3.3.4 Security

All LE security functionality is provided through the Security Manager (SM) - pairing, authentication, and encryption. LE borrows terminology from the BR/EDR protocol and also some of its security mechanisms. LE uses the same pairing process that is used for the Secure Simple Pairing (SSP) in BR/EDR.

Authentication occurs during initial pairing process. This involves entering a pass key into one or both devices (similar to BR/EDR). Devices are said to be bonded when shared secrets are stored. When reconnecting with a device having previously bonded with, a signed command is sent to the device to authenticate that it knows the shared secret that was used previously. In order to prevent replay attacks, part of the command must be a counter that is incremented for each message sent. After reconnecting, either party can initiate encryption where for each packet sent a message integrity check (MIC) is included in order to authenticate the sender.

For encryption, LE symmetric cryptography - AES-128 is used to provide keys, to encrypt and provide integrity checks. There are five main cryptographic keys used:

1. Temporary Key - used during the pairing procedure
2. Short-Term Key - used as an encryption key the first time the pairing process occurs. The Short-Term Key consists of the Temporary Key and two random numbers (contributed by both Slave and Master to provide further security)

[5] https://developer.apple.com/library/ios/documentation/NetworkingInternetWeb/Conceptual/CoreBluetooth_concepts/CoreBluetoothOverview/CoreBluetoothOverview.html

	SingleMode	*DualMode*	*Classic*
Single Mode	LE	LE	none
Dual Mode	LE	Classic	Classic
Classic	none	Classic	Classic

TABLE 3.2: Bluetooth version compatibility [2]

3. Long-Term Key - used to encrypt the link in circumstances where a device is reconnected to a previously paired and bonded device

4. Identity Resolving Key - used to provide a device (who knows a peer device's Identity Resolving Key) with the ability to resolve a device's identity

5. Connection Signature Resolving Key - used to give the receiver the ability to use the signature to authenticate the sender of the message

In LE, every connection that is made uses a different signature - that has no correlation to any identifying information of the communicating devices. It is not possible to identify through packet sniffing. In terms of error detection, CRC is used to protect against all 1, 2, 3, 4, 5 and all odd bit errors.

3.4 Bluetooth Smart Ready

Bluetooth Smart Ready devices feature a dual mode radio, which supports both BR/EDR and LE wireless connections. A Bluetooth Smart Ready device provides the functionality of a peripheral and also is capable of initiating a connection, and again is able to communicate with all older versions of Bluetooth. However for single mode devices, both LE and BR/EDR, there are some compatibility issues (see Table 3.2).

As discussed in the previous sections, architectures for both BR/EDR and LE are similar. Thus, in a dual mode configuration, the implementation of some of the Bluetooth layers can be shared (see Figure 3.8).

The previous sections have shown that both Bluetooth 4.0 technologies are two very different protocols with different capabilities and use cases. At its most simplest, BR/EDR is aimed at episodic data transmission whilst LE is tailored towards episodic communication. When both technologies are combined in a dual-mode

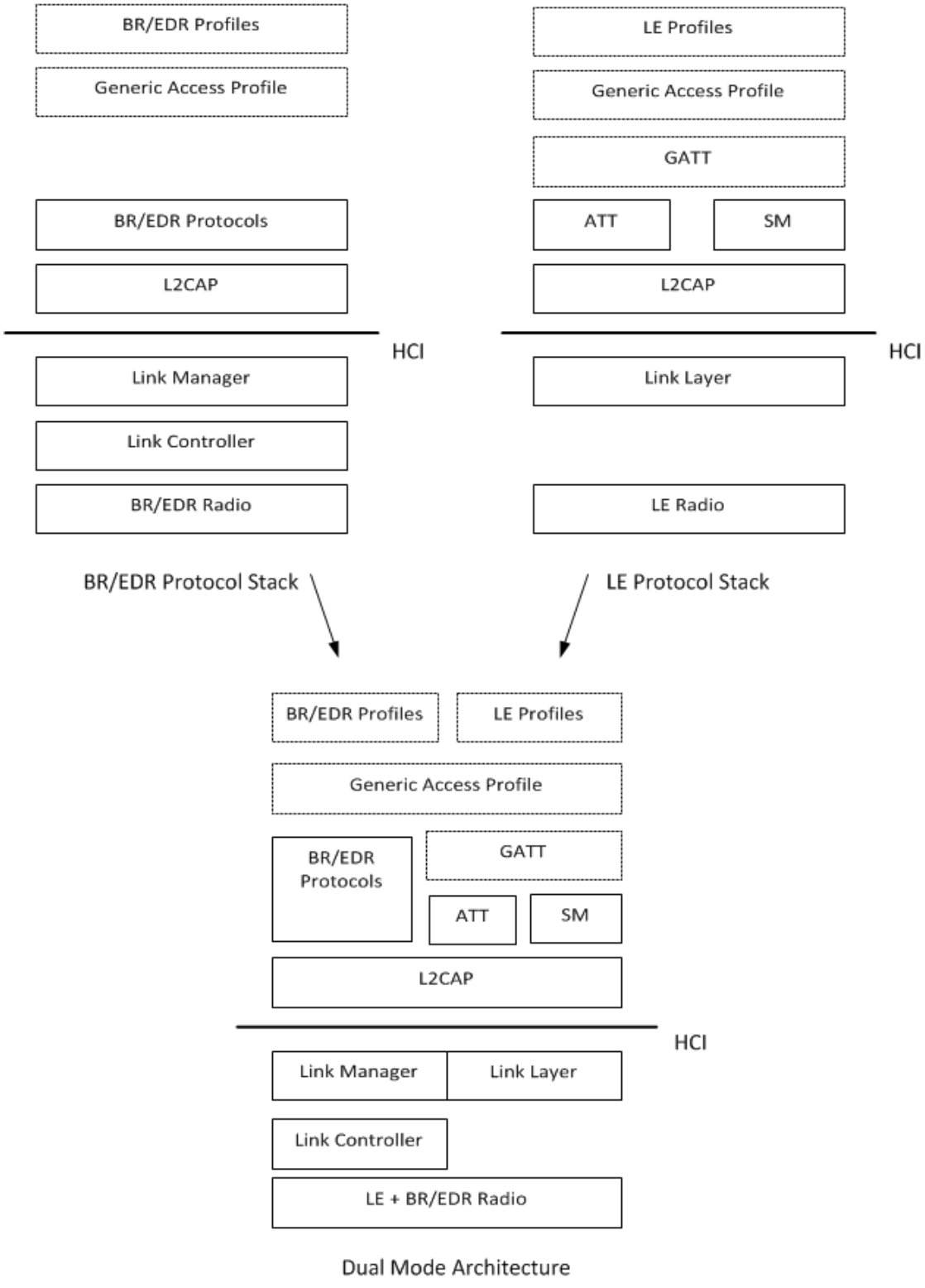


FIGURE 3.8: A high level view of the different Bluetooth stacks combined to make the dual mode architecture (Bluetooth Smart Ready) [2]

implementation, one can for instance, in parallel, connect a number of single-mode BR/EDR modules as well a number of single-mode LE modules to a Bluetooth

dual-mode module. This configuration has scope for some interesting health care use cases.

- Sensors requiring high data rate streams can be transmitted over BR/EDR (ECG, EMG, etc.) and low data rate over LE (heart rate, body temperature, etc.).
- To maintain low power consumption, Bluetooth Low Energy can be used in standby mode
- In a situation of emergency, the most pertinent information is transmitted over BR/EDR. This ensures that caregivers have the most up to date information as quickly as possible.

A dual mode module can use both mediums to receive information - SPP for continuous streaming data and episodic data transferred via GATT. This idea forms the basis of a smart sensor that is capable of communicating through multi bit rate channels.

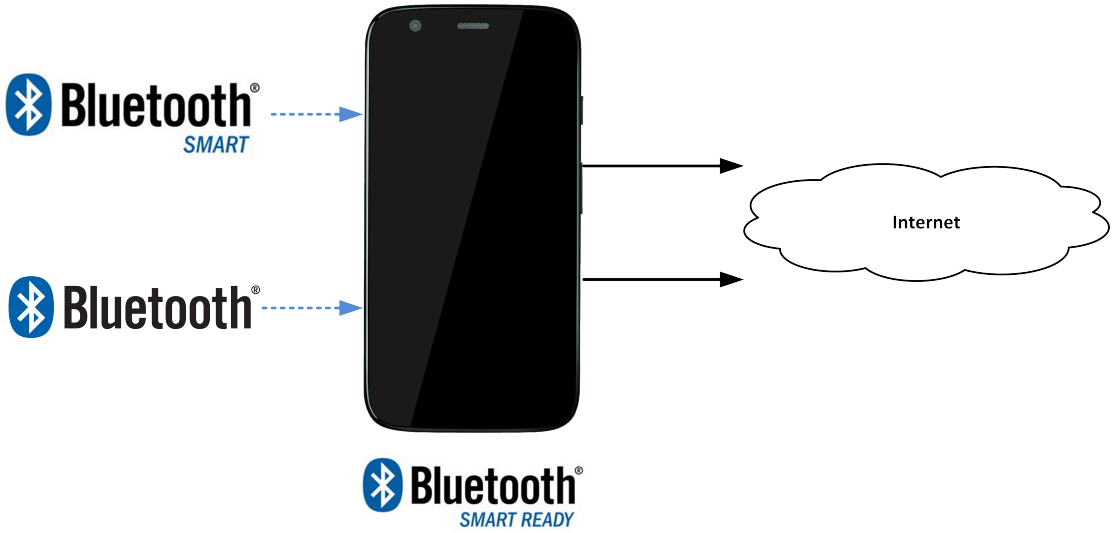


FIGURE 3.9: High level view of Bluetooth Smart Ready gateway being used to expose Bluetooth enabled sensors to the wider Internet

3.5 Summary

This chapter covered both technologies of the Bluetooth 4.0 protocol. Both Bluetooth Classic and Bluetooth Low Energy were detailed to establish the basis for

a prototype that can be used to combine both protocols capabilities under a Telemedicine platform. This chapter has established the concept of a multi-channel Bluetooth specification that serves as the foundation of the system design, which is covered in Chapter 4.

Chapter 4

Design

This chapter covers the design of a Telemedicine system that is used to provide remote palliative care of patients. The chapter outlines the use of profile driven medical sensors that make use of a gateway to publish information to a cloud based service. The cloud service provides access to current and historic data of patients and forms the basis for a decision support system offering granular control of the patient sensor network. Security is also an important concern for the system. In this chapter, two models are discussed.

This chapter begins with a high level design of the system. The following sections will introduce each component of multi-tier architecture, the communication protocol and the security mechanisms used to demonstrate a proof of concept for this dissertation.

4.1 System Overview

The system can be considered as three distinct parts - A sensor network, Internet gateway and Cloud application (see Figure 4.1).

The **sensor network** is a WBAN that consists of non-invasive wearable/implantable Bluetooth 4.0 enabled sensors. These sensors can be Bluetooth Smart or Bluetooth Smart Ready devices. Using a customised GATT profile model, each sensor can be made accessible to the cloud through a gateway. Each sensor is independent and communicates with the gateway via Bluetooth when required.

The **Internet gateway** operates as a Bluetooth Smart Ready hub that routes individual sensor data streams to the cloud once initiated by the gateway. As a dual mode device, the gateway is capable of receiving both high (BR/EDR) and low (LE) bit rate data streams concurrently. In order to control the functionality of the sensors and the data streams, the gateway operates over a low bit rate channel once instructed by the Cloud application.

The **Cloud application** serves as a central service that provides secure access to current and historic patient information and leverages control over the operation of the sensor network, which is made accessible through user accounts.

In order to secure all transmission and storage of patient data, two security models are proposed - both of which combine the strengths of symmetric and asymmetric cryptography.

4.2 Communication Protocol

As can be seen from Figure 4.1, all tiers of the system are connected via communication links. This requires the design of a message protocol to communicate from the sensor network all the way to the cloud and vice versa. All communication is initiated from the cloud application and is thus controlled by the user. These commands are used to control the current operation of the sensor network. Through these commands the user is capable of:

- Controlling what sensors transmit over the *high* and *low data rate streams*
- Controlling what sensors are currently active or inactive

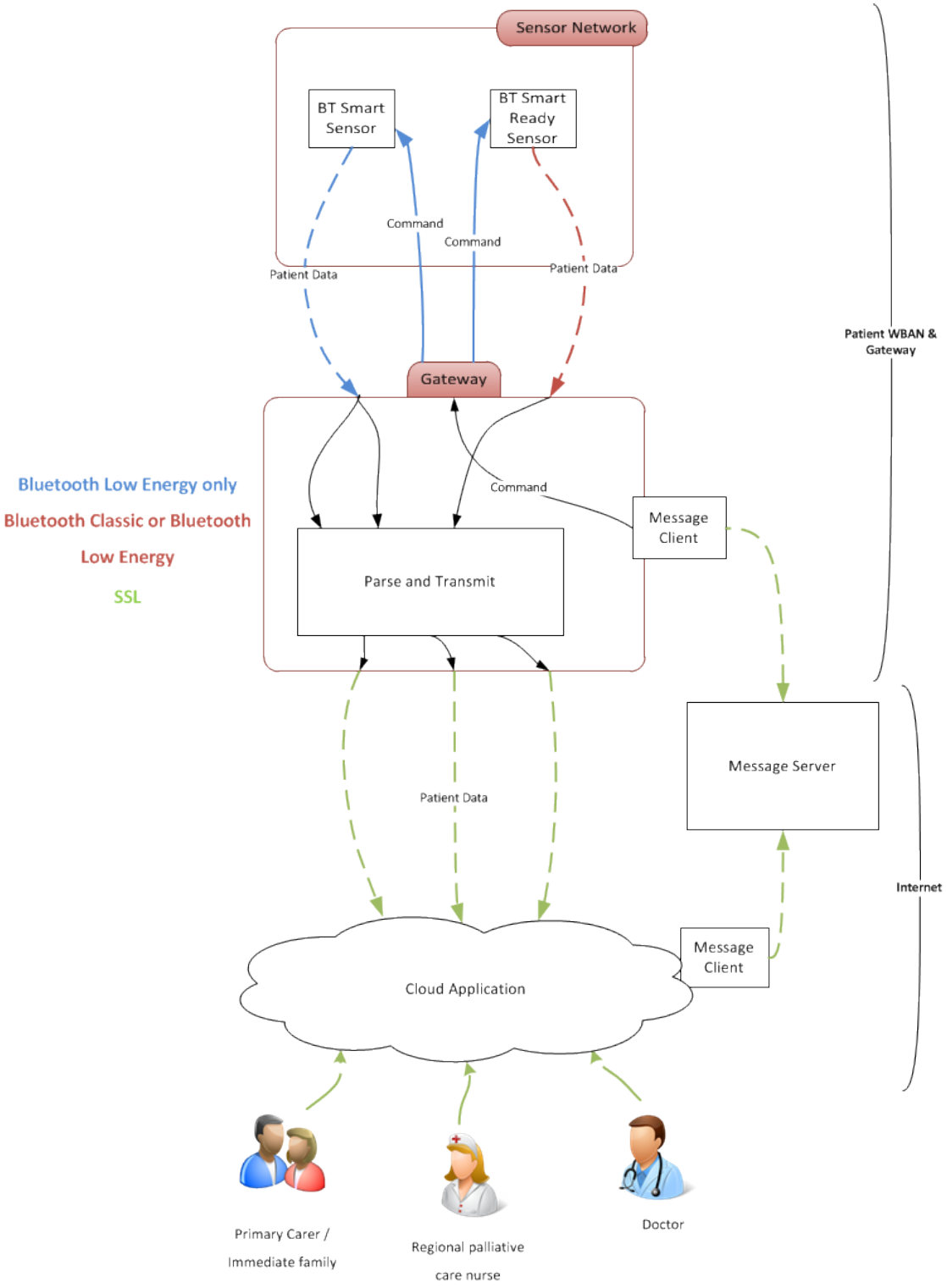


FIGURE 4.1: Sensors ↔ Bluetooth Smart Ready ↔ Internet Gateway

4.2.1 Packet Design

In order to provide the sensor control requirements the following information is required:

1. What sensors to select
2. Whether to activate or deactivate a particular sensor
3. Use of BR/EDR (high bit rate) or LE (low bit rate) stream for a specified sensor

The easiest way to specify an individual sensor is by its Bluetooth characteristic handle*. [*Note: BR/EDR capable sensors do not feature the concept of handles as they are terms specific to the LE protocol - however as the sensor network's control messages are sent using LE, each Bluetooth Smart Ready sensor is assigned a flag in its GATT profile to control the stream. See Section 3.3.3*]. As items 2 and 3 can each be represented by a Boolean value, a single digit is used where *0* denotes *false* and *1* denotes *true*.

* This is defined by the GATT attribute handle of the sensor measurement value. Due to adoption of generic Bluetooth profiles by SIG - discussed in Section 3.3.3 - characteristic handles may not be unique across multiple patients of the system. Thus in order to successfully identify a patient stream at a gateway level, more information may be required. This design assumes handles are unique.

In order to separate the different sections of a message:

SEPARATOR (':') to separate the different pieces of information in a message

TERMINATOR (',') to concatenate multiple commands into a single message

END *(';')* to denote the end of a command sequence

A single command will therefore take the form:
Handle + *SEPARATOR* + ***Active*** + *SEPARATOR* + ***HighDataStreamActive*** + *TERMINATOR* + *END*

For example: *0x0018:1:0:,;*

The application user also has the option to specify multi-commands to transmit at once. For example: *0x0018:1:1:,0x0024:1:0:,:*. This allows for greater network throughput.

Handle	*Active*	*HighDataStream*
0x0018	1	1
0x0024	0	0

TABLE 4.1: Examples of sensor command packets

4.2.2 Cloud → Gateway

Once a message is formed on the cloud server through the application interface, it is transmitted to the gateway via a messaging server. The gateway, like the cloud application, acting as a client of the messaging server will be able to receive each command. This channel between the cloud server and gateway is used as a unidirectional communication pipe. In order for the gateway to initiate communication with the cloud, it instead uses HTTP.

In order to register with the messaging server, the gateway must have a pre-existing "profile". This consists of login credentials (username and password) that must be configured with the server. The cloud must also complete registration with the messaging server in the same way. The username and password used for access to the cloud application is reused for the messaging server. Thus, every time a user logs in to the application, they are automatically signed in to the messaging server. As the gateway is also a user of the cloud application, the process is the same.

To allow communication to occur between a gateway and the cloud application user, they must both agree on profiles they are willing to communicate with. This is to ensure that no user is transmitting to a non intended gateway. This step requires the knowledge of a client's username (email address). A client can then request communication access to another client by supplying the username to the messaging server. To grant this request, the requested client can accept, thus allowing correspondence between both parties.

In the event of conflicting sensor network commands originating from the different users at once, the messaging server can alert both users of the occurrence. The messaging server can keep maintain a record of each last received control message

for each patient. This information is timestamped so should any messages arrive in quick succession, the messaging server can detect a potential conflict and warn both users of the occurrence. These types of scenarios were considered out of scope.

4.2.3 Gateway → Sensor Network

When a command is received at the gateway, it is placed into a buffer. From here, the message is deconstructed into segments that can be rearranged to be configured as an LE command. For instance the incoming command *0x0024:1:0:,;* requests a sensor represented by handle *0x0024* to be *activated* and to communicate via the *low data stream*. In order to control the sensor network in a way that is efficient, has low overhead, all control messages are sent using LE. [Note: It is possible to send commands over the serial port - but this was changed in favour of a lower power approach. However, this is useful in order to support BR/EDR only devices.]

Once commands are received and parsed, the next step is to separate the BR/ER commands from the LE commands because both require separate types of Bluetooth commands:

Low data rate requested → activate GATT notification command using the information provided by the cloud

High data rate requested → enable GATT high data rate characteristic (Write No Response)

In order to route the Bluetooth messages to their destination, the last piece of information required by the gateway is the MAC address of each sensor device. Therefore the gateway holds a mapping from sensor handle to each hardware address.

Handle	*MACAddress*
0x0018	00:11:22:33:44:55
0x0024	00:11:22:33:44:55
0x003A	11:22:33:44:55:66

TABLE 4.2: Each sensor handle is mapped to its corresponding hardware address

At this stage, all information required to exert control over the sensor network is available.

4.2.4 Sensor Network → Gateway

Each sensor stream is individually routed to the gateway once activated by the gateway. As previously mentioned, both Bluetooth protocols use different mechanisms to reach the gateway. BR/EDR communicates with gateway via a serial port and LE communication occurs using GATT. The gateway can receive both protocol streams concurrently.

In order to receive BR/EDR data, the gateway must be listening on the serial port, which buffers all received bytes until the packet's END (';') character. Given complete control over the protocol that occurs over the SPP, the sensor data messages are designed to be very simple. Their format is as follows:

HANDLE *:* ***DATA*** *;*

Each LE data packet is received through a GATT notification loop. This mechanism does not rely on the gateway polling the sensor as the sensor will transmit to the gateway whenever new data is available. Unlike the serial port, the LE protocol is abstracted (uses GATT, which does not offer as much protocol control as SPP) so the received commands must be parsed in a separate manner. (The specific format is discussed in Chapter 5's Section 5.2.2).

Rather than all communication originating from the cloud, sometimes it is useful for a patient to submit data manually. For instance in a situation of distress, a wearable panic button can be used by a patient to submit an alert to the cloud. Any button activity can be registered and transmitted in the sensor data packet format. The only information this would require is a way to uniquely identify the event on the cloud server. This could be achieved using a pre defined unique sensor handle. Receipt of this event on the cloud would signify a panic button alert. From this point, a decision can be made to alert the users of the abnormal sensor reading. For instance, send warning email to medical staff supplied with latest health readings and/or send a text message to the primary carer.

4.2.5 Gateway → Cloud

All data received at the gateway are parsed and repackaged as JavaScript Object Notation (JSON) and subsequently transmitted to the cloud application via HTTP POST. The handle is included in the POST URL as opposed to the message (the reasoning for this is explained in the security section - see Section 4.6). For example, if the gateway receives a heart rate reading, the gateway will create the JSON message:

```
[
    {
        "timestamp" : 1399135769,
        "data" : 72
    }
]
```

. . . and POST to: https:. . . /sensors/data/*heartRateHandle*

In the situation of server unavailability, the gateway can alternatively write the data packets to local storage until the server becomes available. On wake up, the gateway can replay these messages to ensure that no patient information is lost or unavailable at the cloud.

4.3 Sensor Architecture

The sensor network is made up of one or more Bluetooth Smart/Smart Ready enabled sensor devices. Each sensor device can be any vital sign measuring device such as ECG, EMG, heart rate, temperature, etc. that is capable of providing a digital output. Each sensor device is an individual endpoint and transmits as an independent stream to the gateway device.

There are two potential models for each sensor that participates within the sensor network. The chosen model depends on their Bluetooth capabilities and requirements. Sensors that require only episodic, low-bit rate data only require LE while sensors that require higher bit-rate streams would use BR/EDR. All sensor functionality must be exposed to the gateway through a GATT profile in order to successfully control the network. This section will illustrate the different models used for different Bluetooth sensors.

4.3.1 Generic Sensor Model

For every Bluetooth Smart and Bluetooth Smart Ready device participating in the network, a simple model profile is agreed upon. As LE is the common technology for both Bluetooth Smart and Bluetooth Smart Ready devices, the profiles take the form of a GATT profile. This provides compatability for sensors that use only LE or both LE and BR/EDR. For a single mode classic device, it is possible to initiate communication from the gateway through serial port. However, this requires more complexity at the sensor microcontroller.

```
<profile type="SmartReady">
...
    <service name="MedicalSensor" uuid="abcdef">
        <characteristic name="MedicalSensorReading" uuid="12345"></characteristic>

        <characteristic name="HighDataRateStreamControl" uuid="23456">
        </characteristic>
        <characteristic name="" uuid=""></characteristic>
    </service>
...
</profile>
<profile type="Smart">
...
    <service name="MedicalSensor" uuid="abcdef">
        <characteristic name="MedicalSensorReading" uuid="12345"></characteristic>

        <characteristic name="" uuid=""></characteristic>
    </service>
...
</profile>
```

MedicalSensorReading - A notification characteristic which represents the control of the LE stream.

HighDataRateStreamControl - A Write No Response characteristic which facilitates control over the high and low data streams.

It is also possible to add other characteristics to add more behaviour to each sensor. For instance, if more granular timing control of sensors were required, a characteristic could be added to denote the period of delay between each sensor reading transmission over the low bit rate channel. This might be useful in situations such as a doctor requiring hourly updates of a patient's symptoms.

4.4 Gateway

The gateway is intended to function as a simple interoperable lightweight device with a sole responsibility of operating as an intermediary between the cloud and the sensor network. This means that all Bluetooth messages are communicated to the cloud via the gateway, and all sensor network commands are communicated to the sensors via the gateway.

The gateway is designed to be interoperable so that it can function as a gateway to any patient of the system. This means that the gateway requires the following:

- A mapping of each sensor handle to its device's MAC address - this is to ensure that for each cloud command that arrives, the gateway knows what device to initiate communication with.
- The correct serial port to use to connect with BR/EDR capable sensors
- What cloud application users to receive commands from

The gateway is designed to be as portable as possible, thus requiring low power consumption. In order to help achieve this:

No complexity in the software design - No complicated processing step required to route data or commands.

LE control channel - LE is used as the sensor network control channel

Use of GATT notifications - In LE terms, the gateway acts as a client and each sensor is an individual server. All LE communication occurs over the air through GATT notifications. The onus is therefore on each subscribed sensor to send all new patient data. This removes the need for the gateway to poll individual sensors.

Buffering - All BR/EDR data received at the gateway is buffered to optimise performance (see Section below).

As both Bluetooth protocols are inherently different by design and in terms of their transmission capabilities, their respective sensor streams require separate processing. For BR/EDR, all serial received data is placed in a fixed sized queue.

This is necessary in order to optimise the stream as the throughput of BR/EDR is much higher compared to LE, which would cause a bottleneck at the gateway. Once the queue has reached full capacity, the queue is flushed of its contents, where each packet is then combined into a single message and submitted via HTTP to the cloud server. The interaction with the cloud server is designed to asynchronous so as not to slow down the serial channel events.

The gateway functions as a user of the cloud application. However it differs from other users in that it has its own set of permissions and roles which permits access the cloud. With access to the cloud, the gateway can submit data streams that are received, for each patient. There is no user interaction designed directly with the gateway as its primary purpose is to act as a mobile intermediary between the cloud and sensor network. It can therefore be run as a background service on a smartphone device. A patient can go about their life whilst being monitored seamlessly in the background. This is an important focus of palliative care.

In order to initially configure the gateway of the system to establish a *white list* of allowed devices, it must be capable of scanning for nearby devices, reading advertisement data and/or profile information. This information could be provided to the cloud application, where a user can then determine what sensors to associate with the patient.

In the instance of multiple gateways, for example in the context of a nursing home environment where multiple patients are within range of multiple gateways, a stream may be accepted by two or more gateways - leading to wasted bandwidth and data duplication in the database. This type of scenario is considered as future work (see Section 7.1).

4.5 Cloud application

The cloud application acts as a central hub of control and access to the system. It functions as a rich feature-driven application, providing the user with multiple capabilities to monitor and deliver care for palliative care patients. This is provided through a central repository for historic and live patient information, which is made easily accessible to the patient's carer, medical professionals and any immediate family. Being accessible through the Internet gives caregivers easy access

to important patient information in order to ensure proper patient care and management. The cloud application also offers control of patient worn sensors. This control can also be applied automatically by the application logic. This proves useful in ambulatory situations, where for example a low data stream sensor has transmitted an abnormally high/low reading. As a result, the cloud can automatically decide to dynamically enable that sensor over the high bit rate channel, providing care-givers with the most up to date information.

The cloud application offers:

Control - Determine the operation of the sensor network in real time

Analysis - Viewing and sharing of real time or past patient data (EHR) with medical staff

Security - Secure access to application and provision of confidentiality

Decision support - Customisable events that are triggered when certain criteria is met IE abnormal sensor thresholds reached

Access - Multiple permissions providing granular control over all aspects of patient care - combined with a capability of adding/revoking permissions

As the application is the sole way of providing the user with access to the system, it is designed to be UI centric. This inspired the use of the Model View Controller (MVC) software design. MVC is a software pattern used in UI-driven applications. The pattern is made up of three parts:

- Model → Manages the behaviour and data of the application domain, responds to requests for information about its state (usually from the view), and responds to instructions to change state (usually from the controller)
- View → Manages the display of information
- Controller → Interprets the mouse and keyboard inputs from the user, informing the model and/or the view to change as appropriate

Access to the cloud application is granted through user profiles. A user is required to sign up - by entering a username and password pair, along with some patient

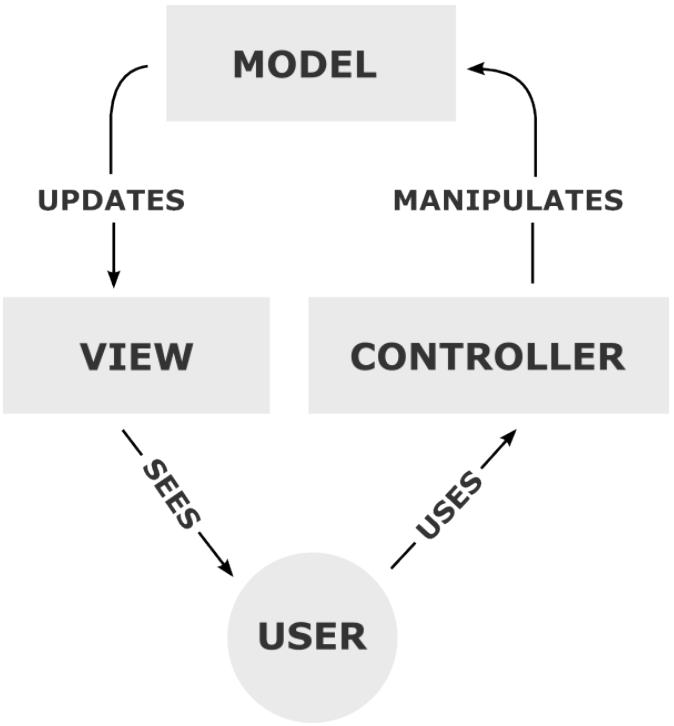


FIGURE 4.2: Model View Controller design[1]

information. This process creates a primary carer account. To create further accounts of different profiles (family, nurse, etc.), they can be created from the primary carer account. Therefore initial full access to the application is granted to only the patient's carer. Any further access is granted through leases. For instance a palliative care team may be given access for a certain time frame, whereas for example the oncology consultant may have 24/7 access.

When receiving patient data on the cloud as JSON, it is immediately stored in the database. In order to allow the user to visualise patient data, a graphing UI tool is used to plot health information requested by the user. To capture a real time physiological snapshot, data is periodically fetched from the database and updated on the view.

4.6 Security

This section details the provision of security mechanisms in the system. Two security models are outlined, which provide methods for authentication and for ensuring the privacy and integrity of patient data.

[1] http://en.wikipedia.org/wiki/File:MVC-Process.png

Security at Sensor level

Other than the level of protection already provided by both Bluetooth protocols, this dissertation does not contribute any extra security features at a sensor network level.

Security at Gateway and Cloud levels

There are two separate approaches taken towards the system's security at the gateway and application layers. Both approaches provide a trade-off between data protection and application fragility:

1. **Trusted server model** - Symmetric and Asymmetric keys will exist in memory on the server when required for encryption/decryption purposes, but the patient data store is encrypted at all times
2. **Host proof model** - No trust is placed in the server and therefore all secret keys exist only on the client. Like the previous model, all patient data is stored in encrypted form

This section will begin by introducing some of the technologies used in both approaches.

4.6.1 Symmetric Key Cryptography

Symmetric encryption is a type of encryption where data can be encrypted and decrypted using the same key. The key acts as a shared secret between two or more parties, meaning that all communicating parties must possess the same key, thus requiring an initial exchange of the key. This exchange can be achieved using another form of cryptography known as asymmetric key cryptography.

4.6.2 Asymmetric Key Cryptography

Asymmetric encryption is a type of encryption that relies on the existence of two keys - a public and a private key pair. Both keys are mathematically related, however it is computationally infeasible for a private key to be determined from its corresponding public key due to the difficulty of factoring large integers.

[2] http://www.ia.nato.int/images/nia/pki/smime-01.jpg

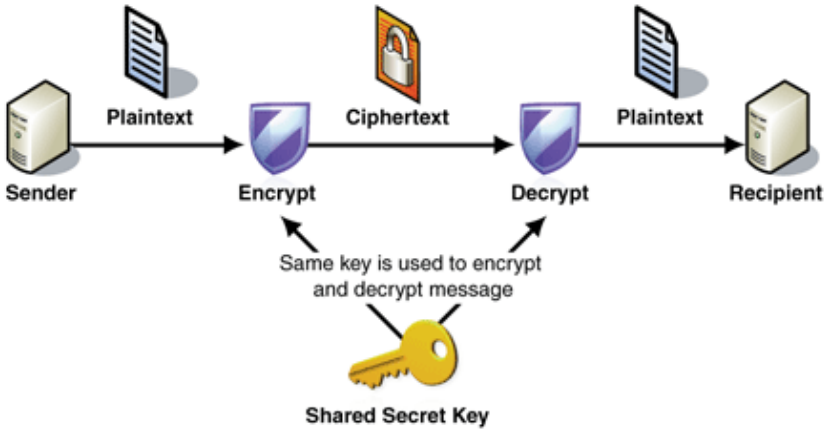


FIGURE 4.3: Simple illustration of symmetric key cryptography[2]

The public key is a non-secret key which can be made publicly accessible without compromising security. In contrast, the private key is a secret key and should be not be shared or made public. Unlike symmetric key cryptography, asymmetric key cryptography does not require all communicating parties to possess a shared secret.

For example, if Alice wants to talk to Bob and both possess an asymmetric key pair, then Alice uses Bob's public key to encrypt the message. Bob then uses his private key to decrypt Alice's message (see Figure 4.4). This approach can be used in reverse to achieve digital signatures to provide message authentication.

4.6.3 Hypertext Transfer Protocol Secure

Hypertext Transfer Protocol Secure (HTTPS) refers to the use of the Hypertext Transfer Protocol (HTTP) on top of the Secure Sockets Layer (SSL)/Transport Layer Security (TLS) protocol. The aim is to prevent wiretapping and man-in-the-middle attacks. It is typically used to provide an encrypted session between a server and a client, most typically a web server and a browser.

Protocol: The server transmits a copy of its asymmetric public key to the client. Once received, the client creates a symmetric key, encrypts it with the server's public key and transmits the result to the server. The server decrypts the data with its asymmetric private key to retrieve the symmetric session key. Both server and client can now encrypt and decrypt all transmitted data with the symmetric

[3] http://www.networkworld.com/subnets/cisco/chapters/1587052423/graphics/02fig02.jpg

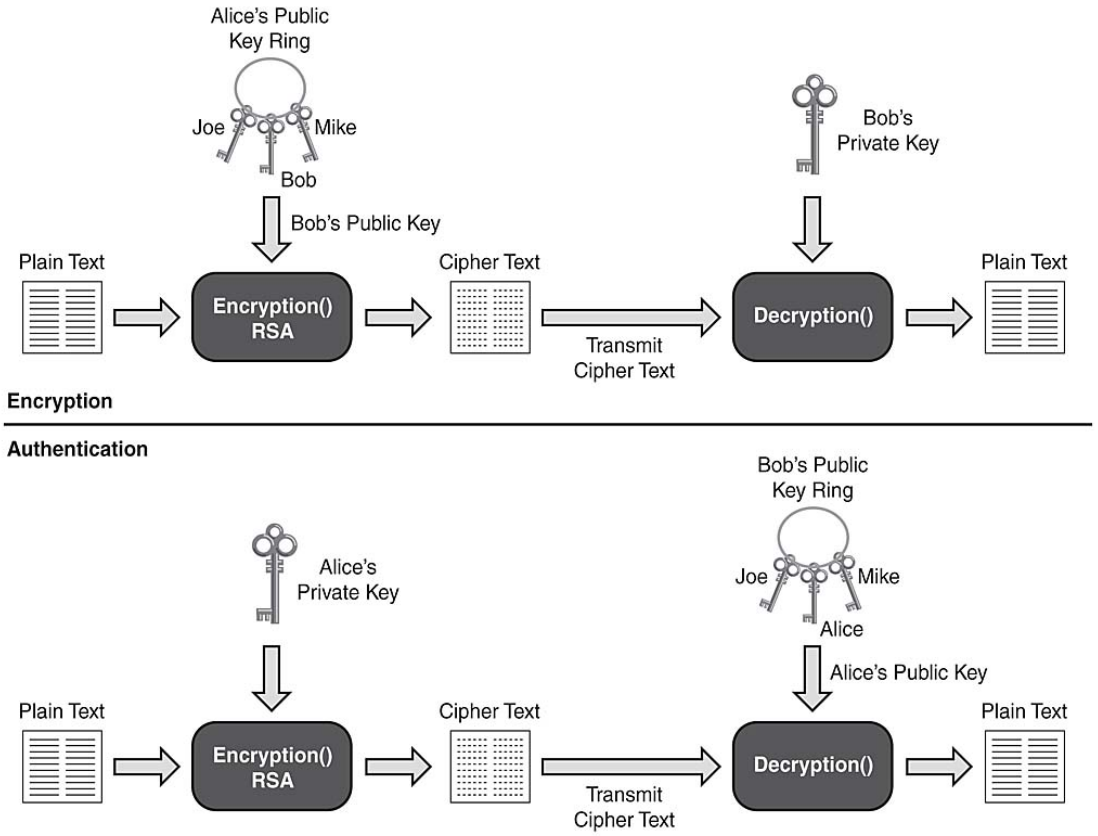


FIGURE 4.4: An example of encryption, decryption, and authentication using Asymmetric Key Cryptography[3]

session key. This allows for a secure channel because only the server and the client know the symmetric session key. The session key persists only for that session, where a new session key is created every time.

This protocol model is used as the basis for security in the system. The next section outlines the two different security models which can be used to enable a secure link between the gateway and the cloud server.

[Note: It is assumed that all interaction between the user's browser and the server occurs over HTTPS. This ensures that all traffic between the browser and the server is completely encrypted.]

4.6.4 Trusted Server model

This model requires both symmetric and asymmetric key cryptography. However, what makes it distinct from the other security model is that secret keys exist

in plain text for a time on the cloud server. This is because any non-encrypted value or waiting to be encrypted value is stored in memory in plain text, which poses potential security risks. For instance, if an attacker has physical access to the hosted machine they can potentially read keys directly out of memory. This model therefore requires trust in the cloud service to prevent any malicious attacks against the machine's RAM. In some instances, there are ways to protect the contents of memory. For example, in the .NET framework the SecureString class allows data to be encrypted in memory. However the key to decrypt will necessarily also be in memory, so at most it provides another level of obfuscation.

Protocol: The gateway possesses a symmetric key K. For each data stream received by the gateway, it is encrypted using K (the sensor handle is not encrypted because it is required in plain text at the server in order to successfully store the patient data). When this information is received at the cloud server, it is untouched and stored in its encrypted form. This means that even if the database is hijacked, no data is exposed.

To allow the patient data to be accessible from the cloud application, the cloud server requires access to the symmetric key K. In order to achieve this in a secure fashion, the cloud application user possesses asymmetric keys. To supply the cloud with symmetric key K the gateway acquires the public key and encrypts the symmetric key K with the public key and stores the result in the Cloud database. In order to view the patient data, the private key is used to decrypt the encrypted symmetric key, which is then used to decrypt the data.

The public key is stored in the database as it should be made publicly available to anybody who wants to communicate with the cloud. The private key is stored in a cloud configuration file, in a form that can only be decrypted by the owner (encrypted with the user's password). When a user is logged in, the private key is decrypted and stored in the user's session and persists only for the duration of the user's session. The user's session is safe from sniffing as long as the interaction occurs over HTTPS. To further mitigate the changes of a successful attack, the private key might only be stored in memory when it is required, rather than the duration of the user's session.

In order to provide secure communication from cloud to the gateway, each command is encrypted with the symmetric key K. Additionally, in order to guarantee authenticity at the gateway level, the cloud digitally signs each command. Once

received at the gateway, the command is authenticated by the decrypting it with cloud's public key. To gain access to the command data itself, the symmetric is used to further decrypt the data.

This model may present some security vulnerabilities, however it is important to note that it is not easily exploited:

- Different instances of the same application can easily have different keys, so a compromise of one will not automatically spread to all others.
- File system security on a web server is quite well-understood
- System compromises that allow full file system access statistically occur less than application or database break-ins

However the primary problem with this approach is that if the server is compromised in a way, the attacker is capable of reading any file. There is no way to fully prevent the attacker from accessing memory/application binary. The attacker can then perform offline dictionary attacks in order to uncover the private key. Similarly, the gateway stores its symmetric key K locally. As a local application, it is not accessible from network attacks. However, as the gateway encrypts all information with a single key, if the key is compromised, then the all data transmitted through the gateway is susceptible to attack.

Optionally, security can be provided on a per sensor or per stream basis, thus requiring multiple symmetric keys. To provide security on a per sensor basis, the gateway generates a key, which is then associated with a specific sensor handle. The gateway can then access each key, by using the sensor handle contained in each received sensor data packet. Alternatively, the sensor device itself may encrypt its own information before transmitting. While this would provide end to end encryption from each sensor to the cloud, it would also mean that an original exchange of each symmetric key with the cloud would be necessary. If a different key was used per stream, the gateway would require as many symmetric keys as there are streams. However, the remaining limiting factor is the lack of a reliable key store.

4.6.5 Host-Proof model

A host-proof application is an application in which the user is not required to trust the host. All user data is encrypted and decrypted on the client side, and the server only ever has access data in its encrypted form. This provides the benefit of a cloud application (accessible from anywhere) while retaining the security of only having access to one's own personal information. This means that even if the server is compromised, all data is completely secured as no secret information exists on the server.

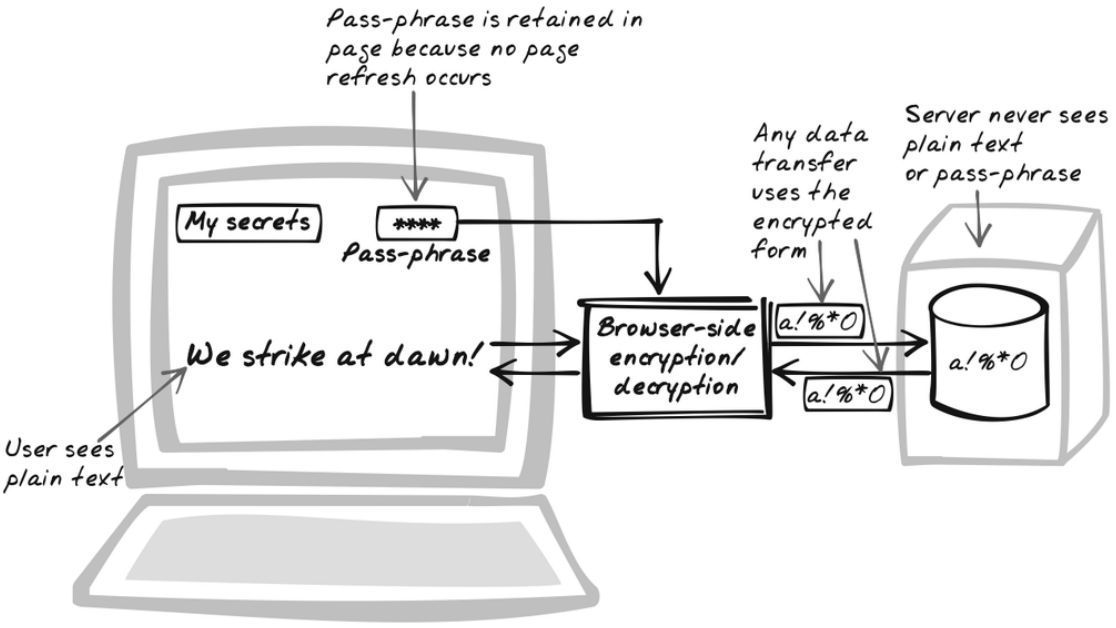


FIGURE 4.5: Host proof design model - Information is stored in encrypted form and all data decryption occurs locally[4]

Typically a user navigates around a website by clicking on hypertext links which instruct the browser to load a new page. However the new page doesn't remember any of the state of the previous page. It is therefore the server's responsibility to maintain state through the use of cookies and the user session. For a host-proof application this presents some issues as the users initial password is usually used as the secret key to the initial encryption and decryption. In order to work in a typical fashion, the user would need to re-enter their password after every page load, making the system very cumbersome to use. To combat this the user is prevented from changing pages and instead XML HTTP Requests (AJAX) are used to fetch the next page's content from the server. This way the secret details

[4] http://librairie.immateriel.fr/fr/read_book/9780596101800/ch17s03

can persist in a variable in JavaScript and at the same time allow the user to change pages.

Protocol: A user's public and private key are generated via the client. A further secret key is created by concatenating the user's username (email address) and password together, which is then run through a cryptographic hash function. This key is used to encrypt and decrypt the private key. The public key is stored in database and the private key is firstly encrypted with the secret key and then stored in the database. Both keys remain in memory in the client side JavaScript code.

Like the previous model, the gateway uses the user's public key to exchange the symmetric key K. When requesting access to patient data via the application, all data is returned to the client in encrypted form, along with the encrypted symmetric key. Using client side decryption with the user's private key, the symmetric key can be used to gain access to the patient data.

This approach offers the benefit of never exchanging secret information with the server. Any attacks on the server would be incapable of retrieving any private information in plain text. However the reliance of AJAX results in a fragile application. Although the technology is used to guide the user appropriately, there is no way to force a user to not use the browser functions. In this case, the application can break down with the loss of the user secrets, requiring the user to repeat the login process. As a result the application may be found cumbersome to use.

4.7 Summary

This chapter has introduced the system design by providing a detailed insight into how each tier of the system is composed and and how they are connected through a message protocol. Two security models were also outlined, capable of providing secure communication between the gateway and cloud application. The next chapter uses the design concepts presented in this chapter to demonstrate a proof of concept.

Chapter 5

Implementation

This chapter continues from the previous chapter by providing detail on how the system was taken from a design specification to a working prototype. This chapter covers specific technologies used to implement each piece of the system and discusses the rationale behind each choice. Figure 5.1 presents a full system overview, detailing the transition from design to implementation. See Figure 4.1 of the previous chapter for comparison.

5.1 Sensor Network

The sensor network is made up of one or more Bluetooth Smart/Smart Ready sensors, which host a customised GATT profile, allowing the gateway to control their functionality. To demonstrate the prototype, two faked sensors sources were used - a heart rate and ECG combination functioning through a dual mode device and a body temperature sensor acting as a single mode device. The heart rate is capable of communicating over the low bit rate channel and as the ECG requires higher throughput, it operates over the high bit rate channel. The body temperature sensor transmits only over the low bit rate channel.

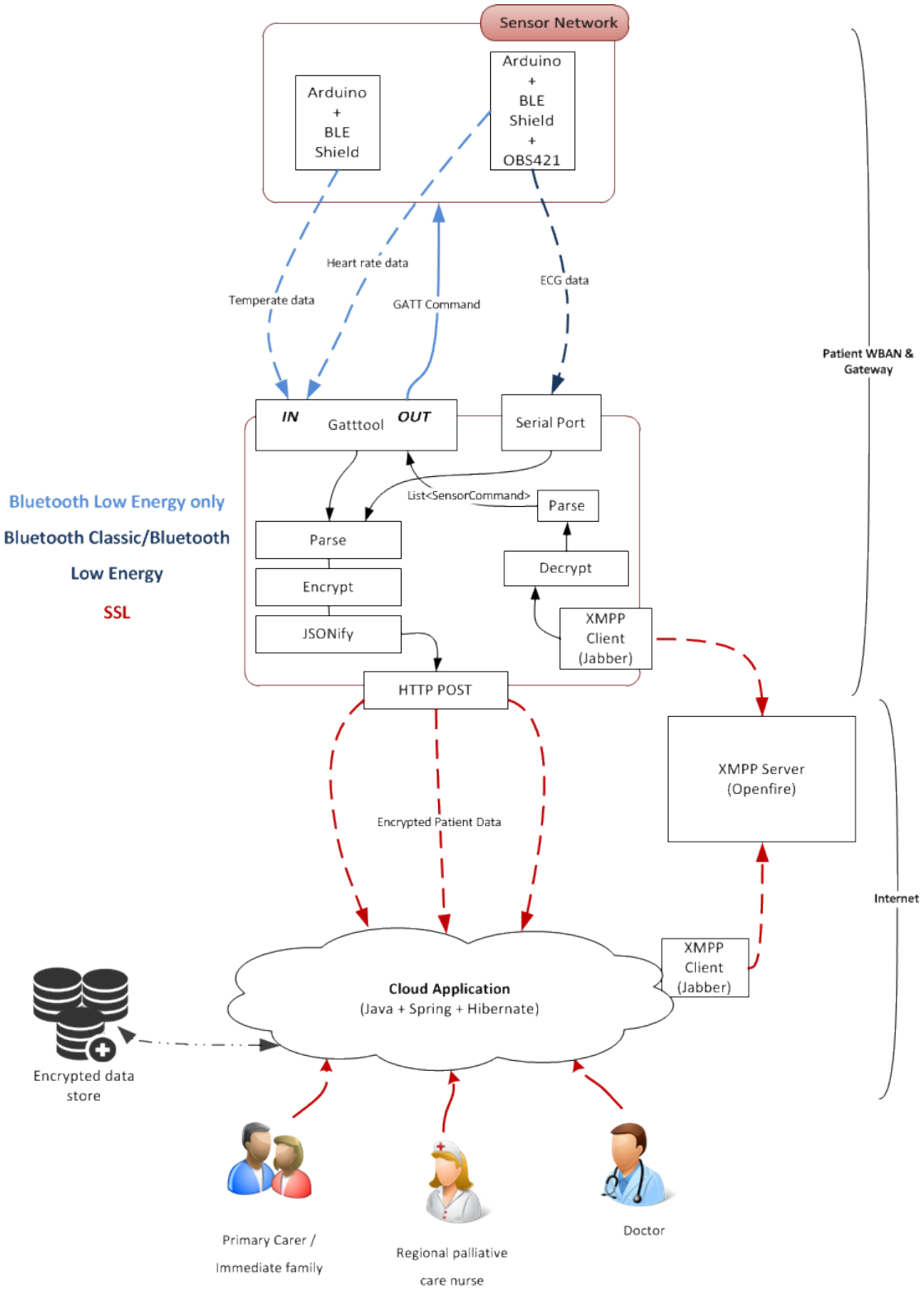


FIGURE 5.1: Full system implementation

5.1.1 Hardware

To emulate a medical sensor, a simple electronic prototype was built using the Arduino platform along with Bluetooth peripherals *. An Arduino Uno microcontroller combined with a Red Bear Lab BLE shield and a Connect Blue OBS421 BR/EDR module formed the basis of the sensor network acting as a Bluetooth Smart Ready sensor (combined heart rate and ECG). An additional Arduino Uno combined a Red Bear Lab shield was used to act as a Bluetooth Smart sensor (body temperature).

* Sensor data was emulated instead of using medical hardware as it meant faster prototyping.

Arduino Uno

Arduino is an open source electronics prototyping platform introduced in 2005. It was intended to provide a more accessible and inexpensive way to work with electronics as well as way to quickly prototype and test ideas. It is packaged with an Integrated Development Environment (IDE) in order to write software for the hardware in either C/C++. The Arduino Uno is one example of the Arduino family.

The Arduino microcontroller provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). These pins are referred to as the Hardware Serial pins. An ATmega16U2 on the board channels the serial communication over USB, which appears as a virtual com port to software on the computer. The Arduino software includes a serial port monitor which allows simple textual data to be sent to and from the Arduino board. This serial port monitor and its corresponding SoftwareSerial library provides all required BR/EDR functionality to send high bit rate data from a sensor to the gateway. However, in order to transmit low bit rate data over the LE channel, a separate tool is required. This tool is discussed in Section 5.1.2.

Connect Blue OBS421

The Connect Blue OBS421 is a dual mode Bluetooth Serial Port module. The module permits serial data to be sent using both BR/EDR and LE technologies

using BR/EDR's native SPP and Connect Blue's custom Low Energy Serial Port Service. However for the purposes of this research, full use of the standard LE protocol was required. This was instead achieved through an Arduino LE shield with an SDK, allowing full customisation of the GATT protocol.

Some features of the OBS421:

Wireless Multidrop™ - Simultaneous connections to both Bluetooth low energy and Classic Bluetooth devices *

Extended Data Mode™ - Separated multipoint data channels (different data can be sent to/received from each slave)

Throughput - Supports up to 1.3 Mbps

Simultaneous slaves - Maximum number of simultaneous slaves: 7 using BR/EDR only, 3 using BR/EDR and LE

* Initially, the OBS421 seemed to promise all required functionality to emulate a dual mode Bluetooth sensor. However, the inability to host custom GATT profiles meant a lack of information that was required to remotely control the sensors.

In order to configure the OBS421 to communicate successfully with the Arduino through the hardware serial pins, the module has to be connected to the Arduino through a series of wires (see Figure 5.5). Furthermore, the module's serial settings had to be configured in order to successfully communicate through the Arduino's hardware serial pins. This was achieved using Connect Blue's Toolbox Utility software. This software allows the device to be completely reconfigured by offering granular control of such things as the device's security, its advertisement data, and its discoverability options.

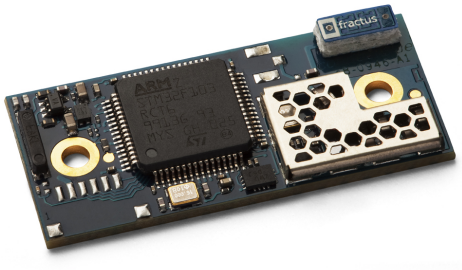


FIGURE 5.2: Connect Blue OBS421 module[1]

Red Bear Lab BLE Shield

The Red Bear Lab BLE shield* is a single mode module designed to work with the Arduino platform (see Figure 5.3). It is based off the Nordic nRF8001 integrated circuit (IC), which operates in slave (peripheral) mode only (see Figure 5.4). The nRF8001 features a simple serial interface referred to as the Application Controller Interface (ACI). This allows an application controller such as an Arduino Uno to communicate with the nRF8001 chip. The nRF8001 implements the lower stack layers, while a host microcontroller (Arduino) must implement the GATT services.

* A shield is a board that can be attached on top of an Arduino PCB to extend its capabilities. In this particular case, the Red Bear Lab shield offers LE functionality.

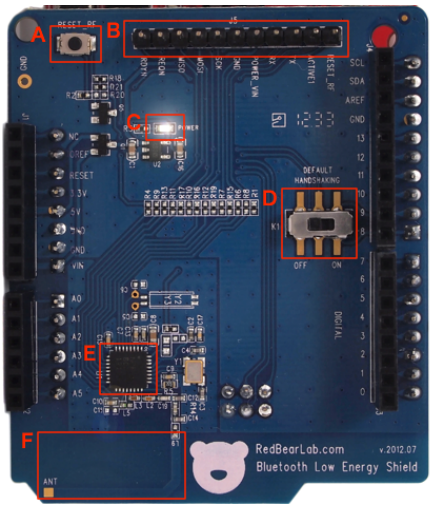


FIGURE 5.3: Red Bear Lab BLE Shield[2]

A → nRF8001 Reset Button

B → Factory Testing Pins

C → Power On LED

D → Default Handshaking Switch

E → Nordic nRF8001

F → Antenna

[1] http://www.connectblue.com/fileadmin/Connectblue/Web2006/Images/Press_Image_downloads/cB-OBS421-antenna-options.jpg

[2] http://redbearlab.com/

[3] http://www.nordicsemi.com/eng/Products/Bluetooth-R-low-energy/nRF8001

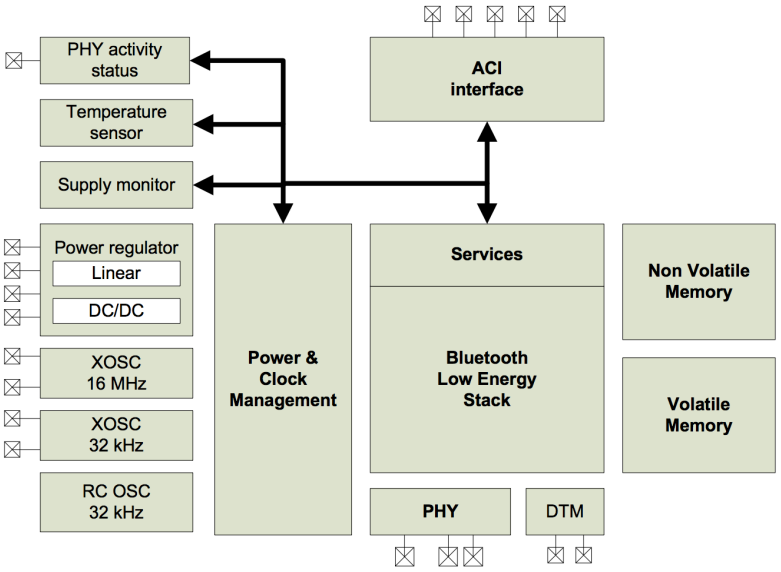


FIGURE 5.4: nRF8001 chip - Nordic Semiconductor's implementation of the LE stack. Its main physical features are the Bluetooth low energy PHY and stack that handles the Link Controller and Host[3]

Hardware Prototype

Figure 5.5 shows the configuration of the customised Bluetooth dual mode device. This device combines all three previously mentioned pieces of hardware to demonstrate a device that is capable of concurrent BR/EDR and LE transmission.

5.1.2 Software

In order to provide control over the sensors and to apply customised GATT profiles, each Arduino runs a customised C++ sketch. Nordic Semiconductor provide an SDK for the Nordic nRF8001 chip which allows for full control over the chip's Bluetooth stack. They also provide a Windows visual editor application nRFgo Studio used for creating customised GATT profiles for the nRF8001, thus allowing one of the standardised GATT models defined in Section 4.3.1 to be applied to each sensor.

nRF8001 SDK

As previously mentioned, interfacing with the Nordic nRF8001 relies on an abstraction called ACI. The default way of providing this is by polling for updates

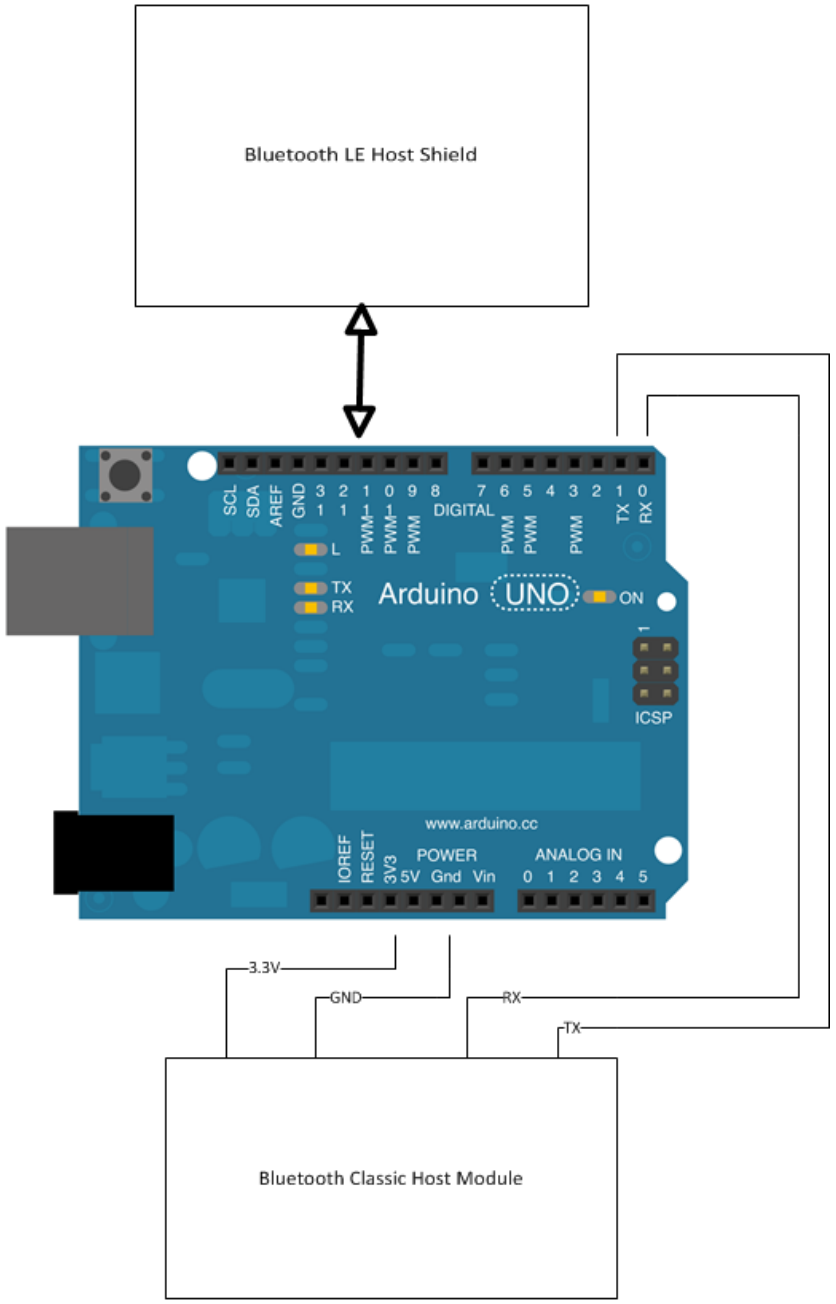


FIGURE 5.5: Bluetooth Smart Ready prototype - Connect Blue OBS421 + Red Bear Lab Shield + Arduino Uno (connected via Hardware serial)

over the ACI communication channel. An aciloop function is called usually once per cycle in the Arduino sketch's loop. This function dispatches any queued messages to the shield and polls for any messages from the shield. Inside the aciloop function the ACI Event's code is used to determine if that message indicates a change in status, receipt of data, an error, etc. See Figure 5.6 for a full overview of ACI.

In order to create a GATT profile, nRFgo Studio creates a services.h file with

a GATT configuration once designed from the UI. This file defines a number of named *pipes*, each of which are uni-directional. A single characteristic (read, write, write with response, notify, etc.) has individual pipes for each mode. These pipe identifiers are used to determine for which characteristic data was received, or through which characteristic to transmit data to the client. When a client **writes** to the BLE shield or the BLE shield **notifys** the client, the data sent/received is ephemeral and the shield acts as an intermediary. However, characteristics that can be **read** maintain that value in the memory of the BLE shield and requests for that value are delivered directly from the shield without involving the application controller (the Arduino).

Here is an example of a heart rate sensor GATT profile generated by the application:

```
<Profile Version="1.3">
    <SetupId>1</SetupId>
    <Device>nRF8001_Dx</Device>
    ...
    <Service Type="local" PrimaryService="true">
        <Name>Heart Rate</Name>
        <Uuid>180e</Uuid>
        <Characteristic>
            <Name>Heart Rate Measurement</Name>
            <Uuid>2a37</Uuid>
            <DefaultValue></DefaultValue>
            <UsePresentationFormat>0</UsePresentationFormat>
            <MaxDataLength>19</MaxDataLength>
            <AttributeLenType>2</AttributeLenType>
            <ForceOpen>false</ForceOpen>
            <Properties>
                <WriteWithoutResponse>false</WriteWithoutResponse>
                <Write>false</Write>
                <Notify>true</Notify>
                <Indicate>false</Indicate>
                <Broadcast>false</Broadcast>
            </Properties>
            <SetPipe>false</SetPipe>
```

```
            <AckIsAuto>true</AckIsAuto>
            <PeriodForProperties/>
        </Characteristic>
        <Characteristic>
            <Name>HighBitRateEnabled</Name>
            <Uuid>2a3a</Uuid>
            <DefaultValue>0</DefaultValue>
            <UsePresentationFormat>0</UsePresentationFormat>
            <MaxDataLength>1</MaxDataLength>
            <AttributeLenType>1</AttributeLenType>
            <ForceOpen>false</ForceOpen>
            <Properties>
                <WriteWithoutResponse>true</WriteWithoutResponse>
                <Write>false</Write>
                <Notify>false</Notify>
                <Indicate>false</Indicate>
                <Broadcast>false</Broadcast>
            </Properties>
            <SetPipe>false</SetPipe>
            <AckIsAuto>false</AckIsAuto>
            <PeriodForProperties/>
        </Characteristic>
    </Service>
    ...
</Profile>
```

Interaction over the ACI interface (see Figure 5.6):

- 1 - ACI commands sent from the application to the nRF8001 are placed at the tail of the ACI Command queue.
- 2 - When a command is placed in the command queue, the REQN line to the nRF8001 is pulled low.
- 3, 4 - The RDYN line interrupt handler reads ACI commands from the head of the ACI command queue and and places ACI Events that are received in the tail of the ACI event queue.

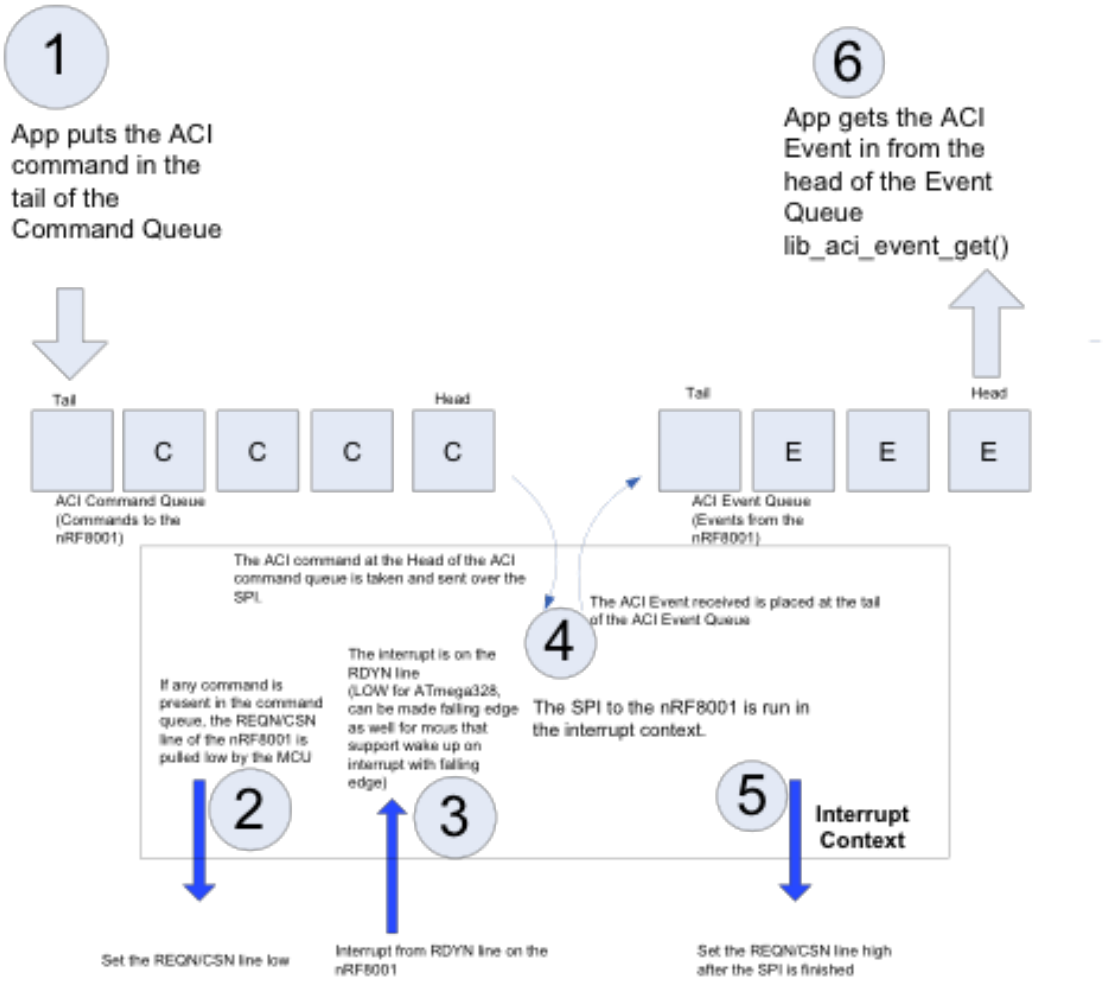


FIGURE 5.6: ACI queue interface[4]

- 5 - An interrupt is configured in the AVR to interrupt the CPU when the RDYN line is LOW. The SPI clockout, sending of an ACI command and receiving of an ACI event with the nRF8001 is done in the RDYN line interrupt.
- 6 - The ACI events are pulled out from the ACI Event queue and the application can then process each ACI Event.

5.1.3 Security

Security of the OBS421 is configured through the Connect Blue Toolbox Utility application. It offers the capability to select any BR/EDR security levels. For the implementation, the *Just Works* protocol was used. Similarly, the Red Bear Lab

[4] http://www.nordicsemi.com/eng/Products/Bluetooth-R-low-energy/nRF8001

BLE shield does not enforce any security mechanisms. As security at a WBAN level is not the focus of this research, the security mechanisms provided by Bluetooth are assumed in a real world scenario.

5.2 Gateway

The gateway provides the sensor network with a view of the wider Internet, with a responsibility of sending sensor network data to the cloud and also executing sensor commands received from the cloud. Although a smartphone would be an ideal candidate in a real life scenario, the proof of concept demonstrates the use of a Bluetooth Smart Ready enabled laptop to function as the gateway.

The Processing language combined with Java and the Bluetooth stack provides all functionality of the gateway[5]. Ubuntu was the chosen operating system[6] due to its feature-rich Bluetooth stack available - Bluez package.

5.2.1 Hardware

The Laird BT820 is a V4.0 dual mode Bluetooth USB dongle. It is based on the BT800 chipset, which in turn is based on CSR8510A10 dual mode chip. The CSR8510A10 is a single-chip radio with on-chip Low-dropout (LDO) regulators and baseband IC for Bluetooth 2.4 GHz systems including EDR up to 3 Mbps. The Bluetooth dongle works on both Windows and Linux distributions - in this case it was used on Ubuntu client 12.04.

5.2.2 Software

The Bluetooth dongle requires configuration in order to communicate over a serial port with the OBS421 and to initiate GATT commands against the Red Bear Lab shields. This is achieved through use of the Linux Bluetooth stack. This section will cover the software requirements used to construct the gateway.

[5] http://processing.org/
[6] http://www.ubuntu.com/

Processing

Processing is an integrated development environment (IDE) and open source programming language based on the Java language. It was originally intended to function solely as a way to teach programming through the use of visual feedback. However it has since become a very popular tool for prototyping especially when used in conjunction with the Arduino platform. As a result, Processing provides good serial interface support - a very simple Serial library[7]which allows for reading a writing data to and from external devices.

For each serial connection made, a serialEvent() synchronous callback function is associated with that serial port, which listens for any data that might be received. The function can be set with bufferUntil() to only trigger after a specific character is read - in this case the data packet's END character ';', allowing full packets to be received at a time.

Each serially received data packet is parsed to extract its contents - the sensor handle and its corresponding sensor reading. During the parsing stage it is also time-stamped. Ideally, time-stamping would occur at each sensor at time of capture, but this would require clock synchronisation with the gateway as the Arduino has no real-time clock. Thus this was avoided for the purposes of this implementation.

In order to buffer the high frequency BR/EDR packets, Java provides a BlockingQueue class[8] which is a thread-safe fixed size queue designed for situations of the producer-consumer model. Once a data packet is received at the gateway, the BlockingQueue's *offer* method is utilised - inserts the specified element into the queue if it is possible to do so without violating capacity restrictions, returning true upon success and false if no space is currently available. If the queue is full, the contents of the queue are emptied and prepared as a HTTP packet.

An asynchronous thread safe HTTP client is used (CloseableHttpAsyncClient - Apache Commons library [9]) to provide interaction with the cloud interface. This allows one to defer the transmission of the HTTP response back to the client in order to reduce latency incurred by HTTP communication with the cloud.

[7] http://www.processing.org/reference/libraries/serial/

[8] http://docs.oracle.com/javase/6/docs/api/java/util/concurrent/BlockingQueue.html

[9] http://hc.apache.org/httpcomponents-asyncclient-4.0.x/httpasyncclient/apidocs/org/apache/http/impl/nio/client/CloseableHttpAsyncClient.html

CloseableHttpAsyncClient provides three callback methods - completed(), failed() and cancelled(). In the event of a *failed* HTTP delivery, the transmission in reattempted. Any subsequent failures are ignored as the protection of the loss of patient data is considered as future work. All other response callbacks can be ignored.

In the implementation, BR/EDR data is only ever received from the serial interface, while all LE data is received through a Linux utility called Gatttool. As the control channel occurs solely over LE, the gateway also uses Gatttool to communicate with the sensors (see Section 5.2.2). To control Gatttool from the processing sketch, Java's ProcessBuilder class was used to initiate the process from within the Processing sketch and redirect its output into an InputStreamReader.

[Note: No concurrency is exploited within the gateway software. This is due to the single core laptop that was used to host the gateway's service, which would offer no performance gain.]

BlueZ

BlueZ is the the official Bluetooth stack for Gnu/linux. BlueZ includes a utility known as *hcitool*, which is used to configure Bluetooth connections and send special commands to Bluetooth devices. It includes another software utility known as Gatttool that allows one to connect, gather advertised data, and execute other GATT-specific LE operations. Gatttool is the primary way of controlling a LE module through GATT using Linux. There is a list of commands provided in order to interact with a LE enabled device (see Table 5.1).

As communication with BR/EDR takes place over a serial port, a connection between the dongle and the serial port is required so that data sent from the sensor network can be received by the gateway. Bluez provides a configuration file */etc/bluetooth/rfcomm.conf* that provides this functionality, which provides automatic serial binding from gateway to sensor.

```
#
# RFCOMM configuration file.
#

rfcomm0 {
    # Automatically bind the device at startup
    bind no;

    # Bluetooth address of the OBS421
    device 00:12:F3:1C:25:DC;

    # RFCOMM channel for the communication
    channel 1;

    # Description of the connection
    comment "Custom serial port with OBS421"
}
```

The command *rfcomm connect 0* will then automatically connect the Bluetooth dongle to the OBS421, which must be active and discoverable. If permissions are blocked from using the serial port, the following command will give permission to the specified user: *sudo chown username:users /dev/rfcomm0*

Command	*Description*
gatttool -b ⟨MAC Address⟩ –primary	Discover available services
gatttool -b ⟨MAC Address⟩ –characteristics	Discover characteristics
gatttool –b ⟨MAC Address⟩ –char-desc	Discover characteristic descriptors
gatttool -b ⟨MAC Address⟩ –char-read –uuid=XXXX	Read characteristic values
gatttool -b ⟨MAC Address⟩ –char-write –uuid=XXXX –value=XXXX	Write to characteristic values
gatttool -b ⟨MAC Address⟩ –char-write-req –handle=0xXXXX –value=0100 –listen	Enable notifications
gatttool -b ⟨MAC Address⟩ –char-write-req –handle=0xXXXX –value=0200 –listen	Enable indications
gatttool -b ⟨MAC Address⟩ –interactive	Enter interactive mode

Table 5.1: Gatttool commands[10]

To communicate with each sensor from the gateway, each sensor adopts one of the standardised model GATT profiles discussed in Section 4.3.1. To initiate data communication with a sensor, a GATT **notification** command is executed from the gateway. Once a notification is executed through Gatttool, the sensor response is in the following form:

"Notification handle = 0xXXXX value: XX XX"

[10] http://www.lairdtech.com/Products/Embedded-Wireless-Solutions/Bluetooth-Radio-Modules/BT800-Series/BT820/

The gateway is then responsible for acquiring each response (through InputStream-Reader as previously mentioned), and extracting the necessary information so that it can be prepared for a HTTP packet. The process of preparing the information is the same for that of received serial information.

To control operation the high and low streams, the gateway can simply access the control characteristic by executing a GATT Write No Response command through Gatttool (01 to activate and 00 to deactivate). To disable a low bit rate stream, Gatttool must issue a command to disable the current notification. However, the Red Bear Lab shield blocks any incoming connections with the following response - *connect: Device or resource busy (16)*. In order to disable the notification the device must be manually reset. Alternatively, another option is to amend the GATT profile so that LE sensor can be read by the client periodically, opposed to the server initiating the transmission through notifications.

5.2.3 Security

Two security models were discussed in the previous chapter's Section 4.6. The Trusted Server model (see Section 4.6.4) was the chosen model for the implementation.

The gateway utilises a combination of both symmetric and asymmetric cryptography in order to encrypt traffic and provide secure communication. Symmetric key cryptography is implemented using the Advanced Encryption Standard (AES) using a 256-bit key and Asymmetric key cryptography is implemented using the Rivest, Shamir and Adleman (RSA) cryptosystem using a 1024-bit key. All security mechanisms were provided through the Java Crypto library [11].

In the implementation, the gateway functions as a device intended to be used for a single patient. However, the design outlines that the gateway is transparent and can be associated with multiple patients once configured through the messaging server. This is due to restrictions imposed by the gateway's messaging client. As a result, the gateway possess a single symmetric key, which is used to encrypt all outgoing data and decrypt all incoming data.

[11] http://docs.oracle.com/javase/6/docs/technotes/guides/security/crypto/CryptoSpec.html

When the gateway initialises the following steps are taken:

- Login to the cloud application
- Login to the messaging server
- Use associated user's public key to encrypt the symmetric key. In order to acquire the user's public key, the user can transmit this information from the cloud to the gateway through the messaging server following successful account creation, which is then persistently stored. Alternatively, the gateway can request the public key from the cloud by providing a user identifier. Once the symmetric key in encrypted, the information is published in the cloud so that the user can gain access to encrypted data
- The symmetric key is stored locally on the device, encrypted with a password.

Each sensor network command received at the gateway is digitally signed and encrypted by the cloud. With access to the user's public key, the gateway can authenticate the message through the digital signature. If authenticated, the command can be fully decrypted using the gateway's symmetric key.

5.3 Cloud Application

The cloud application provides users with capabilities to view biomedical data and to control remote sensors. As the application is the only way a user can access the system, it is designed to be a rich, secure, user interface (UI)-centric, and scalable application. It is built using a combination of modern Java enterprise web technologies. To support a UI driven application, the software adheres to a modular MVC design.

5.3.1 Software

Java

Java was chosen as it was the programming language the author was most adept in. It was also due to the large number of comprehensive frameworks available

that provide a lot of required functionality to build large scale web applications. The Java version used was Java 1.6.[12]

Spring

Spring is an open source Java framework and inversion of control (IoC) container. Spring is designed as a layered architecture allowing developers to use specific modules that they require. The modules used in the prototype system are as follows:

- Spring MVC
- Spring Security
- Spring IoC
- Spring Data Access

Spring MVC

Spring has its own MVC module, which acts as the web component of Spring's framework. See Section 4.5 of the previous chapter, which details the pattern design.

Spring MVC is designed around a central request-driven servlet (DispatcherServlet) that is responsible for delegating control to the various interfaces during the execution phases of a HTTP request. The DispatcherServlet is completely integrated with Spring IoC container and is exposed to all features of the Spring framework.

Spring MVC offers a number of advantages such as:

- Clear separation of roles
- Reduces amount of code
- Customisable handler mapping

Here is an example of the capabilities of Spring MVC by demonstrating the process of user creation:

[12] http://docs.oracle.com/javase/6/docs/api/

```
<!-- View Layer -->
<form:form commandName="user">
    <table>
        <tr>
            <td>Username:</td>
            <td><form:input path="username" /></td>
        </tr>
        <tr>
            <td>Password:</td>
            <td><form:input path="password" /></td>
        </tr>
        <tr>
            <td><input type="submit" value="Submit" /></td>
        </tr>
    </table>
</form:form>

// Controller method found @ /signup through HTTP POST
// ModelAndView is an object that holds the model objects
// as well as the view to be rendered
@RequestMapping(value="/signup", method = RequestMethod.POST)
public ModelAndView signup(ModelAndView model, UserDetails userDetails) {

    User user = null;
    try {
        user = userService.createUser(userDetails.getUsername(),
        userDetails.getPassword());
    } catch (Exception e) {
        // Check to see if the email address is already used.
        result.rejectValue("username", "Email address already registered");
    }

    if (user == null || result.hasErrors()) {
        return signup(model, userDetails);
    }

    // Automatically log them in.
    userService.login(user, userDetails.getPassword());

    return new ModelAndView("redirect:dashboard/index");
}

//Service layer
@Override
public User createUser(String username, String password) {

    User user = new User();
    user.setUsername(username);
    user.setPassword(encodePassword(password));

    entityService.save(user, patient);

    return user;
}

// Model including ORM annotations
@Entity
@Table(name = "users")
public class User  {

    @NotNull
    @Column(nullable = false, unique = true)
    private String username;

    @NotNull
    @Column(nullable = false)
    private String password;
```

```
        public User(String username, String password) {
            this.username = username;
            this.password = password;
        }
    }
```

Spring Security

Spring Security provides a highly adaptable authentication and access control framework. All user access functionality is provided by the Spring Security package. It facilitates a multi permission system, which is used to provide custom application multiple profiles - primary carer, secondary carer, medical professionals, etc. Each of which offers granular control over application accessibility. For example,

```
<http auto-config="true">
        <intercept-url pattern="/admin*" access="ADMIN" />
        <intercept-url pattern="/sensorControl*" access="PRIMARY_CARER" />
        <intercept-url pattern="/patientData*" access="PRIMARY_CARER, NURSE" />
</http>
```

Spring security also manages secure access through system login/logout, through the user session (Figure 5.7 illustrates how Spring handles a user authentication request). Spring also provides automatic hashing and salt of passwords upon user login.

Spring IoC

IoC or dependency injection is a term which refers to when a developer is not required to instantiate objects manually in the code. Instead, only a description of how they should be created is required. Furthermore, components and services are not manually connected by the programmer, instead an IoC container is responsible for tying everything together. There are three different types of dependency injection - constructor, setter and interface, where dependencies can be marked through annotations. Note below how *MessageService* is never instantiated - demonstrating the use of a service layer object operating in a controller.

```
@Controller
@RequestMapping("/messages")
public class MessagesController {

    @Autowired
    private MessageService messageService;

    @RequestMapping(value = "/send", method = RequestMethod.POST)
    public void send(@RequestBody String message) {
        messageService.sendMessage(message);
    }
}
```

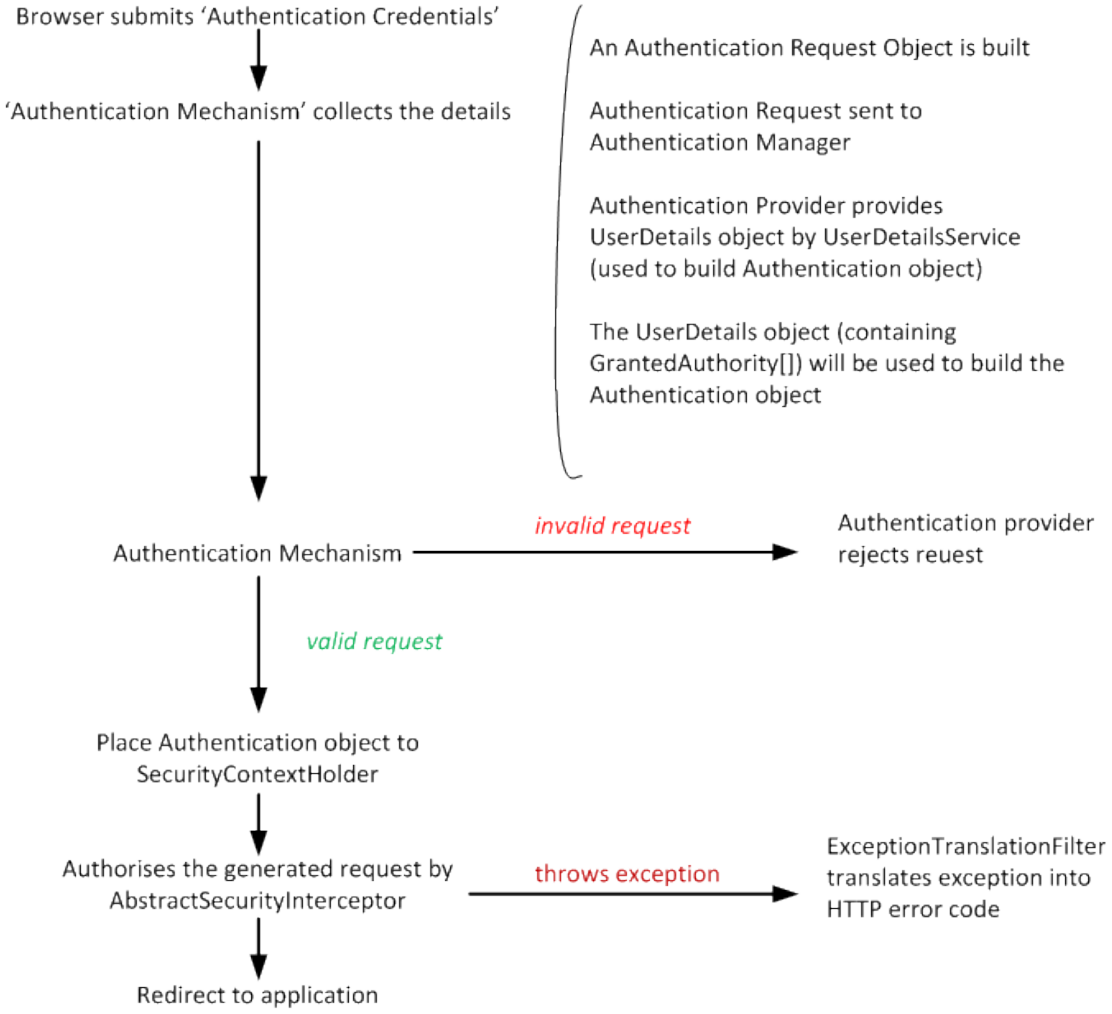


FIGURE 5.7: Spring security authentication[13]

The use of dependency injections provides several benefits such as a reduction in the amount of code, the facilitation of lazy loading, and the promotion of loose coupling.

Spring Data Access

Spring Data Access provides a framework to access databases by decoupling code from the access mechanisms. The sole purpose of Spring's Data Access framework is to provide ease of access to different kinds of persistence stores, permitting interoperability between both relational database systems and NoSQL data stores. The framework is used heavily in conjunction with Spring's IoC container to reduce the software's complexity.

[13] http://asahu05.wordpress.com/category/adobe-flex-tech/

At a persistence level, Spring provides good integration support with Object Relational Mapping (ORM) framework Hibernate. Hibernate forms the basis for all application data storage.

Hibernate

Hibernate is an Object Relational Mapping (ORM) framework for the Java language. Hibernate provides a mapping for an object domain to a traditional relational database model IE mapping Java classes to database tables, and Java data types to SQL data types. Hibernate also provides query and retrieval capabilities. MySQL is the technology used for the database. MySQL is a an open-source Relational Database Management System that uses Structured Query Language (SQL).

Here is an example of how Hibernate can map Java classes to database tables:

```
@Entity
@Table(name = "sensors")
public class Sensor {

        @NotNull
        @Column(nullable = false)
        private String name;

        @NotNull
        @Column(nullable = false)
        private Boolean active;

        @NotNull
        @Column(nullable = false)
        private Boolean smartReady;

        @NotNull
        @Column(nullable = false, unique = true)
        private String handle;

        @OneToMany(mappedBy = "sensor")
        private List<SensorData> sensorData;

        ...
}
```

ORM frameworks are useful as they can provide many benefits for application developers:

- Productivity - relieves developers from writing a lot of SQL so that focus can remain on business logic
- Maintainability - reduces the amount of code

- Portability - portable to all supported SQL databases

Application Database Model:

Figure 5.8 outlines the database model of the application, which helps to provide an overview of the entity relationships within the system.

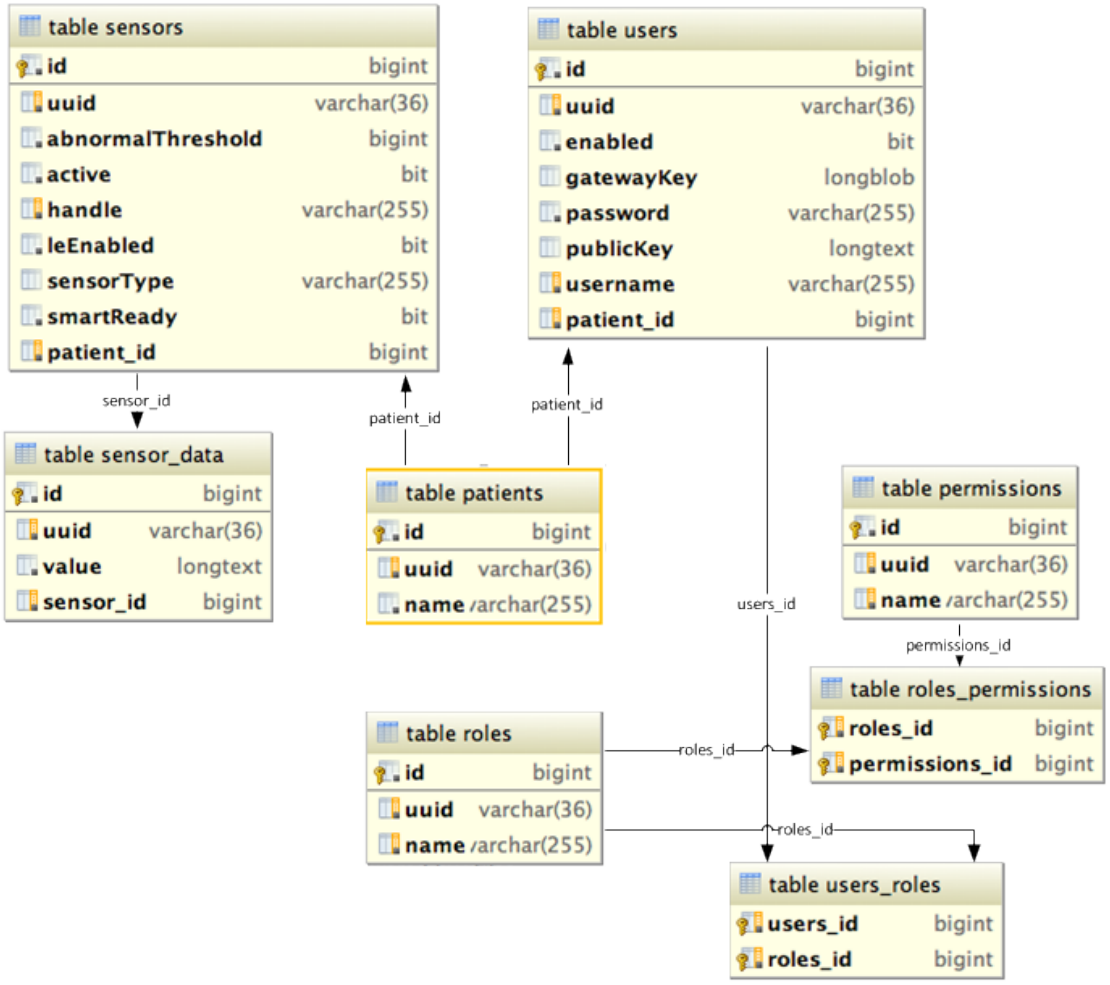


FIGURE 5.8: Database table relationships and dependencies

User - represents a system user. A user can have different roles and permissions which detail application access capabilities. These are defined by a carer, a family member, or a medical professional.

Patient - represents a single patient. A single patient can be associated with multiple users, where users (primary carer, extended family, an oncology team, etc.) can share access to the same patient. However a user is associated with a single patient. This is a current implementation restriction, which results in some limitations. For instance, a palliative care nurse's account is

only associated with a single patient instance. This means that for every of the nurse's patients, she would have to manage multiple accounts. Ideally a local palliative care nurse (user) could access the electronic records of multiple of his/her patients.

Sensor - represents an individual sensor. Its handle is used to identify sensors within the sensor network. Other fields capture the current operation of the sensor IE whether its active/inactive, the current transmission stream its using, etc. A sensor object can have one or more SensorData objects.

SensorData - represents a data capture from a sensor. Each SensorData is tied to a single Sensor object.

XMPP

Extensible Messaging and Presence Protocol (XMPP) is a real-time message protocol, based on Extensible Markup Language (XML). It was originally developed to provide a decentralised real-time instant messenger (IM). The protocol has since been used for many different scenarios such as file transfer, Internet of Things, and social networks.

XMPP is used as the message protocol to provide all communication capabilities from the cloud to gateway. As a open and decentralised technology, there is no central master XMPP server, therefore anyone can run their own XMPP server instance. This allows a server to be hosted locally, isolated from the public network, thus protecting it from network attacks.

An XMPP server instance is hosted through the Openfire library[14]. Openfire provides an admin console which allows the user to control the operation of the XMPP instance. It is through the console that user profiles are created for the gateway and a user (using the same credentials used for the cloud application). Spark is a XMPP IM service[15] that was used to connect the two profiles so that communication could occur.

Smack[16] is a Java XMPP client that is used to provide communication capabilities from the cloud application to the XMPP server. The XMPP client instance for the

[14] http://www.igniterealtime.org/projects/openfire/
[15] http://www.igniterealtime.org/projects/spark/
[16] http://www.igniterealtime.org/projects/smack/

gateway is provided by jabberlib[17]. When both user and the gateway initialises, they connect to the messaging server and initiate a chat. Due to limitations of the jabberlib library implementations, only one chat can be active at any given time. This means that the gateway can only have a single active communication link with a cloud application user.

5.3.2 Security

Security for the cloud application is tied to the security provided at the gateway level. As previously mentioned whilst discussing gateway security in Section 5.2.3, the chosen security model is the Trusted Server (see Section 4.6.4). Similar to the gateway, the Java Crypto library [18] is used to provide all security functionality on the cloud and AES and RSA are the algorithms used to provide symmetric and asymmetric cryptography functionality.

When sending control messages to the sensor network via the gateway, XMPP provides multiple levels of security:

- Concept of a contact list, which is similar to concept of pairing in Bluetooth. This requires two parties to have identified each other as future contacts
- All communication with the cloud is digitally signed and encrypted
- The connection occurs over SSL

As previously mentioned in Section 5.3.1, Spring Security modules provides all required functionality for providing secure access to the user application.

Due to the security implementation of the system, a few restrictions are imposed. For instance, can no longer provide the support for alerts. This would require *unencrypted* patient data to be received at the cloud so that threshold levels could be determined in real time. Alternatively, the responsibility could be placed with the gateway. However, this adds more complexity to the device, weakening its design strengths.

[17] http://go.yuri.at/jabberlib/

[18] http://docs.oracle.com/javase/6/docs/technotes/guides/security/crypto/CryptoSpec.html

5.4 Summary

This chapter has demonstrated the implementation of a system outlined in Chapter 4.

This chapter introduced a prototype Bluetooth Smart Ready sensor that is capable of transferring ECG and heart rate information to another Bluetooth Smart Ready device acting as an Internet gateway over two separate data rate channels in parallel. The chapter also discusses the implementation of a cloud application that demonstrates a system of access and control over patient care by using a real time message protocol to transmit control information from the UI to the sensor network.

A gateway can provide encryption of information to the cloud, and authentication of cloud instructions was also detailed, demonstrating secure communication and key exchange between with the gateway and cloud through a Trusted Server model.

The next chapter provides an evaluation of the system implementation and hardware prototype through multiple experiments.

Chapter 6

Evaluation

This chapter is dedicated to providing an evaluation of the system implementation. This chapter will cover a number of experiments aimed to provide a performance evaluation in various sections of the system. To conclude the chapter, some additional metrics are discussed.

There are number of interesting performance metrics that can be used to evaluate a software system:

Throughput - In the context of this dissertation, throughput refers to the rate of successful message delivery, which is usually measured in bits per second or packets per second. Throughput in an important consideration as it determines the level of detail the system can provide in a given time, for instance, the rate at which the prototype sensor can transmit information to the cloud.

Responsiveness - Refers to the ability of a system to complete a particular task within a given time. In a system designed to adapt transmission according to live patient information and professional intervention, the responsiveness of the system is an important measurement.

It is important to note the proof of concept's hardware limitations. The OBS421 (BR/EDR module) that facilitated the high bit rate channel offers a maximum of 1.3 Mbps throughput. This is a long way off the Bluetooth 4.0 specification of 24 Mbit/s that was introduced in Bluetooth V3.0 + HS. Conversely, the Laird

BT820 (Bluetooth Smart Ready dongle representing the gateway) has a maximum capability of 3 Mbps.

It is assumed for each experiment that:
Packet size - 10-11 bytes
Distance between Bluetooth endpoints - 1m (worn on body)

6.1 Experiments

Experiment 1
Aim: Determine the responsiveness of the high bit rate channel by measuring the round trip time (RTT) between enabling high bit rate channel through the UI until the first data packet is received and saved in the database.
Methodology: When the command request is received by the application, a timestamp is generated and stored. The request then continues uninterrupted. When the first data packet is received by the cloud, a second timestamp is generated and stored. By subtracting the initial timestamp from the second, the result indicates the difference between the initial request and the first data packet receipt time. This process was repeated and the difference summed for each iteration. Finally, an average was taken according to the resulting difference total and the number of iterations.

In order to provide the test automation, a Firefox plugin Selenium[1]was used to write automated browser scripts.In order to measure time, Java provides a method - currentTimeMillis() - which returns the current time in milliseconds[2].

Result:

Mean (ms)	1708.4
Sample Size (iterations)	100
Standard Deviation	438.22
95% Confidence Interval	±85.89
Gateway Buffer Size	1,024

Discussion: As a result of the experiment it was found that an average of roughly 1,708 ms was required to establish the high bit rate channel and to receive the first 1,024 data packets.

[2] http://docs.oracle.com/javase/6/docs/api/java/lang/System.html
[1] http://docs.seleniumhq.org/

Experiment 2

Aim: Determine the responsiveness of the low bit rate channel, by measuring the RTT between enabling the low bit rate channel through the UI until the first data packet is received and saved in the database.

Methodology: This experiment is very similar to Experiment 1, with the only difference being the testing of the low bit rate channel as opposed to the high bit rate channel. The test setup remained the same.

Result:

Mean (ms)	5805.43
Sample Size (iterations)	112
Standard Deviation	402.11
95% Confidence Interval	±74.47

Discussion: The experiment identified an average RTT of 5805 ms is required between initiating the low bit rate channel from the UI and saving the first data packet in the database (note that the low bit rate channel does not buffer packets at the gateway, so only one packet is ever received by the cloud at once).

Compared to the responsiveness of the high bit rate channel (1708.4 - result obtained from Experiment 1), the low bit rate channel takes 4097.03 ms more on average to complete.

Experiment 3

Aim: Determine the overhead incurred by the security mechanisms of the system, by repeating Experiment 1 but removing all security functionality.

Methodology: All security functionality was removed from both the gateway and cloud implementations. The remainder was simply a repeat of Experiment 1. To determine the overhead of security, the result was compared with the result obtained from Experiment 1

Result:

Mean (ms)	1541.43
Sample Size (iterations)	100
Standard Deviation	286.71
95% Confidence Interval	±56.19
Gateway Buffer Size	1024

Discussion: Using the result obtained this Experiment (1541.43 ms) and subtracting it from the result from Experiment 1 (1708.4 ms) - the result indicates an average performance cost incurred by security (in terms of time required to

enabled the high bit rate channel). The result was a difference of **166.97 ms**, illustrating that on average a RTT for enabling the high bit rate channel without using security will provide a speed up of 166.97 ms.

Experiment 4

Aim: To investigate the throughput capabilities of the prototype sensor's low data rate channel by measuring the inter-arrival packet time at the gateway.

Methodology: Measure the average inter-arrival time of low data rate sensor packets arriving at the gateway over a total of 100 packets.

Result: **2000ms**

Discussion: The low bit rate channel was very strict in its timing and over a sample of 100 transmissions it did not diverge. As a result, the prototype sensor's low bit rate channel is capable of a 0.5Hz sample rate - 1 packet arriving every 2 seconds at the gateway. It is important to note that this result is directly related to the specific Bluetooth stack implementation of the nRF8001 chip. It is not to be considered as a system-wide limitation. It merely indicates that for the given prototype, the current throughput performance of LE notifications is limited to 1 packet every 2 seconds. The low bit rate channel is designed for episodic data transmission.

This experiment was conducted to demonstrate the rate at which LE notifications are received at the gateway for the sensor prototype. In reality, LE is used for periodic state updates over long periods of time.

Experiment 5

Aim: To determine the transmission rate of the prototype sensor's high bit rate channel by measuring the inter-transmit time at the sensor.

Methodology: Calculate the average inter transmit time for a total of 10,000 BR/EDR packets, where inter-transmit time refers to the time between the transmission of data packets.

Result:

Mean (ms)	0.889532
Sample Size (iterations)	10,000

Discussion: Each data packet is sent from the sensor at an average rate of **0.889532 ms**. The rate of packets per second is therefore 1 / 0.889532 = 1.12418665 - one packet is transmitted by the sensor every 1.12418665 ms.

Experiment 6

Aim: To investigate the throughput capabilities of the prototype sensor's high data rate channel by measuring the inter-arrival packet time at the gateway.

Methodology: Measure the inter-arrival time of high data rate sensor packets arriving at the gateway for a total of 10,000 packets. The chosen gateway buffer size was 1,024.

Result: Over an average of 10,000 packet receipts, the inter packet arrival time at the gateway for high bit rate channel was measured at **983570 ns / 0.983570 ms**

Discussion: The system's high bit rate channel is capable 1 packet receipt every **0.983570 ms** at the gateway.

It is important to note that this figure does not represent a maximum bandwidth that can be achieved. Given an alternative sensor implementation or sufficient data buffering at a sensor level, this value could be significantly improved. Therefore this is not to be considered a generalised result for the system's limitations. This result is considered for the evaluation of the prototype sensor, which provides as a basis figure used in later experiments.

The inter-transmit time at the sensor is 0.889532 ms, and the inter-arrival time at the gateway is 0.983570. This results in a difference of 0.094038 ms.

Experiment 7

Aim: An experiment to determine the inter-response time from the cloud after submitting data from the gateway.

Methodology: Calculate the inter-response time for a total of 102,400 high data rate packets with a buffer size of 1,024 (100 requests). Inter-response time refers to the time between the cloud returns a HTTP status code to the gateway to indicate the success or failure of the request. Each data packet size is between 10 or 11 bytes. This meant that each request to the cloud was 1,024 * 10 bytes = 10 kilbytes.

Result: An average of **1034 ms** inter-response time over 100 HTTP requests with 1,024 data packets per request.

Discussion: Note that the HTTP client used in this instance is asynchronous

Experiment 8

Aim: To determine the optimal size for the gateway buffer, by measuring the inter arrival time of packets at the cloud

Methodology: As previously mentioned in Chapter 4's Section 4.4, the gateway is responsible for buffering high bit rate data to prevent a network latency induced bottleneck at the gateway. Determining the an optimal size for the gateway buffer was achieved through measuring the rate at which packets are received at the cloud for a variety of different buffer sizes.

According to the result of Experiment 5, each data packet sent over the high bite rate channel from the sensor is transmitted at an average rate of **0.889532 ms**. The gateway was configured to buffer a number of packets before delivering them to the cloud service, as described in Section 4.4. This experiment evaluates the rate at which packets can be delivered to the cloud for buffer sizes ranging from 32 to 8,192 (incrementing in powers of two). The average buffer inter-arrival time at the cloud service was calculated.

What remained constant between different buffer size tests:

- Number of data packets - The experiment calculated the average for a total of 32,768 packets
- Packet size - (between 10 or 11 bytes)
- Rate of sensor transmission - Experiment 5 denotes a single packet is produced by the sensor every 0.897475 milliseconds.

Varying between tests:

- Buffer sizes - 32 to 8,192 (in powers of 2)

Result:

The packet receipt rate (per second) was calculated dividing by the gateway buffer size by the inter arrival time (in seconds).

Discussion: The inter-arrival time of high bit rate data was measure at 0.983570 ms (result from Experiment 6).

Using the result obtained from experiment 7 (1034 ms) and using the inter transmit time at the gateway (912 - this figure was calculated when calculating the inter arrival time at the cloud), by subtracting 912 from 1034, the result shows the

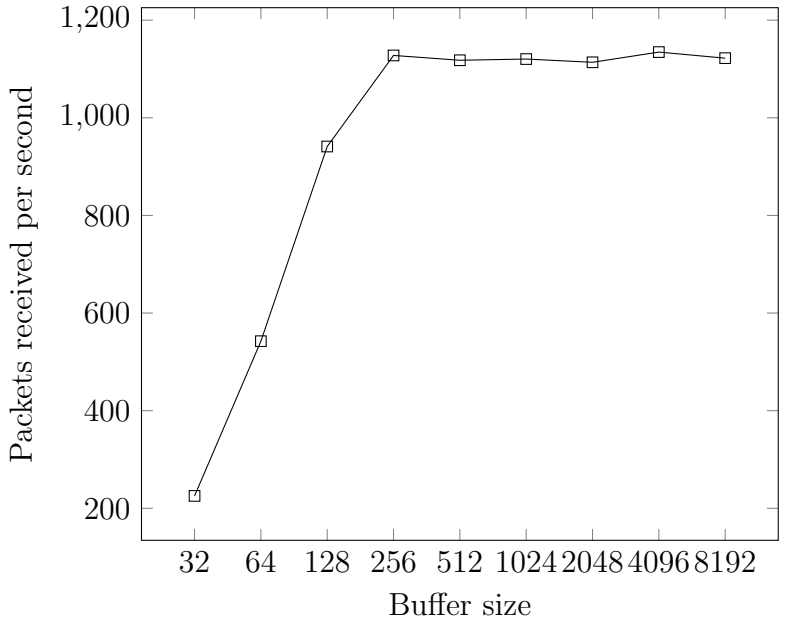


FIGURE 6.1: Line chart showing average rate of packets received per second by the cloud with the corresponding gateway buffer size

$BufferSize$	$InterArrivalTime(ms)$
32	140
64	118
128	136
256	227
512	458
1024	914
2048	1838
4096	3405
8192	7300

TABLE 6.1: Each buffer size with its associated inter arrival time

average time spent waiting for response from the cloud for a buffer size of 1,024 - 122 ms.

Comparing this to information to Table 2.1 of Chapter 2 - the prototype sensor is capable supporting throughput for complex sensors such as 12-lead ECG and EMG.

Best performance was shown using a buffer size of 4,096 results with a data receive rate of 1,134 ms. Buffer sizes less than 32 resulted in a bottleneck at the gateway, causing the software to crash.

There are a number of additional metrics that would have formed useful discussion for the project:

1. CPU usage of gateway - determining CPU usage during multiple execution stages. For instance, measure the CPU usage during high bit rate channel vs low bit rate channel. This may give an expectation of what a smartphone device will require to support the gateway's functionality

2. Bandwidth - monitor the Bluetooth network (can be provided by tools such as Wireshark [3])

3. Power consumption - measurement of OBS421 and Red Bear Lab shield's power consumption during operation VS at rest (when sending data VS when not sending data). In order to maintain a non-invasive network of sensors, a study of the power consumption is crucial.

6.2 Summary

This chapter provided an evaluation of the system by demonstrating aspects of the performance of the system.

The prototype sensor's high bit rate channel was evaluated, demonstrating its capabilities of providing data rates required for complex medical sensors. The system's responsiveness was also evaluated by measuring the time taken to enable a high and low bit rate channels through the application interface until the first packets arrive at the cloud.

The overhead incurred by the system's security mechanisms (encryption and decryption on cloud and gateway) was measured, demonstrating a slight performance impact. Finally, a range of different gateway buffer sizes was explored to determine the optimal size to provide maximum performance. The following chapter offers a conclusion to the document and discusses the potential for future work.

[3] http://www.wireshark.org/

Chapter 7

Conclusion

This chapter presents final remarks to the dissertation and concludes with a discussion of potential future work in the field.

This dissertation outlines a system to provide remote palliative care of patients using an adaptive multi-stream data transmission protocol of low and high bit-rate sensor data using Bluetooth Smart Ready technology. The system demonstrates a physiological sensor prototype that can be used in a real life scenario by palliative care patients to non-invasively monitor during episodic time intervals to provide frequent health updates, while also facilitating high quality medical information through a high bit rate stream. Using a IP enabled gateway, a patient's biomedical data can be captured and published in the cloud.

A cloud service model was designed to provide a means through which care-givers can access some or all of the sensors used by a patient and control those sensors, for example, to modify the level of detail in the information gathered from sensors. The cloud also hosts secured patient information, which forms the basis of a central repository of shared knowledge between care-givers.

Two different security models were discussed which both provide secure transmission and storage of patient data. Both mechanisms combine the strengths of symmetric and asymmetric cryptography to securely exchange secret keys between parties to gain access to relevant data. A *Host Proof* option provides a security solution that shares no secret information with the cloud server in plain text, retaining sole access to one's own personal information.

This dissertation has defined a multifunctional pervasive telemedicine system suitable for use in the context of palliative care, providing high bit rate transmission in times of patient distress or carer request and low bit rate transmission to provide periodic updates of patient data. The proof of concept demonstrates the use of an emulated ECG and heart rate combined sensor by combined two separate Bluetooth radios through the Arduino platform. Once integrated into a full system comprising an Internet gateway and Cloud application, the sensor formed the basis for a Telemedicine system for patients of palliative care. More broadly however, the system provides a framework, which forms as a basis for any application to provide information through adaptive streams.

7.1 Future Work

The implementation of the system design necessitated the creation of a sensor prototype which demonstrated the concurrent use of BR/EDR and LE protocols. There is currently no hardware on the market that provides this capability in a single solution. Focus should be placed on the invention of a dedicated module that features both Bluetooth radios controlled by a microcontroller, offering control over the module's software operation, with both radios offering feature capabilities according to the latest Bluetooth specification.

The implementation currently features a security model discussed in Section 4.6.4. Instead the system's security might be replaced by a Host Proof paradigm detailed in Section 4.6.5. Similarly, to provide an end to end security model, sensors could be investigated to provide encryption at a WBAN level. To optimise performance for greater throughput, buffering at a sensor level should also become an important focus. This work however would increase the complexity at a sensor level and may affect the power consumption levels which are required to remain low to facilitate non-invasiveness.

The proof of concept demonstrates the use of a Bluetooth Smart Ready enabled laptop through a USB dongle. To measure the capabilities of the Telemedicine system as a mobile implementation, a smartphone should replace the current role of the gateway. This would also require a more robust key management mechanism at the gateway to protect symmetric keys. Similarly, as a decentralised service,

XMPP could be evaluated to perform real time communication from the cloud to the gateway in place of HTTP.

The cloud application is a rich web application designed to provide features that offers caregivers maximum support to provide care for patients. The current implementation does not support multi patient/user association. Enabling this would provide a more rich interaction between medical staff and their patients. To further improve the cloud service, support should be made for implementations which use multiple gateways.

Barriers to entry are typically very high in the healthcare sector due to regulations and laws regarding the safety, quality and efficacy of medical products. A qualitative system evaluation involving real palliative care nurses and patients to reveal underlining system qualities and characteristics would be an important step in bringing the solution closer to market.

Bibliography

[1] B. Latré, B. Braem, I. Moerman, C. Blondia, and P. Demeester, "A Survey on Wireless Body Area Networks," *Wirel. Netw.*, vol. 17, no. 1, pp. 1–18, Jan. 2011.

[2] N. Gupta, *Inside Bluetooth Low Energy*, ser. Artech House Remote Sensing Library. Artech House, Incorporated, 2013.

[3] M. Patel and J. Wang, "Applications, challenges, and prospective in emerging body area networking technologies," *Wireless Communications, IEEE*, vol. 17, no. 1, pp. 80–88, February 2010.

[4] J. Dunning, "Taming the Blue Beast: A Survey of Bluetooth Based Threats," *Security Privacy, IEEE*, vol. 8, no. 2, pp. 20–27, March 2010.

[5] J. Wan, C. Zou, S. Ullah, C.-F. Lai, M. Zhou, and X. Wang, "Cloud-enabled wireless body area networks for pervasive healthcare." *IEEE Network*, vol. 27, no. 5, pp. 56–61, 2013.

[6] R. Currell, C. Urquhart, P. Wainwright, and R. Lewis, "Telemedicine versus face to face patient care: effects on professional practice and health care outcomes," vol. 97, no. 35, pp. 35+, Aug. 2001.

[7] C. Sepúlveda, A. Marlin, T. Yoshida, and A. Ullrich, "Palliative Care: the World Health Organization's global perspective." *J Pain Symptom Manage*, vol. 24, no. 2, pp. 91–6, 2002.

[8] S. Park and S. Jayaraman, "Enhancing the quality of life through wearable technology," *Engineering in Medicine and Biology Magazine, IEEE*, vol. 22, no. 3, pp. 41–48, May 2003.

[9] C. Cheng, T. Stokes, and M. Wang, "caREMOTE: The design of a cancer reporting and monitoring telemedicine system for domestic care," in *Engineering in Medicine and Biology Society,EMBC, 2011 Annual International Conference of the IEEE*, Aug 2011, pp. 3168–3171.

[10] C. Doukas, I. Maglogiannis, and G. Kormentzas, "Advanced Telemedicine Services through Context-aware Medical Networks," 2006.

[11] P. Matlani and N. Londhe, "A cloud computing based telemedicine service, year=2013," in *Point-of-Care Healthcare Technologies (PHT), 2013 IEEE*, Jan, pp. 326–330.

[12] B. C. Meyer, R. Raman, and T. Hemmen, "Efficacy of site-independent telemedicine in the STRokE DOC trial: a randomised, blinded, prospective study," pp. 787–795, 2008.

[13] V. N. Dhruva, S. I. Abdelhadi, A. Anis, W. Gluckman, D. Hom, W. Dougan, E. Kaluski, B. Haider, and M. Klapholz, "ST-Segment Analysis Using Wireless Technology in Acute Myocardial Infarction (STAT-MI) Trial," *Journal of the American College of Cardiology*, vol. 50, no. 6, pp. 509 – 513, 2007.

[14] S. Bergrath, A. Reich, R. Rossaint, D. Rörtgen, J. Gerber, H. Fischermann, S. K. Beckers, J. C. Brokmann, J. B. Schulz, C. Leber, C. Fitzner, and M. Skorning, "Feasibility of Prehospital Teleconsultation in Acute Stroke – A Pilot Study in Clinical Routine," *PLoS ONE*, vol. 7, no. 5, p. e36796, 05 2012.

[15] A. G. Ekeland, A. Bowes, and S. Flottorp, "Effectiveness of telemedicine: A systematic review of reviews." *I. J. Medical Informatics*, vol. 79, no. 11, pp. 736–771, 2010.

[16] ——, "Methodologies for assessing telemedicine: A systematic review of reviews." *I. J. Medical Informatics*, vol. 81, no. 1, pp. 1–11, 2012.

[17] M. Rofouei, M. Sinclair, R. Bittner, T. Blank, N. Saw, G. DeJean, and J. Heffron, "A Non-invasive Wearable Neck-Cuff System for Real-Time Sleep Monitoring," in *Proceedings of the 2011 International Conference on Body Sensor Networks*, ser. BSN '11, 2011, pp. 156–161.

[18] D. Rissacher, R. Dowman, and S. A. C. Schuckers, "Identifying frequency-domain features for an EEG-based pain measurement system," in *Bioengineering Conference, 2007. NEBC '07. IEEE 33rd Annual Northeast*, March 2007, pp. 114–115.

[19] J. Bakker, M. Pechenizkiy, and N. Sidorova, "What's Your Current Stress Level? Detection of Stress Patterns from GSR Sensor Data," in *Data Mining Workshops (ICDMW), 2011 IEEE 11th International Conference on*, Dec 2011, pp. 573–580.

[20] C. Otto, E. Jovanov, and A. Milenkovic, "A WBAN-based System for Health Monitoring at Home," in *Medical Devices and Biosensors, 2006. 3rd IEEE/EMBS International Summer School on*, Sept 2006, pp. 20–23.

[21] N. Hunn, *Essentials of Short-Range Wireless*. Cambridge University Press, 2010.

[22] ABI Research, "Bluetooth Smart Will Drive Cumulative Bluetooth Enabled Device Shipments to 20 billion by 2017," https://www.abiresearch.com/press/bluetooth-smart-will-drive-cumulative-bluetooth-en, 2012, [Online; accessed 21-May-2014].

[23] A. J. Jara, D. Fernandez, P. Lopez, M. A. Zamora, A. F. Skarmeta, and L. Marin, "Evaluation of Bluetooth Low Energy Capabilities for Tele-mobile Monitoring in Home-care," *j-jucs*, vol. 19, no. 9, pp. 1219–1241, May 2013.

[24] B. Yu, L. Xu, and Y. Li, "Bluetooth Low Energy (BLE) based mobile electrocardiogram monitoring system," in *Information and Automation (ICIA), 2012 International Conference on*, June 2012, pp. 763–767.

[25] A. Depari, A. Flammini, S. Rinaldi, and A. Vezzoli, "Multi-sensor system with Bluetooth connectivity for non-invasive measurements of human body physical parameters," *Sensors and Actuators A: Physical*, vol. 202, no. 0, pp. 147 – 154, 2013, selected Papers from the 26th European Conference on Solid-State Transducers Kraków, Poland, 9-12 September 2012.

[26] C. Gomez, J. Oller, and J. Paradells, "Overview and Evaluation of Bluetooth Low Energy: An Emerging Low-Power Wireless Technology," *Sensors*, vol. 12, no. 9, pp. 11 734–11 753, 2012.

[27] A. Molina-Markham, R. A. Peterson, J. Skinner, R. J. Halter, J. Sorber, and D. Kotz, "Poster: Enabling Computational Jewelry for mHealth Applications," in *Proceedings of the International Conference on Mobile Systems, Applications, and Services (MobiSys)*. ACM Press, June 2014.

[28] A. Dementyev, S. Hodges, S. Taylor, and J. Smith, "Power Consumption Analysis of Bluetooth Low Energy, ZigBee and ANT Sensor Nodes in a Cyclic Sleep Scenario," in *EEE International Wireless Symposium*, 2013.

[29] E. Cano and I. Garcia, "Design and Development of a BlueBee Gateway for Bluetooth and ZigBee Wireless Protocols," in *Electronics, Robotics and Automotive Mechanics Conference (CERMA), 2011 IEEE*, Nov 2011, pp. 366–370.

[30] P. M. Østhus, "Concurrent operation of Bluetooth low energy and ANT wireless protocols with an embedded controller," Master's thesis, Norwegian University of Science and Technology, Faculty of Information Technology, Mathematics and Electrical Engineering, Department of Electronics and Telecommunications, June 2011.

[31] J. Park, M. Jo, D. Seong, and J. Yoo, "Disjointed Multipath Routing for Real-Time Data in Wireless Multimedia Sensor Networks," *International Journal of Distributed Sensor Networks*, p. 8, 2014.

[32] M. J. Morón, R. Luque, and E. Casilari, "On the Capability of Smartphones to Perform as Communication Gateways in Medical Wireless Personal Area Networks," *Sensors*, vol. 14, no. 1, pp. 575–594, 2014.

[33] G. Mulligan, "The 6LoWPAN Architecture," in *Proceedings of the 4th Workshop on Embedded Networked Sensors*, ser. EmNets '07, 2007, pp. 78–82.

[34] 6LoWPAN, "Transmission of IPv6 Packets over Bluetooth Low Energy," http://tools.ietf.org/html/draft-ietf-6lowpan-btle-12, 2013, [Online; accessed 21-May-2014].

[35] A. J. Jara, M. A. Zamora, and A. F. Gómez-Skarmeta, "Glowbal IP: An adaptive and transparent IPv6 integration in the Internet of Things," *Mobile Information Systems*, vol. 8, no. 3, pp. 177–197, 2012.

[36] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the

Clouds: A Berkeley View of Cloud Computing," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28, Feb 2009.

[37] S. Mohammed, D. Servos, and J. Fiaidhi, "HCX: A Distributed OSGi Based Web Interaction System for Sharing Health Records in the Cloud," in *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2010 IEEE/WIC/ACM International Conference on*, vol. 3, Aug 2010, pp. 102–107.

[38] M.-H. A. Kuo, "Opportunities and Challenges of Cloud Computing to Improve Health Care Services," *J Med Internet Res*, vol. 13, no. 3, p. e67, Sep 2011.

[39] C. Doukas, I. Maglogiannis, and T. Pliakas, "Advanced Medical Video Services through Context-Aware Medical Networks," in *Engineering in Medicine and Biology Society, 2007. EMBS 2007. 29th Annual International Conference of the IEEE*, Aug 2007, pp. 3074–3077.

[40] K. Malasri and L. Wang, "Addressing Security in Medical Sensor Networks," in *Proceedings of the 1st ACM SIGMOBILE International Workshop on Systems and Networking Support for Healthcare and Assisted Living Environments*, ser. HealthNet '07. New York, NY, USA: ACM, 2007, pp. 7–12.

[41] M. Mana, M. Feham, and B. A. Bensaber, "SEKEBAN (Secure and Efficient Key Exchange for wireless Body Area Network)," *International Journal of Advanced Science and Technology*, 2009.

[42] C. C. Y. Poon, Y.-T. Zhang, and S.-D. Bao, "A novel biometrics method to secure wireless body area sensor networks for telemedicine and m-health," *Communications Magazine, IEEE*, vol. 44, no. 4, pp. 73–81, April 2006.

[43] S. Sandhya and K. Devi, "Analysis of Bluetooth threats and v4.0 security features," in *Computing, Communication and Applications (ICCCA), 2012 International Conference on*, Feb 2012, pp. 1–4.

[44] R. Nasim, "Security Threats Analysis in Bluetooth-Enabled Mobile Devices," *CoRR*, vol. abs/1206.1482, 2012.

[45] M. Nkosi, F. Mekuria, and S. Gejibo, "Challenges in mobile bio-sensor based mHealth development," in *e-Health Networking Applications and Services*

(Healthcom), 2011 13th IEEE International Conference on, June 2011, pp. 21–27.

[46] S. Shini, T. Thomas, and K. Chithraranjan, "Cloud Based Medical Image Exchange-Security Challenges," *Procedia Engineering*, vol. 38, no. 0, pp. 3454 – 3461, 2012.

[47] K. Bhargavan and A. Delignat-Lavaud, "Web-based attacks on host-proof encrypted storage," in *6th USENIX Workshop on Offensive Technologies*. Usenix, aug 2012, pp. 97–104.

[48] R. Heydon, *Bluetooth Low Energy: The Developer's Handbook*, ser. Pearson Always Learning. Prentice Hall PTR, 2012.

[49] Bluetooth SIG, "Bluetooth 4.0 specification," https://www.bluetooth.org/en-us/specification/adopted-specifications, 2014, [Online; accessed 21-May-2014].